**Spis treści**

1. Zadania 2

1.1. Idea warsztatu 2

1.2. Mruganie diodą 2

1.3. Zapis do portu szeregowego 2

1.4. Odczyt z portu szeregowego 3

1.5. PWM 3

1.6. 1Wire 4

1.7. Rejestr przesuwny 4

2. Schematy ideowe 5

2.1. Podłączenie diody do wyjścia cyfrowego 5

2.2. Sterowanie silnikiem przy użyciu potencjometru 5

2.3. Podłączenie jednego czujnika temperatury 6

2.4. Podłączenie jednego czujnika temperatury w trybie pasożytniczym 6

2.5. Podłączenie wielu czujników temperatury 7

2.6. Rejestr przesuwny 7

3. Code 8

3.1. Mruganie diodą 8

3.2. Zapis do SerialPort 8

3.3. Odczyt z SerialPort 8

3.4. Sterowanie PWM 9

3.5. Sterowanie PWM przy użyciu SerialPortu 9

3.6. Sterowanie PWM przy użyciu potencjometru 10

3.7. Odczyt urządzeń z linii 1Wire 11

3.8. Odczyt temperatury 12

3.9. Rejestr przesuwny 13

# Zadania

## Idea warsztatu

* 1. **Cel warsztatu:**Celem dzisiejszych ćwiczeń jest konstrukcja uproszczonego systemu automatyki budynkowej. W tym celu wykonane zostaną zadania przedstawiające podstawową funkcjonalność systemu Arduino zawierające:
     1. Obsługę wejść i wyjść cyfrowych
     2. Obsługę wejść i wyjść analogowych
     3. Obsługę portu szeregowego
     4. Podstawy sterowania silnikami DC
     5. Podstawy protokołu komunikacyjnego 1Wire
     6. Obsługę przy użyciu (odpowiednich bibliotek) termometru cyfrowego
     7. Obsługę wyświetlacza segmentowego przy użyciu rejestru przesuwnego.
  2. **Ostatecznie** po wykonaniu początkowych ćwiczeń uczestnicy skonstruują podstawowy system automatyki budynkowej na podstawie wiedzy zdobytej w czasie wykładu.

## Mruganie diodą

1. **Cel zadania:** Zadanie polega na poznaniu podstawowych elementów programu pisanego dla mikrokontrolerów AtMega16 w platformie Arduino oraz konstrukcji najprostszego układu elektrycznego złożonego z diody i rezystora.
2. **Niezbędne kroki:**
   * 1. Konstrukcja układu elektrycznego: ***Schemat 2.1***
     2. Definicja portu, do którego podłączona jest dioda
     3. Ustawienie wybranego portu jako wyjście cyfrowe - metoda *setup()*
     4. Zapis wartości *HIGH / LOW* do wyjścia cyfrowego
3. **Przykładowe rozwiązanie:**
4. ***Code 3.1***

## Zapis do portu szeregowego

1. **Cel zadania:** Zadanie polega na poznaniu portu szeregowego - możliwość wyświetlenia informacji. W zadaniu należy wyświetlić zmienną zawierającą informację o liczbie wykonań pętli *loop().*
2. **Niezbędne kroki:**
3. Inicjalizacja portu szeregowego na częstotliwości 9600.
4. Definicja zmiennej określającej liczbę iteracji.
5. Wyświetlenie w pętli liczby dotychczasowych iteracji.
6. **Przykładowe rozwiązanie: *Code 3.2***

## Odczyt z portu szeregowego

1. **Cel zadania:** Zadanie polega na odczytaniu informacji z portu szeregowego. W zadaniu zakładamy, że odczytywane będą wartości w formacie procentowym (1 - 99), które posłużą nam w kolejnych zadaniach.
2. **Niezbędne kroki:**
3. Inicjalizacja portu szeregowego
4. Odczyt dwóch znaków z portu szeregowego
5. Napisanie metody sprawdzającej format danych
6. Wyświetlenie poprawnie odczytanych wartości
7. **Przykładowe rozwiązanie: *Code 3.3***

