POLITECHNIKA
RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

Systemy mikroprocesorowe

Obsługa potencjometru analogowego

Bartosz Maciołek

Rzeszów 2023

Spis treści

1. Wprowadzenie

1.1 Cel projektu

Celem pracy było zaprojektowanie i wykonanie układu potencjometru i wyświetlacza 7-segmentowego 4-cyfrowego. Wyświetlacz miał za zadanie prezentować procentową wartość położenia potencjometru.

1.2 Założenia projektowe

Założeniem projektu było wykorzystanie wyświetlacza LED do prezentacji procentowego położenia potencjometru analogowego. Cały układ obsługiwany jest przy pomocy mikrokontrolera STM32 umieszczonego na płytce ewaluacyjnej NUCLEO-F103RB.

1.3 Zakres pracy

Zadaniem było zaprojektowanie układu elektronicznego, który będzie komunikować się z mikrokontrolerem STM32 NUCLEO-F103RB poprzez dobranie odpowiednich elementów oraz stworzenie fizycznego połączenia między nimi. Projektowanie obejmowało również zaprojektowanie układów elektronicznych, które zapewnią prawidłowe działanie całego systemu. Opracowano program, który kontroluje pracę elementów oraz mikrokontrolera zgodnie z wymaganiami. W końcowej fazie przeprowadzono testy układu, aby sprawdzić jego poprawność działania, a wyniki analizowano i komentowano.

2. Teoria

2.1 Środowisko programistyczne

Program został napisany przy użyciu środowiska System Workbench for STM32. Jest to darmowe, zintegrowane środowisko programistyczne (IDE) przeznaczone do programowania mikrokontrolerów STM32 firmy STMicroelectronics. SW4STM32 oferuje narzędzia do tworzenia, kompilacji, debugowania i testowania aplikacji na mikrokontrolery STM32.

SW4STM32 opiera się na otwartym oprogramowaniu Eclipse, co oznacza, że zawiera wiele popularnych funkcji IDE, takich jak edycja kodu, zarządzanie projektami, narzędzia budowania i debugowania, integrację z systemami kontroli wersji, a także wiele innych narzędzi.

SW4STM32 oferuje wiele przydatnych funkcji, takich jak:

- Wbudowany debugger: SW4STM32 zawiera wbudowany debugger, który umożliwia debugowanie aplikacji na poziomie kodu źródłowego oraz monitorowanie stanu mikrokontrolera w czasie rzeczywistym.
- Wsparcie dla różnych rodzajów projektów: SW4STM32 umożliwia tworzenie projektów zarówno dla mikrokontrolerów STM32F0, STM32F1, STM32F2, STM32F3, STM32F4, STM32F7, jak i dla mikrokontrolerów z serii STM32L.
- Biblioteki HAL i LL: SW4STM32 zawiera biblioteki HAL (Hardware Abstraction Layer) i LL (Low Level), które ułatwiają programowanie mikrokontrolerów STM32, zapewniając dostęp do sprzętu na różnych poziomach abstrakcji.

2.2 Elementy układu – podstawowe dane techniczne i noty katalogowe

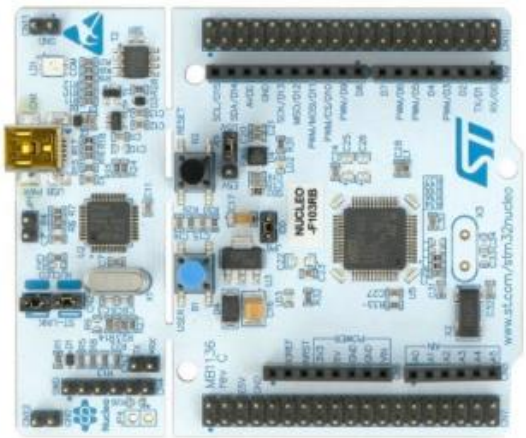
2.2.1 Płytka ewaluacyjna NUCLEO-F103RB

Startowy zestaw NUCLEO-F103RB z serii STM32 Nucleo-64 jest wyposażony w mikrokontroler STM32F103RBT6 w obudowie LQFP64. Dzięki wbudowanemu programatorowi/debuggerowi ST-Link/v2, programowanie i debugowanie układu jest szybkie i łatwe. ST-Link może także działać jako samodzielny programator po odłączeniu go od reszty modułu przez zworki, a specjalne nacięcia pozwalają na oddzielenie programatora od reszty modułu. Moduł charakteryzuje się elastycznym sposobem zasilania, co pozwala na wygodne korzystanie z płytki deweloperskiej w aplikacjach testowych.

