Programowanie mikrokontrolerów w języku wysokiego poziomu 1

(TS1F1008)

Politechnika Białostocka - Wydział Elektryczny
Elektronika i telekomunikacja, sem. I, studia stacjonarne I stopnia
Rok akademicki 2024/2025

Wykład nr 4 (05.12.2024)

dr inż. Jarosław Forenc

Plan wykładu nr 4

- Arduino
  - podstawowe informacje
  - historia
  - moduły Arduino, Arduino Uno R3, ATmega328P
  - moduł wykorzystywany na zajęciach
  - środowisko Arduino IDE
  - środowisko Visual Studio Code, rozszerzenie PlatformIO IDE
  - struktura programu, typy danych
  - sterowanie diodą LED
  - sterowanie przyciskiem

Arduino - podstawowe informacje

- Platforma programistyczno-sprzętowa dla systemów wbudowanych
- Strona internetowa: www.arduino.cc
- Projekt typu open-source (Open Hardware)
- Pojedynczy moduł: mikrokontroler montowany w pojedynczym obwodzie drukowanym, z wbudowaną obsługą układów wejścia-wyjścia i interfejsem mikrokontroler-komputer PC (USB)
- Standaryzowany język programowania: bazuje na Wiring i jest podobny do C/C++
- Środowisko programistyczne: Arduino IDE

Arduino - historia (1/2)

- Interactive Design Institute Ivrea (IDII), Ivrea, Włochy
- 2003 - student Hernando Barragán, w ramach realizacji pracy magisterskiej, tworzy platformę programistyczną Wiring (opiekunowie pracy: Massimo Banzi, Casey Reas)
- Cel projektu: stworzenie prostych, tanich narzędzi do tworzenia projektów cyfrowych przez osoby nie będące inżynierami (artystów, projektantów)
- Platforma: płytka drukowana z mikrokontrolerem ATmega128 oraz środowisko programistyczne (IDE) umożliwiające łatwe programowanie mikrokontrolera
- 2005 - Massimo Banzi, David Mellis (student) i David Cuartielles rozszerzają platformę Wiring poprzez dodanie obsługi tańszego mikrokontrolera ATmega8 - projekt ten zostaje nazwany Arduino

Arduino - historia (2/2)

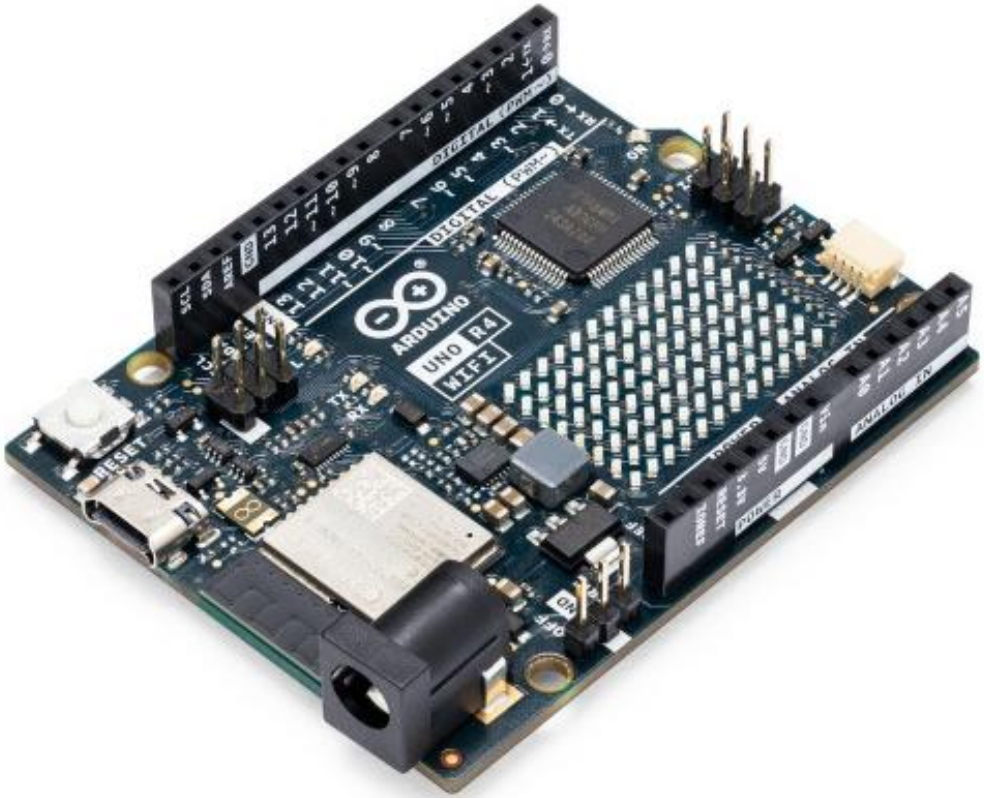
- 2008 - Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino i David Mellis tworzą firmę Arduino LLC, która miała być właścicielem znaku towarowego Arduino
- Produkcją i sprzedażą płytek miały zajmować się zewnętrzne firmy płacące tantiemy Arduino LCC
- „The Untold History of Arduino” by Hernando Barragán:
  - https://arduinohistory.github.io/

Arduino - informacje w Internecie

- Strona internetowa: www.arduino.cc
- Arduino open source community: github.com/arduino
- Kursy internetowe:
  - Forbot (Kurs Arduino, poziom I): https://forbot.pl/blog/kurs-arduino-podstawy-programowania-spis-tresci-kursu-id5290
  - Forbot (Kurs Arduino, poziom II): https://forbot.pl/blog/kurs-arduino-ii-wstep-spis-tresci-id15494
  - Kurs Arduino (Youtube, kanał KoValsky majstruje): https://www.youtube.com/watch?v=TzTmWqoN9i8
  - Kurs Arduino (Youtube, kanał Forbidden Bit): https://www.youtube.com/playlist?list=PLgjf5CWt5eA9mCBgZWinYSTVw960RYAqg

