Kauno technologijos universitetas

Panevėžio technologijų ir verslo fakultetas

Plačiajuosčio ultragarsinio generatoriaus įgyvendinimo galimybių tyrimas

Baigiamasis magistro studijų projektas

Vaidas Živelis

Projekto autorius

doc. dr. Gailius Vanagas

Vadovas

Panevėžys, 2023

Kauno technologijos universitetas

Panevėžio technologijų ir verslo fakultetas

Plačiajuosčio ultragarsinio generatoriaus įgyvendinimo galimybių tyrimas

Baigiamasis magistro studijų projektas

Valdymo technologijos (6211EX014)

Vaidas Živelis

Projekto autorius

doc. dr. Gailius Vanagas

Projekto vadovas

Recenzentas / Recenzentė

Panevėžys, 2023

Kauno technologijos universitetas

Panevėžio technologijų ir verslo fakultetas

Vaidas Živelis

Plačiajuosčio ultragarsinio generatoriaus įgyvendinimo galimybių tyrimas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Vaidas Živelis

Patvirtinta elektroniniu būdu

Kauno technologijos universitetas

Panevėžio technologijų ir verslo fakultetas

TVIRTINU
TVKC vadovė
Doc. dr. Nida Kvedaraitė

Baigiamojo magistro projekto užduotis

Diplomantui **Vaidui Živeliui**

Baigiamojo projekto tema (lietuvių kalba)	Plačiajuosčio ultragarsinio generatoriaus įgyvendinimo galimybių tyrimas
Baigiamojo projekto tema (anglų kalba)	Feasibility Study of a Broadband Ultrasonic Generator

Patvirtinta 2023 m. balandžio 14 d. dekano potvarkiu Nr. V25-13-11

Parengto baigiamojo projekto įkėlimo į Moodle aplinką terminas iki 2023 m. gegužės 29 d.

Duomenys, reikalavimai ir sąlygos baigiamajam projektui

Baigiamojo projekto užduotys / uždaviniai, kurie turi būti atskleisti projekte

1. Išnagrinėti esamu realaus laiko signalų generavimo ir apdorojimo metodus, naudojamus standartiniuose mikroprocesoriuose. 2. Aprašyti sinusinės formos signalų formavimo standartiniame procesoriuje algoritmą. 3. Pateikti elektrostatinio plačiajuosčio ultragarsinio generatoriaus su stiprintuvu, veikiančiu rezonansiniame režime, matematinį modelį. 4. Pateikti plačiajuosčio ultragarsinio generatoriaus amplitudines dažnines charakteristikas (ADCH). 5. Palyginti plačiajuosčio ultragarsinio generatoriaus amplitudinės dažninės charakteristikos modeliavimo ir eksperimentinių matavimų rezultatus.

Vadovas doc. dr. Gailius Vanagas
(vadovo pareigos, vardas, pavardė, parašas)

Užduotį gavau Vaidas Živelis
(studento vardas, pavardė, parašas)

2023 m. balandžio 20 d.

Živelis, Vaidas. Plačiajuosčio ultragarsinio generatoriaus įgyvendinimo galimybių tyrimas. Magistro baigiamasis projektas / vadovas doc. dr. Gailius Vanagas; Kauno technologijos universitetas, Panevėžio technologijų ir verslo fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): elektronikos inžinerija, technologijos mokslai.

Reikšminiai žodžiai: ultragarsas, DSP, „Arduino“, keitiklis, generatorius, ESP32.

Panevėžys, 2023. 79 p.

Santrauka

Šio baigiamojo magistro projekto tikslas – išanalizuoti plačiajuosčio ultragarsinio generatoriaus realizavimo galimybes, naudojant signalų generavimo ir apdorojimo standartiniuose mikroprocesoriuose metodus. Projekto uždaviniai: išnagrinėti esamus realaus laiko signalų generavimo ir apdorojimo metodus, naudojamus standartiniuose mikroprocesoriuose; aprašyti sinusinės formos signalų formavimo standartiniame procesoriuje algoritmą; pateikti elektrostatinio plačiajuosčio ultragarsinio generatoriaus su stiprintuvu, veikiančiu rezonansiniame režime, matematinį modelį; pateikti plačiajuosčio ultragarsinio generatoriaus amplitudines dažnines charakteristikas (ADCH); palyginti plačiajuosčio ultragarsinio generatoriaus amplitudinės dažninės charakteristikos modeliavimo ir eksperimentinių matavimų rezultatus.

Darbe naudota programinė įranga: *Arduino IDE, LTspice, REW, MATLAB, SigmaStudio.*

Analitinėje dalyje išnagrinėta mokslinė literatūra apie laiko signalų generavimo ir apdorojimo metodus, naudojamus standartiniuose mikroprocesoriuose.

Eksperimentinėje dalyje sukurtas sinusinės formos signalų generatorius, naudojantis standartinį mikroprocesorių (ESP32). Plačiajuosčio ultragarso generatoriaus bandymams sukurtas D klasės stiprintuvo su *LC* filtru išėjime modelis.

Tiriamojoje dalyje atlikti realaus plačiajuosčio ultragarso generatoriaus naudojančio 16 ričių signalų generavimo bandymai, bandymų rezultatai palyginti su modeliavimo rezultatais.

Gauti projekto rezultatai:

1. Atlikus mokslinės literatūros analizę nustatyta, kad realaus laiko signalų apdorojimo ir generavimo, naudojamuose skaitmeniniuose mikroprocesoriuose, taikomi įvairūs algoritmai ir transformacijos: FFT, DFT, SWFT, DTFT, CZT, DWT, Furjė transformacija bei filtrai: aukšto, žemo dažnio, dažnių juostos, begalinio ir baigtinio impulsų atsakų filtrai, palengvinantys skaičiavimus apdorojant signalus. Signalų generavimui naudojamos įvairios bibliotekos ir vidiniai mikroprocesorių registrai, leidžiantys pasiekti aukštus dažnius (90 kHz ir daugiau).
2. Įdiegus ir išbandžius sinusinės formos signalų generatoriaus algoritmą į standartinį procesorių (ESP32), buvo išgautas nuo 1 Hz iki 90 kHz generuojamas signalas. Viršijus 90 kHz sinuso formos signalas pradeda deformuotis, dėl valdiklyje integruoto skaitmeninio – analoginio keitiklio ribotos greitaveikos.
3. Sudarius elektrostatinio plačiajuosčio ultragarsinio generatoriaus su stiprintuvu, veikiančiu rezonansiniame režime, matematinį modelį, D klasės stiprintuvo išėjimo *LC* filtrui apskaičiuoti šešiolikos ričių induktyvumai (1895 µH, 1585 µH, 1345 µH, 1105 µH, 900 µH, 725 µH, 582 µH,

460 µH, 370 µH, 303 µH, 253 µH, 212 µH, 183 µH, 159 µH, 139 µH, 123 µH), leidžiantys turėti rezonansinį stiprinimą visame užsiduotame dažnių diapazone (20–85 kHz).

4. Išmatavus plačiajuosčio ultragarsinio generatoriaus amplitudines dažnines charakteristikas (ADCH), nustatyta, kad teoriniai ričių matavimai leidžia pasiekti rezonansinį stiprinimą visame dažnių diapazone (20–85 kHz), o charakteristikų netolygumas neviršija 6 dB (pirmo bandymo metu – 4,18 dB, antro – 4,1 dB), kas garso technikoje yra priimtinas dydis.
5. Tarpusavyje palyginus plačiajuosčio ultragarsinio generatoriaus amplitudinės dažninės charakteristikos (ADCH) eksperimentinių matavimų rezultatus su modeliavimo rezultatais, nustatyta, kad tarp bandymų ir teorinių skaičiavimų yra stipri koreliacija (0,7998 su pirmu bandymu, 0,7807 su antru).

Živelis, Vaidas. Feasibility Study of a Broadband Ultrasonic Generator. Master's Final Degree Project / supervisor doc. dr. Gailius Vanagas; Panevėžys Faculty of Technology and Business, Kaunas University of Technology.

Study field and area (study field group): Electronics Engineering, Technology Sciences.

Keywords: Ultrasound, DSP, „Arduino", Transducer, Generator, ESP32.

Panevėžys, 2023. 79 pages.

Summary

The aim of this final master's project is to analyze the possibilities of implementing a wideband ultrasound generator using signal generation and processing methods in standard microprocessors. The project objectives are as follows: examine existing real-time signal generation and processing methods used in standard microprocessors, describe the algorithm for generating sinusoidal signals in a standard processor, present a mathematical model of a wideband ultrasound generator with an amplifier operating in resonant mode, provide amplitude-frequency characteristics (ADCH) of the wideband ultrasound generator, compare the results of modeling the amplitude-frequency characteristics with experimental measurements.

The following software tools were used in the project: *Arduino IDE*, *LTspice*, *REW*, *MATLAB*, *SigmaStudio*.

The analytical part includes a review of scientific literature on signal generation and processing methods used in standard microprocessors.

In the experimental part, a sinusoidal signal generator was created using a standard microprocessor (ESP32). A model of a wideband ultrasound generator with a Class D amplifier and an *LC* filter at the output was developed for testing purposes.

In the research part, experiments were conducted with a real wideband ultrasound generator that uses 16 coils. The results of the experiments were compared with the modeling results.

Results of the project:

1. The analysis of scientific literature revealed that various algorithms and transformations are used for real-time signal processing and generation in digital microprocessors, such as FFT, DFT, SWFT, DTFT, CZT, DWT, Fourier transformation, and filters like high-pass, low-pass, bandpass, infinite and finite impulse response filters, which facilitate signal processing. Signal generation relies on various libraries and internal microprocessor registers, allowing for high frequencies (90 kHz and above).
2. By implementing and testing the algorithm for a sinusoidal signal generator on a standard processor (ESP32), a signal ranging from 1 Hz to 90 kHz was obtained. Beyond 90 kHz, the sine waveform starts to deform due to the limited performance of the integrated digital-to-analog converter in the controller.
3. By developing a mathematical model of a wideband ultrasound generator with a resonant mode amplifier, and calculating the inductance values (1895 µH, 1585 µH, 1345 µH, 1105 µH, 900 µH, 725 µH, 582 µH, 460 µH, 370 µH, 303 µH, 253 µH, 212 µH, 183 µH, 159 µH, 139 µH, 123 µH)

for the *LC* filter at the output of the Class D amplifier, resonance amplification across the entire specified frequency range (20–85 kHz) was achieved.

4. By measuring the amplitude-frequency characteristics (ADCH) of the wideband ultrasound generator, it was determined that the theoretical measurements allow for resonance amplification across the entire frequency range (20–85 kHz), and the unevenness of the characteristics does not exceed 6 dB (4.18 dB during the first test, 4.1 dB during the second test), which is an acceptable value in audio engineering.
5. Comparing the experimental measurements of the amplitude-frequency characteristics (ADCH) of the wideband ultrasound generator with the modeling results, a strong correlation was found between the tests and theoretical calculations (0.7998 for the first test, 0.7807 for the second test).

Turinys

Paveikslų sąrašas ... **10**
Lentelių sąrašas ... **12**
Santrumpų ir terminų sąrašas ... **13**
Įvadas ... **14**
1. Literatūros analizė ... **15**
1.1. Realus laiko signalų apdorojimas ... 15
1.1.1. Signalų filtrų analizė ... 16
1.1.2. Signalų apdorojimo algoritmų ir technikų analizė ... 17
1.1.3. Begalinio impulso atsako filtro metodas ... 21
1.1.4. Baigtinio impulso atsako filtro metodas ... 22
1.2. Skaitmeninių procesorių analizė ... 23
1.2.1. Skaitmeninių signalų apdorojimo procesorių (DSP) analizė ... 23
1.2.2. Bendro naudojimo mikrovaldiklio ESP32 analizė ... 24
1.2.3. Mikroprocesorių galimybių palyginimas ... 25
1.3. Programinės įrangos apžvalga ... 26
1.3.1. Programavimo įrankiai *Espressif* IDE ir *Eclipse* IDE ... 26
1.3.2. Programavimo kalba *Arduino* IDE ... 28
1.4. Ultragarso keitikliai ... 29
1.4.1. Pjezoelektriniai keitikliai ... 29
1.4.2. Elektrostatinis keitiklis ... 30
1.5. Ultragarso generatoriai ... 31
1.5.1. Ultragarso generatorių tipai ir panaudojimas ... 31
1.5.2. Ultragarso generatoriai medijų sistemose ... 35
2. Eksperimentinė dalis ... **37**
2.1. *LC* žemo dažnio filtro ritės induktyvumo parinkimas ... 38
2.2. ESP32 sinuso formos signalo generatorius ... 42
2.2.1. Sinusinės formos signalo generavimas naudojant matematines formules ... 42
2.2.2. Sinusinės formos signalo generavimas naudojant koordinates ... 43
2.2.3. Sinuso formos signalų generatorius naudojant bibliotekas ... 44
3. Tiriamoji dalis ... **47**
3.1. Plačiajuostis ultragarsinis generatorius ... 47
3.2. Plačiajuosčio ultragarso generatoriaus programa ... 50
3.3. Bandymų įranga ... 54
3.4. Matavimai ... 56
Išvados ... **62**
Literatūros sąrašas ... **63**
Priedai ... **67**
1 priedas. ESP32 sinusinės formos signalo generatoriaus kodas ... 67
2 priedas. Simuliacijos rezultatai ... 79

Paveikslų sąrašas

1 pav. Standartinė skaitmeninių signalų apdorojimo blokinė schema [3] 15
2 pav. Filtrai: a) žemo dažnio; b) aukšto dažnio 16
3 pav. Plačiajuostis filtras 17
4 pav. Dažnių juostos filtras 17
5 pav. Laiko ir dažnio srityse atvaizduotas signalas [5] 18
6 pav. Signalo apdorojimo panaudojus DTFT blokinė schema [8] 19
7 pav. Bandymų rezultatai: a) ir d) pirminis bandymo signalas; b) signalas atkurtas panaudojus faze koduoto spektro metodą; c) faze koduoto metodo atkūrimo paklaida; e) autoriaus [8] siūlomu metodu atkurtas signalas; f) autoriaus [8] siūlomo metodo atkūrimo paklaida 19
8 pav. EQ panaudojant STFT blokinė diagrama [9] 20
9 pav. CZT ir DWT panaudojimo garso signalo vandenženkliui blokinė diagrama [12] 21
10 pav. IIR filtro struktūrinė schema [14] 22
11 pav. FIR filtro struktūrinė schema [5] 22
12 pav. Signalo apdorojimo panaudojus FIR filtrą blokinė diagrama [16] 23
13 pav. Bandymo rezultatai: a) signalas su Gauso triukšmu; b) apdorotas signalas [16] 23
14 pav. ESP32 mikrovaldikliai [19] 25
15 pav. FAUST gramofonas [29] 25
16 pav. *Espressif* IDE programos valdymo langas 27
17 pav. *Eclipse* IDE programos valdymo langas 27
18 pav. *Arduino* IDE programos valdymo langas 28
19 pav. ESP32 UNO su kameros moduliu 29
20 pav. Tiesioginis pjezoelektrinis efektas: a) pjezoelektrikas ramybės būsenoje; b) pjezoelektriką veikia vienos krypties jėga; c) pjezoelektriką veikia kitos krypties jėga [39] 30
21 pav. Atvirkštinis pjezoelektrinis efektas: a) pjezoelektrikas ramybės būsenoje; b, c – atvirkštinis pjezoelektrinis efektas [39] 30
22 pav. Elektrostatinio keitiklio struktūra [42] 30
23 pav. Kraujagyslių valymas ultragarsu [49] 31
24 pav. Ultragarso tyrimų sistemos blokinė diagrama [51] 32
25 pav. Ultragarsinis valymo įrenginys [53] 33
26 pav. Suvirinimo ultragarsu įrenginio blokinė diagrama [54] 34
27 pav. Ultragarsinio pjovimo įrenginys [53] 34
28 pav. Plačiajuosčio ultragarsinio generatoriaus su DSP blokinė schema 37
29 pav. Plačiajuosčio ultragarsinio generatoriaus be DSP blokinė schema 37
30 pav. DSP sinusinių ultragarso signalų generatoriaus blokinė schema 38
31 pav. DSP apėjimo blokinė diagrama 38
32 pav. D klasės stiprintuvo su *LC* filtru modelis 39
33 pav. Sinusinio ir pjūklinio signalo sujungimas 39
34 pav. Sistemos įėjimo ir išėjimo signalai 40
35 pav. Ritės L1 induktyvumas 1895 µH 40
36 pav. Ritės L1 induktyvumas 123 µH 41
37 pav. Stiprinimo charakteristikos esant skirtingiems ričių induktyvumams 41
38 pav. Sinusinės formos signalo generatoriaus kodas 42
39 pav. Sinusinės formos signalai: a) taškai apskaičiuoti programos kodo; b) realus išėjimas 42
40 pav. Sinuso formos signalo generatoriaus kodas naudojant koordinates 43

41 pav. Sinusoidinės formos signalai: a) iš masyvo taškų; b) realus išėjimas 43
42 pav. ESP32 skaitmeninio-analoginio keitiklio sinuso formos signalo generatoriaus principinė schema [43] 44
43 pav. Bandomoji principinė schema (a) ir reali bandomoji sistema (b) 45
44 pav. ESP32 sinuso formos signalo generatoriaus išėjimas nustačius 20 kHz dažnį: a) nustatytas dažnis; b) sinuso formos signalas 45
45 pav. ESP32 sinuso formos signalo generatoriaus išėjimas nustačius 90 kHz dažnį: a) nustatytas dažnis; b) sinuso formos signalas 45
46 pav. ESP32 sinuso formos signalo generatoriaus išėjimas nustačius 102 kHz dažnį: a) nustatytas dažnis; b) sinuso formos signalas 46
47 pav. Plačiajuostis ultragarso generatorius 47
48 pav. ESP32-WROOM-32E mikrovaldiklio blokinė diagrama [43] 47
49 pav. ADAU1462 skaitmeninio garso procesoriaus blokinė diagrama [44] 48
50 pav. AD1938 blokinė diagrama [45] 48
51 pav. IR4301M D klasės stiprintuvo blokinė diagrama [46] 49
52 pav. EVAL-ADUSB2Z programatorius 51
53 pav. ESP32 ir ADAU1462 ryšio sudarymas 51
54 pav. *SigmaStudio* programos kodo eksportavimas 51
55 pav. *SigmaStudio* programos aplinkoje eksportuoti failai 52
56 pav. *Arduino* IDE programos kodas, aprašantis naudojamas bibliotekas 52
57 pav. *Arduino* IDE bibliotekos užkomentavimas ir kitos įtraukimas 53
58 pav. *Arduino* IDE kodo dalis DSP veikimui užtikrinti 53
59 pav. *Arduino* IDE kodo dalis DSP veikimui užtikrinti. Tęsinys 53
60 pav. ESP-prog programatorius 54
61 pav. Keitiklio veikimo principas [47] 54
62 pav. Elektrostatinis keitiklis 54
63 pav. „miniDSP“ UMIK-1 mikrofonas 55
64 pav. *REW* programos matavimų langas 55
65 pav. Plačiajuosčio ultragarsinio generatoriaus ADCH – išėjimo įtampos matavimų rezultatai 57
66 pav. Plačiajuosčio ultragarsinio generatoriaus ADCH – perskaičiuotos SPL reikšmės 59
67 pav. Simuliacijos ir pirmo bandymo palyginimas 60

Lentelių sąrašas

1 lentelė. Ričių induktyvumai 20–45 kHz dažnių diapazone 41
2 lentelė. Ričių induktyvumai 45–85 kHz dažnių diapazone 41
3 lentelė. Pirmo bandymo rezultatai 56
4 lentelė. Antrojo bandymo rezultatai 56
5 lentelė. Pirmo bandymo perskaičiuotos reikšmės 58
6 lentelė. Antro bandymo perskaičiuotos reikšmės 58
7 lentelė. Koreliacijos koeficiento reikšmių lentelė [48] 61

Santrumpų ir terminų sąrašas

Santrumpos:

DSP – skaitmeninių signalų apdorojimas (angl. *digital signal processing*).

Terminai:

Signalas – fizikinis dydis, kuris kinta laike, erdvėje arba priklauso nuo bet kokio nepriklausomo kintamojo arba kintamųjų [1].

Skaitmeninis signalas – skaičių (imčių) seka, kurioje kiekvienas skaičius yra atvaizduojamas baigtiniu skaitmenų skaičiumi (baigtinis tikslumas) [1].

Interpoliacija – prognozavimo metodas, kai kintamojo rodiklio reikšmė apytiksliai nustatoma remiantis žinomomis reikšmėmis. Ja siekiama daugelio statistinių ar loginių duomenų pagrindu rasti ir paaiškinti nežinomas tarpines funkcijos reikšmes ar reiškinio charakteristikas [4].

Įvadas

Skaitmeninių signalų apdorojimo procesoriai turi svarbų vaidmenį įvairiose technologijų ir mokslo srityse, dėl savo greitaveikos ir mažos kainos. Dažniausiai jie naudojami realiuoju laiku skaitmenizuoti ir matematiškai apdoroti garso ir vaizdo signalus. Dėl savo architektūros skaitmeniniai signalų apdorojimo procesoriai puikiai tinka signalų apdorojimo operacijoms, tačiau jie nėra tokie universalūs kaip standartiniai mikrovaldikliai [6]. Šiame darbe bus aptarti skaitmeninių signalų apdorojimo būdai ir skaitmeniniuose signalų apdorojimo procesoriuose naudojama metodika, kurią bus bandoma pritaikyti standartiniame mikrovaldiklyje analizuojant plačiajuosčio ultragarsinio generatoriaus realizavimo galimybes.

Tyrimų objektas – plačiajuostis ultragarsinis generatorius ir signalų generavimo bei apdorojimo standartiniuose mikroprocesoriuose metodai.

Projekto tikslas – išanalizuoti plačiajuosčio ultragarsinio generatoriaus realizavimo galimybes, naudojant signalų generavimo ir apdorojimo standartiniuose mikroprocesoriuose metodus.

Projekto uždaviniai:

1. Išnagrinėti esamus realaus laiko signalų generavimo ir apdorojimo metodus, naudojamus standartiniuose mikroprocesoriuose.
2. Aprašyti sinusinės formos signalų formavimo standartiniame procesoriuje algoritmą.
3. Pateikti elektrostatinio plačiajuosčio ultragarsinio generatoriaus su stiprintuvu, veikiančiu rezonansiniame režime, matematinį modelį.
4. Pateikti plačiajuosčio ultragarsinio generatoriaus amplitudines dažnines charakteristikas (ADCH).
5. Palyginti plačiajuosčio ultragarsinio generatoriaus amplitudinės dažninės charakteristikos modeliavimo ir eksperimentinių matavimų rezultatus.

Tyrimų metodai: mokslinės literatūros analizė, eksperimentai su ESP32 mikrovaldikliu.

Autoriaus publikuotų straipsnių bibliografinis sąrašas:

1. Živelis, Vaidas; Vanagas, Gailius. Signalų apdorojimo realiame laike, naudojant standartinius mikroprocesorius, tyrimas // Technologijų ir verslo aktualijos – 2022: studentų mokslinių darbų konferencijos pranešimų medžiaga, Lietuva, Panevėžys, 2022 lapkričio 25 d. / Kauno technologijos universiteto Panevėžio technologijų ir verslo fakultetas. Kaunas: Kauno technologijos universitetas.

Autoriaus konferencijose skaityti pranešimai:

1. Pranešimas tema „Signalų apdorojimo realiame laike, naudojant standartinius mikroprocesorius, tyrimas“ 22-oje studentų mokslinių darbų konferencijoje „Technologijų ir verslo aktualijos – 2022“. Panevėžys: Kauno technologijos universitetas, 2022 lapkričio 25 d.

1. Literatūros analizė

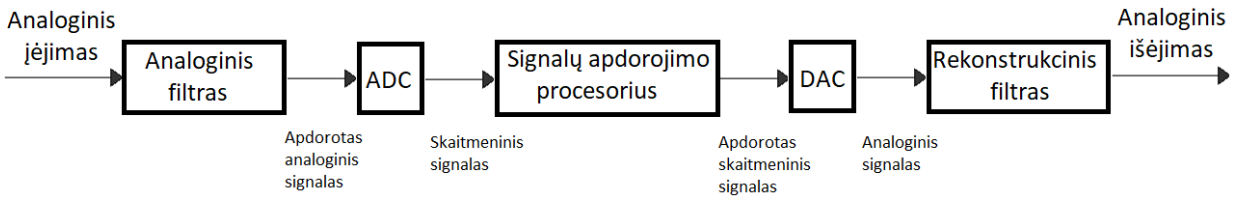
Žmogaus ausis girdi platų garsų diapazoną, kurį garso atkūrimo sistemos ne visada sugeba identifikuoti, todėl taikant standartines DSP funkcijas ir algoritmus galima sukurti įvairius specialius efektus, pvz., dinaminius bosus (nors maži garsiakalbiai ir neatkuria žemų dažnių, tačiau sugeneravus harmonikas į aukštesnius dažnius galima sukurti žemų dažnių atkūrimo iliuziją), garsumo efektą (stiprinamos atskiros harmonikos), dinaminį diapazoną (suspaudus dinaminį diapazoną, mažos amplitudės signalai stiprinami daugiau nei didelės amplitudės signalai, todėl nukertamas tam tikras signalo ruožas) ir pan. [2].

Signalams apdoroti galima naudoti įvairius filtrus sudarytus iš elektronikos elementų (aukšto dažnio, žemo dažnio filtrai ir pan.), bei metodus, kuriems reikalingi mikrovaldikliai su įrašytomis signalo apdorojimo instrukcijomis [3].

1.1. Realus laiko signalų apdorojimas

Signalų apdorojimas naudojamas norint išryškinti reikalingą informaciją iš gaunamo signalo, pašalinant iš jo nereikalingą informaciją ir trikdžius. Standartinė skaitmeninių signalų apdorojimo blokinė schema pateikta 1 paveiksle (žr. 1 pav.). Ją sudaro:

- analoginio signalo filtras;
- analoginis į skaitmeninį signalą keitiklis (angl. *analog-to-digital converter*, trump. ADC);
- skaitmeninio signalo apdorojimo procesoriaus;
- skaitmeninis į analoginį signalą keitiklis (angl. *digital-to-analog converter*, trump. DAC);
- rekonstrukcinis filtras.



1 pav. Standartinė skaitmeninių signalų apdorojimo blokinė schema [3]

Analoginis signalas, gaunamas iš imtuvo, perduodamas įtampos ar srovės stiprio forma į analoginio signalo filtro įėjimą, kuriame prieš signalo nuskaitymą apribojamas jo dažnio spektras. Gautas signalas iš analoginio paverčiamas į skaitmeninį (skaičių seką, turinčią baigtinį tikslumą) panaudojus ADC. Skaitmeninio signalo apdorojimo procesorius gautą informaciją apdoroja instrukcijomis įrašytomis į procesorių. Skaitmeninis signalų procesorius gali būti kompiuteris arba mikroprocesorius, kuris yra programuojamas, kad su įėjimo signalu atliktų pageidaujamas operacijas. Jis taip pat gali būti neprogramuojamas skaitmeninis procesorius, kuris yra sukonstruotas atlikti tam tikrą veiksmų seką su įėjimo signalu. Apdorojus informaciją procesoriuje įrašytomis instrukcijomis, suformuojamas išeinantis skaitmeninis signalas, kuris DAC keitikliu paverčiamas analoginiu signalu („sujungiami taškai“ skaitmeniniame signale atliekant kokią nors interpoliaciją). Šis signalas apdorojamas rekonstrukciniu filtru, kuris sureguliuoja signalo įtampos lygius, kad išeinantis analoginis signalas galėtų būti panaudotas [3].

1.1.1. Signalų filtrų analizė

Daugumos signalų filtrų paskirtis – pašalinti signalo dalį, kuri laikoma triukšmu. Paprasčiausias filtras – *RC* – tai grandinė sudaryta iš vienos varžos (*R*) ir kondensatoriaus (*C*). *RC* grandinę, atsižvelgiant į elementų išdėstymą, galima naudoti kaip žemo dažnio filtrą (angl. *low pass filter*) (žr. 2 pav., a) arba aukšto dažnio filtrą (angl. *high pass filter*) (žr. 2 pav., b) [4, 26].

a) b)

2 pav. Filtrai: a) žemo dažnio; b) aukšto dažnio

RC filtrų veikimas pagrįstas kondensatoriaus priklausomybe nuo veikimo dažnio:

$$X_C = \frac{1}{\omega C} = \frac{1}{2\pi f C}; \quad (1.1)$$

čia X_C – kondensatoriaus reaktyvinė varža, Ω;
ω – kampinis pagreitis, rad/s;
C – kondensatoriaus talpa, F;
f – dažnis, Hz.

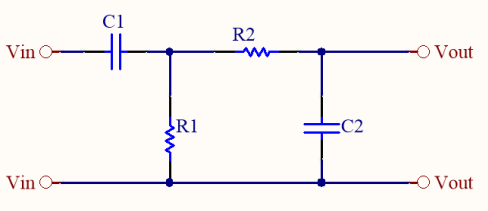
Iš (1) lygties matyti, kad didėjant dažniui, kondensatoriaus reaktyvinė varža mažėja, o esant mažam dažniui, – didėja, tokiu būdu apribojamas išeinančio signalo dažnis. Esant žemo dažnio filtrui esant didesniam signalo dažniui kondensatoriaus reaktyvinė varža bus mažesnė negu varžos *R* ir signalo srovė kondensatoriaus bus šuntuojama. Esant aukšto dažnio filtro atvejui varža *R* ir kondensatorius *C* yra sukeisti vietomis, todėl filtras praleis tik aukštesnio dažnio signalus [4, 26].

Žemo dažnio filtro panaudojimo pavyzdį pateikia Li [24], kuris savo darbe žemo dažnio *RC* filtru sumažina ADC išėjimo svyravimus, dėl to operaciniai stiprintuvai naudojami tokioje sistemoje naudoja mažiau energijos [24].

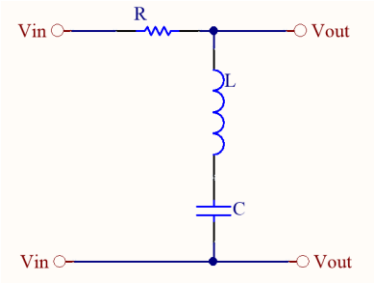
Aukšto dažnio filtro panaudojimo pavyzdį pateikia Seong-Ho [25], kuris savo darbe aukšto dažnio filtru išfiltravo 120 Hz šviesos triukšmą, skleidžiamą aplinkinių lempų, tam, kad galėtų panaudoti matomą šviesą signalui perduoti [25].

Žemo dažnio ir aukšto dažnio filtrais signalą galima apriboti tik vienoje dažnio spektro zonoje. Norint vienu metu apriboti aukšto ir žemo dažnio signalų sritis, naudojami plačiajuosčiai filtrai (angl. *bandpass filter*) (žr. 3 pav.). Paprasčiausias plačiajuostis filtras gaunamas sujungus aukštų dažnių ir žemų dažnių filtrus. Tokiu būdu susidaro filtras iš dviejų varžų *R*1, *R*2 ir dviejų kondensatorių *C*1, *C*2 [5, 26]. Plačiajuosčio filtro panaudojimo pavyzdį pateikia BELOV [27], kuris savo darbe plačiajuostį filtrą, kartu su kitais filtrais ir metodais, panaudojo širdies ir kraujagyslių tyrimams. Kadangi tiriant žmogaus organizmą reikalingas didelis tikslumas ir duomenų perdavimas esamu laiku, nuspręsta panaudoti plačiajuostį filtrą sumažinant duomenų perdavimo trukmę. Toks analoginis elementas sumažina visos sistemos energijos vartojimą [27].

Dar vienas plačiai naudojamas filtras yra dažnių juostos filtras (angl. *band-stop filter*) (žr. 4 pav.), jo veikimas yra atvirkštinis plačiajuosčiam filtrui. Šis filtras apriboja tam tikrą siaurą signalų dažnio sritį esančią tarp žemo ir aukšto dažnio sričių [5, 26].