## PWM

1. **Cel zadania:** Zadanie polega na zapoznaniu się z wyjściami analogowymi Arduino. Zostaną przedstawione trzy warianty ustawienia wyjścia analogowego PWM - poprzez ustawienie wartości stałego wypełnienia, poprzez ustawienie wartości wypełnienia opartej na odczycie potencjometru oraz poprzez ustawienie wartości wypełnienia na podstawie wartości wczytanej poprzez port szeregowy.
2. **Niezbędne kroki (wersja z potencjometrem):**
3. Konstrukcja poniższego układu elektrycznego: ***Schemat 2.2***
4. Inicjalizacja wejścia i wyjścia analogowego
5. Inicjalizacja portu szeregowego w celu wyświetlania wartości
6. Odczyt wartości z potencjometru (1 - 1024)
7. Napisanie metody konwertującej odczyt z potencjometru do formatu rozumianego przez PWM (0 - 255)
8. Wyświetlenie poprawnie odczytanych wartości
9. **Niezbędne kroki (wersja z odczytem z portu szeregowego):**
10. Konstrukcja poniższego układu elektrycznego: ***Schemat 2.2***
11. Inicjalizacja wejścia i wyjścia analogowego
12. Inicjalizacja portu szeregowego w celu wyświetlania wartości
13. Odczyt wartości z portu szeregowego
14. Napisanie metody konwertującej procentowy poziom wypełnienia do formatu rozumianego przez PWM (0 - 255)
15. Wyświetlenie poprawnie odczytanych wartości
16. Ustawienie wartości wypełnienia
17. **Przykładowe rozwiązanie: *Code 3.4, 3.5, 3.6***.

## 1Wire

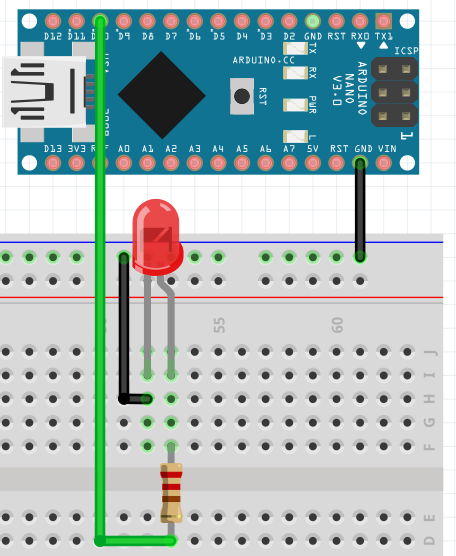
* 1. **Cel zadania:** Zadanie polega na zapoznaniu się z protokołem komunikacyjnym 1Wire, czujnikiem temperatury *DS18B20*. Zadanie składa się z dwóch części: pierwsza identyfikuje adres podłączonego urządzenia, druga dokonuje pomiaru temperatury.
  2. **Niezbędne kroki (identyfikacja urządzenia):**
     1. Konstrukcja poniższego układu elektrycznego: ***Schemat*** ***2.3***
     2. Import odpowiednich bibliotek: 1Wire.h oraz DS18B20.h
     3. Inicjalizacja portu szeregowego w celu wyświetlania wartości
     4. Utworzenie i inicjalizacja odpowiednich zmiennych reprezentujących czujnik oraz protokół komunikacyjny 1Wire
     5. Odczyt i wyświetlenie poprawnego adresu
  3. **Niezbędne kroki (wyświetlenie temperatury):**
     1. Konstrukcja poniższego układu elektrycznego: ***Schemat 2.3***
     2. Import odpowiednich bibliotek: 1Wire.h oraz DS18B20.h
     3. Inicjalizacja portu szeregowego w celu wyświetlania wartości
     4. Definicja tablicy odpowiadającej za odczytany poprzednio adres czujnika
     5. Utworzenie i inicjalizacja odpowiednich zmiennych reprezentujących czujnik oraz protokół komunikacyjny 1Wire
     6. Wyświetlenie odczytanej temperatury na porcie szeregowym.
  4. **Przykładowe rozwiązanie: *Code 3.7, 3.8***.

