Wprowadzenie do programowania urządzeń Arduino

(Arduino dla Informatyków)

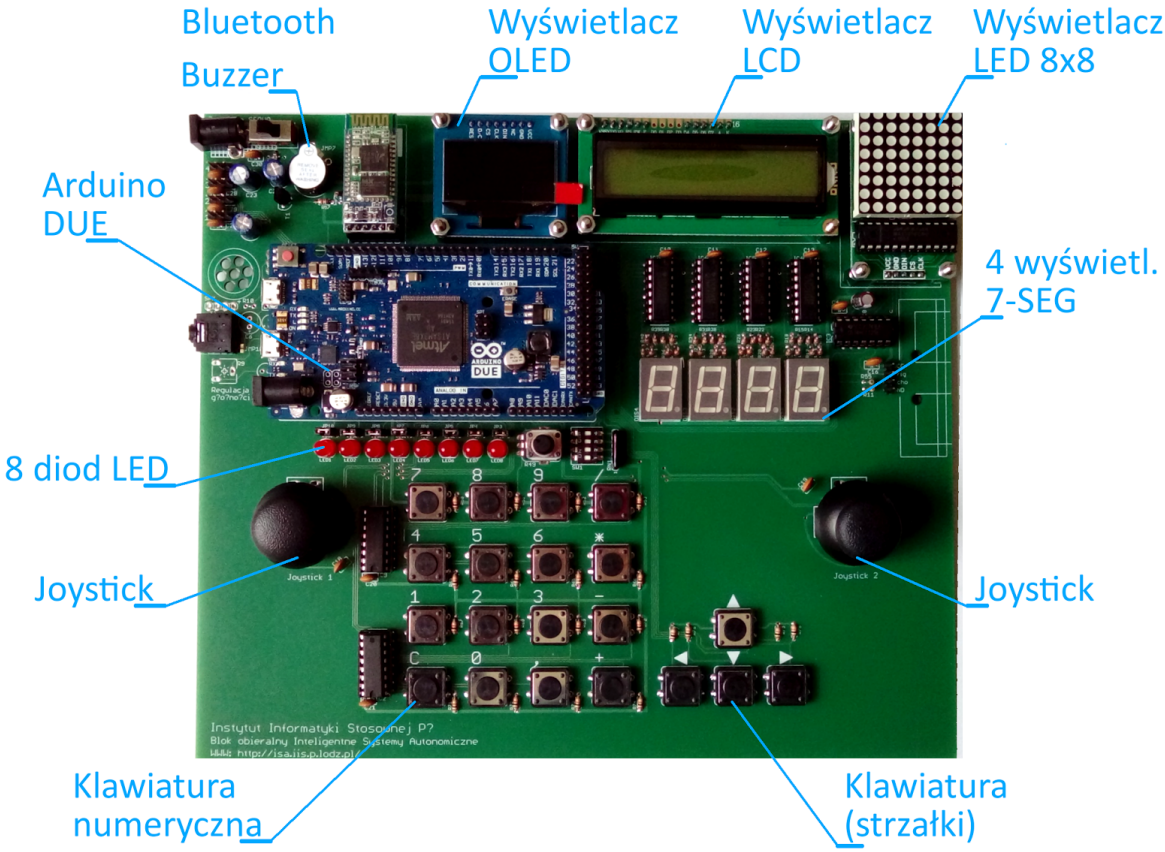
Zajęcia pilotażowe z Arduino

Podstawy Programowania 2 / Systemy Operacyjne 2

Autor: **Piotr Duch, Tomasz Jaworski**
Instytut Informatyki Stosowanej
Politechnika Łódzka

Kontakt: pduch@is.p.lodz.pl, tjaworski@iis.p.lodz.pl

Stanowisko laboratoryjne



Wyświetlacz LCD

Przydatne funkcje

Biblioteka **ISALiquidCrystal** odpowiada za obsługę wyświetlacza LCD zamontowanego na płytce edukacyjnej IIS. Jest to wyświetlacz alfanumeryczny, o rozmiarze 2-óch wierszy i 16-stu kolumn. Klasa **ISALiquidCrystal** odpowiadająca za obsługę wyświetlaczy LCD, dostępna po dołączeniu do programu pliku **LiquidCrystal.h**. W swoim programie możesz mieć tylko **jeden** obiekt klasy **ISALiquidCrystal** .