Arduino - moduły

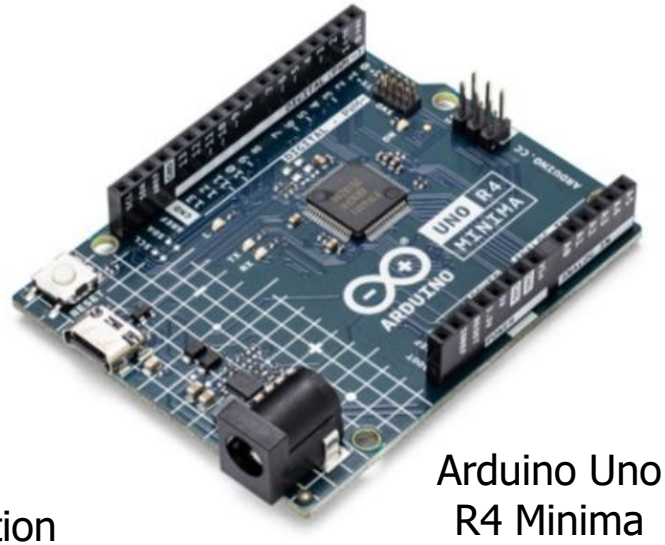
Arduino Uno Rev3



Arduino Uno R4 WiFi

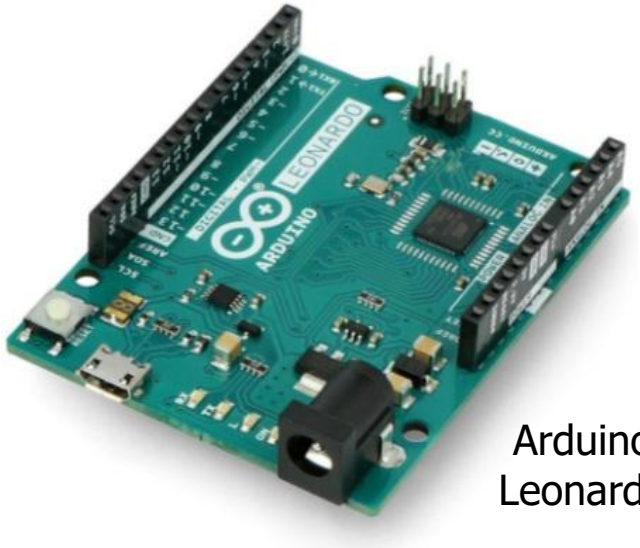
Arduino - moduły

Arduino Uno
Mini Limited Edition

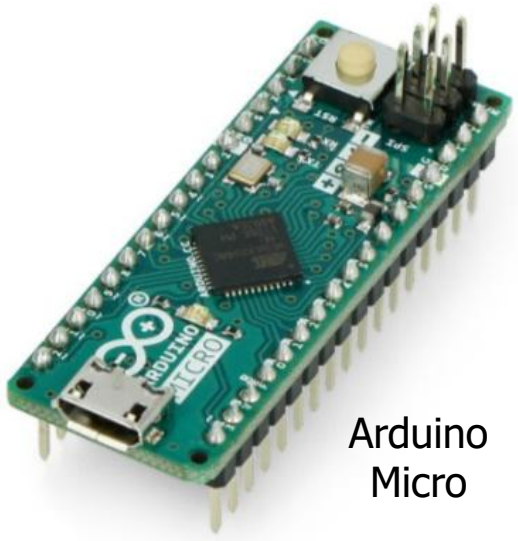


Arduino Uno
R4 Minima

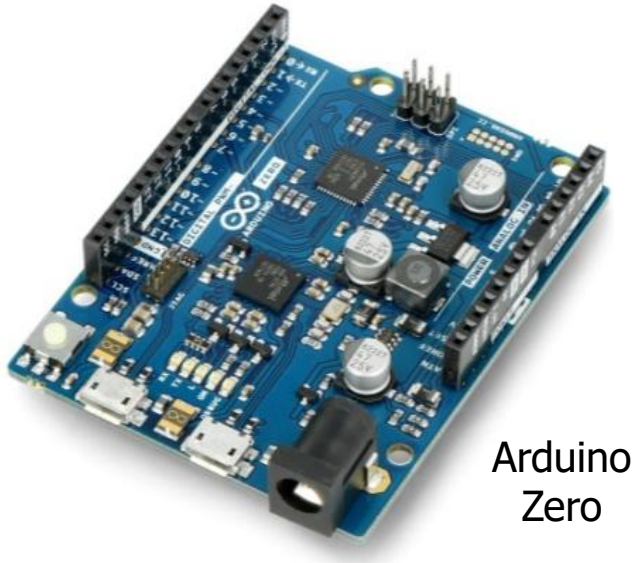
Arduino Uno
R4 WiFi



Arduino
Leonardo



Arduino
Micro



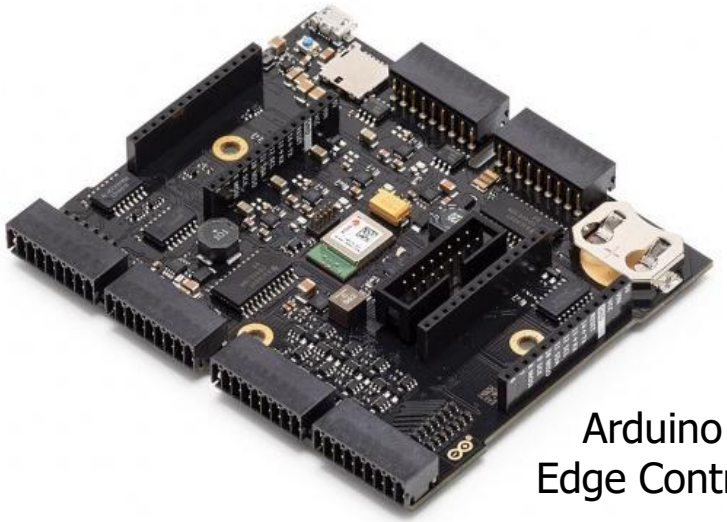
Arduino
Zero

Arduino - moduły

Arduino
MKR Zero

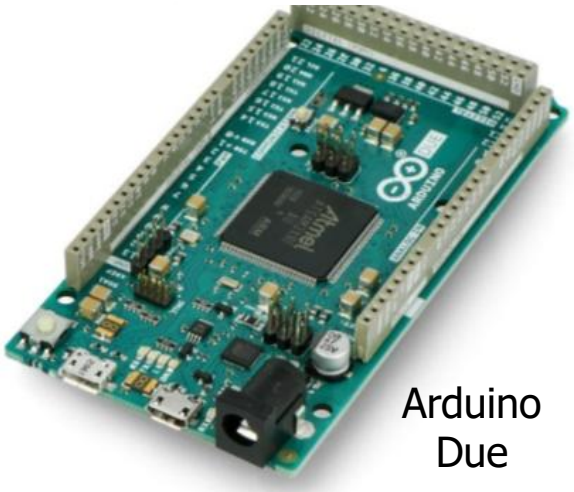
Arduino
Nano

Arduino
Portenta C33



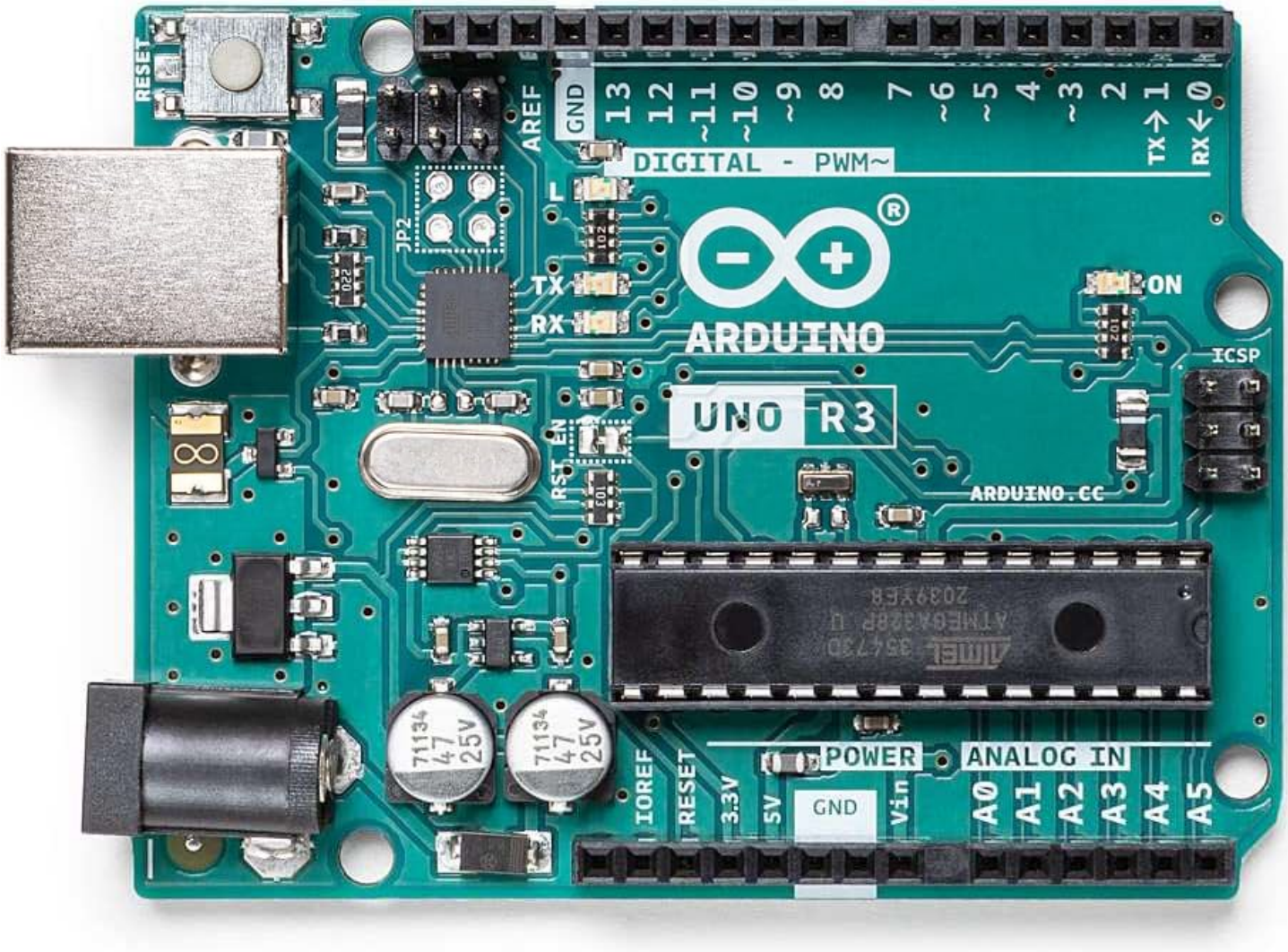
Arduino
Edge Control

Arduino
Nicla Vision



Arduino
Due

Arduino Uno R3



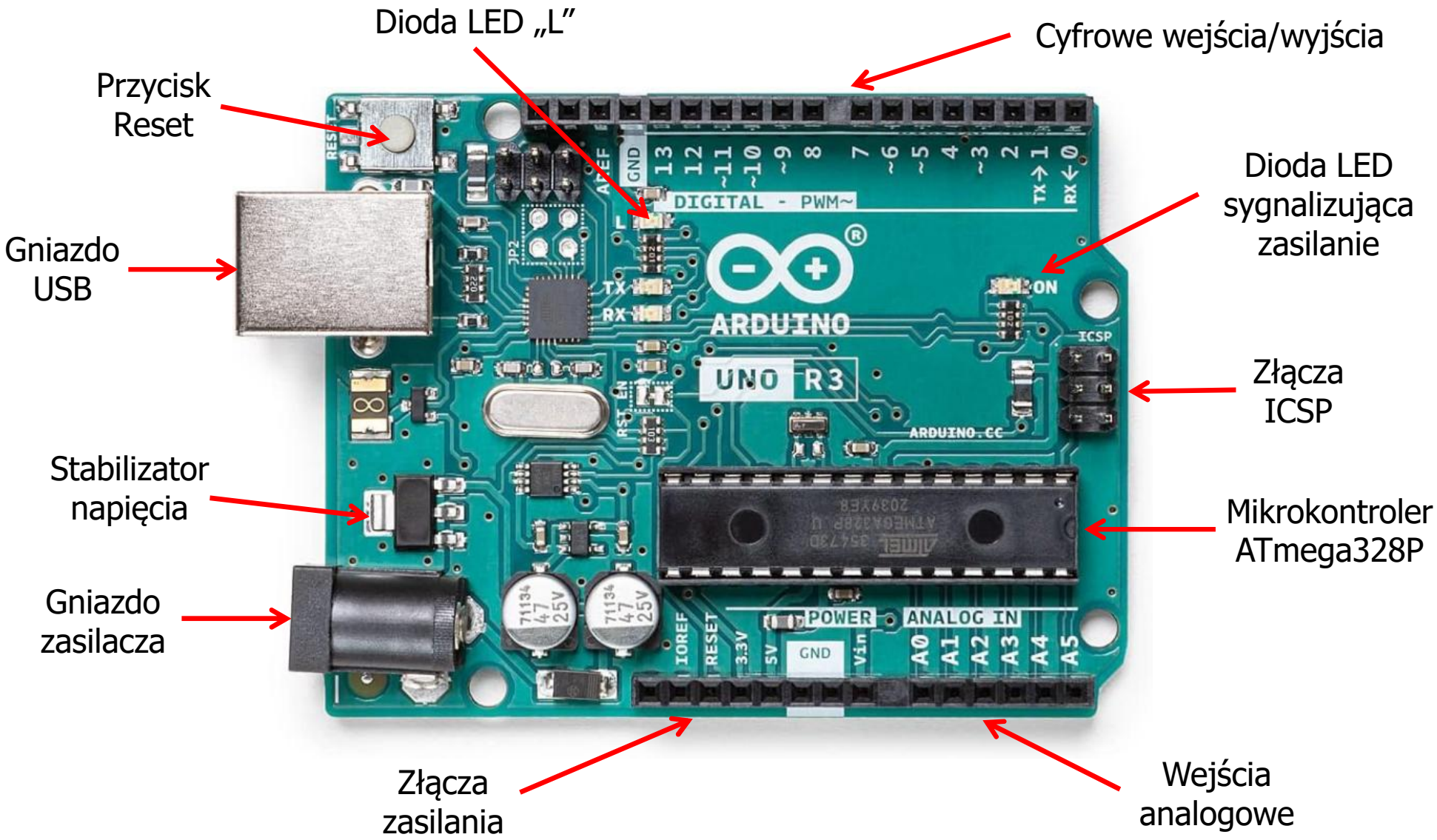
Arduino Uno R3 - specyfikacja

- Zasilanie:
  - złącze koncentryczne 5,5/2,1 mm (napięcie od 7 do 12 V)
  - złącze USB typu B - napięcie 5 V
- Mikrokontroler: ATmega328P w obudowie DIP-28:
  - pamięć SRAM: 2 kB
  - pamięć Flash: 32 kB (0,5 kB zarezerwowane dla bootloadera)
  - pamięć EEPROM: 1 kB
  - częstotliwość zegara: 16 MHz
- 20 uniwersalnych wyprowadzeń wejść/wyjść:
  - 14 cyfrowych wejść/wyjść (w tym 6 z możliwością generowania 8-bitowego sygnału PWM)
  - 6 wejść analogowych o rozdzielczości 10 bitów (mogących pełnić funkcję cyfrowych wejść/wyjść)

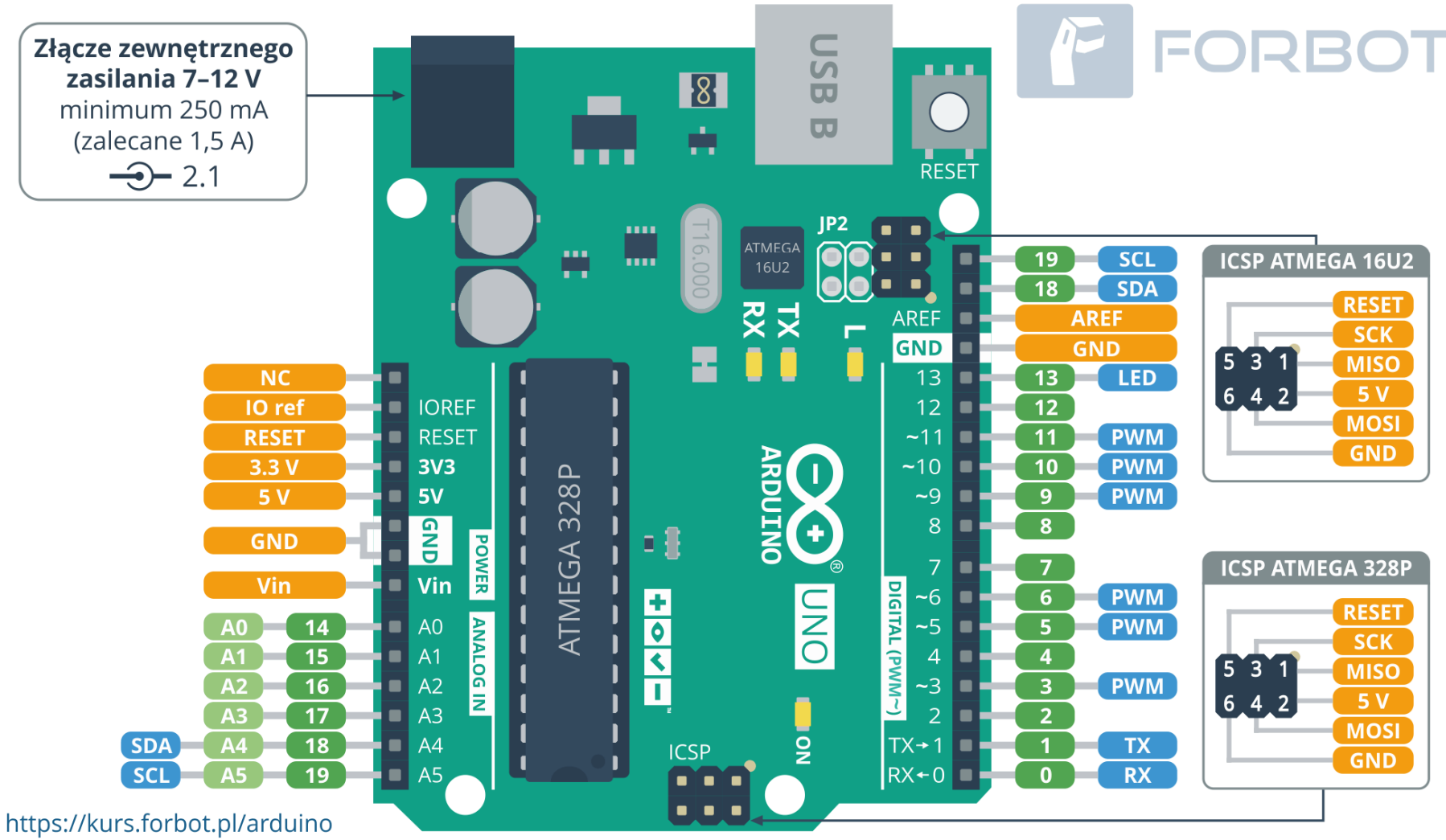
Arduino Uno R3 - specyfikacja

- Interfejsy komunikacyjne: UART, I2C, SPI
- Zewnętrzne przerwania: 2
- Maksymalny prąd:
  - dla wyjścia 5 V: 500 mA
  - dla wyjścia 3,3 V: 50 mA
  - dla poszczególnych GPIO: 20 mA
- Rozmiary płytki: 68,6 × 53,4 mm

