МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования «Южно-Уральский государственный университет»  
(национальный исследовательский университет)

Высшая школа электроники и компьютерных наук

Кафедра «Информационно-измерительная техника»

ОТЧЁТ

по лабораторной работе № 2

по дисциплине «Измерительно информационные системы»

Вариант № 2

Выполнил:

студент группы КЭ–413

/ С.С. Ильин /

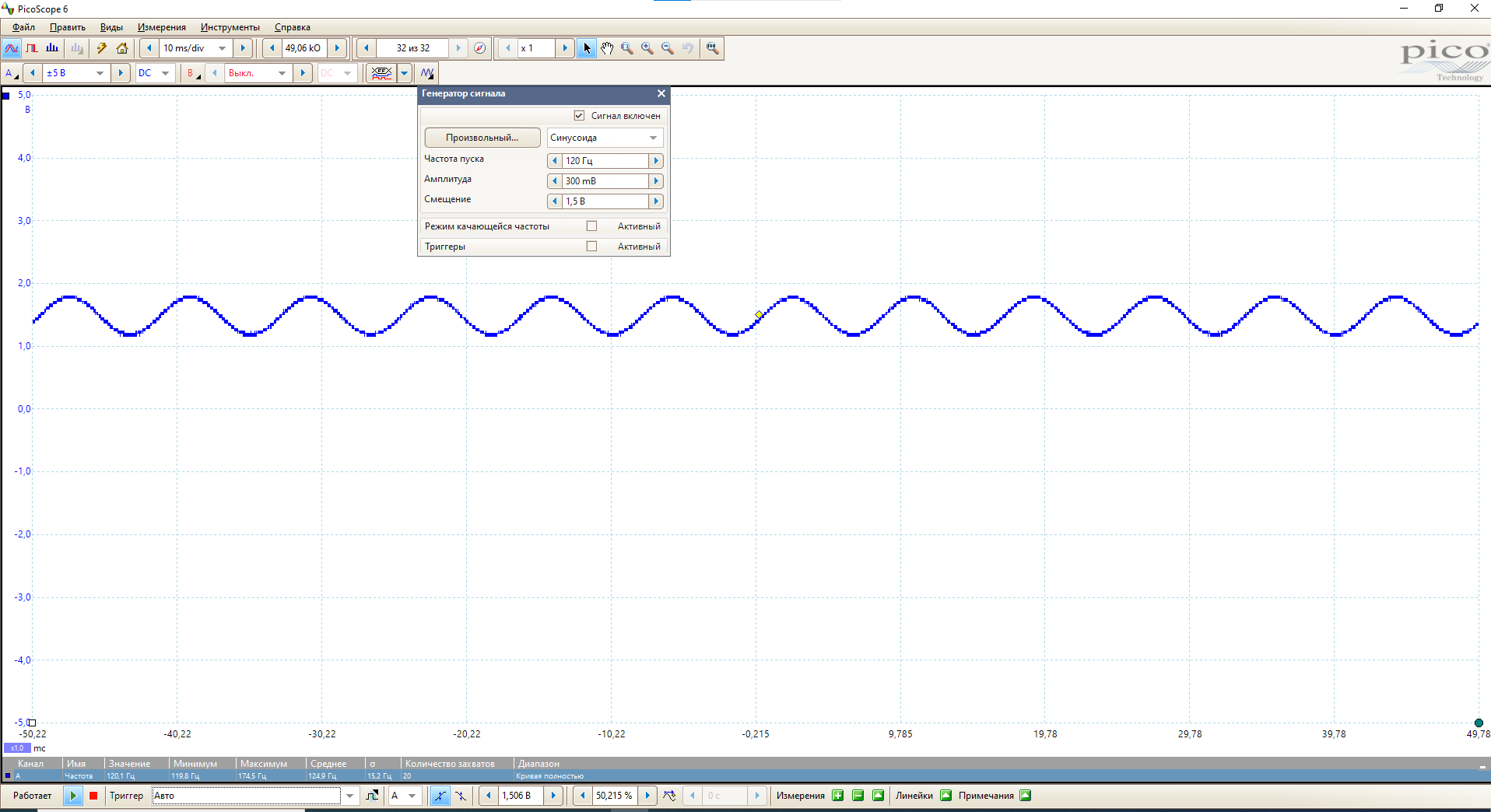
« » 2022 г.

