POLITECHNIKA POZNAŃSKA

Wydział Informatyki i Telekomunikacji

Laboratorium 1.
Pierwsze programowanie.

Poznań 2020

Prawa autorskie

Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z mikrokontrolerem Arduino oraz przedstawienie jego podstawowych funkcji i przykładowych zastosowań praktycznych.

Przebieg ćwiczenia

W naszym ćwiczeniu wykorzystamy Arduino UNO, ponieważ jest to najczęściej wykorzystywany przez wszystkich układ. Dzięki temu modułowi każdy może tworzyć i rozwijać ciekawe, programowalne urządzenia elektroniczne. Wykorzystamy również diody, przyciski oraz czujnik temperatury.

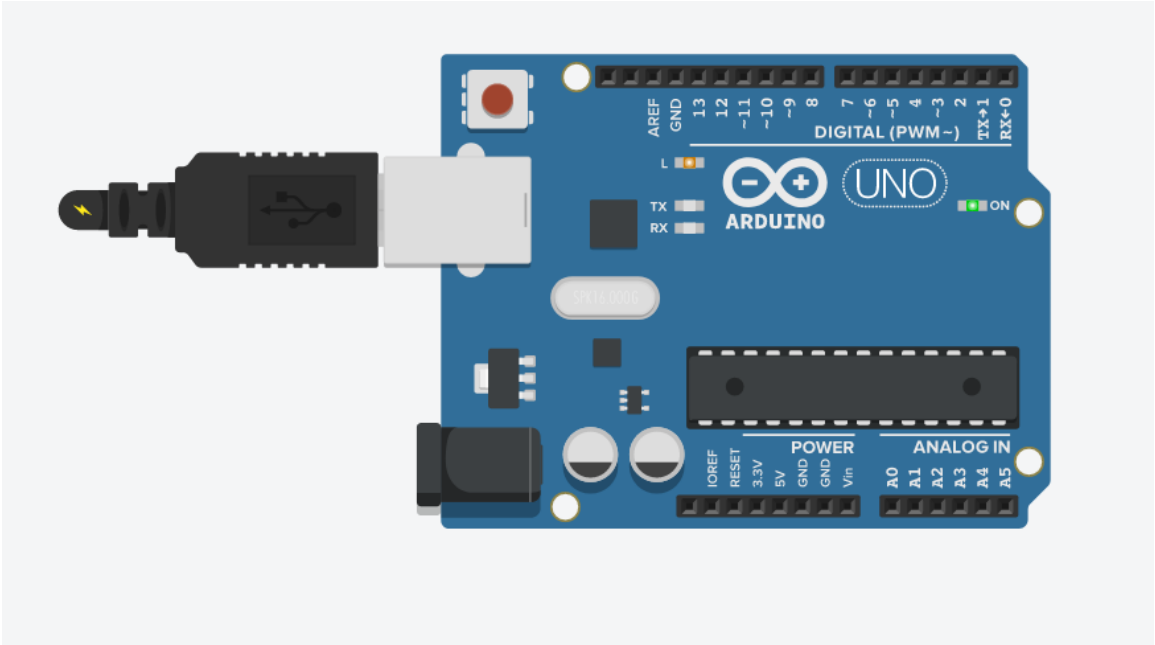
Pierwszy program

Zaczniemy od przetestowania czy nasze Arduino jest sprawne. Mając fizyczne urządzenie, możemy pobrać dedykowane IDE, a następnie po podłączeniu go do komputera, możemy w IDE Arduino wybrać przykładowy program (Plik -> Przykłady -> 01 Basic -> Blink).

Aczkolwiek zadania laboratoryjne zostały przygotowane do wykonania w sposób symultaniczny z wykorzystaniem symulatora internetowego.