3 pav. Plačiajuostis filtras



4 pav. Dažnių juostos filtras

Dažnių juostos filtro panaudojimo pavyzdį pateikia Gadelovits [28], kuris savo darbe dažnių juostos filtrą, kartu su kitais filtrais ir metodais, panaudojo dažnio keitiklio įtampos kokybei pagerinti. Panaudojus dažnių juostos filtrą, sumažinamas netiesinių apkrovų poveikis ir padidinamas atsparumas dažnių svyravimams [28].

Jei atskirti sau reikiamo dažnio signalo rėžio nepavyksta su vienu filtru, galima naudoti ir pirmiau išvardytų filtrų kombinacijas.

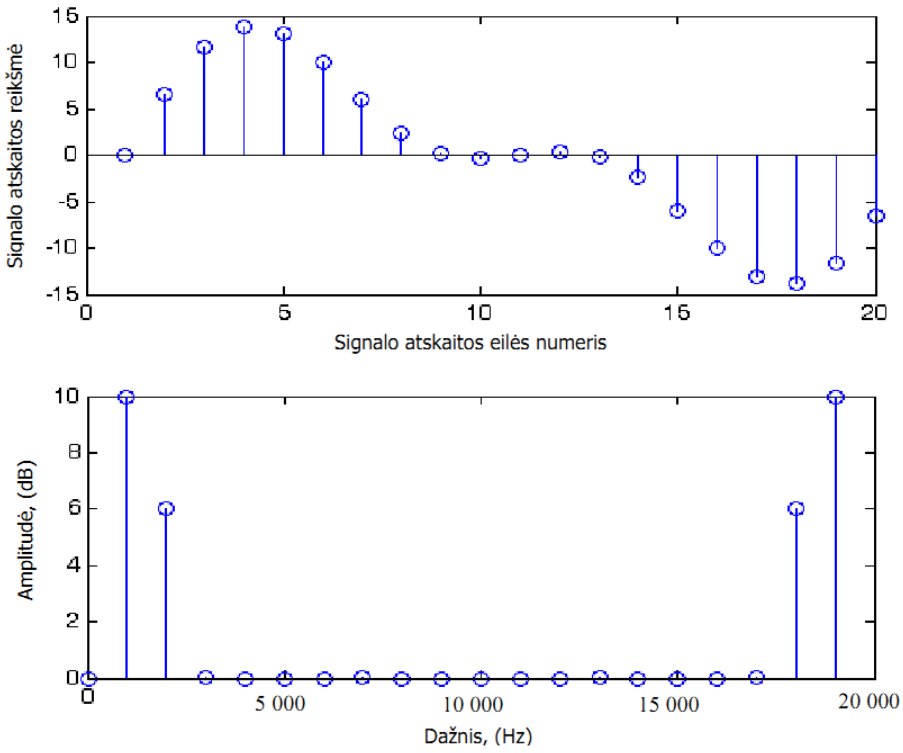
1.1.2. Signalų apdorojimo algoritmų ir technikų analizė

Viena iš pagrindinių skaitmeninio signalo apdorojimo priemonių yra dažninė analizė, kuri iš esmės apima *Furjė* eilutes periodiniams signalams ir *Furjė* transformaciją aperiodiniams signalams. Dažninė signalo analizė atliekama ir tolydiniu, ir diskretiniu laiku. Kadangi gaunami signalai dažniausiai būna laiko srityje, juos analizuojant patogu apdorotą signalą perkelti į dažnio sritį (žr. 5 pav.) atlikus konversiją, panaudojus klasikinį signalų apdorojimo metodą – diskrečioji *Furjė* transformacija (angl. *discrete Fourier transform*, trump. DFT) [5].

Turint nuo laiko priklausantį signalą $x(t)$ (sumą įvairių dažnių ir fazių harmoninių signalų), reikia jį skaitmenizuoti. Skaitmenizavus signalą, vietoje tolydžios funkcijos naudojamas atskaitų masyvą. Diskretizavus signalą $x(t)$, gaunamas diskretizuotas signalas x_n su kuriuo galima atlikti diskrečiąją Furjė transformaciją:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi}{N}kn},\ k = 0,\ 1,\ \ldots,\ N\text{-}1; \qquad (1.2)$$

čia X_k – signalo spektro masyvas (signalas dažnių srityje);
x_n – diskretizuotas signalas (laikinis signalas);
j – menamasis vienetas;
k – diskretaus spektro atskaitos numeris;
n – skaičiaus numeris eilutėje;
N – x_n ir X_k masyvų ilgis.



5 pav. Laiko ir dažnio srityse atvaizduotas signalas [5]

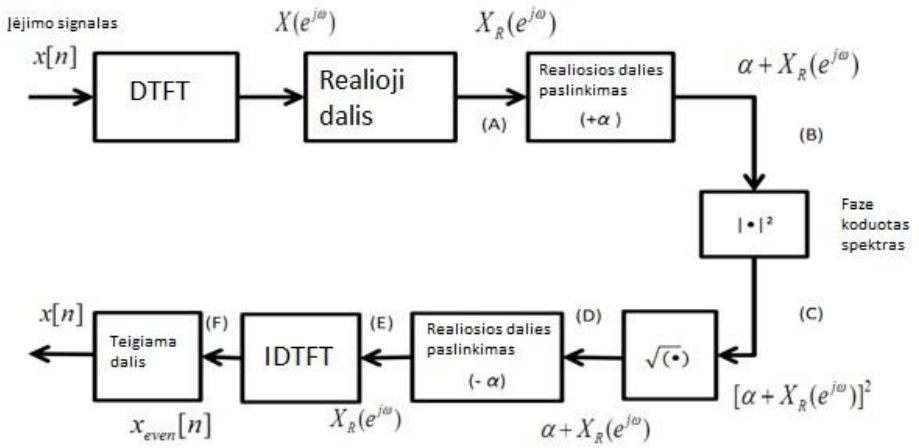
Žinant signalo spektrą, patį signalą galima apskaičiuoti panaudojus atvirkštinę Furjė transformaciją:

$$x_n = \sum_{n=0}^{N-1} X_k e^{-j\frac{2\pi}{N}kn},\ k = 0, 1, ..., N\text{-}1. \qquad (1.3)$$

Nors DFT ir sukurta specialiai skaičiavimams, tačiau tokia forma kaip (1.2) ir (1.3) formulėse retai kada naudojama, nes tokia matematinė forma gana imli skaičiavimams ir naudoja daug resursų, tokių kaip duomenų atmintis, vidiniai ir darbiniai registrai ir pan., dėl to realiuoju laiku gaunamas signalas negali būti apdorotas DFT [5]. Tačiau sugalvota nemažai kitų signalų apdorojimo metodų kurių pagrindas yra DFT.

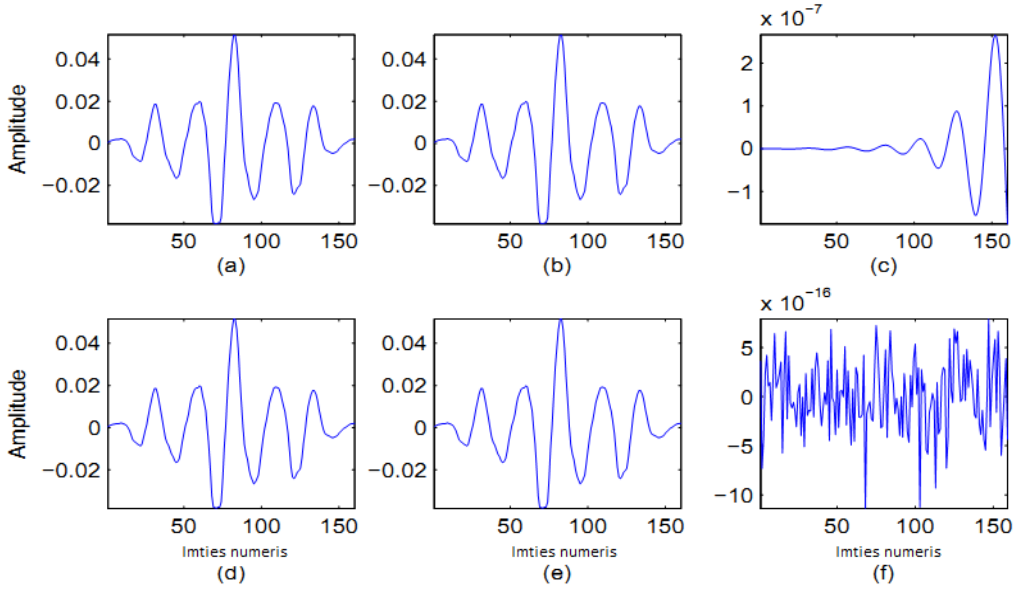
Realiu laiku apdoroti diskretinius signalus dažnai naudojama greitoji *Furjė* transformacija (angl. *fast Fourier transform*, trump. FFT). FFT reikalauja mažiau skaičiavimo operacijų, bet pateikia identišką DFT rezultatą. FFT panaudojimą garso signalams apdoroti aptaria [9], [10] ir [11] autoriai. Jų bandymų metu nustatyta jog FFT pagreitina signalų apdorojimą ir gerai atkuria garso failų dažnių spektrą. Greitoji *Furjė* transformacija yra DTFT algoritmas kuris sumažina reikalingu atlikti skaičiavimų skaičių, norint apskaičiuoti N taškų, nuo N^2 iki $N \log_2 N$ [5].

Dar vieną signalų apdorojimo metodą, diskrečiąją-laiko *Furjė* transformaciją (angl. *discrete-time Fourier transform*, trump. DTFT), savo darbe aptaria Proakis [8], kuris panaudoja DTFT garso signalui iš galios spektro atkurti. Straipsnyje siūloma signalo apdorojimo blokinė diagrama pateikta 6 paveiksle.



6 pav. Signalo apdorojimo panaudojus DTFT blokinė schema [8]

Panaudojus PROAKIS [8] metodą, signalo fazę galima užkoduoti ir atkoduoti dažnių srityje. Pasinaudojus šiuo metodu, signalą galima atkurti tiek iš menamosios tiek iš realiosios DTFT dalies. PROAKIS [8] pasiūlyto signalų apdorojimo metodo bandymo rezultatai pateikti 7 paveiksle.

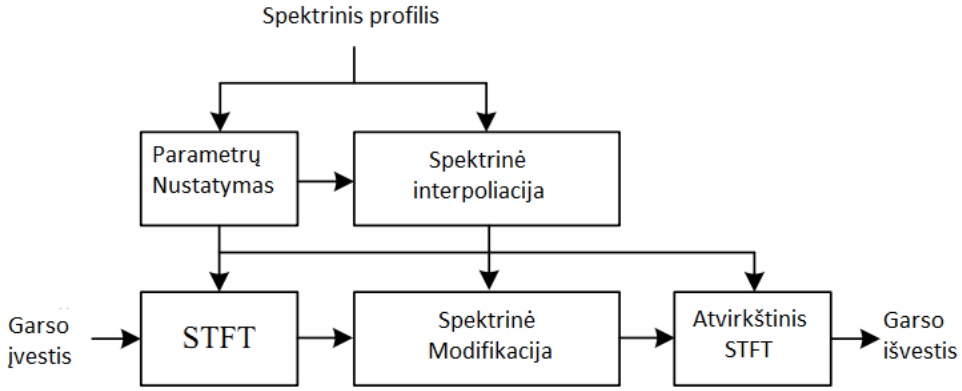


7 pav. Bandymų rezultatai: a) ir d) pirminis bandymo signalas; b) signalas atkurtas panaudojus faze koduoto spektro metodą; c) faze koduoto metodo atkūrimo paklaida; e) autoriaus [8] siūlomu metodu atkurtas signalas; f) autoriaus [8] siūlomo metodo atkūrimo paklaida

Iš 7 paveikslo matyti, kad pasinaudojus autoriaus [8] siūlomu metodu, signalo atkūrimo paklaida yra 10^9 kartų mažesnė lyginant su faze koduoto spektro metodu.

DTFT dažnai naudojama tolydžiųjų funkcijų imtims analizuoti. Iš tolygiai išdėstytų imčių ji sukuria dažnio funkciją, kuri yra lygi pradinės tolydžios funkcijos periodiškai sumuotoms *Furjė* transformacijoms. Atvirkštinė DTFT yra pradinių imčių reikšmių seka [5].

Dar vienas *Furjė* transformaciją naudojantis metodas – trumpo laiko *Furjė* transformacija (angl. s*hort-time Fourier transform*, trump. STFT). STFT metodas naudojamas nustatant signalo dalių fazių turinį ir sinusoidinį dažnį signalui besikeičiant laike. Atliekant signalo apdorojimą STFT metodu, signalas padalinamas į mažesnes vienodo ilgio dalis, kurioms atliekama *Furjė* transformacija [7]. Šį metodą garso apdorojime naudoja Zhang [9]. Savo darbe STFT jis panaudoja parametrinio garso tembro lygintuvui (angl. *equalizer*, trump. EQ) kurti. Jo siūlomo metodo blokinė diagrama pateikta 8 paveiksle.



8 pav. EQ panaudojant STFT blokinė diagrama [9]

Siūlomame modelyje naudojant STFT, galima analizuoti pereinamųjų signalų spektrą. Išlyginimas yra atliekamas dažnių srityje, taip gaunamas lankstus EQ parametrų valdymas [9].

Norint perkelti apdorojamus signalus į *z* sritį, naudojama *z*-transformacija (angl. *z-transform*), kuri atlieka tą patį vaidmenį diskretinio laiko signalų ir sistemų analizėje, kaip ir *Laplaso* transformacija (angl. *Laplace transform*, trump. LT) tolydinio laiko signalų ir sistemų analizėje.

Bendru atveju, *z*-transformacija yra laipsninė eilutė, kur *z* yra kompleksinis kintamasis. Diskretinio laiko sekos x_n *z*-transformacija yra pateikta (1.4) formulėje [7]:

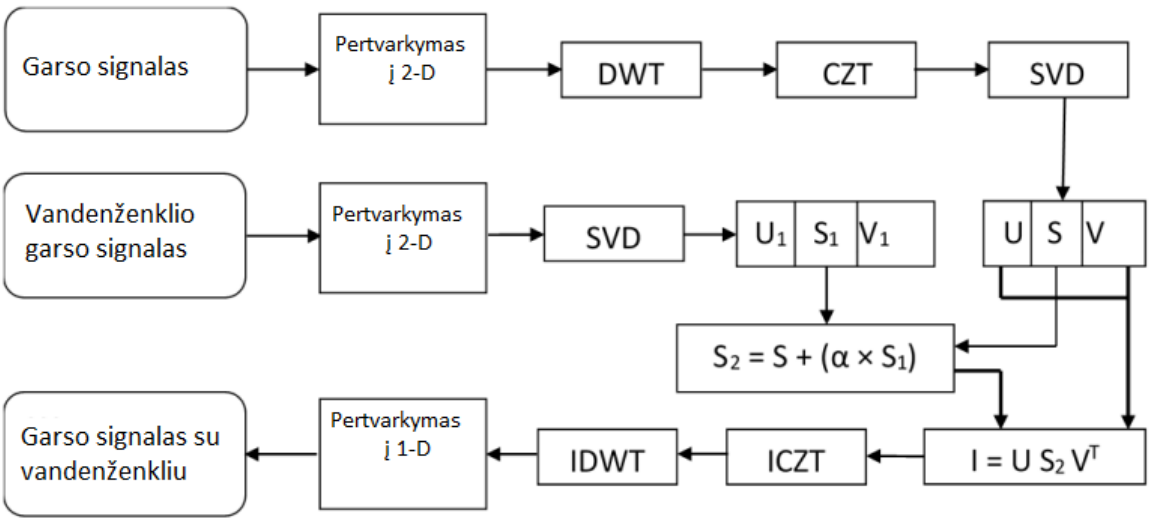
$$X_z = \sum_{n=-\infty}^{\infty} x_n z^{-n} \; ; \qquad (1.4)$$

čia z – kompleksinis kintamasis;
X_z – transformuota funkcija;
x_n – diskretinio laiko seka.

Pasinaudojus *z*-transformacija ir FFT galima atlikti CZT (angl. *chirp-transform*) algoritmą, kurio metu apdorojamo signalo duomenys išdėstomi apskritimu *z* srityje. Apdoroto signalo dažnio sritis yra lanko formos kurią galima padalyti į n dalių, todėl panaudojus CZT *z*-transformacija gali būti efektyviai įvertinta įvairiuose *z* plokštumos taškuose [12].

CZT algoritmo ir diskretinės bangų transformacijos (angl. *discrete wavelet transform*, trump. DWT) panaudojimas garso signalams ženklinti vandenženkliu aptariamas [12] autoriaus, jo siūlomo metodo blokinė diagrama pateikta 9 paveiksle.

Šioje diagramoje SVD (angl. *singular value decomposition)* kvadratinė matricos transformacija į tris atskiras matricas *U, V, S*. Čia *U* ir *V* – kvadratinės matricos, o *S* – įstrižinė matrica, kurios įstrižainės elementai yra teigiami, o α – didinimo faktorius, kuris aprašo vandenženklio stiprumą atkurtame garso signale.



9 pav. CZT ir DWT panaudojimo garso signalo vandenženkliui blokinė diagrama [12]

Pasinaudojus autoriaus [12] metodu, į pasirinktą garso įrašą buvo įterptas norimas vandenženklis.

1.1.3. Begalinio impulso atsako filtro metodas

Begalinio impulso atsako filtras (angl. *infinite impulse response*, trump. IIR) yra skaitmeninių signalų apdorojimo metodas, kurio išėjimo reikšmė priklauso nuo prieš tai buvusios reikšmės. IIR filtro matematinė išraiška pateikta (1.5) formulėje [7, 13]:

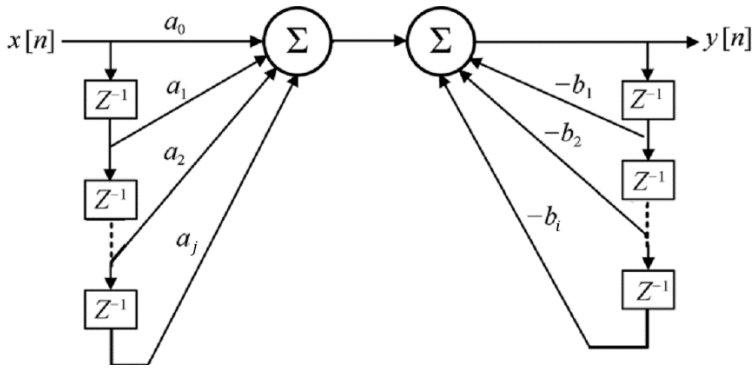
$$\sum_{i=0}^{K} b_i y[n-i] = \sum_{j=0}^{L} a_j x[n-j]; \quad (1.5)$$

čia $y(n)$ – išėjimo seka;
$x(n)$ – įėjimo seka;
n – iteracijos numeris;
K ir L – filtro eilės numeris;
i ir j – koeficiento numeris;
a_j ir b_i – koeficientai, lemiantys atsako pobūdį.

Pasinaudojus IIR tipo filtru, galima neapkraunant įrangos apdoroti duomenis 1000Hz greičiu, tačiau šie filtrai turi ir nemažai trūkumų, tokių kaip prastas efektyvumas ir kvantavimo paklaidos. Norint sumažinti šio tipo filtro trūkumus naudojami įvairūs algoritmai, tokie kaip: dirbtinių bičių kolonijos (angl. *artificial bee colony*, ABC), dalelių spiečiaus optimizavimas (angl. *particle swarm optimization*, PSO) ir kt. [7, 13].

Agrawal [13] savo darbe aptaria IIR filtro optimizavimą pasinaudojus hibridinę techniką, naudojant ABC ir PSO algoritmų dalis, taip gaunamas IIR filtras su mažu kvantavimo efektu ($10e^{-9}$) [13].

IIR filtro struktūrinė schema pateikta 10 paveiksle.



10 pav. IIR filtro struktūrinė schema [14]

Skaitmeninis IIR filtras yra analoginio *RLC* ar aktyvaus *RC* filtro analogas, todėl jį galima apsirašyti apskaičiavus analoginio filtro prototipą ir pavertus jį į ekvivalentišką skaitmeninį filtrą [14].

1.1.4. Baigtinio impulso atsako filtro metodas

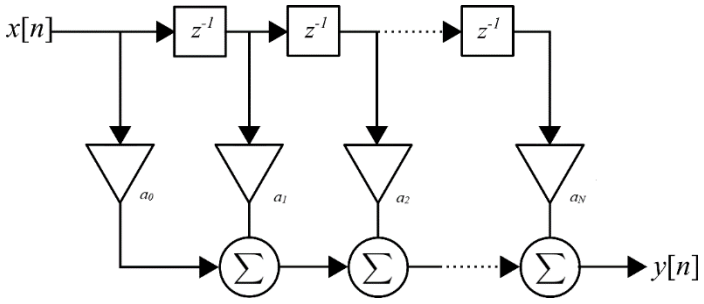
Baigtinio impulso atsako (angl. *finite impulse response*, trump. FIR) filtras yra dalinis IIR filtro atvejis, kai skaičiavime naudojamos tik buvusios įėjimo atskaitos. FIR filtro matematinė išraiška pateikta (1.6) formulėje [15]:

$$y(n) = \sum_{i=0}^{N} a_i x(n-i); \quad (1.6)$$

čia *y(n)* – išfiltruotas signalas;
x(n) – įėjimo signalas;
N – filtro eilė;
a_i – koeficientai;
n – takto numeris.

FIR filtrai dėl savo stabilumo plačiai naudojami skaitmeninių signalų apdorojime. Atsižvelgiant į koeficiento a_i vertę, FIR filtras gali būti naudojamas kaip aukšto, žemo dažnio, dažnio juostos ar plačiajuostis filtras [15, 16].

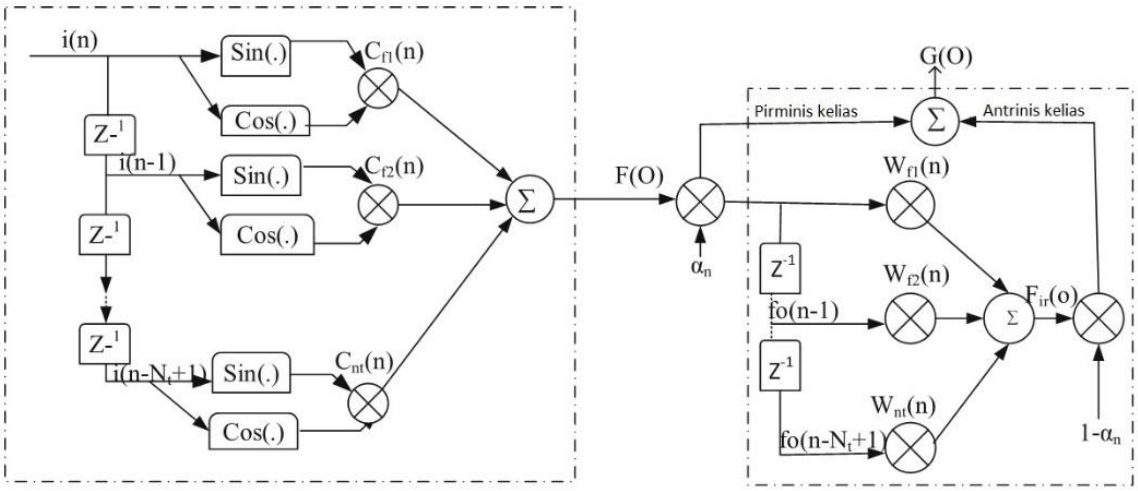
FIR filtro struktūrinė schema pateikta 11 paveiksle.



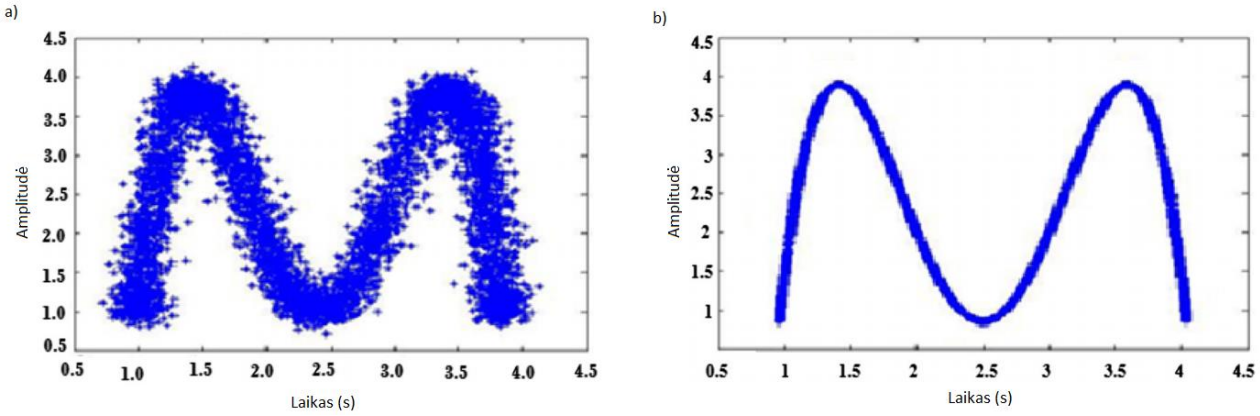
11 pav. FIR filtro struktūrinė schema [5]

Savo darbe, Alia [16], aptaria FIR filtro panaudojimo galimybes triukšmo iš signalo pašalinime. Jo siūloma signalo apdorojimo blokinė diagrama pateikta 12 paveiksle.

Atlikus signalo apdorojimą ALIA [16] siūlomu metodu, bandymo metu iš signalo užteršto *Gauso* triukšmu buvo atstatytas neužterštas signalas. Bandymo rezultatai pateikti 13 paveiksle.



12 pav. Signalo apdorojimo panaudojus FIR filtrą blokinė diagrama [16]



13 pav. Bandymo rezultatai: a) signalas su Gauso triukšmu; b) apdorotas signalas [16]

Identiški rezultatai buvo pasiekti filtruojant signalą užterštą chaotišku triukšmu.

1.2. Skaitmeninių procesorių analizė

Skaitmeniniai procesoriai – tai įrenginiai skirti atlikti įvairiems skaičiavimams, ir skaitmeninių signalų manipuliacijoms. Skaitmeniniais signalais gali būti paversti įvairūs analoginiai signalai, tokie kaip garsas, vaizdas ir pan. Skaitmeniniai procesoriai gali būti skirstomi į specializuotus ir bendro naudojimo procesorius. Bendro naudojimo procesoriai diegiami į mikrovaldiklius, kuriems nėra svarbi duomenų analizė realiu laiku. Specializuoti procesoriai gerai pritaikyti atlikti skaičiavimus ir yra labai greiti, tačiau jų panaudojimui reikalinga daug išorinių elementų ir jie yra brangesni. Tokie procesoriai vadinami DSP procesoriais. Yra nemažai mikroprocesorių gamintojų sukūrusių daugybę įvairių mikroprocesorių modelių [17].

1.2.1. Skaitmeninių signalų apdorojimo procesorių (DSP) analizė

Sparčiai tobulėjant procesorių architektūrai, didžioji dalis skirtumų tarp DSP ir standartinių procesorių išnyko, tačiau yra likę keletas bazinių šių procesorių skirtumų.

– Taikant įterptąsias programas, DSP naudoja mažiau energijos ir geriau apdoroja realaus laiko signalus. DSP naudoja daugybę struktūrų, kad sumažintų galią ir pagerintų apdorojimo greitį, tokių kaip statinį lygiagretumo tikrinimą, netiesioginį pertraukimą, SPM (angl. *scrachpad*

memory), programinės įrangos talpyklos suderinamumą, tiesiogines didelės spartos IO (angl. *input-output)* sąsajas ir kt.
- DSP vartojamas žodis paprastai yra mažesnis arba lygus 32 bitams, o 32 bitų ir 64 bitų standartiniai procesoriai yra vienodai dažnai naudojami. Nors kai kurie DSP, pvz., TI TMS320C6678 gali palaikyti 64 bitų skaičiavimą netiesiogiai, kuris vadinamas netikru 64 bitų skaičiavimu, tačiau jo duomenų perdavimas ir bendrieji registrai yra 32 bitų.
- DSP dažniausiai naudoja instrukcijas orientuotas į signalų apdorojimą, tokias kaip daugyba, kompleksinė daugyba arba *Galois* lauko daugybos instrukcijas. Standartiniai procesoriai tokių operacijų paprastai nepalaiko.
- DSP paprastai turi prievadą, per kurį, pvz., kompiuteris, gali visiškai valdyti DSP būseną, įskaitant vidinės atminties ar registro išteklius, o standartiniai procesoriai paprastai neturi tokio prievado [18, 31].

Skaitmeninių signalų apdorojimo procesoriai turi panašų integracijos lygį ir tokius pačius taktinius dažnius kaip bendrosios paskirties mikroprocesoriai, tačiau jie paprastai pasižymi geresniu našumu, mažesne delsa ir nereikalauja specializuoto aušinimo ar didelių baterijų. Tai leidžia jiems būti pigesnė alternatyva bendrosios paskirties mikroprocesoriams nešiojamuose įrenginiuose [18].

Skaitmeninių signalų apdorojimo procesorius galima panaudoti labai įvairiai, kaip keletą iš panaudojimo galimybių galima pateikti Ahmadi [32] ir Klilou [33] darbus. Ahmadi [32] panaudojo DSP elektrofiziologijos eksperimentams atlikti realiu laiku. Klilou [33] panaudojo daugiabranduolinį skaitmeninių signalų apdorojimo procesorių C6678, apdoroti signalams gautiems iš objektų aptikimo radarų. Panaudojus kelis mikroprocesoriaus branduolius, pasiektas 14 proc. didesnis našumas, nei buvęs ankstesniuose kitų mokslininkų darbuose [33].

1.2.2. Bendro naudojimo mikrovaldiklio ESP32 analizė

Dauguma bendrosios paskirties mikroprocesorių ir operacinių sistemų gali sėkmingai vykdyti DSP algoritmus. Tačiau jie netinka naudoti nešiojamuose įrenginiuose, pvz., mobiliuosiuose telefonuose. Šiame projekte bus naudojamas ESP32 mikrovaldiklis su standartiniu procesoriumi. ESP32 mikrovaldiklių pavyzdžiai pateikti 14 paveiksle.

ESP32 mikrovaldiklis turi integruotą *Wi-Fi* 802, 11 b/g/n, *Bluetooth* 4.2 ir daugybe išorinių įrenginių. Šio mikrovaldiklio procesoriaus taktinis dažnis – 240 MHz. ESP32 mikrovaldiklis turi 36 bendrosios paskirties įvesties ar išvesties kontaktus, bei 16 PWM (angl. *pulse width modulation)* kanalų. Šis mikrovaldiklis tai pat turi 4MB išliekamąją atmintį. ESP32 mikrovaldiklis turi du „Xtensa“ LX6 branduolius kuriuos galima valdyti atskirai, bei 520 KB statinės operatyviosios atminties (angl. *static random access memory*, trump. SRAM) informacijai ir instrukcijoms.

Atsižvelgiant į mikrovaldiklio konfigūraciją, galima naudoti tokius ryšio standartus, kaip: SPI, I2S, I2C, CAN, UART, *Ethernet* MAC ir IR [19, 20].

Kaip vieną iš ESP32 mikrovaldiklių pritaikymo galimybių aptaria Michon [29], kuris panaudodamas FAUST architektūrą skirtą ESP32 mikrovaldikliams, kuri leidžia generuoti skaitmeninių signalų apdorojimo variklius ESP32 mikrovaldiklių šeimoje. Michon [29] savo darbe aptaria ne tik ESP32 galimybes, bet ir panaudoja ESP32 mikrovaldiklį gramofonui sukruti. Šio gramofono modelis pateiktas 15 paveiksle [29].

