Projekt

Sterowniki robotów

Dokumentacja

Oscyloskop z pomiarem parametrów sygnału OSPA

Skład grupy:
Krzysztof Kaliszuk, 248953
Krystian Mirek, 242053

Termin: czwTP19

Prowadzący:
dr inż. Wojciech Domski

14 czerwca 2021

Spis treści

1 Opis projektu

Projekt zakłada stworzenie oscyloskopu na płytce STM32F429I-DISC1. Mierzone sygnały będą pokazane na wyświetlaczu wraz z dodatkowymi informacjami o sygnale. Obejmuje to pomiar napięcia peak-to-peak, częstotliwości oraz podziałek. Wykorzystanie (ADC, DMA, Flash) [1] Oscyloskop z pomiarem parametrów sygnału
Założone funkcjonalności i wymagania dla oscyloskopu:

- generowanie sygnałów testowych,
- bufor zapisujący próbki badanego sygnału,
- funkcja pomiaru napięcia (Vpp),
- funkcja pomiaru częstotliwości i okresu (FFT),
- regulacja progu wyzwalania,
- funkcja stop,
- interfejs graficzny menu,
- wyświetlanie przebiegu sygnału na LCD.

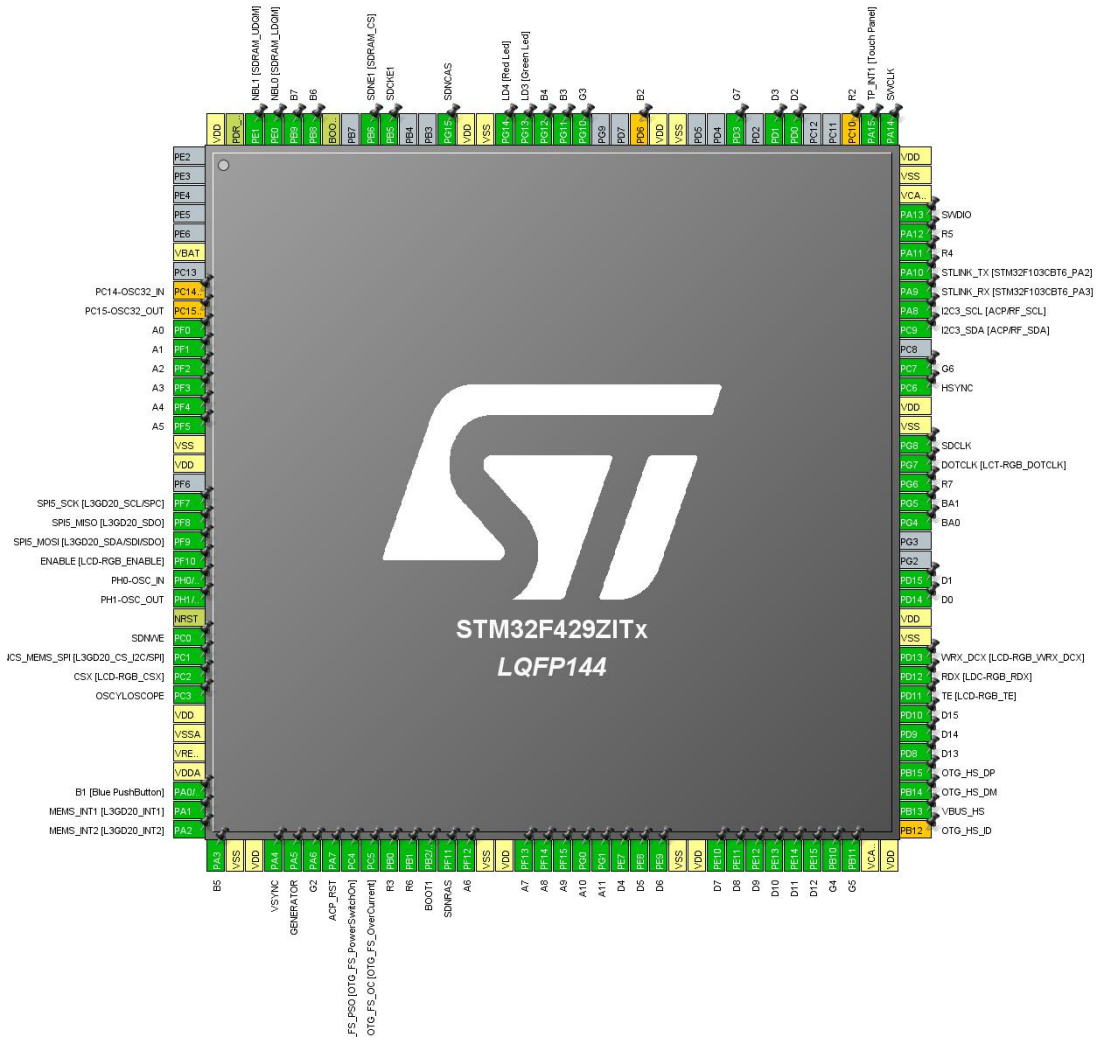
Określiliśmy najważniejsze momenty rozwoju projektu. Zbierając je razem wyklarowaliśmy 3 kamienie milowe rozwoju projektu:

