Kauno technologijos universitetas
Elektros ir elektronikos fakultetas

IIR filtrai

3 Laboratorinis darbas

Žygimantas Marma, EMEI-2 gr.
Studentas

Doc. Šarūnas Kilius
Dėstytojas

Kaunas, 2023

Turinys

Kaunas, 2023

Darbo tikslas: Susipažinti ir suprojektuoti su IIR ir audio efektą.

Laboratorinio darbo uždaviniai:

1. Suprojektuoti žemo dažnio, juostinius bei aukšto dažnio IIR filtrus naudojant MATLAB *FilterDesigner* įrankį.
2. Eksportuoti suprojektuotų filtrų koeficientus į STM32 mikrovaldiklį ir išbandyti filtrų veikimą mikrovaldiklyje bei palyginti rezultatus su MATLAB.
3. Atlikti filtrų impulsinės charakteristikos tyrimą.
4. Išbandyti pateiktus audio efektus su pasirinktu kūriu.
5. Implementuoti naują audio efektą.

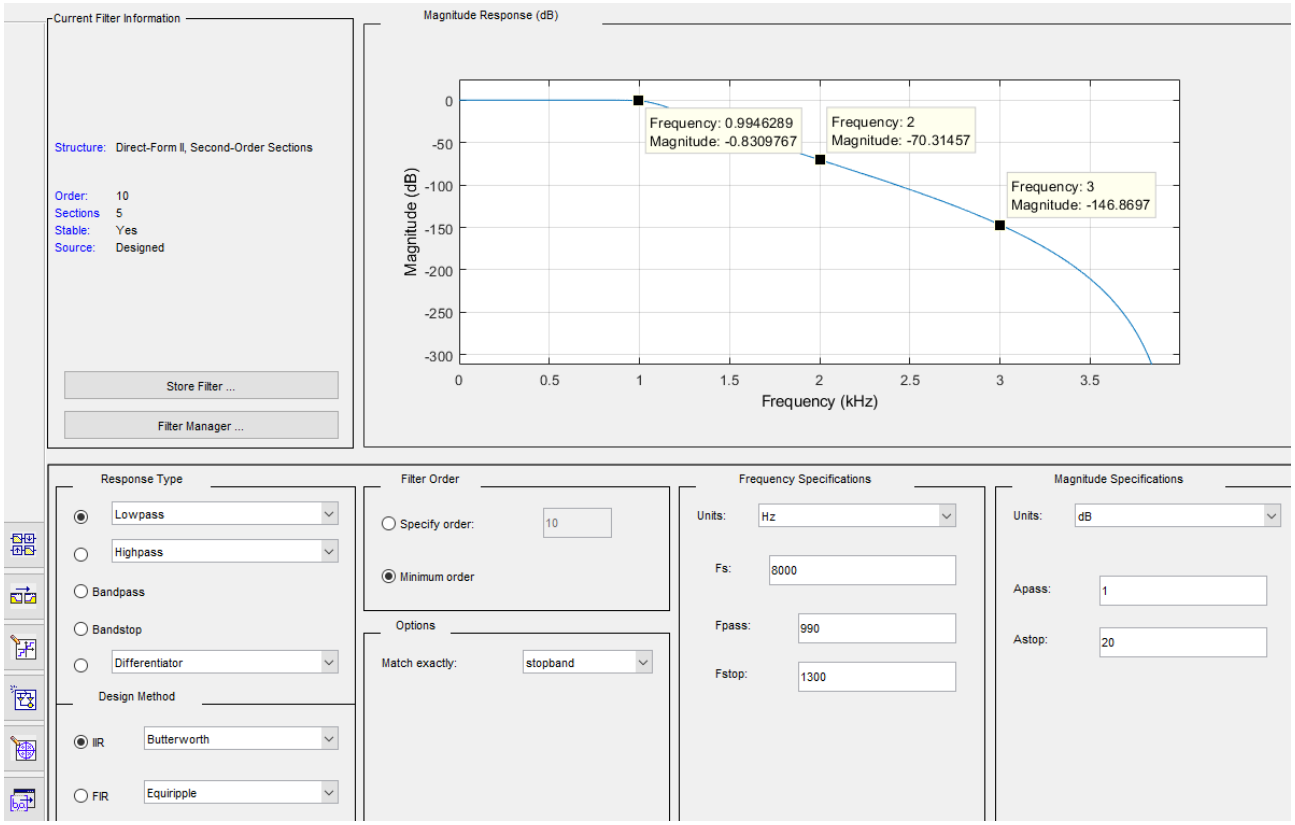
1. IIR filtrų kūrimas

1.1. Praktinė dalis

Testuojant filtrų veikimą pradinis poliharmoninis sinusinis signalas buvo pasirinktas kurio dedamosios yra 1kHz + 2kHz + 3kHz.

1.2. Žemų dažnių filtro kūrimas MATLAB aplinkoje

Žemų dažnių filtrui buvo pasirinktas „Butterworth“ metodas, nustatyti F_{pass} ir F_{stop} analogiškai 990Hz ir 1300Hz, kad nebūtų per aukšta flitro eilė. Galiausiai gautas 10 laipsnio fitltras su charakteristika pavaizduota 1 pav.

