Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Systemy Mikroprocesorowe w Sterowaniu

Sprawozdanie z projektu 2

Michał Kwarciński, Kacper Marchlewicz

Warszawa, 2022

Spis treści

1. Projekt 2

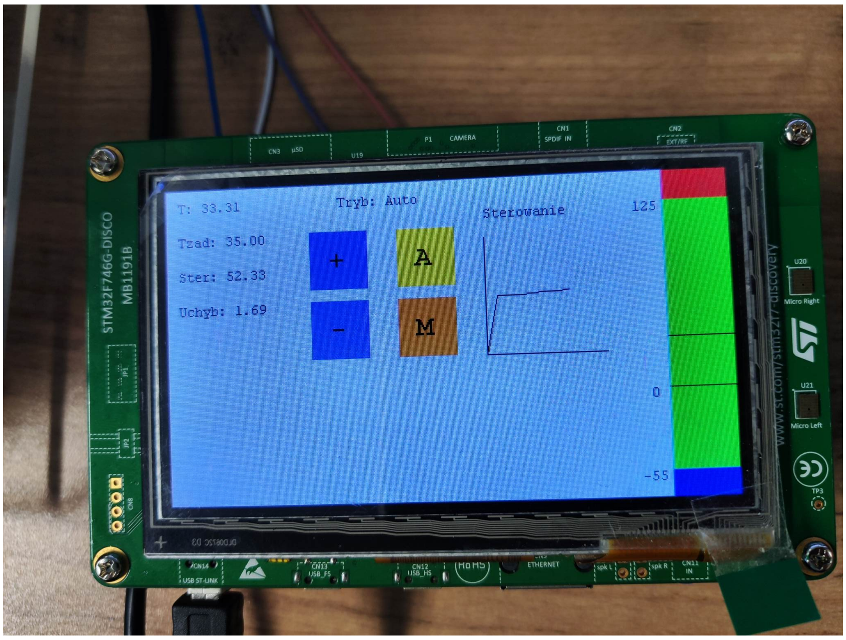
1.1. Interfejs użytkownika

W interfejsie umieściliśmy: na bieżąco aktualizowane wartości temperatury i temperatury zadanej, sterowania, uchybu. Znajduje się również informacja o obecnym trybie pracy, jak i przycisków do przełączania go, oraz guzików służących do zwiększania/zmniejszania temperatury zadanej bądź sterowania zależnie od trybu pracy. Z prawej strony znajduje się wykres sterowania i wskaźnik temperatury w postaci paska, na którym widać przedziały pracy - za gorąco, za zimno i optymalna temperatur, również punkt 0°C. Pod nimi będą pojawiać się komunikaty związane z awariami i alarmami, od lewej: nieprawidłowy odczyt temperatury z czujnika, nieprawidłowa (brak) komunikacji z obiektem i ostrzeżenie o za dużej temperaturze.

Struktura implementacji:

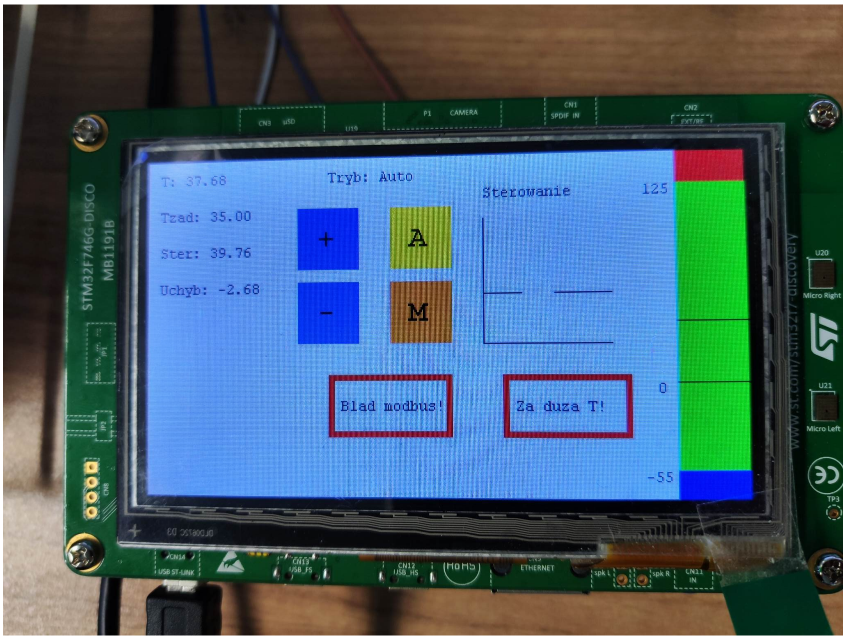
main() - w pętli while umieściliśmy wypisanie stale akutalizowanych parametrów i informacji o trybie auto/manual