- Wizualizacja odczytu sygnału,
- poprawne obliczenie parametrów sygnału,
- przetestowanie wszystkich funkcji.

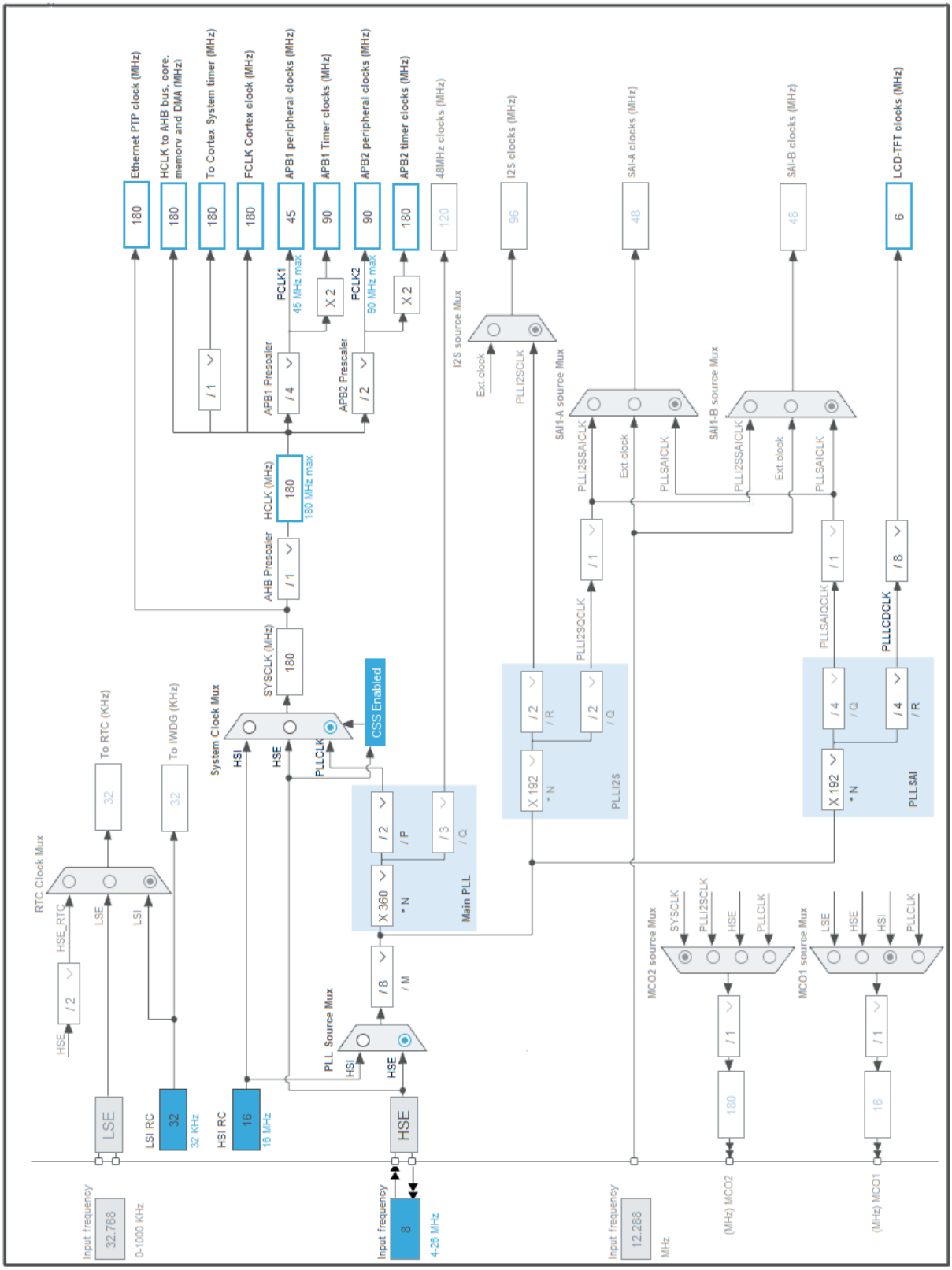
2 Konfiguracja mikrokontrolera

W projekcie zostanie wykorzystane ADC [4] wraz DMA [2] do odczytu analizowanego sygnału. Komunikacja z wyświetlaczem odbywa się z pomocą interfejsu SPI.

2. Pinout Configuration



Rysunek 1: Konfiguracja pinów mikrokontrolera



Rysunek 2: Konfiguracja zegarów

2.1 Konfiguracja pinów

W poniższej tabeli znajdują się tylko skonfigurowane lub zmodyfikowane piny. Wszystkie pozostałe zostały pobrane z domyślnej konfiguracji płytki rozwojowej.

Numer pinu	PIN	Tryb pracy	Funkcja/etykieta
2	PC3	ADC1_IN13	OSCYLOSCOPE
3	PA5	DAC_OUT2	GENERATOR

Tabela 1: Konfiguracja pinów mikrokontrolera

2.2 ADC

DMA request	Stream	Direction	Priority
ADC1	DMA2_Stream0	Perpheral To Memory	Very High

Tabela 2: Konfiguracja DMA

ADC1:DMA2_Stream0 DMA request Settings:	
Mode:	Disable