Проверила: доцент

/ Е.В. Вставская /

« » 2022 г.

1. Рассчитать частоту сигнала по формуле 100+10\*N, где N – номер варианта. Подать частоту с генератора сигналов на аналоговый вход отладочной платы. Установить смещение сигнала равным 1,5В, амплитуду сигнала 300мВ. Привести скриншот сигнала с генератора.



2. Вычислить частоту наивысшей гармоники с использованием БПФ на 1024 точки с частотой дискретизации 4000 Гц. Вывести значение частоты и номер наивысшей гармоники на ЖКИ. Привести фото с результатом.

App.h

#include "main.h"

#include "fonts.h"

#include "ssd1306.h"

extern ADC\_HandleTypeDef hadc1;

extern DMA\_HandleTypeDef hdma\_adc1;

extern TIM\_HandleTypeDef htim6;

extern I2C\_HandleTypeDef hi2c2;

void AppInit();

void AppWork();

App.c

#include "app.h"

#include "arm\_math.h"

uint16\_t ADCs[2048] = {0}; // получение отсчетов АЦП

float ADCin[2048] = {0};

float ADCout[2048] = {0};

uint8\_t f = 0;

arm\_rfft\_fast\_instance\_f32 RS;

uint8\_t s[32]= {0};

//------------------------------------------------------------------------------

// Обработка прерывания DMA ADC

void ADC\_ConvCplt(DMA\_HandleTypeDef\* hdma)

{

arm\_q15\_to\_float ((q15\_t\*)&ADCs[1024], ADCin, 1024);

arm\_scale\_f32 (ADCin, 32768.0, ADCin, 1024);

f = 1;

}

void ADC\_HalfConvCplt(DMA\_HandleTypeDef\* hdma)

{

arm\_q15\_to\_float ((q15\_t\*)&ADCs[0], ADCin, 1024);

arm\_scale\_f32 (ADCin, 32768.0, ADCin, 1024);

f = 1;

}

void AppInit()

{

ssd1306\_Init(hi2c2, 0x78);

HAL\_Delay(10);

ssd1306\_Fill(Black);

ssd1306\_SetCursor(1, 1);

ssd1306\_WriteString("Привет!", 8, White);

ssd1306\_SetCursor(1, 10);

ssd1306\_WriteString("Привет!", 12, White);

ssd1306\_SetCursor(1, 25);

ssd1306\_WriteString("Привет!", 16, White);

ssd1306\_SetCursor(1, 40);

ssd1306\_WriteString("Привет!", 24, White);

HAL\_ADCEx\_Calibration\_Start(&hadc1, ADC\_SINGLE\_ENDED);

HAL\_Delay(10);

arm\_rfft\_fast\_init\_f32(&RS, 1024);

HAL\_TIM\_Base\_Start(&htim6);

HAL\_ADC\_Start\_DMA(&hadc1, (uint32\_t \*)ADCs, 2048);

hdma\_adc1.XferCpltCallback = ADC\_ConvCplt;

hdma\_adc1.XferHalfCpltCallback = ADC\_HalfConvCplt;

}

void AppWork()

{

// HAL\_Delay(100);

if(f == 0)

return;

f = 0;

uint32\_t index;

float harm;

arm\_rfft\_fast\_f32 (&RS, ADCin, ADCout, 0);

arm\_cmplx\_mag\_f32 (ADCout, ADCout, 512);

ADCout[0]=0;