## Rejestr przesuwny

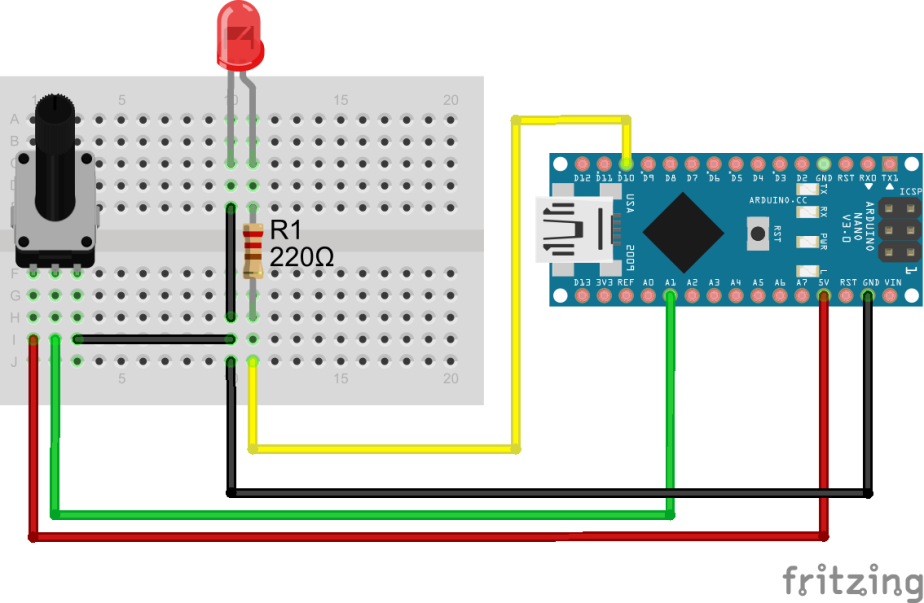
1. **Cel zadania: Uzupełnić**
2. **Niezbędne kroki**
   * 1. UZUPEŁNIĆ

# Schematy ideowe

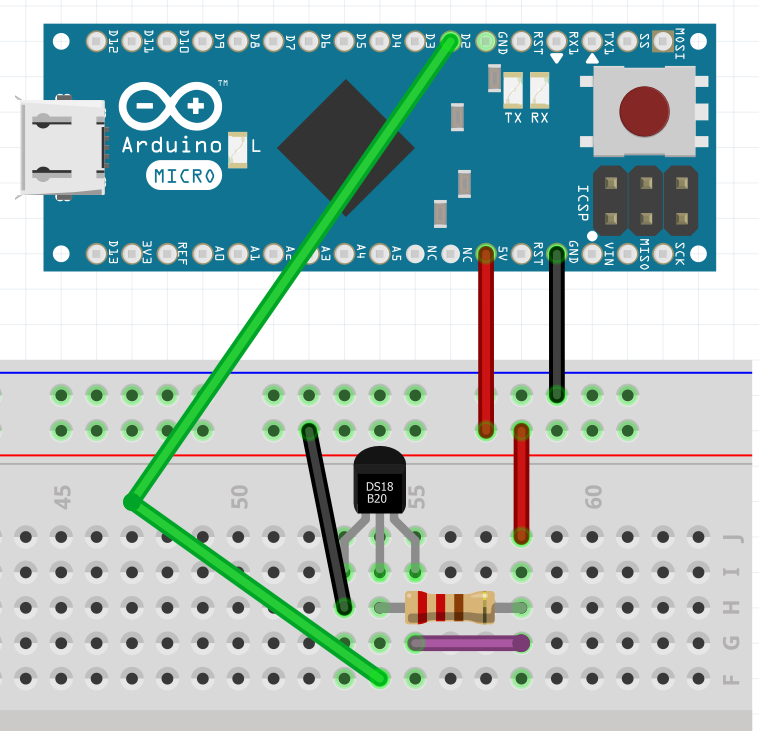
## Podłączenie diody do wyjścia cyfrowego



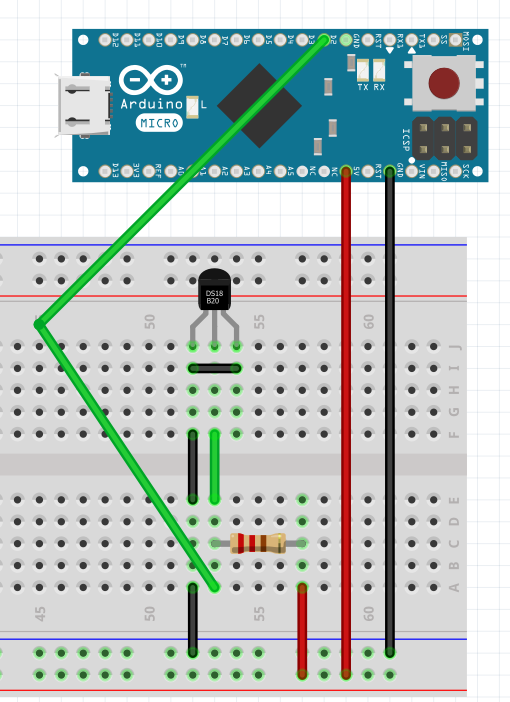
## Sterowanie silnikiem przy użyciu potencjometru



