



Projekt zaliczeniowy  
**Regulator-PID-ogrzewanie**

Julia Wałowska 153302  
Józef Wiatr 51088

## 1. Opis Projektu

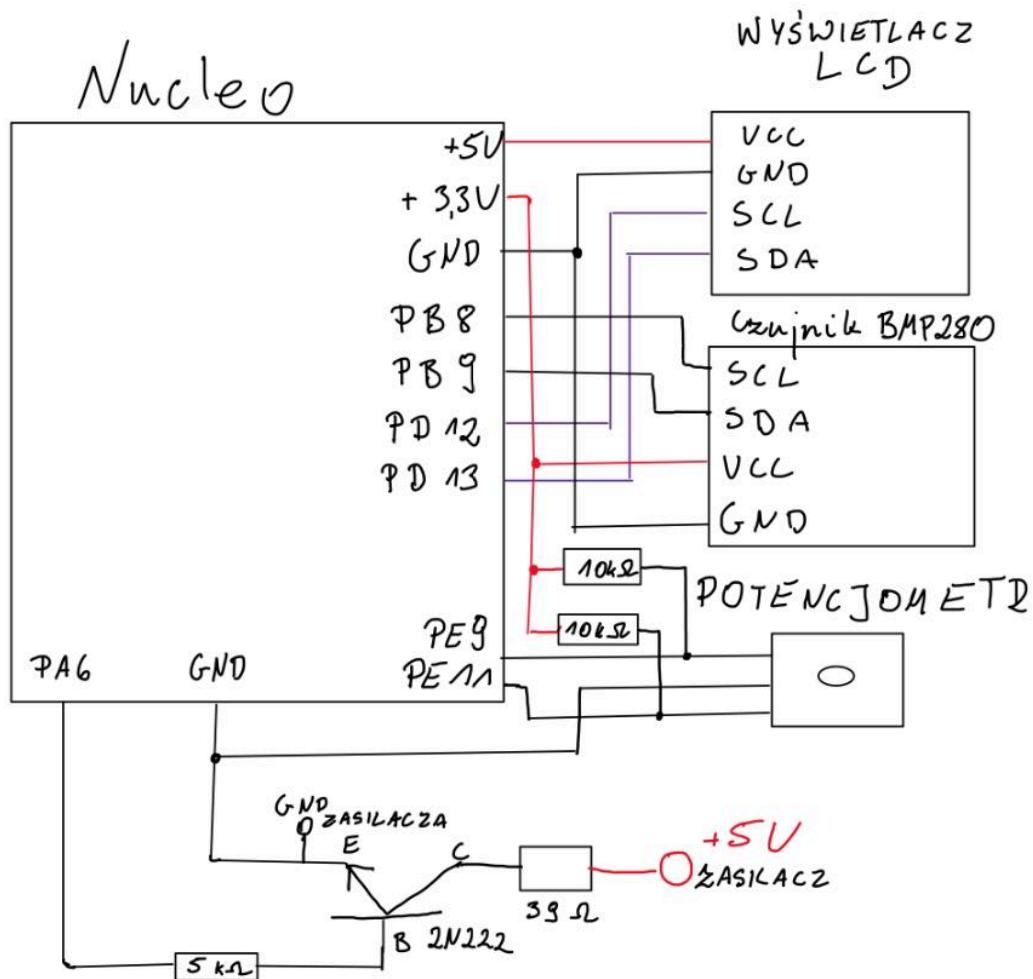
Celem projektu było zbudowanie układu automatycznej regulacji temperatury w oparciu o:

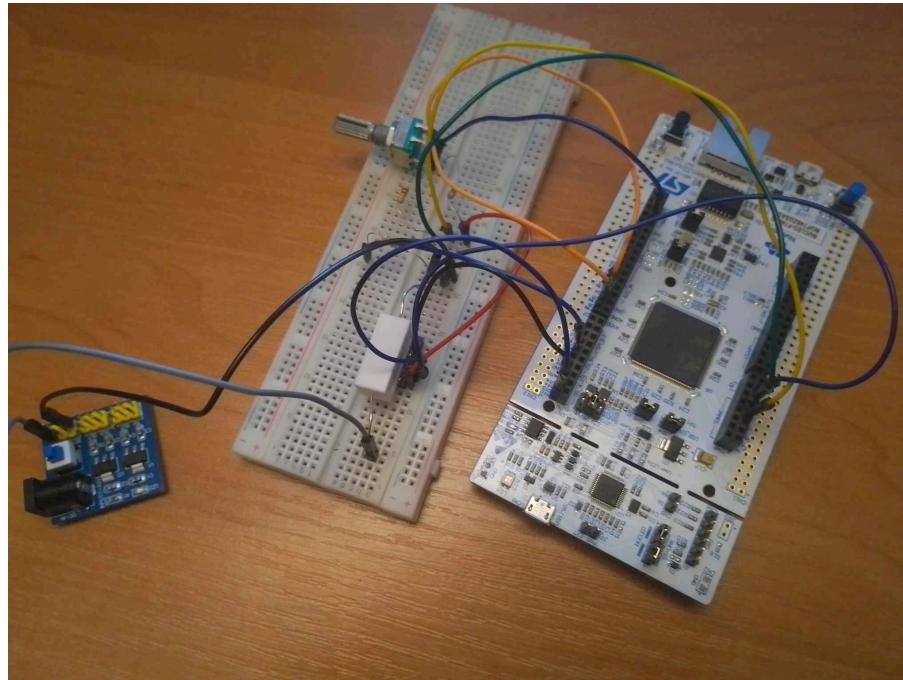
- płytę NUCLEO 144
- regulator PID
- rezystor  $39\Omega$
- czujnik BMP280

Poza elementami podstawowymi wykorzystano również:

- enkoder
- zasilacz 5W
- przewody
- płytka stykowa
- rezystory

## 2. Schemat układu





Zdjęcie 1: Połączony układ

### 3. Analiza modelu układu

Do przeanalizowania układu na początku wykonano obliczenia analityczne:

$$G_p(s) = \frac{1 - \frac{T_0}{2} \cdot s}{1 + \frac{T_0}{2} \cdot s} \cdot \left( k_p + \frac{k_i}{s} + k_d \cdot s \right) \cdot \frac{k}{1 + T_s}$$

$$G(s) = \frac{G_p(s)}{1 + G_p(s)} = \frac{L(s)}{L(s) + M(s)}$$

$$L(s) = \left( 1 - \frac{T_0}{2} \cdot s \right) \cdot k \cdot \left( k_p \cdot s + k_i + k_d \cdot s^2 \right)$$

$$M(s) = \left( 1 + \frac{T_0}{2} \cdot s \right) \cdot s \cdot \left( 1 + T_s \right) = \left( 1 + \frac{T_2 \cdot T_0}{2} + \frac{T_0}{2} + T_s \right) \cdot s$$

$$L(s) = k \cdot \left[ \left( k_d - \frac{T_0 \cdot k_p}{2} \right) \cdot s^2 + \left( k_p - k_i \cdot \frac{T_0 \cdot s}{2} \right) \cdot s - k_d \cdot s^2 \cdot \frac{T_0}{2} + k_i \right]$$

$$M_0(s) = L(s) + M(s):$$

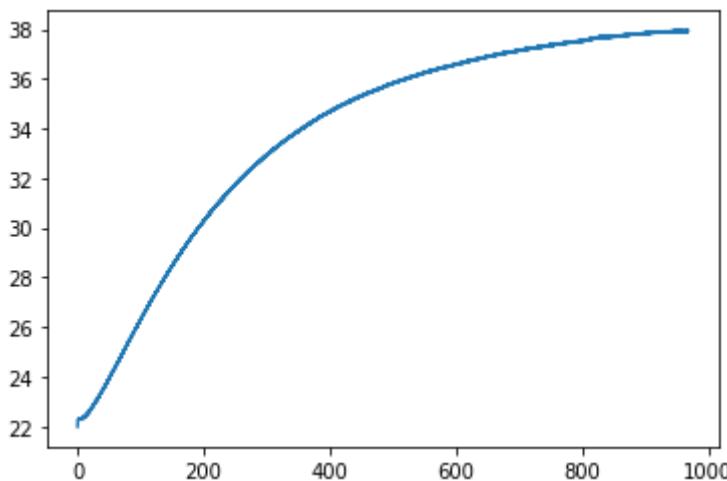
$$a = -k \cdot k_d \cdot \frac{T_0}{2} + \frac{T \cdot T_0}{2}$$