HAL_LTDC_LineEventCallback() - rysowanie guzików, paska temperatur, wykresu sterowania i ostrzeżeń o alarmie/awarii



T: 33.31
Tryb: Auto
Tzad: 35.00
Ster: 52.33
Uchyb: 1.69

Rys. 1.1. Interfejs użytkownika



T: 37.68

Tzad: 35.00

Ster: 39.76

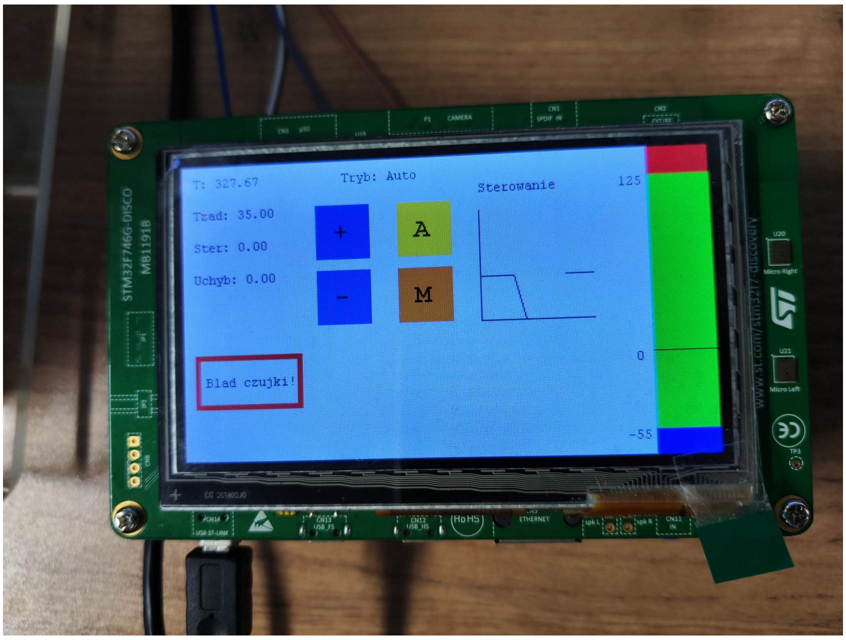
Uchyb: -2.68

Tryb: Auto

Blad modbus!

Za duza T!

Rys. 1.2. Awaria komunikacji z obiektem (MODBUS)