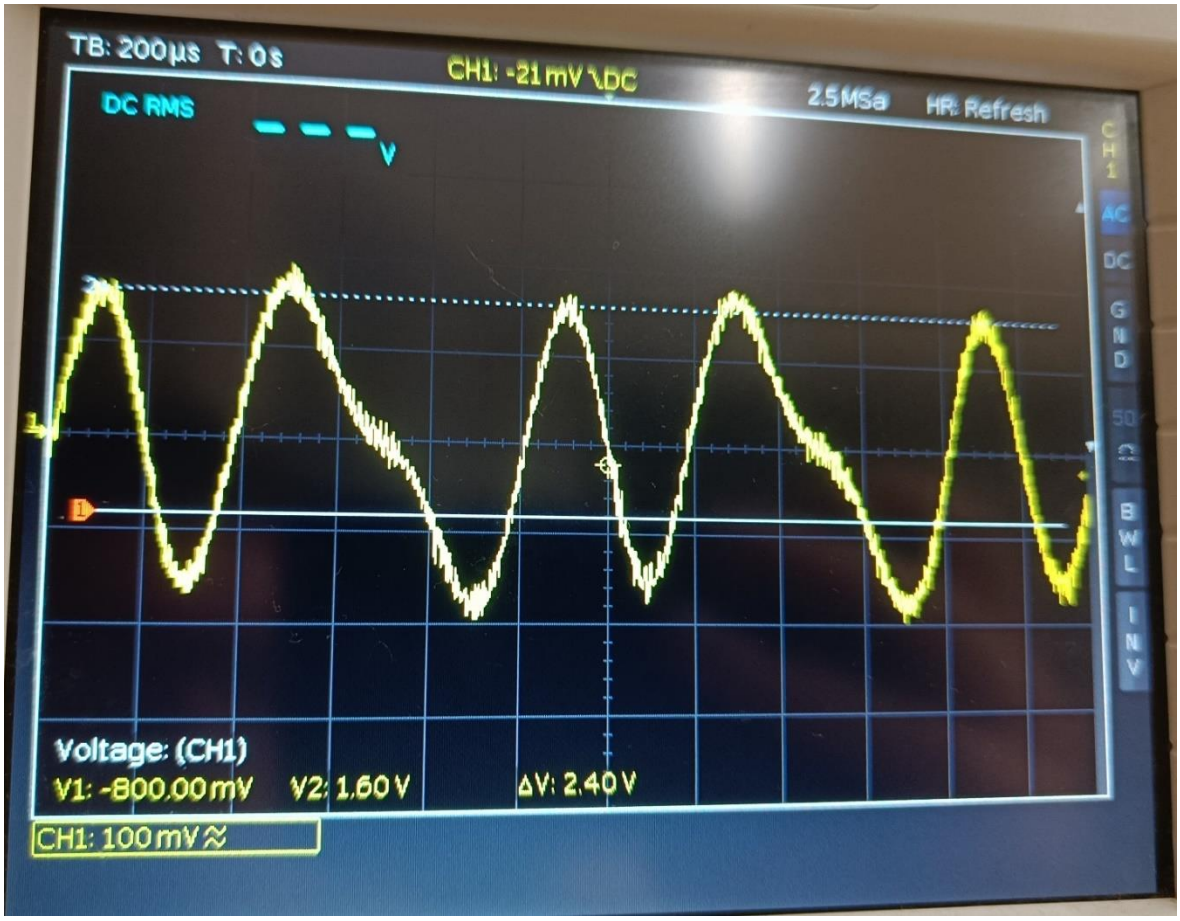


1 pav. Sukurtas žemų dažnių filtras MATLAB pagalba

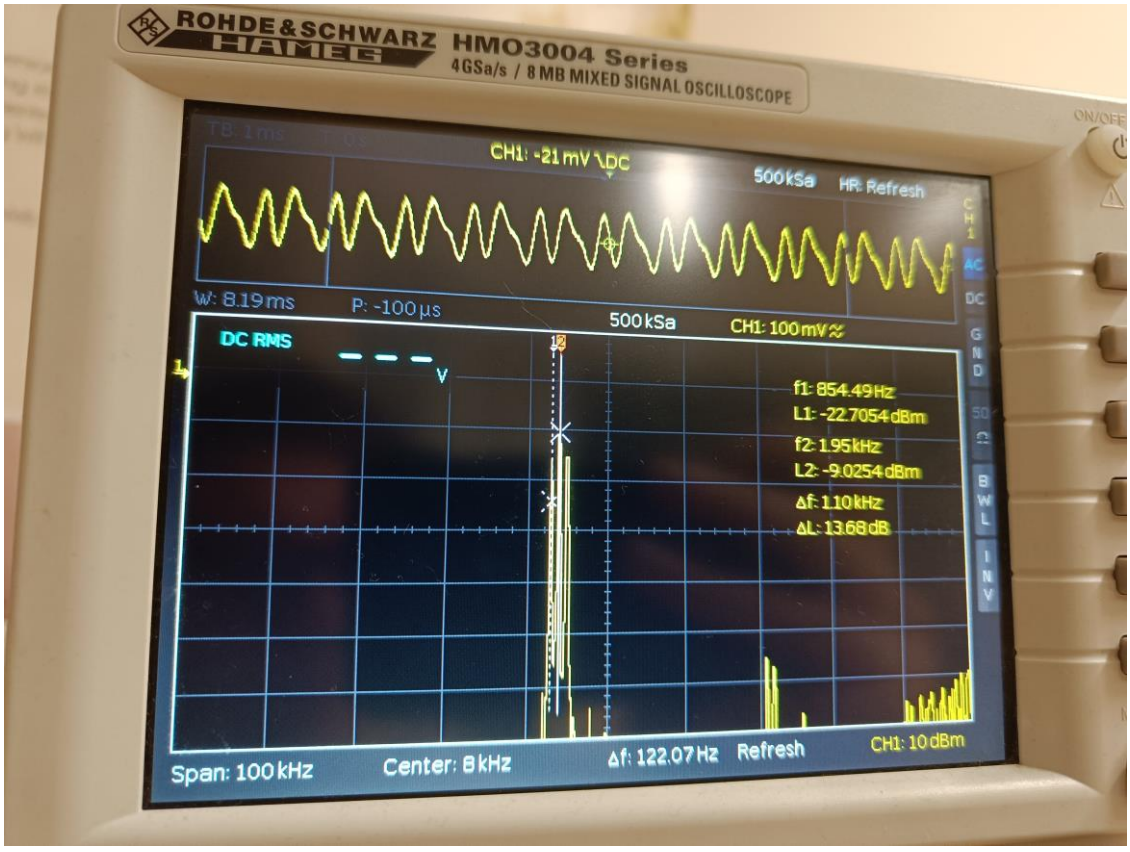
1.3. Žemų dažnių filtro testavimas su STM32 valdikliu

1 lentelė Mikrovaldiklio signalo programinis kodas

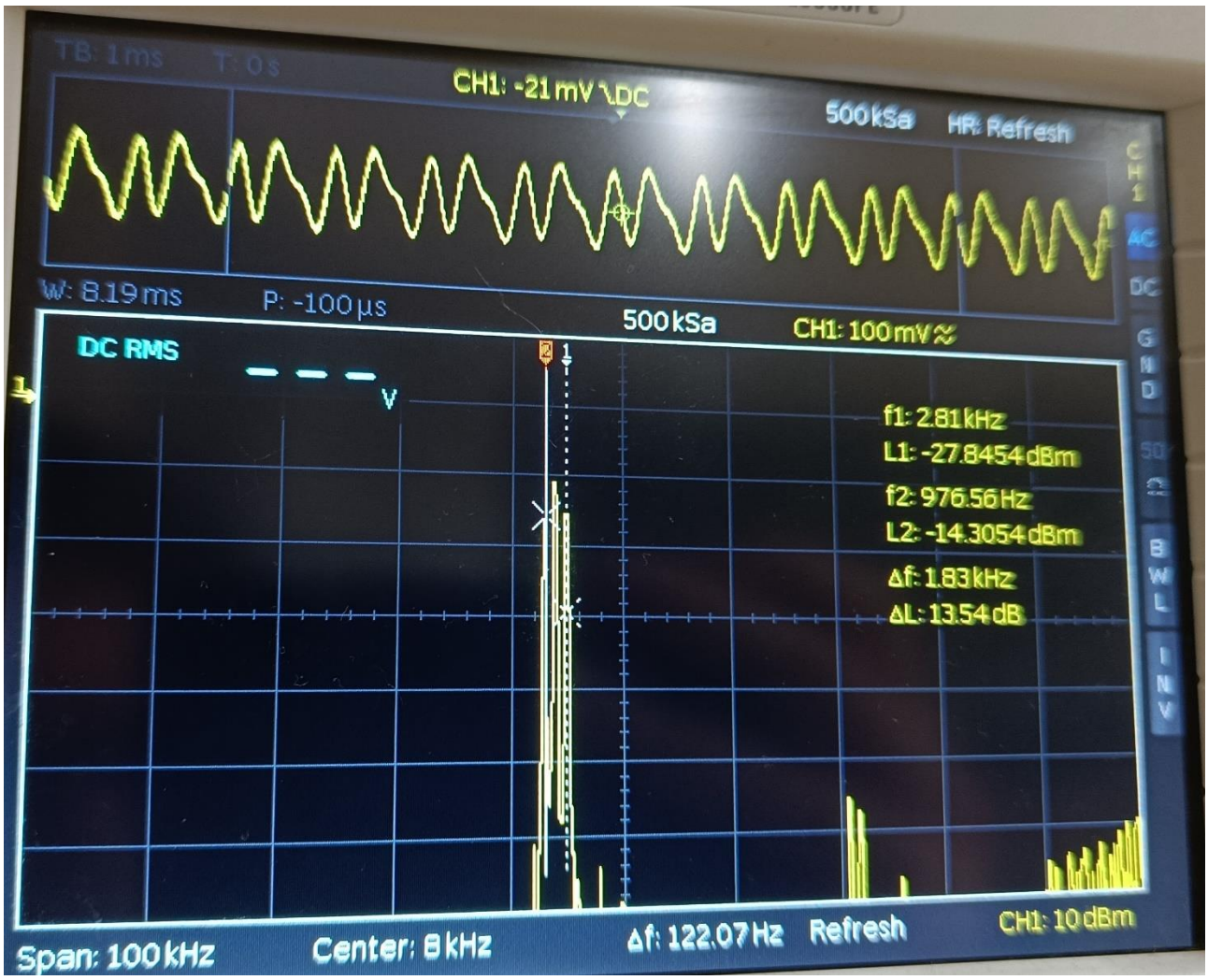
```
/*** Test Signal generation ***/
for(int index = 0; index < TEST_LENGTH_SAMPLES; index++)
{
  //Test signal 1000 Hz + 2000 Hz + 3000 Hz
  testInput_f32[index] = ( 30000*sin(32*2*PI*index/TEST_LENGTH_SAMPLES) +
  20000*sin(64*2*PI*index/TEST_LENGTH_SAMPLES) +
  15000*sin(96*2*PI*index/TEST_LENGTH_SAMPLES));
}
```