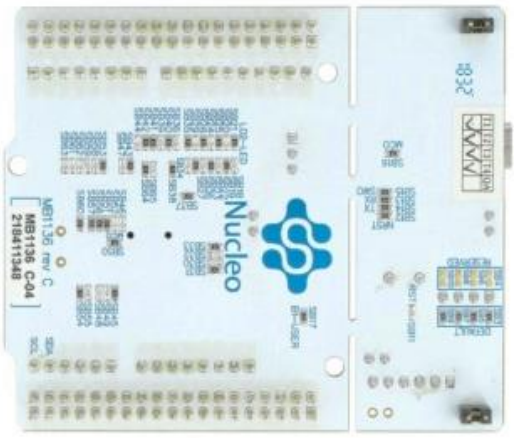
Dostępne dwa rodzaje złącz: Arduino Uno R3 i ST Morpho umożliwiają dostęp do każdego portu mikrokontrolera oraz umożliwiają podłączenie shieldów kompatybilnych z Arduino.

Dane Techniczne:

- Mikrokontroler: STM32F103RBT6
  - Rdzeń: ARM Cortex M3 32-bit
  - Częstotliwość taktowania: 72 MHz
  - Pamięć programu Flash: 128 kB
  - Pamięć SRAM: 20 kB
  - x przetwornik analogowo-cyfrowy: 12-bitowy, 16-kanałowy
  - Ilość Timerów: 7
  - Interfejsy: 3x USART, 2x SPI 18Mbit/s, 2x I2C, USB Full Speed, CAN 2,0B
- Dwa typy złącz:
  - Złącza dla nakładek kompatybilnych z Arduino Uno Rev3
  - Standardowe piny STMicroelectronics Morpho, umożliwiające dostęp do wyprowadzeń mikrokontrolera
- Wbudowane trzy diody LED
- Dwa przyciski
- Moduł wspierany przez większość popularnych środowisk, m.in: IAR, Keil oraz platformy oparte na kompilatorze GCC



Rys. 2.1 NUCLEO-F103RB front



Rys. 2.2 NUCLEO-F103RB tył

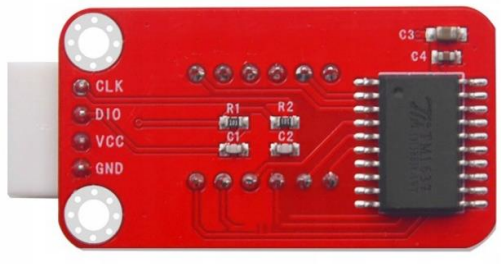
2.2.2 Czterocyfrowy wyświetlacz siedmiosegmentowy Keyes TM1637

Moduł z 4-cyfrowym wyświetlaczem 7-segmentowym z cyframi w kolorze czerwonym. Płytka została wyposażona w sterownik TM1637 i komunikuje się przez interfejs cyfrowy, dzięki czemu za pomocą dwóch pinów można sterować całym wyświetlaczem. Moduł ma regulację jasności i wyposażony został w 4-pinowe złącze XH2.54 oraz 2 otwory ułatwiające montaż w urządzeniu docelowym. Sprawdzi się w projektach wymagających wyświetlania godziny, temperatury czy krótkich informacji.

