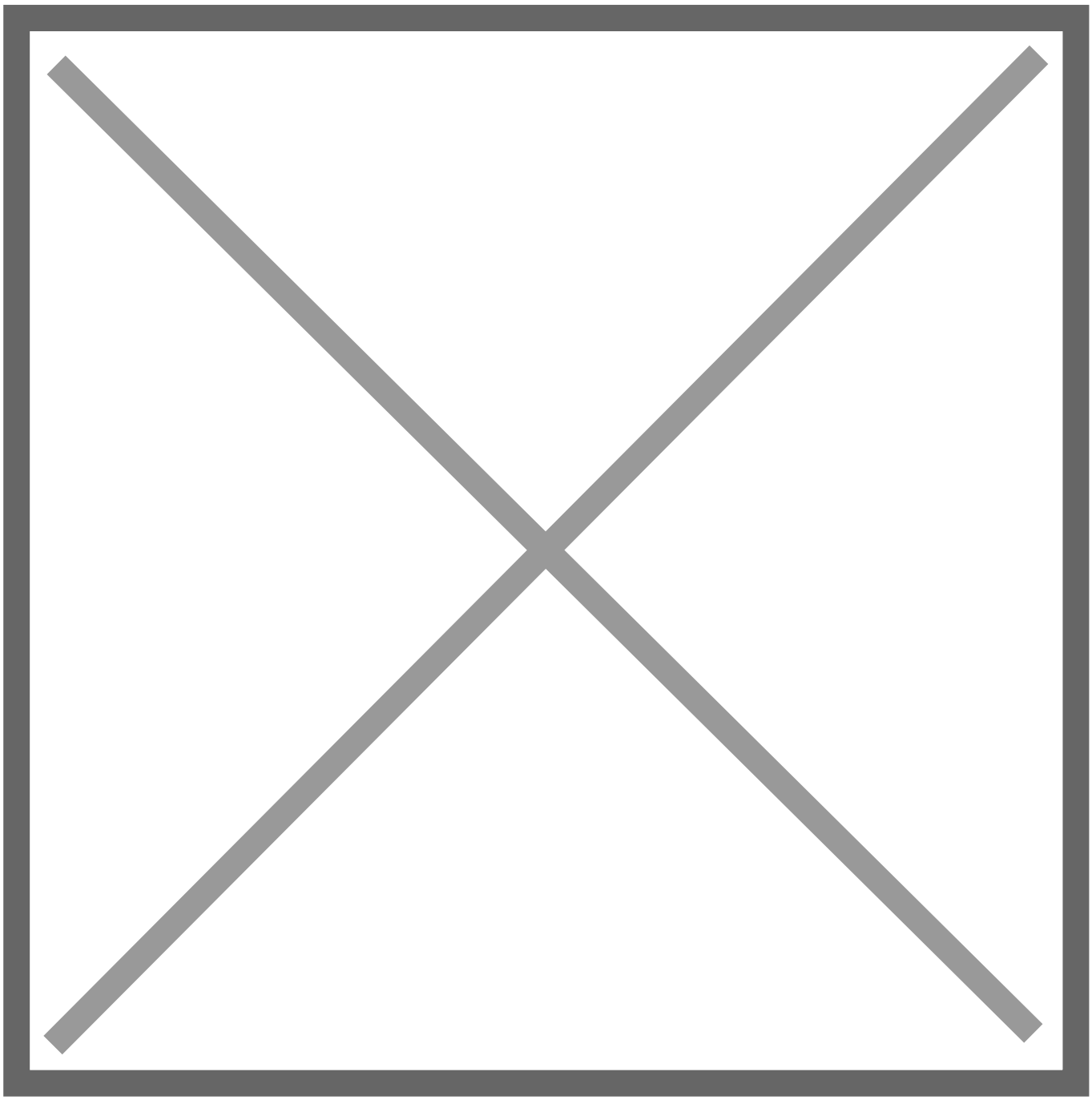
Arduino

ARDUINO - Ma?e urz?dzenie, du?o mo?liwo?ci



Czym jest Arduino?

- Opensourceowa konstrukcja

  https://opensource.org/

  To nowy wynalazek bo całość rozpoczęła się w latach 70!

  Odnosi się zarówno do oprogramowania jak i sprzętu/hardware

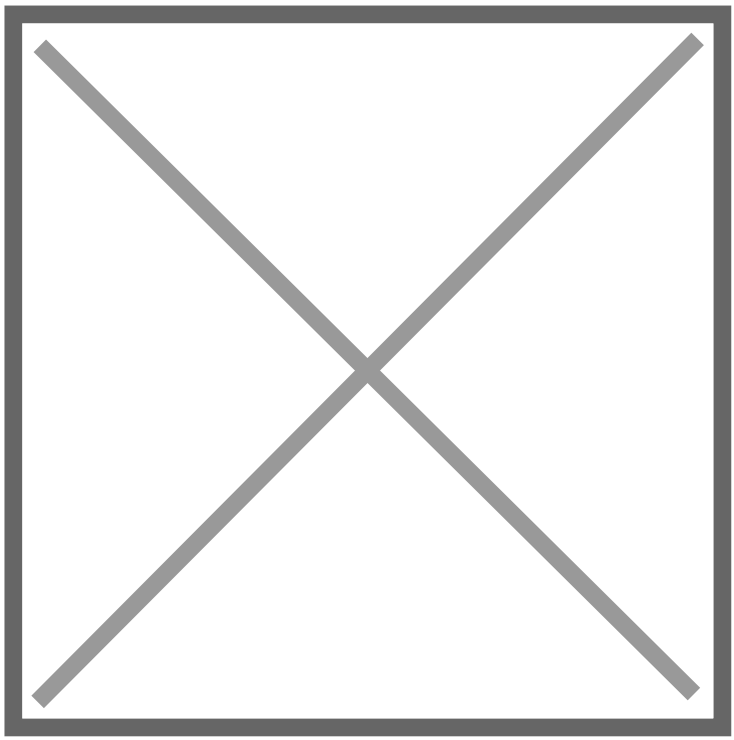
  Każdy może modyfikować przygotowywany przez developerów oprogramowanie lub sprzęt

