SW01

Strategia Wolnych
i Otwartych Implementacji

TOM 2

Program

nauczania-uczenia się infotechniki

Człowiek – najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

ORE Ośrodek Rozwoju Edukacji

FWIOO

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Program

nauczania-uczenia się
infotechniki

Seria wydawnicza
Strategia Wolnych i Otwartych Implementacji
Tom II

Program

nauczania-uczenia się infotechniki

pod redakcją
Stanisława Ubermanowicza i Krzysztofa Wawrzyniaka

Fundacja Wolnego i Otwartego Oprogramowania
Poznań, 2014

Redakcja
Stanisław Ubermanowicz, Krzysztof Wawrzyniak

Projekt graficzny i skład
Krzysztof Marciniak / marciniakkrzych.blogspot.com

Korekta
Emanuela Tatarkiewicz

Wydawca
Fundacja Wolnego i Otwartego Oprogramowania, ul. Staszica 25/8, 60-524 Poznań

ISBN: 978-83-934691-6-1
ISBN: 978-83-934691-7-8 (tom I)
ISBN: 978-83-934691-8-5 (tom II)
ISBN: 978-83-934691-9-2 (PDF)

Materiały dołączone na płycie stanowią integralną obudowę książki.

Materiał bezpłatny - współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego, wydany w ramach Projektu „Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych”.

Spis treści

Wstęp

Tradycyjny program nauczania, realizowany w ramach formalnej edukacji szkolnej w systemie klasowo-lekcyjnym, stanowi propozycję treści i metod realizacji, przy założonych celach i oczekiwanych efektach kształcenia. Taki program musi wypełniać zapisy Podstawy programowej konkretnego przedmiotu i etapu edukacyjnego, pozostawiając niezbyt wiele swobody nauczycielom co do własnej inicjatywy w zakresie szczegółowych tematów. Jeśli nauczyciel będzie uważał za istotne, aby uwzględniać oczekiwania i zainteresowania uczniów, może stanąć przed dylematem: – Czy tworzyć i realizować program oparty na Podstawie programowej, czy jednak bardziej dostosować go do preferencji uczniów i własnych? W rzeczywistości polskiej szkoły wielu nauczycieli wybiera wariant pierwszy. Taki stan rzeczy wpływa negatywnie zarówno na zaangażowanie w proces dydaktyczny i satysfakcję samego nauczyciela, jak też i uczniów, którzy coraz częściej wybierają alternatywne, pozaszkolne źródła wiedzy.

W czasach olbrzymiej roli mediów i nowych technologii informacyjno-komunikacyjnych (TIK) w kształceniu nie da się oddzielić efektów edukacji szkolnej od pozaszkolnej. Dzięki sieci i technologiom mobilnym proces uczenia się jest wszechobecny (*ubiquitous learning*). Rodzi się silna potrzeba tworzenia programów i podstaw metodycznych do efektywnego korzystania z mieszanych form edukacyjnych (*blended learning*), których istotą jest połączenie procesu nauczania stacjonarnego i uczenia się zdalnego. Innowacyjne metody i zróżnicowane ścieżki kształcenia z wykorzystaniem nowych technologii objęte są priorytetem w programach unijnych. Świadczy to o olbrzymiej roli wypracowania skutecznych strategii wdrażania komplementarnych form kształcenia bezpośredniego i pośredniego poprzez Internet. Takim właśnie projektem jest „Strategia Wolnych i Otwartych Implementacji" (SWOI), której istotnym elementem jest prezentowany tu Program nauczania-uczenia się infotechniki, wykorzystujący dedykowane narzędzia: Serwis edukacyjno-społecznościowy e-Swoi i Szkolny Remiks Ubuntu (system i oprogramowanie).

Kompleksowa Strategia edukacyjna i Program nauczania-uczenia się – poprzez mieszane formy edukacji, atrakcyjne treści programowe, innowacyjne metody oddziaływań, wzorcowe narzędzia pracy trenerów, wolne i otwarte środowiska systemowe oraz narzędzia pracy ucznia – animuje i moderuje następujące **założenia taktyczne i realizacyjne**:

- samodzielne lub zespołowe tworzenie implementacji, tj. projektowanie, programowanie lub konstruowanie jako forma wyrównywania szans i aktywizacji poprzez wykonywanie cząstko-

wych zadań na miarę osobistych zdolności, jako alternatywa dla szkolnego nauczania głównie w zakresie posługiwania się technologiami informacyjno-komunikacyjnymi;

- formowanie świadomości i pozytywnych postaw uczniów wobec wartości takich, jak: twórczość, aktywizacja i partycypacja, respekt i partnerstwo, wolność i otwartość, dialog i negocjacja – ważnych w wychowaniu i socjalizacji oraz niezbędnych do budowy filarów pod indywidualne lub zespołowe tworzenie implementacji;
- harmonizowanie oddziaływań na sfery poznawcze, emocjonalne i psychomotoryczne, poprzez dobór właściwych proporcji pomiędzy zakresem i trudnością materiału stanowiącego treść kształcenia a formami ćwiczeniowymi i emocjonalno-motywacyjnymi, w tym także zabawowymi (np. tematyka gier logicznych czy graffiti).
- wielotorowy, międzypokoleniowy i międzyrówieśniczy transfer wiedzy i umiejętności, zarówno w formach stacjonarnych, jak też w formie *on line*, jako strategia wzmocnienia roli społecznych interakcji edukacyjnych i uzupełnienia tradycyjnej ścieżki nauczyciel → uczeń;
- czynnościowe kształtowanie postrzegania i rozumienia kluczowych pojęć informatycznych i mechatronicznych, poprzez wykonywanie atrakcyjnych zadań implementacyjnych, egzemplifikujących dane pojęcia, zamiast wyuczania reguł encyklopedyczno-definicyjnych;
- rozwój umiejętności twórczego posługiwania się oprogramowaniem wolnym i otwartym, jako efektywny sposób kształtowania kompetencji infotechnicznych, zamiast dotychczasowego szkolenia w zakresie obsługi drogich, nadmiarowych, zawierających błędy i wręcz niebezpiecznych zamkniętych systemów operacyjnych oraz programów użytkowych;
- upowszechnienie umiejętności prowadzenia elektronicznej dokumentacji e-Portfolio, potwierdzającej długofalowy proces samorozwoju i osobisty dorobek uczniów – wynikający z ich partycypacji w projektowaniu i realizowaniu implementacji programistycznych bądź mechatronicznych.

W prezentowanym tu Programie wyraźnie podkreślamy, iż nauczanie i uczenie się należy zawsze w kształceniu traktować jako elementy nierozłączne. Chodzi o zaakcentowanie w pełni kontekstu semantycznego, gdyż słowo ‘nauczanie’ w języku polskim wskazuje raczej na transfer od nauczyciela do ucznia. Także samo słowo ‘kształcenie’ nie odzwierciedla jednoznacznie tego, co zawarte jest w jego naukowej definicji, a mianowicie – jako integralnego procesu nauczania-uczenia się. Pojęcie ‘nauczanie-uczenie się’ jest ugruntowane w języku środowisk naukowych i oświatowych. Rzadziej jednak występuje jako doprecyzowanie tego, czym jest konkretny program związany z kształceniem. Wynika to stąd, iż łatwiej jest opracować szczegółowe zalecenia *jak nauczać*, niż *jak się uczyć*.

W systemie oświaty dominuje wciąż słowo ‘nauczanie’, mimo że w dokumentach regulujących funkcjonowanie placówek coraz bardziej cele szczegółowe wyrażane są jako stany osiągnięć uczniów. Przykładowo, w Podstawach programowych nadal używane są śródtytuły „Treści nauczania”, mimo że wymagania szczegółowe zoperacjonalizowano jako to, co uczeń powinien umieć wykonać. Aby jednak potrafił to coś wykonać, nie wystarczy próbować go tego nauczyć – to on sam musi podjąć systematyczne działania, aby się tego uczyć. Właśnie z powyższych względów prezentowany tu model jest określony mianem **Programu nauczania-uczenia się**. Pragniemy w ten sposób podkreślić wagę potrzeby rzeczywistego zaangażowania w proces dydaktyczny obu podmiotów, zarówno nauczycieli, jak i uczniów.

Upowszechnienie niniejszego Programu ma na celu wdrożenie taktyki prowadzącej do zmian jakościowych w nauczaniu i uczeniu się podczas pozalekcyjnych form zajęć informatycznych oraz mechatronicznych (nazywanych tu *infotechnicznymi* i oznaczanych skrótem IT). Wyjaśnijmy, że w publikacji przyjęliśmy zasadę skrótowego określania łącznego obszaru merytorycznego z dziedzin komputerowego przetwarzania informacji, inżynierii programowania i technik budowy interfejsów mikroprocesorowych.

Wprowadzone pojęcie 'infotechnika' obejmuje tu w szczególności problematykę projektowania cyfrowych implementacji, tworzenia oprogramowania i konstruowania układów elektronicznych. W odróżnieniu od tematyki realizowanej w systemie kształcenia ogólnego, skierowanej głównie na użytkowanie technologii informacyjno-komunikacyjnych (TIK), propagowane tu zajęcia pozalekcyjne z infotechniki (IT) mają na celu przede wszystkim formowanie **kompetencji twórczych** w tych obszarach.

Założenie o potrzebie zmian jakościowych w edukacji infotechnicznej wynika z autentycznego zapotrzebowania pracodawców. Okazuje się, że tylko nieliczni absolwenci szkolnictwa zawodowego, a nawet uczelni, potrafią wykonać twórcze zadania implementacyjne, jakie są wymagane w zakładzie pracy na stanowisku informatyka czy mechatronika. Potrzebne jest inne podejście do edukacji w tym zakresie. Pozostawiając szkolnictwu ogólnemu zadania powszechnego przygotowania wszystkich uczniów do posługiwania się technologiami i aplikacjami użytkowymi, wdrożyć trzeba system odpowiednio wczesnego, uzupełniającego formowania poszerzonych kompetencji infotechnicznych na zajęciach pozalekcyjnych. Proces ten nie może obejmować wyłącznie uczniów wybitnie uzdolnionych, lecz powinien zawierać mechanizmy zainteresowania i wspierania każdego chętnego ucznia i uczennicy.

Model kół zainteresowań infotechnicznych powinien różnić się od dotychczasowego następującymi **elementami innowacji**:

Dominuje na tradycyjnych kołach zainteresowań IT	Zalecane w innowacyjnej realizacji kół zainteresowań IT
Organizowanie kół jedynie dla uczniów, którzy wykazują predyspozycje, bez szerszej promocji pożytków płynących z tej formy zajęć	Aktywna forma zainteresowania dziedziną IT i nabór na koła szerszej rzeszy uczniów poprzez prezentację promocyjną
Aktywizacja i doskonalenie jednostek uzdolnionych kierunkowo	Aktywizacja i wczesne ukierunkowanie grup wymagających wsparcia
Nastawienie na osiąganie głównie celów poznawczych i umiejętności (wiedza episteme i techne)	Formowania bardziej trwałych cech kierunkowych, wolicjonalnych, świadomości i postaw (wiedza phronesis, metiers i know how)
Taksonomia celów ogólnych ABC, wyznaczająca poziomy zapamiętania, zrozumienia i zastosowania	Taksonomia dedykowana na potrzeby kół IT, z harmonizowaniem efektów poznawczych, doznaniowych i czynnościowych
Zaprogramowany na realizację celów styl prowadzenia zajęć	Adaptacyjny styl prowadzenia zajęć, nastawiony na realne możliwości
Ocenianie sprawdzające osiąganie celów założonych w planie kół	Ocenianie rzeczywiste działań i efektów bieżących (wytworów) oraz odroczonych (e-portfolio, dystansowe pomiary zmian cech)

Tworzone ad hoc narzędzia oceniania jakości zajęć i ewaluacji efektów	Standaryzowane narzędzia oceniania jakości zajęć i ewaluacji efektów
Poszukiwanie pomysłów na treści i zadania, tworzenie bądź stosowanie dostępnych konspektów o niespójnych strukturach	Stosowanie przetestowanego, kompleksowego pakietu konspektów w układzie alternatywnych modułów i bloków do wyboru
Stosowanie systemów operacyjnych, narzędzi, instrukcji i materiałów dydaktycznych obwarowanych zamkniętymi licencjami, co utrudnia możliwość kontynuacji samokształcenia w domu	Przekazanie mobilnego, wolnego i otwartego systemu z dedykowanym pakietem narzędzi, źródeł, instrukcji i materiałów dydaktycznych, z możliwością ich doskonalenia i nieograniczonego wykorzystywania
Klasyczna metoda projektu, w której dopuszczalny jest brak założonych rezultatów i ew. kontynuacja na kolejnych zajęciach	Metoda kompletnych dzieł, która zakłada działającą implementację jako konieczny rezultat każdej jednostki zajęciowej
Zadania polegają na rozwiązywaniu problemów algorytmicznych, informatycznych, matematycznych bądź mechatronicznych	Zadania polegają na tworzeniu w pełni funkcjonalnych implementacji (gry, animacje, prezentacje, interfejsy, układy pomiarowe)
Implementacje nastawione są na realizację algorytmów oraz stosowanie funkcji, procedur i instrukcji danego języka	Implementacje wspomagają zrozumienie poprzez wizualizację efektów realizacji danego algorytmu, funkcji, procedury lub instrukcji
Większość operacji na obiektach abstrakcyjnych	Przejście od reprezentacji enaktywnych i ikonicznych do abstrakcji
Tendencje do nadmiernego czasowo, wyłącznie werbalnego wprowadzania w tematykę zajęć i w treść zadań do wykonania	Skondensowana zajawka wizualna, działaniowa, ew. słowna, pełniąca rolę zwiastuna tematyki i przybliżająca istotę zadań do wykonania
Pełne wyjaśnianie wszystkiego, co z punktu widzenia nauczyciela należałoby przedstawić uczniom	Redukcja objaśnień, sprowadzona głównie do odpowiedzi na pytania inicjowane przez uczniów (responsywność)
Nadmiarowe treści zajęć, co skutkuje nieefektywnym zagospodarowaniem czasu zajęć pozalekcyjnych	Szybkie przemykanie przez treść, z celowym niedostatkiem informacji możliwych do pozyskania samodzielnego (zapping).
Podpowiadanie /podawanie gotowych rozwiązań problemów, jakie powinny być z niewielkim wsparciem pokonane przez ucznia	Wspieranie niewyręczające, np. pytania naprowadzające na podjęcie właściwej decyzji i rozwiązanie problemu przez ucznia (inquiring)
Przewaga pojęciowego sterowania czynnościami uczniów	Przewaga czynnościowego kształtowania pojęć.
Programowanie imperatywne – tworzenie implementacji głównie poprzez czysto tekstowe pisanie kodu źródłowego	Programowanie wizualne – tworzenie implementacji w środowisku GUI poprzez wykorzystanie gotowych kontrolek (widżetów)
Uczenie się programowania jako od razu próba pisania kodu źródłowego, zamiast najpierw czytania kodu ze zrozumieniem	Uczenie się programowania w cyklu: czytanie kodu, uzupełnianie luk, modyfikacja, próba tworzenia kodu (metoda glottodydaktyczna)
Ciągłe w toku zajęć oddziaływania nauczycieli, a zbyt rzadkie pozostawienie uczniom swobody w ich procesach myślowych	Stopniowe redukowanie w toku zajęć interakcji z trenerem i z grupą na rzecz pełnego zanurzenia się w procesie twórczym (immersja)
Wdrażanie do rozwiązywania zadań na potrzeby własnego rozwoju	Wdrażanie do upowszechniania i dokumentowania swych osiągnięć w e-repozytorium i e-portfolio
Lokalna współpraca i transfer wiedzy	Ogólnopolska współpraca i transfer
Formy stacjonarne zajęć, wzbogacane rozproszonymi zasobami Internetu	Integracja formy stacjonarnej z samokształceniem zdalnym poprzez zasoby i funkcjonalności dedykowanej platformy e-Swoi

Uczenie się twórczości infotechnicznej jest procesem znacznie bardziej złożonym od nauki użytkowania technologii i niedającym się w pełni zunifikować. Ujmowanie tak zawiłego procesu w ramy konkretnego Programu sprawia duże trudności. Potencjalnym rozwiązaniem tego problemu jest wprowadzenie mechanizmów przygotowujących młodych adeptów do samokształcenia i ukierunkowujących ten proces. Kształtowanie umiejętności rozwiązywania problemów inżynierskich musi koniecznie odbywać się poprzez działania praktyczne, z niewyręczającym wsparciem trenera i rówieśników.

W początkowym etapie wdrażania do nowych obszarów wiedzy infotechnicznej potrzebne jest uwzględnianie aspektów psychopedagogicznych. Wynika to z niepełnej gotowości uczniów rozpoczynających naukę w gimnazjum do procesów umysłowych wyższego rzędu, stąd sposób postępowania można przyrównać do zintegrowanej strategii uczenia wczesnoszkolnego. I tak – przygotowanie do działań na obiektach abstrakcyjnych, jakie występują przy programowaniu, musi być wspierane poprzez operowanie na obiektach rzeczywistych lub symulowanych wizualnie. Uczenie „wypowiadania się" w kodzie języka programowania musi być poprzedzone wdrożeniem najpierw do odczytywania elementów składni przykładowego kodu (instrukcji), a dopiero później pisania większych konstruktów (procedur, funkcji).

Na kołach zainteresowań harmonizowane muszą być aspekty poznawczo-kształcące, emocjonalno-motywacyjne i psychomotoryczne, z waloryzacją trudnych treści i formowaniem cech względnie trwałych, przynoszących korzyści w dłuższej perspektywie. Służy temu wypracowana i przedstawiona tu struktura realizacji jednostek dydaktycznych w formie kół, z **metodyką szczegółową i taksonomią efektów**. Zaleca się, aby na każdych zajęciach osiągnąć cztery fazy: *sensytywności* (uwrażliwienie), *responsywności* (uaktywnienie), *problemowości* (decydowanie) i *konstruktywności* (tworzenie). Odbywa się to poprzez uspójnianie wielu metod nauczania-uczenia się i rodzajów oddziaływań, formowanie różnych komponentów postaw i osiąganie kierunkowych efektów świadomościowo-emocjonalnych.

Dedykowany na koła zainteresowań Program nauczania-uczenia się infotechniki doprecyzowany jest przede wszystkim w **Konspektach-scenariuszach zajęć**. To one są zawsze najbardziej przydatnym materiałem dydaktycznym dla nauczycieli. Są nie tylko źródłem pomysłów na to, jaką tematykę warto realizować, jakie stawiać cele i na jakie liczyć efekty, co i jak mają ćwiczyć uczniowie oraz jakimi narzędziami mają się posługiwać. Są także inspiracją do własnych opracowań. Zróżnicowane treści, proponowane do realizacji na kołach zainteresowań IT, wnoszą różnorodność i alternatywę dla standardowego materiału nauczania, umożliwiając fakultatywny wybór z dostosowaniem programu do potrzeb i możliwości uczniów. Konspekty-scenariusze zawierają powiązanie czynności uczniów z działaniami trenera i z mediami-środkami używanymi w danej fazie zajęć. Porządkują także chronologię czynności niezbędnych dla wykonania zadań.

Warto zwrócić uwagę, że Konspekty-scenariusze w proponowanej formie **nie są scenopisami** do wiernej realizacji krok po kroku. Zgodnie z nieodzownym stylem adaptacyjnym – tj. dynamicznym dopasowywaniem się do rzeczywistych sytuacji – i tak trener jest głównym animatorem tego, co dzieje się na zajęciach. W partycypacyjnym modelu także uczniowie stają się moderatorami narracji oraz współautorami dzieł, poprzez propozycje rozwiązań alternatywnych bądź rozszerzających funkcjonalność.

Konspekty są **zalecanymi wzorcami** do przemyślanego wykorzystania, przetestowanymi praktycznie i wyselekcjonowanymi jako wartościowy materiał dydaktyczny, który uzyskał wysoką ocenę uczestników testowania: uczniów, trenerów, nauczycieli-opiekunów obserwujących zajęcia oraz ekspertów. Rolą nauczycieli chcących wykorzystać te materiały jest trafny dobór modułów do możliwości uczniów i ułożenie ich odpowiedniej sekwencji w optymalny plan zajęć.

Zgodnie z nazwą projektu „Strategia wolnych i otwartych implementacji", bardzo ważną rolę dydaktyczną pełnią tu **implementacje**. W tej Strategii edukacyjnej są to wszelkie wytwory w postaci cyfrowej i/lub elektronicznej, jakie powstają w toku rozwiązywania wyznaczonych zadań projektowych, programistycznych lub mechatronicznych. Implementacje są materializacją koncepcji algorytmicznych i urzeczywistnieniem modeli abstrakcyjnych. Sam proces przechodzenia od pomysłu do wytworu jest implementowaniem. Ze względu na to, że procesy inżynierskiego programowania czy konstruowania interfejsów są regulowane wypracowanymi specjalnie w tej dziedzinie metodykami postępowania (np. *programowanie zwinne*), tworzenie na kołach zainteresowań konkretnej implementacji samo w sobie ma niejako zawartą optymalną metodę dydaktyczną. Środowisko pracy i narzędzia służące do implementowania stają się wówczas środkami dydaktycznymi, a działający wytwór pełni dodatkowo funkcję poglądową. Zatem implementacje w Programie nauczania-uczenia się infotechniki są *integralnymi środkami-metodami*, w pełnym znaczeniu tego pojęcia.

Implementacje zalecane do wykorzystywania lub tworzone na zajęciach mają różną postać fizyczną. Najbardziej właściwą formą jest **postać cyfrowa**. W postaci opisu nie daje się odzwierciedlić w pełni istoty funkcjonowania takich wytworów. Ponadto nawet skrócone opisy implementacji powiązanych z Konspektami przekraczałyby dopuszczalną objętość publikacji. Z tego powodu w niniejszej publikacji zawarto jedynie opisy 11 wybranych z ponad stu przygotowanych i przetestowanych, natomiast wszystkie implementacje proponowane do realizacji na kołach zainteresowań IT umieszczono na płycie dołączonej do książki oraz w Serwisie e-Swoi.

W ogólnym ujęciu silnie zróżnicowanych form implementacji są to: opisy zadań, schematy układów, instrukcje postępowania lub przykładowe rozwiązania, łącznie z ilustracjami i kodami źródłowymi. Przyjęto założenie, że to trener w uzgodnieniu z uczniami podejmuje decyzję o tym, jaki fragment opisu implementacji udostępnia uczniom – czy tylko treść zadania do wykonania w całości od podstaw, czy część kodu źródłowego z lukami do wypełnienia, czy pełne kody z zadaniem modyfikacji lub rozszerzenia.

Indywidualne dopasowanie stopnia trudności zadań jest konieczne ze względu na silne zróżnicowanie potencjału uczniów. Z tego też powodu założone cele szczegółowe i oczekiwane efekty muszą być adekwatne do realnych możliwości. Wynika stąd zupełnie inny od tradycyjnego sposób oceniania osiągnięć. Podstawą jest **ocenianie rzeczywiste**, polegające na obserwacji aktywności i czynności wykonywanych przez uczniów. Dotyczy to zarówno oceny jakości interakcji werbalnych, jak też prawidłowości rozwiązywania zadań i wykonania implementacji. Oprócz bezpośredniego wyrabiania sobie opinii przez trenera, znamienna w proponowanym Programie jest możliwość wykorzystania przetestowanych na dużej próbie i wystandaryzowanych narzędzi do ukierunkowanej obserwacji, pogłębionej refleksji i do ewaluacji efektów.

Na potrzeby oceniania jakości realizacji kół zainteresowań IT udostępniamy *Arkusz obserwacji* przeznaczony dla osób hospitujących zajęcia. Także dla nich, lecz przede wszystkim dla trenerów polecamy *Protokół formatywny*, służący do refleksji odnoszonej do tego, jak w rzeczywistości udało się zrealizować najważniejsze elementy metodyczne, w tym innowacyjne. Te **narzędzia pomiarowe** są tak skonstruowane, że dotyczą obu podmiotów: trenera i uczniów, co umożliwia analizę oddziaływań i skutków. Do wartościowania efektów zajęć polecamy *Ankietę ewaluacyjną* dla uczniów, zbudowaną w formie skali psychometrycznej, służącej do pomiaru zmian, jakie zaszły pod wpływem całego cyklu zajęć w świadomości, postawach i cechach wolicjonalnych uczniów.

Oprócz dedykowanych narzędzi pomiarowych, nauczyciele otrzymują obszerny opis kompleksowej **metodologii** oceniania i ewaluacji, wraz ze schematami interpretacji jakościowej i analizy ilościowej w odniesieniu do norm wyznaczonych empirycznie dla populacji. Ta metodologia może być wzorcem dla opracowywania własnych narzędzi, przeznaczonych na inne zajęcia szkolne.

W realizacji Strategii edukacyjnej z obszarów infotechniki istotną rolę odgrywają **narzędzia pracy** uczniów i trenerów. Nie tylko dlatego, że od rodzaju wykorzystywanych systemów oprogramowania i aplikacji użytkowych zależy dostępność tych narzędzi w szkole, lecz także dlatego, że z założenia uczeń powinien mieć możliwość ćwiczenia także w domu lub w dowolnym innym miejscu. Przyjęto zasadę, że dedykowane do wykonywania implementacji narzędzia muszą być autonomiczne, niezależne od różnych systemów zainstalowanych na danym komputerze. Ponadto narzędzia te są udostępniane na liberalnych licencjach, aby ani szkoła, ani uczniowie nie ponosili kosztów zakupu systemu i oprogramowania. Wreszcie – niezbędny jest otwarty dostęp do kodów źródłowych i dokumentacji oprogramowania. Warunki te spełnia wykorzystanie **Wolnego i otwartego oprogramowania** (WiOO).

Założenia ideowe wolności i otwartości oraz ruchy społeczne wspierające te idee zrodziły się z potrzeby przeciwdziałania monopolizacji oprogramowania. Powstały i ciągle doskonalone są systemy oraz aplikacje oparte na wolnych i otwartych licencjach, zwykle zredukowanych jedynie do uznania autorstwa. Dzięki humanizacji informatyki i globalizacji dorobku ludzkości udostępniane są otwarte źródła, zasoby edukacyjne i naukowe, standardy technologiczne i metodologiczne, a także wolna kultura. Z tych dobrodziejstw korzysta Program nauczania-uczenia się infotechniki i obopólnie – dorobek Projektu SWOI jest udostępniany jako wolny i otwarty.

Platformą udostępniania dorobku i wymiany informacji jest **Serwis e-Swoi**. Skupia on Społeczność nazywaną „Swoi", połączoną ideami nabywania umiejętności tworzenia implementacji i doskonalenia kompetencji infotechnicznych. Jego funkcjonalność wyraża się w kilku odmiennych usługach. Pierwszą z nich jest funkcja informacyjna, służąca otwartemu publikowaniu krótkich komunikatów, z mechanizmem promocyjnym *wylęgarni newsów*. Kolejnym miejscem publikacji jest *wiki*, gdzie umieszczane są dłuższe publikacje merytoryczne. Funkcjonalność o nazwie *program* ma na celu umożliwienie trenerowi budowę własnego planu zajęć poprzez wybór modułów. Poprzez usługę *e-tutor* użytkownik może skorzystać z doradztwa, zadać pytania i czytać odpowiedzi (FAQ) bądź dotrzeć do instrukcji. Struktura o skróconej nazwie *repo*, to obszar katalogów e-Repozytorium, w których umieszcza się i udostępnia własne implementacje i zasoby źródłowe. Funkcjonalność *e-portfolio* jest mechanizmem automatycznego rejestrowania aktywności publikacyjnej w Serwisie, a ponadto umożliwia pisanie osobistych refleksji o swym

udziale w Społeczności „Swoi". W usłudze *konkurs* ogłaszane są zadania i warunki konkursowe, a następnie publikowane wyniki.

Oprócz platformy zdalnej, kompleksowym, dedykowanym narzędziem pracy własnej jest **Szkolny Remiks Ubuntu**, oparty na dystrybucji Linux Ubuntu ze środowiskiem graficznym XFCE. Cały pakiet oprogramowania został dobrany i skonfigurowany specjalnie do ćwiczeń i wykonywania implementacji według zadań z Konspektów-scenariuszy. Zawarto tam wszystkie niezbędne materiały dydaktyczne, pakiety edycyjne, środowiska programowania i multimedia.

Na potrzeby mobilności, system i aplikacje są uruchamiane bezpośrednio z pamięci Flash USB, więc nie ma potrzeby instalowania na twardym dysku. Na tym przenośnym nośniku są też zapisywane pliki użytkownika. Szkolny Remiks Ubuntu jest tak bogato wyposażony w sterowniki i aplikacje, że może być zainstalowany na osobistym komputerze lub notebooku jako w pełni wystarczalne środowisko pracy ucznia i nauczyciela, bez potrzeby zakupu żadnych innych systemów bądź programów.

Do zajęć z mechatroniki potrzebny jest zestaw zwany *modułem-interfejsem*. Funkcjonuje on we współpracy z komputerem poprzez złącze USB. Podstawowym elementem jest płytka **Arduino UNO R3** z mikrokontrolerem ATmega328. Drugim elementem jest uniwersalna płytka montażowa, służąca do łączenia podzespołów w układ elektroniczny. Do programowania zestawu służy interfejs IDE lub środowisko Scratch S4a. Są to środowiska interaktywne, pozwalające na sterowanie bądź odczyt wartości z czujników. W skład zestawu do zajęć wchodzą też podstawowe elementy elektroniczne: rezystory, tranzystor, termistor, diody LED i RGB, wyświetlacz LCD, czujniki, przycisk, buzzer itp. W zależności od projektu, wykorzystywane są różne podzespoły, tworząc alternatywne konfiguracje. Taki zestaw w przystępnej cenie może być znakomitym prezentem, sprawiającym podczas konstruowania niesamowitą radość – od najmłodszych, aż po dorosłych.

Jak już wspomnieliśmy – obszerny materiał, służący realizacji Programu nauczania-uczenia się infotechniki, został umieszczony w postaci elektronicznej na płycie dołączonej do wydania książkowego. Stanowi ona integralną część publikacji, jest zbiorem opisów, kodów implementacji i zasobów źródłowych, jest narzędziem pracy ucznia i trenera, nośnikiem systemu, aplikacji użytkowych i niezbędnego do zajęć oprogramowania. Ze względu na potrzebę aktualizacji, nowsze wersje będą udostępniane do pobrania z Serwisu pod adresem http://e-swoi.pl/

W tym miejscu chcemy podkreślić, że opis dorobku Programu „Strategia Wolnych i Otwartych Implementacji" zawarty jest w dwóch tomach. Oprócz niniejszego tomu pt. „Program nauczania-uczenia się infotechniki" integralną publikacją jest tom pt. „Strategia nauczania-uczenia się infotechniki", która zawiera studium definicyjne i studium dydaktyczne. Teksty tam zawarte przybliżają założenia ideowe Strategii, taktykę jej wdrażania i realizacji oraz istotę proponowanych innowacji edukacyjnych. Gorąco zachęcamy Czytelników do zapoznania się z jednym i drugim tomem publikacji.

Stanisław Ubermanowicz, Krzysztof Wawrzyniak

Realizacja
Studium metodyczne

Studium metodyczne REALIZACJA precyzuje założenia i rekomendacje co do sposobów praktycznego urzeczywistnienia wzorcowego Programu nauczania-uczenia się infotechniki. Jest to trzon opisu wszystkich niezbędnych elementów Programu, tj.: jego genezy i podstaw, celów i treści, czynności, metod, środków i efektów oraz metodologii oceniania i ewaluacji. Głównymi narzędziami realizacji są Konspekty-scenariusze przeznaczone do prowadzenia zajęć w pozalekcyjnej formie kół zainteresowań. Realizowane zadania polegają na tworzeniu implementacji infotechnicznych, rozwijających umiejętności programowania, projektowania i konstruowania. Celem tych działań jest wczesne formowanie u uczniów zainteresowań w obszarze specjalistycznych kompetencji, dających perspektywę uzyskania dobrego zawodu.

Ta część opracowania przeznaczona jest dla użytkowników innowacyjnego Programu, zwłaszcza dla nauczycieli realizujących zajęcia komputerowe lub mechatroniczne, instruktorów organizujących zajęcia w placówkach wychowania pozaszkolnego, metodyków prowadzących kursy doskonalenia nauczycieli informatyki bądź elektroniki, a także dla studentów kierunków technicznych ze specjalizacją nauczycielską.

Treści studium metodycznego zgrupowano w rozdziałach:

- **Geneza i zawartość Programu nauczania-uczenia się infotechniki** – to tekst uzasadniający potrzebę zmian, ukazujący cele ogólne, proponowane treści i założone efekty merytoryczne;
- **Metodyka realizacji kół zainteresowań IT i taksonomia efektów** – to opis przedstawiający sposób racjonalizacji zajęć i charakterystykę kluczowych faz w toku jednostki dydaktycznej;
- **Konspekty-scenariusze realizacji kół zainteresowań IT** – to objaśnienie struktury konspektów oraz zbiór 84 zweryfikowanych w praktyce i zalecanych wzorców zajęć pozalekcyjnych;
- **Ocenianie realizacji kół zainteresowań IT i ewaluacja efektów** – to opis strategii, praktyki i narzędzi badania jakości zajęć oraz metodologia obserwacji, refleksji i wartościowania.

Konspekty-scenariusze są zbiorami rekomendacji, a ich dopełnieniem są (zamieszczone na płycie) Implementacje, jako zadania dla uczniów i przykłady rozwiązań. Trener otrzymuje do wyboru zróżnicowane treściowo Moduły zajęć o różnych poziomach trudności:

- **Moduły ścieżki 0 dla szkół podstawowych** wprowadzają dzieci w środowisko wolnego oprogramowania z aplikacjami użytkowymi, służącymi do edycji infografiki, kalkulacji i prezentacji, oraz w świat konstruowania elementarnych układów elektronicznych.
- **Moduły ścieżki A dla szkół gimnazjalnych** wdrażają młodych adolescentów do posługiwania się wolnymi narzędziami, do projektowania grafiki i programowania wizualno-skryptowego oraz do konstruowania mechatronicznych interfejsów pomiarowych.
- **Moduły ścieżki B dla szkół ponadgimnazjalnych** wdrażają adolescentów do programowania wizualno-skryptowego oraz obiektowo-zdarzeniowego, poprzez tworzenie animacji i gier logicznych, a także uczą konstruowania i sterowania modułów-interfejsów.
- **Moduły ścieżki C dla szkół sprofilowanych infotechnicznie** wdrażają do używania wolnych narzędzi programistycznych, poprzez ćwiczenie struktur różnych języków: C, Phyton, HTML, JavaScript oraz uczą konstruowania złożonych układów mechatronicznych.

02

Geneza i zawartość Programu nauczania-uczenia się infotechniki

Krzysztof Wawrzyniak, Stanisław Ubermanowicz

U podstaw Projektu innowacyjno-testującego, mającego na celu poprawę jakości specjalistycznego kształcenia informatycznego i mechatronicznego (zwanego *infotechnicznym*, IT), legło szereg problemów wymagających interwencji systemowej. Nadrzędnym zadaniem projektowym było zwiększenie zainteresowania przyszłymi wyborami kierunków technicznych w ścieżkach rozwojowych uczniów. W realizacji tak specyficznego zadania, mającego dawać **efekty odroczone**, kluczowe było zdiagnozowanie przyczyn złego stanu rzeczy, określenie sposobów naprawy oraz opracowanie, przetestowanie, upowszechnienie i włączenie do polityki oświatowej narzędzi naprawczych. W rezultacie trwających ponad trzy lata działań wypracowano innowacyjną taktykę o nazwie „Strategia Wolnych i Otwartych Implementacji" (SWOI), która obejmuje całościowo aspekty merytoryczne, dydaktyczne i organizacyjne form zajęć pozalekcyjnych, specyficzną metodykę ich realizacji, środki służące osiąganiu i dokumentowaniu efektów oraz kwestie doskonalenia kompetencji kadr.

Zasadniczym narzędziem w Strategii SWOI jest niniejszy Program nauczania-uczenia się infotechniki. Zakłada on celowe, wczesne oddziaływanie na adolescentów poprzez ciekawe treści zajęć i silnie motywujące formy ćwiczeniowe na kołach zainteresowań IT. Z założenia ma to ukształtować trwałą chęć pogłębiania wiedzy i umiejętności poprzez pracę własną oraz wybór dalszych etapów kształcenia w kierunku infotechniki. O tym, jak bardzo ważne jest pobudzenie poprzez ciekawe treści i atrakcyjne formy, świadczy olbrzymie powodzenie oferty Centrum Nauki Kopernik czy oblegane prezentacje podczas Nocy Naukowców. Przyjrzyjmy się, jakie są przyczyny niechętnego nastawienia do tradycyjnej oferty proponowanej w szkole.

Konieczność zmian w edukacji infotechnicznej

Z pogłębionej diagnozy aktualnego sposobu realizacji materiału z dziedzin informatyki i technologii informacyjnych w szkolnictwie ogólnym wynika, że kształcenie na wszystkich etapach obraca się głównie w zakresie użytkowania gotowych, zamkniętych aplikacji. Jest to oczywiście ważne dla przygotowania ogółu adolescentów do posługiwania się komputerem jako narzędziem pracy. Jednakże potrzeby rynku pracy są coraz większe i nie wystarcza już umiejętność wykorzystania pakietu biurowego. Potrzebne są liczne

kadry o umiejętnościach programistycznego tworzenia implementacji, a przynajmniej dostosowywania aplikacji do potrzeb pracodawcy, np. pisanie makroprogramów czy formuł automatyzujących działania bądź wydobywanie optymalnych kwerend z baz danych.

Takich kompetencji powszechna oświata w zasadzie nie kształtuje. Jeszcze gorzej jest z wiedzą i umiejętnościami z obszarów mechatroniki, a przecież mikroprocesory są dziś w bardzo wielu urządzeniach powszechnego użytku i w maszynach produkcyjnych. Wysoce specjalistyczne kompetencje muszą być formowane odpowiednio **wcześnie i długofalowo**, aby rzeczywiście posiąść wiedzę i umiejętności, jakie są potrzebne w zinformatyzowanym świecie. Na wyuczenie się i osiąganie odpowiedniego poziomu kompetencji dopiero na studiach jest już za późno, wskutek czego tak wielu studentów odpada, a ci, którzy kończą wyższe szkoły techniczne, mają zdecydowanie niewystarczające kwalifikacje.

W okresie studiów jest za późno na nadganianie zaległości kompetencyjnych, których nie ukształtowano w systemie oświaty.

Aktualne wady edukacji komputerowej, wymagające naprawy to m.in.:
- nastawienie przede wszystkim na użytkowanie pakietów biurowych;
- nikła wiedza o alternatywnych systemach, programach i narzędziach;
- brak wczesnego kształtowania twórczych kompetencji infotechnicznych;
- brak ukierunkowywania uczniów na pożądane ścieżki specjalizacji;
- uczenie programowania w imperatywnej metodyce pisania kodu;
- pisanie kodu w trybie tekstowym, bez wsparcia narzędzi z widżetami;
- nieobecność problematyki mechatronicznej w programach szkolnych.

Skutkami tak niekorzystnego stanu rzeczy są następujące zjawiska:
- niska zdawalność egzaminów na stopień technika w obszarach IT;
- mała wybieralność i słaby poziom wyników z informatyki na maturze;
- niewielkie zainteresowanie kierunkami IT, zwłaszcza uczennic;
- niewystarczające przygotowanie do studiowania na kierunkach IT;
- liczne rezygnacje studentów nieradzących sobie z treściami studiów;
- niedobór odpowiednio wykształconej kadry do zawodów IT;
- brak umiejętności obsługi systemów WiOO używanych w firmach;
- bardzo niski poziom wynalazczości w Polsce.

Jak już wspomniano – środkiem zaradczym może stać się wypracowana i upowszechniona Strategia SWOI, zakładająca nie tylko modernizację treści, form i metod kształcenia, lecz również przyjęcie innych akcentów w podstawach ideowo-programowych, w zakładanych celach i efektach, a także stosowanie innej taktyki oddziaływań.

W pierwszym rzędzie chodzi o uspójnienie edukacji szkolnej i pozaszkolnej, o wykorzystanie kół zainteresowań jako najefektywniejszej formy specjalizacji, o zachęcenie do uczestnictwa uczniów/uczennic niepewnych co do swych możliwości, o formowanie trwalszych cech osobowościowych i umiejętności twórczych, a nie przekazywanie ulotnych wiadomości, o ocenianie rzeczywiste osiągnięć, a nie sprawdzające zapamiętanie.

Obszerne opisy taktyki racjonalizacyjnej, mechanizmów, uwarunkowań, metod i narzędzi realizacji zawarte są w obu tomach publikacji. Tutaj skoncentrujmy się na sposobie wykorzystania Programu nauczania-uczenia się, na jego celach, treściach i spodziewanych efektach.

Dostosowanie do potrzeb i dyspozycji uczniów

Program jest elastyczny pod względem doboru alternatywnych treści, wyznaczanego stopnia trudności zadań i oczekiwanych efektów.

Proponowany Program nauczania-uczenia się w swojej istocie zakłada aktywność i zaangażowanie w jego rzeczywistą realizację wszystkich uczestników – chodzi o prawdziwe upodmiotowienie trenerów i uczniów. To właśnie oni poprzez akt sprawczy praktycznego wykonywania czynności stają się współautorami programu, choćby poprzez wybory modułów, modyfikację zadań, dostosowywanie do własnych potrzeb, do osobistych preferencji i warunków pracy. Zyskana dzięki temu swoboda umożliwia efektywniejsze osiąganie celów kształcenia, pozostawiając trenerowi możliwość elastycznego podążania za uczniem i dostosowywania procesu kształcenia do indywidualnych oczekiwań wychowanków.

Program ten, jako swoisty wzorzec, ma również na celu zachęcanie nauczycieli do opracowywania różnych innych ofert edukacyjnych, z dopasowaniem do potrzeb i zainteresowań uczniów oraz włączania ich do planowania i realizacji zajęć. Takie wyjście naprzeciw wychowankom jest szansą na zwiększenie odpowiedzialności i zaangażowania uczniów do współtworzenia programów nauczania-uczenia się, a przez to – aktywniejszego uczestnictwa w zajęciach i większej motywacji do pracy, przekładających się na kształtowanie pasji i wiązanie dalszego kształcenia zgodnego z własnymi zainteresowaniami.

Realizacja Programu ma szansę być w pełni satysfakcjonująca dla trenera i uczniów pod warunkiem jego dynamicznego dostosowywania do konkretnych sytuacji i kontekstów edukacyjnych. Oznacza to pożytki z podjęcia przez trenera wskazanych poniżej działań:

- wybór tematyki adekwatnej do formowanych umiejętności i potrzeb uczniów;
- ułożenie sekwencji modułów w cykl spójny środowiskowo i tematycznie;
- modyfikowanie założonych celów i odpowiadających im osiągnięć ucznia;
- skonsultowanie propozycji z uczniami, zachęcenie ich do modyfikacji;
- dobranie form i metod pracy z uczniami stosownie do treści i faz realizacji.

W przypadkach daleko idących przeróbek warto pamiętać, że Program jest dorobkiem grona specjalistów, testowanym w licznych i zróżnicowanych środowiskach, optymalizowanym na podstawie wniosków formatywnych z obserwacji zajęć i ewaluacji efektów. Z tego względu stanowi on wart wykorzystania lub naśladowania wzorzec, a ewentualne modyfikacje zmierzające w stronę dotychczasowych przyzwyczajeń, sprzecznych z zaleceniami naprawczymi, zniweczyłoby cały wysiłek autorów i nie dałoby oczekiwanych efektów.

Cele ogólne kół zainteresowań infotechnicznych

W zajęciach pozalekcyjnych nie ma w zasadzie obowiązku, aby przygotować Program realizujący zapisy Podstawy programowej. Niemniej jednak możemy właśnie w niej znaleźć pewne wskazania co do celów, materiału nauczania i sposobów ich ujmowania. Analizując Podstawę programową kształcenia ogólnego, można trafić na zapis: „Celem kształcenia ogólnego na III i IV etapie edukacyjnym jest:

1. przyswojenie przez uczniów określonego zasobu wiadomości na temat faktów, zasad, teorii i praktyk;
2. zdobycie przez uczniów umiejętności wykorzystania posiadanych wiadomości podczas wykonywania zadań i rozwiązywania problemów;
3. kształtowanie u uczniów postaw warunkujących sprawne i odpowiedzialne funkcjonowanie we współczesnym świecie”.

Cele ogólne kół zainteresowań IT obejmują na równi formowanie wszystkich trzech sfer: poznawczą, wykonawczą i doznaniową.

Poniższe zestawienie celów ogólnych zawartych w proponowanym tu Programie nauczania-uczenia się dotyczy równomiernego formowania wszystkich trzech sfer: reprezentacji poznawczych (wiadomości), sfery wykonawczej (umiejętności) i doznaniowej sfery afektywnej (postawy). W konkretnym odniesieniu do obszarów infotechniki, są to:

Wiadomości

- Przyswojenie zagadnień z informatyki, algorytmiki, elektroniki i mechaniki;
- Znajomość kluczowych pojęć infotechnicznych i ich czynnościowe rozumienie;
- Znajomość istoty i pożytków wolnego systemu operacyjnego i otwartych aplikacji;
- Wiedza o złożonych i różnorodnych procesach tworzenia implementacji;
- Poznanie środowisk programowania i konstruowania układów mechatronicznych;
- Znajomość zasad i procedur osiągania bezpieczeństwa danych i szyfrowania.

Umiejętności

- Skuteczne i sprawne poruszanie się w systemie Linux i w środowisku Ubuntu;
- Umiejętności korzystania z narzędzi wolnego i otwartego oprogramowania;
- Umiejętności tworzenia implementacji i doskonalenia ich działania;
- Planowanie, podejmowanie i organizacja pracy w środowisku programistycznym;
- Projektowanie graficzne, obiektowo-skryptowe i wizualno-obiektowe;
- Programowanie imperatywne, wsadowe i obiektowo-zdarzeniowe;
- Konstruowanie i oprogramowywanie układów mechatronicznych.

Postawy

- Chęć ciągłego zdobywania i modyfikowania wiedzy i umiejętności;
- Przekonanie o potrzebie formowania cech logicznego myślenia i planowania;
- Motywacja do twórczego podejścia w implementowaniu i doskonaleniu dzieł;
- Uznanie dla znaczenia stylu partnerstwa z innymi uczniami oraz z trenerem;
- Przekonanie o znaczeniu interakcji w grupie podczas fazy projektowania;

- Wytrwałość w samodzielnym tworzeniu prawidłowo działających implementacji;
- Staranność działań w środowisku programistycznym i mechatronicznym.

Warto zaznaczyć, że cele ogólne Programu odzwierciedlają przede wszystkim dążenia do osiągnięcia stanu, który jest głównym zadaniem, czyli – **zwiększenia zainteresowania** wyborem kierunków infotechnicznych w dalszym kształceniu. Szczególny nacisk został położony na kształtowanie względnie trwałych postaw, przede wszystkim w warstwie wolicjonalnej.

Odniesienie Programu do Podstaw programowych

Analiza celów ogólnych Programu nauczania-uczenia się infotechniki ukazała, iż zasadniczo realizują one zapisy Podstawy programowej kształcenia ogólnego dla III i IV etapu edukacji, obowiązującej w polskim systemie oświaty. Również treści znajdują swoje odzwierciedlenie w szczegółowych podstawach przedmiotów szkolnych, co ukazuje poniższe zestawienie:

Podstawa programowa gimnazjum: informatyka	Program nauczania-uczenia się (moduły A)
Bezpieczne posługiwanie się komputerem i jego oprogramowaniem, korzystanie z sieci komputerowej. Uczeń: • posługuje się urządzeniami multimedialnymi, na przykład do nagrywania/odtwarzania obrazu i dźwięku; • wyszukuje i uruchamia programy, porządkuje i archiwizuje dane i programy; stosuje profilaktykę antywirusową; • samodzielnie i bezpiecznie pracuje w sieci lokalnej i globalnej;	Praca ze Szkolnym Remiksem Ubuntu – dedykowanym systemem operacyjnym i oprogramowaniem narzędziowym; praca z programami graficznymi Gimp i Inkscape, tworzenie grafiki komputerowej; praca w środowisku wizualno-skryptowym Scratch, projektowanie i programowanie, tworzenie animacji; praca w Serwisie e-Swoi, dedykowanym do zdalnej formy edukacji; poszukiwanie i pobieranie z Internetu wolnych i otwartych zasobów źródłowych na licencji Creative Commons
Wyszukiwanie i wykorzystywanie (gromadzenie, selekcjonowanie, przetwarzanie) informacji z różnych źródeł; współtworzenie zasobów w sieci. Uczeń: • pobiera informacje i dokumenty z różnych źródeł, w tym internetowych, ocenia pod względem treści i formy ich przydatność do wykorzystania w realizowanych zadaniach i projektach; • umieszcza informacje w odpowiednich serwisach internetowych.	Praca w Serwisie e-Swoi, dedykowanym do zdalnej formy edukacji; praca z e-Repozytorium przeznaczonym do gromadzenia, przechowywania i udostępniania zasobów źródłowych i własnych implementacji; praca z e-Portfolio przeznaczonym do opisywania działań i dokumentowania własnych osiągnięć ucznia
Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego. Uczeń: • wyjaśnia pojęcie algorytmu, podaje odpowiednie przykłady algorytmów rozwiązywania różnych problemów; • formułuje ścisły opis prostej sytuacji problemowej, analizuje ją i przedstawia rozwiązanie w postaci algorytmicznej; • wykonuje wybrane algorytmy za pomocą komputera.	Praca z algorytmem w programie Dia, opanowanie podstaw myślenia algorytmicznego, programowanie w środowisku wizualno-skryptowym Scratch – tworzenie skryptów obsługujących obiekty, budowanie kodu źródłowego, wizualizującego i realizującego algorytm w środowisku graficznym

Podstawa programowa gimnazjum: technika	Program nauczania-uczenia się (moduły A)
Opracowywanie koncepcji rozwiązań typowych problemów technicznych oraz przykładowych rozwiązań konstrukcyjnych	Praca z mikroprocesorowym modułem-interfejsem Arduino, montowanie układów mechatronicznych
Podstawa programowa liceum: informatyka	**Program nauczania-uczenia się (moduły B)**
Uczeń wykorzystuje technologie komunikacyjno-informacyjne do komunikacji i współpracy z nauczycielami i innymi uczniami, a także z innymi osobami, jak również w swoich działaniach kreatywnych. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego. Uczeń: • prowadzi dyskusje nad sytuacjami problemowymi; • formułuje specyfikacje dla wybranych sytuacji problemowych; • projektuje rozwiązanie: wybiera metodę rozwiązania, odpowiednio dobiera narzędzia komputerowe, tworzy projekt rozwiązania; • realizuje rozwiązanie na komputerze za pomocą oprogramowania aplikacyjnego lub języka programowania; • testuje otrzymane rozwiązanie, ocenia jego własności, w tym efektywność działania oraz zgodność ze specyfikacją;	Praca w Serwisie e-Swoi, praca nad implementacjami w zespołach projektowych, zadania konkursowe i zadania na kołach, praca w środowisku wizualno-skryptowym Scratch – tworzenie skryptów obsługujących obiekty, budowanie kodu źródłowego, wizualizującego i realizującego algorytm w środowisku graficznym; praca w środowisku programowania wizualno-obiektowego Lazarus (tworzenie, uzupełnianie i komentowanie kodu), mechatronika, programowanie i konstruowanie, projektowanie, tworzenie i testowanie implementacji
Podstawa programowa technikum: informatyka	**Program nauczania-uczenia się (moduły C)**
Komunikowanie się za pomocą komputera i technologii informacyjno-komunikacyjnych. Uczeń: • wykorzystuje zasoby i usługi sieci komputerowych w komunikacji z innymi użytkownikami, w tym do przesyłania i udostępniania danych; • bierze udział w dyskusjach w sieci (forum internetowe, czat). • sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu i uruchamianiu programów; • stosuje podstawowe konstrukcje programistyczne w wybranym języku programowania, instrukcje iteracyjne i warunkowe, rekurencję, funkcje i procedury, instrukcje wejścia i wyjścia • poprawnie tworzy strukturę programu • realizuje indywidualnie lub zespołowo projekt programistyczny	Zaawansowana praca w środowisku Linux i w Szkolnym Remiksie Ubuntu (dedykowany system operacyjny i oprogramowanie narzędziowe); edukacja zdalna i współpraca w społeczności sieciowej Serwis e-Swoi (dedykowana do zajęć platforma edukacyjna); programowanie w językach C, Python, HTML5 i JavaScript, montowanie układów mechatronicznych; praca w grupie nad zadaniami programistycznymi

Należy w tym miejscu jednak wyraźnie zaznaczyć, że Program, który jest przeznaczony na zajęcia pozalekcyjne w formie kół zainteresowań i aktywności pozaszkolnej, zasadniczo nie ma na celu realizacji zapisów Podstawy programowej. Nawiązując do zakresów tematycznych Podstawy, każdorazowo je przekracza, oferując treści spoza tych, jakie w szkolnictwie ogólnym obowiązują wszystkich uczniów. Jego celem nadrzędnym jest uzupełnienie kształcenia powszechnego o formowanie pogłębionych, twórczych umiejętności infotechnicznych, które są konieczne dla wczesnego ukierunkowania specjalistycznego.

Program realizuje założenia Podstawy, lecz przekracza ją zakresem i poziomem trudności, dlatego nie powinien być realizowany dla wszystkich uczniów.

Opis modułów i oczekiwane efekty merytoryczne

Proponowane w Programie treści zostały podzielone na elementarne moduły, zgrupowane w bloki merytoryczne i ścieżki programowe o różnych poziomach trudności. **Moduły** to domknięte treściowo i organizacyjnie jednostki dydaktyczne kół zainteresowań IT, zaplanowane na 90 minut w szkołach gimnazjach i szkołach ponadgimnazjalnych, a w szkołach podstawowych na 45 minut. W tym czasie realizowane są pojedyncze Konspekty-scenariusze, połączone ściśle z wykonywaniem przypisanych im Implementacji – w pełni funkcjonalnych wytworów programistycznych lub mechatronicznych. Z tego powodu niedopuszczalne jest dzielenie modułów i przenoszenie części materiału na następne zajęcia. Całościowe wykonanie implementacji zaplanowanych na daną jednostkę jest konieczne i realne, co potwierdziło testowanie w 110 szkołach o różnych poziomach i profilach. Natomiast w niektórych Konspektach wskazano także część zadań rozszerzających, przeznaczonych do samodzielnego wykonania przez uczniów już poza zajęciami.

Dany moduł powinien być w całości zrealizowany na jednym spotkaniu, podczas którego musi powstać zaplanowana implementacja.

Cechą układu modułowego jest możliwość w miarę dowolnego układania Planu i doboru treści zajęć przez trenerów. Warto przy tym wybierać jako ciągłość treściową przynajmniej cztery moduły stanowiące spójny środowiskowo **blok merytoryczny**. W taki sposób pogrupowano w niniejszej publikacji Konspekty-scenariusze.

Na pełne ścieżki programowe kół zainteresowań IT – oznaczone adekwatnie do poziomu trudności kodami 0, A, B i C – składają się po trzy bloki merytoryczne, dając materiał dydaktyczny na minimum 12 spotkań w formie stacjonarnej. Realizowane zagadnienia powinny obejmować zarówno projektowanie i programowanie, jak też konstruowanie układów mechatronicznych.

Celowym zabiegiem motywująco-waloryzacyjnym jest taka sekwencja, w której najpierw wprowadza się tematykę użytkowania systemu i aplikacji Szkolnego Remiksu Ubuntu, w tym edycję i animację grafiki komputerowej (z wyjątkiem szkół sprofilowanych IT). Dopiero w drugiej części cyklu zajęć realizuje się najtrudniejsze dla początkujących treści, związane z uczeniem się programowania. Zwieńczeniem cyklu jest blok zajęć mechatronicznych, które sprawiają uczniom największą satysfakcję na wszystkich etapach edukacji.

Pakiet Szkolnego Remiksu Ubuntu wraz z kompletnym oprogramowaniem potrzebnym do zajęć, znajduje się na dołączonej do książki płycie.

Przeznaczona na poziom **szkół podstawowych** ścieżka 0 zawiera głównie podstawy użytkowania aplikacji w środowisku Ubuntu, natomiast wprowadzona tam namiastka budowy kodów źródłowych w języku jawnym (po polsku), polegająca na układaniu skryptów w formie puzzli w środowisku graficznym Scratch S4A, służy najmłodszym uczniom wyłącznie do oprogramowywania modułu-interfejsu Arduino. Ze względu na bardzo wysokie walory edukacyjne środowiska Scratch, jest ono zalecane również dla starszych uczniów w niektórych modułach

Do mechatroniki potrzebny jest układ Arduino z mikrokontrolerem Atmega oraz

A, B i C. W przypadkach uczniów o szczególnych zdolnościach w podstawówkach, trener może podjąć z nimi próbę realizacji także wybranych modułów ze ścieżek A1, A2 i A3.

zestaw montażowy z podzespołami elektronicznymi.

Na poziom **szkół gimnazjalnych** zaleca się realizację wszystkich dość łatwych modułów ze ścieżki A. Środowiskiem projektowania są tam aplikacje do tworzenia grafiki komputerowej w systemie Linux z Ubuntu, a narzędziem programowania wizualno-skryptowego jest Scratch. Ta aplikacja narzędziowa, wzbogacona o bibliotekę S4A, służy także do oprogramowywania mikrokontrolera i sterowania podzespołami tworzącymi układ elektroniczny na płytce montażowej. Dla zaawansowanych gimnazjalistów zaleca się próbę realizacji modułów średnio trudnych B1 i B3, bądź też nieco trudniejszych B2 w zintegrowanym środowisku Lazarus, opartym na języku FreePascal.

Na poziom **szkół ponadgimnazjalnych** niesprofilowanych, czyli z wyłączeniem liceów i techników o profilu zbieżnym z infotechniką, polecana jest ścieżka z wybranymi blokami modułów B. Część z tych modułów dotyczy pracy w systemie Linux i w aplikacjach narzędziowych SRU, alternatywne moduły wykorzystują środowisko Scratch, a zasadniczym narzędziem programowania obiektowo-zdarzeniowego jest Lazarus z językiem FreePascal. Język ten (bez GUI) jest dopuszczony na egzaminie maturalnym, dlatego pisanie w nim kodów źródłowych przydaje się jako wprawka do matury z informatyki. Ponadto zarówno Scratch, jak i Lazarus dają implementacje przenośne na inne platformy systemowe. W bloku B są też alternatywne moduły B3 – jedne wykorzystują do oprogramowywania Arduino prosty interfejs IDE, a drugie wizualno-skryptowy interfejs S4A.

Dla **szkół sprofilowanych** infotechnicznie zaplanowano trzy alternatywne bloki modułów C – z językami programowania C, Phyton lub JavaScript. Ten ostatni powiązany jest z rozszerzaniem możliwości stron w HTML, tworzeniem arkuszy stylów CSS i wykorzystaniem biblioteki jQuery. Trudną ścieżkę programistyczną C można realizować dla uczniów, którzy mają w programie szkolnym dany język lub jako rozszerzenie w sytuacji, gdy w szkole taki język bądź treści nie są nauczane. Ze względu na to, że moduły z języków C i Python oparte są na stylu programowania imperatywnego, nie poleca się tych modułów dla szkół o innych profilach niż informatyczne (bądź pochodne). Natomiast z tego poziomu bloki mechatroniczne C3, choć dość trudne, to jednak są najbardziej interesujące, zatem można je polecić także do klas niesprofilowanych w liceach ogólnokształcących.

Prezentowany w poniższej tabeli **opis modułów** ma na celu uzyskanie wstępnej orientacji w zakresie tematyki modułów i realizowanych treści oraz założonych efektów merytorycznych. *Efektami merytorycznymi* nazywamy tu zbiór obserwowanych na kołach i ocenianych bezpośrednio wskaźników osiągania celów szczegółowych, dotyczących przedmiotowej wiedzy i umiejętności z zakresu szeroko rozumianej infotechniki. Są one zaliczane do grupy tzw. *efektów twardych*, przy czym pamiętać należy, że na kołach zainteresowań równie ważne jest formowanie kompetencji społecznych i osiąganie tzw. *efektów miękkich* (zmian świadomości, postaw, cech wolicjonalnych), co jest mierzone w procesie ewaluacji.

Paradoksalnie efekty merytoryczne, zwane „twardymi", szybciej się dezaktualizują i są mniej trwałe niż uformowane efekty „miękkie", oznaczające pożądane cechy osobowości.

Warto podkreślić, że sam proces dochodzenia do wiedzy i umiejętności na kołach zainteresowań powinien być postrzegany jako ważniejszy od określania precyzyjnych efektów kształcenia i nastawienia na ich osiąganie. Stąd tak ważna jest rola trenera – bacznego obserwatora pracy uczniów – dla którego najistotniejsze będzie podążanie za każdym indywidualnie uczniem i za jego potrzebami.

Treści dla szkół podstawowych

MODUŁY 0		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Wprowadzenie do środowiska Szkolnego Remiksu Ubuntu	Podstawowe informacje o komputerach i systemie Linux, operacje na aplikacjach i plikach; higiena pracy z komputerem.	• omawia pojęcia: *komputer, system, urządzenia peryferyjne;* • wykorzystuje podstawowe funkcje obsługi Linuksa; • rozpoznaje funkcje ikon związanych z oprogramowaniem; • korzysta z różnych sposobów bezpiecznego kopiowania plików i zakładania katalogów w Linuksie.
Album zdjęć – wspomnienia z wakacji	Wybór zdjęć do reportażu, wykadrowane i retusz w programie Gimp. Opisanie zdjęć i utworzenie albumu jako strony WWW w programie JAlbum.	• wskazuje i opisuje podstawowe sposoby i metody obróbki fotografii cyfrowej; • wykorzystuje podstawowe narzędzia do grafiki rastrowej; • dokonuje świadomej selekcji zdjęć i porządkuje zbiór fotografii cyfrowych w formie albumu zdjęć; • publikuje zdjęcia jako album na stronie WWW.
Tworzenie reklamy ośrodka wczasowego	Praca z edytorem graficznym TuxPaint i tekstowym LibreOffice Writer, tworzenie napisów w FontWork, łączenie grafiki z tekstem.	• omawia podstawowe pojęcia związane z edytorami tekstu i grafiki, a także formatami plików; • wykorzystuje podstawowe funkcje edytora tekstów; • wstawia do edytora tekstów elementy graficzne; • pozyskuje ozdobne czcionki i wykorzystuje FontWork.
Prezentacja danych do kampanii „Oszczędzaj energię elektryczną”	Tworzenie formuł w arkuszu kalkulacyjnym i diagramów słupkowych na podstawie danych o miesięcznym zużyciu prądu w rodzinach.	• omawia sposoby gromadzenia danych; • obsługuje arkusz kalkulacyjny Calc LibreOffice; • wpisuje poprawne formuły obliczeniowe; • generuje odpowiedni wykres kolumnowy; • prawidłowo formatuje wykres.
Prezentacja danych do kampanii „Zdrowy tryb życia”	Tworzenie formuł w arkuszu kalkulacyjnym i diagramów kołowych na podstawie danych z Ankiety nt.: Co pijesz na śniadanie?	• zbiera dane ankietowe i wprowadza je do odpowiedniej tabeli; • wpisuje formuły w arkuszu kalkulacyjnym Calc; • generuje wykresy z wykorzystaniem kreatora; • prawidłowo formatuje wykresy; • analizuje informacje zawarte na wykresie.

Wyszukiwanie informacji w sieci	Praca z przeglądarkami Internetu i serwisami wyszukującymi: Mozilla Firefox + Web Of Trust, Midori, Chromium, Google, IxQuick, Wikipedia	• obsługuje przeglądarkę WWW; • omawia pojęcia dotyczące sieci komputerowych, związane z wyszukiwaniem i śladami w Internecie; • potrafi wskazać i użyć różne wyszukiwarki; • odróżnia reklamę od wartościowych wyników wyszukiwania.
Programowa sygnalizacja diodą LED	Tworzenie kodu do włączania diody LED na płytce Arduino w środowisku wizualno-skryptowym Scratch S4A.	• omawia podstawowe pojęcia: *mikrokontroler, dioda elektroluminescencyjna*; • podłącza i obsługuje moduł-interfejs Arduino; • korzysta z funkcji obsługi wyjść w programie Scratch S4A; • tworzy kod włączający sygnalizację diodą LED.
Sterowanie diody LED z klawiatury	Tworzenie w środowisku Scratch S4A kodu do sterowania diodą za pomocą klawiatury.	• omawia pojęcia: *symbole matematyczne, sterowanie*; • montuje układ z wykorzystaniem płytki Arduino; • korzysta z funkcji obsługi wejść w programie Scratch S4A; • tworzy kod do sterowania diodą LED z klawiatury.
Losowy czas świecenia diody LED	Tworzenie w Scratch S4A kodu do losowego ustawiania czasu świecenia diody LED w Arduino.	• omawia pojęcia: *kod, zmienne, losowanie*; • wykorzystuje funkcje losowania z programu Scratch S4A; • deklaruje typy zmiennych, definiuje i przypisuje im wartości; • tworzy kod sterujący zmiennym świeceniem diody LED.
Wyświetlanie stanu przycisku Button	Tworzenie w Scratch S4A kodu ukazującego na ekranie tekst zależny do zmiany stanu na wejściu cyfrowym Arduino.	• omawia pojęcia: *button, opornik, wejście/wyjście cyfrowe*; • korzysta z funkcji obsługi *wej-wyj* programu Scratch S4A; • montuje i uruchamia układy na podstawie schematów; • tworzy kod do obsługi i wizualizacji stanu włącznika.
Przełączanie diody LED przyciskiem	Montowanie układu i tworzenie w Scratch S4A kodu sterującego diodą LED za pomocą przycisku.	• omawia pojęcia związane z przełączaniem stanu; • korzysta z funkcji obsługi *wej-wyj* programu Scratch S4A; • montuje układy na podstawie schematów i modyfikuje je; • tworzy kod do sterowania diodą LED poprzez button.
Programowa regulacja jasności diody LED	Wykorzystanie wyjścia PWM do pulsacyjnego sterowania diodą LED w środowisku Scratch S4A.	• omawia pojęcia związane ze zmianą szerokości impulsów; • korzysta z wyjścia PWM i funkcji programu Scratch S4A; • montuje i uruchamia układ na podstawie schematu; • tworzy kod do sterowania jasnością diody LED.

Treści dla szkół gimnazjalnych

MODUŁY A		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Wprowadzenie do środowiska SRU – poszukiwanie skarbu	Ukształtowanie wiedzy o Szkolnym Remiksie Ubuntu, rozwijanie umiejętności poruszania się w środowisku Linux, zdobycie wiedzy o licencjach Creative Commons (CC)	• omawia pojęcia: *dysk, katalog, plik*; • wykorzysta podstawowe funkcje obsługi Linuksa; • omawia licencje Creative Commons; • korzysta z różnych sposobów kopiowania plików, zakładania katalogów w Linuksie;

		• znajduje w Internecie ciekawy obrazek z zachowaniem licencjonowania; • znajduje ciekawe zdjęcie w Internecie i czcionkę w systemie do dalszego wykorzystania.
Tworzenie awatarów	Tworzenie obiektów wektorowych, porównanie grafiki rastrowej i wektorowej	• omawia podstawowe pojęcia grafiki wektorowej; • wskazuje różnice pomiędzy grafiką wektorową a rastrową; • wykorzystuje podstawowe narzędzia programu do grafiki wektorowej; • tworzy określone obiekty wektorowe (awatar, postać z gry komputerowej etc.); • rozpoznaje własne cechy osobowościowe i odzwierciedla je w awatarze.
Tworzenie graffiti	Tworzenie infografiki komputerowej na potrzeby graffiti. Wyszukiwanie i wykorzystanie czcionek oraz zdjęć na licencjach CC	• omawia podstawowe pojęcia grafiki rastrowej; • wskazuje różnice pomiędzy grafiką wektorową a rastrową; • wykorzystuje podstawowe narzędzia programu do obróbki graficznej; • tworzy określone obiekty wektorowe i łączy je z obrazem rastrowym.
Algorytm na mapie myśli	Praca z mapami myśli, zapoznanie z algorytmami i graficznymi formami ich reprezentacji	• tworzy cztery proste algorytmy w postaci przykładów wizualnych, • omawia na konkretnych przykładach pojęcia: *algorytm, instrukcja warunkowa, pętla, system pozycyjny binarny a dziesiętny, sortowanie bąbelkowe*, • wykorzystuje podstawowe funkcje programu FreeMind i Dia, • angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.
Stworzenie animacji biedronki poruszającej się po linii	Poznanie zasad tworzenia programów komputerowych i animacji za pomocą instrukcji języka programowania	• omawia podstawowe pojęcia algorytmiczne: *skrypt, wyrażenia, algorytm, pętla, wyrażenie warunkowe, zmienna, czujnik, instrukcja*; • wykorzystuje podstawowe narzędzia programu Scratch; • tworzy animację interaktywną; • współpracuje z innymi uczniami oraz z nauczycielem.
Moja własna gra komputerowa	Stworzenie własnej animacji oraz algorytmu wyszukiwania binarnego w programie Scratch	• omawia podstawowe pojęcia *algorytmiczne: skrypt, wyrażenia, algorytm, pętla, wyrażenie warunkowe, zmienna, czujnik, instrukcja*; • wykorzystuje podstawowe narzędzia programu Scratch; • tworzy animację interaktywną; • współpracuje z innymi uczniami oraz z nauczycielem.
Labirynt interaktywny	Stworzenie interaktywnej gry w programie Scratch ze skryptami obsługującymi animację obiektów	• omawia podstawowe pojęcia, takie jak: *pętla, wyrażenie warunkowe, zmienna*; • wykorzystuje podstawowe narzędzia programu Scratch; • korzysta z wcześniej przygotowanego obrazu (labiryntu) w przygotowywanej pracy; • tworzy animację interaktywną; • współpracuje z innymi uczennicami i uczniami oraz z nauczycielem.

Systemy liczbowe – zasady konwersji liczb	Stworzenie interaktywnej prezentacji, pokazującej jak liczone są wartości między różnymi pozycyjnymi systemami liczbowymi, ze szczególnym uwzględnieniem systemu binarnego.	• omawia pojęcia: *podstawa, pozycja, cyfra i liczba, pętla, instrukcja warunkowa, zmienna* • dostrzega, w jaki sposób liczby są prezentowane wewnątrz współczesnych maszyn cyfrowych • omawia różnice i podobieństwa między systemem binarnym a dziesiętnym • tworzy interaktywną prezentację z zakresu liczenia wartości w różnych liczbowych systemach pozycyjnych.
Środowisko Scratch z obsługą modułu-interfejsu	Wprowadzenie w świat mikrokontrolerów na przykładzie modułu-interfejsu Arduino oraz jego obsługa w środowisku Scratch for Arduino (S4A)	• omawia pojęcia: *button, wejście cyfrowe; dioda LED; opornik; pętla; wyrażenie warunkowe; zmienna; mikrokontroler; port USB i RS232*; • montuje i uruchamia przykładowe układy na podstawie schematów; • deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • obsługuje środowisko Scratch S4A i zna jego funkcje; • wykorzystuje podstawowe narzędzia programu Scratch S4A; • prawidłowo podłącza i steruje podzespołami: dioda LED, dioda RGB, przycisk; • angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.
Pomiar temperatury i wizualizacja za pomocą diody RGB	Budowa układu do wizualizacji regulacji jasności i kolorów diody RGB. Pomiar temperatury i prezentacja pomiarów przy wykorzystaniu trzech kolorów diody.	• prawidłowo obsługuje środowisko programowania graficznego Scratch S4A; • samodzielnie montuje i uruchamia przykładowe układy na podstawie schematów; • deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • prawidłowo łączy i odczytuje wskazania czujnika temperatury; • trafnie używa zwrotów: *czujnik, przetwornik A/D, dioda RGB, buzzer, modulacja szerokości impulsu, wejście analogowe*; • angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.
Odczytywanie stanu czujników przez układ pomiarowy z fotorezystorem	Konstruowanie i oprogramowanie układów do odczytu stanu czujników na przykładzie interfejsów z układem Arduino do pomiaru natężenia światła	• zgodnie z zasadami działania podłącza czujnik pomiarowy: fotorezystor; • prawidłowo buduje i oprogramowuje moduł-interfejs wskazujący poziomy oświetlenia; • uruchamia ukazywanie odczytów w środowisku Linux; • modyfikuje i rozbudowuje pomiarowe układy elektroniczne oraz kody źródłowe; • trafnie używa sformułowań: *czujnik, czułość, wejście analogowe, przetwornik A/D.*
Interakcja modułu-interfejsu ze środowiskiem S4A	Budowa interfejsu z wykorzystaniem zestawu Arduino i programu w środowisku Scratch S4A w celu stworzenia interaktywnej gry ping-pong.	• prawidłowo obsługuje środowisko programowania graficznego Scratch S4A; • samodzielnie montuje i uruchamia przykładowe układy na podstawie schematów; • deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • programuje mikrokontroler Atmega oraz steruje nim z poziomu komputera; • prawidłowo łączy i steruje przyciskami z zestawu modułu-interfejsu; • trafnie używa zwrotów: *button, wejście cyfrowe*; • angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.

Treści dla szkół ponadgimnazjalnych niesprofilowanych

MODUŁY B – programowanie		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Szyfrowanie – jak chronić ważne informacje	Zaawansowane użytkowanie Linuksa, podstawowe informacje o bezpieczeństwie danych i szyfrowaniu	• wskaże wagę bezpieczeństwa danych oraz zalety podpisu elektronicznego i szyfrowania; • omawia sposób zdobywania informacji z manuali systemowych; • omawia obsługę konsoli; • wykorzystuje podstawowe funkcje obsługi Linuksa, aplikacji gpg, Kgpg; • tworzy własne klucze pgp, obsługuje gpg z linii poleceń.
Wizualizacja instrukcji warunkowej – „Wybór drogi”	Prezentacja za pomocą Scratcha sposobu podejmowania decyzji przez program wykonujący instrukcję warunkową.	• omawia pojęcia: *instrukcja warunkowa, zmienna, algorytm*; • wykorzystuje Scratch do stworzenia wizualizacji działania instrukcji warunkowej.
Wizualizacja działania pętli – „Zakręcona mrówka”	Prezentacja za pomocą Scratcha zasady realizacji powtórzeń przez program wykonujący instrukcję pętli.	• omawia pojęcia: *iteracja, instrukcja pętli, procedura*; • wykorzystuje Scratcha do stworzenia wizualizacji działania pętli.
Wizualizacja operatorów przypisania – „Magiczny operator”	Prezentacja funkcji operatora przypisania wartości do zmiennej w programie Scratch.	• omawia pojęcia: *zmienna, zmienna pusta, deklaracja zmiennej, przypisanie, wyrażenie, ewaluacja*; • wskazuje, w jaki sposób maszyny wyliczają wartości wyrażeń; • wykorzystuje Scratcha do stworzenia wizualizacji przypisania wartości.
Ewaluacja prawej strony wyrażeń – „Zamieszanie z kolorami”	Prezentacja metody ewaluacji prawej strony wyrażenia na przykładzie mieszania kolorów.	• omawia pojęcia: *przypisanie, wyrażenie, ewaluacja*; • wskazuje, w jaki sposób program sprawdzają wartości wyrażeń; • wykorzystuje Scratcha do przedstawienia zasady ewaluacji.
Gra w „Kółko i krzyżyk” ze strategią blokowania przeciwnika	Odnalezienie w istniejącym kodzie miejsca, które należy rozbudować o sprawdzanie istnienia kombinacji figur w grze „Kółko i krzyżyk”, które w następnym ruchu pozwolą przeciwnikowi na wygranie i zablokowanie ruchu przeciwnika	• posługuje się naturalnym językiem przy opisie algorytmów • wskazuje miejsca w algorytmie realizujące dane zadania; • poprzez dokonanie analizy algorytmu proponuje rozwiązanie rozszerzające jego działanie o założone funkcje; • prezentuje dane w macierzy dwuwymiarowej za pomocą liniowej adresacji, wykorzystuje tablice; • dzieli algorytm na podprogramy; • realizuje grę z interfejsem graficznym oraz algorytmem komputerowego przeciwnika
Gra Pong o zmiennej strategii	Zmiany w skryptach gry Pong, dodające kolejne elementy i skrypty zmieniające jej zasady	• analizuje kod i tłumaczy zasadę działania programu; • posługuje się naturalnym językiem przy opisie algorytmów; • wskazuje miejsca w kodzie realizujące poszczególne zadania; • omawia podstawowe pojęcia: *zmienna globalna, zmienna lokalna, lista, kolejka*; • prezentuje dane w strukturach typu lista i kolejka danych; • wskazuje podprogramy w algorytmie; • programuje skrypty obsługujące animację obiektów; • prawidłowo implementuje zmianę w grze.

Wybrane algorytmy sortowania	Wizualizacja sposobu sortowania elementów zbioru w porządku rosnącym, na przykładzie zbioru liczbowego o zmiennej liczbie elementów. Związek ilości kroków algorytmu z liczbą elementów w zbiorze.	• omawia podstawowe pojęcia: *algorytm, sortowanie*; • charakteryzuje sposób działania algorytmów: *bubble sort, inserting sort* oraz *selection sort* oraz uporządkuje 5-elementowy zbiór tymi metodami; • narysuje schemat blokowy algorytmu realizującego sortowanie bąbelkowe; • analizuje ilość kroków algorytmu w zależności od liczby elementów zbioru; • zaimplementuje algorytmy w języku Scratch.
Wprowadzenie do środowiska Lazarus – „Kółko i krzyżyk"	Prezentacja zintegrowanego, graficznego środowiska programowania Lazarus. Tworzenie prostego programu w języku FreePascal.	• wyjaśnia zasadę programowania obiektowo-zdarzeniowego; • określa, do czego służą składowe środowiska Lazarus: *Inspektor obiektów, Właściwości, Zdarzenia, Favorites, Komponenty, Komunikaty, Edytor źródeł - Kod programu, Menu Edytora, Formularz*; • prawidłowo porusza się w środowisku Lazarus; • stosuje podstawowe konstrukcje języka FreePascal; • wskazuje korzyści płynące z umiejętności programowania
Gra w zapalane światełka	Prosta gra w światełka – praca w Lazarusie. Przenoszenie stanów poprzedzających do parametrów bieżących.	• analizuje kod i tłumaczy zasadę działania programu; • programuje w stopniu pozwalającym na zaimplementowanie większej planszy gry; • programuje w stopniu pozwalającym na uzupełnienie brakujących fragmentów kodu • tworzy program, w którym następny krok zależy od bieżącego stanu gry
Automaty – „Gra w życie"	Implementacja „Gry w życie" jako przykład prostej symulacji opartej na automatach	• omawia, czym są automaty i do czego służą • określa zasady działania automatów • konstruuje prosty automat
Zegar binarny	Przedstawienie zasady działania zegara decybinarnego	• omawia zasady działania zegara opartego na systemie binarnym • odczytuje i zapisuje cyfry przy użyciu systemu binarnego • przekształca zapis z systemu dziesiętnego na dwójkowy • wyjaśnia zasadę funkcjonowania zegara binarnego • omawia działanie operatora logicznego *and* i *or* na zmiennych binarnych
Losowanie i porównywanie – gra „Kamień-nożyce-papier"	Realizacja gry losowej, polegającej na równoczesnym wyborze dwóch przedmiotów, z których jeden wygrywa według zasady gry „Kamień-nożyce-papier"	• obsługuje zintegrowane środowisko programowania wspieranego interfejsem graficznym; • tworzy potrzebne obiekty za pomocą GUI i odpowiednio modyfikuje ich właściwości; • nazywa główne bloki struktury kodu źródłowego i elementarne instrukcje języka FreePascal; • trafnie operacjonalizuje i objaśnia pojęcia: *obiekty, atrybuty, kod źródłowy, instrukcje*; • prawidłowo wprowadza kod źródłowy i doprowadza do pełnego działania implementacji; • próbuje wygrać z implementacją sztucznej inteligencji.

Rozrzucanie i porządkowanie – gra „Układanka”	Realizacja gry logicznej typu puzzle, polegającej na porządkowaniu 24 liter alfabetu (A÷X) ułożonych w tablicy 5x5, poprzez przemieszczanie liter z pól stycznych do pola pustego.	• trafnie operacjonalizuje i objaśnia pojęcia: *kody liter, tablica, menu, obsługa zdarzeń*; • tworzy foremną macierz widżetów TPanel i modyfikuje ich atrybuty z poziomu kodu; • prawidłowo uzupełnia fragmenty kodu źródłowego i doprowadza do działania implementacji; • samodzielnie stosuje w praktyce strategię porządkowania w układance alfabetycznej; • chętnie rozwiązuje zadanie uporządkowania całej macierzy o rozmiarach 5x5.
Odkrywanie i stosowanie algorytmu – gra „Wieże Hanoi”	Wizualizacja strategii wygranej w grze decyzyjnej, polegającej na przenoszeniu obiektów o różnej wielkości między 3 cokołami tak, aby obiektu większego nie stawiać na mniejszym.	• trafnie operacjonalizuje i objaśnia pojęcia: *strategia, algorytm, animacja, wizualizacja*; • wpisuje właściwe parametry umiejscawiające obiekty na ekranie w odpowiednim miejscu; • prawidłowo uzupełnia fragmenty kodu źródłowego i doprowadza do działania implementacji; • opisuje werbalnie i optymalnie realizuje strategię wygranej w łamigłówce „Wieże Hanoi”; • chętnie rozwiązuje utrudnione zadanie uporządkowania z pozycji pośredniej nieregularnej.
Namiastka sztucznej inteligencja – gra NIM	Realizacja gry logicznej NIM, w której chodzi o to, aby podczas naprzemiennego pobierania obiektów z jednego rzędu na planszy uniknąć konieczności zabrania obiektu ostatniego. Losowane są różne układy do 10 obiektów w każdym z trzech rzędów. Komputer ma zaprogramowaną strategię wygranej, dlatego rozpoczyna gracz.	• trafnie operacjonalizuje i objaśnia pojęcia: *warunki, operatory logiczne, sterowanie, indeksy*; • z poziomu kodu prawidłowo tworzy obiekty TImage lub modyfikuje ich atrybuty; • prawidłowo uzupełnia fragmenty kodu źródłowego i doprowadza do działania implementacji; • opisuje werbalnie i optymalnie realizuje strategię wygranej w końcowej fazie gry NIM; • chętnie stosuje w praktyce strategię wygranej poprzez analizę parzystości grup binarnych.

MODUŁY B – mechatronika		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Funkcjonalność modułu-interfejsu. Środowisko mikrokontrolera	Zastosowanie modułu-interfejsu Arduino oraz obsługa interaktywnego terminala Arduino IDE, służącego do programowania mikrokontrolera. Zestawianie połączeń na podstawie instrukcji montażu układów. Zaimplementowanie kodu do wyświetlania tekstów oraz do sterowania diodą wbudowaną w moduł-interfejs.	• trafnie objaśnia pojęcia: *mikrokontroler; port USB; dioda LED; button; opornik*; • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje elementy kodu do modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • prawidłowo podłącza diody LED oraz RGB; • steruje diodami i modyfikuje treść wyświetlanych komunikatów.

Sterowanie elementami wykonawczymi z wykorzystaniem czujnika światła	Wizualizacja działania elementów modułu-interfejsu. Wykorzystanie funkcji przetwornika analogowo-cyfrowego do układów pomiarowych. Istota funkcjonowania fotorezystora. Konstruowanie i oprogramowanie układów do odczytu stanu czujnika na przykładzie interfejsu do pomiaru natężenia światła. Prezentacja wyników z wykorzystaniem diody RGB.	• zgodnie z zasadami działania podłącza czujnik pomiarowy: fotorezystor; • prawidłowo buduje i oprogramowuje moduł-interfejs wskazujący poziomy oświetlenia; • uruchamia ukazywanie odczytów na wyświetlaczu LCD lub w środowisku Linux; • modyfikuje i rozbudowuje pomiarowe układy elektroniczne oraz kody źródłowe; • trafnie używa sformułowań: *czujnik, wejście analogowe, przetwornik A/D*.
Pomiar temperatury i obsługa wyświetlacza LCD 2x16	Wizualizacja działania zestawu modułu-interfejsu z układem Arduino. Wykorzystanie funkcji przetwornika analogowo-cyfrowego do budowy układu pomiarowego. Istota funkcjonowania i zastosowania termistora. Podłączenie i sterowanie LCD do wyświetlania tekstów. Konstruowanie i oprogramowanie układu do odczytu stanu czujnika. Prezentacja odczytu temperatury i skrajnych wartości na LCD oraz na ekranie monitora.	• zgodnie z zasadami działania podłącza czujnik pomiarowy termistor; • prawidłowo buduje i oprogramowuje moduł-interfejs służący do pomiaru temperatury; • uruchamia ukazywanie odczytów na wyświetlaczu LCD lub w środowisku Linux; • modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy; • dokonuje przeliczenia wartości pomiarów temperatury do innych skal; • trafnie używa sformułowań: *czujnik, stopnie Celsjusza, stopnie Fahrenheita, Kelvin, czułość, wejście analogowe, przetwornik A/D*.
Interakcja człowieka z modułem-interfejsem	Budowa układu i programu do symulacji losowania jednej z sześciu liczb jak w kostce do gry. Prezentacja wyniku losowania z wykorzystaniem diod LED. Poszerzanie wiedzy o zastosowaniu modułu-interfejsu poprzez rozbudowę o kolejne elementy. Efektem jest opracowanie implementacji pozwalającej ćwiczyć pamięć i zręczność.	• samodzielnie projektuje i oprogramowuje układy na platformie Arduino; • implementuje układy interfejsu z diodami LED i przyciskami button; • mechanicznie i programowo steruje wejściami i wyjściami cyfrowymi; • implementuje grę zręcznościowo-pamięciową z losowaniem liczb; • prawidłowo analizuje i pisze objaśnienia kodu źródłowego implementacji; • rozbudowuje moduł-interfejs, dodając wyświetlacz LCD i buzzer; • trafnie używa sformułowań: *PULLUP, wejście/wyjście cyfrowe, typ logiczny (boolean)*.
Możliwości środowiska Scratch S4A i modułu-interfejsu Arduino	Wprowadzenie w świat mikrokontrolerów na przykładzie modułu-interfejsu Arduino oraz jego obsługa w środowisku Scratch (S4A). Prezentacja i wyjaśnienie sposobu zestawiania połączeń na podstawie instrukcji ilustrującej montaż układów.	• omawia pojęcia: *button, wejście cyfrowe; dioda RGB; opornik; pętla; wyrażenie warunkowe; zmienna; mikrokontroler; port USB i RS232*; • montuje i uruchamia przykładowe układy na podstawie schematów; • deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • obsługuje środowisko Scratch S4A, wykorzystując jego funkcje; • używa podstawowych narzędzi programu Scratch S4A; • omawia istotę działania oraz sposób podłączania i sterowania podzespołami: dioda RGB, przycisk; • angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem;

Działanie fotorezystora i potencjometru – przetwornik analogowo-cyfrowy	Istota funkcjonowania i zastosowania fotorezystora i potencjometru. Konstruowanie i oprogramowanie układów do odczytu stanu potencjometru i wartości fotorezystora	• zgodnie z zasadami działania podłącza czujnik pomiarowy: fotorezystor; • prawidłowo buduje i oprogramowuje moduł-interfejs wskazujący odczyty z wejścia analogowego; • modyfikuje i rozbudowuje pomiarowe układy elektroniczne oraz kody źródłowe; • trafnie używa sformułowań: *czujnik, czułość, wejście analogowe, przetwornik A/D*; • wskazuje zastosowania modułów-interfejsów
Środowisko mikrokontrolera – rozbudowa funkcjonalności	Podłączenie i sterowanie diodami LED na przykładzie sygnalizacji świetlnej. Obsługa przycisków i sterowanie dziękowe buzzera. Zaimplementowanie kodu do sterowania diodą.	• trafnie objaśnia pojęcia: *mikrokontroler; port USB; dioda LED; button; opornik; buzzer*; • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • prawidłowo podłącza diodę LED i steruje jasnością diody; • modyfikuje treść wyświetlanych komunikatów;
Termometr cyfrowy – obsługa wyświetlacza	Podłączenie i sterowanie wyświetlaczem LCD z wykorzystaniem płytki stykowej. Zaimplementowanie kodu do wyświetlania tekstów.	• zgodnie z zasadami działania podłącza czujnik pomiarowy termistor; • prawidłowo buduje i oprogramowuje moduł-interfejs służący do pomiaru temperatury; • uruchamia ukazywanie odczytów na wyświetlaczu LCD lub w środowisku Linux; • modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy; • przedstawia skale i jednostki temperatury oraz omawia zależności między nimi • prawidłowo dokonuje przeliczenia wartości pomiarów temperatury do innych skal; • trafnie używa sformułowań: *czujnik, stopnie Celsjusza, stopnie Fahrenheita, Kelvin, czułość, wejście analogowe, przetwornik A/D*;

Treści dla szkół sprofilowanych infotechnicznie

MODUŁY C – programowanie, język C		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Struktura programu, zmienne oraz typy danych	Poznanie struktury programu, pojęcia zmiennych oraz podstawowych typów danych, jakie zmienne mogą przybierać w języku C.	• omawia podstawowe pojęcia programowania: *zmienna, typy zmiennych* • omawia strukturę programu w języku C • pisze prosty program zawierający kilka zmiennych, który kompiluje się bez błędów oraz poprawnie działa • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux

Wyrażenia warunkowe *if – else*	Poznanie struktury oraz zastosowania wyrażeń warunkowych *if – else* w języku C	• omawia podstawowe pojęcia programowania: *wyrażenia warunkowe if – else* i wskazuje, do czego służą • pisze prosty program zawierający *wyrażenia warunkowe if – else* i zachowujący się różnie, zależnie od wprowadzonych danych • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux
Wyrażenie warunkowe *switch{ case: . . .;}*	Poznanie struktury oraz zastosowania wyrażenia warunkowego *switch {case: . . .;}* w języku C.	• omawia podstawowe pojęcia programowania: *wyrażenia warunkowe switch {case: . . .;}* i wskazuje, do czego służą • pisze prosty program zawierający wyrażenie warunkowe *switch {case: . . .;}* i zachowujący się różnie, zależnie od wprowadzonych danych • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux
Pętle – *while()* i *do {} while()*	Poznanie, czym są pętle i jak korzystać z pętli *while* oraz *do{} while()* w języku C.	• omawia podstawowe pojęcia programowania: *pętle – while()* i *do {} while()* i wskazuje, do czego służą • pisze prosty program korzystający z pętli do wykonywania powtarzających się czynności • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux
Tablice i pętla *for()*	Poznanie, czym są tablice oraz pętla *for()* w języku C	• omawia podstawowe pojęcia programowania: *tablice, pętle for()* i wskazuje, do czego służą • pisze prosty program korzystający z tablic oraz wykorzystuje pętlę *for()* do operowania na nich • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux
Funkcje	Tworzenie oraz używanie funkcji w języku C.	• wyjaśnia, czym są funkcje i kiedy kod należy dzielić na mniejsze funkcje, a także jak się tworzy funkcje oraz jak ich używać • pisze prosty program podzielony na funkcje realizujące pojedyncze zadania • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux
Struktury i unie	Tworzenie oraz praktyczne korzystanie ze struktur oraz unii w języku C	• wyjaśnia, czym są: *struktury i unie* oraz dlaczego są ważne i potrzebne w programowaniu • wyjaśnia, jak się deklaruje struktury i unie oraz do czego się ich używa • pisze program korzystający ze struktur porządkowania danych • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux
Wskaźniki	Poznanie zasad funkcjonowania pamięci w programach oraz wskaźników jako metody bezpośredniego dostępu do niej	• wyjaśnia, jak działa pamięć, jak dane są w niej umieszczane oraz jak można się do nich dostać, używając wskaźników • omawia, jak zarządzana jest pamięć oraz jak zmienne są w niej umieszczane • wyjaśnia, jak się deklaruje i na różne sposoby używa wskaźników • deklaruje i wykorzystuje wskaźniki • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux

MODUŁY C – mechatronika, wersja I		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Budowa i programowanie modułu-interfejsu	Zastosowanie modułu-interfejsu Arduino oraz obsługa interaktywnego terminala Arduino IDE, służącego do programowania mikrokontrolera.	• trafnie objaśnia pojęcia: *mikrokontroler; port USB; dioda LED; button; opornik; potencjometr, modulacja szerokości impulsu – PWM* • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • przesyła wyniki z układu do komputera; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • prawidłowo podłącza diodę LED i steruje jasnością diody; • modyfikuje treść wyświetlanych komunikatów;
Tranzystor NPN	Zastosowanie tranzystora do sterowania pięcioma diodami LED połączonymi równolegle. Wykorzystanie modulacji szerokości impulsów do sterowania jasnością świecenia diod.	• trafnie objaśnia pojęcia: *tranzystor NPN, tranzystor PNP, button, modulacja szerokości imppulsu* • wymienia rodzaje tranzystorów, podaje parametry i rozpoznaje oznaczenia wyprowadzeń; • wykorzystuje tranzystor jako przełącznik; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • prawidłowo wykonuje pomiary i przelicza wartości napięcia, prądu i oporności; • trafnie używa słowa: *klucz tranzystorowy, stan nasycenia i zatkania, Volt, Amper*;
Pole magnetyczne	Wykorzystanie efektu Halla do sygnalizacji przy pomocy diody RGB oraz występowania pola magnetycznego. Zastosowanie, budowa i działanie hallotronu.	• trafnie objaśnia pojęcia: *pole magnetyczne, histereza, półprzewodnik, dioda RGB, efekt Halla*; • objaśnia zasadę działania hallotronu i prawidłowo stosuje go w układach; • podaje parametry i rozpoznaje oznaczenia wyprowadzeń; • prawidłowo rozbudowuje układy o dodatkowe, niezbędne do pracy elementy; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
Zegar	Stworzenie prostego stopera i zegara przy użyciu modułu-interfejsu. Wykorzystanie podstawowych funkcji do sterowania i prezentacji czasu na wyświetlaczu LCD oraz na ekranie monitora. Zasada działania i używania bibliotek.	• prawidłowo buduje i oprogramowuje moduł-interfejs wskazujący aktualny czas; • prawidłowo buduje i oprogramowuje moduł-interfejs służący do pomiaru czasu; • uruchamia ukazywanie teksu na wyświetlaczu LCD; • modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy; • trafnie używa sformułowań: *czas, sekunda, milisekunda, opóźnienie, pętle, LCD, biblioteka*;

MODUŁY C – programowanie, język Python		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Zmienne, typy i pobieranie danych	Poznanie pojęcia zmiennych, ich typów oraz sposoby wczytywania danych z klawiatury w języku Python	• omawia podstawowe pojęcia programowania: *zmienna, interpreter* • pisze prosty program zawierający kilka zmiennych oraz pobiera ich wartości od użytkownika • wykorzystuje interpreter do interaktywnego testowania kawałków kodu przed włączeniem ich do programu • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python
Wyrażenia warunkowe	Poznanie wyrażeń warunkowych *if – elif – else* w języku Python	• omawia podstawowe pojęcia programowania: *wyrażenia warunkowe, struktura wyrażeń warunkowych* oraz docenia ich istotność w programowaniu • pisze programy, które proszą użytkownika o wprowadzenie danych oraz podejmują odpowiednią akcję • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python
Pętle – *while*	Poznanie pętli *while()* w języku Python	• omawia podstawowe pojęcia programowania: *pętla while* • pisze programy, które powtarzają pewne czynności określoną liczbę razy • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python
Zaawansowane typy danych	Poznanie zaawansowanych typów danych oraz nauczenie się, do czego i jak je wykorzystywać w języku Python	• omawia zaawansowane typy danych i ich przydatność w programowaniu • pisze programy, które korzystają z list, krotek i słowników oraz wie, czym są zbiory i jak ich używać • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python
Pętle – *for*	Poznanie innego rodzaju pętli, jaką jest pętla for w języku Python	• objaśnia podstawowe pojęcia programowania: *pętla for*, omawia różnice pomiędzy pętlami • pisze programy, które używają pętli for do operacji na poszczególnych elementach większych zbiorów • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python
Funkcje	Poznanie tego, jak w Pythonie tworzyć własne funkcje oraz kiedy należy program dzielić na funkcje	• omawia podstawowe pojęcia programowania: *funkcje* • pisze kod realizujący własne funkcje • dzieli długie programy na funkcje, aby były czytelniejsze i efektywniejsze • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python
Wyjątki	Poznanie tego, czym są wyjątki i jak ich używać w języku Python	• omawia, czym są *wyjątki* i jak pomagają programiście • przewiduje, kiedy program może wyrzucić wyjątek • programowo wyłapuje i obsługuje wyjątek w programie • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python

Klasy	Poznanie tego, czym są klasy i jak dzięki nim tworzyć efektywniejsze programy w języku Python	• omawia podstawowe pojęcia programowania: *klasy, programowanie obiektowe* • uzasadnia używanie klas dla ułatwienia pisania dużych programów • rozumie paradygmat programowania obiektowego oraz pisze programy korzystające z niego • tworzy własne klasy i korzysta z gotowych bibliotek Phytona • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python

MODUŁY C – mechatronika, wersja II		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Sterowanie diodami – wykorzystanie wejścia analogowego	Zastosowanie modułu-interfejsu Arduino oraz obsługa interaktywnego terminala Arduino IDE, służącego do programowania mikrokontrolera.	• trafnie objaśnia pojęcia: *mikrokontroler; port USB; dioda LED; button; opornik; czujnik nachylenia* • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • prawidłowo podłącza diodę LED i steruje jasnością diody; • modyfikuje parametry wyświetlanych komunikatów;
Pomiar wartości opornika	Zastosowanie modułu-interfejsu Arduino jako narzędzia do pomocy w oszacowaniu wartości oporników. Budowa i sposoby rozpoznawania oznaczeń oporników. Oszacowanie wartości rezystorów połączonych szeregowo i równolegle.	• trafnie objaśnia pojęcia: *dzielnik napięcia, opornik*, • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • stosuje metodę przesyłania wyników z układu do komputera; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • odczytuje wartość opornika na podstawie kodu barwnego lub z użyciem modułu-interfejsu; • oblicza wartość rezystancji w przypadku łączenia rezystorów równolegle lub szeregowo • dokonuje przekształceń wzoru prawa Ohma, aby wyznaczyć wartość rezystancji;

Kontrola diody RGB	Komunikacja modułu-interfejsu z komputerem PC na przykładzie sterowania jasnością i kolorami diody RGB	• trafnie objaśnia pojęcia: *półprzewodnik, dioda RGB, port szeregowy, komunikacja, transmisja danych*; • podaje parametry i rozpoznaje oznaczenia wyprowadzeń diody RGB; • prawidłowo rozbudowuje układy o dodatkowe, niezbędne do pracy elementy; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
Prezentacja temperatury z wykorzystaniem diody RGB i buzzera	Zastosowanie modułu-interfejsu Arduino do odczytu temperatury. Przełożenie odczytów wartości z wejścia analogowego na sterowanie jasnością i barwą diody RGB. Definiowanie wartości progowych temperatury i sygnalizowanie alarmem w przypadku ich przekroczenia.	• zgodnie z zasadami działania podłącza czujnik pomiarowy termistor; buzzer, diodę RGB; • prawidłowo buduje i oprogramowuje moduł-interfejs służący do pomiaru temperatury; • uruchamia ukazywanie odczytów w środowisku Linux; • modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy; • dokonuje przeliczenia wartości pomiarów temperatury do innych skal; • trafnie używa sformułowań: *czujnik, stopnie Celsjusza, Fahrenheita i Kelvina, czułość, wejście analogowe, przetwornik A/D*;

MODUŁY C – HTML i JavaScript		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Podstawy HTML	Tworzenie szkieletu strony internetowej, zawierającej nagłówek, stopkę, nawigację i treść. Walidacja prawidłowości kodu	• objaśnia zastosowanie podstawowych tagów HTML i sposobów rozmieszczania ich na stronie; • tworzy prosty dokument HTML i dopasowuje jego wygląd; • weryfikuje poprawność dokumentu, korzystając z jednego z walidatorów • analizuje strukturę dokumentu, korzystając z narzędzi dostępnych w przeglądarce
CSS i box model	Poznanie podstaw CSS. Planowanie typografii i modyfikacja rozmieszczeń elementów na stronie.	• rozmieszcza elementy HTML na stronie za pomocą CSS; • dopasowuje wygląd dokumentu HTML, korzystając z pliku CSS; • swobodnie rozmieszcza elementy dokumentu na stronie za pomocą różnych technik; • analizuje strukturę dokumentu, korzystając z narzędzi dostępnych w przeglądarce.
Formularze HTML	Tworzenie dokumentu z formularzem umożliwiającym wprowadzanie informacji przez użytkownika, z wykorzystaniem odpowiednich elementów w zależności od rodzaju informacji	• objaśnia zastosowanie tagów HTML służących do budowania formularzy • tworzy rozbudowany formularz HTML • dobiera odpowiednie elementy formularza do typu danych wprowadzanych przez użytkownika • weryfikuje poprawność stworzonego dokumentu HTML

JavaScript	Poznanie podstawy korzystania z języka JavaScript oraz stworzenie implementacji wykorzystującej poznane elementy języka	• dokonuje operacji na zmiennych • tworzy proste struktury danych • definiuje, przekazuje i uruchamia funkcje • tworzy prostą aplikację dokonującą interakcji z użytkownikiem
Podstawy jQuery	Napisanie programu tak, żeby komunikował się z użytkownikiem poprzez formularz i dokument HTML	• wyszukuje dowolne elementy HTML z wykorzystaniem selektorów CSS • pobiera informacje wprowadzone do formularza HTML • modyfikuje dokument HTML poprzez zmianę i dodawanie nowych elementów
jQuery – zdarzenia i efekty	Dodatkowe informacje o tym, jak reagować na różne operacje dokonywane przez użytkownika oraz prezentować wynik działań poprzez zmiany efektów w dokumencie HTML	• dodaje do aplikacji obsługę zdarzeń • tworzy obsługę zdarzenia dla elementów tworzonych dynamicznie przez aplikację • modyfikuje dokument HTML, dokonując zmian z drobnymi animacjami • łączy animacje w sekwencje następujące jedna po drugiej • korzysta z efektów jQuery do modyfikowania dokumentu HTML
Przechowywanie danych	Uzupełnienie aplikacji ToDo o możliwość trwałego zapisania wprowadzonych informacji i ich odczytu przy ponownym uruchomieniu strony	• wymienia sposoby na trwały zapis danych oraz omawia ich wady i zalety • zapisuje informacje na czas działania sesji przeglądarki • trwale zapisuje informacje, niezależnie od czasu działania sesji • pobiera i modyfikuje całość lub część przechowywanych danych • dobiera najlepsze narzędzie w zależności od typu informacji do przechowania
Canvas	Tworzenie implementacji pozwalającej na rysowanie prostych figur geometrycznych	• wskazuje sposoby tworzenia animacji obiektów i omawia różnice między nimi • tworzy implementację i rysuje proste figury geometryczne z wykorzystaniem różnych stylów • tworzy interfejs właściwie reagujący na akcje wykonywane przez użytkownika z użyciem myszki

MODUŁY C – mechatronika, wersja III

Efekty – wskaźniki osiągnięcia celów

Tematyka modułów	Opis realizowanych treści	Uczeń:
Wykorzystanie wejścia analogowego do sterowania diodami	Zastosowanie modułu-interfejsu Arduino oraz obsługa interaktywnego terminala Arduino IDE, służącego do programowania mikrokontrolera.	• trafnie objaśnia pojęcia: *mikrokontroler; port USB; dioda elektroluminescencyjna; button; opornik; potencjometr,* • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • stosuje metodę przesłania wyników z układu do komputera; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • steruje diodą elektroluminescencyjną oraz modyfikuje treść wyświetlanych komunikatów;

Protokół komunikacyjny *1-wire*	Budowa układu i programu do odczytu danych, wykorzystując interfejs 1-wire na przykładzie czujnika Dallas DS18B20. Rozszerzenie wiedzy dotyczącej adresowania czujników ich sposobów zasilania i wykorzystywania bibliotek w celu sterowania.	• zgodnie z zasadami działania podłącza czujnik pomiarowy; • prawidłowo buduje i oprogramowuje moduł-interfejs służący do pomiaru temperatury; • uruchamia ukazywanie odczytów na wyświetlaczu LCD lub w środowisku Linux; • modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy; • przedstawia skale i jednostki temperatury oraz zależności między nimi; • dokonuje odczytu adresu czujników; • trafnie używa sformułowań: *czujnik, stopnie Celsjusza, 1-wire, czułość, wejście cyfrowe, biblioteka*;
Ultradźwiękowy pomiar odległości	Pomiar odległości z wykorzystaniem czujnika ultradźwiękowego HS-SR04. Prezentacja odczytów na wyświetlaczu LCD oraz sygnalizacja wizualna wykorzystująca diodę RGB, gdy obiekt znajduje się w zadanej odległości.	• trafnie objaśnia pojęcia: *czujnik ultradźwiękowy, wyświetlacz ciekłokrystaliczny, RGB* • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje metodę przesyłania wyników z układu do komputera; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • prawidłowo podłącza serwomechanizm i dokonuje pomiarów odległości; • steruje diodą RGB;
Sterowanie elementami wykonawczymi	Budowa, działanie i sposoby sterowania serwomechanizmem. Programowa oraz mechaniczna zmiana parametrów sterujących.	• trafnie objaśnia pojęcia: *serwomechanizm, biblioteka, PWM* • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • steruje kątem osi serwomechanizmu; • podłącza i steruje serwomechanizm programowo oraz wykorzystaniem buttonów i potencjometru;

02

Metodyka realizacji kół zainteresowań IT i taksonomia efektów*

Stanisław Ubermanowicz

Młodzi adolescenci sprawnie posługują się dziś stacjonarnymi i mobilnymi technologiami informacyjno-komunikacyjnymi (TIK), jednakże zakres nabywanych w szkołach umiejętności jest zbyt skromny do ich pełnego wykorzystania w pracy intelektualno-zawodowej. W tych obszarach specjalizacji nie wystarcza wyuczenie posługiwania się komputerem i użytkowania gotowych aplikacji. Potrzebne jest przygotowywanie liczniejszego grona specjalistów potrafiących zaimplementować wszystko to, czego wymagają pracodawcy. Implementowanie jest złożonym procesem twórczym i właśnie takie intelektualne zdolności muszą być u uczniów wcześnie wykrywane i kształtowane. Powoduje to konieczność wdrożenia **systemowej Strategii** wspierającej formowanie kompetencji z dziedzin *infotechnicznych* (IT), ukierunkowującej większą liczbę nastolatków jako przyszłych kandydatów na studia techniczne.