Dane techniczne:

- Ilość cyfr: 4
- 7-segmentowy
- Podświetlenie LED
- Typ: wspólna anoda
- Sterownik TM1637
- Interfejs cyfrowy (DIO, CLK)
- Kolor: czerwony
- Przekątna: 0,36"
- Zasilanie: od 3,3 V do 5 V
- Pobór prądu: od 30 do 80 mA
- Wymiary: 42 x 24 x 11 mm
- 2 otwory montażowe

Rys. 2.3 TM1637 front



Rys. 2.4 TM1637 tył

Dane otrzymane z urządzenia zewnętrznego przez interfejs szeregowy są przechowywane w rejestrze. Rejestr wyświetlaczy TM1637 ma adresy w zakresie od 00H do 05H i może pomieścić 6 bajtów, które kontrolują wyświetlanie zawartości segmentów (wyprowadzenia SEG) poszczególnych wyświetlaczy. Zapis danych powinien odbywać się zgodnie z kolejnością adresów wyświetlaczy, zaczynając od najniższego i kończąc na najwyższym.

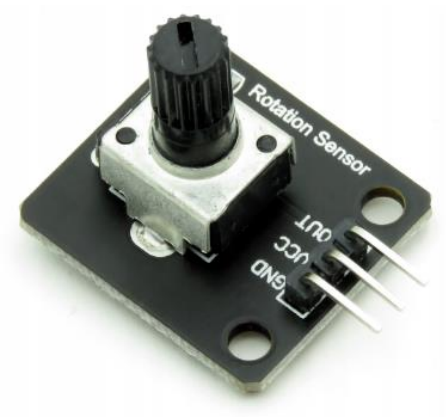
Aby przesłać dane z mikrokontrolera interfejsem dwuprzewodowym, należy odczekać, aż linia zegarowa (pin CLK) będzie w stanie niskim. Podczas gdy linia CLK jest ustawiona na stan wysoki, nie może nastąpić zmiana stanu logicznego na pinie DIO. Jeśli transmisja danych jest poprawna, układ generuje sygnał ACK (po zboczu opadającym 8 taktu zegara) na pinie DIO (stan niski), potwierdzając transmisję. Wysłanie sygnału na linię DIO następuje po zakończeniu dziewiątego taktu zegara.

2.2.3 Rotary Angle Sensor - moduł z potencjometrem

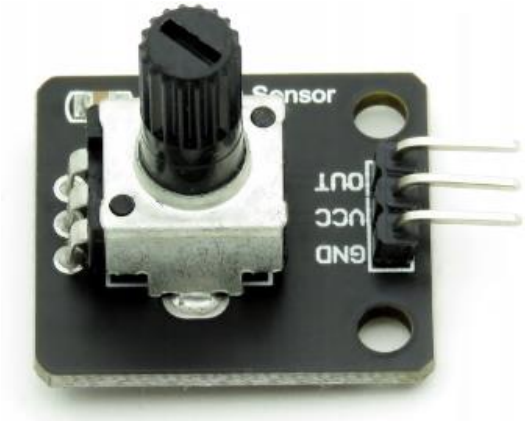
Moduł z potencjometrem o zakresie obrotu 280°. Umożliwia początkującym użytkownikom zapoznanie się z zasadą działania czujników analogowych. Ułatwia stworzenie obwodu sterowania np. oświetleniem lub głośnością za pomocą czujnika obrotowego. Kompatybilny z systemami 3,3 V oraz 5 V, ma wyjście analogowe o zakresie od 0 do VCC.

Dane techniczne:

- Interfejs analogowy
- Zakres obrotu 280°
- Zasilanie: od 3,3 V do 5 V
- Wymiary: 22 x 31 mm
- Rezystancja: od 0Ω do 10kΩ