14 pav. ESP32 mikrovaldikliai [19]

15 pav. FAUST gramofonas [29]

FAUST gramofonas yra programuojamas muzikos instrumentas, sukurtas kaip „Amstramgrame“ projekto dalis, kuris šiuo metu yra bandomojoje stadijoje. „Amstramgrame“ siekia išmokyti fizikos ir matematikos, atliekant programavimo pratimus su gramofonu.

1.2.3. Mikroprocesorių galimybių palyginimas

Nors specializuoti ir bendros paskirties mikroprocesoriai turi pliusų ir minusų, yra išskiriami tokie šių mikroprocesorių skirtumai.

DSP procesorius:

- instrukcijų ciklas – instrukcija vykdoma vienu laikrodžio ciklu;
- instrukcijų vykdymas – galimas lygiagretus vykdymas;
- tinka masyvo apdorojimo operacijoms;
- adresavimo režimas – tiesioginio ir netiesioginio adresavimo režimas;
- skaičiavimo vienetai – trys atskiri skaičiavimo vienetai: ALU, MAC, *Shifter*;
- adreso generavimas – adresą generuoja DAG ir programos sekvencinis derinys;
- programos srauto valdymas – programos sekvencinis ir komandų registras rūpinasi programos eiga;
- atmintys – atskiros duomenų ir programų atmintys;

– operandų gavimas – vienu metu gaunami keli operandai;
– lusto adresų ir duomenų magistralės – atskiros adresų ir duomenų magistralės programoms ir duomenų atmintims, pvz., DMA, DMD, PMD, PMA ir R magistralei;
– konvejerinis sujungimas – sujungimas yra susijęs su komandų registru ir komandų talpyklomis;
– adreso ir duomenų magistralės tankinimas – jos nėra sutankintos, nei luste, nei išorėje;
– taikymas – kalbos apdorojimas, garso apdorojimas, signalų apdorojimas, masyvo apdorojimas ir kt. [17, 18, 19, 20].

Standartinis mikroprocesorius:
– instrukcijų ciklas – vienos komandos vykdymui reikalingi keli laikrodžių ciklai;
– instrukcijos vykdymas –instrukcijų vykdymas yra nuoseklus;
– tinka bendrosios paskirties apdorojimui;
– adresavimo režimas – tiesioginis, netiesioginis ir kt.;
– skaičiavimo vienetai – pagrindinis vienetas: ALU;
– adreso generavimas – programos skaitiklis didinamas nuosekliai, kad būtų sukurtas adresas;
– programos srauto valdymas – programos skaitiklis rūpinasi vykdymo eiga;
– atmintys – paprastai atskirų atminčių nėra;
– operandų gavimas ir atmintis – operandai gaunamai nuosekliai;
– lusto adresų ir duomenų magistralės – adresų ir duomenų magistralės yra dvi lusto magistralės;
– konvejerinis sudarymas – eilės sudarymas atlieka eksplikaciją pagal vieną eilės registrą, kad palaikytų konvejerinį sudarymą;
– adreso ir duomenų magistralės tankinimas – adresai ir duomenų magistralės yra sutankinamos;
– taikymas – bendrosios paskirties programos [17, 18, 19, 20].

1.3. Programinės įrangos apžvalga

Projekte naudojamas ESP32 mikrovaldiklis, kuriam programuoti galima naudoti skirtingas programas, tokias kaip:
– *Arduino* IDE (angl. *integrated development environment*);
– *Espressif* IDE;
– *Micropython*;
– *Eclipse* IDE
– ir kt.

Be to, šį valdiklį galima programuoti skirtingomis kalbomis, tokiomis kaip:
– *C*/*C++*;
– *JavaScript*;
– *LUA*;
– ir kt.

1.3.1. Programavimo įrankiai *Espressif* IDE ir *Eclipse* IDE

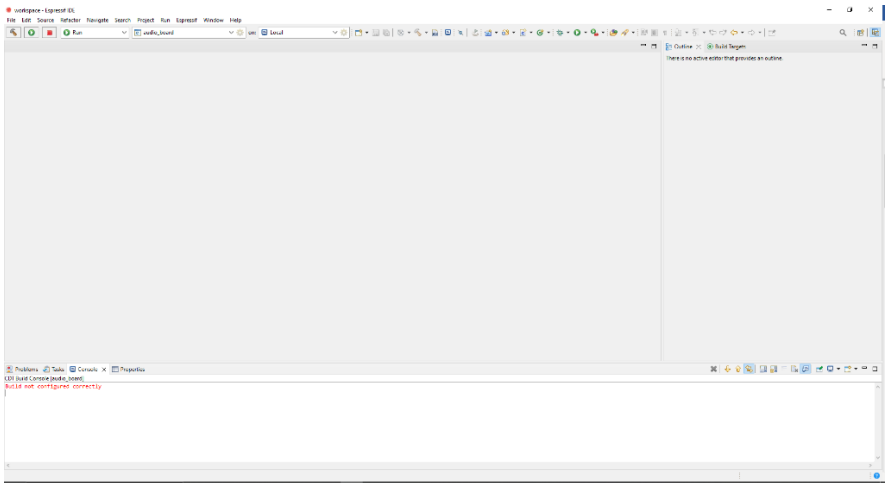
Espressif IDE yra originali ESP32 mikrovaldiklio gamintojo programinė įranga, skirta ESP32, ESP32-S ir ESP32-C serijų mikrovaldikliams. Sukurta *Espressif* IDF (angl. *IoT development framework*) pagrindu bei *Eclipse* CDT (angl. *C*/*C++ development tooling*) programavimo įrankiu.

Šioje programoje galima programuoti tokiomis kalbomis kaip *C* ir *C++*. Šiuo metu pasaulyje veikia milijonai įrenginių programuotų naudojus *Espressif* IDF, nuo paprastų lempučių, žaislų, iki buitinės technikos ir pramonės įrenginių. Programos valdymo langas pateiktas 16 paveiksle.

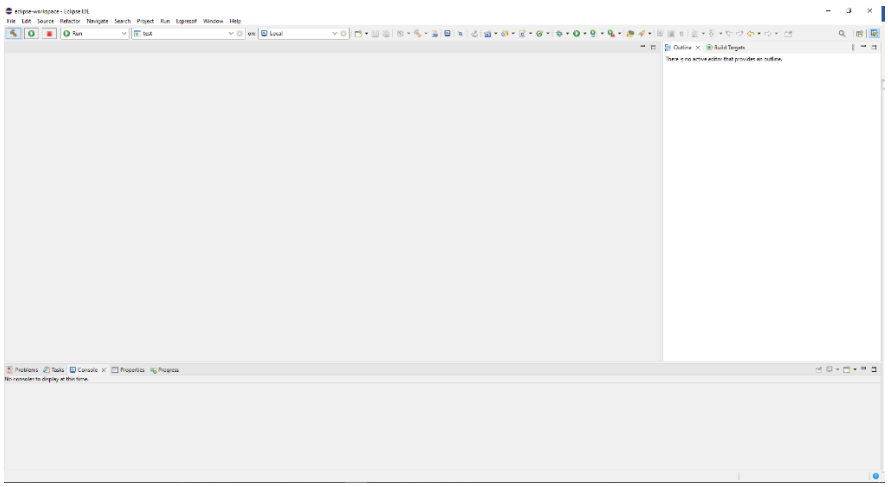
Kadangi *Espressif* IDE yra sukurta naudojant *Eclipse* CDT programos išvaizda (žr. 16 pav.) mažai kuo skiriasi nuo pirminės versijos.

ESP-IDF įskiepio *Eclipse* IDE programoje (žr. 17 pav.) ir IDF-*Eclips*e įskiepis *Espressif* IDE programoje sutampa.

Tam, kad geriau išnaudoti mikrovaldiklių galimybes ir supaprastinti programų rašymą yra sukurtos specialios bibliotekos skirtos signalams apdoroti, pvz. „ESP-DSP“ biblioteka.



16 pav. *Espressif* IDE programos valdymo langas



17 pav. *Eclipse* IDE programos valdymo langas

Šia biblioteką sudaro tokios funkcijos:

- „Dot-product“ – apskaičiuoja dviejų vektorių sandaugą;
- „FFT“ – greitosios Furjė transformacijos funkcija;
- „DCT“ – diskrečioji kosinuso transformacijos funkcija;
- „IIR“ – begalinio impulso atsako filtras;
- „FIR“ – baigtinio impulso atsako filtras;
- „Math“ – pagrindinės vektorinės operacijos;

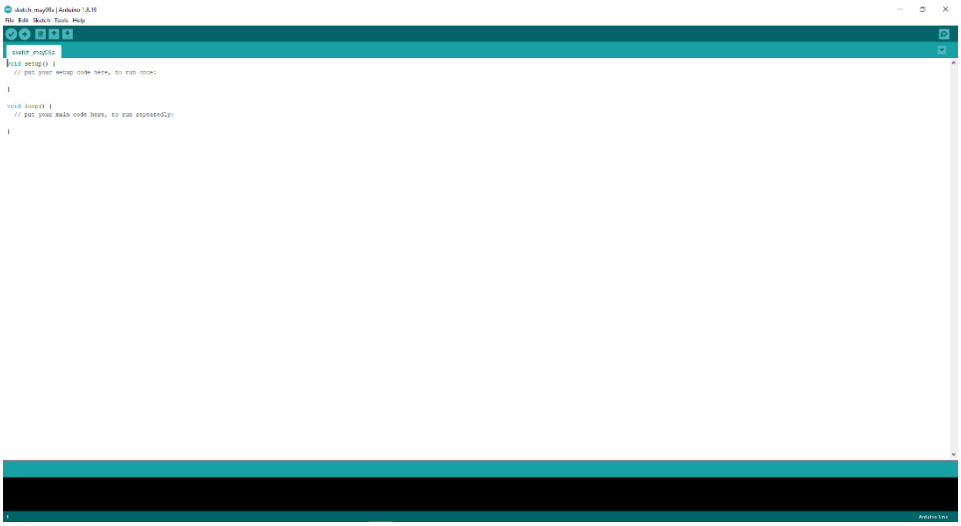
– „Conv“ – konvoliucijos / koreliacijos funkcija;
– „Support“ – palaikymo funkcija;
– „Window functions“ – greitosios *Furjė* transformacijos generavimo funkcija [21, 22].

Day [37] pateikia *Eclipse* programos panaudojimo vieną iš būdų – „sumanusis mokytojo draugas“. Savo darbe [37] mokslininkas pateikia programavimo *Eclipse* programa pavyzdžius, kuriais jis užprogramuoja valdiklį atlikti dirbtinio intelekto uždavinius, tokius kaip pagalba studentams programuojant. Iš atliktų apklausų nustatyta, kad 68 proc. studentų tokia pagalba patiko, o 23 iš 28 studentų išreiškė norą naudotis tokiu pagalbininku sprendžiant programavimo uždavinius.

1.3.2. Programavimo kalba *Arduino* IDE

Dar viena dažnai naudojama programinė įranga yra *Arduino* IDE. Nors ši programinė įranga ir yra pritaikyta *Arduino* šeimos mikrovaldikliams, tačiau ja galima programuoti ir ESP32 mikrovaldiklius. *Arduino* IDE programos pagrindinis langas pateiktas 18 paveiksle.

Kaip ir prieš tai aptartose programose, *Arduino* IDE taipogi turi bibliotekas skirtas signalams apdoroti. Viena iš tokių bibliotekų yra „yummyDSP“ biblioteka.

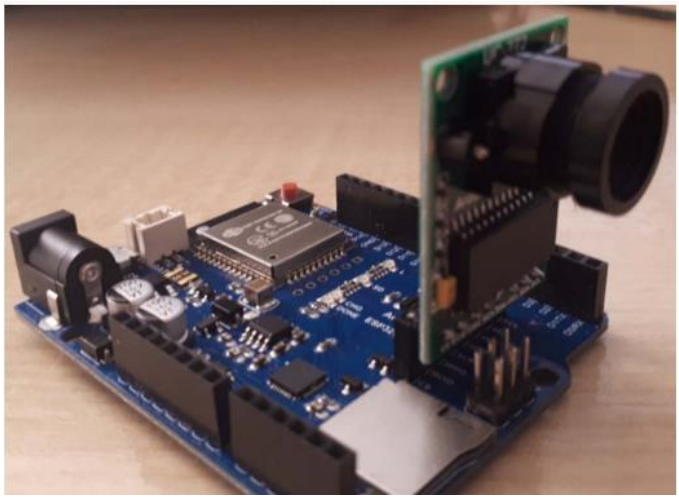


18 pav. *Arduino* IDE programos valdymo langas

Šią biblioteką sudaro tokios funkcijos:
– „Hipass“ – aukšto dažnio filtras;
– „Lopass“ – žemo dažnio filtras;
– „Bandpass“ – plačiajuostis filtras;
– „Waveshaper“ – bangos keitimo funkcija;
– „Saturation“ – sodrinimo funkcija;
– „Delay“ – užlaikymo funkcija [23].

ESP32 panaudojimą kartu su *Arduino* IDE programavimo įrankiu aptaria Rai [30], sukūręs išmaniąją stebėjimo sistemą (žr. 19 pav.). Jo [30] sukurtas prototipas gali būti prijungtas prie vartotojo bevielio interneto prieigos, yra nešiojamas ir palyginus su kitais tiekėjais pigus.

Mohamed [36] pateikė *Arduino* IDE panaudojimo vieną iš galimybių, saulės elementų švarumui aptikti, panaudojus nešvarumo jutiklį.



19 pav. ESP32 UNO su kameros moduliu

Nors tokia sistema ir nepasiteisino dėl dienos metu skleidžiamų didelių triukšmų, tačiau patobulinus įrangą, būtų galima aptikti saulės elementų užterštumą.

1.4. Ultragarso keitikliai

Mechaninių bangų siuntimui ir priėmimui ultragarsiniuose prietaisuose šiuo metu dažniausiai naudojami du metodai: pjezoelektrinis ir elektrostatinis.

Ultragarsiniai keitikliai naudojami mechaniniams virpesiams, turintiems ultragarsinį dažnį, priimti ir perduoti. Didžioji dalis šiuo metu rinkoje esančių ultragarsinių keitiklių yra gaminami iš pjezoelektrinių medžiagų. Kintamo stiprumo elektriniam laukui veikiant pjezoelektrines medžiagas, jos deformuojasi taip sukurdamos atitinkamo dažnio bangas. Priimant signalą, dėl aplinkos slėgio svyravimų, keičiasi medžiagos forma, taip gaunamas elektrinio lauko pokytis, kuris atitinka gaunamo signalo slėgį [38].

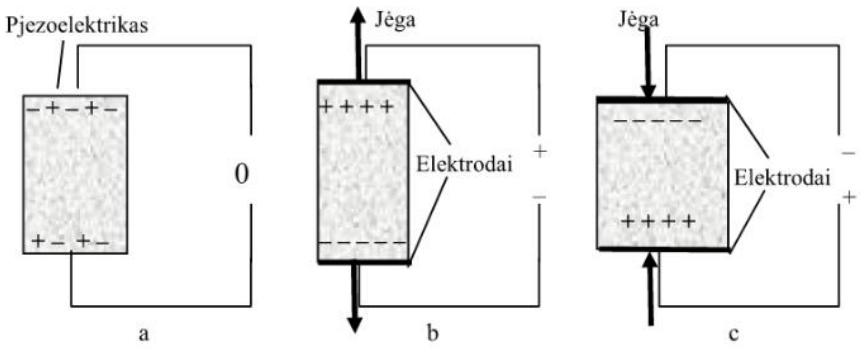
1.4.1. Pjezoelektriniai keitikliai

Šiuo metu vienas iš populiariausių ultragarso generavimo metodų yra pjezoelektrinis metodas. Šis metodas pagrįstas elektriniu lauku veikiamos medžiagos keliamais mechaniniais virpesiais. Šie virpesiai atsiranda poliarizuojantis medžiagos dalelėms. Atsižvelgiant į energijos kilmę (priimama ar atiduodama) pjezoelektrinis efektas gali būti tiesioginis arba atvirkštinis. Jei mechaniniai virpesiai virsta elektriniais, vyksta tiesioginis pjezoelektrinis efektas (žr. 20 pav.). Elektriniams virpesiams virstant mechaniniais, vyksta atvirkštinis pjezoelektrinis efektas (žr. 21 pav.). Kristalą veikiant mechanine energija, pjezoelektrinis kristalas poliarizuojasi, o veikiamas elektrinio lauko – deformuojasi.

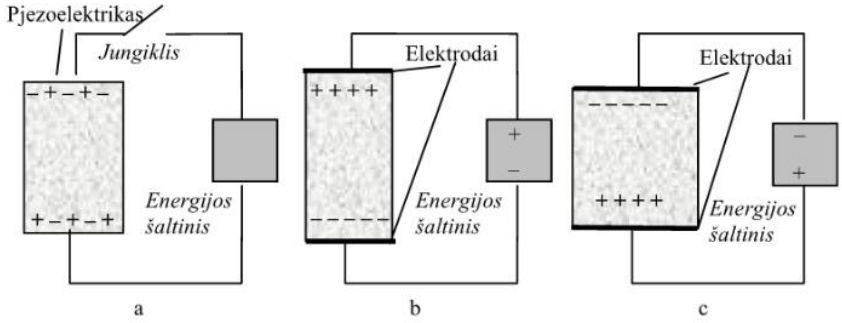
Daugeliui pjezoelektrinių keitiklių pasiekus aukštesnę temperatūrą nei 70 ^{o}C dingsta jų pjezoefektas. Taip nutinka dėl *Kiuri* temperatūros [39].

20 paveiksle pavaizduotas tiesioginis pjezoelektrinis efektas, kuriuo pasinaudojus keitiklį galima naudoti kaip įrašymo priemonę.

21 paveiksle pavaizduotas atvirkštinis pjezoelektrinis efektas.



20 pav. Tiesioginis pjezoelektrinis efektas: a) pjezoelektrikas ramybės būsenoje; b) pjezoelektriką veikia vienos krypties jėga; c) pjezoelektriką veikia kitos krypties jėga [39]



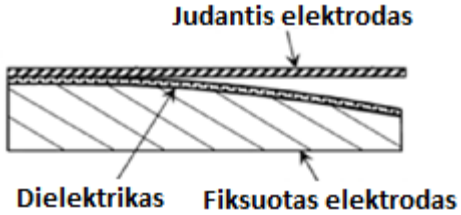
21 pav. Atvirkštinis pjezoelektrinis efektas: a) pjezoelektrikas ramybės būsenoje; b, c – atvirkštinis pjezoelektrinis efektas [39]

Pasinaudojus šiuo efektu, keitiklį galima naudoti kaip bangų skleidimo priemonę.

1.4.2. Elektrostatinis keitiklis

Nors pjezoelektriniai keitikliai dėl savo ištobulintos gamybos ir gerų elektromechaninių parametrų ir yra populiarūs ultragarso sistemose, tačiau jie pasižymi siauru dažnių juostos pločiu. Elektrostatiniai keitikliai (žr. 22 pav.), kitaip nei pjezoelektriniai pasižymi plačia dažnių pralaidumo juosta [40, 41]. Naudojant elektrostatinius keitiklius gali būti pasiektas didesnis negu 100 proc. darbo dažnių juostos plotis. Elektrostatiniai keitikliai gali veikti esant aukštiems dažniams, siekiantiems 100 MHz [39].

Paprasčiausią elektrostatinį keitiklį sudaro du elektrodai, esantys netoli vienas kito. Žadinant šiuos elektrodus kintama įtampa prasideda mechaniški virpesiai virpinantys paviršių, sudarant akustines bangas [42].



22 pav. Elektrostatinio keitiklio struktūra [42]

22 paveiksle pateiktas supaprastinta elektrostatinio keitiklio struktūra.

1.5. Ultragarso generatoriai

Ultragarso generavimas yra procesas, kai sukuriamos garso bangos, kurių dažnis yra didesnis nei žmogaus klausos diapazonas. Šios garso bangos yra sukuriamos ultragarso generatoriais. Yra keletas skirtingų ultragarso generatorių tipų, kiekvienas jų turi savo unikalias charakteristikas ir taikymo sritis. Tačiau visų jų pagrindas – maitinimo šaltinis, valdymo blokas ir keitiklis.

Keitiklis, pagrindinis generatoriaus komponentas, yra atsakingas už elektros energijos pavertimą mechaninėmis vibracijomis. Paprastai jį sudaro pjezoelektriniai kristalai arba kitos medžiagos, kurios keičia formą veikiamos elektros srovės. Greitai keičiant tiekiamą įtampą, keitiklis sukuria pageidaujamo dažnio vibracijas. Ultragarsas gali būti generuojamas naudojant įvairius metodus, įskaitant pjezoelektrinius, elektrostatinius ir naudojant magnetinius laukus mechaninių bangų generavimui.

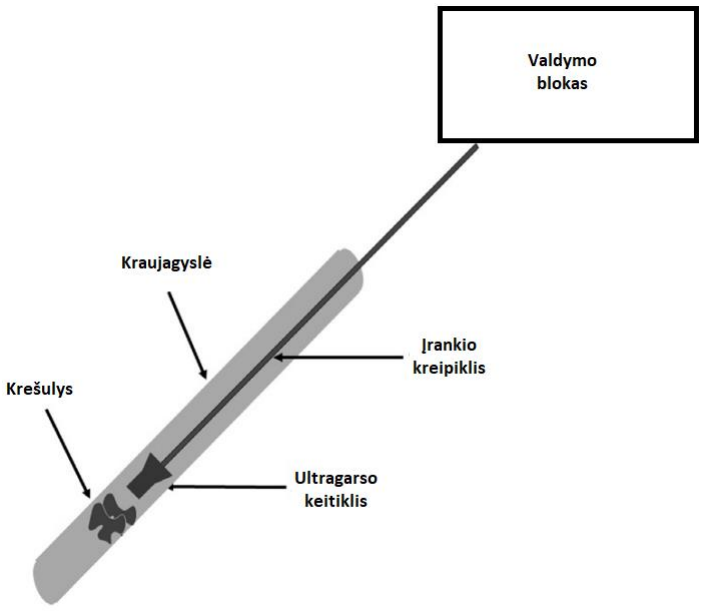
Valdymo blokas valdo ultragarso bangų dažnį, galią ir trukmę, todėl šiuos parametrus galima keisti pagal savo poreikį. Kai kuriais atvejais valdymo blokas taip pat gali būti atsakingas ir už ultragarso bangos krypties valdymą, kad būtų galima sutelkti ultragarso bangą į konkrečias sritis ar taikinius [49, 50, 51].

1.5.1. Ultragarso generatorių tipai ir panaudojimas

Vienas iš ultragarso generatorių tipų yra žemų dažnių generatoriai (dažnis mažesnis nei 1 MHz). Šie generatoriai naudojami daugelyje medicininių procedūrų, tokių kaip pvz., sąnarių terapija ir raumenų stimuliacija. Žemų dažnių ultragarso vibracijos gali pagerinti kraujotaką, sumažinti skausmą ir patinimą, taip pat padėti atkurti suvaržytus judesius. Be to, ultragarsu galima suskaldyti inkstų akmenis, bei išvalyti užsikimšusias kraujagysles [49].

23 paveiksle pateikiamas kraujagyslių valymo ultragarsu pavyzdys. Valymo įrenginys sudarytas iš:

- valdymo bloko;
- įrankio kreiptuvo;
- ultragarso keitiklio.



23 pav. Kraujagyslių valymas ultragarsu [49]

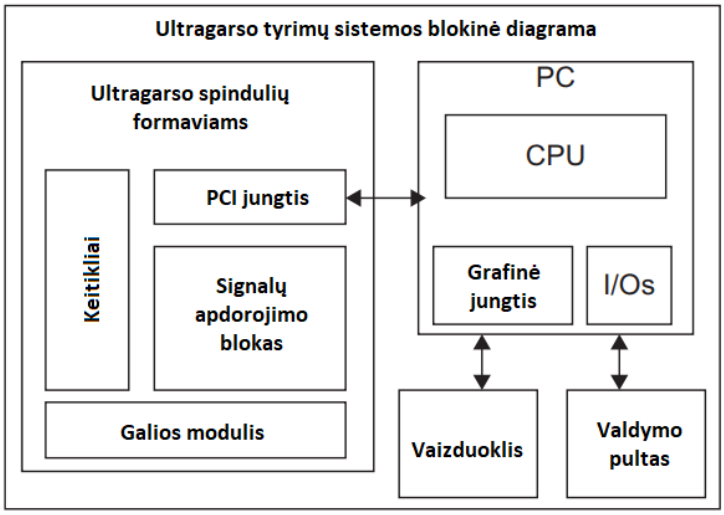
Norint išvalyti kraujagyslę nuo krešulio, įrankio kreiptuvo, pagaminto iš plonos vielos, pagalba ultragarso keitiklis (įrankis) kraujagysle nugabenamas iki krešulio. Pasiekus krešulį, valdymo blokas siunčia aukšto dažnio elektrinius impulsus į ultragarso keitiklį, kuris šiuos impulsus paverčia mechanine energija, taip suskaldydamas krešulį [49].

Kitas ultragarso generatorių tipas yra aukšto dažnio generatoriai (dažnis virš 1 MHz). Šie generatoriai dažnai naudojami medicinoje kaip diagnostikos įrankis, taip pat kirpimo, suvirinimo ir valymo procesuose.

Ultragarso generatoriai dažnai naudojami medicinoje kaip diagnostikos įrankis. Ultragarso skenavimo technologija yra plačiai naudojama gydant pacientus, siekiant vizualizuoti vidaus organus ir kitas kūno struktūras. Ultragarso skenavimas yra neinvazinis, saugus ir efektyvus būdas nustatyti ligas ir diagnozuoti sveikatos sutrikimus. Vaizdavimo technologijose ultragarso generatoriai gali būti naudojami kaip dalis echoskopijos arba kaip garso fokusavimo įrankis.

24 paveiksle pateikta ultragarso tyrimų sistemos blokinė diagrama. Sistemą sudaro:

- centrinis procesorius (angl. *central processing unit*, trump. CPU);
- grafinė jungtis;
- įvesties – išvesties jungtis (angl. *inputs - outputs*, trump. I/Os);
- vaizduoklis;
- valdymo pultas;
- galios modulis;
- signalų apdorojimo blokas;
- periferinė sąsaja (angl. *peripheral component interconnect*, trump. PCI);
- keitikliai.



24 pav. Ultragarso tyrimų sistemos blokinė diagrama [51]

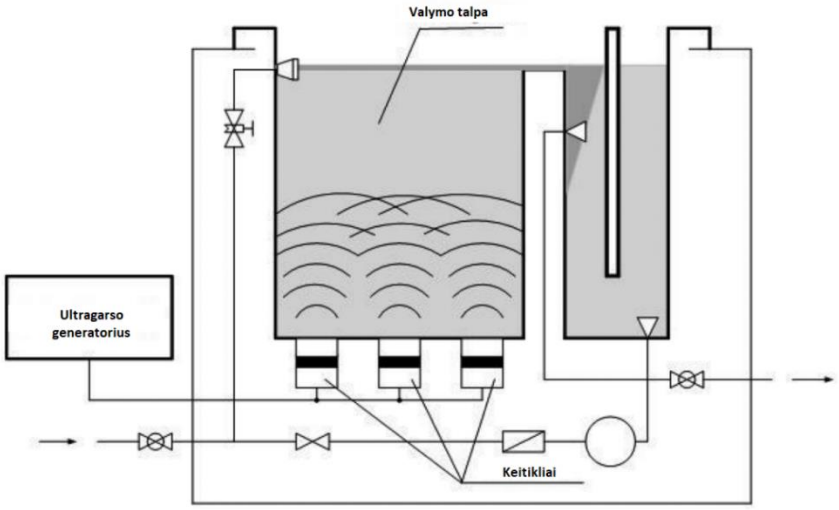
Valdymo pulto pagalba nusistačius norimus parametrus, keitiklio nuskaityti ir signalų apdorojimo bloko apdoroti duomenys atvaizduojami vaizduoklyje [51].

Ultragarso generatorius galima naudoti ir valymui. Valymo procesas vyksta, kai ultragarso vibracijos sukelia kavitaciją, t. y., burbulų susidarymą, kurie skildami sugraužia teršalus nuo valomų paviršių.

Dėl šios priežasties ultragarso valymo procesas yra labai efektyvus ir naudojamas įvairiose pramonės šakose, tokiose kaip medicina, automobilių pramonė, elektronika ir kt. [51].

Ultragarsinį valymo įrenginį (žr. 25 pav.) sudaro:

- valymo talpa;
- ultragarso generatorius;
- keitikliai.



25 pav. Ultragarsinis valymo įrenginys [53]

Valymo įrenginys gali būti tiek su keitikliais pritvirtintais prie talpos, tiek su kilnojamais keitikliais, kuriuos galima naudoti su bet kokia talpa, svarbu, kad talpoje būtų skysčio, kad vyktų kavitacijos procesas [53].

Be to, ultragarso generatoriai naudojami ir suvirinimo procesams. Ultragarso vibracijos naudojamos kaip priemonė sukurti aukštą slėgį tarp dviejų medžiagų, siekiant sujungti jas į vieną. Tai leidžia greitai ir efektyviai sujungti medžiagas, taip pat sumažinti energijos suvartojimą ir išlaikyti gerą kokybę [54].

26 paveiksle pateikta suvirinimo ultragarso įrenginio blokinė diagrama. Ją sudaro:

- ultragarso generatorius;
- keitiklis;
- suvirinimo antgalis;
- amplitudės jutiklis.

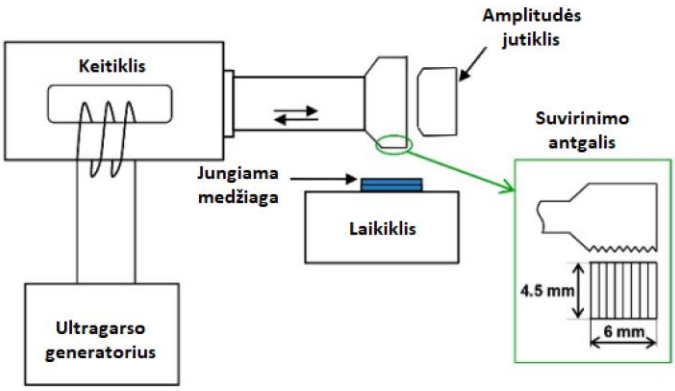
Ultragarso generatoriui veikiant keitiklį, jis veikia suvirinimo antgalį, kuris sukelia aukštą slėgį tarp jungiamų medžiagų, taip jas sujungdamas [54].

Ultragarso generatoriai taip pat naudojami kirpimui ir pjovimui. Ultragarso vibracijos sukuria aukštą slėgį tarp dviejų medžiagų, kuris sukelia suskaidymą ar išpjovimą. Ši technologija yra ypač naudinga tais atvejais, kai tradicinės pjovimo ar kirpimo priemonės nėra efektyvios arba nepajėgia išgauti reikiamų formų.

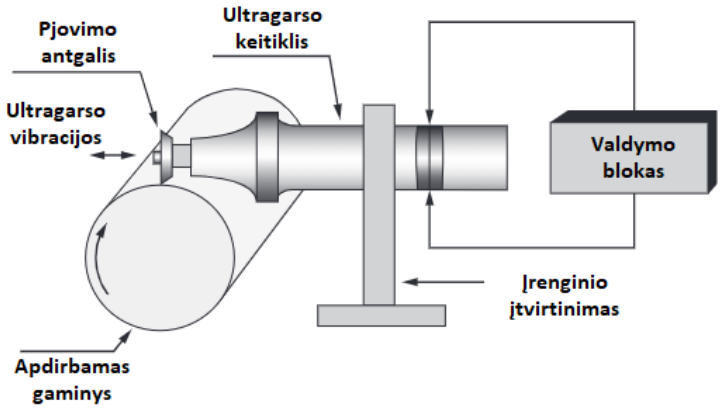
27 paveiksle pateiktas ultragarsinio pjovimo įrenginio pavyzdys. Įrenginį sudaro:

- valdymo blokas;

– ultragarso keitiklis;
– pjovimo antgalis;
– įrenginio stovas.