Use ffo:	Disable
Peripheral Increment:	Disable
Memory Increment:	Enable
Peripheral Data Width:	Half Word
Memory Data Width:	Half Word

2.3 DAC

DMA request	Stream	Direction	Priority
DAC2	DMA1 Strean 6	Memory To Perpheral	Medium

Tabela 3: Konfiguracja DMA

2.4 SPI

Parametr	Wartość
Baud Rate	4.5MBits/s
Word Length	8 Bits
First Bit	MSB First
Clock Polarity	LOW
Clock Phase	1 Edge

Tabela 4: Konfiguracja peryferium SPI

2.5 USART

Parametr	Wartość
Baud Rate	115200 Bits/s
Word Length	8 Bits
PArity	None
Stop Bits	1
Data Direction	Receive and Transmit
Over Sampling	16 Samples

Tabela 5: Konfiguracja peryferium USART

3 Urządzenia zewnętrzne

3.1 Wyświetlacz TFT

Wyświetlacz TFT o przekątnej 2.41", 262k kolorów. Rozdzielczość wynosi QVGA(240x320). Jest on sterowany bezpośrednio z mikrokontrolera wykorzystując protokół RGB. Posiada kontroler ILI9341 do komunikacji przez SPI. Wyświetlacz ten został wykorzystany do wyświetlania odczytanych danych za pomocą grafu.