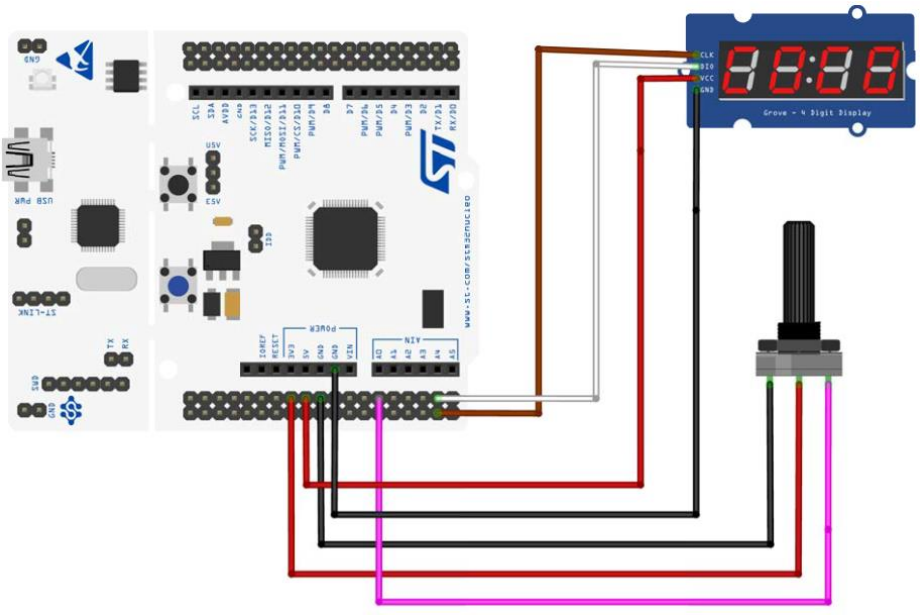
Rys. 2.5 Rotary Sensor(1)



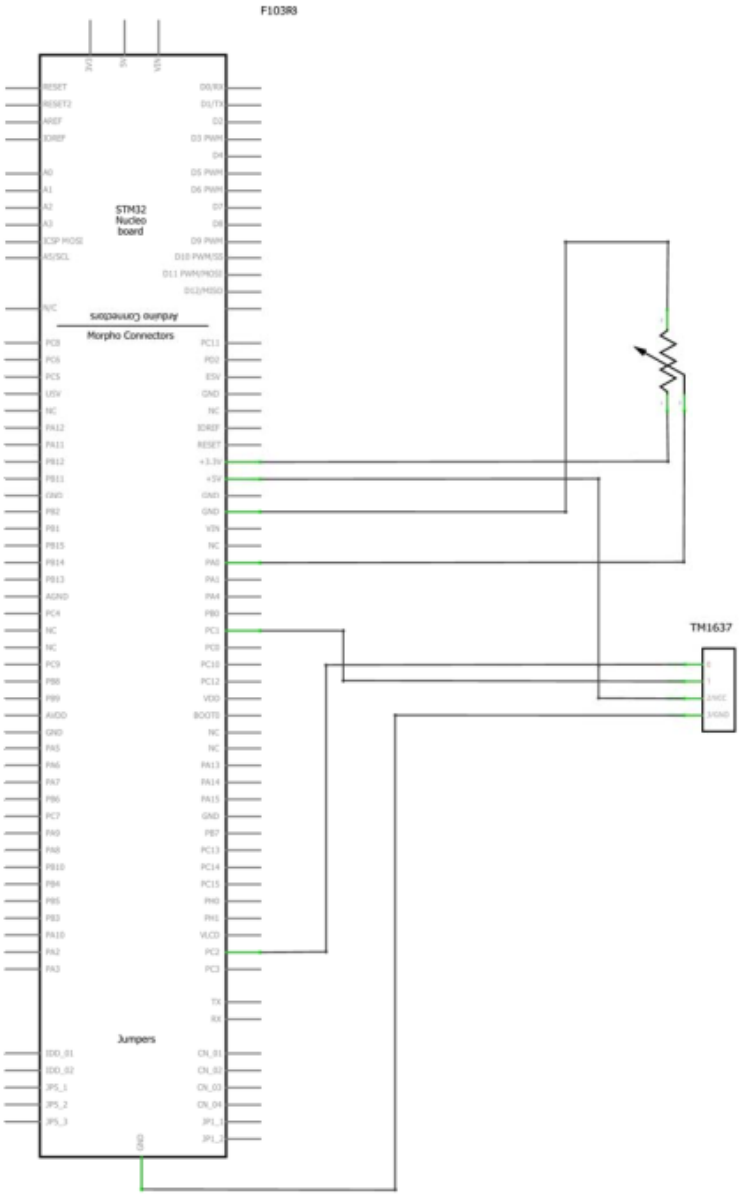
Rys. 2.6 Rotary Sensor(2)

