PRAKTYCZNY PEDAGOG

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

MATERIAŁY SZKOLENIOWE

UKŁADY AUTOMATYKI CYFROWEJ

mgr inż. Michał Straus

SPIS TREŚCI:

1. Platforma Arduino

1.1 Arduino program i zasady jego działania

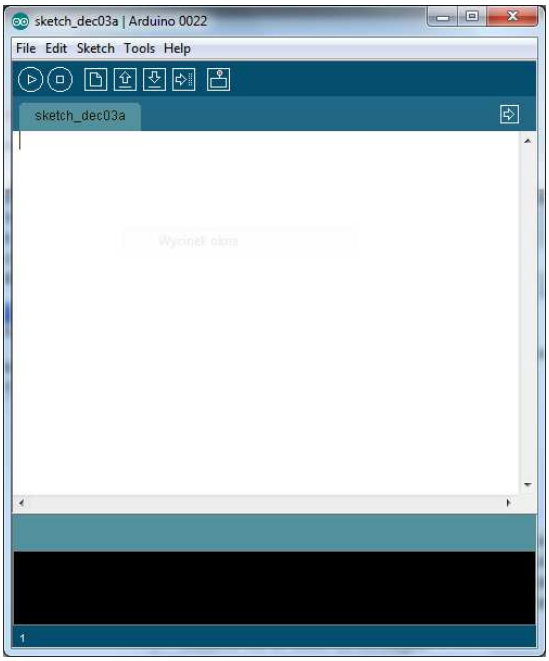
Proces uruchomienia układu Arduino wymaga oprócz samego modułu Arduino odpowiedniego oprogramowanie, które pozwoli nam sterować tym modułem. Jest ono otwarte i dostępne pod tym adresem: *http://arduino.cc/en/Main/Software*. Wersja dla Windows zawiera wszystkie niezbędne narzędzia. Jedynym mankamentem jest potrzeba instalacji sterowników dla układu USB w Arduino.

1.2 Instalacja środowiska

Pobierz najnowszą wersję środowiska programistycznego (IDE) Arduino z oficjalnej strony projektu: http://arduino.cc/en/Main/Software
Pobrane archiwum należy wypakować do wybranego katalogu, np. C:\Arduino.
Podłącz płytkę Arduino do komputera. System spróbuje zainstalować sterownik, instalacja zakończy się niepowodzeniem.
Otwórz menedżera urządzeń. W kategorii „Porty (COM i LPT) powinno widoczne być urządzenie „Arduino UNO (COMxx)".
Kliknij prawym przyciskiem na urządzeniu, z menu kontekstowego wybierz „Aktualizuj oprogramowanie sterownika".
Następnie „Przeszukaj mój komputer w poszukiwaniu oprogramowania sterownika".
Wskaż plik sterownika „Arduino UNO.inf" znajdujący się w podkatalogu drivers oprogramowania Arduino (w naszym przypadku C:\Arduino\drivers\Arduino UNO.inf).
Sterownik zostanie poprawnie zainstalowany.

1.2 Pierwsze kroki z IDE Arduino

Uruchom IDE Arduino (C:\Arduino\arduino.exe).

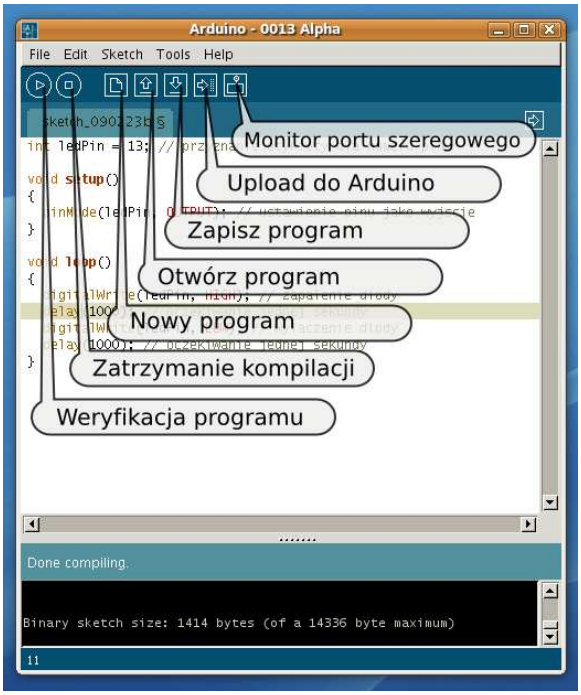


Ustawienie typu posiadanego urządzenia:

