**Technika Mikroprocesorowa**

Kacper Szmajdel

Jakub Rajs

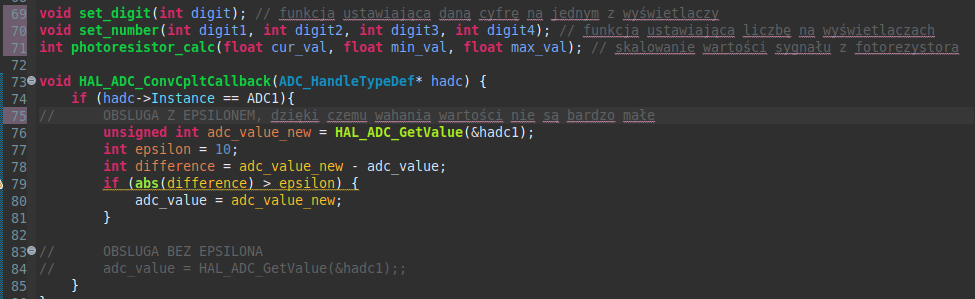
*STM32 - UART – fotorezystor*

# 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:



Kod 1. Przerwanie od zakończenia odbioru

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

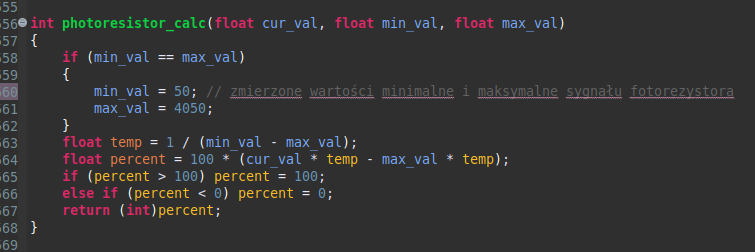
oraz trzech funkcjach wywoływanych w przerwaniach:



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

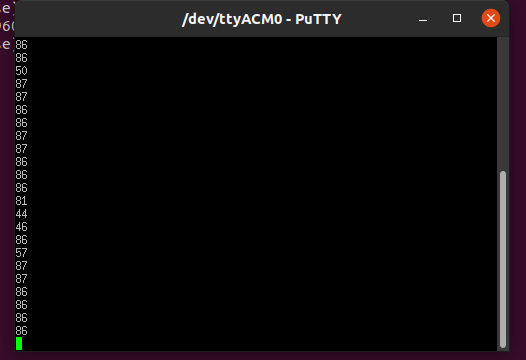


Kod 4. Funkcja ustawiająca liczby na całym wyświetlaczu



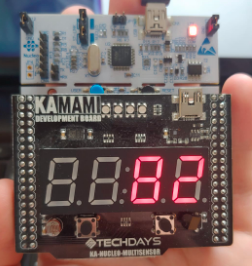
Kod 4. Funkcja przeliczająca wartości sygnału na procenty

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

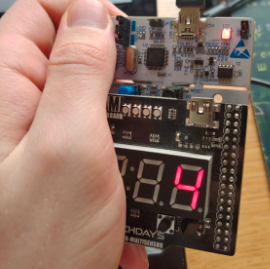


Rys. 1. Okno komunikacji z portem szeregowym, do którego podpięta jest płytka

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 UARTa w ramach poprzedniego laboratorium. Dzięki temu najwięcej pracy było wymagane przy przeliczaniu sygnałów oraz wyświetlaczach, a nie przy konfiguracji projektu.

# Kod wykorzystany do wykonania zadania:

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.

\* All rights reserved.</center></h2>

\*

\* This software component is licensed by ST under BSD 3-Clause license,

\* the "License"; You may not use this file except in compliance with the

\* License. You may obtain a copy of the License at:

\* opensource.org/licenses/BSD-3-Clause

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

ADC\_HandleTypeDef hadc1;

TIM\_HandleTypeDef htim9;

UART\_HandleTypeDef huart2;

DMA\_HandleTypeDef hdma\_usart2\_tx;

DMA\_HandleTypeDef hdma\_usart2\_rx;

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_TIM9\_Init(void);

static void MX\_DMA\_Init(void);

static void MX\_USART2\_UART\_Init(void);

static void MX\_ADC1\_Init(void);

/\* USER CODE BEGIN PFP \*/

#include "string.h"

unsigned int time = 200;

unsigned int com = 1;

unsigned int adc\_value = 0;

char uart\_TX[10];

void set\_digit(int digit); // funkcja ustawiająca daną cyfrę na jednym z wyświetlaczy

void set\_number(int digit1, int digit2, int digit3, int digit4); // funkcja ustawiająca liczbę na wyświetlaczach

int photoresistor\_calc(float cur\_val, float min\_val, float max\_val); // skalowanie wartości sygnału z fotorezystora

void HAL\_ADC\_ConvCpltCallback(ADC\_HandleTypeDef\* hadc) {

if (hadc->Instance == ADC1){

// OBSLUGA Z EPSILONEM, dzięki czemu wahania wartości nie są bardzo małe

unsigned int adc\_value\_new = HAL\_ADC\_GetValue(&hadc1);

int epsilon = 10;

int difference = adc\_value\_new - adc\_value;

if (abs(difference) > epsilon) {

adc\_value = adc\_value\_new;