Połączenie umiejętności projektowania implementacji, tworzenia oprogramowania i konstruowania układów elektronicznych nazywamy tu kompetencjami infotechnicznymi.

Wśród przyczyn zbyt małego zainteresowania wyborem kierunków informatycznych i mechatronicznych wymienia się niewystarczające przygotowanie uczniów i uczennic do podjęcia trudu studiów politechnicznych, a wcześniej do efektywnego (zwieńczonego zdaniem egzaminu państwowego) uczenia się w szkołach ponadgimnazjalnych sprofilowanych technicznie. System powszechnej oświaty – zwłaszcza w gimnazjach – nastawiony jest na kształcenie ogólne, z nadmiarem wpajania treści encyklopedycznych. Dominuje tam wprawianie do definiowania, opisywania, odtwarzania, a zaniedbuje się wykształcenie umiejętności tworzenia, projektowania i konstruowania. Problemem jest także niedobór dobrze przygotowanej kadry nauczycielskiej o poszerzonych specjalnościach infotechnicznych, posiadającej odpowiednie kompetencje psychopedagogiczne do wczesnego formowania

Problemy wymagające interwencji to niska zdawalność egzaminów na tytuł technika z dziedzin IT, zbyt mały nabór i liczne niepowodzenia na studiach politechnicznych.

* Materiał udostępniony do dyskusji na VII Ogólnopolskiej Konferencji Naukowej „Media a edukacja" w publikacji pt. *„Strategia Wolnych i Otwartych Implementacji" w formowaniu kompetencji infotechnicznych* [w:] W. Skrzydlewski, S. Dylak (red.): *Media – Edukacja – Kultura: W stronę edukacji medialnej*, Wyd. PTTiME, Poznań 2012

u uczniów i uczennic procesów umysłowych wyższego rzędu, koniecznych w trudnej sztuce implementowania.

Środkiem zaradczym jest wypracowany w ramach innowacyjnego Projektu SWOI, przetestowany i upowszechniany **Program** nauczania i uczenia się infotechniki, który wspiera nauczycieli w doborze treści i w doskonaleniu oddziaływań metodycznych, a odbiorcom służy w nabywaniu kompetencji infotechnicznych. Jest on polecany dla uczennic i uczniów w wieku od 13 lat jako cykl kształcenia mieszanego (*blended learning*), łączącego edukację pozalekcyjną i pozaszkolną. W fazie stacjonarnej zajęcia w formie kół zainteresowań prowadzi kadra, którą celowo nazywamy trenerami ze względu na specyficzną metodykę *coachingu*. Uczniowie poznają tam nowe systemy, aplikacje, narzędzia i języki programowania, montują interfejsy i układy mechatroniczne, nabywają umiejętności tworzenia implementacji w środowisku Wolnego i Otwartego Oprogramowania (WiOO). Równoległe i dalsze samokształcenie ustawiczne oferowane jest na dedykowanej platformie zdalnej ze wsparciem doradców i społeczności sieciowej.

Narzędziem interwencji jest kompleksowa Strategia edukacyjna, a w niej Program nauczania-uczenia się infotechniki na kołach zainteresowań i poprzez Internet.

Racjonalizacja formowania kompetencji infotechnicznych

Krytycznym momentem na wybór ścieżki specjalizacji zawodowej jest okres gimnazjalny. To jednak wyjątkowo burzliwy i trudny do formowania osobowości czas w życiu nastolatków. Psychologowie szczegółowo scharakteryzowali potencjalne możliwości i ograniczenia tej grupy wiekowej. Proponowany Program uwzględnia psychopedagogiczne prawa określające uwarunkowania i mechanizmy uczenia się populacji młodzieży w wieku dorastania, zwłaszcza w okresie tak zwanej *wczesnej adolescencji*. Rekomendacje wynikające z tych praw muszą być bezwzględnie stosowane przez trenerów formujących kompetencje infotechniczne. Zalecenia przedstawione są w dwóch integralnych publikacjach upowszechniających Strategię i Program, dlatego nauczyciele przygotowujący się do ról trenerów powinni wnikliwie przestudiować cały pakiet materiałów dydaktycznych, metodycznych i infotechnicznych.

Efektywne formowanie kompetencji infotechnicznych warto uaktywniać nie później niż od I klasy gimnazjum.

Do opanowania sztuki programowania potrzebne jest osiągnięcie pewnego poziomu dojrzałości **twórczych procesów umysłowych**. Konieczna jest zdolność do koncypowania dwubieżnego, z naprzemiennymi fazami: analizowania | syntezowania, dedukcji | indukcji, abstrahowania | ukonkretniania, ekstrakcji | agregacji, dywergencji | konwergencji. Oprócz odpowiedniego poziomu procesów inferencyjnych niezbędne jest właściwe formowanie związanych z meritum infotechniki takich struktur umysłu, jak: skrypty, ślady, wzorce, idee, wyobrażenia i pojęcia.

Te struktury z obszarów programowania są u większości beneficjentów w początkowym stadium. Ponadto trudności z percepcją sprawia wielość zupełnie nowych słów

Program nie jest kierowany do ogółu uczniów ani tylko do uzdolnionych, lecz do wszystkich chętnych, w tym także wymagających głębszego wsparcia.

i desygnatów branżowych. A właśnie zdolności do operacji pojęciowych i do ich werbalizacji są tu kluczowe. Tak trudny materiał nauczania nie może być włączany do szkolnictwa ogólnego jako obowiązujący wszystkich. Powinna to być forma rozszerzająca dla tych, którzy mają odpowiednie predyspozycje i zainteresowania, a także dla uczniów ambitnych, którzy nie mieli praktycznej okazji poznać i odkryć swoich twórczych możliwości, lecz chcących zweryfikować ów potencjał.

Istotna przesłanka wpływająca na wybór rozwiązań metodycznych wynika z faktu, że dynamika rozwoju umysłowego uczniów jest silnie zindywidualizowana. Znaleźć można przykłady bardzo wczesnego osiągania znaczących efektów w dziedzinie programowania. Jednak zadaniem Programu nie jest wyłanianie owych talentów, uczniów, którzy już wcześniej obrali ścieżkę autoedukacji infotechnicznej, lecz zdecydowane wyjście poza ten zbyt wąski krąg. Co więcej – z założenia Program ma wspierać beneficjentki i beneficjentów także „tych słabszych", potrzebujących wsparcia organizacyjnego i środowiskowego dla **wyrównywania szans**. Adresowany jest do uczennic i uczniów wykazujących dobrowolną chęć zmierzenia się z trudną materią i sprawdzenia samego siebie pod kątem trafnego ukierunkowania zawodowego.

W edukacji harmonizowane muszą być aspekty poznawczo-kształcące, emocjonalno-motywacyjne i psychomotoryczne.

Warunkiem powodzenia przyjętej taktyki edukacyjnej jest pobudzenie do działań płynących z wewnętrznej potrzeby uczenia się i gotowości doskonalenia w wybranej dziedzinie w optymalnym okresie rozwoju. Do tego konieczne jest kształtowanie świadomości pożytków wynikających z nabywania kompetencji, na które jest duże zapotrzebowanie społeczne. W edukacji osobistej młodych adolescentów ważny jest rozwój zrównoważony. Na równi z poznawaniem niezbędne są przyjemne doznania. To, co podczas uczenia się jest konieczne lecz niełatwe, musi być waloryzowane tym, co miłe lub interesujące. Dlatego kluczem do efektywności Programu nauczania-uczenia się IT jest harmonizowanie proporcji między porcjami treści merytorycznych a elementami doznań emocjonalnych, które aktywują i utrwalają pożyteczne cechy wolicjonalne, w tym chęć działania.

Psychikę i osobowość człowieka regulują mechanizmy mające **cechy względnej trwałości**. Są to zwłaszcza systemy wartości, dobre wzorce i pożądane postawy, ale też niestety mniej korzystne stereotypy, nawyki i rutyny. Pozytywne zmiany w elementach niekorzystnych i wzmacnianie tych pożądanych jest znacznie lepszym motorem rozwoju aniżeli przyswajanie bardziej ulotnych treści nauczania. Wyrobienie woli samokształcenia i samodoskonalenia ustawicznego jest bardziej wartościowe aniżeli chwilowe wyuczenie partii materiału i zdanie egzaminu lub testu. Z tego powodu przedstawiony tu jako wzorcowy Program nauczania-uczenia się w ramach kół zainteresowań jest bardziej nastawiony na formowanie efektów długofalowych, odroczonych, a nie doraźnych.

Program służy zainicjowaniu procesu kształtowania takich umiejętności i cech, które dadzą znaczące efekty w przyszłości.

W formowaniu kompetencji infotechnicznych uruchamiane są wielorakie czynności psychiczne: procesy poznawcze, aktywizacyjne, emocjonalno-motywacyjne

i lingwistyczne. W procesach umysłowych i w działaniach praktycznych podczas tworzenia implementacji wykorzystuje się wszystkie typy reprezentacji świata, tj.: zarówno praktyczne odwzorowania czynności motorycznych, jak też realistyczne odzwierciedlenia ikoniczno-obrazowe, aż po wyrażenia symboliczne, prowadzące do przekształcania rzeczywistości w świat wirtualny.

W konsekwencji osiągane są **efekty wielowymiarowe**, m.in.:

- w sferze poznawczej uczeń wyobraża sobie abstrakcyjne obiekty i potencjalne zdarzenia, myśli logicznie i konstruktywnie, interpretuje algorytmy, analizuje kody i dostrzega prawidłowości, koncypuje, strukturyzuje i syntezuje wirtualne aplikacje;
- w sferze działaniowej uczeń prawidłowo wykonuje indywidualne lub zespołowe zadania implementacyjne, tworzy obiekty graficzne, konstruuje interfejsy, montuje układy elektroniczne, partycypuje w pracach grupowych, wspiera innych;
- w sferze doznaniowej uczeń pasjonuje się zagadnieniami IT, chętnie poświęca czas na uczenie się programowania i konstruowania, ma motywację do tworzenia i doskonalenia implementacji, odczuwa satysfakcję z wykonanych dzieł;
- w sferze językowej uczeń rozumie specjalistyczne pojęcia, zadaje trafne pytania merytoryczne, sensownie odpowiada, podpowiada innym, zna podstawowe struktury języka programowania, pisze kod źródłowy i objaśnia jego działanie.

Część z tych efektów osiąga się bezpośrednio na zajęciach stacjonarnych, lecz ich pełnię zgłębia się dopiero w dłuższej perspektywie systematycznej edukacji. Dlatego najważniejszym celem jest formowanie świadomości potrzeby samodoskonalenia i trwałej woli osiągania efektów kierunkowych.

Uświadamianie pożytków z nabywania kompetencji infotechnicznych stymuluje wolę wysiłku na rzecz dalszego, stałego rozwoju.

W metodyce realizacji zajęć w formie kół zainteresowań infotechnicznych uwzględnia się dorobek wielu dyscyplin, m.in.: dydaktyki, psychologii uczenia się, pedagogiki medialnej, technologii kształcenia, informatyki i inżynierii programowania. Ponadto metodyka wczesnego nauczania-uczenia się języka programowania ma wiele wspólnego z wypracowanymi i sprawdzającymi się metodami glottodydaktycznymi. Najwłaściwsza jest integracja wielu metod i form z wykorzystaniem takich elementów, które w danej fazie zajęć są najbardziej korzystne i skuteczne.

Optymalne połączenie metod uznawanych w dydaktyce za tradycyjne z metodami innowacyjnymi wywołuje zjawisko **synergii**, czyli wzmacniania efektów. Niezbędne są zarówno procesy asymilacji z zapamiętywaniem wiadomości, jak też akomodacji z jakościową rekonstrukcją przyswajanych treści. Nieodzowne i najbardziej cenione są metody problemowe z podejściem badawczym i metody konstruowania wiedzy z większą samodzielnością ucznia.

W realizacji kół zainteresowań potrzebna jest różnorodność metod oddziaływań, przy jednoczesnej redukcji nadmiarowości przekazów.

Znamienną cechą optymalnej metodyki realizacji kół zainteresowań jest nauczanie-uczenie się *zappingowe*. Polega ono na dynamicznym „przemykaniu" przez zagadnienia i formowaniu jedynie zalążków

wiedzy, którą uczniowie uzupełniają, czerpiąc z rezerwuarów własnego intelektu lub z ogromnych zasobów zewnętrznych reprezentacji struktur wiedzy, utrwalonych na różnych nośnikach bądź dostępnych w Internecie. Jest to innowacyjne podejście do strategii edukacyjnej, kiedy to proces kształcenia jest zasadniczo tylko inicjacją struktur ramowych, z pozostawieniem wolnej przestrzeni do samodzielnego wypełniania treścią przez uczącego się. Zamiast redundancji przekazywanego materiału nauczania, intencjonalnie preparowany jest jego niedostatek.

Charakterystyka faz zajęć na kołach zainteresowań IT

Na strukturę zajęć składają się różne metody nauczania-uczenia się, typy oddziaływań i rodzaje treści, komponenty formowanych postaw oraz osiągane poziomy efektów twardych i miękkich. Istotą każdej jednostki dydaktycznej na kołach zainteresowań IT powinno być inicjowanie czterech faz metodycznych: **sensytywności, responsywności, problemowości i konstruktywności** oraz osiąganie czterech poziomów emocjonalno-świadomościowych: aktywnego odbioru, adekwatnego reagowania, niewyręczającego współdziałania oraz zanurzenia w samodzielnym tworzeniu.

Aktywność jest istotą zajęć praktycznych, podczas których uczestnicy, przy wsparciu trenera, poznają sposób efektywnego uczenia się.

Niektóre elementy specyficzne dla danej fazy pojawiają się także w innych fragmentach zajęć, jednak struktury najbardziej charakterystyczne występują w określonej chronologii bądź zachodzą cyklicznie. Chodzi tu o wielorakie sprzężone **aktywności** trenerów i uczniów, tj.: komunikowanie i przyswajanie, motywowanie i przeżywanie, indagowanie i wyjaśnianie, odkrywanie i systematyzowanie, ukierunkowanie i definiowanie, rozważanie i rozwiązywanie problemu, koncypowanie i współtworzenie, naprowadzanie i konstruowanie, rozumowanie, eksponowanie i wartościowanie. W części tych procesów stroną inicjującą jest trener, natomiast w toku zajęć coraz większą rolę kreatywną przejmują uczniowie, aż do działań samodzielnych.

Trenerzy poznają metody harmonijnych, optymalnych oddziaływań na sferę poznawczą i emocjonalną uczniów.

W zajęciach na kołach zainteresowań IT musi być zachowana pełna harmonia pomiędzy doznawaniem pozytywnych emocji a poznawaniem i skutecznym działaniem. W sferze afektywnej niezbędne jest pobudzenie i utrzymanie takiego stanu, ażeby uczniowie chcieli wysłuchać i zaaprobować, pytać i odpowiadać, współdziałać, a dalej podjąć trud pracy indywidualnej. W sferze poznawczej należy zadbać o to, aby byli w stanie asymilować, akomodować i zrozumieć treści, a także dociekać, wnioskować i rozwiązywać zadania. W sferze behawioralnej – aby aktywnie odbierali i postrzegali przekazy, wchodzili w interakcje projektowania zespołowego i docelowo tworzyli działające implementacje. Złożoną strukturę modelowej jednostki dydaktycznej ilustruje tabela.

Tabela

Struktura jednostki dydaktycznej realizowanej w formie koła zainteresowań infotechnicznych

Faza	Nauczanie-uczenie się	Oddziaływania i treści	Formowane komponenty postaw			Osiągane poziomy
			behawioralne	kognitywne	afektywne	
sensytywności – uwrażliwienie	podające ⟶ przyswajanie ekspresyjne ↔ przeżywanie	informacyjne, inicjujące motywacyjne, intrygujące	receptywne	asymilacyjne	aprobujące	aktywny odbiór pobudzenie, zaciekawienie chęć wysłuchania
responsywności – uaktywnienie	poglądowe ⟵ odkrywanie reaktywne ↔ indagowanie	ilustracyjne, odwzorowania interpretacyjne, objaśnienia	komunikacyjne	akomodacyjne	aktywizacyjne	adekwatne reagowanie przejawy zrozumienia chęć pytania i odpowiadania
problemowości – decydowanie	pojęciowe ⟶ definiowanie problemowe ↔ dociekanie wynalazcze ⟵ koncypowanie	ukierunkowujące, zadania naprowadzające, kwestie	eksternalne (współpraca)	badawcze	partycypacyjne	niewyręczające wspieranie próba rozwiązania zadań chęć współdziałania
konstruktywności – tworzenie	praktyczne ⟵ konstruowanie inferencyjne ↔ rozumowanie eksponujące ⟵ wyrażanie	operacyjne, implementacja wartościujące, introspekcja satysfakcjonujące, dzieło	internalne (samorozwój)	twórcze	ambicjonalne	zanurzenie w kreowaniu wysiłek intelektualny chęć tworzenia i refleksji

Warto zwrócić uwagę na wyrażone za pomocą strzałek oddziaływania w kolumnie Nauczanie-uczenie się. Symbolizują one dominujące w danej metodzie kierunki komunikacji między trenerem a uczniami, wskazując na stronę inicjatywną bądź na dwukierunkowość interakcji.

Faza sensytywności – uwrażliwienie

» **Faza sensytywności** na zajęciach pozalekcyjnych łączy w sobie mechanizmy pobudzające sfery emocjonalno-motywacyjne, inicjujące zaciekawienie tematyką, z uświadomieniem problematyki, celów zajęć i istoty zaplanowanych zadań oraz uruchomieniem wiedzy uprzedniej, związanej z meritum zagadnień. Uczniów można wystarczająco zmotywować tylko do takich działań wymagających znacznego wysiłku intelektualnego, które z ich perspektywy są ważne lub interesujące.

Potrzeba zaciekawienia treścią występuje w modułach uczących programowania. W modułach mechatronicznych silne pobudzenie zachodzi niemal samoistnie.

Wprawdzie samo przystąpienie do uczestnictwa w kołach zainteresowań jest już osadzone motywacyjnie, lecz tym bardziej każda jednostka dydaktyczna musi spełniać pokładane w niej nadzieje na pożytki płynące z dodatkowych zajęć. Dlatego pobudzenie na samym wstępie stanu *sensytywności*, jako szczególnej wrażliwości na wpływ i skuteczność oddziaływań, wzmacnia aktywność poznawczą, zwłaszcza w przyswajaniu nowych, trudnych treści. Efektywność stymulacji zewnętrznej jest bowiem najwyższa właśnie w sensytywnym okresie zaciekawienia problematyką.

Abstrakcyjne zagadnienia programistyczne wymagają bliższego upoglądowienia niż praktyczne konstruowanie układów mechatronicznych.

Od trenerów prowadzących zajęcia wymaga się umiejętności nadzwyczaj zwięzłego w tej fazie, przystępnego i atrakcyjnego przekazania treści wprowadzających. Od uczniów oczekuje się chęci wysłuchania lub obejrzenia przekazu w skupieniu ułatwiającym zrozumienie. Zakres treściowy komunikatu inicjuje i reguluje trener, a uczniowie poprzez ciekawość poznawczą uczestniczą w aktywnym, świadomym odbiorze. Wprawdzie formy podające są najniższym poziomem oddziaływań, lecz wzbogacone o elementy poglądowe stanowią nieodzowne podłoże do dalszych faz nauczania-uczenia się.

W recepcji afektywnej kluczowe są pozytywne doznania, jakie uczeń powinien doświadczać podczas poznawania tematyki i celów zajęć.

Upoglądowienie jest potrzebne tym bardziej, że uczestniczący w zajęciach uczniowie w zdecydowanej większości są początkującymi adeptami sztuki programowania i konstruowania mechatronicznego. Są dopiero w rozwojowym stadium przedgotowości do myślenia abstrakcyjnego na tak głębokim poziomie, jaki jest niezbędny do tworzenia implementacji infotechnicznych. Dlatego asocjacyjnej formie przekazu, służącej przyswajaniu nowych elementów w strukturach wiedzy, towarzyszyć powinny formy łączące wizualizację z elementami waloryzacji emocjonalnej. Stanem pożądanym jest tu osiągnięcie pierwszego poziomu **recepcji afektywnej** (*receiving*, Anderson, Krathwohl, 2001).

Nauczanie-uczenie się w fazie sensytywnej opierać się powinno przede wszystkim na metodzie eksponującej pewne intrygujące motywy, wywołujące u uczniów doznania emocjonalne, oraz na znacznie okrojonej metodzie podającej w formie opisu wyłącznie takie bazowe treści, które są wymagane we wprowadzającym stadium danej jednostki dydaktycznej. Bardziej chodzi więc w tym momencie o wyczulenie postrzegania poprzez odbieranie wrażeń, o formowanie pozytywnego nastawienia w sferze afektywnych i receptywnych komponentów uczenia się aniżeli

o samo nauczenie. Taką funkcję znakomicie spełnia *zajawka inspirująca*, będąca rodzajem zwiastuna tego, co jest zaplanowane na dane zajęcia.

Zajawka w ekspresyjnej, silnie skondensowanej formie zawiera kluczową porcję informacji, wzmocnioną mechanizmami intensywnego oddziaływania na sferę emocjonalną. Jej postać wizualną łatwo sobie wyobrazić poprzez analogię do reklam lub zwiastunów programów telewizyjnych. Jakkolwiek tego rodzaju przekazy obrazowe (wideoklipy) są najbardziej wskazane, to z powodzeniem rolę kontekstowej zajawki mogą pełnić także przekazy słowne (np. anegdoty, scenki sytuacyjne) lub formy działaniowe (np. zabawy, gry logiczne).

Zajawka to sposób zwięzłego, atrakcyjnego przedstawienia uczniom problematyki danej jednostki zajęć.

Przykładem wykorzystania *zajawki słownej* jest anegdota o pracy mnichów przestawiających złote kręgi w Wieżach Hanoi. Służy ona do wyobrażenia sobie olbrzymiej liczby kombinacji i czasochłonności przy rozwiązaniu tegoż zadania, a jednocześnie może być znakomitym wstępem do problematyki algorytmów i strategii wygranej, z uzmysłowieniem sobie szybkości działania komputerów. Przykładem *zajawki działaniowej* może być próba rozwiązania przez uczniów układanki w klockowej wersji Wież Hanoi. Odpowiednio przygotowany trener może połączyć obie te formy, opowiadając ową anegdotę i szybko układając klocki, przez co wywołuje wrażenie, jakoby w ogóle nie musiał zastanawiać się nad ruchami. Znakomicie wprowadza to do zrozumienia istoty skutecznego algorytmu opartego na optymalnej strategii wygranej.

Zajawka dodatkowo może pełnić także inną bardzo ważną rolę, polegającą na **ukazaniu wzorca**. Uczeń łatwiej przyswoi sobie istotę tworzenia i działania danej implementacji, jeśli zobaczy jej przykładowe wykonanie. Należy przy tym mieć na uwadze, aby wzorzec implementacji infotechnicznej ilustrował jedynie sposób docelowego funkcjonowania wytworu (jak to ma działać), a nie był matrycą do naśladowania i kopiowania wyglądu. Najcenniejszymi *zajawkami wizualnymi* są filmy poglądowe, przygotowane celowo na potrzeby modułów zajęć mechatronicznych, umieszczone w zasobach Internetu i stanowiące uzupełnienie materiałów drukowanych. W całości mogą one służyć jako filmy instruktażowe do samodzielnego wykonania układu elektronicznego przez uczniów, a ich małe fragmenty jako wzorce inspirujące na samym początku zajęć. W tej drugiej roli pamiętać należy, aby wykorzystywać jedynie krótki urywek wideoklipu, nie dłuższy niż minutowy.

Wzorzec implementacji ma przybliżyć uczniom istotę zadania, lecz nie ma być gotowcem do kopiowania.

Z obserwacji zajęć wynika, że nauczyciele mają niestety tendencje do zbyt obszernego czasowo i treściowo „wprowadzania" uczniów w problematykę danej jednostki zajęć w formie nie tylko komunikowania o zamiarach, lecz także poprzez nadmierne w tej fazie objaśnienia. Tymczasem właściwe wyjaśnianie na kołach zainteresowań powinno towarzyszyć bezpośrednio praktycznym działaniom uczniów – przede wszystkim w momentach wymagających zaakcentowania najistotniejszych wątków i kluczowych pojęć, bądź w przypadkach trudności z wykonywaniem zadań. Odbywać się to jednak powinno dopiero w drugiej, responsywnej fazie aktywności.

Zamiast nadmiaru objaśnień wprowadzających, trzeba wspierać uczniów równocześnie z ich praktycznymi działaniami.

Faza responsywności – uaktywnienie

» **Faza responsywności** charakteryzuje się tym, że następuje aktywizacja uczniów ze zmianą ich wcześniejszej roli z odbiorców na interlokutorów. Stopniowo redukowane jest sterowanie doborem treści przez trenera, a to, co przekazuje on uczniom, wynikać powinno przede wszystkim z ich uzewnętrznianych potrzeb. Istotą responsywności jest uwrażliwienie na konkretne zapotrzebowanie wyrażane poprzez pytania oraz szybkie udzielanie odpowiedzi w pełni adekwatnych do oczekiwań.

Objaśnienia powinny być następstwem przede wszystkim potrzeb zgłaszanych przez uczniów.

W fazie tej niezwykle ważną rolą trenera prowadzącego zajęcia jest umiejętność natychmiastowego i trafnego wyjaśniania każdej kwestii, z jaką zwracają się uczniowie. Jednakże nie mogą to być wyjaśnienia czysto werbalne, z operowaniem wyłącznie pojęciami. W miarę możliwości tłumaczenie warto łączyć z wykonywaniem czynności. Dlatego na zajęciach o charakterze laboratoryjnym należy jak najszybciej uruchamiać **działania praktyczne**, podczas których uczniowie natrafiający na konkretne trudności pytają i uzyskują niezbędne wsparcie.

W metodyce pracy z małymi grupami na kołach zainteresowań znamienne jest to, że wsparcia mogą udzielać sobie wzajemnie sami uczniowie. Ułatwia to organizację zajęć, gdyż z jednej strony trener musi szybko reagować na wielorakie potrzeby jednostek, a z drugiej – powinien w miarę sukcesywnie realizować szczegółowe cele zajęć. Umiejętność optymalnego, dynamicznego przechodzenia pomiędzy objaśnianiem adresowanym do całej grupy a wspieraniem indywidualnym, z wykorzystaniem równoległego wsparcia ze strony uczniów wiodących, jest jedną z kluczowych kompetencji trenera, którą w tradycyjnym systemie klasowo-lekcyjnym trudno jest formować.

Responsywność jest bardziej skuteczna, gdy stronami interakcji i wspierania są wzajemnie uczniowie.

Właśnie pozalekcyjne koła zainteresowań są znakomitym poligonem doskonalenia strategii responsywnej zarówno przez nauczycieli, jak i uczniów, i to także przez tych, którzy zadają pytania. Zwykle bowiem uczniowie mają trudności ze zrozumiałym przez nauczycieli dookreślaniem swoich potrzeb, zatem już samo ćwiczenie umiejętności werbalizacji konkretnego problemu jest motorem procesu kształcenia. Znakomicie pomaga tu **interakcja międzyrówieśnicza**, gdyż – posługując się w podobnej fazie rozwoju osobniczego tymi samymi procesami umysłowymi i prostszymi formami języka – jedni przełamują opory, aby pytać, a inni uczą się odpowiadać. Do porozumienia się między uczniami przebywającymi z sobą na co dzień zazwyczaj wystarczają formy sygnalne, strzępy informacji czy półsłowa, dlatego komunikacja pozioma wewnątrz grupy jest efektywnym środkiem, o ile oczywiście trener odpowiednio ją zaaranżuje, zorganizuje i nadzoruje merytorycznie.

Responsywność, jako dynamiczne komunikowanie ukierunkowane przez oczekiwania, dotyczy bardziej cech wolicjonalnych niż poznawczych. Uczeń aktywnie uczestniczący w kołach zainteresowań nie tylko odbiera przekaz, ale silnie reaguje na treści, oceniając je z własnej perspektywy, pierwotnie głównie w sferze emocjonalnej. Odnosi się najpierw do tego, czy treści są zgodne z jego nastawieniem, a dopiero później odnosi

je do swej wiedzy uprzedniej, próbując wartościować materiał nauczania. Zatem warunkiem efektywności zajęć pozalekcyjnych jest trafny dobór tematyki adekwatnej do oczekiwań uczniów. Ze względu na silne zróżnicowanie oczekiwań i skrajnie różne poziomy umiejętności infotechnicznych nie jest możliwe przygotowanie jednorodnych, uniwersalnych, szczegółowych scenopisów zajęć. Potrzebny jest poszerzony zbiór konspektów-scenariuszy modułowych, z których trener układa optymalnie dobrany do swych uczniów cykl programowy.

Opracowane i zalecane jako wzorcowe materiały dydaktyczne: konspekty-scenariusze, zadania, kody i opisy implementacji, instrukcje i schematy mechatroniczne są kanwą tego, co warto realizować na kołach zainteresowań. W rzeczywistości znaczne fragmenty toku zajęć zależą od interakcji między trenerem i uczniami, gdyż strategia responsywna wymusza bardziej swobodny, adaptacyjny styl realizacji każdej jednostki dydaktycznej. Cele szczegółowe muszą być dynamicznie dopasowywane do możliwości grupy, a nawet do pojedynczych uczniów. Nieodzowny w tej fazie jest jednakże cel ogólny, polegający na osiąganiu przez uczniów drugiego poziomu **reaktywności afektywnej** (*responding*, Anderson, Krathwohl, 2001). Oznacza on przede wszystkim uzewnętrznienie reakcji na pobudzenie, wyrażanie osobistych spostrzeżeń, nawiązywanie równorzędnych relacji, chęć zadawania pytań i prowadzenia konwersacji.

W reaktyności afektywnej kluczowa jest chęć uzewnętrznienia stanów umysłu przez uczniów, co umożliwia dopasowanie zajęć do sytuacji.

W warstwie motywacyjnej kluczem jest tu formowanie postaw reaktywnych, natomiast w sferze poznawczej następuje jakościowe doskonalenie dotychczasowych struktur umysłu głównie poprzez akomodację. W zderzeniu napływających do ucznia komunikatów, które ze względu na specyfikę programowania w większości nie pasują do utrwalonych już struktur, ujawnia się potrzeba przystosowania uprzednich wzorców, redefiniowania pojęć i zogniskowania na abstrakcjach. Ażeby ułatwić uczniom te złożone procesy, niezwykle przydatne są w tej fazie techniki pracy z rzeczywistymi obiektami zgodnie z **zasadą poglądowości**. Przy czym nie wystarczy tu pokaz realizowany przez trenera, lecz chodzi o praktyczne działania uczniów na naturalnych przedmiotach lub modelach ściśle nawiązujących do tematyki zajęć.

Problemy wymagające rozwiązania łatwiej jest pokonać, jeśli zadaniom towarzyszy działanie na obiektach rzeczywistych.

Przykładami mogą być proste środki stosowane w bloku zajęć dotyczących gier logicznych: do gry „Papier-kamień-nożyce" używa się symboliki dłoni, do „Przesuwanki-układanki alfabetycznej" wystarczą małe kartki z literami, do układanki „Wieże Hanoi" mogą być użyte tacki plastikowe lub monety, do gry strategicznej „NIM" wystarczą patyczki lub kamyki. Uczniowie za pomocą tych rekwizytów indywidualnie rozwiązują łamigłówki lub grają parami, starając się budować w umysłach jakąś własną strategię wygranej. Poznają w ten sposób nie tylko istotę zadania, jakie będą później wykonywali, ale także doświadczają bezpośrednio rzeczywistego funkcjonowania czegoś, co jest podejściem i rozwiązaniem algorytmicznym.

Dopiero po tej fazie przystępują do wykonywania zadania polegającego na zaimplementowaniu poznanej gry na komputerze za pomocą narzędzi programistycznych. Proces implementowania dla początkujących jest niezwykle złożony, dlatego do pokonania trudności w dalszym toku zajęć potrzebne jest podejście problemowe, oparte na zespołowym współdziałaniu grupy i trenera.

Faza problemowości – decydowanie

» **Faza problemowości** wyróżnia się istotną zmianą stylu prowadzenia zajęć. Trener ogranicza styl podający i objaśniający, przechodząc na wspieranie uczniów w ich własnych próbach rozwiązywania problemów i realizowania zadań. Ze względu na to, że w większości uczestnicy kół infotechnicznych są zupełnie początkujący w posługiwaniu się wolnymi i otwartymi narzędziami programowania, nie może być całkowitego wyłączenia transferu informacji od trenera i w pełni samodzielnego dochodzenia do wiedzy przez uczniów. Niezbędne w tym momencie jest wspólne kreowanie i uzgadnianie podstaw do wykonania zadania implementacyjnego.

W pokonywaniu problemów ważne jest wspieranie bez wyręczania, poprzez naprowadzanie uczniów na własne dochodzenie do rozwiązań.

Istotą konwersacji w tej fazie staje się zastosowanie metody naprowadzania uczniów na rozwiązania poprzez stosowanie form poszukujących, pytających i ukierunkowujących na właściwą odpowiedź. Nie jest to jednak tradycyjne odpytywanie, lecz raczej „dopytywanie wspierająco-naprowadzające" lub „indagowanie sugerujące korzystne wybory", które z braku trafnego polskiego słowa nazywać tu będziemy *inquiringiem* (Kubicek, 2005). Kluczem skuteczności w tej metodzie jest specyficzna trenerska umiejętność **wspierania uczniów bez wyręczania**, z ułatwianiem znalezienia rozwiązania problemu poprzez stawianie pytań pomocniczych. Nie muszą to być jednak tylko pytania – rolę tę mogą pełnić inne formy, np.: wyrażenie przypuszczenia, ukierunkowanie na trop, wskazanie alternatyw, uwarunkowań, pożądanych cech itp. Taki sposób wsparcia jest cenniejszy edukacyjnie niż dostarczenie gotowego rozwiązania. Warunkiem jest wyrobienie nawyku pełnienia trudnej roli *inquirera*.

Integralnym elementem tej fazy zajęć jest metoda problemowa lub metoda projektów bądź szczególna odmiana, jaką w obszarach infotechniki jest metoda wynalazcza. Na kołach zainteresowań uczniowie otrzymują do realizacji zróżnicowane zadania – jedne są możliwe do wykonania samodzielnego na podstawie instrukcji, a inne wymagają uprzedniego wprowadzenia i pracy zespołowej, a dopiero później działań zindywidualizowanych.

W początkowym etapie implementowania programistycznego pożądana jest współpraca zespołowa, natomiast konstruowanie mechatroniczne winno być indywidualne.

Każde z zadań stwarza dla ucznia **sytuację problemową**, która powoduje odczuwanie trudności. W procesach tworzenia atrakcyjnych implementacji zadania są zbyt złożone, dlatego na wstępie włącza się do działań twórczych całą grupę, wspólnie poszukując idei rozwiązania, koncepcji wykonania, form i elementów prototypu. Poprzez „burzę mózgów" ustala się pewne założenia kluczowe, które po akceptacji stają się wspólnym podłożem do działań indywidualnych, a dzięki uzgodnieniom dają większe szanse na organizacyjne ogarnięcie całości przez trenera i osiągnięcie sukcesów indywidualnych przez uczniów.

Przykładowo – programowanie wizualno-obiektowo-zdarzeniowe rozpoczyna się od wyobrażenia sobie elementów graficznych, jakie mają pojawić się na ekranie. Na tym etapie konieczne jest myślenie

konwergencyjne, prowadzące do pewnego ujednolicenia typów zastosowanych obiektów (tzw. *widżetów*), ich ról, nazw i ogólnych właściwości. Dopuszczalne są odmienne koncepcje graficzne, lecz z zachowaniem zgodnego nazewnictwa obiektów, ażeby później, przy tworzeniu procedur, nie było zamieszania utrudniającego prowadzenie zajęć.

W kolejnym etapie przechodzi się do ustalenia kluczowych założeń co do kodu źródłowego, to jest do uzgodnienia procedur i funkcji, jakie będą potrzebne do obsługi zdarzeń oraz do definiowania zmiennych. Także i w tym przypadku definiowania zmiennych i deklarowania procedur warto ustalić jednolite nazewnictwo ułatwiające dalsze działania. Jakkolwiek te wypracowane struktury są efektem współpracy grupy, to jednak każdy z uczniów wprowadza je do swojej implementacji samodzielnie, nadając im zindywidualizowany wygląd i jednocześnie ćwicząc praktyczną obsługę narzędzi służących programowaniu.

Wspólne wypracowanie założeń do struktur implementacji programistycznej ułatwia organizację zajęć.

Ważnym założeniem w tej fazie jest formowanie **umiejętności pracy zespołowej**. Często bowiem projekty informatyczne wymagają współpracy grupy osób. W takiej grupie istotną sprawą jest koordynacja działań, którą zwykle pełni trener. Warto jednak próbować powierzyć rolę koordynatora któremuś z wiodących w grupie uczniów. Rówieśnicy, operując językiem środowiskowym, łatwiej porozumiewają się między sobą, łatwiej uczą się od siebie niż od nauczyciela. Oczywiście trener nadzoruje interakcję, aby była ona w miarę równoprawna, bez zbytniej dominacji jednostek najzdolniejszych. Wsparcie ze strony liderów powinno sprowadzać się przede wszystkim do niewyręczającego naprowadzania. Osiągnięcie takiego poziomu umiejętności przez niektórych uczniów uformuje „czołówkę", która później na platformie edukacji zdalnej w internetowej społeczności e-Swoi może pełnić kluczową rolę doradców.

W fazie pracy zespołowej korzystne jest przeniesienie roli koordynatora na ucznia, z zachowaniem zasady równoprawności i wymiany ról.

Na tej fazie nie powinny kończyć się zajęcia, gdyż rozwiązanie problemu jak wykonać zadanie jest tylko fragmentem planowanej implementacji. W zasadzie powstaje projekt i zarys niezbędnych komponentów będących półproduktami. Dalsze tworzenie implementacji odbywać się powinno w zupełnie inny metodycznie sposób, opisany w dalszej części. Wskażmy zatem na charakterystyczną cechę odróżniającą fazę rozwiązywania problemów przy projektowaniu od finalnej fazy konstruowania. Chodzi o to, co jest w danym momencie dominującym, pierwotnym motorem procesów i źródłem formowania złożonych struktur umysłowych: **słowo czy czynność**?

Jak objaśniono, istotą trzeciej fazy zajęć jest wspieranie naprowadzające na osiągnięcie rozwiązania i organizujące pracę w zespole. Sterowanie czynnościami odbywa się w warstwie werbalnej poprzez wykorzystanie słów. W dziedzinie infotechniki występuje jednak tak wiele nowych dla uczniów pojęć, że ich przyswajanie musi odbywać się w każdym momencie zajęć – najpierw poprzez odbiór i wyjaśnianie, a później w coraz bardziej zaawansowany sposób. W metodzie *inquiringu* zachodzi pojęciowe ukierunkowywanie czynności, zatem uczeń najpierw musi zrozumieć sens słowa, aby mógł wykonać działanie.

Organizowanie prac wymaga przyswojenia specjalistycznych słów, stąd wykonana czynność jest wskaźnikiem zrozumienia.

W toku zajęć należy przejść od definiowania pojęć do operacji odzwierciedlających sens danego pojęcia.

Wspieranie polega na używaniu innych słów, przybliżających znaczenie pojęcia specjalistycznego, a to oznacza strategię *definiowania pojęciowego*. Jest ona przydatna do kształtowania ważnych w procesie formowania kompetencji IT procesów inferencyjnych, takich jak: kojarzenie, desygnowanie znaczeń, myślenie abstrakcyjne, konwergencyjne, analityczne i logiczne-indukcyjne. Jednakże większą efektywność przypisuje się metodzie komplementarnej, która polega na czynnościowym kształtowaniu pojęć poprzez mechanizmy *definiowania operacyjnego* (Ubermanowicz, Bielawska, 2003). Ta druga metoda realizowana jest częściowo w fazie projektowania, a w pełni na etapie konstruowania implementacji.

Faza konstruktywności – tworzenie

» **Faza konstruktywności** charakteryzuje się zdecydowaną redukcją działań wspierających ze strony trenera bądź grupy. Istotą jest tu bowiem praca samodzielna ucznia skupionego na rozumowaniu i tworzeniu. Zgodnie z zasadą *wygaszania wsparcia*, proces uczenia się jest bardziej efektywny, gdy wcześniejsze ułatwienia są stopniowo zmniejszane. O ile w fazie problemowej trudność jest czynnikiem inspirującym do jej przezwyciężenia, o tyle w fazie konstruowania motywacją jest realna możliwość samodzielnego wykonania, modyfikacji lub dokończenia implementacji.

Podczas tworzenia implementacji infotechnicznych zachodzą **procesy wyższego rzędu**, zwłaszcza: konstruowanie złożonych struktur abstrakcyjnych, myślenie syntetyczne, twórcze, hipotetyczno-dedukcyjne, predykcyjne, dywergencyjne i algorytmiczne. Ponadto w przypadku prób zaimplementowania elementów sztucznego intelektu – niezbędnych do tego, aby komputer mógł na przykład wygrać z człowiekiem – konieczne jest podejście heurystyczne.

Formowanie trafnego rozumienia pojęć jest skuteczniejsze, jeśli uczeń wykonuje czynności, które operacjonalizują dane pojęcie.

Na potrzeby przygotowywania uczniów do coraz bardziej samodzielnej realizacji własnych wytworów infotechnicznych potrzebne jest uspójnianie czynności fizycznych z czynnościami umysłowymi. Najlepszą tego metodą jest czynnościowe kształtowanie pojęć, wspierane zasadą przechodzenia od operacji konkretnych, poprzez wyobrażeniowe, aż do abstrakcyjnych. Następuje wówczas stopniowe uwewnętrznienie doświadczeń rzeczywistych z włączeniem ich sensu do struktur pojęciowych. Zrozumienie wielu zagadnień infotechnicznych wymaga właśnie wsparcia polegającego na wykonaniu specjalistycznej czynności (np. strukturyzacji, konsolidacji, kompilacji).

Podczas procesu implementowania zachodzi też proces odwrotny, kiedy to abstrakcyjny zamysł przybiera postać realnego wytworu. Dzięki dwukierunkowości procesów, czynności umysłowe i praktyczne silniej integrują się. Ze względu na rozmaitość elementów i złożoność problemów, implementowanie może samodzielnie pełnić dydaktyczną rolę jednocześnie **środka i metody** uczenia się.

Właśnie specyfika implementacji – jako zarówno wytworu i procesu konstruowania – powoduje, że jest to najbardziej ceniony metodycznie konglomerat zwany *środkiem-metodą*, integrujący medium będące pomocą dydaktyczną z wewnętrzną strategią formowania wiedzy i umiejętności. Implementacja łączy w sobie hardware, software i zarazem *teachware*, tj. immanentną i koherentną instrukcję metodyczną, dzięki której w fazie konstruktywności oddziaływanie trenera staje się niemal zbędne. Uczeń powinien wówczas całkowicie „zatopić się" w myślach i czynnościach, wyłączając się na pewien czas z interakcji zewnętrznych i osiągając wewnętrzny stan immersji.

Zadania implementacyjne wyznaczają metodę uczenia się, natomiast narzędzia i wytwory implementacyjne są środkami dydaktycznymi.

Taki stan skupienia i wysokiej koncentracji potrzebny jest w twórczości programistycznej oraz w konstruowaniu układów. I nie chodzi tylko o skupienie się na samym kreowaniu dzieła, lecz także na wdrażaniu do działań optymalnych. Brak uwagi prowadzi bowiem do błędów, które trudno jest w złożonym kodzie źródłowym wykryć i naprawić, a co gorsza – w układach mechatronicznych może powodować nieodwracalne uszkodzenia podzespołów.

Z osiąganiem poziomu pełnej **immersji** nie jest łatwo, a składa się na to kilka przyczyn. Uczniowie nie mają nawyku, aby w czasie zajęć całkowicie zanurzać się w rozumowaniu. Na tradycyjnych lekcjach klasowych stan zamyślenia bywa czasem wręcz karany jako... brak uwagi. Ponadto uczeń, który wie jak wykonać zadanie, odczuwa nieodpartą chęć podzielenia się tą wiedzą z innymi. Także trenerom trudno jest powstrzymywać się od wspierania, gdy widzą, że jakaś czynność zajmuje niektórym uczniom zbyt dużo czasu. Tymczasem tworzenie jest zawiłą sztuką, która wymaga właśnie odpowiedniej samodzielności, swobody i przestrzeni czasowej.

Zanurzenie się w myślach i skupienie na samodzielnych działaniach jest kwintesencją podejścia konstruktywistycznego.

Z testowania realizacji Programu wynika, iż nauczyciele hospitujący zazwyczaj nie dostrzegają stanu immersji, choć występuje ona w wielu formach. Wśród przykładowych działań wymagających zanurzenia można wymienić choćby autorskie tworzenie grafiki lub przetwarzanie zdjęć, rozwiązywanie łamigłówek, odkrywanie strategii wygranej, wnikanie w istotę algorytmu, projektowanie obiektów wirtualnych i pisanie kodów źródłowych. Najpełniej immersja zachodzi przy konstruowaniu modułów mechatronicznych, kiedy to praca uczniów opiera się na wykorzystaniu instrukcji. Zanurzenie ma wówczas postać pogłębionej interakcji z rysunkami i schematami, a więc z fundamentalnymi dla techniki przekazami symbolicznymi, na podstawie których trzeba samodzielnie dobrać właściwe elementy elektroniczne i zmontować podzespoły w funkcjonalny układ.

Uczniowie najbardziej lubią moduły mechatroniczne, dlatego warto je łączyć w cykle z trudniejszymi modułami programowania.

Poprzez wykonanie prawidłowo działającej implementacji następuje nie tylko sprawdzenie nabytej umiejętności, ale też osiągane są inne cele. W sferze emocjonalnej odczuwana jest przez uczniów olbrzymia satysfakcja z wykonanego dzieła, a w sferze świadomości wzmacniana jest ocena własnych zdolności. Dzięki tym mechanizmom, które zgodnie z *prawem efektu* utrwalają aktywności nagradzane, formowane są

Sukcesy ucznia na zajęciach wzmacniają świadomość pożytków oraz pozytywne postawy wobec dziedzin informatyki.

kierunkowe **cechy wolicjonalne**, w tym tak bardzo potrzebna chęć systematycznego samokształcenia w dziedzinie infotechniki i doskonalenia tej pożądanej społecznie kwalifikacji zawodowej oraz chęć partycypacji w późniejszych pracach zespołowych w społeczności sieciowej. I właśnie uświadamianie specyfiki podejścia inżynierskiego oraz formowanie sprzyjających postaw wobec trudu tworzenia jest najważniejsze w początkowej fazie uczenia się programowania i konstruowania.

Zajęcia stacjonarne w ramach kół zainteresowań powinny uruchomić dalszy proces uczenia się ustawicznego.

Uczniowie rozpoznający uzdolnienia w danym kierunku powinni sami sterować dalszą swoją aktywnością w nabywaniu kompetencji. Mogą to realizować długofalowo także poza zajęciami stacjonarnymi, na internetowym Serwisie e-Swoi, będącym integralną platformą edukacji pozaszkolnej w Strategii SWOI. Tam procesy *respondingu*, *inquiringu* oraz *immersji* przybierają zupełnie nowe, niezwykle cenne edukacyjnie formy ogólnodostępnego wsparcia i zanurzenia: w branżowym środowisku społecznościowym, w międzyrówieśniczym i międzypokoleniowym transferze wiedzy i umiejętności, w materiałach źródłowych, tutorialnych i repozytoryjnych, w wolnych narzędziach i otwartych zasobach wzorcowych implementacji.

Bibliografia

- Anderson L.W., Krathwohl D.R. (eds.): *A Taxonomy for Learning, Teaching, and Assessing*, Addison Wesley Longman, London 2001
- Bestrzyński W.: *Kształcenie odtwarzające środowisko immersyjne dla przyswajania języka obcego*, „Neodidagmata" 2001, nr 31/ 32, Wyd. Naukowe UAM, Poznań
- Kubicek J.P.: *Inquiry-based learning, the nature of science, and computer technology: New possibilities in science education*, "Canadian Journal of Learning and Technology" 2005, vol. 31(1)
- Szymiec R.: *Nauczanie responsywne – nowe pojęcie w nauczaniu*, „Neodidagmata" 2011, nr 31/ 32, Wyd. Naukowe UAM, Poznań
- Ubermanowicz S., Bielawska H.: *Czynnościowe kształtowanie pojęć*, „Neodidagmata" 2003, nr 25/26, Wyd. Naukowe UAM, Poznań
- Ubermanowicz S.: *„Strategia Wolnych i Otwartych implementacji" w formowaniu kompetencji infotechnicznych*, [w:] W. Skrzydlewski, S. Dylak (red.): *Media - Edukacja - Kultura*, Wyd. PTTiME, Poznań 2012
- Zając A.: *Uczenie się w sieci przez zapping*, „Neodidagmata" 2011, nr 31/ 32, Wyd. Naukowe UAM, Poznań
- Zajenkowski M.: *Emocje i procesy poznawcze jako przykład elementarnych przedmiotów psychicznych*, [w:] J. Szymanik, M. Zajenkowski (red.): *Kognitywistyka*, Wyd. KFpMISH UW, Warszawa 2004

03

Konspekty-scenariusze realizacji kół zainteresowań IT

Krzysztof Wawrzyniak

Myślą przewodnią podczas przygotowywania Konspektów-scenariuszy było takie ich opracowanie, aby bazowały na ustalonym szablonie, składającym się z określonych kategorii treściowych. Założenie co do całkowitego ujednolicenia Konspektów okazało się trudne do spełnienia ze względu na dużą różnorodność tematyki (grafika komputerowa, programowanie, mechatronika) oraz wynikający stąd inny rodzaj zadań i ćwiczeń. Zachodziła potrzeba dostosowywania przedstawionej poniżej struktury ogólnego szablonu do specyfiki konkretnych zajęć, a także do stopnia trudności, zależnego od wieku uczniów i rodzaju szkoły. Mimo to, formy przygotowanych materiałów dydaktycznych są w miarę jednolite.

W niniejszej publikacji umieszczono obok siebie Konspekty-scenariusze zalecane do realizacji w cyklu zajęć tworzących pewną ciągłość merytoryczną. Takie zgrupowane bloki modułów (po trzy dla szkół podstawowych i po cztery dla pozostałych szkół) oznaczono symbolami wskazującymi poziom i numerację danego bloku – odpowiednio: 01, 02, 03, A1, A2, A3, B1, B2, B3 oraz C1, C2, C3. Numeracja 1 i 2 odnosi się głównie do tematyki projektowania i programowania, a nr 3 do mechatroniki. Środowiska pracy i zagadnienia tych zgrupowanych modułów przedstawiono w spisie treści i na przekładkach oddzielających Konspekty-scenariusze przeznaczone do określonych rodzajów szkół: podstawowych, gimnazjalnych, ponadgimnazjalnych niesprofilowanych oraz sprofilowanych infotechnicznie.

Drugim zakładanym priorytetem była czytelność i przejrzystość Konspektów-scenariuszy, a zarazem zwięzła forma, aby nauczyciel mógł bez trudu, po spojrzeniu na krótki materiał (w połączeniu z opisem implementacji), zyskać pełen obraz i zrozumieć nie tylko ideę zajęć, ale również proponowany przebieg i sposób realizacji. W takiej koncepcji – specyfikacja materiału dydaktycznego, przeznaczonego na daną jednostkę zajęć, zajmuje pierwszą stronę Konspektu, a na kolejnej stronicy w tabeli ujęty jest opis czynności w formie scenariusza. Ocenę powodzenia założeń co do funkcjonalności i przydatności Konspektów-scenariuszy pozostawiamy nauczycielom, którzy będą sięgać po tę publikację, aby korzystać z proponowanych modułów we własnej praktyce.

Konstrukcja konspektów i zasady korzystania

Przygotowane i przetestowane w ramach Projektu SWOI Konspekty-scenariusze zawierają kategorie treściowe charakterystyczne dla większości tego typu opracowań. Posiadają jednak

także rozszerzone wpisy, m.in. odniesienie do konkretnej Implementacji, która jest na danych zajęciach zarazem środkiem i metodą. Podkreślmy zatem, że na dany Moduł, będący jednostką dydaktyczną zajęć, przeznaczony jest spójny pakiet materiałów i środków, tj.:

- Konspekt-scenariusz z treścią, celami, metodami, opisem czynności, środków i efektów;
- Opis Implementacji z zadaniem/poleceniem oraz przykładowym rozwiązaniem (np. tekst kodu źródłowego z opisem objaśniającym, bądź instrukcja połączeń modułu-interfejsu);
- Implementacja w postaci elektronicznej (np. pliki z kodem i zasobami źródeł do ćwiczeń);
- Funkcjonalność e-Portfolio i e-Repozytorium na dedykowanym Serwisie e-Swoi.pl.

Poniżej prezentujemy charakterystykę poszczególnych elementów struktury Konspektów, aby nauczyciele bez trudu mogli zrozumieć zamysł ich zastosowania:

- **Temat** – to motyw przewodni zajęć, zagadnienie merytoryczne lub wiodące hasło.
- **Nazwa implementacji** – tytuł danego wytworu infotechnicznego.
- **Opis implementacji** – krótkie omówienie wytworu jaki ma powstać, syntetyczne ujęcie istoty jego funkcjonalności; odpowiedź na pytanie: „Co w ramach zajęć będziemy robić?”, „Co dana implementacja będzie wykonywać i czemu to będzie służyć?”
- **Cele ogólne** – to naczelne przesłanie realizowanych przez nauczyciela zajęć, wynikające z celów Strategii edukacyjnej i odwołujące się do trzech sfer osobowości ucznia:
  - *intelektualnej*, związanej z formowaniem sfery poznawczej (wiedza);
  - *psychomotorycznej*, związanej ze sferą czynnościową (umiejętności);
  - *afektywnej*, związanej ze sferą emocjonalno-motywacyjną (postawy);

  Ich zastosowanie można uzasadnić tym, że nauczyciel poprzez treści wychowuje zarazem oraz choćby demonstruje radzenie sobie z informacją, zapoczątkowując umiejętności. Cele te – ujmowane w kategoriach potencjalnych, odroczonych skutków oddziaływań długofalowych – ukazują sens podejmowania aktywności uczniów, odzwierciedlając zarazem dążenia do zwiększenia zainteresowania wyborem kierunków infotechnicznych.
- **Cele szczegółowe** – zawierają informację o zdobywanej przez uczniów wiedzy, umiejętnościach i kształtowanych postawach w zakresie realizowanego na zajęciach materiału. Zapis tych celów nie jest jednak zbyt szczegółowy – nie wskazują one konkretnych pojęć czy umiejętności związanych z wykorzystaniem narzędzia do konstruowania konkretnego „wytworu”. Cele te ukazują obszary, w których nastąpi zmiana na skutek realizacji zajęć.
- **Materiał nauczania-uczenia się** – wykaz zagadnień informatycznych lub mechatronicznych (zasadniczych treści przedmiotowych, jakie nauczyciel omawia lub do jakich dochodzą sami uczniowie), wyznaczających merytoryczny zakres jednostki zajęć, jak również wskazanie mediów i materiałów niezbędnych do realizacji zajęć (np. typ/wersja oprogramowania, sprzęt, narzędzia itp.).
- **Metody, działania** – wskazanie zalecanych metod i działań, przy pomocy których zostanie zrealizowany powyższy zakres materiału. Zalecane metody i działania w sposób czytelny znajdują odzwierciedlenie w czynnościach uczniów i działaniach nauczyciela, aby dla nauczyciela realizującego Konspekt było jasne, w którym momencie zajęć dana metoda czy działanie ma zastosowanie.
- **Wskaźniki osiągania celów (efekty)** – mierzalny/obserwowalny zapis efektów uzyskanych w ramach obszarów ujętych w celach szczegółowych. Ich sformułowanie umożliwia trenerowi

skontrolowanie i ocenę stopnia realizacji założeń, w perspektywie uczniów. Należy jednak mieć świadomość, że niektóre wskaźniki mogące potwierdzać osiągnięcie celów szczegółowych, można zaobserwować dopiero po upływie dłuższego czasu.

» **Tabela: Czynności uczniów | działania nauczyciela | materiał** – takie zdefiniowanie czynności daje wgląd w relacje między celami i działaniami uczniów a materiałem nauczania; kolejność działań jest sugerowana następstwem w układzie pionowym tabeli, ale pozwala na zestawianie czynności w ramach danego materiału (w układzie poziomym).

Czynności uczniów i nauczyciela są wzajemnie komplementarne – z konspektów wynikają role tych podmiotów podczas zajęć. Celem tego działania było uzyskanie konstrukcji nośnej organizacyjnie oraz informacyjnie. Istotą rozbicia opisu zajęć na czynności uczniów i nauczyciela jest uzyskanie możliwie jak największej przejrzystości, co ma ułatwić nauczycielom pełne zrozumienie intencji autora Konspektu.

Lista czynności uczniów i działań trenera pozwala na inspirowanie do podejmowania i uruchamiania dodatkowych czynności. Łatwo można również, dzięki takiemu układowi, wskazać te czynności, które podejmowane przez uczniów mogłyby doprowadzić do osiągania przez nich określonych celów, np. stanów emocjonalno-motywacyjnych.

» **Zadania rozszerzające** – polecenia możliwe do samodzielnego wykonania przez ucznia (bądź ze wsparciem), zasadniczo przeznaczone do realizacji już poza zajęciami (chociaż można podjąć je z grupą zaawansowaną).

Bardzo istotną jest kwestia dostosowania Konspektów-scenariuszy i powiązanych z nimi Implementacji do oczekiwań, potrzeb, możliwości i preferencji konkretnej grupy uczniowskiej oraz priorytetów danego trenera. Przy założeniu, że wykorzystuje się najlepsze z wzorców i stosuje innowacyjne, wypracowane rekomendacje metodyczne oraz taksonomie.

Byłoby rzeczą niefortunną, gdyby trenerzy starali się realizować wszystkie Konspekty wraz z Implementacjami wyłącznie w takiej formie, w jakiej zostały im udostępnione. Mogłoby to wpływać negatywnie na zaangażowanie się uczniów i motywację do podejmowania kolejnych wyzwań, a także powodować zniechęcenie i znużenie samych nauczycieli. Dlatego zachęcamy trenerów do wybierania spośród oferty tych materiałów, które uznają za najwartościowsze, bądź modyfikowania ich według lokalnych potrzeb, czy wreszcie – do proponowania własnych pomysłów na moduły zajęć infotechnicznych i dzielenia się nimi z całą społecznością na Serwisie e-Swoi.pl.

Niech celem użytkowników i odbiorców będzie czerpanie indywidualnych korzyści z wdrożenia innowacyjnej Strategii, szukanie własnej ścieżki najbardziej efektywnego wykorzystania Programu nauczania-uczenia się, wzbudzanie w swym środowisku zaangażowania, a zwłaszcza motywacji i pasji u uczniów – czego serdecznie życzymy i do czego gorąco zachęcamy.

Moduły 0

Zajęcia dla szkół podstawowych

Środowisko i zagadnienia

01

SRU, JAlbum, TuxPaint: obróbka zdjęć, grafika

02

Calc, Mozilla, Google: tabele, diagramy, Internet

03

S4A, Arduino: sygnalizacja, sterowanie, losowanie

04

S4A, Arduino: wyświetlanie, przełączanie, regulacja

Konspekt-scenariusz **Modułu 01.1**

Adam Jurkiewicz

Temat: Wprowadzenie do środowiska SRU

Nazwa implementacji: Instrukcje, Szkolny Remiks Ubuntu

Opis implementacji: Podstawowe informacje o komputerach, higiena pracy z komputerem. Środowisko graficzne XFCE.

Proponowany czas realizacji: 45 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie wiedzy o zastosowaniach komputerów;
- kształtowanie postaw bezpiecznego korzystania z komputera;
- rozwijanie umiejętności korzystania z aplikacji komputerowych;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu podstawowych pojęć dotyczących urządzeń komputerowych;
- potrafi korzystać z podstawowych zastosowań systemu operacyjnego;
- posiada wiedzę z zakresu zasad bezpiecznego korzystania z komputera;
- posiada wiedzę z zakresu podstawowych operacji z tekstem i z plikami.

Metody, działania:
- pogadanka i dyskusja;
- prezentacja – zapoznanie z oprogramowaniem SRU_SWOI;
- prezentacja – regulamin;
- metoda ćwiczebna – odczytanie pliku, zapisanie pliku w katalogu domowym.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- omówi pojęcia: komputer, okablowanie, urządzenie peryferyjne;
- wykorzysta podstawowe funkcje obsługi Linuksa;
- potrafi rozpoznać różne ikony związane z oprogramowaniem;
- potrafi skorzystać z różnych sposobów bezpiecznego kopiowania plików, zakładania katalogów w Linuksie.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Wprowadza do tematu, opowiada o różnych urządzeniach, połączeniach, kablach – pogadanka.	• Pojęcia: komputer, sieć.
Odpowiadają na pytanie nauczyciela.	Zachęca uczennice/uczniów do zastanowienia się, czy potrafią zdefiniować różne połączenia kablowe i urządzenia peryferyjne komputera.	
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Krótko wyjaśnia regulamin pracowni, pokazuje szatę graficzną systemu SRU_SWOI, przekazuje regulamin.	• Linux SRU_SWOI.
Uruchamiają aplikacje. Zakładają katalogi i kopiują pliki.	Wspiera uczniów w rozpoznaniu interfejsu graficznego XFCE.	• Menadżer plików Nautilus, edytor tekstów Libre Office Writer.

Konspekt-scenariusz **Modułu 01.2**

Natalia Walter

Temat: Album - wspomnienia z wakacji

Nazwa implementacji: Album zdjęć

Opis implementacji: Tworzenie albumu zdjęć w formie strony WWW (program JAlbum), możliwego do przeglądania w dowolnej przeglądarce internetowej. Wszystkie zdjęcia w albumie powinny być opisane (komentarz), wykadrowane, usunięty efekt czerwonych oczu, mora oraz inne defekty. Przy pomocy programu graficznego (Gimp 2.6) ze zdjęć należy usunąć postaci i elementy zbędne oraz dokonać niezbędnych korekt. Uczniowie powinni wcześniej przygotować i przynieść na nośniku około 20 wybranych przez siebie fotografii (awaryjnie nauczyciel powinien posiadać kilka zestawów zdjęć).

Proponowany czas realizacji: 45 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie umiejętności obróbki fotograficznej przy pomocy edytora grafiki;
- rozwijanie postaw związanych z ochroną i porządkowaniem danych;

b) szczegółowe: Uczennica/uczeń...

- zapoznanie z podstawowymi sposobami i metodami obróbki fotografii cyfrowej;
- zaznajomienie z podstawowymi narzędziami programu do grafiki rastrowej;
- wykształcenie umiejętności i nawyku porządkowania zbioru fotografii cyfrowej w formie albumu zdjęć;
- zachęcenie do publikowania zdjęć dotyczących ważnych wydarzeń edukacyjnych;
- uwrażliwienie na problemy dotyczące publikowania intymnych lub obraźliwych zdjęć w Internecie.

Materiał nauczania-uczenia się:

- przeprowadzanie podstawowej korekty zdjęć (JAlbum) oraz opatrzenie ich komentarzem.

Metody, działania:

- pogadanka i dyskusja;
- prezentacja multimedialna – krótki pokaz wzorcowego albumu zdjęć, zapoznanie z oprogramowaniem;
- metoda ćwiczebna – przygotowywanie albumu, obróbka cyfrowa fotografii.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- wskaże i opisze podstawowe sposoby i metody obróbki fotografii cyfrowej;
- wykorzysta podstawowe narzędzia programu do grafiki rastrowej;
- uporządkuje zbiór fotografii cyfrowej w formie albumu zdjęć;
- opublikuje zdjęcia w formie albumu na stronie WWW, dotyczące ważnych wydarzeń edukacyjnych lub społecznych;
- dokona świadomej selekcji zdjęć.

Czynności uczniów	Działania nauczyciela	Materiał
Współuczestniczą w prezentacji, a następnie uczestniczą w pogadance i dyskusji prowadzonej przez nauczyciela.	Zachęca uczniów do krytycznej analizy posiadanych fotografii cyfrowych. Wskazuje na potencjalne błędy na fotografiach oraz sposoby ich usunięcia.	Każdy uczeń musi przynieść nośnik ze zdjęciami z wakacji (min. 20 zdjęć). Nauczyciel powinien mieć przygotowane awaryjne zestawy zdjęć oraz zdjęcia wymagające korekty pogłębionej.
Tematycznie i chronologicznie porządkują zdjęcia i umieszczają je w odpowiednich folderach.	Służy radą, pomocą w pracy.	
Tworzą własne albumy zdjęć z ważnego wydarzenia edukacyjnego, rodzinnego lub kulturowego. Przeprowadzają podstawową korektę zdjęć oraz dodają komentarze (JAlbum). Modyfikują wybrane zdjęcia w programie graficznym (Gimp): usuwają elementy zbędne (patch, klonowanie), stosują efekty oraz filtry, przekolorowują wybrane elementy na zdjęciu (wybielanie zębów, wysmuklanie postaci, zmiana koloru oczu).	Sugeruje, jakie działania muszą podjąć uczennice i uczniowie, by ich album był atrakcyjny.	• JAlbum 9.4 dla Linuksa, Gimp 2.6 dla Linuksa.

Konspekt-scenariusz **Modułu 01.3**

Adam Jurkiewicz

Temat: Tworzenie reklamy ośrodka wczasowego

Nazwa implementacji: Reklama

Opis implementacji: Praca z edytorem graficznym i tekstowym.

Proponowany czas realizacji: 45 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- rozwijanie umiejętności korzystania z aplikacji komputerowych;
- formowanie kreatywności i motywacji do pracy z aplikacjami komputerowymi;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu podstawowych pojęć dotyczących edycji tekstu i grafiki;
- kształtuje umiejętności w zakresie podstawowych operacji z tekstem, grafiką i z plikami;
- odczuwa satysfakcję z pracy z aplikacjami komputerowymi.

Metody, działania:

- pogadanka i dyskusja;
- prezentacja – zapoznanie z oprogramowaniem SRU_SWOI;
- metoda ćwiczebna – edycja tekstów, wczytanie grafiki, opatrzenie kolorowym tekstem.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- omówi podstawowe pojęcia związane z edytorami tekstu i grafiki, a także formatami plików;
- potrafi wstawić do edytora tekstów znak graficzny;
- wykorzysta podstawowe funkcje obsługi edytora tekstów;
- potrafi uruchomić i wykorzystać FontWork.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Wprowadza do tematu, wykorzystuje grafikę i tekst – pogadanka.	• Pojęcia: grafika, plakat, reklama.
Refleksja – odpowiadają na pytanie nauczyciela.	Zachęca uczennice/uczniów do zastanowienia się, jak można reklamować/zachęcać do wyjazdu.	
Zadają pytania, wyjaśniają wątpliwości. Uruchamiają LibreOffice Writer, wczytują zdjęcie/obrazek, tworzą napis w FontWork.	Pomaga dzieciom uruchomić edytor tekstów, w nim wczytać grafikę i opatrzyć tekstem z FontWork.	• SRU_SW01, edytor graficzny TuxPaint, menadżer Nautilus, edytor tekstów LibreOffice.

Konspekt-scenariusz **Modułu 02.1**

Hanna Bielawska

Temat: Oszczędzaj energię elektryczną

Nazwa implementacji: Diagramy kolumnowe

Opis implementacji: Tworzenie diagramów słupkowych na podstawie danych zebranych w Ankiecie – miesięczne zużycie prądu.

Proponowany czas realizacji: 45 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie umiejętności tworzenia diagramów;
- formowanie cech partycypacji i współpracy w grupie;

b) szczegółowe: Uczennica/uczeń…
- nabywa wiedzę z zakresu zbierania danych;
- posługuje się arkuszem kalkulacyjnym;
- zna pojęcie wykres kolumnowy;
- generuje wykres kolumnowy na podstawie zebranych danych;
- doskonali wygenerowany przez siebie wykres;
- wyciąga wnioski z analizy diagramów.

Metody, działania:
- pogadanka i dyskusja;
- prezentacja multimedialna – zapoznanie z oprogramowaniem;
- metoda ćwiczebna – przygotowywanie diagramów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- omówi sposoby gromadzenia danych;
- wpisze poprawne formuły obliczeniowe;
- wygeneruje odpowiedni wykres kolumnowy;
- prawidłowo sformatuje wykres.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Prowadzi dyskusję na temat zbierania i gromadzenia danych.	• Dane empiryczne, podstawowe pojęcia – tabele, komórki.
Współuczestniczą w pokazie, zadają pytania, zgłaszają wątpliwości.	Prezentuje funkcje i narzędzia oprogramowania.	• Program Calc LibreOffice dla Linuksa; • Przykładowe rachunki za zużycie energii elektrycznej.
Przygotowują tabelę miesięcznego zużycia prądu w gospodarstwach domowych uczniów ze swojej grupy – na podstawie przyniesionych na zajęcia rachunków za energię elektryczną.	Przedstawia uczniom wzór tabeli w arkuszu kalkulacyjnym Calc.	
Generują wykres kolumnowy na podstawie zgromadzonych danych. Formatują wykres.	Omawia metodę generowania wykresów z danych zebranych w tabeli.	• Pojęcia – formuły, wykres kolumnowy.
Analizują zużycie prądu na osobę w jednym gospodarstwie domowym – projektują rozbudowaną tabelę. Obliczają średnią zużycia prądu na osobę – tworzą formuły w arkuszu kalkulacyjnym. Porównują swoje wyniki ze średnimi.	Proponuje uczniom rozbudowanie tabeli o dane dotyczące liczby osób w gospodarstwie. Wyjaśnia pojęcie średniego zużycia prądu na osobę.	

Konspekt-scenariusz **Modułu 02.2**

Hanna Bielawska

Temat: Zdrowy tryb życia

Nazwa implementacji: Diagramy kołowe

Opis implementacji: Tworzenie diagramów kołowych na podstawie danych zebranych w Ankiecie dotyczącej zdrowego trybu życia – odpowiedź na pytanie: „Co pijesz na śniadanie?"

Proponowany czas realizacji: 45 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy z arkuszem kalkulacyjnym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć informatycznych;
- kształtowanie umiejętności pracy w grupie;
- kształtowanie nawyków zdrowego odżywiania się;

b) szczegółowe: Uczennica/uczeń…
- gromadzi dane w postaci tabeli;
- generuje wykres kołowy na podstawie danych zebranych w tabeli;
- formatuje wykres;
- wyciąga wnioski z danych przedstawianych graficznie.

Metody, działania:
- zadania wprowadzające – przeprowadzenie badania ankietowego;
- pogadanka wprowadzająca;
- praca samodzielna – opracowanie tabeli w arkuszu, generowanie wykresu.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- zbiera dane ankietowe i wprowadza je do odpowiedniej tabeli;
- wygeneruje wykresy z wykorzystaniem kreatora;
- prawidłowo sformatuje wykresy;
- wykorzysta informacje z wykresu do analiz.

Czynności uczniów	Działania nauczyciela	Materiał
Biorą aktywny udział w dyskusji. Wybierają zagadnienie, które chcą zinterpretować graficznie.	Przedstawia wyniki ankiety przeprowadzonej przez uczniów na temat zdrowego trybu życia, analizuje zadawane pytania (m. in. Co pijesz na śniadanie?)	• Ankiety przeprowadzone przez uczniów.
Uruchamiają program – arkusz kalkulacyjny, projektują tabelę danych.	Wspomaga pracę uczniów, przypomina o zapisaniu pliku.	• Program Calc LibreOffice dla Linuksa.
Generują wykres kołowy.	Omawia metodę generowania wykresów – zastosowanie kreatora wykresów.	• Pasek narzędziowy – kreator wykresów.
Formatują wykres.	Zwraca uwagę na odpowiednie dla rodzaju wykresów etykiety danych.	• Pasek narzędziowy – kreator wykresów.
Wykonują ćwiczenia dodatkowe – interpretacja danych zobrazowanych na wykresie, sortowanie danych w tabelach, obliczanie liczby uczniów biorących udział w ankiecie.	Zadaje pytania dodatkowe, nawiązujące do zdrowego trybu odżywiania się.	• Wykonane wykresy, pasek narzędziowy arkusza kalkulacyjnego.
Zmieniają typ wykresu.	Prowokuje dyskusję na temat dostosowania typu wykresu do rodzaju przedstawianych danych.	

Konspekt-scenariusz **Modułu 02.3**

Adam Jurkiewicz

Temat: Wyszukiwanie informacji w sieci

Nazwa implementacji: Internet

Opis implementacji: Praca z przeglądarkami Internetu i serwisami wyszukującymi.

Proponowany czas realizacji: 45 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- rozwijanie umiejętności korzystania z aplikacji komputerowych;
- formowanie kreatywności i motywacji do pracy z aplikacjami komputerowymi;

b) szczegółowe: Uczennica/uczeń…
- posiada podstawową wiedzę z zakresu sieci WWW;
- kształtuje umiejętności w zakresie korzystania z wyszukiwarek internetowych;
- odczuwa satysfakcję z pracy z aplikacjami komputerowymi w sieci WWW.

Metody, działania:
- pogadanka i dyskusja;
- prezentacja – zapoznanie z oprogramowaniem SRU_SWOI i serwisami wyszukującymi;
- metoda ćwiczebna – obsługa różnych przeglądarek WWW.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- potrafi uruchomić przeglądarkę WWW;
- omawia pojęcia dotyczące sieci komputerowych związanych z serwisami wyszukującymi i śladach, jakie pozostawiamy w Internecie;
- potrafi wskazać i użyć różne wyszukiwarki;
- potrafi odróżnić reklamę od wyników wyszukiwania.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Wprowadza do tematu: Internet, encyklopedie – pogadanka.	• Pojęcia: sieć, Internet, strona WWW, encyklopedia.
Refleksja - odpowiadają na pytanie nauczyciela.	Zachęca uczennice/uczniów do zastanowienia się, czy wiedzą, jak szukać informacji w sieci Internet, czy poza serwisem Google znają inne.	• Google, IxQuick, Wikipedia.
Zadają pytania, wyjaśniają wątpliwości. Uruchamiają przeglądarkę Firefox, wyszukują hasło w serwisie Wikipedii, wyszukują informację w sieci za pomocą wyszukiwarki IxQuick, sprawdzają dostępność materiałów w sieci i reputację stron WoT.	Pomaga dzieciom uruchomić różne przeglądarki, wyszukać to samo w różnych serwisach.	• SRU_SW01, Mozilla Firefox + Web Of Trust, Midori, Chromium, Google.com, Wikipedia, Yandex.com

Konspekt-scenariusz **Modułu 03.1**

Krzysztof Bytow

Temat: Programowa sygnalizacja diodą LED

Nazwa implementacji: LED

Opis implementacji: Stworzenie kodu do sterowania diodą elektroluminescencyjną w środowisku S4A.

Proponowany czas realizacji: 45 min

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- formowanie kreatywności i sprawności w montowaniu modułów-interfejsów;

b) szczegółowe: Uczennica/uczeń…
- rozwija umiejętności obsługi środowiska Scratch S4A i korzystania z jego funkcji;
- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- rozwija umiejętność współpracy;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny.

Metody, działania:
- pogadanka i dyskusja;
- metoda ćwiczebna – zestawienie i oprogramowanie układów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- omówi podstawowe pojęcia: mikrokontroler, dioda elektroluminescencyjna;
- wykorzysta podstawowe funkcje programu Scratch S4A;
- stworzy kod do sterowania diodą elektroluminescencyjną;
- angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Wprowadza do tematu, prezentuje układ moduł-interfejs oraz środowisko programistyczne, na którym będą prowadzone ćwiczenia. Omawia budowę i zasadę działania diody elektroluminescencyjnej. Pokaz sterowania diodą.	• Pojęcia: mikrokontroler, dioda elektroluminescencyjna; • Prezentacja multimedialna; • Filmy dostępne w serwisie http://www.youtube.com/ hasła kluczowe: arduino; arduino cube; arduino led.
Próbują najpierw samodzielnie, a potem przy wsparciu nauczyciela stworzyć kod i uruchomić układ.	Rozdaje uczennicom i uczniom zadania do wykonania przy pomocy programu Scratch S4A i naprowadza ich na właściwe rozwiązanie.	• Program Scratch S4A dla Linuksa; • Arduino (z wgranym kodem do obsługi S4A) lub kompatybilny układ + przewód USB typu A-B; • Tutoriale: http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Mikrokontroler http://e-swoi.pl/wiki/article/arduino-podstawy/ http://e-swoi.pl/wiki/article/mechatronika-faq/ http://arduino.cc/ http://s4a.cat/

Konspekt-scenariusz **Modułu 03.2**

Krzysztof Bytow

Temat: Sterowanie diodą LED z klawiatury

Nazwa implementacji: Sterowanie klawiaturą w środowisku Scratch S4A

Opis implementacji: Stworzenie kodu do sterowania diodą elektroluminescencyjną z wykorzystaniem klawiatury w środowisku S4A.

Proponowany czas realizacji: 45 min

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie umiejętności konstruowania algorytmów;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- formowanie kreatywności i sprawności w montowaniu modułów-interfejsów;

b) szczegółowe: Uczennica/uczeń…
- rozwija umiejętności obsługi środowiska Scratch S4A i korzystania z jego funkcji;
- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- rozwija umiejętność współpracy;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny.

Metody, działania:
- pogadanka i dyskusja;
- metoda ćwiczebna – zestawienie i oprogramowanie układów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- omówi podstawowe pojęcia: symbole matematyczne, mikrokontroler, dioda elektroluminescencyjna;
- wykorzysta podstawowe funkcje programu Scratch S4A;
- stworzy kod do sterowania diodą elektroluminescencyjną z wykorzystaniem klawiatury;
- angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Wprowadza do tematu, prezentuje układ moduł-interfejs z zaimplementowanym kodem do sterowania diodą. Pokaz reakcji na wciskanie klawiszy.	• Pojęcia: mikrokontroler, klawiatura, symbole matematyczne; • Prezentacja multimedialna; • Filmy dostępne w serwisie http://www.youtube.com/ hasła kluczowe: arduino led.
Próbują najpierw samodzielnie, a potem przy wsparciu nauczyciela stworzyć kod i uruchomić układ.	Rozdaje uczennicom i uczniom zadania do wykonania przy pomocy programu Scratch S4A i naprowadza ich na właściwe rozwiązanie.	• Program Scratch S4A dla Linuksa; • Arduino (z wgranym kodem do obsługi S4A) lub kompatybilny układ + przewód USB typu A-B; • Tutoriale: http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://e-swoi.pl/wiki/article/arduino-podstawy/ http://e-swoi.pl/wiki/article/mechatronika-faq/ http://s4a.cat/

Konspekt-scenariusz **Modułu 03.3**

Krzysztof Bytow

Temat: Losowy czas świecenia diody LED

Nazwa implementacji: Wybór losowy

Opis implementacji: Stworzenie kodu do losowego czasu świecenia diodą elektroluminescencyjną w środowisku S4A.

Proponowany czas realizacji: 45 min

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- formowanie kreatywności i sprawności w montowaniu modułów-interfejsów;

b) szczegółowe: Uczennica/uczeń…
- rozwija umiejętności obsługi środowiska Scratch S4A i korzystania z jego funkcji;
- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- rozwija umiejętność współpracy;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny.

Metody, działania:
- pogadanka i dyskusja;
- metoda ćwiczebna – zestawienie i oprogramowanie układów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- omówi podstawowe pojęcia: zmienne, mikrokontroler, dioda elektroluminescencyjna;
- wykorzysta podstawowe funkcje programu Scratch S4A;
- deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
- stworzy kod do sterowania diodą elektroluminescencyjną, wprowadzając element losowy;
- angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Wprowadza do tematu, prezentuje układ moduł-interfejs z wgranym kodem implementacji. Pokaz losowego sterowania diodą.	• Pojęcia: zmienne, mikrokontroler, dioda elektroluminescencyjna, sterowanie losowe; • Prezentacja multimedialna; • Filmy dostępne w serwisie http://www.youtube.com/ hasła kluczowe: arduino led.
Próbują najpierw samodzielnie, a potem przy wsparciu nauczyciela stworzyć kod i uruchomić układ.	Rozdaje uczennicom i uczniom zadania do wykonania przy pomocy programu Scratch S4A i naprowadza ich na właściwe rozwiązanie.	• Program Scratch S4A dla Linuksa; • Arduino (z wgranym kodem do obsługi S4A)lub kompatybilny układ + przewód USB typu A-B; • Tutoriale: http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Mikrokontroler http://pl.wikipedia.org/wiki/Zmienna_(informatyka) http://e-swoi.pl/wiki/article/arduino-podstawy/ http://e-swoi.pl/wiki/article/mechatronika-faq/ http://s4a.cat/

Konspekt-scenariusz **Modułu 03.4**

Krzysztof Bytow

Temat: Wyświetlanie stanu przycisku Button

Nazwa implementacji: Button

Opis implementacji: Stworzenie kodu wyświetlającego tekst, wskutek zmiany stanu na wejściu cyfrowym w środowisku S4A.

Proponowany czas realizacji: 45 min

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- formowanie kreatywności i sprawności w montowaniu modułów-interfejsów;

b) szczegółowe: Uczennica/uczeń…
- rozwija umiejętności obsługi środowiska Scratch S4A i korzystania z jego funkcji;
- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- rozwija umiejętność współpracy;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny.

Metody, działania:
- pogadanka i dyskusja;
- metoda ćwiczebna – zestawienie i oprogramowanie układów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- omówi podstawowe pojęcia: mikrokontroler, button, opornik, wejście/wyjście cyfrowe;
- wykorzysta podstawowe funkcje programu Scratch S4A;
- samodzielnie montuje i uruchamia przykładowe układy na podstawie schematów;
- stworzy kod do obsługi buttona;
- angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Wprowadza do tematu, prezentuje układ zaimplementowanym kodem. Omawia budowę i zasadę działania buttona. Omawia rolę opornika. Pokaz działania układu.	• Pojęcia: button, opornik, wejście/wyjście cyfrowe, mikrokontroler;
Próbują najpierw samodzielnie, a potem przy wsparciu nauczyciela stworzyć kod i uruchomić układ.	Rozdaje uczennicom i uczniom zadania do wykonania przy pomocy programu Scratch S4A i naprowadza ich na właściwe rozwiązanie.	• Program Scratch S4A dla Linuksa; • Arduino (z wgranym kodem do obsługi S4A) lub kompatybilny układ + przewód USB typu A-B; • Rezystor 220 Ω; • Button (mały przycisk); • Przewody połączeniowe; • Płytka montażowa; • Tutoriale: http://pl.wikipedia.org/wiki/Mikrokontroler http://e-swoi.pl/wiki/article/arduino-podstawy/ http://e-swoi.pl/wiki/article/mechatronika-faq/ http://arduino.cc/ http://s4a.cat/ http://pl.wikipedia.org/wiki/Opornik http://arduino.cc/en/Tutorial/Button http://arduino.cc/en/Tutorial/DigitalPins

Konspekt-scenariusz **Modułu 03.5**

Krzysztof Bytow

Temat: Przełączanie diody LED przyciskiem

Nazwa implementacji: Button i LED

Opis implementacji: Stworzenie kodu sterującego diodą elektroluminescencyjną z wykorzystaniem buttona w środowisku S4A.

Proponowany czas realizacji: 45 min

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- formowanie kreatywności i sprawności w montowaniu modułów-interfejsów;

b) szczegółowe: Uczennica/uczeń…
- rozwija umiejętności obsługi środowiska Scratch S4A i korzystania z jego funkcji;
- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- rozwija umiejętność współpracy;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny.

