POLITECHNIKA POZNAŃSKA

WYDZIAŁ AUTOMATYKI, ROBOTYKI I ELEKTROTECHNIKI

INSTYTUT ROBOTYKI I INTELIGENCJI MASZYNOWEJ

ZAKŁAD STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

SPRAWOZDANIE

SYSTEMY MIKROPROCESOROWE (LABORATORIUM)
[WARIE_2021-22_AIR_DZ_1_5_D_LUCZAK_21/22]

BIBLIOTEKA CMSIS DSP
REGULATOR LINIOWY PROPORCJONALNO-CAŁKUJĄCO-RÓŻNICZKUJĄCY ORAZ DWUPOŁOŻENIOWY

(TEMAT ZAJĘĆ)

KAROL DĘBSKI

(AUTOR I: KAROL.DEBSKI@STUDENT.PUT.POZNAN.PL)

FORMA ZAJĘĆ: LABORATORIUM

PROWADZĄCY:

DR INŻ. DOMINIK ŁUCZAK

DOMINIK.LUCZAK@PUT.POZNAN.PL

POZNAŃ 20-*12-2021* 9-45

(*DATA I GODZINA ZAJĘĆ*)

Spis treści

Zadanie #3

1.1 Specyfikacja

Program automatycznie steruje obiektem rzeczywistym by utrzymywał zadaną wartość. Do weryfikacji poprawnego działania programu zostanie użyte oprogramowanie MATLAB do wyświetlania w czasie rzeczywistym przesłanych danych o obiekcie przez UART.

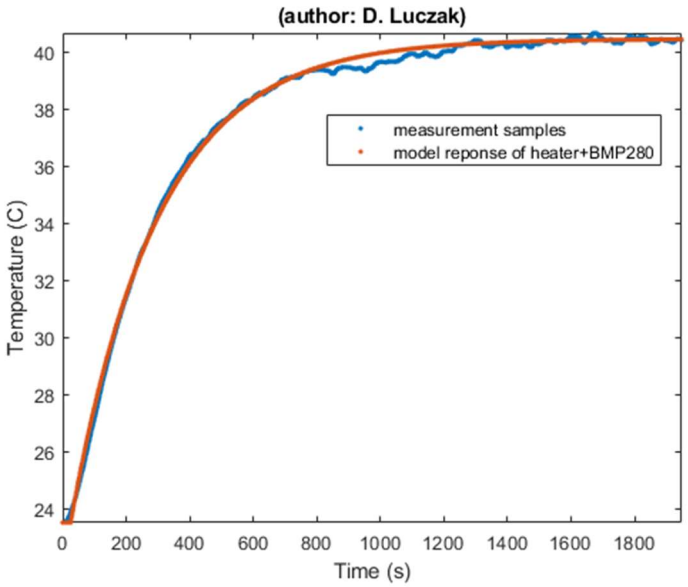
1.2 Implementacja

Rodzaj modelu matematycznego:

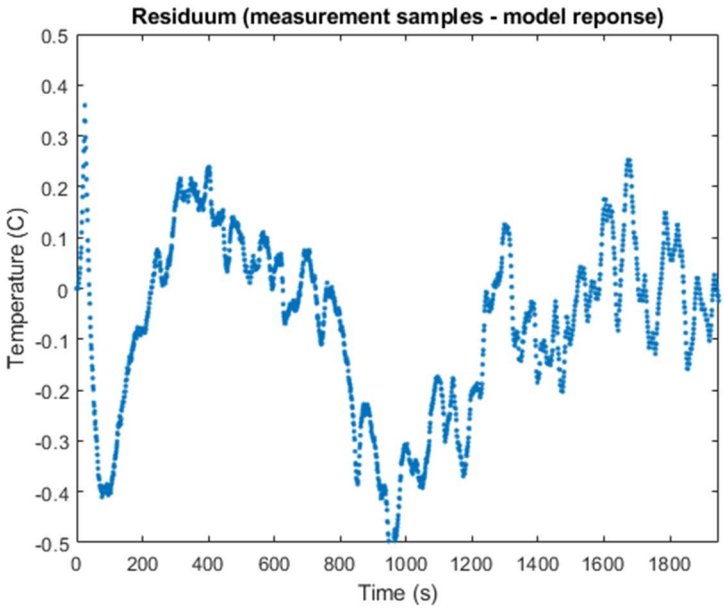
G=k/(1+s*T)*exp(-s*delay)