Arduino Uno R3 - elementy na płytce



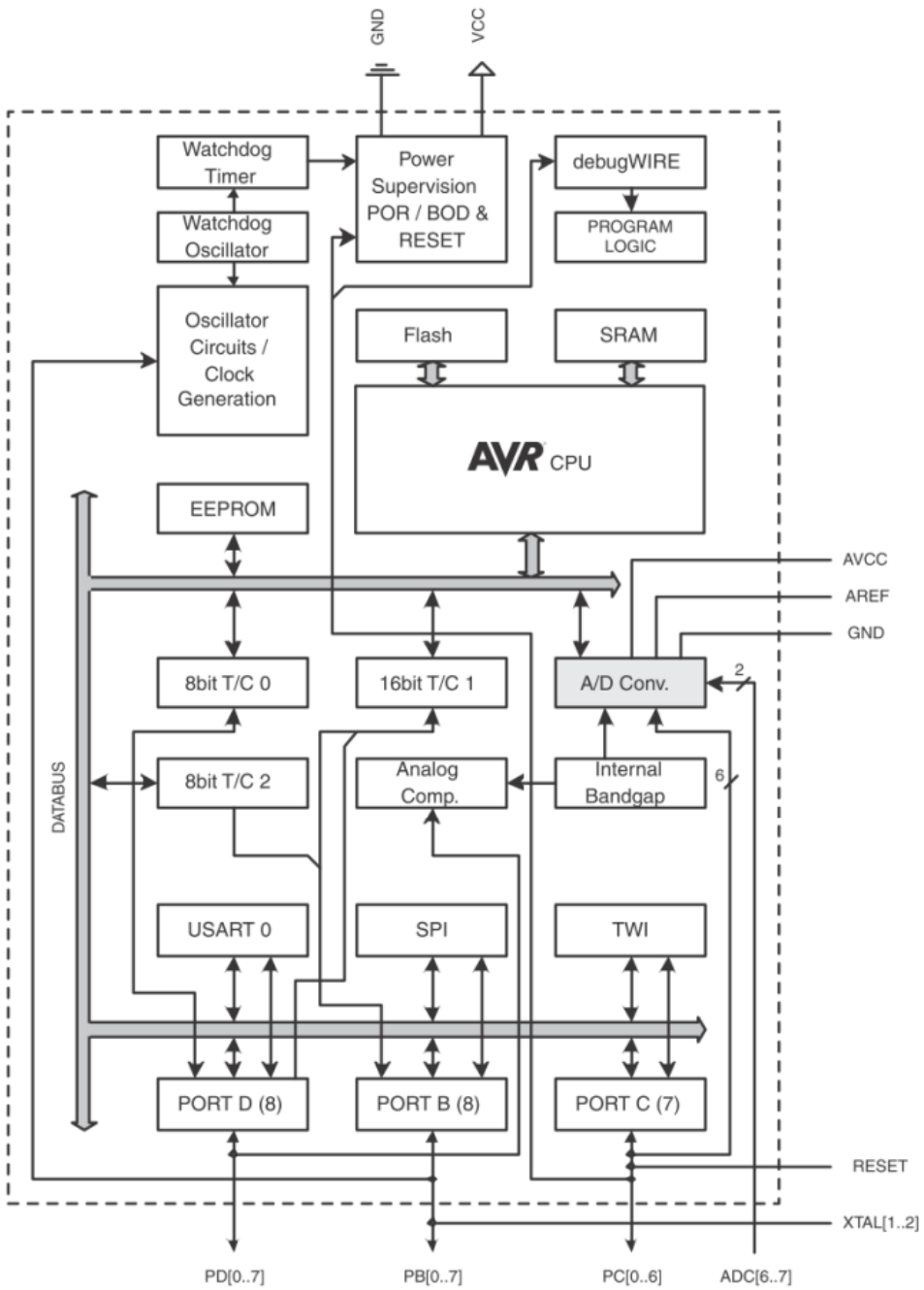
Arduino Uno R3 - opis wyprowadzeń



https://kurs.forbot.pl/arduino

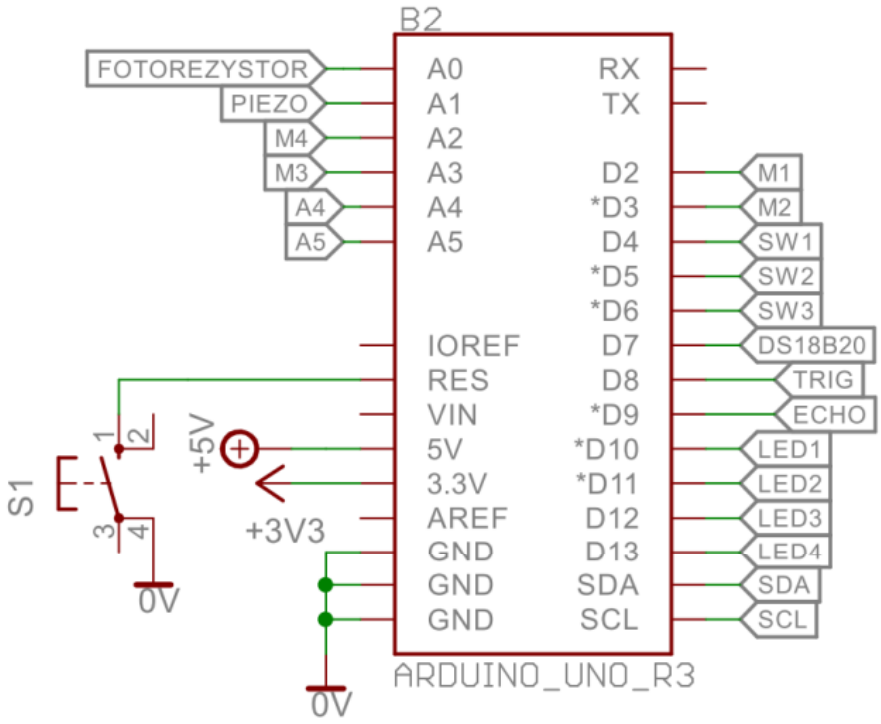
ATmega328P

- Specyfikacja:
  - zasilanie: 1,8 V - 5,5 V
  - taktowanie: do 20 MHz
  - pamięć Flash: 32 KB
  - 23 linie wyjścia/wejścia
  - dwa 8-bitowe liczniki
  - jeden 16-bitowy licznik
  - 6 kanałów PWM
  - 6 kanałów 10-bitowego przetwornika A/D
  - sprzętowe interfejsy komunikacyjne: USART, SPI, TWI (I2C)
  - obudowa DIP