Metody, działania:
- pogadanka i dyskusja;
- metoda ćwiczebna – zestawienie i oprogramowanie układów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- omówi podstawowe pojęcia: mikrokontroler, button, opornik, wejście/wyjście cyfrowe;
- wykorzysta podstawowe funkcje programu Scratch S4A;
- samodzielnie montuje i uruchamia przykładowe układy na podstawie schematów;
- stworzy kod do sterowania diodą elektroluminescencyjną z wykorzystaniem buttona;
- angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Wprowadza do tematu, prezentuje układ moduł-interfejs oraz środowisko programistyczne, na którym będą prowadzone ćwiczenia. Omawia budowę i zasadę działania diody elektroluminescencyjnej w połączeniu z buttonem. Pokaz sterowania diodą.	• Pojęcia: button, opornik, wejście wyjście cyfrowe, mikrokontroler, dioda elektroluminescencyjna; • Prezentacja multimedialna; • Filmy dostępne w serwisie http://www.youtube.com/ hasła kluczowe: arduino cube; arduino led.
Próbują najpierw samodzielnie, a potem przy wsparciu nauczyciela stworzyć kod i uruchomić układ.	Rozdaje uczennicom i uczniom zadania do wykonania przy pomocy programu Scratch S4A i naprowadza ich na właściwe rozwiązanie.	• Program Scratch S4A dla Linuksa; • Arduino (z wgranym kodem do obsługi S4A) lub kompatybilny układ + przewód USB typu A-B; • Rezystor 220 Ω; • Button (mały przycisk); • Przewody połączeniowe; • Płytka montażowa; • Tutoriale: http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Mikrokontroler http://e-swoi.pl/wiki/article/arduino-podstawy/ http://e-swoi.pl/wiki/article/mechatronika-faq/ http://arduino.cc/ http://s4a.cat/ http://arduino.cc/en/Tutorial/DigitalPins http://pl.wikipedia.org/wiki/Opornik http://arduino.cc/en/Tutorial/Button

Konspekt-scenariusz **Modułu 03.6**

Krzysztof Bytow

Temat: Programowa regulacja jasności diody LED

Nazwa implementacji: Jasno i ciemno

Opis implementacji: Wykorzystanie wyjścia PWM do sterowania diodą elektroluminescencyjną w środowisku S4A.

Proponowany czas realizacji: 45 min

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- formowanie kreatywności i sprawności w montowaniu modułów-interfejsów;

b) szczegółowe: Uczennica/uczeń…
- rozwija umiejętności obsługi środowiska Scratch S4A i korzystania z jego funkcji;
- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- rozwija umiejętność współpracy;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny.

Metody, działania:
- pogadanka i dyskusja;
- metoda ćwiczebna – zestawienie i oprogramowanie układów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- omówi podstawowe pojęcia: mikrokontroler, dioda elektroluminescencyjna, opornik;
- wykorzysta podstawowe funkcje programu Scratch S4A;
- stworzy kod do sterowania jasnością diody elektroluminescencyjnej;
- zmontuje i uruchomi przykładowe układy na podstawie schematów;
- angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Wprowadza do tematu, prezentuje układ moduł-interfejs oraz środowisko programistyczne, na którym będą prowadzone ćwiczenia. Omawia budowę i zasadę działania diody elektroluminescencyjnej. Wyjaśnia proces zmiany jasności diody. Pokaz sterowania diodą.	• Pojęcia: opornik, mikrokontroler, dioda elektroluminescencyjna, modulacja szerokości impulsu; • Prezentacja multimedialna; • Filmy dostępne w serwisie http://www.youtube.com/ hasła kluczowe: arduino PWM led;
Próbują najpierw samodzielnie, a potem przy wsparciu nauczyciela stworzyć kod i uruchomić układ.	Rozdaje uczennicom i uczniom zadania do wykonania przy pomocy programu Scratch S4A i naprowadza ich na właściwe rozwiązanie.	• Program Scratch S4A dla Linuksa; • Arduino (z wgranym kodem do obsługi S4A) lub kompatybilny układ + przewód USB typu A-B; • Rezystor 220 Ω; • Przewody połączeniowe; • Płytka montażowa; • Tutoriale: http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Mikrokontroler http://e-swoi.pl/wiki/article/arduino-podstawy/ http://e-swoi.pl/wiki/article/mechatronika-faq/ http://arduino.cc/en/Tutorial/PWM http://arduino.cc/ http://s4a.cat/

Moduły A

Zajęcia dla szkół gimnazjalnych

Środowisko i zagadnienia

A1

Linux, SRU: awatary, graffiti, algorytm

A2

Scratch: animacja, gra, labirynt, konwersja

A3

S4A, Arduino: interfejs, czujnik, wskaźnik, gra

Konspekt-scenariusz **Modułu A1.1**

Adam Jurkiewicz

Temat: Wprowadzenie do środowiska SRU

Nazwa implementacji: Poszukiwanie skarbów

Opis implementacji: Podstawowe informacje o użytkowaniu SRU, wyszukiwanie elementów potrzebnych do kolejnych modułów.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- ukształtowanie wiedzy o użytkowaniu SRU;
- rozwijanie kreatywności i innowacyjności;
- rozwijanie umiejętności poruszania się w środowisku Linux;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu podstawowych pojęć dotyczących plików i katalogów;
- potrafi obsługiwać podstawowe programy w Linuksie;
- posiada wiedzę z zakresu licencji Creative Commons;
- rozwinie umiejętności szukania informacji w sieci Internet z naciskiem na zachowanie praw autorskich w licencjach CC.

Metody, działania:

- pogadanka i dyskusja, analogie między książkami na półce a plikami na dysku w katalogach;
- prezentacja – zapoznanie z oprogramowaniem SRU;
- prezentacja – menadżer plików;
- metoda ćwiczebna – przeszukiwanie serwisu flickr.com, zapisanie pliku w katalogu domowym;
- metoda ćwiczebna – szukanie czcionki i jej pliku, skopiowanie pliku do katalogu domowego.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- omówi pojęcia: dysk, katalog, plik;
- wykorzysta podstawowe funkcje obsługi Linuksa;
- omówi licencje Creative Commons;
- potrafi skorzystać z różnych sposobów kopiowania plików, zakładania katalogów w Linuksie;
- znajdzie w Internecie ciekawy obrazek z zachowaniem licencjonowania;
- znajdzie ciekawe zdjęcie w Internecie i czcionkę w systemie do dalszego wykorzystania.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Wprowadza do tematu, opowiada o dyskach, katalogach i plikach – analogia do pokoju, półek, książek – pogadanka.	Pojęcia: dysk, katalog, plik.
Refleksja. Odpowiadają na pytanie nauczyciela.	Zachęca uczennice/uczniów do zastanowienia się, czy uważają Linuksa za trudny system.	Linux SRU, menadżer plików Dolphin, Gnome FontManager; informacje o katalogach w sieci: http://www.universalsubtitles.org/pl/videos/Bg0y3OFhvcCi/info/what-is-linux-a-n00b-to-n00b-explanation/ http://pl.wikipedia.org/wiki/Katalog_domowy oraz http://pl.wikipedia.org/wiki/Katalog_%28system_plik%C3%B3w%29 oraz http://www.dobreprogramy.pl/Struktura-drzewa-katalogow-systemu-Linux,Artykul,11405.html oraz http://czytelnia.ubuntu.pl/index.php/2007/07/25/katalogi-w-linuksie/; Licencje CC: http://creativecommons.pl/poznaj-licencje-creative-commons/; serwis http://www.flickr.com
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Pokazuje podstawowe oprogramowanie w SRU; pokazuje, że Linux nie różni się wiele od innych systemów (wszędzie operujemy na plikach); tłumaczy kwestie licencji, w szczególności licencji Creative Commons – prezentacja.	
Przeszukują serwis flickr.com, zapisują pliki w katalogu domowym; szukają czcionki i jej pliku, uczniowie kopiują plik czcionki oraz zdjęcie do /home/uczen/Obrazy/materiały_graficzne.	Sugeruje, jakie działania muszą podjąć uczennice i uczniowie, by odnaleźć skarb – ciekawą ich zdaniem czcionkę i ciekawe zdjęcie w serwisie flickr.com (komputer/pingwin/logo Linuksa).	

Konspekt-scenariusz **Modułu A1.2**

Natalia Walter

Temat: Tworzenie awatarów

Nazwa implementacji: Obraz wektorowy

Opis implementacji: Tworzenie obiektów wektorowych charakteryzujących ucznia (awatary), tworzenie elementów potrzebnych do kolejnych modułów.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie umiejętności tworzenia obiektów wektorowych;
- rozwijanie kreatywności i innowacyjności;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu podstawowych pojęć grafiki wektorowej;
- potrafi posługiwać się podstawowymi narzędziami programu do grafiki wektorowej;
- zaprojektuje własny awatar – swój wirtualny graficzny wizerunek;
- określi charakterystyczne dla siebie samego cechy osobowościowe.

Metody, działania:
- pogadanka i dyskusja;
- prezentacja multimedialna – zapoznanie z oprogramowaniem;
- metoda ćwiczebna – przygotowywanie awatarów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- omówi podstawowe pojęcia grafiki wektorowej;
- wskaże różnice pomiędzy grafiką wektorową a rastrową;
- wykorzysta podstawowe narzędzia programu do grafiki wektorowej;
- stworzy określone obiekty wektorowe (awatar, postać z gry komputerowej etc.);
- rozpozna własne cechy osobowościowe i odzwierciedli je w awatarze.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Porównuje grafikę rastrową i wektorową (przypomnienie wiadomości) – prowadzenie pogadanki.	• Grafika wektorowa i rastrowa – podstawowe pojęcia.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Prezentuje funkcje i narzędzia oprogramowania Inkscape – prezentacja multimedialna z omówieniem.	• Program Inkscape dla Linuksa; można skorzystać ze stron http://inkscape.org/doc/basic/tutorial-basic.pl.html, http://inkscape.org/doc/advanced/tutorial-advanced.pl.html oraz http://inkscape.org/doc/shapes/tutorial-shapes.pl.html) oraz przykładowych rysunków w nim przygotowanych;
Tworzą własne awatary, karty do gry oraz postaci do gry komputerowej.	Zachęca uczniów do zastanowienia się, jaki awatar najlepiej by ich odzwierciedlał; wskazuje, do czego awatar zostanie wykorzystany (gra komputerowa, karta do gry). Pokazuje przykładowe awatary przygotowane za pomocą programu Inkscape. Sugeruje, jakie działania muszą podjąć uczennice i uczniowie, by ich awatar był atrakcyjny i estetyczny. **Formułuje zadania:** • Narysuj labirynt (czarne linie prowadzące), który będzie przydatny do następnych zajęć; • Utwórz dowolny wektorowy obraz graficzny o jak największym stopniu trudności.	• Można wykorzystać fragmenty gotowych klipartów: http://openclipart.org/

Konspekt-scenariusz **Modułu A1.3**

Adam Jurkiewicz

Temat: Implementacja Graffiti

Nazwa implementacji: Graffiti

Opis implementacji: Tworzenie grafiki, która będzie projektem graffiti na ścianę budynku 5 x 1,5 m.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- ukształtowanie umiejętności tworzenia graffiti;
- rozwijanie kreatywności, innowacyjności;

b) szczegółowe: Uczennica/uczeń…
- potrafi obsługiwać program Inkscape do grafiki wektorowej;
- rozwinie umiejętności używania licencji CC;
- rozwinie umiejętności tworzenia grafiki komputerowej;
- posiada wiedzę z zakresu podstawowych pojęć grafiki wektorowej i rastrowej.

Metody, działania:
- pogadanka i dyskusja na temat sposobów promocji;
- prezentacja multimedialna – zapoznanie z oprogramowaniem;
- metoda ćwiczebna – przygotowywanie graffiti.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- omówi podstawowe pojęcia grafiki wektorowej;
- wskaże różnice pomiędzy grafiką wektorową a rastrową;
- wykorzysta podstawowe narzędzia programu do grafiki wektorowej;
- stworzy określone obiekty wektorowe i połączy je z rastrowym obrazem.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Porównuje grafikę rastrową i wektorową (przypomnienie wiadomości) – prowadzenie pogadanki.	Grafika wektorowa i rastrowa – podstawowe pojęcia.
Współuczestniczą w pokazie, zadają pytania.	Prezentuje funkcje i narzędzia oprogramowania Inkscape – prezentacja multimedialna z omówieniem, pokazuje przykładowe grafiki wektorowe przygotowane za pomocą programu Inkscape.	• Program Inkscape dla Linuksa; tutoriale Inkscape'a: http://inkscape.org/doc/basic/tutorial-basic.pl.html, http://inkscape.org/doc/advanced/tutorial-advanced.pl.html oraz http://inkscape.org/doc/shapes/tutorial-shapes.pl.html) oraz przykładowe rysunki w nim przygotowane; • Galerie gotowych elementów, np. http://openclipart.org/ oraz zdjęcie i/lub czcionka, zapisane w katalogu na zajęciach;
Tworzą własne graffiti promujące projekt SWOI lub system Linux.	Zachęca uczennice/uczniów do zastanowienia się, co może być zawarte w graffiti. Sugeruje, jakie działania muszą podjąć uczennice i uczniowie, by graffiti spełniało wymagania rozmiaru (5 x 1,5 m przeskalowane do ekranu komputera), użytych kolorów (minimum 4 lub całkowicie czarno-białe) i zawartości merytorycznej (promocja projektu/systemu). **Formułuje zadania:** • Wymyśl i narysuj graffiti – jeśli chcesz, możesz zgłosić pracę na konkurs; • Wybierz odpowiednią licencję CC i opublikuj w Serwisie e-Swoi.pl tę pracę z odpowiednim odniesieniem do licencji.	

Konspekt-scenariusz **Modułu A1.4**

Jarosław Żok, Rafał Brzychcy

Temat: Algorytm na mapie myśli

Nazwa implementacji: Stworzenie algorytmu map myśli

Opis implementacji: Stworzenie struktury koniecznych działań i warunków w postaci prostej mapy myśli w formie nielinearnej notatki zawierającej czynności niezbędne do osiągnięcia stanu docelowego. Mapa myśli przybiera postać planu prostego algorytmu porządkującego kolejne kroki układające się w plan, gdzie jest stan początkowy/wyjściowy (start), warunki determinujące (zmienne) oraz stan docelowy/końcowy (stop).

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- uruchomienie myślenia algorytmicznego przy pomocy konstruowania prostych map myśli;
- rozwijanie umiejętności współpracy z innymi uczennicami i uczniami oraz z nauczycielem;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu podstawowych pojęć algorytmiki;
- potrafi zaplanować i opracowywać wizualną postać działania algorytmu;
- potrafi przedstawić działania algorytmów w odniesieniu do założonego przykładu – np. codziennych czynności mających zastosowanie w życiu;
- potrafi posługiwać się narzędziem do konstruowania map myśli.

Metody, działania:
- pogadanka i dyskusja;
- prezentacja – działanie programu;
- metoda ćwiczebna – tworzenie algorytmu.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- stworzy cztery proste algorytmy w postaci prostych wizualnych przykładów;
- potrafi omówić na konkretnych przykładach pojęcia: algorytm, instrukcja warunkowa, pętla, system pozycyjny binarny a dziesiętny, sortowanie bąbelkowe;
- wykorzysta podstawowe funkcje programu FreeMind i/lub Dia;
- angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Wprowadza do tematu czym jest algorytm; jakie ma zastosowania w życiu; jak zaplanować działania algorytmu w postaci ścieżek na przykładzie prostych czynności.	Wyjaśnienie pojęć: skrypt, wyrażenia, algorytm, pętla, zmienna, instrukcja.
Uczestnictwo w dyskusji w grupie; dają propozycje działań algorytmów.	Objaśnia, na czym będzie polegać praca zespołowa i pyta, jakie warunki muszą być spełnione w danym algorytmie – notuje trafne uwagi na żółtych kartkach. Tłumaczy i wyjaśnia, jak nazywa się ten algorytm i jakie ma zastosowania.	Żółte kartki przyklejane na tablicy, łączone ze sobą relacjami według schematu konkretnego algorytmu.
Współuczestniczą w prezentacji programu do tworzenia map myśli.	Prezentuje program do tworzenia map myśli i przykładowe opracowanie algorytmu na mapie myśli.	Program FreeMind.
Tworzą plan algorytmu na mapie myśli przy wykorzystaniu programu komputerowego.	Pomaga przy przeniesieniu algorytmów na mapę myśli, przy pomocy programu komputerowego.	

Konspekt-scenariusz **Modułu A2.1**

Jarosław Żok

Temat: Animacja biedronki

Nazwa implementacji: Robot-biedronka

Opis implementacji: Poznanie zasad tworzenia programów komputerowych za pomocą instrukcji języka programowania

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrożenie do pracy w środowisku graficznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z tworzeniem programów komputerowych;

b) szczegółowe: Uczennica/uczeń…
- kształtuje umiejętność tworzenia oprogramowania za pomocą instrukcji języka;
- kształtuje umiejętność opisywania algorytmów za pomocą języka naturalnego;
- wykorzystuje elementy programowania zdarzeniowego i obiektowego;
- poznaje strukturę programowania;
- posiada wiedzę z zakresu podstawowych pojęć języków programowania;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:
- programowanie strukturalne;
- programowanie zdarzeniowe;
- instrukcje języka Scratch.

Metody, działania:
- prezentacja zasad programu;
- metoda ćwiczebna – implementacja kodu.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- potrafi objaśnić strukturę programowania;
- potrafi wyjaśnić pojęcia: programowanie zdarzeniowe i obiektowe, pętle, instrukcje warunkowe;
- posługuje się naturalnym językiem przy opisie algorytmów;
- tworzy animację za pomocą elementów programowania zdarzeniowego i obiektowego;
- tworzy prostą implementację graficzną za pomocą instrukcji języka.

Czynności uczniów	Działania nauczyciela	Materiał
	Wyjaśnia strukturę programu, podzieloną na instrukcje.	• Programowanie strukturalne; • Programowanie zdarzeniowe; • Instrukcje języka Scratch; • Pętle; • Instrukcje warunkowe; • Mapy bitowe reprezentujące położenie elementów planszy gry; • Interakcja elementów graficznych na planszy.
Proponują sposób realizacji badania wyjścia poza linię.	Objaśnia, jak element graficzny może poruszać się wzdłuż linii, badając wychodzenie poza linię i korygowanie toru.	
	Pokazuje, jak tworzony jest program w Scratch oraz przekazuje wiedzę na temat tworzenia elementów graficznych.	
Rysują duszka biedronki, toru, po którym porusza się biedronka; implementują ruch biedronki po torze.	Wspiera uczniów, koryguje błędy, naprowadza.	
Implementują zmiany i sprawdzają poprawność jej działania.	Wspiera uczniów, koryguje błędy, naprowadza i motywuje.	

Konspekt-scenariusz **Modułu A2.2**

Natalia Walter

Temat: Moja własna gra komputerowa

Nazwa implementacji: Animacja, Gra logiczna – zgadywanka

Opis implementacji: Stworzenie 1) własnej animacji oraz 2) algorytmu wyszukiwania binarnego z gotowych modułów (Scratch).

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- ukształtowanie umiejętności tworzenia animacji oraz algorytmu wyszukiwania binarnego;
- rozwijanie umiejętności logicznego planowania, kreatywności i innowacyjności;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu podstawowych pojęć algorytmicznych;
- potrafi posługiwać się narzędziem do tworzenia interaktywnej animacji;
- rozwinie umiejętności konstruowania algorytmów;
- rozwinie umiejętności tworzenia animacji interaktywnych.

Metody, działania:

- pogadanka i dyskusja;
- prezentacja multimedialna – zapoznanie z oprogramowaniem;
- metoda ćwiczebna – przygotowywanie animacji interaktywnej.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- omówi podstawowe pojęcia algorytmiczne: skrypt, wyrażenia, algorytm, pętla, wyrażenie warunkowe, zmienna, czujnik, instrukcja;
- wykorzysta podstawowe narzędzia programu Scratch;
- stworzy animację interaktywną;
- współpracuje z innymi uczniami oraz z nauczycielem.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Wprowadza do tematu, przypomina wiadomości na temat tego, czym jest algorytm; zachęca uczniów do zastanowienia się i opisania, czym jest algorytm i po co się go konstruuje. Prosi o scharakteryzowanie ulubionej logicznej gry komputerowej.	Pojęcia: skrypt, wyrażenia, algorytm, pętla, wyrażenie warunkowe, zmienna, czujnik, instrukcja.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Prezentuje funkcje oprogramowania Scratch – prezentacja multimedialna.	• Program Scratch dla Linuksa; • Tutoriale Scratch: http://info.scratch.mit.edu/pl/Support
Tworzą prostą animację (wspólnie z nauczycielem).	Pomaga przy stworzeniu przykładowej animacji.	
Wykonują zadania (algorytmu wyszukiwania binarnego – gry logicznej) z wykorzystaniem awatara z poprzednich zajęć.	Rozdaje uczennicom i uczniom zadania do wykonania przy pomocy programu Scratch i naprowadza ich na właściwe rozwiązanie. **Formułuje zadanie:** • Wybierz awatar stworzony przez Ciebie na poprzednich zajęciach. Napisz program, w którym Twój awatar, za pomocą algorytmu wyszukiwania binarnego, zgadnie liczbę z przedziału 0 do 100, o której pomyślał użytkownik.	

Konspekt-scenariusz **Modułu A2.3**

Natalia Walter

Temat: Labirynt interaktywny

Nazwa implementacji: Labirynt

Opis implementacji: Stworzenie interaktywnej gry - labiryntu

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- ukształtowanie umiejętności stworzenia gry interaktywnej;
- rozwijanie umiejętności logicznego myślenia i planowania, kreatywności i innowacyjności;

b) szczegółowe: Uczennica/uczeń...
- posiada wiedzę z zakresu podstawowych pojęć algorytmicznych;
- rozwinie umiejętność tworzenia interaktywnej animacji;
- rozwinie umiejętność współpracy.

Metody, działania:
- pogadanka i dyskusja;
- metoda ćwiczebna – przygotowywanie animacji interaktywnej.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...
- omówi podstawowe pojęcia, takie jak: pętla, wyrażenie warunkowe, zmienna;
- wykorzysta podstawowe narzędzia programu Scratch;
- wykorzysta wcześniej przygotowany obraz (labiryntu) w przygotowywanej pracy;
- stworzy animację interaktywną;
- współpracuje z innymi uczennicami i uczniami oraz z nauczycielem.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Wprowadza do tematu, przypomina wiadomości na temat tego, czym jest algorytm; omawia rolę funkcji warunkowej i pętli.	Pojęcia: algorytm, pętla, wyrażenie warunkowe, zmienna.
Próbują najpierw samodzielnie, a potem przy wsparciu nauczyciela stworzyć interaktywny labirynt.	Zachęca uczniów do zastanowienia się i opisania, na czym polega poruszanie się w labiryncie. Wspólnie z uczniami ustala, w jaki sposób można poruszać się po labiryncie, co powinno stanowić barierę w poruszaniu się (np. ściany określonego koloru). **Formułuje zadanie:** • Stwórz grę – labirynt interaktywny.	• Program Scratch dla Linuksa; obraz labiryntu przygotowany na wcześniejszych zajęciach; • Tutoriale Scratch: http://info.scratch.mit.edu/pl/Support

Konspekt-scenariusz **Modułu A2.4**

Jarosław Żok

Temat: Różne systemy liczbowe

Nazwa implementacji: Liczyć jak komputer

Opis implementacji: Stworzenie interaktywnej prezentacji, pokazującej jak liczone są wartości między różnymi pozycyjnymi systemami liczbowymi ze szczególnym uwzględnieniem systemu binarnego.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wykorzystanie narzędzi informatycznych, w tym elementów programowania do obliczania wartości między różnymi pozycyjnymi systemami liczbowymi;
- wzbudzenie zainteresowania maszynowym rozwiązywaniem problemów;

b) szczegółowe: Uczennica/uczeń…
- wykorzystuje narzędzia informatyczne, w tym elementy programowania, do tworzenia prezentacji;
- posiada wiedzę z zakresu liczbowych systemów pozycyjnych;
- posiada wiedzę z zakresu funkcjonowania współczesnych maszyn cyfrowych;
- realizuje implementację kodu za pomocą podprogramów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- omówi pojęcia: podstawa, pozycja, cyfra i liczba, pętla, instrukcja warunkowa, zmienna;
- wskaże, w jaki sposób liczby są prezentowane wewnątrz współczesnych maszyn cyfrowych;
- omówi różnice i podobieństwa między systemem binarnym a dziesiętnym;
- stworzy interaktywną prezentację z zakresu liczenia wartości w różnych liczbowych systemach pozycyjnych.

Czynności uczniów	Działania nauczyciela	Materiał
Przygotowują kartki z potęgami liczby 10. Wyjaśniają wątpliwości.	Wyjaśnia pojęcie potęgowania, aby uczniowie mogli potem zastosować je do obliczania wag pozycji cyfr w liczbie (10^0 = 1, 10^1 = 10, 10^2=100 itd).	4 kartki z zapisanymi na narożnikach kolejnymi potęgami liczby 10 (1, 10, 100, 1000).
Zapisują dowolnie wybrane cyfry od 0 do 9 na kartkach i mnożą je przez zapisane na kartce wagi (np. 1234 = 1*10^3 + 2*10^2 + 3*10^1 + 4*10^0).	Wskazuje, w jaki sposób ludzie interpretują liczby, posługując się systemem dziesiętnym.	
Przygotowują kartki z potęgami liczby 2, wyjaśniają wątpliwości.	Wyjaśnia uczniom i uczennicom jak maszyny interpretują cyfry (0 i 1).	4 kartki z zapisanymi na narożnikach kolejnymi potęgami liczby 2 (1, 2, 4, 8).
Zapisują cyfry od 0 do 1 na kartkach i mnożą je, jak w przypadku systemu dziesiętnego – cyfry razy wagi (1011 = 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0), otrzymując wartość dziesiętną liczby zapisanej binarnie.	Wskazuje różnice i podobieństwa między zapisem binarnym a dziesiętnym.	
Tworzą interaktywną prezentację wizualizującą działanie pętli.	**Formułuje zadanie:** • Stworzenie prezentacji pokazującej jak można przekonwertować jeden z systemów liczbowych na dziesiętny. • Tłumaczy na bieżąco działania skryptu (krok po kroku).	• Pętla z wykorzystaniem instrukcji „powtarzaj aż"; • Instrukcja warunkowa z wykorzystaniem instrukcji „jeżeli"; • Programowanie strukturalne wykorzystujące odebranie zdarzenia w instrukcji „Kiedy otrzymam"; • Wywołanie zdarzeń instrukcją „Nadaj"; • Deklaracja, przypisanie i wykorzystanie zmiennych; • Obliczanie wyrażeń .

Konspekt-scenariusz **Modułu A3.1**

Krzysztof Bytow

Temat: Scratch i obsługa modułu-interfejsu

Nazwa implementacji: Poznanie możliwości środowiska Scratch i zestawu modułu-interfejsu

Opis implementacji: Wprowadzenie w świat mikrokontrolerów na przykładzie modułu-interfejsu Arduino oraz jego obsługa w środowisku Scratch (S4A). Prezentacja i wyjaśnienie sposobu zestawiania połączeń na podstawie dokumentacji ilustrującej montaż układów ćwiczeniowych. Wizualizacja działania wybranych elementów zestawu modułu-interfejsu z układem Arduino. Sposoby podłączania, sterowania i programowania podzespołów.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- rozpoznawanie środowiska do programowania wizualnego układów mechatronicznych;
- formowanie kreatywności i sprawności w montowaniu i rozbudowie modułów-interfejsów;
- wzbudzenie satysfakcji z tego, że działa zmontowany własnoręcznie układ elektroniczny;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- ma doświadczenie pracy w środowisku do programowania wizualnego układów mechatronicznych;
- rozwija umiejętności sterowania elementami zestawu modułu-interfejsu;
- rozwija umiejętność współpracy z innymi uczennicami i uczniami oraz z nauczycielem.

Materiał nauczania-uczenia się:

- program S4A (Scratch);
- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- button; dioda elektroluminescencyjna; dioda RGB;
- rezystor 10 kΩ; minimum 3 rezystory 220 Ω.

Metody, działania:

- zajawka inspirująca – pokaz sterowania diodą elektroluminescencyjną;

- zajawka inspirująca – pokaz sterowania diodą z wykorzystaniem buttona;
- zajawka inspirująca – pokaz sterowania diodą RGB;
- prezentacja multimedialna – filmy instruktażowe do implementacji;
- metoda ćwiczebna – zestawienie i oprogramowanie układów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- mówi pojęcia: button, wejście cyfrowe, dioda elektroluminescencyjna, opornik, zmienna, mikrokontroler;
- zmontuje i uruchomi przykładowe układy na podstawie schematów;
- deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
- potrafi obsługiwać środowisko Scratch S4A i zna jego funkcje;
- zna istotę działania oraz sposób podłączania i sterowania podzespołami: dioda elektroluminescencyjna, dioda RGB, przycisk;
- angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Prezentuje układ Arduino, na którym będą prowadzone ćwiczenia. Pokazuje możliwości środowiska – prezentacja multimedialna, filmy instruktażowe. Omawia elementy wchodzące w skład zestawu – pokaz sterowania diodą elektroluminescencyjną, pokaz sterowania diodą z wykorzystaniem buttona, pokaz sterowania diodą RGB.	• Pojęcia: mikrokontroler, dioda elektroluminescencyjna, dioda RGB, button, opornik, wejścia / wyjścia cyfrowe; • Filmy instruktażowe; • Prezentacja multimedialna; • Filmy dostępne w serwisie http://www.youtube.com/ hasła kluczowe: arduino; arduino led; arduino RGB.
	Uruchamia środowisko programistyczne Arduino IDE, objaśniając poszczególne funkcje programu. Pokazuje wstępną konfigurację programu w celu komunikacji między komputerem a modułem. W dalszej części prezentuje, w jaki sposób połączyć Arduino ze Scratchem S4A. Prezentuje wymagany kod do współpracy z oprogramowaniem S4A, a następnie prezentuje możliwości samego środowiska. Dokonuje zestawienia układów i ich uruchomienia. Omawia elementy składowe programu.	• Tutoriale: http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Mikrokontroler http://e-swoi.pl/wiki/article/arduino-podstawy/ http://arduino.cc/ http://s4a.cat/ http://pl.wikipedia.org/wiki/Opornik http://e-swoi.pl/wiki/article/mechatronika-faq/
Wykonują samodzielne zestawienie i oprogramowanie układów.	Zachęca uczniów do samodzielnego podłączenia układu i zaprogramowania. Sugeruje, jakie działania muszą podjąć uczennice i uczniowie, aby ich układ się uruchomił. **Formułuje zadania:** • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	

Konspekt-scenariusz **Modułu A3.2**

Krzysztof Bytow

Temat: Pomiar temperatury i wizualizacja RGB

Nazwa implementacji: Sterowanie elementami z poziomu aplikacji

Opis implementacji: Budowa układu do wizualizacji pomiaru temperatury w środowisku Scratch (S4A). Wykorzystanie funkcji przetwornika analogowo-cyfrowego do budowy układu pomiarowego. Pomiar temperatury i prezentacja pomiarów: w środowisku Scratch (S4A) i przy wykorzystaniu diody RGB.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- rozpoznawanie środowiska do programowania wizualnego układów mechatronicznych;
- formowanie kreatywności i sprawności w montowaniu i rozbudowie modułów-interfejsów;
- rozwinięcie umiejętności sterowania elementami zestawu modułu-interfejsu;
- wzbudzenie satysfakcji z tego, że działa zmontowany własnoręcznie układ elektroniczny;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- ma doświadczenie w pracy w środowisku do programowania wizualnego układów mechatronicznych;
- rozwija umiejętność współpracy z innymi uczennicami i uczniami oraz z nauczycielem.

Materiał nauczania-uczenia się:
- program S4A (Scratch);
- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- czujnik temperatury MCP9700;
- dioda RGB;
- 3 rezystory 220 Ω.

Metody, działania:
- zajawka inspirująca – pokaz odczytu temperatury w środowisku S4A;
- zajawka inspirująca – pokaz wizualizacji odczytu temperatury z wykorzystaniem diody RGB;
- dyskusja dotycząca sposobów i dokładności pomiarów temperatury;
- prezentacja multimedialna – film instruktażowy do implementacji;
- metoda ćwiczebna – zestawienie i oprogramowanie układu do pomiaru temperatury;

» metoda ćwiczebna – zestawienie i oprogramowanie układu do wizualizacji odczytu temperatury z wykorzystaniem diody RGB.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

» prawidłowo obsługuje środowisko programowania graficznego Scratch S4A;
» samodzielnie montuje i uruchamia przykładowe układy na podstawie schematów;
» deklaruje zmienne i przypisuje im wartości;
» prawidłowo łączy i odczytuje wskazania czujnika temperatury;
» trafnie używa zwrotów: *czujnik, przetwornik A/D, dioda RGB;*
» angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w prezentacji i pokazie, zadają pytania, wyjaśnianiają wątpliwości.	Prezentuje, w jaki sposób połączyć Arduino ze Scratchem S4A. Prezentuje wymagany kod do współpracy z oprogramowaniem S4A, a następnie prezentuje możliwości samego środowiska (prezentacja multimedialna). Dokonuje zestawienia układu i jego uruchomienia. Omawia elementy składowe kodu do pomiaru temperatury. Wykonuje pokaz odczytu temperatury w środowisku S4A oraz pokaz wizualizacji odczytu temperatury z wykorzystaniem diody RGB.	• Pojęcia: czujnik, przetwornik A/D, dioda RGB, zmienne; • Filmy instruktażowe; • Prezentacja multimedialna; • Filmy dostępne w serwisie http://www.youtube.com/ hasła kluczowe: arduino RGB, arduino pomiar temperatury; • Tutoriale: http://s4a.cat/ http://e-swoi.pl/wiki/article/arduino-podstawy/ http://arduino.cc/en/Tutorial/AnalogInput
Biorą udział w dyskusji.	Prowadzi dyskusję dotyczącą sposobów i dokładności pomiarów temperatury.	• Dokumentacja techniczna http://ww1.microchip.com/downloads/en/DeviceDoc/21942e.pdf
Wykonują samodzielne zestawienie i oprogramowanie układu do pomiaru temperatury oraz do wizualizacji odczytu temperatury z wykorzystaniem diody RGB. Modyfikują wartości współczynników w celu zwiększenia dokładności odczytu temperatury przy różnych napięciach zasilania z USB.	Zachęca uczennice i uczniów do samodzielnego podłączenia układu i zaprogramowania. Sugeruje, jakie działania muszą podjąć uczennice i uczniowie, aby ich układ się uruchomił.	• Zaimplementowanie kodu do sterowania Arduino z poziomu Scratcha S4A; • Montaż układu i zaimplementowanie kodu do pomiaru temperatury; • Montaż układu i zaimplementowanie kodu do sterowania diodą RGB.
	Formułuje zadania: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	

Zadania rozszerzające:

» zmodyfikować kod i schemat połączeń pomiaru temperatury, zastępując diodę RGB jedną diodą LED

lub: » rozbudować kod o dodatkowe zmienne i obliczenia wyświetlające pomiar temperatury w skali Kelvina lub (oraz) w skali Fahrenheita.

Konspekt-scenariusz **Modułu A3.3**

Krzysztof Bytow

Temat: Odczytywanie stanu czujników

Nazwa implementacji: Układ pomiarowy Arduino

Opis implementacji: Wizualizacja działania dodatkowych elementów zestawu modułu-interfejsu z układem Arduino. Wykorzystanie funkcji przetwornika analogowo-cyfrowego do budowy układów pomiarowych. Istota funkcjonowania i zastosowania fotorezystora. Konstruowanie i oprogramowanie układów do odczytu stanu czujników na przykładzie interfejsów do pomiaru natężenia światła. Prezentacja odczytów na ekranie monitora oraz z wykorzystaniem diod elektroluminescencyjnych.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- rozpoznawanie środowiska do konstruowania i programowania układów mechatronicznych;
- wzbudzenie satysfakcji z tego, że działa zmontowany własnoręcznie układ elektroniczny;
- rozwijanie innowacyjności – koncypowanie, do czego można zastosować moduły-interfejsy;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- ma doświadczenie w pracy w środowisku do programowania wizualnego układów mechatronicznych;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- rozwija umiejętności tworzenia interfejsów mierzących i wyświetlających stan czujników.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- fotorezystor; 3 diody LED; 3 rezystory 220 Ω; rezystor 10 kΩ.

Metody, działania:

- zajawka inspirująca i dyskusja – pokaz odczytu natężenia światła;
- prezentacja multimedialna – film instruktażowy;
- metoda ćwiczebna – wizualizacja zmian odczytów fotorezystora w środowisku S4A;
- metoda ćwiczebna – skonstruowanie i oprogramowanie wskaźnika natężenia światła.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- zgodnie z zasadami działania podłącza czujnik pomiarowy: fotorezystor;
- prawidłowo buduje i oprogramowuje moduł-interfejs wskazujący poziomy oświetlenia;
- modyfikuje i rozbudowuje pomiarowe układy elektroniczne oraz kody źródłowe;
- trafnie używa sformułowań: *czujnik, czułość, wejście analogowe, przetwornik A/D.*

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Biorą udział w dyskusji.	Dyskutuje z uczniami nad sposobami pomiaru natężenia światła.	• Pojęcia: mikrokontroler, fotorezystor, opornik, dioda elektroluminescencyjna.
Współuczestniczą w prezentacji i pokazie, zadają pytań, wyjaśniają wątpliwości.	Prezentuje, w jaki sposób działa fotorezystor i przedstawia zależność jego rezystancji od światła – prezentacja multimedialna. Przeprowadza pokaz odczytu natężenia światła.	• Przedstawienie zależności rezystancji czujników od natężenia oświetlenia.
Próbują najpierw samodzielnie, a potem przy wsparciu nauczyciela stworzyć kod i uruchomić układ.	Zachęca uczniów do samodzielnego montażu wybranych układów na podstawie dostarczonych instrukcji. Nadzoruje działania, aby implementacje wykonywane było prawidłowo.	• Tutoriale: http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Fotorezystor http://pl.wikipedia.org/wiki/Opornik
	Zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	

Konspekt-scenariusz **Modułu A3.4**

Krzysztof Bytow

Temat: Interakcja modułu-interfejsu i środowiska S4A

Nazwa implementacji: Rozszerzenie możliwości programowania w środowisku Scratch.

Opis implementacji: Budowa interfejsu z wykorzystaniem zestawu Arduino i programu w środowisku Scratch (S4A) w celu stworzenia interaktywnej gry – Pong.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- rozpoznawanie środowiska do programowania wizualnego układów mechatronicznych;
- formowanie kreatywności i sprawności w montowaniu i rozbudowie modułów-interfejsów;
- rozwinięcie umiejętności sterowania elementami zestawu modułu-interfejsu;
- wzbudzenie satysfakcji z tego, że działa zmontowany własnoręcznie układ elektroniczny;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- ma doświadczenie pracy w środowisku do programowania wizualnego układów mechatronicznych;
- rozwija umiejętności konstruowania algorytmów;
- rozwija umiejętność współpracy z innymi uczennicami i uczniami oraz z nauczycielem.

Materiał nauczania-uczenia się:

- program S4A (Scratch); środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- 2 buttony; 2 rezystory 10 kΩ.

Metody, działania:

- zajawka inspirująca – pokaz sterowania kontrolerem zbudowanym z elementów modułu-interfejsu w grze osadzonej w środowisku S4A;
- prezentacja multimedialna – film instruktażowy do implementacji;
- metoda ćwiczebna – zestawienie i oprogramowanie układu sterującego grą.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- prawidłowo obsługuje środowisko programowania graficznego Scratch S4A;
- samodzielnie montuje i uruchamia przykładowe układy na podstawie schematów;
- deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
- prawidłowo łączy i steruje przyciskami z zestawu modułu-interfejsu;
- trafnie używa zwrotów: *button, wejście cyfrowe;*
- angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczenie w pokazie i prezentacji, zadawanie pytań, wyjaśnianie wątpliwości.	Prezentuje, w jaki sposób połączyć Arduino ze Scratchem S4A. Prezentuje wymagany kod do współpracy z oprogramowaniem S4A, a następnie prezentuje możliwości samego środowiska. Dokonuje zestawienia układu i jego uruchomienia. Omawia elementy składowe programu.	• Zapoznanie ze środowiskiem i funkcjami S4A; http://s4a.cat/ • Sterowanie kontrolerem zbudowanym z elementów modułu-interfejsu w grze osadzonej w środowisku S4A http://arduino.cc/en/Tutorial/Button
Uczniowie wykonują samodzielne zestawienie i oprogramowanie układu sterującego grą.	Zachęca uczennice i uczniów do samodzielnego podłączenia układu i zaprogramowania. Sugeruje, jakie działania muszą podjąć uczennice i uczniowie, aby ich układ uruchomił się.	• Zaimplementowanie układu i kodu do gry Pong; • Struktury języka – zmienne i stałe, pętle, symbole matematyczne.
	Formułuje zadania: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	

Zadania rozszerzające:

- Rozbudować kod o dodatkową piłkę (odbijamy 2 piłki).

Moduły B

Zajęcia dla szkół ponadgimnazjalnych

Środowisko i zagadnienia

B1

Linux, SRU, Scratch: szyfrowanie, wizualizacja

B1

Scratch: skrypty, wyrażenia, gry, sortowanie

B2

Lazarus, FreePascal: gry, automat, zegar

B2

Lazarus, FreePascal: gry, algorytmy, strategie

B3

Arduino IDE: terminal, przetwornik, pomiar, gra

B3

S4A, Arduino: interfejs, regulacja, sygnalizacja

Konspekt-scenariusz **Modułu B1.1**

Adam Jurkiewicz

Temat: SRU – Jak chronić ważne informacje

Nazwa implementacji: Szyfrowanie

Opis implementacji: Zaawansowane użytkowanie Linuksa, podstawowe informacje o bezpieczeństwie danych i szyfrowaniu.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie wiedzy o bezpieczeństwie danych i szyfrowaniu;
- rozwijanie kreatywności i innowacyjności;
- poznawanie różnych aplikacji i szyfrowania PGP;

b) szczegółowe: Uczennica/uczeń...
- posiada wiedzę z zakresu obsługi linii poleceń i szyfrowania;
- rozwinie umiejętność poruszania się w środowisku Linuksa;
- potrafi obsługiwać podstawowe aplikacje w Linuksie;
- posiada wiedzę z zakresu szyfrowania danych i używania podpisu elektronicznego.

Metody, działania:
- pogadanka i dyskusja – rozpisanie na kartce szyfru analogicznego jak ROT-13;
- prezentacja multimedialna – zapoznanie z oprogramowaniem;
- metoda ćwiczebna – tworzenie własnych kluczy PGP, podpisywanie i szyfrowanie wiadomości.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...
- wskaże wagę bezpieczeństwa danych oraz zalety podpisu elektronicznego i szyfrowania;
- omówi sposoby zdobywania informacji z manuali systemowych;
- omówi obsługę konsoli;
- wykorzysta podstawowe funkcje obsługi Linuksa, aplikacji gpg, Kgpg;
- utworzy własne klucze PGP, nauczy się obsługi gpg z linii poleceń.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance.	Wprowadza do tematu: zachęca uczennice/uczniów do zastanowienia się, czy mają jakieś ważne informacje, które chcieliby szyfrować (np. hasła); wskazuje, do czego możemy wykorzystywać podpis elektroniczny (ważne pisma, ważne dane). Omówia zalety podpisu elektronicznego i szyfrowania – pogadanka.	Pojęcia: szyfrowanie, bezpieczeństwo danych, podpis elektroniczny.
Wspólnie z nauczycielem rozpisują szyfr analogiczny do ROT-13 i w ten sposób zapoznają się ze schematem listy danych i wskaźników (do elementów listy).	Omawia metody ROT-13 dla prostego szyfrowania; rozpisuje na kartce szyfr analogiczny jak ROT-13.	ROT-13 http://pl.wikipedia.org/wiki/ROT13
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Prezentuje funkcje oprogramowania gpg (man systemowy – prezentacja multimedialna).	Opisy technologii PGP: http://pl.wikipedia.org/wiki/GPG oraz http://pl.wikipedia.org/wiki/Pretty_Good_Privacy oraz podręcznik systemowy gpg (man gpg).
Tworzą własne klucze PGP, eksportują je na serwer kluczy w internecie, pobierają klucze kolegów/koleżanek, kopiują sobie nawzajem pliki.	**Formułuje zadania:** • Utwórz własną parę kluczy PGP i wyeksportuj je na serwer kluczy; • Odnajdź na serwerze kluczy i zaimportuj klucz (adam.jurkiewicz@fwioo.pl) oraz klucze kolegów i koleżanek z zajęć; • Stwórz tekst, zaszyfruj go i przekaż do odszyfrowania. Wspiera uczniów podczas tworzenia własnych kluczy PGP, podpisywania i szyfrowania wiadomości.	

Konspekt-scenariusz **Modułu B1.2**

Jarosław Żok

Temat: Wizualizacja instrukcji warunkowej

Nazwa implementacji: Wybór drogi

Opis implementacji: Wizualizacja sposobu podejmowania decyzji przez program wykonujący instrukcję warunkową.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wizualizacja podstawowych mechanizmów programistycznych z wykorzystaniem narzędzi informatycznych;
- wzbudzenie zainteresowania maszynowym rozwiązywaniem problemów;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu podstawowych pojęć algorytmiki;
- wykorzystuje wizualną reprezentację abstrakcyjnego problemu dla rozwijania jego rozumienia;
- wykorzystuje instrukcje warunkowe i pętle w praktyce poprzez implementację wizualizacji;
- wykorzystuje podprogramy do realizacji powtarzających się czynności.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- omówi pojęcia: pętla, instrukcja warunkowa, zmienna, algorytm;
- wykorzysta Scratcha do stworzenia wizualizacji działania instrukcji warunkowej.

Czynności uczniów	Działania nauczyciela	Materiał
	Wyjaśnia uczniom, czym jest instrukcja warunkowa.	Pojęcia warunku i sterowania.
Refleksja uczniów, propozycje rozwiązań.	Zachęca uczniów do zastanowienia się, w jaki sposób można przedstawić rozwidlenie ścieżki programu na przykładzie życiowego problemu wyboru i podjęcia decyzji (rozwidlenie ścieżek w lesie, decyzja kupna tańszego lub droższego towaru za pomocą dostępnych ograniczonych środków itp.).	Przygotowane wcześniej (zamieszczone w katalogu z implementacjami) duszki z wizerunkiem mrówki-robotnicy, mrówki-strażnika, rozwidlenia w mrowisku, dwóch jagód niesionych przez mrówkę; opcjonalnie przygotowanie duszków przez uczniów.
Tworzą interaktywną animację wizualizującą działanie instrukcji warunkowej.	Wyjaśnia pojęcie instrukcji jako elementu programistycznego, sterującego wykonywaniem kodu. **Formułuje zadanie:** • Stworzenie interaktywnej animacji wizualizującej działanie instrukcji warunkowej w programowaniu.	• Pojęcie zmiennych; • Programowanie strukturalne; • Programowanie zdarzeniowe; • Instrukcje języka Scratch.

Konspekt-scenariusz **Modułu B1.3**

Jarosław Żok

Temat: Wizualizacja pętli

Nazwa implementacji: Zakręcona mrówka

Opis implementacji: Wizualizacja sposobu podejmowania decyzji przez program wykonujący instrukcję warunkową.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wizualizacja podstawowych mechanizmów programistycznych z wykorzystaniem narzędzi informatycznych;
- kształtowanie wiedzy z zakresu podstawowych pojęć algorytmiki;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu podstawowych pojęć algorytmiki;
- wykorzystuje wizualną reprezentację abstrakcyjnego problemu dla rozwijania jego rozumienia;
- wykorzystuje instrukcje warunkowe i pętle w praktyce poprzez implementację wizualizacji;
- wykorzystuje podprogramy do realizacji powtarzających się czynności.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- omówi pojęcia: pętla, instrukcja warunkowa, zmienna, algorytm;
- wykorzysta Scratcha do stworzenia wizualizacji działania pętli.

Czynności uczniów	Działania nauczyciela	Materiał
	Wyjaśnia uczniom, czym jest pętla.	• Pojęcie pętli.
Refleksja uczniów, propozycje rozwiązań.	Zachęca uczniów do zastanowienia się, w jaki sposób można wykonać ten sam kod N razy.	
Tworzą interaktywną animację wizualizującą działanie pętli w programowaniu.	Wyjaśnia pojęcie zmiennej jako elementu programistycznego przechowującego wartości. **Formułuje zadanie:** • Stworzenie interaktywnej animacji wizualizującej działanie pętli w programowaniu.	• Pojęcie zmiennych i iteracji; • Programowanie strukturalne; • Programowanie zdarzeniowe; • Instrukcje języka Scratch.

Konspekt-scenariusz **Modułu B1.4**

Jarosław Żok

Temat: Wizualizacja operatorów przypisania i ewaluacji

Nazwa implementacji: Magiczny operator

Opis implementacji: Wizualizacja metody ewaluacji prawej strony wyrażenia.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- kształtowanie umiejętności prezentacji abstrakcyjnych pojęć za pomocą wizualnych metod z wykorzystaniem elementów programowania;
- wzbudzenie zainteresowania maszynowym rozwiązywaniem problemów;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu funkcjonowania współczesnych maszyn;
- wykorzystuje instrukcje warunkowe i pętle w praktyce poprzez implementację wizualizacji;
- wykorzystuje podprogramy do realizacji powtarzających się czynności.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- omówi pojęcia: zmienna, zmienna pusta, deklaracja zmiennej, przypisanie, wyrażenie, ewaluacja;
- wskaże, w jaki sposób maszyny wyliczają wartości wyrażeń;
- wykorzysta Scratch do stworzenia wizualizacji przypisania wartości.

Czynności uczniów	Działania nauczyciela	Materiał
Uczestniczą w pogadance, zadają pytania, wyjaśniają wątpliwości.	Wyjaśnia uczniom, czym jest zmienna, czym się różni zmienna zadeklarowana niezainicjowana od zmiennej zadeklarowanej zainicjowanej, czym jest wyrażenie i w jaki sposób jest ono wyliczane.	• Pojęcie zmiennych; • Pojęcie wyrażenia.
Próbują wyjaśnić czym może być zmienna.	Opcjonalnie: wprowadza pojęcie zmiennej tymczasowej potrzebnej do przechowywania cząstkowych wyników ewaluacji wyrażenia.	• Pojęcie kontenera przechowującego wartości wyrażeń.
Tworzą interaktywną animację wizualizującą działanie operatora przypisania.	Wyjaśnia pojęcie zmiennej jako elementu programistycznego przechowującego wartości. **Formułuje zadanie:** • Stworzenie interaktywnej animacji wizualizującej działanie operatora przypisania.	• Pojęcie przypisania wartości; • Programowanie strukturalne; • Programowanie zdarzeniowe; • Instrukcje języka Scratch.

Konspekt-scenariusz **Modułu B1.5**

Jarosław Żok

Temat: Ewaluacja prawej strony wyrażenia

Nazwa implementacji: Zamieszanie z kolorami

Opis implementacji: Wizualizacja metody ewaluacji prawej strony wyrażenia.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólny (zadanie/przesłanie nauczyciela dla całych zajęć):

- nauczenie prezentacji abstrakcyjnych pojęć za pomocą wizualnych metod z wykorzystaniem elementów programowania;

b) szczegółowe: Uczennica/uczeń...

- posiada wiedzę z zakresu podstawowych pojęć matematyki;
- posiada wiedzę z zakresu funkcjonowania współczesnych maszyn;
- wykorzystuje instrukcje warunkowe, pętle i wyrażenia matematyczne w praktyce poprzez implementację wizualizacji;
- wykorzystuje podprogramy do realizacji powtarzających się czynności.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- omówi pojęcia: zmienna, zmienna pusta, deklaracja zmiennej, przypisanie, wyrażenie, ewaluacja;
- wskaże, w jaki sposób maszyny wyliczają wartości wyrażeń;
- przypisze wartości do zmiennych;
- wykorzysta Scratch do stworzenia wizualizacji przypisania wartości.

Czynności uczniów	Działania nauczyciela	Materiał
Uczeń przygotowuje dwie kartki: jedną pustą, drugą zapisaną. Próbuje określić różnicę między nimi i znaleźć analogię między kartkami a zmiennymi.	Wyjaśnia uczniom, czym jest zmienna; wyjaśnia, czym się różni zmienna zadeklarowana niezainicjowana od zmiennej zadeklarowanej zainicjowanej.	• Pojęcie zmiennych.
Uczeń podaje przykłady wyrażeń.	Wyjaśnia uczniom, czym jest wyrażenie i w jaki sposób jest ono wyliczane.	• Pojęcie wyrażenia matematycznego.
Uczeń wyobraża sobie jak podzielić wyrażenie operatorami i zapisuje wynik ich działania na osobnych kartkach.	Opcjonalnie: wprowadza pojęcie zmiennej tymczasowej potrzebnej do przechowywania cząstkowych wyników ewaluacji wyrażenia.	• Pojęcie operatora i częściowych wyników.
Tworzą interaktywną animację wizualizującą działanie operatora przypisania.	Wyjaśnia pojęcie zmiennej jako elementu programistycznego przechowującego wartości. **Formułuje zadanie:** • Stworzenie interaktywnej animacji wizualizującej działanie operatora przypisania.	• Pojęcie pętli; • Programowanie strukturalne; • Programowanie zdarzeniowe; • Instrukcje języka Scratch.

Konspekt-scenariusz **Modułu B1.6**

Jarosław Żok

Temat: Rozbudowanie gry o strategię blokowania

Nazwa implementacji: „Kółko i krzyżyk”

Opis implementacji: Odnalezienie w istniejącym kodzie miejsca, które należy rozbudować o sprawdzanie istnienia kombinacji figur w grze „Kółko i krzyżyk”, które w następnym ruchu pozwolą przeciwnikowi na wygranie i zablokowanie ruchu przeciwnika.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie motywacji poprzez zaangażowanie w rozbudowę gry komputerowej;

b) szczegółowe: Uczennica/uczeń…
- kształtuje umiejętność opisywania algorytmów za pomocą języka naturalnego;
- dokonuje analizy istniejącego kodu w celu określenia miejsc realizujących poszczególne części algorytmu;
- kształtuje umiejętność analizowania algorytmu i proponowania rozwiązań rozszerzających działanie algorytmu;
- implementuje rozwiązania problemów za pomocą języków programowania;
- odczuwa satysfakcję z wykonania rozbudowy działającej implementacji gry.

Materiał nauczania-uczenia się:
- programowanie strukturalne;
- programowanie zdarzeniowe;
- instrukcje języka Scratch.

Metody, działania:
- prezentacja z instrukcją nauczyciela;
- metoda problemowa;
- programowanie strukturalne i zdarzeniowe;
- gra z komputerem – sprawdzanie poprawności implementacji.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- posługuje się naturalnym językiem przy opisie algorytmów;
- potrafi wskazać miejsca w algorytmie realizujące poszczególne jego części;
- poprzez dokonanie analizy algorytmu proponuje rozwiązanie rozszerzające jego działanie o założone funkcje;
- prezentuje dane w macierzy dwuwymiarowej za pomocą liniowej adresacji;
- wykorzystuje tablice do przechowywania danych tymczasowych;
- dzieli algorytm na podprogramy;
- realizuje grę z interfejsem graficznym oraz algorytmem komputerowego przeciwnika.

Czynności uczniów	Działania nauczyciela	Materiał
	Pokazuje jak można zaprezentować w pamięci komputera planszę do gry w „Kółko i krzyżyk"; wyjaśnia uczniom, jak działa automatyczny algorytm gry w „Kółko i krzyżyk".	• Programowanie strukturalne; • Programowanie zdarzeniowe; • Instrukcje języka Scratch; • Pętle; • Instrukcje warunkowe; • Tablice jedno i dwuwymiarowe.
	Pokazuje, jak algorytm odnajduje dwójki własnych figur spośród trójek gwarantujących wygraną.	
Proponują sposoby blokowania ruchów przeciwnika na podstawie istniejącego algorytmu; odnajdują miejsce, gdzie implementację algorytmu należy zmodyfikować.	Pyta uczniów, jakie mogą być według nich sposoby na blokowanie ruchów przeciwnika.	
Proponują sposób implementacji, implementują zmianę i sprawdzają poprawność jej działania.	Wspiera uczniów, koryguje błędy, naprowadza.	
Sprawdzają, czy algorytm gra lepiej niż za pierwszym uruchomieniem, zanim dokonano modyfikacji.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	

Konspekt-scenariusz **Modułu B1.7**

Jarosław Żok

Temat: Rozbudowa gry Pong o nowe elementy.

Nazwa implementacji: Gra Pong

Opis implementacji: Zmiany w skryptach gry Pong, dodające kolejny element zmieniający jej zasady.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie motywacji poprzez zaangażowanie w rozbudowę gry komputerowej;

b) szczegółowe: Uczennica/uczeń…
- kształtuje umiejętność opisywania algorytmów za pomocą języka naturalnego;
- dokonuje analizy istniejącego kodu w celu określenia miejsc realizujących poszczególne części algorytmu;
- posiada wiedzę z zakresu podstawowych pojęć programowania;
- kształtuje umiejętność prezentowania danych w strukturach danych;
- wykorzystuje elementy programowania zdarzeniowego i obiektowego;
- kształtuje umiejętność dzielenia algorytmu na podprogramy i wykorzystania języka programowania do rozwiązywania problemów w implementacji.

Materiał nauczania-uczenia się:
- programowanie strukturalne;
- programowanie zdarzeniowe;
- instrukcje języka Scratch.

Metody, działania:
- prezentacja z instrukcją nauczyciela;
- metoda problemowa;
- programowanie strukturalne i zdarzeniowe;
- gra z komputerem – sprawdzanie poprawności implementacji.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- posługuje się naturalnym językiem przy opisie algorytmów;
- potrafi wskazać miejsca w algorytmie realizujące poszczególne jego części;
- omawia podstawowe pojęcia programowania: zmienna globalna, zmienna lokalna, lista, kolejka;
- prezentuje dane w strukturach danych typu lista i kolejka danych;
- programuje obiektowo z wykorzystaniem duszków oraz zmiennych lokalnych duszka;
- wskazuje podprogramy w algorytmie;
- prawidłowo implementuje zmianę w grze.

Czynności uczniów	Działania nauczyciela	Materiał
Oglądają prezentację gry, poznają zasadę gry.	Przekazuje zasady gry Pong, elementy gry oraz ich interakcje na planszy; wyjaśnia, jak zaimplementowane są zachowania poszczególnych elementów gry.	• Programowanie strukturalne; • Programowanie zdarzeniowe; • Instrukcje języka Scratch; • Pętle; • Instrukcje warunkowe; • Tablice jedno i dwuwymiarowe; • Kolejki LIFO; • Mapy bitowe reprezentujące położenie elementów planszy gry.
Proponują sposób realizacji nowej funkcji gry.	Przekazuje wymaganą nową funkcjonalność gry.	
Odnajdują miejsce, gdzie implementację algorytmu należy zmodyfikować.	Pyta uczniów, jakie miejsca w skryptach gry są odpowiedzialne za prawidłową implementację zmian.	
Proponują sposób implementacji, implementują zmianę i sprawdzają poprawność jej działania.	Wspiera uczniów, koryguje błędy, naprowadza.	
Sprawdzają, czy gra działa prawidłowo oraz, czy realizuje zaimplementowaną funkcję.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	

Konspekt-scenariusz **Modułu B1.8**

Agata Wawruch

Temat: Wybrane algorytmy sortowania

Nazwa implementacji: Animacja – sortowanie

Opis implementacji: Wizualizacja sposobu sortowania elementów zbioru w porządku rosnącym, na przykładzie zbioru liczbowego o zmiennej liczbie elementów. Związek liczby kroków algorytmu z liczbą elementów w zbiorze.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólny (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie umiejętności sortowania elementów zbioru w algorytmice;

b) szczegółowe: Uczennica/uczeń...
- posiada wiedzę z zakresu podstawowych pojęć algorytmicznych;
- potrafi rozwiązywać problemy w postaci algorytmicznej;
- rozwinie kreatywność, umiejętność planowania.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...
- omówi podstawowe pojęcia algorytmiczne: algorytm, sortowanie;
- scharakteryzuje sposób działania algorytmów „bubble sort", „inserting sort" oraz „selection sort" oraz uporządkuje 5-elementowy zbiór tymi metodami;
- narysuje schemat blokowy algorytmu realizującego sortowanie „bąbelkowe";
- przeanalizuje ilość kroków algorytmu w zależności od liczby elementów zbioru;
- zaimplementuje algorytmy w języku Scratch.

Czynności uczniów	Działania nauczyciela	Materiał
Dyskutują nad potrzebą porządkowania elementów w życiu codziennym.	Zachęca do zastanowienia się, w jakich sytuacjach istnieje konieczność uporządkowania elementów – burza mózgów.	• Pojęcia: algorytm, sortowanie.
Poszukują najefektywniejszej metody porządkowania elementów w zbiorze.	Prosi o uporządkowanie 5 kartek z różnymi liczbami, ułożonych w przypadkowej kolejności w porządku rosnącym. Formułuje polecenia: 1. Mając do dyspozycji dowolną liczbę ruchów, w każdym można zamienić miejscami tylko dwie kartki. 2. Wykonać ćwiczenia w możliwie najmniejszej liczbie ruchów. 3. Ułożyć karty, zamieniając w każdym ruchu tylko dwie sąsiednie.	• 5 karteczek z różnymi liczbami z zakresu 1..100;
Próbują uporządkować strategię układania kartek.	Zachęca uczniów do samodzielnego rozwiązania problemu, stosując wskazówki.	
Uczestniczą w pogadance.	Prowadzi pogadankę dotyczącą sortowania danych jako jednego z podstawowych problemów informatyki; prezentuje ciekawą wizualizację problemu.	• Źródło: http://www.youtube.com/watch?v=lyZQPjUT5B4
Projektują schemat blokowy sortowania metodą bąbelkową.	Zachęca do poszukiwania reguł, obserwowania prawidłowości.	• Arkusz papieru do pracy nad projektem algorytmu
Testują implementacje napisane w języku Scratch algorytmów porządkujących zbiory pięcioelementowe; implementują wersje dla zbioru złożonego z dowolnej liczby losowo wybranych elementów.	Objaśnia instrukcje języka Scratch w odniesieniu do elementów schematu blokowego.	• Instrukcje języka Scratch (warunkowe, pętle, operacje wejścia/wyjścia, sterowanie czasem); • Aplikacja Scratch 1.4
	Formułuje zadania: • **Obligatoryjne:** zmodyfikuj kod każdej implementacji, zmieniając liczbę elementów zbioru na 7, 10, 15; przetestuj działanie aplikacji; • **Rozszerzające:** rozbuduj kod każdej implementacji o możliwość wyboru przez użytkownika liczby elementów zbioru.	

Konspekt-scenariusz **Modułu B2.1**

Piotr Fiorek

Temat: Wprowadzenie do środowiska Lazarus

Nazwa implementacji: „Kółko i Krzyżyk”

Opis implementacji: Implementacja prezentuje środowisko Lazarus oraz prosty program w języku FreePascal.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie motywacji poprzez zaangażowanie w tworzenie prostej gry komputerowej;

b) szczegółowe: Uczennica/uczeń…
- poznaje zasadę programowania jako sztuki wydawania poleceń komputerowi oraz sposób na prostsze wykonywanie skomplikowanych lub często powtarzalnych zadań;
- poznaje środowisko Lazarus jako jedno z narzędzi programowania graficznego;
- kształtuje umiejętność obsługi środowiska Lazarus;
- kształtuje umiejętność stosowania podstawowych konstrukcji języka FreePascal;
- odczuwa satysfakcję z wykonania działającej implementacji gry;
- kształtuje zainteresowanie nauką programowania.

[opcjonalnie] Uczeń zaawansowany…
- rozumie kod w stopniu pozwalającym na wprowadzenie prostych modyfikacji.

Materiał nauczania-uczenia się:
- komputer;
- środowisko Lazarus;
- FreePascal.

Metody, działania:
- prezentacja z instrukcją nauczyciela;
- edycja kodu programu;
- gra z komputerem – sprawdzanie poprawności implementacji.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...
- potrafi wyjaśnić zasadę programowania przy wykorzystaniu komputera;
- potrafi określić, do czego służą poszczególne składowe środowiska Lazarus: Inspektor obiektów, Właściwości, Zdarzenia, Favorites, Komponenty, Komunikaty, Edytor źródeł - Kod programu, Menu Edytora, Formularz;
- potrafi poruszać się w środowisku Lazarus;
- rozumie podstawowe konstrukcje języka FreePascal;
- rozumie i potrafi wskazać korzyści płynące z umiejętności programowania komputerów.

[opcjonalnie] Uczeń zaawansowany...
- samodzielnie wprowadza proste modyfikacje w kodzie.

Czynności uczniów	Działania nauczyciela	Materiał
Zadają pytania, wyjaśniają wątpliwości.	Prezentuje środowiska Lazarus.	• Implementacja: składowe środowiska Lazarus: Inspektor obiektów, Właściwości, Zdarzenia, Favorites, Komponenty, Komunikaty, Edytor źródeł - Kod programu, Menu Edytora, Formularz; • Ukazanie zasady programowania; • implementacja: kod programu.
	Prezentuje programy komputerowe wykonujące czasochłonne i męczące zadania.	
Piszą z pomocą nauczyciela kod prostej gry w „Kółko i Krzyżyk".	Wspiera uczniów, koryguje błędy, naprowadza.	
Sprawdzają poprawność implementacji.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	

Konspekt-scenariusz **Modułu B2.2**

Piotr Fiorek

Temat: Gra w zapalane światełka

Nazwa implementacji: Gra w światełka

Opis implementacji: Poznanie zasad prostej gry w światełka.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie do interakcji przy implementowaniu prostej gry komputerowej;

b) szczegółowe: Uczennica/uczeń…

- rozumie zasady działania programu;
- kształtuje umiejętności pisania prostej gry;
- kształtuje umiejętność tworzenia programów, w których następny krok zależy od bieżącego stanu gry;
- odczuwa satysfakcję z wykonania działającej implementacji gry opartej na zmodyfikowanych zasadach.

Materiał nauczania-uczenia się:

- środowisko Lazarus;
- opis implementacji.

Metody, działania:

- zmiana zasad funkcjonowania gry;
- powiększenie planszy gry;
- uzupełnienie fragmentu kodu usuniętego przez nauczyciela.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- potrafi wytłumaczyć zasadę działania programu;
- potrafi programować w stopniu pozwalającym na zaimplementowanie większej planszy gry;
- potrafi programować w stopniu pozwalającym na uzupełnienie brakujących fragmentów kodu;
- potrafi tworzyć programy, w których następny krok zależy od bieżącego stanu gry.

Czynności uczniów	Działania nauczyciela	Materiał
Wymyślają, jak zmodyfikować dotychczasowe zasady gry.	Pomaga przy wymyślaniu spójnych zasad gry.	• Środowisko programowania Lazarus; • Opis implementacji.
Implementują większą planszę.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	
Uzupełniają brakujące fragmenty kodu.	Pomaga w zrozumieniu brakującej funkcjonalności, wspiera uczniów, koryguje błędy, naprowadza, motywuje.	

Konspekt-scenariusz **Modułu B2.3**

Piotr Fiorek

Temat: Automaty

Nazwa implementacji: „Gra w życie”

Opis implementacji: Implementacja przedstawia „Grę w życie” jako przykład prostej symulacji opartej na automatach.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie do interakcji przy implementowaniu prostej gry komputerowej;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu podstawowych zastosowań programistycznych – czym są automaty, do czego służą, jakie rządzą nimi zasady;
- kształtuje umiejętność konstruowania automatów;
- odczuwa satysfakcję z wykonania działającej implementacji gry opartej na samodzielnie określonych zasadach.

Materiał nauczania-uczenia się:
- środowisko programowania Lazarus;
- opis implementacji.

Metody, działania:
- zmiana zasad gry na wymyślone przez siebie i obserwacja, jaki wpływ mają zmiany na zachowanie automatu;
- implementacja gry.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...
- omawia, czym są automaty i do czego służą;
- potrafi określić zasady działania automatów;
- potrafi skonstruować prosty automat.

Czynności uczniów	Działania nauczyciela	Materiał
Wymyślają własne zasady gry.	Pomaga przy wymyślaniu zasad.	• Środowisko programowania Lazarus; • Opis implementacji.
Implementują je w istniejącym programie.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	
Obserwują zachowania programu po zmianach.	Pomaga w zrozumieniu zachowania programu.	

Konspekt-scenariusz **Modułu B2.4**

Piotr Fiorek

Temat: Zegar

Nazwa implementacji: Zegar binarny

Opis implementacji: Implementacja przedstawia zasadę działania zegara decybinarnego.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie do interakcji przy implementowaniu prostej gry komputerowej;

b) szczegółowe: Uczennica/uczeń...
- posiada wiedzę dotyczącą zasad działania zegara opartego na systemie binarnym;
- kształtuje umiejętność odczytywania cyfr zapisywanych w systemie binarnym;
- kształtuje umiejętność wykonywania działań w systemie binarnym;
- rozumie zasadę funkcjonowania zegara binarnego;
- rozumie działanie operatora logicznego „and" i „or" na zmiennych binarnych.

[opcjonalnie] Uczeń zaawansowany...
- rozumie zasadę działania systemu szesnastkowego.

Materiał nauczania-uczenia się:
- tablica lub kartka papieru i coś do pisania;
- środowisko Lazarus;
- opis implementacji.

Metody, działania:
- konwersja na kartce lub na tablicy liczb z systemu dziesiętnego na dwójkowy i odwrotnie;
- wykonywanie operacji logicznych „and" i „or" na przykładowych liczbach zapisanych w systemie binarnym;
- uzupełnienie fragmentów kodu usuniętych przez nauczyciela przed zajęciami.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- potrafi omówić zasady działania zegara opartego na systemie binarnym;
- potrafi odczytywać oraz zapisywać cyfry przy użyciu systemu binarnego;
- potrafi przekształcać cyfry z systemu dziesiętnego na dwójkowy;
- potrafi wyjaśnić zasadę funkcjonowania zegara binarnego;
- potrafi omówić działanie operatora logicznego „and" i „or" na zmiennych binarnych.

[opcjonalnie] Uczeń zaawansowany...

- potrafi omówić zasadę działania systemu szesnastkowego.

Czynności uczniów	Działania nauczyciela	Materiał
Przekształcają liczby dziesiętne na zapis dwójkowy.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	• Tablica lub kartka papieru i coś do pisania; • Środowisko Lazarus; • Opis implementacji.
Przekształcają liczby dwójkowe na dziesiętne.		
Zapisują wyniki działań operatorów logicznych.		

Konspekt-scenariusz **Modułu B2.5**

Stanisław Ubermanowicz

Temat: Losowanie i porównywanie

Nazwa implementacji: Gra „Papier-kamień-nożyce”

Opis implementacji: Realizacja gry losowej, polegającej na równoczesnym wyborze dwóch przedmiotów, z których jeden wygrywa według zasady: papier owija kamień, kamień tępi nożyce, a nożyce tną papier. Gra powinna mieć 3 przyciski z miniaturami przedmiotów. Kliknięcie oznacza wybór obiektu przez gracza i powoduje wylosowanie obiektu dla komputera jako przeciwnika. Pojawiają się dwa duże obrazki przedmiotów i zależnie od układu liczone są punkty za wygraną lub remis.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- wdrażanie do pracy w środowisku programowania wizualno-obiektowo-zdarzeniowego;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie do interakcji przy implementowaniu od podstaw prostej gry komputerowej;

b) szczegółowe: Uczennica/uczeń…

- rozpoznaje zintegrowane środowisko programowania wspieranego interfejsem graficznym;
- zna zasadę tworzenia na ekranie obiektów za pomocą GUI i modyfikowania ich właściwości;
- zna główne bloki struktury kodu źródłowego oraz elementarne instrukcje języka FreePascal;
- ma przyswojone i rozumie pojęcia: obiekty (widżety) i ich atrybuty, kod źródłowy, instrukcje;
- odczuwa satysfakcję z tego, że wykonał implementację i poznał specyfikę gry losowej.

[opcjonalnie] Uczeń zaawansowany…

- umie modyfikować wygląd i rozbudowywać funkcjonalność wytworzonej implementacji.

Materiał nauczania-uczenia się:

- zintegrowane środowisko programowania: okna edycji, inspekcji obiektów i kompilacji;
- tworzenie obiektów ekranowych gry: przycisków, obrazków i pól tekstowych na wyniki;

- programowanie procedur obsługi zdarzeń inicjowanych przez kliknięcia gracza;
- struktury kodu źródłowego – deklaracje, definicje, nazwy, zmienne, instrukcje;
- procedury – losowanie, porównywanie, zliczanie, składowanie i transfer obrazków.

UWAGA: *Zakres omawiania struktur kodu dobiera trener adekwatnie do możliwości percepcyjnych uczniów.*

Metody, działania:
- zajawka inspirująca – krótki pokaz przykładowej, estetycznie wykonanej gry P-K-N;
- gra w parach uczniów, za pomocą układów dłoni symbolizujących papier, kamień i nożyce;
- metoda problemowa – próba określenia, czy istnieje strategia zapewniająca wygraną;
- metoda projektu – tworzenie na ekranie niezbędnych obiektów, wypełnionych obrazkami;
- programowanie – procedury obsługi kliknięć, ukazywania obrazków i zliczania punktów;
- gra z komputerem – sprawdzanie poprawności implementacji; próba różnych strategii.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...
- obsługuje zintegrowane środowisko programowania wspieranego interfejsem graficznym;
- tworzy potrzebne obiekty za pomocą GUI i odpowiednio modyfikuje ich właściwości;
- nazywa główne bloki struktury kodu źródłowego i elementarne instrukcje języka FreePascal;
- trafnie operacjonalizuje i objaśnia pojęcia: obiekty, atrybuty, kod źródłowy, instrukcje;
- prawidłowo wprowadza kod źródłowy i doprowadza do pełnego działania implementacji;
- próbuje wygrać z implementacją sztucznej inteligencji – nielosowej strategii predykcyjnej.

[opcjonalnie] Uczeń zaawansowany...
- prawidłowo modyfikuje wygląd i rozbudowuje funkcjonalności wytworzonej implementacji.

Czynności uczniów	Działania nauczyciela	Materiał
Oglądają krótką prezentację gry. Poznają zasadę gry.	Pokazuje grę i zachęca uczniów do przyjrzenia się zasadzie działania przykładowej implementacji.	Gra wyświetlana z projektora lub on-line: www.supergry24.pl/zagraj,w,gre,196,2.html
Grają parami z użyciem dłoni do wizualizacji wyboru. Analizują, czy istnieje strategia wygranej.	Zwraca uwagę na brak strategii zapewniającej wygraną w grze czysto losowej.	Pięść zaciśnięta jako *kamień*; palec wskazujący i środkowy jako *nożyce*; dłoń otwarta jako *papier*.

Rozpoznają elementy interfejsu zintegrowanego środowiska Lazarus & FreePascal.	Objaśnia istotę i funkcjonalności środowiska oraz sposób edycji i przełączania między oknami.	Środowisko programowania wizualno-obiektowo-zdarzeniowego: okna edycji, inspekcji i kompilacji.
Ćwiczą sposób umieszczania na *Oknie* obiektów-kontenerów obrazkowych i tekstowych.	Wspiera uczniów w tworzeniu na ekranie niezbędnych obiektów. Wskazuje źródła obrazków.	Pasek narzędziowy Lazarusa z *Paletą komponentów* – widżetów: TImage, TImageList i TLabel.
Ćwiczą nadawanie obiektom odpowiednich atrybutów: nazw, wielkości, położenia na ekranie.	Objaśnia sposób określania właściwości. Uzgadnia z uczniami nazwy i optymalne rozmiary.	Okno Inspektora obiektów Lazarusa z zakładką *Właściwości.*
Deklarują procedury i sprzęgają je z obiektami klikanymi. Definiują typy zmiennych.	Omawia struktury kodu źródłowego. Uzgadnia z uczniami nazwy procedur i zmiennych.	Okno *Edytora źródeł* w Lazarusie oraz Inspektor obiektów z zakładką *Zdarzenia*. Język FreePascal.
Piszą instrukcje realizujące losowanie i porównywanie liczb, wyświetlanie obrazków i napisów.	Objaśnia pojęcia *randomizacji* oraz *procedury obsługującej kliknięcie*. Omawia funkcję GetBitmap(…).	Okno *Edytora źródeł* w Lazarusie. Język FreePascal. Okno dialogowe *Edytor ImageList.*
Uruchamiają implementację i sprawdzają jej poprawność.	Wyjaśnia potrzebę kompilacji kodu i wspiera w korygowaniu błędów.	Oferty poleceń w menu *Uruchom* i Okno *Komunikaty* w Lazarusie.
Grają z komputerem.	Omawia czysto losowy wynik gry.	Własna implementacja gry P-K-N.
Umieszczają swój projekt gry w e-Repozytorium. Opisują swe dokonania w e-Portfolio.	Formułuje i sprawdza zadania obligatoryjne dokumentowania wytworów i osiągnięć.	Internet, przeglądarka. Funkcje Serwisu e-Swoi
Poznają grę *online*, zawierającą elementy sztucznego intelektu.	Zwraca uwagę na implementację „uczącą się od ludzi grających”.	Gra z prognozowaniem ruchów: www.nytimes.com/interactive/science/rock-paper-scissors.html
Poznają inne wersje gry w sieci.	W razie trudności wspiera poprzez pytania naprowadzające.	Zaawansowane odmiany online: www.gry.jeja.pl/3813,papier-kamien-nozyce-25.html

Zadania rozszerzające:

- Zmodyfikuj implementację tak, aby po wylosowaniu obiektów wyświetlane były obrazki ilustrujące efekt wygranej, tj.: owijanie kamienia, tępienie nożyc, przecinanie papieru.

lub:

- Zaprojektuj i wykonaj bardziej złożoną grę z większą liczbą obiektów (zob. np. w Wikipedii zasady gry: „Papier-kamień-nożyce-jaszczurka-Spock").

Uwaga: Zadania rozszerzające przeznaczone są do realizacji poza zajęciami, lecz można podjąć je z grupą zaawansowaną.

Konspekt-scenariusz **Modułu B2.6**

Stanisław Ubermanowicz

Temat: Rozrzucanie i porządkowanie

Nazwa implementacji: Układanka alfabetyczna

Opis implementacji: Realizacja gry logicznej typu puzzle, polegającej na porządkowaniu 24 liter alfabetu (A÷X) ułożonych w tablicy 5x5, poprzez przemieszczanie liter z pól stycznych do pola pustego. Gra powinna mieć Menu z trzema poziomami „Rozrzucenia" (łatwe, średnie, trudne) i z możliwością automatycznego „Uporządkowania" wszystkich liter jednym kliknięciem. Założeniem jest to, że pola z literami nie mają być przesuwane na ekranie, a efekt animacji uzyskuje się przez samą zmianę wyświetlanych liter na nieruchomych polach tekstowych.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- wdrażanie do pracy w środowisku programowania wizualno-obiektowo-zdarzeniowego;
- wzbudzenie zainteresowania poznawaniem podstaw tworzenia prostej gry komputerowej;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie do poszukiwania strategii wygranej, prowadzącej do rozwiązania układanki;

b) szczegółowe: Uczennica/uczeń...

- ma przyswojone i rozumie pojęcia: kody liter, tablica, menu, obsługa zdarzeń;
- zna zasadę kreowania kolekcji obiektów TPanel i nadawania im atrybutów z poziomu kodu;
- umie uzupełnić fragment kodu źródłowego, wzorując się na fragmencie podobnym;
- poszukuje, odkrywa i stosuje w praktyce strategię wygranej w układance alfabetycznej;
- odczuwa satysfakcję z tego, że znalazł strategię wygranej i potrafi ułożyć wszystkie litery.

[opcjonalnie] Uczeń zaawansowany...

- zna sposób definiowania i wiązania wywołań Menu z procedurami obsługi zdarzeń.

Materiał nauczania-uczenia się:

- quasi-losowe rozrzucanie (układy rozwiązywalne); poszukiwanie strategii porządkowania;
- programowanie obiektowe z obsługą zdarzeń sterowanych przez gracza za pomocą myszy;
- struktury języka – iteracja w dostępie do elementów tablicy (pętla w pętli); indeksy (Tag); tworzenie obiektów TPanel.Create(); właściwości obiektów: Caption, Visible ...; kodowanie liter Char(); obsługa kliknięć: procedura ClickAction(); tworzenie opcji Menu i przypisywanie ich do procedur realizujących wybraną opcję.

Metody, działania:

- zajawka inspirująca – krótki pokaz z wirtualną grą w puzzle przesuwane (np. z Internetu);
- gra dydaktyczna – rozwiązywanie układanki 3x3 z użyciem 8 kartek z literami A÷H;
- metoda problemowa – sposób ustawiania ostatniego rzędu: układy nierozwiązywalne;
- metoda projektu – dobór właściwości widżetów TPanel imitujących elementy układanki 5x5;
- metoda ćwiczebna – uzupełnianie kodu: przenoszenie właściwości między obiektami;
- gry logiczne – samodzielne rozwiązywanie układanek o różnych poziomach trudności.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- trafnie operacjonalizuje i objaśnia pojęcia: kody liter, tablica, menu, obsługa zdarzeń;
- tworzy foremną macierz widżetów TPanel i modyfikuje ich atrybuty z poziomu kodu;
- prawidłowo uzupełnia fragmenty kodu źródłowego i doprowadza do działania implementacji;
- samodzielnie stosuje w praktyce strategię porządkowania w układance alfabetycznej;
- chętnie rozwiązuje zadanie uporządkowania całej macierzy o rozmiarach 5x5.

[opcjonalnie] Uczeń zaawansowany...

- sam prawidłowo tworzy komponenty Menu oraz wiąże je z procedurami obsługi zdarzeń.

Czynności uczniów	Działania nauczyciela	Materiał
Oglądają krótką prezentację. Poznają zasadę gry.	Pokazuje przykład gry i zachęca do przyjrzenia się zasadzie działania.	Pokaz z projektora lub *online*: gry.pl/gra/Jungle-Squares.html
Próbują rozwiązać układankę 3x3, przesuwając kartki na wolne pola. Rozpoznają potrzebę rozrzucania, zamiast losowania.	Inspiruje uczniów do samodzielnego ułożenia wszystkich liter. Ukazuje brak rozwiązania po zamianie miejsc 2 ostatnich liter.	Po 8 małych kwadratowych kartek z literami od A do H (po jednej literze na kartce).

Analizują struktury Menu oraz poznają powiązania opcji Menu z procedurami ich obsługi.	Objaśnia sposób tworzenia Menu jako komponenty i właściwości widżetu TMainMenu oraz obsługę zdarzeń.	Środowisko Lazarus & FreePascal. Okno przykładowej implementacji, okno dialogowe *Edytor Menu* oraz *drzewo Inspektora obiektów*
Poznają alternatywną metodę generowania z poziomu kodu wielu obiektów w matrycy 5x5. Poznają funkcję Char()	Omawia struktury kodu, zwłaszcza programowy sposób kreowania i rozmieszczania obiektów TPanel oraz sposób kodowania liter.	Kod przykładowej implementacji i okno Edytora źródeł Lazarusa. Język FreePascal.
Poznają sposób indeksowania obiektów przez atrybut Tag.	Wyjaśnia indeksowe sterownie procedurą obsługi wielu obiektów.	Okno *Edytora źródeł* w Lazarusie. Język FreePascal.
Uzupełniają luki w procedurach obsługi quasi-animacji obiektów.	W razie trudności wspiera w pisaniu brakującego kodu.	Okno *Edytora źródeł* w Lazarusie. Język FreePascal.
Uruchamiają implementację i sprawdzają jej poprawność.	Weryfikuje prawidłowość funkcjonowania gry.	Oferty poleceń w menu *Uruchom* i Okno *Komunikaty* w Lazarusie.
Umieszczają swój projekt w e-Repozytorium. Opisują swe dokonania w e-Portfolio.	Formułuje i sprawdza zadania obligatoryjne – dokumentowania wytworów i osiągnięć.	Internet, przeglądarka. Funkcje Serwisu e-Swoi
Ćwiczą układanie liter.	Naprowadza na strategię wygranej.	Własna implementacja gry.
Samodzielnie rozwiązują różne odmiany gier układanek.	Zachęca do ćwiczenia innych strategii. W razie trudności wspiera poprzez pytania naprowadzające.	Przykłady układanek *online*, np.: gry.pl/gry/przesuwane-ukladanki/przesuwane-ukladanki.html

UWAGA: *Zakres omawiania struktur języka dobiera trener adekwatnie do możliwości percepcyjnych uczniów.*

Zadania rozszerzające:

- Rozbuduj układankę do wymiaru 6x6, zawierającą wszystkie 35 liter polskiego alfabetu, tj.: A, Ą, B, C, Ć, D, E, Ę ... Z, Ź, Ż (trudność wskutek nieciągłości kodów polskich liter).

lub:

- Zaimplementuj puzzle działające na zasadzie faktycznego przesuwania widżetów TImage, (inaczej niż tutaj, tj. nie na zasadzie przepisywania zawartości i odkrywania /ukrywania pól).

Uwaga: Te trudne zadania są przeznaczone do realizacji poza zajęciami, lecz można podjąć je z grupą zaawansowaną.

Konspekt-scenariusz **Modułu B2.7**

Stanisław Ubermanowicz

Temat: Odkrywanie i stosowanie algorytmu

Nazwa implementacji: Wieże Hanoi

Opis implementacji: Wizualizacja strategii wygranej w grze decyzyjnej, polegającej na przenoszeniu obiektów o różnej wielkości między 3 cokołami tak, aby obiektu większego nie stawiać na mniejszym. Początkowo na pierwszym cokole znajduje się 5 obiektów, ustawionych jeden na drugim. Najmniejszy obiekt przemieszcza się w co drugim, nieparzystym ruchu, a podczas ruchów parzystych pozostają tylko jedyne możliwości ułożenia obiektu mniejszego na większym.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- kształtowanie u uczniów umiejętności programowania wizualno-obiektowo-zdarzeniowego;
- zapoznanie ze strategią wygranej, prowadzącą najkrótszą drogą do rozwiązania łamigłówki;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- wzbudzenie motywacji do przyswojenia skutecznego sposobu optymalnego działania.

b) szczegółowe: Uczennica/uczeń...

- ma przyswojone i rozumie pojęcia: strategia, algorytm, animacja, wizualizacja;
- zna zasadę ustalania położenia obiektów na ekranie w układzie współrzędnych;
- umie uzupełnić fragmenty kodu źródłowego, wzorując się na strukturach podobnych;
- dostrzega i stosuje w praktyce strategię wygranej w łamigłówce „Wieże Hanoi";
- odczuwa satysfakcję z tego, że zrozumiał algorytm i potrafi zrealizować zadanie.

[opcjonalnie] Uczeń zaawansowany...

- umie zaprojektować implementację ilustrującą strategię gry „Wieże Hanoi".

Materiał nauczania-uczenia się:

- zintegrowane środowisko programowania: okna edycji, inspekcji obiektów i kompilacji;

- poszukiwanie strategii wygranej poprzez obserwację, odkrywanie optymalnego algorytmu;
- programowanie obiektowe, z procedurą animacji sekwencyjnej, sterowanej Timerem;
- struktury języka – instrukcje: warunkowa [if... else...]; procedura Timer1Timer();
- rozdzielczość (piksele), wymiary i współrzędne obiektów [Width, High, Left, Top].

Uwaga: Zakres omawiania struktur kodu dobiera trener adekwatnie do możliwości percepcyjnych uczniów.

Metody, działania:
- zajawka inspirująca – przekaz legendy o niewyobrażalnie długotrwałej pracy mnichów;
- gra dydaktyczna – próba rozwiązania łamigłówki (np. z użyciem 3 monet różnej średnicy);
- metoda problemowa – próba odkrycia i opisu prawidłowości tworzącej strategię wygranej;
- metoda projektu – tworzenie na ekranie obiektów imitujących 3 cokoły i 5 elementów wieży;
- badanie – analiza struktur kodu źródłowego animacji obiektów z algorytmem iteracyjnym;
- programowanie – uzupełnianie części kodu źródłowego w miejscach celowo zostawionych luk;
- zabawa *online* – samodzielne rozwiązanie łamigłówki z pozycji pośredniej nieregularnej.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...
- trafnie operacjonalizuje i objaśnia pojęcia: strategia, algorytm, animacja, wizualizacja;
- wpisuje właściwe parametry umiejscawiające obiekty na ekranie w odpowiednim miejscu;
- prawidłowo uzupełnia fragmenty kodu źródłowego i doprowadza do działania implementacji;
- opisuje werbalnie i optymalnie realizuje strategię wygranej w łamigłówce „Wieże Hanoi";
- chętnie rozwiązuje utrudnione zadanie uporządkowania z pozycji pośredniej nieregularnej.

[opcjonalnie] Uczeń zaawansowany...
- projektuje prawidłowo od podstaw implementację ilustrującą strategię gry „Wieże Hanoi".

Czynności uczniów	Działania nauczyciela	Materiał
Wysłuchują legendę o mnichach. Oglądają krótki pokaz układanki.	Opowiada pseudo-legendę i szybko przestawia klocki zabawki *Wieży* zgodnie z zasadą.	Opis legendy i strategii: lordya314159.livejournal.com/29140.html Zabawka *Wieża*

Próbują rozwiązać łamigłówkę np. z trzema różnymi monetami.	Inspiruje uczniów do samodzielnego ułożenia *złotych krążków*.	Po 3 obiekty o różnej średnicy (np. monety, krążki, tacki, klocki).
Oglądają pełną wizualizację gry i poszukują strategii wygranej.	Podczas animacji zachęca do prób odkrycia strategii. Naprowadza na zasadę ruchu najmniejszego krążka.	Wzorcowa implementacja z animacją układania krążków, wyświetlana z projektora.
Poznają optymalny algorytm najkrótszego rozwiązania.	Wyjaśnia kluczowe dwa kroki optymalnego algorytmu.	Zabawka-układanka *Wieża* lub inne 5 obiektów różnej średnicy.
Ćwiczą określanie wielkości obiektów i ich usytuowanie na współrzędnych ekranu.	Prosi o modyfikowanie projektu. Uzgadnia z uczniami optymalne rozmiary obiektów i ich położenie.	Środowisko Lazarus & FreePascal. Projekt wzorcowej implementacji. Okno *Inspektora obiektów*: atrybuty [Width, High, Left, Top]
Poznają mechanizm okresowego wywoływania procedur i obsługę zdarzeń cyklicznych.	Omawia struktury implementacji, a szczególnie rolę *widżetu Timer* i procedurę Timer1Timer().	Projekt wzorcowej implementacji i okno *Edytora źródeł* Lazarusa. Język FreePascal.
Uzupełniają luki w procedurach obsługi animacji obiektów.	W razie trudności wspiera w pisaniu brakującego kodu.	Okno *Edytora źródeł* w Lazarusie. Język FreePascal.
Uruchamiają implementację i sprawdzają jej poprawność.	Weryfikuje prawidłowość funkcjonowania wizualizacji.	Oferty poleceń w menu *Uruchom* i Okno *Komunikaty* w Lazarusie.
Umieszczają swój projekt w e-Repozytorium. Opisują swe dokonania w e-Portfolio.	Formułuje i sprawdza zadania obligatoryjne – dokumentowania wytworów i osiągnięć.	Internet, przeglądarka. Funkcje Serwisu e-Swoi
Samodzielnie rozwiązują układy z pozycji pośredniej nieregularnej.	Zachęca do ćwiczenia strategii. W razie trudności wspiera poprzez pytania naprowadzające.	Implementacja gry *online*, np.: wipos.p.lodz.pl/zylla/games/hanoi5p.html

UWAGA: *Początkujący uczniowie powinni tylko uzupełniać celowo usunięte obiekty i fragmenty kodu źródłowego.*

Zadania rozszerzające:

- Rozszerz kod źródłowy implementacji Wieże Hanoi w taki sposób, aby obiektów przenoszonych (tj. bloków tworzących wieżę) było więcej niż w pierwotnym projekcie.

lub:

- Zaprojektuj i wykonaj bardziej realistyczne obiekty graficzne bloków tworzących wieżę, a następnie podstaw je do implementacji w miejsce obiektów o uproszczonej grafice.

Formułujemy treści poleceń możliwych do dalszego wykonania przez ucznia samodzielnie bądź ze wsparciem; np. rozbudowa implementacji, inny sposób wykonania lub odmienny projekt o podobnej funkcjonalności. Te zadania są zasadniczo przeznaczone do realizacji poza zajęciami, lecz można podjąć je z grupą zaawansowaną.

Konspekt-scenariusz **Modułu B2.8**

Stanisław Ubermanowicz

Temat: Namiastka sztucznej inteligencji

Nazwa implementacji: Gra logiczna NIM

Opis implementacji: Realizacja gry logicznej, w której chodzi o to, aby podczas naprzemiennego pobierania obiektów z jednego rzędu na planszy uniknąć konieczności zabrania obiektu ostatniego. Losowane są różne układy do 10 obiektów w każdym z trzech rzędów. Komputer ma zaprogramowaną strategię wygranej, dlatego rozpoczyna gracz, aby mieć szansę zwycięstwa.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- kształtowanie u uczniów umiejętności programowania wizualno-obiektowo-zdarzeniowego;
- zapoznanie z implementacją zawierającą logikę działań imitujących sztuczną inteligencję;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- wzbudzenie motywacji do działań twórczych, inferencyjnych i konstruktywistycznych.

b) szczegółowe: Uczennica/uczeń...

- ma przyswojone i rozumie pojęcia: *warunki, operatory logiczne, sterowanie, indeksy;*
- zna zasadę kreowania kolekcji obiektów TImage i nadawania im atrybutów z poziomu kodu;
- umie uzupełnić fragmenty kodu źródłowego, wzorując się na strukturach podobnych;
- wie, jak stosować w praktyce strategię wygranej poprzez analizę parzystości grup binarnych;
- odczuwa satysfakcję z tego, że zrozumiał strategię i wygrywa z grającymi bezbłędnie.

[opcjonalnie] Uczeń zaawansowany...

- umie zaimplementować grę NIM w wersji dla dwóch osób, bez kodu sztucznej inteligencji.

Materiał nauczania-uczenia się:

- poszukiwanie strategii wygranej, prowadzącej do układów końcowych: 1-1-1, 2-2, 3-2-1;

- programowanie obiektowe z obsługą zdarzeń sterowanych przez gracza za pomocą myszy;
- instrukcje ukazywania /ukrywania obiektów oraz blokowania repetycji szybkich kliknięć;
- struktury języka – indeksowanie obiektów TImage; sterowanie właściwością Visible; funkcje Random(), GetBitmap(); instrukcje if... then...; operatory: AND, XOR.

Uwaga: Zakres omawiania struktur języka dobiera trener adekwatnie do możliwości percepcyjnych uczniów.

Metody, działania:
- zajawka inspirująca – pokaz zasad gry za pomocą rekwizytów lub prezentacji z projektora;
- gra dydaktyczna – uczniowie grają parami, analizując końcówki z max. 6 rekwizytami;
- metoda problemowa – próba odkrycia strategii wygranej w końcowej fazie gry;
- metoda projektu – analiza procedury tworzenia obiektów z obrazkami w trzech rzędach;
- metoda ćwiczebna – analiza procedur obsługi gry: losowanie, ukrywanie obiektów;
- operacjonalizacja – zobrazowanie klucza do wygranej, liczenie w systemie dwójkowym;
- gra logiczna – gra z komputerem, z zastosowaniem strategii parzystości grup binarnych.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...
- trafnie operacjonalizuje i objaśnia pojęcia: *warunki, operatory logiczne, sterowanie, indeksy;*
- z poziomu kodu prawidłowo tworzy obiekty TImage lub modyfikuje ich atrybuty;
- prawidłowo uzupełnia fragmenty kodu źródłowego i doprowadza do działania implementacji;
- opisuje werbalnie i optymalnie realizuje strategię wygranej w końcowej fazie gry NIM;
- chętnie stosuje w praktyce strategię wygranej poprzez analizę parzystości grup binarnych.

[opcjonalnie] Uczeń zaawansowany...
- tworzy od podstaw prawidłowo działającą implementację gry NIM dla dwóch osób.