Z menu programu wybierz Tools->Board->Arduino Uno

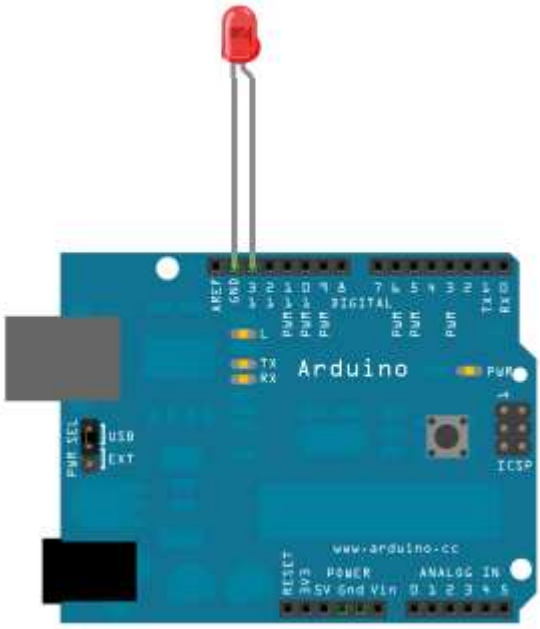
Ustawienie portu na którym podłączone jest urządzenie:

Do komputera może być podłączone kilka urządzeń Arduino, jednak w danej chwili można obsługiwać tylko jedno z nich. Wyboru urządzenia dokonuje się poprzez wybór portu COM pod którym widziane jest urządzenie. Domyślnie wybrane jest pierwsze podpięte urządzenie. Zmiana portu Tools->Serial Port->Cmxx



1.4 Ćwiczenia z Arduino

Ćwiczenie 1. Wprowadzenie do układów wbudowanych - sterowanie diodą



Rysunek 1 Schemat połączeń

Wymagania: Arduino, dioda

Na płytce Arduino pomiędzy pinem 13 a masą umieszczona jest czerwona dioda, którą można wykorzystać w tym ćwiczeniu, alternatywnie można wpiąć inna diodę pomiędzy dowolny pinem cyfrowym a masą.

Załaduj przykładowy program „Blink”, z menu programu wybierz File->Examples->Basisc->Blink. Kod programu składa się z dwóch funkcji: setup oraz loop. Funkcja setup jest wykonywana tylko raz po podłączeniu zasilania, natomiast kod funkcji loop jest wykonywany w pętli.

Kod programu:

```
void setup() {
  // Inicjalizacja pinu 13 jako pin wyjściowy.
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // Ustawienie stanu wysokiego na pinie 13
  delay(1000);              // Odczekanie 1000ms
  digitalWrite(13, LOW);    // Ustawienie stanu niskiego na pinie 13
  delay(1000);              // Odczekanie 1000ms
}
```

Ćwiczenie 2. Obsługa wyświetlacza 7 segmentowego

Wymagania: Arduino, wyświetlacz 7 segmentowy, oporniki, płytka stykowa

Wyświetlacz 7 segmentowy składa z siedmiu diód umieszczonych w jednej obudowie (czasem jest to osiem diod – wyświetlacze z kropką). Wyświetlacze te dzielą się na dwa typy: ze wspólna anodą lub wspólną katodą.