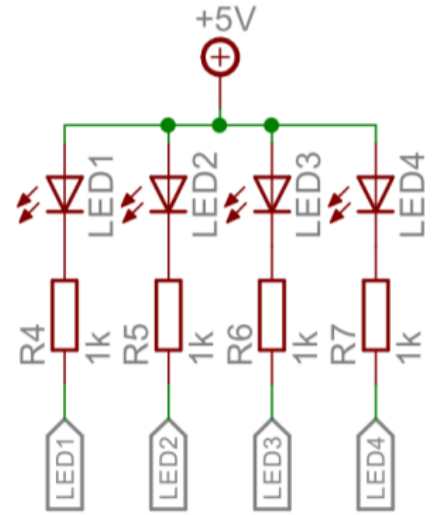


Arduino - moduł stosowany na zajęciach

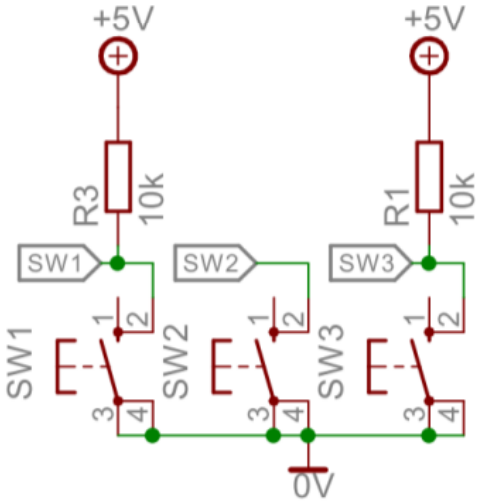
Moduł stosowany na zajęciach



Schemat podłączenia wyprowadzeń modułu Arduino

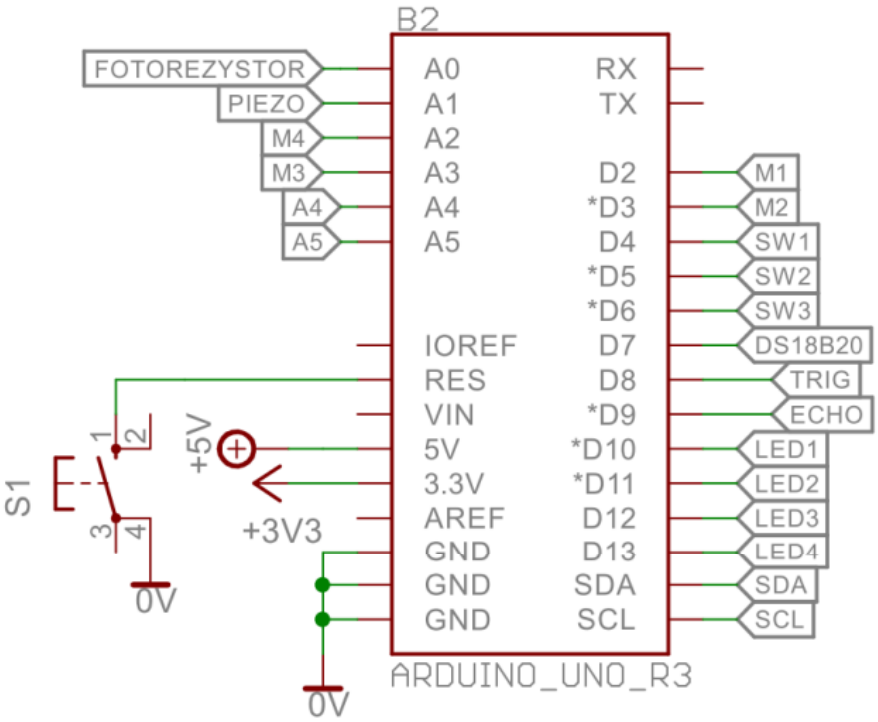


Schemat podłączenia diod LED

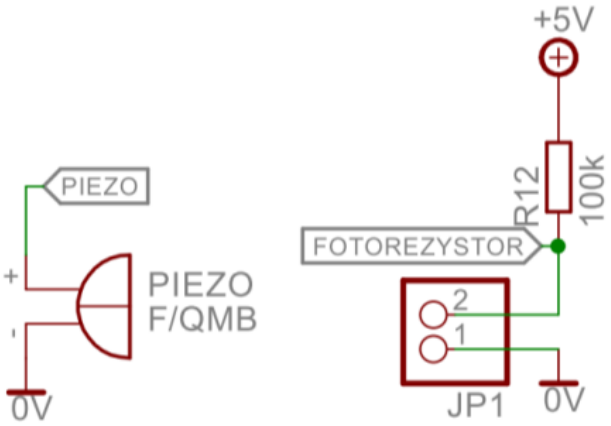


Schemat podłączenia przycisków

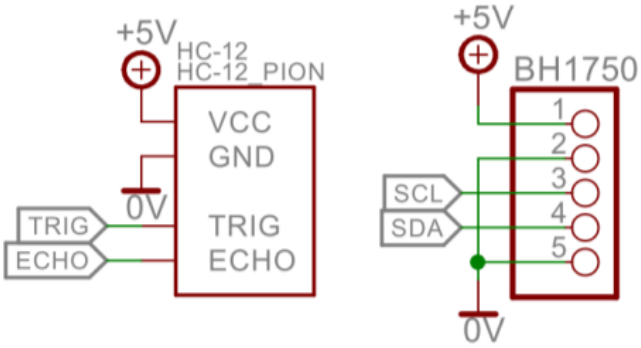
Moduł stosowany na zajęciach



Schemat podłączenia
wyprowadzeń modułu Arduino

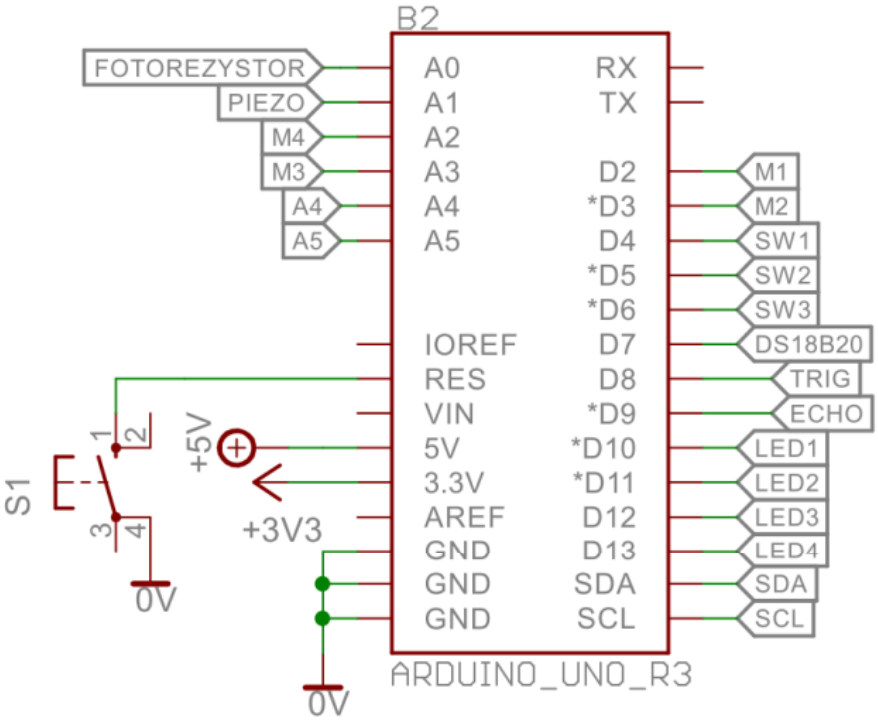


Schemat podłączenia buzzera i fotorezystora

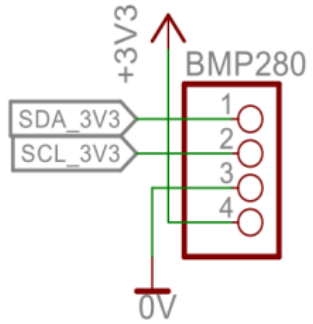


Schemat podłączenia czujnika odległości
i czujnika natężenia światła

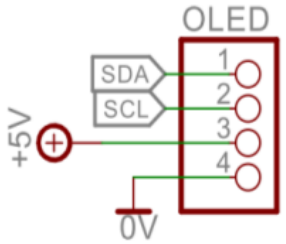
Moduł stosowany na zajęciach



Schemat podłączenia
wyprowadzeń modułu Arduino



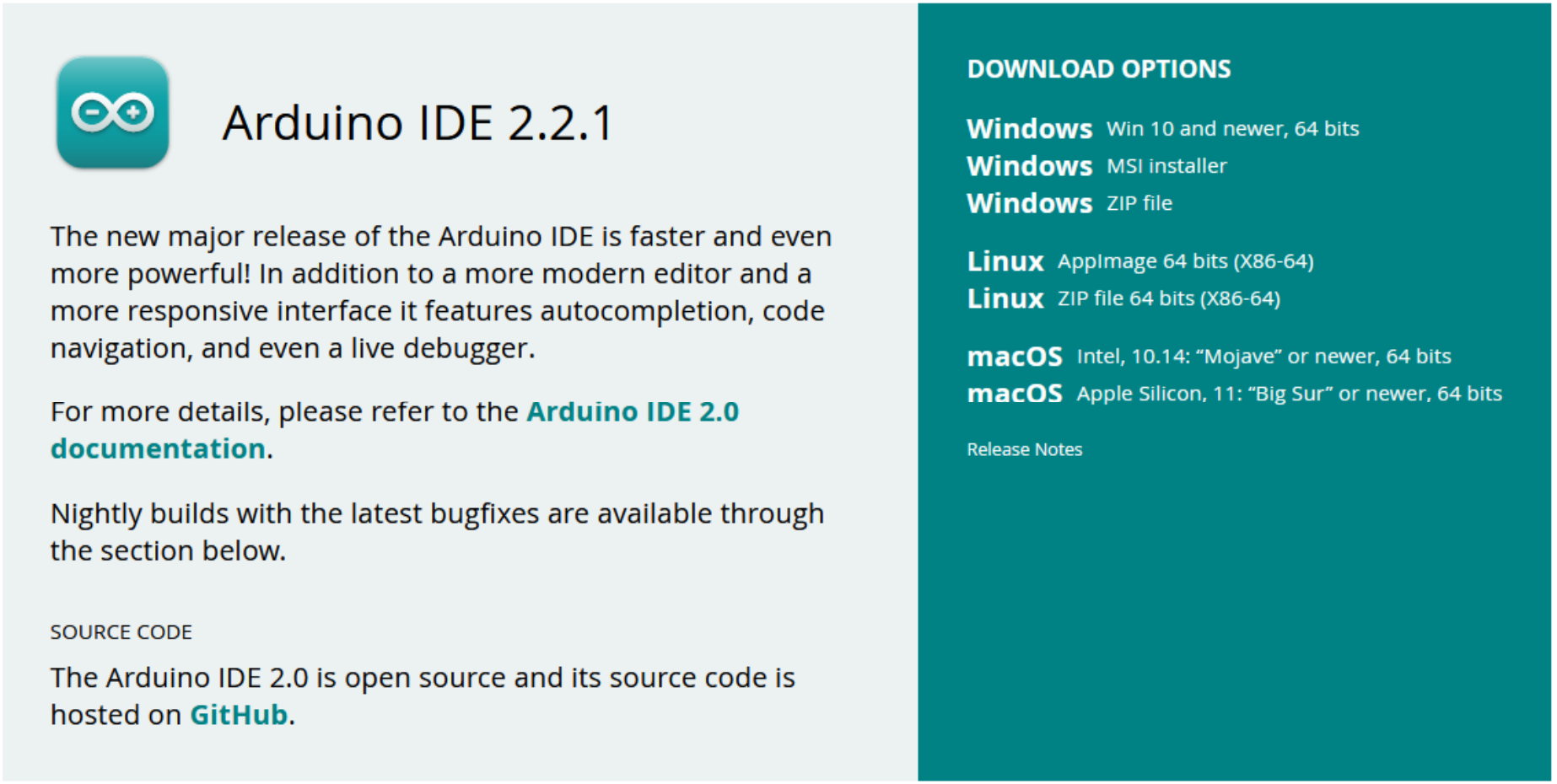
Schemat podłączenia
czujnika ciśnienia i temperatury



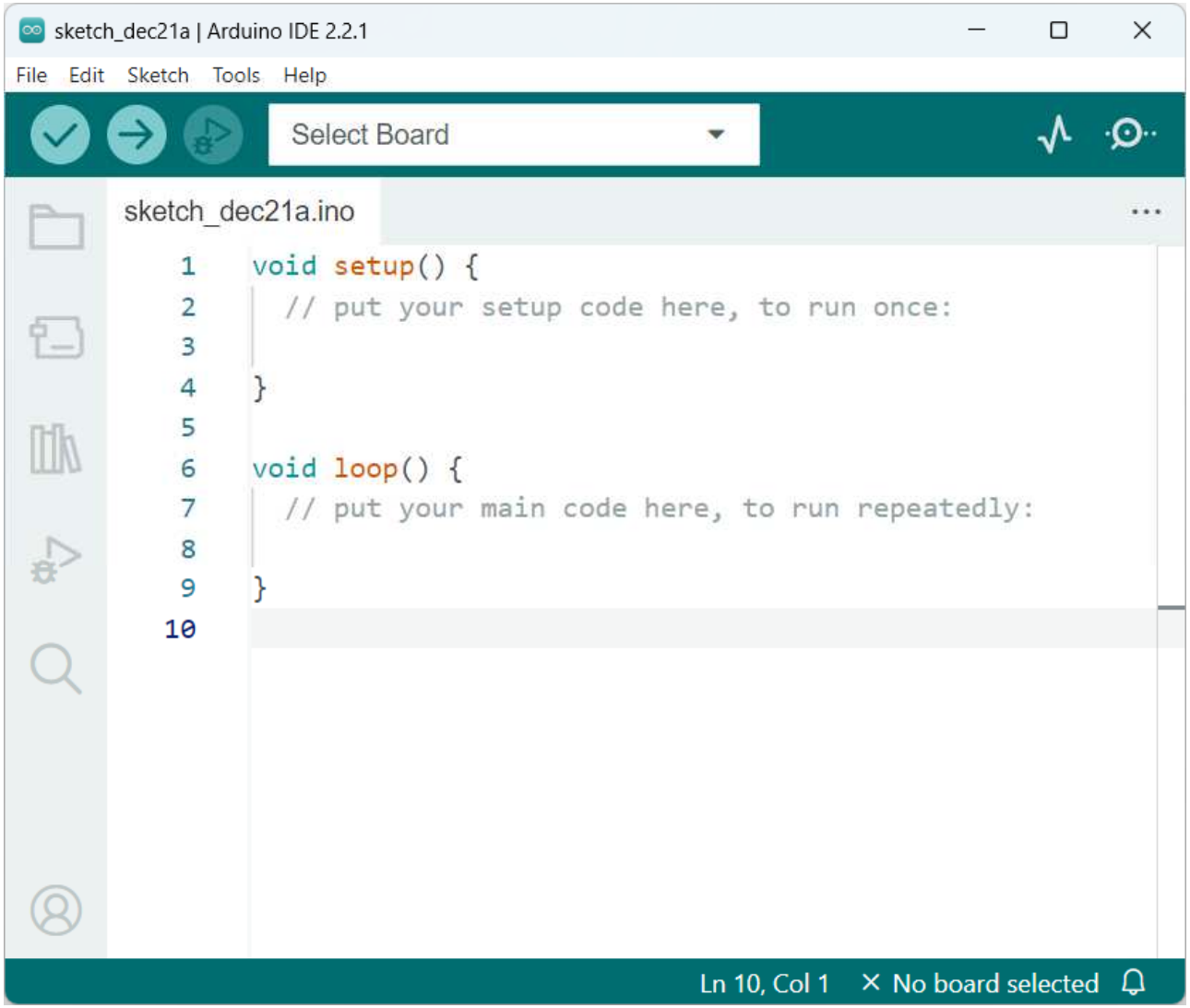
Schemat podłączenia
wyświetlacza OLED

Arduino IDE

- https://www.arduino.cc/en/software (najnowsza wersja: 2.3.3)



