TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Marten Niinepuu 142843

TARKVARALINE MODEM

Bakalaureusetöö

Juhendaja: Ivo Müürsepp
dotsent

Tallinn 2017

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Marten Niinepuu

10.05.2017

Annotatsioon

Seoses tegevusega Kaitseliidus jäi silma probleem, kus laialdaselt kasutatakse käsiraadiojaamu (Motorola, Icom), milledel esineb kaks puudust. Esiteks puudub andmeedastus võimekus ning lisaks ei ole tagatud side turvalisus. Tulenevalt eelnimetatud situatsioonist tekkis autoril idee töötada välja lihtne tarkvaraline modem, mis võimaldaks eelnevalt mainitud raadiojaamadega teostada andmeedastust. Kui andmevahetus edukalt toimib oleks võimalik väheste vahenditega luua aeglane, kuid krüpteeritud andmesideühendus Kaitseliidu üksuste siseselt.

Töö eesmärgiks on uurida võimalike viise andmete analoogkujul edastamiseks läbi tarkvaralise modemi ning edastatud andmete töötlemist vastuvõtjas. Antud töö käigus püüab autor leida sobiva programmi, et andmeid saaks edastada ka läbi raadiokanali. Vastavalt antud piirangule valitakse ka sobivad standardid ning viisid kuidas andmeid töödelda.

Töö käigus uuriti ning testiti erinevaid programme ning lahendusi kõige efektiivsemaks andmeedastuseks. Lisaks teostas autor mitmeid mõõtmisi, mille tulemused on töös kajastatud.

Töö tulemusena valmis tarkvara, millega on võimalik edastada andmeid ühest seadmest teise läbi audiokaabli ja raadiokanali.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 27 leheküljel, 5 peatükki, 26 joonist, 2 tabelit.

Abstract

Software modem

Author is a member of Estonian Defence League where they use portable radio stations such as Motorola and Icom for communication. Author concerns because radio communication with this kind devices are not secured and do not allow data transmission. The project target is to make simple software modem which afford data transmission through radio channel. Next step would be modem what can be used to have encrypted link between radio stations.

The main purpose of this project is to explore how to send analog data through software modem and process it in receiver. What are the best solutions and which programming language to use to send data over radio channel also. Important is to keep in mind that the device must exploit existing communication standards.

During the research author examine and tested different programs and solutions to figure out most effective way for data transmission. In project are shown results of measurements and tests.

As a result created program which can be used to transmit data from one computer to another through audiocable and radio channel using portable radio stations.

The thesis is in Estonian and contains 27 pages of text, 5 chapters, 26 figures, 2 tables.

Lühendite ja mõistete sõnastik

CW	*Continuous wave* – pidev laine
AM	*Amplitude modulation* – amplituudmodulatsioon
FM	*Frequency modulation* – sagedusmodulatsioon
BPS	*Bits per second* – bittide arv sekundis
HSP	*Host Signal Processing* – vastuvõtva signaali töötlemine
FSK	*Frequency Shift Keying* – sagedusmanipulatsioon
PSK	*Phase Shift Keying* – faasmanipulatsioon
BW	*Bandwidth* – ribalaius
QAM	*Quadrature amplitude modulation* – kvadratuurne amplituudmodulatsioon
FIR	*Finite impulse response* – lõpliku impulsskajaga filter
USB	*Universal serial bus* – universaalne järjestiksiin

Sisukord

1 Sissejuhatus 9
2 Modem 10
2.1 Tarkvaraline modem 10
2.2 Riistvaraline modem 10
3 Modulatsioon 11
3.1 Amplituudmodulatsioon 11
3.2 Sagedusmodulatsioon 12
3.3 Sagedus- ja amplituudmodulatsiooni võrdlus 14
3.4 Sagedusmanipulatsioon 16
3.5 Faasmanipulatsioon 16
4 Signaal 18
4.1 Analoogsignaal 18
4.2 Digitaalsignaal 18
4.3 Nyquist´i kriteerium 18
4.4 Lõpliku impulsskajaga filter 19
5 Rakendus 21
5.1 Saatja 21
5.2 Vastuvõtja 24
5.3 Katsed / mõõtmised 25
5.3.1 Katse I 25
5.3.2 Katse II 28
5.3.3 Katse III 31
5.3.4 Katse IV 33
Kokkuvõte 36
Kasutatud kirjandus 37
Lisa 1 – Saatjas kasutatav lähtekood 39
Lisa 2 – Vastuvõtjas kasutatav lähtekood 41
Lisa 3 – Lähtekood signaali graafiliseks kujutamiseks 44

