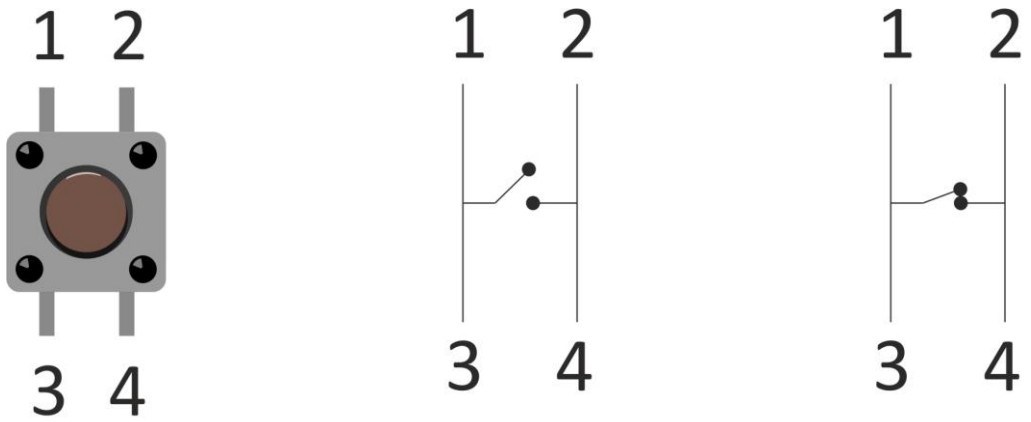
PRZYCISK (MICROSWITCH)

PRZYKŁAD UŻYCIA PRZYCISKU (ARDUINO UNO) + REZYSTORY PODCIĄGAJĄCE

WPROWADZENIE

Przyciski są powszechnie stosowanymi elementami w projektach Arduino i nie tylko. Za jego pośrednictwem można wpłynąć na przebieg procesu, stan urządzenia lub wynik instrukcji warunkowej w zaimplementowanym algorytmie. Przyciski typu microswitch posiadają cztery wyprowadzenia (przedstawione na fotografii 1), które w zależności od naciśnięcia ulegają zwarciu. Domyślnie wyprowadzenia 1 i 3 oraz 2 i 4 są na stałe połączone. Po naciśnięciu przycisku, zwarciu ulegają wszystkie wyprowadzenia (wszystkie są ze sobą połączone).



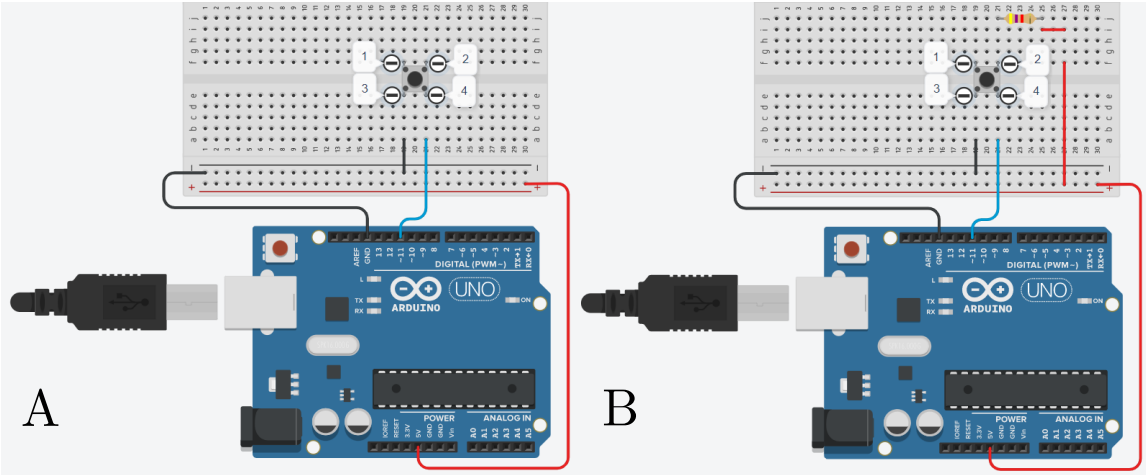
Fotografia 1. Budowa przycisku typu microswitch.

