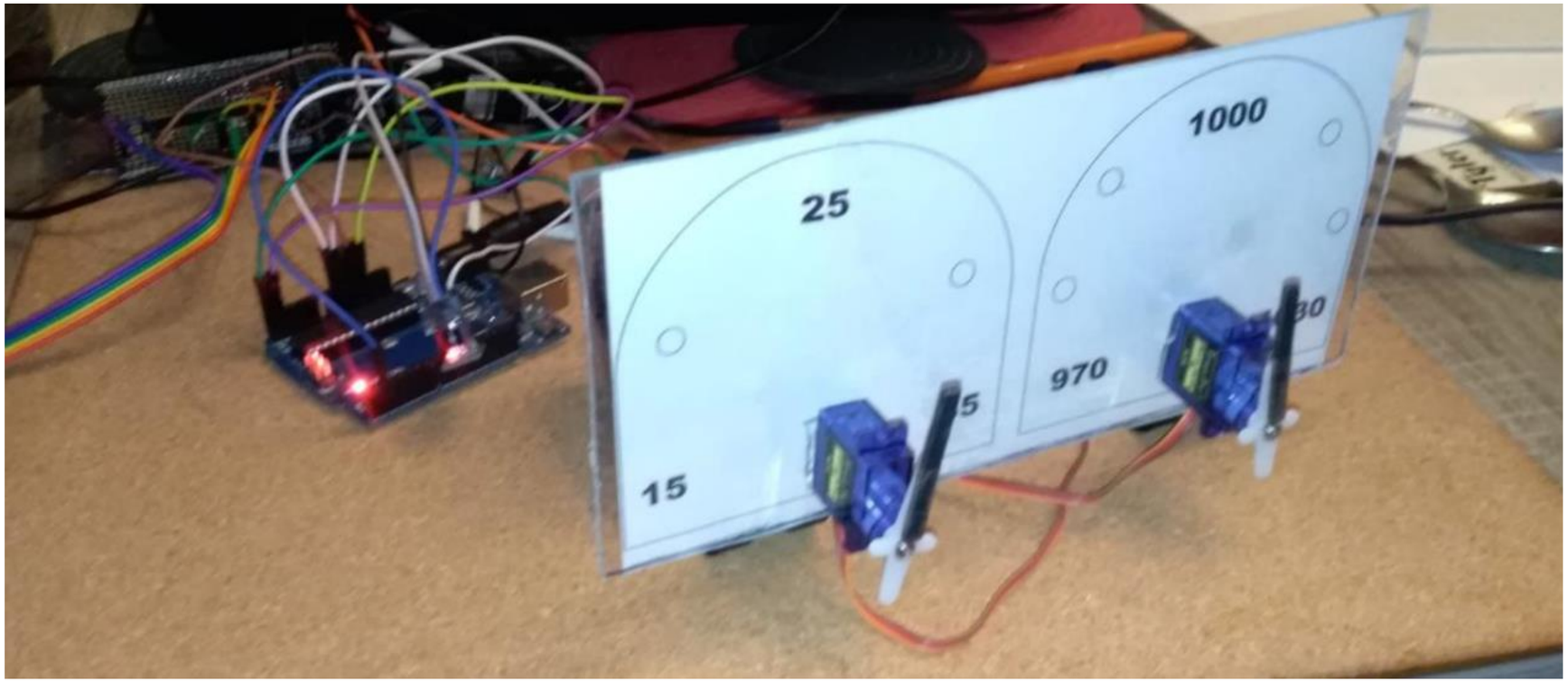
Projekt stacji meteo



Aleksander Hebda

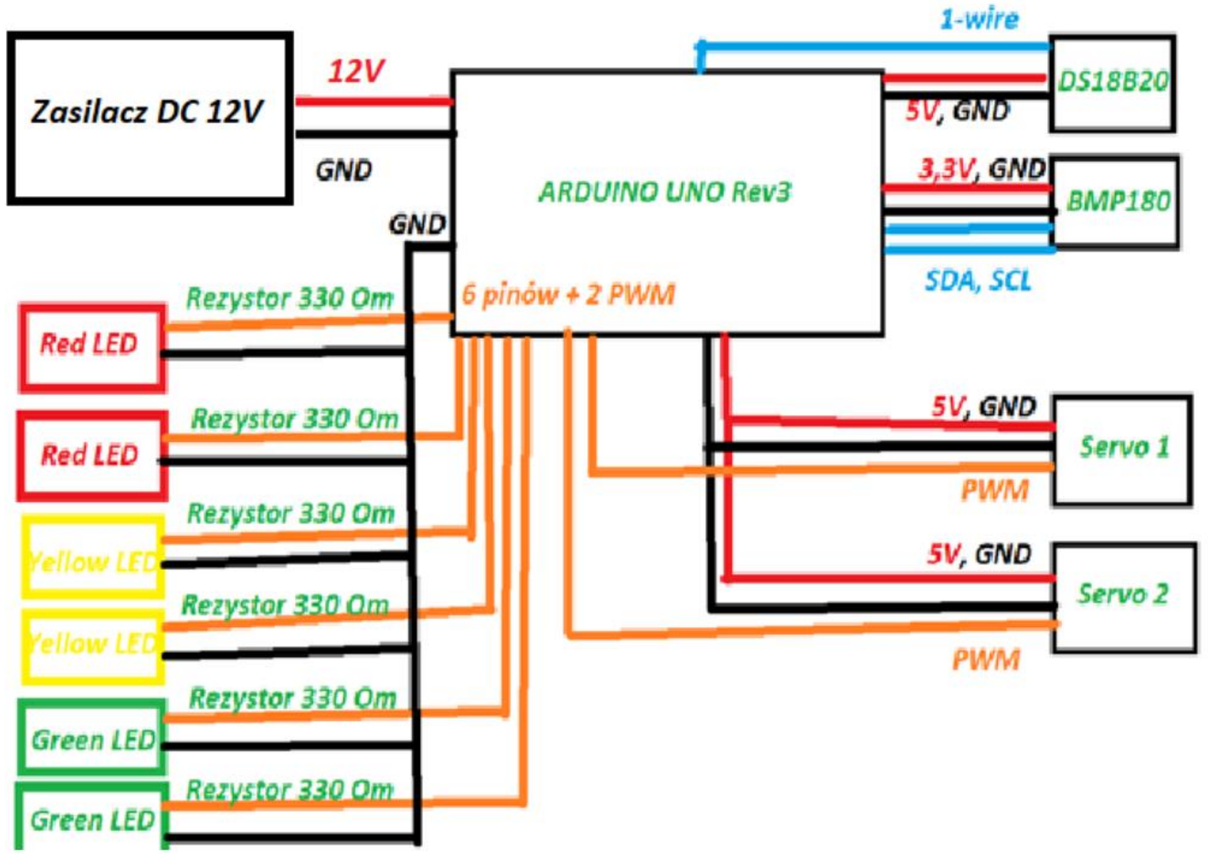
Opis projektu:

Celem projektu jest utworzenie analogowego przedstawienia mierzonej temperatury oraz panującego ciśnienia atmosferycznego za pomocą serwomechanizmów modelarskich (zegary wskazówkowe) na przygotowanej wcześniej tarczy, dodatkowo podświetlenie tych zegarów w zależności od warunków panujących.

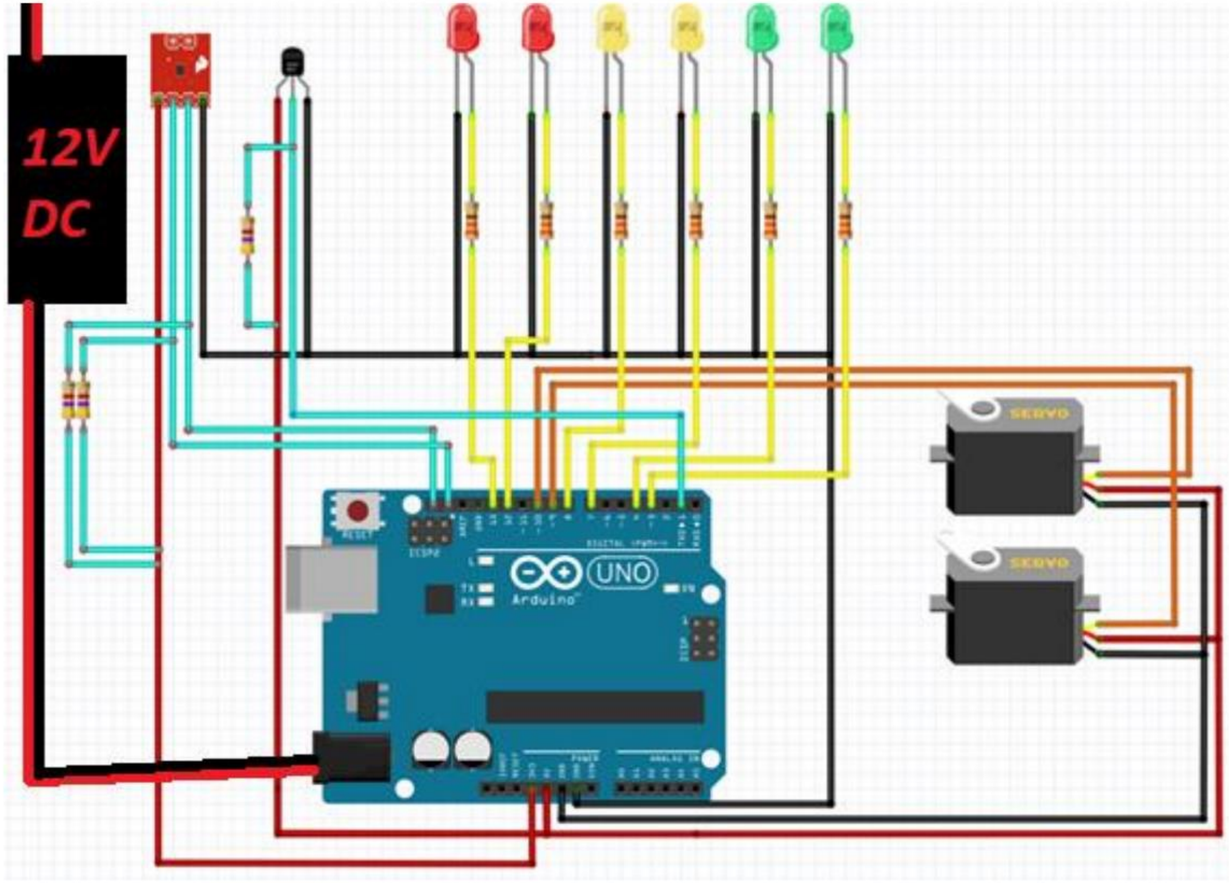
Założenia projektu:

