DOKUMENTACJA PROJEKTU		Rok akademicki 2024/25
Przedmiot: **SYSTEMY MIKROPROCESOROWE**		
Temat projektu: **REGULATOR TEMPERATURY PID**		Termin zajęć: **czwartek 16:50 – 18:20**
Wydział, kierunek, semestr, grupa: **WARiE, AiR, 5, A3-L5**	Imię i Nazwisko: 1. **Artsiom Kruk** 2. **Mateusz Krawczyński**	Punkty:
Data wykonania projektu: **22.01.2025**		

Analiza modelu regulatora:

W projekcie został stworzony dyskretny regulator PID.

```
typedef struct{
	float32_t Kp;
	float32_t Ki;
	float32_t Kd;
	float32_t dt;
}pid_parameters_t; //struktura parametrow PID

typedef struct{
	pid_parameters_t p;
	float32_t previous_error, previous_integral;
}pid_t; //struktura obiektu PID
```

```
//Tworzenie obiektu PID
pid_t pid1 = {.p.Kp = 1.2, .p.Ki = 0.003, .p.Kd = 1, .p.dt = 1, .previous_error=0,
.previous_integral=0};
```

```
float32_t calculate_discrete_pid(pid_t* pid, float32_t setpoint, float32_t measured){
	float32_t u=0, P, I, D, error, integral, derivative;

	error = setpoint-measured; //uchyb


	P = pid->p.Kp * error; //blok proporcjonalny

	integral = pid->previous_integral + (error+pid->previous_error) ; //numeryczna
calka
	pid->previous_integral = integral;
	I = pid->p.Ki*integral*(pid->p.dt/2.0); //blok calkujacy


	derivative = (error - pid->previous_error)/pid->p.dt; //numeryczne rozniczkowanie
	pid->previous_error = error;
	D = pid->p.Kd*derivative; //blok rozniczkujacy


	u = P  + I + D; //sygnal sterujacy

	return u;
}
```

```
//Obliczenie wypelnienia PWM
        float pwm_duty_f = (999*calculate_discrete_pid(&pid1, temp_zadana, temperature));

        //Saturacja
      uint16_t pwm_duty = 0;
        if(pwm_duty_f<0) pwm_duty = 0;
        else if(pwm_duty_f>999.0) pwm_duty = 999;
        else pwm_duty = (uint16_t)pwm_duty_f;
```

Najpierw tworzymy strukturę parametrów regulatora: zawiera ona współczynniki kp, ki, kd oraz dt (czas próbkowania).
Zatem struktura parametrów wraz z wartościami uchybu i całki tworzy strukture regulatora.

Po tym tworzymy obiekt regulatora PID, argumentami którego są podawane współczynniki kp, ki, kd oraz czas próbkowania dt. Wartości parametrów zostały dobrane poprzez analizę zachowania obiektu regulacji dla różnych temperatur zadanych.

Funkcja „calculate_discrete_pid" oblicza sygnał sterujący regulatora PID. Argumentami funkcji są obiekt struktury pid_t, temperatura zadana oraz temperatura bieżąca. Wewnątrz tej funkcji są obliczane składowe sygnału sterującego u – P, I oraz D. Różniczkowanie i całkowanie są realizowane poprzez wzory numeryczne.

W pętli while(1) funkcja „calculate_discrete_pid" iteracyjnie oblicza sygnał sterujący, który potem jest mnożony razy 999 (maksymalna wartość wypełnienia PWM zadana projektowo). Otrzymana wartość pwm_duty_f podlega saturacji, w wyniku czego otrzymujemy końcową wartość pwm_duty.

Co zostało zrealizowane:

1. **Komunikacja szeregowa (dwukierunkowa)**

```c
/Funkcja wysylania danych
void SendDataOverUART(UART_HandleTypeDef *huart, float temperature, float temp_zadana) {
            uint32_t time_ms = HAL_GetTick();
            char text[50];
            sprintf(text, "%lu,%.2f,%.2f\n", time_ms, temperature, temp_zadana);
            HAL_UART_Transmit(huart, (uint8_t *)text, strlen(text), 1000);
        }
```

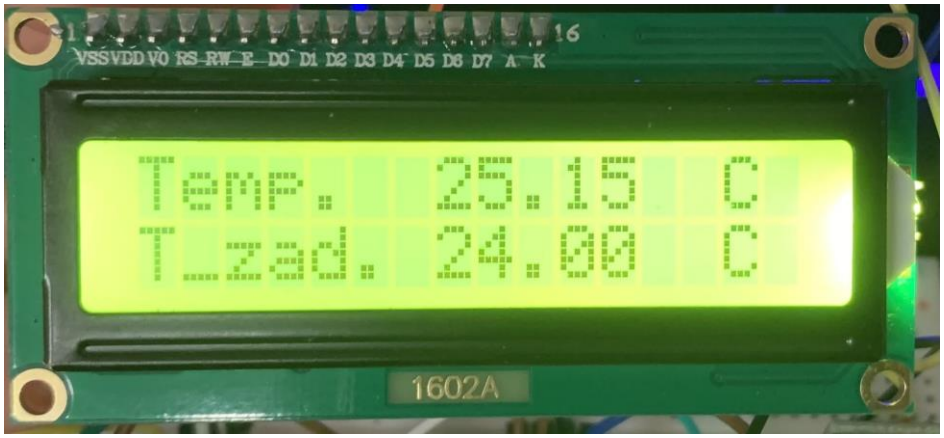
```c
//Funkcja przerwania dla odbioru danych
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {

    //Konwersja z ASCII do integera
      first = command[0] - '0';
      second = command[1] - '0';
      decimal = command[3] - '0';

      //Obliczenie temperatury zadanej
      temp_zadana = first*10+second+0.1*decimal;


      pid1.previous_integral=0; //Zerowanie calki przy zmianie wart. zadanej
      HAL_UART_Receive_IT(&huart3, (uint8_t *)command, sizeof(command));
}
```