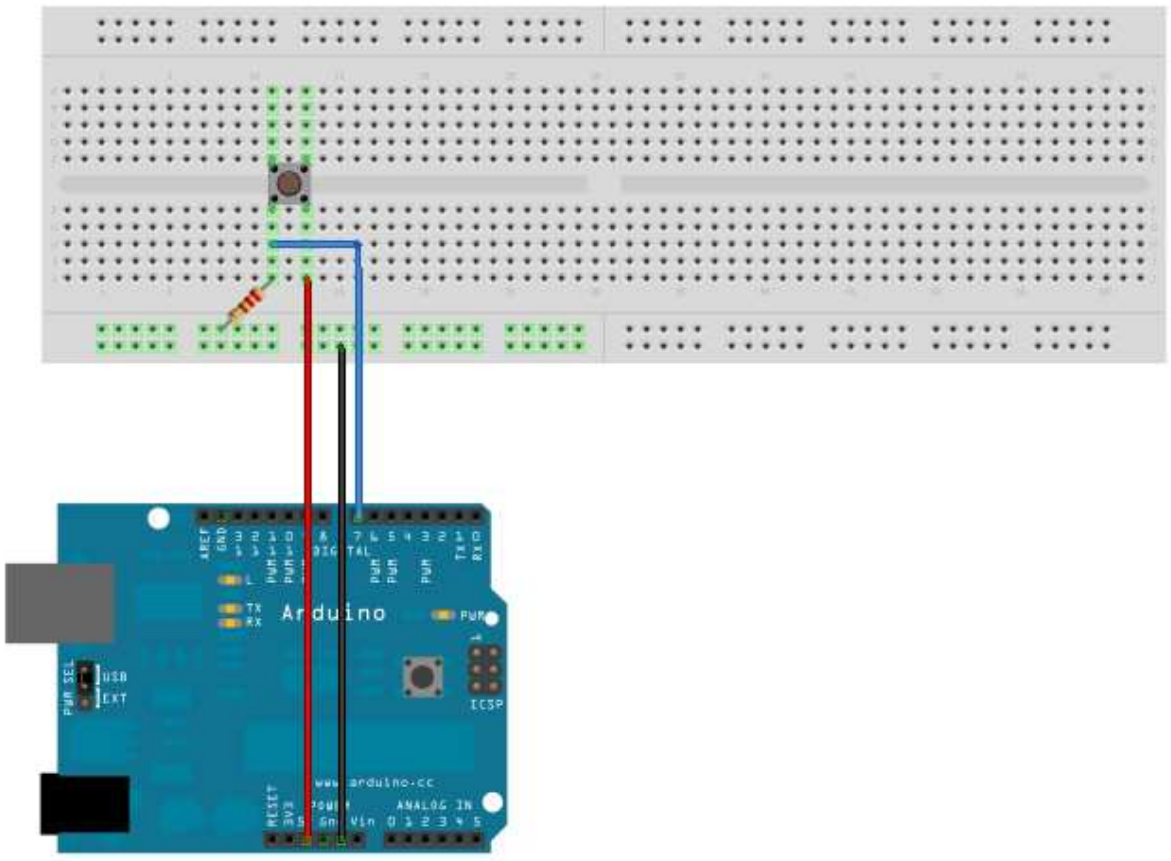
Należy ustalić jakiego typu jest posiadany wyświetlacz (napisy z obudowy, specyfikacja w Internecie)

Każdy segmentów połączy przez opornik z jednym z wybranych pinów. Pin wejściowy wyświetlacza do napięcia lub masy. Napisać program wyświetlający kolejne cyfry w pętli (analogicznie do ćwiczenia 1)

Ćwiczenie 3. Odczytanie stanu przycisku

Wymagania: Arduino, płytka stykowa, opornik, przycisk

Podłącz elementy wg poniższego schematu. Pin 7 jest połączony z jednym z pinów przycisku oraz z masą poprzez opornik. Drugi pin przycisku jest połączony z zasilaniem. W momencie naciśnięcia przycisku na pinie 7 pojawi się stan wysoki.



Rysunek 2 Schemat połączeń

Program ma za zadanie zapalenie czerwonej diody w momencie gdy przycisk zostanie użyty.

