Spis treści

1. Wprowadzenie

Poniższy raport dotyczy projektu zaliczeniowego na zajęcia laboratoryjne z przedmiotu *Systemy mikroprocesorowe*. Celem zasadniczym jest zaprojektowanie, zbudowanie i przetestowanie mikroprocesorowego systemu sterowania i pomiaru w oparciu o mikrokontroler z rodziny STM32 oraz elementy wykonawcze i pomiarowe umieszczone na płytce stykowej. Zadanie polega na stałowartościowej regulacji temperatury w oparciu o regulator PID z korekcją *anty-windup*. Obiektem regulacji jest rezystor ceramiczny (parametry podane w dalszej części raportu), którego temperatura zależy od przepływającego prądu. Przy użyciu czujnika temperatury realizujemy w układzie sprzężenie zwrotne, co jest niezbędne w regulacji automatycznej. Przed przystąpieniem do pracy z kompletnym układem regulacji automatycznej, analizie zostanie poddany model obiektu oraz regulatora. System umożliwia bieżący podgląd sygnału regulowanego oraz zadawanie wartości referencyjnej. W układzie zamontowano wentylator o odpowiedniej mocy, wspierający obniżanie temperatury całego obiektu.

Założenia projektowe systemu mikroprocesorowego:

- uchyb ustalony na poziomie 1% wartości zakresu regulacji,
- możliwość zadawania wartości referencyjnej za pomocą komunikacji szeregowej,
- możliwość zadawania wartości referencyjnej za pomocą potencjometru,
- możliwość podglądu aktualnej wartości mierzonej (port szeregowy),
- graficzna wizualizacja (real-time),
- model symulacyjny z uwzględnionym regulatorem,
- wyświetlacz LCD ukazujący bieżące informacje o sygnale referencyjnym i regulowanym,
- chłodzenie obiektu przy pomocy wentylatora,
- wykorzystanie systemu kontroli wersji (GitHub).