ADCout[1]=0;

arm\_max\_f32 (ADCout, 512, &harm, &index);

uint32\_t freq = index\*400000/1024;

s[0] = freq/10000 + '0';

s[1] = freq/1000%10 + '0';

s[2] = freq/100%10 + '0';

s[3] = ',';

s[4]=freq/10%10 + '0';

s[5] = freq%10 + '0';

s[6] = 0;

ssd1306\_Fill(Black);

ssd1306\_SetCursor(1, 1);

ssd1306\_WriteString(s, 24, White);

s[0] = index/100 + '0';

s[1] = index/10%10 + '0';

s[2] = index/1%10 + '0';

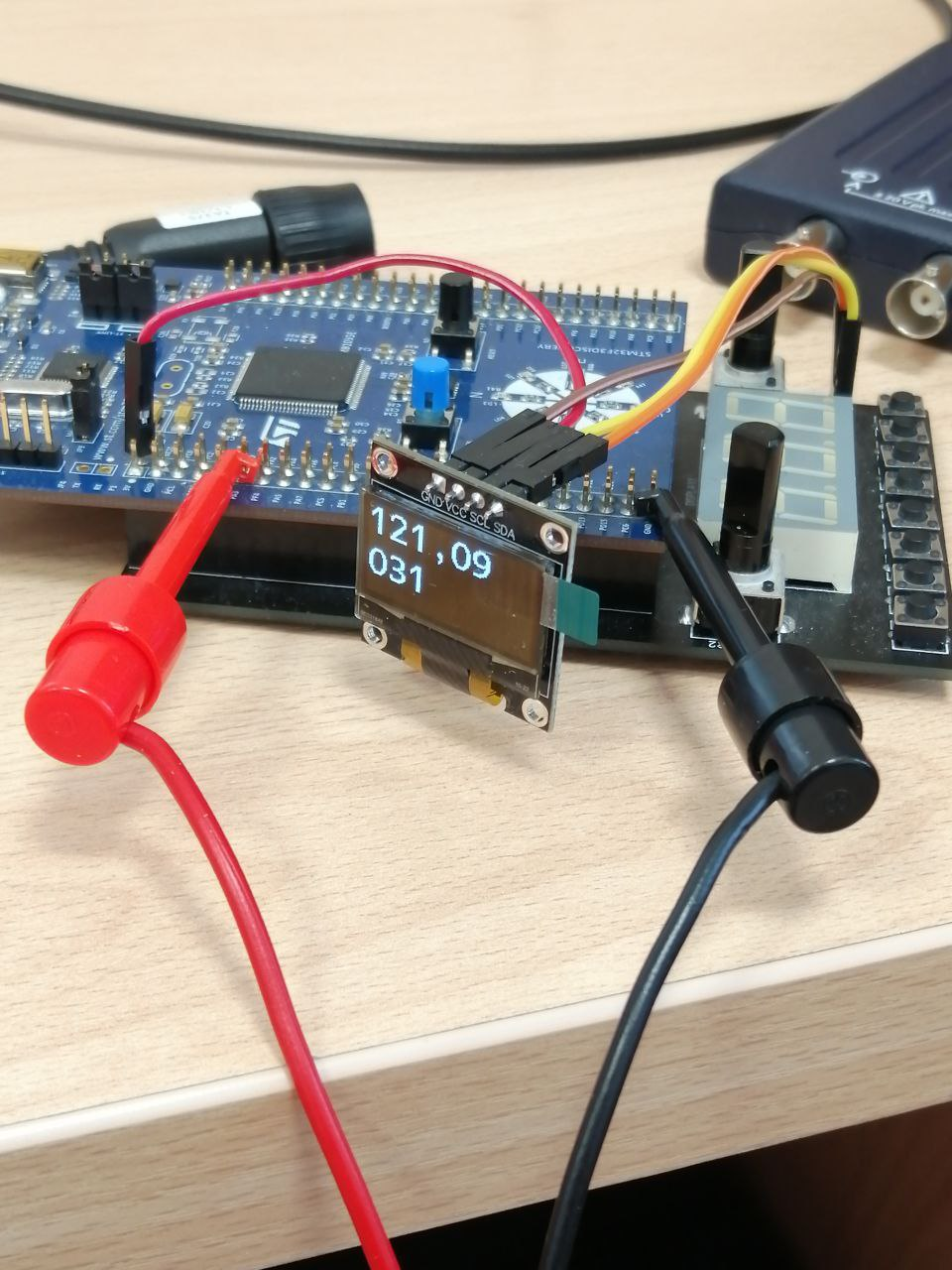
s[3] = 0;

