KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

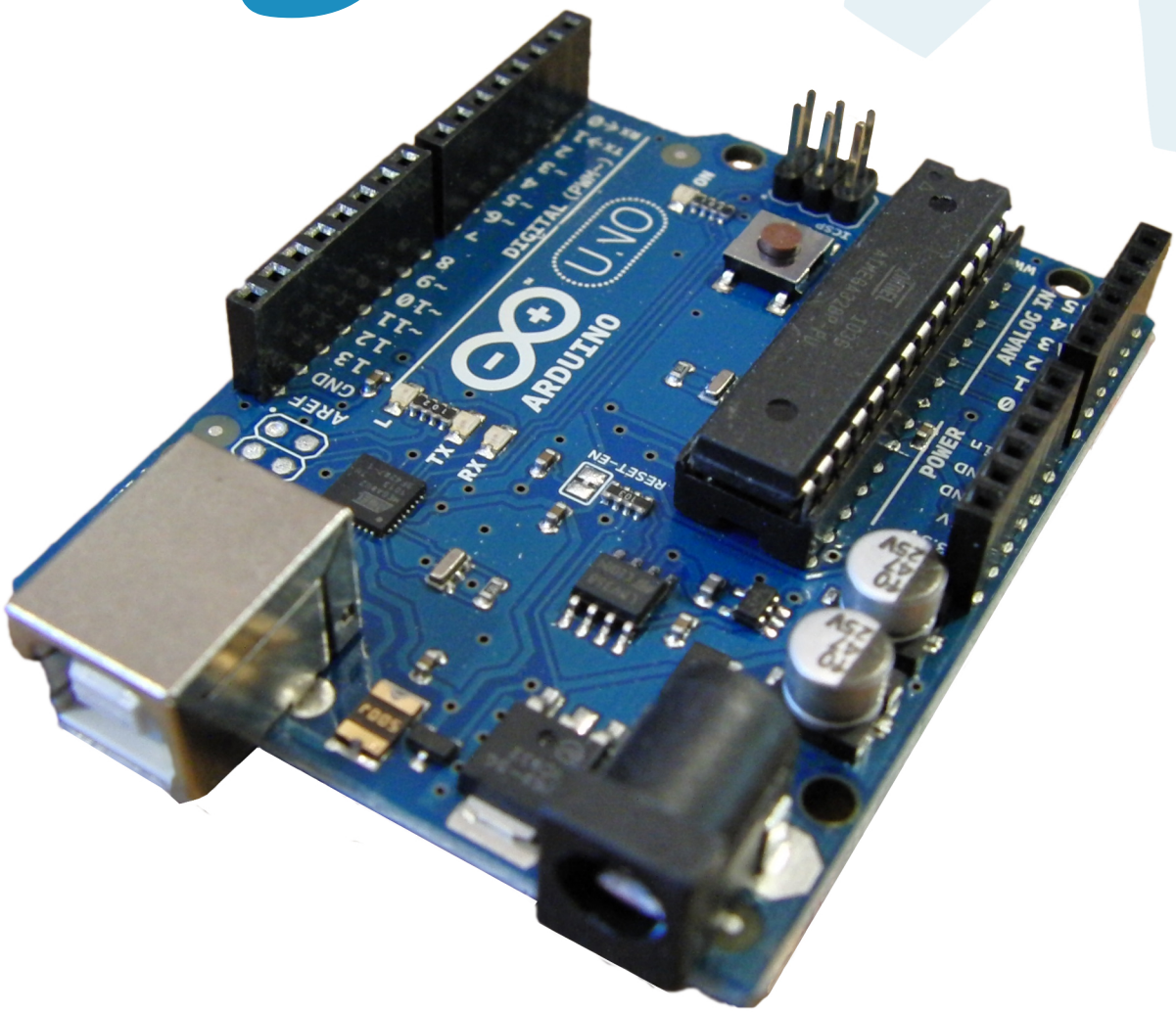
MAŁY INŻYNIER .pl

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

HIGH TECHNIKA

Program zajęć z techniki

Elektronika cyfrowa



Autor programu zajęć:
mgr inż. Marcin Jukiewicz

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Spis treści

Zajęcia 1: Pierwszy program

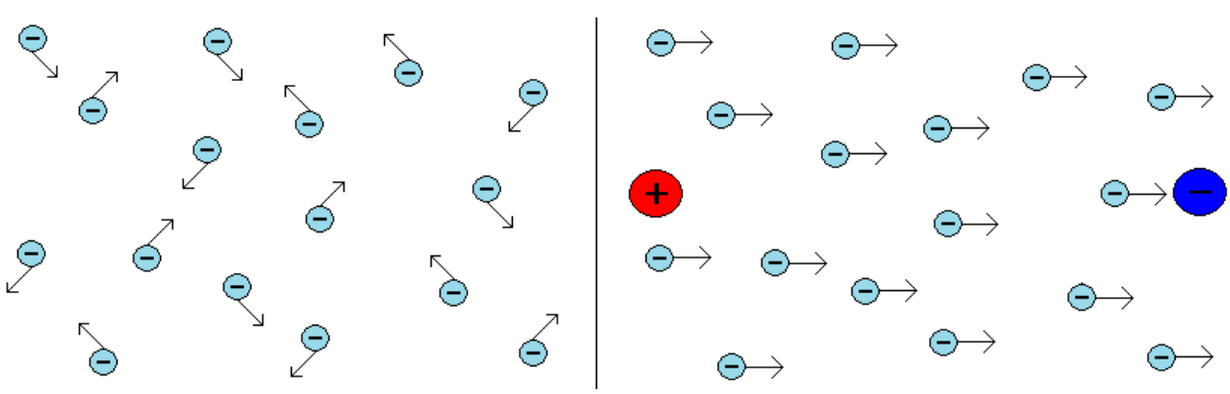
Elektronika to jedno z najważniejszych osiągnięć ludzkości. W XXI wieku z każdej strony otaczają nas przeróżne układy elektroniczne. Są one podstawowym elementem komputerów, tabletów, smartfonów, telewizorów, konsol, aparatów cyfrowych, samochodów, lodówek, pralek czy kuchenek mikrofalowych. Wykorzystywane są do sterowania taśmami produkcyjnymi, robotami, do nawigacji (GPS), w urządzeniach medycznych (pomiar akcji serca, mózgu, tętna czy obrazowanie ludzkiego ciała), w misjach kosmicznych (sondy, łaziki, wahadłowce), w budynkach inteligentnych (sterowanie oświetleniem, ogrzewaniem, temperaturą pomieszczenia), w samochodach, statkach, samolotach. Ciężko jest wyobrazić sobie życie bez elektroniki.

Czym jest elektronika? Jest to dziedzina zajmująca się wytwarzaniem i przetwarzaniem sygnałów w postaci prądów i napięć elektrycznych. Rozróżnia się elektronikę analogową i elektronikę cyfrową.

Powyższa definicja może wydawać się nieco skomplikowana, dlatego wyjaśnimy ją szczegółowo poniżej.

Prąd elektryczny

Podstawowym składnikiem materii we wszechświecie jest atom. Każdy atom składa się z elektronów, protonów i neutronów. Przyjmuje się, że elektron ma ładunek ujemny, a proton dodatni. W normalnych warunkach te ładunki poruszają się losowo, czyli każdy z nich w innym kierunku. Natomiast ich uporządkowany ruch nazwiemy prądem elektrycznym. Ilość przepływającego ładunku nazywamy natężeniem prądu elektrycznego.



Rysunek 1. Ruch chaotyczny i uporządkowany.

Napięcie elektryczne

Prąd elektryczny oczywiście nie pojawia się sam z siebie. Jest wymuszony przez "zewnętrzną siłę", którą nazywamy napięciem elektrycznym. Bardziej skomplikowana, znana z lekcji fizyki, definicja określa napięcie jako zależność pomiędzy pracą wykonaną podczas przenoszenia ładunku, a wartością tego ładunku.

Rezystancja (opór)

Podczas definiowania prądu i napięcia nie może zabraknąć omówienia rezystancji. Opór elektryczny to miara przeciwstawienia się przepływowi prądu.

Prawo Ohma

Trzy wyżej wymienione pojęcia łączy tak zwane prawo Ohma. Nazwane tak na cześć Georga Ohma, niemieckiego matematyka, który w 1826 roku sformułował to prawo. Przyjmuje się następujący zapis:

- natężenie prądu oznacza się symbolem I, jego wartość podaje się w amperach, które oznacza się symbolem A;
- napięcie oznacza się symbolem U, jego wartość podaje się w woltach, które oznacza się symbolem V;
- rezystancję oznacza się symbolem R, jego wartość podaje się w omach, które oznacza się symbolem Ω.

Prawo Ohma zapisujemy następująco:

$$U = I \cdot R$$

Spróbujmy obrazowo przybliżyć te trudne pojęcia, posłużymy się porównaniem do autostrady. Autostrada jest pełna samochodów – w naszym porównaniu – ładunków elektrycznych. Ruch na autostradzie nazwiemy prądem elektrycznym, ponieważ samochody poruszają się w sposób uporządkowany (od bramki wjazdowej do bramki wyjazdowej). Jeśli staniemy na poboczu i policzymy ilość przejeżdżających samochodów w ciągu np. sekundy lub minuty, otrzymamy wartość natężenia. Napięcie elektryczne możemy porównać do prędkości jadących samochodów. Natomiast, rezystancja jest odwrotnością szerokości drogi (liczby pasów).

O czym mówi nam prawo Ohma? Zakładamy, że prędkość pojazdów jest stała. Im szersza jest droga (więcej pasów, mniejsze utrudnienia ruchu,), tym więcej samochodów przejedzie przez dany odcinek (więc mniejsza rezystancja). Im węższa (mniej pasów, większe utrudnienia ruchu), tym mniej samochodów przejedzie ten odcinek w tym samym czasie.

Sygnał

Sygnał to nośnik informacji. Najprostsze przykłady to np. sygnał dźwiękowy, wykorzystywany w zawodach i informujący kiedy należy rozpocząć bieg lub sygnał świetlny, w postaci świateł drogowych, informujący kiedy można wjechać na skrzyżowanie, a kiedy nie.

W elektronice, za pomocą sygnałów, przesyłane są w urządzeniach lub pomiędzy nimi informacje o zmianie lub zaistnieniu pewnych zjawisk np. informujące o temperaturze, ciśnieniu lub włączeniu czy wyłączeniu urządzenia. Obieranie i wysyłanie sygnałów jest więc pewną formą konwersacji pomiędzy urządzeniami lub pomiędzy elementami danego urządzenia.

Elektronika analogowa a elektronika cyfrowa

Podstawowe rozróżnienie pomiędzy elektroniką analogową i cyfrową dotyczy typu wykorzystywanych sygnałów. W elektronice analogowej wykorzystujemy sygnały analogowe, czyli takie, które mogą przybierać dowolne wartości zmieniające się w sposób ciągły. Charakter analogowy ma regulacja głośności np. sprzętu audio.

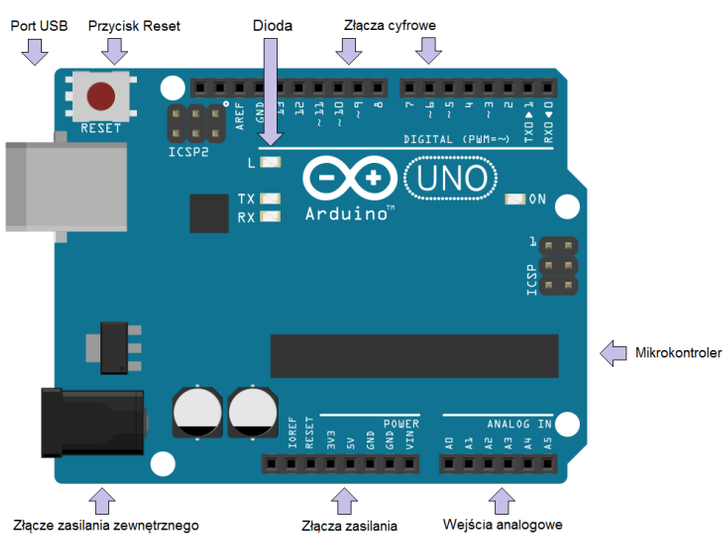
W elektronice cyfrowej używane są sygnały cyfrowe, czyli takie, które przybierają jedynie tak zwany stan niski lub wysoki, reprezentowane zwykle za pomocą cyfr 0 i 1. Standardowe włączniki światła mają charakter cyfrowy.

Mikrokontroler Arduino

Mikrokontroler to mały komputer zamknięty w pojedynczym chipie. Typowy model wyposażony jest w procesor, pamięć służącą do przechowywania danych (RAM, ang. *Random Access Memory* - pamięć o dostępie swobodnym), pamięć stałą (EEPROM, ang. *Electrically-Erasable Programmable Read-Only Memory* – tzw. pamięć nieulotna) i pamięć flash, służącą do przechowywania programów. Podobnie jak komputer posiada złącza wejścia i wyjścia. Urządzenia wejściowe wykorzystywane przez komputer to np. myszka, klawiatura, skaner. Urządzenia wyjściowe to drukarka lub monitor. Złącza wejścia w Arduino służą do odbierania sygnałów np. z czujników oświetlenia, ciśnienia lub temperatury. Złącza wyjścia mogą posłużyć np. do zapalania diody lub sterowania silnikiem.

Na naszych zajęciach wykorzystywać będziemy platformę Arduino Uno, wyposażoną w mikrokontroler ATmega328.

Budowa Arduino Uno



Rysunek 2. Płytka Arduino Uno.

Złącze zasilania zewnętrznego – aby mikrokontroler działał, niezależnie od tego czy jest podłączony do komputera czy nie, musimy zapewnić mu źródło zasilania. Może być nim zestaw baterii lub zewnętrzny zasilacz.

Port USB – port wykorzystywany do komunikacji z komputerem. Za każdym razem kiedy chcemy umieścić nowy program (lub stary, ale zmodyfikowany) w pamięci Arduino, musimy płytkę podłączyć do komputera za pomocą kabla USB i tego złącza.
Złącza cyfrowe - te złącza można wykorzystywać zarówno jako wejścia jak i wyjścia sygnałów. Są one przystosowanie do obierania i wysyłania tak zwanych stanów niskich (0 V) oraz stanów wysokich (5 V).
Wejścia analogowe – złącza te można wykorzystywać do pomiaru przyłożonego napięcia. Zwykle wykorzystuje się je do obierania wartości sygnałów analogowych pochodzących z czujników. Do każdego z tych wejść można doprowadzić napięcie w zakresie od 0 do 5 V.
Złącza zasilania – jeśli dodatkowe moduły, które chcemy podłączyć do naszej płytki Arduino wymagają dodatkowego zasilania, możemy wykorzystać te złącza. Dostarczają one napięcie 3,3 V oraz 5 V.
Mikrokontroler – Najważniejszy element systemu. Opisany powyżej.
Przycisk reset – przycisk umożliwiający ponowne uruchomienie programu z pamięci.

Pierwsze uruchomienie

Podłącz płytkę Arduino do dowolnego portu USB komputera za pomocą kabla. Jeśli wszystkie urządzenia są sprawne i połączenie jest poprawne, na płytce powinny zacząć migać diody.

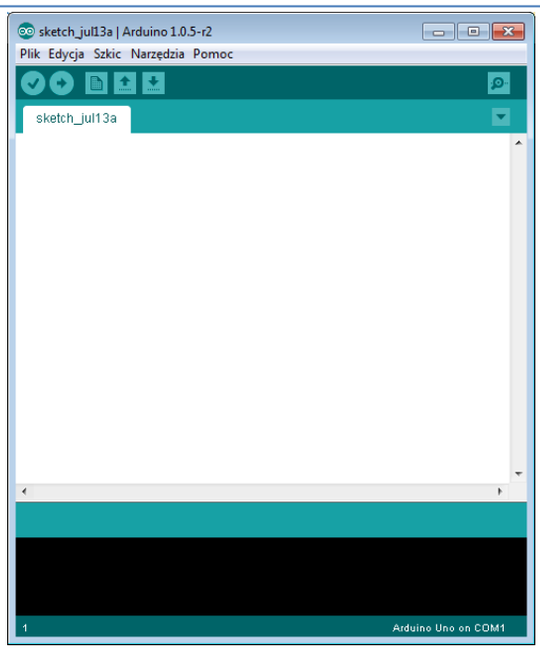
Następnie na pulpicie komputera znajdź ikonę:

Rysunek 3. Ikona środowiska Arduino.

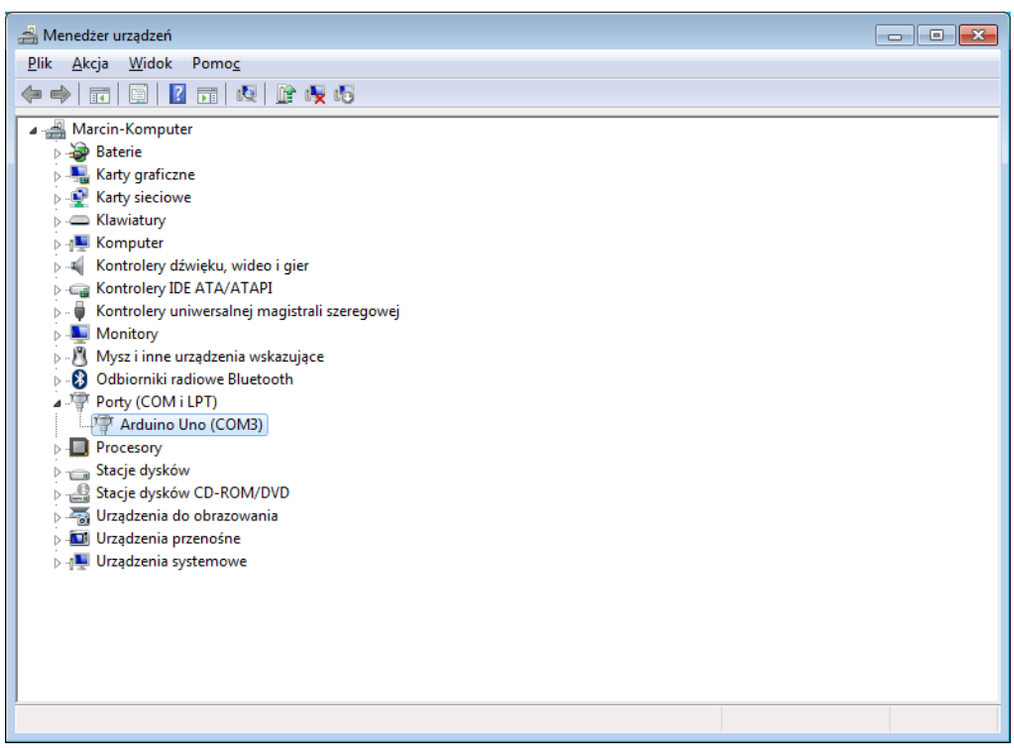
lub na dysku twardym w lokalizacji ‘C:\Program Files (x86)\Arduino” poszukaj pliku „Arduino.exe”

Dwukrotnie klikając w ikonę uruchomisz środowisko programistyczne Arduino. Jeśli program został zainstalowany i uruchomiony poprawnie, na ekranie komputera powinno pojawić się okno podobne jak na rysunku poniżej.

Podczas pierwszego uruchomienia warto sprawdzić czy ustawienia programu są poprawne. Aby to sprawdzić w górnym menu wybierz „Narzędzia”, następnie „Płytka” i wybierz „Arduino Uno”. Teraz sprawdź czy wybrano prawidłowy port. W „Panelu Sterowania” komputera znajdź „Menadżer Urządzeń”. Na liście kategorii urządzeń rozwiń „Porty (COM i LPT)”. Nasz port opisany jest jako „Arduino Uno (COMxx)”, gdzie xx to numer portu, który poszukujemy. Wejdź teraz z powrotem do programu Arduino. W górnym menu wybierz „Narzędzia”, następnie „Port szeregowy” i wybierz ten port, który został znaleziony w Menadżerze Urządzeń.



Rysunek 4. Standardowe okno środowiska Arduino.



Rysunek 5 Panel sterowania z zainstalowanym Arduino.

Programowanie

Aby przekazać mikrokontrolerowi jakie zadania ma zrealizować musimy napisać program. Niestety komputery nie rozumieją poleceń opisanych w języku wykorzystywanym do porozumiewania się między ludźmi jak np. „zapal czerwoną diodę”, „zmierz prąd z czujnika”. Z tego względu niezbędne jest zastosowanie języka zrozumiałego dla danej maszyny. Wykorzystamy język sztuczny, stworzony specjalnie na potrzeby komunikacji z komputerami, nazywany językiem C++. Jeśli uda nam się opanować podstawy tego języka będziemy w stanie porozumieć się z płytką Arduino.

Mówi się, że komputery porozumiewają się ciągami zer i jedynek. Potrzebujemy więc tłumacza, który przetłumaczy to co napisaliśmy w języku C na wspomniane zera i jedynki. Tacy tłumacze fachowo nazywani są kompilatorami. Kompilacja i wysłanie programu na płytkę uruchomi się gdy w oknie programu klikniesz przycisk ze strzałką skierowaną w prawo.

Za każdym razem gdy będziemy wysłać nowy program na płytkę, będziemy program kompilować i używać do tego tej ikonki.

Pierwszy program

Każdy napisany program powinien zaczynać się następująco:

```
void setup(){
}
void loop(){
}
```

Wyślij program na płytkę. Co się wydarzyło? Prawdopodobnie w dolnym pasku programu pojawiła się informacja „Wielkość binarna szkicu: 1 084 bajtów (maksymalnie: 32 256 bajtów)". Oznacza to, że program skompilował i wysłał się poprawnie. Jednak nic się nie dzieje. Dlaczego? Ponieważ nasz program jest pusty. Powyższa ramka jest szkieletem programu i służy kompilatorowi do rozpoznania, że posługujemy się językiem C (a nie np. polskim).

Program podzielony jest na dwie funkcje. Pierwsza opisana jako „void setup()" (na nasze potrzeby „setup" przetłumaczymy jako ustawienia) będzie zawierała podstawowe ustawienia programu, np. które złącza cyfrowe będziemy wykorzystywać. Jej zawartość będzie wykonywa tylko raz, podczas uruchomienia lub zrestartowania płytki. Druga funkcja to „void loop()" ("loop" – pętla). Tu wpisujemy główną część naszego programu, czyli dokładnie to, co ma się wykonać. Jak długo? Jeśli nie zaznaczymy tego inaczej, w nieskończoność lub do czasu odłączenia zasilania.

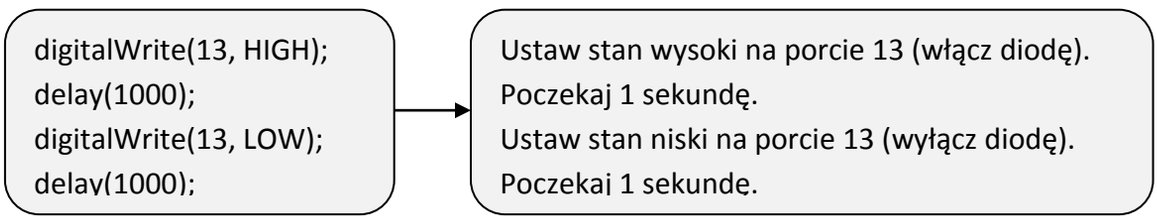
Zmodyfikujemy program tak, aby sterować wbudowaną w Arduino diodą. Kod programu wygląda następująco:

```
void setup(){
      pinMode(13, OUTPUT);
}
void loop(){
      digitalWrite(13, HIGH);
      delay(1000);
      digitalWrite(13, LOW);
      delay(1000);
}
```

Rozbudowaliśmy nasz program o trzy polecenia: „pinMode", „digitalWrite" oraz „delay". Przyjrzymy się najpierw funkcji „pinMode". Służy ona do określenia czy dany pin (złącze) ma być wykorzystywany jako wejście (INPUT) czy wyjście (OUTPUT). Następna funkcja to „digitalWrite". Jak sama nazwa wskazuje służy ona do zapisania stanu (wysokiego lub niskiego) na wyjściu portu cyfrowego. W nawiasie zapisane są parametry tej funkcji, które oznaczają numer portu

cyfrowego, na którym mamy ustawić stan, oraz to jaki stan ma zostać ustawiony. HIGH to stan wysoki, LOW to stan niski. Funkcja „delay" to funkcja wstrzymująca wykonanie następnej funkcji na czas podany w nawiasie, wyrażony w milisekundach.

W zrozumieniu powyższego opisu może przydać się poniższy schemat.



Sprawdź teraz co się stanie, jeśli:

1. Usuniesz linijki digitalWrite(13, LOW) oraz delay(1000);
2. Usuniesz linijki digitalWrite(13, HIGH) oraz delay(1000);
3. Zamienisz delay(1000) na delay(500);
4. Zamienisz delay(1000) na delay(2000).

A teraz spróbuj napisać program, który po kolei wykonuje następujące czynności:

1. Dioda włączona jest przez sekundę i następnie przez sekundę jest wyłączona;
2. Dioda włączona jest przez dwie sekundy i następnie przez dwie sekundy jest wyłączona;
3. Dioda włączona jest przez trzy sekundy i następnie przez trzy sekundy jest wyłączona.

O czym należy pamiętać?

1. Pamiętaj, aby wszystko co dotyczy funkcji „setup()" i funkcji „loop()" umieszczone było wewnątrz nawiasów klamrowych.
2. Pamiętaj, aby każda linijka kończyła się średnikiem. O wyjątkach powiemy sobie na kolejnych zajęciach.

Zmienne

W pierwotnym programie z diodą zapaloną przez jedną sekundę, do pinu numer 13 odwołaliśmy się trzykrotnie. Gdyby tych odwołań było znacznie więcej, na przykład 20 i chcielibyśmy zmienić pin z 13 na 12 to musielibyśmy dokonać aż 20 poprawek! Jest to bardzo kłopotliwe, dlatego z pomocą przychodzą nam zmienne (oczywiście to nie jest ich jedyne zastosowanie). Zmienne możemy potraktować jako wartości liczbowe, którym nadajemy imię.

Do naszego programu (poniżej) dopisaliśmy pierwszą linijkę (pogrubioną). Nasza pierwsza zmienna otrzymała imię „dioda", oczywiście może nazywać się dowolnie: MAREK, zmienna, MaRgArYnA lub Arduino. Warto jednak zmiennym nadawać takie nazwy, które później ułatwią nam identyfikację, tzn. powiedzą nam coś o tym co kryje się za dana zmienna: dioda1, dioda2, portCzujnika, sterowanieSilnikiemPrawym, itp.

```
int dioda = 13;
void setup(){
      pinMode(dioda, OUTPUT);
}
void loop(){
      digitalWrite(dioda, HIGH);
      delay(1000);
      digitalWrite(dioda, LOW);
      delay(1000);
}
```

Przed nazwą „dioda” musieliśmy dopisać „int”. Jest to określenie typu liczby, której nadajemy imię. „Int” to skrót od angielskiego słowa „integer” oznaczającego liczbę całkowitą. Zwróć uwagę na to, że zastąpiliśmy wszystkie 13 (oczywiście poza pierwszą linijką) imieniem „dioda”.

A co jeśli chcielibyśmy, żeby dioda za każdym razem świeciła o 0,1 s dłużej i o tyle samo dłużej pozostawała zgaszona? Można za każdym razem deklarować wartość funkcji delay o 0,1 s więcej, tak jak w poniższym przykładzie. Nie uzyskamy jednak oczekiwanego efektu, czas nie będzie zwiększał się w nieskończoność.

```
void setup(){
      pinMode(13, OUTPUT);
}
void loop(){
      digitalWrite(13, HIGH);
      delay(1000);
      digitalWrite(13, LOW);
      delay(1000);
      digitalWrite(13, HIGH);
      delay(1100);
      digitalWrite(13, LOW);
      delay(1100);
      digitalWrite(13, HIGH);
      delay(1200);
      digitalWrite(13, LOW);
      delay(1200);
}
```

Nasz program wykona się następująco:

1. Włącz diodę na 1 s. wyłącz na 1 s,
2. Włącz diodę na 2 s. wyłącz na 2 s,
3. Włącz diodę na 3 s. wyłącz na 3 s,
4. Włącz diodę na 1 s. wyłącz na 1 s,
5. Włącz diodę na 2 s. wyłącz na 2 s,
6. Włącz diodę na 3 s. wyłącz na 3 s,
7. Włącz diodę na 1 s. wyłącz na 1 s itd.

Oczywiście możemy doklejać kolejne linijki kodu, jednak bardzo szybko zapełnimy dostępną pamięć Arduino. Jak wiec w prosty sposób osiągnąć cel? Tu także pomocne okażą się zmienne. Nasz program musimy powiększyć o dwie linijki (pogrubione):

```
int dioda = 13;
int czas = 1000;
void setup(){
      pinMode(dioda, OUTPUT);
}
void loop(){
      digitalWrite(dioda, HIGH);
      delay(czas);
      digitalWrite(dioda, LOW);
      delay(czas);
      czas=czas+100;
}
```

Pierwsza dołączona linijka odnosi do początkowego czasu równego 1000 ms. Druga z dołączonych linijek może wydawać się bardziej zagadkowa. Można ją interpretować następująco: „oblicz nową wartość zmiennej ‘czas’ dodając do starej wartości liczbę 100”.

W kolejnych wykonaniach programu (nazywanych też iteracjami), zmienna czas zmieniać się będzie następująco:

1. 1000,
2. 1000+100=1100,
3. 1100+100=1200,
4. 1200+100=1300 itd.

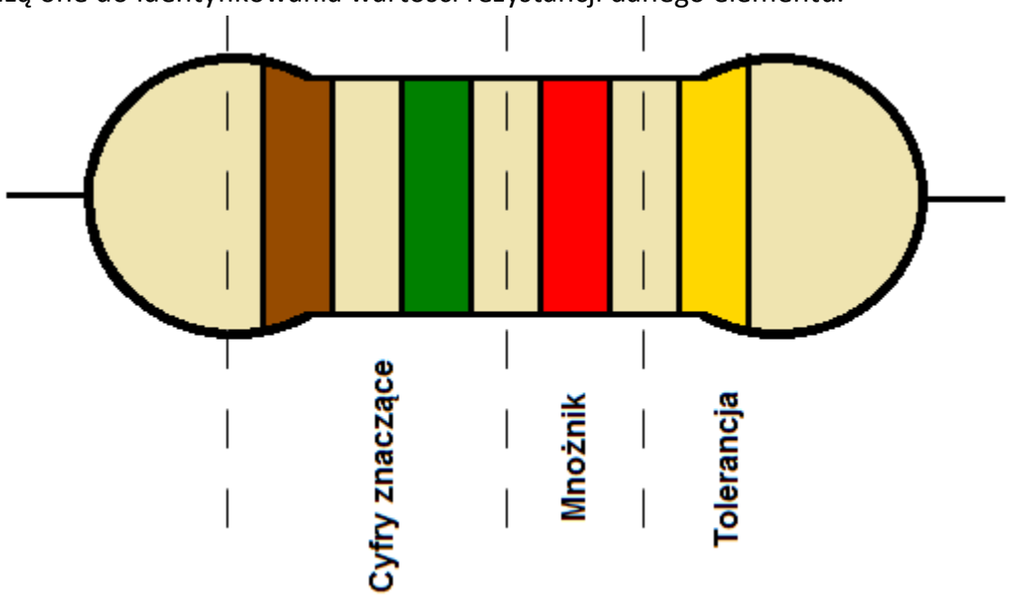
Czas będzie się zwiększał aż do zatrzymania programu (odłączenia zasilania, kabla USB itp.). Sprawdź co się stanie, jeśli na płytce wciśniesz przycisk „RESET”.