2. Struktura, komponenty, schemat układu

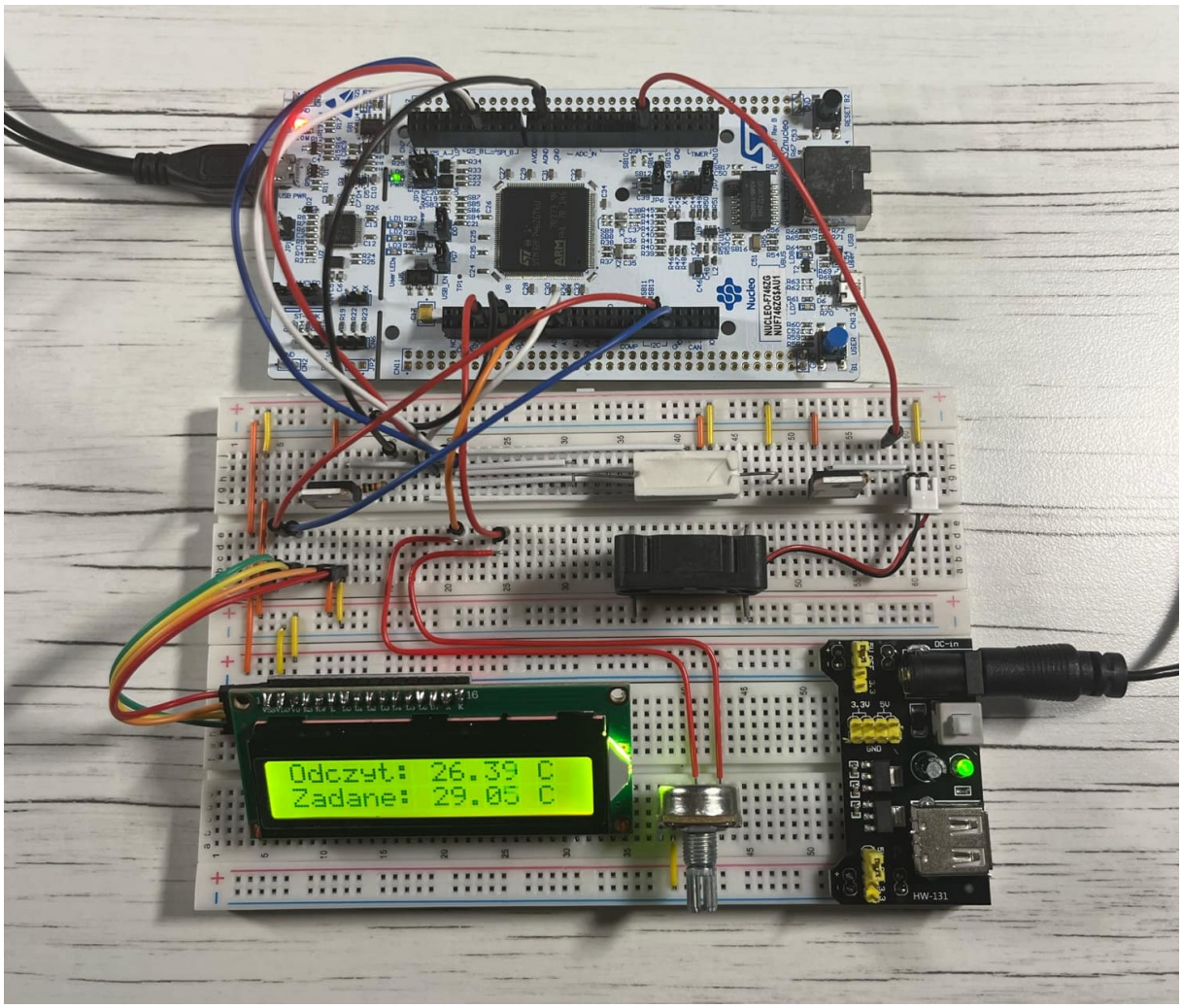
2.1. Spis elementów

W układzie systemu mikroprocesorowego wykorzystano poniższe części:

- płytka rozwojowa NUCLEO-F746ZG,
- rezystor ceramiczny 39Ω $5W$,
- tranzystor bipolarny NPN Darlington TIP122 (2 sztuki),
- czujnik ciśnienia i temperatury BMP280,
- wyświetlacz LCD 2x16 w połączeniu z konwerterem I^2C,
- potencjometr obrotowy $10k\Omega$ liniowy,
- moduł zasilający do płytek stykowych MB102 - $5V$,
- zasilacz impulsowy $12V/2,5A$,
- rezystor przewlekany $1k\Omega$ (2 sztuki),
- wentylator $5V$,
- płytka stykowa (2 sztuki),
- przewody połączeniowe.

Element wykonawczy stanowi tranzystor bipolarny NPN Darlington TIP122, który pełni rolę klucza, umożliwiając przepływ prądu z zasilacza impulsowego, przez rezystor, złącze kolektor-emiter do masy. To samo dotyczy układu z wentylatorem. W tego typu zastosowaniach tranzystor w momencie przewodzenia (stan wysoki na bazie, pochodzący z sygnału PWM - wystawiany przez mikrokontroler zgodnie z algorytmem regualcji PID) powinien znajdować się w stanie nasycenia w celu ograniczenia do minimum spadku napiecia na złączu kolektor-emiter tranzystora, co prowadzi do minimalizacj strat mocy wydzielanych na tranzystorze.

