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 № 3

по дисциплине «Измерительно информационные системы»

Вариант № 2

Выполнил:

студент группы КЭ–413

/ С.С. Ильин /

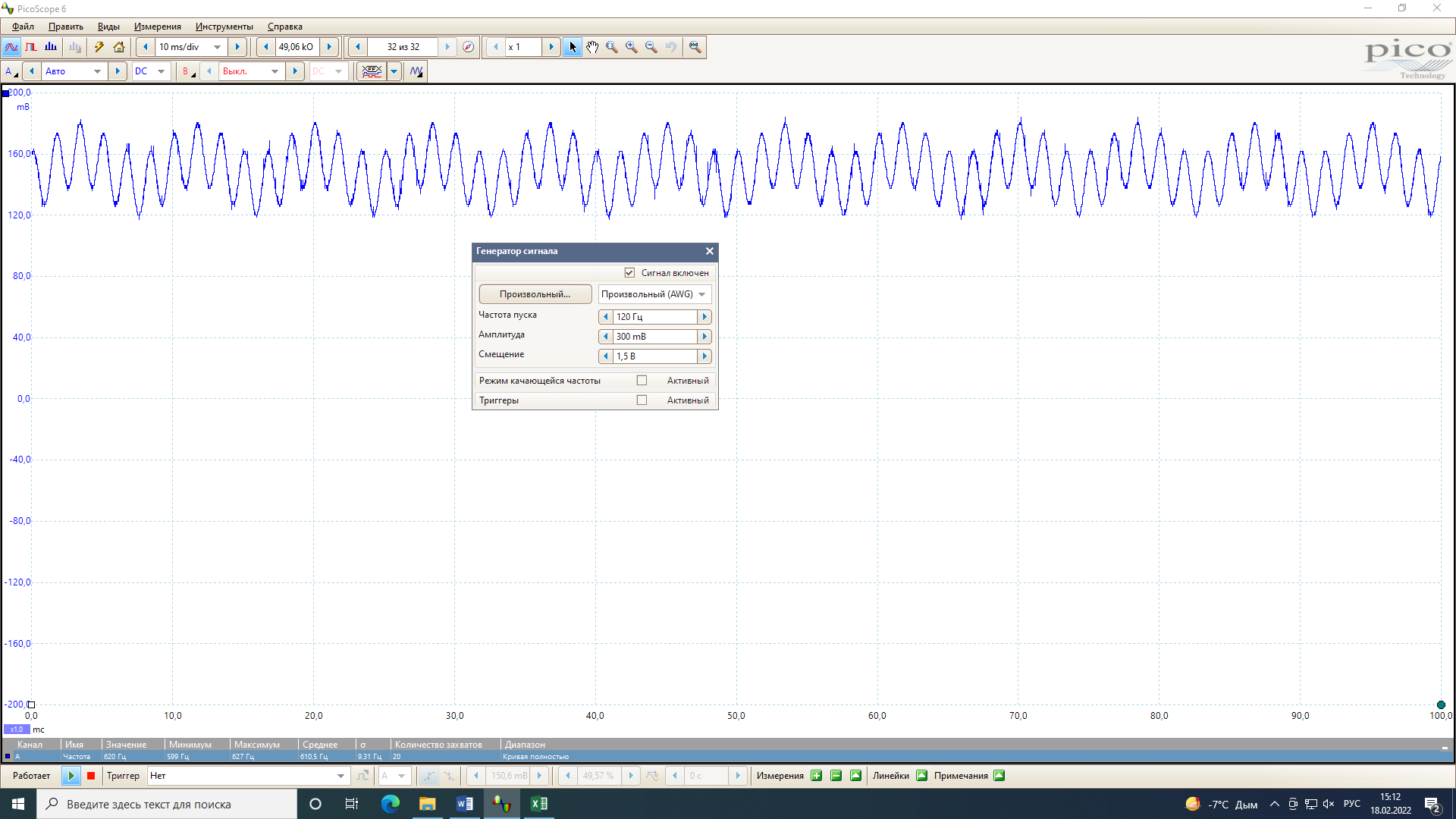
« » 2022 г.