W symulatorze wystarczy wkleić poniższy kod:

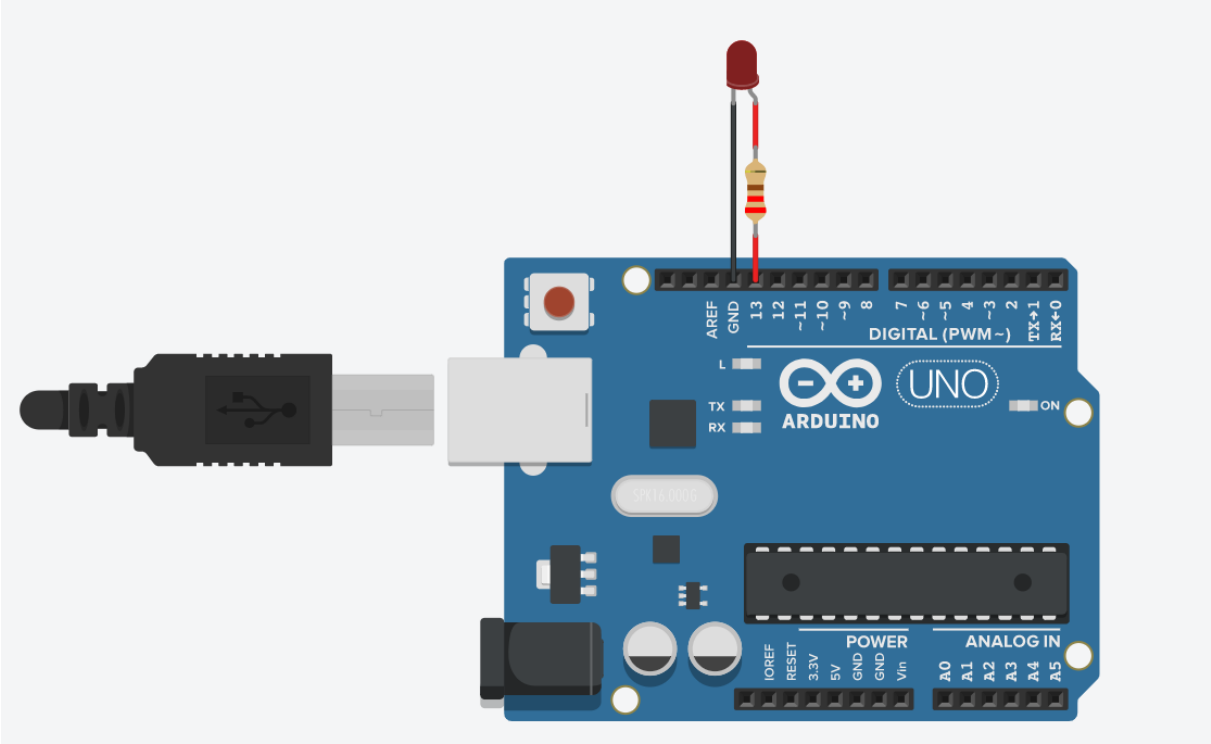
```
void setup()
{
      pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
      digitalWrite(LED_BUILTIN, HIGH);
      delay(1000);
      digitalWrite(LED_BUILTIN, LOW);
      delay(1000);
}
```



Rys.1. Program Blink.