T: 327.67

Tzad: 35.00

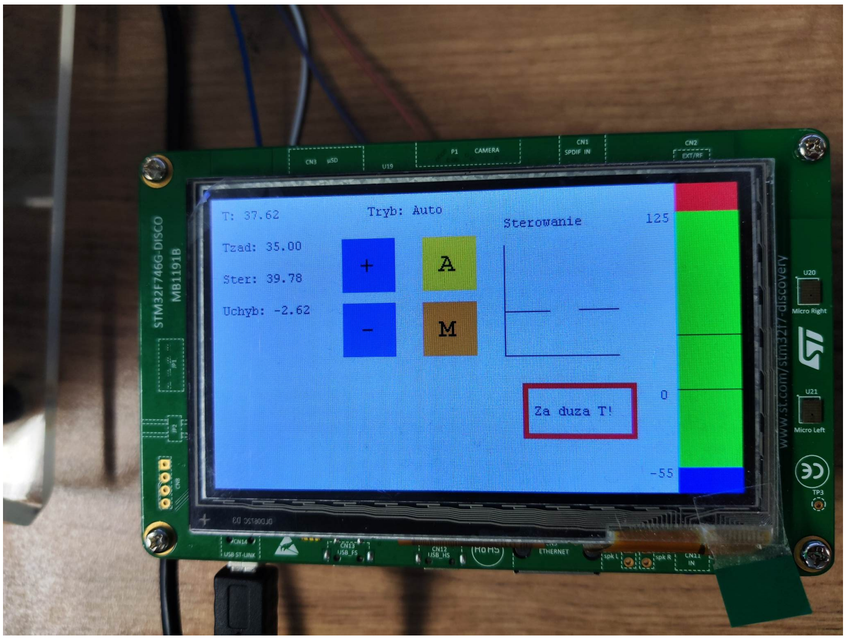
Ster: 0.00

Uchyb: 0.00

Tryb: Auto

Blad czujki!

Rys. 1.3. Odczyt czujnika temperatury wskazujący na jego awarie


T: 37.62

Tryb: Auto

Tzad: 35.00

Ster: 39.78

Uchyb: -2.62

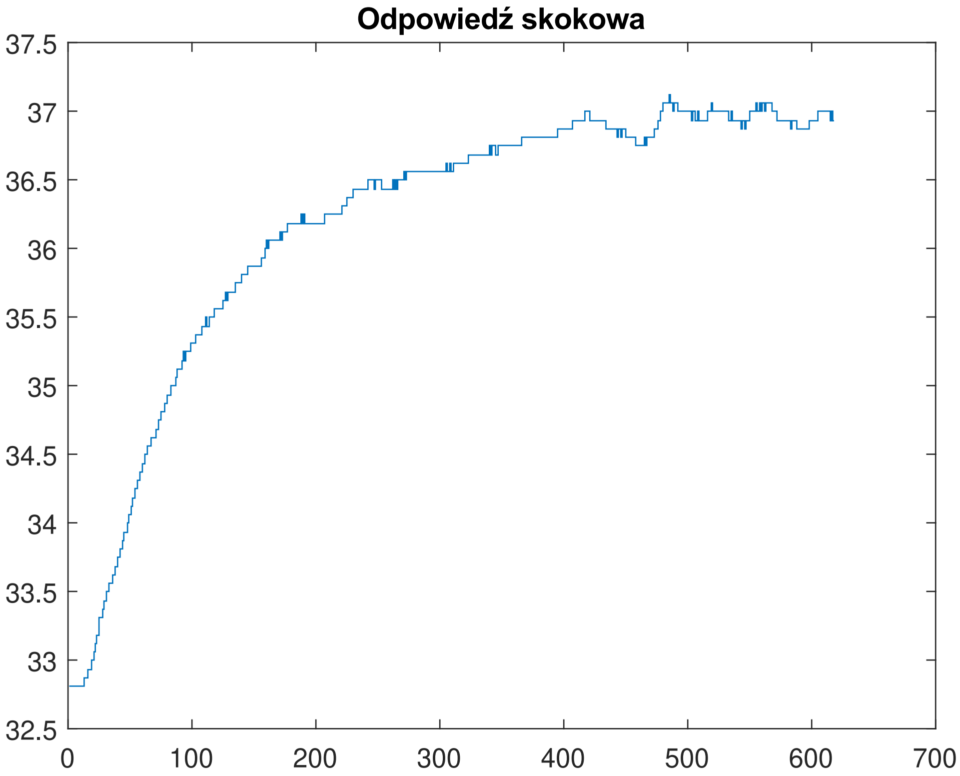
Za duza T!

Rys. 1.4. Przegrzanie grzałki - za duża temperatura

HAL_TIM_PeriodElapsedCallback() - w TIM2 umieściliśmy pętlę sterowania, ustawianie zmiennych związanych z alarmami i awariami ; w TIM5 natomiast mechanizm zmiany wartości sterowania/temperatury zadanej i trybu pracy poprzez kliknięcie odpowiednich przycisków

1.2. DMC

Rozpoczęliśmy od uzyskania odpowiedzi skokowej obiektu. Z uwagi pracy na tym samym stanowisku, skorzystaliśmy z pozyskanej wcześniej charakterystyki na laboratorium z przedmiotu PUST.