}

// OBSLUGA BEZ EPSILONA

// adc\_value = HAL\_ADC\_GetValue(&hadc1);;

}

}

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++;

}

}

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_TIM9\_Init();

MX\_DMA\_Init();

MX\_USART2\_UART\_Init();

MX\_ADC1\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_GPIO\_WritePin(COM1\_GPIO\_Port,COM1\_Pin,GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(COM2\_GPIO\_Port,COM1\_Pin,GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(COM3\_GPIO\_Port,COM1\_Pin,GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(COM4\_GPIO\_Port,COM1\_Pin,GPIO\_PIN\_SET);

HAL\_TIM\_Base\_Start\_IT(&htim9);

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSI;

RCC\_OscInitStruct.PLL.PLLM = 8;

RCC\_OscInitStruct.PLL.PLLN = 100;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 4;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV2;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_3) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief ADC1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_ADC1\_Init(void)

{

/\* USER CODE BEGIN ADC1\_Init 0 \*/

/\* USER CODE END ADC1\_Init 0 \*/

ADC\_ChannelConfTypeDef sConfig = {0};

/\* USER CODE BEGIN ADC1\_Init 1 \*/

/\* USER CODE END ADC1\_Init 1 \*/

/\*\* Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)

\*/

hadc1.Instance = ADC1;

hadc1.Init.ClockPrescaler = ADC\_CLOCK\_SYNC\_PCLK\_DIV4;

hadc1.Init.Resolution = ADC\_RESOLUTION\_12B;

hadc1.Init.ScanConvMode = DISABLE;

hadc1.Init.ContinuousConvMode = ENABLE;

hadc1.Init.DiscontinuousConvMode = DISABLE;

hadc1.Init.ExternalTrigConvEdge = ADC\_EXTERNALTRIGCONVEDGE\_NONE;

hadc1.Init.ExternalTrigConv = ADC\_SOFTWARE\_START;

hadc1.Init.DataAlign = ADC\_DATAALIGN\_RIGHT;

hadc1.Init.NbrOfConversion = 1;

hadc1.Init.DMAContinuousRequests = DISABLE;

hadc1.Init.EOCSelection = ADC\_EOC\_SINGLE\_CONV;

if (HAL\_ADC\_Init(&hadc1) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.

\*/

sConfig.Channel = ADC\_CHANNEL\_1;

sConfig.Rank = 1;

sConfig.SamplingTime = ADC\_SAMPLETIME\_3CYCLES;

if (HAL\_ADC\_ConfigChannel(&hadc1, &sConfig) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN ADC1\_Init 2 \*/

/\* USER CODE END ADC1\_Init 2 \*/

}

/\*\*

\* @brief TIM9 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_TIM9\_Init(void)

{

/\* USER CODE BEGIN TIM9\_Init 0 \*/

/\* USER CODE END TIM9\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

/\* USER CODE BEGIN TIM9\_Init 1 \*/

/\* USER CODE END TIM9\_Init 1 \*/

htim9.Instance = TIM9;

htim9.Init.Prescaler = 500;

htim9.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim9.Init.Period = 1000;

htim9.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim9.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

if (HAL\_TIM\_Base\_Init(&htim9) != HAL\_OK)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

if (HAL\_TIM\_ConfigClockSource(&htim9, &sClockSourceConfig) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM9\_Init 2 \*/

/\* USER CODE END TIM9\_Init 2 \*/

}

/\*\*

\* @brief USART2 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_USART2\_UART\_Init(void)

{

/\* USER CODE BEGIN USART2\_Init 0 \*/

/\* USER CODE END USART2\_Init 0 \*/

/\* USER CODE BEGIN USART2\_Init 1 \*/

/\* USER CODE END USART2\_Init 1 \*/

huart2.Instance = USART2;

huart2.Init.BaudRate = 9600;

huart2.Init.WordLength = UART\_WORDLENGTH\_8B;

huart2.Init.StopBits = UART\_STOPBITS\_1;

huart2.Init.Parity = UART\_PARITY\_NONE;

huart2.Init.Mode = UART\_MODE\_TX\_RX;

huart2.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart2.Init.OverSampling = UART\_OVERSAMPLING\_8;

if (HAL\_UART\_Init(&huart2) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN USART2\_Init 2 \*/

