

**Kauno technologijos universitetas**

Elektros ir elektronikos fakultetas

**Skaitmeniniai filtrai**

2 Laboratorinis darbas

|  |
| --- |
|  |
| **Žygimantas Marma, EMEI-2 gr.**  Studentas |
|  |
| **Doc. Šarūnas Kilius**  Dėstytojas |
|  |

**Kaunas, 2023**

**Kaunas, 2023**

Turinys

[1. Dažnių persidengimas (angl. aliasing) 4](#_Toc147826595)

[1.1. Praktinė dalis 4](#_Toc147826596)

[2. Išvados 6](#_Toc147826597)

[3. Priedai 7](#_Toc147826598)

**Darbo tikslas**: Susipažinti su DSP signalų generavimu ir analize įterptinėse sistemose panaudojant Audio signalą.

**Laboratorinio darbo uždaviniai:**

1. Suprojektuoti žemo dažnio, juostinius bei aukšto dažnio FIR filtrus naudojant MATLAB *FilterDesigner* įrankį.
2. Eksportuoti suprojektuotų filtrų koeficientus.
3. Išbandyti filtrų veikimą STM32F407 mikrovaldikliu. Palyginti rezultatus su MATLAB.

# FIR filtrų kūrimas

Patobulinkite pateiktame pavyzdyje esančią sinusinio signalo skaičiavimui skirtą funkciją, įtraukdami kintamajį arba konstantą dažniui nurodyti.

## Praktinė dalis

Signalo dedamosios yra 1kHz + 3kHz + 10kHz

Atkirtos dažnis (Fc) buvo pasirinktas 980Hz.

## Lowpass filtro kūrimas MATLAB aplinkoje

A screenshot of a computer

Description automatically generated

A screenshot of a graph

Description automatically generated

Galima dar dazni imesti kazka tokio bet ne same

A screenshot of a graph

Description automatically generated

## Lowpass filtro testavimas su STM32 valdikliu

Norint testuoti sukurtus filtrus STM32 makete buvo naudojamas pavyzdyje pateiktas projektas. Vienintelis pakeitimas buvo sinuso funkcijoje, kad joje atsirastų 3 dedamosios:

2 lentelė pagrindinis laboratorinio darbo programinis kodas

|  |
| --- |
| /\*\*\* Test Signal generation \*\*\*/    for(int index = 0; index < TEST\_LENGTH\_SAMPLES; index++)    {      // 1kHz + 5kHz + 10 kHz      testInput\_f32[index] = (30000\*sin(21\*2\*PI\*index/TEST\_LENGTH\_SAMPLES) + 20000\*sin(109\*2\*PI\*index/TEST\_LENGTH\_SAMPLES) + 15000\*sin(220\*2\*PI\*index/TEST\_LENGTH\_SAMPLES));    } |

Laiko asyje

FFT

## Highpass filtro kūrimas MATLAB aplinkoje

A screenshot of a computer

Description automatically generated

A screenshot of a graph

Description automatically generated

## Highpass filtro testavimas su STM32 valdikliu

Laiko asyje

FFT

## Band Pass filtro kūrimas MATLAB aplinkoje

A screenshot of a computer

Description automatically generated

A screenshot of a screen

Description automatically generated

A screenshot of a graph

Description automatically generated

## Band Pass filtro testavimas su STM32 valdikliu

Laiko asyje

FFT

# Slenkančio vidurkio (Moving Average) filtras

Sadasd

Asdasd

# Išvados

1. Atlikus
2. Tyrimas parodė
3. Filtro atsako

# Priedai

**1 lentelė.** MATLAB programa sukurtų filtrų koeficientų konvertavimui į C kodą

|  |
| --- |
| % FIR testas  % Filtro koeficientai saugomi faile    clear;  % HD - this is a generated MATLAB filter function  % Hd = FIR\_filter\_30ord; % example  Hd = lowpassFinal;  % Hd = highPass;  %Hd = bandpass;      Num2 = Hd.Numerator;  Den = zeros(size(Num2)); Den(1)=1;  Fs = 48000;  t = 0:1/Fs:0.003;  % Input signal: 1kHz + 5kHz + 10kHz  x = sin(2\*pi\*1000\*t) + sin(2\*pi\*5000\*t) + sin(2\*pi\*10000\*t);    y = filter(Num2,Den,x);  subplot(2,1,1); plot(t,x);  title('Input to the Filter');  subplot(2,1,2); plot(t,y);  title('Output of the Filter'); xlabel('time (sec)');    % ----------------------------------------------  % Save created filter coefs to a file  fid=fopen('koeficientai\_STM32.txt','w');  % number of filter coefs  fprintf(fid, 'NUM\_TAPS=%d \r\n', [length(Num2)]');  % save coefs + comma  fprintf(fid, [' const float32\_t firCoeffs32[NUM\_TAPS] = { \r\n']);  for j=1:length(Num2)-1  fprintf(fid, '%f,', [Num2(j)]');  end  % Last coef after the comma  fprintf(fid, '%f', [Num2(length(Num2)-1)]');    fprintf(fid, [' \r\n };']);  fclose(fid);      plotFrequencySpectrum(x, y, Fs);    function plotFrequencySpectrum(x, y, Fs)  % Compute the FFT for the input signal (x)  N = length(x);  f = Fs \* (0:(N/2)) / N;  X = fft(x);  P\_x = abs(X / N);    % Compute the FFT for the filtered signal (y)  Y = fft(y);  P\_y = abs(Y / N);    % Plot the frequency spectra  figure;  subplot(2, 1, 1);  plot(f, P\_x(1:N/2+1));  title('Frequency Spectrum - Input Signal');  xlabel('Frequency (Hz)');  ylabel('Amplitude');    % Display only positive frequencies  axis([0 Fs/2 0 max(P\_x)]);    subplot(2, 1, 2);  plot(f, P\_y(1:N/2+1));  title('Frequency Spectrum - Output Signal');  xlabel('Frequency (Hz)');  ylabel('Amplitude');    % Display only positive frequencies  axis([0 Fs/2 0 max(P\_y)]);  end |

2 lentelė pagrindinis laboratorinio darbo programinis kodas

|  |
| --- |
| /\* Includes ------------------------------------------------------------------\*/  #include "main.h"  #include "usb\_device.h"  /\* Private includes ----------------------------------------------------------\*/  /\* USER CODE BEGIN Includes \*/  #include "stm32f4\_discovery\_audio.h"  #include "arm\_math.h"  } |

Sugeneruoti low-pass filtro koeficientai

Sugeneruoti low-pass filtro koeficientai

Sugeneruoti low-pass filtro koeficientai