$$b = k \cdot k_d \cdot \frac{T_0 \cdot k_p}{2} + \frac{T_0}{2} = T$$

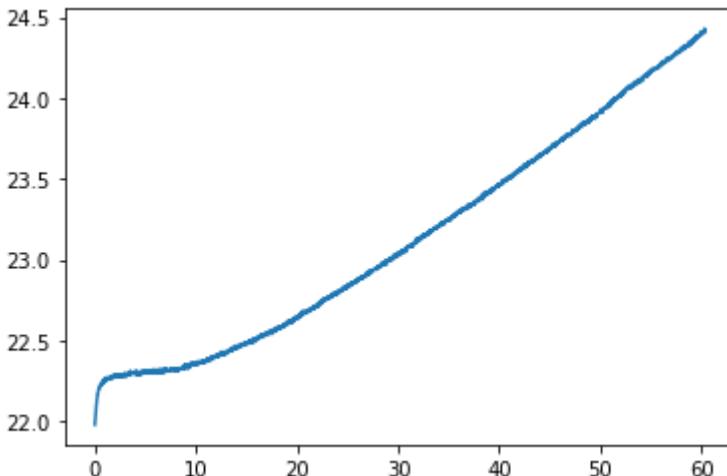
$$c = k_p \cdot k - k \cdot k_i \cdot T_0 \cdot \frac{1}{2} + 1$$

$$d = k_i$$

Wniosek z obliczeń - nie można lokować biegunów.



Wykres 1: Odpowiedź skokowa na podstawie której wyznaczono parametry.



Wykres 2: Zbliżenie na początek wykresu.

Następnie na podstawie odpowiedzi skokowej i przygotowanego skryptu wyliczono  $T_0$ ,  $T$  oraz wzmacnienie obiektu k Analizy dokonano na podstawie odpowiedzi skokowej i dopasowania obiektu inercyjnego z opóźnieniem transportowym, a wyniki zapisano w pliku. Wyliczone wartości zaimplementowano w modelu.

timeconstant	float64	1	191.13326394557953
timedelay	float64	1	71.1564816236496
k	float64	1	6.103515625e-05

Wyliczone parametry są dla transmitancji pomiędzy sygnałem sterującym tj. wypełnieniem(konwencjonalnym 0-100%) razy 65536(w takiej formie jest ono podawane w kodzie) a temperaturą odczytywaną z czujnika.

W celu zapewnienia lepszej regulacji zaimplementowano także wind-up:

```

if(calkowanyuchyb*ki>65535){
    calkowanyuchyb=65535/ki;
}
if(calkowanyuchyb*ki<-65535){
    calkowanyuchyb=-65535/ki;
}

```

Oraz poddano sygnał prostej filtracji przed podaniem go różniczkowaniu:

```

temperature=BMP280_ReadTemperature();
//SBMP280_ReadTemperatureAndPressure(&temperature, &pressure);
poprzedniuchyb=filtrowanyuchyb;
uchyb=zadana-temperature;
filtrowanyuchyb=0.20787957635076193/2*filtrowanyuchyb+(1-0.20787957635076193/2)*uchyb;
calkowanyuchyb=calkowanyuchyb+uchyb;
pTp=Tp;
Tp=HAL_GetTick();
sterowanie=round(kp*uchyb+kd/(Tp-pTp)*(filtrowanyuchyb-poprzedniuchyb)+ki*calkowanyuchyb);
if(sterowanie<0){
    sterowanie=0;
}
if(sterowanie>65535){
    sterowanie=65535;
}