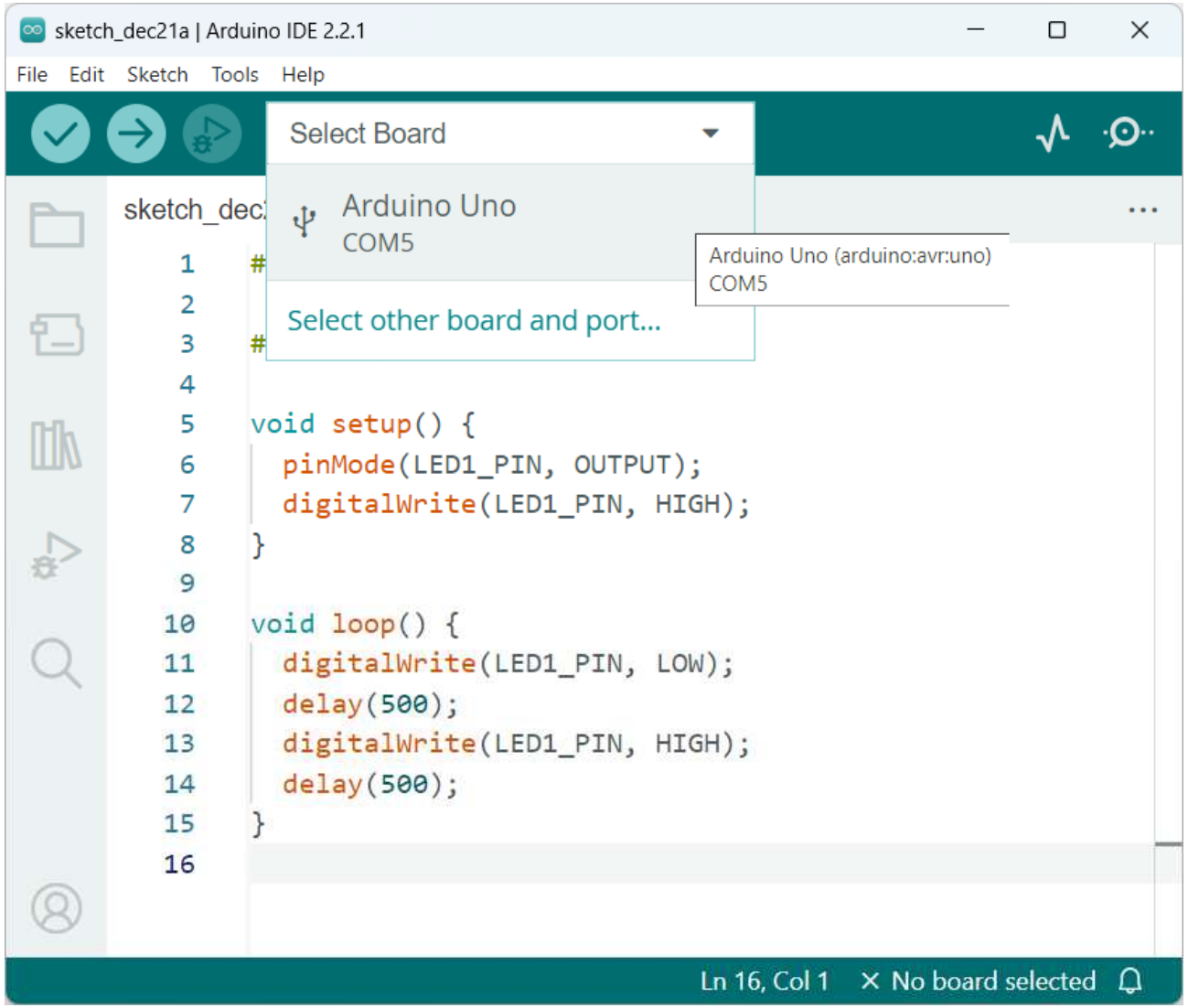
Arduino IDE 2.2.1



Środowisko Arduino IDE z przykładowym kodem programu

Arduino IDE 2.2.1

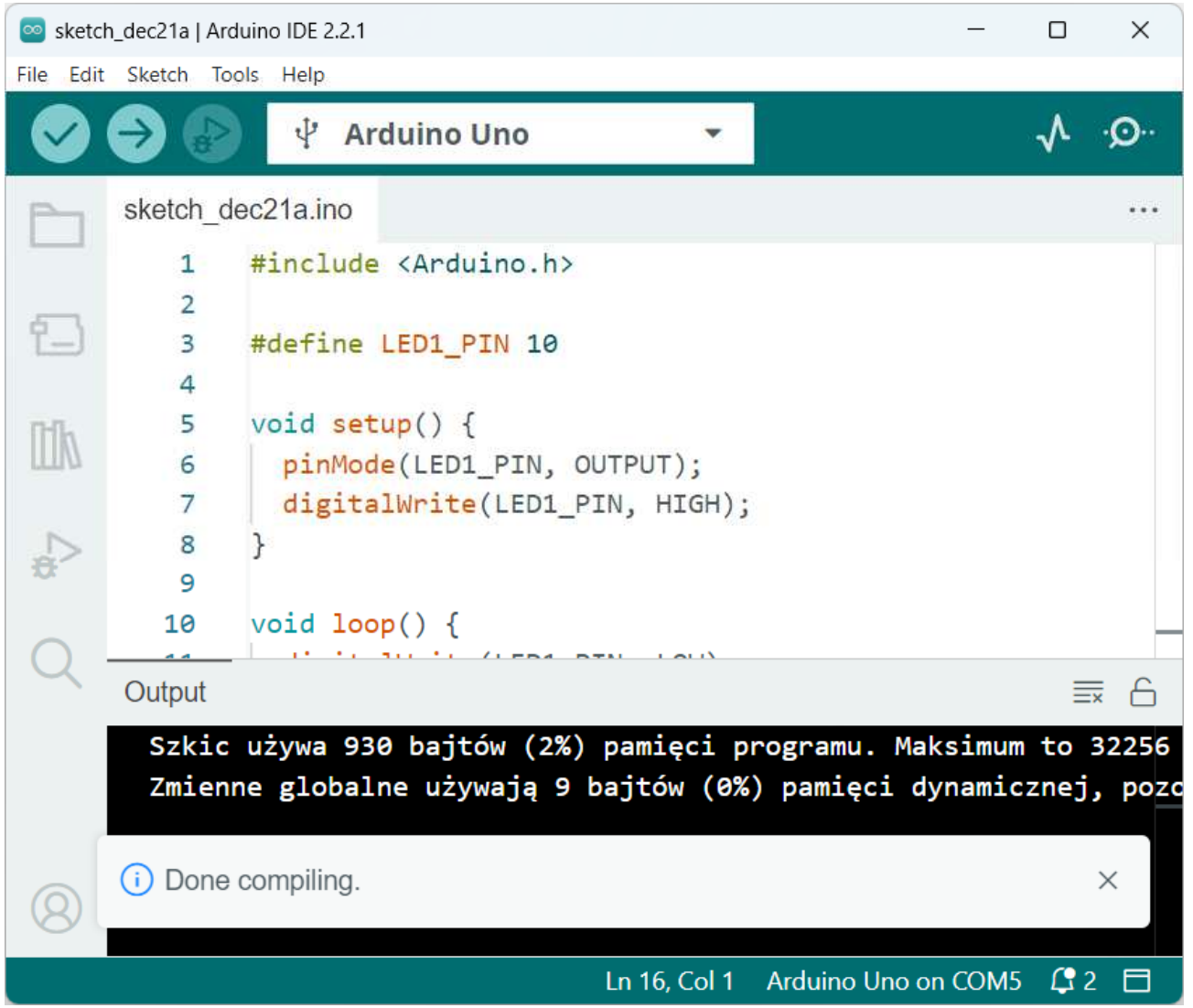
- wprowadzamy kod programu
- wybieramy typ płytki



Arduino IDE 2.2.1

- wprowadzamy kod programu
- wybieramy typ płytki

- Verify - sprawdzenie i kompilacja programu

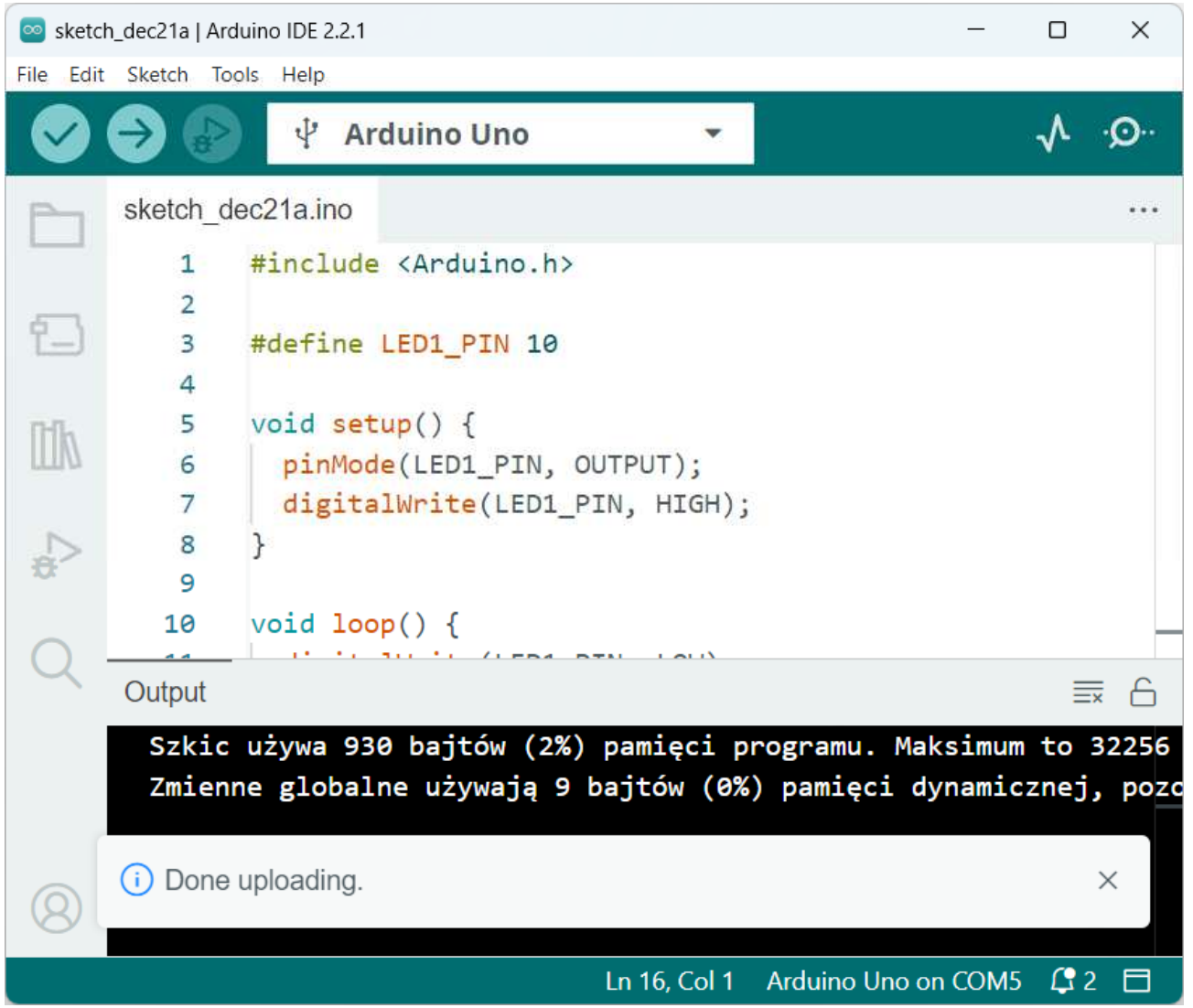


Arduino IDE 2.2.1

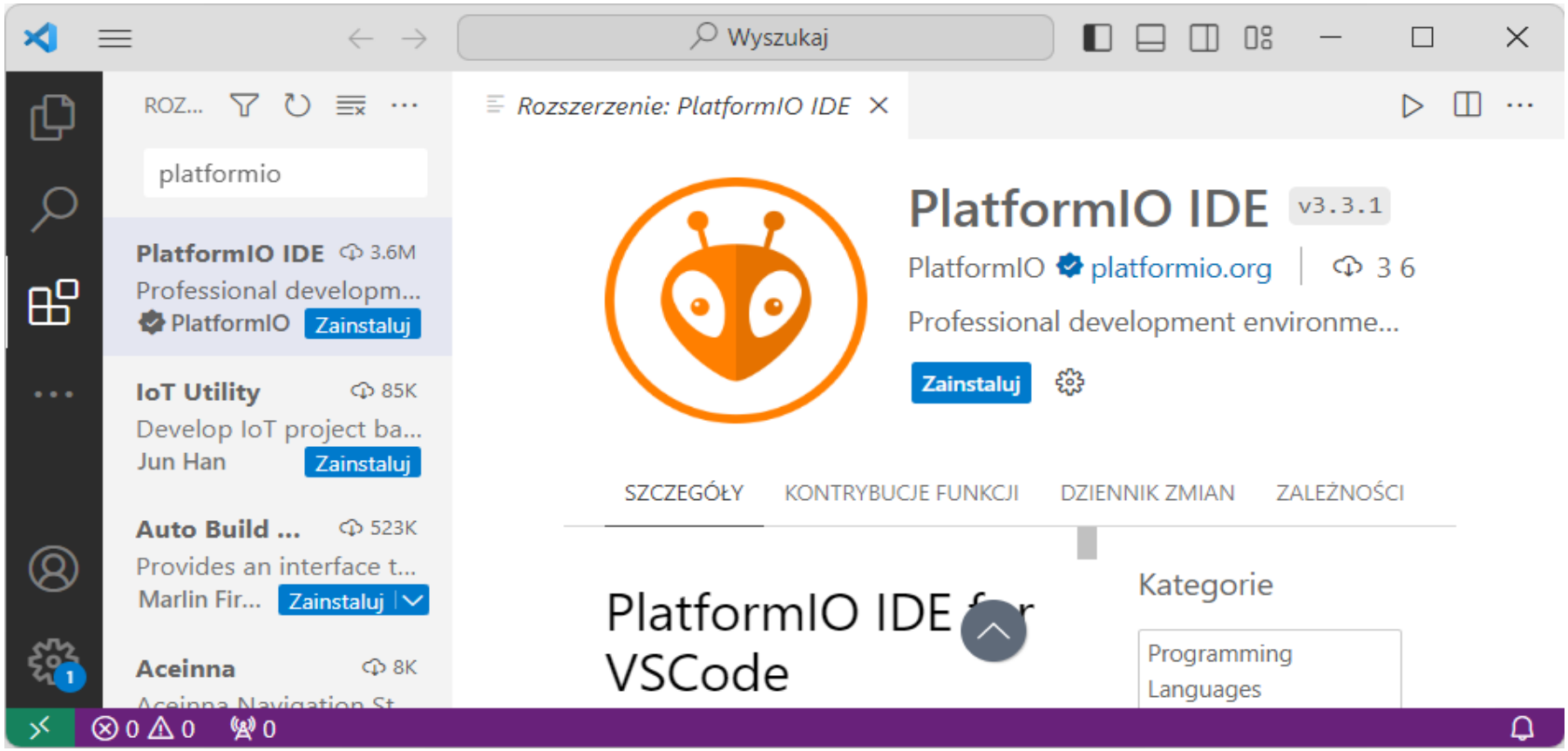
- wprowadzamy kod programu
- wybieramy typ płytki