Metody dostępne w ramach klasy ISALiquidCrystal:

- *lcd.begin()* - uruchamia (inicjuje) sterownik wyświetlacza LCD. Należy ją wywołać w funkcji *setup()*. *lcd* - obiekt klasy **ISALiquidCrystal.**
- *lcd.print(data)* - metoda wyświetla wiadomość przekazaną w parametrze *data* na wyświetlaczu LCD. Dane do wyświetlenia mogą być następujących typów: **char**, **byte**, **int**, **long** lub **String**. *data* - dane, które mają zostać wyświetlone na wyświetlaczu LCD.
- *lcd.setCursor(col, row)* - metoda do ustawiania kursora w zadanej pozycji. Parametr *col* - numer kolumny, w której ma zostać ustawiony kursor, *row* - numer wiersza, w którym ma zostać ustawiony kursor. **Wiersze i kolumny są numerowane od 0.**
- *lcd.clear()* - metoda czyści ekran wyświetlacza LCD i ustawia kursor w lewym górnym rogu.
- *delay(ms)* - umożliwia zatrzymanie aktualnie wykonywanego programu na określony czas. *ms* - czas w milisekundach, na jaki ma zostać zatrzymany program.

Przykład - program wyświetlający napis “hello, world!” na wyświetlaczu LCD:

```
#include <ISALiquidCrystal.h>
ISALiquidCrystal lcd;
void setup()
{
  lcd.begin();
  lcd.print("hello, world!");
}
void loop() {}
```

Zadanie 1.

Napisz program, który wyświetli na wyświetlaczu LCD czas, jaki minął od uruchomienia programu, w formacie mm:ss. Pamiętaj o uzupełnianiu liczby nieznaczącymi zerami przed wyświetleniem. (Wykorzystaj funkcję *delay*()).

Diody

Przydatne funkcje i symbole:

- *pinMode(pin, mode)* - umożliwia określenie trybu pracy poszczególnych pinów. *pin* - numer pinu, którego tryb pracy ma być ustawiony, *mode* - tryb pracy pinu, może przyjąć jedną z dwóch wartości: **OUTPUT** lub **INPUT**.
    - Stała **INPUT** oznacza, że pin będzie pracował w trybie wejścia, co pozwoli uruchomionemu programowi wczytywać informacje od dołączonego do pinu *pin* urządzenia (np. przycisk strzałki w górę).
    - Stała **OUTPUT** oznacza, że pin będzie pracował w trybie wyjścia, co pozwoli uruchomionemu programowi na sterowanie urządzeniem podłączonym do danego pinu, np. włączaniem/wyłączaniem brzęczyka lub diody LED.
- *digitalWrite(pin, value)* - umożliwia ustawienie stanu pinu. *pin* - numer pinu, którego stan ma zostać ustawiony, *value* - stan pinu, może przyjąć jedną z dwóch wartości: **HIGH** lub **LOW**
    - Stała **HIGH** odpowiada logicznej wartości prawdy (1) w języku C (np. brzęczyk włączony).
    - Stała **LOW** odpowiada logicznej wartości fałszu (0) w języku C (np. brzęczyk wyciszony).
- Płytka edukacyjna posiada osiem diod LED, oznaczonych symbolami **LED1**, **LED2**, …, **LED7**, **LED8**. Dostępna jest również tablica **LEDS**[], pozwalająca na odwoływanie się do odpowiedniej diody za pomocą wyrażenia indeksującego, np. kod *int i* = 4; *LEDS*[i] oznacza diodę **LED5**. Wszystkie te symbole dostępne są po dołączeniu pliku nagłówkowego **ISADefinitions.h.**

Przykład - program włączający diodę LED1:

```
#include <ISADefinitions.h>

void setup(){
  pinMode(LED1, OUTPUT);
  digitalWrite(LED1, HIGH);
}
void loop(){}
```

Zadanie 2.

Napisz program, który będzie włączał kolejne diody, wyłączając poprzednie w ten sposób, że włączona dioda będzie się najpierw przesuwała z lewej strony w prawą (na skraj linijki), a następnie w przeciwnym kierunku. W każdej chwili ma być włączona dokładnie jedna dioda.

Przyciski

Przydatne funkcje:

Biblioteka **ISAButtons** odpowiada za obsługę przycisków (obecnie 16 przycisków klawiatury numerycznej, przyciski numerowane są od 0 do 15), dostępna po dołączeniu do programu pliku **ISAButtons.h**. W swoim programie możesz mieć tylko **jeden** obiekt klasy **ISAButtons**.