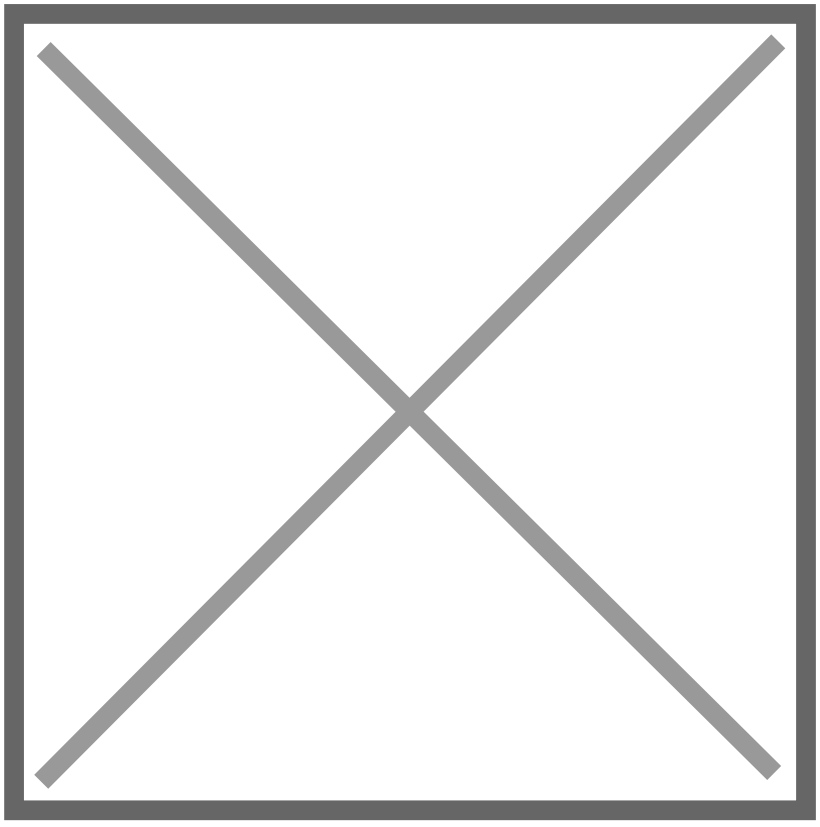
  e- nable - protezy drukowane w 3D

  Arduino - mikrokontroles

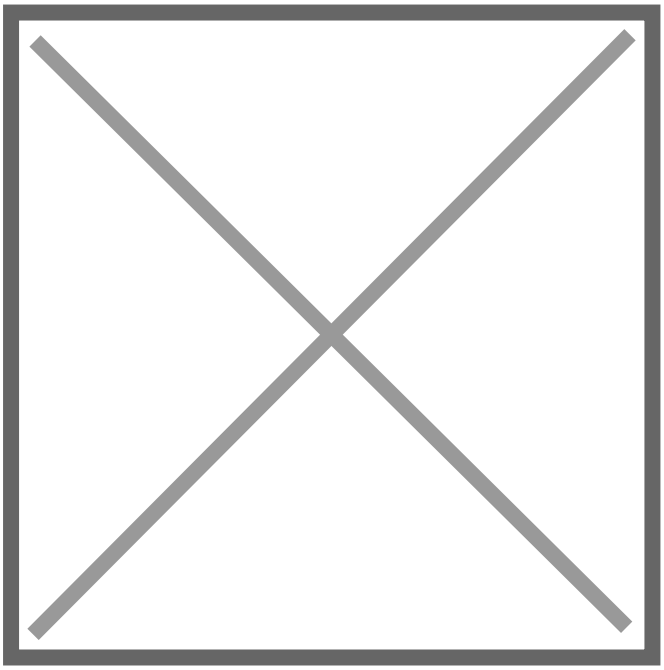
  Jądro Linuxa jest open source dzieki czemu wiele dystrubucji Linuxa jest tez otwartoźródłowa np. Debian czy Ubuntu

