Mikroprocesory i mikrokontrolery Laboratorium nr 9	Temat: Przetwornik ADC
Grupa: 21b	Michał Lechowicz

Cel ćwiczenia:

Celem ćwiczenia było zapoznanie się z działaniem i obsługą przetwornika ADC w mikrokontrolerze **STM32FO**. Mikrokontroler oparty jest o technologię ARM. Przetwornik służy do konwersji sygnału analogowego na cyfrowy. Do konfiguracji kontrolera posłużył program CubeMX oraz IDE IAR.

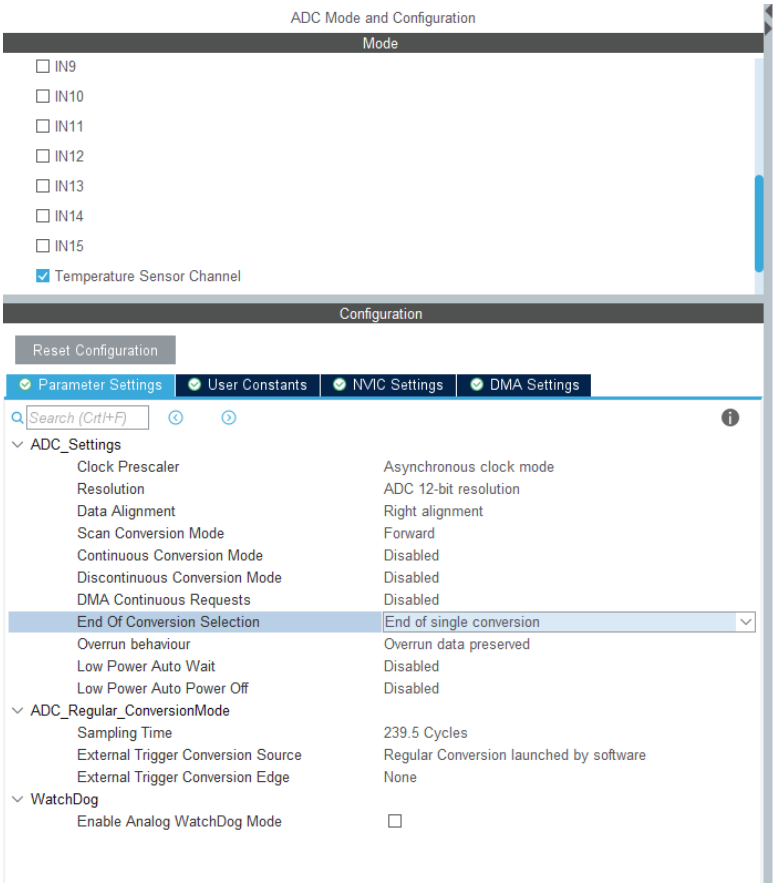
Zadanie na ocenę 5.0:

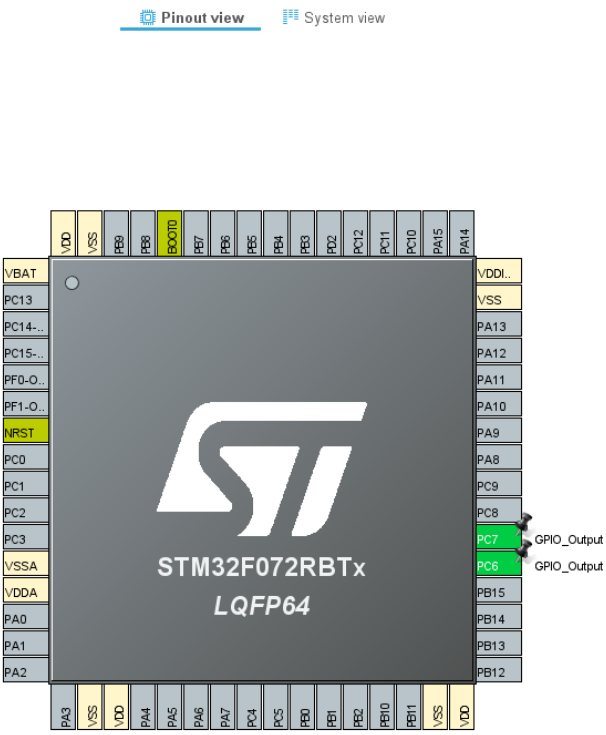
1. Opis zadania:

W zadaniu należało napisać program, w którym wykorzystamy dwie diody LED oraz termometr wbudowany w płytkę. Program ma za zadanie zbadać trend temperatury, jeśli temperatura rośnie ma on zapalić diodę czerwoną. Jeśli maleje, diodę niebieską.