2 pav. Nefiltruotas signalas laiko srityje

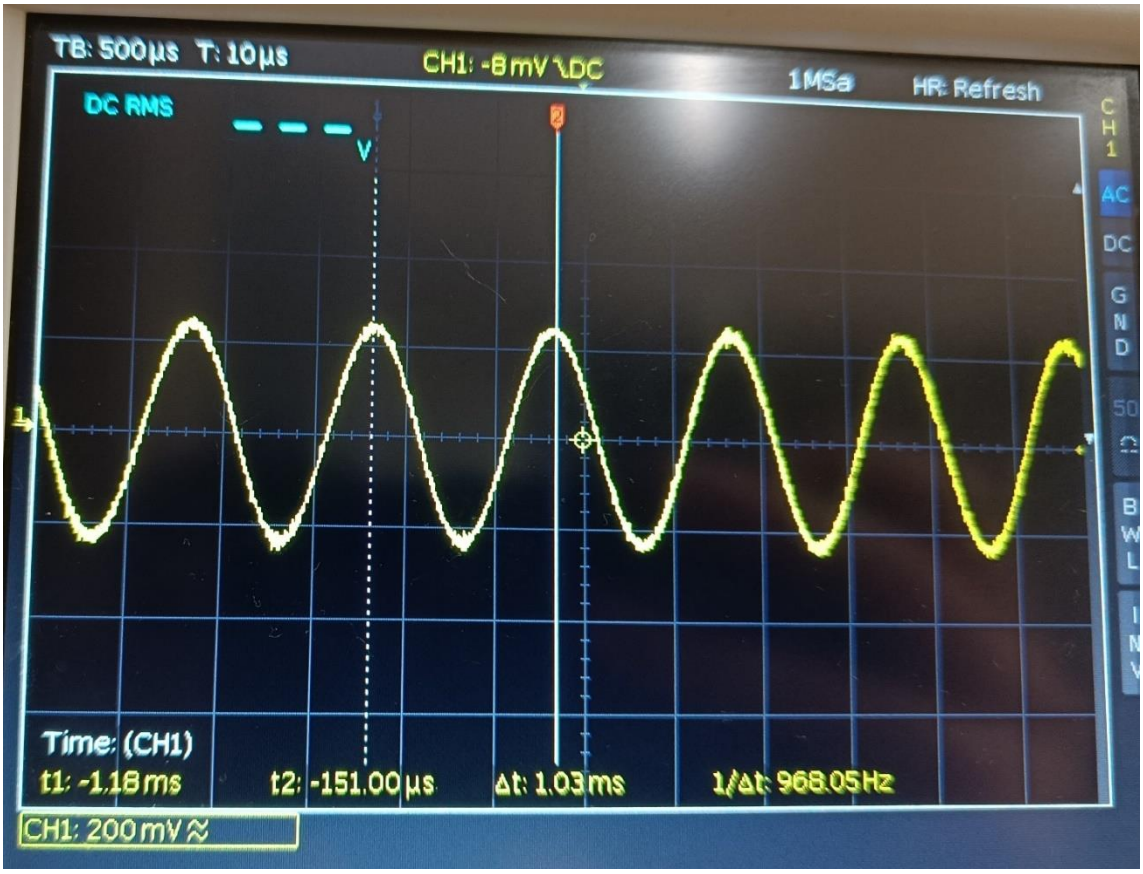


3 pav. Nefiltruoto signalo spektras

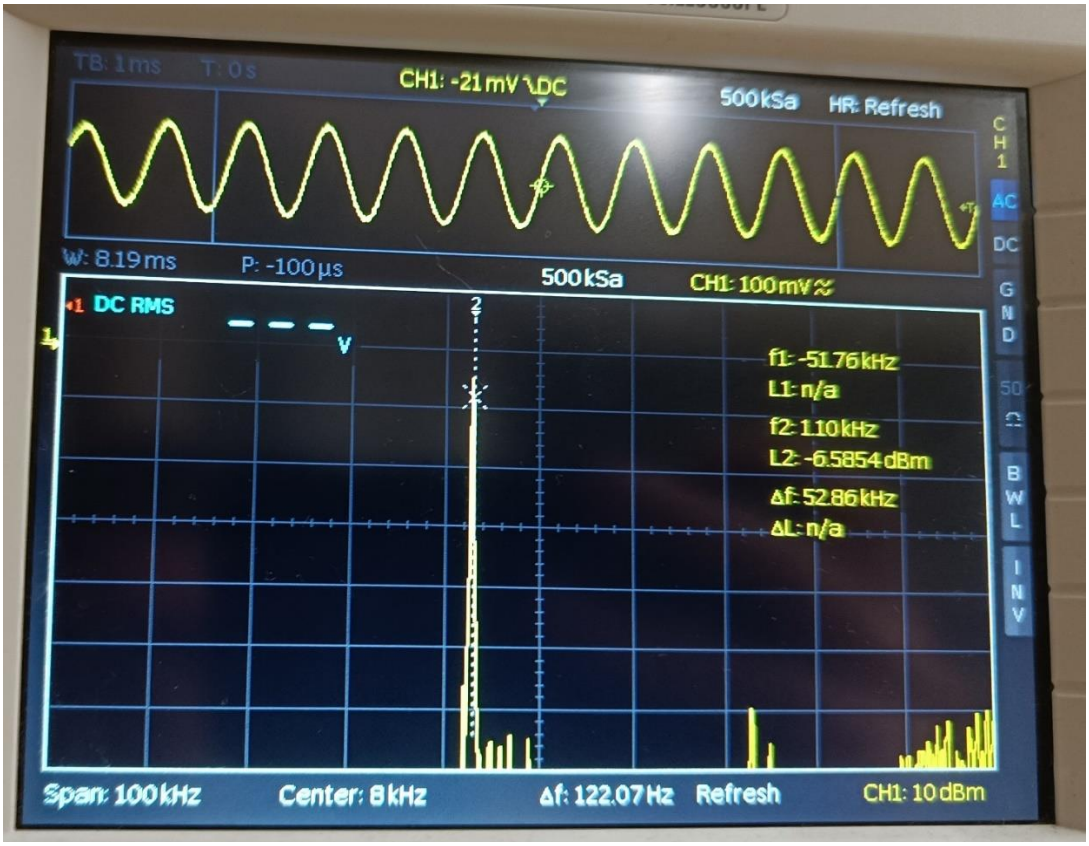


4 pav. Nefiltruoto signalo spektras

Pritaikius suprojektuotą žemų dažnių IIR filtrą gauti tokie rezultatai:

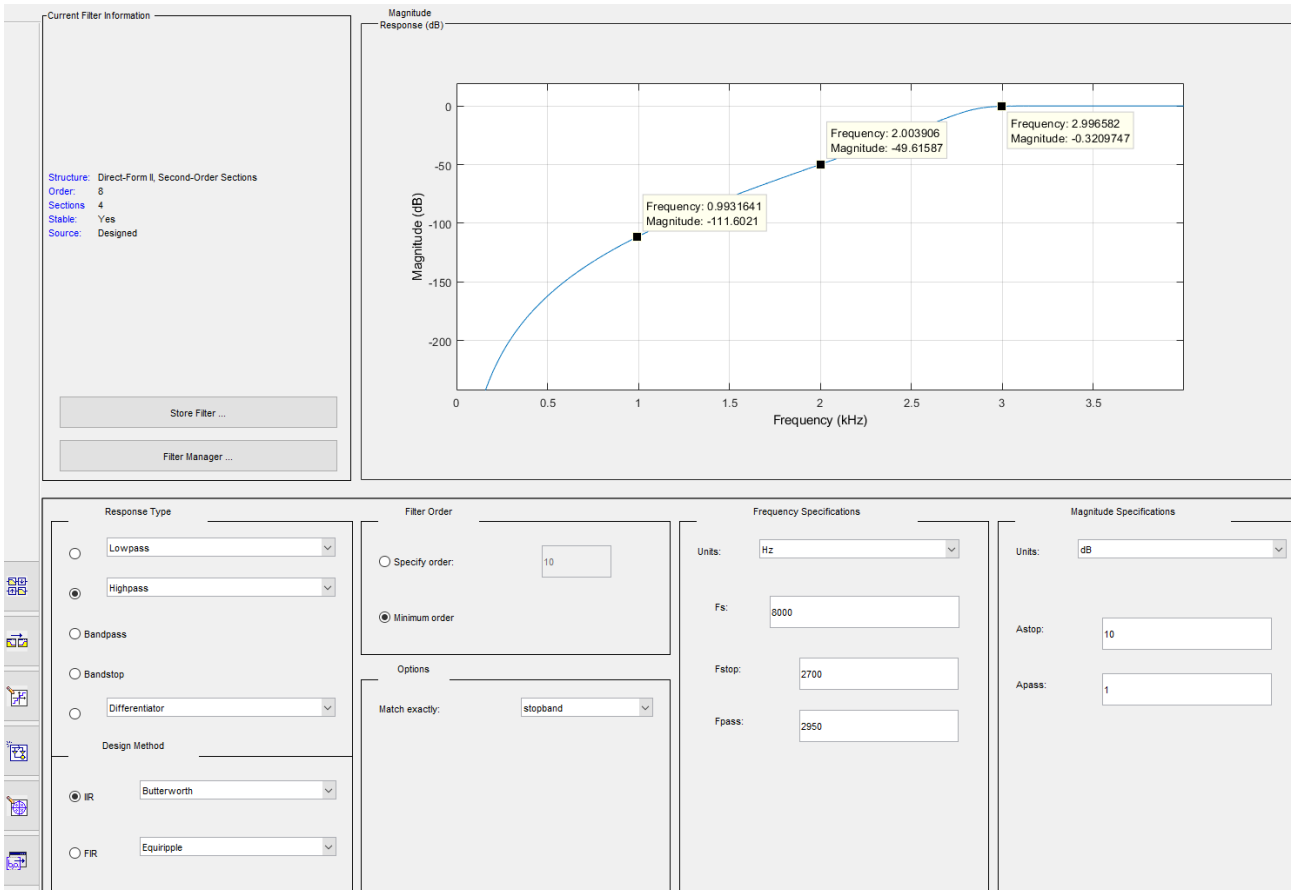


5 pav. Filtruotas signalas laiko srityje (žemų dažniu filtras)