Kod programu:

```
const int buttonPin = 7;     // numer pinu pod który podpięty jest przycisk
```

```
const int ledPin =  13;      // numer pinu pod który podłączona jest dioda

int buttonState = 0;         // zmienna w której będzie przechowywany stan przycisku

void setup() {
  // inicjalizacja pinu diodu jako pinu wyjściowego
  pinMode(ledPin, OUTPUT);
  // inicjalizacja pinu przycisku jako pinu wejściowego
  pinMode(buttonPin, INPUT);
}

void loop(){
  // odczyt stanu przycisku
  buttonState = digitalRead(buttonPin);

  // jeżeli przycisk jest wciśnięty
  if (buttonState == HIGH) {
    // zapal diodę
    digitalWrite(ledPin, HIGH);
  }
  // w przeciwnym razie
  else {
    // wyłącz diodę
    digitalWrite(ledPin, LOW);
  }
}
```

Ćwiczenie 4. Obsługa znakowego wyświetlacza LCD

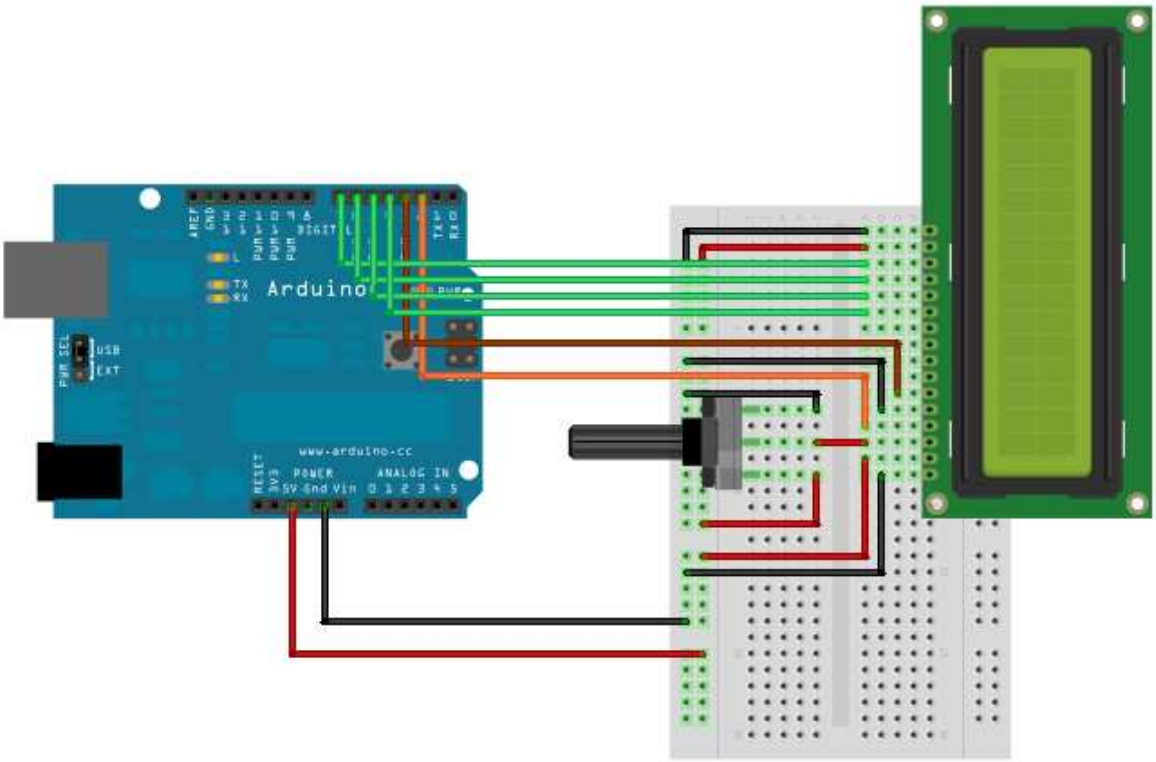
Wymagania: Arduino, wyświetlacz znakowy LCD, potencjometr

Znajdź specyfikację techniczną posiadanego wyświetlacza. Większość wyświetlaczy znakowych opiera się na standardzie HD44780, jednak wyprowadzenia pinów mogą się różnić. Zazwyczaj wyprowadzenie jest następujące:

1. Ground
2. VCC (+3.3 to +5V)
3. Contrast adjustment (VO)
4. Register Select (RS). RS=0: Command, RS=1: Data
5. Read/Write (R/W). R/W=0: Write, R/W=1: Read
6. Clock (Enable). Falling edge triggered
7. Bit 0 (Not used in 4-bit operation)
8. Bit 1 (Not used in 4-bit operation)

9. Bit 2 (Not used in 4-bit operation)
10. Bit 3 (Not used in 4-bit operation)
11. Bit 4
12. Bit 5
13. Bit 6
14. Bit 7
15. Backlight Anode (+)
16. Backlight Cathode (-)

W 4 bitowym trybie pracy do obsługi wyświetlacza wystarczy 6 pinów. Dodatkowo należy podłączyć zasilanie i ustawić kontrast przy pomocy potencjometru.
Podłącz elementy wg poniższego schematu



Program ma za zadanie wyświetlenie tekstu „Hello world”
Kod programu:

```
// dołączenie biblioteki lcd
#include <LiquidCrystal.h>

// inicjalizacja biblioteki – ustawienie odpowiednich pinów
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

void setup() {
```

```
  // ustawienie ilości znaków i wierszy
  lcd.begin(16, 2);
  // wyświetlenie wiadomości
  lcd.print("hello, world!");
}

void loop()
{}
```

Program ma za zadanie wyświetlenie „uciekającego” znaku „x”

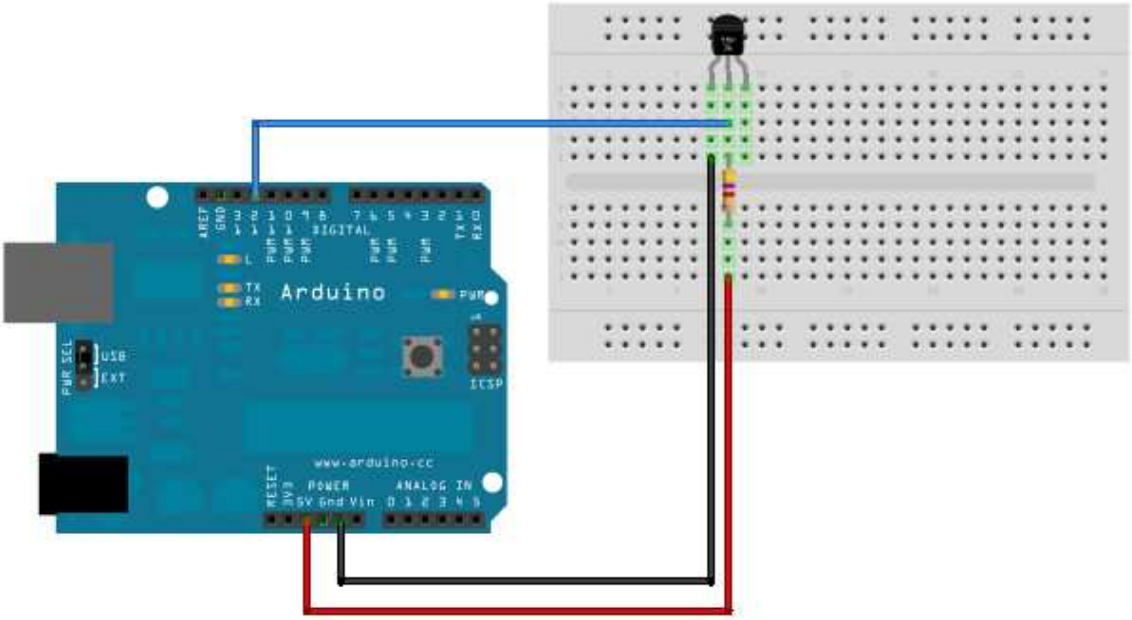
Kod programu:

```
// dołączenie biblioteki lcd
#include <LiquidCrystal.h>

// inicjalizacja biblioteki – ustawienie odpowiednich pinów
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

// pozycja na wyświetlaczu
int pos = 0;

void setup() {
  // ustawienie ilości znaków i wierszy
  lcd.begin(16, 2);
  // ustawienie pozycji na początek pierwszego wiersza
  lcd.setCursor(pos, 0);
}

void loop()
{
  // tekst do wyświetlenia, 16 spacji
  String txt = "                ";
  // ustawienie znaku x
  txt[pos] = 'x';
  // wyświetlenie tekstu
  lcd.print(txt);
  // zmiana pozycji x
  pos++;
  // pozycja x nie może być większa niż ilość znaków w wierszu
  pos =  pos % 16;
  // odczekanie 200ms przed kolejnym wyświetleniem
  delay(200);
}
```