ssd1306\_SetCursor(1, 30);

ssd1306\_WriteString(s, 24, White);

ssd1306\_UpdateScreen();

}



3. Уменьшить частоту дискретизации до 1000 Гц. Вывести значение частоты и номер наивысшей гармоники на ЖКИ. Привести фото с результатом.

App.h

#include "main.h"

#include "fonts.h"

#include "ssd1306.h"

extern I2C\_HandleTypeDef hi2c2;

extern ADC\_HandleTypeDef hadc1;

extern DMA\_HandleTypeDef hdma\_adc1;

extern TIM\_HandleTypeDef htim6;

void AppInit();

void AppWork();

App.c

#include "app.h"

#include "arm\_math.h"

uint16\_t ADCs[2048] = {0}; // получение отсчетов АЦП

float ADCin[2048] = {0};

float ADCout[2048] = {0};

uint8\_t f = 0;

arm\_rfft\_fast\_instance\_f32 RS;

uint8\_t s[32]= {0};

//------------------------------------------------------------------------------

// Обработка прерывания DMA ADC

void ADC\_ConvCplt(DMA\_HandleTypeDef\* hdma)

{

arm\_q15\_to\_float ((q15\_t\*)&ADCs[1024], ADCin, 1024);

arm\_scale\_f32 (ADCin, 32768.0, ADCin, 1024);

f = 1;

}

void ADC\_HalfConvCplt(DMA\_HandleTypeDef\* hdma)

{

arm\_q15\_to\_float ((q15\_t\*)&ADCs[0], ADCin, 1024);

arm\_scale\_f32 (ADCin, 32768.0, ADCin, 1024);

f = 1;

}

void AppInit()

{

ssd1306\_Init(hi2c2, 0x78);

HAL\_Delay(10);

ssd1306\_Fill(Black);

ssd1306\_SetCursor(1, 1);

ssd1306\_WriteString("Привет!", 8, White);

ssd1306\_SetCursor(1, 10);

ssd1306\_WriteString("Привет!", 12, White);

ssd1306\_SetCursor(1, 25);

ssd1306\_WriteString("Привет!", 16, White);

ssd1306\_SetCursor(1, 40);

ssd1306\_WriteString("Привет!", 24, White);

HAL\_ADCEx\_Calibration\_Start(&hadc1, ADC\_SINGLE\_ENDED);

HAL\_Delay(10);

arm\_rfft\_fast\_init\_f32(&RS, 1024);

HAL\_TIM\_Base\_Start(&htim6);

HAL\_ADC\_Start\_DMA(&hadc1, (uint32\_t \*)ADCs, 2048);

hdma\_adc1.XferCpltCallback = ADC\_ConvCplt;

hdma\_adc1.XferHalfCpltCallback = ADC\_HalfConvCplt;

}

void AppWork()

{

// HAL\_Delay(100);

if(f == 0)

return;

f = 0;

uint32\_t index;

float harm;

arm\_rfft\_fast\_f32 (&RS, ADCin, ADCout, 0);

arm\_cmplx\_mag\_f32 (ADCout, ADCout, 512);

ADCout[0]=0;