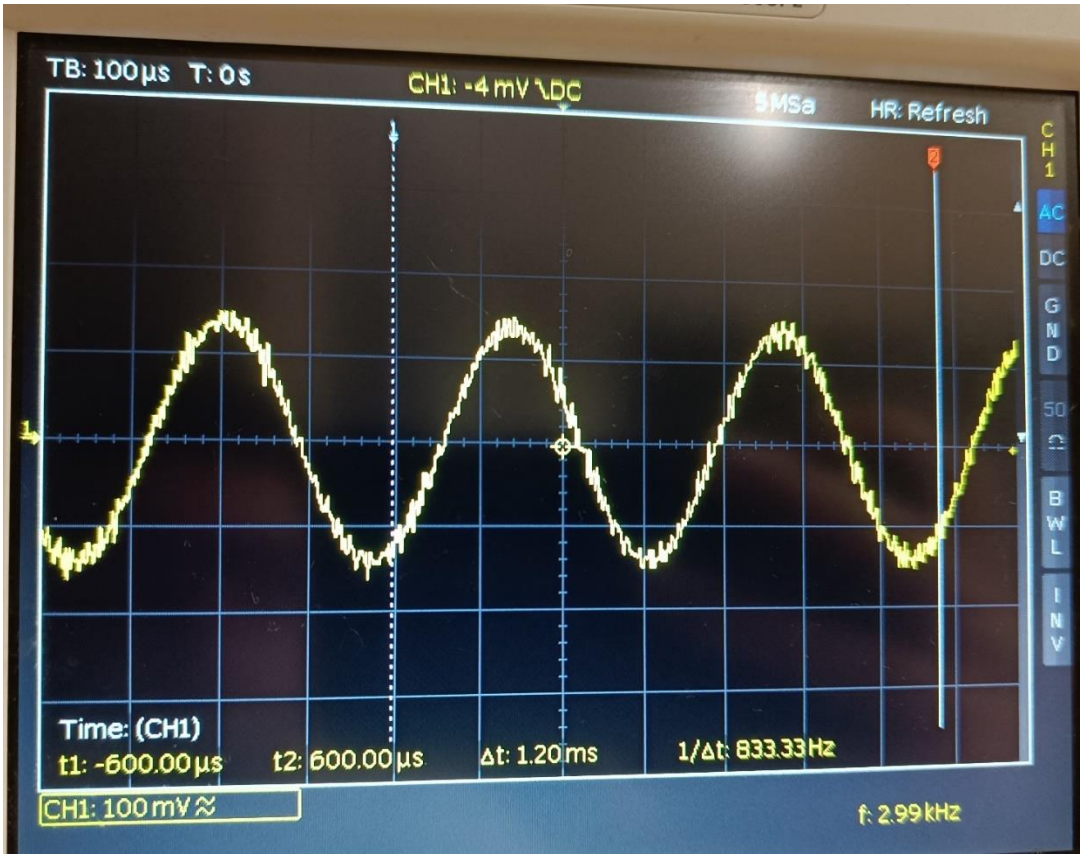
6 pav. Filtruoto signalo spektras (žemų dažniu filtras)

1.4. Aukštų dažnių filtro kūrimas MATLAB aplinkoje

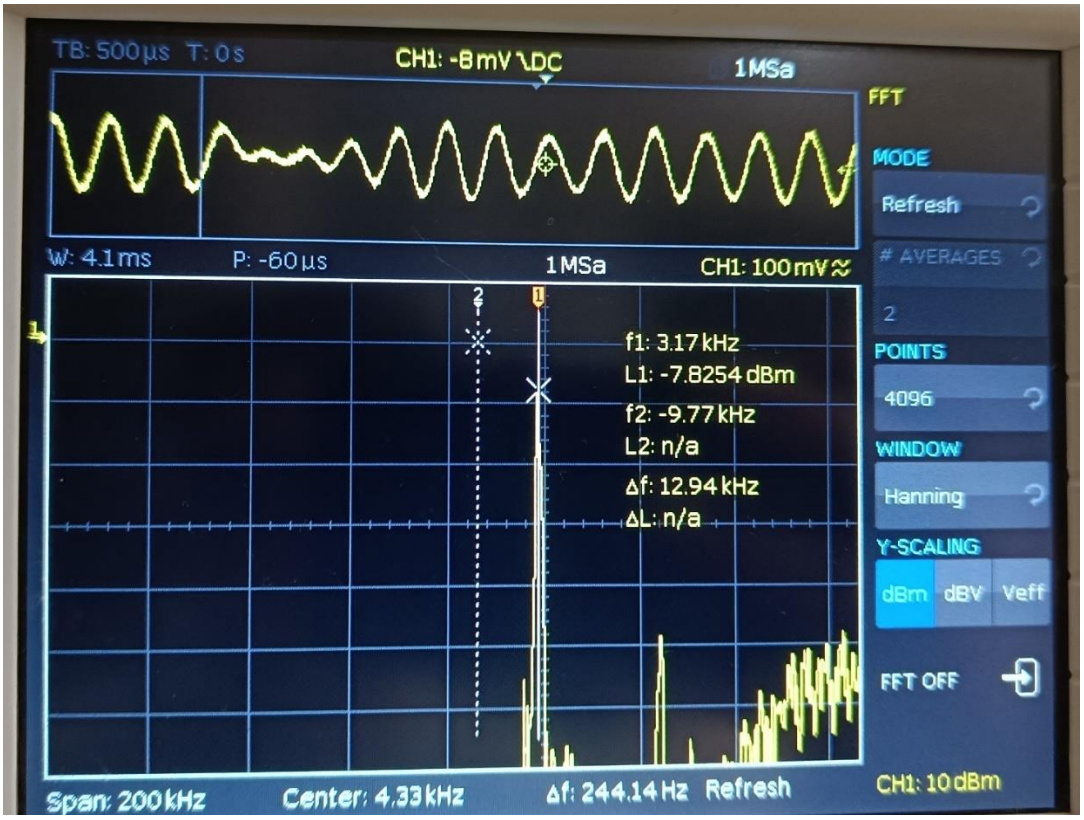


7 pav. Sukurtas aukštų dažnių filtras MATLAB pagalba

1.5. Aukštų dažnių filtro testavimas su STM32 valdikliu

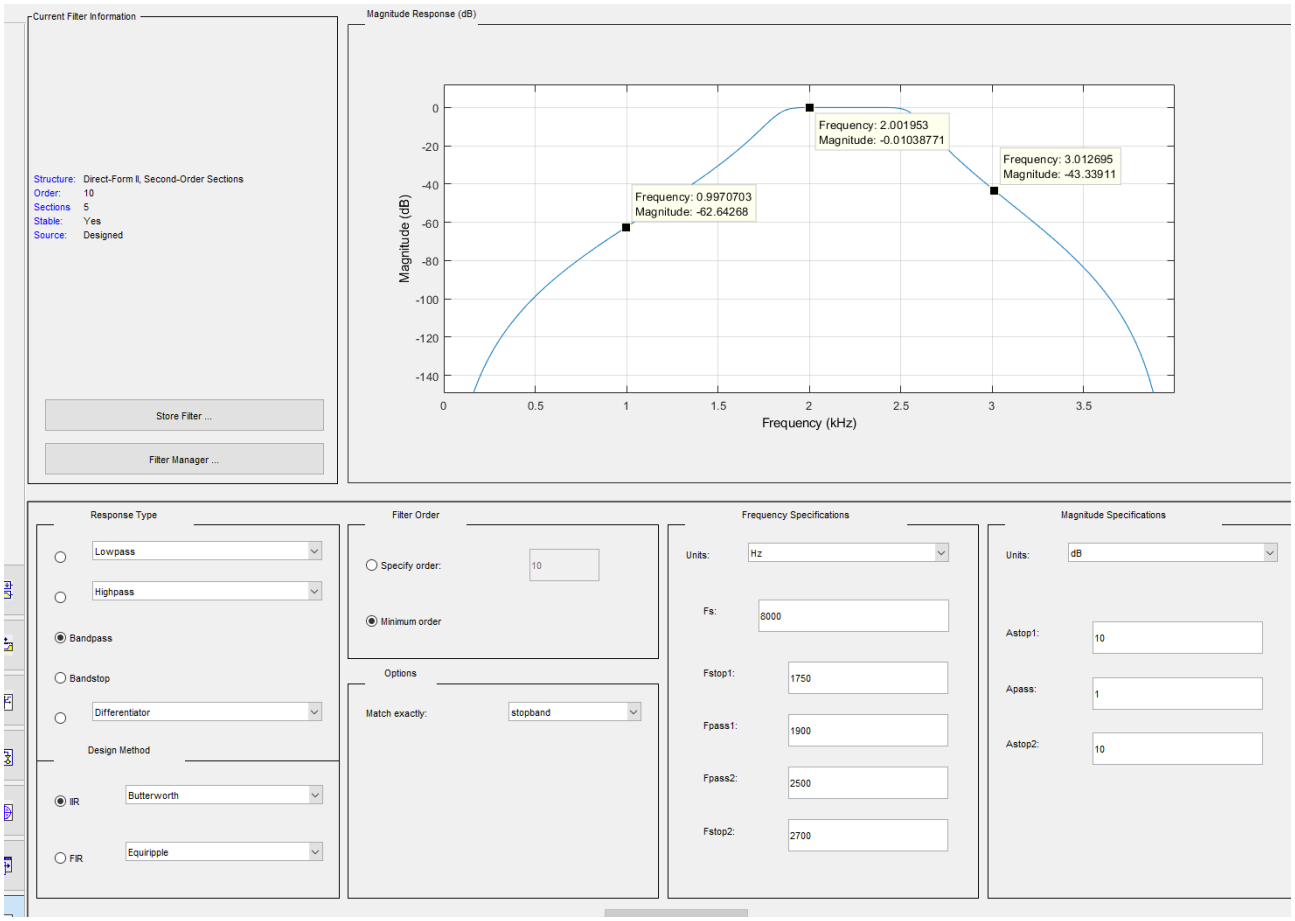


8 pav. Filtruotas signalas laiko srityje (aukštų dažniu filtras)