Czynności uczniów	Działania nauczyciela	Materiał
Oglądają prezentację. Poznają grę o strategii „ostatni przegrywa".	Pokazuje przykład gry i zachęca do przyjrzenia się zasadzie działania.	Wzorcowa implementacja gry NIM wyświetlana z projektora.
Grają parami analizując taktykę wygrania w układach końcowych. Odkrywają strategię wygranej przy małej liczbie bierek.	Zachęca do analizy możliwych przypadków pod koniec gry. Prosi o słowne opisywanie strategii podczas 2 i 3 ostatnich ruchów.	Po 6 małych przedmiotów (np. patyczki, bierki, kamyczki).

Rozpoznają obiekty na oknie gry: przycisk Start i napisy. Analizują kolekcję obrazków w TImageList.	Wspiera w analizie roli widżetów zastosowanych we wzorcowej implementacji gry.	Środowisko Lazarus & FreePascal. Okno przykładowej implementacji; okno dialogowe *Edytor ImageList.*
Utrwalają metodę generowania z poziomu kodu do 10 obiektów w matrycy 3 rzędów.	Omawia struktury kodu, zwłaszcza programowy sposób kreowania i rozmieszczania obiektów TImage.	Kod przykładowej implementacji i okno *Edytora źródeł* Lazarusa. Język FreePascal.
Poznają sposób sterowania liczbą obiektów widocznych na ekranie.	Wyjaśnia procedury obsługujące ukazywanie /ukrywanie obiektów.	Okno *Edytora źródeł* w Lazarusie. Język FreePascal.
Uzupełniają luki w procedurach kreowania i ukrywania obiektów.	W razie trudności wspiera w pisaniu brakującego kodu.	Okno *Edytora źródeł* w Lazarusie. Język FreePascal.
Uruchamiają implementację i sprawdzają jej poprawność.	Weryfikuje prawidłowość funkcjonowania gry.	Oferty poleceń w menu *Uruchom* i Okno *Komunikaty* w Lazarusie.
Umieszczają swój projekt w e-Repozytorium. Opisują swe dokonania w e-Portfolio.	Formułuje i sprawdza zadania obligatoryjne – dokumentowania wytworów i osiągnięć.	Internet, przeglądarka. Funkcje Serwisu e-Swoi
Starają się zrozumieć strategię prowadzącą do wygranej. Głośno proponują optymalny ruch dla gracza.	Wyjaśnia strategię: rysuje po kilka kresek w 3 rzędach, zaznacza pary grup binarnych i wymazuje obiekty niepasujące do par (różne układy).	Szkolna tablica, pisak lub kreda i gąbka do zmazywania.
Ćwiczą w grze z komputerem zastosowanie strategii wygranej.	Wspiera w przypadkach trudności ze zrozumieniem strategii wygranej.	Własna implementacja gry.
Grają *online* w grę NIM w wersji „ostatni wygrywa".	Zachęca do poznania implementacji z odwrotną strategią w Scratchu.	Gra online: scratch.mit.edu/projects/jehsom/50329

Zadania rozszerzające:

- Zmień wystrój graficzny opracowanej gry NIM według całkowicie własnego projektu.

lub:

- Rozbuduj grę NIM tak, aby losowane było w każdym z trzech rzędów do 16 obiektów.

lub:

- Zmień grę NIM tak, aby generowane były cztery rzędy obiektów i aby dla takich układów prawidłowo funkcjonowała procedura „sztucznej inteligencji" ze strategią wygranej.

Uwaga: Zadania te przeznaczone są do realizacji poza zajęciami, lecz można podjąć je z grupą zaawansowaną.

Konspekt-scenariusz **Modułu B3.1**

Krzysztof Bytow

Temat: Funkcjonalność modułu-interfejsu

Nazwa implementacji: Środowisko mikrokontrolera

Opis implementacji: Zastosowanie modułu-interfejsu Arduino oraz obsługa interaktywnego terminala Arduino IDE, służącego do programowania mikrokontrolera. Prezentacja i wyjaśnienie sposobu zestawiania połączeń na podstawie dokumentacji ilustrującej montaż układów ćwiczeniowych. Podłączenie i sterowanie diodą elektroluminescencyjną w różnych wariantach. Zaimplementowanie kodu do wyświetlania tekstów oraz do sterowania diodą wbudowaną w moduł-interfejs.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- zapoznanie ze środowiskiem do konstruowania i programowania układów mechatronicznych;
- formowanie kreatywności i sprawności w montowaniu i rozbudowie modułów-interfejsów;
- wzbudzenie satysfakcji z tego, że działa zmontowany własnoręcznie układ elektroniczny;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- doskonali umiejętność łączenia, konfiguracji i programowego sterowania diodami;
- potrafi obsługiwać terminal do pisania kodu sterującego;
- stosuje elementy kodu do tworzenia programów sterujących moduł-interfejs;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- dioda elektroluminescencyjna, button;
- 3 rezystory 220 Ω;
- dokumentacja techniczna mikrokontrolera Atmega 328 z układu Arduino.

Metody, działania:

- zajawka inspirująca – pokaz działania robota MAOR opartego na układzie Atmega;
- pogadanka i dyskusja – zasada działania i programowania zestawu montażowego Arduino;
- prezentacja multimedialna – pokaz wykorzystania mikrokontrolerów i omówienie zestawu;
- metoda ćwiczebna – montaż przykładowych układów sterowania diodami;
- metoda ćwiczebna – wprowadzanie kodu sterującego i testowanie działania układu;
- metoda ćwiczebna – modyfikowanie fragmentów kodu i obserwowanie skutków zmian.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- trafnie objaśnia pojęcia: *mikrokontroler; port USB; dioda elektroluminescencyjna; button; opornik;*
- poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino;
- stosuje elementy kodu do modyfikacji programów sterujących moduł-interfejs;
- poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
- potrafi podłączyć diodę elektroluminescencyjną oraz RGB;
- steruje diodą elektroluminescencyjną oraz modyfikuje treść wyświetlanych komunikatów.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Przeprowadza pokaz działania robota MAOR opartego na układzie Atmega.	• Pojęcia: mikrokontroler, button, opornik, dioda elektroluminescencyjna; http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Mikrokontroler http://arduino.cc/en/Tutorial/Button • Wprowadzenie do środowiska Arduino; http://e-swoi.pl/wiki/article/arduino-podstawy/ http://e-swoi.pl/wiki/article/mechatronika-faq/ • Przypomnienie podstawowych zasad dotyczących napięcia i prądu; • Filmy instruktażowe.
	Prezentuje układ Arduino, na którym będą prowadzone ćwiczenia. Omawia elementy wchodzące w skład zestawu. Uruchamia środowisko programistyczne Arduino IDE, objaśniając poszczególne funkcje programu. Pokazuje wstępną konfigurację programu w celu komunikacji między komputerem a modułem.	
Montują przykładowe układy sterowania diodami. Wprowadzenie kodu sterującego i testowanie działania układu. Modyfikowanie fragmentów kodu i obserwowanie skutków zmian.	Zachęca uczennice i uczniów do samodzielnego podłączenia układu i do zaprogramowania mikrokontrolera przykładową procedurą obsługi diody. Podpowiada, jakie szczegółowe działania muszą podjąć uczennice i uczniowie, aby ich układ funkcjonował prawidłowo, w pełni zgodnie z zadaniem. Omawia kod źródłowy i jego poszczególne elementy.	
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	

Zadania rozszerzające:

- Zmodyfikuj program tak, aby dioda na płytce Arduino mrugała w odstępach losowych.

Konspekt-scenariusz **Modułu B3.2**

Krzysztof Bytow

Temat: Sterowanie z użyciem fotorezystora

Nazwa implementacji: Pomiar oświetlenia

Opis implementacji: Wizualizacja działania dodatkowych elementów zestawu modułu-interfejsu z układem Arduino. Wykorzystanie funkcji przetwornika analogowo-cyfrowego do budowy układów pomiarowych. Istota funkcjonowania i zastosowania fotorezystora. Konstruowanie i oprogramowanie układów do odczytu stanu czujnika na przykładzie interfejsu do pomiaru natężenia światła. Prezentacja wyników z wykorzystaniem diod elektroluminescencyjnych oraz diody RGB.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- rozpoznawanie środowiska do konstruowania i programowania układów mechatronicznych;
- formowanie kreatywności i sprawności w montowaniu i rozbudowie modułów-interfejsów;
- wzbudzenie satysfakcji z tego, że działa zmontowany własnoręcznie układ elektroniczny;
- rozwijanie innowacyjności – koncypowanie, do czego można zastosować moduły-interfejsy;

b) szczegółowe: Uczennica/uczeń...

- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- rozwija umiejętność tworzenia interfejsów mierzących i wyświetlających stan czujników.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- fotorezystor, button, 3 diody elektroluminescencyjne, dioda RGB;
- 3 rezystory 220Ω, rezystor 10kΩ.

Metody, działania:

- zajawka inspirująca i dyskusja – pokaz sposobów odczytu natężenia światła;
- metoda ćwiczebna – skonstruowanie i oprogramowanie modułu-interfejsu do sterowania diodą RGB;
- metoda ćwiczebna – skonstruowanie i oprogramowanie wskaźnika natężenia światła;
- metoda ćwiczebna – modyfikowanie lub rozbudowa (np. połączenie) implementacji.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- zgodnie z zasadami działania podłącza czujnik pomiarowy: fotorezystor;
- prawidłowo buduje i oprogramowuje moduł-interfejs wskazujący poziomy oświetlenia;
- modyfikuje i rozbudowuje pomiarowe układy elektroniczne oraz kody źródłowe;
- trafnie używa sformułowań: *czujnik*, *wejście analogowe*, *przetwornik A/D*.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Uczestniczą w dyskusji.	Dyskutuje z uczniami nad sposobami pomiaru natężenia światła.	
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Prezentuje, w jaki sposób działa fotorezystor i przedstawia zależność jego rezystancji od światła; pokazuje sposób odczytu natężenia światła.	• Pojęcia: mikrokontroler, fotorezystor, opornik, button, dioda elektroluminescencyjna, dioda RGB.
Konstruują i oprogramowują moduł-interfejs do sterowania diodą RGB; konstruują i oprogramowują wskaźnik natężenia światła; następnie modyfikują lub rozbudowują (np. łączą) implementację.	Zachęca do samodzielnego montażu wybranych układów na podstawie dostarczonych instrukcji. Nadzoruje działania, aby implementacje wykonywane były prawidłowo. W zależności od możliwości, uczniowie montują część z przykładowych układów pomiarowych, wprowadzając kody źródłowe i wspólnie analizują istotę działania danej implementacji. Modyfikują wartości współczynników w celu obserwacji zmian.	• Filmy instruktażowe; • Tutoriale: http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Fotorezystor http://pl.wikipedia.org/wiki/Opornik http://arduino.cc/en/Tutorial/Button
	Formułuje zadania: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	

Konspekt-scenariusz **Modułu B3.3**

Krzysztof Bytow

Temat: Pomiar temperatury i wyświetlacza LCD

Nazwa implementacji: Termometr cyfrowy

Opis implementacji: Wizualizacja działania elementu zestawu modułu-interfejsu z układem Arduino. Wykorzystanie funkcji przetwornika analogowo-cyfrowego do budowy układu pomiarowego. Istota funkcjonowania i zastosowania termistora. Podłączenie i sterowanie wyświetlaczem LCD z wykorzystaniem płytki stykowej. Zaimplementowanie kodu do wyświetlania tekstów. Konstruowanie i oprogramowanie układu do odczytu stanu czujnika na przykładzie interfejsu do pomiaru temperatury. Prezentacja odczytu temperatury i skrajnych wartości na LCD oraz na ekranie monitora.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- rozpoznawanie środowiska do konstruowania i programowania układów mechatronicznych;
- formowanie kreatywności i sprawności w montowaniu i rozbudowie modułów-interfejsów;
- ćwiczenie umiejętności tworzenia interfejsu mierzącego i wyświetlającego stan czujnika temperatury;
- wzbudzenie satysfakcji z tego, że działa zmontowany własnoręcznie układ elektroniczny;

b) szczegółowe: Uczennica/uczeń...

- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- potrafi przedstawić skale i jednostki temperatury oraz zależności między nimi;
- wykorzystuje komputer, aby dokonał potrzebnych przeliczeń do zmian skal temperatury.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- czujnik temperatury MCP9700 i dokumentacja techniczna;
- potencjometr 10 kΩ i wyświetlacz LCD.

Metody, działania:

- zajawka inspirująca i dyskusja – pokaz sposobów odczytu temperatury;

- prezentacja multimedialna – układy do odczytu temperatury i wizualizacji jej zmienności;
- metoda ćwiczebna – zaimplementowanie modułu-interfejsu do pomiaru temperatury;
- metoda ćwiczebna – modyfikowanie lub rozbudowa (np. połączenie) implementacji o dodanie procedury uśredniającej pomiary.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- zgodnie z zasadami działania podłącza czujnik pomiarowy termistor;
- prawidłowo buduje i oprogramowuje moduł-interfejs służący do pomiaru temperatury;
- uruchamia ukazywanie odczytów na wyświetlaczu LCD lub w środowisku Linux;
- modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy;
- dokonuje przeliczenia wartości pomiarów temperatury do innych skal;
- trafnie używa sformułowań: *czujnik, stopnie Celsjusza, stopnie Fahrenheita, Kelvin, czułość, wejście analogowe, przetwornik A/D.*

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w prezentacji i pokazie, zadają pytania, wyjaśniają wątpliwości.	Prezentuje złożony układ Arduino z zaimplementowanym programem do odczytu temperatury z czujnika. Omawia zasadę odczytu, zwraca uwagę na dokładność pomiaru, zachęca do zapoznania z dokumentacją czujnika. Porusza temat sposobów zwiększania dokładności, co ma na nią wpływ i jak można ją poprawić. Wykorzystuje prezentację multimedialną.	• Pojęcia: czujnik, przetwornik A/D, LCD, potencjometr, biblioteka; • Filmy instruktażowe; • Prezentacja multimedialna • Filmy dostępne w serwisie http://www.youtube.com/ hasła kluczowe: arduino lcd, arduino pomiar temperatury • Tutoriale: http://e-swoi.pl/wiki/article/arduino-podstawy/ http://arduino.cc/en/Tutorial/AnalogInput • Przedstawienie zależności napięcia wyjściowego od temperatury;
Biorą udział w dyskusji.	Prowadzi dyskusję dotyczącą sposobów odczytu temperatury.	
Wykonują implementację modułu-interfejsu do pomiaru temperatury; modyfikują lub rozbudowują implementację o dodanie procedury uśredniającej pomiary.	Zachęca uczennice i uczniów do samodzielnego podłączenia układu i zaprogramowania przykładowym programem. Omawia kod źródłowy i jego poszczególne elementy. Prezentuje podłączenie wyświetlacza LCD oraz konfigurację w kodzie źródłowym. W dalszej części wyjaśnia i prezentuje przesyłanie wyników do komputera i sposoby na ich obróbkę.	• Dokumentacja techniczna http://ww1.microchip.com/downloads/en/DeviceDoc/21942e.pdf • Zależność między temperaturą wyrażoną w stopniach Celsjusza t [°C] a wyrażoną w Kelwinach t [K]; • Zależność między temperaturą wyrażoną w stopniach Celsjusza t [°C] i Fahrenheita t [°F];
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	

Zadania rozszerzające:

- Zmodyfikuj kod B3.3a sterujący tak, aby wyświetlać temperaturę w środowisku Linux w różnych skalach.

Konspekt-scenariusz **Modułu B3.4**

Krzysztof Bytow

Temat: Interakcja człowiek – moduł-interfejs

Nazwa implementacji: Pamięć i zręczność

Opis implementacji: Budowa układu i programu do symulacji losowania jednej z sześciu liczb jak w kostce do gry. Prezentacja wyniku losowania z wykorzystaniem diod elektroluminescencyjnych. Rozwijając wiedzę o zastosowaniu diod elektroluminescencyjnych, układ moduł-interfejs zostaje rozbudowany o kolejne elementy. Efektem jest opracowanie implementacji pozwalającej ćwiczyć pamięć i zręczność.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- doskonalenie umiejętności pracy w środowisku implementowania układów mechatronicznych;
- formowanie kreatywności i sprawności w montowaniu i rozbudowie modułów-interfejsów;
- wzbudzenie satysfakcji z wytworzenia zaawansowanego układu gry losowo-zręcznościowej;
- rozwijanie innowacyjności i kreatywności w rozbudowie układu o wyświetlacz LCD i buzzer;

b) szczegółowe: Uczennica/uczeń...

- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- ma doświadczenie w pracy w środowisku do programowania wizualnego układów mechatronicznych;
- rozwija umiejętności łączenia i programowego sterowania elementów wykonawczych;
- rozwija umiejętności sterowania elementami zestawu modułu-interfejsu.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- 4 lub 6 czerwonych diod LED, 4 lub 6 rezystorów 220 Ω, 1 lub 4 przyciski button;
- opcjonalnie wyświetlacz LCD 2x16; potencjometr 10 kΩ i buzzer.

Metody, działania:

- zajawka inspirująca i dyskusja – pokaz elektronicznej kostki do gry;
- prezentacja multimedialna – pokaz filmu instruktażowego do implementacji;

- metoda ćwiczebna – zmontowanie układu i wgranie kodu do modułu-interfejsu;
- metoda ćwiczebna – rozbudowa układu o dodatkowe elementy wykonawcze;
- gra logiczno-zręcznościowa – zabawa w interakcji z układem mechatronicznym

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- samodzielnie projektuje i oprogramowuje układy na platformie Arduino;
- implementuje układy interfejsu z diodami LED i przyciskami button;
- programowo steruje wejściami i wyjściami cyfrowymi;
- prawidłowo analizuje i pisze objaśnienia kodu źródłowego implementacji;
- rozbudowuje moduł-interfejs, dodając wyświetlacz LCD i buzzer;
- trafnie używa sformułowań: *PULLUP, wejście/wyjście cyfrowe, typ logiczny (boolean).*

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie.	Przeprowadza pokaz elektronicznej kostki do gry.	• Pojęcia: opornik, mikrokontroler, dioda elektroluminescencyjna, button, typ logiczny, pullup; • Filmy instruktażowe; • Prezentacja multimedialna filmy dostępne w serwisie http://www.youtube.com/ hasła kluczowe: arduino led.
Biorą udział w dyskusji.	Zachęca uczniów do zastanowienia się nad sposobami wyboru losowych liczb z zakresu od 1-6.	
Współuczestniczą w prezentacji, zadają pytania, wyjaśniają wątpliwości.	Przekazuje uczniom sposób, w jaki dokonuje się zapisu i odczytu wartości z tablic. Wraz z uczniami analizuje budowę kodu symulacji działania kostki do gry i kodu gry zręcznościowo - pamięciowej.	
Samodzielnie montują układ i wgrywają kod do modułu-interfejsu. Rozbudowują układ o dodatkowe elementy wykonawcze. Sprawdzają działanie poprzez grę – zabawa w interakcji z układem mechatronicznym.	Zachęca uczniów do samodzielnego podłączenia układu i zaprogramowania układu. Podaje wskazówki, jak rozbudować układ o dodatkowe elementy i jak je oprogramować.	• Tutoriale: http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://e-swoi.pl/wiki/article/arduino-podstawy/ http://pl.wikipedia.org/wiki/Opornik http://arduino.cc/en/Tutorial/Button http://arduino.cc/de/Reference/Random
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	

Zadania rozszerzające:

- Rozbuduj układ z implementacji C3.4a o wyświetlacz LCD i zmodyfikuj kod sterujący o sekwencję losowania liczb z prezentacją na dołączonym LCD.

lub:
- Rozbuduj układ z implementacji C3.4a o buzzer i zmodyfikuj kod sterujący, dodając polecenia odtwarzania dźwięku przy losowaniu lub po wylosowaniu liczby.

lub:
- Rozbuduj układ z implementacji C3.4b o buzzer i zmodyfikuj kod sterujący, dodając polecenia odtwarzania dźwięku przy przegranej (tj. przy wduszeniu złego buttona).

Konspekt-scenariusz **Modułu B3.5**

Krzysztof Bytow

Temat: Możliwości S4A i modułu - interfejsu

Nazwa implementacji: Środowisko Scratch S4A i moduł-interfejs

Opis implementacji: Wprowadzenie w świat mikrokontrolerów na przykładzie modułu--interfejsu Arduino oraz jego obsługa w środowisku Scratch (S4A). Prezentacja i wyjaśnienie sposobu zestawiania połączeń na podstawie dokumentacji ilustrującej montaż układów ćwiczeniowych. Wizualizacja działania diody RGB podłączonej do modułu-interfejsu. Sposoby podłączania, sterowania i programowania podzespołów.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń...

- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- rozwija umiejętności sterowania elementami zestawu modułu-interfejsu;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny;
- rozwija umiejętność współpracy z innymi uczennicami i uczniami oraz z nauczycielem.

Materiał nauczania-uczenia się:

- program S4A (Scratch);
- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- 2 buttony; dioda RGB;
- 2 rezystory 10 kΩ; 3 rezystory 220 Ω.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- omówi pojęcia: button, wejście cyfrowe; dioda RGB; opornik; pętla; wyrażenie warunkowe; zmienna; mikrokontroler;
- zmontuje i uruchomi przykładowe układy na podstawie schematów;
- deklaruje zmienne i przypisuje im wartości;
- potrafi obsługiwać środowisko Scratch S4A i zna jego funkcje;
- omówi istotę działania oraz sposób podłączania i sterowania podzespołami: dioda RGB, przycisk;
- angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Prezentuje układ Arduino, na którym będą prowadzone ćwiczenia. Omawia elementy wchodzące w skład zestawu – pokaz sterowania diodą RGB w różnych wariantach (sterowanie programowe, z wykorzystaniem klawiatury, małych przycisków – buttonów).	• Pojęcia: mikrokontroler, dioda elektroluminescencyjna i RGB, opornik, button, wej./wyj. cyfrowe; • Prezentacja multimedialna; • Filmy dostępne w serwisie http://www.youtube.com/ hasła kluczowe: arduino; arduino cube; arduino rgb.
Uczestniczą w pogadance.	Prezentuje, w jaki sposób działa fotorezystor i przedstawia zależność jego rezystancji od światła; pokazuje sposób odczytu natężenia światła.	• Pojęcia: mikrokontroler, fotorezystor, opornik, button, dioda elektroluminescencyjna, dioda RGB.
Konstruują i oprogramowują moduł-interfejsu do sterowania diodą RGB; konstruują i oprogramowują wskaźnik natężenia światła; następnie modyfikują lub rozbudowują (np. łączą) implementację.	Uruchamia środowisko programistyczne Arduino IDE, objaśniając poszczególne funkcje programu. Pokazuje wstępną konfigurację programu w celu komunikacji między komputerem a modułem. Prezentuje, w jaki sposób połączyć Arduino ze Scratchem S4A. Prezentuje wymagany kod do współpracy z oprogramowaniem S4A, a następnie prezentuje możliwości samego środowiska. Po prezentacji kodu wymaganego do sterowania Arduino z poziomu aplikacji S4A można przedstawić kod do sterowania diodą wbudowaną w układ modułu-interfejsu (PIN13), następnie przejść do dalszej części ćwiczenia. Dokonuje zestawienia układów i ich uruchomienia. Omawia elementy składowe programu.	• Filmy instruktażowe; • Tutoriale: http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Fotorezystor http://pl.wikipedia.org/wiki/Opornik http://arduino.cc/en/Tutorial/Button
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące. **Zadania rozszerzające:** • Rozbudować kod o losowy wybór koloru diody RGB, (w dalszej części wykluczyć powtórzenia tego samego koloru).	

Konspekt-scenariusz **Modułu B3.6**

Krzysztof Bytow

Temat: Działanie fotorezystora i potencjometru

Nazwa implementacji: Przetwornik analogowo-cyfrowy

Opis implementacji: Wizualizacja działania dodatkowych elementów zestawu modułu-interfejsu z układem Arduino. Wykorzystanie funkcji przetwornika analogowo-cyfrowego do budowy układów pomiarowych. Istota funkcjonowania i zastosowania fotorezystora i potencjometru. Konstruowanie i oprogramowanie układów do odczytu stanu potencjometru i wartości fotorezystora. Prezentacja odczytów na ekranie monitora oraz z wykorzystaniem diody RGB.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń…

- ma doświadczenie pracy w środowisku do programowania wizualnego układów mechatronicznych;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- rozwija umiejętności tworzenia interfejsów mierzących i wyświetlających stan czujników;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny;
- obmyśla, do czego można zastosować moduły-interfejsy.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- fotorezystor, potencjometr 10 kΩ;
- 3 diody LED; dioda RGB; 3 rezystory 220 Ω; rezystor 10 kΩ.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- zgodnie z zasadami działania podłącza czujnik pomiarowy: fotorezystor;
- prawidłowo buduje i oprogramowuje moduł-interfejs wskazujący odczyty z wejścia analogowego;
- modyfikuje i rozbudowuje pomiarowe układy elektroniczne oraz kody źródłowe;
- trafnie używa sformułowań: czujnik, czułość, wejście analogowe, przetwornik A/D;
- wskazuje zastosowania modułów-interfejsów.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Biorą udział w dyskusji.	Dyskutuje z uczniami nad zasadą działania przetwornika analogowego-cyfrowego.	
Współuczestniczą w prezentacji i pokazie, zadają pytania, wyjaśniają wątpliwości.	Prezentuje, w jaki sposób działa fotorezystor i przedstawia zależność jego rezystancji od światła. Przeprowadza pokaz odczytu natężenia światła. Następnie przedstawia przykład sterowania diodą RGB z wykorzystaniem potencjometru i wejścia analogowego.	• Pojęcia: mikrokontroler, dioda elektroluminescencyjna i RGB, potencjometr, fotorezystor, wej. analogowe; • Prezentacja multimedialna • Filmy dostępne w serwisie http://www.youtube.com/ hasła kluczowe: arduino; arduino rgb; arduino led.
Próbują najpierw samodzielnie, a potem przy wsparciu nauczyciela stworzyć kod i uruchomić układ.	Zachęca uczniów do samodzielnego montażu wybranych układów na podstawie dostarczonych instrukcji. Nadzoruje działania, aby implementacje wykonywane było prawidłowo.	• Filmy instruktażowe; • Tutoriale: http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Mikrokontroler http://e-swoi.pl/wiki/article/arduino-podstawy/ http://s4a.cat/ http://pl.wikipedia.org/wiki/Opornik http://pl.wikipedia.org/wiki/Fotorezystor
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące. **Zadania rozszerzające:** • Budowa trzystopniowego wskaźnika natężenia światła, z wykorzystaniem 3 diod LED lub diody RGB.	

Konspekt-scenariusz **Modułu B3.7**

Krzysztof Bytow

Temat: Środowisko mikrokontrolera

Nazwa implementacji: Funkcjonalność modułu-interfejsu

Opis implementacji: Zastosowanie modułu-interfejsu oraz obsługa interaktywnego terminala Arduino IDE, służącego do programowania mikrokontrolera. Podłączenie i sterowanie diodami LED na przykładzie sygnalizacji świetlnej. Obsługa przycisków i sterowanie buzerem. Zaimplementowanie kodu do sterowania diodą.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń...
- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- doskonali umiejętność łączenia, konfiguracji i programowego sterowania diodami;
- kształtuje umiejętność obsługi terminala do pisania kodu sterującego;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- rozwija sprawność deklarowania podstawowych typów zmiennych, definiowania i przypisywania im wartości;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny.

Materiał nauczania-uczenia się:
- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- 5 diod elektroluminescencyjnych; button; buzzer;
- 5 rezystorów 220 Ω;
- dokumentacja techniczna mikrokontrolera Atmega 328 z układu Arduino.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- trafnie objaśnia pojęcia: mikrokontroler; dioda elektroluminescencyjna; button; opornik; buzzer;
- poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino;
- poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
- steruje diodą elektroluminescencyjną oraz modyfikuje treść wyświetlanych komunikatów.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Przeprowadza pokaz działania robota MAOR opartego na układzie Atmega lub pokaz możliwości działania i zastosowania Arduino w praktyce.	• Pojęcia: mikrokontroler, button, opornik, buzzer, dioda elektroluminescencyjna; http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Mikrokontroler http://arduino.cc/en/Tutorial/Button • Wprowadzenie do środowiska Arduino; http://e-swoi.pl/wiki/article/arduino-podstawy/ • Przypomnienie podstawowych zasad dotyczących napięcia i prądu; • Filmy instruktażowe.
	Prezentuje układ Arduino, na którym będą prowadzone ćwiczenia. Omawia elementy wchodzące w skład zestawu. Uruchamia środowisko programistyczne Arduino IDE, objaśniając poszczególne funkcje programu. Pokazuje wstępną konfigurację programu w celu komunikacji między komputerem, a modułem.	
Próbują najpierw samodzielnie, a potem przy wsparciu nauczyciela uruchomić układ. Wprowadzają kod sterujący i testują działanie układu; modyfikowanie fragmentów kodu i obserwowanie skutków zmian.	Zachęca uczennice i uczniów do samodzielnego podłączenia układu i do zaprogramowania mikrokontrolera przykładową procedurą obsługi diody. Omawia kod źródłowy i jego poszczególne elementy. W dalszej części wyjaśnia i prezentuje podłączenie diod symulujących przejście dla pieszych, jak i działanie buzzera. Podpowiada, jakie szczegółowe działania muszą podjąć uczennice i uczniowie, aby ich układ funkcjonował prawidłowo, w pełni zgodnie z zadaniem.	
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	
	Formułuje zadania rozszerzające: • Zmodyfikuj program tak, aby dioda mrugała w odstępach losowych; • Rozbuduj układ sygnalizacji świetlnej o buzzer, który sygnalizowałby zielone światło dla pieszych.	

Konspekt-scenariusz **Modułu B3.8**

Krzysztof Bytow

Temat: Termometr cyfrowy

Nazwa implementacji: Obsługa wyświetlacza

Opis implementacji: Podłączenie i sterowanie wyświetlaczem LCD z wykorzystaniem płytki stykowej. Zaimplementowanie kodu do wyświetlania tekstów. Konstruowanie i oprogramowanie układu do odczytu stanu czujnika na przykładzie interfejsu do pomiaru temperatury. Prezentacja odczytu temperatury na ekranie monitora i na wyświetlaczu LCD.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych.

b) szczegółowe: Uczennica/uczeń...
- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- poznaje skale i jednostki temperatury oraz zależności między nimi;
- kształtuje umiejętność wykorzystania komputera do wykonania potrzebnych przeliczeń do zmian skal temperatury;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny.

Materiał nauczania-uczenia się:
- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- czujnik temperatury MCP9700 i dokumentacja techniczna;
- potencjometr 10 kΩ i wyświetlacz LCD.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...
- zgodnie z zasadami działania podłącza czujnik pomiarowy termistor;
- prawidłowo buduje i oprogramowuje moduł-interfejs służący do pomiaru temperatury;
- uruchamia ukazywanie odczytów na wyświetlaczu LCD lub w środowisku Linux;

- modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy;
- potrafi przedstawić skale i jednostki temperatury oraz omawia zależności między nimi;
- prawidłowo dokonuje przeliczenia wartości pomiarów temperatury na inne skale;
- trafnie używa sformułowań: *czujnik, stopnie Celsjusza, stopnie Fahrenheita, Kelvin, czułość, wejście analogowe, przetwornik A/D.*

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w prezentacji i pokazie, zadają pytania, wyjaśniają wątpliwości.	Prezentuje złożony układ Arduino z zaimplementowanym programem do odczytu temperatury z czujnika. Omawia zasadę odczytu, zwraca uwagę na dokładność pomiaru, zachęca do zapoznania z dokumentacją czujnika. Porusza temat sposobów zwiększania dokładności, co ma na nią wpływ i jak można ją poprawić. Omawia kod źródłowy i jego poszczególne elementy. Prezentuje podłączenie wyświetlacza LCD.	• Pojęcia: *czujnik, przetwornik A/D, LCD, potencjometr, biblioteka;* • Filmy instruktażowe; • Prezentacja multimedialna; • Filmy dostępne w serwisie http://www.youtube.com/ hasła kluczowe: arduino lcd, arduino pomiar temperatury • Tutoriale: http://e-swoi.pl/wiki/article/arduino-podstawy/ http://arduino.cc/en/Tutorial/AnalogInput przedstawienie zależności napięcia wyjściowego od temperatury;
Biorą udział w dyskusji.	Prowadzi dyskusję dotyczącą sposobów odczytu temperatury.	
Wykonują implementację do pomiaru temperatury oraz do podłączenia i sterowania wyświetlaczem LCD, a następnie: • modyfikują lub rozbudowują (np. łączą) implementację o dodanie procedury uśredniającej pomiary; • tworzą implementację służącą do mierzenia i wyświetlania temperatury na LCD; • prezentują temperaturę wyrażoną w Kelwinach t [K]; • prezentują temperaturę wyrażoną w skali Fahrenheita t [°F];	Zachęca uczennice i uczniów do samodzielnego podłączenia układu i zaprogramowania przykładowym kodem.	• Dokumentacja techniczna http://ww1.microchip.com/downloads/en/DeviceDoc/21942e.pdf • Zależność między temperaturą wyrażoną w stopniach Celsjusza t [°C] a wyrażoną w Kelwinach t [K]; • Zależność między temperaturą wyrażoną w stopniach Celsjusza t [°C] i Fahrenheita t [°F];
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące;	
	Formułuje zadania rozszerzające: • w pracy zespołowej podłączyć 2 czujniki do Arduino, oprogramować, odczyty zaprezentować na wyświetlaczu LCD dodatkowo w dalszej części pomiary z obu czujników zsumować i uśrednić.	

Moduły C

-Język C-

Zajęcia dla szkół sprofilowanych infotechnicznie

Środowisko i zagadnienia

C1

→ język C: zmienne, wyrażenia, pętle

C2

→ język C: tablice, funkcje, struktury, wskaźniki

C3

→ Arduino: tranzystor, hallotron, zegar

Konspekt-scenariusz **Modułu C1.1**

Piotr Fiorek

Temat: Struktura programu, zmienne oraz typy danych

Nazwa implementacji: Nauka języka C – zmienne

Opis implementacji: Poznanie struktury programy, pojęcia zmiennych oraz podstawowych typów danych, jakie zmienne mogą przybierać.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu podstawowych pojęć programowania;
- posiada wiedzę dotyczącą struktury programu w języku programowania;
- kształtuje umiejętność pisania prostego programu zawierającego kilka zmiennych, skompilowania go i uruchomienia oraz eliminacji błędów;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:
- prosty edytor tekstu;
- kompilator języka C np. gcc.

Metody, działania:
- pisanie kodu w edytorze tekstu;
- kompilacja napisanego kodu przy użyciu kompilatora gcc;
- uruchamianie stworzonych programów i analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- omawia podstawowe pojęcia programowania: zmienna, typy zmiennych;
- potrafi omówić strukturę programu w języku C;
- potrafi napisać prosty program zawierający kilka zmiennych, który się kompiluje bez błędów oraz poprawnie działa;
- potrafi obsługiwać kompilator w podstawowym zakresie;
- potrafi uruchomić program z konsoli systemu Linux.

Czynności uczniów	Działania nauczyciela	Materiał
Piszą kod.	Wprowadza pojęcia, wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Pojęcia: *zmienna, typy zmiennych*; edytor tekstu.
Kompilują kod.	Pomaga w zrozumieniu błędów kompilatora.	Kompilator języka C np. gcc
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C1.2**

Piotr Fiorek

Temat: Wyrażenia warunkowe *if-else*

Nazwa implementacji: Nauka języka C – Wyrażenia warunkowe *if-else*

Opis implementacji: Poznanie struktury oraz zastosowania wyrażeń warunkowych *if-else* w języku C.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu podstawowych pojęć programowania;
- kształtuje umiejętności używania wyrażeń warunkowych *if-else* w programie do osiągania zamierzonych celów;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:
- prosty edytor tekstu;
- kompilator języka C np. gcc.

Metody, działania:
- pisanie kodu w edytorze tekstu;
- kompilacja napisanego kodu przy użyciu kompilatora gcc;
- uruchamianie stworzonych programów i analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- omawia podstawowe pojęcia programowania: wyrażenia warunkowe *if-else* i wskazuje, do czego służą;
- potrafi napisać prosty program zawierający wyrażenia warunkowe *if-else* i zachowujący się różnie zależnie od wprowadzonych danych;
- potrafi obsługiwać kompilator w podstawowym zakresie;
- potrafi uruchomić program z konsoli systemu Linux.

Czynności uczniów	Działania nauczyciela	Materiał
Piszą kod.	Wprowadza pojęcia, wspiera uczniów, koryguje błędy, naprowadza, motywuje.	• Pojęcia: *wyrażenia warunkowe if-else*; • Edytor tekstu.
Kompilują kod.	Pomaga w zrozumieniu błędów kompilatora.	Kompilator języka C np. gcc
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C1.3**

Piotr Fiorek

Temat: Wyrażenie warunkowe *switch{ case: ...;}*

Nazwa implementacji: Nauka języka C – Wyrażenie warunkowe *switch {case: ...;}*

Opis implementacji: Poznanie struktury oraz zastosowania wyrażenia warunkowego *switch {case: ...;}*

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym.

b) szczegółowe: Uczennica/uczeń...
- posiada wiedzę z zakresu podstawowych pojęć programowania;
- kształtuje umiejętności używania wyrażeń warunkowych *switch {case: ...;}* w programie do osiągania zamierzonych celów;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:
- prosty edytor tekstu;
- kompilator języka C np. gcc.

Metody, działania:
- pisanie kodu w edytorze tekstu;
- kompilacja napisanego kodu przy użyciu kompilatora gcc;
- uruchamianie stworzonych programów i analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- omawia podstawowe pojęcia programowania: wyrażenia warunkowe *switch {case: ...;}* i wskazuje, do czego służą;
- potrafi napisać prosty program zawierający wyrażenie warunkowe *switch {case: ...;}* i zachowujący się różnie zależnie od wprowadzonych danych;
- potrafi obsługiwać kompilator w podstawowym zakresie;
- potrafi uruchomić program z konsoli systemu Linux.

Czynności uczniów	Działania nauczyciela	Materiał
Piszą kod.	Wprowadza pojęcia, wspiera uczniów, koryguje błędy, naprowadza, motywuje.	• Pojęcia: wyrażenia warunkowe *switch {case: . . .;};* • Edytor tekstu.
Kompilują kod.	Pomaga w zrozumieniu błędów kompilatora.	Kompilator języka C np. gcc
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C1.4**

Piotr Fiorek

Temat: Pętle – *while()* i do *{} while()*

Nazwa implementacji: Nauka języka C – Pętle – *while()* i *do {} while()*

Opis implementacji: Poznanie czym są pętle i jak korzystać z pętli *while* i *do {} while()*

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu podstawowych pojęć programowania;
- kształtuje umiejętności korzystania z pętli *while()*;
- kształtuje umiejętności korzystania z pętli do *{} while()*;
- kształtuje umiejętności pisania prostego programu wykonującego powtarzające się czynności z wykorzystaniem pętli;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:

- prosty edytor tekstu;
- kompilator języka C np. gcc.

Metody, działania:

- pisanie kodu w edytorze tekstu;
- kompilacja napisanego kodu przy użyciu kompilatora gcc;
- uruchamianie stworzonych programów i analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- omawia podstawowe pojęcia programowania: Pętle – *while()* i *do {} while()* i wskazuje, do czego służą;
- potrafi napisać prosty program korzystający z pętli do wykonywania powtarzających się czynności;
- potrafi obsługiwać kompilator w podstawowym zakresie;
- potrafi uruchomić program z konsoli systemu Linux.

Czynności uczniów	Działania nauczyciela	Materiał
Piszą kod.	Wprowadza pojęcia, wspiera uczniów, koryguje błędy, naprowadza, motywuje.	• Pojęcia: Pętle – *while()* i *do {} while();* • Edytor tekstu.
Kompilują kod.	Pomaga w zrozumieniu błędów kompilatora.	Kompilator języka C np. gcc
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C2.1**

Piotr Fiorek

Temat: Tablice i pętla *for()*

Nazwa implementacji: Nauka języka C – tablice i pętla *for()*

Opis implementacji: Poznanie, czym są tablice oraz pętla *for()*

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu podstawowych pojęć programowania;
- kształtuje umiejętności wykorzystywania tablic oraz pętli *for()*;
- kształtuje umiejętności napisania prostego programu wykonującego powtarzające się czynności z wykorzystaniem tablic i pętli;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:
- prosty edytor tekstu;
- kompilator języka C np. gcc.

Metody, działania:
- pisanie kodu w edytorze tekstu;
- kompilacja napisanego kodu przy użyciu kompilatora gcc;
- uruchamianie stworzonych programów i analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- omawia podstawowe pojęcia programowania: Tablice, pętle *for()* i wskazuje, do czego służą;
- potrafi napisać prosty program korzystający z tablic oraz wykorzystać pętlę *for()* do operowania na nich;
- potrafi obsługiwać kompilator w podstawowym zakresie;
- potrafi uruchomić program z konsoli systemu Linux.

Czynności uczniów	Działania nauczyciela	Materiał
Piszą kod.	Wprowadza pojęcia, wspiera uczniów, koryguje błędy, naprowadza, motywuje.	• Pojęcia: Tablice, pętle *for()*; • Edytor tekstu.
Kompilują kod.	Pomaga w zrozumieniu błędów kompilatora.	Kompilator języka C np. gcc
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C2.2**

Piotr Fiorek

Temat: Funkcje

Nazwa implementacji: Nauka języka C – funkcje

Opis implementacji: Nauka tworzenia oraz używania funkcji w języku C

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu podstawowych pojęć programowania;
- kształtuje umiejętności tworzenia własnych funkcji w programach;
- kształtuje umiejętności pisania prostego programu podzielonego na mniejsze funkcje realizujące poszczególne zadania programu;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:

- prosty edytor tekstu;
- kompilator języka C np. gcc.

Metody, działania:

- pisanie kodu w edytorze tekstu;
- kompilacja napisanego kodu przy użyciu kompilatora gcc;
- uruchamianie stworzonych programów i analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- potrafi wyjaśnić, czym są funkcje i kiedy kod należy dzielić na mniejsze funkcje, a także jak się tworzy funkcje oraz jak ich używać;
- potrafi napisać prosty program podzielony na funkcje realizujące pojedyncze zadania;
- potrafi obsługiwać kompilator w podstawowym zakresie;
- potrafi uruchomić program z konsoli systemu Linux.

Czynności uczniów	Działania nauczyciela	Materiał
Piszą kod.	Wprowadza pojęcia, wspiera uczniów, koryguje błędy, naprowadza, motywuje.	• Pojęcia: *funkcje;* • Edytor tekstu.
Kompilują kod.	Pomaga w zrozumieniu błędów kompilatora.	Kompilator języka C np. gcc
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C2.3**

Piotr Fiorek

Temat: Struktury i unie

Nazwa implementacji: Nauka języka C – struktury i unie

Opis implementacji: Nauka tworzenia oraz praktycznego korzystania ze struktur i unii

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu podstawowych pojęć programowania;
- kształtuje umiejętności poprawnego wykorzystywania struktur oraz unii;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:
- prosty edytor tekstu;
- kompilator języka C np. gcc.

Metody, działania:
- pisanie kodu w edytorze tekstu;
- kompilacja napisanego kodu przy użyciu kompilatora gcc;
- uruchamianie stworzonych programów i analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- potrafi wyjaśnić, czym są struktury, unie i dlaczego są ważne i potrzebne w programowaniu;
- wyjaśnia, jak się deklaruje struktury i unie oraz do czego się ich używa;
- potrafi napisać program korzystający ze struktur jako formy porządkowania danych w programach;
- potrafi obsługiwać kompilator w podstawowym zakresie;
- potrafi uruchomić program z konsoli systemu Linux.

Czynności uczniów	Działania nauczyciela	Materiał
Piszą kod.	Wprowadza pojęcia, wspiera uczniów, koryguje błędy, naprowadza, motywuje.	• Pojęcia: *struktury, unie*; • Edytor tekstu.
Kompilują kod.	Pomaga w zrozumieniu błędów kompilatora.	Kompilator języka C np. gcc
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C2.4**

Piotr Fiorek

Temat: Wskaźniki

Nazwa implementacji: Nauka języka C – wskaźniki

Opis implementacji: Poznanie zasad funkcjonowania pamięci w programach oraz wskaźników jako metody bezpośredniego dostępu do niej.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu podstawowych pojęć programowania;
- kształtuje umiejętności deklarowania wskaźników i uzyskiwania dostępu do pamięci oraz modyfikowania jej;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:

- prosty edytor tekstu;
- kompilator języka C np. gcc.

Metody, działania:

- pisanie kodu w edytorze tekstu;
- kompilacja napisanego kodu przy użyciu kompilatora gcc;
- uruchamianie stworzonych programów i analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- wyjaśnia, jak działa pamięć, jak dane są w niej umieszczane oraz jak można się do nich dostać, używając wskaźników;
- potrafi omówić, jak zarządzana jest pamięć oraz jak zmienne są w niej umieszczane;
- wyjaśnia, jak się deklaruje i na różne sposoby używa wskaźników;
- potrafi deklarować wskaźniki;
- potrafi obsługiwać kompilator w podstawowym zakresie;
- potrafi uruchomić program z konsoli systemu Linux.

Czynności uczniów	Działania nauczyciela	Materiał
Piszą kod.	Wprowadza pojęcia, wspiera uczniów, koryguje błędy, naprowadza, motywuje.	• Pojęcia: *pamięć, wskaźniki*; • Edytor tekstu.
Kompilują kod.	Pomaga w zrozumieniu błędów kompilatora.	Kompilator języka C np. gcc
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C3.1**

Krzysztof Bytow

Temat: Środowisko mikrokontrolera

Nazwa implementacji: Budowa układów i programowanie modułu

Opis implementacji: Zastosowanie modułu-interfejsu Arduino oraz obsługa interaktywnego terminala Arduino IDE, służącego do programowania mikrokontrolera. Prezentacja i wyjaśnienie sposobu zestawiania połączeń na podstawie dokumentacji ilustrującej montaż układów ćwiczeniowych. Podłączenie i sterowanie diodą LED z wykorzystaniem potencjometru oraz z wykorzystaniem wyjścia PWM. Odczyt wartości z wejścia analogowego. Mechaniczne – z wykorzystaniem potencjometru - sterowanie jasnością diody i programowe z wykorzystaniem wyjścia PWM. Zaimplementowanie kodu do sterowania diodą.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- doskonali umiejętność łączenia, konfiguracji i programowego sterowania diodami;
- kształtuje umiejętność obsługi terminala do pisania kodu sterującego;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów.

Materiał nauczania-uczenia się:
- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- dioda elektroluminescencyjna; button;
- 2 rezystory 220 Ω; potencjometr 10kΩ.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- trafnie objaśnia pojęcia: *mikrokontroler; dioda elektroluminescencyjna; button; opornik; potencjometr, modulacja szerokości impulsu - PWM;*
- poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino;
- potrafi przesłać wyniki z układu do komputera;
- poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
- potrafi podłączyć diodę elektroluminescencyjną oraz sterować jasnością diody;
- steruje diodą elektroluminescencyjną oraz modyfikuje treść wyświetlanych komunikatów.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Przeprowadza pokaz działania robota MAOR opartego na układzie Atmega, dodatkowo prezentacja wykorzystania układów Arduino w praktyce.	• Pojęcia: mikrokontroler, button, opornik, potencjometr, wejście analogowe, wej./wyj. cyfrowe, dioda elektroluminescencyjna, modulacja szerokości impulsu; http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Mikrokontroler http://arduino.cc/en/Tutorial/Button http://arduino.cc/en/Tutorial/DigitalPins http://arduino.cc/en/Tutorial/AnalogInput http://pl.wikipedia.org/wiki/Opornik http://pl.wikipedia.org/wiki/Potencjometr wprowadzenie do środowiska Arduino; http://e-swoi.pl/wiki/article/arduino-podstawy/ http://e-swoi.pl/wiki/article/mechatronika-faq/ • Filmy instruktażowe.
Uczestniczą w pogadance.	Prezentuje układ Arduino, na którym będą prowadzone ćwiczenia. Omawia elementy wchodzące w skład zestawu. Uruchamia środowisko programistyczne Arduino IDE, objaśniając poszczególne funkcje programu. Pokazuje wstępną konfigurację programu w celu komunikacji między komputerem, a modułem. Omawia kod źródłowy i jego poszczególne elementy. W dalszej części wyjaśnia i prezentuje podłączenie diody led pod wyjście PWM.	
Montują przykładowe układy sterowania diodami. Wprowadzają kod sterujący i testują działanie układu: modyfikowanie fragmentów kodu i obserwowanie skutków zmian.	Zachęca uczennice i uczniów do samodzielnego podłączenia układu i do zaprogramowania mikrokontrolera przykładową procedurą obsługi diody. Podpowiada, jakie szczegółowe działania muszą podjąć uczennice i uczniowie, aby ich układ funkcjonował prawidłowo, w pełni zgodnie z zadaniem.	• Filmy instruktażowe; • Tutoriale: http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Fotorezystor http://pl.wikipedia.org/wiki/Opornik http://arduino.cc/en/Tutorial/Button
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	
	Formułuje zadania rozszerzające: • Zmodyfikuj program tak, aby dioda mrugała w odstępach losowych; • Rozbudować układ o dodatkowy button, pierwszy button zapala diodę, drugi ją gasi (w przypadku niewduszenia przycisku gaszącego diodę, zaimplementować funkcję, która gasi ją po czasie 60 sekund).	http://arduino.cc/de/Reference/Random

Konspekt-scenariusz **Modułu C3.2**

Krzysztof Bytow

Temat: Tranzystor NPN

Nazwa implementacji: Zwiększenie możliwości wyjścia cyfrowego.

Opis implementacji: Zastosowanie tranzystora do sterowania pięcioma diodami LED połączonymi równolegle, które łącznie potrzebują do pracy więcej niż 40mA, jakie może zapewnić wyjście w module-interfejsie.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- zna rodzaje tranzystorów, ich parametry, budowę i oznaczenia wyprowadzeń;
- kształtuje umiejętność dokonywania pomiarów wartości napięcia i prądu;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny;
- kształtuje umiejętność szacowania prądu pobieranego przez elementy wyświetlające.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- diody elektroluminescencyjne – 5 szt. ;
- button; 5 rezystorów 220 Ω.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- trafnie objaśnia pojęcia: *tranzystor NPN, tranzystor PNP, button, modulacja szerokości impulsu;*
- wymienia rodzaje tranzystorów, podaje parametry i rozpoznaje oznaczenia wyprowadzeń;
- potrafi wykorzystać tranzystor jako przełącznik;
- prawidłowo wykonuje pomiary i przelicza wartości napięcia, prądu i oporności; trafnie używa słowa: *klucz tranzystorowy, stan nasycenia i zatkania, Volt, Amper.*

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Przedstawia budowę, rodzaje tranzystorów, zasadę działania. Prezentuje zasadę polaryzacji tranzystora NPN jako przełącznik. Prezentuje złożony układ Arduino z zaimplementowanym programem. Omawia zasadę działania tranzystora oraz opisuje wyprowadzenia. Razem z grupą oszacowuje prąd płynący przez diody, napięcie na diodach LED podczas świecenia, napięcie na tranzystorze w stanie aktywnym.	• Budowa tranzystora (oznaczenie wyprowadzeń - instrukcja); • Objaśnienie tego, czym są: tranzystor NPN, PNP; http://pl.wikipedia.org/wiki/Tranzystor; • Filmy instruktażowe • Wprowadzenie do środowiska Arduino; http://e-swoi.pl/wiki/article/arduino-podstawy/ http://e-swoi.pl/wiki/article/mechatronika-faq/
Montują układ z tranzystorem NPN jako przełącznikiem – obserwacja i analiza zasady działania. Montują układ sterowania z większą liczbą diod elektroluminescencyjnych podłączonych pod jedno wyjście cyfrowe. Wprowadzają kod sterujący i testują działanie układu: modyfikowanie fragmentów kodu i obserwowanie skutków zmian.	Zachęca uczennice i uczniów do samodzielnego montażu i oprogramowania układu.	
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	
	Formułuje zadania rozszerzające: • Zmodyfikuj kod i schemat połączeń, zastępując wyjście cyfrowe wyjściem PWM, które pozwala sterować szerokością modulacji impulsu diody LED. Współczynnik wypełnienia impulsów zmieniaj poprzez regulację wartości potencjometru 10kΩ, podłączonego do wejścia analogowego.	http://pl.wikipedia.org/wiki/Potencjometr http://arduino.cc/en/Tutorial/AnalogInput http://arduino.cc/en/Tutorial/PWM

Konspekt-scenariusz **Modułu C3.3**

Krzysztof Bytow

Temat: Pole magnetyczne

Nazwa implementacji: Budowa, działanie i zastosowanie hallotronu.

Opis implementacji: Wykorzystanie efektu Halla do sygnalizacji przy pomocy diody RGB występowania pola magnetycznego. Zastosowanie, budowa i działanie hallotronu.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- zna zasadę działania hallotronu, diody RGB i kształtuje umiejętność jej stosowania w praktyce;
- zna budowę i oznaczenia wyprowadzeń: hallotronu; diody RGB;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- hallotron CS3144E, buzzer;
- 4 rezystory 220 Ω;
- dioda RGB, 3 diody elektroluminescencyjne.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- trafnie objaśnia pojęcia: *pole magnetyczne*, *histereza*, *półprzewodnik*, *dioda RGB*, *efekt Halla;*
- objaśnia zasadę działania hallotronu i prawidłowo stosuje go w układach;
- podaje parametry i rozpoznaje oznaczenia wyprowadzeń diody RGB, hallotronu;
- prawidłowo rozbudowuje układy o dodatkowe, niezbędne do pracy elementy.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Prezentuje złożony układ Arduino z zaimplementowanym programem. Omawia zasadę działania hallotronu oraz opisuje wyprowadzenia. Zwraca uwagę na sposób podłączenia diody RGB i omawia zasadę działania. Omawia kod źródłowy i jego poszczególne elementy. Zachęca uczennice/uczniów do samodzielnego montażu i oprogramowania układu.	• Zasada działania hallotronu i diody RGB http://pl.wikipedia.org/wiki/Hallotron http://pl.wikipedia.org/wiki/RGB • Pojęcia: pole magnetyczne, histereza, półprzewodnik, dioda RGB, efekt Halla.
Montują przykładowy układ sterowania diodą RGB z wykorzystaniem hallotronu.	Zachęca uczennice i uczniów do samodzielnego montażu układu i do zaprogramowania mikrokontrolera przykładową procedurą obsługi halotronu.	
Wprowadzają kod sterujący i testują działanie układu: modyfikowanie fragmentów kodu i obserwowanie skutków zmian. Montują przykładowy układ sterowania buzzerem.	W dalszej części wyjaśnia i prezentuje podłączenie buzzera. Podpowiada, jakie szczegółowe działania muszą podjąć uczennice i uczniowie, aby ich układ funkcjonował prawidłowo, w pełni zgodnie z zadaniem.	
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	
	Zadania rozszerzające: • Rozbudować układ i kod implementacji hallotronu o sygnalizację dźwiękową z wykorzystaniem buzzera (po wykryciu pola magnetycznego generowany jest przerywany sygnał alarmowy).	

Konspekt-scenariusz **Modułu C3.4**

Krzysztof Bytow

Temat: Zegar

Nazwa implementacji: Pomiar czasu i obsługa wyświetlacza LCD 2x16

Opis implementacji: Stworzenie prostego stopera i zegara, z użyciem modułu-interfejsu. Wykorzystanie podstawowych funkcji do sterowania i prezentacji czasu na wyświetlaczu LCD, jak i ekranie monitora. Zasada działania i używania bibliotek.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- posiada wiedzę z zakresu obsługi bibliotek;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- potencjometr 10 kΩ; wyświetlacz LCD;
- opcjonalnie: dioda LED; buzzer; 2 szt. rezystorów 220 Ω; buton.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- prawidłowo buduje i oprogramowuje moduł-interfejs wskazujący aktualny czas;
- prawidłowo buduje i oprogramowuje moduł-interfejs służący jako stoper;
- uruchamia ukazywanie teksu na wyświetlaczu LCD;
- modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy;
- trafnie używa sformułowań: czas, sekunda, milisekunda, opóźnienie, pętle, LCD, biblioteka.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w prezentacji i pokazie, zadają pytania, wyjaśniają wątpliwości.	Zachęca uczennice i uczniów do zastanowienia się nad metodami pomiaru czasu, jak i jego prezentacji, jak działa zegarek cyfrowy, a jak zegar w systemie operacyjnym, co daje synchronizacja.	
Biorą udział w dyskusji.	Prowadzi dyskusję dotyczącą sposobów pomiaru czasu.	• Stoper, zegarek.
Montują układ z wyświetlaczem LCD, wprowadzają kod sterujący i testują działanie układu: modyfikowanie fragmentów kodu i obserwowanie skutków zmian.	Zachęca uczennice i uczniów do samodzielnego podłączenia układu i zaprogramowania przykładowym programem. Omawia kod źródłowy i jego poszczególne elementy. Prezentuje podłączenie wyświetlacza LCD oraz konfigurację w kodzie źródłowym. W dalszej części wyjaśnia i prezentuje działanie zegara i stopera.	• Pojęcia: mikrokontroler, LCD, potencjometr, wej./wyj. cyfrowe, biblioteka; http://pl.wikipedia.org/wiki/Mikrokontroler http://arduino.cc/en/Tutorial/DigitalPins http://pl.wikipedia.org/wiki/Potencjometr http://e-swoi.pl/wiki/article/mechatronika-faq/ http://pdf1.alldatasheet.com/datasheet-pdf/view/63663/HITACHI/HD44780U.html • Filmy instruktażowe.
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	
	Formułuje zadania rozszerzające: • Zmodyfikować kod sterujący wykorzystujący dodatkowe elementy w celu uruchomienia budzika (przycisk odpowiedzialny za włączenie lub wyłączenie alarmu, buzzer sygnał dźwiękowy uruchamiany o zadanej godzinie zaprogramowanej w kodzie sterującym).	

Moduły C
-Python-

Zajęcia dla szkół sprofilowanych infotechnicznie

Środowisko i zagadnienia

C1

- język Python: zmienne, wyrażenia, pętle, dane

C2

- język Python: pętle, funkcje, wyjątki, klasy

C3

- Arduino: omomierz, sterowanie, termometr

Konspekt-scenariusz **Modułu C1.1**

Piotr Fiorek

Temat: Zmienne, ich typy i dane

Nazwa implementacji: Nauka języka Python – zmienne

Opis implementacji: Poznanie pojęcia zmiennych ich typów oraz sposoby wczytywania danych z klawiatury.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń...
- posiada wiedzę z zakresu podstawowych pojęć programowania;
- kształtuje umiejętność wczytywania danych do zmiennych;
- potrafi efektywnie używać interpretera dla ułatwienia własnej pracy w języku Python;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:
- system Linux;
- interpreter języka Python;
- prosty edytor tekstu.

Metody, działania:
- interaktywne testowanie kodu w interpreterze;
- pisanie kodu w edytorze tekstu;
- uruchamianie kodu zapisanego w pliku za pomocą interpretera;
- analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- omawia podstawowe pojęcia programowania: zmienna, interpreter;
- potrafi napisać prosty program zawierający kilka zmiennych oraz pobierać ich wartości od użytkownika;
- potrafi wykorzystywać interpreter do interaktywnego testowania kawałków kodu przed włączeniem ich do programu;
- potrafi uruchomić program z konsoli systemu Linux przy użyciu interpretera języka Python.

Czynności uczniów	Działania nauczyciela	Materiał
Testują fragmenty kodu w interpreterze.	Wprowadza pojęcia, pomaga oraz tłumaczy ewentualnie pojawiające się błędy.	• Pojęcia: zmienna, interpreter; • Interpreter języka Python.
Piszą kod.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Edytor tekstu
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C1.2**

Piotr Fiorek

Temat: Wyrażenia warunkowe

Nazwa implementacji: Nauka języka Python – Wyrażenia *if – else*

Opis implementacji: Poznanie wyrażeń warunkowych *if – elif – else*.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu podstawowych pojęć programowania i jest świadomy ich znaczenia;
- kształtuje umiejętność poprawnego używania wyrażeń warunkowych;
- rozpoznaje strukturę wyrażeń warunkowych;
- kształtuje umiejętność pisania programów;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:
- system Linux;
- interpreter języka Python;
- prosty edytor tekstu.

Metody, działania:
- interaktywne testowanie kodu w interpreterze;
- pisanie kodu w edytorze tekstu;
- uruchamianie kodu zapisanego w pliku za pomocą interpretera;
- analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- omawia podstawowe pojęcia programowania: wyrażenia warunkowe, struktura wyrażeń warunkowych, potrafi omówić ich istotność w programowaniu;
- potrafi pisać programy, które proszą użytkownika o wprowadzenie danych oraz podejmują odpowiednią akcję zależnie od nich;
- potrafi uruchomić program z konsoli systemu Linux przy użyciu interpretera języka Python.

Czynności uczniów	Działania nauczyciela	Materiał
Testują fragmenty kodu w interpreterze.	Wprowadza pojęcia, pomaga oraz tłumaczy ewentualnie pojawiające się błędy.	• Pojęcia: wyrażenie warunkowe, struktura wyrażenia warunkowego; • Interpreter języka Python.
Piszą kod.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Edytor tekstu.
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C1.3**

Piotr Fiorek

Temat: Pętle - *while*

Nazwa implementacji: Nauka języka Python – Pętle – *while*

Opis implementacji: Poznanie pętli *while()*

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu podstawowych pojęć programowania;
- kształtuje umiejętność wykorzystania pętli-*while*;
- kształtuje umiejętność pisania programów, które powtarzają czynności przy użyciu pętli-*while*;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:

- interpreter języka Python;
- prosty edytor tekstu.

Metody, działania:

- interaktywne testowanie kodu w interpreterze;
- pisanie kodu w edytorze tekstu;
- uruchamianie kodu zapisanego w pliku za pomocą interpretera;
- analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- omawia podstawowe pojęcia programowania: *pętla-while*;
- potrafi pisać programy, które powtarzają pewne czynności określoną ilość razy, sterowane pętlą;
- potrafi uruchomić program z konsoli systemu Linux przy użyciu interpretera języka Python.

Czynności uczniów	Działania nauczyciela	Materiał
Testują fragmenty kodu w interpreterze.	Wprowadza pojęcia, pomaga oraz tłumaczy ewentualnie pojawiające się błędy.	• Pojęcia: *pętla-while*; • Interpreter języka Python.
Piszą kod.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Edytor tekstu.
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C1.4**

Piotr Fiorek

Temat: Zaawansowane typy danych

Nazwa implementacji: Nauka języka Python – zaawansowane typy danych

Opis implementacji: Poznanie zaawansowanych typów danych oraz nauczenie się, do czego i jak je wykorzystywać.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu podstawowych pojęć programowania;
- kształtuje umiejętność wykorzystywania zaawansowanych typów danych w programach oraz wskazuje metody ich obsługi;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:
- interpreter języka Python;
- prosty edytor tekstu.

Metody, działania:
- interaktywne testowanie kodu w interpreterze;
- pisanie kodu w edytorze tekstu;
- uruchamianie kodu zapisanego w pliku za pomocą interpretera;
- analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- omawia zaawansowane typy danych i ich przydatność w programowaniu;
- potrafi pisać programy, które korzystają z list, krotek i słowników oraz wie, czym są zbiory i jak ich używać;
- potrafi uruchomić program z konsoli systemu Linux przy użyciu interpretera języka Python.

Czynności uczniów	Działania nauczyciela	Materiał
Testują fragmenty kodu w interpreterze.	Wprowadza pojęcia, pomaga oraz tłumaczy ewentualnie pojawiające się błędy.	• Zaawansowane typy danych; • Interpreter języka Python.
Piszą kod.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Edytor tekstu.
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C2.1**

Piotr Fiorek

Temat: Tablice i pętle-*for*

Nazwa implementacji: Tablice i pętle-*for*

Opis implementacji: Poznanie innego rodzaju pętli, jaką jest pętla *for* w języku Python.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu podstawowych pojęć programowania;
- potrafi wskazać różnice pomiędzy dwoma poznanymi pętlami i ich wykorzystania;
- kształtuje umiejętność efektywnego wykorzystać pętlę *for* podczas programowania;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:
- interpreter języka Python;
- prosty edytor tekstu.

Metody, działania:
- interaktywne testowanie kodu w interpreterze;
- pisanie kodu w edytorze tekstu;
- uruchamianie kodu zapisanego w pliku za pomocą interpretera;
- analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- omawia podstawowe pojęcia programowania: pętla-*for*, różnice pomiędzy pętlami;
- potrafi pisać programy, które używają pętli *for* do operacji na poszczególnych elementach większych zbiorów;
- potrafi uruchomić program z konsoli systemu Linux przy użyciu interpretera języka Python.

Czynności uczniów	Działania nauczyciela	Materiał
Testują fragmenty kodu w interpreterze.	Wprowadza pojęcia, pomaga oraz tłumaczy ewentualnie pojawiające się błędy.	• Pojęcia: pętla-for, różnice pomiędzy pętlą-while i pętlą-for; • Interpreter języka Python.
Piszą kod.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Edytor tekstu.
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C2.2**

Piotr Fiorek

Temat: Funkcje

Nazwa implementacji: Nauka języka Python – Funkcje

Opis implementacji: Poznanie jak w Pythonie tworzyć własne funkcje oraz kiedy należy program dzielić na funkcje.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń...
- posiada wiedzę z zakresu podstawowych pojęć programowania;
- kształtuje umiejętność tworzenia i używania funkcji;
- kształtuje umiejętność efektywnego wykorzystywania funkcji w programowaniu;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:
- interpreter języka Python;
- prosty edytor tekstu.

Metody, działania:
- interaktywne testowanie kodu w interpreterze;
- pisanie kodu w edytorze tekstu;
- uruchamianie kodu zapisanego w pliku za pomocą interpretera;
- analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- omawia podstawowe pojęcia programowania: funkcje;
- potrafi pisać własne funkcje;
- potrafi dzielić długie programy na funkcje, aby były czytelniejsze i efektywniejsze;
- potrafi uruchomić program z konsoli systemu Linux przy użyciu interpretera języka Python.

Czynności uczniów	Działania nauczyciela	Materiał
Testują fragmenty kodu w interpreterze.	Wprowadza pojęcia, pomaga oraz tłumaczy ewentualnie pojawiające się błędy.	• Pojęcia: funkcje; • Interpreter języka Python.
Piszą kod.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Edytor tekstu.
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C2.3**

Piotr Fiorek

Temat: Wyjątki

Nazwa implementacji: Nauka języka Python – wyjątki

Opis implementacji: Poznanie czym są wyjątki i jak ich używać.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń…
- potrafi korzystać z wiedzy dotyczącej wyjątków w pracy w środowisku programistycznym;
- kształtuje umiejętność pisania odporniejszych na błędy programów;
- odczuwa satysfakcję z prawidłowo wykonanego zadania;
- kształtuje cierpliwość w pracy w środowisku programistycznym.

Materiał nauczania-uczenia się:
- interpreter języka Python;
- prosty edytor tekstu.

Metody, działania:
- interaktywne testowanie kodu w interpreterze;
- pisanie kodu w edytorze tekstu;
- uruchamianie kodu zapisanego w pliku za pomocą interpretera;
- analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- potrafi omówić, czym są wyjątki i jak pomagają programiście;
- potrafi przewidzieć, kiedy program może wyrzucić wyjątek;
- potrafi wyłapać oraz obsłużyć wyjątek w programie;
- potrafi uruchomić program z konsoli systemu Linux przy użyciu interpretera języka Python.

Czynności uczniów	Działania nauczyciela	Materiał
Testują fragmenty kodu w interpreterze.	Wprowadza pojęcia, pomaga oraz tłumaczy ewentualnie pojawiające się błędy.	• Obsługa wyjątków • Iterpreter języka Python.
Piszą kod.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Edytor tekstu.
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C2.4**

Piotr Fiorek

Temat: Klasy

Nazwa implementacji: Nauka języka Python – Klasy

Opis implementacji: Poznanie czym są klasy i jak dzięki nim tworzyć efektywniejsze programy.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- wdrażanie do pracy w środowisku programistycznym;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- kształtowanie nawyków związanych z pracą w środowisku programistycznym;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu podstawowych pojęć programowania i wskazuje zasadność ich wykorzystania;
- kształtuje umiejętność tworzenia własnych klas oraz korzystania z gotowych klas biblioteki standardowej Pythona;
- kształtuje umiejętność pisania programów obiektowych;
- zna najczęściej używane elementy biblioteki standardowej oraz potrafi w dokumentacji znaleźć informacje na temat pozostałych;
- odczuwa satysfakcję z prawidłowo wykonanego zadania.

Materiał nauczania-uczenia się:

- interpreter języka Python;
- prosty edytor tekstu.

Metody, działania:

- interaktywne testowanie kodu w interpreterze;
- pisanie kodu w edytorze tekstu;
- uruchamianie kodu zapisanego w pliku za pomocą interpretera;
- analiza ewentualnych błędów.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- omawia podstawowe pojęcia programowania: klasy, programowanie obiektowe;
- potrafi uzasadnić używanie klas dla ułatwienia pisania dużych programów;
- rozumie paradygmat programowania obiektowego oraz potrafi pisać programy korzystające z niego;
- potrafi pisać własne klasy i korzystać z gotowych bibliotek Pythona;
- potrafi uruchomić program z konsoli systemu Linux przy użyciu interpretera języka Python.

Czynności uczniów	Działania nauczyciela	Materiał
Testują fragmenty kodu w interpreterze.	Wprowadza pojęcia, pomaga oraz tłumaczy ewentualnie pojawiające się błędy.	• Pojęcia: klasa, programowanie obiektowe; • Interpreter języka Python.
Piszą kod.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Edytor tekstu.
Uruchamiają powstały program.	Pomaga przy wyjaśnianiu niepożądanych zachowań programu.	

Konspekt-scenariusz **Modułu C3.1**

Krzysztof Bytow

Temat: Wykorzystanie wejścia analogowego

Nazwa implementacji: Wprowadzenie do środowiska mikrokontrolera.

Opis implementacji: Zastosowanie modułu-interfejsu Arduino oraz obsługa interaktywnego terminala Arduino IDE, służącego do programowania mikrokontrolera. Prezentacja i wyjaśnienie sposobu zestawiania połączeń na podstawie dokumentacji ilustrującej montaż układów ćwiczeniowych. Podłączenie i sterowanie diodą led z wykorzystaniem czujnika nachylenia i buttona.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń…

- doskonali umiejętność łączenia, konfiguracji i programowego sterowania diodami;
- kształtuje umiejętność obsługi terminala do pisania kodu sterującego;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- dioda elektroluminescencyjna; button; czujnik nachylenia;
- rezystor 220 Ω.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- trafnie objaśnia pojęcia: *mikrokontroler; dioda elektroluminescencyjna; button; opornik; czujnik wychylenia*;
- poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino;
- poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
- potrafi podłączyć i sterować diodą elektroluminescencyjną.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Przeprowadza pokaz działania robota MAOR opartego na układzie Atmega, dodatkowo prezentacja wykorzystania układów Arduino w praktyce.	• Pojęcia: mikrokontroler, button, opornik, czujnik wychylenia, wej./wyj. cyfrowe, dioda elektroluminescencyjna; http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Mikrokontroler http://arduino.cc/en/Tutorial/Button http://arduino.cc/en/Tutorial/DigitalPins http://pl.wikipedia.org/wiki/Opornik • Wprowadzenie do środowiska Arduino; http://e-swoi.pl/wiki/article/arduino-podstawy/ http://e-swoi.pl/wiki/article/mechatronika-faq/ • Filmy instruktażowe.
	Prezentuje układ Arduino, na którym będą prowadzone ćwiczenia. Omawia elementy wchodzące w skład zestawu. Uruchamia środowisko programistyczne Arduino IDE, objaśniając poszczególne funkcje programu. Pokazuje wstępną konfigurację programu w celu komunikacji między komputerem, a modułem. Omawia kod źródłowy i jego poszczególne elementy.	
Montują przykładowe układy sterowania diodą z wykorzystaniem czujnika wychylenia, buttonu.	Zachęca uczennice i uczniów do samodzielnego podłączenia układu i do zaprogramowania mikrokontrolera przykładową procedurą obsługi diody.	
Wprowadzają kod sterujący i testują działanie układu; modyfikują fragmenty kodu i obserwują skutki zmian.	W dalszej części wyjaśnia i prezentuje podłączenie diody led. Podpowiada, jakie szczegółowe działania muszą podjąć uczennice i uczniowie, aby ich układ funkcjonował prawidłowo, w pełni zgodnie z zadaniem.	
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące;	
	Formułuje zadania rozszerzające: • Zmodyfikuj program tak, aby dioda świeciła się w zależności od stanu, w jakim jest button (HIGH lub LOW); • zmodyfikować kod i schemat – wykorzystać wyjście PWM, aby sterować jasnością diody.	

Konspekt-scenariusz **Modułu C3.2**

Krzysztof Bytow

Temat: Pomiar wartości opornika

Nazwa implementacji: Budowa prostego omomierza

Opis implementacji: Zastosowanie modułu-interfejsu Arduino jako narzędzia do pomocy w oszacowaniu wartości oporników. Budowa i sposoby rozpoznawania oporników. Oszacowanie wartości rezystorów połączonych szeregowo i równolegle.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń…

- poszerza wiedzę w zakresie sposobów odczytu wartości oporników;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- 1 rezystor 10 kΩ lub potencjometr 10kΩ;
- rezystor o dowolnej wartości.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- trafnie objaśnia pojęcia: dzielnik napięcia, opornik;
- poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino;

- poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino;
- potrafi przesłać wyniki z układu do komputera;
- poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
- potrafi odczytać wartość opornika na podstawie kodu barwnego lub z użyciem modułu-interfejsu;
- potrafi wyliczyć wartość rezystancji w przypadku łączenia rezystorów równolegle lub szeregowo;
- potrafi dokonać przekształceń wzoru prawa Ohma, aby wyznaczyć wartość rezystancji.
- potrafi podłączyć i sterować diodą elektroluminescencyjną.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Przeprowadza pokaz odczytu wartości na podstawie kodu barwnego. Prezentuje przykładowe stron www, z aplikacjami do przedstawienia kodu barwnego jako wartość opornika. Przeprowadza dyskusję lub prezentację, w jaki sposób można dokonać jeszcze pomiaru wartości rezystora.	• http://pl.wikipedia.org/wiki/Opornik • Objaśnienie tego, czym jest: dzielnik napięcia; przetwornik analogowo-cyfrowy; • Przypomnienie podstawowych zasad dotyczących: oporników, rezystancji, prawa Ohma; przekształceń wzorów.
	Prezentuje układ Arduino, na którym będą prowadzone pomiary wartości oporników. Zwraca uwagę, iż odczyt może odbiegać od wartości rezystora – dyskusja co ma na to wpływ.	• Zalecenie posiadania miernika uniwersalnego, który ułatwi zobrazowanie.
Montują przykładowy układ do odczytu wartości opornika.	Zachęca uczennice i uczniów do samodzielnego podłączenia układu i do zaprogramowania mikrokontrolera przykładową procedurą odczytu wartości opornika.	• Tabela oznaczeń barwnych oporników; • Opis wyprowadzeń potencjometru - instrukcja.
Wprowadzają kod sterujący i testują działanie układu, modyfikują fragmenty kodu i schematu – obserwują skutki zmian.	Omawia kod źródłowy i jego poszczególne elementy. W dalszej części przedstawia wzory na obliczenie rezystancji w połączeniu szeregowym i równoległym. Podpowiada, jak połączyć oporniki i odczytać wartość rezystancji.	
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące;	

	Formułuje zadania rozszerzające:
	• połączyć dwa lub więcej oporników w sposób:
	  • *szeregowy*
	  • *równoległy*
	• dokonać pomiaru.	

Konspekt-scenariusz **Modułu C3.3**

Krzysztof Bytow

Temat: Kontrola diody RGB

Nazwa implementacji: Sterowanie układem z interfejsu Arduino IDE.

Opis implementacji: Komunikacja modułu-interfejsu z komputerem PC na przykładzie sterowania jasnością i kolorami diody RGB.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu zasady działania diody RGB i kształtuje umiejętność jej zastosowania w praktyce;
- posiada wiedzę z zakresu budowy i oznaczenia wyprowadzeń: diody RGB;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- rozwija sprawność deklarowania podstawowych typów zmiennych, definiowania i przypisywania im wartości.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- 3 rezystory 220 Ω; 3 buttony;
- dioda RGB.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- trafnie objaśnia pojęcia: *półprzewodnik, dioda RGB, port szeregowy, komunikacja, transmisja danych*;
- podaje parametry i rozpoznaje oznaczenia wyprowadzeń diody RGB;
- prawidłowo rozbudowuje układy o dodatkowe, niezbędne do pracy elementy;
- stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs;
- poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Prezentuje złożony układ Arduino z zaimplementowanym programem. Omawia zasadę działania oraz zwraca uwagę na sposób podłączenia diody RGB. Zachęca uczennice i uczniów do samodzielnego montażu i oprogramowania układu.	• Pojęcia: mikrokontroler, button, opornik, wej./wyj. cyfrowe, dioda RGB, modulacja szerokości impulsu, tablica; http://pl.wikipedia.org/wiki/RGB http://arduino.cc/en/Tutorial/Button http://arduino.cc/en/Tutorial/DigitalPins http://pl.wikipedia.org/wiki/Opornik http://arduino.cc/en/Tutorial/PWM • Filmy instruktażowe.
Montują przykładowe układy sterowania diodą RGB. Wprowadzają kod sterujący i testują działanie układu: modyfikowanie fragmentów kodu i obserwowanie skutków zmian.	Omawia kod źródłowy i jego poszczególne elementy. W dalszej części wyjaśnia i prezentuje sterowanie diodą RGB z poziomu PC, wydając proste polecenia z klawiatury i przesyłając je przez terminal portu w Arduino IDE. Podpowiada, jakie szczegółowe działania muszą podjąć uczennice i uczniowie, aby ich układ funkcjonował prawidłowo, w pełni zgodnie z zadaniem.	
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące;	

Konspekt-scenariusz **Modułu C3.4**

Krzysztof Bytow

Temat: Prezentacja pomiarów temperatury na RGB.

Nazwa implementacji: Prezentacja pomiarów

Opis implementacji: Zastosowanie modułu-interfejsu Arduino do odczytu temperatury. Przełożenie odczytów wartości z wejścia analogowego na sterowanie jasnością i barwą diody RGB. Definiowanie wartości progowych temperatury i sygnalizowanie alarmem w przypadku ich przekroczenia.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń…

- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- kształtuje umiejętność przedstawiania skal i jednostek temperatury oraz zależności między nimi;
- wykorzystuje komputer aby dokonał potrzebnych przeliczeń do zmian skal temperatury;
- kształtuje umiejętność generowania prostych sygnałów dźwiękowych;
- kształtuje umiejętność sterowania jasnością i barwą świecenia diody RGB.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- dioda RGB; buzzer; czujnik MCP9700.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- zgodnie z zasadami działania podłącza czujnik pomiarowy termistor; buzzer, diodę RGB;
- prawidłowo buduje i oprogramowuje moduł-interfejs służący do pomiaru temperatury;
- uruchamia ukazywanie odczytów w środowisku Linux;
- modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy;
- dokonuje przeliczenia wartości pomiarów temperatury do innych skal;
- trafnie używa sformułowań: *czujnik, stopnie Celsjusza, stopnie Fahrenheita, Kelvin, czułość, wejście analogowe, przetwornik A/D.*

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Prezentuje złożony układ Arduino z zaimplementowanym programem do odczytu temperatury z czujnika. Omawia zasadę odczytu, zwraca uwagę na dokładność pomiaru, zachęca do zapoznania z dokumentacją czujnika. Porusza temat sposobów zwiększania dokładności, co ma na nią wpływ i jak można ją poprawić – prezentacja multimedialna.	• Pojęcia: czujnik, przetwornik A/D, dioda RGB, buzzer, biblioteka, opornik, modulacja szerokości impulsu; http://arduino.cc/en/Tutorial/PWM http://pl.wikipedia.org/wiki/Opornik http://pl.wikipedia.org/wiki/RGB http://pl.wikipedia.org/wiki/Przetwornik_analogowo-cyfrowy • Filmy instruktażowe; • Prezentacja multimedialna; • Filmy dostępne w serwisie http://www.youtube.com/ hasła kluczowe: arduino pomiar temperatury • Dokumentacja techniczna http://ww1.microchip.com/downloads/en/DeviceDoc/21942e.pdf • Przedstawienie zależności napięcia wyjściowego od temperatury.
Biorą udział w dyskusji.	Prowadzi dyskusję dotyczącą sposobów odczytu temperatury.	• Termometr cyfrowy, termometr rtęciowy lub multimetr z możliwością pomiaru temperatury.
Wykonują implementację modułu-interfejsu do pomiaru temperatury; Modyfikują lub rozbudowują (np. łączą) implementację o dodanie procedury uśredniającej pomiary.	Zachęca uczennice i uczniów do samodzielnego podłączenia układu i zaprogramowania przykładowym programem. Omawia kod źródłowy i jego poszczególne elementy. Prezentuje podłączenie diody RGB i buzzera oraz konfigurację w kodzie źródłowym. W dalszej części wyjaśnia i prezentuje przesyłanie wyników do komputera i sposoby na ich obróbkę.	• Zależność między temperaturą wyrażoną w stopniach Celsjusza t [°C] a wyrażoną w Kelwinach t [K]; • Zależność między temperaturą wyrażoną w stopniach Celsjusza t [°C] i Fahrenheita t [°F].
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	

Moduły C

-HTML-

Zajęcia dla szkół sprofilowanych infotechnicznie

Środowisko i zagadnienia

C1

HTML, CSS, JavaScript, formularz

C2

jQuery, lista, baza danych, Canvas

C3

Arduino: sterowanie, dalmierz, serwo

Konspekt-scenariusz **Modułu C1.1**

Daniel Mendalka, Michał Czyżewski

Temat: Podstawy HTML

Nazwa implementacji: Podstawy HTML

Opis implementacji: Zaczynając od pustego szablonu, stworzymy szkielet strony internetowej zawierającej nagłówek, stopkę, nawigację i treść.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- wdrażanie do tworzenia witryn i aplikacji internetowych z wykorzystaniem HTML;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z tworzeniem witryn i aplikacji internetowych;

b) szczegółowe: Uczennica/uczeń…

- zapoznaje się z podstawowym szablonem dla strony / aplikacji internetowej;
- poznaje podstawowe tagi HTML;
- kształtuje umiejętność weryfikowania przygotowanego dokumentu HTML;
- odczuwa satysfakcję z prawidłowego wykonania dokumentu HTML.

Materiał nauczania-uczenia się:

- podstawowy szablon HTML5;
- Firefox i Narzędzia dla twórców witryn;
- Tagi HTML: <*p*>, <*a*>, <*h1*>, <*h2*>, <*h3*>, <*h4*>, <*h5*>, <*h6*>, <*div*>, <*img*>, <*ul*>, <*ol*>, <*li*>;
- Tagi HTML5: <*header*>, <*footer*>, <*section*>, <*article*>, <*nav*>.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- objaśnia zastosowanie podstawowych tagów HTML;
- tworzy prosty dokument HTML.

Czynności uczniów	Działania nauczyciela	Materiał
Słuchają wprowadzenia do tematyki HTML.	Omawia typy plików i elementy składające się na podstawowy szablon HTML.	Szablon HTML5.
Biorą udział w prezentacji.	Prezentuje narzędzia dostępne w przeglądarce do analizy dokumentu HTML.	Firefox, Narzędzia dla twórców witryn.
Zapoznają się z treścią na temat tagów HTML.	Wyjaśnia wątpliwości dotyczące tagów.	Opis implementacji.
Wykonują praktyczne ćwiczenia.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Opis implementacji i pliki z implementacją.

Konspekt-scenariusz **Modułu C1.2**

Daniel Mendalka, Michał Czyżewski

Temat: CSS i box model

Nazwa implementacji: CSS i box model

Opis implementacji: Poznajemy podstawy CSS

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- wdrażanie do tworzenia witryn i aplikacji internetowych z wykorzystaniem HTML i CSS;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z tworzeniem witryn i aplikacji internetowych;

b) szczegółowe: Uczennica/uczeń…

- zapoznaje się z budową arkusza stylów;
- poznaje sposoby rozmieszczania elementów HTML na stronie za pomocą CSS;
- odczuwa satysfakcję z prawidłowego wykonania arkusza stylów CSS.

Materiał nauczania-uczenia się:

- właściwości CSS: *display*, *margin*, *padding*, *border*, *float*, *position*;
- CSS Box Model.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- rozmieszcza elementy HTML na stronie za pomocą CSS;
- dopasowuje wygląd dokumentu HTML korzystając z pliku CSS;
- swobodnie rozmieszcza elementy dokumentu na stronie za pomocą różnych technik;
- analizuje strukturę dokumentu korzystając z narzędzi dostępnych w przeglądarce.

Czynności uczniów	Działania nauczyciela	Materiał
Słuchają wprowadzenia do tematyki CSS.	Omawia elementy składające się na podstawowy arkusz CSS.	Opis implementacji.
Biorą udział w prezentacji.	Prezentuje narzędzia dostępne w przeglądarce do analizy dokumentu HTML i właściwości CSS.	Firefox, Narzędzia dla twórców witryn
Zapoznają się z treścią na temat właściwości CSS.	Wyjaśnia sens stosowania CSS.	Opis implementacji.
Wykonują praktyczne ćwiczenia.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Opis implementacji i pliki z implementacją.

Konspekt-scenariusz **Modułu C1.3**

Daniel Mendalka, Michał Czyżewski

Temat: Formularze

Nazwa implementacji: Formularze

Opis implementacji: Stworzymy dokument z formularzem umożliwiającym wprowadzanie informacji przez użytkownika, z wykorzystaniem odpowiednich elementów w zależności od rodzaju informacji.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- wdrażanie do tworzenia witryn i aplikacji internetowych z wykorzystaniem HTML;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z tworzeniem witryn i aplikacji internetowych;

b) szczegółowe: Uczennica/uczeń…

- poznaje nowe grupy tagów HTML służące do budowania formularzy;
- kształtuje umiejętność tworzenia formularzy w dokumencie HTML z wykorzystaniem elementów poprawnie dobranych do rodzaju danych wprowadzanych przez użytkownika;
- kształtuje umiejętność weryfikowania przygotowanego formularza;
- odczuwa satysfakcję z prawidłowego wykonania dokumentu HTML z formularzem.

Materiał nauczania-uczenia się:

- Tagi <*form*>, <*input*>, <*textarea*>, <*label*>, <*select*>, <*option*>;
- Atrybuty tagów: *name*, *value*, *type*, *placeholder*, *autofocus*, *required*, *title*.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- objaśnia zastosowanie tagów HTML służących do budowania formularzy;
- tworzy rozbudowany formularz HTML;

- dobiera odpowiednie elementy formularza do typu danych wprowadzanych przez użytkownika;
- weryfikuje poprawność stworzonego dokumentu HTML.

Czynności uczniów	Działania nauczyciela	Materiał
Zapoznają się z informacjami zawartymi w opisie implementacji.	Wyjaśnia istotę tworzenia formularzy.	Opis implementacji.
Biorą udział w prezentacji.	Krótko przypomina o narzędziach dla twórców witryn w przeglądarce oraz narzędziach do weryfikacji dokumentu HTML.	Firefox, http://html5.validator.nu
Wykonują ćwiczenia praktyczne.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Opis implementacji i pliki z implementacją.

Konspekt-scenariusz **Modułu C1.4**

Daniel Mendalka, Michał Czyżewski

Temat: JavaScript

Nazwa implementacji: JavaScript

Opis implementacji: Stworzymy aplikację w języku JavaScript.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- poznawanie języka JavaScript;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;

b) szczegółowe: Uczennica/uczeń…

- poznaje działanie aplikacji napisanych w języku JavaScript;
- kształtuje nawyk dobrych praktyk pisania kodu w JavaScript;
- dokonuje podstawowych operacji matematycznych oraz na ciągach znaków;
- kształtuje umiejętność tworzenia i modyfikowania podstawowych struktur danych w JavaScript.

Materiał nauczania-uczenia się:

- typy zmiennych i struktur danych w JavaScript;
- tworzenie funkcji i sposoby ich wywoływania;
- podstawowe sposoby na interakcję z użytkownikiem;
- obsługa konsoli JavaScript w przeglądarce.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- dokonuje operacji na zmiennych;
- tworzy proste struktury danych;
- definiuje, przekazuje i uruchamia funkcje;
- tworzy prostą aplikację dokonującą prostej interakcji z użytkownikiem.

[opcjonalnie] Uczeń zaawansowany...

- tworzy zamknięty moduł JavaScript;
- definiuje zmienne i funkcje dostępne tylko wewnątrz modułu;
- udostępnia wybrane zmienne i funkcje jako publiczne na zewnątrz modułu.

Czynności uczniów	Działania nauczyciela	Materiał
Biorą udział w prezentacji.	Prezentuje działania konsoli JavaScript w przeglądarce.	Firefox.
Zapoznają się z treściami zawartymi w opisie implementacji.	Wyjaśnia główne struktury języka.	Opis implementacji.
Wykonują ćwiczenia praktyczne.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Opis implementacji, konsola JavaScript

Konspekt-scenariusz **Modułu C2.1**

Daniel Mendalka

Temat: Podstawy jQuery

Nazwa implementacji: Podstawy jQuery

Opis implementacji: Wykorzystując wiedzę z lekcji o formularzach HTML i JavaScript stworzymy napisany program tak, żeby komunikował się z użytkownikiem poprzez formularz i dokument HTML.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- wdrażanie do tworzenia aplikacji internetowych z wykorzystaniem HTML i JavaScript;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z tworzeniem aplikacji internetowych;

b) szczegółowe: Uczennica/uczeń…

- kształtuje umiejętność tworzenia aplikacji komunikującej się z użytkownikiem poprzez dokument HTML;
- poznaje podstawowe funkcje dostępne w jQuery;
- kształtuje umiejętność integrowania aplikacji JavaScript wykorzystująca jQuery z dokumentem HTML.

Materiał nauczania-uczenia się:

- uruchomienie aplikacji po pełnym załadowaniu strony;
- wyszukiwanie elementów HTML poprzez selectory CSS;
- pobieranie wartości z formularza HTML;
- modyfikowanie dokumentu HTML.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- wyszukuje dowolne elementy HTML z wykorzystaniem selektorów CSS;

- pobiera informacje wprowadzone do formularza HTML;
- modyfikuje dokument HTML poprzez zmianę i dodaje nowe elementy.

[opcjonalnie] Uczeń zaawansowany...

- potrafi zastosować dobre praktyki tworzenia aplikacji z jQuery, optymalizując szybkość działania aplikacji.

Czynności uczniów	Działania nauczyciela	Materiał
Zapoznają się z treściami zawartymi w opisie implementacji.	Wyjaśnia wątpliwości dotyczące treści zadań.	Opis implementacji.
Wykonują ćwiczenia praktyczne.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Opis implementacji, pliki z implementacją, http://jquery.com/

Konspekt-scenariusz **Modułu C2.2**

Daniel Mendalka

Temat: jQuery – zdarzenia i efekty

Nazwa implementacji: jQuery – zdarzenia i efekty

Opis implementacji: Do już znanych podstaw jQuery wprowadzimy dodatkowe informacje o tym jak reagować na różne operacje dokonywane przez użytkownika oraz prezentować wynik działań poprzez zmiany w dokumencie HTML, używając odpowiednich efektów. Na koniec całość pozwoli nam stworzyć zalążek aplikacji ToDo, która będzie rozwijana przy kolejnych lekcjach.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- wdrażanie do tworzenia aplikacji internetowych z wykorzystaniem HTML i JavaScript;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z tworzeniem aplikacji internetowych;

b) szczegółowe: Uczennica/uczeń…

- rozszerza wiedzę na temat biblioteki jQuery i możliwości tworzenia użytecznych aplikacji internetowych;
- kształtuje umiejętność tworzenia aplikacji reagującej na różne akcje użytkownika;
- kształtuje umiejętność modyfikowania dokumentu HTML z wykorzystaniem efektów jQuery.

Materiał nauczania-uczenia się:

- obsługa zdarzeń generowanych przez formularz, myszkę i klawiaturę;
- delegowanie obsługi zdarzenia na nowo dodanych elementach;
- pokazywanie i ukrywanie elementów z efektami podkreślającymi zmianę.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- dodaje do aplikacji obsługę zdarzeń;

- tworzy obsługę zdarzenia dla elementów tworzonych dynamicznie przez aplikację;
- modyfikuje dokument HTML dokonując zmian z drobnymi animacjami;
- łączy animacje w sekwencje następujące jedna po drugiej;
- korzysta z efektów jQuery do modyfikowania dokumentu HTML.

[opcjonalnie] Uczeń zaawansowany...

- potrafi dołączyć rozszerzenie do biblioteki jQuery w postaci jQuery UI;
- potrafi wykorzystać bardziej zaawansowane efekty animacji dostępne w jQuery UI;
- potrafi dodać bardziej zaawansowane metody interakcji użytkownika z aplikacją.

Czynności uczniów	Działania nauczyciela	Materiał
Zapoznają się z treściami zawartymi w opisie implementacji.	Wyjaśnia wątpliwości dotyczące treści zadań.	Opis implementacji.
Wykonują ćwiczenia praktyczne.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Opis implementacji, pliki z implementacją, http://jquery.com/

Konspekt-scenariusz **Modułu C2.3**

Daniel Mendalka

Temat: Przechowywanie danych

Nazwa implementacji: Przechowywanie danych

Opis implementacji: Aplikację ToDo uzupełnimy o możliwość trwałego zapisania wprowadzonych informacji i ich odczytu przy ponownym uruchomieniu strony.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- wdrażanie do tworzenia aplikacji internetowych z wykorzystaniem HTML i JavaScript;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z tworzeniem aplikacji internetowych;

b) szczegółowe: Uczennica/uczeń…

- zapoznaje się ze sposobami trwałego zapisu danych w przeglądarce;
- kształtuje umiejętność korzystania z kilku różnych sposobów zapisu danych;
- jest świadomy zalet i wad każdego ze sposobu;
- kształtuje nawyk świadomego dobierania narzędzia w zależności od typu informacji do przechowania.

Materiał nauczania-uczenia się:

- pobieranie i zapisywanie danych do *sessionStorage* i *localStorage*.
- tworzenie i zarządzanie bazą danych *indexDB*.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…

- wymienia sposoby na trwały zapis danych oraz omawia ich wady, oraz zalety;
- zapisuje informacje na czas działania sesji przeglądarki;
- trwale zapisuje informacje, niezależnie od czasu działania sesji;
- pobiera i modyfikuje całość lub część przechowywanych danych;
- dobiera najlepsze narzędzie w zależności od typu informacji do przechowania.

- tworzy obsługę zdarzenia dla elementów tworzonych dynamicznie przez aplikację;
- modyfikuje dokument HTML dokonując zmian z drobnymi animacjami;
- łączy animacje w sekwencje następujące jedna po drugiej;
- korzysta z efektów jQuery do modyfikowania dokumentu HTML.