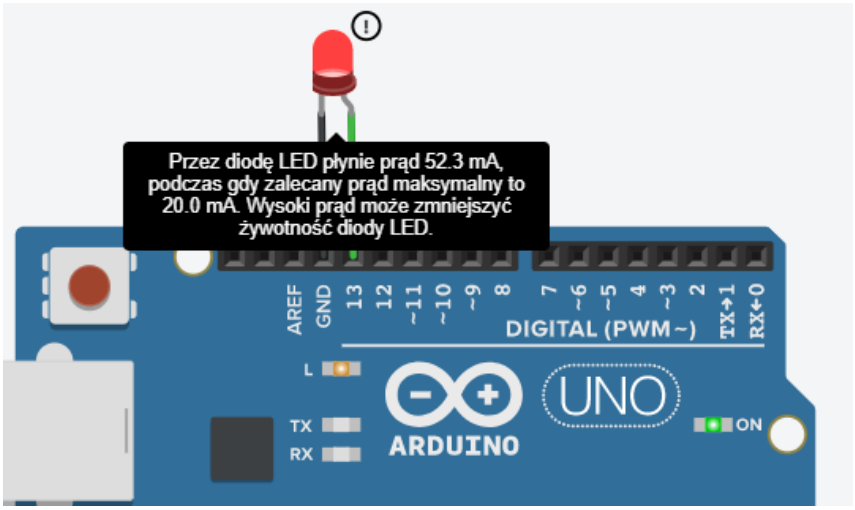
Na rysunku 1 widać zapaloną diodę na płytce (pod pinem GND). Dioda włącza się i wyłącza co 1 sek. Dzięki temu wiemy, że nasze Arduino działa poprawnie.

Miganie diodą



Rys. 2. Dioda.

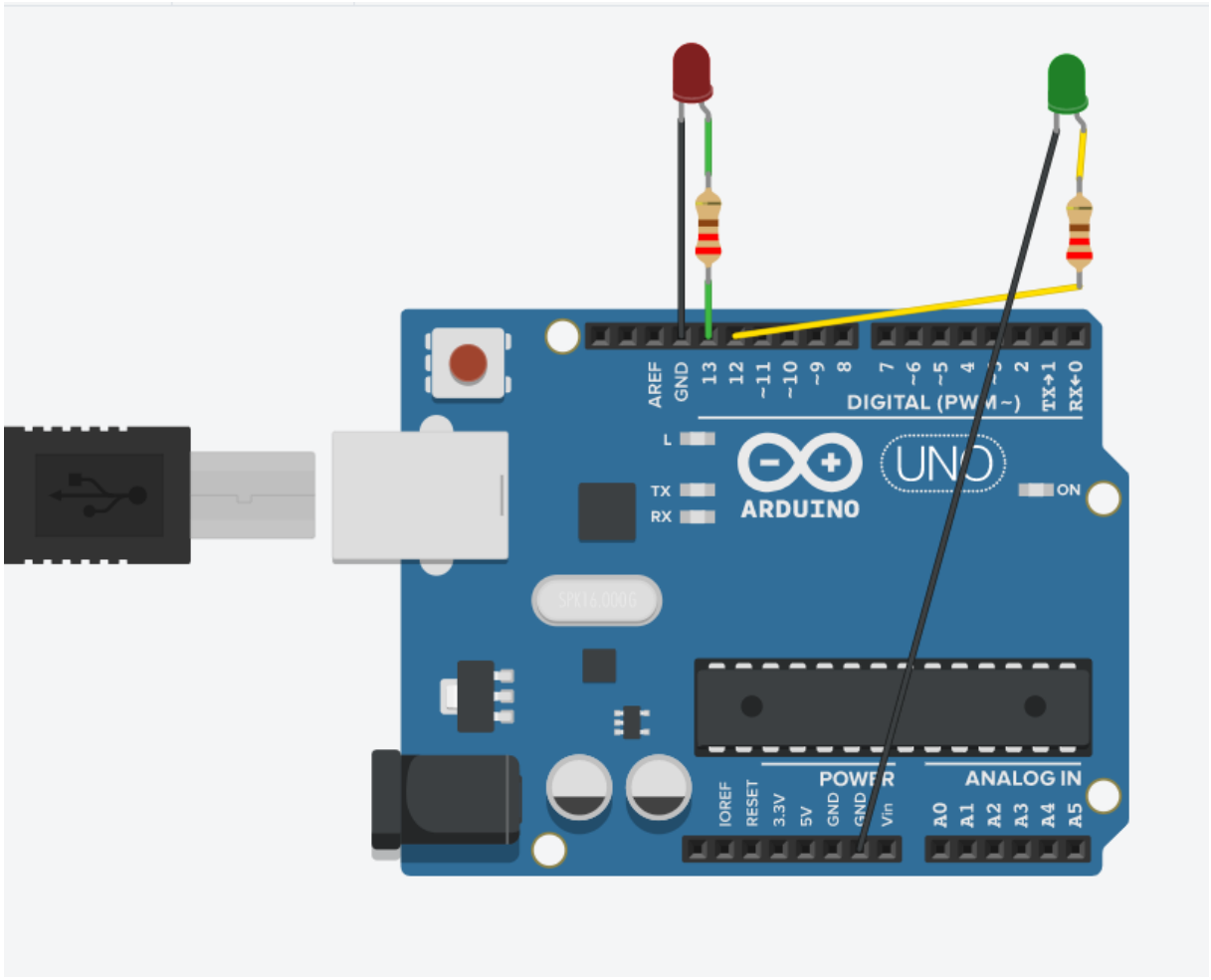
W programie wykorzystano jedną diodę oraz rezystor 220Ω (Rys.2). Dioda co sekundę włącza się oraz wyłącza. Może się zdarzyć sytuacja, że zapomnimy o użyciu rezystora (tu albo w przyszłych programach). Tutaj przewagę ma symulator nad fizycznym sprzętem. Robiąc na fizycznym sprzęcie zapewne taka dioda by się spaliła. W symulatorze natomiast dostajemy ostrzeżenie (Rys. 3.). Dzięki temu wiemy, że zapomnieliśmy wykorzystać rezystor.