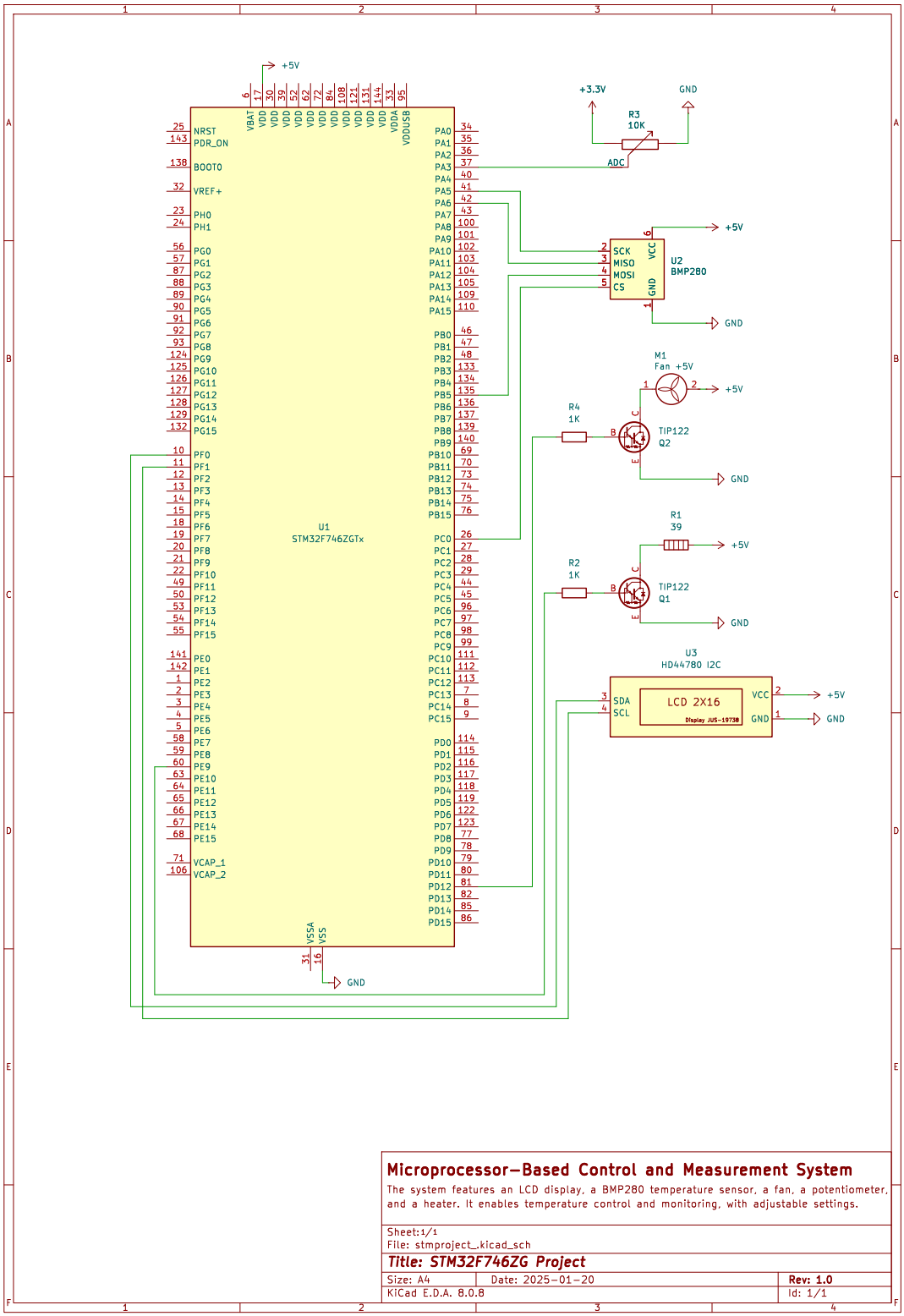
2.2. Układ rzeczywisty



Rysunek 2.1. Układ rzeczywisty systemu mikroprocesorowego.

Na rysunku 2.1 przedstawiono zbudowany układ naszego systemu. Zasilanie DC $5V$ pochodzi z modułu zasilającego do płytek stykowych. Zewnętrzne zasilanie urządzeń takich jak, czujnik temperatury, wyświetlacz LCD czy wentylator wynika z stosunkowo sporego prądu, który te urządzenia pobierają. Zasilanie DC $3,3V$ pochodzi bezpośrednio z płytki Nucleo.

2.3. Schemat elektryczny



Rysunek 2.2. Schemat układu wykonany w środowisku KiCad.