Czynności uczniów	Działania nauczyciela	Materiał
Zapoznają się z treściami zawartymi w opisie implementacji.	Wyjaśnia wątpliwości dotyczące treści zadań.	Opis implementacji.
Wykonują ćwiczenia praktyczne.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Opis implementacji, pliki z implementacją, http://jquery.com/

Konspekt-scenariusz **Modułu C2.4**

Daniel Mendalka

Temat: Canvas

Nazwa implementacji: Canvas

Opis implementacji: Stworzymy aplikację pozwalającą na rysowanie prostych figur geometrycznych.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- wdrażanie do tworzenia aplikacji z wykorzystaniem HTML;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z tworzeniem aplikacji;

b) szczegółowe: Uczennica/uczeń…
- poznaje sposoby animacji rysowanych obiektów;
- kształtuje umiejętność rysowania i animowania z wykorzystaniem elementu *Canvas*;
- kształtuje umiejętność interakcji pomiędzy użytkownikiem a narysowanymi obiektami;
- odczuwa satysfakcję z wykonania działającej aplikacji.

Materiał nauczania-uczenia się:
- element *Canvas* i dostępne w nim metody do rysowania obiektów;
- zdarzenia wywoływane poprzez kliknięcia myszką;
- animowanie z wykorzystaniem funkcji: *setInterval*, *setTimeout*, *requestAnimationFrame*.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- wskazuje sposoby tworzenia animacji rysowanych obiektów i omawia różnice między nimi;
- tworzy i rysuje proste figury geometryczne z wykorzystaniem różnych stylów.
- właściwie reaguje na akcje wykonywane przez użytkownika z użyciem myszki.

Czynności uczniów	Działania nauczyciela	Materiał
Zapoznają się z treściami zawartymi w opisie implementacji.	Wyjaśnia wątpliwości dotyczące treści zadań.	Opis implementacji.
Wykonują ćwiczenia praktyczne.	Wspiera uczniów, koryguje błędy, naprowadza, motywuje.	Opis implementacji, pliki z implementacją.

Konspekt-scenariusz **Modułu C3.1**

Krzysztof Bytow

Temat: Środowisko mikrokontrolera

Nazwa implementacji: Wykorzystanie wejścia analogowego do sterowania diodami.

Opis implementacji: Zastosowanie modułu-interfejsu Arduino oraz obsługa interaktywnego terminala Arduino IDE, służącego do programowania mikrokontrolera. Prezentacja i wyjaśnienie sposobu zestawiania połączeń na podstawie dokumentacji ilustrującej montaż układów ćwiczeniowych. Podłączenie i sterowanie diodami elektroluminescencyjnymi z wykorzystaniem potencjometru i buttonów. Przełożenie odczytów wartości z wejścia analogowego na sterowanie liczbą świecących diod.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- doskonali umiejętność łączenia, konfiguracji i programowego sterowania diodami;
- kształtuje umiejętność obsługi terminala do pisania kodu sterującego;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- 10 diod elektroluminescencyjnych; 3 buttony;
- 10 rezystorów 220 Ω; potencjometr 10kΩ
- dokumentacja techniczna mikrokontrolera Atmega 328 z układu Arduino.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- trafnie objaśnia pojęcia: *mikrokontroler*; *potencjometr*; *dioda elektroluminescencyjna*; *button*; *opornik*;
- poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino;
- stosuje elementy kodu do tworzenia i modyfikacji programów sterujących modułem-interfejsu;
- potrafi przesłać wyniki z układu do komputera;
- poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
- steruje diodą elektroluminescencyjną oraz modyfikuje treść wyświetlanych komunikatów.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości	Przeprowadza pokaz działania robota MAOR opartego na układzie Atmega.	• Pojęcia: mikrokontroler, button, opornik, potencjometr, wejście analogowe, wejście/wyjście cyfrowe, dioda elektroluminescencyjna; http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Mikrokontroler http://arduino.cc/en/Tutorial/Button http://arduino.cc/en/Tutorial/DigitalPins http://arduino.cc/en/Tutorial/AnalogInput http://pl.wikipedia.org/wiki/Opornik http://pl.wikipedia.org/wiki/Potencjometr • Wprowadzenie do środowiska Arduino; http://e-swoi.pl/wiki/article/arduino-podstawy/ http://e-swoi.pl/wiki/article/mechatronika-faq/ • Filmy instruktażowe.
	Prezentuje układ Arduino, na którym będą prowadzone ćwiczenia. Omawia elementy wchodzące w skład zestawu. Uruchamia środowisko programistyczne Arduino IDE, objaśniając poszczególne funkcje programu. Pokazuje wstępną konfigurację programu w celu komunikacji między komputerem, a modułem. Omawia kod źródłowy i jego poszczególne elementy.	
Montują przykładowe układy do sterowania diodami. Wprowadzają kod sterujący i testują działanie układu; modyfikują fragmenty kodu i obserwują skutki zmian.	Zachęca uczennice i uczniów do samodzielnego podłączenia układu i do zaprogramowania mikrokontrolera przykładową procedurą obsługi diody. Podpowiada, jakie szczegółowe działania muszą podjąć uczennice i uczniowie, aby ich układ funkcjonował prawidłowo, w pełni zgodnie z zadaniem.	• Termometr cyfrowy, termometr rtęciowy lub multimetr z możliwością pomiaru temperatury.
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące.	
	Formułuje zadania rozszerzające: • Zmodyfikuj program tak, aby diody mrugały w odstępach losowych; • Zaprogramuj efekt fali z regulacją szybkości działania wykorzystując potencjometr	

Konspekt-scenariusz **Modułu C3.2**

Krzysztof Bytow

Temat: Protokół komunikacyjny 1-wire

Nazwa implementacji: Protokół komunikacyjny 1-wire

Opis implementacji: Budowa układu i programu do odczytu danych wykorzystując interfejs 1-wire na przykładzie czujników Dallasa DS18B20. Rozszerzenie wiedzy dotyczącej adresowania czujników ich sposobów zasilania i wykorzystywania bibliotek w celu sterowania.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- poznaje skale i jednostki temperatury oraz zależności między nimi;
- kształtuje umiejętność wykorzystania bibliotek;
- odczuwa satysfakcję z tego, że działa zmontowany własnoręcznie układ elektroniczny.

Materiał nauczania-uczenia się:
- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa z zestawem przewodów połączeniowych;
- 2 czujnik ds18b20;
- 1 rezystor 4,7kΩ ;
- wyświetlacz LCD 2x16; potencjometr 10 kΩ.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- zgodnie z zasadami działania podłącza czujnik pomiarowy;
- prawidłowo buduje i oprogramowuje moduł-interfejs służący do pomiaru temperatury;

- uruchamia ukazywanie odczytów na wyświetlaczu LCD lub w środowisku Linux;
- modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy;
- potrafi przedstawić skale i jednostki temperatury oraz zależności między nimi;
- dokonuje odczytu adresu czujników;
- trafnie używa sformułowań: *czujnik*, *stopnie Celsjusza*, *1-wire*, *czułość*, *wejście cyfrowe*, *biblioteka*.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w prezentacji i pokazie, zadają pytania, wyjaśniają wątpliwości.	Prezentuje złożony układ Arduino z zaimplementowanym programem do odczytu temperatury z czujnika. Omawia zasadę odczytu, zwraca uwagę na dokładność pomiaru, zachęca do zapoznania z dokumentacją czujnika. Porusza temat sposobów zwiększania dokładności, co ma na nią wpływ i jak można ją poprawić. Omawia kod źródłowy i jego poszczególne elementy. W dalszej części wyjaśnia i prezentuje przesyłanie wyników do komputera i sposoby na ich obróbkę. Prezentuje podłączenie wyświetlacza LCD oraz konfigurację w kodzie źródłowym.	• Pojęcia: czujnik, opornik, LCD, potencjometr, biblioteka, 1-wire; • Filmy instruktażowe; • Prezentacja multimedialna; • Filmy dostępne w serwisie http://www.youtube.com/ hasła kluczowe: arduino lcd, arduino pomiar temperatury, 1-wire; • Tutoriale: http://e-swoi.pl/wiki/article/arduino-podstawy/ http://www.pjrc.com/teensy/td_libs_OneWire.html http://milesburton.com/Main_Page?title=Dallas_Temperature_Control_Library#Introduction
Biorą udział w dyskusji.	Prowadzi dyskusję dotyczącą sposobów odczytu temperatury.	• Termometr cyfrowy, termometr rtęciowy lub multimetr z możliwością pomiaru temperatury.
Wykonują implementację modułu-interfejsu do pomiaru temperatury z jednym czujnikiem i dwoma czujnikami. Wykonują implementację sterowanie wyświetlaczem LCD.	Zachęca uczennice i uczniów do samodzielnego podłączenia układu i zaprogramowania przykładowym programem.	• Dokumentacja techniczna http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf http://pdf1.alldatasheet.com/datasheet-pdf/view/63663/HITACHI/HD44780U.html • Zależność między temperaturą wyrażoną w stopniach Celsjusza t [°C] a wyrażoną w Kelwinach t [K]; • Zależność między temperaturą wyrażoną w stopniach Celsjusza t [°C] i Fahrenheita t [°F];
	Formułuje zadania obligatoryjne: • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące;	
	Formułuje zadania rozszerzające: • Rozbuduj i oprogramuj układ pomiaru temperatury o wyświetlacz LCD, na którym będą prezentowane (wyświetlane) wyniki pomiaru temperatury.	

Konspekt-scenariusz **Modułu C3.3**

Krzysztof Bytow

Temat: Pomiar i odczyt odległości na LCD

Nazwa implementacji: Ultradźwiękowy pomiar odległości

Opis implementacji: Pomiar odległości z wykorzystaniem czujnika ultradźwiękowego HS-SR04. Prezentacja odczytów na wyświetlaczu LCD oraz sygnalizacja wizualna wykorzystująca diodę RGB, gdy obiekt znajduje się w zadanej odległości.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):

- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń…

- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- doskonali umiejętność łączenia, konfiguracji i programowego sterowania diodą RGB;
- kształtuje umiejętność obsługi terminala do pisania kodu sterującego;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- kształtuje umiejętność podłączania czujnika ultradźwiękowego i dokonywania pomiarów odległości.

Materiał nauczania-uczenia się:

- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa z zestawem przewodów połączeniowych;
- czujnik hs-sr04;
- 3 rezystory 220 Ω,
- wyświetlacz LCD 2x16; potencjometr 10 kΩ.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń...

- trafnie objaśnia pojęcia: *czujnik ultradźwiękowy*, *wyświetlacz ciekłokrystaliczny*, *RGB*;
- poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino;
- potrafi przesłać wyniki z układu do komputera;
- poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
- potrafi podłączyć czujnik ultradźwiękowy i dokonać pomiarów odległości;
- steruje diodą RGB.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Trener prezentuje złożony układ Arduino z zaimplementowanym programem do odczytu danych z czujników. Omawia zasadę działania protokołu komunikacyjnego oraz działania czujnika ultradźwiękowego. Omawia kod źródłowy i jego poszczególne elementy. W dalszej części wyjaśnia i prezentuje podłączenie diody RGB do układu z czujnikiem ultradźwiękowym.	• Pojęcia: mikrokontroler, opornik, potencjometr, dioda RGB, wejście/wyjście cyfrowe, czujnik ultradźwiękowy, LCD; http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Mikrokontroler http://arduino.cc/en/Tutorial/DigitalPins http://pl.wikipedia.org/wiki/Opornik http://pl.wikipedia.org/wiki/Potencjometr • Wprowadzenie do środowiska Arduino; http://e-swoi.pl/wiki/article/arduino-podstawy/ • Filmy instruktażowe.
Montują przykładowy układ pomiaru odległości z wykorzystaniem czujnika ultradźwiękowego. Wprowadzają kod sterujący i testują działanie układu: modyfikowanie fragmentów kodu i obserwowanie skutków zmian. Montują przykładowy układ pomiaru odległości rozszerzonego o diodę RGB.	Zachęca uczennice i uczniów do samodzielnego podłączenia układu i do zaprogramowania mikrokontrolera przykładową procedurą obsługi pomiaru odległości. Podpowiada, jakie szczegółowe działania muszą podjąć uczennice i uczniowie, aby ich układ funkcjonował prawidłowo, w pełni zgodnie z zadaniem.	• Dokumentacja techniczna http://pdf1.alldatasheet.com/datasheet-pdf/view/63663/HITACHI/HD44780U.html http://users.ece.utexas.edu/~valvano/Datasheets/HCSR04b.pdf
Wykonują implementację modułu-interfejsu do pomiaru temperatury z dwoma czujnikami (zadanie w parach).	**Formułuje zadania obligatoryjne:** • Opisz w e-Portfolio Serwisu e-Swoi jak najkrócej to, co uważasz za osiągnięcie z zajęć; • Umieść w e-Repozytorium Serwisu e-Swoi zmodyfikowane przez siebie kody sterujące;	

Konspekt-scenariusz **Modułu C3.4**

Krzysztof Bytow

Temat: Sterowanie elementami wykonawczymi

Nazwa implementacji: Sterowanie serwomechanizmem

Opis implementacji: Budowa, działanie i sposoby sterowania serwomechanizmem.

Proponowany czas realizacji: 90 minut

Cele:

a) ogólne (zadanie/przesłanie nauczyciela dla całych zajęć):
- kształtowanie umiejętności programowania wizualnego układów mechatronicznych;
- czynnościowe kształtowanie właściwego rozumienia kluczowych pojęć infotechnicznych;
- motywowanie i kształtowanie nawyków związanych z obsługą układów mechatronicznych;

b) szczegółowe: Uczennica/uczeń…
- posiada wiedzę z zakresu kluczowych pojęć mechatronicznych;
- doskonali umiejętność czytania dokumentacji technicznej;
- rozwija sprawność i kreatywność w montowaniu i rozbudowie modułów-interfejsów;
- kształtuje umiejętność sterowania serwomechanizmem.

Materiał nauczania-uczenia się:
- środowisko programowania Arduino IDE, układ Arduino i kabel USB;
- komputer PC z dystrybucją systemu i aplikacji Szkolnego Remiksu Ubuntu;
- płytka stykowa, zestaw przewodów połączeniowych;
- 2 diody elektroluminescencyjne; 2 buttony;
- 2 rezystory 220 Ω; potencjometr 10kΩ;
- dokumentacja techniczna serwomechanizmu.

Wskaźniki osiągania celów (efekty): Uczennica/uczeń…
- trafnie objaśnia pojęcia: serwomechanizm, biblioteka, PWM;
- poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino;

- poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
- potrafi sterować kątem osi serwomechanizmu;
- potrafi podłączyć i sterować serwomechanizmem programowo, jak i z wykorzystaniem buttonów lub potencjometru.

Czynności uczniów	Działania nauczyciela	Materiał **UWAGA:** Zakres materiału dobiera nauczyciel stosownie do możliwości, a uczniowie wybierają część zadań do realizacji.
Współuczestniczą w pokazie, zadają pytania, wyjaśniają wątpliwości.	Prezentacja działania różnych serwomechanizmów. Przybliżenie parametrów – danych technicznych serwomechanizmu użytego do zajęć.	• Pojęcia: mikrokontroler, button, opornik, potencjometr, wejście analogowe, wejście/wyjście cyfrowe, dioda elektroluminescencyjna; http://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna http://pl.wikipedia.org/wiki/Mikrokontroler http://arduino.cc/en/Tutorial/Button http://arduino.cc/en/Tutorial/DigitalPins http://arduino.cc/en/Tutorial/AnalogInput http://pl.wikipedia.org/wiki/Opornik http://pl.wikipedia.org/wiki/Potencjometr • Wprowadzenie do środowiska Arduino; http://e-swoi.pl/wiki/article/arduino-podstawy/ • Filmy instruktażowe.
Montują przykładowy układ do sterowania serwomechanizmem. Wprowadzają kod sterujący i testują działanie układu: modyfikowanie fragmentów kodu i obserwowanie skutków zmian.	Razem z uczniami dokonują analizy wyprowadzeń serwomechanizmu (rozpoznanie oznaczeń przewodów 5V,GND, sygnał sterujący). Analizują budowę kodu. Zachęca uczennice i uczniów do samodzielnego podłączenia układu i do zaprogramowania mikrokontrolera.	
	Formułuje zadaniarozszerzające: • Rozbuduj układ o czujnik ultradźwiękowy, który przy wykryciu obiektu uruchamiał by serwomechanizm i sygnalizował ruch zapaleniem diody elektroluminescencyjnej • Rozbuduj układ o czujnik temperatury i z wykorzystaniem serwomechanizmu zbuduj prosty wskaźnik temperatury (wymaga przygotowania skali temperatury i strzałki do prezentacji wskazań temperatury na skali).	

04

Ocenianie realizacji kół zainteresowań IT i ewaluacja efektów

Stanisław Ubermanowicz

Zajęcia szkolne, w tym także pozalekcyjne, podlegają obowiązkom nadzoru pedagogicznego i sprawdzania jakości kształcenia. Ze względu na specyfikę realizacji kół zainteresowań infotechnicznych szczególną rolę wyznacza się wewnętrznej ocenie i ewaluacji takich zajęć. Wskazanie adekwatnych, zintegrowanych, wystandaryzowanych procedur kontrolnych i formatywnych jest istotnym elementem wdrażania innowacyjnego Programu nauczania-uczenia się. Program ma formę opisów i materiałów dydaktycznych, na podstawie których prowadzący koła Trenerzy starają się realizować autorskie koncepcje. Wartość w takim przypadku stanowią nie same treści, lecz autentyczna funkcjonalność wdrażanej Strategii w praktyce: prawidłowa interpretacja zapisów, trafność metod oddziaływań, skuteczna realizacja wyznaczanych zadań, jakość rezultatów obserwowanych bezpośrednio i efektów miękkich wyznaczanych pośrednio. Do oszacowania tak rozumianej *funkcjonalności edukacyjnej* służą rozmaite procedury oceniania i ewaluacji. Procedury te po części mają elementy podobne (eksploracja danych, interpretacja i wnioskowanie), lecz ich różnorodność utrudnia jednoznaczne zdefiniowanie specyfiki tych dwóch odmiennych kategorii.

Ocenianie od ewaluacji należy więc rozróżniać na podstawie takich elementów, które w danej kategorii dominują:

- **Ocenianie** jest przede wszystkim *kryterialnym wyznaczaniem stanu rezultatów* poprzez różnorodne planowe lub doraźne procedury, polegające głównie na sprawdzaniu spełniania progów wymagań, wartościowaniu osiąganych efektów względem oczekiwań, różnicowaniu jednostek, grup, wytworów, procesów bądź szacowaniu jakości projektów, programów, treści, celów i środków.
- **Ewaluacja** jest przede wszystkim *relatywnym wartościowaniem kontekstowym* w procesie całościowego ujmowania zjawisk, prowadzącym do diagnozowania jakości, wyszczególniania właściwości i zależności, doskonalenia oddziaływań i formowania przemian na podstawie względnych interpretacji wielu danych źródłowych oraz długodystansowych badań i analiz. Kryteria wartościowania wynikają z założeń strategii edukacyjnej i norm populacyjnych, a nie z założeń ewaluatora.

Ocenianie w edukacji to określanie poziomów osiągnięć i spełniania założeń.

Ewaluacja w edukacji to wartościowanie procesów służące ich ulepszaniu.

Włączanie innowacji pedagogicznych do zajęć szkolnych regulują stosowne rozporządzenia. Warto tu podkreślić, że Strategia edukacyjna SWOI przeszła kompleksowe procedury trzyletniego testowania, monitorowania, oceniania, optymalizacji, ewaluacji i walidacji, nieodzowne przed jej upowszechnieniem w systemie oświaty. Dzięki temu czynności wdrożenia tej innowacji do szkół zredukowane są do kwestii formalnych. Walory merytoryczne niniejszego Programu zostały bowiem w pełni zweryfikowane empirycznie i potwierdzone w procesie instytucjonalnej walidacji.

- **Walidacja** programu nauczania-uczenia się jest *eksperckim zatwierdzeniem wartości edukacyjnej*, tj.: zasadności jego wdrażania, słuszności i realności wyznaczonych celów, prawidłowości założeń dydaktycznych i metodycznych, przydatności treści i odpowiedniości dla uczniów, adekwatności zalecanych form i środków oraz trafności prognozowania i sprawdzania efektów.

Walidacja programu to legalizacja spełniania standardów jakościowych.

Pozostawiając ekspertom i recenzentom opiniowanie co do wartości opublikowanych materiałów, w tym rozdziale przedstawiamy nauczycielom wybrane, najbardziej przydatne strategie oceniania funkcjonalności Programu w praktycznym działaniu oraz jedną z metod ewaluacji efektów.

Strategie i podmiotowość w ocenianiu

Nauczyciele mają tak silnie zakorzenioną potrzebę oceniania, że czynią to nieustannie, niemal bezwiednie. Cecha ta sprzyja niezbędnemu przy realizacji kół zainteresowań *adaptacyjnemu stylowi prowadzenia zajęć*, tj. dynamicznemu dopasowywaniu treści, metod i toku zajęć do sytuacji. Takie ocenianie ad hoc ma charakter niesformalizowany i służy przede wszystkim celom doraźnym. Ma ono jednak także olbrzymie znaczenie w formowaniu umiejętności nauczycielskich, opartych na wyćwiczeniu gotowych skryptów trafnej reakcji i oddziaływań adekwatnych do zdarzeń. Efektami są trwałe, pożyteczne ślady w strukturach umysłu osób prowadzących zajęcia, jednakże na potrzeby sprawdzania i dokumentowania jakości zajęć konieczne jest również planowe ocenianie formalne.

W kompleksowym planowaniu oceniania i ewaluacji kluczowe są: cele (po co?), podmioty (kto?), zjawiska (co?), zagadnienia (jakie kwestie?), plany (jaki zasięg?), terminy (kiedy?), metody (jak?) oraz kryteria (na podstawie czego?). Wszystkie te elementy są ze sobą ściśle powiązane, toteż na potrzeby osiągania ogólnego celu, jaki jest np. sprawdzanie i doskonalenie jakości kształcenia na kołach zainteresowań, równocześnie uwzględniać należy całość metodologiczną:

- kto lub co ma być obiektem oceny oraz kto będzie wartościował i dla kogo formułował wnioski;
- jakie komponenty, cechy, stany, procesy i efekty będą badane i pod kątem jakich problemów;
- ile grup, jak licznych, z jakich środowisk, gdzie i jak często będzie poddawanych badaniom;

» jakimi technikami i narzędziami zbierze się dane oraz do jakich norm odnoszone będą rezultaty.

Jakkolwiek realizacja tak złożonej metodologii wymaga kompleksowego zaplanowania i docelowo ujęcia zbiorczego, to jednak na potrzeby praktyki przydatny jest podział na zadania elementarne. Przykładem może być rozróżnienie ze względu na moment oceniania czy wartościowania:

» **natychmiastowe**, bezpośrednie ocenianie na zajęciach percepcji treści i prawidłowości wykonywania zadań oraz stopnia samodzielności uczniów i partycypacji w pracach zespołowych.

» **odroczone**, dokonywane po cyklu zajęć wartościowanie całościowego dorobku uczniów i indywidualnych osiągnięć udokumentowanych w e-Portfolio lub zdeponowanych w e-Repozytorium.

» **dystansowe**, pośrednie wartościowanie efektów na podstawie zmian postaw, świadomości i cech wolicjonalnych, jakie udało się wywołać u uczniów wskutek oddziaływań strategią edukacyjną.

Ocenianie natychmiastowe jest integralnym elementem stacjonarnych zajęć praktycznych.

» **Ocenianie natychmiastowe** ma szczególne znaczenie ze względu na specyfikę kompetencji infotechnicznych. Otóż w programowaniu i konstruowaniu najważniejsze są umiejętności praktyczne oraz tzw. „wiedza milcząca", wspierająca myślenie abstrakcyjne. Zwłaszcza początkujący adepci IT nie są w stanie słownie przedstawić swe koncepcje i osiągnięcia, mimo że prawidłowo wykonują zadania. Bezpośrednio na podstawie ich działań zaobserwować można to, iż nabywają pożądane dyspozycje zawodowe, wiedzę *know how* (jak wykonać) i wiedzę *techne* (zdolność wykonania), lecz nie potrafią tego wyrazić. Właśnie ta bariera językowa jest przyczyną niskiej zdawalności egzaminów na tytuł zawodowy technika informatyka czy elektronika. Tam bowiem oceniane jest sprawozdanie z działań, a nie sam rezultat wykonanych prawidłowo prac. Natomiast na kołach infotechnicznych dochodzenie do rezultatu i działająca implementacja jest wprost miarą sukcesu, widoczną i odczuwaną emocjonalnie przez wykonawcę dzieła. A sukces na początku drogi jest silną motywacją i warunkiem edukacyjnego zaangażowania ucznia w tę tak trudną dziedzinę techniki.

Ocenianie odroczone jest nieodzownym elementem mieszanej formy edukacji (b-learningu).

» **Ocenianie odroczone** dorobku uczestników kół zainteresowań IT jest konieczne z tego powodu, że część rozszerzonych zadań implementacyjnych uczniowie mają wykonywać poza szkołą, w czasie pomiędzy cotygodniowymi zajęciami stacjonarnymi. Są to zadania dodatkowe, wymagające większego nakładu pracy w zależności od indywidualnych predyspozycji. Ocena jest więc możliwa dopiero po zakończeniu prac. Ponadto – mimo braku u uczniów takich nawyków – właśnie na kołach formowana jest umiejętność dokumentowania osiągnięć w osobistym e-Portfolio. Jest to proces długotrwały, lecz konieczny ze względu na

rolę języka i dokumentacji w środowisku programistów. Dopracowanie komentarzy kodów źródłowych bądź opisanie tego, co jest dorobkiem, wymaga dłuższego czasu niż dostępny na kołach, stąd ocenianie także musi być odroczone. Warto w tym miejscu zwrócić uwagę, że łącznie ocenianie natychmiastowe i odroczone jest w tym przypadku *ocenianiem rzeczywistym*. Polega ono na ocenie realizacji przez uczniów zadań wprawdzie prostszych, ale takiego rodzaju, jakie w pracy zawodowej wykonują programiści i mechatronicy. Tak samo tworzą oni implementacje, opisują je, archiwizują i udostępniają pod ocenę zewnętrzną.

» **Ocenianie dystansowe** wymaga porównania stanów na początku i na końcu procesu. Jest ono najtrafniejszym sposobem wartościowania efektywności działań edukacyjnych. Do tego konieczne są badania panelowe, polegające na dwukrotnych pomiarach tym samym narzędziem, tych samych grup uczniów w odstępie czasu trwania pełnego cyklu zajęć. Uzupełniającą formą oceniania dystansowego są tzw. *szeregi czasowe*, kiedy to bada się i porównuje w kolejnych latach inne grupy z tej samej populacji, poddawane analogicznemu procesowi edukacyjnemu. Koła zainteresowań w danej szkole są nieliczne, a uczniowie silnie zróżnicowani, dlatego porównywanie rezultatów kolejnych roczników prowadzonych przez tego samego trenera jest najwłaściwszą przesłanką do rzetelnego oceniania jakości i doskonalenia zajęć. Błędem byłoby jednak mierzenie efektów kół zainteresowań za pomocą tradycyjnych testów dydaktycznych. W dziedzinie infotechniki zasoby wiedzy specjalistycznej zmieniają się tak gwałtownie, że zaciera się tu sens wiedzy pewnej (*episteme*), a kluczowe znaczenie nabiera kształtowanie u młodych adeptów niedocenianej w oświacie wiedzy ukierunkowującej (*phronesis*), tj. mądrości praktycznej, roztropności, racjonalności, sztuki samodoskonalenia i świadomych wyborów. Do pomiaru efektów formowania takich kompetencji używa się testów psychometrycznych. I coraz częściej cechy te są weryfikowane przy zatrudnianiu pracowników.

Ocenianie dystansowe jest kwintesencją w badaniu całościowych efektów edukacji (u-learningu)

Drugim przykładem rozczłonkowania złożonej metodologii oceniania i ewaluacji na prostsze procedury jest podział ze względu na kryteria oceniania i standardy, w powiązaniu z oceną różnych elementów – wytworów uczniowskich, przebiegu procesów oraz zmian cech osobowościowych:

» **W ocenianiu sprawdzającym realizację zadań** istotą jest weryfikacja, czy i na jakim poziomie wykonane zostały ćwiczenia i projekty implementacyjne, wyznaczone jako produkty finalne każdej jednostki zajęć pozalekcyjnych. Przyjmuje się bowiem, że na każdym spotkaniu w formie kół zainteresowań uczeń powinien starać się zrealizować domknięty, funkcjonujący wytwór infotechniczny. Osiąga to poprzez zalecany w materiałach dydaktycznych zespół aktywności i czynności umysłowych oraz wytyczony przez trenera, dopasowany do możliwości zakres prac programistycznych lub mechatronicznych. Wyznaczone zadania są podłożem *standardów wymagań*, które mogą być osiągane indywidualnie lub zespołowo, samodzielnie lub z niewyręczającym wsparciem.

Kryteriami oceniania sprawdzającego są obligatoryjne standardy wymagań wyznaczonych dla danej grupy.

Kryteriami oceniania różnicującego procesy są standardy osiągnięć innych grup podczas realizacji danej strategii.

» **W ocenianiu różnicującym procesy** istotą jest porównanie wyników obserwacji badanej grupy odbiorców i realizatorów strategii edukacyjnej do uśrednionych poziomów dużej próby badawczej. Wykorzystuje się rezultaty innych zespołów, uzyskane z wcześniejszego, w pełni kontrolowanego testowania Programu nauczania-uczenia się w różnych środowiskach. Na podstawie licznych danych, zebranych zunifikowanymi narzędziami, wyznaczane są *standardy osiągnięć* możliwych do uzyskania na analogicznych zajęciach. Taki sposób sytuowania wskaźników lokalnych na tle realnych wyników ogółu obserwacji jest podstawą trafnej oceny jakości realizacji konkretnego procesu. Trener może ocenić prowadzone przez siebie zajęcia zarówno poprzez pryzmat aktywności grupy swoich uczniów, jak też skuteczność własnych oddziaływań względem rezultatów innych trenerów.

Kryteriami oceniania wartościującego efekty są normatywne dla danej populacji standardy oczekiwań co do zmian.

» **W ocenianiu wartościującym efekty** istotą jest wyznaczenie standardów cech populacji, które stają się poziomami odniesienia. Służą one do wyznaczania dynamiki zmian cech grup zajęciowych pod wpływem intencjonalnych oddziaływań. Z badań długofalowych kumuluje się wyniki zebrane skalą psychometryczną. W miarę gromadzenia danych ustala się coraz trafniejsze normy skalujące to narzędzie. Ustalane są nie tylko poziomy standardów, lecz także to, na ile oceniany proces był wartością dla uczniów jako podmiotów edukacji. Wypowiedzi ogółu respondentów wobec kwestii badawczych wytyczają wartości oczekiwane. Do poziomów tych *standardów oczekiwań* przyrównuje się rezultaty pomiarów lokalnych na początku i na końcu całkowitego cyklu edukacyjnego. Na podstawie szczegółowych wskaźników trener może w następnym cyklu zajęć zmodyfikować swoje oddziaływania tak, aby zrównoważyć te cechy uczniów, które wymagają wzmocnienia.

Trzecim przykładem ustrukturyzowanego podejścia do oceniania i ewaluacji procesu edukacyjnego jest podział procedur ze względu na przedmiot badań oraz podmioty badane i opiniujące. W takim ujęciu najważniejsze jest to, czyje wypowiedzi (jako respondentów i zarazem podmiotów) oraz na jakiej podstawie stanowią materiał do analiz. Na kołach zainteresowań IT nieodzowne jest:

» wartościowanie działań trenera i uczniów na podstawie obserwacji podmiotów procesu;
» ocena jakości koncepcji i przebiegu zajęć na podstawie refleksji podmiotów obserwacji;
» wartościowanie zmian cech u uczniów na podstawie autorefleksji podmiotów edukacji.

W tym kontekście niezbędne jest wyjaśnienie poszerzonego zakresu użycia pojęcia „podmiot". Oczywiste jest, że głównymi podmiotami edukacji są uczniowie. Jednakże głębokość interakcji, jakie zachodzą między trenerami i uczniami, upoważnia do

włączenia nauczycieli prowadzących koła zainteresowań do kategorii **podmiotów procesu** nauczania-uczenia się. Celowo nazywamy te osoby *trenerami*, gdyż ich rolą jest *coaching* rozumiany nie tylko jako wspieranie, naprowadzanie, pobudzanie i wzmacnianie samodzielnych działań uczniów, lecz także jako formowanie świadomości celów, automotywacji, odpowiedzialności i umiejętności samosterowania własnym rozwojem. Odbywa się to poprzez partnerski dialog, co równoważy podmiotowość obu stron edukacyjnej interakcji i nakłada obowiązek wartościowania na zajęciach zarówno działań uczniów, jak i trenerów.

Podmiotami obserwacji mogą być specjaliści metodycy, ale i praktykanci uczący się zawodu nauczyciela.

Uzasadnienia wymaga zastosowane tu, a rzadko spotykane objęcie podmiotowością także osób przeprowadzających planową obserwację zajęć. Powinna to być to grupa rzeczywistych lub potencjalnych użytkowników upowszechnianej Strategii SWOI – stanowić ją mogą trenerzy prowadzący już koła IT, bądź nauczyciele informatyki lub nawet studenci-praktykanci, przygotowujący się do realizacji takich zajęć. Podkreślmy, że nie chodzi tu o *obserwacje kontrolno-oceniające*, dokonywane przez dyrekcję, lecz o *doradczo-doskonalące* i *diagnozująco-kształtujące*, służące optymalizacji procesów i rozwojowi kadry. Z jednej strony rzetelne doradztwo wymaga przeprowadzania hospitacji przez osoby o wysokich kompetencjach merytorycznych i metodycznych, a z drugiej bardzo wskazane jest uczestnictwo osób, które poprzez obserwację nabędą umiejętności realizacji specjalistycznych kół oraz przećwiczą czynności postrzegania i oceniania elementów innowacji. Od refleksyjności wszystkich tych osób zależy rzetelność oceny zajęć i przydatność hospitacji, dlatego stają się oni ważnymi **podmiotami obserwacji** i zarazem ogniwami upowszechniania Strategii.

Ocenianie realizacji innowacyjnego Programu zajęć pozalekcyjnych powinno mieć charakter demokratyczny. W odróżnieniu od typowego w szkolnictwie sprawdzania nabytej wiedzy, w ocenie podmiotów procesu edukacyjnego analizuje się głównie wskaźniki jakości pracy. Nie chodzi jednak o indywidualną ocenę ucznia lub trenera, lecz o szacowanie, jak poprzez obserwowalne wskaźniki rzeczywistych działań na zajęciach funkcjonują elementy innowacji. Ocenianie należy tu rozumieć jako wartościowanie postępowania w odniesieniu do wzorców, w kontekście oddziaływań danego podmiotu i reakcji innego podmiotu na to oddziaływanie.

Podmiotowość na kołach zainteresowań ma charakter demokratyczny i dynamiczny, z przemiennością ról oceniających i ocenianych.

Tak rozumiane role ocenianych i oceniających są dynamiczne i wielostronne. Wartościowane są aktywności uczniów, a nauczyciele hospitujący lub praktykanci oceniają działania trenerów z refleksją nad samym sobą w przyszłej roli prowadzących zajęcia. Trenerzy dokonują autorefleksji co do jakości każdej zrealizowanej jednostki dydaktycznej oraz odnoszą koncepcje modułów do potencjału szerszej kadry nauczycieli i populacji uczniów. Uczniowie w dwukrotnie wypełnianych Ankietach ewaluacyjnych oceniają doznawane procesy i nade wszystko samych siebie. Ujawniane poprzez autorefleksję zmiany ich osobowości, świadomości, postaw, dyspozycji, racjonalności i intencjonalności są najważniejszymi wyznacznikami efektów osiąganych przez **podmioty edukacji** w pozalekcyjnej formie kół zainteresowań.

Przedstawiamy tu wypracowane w Projekcie metody i narzędzia upodmiotowionego oceniania oraz ewaluacji, zgrupowane według technik bezpośredniej obserwacji działań uczestników procesu, bezzwłocznej refleksji obserwatorów i trenerów oraz dystansowej autorefleksji uczniów.

Obserwacja, hospitacja i refleksja z zajęć

Umiejętności obserwacji i samoobserwacji to nieodzowne, istotne elementy kompetencji nauczycielskich. Mają one szczególnie duże znaczenie właśnie podczas realizacji kół zainteresowań. Także refleksyjność i zdolność do konstatacji to immanentne cechy tej grupy zawodowej. Nauczyciele mają bowiem wysoce pożyteczne tendencje do wymiany spostrzeżeń i dyskusji o różnych przypadkach i sytuacjach, z jakimi spotykają się podczas zajęć. Jednakże przy wprowadzaniu innowacji pedagogicznej, do jej optymalnej realizacji i trafnej oceny nie wystarczają ogólne zdolności postrzegania i okazjonalne dyskusje. Niezbędna jest obserwacja planowa, z konstatacją rzetelną i obiektywną, ściśle ukierunkowaną na weryfikację aspektów innowacyjnych. Poszczególne rodzaje obserwacji toku zajęć można sklasyfikować według wyznaczonych celów i realizowanych funkcji:

Obserwacja kontrolno-oceniająca jest elementem nadzoru pedagogicznego.

- Obserwacja **kontrolno-oceniająca** służy dyrekcji szkoły do pozyskiwania informacji o jakości pracy nauczyciela z uczniami i zbieżności realizowanego procesu dydaktycznego z ogólnymi zadaniami placówki. W przypadku hospitacji innowacyjnych kół zainteresowań jest to bardziej ocenianie formalne niż merytoryczne, gdyż rzadko dyrektorzy mają wystarczające kompetencje infotechniczne. Ze względu na pełną autonomię dyrektorów nie będziemy tu proponować żadnej konkretnej formy takiej kontroli, podkreślając jedynie, że specyfika kół infotechnicznych wymaga całościowej obserwacji wszystkich czterech kluczowych faz realizacji danej jednostki dydaktycznej.

Obserwacja doradczo-doskonaląca służy ulepszaniu pracy trenera poprzez wsparcie kompetentnego doradcy.

- Obserwacja **doradczo-doskonaląca** służy wspieraniu trenerów prowadzących koła zainteresowań w pozyskiwaniu zewnętrznych spostrzeżeń metodycznych i uwag merytorycznych, przydatnych w optymalizacji warsztatu pracy. Najkorzystniejszą formą jest hospitacja przez innego trenera, realizującego w swojej szkole taką samą strategię innowacyjną. Obserwator musi bowiem posiadać wiedzę o ideach strategii, o istotnych elementach toku zajęć pozalekcyjnych, o zalecanych metodach oddziaływań, a zwłaszcza o taktyce niewyręczającego naprowadzania na rozwiązanie zadań. Warunkiem właściwego spełniania funkcji doradczej jest odpowiedni poziom nie tylko kompetencji psychopedagogicznych, lecz także specjalistycznych z zakresu programowania i mechatroniki.

Obserwacja diagnozująco-kształtująca służy doskonaleniu kadry poprzez doświadczanie i wymianę refleksji.

- Obserwacja **diagnozująco-kształtująca** ma na celu wartościowanie aktywności uczestników zajęć, zarówno w zakresie reakcji na oddziaływania trenera i prospołecznych interakcji w grupie, jak też w obszarach działań samodzielnych, zachowań inicjatywnych i konstruktywistycznych. Taką formę hospitacji mogą prowadzić różni nauczyciele lub nawet praktykanci, gdyż celem jest ocena obserwowalnych umiejętności

i uzewnętrznianych postaw, a te przejawy potrafią dostrzegać osoby z nauczycielskim przygotowaniem pedagogicznym. Do rzetelnej hospitacji należy jednak specjalnie przygotować się, studiując materiały metodyczne i dydaktyczne, dotyczące danej jednostki zajęć. Dyskusja po hospitacji jest wówczas pożyteczna i dla trenera, i dla obserwatora.

» Obserwacja **prognozująco-projekcyjna** służy trenerowi do ukierunkowania alternatywnych elementów Programu nauczania-uczenia się na konkretny Plan zajęć, poprzez: dobór adekwatnego dla środowiska i profilu szkoły materiału dydaktycznego, wybór z poszerzonej oferty właściwego poziomu ścieżek, modułów i implementacji oraz wykorzystanie optymalnych w danym okresie rozwojowym narzędzi, aplikacji, metodyk i języków programowania. Obserwacja toku zajęć i adaptacja Konspektów odbywa się na bieżąco, lecz niektóre wnioski optymalizacyjne ze spostrzeżeń i z autorefleksji trenera zrealizować można dopiero w następnym cyklu. Każdy kolejny cykl jest więc modyfikowany z uwzględnieniem tych wniosków. Łącznie wszystkie formy i rezultaty obserwacji wraz z refleksją mają olbrzymie znaczenie w ewaluacji formatywnej.

Obserwacja prognozująco-projekcyjna pomaga trenerowi w optymalizacji trafnego doboru treści i środków.

Wyjaśnijmy w tym miejscu zasadność używania pojęć „obserwacja" i „hospitacja" w zestawieniu z „refleksją". *Obserwacja procesu edukacyjnego* różni się od metody badań naukowych tym, że obserwator nie jest tu jedynie bezstronnym rejestratorem spostrzeżeń. Wręcz przeciwnie – jego rola w wartościowaniu zjawisk jest kluczowa, o ile formułowaniu wniosków towarzyszy głębszy namysł i trafna interpretacja. W obserwacji prowadzonej przez trenera, *refleksja* może jednak przyjąć częściowo stan skupienia się na własnej aktywności i niedostrzegania obiektywnej rzeczywistości. Dlatego równoległą, wysoce pożyteczną formą obserwacji jest *hospitacja* prowadzona przez zewnętrznego obserwatora. Umożliwia ona porównanie spostrzeżeń dokonanych z dwóch perspektyw i rozważenie trafności refleksji obydwu podmiotów obserwujących zajęcia. Należy przy tym podkreślić, że obserwacja i hospitacja dotyczą postrzegania i kompleksowego wartościowania podmiotów i elementów procesu nauczania-uczenia się, a więc na równi trenerów i uczniów, ich aktywności i interakcji, celów, metod i środków, dopasowania treści, atrakcyjności zadań itp. Z tego względu wizyt trwających krócej niż pełna jednostka dydaktyczna nie traktujemy jako hospitacji.

Hospitacja to forma planowej, ukierunkowanej obserwacji jednostki zajęć przez osobę zaproszoną do wykonania takiej procedury.

Przyjmijmy założenie, że hospitacja kół zainteresowań IT musi być czynnością **planową**, w pełni przygotowaną metodycznie i narzędziowo. Oznacza to potrzebę wcześniejszego ustalenia celów, obszarów i obiektów obserwacji, sposobów rejestracji zjawisk i wyrażania refleksji. Osoba hospitująca powinna poznać koncepcję danej jednostki dydaktycznej: jej cele i treści, planowane aktywności, metody, środki i efekty, a ponadto winna posługiwać się arkuszem zawężającym pole obserwacji. Natomiast trener prowadzący zajęcia jest automatycznie przygotowany do obserwacji, choć jemu także przydaje się narzędzie ukierunkowujące refleksję. Dzięki ukierunkowaniu na te same zjawiska, możliwe jest porównywanie spostrzeżeń obu podmiotów prowadzących

Warunkiem rzetelności ocen i trafności refleksji jest staranne przygotowanie się do hospitacji danego modułu zajęć na kole zainteresowań.

obserwację. Ma to jeszcze większe znaczenie w przypadku standaryzacji narzędzi i pozyskaniu wartości normatywnych, umożliwiających odniesienie do średnich poziomów uzyskiwanych przy realizacji innowacyjnej Strategii SWOI w wielu różnych środowiskach. Takie zunifikowane obszary obserwacji i wystandaryzowane narzędzia zostały wyznaczone i przetestowane, mogą więc być upowszechniane i wykorzystywane w praktyce oceniania kół zainteresowań infotechnicznych.

Praktyka oceniania działań i jakości zajęć

Wykorzystanie dedykowanych form i narzędzi obserwacji w powiązaniu z opisem Strategii znacznie ułatwia ocenianie i optymalizację zajęć. Ani trener, ani osoby hospitujące nie muszą już formułować szczegółowych celów obserwacji, gdyż uwzględnione są one w adekwatnych do innowacji kwestiach poruszanych w narzędziach pomiarowych. Kwestie te są szczegółowo objaśnione w dalszej części, natomiast w uogólnieniu można je wyrazić w formie pytań badawczych:

- czy odczytywanie opisów innowacji, interpretacja konspektów i ich realizacja są prawidłowe?
- czy działania jakie podejmują trenerzy i uczniowie oraz ich interakcje są zgodne z koncepcją?
- czy treści, poziomy trudności i czas na realizację są dopasowane do możliwości uczniów?
- jak wypada ocena jakości formowania kluczowych cech uczniów i idei założonych w Strategii?
- jakie elementy zajęć wymagają optymalizacji na podstawie zgodnych refleksji obserwatorów?

Hospitujący zawsze odnosi spostrzeżenia do istoty Strategii, wytycznych Konspektu i założeń trenera, a nie do własnej koncepcji zajęć.

W praktyce ocenianie sprowadza się do obserwacji rzeczywistego przebiegu zajęć w odniesieniu do opublikowanych zaleceń dydaktycznych i metodycznych oraz do oszacowywania wartości wskaźników na skalibrowanych podziałkach kwestionariuszy. Niezbędna do tego jest pełna znajomość treści Konspektu-scenariusza danej jednostki zajęć oraz wiedza o istotnych elementach innowacji, o modelowych fazach realizacji kół zainteresowań, o specyfice oddziaływań, czynności i efektów. Takie ocenianie bazuje przede wszystkim na novum założonym w Strategii, a w mniejszym stopniu na osobistych przeświadczeniach obserwatora, wynikających z jego dotychczasowych doświadczeń.

Na kołach IT zalecana jest hospitacja czynna, łącząca obserwację z wykonywaniem zadań uczniowskich.

Obserwacja jest z góry programowo ukierunkowana na istotne w Strategii zagadnienia, natomiast określanie wartości zjawisk pozostaje w gestii osób hospitujących zajęcia. Do wyboru mają oni aż po 11 poziomów oszacowywania natężenia każdego wskaźnika. Wymaga to nie tylko dostrzegania zjawisk, lecz również przypisywania im pewnych kategorii jakościowych związanych z częstością zdarzeń, poprawnością realizacji, intensywnością oddziaływań, stopniem aktywności, głęboko-

ścią interakcji, poziomem cech itp. Do tego potrzebna jest refleksja z bezpośredniego uczestnictwa, odniesiona do wiedzy merytorycznej i osobistych kompetencji, a nie tylko do opisów zalecanej metodyki realizacji. Dlatego obok formy *hospitacji biernej*, polegającej na wyłącznym przyglądaniu się procesowi, bardziej korzystną formą jest *hospitacja czynna*, połączona z wcieleniem się w rolę ucznia i wykonywaniem tych samych zadań. Jest to szczególnie wskazane, gdy obserwatorem jest kandydat na nauczyciela odbywający praktykę lub nauczyciel przygotowujący się do prowadzenia kół IT. Jeśli nie ma takiej możliwości, to obserwator powinien wczuwać się w rolę trenera bądź ucznia, adekwatnie do pól obserwacji, lecz bez jakiejkolwiek ingerencji w tok zajęć.

Przyjmujemy założenie, że osoba hospitująca wartościuje dwie kategorie **aktywności** poddane obserwacji, tj.: działania trenera i działania uczniów, a ponadto ocenia elementy **jakości zajęć** w odniesieniu do możliwości realizacyjnych trenera przy rzeczywistym potencjale danej grupy uczniów. Dla porównania trener także powinien oceniać jakość hospitowanych zajęć za pomocą tego samego narzędzia. Ponadto, w pierwszych latach realizacji kół zainteresowań, wskazane jest, aby trener oceniał każdą jednostkę dydaktyczną zajęć na potrzeby prawidłowego wdrożenia i trafnej adaptacji Programu. W tym początkowym okresie optymalnym rozwiązaniem byłoby, gdyby osoba hospitująca oceniła jeden kompletny blok złożony z czterech modułów jednorodnych (np. 4 kolejne zajęcia z programowania obiektowo-zdarzeniowego) lub po dwa moduły zwieńczające każdy z trzech zróżnicowanych bloków (np. projektowanie graficzno-skryptowe w Scratchu, programowanie obiektowo-zdarzeniowe w Lazarusie oraz mechatronikę z Arduino). Tak liczne hospitacje wydają się nierealne, lecz jeśli pełnią one jednocześnie funkcję kształcenia i doskonalenia kadry nauczycielskiej, to obserwacja uczestnicząca jest najlepszą praktyczną formą uczenia się.

Na potrzeby oceniania działań trenera i uczniów wykorzystuje się **Arkusz obserwacji**, służący do natychmiastowego dokumentowania najważniejszych elementów hospitowanej jednostki dydaktycznej. Działania te są dostrzegalne, lecz różnorodne, szybko zmieniające się, dlatego duże znaczenie ma bieżące rejestrowanie elementarnych spostrzeżeń. W praktyce można wstępnie odnotowywać jakimiś symbolami np. intensywność lub częstość występowania ocenianych elementów, lecz natychmiast po zakończeniu zajęć Arkusz obserwacji powinien zostać całkowicie wypełniony poprzez nadanie wartości każdemu ze wskaźników. Ważniejsze są wówczas spostrzeżenia, a nie głębsze przemyślenia. Natomiast na potrzeby refleksyjnego oceniania jakości zajęć wykorzystuje się **Protokół formatywny**. Wprawdzie w narzędziu tym ocenianie po części także odnoszone jest do czynności trenera i uczniów, jednak nie jako wartościowanie podmiotów, lecz procesów. Innymi słowy – refleksja polega tu na ocenie istotnych elementów jakości zajęć, ich dopasowania, proporcji, efektywności, atrakcyjności itp., a wszystko pod kątem optymalizacji przyszłych działań. Dzięki takiej formule, ten sam Protokół formatywny służy zarówno osobie hospitującej, jak i prowadzącej zajęcia, przy czym najcenniejsza jest właśnie możliwość porównania refleksji obserwatora z autorefleksją trenera. Przyjrzyjmy się narzędziom pomiarowym i schematom ich interpretacji.

Arkusz obserwacji służy osobom hospitującym do ukierunkowania uwagi na aspekty metodyczne i do oceniania kluczowych w toku zajęć aktywności trenerów i uczniów.

Protokół formatywny przeznaczony jest dla trenerów i dla osób hospitujących, ukierunkowując ich refleksję na ocenę ważnych wskaźników jakości danej jednostki dydaktycznej.

Arkusz obserwacji

Działania trenera

1. Wprowadza w tematykę i cele zajęć	zbyt płytko	wystarczająco	optymalnie	nieco za głęboko	zbyt głęboko
2. Przedstawia założony efekt modułu	niewystarczająco	wystarczająco	optymalnie	nieco szerzej	nadmiernie
3. Wyznacza uczniom działania i zadania	zbyt typowe	mało ambitne	optymalnie	dość ambitne	zbyt ambitne
4. Zaciekawia i ukierunkowuje uwagę	niewystarczająco	nieco za mało	optymalnie	dość znacznie	nadmiernie
5. Buduje pozytywne relacje i zachowania	zbyt słabo	trochę za słabo	optymalnie	trochę za silnie	zbyt silnie
6. Motywuje do podejmowania wysiłku	niewystarczająco	nieco za mało	optymalnie	dość znacznie	nadmiernie
7. Obserwuje i dostrzega trudności uczniów	niewystarczająco	nieco za mało	optymalnie	dość wnikliwie	zbyt wnikliwie
8. Naprowadza poprzez pytania (inquiring)	zbyt rzadko	wystarczająco	optymalnie	dość często	zbyt często
9. Akcentuje najistotniejsze treści zajęć	zbyt rzadko	wystarczająco	optymalnie	dość często	zbyt często
10. Wdraża do prowadzenia e-Portfolio	zbyt rzadko	dość rzadko	optymalnie	dość często	zbyt często
11. Formuje kluczowe idee Strategii SWOI	niewystarczająco	nieco za mało	optymalnie	nieco za mocno	nadmiernie

Działania uczniów

1. Uważnie słuchają i obserwują	nieuważnie	mało uważnie	umiarkowanie	dość uważnie	uważnie
2. Wykazują zainteresowanie zadaniami	zbyt małe	trochę za małe	umiarkowane	dość duże	duże
3. Pytają, dyskutują, wypowiadają się	rzadko	dość rzadko	umiarkowane	dość często	często
4. Wykazują zrozumienie materiału	nikłe	nieco za małe	umiarkowane	dość znaczne	znaczne
5. Zgłaszają choćby drobne pomysły	rzadko	dość rzadko	umiarkowane	dość często	często
6. Tworzą coś, piszą kod lub montują	niepoprawnie	niezbyt dobrze	umiarkowane	dość poprawnie	prawidłowo
7. Wypełniają zasoby e-Repozytorium	niechętnie	niezbyt chętnie	umiarkowane	dość chętnie	chętnie
8. Starają się pomóc innym w grupie	niechętnie	niezbyt chętnie	umiarkowane	dość chętnie	chętnie
9. Przychylnie odnoszą się do Trenera	niepoprawnie	niezbyt dobrze	umiarkowane	dość poprawnie	prawidłowo
10. Realizują idee i cele Strategii SWOI	niechętnie	niezbyt chętnie	umiarkowane	dość chętnie	chętnie

Interpretacja merytoryczna 11 wskaźników **Działań trenera** (DT):

- **DT1. Ocena prawidłowości wprowadzania uczniów w tematykę i w cele jednostki dydaktycznej.**
  Najważniejsza jest tu ocena, czy wprowadzenie jest dynamiczne, zwięzłe, bez nadmiaru przekazywania tego, co później i tak trzeba objaśniać w toku ćwiczeń. Podczas testowania Strategii ujawniła się właśnie skłonność do zbyt głębokich wprowadzeń na wstępie zajęć. Wartościowanie sposobu ukierunkowania uczniów na treść wymaga zapoznania się z Konspektem-scenariuszem.
- **DT2. Ocena sposobu przedstawiania uczniom z góry założonych efektów jednostki dydaktycznej.**
  Głównym, koniecznym efektem każdego modułu zajęć musi być doprowadzenie do realizacji w pełni dokończonego dzieła. Prezentowany wzorzec powinien inspirować uczniów, a nie być gotową matrycą do skopiowania. Wartościowanie formy ukazywania uczniom tego, co ma powstać jako rezultat ich działań, wymaga zapoznania się z opisem Implementacji tworzonej na zajęciach.
- **DT3. Ocena tego, na ile działania i zadania wyznaczane uczniom w toku zajęć były ambitne.**
  Osąd wartości wyznaczonych uczniom zadań warto odnieść w kontekście tzw. *strefy najbliższego rozwoju* Wygotskiego, tj. wymagań nieco powyżej możliwości jednostki, jednakże realnych do wykonania ze względu na wspierające oddziaływanie otoczenia. Wytyczanie zadań szczegółowych leży w gestii trenera, który powinien dopasować sugestie z Konspektu do poziomu uczniów.
- **DT4. Ocena poziomu wyzwalania zaciekawienia i ukierunkowania uwagi uczniów w toku zajęć.**
  Osąd prawidłowości w pobudzaniu zaciekawienia i koncentrowaniu uwagi jest trudny do wyrażenia, gdyż silnie zależy od indywidualnych zachowań uczniów. Zróżnicowanie cech osobniczych uczestników kół jest tak duże, że należy abstrahować od pojedynczych przypadków, koncentrując się głównie na ocenie metodycznej jakości oddziaływań trenera i reakcji większości w grupie.
- **DT5. Ocena prawidłowości w budowaniu pozytywnych relacji i zachowań uczniów na zajęciach.**
  Budowa silnych więzi z uczestnikami kół jest ważnym zadaniem trenerów, wymagającym waloryzacji relaksacyjno-emocjonalnej. Wartościowanie sposobu formowania relacji i pożądanych zachowań odnosi się do nieodzownych na kołach zainteresowań takich cech i kompetencji społecznych, jak: wzajemny respekt, styl partnerski i negocjacyjny, taktyka wyzwalająca i adaptacyjna.
- **DT6. Ocena mechanizmów stymulacji i motywacji uczniów do podejmowania twórczego wysiłku.**
  W pobudzaniu uczniów do aktywności trenerzy powinni stosować adekwatne do sytuacji zabiegi metodyczne i socjotechniczne, lecz z zachowaniem umiaru wynikającego z zasady optymalnego, niższego poziomu stymulacji wymaganego w przypadku zadań trudnych. Należy tu uwzględniać także silną autostymulację uczniów, zwłaszcza przy wykonywaniu ćwiczeń mechatronicznych.
- **DT7. Ocena mechanizmów sprzężenia zwrotnego przy dostrzeganiu indywidualnych trudności.**
  Wartościowanie to dotyczy obserwacji przez trenera uczniów pod kątem wychwytywania problemów z wykonywaniem zadań. Należy jednak podkreślić, że rozwiązywanie problemów jest ważną składową metodyki kół zainteresowań, dlatego w ocenie uwzględniać należy, czy problemy wymagały interwencji, czy może uczniowie powinni trudności pokonywać bardziej samodzielnie.

- **DT8. Ocena realizacji inquiringu z naprowadzaniem na rozwiązania poprzez stawianie pytań.**
  Chodzi tu o oszacowanie adekwatnej do potrzeb częstości zadawania pytań pomocniczych bądź innych form indagujących, naprowadzających na samodzielne dochodzenie do wykonania zadań. Istotą jest niewyręczające wspieranie poprzez taką postać podpowiedzi, która nie podaje gotowego rozwiązania. Takie formy wspierania są znacznie trudniejsze dla trenerów niż wyjaśnianie.
- **DT9. Ocena realizacji kształtowania pojęć kluczowych poprzez akcentowanie istotnych treści.**
  Wartościowanie dotyczy tego, czy trafnie dobrano częstość podkreślania najważniejszych elementów w toku zajęć, celem ukierunkowania na najistotniejsze zagadnienia w materiale nauczania-uczenia się. W charakterystycznej dla infotechniki potrzebie przyswajania wielu nowych słów chodzi o optymalne akcentowanie tych czynności, które kształtują prawidłowe rozumienie pojęć.
- **DT10. Ocena systematyczności wdrażania uczniów do samodzielnego prowadzenia e-Portfolio.**
  Podczas zajęć Trenerzy powinni formować u uczniów umiejętność skrupulatnego dokumentowania swych choćby drobnych osiągnięć w postaci zwięzłych notatek i plików archiwizowanych w e-Portfolio bądź udostępnianych w e-Repozytorium. Ze względu na trudności uczniów z opisywaniem dorobku, te umiejętności językowe muszą być kształtowane na kołach zainteresowań IT.
- **DT11. Ocena intensywności formowania u uczniów chęci realizacji kluczowych idei Strategii.**
  Idee te są podbudową programową wynikającą z potrzeby kształtowania nastawionego na efekty przyszłe, na ukierunkowanie rozwoju w pożądanych obszarach kompetencji społecznych i zawodowych. Do prawidłowego wartościowania tego wskaźnika konieczne jest wnikliwe zapoznanie się z opisem założeń Strategii SWOI, zawartym w materiałach pt.: „Idee – studium definicyjne".

Interpretacja merytoryczna 10 wskaźników **Działań uczniów** (DU):

- **DU1. Ocena aktywności recepcyjno-asymilacyjnej – uważne słuchanie trenera i obserwacja.**
  Wskaźnik ten związany jest z oszacowaniem poziomu skupienia uwagi uczniów na przekazach werbalnych i wizualnych podczas realizowania form podających. Ma to miejsce zwłaszcza w początkowej fazie wprowadzania do zajęć, gdy uczniowie przyswajają nowe informacje (*receiving*).
- **DU2. Ocena aktywności emocjonalno-motywacyjnej – stopień zainteresowania zadaniami.**
  Wartościowanie tego wskaźnika polega na oszacowaniu zaciekawienia problematyką i chęci wykonywania stawianych zadań, zarówno czynnościowo-manualnych, jak i umysłowo-logicznych. Najsilniej powinno to być widoczne jako efekt pobudzania uczniów do działań (*faza sensytywna*).
- **DU3. Ocena aktywności interakcyjno-komunikacyjnej – częstość pytań, dyskusji i wypowiedzi**.
  Wskaźnik ten dotyczy natężenia wszelkich aktywności wyrażanych werbalnie przez uczniów, lecz w ujęciu uogólnionym na grupę. Najintensywniej występuje to wtedy, gdy uczniowie indagują trenera, starając się pozyskać niezbędne do zrozumienia, brakujące informacje (*faza responsywna*).

- **DU4. Ocena aktywności percepcyjno-inferencyjnej – poziom oznak zrozumienia materiału.**
  Oszacowanie oznak zrozumienia może być dokonane jedynie na podstawie obserwacji wskaźników pośrednich, np. sensowności pytań, trafności wniosków, skuteczności w grach logicznych. Ujawnia się to zwłaszcza wtedy, gdy uczniom wystarcza wsparcie niewyręczające (*inquiring*).
- **DU5. Ocena aktywności konceptualnej – częstość zgłaszania propozycji drobnych rozwiązań.**
  Wychwytuje się wszelkie przejawy własnej inwencji uczniów, nawet pozornie błahych. Mogą to być koncepcje indywidualne (np. grafiki) lub wkłady do pracy zespołowej (np. propozycje nazw, obiektów, procedur, układów), służące poszukiwaniu rozwiązań w *fazie problemowej*.
- **DU6. Ocena aktywności konstruktywistycznej – poprawność tworzenia grafiki, kodu lub układu.**
  Obserwuje się prawidłowość wykonywania elementów implementacji zaplanowanej do realizacji w danym module. Zależnie od rodzaju zadań, ocenia się nie tylko sam wytwór, lecz także proces w miarę samodzielnego dochodzenia do rezultatu końcowego w *fazie konstruktywnej*.
- **DU7. Ocena aktywności w udostępnianiu swych dzieł – chęć dzielenia się dorobkiem w Sieci.**
  Wskaźnik ten dotyczy oceny nastawienia uczniów do umieszczania opracowanych przez siebie wytworów w katalogach e-Repozytorium. Wiąże się z potrzebą formowania pozytywnych postaw wobec kluczowej idei wolności dzieł programistycznych i otwartości zasobów źródłowych.
- **DU8. Ocena aktywności pro-społecznościowej – chęć udzielania wsparcia innym z grupie.**
  Wskaźnik ten dotyczy oceny nastawienia uczniów do współpracy w zespole i udzielania koleżeńskiej pomocy na miarę własnych możliwości. Wiąże się to z przygotowywaniem do partycypacji w działaniach społeczności sieciowej i przyszłego uczestnictwa w programowaniu zespołowym.
- **DU9. Ocena zachowań uczniów wobec trenera – respekt, przychylność, prawidłowość relacji.**
  Wartościowanie wynika z obserwacji stylu odnoszenia się uczniów do trenera, co jest pośrednim wskaźnikiem postaw wobec osoby prowadzącej zajęcia. Należy tu jednak uwzględniać specyfikę relacji na kołach zainteresowań, zarówno wzajemny respekt, jak i nieodzowny styl partnerski.
- **DU10. Ocena nastawienia wobec Strategii – chęć realizacji jej kluczowych idei i celów.**
  Wskaźnik ten jest oszacowaniem ogólnego nastawienia uczestników do innowacji edukacyjnej, zarówno co do założeń podbudowy ideowej (zwłaszcza idei tworzenia implementacji), jak też w zakresie akceptacji długofalowych celów ogólnych oraz doraźnych, szczegółowych celów zajęć.

Wartościowanie działań trenerów i uczniów

Wartościowanie działań edukacyjnych poprzez obserwację jest czynnością złożoną z dwóch zasadniczych faz, tj.: gromadzenia spostrzeżeń i analizy dostrzeżonych zjawisk. W przyjętym modelu osoba hospitująca już w trakcie obserwacji dokonuje wstępnych oszacowań co do wartości wyznaczonych zagadnień i może odnotowywać swe oceny w kwestionariuszu. Po zakończeniu zajęć przypisuje wszystkim pozostałym pozycjom w Arkuszu określone poziomy, wynikające z intensywności zaobserwowanych zachowań. Powstaje w ten sposób dokument ze zbiorem wskaźników **jakościowych**, które powinny być wykorzystane

Po hospitacji należy natychmiast podzielić się z trenerem spostrzeżeniami dotyczącymi aspektów jakościowych.

wprost do bezzwłocznego omówienia z trenerem. Chodzi o to, aby podczas dyskusji mieć świeżo w pamięci sytuacje wpływające na konkretne wybory opcji wytyczonych na skali. Taka bezpośrednia, wspólna analiza jakości dostrzeżonych oddziaływań trenera i aktywności uczestników ma olbrzymie znaczenie w doskonaleniu zajęć.

Oprócz doraźnej wymiany spostrzeżeń i wniosków co do jakości zajęć, bardzo przydatna jest porównawcza analiza rezultatów obserwacji **w ujęciu ilościowym**. Ma to głębokie uzasadnienie zwłaszcza wtedy, gdy hospitacja jest kilkurazowa w danym cyklu zajęć bądź powtarzana w ciągu kilku lat. Przejście na wartościowanie wyrażane parametrami statystycznymi daje możliwość porównywania różnych modułów, cykli i roczników, a także odnoszenia osiągnięć lokalnych do norm charakteryzujących populację.

Skala szacunkowa jest próbą ilościowego wyznaczania poziomów cech jakościowych.

Na potrzeby analiz wykorzystuje się podstawowe statystyki pozycyjne, a więc wystarczy kalibracja wskaźników i porównywanie poziomów oraz obliczanie średnich arytmetycznych w celu ustalenia wartości zmiennych zagregowanych z kilku wskaźników. Kluczowym zagadnieniem jest przypisanie odpowiednich wag liczbowych do opcji zaznaczonych przez osoby wypełniające Arkusz obserwacji. Warto podkreślić, że wybory opcji są swego rodzaju oszacowaniem poziomu każdego wskaźnika, stąd nadanie im wartości liczbowych odbywa się w przestrzeni zwanej *skalą szacunkową*. Liczby spełniają tu jedynie rolę pomocniczą w określaniu poziomów i nie należy np. traktować poziomu 4 jako dwukrotnie lepszego od 2. Ponadto przejście z poziomu 3 na 4 wcale nie musi być tak samo wartościowe jak zmiana z 4 na 5.

Skala ocen akademickich może służyć do zrozumiałego wyrażania poziomów ilościowych i jakościowych.

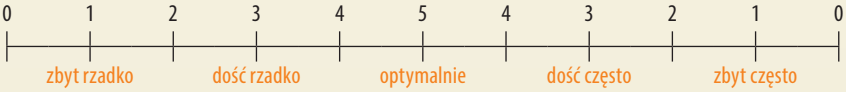
Zwróćmy uwagę, że właśnie takie cechy ma wyrażana liczbowo skala ocen szkolnych, dla których dopuszcza się obliczanie średnich. Z tego względu najlepszym sposobem kalibrowania Arkusza obserwacji jest nadanie opcjom wypowiedzi wag, które umożliwią wyrażanie wyników w skali o utrwalonym społecznie skrypcie interpretacji. Takim skryptem w Polsce jest intuicyjne wiązanie liczbowych wartości ocen z poziomami jakościowymi: 5 = *bardzo dobry*, 4 = *dobry*, 3 = *dostateczny*. Co do niższych poziomów, to zastosujemy tu **skalę ocen akademickich**, gdyż jest ona bardziej przydatna do wartościowania procesów edukacyjnych i ma bardziej spójne nazewnictwo stopni (bez '*dopuszczający*'). Przyjmujemy wartość progową 2 = *niedostateczny*, a wszystkie wartości poniżej 2 interpretujemy również jako niedostateczny poziom wskaźnika.

Kalibracja wskaźników polega na przypisaniu do każdej opcji wypowiedzi odpowiedniej wartości liczbowej.

Sposób przypisywania wag działaniom trenera różni się zasadniczo od wymiarowania działań uczniów. Wynika to z odmiennych charakterystyk skalowania narzędzia w zakresie opcji wyborów. W ocenianiu działań trenera środkową opcją jest zwrot '*optymalnie*', który umożliwia wartościowanie względne, odnoszone do rzeczywistych uwarunkowań w konkretnych sytuacjach i środowiskach. W uproszczonym podejściu można byłoby tej opcji przypisywać najwyższą wagę. Jednak specyfika innowacyjności kół zainteresowań powoduje, że niektóre wskaźniki

są korzystniejsze, jeśli zaobserwuje się zjawisko przekraczania pozornego optimum, tzn. gdy coś pozytywnego jest intensywniejsze (np. nieco ambitniejsze zadania), a coś niewskazanego rzadsze (np. nadmierne objaśnianie). Z tego względu potrzebne jest zastosowanie trzech rodzajów kalibracji **wskaźników działań trenera**:

Kalibracja symetryczna 0...∧...0 Wartości przypisane wypowiedziom:



Kalibracja lewoskośna 0... ∧ ...0

Kalibracja prawoskośna 0... ∧ ...0

W ocenianiu działań trenera wybory opcji skrajnych oznaczają stany nieprawidłowe, a blisko środka skali – najbardziej korzystne.

Sposób kalibracji poszczególnych wskaźników wynika nie tylko z meritum każdej pozycji w Arkuszu obserwacji, lecz także z badań. Na podstawie planowej obserwacji ponad 1300 jednostek dydaktycznych kół zainteresowań, realizowanych w różnych środowiskach, uzyskano empiryczne rozkłady statystyk, które mają charakterystyki symetryczne lub asymetryczne. W zależności od rodzaju asymetrii lewo- lub prawostronnej podjęto decyzję o przypisywaniu najwyższych wag 5,5 odpowiednio tym najbliższym opcjom obok '*optymalnie*', które wybierała większa liczba hospitujących. Wprawdzie wartość 5,5 jest już poza zakresem standardowych ocen akademickich, lecz oznacza stan najbardziej korzystny, wręcz idealny, co jest np. wyrażane na uczelniach stosujących dopisek '*celujący*' mimo oceny 5. Nieco wyższe punktowanie ściśle określonych opcji uwydatnia w rezultatach ilościowych maksymalnie pozytywne oddziaływania trenera, a redukuje nieco skutki hospitacji nierzetelnych, z automatycznym zaznaczaniem opcji '*optymalnie*'.

Niejednorodne przypisywanie wag może wydawać się trudne, lecz dla nauczycieli informatyki, będących głównymi odbiorcami Strategii, przygotowanie odpowiednich szablonów przetwarzających dane surowe w arkuszu kalkulacyjnym nie stanowi żadnej przeszkody. Najkorzystniej jest wprowadzać dane z kwestionariuszy numerując od

Kalibracja i wyznaczenie wartości normatywnych to elementy standaryzacji narzędzia pomiarowego.

lewej opcje wypowiedzi liczbami od 1 do 11 (zob. ryc.), a przetwarzanie na wagi w skali ocen zrealizować za pomocą formuł liczących.

Zastosowanie zaproponowanych tu, wystandaryzowanych schematów kalibracji wskaźników działań trenerów i uczniów umożliwi odnoszenie lokalnych rezultatów do średnich poziomów empirycznych, jakie uzyskano z bardzo licznych prób służących wyznaczeniu wartości oczekiwanych, normatywnych dla populacji (zob. tab.).

Tabela 1.

Schemat kalibracji i wartości normatywne wskaźników działań trenera

Kod wskaźnika	Rodzaj kalibracji	Średnia wartość oczekiwana
DT1	prawoskośna 0...⩘...0	4,0
DT2	prawoskośna 0...⩘...0	4,4
DT3	lewoskośna 0...⩙...0	4,6
DT4	lewoskośna 0...⩙...0	4,9
DT5	symetryczna 0...⩓...0	4,8
DT6	lewoskośna 0...⩙...0	4,9
DT7	lewoskośna 0...⩙...0	4,7
DT8	prawoskośna 0...⩘...0	4,3
DT9	prawoskośna 0...⩘...0	4,4
DT10	symetryczna 0...⩓...0	4,4
DT11	symetryczna 0...⩓...0	4,8

W ocenianiu działań uczniów najbardziej korzystny jest wybór opcji skrajnej po prawej stronie skali.

Jak już wspomniano, wymiarowanie skali służącej obserwacji uczniów ma zupełnie inną, monotonicznie narastającą charakterystykę. Środkowa opcja '*umiarkowanie*' jest w ujęciu jakościowym poziomem zaledwie dostatecznym. Wybór skrajnej opcji po stronie lewej oznacza najniższą wartość wskaźnika, a po prawej – najwyższą. Także i tu wagi przypisane opcjom wykraczają poza skalę ocen akademickich, lecz w oszacowywaniu natężenia obserwowanych zjawisk ma to specjalne znaczenie. Otóż w praktyce pomiarowej skrajne opcje na skali są bardzo rzadko wybierane. Obserwatorom pozostawia się jednak większą swobodę wyboru, przydatną zwłaszcza w sytuacji, gdy

dostrzegają oni zjawiska nad wyraz pozytywne bądź mniej lub bardziej negatywne. Dzięki większej rozdzielczości 11-stopniowa skala pomiarowa ma większą czułość i daje bardziej wnioskotwórcze dane empiryczne. Przyjmijmy zatem następujący rodzaj kalibracji **wskaźników działań uczniów**:

Kalibracja liniowa narastająca 0,5... /...5,5

Wartości przypisane wypowiedziom:

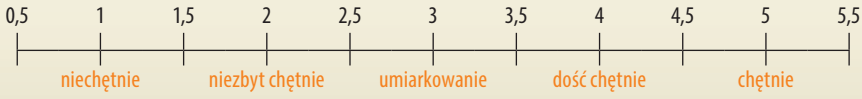
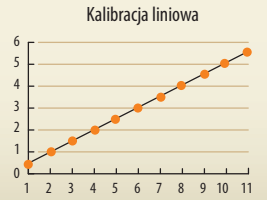


Tabela 2.

Schemat kalibracji i wartości normatywne wskaźników działań uczniów

Kod wskaźnika	Rodzaj kalibracji	Średnia wartość oczekiwana
DU1	liniowa narastająca 0,5... /...5,5 dla wszystkich wskaźników	4,3
DU2		4,2
DU3		3,8
DU4		3,9
DU5		3,5
DU6		3,8
DU7		3,1
DU8		4,2
DU9		4,7
DU10		4,3



Arkusz obserwacji jest narzędziem przeznaczonym dla osób hospitujących do wyrażania spostrzeżeń i wartościowania działań w toku zajęć. Natomiast do wyrażania refleksji i oceniania jakości zajęć służy Protokół formatywny, wykorzystywany zarówno przez hospitujących, jak i trenerów.

Protokół formatywny

Ocena zajęć

1. Trudność zajęć dla trenera	łatwe	dość łatwe	dość trudne	trudne	bardzo trudne
2. Trudność zajęć dla uczniów	zbyt łatwe	dość łatwe	optymalne	dość trudne	zbyt trudne
3. Dopasowanie treści do potencjału kadry	typowe	dość typowe	optymalne	dość ambitne	ambitne
4. Dopasowanie treści do rozwoju populacji	zbyt typowe	dość typowe	optymalne	dość ambitne	zbyt ambitne
5. Czas na realizację treści przez trenera	zbyt mało	trochę za mało	optymalnie	trochę za dużo	zbyt dużo
6. Czas na wykonanie zadań przez uczniów	zbyt mało	trochę za mało	optymalnie	trochę za dużo	zbyt dużo
7. Realizacja celów przez trenera	okrojona	nieco węższa	optymalna	nieco szersza	rozszerzona
8. Osiągnięcie efektów przez uczniów	okrojone	nieco węższe	optymalne	nieco szersze	rozszerzone
9. Reaktywność trenera (responsywność)	niewykorzystana	optymalna	nieco częstsza	nadużywana	zbyt często
10. Aktywność uczniów (sensytywność)	znikoma	nieco za mała	optymalna	dość duża	zbyt duża
11. Pochłanianie prowadzącego zajęcia	nieznaczne	wystarczające	optymalne	dość znaczne	nadmierne
12. Zanurzenie uczniów (immersja)	zbyt płytkie	nieco płytsze	optymalne	nieco głębsze	zbyt głębokie
13. Moje zadowolenie z przebiegu zajęć	małe	nieco za małe	umiarkowane	dość znaczne	znaczne
14. Atrakcyjność czynności dla uczniów	mała	nieco za mała	umiarkowana	dość znaczna	znaczna
15. Jakość koncepcji modułu zajęć	mała	nieco za mała	umiarkowana	dość znaczna	znaczna
16. Zaciekawienie uczniów treścią modułu	małe	nieco za małe	umiarkowane	dość znaczne	znaczne

Sugestie optymalizacyjne: .....

Warto zwrócić uwagę na układ zagadnień tworzący w kwestionariuszu **pary kontrolne**, czyli – podobne kwestie raz ujmowane są w perspektywie trenera, obserwatora bądź potencjalnej kadry, a drugi raz w odniesieniu do perspektywy uczniów. Umożliwia to pełniejszą analizę porównawczą.

Interpretacja merytoryczna 16 wskaźników **Oceny zajęć** (OZ):

- **OZ1. Oszacowanie trudności realizacji zajęć przez trenera prowadzącego dane koło zainteresowań.** Wskaźnik ten jest kalibrowany odwrotnie, wskutek czego wyższa ocena oznacza większą łatwość. Znamienne jest to, że zwykle w opinii trenerów zajęcia są łatwiejsze niż w opinii obserwatorów.
- **OZ2. Ocena adekwatnego doboru stopnia trudności materiału dla obserwowanej grupy zajęciowej.** Jest to oszacowanie, czy skala trudności zajęć została dobrze dopasowana do możliwości uczniów. Wskaźnik ten odnosi się do oceny prawidłowości adaptacji zaleceń Konspektu do potencjału grupy. Dla zapewnienia efektywności kół zainteresowań, zadania problemowe muszą być w miarę trudne.
- **OZ3. Ocena dopasowania realizowanych treści do uogólnionego potencjału kadry nauczycielskiej.** Chodzi tu o refleksję, czy przeznaczony na zajęcia materiał z Konspektu jest na miarę możliwości szerszej kadry nauczycieli, którzy mogliby podjąć się roli trenerów na kołach zainteresowań IT.
- **OZ4. Ocena dopasowania treści zajęć do poziomu rozwoju ogółu potencjalnych uczestników kół.** Refleksja dotyczy adekwatności wybranych z Programu do realizacji celów, metod, działań i efektów wobec realnych możliwości intelektualnych młodych adolescentów. Ocenę należy odnosić do potencjału uczniów szkół niesprofilowanych, do których głównie adresowana jest Strategia.
- **OZ5. Ocena wystarczalności czasu na realizację przez trenerów treści wyznaczonych w modułach.** Do takiej oceny niezbędne jest wcześniejsze zapoznanie się z zapisami Konspektów-scenariuszy. Refleksja dotyczy tego, czy zrealizowany materiał był odczuwany jako domknięta całość, przy czym korzystniejsza jest dynamika i swoista skrótowość (*zapping*) niż nadmiarowość czasu.
- **OZ6. Ocena wystarczalności czasu na wykonanie przez uczniów zadań wyznaczonych w modułach.** Na każdej jednostce kół zainteresowań powstaje implementacja, dlatego do oceny realizacji wytworu finalnego potrzebna jest znajomość materiałów do ćwiczeń. Refleksja odnosi się do tego, czy na elementarne czynności uczniów przeznaczano proporcjonalnie odpowiedni czas.
- **OZ7. Ocena zakresu zrealizowania celów zalecanych w Konspekcie i dopasowanych przez trenera.** W perspektywie trenera dążenie do celów oznacza kierunkowe oddziaływania na uczniów, przy czym forma kół wymusza styl adaptacyjny. Do trafnej oceny należy ustalić, czy trener zaplanował szerszy/węższy zakres celów szczegółowych, czy może był zmuszony dopasować się do sytuacji.
- **OZ8. Ocena osiągnięcia przez uczniów obserwowalnych efektów, zaplanowanych dla modułu.** W perspektywie uczniów osiąganie efektów oznacza działania, które bezpośrednio lub pośrednio świadczą o dochodzeniu do oczekiwanych rezultatów. Także i tu, oprócz odniesienia do zapisów z Konspektu, potrzebne jest uwzględnienie efektów wyznaczonych lokalnie przez trenera.
- **OZ9. Ocena responsywności trenera w odniesieniu do potrzeb wynikających z częstości pytań.** Istotą *stylu responsywnego* jest szybkie udzielanie odpowiedzi, wyrażające dokładnie to, czego oczekiwała osoba pytająca. W relatywnym ocenianiu adekwatnej reaktywności trenera musi być uwzględniane rzeczywiste zapotrzebowanie ze strony uczniów, a nie inicjatywność trenera.
- **OZ10. Ocena aktywności uczniów względem czynności oczekiwanych przy danym typie zadań.** Wskaźnik ten jest też silnie zrelatywizowany, gdyż oszacowanie poziomu aktywności uczniów należy odnosić do zaplanowanych działań, zależnych od problematyki zajęć i rodzaju ćwiczeń. Sensytywność ujawnia się poprzez czynności, a także interakcję słowną i aktywność umysłową.
- **OZ11. Ocena pochłaniania trenera w kontekście niezbędnych w toku zajęć działań i interakcji.** Wielowątkowy kompleks czynności podczas prowadzenia kół zainteresowań wymaga oceny tego, czy trener znalazł złoty środek między oddziaływaniem na całą grupę a potrzebą in-

terakcji indywidualnych. Wspieranie pojedynczych uczniów nie może odbywać się kosztem pozostałych.

» **OZ12. Ocena zanurzenia uczniów w kontekście nieodzownych, złożonych czynności umysłowych.**
Poziom *immersji* jest silnie zależny od rodzaju zadań. W pracy grupowej potrzebna jest interakcja, a w pracy indywidualnej właśnie zanurzenie się w myślach. Z tego powodu wskaźnik ten ma inną kalibrację w modułach mechatronicznych, gdyż tam dominują działania w pełni samodzielne.

» **OZ13. Ocena zadowolenia osoby wypełniającej Protokół formatywny z całości przebiegu zajęć.**
Jest to wyrażenie odczucia doznanego wskutek udziału w jednostce zajęć. Osoba hospitująca wyraża swój stosunek emocjonalny z perspektywy obserwatora, a trener z perspektywy animatora.

» **OZ14. Oszacowanie stopnia atrakcyjności dla uczniów ogółu wykonywanych przez nich czynności.**
Osoba wypełniająca Protokół może jedynie na podstawie oznak pośrednich wyrazić taką ocenę. Trudno jest jej jednak wyzbyć się wpływu osobistej opinii o jakości realizowanych zadań. Dlatego wskaźnik ten porównuje się później z wypowiedziami uczniów w Ankiecie ewaluacyjnej.

» **OZ15. Ocena koncepcji modułu zajęć, dokonana z perspektywy praktycznej realizacji Konspektu.**
Jest to wskaźnik wyrażający w większym stopniu osobistą refleksję co do jakości wykorzystanych koncepcji i materiałów dydaktycznych, jednak w odniesieniu do rzeczywistego przebiegu zajęć. Chodzi w praktyce o to, na ile proponowany moduł zajęć sprawdził się w danym środowisku.

» **OZ16. Oszacowanie stopnia zaciekawienia uczniów treścią zajęć na podstawie oznak pośrednich.**
Przy wartościowaniu tych oznak osoby oceniające powinny uwzględniać obserwowalne zachowania uczniów w odniesieniu do ich potencjału i cech wolicjonalnych. Te właściwości danej grupy uczniów ustala się na podstawie tzw. *wskaźnika buforowego* nr 18 w Ankiecie ewaluacyjnej.

Wartościowanie wskaźników jakości zajęć

W praktycznym wykorzystaniu Protokołu formatywnego olbrzymie znaczenie ma jakościowe porównanie refleksji osoby hospitującej z autorefleksją trenera. Odbywa się to poprzez zestawienie opcji wybranych na skali przez jedną i drugą osobę, wraz z dyskusją formatywną, odnoszącą się zwłaszcza do tych wskaźników, którym przypisano inny poziom. Na potrzeby dalszych analiz, w pomiarach cyklicznych przydatne jest ilościowe ujęcie wskaźników kalibrowanych w skali ocen. Analogicznie jak z Arkuszem obserwacji, tak i tu do kalibracji i normalizacji wskaźników wykorzystano bardzo liczne dane empiryczne (ponad 2600 Protokołów z testowania innowacji). Oprócz opisanych wcześniej rodzajów wymiarowania skali, dochodzi tu jeszcze kalibracja liniowa odwrócona:

Trenerzy oceniają zajęcia bardziej krytycznie niż hospitujący, dlatego podano odrębne normy dla obu tych grup.

Kalibracja liniowa odwrócona 0,5...\...5,5

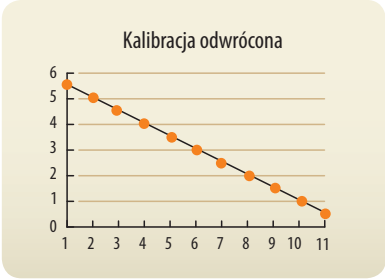
Wartości przypisane wypowiedziom:

W tabeli przedstawiono schemat kalibracji poszczególnych wskaźników z Protokołu formatywnego oraz wartości empiryczne, uzyskane z uśrednienia osobno ocen wystawionych przez osoby hospitujące i przez trenerów. Oceny te są poziomami odniesienia dla rezultatów lokalnych.

Tabela 3.

Schemat kalibracji i wartości normatywne wskaźników oceny zajęć

Kod wskaźnika	Rodzaj kalibracji	Średnia wartość oczekiwana	
		hospitujący	trenerzy
OZ1	odwrócona 5,5…\…0,5	3,5	4,1
OZ2	lewoskośna 0…∧…0	4,6	4,1
OZ3	lewoskośna 0…∧…0	4,7	4,0
OZ4	lewoskośna 0…∧…0	4,7	4,3
OZ5	prawoskośna 0…∧…0	4,6	4,5
OZ6	prawoskośna 0…∧…0	4,6	4,5
OZ7	prawoskośna 0…∧…0	4,7	4,2
OZ8	prawoskośna 0…∧…0	4,7	4,2
OZ9	lewoskośna 0…∧…0	5,0	4,8
OZ10	lewoskośna 0…∧…0	4,7	4,7
OZ11	prawoskośna 0…∧…0	4,7	4,6
OZ12 informatyka	prawoskośna 0…∧…0	4,7	4,4
OZ12 mechatronika	lewoskośna 0…∧…0	4,6	4,1
OZ13	narastająca 0,5…/…5,5	4,0	3,8
OZ14	narastająca 0,5…/…5,5	4,0	3,9
OZ15	narastająca 0,5…/…5,5	3,9	3,8
OZ16	narastająca 0,5…/…5,5	4,0	4,0



Poprzez porównanie wskaźników stanów, które nie mogą zachodzić równocześnie (interakcja i immersja) można ocenić jakość hospitacji.

Jak wynika z tabeli, dla modułów mechatronicznych wskaźnik nr 12 musi być kalibrowany odmiennie niż dla modułów informatycznych. Wynika to ze specyfiki zajęć, kiedy w pełni samodzielne konstruowanie układu na podstawie schematu wymaga całkowitego zanurzenia się ucznia w czynnościach. Natomiast podczas programowania immersja jest znacznie mniejsza, gdyż pewną część zadań realizuje się zespołowo. Proponowane narzędzia pomiarowe umożliwiają pośrednią ocenę **rzetelności** wypowiedzi obserwatorów i **trafności** oceniania zajęć, np. poprzez analizę proporcji między aktywnością a zanurzeniem uczniów. Jeśli dominuje pobudzenie, sensytywność i silna interakcja z trenerem lub grupą, to nie może występować jednocześnie skupienie na samodzielnym wykonywaniu zadań i pełna immersja z odcięciem się od otoczenia. Oczywiście mogą i powinny to być odrębne fazy zajęć, dlatego istotna jest ocena właściwych proporcji czasu przeznaczonego na każdą z faz, adekwatnie do specyfiki danego modułu.

Analiza jakości zajęć polega głównie na wyjaśnianiu przyczyn odstawania wyników od wartości oczekiwanych.

W analizie wykorzystującej aparat statystyki nie chodzi o ocenę pojedynczych przypadków. Opinie wartościujące, wyrażane przez nauczycieli obserwujących i trenerów prowadzących zajęcia, są skupiane wokół problemu badawczego. Pojedyncza wypowiedź stanowi jedynie cząstkę składową wypośrodkowanej oceny. W modelach analiz pozycyjnych wartość średnia jest optymalnym estymatorem przybliżającym wyniki do wartości oczekiwanych. Na tej podstawie zestawia się wartości lokalne z wartościami normatywnymi. W takich porównaniach najbardziej wnioskotwórcze, ale i wymagające najstaranniejszej interpretacji są tzw. **wyniki odstające**. Zasadą jest, że trzeba starać się wyjaśnić przyczynę odstawania, natomiast rezultaty, które są zgodne z oczekiwaniami, nie wymagają żadnych omówień. Wyniki odstające zwykle bezpośrednio sygnalizują to, co należałoby w procesie edukacyjnym poprawić. Przyczyny odstawania ocen mogą mieć jednak źródła niezwiązane z meritum danego wskaźnika i mogą być skutkiem występowania czynników zakłócających (np. awarii sprzętu lub systemu).

Analiza prowadzi przede wszystkim do wyszczególnienia i objaśnienia tego, co było oznakami jakichś zjawisk mniej spodziewanych, a więc nie tylko wskaźników, które osiągają niższy poziom, ale też tych, które są bezzasadnie zbyt wysokie. W rzeczywistych pomiarach wypowiedzi wobec prawidłowo sformułowanych kwestii badawczych powinny być choćby minimalnie zróżnicowane. Zatem nawet najbardziej prawidłowe wyniki empiryczne po uśrednieniu nie osiągają najwyższego poziomu, gdyż wskaźniki takie nie miałyby wymaganej mocy wnioskotwórczej. W konsekwencji – przy wyrażaniu rezultatów w skali ocen akademickich, średnie oczekiwane mają różne wartości, niższe od oceny maksymalnej.

Wartościowanie wskaźników w każdym przypadku odnoszone musi być do innowacyjnych założeń programu oddziaływań i do treści nauczania-uczenia się, opisanych w kompleksowym pakiecie materiałów metodycznych oraz zawartych w zalecanych trenerom metodach, środkach i narzędziach służących

realizacji Strategii. W ocenie harmonizowania oddziaływań bierze się pod uwagę proporcje między formowaniem poznawczym i emocjonalnym oraz między podejściem behawioralnym (rozwiązanie zadania wg podanego wzoru i ocena zrozumienia) a konstruktywistycznym (poszukiwanie rozwiązań, ocena myśli twórczej i zdolności realizacyjnych).

Określone podejścia metodyczne mają adekwatne zastosowania w sytuacjach zależnych od poziomu uprzedniego przygotowania uczących się. W początkowych fazach wczesnej edukacji z dziedzin programowania i mechatroniki większe znaczenie mają formy asymilacyjne i ćwiczeniowe, a w dalszych fazach coraz bardziej potrzebne są formy projektowe i twórcze, dlatego od trenerów wymagana jest umiejętność adaptacji do dynamicznych uwarunkowań w danym środowisku. Właśnie ze względu na zmienność sytuacji, ocena oddziaływań metodycznych musi być ściśle związana z lokalnymi okolicznościami i kontekstami, stąd narzędzia badawcze próbkują wskaźniki w sposób komplementarny, jako kontrolne pary oddziaływań i reakcji. I tak np. reaktywność trenera analizuje się w odniesieniu do sensytywności uczniów, a nasilenie pobudzenia i pochłanianie prowadzącego – do aktywności bądź zanurzenia wykonujących zadania. Na potrzeby pełniejszej oceny zachodzących procesów przydatna jest synteza materiału empirycznego.

Ocenianie jakości zajęć i analiza rezultatów muszą być zawsze odnoszone do uwarunkowań lokalnych.

Scalanie wskaźników w czynniki strategiczne

Synteza zasobów empirycznych polega na agregacji grupy ściśle wyznaczonych wskaźników i łącznym rozpatrywaniu tych czynników, które są kluczowe dla realizacji Strategii. Umożliwia ona skupienie się wokół uogólnionego problemu badawczego. Jeśli zagadnieniem strategicznym jest np. ocena jakości formowania idei i cech wolicjonalnych, to siłą rzeczy muszą być łącznie interpretowane sploty oddziaływań trenera i skutków u uczniów.

O ile odrębna analiza wskaźników daje przesłanki do oceny tego, czy pobudzenie było prawidłowe i czy reakcja była odpowiednia, o tyle synteza daje podstawy do łącznej oceny obserwowanego procesu nauczania-uczenia się i odniesienia do wartości normatywnych z ogółu badanych procesów. W zasadzie czynności analizy i syntezy realizowane są paralelnie, gdyż prawidłowa interpretacja zjawisk w danym procesie wymaga relatywnej oceny wszystkich składników elementarnych i czynników zagregowanych.

W praktyce agregacja polega na scalaniu kilku wskaźników takich zmiennych elementarnych, które merytorycznie dają się powiązać w pewną ogólniejszą kategorię. W przypadku, gdy chodzi o ocenę **jakości formowania** bazowych idei i pożądanych cech wolicjonalnych – zmiennymi uogólnionymi w ujęciu jakościowym są adekwatne kategorie: aktywizacji, twórczości, partycypacji, wolności i otwartości oraz

Synteza dwóch wskaźników oddziaływań trenera i dwóch dot. aktywności uczniów

daje trafniejszą ogólną ocenę jakości formowania cech.

dialogu i negocjacji. Każda z nich grupuje po cztery wskaźniki: dwa dotyczące oddziaływań trenera i dwa wynikające z działań uczniów. W ujęciu ilościowym przyjmujemy, że są to średnie arytmetyczne z wartości wskaźników składowych, wyrażone przez analogię w skali ocen akademickich. Przedstawiony poniżej Schemat agregacji zawiera nazwy czynników odnoszące się do idei, syntetycznie ujętą ścieżkę formowania cech, wykaz czterech wskaźników tworzących splot i wartość normatywną, wyznaczoną empirycznie na podstawie dużej próby badawczej.

Jakość formowania kluczowych idei i cech wolicjonalnych

- **Idea aktywizacji:**
  ukierunkowanie uwagi → zainteresowanie tematyką → wnikliwy odbiór treści
  Splot wskaźników: DT4, DT9, DU1 i DU2
  Średnia wartość oczekiwana: 4,5

- **Idea twórczości:**
  wyznaczanie zadań → działania implementacyjne → zrozumienie materiału
  Splot wskaźników: DT3, DT10, DU4 i DU6
  Średnia wartość oczekiwana: 4,2

- **Idea partycypacji:**
  motywacja do wysiłku → zgłaszanie pomysłów → udzielanie wsparcia
  Splot wskaźników: DT6, DT8, DU5 i DU8
  Średnia wartość oczekiwana: 4,2

- **Idea wolności i otwartości:**
  formowanie idei → realizacja celów → udostępnianie wytworów
  Splot wskaźników: DT2, DT11, DU7 i DU10
  Średnia wartość oczekiwana: 4,2

- **Idea dialogu i negocjacji:**
  budowanie relacji → interakcje w grupie → stosunek do trenera
  Splot wskaźników: DT5, DT7, DU3 i DU9
  Średnia wartość oczekiwana: 4,5

Scalanie grup sześciu odpowiednich wskaźników w czynniki strategiczne daje najtrafniejszą przesłankę do oceny jakości realizacji zajęć.

W przypadku uogólniania ocen odnoszących się do **jakości realizacji** aspektów metodycznych i dydaktycznych, każda pięciu zmiennych splotowych obejmuje po sześć różnych wskaźników. Dotyczą one komponentów składających się na charakterystyczne mikroprocesy, jakie zachodzą równolegle lub cyklicznie w toku zajęć. Jedne z nich są kanonami metodyki klasycznej, inne elementami innowacji. Umożliwia to porównywanie natężenia wdrażanych nowości względem ciągu działań tradycyjnie nieodzownych w procesie nauczania-uczenia się.

