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]

WEJŚCIA ANALOGOWE (ADC - PRZETWORNIK A/C; OBSŁUGA ANALOGOWYCH CZUJNIKÓW (FOTOREZYSTOR, TERMISTOR))

(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 29-*11-2021* 9-45

(*DATA I GODZINA ZAJĘĆ*)

Spis treści

Zadanie #1

1.1 Specyfikacja

Program odczytuje wartość napięcia na nóżce potencjometru z wykorzystaniem ADC. Do weryfikacji poprawnego działania programu zostanie użyte SWV.

1.2 Implementacja

Listing 1 Główna pętla programu

```
while(1){
          HAL_ADC_Start(&hadc1);
          variable=HAL_ADC_GetValue(&hadc1);
          result = 2950*(variable/(pow(2,12)));
          HAL_Delay(10);
          /* USER CODE END WHILE */
          MX_USB_HOST_Process();
          /* USER CODE BEGIN 3 */
}
```

Najpierw inicjowany jest pomiar na ADC, następnie z rejestru pobierana jest wartość odczytu a kolejno obliczana jest wartość odczytu w miliwoltach.

1.3 Wynik testów

Rys. 1 Podgląd SWV

1.4 Wnioski

Wartość napięcia zmienia się od 0 do 2 V tak jak wynikałoby to z obliczeń, program działa poprawnie. Poziom szumów jest dość wysoki, w wcześniejszych pomiarach występowały piki.

Zadanie #2

2.1 Specyfikacja

Program ustawia stan linijki LED na podstawie odczytanego napięcia z nóżki potencjometru. Do weryfikacja poprawnego działania programu zostanie nagrane video.

2.2 Implementacja

Listing 2 Główna pętla programu

```
while (1) {
                    HAL ADC Start(&hadc1);
                    variable=HAL ADC GetValue(&hadc1);
                    result = 2.95*(variable/(pow(2,12)));
                    if(result<500){
                              HAL_GPIO_WritePin(LD4_GPIO_Port, LD4_Pin, GPIO_PIN_RESET);
                              HAL GPIO WritePin(LD3 GPIO Port, LD3 Pin, GPIO PIN RESET);
                              HAL GPIO WritePin(LD5 GPIO Port, LD5 Pin, GPIO PIN RESET);
                    }else if(result>500 && result < 1000){
                              HAL_GPIO_WritePin(LD4_GPIO_Port, LD4_Pin, GPIO_PIN_SET);
                              HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
                              HAL_GPIO_WritePin(LD5_GPIO_Port, LD5_Pin, GPIO_PIN_RESET);
                    }else if(result>1000 && result < 1500){
                              HAL GPIO WritePin(LD4 GPIO Port, LD4 Pin, GPIO PIN SET);
                              HAL GPIO WritePin(LD3 GPIO Port, LD3 Pin, GPIO PIN SET);
                              HAL_GPIO_WritePin(LD5_GPIO_Port, LD5_Pin, GPIO_PIN_RESET);
                    }else{
                              HAL GPIO WritePin(LD4 GPIO Port, LD4 Pin, GPIO PIN SET);
                              HAL GPIO WritePin(LD3 GPIO Port, LD3 Pin, GPIO PIN SET);
                              HAL GPIO WritePin(LD5 GPIO Port, LD5 Pin, GPIO PIN SET);
                    }
                    HAL_Delay(10);
                    /* USER CODE END WHILE */
                    MX USB HOST Process();

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

Najpierw inicjowany jest pomiar na ADC, następnie z rejestru pobierana jest wartość odczytu a kolejno obliczana jest wartość odczytu w miliwoltach. Na podstawie odczytu w miliwoltach ustawiany jest stan linijki LED.

2.3 Wynik testów

https://drive.google.com/file/d/1vNnHiQZA89JJ5olGjRARQ8nYFjj-RCbx/view?usp=sharing

(15 sekund)

2.4 Wnioski

Program działa poprawnie, w miarę przekręcania gałki potencjometru zapalają się diody.

Zadanie #3

3.1 Specyfikacja

Program wysyła na terminal aktualny odczyt z ADC i jego wartość wyrażoną w woltach. Do weryfikacji poprawnego działania programu zostanie nagrane video.

3.2 Implementacja

Listing 3 Główna pętla programu