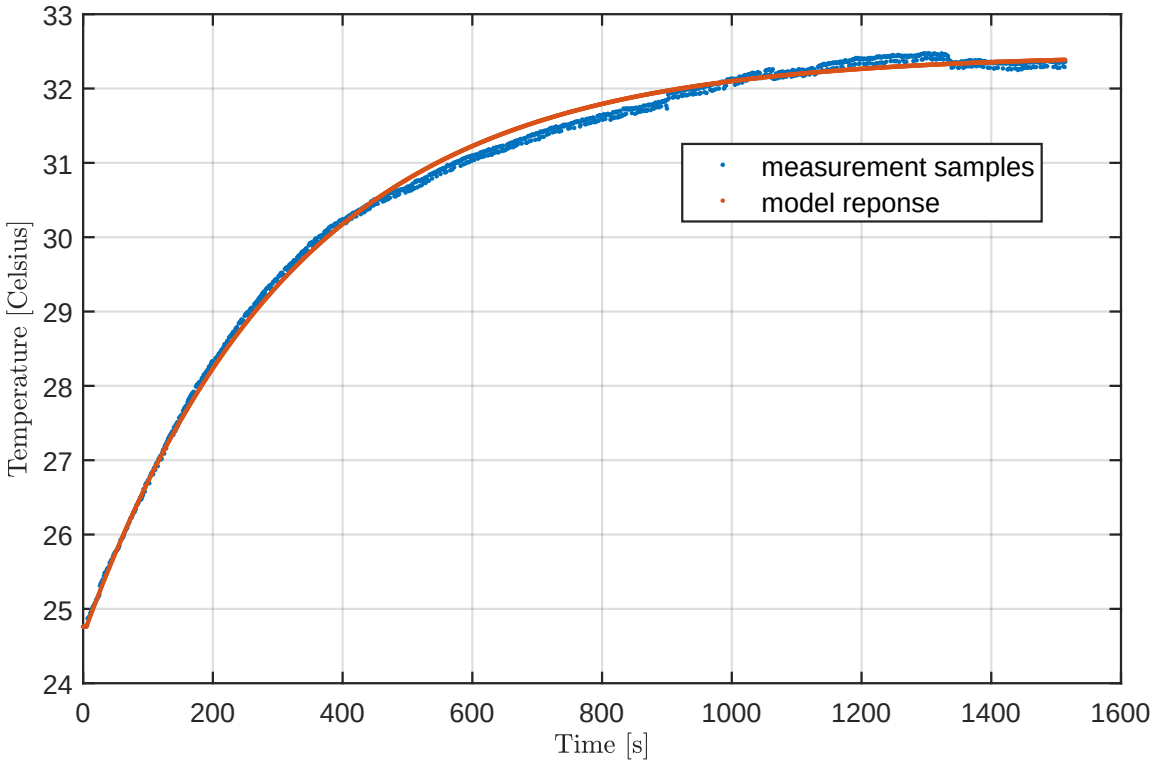
Opis schematu elektrycznego:

- Mikrokontroler NUCLEO-F756ZG (U1) – centralna jednostka sterująca systemem, odpowiedzialna za przetwarzanie danych oraz zarządzanie sygnałami sterującymi. Mikrokontroler jest zasilany za pomocą złącza microUSB, połączonego z dołączonym debugerem, co pozwala na wygodne programowanie oraz zasilanie układu.
- Czujnik BMP280 (U2) – odpowiedzialny za pomiar temperatury. Komunikuje się z mikrokontrolerem za pomocą interfejsu SPI, przesyłając dane pomiarowe do dalszego przetwarzania.
- Wyświetlacz LCD (U3) – dwuwierszowy ekran z konwerterem I2C (HD44780), który umożliwia uproszczoną komunikację z mikrokontrolerem. Wyświetlacz prezentuje wartość temperatury odczytanej z czujnika oraz wartość referencyjną w czasie rzeczywistym.
- Wentylator (M1) – sterowany za pomocą tranzystora TIP122 (Q1). TIP122 to układ Darlingtona, składający się z dwóch tranzystorów w konfiguracji, która zapewnia wysokie wzmocnienie prądowe. Pozwala to na efektywne sterowanie prądem wentylatora przy niewielkim sygnale sterującym z mikrokontrolera. Wentylator zapewnia aktywne chłodzenie systemu.
- Rezystor ceramiczny (R1) – obiekt regulacji temperatury. Sterowany jest za pomocą tranzystora TIP122 (Q2), działającego na podobnej zasadzie jak w przypadku wentylatora. Tranzystor, jako element wykonawczy, umożliwia podgrzewanie kontrolowanego środowiska w celu utrzymania zadanej temperatury.
- Potencjometr (R3) – pozwala na nastawę temperatury zadanej poprzez regulację napięcia podawanego na wejście ADC mikrokontrolera. Użytkownik może w prosty sposób zmieniać wartości docelowe temperatury za pomocą tego elementu.
- Elementy pasywne – w układzie wykorzystane są rezystory, które służą do zabezpieczania układów oraz ograniczania prądów sterujących w poszczególnych obwodach.