W skład układu wchodzi mikrokontroler Arduino – jako jednostka sterująca. Elementy mierzące temperaturę oraz ciśnienie to termometr cyfrowy DS18B20 oraz czujnik ciśnienia BMP180. DS18B20 pracuje na magistrali 1-wire, zaś barometr na magistrali I2C. Za wskazywanie temperatury i ciśnienia na tarczach posłużą dwa serwomechanizmy modelarskie o kącie 180 stopni. Całość zasilana będzie za pomocą dwóch ogniw litowo-jonowych XTAR 2600mAh, podłączonych szeregowo w celu uzyskania nominalnych 7,4V zasilania. Opcjonalnie jeżeli jest potrzeba ciągłej niezakłóconej rozładowaniem ogniw pracy, można wykorzystać zasilacz impulsowy 12V. Oświetlenie będą pełnić diody sygnalizacyjne LED o prądzie 20mA, podpięte za pomocą odpowiednich rezystorów (330 om), celem ograniczenia prądu przepływającego przez diody.

Schemat ideowy układu sterowania:

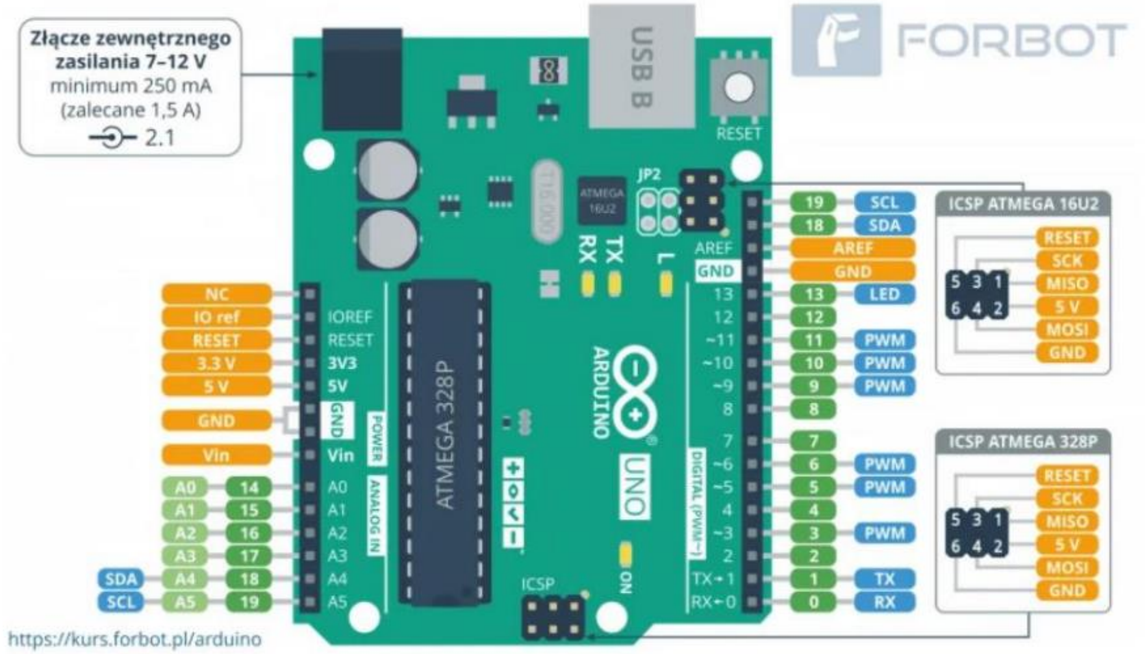


Schemat podłączenia całego projektu wykonany w programie Fritzing



Części wchodzące w skład projektu:

Mikrokontroler Arduino Uno Rev3:



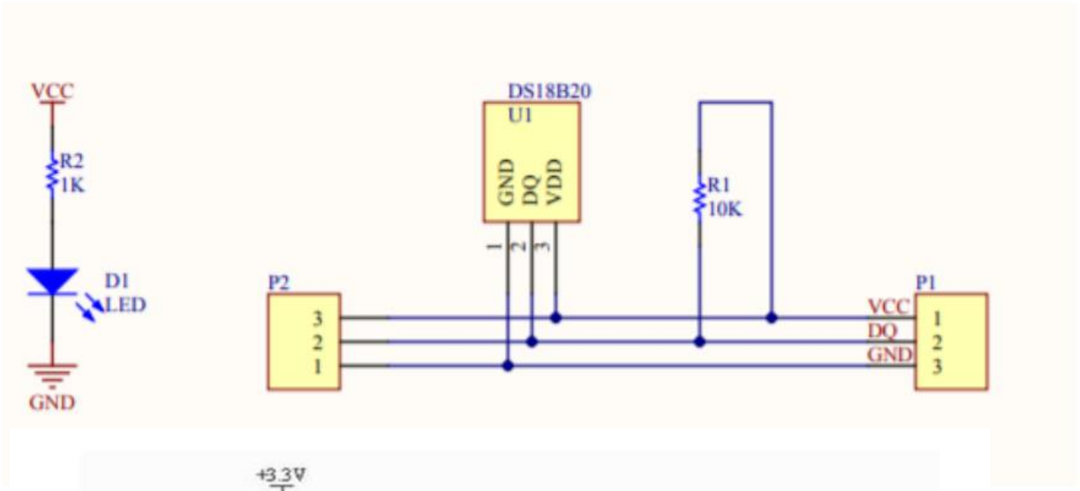
Części wchodzące w skład projektu:

Moduł czujnika temperatury DS18B20 oraz moduł czujnika ciśnienia BMP180:

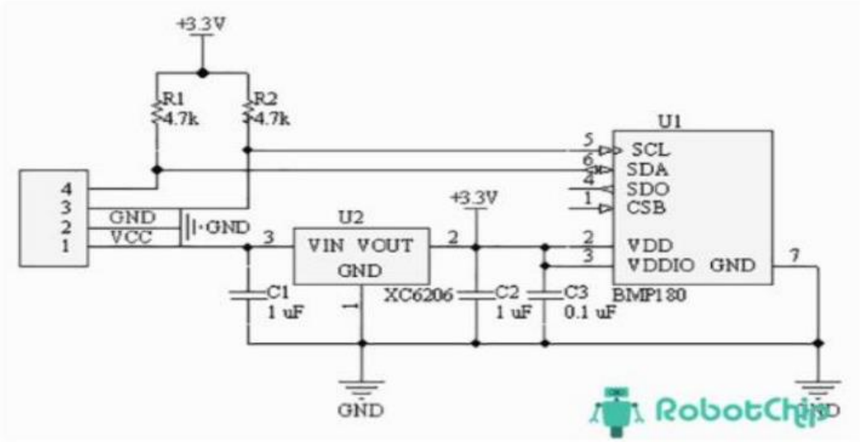
Moduł termometru DS18B20

Moduł barometru BMP180

Części wchodzące w skład projektu:



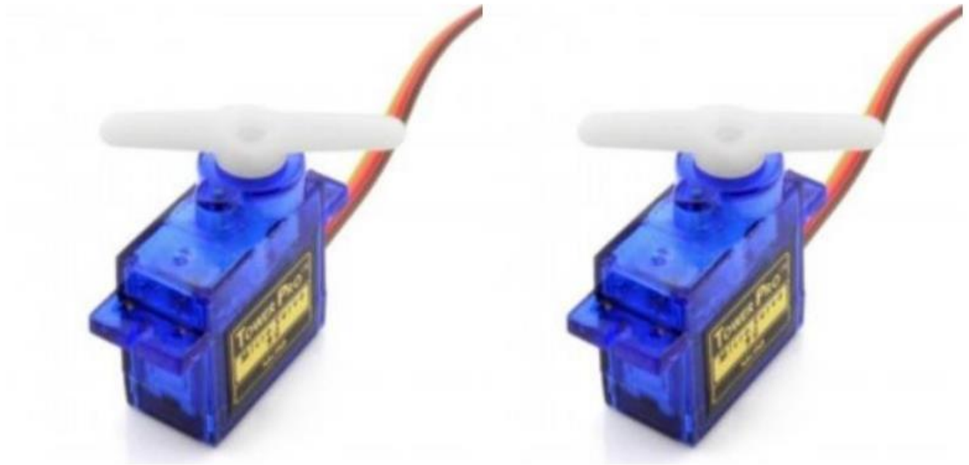
Moduł termometru DS18B20

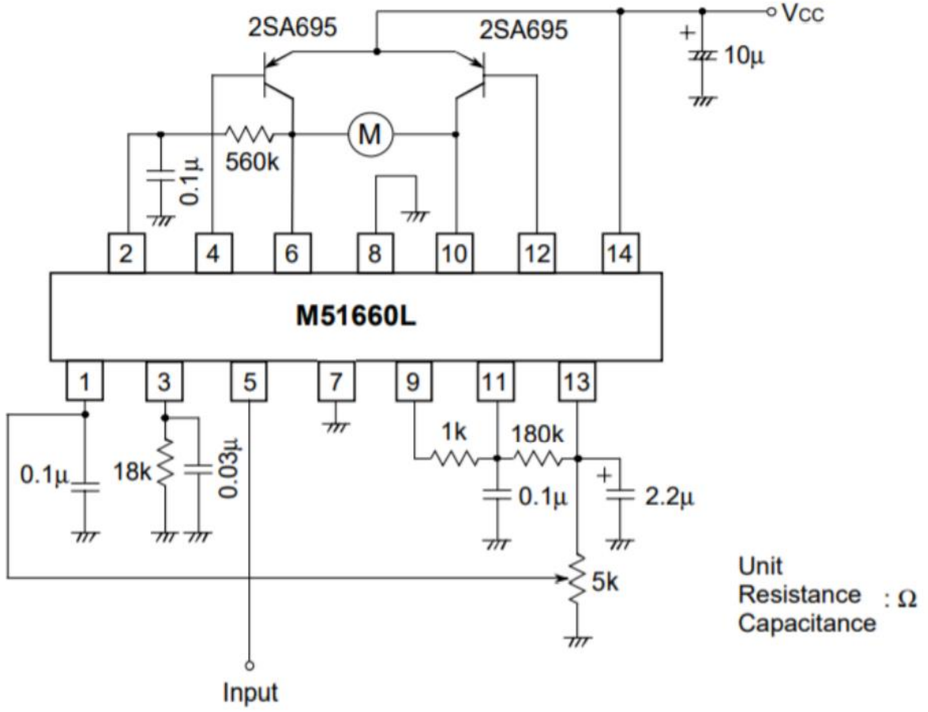


Moduł barometru BMP180

Części wchodzące w skład projektu:

Serwomechanizmy TowerPro SG-90 typu micro – 180 stopni





Części wchodzące w skład projektu:

Diody LED białe 20mA z osłonami: zielonymi, żółtymi, czerwonymi

$$R = \frac{U_{zas} - U_{diody}}{I_{diody}}$$

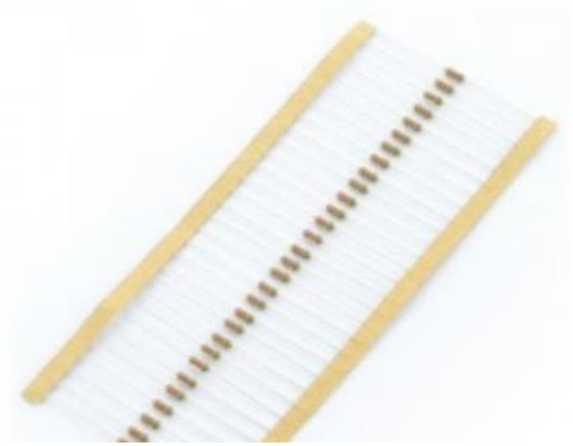
Wzór na obliczenie poprawnej rezystancji oporników do pracy z diodami

Części wchodzące w skład projektu:

Dioda LED biała napięcie odkładane na diodzie 3.0V - 3,6V

$$R = \frac{5V - 3V}{0{,}020A} = 100\Omega$$

Z racji braku potrzeby montowania rezystora 100 om, ze względu na niewielkie korzyści jasności a większy prąd w obwodzie, wybrane są rezystory THT CF węglowe: 330 om, 1/8 W



Program Arduino:

Wykorzystujemy program do programowania płytek Arduino, czyli program „ArduinoIDE”.

Dołączamy biblioteki dotyczące obsługi serwomechanizmów, termometru DS18B20, oraz barometru BMP180 (#include).

W „Void Setup” inicjujemy 6 pinów cyfrowych do obsługi zapalania i gaszenia 6 diod LED, komendą „PinMode”, następnie numer pinu oraz że w trybie: „OUTPUT”. Ustawiamy początkowe wartości wszystkich diod na „LOW” komendą „digitalWrite”. Inicjujemy dwa piny cyfrowe typu PWM jako piny obsługujące serwomechanizmy i ustawiamy wychylenie na minimum. Sprawdzamy połączenie między czujnikami a kontrolerem. Jeżeli nie ma go, to powtarzamy do skutku, jeżeli jest ok to program przechodzi dalej.

Program Arduino:

W „Void loop” na bieżąco sprawdzamy odpowiedzi z czujników i na ich podstawie dokonujemy zmian położenia serwomechanizmów, tak aby wartości zmierzone pokrywały się z rzeczywistym wyświetlaniem na zegarach. Do tego celu używamy komendy „map”, dzięki której otrzymujemy przelicznik wartości temperatury i ciśnienia (granic, które chcemy przedstawić na zegarach) na wychylenie serwomechanizmów (0’ – 180’). Wartość uzyskaną z „map” wrzucamy do funkcji „Servo.Write” dzięki temu otrzymujemy wychylenie potrzebne do odzwierciedlenia wartości na zegarach. Aby program nie powtarzał się bez sensu bardzo dużą liczbę razy na jednostkę czasu, stosujemy funkcję opóźniającą „millis”, która pozwala na wstrzymanie wykonywania dalszych instrukcji, przed upływem wskazanej przerwy. Funkcja „delay” niby działa podobnie, lecz zamraża wykonywanie programu, nie można podczas tej funkcji wykonywać innych czynności, a funkcja „millis” robi przerwę ale umożliwia wykonywanie zadań innych podczas tej pauzy. Jeżeli na zegarze osiągniemy wystarczające wartości temperatury oraz ciśnienia to odpowiednie kolory diod będą wspomagały to przedstawienie, włączając się po przekroczeniu wartości granicznych.

Program Arduino:

Implementacja bibliotek

```
#include <DallasTemperature.h>                    //biblioteka termometru DS18B20
#include <OneWire.h>                              //biblioteka obsługi magistrali OneWire
#include <SFE_BMP180.h>                           //biblioteka barometru BMP180
#include <Servo.h>                                //biblioteka obsługi serwomechanizmów
#include <Wire.h>
```

Ustawienie parametrów startowych

```
#define ONE_WIRE_BUS 2                            //zdefiniowanie cyfrowego pinu nr 2 jako OneWire
OneWire oneWire(ONE_WIRE_BUS);                    //zdefiniowanie cyfrowego pinu nr 2 jako OneWire
DallasTemperature sensors(&oneWire);              //powiązanie magistrali z biblioteką termometru
SFE_BMP180 pressure;                              //powiązanie zmiennej pressure z biblioteką BMP180
Servo servo_1;                                    //zdefiniowanie zmiennej servo_1 jako serwomechanizm
Servo servo_2;                                    //zdefiniowanie zmiennej servo_2 jako serwomechanizm

int zielona_1 = 3;                                //dioda zielona_1 jako pin cyfrowy nr 3
int zielona_2 = 4;                                //dioda zielona_2 jako pin cyfrowy nr 4
int zolta_1 = 7;                                  //dioda zolta_1 jako pin cyfrowy nr 7
int zolta_2 = 8;                                  //dioda zolta_2 jako pin cyfrowy nr 8
int czerwona_1 = 12;                              //dioda czerwona_1 jako pin cyfrowy nr 12
int czerwona_2 = 13;                              //dioda czerwona_2 jako pin cyfrowy nr 13
```