26 pav. Suvirinimo ultragarsu įrenginio blokinė diagrama [54]



27 pav. Ultragarsinio pjovimo įrenginys [53]

Pateiktame pavyzdyje apdirbamas gaminys sukasi tekinimo staklėse, o valdymo blokas keitiklio pagalba perduoda aukšto dažnio vibracijas į pjovimo antgalį. Vibruodamas antgalis, pagal užduotus parametrus, daro įpjovas apdirbamame gaminyje [53].

Trečias ultragarso generatoriaus tipas yra plokštuminis generatorius, kuris dažnai naudojamas medicinoje kaip diagnostikos įrankis, bei plačiai taikomas pramonėje. Plokštuminis ultragarso generatorius gali sukurti platų ultragarso spindulį, leidžiantį apdoroti didelius medžiagų paviršių plotus.

Ultragarso generatorius galima naudoti neardomiesiems bandymams, kurių metu įvertinama medžiagų vidinė struktūra nepažeidžiant gaminio. Garso bangos prasiskverbia į medžiagą ir atspindi visus vidinius defektus, todėl galima tiksliai patikrinti gaminio struktūrą [53].

Be medicinos ir pramonės taikymų, ultragarso generatoriai taip pat naudojami kitose srityse, tokiose kaip maisto apdorojimas ir vaizdavimo technologija. Pavyzdžiui, maisto apdorojimo pramonėje ultragarso generatoriai gali būti naudojami:

- Pjovimui. Vienas iš dažniausių ultragarso panaudojimo maisto perdirbimo srityje yra pjovimas. Ultragarso pjovimo procesas apjungia vibracijos energiją su įprastu peilio judėjimu, siekiant pagerinti pjovimo kokybę. Pjovimo peilis vibracijas atlieka labai greitai, todėl sumažinamas bendras pjovimo jėgos kiekis, išvengiama skilinėjimo ir grūdimo, o pjaunamas paviršius išlieka labai lygus. Kitas pjovimo ultragarsu pranašumas yra tas, kad maistas nelimpa prie peilio. Tai yra ypač efektyvu pjaunant lipnų maistą. Taip pat tai gerai tinka šaldytam, trapiam ir nehomogeniškam maistui.
- Maisto konservavimas. Ultragarso naudojimas gali būti naudingas mikroorganizmų ir fermentų aktyvumui sumažinti, siekiant išsaugoti maisto produktų kokybę ilgesnį laiką. Ultragarso bangų sukeliama kavitacija sukelia ultragarso antimikrobinį poveikį.
- Filtravimas. Ultragarso panaudojimas gali padidinti medžiagų srautą filtravimo metu, nes vibracijos neleidžia užsikišti filtrams. Naudojant ultragarsą galima filtruoti pramonines nuotekas, vaisių sultis ir ekstraktus.
- Dehidratacija. Ultragarsas gali būti naudojamas siekiant padidinti vaisių ir daržovių osmosinį džiūvimą. Naudojant ultragarsą osmosiniam džiovinimui, drėgmės ištraukimo greitis yra didesnis, o cukraus kiekis produktuose išlieka didesnis. Didelis oro srautas panaikina bet kokią ultragarso naudą.
- Šaldymas ir atšildymas. Daugybė tyrimų patvirtino, kad ultragarso vibracijos nuslopina ledo kristalus, taip pagreitėja veikiamų produktų atšildymas. Ultragarsas taip pat naudojamas gaminti aukštos kokybės šaldytus maisto produktus, pvz., ledus, kontroliuojant kristalizavimą sonokristalizacijos būdu.
- Mėsos apdorojimas. Ultragarsas taikomas siekiant suminkštinti mėsą, keičiant baltymų struktūrą.
- Ekstrakcija. Ultragarsas gali padėti padidinti ištraukiamos medžiagos kiekį, pvz., sulčių kiekį iš kauliukų, antioksidantų ekstrakciją iš žolelių ir aliejų ekstrakciją iš sėklų [50].

Be ankščiau išvardintų tipų, ultragarso generatorius galima surūšiuoti į analoginius, skaitmeninius ir impulsinius:

- analoginiai generatoriai yra paprasčiausias generatorių tipas ir paprastai naudojami nebrangiuose įrenginiuose. Šie generatoriai sukuria nuolatinę garso bangą fiksuotu dažniu ir amplitude. Pagrindinis analoginių generatorių trūkumas yra tai, kad jie turi prastą dažnio stabilumą, o tai gali turėti įtakos matavimų tikslumui.
- Skaitmeniniai ultragarso generatoriai naudoja skaitmeninį signalo apdorojimą, kad sukurtų tikslias, stabilias garso bangas. Šie generatoriai gali sukurti platų dažnių ir bangų formų spektrą, todėl jie tinka įvairioms reikmėms. Skaitmeniniai generatoriai yra energetiškai efektyvesni nei analoginiai generatoriai, todėl jie idealiai tinka nešiojamiesiems įrenginiams.
- Impulsiniai ultragarso generatoriai sukuria trumpus garso bangų pliūpsnius, atskirtus tylos periodais. Šio tipo generatorius ypač naudingi medicinoje, nes jie leidžia gydytojams atskirti skirtingo tankio audinius [49, 52].

1.5.2. Ultragarso generatoriai medijų sistemose

Dar vienas ultragarso generatorių panaudojimas – garso sistemos. Šie prietaisai sukuria aukšto dažnio garso bangas, kurios yra už žmogaus klausos diapazono ribų. Tačiau tinkamai moduliuojamos ir derinamos šios ultragarso bangos gali generuoti unikalių savybių garsus.

Ultragarso garso generavimas priklauso nuo netiesinės ultragarso bangų sąveikos su oru ar kita terpe. Kai susikerta kelios skirtingo dažnio ultragarso bangos, jos gali sukurti girdimus garsus per procesą, vadinamą „parametrine sąveika“. Ši sąveika sukuria naujus dažnius, kai kurie iš jų patenka į žmogaus klausos diapazoną (20 Hz – 20 kHz).

Ultragarso generatorių taikymas medijose:

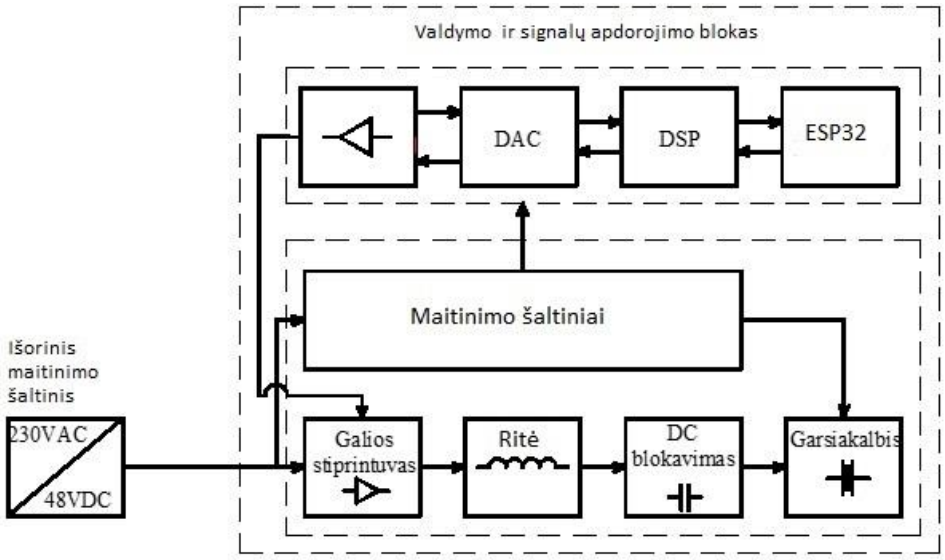
- Kryptinės garso sistemos. Vienas ryškiausių ultragarso generatorių pritaikymo garso generavimui yra kryptingų garso sistemų kūrimas. Šios sistemos naudoja ultragarso bangas, kad sukurtų siaurą garso spindulį, kuris gali būti nukreiptas į konkrečias vietas ar klausytojus. Ši funkcija leidžia mėgautis labiau įtraukiančiomis ir privačiomis klausymosi patirtimi be ausinių ar įprastų garsiakalbių.
- Haptinis grįžtamasis ryšys. Ultragarso generatoriai taip pat gali būti naudojami norint sukurti haptinį grįžtamąjį ryšį, tai yra vibracijų sukuriamas prisilietimo pojūtis. Moduliuojant ultragarso bangas, sukuriami specifiniai vibracijos modeliai, kuriais įrenginiai gali teikti lytėjimo grįžtamąjį ryšį vartotojams, pagerindami vartotojo sąsajas ir virtualios realybės patirtį.
- Ultragarsinis ryšys. Ultragarso garso generavimas gali būti naudojamas ryšių sistemose, ypač trumpojo nuotolio ryšiui ir duomenų perdavimui. Šios sistemos naudojasi unikaliomis ultragarso bangų savybėmis, tokiomis kaip jų gebėjimas prasiskverbti pro kietus objektus ir mažas jautrumas trikdžiams.

Ultragarso generatorių naudojimas garso generavimui turi keletą privalumų. Pirma, jie gali sukurti kryptingą garsą, suteikdami labiau sutelktą ir įtraukiantį klausymosi patirtį. Antra, jie gali generuoti garsą be tradicinių garsiakalbių komponentų, todėl garso sistemos gali būti kompaktiškesnės ir efektyvesnės.

Nors ultragarso generatoriai turi daug privalumų, tokie kaip greitis, efektyvumas ir maža poveikio aplinkai rizika, jie taip pat turi minusų, susijusių su jų naudojimu. Jei ultragarso bangos yra nukreiptos tiesiai į odą, tai gali sukelti termišką pažeidimą, o jei jos yra nukreiptos į akis, tai gali sukelti akių pažeidimus. Be to, ultragarso garso efektyvumą ir kokybę gali paveikti tokie veiksniai kaip oro temperatūra, drėgmė ir kliūčių buvimas. Todėl ultragarso generatoriai turėtų būti naudojami tik patvirtintų ir atsakingų specialistų, kurie žino, kaip juos naudoti saugiai ir efektyviai [47].

2. Eksperimentinė dalis

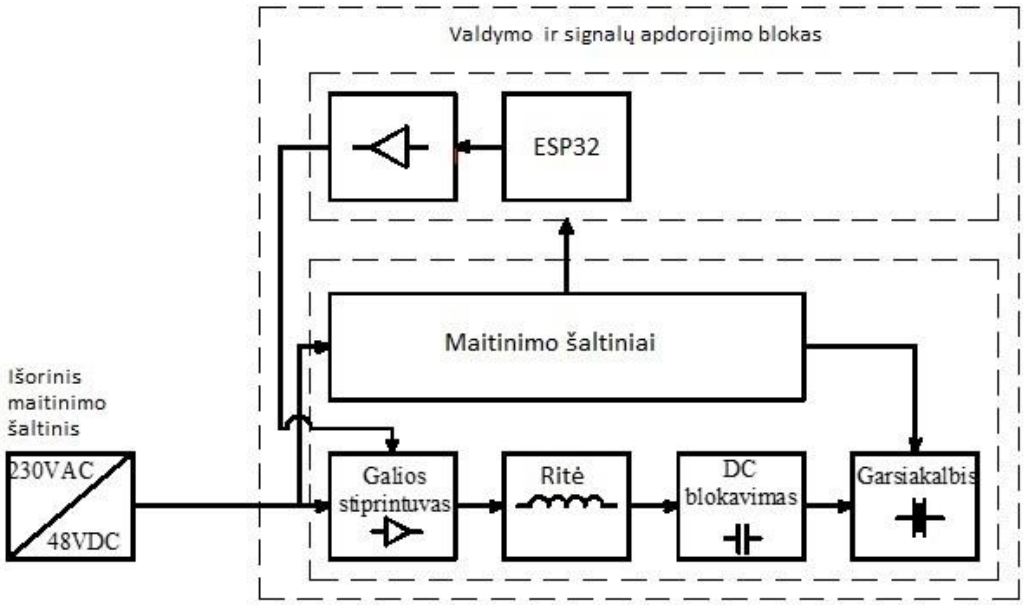
Projekte analizuojami plačiajuosčiai ultragarso generatoriai, kurie realizuoti, panaudojant ADAU1466 DSP ir ESP32 procesorius, taip pat palyginami generatoriai, naudojantys DSP kartu su ESP32 ir naudojantys tik ESP32. Plačiajuosčio ultragarsinio generatoriaus naudojančio DSP blokinė schema pateikta 28 paveiksle.



28 pav. Plačiajuosčio ultragarsinio generatoriaus su DSP blokinė schema

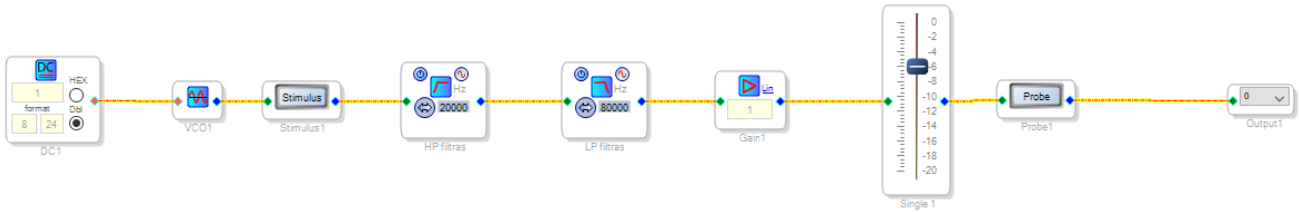
ADAU1466 procesorius savyje neturi pastovios atminties valdymo programos kodui ir skaitmeninio analoginio keitiklio signalui pakeisti iš skaitmeninio į analoginį, todėl jam reikalingas mikrovaldiklis kodui išduoti, bei skaitmeninis-analoginis keitiklis analoginiam signalui išgauti.

29 paveiksle pateikta sistemos, kuri skirta signalams be DSP ir naudojant tik ESP32 valdiklį apdoroti, blokinė schema.



29 pav. Plačiajuosčio ultragarsinio generatoriaus be DSP blokinė schema

SigmaStudio programine įranga buvo sudarytas programuojamo kintamo dažnio sinusinės bangos generatorius, naudojantis vidinį DSP signalų generatorių bei aukšto ir žemo dažnio filtrus. Programos blokinė schema pateikta 30 paveiksle.



30 pav. DSP sinusinių ultragarso signalų generatoriaus blokinė schema

SigmaStudio programa diagramą paverčia programiniu kodu, kurį pertvarkius galima suderinti su *Arduino* IDE programine įranga. *ESP-prog* programatoriumi parašytas kodas įkeliamas į ESP32 valdiklį, valdantį ir DSP.

Esamai sistemai reikalingas programos kodas DSP ignoruoti, jei jis nėra naudojamas. Kodui sukurti taip pat naudojama *SigmaStudio* programinė įranga. Programos blokinė diagrama pateikiama 31 paveiksle.

31 pav. DSP apėjimo blokinė diagrama

Programiškai DSP įėjimas su išėjimu yra sujungiamas tiesiogiai. 31 paveiksle pateiktą diagramą pavertus į programinį kodą ir jį pertvarkius, galima kurti sinusoidinį ultragarso generatorių naudojant tik ESP32 valdiklį.

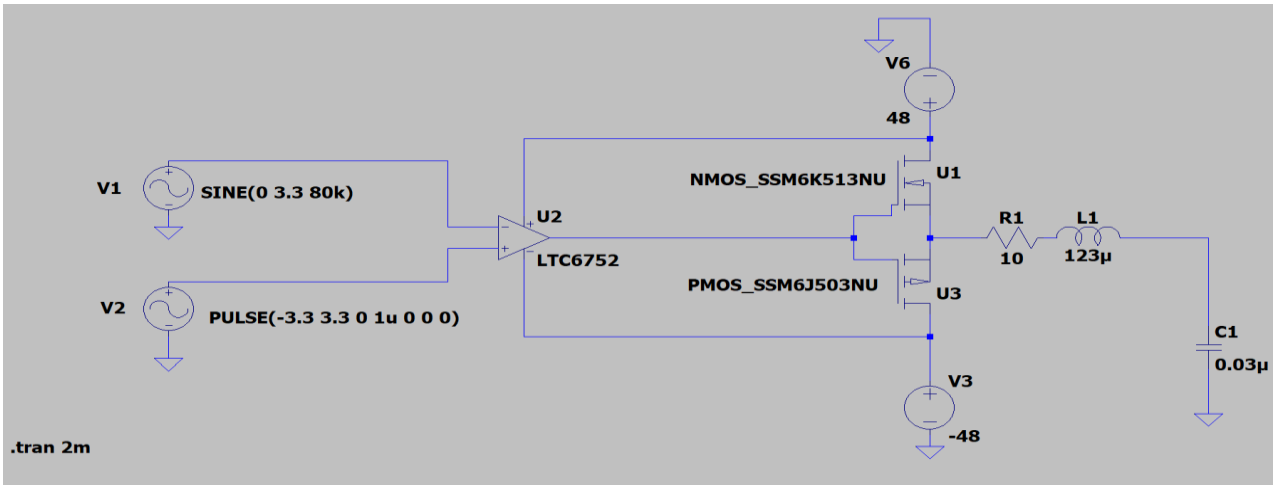
2.1. *LC* žemo dažnio filtro ritės induktyvumo parinkimas

Plačiajuosčio ultragarso generatoriaus sistemos maitinimo šaltinio įtampa siekia tik 48 V. Išėjime esantis elektrostatinis keitiklis žadinamas apie 300 V įtampa, kurią pasiekti šiuolaikine elektronika sudėtinga, todėl tam tikslui panaudotas *LC* filtras veikiantis rezonansiniame režime. Kylant dažniui amplitudė mažėja, todėl būtina keisti ritės arba kondensatoriaus parametrus. Tiriamosios sistemos kondensatorius yra pastovus (30 pF), reikalinga rasti rezonuojančio *LC* filtro ritės induktyvumus 20–85 kHz dažnių juostoje. Šiam tikslui *LTspice* programine įranga sukurtas D klasės stiprintuvo modelis (žr. 32 pav.), kurio išėjime yra *LC* filtras. D klasės stiprintuvo modelis pateikiamas.

Modelį sudaro:

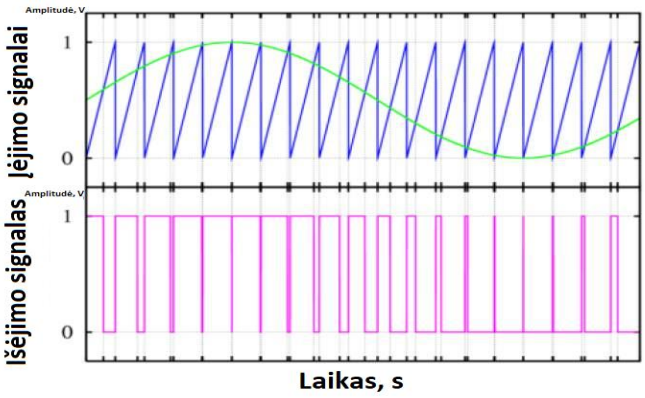
- V1 – sinusinio signalo generatorius, kurio dažnis keičiamas nuo 20 iki 85 kHz, 3,3 V;
- V2 – pjūklinio signalo generatorius, 3 V; bandymams naudotas 1 MHz dažnis, dėl kitokių nei realios sistemos komponentų parametrų;

– U2 – komparatorius;
– U1-N – lauko tranzistorius (angl. *metal oxide semiconductor field effect transistors*, trump. MOSFET);
– U3-P – lauko tranzistorius;
– V3, V6 – maitinimo šaltiniai, 48 V;
– R1 – varža – 10 Ω;
– *LC* filtro dedamosios – ritė *L*1 keičiamo induktyvumo (123–1895 µH) ir kondensatorius *C*1 – 30 pF.



32 pav. D klasės stiprintuvo su *LC* filtru modelis

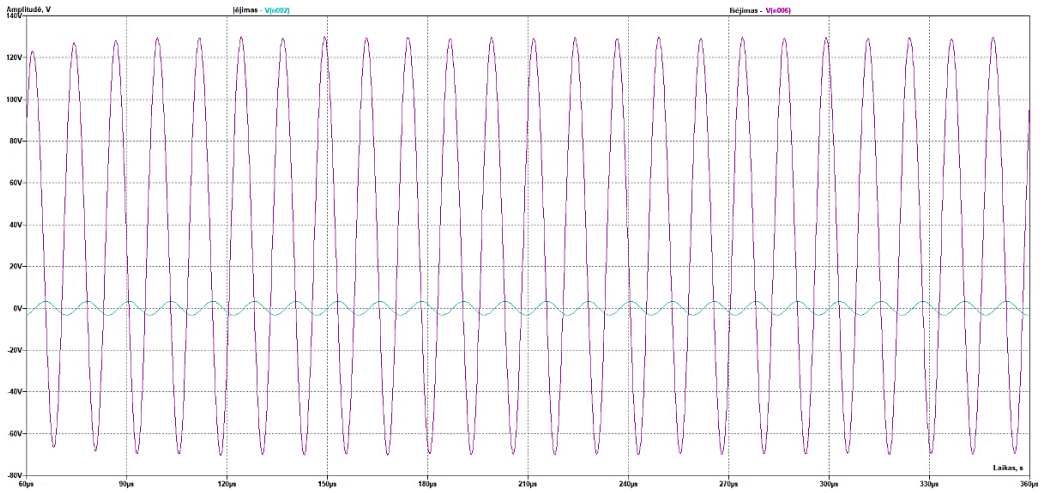
V1 ir V2 signalai sujungiami komparatoriumi. Kai signalas V1 yra aukštesnis nei V2 bangos lygis, išvestis tampa teigiama. Kai signalas V1 yra žemesnis nei V2 bangos lygis, išvestis tampa neigiama. Rezultatas yra impulsų grandinė, kurioje impulso plotis yra proporcingas momentiniam signalo lygiui. Signalų moduliavimo pavyzdys pateiktas 33 paveiksle.



33 pav. Sinusinio ir pjūklinio signalo sujungimas

Kaip matyti iš 33 paveikslo, sujungus sinusinį ir pjūklinį signalus, gaunamas impulsų pločio moduliacijos (angl. *pulse width modulation*, trump. PWM) signalas.

Gautas signalas junginėja „N“ ir „P“ lauko tranzistorius didindamas išeinanti signalą V3 ir V6 įtampomis. Varža R1 reikalinga realesniam rezonansui sukurti. *LC* žemo dažnio filtras atkuria sustiprintą pradinį signalą (žr. 34 pav.). 34 paveiksle esančioje charakteristikoje, *y* ašyje pavaizduota amplitudė, o *x* ašyje – laikas.

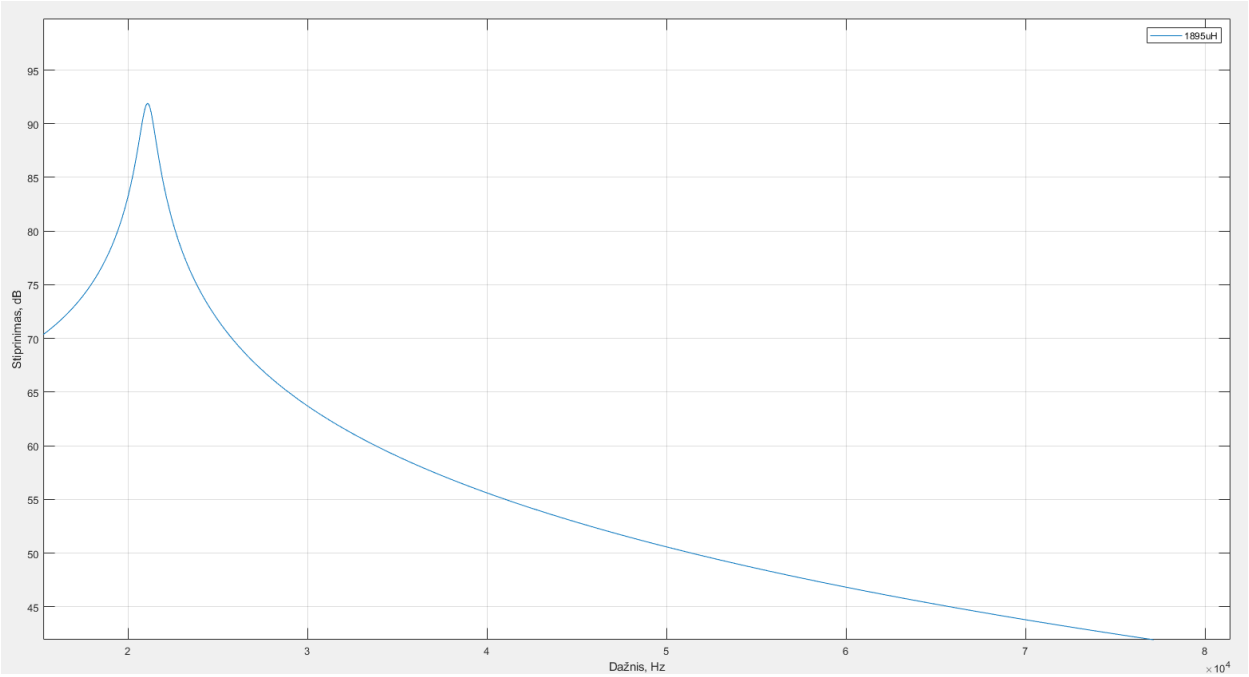


34 pav. Sistemos įėjimo ir išėjimo signalai

Iš 34 paveikslo matyti, kad sistema sustiprina įėjimo signalą rezonuodama, nes signalas sustiprėja daugiau, nei naudojami maitinimo šaltiniai.

Rezonanso induktyvumui nustatyti atliekama modelio simuliacija, naudojant kintamo dažnio sinusinį signalą įėjime V1 ir keičiant L1 induktyvumą. Induktyvumai parenkami 20–85 kHz dažnių juostą padalinus į 16 dalių ir charakteristikos netolygumus palaikant ne didesniame nei 6 dB lygyje.

35 paveiksle pateikiama sistemos charakteristika, kai L1 = 1895 µH. Iš šios charakteristikos matyti, kad esant tokiam ritės induktyvumui sistema rezonuos esant 20–22 kHz.

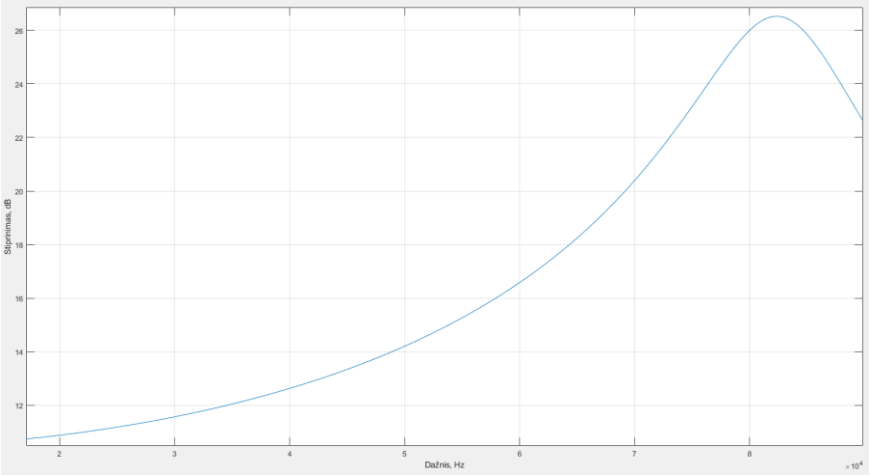


35 pav. Ritės L1 induktyvumas 1895 µH

36 paveiksle pateikiama sistemos charakteristika, kai L1 = 123µH. Iš šios charakteristikos matyti, kad esant tokiam ritės induktyvumui sistema rezonuos esant 80–85 kHz dažniui.

Ričių induktyvumų matavimai pateikiami 1 ir 2 lentelėse.

Kintant dažniui, perjungiant rites relėmis, galima išgauti rezonansinį stiprinimą visoje pasirinktoje dažnių juostoje (žr. 37 pav.).



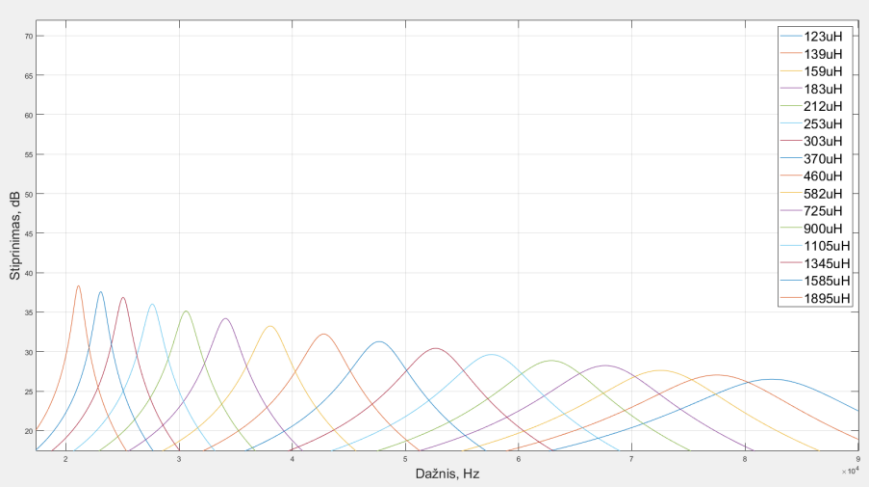
36 pav. Ritės L1 induktyvumas 123 µH

1 lentelė. Ričių induktyvumai 20–45 kHz dažnių diapazone

Dažnių diapazonas, kHz	20–22	22–24	24–26	26–29	29–32	32–36	36–40	40–45
Ritės induktyvumas, µH	1895	1585	1345	1105	900	725	582	460

2 lentelė. Ričių induktyvumai 45–85 kHz dažnių diapazone

Dažnių diapazonas, kHz	45–50	50–55	55–60	60–65	65–70	70–75	75–80	80–85
Ritės induktyvumas, µH	370	303	253	212	183	159	139	123



37 pav. Stiprinimo charakteristikos esant skirtingiems ričių induktyvumams

Iš 37 paveikslo matyti, kad prasilenkiančios charakteristikos sukuria galimybę atlikti signalo stiprinimą visoje dažnių juostoje, o charakteristikų netolygumai neviršija užsiduotų 6 dB.

2.2. ESP32 sinuso formos signalo generatorius

ESP32 sinuso generatorius, generuojantis 20–85 kHz dažnio sinusoidę, programuojamas *Arduino* IDE programine įranga.

2.2.1. Sinusinės formos signalo generavimas naudojant matematines formules

38 paveiksle pateiktas programos kodas, kuris apskaičiuoja sinusą (dešimta eilutė) pagal formulę:

$$128 + 80\left(\sin\left(\frac{\text{deg} * PI}{180}\right)\right); \quad (2.1)$$

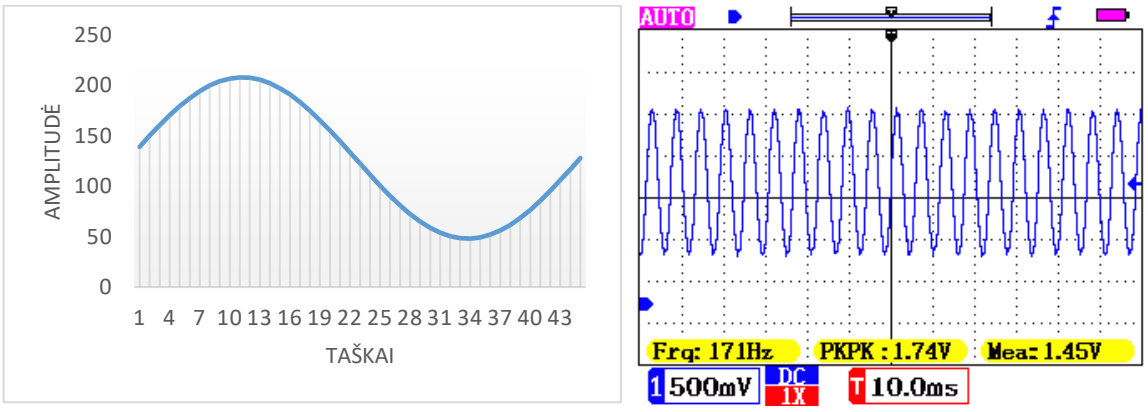
čia deg – laipsnis sinusoidei apskaičiuoti.