Zajęcia 2: Pierwszy układ

Na drugich zajęciach nauczymy się jak podłączyć elementy elektroniczne do naszego Arduino. Będą nimi dioda LED, rezystor oraz przycisk.

Rezystor (opornik)

Rezystory to podstawowe elementy układów elektronicznych. Przypomnij sobie co na temat oporu mówiliśmy na poprzednich zajęciach. Rezystancja to miara przeciwstawienia się przepływowi prądu. Wszystkie używane przez nas rezystory posiadają na obudowie kolorowe paski. Służą one do identyfikowania wartości rezystancji danego elementu.



Rysunek 6. Oznaczenia rezystora.

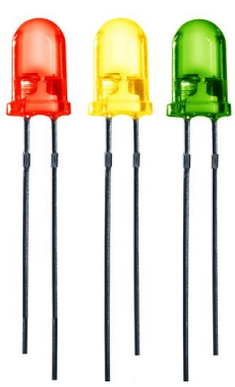
Tabela 1. Oznaczenia rezystora.

Kolor		Liczba	Mnożnik	Tolerancja
Srebrny		-	x 0,01	± 10 %
Złoty		-	x 0,01	± 5 %
Czarny		0	x 1	-
Brązowy		1	x 10	± 1 %
Czerwony		2	x 100	± 2 %
Pomarańczowy		3	x 1000	± 15 %
Żółty		4	x 10000	-
Zielony		5	x 100000	± 0,5 %
Niebieski		6	x 1000000	± 0,25 %
Fioletowy		7	x 10000000	± 0,1 %
Szary		8	-	-
Biały		9	-	-
Brak		-	-	± 20 %

Do wyjaśnienia w jaki sposób odczytujemy wartość rezystancji za pomocą pasków posłużymy się rezystorem z rysunku 5. Pasek brązowy to cyfra 1, pasek zielony to cyfra 5. W ten sposób powstała liczba 15. Następny, czerwony pasek to mnożnik „x100". Aby wyznaczyć wartość rezystancji otrzymaną liczbę 15 musimy przemnożyć przez 100. Rezystor z naszego przykładu ma więc rezystancję równą 1500 Ω. Ostatni, złoty pasek oznacza tolerancję ± 5 %. Oznacza to, że rezystor nigdy nie będzie miał wartości rezystancji równej idealnie 1500 Ω, ale jego wartość znajdzie się w przedziale 1500 ± 0,5 % czyli pomiędzy 1425 Ω a 1575 Ω.

Dioda LED

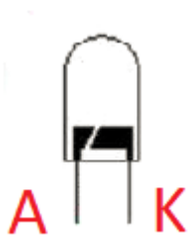
Diody elektroluminescencyjne (LED, pochodzi od angielskiego *Light Emitting Diode*) są zaprojektowane tak, aby emitowały duże ilości światła widzialnego. Dioda to element półprzewodnikowy, co oznacza, że czasem jest przewodnikiem, a czasem nie. Jest to zależne od jej połączenia. W odróżnieniu od rezystora, dioda jest elementem elektronicznym posiadającym dwie nóżki różnej długości. Dłuższą nóżkę nazywamy anodą, a krótszą katodą. Jeśli chcemy aby dioda emitowała światło, anodę musimy podłączyć do złącza zasilania oznaczonego jako „+", natomiast katodę do złącza oznaczonego jako „-".



Rysunek 7. Diody LED.

UWAGA!! Nie podłączaj jednak diody nigdy bezpośrednio do zasilania. Dioda, aby świecić potrzebuje odpowiedniego natężenia prądu (ok. 20mA). Jeśli zostanie podłączona bezpośrednio, prąd przez nią płynący będzie dużo większy. Dlatego pamiętajmy o wykorzystaniu rezystora, który obniży wartość płynącego prądu.

Co zrobić jeśli z jakiegoś powodu obie nóżki diod są tej samej długości? Zajrzyjmy do środka diody. Anodę możemy poznać po krótszej blaszce, natomiast katodę po dłuższej. Przy diodzie czerwonej zasada ta jest odwrotna.

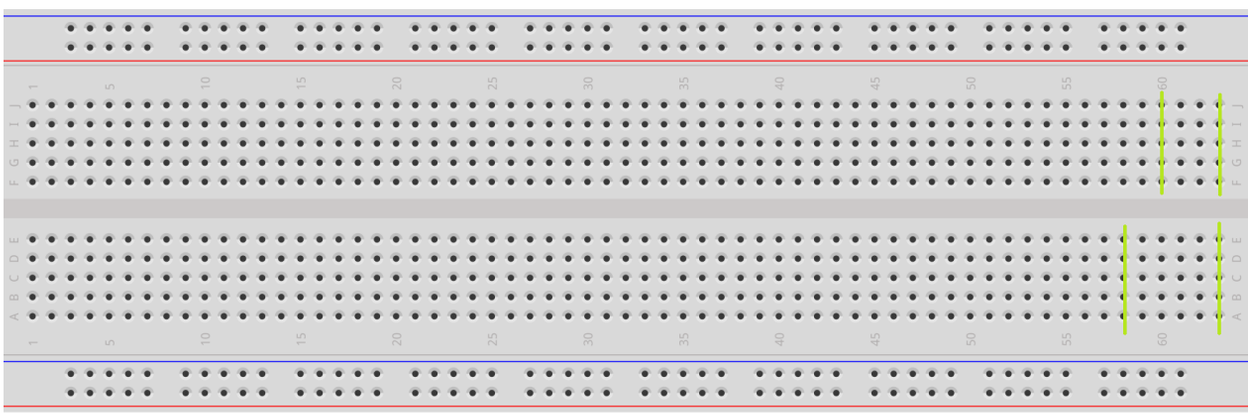


Rysunek 8. Budowa diody LED.

Płytka stykowa

Płytki stykowe wykorzystywane są zwykle w celu wypróbowania prototypowych wersji układów, bądź do nauki działania różnych elementów elektronicznych.

Płytki będziemy używać w prawie każdym budowanym układzie do łączenia elementów elektronicznych, dlatego ważne jest, by zrozumieć jak prawidłowo z niej korzystać.

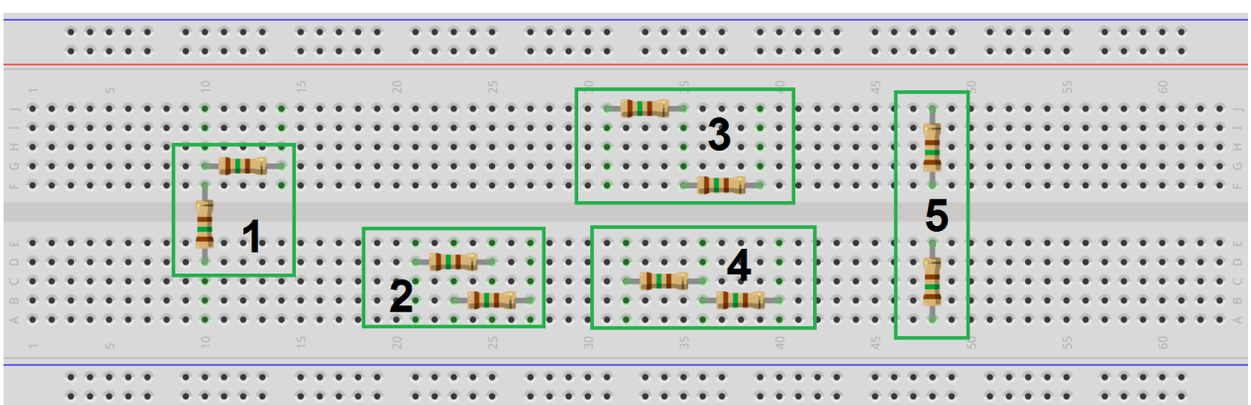


Rysunek 9. Płytka stykowa.

Płytkę możemy podzielić na dwa obszary: dziurki, do których będziemy podłączać zasilanie (wzdłuż niebieskiej i czerwonej linii) oraz dziurki, do których będziemy podłączać elementy elektroniczne i złącza wykorzystywane do połączeń elementów (oznaczone zieloną linią, a także liczbami od 1 do 60 oraz literami od A do J).

Pamiętaj, że wszystkie dziurki wydłuż czerwonej linii są ze sobą podłączone, można porównać do przedłużacza. Wystarczy, że napięcie 5 V z płytki Arduino podłączymy do którejkolwiek z dziurek przy czerwonej linii, a potencjał ten pojawi się na każdej z dziurek w tej linii. Podobnie jest w przypadku niebieskiej linii, tam też wszystkie dziurki są ze sobą połączone.

Sprawa nieco się komplikuje przy reszcie dziurek. Tam są one połączone wzdłuż zielonej linii. Tak więc w rzędzie oznaczonym jako „1” połączone są ze sobą dziurki A,B,C,D,E oraz F,G,H,I,J. Podobnie w przypadku rzędu oznaczonego jako „2”, „3”, itd., aż do „60”. Pamiętaj, że nie ma połączenia pomiędzy E i F!



Rysunek 10. Przykłady prawidłowych i nieprawidłowych połączeń rezystorów.

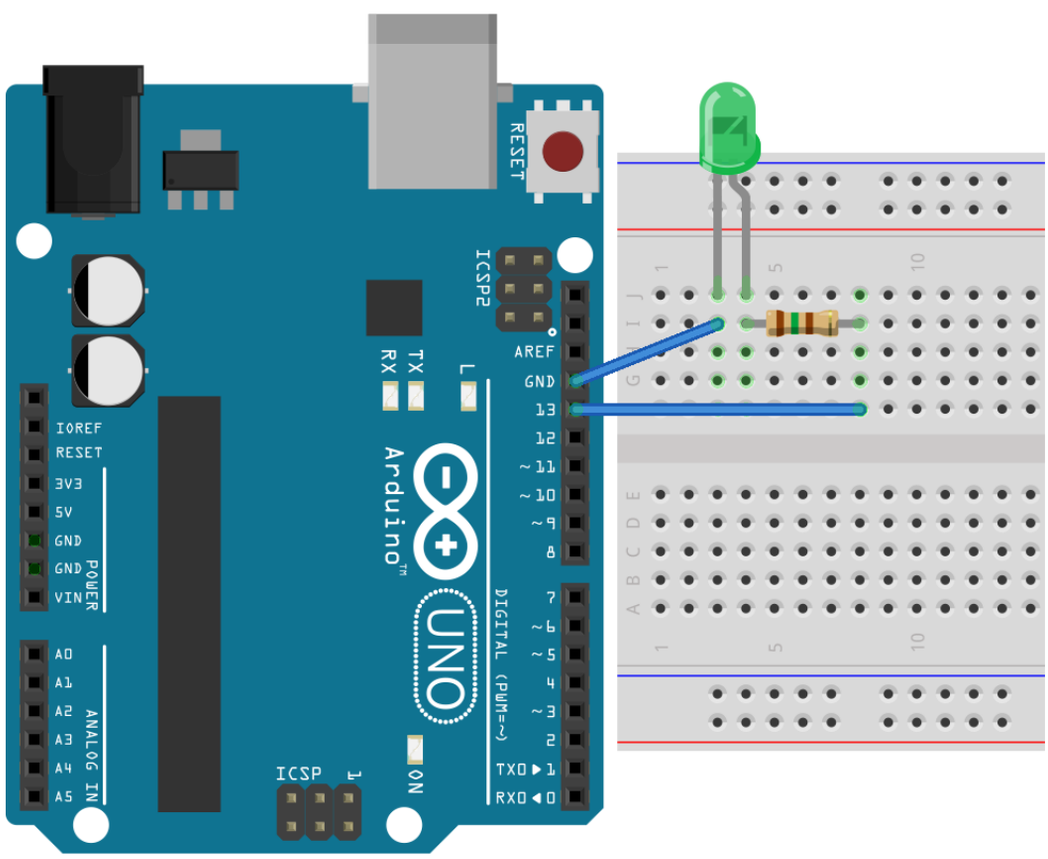
Na rysunku powyżej przedstawiono przykładowe połączenia rezystorów. Jak myślisz, które połączone są poprawnie? Poprawnie połączone są rezystory z sekcji 1,3 oraz 4. Między rezystorami z sekcji 2 i 5 nie ma połączenia.

Pierwszy układ

Do budowy pierwszego układu potrzebujemy:

1. Komputer,
2. Kabel USB,
3. Płytkę Arduino,
4. Płytkę stykową,
5. Rezystor 150 Ω,
6. Diody LED,
7. Przewody.

Układ połącz jak na poniższym rysunku i uruchom program z poprzednich zajęć. Zauważ, że obie diody (wbudowana i na płytce stykowej migają tak samo). Jeśli nie chcemy, aby dioda wbudowana migała wystarczy, że zmienimy w pierwszej linijce naszego programu numer złącza (int dioda = 13) na dowolny inny.



Rysunek 11. Pierwszy układ z diodą i rezystorem.

Polecenie „if…else”

W naszym programie czas świecenia diody zwiększał się z każdą iteracją (powtórzeniem) o 0,1 s. A co jeśli chcemy aby czas zwiększał się tylko w zakresie od 0,1 do 2 s? Aby nam się to udało potrzebujemy skorzystać z instrukcji warunkowej „if” (pol. *jeśli*) wraz z „else” (pol. *w przeciwnym wypadku*). Zasadę jej działania można przedstawić następująco:

```
if(pada deszcz){
      WezParasol;
}
else {
      NieBierzParasola;
}
```

Zastosowanie w programie:

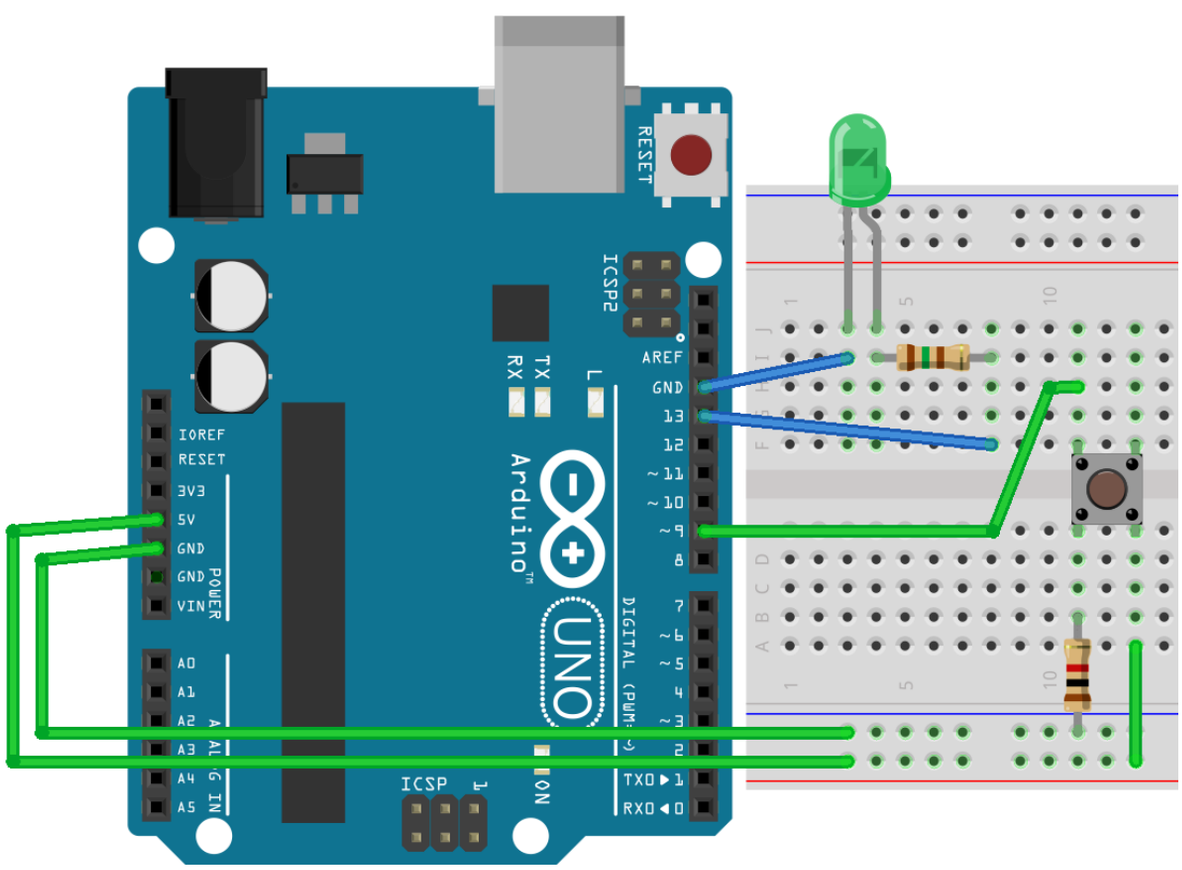
```
int diode = 12;
int czas = 500;
void setup(){
      pinMode(dioda, OUTPUT);
}
void loop(){
      digitalWrite(dioda, HIGH);
      delay(czas);
      digitalWrite(dioda, LOW);
      delay(czas);
      czas = czas+500;
      if(czas > 2500){
            czas = 500;
      }
      else{
      }
}
```

Program został nieco zmodyfikowany. Złącze diody zmieniliśmy na 12, a podstawę czasu na 0,5 s. Dodany został ostatni, pogrubiony fragment. O czym mówi? W każdej iteracji nasza zmienna „czas” zwiększa się o 0,5 s, na końcu iteracji sprawdza warunek czy zmienna czas jest większa niż 2500 ms - linia „if(czas>2500)”. Jeśli warunek nie jest spełniony to nic się nie dzieje. Jeśli jest spełniony wykonuje się to co zapisane jest w nawiasie klamrowym czyli „czas=500”, zmieniamy więc wartość zmiennej czas. To co się dzieje w kolejnych iteracjach można rozpisać następująco:

1. Podczas uruchomienia programu zmienna czas=500,
2. W pierwszej iteracji dioda załącza się i wyłącza, zmienna „czas” zwiększana jest o 500 ms i jej wartość wynosi 1000. Na końcu sprawdzany jest warunek. Ponieważ 1000 nie jest większe niż 2500, warunek nie jest spełniony i nic się nie dzieje.

3. W drugiej iteracji dioda załącza się i wyłącza, zmienna „czas” zwiększana jest o 500 ms i jej wartość wynosi 1500. Na końcu sprawdzany jest warunek. Ponieważ 1500 nie jest większe niż 2500, warunek nie jest spełniony i nic się nie dzieje.
4. W trzeciej iteracji dioda załącza się i wyłącza, zmienna „czas” zwiększana jest o 500 ms i jej wartość wynosi 2000. Na końcu sprawdzany jest warunek. Ponieważ 2000 nie jest większe niż 2500, warunek nie jest spełniony i nic się nie dzieje.
5. W czwartej iteracji dioda załącza się i wyłącza, zmienna „czas” zwiększana jest o 500 ms i jej wartość wynosi 2500. Na końcu sprawdzany jest warunek. Ponieważ 2500 nie jest większe niż 2500, warunek nie jest spełniony i nic się nie dzieje.
6. W piątej iteracji dioda załącza się i wyłącza, zmienna „czas” zwiększana jest o 500 ms i jej wartość wynosi 3000. Na końcu sprawdzany jest warunek. Ponieważ 3000 jest większe niż 2500, warunek jest spełniony i wartość zmiennej „czas” zmieniana jest na 500.
7. W szóstej iteracji dioda załącza się i wyłącza, zmienna „czas” zwiększana jest o 500 ms i jej wartość wynosi 1000. Na końcu sprawdzany jest warunek. Ponieważ 1000 nie jest większe niż 2500, warunek nie jest spełniony i nic się nie dzieje, itd.

W drugim przykładzie wykorzystamy przycisk, który włączy diodę. Układ rozbuduj tak, jak jest to przedstawione na poniższym rysunku (część z przewodami oznaczonymi kolorem zielonym). Dodatkowo potrzebujemy przycisku i rezystora 10 kΩ.



Rysunek 12. Rozbudowany układ z przyciskiem.

W programie deklarujemy do których złącz podłączymy diodę i przycisk. Tworzymy także zmienną, która będzie sprawdzać czy przycisk został wciśnięty (czyWcisnietyPrzycisk) i ustawiamy jej wartość na LOW, co oznacza, że podczas uruchomienia programu przycisk będzie uważany za niewciśnięty.

W funkcji „setup()" deklarujemy, że złącze połączone z diodą będzie wysyłało sygnał (OUTPUT – wyjście), a złącze połączone z przyciskiem będzie odbierało sygnały (INPUT – wejście).

Pierwsza linia w funkcji „loop()" sprawdza czy przycisk został wciśnięty. Funkcję „digitalRead(przycisk)" wykorzystuje się do sprawdzenia stanu na złączu „przycisk". Jeśli przycisk jest niewciśnięty, to tym stanem jest stan niski (LOW), jeśli przycisk jest wciśnięty, to tym stanem jest stan wyskoki (HIGH). Odczytany stan jest zapisywany do zmiennej „czyWcisnietoPrzycisk ". O zapaleniu diody decyduje instrukcja "if". Jeśli stan zapisany w zmiennej "czyWcisnietoPrzycisk " jest wysoki <if (czyWcisnietoPrzycisk == HIGH)> to dioda jest załączana <digitalWrite(dioda, HIGH)>, w przeciwnym razie (else) dioda jest wyłączona <digitalWrite(dioda, LOW)>.

```
int przycisk = 9;
int dioda =  12;
int czyWcisnietoPrzycisk  = LOW;

void setup(){
      pinMode(dioda, OUTPUT);
      pinMode(przycisk, INPUT);
}

void loop(){
      czyWcisnietoPrzycisk  = digitalRead(przycisk);

      if (czyWcisnietoPrzycisk  == HIGH){
            digitalWrite(dioda, HIGH);
       }
      else{
            digitalWrite(dioda, LOW);
      }
}
```

Jak myślisz, co się stanie jeśli wszystkie wartości „HIGH" zamienimy na „LOW", a wszystkie wartości „LOW" na „HIGH"? Sprawdź!

Polecenie „for"

Często zdarza się, że potrzebujemy określoną część programu wykonać większą liczbę razy. Pierwszą twoją myślą jest pewnie to aby potrzebny fragment kodu skopiować odpowiednią liczbę razy. Pamiętajmy jednak o ograniczonej pamięci Arduino, która w ten sposób bardzo szybko się zapełni.

```
int diode = 12;
int czas = 500;

void setup(){
	pinMode(dioda, OUTPUT);
}

void loop(){
	digitalWrite(dioda, HIGH);
	delay(czas);
	digitalWrite(dioda, LOW);
	delay(czas);
	digitalWrite(dioda, HIGH);
	delay(czas);
	digitalWrite(dioda, LOW);
	delay(czas);
	digitalWrite(dioda, HIGH);
	delay(czas);
	digitalWrite(dioda, LOW);
	delay(czas);
	digitalWrite(dioda, HIGH);
	delay(czas);
	digitalWrite(dioda, LOW);
	delay(czas);
	digitalWrite(dioda, HIGH);
	delay(czas);
	digitalWrite(dioda, LOW);
	delay(czas);
	digitalWrite(dioda, HIGH);
	delay(czas);
	digitalWrite(dioda, LOW);
	delay(czas);

	delay(3000);
}
```

Kod powyższego programu powoduje, że dioda sześciokrotnie zapala się i gaśnie, a następnie pozostaje zgaszona przez 3 s. Jak widać program może wydawać się dość długi. Jak go skrócić? Wykorzystamy do tego polecenie „for”.

```
int diode = 12;
int czas = 500;
void setup(){
	pinMode(dioda, OUTPUT);
}
void loop(){
	for (int i = 0; i < 6; i = i + 1){
		digitalWrite(dioda, HIGH);
		delay(czas);
```

```
            digitalWrite(dioda, LOW);
            delay(czas);
        }
        delay(3000);
}
```

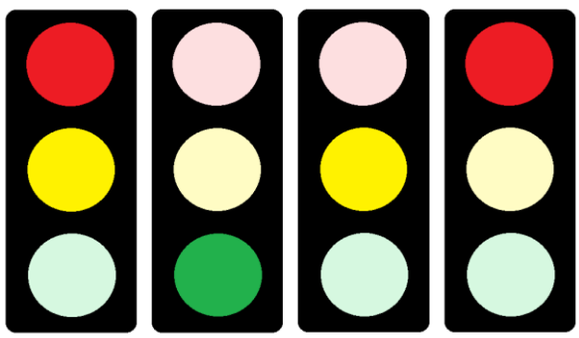
Program w powyższej formie jest znacznie bardziej przejrzysty i krótszy. Przeanalizujmy linijkę „for (int i=0; i<6; i=i+1)". Pozwala ona określić nam ile razy ma wykonywać się zawartość nawiasu klamrowego (czyli włączenie i wyłącznie diody). W komendzie tej wykorzystywany jest licznik oznaczony jako „i". Możemy go skojarzyć z licznikiem przejechanych kilometrów w samochodzie. W naszym programie pozwala liczyć ile razy zostało wykonane to co zapisane jest w nawiasie klamrowym. Pierwsza część „int i=0" mówi nam od jakiej wartości mamy rozpocząć liczenie liczby powtórzeń, natomiast „i<6" kiedy to liczenie zakończyć. Aby obliczyć liczbę wykonań (gdyby wpisane cyfry były inne) wystarczy od 6 odjąć 0, co da nam 6 powtórzeń. Ostatnia część „i=i+1" informuje o tym o ile mamy zwiększać licznik. Zapis „i=i+1" oznacza zwiększenie o jeden. Oczywiście, inaczej niż jest to w samochodzie, nasz licznik może zwiększać się o więcej niż jeden, w zależności od potrzeb.

Zadanie na koniec

Zadaniem na koniec jest napisanie programu wykorzystującego polecenie „for" i wysyłającego sygnał SOS w alfabecie Morse'a. Sygnał ten jest kombinacją następujących sygnałów: krótki, krótki, krótki, długi, długi, długi, krótki, krótki, krótki. Pamiętaj, aby na zakończenie każdej sekwencji dołączyć dłuższą przerwę, aby między kolejnymi sygnałami SOS była wyraźna przerwa.

Zajęcia 3: Światła drogowe

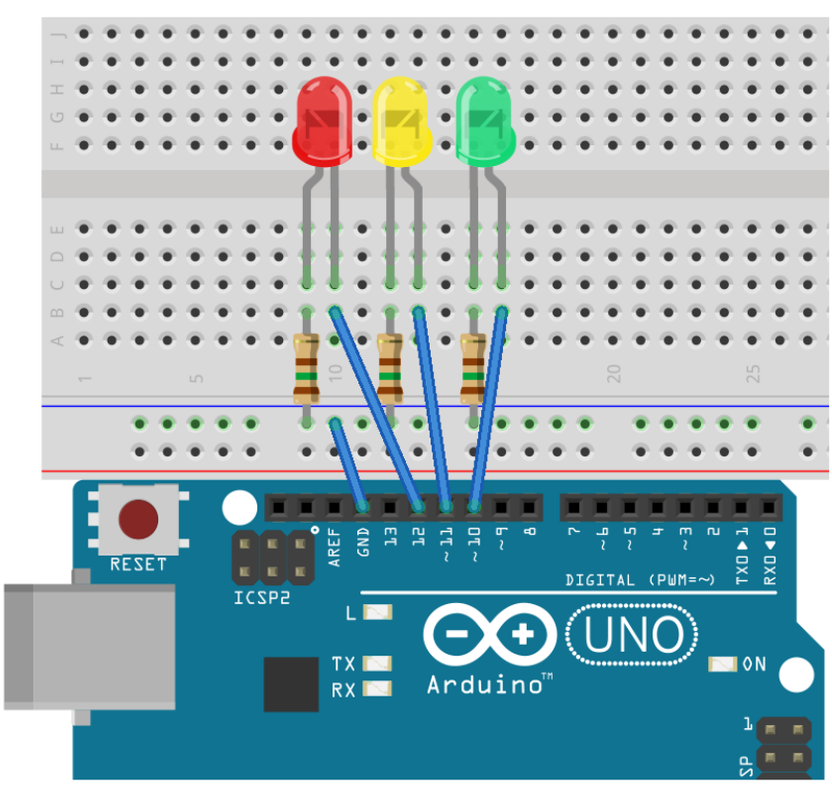
Celem zajęć jest zmontowanie i zaprogramowanie układu sterującego sygnalizacją świetlną. Kolejność zapalania się świateł w sygnalizacji ulicznej w Polsce wygląda następująco:



Rysunek 13. Sygnalizacja drogowa.

Do budowy prototypu potrzebujemy:

1. Komputer,
2. Kabel USB,
3. Płytkę Arduino,
4. Płytkę stykową,
5. 3 rezystory150 Ω,
6. 3 diody LED, po jednej z każdego koloru: czerwonego, żółtego i zielonego.
7. Przewody.



Rysunek 14. Połączenie układu "Sygnalizacja świetlna".

Przygotuj układ połączony jak na powyższym rysunku. Program sterujący układem nie jest trudny. Spróbuj napisać go samodzielnie, korzystając z wiedzy zdobytej na poprzednich zajęciach. Przyjmij, że:

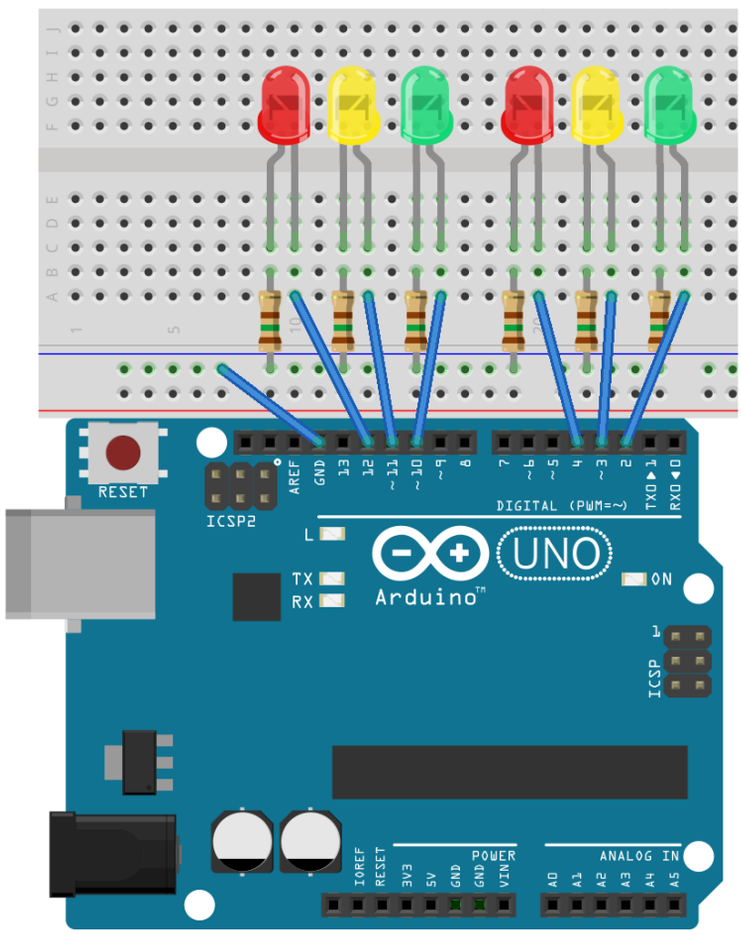
1. Dioda czerwona i zółta włącza się na 2 s.
2. Dioda zielona włącza się na 5 s.
3. Dioda żólta włącza się na 2 s.
4. Dioda czerwona włącza się na 5.