- Verify - sprawdzenie i kompilacja programu

- Upload - sprawdzenie, kompilacja i przesłanie programu do Arduino

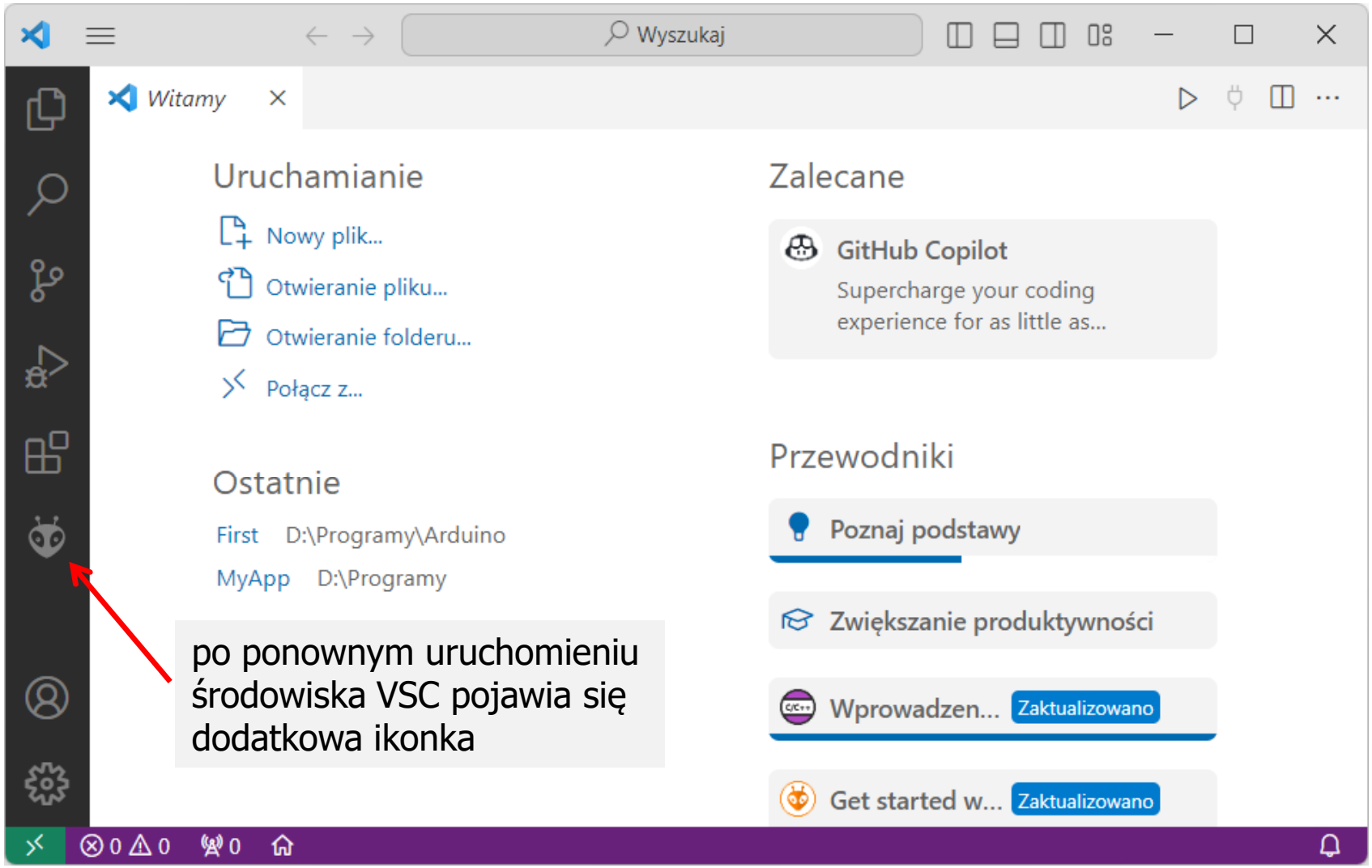


Visual Studio Code + PlatformIO IDE 3.3.1

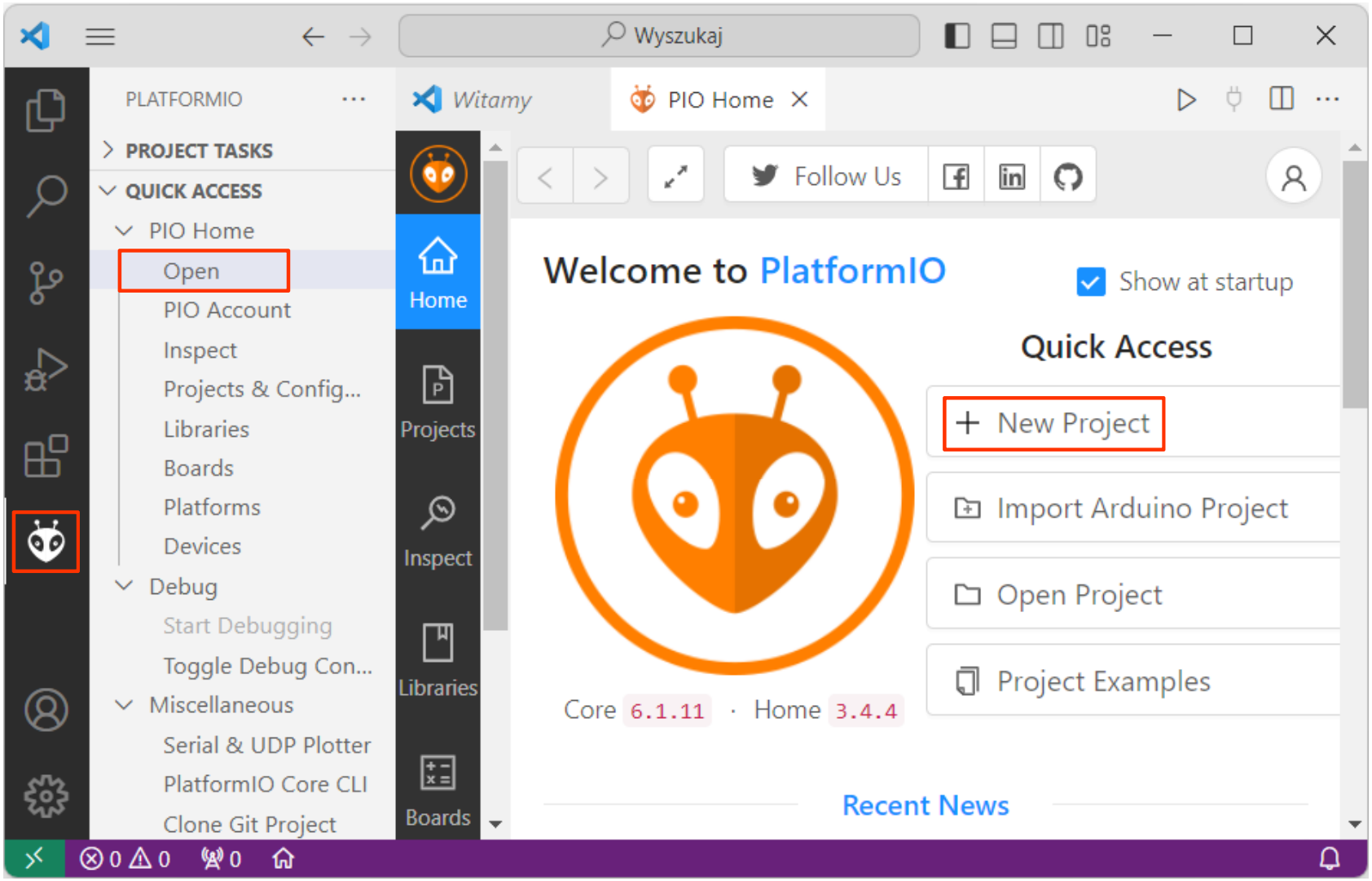


- rozszerzenia: Ctr + Shift + X
- wyszukanie i instalacja rozszerzenia PlatformIO IDE

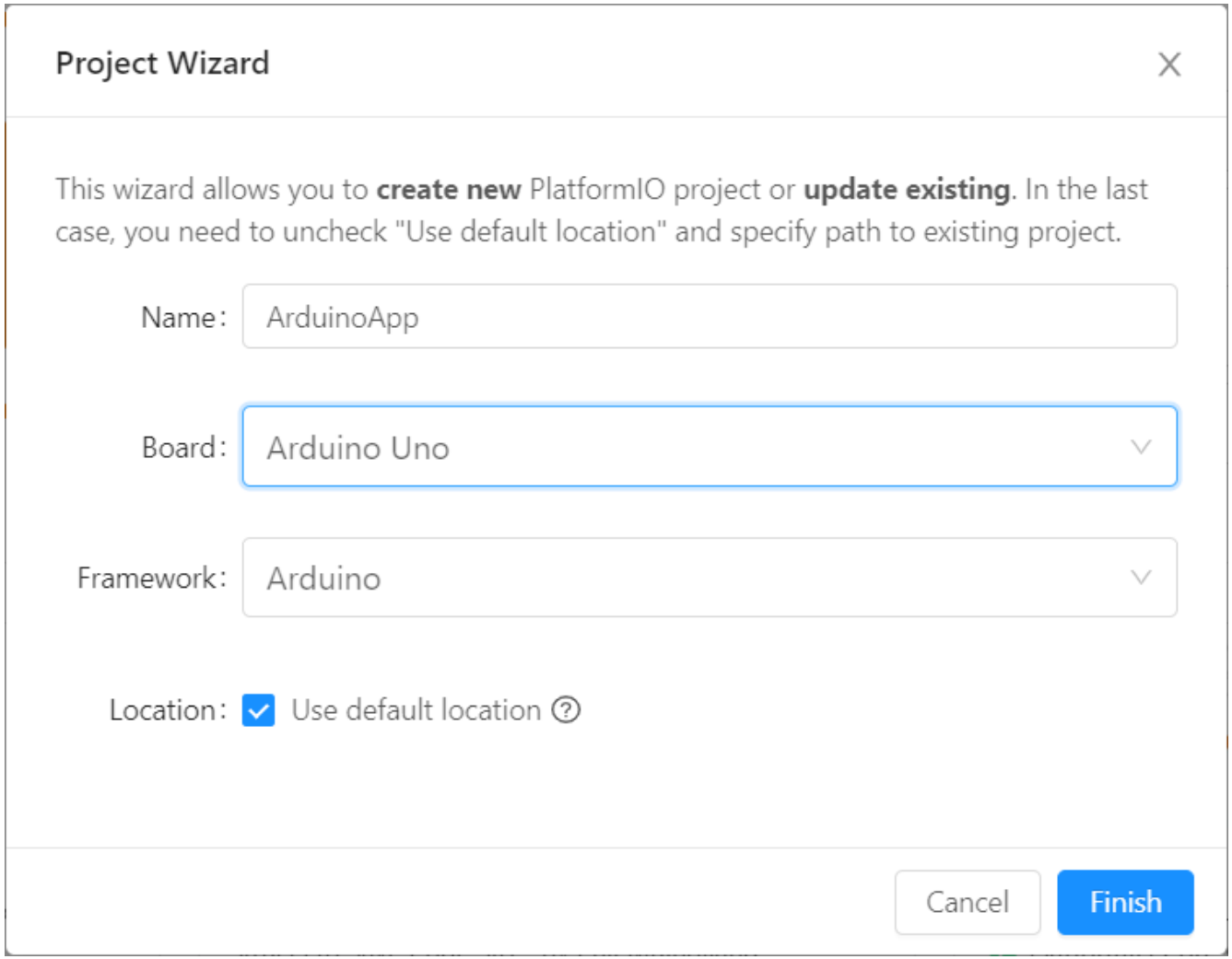
Visual Studio Code + PlatformIO IDE 3.3.1