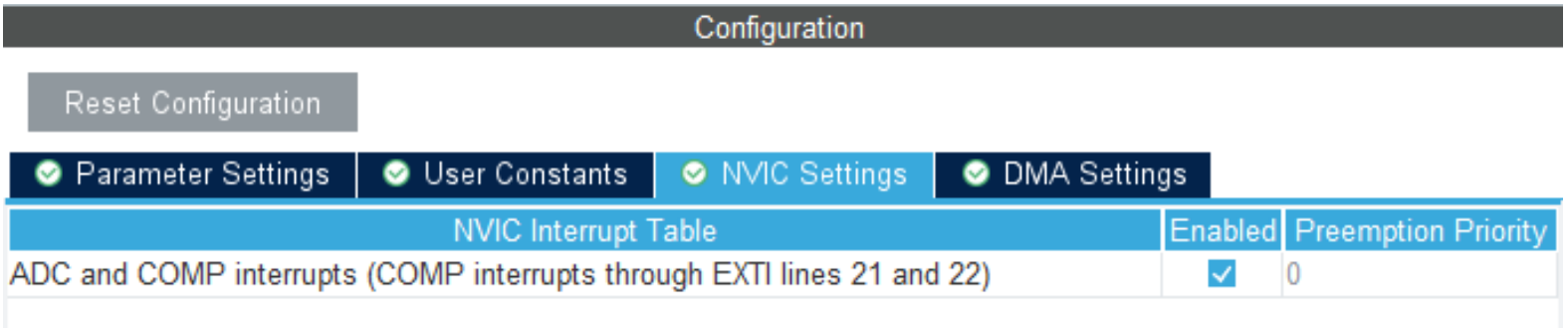
2. Opis programu:

1. Konfiguracja programu STM32CubeMX;





Rys. 1 Widok na konfigurację programu



Rys. 2 Konfiguracja przerwań

2. Konfiguracja portów wykonana przez Cube;

main.c * | stm32f0xx_hal_adc.c | stm32f0xx_hal_adc_ex.c | stm32f0xx_hal.c | stm32f0xx_it.c

MX_GPIO_Init()

```
274   * @param None
275   * @retval None
276   */
277  static void MX_GPIO_Init(void)
278  {
279    GPIO_InitTypeDef GPIO_InitStruct = {0};
280
281    /* GPIO Ports Clock Enable */
282    __HAL_RCC_GPIOC_CLK_ENABLE();
283
284    /*Configure GPIO pin Output Level */
285    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6|GPIO_PIN_7, GPIO_PIN_RESET);
286
287    // konfiguracja odpowiedzialna za diody
288    /*Configure GPIO pins : PC6 PC7 */
289    GPIO_InitStruct.Pin = GPIO_PIN_6|GPIO_PIN_7;
290    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
291    GPIO_InitStruct.Pull = GPIO_NOPULL;
292    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
293    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
294
295  }
296
```

Rys. 3. Konfiguracja portów

3. Ustawienie zmiennych odpowiedzialnych za przechowywanie diod. Zaświecenie czerwonej diody;

main.c | stm32f0xx_hal_adc.c | stm32f0xx_hal_adc_ex.c | stm32f0xx_hal.c | stm32f0xx_it.c