- **Zaprojektowania do tworzenia interaktywnych projektów**

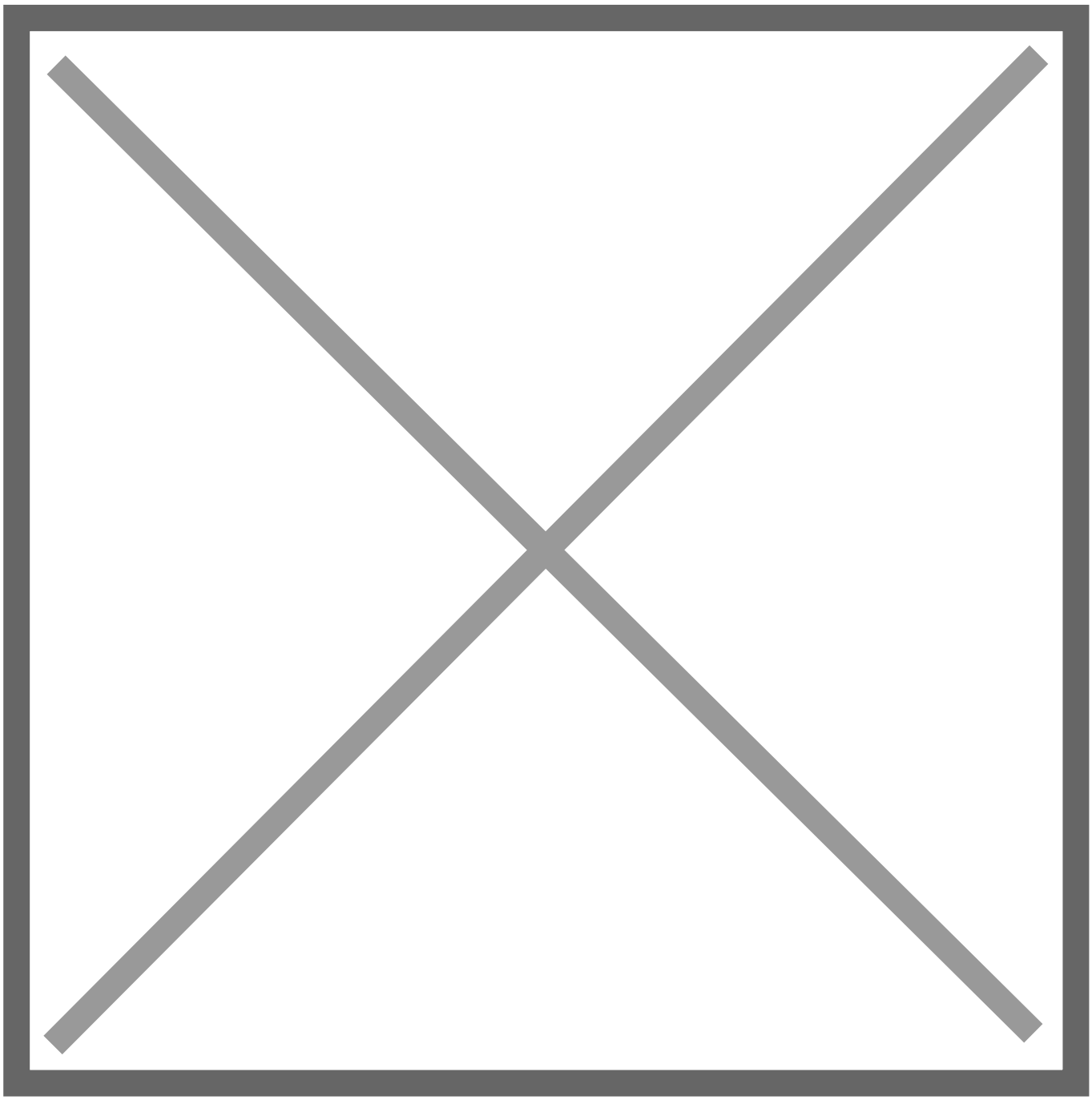




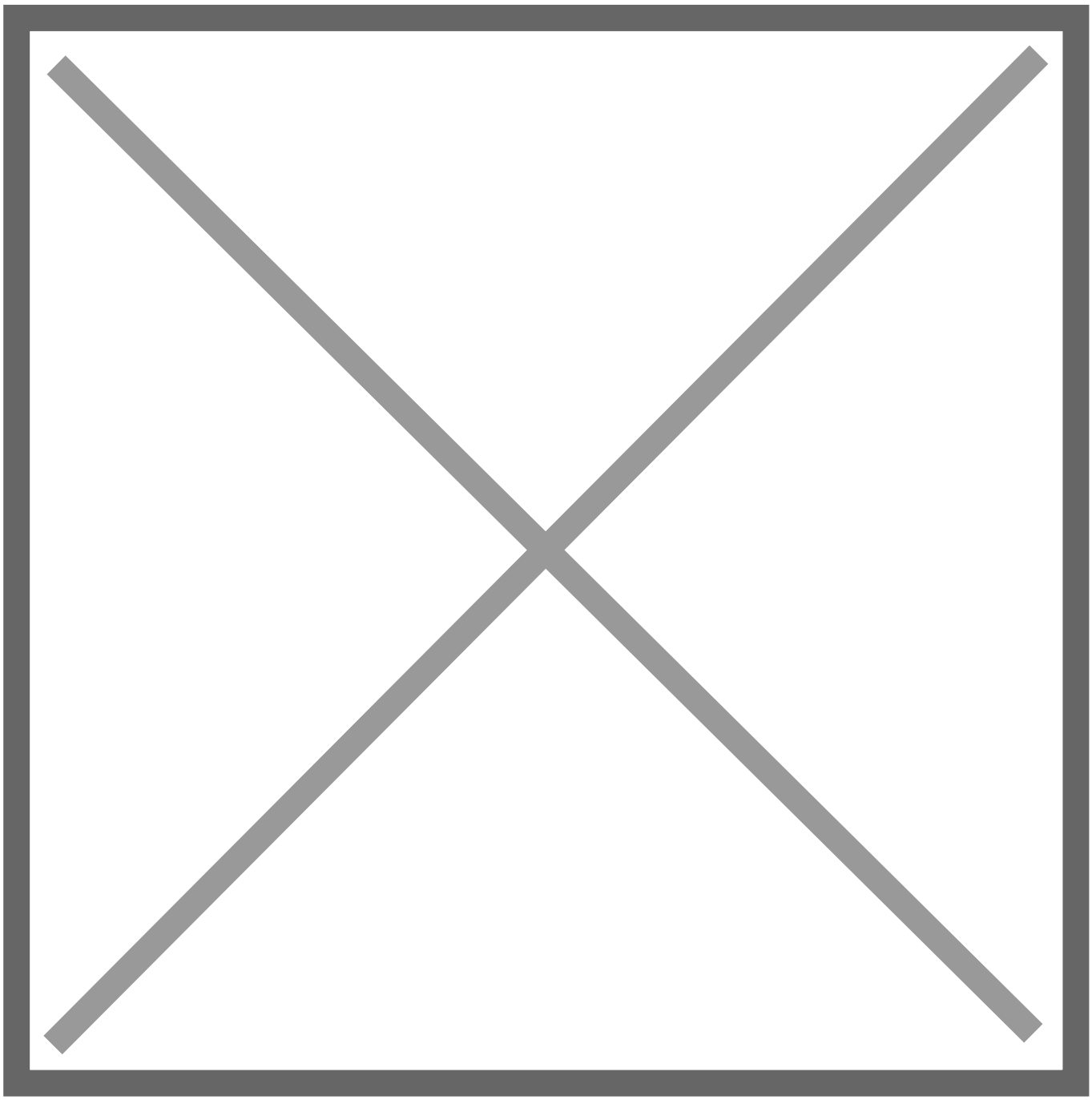
- Prostota z korzystania



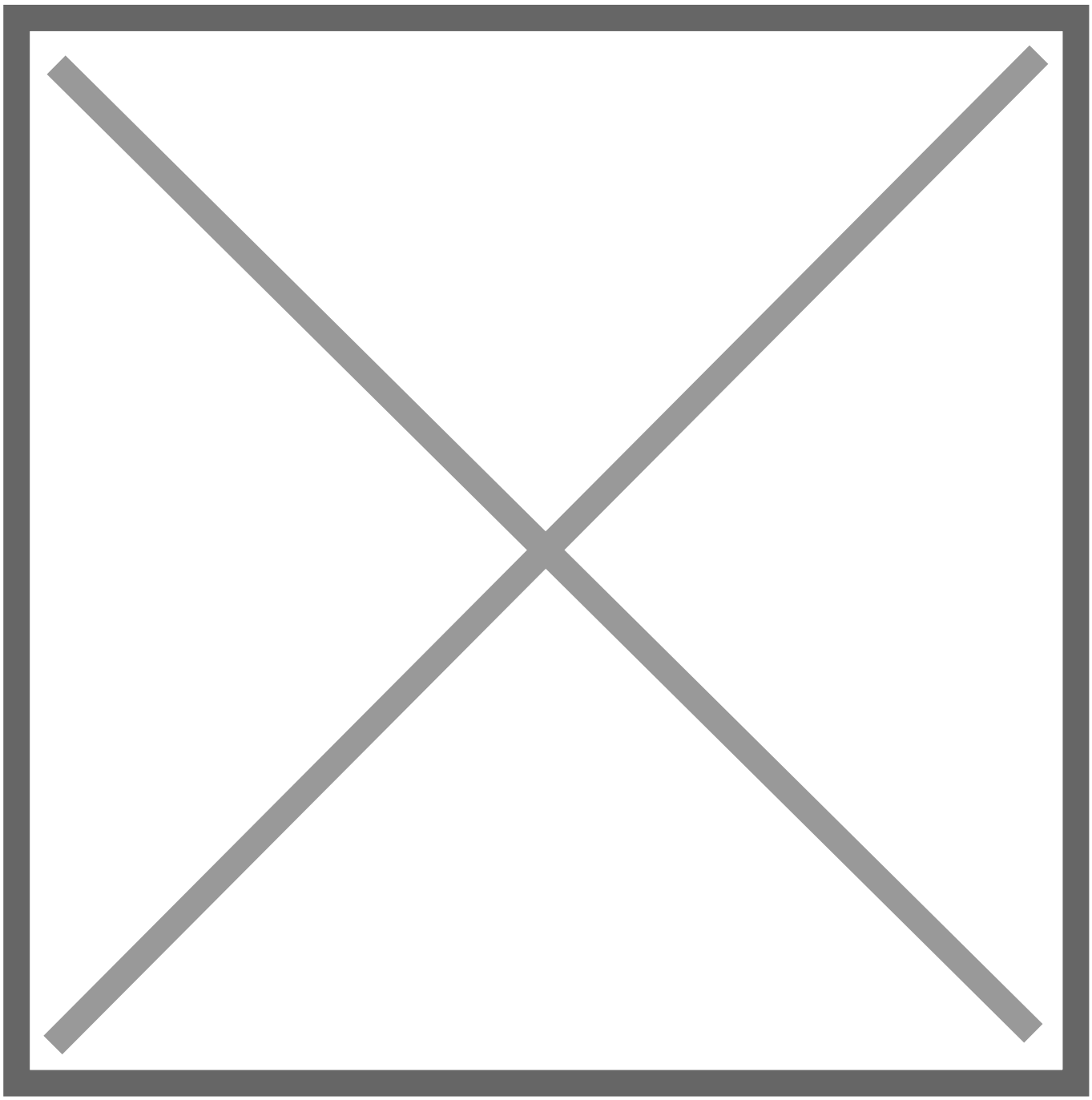
- Duże community



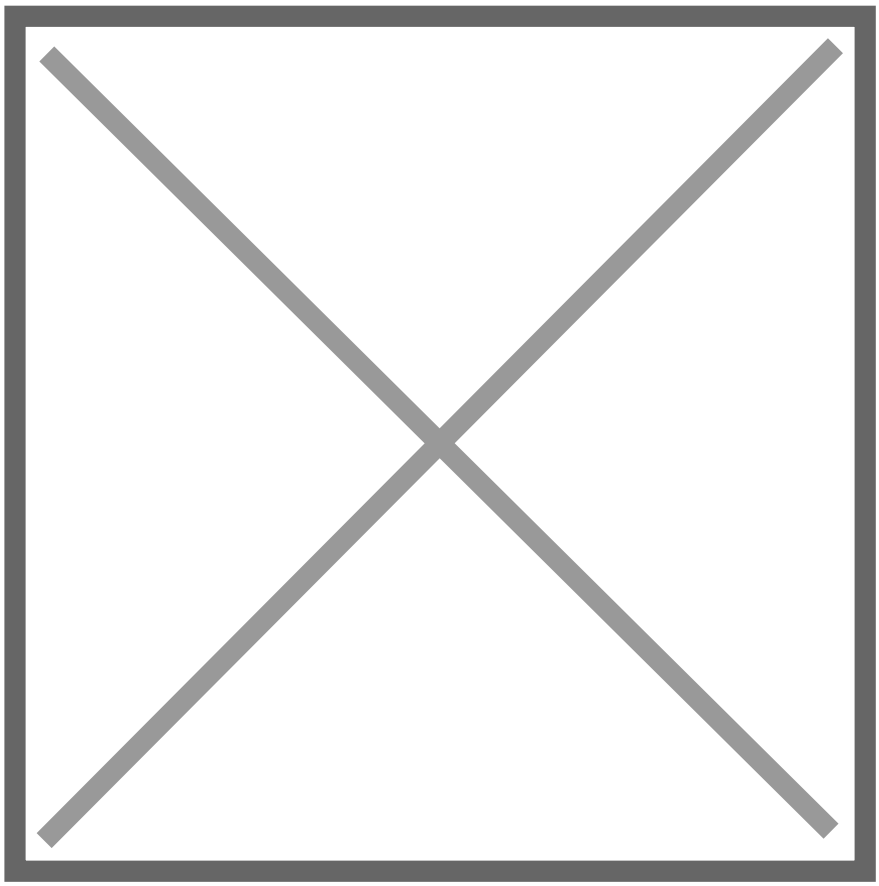
- Prosta dokumentacja



- Duża liczba współpracujących urządzeń i sensorów

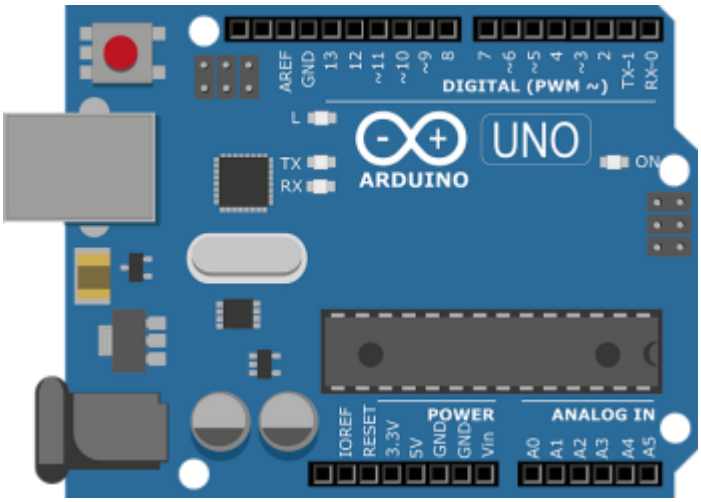


- Język programowania C++ ale jest możliwość programowania blokowego



> Open Source to jak budowanie wspólnego zamku z klocków Lego – każdy dorzuca swój element, a końcowy wynik jest o wiele większy i lepszy niż suma poszczególnych kawałków. I to właśnie dzięki tej filozofii projekty takie jak Arduino mogły zmienić świat technologii!

Z czego si? sk?ada Arduino?



- Port USB B
- Przycisk RESET
- Piny zasilające 5V i 3v3
- Piny GND
- Piny cyfrowe 0/1 (0-13)
- Piny Analogowe 0 - 1023 (A0-A5)

Arduino - Nie?atwa Historia