Program Arduino:

```
21 void setup() {
22
23 Serial.begin(9600);                              //ustawiona predkość komunikacji z komputerem
24 sensors.begin();                                 //ustawienie komunikacji z czujnikiem
25 if (pressure.begin())                            //sprawdzenie połączenia
26 {
27     Serial.println("BMP180 dziala");             //jeżeli odpowiada program wykonuje się dalej
28     } else {
29     Serial.println("BMP180 nie dziala\n\n");     //jeżeli nie działa przerwij wykonywanie programu
30     while(1);                                    //pętla nieskończona
31           }
32 servo_1.attach(9);                               //przypisanie servo_1 do pinu PWM nr 9
33 servo_2.attach(10);                              //przypisanie servo_2 do pinu PWM nr 10
34
35 pinMode(zielona_1, OUTPUT);                      //zielona_1 jako wyjście
36 digitalWrite(zielona_1, LOW);                    //zielona_1 stan niski
37 pinMode(zielona_2, OUTPUT);                      //zielona_2 jako wyjście
38 digitalWrite(zielona_1, LOW);                    //zielona_2 stan niski
39 pinMode(zolta_1, OUTPUT);                       //zolta_1 jako wyjście
40 digitalWrite(zolta_1, LOW);                     //zolta_1 stan niski
41 pinMode(zolta_2, OUTPUT);                       //zolta_2 jako wyjście
42 digitalWrite(zolta_2, LOW);                     //zolta_2 stan niski
43 pinMode(czerwona_1, OUTPUT);                    //czerwona_1 jako wyjście
44 digitalWrite(czerwona_1, LOW);                  //czerwona_1 stan niski
45 pinMode(czerwona_2, OUTPUT);                    //czerowna_2 jako wyjście
46 digitalWrite(czerwona_2, LOW);                  //czerwona_2 stan niski
47
48 }
```

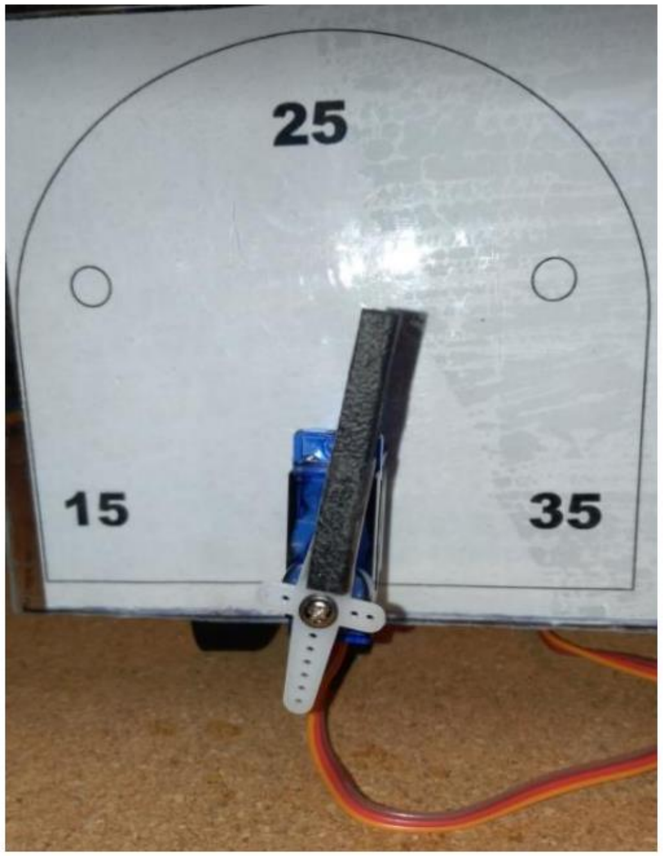
Program Arduino:

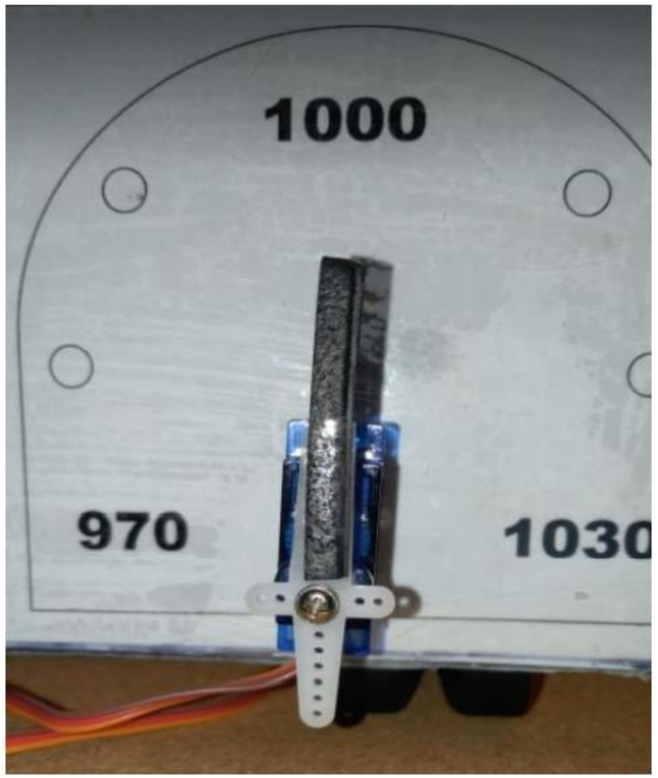
```
50 void loop() {
51
52 sensors.requestTemperatures();                    //odczytanie wartości temeratury z DS18B20
53 float tempC = sensors.getTempCByIndex(0);         //przekonwertowanie tej wartości na stopnie celcjusza
54 char status;                                      //zmienna typu char do
55 double T, P;                                  //zmienne typu double dla wartości temperatury
56 status = pressure.startTemperature();            //rozpoczęcie pomiaru temperatury
57 if (status != 0){
58     delay(status);                               //odczekanie aż pomiar zostanie zakończony
59     status = pressure.getTemperature(T);         //przekonwertowanie na stopnie Celcjusza
60     if (status != 0){
61       Serial.print(T,2);                         //wypisanie wartości z dokładnością do dwóch miejsc po przecinku
62       Serial.println("C, ");                     //dopisanie po tym, że stopnie Celcjusza
63 status = pressure.startPressure(3);              //rozpoczęcie pomiaru ciśnienia
64 if (status != 0){
65      delay(status);                              //odczekanie aż pomiar zostanie zakończony
66      status = pressure.getPressure(P,T);         //przeliczenie wartości na hPa
67      if (status != 0){
68        Serial.print(P,2);                        //wypisanie wartości z dokładnością do dwóch miejsc po przecinku
69        Serial.println("hPa, ");                  //opisanie po tym, że hPa
70         }
71         else Serial.println("błąd przy pomiarze ciśnienia_koniec\n");
72       }
73       else Serial.println("błąd przy pomiarze ciśnienia_start\n");
74     }
75     else Serial.println("błąd przy pomiarze temperatury_koniec\n");
76   }
77   else Serial.println("błąd przy pomiarze temperatury_start\n");
78
```