Rys. 3. Ostrzeżenie w przypadku braku rezystora.

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);

  delay(1000);

  digitalWrite(13, LOW);

  delay(1000);
}
```

Miganie dwoma diodami



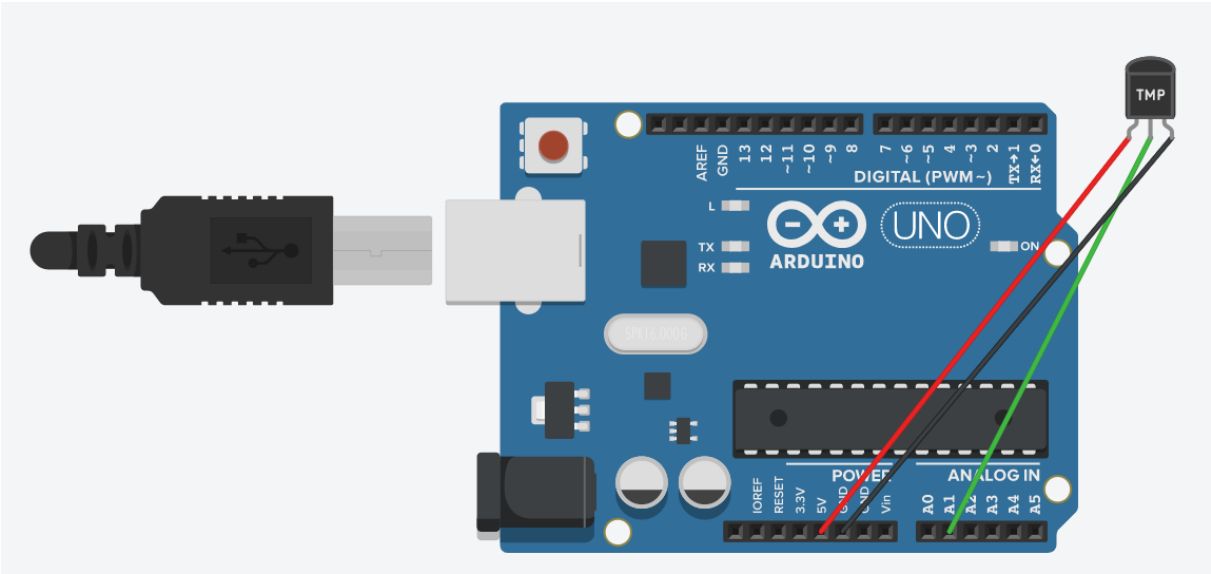
Rys. 4. Dwie diody.

Ten program jest lekko rozbudowaną wersją poprzedniego programu. Zastosowano w nim dwie diody różnego koloru (Rys.4). Diody będą się zapalać i wyłączać na zmianę.