2000 - Massimo Banzi, jeden z założycieli, był profesorem w tej szkole, a jego celem było stworzenie taniej i łatwej w użyciu platformy dla studentów. Wspólnie z Davidem Cuartiellesem, Tomem Igoe, Gianlucą Martino i Davidem Mellisem opracowali pierwsze płytki i oprogramowanie Arduino.

2005 - gdy powstała pierwsza płytka, platforma szybko zdobyła popularność. Stała się narzędziem dostępnym dla hobbystów, artystów i inżynierów, oferując otwartą architekturę, którą każdy mógł zmieniać i dostosowywać. Dzięki niskiemu kosztowi oraz wsparciu społeczności, Arduino przyciągało coraz więcej użytkowników.

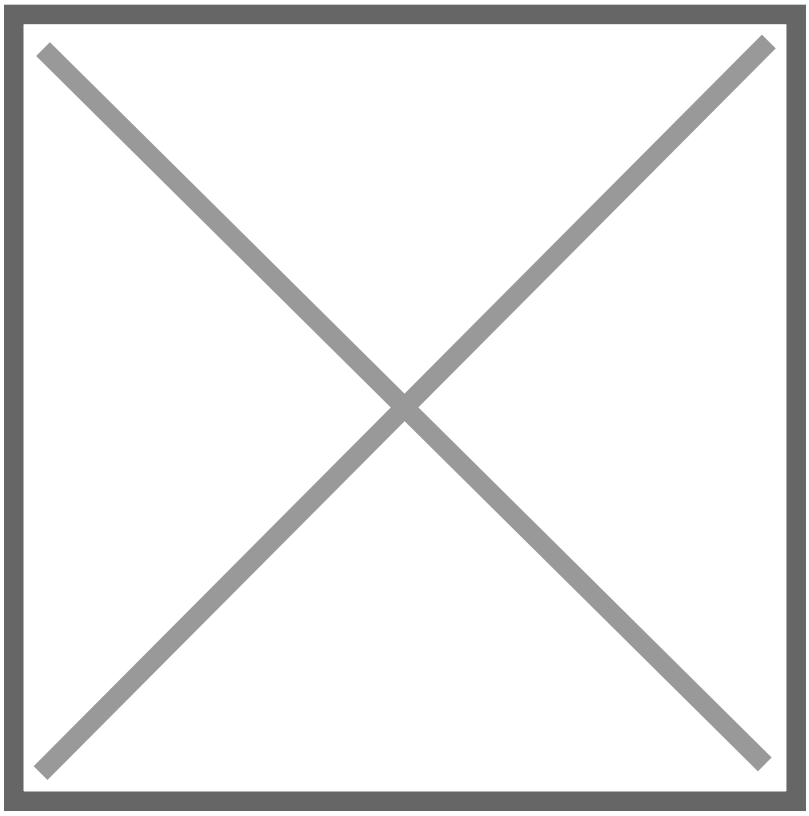
W miarę jak projekt rósł, pojawiły się tarcia między założycielami, szczególnie dotyczące praw do marki “Arduino”. W pewnym momencie doszło do podziału, gdy Gianluca Martino, odpowiedzialny za produkcję płytek Arduino, zarejestrował nazwę “Arduino” w swoim imieniu we Włoszech, bez porozumienia z innymi założycielami. To doprowadziło do konfliktu, w wyniku którego powstały dwie równoległe firmy.

