Politechnika Poznańska

Wydział Automatyki, Robotyki i Elektrotechniki

Instytut Robotyki i Inteligencji Maszynowej

Zakład Sterowania i Elektroniki Przemysłowej

UKŁAD REGULACJI TEMPERATURY Z WYKORZYSTANIEM CZUJNIKA MCP9808

Systemy mikroprocesorowe

Raport z realizacji projektu

Tomasz Kowalewski, 147507

tomasz.kowalewski@student.put.poznan.pl

Michał Olszowy, 147513

michal.olszowy@student.put.poznan.pl

Prowadzący:

mgr inż. Adrian Wójcik

Adrian.Wojcik@put.poznan.pl

04-01-2023

Spis treści

1 Wstęp **3**

2 Opis układu regulacji **4**

3 Model układu w środowisku Matlab/Simulink **4**

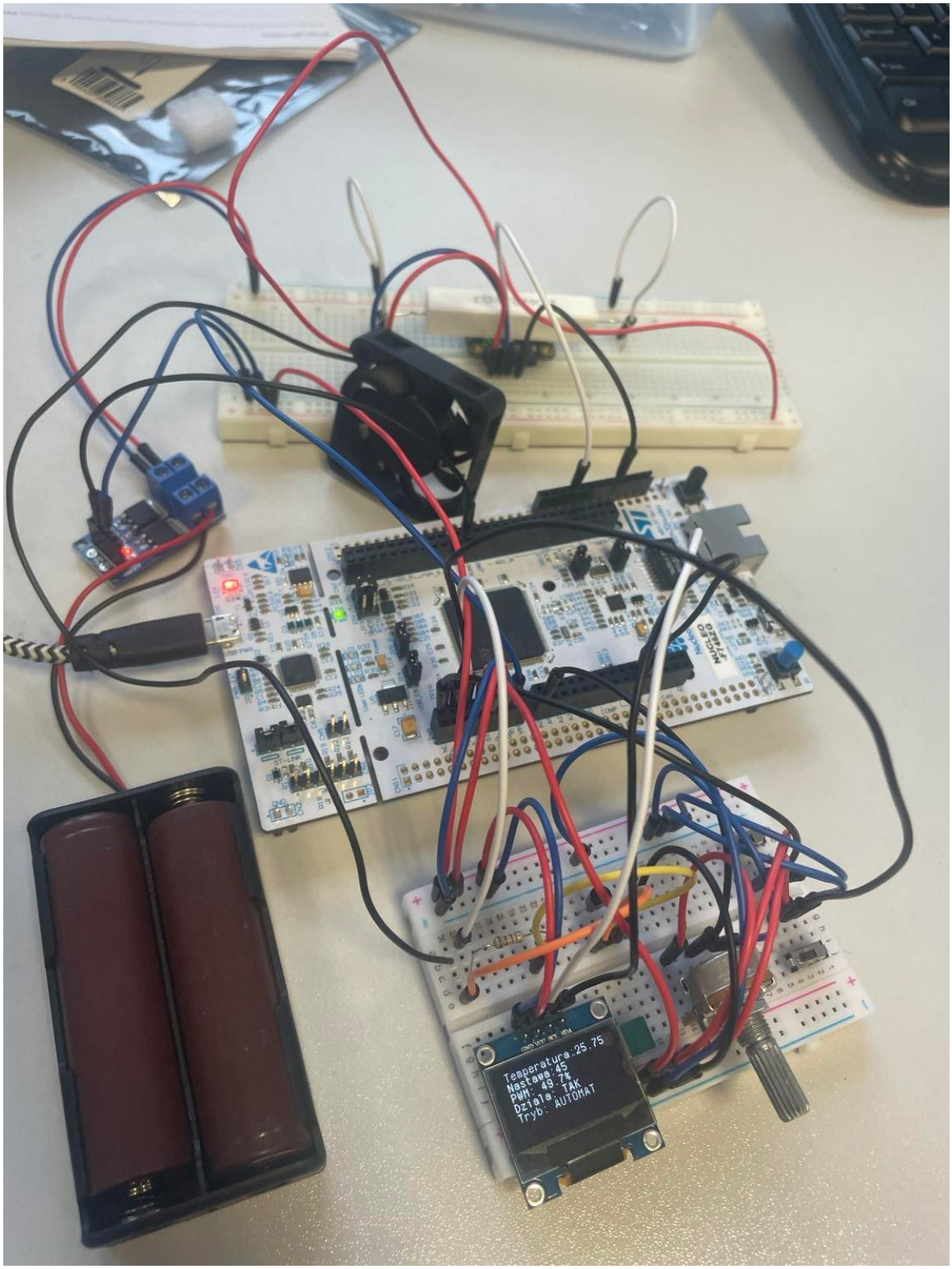
4 Układ rzeczywisty **6**

5 Implementacja **6**

6 Testy układu rzeczywistego z wykorzystaniem GUI **8**

Wstęp

Niniejszy raport jest dokumentacją projektu zaliczeniowego zajęć laboratoryjnych z przedmiotu Systemy mikroprocesorowe w roku akademickim 2022/2023. W ramach projektu został zrealizowany układ regulacji temperatury opartego na zestawie uruchomieniowym NUCLEO-F746ZG.



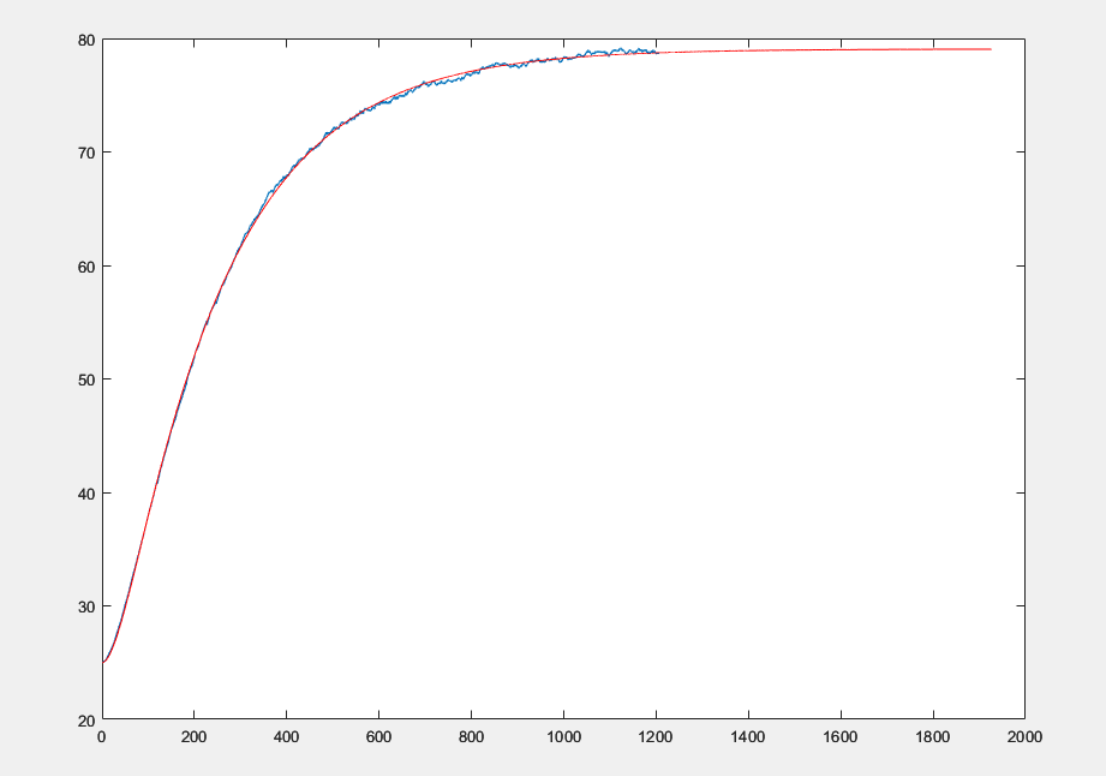
Rys. 1. Zdjęcie przedstawiające zmontowany układ