Parametry modelu matematycznego:

k = 34, T = 275, delay = 25



Rys 1. Wykres pomiarów i aproksymowanego modelu matematycznego



Rys 2. Wykres różnic w wartościach dla odpowiedzi modelu aproksymacji i próbek z rzeczywistego obiektu.

Listing 1 Funkcja obliczająca sygnał wyjściowy regulatora PID

```
float calculate_discrete_pid(PID_t* pid, float setpoint, float measured){
          error = setpoint-measured;
          P = pid->p.Kp * error;
          if(pid->previous pid output<1 && pid->previous pid output>0)
          {
                    integral = pid->previous_integral + (error+pid->previous_error);
                    pid->previous_integral = integral;
                    I = (pid->p.Ki)*integral*(pid->p.dt/2.0);
          }
          derivative=(error-pid->previous error)/pid->p.dt;
          pid->previous error = error;
          D = pid->p.Kd*derivative;
          u = P + I + D;
          return u;
}
```

Listing 2 Funkcja zwrotna dla przerwania na timerze

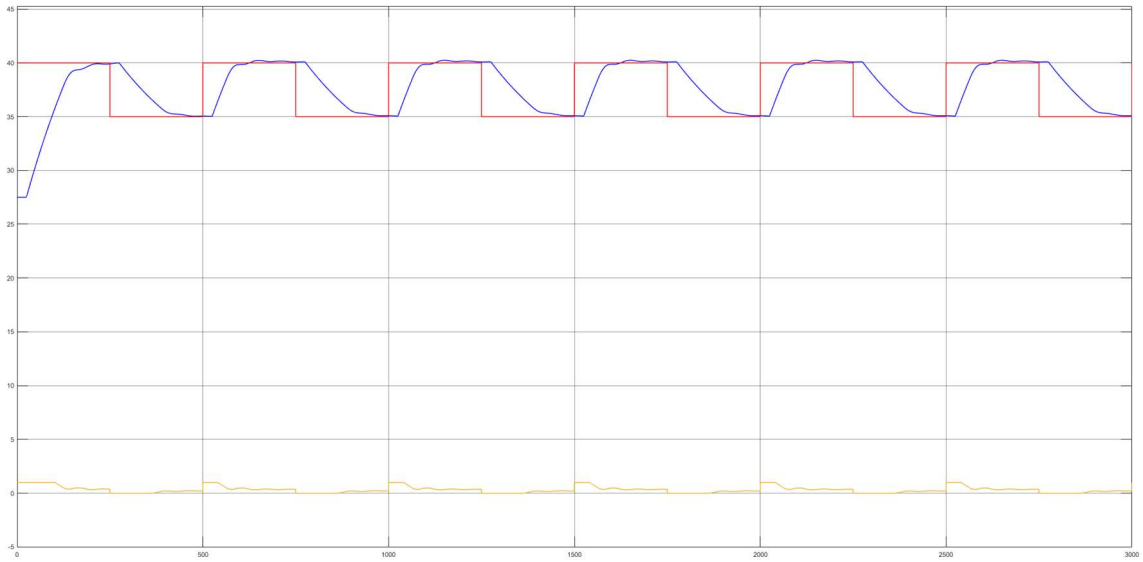
```
void HAL TIM PeriodElapsedCallback(TIM HandleTypeDef *htim){
          if(htim->Instance == TIM11){
                    set point=(t<250.0)? 40.0:35.0;
                    t=(t<500.0)?t+dt:0.0;
                    feedback_mesaurement=(float)temperature;
                    HAL_GPIO_TogglePin(LD3_GPIO_Port, LD3_Pin);
          pid output=calculate discrete pid(&pid1,set point,feedback mesaurement);
                    previous pid output=pid output;
                    if(pid output>1.0) pid output=1.0;
                    if(pid output<0) pid output=0;
                    pwm_duty=(uint16_t)(999.0*pid_output);
                    __HAL_TIM_SET_COMPARE(&htim10,TIM_CHANNEL_1,pwm_duty);
                    start measurement=1;
          }
}
```