REZYSTORY PODCIĄGAJĄCE / ŚCIĄGAJĄCE

Rezystory podciągające/ściągające wykorzystuje się w celu ustalenia stanu spoczynkowego wejścia układu cyfrowego (brak rezystora powoduje wystąpienie zakłóceń na wejściu układu). Rezystory podciągające to zwykłe oporniki (np. 4.7 kΩ lub 10 kΩ), które podłącza się do dodatniej szyny zasilania (lub w przypadku rezystora ściągającego do masy układu). Zadaniem tego typu układów jest polaryzacja wejścia układu stabilnym napięciem. W zależności od potrzeby i podłączenia układu należy wybrać odpowiedni typ rezystora. Gdy po naciśnięciu przycisku stan odczytany na wejściu ma być wartością:

- logiczną 1 (stan wysoki) przycisk powinien być podłączony do zasilania (5 V) oraz należy wykorzystać rezystor ściągający (ang. pulldown);
- logiczną 0 (stan niski) przycisk powinien być podłączony do masy układu (GND) oraz należy wykorzystać rezystor podciągający (ang. pullup).

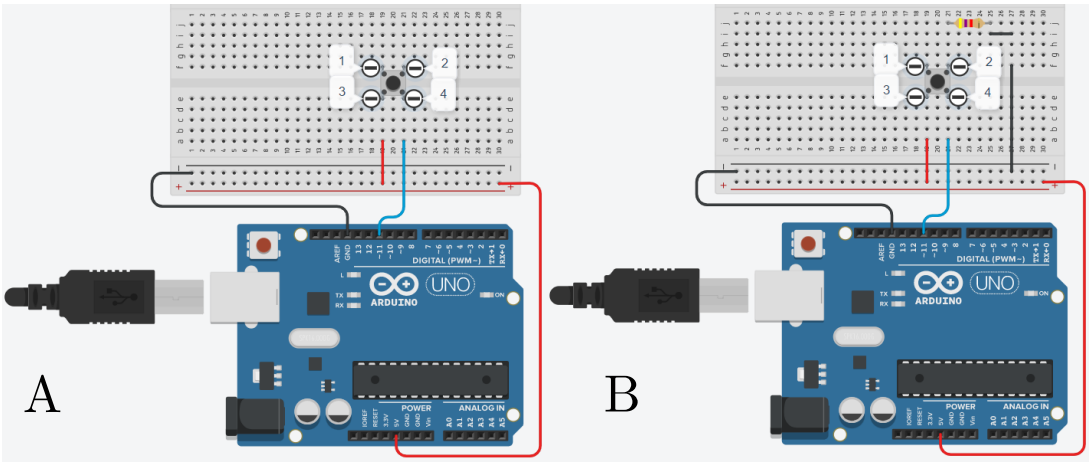
Na fotografii 2 przedstawiono sytuację podłączenia przycisku do Arduino Uno do masy układu bez wykorzystania rezystora podciągającego (A) oraz z wykorzystaniem tego typu rezystora (B).



Fotografia 2. Podłączenie przycisku do Arduino bez i z rezystorem podciągającym.

Sytuacja przedstawiona na fotografii 2 (A) spowoduje wystąpienia zakłóceń na wejściu cyfrowym (PIN 11). Domyślnie (przycisk niewciśnięty) wyprowadzenie 2 i 4 są połączone oznacza to, że dopiero naciśnięcie przycisku spowoduje pełne zwarcie wyprowadzeń (na PINie 11 zostanie wykryty stan logiczny 0 – niski). W celu poprawy powyższego schematu konieczne jest podłączenie rezystora podciągającego do przycisku (wyprowadzenie nr 2) jak pokazano na fotografii 2 (B). Gdy przycisk pozostaje niewciśnięty na wejściu układu jest stan logiczny 1 (wysoki), natomiast po naciśnięciu przycisku na wejściu odczytana zostanie wartość logiczna 0 (stan niski).

Na fotografii 3 przedstawiono sytuację podłączenia przycisku do Arduino Uno do zasilania bez wykorzystania rezystora ściągającego (A) oraz z wykorzystaniem tego typu rezystora (B).