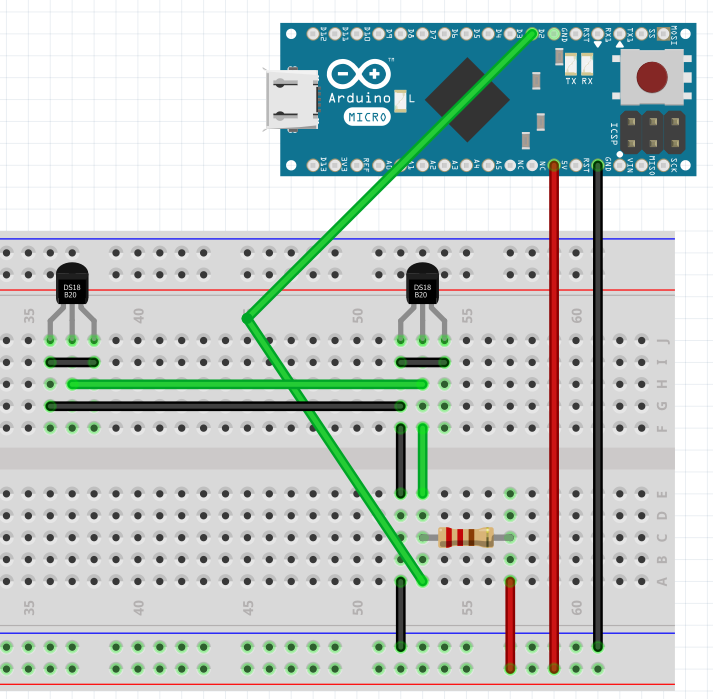
## Podłączenie jednego czujnika temperatury



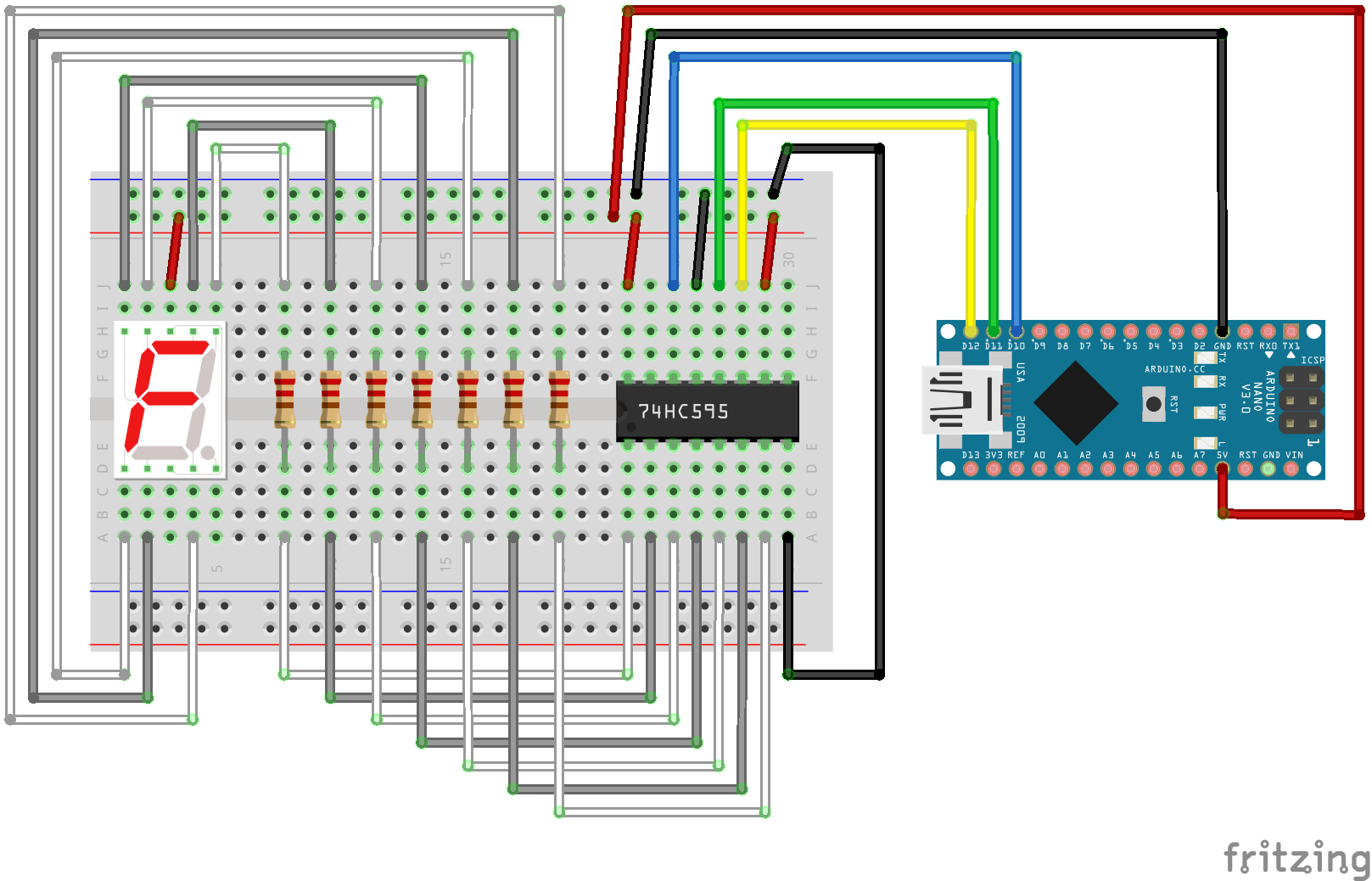
## Podłączenie jednego czujnika temperatury w trybie pasożytniczym



