

Systemy mikroprocesorowe

SPRAWOZDANIE – REGULATOR TEMPERATURY PID

REGULATOR TEMPERATURY PID

Naskręt ,Naumienko

1 Cel projektu

Celem projektu było zaprojektowanie i zrealizowanie mikroprocesorowego układu regulacji temperatury w zamkniętej pętli sprzężenia zwrotnego. Projekt zakładał wykorzystanie mikrokontrolera z rodziny STM32, cyfrowego czujnika temperatury oraz algorytmu PID do sterowania elementem grzejnym. Dodatkowymi wymaganiem była realizacja interfejsu użytkownika (wyświetlacz LCD, enkoder) oraz dwukierunkowej komunikacji szeregowej (UART) do celów telemetrycznych.

2 Wykorzystane elementy i konfiguracja sprzętowa

2.1 Elementy składowe

W projekcie wykorzystano następujące podzespoły:

- **Mikrokontroler:** STM32 Nucleo.
- **Czujnik pomiarowy:** BMP280 (Interfejs I2C) – pomiar temperatury otoczenia.
- **Element wykonawczy:** Rezystor mocy sterowany tranzystorem (Grzałka).
- **Interfejs HMI:** Wyświetlacz LCD 2x16 (sterownik I2C HD44780) oraz enkoder inkrementalny.

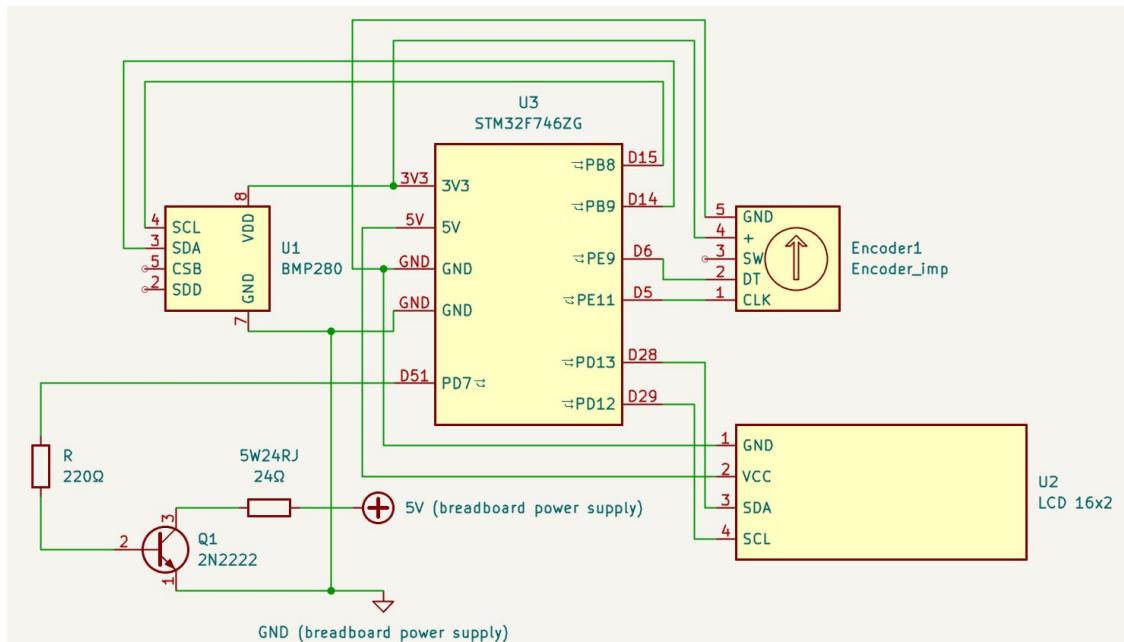
2.2 Konfiguracja Pinout

Tabela ?? przedstawia przypisanie wyprowadze  mikrokontrolera do poszczególnych peryferiów.

Tablica 1: Konfiguracja wyprowadze  mikrokontrolera

Peryferium	Pin / Port	Funkcja
Grza�ka	PD7	Wyj�cie steruj�ce (On-Off / PWM programowy)
BMP280	I2C1 (PB8, PB9)	Komunikacja z czujnikiem temperatury
LCD 2x16	I2C4	Obs�uga wy�wietlacza
Enkoder	TIM1	Ustawianie temperatury zadanej
UART	USART3	Komunikacja z PC (115200 baud)
LED Status	PB0 (LD1)	Sygnalizacja grzania

3 Schemat elektroniczny uk adu pomiarowego



Rysunek 1: Schemat badanego uk adu

4 Wykorzystane s rodowisko programistyczne

4.1 STM32 CubeIDE

STM32 CubeIDE to darmowe s rodowisko programistyczne firmy STMicroelectronics przeznaczone do tworzenia aplikacji na mikrokontrolery STM32.  aci y w jednym narz dziu edytor kodu, kompilator, debugger oraz modu  STM32CubeMX, kt ry umo wi a graficzn  konfiguracj  pinów, zegarów i peryferi w mikrokontrolera. S rodowisko bazuje na platformie pozwala na programowanie w j zyku C oraz C++, z wykorzystaniem bibliotek HAL i LL. Dzi ki wbudowanemu debuggerowi, mamy dost y do monitoringu stanu i zmiany danych w czasie rzeczywistym.

dowanej obsłudze programatora i debuggera ST-LINK ułatwia uruchamianie, testowanie i analizę działania aplikacji na sprzęcie.