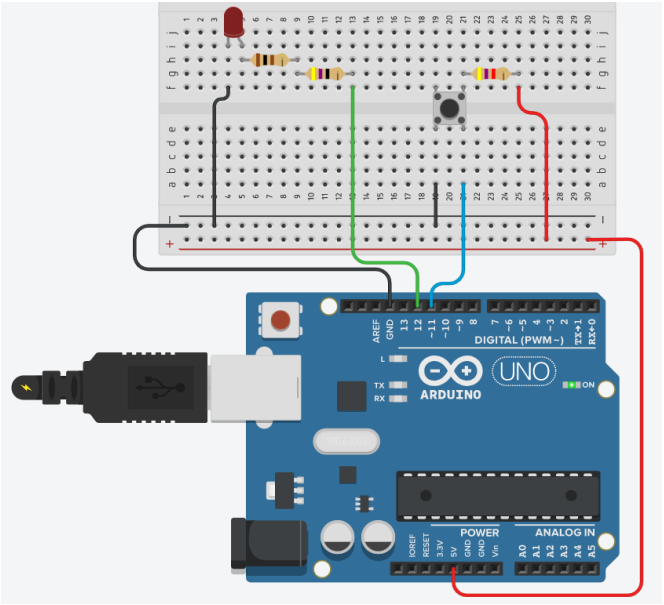


Fotografia 3. Podłączenie przycisku do Arduino bez i z rezystorem ściągającym.

Sytuacja przedstawiona na fotografii 3 (A) spowoduje wystąpienia zakłóceń na wejściu cyfrowym (PIN 11). Domyślnie (przycisk niewciśnięty) wyprowadzenie 2 i 4 są połączone oznacza to, że dopiero naciśnięcie przycisku spowoduje pełne zwarcie wyprowadzeń (na PINie 11 zostanie wykryty stan logiczny 1 – wysoki). W celu poprawy powyższego schematu konieczne jest podłączenie rezystora ściągającego do przycisku (wyprowadzenie nr 2) jak pokazano na fotografii 3 (B). Gdy przycisk pozostaje niewciśnięty na wejściu układu jest stan logiczny 0 (niski), natomiast po naciśnięciu przycisku na wejściu odczytana zostanie wartość logiczna 1 (stan wysoki).

ARDUINO UNO – PRZYCISK Z REZYSOTREM PODCIĄGAJĄCYM

Na fotografii 4 przedstawiono prosty układ z przyciskiem (z rezystorem podciągającym) i diodą LED.



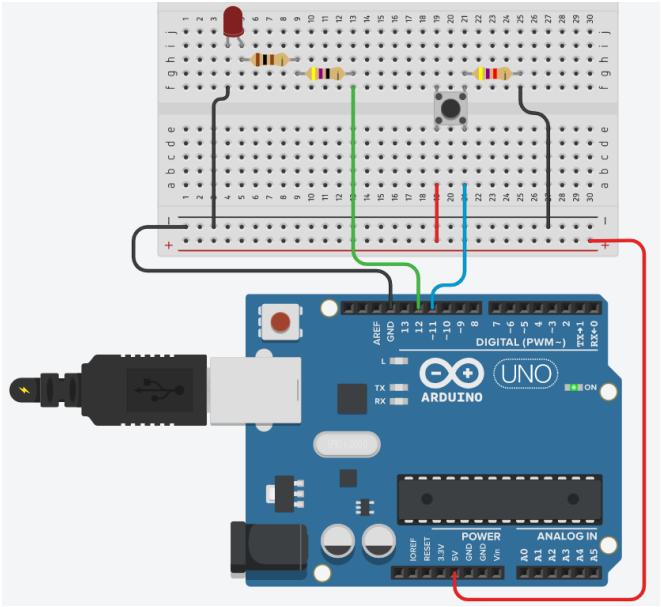
Fotografia 4. Analizowany układ diody LED i przycisku z rezystorem podciągającym.

```
void setup()
{
  pinMode(12, OUTPUT);     // ustaw PIN 12 jako wyjście
  pinMode(11, INPUT);      // ustaw PIN 11 jako wejście
}
void loop()
{
  if (digitalRead(11) == LOW) {
   digitalWrite(12, HIGH);  // ustaw na PINie 12 stan HIGH
  }
  else{
    digitalWrite(12, LOW);  // ustaw na PINie 12 stan LOW
  }
}
```

Dioda zaświeci się w sytuacji naciśnięcia przycisku (w tym przypadku gdy na wejściu układu będzie wartość logicznego 0 – niskiego). Minusem tego rozwiązania jest fakt, że aby dioda świeciła trzeba cały czas trzymać przycisk naciśnięty.