Metody dostępne w ramach klasy ISAButtons:

- *buttons.init()* - metoda inicjalizująca przyciski. Należy ją wywołać w funkcji *setup()*. *buttons* - obiekt klasy **ISAButtons**.
- *buttons.buttonPressed(buttonId)* - metoda sprawdza, czy przycisk o podanym indeksie został przyciśnięty i zwraca **true** jeżeli przycisk był wciśnięty, a **false** w przeciwnym wypadku. *buttons* - obiekt klasy **ISAButtons**, *buttonId* - id przycisku, którego stan ma zostać sprawdzony. Stan przycisku odświeżany jest co 25 milisekund.
- *buttons.buttonReleased(buttonId)* - metoda sprawdza czy przycisk o podanym indeksie został zwolniony, zwraca **true** jeżeli przycisk był zwolniony, a **false** w przeciwnym wypadku. *buttons* - obiekt klasy **ISAButtons**, *buttonId* - id przycisku, którego stan ma zostać sprawdzony. Stan przycisku odświeżany jest co 25 milisekund.
- *buttons.buttonState(buttonId)* - metoda sprawdza stan przycisku o podanym indeksie, zwraca **true** jeżeli przycisk jest wciśnięty, a **false** w przeciwnym wypadku. *buttons* - obiekt klasy **ISAButtons**, *buttonId* - id przycisku, którego stan ma zostać sprawdzony. Stan przycisku odświeżany jest co 25 milisekund.

Klawiatura strzałek

Dodatkowo dostępne są klawisze strzałek. Nie są one jednak obsługiwane przez klasę **ISAButtons** (która służy do odczytu klawiszy numerycznych). Aby odczytywać stany tych klawiszy, należy skorzystać z funkcji *pinMode* oraz *digitalRead*. Same klawisze dostępne są pod symbolami **KEY_UP**, **KEY_DOWN**, **KEY_RIGHT** oraz **KEY_LEFT**. Istnieje również tablica klawiszy strzałek - **KEY_ARROWS**[]. Symbole dostępne są w pliku nagłówkowym **ISADefinitions.h**.

Uwaga! Klawisze strzałek działają w logice odwrotnej - funkcja *digitalRead* dla klawisza wciśniętego zwraca *false* a dla puszczonego zwraca *true*. Należy skorzystać z negacji (!).

Przykład - sprawdzenie stanu przycisku z biblioteką ISAButtons:

```
#include <ISADefinitions.h>
#include <ISAButtons.h>
ISAButtons button;
bool state = false;
void setup(){
  pinMode(LED1, OUTPUT);
  button.init();
```

```
}
void loop(){
  if (button.buttonPressed(0)) {
     state = !state;
     digitalWrite(LED1, state);
  }
  delay(30);
}
```

Przykład - sprawdzenie stanu klawisza “w górę”:

```
#include <ISADefinitions.h>

void setup(){
  pinMode(KEY_UP, INPUT);
  pinMode(LED1, OUTPUT);
}
void loop(){
  bool up = !digitalRead(KEY_UP);
  digitalWrite(LED1, up);
}
```

Zadanie 3.

Wykorzystując zestaw dwóch przycisków przesuwaj włączoną diodę w lewo lub w prawo. W każdej chwili ma być włączona dokładnie jedna dioda.

Potencjometr i Joysticki

Przydatne funkcje:

- *analogRead(pin)* - umożliwia odczytanie wartości z pinów będących wejściem analogowym. Zwraca wartość całkowitą z zakresuu 0 - 1023. *pin* - numer pinu, z którego ma być odczytana wartość.
- *analogWrite(pin, value)* - umożliwia przypisanie wartości analogowej, która w przypadku pinu cyfrowego jest realizowana za pomocą sygnału PWM. Może być wykorzystywana do ustawiania jasności diody lub prędkości obracania się silnika. *pin* - numer pinu, do którego ma być przypisana wartość, *value* - wartość z zakresu 0 – 255, gdzie 0 odpowiada diodzie wyłączonej, a 255 świecącej z maksymalną jasnością.
- Potencjometr oraz joysticki dostępne są za pomocą symboli z pliku nagłówkowego **ISADefinitions.h**. Symbole te to: **POT** (potencjometr), **JOY1X**, **JOY1Y** (joystick 1, osie X oraz Y), **JOY2X**, **JOY2Y** (joystick 2, osie X oraz Y).