4.2 Visual Studio Code

Visual Studio Code to darmowe środowisko programistyczne stworzone przez firmę Microsoft. Jest to edytor kodu z rozbudowanymi funkcjami i wieloma rozszerzeniami. Obsługuje wiele języków programowania, w tym użyty do stworzenia charakterystyki regulacji ?? Python. VS Code oferuje m.in. podświetlanie składni, inteligentne podpowiedzi, debugowanie oraz integrację z systemami kontroli wersji, takimi jak Git. Głównymi zaletami tego środowiska jest jego prostota i elastyczność.

5 Algorytm sterowania (PID)

W projekcie zaimplementowano dyskretny regulator PID. Wyjście regulatora steruje czasem włączenia grzałki w cyklu 1-sekundowym (Time Proportional Control).

Równanie regulatora w dziedzinie czasu ciągłego:

$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \quad (1)$$

gdzie: $e(t)$ to uchyb regulacji (różnica między wartością zadaną a mierzoną).

5.1 Implementacja programowa

Poniżej przedstawiono kluczowy fragment kodu odpowiedzialny za obliczanie nastaw regulatora. Zastosowano zabezpieczenie przed nasyceniem członu całkującego (Anti-Windup Clamp).

Listing 1: Struktura i funkcja obliczająca PID

```
typedef struct {
    float Kp;
    float Ki;
    float Kd;
    float prevError;
    float integral;
    float outMin;
    float outMax;
} PID_Controller;

PID_Controller pid = {
    .Kp = 400.0f,
    .Ki = 10.0f,
    .Kd = 50.0f,
    .prevError = 0.0f,
    .integral = 0.0f,
    .outMin = 0.0f,
    .outMax = 1000.0f
```

```

};

float PID_Compute( PID_Controller *pid , float setpoint , float measured)
{
    float error = setpoint - measured;

    // Cz on Proporcjonalny
    float P = pid->Kp * error;

    // Cz on Ca kuj cy
    pid->integral += error;
    if (pid->integral > pid->outMax) pid->integral = pid->outMax;
    else if (pid->integral < pid->outMin)
        pid->integral = pid->outMin;

    float I = pid->Ki * pid->integral;

    // Cz on R  niczkuj cy
    float D = pid->Kd * (error - pid->prevError);
    pid->prevError = error;

    // Suma
    float output = P + I + D;

    // Ograniczenie wyj cia (Clamp)
    if (output > pid->outMax) output = pid->outMax;
    else if (output < pid->outMin) output = pid->outMin;

    return output;
}

```

6 Analiza modelu i dob r nastaw

Do wyznaczenia nastaw regulatora wykorzystano zmodyfikowan  **II Metod  Zieglera-Nicholsa** (metod  oscylacji krytycznych). Ze wzgl du na bezpieczeństwo uk adu, parametry estymowano na podstawie oscylacji gasn cych w pobli u punktu pracy, a nie doprowadzaj c uk ad do trwa ej niestabilno ci.

6.1 Identyfikacja parametrów dynamicznych

Na podstawie zarejestrowanej charakterystyki skokowej (Rys. ??) wyznaczono naturalny okres oscylacji uk adu.

- Czas wyst pienia pierwszego szczytu: $t_1 \approx 400\text{ s}$
- Czas wyst pienia drugiego szczytu: $t_2 \approx 1000\text{ s}$

- Okres oscylacji (T_{osc}): $1000\text{ s} - 400\text{ s} = 600\text{ s}$
- Wzmocnienie, przy którym wystąpiły oscylacje (K_p): ≈ 400

6.2 Obliczenia analityczne

Przyjmując $T_{osc} = 600\text{ s}$ jako bazę do obliczeń (zgodnie z regułami Z-N dla regulatora PID):

- **Czas zdwojenia (T_i)**: $0.5 \cdot T_{osc} = 300\text{ s}$.
- **Czas wyprzedzenia (T_d)**: $0.125 \cdot T_{osc} = 75\text{ s}$.

Przeliczenie na wzmocnienia dla algorytmu dyskretnego (przyjmując czas próbkowania $T_s = 1\text{ s}$):

$$K_i(\text{obl}) = \frac{K_p \cdot T_s}{T_i} = \frac{400 \cdot 1}{300} \approx 1.33 \quad (2)$$

$$K_d(\text{obl}) = \frac{K_p \cdot T_d}{T_s} = \frac{400 \cdot 75}{1} = 30000 \quad (3)$$