```

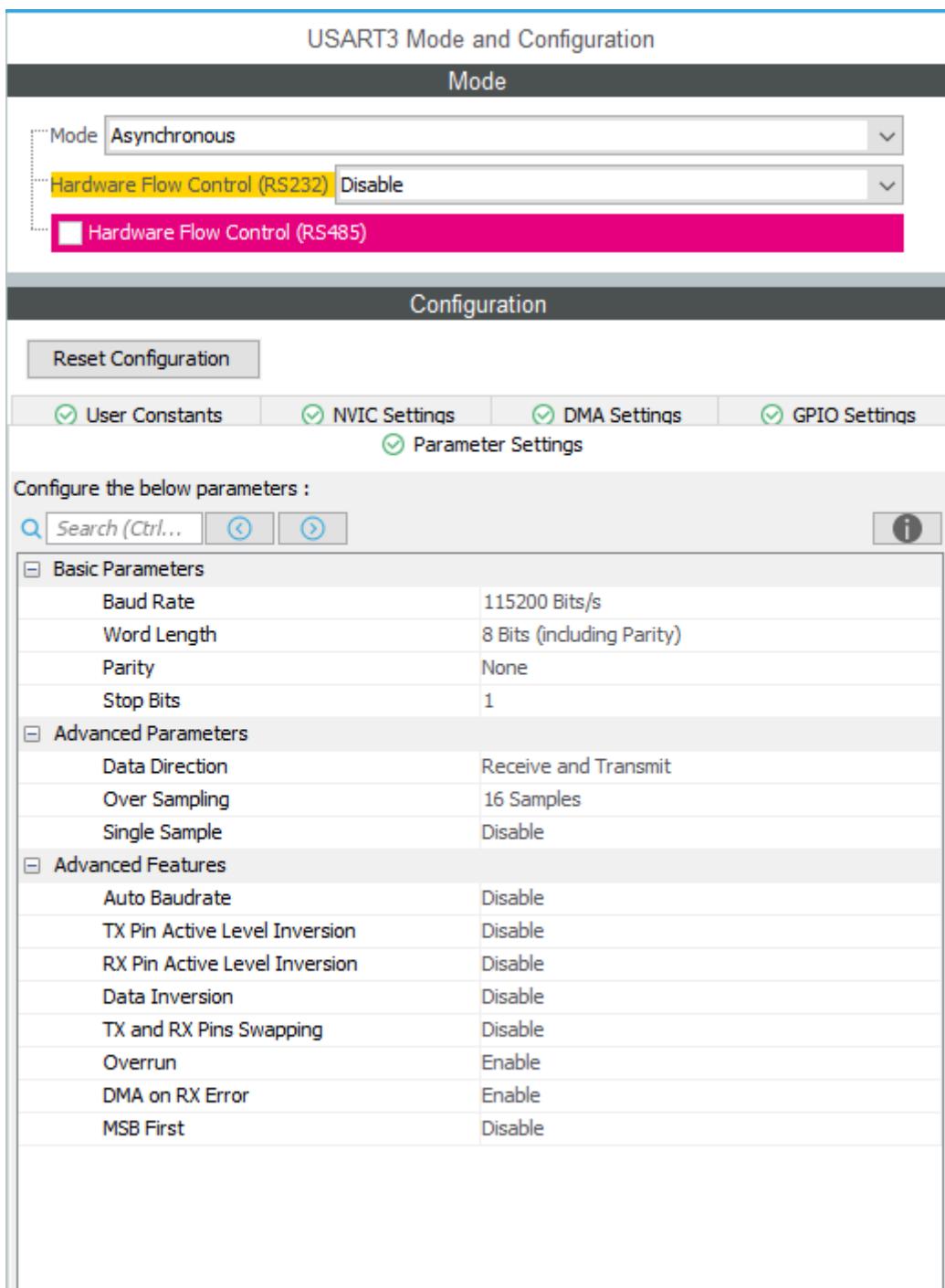
$$v_{\text{out}}(nT) = \beta v_{\text{out}}((n-1)T) + (1-\beta)V_i$$

$$\beta = e^{-\omega_0 T}$$

Warto też zauważyć, że konieczne było wprowadzenie nasycenia sygnału sterującego w przypadku, z powodu ograniczenia jakie nakłada płytka nucleo na sterowanie wypełnieniem sygnału PWM dawanego na bazę tranzystora.