4 Opis działania programu

Program generuje sygnał testowy na jednym z wyjść, podłączonym do pinu przetwornika ADC. Sygnał jest odczytywany z przetwornika a następnie wyświetlany na wbudowanym wyświetlaczu.

4.1 Funkcja generująca sygnały testowe z wykorzystaniem DAC

Sygnały generowane są z wykorzystaniem DAC oraz DMA (direct access memory), aby nie zabierać czasu procesorowi. Próbne sygnały ustawiane są na stałe pod dany test. Mozliwe do wygenerowania są sygnały:

- Sygnał trójkątny - wbudowany w DAC możliwość ustawienia offsetu amplitudy oraz częstotliwości,
- sygnał sinusoidalny, z wykorzystaniem lookup table (LUT),
- inne funkcje, dla których można wygenerować tablicę danych.

Inicjacja DAC z włączonym DMA z pomocą poniższego kodu:

```
1 HAL_DAC_Start_DMA(&hdac, DAC_CHANNEL_2,(uint32_t*)Wave_LUT,128, DAC_ALIGN_12B_R);
```

4.1.1 Wypełnianie buforu bez DAC/ADC

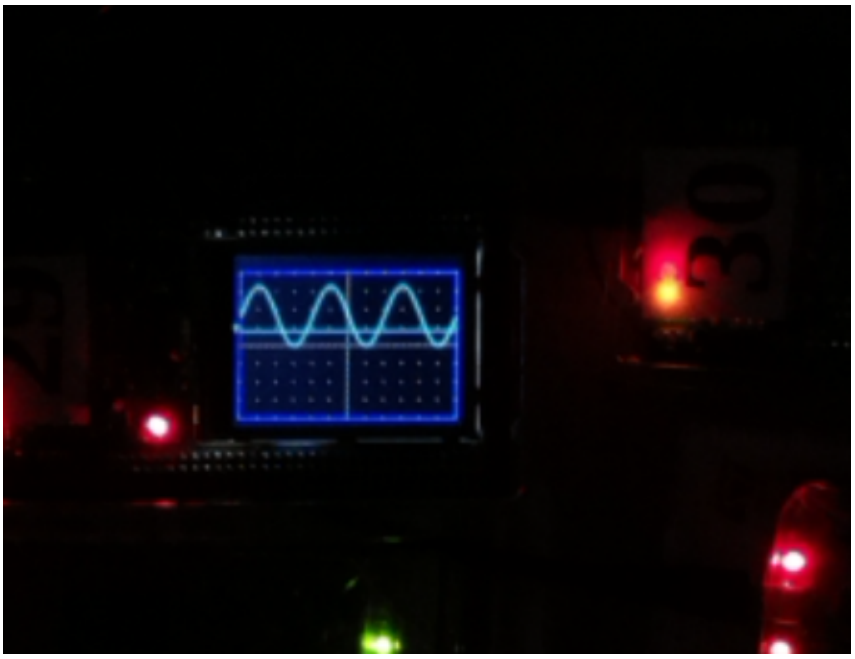
Aby testować oscyloskop bez działania DAC oraz ADC, wypełnialiśmy bezpośrednio bufor ADC_DMA_Buffer_A:

```
1  for(int i=0; i<300; ++i)
2      {
3      ADC_DMA_Buffer_A[i] = (uint16_t) 1100*sin(2*PI*10*i*0.001)+2048;
4      }
```

Generowaliśmy sygnały sinusoidalne o zadanej częstotliwości.

4.2 Obsługa wyświetlacza

Od początku trwania projektu udało się przetestować kilka bibliotek graficznych [3], z mniejszym lub większym skutkiem. Ostatecznie zdecydowaliśmy się na bibliotekę "ub_lib" znalezioną na blogu mikrocontroller.net. Biblioteka ta idealnie spełnia nasze wymagania i jest prosta w obsłudze. Pozwala w łatwy sposób na tworzenie warstw wyświetlanego obrazu, co okazało się być dużym ułatwieniem w zastosowaniu w oscyloskopie.