Z elementów **klasycznych** ważna jest nie tylko skupiająca uwagę narracja, z prawidłowym doborem przekazywanych treści, ale też kluczowa jest dynamika przekazu i odpowiednie akcentowanie tego, co najistotniejsze. Wartość w warstwie metodycznej zależy od trafnego rozpoznania poziomu uczniów w danej grupie, od właściwego doboru modułów i sposobów realizacji konspektów-scenariuszy, od jakości sprzężenia zwrotnego i od wcześniejszej autorefleksji trenerów, którzy na podstawie własnych doświadczeń doskonalą kolejne prowadzone przez siebie zajęcia.

Z elementów **innowacyjnych** istotnym mikroprocesem jest dynamiczne przechodzenie przez kolejne poziomy interakcji, od uwrażliwiającego pobudzenia, poprzez komunikowanie, indagowanie i niewyręczające wsparcie, aż po zanurzenie się w twórczym działaniu. Równoległym mikroprocesem jest formowanie motywacji, począwszy od zainspirowania, zaciekawienia, waloryzacji i stymulacji, aż do osiągnięcia satysfakcji z wykonania zadań. Wiąże się to ściśle z wartością merytoryczną modułów, z przyswajalnością zagadnień i atrakcyjnością zadań, z właściwym zagospodarowaniem czasu, z realizacją celów oraz osiąganiem efektów założonych na daną jednostkę zajęć.

Jakość realizacji aspektów dydaktycznych i innowacji metodycznych

- **Kanony narracji:**
  wprowadzenie → ukazanie zamierzeń → przekaz treści → trafne akcentowanie
  Splot wskaźników: DT1, DT2, DT9, DT11, DU1 i OZ4
  Średnia wartość oczekiwana: 4,4

- **Kanony metodyki:**
  rozpoznanie stanu → dostosowanie metod → sprawne wykonanie → refleksja
  Splot wskaźników: DT3, DT7, DU9, DU10, OZ2 i OZ13
  Średnia wartość oczekiwana: 4,4

- **Elementy innowacji:**
  pobudzenie (sensytyzacja) → responding → inguiring → zanurzenie (immersja)
  Splot wskaźników: DT8, DU3, OZ9, OZ10, OZ11 i OZ12
  Średnia wartość oczekiwana: 4,5

- **Elementy motywacji:**
  inspiracja zwiastunem → zaciekawienie → wsparcie → zadowolenie z dzieła
  Splot wskaźników: DT4, DT5, DT6, DU2, OZ14 i OZ1
  Średnia wartość oczekiwana: 4,5

- **Elementy efektywności:**
  dobór właściwych treści → gospodarowanie czasem → osiąganie efektów
  Splot wskaźników: DU4, OZ5, OZ6, OZ7, OZ8 i OZ15
  Średnia wartość oczekiwana: 4,3

Ewaluacja zmian w świadomości i w postawach uczniów

Głównym celem Strategii SWOI jest formowanie takich zmian w dyspozycjach młodych adolescentów, które ukierunkują lub wzmocnią ich wybory dalszej ścieżki edukacyjnej w pożądanej społecznie dziedzinie infotechniki. Chodzi o **przemiany mentalne** w świadomości uczniów, w poznawczych i emocjonalnych komponentach postaw, a szczególnie w cechach wolicjonalnych (chęć działania) i predylekcyjnych (skłonności i upodobania).

Te struktury umysłu mają charakterystyczną właściwość względnej stałości, co oznacza, że nie poddają się łatwo oddziaływaniom modyfikującym, lecz jeśli już uda się zmienić je w pożądanym kierunku, to wykazują znaczną trwałość. Dzięki temu stanowią lepsze wskaźniki pośrednie „zalążków" formowania ukierunkowanych kompetencji aniżeli testy wiedzy. W błyskawicznie rozwijającej się dziedzinie IT treści nauczania szybko tracą aktualność i wystarczalność, zatem chwilowe ukształtowanie i doraźne zweryfikowanie wiedzy jest mniej wartościowe w perspektywie rozwoju ucznia niż formowanie pozytywnych cech osobowości, zwłaszcza woli i dyspozycji do samokształcenia ustawicznego. Kształtowanie cech kierunkowych jest warunkiem wytrwałości w długofalowym nabywaniu i doskonaleniu wysoko cenionych i pożądanych przez pracodawców kompetencji.

W ewaluacji efektów kół zainteresowań nie mierzy się znacznego przyrostu wiedzy, lecz bada subtelne, trwałe zmiany osobowości uczniów.

Przedstawione założenia co do specyficznej materii podlegającej formowaniu mają fundamentalne znaczenie **w ewaluacji efektów** innowacyjnej Strategii edukacyjnej. Celowo podczas badań nie mierzy się wiedzy, gdyż jej przyrost jest znaczny, tym bardziej, że większość uczniów startuje z zupełnie zerowego poziomu w obszarach programowania i mechatroniki. Tak dynamiczne zmiany tłumiłyby możliwość rzetelnej ewaluacji innych ważnych, lecz bardzo subtelnych wskaźników złożonego procesu edukacyjnego. Zresztą przyrosty wiedzy praktycznej i umiejętności obserwuje się bezpośrednio na kolejnych zajęciach, o czym świadczą aktywności uczniów: wykonywanie stawianych zadań i konstruowanie działających implementacji.

Metodologia badania efektów w *Ewaluacji splotowej* osadzona jest na pomiarach niewielkich zmian wskaźników względnie trwałych, lecz również takich, które zgodnie z wiedzą naukową mają tendencje do okresowych spadków (np. słabnie motywacja po zadowalającym ucznia zaspokojeniu aspiracji). W konsekwencji nie można oczekiwać spektakularnych przyrostów, lecz tylko takich, jakie są wyznaczone empirycznie jako realne normy za pomocą standaryzowanej Skali pomiarowej.

Zmiany poziomów muszą być analizowane w kontekście specyfiki danego wskaźnika, uwarunkowań czasoprzestrzennych i środowiskowych oraz wzajemnych powiązań między stanami na wejściu i wyjściu a wartościami oczeki-

wanymi jako *standardy ewaluatywne*. Przedstawione w dalszej części skalowane narzędzie pomiarowe i standardy oczekiwań wypracowano w cyklicznych badaniach na próbach o dużej liczebności N > 1000 uczniów. Odniesienia do standardów są potrzebne tym bardziej, że uczestnictwo w kole zainteresowań pobudza silne stany emocjonalne (afekt), co powoduje osiąganie wysokich poziomów już w pierwszym pomiarze i ogranicza możliwość dalszego przyrostu ze względu na kres skali. Wszystko to sprawia, że do rzetelnych analiz efektów w badaniach dystansowych niezbędne jest zastosowanie odpowiednich strategii pomiarowych i wartościowanie względem norm bazowych na początku i oczekiwanych na zakończeniu cyklu zajęć.

Wartościowanie efektów lokalnych odbywa się poprzez odniesienie do standardów charakterystycznych dla populacji.

Strategie ewaluacji splotowej

W systemowej ewaluacji wykorzystuje się różne źródła i dostępne zasoby empiryczne, w tym także pochodzące z obserwacji. Jednak sama analiza dostrzeganych zjawisk byłaby niepełna, gdyby pominięto nieobserwowalne stany i cechy uczniów, będących podmiotami edukacji. Dlatego niezbędne jest zastosowanie strategii służących pomiarom tego, co odczuwają uczestnicy zajęć i co zmieniło się w ich umysłach. Do tego celu wykorzystuje się kompleksową metodologię, nazwaną *ewaluacją splotową*, celowo wypracowaną na potrzeby szacowania efektów realizacji zajęć z obszarów infotechniki. Jej istotą jest wartościowanie podmiotowe, dystansowe i deklaracyjne, z indeksowym, różnicowym i dwuważonym wartościowaniem rezultatów za pomocą sześciu strategii:

- **Strategia demokratyczna** – wartościowanie podmiotowe
  Uczniowie powinni mieć autentyczny, większy udział w wyrażaniu swych doznań i ocenianiu procesów kształcenia specjalistycznego. Dlatego opinie uczestników kół zainteresowań, poszerzone o odniesienia do kontekstów, stają się tu głównym materiałem empirycznym do analizy wartościującej proces edukacyjny. W taktyce demokratycznej uznaje się, że to właśnie beneficjenci edukacji (a nie eksperci ani obserwatorzy) są najbardziej odpowiednimi respondentami do orzekania, co jest rzeczywistą wartością w ich rówieśniczym kręgu kulturowo-społecznym. Uśrednione poziomy wypowiedzi licznych reprezentantów populacji, będących podmiotami podczas testowania Programu nauczania-uczenia się, wytyczyły *standardy*, jakich należy oczekiwać od kolejnych grup. Do tych wartości normatywnych trener może odnosić rezultaty uzyskane w swoim środowisku.

Strategia demokratyczna oznacza, że wartościowanie odnoszone jest do opinii uczestników kół, a nie ekspertów.

- **Strategia badań panelowych** – wartościowanie dystansowe
  W ewaluacji procesu kształcenia kluczowe znaczenie ma wartościowanie przemian, jakie zachodzą u uczniów pod wpływem oddziaływań edukacyjnych. Celowe oddziaływanie innowacją i mierzenie efektów w warunkach naturalnych jest formą quasi-eksperymentu. W zalecanej tu *metodzie panelowej* zbadane na wstępie grupy uczniów zostają poddane powtórnym pomiarom tym samym narzędziem, w odstępie czasu równym realizacji pełnego cyklu zajęć. „Panel”

Strategia panelowa polega na badaniu tych samych grup identycznym narzędziem po zakończeniu oddziaływań.

w metrologii oznacza jednorodną środowiskowo próbę badawczą, tj. klasę, grupę lub ogół badanych, wybranych z konkretnej zbiorowości. Uzupełnieniem wartościowania dystansowego są *szeregi czasowe*, kiedy to bada się w kolejnych latach inne grupy jako kolejnych reprezentantów tej samej populacji np. gimnazjalistów, poddawanych analogicznym procesom edukacyjnym.

Strategia psychometryczna oparta jest na pośrednim badaniu stanów umysłu poprzez deklaracje w skalach pomiarowych.

» **Strategia psychometryczna** – wartościowanie deklaracyjne
W ocenianiu efektów procesu edukacyjnego należy oszacowywać zmiany, jakie zaszły w umysłach podmiotów tegoż procesu. Jednak w ewaluacji splotowej nie bada się wiedzy ani umiejętności. Jej celem jest pomiar tzw. *cech miękkich*, które uzupełniają kompetencje merytoryczne. W ogólności bada się: świadomość przedmiotową, postawy wobec infotechniki, cechy mentalne i wolicjonalne. W szczególności są to różne komponenty poznawcze (kognitywne) i doznaniowe (afektywne), takie jak: opinie, poglądy, rozwaga, zdolność, dyspozycje, emocje, motywacje, ambicje itp. Do pomiarów tych stanów umysłu wykorzystuje się deklaracje wyrażane w wystandaryzowanym narzędziu psychometrycznym. Wypowiedzi wynikające z autorefleksji uczniów uczestniczących w kołach zainteresowań są w ewaluacji zajęć najbardziej cenne.

Strategia agregacyjna to łączenie wskaźników i przypadków w celu zapewnienia wyższej wiarygodności wyników.

» **Strategia agregacyjna** – wartościowanie indeksowe
Dla zapewnienia trafności i rzetelności ewaluacji konieczne jest wyznaczenie właściwego zbioru indeksów, będących pośrednimi miernikami badanych stanów. Są to elementarne pojęcia, którym nadaje się empiryczny sens zmiennych w formie wskaźników współtworzących złożoną, trudno mierzalną właściwość. *Agregacja zmiennych* polega na scalaniu pojedynczych wskaźników w zbiory ogólniejsze. Ze zmiennych elementarnych buduje się zmienne cząstkowe, splotowe i globalne. Umożliwia to analizę składników, czynników i wyników ogólnych. W ewaluacji splotowej pomiary psychometryczne nie służą jednakże do oceny pojedynczych uczniów. Zbiory danych indywidualnych łączone są po to, aby ustalić zbiorcze cechy prób badawczych, a zatem *agregacja przypadków* polega na obliczaniu statystyk dla klas, grup i dla ogółu badanych.

Strategia komparacyjna to stosowanie różnych porównań, np. grup, zmiany stanów, odniesienia do norm.

» **Strategia komparacyjna** – wartościowanie różnicowe
W ewaluacji efektów potrzebne są dwa pomiary – ‚przed' i ‚po' zrealizowaniu pełnego cyklu kształcenia. Celem jest kompleksowa ocena różnic, przyczyn i rodzajów przemian, jakie udało się wywołać w umysłach uczących się. Analizuje się czynniki wpływające na różne przyrosty w kontekście lokalnych uwarunkowań. Wykrywa się zjawiska pozytywne i negatywne oraz formułuje zalecenia optymalizacyjne. O wartości elementów procesu kształcenia świadczą statystyczne różnice poziomów między stanem końcowym i początkowym. *Komparacja* oznacza także porównywanie efektów osiąganych przez różne grupy, poddawane tym samym procesom kształcenia.

W bardziej zaawansowanych metodach statystycznych można analizować m.in.: harmonię i dynamikę zmian, spójność i zbieżność opinii respondentów, trafność i rzetelność badań.

» **Strategia pomiarów skalowanych** – wartościowanie dwuważone Na potrzeby ewaluacji splotowej opracowano specjalne narzędzie pomiarowe, zwane *Skalą dwuważonych ocen*. Tworzenie Skali wymaga statystycznego zweryfikowania ścisłych kryteriów doboru wskaźników. W przyjętym tu modelu wartościowania wykorzystuje się zmodyfikowane opcje Skali Likerta. Jednakże – w odróżnieniu od pierwowzoru, gdzie głównym kryterium jest zdolność do różnicowania respondentów – tutaj kluczowe jest kryterium konkluzyjności (mocy wnioskotwórczej). Pozycje w Skali są elementarnymi tezami, które respondenci akceptują lub negują z większym lub mniejszym natężeniem. Każda pozycja kwestionariusza testuje konkretny indeks jakościowy, a poprzez kalibrację wyznaczany jest poziom ilościowy. Uzyskuje się dane liczbowe, określające dwa wymiary: wartości wypowiedzi (stopnia zgodności z większością opinii) oraz intensywności wypowiedzi (stopnia przekonania respondentów co do swych racji).

Strategia skalowania oznacza zastosowanie wystandaryzowanych i unormowanych narzędzi pomiarowych.

Praktyka ewaluacji efektów zajęć

W ewaluacji procesów edukacyjnych najważniejsze jest ustalenie efektów w środowisku lokalnym, zatem obszarem pomiarów jest grupa podmiotów uczestniczących w konkretnym kształceniu. Ażeby mieć z czym porównywać efekty, warto pozyskiwać dane empiryczne z podobnych środowisk lub kursów (własne lub od innych badaczy). Na potrzeby analiz porównawczych przeprowadza się pomiary wzdłużne kolejnych roczników, a także zestawia rezultaty kilku grup zajęciowych, poddanych analogicznemu procesowi kształcenia. Dla ułatwienia pomiarów i odniesień, polecamy standaryzowaną *Ankietę ewaluacyjną* wraz z wartościami normatywnymi i schematem interpretacji.

W sytuacji, gdy nauczyciel dysponuje dedykowaną skalą do wartościowania miękkich efektów kształcenia, czynności badawcze sprowadzają się do pomiarów, przetworzenia danych i analizy rezultatów. Zasadą jest przeprowadzenie pierwszego pomiaru na samym początku pierwszych zajęć i powtórzenie pomiaru na ostatnim spotkaniu. Obydwa pomiary muszą objąć tych samych uczniów. Minimalny dystans czasu między pomiarami to 3-miesięczny cykl cotygodniowych zajęć po 2 godziny lekcyjne, a więc łącznie co najmniej 24 godziny nauczania-uczenia się. Zalecanym sposobem zbierania danych jest wypełnianie kwestionariuszy w formie drukowanej. Przed odbiorem ankiet trener prosi uczniów o sprawdzenie kompletności wypełnień. Bardzo ważne jest oznakowywanie ankiet kodami w taki sposób, ażeby do analiz możliwe było ułożenie parami obu kwestionariuszy (‘przed’ i ‘po’) tych samych uczniów.

Badanie efektów zajęć polega na dwukrotnym pomiarze dedykowanym narzędziem i na analizie zmian.

Wskaźnikami są tu wypowiedzi uczniów, wyrażające stopień akceptacji lub negacji stwierdzeń zawartych w kwestionariuszu.

Podstawowymi elementami w skali pomiarowej są *wskaźniki*. Każdy ze wskaźników jest próbkowany pojedynczą pozycją w kwestionariuszu. Pozycje te składają się ze specjalnie dobranych stwierdzeń i z opcji wypowiedzi do wyboru. W naszym modelu zamiast pytań stosujemy stwierdzenia wyrażające elementarne tezy badawcze. Sens merytoryczny wynika wprost z treści, jaką odczytuje respondent. Sformułowania muszą być jednoznacznie rozumiane przez uczniów i emocjonalnie nieobojętne. Za ich pomocą pobudzamy respondentów do wypowiadania się wobec danej kwestii. Część tez celowo jest „nieprawdziwa", aby ograniczyć automatyzm udzielania wypowiedzi.

Istotą budowania rzetelnej Skali jest dobór stwierdzeń, co do których większość się zgadza, ale też opinie są odpowiednio zróżnicowane.

W doborze optymalnych tez do skali ważna jest nie tylko treść zagadnień badawczych, lecz także *konkluzyjność wskaźników*, tj. przydatność do formułowania istotnych wniosków na bazie wyznaczenia „złotego środka" między wymaganą zgodnością co do tez, a niezbędnym zróżnicowaniem wypowiedzi. Uczniowie różnie reagują na stwierdzenia zawarte w kwestionariuszu i ustosunkowują się do nich. Większościowa akceptacja lub negacja stwierdzeń jest empiryczną weryfikacją prawdziwości bądź fałszywości tez i stanowi podstawę do ustalania polaryzacji wskaźników. Podczas testowania i optymalizacji Ankiety ewaluacyjnej sprawdzono jakość pomiarową 35 różnych wskaźników, pozostawiając finalnie 24 pozycje o najlepszych właściwościach wnioskotwórczych.

W skalach standaryzowanych wypowiedzi są zunifikowane, ograniczone do zamkniętych opcji wyboru. Kolejne opcje wyznaczają narastającą wartość – od zaprzeczenia do zatwierdzenia tezy. W *Skali dwuważonych ocen* stosuje się jednolitą, siedmiostopniową kafeterię możliwych wypowiedzi:

absolutnie nie | nie | raczej nie | brak zdania | raczej tak | tak | absolutnie tak

Wybory opcji przez respondentów wskazują pośrednio, na jakich poziomach znajdują się ich wewnętrzne cechy i stany umysłu. Wskaźnikom nie nadaje się nazw własnych, ani nie definiuje jako pojęć abstrakcyjnych. Natomiast na potrzeby analiz statystycznych przypisuje się im wartości liczbowe (wagi). Po przypisaniu opcjom wypowiedzi odpowiednich wag otrzymujemy dane ilościowe, które można przetwarzać statystycznie, głównie pod kątem oceny rezultatów.

Ankieta ewaluacyjna dla uczniów

Kod ucznia ..... Data .....

Prosimy o rzetelne wyrażenie własnych odczuć wobec wszystkich stwierdzeń!
Zakreślaj wyraźnie swoje wybory wypowiedzi

1. Odczuwam pewną niechęć do wszechobecnej dziś technicyzacji.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
2. Tylko uzdolnieni technicznie skorzystają z nowoczesnych technologii.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
3. Wątpię w przydatność dodatkowych zajęć pozalekcyjnych.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
4. Zajęcia dodatkowe są o wiele ciekawsze niż lekcje szkolne.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
5. W szkole komputer winien być używany tylko na lekcji Informatyki.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
6. Mam świadomość potrzeby uczenia się informatyczno-technicznego.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
7. Uważam, że komputery są użyteczne w każdej dziedzinie.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
8. W mojej przyszłej pracy komputer będzie zbędny.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
9. Komputer w moim przypadku jest niestety złodziejem czasu.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
10. Odczuwam na sobie rodzaj zniewolenia przez gry lub Internet.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
11. Czuję się niepewnie, gdy mam samodzielnie wykonać coś na komputerze.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
12. Źle mi w grupie, gdy inni lepiej znają język programowania.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
13. Praca z mikrokontrolerem wymaga dużej koncentracji uwagi.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
14. Nowo zakupionego robota można używać bez czytania instrukcji.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
15. Robot może radzić, jak rozwiązywać życiowe problemy.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
16. Można złożyć moduł interfejsu już po krótkim poinstruowaniu.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
17. Komputer tylko w małym stopniu zaspokaja moje aspiracje twórcze.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
18. Zawsze wyrażam gotowość uczenia się trudnych rzeczy.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
19. W przyszłości umiejętności informatyczne pomogą mi znaleźć pracę.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
20. Praca mechatronika może być tak przyjemna jak zabawa.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
21. Umiem poradzić sobie z montowaniem układu elektronicznego.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
22. Uważam, że zrozumienie algorytmów przekracza moje zdolności.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
23. Czuję, że dał(a)bym sobie radę na studiach informatycznych.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak
24. Sądzę, że po maturze mogę podjąć studia na politechnice.
absolutnie nie nie raczej nie brak zdania raczej tak tak absolutnie tak

Sprawdź uważnie, czy nie zostały pominięte jakieś wypowiedzi.
Dziękujemy za udział w badaniach Kultury infotechnicznej. Życzymy satysfakcji z działań w Społeczności e-Swoi!

Skala wyznacza dwa wymiary, tj. jakość i intensywność wypowiedzi, czyli słuszność co do wyboru i zdecydowanie co do słuszności.

Wartościowanie i agregacja wskaźników

W fazie kwantyzacji kolejnym progom natężenia cech jakościowych przypisuje się wagi ilościowe. Ze względu na to, że wynikające z pomiarów progi natężeń są umowne, najprościej jest przyjąć liniową zależność między stanem cech a ich wagą. W praktyce najbardziej pozytywnemu stanowi cechy nadaje się najwyższą wagę dodatnią, a stanowi skrajnie negatywnemu – najniższą. Odstępy między progami stanowią jednostki miar. Wyznaczona w taki sposób wartość cech jest zależna od jakości wypowiedzi respondentów. Jakość ta jest tym wyższa, im wyższa jest zgodność danej wypowiedzi z opinią większości co do tezy zawartej w bodźcu-stwierdzeniu. Pewność siebie w wypełnianiu ankiet po części zależy jednak od osobowości uczniów, gdyż są osoby, które z natury nie wybierają opcji skrajnych 'absolutnie...'. Wymiar intensywności wskazuje, na ile ważkie dla badań były wypowiedzi respondentów. Z kolei wybory opcji 'brak zdania' nie są przydatne poznawczo, gdyż mogą być celowym unikaniem wypowiedzi, mimo faktycznej znajomości problemu.

Pamiętajmy, ze część stwierdzeń celowo sformułowano jako nieprawdziwe, aby zapobiec schematyzmowi wypełniania Ankiety. Dlatego konieczna jest depolaryzacja.

Część stwierdzeń w Ankiecie celowo sformułowano jako tezy nieprawdziwe, dlatego w przypisywaniu wag należy zwrócić uwagę na polaryzację wskaźnika i adekwatną kalibrację. Polaryzacja wynika z prawdziwości bądź fałszywości bodźca-stwierdzenia, co zostało ustalone przez twórcę Skali i potwierdzone empirycznie przez kwalifikowaną większość respondentów. Przyjmuje się, że zdepolaryzowane i skalibrowane według poniższego schematu dane szacunkowe z Ankiety ewaluacyjnej są addytywne, tj. upoważniają do kumulowania i do obliczania wartości średnich.

Kalibracja narastająca -0,5.../...5,5 Wagi przypisane wypowiedziom:

-0,5	0,5	1,5	2,5	3,5	4,5	5,5
absolutnie nie	nie	raczej nie chyba nie	brak zdania	raczej tak chyba tak	tak	absolutnie tak

Kalibracja odwrócona 5,5...\...-0,5 Wagi przypisane wypowiedziom:

5,5	4,5	3,5	2,5	1,5	0,5	-0,5
absolutnie nie	nie	raczej nie chyba nie	brak zdania	raczej tak chyba tak	tak	absolutnie tak

Kalibrację neutralnej opcji środkowej osadzono na granicy między wartością

Z czego wynika powyższy sposób kalibracji? Wybór opcji 'brak zdania' nie jest ani akceptacją, ani negacją tezy. Stan ten na uczelniach odpowiada progowi między *niedostatecznym* a *dostatecznym* poziomem wypowiedzi, gdy jest dylemat: zaliczyć czy nie, stąd przypisana wartość progowa 2,5. Z kolei wartość maksymalna 5,5 ma uzasadnienie w tym, że wokół ocen głównych występują odchylenia ± pół działki. Skutkuje

to także dalszym podziałem skali na przedziały o progach 3,5 i 4,5 (odpowiedniki ocen z plusem). W praktyce badań Skalą dwuważoną, po agregacji wskaźników i uśrednieniu rezultatów grupowych, wyniki poniżej wartości 2 i powyżej 5 nie występują. W tabeli przedstawiono Schemat kalibracji, wartości normatywne i strukturę łączenia wskaźników w zmienne wyższego rzędu, zwane składnikami i czynnikami.

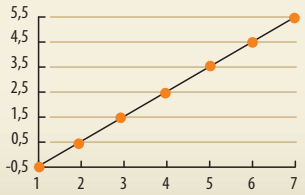
pozytywną a negatywną, tj. 2,5 w skali ocen akademickich.

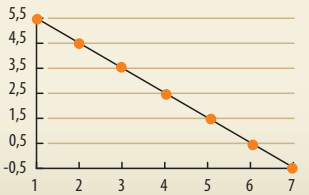
Tabela 4.

Schemat kalibracji i scalania wskaźników z Ankiety ewaluacyjnej

Numer wskaźnika	Rodzaj kalibracji	Średnia wartość oczekiwana na wejściu	Średnia wartość oczekiwana na wyjściu	Nazwa składnika	Nazwa czynnika	Kategoria czynnika
1	odwrócona: 5,5…\…-0,5	4,2	4,3	*Aprobata*	*Opinie*	doznaniowy
2	odwrócona: 5,5…\…-0,5					
3	odwrócona: 5,5…\…-0,5	4,4	4,5	*Ocena*		
4	narastająca: -0,5…/…5,5					
5	odwrócona: 5,5…\…-0,5	4,3	4,4	*Wgląd*	*Poglądy*	poznawczy
6	narastająca: -0,5…/…5,5					
7	narastająca: -0,5…/…5,5	4,1	4,2	*Osąd*		
8	odwrócona: 5,5…\…-0,5					
9	odwrócona: 5,5…\…-0,5	3,4	3,5	*Odpór*	*Emocje*	doznaniowy
10	odwrócona: 5,5…\…-0,5					
11	odwrócona: 5,5…\…-0,5	3,6	3,8	*Spokój*		
12	odwrócona: 5,5…\…-0,5					
13	narastająca: -0,5…/…5,5	3,5	3,6	*Ogłada*	*Rozwaga*	poznawczy
14	odwrócona: 5,5…\…-0,5					
15	odwrócona: 5,5…\…-0,5	3,1	3,5	*Refleksja*		
16	narastająca: -0,5…/…5,5					
17	odwrócona: 5,5…\…-0,5	3,3	3,5	*Ambicje*	*Motywacje*	doznaniowy
18	narastająca: -0,5…/…5,5					
19	narastająca: -0,5…/…5,5	4,2	4,3	*Intencje*		
20	narastająca: -0,5…/…5,5					
21	narastająca: -0,5…/…5,5	3,2	4,1	*Pewność*	*Zdolność*	poznawczy
22	odwrócona: 5,5…\…-0,5					
23	narastająca: -0,5…/…5,5	3,6	3,9	*Potencjał*		
24	narastająca: -0,5…/…5,5					

Prostym sposobem wprowadzania danych jest przypisanie opcjom wypowiedzi kolejnych liczb od 1 do 7, a następnie zautomatyzowana konwersja na wagi prowadzące do skali ocen akademickich (ryc.).





Podczas analiz należy uwzględniać właściwości badanych cech osobowościowych, m.in. to, że zmieniają się one nieznacznie.

Dzięki agregacji zmiennych elementarnych uzyskuje się coraz bardziej trafne rezultaty pomiarów. Przyjęto, że *składniki* to średnia z dwóch wskaźników, a *czynniki* – z czterech. Składniki służą do szczegółowych analiz jakości formowania cech podstawowych, ważnych w dziedzinie Infotechniki, a czynniki wyznaczają kluczowe kategorie psychopedagogiczne. Rezultaty połówkowe oblicza się z 12 wskaźników doznaniowych i 12 poznawczych. Istotą optymalnego procesu edukacyjnego jest równoważenie obu tych sfer: afektywnej i kognitywnej. Po scaleniu wszystkich wskaźników uzyskuje się wynik ogólny, będący najbardziej wiarygodnym rezultatem ewaluacji splotowej.

Dla ułatwienia interpretacji wyników ewaluacji i w celu uproszczenia czynności porównawczych, zmiennym wyższego rzędu nadano krótkie nazwy, wyrażające ich sens merytoryczny:

- **Afektywno-afirmacyjne składniki *Opinii:***
  Aprobata – wyraz emocjonalnej akceptacji nowych technologii ITC i powszechnej technicyzacji.
  Ocena – stopień akceptacji zajęć realizowanych w formie kół zainteresowań infotechnicznych.

- **Świadomościowe składniki *Poglądów:***
  Wgląd – trafność oceny roli nabywania kompetencji IT i znaczenia pozalekcyjnych form edukacji.
  Osąd – uznanie przydatności szerokich zastosowań komputera i niezbędności w pracy zawodowej.

- **Składniki dotyczące samokontroli *Emocji:***
  Odpór – odporność na uzależnienie się od komputera lub Internetu i marnotrawienie czasu.
  Spokój – brak stresu przed działaniami samodzielnymi i dyskomfortu, że inni w grupie są lepsi.

- **Racjonalnościowe składniki *Rozwagi:***
  Ogłada – roztropność przy pracy z układami elektronicznymi oraz uznanie ważnej roli instrukcji.

Refleksja – trafność rozpoznania przydatności i możliwości konstruowania modułów-interfejsów.

- **Predylekcyjno-wolicjonalne składniki *Motywacji:***
  Ambicje – chęć uczenia się programowania i gotowość do twórczych działań przy komputerze.
  Intencje – zamiary co do wyboru ścieżki rozwoju i przyszłej pracy w dziedzinie infotechnicznej.

- **Predyspozycyjne składniki *Zdolności:***
  Pewność – samoocena poziomu rozumienia algorytmów i radzenia sobie z montażem układów.
  Potencjał – samoocena dyspozycji do przyszłego studiowania politechnicznego, informatycznego.

Efekty zmierzone Ankietą ewaluacyjną wyrażają trwałe, pożądane przemiany w świadomości i w postawach uczniów.

W splotowej ewaluacji efektów kształcenia kluczowe jest ustalenie zachodzących zmian. W przyjętej metodologii pomija się znaczne przyrosty wiedzy, a mierzy cechy mentalne i wolicjonalne, trudno poddające się zmianom. Oczekuje się, że nastąpią choćby minimalne przyrosty. Ze względu na ich względną stałość (co jest widoczne w tabeli jako zmiany rzędu 0,1), pomiędzy pomiarami należy zachować wystarczający dystans czasu na oddziaływania formujące. Skala jest czuła na niewielkie zmiany, lecz pewne poziomy cech wcale już nie muszą się zmieniać pod wpływem, np. aprobata nowych technologii. W takich przypadkach ważne jest, aby nie następowały spadki wartości czynników zagregowanych. Natomiast wynik ogólny w lokalnych środowiskach po uśrednieniu wszystkich wskaźników powinien wzrosnąć przynajmniej o 0,2 stopnia.

Rzetelność wartościowania i trafność wnioskowania

Na jakość każdej z osobna jednostki zajęć dydaktycznych wpływa splot bardzo wielu czynników. Dużo zależy od doboru treści, lecz jeszcze więcej od metod realizacji i od percepcji uczniów. Ze względu na silne zróżnicowanie odbiorców Strategii, nie da się opracować jakichś uniwersalnych, szczegółowych scenopisów zajęć. Proponowane w Programie nauczania-uczenia się konspekty i opisy implementacji wyznaczają jedynie kanwę tego, co musi być przetworzone na niezwykle złożony spektakl. W spektaklu tym trenerzy są nie tylko aktorami, lecz przede wszystkim reżyserami i współautorami. Uczniowie nie mogą być wyłącznie widzami bądź statystami, lecz w dużej mierze są współtwórcami całościowego dzieła. Wreszcie wszyscy współuczestnicy, w tym także obserwatorzy recenzujący spektakl, winni wykazać się obiektywnością, starannością i odpowiedzialnością w ocenianiu i wyrażaniu opinii na potrzeby doskonalenia procesów edukacyjnych.

Na efekty edukacyjne i ich wartościowanie silny wpływ mają uwarunkowania środowiskowe:

- **kadrowe** – kompetencje trenerów, ale także nauczycieli pełniących rolę obserwatorów,
- **podmiotowe** – predyspozycje uczniów, uczestników konkretnej grupy zajęciowej,
- **organizacyjne** – optymalne liczebności grup i czas realizacji zajęć pozalekcyjnych,
- **sprzętowe** – jakość wykorzystywanych urządzeń i kompatybilność oprogramowania.

Jakość realizacji i oceny kół zainteresowań zależy od kompetencji kadr, a właśnie Strategia umożliwia głębokie doskonalenie zawodowe.

Wpływają one nie tylko na jakość zajęć, lecz także na jakość samej ewaluacji. Wśród uwarunkowań kadrowych warto wrócić uwagę na to, że specjaliści przedmiotowi, nauczyciele informatyki lub mechatroniki, mogą mieć niewystarczające kompetencje psychopedagogiczne. Po części jest to skutkiem innych priorytetów na studiach politechnicznych, a po części wynika z realizacji skróconych kursów uzupełniających przygotowanie pedagogiczne wymagane do pracy w szkole. W rezultacie trudniej jest im trafnie oceniać elementy dydaktyczne i metodyczne. Fakt ten nie umniejsza jednak korzyści z hospitacji i ewaluacji, a wręcz przeciwnie – wzajemna wymiana wniosków z obserwacji i autorefleksji stanowi cenne źródło doskonalenia tych kompetencji pedagogicznych.

Przeprowadzanie obserwacji i ewaluacji stwarza nauczycielom szansę na doskonalenie umiejętności trafnego oceniania i wartościowania.

Hospitacje koleżeńskie obarczone są nie tyle życzliwym zawyżaniem ocen, co raczej zbyt mało wnikliwym oszacowywaniem wskaźników. W rzeczywistych procesach edukacyjnych zachodzą tak złożone sytuacje i wielorakie uwarunkowania, że siłą rzeczy nie wszystko może być „optymalne". Wprawdzie kategoria ta jest ujęciem względnym, odnoszonym w znacznym stopniu do konkretnych realiów i potencjalnych możliwości, lecz mimo to trudno w praktyce o ideał zajęć. Z tego względu nazbyt liczne, bezkrytyczne wypełnianie w kwestionariuszu opcji „optymalne" daje rezultaty mało wnioskotwórcze. W konsekwencji są one mniej przydatne do oceny zajęć niż wybory sąsiednich opcji, odchylonych nieco w lewo lub w prawo od pozycji środkowych na skali. Podczas wartościowania zjawisk, choćby nawet tylko subiektywne odczucia obserwatora (że czegoś było trochę mniej lub więcej od optimum), jeśli zostaną wyrażone w Arkuszu, to stają się właściwą postawą konstruktywnej dyskusji i trafniejszego wnioskowania.

Analogicznie z jakością wypowiedzi uczniów – im bardziej są zróżnicowane a jednocześnie wewnętrznie spójne, tym większa jest ich przydatność w optymalizacji oddziaływań. W praktyce pomiarowej zdarzają się braki danych lub przypadki automatyzmu w wypełnianiu kwestionariuszy przez uczniów. Do analiz statystycznych włącza się tylko dane kompletne, stanowiące ułożone parami wypowiedzi tych samych respondentów z początku i końca zajęć. Nie należy wykorzystywać kwestionariuszy, w których są wypowiedzi nierzetelne (np. automatyczne wybory jednej opcji 'tak') lub odmowne (np. liczne serie opcji 'brak zdania'). Przyczyny takich przypadków warto ustalać indywidualnie, natomiast nie wolno tych danych włączać do analiz statystycznych, gdyż mimo uśredniania wypaczałyby one obiektywną ocenę jakości zajęć.

Pamiętajmy, że rolą Kół zainteresowań IT jest m.in. umożliwienie uczniom bezpośredniego doświadczenia owej trudnej do przyswojenia materii, jaką są umiejętności

programistyczne i mechatroniczne. Doświadczenie to ma umożliwić uczestnikom trafną samoocenę dyspozycji i podjęcie właściwej drogi wyboru ścieżki dalszego kształcenia. Program nauczania-uczenia się nie jest więc nastawiony na zachęcającą psychomanipulację, lecz na twardą praktykę umożliwiającą weryfikację osobistego potencjału. W konsekwencji – zdarzające się przypadki uznania przez ucznia, że jednak specjalizacja IT nie jest dla niego, nie mogą być traktowane jako wskaźniki gorszej jakości zajęć. Do oszacowania edukacyjnego potencjału konkretnej grupy uczniów w odniesieniu do norm populacyjnych gotowości do uczenia się służy *wskaźnik buforowy* (poz. 18 z Ankiety ewaluacyjnej).

Celem oceniania kół zainteresowań i ewaluacji formowania cech nie jest wystawianie ocen, lecz ulepszanie oddziaływań i pomiar skutków.

Istotne uwarunkowania organizacyjne dotyczą liczebności grup zajęciowych. Metodyka realizacji kół zainteresowań IT wymusza pracę z małą grupą, a w dużej mierze także pracę zindywidualizowaną. Każdy uczeń może wymagać odrębnego podejścia w interakcji, dlatego grupa laboratoryjna nie powinna być liczniejsza niż 12 osób. Równie ważna ze względu na stopień trudności jest pora dnia, w jakiej odbywają się koła. Zwykle są to godziny popołudniowe, kiedy zmęczonym uczniom trudniej jest o niezbędną koncentrację. Także niekorzystnym czynnikiem może być usytuowanie kół w takim okresie, gdy uczniowie muszą realizować inne ważne aktywności (np. podczas przygotowań do egzaminów zewnętrznych). Ustalenie optymalnych terminów realizacji kół ma znaczący wpływ na wskaźniki oceny zajęć i na uzyskane efekty.

Koła zainteresowań muszą odbywać się w małych grupach, aby możliwa była realizacja wszystkich elementów innowacji metodycznych.

Opinie uczniów, trenerów i obserwatorów o jakości zajęć silnie zależą od uwarunkowań sprzętowych. Chodzi zarówno o sprawne funkcjonowanie komputerów i sieci w pracowni, jak też o niezawodność urządzeń peryferyjnych, w tym głównie nośnika z dedykowanym systemem *Szkolnego Remiksu Ubuntu*. Ze względu na różnorodność i przestarzałość aparatury w szkołach, pojawiające się trudności z uruchomieniem aplikacji lub awarie powodują bardzo niekorzystne obniżanie wielu wskaźników wartościowania zajęć. Takie przypadki można zdecydowanie ograniczyć poprzez zainstalowanie systemu i całego pakietu oprogramowania na dyskach stacjonarnych. Jeśli jednak zdarzy się awaria, to w praktyce wskaźniki z obserwacji tak zakłóconego procesu stają się bezużyteczne, a w ewaluacji, która obejmuje cały cykl zajęć, rezultaty z pewnością będą zaniżone.

Realizację kół zainteresowań infotechnicznych ułatwia zainstalowanie na dyskach dedykowanego systemu i narzędzi Szkolnego Remiksu Ubuntu.

Systematyczne przeprowadzanie obserwacji i pomiarów skalowanych jest niezwykle istotnym elementem wartościowania oddziaływań edukacyjnych i ich efektów. Rzetelne dane z prawidłowo wykonanych hospitacji oraz trafnie wyciągnięte wnioski z ewaluacji są bezcenne dla nauczycieli prowadzących innowacyjne koła zainteresowań IT. Służą do porównań własnych refleksji jako trenerów ze spostrzeżeniami obserwatorów i z odczuciami uczniów, co w konsekwencji umożliwia doskonalenie tak bardzo złożonych procesów edukacyjnych w kolejnych cyklach.

Bibliografia

» Ubermanowicz S.: *Ewaluacja splotowa InfoKultury – Skala dwuważonych ocen*. Wydawnictwo Naukowe UAM, Poznań 2005

Narzędzia i wytwory
Studium infotechniczne

Studium infotechniczne NARZĘDZIA I WYTWORY jest opisem środowisk i platform służących do realizacji Strategii Wolnych i Otwartych Implementacji oraz opisem instrukcji i narzędzi realizacji Programu nauczania-uczenia się infotechniki. Głównymi środowiskami programistycznymi są scharakteryzowane tu systemy oraz aplikacje Wolnego i otwartego oprogramowania, zgromadzone w pakiecie Szkolnego Remiksu Ubuntu, a środowiskiem konstruowania mechatronicznego jest moduł-interfejs Arduino. Platformę dedykowaną do edukacji pozaszkolnej stanowi internetowy Serwis e-Swoi o przedstawionych tu funkcjach. Narzędziami o ciekawych właściwościach środków-metod dydaktycznych są Implementacje infotechniczne. Ze względu na obszerność dokumentacji i cyfrowy charakter wytworów implementacyjnych, w formie drukowanej zamieszczono tu jedynie 11 przykładowych rozwiązań. Cały pakiet opisów implementacji, kodów źródłowych, instrukcji i schematów znajduje się na dołączonej płycie oraz na Serwisie e-Swoi.

Ta część opracowania przeznaczona jest dla wszystkich odbiorców Strategii SWOI i użytkowników innowacyjnego Programu, tj. dla uczniów zainteresowanych infotechniką, nauczycieli realizujących zajęcia komputerowe lub mechatroniczne, animatorów kształcenia zdalnego, instruktorów organizujących zajęcia w placówkach wychowania pozaszkolnego, metodyków prowadzących kursy doskonalenia nauczycieli informatyki bądź elektroniki, a także dla wszystkich osób, które chcą poznać ogólnodostępne, wolne i otwarte systemy oraz aplikacje użytkowe.

Treści studium metodycznego zgrupowano w rozdziałach:

- **Wolne i otwarte oprogramowanie oraz licencje** – to tekst wyjaśniający zasady korzystania z programów komputerowych oraz aspekty ochrony wynikające z praw autorskich i z licencji.
- **Serwis edukacyjno-społecznościowy e-Swoi** – to opis walorów pracy w społeczności sieciowej oraz charakterystyka funkcji platformy internetowej dedykowanej na potrzeby Strategii.
- **Szkolny Remiks Ubuntu – środowisko pracy** – to instrukcja przygotowania nośnika z WiOO oraz uruchamiania systemu i aplikacji użytkowych, niezbędnych do uczenia się infotechniki.
- **Instrukcja korzystania z modułu-interfejsu Arduino** – to charakterystyka środowiska do ćwiczeń mechatronicznych oraz instrukcja użytkowania i specyfikacja zestawu podzespołów.
- **Implementacje do realizacji na kołach zainteresowań IT** – to opis przykładowych wytworów programistycznych i mechatronicznych jako możliwe rozwiązania zadań uczniowskich.

01

Wolne i otwarte oprogramowanie oraz licencje

Rafał Brzychcy

Wolność w ujęciu społecznym wydaje się być stałym i niezmiennym paradygmatem, a otwartość pożądaną, trwałą postawą. Wraz z szerokim upowszechnieniem technologii informacyjno-komunikacyjnych pojawiły się nowe formy i miejsca wyrażania wspomnianych idei – w przestrzeni cyfrowej, w globalnym świecie splecionym pajęczyną Internetu. Owa przestrzeń dostępna jest dzięki narzędziom teleinformatycznym, które dziś postrzegamy jako wysoce pożyteczne i umożliwiające m.in. partycypację w edukacji. Dlatego na kształt i dostępność tych *stricte* technicznych środków znaczący wpływ mają m.in. ich wartości prospołeczne i walory proedukacyjne, a te na gruncie informatycznym zdefiniował *Ruch wolnego i otwartego oprogramowania*. Wspomniany *Ruch* oraz samo *Wolne i otwarte oprogramowanie* ukształtowały się dzięki połączeniu idei wolności, postaw otwartości oraz technologii informacyjno-komunikacyjnych włącznie.

Monopolizacja a humanizacja informatyki

Zamykanie kodów źródłowych aplikacji komputetowych, z pozostawieniem tzw. luk dostępu do nich, maskuje potwierdzony fakt globalnej inwigilacji użytkowników oprogramowania.

Pierwszy komputer o nazwie ENIAC powstał w 1943 roku. Wówczas i w kolejnych latach kwestia potencjalnych korzyści majątkowych czerpanych z oprogramowania komputerowego nie była istotna – traktowano je jako dodatek, w naturalny sposób związany ze sprzętem. Niegdyś nie budowano modeli biznesowych opartych na kodach źródłowych programów – pojawiły się one wraz z powstaniem firmy Microsoft. Firma ta, na czele z Billem Gatesem, jako pierwsza upowszechniła nieznaną wcześniej praktykę tzw. zamykania kodów źródłowych aplikacji komputerowych, aby z takiego zmonopolizowanego wytworu móc czerpać korzyści majątkowe[1].

Coraz bardziej popularna **monopolizacja oprogramowania** uniemożliwiła swobodne uruchamianie, modyfikację oraz rozpowszechniane

1 Gates B.: *An Open Letter To Hobbyists*, "Homebrew Computer Club Newsletter" 1976, vol. 2, issue 1, p. 2, Mountain View, CA [http://www.digibarn.com/collections/newsletters/homebrew/V2_01/gatesletter.html]

zamkniętego oprogramowania, licencjonowanego tak, aby wymienione czynności utrudnić w imię szybkich korzyści finansowych firm informatycznych. Z założeń ekonomii jednostkowego podmiotu gospodarczego, nastawionego na wysokie i szybkie korzyści pieniężne, taka sytuacja wydaje się optymalna. Jednak rozwój i wzrost gospodarczy to proces złożony, wskazujący na powiązanie jego dynamiki nie tylko z zyskiem ujmowanym jako dodatni wynik finansowy, ale również z tym, ujmowanym jako korzyści społeczne.

Zgodnie z zasadą *humanizacji ekonomii*, sformułowaną we Francji już w latach 40-tych XX wieku i upowszechnianą przez Emmanuela Mouniera, zysk ekonomiczny nie może być wyłącznie finansowy, lecz powinien uwzględniać aspekty społeczne, normowane przez główne zasady etyki życia gospodarczego. Zysk synergiczny, finansowy i zarazem społeczny, jest wyrazem odpowiedzialnej postawy społecznej, a to właśnie warunkuje niezakłócony, zrównoważony rozwój gospodarczy. Kapitał finansowy i społeczna odpowiedzialność za niego, przy uwzględnieniu aspektów moralnych, **powinny się równoważyć**, a zysk należy podporządkować potrzebom człowieka. Zasadę humanizacji ekonomii reprezentują współcześni laureaci Nagrody im. A. Nobla (m.in. Reinhard Selten, Joseph Stiglitz, Daniel Kahneman, Vernon Smith)[2].

Jak każda monopolizacja, tak i monopolizacja oprogramowania stawała w sprzeczności z pełnymi możliwościami prospołecznymi i proedukacyjnymi informatyki. Walory edukacyjne, związane z możliwością analizowania i zmian kodu źródłowego programów, zostały silnie ograniczone. Ponadto obostrzone licencje na oprogramowanie były i są przyczyną pogłębiającego się wykluczenia cyfrowego, związanego przede wszystkim z niekorzystną sytuacją ekonomiczną wielu użytkowników narzędzi cyfrowych. Zamknięte oprogramowanie, używane do nauki i pracy, bywa bowiem wielokrotnie droższe od samego sprzętu elektronicznego, na którym jest uruchamiane. Nie dziwi więc, że dla pewnej równowagi rozwoju gospodarczego, w tym gospodarki opartej na wiedzy, naturalne i konieczne stało się zhumanizowanie informatyki.

Monopolizacja kodów źródłowych aplikacji komputerowych ogranicza konkurencję, powodując, że na rynek trafia oprogramowanie z licznymi błędami, mimo znacznej ceny.

Wyraźny rys **humanizacji informatyki** nadał i zaznaczył w 1985 roku Richard Stallman. Opublikował on wówczas tzw. Manifest GNU[3]. Skupił się w nim na konieczności odejścia od monopolizacji oprogramowania, proponując tym samym pewną równowagę dla agresywnej komercjalizacji wyników prac programistów. Manifest zawierał idee, wartości i pewną filozofię postaw – bliskich każdemu człowiekowi i wynikających z jego natury. Ta właśnie bliskość była powodem szerokiego upowszechnienia się ideologicznej filozofii Manifestu. W 1985 roku Stallman założył Fundację Wolnego Oprogramowania (*Free Software Foundation*). W ramach jej prac określono dwie filozoficzne wolności użytkowników oprogramowania w zakresie korzystania z niego. W Manifeście GNU czytamy: „słowo 'wolne' w naszej nazwie odnosi się do dwóch konkretnych wolności: po pierwsze, wolności kopiowania programu i dzielenia się

2 *Humanizacja ekonomii* [http://pl.wikipedia.org/wiki/Humanizacja_ekonomii]
3 Stallman R.: *The GNU Manifesto*, "Dr. Dobb's Journal of Software Tools", March 1985, vol. 10, no. 3

nim ze swoimi przyjaciółmi i współpracownikami; po drugie, wolności modyfikowania programu wedle własnego uznania, dzięki pełnemu dostępowi do kodu źródłowego”[4].

W późniejszym okresie wypracowano i zdefiniowano cztery przysługujące użytkownikom oprogramowania wolności, przy czym dla wolności oznaczonych numeracją 1 i 3 warunkiem koniecznym jest dostęp do kodu źródłowego:

- wolność do uruchamiania programu, w dowolnym celu (wolność 0);
- wolność do analizowania, jak działa program i zmieniania go, aby robił to, co potrzebne (wolność 1);
- wolność do rozpowszechniania kopii, aby móc pomóc innym ludziom (wolność 2);
- wolność do udoskonalania programu i publicznego rozpowszechniania własnych ulepszeń, dzięki czemu może z nich skorzystać cała społeczność (wolność 3)[5].

Mniej więcej w okresie publikacji Manifestu GNU upowszechniło się pojęcie **wolne oprogramowanie** (*free software*), a po raz pierwszy użyto go formalnie w Biuletynie GNU w 1989 roku[6]. Oprogramowanie, aby mogło być uznane za wolne, z definicji musiało wpisywać się początkowo w dwie, a w późniejszym okresie w cztery wymienione wolności łącznie.

Różnica między wolnością a otwartością

Filozoficzne podejście do kwestii wolności oprogramowania, ujęte z początku w ramach tylko dwóch wolności, według niektórych wydawało się nie dość wystarczająco jasno ujmować techniczne aspekty oprogramowania, dotyczące źródeł jego kodu. Dlatego w 1989 roku powstał *Ruch otwartego oprogramowania* i termin **otwarte oprogramowanie** (*open software*), czyli takie, które ma **otwarte kody źródłowe** (*open source*). Ruch ten kładzie większy nacisk na kwestie techniczno-organizacyjne związane z wolnością kodu, odsuwając tym samym na dalszy plan kwestie ideologiczno-filozoficzne. Za twórców Ruchu otwartego oprogramowania uważa się dwie kluczowe postacie, tj. Erica S. Raymonda oraz Bruce’a Perensa. Ich koncepcję rozwinęli Maddog Hall, Larry Augustin i inni, w tzw. *Definicji otwartego źródła*[7].

Monopolizacja narusza normy prawne, co wykazał sąd USA w sporze o wolność przeglądarek internetowych.

Nazwa O*twarte oprogramowanie* powstała m.in. również na potrzeby rozpowszechnienia kodu źródłowego oprogramowania przeglądarki internetowej Netscape Navigator firmy Netscape Communications Corporation. Firma ta w 1989 roku udostępniła wspomnianą przeglądarkę na licencji gwarantującej każdemu prawo do dowolnego użytku, modyfikacji i redystrybucji kodu. Była to jedna z pierwszych prób upowszechnienia na rynku informatycznym wolnego i otwartego oprogra-

4 *Dwie wolności* – definicja pierwotna [http://pl.wikipedia.org/wiki/Definicja_Wolnego_Oprogramowania]
5 *Cztery wolności*, GNU: Definicja Wolnego Oprogramowania [http://www.gnu.org/philosophy/free-sw.html]
6 *Free software*, “GNU’s Bulletin” 1989, vol. 1, no. 6 [http://www.gnu.org/bulletins/bull6.html#SEC5]
7 *The Open source definition*, Open Source Initiative [http://opensource.org/docs/osd]

mowania. Wprowadzenia takiego oprogramowania, ze względu na potencjalne straty finansowe, obawiały się duże firmy zajmujące się rozwojem i sprzedażą zamkniętych programów komputerowych.

Należy zdawać sobie sprawę z tego, że zachodzi **zasadnicza różnica** pomiędzy wolnościowymi a otwartościowymi ruchami czy wolnym a otwartym oprogramowaniem, choć dotyczy to niuansów i jest trudne do zrozumienia. Ruch wolnościowy czy wolne oprogramowanie zawierają w sobie elementy ideologii i wartości Ruchu otwartościowego i otwartego oprogramowania. Natomiast idee Ruchu otwartościowego i otwarte oprogramowanie posiadają węższe znaczenia i nie zawsze są w pełni tożsame z wartościami przyjętymi przez społeczność Ruchu wolnościowego czy z zasadami wolnego oprogramowania.

Otwartość polega na swobodnej dostępności do utworów czy informacji w przestrzeni cyfrowej, w tym w Internecie i nie musi być ona tożsama z możliwością ich kopiowania, modyfikacji, dalszej dystrybucji, szczególnie na warunkach komercyjnych. 'Otwartość' bowiem jest „niczym nieograniczona przestrzennie; skierowana do i przeznaczona dla wszystkich"[8]. A więc można powiedzieć, że termin ten dotyczy swobodnej technicznie dostępności do utworów. Przykładem mogą być **otwarte zasoby edukacyjne**, które są po prostu dostępne w sieci, a ich twórca może za nie pobierać opłaty, np. w przypadku ich dalszego wykorzystania do użytku komercyjnego. Błędne rozumienie pojęcia otwartości, jako tożsamego z wolnością, wzięło się z wprowadzenia *Definicji otwartego kodu źródłowego*.

W edukacji bezcenne są wszystkie wolne i otwarte dzieła: zasoby naukowe i kulturowe, technologie i standardy, sprzęt i instrukcje serwisowe.

Ze względu na ewolucję formalnej definicji wolnego oprogramowania z dwóch do czterech, która w ostateczności precyzowała otwartość kodu źródłowego, a także z uwagi na bliskość założeń filozofii Wolnego oprogramowania i technicznych aspektów Otwartego oprogramowania, dziś używa się zbiorczego określenia **Wolne i otwarte oprogramowanie** (WiOO), od angielskiego *Free Libre/Open Source Software*. Podkreśla się tym samym znaczenie otwartych kodów źródłowych, których ramy są ujęte w samej definicji Wolnego oprogramowania.

Filozoficzne idee i postawy ujęte w czterech wolnościach przysługujących użytkownikom oprogramowania zostały bezpośrednio bądź pośrednio zaadaptowane w innych dziedzinach życia. Zwolennikiem i propagatorem tego pomysłu jest Lawrence Lessig. Na kanwie takiej koncepcji w późniejszym okresie powstało wiele innych wolnościowych i otwartościowych ruchów oraz wytworów, m.in. *otwarte zasoby edukacyjne, otwarty sprzęt, otwarte standardy* czy *wolna kultura*.

Ruch wolnościowy i otwartościowy wymusił również regulację rozwiązań prawnych, adekwatnych do zjawiska dzielenia się swoją twórczością oraz wykorzystywania dzieł innych twórców w zakresie praw autorskich osobistych i majątkowych. Powstał katalog wolnych i otwartych licencji, nazywanych czasem *licencjami liberalnymi*.

8 *Otwarty, Słownik języka polskiego*, Wyd. Naukowe PWN [http://sjp.pwn.pl/slownik/2497100/otwarty]

Prawa autorskie i wolne/otwarte licencje

Prawo autorskie nie chroni idei, funkcji ani algorytmów, a jedynie formę utworu. Dlatego zamykanie kodów jest wybiegiem zamykającym dostęp do tego, co dla dobra nauki powinno być jawne.

Każda twórczość może być urzeczywistniona w postaci dzieła, mającego swoją intelektualną wartość. Urzeczywistniona twórczość może mieć różną formę, jako np.: utwór artystyczny, utrwalona myśl naukowa, wynalazek o charakterze technicznym czy nietechnicznym, a także program komputerowy. Oryginalne dzieła, nazywane w prawie *utworami*, mają autora lub autorów, a ci, co do zasady, mają do swojego utworu niezbywalne **prawo autorskie osobiste**. Utwory mają jeszcze drugą wartość – finansową, a co za tym idzie mogą być oszacowane w ramach pewnej wartość pieniężnej. Pierwotnie, potencjalna wartość finansowa, przysługuje jego autorowi w zakresie **praw autorskich majątkowych**, które można zbywać, a więc przekazywać osobom trzecim. Dyspozycja autorskich praw majątkowych nie zawsze musi wiązać się z korzyściami *stricte* finansowymi dla autorów – czasem z góry zakładają oni rezygnację z takiej gratyfikacji za swój utwór. Wystarczą im inne korzyści, np. uznanie w środowisku, bądź kierują się innymi motywami.

Jak wspomniano wyżej, w humanistycznej ekonomii nie tylko jednostkowy krótkotrwały zysk finansowy, ale i ogólne korzyści społeczne mają swe znaczenie dla rozwoju zrównoważonej gospodarki. Jednakże – nawet jeśli autor rezygnuje z gratyfikacji finansowej za swój utwór – co do zasady musi to jasno określić i wydać **stosowne dyspozycje**. Jak wskazano wyżej, tylko twórcom przysługuje pierwotne prawo do korzystania z utworu i rozporządzania nim na wszystkich polach eksploatacji – pola te ze względu na różne formy zapisu, prezentowania czy rozpowszechniania są inne dla treści tekstowych, a inne dla aplikacji komputerowych. Zasady prawa autorskiego osobistego i prawa autorskiego majątkowego (w tym ich niezbywalość i zbywalność) określa w Polsce *Ustawa o prawie autorskim i prawach pokrewnych*[9].

Oryginalne wytwory intelektualnej pracy programistów, poprzez ich wyszczególnienie w Prawie autorskim, mają wysoką rangę utworów.

Zbycie autorskich praw majątkowych do utworów można dokonać na zasadzie przyznawania do nich licencji. Zgodnie z Ustawą podmiot będący licencjodawcą może przekazać licencjobiorcy autorskie prawa majątkowe do utworu w formie tzw. *licencji*, czyli umowy o korzystaniu z utworu na wyraźnie określonych, znanych dotychczas polach eksploatacji. Bez określenia tych pól umowa licencyjna może być nieważna. Definicyjnie **licencja** (z łac. *licet*, „jest dozwolone"; imiesłów przymiotnikowy czynny: *licens*, „wolny")[10] to w ogólności rodzaj „upoważnienia na wykonanie czynności, których nie wolno wykonywać bez zezwolenia"[11]. Możemy wyróżnić m.in. licencje: pełne, wyłączne, niewyłączne, wolne i/lub otwarte oraz tzw. sublicencje. Licencja niewyłączna nie ogranicza grona licencjobiorców i adresowana jest do wszystkich potencjalnych użytkowników, dlatego też ma ona zastosowanie dla wolnych czy otwartych licencji.

9 *Ustawa z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych* (Dz.U. 1994 nr 24 poz. 83 z póź. zm.)
10 *Licencja (prawo)* [http://pl.wikipedia.org/wiki/Licencja_(prawo)]
11 *Licencja*, Encyklopedia PWN [http://encyklopedia.pwn.pl/haslo/3932302/licencja.html]

Na potrzeby edukacji szczególnie interesujące są **licencje wolne i/lub otwarte**. Można przyjąć, że *wolne licencje* to takie, które pozwalają na: korzystanie z dzieł w dowolnym celu, ich modyfikowanie, rozpowszechnianie kopii, dalsze publiczne udostępnianie własnych ulepszeń z dwoma zastrzeżeniami, a są to: 1) podanie autora pierwotnej wersji utworu oraz 2) rozpowszechnianie ulepszeń na takiej samej licencji, co utwór pierwotny. *Otwarte licencje* to takie, które pozwalają przynajmniej na ogólnodostępność utworów przy równoczesnej możliwości korzystania z nich w dowolnym celu.

Wymóg dalszego licencjonowania na identycznych warunkach jak utwór pierwotny nazywa się *efektem zarażania licencji* utworów zależnych. Efekt zarażania lub niezarażania licencją utworów zależnych przez utwory pierwotne, ujmowany jest również za Richardem Stalmannem jako licencja *copyleft* (zaraźliwa) i *non-copyleft* (niezaraźliwa). Wybór pomiędzy licencją *zaraźliwą a niezaraźliwą,* czy inaczej *copyleft* lub *non-copyleft*, opiera się w istocie na tym, czyje prawa twórca uznaje za ważniejsze – czy prawa kolejnego twórcy do zamknięcia dzieła zależnego, czy też prawa wszystkich potencjalnych użytkowników utworów do tego, by mieć zapewnione takie same wolności oraz by sami mogli stać się twórcami dzieł pochodnych.
Przykładami tego rodzaju **wolnych licencji** są:

- licencja na oprogramowanie – *niezaraźliwa / non-copyleft* BSDL (*Berkeley Software Distribution License*), *zaraźliwa / copyleft* GPL (*General Public License*);
- licencja na inne utwory – *niezaraźliwa / non-copyleft* Creative Commons BY (uznanie autorstwa), *zaraźliwa / copyleft* Creative Commons BY-SA (uznanie autorstwa i upowszechnianie na tych samych warunkach).

W odróżnieniu od licencji wolnych, w **licencjach otwartych** nie wszystkie cztery wspomniane wolności muszą być spełnione łącznie. Otwartość to dostępność w przestrzeni cyfrowej lub dostępność do kodu źródłowego, jednak licencja taka może ograniczać użytkownika czy odbiorcę danego utworu w zakresie dalszego użytku komercyjnego lub tworzenia utworów zależnych. W praktyce licencja otwarta to taka, która spełnia przynajmniej jedną wolność z czterech wspomnianych, przysługujących użytkownikowi utworu. Na przykład licencja CC BY-NC, pozwalająca na wykorzystanie, rozpowszechnianie i tworzenie dzieł zależnych wyłącznie w celach niekomercyjnych, nie jest licencją wolną, lecz otwartą – nie spełnia bowiem ona wszystkich czterech wolności łącznie. Nie są też wolne licencje z rodziny CC z warunkiem ND, zabraniające modyfikacji i tworzenia utworów zależnych.

Wygodną formą udzielania licencji jest podanie symbolu z odniesieniem do pełnego tekstu zastrzeżeń na dedykowanej stronie internetowej.

Wolne i otwarte narzędzia w Strategii SWOI

W Strategii wolnych i otwartych implementacji założono, że u adolescentów niezbędne jest formowanie nie tylko twardych umiejętności informatyczno-technicznych, ale również miękkich, w tym sztuki prowadzenia dialogu i negocjacji oraz partycy-

pacji w działaniach grupowych. Takie łączenie umiejętności z wartościowymi ideami i pożądanymi postawami jest konieczne dla wczesnego kształtowania kluczowych kompetencji, oczekiwanych w gospodarce opartej na wiedzy i innowacji. W realizacji Programu nauczania-uczenia się infotechniki, wartości i postawy prospołeczne oraz walory proedukacyjne obudowy dydaktycznej zajęć mają istotne znaczenie zarówno dla formowania cech wolicjonalnych uczniów, jak i dla wzbudzenia w nich zainteresowania nowoczesnymi technologiami. Z tego też powodu świadomie wybrano taką obudowę, która oparta jest na **Wolnym i otwartym oprogramowaniu** oraz na **Otwartym sprzęcie**. Są to optymalne narzędzia programistyczno-elektroniczne, służące do przeprowadzania edukacyjnych kół zainteresowań infotechnicznych, a jednocześnie umożliwiające uczniom samodzielne ćwiczenie ich użytkowania w domu.

Na potrzeby realizacji Strategii przygotowano i udostępniono wolne i otwarte narzędzia oraz materiały w Szkolnym Remiksie Ubuntu, który jest dystrybucją Linuksa.

Wśród **zalet edukacyjnych WiOO** można wymienić ich najistotniejszy aspekt, wynikający z jawności kodów źródłowych, dzięki czemu można uczyć się tworzenia implementacji poprzez analizę oraz modyfikację rozwiązań wzorcowych. Z założenia udostępnione na potrzeby Programu pakiety oprogramowania – Szkolny Remiks Ubuntu (dystrybucja oparta na jądrze Linux), aplikacje użytkowe i narzędziowe – są wolne i mają otwarte źródła. W zasobach Internetu dostępne są szczegółowe dokumentacje wyjaśniające funkcje użytych procedur, modułów, bibliotek, interfejsów etc. Ogólnodostępna dokumentacja stanowi bogaty materiał dydaktyczny, przydatny zwłaszcza ze względu na postać elektroniczną i możliwość szybkiego dotarcia do potrzebnego opisu. Poza tym społeczność skupiona wokół Ruchu wolnego i otwartego oprogramowania publikuje w Internecie olbrzymią liczbę różnych implementacji, wśród których każdy uczeń znajdzie coś ciekawego dla siebie, co zachęci go do własnych prób.

Idea **wolnych i otwartych implementacji** służy sprawie tworzenia dzieł na miarę osobistych możliwości, upowszechniania i doskonalenia ich we współpracy z innymi użytkownikami Serwisu e-Swoi. Taka metodyka *zwinnego programowania* i udostępniania kodów źródłowych, choćby krótkich implementacji, rozwijanych i doskonalonych następnie przez innych, jest najkorzystniejsza dla opanowania umiejętności programistycznych. Również udział w *projektach zespołowych*, z podziałem na zadania przy wykorzystaniu otwartego dostępu do edytowania kodu, jest znakomitym polem nabywania doświadczeń.

W miarę rozwoju Społeczności SWOI, skupionej wokoło Serwisu e-Swoi, poprzez wzbogacanie zasobów na zasadzie użytych w Serwisie mechanizmów wiki, funkcji tutorialnych i repozytoryjnych, ta dedykowana, otwarta dla wszystkich platforma wymiany dorobku, transferu wiedzy i umiejętności, może stać się narzędziem wspierania uczniów w długofalowym formowaniu kompetencji specjalistycznych i społecznych, dzięki współpracy rówieśniczej, a także dzięki wsparciu ze strony doradców.

Otwartość kodów źródłowych oprogramowania i pełen dostęp do specyfikacji otwartego sprzętu elektronicznego, mają dodatkowy **walor prospołeczny**. Tworzenie i oprogramowywanie otwartego sprzętu np. Arduino najczęściej powstaje na

zasadzie wolontariatu, przy współpracy wielu osób w sieci o różnych specjalnościach: programistów, elektroników, grafików, tłumaczy, testerów itp. Udział w zespołowej budowie programów komputerowych czy otwartych układów mechatronicznych współpracujących z komputerem oraz partycypacja w ich tworzeniu i doskonaleniu, kształci u uczniów umiejętność prowadzenia dialogu w przestrzeni cyfrowej, współpracy w grupie i pozwala doskonalić styl negocjacyjny.

Otwarty sprzęt, przeznaczony do zajęć, nie jest objęty patentami, jego specyfikacja techniczna jest jawna i ogólnodostępna. Zdolni uczniowie mogą sami odtworzyć struktury układów elektronicznych i ich interfejsów, budując i oprogramowując je według opisanych w wielu miejscach schematów. Mogą także, na zewnętrznej uniwersalnej płytce montażowej, konstruować własne układy o różnych funkcjach, z użyciem dodatkowych podzespołów (np. półprzewodników, czujników i przetworników dźwiękowych, optycznych, termicznych, indukcyjnych, pojemnościowych), których koszt jednostkowy to rząd najwyżej kilku złotych. Ponadto możliwe jest przyłączanie różnych posiadanych w domu urządzeń (np. słuchawek, mikrofonów), a nawet elementów z niewykorzystywanych już telefonów komórkowych (np. klawiatury czy wyświetlacza).

Otwarty sprzęt, np. układ elektroniczny Arduino, umożliwia dowolną jego konfigurację mechatroniczną i programistyczną.

Niebagatelny jest również **walor ekonomiczny** dla uczniów i szkół. Wolne i otwarte oprogramowanie, tworzone na liberalnych licencjach, nie przewiduje opłat licencyjnych. Tak jest z wieloma dystrybucjami opartymi na jądrze Linux np. Ubuntu, Red Hat, Suse, Mandriva, Debian i inne oraz z wieloma użytkowymi i narzędziowymi aplikacjami przygotowanymi dla wymienionych. Także zalecany do zajęć układ elektroniczny Arduino jest wielokrotnie tańszy od swoich odpowiedników o niejawnych i opatentowanych specyfikacjach technicznych.

Walory edukacyjne, społeczne i ekonomiczne przemawiają za wykorzystywaniem w szkołach wolnego i otwartego oprogramowania oraz otwartego sprzętu.

Ze względu na duże koszty zamkniętych systemów, programów i urządzeń, przy przejściu na Wolne i otwarte oprogramowanie oraz otwarte układy mechatroniczne, szkoła może racjonalniej zarządzać mniejszymi środkami finansowymi, skupiając się bardziej na doposażaniu w wydajniejszy sprzęt. Z perspektywy ucznia brak opłat licencyjnych i patentowych pozwala na niewykluczanie go ze względu na zasoby finansowe rodziców, a także daje możliwość realizacji ćwiczeń w domu, na takim samym oprogramowaniu i sprzęcie do mechatroniki jak w szkole.

Ważne z punktu widzenia wychowawczego jest kształtowanie odpowiedzialności uczniów za zgodne z prawem użytkowanie oprogramowania. Ich przekonanie w toku zajęć **o pełnej dostępności i funkcjonalności** Wolnego i otwartego oprogramowania, poparte bezpośrednim doświadczeniem pożytków z jego korzystania, stwarza szansę na zredukowanie dość powszechnego uprawiania piractwa komputerowego, bardzo często wręcz wymuszanego sytuacją finansową rodziny. Co więcej – głębsze poznanie wolnych programów o otwartych kodach źródłowych daje podwalinę

dla doskonalenia kwalifikacji oczekiwanych przez pracodawców. Elementy te tym samym dają wsparcie w wyrównywaniu szans i otwierają perspektywę zdobycia dobrze płatnego zawodu.

Bibliografia

- *Cztery wolności*, GNU: *Definicja Wolnego Oprogramowania* [http://www.gnu.org/philosophy/free-sw.html]
- *Dwie wolności* – definicja pierwotna [http://pl.wikipedia.org/wiki/Definicja_Wolnego_Oprogramowania]
- *Free software*, "GNU's Bulletin" 1989, vol. 1, no. 6 [http://www.gnu.org/bulletins/bull6.html#SEC5]
- Gates B.: *An Open Letter To Hobbyists*, "Homebrew Computer Club Newsletter" 1976, vol. 2, issue 1, p. 2,, Mountain View, CA [http://www.digibarn.com/collections/newsletters/homebrew/V2_01/gatesletter.html]
- *Humanizacja ekonomii* [http://pl.wikipedia.org/wiki/Humanizacja_ekonomii]
- *Otwarty,* Słownik języka polskiego, Wyd. Naukowe PWN [http://sjp.pwn.pl/slownik/2497100/otwarty]
- *Licencja*, Encyklopedia PWN [http://encyklopedia.pwn.pl/haslo/3932302/licencja.html]
- *Licencja (prawo)* [http://pl.wikipedia.org/wiki/Licencja_(prawo)]
- Stallman R.: *The GNU Manifesto*, "Dr. Dobb's Journal of Software Tools", March 1985, vol. 10, no. 3
- *The Open source definition*, Open Source Initiative [http://opensource.org/docs/osd]
- *Ustawa z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych* (Dz.U. 1994 nr 24 poz. 83 z póź. zm.)

dostęp do źródeł *on line* 2.1.2014 r.

02

Serwis edukacyjno-społecznościowy e-Swoi

Rafał Brzychcy

Jesteśmy świadkami końca epoki industrialnej i rozpoczęcia się epoki cyfrowej, w której obecni adolescenci wzrastają. Komputer i Internet są ich naturalnym środowiskiem codziennej aktywności. W sieciowych rozwiązaniach jest duży potencjał edukacyjny, który warto wykorzystać, bowiem uczniowie chętnie spędzają tam swój wolny czas. Wiele badań wskazuje, że młodzi ludzie, szczególnie ci, którzy wzrastali już w epoce cyfrowej, bardzo często korzystają z serwisów społecznościowych. Rozwiązania technologiczne umożliwiają logowanie się do nich w dowolnym miejscu i czasie, szczególnie poprzez powszechne już dziś urządzenia mobilne. Nauczyciele, bazując na tym zjawisku zanurzenia w świecie nowych mediów, powinni aktywnie i mądrze zagospodarowywać część czasu poświęcanego przez uczniów na pracę w Internecie, w celu realizacji procesu edukacyjnego.

Wykorzystywanie przez nauczających tak chętnie odwiedzanego i naturalnego dla młodzieży środowiska komunikacji społecznej może przynieść wiele korzyści i jest nieuniknione. Środowiska cyfrowe i sieciowe powinny być środkami wspierającymi proces edukacji, dlatego należy szukać różnych sposobów na ich efektywne spożytkowanie, czerpiąc z nich to, co najlepsze. Warto jednak pamiętać, aby przy okazji nie redukować spotkań stacjonarnych, a zwłaszcza zajęć pozalekcyjnych, gdyż taka forma szybciej i skuteczniej kształci postawy prospołeczne poprzez bezpośredni kontakt z nauczycielem i bezpośrednie interakcje między rówieśnikami. Najkorzystniejsze jest połączenie obu form – edukacji stacjonarnej i zdalnej. Dlatego współczesny nauczyciel powinien świadomie wplatać cele edukacyjne w społeczny dialog z wykorzystaniem nowoczesnych technologii informacyjno-komunikacyjnych.

Społeczności sieciowe w procesie edukacji

Zjawiskiem gwałtownie rozwijającym się w świecie cyfrowym jest tzw. **konwergencja mediów**. Następuje integracja w pojedynczym urządzeniu (np. w Smartfonie) wielu technologii i funkcji, jakie przedtem realizowały zupełnie odrębne urządzenia (telefon, mikrokomputer, fotoaparat, kamera, odtwarzacz, nawigator GPS, pilot itp.). Zachodzi też upodabnianie się roli urządzeń, standaryzacja protokołów transmisyjnych i formatów oraz ujednolicanie intuicyjnie obsługiwanych, graficzno-sensorycznych interfejsów użytkownika. Konwergencja objęła także portale, serwisy i edukacyjne platformy internetowe, które dostarczają coraz bogatszą ofertę zróżnicowanych, zaawansowanych funkcji i usług. Dzięki temu bardzo łatwo można dotrzeć do wielu funkcjonalności przydatnych w procesie nauczania-uczenia się.

Rozwój serwisów społecznościowych, na których obserwujemy interakcję użytkowników oraz współpracę w tworzeniu treści powoduje, że kształcenie zyskuje nową jakość dzięki szybkiej i szerokiej wymianie informacji, życzliwemu wsparciu przez kompetentnych doradców i łatwemu dostępowi do zasobów, mających wartość edukacyjną.

Serwisy internetowe pozwalają nie tylko na pobieranie materiałów, ale także na dostosowanie komunikacji czy transmisji do rytmu życia odbiorcy. Dają użytkownikom możliwość decydowania o tym, gdzie i kiedy korzystają z zawartości serwisu. Wiele badań wykazało, że uczniowie w społecznościowych serwisach poszukują nie tylko rozrywki, ale również źródeł wiedzy. Źródłem zasobów wiedzy może być bezpośrednio każdy inny użytkownik serwisu, w tym zwłaszcza **kreatywny nauczyciel**, który powinien tworzyć w sieci warunki zachęcające własnych uczniów do nauki i współpracy.

Wydaje się, że nauczycielskie środowisko nie docenia potężnych możliwości narzędzi cyfrowych i sieciowych w procesie edukacji, w tym przede wszystkim funkcjonalności serwisów społecznościowych. Często wręcz unika tego medium masowej komunikacji i dostrzega raczej złe wpływy na uczniów. Akcentowane są zwłaszcza negatywne skutki nadmiernego spędzania czasu przez młodzież w sieci. Jeśli jednak intensywne używanie serwisów społecznościowych przez uczniów jest faktem, to właśnie nauczyciele powinni ten fakt wykorzystać, czerpiąc pożytki z oferty nowych technologii informacyjno-komunikacyjnych.

Z obserwacji obecnych serwisów edukacyjnych można odnieść wrażanie, że – mimo potencjału technologicznego – nie mają one jednak pełnej funkcjonalności w zakresie dydaktyki i metodyki nauczania-uczenia się. Zazwyczaj nie odpowiadają oczekiwaniom i potrzebom w miarę szerokiego grona zainteresowanych. Są kierowane albo do uczniów, albo tylko do nauczycieli i innych podmiotów działających w sferze edukacji. Często budowane są z pominięciem aspektów psychologicznych. Dlatego też korzystne jest obserwowanie sprawdzających się i chętnie wykorzystywanych funkcji tych serwisów, w których dostrzec można uwzględnianie psychopedagogicznych i rozwojowych potrzeb adolescentów. W szczególności są to serwisy zwane Web 2.0.

W literaturze spotyka się wiele opinii, że dla uczniów funkcja informacyjna serwisów edukacyjnych jest mało istotna, gdyż koncentracja tylko na odbiorze informacji utrudnia ich zrozumienie. Jednak dzięki konwergencji nowych technologii uczniowie mogą być nie tylko odbiorcami, ale również **twórcami przekazów**. Służy temu znany mechanizm *wiki* – tj. funkcja edycyjno-informacyjna serwisów, dzięki której autorami lub współautorami są również uczniowie. Przyczynia się to do kształtowania umiejętności sprawnego posługiwania się językiem i wspólnego rozwiązywania zadań czy problemów.

Proces socjalizacji ucznia poprzez aktywne uczestnictwo w społecznościach sieciowych ma duży wpływ na jego rozwój, gdyż komunikaty otrzymywane od rówieśników są bardziej zrozumiałe. Zatem serwis edukacyjny powinien mieć charakter nie tylko dwukierunkowej komunikacji pomiędzy nauczycielem i uczniem, ale również wielostronnej interakcji pomiędzy młodzieżą i przedstawicielami różnych środowisk.

Edukacyjne witryny WWW, pretendujące do miana serwisów edukacyjnych, można podzielić na trzy grupy: **informacyjne i podawcze, interaktywne** oraz **społecznościowe**. Biorąc pod uwagę, że edukacja to proces znacznie szerszy niż przekazywanie informacji, warto wykorzystać

elementy interaktywności, dzięki której można budować wokół serwisu zaangażowaną społeczność. Informacja połączona z elementami interakcji w postaci komentarzy czy ocen treści stanowi merytoryczne i metodyczne wsparcie dla użytkowników.

Każdy serwis edukacyjny, poprzez aktywność jego społeczności, powinien wspomagać proces logicznego, konstruktywnego oraz kreatywnego myślenia, czytania ze zrozumieniem, redagowania tekstów, samodzielnego uczenia się, twórczego rozwiązywania problemów, wyszukiwania, selekcjonowania, porządkowania i przetwarzania informacji, współpracy w grupie i rozumienia innych. Są to kompetencje uznawane współcześnie jako kluczowe dla rozwoju społecznego.

Edukacyjne serwisy mają tę zaletę, że gromadzi się w nich przydatne informacje i dorobek z różnych dziedzin wiedzy, a także umożliwia ocenianie ich poprawności i wartości przez szerokie grono odbiorców. Oprócz mechanizmów współtworzenia i modyfikowania treści wolnych zasobów, serwisy jeszcze jedną ważną funkcję – można analizować **wersje rozwojowe** modyfikowanego dokumentu, poprzez śledzenie wprowadzanych zmian. Wgląd do poprzednich wersji pozwala uczniowi zauważyć, w jaki sposób subiektywne przeświadczenia pojedynczych osób są przekształcane w procesie dialogu społecznego i poszukiwania prawdy obiektywnej. W interesującej nas dziedzinie infotechniki, sieciowa współpraca nad śledzeniem rozwojowych wersji implementacji lub opracowywaniem specyfikacji technicznych, kształtuje u ucznia umiejętność optymalizacji udostępnianych wytworów.

Głównym symptomem wartości dobrego serwisu edukacyjnego, oprócz jego funkcji i charakteru społecznościowego, powinna być bogata **baza materiałów** edukacyjnych, przeznaczonych dla wszystkich zainteresowanych użytkowników. Ważne jest, aby serwis nie był adresowany wyłącznie do wąskiej, zamkniętej grupy (np. tylko dla specjalistów), ale aby każdy odwiedzający daną platformę znalazł na początku coś ciekawego dla siebie. Stwarza to szansę na zaangażowanie się w partycypację, nawiązanie międzyrówieśniczej i międzypokoleniowej współpracy oraz poczucie przynależności do grona pasjonatów danej dziedziny wiedzy.

Serwisy Web 2.0 pomagają w kształtowaniu postaw obywatelskich i prorozwojowych. Młodzież bowiem szuka interakcji społecznych nie tylko w celu nawiązania znajomości, przyjaźni, utrzymywania wirtualnych relacji czy zmniejszania barier występujących w komunikacji bezpośredniej. W tym wieku młodzi adolescenci szukają też przyjaznej dla nich **platformy współdziałań** o charakterze edukacyjnym, polegających m.in. na: łączeniu się w grupy zainteresowań, informowaniu i wymianie opinii o wydarzeniach, dyskutowaniu na tematy związane z nauką, planowaniu i realizowaniu projektów, a także pozyskiwanie rekomendacji dotyczących korzystania z przydatnych aplikacji i źródeł, które mogą wspomagać uczenie się.

Zaletą serwisów społecznościowych – pod warunkiem, że będą one wykorzystywane przez uczniów i nauczycieli – może być przeniesienie części komunikacji pomiędzy nimi w jedno dedykowane miejsce. Użytkowanie otwartych serwisów jest tanie, proste i przyjemnie. Uczniowie mogą sami organizować się w grupy zainteresowań, a interwencja administratora powinna ograniczać się do minimum. Moderatorami takich platform powinni być właśnie nauczyciele, którzy poprzez sieć łatwiej wchodzą w równoprawne interakcje, gdy trzeba służą radą, mądrze organizują i dyskretnie motywują uczniów do aktywnego uczestnictwa. Muszą jednak być także współtwórcami zasobów edukacyjnych danego serwisu.

Funkcjonalności Serwisu e-Swoi

Oddajemy uczniom i nauczycielom dedykowane narzędzie, które wspomaga osiąganie celów Strategii SWOI w formowaniu kompetencji infotechnicznych i społecznych. Serwis edukacyjno-społecznościowy e-Swoi, jako wirtualna przestrzeń cyfrowa, służy do realizacji nauczania-uczenia się w formie zdalnej, dopełniającej formę zajęć stacjonarnych w ramach kół zainteresowań. Istotą Strategii jest kształcenie poprzez uruchamianie **twórczości implementacyjnej**, a taki rodzaj uczenia się konstruktywistycznego wymaga przeznaczenia więcej czasu niż dostępny na zajęciach szkolnych. Projektowanie i programowanie wymaga pełnego zanurzenia się w czynnościach umysłowych, jest to więc proces złożony i długotrwały. W pokonywaniu trudności konieczne jest wspieranie zewnętrzne i właśnie tę rolę znakomicie może pełnić społeczność sieciowa.

Specyfika edukacyjnych serwisów społecznościowych powoduje, że choćby nawet najbardziej rozbudowane funkcjonalnie są jedynie narzędziami, a cała ich istota efektywności sprowadza się do aktywności wszystkich grup użytkowników i osób wspomagających. Przede wszystkim potrzebna jest chęć działania uczniów, ale też konieczna jest partycypacja nauczycieli, metodyków i moderatorów, a w przypadku Serwisu z dziedziny infotechniki także doradców, specjalistów od informatyki i mechatroniki, zwłaszcza od programowania i konstruowania układów elektronicznych.

Funkcjonalność społeczna Serwisu edukacyjnego e-Swoi jest zatem najważniejsza i niezbędna, aby uruchamiać działania mające na celu: ukierunkowanie młodzieży na pożyteczną ścieżkę rozwoju, formowanie ich wiedzy i umiejętności, kształtowanie twórczych i społecznych postaw oraz rozbudzanie świadomości i cech wolicjonalnych, niezbędnych w wytrwałym dążeniu do celu.

Drugim aspektem wpływającym na przydatność i efektywność narzędzia jest **funkcjonalność użytkowa** Serwisu. Na tę właściwość platformy składają się wszystkie dostępne funkcje, usługi i zasoby, ale także w ujęciu ich cech jakościowych, tj. przydatności, intuicyjności obsługi, szybkości działania, dobrego ustrukturyzowania i bogactwa materiału itp. Na Wśród całego kompleksu elementów użytkowych Serwisu e-Swoi można dostrzec takie funkcje, jak: informacyjna i komunikacyjna, organizacyjna i wspomagająca uczenie się, gromadząca zasoby źródłowe i wytwory, kategoryzująca treści, porządkująca i udostępniająca materiały dydaktyczne, aktywizująca poprzez konkursy, dokumentująca aktywności i dorobek, z możliwością ich oceniania. Część z tych funkcji realizowana jest automatycznie, a część uruchamiana poprzez wybór dostępnych modułów, o nazwach: *e-portfolio, repo, e-tutor, zasoby, program* oraz *konkurs*. Innym dostępem do oferowanych usług jest wybór Kategorii Serwisu: *Newsy, Programy, Tutoring, Roboty, FAQ, Dydaktyka, Idee, Taktyka* itd. Moduły funkcjonalne oraz kategorie zasobów są stale doskonalone i rozbudowywane, dlatego dalszy opis dotyczy postaci z początku roku 2014.

Podstawową funkcją Serwisu e-Swoi jest mechanizm **informacyjno-edycyjny**, oparty na koncepcji *wiki*. Umożliwia on każdemu zainteresowanemu dzielenie się ogłoszeniami, wiadomościami, przemyśleniami czy opisami swoich prac. Artykuły są przez użytkowników Serwisu oceniane i wartościowane, dzięki czemu następuje podział na dwie kategorie: *oczekujące na ocenę* użytkowników i *ocenione jako wartościowe*, które powinny pojawić się na stronie głównej. Artykuły oczekujące na ocenę umieszczane są w tzw. „wylęgarni newsów". Treść można oceniać poprzez oddanie głosu, przy czym artykuły ocenione pozytywnie przez min. 10 osób widoczne

są na stronie głównej, a tekst, który uzyska powyżej 15 głosów, pojawia się na czołowym miejscu strony powitalnej, jako artykuł dnia. Każdy użytkownik może komentować artykuły, a ponadto istnieje możliwość dyskretnego moderowania poprzez mechanizm „wykopywania" najwartościowszych artykułów w taki sposób, aby mogły pojawić się na stronie głównej.

Mechanizm *wiki* umożliwia współtworzenie artykułów. Autorzy udostępniają swoje opracowania w formie otwartej dla innych użytkowników serwisu. Treści te mogą być edytowane, moderowane, modyfikowane przez każdego, kto posiada chęć i wiedzę, aby to uczynnić. Dodatkowo można przeglądać listę i autorów zmian treści, bowiem wszystkie zmiany są rejestrowane i wersjonowane. Pierwotny autor artykułu może przywrócić dowolną jego wersję.

Kolejne funkcjonalne moduły Serwisu są w różnym stopniu przeznaczone głównie dla uczniów bądź nauczycieli. Przykładowo – uczniom najbardziej przydaje się dostęp do usług wspierających (np. doradztwo, tutoring, FAQ, instrukcje), a dla nauczycieli ważne jest pozyskiwanie materiałów dydaktycznych i metodycznych (opisów idei, metodyk szczegółowych, konspektów, zadań). Mimo dwutomowej postaci drukowanej, upowszechniającej Strategię SWOI oraz Program nauczania-uczenia się, treści i narzędzia w dziedzinie infotechniki zmieniają się tak szybko, że aktualizacja jest możliwa jedynie w postaci cyfrowej. I temu właśnie służy **funkcjonalność programowa**, dedykowana dla nauczycieli-trenerów.

Nauczyciele w pierwszy rzędzie otrzymują do dyspozycji całość materiałów do pobrania w formie elektronicznej, dzięki czemu mogą z obszernych opracowań wydobywać fragmenty potrzebne do konkretnych zajęć, a przede wszystkim, dzięki otwartym zasobom mogą dostosowywać treści zadań i zakresy udostępnianych kodów do potrzeb swoich uczniów. Zasoby dydaktyczne znajdują się w bloku funkcjonalnym o nazwie *program*, zawierającym opisy idei i taktyki oraz zbiory konspektów i implementacji. Mechanizm funkcjonalności programowej jest tak skonstruowany, że nauczyciel w prosty sposób może złożyć z wybranych elementów swój własny Plan nauczania, przeznaczony na adekwatną do potrzeb liczbę godzin zajęć dydaktycznych o optymalnej dla swych uczniów treści.

Z kolei głównie dla uczniów dedykowana jest **funkcjonalność e-portfolio**. Mechanizm ten automatycznie rejestruje aktywności użytkownika serwisu a także umożliwia przedstawienie się, opisywanie własnych przemyśleń i refleksji z zajęć. Posiadacz takiego e-portfolio decyduje o tym, jaka część ma mieć charakter prywatny, a jaka publiczny. Do opisów publicznych każdy zalogowany i zainteresowany użytkownik ma bezpośredni wgląd. Wykorzystanie e-portfolio jest niezwykle przydatne dla uczniów w gromadzeniu i dokumentowaniu swoich osiągnięć, a zwłaszcza dyscyplinowaniu i doskonaleniu poprzez samoocenę dokonań. Udokumentowane fakty i artefakty z e-portfolio mogą służyć do argumentacji podczas starań o przyjęcie do upragnionej szkoły, a w przyszłości podczas ubiegania się o pracę.

Nauczyciele muszą rozumieć ważną rolę elektronicznego portfolio i bezwzględnie pilnować jego systematycznego uzupełniania. Powinni też kształtować u uczniów wcześniej nieznaną umiejętność prawidłowego tworzenia e-portfolio. Chodzi nie tylko o przypominanie o konieczności składowania wytworów, ale także o wprawianie do pisania zwięzłych komentarzy, opinii,

wniosków i refleksji na temat zajęć i nabywanych umiejętności. W partnerskim stylu współpracy uczeń powinien udostępniać zasoby e-portfolio trenerowi, gdyż jest to ważny materiał do oceniania efektów zajęć, do ewaluacji formatywnej oraz do autorefleksji i samodoskonalenia nauczycieli.

Funkcjonalność tutorialna może być i jest wykorzystywana zarówno przez nauczycieli, jak i przez uczniów. Pierwszy mechanizm e-tutoringu umożliwia uzyskanie porady poprzez bezpośredni kontakt z Doradcami, którymi są specjaliści od mechatroniki, programowania, systemu i aplikacji linuksowych oraz od dydaktyki i ewaluacji. Odpowiedzi mogą być adresowane zwrotnie wyłącznie do osoby pytającej, lecz jeśli poruszona kwestia może być ciekawa także dla innych, to jest wówczas wykorzystywany mechanizm FAQ, czyli ogólnodostępna publikacja wyjaśnień na najczęściej pojawiające się pytania.

Na potrzeby zdalnego porozumiewania się użytkowników Serwisu opracowano narzędzie do bezpośredniej komunikacji dwustronnej. Służy ono do tworzenia i wysyłania wiadomości prywatnych. Mechanizm **poczty** o nazwie *Twoje wiadomości* umożliwia pisanie komunikatów i odpowiadanie na listy oraz zarządzanie zasobami wiadomości odebranych i wysłanych. Funkcjonalność ta z powodzeniem może być wykorzystana zarówno przez uczniów, jak i nauczycieli oraz doradców.

Kolejny mechanizm e-tutoringu to skategoryzowana **lista artykułów i instrukcji**, w których zawarte są materiały multimedialne, filmy instruktażowe, porady i wskazówki przydatne dla użytkowników indywidualnych w samokształceniu, lecz także dla trenerów do optymalizacji zajęć. Znajdziemy tu porady dotyczące Serwisu, Szkolnego Remiksu Ubuntu, instrukcje związane z wykorzystywaniem układu elektronicznego Ardiuno itp. Całość zróżnicowanych tematycznie zasobów e-tutora stanowi swoisty przewodnik, wspierający skuteczne nauczanie i uczenie się. Jego odbiorcami mogą być także goście i obserwatorzy – wszyscy oni mogą brać aktywny udział w rozwoju tych wolnych zasobów. Jest to zatem ogólnodostępna platforma transferu wiedzy i umiejętności z dziedzin infotechnicznych oraz pomost międzypokoleniowej współpracy i międzyrówieśniczej koedukacji *on line*.

Funkcjonalność repozytoryjna na Serwisie e-Swoi ma zróżnicowane formy realizacji. Najbardziej pożądanym sposobem udostępniania autorskich implementacji jest przedstawienie ich w postaci opisów tworzonych jako artykuły zawierające instrukcje i kody źródłowe, a jeszcze lepiej – wstawione linki do plików z elektroniczną formą implementacji. Każdy artykuł jest **kategoryzowany**, dlatego w trakcie publikacji potrzebny jest wybór odpowiedniej grupy tematycznej. Istnieją grupy dedykowane, zatem użytkownik wybiera kategorię najbardziej adekwatną do publikowanej treści. Trafne dopasowanie kategorii do treści pomaga w katalogowaniu, wyszukiwaniu i porządkowaniu zasobów w Serwisie.

Inna dostępną formą realizacji e-repozytorium jest struktura hierarchiczna folderów na udostępnionej przestrzeni dysków serwera. Jest to rodzaj e-katalogu z webowym interfejsem do zarządzania dodanymi przez użytkownika plikami. Właściciel danego zasobu ma możliwość gromadzenia plików w katalogu osobistym, względnie udostępniania w katalogu współdzielonym bądź publicznym. W zależności od rodzaju katalogu i zasobów, ma też możliwość ustawiania atrybutów otwartości plików tylko do odczytu lub także do otwierania i edycji.

Najbardziej przydatnymi dla społeczności są wszelkie autorskie, oryginalne implementacje, udostępniane na wolnych i otwartych licencjach. Mogą to być także półprodukty przydatne dla innych użytkowników (np. elementy graficzne, próbki dźwiękowe lub fragmenty

kodów). Repozytorium jest więc nie tylko miejscem prezentacji swej twórczości, lecz także platformą dzielenia się wytworami z innymi użytkownikami Serwisu.