#### 4. Komunikacji szeregowa

W celu zrealizowania zadania zdecydowaliśmy się korzystać z komunikacji szeregowej USART3. Początkowo skonfigurowano ioc:



Zdjęcie 2: Konfiguracja ioc dla komunikacji szeregowej

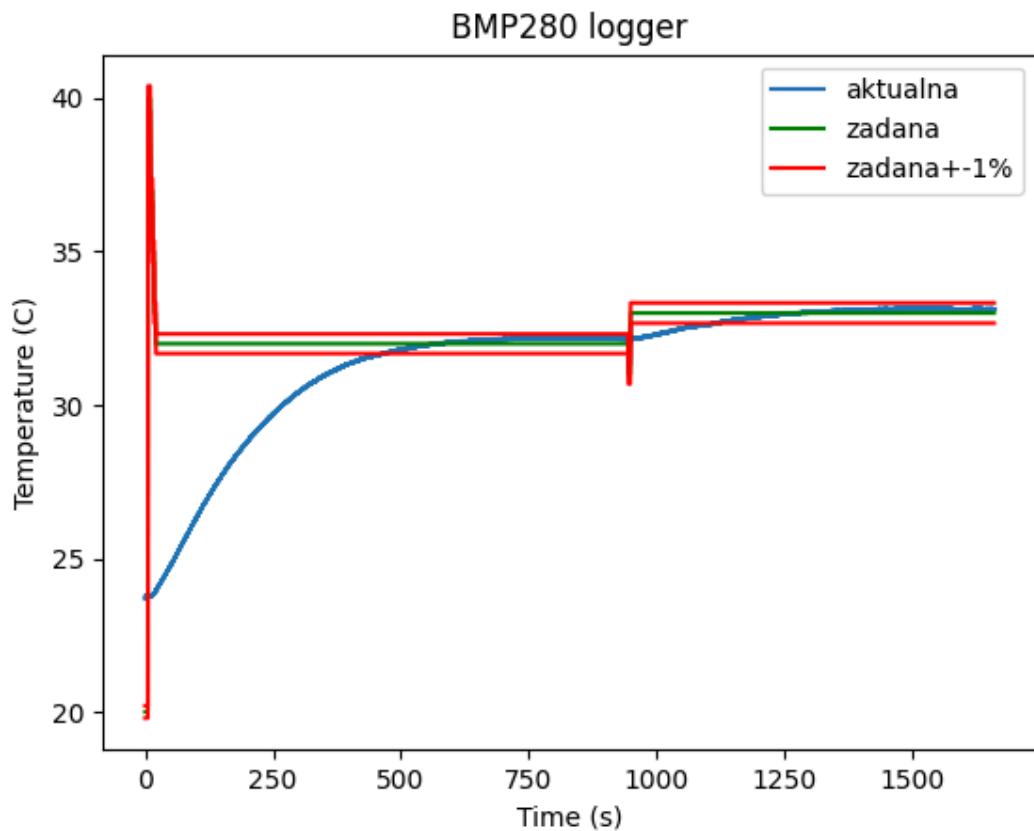
Następnie za pomocą kodu zainicjalizowano dwukierunkową komunikację szeregową:

- zadawanie wartości:

```
if (HAL_GPIO_ReadPin(GPIOC, USER_Btn_Pin) == 1 && startuart){  
    HAL_GPIO_TogglePin(GPIOB, LD3_Pin);  
    if (HAL_GPIO_ReadPin(GPIOB, LD3_Pin) == 1){  
        HAL_UART_Receive(&huart3,(uint8_t*)otrzymana, 8, 10000);  
        zadana=atof(otrzymana);  
    }  
    while(HAL_GPIO_ReadPin(GPIOC, USER_Btn_Pin) == 1){}  
}
```