Programa apskaičiuoja 45 sinusoidės taškus (viso 360 laipsnių, žingsnis – 8 laipsniai), kurie sudaro sinusoidę ir juos perduoda į 25 valdiklio išėjimo jungtį. Formulėje „128“ nurodo perslinkimą *Y* ašyje, o „80“ sinusoidės amplitudę.

```
1   int sensorValue = 0;
2   int analogPin = 34;
3  void setup() {
4   Serial.begin(115200);
5  }
6
7  void loop() { // Generate a Sine wave
8   for (int deg = 0; deg < 360; deg = deg + 8){
9
10    dacWrite(25, int(128 + 80 * (sin(deg*PI/180))));
11    sensorValue = analogRead(analogPin);
12    Serial.println(sensorValue);
13    delayMicroseconds(1);
14   }
15 }
```

38 pav. Sinusinės formos signalo generatoriaus kodas

39 paveiksle pateikiama sinusinės formos signalas, suformuotas pasinaudojus (2.1) formule.



a) b)

39 pav. Sinusinės formos signalai: a) taškai apskaičiuoti programos kodo; b) realus išėjimas

Atlikti programos bandymai parodė, kad ši programa gali generuoti sinusinės formos signalą, kurių dažnis yra 171 Hz.

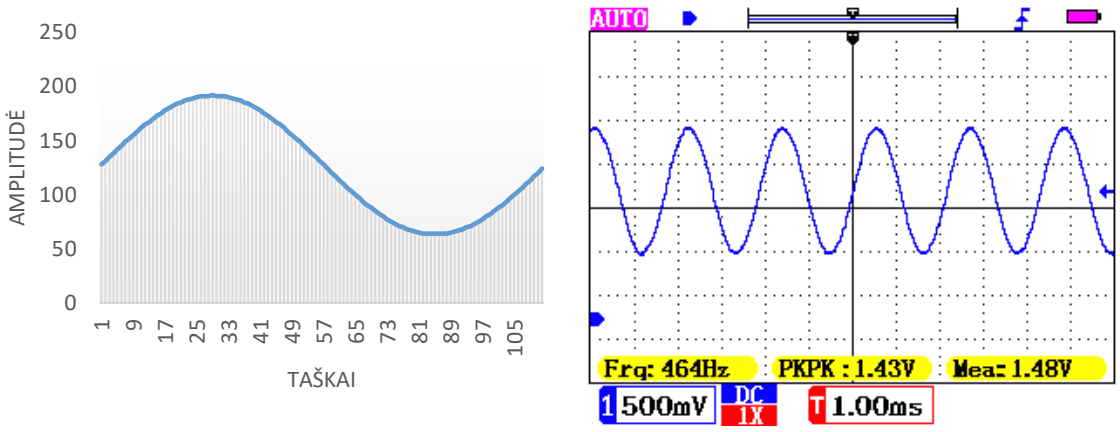
2.2.2. Sinusinės formos signalo generavimas naudojant koordinates

Kitas programos įgyvendinimo būdas – naudoti duomenų masyvą su sinusoidės taškų koordinatėmis (žr. 40 pav.).

```
int sensorValue = 0;
int analogPin = 34;
#define Num_Samples  112
#define MaxWaveTypes 1
int i = 0;
static byte WaveFormTable[MaxWaveTypes][Num_Samples] = {

  {
   0x80, 0x83, 0x87, 0x8A, 0x8E, 0x91, 0x95, 0x98, 0x9B, 0x9E, 0xA2, 0xA5, 0xA7, 0xAA, 0xAD, 0xAF,
   0xB2, 0xB4, 0xB6, 0xB8, 0xB9, 0xBB, 0xBC, 0xBD, 0xBE, 0xBF, 0xBF, 0xBF, 0xC0, 0xBF, 0xBF, 0xBF,
   0xBE, 0xBD, 0xBC, 0xBB, 0xB9, 0xB8, 0xB6, 0xB4, 0xB2, 0xAF, 0xAD, 0xAA, 0xA7, 0xA5, 0xA2, 0x9E,
   0x9B, 0x98, 0x95, 0x91, 0x8E, 0x8A, 0x87, 0x83, 0x80, 0x7C, 0x78, 0x75, 0x71, 0x6E, 0x6A, 0x67,
   0x64, 0x61, 0x5D, 0x5A, 0x58, 0x55, 0x52, 0x50, 0x4D, 0x4B, 0x49, 0x47, 0x46, 0x44, 0x43, 0x42,
   0x41, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x41, 0x42, 0x43, 0x44, 0x46, 0x47, 0x49, 0x4B,
   0x4D, 0x50, 0x52, 0x55, 0x58, 0x5A, 0x5D, 0x61, 0x64, 0x67, 0x6A, 0x6E, 0x71, 0x75, 0x78, 0x7C
  }
};

void setup() {
  Serial.begin(19200);

}

void loop() {
  byte wave_type = 0;

  dacWrite(25, WaveFormTable[wave_type][i]);
  i++;
  if (i >= Num_Samples) i = 0;

   sensorValue = analogRead(analogPin);
   Serial.println(sensorValue);
delayMicroseconds(1);

}
```

40 pav. Sinuso formos signalo generatoriaus kodas naudojant koordinates

Programa generuoja 112 taškų, kurių koordinatės aprašytos šešioliktainiu kodu „WaveFormTable“ skaičių masyve. Naudojant šį masyvą, programa kuria sinusoidinės formos signalą (žr. 41 pav.).



a) b)

41 pav. Sinusoidinės formos signalai: a) iš masyvo taškų; b) realus išėjimas

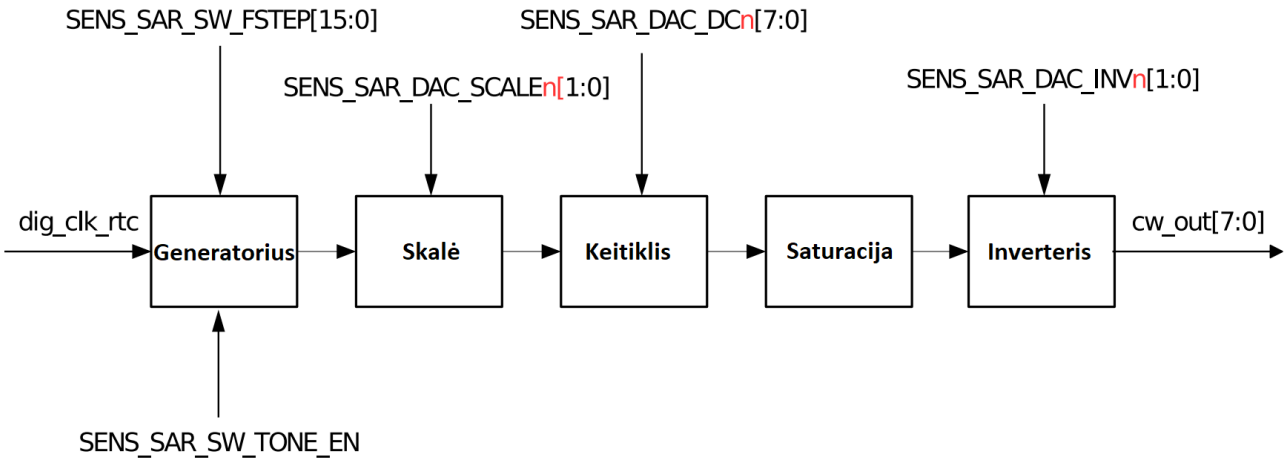
Atlikti programos bandymai parodė, kad generuojamų sinuso formos signalų dažnis 464 Hz. Išgaunamas dažnis nėra didelis, nes pilnai neišnaudojamos skaitmeninio-analoginio keitiklio galimybės, kurios pasiekiamos per specialias bibliotekas.

2.2.3. Sinuso formos signalų generatorius naudojant bibliotekas

Naudojamos tokios bibliotekos:

- „Arduino.h“ – aprašo dažniausiai *Arduino* IDE naudojamas funkcijas, nuo jungčių priskyrimo iki konstantų. Ši biblioteka įgalina ESP32 valdiklyje naudoti *Arduino* valdiklio funkcijas;
- „sens_reg.h“ – aprašo galimus naudoti registrus;
- „rtc.h“ – leidžia keisti procesoriaus parametrus, susijusius su duomenų apdorojimu, tokius kaip procesoriaus taktinis dažnis;
- „dac.h“ – aprašo ESP32 valdiklyje esančių dviejų 8-bitų skaitmeninių-analoginių keitiklių funkcijas.

42 paveiksle pateikiama ESP32 skaitmeninio-analoginio keitiklio sinuso formos signalo generatoriaus principinė schema, kuri gali generuoti ir kosinuso formos signalus.



42 pav. ESP32 skaitmeninio-analoginio keitiklio sinuso formos signalo generatoriaus principinė schema [43]

Principinėje schemoje (žr. 42 pav.) pateikiami bibliotekų kintamieji, kurie nusako generatoriaus parametrus: skalę, signalo invertavimą, generatoriaus ir keitiklio įjungimą.

Registrų paskirtis:

- SENS_SAR_DAC_SCALEn[1:0] – skalės keitimas žingsniu 1, 1/2, 1/4, ar 1/8 karto;
- SENS_SAR_DAC_INVn[1:0] – sinusoidės perslinkimas per 0, 90, 180, 270 laipsnių;
- SENS_SAR_DAC_DCn[7:0] – sinusoidės perslinkimas amplitudinėje ašyje;
- SENS_SAR_SW_TONE_EN – generatorių įjungimas;
- CW_OUT[7:0] – signalo apdorojimas prieš perduodant jį į analoginę jungtį.

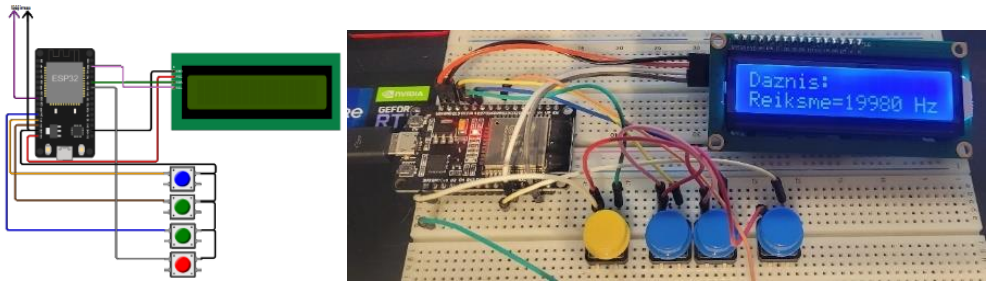
Dažnis keičiamas pagal formulę:

$$f = dig\_clk\_rtc * \frac{SENS\_SAR\_SW\_FSTEP}{65536}; \quad (2.2)$$

čia f – dažnis, Hz;

dig_clk_rtc – procesoriaus takto greitis (dažniausiai 8 MHz);
SENS_SAR_SW_FSTEP – dažnio pasirinkimo registras (SENS_SAR_SW_FSTEP[15:0]) [43].

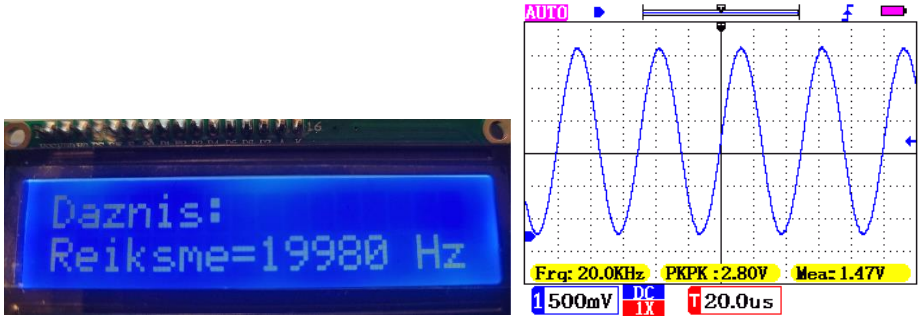
Bandomoji principinė schema pateikta 43 paveiksle. Bandomoji sistema pagaminta, pasinaudojus ESP-WROOM-32 plokšte, keturiais mygtukais, skirtais nustatymams pasirinkti, bei LCD1602 ekranu parametrams vizualizuoti.



a) b)

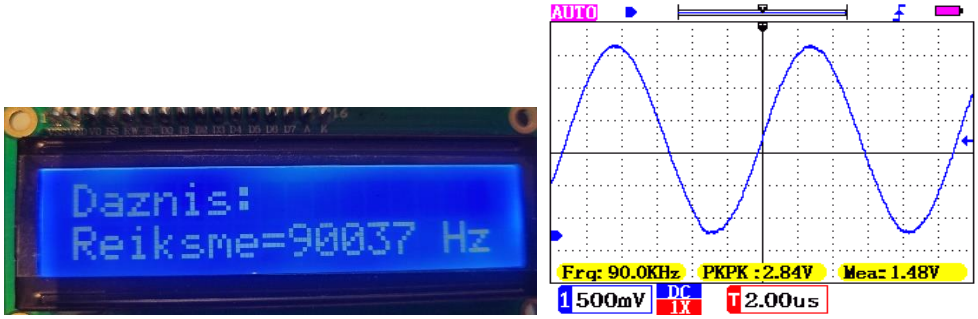
43 pav. Bandomoji principinė schema (a) ir reali bandomoji sistema (b)

Sistema maitinama per USB jungtį. Generatoriaus išėjimas matuojamas tarp įžeminimo ir 25-tos ESP32 išėjimo jungties, kuri taip pat yra pirmojo skaitmeninio-analoginio keitiklio išėjimo jungtis.



a) b)

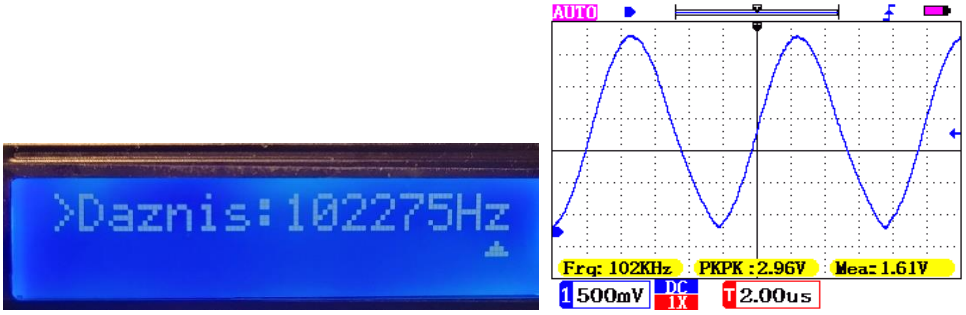
44 pav. ESP32 sinuso formos signalo generatoriaus išėjimas nustačius 20 kHz dažnį: a) nustatytas dažnis; b) sinuso formos signalas



a) b)

45 pav. ESP32 sinuso formos signalo generatoriaus išėjimas nustačius 90 kHz dažnį: a) nustatytas dažnis; b) sinuso formos signalas

Sukurta sistema gali generuoti sinuso formos signalą nuo 1 Hz iki 90 kHz ir didesnį dažnį, tačiau atliekant bandymus pastebėta, kad viršijus 90 kHz, sinuso forma pradeda kisti (žr. 46 pav.). Toliau pateikiami keli sistemos sugeneruoti sinuso formos signalai.



a) b)

46 pav. ESP32 sinuso formos signalo generatoriaus išėjimas nustačius 102 kHz dažnį: a) nustatytas dažnis; b) sinuso formos signalas

Iš 41 paveikslo (žr. 46 pav.) b dalies matyti, kad generuojant 102 kHz dažnio sinuso formos signalą, signalo apatinė dalis deformuojasi, nes ESP32 valdikliui nebeužtenka greitaveikos tokio dažnio sinuso formos signalams generuoti.

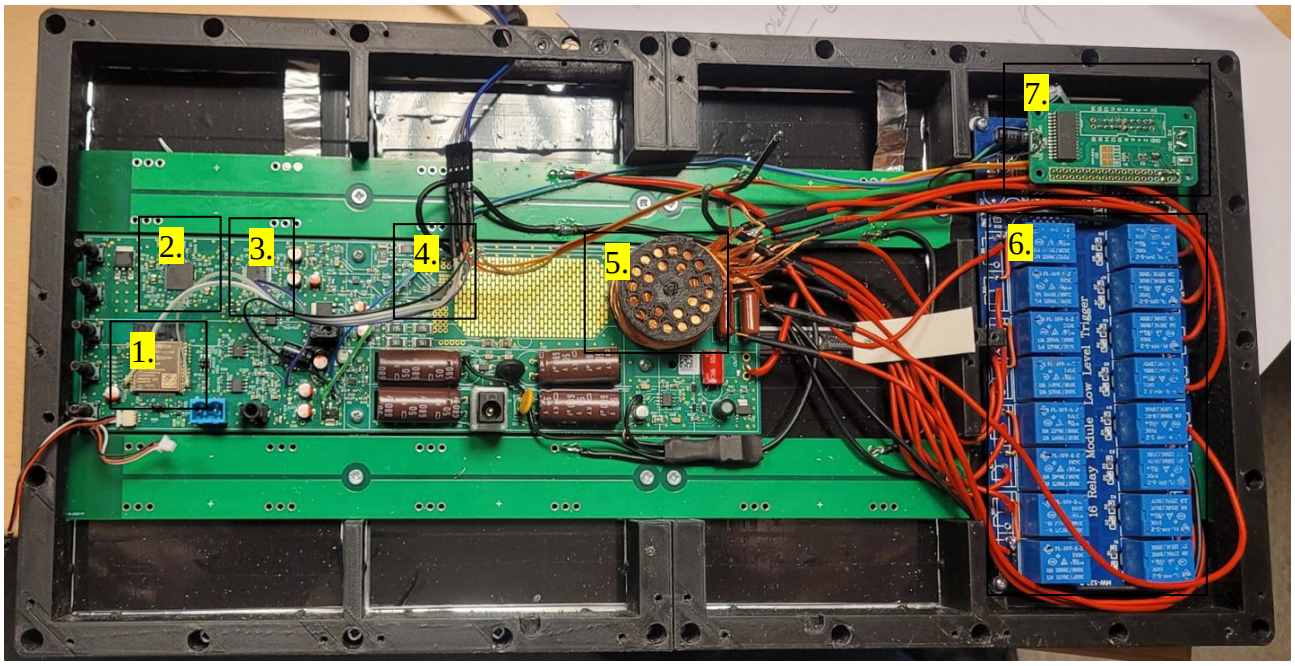
Pilnas programos kodas pateiktas 1 priede.

3. Tiriamoji dalis

Šioje dalyje apžvelgtas surinktas plačiajuostis ultragarso generatorius, jo matavimų eiga ir rezultatai.

3.1. Plačiajuostis ultragarsinis generatorius

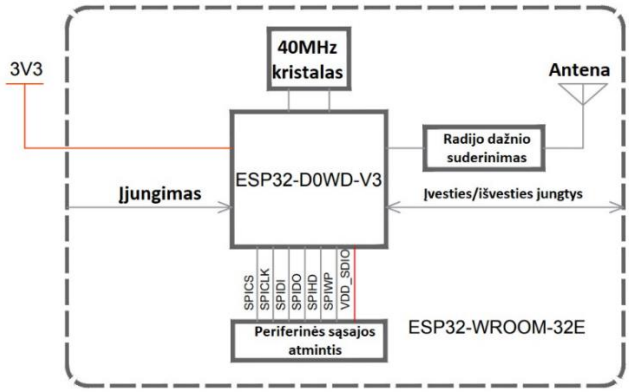
Ultragarsinis generatorius surinktas pasinaudojus „NEUROTECHNOLOGY“ valdymo plokšte ir relių moduliu. Ultragarsinis generatorius pateiktas 47 paveiksle.



47 pav. Plačiajuostis ultragarso generatorius

Valdymo plokštės pagrindiniai komponentai (žr. 47 pav., 1–5):

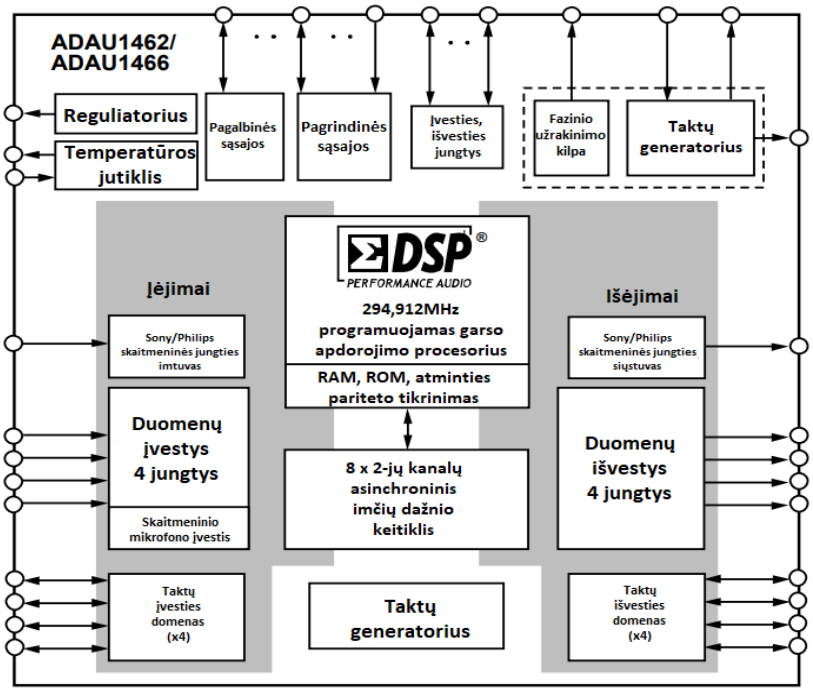
1. ESP32-WROOM-32E mikrovaldiklis, naudojantis ESP32-D0WD-V3 integruotą „Xtensa“ dviejų branduolių, 32 bitų LX6 mikroprocesorių, kurio taktinis dažnis 240 MHz. Šis valdiklis turi integruotą anteną, todėl jį galima sujungti su kitais įrenginiais *Wi-Fi* bei *Bluetooth* ryšiu. 48 paveiksle (žr. 48 pav.) pateikta ESP32-WROOM-32E mikrovaldiklio blokinė diagrama.



48 pav. ESP32-WROOM-32E mikrovaldiklio blokinė diagrama [43]

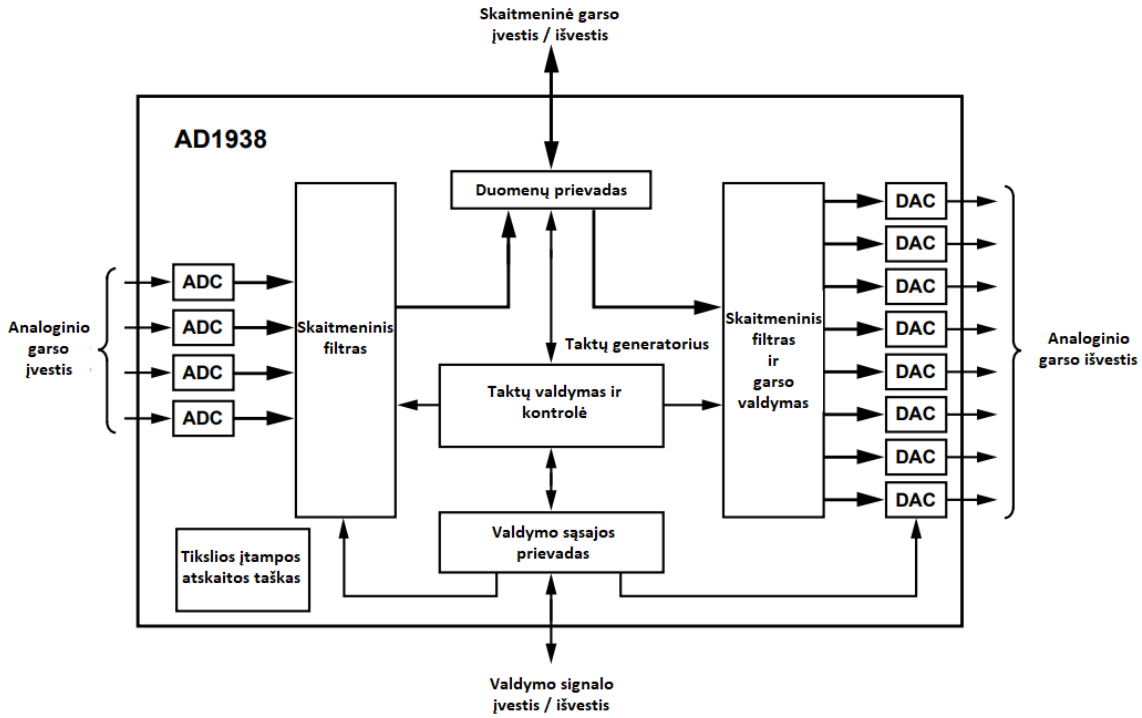
ESP32-WROOM-32E mikrovaldiklis maitinamas 3–3,6 V nuolatine įtampa ir turi 34 fizines bendrosios paskirties įvesties-išvesties jungtis.

2. „SigmaDSP“ ADAU1462 skaitmeninis garso procesorius, turintis 32 bitų branduolį ir gebantis dirbti 294,912 MHz dažniu. ADAU1462 skaitmeninio garso procesoriaus blokinė diagrama pateikta 49 paveiksle.



49 pav. ADAU1462 skaitmeninio garso procesoriaus blokinė diagrama [44]

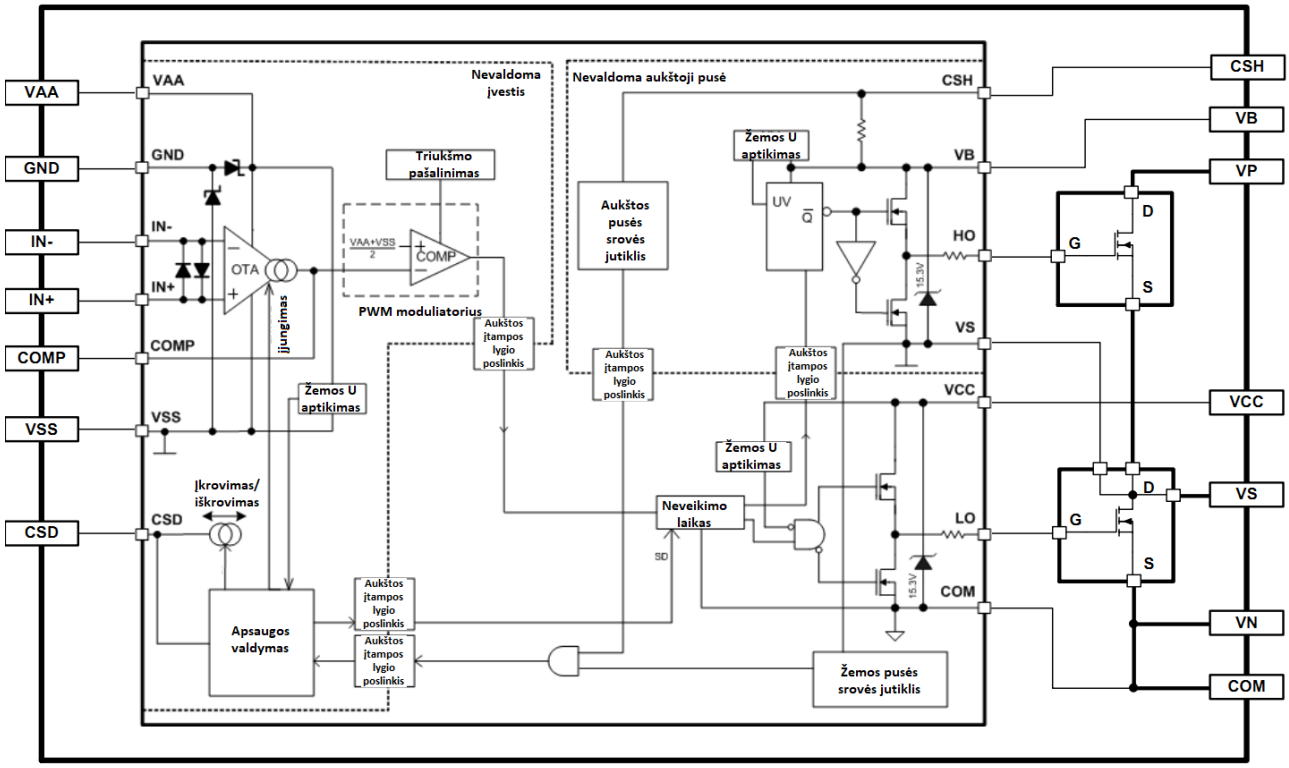
3. AD1938 skaitmeninis-analoginis / analoginis-skaitmeninis keitiklis, kuriuo blokinė diagrama pateikta 50 paveiksle.



50 pav. AD1938 blokinė diagrama [45]

Šis keitiklis palaiko 24 bitų, 8–192 kHz imčių dažnį. Šis keitiklis vienu metu gali priimti 4 analoginius signalus ir gražinti 8, nes jame integruoti 4 analoginiai-skaitmeniniai keitikliai ir 8 skaitmeniniai-analoginiai keitikliai.

4. IR4301M D klasės stiprintuvas, gebantis sustiprinti įėjimo signalą iki 160 W / 4 Ω. Šiam D klasės stiprintuvui nėra reikalingi radiatoriai, be to, jį galima maitinti vienu arba atskirais maitinimo šaltiniais. Jis atsparus išoriniams triukšmams, savyje turi integruotą įjungimo ir išjungimo triukšmo slopintuvą. IR4301M D klasės stiprintuvas turi apsaugas nuo per didelės srovės, nuo aukštos temperatūros, per mažos įtampos, bei turi savaiminio persikrovimo funkciją. IR4301M D klasės stiprintuvo blokinė diagrama pateikta 51 paveiksle.



51 pav. IR4301M D klasės stiprintuvo blokinė diagrama [46]

Diagramoje naudojami trumpiniai:

- VAA – nevaldomos įvesties teigiama maitinimo įtampa;
- GND – įžeminimas;
- IN- – analoginis invertuojantis įėjimas;
- IN+ – analoginis neinvertuojantis įėjimas;
- COMP – impulso pločio moduliacijos komparatoriaus įėjimas;
- VSS – nevaldomos įvesties neigiama maitinimo įtampa;
- CSD – išjungimo įvestis;
- CSH – aukštos pusės per didelės srovės jutimo įvestis;
- VB – aukštos pusės nevaldomas maitinimas;
- VP – teigiama maitinimo įtampa;
- VCC – žemos pusės maitinimas;
- VS – impulso pločio moduliacijos išėjimas;
- VN – neigiama maitinimo įtampa;
- COM – žemos pusės maitinimo grįžimas.

5. D klasės stiprintuvo išėjimo *LC* žemo dažnio filtro ritės. Viename korpuse sutalpinta 16 skirtingo induktyvumo ričių, kurių dydžiai pateikiami pirmoje ir antroje lentelėse. Ritės pagamintos iš lakuoto, varinio, daugiagyslio laido.

47 paveiksle 6 numeriu pažymėtas šešiolikos kanalų, žemo paleidimo lygio relių modulis, skirtas ritėms perjungti, generatoriaus dažniui kintant. Relių modulį sudaro:

- šešiolika FL-3FF-S-Z relių;
- šešiolika EL817C optronų;
- dvi ULN2803 aukštos įtampos ir srovės, 8 kanalų Darlingtono tranzistorių matricos.