Ta sekwencja powinna poważać się w nieskończoność.

```
int swiatloCzerwone = 12;
int swiatloZolte = 11;
int swiatloZielone = 10;

void setup(){
      pinMode(swiatloCzerwone , OUTPUT);
      pinMode(swiatloZolte, OUTPUT);
      pinMode(swiatloZielone, OUTPUT);
}

void loop(){
      digitalWrite(swiatloCzerwone , HIGH);
      digitalWrite(swiatloZolte, HIGH);
      delay(2000);

      digitalWrite(swiatloCzerwone , LOW);
      digitalWrite(swiatloZolte, LOW);
      digitalWrite(swiatloZielone, HIGH);
      delay(5000);

      digitalWrite(swiatloZielone, LOW);
      digitalWrite(swiatloZolte, HIGH);
      delay(2000);

      digitalWrite(swiatloZolte, LOW);
      digitalWrite(swiatloCzerwone , HIGH);
      delay(5000);
}
```

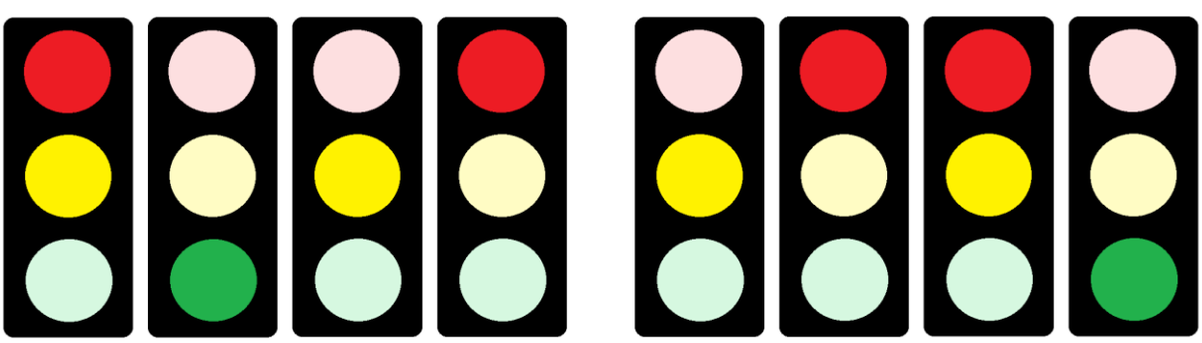
W pierwszej części programu nazwaliśmy użyte złącza (10 dla zielonej diody, 11 dla żółtej i 12 dla czerwonej). W funkcji „setup()” zadeklarowaliśmy, że złącza będą złączami wyjściowymi. W funkcji „loop()” najpierw zostają włączone diody z pierwszej sekcji z rysunku 9, czyli czerwona i żółta. Następnie zostają one wyłączone i włączona zostaje zielona. W trzeciej sekcji zielone zostaje wyłączone i włączone zostaje żółte. W sekcji czwartej żółte zostaje zgaszone i zapala się dioda czerwona.

W drugiej części zajęć przygotujemy światła, które możemy spotkać na skrzyżowaniu dla przecinających się kierunków jazdy. Potrzebujemy dodatkowe diody (zieloną, czerwoną i żółtą), trzy rezystory 150 Ω i przewody. Układ należy połączyć tak, jak na poniższym rysunku.



Rysunek 15. Połączenie układu.

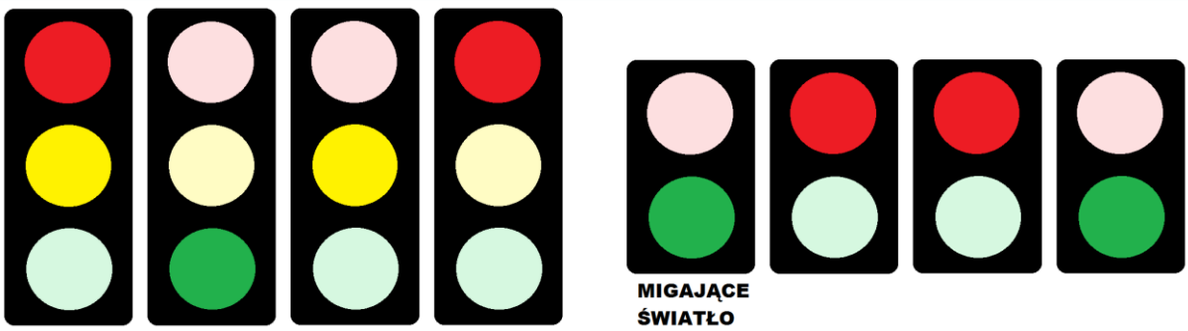
Ten program także nie jest skomplikowany, spróbuj napisać go samodzielnie i w razie problemów spójrz na gotowe rozwiązanie (zamieszone poniżej).



Rysunek 16. Sygnalizacja świetlna dla przecinających się kierunków jazdy.

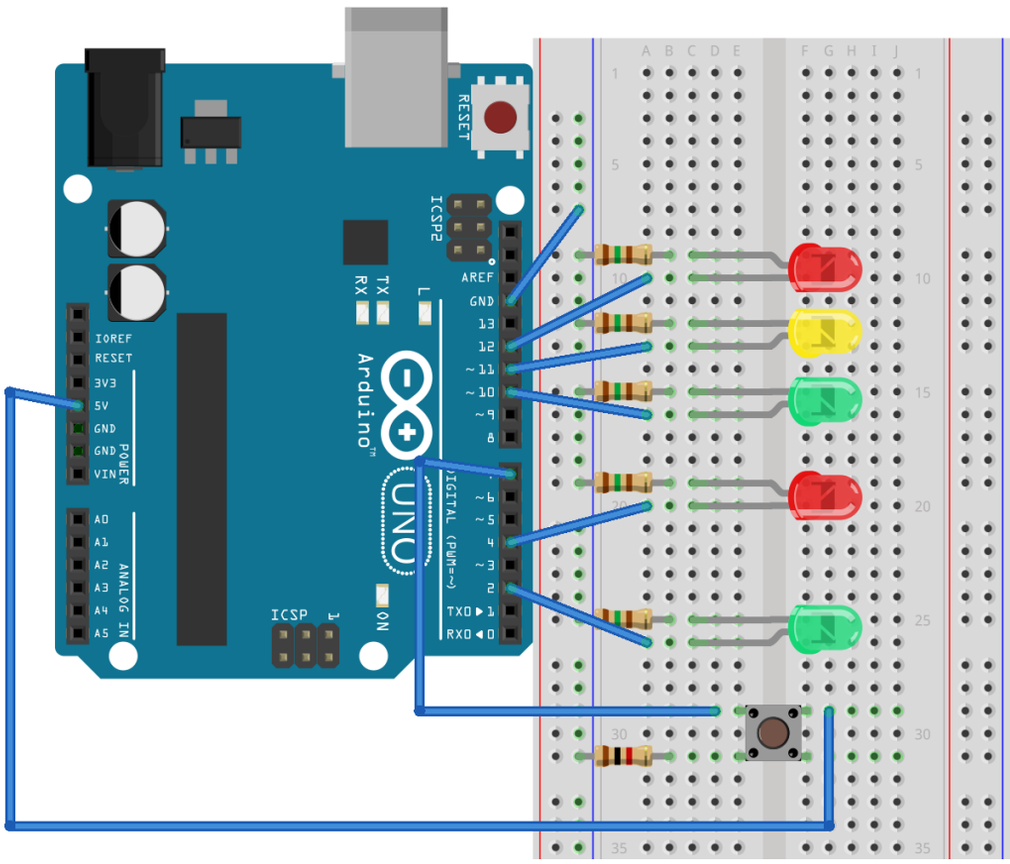
```
int swiatloCzerwone1 = 12;
int swiatloZolte1 = 11;
int swiatloZielone1 = 10;
int swiatloCzerwone2 = 4;
int swiatloZolte2 = 3;
int swiatloZielone2 = 2;

void setup(){
      pinMode(swiatloCzerwone1, OUTPUT);
      pinMode(swiatloZolte1, OUTPUT);
      pinMode(swiatloZielone1, OUTPUT);
      pinMode(swiatloCzerwone2, OUTPUT);
      pinMode(swiatloZolte2, OUTPUT);
      pinMode(swiatloZielone2, OUTPUT);
}

void loop(){
      digitalWrite(swiatloCzerwone1, HIGH);
      digitalWrite(swiatloZolte1, HIGH);
      digitalWrite(swiatloZolte2, HIGH);
      delay(2000);

      digitalWrite(swiatloCzerwone1, LOW);
      digitalWrite(swiatloZolte1, LOW);
      digitalWrite(swiatloZolte2, LOW);
      digitalWrite(swiatloZielone1, HIGH);
      digitalWrite(swiatloCzerwone2, HIGH);
      delay(5000);

      digitalWrite(swiatloZielone1, LOW);
      digitalWrite(swiatloZolte1, HIGH);
      digitalWrite(swiatloZolte2, HIGH);
      delay(2000);

      digitalWrite(swiatloZolte1, LOW);
      digitalWrite(swiatloCzerwone1, HIGH);
      digitalWrite(swiatloCzerwone2, LOW);
      digitalWrite(swiatloZolte2, LOW);
      digitalWrite(swiatloZielone2, HIGH);
      delay(5000);
      digitalWrite(swiatloZielone2, LOW);
}
```

Ostatnia modyfikacja sygnalizacji drogowej jaką wykonamy to sygnalizacja na przejściu dla pieszych. Sekwencję zmian świateł na przejściu prezentuje poniższy rysunek. Pierwsze z lewej ustawienie świateł potrójnych odpowiada pierwszemu z lewej ustawieniu świateł podwójnych.



Rysunek 17. Sekwencja zmian świateł na przejściu dla pieszych.

W tym układzie usuń jedną z żółtych diod wraz z rezystorem. Układ rozbuduj o rezystor 1000 Ω i przycisk, którym będziemy uruchamiać sekwencję zmiany świateł.



Rysunek 18. Układ sygnalizacji świetlnej z przyciskiem.

Program sterujący jest kombinacją poprzedniego programu z programem wykorzystującym przycisk z ostatnich zajęć.

```
int swiatloCzerwoneSamochody = 12;
int swiatloZolteSamochody = 11;
int swiatloZieloneSamochody = 10;
int swiatloCzerwonePiesi = 4;
int swiatloZielonePiesi = 2;
int przycisk = 7;
int czyWcisnietoPrzycisk  = LOW;
```

```
void setup(){
      pinMode(swiatloCzerwoneSamochody, OUTPUT);
      pinMode(swiatloZolteSamochody, OUTPUT);
      pinMode(swiatloZieloneSamochody, OUTPUT);
      pinMode(swiatloCzerwonePiesi, OUTPUT);
      pinMode(swiatloZielonePiesi, OUTPUT);
      pinMode(przycisk, OUTPUT);
      digitalWrite(swiatloZieloneSamochody, HIGH);
      digitalWrite(swiatloCzerwonePiesi, HIGH);
}

void loop(){
      czyWcisnietoPrzycisk  = digitalRead(przycisk);

      if (czyWcisnietoPrzycisk  == HIGH) {
            delay(2000);
            digitalWrite(swiatloZolteSamochody, HIGH);
            digitalWrite(swiatloZieloneSamochody, LOW);
            delay(2000);

            digitalWrite(swiatloCzerwoneSamochody, HIGH);
            digitalWrite(swiatloZolteSamochody, LOW);
            digitalWrite(swiatloCzerwonePiesi, LOW);
            digitalWrite(swiatloZielonePiesi, HIGH);
            delay(5000);

            digitalWrite(swiatloZolteSamochody, HIGH);
            for (int i = 0; i < 10; i = i + 1){
                  digitalWrite(swiatloZielonePiesi, HIGH);
                  delay(250);
                  digitalWrite(swiatloZielonePiesi, LOW);
                  delay(250);
            }
            digitalWrite(swiatloCzerwoneSamochody, LOW);
            digitalWrite(swiatloZolteSamochody, LOW);
            digitalWrite(swiatloZieloneSamochody, HIGH);
            digitalWrite(swiatloCzerwonePiesi, HIGH);
      }
}
```

Program rozpoczynamy od przypisania zmiennych do numerów złącz cyfrowych dla każdej diody i przycisku oraz zadeklarowania zmiennej "czyWcisnietoPrzycisk ".

W funkcji "setup()" podobnie jak w poprzednim programie i programie wykorzystującym przycisk ustawiamy, które z pinów mają być pinami wejściowymi, a które wyjściowymi. Chcemy, by po uruchomieniu programu na sygnalizacji świetlnej dla samochodów włączyło się światło zielone, a na sygnalizacji dla pieszych czerwone, dlatego na pinach odpowiedzialnych za te diody ustawimy stan wysoki.

W funkcji "loop()" podobnie jak w programie obsługującym przycisk najpierw sprawdzamy za pomocą polecenia „if” czy przycisk jest wciśnięty. Jeśli jest, to wykonywana jest sekwencja zmiany świateł.

Warto zwrócić uwagę na to, jak realizowane jest miganie zielonego światła dla pieszych. Dziesięciokrotnie, za pomocą polecenia „for”, co 250 ms zmieniany jest stan z wysokiego na niski dla zielonej diody (światła dla pieszych).

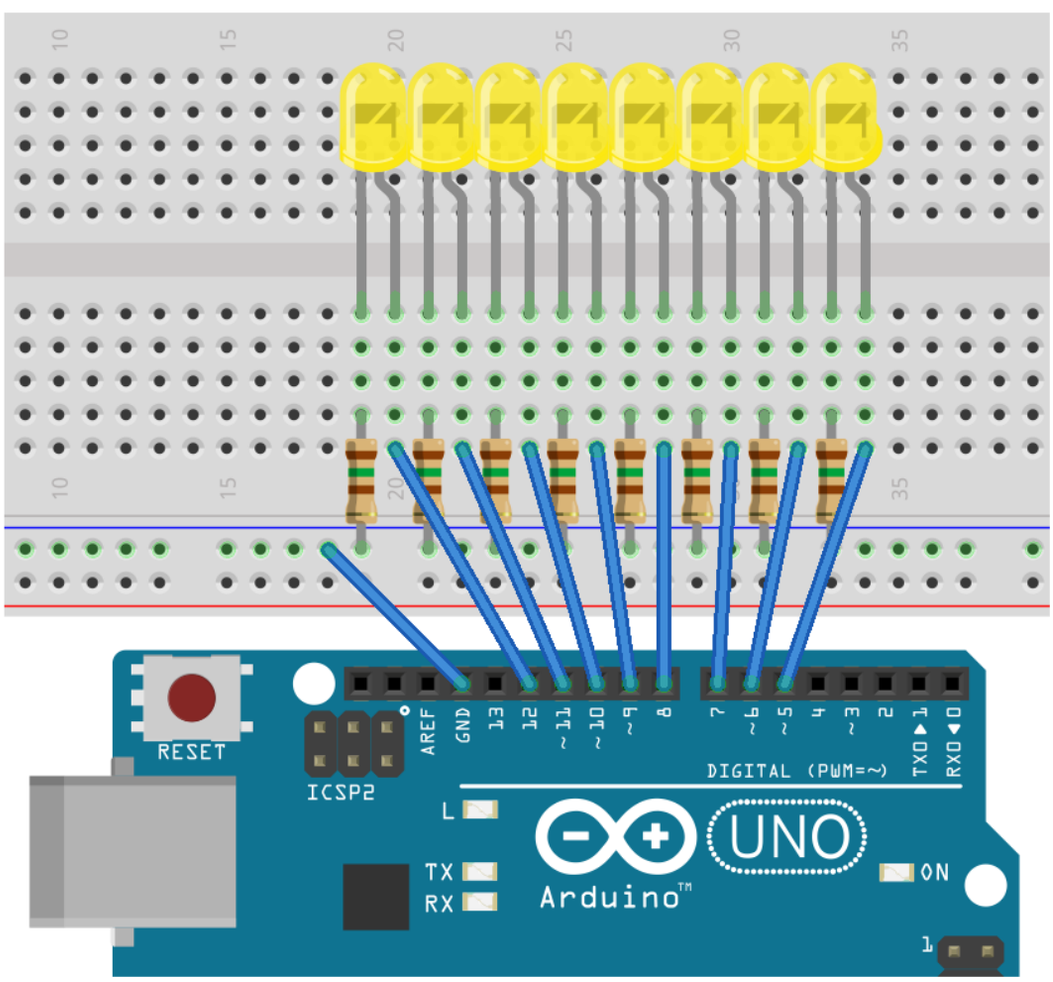
```
    for (int i = 0; i < 10; i = i + 1){
        digitalWrite(swiatloZielonePiesi, HIGH);
        delay(250);
        digitalWrite(swiatloZielonePiesi, LOW);
        delay(250);
    }
```

Zajęcia 4: Wędrująca dioda

Celem zajęć jest zmontowanie i zaprogramowanie układu z wieloma diodami, tak aby uzyskać efekt „wędrującej diody”. Potrzebujemy do tego:

1. Komputer,
2. Kabel USB,
3. Płytkę Arduino,
4. Płytkę stykowej,
5. 8 rezystorów ok. 150 Ω,
6. 8 diod LED, dowolnego koloru.
7. Przewody.

Układ jest bardzo podobny do tego, który obsługiwał sygnalizację drogową i powinien wyglądać jak na rysunku poniżej.



Rysunek 19. Układ "Wędrująca dioda".

Sterowanie diodami powinno odbywać się następująco:

1. włączana jest pierwsza od lewej dioda na 300 ms.
2. Wyłączana jest pierwsza dioda i druga od lewej włączana jest na 300 ms.
3. Wyłączana jest druga dioda i trzecia od lewej dioda włączana jest na 300 ms.
4. I tak dalej…

Program jest bardzo prosty i jest rozbudowaną wersją programu z pierwszych zajęć.

```
int dioda1 = 12;
int dioda2 = 11;
int dioda3 = 10;
int dioda4 = 9;
int dioda5 = 8;
int dioda6 = 7;
int dioda7 = 6;
int dioda8 = 5;
int czas = 300;

void setup() {
      pinMode(dioda1, OUTPUT);
      pinMode(dioda2, OUTPUT);
      pinMode(dioda3, OUTPUT);
      pinMode(dioda4, OUTPUT);
      pinMode(dioda5, OUTPUT);
      pinMode(dioda6, OUTPUT);
      pinMode(dioda7, OUTPUT);
      pinMode(dioda8, OUTPUT);
}
void loop() {
      digitalWrite(dioda1, HIGH);
      delay(czas);
      digitalWrite(dioda1, LOW);
      delay(czas);
      digitalWrite(dioda2, HIGH);
      delay(czas);
      digitalWrite(dioda2, LOW);
      delay(czas);
      digitalWrite(dioda3, HIGH);
      delay(czas);
      digitalWrite(dioda3, LOW);
      delay(czas);
      digitalWrite(dioda4, HIGH);
      delay(czas);
      digitalWrite(dioda4, LOW);
      delay(czas);
      digitalWrite(dioda5, HIGH);
      delay(czas);
      digitalWrite(dioda5, LOW);
```

```
      delay(czas);
      digitalWrite(dioda6, HIGH);
      delay(czas);
      digitalWrite(dioda6, LOW);
      delay(czas);
      digitalWrite(dioda7, HIGH);
      delay(czas);
      digitalWrite(dioda7, LOW);
      delay(czas);
      digitalWrite(dioda8, HIGH);
      delay(czas);
      digitalWrite(dioda8, LOW);
      delay(czas);
}
```

Port szeregowy

Ponieważ sama „wędrująca dioda” jest mało przydatna nauczymy się jak sterować prędkością jej „wędrówki” za pomocą portu szeregowego (ang. *serial port*). Zwykle wykorzystuje się go do przesyłania i odbierania informacji z i do komputera. Zanim przejdziemy do sterowania diodami, przeanalizujemy dwa proste programy wprowadzające w zagadnienie obsługi portu szeregowego.

```
int kolejnaLiczba = 0;

void setup() {
      Serial.begin(9600);
      Serial.println("Witaj w programie odliczającym. A teraz kolejno
odlicz!");
}

void loop() {
      Serial.println(kolejnaLiczba);
      kolejnaLiczba = kolejnaLiczba + 1;
      delay(1000);
}
```

Zmienna „kolejnaLiczba” to zmienna, w której zapisujemy liczbę wykonanych do tej pory iteracji. W funkcji „setup()” polecenie „Serial.begin(9600)” służy do otwarcia portu szeregowego i ustanawia prędkość przesyłu danych. Kolejna funkcja „Serial.println()” to funkcja umożliwiająca wyświetlenie informacji na ekranie komputera. W naszym przypadku jest nią „Witaj w programie odliczającym. A teraz kolejno odlicz!".

W funkcji „loop()” wyświetlamy daną liczbę iteracji <Serial.println(kolejnaLiczba)> i zwiększamy wartość zmiennej „kolejnaLiczba” o 1 <kolejnaLiczba = kolejnaLiczba + 1>. Na końcu czekamy jedną sekundę.

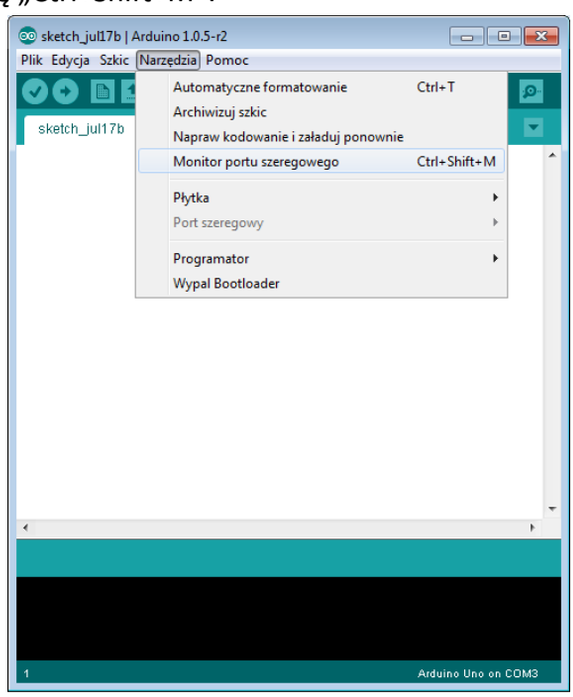
UWAGA! Pamiętaj, że jeśli chcesz na ekranie monitora wyświetlić dany tekst musisz umieścić go w cudzysłowie.

```
Serial.println("Witaj w programie odliczającym. A teraz kolejno odlicz!");
```

Jeśli chcesz wyświetlić wartość zmiennej, nie używaj cudzysłowu.

```
Serial.println(kolejnaLiczba);
```

Aby zobaczyć efekt wybierz „Narzędzia”, a następnie „Monitor portu szeregowego” lub wciśnij na klawiaturze kombinację „Ctrl+Shift+M”.



Rysunek 20. Monitor portu szeregowego.

W drugim przykładzie pokażemy jak wyświetlić na ekranie liczby wpisane przez nas przy użyciu klawiatury i w jaki sposób port szeregowy wykorzystać do sterowania częstotliwością migania diody. Wykorzystamy zmontowany wcześniej układ, jednak użyjemy tylko jednej z diod.

```
int predkosc = 10;
int dioda = 12;

void setup()
{
      Serial.begin(9600);
      pinMode(dioda, OUTPUT);
}
void loop() {
       if (Serial.available()){
             predkosc = Serial.parseInt();
         }
         Serial.println(predkosc);
         digitalWrite(dioda, HIGH);
         delay(predkosc *10);
         digitalWrite(dioda, LOW);
         delay(predkosc *10);
}
```

Program zaczynamy od ustawienia początkowej częstotliwości migania diody oraz złącza, które będziemy wykorzystywać. W funkcji „setup()", podobnie jak poprzednio, otwieramy port szeregowy i ustawiamy złącze diody na „OUTPUT". Polecenie „if (Serial.available())" w tym przykładzie sprawdza, czy dane zostały już wprowadzone (wpisane). Jeśli tak to obierane dane zamienia na liczbę całkowitą „Serial.parseInt()" i zapisuje jako nową częstotliwość w zmiennej „predkosc". Dzięki linii „Serial.println(predkosc)" na ekranie wyświetlany jest aktualny czas jednego włączenia diody. Następnie dioda miga w zależności od wartości zmiennej „predkosc".

Łącząc pierwszy i trzeci program z tych zajęć otrzymamy kod sterujący prędkością przemieszczania się „wędrującej diody".

```
int dioda1 = 5;
int dioda2 = 6;
int dioda3 = 7;
int dioda4 = 8;
int dioda5 = 9;
int dioda6 = 10;
int dioda7 = 11;
int dioda8 = 12;
int predkosc = 10;

void setup() {
      Serial.begin(9600);
      pinMode(dioda1, OUTPUT);
      pinMode(dioda2, OUTPUT);
      pinMode(dioda3, OUTPUT);
      pinMode(dioda4, OUTPUT);
      pinMode(dioda5, OUTPUT);
      pinMode(dioda6, OUTPUT);
      pinMode(dioda7, OUTPUT);
      pinMode(dioda8, OUTPUT);
}
void loop() {
      if (Serial.available()){
            predkosc = Serial.parseInt();
            }
      Serial.println(predkosc);

      digitalWrite(dioda1, HIGH);
      delay(predkosc *10);
      digitalWrite(dioda1, LOW);
      delay(predkosc *10);
      digitalWrite(dioda2, HIGH);
      delay(predkosc *10);
      digitalWrite(dioda2, LOW);
      delay(predkosc *10);
      digitalWrite(dioda3, HIGH);
      delay(predkosc *10);
      digitalWrite(dioda3, LOW);
```

```
      delay(predkosc *10);
      digitalWrite(dioda4, HIGH);
      delay(predkosc *10);
      digitalWrite(dioda4, LOW);
      delay(predkosc *10);
      digitalWrite(dioda5, HIGH);
      delay(predkosc *10);
      digitalWrite(dioda5, LOW);
      delay(predkosc *10);
      digitalWrite(dioda6, HIGH);
      delay(predkosc *10);
      digitalWrite(dioda6, LOW);
      delay(predkosc *10);
      digitalWrite(dioda7, HIGH);
      delay(predkosc *10);
      digitalWrite(dioda7, LOW);
      delay(predkosc *10);
      digitalWrite(dioda8, HIGH);
      delay(predkosc *10);
      digitalWrite(dioda8, LOW);
      delay(predkosc *10);
}
```

Program jest długi i można go znacznie skrócić wykorzystując pętlę „for”. Poniżej zawartość funkcji „loop()” po uproszczeniu.

```
void loop(){
     if (Serial.available()){
           predkosc = Serial.parseInt();
     }
     Serial.println(predkosc);

     for (int i = 5; i <= 12;i = i + 1){
           digitalWrite(i, HIGH);
           delay(predkosc *10);
           digitalWrite(i, LOW);
           delay(predkosc *10);
     }
}
```

W „digitalWrite(…)” zmienną odpowiadającą za kolejną diodę (dioda1,dioda2…) zamieniono zmienną „i”. Jeśli przyjrzymy się dokładnie zawartości nawiasu „(int i=5;i<=12;i=i+1)”, zauważymy, że zmienna „i” zmienia swoją wartość w zakresie od 5 do 12. Przypomnijmy, że złącza wykorzystane do podłączenia diod, także zawierając się w zakresie do 5 do 12. Tak więc pętla „for” w każdej kolejnej iteracji włącza i wyłącza kolejną diodę.

Problemem jest jednak to, że diody „poruszają się” tylko w jednym kierunku. Aby uzyskać efekt „odbijania”, funkcja „loop()” powinna wyglądać jak poniżej.

```
void loop(){
            if (Serial.available()){
                  predkosc = Serial.parseInt();
            }
             Serial.println(predkosc);

      for (int i = 5; i<= 12;i = i + 1){
            digitalWrite(i, HIGH);
            delay(predkosc *10);
            digitalWrite(i, LOW);
            delay(predkosc *10);
      }
      for (int i = 12;i >= 5; i = i-1){
            digitalWrite(i, HIGH);
            delay(predkosc *10);
            digitalWrite(i, LOW);
            delay(predkosc *10);
      }
}
```

W pierwszej pętli „for” włączane są diody od 5 do 12, natomiast w drugiej odwrotnie, od 12 do 5.

Zajęcia 5: Fotorezystor

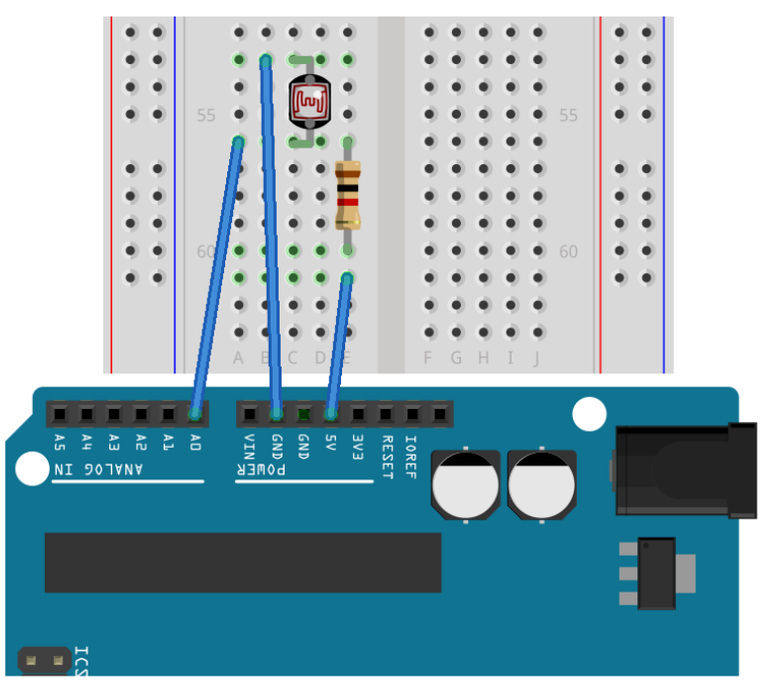
Fotorezystor jest specjalnym typem rezystora, którego wartość rezystancji zależy od oświetlenia. Można przyjąć, że nieoświetlony ma bardzo dużą wartość rezystancji (niemal nieskończoną), która zacznie spadać wraz ze zwiększeniem ilości padającego światła.

Rysunek 21. Fotorezystor.

Pierwszy układ z wykorzystaniem fotorezystora będzie sprawdzał natężenie światła w otoczeniu i przesyłał za pośrednictwem portu szeregowego zmierzoną wartość. Potrzebujemy do tego:

1. Komputer,
2. Kabel USB,
3. Płytkę Arduino,
4. Płytkę stykową,
5. 1 rezystor ok. 1000 Ω,
6. Fotorezystor.

Połącz elementy jak na rysunku poniżej.



