Kacper Szmajdel
Jakub Rajs

Implementacja obsługi fotorezystora i komunikacji za pomocą UART

Wstęp

W ramach laboratorium mieliśmy za zadanie zrealizować zadanie polegające na zaprogramowaniu płytki STM32 F411RE z nakładką Nucleo Multisensor w środowisku STM32CubeIDE w języku C. Program miał realizować zadanie poprzez sczytywanie sygnału na fotorezystorze oraz wysyłanie go za pomocą UART na port szeregowy. Dodane zostało również wyświetlanie przeliczonej wartości na wyświetlaczu nakładki Nucleo Multisensor.

Wykonanie zadania

Zadanie realizowane jest w dwóch przerwaniach oraz w trzech funkcjach:

```
69 void set_digit(int digit); // funkcja ustawiająca daną cyfrę na jednym z wyświetlaczy
70 void set_number(int digit1, int digit2, int digit3, int digit4); // funkcja ustawiająca liczbę na wyświetlaczach
71 int photoresistor_calc(float cur_val, float min_val, float max_val); // skalowanie wartości sygnału z fotorezystora
72
73 void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {
74     if (hadc->Instance == ADC1){
75 //      OBSLUGA Z EPSILONEM, dzięki czemu wahania wartości nie są bardzo małe
76         unsigned int adc_value_new = HAL_ADC_GetValue(&hadc1);
77         int epsilon = 10;
78         int difference = adc_value_new - adc_value;
79         if (abs(difference) > epsilon) {
80             adc_value = adc_value_new;
81         }
82
83 //      OBSLUGA BEZ EPSILONA
84 //      adc_value = HAL_ADC_GetValue(&hadc1);;
85     }
```

Kod 1. Przerwanie od zakończenia odbioru

```c
void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef *htim ) {
  if (htim->Instance == TIM9)
  {
//    WYSWIETLANIE WARTOSCI FOTOREZYSTORA
//    HAL_ADC_Start_IT(&hadc1);
//    set_number(adc_value/1000%10, adc_value/100%10, adc_value/10%10, adc_value%10);

//    WYSWIETLANIE WARTOSCI FOTOREZYSTORA Z USUWANIEM ZER PRZODUJACYCH
      HAL_ADC_Start_IT(&hadc1);
      int percent = photoresistor_calc((float)adc_value, 100, 4000);      // WARTOSCI RZECZYWISTE (Z POMIARU)
//    int percent = photoresistor_calc((float)adc_value, 700, 3000);      // WARTOSCI TESTOWE
      unsigned short int digit1 = percent/100%10;
      unsigned short int digit2 = percent/10%10;
      unsigned short int digit3 = percent%10;
      if (percent < 100) digit1 = 10;
      if (percent < 10) digit2 = 10;
      set_number(10, digit1, digit2, digit3);

//    WYSYLANIE WARTOSCI PRZEZ UART CO 1 SEKUNDE
      if(time == 200){
          int size = sprintf(uart_TX, "%d\r\n", percent);
          HAL_UART_Transmit_DMA(&huart2, uart_TX, size);
          time = 0;
      }
      else time++;
  }
}
```

Kod 2. Przerwanie od timera wraz z przeliczaniem sygnału na procenty

```c
void set_digit(int digit) {
    // Jesli podana "cyfra" nie jest cyfra to znaczy to, ze znak powinien być pusty
    switch (digit)
    {
        case 0:
        HAL_GPIO_WritePin(SEG_A_GPIO_Port, SEG_A_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(SEG_B_GPIO_Port, SEG_B_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(SEG_C_GPIO_Port, SEG_C_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(SEG_D_GPIO_Port, SEG_D_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(SEG_E_GPIO_Port, SEG_E_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(SEG_F_GPIO_Port, SEG_F_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(SEG_G_GPIO_Port, SEG_G_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(SEG_DP_GPIO_Port, SEG_DP_Pin, GPIO_PIN_RESET);
        break;

        case 1:
        HAL_GPIO_WritePin(SEG_A_GPIO_Port, SEG_A_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(SEG_B_GPIO_Port, SEG_B_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(SEG_C_GPIO_Port, SEG_C_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(SEG_D_GPIO_Port, SEG_D_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(SEG_E_GPIO_Port, SEG_E_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(SEG_F_GPIO_Port, SEG_F_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(SEG_G_GPIO_Port, SEG_G_Pin, GPIO_PIN_RESET);
```

Kod 3. Funkcja służąca do ustawiania odpowiednich segmentów wyświetlacza w zależności od podanej cyfry