## Podłączenie wielu czujników temperatury



## Rejestr przesuwny



# Code

## Mruganie diodą

|  |
| --- |
| **int** portNumber = 10;  **void** setup() {  pinMode(portNumber, OUTPUT);  }  **void** loop() {  digitalWrite(portNumber, HIGH);  delay(1000);  digitalWrite(portNumber, LOW);  delay(1000);  } |

## Zapis do SerialPort

|  |
| --- |
| **int** iterator = 0;  **void** setup(){  Serial.begin(9600);  }  **void** loop(){  iterator = iterator + 1;  Serial.print("Iteration : "); //send the string “hello” and return the length of the string.  Serial.println(iterator);  delay(1000);  } |

## Odczyt z SerialPort

|  |
| --- |
| // whole content  String incomingContent = "";  // single character  **char** character;  **void** setup() {  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps  Serial.println("Please insert value");  }  **void** loop() {    **while** (Serial.available()) {  character = Serial.read();  incomingContent.concat(character);  }  **if** (incomingContent != "") {  Serial.println(incomingContent);  incomingContent = "";  }  } |

## Sterowanie PWM

|  |
| --- |
| **int** pwmPort = 6;  **int** pwmValueLow = 100;  **int** pwmValueHigh = 255;  **void** setup() {  // SetUp proper pinmode  pinMode(pwmPort, OUTPUT);  }  **void** loop() {  // write proper value to PWM output  analogWrite(pwmPort, pwmValueLow);  //Delay  delay(3000);  // write proper value to PWM output  analogWrite(pwmPort, pwmValueHigh);  //Delay  delay(3000);  } |