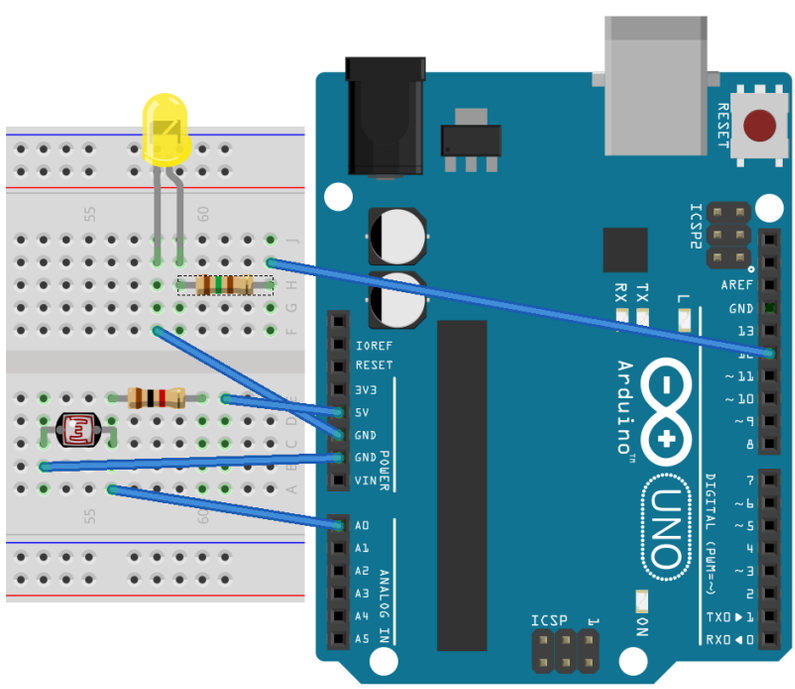
Rysunek 22. Układ z fotorezystorem.

```
int oswietlenie = 0;

void setup() {
      Serial.begin(9600);
}

void loop() {
     oswietlenie = analogRead(A0);
     Serial.println(oswietlenie);
     delay(1000);
}
```

W pierwszej kolejności inicjalizujemy zmienną „oświetlenie”. W funkcji „setup()” uruchamiany port szeregowy. W funkcji „loop()” program odczytuje wartość z portu analogowego A0 <oswietlenie = analogRead(A0)> wyświetla ją na ekranie <Serial.println(oswietlenie)> i czeka 1000 ms z następnym pobraniem wartości z portu analogowego.

Dotychczas wykorzystywaliśmy tylko złącza cyfrowe. W tym programie po raz pierwszy skorzystaliśmy ze złącza analogowego. Gdy odczytujemy wartość z każdego z tych złącz, główna różnica polega na tym, co odbieramy. W przypadku złącz cyfrowych są to 0 i 1, w przypadku złącz analogowych są to wartości od 0 do 1023. Uruchom monitor portu szeregowego i sprawdź co odbierasz z poru A0. Od czego zależna jest odbierana wartość? Im wyższa wartość tym jest ciemniej, im niższa tym jaśniej.

Do czego można wykorzystać taki układ? Na przykład do automatycznego włączania oświetlenia w pomieszczeniu gdy jest ciemno. Do rozbudowy układu potrzebujemy diody i rezystora 150 Ω.



Rysunek 23. Rozbudowany układ z fotorezystorem.

Proponowana rozbudowa programu:

```
int oswietlenie = 0;
int diode = 12;

void setup(){
      Serial.begin(9600);
      pinMode(dioda, OUTPUT);
}

void loop(){
      int oswietlenie = analogRead(A0);
      if (oswietlenie > 511){
            digitalWrite(dioda, HIGH);
      }
      else{
            digitalWrite(dioda, LOW);
      }
      Serial.println(oswietlenie);
      delay(1000);
}
```

W powyższym programie została dodana obsługa diody. Do 12 łącza podłączono diodę <int dioda=12> i tryb tego złączą ustawiono na OUTPUT <pinMode(dioda,OUTPUT)>.

Najważniejsza modyfikacja występuje w funkcji „loop()". Dopisaliśmy w niej instrukcję „if", która sprawdza czy w pomieszczeniu jest ciemno (wartości zmiennej „oswietlenie" zawiera się od 512 do 1023, dioda jest włączana), czy w pomieszczeniu jest jasno (wartości zmiennej „oswietlenie" zawiera się od 0 do 511, dioda wyłączona). Oczywiście warunek „oswietlenie >511" możesz zmieniać dowolnie, w zależności od potrzeb.

Instrukcja warunkowa „If, else if, else"

Posłużymy się poniższym przykładem, aby przedstawić rozbudowaną wersję instrukcji warunkowej.

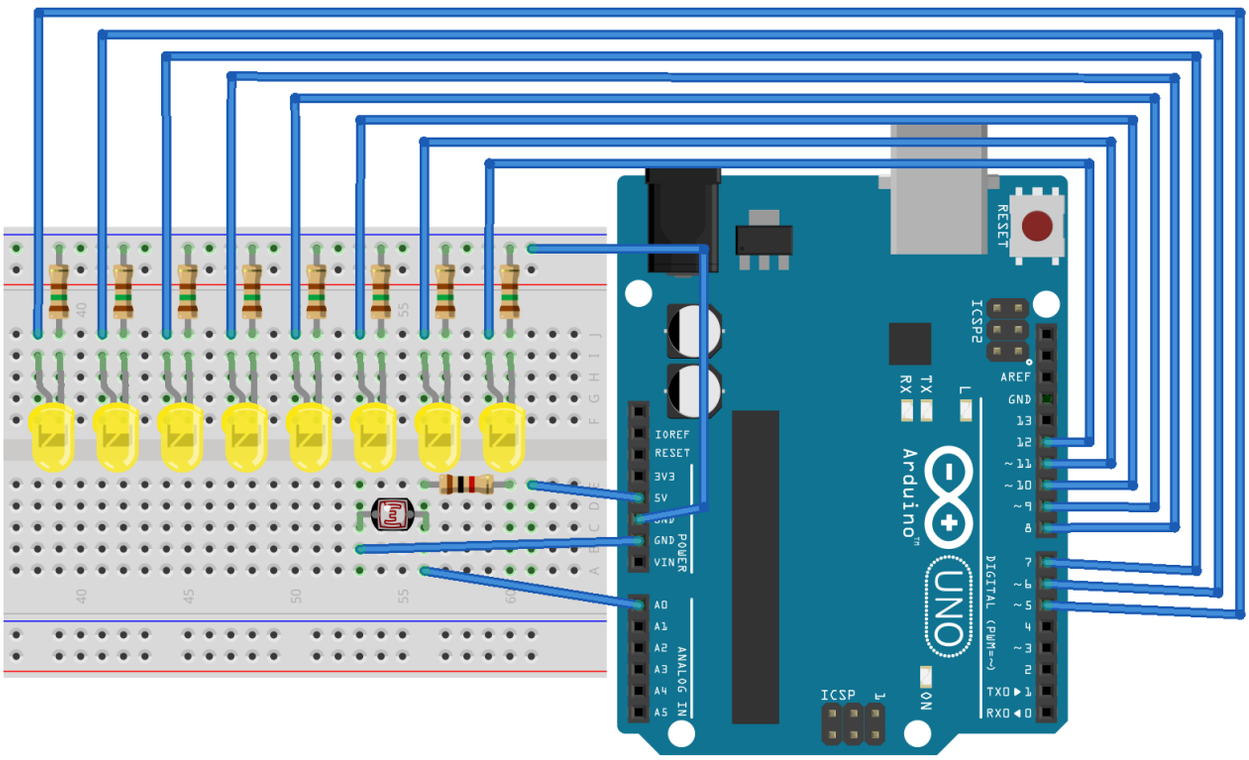
```
int oswietlenie = 0;
int dioda8 = 12;
int dioda7 = 11;
int dioda6 = 10;
int dioda5 = 9;
int dioda4 = 8;
int dioda3 = 7;
int dioda2 = 6;
int dioda1 = 5;

void setup(){
      Serial.begin(9600);
      pinMode(dioda1, OUTPUT);
      pinMode(dioda2, OUTPUT);
```

```
      pinMode(dioda3, OUTPUT);
      pinMode(dioda4, OUTPUT);
      pinMode(dioda5, OUTPUT);
      pinMode(dioda6, OUTPUT);
      pinMode(dioda7, OUTPUT);
      pinMode(dioda8, OUTPUT);
}

void loop() {
      int oswietlenie = analogRead(A0);

       if (oswietlenie > 895)
      {
            digitalWrite(dioda1, HIGH);
            digitalWrite(dioda2, HIGH);
            digitalWrite(dioda3, HIGH);
            digitalWrite(dioda4, HIGH);
            digitalWrite(dioda5, HIGH);
            digitalWrite(dioda6, HIGH);
            digitalWrite(dioda7, HIGH);
            digitalWrite(dioda8, HIGH);
      }
      else{
            digitalWrite(dioda1, LOW);
            digitalWrite(dioda2, LOW);
            digitalWrite(dioda3, LOW);
            digitalWrite(dioda4, LOW);
            digitalWrite(dioda5, LOW);
            digitalWrite(dioda6, LOW);
            digitalWrite(dioda7, LOW);
            digitalWrite(dioda8, LOW);

      }
      if (oswietlenie > 767)
      {
            digitalWrite(dioda1, HIGH);
            digitalWrite(dioda2, HIGH);
            digitalWrite(dioda3, HIGH);
            digitalWrite(dioda4, HIGH);
            digitalWrite(dioda5, HIGH);
            digitalWrite(dioda6, HIGH);
            digitalWrite(dioda7, HIGH);
      }
      else{
            digitalWrite(dioda1, LOW);
            digitalWrite(dioda2, LOW);
            digitalWrite(dioda3, LOW);
            digitalWrite(dioda4, LOW);
            digitalWrite(dioda5, LOW);
            digitalWrite(dioda6, LOW);
```

```
            digitalWrite(dioda7, LOW);
      }

      if (oswietlenie > 639)
      {
            digitalWrite(dioda1, HIGH);
            digitalWrite(dioda2, HIGH);
            digitalWrite(dioda3, HIGH);
            digitalWrite(dioda4, HIGH);
            digitalWrite(dioda5, HIGH);
            digitalWrite(dioda6, HIGH);
      }
      else{
            digitalWrite(dioda1, LOW);
            digitalWrite(dioda2, LOW);
            digitalWrite(dioda3, LOW);
            digitalWrite(dioda4, LOW);
            digitalWrite(dioda5, LOW);
            digitalWrite(dioda6, LOW);
      }

      if (oswietlenie > 511)
      {
            digitalWrite(dioda1, HIGH);
            digitalWrite(dioda2, HIGH);
            digitalWrite(dioda3, HIGH);
            digitalWrite(dioda4, HIGH);
            digitalWrite(dioda5, HIGH);
      }
      else{
            digitalWrite(dioda1, LOW);
            digitalWrite(dioda2, LOW);
            digitalWrite(dioda3, LOW);
            digitalWrite(dioda4, LOW);
            digitalWrite(dioda5, LOW);
      }

      if (oswietlenie > 383)
      {
            digitalWrite(dioda1, HIGH);
            digitalWrite(dioda2, HIGH);
            digitalWrite(dioda3, HIGH);
            digitalWrite(dioda4, HIGH);
      }
      else{
            digitalWrite(dioda1, LOW);
            digitalWrite(dioda2, LOW);
            digitalWrite(dioda3, LOW);
            digitalWrite(dioda4, LOW);
      }
```

```
      if (oswietlenie > 255)
      {
            digitalWrite(dioda1, HIGH);
            digitalWrite(dioda2, HIGH);
            digitalWrite(dioda3, HIGH);
      }
      else{
            digitalWrite(dioda1, LOW);
            digitalWrite(dioda2, LOW);
            digitalWrite(dioda3, LOW);
      }

      if (oswietlenie > 127)
      {
            digitalWrite(dioda1, HIGH);
            digitalWrite(dioda2, HIGH);
      }
      else{
            digitalWrite(dioda1, LOW);
            digitalWrite(dioda2, LOW);
      }

      if (oswietlenie < 127)
      {
            digitalWrite(dioda1, HIGH);
      }
      else{
            digitalWrite(dioda1, LOW);
      }

      Serial.println(oswietlenie);
      delay(200);
}
```

Przygotuj 7 diod oraz 7 rezystorów 150 Ω i układ rozbuduj jak na rysunku poniżej.



Rysunek 24. Diodowy wskaźnik naświetlenia.

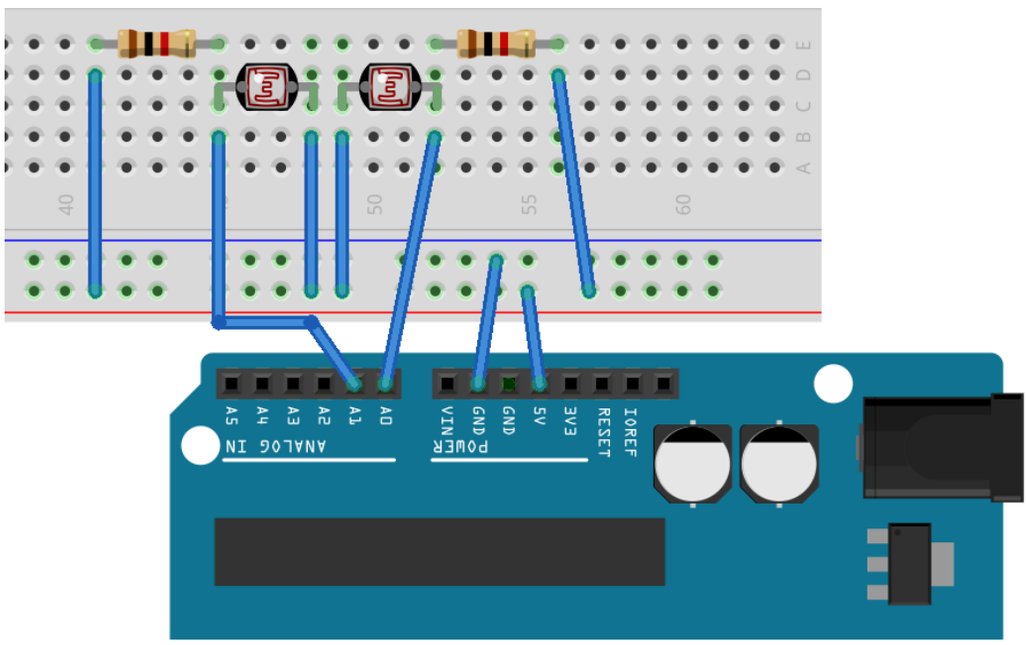
Powyższy kod jest bardzo podobny do tego z poprzedniego programu. Różnica polega na zadeklarowaniu większej ilości diod oraz wykorzystania instrukcji „else if" (pol. *inaczej, inaczej wystarcza*). Instrukcja ta wykorzystywana jest gdy sprawdzić musimy więcej niż jeden warunek. W zależności od tego, który z warunków jest spełniony, włączana jest odpowiednia liczba diod.

Jak zostały wyznaczone wartości sprawdzane w warunkach? Zakres otrzymywanych z fotorezystora wartości zawiera się między 0 a 1023, jest to łącznie 1024 liczb. Wartość ta, podzielona przez 8 (ponieważ tyle jest diod), daje 128. Wartości w pierwszym warunku uzyskano w następujący sposób: 1023-128=895. W drugim warunku: 895-128=767. W trzecim warunku 767-128=639 itd.

Jak powinien wyglądać program, w którym im jaśniej w otoczeniu fotorezystora tym więcej zapala się diod?

Kierunek padania światła

Skąd pochodzi źródło światła? Aby to sprawdzić wystarczy układ rozbudować jak na rysunku poniżej.



Rysunek 25. Układ z dwoma fotorezystorami.

```
int oswietlenieLewaStrona = 0;
int oswietleniePrawaStrona = 0;

void setup(){
      Serial.begin(9600);
}

void loop(){
      oswietlenieLewaStrona = analogRead(A0);
      oswietleniePrawaStrona = analogRead(A1);
      Serial.print(oswietlenieLewaStrona);
      Serial.print(" ");
      Serial.println(oswietleniePrawaStrona);
      delay(1000);
}
```

Kod ten jest rozbudowanym, pierwszym programem z tych zajęć. W linii „int oswietleniePrawaStrona=0” pod zmienną zapisujemy początkową wartość dla drugiego fotorezystora równą zero. W linii „oswietleniePrawaStrona = analogRead(A1)” odczytujemy wartość z drugiego złącza analogowego.

Do tej pory, aby wyświetlić coś w okienku monitora portu szeregowego używaliśmy funkcji „Serial.println()”. Ostatnie litery „ln” oznaczają, że po wyświetleniu danej informacji, następna informacja ma być wyświetlona w następnej linii. Użycie funkcji „Serial.print()”

powoduje, że następna informacja będzie w tej samej linijce. Użyta linia <Serial.print(" ")> służy do wstawieniu odstępu pomiędzy wartością pierwszą i drugą.

Informacja jest wyświetlana następująco: pierwsza jest wartość zmiennej „oswietlenieLewaStrona", w tej samej linii wstawiona jest spacja i na końcu tej linii wyświetlana jest wartość zmiennej „oswietleniePrawaStrona". Najłatwiej będzie to zrozumieć, jeśli pozmieniasz samodzielnie „Serial.print()" na „Serial.println()" i odwrotnie.

Same wartości liczbowe wydają się mało intuicyjne. Dlatego zastąpmy je tekstem informującym o kierunku źródła światła. Wykorzystamy do tego instrukcję warunkową „if" i jej rozbudowaną wersję poznaną na tych zajęciach, która sprawdza, która z otrzymanych wartości jest większa. Większa wartość wyznacza kierunek źródła światła.

```
int oswietlenieLewaStrona = 0;
int oswietleniePrawaStrona = 0;

void setup() {
      Serial.begin(9600);
}

void loop() {
      oswietlenieLewaStrona = analogRead(A0);
      oswietlenie2 = analogRead(A1);

      if(oswietlenieLewaStrona > oswietleniePrawaStrona){
            Serial.println("zrodlo swiatla z lewej");
      }
      else if(oswietlenieLewaStrona < oswietleniePrawaStrona){
            Serial.println("zrodlo swiatla z prawej");
      }
      else{
            Serial.println("zrodlo swiatla centralnie nad
      fotorezysotrami");
      }
      delay(1000);
}
```

Zajęcia 6: Alarm

Potencjometr i odczyt z portu analogowego

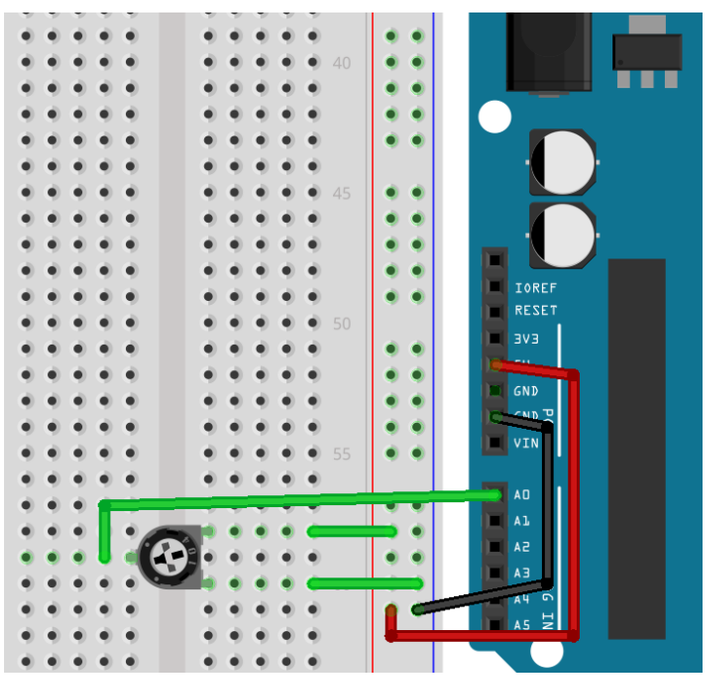
Potencjometr jest, obok fotorezystora, kolejnym, specjalnym typem rezystora. Potencjometr jest rezystorem o zmiennej (regulowanej) wartości rezystancji. Regulacja odbywa się poprzez manipulowanie pokrętłem na obudowie w zakresie od zera ohmów do określonej wartości maksymalnej.

Rysunek 26. Potencjometr.

Aby wykonać pierwszy układ z potencjometrem, potrzebujemy:

1. Komputer,
2. Kabel USB,
3. Płytkę Arduino,
4. Płytkę stykową,
5. Potencjometr.

Układ połącz jak na poniższym rysunku.



Rysunek 27. Pierwszy układ z potencjometrem.

Pod zmienną „potencjometr” zapisany jest numer złącza wykorzystanego do odczytu wartości analogowej. Pod zmienną „wartoscPotencjometr” zapisaliśmy początkową wartość (równą 0), a później zapisywać będziemy wartości pobrane ze złącza analogowego za pomocą niewykorzystywanej do tej pory funkcji „analogRead()”. Uruchom „Monitor portu szeregowego” i zmieniając nastawy potencjometru obserwuj jak zmieniają się wartości w oknie monitora.

```
int potencjometr = 0;
int wartoscPotencjometr = 0;

void setup(){
      Serial.begin(9600);
}

void loop(){
      wartoscPotencjometr = analogRead(potencjometr);
      Serial.println(wartoscPotencjometr);
      delay(100);
 }
```

Buzzer

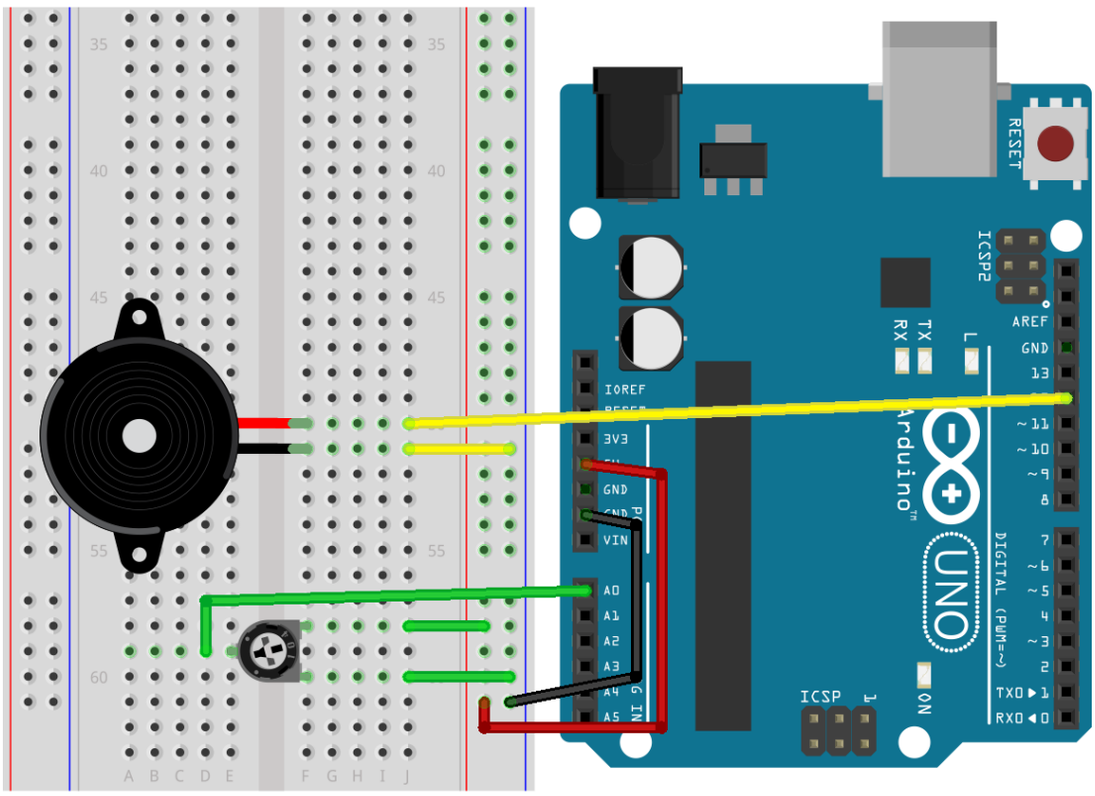
Inną, czasem używaną nazwą tego urządzenia jest brzęczyk. Jego cechą charakterystyczną jest to, że po podłączeniu do niego źródła napięcia wyda on dźwięk (pisk). Dźwięk jest generowany za pomocą tzw. zjawiska piezoelektrycznego, polegającego na pojawieniu się naprężeń mechanicznych (ruchu) na membranie, pod wpływem przyłożonego napięcia.

Podobnie jak w przypadku diody LED, buzzer jest elementem biegunowym. Nóżka dłuższa jest nóżką o biegunowości dodatniej, nóżka krótsza - nóżką o biegunowości ujemnej.

Rysunek 28. Buzzer.

Rozbuduj poprzedni układ o buzzer, tak jak na rysunku poniżej. W przypadku sterowania buzzerem i diodą program wygląda tak samo. Poniżej przedstawiono kod programu, w którym wykorzystujemy nastawy potencjometru do ustawienia długości czasu przez jaki buzzer ma wydobywać z siebie dźwięk.

Poprzedni kod rozbudowaliśmy o obsługę buzzera (wykorzystujemy do tego złącze 12). Włączanie i wyłączanie buzzera odbywało się identycznie jak Włączanie i wyłączanie diody na pierwszych zajęciach poprzez ustawienie stanu wysokiego lub niskiego na złączu 12.

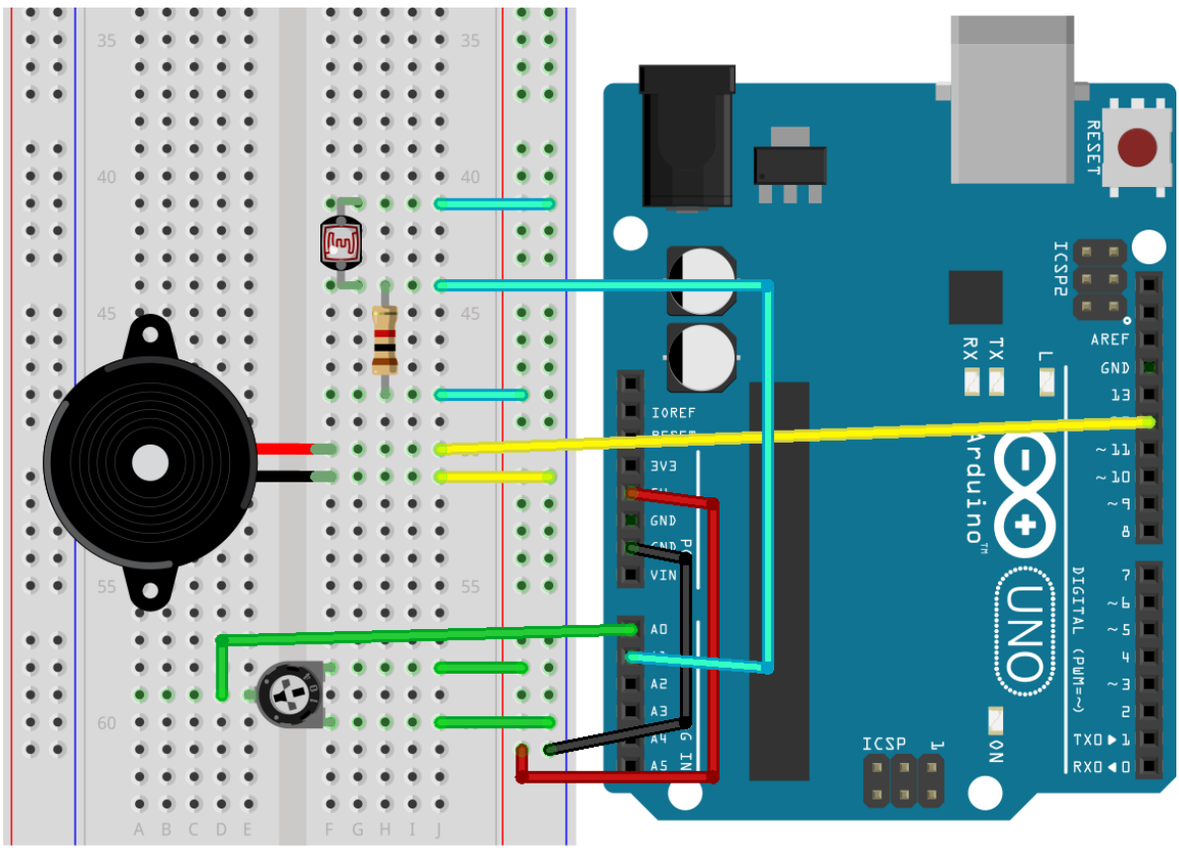


Rysunek 29. Układ z buzzerem i potencjometrem.

```
int potencjometr = 0;
int wartoscPotencjometr = 0;
int buzzer = 12

void setup() {
      Serial.begin(9600);
      pinMode(buzzer, OUTPUT);
}

void loop() {
      wartoscPotencjometr =
analogRead(potencjometr);
      Serial.println(wartoscPotencjometr);
      digitalWrite(buzzer, HIGH);
      delay(wartoscPotencjometr);
      digitalWrite(buzzer, LOW);
      delay(wartoscPotencjometr);
}
```

Alarm

Rozbudowując powyższy układ o fotorezystor znany z poprzednich zajęć możemy skonstruować prosty alarm chroniący np. szufladę lub szafkę przez nieproszonymi gośćmi. Dołącz fotorezystor i rezystor 1000 Ω jak na poniższym rysunku.



Rysunek 30. Układ „Alarm”.

Programowanie alarmu będzie składało się z kilku etapów. Pierwszym będzie napisanie programu, który będzie odczytywał wartości z dwóch pinów analogowych. Wartość odczytana ze złącza połączonego z potencjometrem służy, podobnie jak w poprzednim programie, do ustawienia długości impulsu dźwiękowego.

Poprzedni program został rozbudowany jedynie o obsługę fotorezystora „int fotorezystor = A0” i „int wartoscFotorezystor = 0” oraz o kod pozwalający na wyświetlanie obu odczytanych wartości analogowych „Serial.print(wartoscPotencjometr); Serial.print(" "); Serial.println(wartoscFotorezystor);”

```
int potencjometr = 0;
int fotorezystor = A0;
int wartoscPotencjometr = 0;
int wartoscFotorezystor = 0;
int buzzer=12;

void setup() {
     Serial.begin(9600);
```

```
      pinMode(buzzer, OUTPUT);
}

void loop(){
      wartoscPotencjometr = analogRead(potencjometr);
      wartoscFotorezystor = analogRead(fotorezystor);
      Serial.print(wartoscPotencjometr);
      Serial.print(" ");
      Serial.println(wartoscFotorezystor);
      digitalWrite(buzzer, HIGH);
      delay(wartoscPotencjometr);
      digitalWrite(buzzer, LOW);
      delay(wartoscPotencjometr);
}
```

Budowany przez nas alarm ma być umieszony w szafce bądź w szufladzie. Jeśli są one zamknięte, w środku jest oczywiście ciemno. Dlatego w naszym programie powinniśmy umieścić warunek uruchomienia alarmu, gdy wykryje on, że jest zbyt jasno (ktoś otworzył szafkę).

W funkcji „loop()" umieścimy instrukcję „if" sprawdzającą ilość światła. Jeśli jest go więcej niż założony próg (200, wartość ta może być zupełnie inna, dopasowana do sytuacji), alarm zostanie uruchomiony. Poniższy kod to jedynie pętla „loop()", reszta pozostaje bez zmian!