•Arduino LLC (zarządzana przez Banzi’ego i pozostałych twórców)

•Arduino SRL (kierowana przez Martino, odpowiedzialna za produkcję płytek)

Obie firmy zaczęły niezależnie od siebie produkować i sprzedawać produkty o nazwie “Arduino”, co spowodowało niemałe zamieszanie w społeczności. W tym czasie obie firmy wypuszczały własne wersje płytek oraz oprogramowania, a konflikt trwał w sądzie o prawa do marki “Arduino”.

2017 - doszło jednak do porozumienia między stronami. Arduino LLC i Arduino SRL połączyły się w jedną firmę, która działa obecnie jako Arduino Holdings oraz Arduino AG. Ostatecznie platforma kontynuowała swój rozwój, a społeczność Arduino nadal się rozrasta.

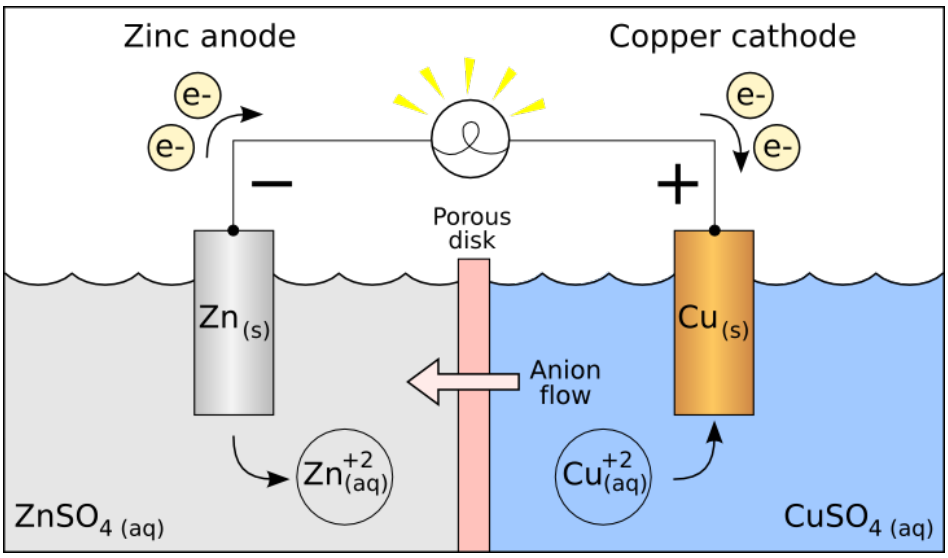


Katoda i Anoda oraz LED

Anoda i Katoda - Co to i dlaczego ta informacja jest dla nas wa?na

Anoda - Słowo “anoda” pochodzi z greckiego słowa “anodos”, co oznacza “droga w górę” lub “pod górę”. Nazwa ta odnosi się do tego, że w przypadku elektrochemicznych urządzeń (takich jak ogniwa elektrochemiczne) prąd płynie “w górę” – czyli do anody wpływają dodatnie ładunki (lub w konwencjonalnym rozumieniu prądu: prąd płynie do anody).

Katoda - Słowo “katoda” pochodzi od greckiego słowa “kathodos”, co oznacza “droga w dół”. Oznacza to, że w przypadku tych samych urządzeń prąd płynie “w dół”, czyli z katody. Jest to biegun, z którego prąd “ucieka” lub wypływa.



https://www.scribd.com/document/600800539/Galvanic-Cells-intro

LED

Diody LED mają dwie nóżki:

•Anoda (dłuższa nóżka): Podłączasz ją do dodatniego zasilania (+).

•Katoda (krótsza nóżka): Podłączasz ją do ujemnego zasilania (–, GND).

Pierwsze kroki z Arduino i Tinkercad

Poniżej Piotrek prowadzi was przez pierwsze kroki z prądem oraz programowaniem blokowym.

https://www.youtube.com/embed/n93QcT-FZl8?t=3s

https://www.youtube.com/embed/_ISBtIXTnyk?t=1s

Czemu void?

Słowo void w Arduino (i w ogóle w języku C/C++) oznacza typ zwracany przez funkcję. W tym przypadku void wskazuje, że funkcja nic nie zwraca po zakończeniu swojego działania.

Funkcjach takich jak void setup() lub void loop(), robi to i nie przechowuje żadnych wyników z tych funkcji, ponieważ są one oznaczone jako void – czyli nie zwracają żadnej wartości.

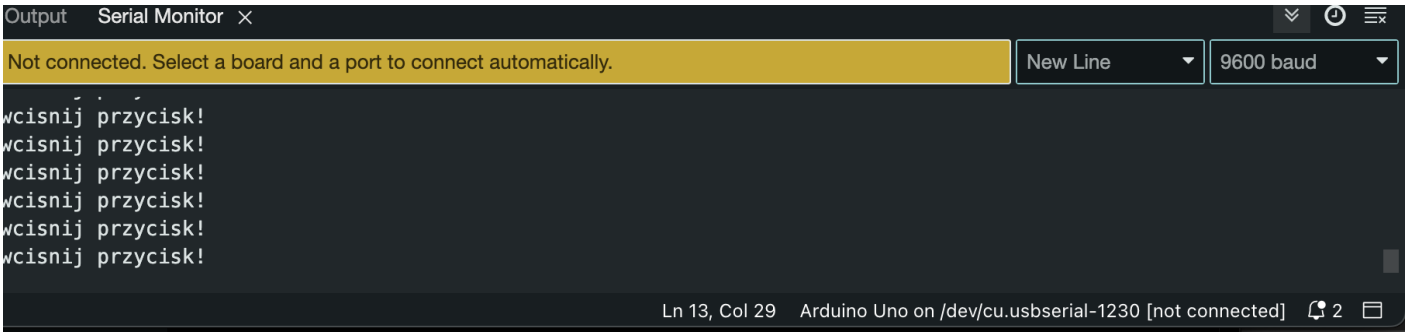
Co to oznacza?

•W przypadku funkcji setup(), Arduino wykonuje ustawienia, takie jak konfiguracja pinów czy inicjalizacja komunikacji, ale po zakończeniu tej funkcji nie ma potrzeby zwracania jakiejkolwiek wartości, ponieważ jej jedynym celem jest wstępna konfiguracja.

•W przypadku loop(), Arduino wykonuje zawarte w niej instrukcje, po czym zaczyna je od nowa, nie przechowując niczego z poprzedniego przebiegu. Po prostu zapętla działanie – w nieskończoność – wykonując kod raz za razem.

```
void setup() {
  //wszytstko tutaj dzieje się tylko raz
}
void loop() {
//tutaj działamy w pętli
}
```

Monitor portu szeregowego



Monitor portu szeregowego ma nam pomóc w odczytywaniu danych z czujników i sensorów. To tam właśnie wyświetlają się wszystkie dane które zbieramy i **printujemy** poprzez kod.