Opis układu regulacji

Układ regulacji, który jest przedmiotem tejże dokumentacji umożliwia użytkownikowi regulację temperatury na rezystorze grzejnym. Układ został zaprojektowany w taki sposób aby możliwe było zadawanie temperatury poprzez potencjometr (tryb MANUAL), jak również przez dedykowany interfejs graficzny (tryb AUTO). Podgląd wartości zadanej temperatury, jak również odczyt temperatury aktualnej jest możliwy za pośrednictwem GUI, jak również za pośrednictwem wyświetlacza LCD. Do komunikacji między STM32, a czujnikiem temperatury wykorzystano magistralę I2C. Za sterowanie układem odpowiada regulator PI, który reguluje wartość sygnału sterującego (PWM) w zakresie 0-50% (taki zakres został dobrany celowo, aby moc średnia na rezystorze oscylowała maksymalnie w okolicach jego mocy znamionowej). Za zasilanie układu odpowiadały 2 akumulatory litowo-jonowe połączone szeregowo o napięciu $U \approx 4.2V$ każdy. Do chłodzenia układu został zastosowany wentylator. Układ przystosowano do pracy w zakresie temperatur od 0 do 50°C.

Model układu w środowisku Matlab/Simulink

Aby wyznaczyć model matematyczny obiektu regulacji, skorzystano ze środowiska Matlab. W tym celu zebrana została odpowiedź skokowa na stałą wartość sygnału sterującego - poziomu wypełnienia PWM. Następnie odpowiedzi i sygnał sterujący wprowadzono do oprogramowania Matlab. Korzystając z funkcji tftest() uzyskano następującą aproksymacje obiektu.

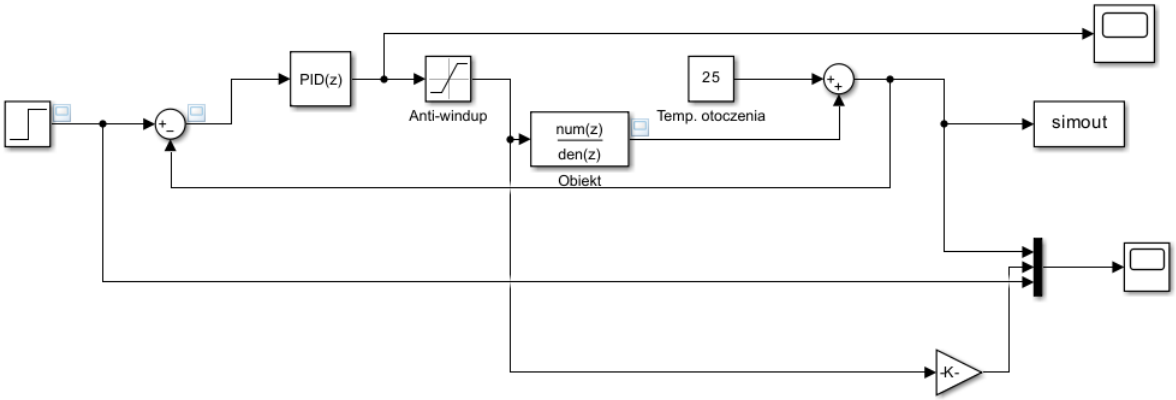


Rys. 2. Odpowiedź skokowa układu rzeczywistego (niebieski) oraz dobrany model matematyczny (czerwony)