## Sterowanie PWM przy użyciu SerialPortu

|  |
| --- |
| // PWM properties  **int** pwmPort = 6;  **int** pwmValueLow = 100;  **int** pwmValueHigh = 255;  // whole content  String incomingContent = "";  // single character  **char** character;  **void** setup() {  // SetUp proper pinmode  pinMode(pwmPort, OUTPUT);  // opens serial port, sets data rate to 9600 bps  Serial.begin(9600);  Serial.println("Please insert value");  }  **void** loop() {  **while** (Serial.available()) {  character = Serial.read();  incomingContent.concat(character);  }  **if** (incomingContent != "" && incomingContent.length() == 2) {  **if** (validateContent(incomingContent)) {  Serial.println("Recieved valid percentage value");  analogWrite(pwmPort, convertPercentageToPWM(incomingContent));  } **else** {  Serial.println("Something went wrong");  analogWrite(pwmPort, pwmValueLow);  }  incomingContent = "";  }  }  **boolean** validateContent(String incomingContent) {  Serial.print("Recieved content:");  Serial.println(incomingContent);  // always = 2  **int** stringLength = incomingContent.length();  **boolean** result = **true**;  **for** (**int** i = 0; i < stringLength; i++) {  **char** singleChar = incomingContent.charAt(i);  result &= isDigit(singleChar);  **if** (!result) {  Serial.println("Recieved non digital character");  **return** result;  }  }  **return** result;  }  **int** convertPercentageToPWM(String incomingContent) {  **float** percentage = incomingContent.toFloat();  Serial.print("Percentage: ");  Serial.println(percentage);  **float** result = (percentage / 100) \* pwmValueHigh;  Serial.print("Writing to PWM: ");  Serial.println((**int**) result);  **return** (**int**) result;  } |

## Sterowanie PWM przy użyciu potencjometru

|  |
| --- |
| Sint inputPin = A0; // set input pin for the potentiometer  **int** inputValue = 0; // potentiometer input variable  **int** ledPin = 6; // set output pin for the LED  **void** setup() {  // declare the ledPin as an OUTPUT:  pinMode(ledPin, OUTPUT);  // opens serial port, sets data rate to 9600 bps  Serial.begin(9600);  }  **void** loop() {  // read the value from the potentiometer:  inputValue = analogRead(inputPin);  Serial.print("Potentiometer value: ");  Serial.println(inputValue);  // send the square wave signal to the LED:  analogWrite(ledPin, inputValue / 4);  delay(500);  } |

## Odczyt urządzeń z linii 1Wire

|  |
| --- |
| #include <1Wire.h>  // sensor Port  **const** **byte** 1WIRE\_PIN = 2;  1Wire 1Wire( 1WIRE\_PIN);  **void** setup() {  **while** (!Serial)  ;  Serial.begin(9600);  }  **void** loop() {  **byte** address[8];  1Wire.reset\_search();  **while** (1Wire.search(address)) {  **if** (address[0] != 0x28)  **continue**;  // something went wrong  **if** (1Wire::crc8(address, 7) != address[7]) {  Serial.println("Wrong address");  **break**;  }  // print address starting with Ox  **for** (**byte** i = 0; i < 8; i++) {  Serial.print(F("0x"));  Serial.print(address[i], HEX);  **if** (i < 7)  Serial.print(", ");  }  Serial.println();  }  **while** (1) {  // "ugly" loop which avoids reading sensors all over again  }  } |

## Odczyt temperatury

|  |
| --- |
| #include <1Wire.h>  // sensor Port  **const** **byte** 1WIRE\_PIN = 2;  1Wire 1Wire( 1WIRE\_PIN);  **void** setup() {  **while** (!Serial)  ;  Serial.begin(9600);  }  **void** loop() {  **byte** address[8];  1Wire.reset\_search();  **while** (1Wire.search(address)) {  **if** (address[0] != 0x28)  **continue**;  // something went wrong  **if** (1Wire::crc8(address, 7) != address[7]) {  Serial.println("Wrong address");  **break**;  }  // print address starting with Ox  **for** (**byte** i = 0; i < 8; i++) {  Serial.print(F("0x"));  Serial.print(address[i], HEX);  **if** (i < 7)  Serial.print(", ");  }  Serial.println();  }  **while** (1) {  // "ugly" loop which avoids reading sensors all over again  } |

## Rejestr przesuwny

|  |
| --- |
| **#include** <Arduino.h>  **int** dataPin = 10;  **int** latchPin = 11;  **int** clockPin = 12;  byte dec\_digits[] = {0b00010000,0b01111101};  **void** **setup**() {  pinMode(dataPin, OUTPUT);  pinMode(latchPin, OUTPUT);  pinMode(clockPin, OUTPUT);  }  **void** **loop**() {  **for** (**int** i = 0; i < 2; i++) {  digitalWrite(latchPin, LOW);  shiftOut(dataPin, clockPin, MSBFIRST, dec\_digits[i]);  digitalWrite(latchPin, HIGH);  delay(500);  }  } |