Przykład - program odczytujący aktualną wartość potencjometru i wyświetlający ją na wyświetlaczu LCD:

```
#include <ISADefinitions.h>
#include <ISALiquidCrystal.h>
ISALiquidCrystal lcd;
void setup(){
     lcd.begin();
}
void loop()
{
     lcd.clear();
     int pot = analogRead(POT);
     lcd.print(pot);
     delay(250);
}
```

Zadanie 4.

Wykorzystując potencjometr przesuwaj włączoną diodę w lewo lub w prawo (zwróć uwagę, że funkcja *analogRead* zwraca wartości z przedziału 0 - 1023).

Zadanie 5.

Napisz program, który będzie wyświetlał na wyświetlaczu LCD bieżące wychylenia Joysticków.

Wyświetlacz siedmiosegmentowy

Przydatne funkcje:

Biblioteka **ISA7SegmentDisplay** odpowiada za obsługę wyświetlaczy siedmiosegmentowych. Dostępna jest po dołączeniu do programu pliku **ISA7SegmentDisplay.h**.

Metody dostępne w ramach klasy ISA7SegmentDisplay:

- *seg.init()* - metoda inicjalizująca wyświetlacze siedmiosegmentowe. *seg* - obiekt typu **ISA7SegmentDisplay**. Metodę należy wywołać w funkcji *setup*();
- *seg.displayDigit(digit, dispID, dot = false)* - metoda pozwalająca wyświetlić cyfrę na wyświetlaczu. *seg* - obiekt typu **ISA7SegmentDisplay**, *digit* - cyfra, która ma zostać wyświetlona, musi być z zakresu <0, 9>, w innym przypadku funkcja nic nie zrobi, *dispID* - id wyświetlacza, na którym ma zostać wyświetlona cyfra, musi być wartością z zakresu <0, 4), w inny przypadku funkcja nic nie zrobi, *dot* - przekazuje informację czy kropka ma zostać zapalona czy nie, domyślnie wartość jest ustawiona na **false**.
- *seg.setLed(values, dispID)* - metoda do ustawiania stanu diod na zadanym wyświetlaczu. *seg* - obiekt typu **ISA7SegmentDisplay**, *values* - ośmiobitowa wartość, w któej każdy bit ustawiony na 1 włączy daną diodę, a ustawiony na 0 wyłącza, *dispID* - id wyświetlacza, na którym mają zostać włączone diody, musi być wartością z zakresu <0, 4), w inny przypadku funkcja nic nie zrobi.

Przykład - program wyświetlający cyfry na wyświetlaczu siedmiosegmentowym:

```
#include <ISADefinitions.h>
#include <ISA7SegmentDisplay.h>
ISA7SegmentDisplay sseg;
void setup(){
  sseg.init();
}
void loop(){
  for (int i = 0; i < 10; ++i) {
     sseg.displayDigit(i, 0);
     delay(250);
  }
}
```

Zadanie 6.

Napisz program, który wyświetli na wyświetlaczu siedmiosegmentowym czas, jaki minął od uruchomienia programu, w formacie mm:ss. Dwukropek wyświetl, wykorzystując kropkę (parametr *dot*).

Wyświetlacz matrycowy LED 8x8

Przydatne funkcje:

Biblioteka **ISALedControl** odpowiada za obsługę wyświetlacza matrycowego LED 8x8, dostępna jest po dołączeniu do programu pliku **ISALedControl.h**.

Metody dostępne w ramach klasy ISALedControl:

- *lc.init()* - metoda inicjalizująca wyświetlacz matrycowy LED 8x8. *lc* - obiekt typu **ISALedControl**. Metodę należy uruchomić w funkcji *setup*().
- *lc.clearDisplay()* - metoda wyłączająca wszystkie diody na wyświetlaczu. *lc* - obiekt typu **ISALedControl**.
- *lc.setRow(row, value)* - metoda do ustawiania stanu diod w danym wierszu. *lc* - obiekt typu **ISALedControl**, *row* - numer wiersza, *value* - wartość ośmiobitowa, w której każdy bit ustawiony na 1 włączy daną diodę, a ustawiony na 0 wyłączy.
- *lc.setColumn(col, value)* - metoda do ustawiania stanu diod w danej kolumnie. *lc* - obiekt typu **ISALedControl**, *col* - numer kolumny, *value* - wartość ośmiobitowa, w której każdy bit ustawiony na 1 włączy daną diodę, a ustawiony na 0 wyłączy.
- *lc.setLed(row, col, state)* - metoda włącza lub wyłącza diodę. *lc* - obiekt typu **LedControl**, *row* - numer wiersza (z zakresu <0; 7>), *col* - numer kolumny (z zakresu <0; 7>), *state* - stan diody (**true** - włączona, **false** - wyłączona).