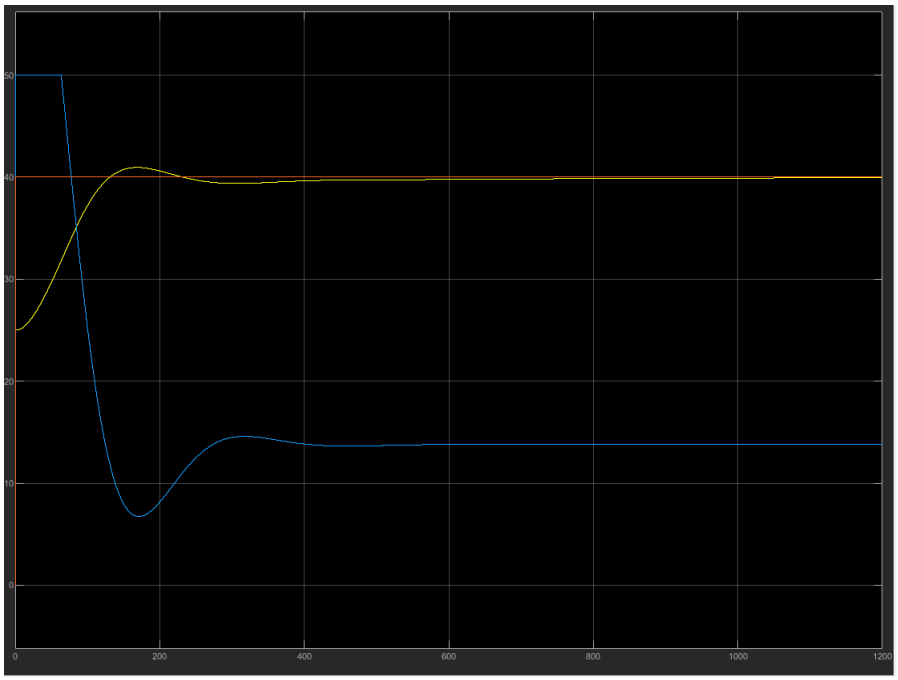
W rezultacie model przybliżono transmitancją dyskretną:

$$G(z) = \frac{2.039e-07}{1 - 1.331z^{-1} - 0.3332z^{-2} + 0.6641z^{-3}} \tag{1}$$

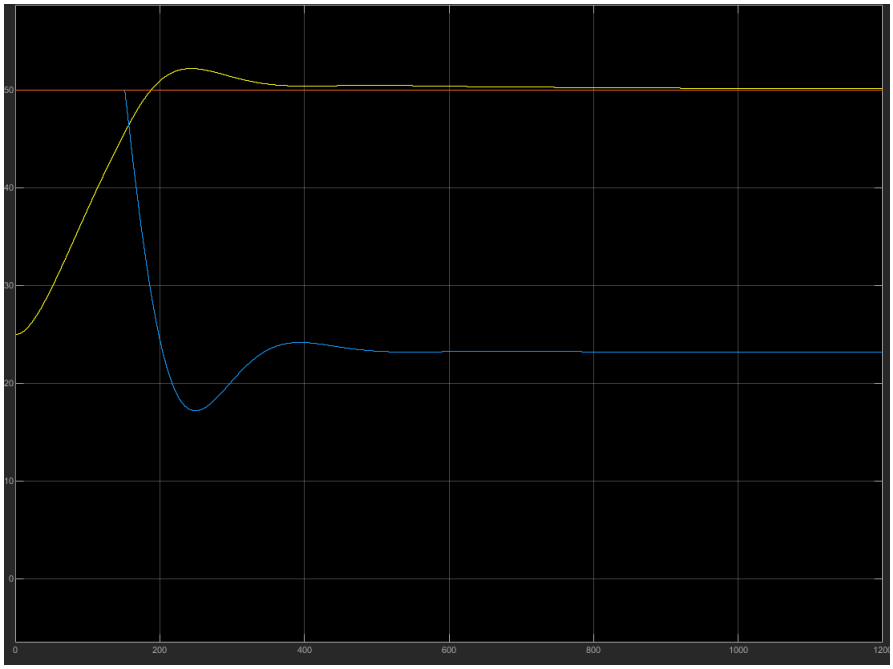
Kolejnym etapem było stworzenie modelu w środowisku Simulink oraz dobranie odpowiedniego regulatora. Ponieważ projektowany model charakteryzuje się dużą inercją, a zakłócenia zewnętrzne nie mają dużej dynamiki zmian, zdecydowano się na użycie regulatora PI.