Relių modulio veikimas:

1. Valdymo signalas paduodamas į vieną iš 16 įėjimo jungčių.
2. Gautu signalu įjungiamas atitinkamas optronas.
3. Optrono išėjimo signalas stiprinamas Darlingtono tranzistoriais.
4. Sustiprintu signalu įjungiama atitinkama relė.

47 paveiksle 7 numeriu pažymėtas IO PI ZERO, 16 skaitmeninių įėjimų-išėjimų plokštė, valdoma per I2C jungtį. Ši plokštė supaprastina relių modulio valdymą, nes reikalingas mažesnis kiekis laidų iš valdiklio.

Plačiajuosčiam ultragarso generatoriaus maitinimui panaudotas 72 W, 48 V nuolatinės srovės maitinimo šaltinis.

3.2. Plačiajuosčio ultragarso generatoriaus programa

Surinkto plačiajuosčio ultragarso generatoriaus veikimui užtikrinti panaudotos dvi programos: *Arduino* IDE, *SigmaStudio.* Kaip jau išsiaiškinta eksperimentinėje projekto dalyje, plačiajuostį ultragarso generatorių galima realizuoti keliais būdais, t. y. naudojant DSP kartu su ESP32 valdikliu ir skaitmeniniu-analoginiu keitikliu (žr. 28 pav.), arba supaprastinus ultragarso generatorių, naudoti tik ESP32 teikiamas galimybes (žr. 29 pav.), kurios, kaip parodė bandomasis programavimas, pakankamos ultragarsui generuoti (sugeneruotos viršijančios 90 kHz sinuso formos bangos).

Vienu ir kitu atvejais, naudojant surinktą plačiajuostį generatorių reikalingos abi programos, kadangi pirma, reikia sukurti signalų generatorių DSP procesoriuje (žr. 30 pav.) ir jį kartu su kitomis valdymo komandomis įrašyti į ESP32 valdiklį. Antra, DSP procesoriaus įėjimą sujungti su išėjimu (žr. 31 pav.) ir gautą programą įkelti į ESP32 valdiklį.

Norint sukurti *SigmaStudio* aplinkoje parašyti programą DSP, visų pirma reikia nustatyti procesoriaus registrų adresus. Šiam tikslui naudojamas „sigmaDSP“ programatorius EVAL-ADUSB2Z (žr. 52 pav.).

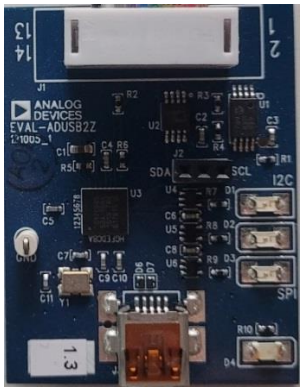
Fizinės įrangos parametruose nustatomi atminties ir DSP ryšiai (žr. 53 pav.).

Sukūrus grafinę programą (žr. 30, 31 pav.) ir sukonfigūravus ADAU1462, programos kodas kompiliuojamas F7 mygtuku ir eksportuojamas (žr. 54 pav.).

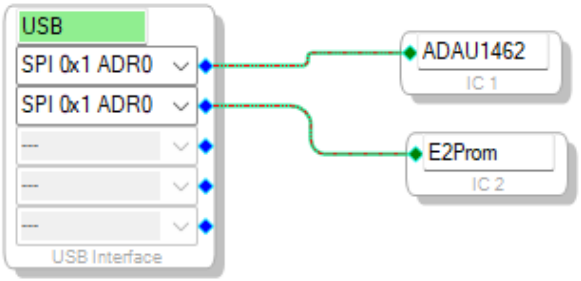
Iš iškeltų programos failų (žr. 55 pav.), reikalingi du:

- „Projekto_pavadinimas_IC_1.h“;
- „Projekto_pavadinimas_IC_1_REG.h“.

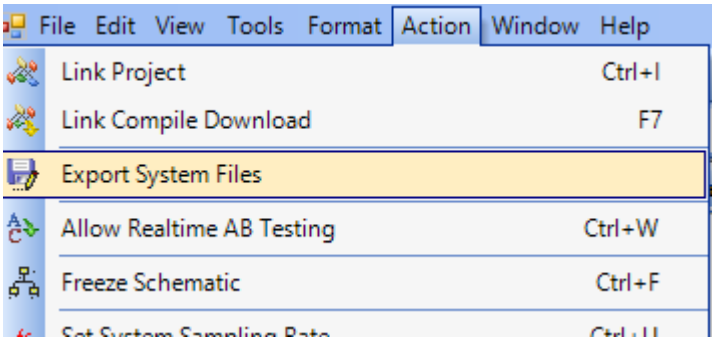
Šie failai reikalingi ESP32 su DSP sujungti, nes juose surašyti visų DSP registrų adresai ir jų paskirtis.



52 pav. EVAL-ADUSB2Z programatorius



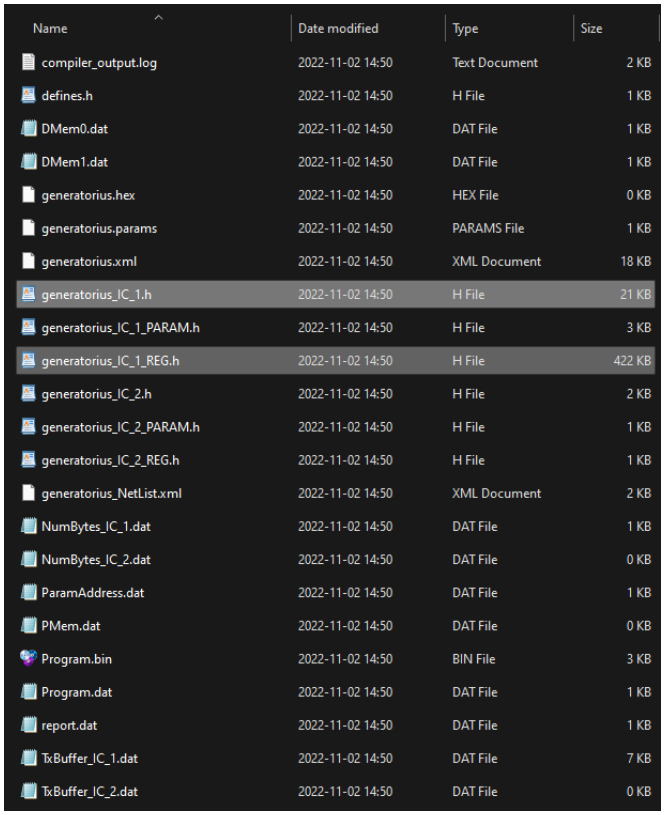
53 pav. ESP32 ir ADAU1462 ryšio sudarymas



54 pav. *SigmaStudio* programos kodo eksportavimas

Sekančiame žingsnyje *Arduino* IDE programoje aprašomi iš *SigmaStudio* programinio paketo gautas failas „generatorius_IC_1.h“, ESP32 sinusinio signalo generatoriaus programos kodas – „generatorius.h“ ir kitos generatoriaus valdymo plokštei veikti reikalingos bibliotekos, tokios kaip:

- EEPROM.h – skirta veiksmams su EEPROM atmintimi atlikti (įrašyti, ištrinti, perkelti ir t. t.);
- math.h – skirta matematiniams veiksmams su skaičiais atlikti;
- SPI.h – skirta duomenų mainams tarp įrenginių mažais atstumais atlikti;
- Timer.h – skirta paleidimo-stabdymo laikmačiams kurti;
- Adafruit_NeoPixel.h – skirta LED juostos pikseliams valdyti;
- BluetoothA2DPSink.h – skirta *Bluetooth* komunikacijai vykdyti;
- Button.h – skirta fizikiniams mygtukams aprašyti.



Name	Date modified	Type	Size
compiler_output.log	2022-11-02 14:50	Text Document	2 KB
defines.h	2022-11-02 14:50	H File	1 KB
DMem0.dat	2022-11-02 14:50	DAT File	1 KB
DMem1.dat	2022-11-02 14:50	DAT File	1 KB
generatorius.hex	2022-11-02 14:50	HEX File	0 KB
generatorius.params	2022-11-02 14:50	PARAMS File	1 KB
generatorius.xml	2022-11-02 14:50	XML Document	18 KB
generatorius_IC_1.h	2022-11-02 14:50	H File	21 KB
generatorius_IC_1_PARAM.h	2022-11-02 14:50	H File	3 KB
generatorius_IC_1_REG.h	2022-11-02 14:50	H File	422 KB
generatorius_IC_2.h	2022-11-02 14:50	H File	2 KB
generatorius_IC_2_PARAM.h	2022-11-02 14:50	H File	1 KB
generatorius_IC_2_REG.h	2022-11-02 14:50	H File	1 KB
generatorius_NetList.xml	2022-11-02 14:50	XML Document	2 KB
NumBytes_IC_1.dat	2022-11-02 14:50	DAT File	1 KB
NumBytes_IC_2.dat	2022-11-02 14:50	DAT File	0 KB
ParamAddress.dat	2022-11-02 14:50	DAT File	1 KB
PMem.dat	2022-11-02 14:50	DAT File	0 KB
Program.bin	2022-11-02 14:50	BIN File	3 KB
Program.dat	2022-11-02 14:50	DAT File	1 KB
report.dat	2022-11-02 14:50	DAT File	1 KB
TxBuffer_IC_1.dat	2022-11-02 14:50	DAT File	7 KB
TxBuffer_IC_2.dat	2022-11-02 14:50	DAT File	0 KB

55 pav. *SigmaStudio* programos aplinkoje eksportuoti failai

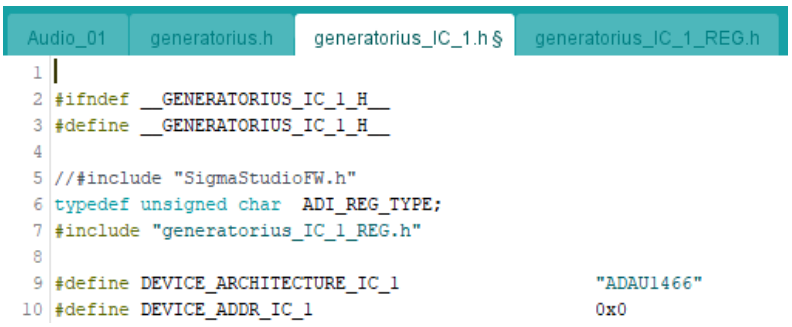
Programos kodo dalis aprašanti bibliotekas pateikta 56 paveiksle.

Audio_01 | generatorius.h | generatorius_IC_1.h § | generatorius_IC_1_REG.h

```
#include <EEPROM.h>
#define EEPROM_SIZE 5  //0-ServiceModeNr, 1-VolVal, 2-LedOnOff, 3-Au
#include <math.h>
#include <SPI.h>
#include "Timer.h"
#include "generatorius_IC_1.h"
#include "generatorius.h"
//LED strip
#include <Adafruit_NeoPixel.h>
#define PIN       27
#define NUMPIXELS 10
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
#define DELAYVAL 500
int LEDspalva=1;
int LEDilgis=0;
boolean LEDuzp = true;
int LedOnOff = 2;

#include "BluetoothA2DPSink.h"
BluetoothA2DPSink a2dp_sink;
bool play_activated=1;

//Mygtukai be pertraukimu
#include <Button.h>
```

56 pav. *Arduino* IDE programos kodas, aprašantis naudojamas bibliotekas

„Generatorius_IC_1.h“ faile užkomentuojama „SigmaStudioFW.h“ biblioteka, tam, kad programa jos nenaudotų. Į programos kodą įtraukiama papildoma „ADI_REG_TYPE“ biblioteka, atsakinga už registrų naudojimą (žr. 57 pav.).



57 pav. *Arduino* IDE bibliotekos užkomentavimas ir kitos įtraukimas

Iš „generatorius_IC_1.h“ failo iškerpama programos kodo dalis (žr. 58 ir 59 pav.) ir įklijuojama į pagrindinį programos kodą. Ši kodo dalis apibrėžia už pagrindinius DSP parametrus, tokius kaip įjungimas, išjungimas, perkrovimas ir kitos funkcijos reikalingos sklandžiam DSP darbui užtikrinti.

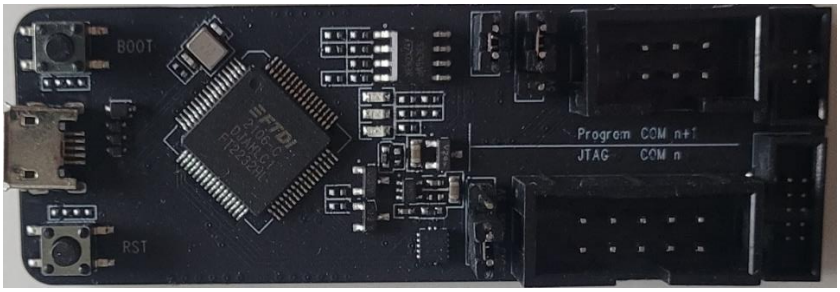
```
void default_download_IC_1() {
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOFT_RESET_IC_1_ADDR, REG_SOFT_RESET_IC_1_BYTE, R0_SOFT_RESET_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOFT_RESET_IC_1_ADDR, REG_SOFT_RESET_IC_1_BYTE, R1_SOFT_RESET_IC_1_Default );
  SIGMA_WRITE_DELAY( DEVICE_ADDR_IC_1, R2_RESET_DELAY_IC_1_SIZE, R2_RESET_DELAY_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_HIBERNATE_IC_1_ADDR, REG_HIBERNATE_IC_1_BYTE, R3_HIBERNATE_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_HIBERNATE_IC_1_ADDR, REG_HIBERNATE_IC_1_BYTE, R4_HIBERNATE_IC_1_Default );
  SIGMA_WRITE_DELAY( DEVICE_ADDR_IC_1, R5_HIBERNATE_DELAY_IC_1_SIZE, R5_HIBERNATE_DELAY_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_KILL_CORE_IC_1_ADDR, REG_KILL_CORE_IC_1_BYTE, R6_KILL_CORE_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_KILL_CORE_IC_1_ADDR, REG_KILL_CORE_IC_1_BYTE, R7_KILL_CORE_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_PLL_ENABLE_IC_1_ADDR, REG_PLL_ENABLE_IC_1_BYTE, R8_PLL_ENABLE_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_PLL_CTRL1_IC_1_ADDR, REG_PLL_CTRL1_IC_1_BYTE, R9_PLL_CTRL1_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_PLL_CLK_SRC_IC_1_ADDR, REG_PLL_CLK_SRC_IC_1_BYTE, R10_PLL_CLK_SRC_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_MCLK_OUT_IC_1_ADDR, REG_MCLK_OUT_IC_1_BYTE, R11_MCLK_OUT_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_PLL_ENABLE_IC_1_ADDR, REG_PLL_ENABLE_IC_1_BYTE, R12_PLL_ENABLE_IC_1_Default );
  SIGMA_WRITE_DELAY( DEVICE_ADDR_IC_1, R13_PLL_LOCK_DELAY_IC_1_SIZE, R13_PLL_LOCK_DELAY_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_POWER_ENABLE0_IC_1_ADDR, REG_POWER_ENABLE0_IC_1_BYTE, R14_POWER_ENABLE0_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_POWER_ENABLE1_IC_1_ADDR, REG_POWER_ENABLE1_IC_1_BYTE, R15_POWER_ENABLE1_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE0_IC_1_ADDR, REG_SOUT_SOURCE0_IC_1_BYTE, R16_SOUT_SOURCE0_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE1_IC_1_ADDR, REG_SOUT_SOURCE1_IC_1_BYTE, R17_SOUT_SOURCE1_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE2_IC_1_ADDR, REG_SOUT_SOURCE2_IC_1_BYTE, R18_SOUT_SOURCE2_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE3_IC_1_ADDR, REG_SOUT_SOURCE3_IC_1_BYTE, R19_SOUT_SOURCE3_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE4_IC_1_ADDR, REG_SOUT_SOURCE4_IC_1_BYTE, R20_SOUT_SOURCE4_IC_1_Default );
```

58 pav. *Arduino* IDE kodo dalis DSP veikimui užtikrinti

```
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE5_IC_1_ADDR, REG_SOUT_SOURCE5_IC_1_BYTE, R21_SOUT_SOURCE5_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE6_IC_1_ADDR, REG_SOUT_SOURCE6_IC_1_BYTE, R22_SOUT_SOURCE6_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE7_IC_1_ADDR, REG_SOUT_SOURCE7_IC_1_BYTE, R23_SOUT_SOURCE7_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE8_IC_1_ADDR, REG_SOUT_SOURCE8_IC_1_BYTE, R24_SOUT_SOURCE8_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE9_IC_1_ADDR, REG_SOUT_SOURCE9_IC_1_BYTE, R25_SOUT_SOURCE9_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE10_IC_1_ADDR, REG_SOUT_SOURCE10_IC_1_BYTE, R26_SOUT_SOURCE10_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE11_IC_1_ADDR, REG_SOUT_SOURCE11_IC_1_BYTE, R27_SOUT_SOURCE11_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE12_IC_1_ADDR, REG_SOUT_SOURCE12_IC_1_BYTE, R28_SOUT_SOURCE12_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE13_IC_1_ADDR, REG_SOUT_SOURCE13_IC_1_BYTE, R29_SOUT_SOURCE13_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE14_IC_1_ADDR, REG_SOUT_SOURCE14_IC_1_BYTE, R30_SOUT_SOURCE14_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE15_IC_1_ADDR, REG_SOUT_SOURCE15_IC_1_BYTE, R31_SOUT_SOURCE15_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE16_IC_1_ADDR, REG_SOUT_SOURCE16_IC_1_BYTE, R32_SOUT_SOURCE16_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE17_IC_1_ADDR, REG_SOUT_SOURCE17_IC_1_BYTE, R33_SOUT_SOURCE17_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE18_IC_1_ADDR, REG_SOUT_SOURCE18_IC_1_BYTE, R34_SOUT_SOURCE18_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE19_IC_1_ADDR, REG_SOUT_SOURCE19_IC_1_BYTE, R35_SOUT_SOURCE19_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE20_IC_1_ADDR, REG_SOUT_SOURCE20_IC_1_BYTE, R36_SOUT_SOURCE20_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE21_IC_1_ADDR, REG_SOUT_SOURCE21_IC_1_BYTE, R37_SOUT_SOURCE21_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE22_IC_1_ADDR, REG_SOUT_SOURCE22_IC_1_BYTE, R38_SOUT_SOURCE22_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SOUT_SOURCE23_IC_1_ADDR, REG_SOUT_SOURCE23_IC_1_BYTE, R39_SOUT_SOURCE23_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SERIAL_BYTE_0_0_IC_1_ADDR, REG_SERIAL_BYTE_0_0_IC_1_BYTE, R40_SERIAL_BYTE_0_0_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SERIAL_BYTE_1_0_IC_1_ADDR, REG_SERIAL_BYTE_1_0_IC_1_BYTE, R41_SERIAL_BYTE_1_0_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SERIAL_BYTE_2_0_IC_1_ADDR, REG_SERIAL_BYTE_2_0_IC_1_BYTE, R42_SERIAL_BYTE_2_0_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SERIAL_BYTE_3_0_IC_1_ADDR, REG_SERIAL_BYTE_3_0_IC_1_BYTE, R43_SERIAL_BYTE_3_0_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SERIAL_BYTE_4_0_IC_1_ADDR, REG_SERIAL_BYTE_4_0_IC_1_BYTE, R44_SERIAL_BYTE_4_0_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SERIAL_BYTE_5_0_IC_1_ADDR, REG_SERIAL_BYTE_5_0_IC_1_BYTE, R45_SERIAL_BYTE_5_0_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SERIAL_BYTE_6_0_IC_1_ADDR, REG_SERIAL_BYTE_6_0_IC_1_BYTE, R46_SERIAL_BYTE_6_0_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_SERIAL_BYTE_7_0_IC_1_ADDR, REG_SERIAL_BYTE_7_0_IC_1_BYTE, R47_SERIAL_BYTE_7_0_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, PROGRAM_ADDR_IC_1, PROGRAM_SIZE_IC_1, Program_Data_IC_1 );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, PARAM_ADDR_IC_1, PARAM_SIZE_IC_1, Param_Data_IC_1 );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, DM1_DATA_ADDR_IC_1, DM1_DATA_SIZE_IC_1, DM1_DATA_Data_IC_1 );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_KILL_CORE_IC_1_ADDR, REG_KILL_CORE_IC_1_BYTE, R51_KILL_CORE_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_START_ADDRESS_IC_1_ADDR, REG_START_ADDRESS_IC_1_BYTE, R52_START_ADDRESS_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_START_PULSE_IC_1_ADDR, REG_START_PULSE_IC_1_BYTE, R53_START_PULSE_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_START_CORE_IC_1_ADDR, REG_START_CORE_IC_1_BYTE, R54_START_CORE_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_START_CORE_IC_1_ADDR, REG_START_CORE_IC_1_BYTE, R55_START_CORE_IC_1_Default );
  SIGMA_WRITE_DELAY( DEVICE_ADDR_IC_1, R56_START_DELAY_IC_1_SIZE, R56_START_DELAY_IC_1_Default );
  SIGMA_WRITE_REGISTER_BLOCK( DEVICE_ADDR_IC_1, REG_HIBERNATE_IC_1_ADDR, REG_HIBERNATE_IC_1_BYTE, R57_HIBERNATE_IC_1_Default );
}
```

59 pav. *Arduino* IDE kodo dalis DSP veikimui užtikrinti. Tęsinys

Sukompiliuotas kodas į ESP32 valdiklį įkeliamas pasinaudojus „ESP-prog“ programatoriumi (žr. 60 pav.), kuris prie ESP32 valdiklio prijungiamas per ant valdymo plokštės specialiai išvestas jungtis.

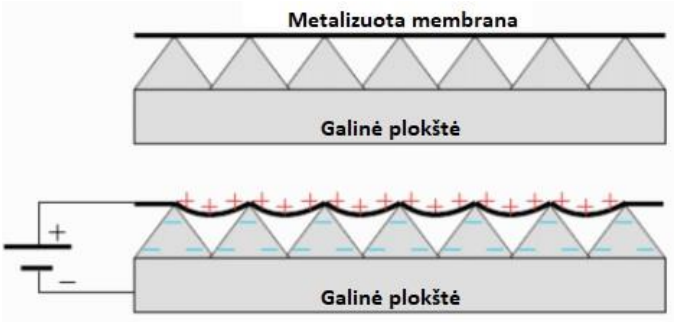


60 pav. ESP-prog programatorius

Sėkmingai įkėlus programinę įrangą į plačiajuosčio ultragarso generatoriaus valdiklį, galima atlikti jo bandymus.

3.3. Bandymų įranga

Surinkto ultragarsinio generatoriaus bandymams atlikti pasirinktas elektrostatinis ultragarso keitiklis, kuris kitaip, nei pjezoelektrinis keitiklis gali dirbti plačioje dažnių juostoje. Ultragarsinis elektrostatinis keitiklis sudarytas iš lanksčios laidžios membranos, kuri sumontuota ant laidžios galinės plokštės, sudarytoje iš griovelių. Kai tarp membranos ir galinės plokštės atsiranda elektrinis signalas, membrana juda link galinės plokštės arba tolyn nuo jos [47]. Keitiklio veikimo principas pavaizduotas 61 paveiksle.



61 pav. Keitiklio veikimo principas [47]

Naudotas ultragarso elektrostatinis keitiklis pavaizduotas 62 paveiksle.

62 pav. Elektrostatinis keitiklis

Generuojamos bangos matuojamos „miniDSP“ UMIK-1 mikrofonu (žr. 63 pav.). Jis maitinamas per USB jungtį, savyje turi integruotą 24 bitų analoginį-skaitmeninį keitiklį, todėl naudojimui nėra reikalinga papildoma įranga, tokia kaip garso plokštės ar stiprintuvai. Kiekvienas „miniDSP“ UMIK-1 mikrofonas turi savo kalibravimo failą, užtikrinantį vienodą dažnio atsaką nuo 20 Hz iki 20 kHz.

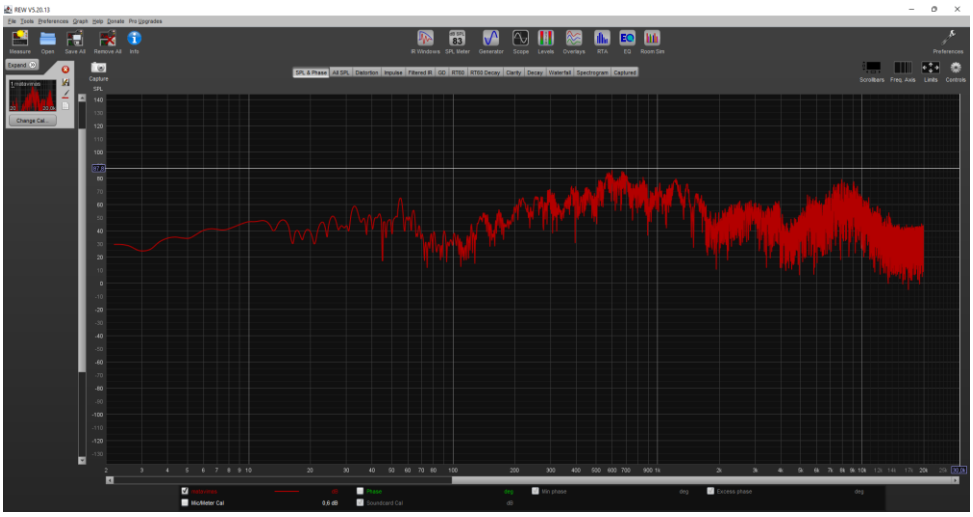
63 pav. „miniDSP“ UMIK-1 mikrofonas

Mikrofonas naudojamas kartu su *REW* programine įranga, skirta akustiniams, garsiakalbių ir garso įrenginių matavimams. *REW* garso matavimo ir analizės funkcijos apima:

- bandymo signalų generavimo įrankius;
- SPL ir varžos matavimus;
- dažnio ir impulsų atsakų matavimus;
- iškraipymo matavimus;
- fazių, grupių vėlavimo ir spektrinio nykimo diagramų, krioklių, spektrogramų ir energijos laiko kreivių generavimus;
- realaus laiko analizavimo diagramų generavimus;
- aidėjimo laiko skaičiavimus;
- modalinių rezonansų dažnių ir skilimo laikų nustatymus.

Sukalibravus „miniDSP“ UMIK-1 mikrofoną *REW* programinėje įrangoje (dėl kalibravimo sumažėja matavimų nuokrypiai), galima atlikti matavimus.

64 paveiksle pateikiamas *REW* programos matavimų langas.



64 pav. *REW* programos matavimų langas

64 paveiksle pateiktame *REW* programos matavimų lange matomas SPL matavimų pavyzdys.

3.4. Matavimai

Surinkto ultragarsinio generatoriaus ADCH matavimai atlikti, naudojant „miniDSP“ UMIK-1 mikrofoną, esant 20 kHz dažniui. UMIK-1 mikrofonu kartu su *REW* programine įranga galima išmatuoti akustinio signalo parametrus iki 20 kHz, likusiems dažniams garso slėgio lygio (trump. SPL) reikšmės turi būti perskaičiuotos pagal išmatuotą įtampą. Bandymai atlikti tarp keitiklių, palaikant pastovų vieno metro tarpą. Pirmo bandymo rezultatai pateikiami 3 lentelėje. Matavimai atlikti keičiant dažnį kas 500 Hz.

3 lentelė. Pirmo bandymo rezultatai

***f*, kHz**	***U*, V**	***f*, kHz**	***U*, V**	***f*, kHz**	***U*, V**	***f*, kHz**	***U*, V**	***f*, kHz**	***U*, V**	***f*, kHz**	***U*, V**	***f*, kHz**	***U*, V**
20	135	**29,5**	186	**39**	192	**48,5**	210	**58**	141	**67,5**	145	**77**	167
20,5	178	**30**	227	**39,5**	156	**49**	187	**58,5**	129	**68**	138	**77,5**	148
21	202	**30,5**	240	**40**	106	**49,5**	162	**59**	121	**68,5**	132	**78**	136
21,5	174	**31**	208	**40,5**	122	**50**	124	**59,5**	111	**69**	122	**78,5**	122
22	140	**31,5**	168	**41**	140	**50,5**	140	**60**	131	**69,5**	122	**79**	111
22,5	183	**32**	128	**41,5**	162	**51**	153	**60,5**	132	**70**	114	**79,5**	100
23	211	**32,5**	156	**42**	190	**51,5**	167	**61**	132	**70,5**	167	**80**	226
23,5	190	**33**	192	**42,5**	218	**52**	179	**61,5**	134	**71**	167	**80,5**	223
24	146	**33,5**	233	**43**	235	**52,5**	187	**62**	131	**71,5**	165	**81**	211
24,5	187	**34**	248	**43,5**	233	**53**	187	**62,5**	130	**72**	147	**81,5**	195
25	220	**34,5**	222	**44**	210	**53,5**	180	**63**	126	**72,5**	137	**82**	175
25,5	204	**35**	103	**44,5**	177	**54**	167	**63,5**	122	**73**	124	**82,5**	157
26	119	**35,5**	119	**45**	122	**54,5**	148	**64**	117	**73,5**	114	**83**	140
26,5	149	**36**	140	**45,5**	137	**55**	146	**64,5**	112	**74**	105	**83,5**	125
27	188	**36,5**	168	**46**	157	**55,5**	151	**65**	139	**74,5**	94	**84**	112
27,5	225	**37**	204	**46,5**	180	**56**	156	**65,5**	145	**75**	202	**84,5**	101
28	220	**37,5**	237	**47**	200	**56,5**	155	**66**	149	**75,5**	202	**85**	92
28,5	181	**38**	250	**47,5**	216	**57**	153	**66,5**	151	**76**	194		
29	149	**38,5**	228	**48**	220	**57,5**	148	**67**	148	**76,5**	182		

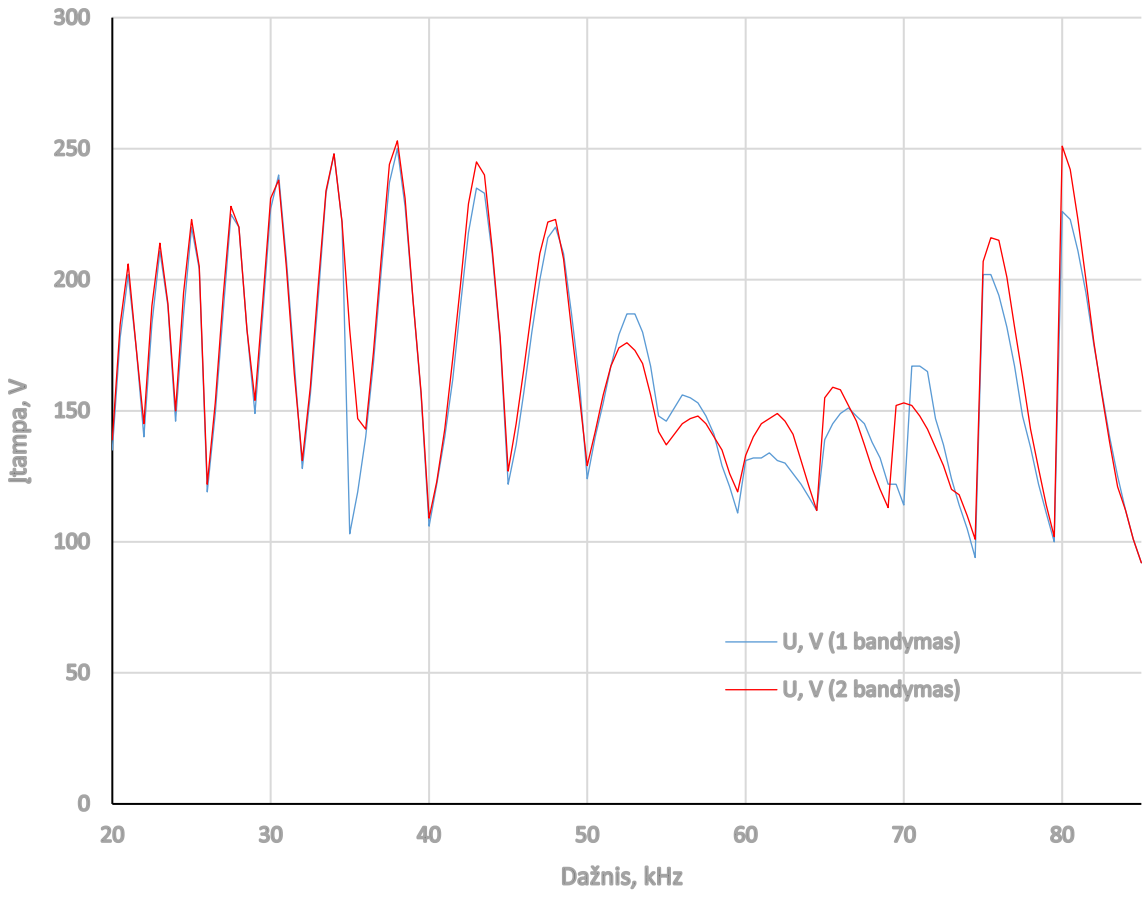
Kitas bandymas atliktas praėjus 24 valandoms nuo pirmojo bandymo. Antrojo bandymo rezultatai pateikti 4 lentelėje.