```
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  digitalWrite(12, HIGH);
}

void loop()
{
  digitalWrite(13, HIGH);
  digitalWrite(12, LOW);
  delay(1000);
```

```
  digitalWrite(13, LOW);
  digitalWrite(12, HIGH);
  delay(1000);
}
```

Pomiar temperatury



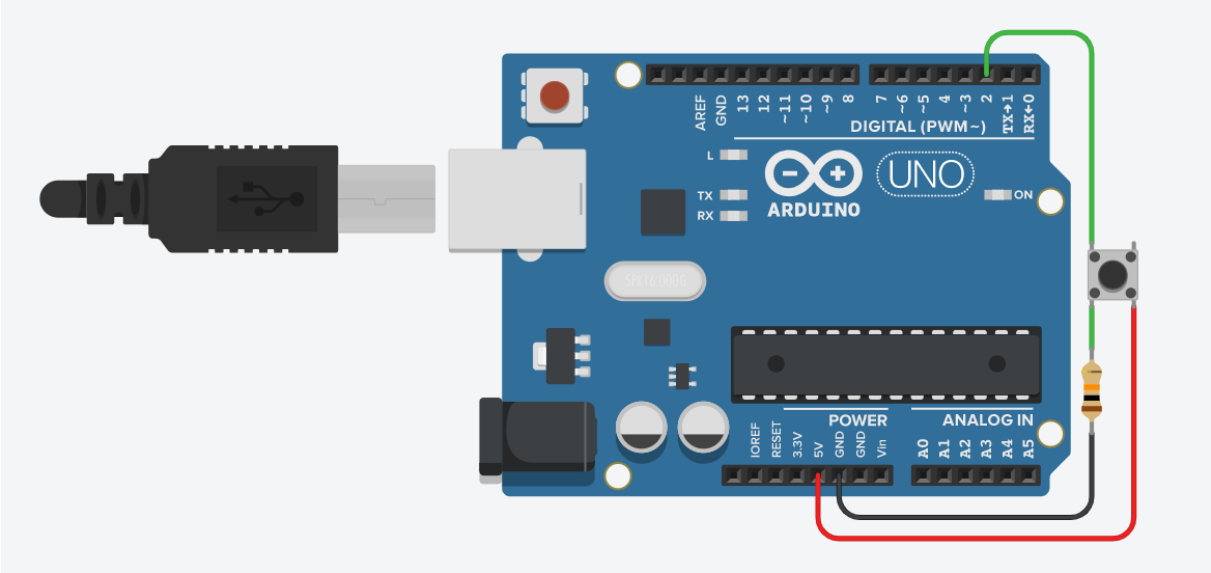
Rys. 5. Czujnik temperatury.

W tym programie został wykorzystany analogowy czujnik temperatury (cyfrowy niestety nie jest dostępny w symulatorze) (Rys. 5). Program realizuje odczyt wartości, którą następnie przeliczamy na wartość napięcia. Otrzymaną wartość konwertujemy na temperaturę, a następnie wypisujemy wartość w monitorze portu szeregowego.