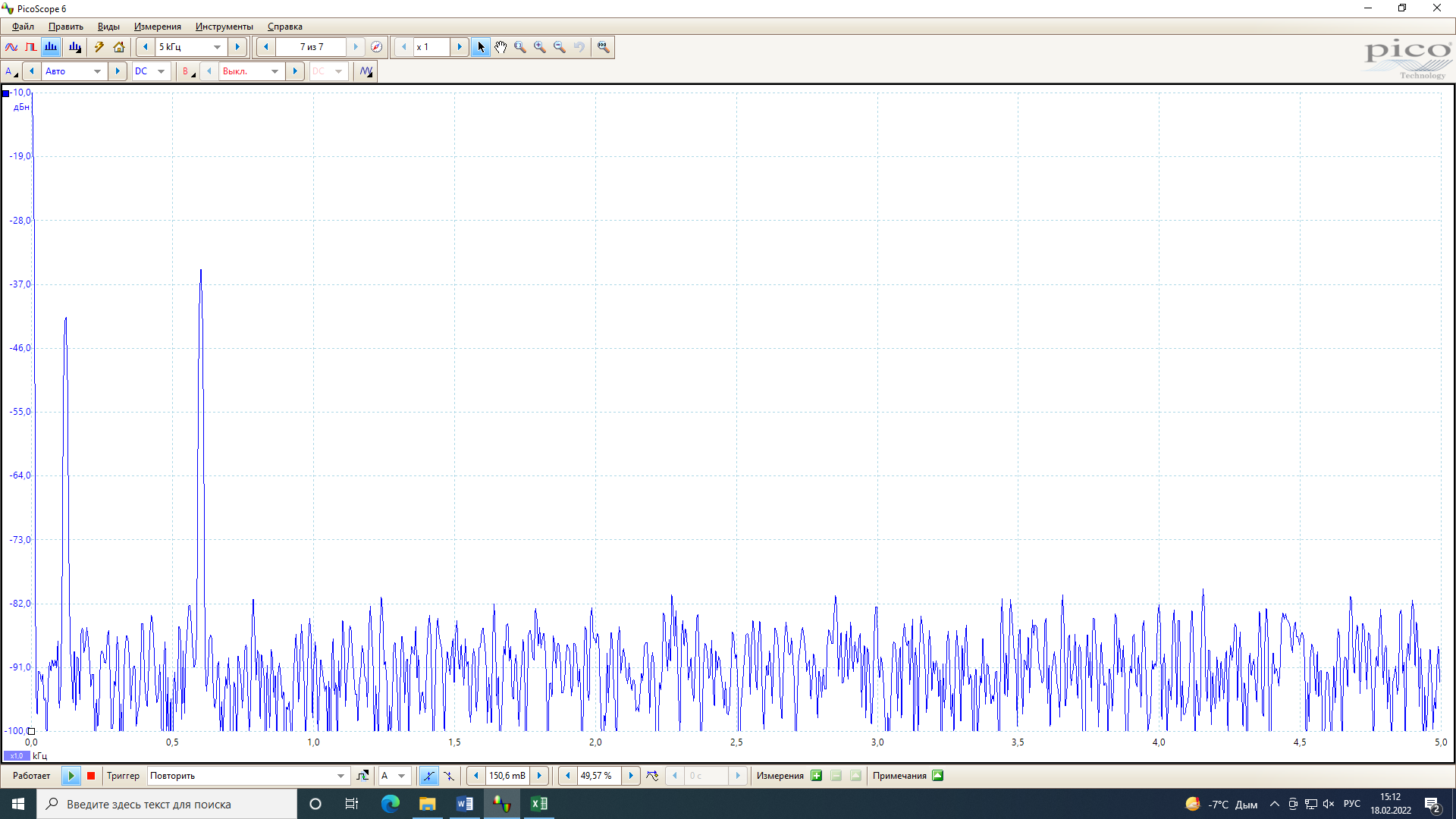
Проверила: доцент

/ Е.В. Вставская /

« » 2022 г.

1. Рассчитать частоту сигнала по формуле 100+10\*N, где N – номер варианта. Сформировать частоту сигнала с генератора произвольной формы, добавив к собственной частоте 5-ую гармонику 2-кратной амплитуды. 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.

#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. Произвести фильтрацию в частотной области, убрав из рассмотрения все гармоники спектра с частотой 150% от рассматриваемой и выше. Вывести значение частоты и номер наивысшей гармоники 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};

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);

arm\_fill\_f32(0, &ADCout[77], 435);

arm\_fill\_f32(0, &ADCout[47], 466);

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();