main()

```
35  /* USER CODE BEGIN PD */
36  #define size_T 20 // rozmiar tablicy na odczytu z sensora
37  /* USER CODE END PD */
38
39  /* Private macro -------------------------------------------------------------*/
40  /* USER CODE BEGIN PM */
41
42  /* USER CODE END PM */
43
44  /* Private variables ---------------------------------------------------------*/
45  ADC_HandleTypeDef hadc;
46
47  /* USER CODE BEGIN PV */
48  int ADC_raw = 0; // zmienna na ostatni odczyt z sensora
49  int ADC_last = 0; // zmienna na przedostatni odczyt z sensora
50  int difference = 3; // dokladnosc pomiaru, zalozony blad odczytu
51  int sizeOfTable = size_T; // przypisanie rozmiaru tablicy
52  int hist_data[size_T]; // definicja bufora cyklicznego na odczyty z sensora
53  int index = 0; // index bufora
54  int first_run = 1; // wskaznik pierwszego uruchomienia
55
56  uint32_t red_diode = GPIO_PIN_6; /* Przypisanie startowych wartosci dla obu diod */
57  uint32_t blue_diode = GPIO_PIN_7;
58
59  /* USER CODE END PV */
60
61  /* Private function prototypes -----------------------------------------------*/
62  void SystemClock_Config(void);
63  static void MX_GPIO_Init(void);
64  static void MX_ADC_Init(void);
65  /* USER CODE BEGIN PFP */
66
67  /* USER CODE END PFP */
68
69  /* Private user code ---------------------------------------------------------*/
```

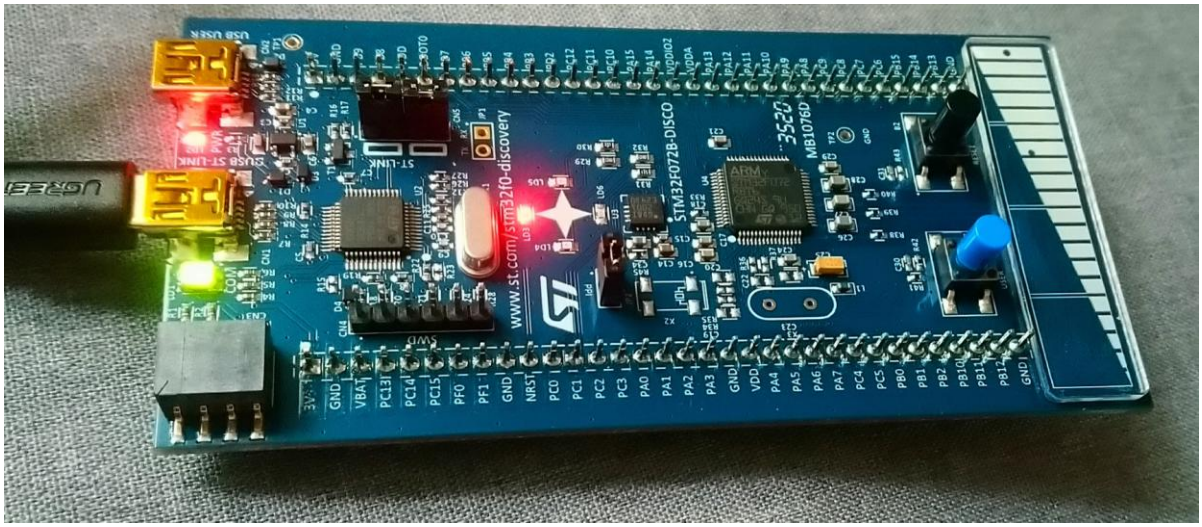
Rys. 4. Inicjalizacja zmiennych oraz ustawienie stałych

main.c | stm32f0xx_hal_adc.c | stm32f0xx_hal_adc_ex.c | stm32f0xx_hal.c | stm32f0xx_it.c

main()

```
145  int main(void)
146  {
147    /* USER CODE BEGIN 1 */
148    for(int i = 0; i < sizeOfTable; i++){
149      hist_data[i] = 0;
150    }
151    /* USER CODE END 1 */
152
153    /* MCU Configuration--------------------------------------------------------*/
154
155    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
156    HAL_Init();
157
158    /* USER CODE BEGIN Init */
159
160    /* USER CODE END Init */
161
162    /* Configure the system clock */
163    SystemClock_Config();
164
165    /* USER CODE BEGIN SysInit */
166
167    /* USER CODE END SysInit */
168
169    /* Initialize all configured peripherals */
170    MX_GPIO_Init();
171    MX_ADC_Init();
172    /* USER CODE BEGIN 2 */
173    //ADC_Enable(&hadc); // wlaczenie ADC
174    HAL_ADC_Start_IT(&hadc); // start AD
175    HAL_GPIO_WritePin(GPIOC, red_diode, GPIO_PIN_SET); /* zaswiecenie diody czerwonej*/
176    /* USER CODE END 2 */
177
178    /* Infinite loop */
179    /* USER CODE BEGIN WHILE */
```