3. Analiza dynamiki systemu

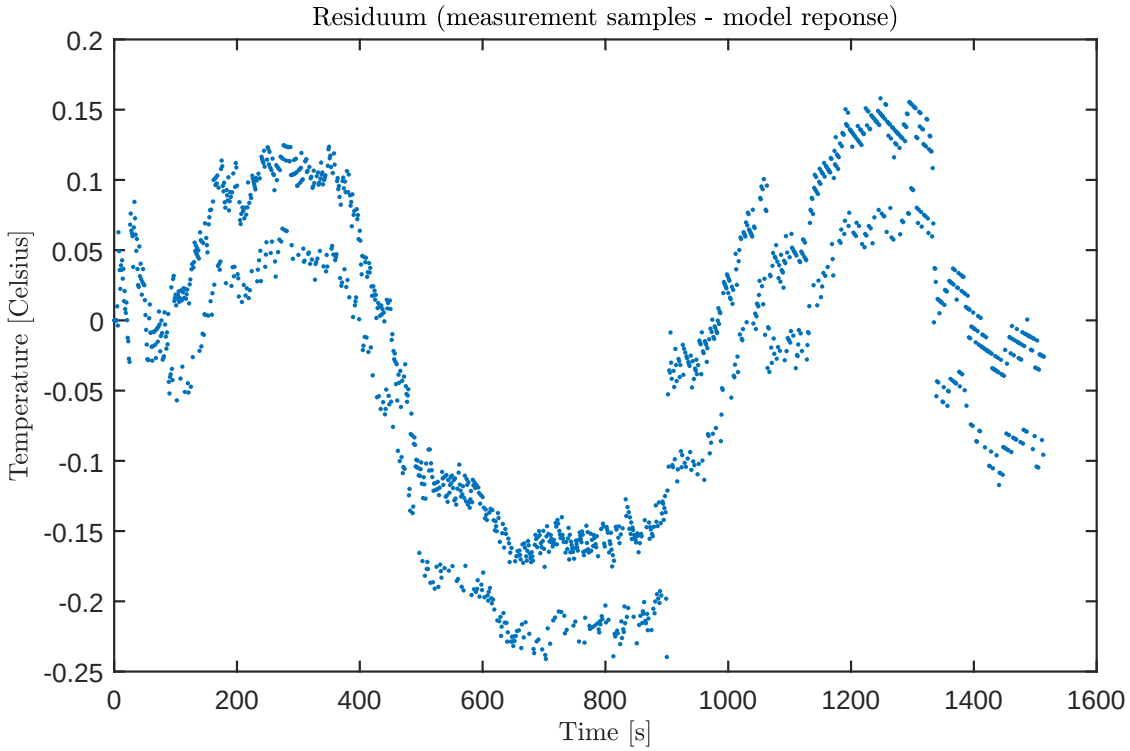
3.1. Pozyskanie danych i identyfikacja parametrów obiektu