3. Realizacja praktyczna

3.1 Schemat układu



Rys. 3.1 Obrazowy schemat układu



Rys. 3.2 Schemat układu

3.2 Przetwornik ADC – obliczenia

Mikrokontroler STM32F103RB posiada dwa przetworniki analogowo-cyfrowe (ADC) o rozdzielczości 12 bitów. Każdy z tych przetworników posiada multiplekser na wejściu, co umożliwia odczyt danych z maksymalnie 16 różnych linii wejściowych.

W dodatku, mikrokontroler STM32F103RB posiada wbudowany czujnik temperatury oraz źródło napięcia referencyjnego o wartości 1,2V. Czujnik temperatury jest mniej dokładny i pozwala na pomiar jedynie zmian temperatury, podczas gdy napięcie referencyjne może być użyte do przetestowania metody odczytu napięcia z wykorzystaniem przetwornika analogowo-cyfrowego.

Interpretacja wyniku pomiaru wykonanego względem napięcia zasilania 3.3V zwracanej przez przetwornik polega na odczycie wartości z zakresu do 0 do 4095.Dla napięcia na wejściu przetwornika VADC funkcja wynikowa będzie wyglądała następująco:

$$Wartość = \frac{VADC * 4096}{3{,}3V}$$

Naszym zadaniem jest przekonwertowanie wartości zwracanej przez przetwornik na wielkość od 0 do 100. W tym celu przygotowujemy odpowiednią funkcję:

$$Wielkość = 100 - \frac{VADC * 100}{4095}$$

3.3 Program

W celu wyświetlenia wartości na wyświetlaczu TM1637, importujemy obsługującą go bibliotekę `stm32_tm1637.c`.

```
/*
 * stm32_tm1637.c
 *
 *  Created on: 23.04.2023
 *      Author: barto
 */

#include "stm32f1xx_hal.h"

#include "stm32_tm1637.h"


void _tm1637Start(void);
void _tm1637Stop(void);
void _tm1637ReadResult(void);
void _tm1637WriteByte(unsigned char b);
void _tm1637DelayUsec(unsigned int i);
void _tm1637ClkHigh(void);
void _tm1637ClkLow(void);
void _tm1637DioHigh(void);
void _tm1637DioLow(void);


#define CLK_PORT GPIOC
#define DIO_PORT GPIOC
#define CLK_PIN GPIO_PIN_2
#define DIO_PIN GPIO_PIN_1
#define CLK_PORT_CLK_ENABLE __HAL_RCC_GPIOC_CLK_ENABLE
#define DIO_PORT_CLK_ENABLE __HAL_RCC_GPIOC_CLK_ENABLE


const char segmentMap[] = {
    0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, // 0-7
    0x7f, 0x6f, 0x77, 0x7c, 0x39, 0x5e, 0x79, 0x71, // 8-9, A-F
    0x00
};


void tm1637Init(void)
```

```
{
    CLK_PORT_CLK_ENABLE();
    DIO_PORT_CLK_ENABLE();
    GPIO_InitTypeDef g = {0};
    g.Pull = GPIO_PULLUP;
    g.Mode = GPIO_MODE_OUTPUT_OD;
    g.Speed = GPIO_SPEED_FREQ_HIGH;
    g.Pin = CLK_PIN;
    HAL_GPIO_Init(CLK_PORT, &g);
    g.Pin = DIO_PIN;
    HAL_GPIO_Init(DIO_PORT, &g);

    tm1637SetBrightness(8);
}

void tm1637DisplayDecimal(int v, int displaySeparator)
{
    unsigned char digitArr[4];
    for (int i = 0; i < 4; ++i) {
        digitArr[i] = segmentMap[v % 10];
        if (i == 2 && displaySeparator) {
            digitArr[i] |= 1 << 7;
        }
        v /= 10;
    }

    _tm1637Start();
    _tm1637WriteByte(0x40);
    _tm1637ReadResult();
    _tm1637Stop();

    _tm1637Start();
    _tm1637WriteByte(0xc0);
    _tm1637ReadResult();

    for (int i = 0; i < 4; ++i) {
        _tm1637WriteByte(digitArr[3 - i]);
        _tm1637ReadResult();
    }

    _tm1637Stop();
}


void tm1637SetBrightness(char brightness)
{
    // Brightness command:
    // 1000 0XXX = display off
    // 1000 1BBB = display on, brightness 0-7
    // X = don't care
    // B = brightness
    _tm1637Start();
    _tm1637WriteByte(0x87 + brightness);
    _tm1637ReadResult();
    _tm1637Stop();
}

void _tm1637Start(void)
```

```
{
    _tm1637ClkHigh();
    _tm1637DioHigh();
    _tm1637DelayUsec(2);
    _tm1637DioLow();
}

void _tm1637Stop(void)
{
    _tm1637ClkLow();
    _tm1637DelayUsec(2);
    _tm1637DioLow();
    _tm1637DelayUsec(2);
    _tm1637ClkHigh();
    _tm1637DelayUsec(2);
    _tm1637DioHigh();
}

void _tm1637ReadResult(void)
{
    _tm1637ClkLow();
    _tm1637DelayUsec(5);
    _tm1637ClkHigh();
    _tm1637DelayUsec(2);
    _tm1637ClkLow();
}

void _tm1637WriteByte(unsigned char b)
{
    for (int i = 0; i < 8; ++i) {
        _tm1637ClkLow();
        if (b & 0x01) {
            _tm1637DioHigh();
        }
        else {
            _tm1637DioLow();
        }
        _tm1637DelayUsec(3);
        b >>= 1;
        _tm1637ClkHigh();
        _tm1637DelayUsec(3);
    }
}

void _tm1637DelayUsec(unsigned int i)
{
    for (; i>0; i--) {
        for (int j = 0; j < 10; ++j) {
            __asm__ __volatile__("nop\n\t":::"memory");
        }
    }
}

void _tm1637ClkHigh(void)
{
    HAL_GPIO_WritePin(CLK_PORT, CLK_PIN, GPIO_PIN_SET);
}
```

```
void _tm1637ClkLow(void)
{
    HAL_GPIO_WritePin(CLK_PORT, CLK_PIN, GPIO_PIN_RESET);
}

void _tm1637DioHigh(void)
{
    HAL_GPIO_WritePin(DIO_PORT, DIO_PIN, GPIO_PIN_SET);
}

void _tm1637DioLow(void)
{
    HAL_GPIO_WritePin(DIO_PORT, DIO_PIN, GPIO_PIN_RESET);
}
```

Funkcje zawarte w pliku `stm32_tm1637.c` umożliwią wyświetlanie na ekranie wyświetlacza TM1637 wartości z zakresu od 0 do 9999, włączenie lub wyłączenie dwukropka oddzielającego liczby oraz obsługą podświetlenia z zakresu od 0 do 8 (0 – podświetlenie wyłączone).