4 lentelė. Antrojo bandymo rezultatai

***f*, kHz**	***U*, V**	***f*, kHz**	***U*, V**	***f*, kHz**	***U*, V**	***f*, kHz**	***U*, V**	***f*, kHz**	***U*, V**	***f*, kHz**	***U*, V**	***f*, kHz**	***U*, V**
20	139	**29,5**	191	**39**	192	**48,5**	208	**58**	140	**67,5**	137	**77**	182
20,5	183	**30**	231	**39,5**	157	**49**	181	**58,5**	135	**68**	128	**77,5**	163
21	206	**30,5**	238	**40**	109	**49,5**	156	**59**	126	**68,5**	120	**78**	143
21,5	174	**31**	205	**40,5**	123	**50**	129	**59,5**	119	**69**	113	**78,5**	128
22	145	**31,5**	164	**41**	143	**50,5**	142	**60**	133	**69,5**	152	**79**	114
22,5	190	**32**	131	**41,5**	170	**51**	156	**60,5**	140	**70**	153	**79,5**	102
23	214	**32,5**	159	**42**	198	**51,5**	167	**61**	145	**70,5**	152	**80**	251
23,5	191	**33**	197	**42,5**	229	**52**	174	**61,5**	147	**71**	148	**80,5**	242
24	150	**33,5**	234	**43**	245	**52,5**	176	**62**	149	**71,5**	143	**81**	223
24,5	195	**34**	248	**43,5**	240	**53**	173	**62,5**	146	**72**	136	**81,5**	200
25	223	**34,5**	222	**44**	212	**53,5**	168	**63**	141	**72,5**	129	**82**	176
25,5	205	**35**	181	**44,5**	178	**54**	156	**63,5**	131	**73**	120	**82,5**	156

f, kHz	*U*, V	*f*, kHz	*U*, V	*f*, kHz	*U*, V	*f*, kHz	*U*, V	*f*, kHz	*U*, V	*f*, kHz	*U*, V	*f*, kHz	*U*, V
26	122	**35,5**	147	**45**	127	**54,5**	142	**64**	121	**73,5**	118	**83**	138
26,5	153	**36**	143	**45,5**	145	**55**	137	**64,5**	112	**74**	110	**83,5**	121
27	194	**36,5**	172	**46**	166	**55,5**	141	**65**	155	**74,5**	101	**84**	112
27,5	228	**37**	209	**46,5**	189	**56**	145	**65,5**	159	**75**	207	**84,5**	101
28	220	**37,5**	244	**47**	210	**56,5**	147	**66**	158	**75,5**	216	**85**	92
28,5	181	**38**	253	**47,5**	222	**57**	148	**66,5**	152	**76**	215		
29	154	**38,5**	231	**48**	223	**57,5**	145	**67**	146	**76,5**	201		

Pirmo ir antro bandymų rezultatų grafinis atvaizdavimas pateiktas 65 paveiksle.



65 pav. Plačiajuosčio ultragarsinio generatoriaus ADCH – išėjimo įtampos matavimų rezultatai

UMIK-1 mikrofonu kartu su *REW* programine įranga išmatavus akustinio signalo parametrus, esant 20 kHz, gautas 108 dB garso slėgio lygis. Pasinaudojus šiuo matmeniu ir 3.1 formule, pateiktą UMIK-1 mikrofono gamintojo, galima apskaičiuoti atraminę įtampos reikšmę, reikalingą SPL reikšmėms visame dažnių diapazone apskaičiuoti.

$$SPL = 20log_{10}\left(\frac{U}{U_{ref}}\right); \quad (3.1)$$

čia SPL – garso slėgio lygis, dB;
U – išmatuotas įtampa, V;
U_{ref} – atraminė įtampos reikšmė, V.

Pertvarkius (3.1) formulę, gaunama (3.2) formulė, kuri skirta atraminei įtampos reikšmei apskaičiuoti.

$$U_{ref} = \frac{U}{10^{\frac{SPL}{20}}}. \quad (3.2)$$

Įsistačius turimas reikšmes į (3.2) formulę nustatyta, kad atraminė įtampa lygi $5{,}37 \cdot 10^{-4}$ V.

Pasinaudojus (3.1) formule pirmojo ir antrojo bandymų įtampos perskaičiuojamos į garso slėgio lygį. Perskaičiuotos reikšmės pateiktos 5 ir 6 lentelėse.

5 lentelė. Pirmo bandymo perskaičiuotos reikšmės

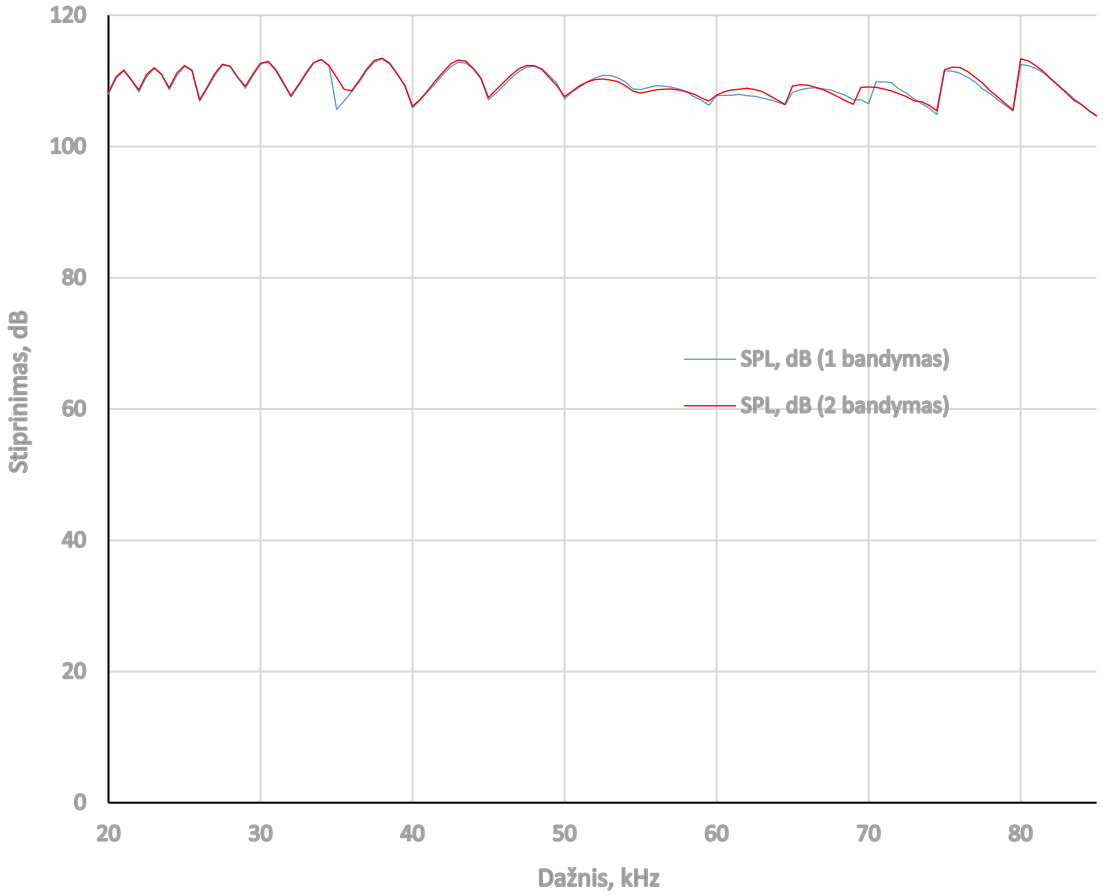
***f*, kHz**	***SPL*, dB**	***f*, kHz**	***SPL*, dB**	***f*, kHz**	***SPL*, dB**	***f*, kHz**	***SPL*, dB**	***f*, kHz**	***SPL*, dB**	***f*, kHz**	***SPL*, dB**	***f*, kHz**	***SPL*, dB**
20	108	**29,5**	110,78	**39**	111,06	**48,5**	111,84	**58**	108,38	**67,5**	108,62	**77**	109,85
20,5	110,40	**30**	112,51	**39,5**	109,26	**49**	110,83	**58,5**	107,61	**68**	108,19	**77,5**	108,80
21	111,50	**30,5**	113,00	**40**	105,90	**49,5**	109,58	**59**	107,05	**68,5**	107,80	**78**	108,06
21,5	110,20	**31**	111,75	**40,5**	107,12	**50**	107,26	**59,5**	106,30	**69**	107,12	**78,5**	107,12
22	108,32	**31,5**	109,90	**41**	108,32	**50,5**	108,32	**60**	107,74	**69,5**	107,12	**79**	106,30
22,5	110,64	**32**	107,54	**41,5**	109,58	**51**	109,09	**60,5**	107,80	**70**	106,53	**79,5**	105,39
23	111,88	**32,5**	109,26	**42**	110,97	**51,5**	109,85	**61**	107,80	**70,5**	109,85	**80**	112,48
23,5	110,97	**33**	111,06	**42,5**	112,16	**52**	110,45	**61,5**	107,94	**71**	109,85	**80,5**	112,36
24	108,68	**33,5**	112,74	**43**	112,81	**52,5**	110,83	**62**	107,74	**71,5**	109,74	**81**	111,88
24,5	110,83	**34**	113,28	**43,5**	112,74	**53**	110,83	**62,5**	107,67	**72**	108,74	**81,5**	111,19
25	112,24	**34,5**	112,32	**44**	111,84	**53,5**	110,50	**63**	107,40	**72,5**	108,13	**82**	110,25
25,5	111,59	**35**	105,65	**44,5**	110,35	**54**	109,85	**63,5**	107,12	**73**	107,26	**82,5**	109,31
26	106,90	**35,5**	106,90	**45**	107,12	**54,5**	108,80	**64**	106,76	**73,5**	106,53	**83**	108,32
26,5	108,86	**36**	108,32	**45,5**	108,13	**55**	108,68	**64,5**	106,38	**74**	105,82	**83,5**	107,33
27	110,88	**36,5**	109,90	**46**	109,31	**55,5**	108,97	**65**	108,25	**74,5**	104,86	**84**	106,38
27,5	112,44	**37**	111,59	**46,5**	110,50	**56**	109,26	**65,5**	108,62	**75**	111,50	**84,5**	105,48
28	112,24	**37,5**	112,89	**47**	111,41	**56,5**	109,20	**66**	108,86	**75,5**	111,50	**85**	104,67
28,5	110,55	**38**	113,35	**47,5**	112,08	**57**	109,09	**66,5**	108,97	**76**	111,15		
29	108,86	**38,5**	112,55	**48**	112,24	**57,5**	108,80	**67**	108,80	**76,5**	110,59		

6 lentelė. Antro bandymo perskaičiuotos reikšmės

***f*, kHz**	**SPL, dB**	***f*, kHz**	**SPL, dB**	***f*, kHz**	**SPL, dB**	***f*, kHz**	**SPL, dB**	***f*, kHz**	**SPL, dB**	***f*, kHz**	**SPL, dB**	***f*, kHz**	**SPL, dB**
20	108,25	**29,5**	111,01	**39**	111,06	**48,5**	111,75	**58**	108,32	**67,5**	108,13	**77**	110,59
20,5	110,64	**30**	112,67	**39,5**	109,31	**49**	110,55	**58,5**	108,00	**68**	107,54	**77,5**	109,64
21	111,67	**30,5**	112,92	**40**	106,14	**49,5**	109,26	**59**	107,40	**68,5**	106,98	**78**	108,50
21,5	110,20	**31**	111,63	**40,5**	107,19	**50**	107,61	**59,5**	106,90	**69**	106,45	**78,5**	107,54
22	108,62	**31,5**	109,69	**41**	108,50	**50,5**	108,44	**60**	107,87	**69,5**	109,03	**79**	106,53
22,5	110,97	**32**	107,74	**41,5**	110,00	**51**	109,26	**60,5**	108,32	**70**	109,09	**79,5**	105,57
23	112,00	**32,5**	109,42	**42**	111,33	**51,5**	109,85	**61**	108,62	**70,5**	109,03	**80**	113,39
23,5	111,01	**33**	111,28	**42,5**	112,59	**52**	110,20	**61,5**	108,74	**71**	108,80	**80,5**	113,07
24	108,92	**33,5**	112,78	**43**	113,18	**52,5**	110,30	**62**	108,86	**71,5**	108,50	**81**	112,36
24,5	111,19	**34**	113,28	**43,5**	113,00	**53**	110,15	**62,5**	108,68	**72**	108,06	**81,5**	111,41
25	112,36	**34,5**	112,32	**44**	111,92	**53,5**	109,90	**63**	108,38	**72,5**	107,61	**82**	110,30
25,5	111,63	**35**	110,55	**44,5**	110,40	**54**	109,26	**63,5**	107,74	**73**	106,98	**82,5**	109,26
26	107,12	**35,5**	108,74	**45**	107,47	**54,5**	108,44	**64**	107,05	**73,5**	106,83	**83**	108,19

f, kHz	SPL, dB	*f*, kHz	SPL, dB	*f*, kHz	SPL, dB	*f*, kHz	SPL, dB	*f*, kHz	SPL, dB	*f*, kHz	SPL, dB	*f*, kHz	SPL, dB
26,5	109,09	**36**	108,50	**45,5**	108,62	**55**	108,13	**64,5**	106,38	**74**	106,22	**83,5**	107,05
27	111,15	**36,5**	110,10	**46**	109,80	**55,5**	108,38	**65**	109,20	**74,5**	105,48	**84**	106,38
27,5	112,55	**37**	111,80	**46,5**	110,92	**56**	108,62	**65,5**	109,42	**75**	111,71	**84,5**	105,48
28	112,24	**37,5**	113,14	**47**	111,84	**56,5**	108,74	**66**	109,37	**75,5**	112,08	**85**	104,67
28,5	110,55	**38**	113,46	**47,5**	112,32	**57**	108,80	**66,5**	109,03	**76**	112,04		
29	109,14	**38,5**	112,67	**48**	112,36	**57,5**	108,62	**67**	108,68	**76,5**	111,46		

Bandymų rezultatų grafinis atvaizdavimas pateiktas 66 paveiksle.



66 pav. Plačiajuosčio ultragarsinio generatoriaus ADCH – perskaičiuotos SPL reikšmės

Iš 66 paveikslo matyti, kad panaudojus teorinius ričių skaičiavimus praktiniame modelyje ir keičiant jų induktyvumus skirtinguose dažnių rėžiuose, galima pasiekti vidutinį 109,49 dB SPL lygį visame užsiduotame dažnių diapazone (20÷85 kHz).

Pasinaudojus (3.3) formule, galima apskaičiuoti gautų rezultatų dispersiją (labiausiai tikėtiną eilinio matavimo vertės nukrypimą nuo aritmetinio vidurkio).

$$D(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{X})^2; \qquad (3.3)$$

čia $D(x)$ – dispersija;
n – imties dydis;
i – vieta imtyje;
x_i – skaičius imtyje;

$\bar{X}$ – empirinis vidurkis.

Empirinis vidurkis apskaičiuotas pagal (3.4) formulę.

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} x_i. \quad (3.4)$$

Pasinaudojus (3.4) formule pirmo bandymo perskaičiuotas SPL empirinis vidurkis lygus 109,39 dB, o antrojo bandymo empirinis vidurkis lygus 109,59 dB.

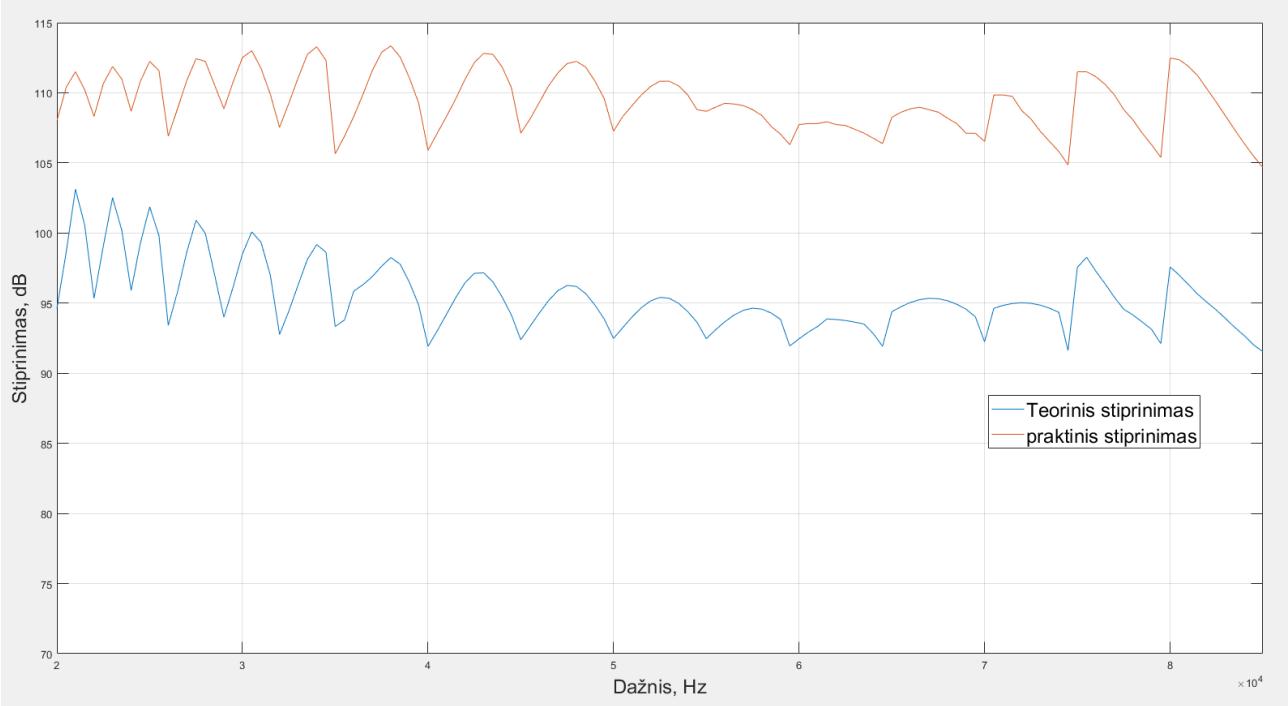
Gautus vidurkius įsistačius į (3.3) formulę, apskaičiuojamos bandymų dispersijos. Pirmojo bandymo SPL dispersija lygi 4,36 dB^2, o antrojo – 4,21 dB^2.

Ištraukus šaknį iš dispersijos vertės, gaunamas standartinis nuokrypis, kuris pirmojo bandymo metu lygus 2,09 dB, o antrojo – 2,05 dB. Palyginus gautas vertes, matyti, kad pirmojo ir antrojo bandymų standartiniai nuokrypiai skiriasi 0,019 proc.

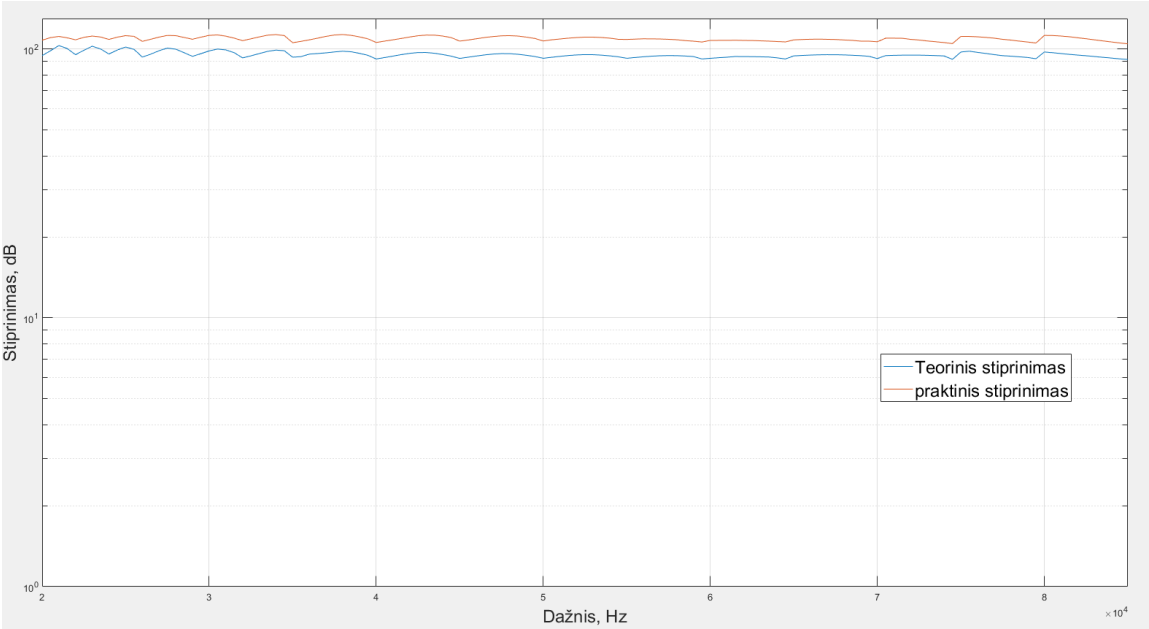
Teorinių ričių skaičiavimų stiprinimo empirinis vidurkis lygus 95,398 dB. Teorinių skaičiavimų vidurkis dėl ne identiškų komponentų naudojimo simuliacijoje ir bandymuose, 13,987 dB mažesnis nei gautas praktinis.

Simuliacijos dispersija lygi 5,567 dB^2, o standartinis nuokrypis – 2,36 dB.

Simuliacijos ir realaus bandymo palyginimas pateikiamas 67 ir 68 paveiksluose.



67 pav. Simuliacijos ir pirmo bandymo palyginimas



68 pav. Simuliacijos ir bandymo palyginimas logaritminėje skalėje

Iš grafikų matyti, kad apskaičiuoti teoriniai ir panaudoti praktiškai ričių induktyvumai, sukelia rezonansinį stiprinimą užsibrėžtuose dažnių diapazonuose, o charakteristikų netolygumas neviršija 6 dB, kas garso technikoje yra priimtinas dydis.

Pasinaudojus (3.5) formule, galima apskaičiuoti simuliacijos ir bandymų koreliacijos koeficientą.

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}; \quad (3.5)$$

čia r_{xy} – koreliacijos koeficientas;
n – imties dydis;
x_i, y_i – i eilės skaičius imtyje;
$\bar{x}, \bar{y}$ – imties vidurkiai.

Kadangi bandymų metu buvo gauta 131 vertė, o simuliacijos metu – 1000, visu pirma reikia suvienodinti vektorių ilgius. Tam tikslui atlikta simuliacija su mažiau imties taškų. Atlikus naują simuliaciją gautas 131 × 16 duomenų masyvas, kuriame aprašytos šešiolikos ričių stiprinimo reikšmės 20–85 kHz dažnio diapazone. Sujungus prasilenkiančių ričių duomenis gautas 13 × 1 duomenų masyvas, kurio duomenys, kartu su bandymų duomenimis, sustatyti į (3.5) formulę ir apskaičiuoti koreliacijos koeficientai. Simuliacijos rezultatai pateikti 2 priede.

7 lentelėje pateiktos koreliacijos koeficiento reikšmės. Pirmo bandymo ir simuliacijos koreliacijos koeficientas – 0,7998. Antro bandymo ir simuliacijos koreliacijos koeficientas – 0,7807. Pirmojo bandymo su antruoju koreliacijos koeficientas – 0,9482.

7 lentelė. Koreliacijos koeficiento reikšmių lentelė [48]

Labai stipri	**Stipri**	**Vidutinė**	**silpna**	**Labai silpna**	**Nėra ryšio**	**Labai silpna**	**Silpna**	**Vidutinė**	**Stipri**	**Labai stipri**
-1	Nuo -1 iki -0,7	Nuo -0,7 iki -0,5	Nuo -0,5 iki -0,2	Nuo -0,2 iki 0	0	Nuo 0 iki 0,2	Nuo 0,2 iki 0,5	Nuo 0,5 iki 0,7	Nuo 0,7 iki 1	+1

Iš 7 lentelės matyti, kad tarp simuliacijos ir bandymų, bei tarp pačių bandymų yra stipri koreliacija.

Išvados

1. Atlikus mokslinės literatūros analizę nustatyta, kad realaus laiko signalų apdorojimo ir generavimo, naudojamuose skaitmeniniuose mikroprocesoriuose, taikomi įvairūs algoritmai ir transformacijos: FFT, DFT, SWFT, DTFT, CZT, DWT, *Furjė transformacija* bei filtrai: aukšto, žemo dažnio, dažnių juostos, begalinio ir baigtinio impulsų atsakų filtrai, palengvinantys skaičiavimus apdorojant signalus. Signalų generavimui naudojamos įvairios bibliotekos ir vidiniai mikroprocesorių registrai, leidžiantys pasiekti aukštus dažnius (90 kHz ir daugiau).
2. Įdiegus ir išbandžius sinusinės formos signalų generatoriaus algoritmą į standartinį procesorių (ESP32), buvo išgautas nuo 1 Hz iki 90 kHz generuojamas signalas. Viršijus 90 kHz sinuso formos signalas pradeda deformuotis, dėl valdiklyje integruoto skaitmeninio – analoginio keitiklio ribotos greitaveikos.
3. Sudarius elektrostatinio plačiajuosčio ultragarsinio generatoriaus su stiprintuvu, veikiančiu rezonansiniame režime, matematinį modelį, D klasės stiprintuvo išėjimo *LC* filtrui apskaičiuoti šešiolikos ričių induktyvumai (1895 µH, 1585 µH, 1345 µH, 1105 µH, 900 µH, 725 µH, 582 µH, 460 µH, 370 µH, 303 µH, 253 µH, 212 µH, 183 µH, 159 µH, 139 µH, 123 µH), leidžiantys turėti rezonansinį stiprinimą visame užsiduotame dažnių diapazone (20–85 kHz).
4. Išmatavus plačiajuosčio ultragarsinio generatoriaus amplitudines dažnines charakteristikas (ADCH), nustatyta, kad teoriniai ričių matavimai leidžia pasiekti rezonansinį stiprinimą visame dažnių diapazone (20–85 kHz), o charakteristikų netolygumas neviršija 6 dB (pirmo bandymo metu – 4,18 dB, antro – 4,1 dB), kas garso technikoje yra priimtinas dydis.
5. Tarpusavyje palyginus plačiajuosčio ultragarsinio generatoriaus amplitudinės dažninės charakteristikos (ADCH) eksperimentinių matavimų rezultatus su modeliavimo rezultatais, nustatyta, kad tarp bandymų ir teorinių skaičiavimų yra stipri koreliacija (0,7998 su pirmu bandymu, 0,7807 su antru).

Literatūros sąrašas

1. BENDORIENĖ, A. ir kt. *Tarptautinių žodžių žodynas*. Vilnius: Alma litera, 2008. ISBN: 9789955081005.
2. JIANG, J. Audio processing with channel filtering using DSP techniques. *2018 IEEE 8th Annual Computing and Communication Workshop and Conference* [interaktyvus]. 2018, 545–550, ISBN: 978-1-5386-4650-2 [žiūrėta 2022-03-28]. Prieiga per: https://doi.org/10.1109/CCWC.2018.8301696.
3. IBRAHIM, D., DAVIES, A. The evolution of digital signal processors. *2019 6th IEEE History of Electrotechnology Conference: Histelcon* [interaktyvus]. 2019, 25–29, ISBN: 978-1-7281-5058-1 [žiūrėta 2022-03-16]. Prieiga per: https://doi.org/10.1109/HISTELCON47851.2019.9040130.
4. van DRONGELEN, W. *Signal Processing for Neuroscientists*. Second Edit. Academic Press, 2018, 307–314 ISBN 978-0-12-810482-8.
5. TAN, L., JIANG, J. *Digital Signal Processing: Fundamentals and Applications*. Third Edit. 2019. ISBN 978-0-12-815071-9.
6. TOLIĆ, I. ir kt. Efficient applications and architecture of modern digital signal processors. *Journal of Energy Technology* [interaktyvus]. 2017, 10, 35–50, ISSN: 1855-5748 [žiūrėta 2022-03-28]. Prieiga per: https://www.fe.um.si/images/jet/03_poglavje_-_JET_junij_2017.pdf.
7. PROAKIS, G., MANOLAKIS, G. *Digital Signal Processing*. 4th edition. New Jerssey: Pearson Prentice Hall, 2007. ISBN: 0-13-228731-5
8. MEET, S., RISHABH, T., HEMANT, P. Novel shifted real spectrum for exact signal reconstruction. *Interspeech 2017* [interaktyvus]. 2017, 3112–3116, ISBN: 978-1-5108-4876-4 [žiūrėta 2022-04-01]. Prieiga per: https://doi.org/10.21437/Interspeech.2017-1422.
9. ZHANG, P. ir kt. Parametric audio equalizer based on short-time Fourier transform. *2017 17th IEEE International Conference on Communication Technology* [interaktyvus]. 2017, 1648–1651, ISBN: 978-1-5090-3944-9 [žiūrėta 2022-03-20]. Prieiga per: https://doi.org/10.1109/ICCT.2017.8359910.
10. CASEBEER, J. ir kt. Auto-DSP: learning to optimize acoustic echo cancellers. *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* [interaktyvus]. 2021, 291–295, ISBN: 978-1-4577-0539-7 [žiūrėta 2022-03-30]. Prieiga per: https://doi.org/10.1109/ICASSP.2011.5947741.
11. NAG, H. ir kt. Automation in audio enhancement using unsupervised learning for ubiquitous computational environment. *Fifth International Conference on Inventive Computation Technologies*[interaktyvus]. 2020, 369–375, ISBN: 978-1-7281-4685-0 [žiūrėta 2022-03-12]. Prieiga per: https://doi.org/10.1109/ICICT48043.2020.9112473.
12. ABODENA, O. Robust and high-capacity audio watermarking based on chirp z-transform. *2021 29th Signal Processing and Communications Applications Conference* (SIU) [interaktyvus]. 2021, ISBN: 978-1-6654-3649-6 [žiūrėta 2022-04-02]. Prieiga per: https://doi.org/10.1109/SIU53274.2021.9477976.
13. AGRAWAL, N. ir kt. Design of digital IIR filter with low quantization error using hybrid optimization technique. *Soft Computing* [interaktyvus]. 2018, 22, 2953–2971 [žiūrėta 2022-04-08]. Prieiga per: https://doi.org//10.1007/s00500-017-2548-0.