ADCout[1]=0;

arm\_max\_f32 (ADCout, 512, &harm, &index);

uint32\_t freq = index\*100000/1024;

s[0] = freq/10000 + '0';

s[1] = freq/1000%10 + '0';

s[2] = freq/100%10 + '0';

s[3] = ',';

s[4]=freq/10%10 + '0';

s[5] = freq%10 + '0';

s[6] = 0;

ssd1306\_Fill(Black);

ssd1306\_SetCursor(1, 1);

ssd1306\_WriteString(s, 24, White);

s[0] = index/100 + '0';

s[1] = index/10%10 + '0';

s[2] = index/1%10 + '0';

s[3] = 0;

ssd1306\_SetCursor(1, 30);

ssd1306\_WriteString(s, 24, White);

ssd1306\_UpdateScreen();

}



4. Вычислить частоту и период сигнала с использованием БПФ. Вывести значения на ЖКИ. Привести фото с результатом.

#include "app.h"

#include "arm\_math.h"

uint16\_t ADCs[2048] = {0}; // получение отсчетов АЦП

float ADCin[2048] = {0};

float ADCout[2048] = {0};

float K = 205;

float K1 = 0;

float K2 = 0;

float K3 = 0;

float fi = 0.9765625;

float alfa = 1,456928781;

float ReW = 0,72846439;

float InW = -0,685083668;

uint8\_t f = 0;

arm\_rfft\_fast\_instance\_f32 RS;

uint8\_t s[32]= {0};

//------------------------------------------------------------------------------

// Обработка прерывания DMA ADC

void ADC\_ConvCplt(DMA\_HandleTypeDef\* hdma)

{

arm\_q15\_to\_float ((q15\_t\*)&ADCs[1024], ADCin, 1024);

arm\_scale\_f32 (ADCin, 32768.0, ADCin, 1024);

f = 1;

}

void ADC\_HalfConvCplt(DMA\_HandleTypeDef\* hdma)

{

arm\_q15\_to\_float ((q15\_t\*)&ADCs[0], ADCin, 1024);

arm\_scale\_f32 (ADCin, 32768.0, ADCin, 1024);

f = 1;

}

void AppInit()

{

ssd1306\_Init(hi2c2, 0x78);

HAL\_Delay(10);

ssd1306\_Fill(Black);

ssd1306\_SetCursor(1, 1);

ssd1306\_WriteString("Привет!", 8, White);

ssd1306\_SetCursor(1, 10);

ssd1306\_WriteString("Привет!", 12, White);

ssd1306\_SetCursor(1, 25);

ssd1306\_WriteString("Привет!", 16, White);

ssd1306\_SetCursor(1, 40);

ssd1306\_WriteString("Привет!", 24, White);

HAL\_ADCEx\_Calibration\_Start(&hadc1, ADC\_SINGLE\_ENDED);

HAL\_Delay(10);

arm\_rfft\_fast\_init\_f32(&RS, 1024);

HAL\_TIM\_Base\_Start(&htim6);

HAL\_ADC\_Start\_DMA(&hadc1, (uint32\_t \*)ADCs, 2048);

hdma\_adc1.XferCpltCallback = ADC\_ConvCplt;

hdma\_adc1.XferHalfCpltCallback = ADC\_HalfConvCplt;

}

void AppWork()

{

// HAL\_Delay(100);

if(f == 0)

return;

f = 0;

uint32\_t index;

float harm;

arm\_rfft\_fast\_f32 (&RS, ADCin, ADCout, 0);

arm\_cmplx\_mag\_f32 (ADCout, ADCout, 512);

ADCout[0]=0;

ADCout[1]=0;

arm\_max\_f32 (ADCout, 512, &harm, &index);

uint32\_t freq = index\*100000/1024;

K1= ADCout[index-1];

K2=ADCout[index];

K3=ADCout[index+1];

float f=(fi\*(index)+0.5\*fi\*(K3-K1)/(2\*K2-K3-K1));

uint32\_t Ferr = (uint32\_t)(f\*100);