Plik `main.c`:

```
 * main.c
 *
 *  Created on: 23.04.2023
 *      Author: barto
 */


#include "stm32f1xx_nucleo.h"
#include <string.h>
#include "stm32f1xx.h"

UART_HandleTypeDef uart;

void send_char(char c)
{
      HAL_UART_Transmit(&uart, (uint8_t*)&c, 1, 1000);
}

int __io_putchar(int ch)
{
      if (ch == '\n')
            send_char('\r');
      send_char(ch);
      return ch;
}

int main(void)
{
      SystemCoreClock = 8000000;
      HAL_Init();

      __HAL_RCC_GPIOA_CLK_ENABLE();
      __HAL_RCC_USART2_CLK_ENABLE();
```

```
    __HAL_RCC_ADC1_CLK_ENABLE();
    __HAL_RCC_GPIOC_CLK_ENABLE();


    GPIO_InitTypeDef gpio;
    gpio.Mode = GPIO_MODE_AF_PP;
    gpio.Pin = GPIO_PIN_2;
    gpio.Pull = GPIO_NOPULL;
    gpio.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOA, &gpio);

    gpio.Mode = GPIO_MODE_AF_INPUT;
    gpio.Pin = GPIO_PIN_3;
    HAL_GPIO_Init(GPIOA, &gpio);

    gpio.Mode = GPIO_MODE_ANALOG;
    gpio.Pin = GPIO_PIN_0;
    HAL_GPIO_Init(GPIOA, &gpio);

    uart.Instance = USART2;
    uart.Init.BaudRate = 115200;
    uart.Init.WordLength = UART_WORDLENGTH_8B;
    uart.Init.Parity = UART_PARITY_NONE;
    uart.Init.StopBits = UART_STOPBITS_1;
    uart.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    uart.Init.OverSampling = UART_OVERSAMPLING_16;
    uart.Init.Mode = UART_MODE_TX_RX;
    HAL_UART_Init(&uart);

    RCC_PeriphCLKInitTypeDef adc_clk;
    adc_clk.PeriphClockSelection = RCC_PERIPHCLK_ADC;
    adc_clk.AdcClockSelection = RCC_ADCPCLK2_DIV2;
    HAL_RCCEx_PeriphCLKConfig(&adc_clk);

    ADC_HandleTypeDef adc;
    adc.Instance = ADC1;
    adc.Init.ContinuousConvMode = ENABLE;
    adc.Init.ExternalTrigConv = ADC_SOFTWARE_START;
    adc.Init.DataAlign = ADC_DATAALIGN_RIGHT;
    adc.Init.ScanConvMode = ADC_SCAN_DISABLE;
    adc.Init.NbrOfConversion = 1;
    adc.Init.DiscontinuousConvMode = DISABLE;
    adc.Init.NbrOfDiscConversion = 1;
    HAL_ADC_Init(&adc);

    HAL_ADCEx_Calibration_Start(&adc);

    ADC_ChannelConfTypeDef adc_ch;
    adc_ch.Channel = ADC_CHANNEL_0;
    adc_ch.Rank = ADC_REGULAR_RANK_1;
    adc_ch.SamplingTime = ADC_SAMPLETIME_13CYCLES_5;
    HAL_ADC_ConfigChannel(&adc, &adc_ch);

    HAL_ADC_Start(&adc);

    tm1637Init();
    tm1637SetBrightness(1);
    tm1637DisplayDecimal(2123, 0);
```

```
	while (1)
	{
		uint32_t value = HAL_ADC_GetValue(&adc);
		printf("%ld\n", value);
		tm1637DisplayDecimal(100-value*100/4095, 0);
	}
}
```

4. Podsumowanie

Podsumowując, w projekcie zakres prac obejmował zaprojektowanie układu elektronicznego z wykorzystaniem odpowiednich elementów, które będą komunikować się z mikrokontrolerem STM32 NUCLEO-F103RB. Projektowanie obejmowało również zaprojektowanie układów elektronicznych, które zapewnią prawidłowe działanie całego systemu. Opracowano program, który kontroluje pracę elementów oraz mikrokontrolera zgodnie z wymaganiami. W końcowej fazie projektu przeprowadzono testy układu, aby zweryfikować jego poprawność działania.

5. Bibliografia

[1] https://download.kamami.pl/p212017-Nucleo-UM1724_manual.pdf

[2] https://kaktusa.pl/tm1637-w-module-z-wyswietlaczem-led-7seg-4-cyfry/

[3] https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/docs/datasheet/unit/digi_clock/TM1637.pdf

[4] https://kamami.pl/en/position-sensors/581947-rotary-angle-sensor-module-with-potentiometer.html

[5] https://kamami.pl/stm-nucleo-64/212017-nucleo-f103rb-zestaw-startowy-z-mikrokontrolerem-z-rodziny-stm32-stm32f103.html