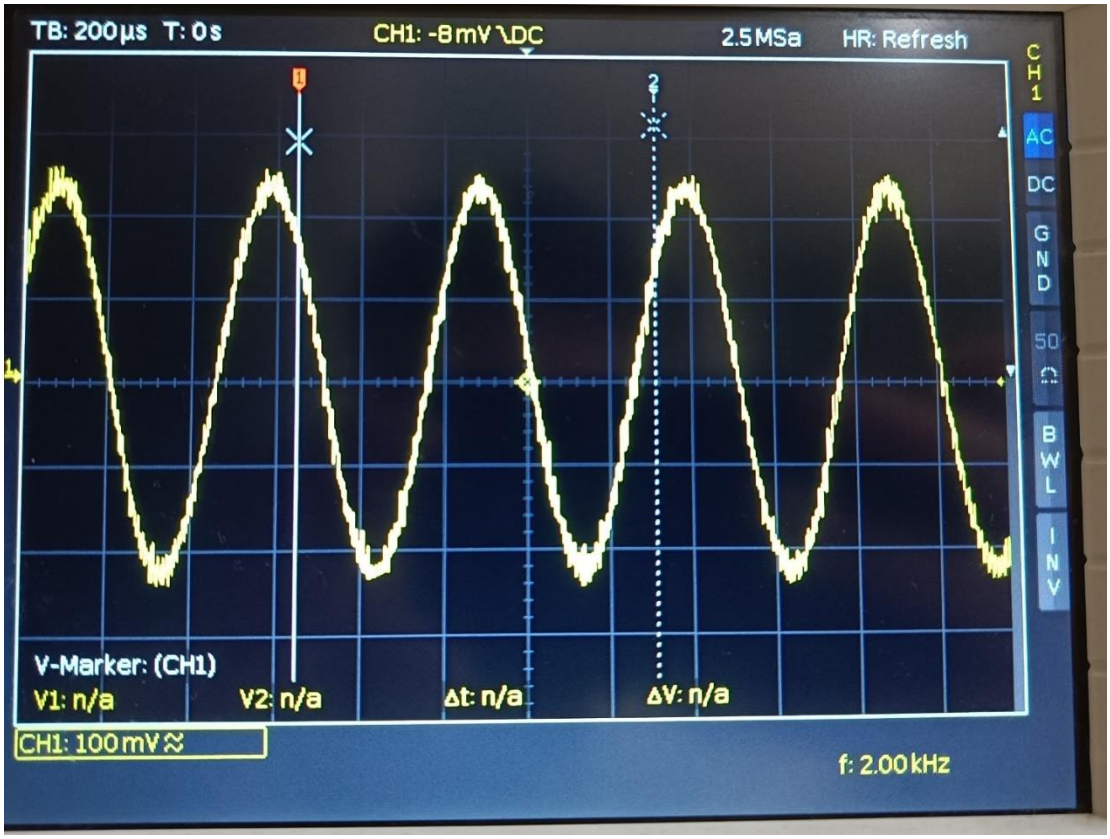
9 pav. Filtruoto signalo spektras (žemų dažniu filtras)

1.6. Juostinio filtro kūrimas MATLAB aplinkoje

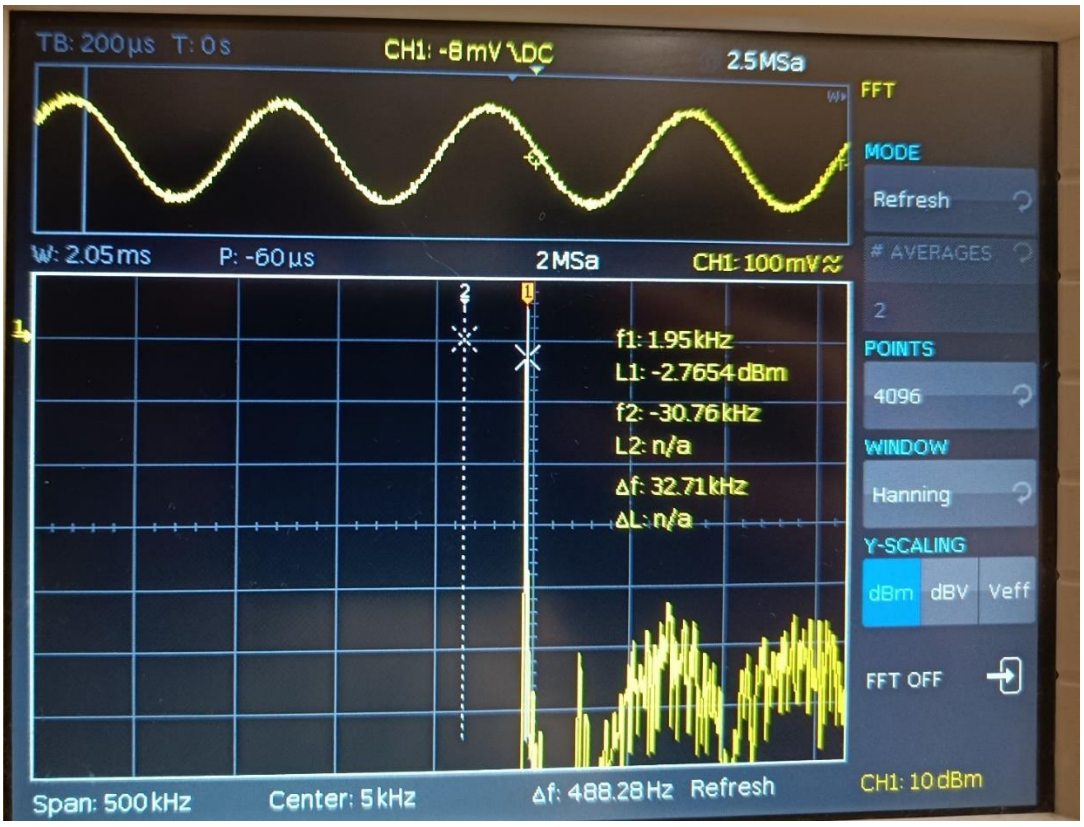


10 pav. Sukurtas juostinis filtras MATLAB pagalba

1.7. Juostinio filtro testavimas su STM32 valdikliu



11 pav. Filtruotas signalas laiko srityje (juostinis filtras)



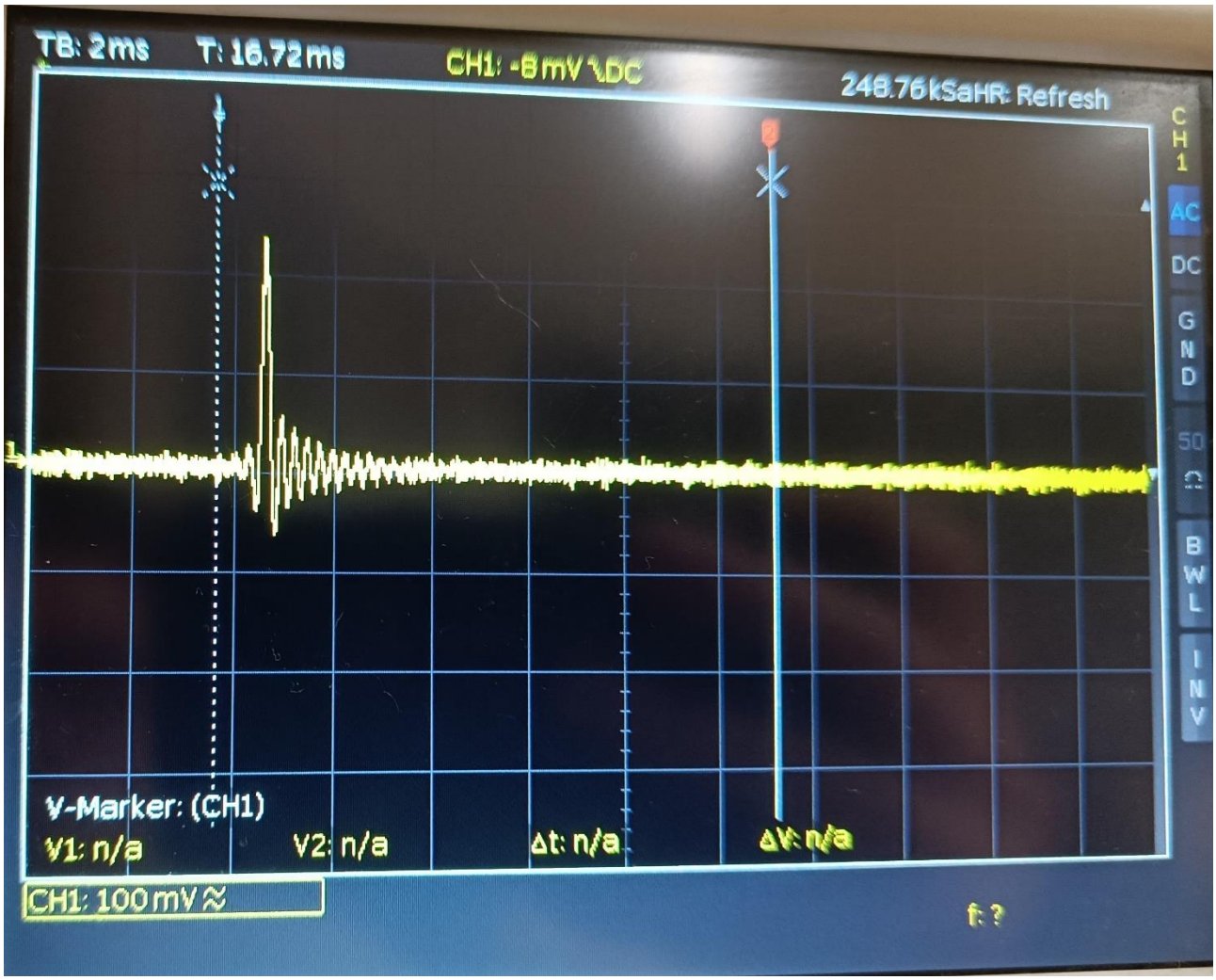
12 pav. Filtruoto signalo spektras (juostinis filtras)