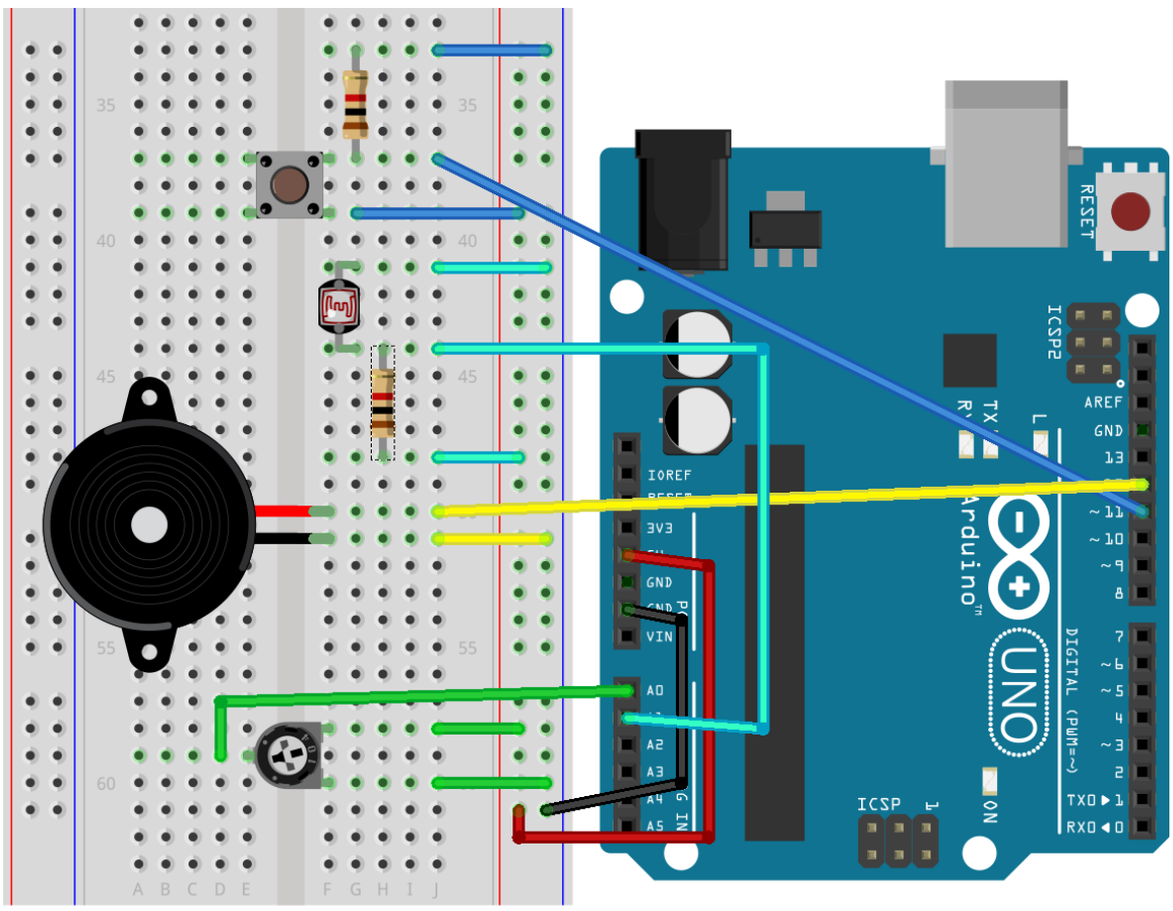
```
void loop(){
      wartoscPotencjometr = analogRead(potencjometr);
      wartoscFotorezystor = analogRead(fotorezystor);

      Serial.println(wartoscFotorezystor);

      if (wartoscFotorezystor < 200){
            digitalWrite(buzzer, HIGH);
            delay(wartoscPotencjometr);
            digitalWrite(buzzer, LOW);
            delay(wartoscPotencjometr);
      }
}
```

Jeśli chcesz zmienić wartość progu zadziałania alarmu, wykorzystaj „Monitor portu szeregowego", aby ustalić optymalną wartość.

Czy alarm działa w sposób zadowalający? Wyłącza się zaraz po tym, gdy zniknie źródło światła. Alarmy, które spotykamy w sklepach zwykle musi wyłączyć osoba do tego upoważniona za pomocą specjalnego przycisku lub kodu.

Alarm z wyłącznikiem i pętla „while”

Rozbudujemy teraz nasz alarm o przycisk wyłączający. Do układu dołącz przycisk i rezystor 1000 Ω, jak na rysunku poniżej.



Rysunek 31. Ostateczna wersja układu „Alarm”.

Kod programu nieco zmodyfikujemy. Najważniejsza zmiana polega na zamieszczeniu w nim pętli „while” (pol. *dopóki*). Pętla służy do wykonywania czynności do czasu, aż zostanie spełniony warunek w nawiasie. Najłatwiej przedstawić to na przykładzie z życia wziętym. Przykład ten dotyczy wbijania gwoździa. Jak za pomocą pętli „while” przedstawić wbijanie gwoździa?

```
while (gwozdz_nie_jest_wbity){
     uderz_mlotkiem_w_gwozdz();
}
```

Warunkiem, który jest sprawdzany jest „gwozdz_nie_jest_wbity” Jeśli jest spełniony to wykonywana jest jednokrotnie funkcja „uderz_mlotkiem_w_gwozdz()” a następnie znowu sprawdzany jest warunek. Operacje są wykonywane do czasu, aż warunek zostanie spełniony, czyli gwóźdź zostanie wbity.

W naszym programie warunkiem, który jest sprawdzany jest to, czy przycisk został wciśnięty <digitalRead(przycisk) == LOW)>. Jeśli nie jest - odzywa się alarm, jeśli jest - alarm jest wyłączany.

```
int potencjometr = 0;
int fotorezystor = 1;
int wartoscPotencjometr = 0;
int wartoscFotorezystor = 0;
int buzzer = 12;
int przycisk = 11;

void setup() {
      pinMode(buzzer, OUTPUT);
      pinMode(przycisk, OUTPUT);
}

void loop() {
      wartoscPotencjometr = analogRead(potencjometr);
      wartoscFotorezystor = analogRead(fotorezystor);

      if (wartoscFotorezystor < 200){
            while(digitalRead(przycisk) == LOW){
                  digitalWrite(buzzer, HIGH);
                  delay(wartoscPotencjometr);
                  digitalWrite(buzzer, LOW);
                  delay(wartoscPotencjometr);
            }
      }
}
```

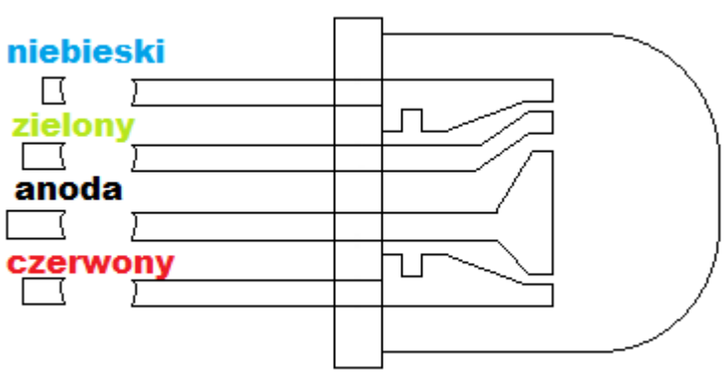
Zajęcia 7: Dioda RGB

Dioda RGB

Dioda wielokolorowa RGB (ang. *red, green, blue* - czerwony, zielony, niebieski) to dioda, która dzięki swej budowie może generować wyżej wymienione kolory, a także mieszać je dowolnie między sobą, co pozwala na uzyskanie praktycznie dowolnej barwy.

Rysunek 32. Z prawej strony na dole dioda RGB.

W odróżnieniu od zwykłej diody, dioda RGB ma aż 4 nóżki. Ta, którą będziemy wykorzystywać zbudowana jest z trzech katod i jednej, wspólnej anody.

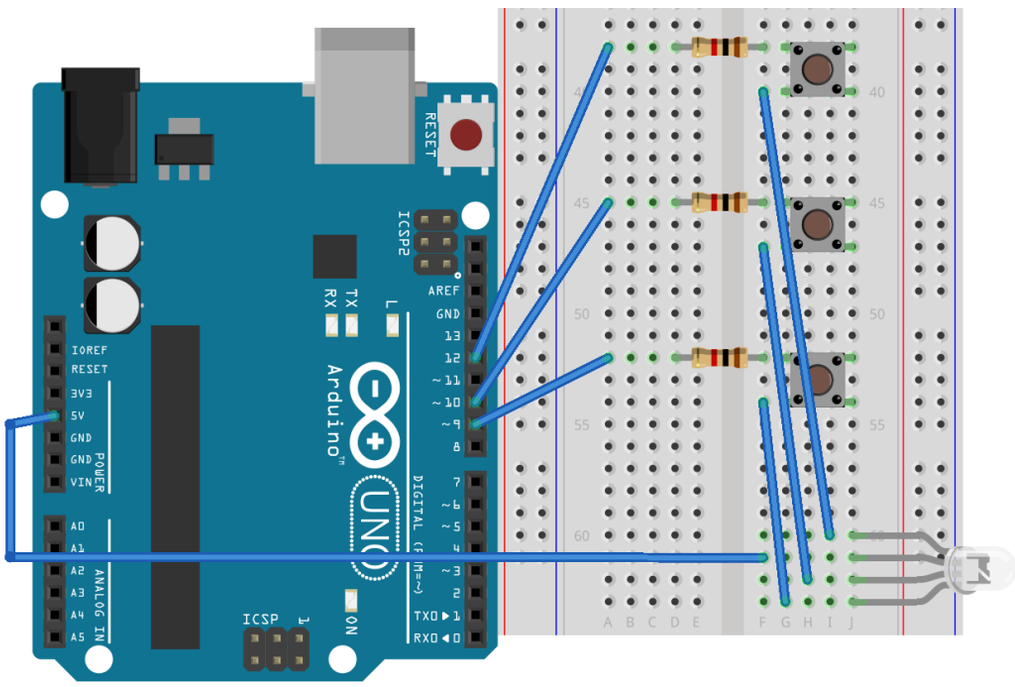


Rysunek 33. Nóżki diody RGB i odpowiadające im kolory.

Przygotujmy układ i program tak, aby obejrzeć podstawowe kolory diody. Potrzebujemy:

1. Komputer,
2. Kabel USB,
3. Płytkę Arduino,
4. Płytkę stykową,
5. Diody RGB,
6. 3 przyciski,
7. 3 rezystory 150 Ω,
8. Przewody.

Układ połącz jak na rysunku:



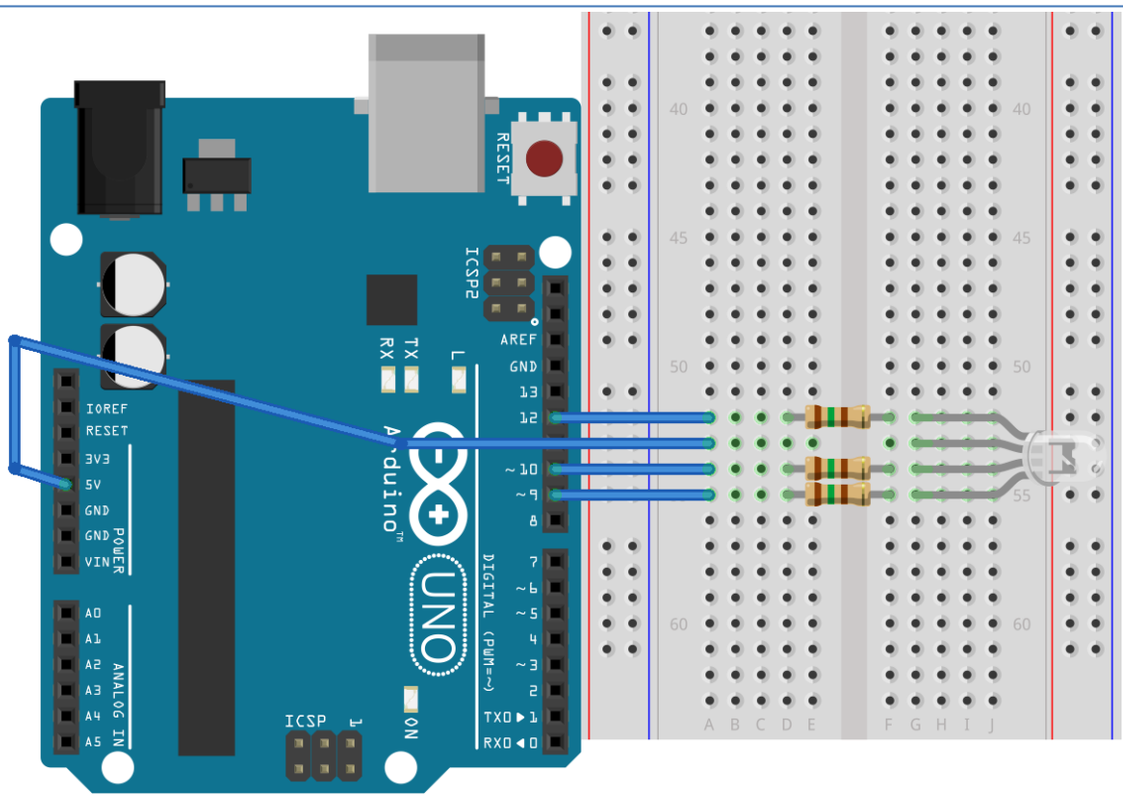
Rysunek 34. Pierwszy układ z diodą RGB.

Program powinien wyglądać następująco:

```
int diodaB = 9;
int diodaG = 10;
int diodaR = 11;

void setup() {
      pinMode(diodaB, OUTPUT);
      pinMode(diodaG, OUTPUT);
      pinMode(diodaR, OUTPUT);
}

void loop(){
      digitalWrite(diodaR, LOW);
      digitalWrite(diodaG, LOW);
      digitalWrite(diodaB, LOW);
}
```

Za pomocą przycisków sprawdź ile kolorów jesteś w stanie uzyskać. Zwróć uwagę na to, że tym razem włączamy diodę ustawiając stan niski. Jest tak, ponieważ używamy wspólnej anody. Pamiętaj o tej różnicy!

Ponieważ taki sposób na włączanie poszczególnych kolorów ma niewiele wspólnego z elektroniką cyfrową, przejdźmy do następnego układu.



Rysunek 35. Drugi układ z diodą RGB.

Spróbuj samodzielnie przygotować program, który będzie włączał każdy z podstawowych kolorów po kolei. W razie problemów, poniżej umieszczono kompletny kod.

```
int diodaB = 9;
int diodaG = 10;
int diodaR = 11;

void setup() {
      pinMode(diodaB, OUTPUT);
      pinMode(diodaG, OUTPUT);
      pinMode(diodaR, OUTPUT);
}

void loop(){
      digitalWrite(diodaB, LOW);
      delay(1000);
      digitalWrite(diodaB, HIGH);
      digitalWrite(diodaG, LOW);
      delay(1000);
      digitalWrite(diodaG, HIGH);
      digitalWrite(diodaR, LOW);
      delay(1000);
      digitalWrite(diodaR, HIGH);
}
```

„analogWrite”

„analogWrite” jest analogowym odpowiednikiem „digitalWrite”. Gdy używaliśmy „digitalWrite” mogliśmy na danym złączu ustawić „HIGH” lub „LOW”, w przypadku „analogWrite”, możemy ustawić dowolną wartość z zakresu od 0 do 255. Pamiętaj o tym, że jeśli chcesz skorzystać z tej funkcji, musisz wykorzystać złącza analogowe, oznaczone na płytce od A0 do A5.

Teraz spróbujemy sterować kolorem diody na podstawie wartości podanej z komputera. Zacznijmy od jednego koloru. Sprawdź, co dzieje się z diodą w zależności od wpisanej wartości (z zakresu od 0 do 255).

```
int diodaR = 11
int diodaG = 10;
int diodaB = 9
int odczyt = 0;

void setup(){
      pinMode(diodaR, OUTPUT);
      pinMode(diodaG, OUTPUT);
      pinMode(diodaB, OUTPUT);
      Serial.begin(9600);
}

void loop(){
      Serial.println("Wprowadz wartosc dla koloru czerwonego");
      while (Serial.available() == 0){}
      odczyt = Serial.parseInt();
      analogWrite(diodaR, odczyt);

      digitalWrite(diodaG, HIGH);
      digitalWrite(diodaB, HIGH);
      delay(100);
}
```

Program zaczynamy od zadeklarowania 3 złączy dla każdej z nóżek diody oraz wartości początkowej zmiennej „odczyt”, do której będziemy zapisywać wartości pobrane z klawiatury. Zaczniemy od sterowania jedynie kolorem czerwonym.

Funkcja „Serial.println()” służy nam do wysyłania informacji tekstowej na ekran monitora. Dzięki linijce „while (Serial.available() == 0){}” program oczekuje na wpisanie wartości przez użytkownika, która w następnej linii zapisywana jest do zmiennej „odczyt”. „analogWrite(diodaR, odczyt)” ustawia na złączu wartość zmiennej „odczyt”. ”digitalWrite(diodaG, HIGH)” oraz „digitalWrite(diodaB, HIGH)” wyłączają kolor zielony i niebieski.

Program zachowuje się mało intuicyjnie. Jeśli wpiszemy wartość 0 uzyskujemy najbardziej czerwony kolor, gdy wpiszemy 255 kolor czerwony jest bardzo blady. Zamień:

```
analogWrite(diodaR, odczyt);
```

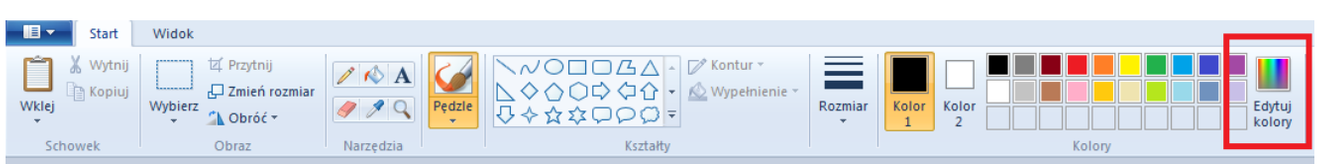
na:

```
analogWrite(diodaR, 255-odczyt);
```

Teraz im wyższą wartość wpiszemy, tym kolor będzie bardziej intensywny. Analogicznie przygotuj program obsługujący wszystkie trzy kolory.

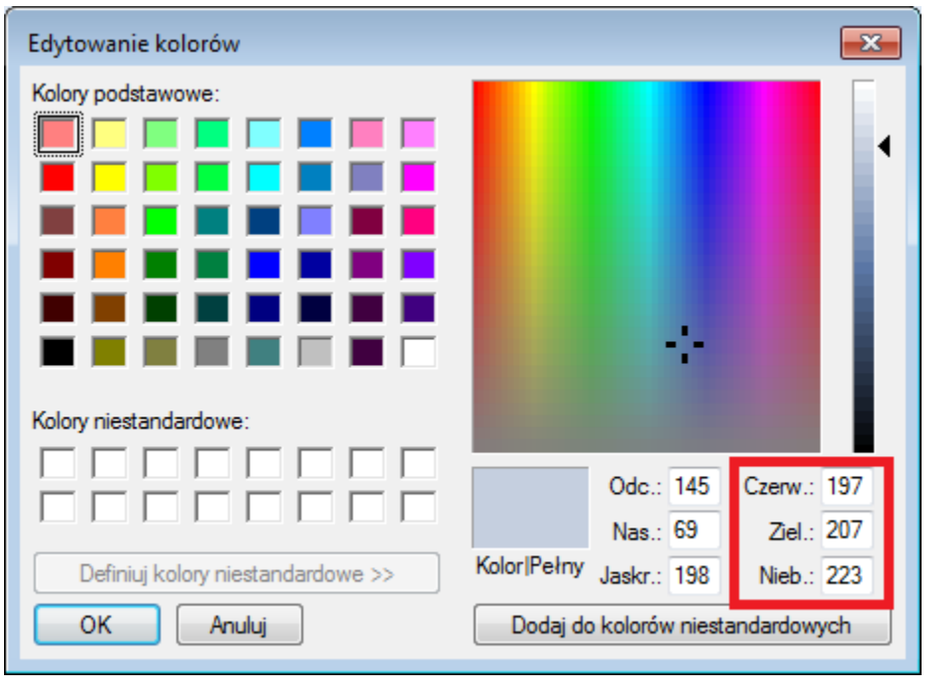
```
int diodaR = 11
int diodaG = 10;
int diodaB = 9
int odczyt = 0;

void setup(){
      pinMode(diodaR, OUTPUT);
      pinMode(diodaG, OUTPUT);
      pinMode(diodaB, OUTPUT);
      Serial.begin(9600);
}

void loop(){
      Serial.println("Wprowadz wartosc dla koloru czerwonego");
      while (Serial.available() == 0){}
      odczyt = Serial.parseInt();
      analogWrite(diodaR, 255-odczyt);

      Serial.println("Wprowadz wartosc dla koloru zielonego");
      while (Serial.available() == 0){}
      odczyt = Serial.parseInt();
      analogWrite(GREENPin, 255- odczyt);

      Serial.println("Wprowadz wartosc dla koloru niebieskiego");
      while (Serial.available() == 0){}
      odczyt = Serial.parseInt();
      analogWrite(BLUEPin, 255- odczyt);

      delay(100);
}
```

Teraz gdy mamy już dostęp do bardzo wielu kolorów, nauczmy się jak ustawić ten oczekiwany. Posłużymy się do tego paletą kolorów RGB dostępną w programie Paint. Uruchom go. Na pasku narzędziowym znajdź „Edytuj kolory” (na poniższym obrazku zaznaczone czerwonym kwadratem).



Rysunek 36. Pasek narzędziowy programu Paint.

W edytorze kolorów znajdź interesujący Cię kolor. Wartości zaznaczone czerwonym kwadratem wprowadź w „monitorze portu szeregowego”. Sprawdź czy kolor diody jest taki jak oczekiwany.



Rysunek 37. Edytor kolorów programu Paint.

Funkcja „random()”

Aby uzyskać ciekawy efekt losowej zmiany kolorów zamiast wczytywania wartości z klawiatury musimy zastosować funkcję „random()”.

Jednak po kilku uruchomieniach, możesz zaobserwować, że za każdym razem losowane są te same ciągi kolorów. Jest tak ponieważ liczby te nie są na prawdę losowane. Jeżeli uruchomisz ten program i zgromadzisz milion liczb, uzyskasz podobną liczbę jedynek, dwójek, trójek i tak dalej. Liczby te nie są losowe w sensie ich nieprzewidywalności. Takie liczby noszą nazwę liczb pseudolosowych. Niestety, mikrokontrolery nie potrafią być nieprzewidywalne.

Co oznaczają liczby 0 i 255 wewnątrz funkcji „random()”? Liczby te to zakres z jakiego ma pochodzić liczba. Wartość pierwsza to początek tego zakresu, wartość druga to jego koniec.

```
int diodaB = 9;
int diodaG = 10;
int diodaR = 11;
int wartoscWylosowana  = 0;

void setup() {
pinMode(diodaR, OUTPUT);
pinMode(diodaB, OUTPUT);
pinMode(diodaG, OUTPUT);
}
```

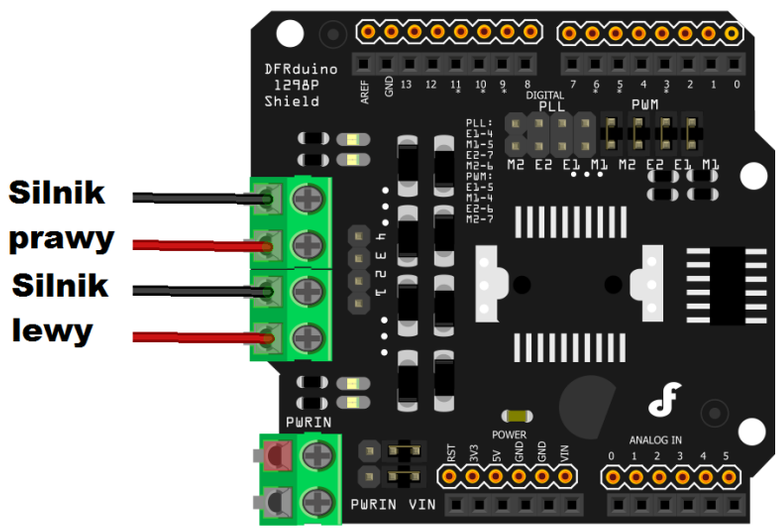
```
void loop(){
      wartoscWylosowana = random(0, 255);
      analogWrite(diodaR, wartoscWylosowana );
      wartoscWylosowana = random(0, 255);
      analogWrite(diodaB, wartoscWylosowana );
      wartoscWylosowana = random(0, 255);
      analogWrite(diodaG, wartoscWylosowana );
      delay(100);
 }
```

Zajęcia 8: Platforma mobilna

Czas wprawić Arduino w ruch. Nauczymy się obsługi silników i podwozia robota. Potrzebujemy:

1. Komputer,
2. Kabel USB,
3. Płytkę Arduino,
4. Platformę mobilną,
5. Sterownik silników,
6. 5 baterii,
7. Przewód zasilającego,
8. Śrubokręt.

Zamontuj Arduino na platformie, a następnie dołącz sterownik silników. Połącz Arduino z platformą za pomocą małego przewodu zasilającego. Do silników doprowadzone są przewody. Ich wolne końce podłącz do sterownika jak na rysunku poniżej.



Rysunek 38. Kolejność podłączenia przewodów do sterownika.

Sprawdźmy teraz czy platforma pojedzie do przodu.

```
int predkoscPrawo = 5;
int kierunekPrawo = 4;
int predkoscLewo = 6;
int kierunekLewo = 7;

void setup(){
      pinMode(kierunekPrawo, OUTPUT);
      pinMode(kierunekLewo, OUTPUT);
}

void loop(){
      digitalWrite(kierunekPrawo, HIGH);
```

```
      digitalWrite(kierunekLewo, HIGH);
      analogWrite(predkoscPrawo, 150);
      analogWrite(predkoscLewo, 150);
}
```

Kod programu zaczynamy od zadeklarowania zmiennych, pod którymi zapisane będą numery złącz odpowiedzialnych za kierunek obrotu każdego z silników (**kierunekPrawo** i **kierunekLewo**) oraz prędkości obu silników (**predkoscPrawo** i **predkoscLewo**). Kierunki, sterowane cyfrowo, są dwa: przód (HIGH) oraz tył (LOW). Prędkość silnika regulujemy analogowo w zakresie od 0 do 255.

Jazda do przodu

Powyższy program umożliwia powolną jazdę do przodu. Przygotujcie na podłodze kilkumetrową trasę. Pierwszym zadaniem jest zatrzymanie się tuż za linią mety. Ponieważ funkcja „loop()" wykonuje swoją zawartość w nieskończoność, musimy znacząco opóźnić kolejne wykonanie po zatrzymaniu się na mecie. Poniższy program pozwala na jazdę trwającą 2 s, następnie następuje wyłączenie silników na 10 s.

```
int predkoscPrawo = 5;
int kierunekPrawo = 4;
int predkoscLewo = 6;
int kierunekLewo = 7;

void setup() {
      pinMode(kierunekPrawo, OUTPUT);
      pinMode(kierunekLewo, OUTPUT);
}

void loop() {
      digitalWrite(kierunekPrawo,HIGH);
      digitalWrite(kierunekLewo, HIGH);
      analogWrite(predkoscPrawo, 150);
      analogWrite(predkoscLewo, 150);
      delay(2000);
      analogWrite(predkoscPrawo, 0);
      analogWrite(predkoscLewo, 0);
      delay(10000);
}
```

Zmieniaj czas jazdy platformy oraz jej prędkość tak aby wykonać postawione zadanie.

Zakręcanie (zawracanie)

Zmodyfikujemy zadanie: po dojechaniu do mety obróć platformę i wróć na start. Są dwie metody pozwalające na obrót platformy.

Metoda pierwsza polega na znacznym obniżeniu (lub zredukowaniu do zera) prędkości jednego z silników. Jeśli lewy będzie obracał się wolniej, platforma skręci w lewo, jeśli prawy będzie się obracał wolniej, platforma skręci w prawo. Pierwszy silnik obraca się szybciej (wartość 200), drugi wolniej (wartość 100). Odpowiadający temu fragment kodu został pogrubiony.

```
int predkoscPrawo = 5;
int kierunekPrawo = 4;
int predkoscLewo = 6;
int kierunekLewo = 7;

void setup() {
     pinMode(kierunekPrawo, OUTPUT);
     pinMode(kierunekLewo, OUTPUT);
}

void loop() {
     digitalWrite(kierunekPrawo,HIGH);
     digitalWrite(kierunekLewo, HIGH);
     analogWrite(predkoscPrawo, 200);
     analogWrite(predkoscLewo, 100);
     delay(2000);
     analogWrite(predkoscPrawo, 0);
     analogWrite(predkoscLewo, 0);
     delay(10000);
}
```

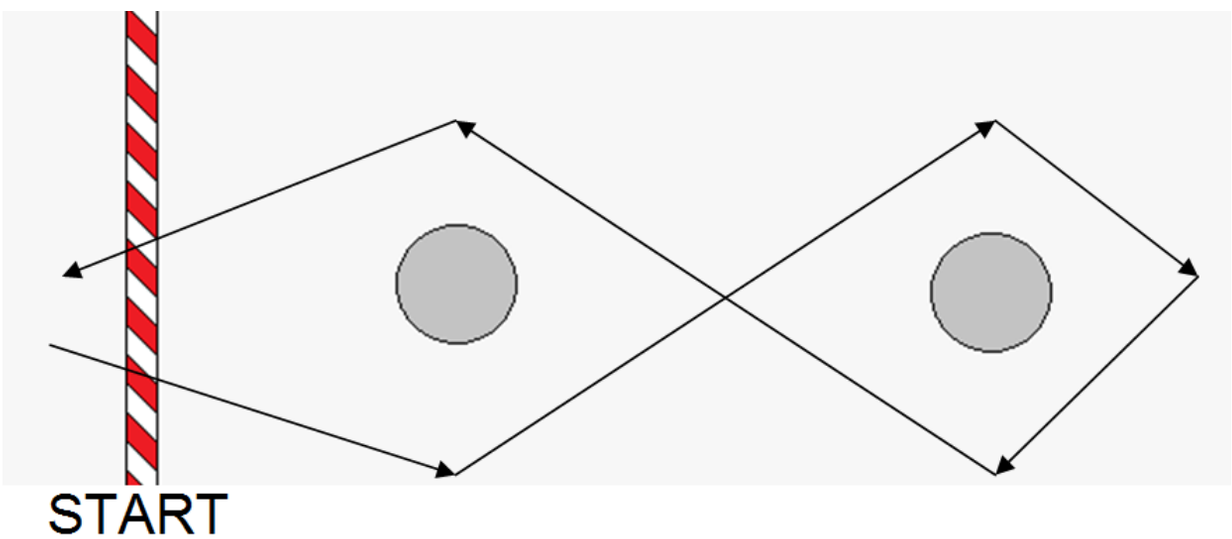
W drugiej metodzie, silniki powinny obracać się w odwrotnych kierunkach (przy takiej samej prędkości obrotu). Odpowiadający temu fragment kodu został pogrubiony.

```
int predkoscPrawo = 5;
int kierunekPrawo = 4;
int predkoscLewo = 6;
int kierunekLewo = 7;

void setup(){
     pinMode(kierunekPrawo, OUTPUT);
     pinMode(kierunekLewo, OUTPUT);
}

void loop(){
     digitalWrite(kierunekPrawo,LOW);
     digitalWrite(kierunekLewo, HIGH);
     analogWrite(predkoscPrawo, 200);
     analogWrite(predkoscLewo, 200);
```

```
      delay(2000);
      analogWrite(predkoscPrawo, 0);
      analogWrite(predkoscLewo, 0);
      delay(10000);
}
```

Slalom

Przygotuj program, który pokonuje trasę jak na rysunku.



Rysunek 39. Slalom.