W środowisku STM32CubeIDE widnieje nasz pusty projekt. W pierwszej kolejności realizujemy odczyt wartości temperatury z czujnika BMP280. Ta zmienna zostaje wysyłana poprzez port szeregowy (USART), do skryptu w Pythonie, który przechwytuje próbki i zapisuje je do pliku.txt. Za pomocą sygnału PWM sterujemy przełączaniem tranzystora TIP122, który dostarcza zasilanie do obiektu regulacji (rezystora ceramicznego). W celu identyfikacji obiektu, ustawiamy stałą wartość sygnału sterującego na poziomie 90% wypełnienia PWM. Tym samym otrzymamy odpowiedź obiektu na „umowny" skok jednostkowy. Taki przebieg umożliwi zdobycie informcji na temat dynamiki systemu i pozwolić dokonać aproksymacji modelu jako obiektu pierwszego rzędu z opóźnieniem transportowym. Zebrany zestaw danych został wyeksportowany do matlaba, tam dokonaliśmy nałożenia prostej i odczytania parametrów obiektu.



Rysunek 3.1. Porównanie zebranych danych z modelem obiektu.

Na rysunku 3.1 przedstawiono odpowiedź skokową obiektu w postaci porównania zebranych danych z nałożoną aproksymacją. Wyznaczono i wykreślono na przebiegu błąd bezwzględny tego przybliżenia. Na rysunku 3.2 widnieje wykres błędu. Wyniki uznajemy za satysfakcjonujące.



Rysunek 3.2. Przebieg błędu pomiędzy obiektem rzeczywistym a modelem.

Modelu jako obiekt pierwszego rzędu z opóźnieniem transportowym, opisany jest transmitancją:

$$G_o = \frac{K}{1+sT} e^{-sT_0}, \tag{3.1}$$

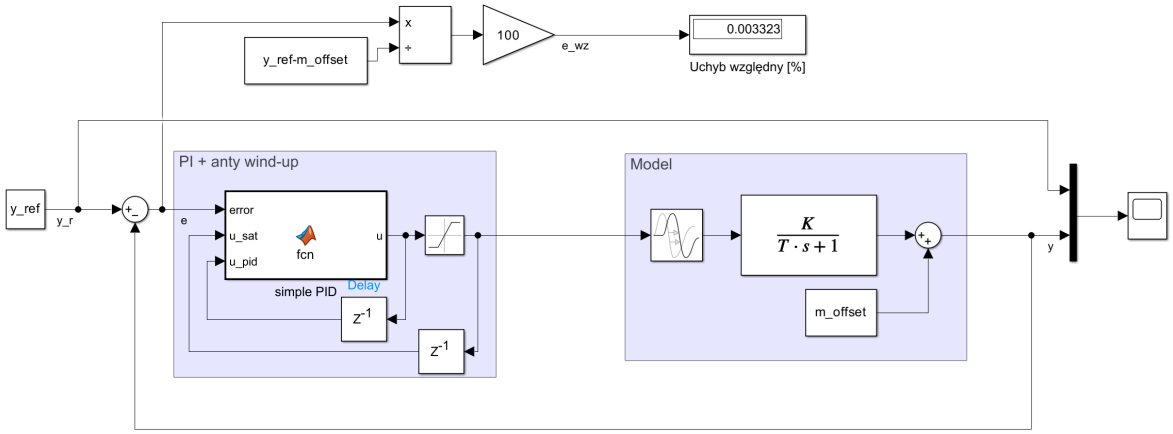
gdzie: K - wzmocnienie obiektu, T - stała czasowa, T_0 - opóźnienie transportowe. Przyjmujemy następujące parametry:

$$K = 10,6 \qquad T = 340 \qquad T_0 = 10$$

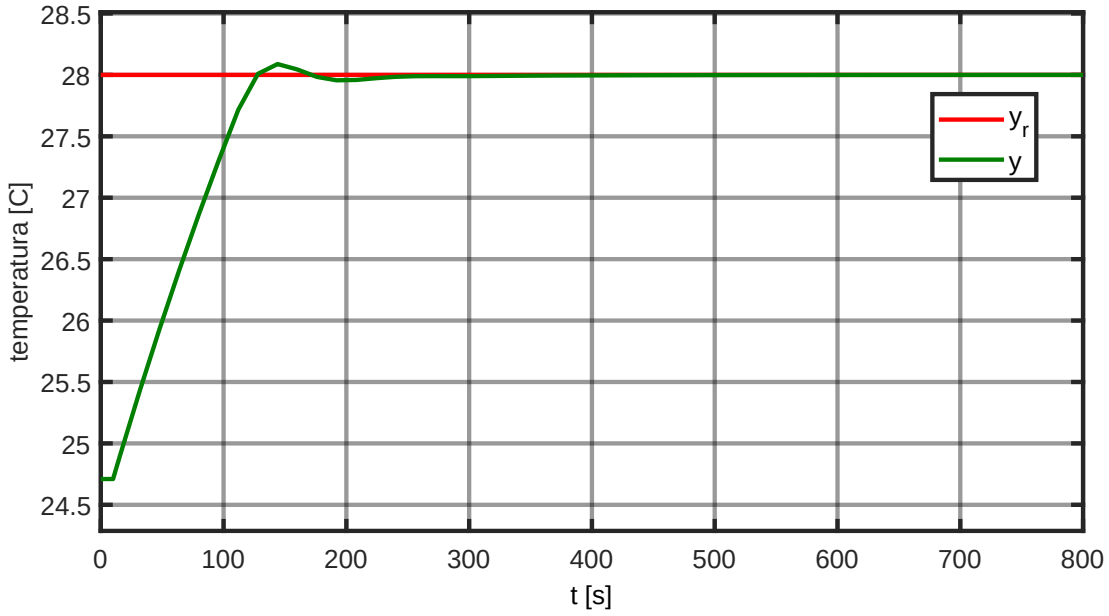
3.2. Wykonanie symulacji w środowisku *Simulink*

Wraz z wyznaczonymi parametrami obiektu przechodzimy do *Simulinka*. Modelujemy układ i implementujemy regulator PI z anty wind-up w postaci funkcji. Schemat takiego układu został przedstawiony na rysunku 3.3. Dla wartości referencyjnej $y_{ref} = 28^oC$ uzyskaliśmy przebieg pokazany na rysunku 3.4. Możemy zauważyć, że wstępne założenia projektowe są spełnione. Uchyb ustalony mieści się w tunelu 1%. Parametry zastosowanego regulatora:

$$k_p = 1,2 \qquad k_i = 0,02 \qquad k_d = 0$$



Rysunek 3.3. Układ regulacji w środowisku *Simulink*.