Listing 3 Główna pętla programu

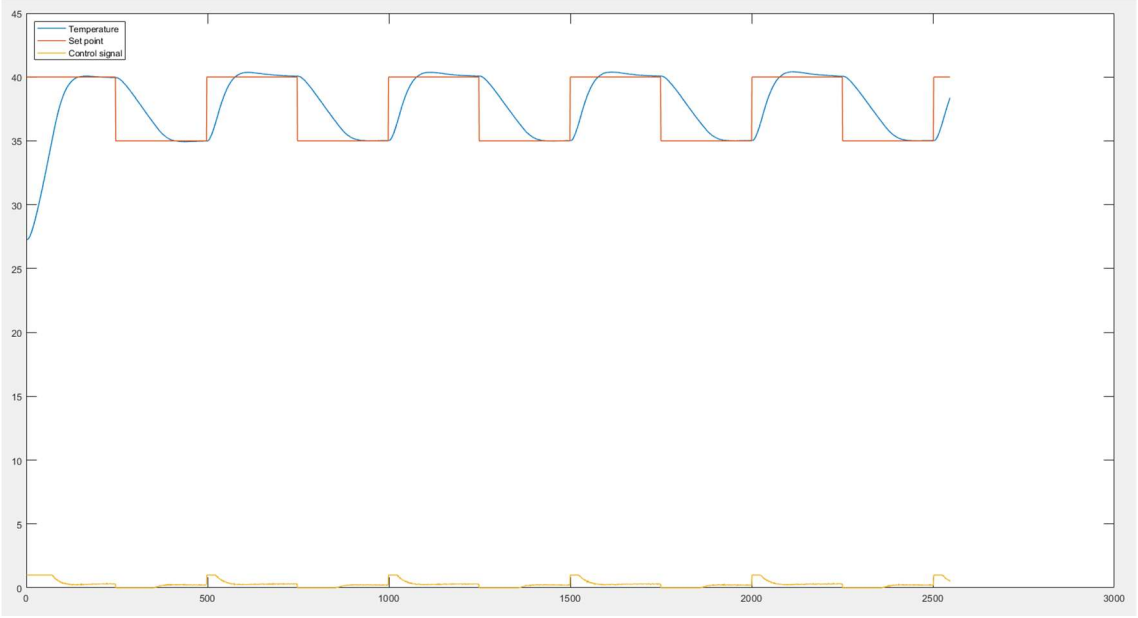
```
while (1)
  {
          if(start measurement){
                    BMP2_user_app_stream_sensor_data_by_uart(&dev);
                    start_measurement=0;
          }
    /* USER CODE END WHILE */
    MX USB HOST Process();

    /* USER CODE BEGIN 3 */
  }
```

Po przerwaniu na timerze obliczany jest sygnał wyjściowy regulatora PID i zadawany jest sygnał PWM na bramce tranzystora MOSFET. Następnie w pętli głównej programu mierzona jest temperatura rezystora grzewczego i przesyłane są wartości sygnału sterującego, wartości zadanej i temperatury.

1.3 Wynik testów



Rys. 3 Wykres wartości zadanej (czerwony), temperatury (niebieski) i sygnału sterującego (żółty) – symulacja w simulinku



Rys. 4 Wykres wartości zadanej, temperatury i sygnału sterującego – obiekt rzeczywisty

1.4 Wnioski

Program działa poprawnie, temperatura otoczenia rezystora dąży do zadanej wartości.

Podsumowanie

Zadanie 1 i 2 zostało sprawdzone na zajęciach przez prowadzącego.