```
int czujnik = A1;    //pin analogowy A1 połączony z sygnałem z czujnika
float VOLT;
float TEMP;

void setup(){
  Serial.begin(9600);          //inicjalizacja monitora szeregowego
  Serial.println("Test czujnika temperatury");
}

void loop(){
  int odczyt = analogRead(czujnik);        //odczytanie wartości z czujnika
  VOLT = (odczyt * 5.0) / 1024.0; //przeliczenie odczytanej wartości na napięcie
                                          w woltach (dla podłączenia pod 5 V)
  TEMP = (VOLT - 0.5) * 100;   //konwersja z napięcia na temperaturę, rozdzielczość
           czujnika wynosi 10 mV na stopień, dodatkowo należy zastosować offset 500 mV
  Serial.print("Temperatura (C): ");       //wyświetlenie jej na monitorze
  Serial.println(TEMP);
```

```
  delay(200);                          //opóźnienie między kolejnymi odczytami
}
```

Przycisk chwilowy



Rys. 6. Działanie przycisku monostabilnego.

Program ten przedstawia sposób działania przycisku monostabilnego.

```
int buttonState = 0;


void setup()
{
  pinMode(2, INPUT);
  pinMode(13, OUTPUT);
}


void loop()
{
  buttonState = digitalRead(2);