## 5. Wizualizacja GUI - otrzymany uchyb

Wizualizację GUI przygotowano przy pomocy języka python. Aby umożliwić wizualizację należało “wysłać” informacje z płytki:

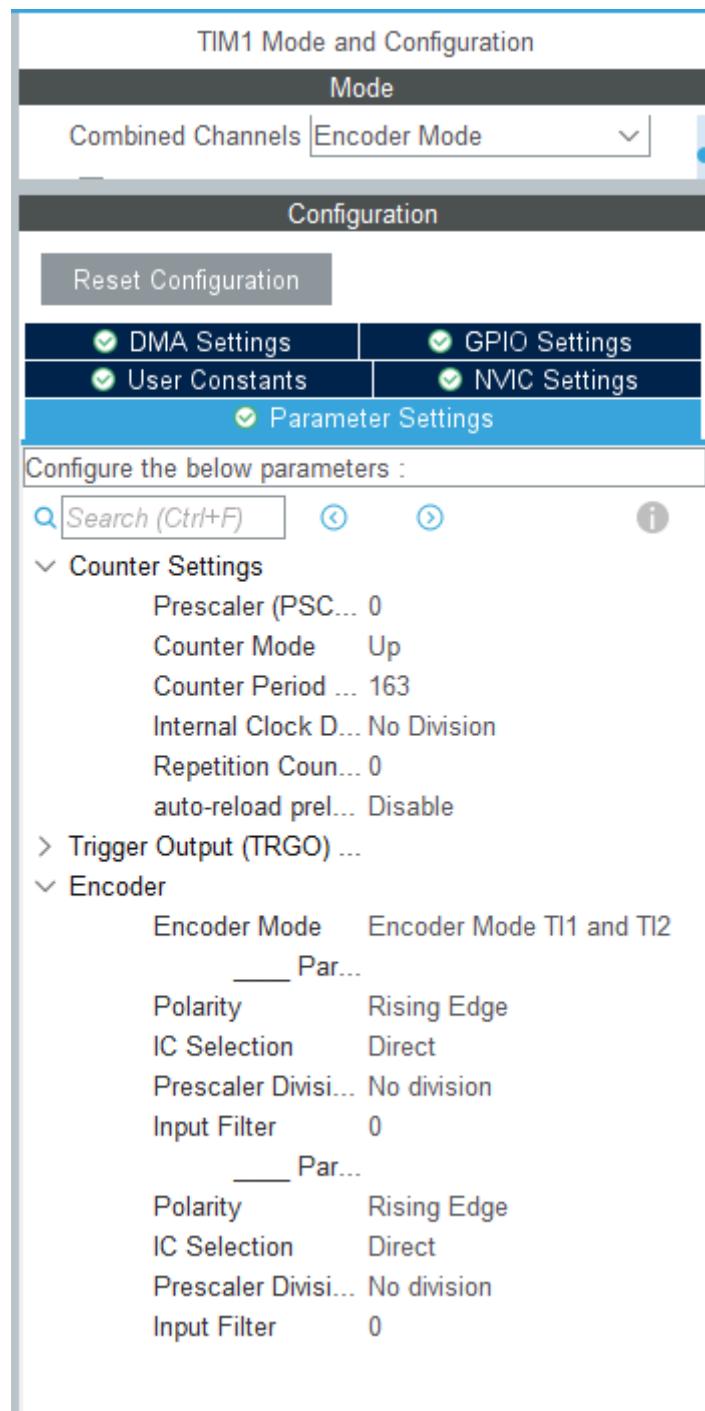


Wykres 2: Wizualizacja GUI z zaznaczonym 1% uchybem

Na wykresie widać, że uzyskany uchyb regulacji mieści się w 1%.