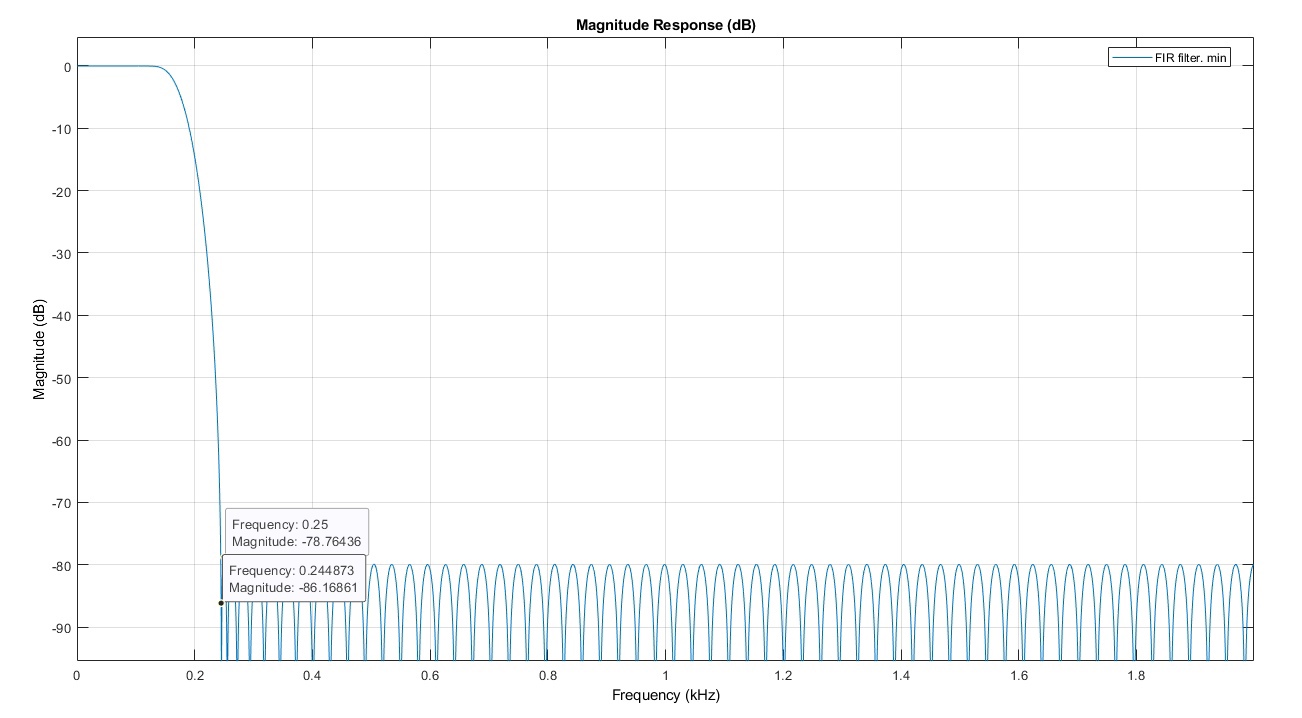
}

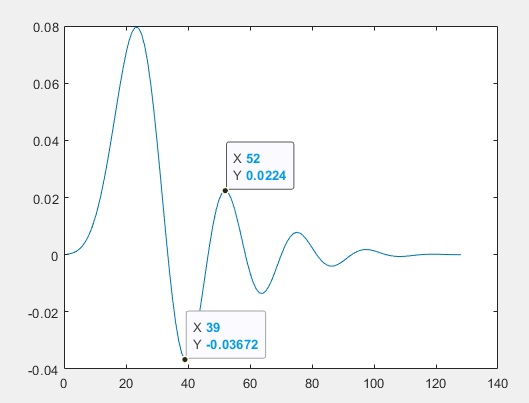


4. Рассчитать КИХ-фильтр нижних частот 128 порядка с частотой среза, соответствующей 150% заданной частоты по варианту. Привести результат расчета в виде АЧХ фильтра.

N = 127; % порядок фильтра  
Fp = 180; % частота среза  
Fs = 4000; % частота дискретизации  
Rp = 0.000057565;  
Rst = 1e-4;  
eqnum3 = firceqrip(N,Fp/(Fs/2),[Rp Rst],'min');

fvt = fvtool(eqnum3,1,'Fs',Fs,'Color','White');  
legend(fvt,'FIR filter. min')  
fp = fopen("C:\4\coefs.txt","wt");  
fprintf(fp,"%.20f,\n",eqnum3);





5. Использовать фильтр с децимацией с коэффициентом, равным 4. Выполнить фильтрацию сигнала во временной области, после чего выполнить БПФ. Вывести значение частоты и номер наивысшей гармоники (после фильтра) на ЖКИ. Привести фото с результатом.

**App.c**

#include "app.h"

#include "arm\_math.h"