Rysunek 3: Wyświetlenie sinusa podanego do tablicy

Jak na powyższej grafice widać, wyświetlony został obraz sygnału sinusoidalnego rozrysowanego na czarnym tle z siatką ułatwiającą odczyt.

```
1   void ospa_draw_scale(void)
2 {
3   uint32_t n,m;
4   uint16_t xs,ys;
5   int16_t signed_int;
6
7   xs=SCALE_START_X;
8   ys=SCALE_START_Y;
9
10  // grid
11  for(m=0;m<=SCALE_H;m+=SCALE_SPACE) {
12    for(n=0;n<=SCALE_W;n+=SCALE_SPACE) {
13      UB_Graphic2D_DrawPixelNormal(m+xs,n+ys,SCALE_COL);
14    }
15  }
16
17  // X-axis (horizontal center line)
18  signed_int=SCALE_Y_MITTE+xs;
19  ospa_draw_line_h(signed_int,SCALE_COL,0);
20
21  // Y-axis (vertical center line)
22  signed_int=SCALE_X_MITTE+ys;
23  ospa_draw_line_v(signed_int,SCALE_COL,0);
24
25  // border
26  UB_Graphic2D_DrawStraightDMA(xs-1,ys-1,SCALE_W+2,LCD_DIR_HORIZONTAL,SCALE_COL);
27  UB_Graphic2D_DrawStraightDMA(xs+SCALE_H+1,ys-1,SCALE_W+2,LCD_DIR_HORIZONTAL,
```

```
        SCALE_COL);
28    UB_Graphic2D_DrawStraightDMA(xs-1,ys-1,SCALE_H+2,LCD_DIR_VERTICAL,SCALE_COL);
29    UB_Graphic2D_DrawStraightDMA(xs-1,ys+SCALE_W+1,SCALE_H+2,LCD_DIR_VERTICAL,
          SCALE_COL);
30
31     // trigger line (always visible)
32     if(setup.trigger_source == 0) {
33      signed_int=ospa_adc2pixel(setup.trigger_treshold, setup.factor_value);
34      signed_int+=SCALE_Y_MITTE+SCALE_START_X+setup.v_position;
35      if(signed_int<SCALE_START_X) signed_int=SCALE_START_X;
36      if(signed_int>SCALE_MX_PIXEL) signed_int=SCALE_MX_PIXEL;
37
38      ospa_draw_line_h(signed_int,ADC_CH1_COL,1);
39      UB_Font_DrawString(signed_int-3,0,"T",&Arial_7x10,ADC_CH1_COL,BACKGROUND_COL);
40    }
41  }
```

4.2.1 Wyświetlanie wartości

Na wyświetlaczu pokazujemy również parametry sygnału w formie tekstu.

Rysunek 4: Wyświetlenie napisu z danymi sygnału

Jednak ze względu na słabą jakość kamery do podglądu i małą czcionkę dostępną w bibliotece nie zdecydowaliśmy się na wyswietlanie parametrów na ekranie. W przypadku dostępu do płytki taki rozmiar tekstu byłby czytelny dla użytkownika.

4.3 Bufor na próbki badanego sygnału

Buforem przechowującym dane jest jednowymiarowa tablica o długości 300 pól.

```
1   volatile uint16_t ADC_DMA_Buffer_A[ADC1d_ANZ*ADC_ARRAY_LEN];
```

Dodatkowy bufor o tej samej długości jest przeznaczony do przetworzenia danych, aby byly wyświetlane od momentu wykrycia przekroczenia wartości progu(trigger).

```
1   volatile uint16_t ADC_DMA_Buffer_C[ADC1d_ANZ*ADC_ARRAY_LEN];
```

Do obliczenia FFT używany jest bufor zmiennych typu float, co jest wymagane dla funkcji obliczającej transformatę.

```
1 float FFT_DATA_IN[FFT_LENGTH];
```