Program Arduino:

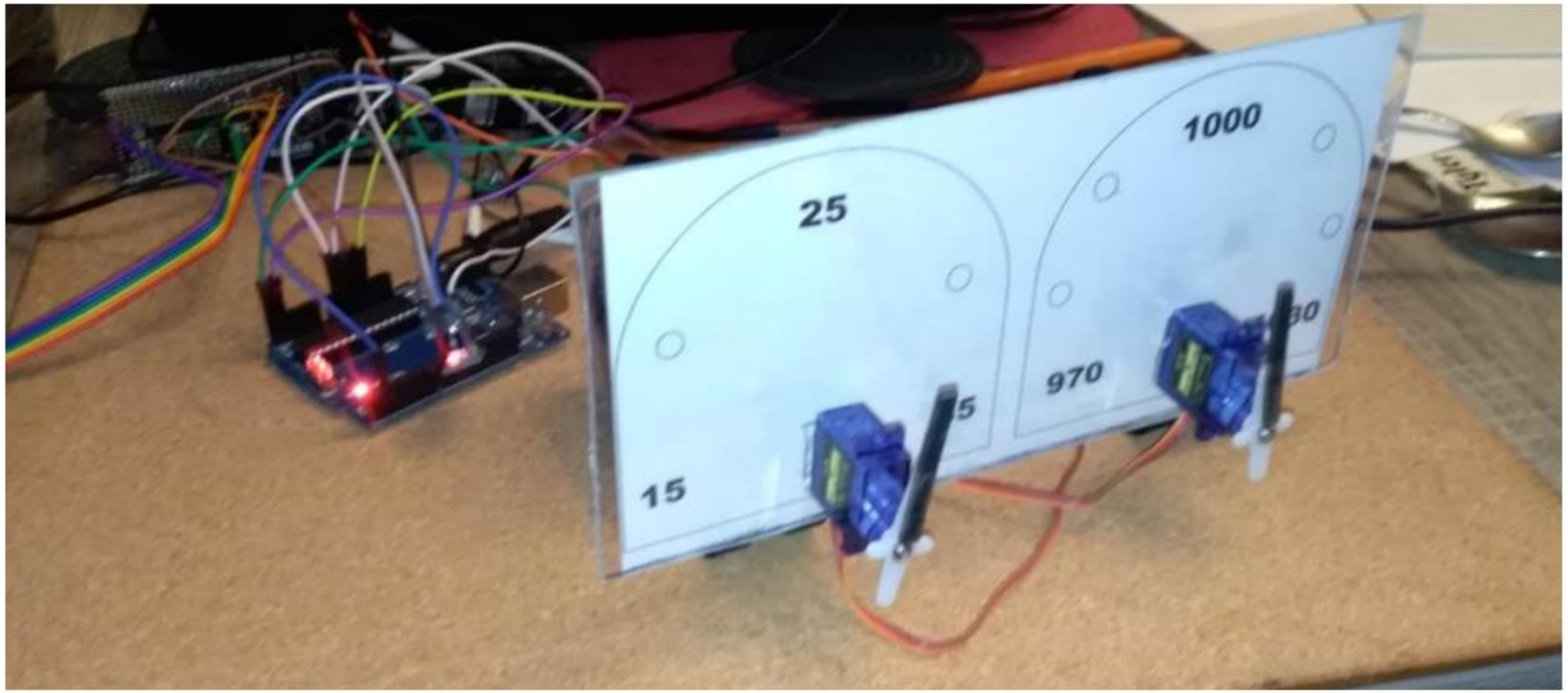
```
int val_1 = tempC;                                  //przypisanie zmiennej wychylenia wartości temperatury
int val_2 = P;                                      //przypisanie zmiennej wychylenia wartości ciśnienia
val_1 = map(val_1, 15, 35, 0, 180);                 //przeskalowanie wartości wychylenia dla zakresu temperatury
val_1 = 180 - val_1;                                //zmiana wychylenia z CCW na CW
servo_1.write(val_1);                               //ustawienie wychylenia serwomechanizmu nr 1
Serial.println(val_1);                              //wypisanie aktualnego wychylenia (diagnostyka)
val_2 = map(val_2, 950, 1050, 0, 180);              //przeskalowanie wartości wychylenia dla zakresu ciśnienia
val_2 = 180 - val_2;                                //zminana wychylenia z CCW na CW
servo_2.write(val_2);                               //ustawienie wychylenia serwomechanizmu nr 2
Serial.println(val_2);                              //wypisanie aktualnego wychylenia (diagnostyka)

if (tempC < 25){                                        //sekcja z warunkami dla zapalania konkretnych diod
  digitalWrite(zielona_1, HIGH);
  digitalWrite(zolta_1, LOW);
  digitalWrite(czerwona_1, LOW);
}
if (temp > 25){
  digitalWrite(zielona_1, HIGH);
  digitalWrite(zolta_1, HIGH);
  digitalWrite(czerwona_1, LOW);
}
if (temp > 30){
  digitalWrite(zielona_1, HIGH);
  digitalWrite(zolta_1, HIGH);
  digitalWrite(czerwona_1, HIGH);
}
if (P > 990){
  digitalWrite(zielona_2, HIGH);
  digitalWrite(zolta_2, LOW);
  digitalWrite(czerwona_2, LOW);
}
if (P < 990){
  digitalWrite(zielona_2, HIGH);
  digitalWrite(zolta_2, HIGH);
  digitalWrite(czerwona_2, LOW);
}
if (P < 970){
  digitalWrite(zielona_2, HIGH);
  digitalWrite(zolta_2, HIGH);
  digitalWrite(czerwona_2, HIGH);
}
delay(250);                                             //opóźnienie pętli
}
```