```
while (1) {
          HAL_ADC_Start(&hadc1);
          variable=HAL_ADC_GetValue(&hadc1);
          result = 2.95*(variable/(pow(2,12)));
          HAL_Delay(10);
        /* USER CODE END WHILE */
        MX_USB_HOST_Process();

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

Listing 4 Funkcja zwrotna dla przerwania na przycisku

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
          if(GPIO_Pin == Button_Pin){
                    uart_size = sprintf((char*)uart_data, "{\"channel\":\"adc_in %d\",
\"voltage\":%.2f}\r\n",(int16_t)variable,result);
                    HAL_UART_Transmit(&huart2, uart_data, uart_size, 1000);
          }
}
```

W pętli głównej dokonywany jest odczyt z ADC oraz przekształca się go na wartość w woltach. W funkcji zwrotnej tworzony jest tekst z wartością odczytu ADC oraz z wartością w woltach odczytu. Następnie ten tekst jest wysyłany na terminal.

3.3 Wynik testów

https://drive.google.com/file/d/1w18kk5EMLVWd_0fNCNs4V_FgPC6tlXD5/view?usp=sharing

(9 sekund)

3.4 Wnioski

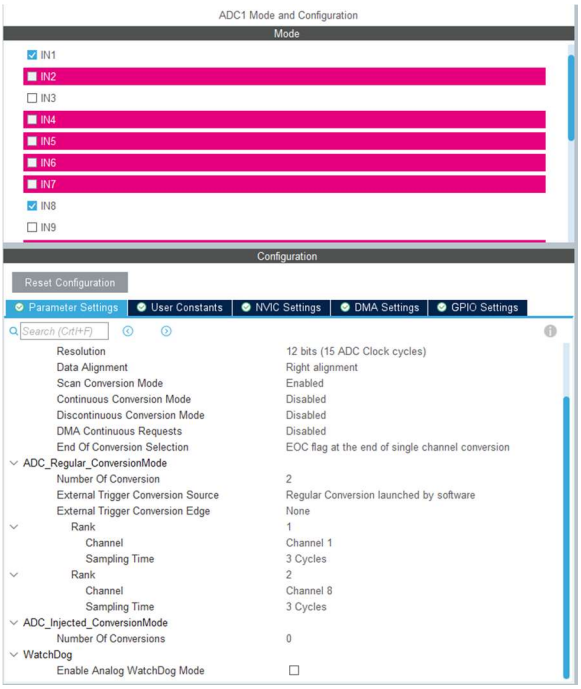
Po wciśnięciu przycisku wysyłany jest tekst, program działa poprawnie.

Zadanie #4

4.1 Specyfikacja

Program odczytuje wartości z 2 kanałów ADC1. Do weryfikacji poprawnego działania programu zostanie użyte SWV.

4.2 Implementacja



Rys. 2 Konfiguracja ADC

Listing 4 Główna pętla programu

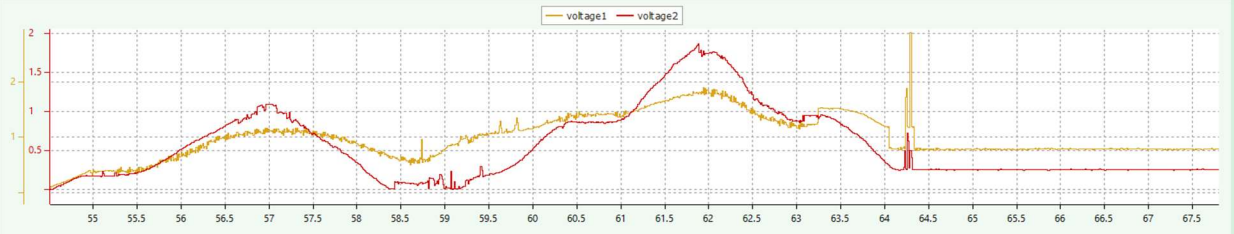
```
while (1) {
          if(HAL_ADC_PollForConversion(&hadc1, 10) == HAL_OK)
               {
                 raw val1 ADC = HAL ADC GetValue(&hadc1);
                 voltage1=(float t)(2.9*(raw val1 ADC/pow(2,12)));
                 ADC SetActiveChannel(&hadc1, 8);
                 HAL_ADC_Start(&hadc1);
               }
         if(HAL_ADC_PollForConversion(&hadc1, 10) == HAL_OK)
               {
                 raw val2 ADC = HAL ADC GetValue(&hadc1);
                 voltage2=(float t)(2.9*(raw val2 ADC/pow(2,12)));
                 ADC_SetActiveChannel(&hadc1, 1);
                 HAL_ADC_Start(&hadc1);
               }
        HAL Delay(10);
    /* USER CODE END WHILE */
    MX USB HOST Process();