#define numTaps 128 // порядок фильтра

#define blockSize 1024 // количество фильтруемых отсчетов

arm\_fir\_decimate\_instance\_f32 S ; // структура

const float FilterCoefs[128] = {0.00034157976782784368,

0.00015152465157207174,

0.00032762131140274251,

0.00066121771254317330,

0.00119367510177155987,

0.00199302939846918159,

0.00313516182173613706,

0.00470031420080834188,

0.00676819168346681072,

0.00941186168407926599,

0.01269066304626316130,

0.01664255906946410235,

0.02127644719292338948,

0.02656500256974908894,

0.03243886552242989524,

0.03878283184312875803,

0.04543457183758551482,

0.05218651186940612835,

0.05879127513436652880,

0.06497064112651099177,

0.07042786074033466348,

0.07486292959956038218,

0.07799002035329651028,

0.07955598404349585595,

0.07935876908725143786,

0.07726440770917455958,

0.07322113674506863790,

0.06726946690578851362,

0.05954712581788928344,

0.05028805335509076158,

0.03981510634424263034,

0.02852652266021239724,

0.01687675137649597215,

0.00535264007658737672,

-0.00555367338557173889,

-0.01537335519793556551,

-0.02368906091284464152,

-0.03016071377177571150,

-0.03454695402575598540,

-0.03672075686202770639,

-0.03667784935521136574,

-0.03453710009205555875,

-0.03053292870250685215,

-0.02500004882208001211,

-0.01835144921725641309,

-0.01105124183938751420,

-0.00358426637476371400,

0.00357540826403879384,

0.00999479863699280838,

0.01530898660332288136,

0.01924284801082243659,

0.02162604382880410278,

0.02240029225930357387,

0.02161856969562033978,

0.01943647107089004761,

0.01609651577982458212,

0.01190685282796770036,

0.00721615866996188488,

0.00238662453524698514,

-0.00223281138052001385,

-0.00633092451388194907,

-0.00965468143812230929,

-0.01202427319836833590,

-0.01334179373840318691,

-0.01359329931693705612,

-0.01284434808667054249,

-0.01122970096469703971,

-0.00893839354763215597,

-0.00619552701767399697,

-0.00324246374004744922,

-0.00031714816353847541,

0.00236398576469781579,

0.00462112592888833367,

0.00632215085106082025,

0.00738912335370835605,

0.00779992996462816005,

0.00758527608687048060,

0.00682173840272307184,

0.00562167825272489891,

0.00412104114138738768,

0.00246629810048485353,

0.00080168353841976253,

-0.00074224528004770778,

-0.00205786590158003966,

-0.00306698644243463521,

-0.00372426914403515023,

-0.00401758339589206037,

-0.00396552243911391968,

-0.00361260040642286942,

-0.00302275934850281588,

-0.00227183262978483810,

-0.00143985087643164547,

-0.00060390347390355543,

0.00016807215420674485,

0.00082198636181094744,

0.00132047316723110306,

0.00164402370315303676,

0.00179040834146424756,

0.00177264320346589275,

0.00161587466189913056,

0.00135353441100539811,

0.00102328234592731862,

0.00066312368845244907,

0.00030798909476912798,

-0.00001293682510553911,

-0.00027801060226038471,

-0.00047394205960644612,

-0.00059570185143487998,

-0.00064571514288907139,

-0.00063244246076861747,

-0.00056859799668502748,

-0.00046928092845412454,

-0.00035014240393270056,

-0.00022576911258506512,

-0.00010848469207435182,

-0.00000759789304004645,

0.00007100030788214359,

0.00012480878984804007,

0.00015427457426005363,

0.00016217394460579164,

0.00015289813142450526,

0.00013164614075849069,

0.00010370219401218202,

0.00007385936428750537,

0.00004600704326009998,

0.00002293128218759731,

0.00000624787589657057,

-0.00000344890126458452,

-0.00005348506844816352,

}; // коэффициенты фильтра

float FilterRes[numTaps + blockSize] = {0};

float pSrc[blockSize], pDst[blockSize];

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

float ADCin[2048] = {0};

float ADCout[2048] = {0};

float ADCout1[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);

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

HAL\_Delay(10);

arm\_rfft\_fast\_init\_f32(&RS, 1024/4);

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

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

hdma\_adc1.XferCpltCallback = ADC\_ConvCplt;

hdma\_adc1.XferHalfCpltCallback = ADC\_HalfConvCplt;

arm\_fir\_decimate\_init\_f32(&S, numTaps, 4, FilterCoefs, FilterRes,blockSize);

}

void AppWork()

{

// HAL\_Delay(100);

if(f == 0)

return;

f = 0;

arm\_fir\_decimate\_f32 (&S, ADCin, ADCout, blockSize);

uint32\_t index;

float harm;

arm\_rfft\_fast\_f32 (&RS, ADCout, ADCout1, 0);

arm\_cmplx\_mag\_f32 (ADCout1, ADCout1, 512);

// arm\_fill\_f32(0,&ADCout[76], 435);

ADCout1[0]=0;

ADCout1[1]=0;

arm\_max\_f32 (ADCout1, 512, &harm, &index);

ssd1306\_UpdateScreen();

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%10 + '0';

s[3] = 0;

s[4] = 0;

s[5] = 0;

s[6] = 0;

ssd1306\_SetCursor(1, 26);

ssd1306\_WriteString(s, 16, White);

}