4.4 ADC

Odczyt sygnału z ADC odbywa się w sposób ciągły z wykorzystaniem DMA. Dane zapisywane są do bufora, a następnie wywoływane są funkcje:

```
1 void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc)
2 void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
```

,odpowiednio w połowie zapisu do bufora oraz na koniec zapisu do bufora. Następnie sprawdzane jest przekroczenie wartości wyzwolenia w danej części bufora, jesli sygnał osiągnał wymagany poziom jest wyświetlany na wyświetlaczu. Z powodu braku połaczenia ADC z DAC na płytce numer 29, zaimplementowaliśmy bezpośrednie wpisywanie próbek sygnału do buforu `ADC_DMA_Buffer_A[i]`.

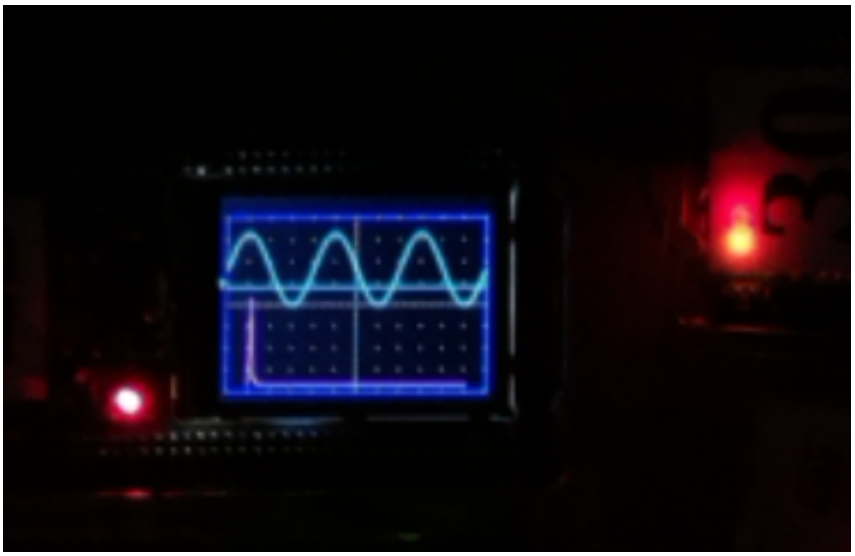
4.5 FFT

Czym jest FFT?

Transformacja Fouriera rozkłada funkcję okresową na szereg funkcji okresowych tak, że uzyskana transformata podaje w jaki sposób poszczególne częstotliwości składają się na pierwotną funkcję.

Ponieważ w praktyce w wyniku pomiarów otrzymujemy dane o charakterze dyskretnym, a nie ciągłym, konieczne jest zdefiniowanie dyskretnego odpowiednika ciągłej transformaty Fouriera

FFT jest to DFT ze zmniejszoną liczbą niezbędnych operacji arytmetycznych. Celem FFT jest zmniejszenie długiego algorytmu obliczeniowego przez jego podział na krótsze i prostsze obliczenia DFT i skrócenie czasu obliczeń.



Rysunek 5: Wyświetlenie sinusa oraz obliczenie FFT

Do obliczenia FFT również skorzystaliśmy z gotowej biblioteki i obliczenia są wykonywane tylko na widocznej części wyświetlanego obrazu.

```
1   void fft_calc(void)
2 {
3   float maxValue;
4   uint32_t n;
5
6
7   arm_rfft_f32(&S, FFT_DATA_IN, FFT_CMPLX_DATA);
8   arm_cmplx_mag_f32(FFT_CMPLX_DATA, FFT_MAG_DATA, FFT_LENGTH);
9
10  maxValue=0.1;
11  for(n=2;n<FFT_VISIBLE_LENGTH;n++) {
12    if(FFT_MAG_DATA[n]>maxValue) maxValue=FFT_MAG_DATA[n];
13  }
14
15  FFT_UINT_DATA[0]=0;
16  FFT_UINT_DATA[1]=0;
17  for(n=2;n<FFT_VISIBLE_LENGTH;n++) {
18    FFT_UINT_DATA[n]=(uint16_t)(FFT_UINT_MAXdata*FFT_MAG_DATA[n]/maxValue);
```

```
19   }
20 }
```