Rys. 5. Zerowanie tablicy z wartościami odczytów, inicjalizacja przerwań oraz ADC, zapalenie czerwonej diody



Zdjęcie. 1. Zapalenie diody czerwonej

4. Wejście do pętli nieskończonej;

```
177
178   /* Infinite loop */
179   /* USER CODE BEGIN WHILE */
180   while (1)
181   {
182     /* USER CODE END WHILE */
183
184     /* USER CODE BEGIN 3 */
185     HAL_Delay(200); // opoznienie 200ms
186     HAL_ADC_Start_IT(&hadc); // start AD
187
188   }
189   /* USER CODE END 3 */
190 }
191
```

Rys. 6. Pętla nieskończona.

5. Po wystąpieniu przerwania od ADC, program wchodzi do funkcji obsługi odczytu;

```
90  // funkcja odpowiadajaca za odczyt z ADC
91  void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
92  {
93    if(__HAL_ADC_GET_FLAG(hadc, ADC_FLAG_EOC)){
94    // jesli program zostal wlaczony uzupelnij bufor cykliczny aktualnym pomiarem
95      if(first_run == 1){
96        for(int i = 0; i < sizeOfTable; i++){
97          hist_data[i] = HAL_ADC_GetValue(hadc);
98        }
99        first_run = 0;
100     }
101     // jesli osiagnieto koniec bufora, zeruj index
102     if(index == sizeOfTable){
103       index = 0;
104     }
105     // znajdz najstarszy element w buforze
106     int ADC_the_oldest;
107     if (index + 1 == sizeOfTable){
108       ADC_the_oldest = hist_data[0];
109     } else{
110       ADC_the_oldest = hist_data[index + 1];
111     }
112     // poprzedni odczyt
113     ADC_last = ADC_raw;
114     // odczyt z sensora
115     ADC_raw = HAL_ADC_GetValue(hadc);
116     hist_data[index] =  ADC_raw;
117     // printf("aktualny: %u ", ADC_raw);
118     // fflush(stdout);
119     int ADC_max = max_element();
120     int ADC_min = min_element();
121     // Czy roznica miedzy najwiekszym a najmniejszym elementem bufora jest
122     // wieksza nic zakladany blad odczytu?
123     if(fabs(ADC_max - ADC_min) > difference){
124       // czy wartosc rosnie czy maleje?
125       if(hist_data[index] - ADC_the_oldest > 0){
126         // printf("Maleje\n");
127         HAL_GPIO_WritePin(GPIOC, red_diode, GPIO_PIN_RESET); /* zgaszenie diody czerwonej*/
128         HAL_GPIO_WritePin(GPIOC, blue_diode, GPIO_PIN_SET); /* zaswiecenie diody niebieskiej*/
129       } else{
130         // printf("Rosnie\n");
131         HAL_GPIO_WritePin(GPIOC, red_diode, GPIO_PIN_SET); /* zaswiecenie diody czerwonej*/
132         HAL_GPIO_WritePin(GPIOC, blue_diode, GPIO_PIN_RESET); /* zgaszenie diody niebieskiej*/
133       }
134       // fflush(stdout);
135     }
136     index++;
137   }
138 }
139 /* USER CODE END 0 */
140
```