ARDUINO UNO – PRZYCISK Z REZYSOTREM ŚCIĄGAJĄCYM

Na fotografii 5 przedstawiono prosty układ z przyciskiem (z rezystorem ściągającym) i diodą LED.



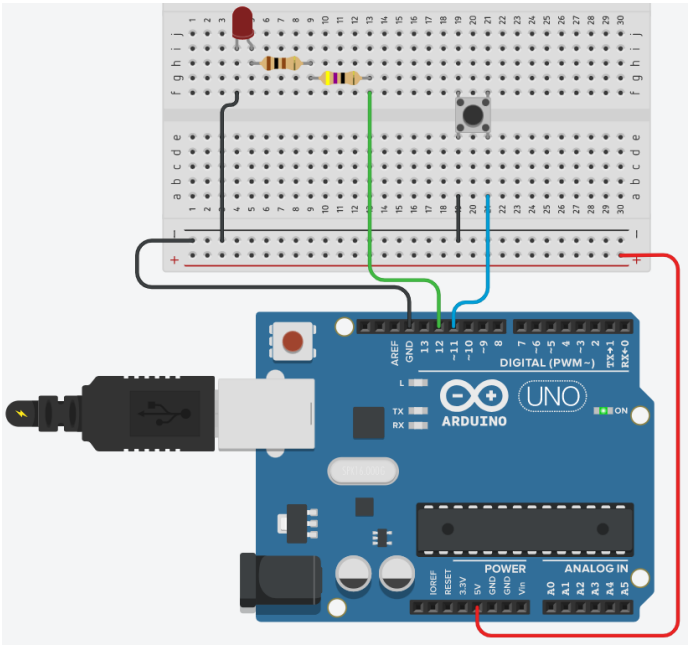
Fotografia 5. Analizowany układ diody LED i przycisku z rezystorem ściągającym.

```
void setup()
{
  pinMode(12, OUTPUT);     // ustaw PIN 12 jako wyjście
  pinMode(11, INPUT);      // ustaw PIN 11 jako wejście
}

void loop()
{
  if (digitalRead(11) == HIGH) {
   digitalWrite(12, HIGH);  // ustaw na PINie 12 stan HIGH
  }
  else{
    digitalWrite(12, LOW);  // ustaw na PINie 12 stan LOW
  }
}
```

Dioda zaświeci się w sytuacji naciśnięcia przycisku (w tym przypadku gdy na wejściu układu będzie wartość logiczna 1 – stan wysoki). Minusem tego rozwiązania jest fakt, że aby dioda świeciła trzeba cały czas trzymać przycisk naciśnięty.

ARDUINO UNO – PRZYCISK Z WBUDOWANYM REZYSOTREM PODCIĄGAJĄCYM

Na fotografii 6 przedstawiono prosty układ z przyciskiem (wykorzystany wbudowany rezystor podciągający pullup) i diodą LED.



Fotografia 6. Analizowany układ diody LED i przycisku.

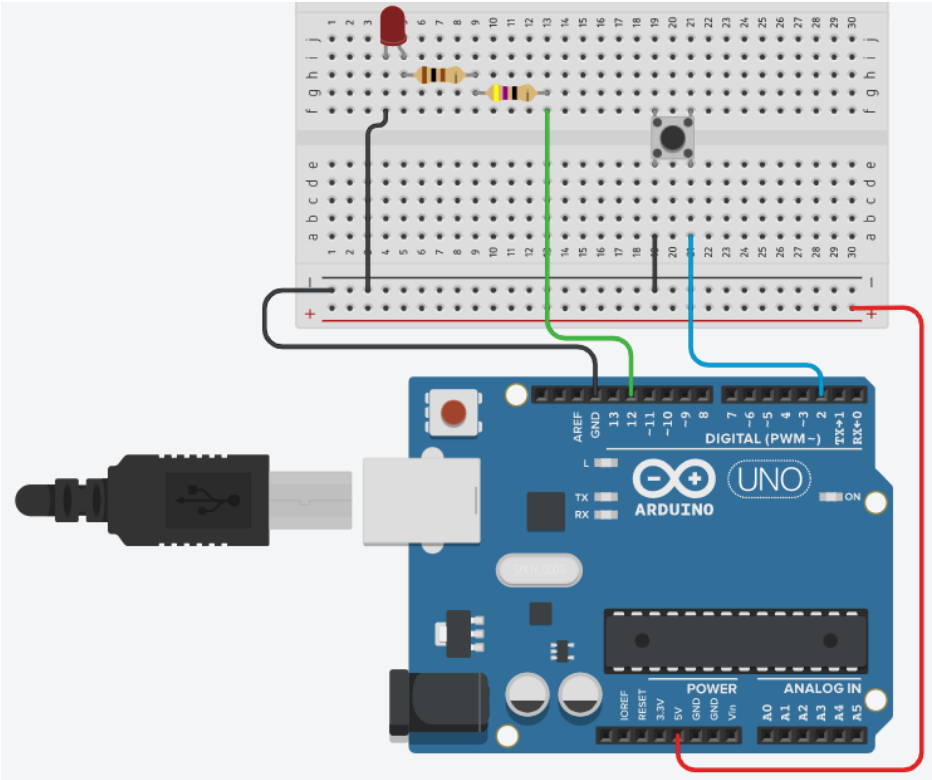
```
void setup()
{
  pinMode(12, OUTPUT);           // ustaw PIN 12 jako wyjście
  pinMode(11, INPUT_PULLUP);     // ustaw PIN 11 jako wejście (wykorzystaj
                                 // wybudowany rezystor pullup)
}

void loop()
{
  if (digitalRead(11) == HIGH) {
   digitalWrite(12, HIGH);       // ustaw na PINie 12 stan HIGH
  }
  else{
    digitalWrite(12, LOW);       // ustaw na PINie 12 stan LOW
  }
}
```