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

Listing 5 Funkcja wybierająca kanał ADC do konwersji

```
void ADC_SetActiveChannel(ADC_HandleTypeDef *hadc, uint32_t AdcChannel){
          ADC_ChannelConfTypeDef sConfig = {0};
          sConfig.Channel = AdcChannel;
          sConfig.Rank = 1;
          sConfig.SamplingTime = ADC SAMPLETIME 3CYCLES;
          if (HAL_ADC_ConfigChannel(hadc, &sConfig) != HAL_OK)
               {
                  Error Handler();
               }
}
```

Kiedy konwersja zostanie zakończona na przetworniku przetwarzana jest surowa wartość odczytu na wartość którą można wyrazić w woltach. Następnie wybierany jest kanał 8 przetwornika i inicjalizowany jest odczyt napięcia. Kolejno znowu program czeka na zakończenie konwersji i ponownie przetwarzana jest wartość surowa odczytu ale już dla kanału 8 a nie jak poprzednio dla kanału 1.

4.3 Wynik testów



Rys. 3 Podgląd SWV

4.4 Wnioski

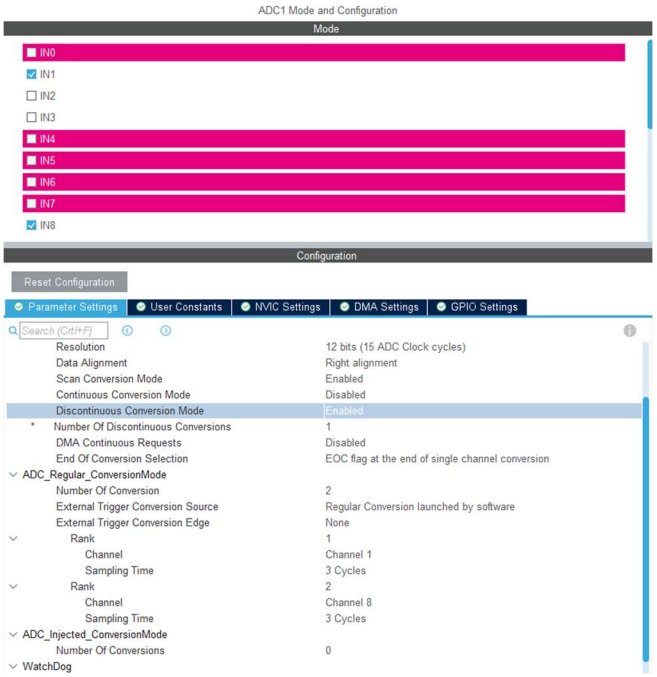
Poziom szumów jest dość duży, wzmacniacz pomiarowy lub filtr wygładziłby przebiegi. Kanały nie pracują niezależnie, w tym samym czasie pomiar może być dokonany tylko na jednym kanale.

Zadanie #5

5.1 Specyfikacja

Program będzie dokonywał odczytu z 2 kanałów ADC z wykorzystaniem przerwań. Do weryfikacji poprawnego działania programu zostanie użyte SWV.