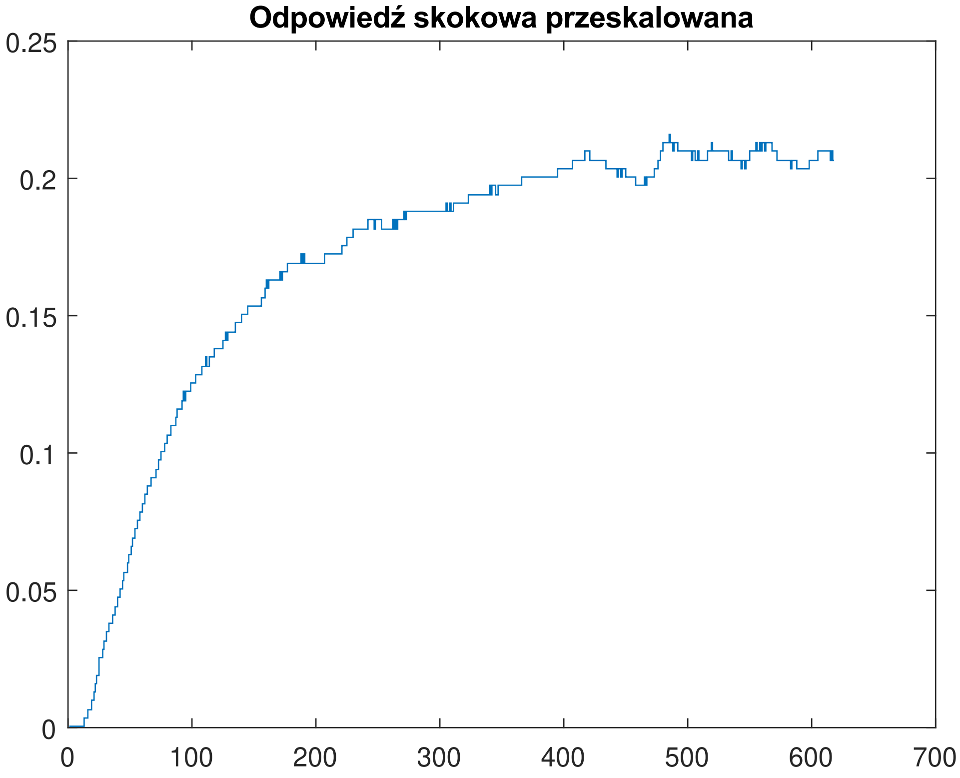
Rys. 1.5. Odpowiedź skokowa obiektu

Następnie otrzymane dane przeskalowaliśmy korzystając ze wzoru:

$$s = \frac{y_k - y_p}{dY} \tag{1.1}$$

gdzie:

s - przeskalowana odpowiedź skokowa.
dY - skok wartości.
y - odpowiedź skokowa przed przeskalowaniem.



Rys. 1.6. Przeskalowana odpowiedź skokowa obiektu

Na podstawie przeskalowanej odpowiedzi w Matlabie i parametrów D, λ, N i N_u (parametry te wyznaczaliśmy na przedmiocie PUST) stworzyliśmy skrypt w którym obliczamy macierze K (M i $\lambda * I$) i M_p, dzięki którym otrzymujemy parametr K_e i macierz K_u, które przepisujemy do kodu wgrywanego do mikrokontrolera.

W kodzie w C, stworzyliśmy funkcje do obsługi algorytmu DMC, w której liczymy przyrost sterowania w danej chwili i wektor DU_p, z którego pierwszy element jest szukanym przyrostem sterowania.

```
float DMC()
{
        float Ku_deltaUp = 0.0f;
        int i = 0, j = D;
        while (i < D - 1) {
                Ku_deltaUp += Ku[i] * deltaUP[i];
                i++;
        }
        deltaU = Ke * e - Ku_deltaUp;
        while (j > 0) {
                deltaUP[j] = deltaUP[j - 1];
                j--;
        }
        deltaUP[0] = deltaU;
        float x = u_previous + deltaU;
        u_previous = x;
        return x;
}
```