14. AGRAWAI, N. ir kt. High order stable infinite impulse response filter design using cuckoo search algorithm. *International Journal of Automation and Computing* [interaktyvus]. 2017, 589–602 [žiūrėta 2022-04-20]. Prieiga per: https://doi.org/10.1007/s11633-017-1091-x.
15. KWON, B., KIM, S. Recursive optimal finite impulse response filter and its application to adaptive estimation. *Applied Sciences* [interaktyvus]. 2022, 2757 [žiūrėta 2022-04-20]. Prieiga per https://doi.org/10.3390/app12052757.
16. ALIA, R., GHOSH, S. Design of active noise control system using hybrid functional link artificial neural network and finite impulse response filters. *Neural Computing and Applications* [interaktyvus]. 2020, 2257–2266 [žiūrėta 2022-04-20]. Prieiga per: https://doi.org/10.1007/s00521-018-3697-5.
17. JONKUS, V. *Mikrovaldikliai elektroninėse grandinėse*. Mokymo priemonė. Vilnius: 2008.
18. YANG, C. ir kt. A Novel DSP architecture for scientific computing and deep learning. *Digital Object Identifier* [interaktyvus]. 2019, 36413–36425 [žiūrėta 2022-04-29]. Prieiga per: https://doi.org/10.1109/ACCESS.2019.2905302.
19. BABIUCH, M. ir kt. Using the ESP32 microcontroller for data processing. *20th International Carpathian Control Conference (ICCC)* [interaktyvus]. 2019 [žiūrėta 2022-04-29]. Prieiga per: https://doi.org/10.1109/CarpathianCC.2019.8765944.
20. KARNA, N. ir kt. Performance analysis on x86 architecture microprocessor for lightweight encryption. *3rd International Conference on Information and Communications Technology (ICOIACT)* [interaktyvus]. 2020, 262–265 [žiūrėta 2022-04-29]. Prieiga per: https://doi.org/10.1109/ICOIACT50329.2020.9331966.
21. *Espressif* [interaktyvus]. 2022 [žiūrėta 2022-05-01]. Prieiga per: https://www.espressif.com/.
22. *Eclipse: The Community for Open Innovation and Collaboration* [interaktyvus]. 2022 [žiūrėta 2022-05-01]. Prieiga per: https://www.eclipse.org/.
23. *Arduino* [interaktyvus]. 2022 [žiūrėta 2022-05-01]. Prieiga per: https://www.arduino.cc/.
24. LI, D. ir kt. Improving power efficiency for Active-RC delta-sigma modulators using a Passive-RC low-pass filter in the feedback. *IEEE Transactions on Circuits and Systems II: Express Briefs* [interaktyvus]. 2018, 1559–1563 [žiūrėta 2022-05-24]. Prieiga per: 10.1109/TCSII.2017.2762325.
25. SEONG-HO, L. Flicker prevention through transition-frequency modulation in visible light communication. *Journal of Sensor Science and Technology* [interaktyvus]. 2020, 243–248 [žiūrėta 2022-05-24]. Prieiga per: https://doi.org/10.46670/JSST.2020.29.4.243
26. RAVELO, B. On low-pass, high-pass, bandpass, and stop-band NGD RF passive circuits. *URSI Radio Science Bulletin* [interaktyvus]. 2017, 10–27 [žiūrėta 2022-05-24]. Prieiga per: 10.23919/URSIRSB.2017.8409424.
27. BELOV, V. ir kt. Tunable band-pass filter for continuous spectral analysis of cardiointervalogram. *Journal of Physics: Conference Series* [interaktyvus]. 2020, 1–5 [žiūrėta 2022-05-24]. Prieiga per: https://doi.org/10.1088/1742-6596/1515/5/052064.
28. GADELOVITS, S. ir kt. UDE-Based controller equipped with a multi-band-stop filter to improve the voltage quality of inverters. *IEEE Transactions on Industrial Electronics* [interaktyvus]. 2017, 7433–7443 [žiūrėta 2022-05-24]. Prieiga per: https:// 10.1109/TIE.2017.2698371.
29. MICHON, R. ir kt. A Faust architecture for the ESP32 microcontroller. *Sound and Music Computing Conference* [interaktyvus]. 2020, 1–7 [žiūrėta 2022-05-24]. Prieiga per: https://hal.archives-ouvertes.fr/hal-02988312.

30. RAI, P. ir kt. ESP32 based smart surveillance system. *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)* [interaktyvus]. 2019, 1–3 [žiūrėta 2022-05-24]. Prieiga per: 10.1109/ICOMET.2019.8673463.
31. MAKAROVA, A. ir kt. The use of redundant modular codes for improving the fault tolerance of special processors for digital signal processing. *YSIP2 – Proceedings of the Second Young Scientist's International Workshop on Trends in Information Processing* [interaktyvus]. 2017, 115–122 [žiūrėta 2022-05-24]. Prieiga per: http://ceur-ws.org/Vol-1837/paper17.pdf.
32. AHMADI, J. ir kt. Digital-signal-processor realization of izhikevich neural network for real-time interaction with electrophysiology experiments. *2019 26th IEEE International Conference on Electronics, Circuits and Systems* (ICECS) [interaktyvus]. 2019, 899–902 [žiūrėta 2022-05-24]. Prieiga per: 10.1109/ICECS46596.2019.8965064.
33. KLILOU, A., ARSLANE, A. Parallel implementation of pulse compression method on a multi-core digital signal processor. *International Journal of Electrical and Computer Engineering* (IJECE) [interaktyvus]. 2020, 6541–6548 [žiūrėta 2022-05-24]. Prieiga per: 10.11591/ijece.v10i6.pp6541-6548.
34. MAIER, A. ir kt. Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. *2017 Internet Technologies and Applications* (ITA) [interaktyvus]. 2017, 143–148 [žiūrėta 2022-05-24]. Prieiga per: 10.1109/ITECHA.2017.8101926.
35. PEARSON, B. ir kt. Building a low-cost and state-of-the-art iot security hands-on laboratory. *IFIP International Internet of Things Conference* [interaktyvus]. 2017, 289–306 [žiūrėta 2022-05-24]. Prieiga per: https://doi.org/10.1007/978-3-030-43605-6_17.
36. MOHAMED, A., DAHOUD, A. Integrated development environment „IDE" for Arduino. *WSN applications* [interaktyvus]. 2018, 1–11 [žiūrėta 2022-05-24]. Prieiga per: https://www.researchgate.net/profile/Mohamed-Fezari-2/publication/357157372_Automatic_solar_panel_cleaning_system_Design/links/61bdf95b4b318a6970eed1c5/Automatic-solar-panel-cleaning-system-Design.pdf.
37. DAY, M. ir kt. Annete: An intelligent tutoring companion embedded into the Eclipse IDE. *2019 IEEE First International Conference on Cognitive Machine Intelligence (CogMI)* [interaktyvus]. 2019, 71–80 [žiūrėta 2022-05-24]. Prieiga per: 10.1109/CogMI48466.2019.00018.
38. KINO, G. *Acoustic Waves: Devices, Imaging, and Analog Signal Processing (Prentice-Hall Signal Processing Series)*. New Jerssey: Pearson Prentice Hall, 1987. ISBN: 978-0130030474
39. JASIŪNIENĖ, E. *Ultragarsinė medžiagotyra*. Mokomoji knyga. Vitae Litera. Kaunas: 2012. ISBN: 978-9955-686-35-4.
40. CAROVAC, A. ir kt. Application of ultrasound in medicine. *Acta Inform Med* [interaktyvus]. 2011, 168–171 [žiūrėta 2022-10-24]. Prieiga per: doi: 10.5455/aim.2011.19.168-171.
41. ALLAM, A. ir kt. Piezoelectric transducer design for simultaneous ultrasonic power transfer and backscatter communication. *Smart Materials and Structures* [interaktyvus]. 2022, 1–12 [žiūrėta 2022-10-24]. Prieiga per: https://doi.org/10.1088/1361-665X/ac7b57.
42. MCLNTOSH, R. ir kt. Capacitive transducers with curved electrodes. *IEEE Sensors Journal* [interaktyvus]. 2006, 1–14 [žiūrėta 2022-12-05]. Prieiga per: 10.1109/JSEN.2005.854137.
43. *ESP32 Technical Reference Manual* [interaktyvus]. 2022 [žiūrėta 2022-12-28]. Prieiga per: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.

44. *ADAU1462/ADAU1466 Data Sheet* [interaktyvus]. 2017 [žiūrėta 2023-04-01]. Prieiga per: https://www.analog.com/media/en/technical-documentation/data-sheets/adau1462-1466.pdf.
45. *AD1938 Data Sheet* [interaktyvus]. 2013 [žiūrėta 2023-04-01]. Prieiga per: https://www.analog.com/media/en/technical-documentation/data-sheets/ad1938.pdf
46. *IR43x1 Data Sheet* [interaktyvus]. 2013 [žiūrėta 2023-04-01]. Prieiga per: https://www.infineon.com/dgdl/ir4301.pdf?fileId=5546d462533600a4015355d5fc691819
47. *Ultrasound Research Group* [interaktyvus]. [žiūrėta 2023-04-01]. Prieiga per: https://www.ultrasoundresearchgroup.com
48. RUDZKIENĖ, V. *Statistinės technologijos teisėje ir valdyme. Praktiniai darbai*. Mokomasis leidinys. Kaunas: Vitae Litera. 2012. ISBN: 9955 – 563 – 21 – 4.
49. MASON, T. Therapeutic ultrasound an overview. *Ultrasonics Sonochemistry* [interaktyvus]. 2011, 847–852 [žiūrėta 2023-04-27]. Prieiga per: doi:10.1016/j.ultsonch.2011.01.004.
50. MCHUGH, T. Putting ultrasound to use in food processing. *FoodTechnology* [interaktyvus]. 2016, 72–74 [žiūrėta 2023-04-27]. Prieiga per: doi:10.1007/978-1-44-19-7472-3_3.
51. SZABO, T. *Diagnostic Ultrasound Imaging: Inside Out*. 2nd edition. Boston: Academic Press 2014. ISBN: 978-0-12-396487-8
52. SHORT, M. ir kt. *Power Ultrasonics*. Sawston: Woodhead Publishing 2015. ISBN: 978-1-78242-028-6
53. SHAYAKHMETOVA, E. ir kt. Sonochemistry: Synthesis of bioactive heterocycles. *Synthetic Communications: An International Journal for Rapid Communication of Synthetic Organic Chemistry* [interaktyvus]. 2015, 2155–2191 [žiūrėta 2023-04-27]. Prieiga per: 10.1080/00397911.2014.893360.
54. HUSSEIN, E. ir kt. Ultrasonic welding of nickel with coarse and ultrafine grained structures. *Metals* [interaktyvus]. 2021, 1–17 [žiūrėta 2023-04-27]. Prieiga per: https://doi.org/10.3390/met11111800.

Priedai

1 priedas. ESP32 sinusinės formos signalo generatoriaus kodas

generatorius_OK §

```
#include "arduino.h"
#include "soc/sens_reg.h"
#include "soc/rtc.h"
#include "driver/dac.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);

#define button_Back    19  // E
#define button_Down    14  // U
#define button_Up    13  // D
#define button_Ok    12  // B
#define DEFAULT_DELAY 300

int clk_8m_div = 0;       // RTC 8M clock divider (division is by clk_8m_div+1
int frequency_step = 1;  // Frequency step for CW generatorint
int scale = 0;         // 50% of the full scale
int offset;              // leave it default / 0 = no any offset
int invert = 0;  // invert MSB to get sine waveform
char buttonPressed = '0';

byte menuLevel = 0;     // Level 0: no menu display, display anything you like
                        // Level 1: display main menu
                        // Level 2: display sub menu
                        // Level 3: display sub menu of sub menu

byte menu = 1;
byte sub = 1;

unsigned long relay_val_1 = 0;
unsigned long relay_val_2 = 0;
unsigned long relay_val_3 = 0;

bool LED_STATE = false;

bool currState_B = HIGH;
bool currState_D = HIGH;
bool currState_U = HIGH;
bool currState_E = HIGH;

bool prevState_B = HIGH;
bool prevState_D = HIGH;
bool prevState_U = HIGH;
bool prevState_E = HIGH;

unsigned long prevTime_B = 0;
unsigned long prevTime_D = 0;
unsigned long prevTime_U = 0;
unsigned long prevTime_E = 0;

unsigned long waitTime_B = 50;
unsigned long waitTime_D = 50;
unsigned long waitTime_U = 50;
unsigned long waitTime_E = 50;

byte charUp[8] = {
        B00100,
```

```
        B01110,
        B11111,
        B00000,
        B00000,
        B00000,
        B00000,
        B00000
};
byte charDown[8] = {
        B00000,
        B00000,
        B00000,
        B00000,
        B00000,
        B11111,
        B01110,
        B00100
};
byte charUpDown[8] = {
        B00100,
        B01110,
        B11111,
        B00000,
        B00000,
        B11111,
        B01110,
        B00100
};

void checkButton() {

  // Button Debouncing
  bool currRead_B = digitalRead(button_Back);
  bool currRead_D = digitalRead(button_Down);
  bool currRead_U = digitalRead(button_Up);
  bool currRead_E = digitalRead(button_Ok);

  if (currRead_B != prevState_B) {
    prevTime_B = millis();
  }
  if (currRead_D != prevState_D) {
    prevTime_D = millis();
  }
  if (currRead_U != prevState_U) {
    prevTime_U = millis();
  }
  if (currRead_E != prevState_E) {
    prevTime_E = millis();
  }

  if ((millis() - prevTime_B) > waitTime_B) {
    if (currRead_B != currState_B) {
      currState_B = currRead_B;
      if (currState_B == LOW) {
        buttonPressed = 'B';
      }
    }
```

```
  } else buttonPressed = '0';
  if ((millis() - prevTime_D) > waitTime_D) {
    if (currRead_D != currState_D) {
      currState_D = currRead_D;
      if (currState_D == LOW) {
        buttonPressed = 'D';
      }
    }
  } else buttonPressed = '0';
  if ((millis() - prevTime_U) > waitTime_U) {
    if (currRead_U != currState_U) {
      currState_U = currRead_U;
      if (currState_U == LOW) {
        buttonPressed = 'U';
      } else {
        //buttonPressed = '0';
      }
    }
  } else buttonPressed = '0';
  if ((millis() - prevTime_E) > waitTime_E) {
    if (currRead_E != currState_E) {
      currState_E = currRead_E;
      if (currState_E == LOW) {
        buttonPressed = 'E';
      }
    }
  } else buttonPressed = '0';

  prevState_B = currRead_B;
  prevState_D = currRead_D;
  prevState_U = currRead_U;
  prevState_E = currRead_E;

  processButton(buttonPressed);

}

void processButton(char buttonPressed) {

  switch(menuLevel) {
    case 0:                       // Level 0, home screen
      switch ( buttonPressed ) {
        case 'E':                 // Enter
          menu = 1;
          menuLevel = 1;          // go to main menu
          updateLevel_1();        // show main menu
          delay(DEFAULT_DELAY);
          break;
        case 'U': // Up
          break;
        case 'D': // Down
          break;
        case 'B': // Back
          menuLevel = 0;          // go to home screen
          updateLevel_0();        // show home screen
          delay(DEFAULT_DELAY);
          break;
```

```
      default:
        break;
    }
    break;
  case 1:                     // Level 1, main menu
    switch ( buttonPressed ) {
      case 'E':               // Enter
        sub = 1;
        menuLevel = 2;        // go to sub menu
        updateLevel_2();      // show sub menu
        delay(DEFAULT_DELAY);
        break;
      case 'U':               // Up
        menu++;
        updateLevel_1();      // show main menu
        delay(DEFAULT_DELAY);
        break;
      case 'D':               // Down
        menu--;
        updateLevel_1();      // show main menu
        delay(DEFAULT_DELAY);
        break;
      case 'B': // Back
        menuLevel = 0;        // hide menu, go back to level 0
        updateLevel_0();      // show home screen
        delay(DEFAULT_DELAY);
        break;
      default:
        break;
    }
    break;
  case 2:                     // Level 2, sub menu
    switch ( buttonPressed ) {
      case 'E':               // Enter
        if (sub == 2) {       // Jump to sub menu of sub menu
          menuLevel = 3;        // go to sub menu of sub menu
          updateLevel_3();      // show sub menu of sub menu
        } else if (sub == 3) {
          executeAction();
          delay(1000);
          menuLevel = 2;        // go to sub menu
          updateLevel_2();      // show sub menu
        }
        delay(DEFAULT_DELAY);
        break;
      case 'U':               // Up
        sub++;
        updateLevel_2();
        delay(DEFAULT_DELAY);
        break;
      case 'D':               // Down
        sub--;
        updateLevel_2();      // show main menu
        delay(DEFAULT_DELAY);
        break;
      case 'B':               // Back
        menuLevel = 1;        // go back to level 1
```

```
229           updateLevel_1();       // show main menu
230           delay(DEFAULT_DELAY);
231           break;
232         default:
233           break;
234       }
235       break;
236     case 3:                      // Level 3, sub menu of sub menu
237       switch ( buttonPressed ) {
238         case 'E':                // Enter
239           menuLevel = 2;         // go back to level 2
240           updateLevel_2();       // show sub menu
241           delay(DEFAULT_DELAY);
242           break;
243         case 'U':                // Up
244           if (sub == 2) {        // edit value
245             if        (menu == 1) {
246               if (relay_val_1 < 7) {  // 1 hour max
247                 relay_val_1 = relay_val_1 + 1;
248               } else {
249                 relay_val_1 = 7;
250               }
251             } else if (menu == 2) {
252               if (relay_val_2 < 3) {  // 1 hour max
253                 relay_val_2 = relay_val_2 + 1;
254               } else {
255                 relay_val_2 = 3;
256               }
257             } else if (menu == 3) {
258               if (relay_val_3 < 1000) {  // 1 hour max
259                 relay_val_3 = relay_val_3 + 1;
260               } else {
261                 relay_val_3 = 1000;
262               }
263             }
264           }
265           updateLevel_3();       // show sub menu
266           delay(DEFAULT_DELAY);
267           break;
268         case 'D':                // Down
269           if (sub == 2) {        // edit value
270             if        (menu == 1) {
271               if (relay_val_1 == 0) {
272                 relay_val_1 = 0;
273               } else {
274                 relay_val_1 = relay_val_1 - 1;
275               }
276             } else if (menu == 2) {
277               if (relay_val_2 == 0) {
278                 relay_val_2 = 0;
279               } else {
280                 relay_val_2 = relay_val_2 - 1;
281               }
282             } else if (menu == 3) {
283               if (relay_val_3 == 0) {
284                 relay_val_3 = 0;
285               } else {
```

```
                relay_val_3 = relay_val_3 - 1;
              }
            }
          }
          updateLevel_3();       // show sub menu
          delay(DEFAULT_DELAY);
          break;
        case 'B':                // Back
          menuLevel = 2;         // go back to main menu
          updateLevel_2();       // show main menu
          delay(DEFAULT_DELAY);
          break;
        default:
          break;
      }
      break;
    default:
      break;
  }

}

void updateLevel_0() {
  lcd.clear();
  lcd.println("Generatorius    ");
  lcd.setCursor(0,1);
  lcd.println("- OK del meniu  ");
}

void updateLevel_1 () {

  switch (menu) {
    case 0:
      menu = 1;
      break;
    case 1:
      lcd.clear();
      lcd.print(">clc div.: ");
      lcd.print(relay_val_1);
      lcd.setCursor(0, 1);
      lcd.print(" Offset: ");
      lcd.setCursor(15,1);
      lcd.write((byte)1);      // down arrow
      break;
    case 2:
      lcd.clear();
      lcd.print(" clc div. ");
      lcd.setCursor(0, 1);
      lcd.print(">Offset: ");
      lcd.print(relay_val_2);
      lcd.setCursor(15,1);
      lcd.write((byte)2);      // up and down arrow
      break;
    case 3:
      lcd.clear();
      lcd.print(">Dazn:");
      lcd.print(relay_val_3 * 2048 / 16.4 / ((relay_val_1 + 1)), 0);
```

```
      lcd.print("Hz");
      lcd.setCursor(15,1);
      lcd.write((byte)0);      // up arrow
      break;
    case 4:
      menu = 3;
      break;
  }

}

void updateLevel_2 () {
  switch (menu) {
    case 0:
      break;
    case 1:                                   // Relay 1
      switch (sub) {
        case 0:
          break;
        case 1:
          lcd.clear();
          lcd.print(" clc div.");
          lcd.setCursor(0, 1);
          lcd.print("  Reiksme = ");
          lcd.print(relay_val_1);
          lcd.setCursor(15,1);
          lcd.write((byte)1);      // down arrow
          break;
        case 2:
          lcd.clear();
          lcd.print(" clc div.");
          lcd.setCursor(0, 1);
          lcd.print("Nustatyti dydi");
          lcd.setCursor(15,1);
          lcd.write((byte)2);      // up and down arrow
          break;
        case 3:
          lcd.clear();
          lcd.print(" clc div.");
          lcd.setCursor(0, 1);
          lcd.print(" Patvirtinti ");
          lcd.setCursor(15,1);
          lcd.write((byte)0);      // up arrow
          break;
        default:
          break;
      }
      break;
    case 2:                                   // Relay 2
      switch (sub) {
        case 0:
          break;
        case 1:
          lcd.clear();
          lcd.print(" Offset:");
          lcd.setCursor(0, 1);
          lcd.print("  Reiksme = ");
```

```
400           lcd.print(relay_val_2);
401           lcd.setCursor(15,1);
402           lcd.write((byte)1);      // down arrow
403           break;
404         case 2:
405           lcd.clear();
406           lcd.print(" Offset:");
407           lcd.setCursor(0, 1);
408           lcd.print(" Nustatyti dydi ");
409           lcd.setCursor(15,1);
410           lcd.write((byte)2);      // up and down arrow
411           break;
412         case 3:
413           lcd.clear();
414           lcd.print(" Offset:");
415           lcd.setCursor(0, 1);
416           lcd.print(" Patvirtinti ");
417           lcd.setCursor(15,1);
418           lcd.write((byte)0);      // up arrow
419           break;
420         default:
421           break;
422       }
423       break;
424     case 3:                                 // Relay 3
425       switch (sub) {
426         case 0:
427           break;
428         case 1:
429           lcd.clear();
430           lcd.print(" Dazmis:");
431           lcd.setCursor(0, 1);
432           lcd.print("Dazmis = ");
433       lcd.print(relay_val_3 * 2048 / 16.4 / ((relay_val_1 + 1)), 0);
434           lcd.setCursor(15,1);
435           lcd.write((byte)1);      // down arrow
436           break;
437         case 2:
438           lcd.clear();
439           lcd.print(" Dazmis:");
440           lcd.setCursor(0, 1);
441           lcd.print(" Nustatyti dazni");
442           lcd.setCursor(15,1);
443           lcd.write((byte)2);      // up and down arrow
444           break;
445         case 3:
446           lcd.clear();
447           lcd.print(" Dazmis:");
448           lcd.setCursor(0, 1);
449           lcd.print(" Patvirtinti ");
450           lcd.setCursor(15,1);
451           lcd.write((byte)0);      // up arrow
452           break;
453         default:
454           break;
455       }
456       break;
```

```
    case 4:
      sub = 3;
      break;
  }
}

void updateLevel_3 () {
  switch (menu) {
    case 0:

      break;
    case 1:
      lcd.clear();
      lcd.print("clc div.");
      lcd.setCursor(0, 1);
      lcd.print("Reiksme = ");
      lcd.print(relay_val_1);
      break;
    case 2:
      lcd.clear();
      lcd.print("offset:");
      lcd.setCursor(0, 1);
      lcd.print("Reiksme = ");
      lcd.print(relay_val_2);
      break;
    case 3:
      lcd.clear();
      lcd.print("Daznis:");
      lcd.setCursor(0, 1);
      lcd.print("Reiksme=");
      lcd.print(relay_val_3 * 2048 / 16.4 / ((relay_val_1 + 1)), 0);
      lcd.setCursor(14, 1);
      lcd.print("Hz");
      break;
    case 4:
      sub = 3;
      break;
  }
}

void executeAction () {
  switch (menu) {
    case 0:

      break;
    case 1:
      lcd.clear();
      lcd.print("Tvirtinama");
      break;
    case 2:
      lcd.clear();
      lcd.print("Tvirtinama");
      break;
    case 3:
      lcd.clear();
      lcd.print("Tvirtinama");
      break;
```

```
    case 4:
      sub = 3;
      break;
  }
}
 void dac_cosine_enable(dac_channel_t channel)
{
  // Enable tone generator common to both channels
  SET_PERI_REG_MASK(SENS_SAR_DAC_CTRL1_REG, SENS_SW_TONE_EN);
  switch(channel)
  {
      case DAC_CHANNEL_1:
          // Enable / connect tone tone generator on / to this channel
          SET_PERI_REG_MASK(SENS_SAR_DAC_CTRL2_REG, SENS_DAC_CW_EN1_M);
          // Invert MSB, otherwise part of waveform will have inverted
          SET_PERI_REG_BITS(SENS_SAR_DAC_CTRL2_REG, SENS_DAC_INV1, 2, SENS_DAC_INV1_S);
          break;
      case DAC_CHANNEL_2:
          SET_PERI_REG_MASK(SENS_SAR_DAC_CTRL2_REG, SENS_DAC_CW_EN2_M);
          SET_PERI_REG_BITS(SENS_SAR_DAC_CTRL2_REG, SENS_DAC_INV2, 2, SENS_DAC_INV2_S);
          break;
  }
}


/*
// * Set frequency of internal CW generator common to both DAC channels
// *
// * clk_8m_div: 0b000 - 0b111
// * frequency_step: range 0x0001 - 0xFFFF
// *
// */
void dac_frequency_set(int clk_8m_div, int frequency_step)
{
    clk_8m_div = relay_val_1;
    REG_SET_FIELD(RTC_CNTL_CLK_CONF_REG, RTC_CNTL_CK8M_DIV_SEL, clk_8m_div);
    SET_PERI_REG_BITS(SENS_SAR_DAC_CTRL1_REG, SENS_SW_FSTEP, frequency_step, SENS_SW_FSTEP_S);
}

/*
 * Scale output of a DAC channel using two bit pattern:
 *
 * - 00: no scale
 * - 01: scale to 1/2
 * - 10: scale to 1/4
 * - 11: scale to 1/8
 *
 */
void dac_scale_set(dac_channel_t channel, int scale)
{
  switch(channel)
  {
      case DAC_CHANNEL_1:
          SET_PERI_REG_BITS(SENS_SAR_DAC_CTRL2_REG, SENS_DAC_SCALE1, scale, SENS_DAC_SCALE1_S);
          break;
      case DAC_CHANNEL_2:
          SET_PERI_REG_BITS(SENS_SAR_DAC_CTRL2_REG, SENS_DAC_SCALE2, scale, SENS_DAC_SCALE2_S);
```

```
            break;
    }
}


/*
 * Offset output of a DAC channel
 *
 * Range 0x00 - 0xFF
 *
 */
void dac_offset_set(dac_channel_t channel, int offset)
{
    switch(channel)
    {
        case DAC_CHANNEL_1:
            SET_PERI_REG_BITS(SENS_SAR_DAC_CTRL2_REG, SENS_DAC_DC1, offset, SENS_DAC_DC1_S);
            break;
        case DAC_CHANNEL_2:
            SET_PERI_REG_BITS(SENS_SAR_DAC_CTRL2_REG, SENS_DAC_DC2, offset, SENS_DAC_DC2_S);
            break;
    }
}


/*
 * Invert output pattern of a DAC channel
 *
 * - 00: does not invert any bits,
 * - 01: inverts all bits,
 * - 10: inverts MSB,
 * - 11: inverts all bits except for MSB
 *
 */
void dac_invert_set(dac_channel_t channel, int invert)
{
    switch(channel)
    {
        case DAC_CHANNEL_1:
            SET_PERI_REG_BITS(SENS_SAR_DAC_CTRL2_REG, SENS_DAC_INV1, invert, SENS_DAC_INV1_S);
            break;
        case DAC_CHANNEL_2:
            SET_PERI_REG_BITS(SENS_SAR_DAC_CTRL2_REG, SENS_DAC_INV2, invert, SENS_DAC_INV2_S);
            break;
    }
}

void setup(void) {
  lcd.begin();
  lcd.createChar(0, charUp);
  lcd.createChar(1, charDown);
  lcd.createChar(2, charUpDown);

  Serial.begin(9600);

  pinMode(button_Back, INPUT_PULLUP);
  pinMode(button_Down, INPUT_PULLUP);
```

```
  pinMode(button_Up, INPUT_PULLUP);
  pinMode(button_Ok, INPUT_PULLUP);


  updateLevel_0();


  dac_frequency_set(clk_8m_div, 2);

  dac_cosine_enable(DAC_CHANNEL_1);
  dac_cosine_enable(DAC_CHANNEL_2);

  dac_output_enable(DAC_CHANNEL_1);
  dac_output_enable(DAC_CHANNEL_2);
}

void loop() {


  checkButton();

  dac_frequency_set(clk_8m_div, relay_val_3);

}
```

2 priedas. Simuliacijos rezultatai

***f*, kHz**	***SPL*, dB**	***f*, kHz**	***SPL*, dB**	***f*, kHz**	***SPL*, dB**	***f*, kHz**	***SPL*, dB**	***f*, kHz**	***SPL*, dB**	***f*, kHz**	***SPL*, dB**	***f*, kHz**	***SPL*, dB**
20	94,62	**29,5**	96,16	**39**	96,48	**48,5**	95,68	**58**	94,57	**67,5**	95,32	**77**	95,41
20,5	98,63	**30**	98,51	**39,5**	94,86	**49**	94,86	**58,5**	94,29	**68**	95,17	**77,5**	94,56
21	103,14	**30,5**	100,09	**40**	91,92	**49,5**	93,86	**59**	93,84	**68,5**	94,92	**78**	94,15
21,5	100,53	**31**	99,34	**40,5**	93,03	**50**	92,48	**59,5**	91,94	**69**	94,57	**78,5**	93,65
22	95,34	**31,5**	97,01	**41**	94,21	**50,5**	93,26	**60**	92,46	**69,5**	94,03	**79**	93,13
22,5	99,07	**32**	92,77	**41,5**	95,40	**51**	94,01	**60,5**	92,93	**70**	92,24	**79,5**	92,11
23	102,52	**32,5**	94,43	**42**	96,46	**51,5**	94,67	**61**	93,34	**70,5**	94,63	**80**	97,58
23,5	100,17	**33**	96,29	**42,5**	97,13	**52**	95,16	**61,5**	93,87	**71**	94,84	**80,5**	96,99
24	95,90	**33,5**	98,12	**43**	97,16	**52,5**	95,41	**62**	93,83	**71,5**	94,98	**81**	96,30
24,5	99,29	**34**	99,18	**43,5**	96,52	**53**	95,35	**62,5**	93,76	**72**	95,03	**81,5**	95,62
25	101,87	**34,5**	98,64	**44**	95,42	**53,5**	94,99	**63**	93,65	**72,5**	94,99	**82**	95,05
25,5	99,81	**35**	93,33	**44,5**	94,15	**54**	94,39	**63,5**	93,51	**73**	94,85	**82,5**	94,50
26	93,41	**35,5**	93,80	**45**	92,39	**54,5**	93,64	**64**	92,82	**73,5**	94,64	**83**	93,89
26,5	95,80	**36**	95,85	**45,5**	93,34	**55**	92,47	**64,5**	91,92	**74**	94,35	**83,5**	93,25
27	98,65	**36,5**	96,32	**46**	94,29	**55,5**	93,10	**65**	94,38	**74,5**	91,62	**84**	92,67
27,5	100,91	**37**	96,89	**46,5**	95,18	**56**	93,67	**65,5**	94,75	**75**	97,55	**84,5**	92,02
28	99,97	**37,5**	97,63	**47**	95,89	**56,5**	94,15	**66**	95,05	**75,5**	98,27	**85**	91,54
28,5	97,01	**38**	98,25	**47,5**	96,26	**57**	94,49	**66,5**	95,25	**76**	97,27		
29	94,00	**38,5**	97,78	**48**	96,19	**57,5**	94,64	**67**	95,34	**76,5**	96,37		