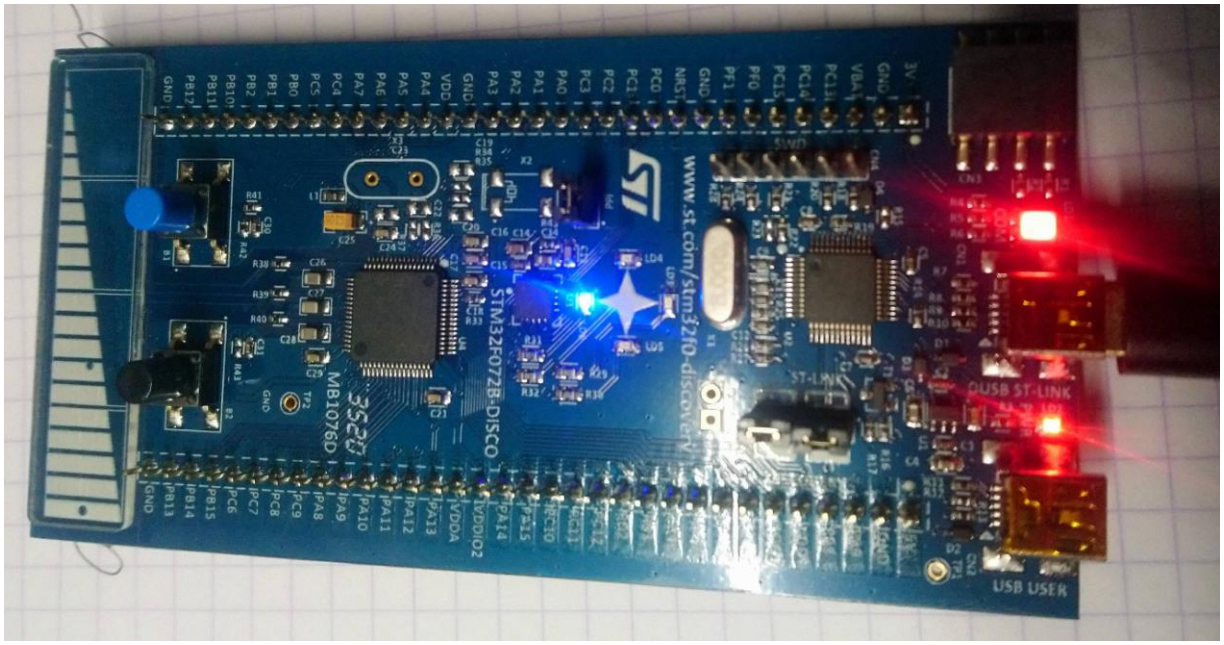
Rys. 7. Funkcja odpowiedzialna za odczyt z ADC

6. Program uzupełnia bufor cykliczny wartościami z aktualnego odczytu z ADC;
7. Sprawdza, czy osiągnięto koniec bufora cyklicznego;
8. Znajduje najstarszy element w buforze;
9. Zapisuje bieżący odczyt do bufora cyklicznego;
10. Pobiera wartości największą i najmniejszą z bufora, do dwóch zmiennych;

```
69  /* Private user code ---------------------------------------------
70  /* USER CODE BEGIN 0 */
71  int max_element(){ // znajdowanie najwiekszego elementu w buforze
72    int max = hist_data[0];
73    for(int i = 1; i < sizeOfTable; i++){
74      if(hist_data[i] > max){
75        max = hist_data[i];
76      }
77    }
78    return max;
79  }
80
81  int min_element(){ //znajdowanie najmniejszego elementu w buforze
82    int min = hist_data[0];
83    for(int i = 1; i < sizeOfTable; i++){
84      if(hist_data[i] < min){
85        min = hist_data[i];
86      }
87    }
88    return min;
89  }
```

Rys. 8. Funkcję do znajdowania największego i najmniejszego elementu w buforze.

11. Sprawdza, czy wartość bezwzględna z różnicy między nimi jest większa niż założony błąd pomiarowy;
12. Jeśli nie, zwiększa indeks bufora cyklicznego i wychodzi z obsługi ADC,
13. Jeśli tak, sprawdza, czy wartość maleje , czy rośnie, w zależności od tego zapala i gasi odpowiednią diodę.



Zdjęcie .2 .Zapalona dioda niebieska

4. Podsumowanie i wnioski:

Obsługa ADC otwiera możliwości wykorzystania sensorów zarówno wewnętrznych jak i zewnętrznych. Udostępnia duże możliwości konfiguracyjne. Dzięki łatwości konfiguracji w Cube umożliwia nawet niewprawnemu użytkownikowi zbudowanie skomplikowanych układów.