Do portu szeregowego za pomocą funkcji SendDataOverUART co sekundę są wysyłane czas próbki, próbka temperatury i wartość temperatury zadanej.

Odczyt temperatury zadanej jest zrealizowany przez przerwania. W funkcji HAL_UART_RxCpltCallback jest prowadzana konwersja z ASCII do integera, a potem obliczona nowa wartość temperatury zadanej. Zerowanie całki jest potrzebne dla zmiejszenia jej wpływu na odpowiedź obiektu przy skokowej zmianie temperatury zadanej.

2. **LCD**

Element dodatkowy wyjścia LCD podłączony za pomocą interfejsu I2C.
Jest używany do wyświetlania temperatury bieżącej i zadanej.



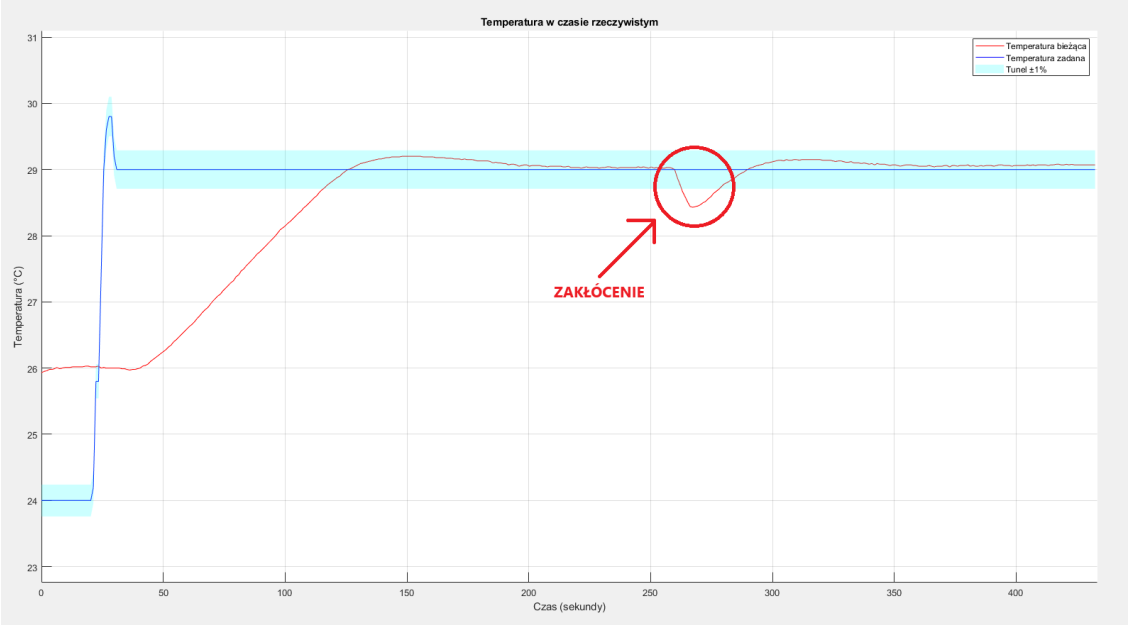
3. Enkoder

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
	int i = __HAL_TIM_GET_COUNTER(&htim3);
	temp_zadana = 24+i*0.1;
	pid1.previous_integral=0; //Zerowanie calki przy zmianie temperatury zadanej
}
```

Dodatkowym elementem wejścia jest enkoder inkrementalny. Jest podłączony do kanałów 1 i 2 TIM3. Odczyt liczby impulsów jest zrealizowany przez funkcję obsługi przerwań. Nowa temperatura zadana jest obliczana na podstawie wzoru $t = 24 + impulsy * 0.1$ Counter Period TIM3 jest ustawiony na 80. W wyniku mamy możliwość ustawiania temperatury zadanej w przedziale od 24 do 32 stopni celsiusza.

4. Wizualizacja graficzna

Wizualizacja graficzna została zrealizowana za pomocą MatLab.