/\*ssd1306\_Fill(Black);

ssd1306\_SetCursor(1, 1);

ssd1306\_WriteString(Ferr, 24, White);

\*/

s[0] = Ferr/10000 + '0';

s[1] = Ferr/1000%10 + '0';

s[2] = Ferr/100%10 + '0';

s[3] = ',';

s[4]=Ferr/10%10 + '0';

s[5] = Ferr%10 + '0';

s[6] = 0;

ssd1306\_Fill(Black);

ssd1306\_SetCursor(1, 1);

ssd1306\_WriteString(s, 24, White);

uint32\_t period = (uint32\_t)(100000.0/f);

s[0] = period/100 + '0';

s[1] = period/10%10 + '0';

s[2] = period/1%10 + '0';

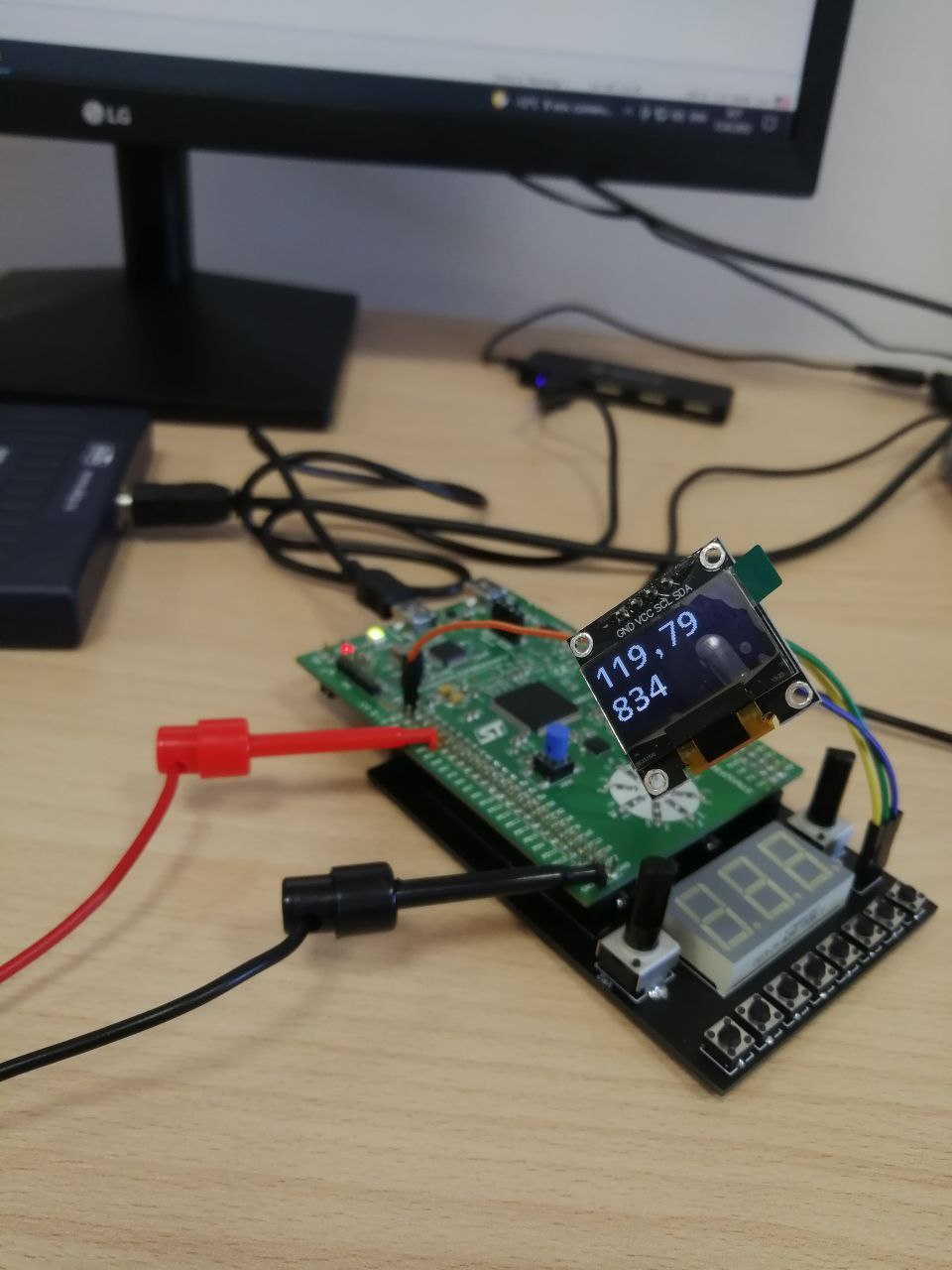
s[3] = 0;

