**Nazwa przedmiotu**

**Nazwa kierunku**

**Arduino**

**Podstawy mikrokontrolerów**

Instrukcja do ćwiczenia

**Arduino. Podstawy mikrokontrolerów**

# Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z budową, sposobem działania oraz możliwościami mikrokontrolera Arudino UNO. Zadaniem studentów jest skonstruowanie kilku układów   
na płytce stykowej oraz implementacja związanego z nimi kodu przy pomocy środowiska Arduino IDE.

# Wprowadzenie

Arduino to platforma programistyczna dla systemów wbudowanych oparta na prostym projekcie Open Hardware przeznaczonym dla mikrokontrolerów montowanych w pojedynczym obwodzie drukowanym, z wbudowaną obsługą układów wejścia/wyjścia oraz standardowym językiem programowania zbliżonym do C/C++. Praca z produktami tej włoskiej firmy jest łatwa   
z powodu dużej ilości ogólnodostępnych bibliotek, które zostaną użyte w dalszej części kursu.

Ćwiczenia będą wymagać podstawowych praw fizycznych stosowanych w elektronice. Będzie   
to głównie prawo Ohma pozwalające obliczyć potrzebną rezystancje w celu dobrania odpowiedniego rezystora.

## Najważniejsze elementy płytki Arduino Uno

Obraz zawierający tekst, sprzęt elektroniczny

Opis wygenerowany automatycznie

**Rys. 1.** Uproszczona ilustracja Arduino Uno

1. Złącze USB wykorzystywane do zasilania, programowania oraz komunikacji z komputerem.
2. Koncentryczne złącze zasilania DC 7-12 (dzięki niemu arduino można zasilić z zewnętrznego źródła zasilania).
3. Stabilizator napięcia, który napięcie wejściowe z złącza 2 obniża do 5V.
4. Mikro-kontroler odpowiedzialny za komunikacje pomiędzy komputerem a mikrokontrolerem głównym.
5. Cyfrowe piny wejścia/wyjścia, model UNO wyposażony jest w 14 tego typu pinów, możemy   
   za ich pomocą sterować elementami zewnętrznymi, oraz odbierać z nich sygnały. Złącza oznaczone znakiem tyldy (~) są to piny z funkcją PWM, dzięki której możemy sterować szerokością impulsów. Funkcja ta daje nam możliwość np. sterować jasnością świecenia diody, czy szybkością obrotów silnika.
6. Dioda LED sygnalizująca podłączenie napięcia do arduino.
7. Piny analogowe, najczęściej wykorzystywane do odczytu danych z czujników analogowych.
8. Piny GND to masy płytki, 5 V oraz 3 V służą do wyprowadzenia zasilania poza płytkę. Pin Vin służy do podłączenia zewnętrznego zasilania, połączony jest z gniazdem DC 7-12V. Pin IOREF sygnalizuje modułom rozszerzającym arduino, na jakich napięciach pracuje płytka.
9. 8-bitowy mikro-kontroler firmy Atmel – AVR ATmega328 pełniący funkcje serca płytki. To on przetwarza wszystkie dane i steruje wszystkimi wejściami/wyjściami na płytce.
10. Diody LED sygnalizujące transmisję do/z komputera.
11. Dioda LED do dyspozycji użytkownika. Połączona z pinem 13, wykorzystywana do wizualizacji różnych operacji. Nie może zostać przypisana do innego pinu.

## Przykładowe zastosowanie arduino

### Przykład 1 – Prosty obwód z 1 diodą.

W ramach wprowadzenia i zobaczenia jak arduino działa zostanie zaprezentowany prosty przykład zapalenia diody LED za jego pomocą.

* Pierwszym krokiem będzie umieszczenie elementów na płytce stykowej. Powinno wyglądać to tak:

Obraz zawierający tekst, sprzęt elektroniczny

Opis wygenerowany automatycznie

**Rys. 2.** Przykładowe połączenie diody do arduino.

* Kod napisany w Arduino IDE:



### Przykład 2 – Prosty obwód z przyciskiem i diodą.

Rozbudowanie przykładu pierwszego o nowy element. Będzie nim przycisk podłączony do pinu nr. 5. W bardziej rozbudowanych układach przy większej ilości stosowanych pinów można łatwo się pogubić. Z pomocą przychodzą aliasy definiowane za pomocą dyrektywy **#define**.

* Połączenie elementów na płytce:

Obraz zawierający tekst, sprzęt elektroniczny

Opis wygenerowany automatycznie

**Rys. 3.** Przykładowe połączenie diody i przycisku do arduino

* Kod napisany w Arduino IDE:





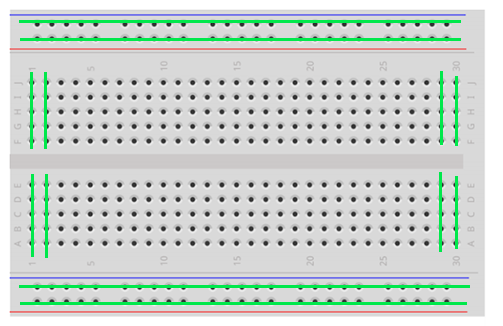
**Uwagi**

Zamiast pinu 7 i 5 można wybrać dowolny z przedziału 2-13 (oczywiście należy   
to uwzględnić zarówno w połączeniach fizycznych jak i kodzie).