Gotowy układ:

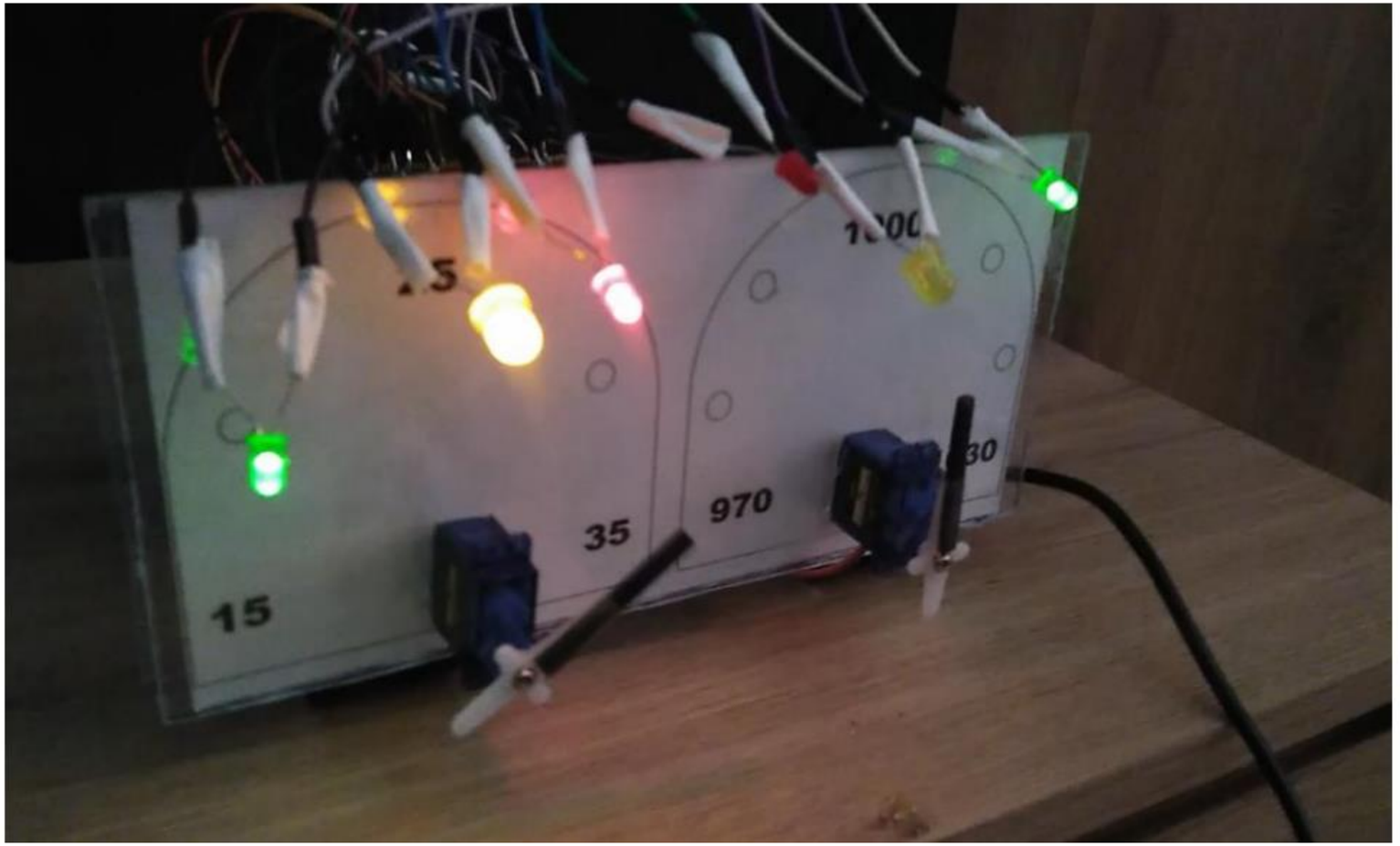




Gotowy układ:



Gotowy układ:



Dziękuję za uwagę