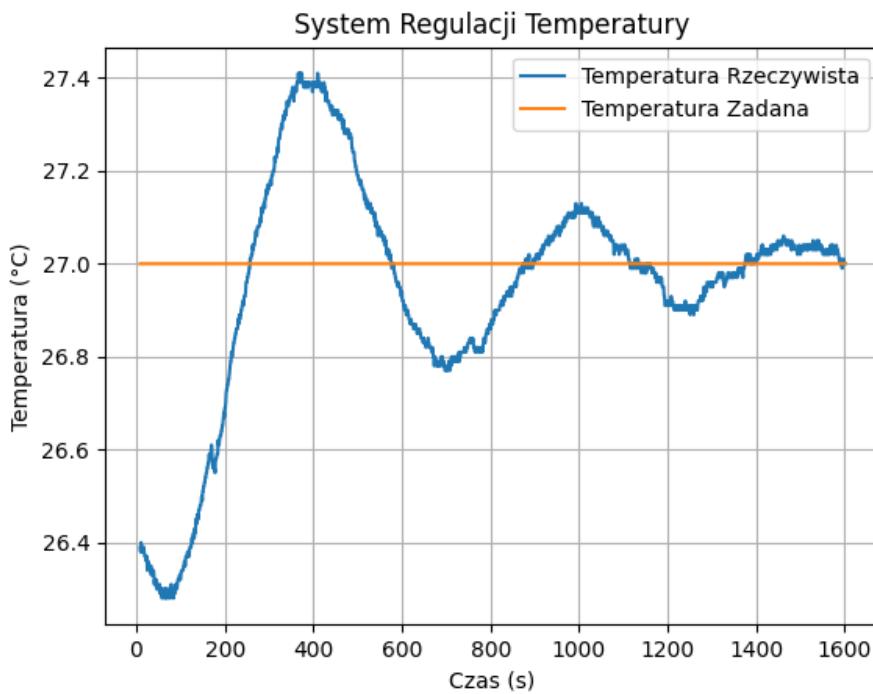
6.3 Korekta eksperimentalna (Fine-tuning)

Obliczone wartości teoretyczne wymagały korekty ze względu na specyfikę obiektu (du a bezw adno c cieplna) oraz charakter cyfrowy (szumy kwantyzacji):

1. **Zwiększenie K_i (do 10.0):** Teoretyczna warto c (1.33) powodowała zbyt wolną redukcję uchybu statycznego.
2. **Zmniejszenie K_d (do 50.0):** Teoretyczna warto c (30000) była zbyt wysoka dla implementacji dyskretnej, powodując silne wzmacnianie szumów pomiarowych czujnika BMP280, co prowadziło do niestabilnego sterowania grza k .

7 Wyniki i weryfikacja

Poni szy wykres przedstawia charakterystyk  regulacji dla zadanej temperatury 27.0°C .



Rysunek 2: Przebieg regulacji temperatury. Linia pomarańczowa - warto   zadana, linia niebieska - temperatura mierzona.

Wykres wskazuje na stabiln   pr  c   uk  du. Wyst  puj   oscylacje gasn  ce (przeregulowanie), co jest charakterystyczne dla przyj  tych nastaw, jednak uk  du skutecznie sprowadza temperatur   do warto  ci zadanej. Po ustabilizowaniu si   przebiegu, b  d regu  acji (uchyb) utrzymywa   si   na niskim poziomie wynosz  cym ok. 0.3%–0.5%.

8 Instrukcja obslugi i funkcjonalno  ci

1. Po uruchomieniu uk  du wyświetla na LCD aktualn   temperatur   oraz zadan   warto  ci (T_{zad}).
2. **Zmiana lokalna:** Obr  t enkoderem powoduje zmian   temperatury zadanej.
3. **Sterowanie zdalne (Terminal):** Pod  czenie terminala (115200 baud) umo  wiwa:
 - Odbi  r danych telemetrycznych np. w programie PuTTY (format: `temp_akt` `temp_zad`).
 - Zmian   nastaw komendami: + (zwi  ksz), - (zmniejsz).
4. **Wizualizacja w czasie rzeczywistym (SerialPlot):** Uk  ad przystosowany jest do współpracy z oprogramowaniem **SerialPlot** w celu rysowania wykresu temperatury na   wo. Konfiguracja programu:
 - **Port:** Wyb  r odpowiedniego portu COM (STLink Virtual COM Port).
 - **Baud Rate:** 115200.

- **Data Format:** ASCII.
- **Number of Channels:** 2 (Kanał 1: Temperatura mierzona, Kanał 2: Temperatura zadana).
- **Column Delimiter:** Space (spacja).

Program automatycznie interpretuje ramki danych wysy ane przez mikrokontroler i rysuje przebieg regulacji.

5. Dioda LED na płytce Nucleo sygnalizuje momenty za czenia elementu grzejnego.

9 Podsumowanie

Uk ad poprawnie odczytuje temperatur , a zaimplementowany algorytm PID skutecznie steruje procesem grzania. Zastosowanie metody Zieglera-Nicholsa pozwoli  na wst epne oszacowanie nastaw, a p o niejsza korekta eksperimentalna wyeliminowa a problemy zwi zane z szumami cyfrowymi oraz du  ą inercj  termiczn  obiektu. Dodatkowo zaimplementowano wizualizacj  danych w czasie rzeczywistym, co pozwala na bie  ac  analiz  jakosti regulacji.