Ósemka

Ostatnim zadaniem jest jazda po „ósemce”. Program jest bardzo podobny do tego z poprzedniego zadania. Różni się jedynie brakiem stopu po wykonaniu pierwszego slalomu. To czy ósemka zostanie wykonana poprawie zależy od wielu czynników, dlatego samodzielnie popraw czas wykonywania obrotu, jazdy do przodu oraz prędkość pojazdu.

```
int predkoscPrawo = 5;
int kierunekPrawo = 4;
int predkoscLewo = 6;
int kierunekLewo = 7;

void setup() {
      pinMode(kierunekPrawo, OUTPUT);
      pinMode(kierunekLewo, OUTPUT);
}

void loop() {
      digitalWrite(kierunekPrawo, HIGH);
      digitalWrite(kierunekLewo, HIGH);
```

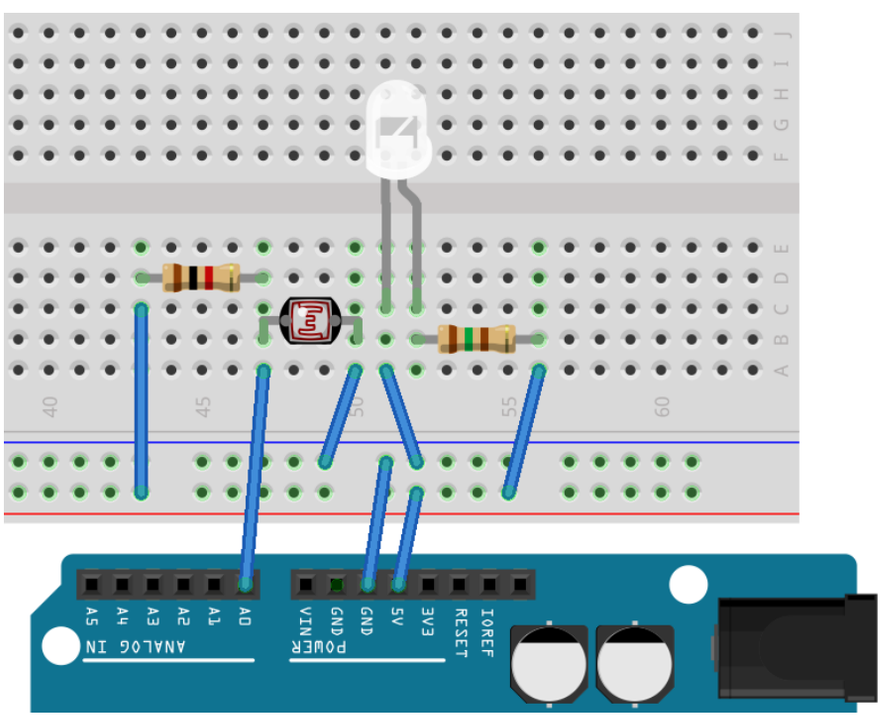
```
      analogWrite(predkoscPrawo, 0);
      analogWrite(predkoscLewo, 200);
      delay(4000);
      analogWrite(predkoscPrawo, 200);
      analogWrite(predkoscLewo, 200);
      delay(1000);
      analogWrite(predkoscPrawo, 200);
      analogWrite(predkoscLewo, 0);
      delay(4000);
      analogWrite(predkoscPrawo, 200);
      analogWrite(predkoscLewo, 200);
      delay(1000);
}
```

Zajęcia 9: Czujnik linii i line follower

Człowiek używa zmysłów do odbierania zewnętrznych bodźców. Dzięki zmysłom widzimy, słyszymy, czujemy (zapach, smak, dotyk). Urządzenia wykorzystujące mikrokontrolery (np. roboty) do obierania zewnętrznych bodźców używają czujników. Są to urządzenia dostarczające informacji o pojawieniu się określonego bodźca, przekroczeniu pewnej wartości progowej lub o wartości rejestrowanej wielkości fizycznej. Czujniki nie przekazują tych informacji bezpośrednio. Informacja zmierzona zamieniana jest zwykle na wielkości elektryczne: napięcie lub natężenie prądu, ponieważ wielkości te łatwo wzmocnić, przesłać na duże odległości, poddać dalszemu przetwarzaniu lub zapisać w pamięci komputerów.

Zajęcia rozpoczniemy od zapoznania się z czujnikiem wykrywania linii. Zanim przejdziemy do obsługi czujnika, przetestujemy go, żeby zrozumieć w jaki sposób działa. Pomoże nam w tym złożenie poniższego układu na płytce stykowej. Potrzebujemy:

1. Komputer,
2. Kabel USB,
3. Płytkę Arduino,
4. Czujnik linii,
5. Diodę białą,
6. Fotorezystor,
7. Rezystor 1000 Ω,
8. Rezystor 150 Ω,
9. Przewody,
10. Białą kartkę,
11. Czarną taśmę lub pisak/flamaster/marker.



Rysunek 40. Testowy układ wykrywający linię.

Prototyp

Program obsługujący testowy układ przygotowaliśmy już na zajęciach dotyczących fotorezystora. Dla przypomnienia:

```
int oswietlenie = 0;
void setup(){
      Serial.begin(9600);
}

void loop(){
      oswietlenie = analogRead(A0);
      Serial.println(oswietlenie);
      delay(500);
}
```

Na białej kartce naklej pasek czarnej taśmy o długości 2 cm lub narysuj na kartce czarnym pisakiem/flamastrem/markerem czarny prostokąt o wymiarach 2 cm na 1 cm. Uruchom monitor portu szeregowego. Kartę umieść równolegle nad diodą i fotorezystorem, tak aby oświetlana była jej biała cześć. Odczytaj wartości z ekranu komputera. Teraz umieść kartę nad diodą i fotorezystorem tak, aby oświetlana była jej ciemna cześć. Odczytaj wartości. Pomiędzy pomiarami powinna być znacząca różnica.

Z czego wynika ta różnica? Przypomnijmy to z lekcji fizyki. Czarny kolor jest kolorem pochłaniającym światło, natomiast kolor biały odbija światło. Dioda w układzie generuje światło, które w większości pochłaniane jest przez czarną powierzchną, jego resztka trafia do fotorezystora. Wtedy w monitorze portu szeregowego wyświetlają się większe wartości (więcej pochłoniętego światła). Natomiast gdy światło odbija się od powierzchni białej, światło nie jest pochłaniane i w większości trafia do fotorezystora a w monitorze portu szeregowego wyświetlają się wartości mniejsze (mniej światła zostało pochłonięte).

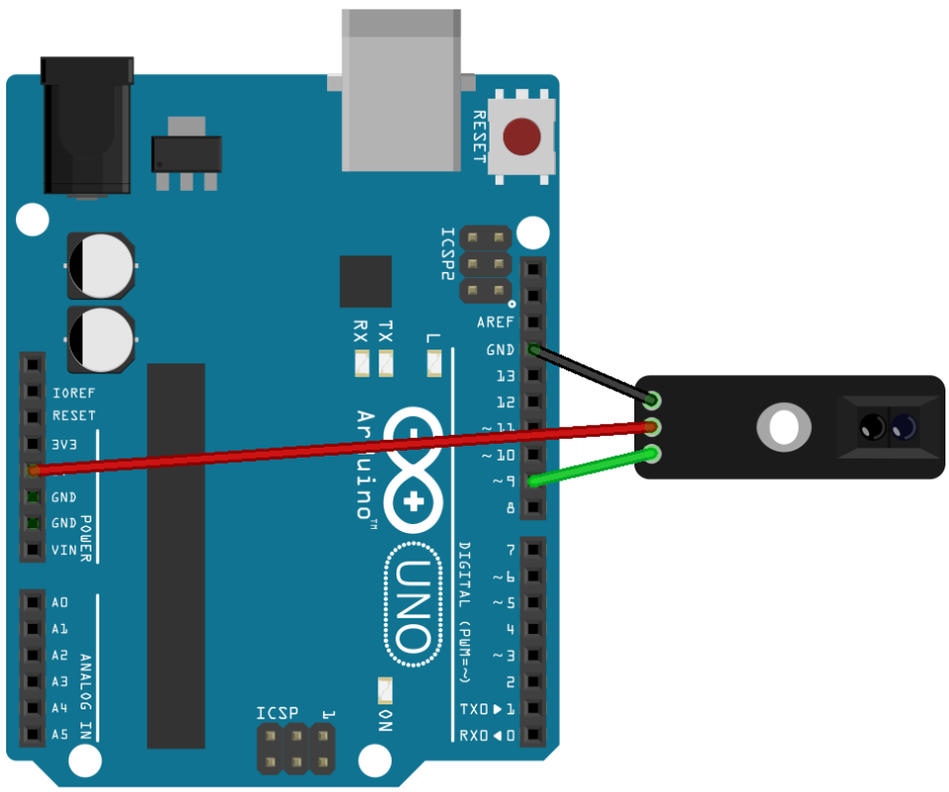
Przekształć teraz kod programu tak, aby zamiast liczby wyświetlał informację o kolorze kartki znajdującej się nad elementami.

```
int oswietlenie = 0;

void setup(){
      Serial.begin(9600);
}

void loop(){
      oswietlenie = analogRead(A0);
      if(oświetlenie > 600){
            Serial.println("kolor czarny");
      }
      else{
            Serial.println("kolor bialy");
      }
      delay(500);
}
```

W podobny sposób działa czujnik linii, z tą różnicą, że zamiast diody emitującą światło widzialne i fotorezystora wykorzystuje diodę emitującą podczerwień oraz fototranzystor.

Czujnik koloru linii

Podłącz czujnik jak na rysunku poniżej:



Rysunek 41. Podłączenie czujnika linii.

Podstawowy kod programu umożliwiający obsługę czujnika wygląda następująco:

```
int czujnik = 9;
int oswietlenie = 0;

void setup(){
      Serial.begin(9600);
}
void loop(){
      oswietlenie = digitalRead(czujnik);
      Serial.println(oswietlenie);
      delay(500);
}
```

Program umożliwia odczyt danych co 0,5 s ze złącza numer 9, gdzie podłączony jest czujnik. Uruchom monitor portu szeregowego i sprawdź, jakie informacje pojawiają się na ekranie komputera. Posłuż się wcześniej przygotowaną kartką.

Kiedy umieścimy czujnik nad kolorem białym, na ekranie wyświetla się wartość 1, gdy czujnik ustawimy nad czarnym kolorem, na ekranie pojawia się wartość 0. Przekształć program podobnie jak w przypadku poprzedniego układu, tak aby zamiast cyfr wyświetlał tekst.

```
int czujnik = 9;
int oswietlenie = 0;

void setup(){
      Serial.begin(9600);
}
void loop(){
      oswietlenie = digitalRead(czujnik);
      if(oświetlenie == 0){
            Serial.println("kolor czarny");
      }
      else{
            Serial.println("kolor bialy");
      }
      delay(500);
}
```

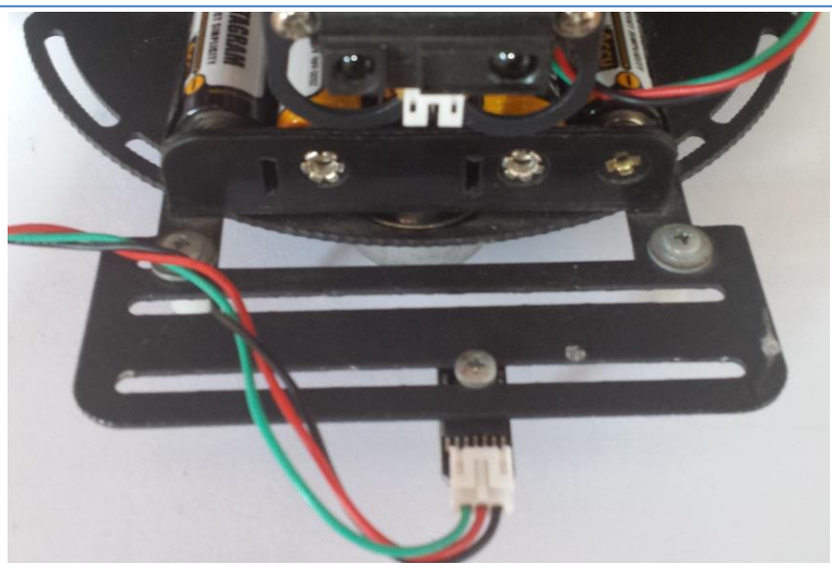
Platforma jeżdżąca – line follower

Potrzebujemy:

1. Komputer,
2. Kabel USB,
3. Płytkę Arduino,
4. Czujnik linii,
5. Przewody,
6. Sterownik silników,
7. Platformę jeżdżącą,
8. Trasę do przejechania.

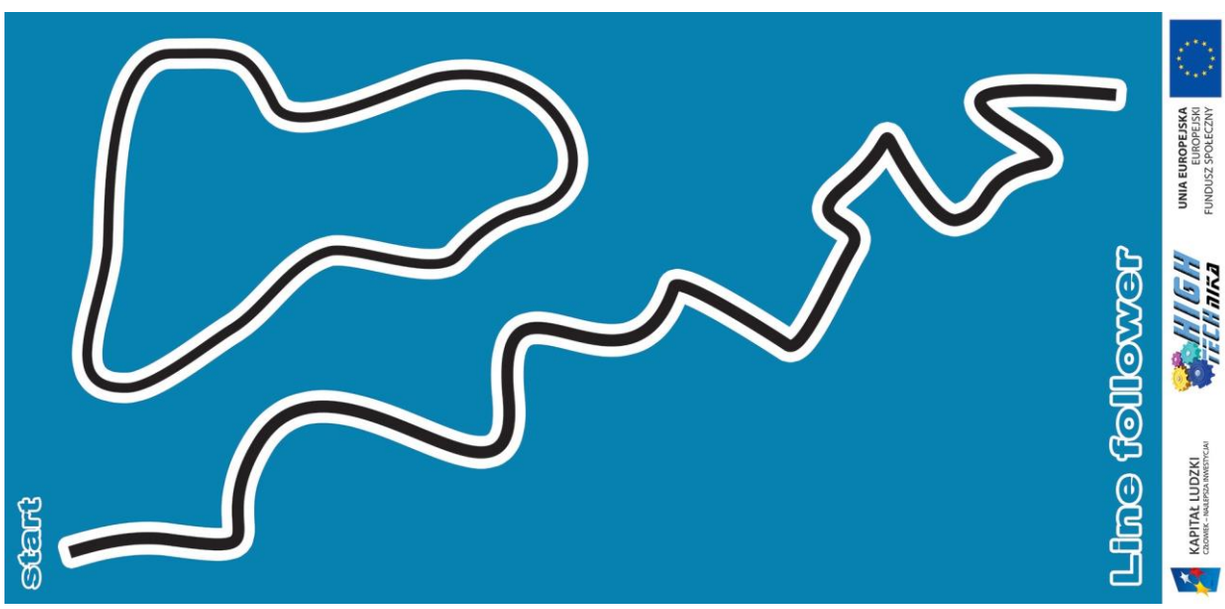
Line follower to nazwa bardzo prostego robota. Nazwa powstała z połączenia dwóch angielskich słów *line* (pol. *linia*) i follow (pol. *podążać*). Line follower oznacza robota podążającego wzdłuż linii. Takie roboty spotyka się np. w fabrykach na liniach produkcyjnych.

Przygotujcie na podłodze trasę, którą będzie miał pokonać robot. Powinna się ona składać ze stykających się ze sobą dwóch pasków: białego i czarnego.



Rysunek 42. Propozycja montażu czujnika linii.

Jak działa line follower? Jeśli czujnik wykrywa kolor biały, to skręca on np. w prawo, natomiast jeśli wykryje czarny kolor to skręca w przeciwną stronę. Oczywiście możesz to odwrócić. Skręt w prawo może odbywać się po wykryciu koloru czarnego. Ruch do przodu składa się więc z bardzo wielu małych zakrętów.



Rysunek 43. Trasa dla line-followera.

```
int czujnik = 9;
int oświetlenie = 0;
int predkoscPrawo = 5;
int kierunekPrawo = 4;
int predkoscLewo = 6;
int kierunekLewo = 7;

void setup(){
     pinMode(kierunekPrawo, OUTPUT);
```

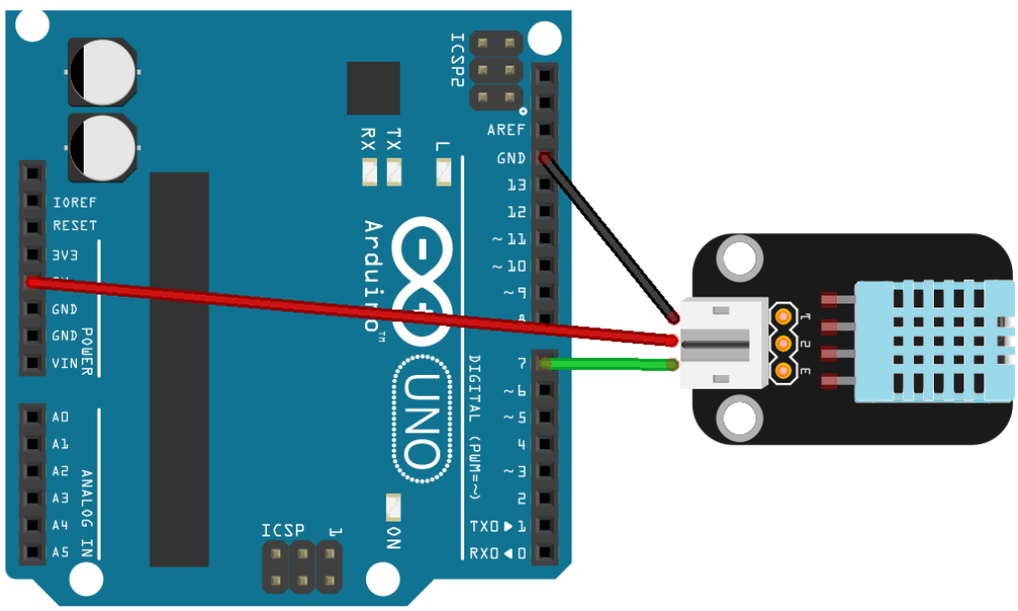
```
      pinMode(kierunekLewo, OUTPUT);
}
void loop(){
      oswietlenie = digitalRead(czujnik);
      if(oswietlenie == 0){
            digitalWrite(kierunekPrawo, HIGH);
            digitalWrite(kierunekLewo, HIGH);
            analogWrite(predkoscPrawo, 100);
            analogWrite(predkoscLewo, 0);
      }
      else{
            digitalWrite(kierunekPrawo, HIGH);
            digitalWrite(kierunekLewo, HIGH);
            analogWrite(predkoscPrawo, 0);
            analogWrite(predkoscLewo, 100);
      }
}
```

Zajęcia 10: Czujnik temperatury i wilgotności

Na dzisiejszych zajęciach omówiony zostanie jest czujnik temperatury i wilgotności – DHT11. Potrzebujemy:

1. Komputer,
2. Kabel USB,
3. Płytkę Arduino,
4. Czujnik temperatury i wilgotności,
5. Przewody.

Układ połącz jak na rysunku poniżej:



Rysunek 44. Podłączenie czujnika temperatury i wilgotności.

Czujnik temperatury

Arduino standardowo nie obsługuje naszego czujnika, w związku z czym musimy posłużyć się dodatkowymi bibliotekami programistycznymi. W tym wypadku pomagają kompilatorowi „zrozumieć" funkcje obsługujące czujnik. To tak jak my, gdy czegoś nie rozumiemy, sięgamy po dodatkowe materiały (słownik, encyklopedię).

Bibliotekę do tego czujnika deklarujemy na początku programu (#include <dht11.h>). W następnej linijce deklarujemy zmienną, pod którą zapisana będzie nazwa własna czujnika (dht11 czujnik) oraz zmienna „sygnal", pod którą zapiszemy numer wybranego złącza.

W funkcji „loop()" pod zmienną „odczyt" zapisujemy pobrane dane z urządzenia „czujnik" podłączonego pod złącze „sygnal". Następna linijka służy do wyświetlenia temperatury w stopniach Celsjusza. Kolejny pomiar realizowany jest po 1000 ms.

```
#include <dht11.h>
dht11 czujnik;
int sygnal = 7;
int odczyt = 0;

void setup(){
      Serial.begin(9600);
}

void loop(){
      odczyt = czujnik.read(sygnal);
      Serial.println(czujnik.temperature);
      delay(1000);
}
```

W przypadku konieczności wyrażenia temperatury w Kelwinach bądź stopniach Fahrenheita niezbędne jest przeliczenie temperatury według poniższych wzorów. Aby przejść z jednej skali na drugą wystarczy znać odpowiednie wzory.

Zamiana skali Celsjusza na skalę Kelwina (jako „wynikWStopniachCelsjusza" rozumiemy wynik w stopniach Celsjusza):

$$wynik\ w\ Kelwinach = \text{wynikWStopniachCelsjusza} + 273{,}15$$

Zamiana skali Celsjusza na skalę Fahrenheita:

$$wynik\ w\ stopniach\ Fahrenheita = (\text{wynikWStopniachCelsjusza} * 1{,}8) + 32$$

Przykład:
Otrzymaliśmy pomiar wynoszący 24° Celsjusza. Zamiana na skalę Kelwina:

$$24 + 273{,}15 = 297{,}15\ K$$

Zamiana na skalę Fahrenheita:

$$(24 * 1{,}8) + 32 = 43{,}2 + 32 = 75{,}2°F$$

Wcześniej wspominaliśmy, że jeśli określimy, że zmienna jest typu „int" to możemy zapisać pod nią jedynie liczby całkowite. Oba wyniki, które otrzymaliśmy w przykładzie są jednak liczbami rzeczywistymi. Do tworzenia zmiennych tego typu wykorzystujemy typ „double".

```
#include <dht11.h>
dht11 czujnik;
int sygnal = 7;
int odczyt = 0;
```

```
int tempC = 0;
double tempK = 0;
double tempF = 0;

void setup(){
      Serial.begin(9600);
}

void loop(){
      odczyt = czujnik.read(sygnal);
      tempC = czujnik.temperature;
      tempK = tempC+273.15;
      tempF =(tempC*1.8)+32;
      Serial.print(tempC);
      Serial.print(" ");
      Serial.print(tempK);
      Serial.print(" ");
      Serial.println(tempF);
      delay(1000);
}
```

W kodzie programu zadeklarowaliśmy 3 zmienne (jedna typu całkowitego i dwie typu rzeczywistego), w których zapiszemy pomierzoną (tempC=czujnik.temperature) i obliczoną temperaturę (tempK=tempC+273.15 oraz tempF=(tempC*1.8)+32).

Program w tej formie zwraca jedynie liczby. Aby program uczynić bardziej użytecznym dołączymy część kodu wyświetlającego informację o tym jaka dana jest obecnie zmierzona. Funkcję „loop()” zmień następująco:

```
void loop(){
      odczyt = czujnik.read(sygnal);
      tempC = czujnik.temperature;
      tempK = tempC+273.15;
      tempF = (tempC*1.8)+32;
      Serial.println("Temperatura w:");
      Serial.print("stopniach Celsjusza: ");
      Serial.print(tempC);
      Serial.println(" C,");
      Serial.print("Kelwinach: ");
      Serial.print(tempK);
      Serial.println(" K,");
      Serial.print("stopniach Fahrenheita: ");
      Serial.print(tempF);
      Serial.println(" F.");
      Serial.println("------------------------------");
      delay(1000);
}
```

Czujnik wilgotności

Czujnik DHT11 umożliwia dodatkowo pomiar wilgotności względnej. Jest to stosunek ciśnienia cząstkowego pary wodnej zawartej w powietrzu do ciśnienia nasycenia, określającego maksymalne ciśnienie cząstkowe pary wodnej w danej temperaturze. Wartość ciśnienia względnego zwykle podawana jest w procentach. Wilgotność względna równa 0% oznacza powietrze suche, zaś równa 100% oznacza powietrze całkowicie nasycone parą wodną.

```
#include <dht11.h>
dht11 czujnik;
int signal = 7;
int odczyt = 0;

void setup(){
      Serial.begin(9600);
}

void loop(){
      odczyt = czujnik.read(sygnal);
      Serial.println(czujnik.humidity);
      delay(1000);
}
```

Program wyświetlający wartość wilgotności względnej różni się jedną linijką (pogrubioną). Zmodyfikujmy funkcję „loop()”, tak aby na ekranie pojawiała się informacja o tym co jest wyświetlane oraz znak „%” po każdej wartości wilgotności.

```
void loop(){
      odczyt = czujnik.read(sygnal);
      Serial.print("Wilgotnosc: ");
      Serial.print(czujnik.humidity);
      Serial.println(" %");
      delay(1000);
}
```

Czujnik temperatury i wilgotności

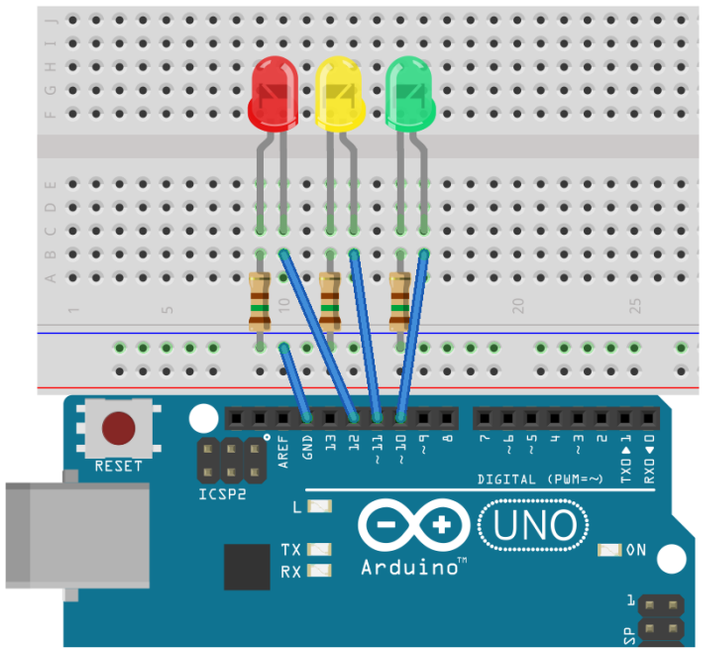
Połączymy oba programy, aby jednocześnie otrzymywać informacje o temperaturze i wilgotności.

```
#include <dht11.h>
dht11 czujnik;
int sygnal = 7;
int odczyt = 0;
int tempC = 0;
double tempK = 0;
```

```
double tempF = 0;

void setup(){
      Serial.begin(9600);
}

void loop(){
      odczyt = czujnik.read(sygnal);
      tempC = czujnik.temperature;
      tempK = tempC+273.15;
      tempF = (tempC*1.8)+32;
      Serial.println("Temperatura w:");
      Serial.print("stopniach Celsjusza: ");
      Serial.print(tempC);
      Serial.println(" C,");
      Serial.print("Kelwinach: ");
      Serial.print(tempK);
      Serial.println(" K,");
      Serial.print("stopniach Fahrenheita: ");
      Serial.print(tempF);
      Serial.println(" F,");
      Serial.print("Wilgotnosc: ");
      Serial.print(czujnik.humidity);
      Serial.println(" %.");
      Serial.println("------------------------------");
      delay(1000);
}
```

Sygnalizacja

Ostatnim etapem ćwiczenia będzie rozbudowa układu o diody sygnalizacyjne. Ponieważ łatwiej i bezpieczniej jest manipulować wilgotnością (np. poprzez chuchnięcie), to o niej będą informować diody. Połącz diody jak na rysunku poniżej.



Rysunek 45. Diody sygnalizacyjne dla czujnika.

```
#include <dht11.h>
dht11 czujnik;
int sygnal = 7;
int odczyt = 0;
int tempC = 0;
double tempK = 0;
double tempF = 0;
int diodaG = 10;
int diodaY = 11;
int diodaR = 12;

void setup(){
      Serial.begin(9600);
      pinMode(diodaG, OUTPUT);
      pinMode(diodaY, OUTPUT);
      pinMode(diodaR, OUTPUT);
}

void loop(){
      odczyt = czujnik.read(sygnal);
      tempC = czujnik.temperature;
      tempK = tempC+273.15;
      tempF = (tempC*1.8)+32;
      Serial.println("Temperatura w:");
      Serial.print("stopniach Celsjusza: ");
      Serial.print(tempC);
      Serial.println(" C,");
      Serial.print("Kelwinach: ");
      Serial.print(tempK);
      Serial.println(" K,");
```

```
      Serial.print("stopniach Fahrenheita: ");
      Serial.print(tempF);
      Serial.println(" F,");
      Serial.print("Wilgotnosc: ");
      Serial.print(czujnik.humidity);
      Serial.println(" %.");
      Serial.println("------------------------------");
      if (czujnik.humidity < 40){
            digitalWrite(diodaG, HIGH);
      }
      else{
            digitalWrite(diodaG, HIGH);
      }
      if(czujnik.humidity < 60) {
            digitalWrite(diodaG, HIGH);
            digitalWrite(diodaY, HIGH);
      }
      else {
            digitalWrite(diodaG, HIGH);
            digitalWrite(diodaY, HIGH);
      }
      if(czujnik.humidity < 80) {
            digitalWrite(diodaG, HIGH);
            digitalWrite(diodaY, HIGH);
            digitalWrite(diodaR, HIGH);
      }
      else {
            digitalWrite(diodaG, HIGH);
            digitalWrite(diodaY, HIGH);
            digitalWrite(diodaR, HIGH);
      }
      delay(1000);
      digitalWrite(diodaG, LOW);
      digitalWrite(diodaY, LOW);
      digitalWrite(diodaR, LOW);
}
```

Za liczbę zapalonych diod odpowiada instrukcja „if”. Dostosuj wartości, przy których włączane są kolejne diody do warunków panujących w pomieszczeniu, w którym jesteś.

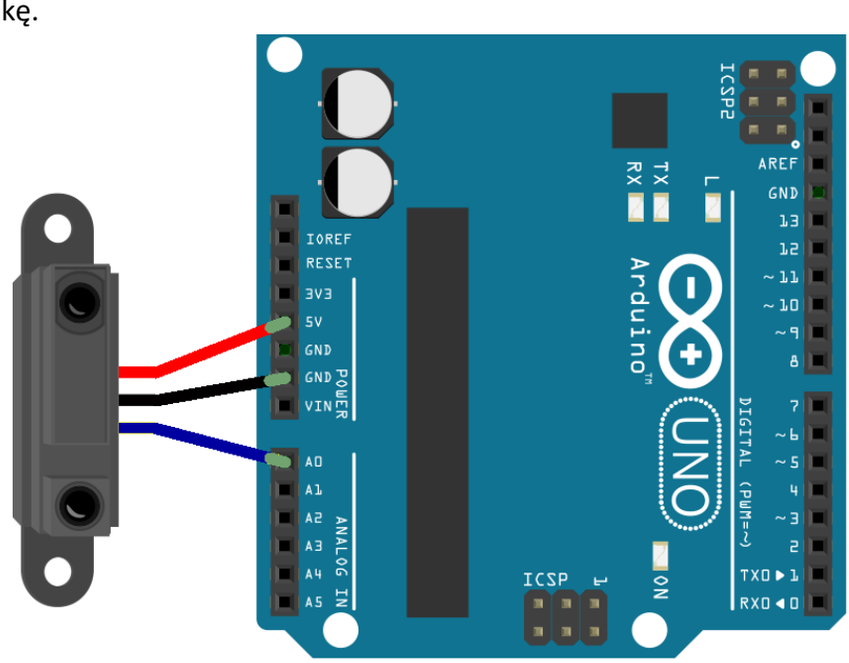
Zajęcia 11: Czujnik odległości

Czujniki odległości są też nazywane dalmierzami. Jest to przyrząd służący do pomiaru odległości bez potrzeby jej przebywania, w odróżnieniu od np. tradycyjnej miarki, którą musimy rozłożyć na mierzonej powierzchni.

Dalmierze wykorzystywane są w wojsku, nawigacji, geodezji, budownictwie, a także w robotyce. Użycie czujnika odległości wraz z platformą jezdną pozwoli robotowi na proste poruszanie się w przestrzeni.

Wykorzystany zostanie czujnik Sharp GP2Y0A21. Podłącz go jak na rysunku poniżej. Potrzebujemy:

1. Komputer,
2. Kabel USB,
3. Płytkę Arduino,
4. Czujnik odległości,
5. Przewody,
6. Linijkę.



Rysunek 46. Podłączenie czujnika odległości.