5.2 Implementacja



Rys. 4 Konfiguracja ADC

Listing 5 Funkcja zwrotna dla przerwania na ADC

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef * hadc){
        if(hadc == &hadc1){
                while(__HAL_ADC_GET_FLAG(hadc, ADC_FLAG_EOC)){
                  ADC_raw[index_] = HAL_ADC_GetValue(hadc);
                  index ++;
                  if(index_!=2){
                            HAL ADC Start IT(&hadc1);
                            while(!__HAL_ADC_GET_FLAG(hadc, ADC_FLAG_EOC)){}
                            }
                }
                index =0;
                voltage1 = 2.9 * (ADC raw[0])/4096;
                voltage2 = 2.9 * (ADC_raw[1])/4096;
                __delay_ms(10);
                HAL_ADC_Start_IT(&hadc1);
        }
}
```

Po zakończeniu konwersji dla pierwszego kanału jest wywołana funkcja zwrotna. Następnie w funkcji zwrotnej odczytywana jest wartość zmierzona z kanału 1. Następnie ADC jest wyzwalany do następnej konwersji. Jeśli dla całej grupy kanałów została dokonana konwersja to obliczane są napięcia i ponownie wyzwalany jest ADC do konwersji.

5.3 Wynik testów

Rys. 5 Podgląd SWV

5.4 Wnioski

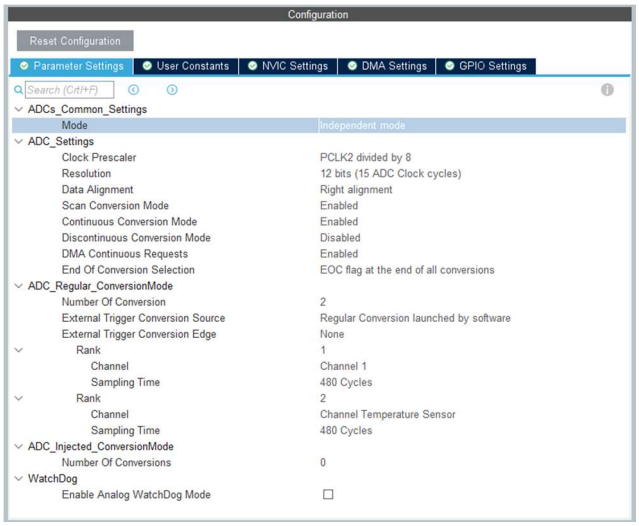
Program dokonuje konwersji dla dwóch kanałów w jednej wywołaniu funkcji zwrotnej, wszystko działa poprawnie.

Zadanie #6

6.1 Specyfikacja

Program dokonuje konwersji przez ADC dla dwóch kanałów z wykorzystaniem DMA. Do weryfikacji poprawnego działania programu zostanie użyty SWV.

6.2 Implementacja



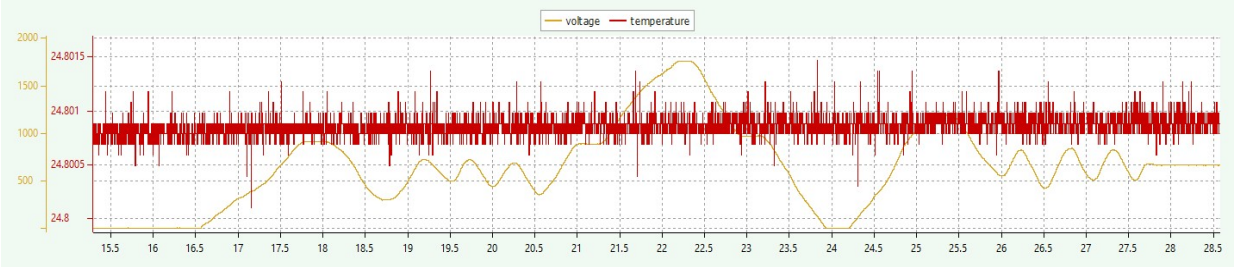
Rys. 6 Konfiguracja ADC

Listing 6 Funkcja zwrotna dla przerwania na ADC

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef * hadc){
          if(hadc == &hadc1){
                    voltage = 2950 * (ADC_raw[0])/4096.0;
                    temperature=(((float t)(ADC raw[1]/4096.0)-0.76)/2.5)+25;
                      delay ms(10);
          }
}
```

Po zakończeniu konwersji dla dwóch kanałów wywoływana jest funkcja zwrotna która oblicza napięcie na kanale pierwszym i temperaturę wskazywaną przez czujnik.

6.3 Wynik testów



Rys. 7 Podgląd SWV

6.4 Wnioski

Podczas przekręcania gałki potencjometru zmienia się napięcie a wyliczona temperatura jest wiarygodna.

Podsumowanie

Do poprawnego zinterpretowania odczytu z ADC dla potencjometrów trzeba przyjąć mnożnik równy napięciu na pinie 3.3 V dlatego, że właśnie takie napięcie jest odniesieniem dla układu.