Ćwiczenie 5. Pomiar temperatury czujnikiem cyfrowym

Wymagania: Arduino, czujnik temperatury Dallas 18B20

Czujnik temperatury firmy Dallas komunikuje się za pomocą protokołu 1-wire. Protokół ten pozwala na złączenie kilku urządzeń jednym kablem i komunikację między nimi. Arduino standardowo posiada bibliotekę do obsługi protokołu 1-wire.



Program ma za zadanie odczytanie temperatury z czujnika.

Kod programu:

```
// załadowanie bibilioteki do obsługi 1-wire
#include <OneWire.h>

// inicjalizacja biblioteki 1-wire, komunikacja protokołem 1-wire będzie odbywać się poprzed pin 12
OneWire ds(12);
int i;
void setup() {
  // inicjalizacja pinu 13 jako pinu wyjściowego
  pinMode(13, OUTPUT);
  // inicjalizacja portu serial z prędkością 9600
  Serial.begin(9600);

  i = 0;
}
```

```
void loop() {
  byte i;
  byte present = 0;
  byte data[12];
  byte addr[8];

  // wyszukiwanie urządzeń
  if ( !ds.search(addr)) {
      Serial.print("No more addresses.\n");
      ds.reset_search();
      delay(5000);
      return;
  }

  Serial.print("R=");
  for( i = 0; i < 8; i++) {
    Serial.print(addr[i], HEX);
    Serial.print(" ");
  }

  if ( OneWire::crc8( addr, 7) != addr[7]) {
      Serial.print("CRC is not valid!\n");
      return;
  }

  if ( addr[0] == 0x10) {
      Serial.print("Device is a DS18S20 family device.\n");
  }
  else if ( addr[0] == 0x28) {
      Serial.print("Device is a DS18B20 family device.\n");
  }
  else {
      Serial.print("Device family is not recognized: 0x");
      Serial.println(addr[0],HEX);
      return;
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44,1);

  delay(1000);
```

```
  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);

  Serial.print("P=");
  Serial.print(present,HEX);
  Serial.print(" ");
  for ( i = 0; i < 9; i++) {
    data[i] = ds.read();
    Serial.print(data[i], HEX);
    Serial.print(" ");
  }
  Serial.print(" CRC=");
  Serial.print( OneWire::crc8( data, 8), HEX);
  Serial.println();

  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
  Serial.println(i++, DEC);

  int Temp=(data[1]<<8)+data[0];

  float fTemp = Temp / 16.0;
  Serial.println("Temp: ");
  Serial.print(fTemp);
  Serial.println("");
}
```

Ćwiczenie 6. Sterowanie brzęczykiem przez WWW.

Wymagania: Arduino, Ethernet Shield, brzęczyk

Prezentowany program ma zadanie umożliwienie sterowania brzęczykiem (tzw. buzer) przez przeglądarkę internetową.