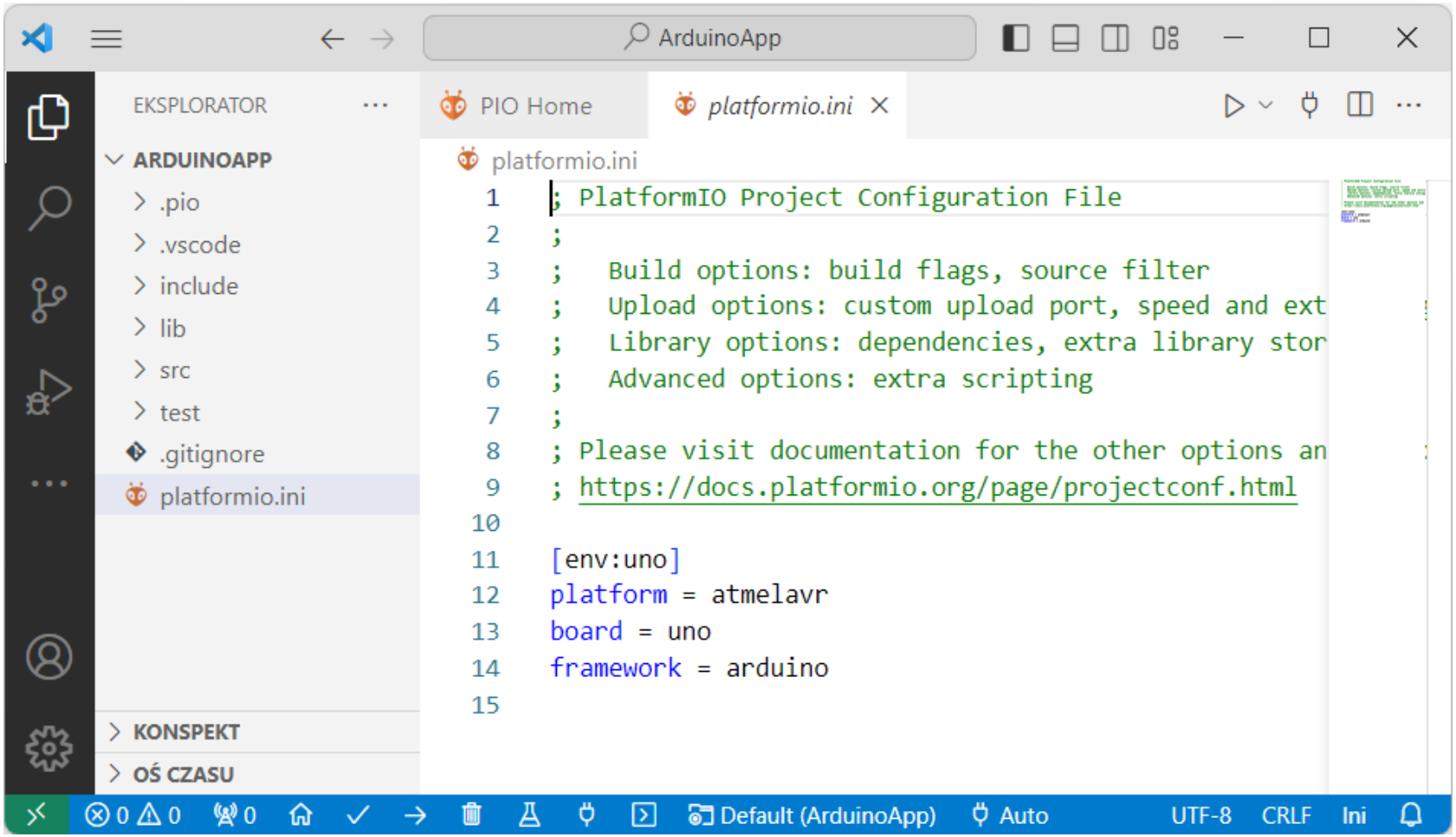
VS Code: stworzenie projektu



VS Code: stworzenie projektu

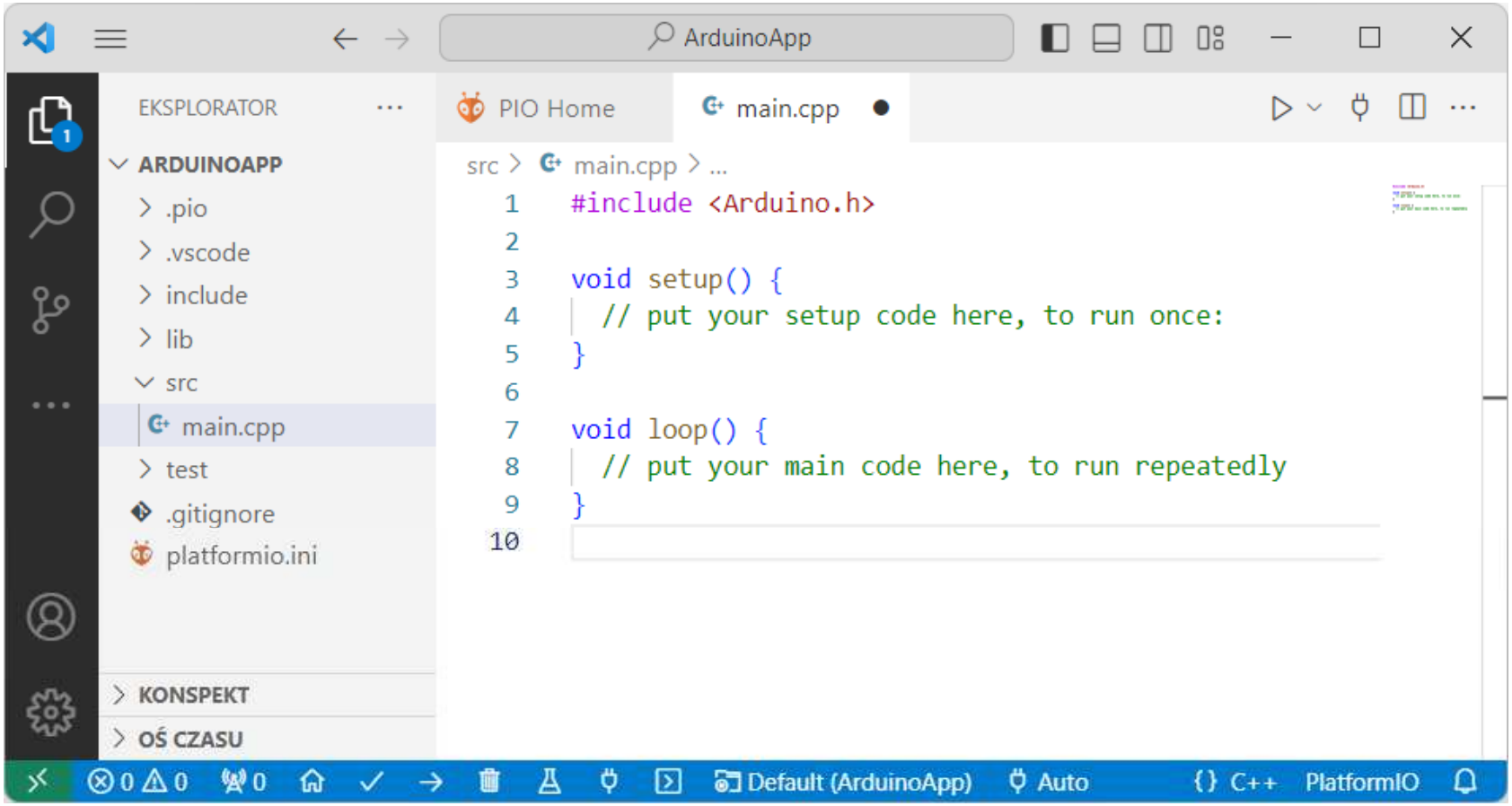


VS Code: stworzenie projektu



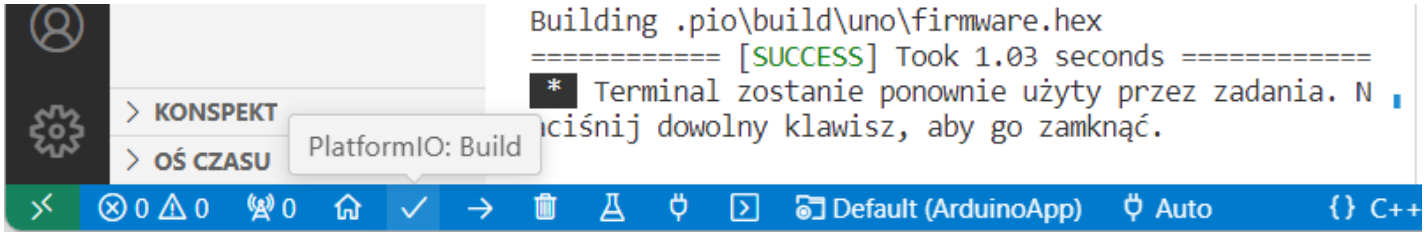
- platformio.ini - główny plik konfiguracyjny projektu

VS Code: stworzenie projektu

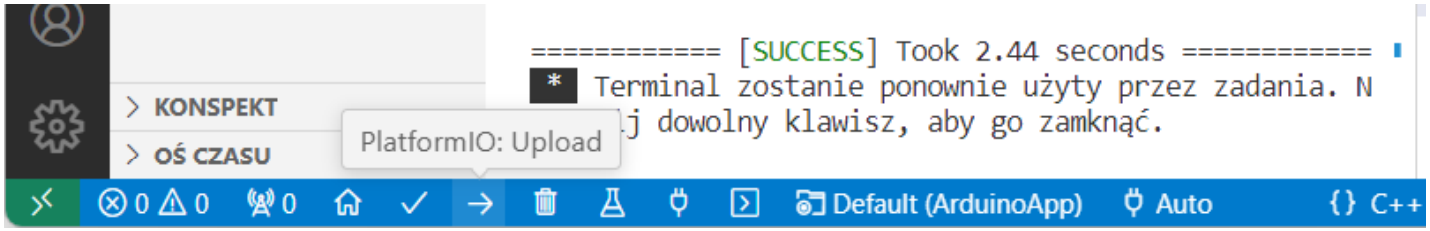


- src → main.cpp - plik z kodem źródłowym programu

VS Code: kompilacja i przesłanie programu



- Build - sprawdzenie i kompilacja programu



- Upload - sprawdzenie, kompilacja i przesłanie programu do Arduino

Arduino - ogólna struktura programu

- Ogólna struktura programu

```
#include <Arduino.h>

void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

  - setup() - funkcja wykonywana tylko jeden raz po uruchomieniu Arduino
  - loop() - funkcja uruchamiana w pętli (w kółko)

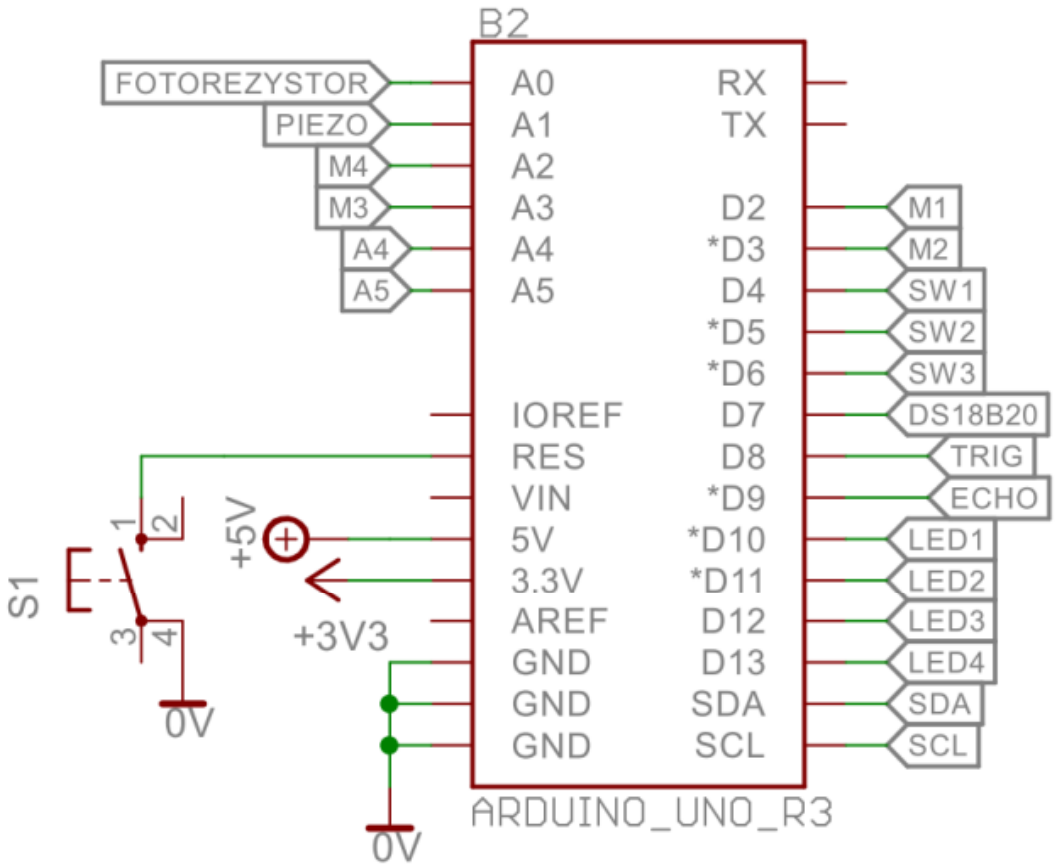
Arduino Uno - typy danych

Nazwa	Rozmiar (bajty)	Uwagi
`bool`	1	przechowuje jedną z dwóch wartości: true lub false
`boolean`	1	oznacza to samo co bool
`byte`	1	8-bitowa liczba całkowita bez znaku, zakres: 0 .. 255
`char`	1	jeden znak w postaci kodu ASCII, np. 'A' → 65
`double`	4	liczba zmiennoprzecinkowa, oznacza to samo co float
`float`	4	liczba zmiennoprzecinkowa, zakres: -3,4E+38 … 3,4E+38
`int`	2	liczba całkowita ze znakiem, zakres: -32.768 … 32.767
`long`	4	liczba całkowita ze znakiem, zakres: -2.147.483.648 … 2.147.483.647

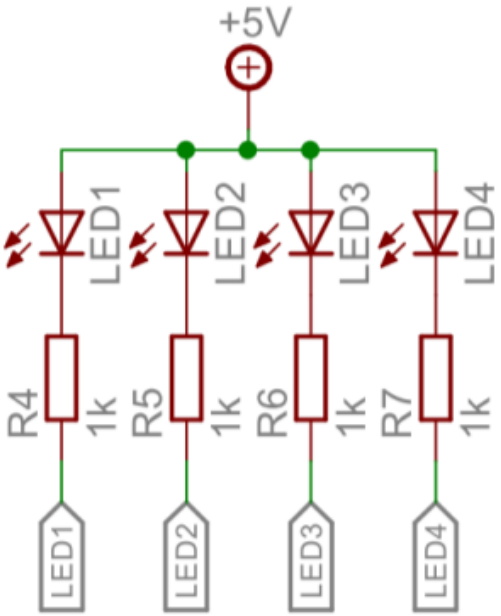
Arduino Uno - typy danych

Nazwa	Rozmiar (bajty)	Uwagi
`short`	2	liczba całkowita ze znakiem, zakres: -32.768 … 32.767
`size_t`	–	typ reprezentujący rozmiar obiektu w bajtach
`string`	–	łańcuch znaków jako tablica znaków, np. char Str[10];
`String()`	–	klasa reprezentująca łańcuchy znaków
`unsigned char`	1	8-bitowa liczba całkowita bez znaku, zakres: 0 .. 255
`unsigned int`	2	16-bitowa liczba całkowita bez znaku, zakres: 0 .. 65.535
`unsigned long`	4	liczba całkowita bez znaku, zakres: 0 … 4.294.967.295
`void`	–	stosowany tylko w funkcjach, gdy nie zwraca ona wartości
`word`	2	16-bitowa liczba całkowita bez znaku, zakres: 0 .. 65.535

Arduino - sterowanie diodą LED



Schemat podłączenia
wyprowadzeń modułu Arduino



Schemat podłączenia
diod LED

Arduino - sterowanie diodą LED

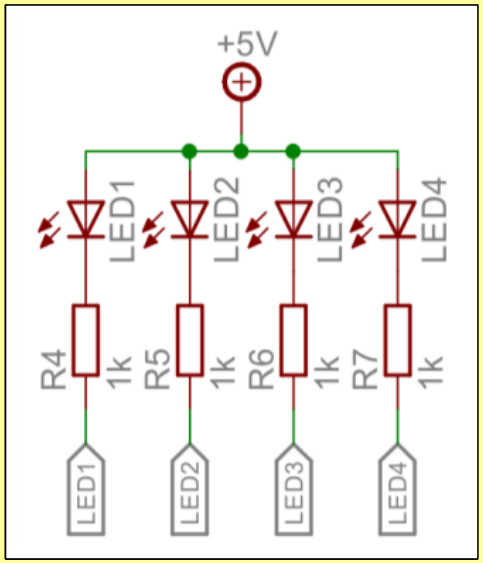
Migająca dioda LED.