2. Filtrų impulsinės charakteristikos

Vienetins signalas mikrovaldiklyje buvo gautas su šiuo kodu:

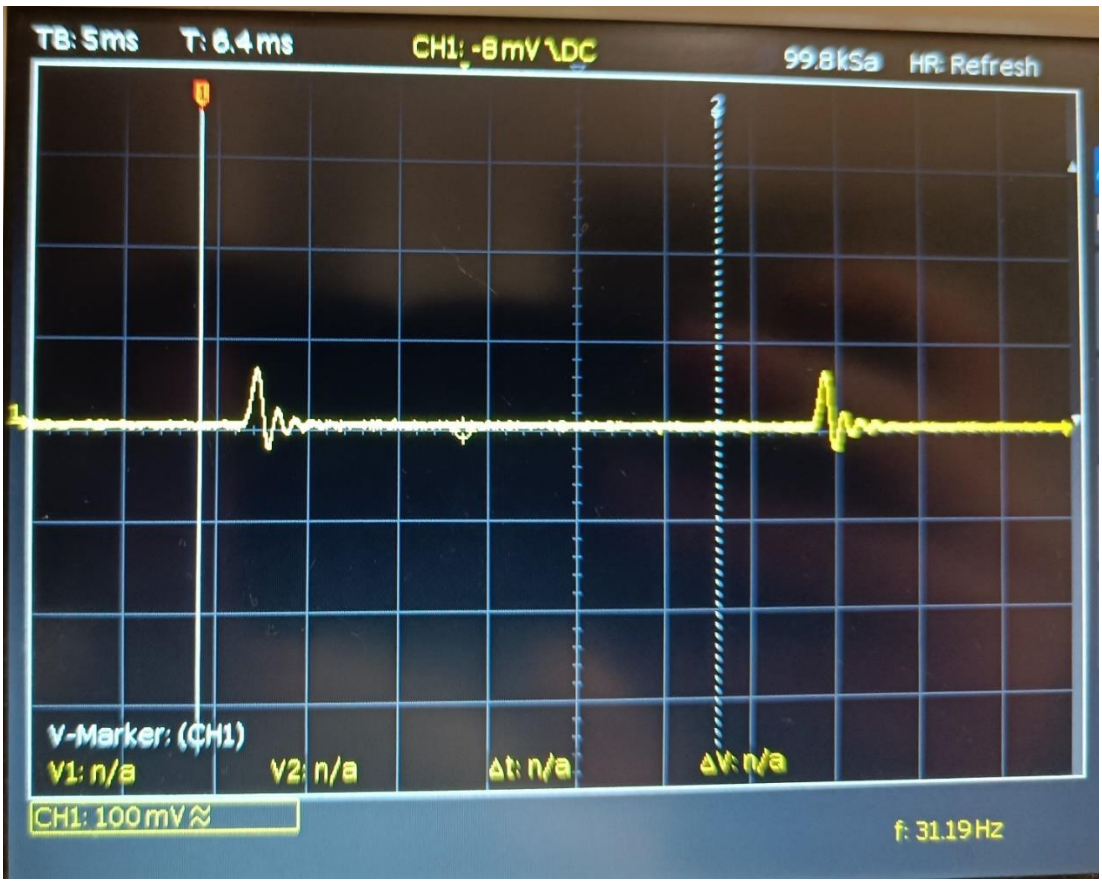
2 lentelė Vienetinio impulso funkcija

```
      for(int index = 0; index < TEST_LENGTH_SAMPLES; index++)
      {
          if (index == 10) {
            testInput_f32[index] = 20000;
          } else {
            testInput_f32[index] = 0;
          }
      }
```

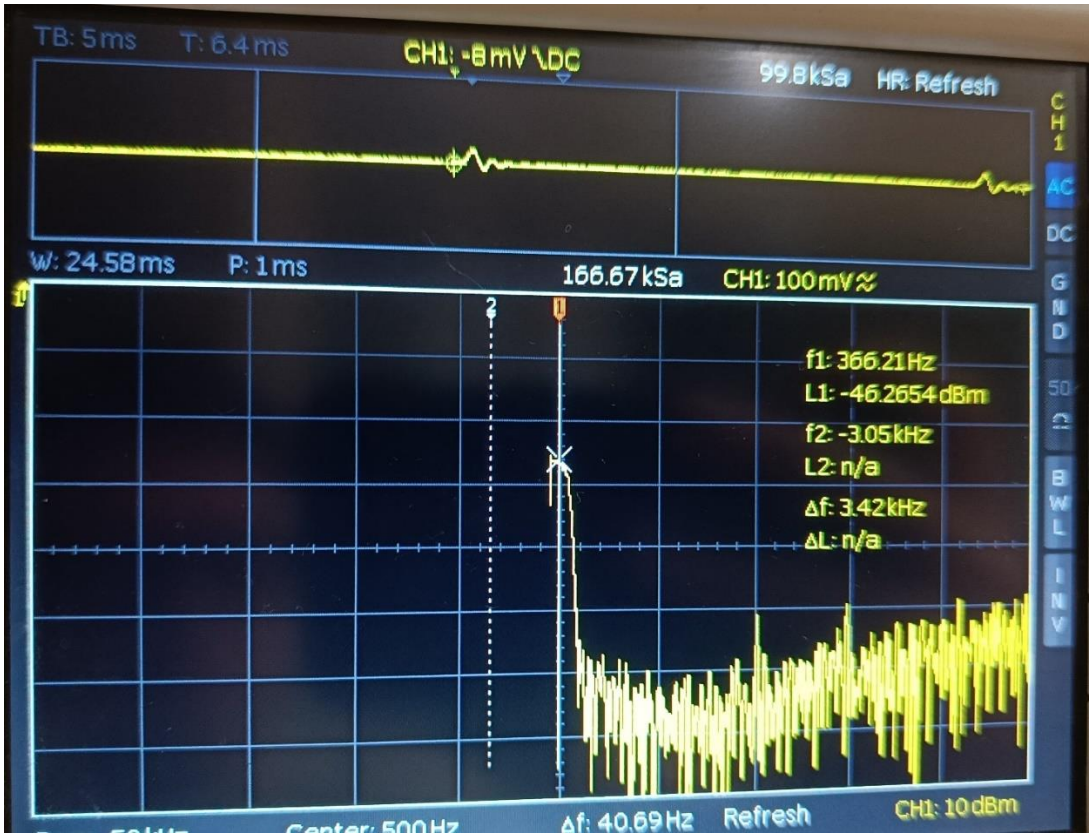


13 pav. Sugeneruotas vienetinis signalas.

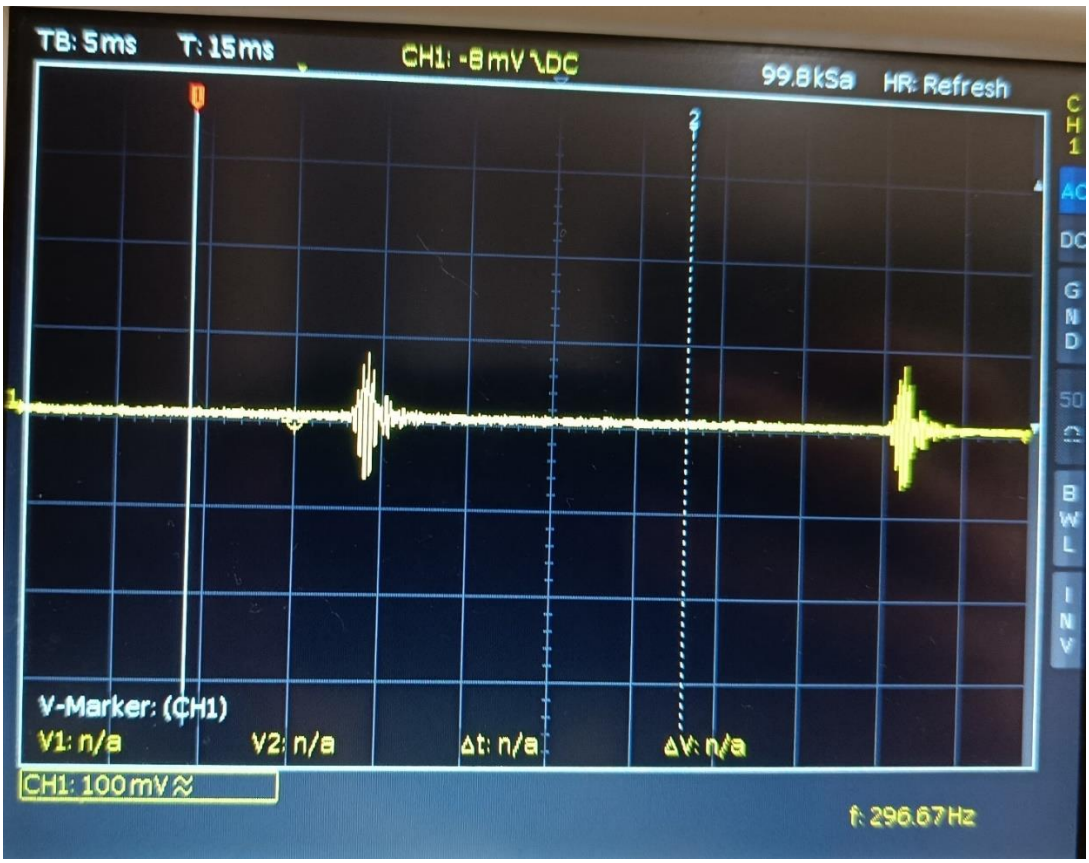
Galima matyti, jog vienetinio signalo forma nėra tobula.