Rys. 3. Model układu regulacji w środowisku Simulink



Rys. 4. Odpowiedź układu na zadaną temperaturę 40 °C



Rys. 5. Odpowiedź układu na zadaną temperaturę 50 °C

Układ rzeczywisty

Tab. 1. Spis elementów wykorzystanych w układzie

Element	Szczególne parametry/model
wentylator	50x50mm 0.8W 5V
rezystor	10 W, 3.3Ω
wyświetlacz ze zintegrowanym sterownikiem SSD1306	0.96 OLED, I2C, U_{zas} = 3-5V
cyfrowy czujnik temperatury	MCP9808
tranzystor	BC547B
rezystor	5.1$k\Omega$, 0.25W
moduł PWM	HW-517
potencjometr	20$k\Omega$, 0.25W
przełącznik suwakowy	SS22T25
akumulatory 18650	LG HG2
płytka NUCLEO	STM32F746ZGT6
płytki stykowe	-
przewody	-

Implementacja

Poniżej znajdują się listingi za pomocą których zaimplementowano najważniejsze funkcje. Repozytorium całego projektu dostępne jest w serwisie Github.

Listing 1. fragment kodu odpowiadający za wyliczanie i wystawianie sygnału sterującego

```
01.     if(htim->Instance == TIM2){ // If the interrupt is from timer 2 - 10Hz
02.                 current_duty_cycle = sterowanie;
03.                 transmit_data(akutalna_temperatura);
```

```
04.                 aktualny_blad = (zadana_temperatura-akutalna_temperatura);
05.                 if(grzanie_on_off()){
06.                         //sterowanie = 500;
07.                         sterowanie = round(arm_pid_f32(&pid, aktualny_blad));
08.                         if(sterowanie>500){
09.                                 sterowanie = 500;
10.                                 HAL_GPIO_WritePin(wentylator_GPIO_Port,
                                        wentylator_Pin, GPIO_PIN_RESET);
11.                         }
12.                         else if(sterowanie>0&&aktualny_blad>0){
13.                                 HAL_GPIO_WritePin(wentylator_GPIO_Port,
                                        wentylator_Pin, GPIO_PIN_RESET);
14.                         }
15.                         else if(sterowanie==0){
16.                                 HAL_GPIO_WritePin(wentylator_GPIO_Port,
                                        wentylator_Pin, GPIO_PIN_SET);
17.                         }
18.                         ograniczenie_sygnału_cmsis(&pid);
19.                         if(HAL_GPIO_ReadPin(wentylator_GPIO_Port, wentylator_Pin)
                                ==GPIO_PIN_SET){
20.                             sterowanie = 0;
21.                         }
22.                 }
23.                 else{
24.                         arm_pid_reset_f32(&pid);
25.                         nastawy_pid_cmsis(&pid);
26.                         sterowanie = 0;
27.                         HAL_GPIO_WritePin(wentylator_GPIO_Port, wentylator_Pin,
                                GPIO_PIN_SET);
28.                 }
29.                 change_current_duty_cycle(&htim1, TIM_CHANNEL_1, sterowanie);
```

Listing 2. funkcja ograniczająca Windup całkowania

```
01. void ograniczenie_sygnału_cmsis(arm_pid_instance_f32 * S){
02.         if(S->state[2]>500){
03.                 S->state[2] = 500;
04.         }
05.         if(S->state[2]<0){
06.             S->state[2] = 0;
07.         }
08. }
```