/\* USER CODE END USART2\_Init 2 \*/

}

/\*\*

\* Enable DMA controller clock

\*/

static void MX\_DMA\_Init(void)

{

/\* DMA controller clock enable \*/

\_\_HAL\_RCC\_DMA1\_CLK\_ENABLE();

/\* DMA interrupt init \*/

/\* DMA1\_Stream5\_IRQn interrupt configuration \*/

HAL\_NVIC\_SetPriority(DMA1\_Stream5\_IRQn, 0, 0);

HAL\_NVIC\_EnableIRQ(DMA1\_Stream5\_IRQn);

/\* DMA1\_Stream6\_IRQn interrupt configuration \*/

HAL\_NVIC\_SetPriority(DMA1\_Stream6\_IRQn, 0, 0);

HAL\_NVIC\_EnableIRQ(DMA1\_Stream6\_IRQn);

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOC, SEG\_G\_Pin|SEG\_D\_Pin|SEG\_E\_Pin|SEG\_C\_Pin

|SEG\_B\_Pin|SEG\_F\_Pin|SEG\_A\_Pin|SEG\_DP\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOC, COM4\_Pin|COM3\_Pin|COM2\_Pin|COM1\_Pin, GPIO\_PIN\_SET);

/\*Configure GPIO pins : SEG\_G\_Pin SEG\_D\_Pin SEG\_E\_Pin SEG\_C\_Pin

SEG\_B\_Pin SEG\_F\_Pin SEG\_A\_Pin SEG\_DP\_Pin

COM4\_Pin COM3\_Pin COM2\_Pin COM1\_Pin \*/

GPIO\_InitStruct.Pin = SEG\_G\_Pin|SEG\_D\_Pin|SEG\_E\_Pin|SEG\_C\_Pin

|SEG\_B\_Pin|SEG\_F\_Pin|SEG\_A\_Pin|SEG\_DP\_Pin

|COM4\_Pin|COM3\_Pin|COM2\_Pin|COM1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOC, &GPIO\_InitStruct);

}

/\* USER CODE BEGIN 4 \*/

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);

HAL\_GPIO\_WritePin(SEG\_DP\_GPIO\_Port, SEG\_DP\_Pin, GPIO\_PIN\_RESET);

break;

case 2:

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\_RESET);

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\_RESET);

HAL\_GPIO\_WritePin(SEG\_G\_GPIO\_Port, SEG\_G\_Pin, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(SEG\_DP\_GPIO\_Port, SEG\_DP\_Pin, GPIO\_PIN\_RESET);

break;

case 3:

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\_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\_SET);

HAL\_GPIO\_WritePin(SEG\_DP\_GPIO\_Port, SEG\_DP\_Pin, GPIO\_PIN\_RESET);

break;

case 4:

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\_SET);

HAL\_GPIO\_WritePin(SEG\_G\_GPIO\_Port, SEG\_G\_Pin, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(SEG\_DP\_GPIO\_Port, SEG\_DP\_Pin, GPIO\_PIN\_RESET);

break;

case 5:

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\_RESET);

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\_RESET);

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\_SET);

HAL\_GPIO\_WritePin(SEG\_DP\_GPIO\_Port, SEG\_DP\_Pin, GPIO\_PIN\_RESET);

break;

case 6:

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\_RESET);

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\_SET);

HAL\_GPIO\_WritePin(SEG\_DP\_GPIO\_Port, SEG\_DP\_Pin, GPIO\_PIN\_RESET);

break;

case 7:

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\_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);

HAL\_GPIO\_WritePin(SEG\_DP\_GPIO\_Port, SEG\_DP\_Pin, GPIO\_PIN\_RESET);

break;

case 8:

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\_SET);

HAL\_GPIO\_WritePin(SEG\_DP\_GPIO\_Port, SEG\_DP\_Pin, GPIO\_PIN\_RESET);

break;

case 9:

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\_RESET);

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\_SET);

HAL\_GPIO\_WritePin(SEG\_DP\_GPIO\_Port, SEG\_DP\_Pin, GPIO\_PIN\_RESET);

break;

default:

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\_RESET);

HAL\_GPIO\_WritePin(SEG\_C\_GPIO\_Port, SEG\_C\_Pin, GPIO\_PIN\_RESET);

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);

HAL\_GPIO\_WritePin(SEG\_DP\_GPIO\_Port, SEG\_DP\_Pin, GPIO\_PIN\_RESET);

break;

}

}

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;

case 4:

// Wyswietlacz 4

HAL\_GPIO\_WritePin(COM3\_GPIO\_Port,COM3\_Pin,GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(COM4\_GPIO\_Port,COM4\_Pin,GPIO\_PIN\_RESET);

set\_digit(digit4);

com = 1;

break;

}

}

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;

}

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

\_\_disable\_irq();

while (1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

#endif /\* USE\_FULL\_ASSERT \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/