```
#include <Arduino.h>

#define LED1_PIN 10

void setup() {
  pinMode(LED1_PIN, OUTPUT);
  digitalWrite(LED1_PIN, HIGH);
}

void loop() {
  digitalWrite(LED1_PIN, LOW);
  delay(500);
  digitalWrite(LED1_PIN, HIGH);
  delay(500);
}
```



Arduino - funkcja pinMode()

```
pinMode(pin, mode)
```

- Konfiguruje określony pin jako wejście lub wyjście
- pin - numer pinu Arduino, dla którego ma zostać ustawiony tryb
- mode - tryb działania określonego pinu Arduino
  - INPUT - pin działa jako wejście, umożliwia odczytanie stanu logicznego pinu (wysoki lub niski)
  - OUTPUT - pin działa jako wyjście, umożliwia ustawienie sygnału cyfrowego pinu (0 V dla stanu LOW lub 5 V dla stanu HIGH) za pomocą funkcji DigitalWrite()
  - INPUT_PULLUP - pin działa jako wejście z włączonym wewnętrznym rezystorem podciągającym
- Funkcja nie zwraca żadnej wartości

Arduino - funkcja DigitalWrite()

```
DigitalWrite(pin, value)
```

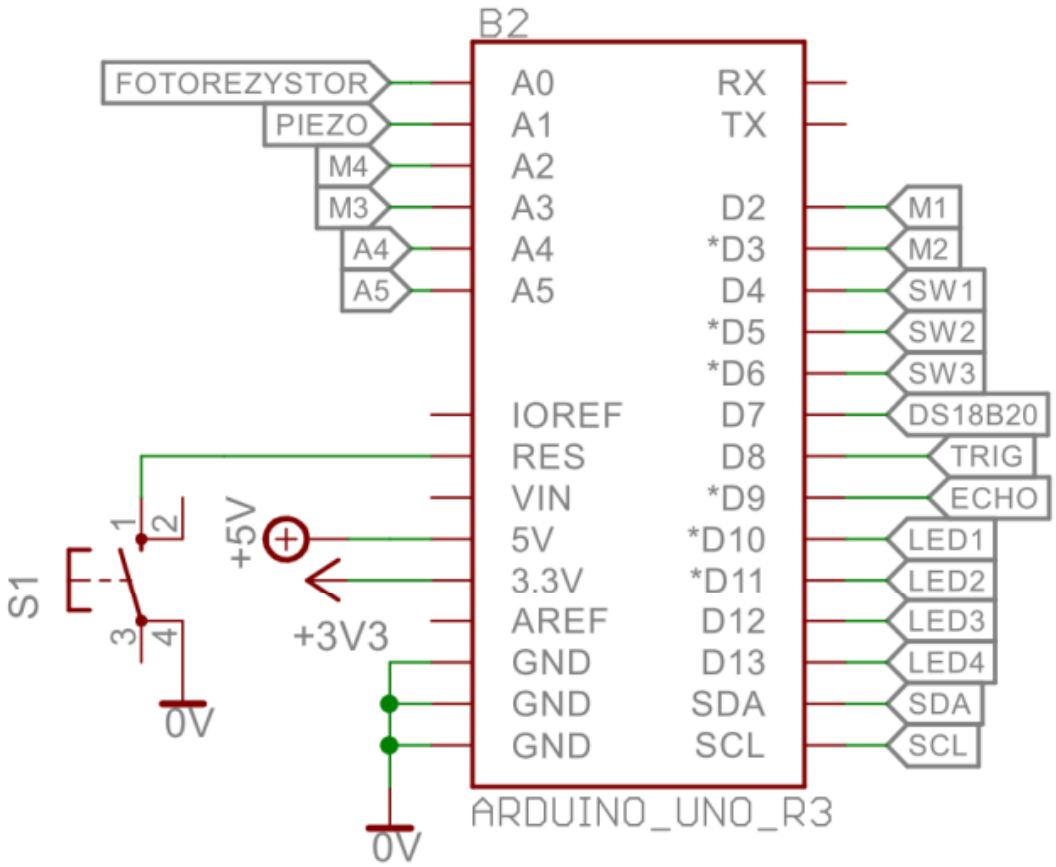
- Zapisuje wartość HIGH lub LOW na cyfrowym pinie
- pin - numer pinu Arduino
- value - zapisywana wartość na cyfrowym pinie (HIGH lub LOW)
- Jeśli pin został skonfigurowany jako OUTPUT za pomocą funkcji pinMode(), jego napięcie zostanie ustawione na wartość 5 V dla stanu HIGH lub 0 V (masa) dla stanu LOW
- Jeśli pin został skonfigurowany jako INPUT, funkcja digitalWrite() włącza (HIGH) lub wyłącza (LOW) wewnętrzny rezystor podciągający na pinie wejściowym
- Funkcja nie zwraca żadnej wartości

Arduino - funkcja delay()

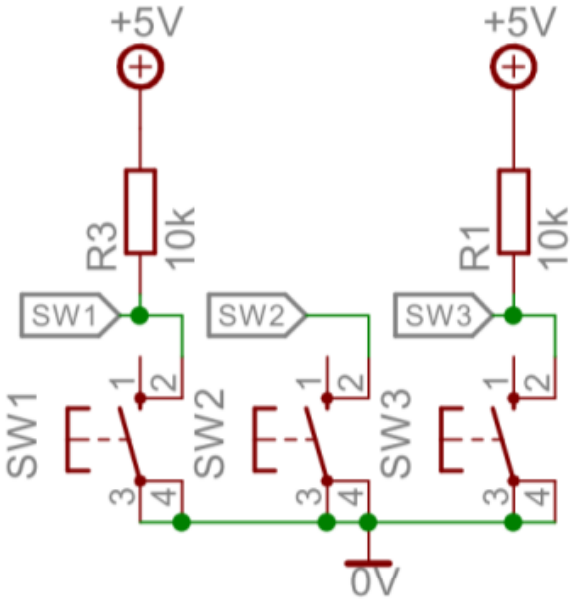
```
delay(ms)
```

- Wstrzymuje wykonanie programu na czas (w milisekundach) określony jako parametr (jedna sekunda to 1000 milisekund)
- ms - liczba milisekund, na którą program zostanie wstrzymany, dozwolone typy danych: unsigned long
- Funkcja nie zwraca żadnej wartości

Arduino - sterowanie przyciskiem



Schemat podłączenia
wyprowadzeń modułu Arduino



Schemat podłączenia
przycisków

Arduino - sterowanie przyciskiem

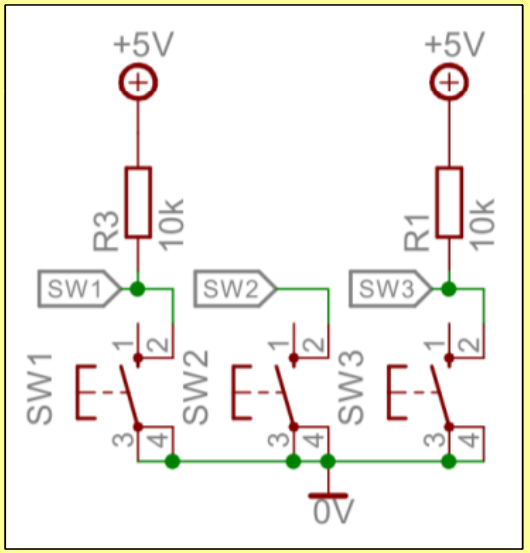
Przycisk zapalający diodę LED na 2 sekundy.

```
#include <Arduino.h>

#define LED1_PIN 10
#define SW1_PIN 4

void setup() {
  pinMode(LED1_PIN, OUTPUT);
  pinMode(SW1_PIN, INPUT);
  digitalWrite(LED1_PIN, HIGH);
}

void loop() {
  if (digitalRead(SW1_PIN) == LOW)
  {
    digitalWrite(LED1_PIN, LOW);
    delay(2000);
    digitalWrite(LED1_PIN, HIGH);
  }
}
```



Arduino - funkcja DigitalRead()

```
DigitalRead(pin)
```

- Odczytuje wartość logiczną (HIGH lub LOW) z określonego cyfrowego pinu
- pin - numer pinu Arduino
- Funkcja zwraca wartość:
  - HIGH - wysoki stan logiczny (5 V)
  - LOW - niski stan logiczny (0 V)

Arduino - sterowanie przyciskiem

Przycisk zapalający i gaszący diodę LED.

```
#include <Arduino.h>

#define LED1_PIN 10
#define SW1_PIN 4

bool led_on = false;

void setup() {
  pinMode(LED1_PIN, OUTPUT);
  pinMode(SW1_PIN, INPUT);
  digitalWrite(LED1_PIN, HIGH);
}
```

Arduino - sterowanie przyciskiem

Przycisk zapalający i gaszący diodę LED.

```
void loop() {
  if (digitalRead(SW1_PIN) == LOW)
  {
    delay(10);
    while (digitalRead(SW1_PIN) == LOW);
    delay(10);

    if (led_on == true)
    {
      led_on = false; digitalWrite(LED1_PIN, HIGH);
    }
    else
    {
      led_on = true; digitalWrite(LED1_PIN, LOW);
    }
  }
}
```

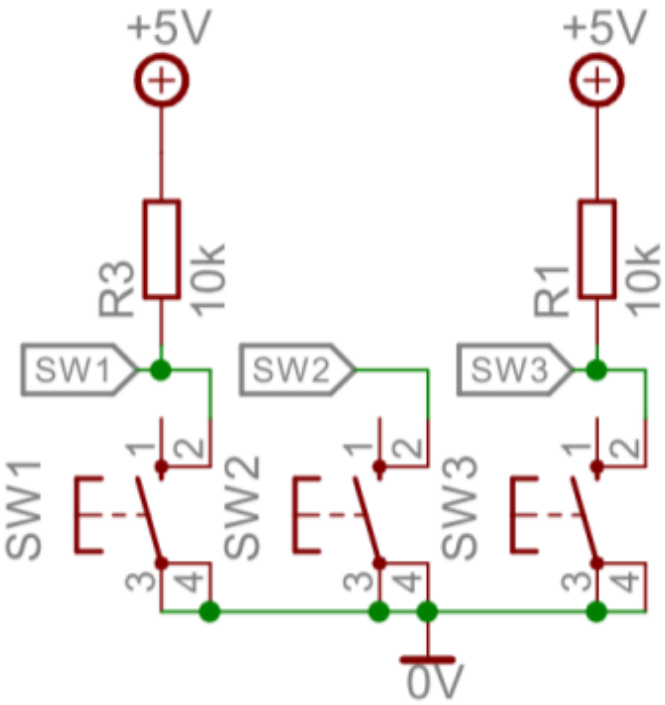
Arduino - sterowanie przyciskiem

- Eliminacja zjawiska drgania styków przycisku

```
delay(10);
while (digitalRead(SW1_PIN) == LOW);
delay(10);
```

- Kiedy styki przycisku stykają się lub rozłączają, zwarcie styków nie następuje do razu, ale może dochodzić do wielokrotnych krótkotrwałych połączeń i rozłączeń
- Dodanie opóźnienia delay(10); pozwala pominąć czas trwania tego zjawiska
- Pętla while powoduje, że program czeka w niej na zwolnienie przycisku - dioda LED1 jest zapalana i gaszona w momencie zwalniania przycisku

Arduino - rezystor podciągający



- Rezystor podciągający jest używany do ustabilizowania sygnału na wejściu cyfrowym układu logicznego
- Jeśli pin wejściowy Arduino nie jest podłączony ani do masy, ani do napięcia zasilania), to na wejściu mogą pojawiać się przypadkowe stany logiczne
- Rezystor podciągający łączy pin wejściowy (SW1, SW3) z napięciem zasilania (+5V), zapewniając stan logiczny HIGH, gdy przycisk (SW1, SW3) nie jest wciśnięty
- Gdy przycisk (SW1, SW3) zostanie wciśnięty, to pin wejściowy (SW1, SW3) zostanie podłączony do masy czyli sygnał na pinie zmieni się na LOW

Koniec wykładu nr 4

Dziękuję za uwagę!