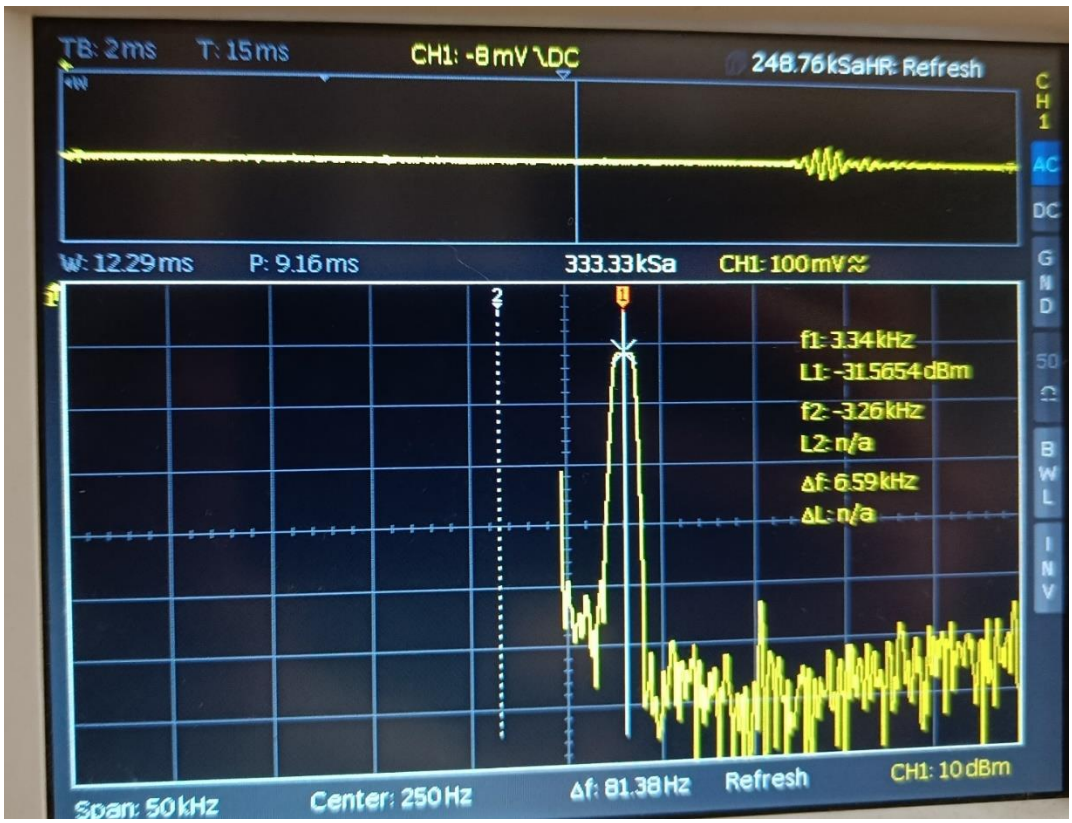
14 pav. Žemų dažnių filtras su vienetinio impulso ivestimi



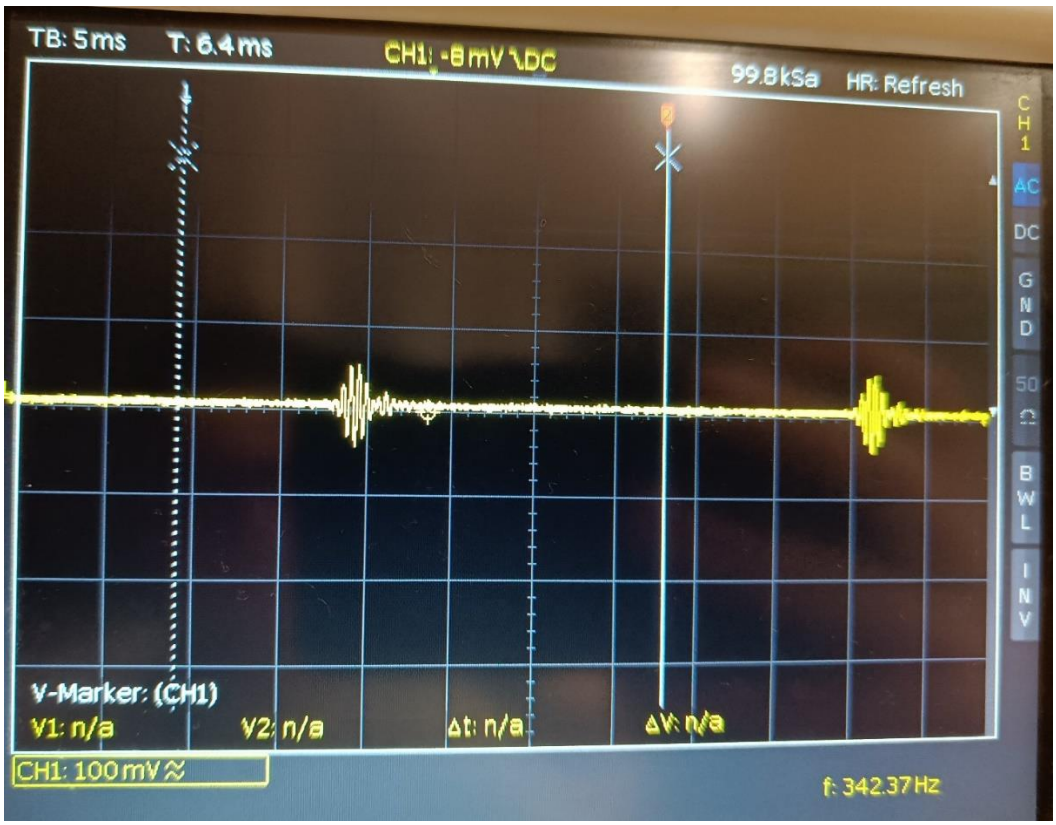
15 pav. Žemų dažnių filtro spektras su vienetinio impulso ivestimi



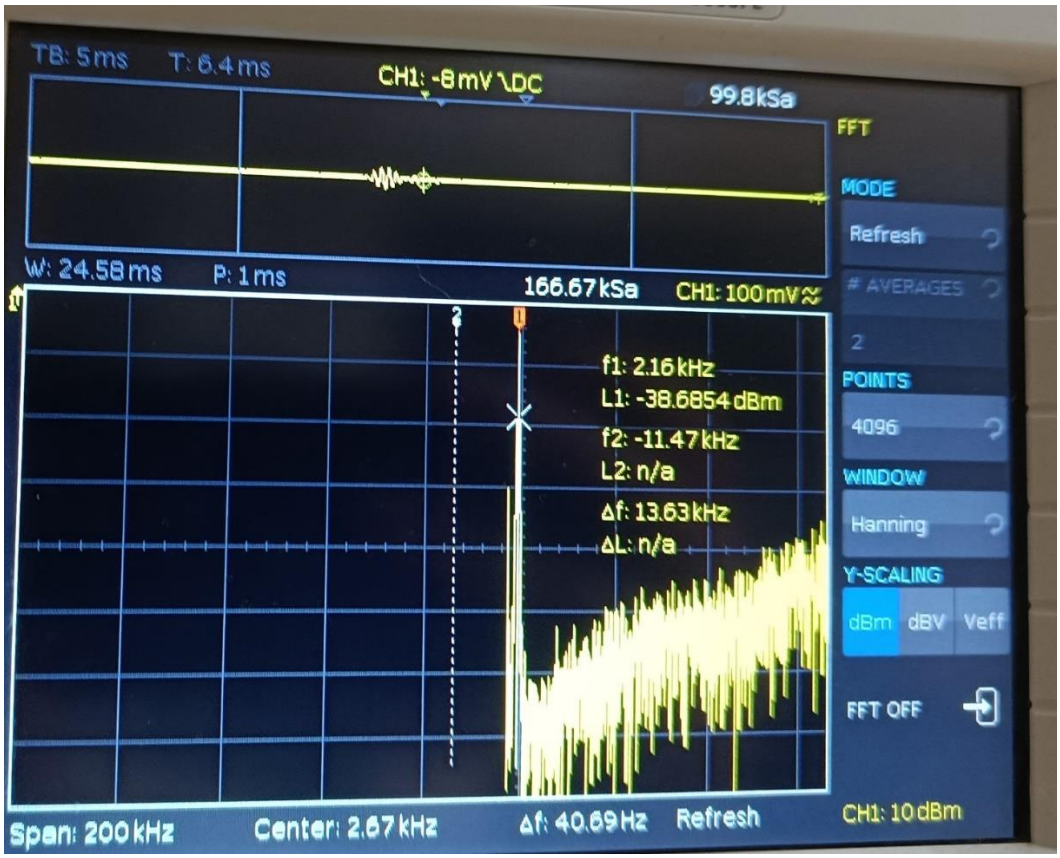
16 pav. Aukštų dažnių filtras su vienetinio impulso ivestimi