Serial.println(tutaj wpisujemy dane) - możemy printować jakąś daną albo informacje w “”

Serial.print(tutaj wpisujemy dane) - możemy printować jakąś daną albo informacje w “”

Serial.begin(9600)

Serial.begin(prędkość) - rozpoczyna komunikację szeregową Arduino z komputerem. To dzięki temu komputer może uzyskać informacje z czujników podłączonych do Arduino, łatwiej nam szukać rozwiązań problemów z niedziałającym układem.

prędkość: Jest to wartość, która określa szybkość transmisji danych (w bitach na sekundę), czyli baud rate. Wartość ta mówi, ile bitów na sekundę będzie przesyłanych między Arduino a urządzeniem odbierającym (np. komputerem). Najczęściej stosowaną wartością jest 9600, ale inne możliwe wartości to np. 4800, 14400, 115200, w zależności od wymagań projektu.

Komunikacja szeregowa

Komunikacja szeregowa (ang. Serial Communication) to sposób przesyłania danych, w którym bity (jednostki informacji) są wysyłane jeden po drugim przez jeden kanał (jedną linię transmisyjną), w przeciwieństwie do komunikacji równoległej, gdzie wiele bitów jest przesyłanych jednocześnie przez wiele linii.

Przyk?ad

```
void setup() {

  Serial.begin(9600);  // Rozpocznij komunikację szeregową z prędkością 9600 baud

}



void loop() {

  Serial.println("Witaj świecie!");  // Wydrukuj tekst na monitorze szeregowym

  delay(1000);  // Czekaj 1 sekundę

}
```

Sta?e i zmienne

Sta?e i zmienne

Constans & Variables

- **Stałe** to wartości, które nie zmieniają się w czasie, np. numer PIN, delay (opóźnienie), maksymalną wartość.
- **Zmienne** są niezbędne, gdy wartości muszą się zmieniać w trakcie działania programu, np. temperatura odczytana z czujnika, liczba obrotów silnika, czas trwania sygnału, waga.

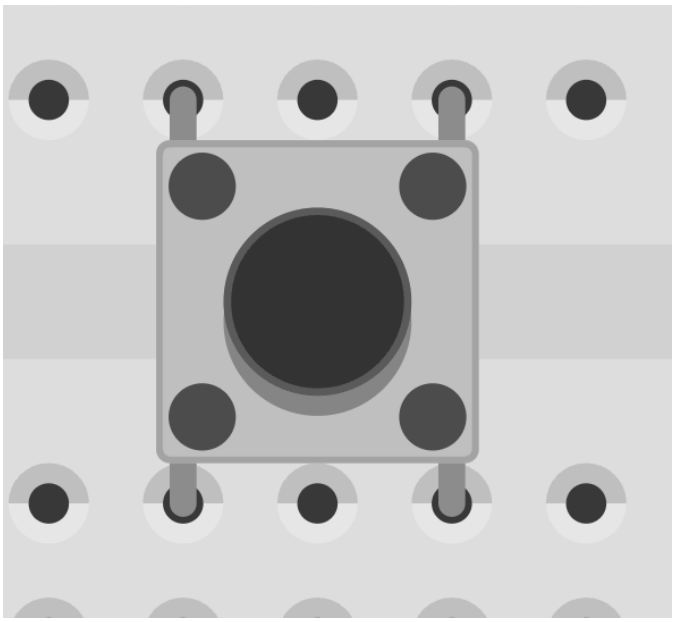
Ró?nice

Stałe	Zmienne
• Deklarowana za pomocą const. • Wartość nie może się zmieniać. • Używana do oznaczania rzeczy stałych (np. numerów pinów). • Poprawia czytelność i bezpieczeństwo kodu. const int ledPin = 13;	• Deklarowana bez const. • Wartość może się zmieniać. • Używana do przechowywania dynamicznych danych (np. pomiarów z czujników). • Pozwala na elastyczne działanie programu. int brightness = 255

Przycisk monostabilny

Czyli jaki?

Przycisk monostabilny (inaczej zwany przyciskiem chwilowym) to rodzaj przycisku, który **pozostaje w jednym stanie** (zazwyczaj otwartym, czyli “wyłączonym”) i **zmienia swój stan tylko wtedy, gdy jest wciśnięty.** Gdy puścisz przycisk, wraca on do swojego pierwotnego stanu. Dlatego mówi się, że ma tylko jeden stabilny stan – stan wyłączony (otwarty).



Budowa przycisku monostabilnego

1. **Obudowa:** Przycisk jest zamknięty w obudowie, która utrzymuje wszystkie jego części wewnętrzne. Obudowa chroni mechanizm i zapewnia, że użytkownik może wygodnie naciskać przycisk.
2. **Dwa styki:** Wewnątrz przycisku znajdują się dwa metalowe elementy zwane stykami. Jeden z nich jest podłączony do źródła zasilania (np. pin Arduino lub inny obwód), a drugi do elementu, który ma zostać aktywowany (np. masa lub inny punkt obwodu). Gdy te dwa styki zostaną połączone, zamykają obwód, co pozwala prądowi przepływać.
3. **Mechanizm sprężynowy:** Kluczowym elementem przycisku monostabilnego jest sprężyna. Sprężyna utrzymuje przycisk w jego stabilnym stanie – zazwyczaj otwartym (czyli takim, w którym obwód jest przerwany). Sprężyna sprawia, że po puszczeniu przycisku wraca on do swojego pierwotnego stanu.

Jak dzia?a przycisk monostabilny?

-
  - Stan spoczynku (brak nacisku): Kiedy przycisk nie jest wciśnięty, obwód jest przerwany, co oznacza, że prąd nie płynie. Mówimy, że obwód jest otwarty.
  - Stan aktywacji (nacisk): Kiedy przycisk jest wciśnięty, zamyka on obwód i pozwala na przepływ prądu. W tym momencie obwód jest zamknięty.

  Po puszczeniu przycisku, automatycznie wraca on do stanu otwartego, przerywając obwód. Stąd nazwa “monostabilny”, ponieważ tylko jeden stan (otwarty) jest stabilny bez zewnętrznej ingerencji.