Serwis, który stanowi nie tylko wsparcie i narzędzie do bieżącej pracy uczniów na kołach zainteresowań, służy również jako środek upowszechniania innowacyjnego modelu edukacyjnego. Umieszczono tam opis wypracowanej Strategii i przetestowanego Programu, a więc zarówno część koncepcyjną, jak również praktyczną, związaną z Konspektami-scenariuszami zajęć i zadaniami dla uczniów w postaci wzorcowych Implementacji. Przyporządkowane są one do odpowiednich poziomów uczniów i modułów zajęć. Te zweryfikowane i zwalidowane materiały dydaktyczne są szczególnie zalecane do realizacji.

Jednakże otwartość e-repozytorium umożliwia publikację i udostępnianie dalszych pomysłów na Moduły zajęć, tworzonych przez nauczycieli bądź informatyków czy mechatroników, a nawet przez rodziców czy samych uczniów, z metodycznym wsparciem doradców. Liczymy na to, że Serwis stanie się pomostem międzypokoleniowym entuzjastów infotechniki. Każde nowe **implementacje** wzbogacają dynamicznie rozwijający się materiał do ćwiczeń. W prosty i intuicyjny sposób można tam umieszczać własne propozycje Konspektów, zadań i rozwiązań implementacyjnych. Udostępnione pliki można ściągnąć lub wydrukować. Można tam również skonstruować własny Program zajęć z uczniami, umieszczając w nim te Moduły, które są najkorzystniejsze do realizacji z uczniami. Jak każdy otwarty zasób w Serwisie, także tu istnieje możliwość edytowania i aktualizowania treści scenariuszy, zadań i implementacji.

Na potrzeby porządkowania i kategoryzacji zawartości Serwisu, a tym samym sprawnego dostępu do informacji na interesujący użytkownika temat, stworzono **wyszukiwarkę zasobów**. Rejestrowane są tam wszystkie treści, jakie pojawiły się w serwisie, bez względu na ich ocenę przez użytkowników. Serwis społecznościowy w dużej mierze moderowany jest przez samą jego społeczność, zatem najważniejszą rolę moderacyjną powinni pełnić nauczyciele. Zaangażowana społeczność skuteczniej moderuje treści tak, aby były one obiektywne i wartościowe, dlatego warto, aby nauczyciele dbali o poprawność treści umieszczanych przez swoich wychowanków.

Aby kształcenie przy wykorzystaniu Serwisu było efektywne, konieczne są bliższe interakcje pomiędzy trenerem a wychowankiem. Nauczyciele powinni być nastawieni na aktywny dialog z uczniami. Należy pamiętać, że dla uczestników wspieranych w edukacji zdalnej duże znaczenie w przyswajaniu wiedzy mają **czynniki motywacyjne** – takie jak nagradzanie, wzmacnianie, czy zachęcanie do angażowania się w określone działania poprzez mądrą moderację, komentowanie, udział w pracach uczniów. Bez aktywności nauczyciela w Serwisie, osiągnięcie celów edukacyjnych Strategii wydaje się być niemożliwe. Nieodłącznym elementem pobudzania motywacji są **konkursy**. W Serwisie przewidziano, że każdy jego użytkownik może ogłosić konkurs dla uczniów, nauczycieli czy dowolnych zainteresowanych – służy do tego dział *Konkurs*. Właśnie prace zespołowe pod kierunkiem trenera są znakomitą formą zbliżania mistrzów i ich uczniów, a co ciekawe – w dziedzinie infotechniki często te role odwracają się…

03

Szkolny Remiks Ubuntu – środowisko pracy

Adam Jurkiewicz

Szkolny Remiks Ubuntu bazuje na dystrybucji Linux Ubuntu 12.04 LTS wraz ze środowiskiem graficznym XFCE. Wybraliśmy taką konfigurację, ponieważ bardzo dużo komputerów w szkołach to sprzęt stary i rozbudowane środowiska graficzne nie są w stanie płynnie pracować na słabszych konfiguracjach sprzętowych. Został on dostosowany specjalnie na potrzeby Projektu SWOI. Więcej informacji dotyczących Ubuntu oraz ogólnie Linuksa można znaleźć na stronach serwisów:

- http://www.ubuntu.pl
- http://jakilinux.org/linux/ubuntu/
- http://pl.wikipedia.org/wiki/Linux

Szkolny Remiks Ubuntu uruchamiany jest bezpośrednio z Pamięci *Flash* (*pendrive* USB, klucz USB), zatem nie ma konieczności instalowania go na dysku twardym komputera, na którym chcemy pracować. Wyposażony jest w narzędzia, które automatycznie konfigurują urządzenia podczas uruchamiania. Wykrywane są zarówno karty bezprzewodowe, telewizyjne, jak również urządzenia peryferyjne: skanery, drukarki, karty sieciowe (również WiFi), karty dźwiękowe czy też zewnętrzne napędy.

Należy pamiętać, że dane użytkownika, który pracuje ze *Szkolnym Remiksem Ubuntu*, są zapisywane na p*endrive* w jego specjalnej części, tak więc zapisana praca nie przepada po zakończeniu pracy ze *Szkolnym Remiksem Ubuntu*. Konieczne jest jednak, aby przestrzegać procedury poprawnego wyłączenia systemu (opisane to będzie w dalszej części).

Ogólny opis oprogramowania

Oprogramowanie zawarte w *Szkolnym Remiksie Ubuntu* to pakiety edukacyjne, graficzne, biurowe i multimedialne oraz pakiety i środowiska programowania niezbędne do wykonywania zadań określonych przez Strategię SWOI. Dystrybucja nie wymaga praktycznie żadnych dodatkowych konfiguracji (oczywiście poza konfiguracją drukarki czy połączenia z siecią, jeśli nie posiadamy DHCP). Startuje z gotowym, spolszczonym środowiskiem graficznym (XFCE 4.8). Domyślnym językiem dystrybucji jest język polski.

Na pulpicie zamieszczono ikony do najważniejszych elementów wykorzystywanych podczas zajęć, m.in.: Arduino, Gimp, Inkscape, Lazarus, Scratch i inne programy, które uczniowie będą wykorzystywać na zajęciach.

Więcej informacji o środowisku XFCE można znaleźć na:
- http://pl.wikipedia.org/wiki/Xfce
- http://www.xfce.org/about/tour4.8

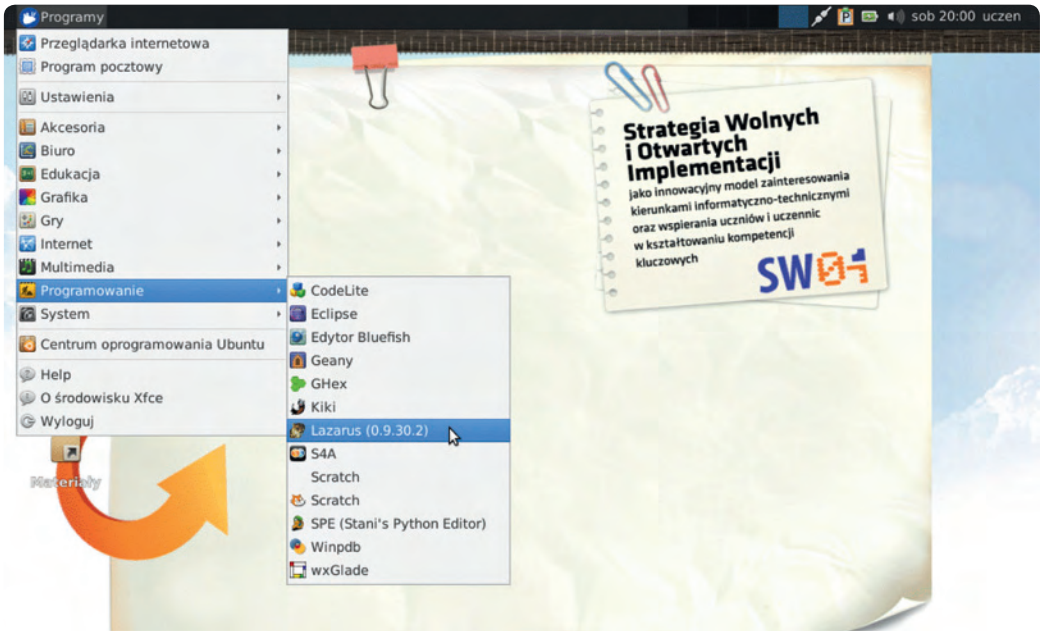
Techniczny sposób uruchamiania

Żeby poznać coś naprawdę – najlepiej tego spróbować. Żaden opis nie zastąpi praktyki. Jeśli chcemy uruchomić system – należy włożyć p*endrive* do portu USB. Konieczne jest wówczas ponowne uruchomienie komputera. Ponieważ system ma pracować z pamięci *Flash* zupełnie tak, jakby pracował z twardego dysku, należy „poinformować" o tym maszynę, na której go uruchamiamy. Zwykle kolejnością uruchamiania (bootowania) poszczególnych napędów steruje oprogramowanie w pamięci płyty głównej komputera (*BIOS*). Jeśli po restarcie komputera Linux uruchamia się, oznacza to, że mieliśmy ustawioną opcję bootowania w pierwszej kolejności z USB-HDD. Jeśli jednak nie startuje, podczas uruchamiania należy wcisnąć klawisz *Del, Esc, F1, F2* lub inny, który powinien być opisany przy starcie komputera. Zazwyczaj któryś z nich spowoduje wyświetlenie menu *BIOS*, z którego będzie można wybrać kolejność startowania napędów komputera.

Po pomyślnym wystartowaniu systemu z p*endrive* – powinien nam się ukazać ekran startowy (*tzw. splah screen*), a po pewnym czasie system po dekompresji automatycznie wykryje i skonfiguruje wszystkie urządzenia naszego komputera, takie jak: karta sieciowa, dźwiękowa, itp. Następnie uruchomi się graficzne środowisko pracy – XFCE. W tym momencie możemy już spokojnie rozpocząć pracę ze *Szkolnym Remiksem Ubuntu*.

Edukacja i aplikacje użytkowe

Szkolny Remiks Ubuntu zawiera wiele programów edukacyjnych, użytkowych i innych. Uruchamianie programów zawartych w *Szkolnym Remiksie Ubuntu*, dla których nie ma ikon na pulpicie, jest możliwe poprzez Panel Aplikacji na górnej belce z opisem **„Programy”**:



Zakończenie pracy z systemem i wyłączanie komputera

Aby zapobiec awariom systemu, należy duży nacisk położyć na bezpieczne jego wyłączanie. Ponieważ system w czasie swojej pracy zapisuje na dysku (*klucz pendrive*) pewne dane, dlatego pod żadnym pozorem **nie wolno** w trakcie działania systemu wyciągać urządzenia z portu USB komputera.

Uwaga!
Wyciągnięcie urządzenia (pendrive USB) może skutkować zniszczeniem systemu, utratą danych i niemożnością dalszej pracy. Nikt nie zagwarantuje odzyskania uszkodzonych w ten sposób danych.

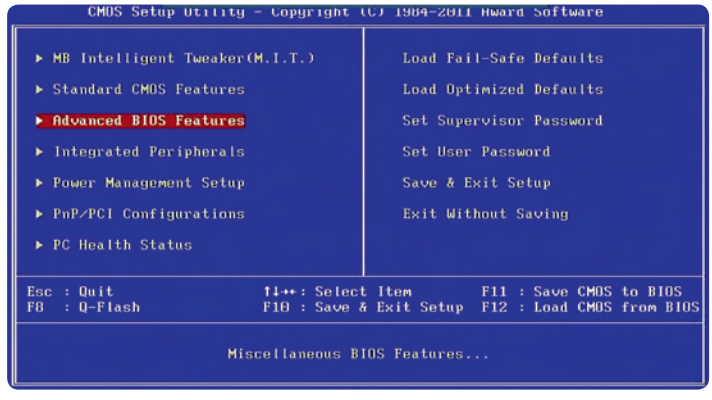
Wyłączenie systemu powinno nastąpić wtedy, kiedy zapiszemy wszystkie nasze dane na dysku (klucz pendrive). Służy do tego opcja: **Wyłącz – w prawym górnym rogu ekranu. Następnie zobaczymy okno, w którym należy wybrać opcję WYŁĄCZ.**

Instrukcja uruchamiania Szkolnego Remiksu Ubuntu z klucza USB

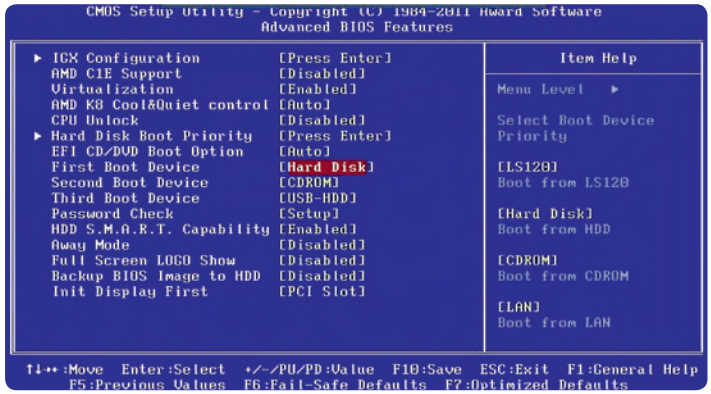
1 Przed uruchomieniem komputera umieść klucz USB z systemem operacyjnym w porcie USB komputera.

2 Włącz komputer.

3 Po uruchomieniu komputera włącz ustawienia BIOS – pierwsza informacja, która pojawia się na ekranie po włączeniu komputera, dotyczy między innymi ustawienia BIOS (z reguły jest to jeden z przycisków funkcyjnych, np. F2, F10). Przycisk ten należy nacisnąć przed zniknięciem tej informacji z ekranu w trakcie uruchamiania komputera.

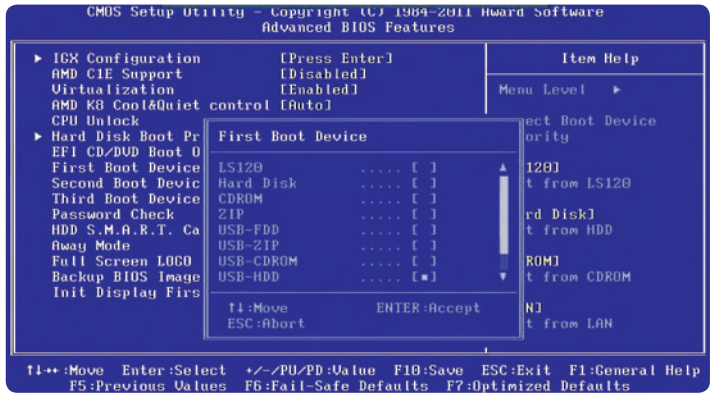


4 W BIOS-ie szukamy opcji uruchamiania (boot options, boot order, boot devices). Często znajdują się w sekcji Advanced BIOS Features lub posiadają odrębną część w BIOS. Niestety nie jesteśmy w stanie powiedzieć, gdzie Ty znajdziesz takie opcje.

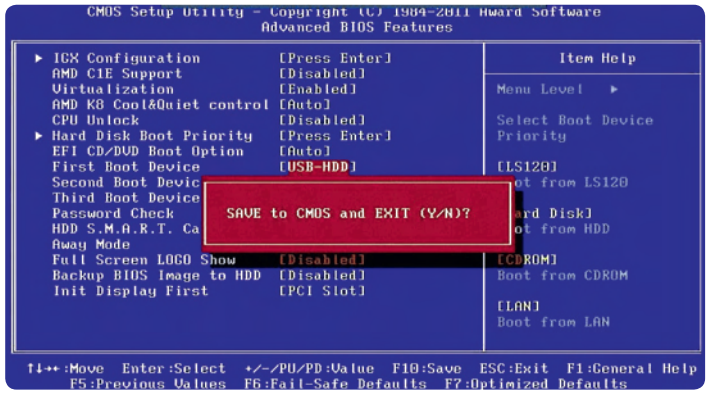


5 Wejdź do ustawień kolejności urządzeń, z których komputer próbuje załadować system operacyjny (boot order, boot devices itp.).

6 Ustaw klucz USB jako pierwsze urządzenie.



7 Zapisz ustawienia BIOS i uruchom komputer ponownie. Uwaga! W większości BIOS-ów klawisz F10 zapisuje ustawienia, jednak nie możemy dać pełnej gwarancji – zawsze czytaj na ekranie, powinna być tam informacja.



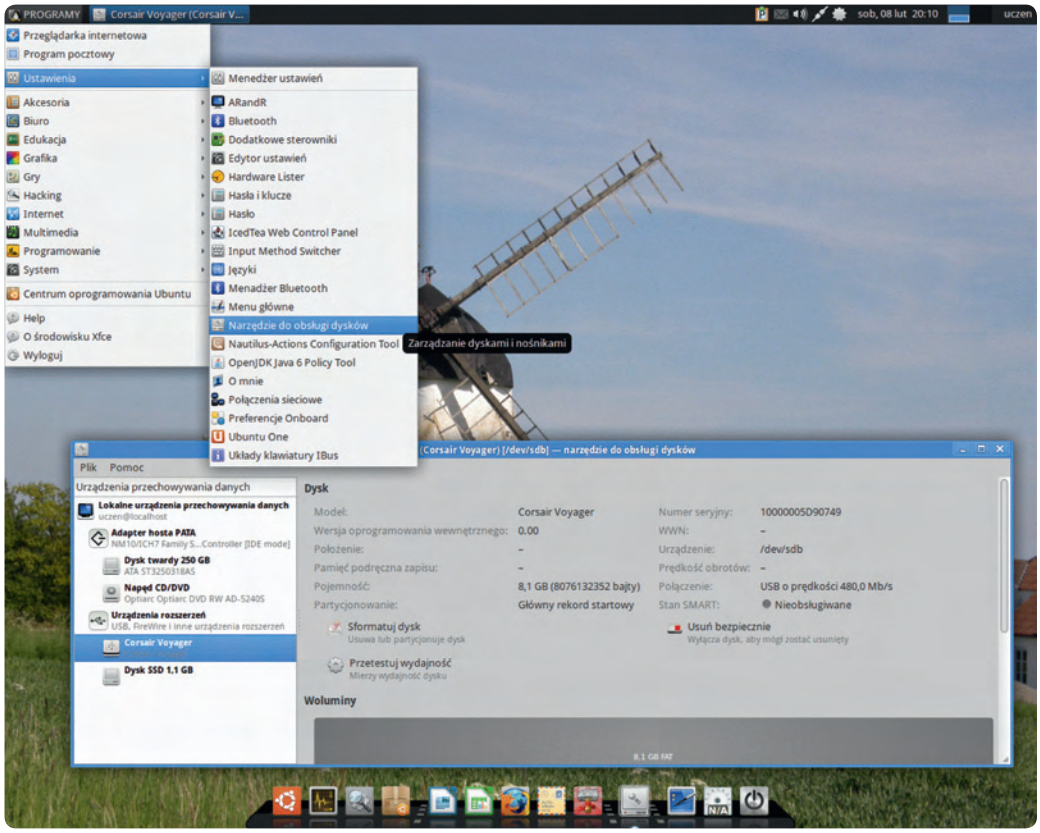
8 Po ponownym uruchomieniu czekaj – komputer powinien załadować system operacyjny znajdujący się na kluczu USB.

Instrukcja instalacji obrazu SRU_SWOI 12.04 na kluczu USB

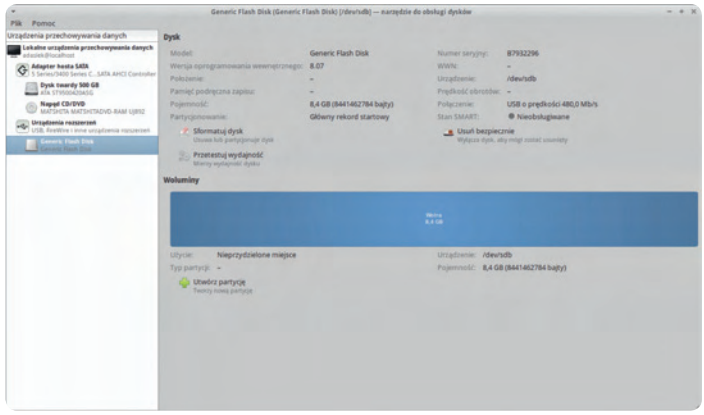
Ponieważ najnowsza wersja zakłada istnienie dwóch partycji na kluczu USB, z czego druga (na dane użytkownika) musi mieć system plików ext4 (linuksowy), więc przygotowywać klucze USB można jedynie pod kontrolą systemu Ubutnu 12.04 LTS (Ubuntu, Kubuntu, xUbuntu, Lubuntu). Poniższa instrukcja i zrzuty ekranowe wykonywane są na xUbuntu 12.04 LTS (a więc Ubuntu z XFCE).
Do wykonania instalacji konieczne są dwa programy:

- Narzędzie do obsługi dysków;
- Asystent nośnika rozruchowego (usb-creator -gtk lub -kde);

1 Pobierz plik .iso z obrazem systemu, zapisz gdzieś w systemie plików;



2 Umieść klucz USB w porcie USB, uruchom narzędzie do obsługi dysków;



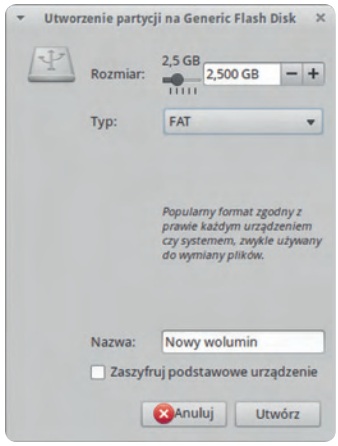
3 Skasuj wszystkie istniejące partycje oraz załóż dwie nowe:

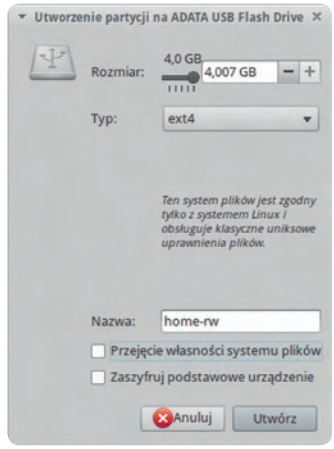
- Partycja podstawowa (np. sdb1), typy VFAT, wielkości około 3,5 GB (aby ISO się zmieściło);
- Partycja podstawowa (np. sdb2), typu ext4, wielkości reszty pendrive, ważne: „home-rw”.

Ważne!

NIE dla „przejęcie własności plików”.

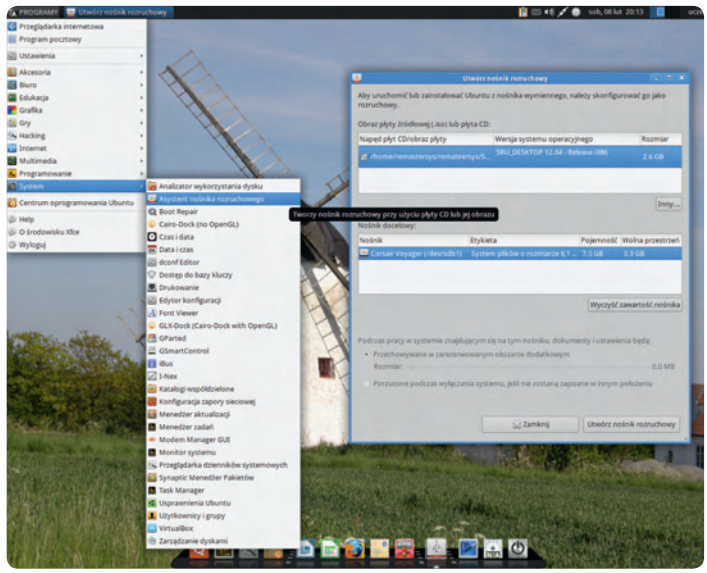
(Tak powinien wyglądać program, kiedy na kluczu USB skasujemy wszystkie partycje, a obok:
jak tworzyć partycje, jakich potrzebujemy)





4 Po poprawnym tworzeniu partycji powinniśmy zobaczyć obraz mniej więcej taki:

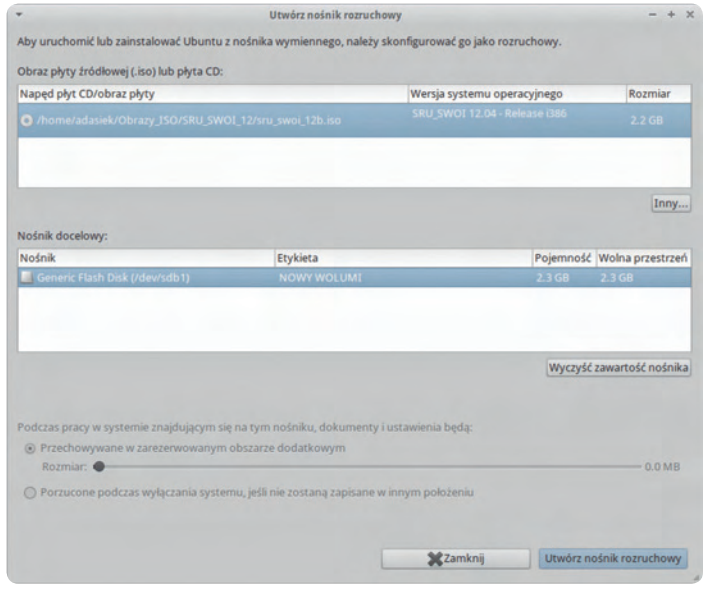
5 Uruchom program Asystent nośnika rozruchowego:



6 W programie wybierz plik ISO z obrazem;

7 Wybierz urządzenie (klucz USB), na jakim chcesz zainstalować obraz;

8 Zatwierdź, klikając w przycisk w prawym dolnym rogu okna;



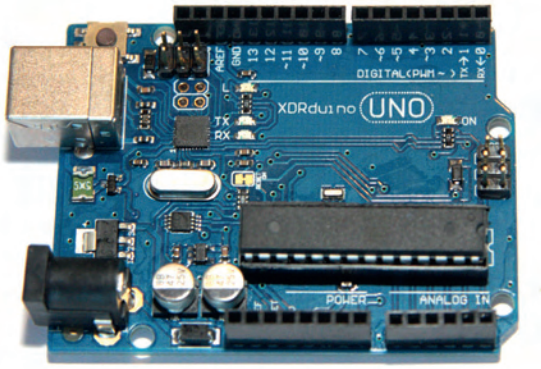
Uwaga!
Jeśli w oknie programu „Asystent nośnika rozruchowego" będzie aktywna opcja opisana jako: Podczas pracy w systemie znajdującym się na tym nośniku, dokumenty i ustawienia będą: wówczas należy zaznaczyć drugą opcję, czyli **Porzucone podczas wyłączania systemu...**

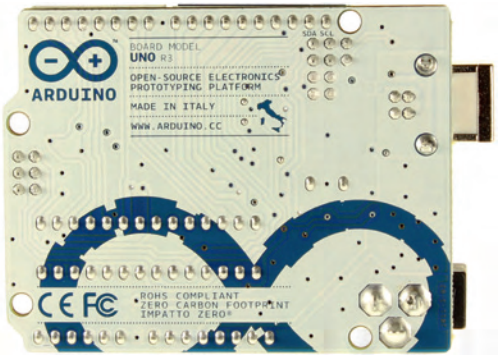
Powodzenia!

04 Instrukcja korzystania z modułu-interfejsu Arduino

Krzysztof Bytow

Arduino powstało we Włoszech w roku 2005 jako projekt typu OpenHardware z myślą o studentach robotyki i kierunków technicznych. W skład wchodzi fizyczna platforma, stworzona w oparciu o prostą płytkę z mikrokontrolerem oraz środowisko programistyczne służące do tworzenia oprogramowania. Arduino jest w pełni interaktywne, pozwala na podłączenie oraz odczyt wartości z czujników, przełączników oraz innych urządzeń wejściowych. Pozwala także na podłączenie peryferii, którymi może sterować, takich jak np. diody LED, silniczki, buzzery itp. Projekt oparty na Arduino może działać w połączeniu z komputerem, komunikując się z przeróżnym oprogramowaniem. Może także stanowić niezależny, działający samodzielnie moduł.





Budowa i opis wyprowadzeń Arduino UNO R3

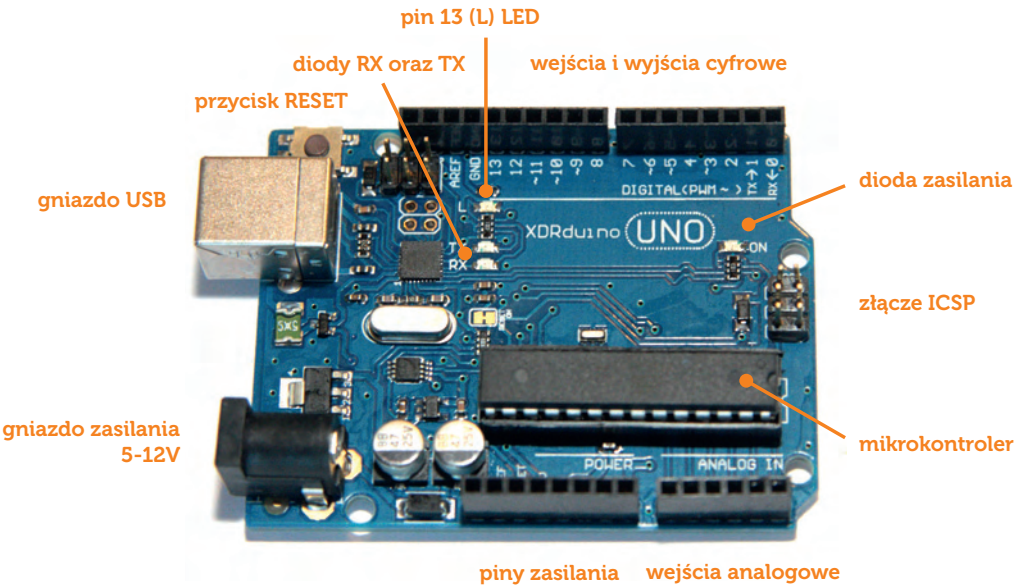
Arduino jest płytką uruchomieniową opartą o układ ATmega328. Programowaną, jak i zasilaną z portu USB (dodatkowo układ wyposażono w gniazdo zasilania 3,5mm – napięcie od 5 do 12 V). Port szeregowy USB służy także do transmisji danych pomiędzy komputerem a Arduino. Komunikacja

odbywa się dwukierunkowo (transmisja z komputera do układu w celu załadowania programu do mikrokontrolera Atmega328, napisanego w specjalnej wersji języka Wiring, z kolei dane z modułu-interfejsu do PC wysyłane są w wyniku działania wcześniej załadowanego kodu, mogą to być dane pochodzące z analizy sygnałów pojawiających się na wejściach analogowych lub cyfrowych).

Układ został wyposażony w 14 pinów cyfrowych służących jako wejścia lub wyjścia, sześć z nich może działać w trybie PWM (Pulse-width modulation) – czyli modulacji szerokości impulsu, metodzie polegającej na zmianie wypełnienia sygnału (możemy sterować jasnością diody lub obrotami silnika). Każdy pin może dostarczyć lub odebrać maksymalnie 40 mA i posiada wewnętrzny rezystor 20-50 kΩ (domyślnie odłączony). Płytka dodatkowo została wyposażona w 6 wejść analogowych do mierzenia wartości napięcia w zakresie od 0 do +5V. Konwenter analogowo-cyfrowy Arduino pracuje w rozdzielczości 10-bitowej, co oznacza, że wartości na wejściach reprezentowane są przez liczby od 0 do 1023. Istotne jest, aby polaryzacja na wejściach analogowych była dodatnia (pozytywna). Wejście nie reaguje poprawnie na napięcia ujemne (może to doprowadzić do uszkodzenia układu).

Specyfikacja

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

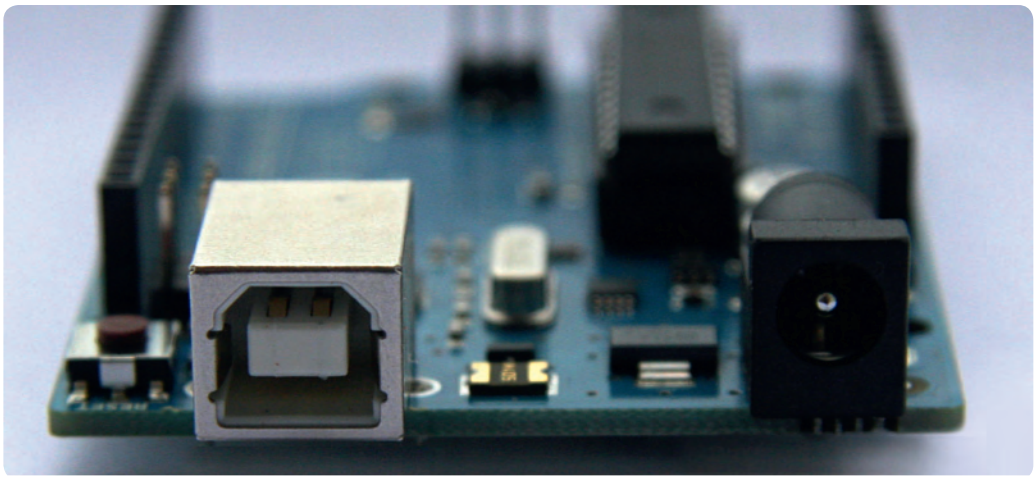


Wyprowadzenia:

- pusty port zarezerwowany do przyszłego użycia
- IOREF – w zależności od stanu na porcie, informuje, na jakich napięciach pracuje dane Arduino
- Reset – stan niski na tym pinie resetuje mikrokontroler

Piny zasilania:

- 3.3V - napięcie stabilizowane +3,3 V
- 5V - napięcie stabilizowane +5 V
- GND – masa (0V)
- GND – masa (0V)
- Vin – napięcie, jakie podłączyliśmy do gniazda zasilania



Gniazdo USB

Gniazdo zasilania

Wejścia analogowe:
- od A0 do A5 – napięcia od 0 do 5 Voltów
- Funkcja "analogRead" zwraca wartość od 0 do 1023 w zależności od napięcia

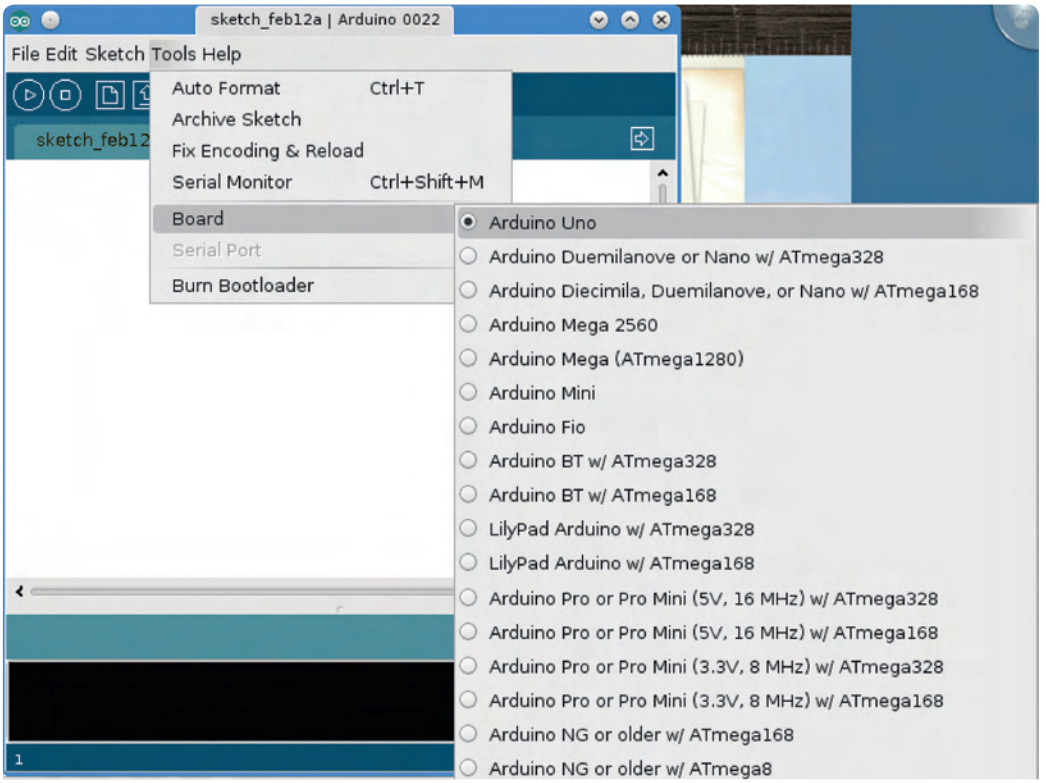
Wejścia-wyjścia cyfrowe:
- 13 pinów – służy jako cyfrowe wejście lub wyjście, obsługiwane za pomocą funkcji "digitalRead", "digitalWrite" oraz "pinMode", mogą się pojawić napięcia stanów logicznych, czyli 0 lub w przybliżeniu 5 V.
- PWM widniejący przy niektórych pinach, oznacza, że na ich wyjściu może pojawić się prąd o przebiegu fali prostokątnej z różnym stopniem wypełnienia. Steruje tym funkcja "analogWrite". Wykorzystać można do regulacji jasności podłączonej do pinu diody świecącej.

Niektóre piny wyjściowe pełnią także inną rolę:
- 0 (RX) i 1 (TX) używany do odbioru (RX) i nadawania (TX) TTL dane szeregowe. Te są podłączone do odpowiednich pinów ATmega8U2 USB-TTL układ szeregowy.
- 2 i 3 te piny mogą być skonfigurowane do wyzwalania przerwania na niską wartość, zboczem narastającym lub opadającym, lub zmiany wartości.
- 3, 5, 6, 9, 10 i 11 -PWM
- 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) - SPI
- 13 pin - wbudowana dioda LED. Gdy pin jest w stanie wysokim, dioda świeci się, gdy pin jest w stanie niskim, to dioda nie świeci.
- A4 (SDA) i A5 (SCL) – obsługa komunikacji TWI (dodatkowo wyprowadzone obok [nad] AREF od wersji Arduino UNO R3)
- AREF – napięcie odniesienia dla wejść analogowych

Diody kontrolne LED:
- LED ON – świeci się, kiedy Arduino jest zasilane z dowolnego źródła (będzie świecić także, jeśli nie wgrano do procesora żadnego programu)
- dioda L – sprzężona z cyfrowym pinem nr13 w Arduino, w przypadku pojawienia się stanu wysokiego (cyfrowa 1), dioda świeci, w przeciwnym przypadku dioda pozostaje zgaszona
- diody RX i TX – migają w momencie transmisji danych „do" (RX) lub „z" (TX) Arduino poprzez złącze USB (lub piny cyfrowe 0 i 1);

Podłączenie i programowanie układu

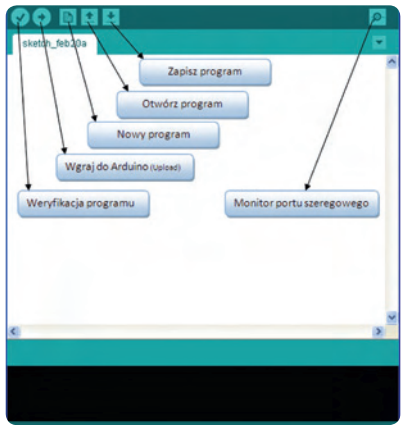
Uruchomienie układu Arduino sprowadza się do połączenia odpowiednim przewodem USB (tzw. A-B) modułu z komputerem. Uruchomiony układ będzie sygnalizował swoje działanie świecącą na zielono diodą (LED ON). Aby przystąpić do pracy, należy jeszcze zaopatrzyć się w odpowiedni edytor Arduino IDE, który można pobrać ze strony http://arduino.cc/en/Main/Software. Aby skonfigurować środowisko Arduino IDE, wybieramy z zakładki Tools (narzędzia) pozycję Board (płyta), z rozwiniętych pozycji wybieramy wersję posiadanego układu → w tym przypadku Arduino UNO. Następnie wybieramy port, z którym komunikuje się Arduino (Tools → Serial Port → /dev/ttACM0 – w zależności od posiadanego systemu i podpiętych urządzeń wybór może się różnić).



Programy dla Arduino są napisane głównie w języku C/C++. Aby otrzymać gotowy do uruchomienia program, użytkownicy muszą zdefiniować jedynie dwie funkcje,:

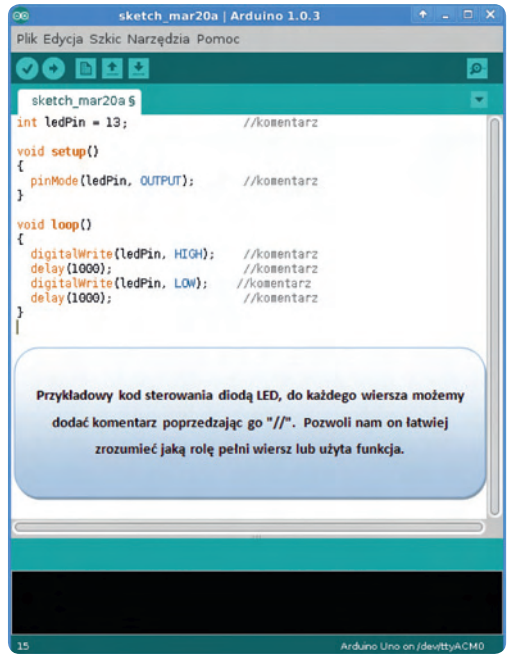
- setup() – funkcja wykonywana raz, na początku działania programu, wykorzystywana najczęściej do ładowania ustawień,
- loop() – funkcja wywoływana cyklicznie, przez cały okres działania programu, czyli do czasu odłączenia zasilania od układu lub wciśnięcia reset.

Opis interfejsu Arduino IDE

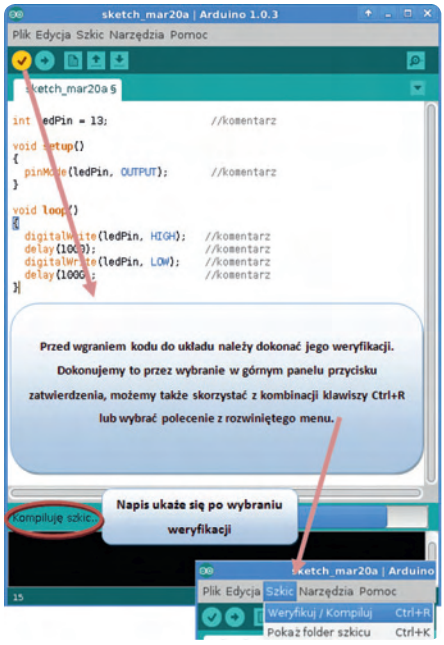


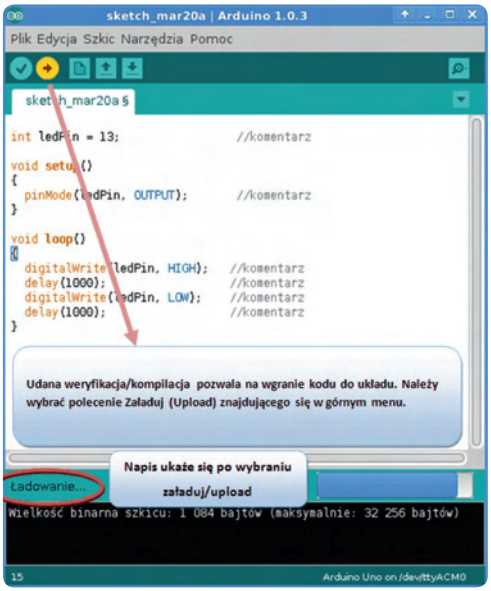
Na przykładzie sterowania wbudowaną w układ diodą analiza krok po kroku czynności niezbędnych do uruchomienia kodu.

kod możemy wpisać, skopiować/wkleić lub otworzyć gotowy projekt

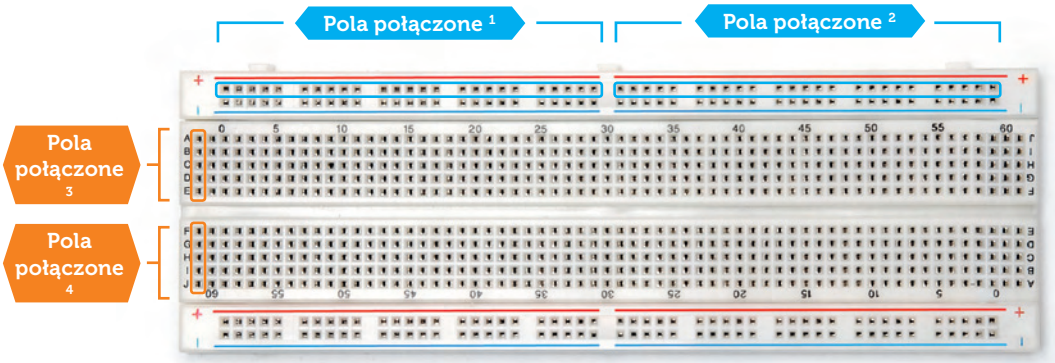


sprawdzanie kodu, czy nie zawiera błędów





Uniwersalna płytka prototypowa – plastikowa podstawka z licznymi otworami, wewnątrz których znajduje się materiał przewodzący uformowany w rodzaj minizacisku, dzięki czemu elementy włożone w otwór nie wypadają samoczynnie.

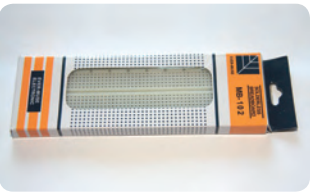
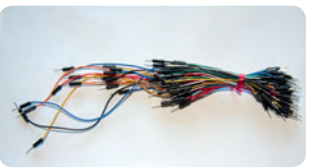


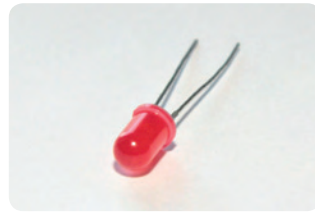
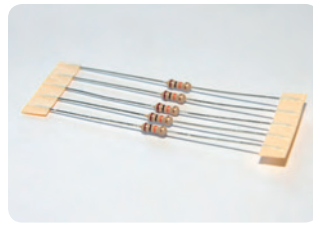
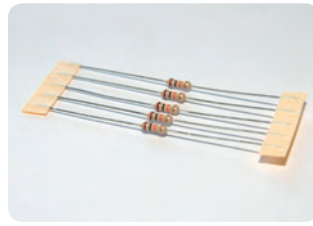
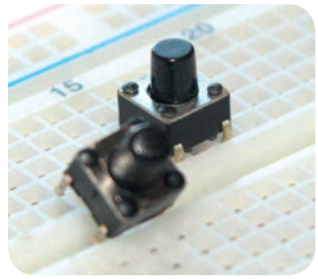
Uwagi:

- Pola połączeniowe z indeksem 1 – zaznaczony obszar na zielono oznacza, iż jak w pierwszym pinie podłączymy +5V, to na całej długości będziemy mieli dane napięcie (podobnie z indeksem 2);
- Pola połączeniowe z indeksem 3 – zaznaczony obszar na czerwono oznacza, iż piny od A do E są połączone (podobnie z indeksem 4);
- Pola połączeniowe z indeksem 1 nie są połączone z Polami połączeniowymi z indeksem 2 – jeśli zaistnieje potrzeba, należy połączyć je przewodem;
- Pola połączeniowe z indeksem 3 nie są połączone z Polami połączeniowymi z indeksem 4 – jeśli zaistnieje potrzeba, należy połączyć je przewodem.

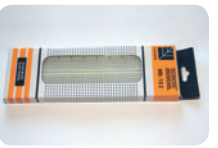
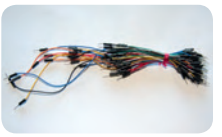
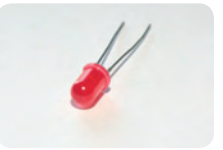
Specyfikacje zestawów mechatronicznych

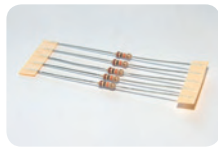
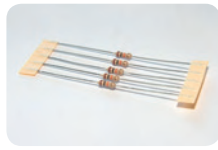
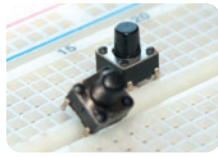
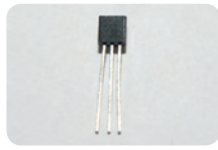
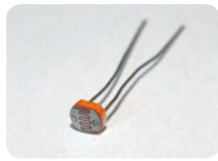
Specyfikacja elementów zestawu 0

L.p.	Pozycja	Sztuk	Uwagi
1.	Arduino UNO R3 lub kompatybilny	1	
2.	kabel USB 2.0 typu A/B min. długość 1,5m do podłączenia Arduino z komputerem	1	
3.	duża płytka stykowa 830 otworów lub kompatybilna	1	
4.	komplet kabli do płytki stykowej	10	
5.	pojemnik na drobiazgi	1	

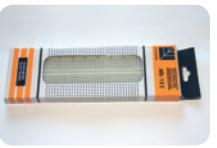
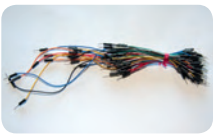
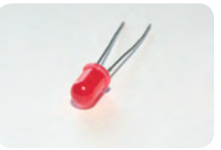
6.	dioda elektroluminescencyjna o średnicy 5mm czerwona Uf czerwona = 2V ; If = 20mA ; kąt świecenia 80°; jasność ok 80-100mcd;	1	
7.	opornik - rezystor 220 Ω	1	
8.	opornik – rezystor 10 kΩ	1	
9.	mały przycisk - button	1	

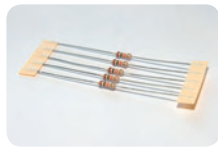
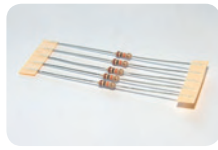
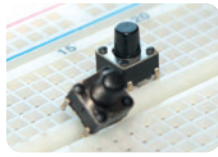
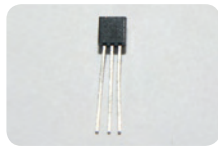
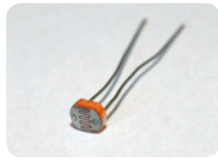
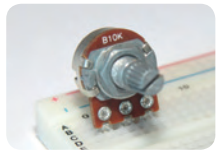
Specyfikacja elementów zestawu A

L.p.	Pozycja	Sztuk	Uwagi
1.	Arduino UNO R3 lub kompatybilny	1	
2.	kabel USB 2.0 typu A/B min. długość 1,5mdo podłączenia Arduino z komputerem	1	
3.	duża płytka stykowa 830 otworów lub kompatybilna	1	
4.	komplet kabli do płytki stykowej	65	
5.	pojemnik na drobiazgi	1	
6.	dioda elektroluminescencyjna o średnicy 5mm czerwona, Uf czerwona = 2V ; If = 20mA ; kąt świecenia 80°; jasność ok 80-100mcd;	3	
7.	dioda RGB o średnicy 5mm ze wspólną katodą Uf czerwona = 2V ; Uf zielony = 2,5V ; Uf niebieska = 3,3V; If = 22mA ; kąt świecenia 60°; jasność ok 60mcd; 4 wyprowadzenia	1	

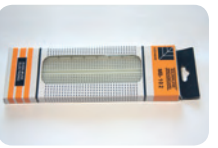
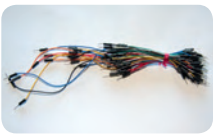
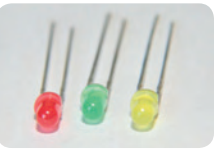
8.	opornik - rezystor 220 Ω	3	
9.	opornik – rezystor 10 kΩ	2	
10.	mały przycisk - button	2	
11.	czujniki temperatury (MCP-9700)	1	
12.	fotorezystor	1	

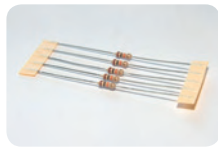
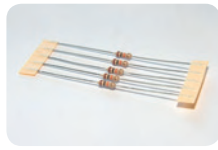
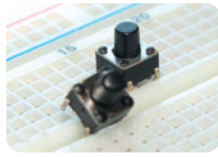
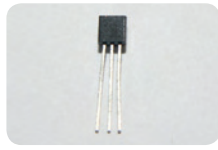
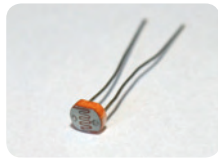
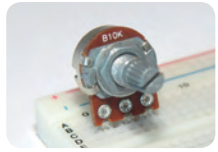
Specyfikacja elementów zestawu B

L.p.	Pozycja	Sztuk	Uwagi
1.	Arduino UNO R3 lub kompatybilny	1	
2.	kabel USB 2.0 typu A/B min. długość 1,5mdo podłączenia Arduino z komputerem	1	
3.	duża płytka stykowa 830 otworów lub kompatybilna	1	
4.	komplet kabli do płytki stykowej	65	
5.	pojemnik na drobiazgi	1	
6.	dioda elektroluminescencyjna o średnicy 5mm czerwona, Uf czerwona = 2V ; If = 20mA ; kąt świecenia 80°; jasność ok 80-100mcd;	6	
7.	dioda RGB o średnicy 5mm ze wspólną katodą Uf czerwona = 2V ; Uf zielony = 2,5V ; Uf niebieska = 3,3V; If = 22mA ; kąt świecenia 60°; jasność ok 60mcd; 4 wyprowadzenia	1	

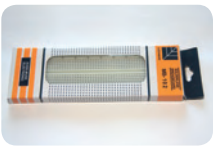
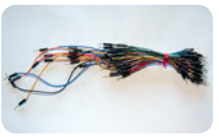
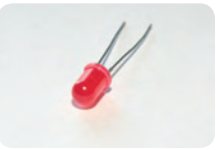
8.	opornik - rezystor 220 Ω	6	
9.	opornik – rezystor 10 kΩ	1	
10.	mały przycisk - button	4	
11.	czujniki temperatury (MCP-9700)	1	
12.	fotorezystor	1	
13.	wyświetlacz alfanumeryczny LCD 2x16 tekstowy ze sterownikiem zgodnym z HD44780 - z wluto-wanym goldpinem (drabinką)	1	
14.	potencjometr montażowy 10 kΩ	1	
15.	buzzer bez generatora 5V	1	

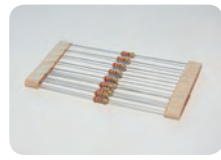
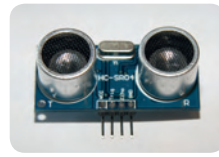
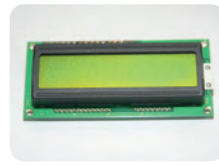
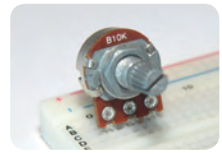
Specyfikacja elementów zestawu B S4A

L.p.	Pozycja	Sztuk	Uwagi
1.	Arduino UNO R3 lub kompatybilny	1	
2.	kabel USB 2.0 typu A/B min. długość 1,5m do podłączenia Arduino z komputerem	1	
3.	duża płytka stykowa 830 otworów lub kompatybilna	1	
4.	komplet kabli do płytki stykowej	65	
5.	pojemnik na drobiazgi	1	
6.	dioda elektroluminescencyjna o średnicy 5mm Uf czerwona = 2V ; If = 20mA ; kąt świecenia 80°; jasność ok 80-100mcd; 2 szt. czerwona, 2 szt. żóta, 2 szt. zielona	6	
7.	dioda RGB o średnicy 5mm ze wspólną katodą Uf czerwona = 2V ; Uf zielony = 2,5V ; Uf niebieska = 3,3V If = 22mA ; kąt świecenia 60°; jasność ok 60mcd; 4 wyprowadzenia	1	

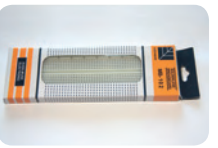
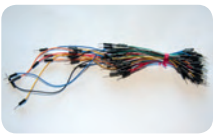
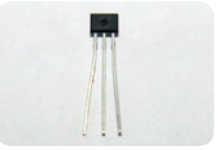
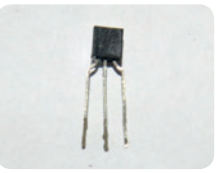
8.	opornik - rezystor 220 Ω	5	
9.	opornik – rezystor 10 kΩ	2	
10.	mały przycisk - button	2	
11.	czujniki temperatury (MCP-9700)	1	
12.	fotorezystor	1	
13.	wyświetlacz alfanumeryczny LCD 2x16 tekstowy ze sterownikiem zgodnym z HD44780 - z wluto-wanym goldpinem (drabinką)	1	
14.	potencjometr montażowy 10 kΩ	1	
15.	buzzer bez generatora 5V	1	

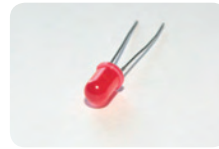
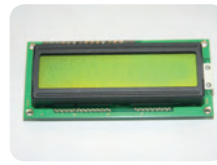
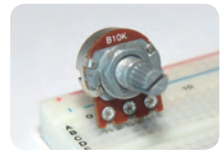
Specyfikacja elementów zestawu C1

L.p.	Pozycja	Sztuk	Uwagi
1.	Arduino UNO R3 lub kompatybilny	1	
2.	kabel USB 2.0 typu A/B min. długość 1,5m do podłączenia Arduino z komputerem	1	
3.	duża płytka stykowa 830 otworów lub kompatybilna	1	
4.	komplet kabli do płytki stykowej	65	
5.	pojemnik na drobiazgi	1	
6.	dioda elektroluminescencyjna o średnicy 5mm czerwona Uf czerwona = 2V ; If = 20mA ; kąt świecenia 80°; jasność ok 80-100mcd;	10	
7.	dioda RGB o średnicy 5mm ze wspólną katodą Uf czerwona = 2V ; Uf zielony = 2,5V ; Uf niebieska = 3,3V If = 22mA ; kąt świecenia 60°; jasność ok 60mcd; 4 wyprowadzenia	1	

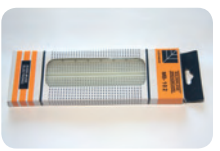
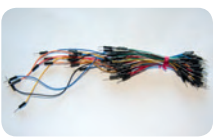
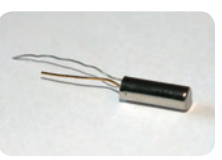
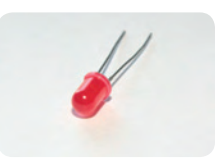
8.	opornik - rezystor 220 Ω	10	
9.	opornik – rezystor 4,7 kΩ	1	
10.	mały przycisk - button	3	
11.	czujnik temperatury Dallas DS18B20	2	
12.	czujnik ultradźwiękowy HC-SR04	1	
13.	wyświetlacz alfanumeryczny LCD 2x16 tekstowy ze sterownikiem zgodnym z HD44780 - z wlutowanym goldpinem (drabinką)	1	
14.	potencjometr montażowy 10 kΩ	1	
15.	Serwo TowerPro SG-90	1	

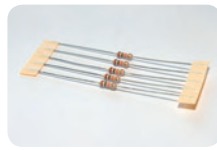
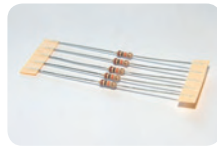
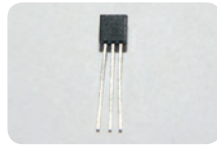
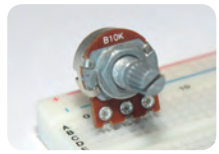
Specyfikacja elementów zestawu C2

L.p.	Pozycja	Sztuk	Uwagi
1.	Arduino UNO R3 lub kompatybilny	1	
2.	kabel USB 2.0 typu A/B min. długość 1,5m do podłączenia Arduino z komputerem	1	
3.	duża płytka stykowa 830 otworów lub kompatybilna	1	
4.	komplet kabli do płytki stykowej	65	
5.	pojemnik na drobiazgi	1	
6.	hallotron CS3140	6	
7.	tranzystor MPS2222A	1	

8.	dioda elektroluminescencyjna o średnicy 5mm Uf czerwona = 2V ; If = 20mA ; kąt świecenia 80°; jasność ok 80-100mcd;	5	
9.	dioda RGB o średnicy 5mm ze wspólną katodą Uf czerwona = 2V ; Uf zielony = 2,5V ; Uf niebieska = 3,3V If = 22mA ; kąt świecenia 60°; jasność ok 60mcd; 4 wyprowadzenia	1	
10.	opornik - rezystor 220 Ω	6	
11.	magnes neodymowy	1	
12.	mały przycisk - button	2	
13.	wyświetlacz alfanumeryczny LCD 2x16 tekstowy ze sterownikiem zgodnym z HD44780 - z wluto-wanym goldpinem (drabinką)	1	
14.	potencjometr montażowy 10 kΩ	1	
15.	buzzer bez generatora 5V	1	

Specyfikacja elementów zestawu C3

L.p.	Pozycja	Sztuk	Uwagi
1.	Arduino UNO R3 lub kompatybilny	1	
2.	kabel USB 2.0 typu A/B min. długość 1,5m do podłączenia Arduino z komputerem	1	
3.	duża płytka stykowa 830 otworów lub kompatybilna	1	
4.	komplet kabli do płytki stykowej	65	
5.	pojemnik na drobiazgi	1	
6.	kulowy czujnik nachylenia 60 stopni	1	
7.	dioda elektroluminescencyjna o średnicy 5mm Uf czerwona = 2V ; If = 20mA ; kąt świecenia 80°; jasność ok 80-100mcd;	1	

8.	dioda RGB o średnicy 5mm ze wspólną katodą Uf czerwona = 2V ; Uf zielony = 2,5V ; Uf niebieska = 3,3V If = 22mA ; kąt świecenia 60°; jasność ok 60mcd; 4 wyprowadzenia	1	
9.	opornik - rezystor 220 Ω	6	
10.	opornik – rezystor 4,7 kΩ - 2szt. 1 kΩ - 2szt. 10 kΩ - 2szt.	6	
11.	mały przycisk - button	3	
12.	czujniki temperatury (MCP-9700)	2	
13.	potencjometr montażowy 10 kΩ	1	
14.	buzzer bez generatora 5V	1	

05

Implementacje do realizacji na kołach zainteresowań IT

Stanisław Ubermanowicz

W formowaniu kompetencji infotechnicznych kluczową rolę odgrywają *implementacje*. To tytułowe pojęcie w Strategii Wolnych i Otwartych Implementacji mieści w sobie całe spektrum znaczeń – od koncepcji rozwiązania problemu, poprzez proces urzeczywistniania pomysłu, aż do końcowego wytworu realizującego założoną funcjonalność. W dziedzinie dydaktyki implementacja służy jako środek-metoda uczenia się, na gruncie inżynierii oznacza proces projektowania, programowania lub konstruowania, a w obszarze infotechniki określa finalny wytwór informatyczny lub mechatroniczny.

Trafny dobór trudności zadań i zakresu samodzielnych czynności uczniów przy tworzeniu implementacji jest kluczowym zadaniem dla trenerów.

Wszystkie te specyficzne, cenne edukacyjnie walory implementowania wykorzystywane są podczas realizacji kół infotechnicznych. Implementacje mają jednak także tę właściwość, że możliwych rozwiązań wyznaczonego zadania jest bardzo wiele. Z tego względu zawarte w książce i na płycie opisy implementacji należy traktować jako przykłady do wykorzystania w starannie przemyślany sposób – nie jako gotowce do powielania, lecz jako materiał pomocniczy dla trenerów, którzy na tej podstawie wyznaczają zróżnicowany zakres ćwiczeń dopasowanych do możliwości uczniów.

W pierwszym podejściu trener wydobywa z opisu **istotę zadania implementacyjnego**. Zadaniem tym może być np. wykonanie prostej grafiki, wizualizacji, animacji, gry, interfejsu, układu pomiarowego lub sygnalizacyjnego. W modułach zaawansowanych zadaniem może być całościowe napisanie kodów realizujących procedury, funkcje lub złożone struktury w danym języku programowania. Mimo że implementacje opublikowano w pełnych wersjach, jako przykłady rozwiązań, to czynności uczniów nie powinny polegać na przepisywaniu bądź wgrywaniu kompletnych kodów źródłowych.

Na podstawie opisu każdej konkretnej implementacji, dedykowanej dla określonych grup wiekowych i rodzajów szkół, trener wyznacza na dane zajęcia adekwatne dla uczniów zadania, obejmujące albo tworzenie od podstaw wszystkich elementów, albo uzupełnianie celowo usuniętych obiektów lub fragmentów kodu.

Konieczne jest optymalne wypośrodkowanie pomiędzy zakresem dostarczonych uczniom niezbędnych półproduktów, a pozostawieniem pola do wykazania się przez nich własną inwencją.

Im bardziej dostarczona postać implementacji jest kompletna, tym bardziej zadanie powinno polegać na daleko idącej samodzielnej modyfikacji rozwiązania. Taka forma ma zastosowanie zwłaszcza przy tworzeniu modułów-interfejsów. Uczeń najpierw montuje układ wzorcowy według schematu, a następnie rozbudowuje elementy i modyfikuje oprogramowanie sterujące. Na zajęciach dotyczących projektowania graficznego należy podać wyłącznie treść zadania, określając to, co ma powstać, i pozostawiając pełną swobodę dla indywidualnej twórczości. W trudniejszych zadaniach programistycznych część kodów może być wczytywana, część uzupełniana, a ponadto – trener winien omawiać tylko takie fragmenty kodu, które są ściśle związane z tematem danej jednostki zajęć. Resztę warto pozostawić do **samodzielnego interpretowania** przez uczniów, gdyż umiejętność odczytywania znaczenia struktur i instrukcji w kodzie jest ważnym składnikiem kompetencji infotechnicznych.

W zróżnicowanych postaciach opisów i rodzajach implementacji zawarte są pewne **wskazówki dla trenerów** co do sposobów i zakresów wyznaczania uczniom zadań. Jedną z form podpowiedzi jest przygotowanie do danej tematyki zajęć implementacji alternatywnych. Ma to miejsce w propozycjach zajęć mechatronicznych, kiedy to podobne tematycznie wersje różnią się stopniem rozbudowania układu. W modułach programistycznych zróżnicowany zakres ćwiczeń sugerowany jest poprzez zadania rozszerzające sformułowane w niektórych Konspektach-scenariuszach. Wówczas nie podaje się jednak gotowych rozwiązań, gdyż w większości są to przykłady formułowania zadań do wykonania już poza zajęciami stacjonarnymi. Inną formą zaleceń jest opracowanie kodów źródłowych w wersji kompletnej dla trenera oraz w wersji z celowo wyznaczonymi miejscami do uzupełnienia przez uczniów. Bez uzupełnienia taka implementacja nie działa, zatem uczeń chcący ją uruchomić musi najpierw prawidłowo wpisać brakujące instrukcje.

Kody źródłowe są głównie dla trenerów, którzy decydują, jaki ich zakres udostępnić uczniom. Natomiast instrukcje mechatroniczne są przeznaczone wprost dla uczniów.

Różnorodność implementacji opracowanych na potrzeby realizacji kół infotechnicznych wynika także z odmiennego podejścia do optymalnych metod i zagadnień tematycznych na danym etapie rozwoju uczniów. Otóż we wczesnej fazie wdrażania do problematyki projektowania najważniejsze jest **upoglądowienie**, a to uzyskuje się poprzez dominację reprezentacji wizualnych. Tak jest właśnie w propozycjach implementacji dla szkół podstawowych, gdzie zaleca się głównie zadania polegające na cyfrowej obróbce obrazów, na tworzeniu grafiki oraz na generowaniu i interpretacji wykresów. Umiejętność odczytywania wykresów jest m.in. weryfikowana na zewnętrznym sprawdzianie w klasach szóstych, stąd praktyczny walor takiej tematyki zajęć. Proponowane dla dzieci zajęcia z mechatroniki polegają na budowaniu układów, które także pełnią rolę poglądową, gdyż optycznie sygnalizują określone stany napięć. Ponadto elementarne oprogramowywanie mikrokontrolera odbywa się w graficznym środowisku wizualno-skryptowym Scratch for Arduino (S4A).

W początkowych etapach uczenia się programowania najważniejsze są metody poglądowe z wizualizacją efektów działania kodu źródłowego.

Na poziomie szkół gimnazjalnych proponuje się propedeutyczne wdrażanie do problematyki programowania, głównie poprzez **wizualizację procedur** obsługujących obiekty ekranowe. Punktem wyjścia jest tu także tworzenie obrazów, a następnie ożywianie obiektów za pomocą skryptów zmieniających ich położenie oraz właściwości. Znakomitym zintegrowanym środowiskiem narzędziowym do programowania wizualno-skryptowego jest Scratch, w którym obiekty zwane „duszkami" są sterowane przez automatycznie integrujące się polecenia, formułowane po polsku. A co najważniejsze – gotowe instrukcje języka programowania dostępne są i układane w graficznej formie dopasowujących się puzzli, ułatwiających budowę prawidłowej struktury kodu źródłowego. Moduły mechatroniczne dla gimnazjalistów realizują zagadnienia sterowania, odczytywania stanów i wskaźnikowania z użyciem środowiska graficznego Scratch for Arduino.

Dla szkół ponadgimnazjalnych niesprofilowanych adresowane są moduły wdrażające do programowania, wykorzystujące metodę **operacjonalizacji pojęć informatycznych**, będących w znacznej mierze kategoriami abstrakcyjnymi. Działania realizowane przez implementacje ułatwiają zrozumienie najważniejszych struktur języka programowania oraz zasad realizacji algorytmów, strategii wygranej i sztucznego intelektu. Także i tu kluczowe znaczenie ma wizualizacja, ale dochodzi ponadto silny środek motywujący, jakim jest tworzenie gier logicznych. Na tym poziomie – oprócz pracy w Scratchu – proponowane jest też programowanie wizualno-obiektowo-zdarzeniowe w zintegrowanym środowisku graficznym Lazarus z językiem FreePascal. Język ten jest dopuszczony do wykonywania zadań na maturze z informatyki, dlatego poznanie jego struktur i instrukcji ma praktyczne znaczenie w przygotowaniu się do zdania egzaminu z tej dziedziny i do dalszego studiowania infotechniki. Implementacje do modułów mechatronicznych wykorzystują dwie wersje środowisk współpracy z układem Arduino – jeden interfejs zorientowany jest na emulację interaktywnego terminala, a drugi na formę graficzno-skryptową. Polecane układy mechatroniczne służą do pomiarów światła i temperatury.

W miarę nabywania umiejętności infotechnicznych przechodzi się coraz bardziej na zadania wymagające operowania na obiektach abstrakcyjnych.

Na potrzeby szkół sprofilowanych infotechnicznie opracowano implementacje oparte bardziej na **metodach wyobrażeniowych** niż poglądowych. Uczniowie z tych szkół wybrali już taki kierunek specjalizacji IT, dlatego dla nich ważniejsze jest doskonalenie umiejętności programowania bądź poznanie na kołach zainteresowań innego niż na lekcjach języka. Z tego względu implementowanie odbywa się tam na wyższym poziomie myślenia abstrakcyjnego, w metodykach programowania obiektowego i imperatywnego. Generowanie obiektów i struktur polega głównie na pisaniu kodu źródłowego, bez wsparcia graficznego, bez *widżetów* narzędziowych, dostarczających gotowe obiekty funkcjonalne. Taki sposób implementowania algorytmów wyłącznie z poziomu kodu wymagany jest m.in. na egzaminie maturalnym, a później na studiach. Proponowane tu do wyboru implementacje są zwięzłymi kursami konstruowania głównych struktur w języku C lub w języku Python oraz przykładami wzbogacania HTML o język JavaScript z biblioteką jQuery. Dla tych szkół opracowano też alter-

natywne moduły mechatroniczne, ilustrujące zaawansowane sposoby wykorzystania układów z mikrokontrolerem do budowy mierników różnych parametrów fizycznych, do konstruowania serwomechanizmów, sterowników prądowych i transmisyjnych.

Należy w tym miejscu podkreślić, że bloki Modułów A i B, dedykowanych na konkretne poziomy i rodzaje szkół niesprofilowanych, są przeznaczone głównie dla uczniów początkujących w dziedzinie programowania i konstruowania. Chodzi o poszerzenie rzeszy uczniów, których zaciekawiłaby ta problematyka. Z tego względu zasadniczy cykl zajęć służących uczniom do zapoznania się z zagadnieniami i sprawdzenia swego potencjału powinien składać się z minimum 12 jednostek dydaktycznych.

Warto jednakże realizować dalsze lub alternatywne Moduły dla tych uczniów, którzy chcą kontynuować udział w zajęciach dodatkowych. Z uczniami uzdolnionymi, przygotowanymi w cyklu zasadniczym, można z powodzeniem przeprowadzać także zajęcia dedykowane na wyższy etap szkolnictwa. Warto przy tym pamiętać, aby **przeplatać tematykę** programowania z mechatroniką. To właśnie konstruowanie interfejsów sprawia uczniom we wszystkich rodzajach szkół najwięcej satysfakcji, lecz uczenie się programowania jest tu równie ważne.

Trenerzy mają możliwość ułożenia własnego Programu, poprzez wybór Modułów do realizacji na kołach zainteresowań na podstawie analizy treści Konspektów-scenariuszy albo opisów zadań implementacyjnych i przykładowych rozwiązań. Wstępne rozpoznanie i dobór ułatwia tabela, zawierająca zestawienie oznaczeń kodowych, nazw implementacji i tematyki zajęć oraz wskazanie środowisk pracy i zagadnień infotechnicznych.

Wykaz implementacji, środowisk i zagadnień

Kod	Nazwa Implementacji	Temat Konspektu-scenariusza	Środowisko: zagadnienia
Moduły dla szkół podstawowych			
01.1	Instrukcje	Wprowadzenie do środowiska SRU	01 - SRU, JAlbum, TuxPaint: obróbka zdjęć, grafika
01.2	Album zdjęć	Album - wspomnienia z wakacji	
01.3	Reklama	Tworzenie reklamy ośrodka wczasowego	
02.1	Diagramy kolumnowe	Oszczędzaj energię elektryczną	02 - Calc, Firefox, Google: tabele, diagramy, Internet
02.2	Diagramy kołowe	Zdrowy tryb życia	
02.3	Internet	Wyszukiwanie informacji w sieci	
03.1	LED	Programowa sygnalizacja diodą LED	03 - S4A, Arduino: sygnalizacja, sterowanie, losowanie
03.2	Sterowanie klawiaturą w S4A	Sterowanie diody LED z klawiatury	
03.3	Wybór losowy	Losowy czas świecenia diody LED	
03.4	Button	Wyświetlanie stanu przycisku Button	03 - S4A, Arduino: wyświetlanie, przełączanie, regulacja
03.5	Button i LED	Przełączanie diody LED przyciskiem	
03.6	Jasno i ciemno	Programowa regulacja jasności diody	

Moduły dla szkół gimnazjalnych

A1.1	Poszukiwanie skarbów	Wprowadzenie do środowiska SRU	A1 - Linux, SRU: awatary, graffiti, algorytm
A1.2	Obraz wektorowy	Tworzenie awatarów	
A1.3	Graffiti	Implementacja Graffiti	
A1.4	Stworzenie algorytmu map myśli	Algorytm na mapie myśli	
A2.1	Robot biedronka	Animacja biedronki	A2 - Scratch: animacja, gra, labirynt, konwersja
A2.2	Animacja, gra logiczna - zgadywanka	Moja własna gra komputerowa	
A2.3	Labirynt	Labirynt interaktywny	
A2.4	Liczyć jak komputer	Różne systemy liczbowe	
A3.1	Środowisko Scratch S4A i moduł-interfejs	Scratch i obsługa modułu-interfejsu	A3 - S4A, Arduino: interfejs, czujnik, wskaźnik, gra
A3.2	Pomiar temperatury	Pomiar temperatury i wizualizacja RGB	
A3.3	Układ pomiarowy Arduino	Odczytywanie stanu czujników	
A3.4	Rozszerzenie możliwości S4A	Interakcja modułu-interfejsu i środowiska S4A	

Moduły dla szkół ponadgimnazjalnych niesprofilowanych

B1.1	Szyfrowanie	SRU - Jak chronić ważne informacje	B1 - Linux, SRU, Scratch: szyfrowanie, wizualizacja
B1.2	Wybór drogi	Wizualizacja instrukcji warunkowej	
B1.3	Zakręcona mrówka	Wizualizacja pętli	
B1.4	Magiczny operator	Wizualizacja operatorów przypisania	
B1.5	Zamieszanie z kolorami	Ewaluacja prawej strony wyrażeń	B1 - Scratch: skrypty, wyrażenia, gry, sortowanie
B1.6	Kółko i krzyżyk - Scratch	Rozbudowa gry o strategię blokowania	
B1.7	Gra Pong	Rozbudowa gry Pong o nowe elementy	
B1.8	Animacja – sortowanie	Wybrane algorytmy sortowania	
B2.1	Kółko i krzyżyk - Lazarus	Wprowadzenie do środowiska Lazarus	B2 - Lazarus, Free Pascal: gry, automat, zegar
B2.2	Gra w światełka	Gra w zapalone światełka	
B2.3	Gra w życie	Automaty	
B2.4	Zegar binarny	Zegar	
B2.5	Gra „Papier-kamień-nożyce”	Losowanie i porównywanie	B2 - Lazarus, Free Pascal: gry, algorytmy, strategie
B2.6	Układanka alfabetyczna	Rozrzucanie i porządkowanie	
B2.7	Wieże Hanoi	Odkrywanie i stosowanie algorytmu	
B2.8	Gra logiczna NIM	Namiastka sztucznej inteligencji	
B3.1	Środowisko mikrokontrolera	Funkcjonalność modułu-interfejsu	B3 - Arduino IDE: terminal, przetwornik, pomiar, gra
B3.2	Pomiar oświetlenia	Sterowanie z użyciem fotorezystora	
B3.3	Termometr cyfrowy	Pomiar temperatury i wyświetlacz LCD	
B3.4	Pamięć i zręczność	Interakcja człowiek - moduł-interfejsu	
B3.5	Środowisko Scratch S4A i moduł-interfejs	Możliwości S4A i modułu-interfejsu	B3 - S4A, Arduino: interfejs, regulacja, sygnalizacja
B3.6	Przetwornik analogowo-cyfrowy	Działanie fotorezystora i potencjometru	
B3.7	Funkcjonalność modułu-interfejsu	Środowisko mikrokontrolera	
B3.8	Obsługa wyświetlacza	Termometr cyfrowy	

Moduły dla szkół sprofilowanych infotechnicznie - język C

C1.1	Nauka języka C - zmienne	Struktura programu, zmienne oraz typy danych	C1 - język C: zmienne, wyrażenia, pętle
C1.2	Nauka języka C - wyrażenia *if-else*	Wyrażenia warunkowe *if-else*	
C1.3	Nauka języka C - wyrażenie *switch*	Wrażenie warunkowe *switch{case:. . .;}*	
C1.4	Nauka języka C - pętle *while*	Pętle - *while()* i *do {} while()*	
C2.1	Nauka języka C - tablice i pętla *for*	Tablice i pętla *for()*	C2 - język C: tablice, funkcje, struktury, wskaźniki
C2.2	Nauka języka C - funkcje	Funkcje	
C2.3	Nauka języka C - struktury i unie	Struktury i unie	
C2.4	Nauka języka C - wskaźniki	Wskaźniki	
C3.1	Budowa układów i programowanie modułu	Środowisko mikrokontrolera	C3 - Arduino: tranzystor, hallotron, zegar
C3.2	Zwiększenie możliwości wyjścia cyfrowego	Tranzystor NPN	
C3.3	Budowa, działanie i zastosowanie hallotronu	Pole magnetyczne	
C3.4	Pomiar czasu i wyświetlacz LCD	Zegar	

Moduły dla szkół sprofilowanych infotechnicznie - język Python

C1.1	Nauka języka Python - zmienne	Zmienne, ich typy i dane	C1 - język Python: zmienne, wyrażenia, pętle, dane
C1.2	Nauka języka Python - wyrażenia warunkowe	Wyrażenia warunkowe	
C1.3	Nauka języka Python - pętle *while*	Pętle - *while*	
C1.4	Nauka języka Python - typy danych	Zaawansowane typy danych	
C2.1	Nauka języka Python - pętle *for*	Pętle - *for*	C2 - język Python: pętle, funkcje, wyjątki, klasy
C2.2	Nauka języka Python - funkcje	Funkcje	
C2.3	Nauka języka Python - wyjątki	Wyjątki	
C2.4	Nauka języka Python - klasy	Klasy	
C3.1	Wprowadzenie do środowiska mikrokontrolera	Wykorzystanie wejścia analogowego	C3 - Arduino: omomierz, sterowanie, termometr
C3.2	Budowa prostego omomierza	Pomiar wartości opornika	
C3.3	Sterowanie układem z interfejsu Arduino IDE	Kontrola diody RGB	
C3.4	Prezentacja pomiarów	Prezentacja pomiaru temperatury na RGB	

Moduły dla szkół sprofilowanych infotechnicznie - HTML, JavaScript

C1.1	Podstawy HTML	Podstawy HTML	C1 - HTML, CSS, JavaScript, formularz
C1.2	CSS i box model	CSS i box model	
C1.3	Formularze HTML	Formularze HTML	
C1.4	JavaScript	JavaScript	
C2.1	Podstawy jQuery	Podstawy jQuery	C2 – jQuery, lista, baza danych, Canvas
C2.2	jQuery - zdarzenia i efekty	jQuery - zdarzenia i efekty	
C2.3	Przechowywanie danych	Przechowywanie danych	
C2.4	Canvas	Canvas	
C3.1	Wykorzystanie wejścia analogowego	Środowisko mikrokontrolera	C3 - Arduino: sterowanie, dalmierz, serwo
C3.2	Protokół komunikacyjny 1-wire	Protokół komunikacyjny 1-wire	
C3.3	Ultradźwiękowy pomiar odległości	Pomiar i odczyt odległości na LCD	
C3.4	Sterowanie serwomechanizmem	Sterowanie elementami wykonawczymi	

Przykłady zadań i wytworów implementacyjnych

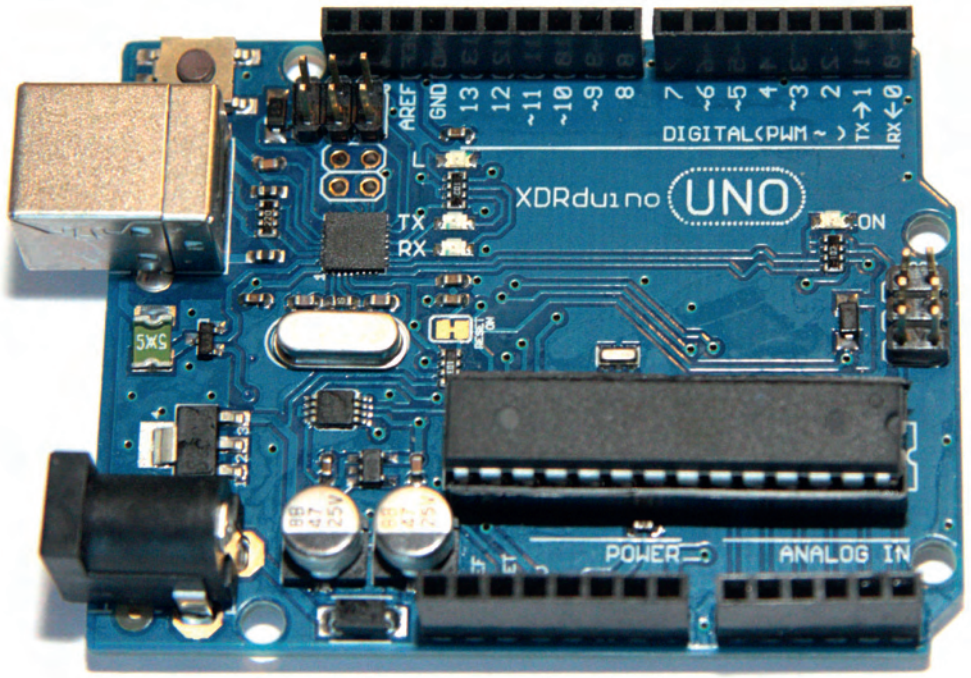
- 03.2 S4A, Arduino: Sterowanie klawiaturą w S4A
- A1.1 Linux, SRU: Poszukiwanie skarbów
- A2.1 Scratch: Robot-biedronka
- A3.2b S4A, Arduino: Pomiar temperatury
- B1.3 Scratch: Zakręcona mrówka
- B2.8 Lazarus: Gra logiczna NIM
- B3.2c Arduino IDE: Pomiar oświetlenia
- C2.4 Nauka języka C – wskaźniki
- C2.4 Nauka języka Python – klasy
- C2.4 HTML, JavaScript: Canvas
- C3.3a Arduino: Ultradźwiękowy pomiar odległości

Implementacja Modułu 03.2
Sterowanie klawiaturą w S4A

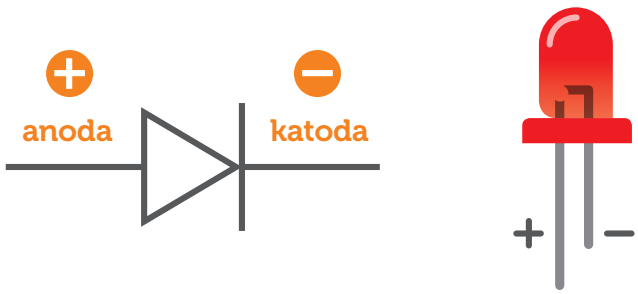
Krzysztof Bytow

Schemat połączeń

Sterowanie diodą LED PIN 13 z wykorzystaniem klawiatury – układ należy połączyć przewodem USB z komputerem, wcześniej do układu należy wgrać kod do sterowania z wykorzystaniem S4A:



Uczeń/Uczennica po zestawieniu połączeń zgłasza nauczycielowi gotowość do sprawdzenia układu i wszystkich połączeń.

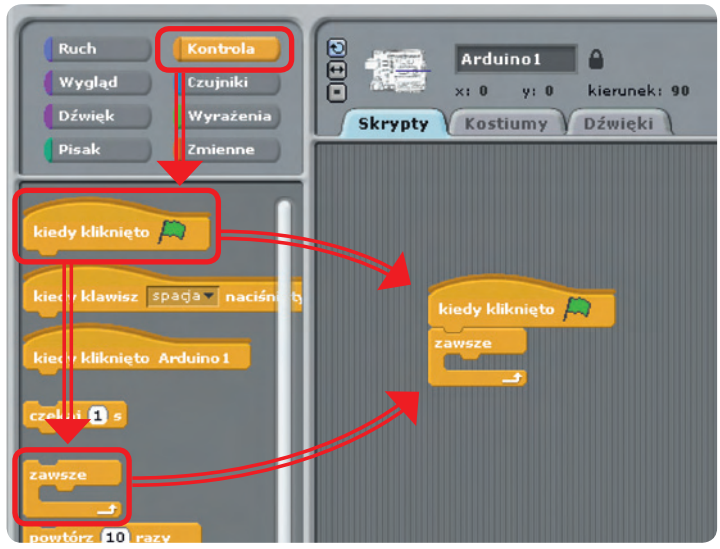


dioda led czerwona

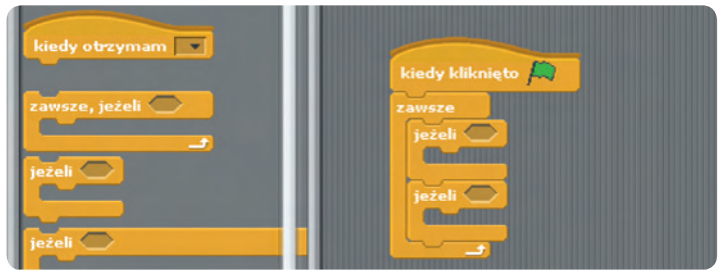
Następnie możemy przystąpić do budowy programu do sterowania diodą LED z wykorzystaniem klawiatury.

Wybieramy odpowiednie elementy skryptów:

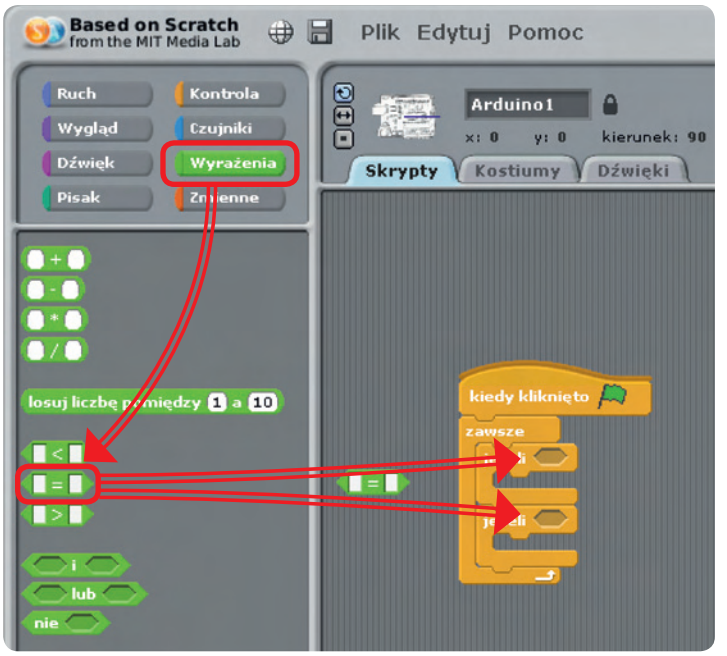
1 **Kontrola** » wybieramy *„kiedy kliknięto”* i *„zawsze”*



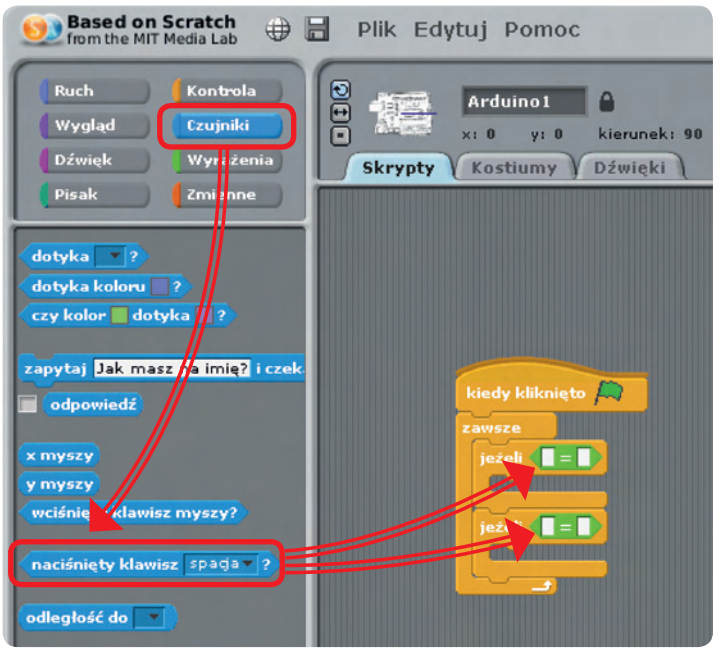
2 **Kontrola** » wybieramy dwa razy *„jeżeli”*



3 **Wyrażenia »** wybieramy dwa razy *„znak równości”*, które umieszczamy w pętli *„zawsze”*



4 **Czujniki »** wybieramy dwa razy *„naciśnięty klawisz”* i umieszczamy zgodnie ze zdjęciem, następnie wybieramy klawisze, które będą uruchamiały poszczególne kody (strzałka w lewo, strzałka w prawo).

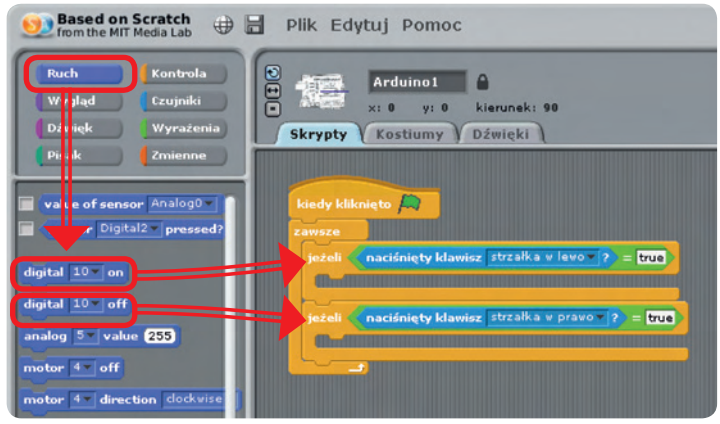


Dodatkowo w polu równości dopisujemy wartość true (prawda)

5

Ruch » wybieramy *„digital on”*, *„digital off”*, które umieszczamy w „jeżeli”

Parametr przy digital mówi nam o wejściu/wyjściu, pod które jest podłączona dioda LED do Arduino. Należy wybrać odpowiednią wartość (w naszym przypadku sterujemy diodą wbudowaną w układ Arduino PIN 13).



6

Widok ostatecznego kodu

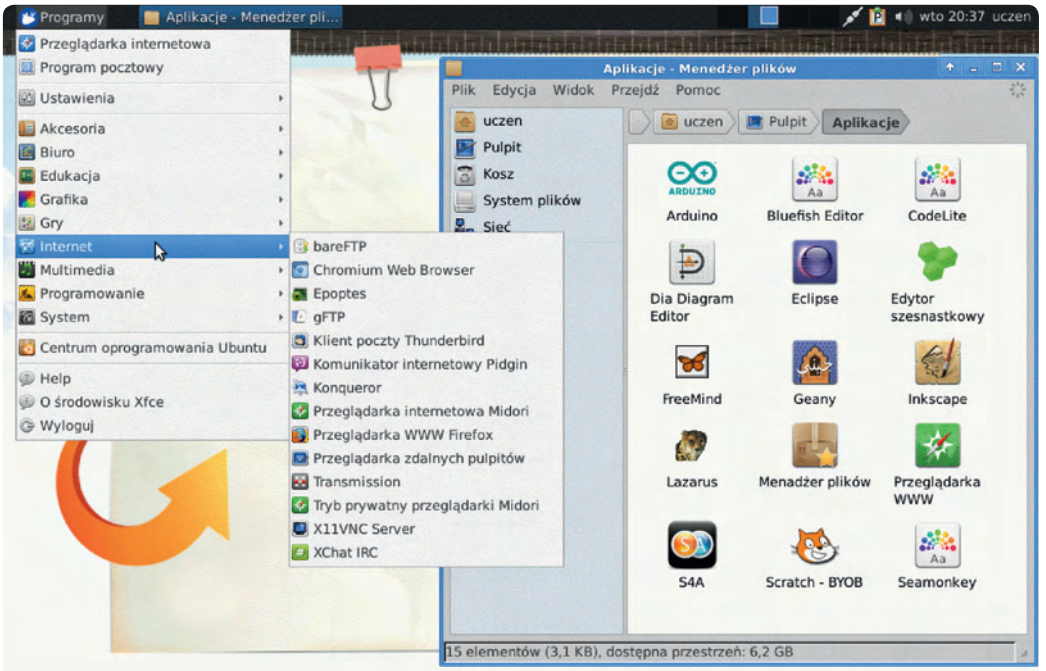
Pozostaje już tylko uruchomienie programu zieloną flagą (prawy górny róg programu). Diodą sterujemy wcześniej zdefiniowanymi klawiszami (strzałka w lewo i w prawo)

Implementacja Modułu A1.1
Poszukiwanie skarbów

Adam Jurkiewicz

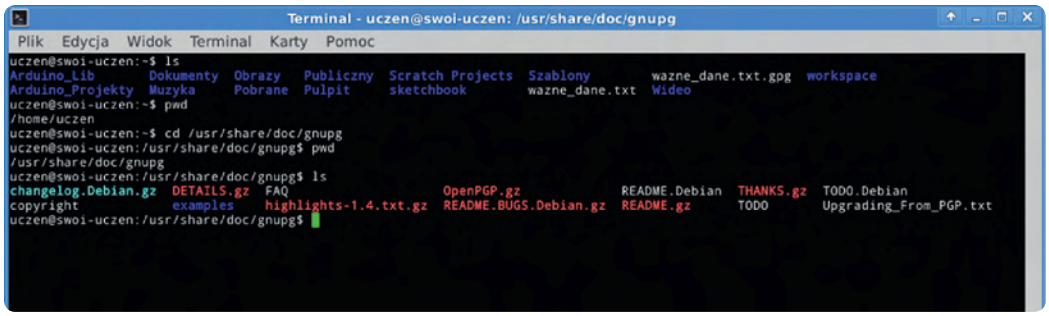
1

- Popatrz na książki na półkach, pomyśl, że katalogi i pliki na dysku są ułożone w podobny sposób.
- Uruchom Szkolny Remiks Ubuntu.
- Zobacz, jak działa menu z aplikacjami i w jaki sposób można łatwo wyszukiwać programy.