Najpierw wywoływana jest funkcja przetwarzająca bufor wejściowy (odczytane próbki) do bufora **FFT_CMPLX_DATA**. Jest on 2 razy większy ponieważ zawiera część rzeczywistą i urojoną każdej próbki.

```
1 arm_rfft_fast_f32(&S,FFT_DATA_IN, FFT_CMPLX_DATA, ifftFlag);
```

Następnie obliczamy moduł każdej próbki i zapisujemy do bufora **FFT_MAG_DATA**. Kolejnym krokiem jest znalezienie maksymalnej wartości modułu dla przetwarzanych próbek oraz przeskalowanie wartości zależnie od maxValue.
Użyte funkcje są dostępne w bibliotece DSP (digital signal processing). [5]

4.6 Komunikacja z oscyloskopem po UART

Menu użytkownika jest zrealizowane za pomocą komunikacji UART z urządzeniem. Z pośród dostępnych do modyfikacji parametrów mamy:

- zmianę poziomu wyzwalania
- zmianę skali pionowej (amplituda)
- zmianę skali poziomej (czas)
- zmianę częstotliwości generowanego sygnału
- zmianę amplitudy generowanego sygnału
- start/stop

```
1  void ospa_menu_show(void)
2  {
3    printf("*******************************\r\n");
4    printf("*   OSCYLOSKOP OSPA WITA!!!    *\r\n");
5    printf("*******************************\r\n");
6    printf("* Dostępne opcje:              *\r\n");
7    printf("* 1.Zmien poziom wyzwalania    *\r\n");
8    printf("* 2.Zmien skale pionowa        *\r\n");
9    printf("* 3.Zmien skale pozioma        *\r\n");
10   printf("* 4.Zmien czestotliwosc sygnalu*\r\n");
11   printf("* 5.Zmien amplitude sygnalu    *\r\n");
12   printf("* 6. START/STOP                *\r\n");
13   printf("*******************************\r\n");
14   printf("Wybierz opcje: ");
15   fflush(stdout);
16 }
```

5 Zarządzanie projektem

Do kontroli wersji oraz łatwej wymiany postępów skorzystaliśmy z sytemu GIT, nasze repozytorium znajduje się pod adresem `https://github.com/kaszmirek/SR_OSPA`.

6 Podsumowanie

Największy problem stanowiło dobranie odpowiedniej i działającej biblioteki graficznej do obsługi wyświetlacza, przetestowaliśmy wiele pozycji, większość z nich przestała być wspierana przez nowsze wersje STM32CubeIDE. Ostatecznie udało się znaleźć odpowiednie oprogramowanie na stronie poświęconej projektom na płytkach rozwojowych z rodziny STM32. W wykonaniu projektu ułatwiły nam obszerne opracowania naukowe zarówno jak i wpisy społeczności miłośników mikro-kontrolerów które pomogły w odnalezieniu odpowiednich bibliotek.

Udało się nam zrealizować wyświetlanie sygnału na wyświetlaczu wraz z obliczaniem parametrów sygnału. Dodatkowo możliwa jest kontrola wyświetlania sygnału poprzez dostosowanie skali oraz zmiana parametrów sygnału,

Literatura

[1] T. Ostrowski. Miniscope v2c `https://tomeko.net//miniscope_v2c`.

[2] ST. Extending the DAC performance of STM32 microcontrollers. Wrze. 2018.

[3] ST. Getting started with STemWin Library. Kwi. 2018.

[4] ST. AN2834 Application note How to get the best ADC accuracy in STM32 microcontrollers. Gru. 2020.

[5] J. Szemiet. Analizator widma z FFT na STM32 z Cortex-M4. Gru. 2013.