Schemat dla Arduino

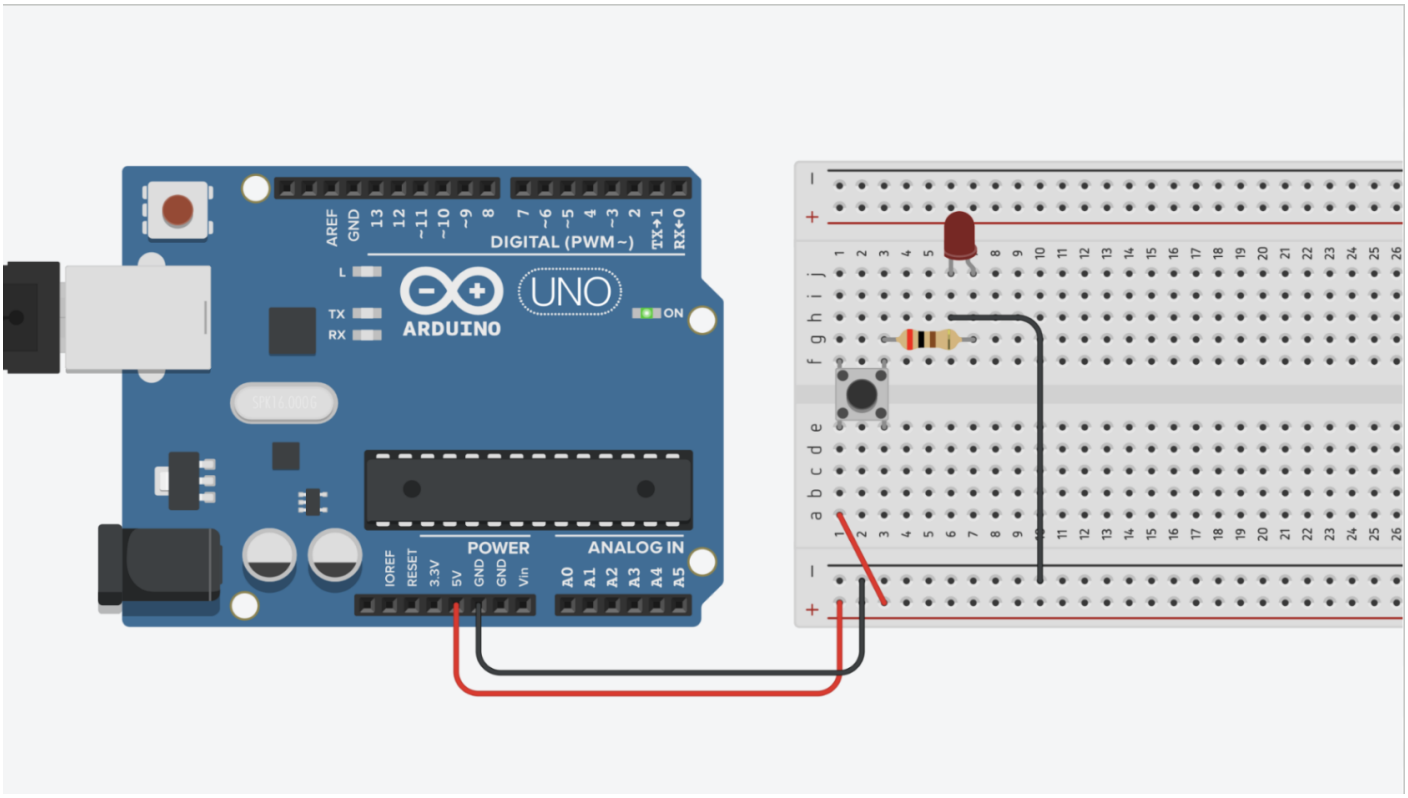
Schemat dla włącznika działającego z LEDem bez użycia kodu.
1. 1x przycisk monostabilny

2. 1x LED

3. rezystor 180 Ohm

4. Arduino UNO

5. 4x Jumpery



Jak dzia?a powy?szy schemat:

1. Stan początkowy (przycisk nie wciśnięty):
• W stanie spoczynku (kiedy przycisk nie jest wciśnięty), styki wewnątrz przycisku są rozłączone.
• Oznacza to, że obwód jest otwarty, więc prąd nie przepływa.
• W takim stanie prąd nie może przejść przez przycisk, ponieważ nie ma fizycznego połączenia między stykami.

2. Wciśnięcie przycisku:
• Kiedy naciskasz przycisk, mechanizm sprężynowy pozwala, aby część przycisku przemieszczała się w dół.
• Podczas tego ruchu, styki wewnątrz przycisku łączą się, zamykając obwód.
• Gdy styki zostaną połączone, prąd może przepływać przez przycisk, co oznacza, że urządzenie podłączone do obwodu (np. dioda LED lub silnik) zostaje aktywowane.

3. Po zwolnieniu przycisku:
• Gdy puścisz przycisk, sprężyna wewnątrz automatycznie powraca go do pierwotnej pozycji.
• Styki wewnątrz przycisku rozłączają się, co powoduje przerwanie obwodu.
• Prąd przestaje płynąć, a urządzenie podłączone do obwodu przestaje działać (np. dioda LED gaśnie).

Schemat dla przycisku podającym sygnał na pin 8 wywołującym

1. 1x przycisk monostabilny
2. rezystor 10 kOhm
3. Arduino UNO
4. 4x Jumpery

Dlaczego jest nam potrzebny rezystor o warto?ci a? 10 kOhm? - Pulldown

Rezystor typu ***“pull-down”*** w układzie z przyciskiem monostabilnym jest potrzebny, aby zapewnić stabilny stan logiczny (niski - LOW) na wejściu cyfrowym Arduino, gdy przycisk nie jest wciśnięty. Pomaga to uniknąć niepożądanych zakłóceń, które mogą prowadzić do przypadkowych odczytów sygnału.

Problem bez rezystora pull-down:

Jeśli podłączysz przycisk bez rezystora typu “pull-down”, pin wejściowy Arduino może znajdować się w tzw. stanie pływającym (ang. “floating”) – oznacza to, że pin nie jest ani podłączony do napięcia, ani do masy. W takim stanie pin może losowo odbierać zakłócenia z otoczenia, co może prowadzić do nieprawidłowych odczytów sygnału.

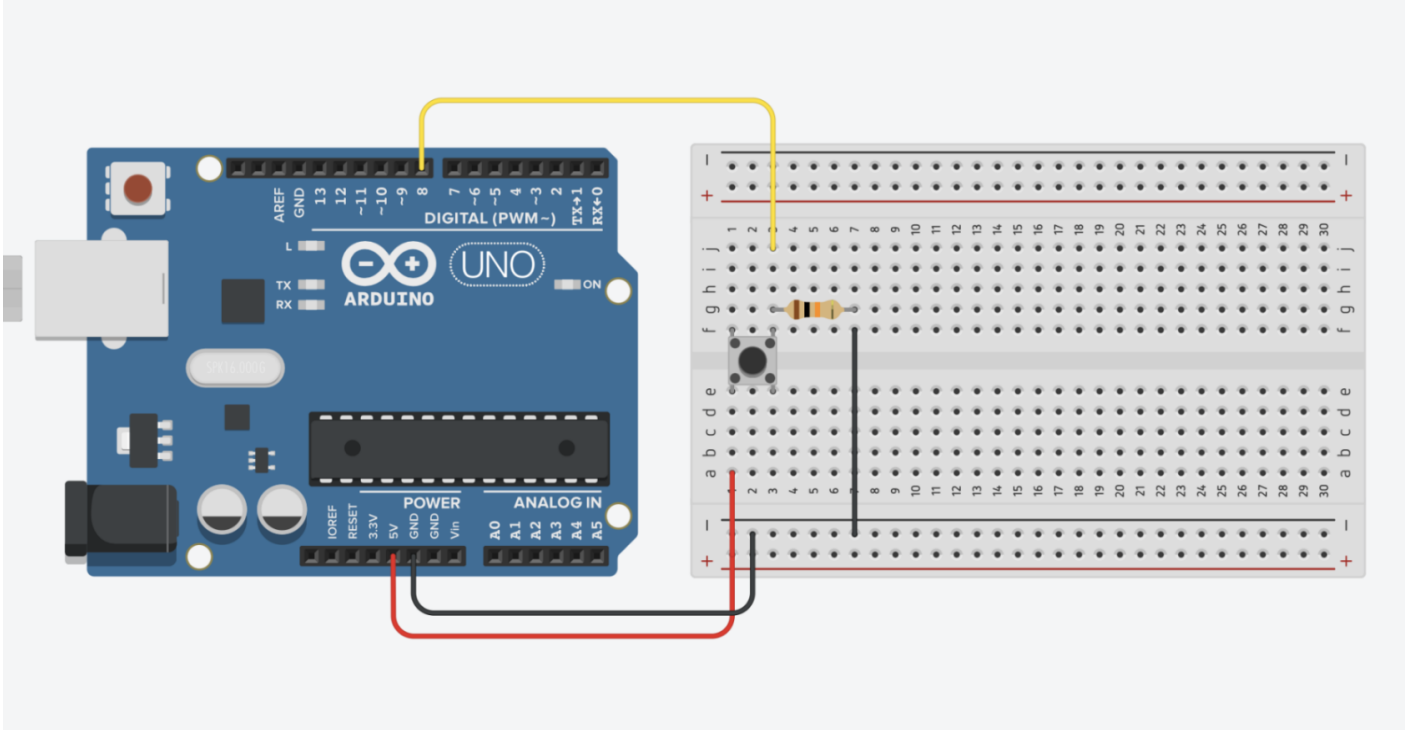
Gdy przycisk nie jest wciśnięty, Arduino może nie wiedzieć, czy pin jest w stanie niskim (LOW), czy wysokim (HIGH), co prowadzi do przypadkowych odczytów. W rezultacie program może błędnie reagować, np. myśląc, że przycisk został wciśnięty, gdy w rzeczywistości tak się nie stało.