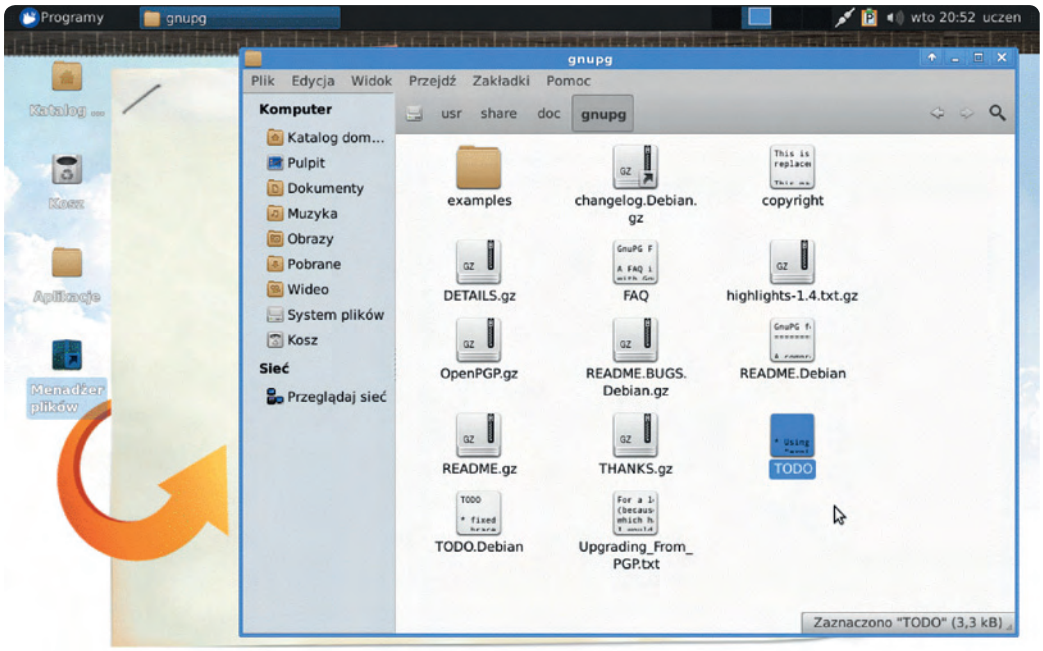
2

Uruchom terminal (konsola) i wprowadź polecenia:
- *pwd* – pokazuje, w jakim jesteś katalogu
- *ls* – pokazuje listę plików
- *cd /usr/share/doc/gnupg* – przechodzi do innego katalogu
- *ls* – pokazuje zawartość tego katalogu



3

Następnie to samo wykonaj za pomocą menadżera plików Nautilus (ikonę znajdziesz na pulpicie).

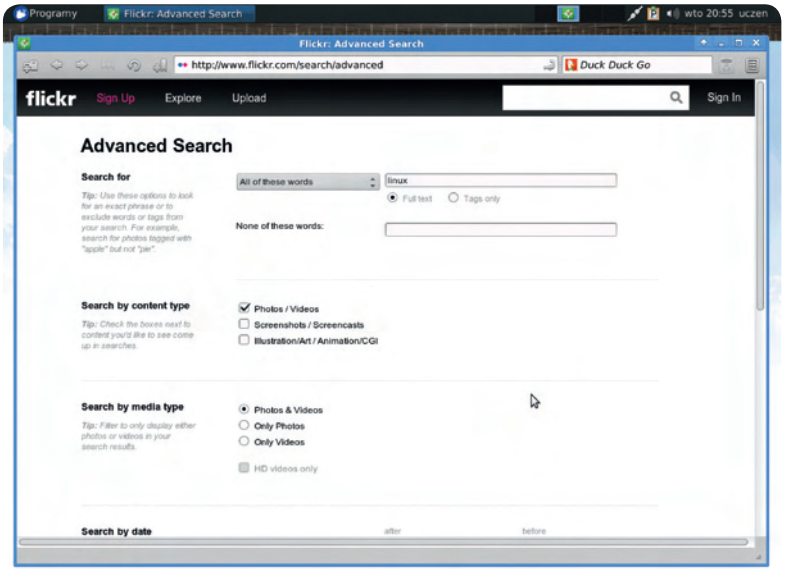


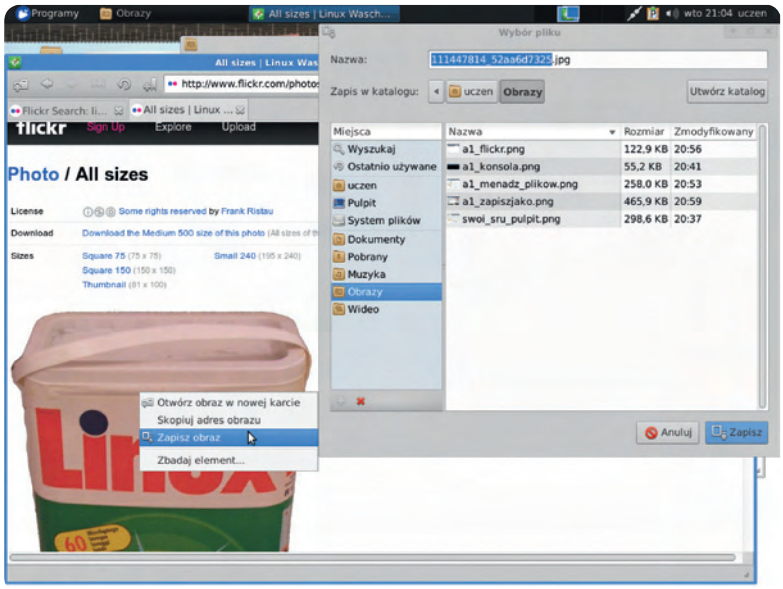
4

W ten sposób zobaczysz, że to samo można przedstawić w sposób zaawansowany i skomplikowany, jak również w łatwy i okienkowy. A fizycznie tak czy inaczej, na końcu są pliki z danymi (tekstem, obrazem, itp.)

- ćwiczenie 1 – znajdź ciekawą czcionkę w katalogu /usr/share/fonts/truetype, a następnie skopiuj ją do /home/uczen;
- ćwiczenie 2 - w /home/uczen/Obrazy utwórz materiały_graficzne;

W serwisie http://flickr.com/search/advanced/? wyszukaj zdjęcie o licencji CC dla słowa kluczowego linux lub pingwin lub komputer; pamiętaj o zaznaczeniu odpowiednich opcji przy wyszukiwaniu.
Zapisz znaleziony plik do katalogu /home/uczen/Obrazy/materiały_graficzne, aby potem móc go wykorzystać.





Zadanie domowe:

Spróbuj czcionkę z katalogu /home/uczen skopiować na inny *pendrive*, a potem w domu na inny komputer, następnie spróbuj zainstalować ją w systemie i sprawdzić, czy widzisz ją w programie edytora tekstów.

Dodatkowe informacje dla dociekliwych:

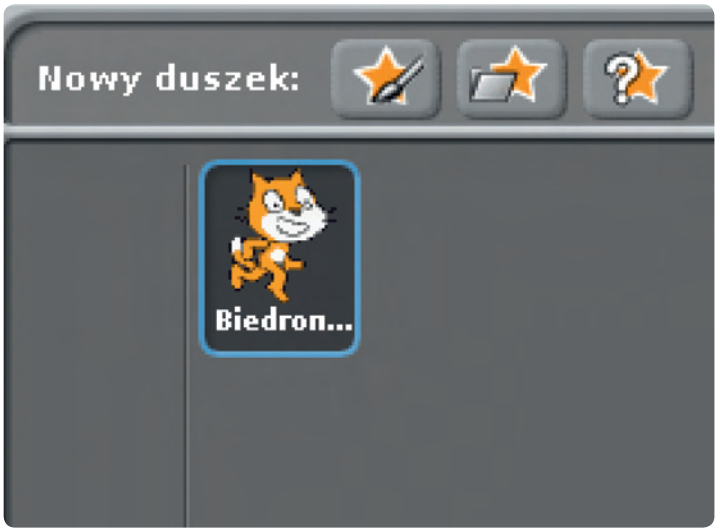
- http://www.universalsubtitles.org/pl/videos/Bg0y3OFhvcCi/info/what-is-linux-a-n00b-to--n00b-explanation/
- http://pl.wikipedia.org/wiki/Katalog_domowy
- http://pl.wikipedia.org/wiki/Katalog_%28system_plik%C3%B3w%29
- http://www.dobreprogramy.pl/Struktura-drzewa-katalogow-systemu-Linux,Artykul,11405.html
- http://czytelnia.ubuntu.pl/index.php/2007/07/25/katalogi-w-linuksie/
- http://creativecommons.pl/poznaj-licencje-creative-commons/

Implementacja Modułu A2.1
Robot-biedronka

Jarosław Żok

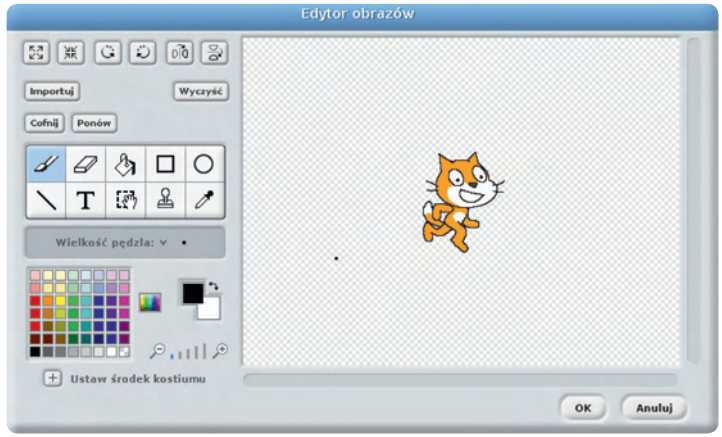
Gra została zaimplementowana z wykorzystaniem programu Scratch.
W tym ćwiczeniu stworzysz robota – biedronkę, który będzie podążał za narysowaną przez Ciebie linią.

1 Pierwszym krokiem będzie narysowanie biedronki.
Kliknij duszka kota.

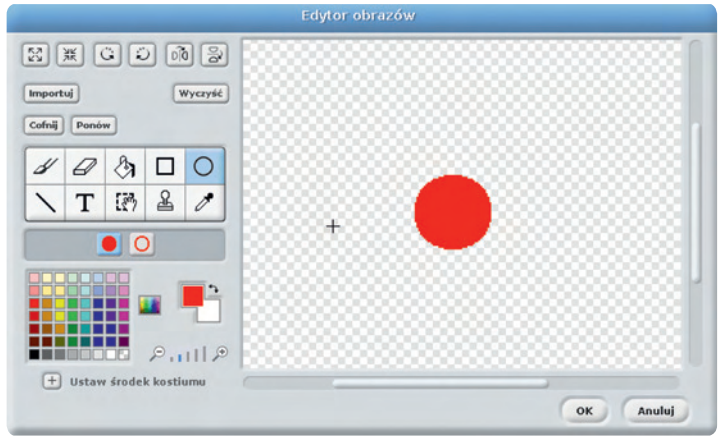


2 W środkowym oknie, w zakładce *„Kostiumy”* kliknij guzik *„Edytuj”*:

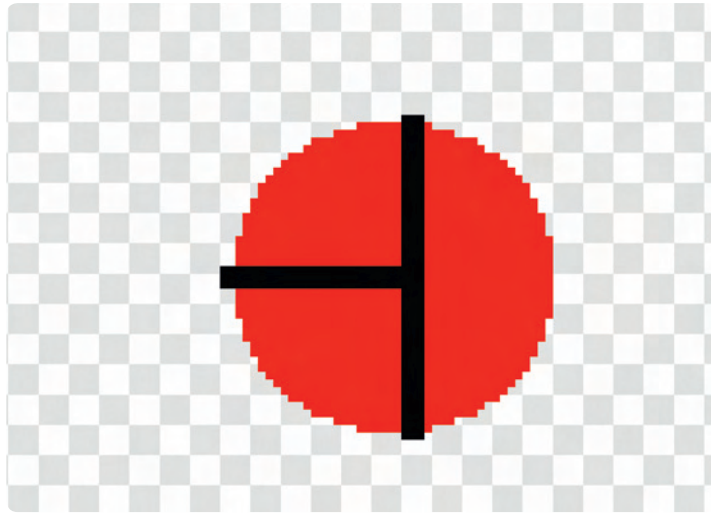
3 W edytorze kliknij przycisk *„Wyczyść”*, aby usunąć rysunek kota.



4 W edytorze obrazów, wybierz z narzędzi do rysowania – koło, a następnie w palecie – kolor czerwony w odcieniu drugim od góry. I narysuj niewielkie koło wypełnione wybranym kolorem.



5 Zmień kolor na czarny oraz wybierz narzędzie do rysowania linii. Narysuj dwie linie prostopadłe do siebie, tak żeby pozioma linia zaczynała się w lewej krawędzi koła i kończyła na linii pionowej, ustawionej w prawo zaraz za środkiem koła.



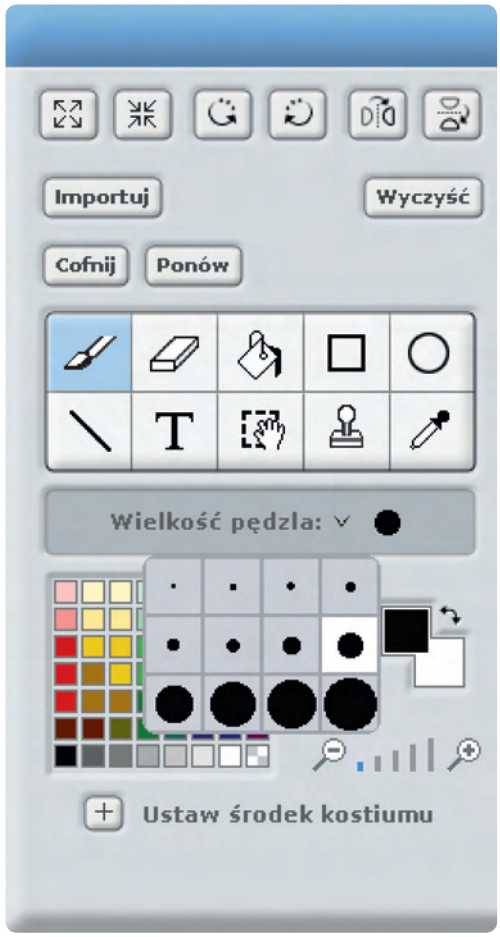
6 Wybierz teraz narzędzie do wypełniania kolorem i prawą część koła wypełnij na czarno. Możesz też na skrzydłach biedronki domalować czarne kropki:

7 Dorysujemy biedronce dwa czułki, dzięki którym będzie wiedzieć, że dotyka linii. Każdy z nich będzie miał swój indywidualny kolor. Wybieramy rysowanie linii prostej z narzędzi oraz wybieramy kolor na przykład żółty (drugi od dołu) i rysujemy krótki czułek biedronce, żółty niech będzie po jej prawej stronie. Wybierzmy teraz kolor turkusowy, także drugi od dołu i symetrycznie narysujmy drugi czułek wybranym kolorem turkusowym.

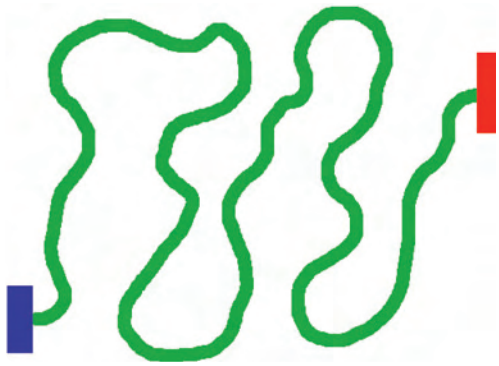
8 Wybierzmy szary kolor (czwarty od lewej na samym dole) oraz narzędzie do rysowania wypełnionych prostokątów. Między czułkami biedronki narysujmy głowę biedronki.

9 Biedronka jest już gotowa. Narysujmy tło z linią, po której będzie się poruszać. W oknie po prawej stronie na dole kliknij *„Scena”*, przejdź do zakładki *„Tła”* w środkowym oknie Scratcha i kliknij *„Edytuj”*.

10 Otworzy się edytor podobny do edycji wyglądu duszków. Wybierz kolor zielony (drugi od dołu) oraz wielkość pędzla – czwarty w drugim rzędzie.

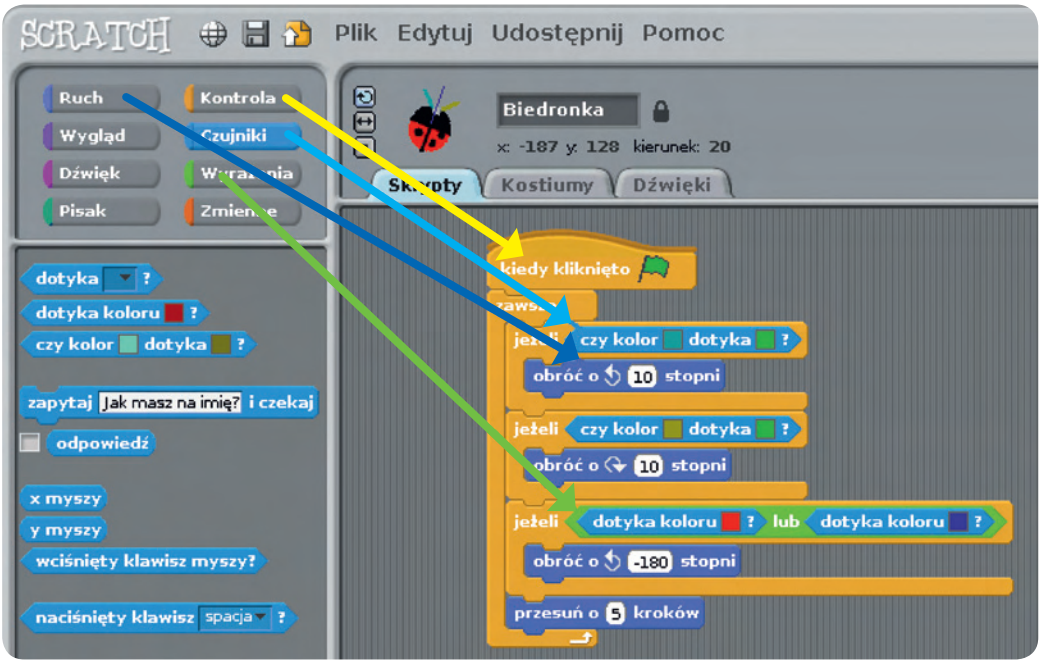


11 Narysuj, wybierając narzędzie pędzel, linię od lewej strony sceny do prawej. Linia niech łagodnie skręca, jednak nie rób zbyt mocnych skrętów, gdyż biedronka będzie na nich wypadać z toru. Na końcach linii umieść prostokąty - na jednym czerwony, na drugim niebieski.



12

Mamy gotową planszę z torem poruszania się biedronki. Kliknij teraz na biedronkę i umieść ją gdzieś na początku toru. Ważne, żeby między jej czułkami znalazł się tor, po którym będzie się poruszać.
Napiszemy teraz skrypt, który będzie animował ruch biedronki. Zauważ, jak kolory bloczków odpowiadają kolorom przycisków po lewej stronie okna Scratcha. Chcąc przenieść dany bloczek do okna skryptu, klikasz po prostu na przycisk w odpowiadającym mu kolorze i stamtąd wybierasz bloczek, który Cię interesuje.



13

Zaprogramujmy ruch biedronki. Ma się on rozpocząć po kliknięciu w zieloną flagę:

14 Następnie ruch biedronki będzie się odbywał w pętli obracającej się *"zawsze"*:

15 Umieśćmy warunek sprawdzający, czy turkusowy czułek biedronki dotyka koloru zielonego, który wybraliśmy na kolor linii. Jeżeli tak, obracamy biedronkę w lewo o 10 stopni. Kolory wybieramy, klikając kolorowy kwadrat bloczka *„czy kolor [] dotyka []?"* oraz wybierając za pomocą próbnika kolory, które nas interesują. W tym przypadku pierwszy kwadrat określa kolor czułka, drugi kwadrat określa kolor linii.

16 Podobnie postępujemy z czułkiem żółtym, zmieniamy tylko kierunek obrotów w prawo:

17 Musimy teraz tylko ruszyć naszą biedronkę z miejsca za pomocą bloczka *„przesuń o () kroków”*

18

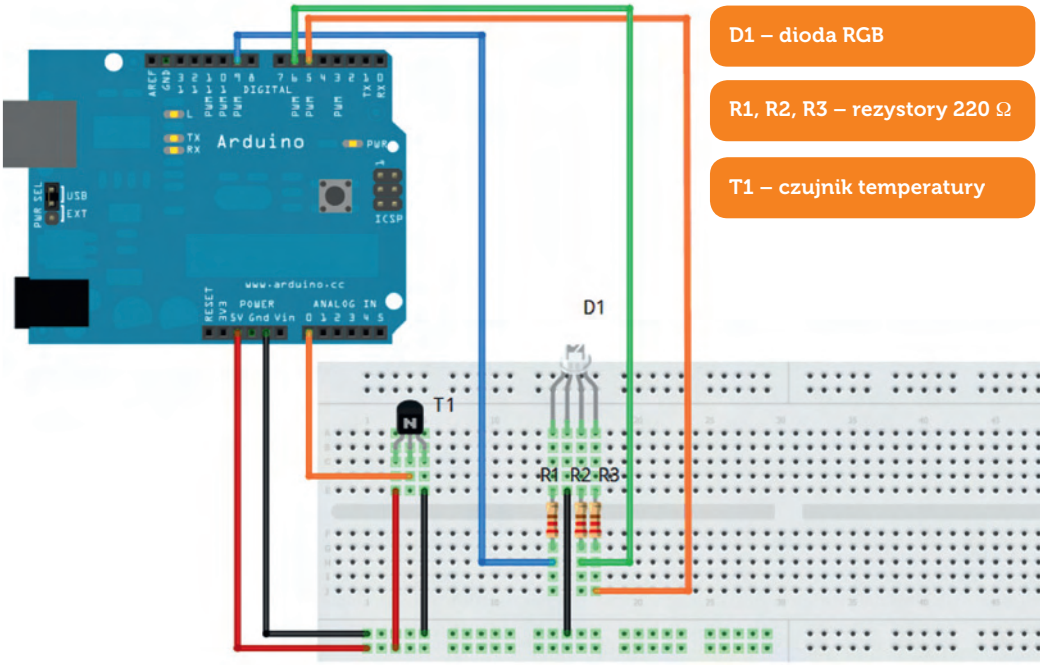
Umieśćmy teraz biedronkę na początku naszej linii i uruchommy program za pomocą zielonej flagi. Biedronka będzie podążać wzdłuż zielonej linii, dotykając raz lewym, raz prawym czułkiem linii i skręcając w kierunku, w którym znajduje się czułek. Dodajmy jeszcze warunek sprawdzający, czy biedronka dotyka czerwonego lub niebieskiego prostokąta. Po dotknięciu jednego lub drugiego zmienia kierunek o 180 stopni.

Implementacja Modułu A3.2b
Pomiar temperatury

Krzysztof Bytow

Schemat połączeń

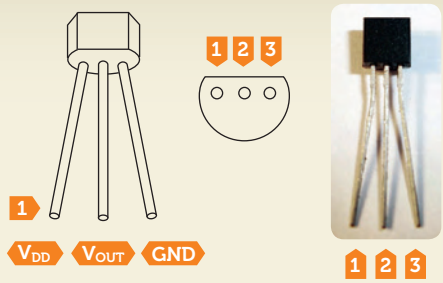
Wizualizacja pomiaru temperatury z wykorzystaniem diody RGB:



Uczeń/Uczennica po zestawieniu połączeń zgłasza nauczycielowi gotowość do sprawdzenia układu i wszystkich połączeń.

Czujnik temperatury MCP9700 – opis wyprowadzeń

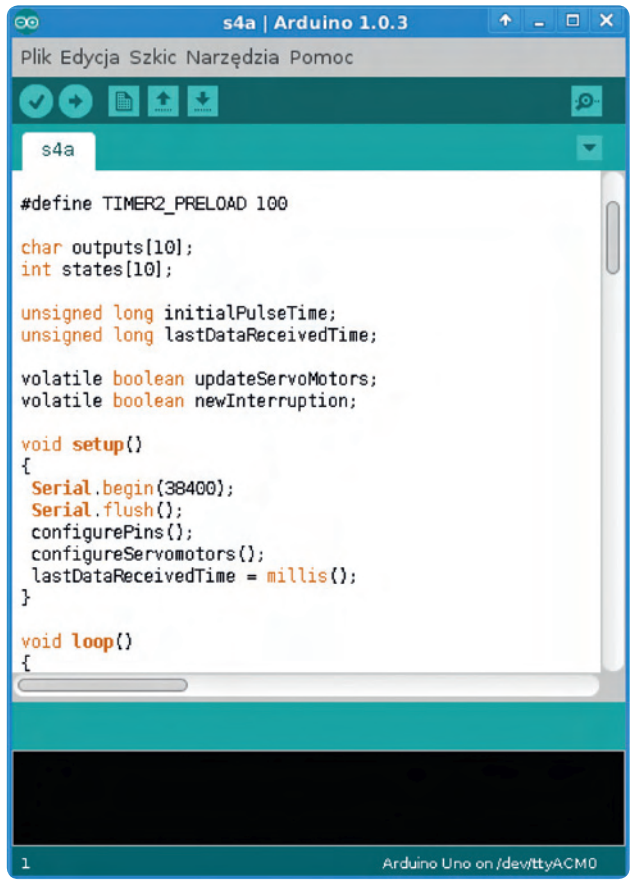
1 – napięcie zasilania (+5V);
2 – wyjście podłączone do pinu Analog 0 na Arduino;
3 – masa (GND);



opis wyprowadzeń diody RGB ze wspólną katodą

oznaczenie kodem barwnym rezystora 220 Ω

Na sterowanie układem przy użyciu S4A pozwala kod, który należy wgrać przy użyciu ArduinoIDE (otwieramy plik o nazwie S4AFirmware14.pde i wgrywamy go do układu):



Następnie możemy przystąpić do budowy programu do sterowania barwą diody RGB w zależności od temperatury.

Wybieramy odpowiednio:

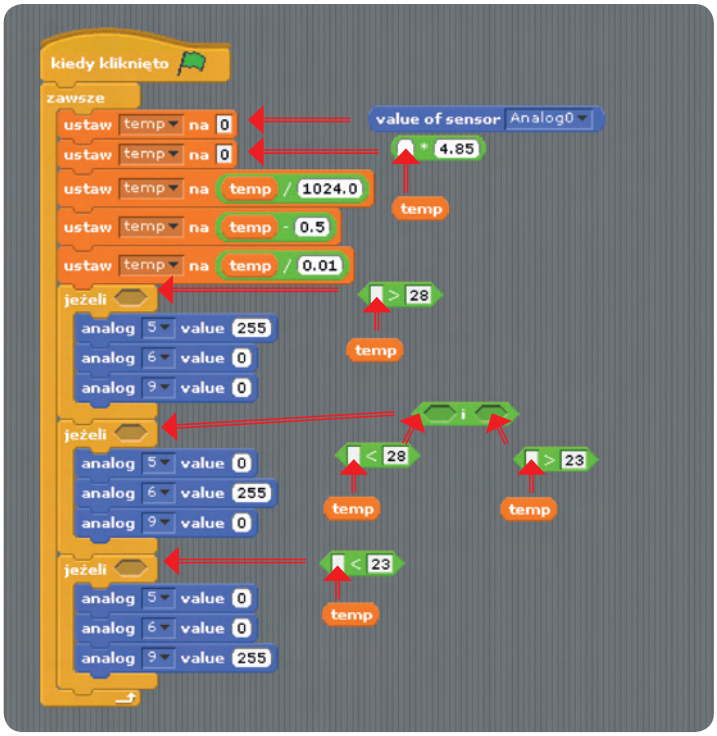
1. **Kontrola** » wybieramy *„kiedy kliknięto”* następnie *„zawsze”* i trzy razy *„jeżeli”*;

2. **Ruch** » wybieramy dziewięć razy *„analog (5) value 255”*, które wstawiamy trzy razy do *„jeżeli”*, ustawiając w każdej trójce odpowiednie wartości (5,6,9, które są wejściami/wyjściami w Arduino), wartości value ustawiamy zgodnie z rysunkiem (możliwa zmiana w celu uzyskania innej barwy i jasności diody);

3. Aby dokonać obliczeń temperatury, niezbędne jest stworzenie zmiennej *„temp”* **Zmienne** » **Utwórz zmienną** » **temp**;

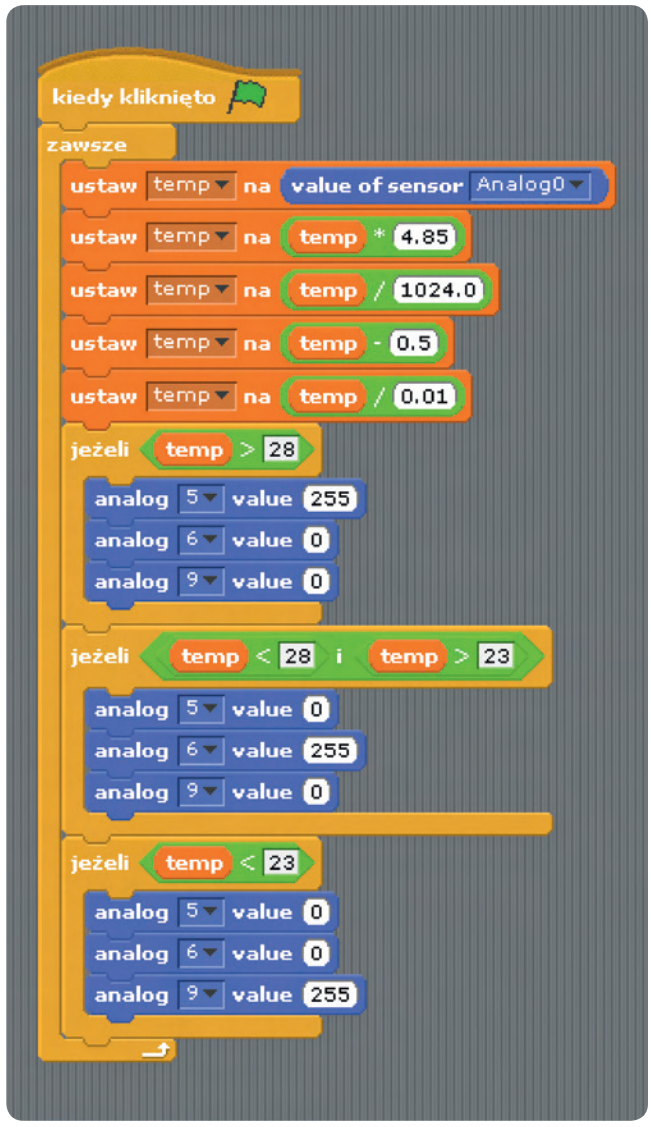
4. Niezbędne obliczenia temperatury: **Zmienne** » *„Ustaw temp na 0”* do obliczeń niezbędne jest użycie 5 klocków. Do pierwszej wartości przeciągamy: **Ruch** » *„value of sensor Analog0”* (pamiętajmy o zamianie na Analog0, gdy przeciągniemy). Kolejne obliczenia uzupełniamy z *„Wyrażeń”* odpowiednimi operacjami matematycznymi zgodnie z rysunkiem;

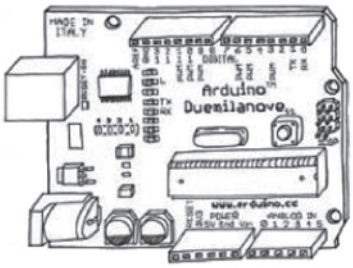
5. Aby dioda zapalała się w zadanych temperaturach, należy dodać do *„jeżeli”* warunki – wybieramy je odpowiednio z *„Wyrażeń”*.



6 Gotowy program

Odczyt temperatury obserwujemy po uruchomieniu zielonej flagi w górnej prawej części Scratcha (S4A), która jest przypisana do zmiennej temp. W zależności od temperatury obserwować będziemy mogli świecenie odpowiedniego koloru na diodzie RGB.



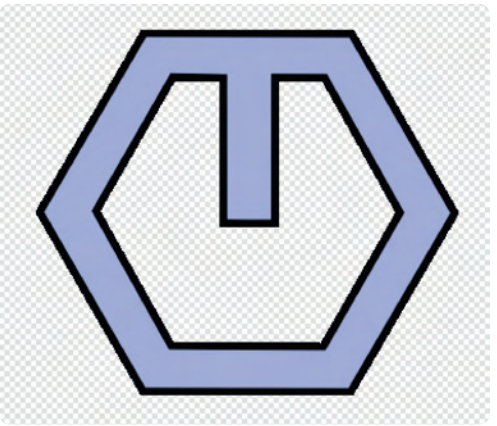


Implementacja Modułu B1.3
Zakręcona mrówka

Jarosław Żok

Będziemy implementować wizualizację pętli za pomocą mrówki zbierającej punkty, poruszającej się po zamkniętej sześciokątnej drodze. Musimy przygotować kilka nowych duszków w Scratchu. Pierwszy to nasza mrówka, drugi to mrówka strażnik, którego mrówka robotnica będzie pytać, czy zebrała odpowiednią liczbę punktów i może wrócić do mrowiska. Przygotujemy także sześć punktów w postaci kół w różnych kolorach oraz sam tor, po którym mrówka robotnica będzie się poruszać. Musimy pamiętać, żeby tor był odpowiednio szeroki, a punkty mieściły się wewnątrz rogów toru.

1 Od górnej krawędzi ścieżki do jej środka przebiega ścieżka, którą mrówka wróci do mrowiska po zebraniu odpowiedniej liczby punktów.
Ścieżka będzie wyglądała na przykład tak:



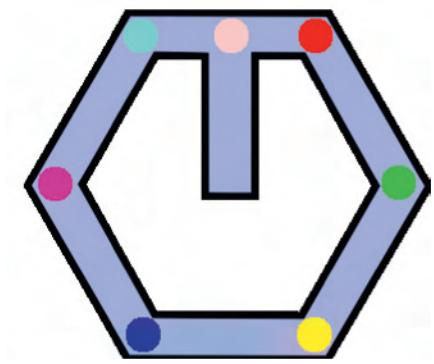
2 Przygotujmy sześć kół w różnych kolorach, które umieścimy później w rogach sześciokątnej ścieżki. Gdy mrówka robotnica znajdzie się w obszarze któregoś z nich, dodamy odpowiednią wartość do koszyka mrówki.
Dodatkowym siódmym punktem będzie wejście do mrowiska, nazwiemy je sobie domem. Mrówka trafi tam, po zebraniu odpowiedniej liczby punktów.

3

Duszka mrówki możemy wykorzystać z ćwiczenia dotyczącego warunku. Musimy zaimportować duszka oraz klikając na duszka usunąć jego skrypt. Napiszemy zupełnie nowy. Podobnie zróbmy z mrówką strażnikiem. Umieśćmy na scenie duszka reprezentującego ścieżkę, po której porusza się mrówka. Klikając na niego raz, dodajmy prosty skrypt pozycjonujący ścieżkę w określonym miejscu, w tym przypadku jest to środek sceny.

4

Dodajmy wszystkie siedem punktów do sceny. Pamiętajmy, żeby punkty ustawione były w odpowiedniej kolejności. Dla ułatwienia duszki punktów mogą zostać ponazywane numerami od 1 do 6. Jak na obrazku wyżej. Umieśćmy punkty w rogach naszej ścieżki. Punkt „dom" oznaczamy w miejscu gdzie rozgałęzia się ścieżka.



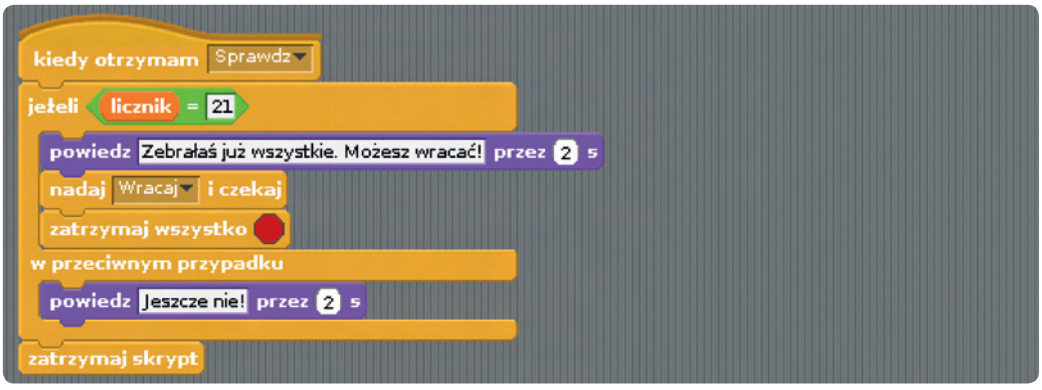
5

Mrówka będzie poruszać się wzdłuż ścieżki w prawą stronę, napotykając punkty. Musimy zaprogramować ruch mrówki, przypadek wejścia na punkt oraz powrót do domu po zebraniu odpowiedniej liczby punktów. Zacznijmy o stworzenia skryptu mrówki. Kliknijmy w nią jeden raz. **Skrypt *"powrót do domu"*:**

Skrypt jest prosty i służy skierowaniu mrówki do domu, do momentu, aż nie dotknie strażnika, który czeka przy wejściu do mrowiska.

6 Stwórzmy zmienną licznik, będzie ona przechowywać aktualnie zebraną liczbę punktów oraz zmienną krok, która określi szybkość, z jaką porusza się mrówka.
Skrypt *"wejście mrówki na pole z punktami"*:

7 Sprawdzamy, czy mrówka dotyka punktu o określonym kolorze i dodajemy do licznika punktów wartość zależną od koloru punktu. Po czym przesuwamy mrówkę o liczbę punktów wystarczającą, żeby mrówka wyszła z obszaru pętli. Zadajemy pytanie strażnikowi i wywołujemy skrypt *„sprawdź"*, którego kod należy do mrówki strażnika i jest opisany poniżej.
Skrypt *"sprawdź mrówki strażnika"*:



Skrypt sprawdza wartość zmiennej licznik i jeżeli suma w koszyku mrówki zgadza się z sumą punktów, które powinna zebrać mrówka, pozwala jej wrócić do mrowiska, wywołując skrypt *„wracaj"*. W przeciwnym wypadku skrypt jest kontynuowany.

8 Zajmiemy się teraz ruchem samej mrówki. Ustawiamy mrówkę w miejscu startowym, ustawiamy zmienne licznik i krok na odpowiednie wartości, obracamy mrówkę w kierunku pierwszego punktu i przesuwamy mrówkę, aż nie dotknie jednego z sześciu punktów rozmieszczonych na rogach ścieżki.
Skrypt *"mrówki robotnicy implementujący jej ruch po ścieżce"*:

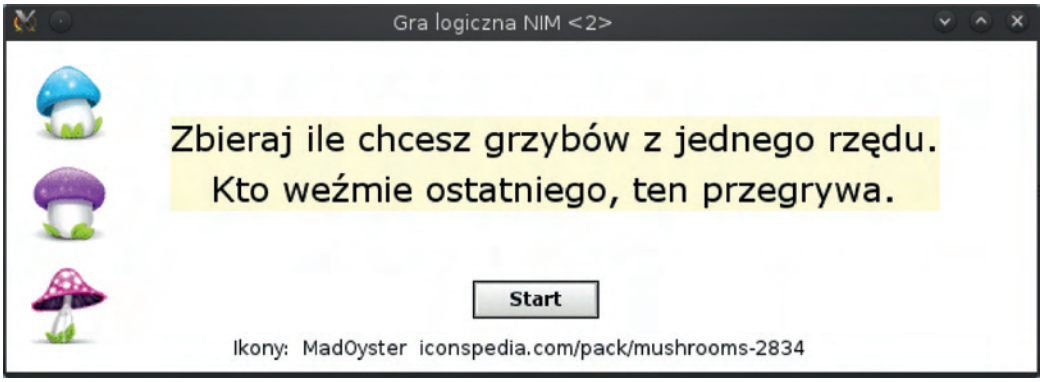
9 Pozostał nam jeszcze jeden element do wykonania, czyli uruchomienie całości. Kot będzie duszkiem, który nakaże mrówce wykonanie zadania, wywołując jej skrypt *„zbieraj"*. Kliknijmy na kota i dopiszmy krótki skrypt:

Implementacja Modułu B2.8
Gra logiczna NIM

Stanisław Ubermanowicz, Piotr Fiorek

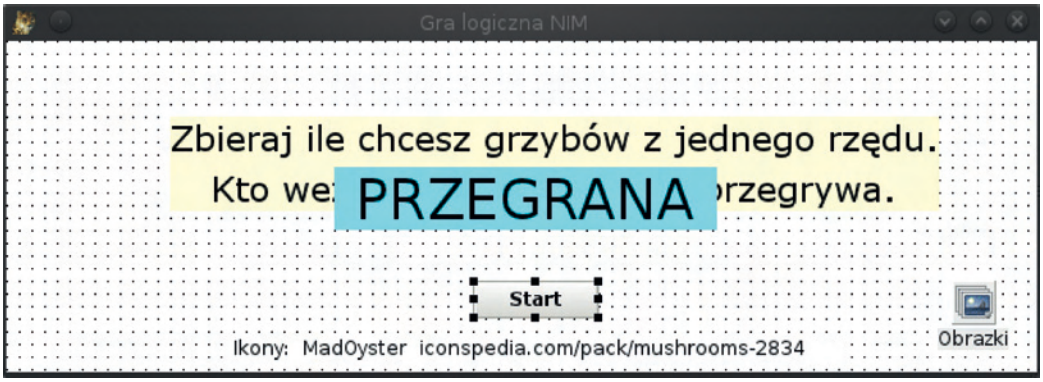
Zaprojektuj grę logiczną NIM. Program losuje w każdym rzędzie od 1 do 10 grzybów. Gracz rywalizuje z komputerem. Podczas ruchu można brać dowolną liczbę grzybów, ale tylko z jednego rzędu. Przegrywa ten, kto musi zabrać ostatniego grzyba. Strategia wygranej polega na tym, aby utrzymywać parzystość grup binarnych.

Widok po uruchomieniu



Widok w trakcie gry

Widok okna projektu



Projektowanie obiektów przykładowej implementacji

Na formularzu umieszczamy obiekty tekstowe TLabel, które będą pojawiały się w zależności od stanu gry. Na jednym z pól umieszczamy tekst instrukcji dla gracza, inne będzie służyło do informowania, kto wygrał. Dodajemy obiekt kontenera obrazków TimageList, obiekt przycisku TButton i teksty instrukcji dla gracza.

W formularzu właściwości Object Inspector ustawiamy nazwy poszczególnych elementów:

- „Link" dla obiektu typu TLabel prezentującego adres internetowy;
- „Opis" dla obiektu typu TLabel przedstawiającego opis gry;
- „Info" dla obiektu typu TLabel prezentującego informację o wygranej lub przegranej;
- „Obrazki" dla obiektu typu TImageList przechowującego obrazki kolejnych grzybków;
- „Start" dla obiektu typu TButton będącego przyciskiem rozpoczynającym grę;

UWAGA: *Do zapisywania implementacji nie wolno używać tych samych nazw dla pliku z kodem (*.pas) oraz dla pliku projektu (*.lpi').*

Kod źródłowy implementacji

{ Część elementów kodu generowana jest automatycznie }

```
unit gra;
{$mode objfpc}{$H+}
interface
uses
  Classes, SysUtils, FileUtil, LResources, Forms, Controls, Graphics, Dialogs,
ExtCtrls, StdCtrls;                    { Dopisać ExtCtrls  obsługujące dodatkowe typy obiektów }
type
  { TForm1 }
  TForm1 = class(TForm)
    Link: TLabel;                      { Zmienne reprezentujące obiekty na formularzu }
     Start: TButton;
    Opis: TLabel;
     Obrazki: TImageList;
    Info: Tlabel;
    procedure StartClick(Sender: TObject);          { Procedury obsługujące grę }
    procedure FormCreate(Sender: TObject);
    procedure Rzad1Click(Sender: TObject);
```

```
    procedure Rzad2Click(Sender: TObject);
    procedure Rzad3Click(Sender: TObject);
    procedure Rzad1Action(grzyb: Integer);
    procedure Rzad2Action(grzyb: Integer);
    procedure Rzad3Action(grzyb: Integer);
    procedure SprawdzKoniec();
    procedure PCMove();
  privat                                                    { private declarations }
    ile1, ile2, ile3: Integer;              { Wskazują, ile grzybów jest w danej chwili w rzędzie}
    tura: Integer;                          { Zmienna, która określa liczbę tur i to, czyj był ruch }
    koniec: Boolean;                        { Zmienna określająca czy nastąpił koniec gry }
    Grzyb1: array[1..10] of Timage;         { Lista obiektów pierwszego rzędu }
    Grzyb2: array[1..10] of Timage;         { Lista obiektów drugiego rzędu }
    Grzyb3: array[1..10] of Timage;         { Lista obiektów trzeciego rzędu }

  public                                                    { public declarations }
  end;
var
  Form1: TForm1;
  i: Integer;                                               { Zmienna pomocnicza, iteracyjna }
implementation
{ TForm1 }                                  { Procedura wywoływana tylko raz podczas tworzenia okna }
procedure TForm1.FormCreate(Sender: TObject);
begin
  for i:= 1 to 10 do                                        { Twórz wszystkie elementy tablicy }
  begin
    Grzyb1[i]:= TImage.Create(self);                        { twórz element }
    Grzyb1[i].Parent:= self;                                { przypisz go do okna }
    Grzyb1[i].Visible:= False;                              { ukryj go }
    Obrazki.GetBitmap(0, Grzyb1[i].Picture.Bitmap);         { przypisz mu obrazek }
    Grzyb1[i].Top:= 16;                     { określ jego położenie od górnej krawędzi ekranu }
    Grzyb1[i].Left:= (64*i)-48;             { określ jego położenie od lewej krawędzi ekranu }
    Grzyb1[i].Tag:= i;                                      { przypisz mu numer porządkowy }
    Grzyb1[i].OnClick:= @Rzad1Click;        { określ procedurę obsługującą kliknięcie na obiekt }
  end;

  for i:= 1 to 10 do                                        { drugi rząd - wszystko tak jak wyżej }
  begin
    Grzyb2[i]:= TImage.Create(self);
    Grzyb2[i].Parent:= self;
    Grzyb2[i].Visible:= False;
    Obrazki.GetBitmap(1, Grzyb2[i].Picture.Bitmap);
    Grzyb2[i].Top:= 80;
    Grzyb2[i].Left:= (64*i)-48;
    Grzyb2[i].Tag:= i;
    Grzyb2[i].OnClick:= @Rzad2Click;
  end;

  for i:= 1 to 10 do                                        { i trzeci rząd jak wyżej }
  begin
    Grzyb3[i]:= TImage.Create(self);
    Grzyb3[i].Parent:= self;
    Grzyb3[i].Visible:= False;
    Obrazki.GetBitmap(2, Grzyb3[i].Picture.Bitmap);
```

```
    Grzyb3[i].Top:= 144;
    Grzyb3[i].Left:= (64*i)-48;
    Grzyb3[i].Tag:= i;
    Grzyb3[i].OnClick:= @Rzad3Click;
  end;
  Grzyb1[1].Visible:=True;                    { Pokazanie pierwszych obiektów z każdego rzędu }
  Grzyb2[1].Visible:=True;
  Grzyb3[1].Visible:=True;
end;
                                              { Procedura obsługująca kliknięcie przycisku START }
procedure TForm1.StartClick(Sender: TObject);
begin
  Start.Visible:= False;                                          { ukryj przycisk }
  Link.Visible:= False;                                           { ukryj adres autora ikon }
  Opis.Visible:= False;                                           { ukryj opis gry }
  Info.Visible:= False;                                           { ukryj Info o wygranej lub przegranej }
  koniec:= False;                                                 { ustaw zmienną oznaczającą koniec gry na Fałsz }
  tura:= 1;                                                       { ustaw licznik tur na pierwszą turę }
  Randomize;                                                      { ustaw pozycję startową generatora liczb losowych }
  ile1:= Random(10)+1;                                            { wylosuj liczbę grzybów w pierwszym rzędzie }
  for i:= 1 to ile1 do                          { pętla przebiega przez elementy od pierwszego do wylosowanego }
    Grzyb1[i].Visible:= True;                                     { i ustawia, aby elementy były widoczne }
  ile2:= Random(10)+1;                                            { to samo dla drugiego rzędu }
  for i:= 1 to ile2 do
    Grzyb2[i].Visible:= True;
  ile3:= Random(10)+1;                                            { i dla trzeciego }
  for i:= 1 to ile3 do
    Grzyb3[i].Visible:= True;
end;
                                              { Procedura obsługująca kliknięcie w element z pierwszego rzędu }
procedure TForm1.Rzad1Click(Sender: TObject);
begin
{ Sender jest ogólną reprezentacją obiektu, który został kliknięty. Przekształcamy go na konkretny obiekt,
w tym przypadku typu TImage i pobieramy z niego wartość atrybutu Tag, która reprezentuje numer obiektu.
Numer ten od razu przekazujemy do procedury obsługującej kliknięcie w określonym rzędzie (Rzad1Action) }
  Rzad1Action((Sender as TImage).Tag);
{ Sprawdź, czy zmienna 'koniec' została ustawiona na Fałsz, jeśli tak, to obsłuż zakończenie gry }
  if koniec = False then PCMove();
end;

procedure TForm1.Rzad2Click(Sender: TObject);                     { jak wyżej tylko dla drugiego rzędu }
begin
  Rzad2Action((Sender as TImage).Tag);
  if koniec = False then PCMove();
end;
procedure TForm1.Rzad3Click(Sender: TObject);                     { jak wyżej dla trzeciego rzędu }
begin
  Rzad3Action((Sender as TImage).Tag);
  if koniec = False then PCMove();
end;
procedure TForm1.Rzad1Action(grzyb: Integer);                     { procedura zmiany stanów rzędu 1 }
begin
  tura:= tura+1;                                                  { zwiększ licznik tur }
  for i:= grzyb to ile1 do                          { wykonaj od grzyba kliknięteg aż do ostatniego: }
```

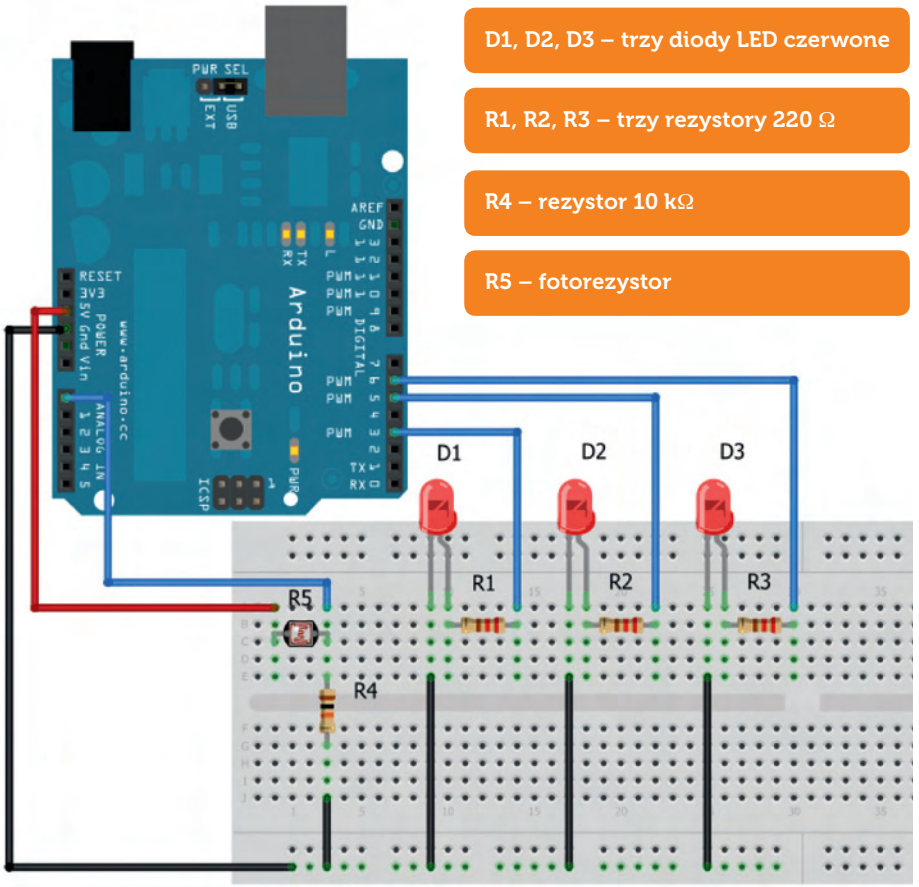
```
    Grzyb1[i].Visible:= False;                          { ukryj grzyba }
  ile1:= grzyb-1;                          { ustaw nową wartość pozostałych grzybów }
  SprawdzKoniec;                                  { sprawdź czy nastąpił koniec gry }
end;
procedure TForm1.Rzad2Action(grzyb: Integer);          { to samo dla rzędu drugiego }
begin
  tura:= tura+1;
  for i:= grzyb to ile2 do
    Grzyb2[i].Visible:= False;
  ile2:= grzyb-1;
  SprawdzKoniec;
end;
procedure TForm1.Rzad3Action(grzyb: Integer);          { i dla rzędu trzeciego }
begin
  tura:= tura+1;
  for i:= grzyb to ile3 do
    Grzyb3[i].Visible:= False;
  ile3:= grzyb-1;
  SprawdzKoniec;
end;
{ Procedura sprawdzająca, czy wszystkie grzyby zostały zabrane }
procedure TForm1.SprawdzKoniec();
begin
  if ile1+ile2+ile3 < 2 then                  { Sprawdź, czy grzybów jest mniej niż dwa }
  begin
    if (tura mod 2) = 1 then                                { Jeśli była tura gracza }
      Info.Caption:= 'Przegrana'                          { ustaw informację o przegranej }
    else if ile1+ile2+ile3 = 1 then                        { Jeśli pozostał ostatni grzyb }
      Info.Caption:= 'Wygrana'                            { ustaw informację o wygranej }
    else                                                  { a jeśli był ruch komputera }
      Info.Caption:= 'Przegrana';                         { ustaw informację o przegranej }
    koniec:= True;                                  { Ustaw zmienną oznaczającą koniec gry }
    Info.Visible:= True;                      { Pokaż napis z informacją o tym, kto wygrał}
    Start.Visible:= True;                  { Pokaż przycisk START, dla uruchomienia nowej gry }
  end;
end;
procedure TForm1.PCMove();                                { Procedura realizująca ruch komputera }
begin
  if (ile1=ile2) and (ile1<2) and (ile3>1) then                                {gdy w rzędach 1 i 2 jest}
begin                                                {0 lub 1 grzyb, to pozostaw 1 grzyba w rzędzie 3.}
   Rzad3Action(2);                                                { kliknij grzyba 2 w rzędzie 3. }
   Exit;                                                {po wykonanym ruchu wyjdź z funkcji}
  end
  else if (ile1=ile3) and (ile1<2) and (ile2>1) then                           {gry w rzędach 1 i 3 jest}
  begin                                              {0 lub 1 grzyb, to pozostaw 1 grzyba w rzędzie 2.}
    Rzad2Action(2);                                              {kliknij grzyba 2 w rzędzie 2.}
    Exit;
  end
  else if (ile2=ile3) and (ile2<2) and (ile1>1) then                           {gdy w rzędach 2 i 3 jest}
  begin                                              {0 lub 1 grzyb, to pozostaw 1 grzyba w rzędzie 1.}
    Rzad1Action(2);                                              { kliknij grzyba 2 w rzędzie 1.}
    Exit();
  end;
  If (ile1+ile2)=1 then                                {gdy w rzędach 1 i 2 pozostał łącznie jeden grzyb}
```

```
  begin
    Rzad3Action(1);                                   {usuń wszystkie z rzędu 3}
    Exit();
  end
  else if (ile1+ile3)=1 then                  {gdy w rzędach 1 i 3 pozostał łącznie jeden grzyb}
  begin
    Rzad2Action(1);                                   {usuń wszystkie z rzędu 2}
    Exit();
  end
  else if (ile2+ile3)=1 then                  {gdy w rzędach 2 i 3 pozostał łącznie jeden grzyb}
  begin
    Rzad1Action(1);                                   {usuń wszystkie z rzędu 1}
    Exit();
  end;
{Sprawdzenie parzystości binarnej w środkowej fazie gry. Szukanie układu dającego parzystość grup. Jeśli
dla danej wartości 'i' poniższe wyrażenia logiczne z operatorem XOR się zerują, to jest to poszukiwany układ
parzystości, dający szansę na wygraną.}
  for i:= 1 to 10 do
  begin
    if (ile1>=i) and (((ile1-i) xor ile2 xor ile3) = 0) then
    begin
      Rzad1Action(ile1 - i + 1)                    {doprowadź układ do parzystości}
      Exit();                                      {po wykonanym ruchu wyjdź z funkcji}
    end
    else if (ile2>=i) and ((ile1 xor (ile2-i) xor ile3) = 0) then
    begin
      Rzad2Action(ile2 - i + 1);                   {doprowadź układ do parzystości}
      Exit();                                      {po wykonanym ruchu wyjdź z funkcji}
    end
    else if (ile3>=i) and ((ile1 xor ile2 xor (ile3-i)) = 0) then
    begin
      Rzad3Action(ile3 - i + 1);                   {doprowadź układ do parzystości}
      Exit();
    end;
  end;
{Losowy wybór posunięcia, jeśli strategię wykorzystał gracz i nie ma ruchu wygrywającego}
  if (ile1>=ile2) and (ile1>=ile3) and ((Random(10)/10)>0.5) then
    Rzad1Action(Random(ile1) + 1)
  else if (ile2>=ile1) and (ile2>=ile3) then
    Rzad2Action(Random(ile2) + 1)
  else
    Rzad3Action(ile3);
end;

initialization
  {$I gra.lrs}

end.
```

Implementacja modułu B3.2c
Pomiar oświetlenia

Krzysztof Bytow

Schematy połączeń:



D1, D2, D3 – trzy diody LED czerwone

R1, R2, R3 – trzy rezystory 220 Ω

R4 – rezystor 10 kΩ

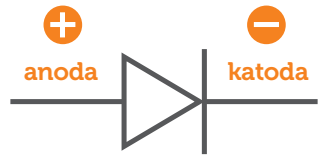
R5 – fotorezystor

Uczeń/Uczennica po zestawieniu połączeń zgłasza nauczycielowi gotowość do sprawdzenia układu i wszystkich połączeń.

fotorezystor

oznaczenie kodem barwnym rezystora 220 Ω

oznaczenie kodem barwnym rezystora 10 Ω



dioda LED czerwona

Kod implementacji

W zależności od oświetlenia fotorezystora, świeci się jedna, dwie lub trzy diody.

```
int diody[] = {-1,3,5,6};                    // tablica zawierająca numery wyjść cyfrowych
int size = 4;                                // wielkość tablicy
int j = 0;                                   // tworzymy zmienną „j" i przypisujemy jej wartość 0

void setup()                                 // część przygotowawcza

{
    for (int i=1; i< size; i++)              // pętla for
 {
    pinMode(diody[i], OUTPUT);               //ustawienie pinów jako wyjścia
 }
}

void loop()                                  // główna pętla programu

{
j = analogRead(0);          //przypisanie do zmiennej „j" wartości odczytanej z wejścia analogowego 0
j = j/128;                                   //dzielimy otrzymaną wartość przez 128
for (int i=1;i<size;i++)
{
    if (i <= j )                 //wejście w przypadku poprawności wyrażenia „j" większe równe „i"
        {
      digitalWrite(diody[i], HIGH);          // ustaw stan wysoki na diodach
    }
else                                         //wykonanie kodu w przypadku niespełnienia warunku
   {
     digitalWrite(diody[i], LOW);            //ustaw stan niski na diodach
    }
  }
  delay(200);                                //czekaj 200 milisekund
}
```

Implementacja modułu C2.4
Nauka Języka C – wskaźniki

Piotr Fiorek

Wskaźniki są specjalnym typem zmiennych. Zamiast przechowywać wartości, przechowują adresy pamięci. W ten sposób można ustawić wskaźnik, aby wskazywał na adres w pamięci, pod którym znajduje się zmienna i w ten sposób, niebezpośrednio operować na tej zmiennej.
Wskaźnik tak samo jak normalna zmienna musi mieć zdefiniowany typ i musi on być taki sam jak typ zmiennej, na którą będzie on wskazywał. Wskaźniki deklaruje się tak samo jak zmienne, tylko nazwę poprzedzamy gwiazdką:

```
int *wskaznik;
```

Następnie do wskaźnika przypisujemy wartości tak samo jak do normalnej zmiennej, tylko że wartość przypisywana musi być poprawnym adresem w pamięci w dodatku takim, do którego nasz program ma dostęp. Najlepiej do wskaźników przypisywać po prostu adresy zmiennych. Służy do tego operator „&”. Jeśli poprzedzimy nim nazwę zmiennej, to zamiast jej wartości otrzymamy adres w pamięci, pod którym nasza zmienna się znajduje. Przypisanie adresu zmiennej do wskaźnika robimy tak:

```
int zmienna;
int *wskaznik;
wskaznik = &zmienna;
```

Aby uzyskać wartość zmiennej, na którą wskaźnik wskazuje, używamy operatora „*”.
Prosty program pokazujący to wszystko może wyglądać tak:

```
#include <stdio.h>

int main(void)
{
    int zmienna=42, zmienna2=24;
    int *wskaznik;

    wskaznik = &zmienna;
```

```
    printf("Zmienna ma wartosc: %d\n", zmienna);
    printf("Zmienna znajduje się pod adresem: %p\n\n", &zmienna);

    printf("Wskaznik wskazuje na adres: %p\n", wskaznik);
    printf("Zmienna pod tym adresem ma wartosc: %d\n", *wskaznik);
    printf("Wskaznik znajduje się pod adresem: %p\n\n\n", &wskaznik);

    *wskaznik = 5;

    printf("Zmienna ma wartosc: %d\n", zmienna);
    printf("Zmienna znajduje się pod adresem: %p\n\n", &zmienna);

    printf("Wskaznik wskazuje na adres: %p\n", wskaznik);
    printf("Zmienna pod tym adresem ma wartosc: %d\n", *wskaznik);
    printf("Wskaznik znajduje się pod adresem: %p\n\n\n", &wskaznik);

    wskaznik = &zmienna2;

    printf("Zmienna ma wartosc: %d\n", zmienna2);
    printf("Zmienna znajduje się pod adresem: %p\n\n", &zmienna2);

    printf("Wskaznik wskazuje na adres: %p\n", wskaznik);
    printf("Zmienna pod tym adresem ma wartosc: %d\n", *wskaznik);
    printf("Wskaznik znajduje się pod adresem: %p\n\n\n", &wskaznik);

    return 0;
}
```

Wspomniane wcześniej zapisywanie wyników operacji funkcji do zmiennej wygląda tak:

```
#include <stdio.h>

void dodawanie(int x, int y, int *suma);

int main(void)
{
    int zmienna1, zmienna2, wynik;

    printf("Podaj pierwsza zmienna:");
    scanf("%d", &zmienna1);

    printf("Podaj druga zmienna:");
    scanf("%d", &zmienna2);
```

```
    dodawanie(zmienna1, zmienna2, &wynik);

    printf("Wynik wynosi: %d\n", wynik);

    return 0;
}

void dodawanie(int x, int y, int *suma)
{
    *suma = x + y;
}
```

Funkcja tak jak wcześniej przyjmuje dwie zmienne, które zostaną do siebie dodane, ale zamiast zwracać wynik, zapisuje go do zmiennej. Do funkcji przekazywana jest nie cała zmienna, tylko jej wartość. Dlatego właśnie przekazujemy wartość, jaką jest adres zmiennej. Adres trafia do trzeciego parametru funkcji, który jest wskaźnikiem i dalej już używamy go jak każdego innego wskaźnika.

Można również deklarować wskaźniki na struktury oraz unie. Wygląda to tak samo jak w przypadku deklarowania wskaźników na zwykłe typy danych. Jedyna różnica pojawia się w momencie, kiedy chcemy się odnieść do elementu struktury, na którą wskazuje nasz wskaźnik. Można w tym celu wyciągnąć ze wskaźnika strukturę, na który wskazuje i odnieść się do jej elementów, ale jest również prostsze i bardziej eleganckie rozwiązanie i wygląda tak:

```
#include <stdio.h>

struct cyfra
{
  int x;
};

int main(void)
{
  struct cyfra zmienna;
  struct cyfra *wskaznik;

  zmienna.x = 42;
  wskaznik = &zmienna;

  printf("Wartosc x: %d\n", zmienna.x);
  wskaznik->x = 56;
  printf("Wartosc x: %d\n", wskaznik->x);

  return 0;
}
```

Jak widać, aby odnieść się do elementu struktury, poprzez wskaźnik wskazujący na zmienną typu tej struktury, należy użyć operatora „->”. Taka strzałeczka wyciągnie zmienną bez potrzeby odnoszenia się do obiektu, na który wskaźnik wskazuje.
Warto dobrze zrozumieć wskaźniki i nauczyć się na nich sprawnie operować, ponieważ są one niezwykle często używane podczas programowania w C.

Zadania:

1. Napisz program, który używając wskaźników, obliczy średnią trzech liczb.
2. Zadanie z rozdziału o strukturach przerobić tak, aby odnoszenie się do elementów struktury odbywało się wyłącznie poprzez wskaźniki.
3. Napisz program obliczający miejsca zerowe funkcji kwadratowej (program musi być podzielony na funkcje realizujące poszczególne kawałki obliczenia).
4. Napisz program tworzący listę łączoną z elementów podawanych przez użytkownika (wykorzystaj menu z zadań o instrukcji „switch” i strukturę z zadania o strukturach).

Implementacja modułu C2.4
Nauka języka Python - klasy

Piotr Fiorek

Klasy są najtrudniejszym koncepcyjnie zagadnieniem w Pythonie. W dużym uproszczeniu – wszystkie typy danych, które do tej pory poznaliśmy to osobne typy danych, a tworząc zmienną danego typu, tworzymy obiekt typu danej klasy. Zatem jeśli tworzymy zmienną przetrzymującą cyfrę, tworzymy obiekt typu klasy „int", a tworząc zmienną przechowującą listę, tworzymy obiekt typu klasy „list". Obiekty danych klas mają tę właściwość, że poza przetrzymywaniem danych mają też własne, prywatne funkcje do operowania na zmiennych, które przechowują. Python jak wszystkie języki obiektowe pozwala na tworzenie własnych klas. Składniowo jest to bardzo proste i wygląda tak:

```
class NaszaNowaKlasa:
    pass
```

W ten sposób stworzyliśmy nową klasę o nazwie *„NaszaNowaKlasa"*. Tak jak w przypadku przykładu funkcji, klasa ta składa się jedynie ze słowa kluczowego *„pass"*, a więc nic nie robi i nie jest do niczego przydatna. Aby stworzyć klasę, która może nam się do czegoś przydać, trzeba dodać do niej funkcje, które będą wykonywały dla nas operacje na danych, które umieścimy w naszej klasie.

Na początek stworzymy klasę, która będzie służyła do przetrzymywania informacji o figurze geometrycznej, jaką jest prostokąt. Aby opisać prostokąt, potrzebujemy tylko dwóch informacji – długości obydwu boków.

Każda klasa, która ma być do czegoś przydatna, potrzebuje specjalnej funkcji nazywanej konstruktorem. Konstruktor jest to specjalna funkcja, która jest automatycznie wywoływana w momencie tworzenia nowego obiektu danej klasy. Funkcja ta przyjmuje jako parametry dane, które chcemy umieścić w obiekcie naszej klasy. W Pythonie, konstruktor ma specjalną nazwę *„__init__"*:

```
>>> class Prostokat:
...     def __init__(self, x, y):
...         self.jeden_bok = x
...         self.drugi_bok = y
```

Jak widać, nasz konstruktor tworzymy tak samo, jak każdą inną funkcję, tylko że wewnątrz klasy. Funkcja ta tak naprawdę przyjmuje trzy parametry, ale pierwszy z nich to specjalny parametr zwyczajowo nazywany *„self"*. Nie należy się nim przejmować, ale należy pamiętać, żeby zawsze umieścić go jako pierwszy parametr każdej funkcji wewnątrz tworzonej klasy. Pozostałe dwa

parametry to boki naszego prostokąta. Wewnątrz konstruktora zmienne podane jako parametry zapisujemy jako prywatne zmienne obiektu, który będzie tworzony przy użyciu konstruktora tej klasy. Tworzenie obiektów typu *„Prostokat”*:

```
>>> a = Prostokat(3, 4)
>>> b = Prostokat(7, 11)
>>> a.jeden_bok
3
>>> a.drugi_bok
4
>>> b.jeden_bok
7
>>> b.drugi_bok
11
```

„a” oraz „b” są obiektami klasy „Prostokat”. Dane podawane im podczas ich tworzenia są zapamiętywane i są prywatne dla każdego z nich. Dodajmy do naszej klasy funkcję liczącą pole prostokąta:

```
def pole(self):
    pole = self.jeden_bok * self.drugi_bok
    return pole
```

Oczywiście funkcję należy wpisać jako część klasy „Prostokat”, a więc odpowiednio wciętą (w interpreterze trzeba jeszcze raz zadeklarować całą klasę, razem z nową funkcją).

Wywołuje się ją tak, jak już wcześniej widzieliśmy np. przy obiektach typu listy, czyli nazwa obiektu, potem kropka, a potem wywołanie naszej funkcji. Czyli jeśli jeszcze raz zdefiniujemy naszą klasę wraz z nową metodą i utworzymy obiekt „a” z parametrami o wartościach „3” i „4”, to wywołanie wyglądać będzie tak:

```
>>> print(a.pole())
12
```

Funkcja nie przyjmuje żadnych parametrów (poza „self”, ale on jest ważny tylko podczas deklarowania funkcji), więc wpisujemy jej nazwę oraz puste nawiasy.

A gdybyśmy chcieli cały nasz obiekt podać jako parametr funkcji „print”? Jeśli zrobimy to z naszą klasą w takiej formie, w jakiej jest teraz, to dostaniemy mniej więcej taki wynik:

```
>>> print(a)
<__main__.Prostokat object at 0x7fc0324d6610>
```

Nie jest to tym, czego się spodziewaliśmy i czego chcieliśmy. Aby funkcja „print” wiedziała, jak poprawnie wydrukować nasz obiekt, musimy wcześniej w klasie zadeklarować specjalną funkcję o nazwie *„__str__”*. Jest ona później automatycznie wywoływana przez *„print”* za każdym razem, kiedy chcemy wyświetlić nasz obiekt.

```
def __str__(self):
    opis = 'jeden bok: ' + str(self.jeden_bok) + '\ndrugi bok: ' + str(self.drugi_
bok)
    return opis
```

Jest to po prostu funkcja o specjalnej nazwie, nieprzyjmująca żadnych parametrów, która zwraca napis. To właśnie ten napis będzie drukowany przez funkcję *„print”*. Rzeczą, na którą warto zwrócić uwagę w tym przykładzie, jest to, że długości boków podajemy jako parametry funkcji *„str”* i dopiero potem sklejamy przy użyciu plusa z innymi napisami. Robimy tak, ponieważ funkcja *„str”* zamienia liczby na napisy (a tak naprawdę tworzy obiekty klasy *„string”*, których nie przypisujemy do żadnych zmiennych, ale wykorzystujemy, aby dodać do innych napisów) i dopiero te napisy możemy dodawać do innych napisów. Gdybyśmy pominęli przekazanie długości boków do funkcji i od razu próbowali je sklejać z tekstem, to Python zwróciłby błąd mówiący, że nie wie, w jaki sposób dodać cyfrę do napisu.

W Pythonie jest wiele takich specjalnych funkcji, które mają swoje zastosowania i są wykorzystywane przez samego Pythona w różnych sytuacjach. Pełny spis tych funkcji wraz z wyjaśnieniami można znaleźć w dokumentacji pod adresem: *http://docs.python.org/3/reference/datamodel.html#special-method-names*. Gotowe klasy służące do wielu przydatnych rzeczy można znaleźć w tak zwanych modułach. Opis wszystkich standardowych modułów jest pod adresem: *http://docs.python.org/3/library/index.html*. Znajduje się tam lista modułów, a pod linkiem do każdego z nich opis klas oraz funkcji z tych klas, które dostarcza.

Aby użyć jakiegoś modułu w naszym programie, wystarczy na początku dopisać *„import <nazwa-modułu”*. Każdy moduł wystarczy zaimportować tylko raz, na początku pliku z kodem.

Jeśli byśmy np. chcieli zaimportować moduł „calendar” i przy użyciu funkcji o takiej samej nazwie wydrukować kalendarz na rok 2014, wyglądałoby to tak:

```
>>> import calendar
>>> print(calendar.calendar(2014))
                                  2014

      January                   February                  March
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
       1  2  3  4  5                      1  2                      1  2
 6  7  8  9 10 11 12       3  4  5  6  7  8  9       3  4  5  6  7  8  9
13 14 15 16 17 18 19      10 11 12 13 14 15 16      10 11 12 13 14 15 16
20 21 22 23 24 25 26      17 18 19 20 21 22 23      17 18 19 20 21 22 23
27 28 29 30 31            24 25 26 27 28            24 25 26 27 28 29 30
                                                    31

      April                     May                       June
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6                1  2  3  4                         1
 7  8  9 10 11 12 13       5  6  7  8  9 10 11       2  3  4  5  6  7  8
14 15 16 17 18 19 20      12 13 14 15 16 17 18       9 10 11 12 13 14 15
```

```
21 22 23 24 25 26 27
28 29 30

      July
Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31

     October
Mo Tu We Th Fr Sa Su
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

```
19 20 22 23 24 25 26
27 28 29 30 31

     August
Mo Tu We Th Fr Sa Su
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31

    November
Mo Tu We Th Fr Sa Su
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

```
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30

    September
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30

    December
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

Aby użyć czegoś z modułu, musimy nazwę „tego czegoś" poprzedzić nazwą modułu i kropką, dokładnie tak jak robiliśmy z funkcjami prywatnymi klasy. Robimy tak, ponieważ wykorzystaliśmy tutaj funkcję prywatną modułu.

Zadania:

1. Napisać klasę reprezentującą dane człowieka (zmienne takie jak: imię, nazwisko, płeć, wiek, wzrost), posiadającą kilka podstawowych funkcji i wykorzystać ją w programie.
2. Napisz klasę bazową reprezentującą samochód i dwie klasy dziedziczące po niej – jedną reprezentującą samochód osobowy z dodatkowymi metodami do „obsługiwania" liczby pasażerów i drugą reprezentującą samochód ciężarowy z metodami do obsługi typu i masy ładunku.

Implementacja modułu C2.4 Canvas

Daniel Mendalka

Gdy chcemy stworzyć aplikację, która prezentuje informacje, wykorzystując różne obiekty graficzne, obrazki i animacje najlepiej jest wykorzystać element <canvas>, który daje nam najwięcej możliwości w rysowaniu obiektów oraz najefektywniej wykorzystuje możliwości komputera łącznie ze wsparciem karty graficznej przy złożonych animacjach.

Zadanie: Żeby rozpocząć pracę, wystarczy stworzyć prosty dokument HTML z jednym elementem:

```
<canvas id="canvas" width="700" height="500"></canvas>
```

Kilka parametrów CSS określających wielkość i położenie naszego pola do rysowania:

```
#canvas {
  position: relative;
  display: block;
  margin: 100px auto 0;
  border: 3px solid #999;
}
```

oraz proste linijki JavaScript tworzące obiekt, na którym będziemy dokonywać operacji rysowania:

```
var canvas = document.getElementById('canvas');
var ctx = canvas.getContext('2d');
```

Uwaga: W tym przypadku nie korzystamy z pomocy biblioteki jQuery, gdyż byłaby użyta zaledwie w kilku miejscach, lecz bez problemu możemy zastąpić ją zwykłym kodem JavaScript:

- Konstrukcja, która opóźnia wywołanie kodu, dopóki nie załaduje się cały dokument HTML

  ```
  $(function() { /* kod aplikacji */ });
  ```

  zostaje zastąpiona obsługą zdarzenia `DOMContentLoaded`:

  ```
  document.addEventListener('DOMContentLoaded', function() {
  /* kod aplikacji */
      });
  ```

- Wyszukiwanie elementu po atrybucie id: `$('#canvas')` zamieniamy na:

  ```
  document.getElementById('canvas');
  ```

Rysowanie podstawowych obiektów geometrycznych

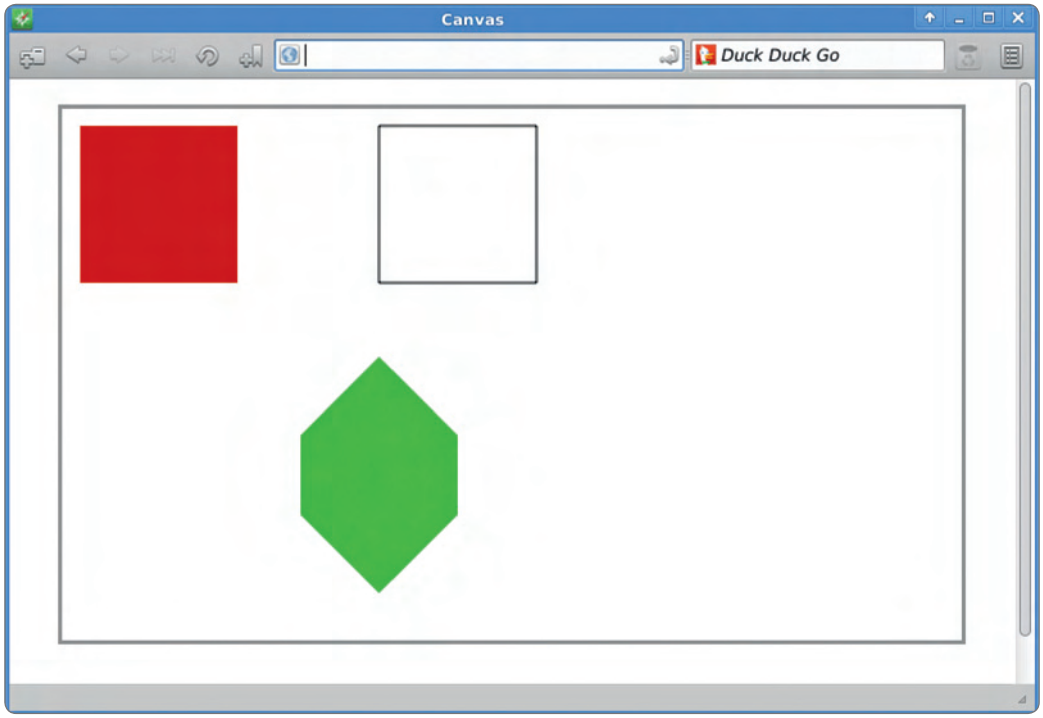
Utworzony obiekt pod zmienną ctx pozwala nam na rysowanie z wykorzystaniem metod:

`fillRect(x,y,width,height)` – rysuje prostokąt zaczynający się w punkcie x, y o szerokości i wysokości: `width, height`.
`strokeRect(x,y,width,height)` – tak samo, jak `fillRect()`, lecz w trakcie rysowania powstaną tylko krawędzie prostokąta.
`clearRect(x,y,width,height)` – pozwala na wyczyszczenie wskazanego fragmentu lub całości canvas.
`beginPath()` – informuje, że zaczynamy rysować wielokąt.
`closePath()` – informuje o zakończeniu rysowania wielokąta.
`lineTo(x,y)` – rysuje linie do punktu x, y.
`moveTo(x,y)` – oznacza punkt, w którym rozpoczynamy rysowanie.
`fill()` – po zdefiniowaniu wielokąta wypełnia jego zawartość.
`stroke()` – po zdefiniowaniu wielokąta rysuje jego krawędzie.

Oprócz powyższych metod mamy jeszcze dwa parametry:
`fillStyle` – przypisujemy do niego kolor wypełnienia rysowanego obiektu.
`strokeStyle` – przypisujemy do niego kolor dla krawędzi obiektu.

Uwaga: Jeśli chcemy, żeby rysowany obiekt miał zarówno wypełnienie, jak i krawędzie, musimy wywołać zarówno metodę `fillRect()` i `strokeRect()` lub `fill()` i `stroke()`.

Zadanie: Napisz kod rysujący krawędzie i wypełnienie prostokąta w różnych kolorach oraz co najmniej pięcioramienny wielokąt.



Obsługa zdarzeń myszki

Żeby pozwolić użytkownikowi rysować nowe figury, potrzebujemy stworzyć obsługę dwóch zdarzeń:

`mousedown` – użytkownik naciska przycisk myszki i zaczyna rysować prostokąt.
`mouseup` – użytkownik puszcza przycisk myszki i kończy rysować prostokąt.

Miejsce, w którym użytkownik kliknął myszką, możemy pobrać z `event.layerX` i `event.layerY` lub w przypadku gdy korzystamy z przeglądarki Opera: `event.offsetX` i `event.offsetY`.

Uwaga: Parametry te podają odległość względną do krawędzi najbliższego kontenera, który w CSS ma ustawione: `position:relative;` lub `position:absolute;`. Bez tej właściwości otrzymamy odległość przycisku do krawędzi okna przeglądarki i będziemy musieli dodatkowo obliczać punkt początkowy, w którym znajduje się nasz tag `<canvas>`.

Zadanie: Korzystając z poniższego szablonu napisz kod, który pozwoli użytkownikowi rysować prostokąty.

```
canvas.addEventListener('mousedown', function(event){
  /* Zachowaj wartości początkowe layerX i layerY */
});

canvas.addEventListener('mouseup', function(event){
  /* Korzystając z zapisanych wartości w mousedown
     oblicz szerokość i wysokość rysowanego prostokąta
     i narysuj go używając fillRect(); */
});
```

Zadanie dodatkowe: Dodaj kilka przycisków, które pozwolą zmienić kolor rysowanych obiektów.

Animacje

Animacja w przypadku korzystania z **canvas** to nic innego jak ciągłe czyszczenie i ponowne rysowanie obiektów, które zmieniły swoje położenie od poprzedniego rysowania. W naszej aplikacji możemy zaprezentować prostą animację, gdy użytkownik rysuje swój prostokąt, pokazując w trakcie jego kontury. Żeby móc to zrobić, potrzebujemy też tabeli, w której będziemy przechowywać informacje o wszystkich już wcześniej narysowanych obiektach, aby móc je wykorzystać przy odświeżaniu ekranu. Pamiętajmy też o zapisaniu koloru prostokąta.

```
ctx.fillRect(x,y,width,height);
objects.push({x:x, y:y, w:width, h:height, c:ctx.fillStyle});
```

Zadanie: Dodaj obsługę kolejnego zdarzenia `mousemove`, które będzie:

- czyściło całą przestrzeń do rysowania: `ctx.clearRect(0,0,700,500);`
- ponownie rysowało wszystkie wcześniej stworzone obiekty;
- pokazywało aktualne kontury rysowanego prostokąta;

Animacja z użyciem setInterval, setTimeout i requestAnimationFrame

Poprzednia animacja była uruchamiana poprzez akcje wywoływane przez użytkownika. Natomiast jeśli chcemy, żeby animacja działała nawet, gdy nic nie jest robione w aplikacji, możemy odświeżać ekran, używając jednej z metod:

`setInterval()` – wywołuje bez przerwy funkcję w odstępie podanej liczby milisekund.
`setTimeout()` – wywołuje funkcję po podanej liczbie milisekund. Może zastąpić działanie `setInterval()`, jeśli na końcu wywołanej funkcji będzie ponownie użyty `setTimeout()`.

```
var draw = function() {
/* Uruchamiany kod */
setTimeout(draw, 100);
};
setTimeout(draw, 100);
```

Zadanie: Spraw, że prostokąty po narysowaniu będą spadały na ziemię. Porównaj działanie obu opisanych metod.

Zadanie dodatkowe: Wykonaj powyższe zadanie z użyciem `requestAnimationFrame()`. `requestAnimationFrame()` jest specjalnie zmodyfikowaną wersją `setTimeout()`. Silnik przeglądarki sam oblicza, jak często ma nastąpić odświeżenie ekranu w zależności od intensywności obliczeń i obciążenia procesora.

```
var draw = function() {
  /* Uruchamiany kod */
  requestAnimationFrame(draw);
};
requestAnimationFrame(draw);
```

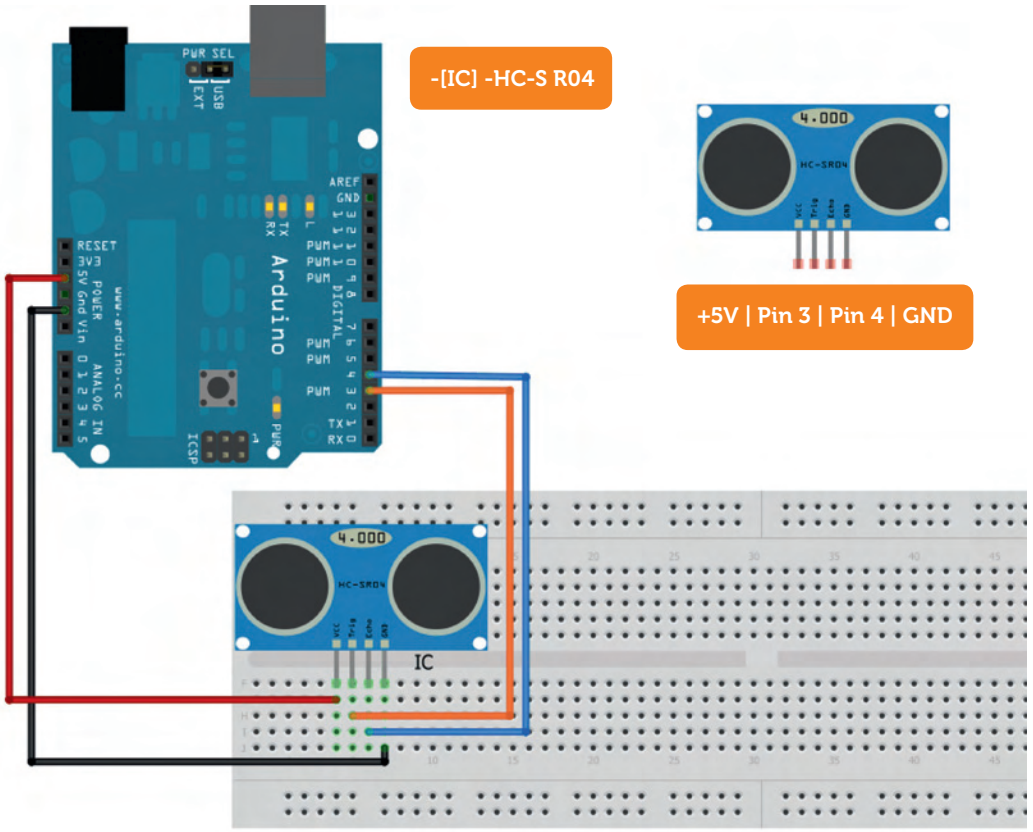
Uwaga: Funkcja ta nie jest jeszcze stabilnie zaimplementowana we wszystkich przeglądarkach, więc każda z nich udostępnia ją pod nazwą z prefixem. Żeby móc z niej swobodnie korzystać, należy dodać na początku ten kod:

```
var requestAnimationFrame = window.requestAnimationFrame ||
window.mozRequestAnimationFrame ||
window.webkitRequestAnimationFrame ||
window.msRequestAnimationFrame;
```

Implementacja modułu C3.3a
Ultradźwiękowy pomiar odległości

Krzysztof Bytow

Schemat połączeń:



Uczeń/Uczennica po zestawieniu połączeń zgłasza nauczycielowi gotowość do sprawdzenia układu i wszystkich połączeń.

Czujnik ultradźwiękowy – opis wyprowadzeń

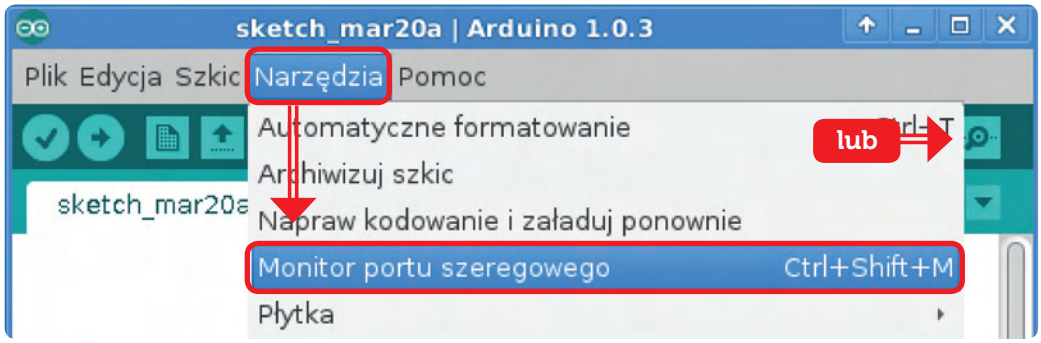
1 – Vcc +5V
2 – Trig
3 – Echo
4 – GND-masa

Aby czujnik ultradźwiękowy działał należy dograć bibliotekę Ultrasonic do pobrania np.: http://iteadstudio.com/application-note/arduino-library-for-ultrasonic-ranging-module-hc-sr04/

Kod implementacji

```
#include <Ultrasonic.h>                          // podłączamy bibliotekę
Ultrasonic miernik(3,4);                         //definiujemy porty pod które podpięty jest czujnik

void setup()                                     // początkowa konfiguracja – część przygotowująca układ
{
 Serial.begin(9600);                             //rozpoczęcie komunikacji
}

void loop()                                      // główna pętla
{
int x=miernik.Ranging(CM);                       // definiujemy zmienną x i przypisujemy jej wartość
Serial.print(x);                                 // wyświetl wartość na monitorze
Serial.println( "cm" );                          // wyświetl tekst na monitorze
delay(1000);                                     //czekaj 1000ms
}
```

Po wgraniu kodu należy w programie Arduino IDE uruchomić Serial Monitor (lub terminal), aby obserwować wyniki pomiaru.



Zawartość płyty z narzędziami i wytworami SWOI w wersji elektronicznej

Wersja elektroniczna narzędzi i wytworów SWOI umieszczona została na płycie, stanowiącej integralną część publikacji. Płyta pozwala na pracę w Szkolnym Remiksie Ubuntu – przygotowanym specjalnie do realizacji zajęć w szkołach, zawierającym pakiet oprogramowania i zasobów niezbędnych do ćwiczeń.

Na płycie znajdują się:

Implementacje
- Opisy ponad 100 implementacji do zajęć
- Kody źródłowe implementacji

Publikacje w PDF
- Tom 1: „Strategia nauczania-uczenia się infotechniki”
- Tom 2: „Program nauczania-uczenia się infotechniki”
- Prezentacje promujące Strategię

System
- Linux Ubuntu 12.04 LTS, środowisko XFCE
- aplikacja Centrum Oprogramowania
- oprogramowanie do zarządzania aktualizacjami
- usprawnienia systemu, systemem drukarek

Aplikacje biurowe
- Evince – aplikacja do podglądu plików PDF
- FreeMind – edytor map myśli
- Libre Office Writer – edytor tekstu
- Libre Office Impress – edytor prezentacji
- Libre Office Base – baza danych
- Libre Office Calc, Gnumeric – arkusze kalkulacyjne

Aplikacje narzędziowe
- Ark – menadżer archiwów

- Gedit, LeafPad, MousePad – edytory tekstowe ASCII
- Goodies – program do notatek na pulpicie
- Gnome Commander, Midnight Commander – menadżery plików
- Nautilus, Thunar – menadżery plików
- Parcellite – menadżer schowka
- Screenshot – program do zrzutów ekranu
- Xterm – emulator terminala

Programowanie

- Eclipse, CodeLite, Geany, Lazarus, SPE – środowiska programowania w Javie, C, Pythonie i Pascalu
- Scratch, S4A – środowiska programowania graficznego
- Winpdb, wxGlade – aplikacje do tworzenia interfejsów
- Bluefish – edytor stron HTML

Edukacja

- Celestia – wirtualna mapa nieba
- Gcompris – aktywności dla edukacji wczesnoszkolnej
- Genius Math Tool – zaawansowany kalkulator i kreator wykresów
- Kig – interaktywna geometria
- KmPlot – aplikacja do tworzenia wykresów funkcji
- KTurtle – nauka programowania w LOGO
- Lybniz Graph Plotter – rysowanie funkcji
- Stellarium – wirtualna mapa nieba
- Step – symulator zjawisk fizycznych
- Tux Math – matematyka dla edukacji wczesnoszkolnej
- Tux Paint – rysowanie dla edukacji wczesnoszkolnej
- Tux Typing – nauka pisania bezwzrokowego

Grafika

- Blender – generowanie modeli 3D
- Dia – edytor diagramów
- digiKam – pobieranie zdjęć z aparatów fotograficznych i zarządzanie nimi
- KolourPaint, LibreOffice Draw, GIMP – bitmapowe edytory graficzny
- gThub, Gwenview, Risretto – przeglądarki plików graficznych
- Hugin – aplikacja do tworzenia obrazów panoramicznych
- Inkscape – wektorowy program graficzny operujący na krzywych
- SweetHome 3D – program do modelowania wyposażenia domu

Gry

- Kanagram, KHangMan – edukacyjne gry typu wisielec
- Laby – gra w programowanie do nauki 8 języków programowania
- Mahjong, Sudoku – gry logiczne

Internet

- bareFTP, gFTP – graficzne programy FTP
- Chromium, Mozilla Firefox, Konqueror, Midori – przeglądarki WWW
- XChat, Pidgin, Transmission – aplikacje wymiany danych i wiadomości
- Mozilla Thunderbird – klient poczty elektronicznej

Multimedia

- Brasero, XFBurn – programy do wypalania płyt CD/DVD
- VLC, gmusicbrowser, Totem, Parole – odtwarzanie multimediów
- OpenShot – nieliniowy edytor wideo
- RecordMyDesktop – program do nagrywania obrazu pulpitu
- XCFA – program do konwersji płyt AUDIO CD na różne formaty
- Rejestrator dźwięku – program do nagrywania dźwięku

Skorowidz

W

Z

SWOI

Biuro projektu:

Fundacja Wolnego i Otwartego Oprogramowania
ul. Staszica 25/8
60-524 Poznań
tel. +48 61 623 25 36, 61 624 34 74
fax: +48 61 623 25 04
www.fwioo.pl

www.e-Swoi.pl