**AKADEMIA GÓRNICZO-HUTNICZA**

Raport z projektu

# Sterowanie serwomechanizmem z wykorzystaniem ADC, PWM, NRF24L01+ oraz STM32FE411RE

z przedmiotu

**Standardy i Systemy Komunikacyjne**

Systemy Wbudowane, rok I studiów magisterskich

*Jakub Górnisiewicz*

*Marcin Maj*

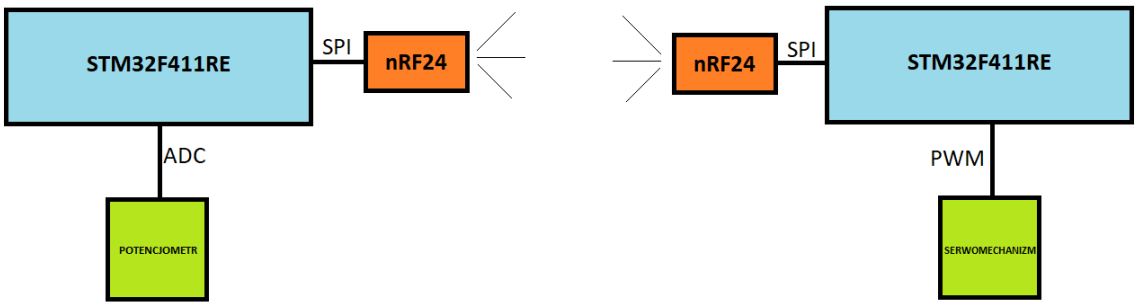
# Spis treści

# 1. Opis projektu

Celem projektu jest zrealizowanie funkcji zdalnego sterowania serwomechanizmem za pomocą pokrętła jakiś jest potencjometr. W tym celu wykorzystano dwie płytki ewaluacyjne wraz z ADC (ang. *Analog To Digital Converter*), PWM (ang. *Pulse-Width Modulation*). Aby zapewnić komunikacje bezprzewodową między Nadajnikiem oraz Odbiornikiem wykorzystano moduły radiowe nRF24L01+ oraz standard komunikacyjny SPI (ang. *Serial Peripheral Interface*).

**Opis funkcjonalności:**

Zmieniając wartość potencjometru na płytce nadawczej (Nadajnik) użytkownik jest w stanie wpływać na zmiany wychylenia orczyka serwomechanizmu w Odbiorniku.




*Rysunek 1. Schemat ideowy projektu*

# 2. HARDWARE

## Płytka ewaluacyjna

Wybranym rozwiązaniem jest płytka ewaluacyjna STM32 NUCLEO-F411RE, przedstawiona na Rysunku 2. Jest to zestaw uruchomieniowy z 32 bitowym mikrokontrolerem, posiadającym rdzeń ARM Cortex M4, pozostałe podstawowe parametry komponentu przedstawia Tabela 1.

*Rysunek 2. STM32 NUCLEO-F411RE*

*Tabela 1. Parametry STM32F411RE*

| Parametr | Wartość |
|---|---|
| Maksymalna częstotliwość taktowania | 100 MHz |
| Pamięć programu Flash | 512 kB |
| Pamięć SRAM | 128 kB |
| Ilość timerów | 10 |
| Przetwornik analogowo-cyfrowy | 12 bit, 16 kanałów |
| Interfejsy komunikacyjne | 3x I2C, 3x USART, 5x SPI |

## Serwomechanizm

Wykorzystany w projekcie silniczek to Serwo Okystar SG-90 - micro - 180°, Rysunek 2.

*Rysunek 3. Serwo Okystar SG-90 - micro - 180°.*

Jest on zasilany napięciem z zakresu 3.5 V do 6 V. W projekcie wybrano napięcie 5 V. Sterowanie odbywa się sygnałem PWM, o częstotliwości 50 Hz.

*Tabela 2. Parametry Serwo Okystar*

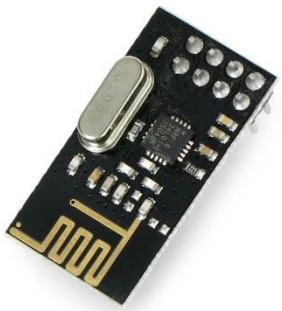
| Parametr | Wartość |
|---|---|
| Prędkość bez obciążenia | 0,12 s/60° przy 4,8 V |
| Kąt obrotu | 180° |
| Wymiary | 23 x 12,2 x 29 mm |
| Masa | 9 g |

### nRF24L01+

Wybrany element odpowiedzialny za komunikacyjne bezprzewodową jest nRF24L01+. Został on wybrany, ponieważ jest łatwy w obsłudze oraz znacznie tańszy niż np. Bluetooth.

Dodatkowo:

- pobierany prąd jest stosunkowo niski i wynosi około 11 mA (przy 0 dbm mocy wejściowej),
- prędkość transmisji danych jest wystarczająca jak na potrzeby projektu (komponent pozwala uzyskać przepustowość do 2 Mbps).



*Rysunek 4. nRF24L01+*

## 3. Wykorzystywane peryferia

### SPI

SPI (ang. Serial Peripherial Interface) jest to szeregowy interfejs komunikacyjny, posiada 4 porty: CS, SCLK, MOSI, MISO. Nie posiada adresowania. Jest „luźno" ustandaryzowany, to znaczy, że: nie ma określonych poziomów napięć, formatu ramki, ilości transmitowanych bitów, nie ma określonej kolejności bitów (Little endian/Big endian).