  if (buttonState == HIGH) {
    digitalWrite(13, HIGH);
  } else {
    digitalWrite(13, LOW);
  }
  delay(10);
}
```

Zadania do wykonania

Do każdego zadania proszę wykonać schemat oraz implementację programu.

Zad.1. Sygnalizacja czasowa.

Zbuduj symulator świateł drogowych. Światło czerwone i zielone mają się palić przez 30 sek. Światło pomarańczowe przełączające się pomiędzy czerwonym i zielonym ma trwać 3 sek.

Zad. 2. Sygnalizacja warunkowa.

Przerób program z zadania 1, aby zmieniał sygnalizację za pomocą przycisku chwilowego. Każde kliknięcie przycisku ma zmieniać kolor diody – wejściowo niech się pali jedna z głównych diod sygnalizacji (czerwona bądź zielona). W momencie kliknięcia przycisku, dioda włączona ma się wyłączyć, a włączyć się mają dwie pozostałe diody, jednak dioda pomarańczowa ma po chwili zgasnąć.

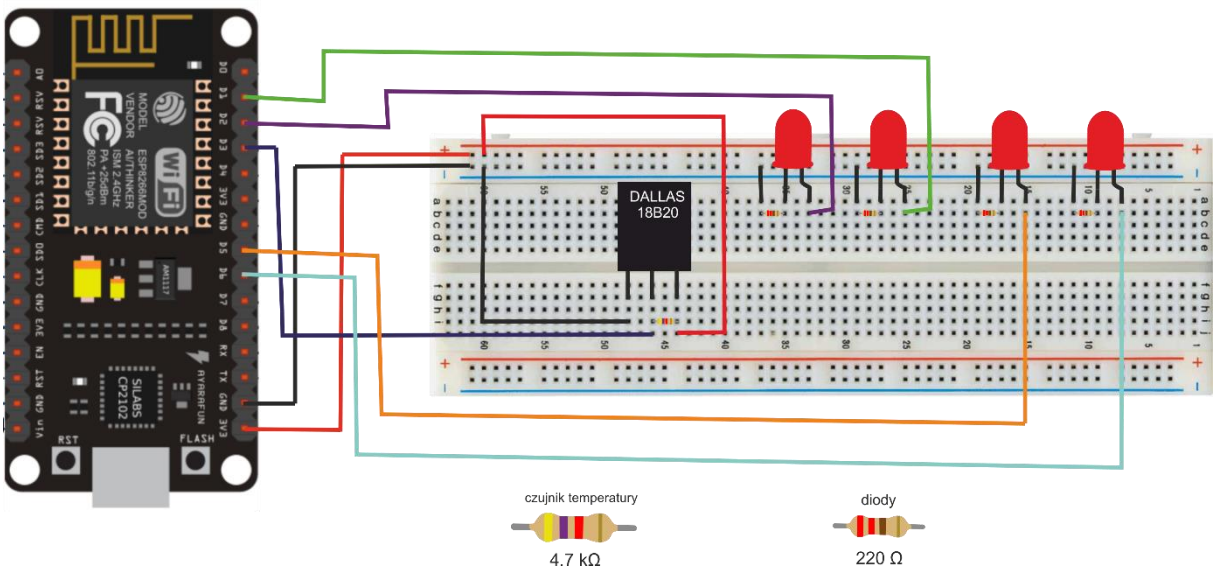
Zad. 3. Cyfrowy termometr.

Zaprojektuj obwód i zaimplementuj program z użyciem termometru, diody RGB i pojedynczej diody (lub z wykorzystaniem 4 osobnych diod różnego koloru). Dioda ma zmieniać kolor zależny od wprowadzonych zakresów temperaturowych. Dla temperatury poniżej 0 ma się świecić niebieski kolor, dla wartości od 0 do 10^{o}C kolor żółty, od 10 ^{o}C do 20 ^{o}C kolor zielony i od 20 ^{o}C wzwyż kolor czerwony.

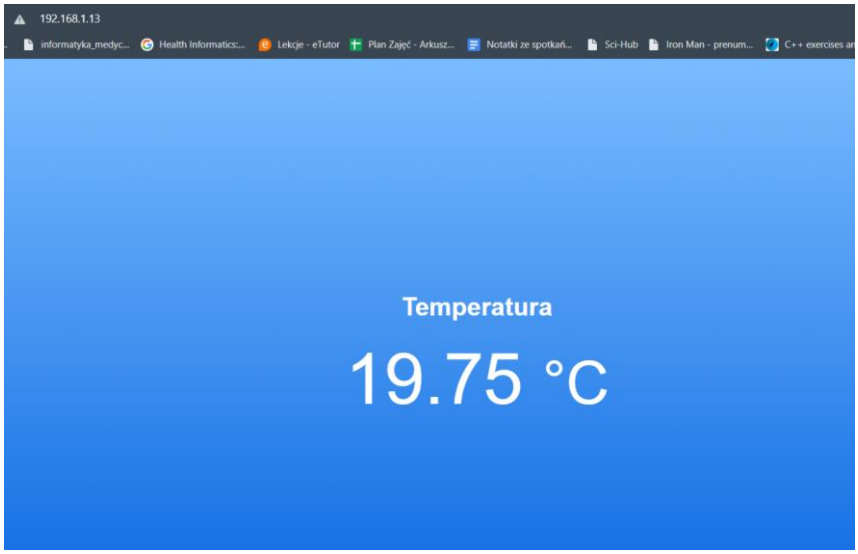
Dodatek

Rozwiązania dla fizycznego sprzętu

Zadanie trzecie wykorzystaniem cyfrowego termometru Dallas ds18b20 oraz informacją na prostym serwerze.



Rysunek przedstawia prosty schemat połączenia. W projekcie wykorzystano płytkę NodeMCU, ze względu na posiadanie na swoim pokładzie modułu Wi-Fi.



Kod programu:

```
#include <OneWire.h>
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <DallasTemperature.h>


#define ONE_WIRE_BUS D3


OneWire oneWire(ONE_WIRE_BUS);

DallasTemperature DS18B20(&oneWire);


//Dane WIFI

const char* ssid = "PLAY INTERNET 4G LTE-F928";
const char* password = "93642682";


ESP8266WebServer server(80);


//Termometr

char temperatureString[6];
const int led = 13;


float getTemperature() {
  float temp;
  do {
    DS18B20.requestTemperatures();
    temp = DS18B20.getTempCByIndex(0);
    delay(100);
     }
while (temp == 85.0 || temp == (-127.0));

return temp;
}


//DIODA

String output5State = "off";
String output4State = "off";
String output14State = "off";
String output12State = "off";
```