Kod najprostszego programu obsługującego czujnik wygląda następująco:

```
void setup (){
      Serial.begin (9600);
      pinMode (A0, INPUT);
}

void loop (){
      Serial.println (analogRead (A0));
      delay (1000);
}
```

Uruchom monitor portu szeregowego i spróbuj zmierzyć tę samą odległość za pomocą linijki i czujnika. Czy wyniki się pokrywają? W jakich jednostkach zwracana jest wartość odczytana przez czujnik?

Aby wynik pomiaru zwracany był w znanych nam jednostkach, musimy go przeliczyć wg wzoru:

$$odlegloscWMilimetrach = \left(\frac{67870}{pomiar - 3}\right) - 40$$

Gdzie „pomiar" to wartość odczytana, a „odlegloscWMilimetrach" to odległość wyrażona w milimetrach.

Przekształć kod programu tak, aby w monitorze portu szeregowego wyniki wyświetlane były w milimetrach.

```
int pomiar = 0;
double odleglosc = 0;

void setup(){
      Serial.begin (9600);
      pinMode (A0, INPUT);
}

void loop(){
      pomiar = analogRead (A0);
      odlegloc =((67870 / (pomiar - 3)) - 40);
      Serial.print(odleglosc);
      Serial.println(" mm");
      delay (1000);
}
```

Sprawdź teraz czy pomiary wykonane linijką i wartości wyświetlone na ekranie pokrywają się.

Sprawdź teraz w jakim zakresie wyniki podawane przez mikrokontroler są prawdziwe. Zacznij od 200 mm. Jeśli wyniki się pokrywają, sprawdź wartości o 10 mm mniejsze, i tak dalej, aż do momentu kiedy wyniki będą się między sobą różnić. W ten sposób znajdziesz najmniejszą wartość, jaką jest w stanie pomierzyć czujnik. Powtórz te same operacje, aby znaleźć największą wartość, jaką jest w stanie zmierzyć poprawnie czujnik.

Producent urządzenia podaje, że pomiary wykonywane są prawidłowo w zakresie od 100 do 800 mm. Ze względu na sposób działania czujnika, nie jesteśmy w stanie uwzględnić dolnego ograniczenia (100mm). Rozbuduj kod programu tak, aby uwzględniał górne ograniczenie (800mm).

```
int pomiar = 0;
int odleglosc = 0;

void setup(){
```

```
      Serial.begin(9600);
      pinMode(A0, INPUT);
}

void loop(){
      pomiar = analogRead (A0);
      odległość =((67870 / (pomiar - 3)) - 40);
      if (odleglosc > 800){
            Serial.println("Obiekt zbyt daleko czujnika!");
      }
      else{
            Serial.print (odleglosc);
            Serial.println(" mm");
      }
      delay (1000);
}
```

Program został rozbudowany o instrukcję „if”, która wyświetla informację gdy obiekt jest zbyt blisko lub zbyt daleko lub wyświetla wartość, jeśli pochodzi ona z prawidłowego przedziału.

Czujnik ruchu

Wcześniej przygotowany kod programu łatwo przekształć tak, aby czujnik odległości stał się czujnikiem ruchu.

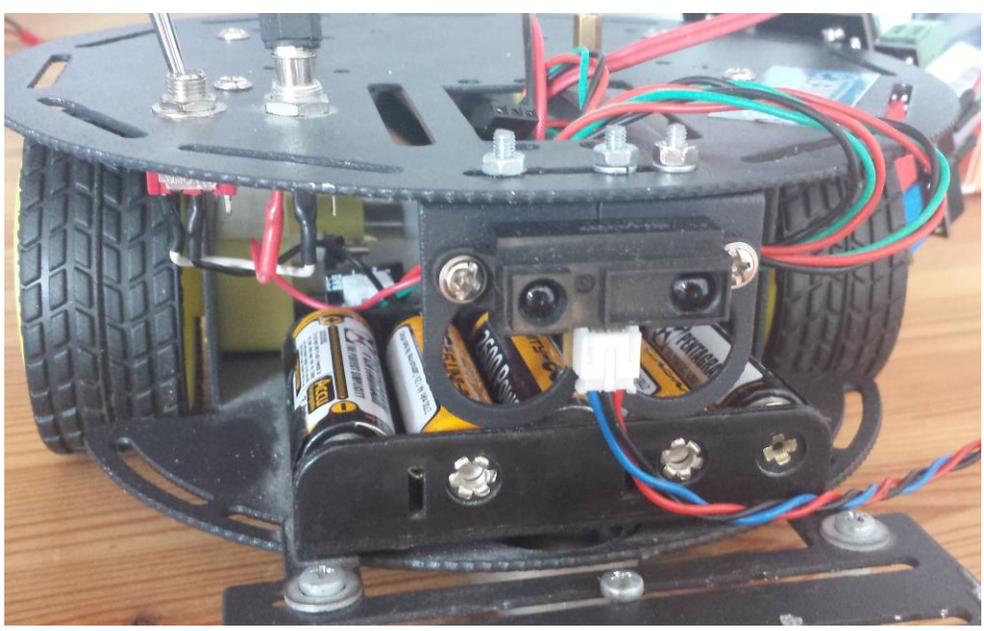
```
int pomiar = 0;
int odleglosc1 = 0;
int odleglosc2 = 0;
int zmiana = 0;

void setup(){
      Serial.begin (9600);
      pinMode (A0, INPUT);
}

void loop(){
      pomiar = analogRead(A0);
      odleglosc1 = ((67870 /(pomiar - 3)) - 40);
      delay(500);
      pomiar = analogRead(A0);
      odleglosc2 = ((67870 / (pomiar - 3)) - 40);

      zmiana = abs(odleglosc2 - odleglosc1);
      if (zmiana > 100){
            Serial.println("RUCH!");
      }
}
```

Mikrokontroler mierzy odległość dwukrotnie <pomiar= analogRead(A0)> w odstępie 0,5 s <delay(500)>. Następnie obliczana jest wartość bezwzględna zmierzonych różnic <zmiana=abs(odleglosc2-odleglosc1)>. Czemu wartość bezwzględna? Jeśli obiekt się oddalił to różnica zmierzonych odległości jest dodatnia, jeśli jednak się przybliżał, to różnica odległości jest ujemna. Wyznaczenie wartości bezwzględnej pozwala zawsze uzyskać wartość dodatnią, którą wykorzystujemy w instrukcji warunkowej decydującej o tym czy ruch został wykryty czy nie <if (zmiana>100)>. Jeśli ruch został wykryty w monitorze portu szeregowego pojawi się informacja "RUCH!".

Sprawdź czy pomiar wykonywany co 0,5 s oraz różnica zmierzonych odległości wynosząca 100 mm to optymalne wartości dla prawidłowego działania czujnika.

Czujnik odległości i jeżdżąca platforma

Wykorzystajmy czujnik odległości do wykrywania przeszkód. Platforma powinna jechać prosto do czasu wykryciu przeszkody (czegokolwiek w odległości mniejszej niż 100 mm). Kiedy ją znajdzie, powinna wycofać się, obrócić i dalej jechać prosto.



Rysunek 47. Propozycja montażu czujnika.

```
int pomiar = 0;
int odleglosc = 0;
int predkoscPrawo = 5;
int kierunekPrawo = 4;
int predkoscLewo = 6;
int kierunekLewo = 7;

void setup(){
      pinMode (A0, INPUT);
      pinMode(kierunekPrawo, OUTPUT);
      pinMode(kierunekLewo, OUTPUT);
```

```
}

void loop(){
      pomiar = analogRead (A0);
      odleglosc =((67870 /(pomiar - 3)) - 40);

      if(odleglosc > 100){
            digitalWrite(kierunekPrawo, HIGH);
            digitalWrite(kierunekLewo, HIGH);
            analogWrite(predkoscPrawo, 200);
            analogWrite(predkoscLewo, 200);
      }
      else{
            digitalWrite(kierunekPrawo, LOW);
            digitalWrite(kierunekLewo, LOW);
            analogWrite(predkoscPrawo, 200);
            analogWrite(predkoscLewo, 200);
            delay(1000);
            digitalWrite(kierunekPrawo, HIGH);
            digitalWrite(kierunekLewo, HIGH);
            analogWrite(predkoscPrawo, 200);
            analogWrite(predkoscLewo, 20);
            delay (1000);
      }
      delay (100);
}
```

Zajęcia 12 i 13: Gra elektroniczna

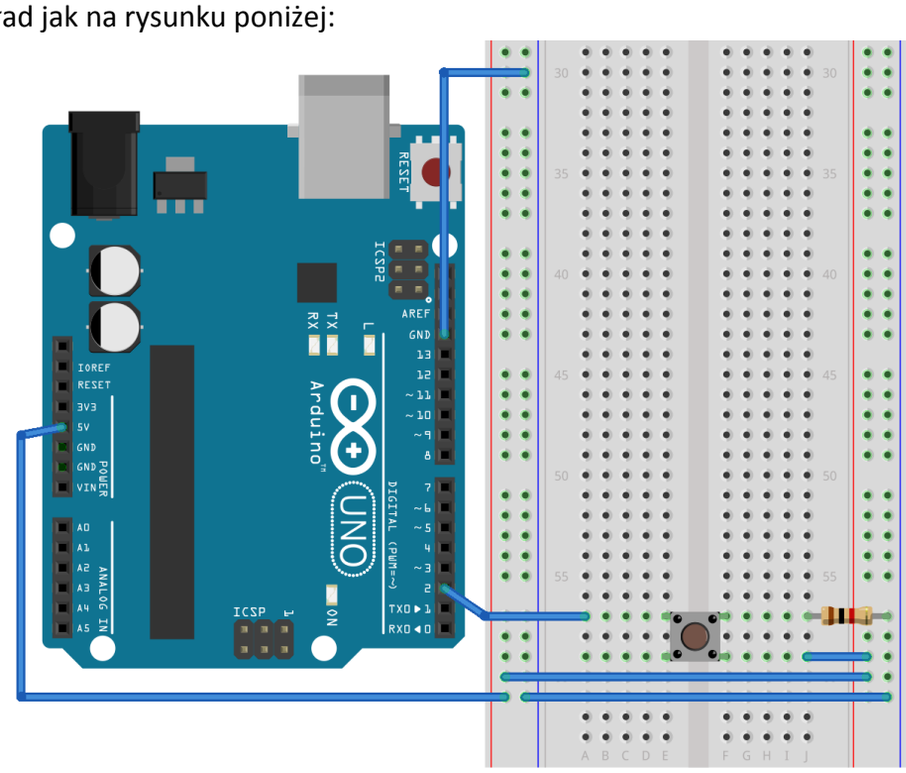
Gry elektroniczne dziś mogą wydawać się bardzo proste, sam proces ich tworzenia do łatwych nienależny. Aby przystąpić zbudowania i zaprogramowania układu imitującego grę, musimy najpierw przygotować kilka mniejszych systemów, takich jak stoper lub generator melodii.

Stoper i pętla do{…}while()

Do przygotowania stopera potrzebujemy:

1. Komputer,
2. Kabel USB,
3. Płytkę Arduino,
4. Płytkę stykową,
5. Przycisk,
6. Rezystor 1000 Ω,
7. Przewody.

Przygotuj układ jak na rysunku poniżej:



Rysunek 48. Pierwszy układ stopera.

Poniższy kod, poza linijką „czas=millis()”, jest zupełnie standardowym programem obsługującym przycisk. Dodatkowa linijka wymaga omówienia. Funkcja „millis()” zwraca czas w milisekundach, który upłynął od czasu uruchomienia programu, który następnie zapisywany jest do zmiennej „czas”. Uruchom monitor portu szeregowego i sprawdź działanie programu. W jaki sposób wyzerować licznik, tak aby pomiar rozpoczął się od początku? Wciśnij przycisk ‘Reset’ na płytce.

```
int przycisk = 2;
int czyWcisnieto = 0;
int czas = 0;

void setup() {
      pinMode(przycisk, INPUT);
      Serial.begin(9600);
}

void loop(){
      Czas = millis();
      czyWcisnieto = digitalRead(przycisk);
      if (czyWcisnieto == HIGH){
            Serial.println(czas);
            delay(200);
      }
}
```

Bardziej intuicyjny jest czas podawany w sekundach. Aby program umożliwiał wyświetlenie wyniku w sekundach zamień linijkę:

```
int czas = 0;
```

na następującą:

```
double czas = 0;
```

Zmieniamy typ zmiennej, ponieważ teraz wynikiem będzie liczba rzeczywista. Druga zmiana dotyczy linijki:

```
Serial.println(czas);
```

Teraz powinna wyglądać tak:

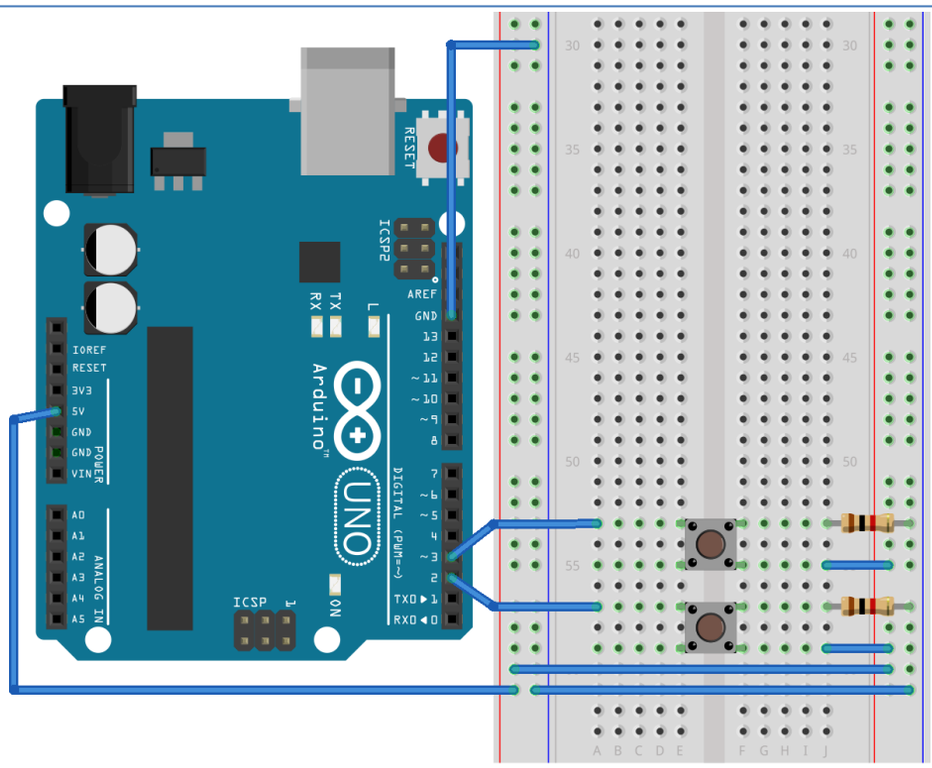
```
Serial.println(czas/1000);
```

Ostatnia linijka wyświetla wartość zmiennej „czas” podzielonej przez 1000, czyli umożliwia przejście z milisekund na sekundy.

Zapoznanie z funkcją zliczającą czas pozwoli nam na przygotowanie stopera, czyli urządzenia mierzącego czas od jednej do drugiej chwili czasowej (czyli od wciśnięcia przycisku ‘start’ do wciśnięcia przycisku „stop”). Aby przygotować taki układ, ten powyższy powinniśmy rozbudować o:

1. Przycisk,
2. Rezystor 1000 Ω.

I powinien wyglądać jak na rysunku poniżej:



Rysunek 49. Drugi układ stopera.

Kod programu dla stopera z dwoma przyciskami powinien wyglądać następująco:

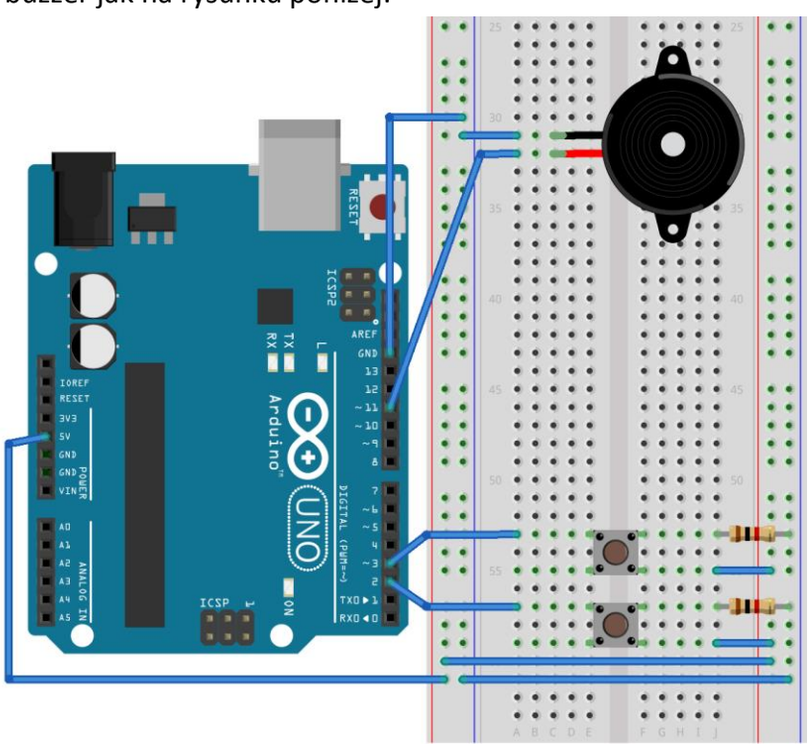
```
int przyciskStart = 2;
int przyciskStop = 3;

int czyWcisnietoPrzyciskStart = 0;
int czyWcisnietoPrzyciskStop = 0;

int czasStart;
int czasStop;

void setup(){
      pinMode(przyciskStart, INPUT);
      pinMode(przyciskStop, INPUT);
      Serial.begin(9600);
}

void loop(){
      czyWcisnietoPrzyciskStart = digitalRead(przyciskStart);
      if (czyWcisnietoPrzyciskStart == HIGH){
            czasStart = millis();
            do{
                  czyWcisnietoPrzyciskStop = digitalRead(przyciskStop);
                  czasStop = millis();
            }
      while(czyWcisnietoPrzyciskStop != HIGH);
      Serial.println((czasStop - czasStart)/1000);
      } }
```

Wykorzystana w programie pętla „do{…}while()” działa podobnie jak wcześniej poznana pętla „while()”. Jedyną różnicą jest moment sprawdzania warunku kończącego działanie pętli. W przypadku „while()”, warunek sprawdzany jest przed wykonaniem zawartości pętli. W przypadku „do{…}while()”, warunek sprawdzany jest po wykonaniu zawartości pętli.

Jeśli został wciśnięty pierwszy przycisk, czyli został spełniony warunek „czyWcisnietoPrzyciskStart == HIGH”, do zmiennej czasStart zapisywany jest czas, który upłynął od uruchomienia programu do wciśnięcia przycisku. Następnie, za pomocą pętli do{…}while(czyWcisnietoPrzyciskStop != HIGH) program oczekuje na wciśnięcie przycisku drugiego. Jeśli to nastąpi, do zmiennej czas zapisywany jest czas, który upłynął od uruchomienia programu do wciśnięcia przycisku. Ostatecznie, na ekranie komputera wyświetlona zostanie różnica zmiennych czasStart i czasStop czyli czas, który upłynął pomiędzy wciśnięciem obu przycisków.

Generator melodii

Za pomocą buzzera i Arduino możemy stworzyć prosty generator melodii. Układ rozbuduj o buzzer jak na rysunku poniżej:



Rysunek 50. Układ generatora melodii.

Dźwięki są rozróżnialne między sobą, dzięki temu, że każdy z nich charakteryzuje się inna wysokością, czasem trwania, głośnością oraz barwą.

Wysokość dźwięku związana jest z częstotliwością drgań, elementu lub elementów, który dany dźwięk wytwarzają. Takimi elementami są zwykle struny, membrany lub powierzchnie instrumentów.

W układzie nie będziemy wykorzystywać prawdziwych instrumentów. Do generowania dźwięku posłużymy się poznanym wcześniej buzzerem. Sterując częstotliwością jego drgań możemy skonstruować prosty generator melodii.

Każdej nucie odpowiada inna częstotliwość (wyrażona w hercach). Pełną oktawę i częstotliwość odpowiadającą kolejnym dźwiękom prezentuje poniższa tabela.

Tabela 2. Dźwięki i odpowiadające im częstotliwości.

nuta	c	d	e	f	g	a	h	C
częstotliwość [Hz]	261,6	293,7	329,2	349,6	391,9	440,0	493,9	523,2

Do generowania dźwięku wykorzystywana jest funkcja „tone()”. Jej pierwszy argument dotyczy złącza, do którego podłączony jest buzzer. Drugi odpowiada częstotliwości generowanego dźwięku. Po każdej funkcji „tone()”, powinna znaleźć się funkcja „delay()”, która pozwoli na odtwarzanie dźwięku przez określony czas. Dźwięk wyłączany jest funkcją „noTone()” z jednym argumentem odpowiadającym złączu, do którego podłączony jest buzzer. Poniżej zaprezentowano kod programu, który odtwarza całą oktawę.

```
int buzzer = 11;

void setup(){
}

void loop(){
      tone(buzzer, 262);
      delay(500);
      noTone(buzzer);
      delay(500);
      tone(buzzer, 294);
      delay(500);
      noTone(buzzer);
      delay(500);
      tone(buzzer, 329);
      delay(500);
      noTone(buzzer);
      delay(500);
      tone(buzzer, 350);
      delay(500);
      noTone(buzzer);
      delay(500);
      tone(buzzer, 392);
      delay(500);
      noTone(buzzer);
      delay(500);
      tone(buzzer, 440);
      delay(500);
      noTone(buzzer);
      delay(500);
      tone(buzzer, 494);
      delay(500);
      noTone(buzzer);
```

```
      delay(500);
      tone(buzzer, 523);
      delay(500);
      noTone(buzzer);
      delay(500);
}
```

Czas na odtworzenie znanej melodii. Jaki to utwór?

```
int buzzer = 11;

void setup(){
}

void loop(){
      tone(buzzer, 392);
      delay(250);
      noTone(buzzer);
      delay(250);
      tone(buzzer, 329);
      delay(250);
      noTone(buzzer);
      delay(250);
      tone(buzzer, 329);
      delay(250);
      noTone(buzzer);
      delay(250);
      tone(buzzer, 350);
      delay(250);
      noTone(buzzer);
      delay(250);
      tone(buzzer, 294);
      delay(250);
      noTone(buzzer);
      delay(250);
      tone(buzzer, 294);
      delay(250);
      noTone(buzzer);
      delay(250);
      tone(buzzer, 262);
      delay(250);
      noTone(buzzer);
      delay(250);
      tone(buzzer, 329);
      delay(250);
      noTone(buzzer);
      delay(250);
     tone(buzzer, 392);
      delay(250);
```

```
      noTone(buzzer);
      delay(250);

      delay(2000);
}
```

Gra elektroniczna

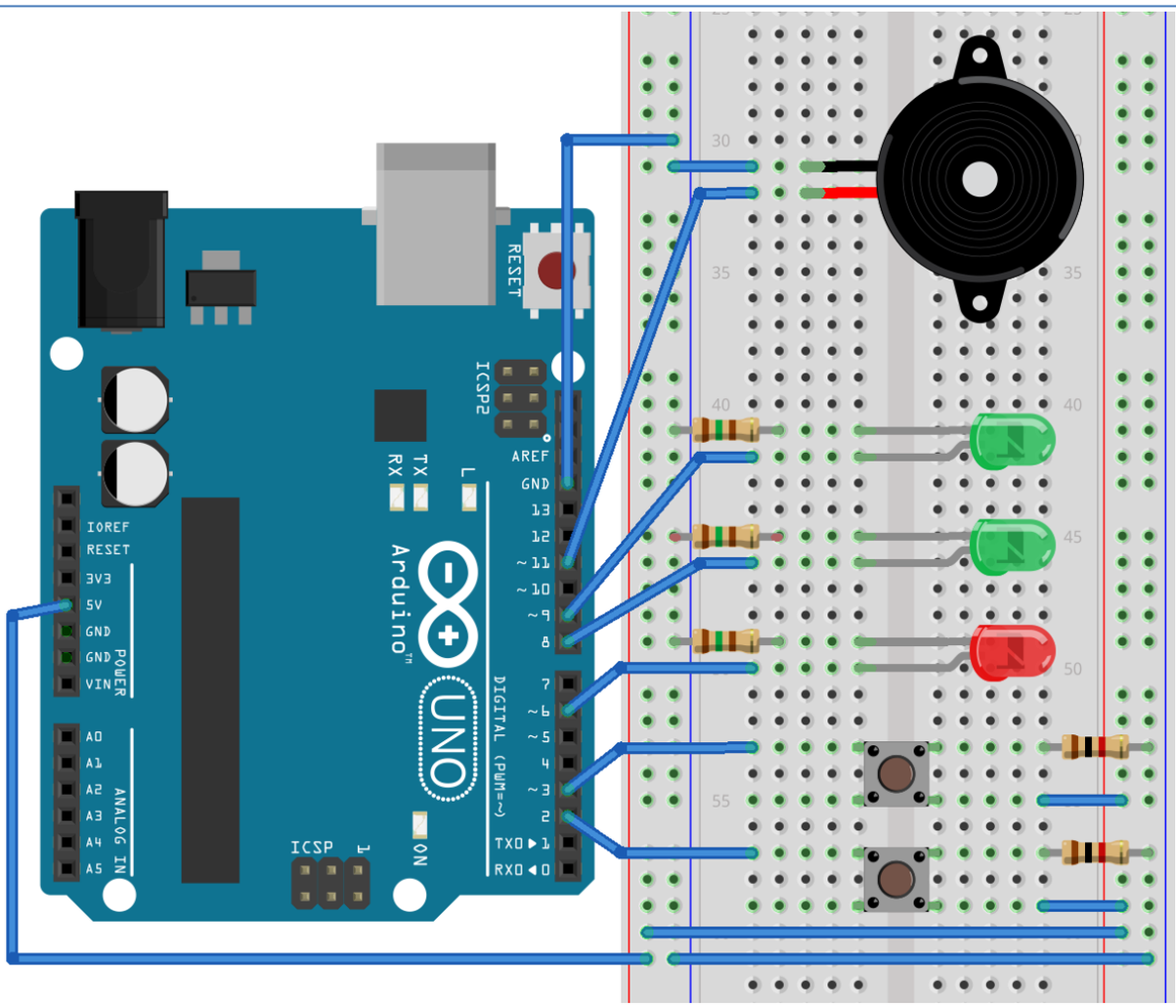
Ponieważ gra ma długi (ale nieskomplikowany!) kod, będziemy go omawiać fragmentami. Ostateczny układ powinien być złożony jak na rysunku poniżej. Do rozbudowy potrzebujemy:

1. 2 diody zielone,
2. 1 diodę czerwoną,
3. 3 rezystory 150 Ω,
4. Przewody.

Kolejne omawiane fragmenty kodu, powinny być wklejane po sobie do pliku z programem sterującym.

W pierwszym etapie deklarujemy zmienne. Podobnie jak w przypadku stopera, pod zmiennymi „PrzyciskGracz1” oraz „PrzyciskGracz2” zapisano złącza, do których podłączyliśmy przyciski, a pod zmiennymi „czyWcisnietoPrzycisk1” oraz „czyWcisnietoPrzycisk2” początkowy stan przycisków (0, czyli niewciśnięte). Do zmiennych „czasPoczatkowy”, „czasGracz1” i „czasGracz2” będziemy zapisywać czas początkowy (moment uruchomienia kolejnej gry) i czasy wciśnięcia przycisków. Następnie zadeklarowaliśmy zmienne, pod którymi zapisaliśmy numery złącz diod i buzzera. Ostatnia zmienna to „dzwiekPoczatkowy”, pod którą zapisaliśmy częstotliwość dźwięku „C”.

```
int PrzyciskGracz1 = 2;
int PrzyciskGracz2 = 3;
int diodaGlowna =  6;
int diodaGracz1 =  8;
int diodaGracz2 =  9;
int buzzer = 11;

int czyWcisnietoPrzycisk1 = 0;
int czyWcisnietoPrzycisk2 = 0;

int czasPoczatkowy;
int czasGracz1;
int czasGracz2;

int dzwiekPoczatkowy= 262;
```



Rysunek 51. Układ "Gra".

```
void setup(){
      pinMode(diodaGlowna, OUTPUT);
      pinMode(diodaGracz1, OUTPUT);
      pinMode(diodaGracz2, OUTPUT);
      pinMode(PrzyciskGracz1, INPUT);
      pinMode(PrzyciskGracz2, INPUT);
      pinMode(buzzer, OUTPUT);
      Serial.begin(9600);
}
```

Następnym fragmentem jest funkcja „setup()". Określiliśmy, że złącza diod i buzzera będą złączami wyjściowymi (OUTPUT), a złącza przycisków, złączami wejściowymi (INPUT). Została także ustalona prędkość transmisji danych „Serial.begin(9600)".

```
void loop(){
      for (int i = 0;i < 3;i = i + 1){
            digitalWrite(diodaGracz1, HIGH);
            digitalWrite(diodaGracz2, HIGH);
            tone(buzzer, dzwiekPoczatkowy);
            delay(500);
```

```
            digitalWrite(diodaGracz1, LOW);
            digitalWrite(diodaGracz2, LOW);
            noTone(buzzer);
            delay(500);
            dzwiekPoczatkowy = dzwiekPoczatkowy+60;
        }
```

Jest to pierwszy fragment funkcji „loop()". Można go nazwać sekwencją startową, ponieważ informuje o rozpoczynającej się grze. Pętla „for()" powtarzana jest trzykrotnie. Za każdym razem włączane i wyłączane są zielone diody. Każdemu załączeniu diod towarzyszy coraz wyższy dźwięk. Dźwięk początkowy został ustalony jako „C" (zmienna „dzwiekPoczatkowy") i w każdej iteracji zwiększany jest o 60 Hz (dzwiekPoczatkowy=dzwiekPoczatkowy+60). Po trzykrotnym sygnale dźwiękowym i zapaleniu się diod, uruchamiana jest gra.

```
        int przerwa = random(2000, 10000);
        delay(przerwa);
        digitalWrite(diodaGlowna, HIGH);
        delay(100);
        digitalWrite(diodaGlowna, LOW);
        czasPoczatkowy = mills();
```

Ten fragment odpowiada za sterowanie diodą czerwoną. Pierwszym jego zadaniem jest wylosowanie długości odstępu czasu pomiędzy zakończeniem odliczania a zapaleniem się czerwonej diody. Do wylosowania tej wartości wykorzystaliśmy „random(2000,10000)", więc ten odstęp czasowy będzie wynosił od 2 do 10 s. „delay(przerwa)" służy do zrealizowania tego odstępu. Następnie dioda czerwona jest załączana i odbywa się pierwszy pomiar czasu „czasPoczatkowy=millis()".

```
    do{
        czyWcisnietoPrzycisk1 = digitalRead(PrzyciskGracz1);
        czyWcisnietoPrzycisk2 = digitalRead(PrzyciskGracz2);
        if (czyWcisnietoPrzycisk1 == HIGH) {
            czasGracz1 = millis();
        }
        if (czyWcisnietoPrzycisk2 == HIGH) {
            czasGracz2 = millis();
        }
    }
    while((czasGracz1 == 0) || (czasGracz2 == 0));
```

Fragment ten odpowiada za obsługę przycisków. Po zapaleniu się diody czerwonej, pętla „do{…}while()" działa tak długo, póki obaj gracze nie wcisną przycisków. Wciśnięcie przycisków powoduje pomiar czasu (np.: if (czyWcisnietoPrzycisk1 == HIGH) {czasGracz1=micros();} dla gracza pierwszego).

```
      if (czasGracz1 < czasGracz2){
            digitalWrite(diodaGracz1, HIGH);
      }
      else{
            digitalWrite(diodaGracz2, HIGH);
      }
      delay(5000);
```

Przedostatni fragment kodu służy do ustalenia zwycięzcy. Służy do tego porównanie czasów obu graczy w instrukcji „if”. Dioda gracza, który szybciej wcisnął przycisk załączy się na 5s.

```
      czasGracz1 = 0;
      czasGracz2 = 0;
      dzwiekPoczatkowy = 262;
      digitalWrite(diodaGracz1, LOW);
      digitalWrite(diodaGracz2, LOW);
}
```

Na koniec czasy obu graczy są zerowane, dźwięk „C” znowu staje się dźwiękiem początkowym a włączona w poprzednim fragmencie dioda zielona zostaje wyłączona.