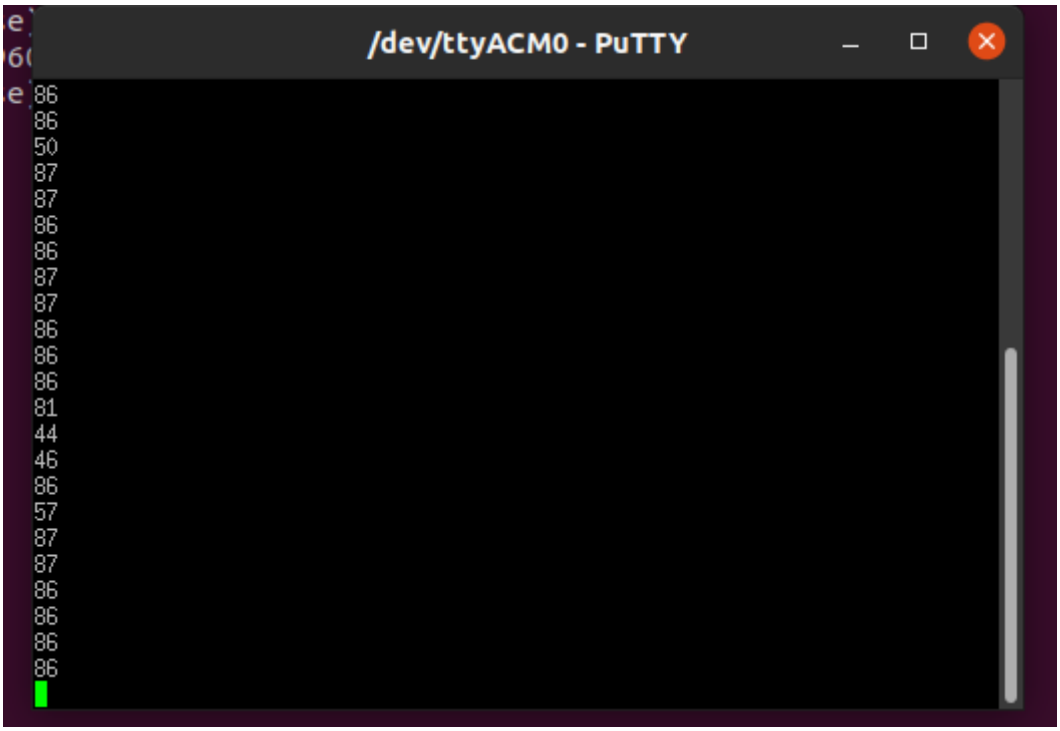
```c
void set_number(int digit1, int digit2, int digit3, int digit4) {
    switch(com){
        case 1:
        // Wyswietlacz 1
        HAL_GPIO_WritePin(COM4_GPIO_Port,COM4_Pin,GPIO_PIN_SET);
        HAL_GPIO_WritePin(COM1_GPIO_Port,COM1_Pin,GPIO_PIN_RESET);
        set_digit(digit1);
        com = 2;
        break;

        case 2:
        // Wyswietlacz 2
        HAL_GPIO_WritePin(COM1_GPIO_Port,COM1_Pin,GPIO_PIN_SET);
        HAL_GPIO_WritePin(COM2_GPIO_Port,COM2_Pin,GPIO_PIN_RESET);
        set_digit(digit2);
        com = 3;
        break;

        case 3:
        // Wyswietlacz 3
        HAL_GPIO_WritePin(COM2_GPIO_Port,COM2_Pin,GPIO_PIN_SET);
        HAL_GPIO_WritePin(COM3_GPIO_Port,COM3_Pin,GPIO_PIN_RESET);
        set_digit(digit3);
        com = 4;
        break;
```

Kod 4. Funkcja ustawiająca liczby na całym wyświetlaczu

```c
int photoresistor_calc(float cur_val, float min_val, float max_val)
{
    if (min_val == max_val)
    {
        min_val = 50; // zmierzone wartości minimalne i maksymalne sygnału fotorezystora
        max_val = 4050;
    }
    float temp = 1 / (min_val - max_val);
    float percent = 100 * (cur_val * temp - max_val * temp);
    if (percent > 100) percent = 100;
    else if (percent < 0) percent = 0;
    return (int)percent;
}
```

Kod 5. Funkcja przeliczająca wartości sygnału na procenty

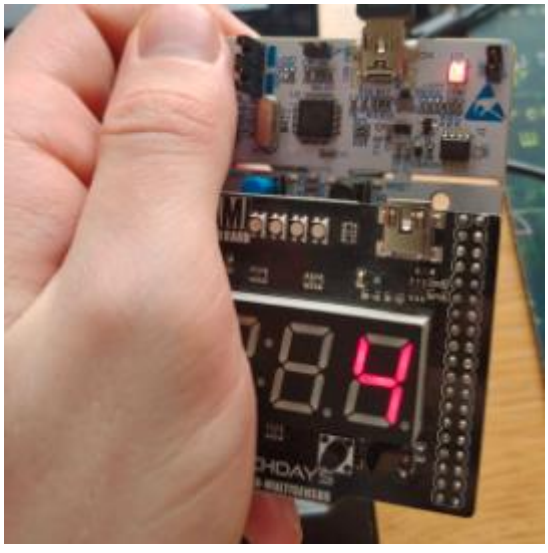
Po połączeniu się z płytką przez port szeregowy można zaobserwować jej działanie:



Rys. 1. Okno komunikacji z portem szeregowym, do którego podpięta jest płytka

Dodatkowa funkcjonalność, tj. wyświetlanie odczytanych wartości procentowych na wyświetlaczu:

Zdj. 1. Działanie bez zakrycia fotorezystora



Zdj. 2. Działanie z zakrytym fotorezystorem

Wnioski

Dzięki temu ćwiczeniu dowiedzieliśmy się, jak wykorzystywać peryferia takie jak fotorezystor dołączony do płytki oraz jak sczytywać z niego sygnał i manipulować nim, aby uzyskać wynik przeliczony na procenty. Dodatkowo, pogłębiliśmy swoją wiedzę z zakresu płytek STM32 oraz środowiska STM32CubeIDE i języka C, w którym są one programowane. Zadanie nie stanowiło dla nas większego problemu ze względu na fakt, że nauczyliśmy się korzystać z UART'a w ramach poprzedniego laboratorium. Dzięki temu najwięcej pracy było wymagane przy przeliczaniu sygnałów oraz wyświetlaczach, a nie przy konfiguracji projektu.