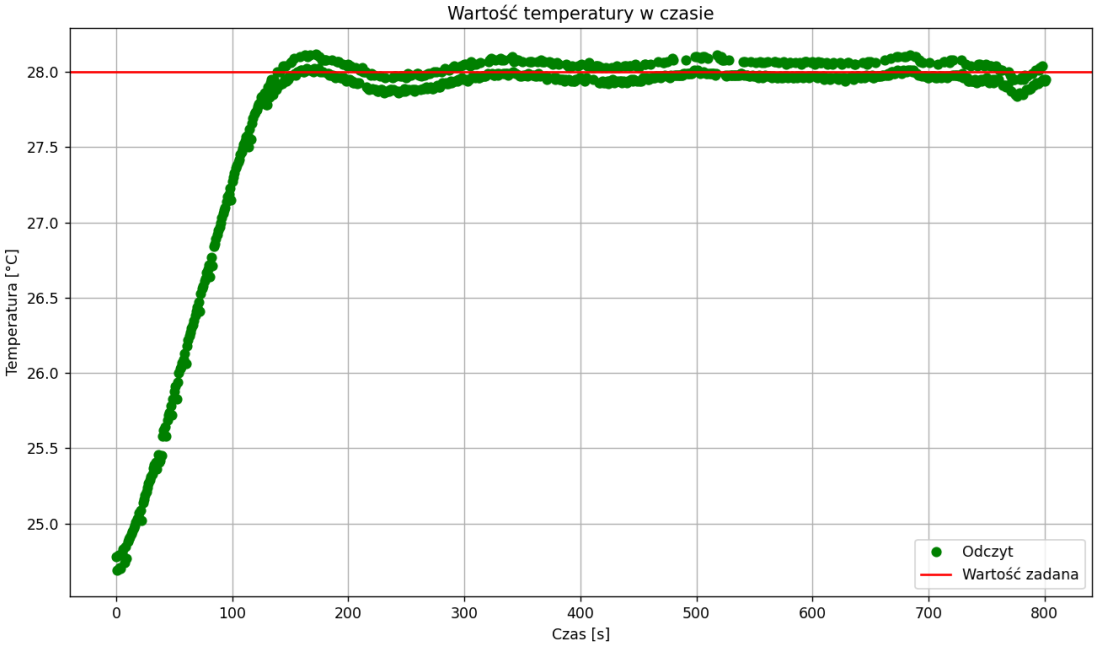


Rysunek 3.4. Przebieg odpowiedzi układu regulacji na stałą wartość zadaną.

4. Implementacja na rzeczywistym sprzęcie

4.1. Realizacja zaprojektowanego regulatora

Wybraną strukturą regulatora jest regulator typu PI z korektorem anty wind-up. Zastosowano wyznaczone parametry regulatora i przy pomocy skryptu w Pythonie, rozpoczęto zbieranie danych dla wartości referencyjnej $y_{ref} = 28^oC$. Na rysunku 4.1 przedstawiono odpowiedź obiektu rzeczywistego. Widzimy, że przebieg jest zbliżony z tym, uzyskanym w symulacji (rysunek 3.4).



Rysunek 4.1. Przebieg odpowiedzi układu regulacji (obiekt rzeczywisty).

Zastosowany korektor przy składowej całkującej regulatora zatrzymuje naładowanie się całki podczas nasycenia sygnału sterującego. Zmniejsza to przeregulowanie odpowiedzi i skraca czas regulacji.

4.2. Opis zastosowanych rozwiązań

4.2.1. Zadawanie wartości referencyjnej

System mikroprocesorowy został wyposażony w dwie możliwości zadawania wartości referencyjnej. Tryb wybiera się poprzez przyciśnięcie przycisku *user_button* na module Nucleo. Pierwszą z nich (domyślną) jest zadawanie wartości *setpoint* poprzez potencjometr. Odczytana wartość z przetwornika ADC zostaje odpowiednio przeliczona na zakres $23^oC - 30^oC$, a następnie przypisana do odpowiedniej zmiennej. Takie rozwiązanie umożliwia użytkownikowi wygodne zmienianie wartości zadanej.

Drugim sposobem na podanie wartości referencyjnej jest wykorzystanie portu szeregowego. Należy przycisnąć przycisk *user_button* (dioda LED1 zaczyna się świecić) i wysłać wartość referencyjną y_{ref} jako liczbę całkowitą 3-cyfrową (10-krotnie zwięszkoną, aby operować na zmiennej typu *int*). W tym trybie potencjometr jest wyłączony.

4.2.2. Regulacja temperatury

W domyślnym przypadku, nasza regulacji odbywa się z temperatury niższej do wyższej (zadanej). W takiej sytuacji, działanie wentylatora nie jest konieczne. Jednak na mikrokontrolerze została zaimplementowana regulacja z wykorzystaniem właśnie tego urządzenia. W momencie, gdy wartość zadana będzie niższa niż aktualna, wentylator rozpocznie pracę, aby schłodzić obiekt i szybciej osiągnąć temperaturę zadaną. Silnik wiatraka sterowany jest za pomocą sygnału PWM.

4.2.3. Wyświetlanie aktualnej temperatury

W systemie znajduje się wyświetlacz LCD, który na bieżąco uaktualnia wartość zadaną i odczytaną. Do głębszej analizy przydaje się zbierać dane w całym okresie regulacji. W tym celu został zaimplementowany skrypt w Pythonie, którego działanie przedstawia rysunek 4.1. Przy użyciu portu szeregowego, pokazuje odpowiedź układu regulacji w czasie rzeczywistym. Dodatkową funkcjonalnością jest zapis zebranych próbek do pliku.txt.

Spis rysunków