ssd1306\_SetCursor(1, 30);

ssd1306\_WriteString(s, 24, White);

ssd1306\_UpdateScreen();

}



5. Вычислить амплитуду «своей» гармоники и двух соседних, используя алгоритм Гёрцеля. Вывести результат на ЖКИ. Привести фото с результатом.

#include "app.h"

#include "arm\_math.h"

uint16\_t ADCs[2048] = {0}; // получение отсчетов АЦП

float ADCin[2048] = {0};

float ADCout[2048] = {0};

float K = 205;

float K1 = 0;

float K2 = 0;

float K3 = 0;

float fi = 0.9765625;

float alfa1 = 1.465308543;

float ReW1 = 0.732654272;

float InW1 = -0.680600998;

float alfa = 1.456928781;

float ReW = 0.72846439;

float InW = -0.685083668;

float alfa2 = 1.448494166;

float ReW2 = 0.724247083;

float InW2 = -0.689540545;

uint8\_t f = 0;

arm\_rfft\_fast\_instance\_f32 RS;

uint8\_t s[32]= {0};

//------------------------------------------------------------------------------

// Обработка прерывания DMA ADC

void ADC\_ConvCplt(DMA\_HandleTypeDef\* hdma)

{

arm\_q15\_to\_float ((q15\_t\*)&ADCs[1024], ADCin, 1024);

arm\_scale\_f32 (ADCin, 32768.0, ADCin, 1024);

f = 1;

}

void ADC\_HalfConvCplt(DMA\_HandleTypeDef\* hdma)

{

arm\_q15\_to\_float ((q15\_t\*)&ADCs[0], ADCin, 1024);

arm\_scale\_f32 (ADCin, 32768.0, ADCin, 1024);

f = 1;

}

void AppInit()

{

ssd1306\_Init(hi2c2, 0x78);

HAL\_Delay(10);

ssd1306\_Fill(Black);

ssd1306\_SetCursor(1, 1);

ssd1306\_WriteString("Привет!", 8, White);

ssd1306\_SetCursor(1, 10);

ssd1306\_WriteString("Привет!", 12, White);

ssd1306\_SetCursor(1, 25);

ssd1306\_WriteString("Привет!", 16, White);

ssd1306\_SetCursor(1, 40);

ssd1306\_WriteString("Привет!", 24, White);

HAL\_ADCEx\_Calibration\_Start(&hadc1, ADC\_SINGLE\_ENDED);

HAL\_Delay(10);

arm\_rfft\_fast\_init\_f32(&RS, 1024);

HAL\_TIM\_Base\_Start(&htim6);

HAL\_ADC\_Start\_DMA(&hadc1, (uint32\_t \*)ADCs, 2048);

hdma\_adc1.XferCpltCallback = ADC\_ConvCplt;

hdma\_adc1.XferHalfCpltCallback = ADC\_HalfConvCplt;

}

void AppWork()

{

// HAL\_Delay(100);

if(f == 0)

return;

f = 0;

/\*

K1= ADCout[index-1];

K2=ADCout[index];