Kod programu:

```
#include "Ethernet.h"
#include "WebServer.h"

// MAC adres kart sieciowej
```

```
static uint8_t mac[6] = { 0x02, 0xAA, 0xBB, 0xCC, 0x00, 0x22 };

// Adres IP
static uint8_t ip[4] = { 192, 168, 42, 51 };

#define PREFIX "/buzz"
/* nasłuch na standardowym porcie http – 80 */
WebServer webserver(PREFIX, 80);

/* pin na którym podpięty jest brzęczyk */
#define BUZZER_PIN 3

/* wartość określa czas w milisekundach po jakim brzęczyk zostanie wyłączony */
int buzzDelay = 0;

/* określa czy brzęczyk ma być uruchomiony w danym przebiegu pętli */
char toggle = 0;

/* funkcja obsługuje zapytania do serwera
Dla GET zwraca stronę z interfejsem do obsługi buzera
Dla żądań typu POST zapamiętuje wartość parametru buzzDelay
*/
void buzzCmd(WebServer &server, WebServer::ConnectionType type)
{
    /* żadanie typu POST */
  if (type == WebServer::POST)
  {
    bool repeat;
    char name[16], value[16];
    do
    {
      /* wczytuje parametry żądania. */
      repeat = server.readURLParam(name, 16, value, 16);

      /* sprawdza nazwę parametru */
      if (strcmp(name, "buzz") == 0)
      {
       /* zamienia tekst na integer i zapisuje w zmiennej buzzDelay */
        buzzDelay = strtoul(value, NULL, 10);
      }
    } while (repeat);
```

```
    server.httpSeeOther(PREFIX);
    return;
  }

  server.httpSuccess();

  if (type == WebServer::GET)
  {
    /* zawiera kod HTML strony */
    P(message) =
      "<html><head><title>Webduino Buzzer Example</title>"
      "<body>"
      "<h1>Test the Buzzer!</h1>"
      "<form action='/buzz' method='POST'>"
      "<p><button name='buzz' value='0'>Turn if Off!</button></p>"
      "<p><button name='buzz' value='500'>500</button></p>"
      "<p><button name='buzz' value='1975'>1975</button></p>"
      "<p><button name='buzz' value='3000'>3000</button></p>"
      "<p><button name='buzz' value='8000'>8000</button></p>"
      "</form></body></html>";
    /* wyświetla stronę html */
    server.printP(message);
  }
}

void setup()
{
  // ustawia pin buzera jako pin wyjściowy
  pinMode(BUZZER_PIN, OUTPUT);

  // inicjuje Ethernet shield
  Ethernet.begin(mac, ip);

  webserver.setDefaultCommand(&buzzCmd);

  /* uruchom serwer */
  webserver.begin();
}

void loop()
{
  // Obsługuj nadchodzące połączenia w pętli
```

```
  webserver.processConnection();

  /* jeżeli opóźnienie brzęczyka jest ustawione na większe od zera
       Włącz i wyłacz brzęczyk */
  if ((++toggle & 1) && (buzzDelay > 0))
  {
    digitalWrite(BUZZER_PIN, HIGH);
    delayMicroseconds(buzzDelay);
    digitalWrite(BUZZER_PIN, LOW);
  }
}
```

1.5 Tematy projektów na zaliczenie

Tematy przykładowych projektów do wykonania na zaliczenie w grupach. Ilość gwiazdek przy temacie określa stopień trudności projektu. Podkreślone tematy oparte są o platformę Arduino.

1. Zegar cyfrowy: wyświetlacz 7 segmentowy + 2 przyciski do ustawienia ***
2. Zegar cyfrowy. wyświetlacz 7 segmentowy + Aktualizacja czasu protokołem NTP ***
3. Prosta gra. Obsługa wyświetlacza 2x16 znaków ***
4. Prosta gra. Obsługa wyświetlacza Nokia 3100 ***
5. Odbiornik podczerwieni. Montaż wg schematu, uruchomienie pod Windows, pomiar pilota IR **
6. Odbiornik podczerwieni. Montaż wg schematu, uruchomienie pod Linux, architektura mips, pomiar pilota IR ****
7. *Gotowy odbiornik podczerwieni. Uruchomienie pod Linux, pomiar pilota IR **
8. Sieciowy czujnik temperatury. Pomiar tempertury i przesłanie danych przez sieć lub udostępnienie na “serwerze www” **
9. Pomiar temperatury przez port RS pod Linuxem, architektura mips ***
10. Serwer odpowiadający na pingi z ustalonym opóźnieniem *
11. Serwer odpowiadający na pingi z ustalonym opóźnieniem. Logowanie na kartę SD **
12. Własna propozycja