Przetestuj czy gra działa poprawnie.

Powyższy kod nie uwzględnia wyświetlenia wyników i ustalenia ostatecznego zwycięzcy. Aby dodać te funkcje do gry, w miejscu gdzie deklarowaliśmy zmienne wklej poniższe deklaracje:

```
int wygraneGracz1 = 0;
int wygraneGracz2 = 0;
int liczbaRund = 0;
```

Dwa ostatnie fragmenty zamień następująco (dodana część została pogrubiona):

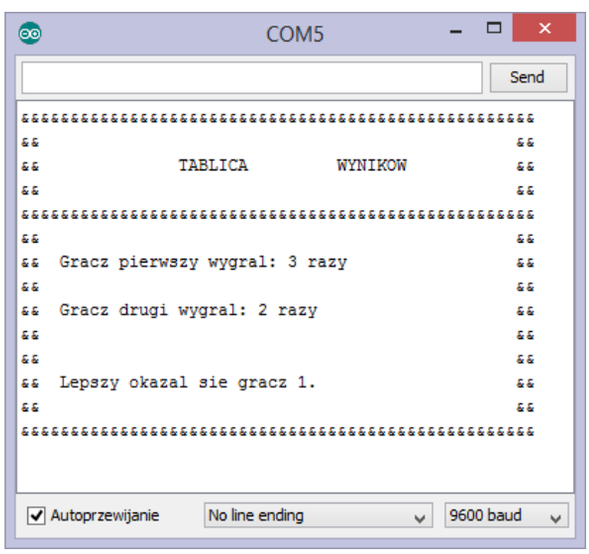
```
      if (czasGracz1 < czasGracz2){
            digitalWrite(diodaGracz1, HIGH);
            wygraneGracz1 = wygraneGracz1 + 1;
      }
      else{
            digitalWrite(diodaGracz2, HIGH);
            wygraneGracz2 = wygraneGracz2 + 1;
      }
      delay(5000);
      liczbaRund = liczbaRund + 1;

      if (liczbaRund == 5){
            Serial.print("Gracz pierwszy wygral: ");
            Serial.print(wygraneGracz1);
            Serial.println(" razy");
            Serial.print("Gracz pierwszy wygral: ");
            Serial.print(wygraneGracz2);
            Serial.println(" razy");
```

```
            if (wygraneGracz1 > wygraneGracz2){
                  Serial.println("Lepszy okazal sie gracz 1.");
            }
            else{
                  Serial.println("Lepszy okazal sie gracz 2.");
            }
      }

      czasGracz1 = 0;
      czasGracz2 = 0;
      dzwiekPoczatkowy = 262;
      wygraneGracz1 = 0;
      wygraneGracz2 = 0;
      liczbaRund = 0;
      digitalWrite(diodaGracz1, LOW);
      digitalWrite(diodaGracz2, LOW);
}
```

Pod zmienne „wygraneGracz1” oraz „wygraneGracz2” zapisywana jest liczba wygranych każdego z graczy. Początkowo, pod każdą z nich zapisane jest 0. Za każdym razem kiedy gracz pierwszy wygrywa liczba wygranych zwiększana jest o 1 (wygraneGracz1=wygraneGracz1+1), podobnie dla gracza 2 (wygraneGracz2=wygraneGracz2+1). Po każdej z rund zwiększana o 1 jest wartość zmiennej „iloscRund” (iloscRund=iloscRund+1). Po 5 rundach wyświetlane są wyniki (if (iloscRund==5)…).

Tablica wyników

Ostatnia część modyfikacji programu dotyczy wyświetlenia wyników w formie jak na rysunku poniżej. Polega ona jedynie na edycji zawartości instrukcji warunkowej „if (iloscRund==5)…”.



Rysunek 52. Tablica wyników w monitorze portu szeregowego.

```
      if (liczbaRund == 5){
            Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
      &&&&&&&&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&\t\tTABLICA         WYNIKOW\t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
      &&&&&&&&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.print("&&  Gracz pierwszy wygral: ");
            Serial.print(wygraneGracz1);
            Serial.println(" razy  \t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.print("&&  Gracz drugi wygral: ");
            Serial.print(wygraneGracz2);
            Serial.println(" razy  \t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            if (wygraneGracz1 > wygraneGracz2){
                  Serial.println("&&  Lepszy okazal sie gracz 1.\t\t\t
&&");
            }
            else{
                  Serial.println("&&  Lepszy okazal sie gracz 2.\t\t\t
&&");
            }
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
      &&&&&&&&");
      }
```

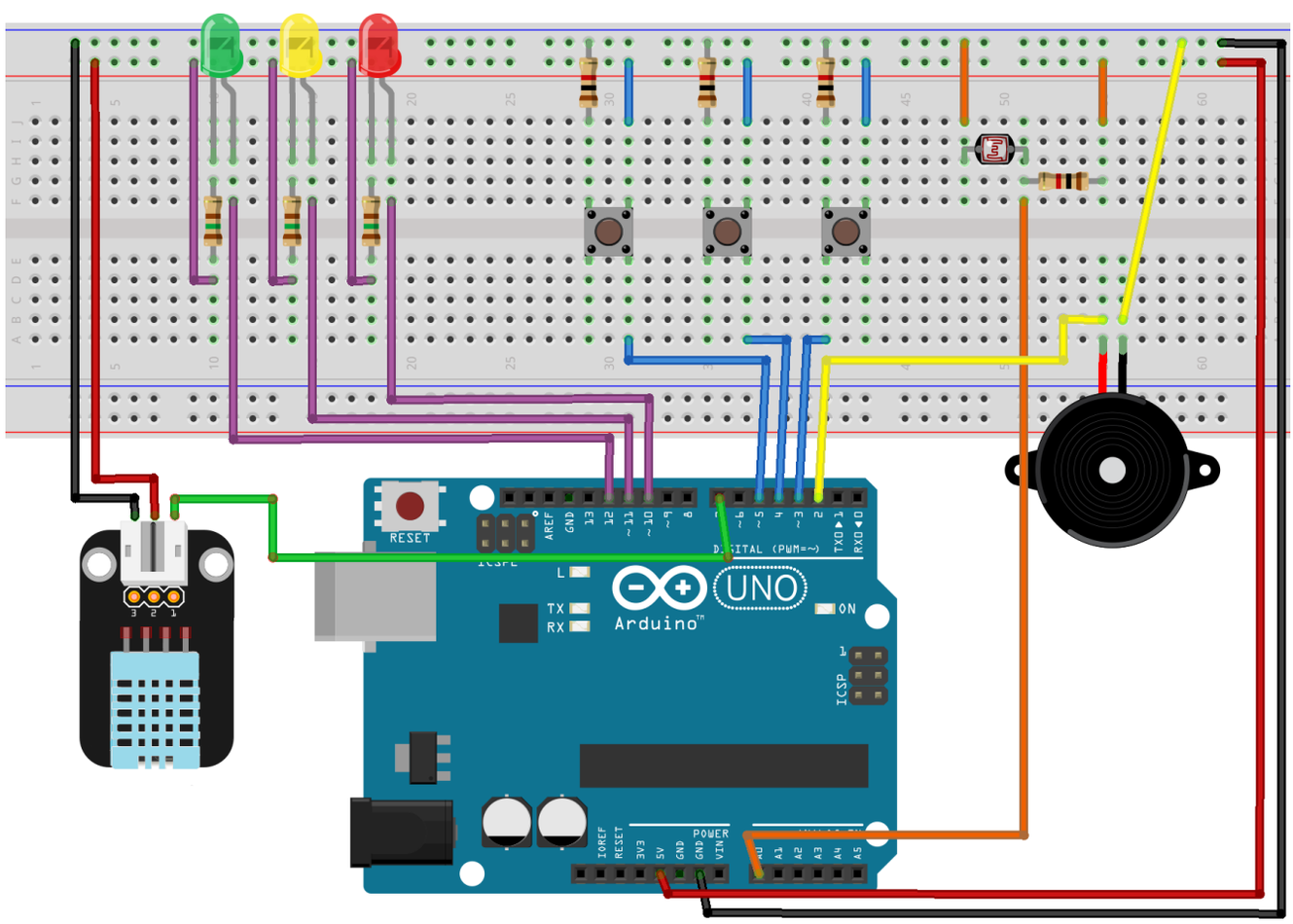
Tablica wyników generowana jest linijka po linijce . Nowością w tej części jest „\t”. Jest to oznaczenie tabulatora i pozwala na uzyskanie takiego samego efektu (odstępu) jak użycie przycisku „TAB” z klawiatury w edytorze tekstu.

Zajęcia 14 i 15: Stacja meteorologiczna

Stacja meteorologiczna to miejsce gdzie prowadzi się obserwacje (pomiary) pogody i klimatu. Podstawowy zestaw przyrządów to termometr, wiatromierz, deszczomierz, termometr gruntowy, heliograf (urządzenie służące do pomiaru czasu usłonecznienia), ewentualnie przyrządy do pomiarów promieniowania słonecznego.

Do budowy stacji meteorologicznej potrzebujemy:

1. Komputer,
2. Kabel USB,
3. Płytkę Arduino,
4. Płytkę stykową,
5. Czujnik temperatury i wilgotności,
6. Fotorezystor,
7. 4 rezystory 1000 Ω,
8. 3 rezystory 150 Ω,
9. 3 przyciski,
10. Diodę zieloną, żółtą i czerwoną,
11. Przewody.



Rysunek 53. Układ „Stacja meteorologiczna”.

Prosta stacja meteorologiczna

Najprostsza stacja meteorologiczna powstanie poprzez połączenie programów obsługujących fotorezystor i czujnik temperatury i wilgotności.

```
#include <dht11.h>
dht11 czujnik;
int sygnalCzujnik = 7;
int odczytCzujnik = 0;
int oswietlenie = 0;

void setup() {
      Serial.begin(9600);
}

void loop() {
      oswietlenie = analogRead(A0);
      odczyt = czujnik.read(sygnal);
      Serial.println(oswietlenie);
      Serial.println(czujnik.temperature);
      Serial.println(czujnik.humidity);
      delay(1000);
}
```

Za pomocą tego programu w monitorze portu szeregowego co sekundę zostaną podane informacje o temperaturze, wilgotności i oświetleniu. Jednak taka stacja nie jest zbyt użyteczna, ponieważ wyświetla tylko „surowe”, nieopisane liczby.

Stacja meteorologiczna z sygnalizacją diodową

Nieco bardziej przydatna będzie stacja z sygnalizacją diodową, niewykorzystującą monitora portu szeregowego. Dzięki temu będzie ona spełniała swoją funkcję po odłączeniu od komputera i podłączeniu zewnętrznego zasilania.

Do informowania użytkownika na temat aktualnej temperatury, wilgotności lub oświetlenia wykorzystywane są trzy, różnokolorowe diody. Włączenie zielonej oznacza, że zmierzone wartości danego parametru są optymalne, co np. w przypadku temperatury oznacza, że wynosi ona mniej niż 22 stopnie Celsjusza.

Włączenie zielonej i żółtej diody oznacza, że dany parametr jest nieznacznie ponad prawidłową wartością. Natomiast Włączenie wszystkich trzech diod oznacza, że zmierzone wartości znacząco przekraczają normę (np. jest za gorąco).

Diody zostaną Włączone na 2 s po wciśnięciu jednego z przycisków. Każdy z nich odpowiada jednemu z mierzonych parametrów. Od programisty zależy, za który z nich odpowiada dany przycisk.

Podobnie jak w przypadku gry elektronicznej kod programu jest długi ale nieskomplikowany, dlatego będziemy omawiać go partami.

```
#include <dht11.h>
dht11 czujnik;
int sygnalCzujnik = 7;
int odczytCzujnik = 0;
int oświetlenie = 0;

int buzzer = 2;

int przyciskTemperatura = 3;
int przyciskWilgotnosc = 4;
int przyciskOswietlenie = 5;

int diodaCzerwona = 10;
int diodaZolta = 11;
int diodaZielona = 12;

int czyWcisnietoPrzyciskTemperatura= 0;
int czyWcisnietoPrzyciskWilgotnosc = 0;
int czyWcisnietoPrzyciskOswietlenie = 0;
```

Kod programu rozpoczynamy od załączenia biblioteki obsługującej czujnik temperatury i wilgotności (#include <dht11.h>). Następnie do zmiennych zapisujemy złącza, do których podłączone są przyciski, diody, buzzer, czujnik, a także początkowe parametry zmiennych, do których będziemy zapisywać stan przycisku (czyWcisnieto…), poziom oświetlenia (oświetlenie) i dane pobrane z czujnika (odczytCzujnik).

```
void setup(){
      pinMode(diodaCzerwona, OUTPUT);
      pinMode(diodaZolta, OUTPUT);
      pinMode(diodaZielona, OUTPUT);
      pinMode(przyciskTemperatura, INPUT);
      pinMode(przyciskWilgotnosc, INPUT);
      pinMode(przyciskOswietlenie, INPUT);
      pinMode(buzzer, OUTPUT);
 }
```

Następnie w funkcji „setup()” ustawiamy złącza przycisków jako pobierające sygnał (wejściowe), a złącza diod jako nadające sygnał sterujący (wyjściowe).

Funkcję „loop()” rozłożymy na trzy mniejsze fragmenty, każdy z nich odpowiada zachowaniu układu po wciśnięciu kolejnego przycisku.

Jeśli został wciśnięty przyciskTemperatura (czyWcisnietoPrzyciskTemperatura= digitalRead(przyciskTemperatura); if (czyWcisnietoPrzyciskTemperatura== HIGH)…”) odczytywana jest aktualna temperatura i jeśli jej wartość jest mniejsza niż 22 stopnie Celsjusza to zapala się jedynie zielona dioda. Jeśli jej wartość jest mniejsza niż 28 stopni ale większa lub

równa 22 stopnie to zapalają się diody zielona i żółta. Jeśli natomiast odczytana wartość wynosi więcej niż 28 stopni, włączne są wszystkie trzy diody.

```
void loop(){
     czyWcisnietoPrzyciskTemperatura =
digitalRead(przyciskTemperatura);
     if (czyWcisnietoPrzyciskTemperatura == HIGH) {
          odczytCzujnik = czujnik.read(sygnalCzujnik);
           if (czujnik.temperature < 22){
                    digitalWrite(diodaZielona, HIGH);
                    delay(2000);
                    digitalWrite(diodaZielona, LOW);
          }
          else if (czujnik.temperature < 28){
               digitalWrite(diodaZielona, HIGH);
                    digitalWrite(diodaZolta, HIGH);
                    delay(2000);
                    digitalWrite(diodaZielona, LOW);
                    digitalWrite(diodaZolta, LOW);
          }
          else{
                    digitalWrite(diodaZielona, HIGH);
                    digitalWrite(diodaZolta, HIGH);
                    digitalWrite(diodaCzerwona, HIGH);
                    delay(2000);
                    digitalWrite(diodaZielona, LOW);
                    digitalWrite(diodaZolta, LOW);
                    digitalWrite(diodaCzerwona, LOW);
           }
     }
```

Drugi fragment jest niemal identyczny. Jeśli został wciśnięty przyciskWilgotnosc (czyWcisnietoPrzyciskWilgotnosc = digitalRead(przyciskWilgotnosc); if (czyWcisnietoPrzyciskTemperatura== HIGH)…") odczytywana jest aktualna wilgotność i jeśli jej wartość jest mniejsza niż 30%, to zapala się jedynie zielona dioda. Jeśli jej wartość jest mniejsza niż 60%, ale większa lub równa 30%, to zapalają się diody zielona i żółta. Jeśli natomiast odczytana wartość wynosi więcej niż 60%, włączne są wszystkie trzy diody.

```
     czyWcisnietoPrzyciskWilgotnosc =
digitalRead(przyciskWilgotnosc);
     if (czyWcisnietoPrzyciskWilgotnosc == HIGH) {
          odczytCzujnik = czujnik.read(sygnalCzujnik);
           if (czujnik.humidity < 30){
                    digitalWrite(diodaZielona, HIGH);
                    delay(2000);
                    digitalWrite(diodaZielona, LOW);
          }
          else if (czujnik.humidity < 60){
```

```
                digitalWrite(diodaZielona, HIGH);
                digitalWrite(diodaZolta, HIGH);
                     delay(2000);
                     digitalWrite(diodaZielona, LOW);
                     digitalWrite(diodaZolta, LOW);
           }
           else {
                     digitalWrite(diodaZielona, HIGH);
                     digitalWrite(diodaZolta, HIGH);
                     digitalWrite(diodaCzerwona, HIGH);
                     delay(2000);
                     digitalWrite(diodaZielona, LOW);
                     digitalWrite(diodaZolta, LOW);
                     digitalWrite(diodaCzerwona, LOW);
           }
     }
```

Ostatni fragment programu jest bardzo podobny do poprzednich. Jeśli został wciśnięty przyciskOswietlenie (czyWcisnietoPrzyciskOswietlenie = digitalRead(przyciskOswietlenie); if (czyWcisnietoPrzyciskTemperatura== HIGH)…") odczytywana jest aktualna wartość oświetlenia, przeliczana na procenty i jeśli jej wartość jest mniejsza niż 50%, to zapala się jedynie zielona dioda. Jeśli jego wartość jest mniejsza niż 75%, ale większa lub równa 50%, to zapalają się diody zielona i żółta. Jeśli natomiast odczytana wartość wynosi więcej niż 75%, złączne są wszystkie trzy diody.

Wartość odczytana jest zamieniana na wartość procentową następująco:

$$wartość\ w\ procentach = \frac{wartość\ odczytana}{1024} * 100\%$$

```
     czyWcisnietoPrzyciskOswietlenie =
digitalRead(przyciskOswietlenie);
     if (czyWcisnietoPrzyciskOswietlenie == HIGH) {
          oswietlenie = analogRead(A0);
          if ((oświetlenie / 1024) * 100 < 50){
                    digitalWrite(diodaZielona, HIGH);
                    delay(2000);
                    digitalWrite(diodaZielona, LOW);
           }
          else if ((oswietlenie / 1024) * 100 < 75){
                    digitalWrite(diodaZielona, HIGH);
                    digitalWrite(diodaZolta, HIGH);
                    delay(2000);
                    digitalWrite(diodaZielona, LOW);
                    digitalWrite(diodaZolta, LOW);
           }
          else{
                    digitalWrite(diodaZielona, HIGH);
                    digitalWrite(diodaZolta, HIGH);
```

```
                        digitalWrite(diodaCzerwona, HIGH);
                        delay(2000);
                        digitalWrite(diodaZielona, LOW);
                        digitalWrite(diodaZolta, LOW);
                        digitalWrite(diodaCzerwona, LOW);
                }
        }
}
```

Stacja meteorologiczna z informacją wyświetlaną na ekranie komputera

W tej wersji programu informacja o aktualnych warunkach pogodowych nie jest przekazywana za pomocą diod, ale w postaci komunikatów wyświetlanych na ekranie komputera.

Aby umożliwić komunikację za pomocą portu szeregowego w funkcji „setup()” dołącz linijkę:

```
Serial.begin(9600);
```

Funkcja „loop()” powinna być zmodyfikowana następująco:

```
void loop(){
      odczytCzujnik = czujnik.read(sygnalCzujnik);
      oswietlenie = analogRead(A0);
      czyWcisnietoPrzyciskTemperatura =
digitalRead(przyciskTemperatura);
      if (czyWcisnietoPrzyciskTemperatura == HIGH){
            Serial.print("Aktualna temperatura: ");
            Serial.print(czujnik.temperature);
            Serial.println(" stopni Celsjusza");
            delay(500);
      }
      czyWcisnietoPrzyciskWilgotnosc =
digitalRead(przyciskWilgotnosc);
      if (czyWcisnietoPrzyciskWilgotnosc == HIGH){
            Serial.print("Aktualna wilgonosc: ");
            Serial.print(czujnik.humidity);
            Serial.println(" %");
            delay(500);
      }
      czyWcisnietoPrzyciskOswietlenie =
digitalRead(przyciskOswietlenie);
      if (czyWcisnietoPrzyciskOswietlenie == HIGH){
            Serial.print("Aktualna naswietlenie: ");
            Serial.print((oswietlenie/1024)*100);
            Serial.println(" %");
            delay(500);
      }
}
```

W kodzie programu usunięto wszystkie fragmenty odpowiedzialne za sterowanie diodami <digitalWrite()> i zastąpiono je funkcjami wyświetlającymi informacje na ekranie <Serial.print()> lub <Serial.println()>. Odczytywanie wartości mierzonych zostało przeniesione na początek funkcji „loop()".

Alarm po przekroczeniu wartości dopuszczalnych

Ponieważ w powyższym programie diody nie są używane, wykorzystajmy je, wraz z buzzerem, do informowania o przekroczeniu wartości dopuszczalnych (np. o zbyt wysokiej temperaturze). System alarmowy może być oczywiście dowolny, ten poniżej jest tylko przykładem. Dodaj go na końcu fukcji „loop()".

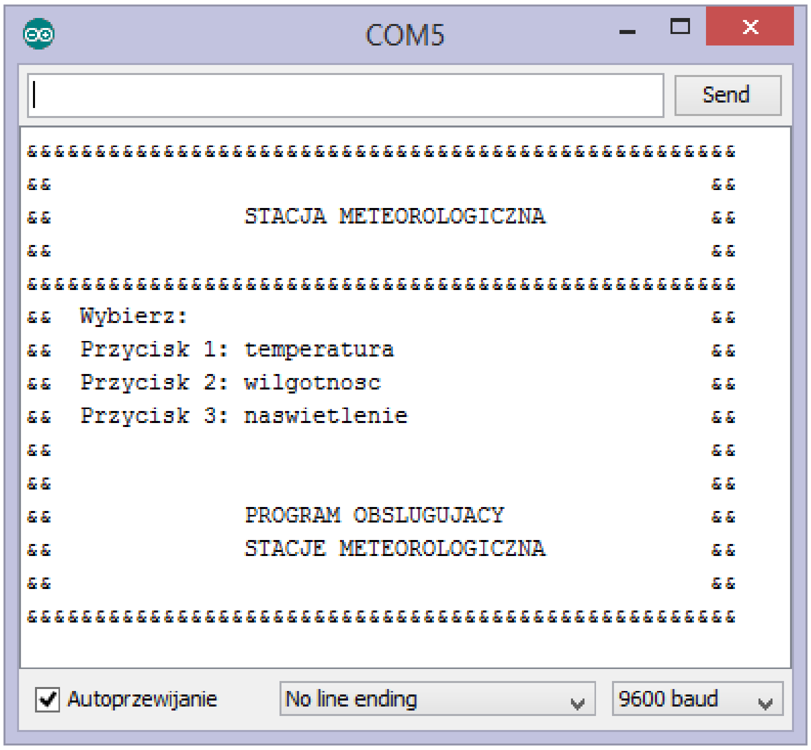
```
      if (czujnik.temperature > 30){
            digitalWrite(diodaCzerwona, HIGH);
            tone(buzzer,250);
            delay(100);
            digitalWrite(diodaCzerwona, LOW);
            noTone(buzzer);
            delay(100);
      }

      if (czujnik.humidity > 40){
            digitalWrite(diodaZolta, HIGH);
            tone(buzzer,500);
            delay(100);
            digitalWrite(diodaZolta, LOW);
            noTone(buzzer);
            delay(100);
}

      if ((oswietlenie / 1024) * 100 > 80){
            digitalWrite(diodaZielona, HIGH);
            tone(buzzer,1000);
            delay(100);
            digitalWrite(diodaZielona, LOW);
            noTone(buzzer);
            delay(100);
            }
      }
```

W zależności od tego, która z wartości została przekroczona, zostaje włączona odpowiadająca jej dioda i generowany jest dźwięk o określonej częstotliwości. Zestawienie wielkości mierzonych oraz odpowiadającym im dźwiękom i diodom zaprezentowano w tabeli poniżej.

Tabela 3. Zestawienie wielkości mierzonych oraz odpowiadającym im dźwiękom i diodom

Wielość mierzona	Dioda	Częstotliwość dźwięku [Hz]
Temperatura	Czerwona	250
Wilgotność	Żółta	500
Oświetlenie	Zielona	1000

Stacja meteorologiczna z prostym interfejsem użytkownika



Rysunek 54. Ekran powitalny stacji meteorologicznej.

Ostatnim etapem przygotowania stacji jest stworzenie prostego interfejsu użytkownika. Wygląda on podobnie jak ten wykorzystany w grze elektronicznej z poprzednich zajęć. Zwróć uwagę, że w tym przypadku aplikacja zawiera także ekran powitalny, którego kod umieszony jest w funkcji „setup()”, a więc wykonywany jest tylko raz. Na ekranie powitalnym wyświetlany jest tekst „PROGRAM OBSLUGUJACY STACJE METEOROLOGICZNA”, później w miejscu tego tekstu wyświetlane będą żądane wartości mierzone.

```
#include <dht11.h>
dht11 czujnik;
int sygnalCzujnik = 7;
int odczytCzujnik = 0;
double oświetlenie = 0;

int przyciskTemperatura = 3;
int przyciskWilgotnosc = 4;
int przyciskOswietlenie = 5;

int diodaCzerwona = 10;
int diodaZolta = 11;
```

```
int diodaZielona = 12;

int czyWcisnietoPrzyciskTemperatura= 0;
int czyWcisnietoPrzyciskWilgotnosc = 0;
int czyWcisnietoPrzyciskOswietlenie = 0;

void setup(){
      pinMode(diodaCzerwona, OUTPUT);
      pinMode(diodaZolta, OUTPUT);
      pinMode(diodaZielona, OUTPUT);
      pinMode(przyciskTemperatura, INPUT);
      pinMode(przyciskWilgotnosc, INPUT);
      pinMode(przyciskOswietlenie, INPUT);
      Serial.begin(9600);

      Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
      &&&");
      Serial.println("&&\t\t\t\t\t\t  &&");
      Serial.println("&&\t\tSTACJA METEOROLOGICZNA\t\t  &&");
      Serial.println("&&\t\t\t\t\t\t  &&");
      Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
      &&&");
      Serial.println("&&  Wybierz:\t\t\t\t\t  &&");
      Serial.println("&&  Przycisk 1: temperatura \t\t\t  &&");
      Serial.println("&&  Przycisk 2: wilgotnosc \t\t\t  &&");
      Serial.println("&&  Przycisk 3: naswietlenie \t\t\t  &&");
      Serial.println("&&\t\t\t\t\t\t  &&");
      Serial.println("&&\t\t\t\t\t\t  &&");
      Serial.println("&&\t\t\t\t\t\t  &&");
      Serial.println("&&\t\t\t\t\t\t  &&");
      Serial.println("&&\t\t\t\t\t\t  &&");
      Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
      &&&");
 }

void loop() {
      if (digitalRead(przyciskTemperatura) == HIGH) {
            odczytCzujnik = czujnik.read(sygnalCzujnik);

            Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
            &&&&&&&&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&\t\tSTACJA METEOROLOGICZNA\t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
            &&&&&&&&");
            Serial.println("&&  Wybierz:\t\t\t\t\t  &&");
            Serial.println("&&  Przycisk 1: temperatura \t\t\t  &&");
            Serial.println("&&  Przycisk 2: wilgotnosc \t\t\t  &&");
            Serial.println("&&  Przycisk 3: naswietlenie \t\t\t  &&");
```

```
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.print("&&  Aktualna temperatura: ");
            Serial.print(czujnik.temperature);
            Serial.println(" stopni Celsjusza \t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
            &&&&&&&&");
            delay(2000);
      }

      else if (digitalRead(przyciskWilgotnosc) == HIGH) {
            odczytCzujnik = czujnik.read(sygnalCzujnik);
            Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
            &&&&&&&&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&\t\tSTACJA METEOROLOGICZNA\t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
            &&&&&&&&");
            Serial.println("&&  Wybierz:\t\t\t\t\t  &&");
            Serial.println("&&  Przycisk 1: temperatura \t\t\t  &&");
            Serial.println("&&  Przycisk 2: wilgotnosc \t\t\t  &&");
            Serial.println("&&  Przycisk 3: naswietlenie \t\t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.print("&&  Aktualna wilgonosc: ");
            Serial.print(czujnik.humidity);
            Serial.println(" %  \t\t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
            &&&&&&&&");
            delay(2000);
      }

      else if (digitalRead(przyciskOswietlenie) == HIGH) {
            oswietlenie = analogRead(A0);
            Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
            &&&&&&&&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&\t\tSTACJA METEOROLOGICZNA\t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
            &&&&&&&&");
            Serial.println("&&  Wybierz:\t\t\t\t\t  &&");
            Serial.println("&&  Przycisk 1: temperatura \t\t\t  &&");
            Serial.println("&&  Przycisk 2: wilgotnosc \t\t\t  &&");
            Serial.println("&&  Przycisk 3: naswietlenie \t\t\t  &&");
```

```
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.print("&&   Aktualna naswietlenie: ");
            Serial.print((oswietlenie/1024)*100);
            Serial.println(" %  \t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&\t\t\t\t\t\t  &&");
            Serial.println("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
            &&&&&&&&");
            delay(2000);
      }
      if (czujnik.temperature > 30){
            digitalWrite(diodaCzerwona, HIGH);
            tone(buzzer,250);
            delay(100);
            digitalWrite(diodaCzerwona, LOW);
            noTone(buzzer);
             delay(100);
      }

      if (czujnik.humidity > 40){
            digitalWrite(diodaZolta, HIGH);
            tone(buzzer,500);
            delay(100);
            digitalWrite(diodaZolta, LOW);
            noTone(buzzer);
            delay(100);
      }

      if ((oswietlenie / 1024) * 100 > 80){
            digitalWrite(diodaZielona, HIGH);
            tone(buzzer,1000);
            delay(100);
            digitalWrite(diodaZielona, LOW);
            noTone(buzzer);
            delay(100);
            }
      }
}
```

Literatura

1. Bolkowski Stanisław, "*Elektrotechnika*", WSiP 2004.
2. Boxall John, Arduino. „*65 praktycznych projektów*”, Helion 2013.
3. Górecki Piotr, „*Mikrokontrolery dla początkujących*”, Wydawnictwo btc, 2006.
4. Igoe Tom, „*Spraw, by rzeczy przemówiły. Programowanie urządzeń elektronicznych z wykorzystaniem Arduino*”, Helion 2013.
5. Jukiewicz Marcin, „*Elektronika Analogowa. Podręcznik Ucznia*”, dokumentacja projektu HIGH-TECHnika.
6. Monk Simon, „*Arduino dla początkujących*”, Helion 2014.
7. Platt Charles, "*Elektronika. Od praktyki do teorii*", Helion, 2012.
8. Shamieh Cathleen, McComb Gordon, “*Elektronika dla bystrzaków*”, Helion, 2012.