Przykład - program zapalający diodę w lewym górnym rogu wyświetlacza matrycowego LED 8x8:

```
#include <ISALedControl.h>
ISALedControl lc;
void setup()
{
  lc.init();
  lc.setLed(0, 0, true);
}
void loop() {}
```

Zadanie 7.

Napisz program, który będzie po kolei (wężykiem) włączał każdą diodę na wyświetlaczu matrycowym LED 8x8.

Wyświetlacz OLED

Przydatne funkcje:

Biblioteka **ISAOLED** odpowiada za obsługę wyświetlacza OLED. Dostępna jest po dołączeniu do programu pliku **ISAOLED.h**. Obiekt klasy **ISAOLED** przechowuje ramkę obrazu (o wymiarach 128 kolumn i 64 wierszy) oraz pozwala wykonywać na niej proste operacje graficzne, np. rysowanie punktów. Po wykonaniu operacji graficznych należy uruchomić metodę *renderAll()* która spowoduje przesłanie ramki obrazu z obiektu (pamięci Arduino) do wyświetlacza. W efekcie utworzony wcześniej obraz zostanie wyświetlony.

Metody dostępne w ramach klasy ISAOLED:

- *oled.begin()* - metoda inicjalizująca wyświetlacz matrycowy OLED o rozdzielczości 128x64. *oled* - obiekt typu **ISAOLED**. Metodę należy uruchomić w funkcji *setup*().
- *lc.clear(bool render = true)* - metoda wyłączająca wszystkie punkty na wyświetlaczu. *oled* - obiekt typu **ISAOLED**, *render* - wartość logiczna powoduje natychmiastowe usunięcie obrazu z wyświetlacza.
- *oled.write(data)* - metoda znak na podstawie kodu ASCII. *oled* - obiekt typu **ISAOLED**, *data* - kod ASCII znaku, który ma zostać wyświetlony, znaki nie będące kodami ASCII nie są wyświetlane.
- *oled.*gotoXY*(cx, cy)* - metoda ustawia kursor do wyświetlania tekstu na pozycji *cx*, *cy*. *oled* - obiekt typu **ISAOLED**, *cx* - współrzędna na osi X, na której ma zostać ustawiony kursor (przyjmuje wartość z zakresu <0; 127>), *cy* - współrzędna na osi Y, na której ma zostać ustawiony kursor (przyjmuje ona wartości z zakresu <0; 7>).
- *oled.setPixel(x, y, v)* - metoda włącza lub wyłącza dany punkt. *oled* - obiekt typu **ISAOLED**, *x, y* - współrzędne punktu, *v* - stan punktu (**true** - włączony, **false** - wyłączony).
- *oled.writeLine(x1, y1, x2, y2)* - metoda rysuje wyłącznie linię pionową lub poziomą. *oled* - obiekt typu **ISAOLED**, *x1, y1* - współrzędne początku odcinka, *x2, y2* - współrzędne końca odcinka.
- *oled.writeRect(x, y, w, h, fill)* - metoda rysuje prostokąt. *oled* - obiekt typu **ISAOLED**, *x, y* - współrzędne lewego górnego rogu prostokąta, *w* - szerokość prostokąta, *h* - wysokość prostokąta, *fill* - wypełnienie prostokąta (**true** - wypełniony, **false** - tylko krawędzie).
- *oled.print(text)* - metoda wyświetla tekst na ekranie. *oled* - obiekt typu **ISAOLED**, *text* - tekst, który ma zostać wyświetlony (może to być również liczba).
- *oled.renderAll()* - przesyła zbudowaną ramkę obrazu do wyświetlacza i wyświetla ją natychmiast. *oled* - obiekt typu **ISAOLED**.

Przykład - program wyświetlający napis “Hello world” na wyświetlaczu OLED:

```
#include <ISAOLED.h>
ISAOLED oled;
void setup() {
  oled.begin();
```

```
  oled.gotoXY(20, 2);
  oled.print("Hello world");
  oled.renderAll();
  delay(1000);
}
void loop() {}
```

Zadanie 8.

Napisz program, który będzie rysował na ekranie OLED szachownicę o liczbie pól 8x8.

Dziękujemy za wzięcie udziału w zajęciach pilotażowych:)