### ADC

ADC (ang. Analog to Digital Converter) to przetwornik przetwarzający sygnał analogowy na sygnał cyfrowy. W projekcie wykorzystano 12 bitowy przetwornik z napięciem referencyjnym 3.3 V, dzięki czemu zakres odczytywanych wartości analogowych wynosi od 0 do 3.3 V z krokiem 0.8 mV.

### PWM

PWM (ang. Pulse Width Modulation) jest to metoda regulacji sygnału o stałej amplitudzie i częstotliwości polegająca na zmianie wypełnienia sygnału. Metoda ta jest często stosowana do sterowania silnikami ponieważ silniki nie są aż tak wrażliwe na zakłócenia powodowane częstymi przełączeniami elementów CMOS.

Zegar na płytce ewaluacyjnej STM32F411RE doprowadzony do zegara TIM wynosi 84 MHz, z tego powodu należało wprowadzić odpowiednie ustawienia tak aby częstotliwość taktowania była zgodna z wymaganiami serwomechanizmu. W tym celu ustawiono prescaler na wartość 83 (realnie 84), aby uzyskać częstotliwość równą 1 MHz. Następnie licznik zliczający w górę na wartość 19999.

# 4. SOFTWARE

## Nadajnik (TX)

W kodzie zaimplementowano funkcje pobierającą dane z ADC o nazwie **get_adc_value()**. Wartości które można wysłać są z przedziału 0 do 4095.

```
uint32_t get_adc_value(void)
{
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
    uint32_t value_adc = HAL_ADC_GetValue(&hadc1);
    float real_voltage = (3.3f * value_adc / 4096.0f);
#ifdef DEBUG_PRINTF
    printf("ADC = %lu (%.3f V)\n", value_adc, real_voltage);
#endif
    return value_adc;
}
```

W pętli głównej programu odczytana wartość jest wysyłana za pomocą funkcji **nRF24_WriteTXPayload()**

```
while (1)
  {
          readed_adc_value = get_adc_value();
          MessageLength = sprintf(Message, "%d", readed_adc_value);
          nRF24_WriteTXPayload(Message);
#ifdef DEBUG_PRINTF
          printf("Message: %d, atoi(Message): %d, readed_adc_value: %d\n\r",
Message, atoi(Message), readed_adc_value);
#endif
          HAL_Delay(1);
          nRF24_WaitTX();
          HAL_Delay(1000);
  }
```

## Odbiornik (RX)

```
while (1)
  {
      if(nRF24_RXAvailible())
      {
          nRF24_ReadRXPaylaod(Nrf24_Message);
          char *endptr;
                recived_value = strtol((char *)Nrf24_Message, &endptr, 10);
#ifdef DEBUG_PRINTF
                if (*endptr == '\0')
                {
                    printf("Udalo sie odebrac: %i\n\r", recived_value);
                }
                else
                {
                    printf("Error: %s\n\r", (char *)Nrf24_Message);
                }
#endif DEBUG_PRINTF
                /*----------------SERWO---------------
Czestowliowsc bazowa PWM 50Hz = okres podstawowy 20ms
 Czas trwania impulsu od 1ms do 2ms

1000us = 1ms - minimalne wychylenie (0    st.)
2000us = 2ms - maksymlane wychylenie (90  st.)
                  ------------------------------------------------
W celu wypionowania smigla mozna wprowadzic offset, tak aby smiglo znajdowalo
sie w kacie 0st.
np:
100us = 0.1ms
200us = 0.2ms
                  ------------------------------------------------
Wartosci, które można otrzymac z ADC       |       Wartosci dla PWM
                    0     (min)            |               1000        (0st.)
                    4095  (max)            |               2000        (90st.)
Z tego powodu nalezy dokonac konwersji danych. Wartosci nalezy podzielic przez
4.095 oraz dodac wartoisc 1000.
                  ----------------------------------------*/

                PWM = 1000 + (int)(recived_value / 4.095);      //konwersja
#ifdef DEBUG_PRINTF
                printf("recived_value: %d,  PWM = %d\n\r", recived_value,
PWM);     //weryfikacja konwersji
#endif
                  __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1, PWM);
                  HAL_Delay(100);
      }
  }
```

W odbiorniku za pomocą funkcji **nRF24_ReadRXPaylaod()** oraz **strtol()** odebrano wartości odczytane z ADC. Funkcja strtol i jej argumenty służą do konwersji znaków ASCII do inta, gdyż nRF właśnie taką formę danych przesyła.

Następnie dokonywana jest konwersja wartości z ADC do wartości PWM. Tutaj pojawia się problem gdyż próbek z ADC może być aż 4096 natomiast wartości PWM tylko 1000 (jedna dla każdej 1 us wypełnienia PWM). Z tego powodu część poziomów ADC jest gubiona co wynika z dzielenia i zaokrąglania.

## 5. Podsumowanie

Cały projekt działa zgodnie z założeniami. Udało się zweryfikować poprawność funkcjonowania z wynikiem pozytywnym.

Projekt mimo „prostoty” działania wymagał dosyć czasochłonnego zgłębienia tematów takich jak SPI, ADC czy PWM.