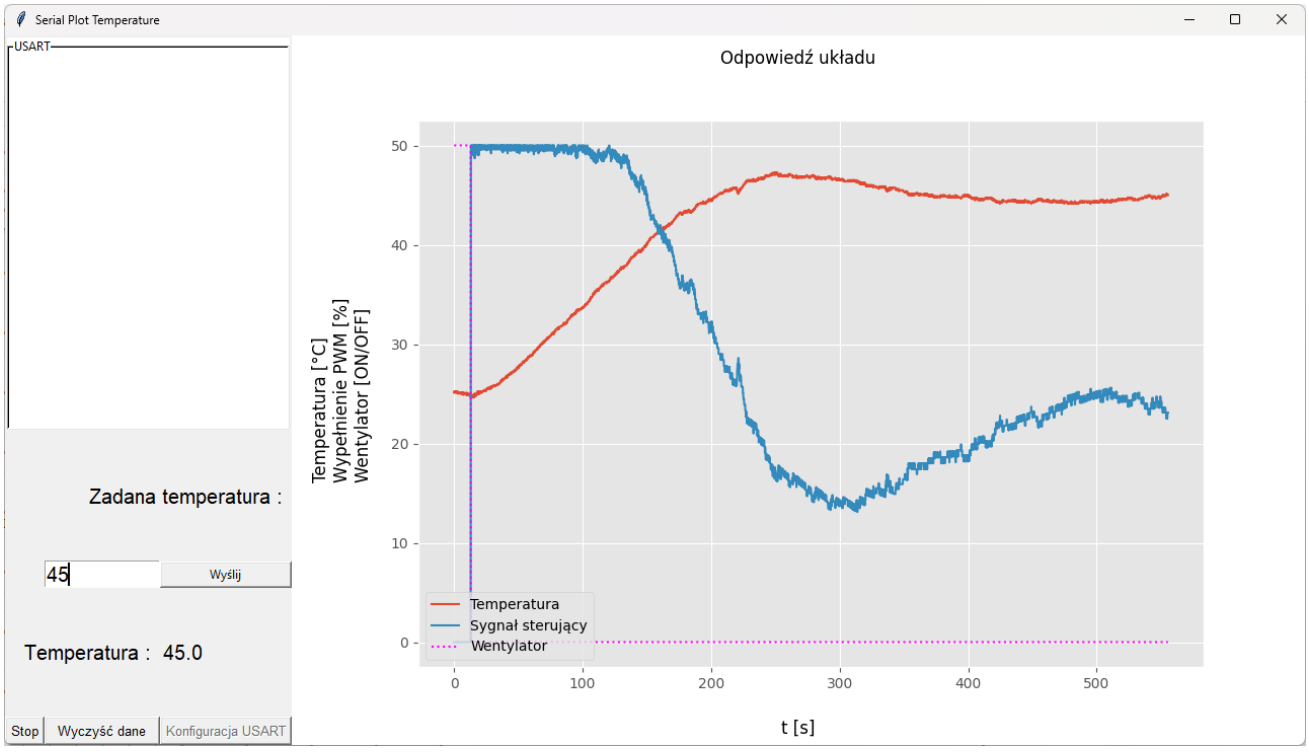
Listing 3. funkcja wysylająca dane do portu szeregowego

```
01. void transmit_data(float current_temp){
02.     char data_buf[100];
03.     gcvt(current_temp, 6, data_buf); // convertuje float na string
04.     strcat(data_buf, ";"); // dodaje srednik
05.     gcvt(sterowanie/1.0f, 6, data_buf+strlen(data_buf));
06.     if(HAL_GPIO_ReadPin(wentylator_GPIO_Port, wentylator_Pin)==GPIO_PIN_RESET){
07.         strcat(data_buf, ";0.0");
08.     }
09.     else{
10.         strcat(data_buf, ";1.0");
11.     }
12.     strcat(data_buf, "\r\n");
13.     HAL_UART_Transmit(&huart3, data_buf, strlen(data_buf), 100);
14. }
```