```
const int output5 = 5; //Pin D1
const int output4 = 4; //Pin D2
const int output14 = 14; //Pin D5
const int output12 = 12; //Pin D6


// obecny czas

unsigned long currentTime = millis();

// poprzedni czas

unsigned long previousTime = 0;

// definicja czasu w milisekundach

const long timeoutTime = 2000;

////////////////////////////////////

void setup(void){

  Serial.begin(115200);


  pinMode(output5, OUTPUT);
  pinMode(output4, OUTPUT);
  pinMode(output14, OUTPUT);
  pinMode(output12, OUTPUT);

  digitalWrite(output5, LOW);
  digitalWrite(output4, LOW);
  digitalWrite(output14, LOW);
  digitalWrite(output12, LOW);


  WiFi.begin(ssid, password);
  Serial.println("");


  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }


  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
```

```
server.on("/", [](){
  float temperature = getTemperature();
  dtostrf(temperature, 2, 2, temperatureString);


  String title = "Temperatura";
  String cssClass = "mediumhot";


    if (temperature < 25)
   {
    cssClass = "cold";
    output4State = "off";
    output5State = "off";
    output14State = "off";
    digitalWrite(output4, LOW);
    digitalWrite(output5, LOW);
    digitalWrite(output14, LOW);
    }

    else if(temperature > 25 && temperature < 27)
   {
    cssClass = "cold";
    output4State = "on";
    output5State = "off";
    output14State = "off";
    digitalWrite(output4, HIGH);
    digitalWrite(output5, LOW);
    digitalWrite(output14, LOW);
    }

    else if (temperature > 27 && temperature < 29)
    {
    cssClass = "mediumhot";
    output4State = "on";
    output5State = "on";
    output14State = "off";
    digitalWrite(output5, HIGH);
    digitalWrite(output4, HIGH);
    digitalWrite(output14, LOW);
    }

   else if (temperature > 29 && temperature < 31)
    {
    cssClass = "mediumhott";
    output4State = "on";
    output5State = "on";
    output14State = "on";
    digitalWrite(output4, HIGH);
    digitalWrite(output5, HIGH);
    digitalWrite(output14, HIGH);
    }
```

```
    else if (temperature > 31)
      {
      cssClass = "hot";
      output4State = "off";
      output5State = "off";
      output14State = "off";
      digitalWrite(output4, LOW);
      digitalWrite(output5, LOW);
      digitalWrite(output14, LOW);


      digitalWrite(output12, HIGH);
      delay(5);
      digitalWrite(output12, LOW);
      delay(5);
      }


    //HTML

    String message = "<!DOCTYPE html><html><head><title>" + title +
"</title><meta charset=\"utf-8\" /><meta name=\"viewport\"
content=\"width=device-width\" /><style>\n";

    message += "html {height: 100%;}";

    message += "div {color: #fff;font-family: 'Arial';font-weight:
400;left: 50%;position: absolute;text-align: center;top:
50%;transform: translateX(-50%) translateY(-50%);}";

    message += "h2 {font-size: 90px;font-weight: 400; margin: 0}";

    message += "body {height: 100%;}";

    message += ".cold {background: linear-gradient(to bottom,
#7abcff, #0665e0 );}";

    message += ".mediumhot {background: linear-gradient(to bottom,
#81ef85,#057003);}";

    message += ".mediumhott {background: linear-gradient(to bottom,
#fe7f00, #ff8413);}";

    message += ".hot {background: linear-gradient(to bottom,
#fcdb88,#d32106);}";

    message += "</style>";

    message += "<meta http-equiv=\"refresh\" content=\"1\">";

    message += "</head><body class=\"" + cssClass + "\"><div><h1>" +
title +  "</h1><h2>" + temperatureString +
" <small>&deg;C</small></h2></div></body></html>";


    server.send(200, "text/html", message);
```

```
  });

  server.begin();

  Serial.println("Start strony!");
}

void loop(void){
  server.handleClient();
}
```