## 6. Implementacja enkodera

Konfiguracja ioc:



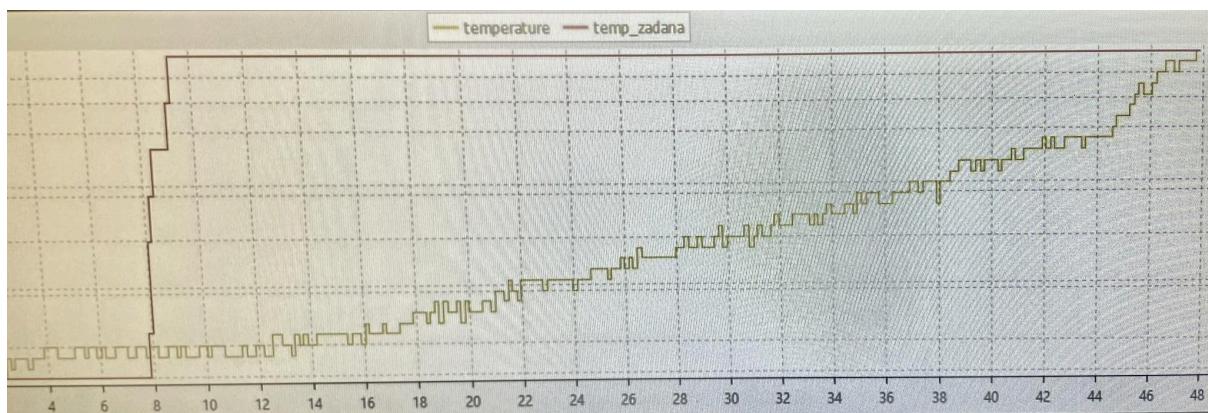
Zdjęcie 3: Konfiguracja ioc dla enkodera

Przygotowano kod, którego zadaniem jest przypisanie wartości ustawionej za pomocą enkodera jako wartość ustaloną:

```

/* USER CODE BEGIN 3 */
if(HAL_GPIO_ReadPin(GPIOB, LD3_Pin) == 0){
zadana=__HAL_TIM_GET_COUNTER(&htim1)/8+20;
}

```



Wykres 3: Grzanie rezystora po zmienieniu nastaw prz uzytci enkodera.

## 7. Implementacja wyświetlacza

Konfiguracja ioc:



Zdjęcie 4: Konfiguracja ioc dla wyświetlacza LCD

Przygotowano kod, którego zadaniem jest wyświetlenie wartości zadanej i odczytanej na ekranie. Dodatkowo ekran jest uruchamiany za pomocą przycisku USER:

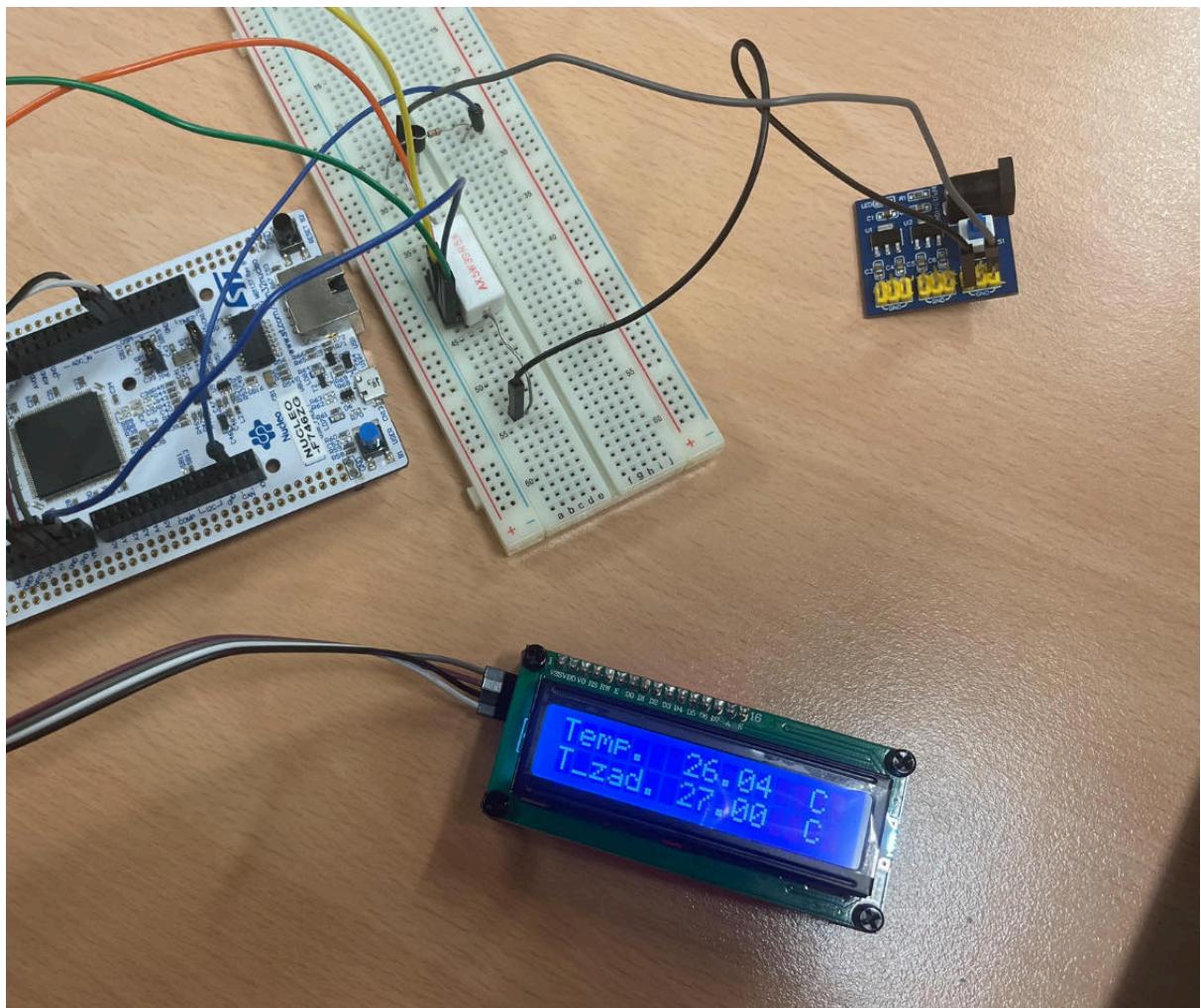
```

    while(1){HAL_UART_ReceiveItf(UART1, &rxn_rxn_rxn); -- +/\\}
}
if(HAL_GPIO_ReadPin(GPIOB, LD2_Pin) == 1){
    BMP280_ReadTemperatureAndPressure(&temperature, &pressure);
    lcd_clear ();
    lcd_put_cur(0, 0);
    sprintf((char*)text, "Temp. %.2f C", temperature);
    lcd_send_string(text);
    lcd_put_cur(1, 0);
    sprintf((char*)text, "T_zad. %.2f C", zadana);
    lcd_send_string(text);
}
/* USER CODE END 3 */

```

```
while( HAL_GetTick() - wait < 5000){  
    if (HAL_GPIO_ReadPin(GPIOC, USER.Btn_Pin) == 1){  
        HAL_GPIO_TogglePin(GPIOB, LD2_Pin);  
        lcd_init();  
        break;  
    }  
}  
HAL_GPIO_TogglePin(GPIOB, LD1_Pin);
```

Niestety ze względu na brak wyświetlacza w domu, nie udało się korzystać z niego w trakcie nagrania. Przygotowany kod wykorzystano jednak w trakcie laboratoriów dodatkowych).



Zdjęcie 5: Wyświetlana temperatura zadana i odczytana.

## 8. Kontrola wersji

Projekt realizowano z wykorzystaniem kontroli wersji github.

Link do repozytorium: <https://github.com/juliawalowska/Regulator-PID-ogrzewanie>