Listing 4. funkcja odbierająca i przetwarzająca dane z portu szeregowego

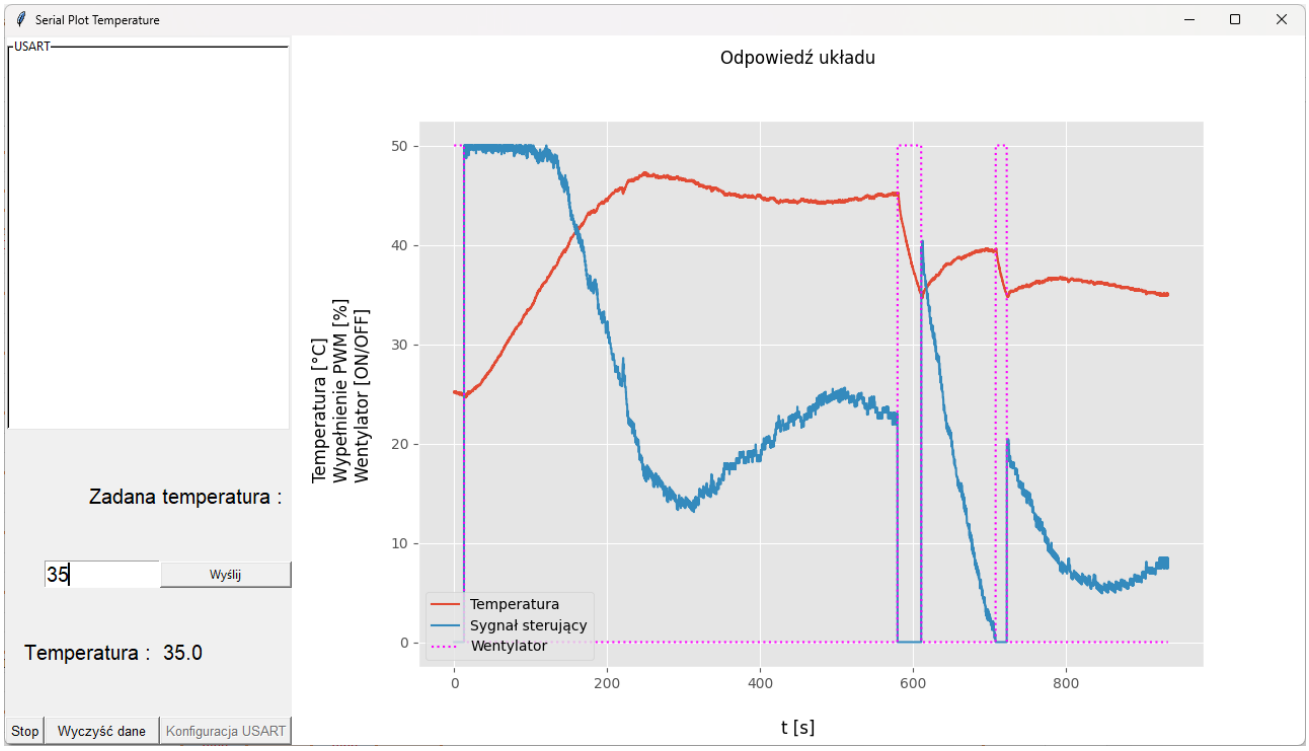
```
01. void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
02.         //HAL_UART_Receive_IT(&huart3, received_data, 3); // Tu włącza sie to
                gowno znowu :)
03.         if(auto_manual_on_off()){
04.                 if(atof(received_data)>50){
05.                         zadana_temperatura = 50;
06.                 }
07.                 else if(atof(received_data)<20){
08.                         zadana_temperatura = 20;
09.                 }
10.                 else{
11.                         zadana_temperatura = atof(received_data);
12.                 }
13.                 arm_pid_reset_f32(&pid);
14.                 nastawy_pid_cmsis(&pid);
15.         }
16. }
```

Testy układu rzeczywistego z wykorzystaniem GUI



Rys. 6. Zrzut ekranu z GUI, obrazujący przebieg sygnału sterującego i wyjściowego dla temperatury zadanej 45°C

Układ został przetestowany dla temperatury zadanej 45 °C. Po upływie niespełna 10 minut, temperatura osiąga stan ustalony przy uchybie równym ok. 0%.



Rys. 7. Zrzut ekranu z GUI, obrazujący przebieg sygnału sterującego i wyjściowego dla temp 45°C, a następnie dla 35°C

Podczas kolejnego testu najpierw zadano temperaturę 45°C, a po jej osiągnięciu zadano temperaturę 35 °C. Aby zredukować temperaturę, układ załączył wentylator, co spowodowało szybsze wychłodzenie rezystora. Niestety, ponieważ układ jest oddzielony od czujnika warstwą powietrza, a co za tym idzie - jego inercja jest dość duża, wystąpiły pewne przeregulowania w odpowiedzi. Jednakże, po upływie ok. 3 minut uchyb ustalony jest równy w przybliżeniu 0 %.