Jak działa rezystor pull-down?

Rezystor pull-down zapewnia, że pin jest zawsze podłączony do masy (GND), gdy przycisk nie jest wciśnięty. Dzięki temu na pinie wejściowym panuje stabilny stan niski (LOW). Kiedy naciskasz przycisk, obwód jest zamykany i na pinie pojawia się stan wysoki (HIGH).

Alternatywa – rezystor pull-up:

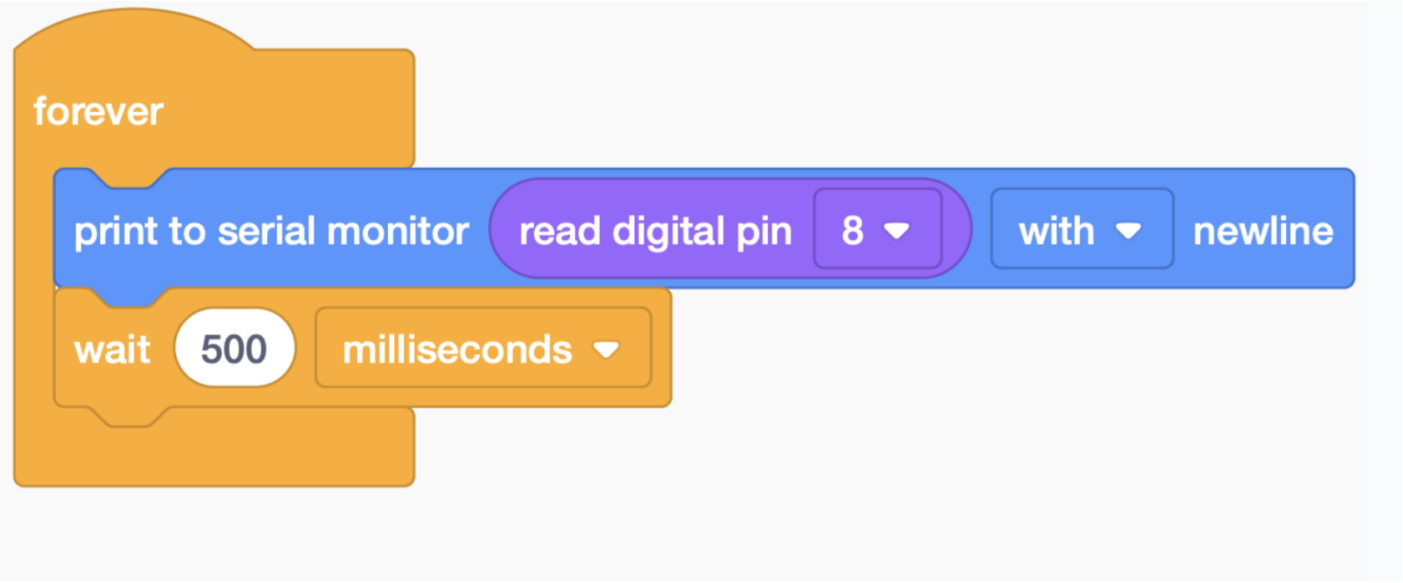
W Arduino często zamiast rezystora typu pull-down używa się wewnętrznego rezystora pull-up, który jest wbudowany w mikrokontroler. W takim przypadku pin domyślnie jest podciągnięty do stanu wysokiego **(HIGH)**, a gdy przycisk jest wciśnięty, pin zostaje połączony z masą, co daje stan niski **(LOW)**. Używanie **pinMode(pin, INPUT_PULLUP)** aktywuje ten wewnętrzny rezystor




```
void setup()
  {
    pinMode(8, INPUT);
    Serial.begin(9600);
  }

  void loop()
  {
    Serial.println(digitalRead(8));
    delay(500); // Wait for 500 millisecond(s)
  }
```

Lub kod z Tinkercada:


forever
print to serial monitor
read digital pin
8
with
newline
wait
500
milliseconds

Biblioteki w Arduino

Czym s? biblioteki?

Biblioteki to zestawy funkcji i gotowych programów, które rozwiązują specyficzne zadania. Zamiast pisać kod do obsługi np. wyświetlacza LCD lub czujnika temperatury od podstaw, możesz po prostu zaimportować bibliotekę, która już zawiera wszystkie potrzebne funkcje i gotowe fragmenty kodu.

```
#include <Twoja_biblioteka.h>
```

Dodawanie biblioteki

Jakie biblioteki mamy dost?pne w Arduino IDE

- Zainstalowane automatycznie: Arduino IDE zawiera wiele bibliotek, które są preinstalowane. Są to popularne biblioteki, takie jak obsługa komunikacji szeregowej, PWM, I2C, czy SPI.
- Z opcją instalacji przez Manager Bibliotek: Aby to zrobić, kliknij na „Sketch” → „Include Library” → „Manage Libraries...” i wyszukaj interesującą Cię bibliotekę.

Alternatywne ?ród?a bibliotek

- Strony takie jak github
- Strony producentów sprzętu elektronicznego
- Fora internetowe / grupy na Mediach Społecznościowych /
- Tworzenie własnych bibliotek

Jak dodać alternatywne biblioteki:

- Kliknij „Sketch” → „Include Library” → „Add .ZIP Library...”.
- Wskaż plik .zip z biblioteką lub folder, w którym się znajduje.

Implementacje biblioteki do kodu

#include <Twoja_biblioteka.h>

Dodając bibliotekę warto sprawdzić jej dokumentacje, żeby móc z niej poprawnie korzystać lub przejrzeć internet z przykładowych wykorzystaniu bibliotek na ŻYWYM kodzie.

Przyk?ady:

Wy?wietlacz ciek?okrystaliczny - LiquidCrystal

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {

  lcd.begin(16, 2);

  lcd.print("Hello, World!");

}
```

Serwomechanizm - Servo

```
#include <Servo.h>

Servo myServo;

void setup() {

  myServo.attach(9);

  myServo.write(90); // Ustaw serwo na 90 stopni

}
```

Czujnik DHT - temperatura - DHT

```
#include <DHT.h>

#define DHTPIN 2
```

```
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {

  dht.begin();

}
```