Skrypt odczytuje dane przesyłane przez port szeregowy (czas, temperatura bieżąca oraz zadana) i wykreśla ich na bieżąco.



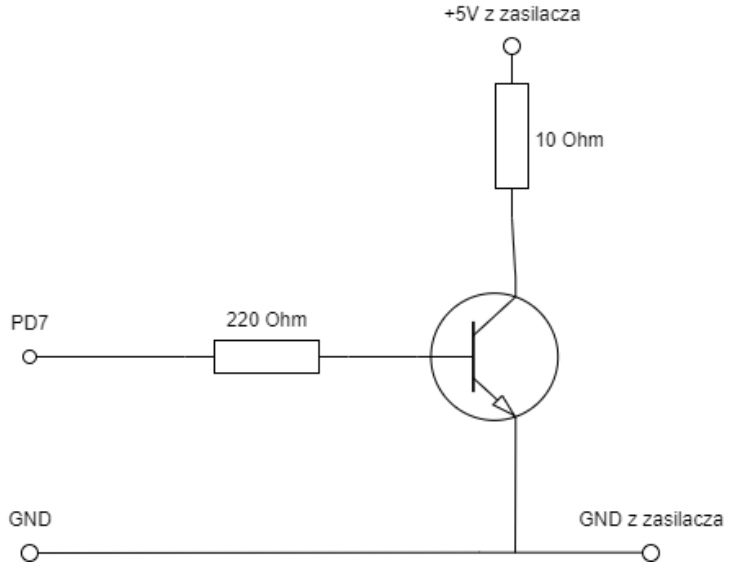
Wykres 1 (przebieg odpowiedzi obiektu w układzie zamkniętym)

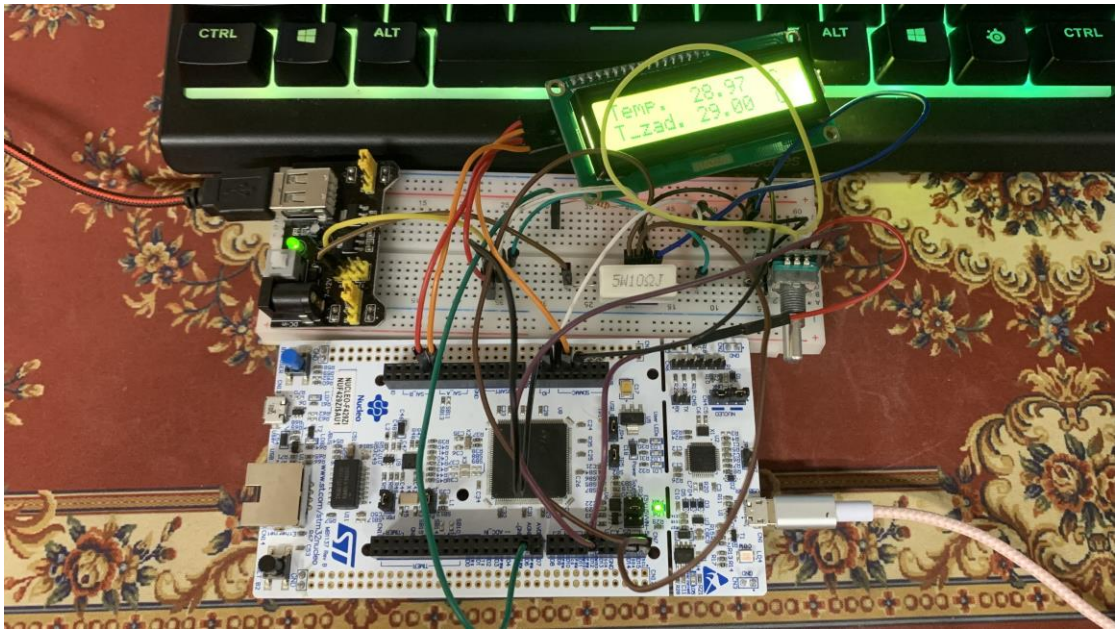
Fizyczny układ

Użyte elementy:

- STM32 NUCLEO-F429ZI
- Transistor 2N2222A
- Resistor 220Ohm
- Resistor ceramiczny 10Ohm 5W
- Wyświetlacz LCD
- Enkoder inkrementalny
- Zasilacz
- Czujnik temperatury i cisnienia BMP280

Fizyczny układ został zrealizowany zgodnie ze schematem:





Wyniki

W wyniku pracy został zrealizowany regulator temperatury PID. Jak widać z wykresu 1 uchyb ustalony jest w przedziale 1% wartości zadanej, co swiadczy o poprawnym działaniu regulatora i wysokiej jakości regulacji. Przeregulowanie jest około 0.5 stopni celsiusza i nie jest niebezpieczne. Czas regulacji do tunelu 1% wynosi około 80 sekund (dla różnych wartości zadanych czas regulacji może różnić się). Z wykresu widać, że układ jest odporny na zakłocenia i eleminuje uchyb prawie do zera.
Również zostały zrealizowane dodatkowe elementy wejścia i wyjścia, które umożliwiają prostszą kontrolę wartości zadanej oraz wyświetlanie odpowiedzi układu na bieżąco. Komunikacja szeregowa za pomocą UART umożliwia wygodniejsze sterowanie i wykreślanie wyników za pomocą oprogramowania MatLab.