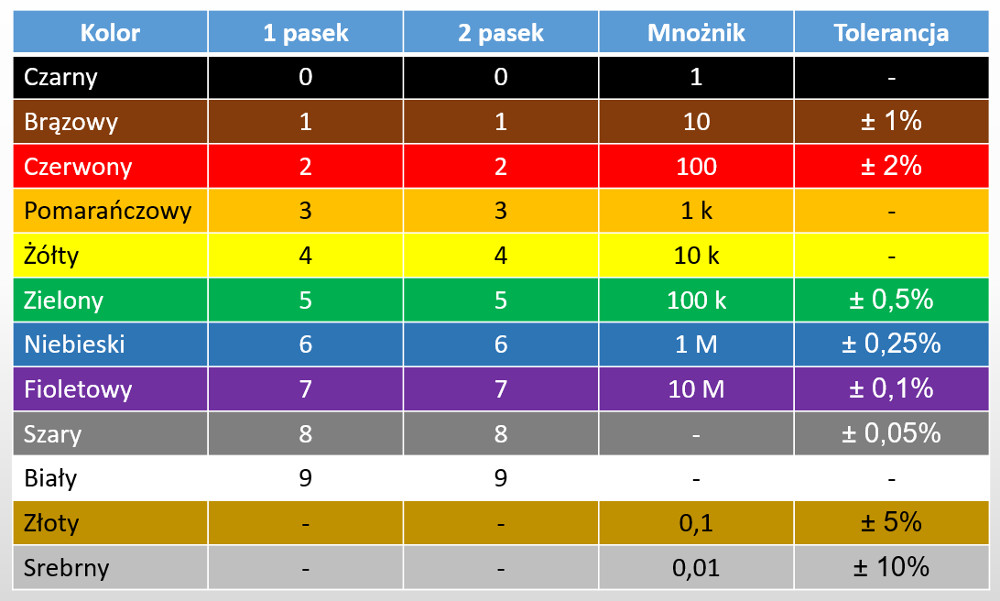
Powodem dla którego w drugim przykładzie w 13 linijce oczekujemy że wciśnięcie przycisku spowoduje zwrócenie z funkcji **digitalWrite(button)** wartości LOW jest to, że przycisk jest połączony z masą.

Uwaga: Bardzo ważnym aspektem jest podłączanie większości elementów (diody, buzzery) do arduino przez rezystory! Jego brak może doprowadzić do TRWAŁEGO uszkodzenia podłączanych urządzeń, a nawet arduino!

W ramach pierwszego ćwiczenia proszę spróbować połączyć czerwoną diodę i przycisk   
z arduino oraz zaprogramować to w taki sposób, aby dioda zaświecała się gdy przycisk jest wciśnięty. Pamiętać należy o sposobie wewnętrznych połączeń płytki stykowej oraz sposobie odczytywania rezystancji z rezystorów.



**Rys. 4.**  Schemat połączeń wewnętrznych płytki stykowej



**Rys. 5.** Tabela pozwalająca odczytać rezystancje z rezysora.

## Pomocne przydatne przy wykonywaniu ćwiczeń

* **pinMode(Pin, Mode);**

Funkcja umożliwiająca wybór czy dany **Pin** jest wejściem czy wyjściem. Jest on liczbą całkowitą z zakresu <0,13>.  **Mode**  to {INPUT, OUTPUT, INPUT\_PULLUP}. W przypadku, gdy chcemy sterować wyjściem (np. diodami), należy użyć OUTPUT. Natomiast dla elementów pokroju przyciski używa się INPUT\_PULLUP. Tryb ten będzie wykorzystywany za każdym razem, gdy do arudino będzie podłączany przełącznik.

* **digitalWrite(Pin, Condition);**

Po wcześniejszym użyciu funkcji **pinMode()** można ustawić stan logiczny na wyjściu (tym samym włączyć diodę). Służy do tego funkcja **digitalWrite()** gdzie **Condition** jest stanem logicznym mającym wartości {HIGH, LOW} a **Pin** to numer pinu z poprzedniej funkcji. Z racji,   
że diodę łączymy z masą, dioda zapali się po dostarczeniu jej przez arduino stanu wysokiego   
za pomocą **digitalWrite(Pin, HIGH)**;

* **delay(Sec);**

Funkcja realizująca opóźnienie programu. Jako argument przyjmuje liczbę milisekund. **delay(1000);** jest równoznaczne z zatrzymaniem programu na 1 sekundę.

* **#define <name> <numer>**

Dyrektywa #define pozwala na zdefiniowanie symbolu, który będzie podmieniony przed kompilacją w każdym miejscu programu. Dla przykładu zastosowanie **#define red\_diode 13** spowoduje zastąpienie napisu **red\_diode** w kodzie na 13.

* **digitalRead(Pin);**

Funkcja pozwala na odczytywania wejścia i pozwala uzależnić pewne operacje w zależności od stanu. Zwraca wartość logiczną {HIGH, LOW}. Wciśniecie przycisku powinno powodować zwarcie wejścia z masą arduino. Wynikiem czego w momencie wciśnięcia przycisku i próby użycia tej funkcji otrzymamy wartość LOW.

* **Serial.begin(Speed);**

Funkcja uruchamia transmisję pomiędzy komputerem i arduino. **Speed** określa prędkość transmisji, która w tym przypadku wynosi 9600 baud/sec.

* **Serial.println(Val);**

Funkcja pozwala na wysłanie ciągu znaków **Val** i wyświetlenia ich w terminalu.

# Program ćwiczenia

Do instrukcji w pliku zip dołączone zostały 3 foldery zawierające zadania, których treść jest zamieszczona na początku pliku w postaci komentarza. Celem studentów jest zrealizowanie każdego z nich w ustalonej kolejności pamiętając o podstawowych zasadach BHP oraz prawach którymi rządzą się wszystkie elementy elektroniczne.

W razie niejasności zachęcam do indywidualnego kontaktu.