17 pav. Aukštų dažnių filtro spektras su vienetinio impulso ivestimi



18 pav. Juostinis filtras su vienetinio impulso ivestimi



19 pav. Juostinio filtro spektras su vienetinio impulso ivestimi

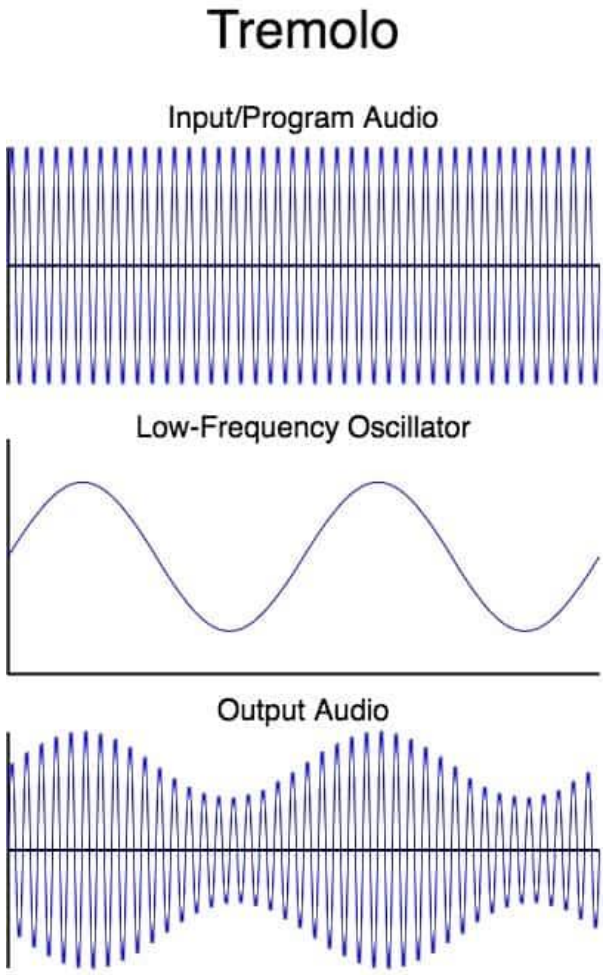
Galima matyti kad visi filtrai yra stabilūs. Analizuojant spektrus matyti kad yra praleidžiamos tik tos dedamosios kurias projektuoti filtrai turėjo praleisti.

3. Audio efektai

Eektų analizei buvo naudojamas The Weeknd daina „Blinding Lights“. Išklausisu pavyzdinius muzikos efektus buvo pastebėta, jog žemų dažnių filtras slopina signalą. Klausantis distortion – iškraipymo efekto galima išgirsti daugiau triukšmo ir atrodo stipresni audio signalą. Galiausiai aido signalas veikė taip kaip jo pavadininmas ir nurodo.

3.1. Tremolo efektas

„Tremolo“ laikui bėgant moduliuoja signalo amplitudę, sukurdamas pulsuojantį ar virpantį garsą. Audio tremolo efektu siekiama imituoti akustinį tremolo efektą, efektyviai ciklišKai keičiant garso signalo amplitudę. Taigi „Tremolo“ įrenginiai yra atsakingi už signalo lygių sumažinimą ir vėl įvedimą į įrenginio išvestį. Šie moduliavimo efekto įrenginiai naudoja žemo dažnio osciliatorius (LFO), kad valdytų amplitudės keitimo programą.



20 pav. Tremolo signalo vizualizacija

3 lentelė Tremolo efekto funkcija

```
// Tremolo effect function
int16_t Tremolo(int16_t InSample)
{
    // Update tremolo oscillator
    static float_t tremolo_phase = 0.0;
```

```
    tremolo_phase += tremolo_rate / SAMPLING_FREQ;
    if (tremolo_phase > 1.0) tremolo_phase -= 1.0;

    // Apply tremolo modulation to the input signal
    float_t tremolo_factor = 1.0 + tremolo_depth * sin(TWO_PI * tremolo_phase);
    float_t OutSampleF = InSample * tremolo_factor;

    return (int16_t)OutSampleF;
}
```

Gautas audio signalas pasižymėjo pulsavimu ir papildomu triukšmu.

4. Išvados

1. Laboratorinio darbo metu buvo sukurti ir eksportuoti į mikrovaldiklį žemų dažnių, aukštų dažnių ir juostinis IIR filtrai.
2. Tiek MATLAB simuliacijose, tiek realiame mikrovaldiklyje skurti filtrai veikė tinkamai ir pašalino nepageidaujamų dažnių dedamąsias.
3. Antroje darbo dalyje buvo išbandyti visi garso efektai su muzikos kūriniu, bei įgyvendintas Tremolo audio efektas.

5. Priedai

4 lentelė. MATLAB programa sukurtų filtrų koeficientų konvertavimui į C kodą

```
%
function STM32F4_iirsos_coeffs(coeff,gain)
%
num_sections=length(gain)-1;
fname = input('enter filename for coefficients ','s');
fid = fopen(fname,'wt');
fprintf(fid,'// %s\n',fname);
fprintf(fid,'// this file was generated using');
fprintf(fid,'\n// function STM32F4_iirsos_coeffs.m\n',fname);
fprintf(fid,'\n#define NUM_SECTIONS %d\n',num_sections);
% first write the numerator coefficients b
% i is used to count through sections
fprintf(fid,'\nfloat b[NUM_SECTIONS][3] = { \n');
for i=1:num_sections
if i==num_sections
fprintf(fid,'{%2.8E, %2.8E, %2.8E} };\n',...
coeff(i,1)*gain(i),coeff(i,2)*gain(i),coeff(i,3)*gain(i));
else
fprintf(fid,'{%2.8E, %2.8E, %2.8E},\n',...
    coeff(i,1)*gain(i),coeff(i,2)*gain(i),coeff(i,3)*gain(i));
end
end
% then write the denominator coefficients a
% i is used to count through sections
fprintf(fid,'\nfloat a[NUM_SECTIONS][3] = { \n');
for i=1:num_sections
if i==num_sections
fprintf(fid,'{%2.8E, %2.8E, %2.8E} };\n',...
coeff(i,4),coeff(i,5),coeff(i,6));
else
fprintf(fid,'{%2.8E, %2.8E, %2.8E},\n',...
coeff(i,4),coeff(i,5),coeff(i,6));
end
end
fclose(fid);
```

5 lentelė. Sugeneruoti žemų dažnių filtro koeficientai

```
#define NUM_SECTIONS 5

float b[NUM_SECTIONS][3] = {
{1.48120186E-01, 2.96240372E-01, 1.48120186E-01},
{1.23633875E-01, 2.47267751E-01, 1.23633875E-01},
{1.08391408E-01, 2.16782815E-01, 1.08391408E-01},
{9.94805969E-02, 1.98961194E-01, 9.94805969E-02},
{9.53591618E-02, 1.90718324E-01, 9.53591618E-02} };

float a[NUM_SECTIONS][3] = {
```

```
{1.00000000E+00, -1.19927188E+00, 7.91752620E-01},
{1.00000000E+00, -1.00101569E+00, 4.95551188E-01},
{1.00000000E+00, -8.77603320E-01, 3.11168950E-01},
{1.00000000E+00, -8.05455930E-01, 2.03378318E-01},
{1.00000000E+00, -7.72086263E-01, 1.53522910E-01} };
```

6 lentelė. Sugeneruoti aukštų dažnių filtro koeficientai

```
#define NUM_SECTIONS 4

float b[NUM_SECTIONS][3] = {
{1.66790712E-01, -3.33581424E-01, 1.66790712E-01},
{1.33825671E-01, -2.67651342E-01, 1.33825671E-01},
{1.16241827E-01, -2.32483653E-01, 1.16241827E-01},
{1.08524666E-01, -2.17049332E-01, 1.08524666E-01} };

float a[NUM_SECTIONS][3] = {
{1.00000000E+00, 1.06621340E+00, 7.33376244E-01},
{1.00000000E+00, 8.55483628E-01, 3.90786313E-01},
{1.00000000E+00, 7.43078504E-01, 2.08045810E-01},
{1.00000000E+00, 6.93746382E-01, 1.27845047E-01} };
```

7 lentelė. Sugeneruoti juostinio filtro koeficientai

```
#define NUM_SECTIONS 5

float b[NUM_SECTIONS][3] = {
{2.59047000E-01, 0.00000000E+00, -2.59047000E-01},
{2.59047000E-01, 0.00000000E+00, -2.59047000E-01},
{2.33543913E-01, 0.00000000E+00, -2.33543913E-01},
{2.33543913E-01, 0.00000000E+00, -2.33543913E-01},
{2.25616630E-01, 0.00000000E+00, -2.25616630E-01} };

float a[NUM_SECTIONS][3] = {
{1.00000000E+00, 7.66866044E-01, 8.51990426E-01},
{1.00000000E+00, -2.12333336E-01, 8.39565783E-01},
{1.00000000E+00, 5.35714765E-01, 6.36464291E-01},
{1.00000000E+00, -2.47588879E-02, 6.19336420E-01},
{1.00000000E+00, 2.49164944E-01, 5.48766740E-01} };
```