Dioda zaświeci się w sytuacji naciśnięcia przycisku (w tym przypadku gdy na wejściu układu będzie wartość logicznego 0 – niskiego). Minusem tego rozwiązania jest fakt, że aby dioda świeciła trzeba cały czas trzymać przycisk naciśnięty. Zaletą tego rozwiązania jest natomiast wykorzystania wbudowanych (w Arduino Uno) rezystorów podciągających (brak konieczności podłączenia

fizycznego rezystora do układu – konieczna konfiguracja w programie). W Arduino Uno nie ma wbudowanych rezystorów ściągających.

ARDUINO UNO – PRZYCISK + PRZERWANIE

Minusem wcześniej prezentowanych przykładów jest fakt, że naciśnięcie przycisku powinno nastąpić w chwili wywoływania metody *digitalRead()*. Dopiero w tej sytuacji stan diody może ulec zmianie. W przypadku długich i rozbudowanych algorytmów sterowania istnieje możliwość wystąpienia sytuacji, w której naciśnięcie przycisku nic nie zmieni (ponieważ algorytm będzie analizował inną funkcję). Arduino Uno wspiera przerwania wyłącznie na PINach 2 oraz 3. Na fotografii 7 przedstawiono schemat analizowanego układu.



Fotografia 7. Analizowany układ diody LED i przycisku.

```
String state = "off";       // zmienna przechowująca stan diody
                            // (on = zaświecona, off = zgaszona)
void setup()
{
  pinMode(12, OUTPUT);        // ustaw PIN 12 jako wyjście
  pinMode(2, INPUT_PULLUP);   // ustaw PIN 2 jako wejście
                              // i wykorzystaj wbudowany rezystor podciągający
  attachInterrupt(digitalPinToInterrupt(2), change_state, FALLING);
          // wykorzystaj przerwanie na PINie 2 (digitalPinToInterrupt(2))
          // w sytuacji wywołania przerwania -> wywołaj funkcje change_state()
          // wywołaj przerwanie gdy zmienia się stan HIGH na LOW (FALLING)
}
void loop()
{
}

void change_state()
{
  if (state == "off"){
    digitalWrite(12, LOW);
    state = "on";
  }
  else{
    digitalWrite(12, HIGH);
    state = "off";
  }
}
```

Funckja *attachInterrupt()* umożliwia następujące ustawienia wywoływania przerwań:

- LOW – kiedy na pinie jest stan niski;
- CHANGE – kiedy zmienia się stan;
- RISING – kiedy zmienia się stan z LOW na HIGH;
- FALLING – kiedy zmienia się stan z HIGH na LOW.

ŹRÓDŁA

[1] https://forbot.pl/blog/leksykon/rezystor-podciagajacy-rezystor-pull-up

[2] https://forbot.pl/blog/leksykon/rezystor-sciagajacy-rezystor-pull-down

[3] http://akademia.nettigo.pl/arduino_przerwania/