K3=ADCout[index+1];

float f=(fi\*(index)+0.5\*fi\*(K3-K1)/(2\*K2-K3-K1));

uint32\_t Ferr = (uint32\_t)(f\*100);

ssd1306\_Fill(Black);

ssd1306\_SetCursor(1, 1);

ssd1306\_WriteString(Ferr, 24, White);

s[0] = Ferr/10000 + '0';

s[1] = Ferr/1000%10 + '0';

s[2] = Ferr/100%10 + '0';

s[3] = ',';

s[4]=Ferr/10%10 + '0';

s[5] = Ferr%10 + '0';

s[6] = 0;

ssd1306\_Fill(Black);

ssd1306\_SetCursor(1, 1);

ssd1306\_WriteString(s, 24, White);

uint32\_t period = (uint32\_t)(100000.0/f);

s[0] = period/100 + '0';

s[1] = period/10%10 + '0';

s[2] = period/1%10 + '0';

s[3] = 0;

\*/

float m\_0=0, m\_1=0, m\_2=0;

for(uint16\_t i=2; i<1024; i++)

{

m\_2=alfa1\*m\_1+ADCin[i]-m\_0;

m\_0=m\_1;

m\_1= m\_2;

}

float Re1 =ReW1\*m\_1 - m\_0;

float Im1 = InW1\*m\_1;

float AmpF1 = (Re1\*Re1) + (Im1\*Im1);

arm\_sqrt\_f32 (AmpF1, &AmpF1);

ssd1306\_Fill(Black);

uint32\_t Amp1 = (uint32\_t)(AmpF1);

s[0] = Amp1/100000 + '0';

s[1] = Amp1/10000%10 + '0';

s[2] = Amp1/1000%10 + '0';

s[3]=Amp1/100%10 + '0';

s[4]=Amp1/10%10 + '0';

s[5] = Amp1%10 + '0';

s[6] = 0;

ssd1306\_SetCursor(1, 1);

ssd1306\_WriteString(s, 16, White);

float v\_0=0, v\_1=0, v\_2=0;

for(uint16\_t i=2; i<1024; i++)

{

v\_2=alfa\*v\_1+ADCin[i]-v\_0;

v\_0=v\_1;

v\_1= v\_2;

}

float Re =ReW\*v\_1 - v\_0;

float Im = InW\*v\_1;

float AmpF = (Re\*Re) + (Im\*Im);

arm\_sqrt\_f32 (AmpF, &AmpF);

uint32\_t Amp = (uint32\_t)(AmpF);

s[0] = Amp/100000 + '0';

s[1] = Amp/10000%10 + '0';

s[2] = Amp/1000%10 + '0';

s[3]=Amp/100%10 + '0';

s[4]=Amp/10%10 + '0';

s[5] = Amp%10 + '0';

s[6] = 0;

ssd1306\_SetCursor(1, 20);

ssd1306\_WriteString(s, 16, White);

float b\_0=0, b\_1=0, b\_2=0;

for(uint16\_t i=2; i<1024; i++)

{

b\_2=alfa2\*b\_1+ADCin[i]-b\_0;

b\_0=b\_1;

b\_1= b\_2;

}

float Re2 =ReW2\*b\_1 - b\_0;

float Im2 = InW2\*b\_1;

float AmpF2 = (Re2\*Re2) + (Im2\*Im2);

arm\_sqrt\_f32 (AmpF2, &AmpF2);

uint32\_t Amp2 = (uint32\_t)(AmpF2);

s[0] = Amp2/100000 + '0';

s[1] = Amp2/10000%10 + '0';

s[2] = Amp2/1000%10 + '0';

s[3]=Amp2/100%10 + '0';

s[4]=Amp2/10%10 + '0';

s[5] = Amp2%10 + '0';

s[6] = 0;

ssd1306\_SetCursor(1,40 );

ssd1306\_WriteString(s, 16, White);

ssd1306\_UpdateScreen();