Jooniste loetelu

Joonis 1. Modulatsiooniviisid [4, p. 8] 11
Joonis 2. Amplituudmodulatsioon [4, p. 17] 12
Joonis 3. Sagedusmodulatsioon [4, p. 10] 13
Joonis 4. FSK ajaline kuju [5, p. 14] 16
Joonis 5. Näide faasmanipulatsioonist [5, p. 15] 17
Joonis 6. FIR filtri plokkskeem [14] 20
Joonis 7. ASCII tabel [15] 22
Joonis 8 Moduleeritud signaali graafiline kuju 23
Joonis 9 Moduleeritud signaali ajaline kuju 23
Joonis 10 Teoreetilise signaali spektripilt 24
Joonis 11. Katse I skeem 26
Joonis 12. Katse I sisendandmed 26
Joonis 13. Katse I signaal 27
Joonis 14. Katse I ostsilloskoobi spektripilt 27
Joonis 15. Katse II skeem 28
Joonis 16. Katse II sisendandmed 29
Joonis 17. Katse II väljundandmed 29
Joonis 18. Katse II signaal graafiliselt 30
Joonis 19. Katse III skeem 31
Joonis 20. Katse III sisendandmed 32
Joonis 21. Katse III väljundandmed 32
Joonis 22. Katse III signaal graafiliselt 33
Joonis 23. Katse IV skeem 34
Joonis 24. Katse IV sisendandmed 34
Joonis 25. Katse IV väljundandmed 35
Joonis 26. Katse IV signaal graafiliselt 35

Tabelite loetelu

Table 1. Sagedus- ja amplituudmodulatsiooni erinevused [3, p. 191] 15
Table 2. Modemite standardid [11, p. 129] 19

1 Sissejuhatus

Autor on jõudnud antud teema valikuni seoses Kaitseliidus kasutatavate käsiraadiojaamade võimekusele. Olles Kaitseliidu tegevliige, on autor avastanud ennast olukorrast, kus on käimas riiklikul tasandil julgeoleku tagamise õppus, kuid sidepidamisvahenditena kasutatakse tsiviilotstarbelisi seadmeid, mille andmevahetus on pealtkuulatav või puudulik. Krüptoseadmete puudumine on tingitud nende hinnaklassi tõttu ning teatava taseme seadmete kasutamisega kaasnevate juriidiliste julgeolekuprobleemide teke, kus kaitseliitlasel ei pruugi olla vajaliku taseme riigisaladuse luba. Seoses antud olukorraga tekkis idee luua lihtne tarkvaraline modem, mis leiaks kasutust odavate ning krüpteerimata sidepidamisvahendite jaoks. Tarkvaralise modemi kasutamine võimaldab vältida kulutusi, mis tuleks teha täiendava riistvara soetamiseks, kui on võimalik kasutada igat ettejuhtuvat arvutit. Ideaalse lahendusena saaks tarkvara luua nutitelefonile ning sidepidamiseks vajalik riistvara oleks igal sõduril olemas.

Antud töö eesmärgiks on luua programm, millega teostada andmeedastust, läbi tarkvaralise modemi, kahe seadme vahel. Edastuskanaliks kasutatakse raadiokanalit. Tarkvaralise modemina on kasutusel arvuti helikaart, mis moduleerib ning demomoduleerib kasutaja poolt sisestatud andmed vastavalt standardile, milleks on Bell 202. Eesmärgini jõudmiseks uuritakse andmete töötlemise võimalusi ning luuakse tarkvara, millega lahendada püstitatud probleem.

Töö on jaotatud neljaks osaks, mille jooksul kirjeldatakse modemeid ning nendes kasutatavaid modulatsiooni viise. Eraldi tuuakse välja signaalikujud ning nende töötlemise võimalused ja standardid. Töö viimases osas on koostatud programm ning teostatud mõõtmisi ja katseid testimaks loodud tarkvara töötamist.

Antud prototüübi eesmärk on leida sobivaid lahendusi, et tulevikus luua modem, mis leiaks rakendust käsiraadiojaamade vahelise turvalise andmeedastuse loomiseks.

2 Modem

Füüsilises kanalis on võimalik andmeid edastada kui need on analoog kujul ning selleks kasutatakse seadet, modemit, millega andmed antud kujule viiakse. Modulaator konverteerib digitaalsed andmed selliselt, et neid saaks edastada läbi piiratud sagedusribaga analoog kanali. Demodulaator omakorda konverteerib andmed tagasi esialgsele, digitaalsele kujule. Modemi edastuskiirust väljendatakse bittide arvuga sekundis või baudides. Aeglaste modemite puhul jääb edastuskiirus vahemiku 0 – 1200 BPS [1, pp. 121-122].

2.1 Tarkvaraline modem

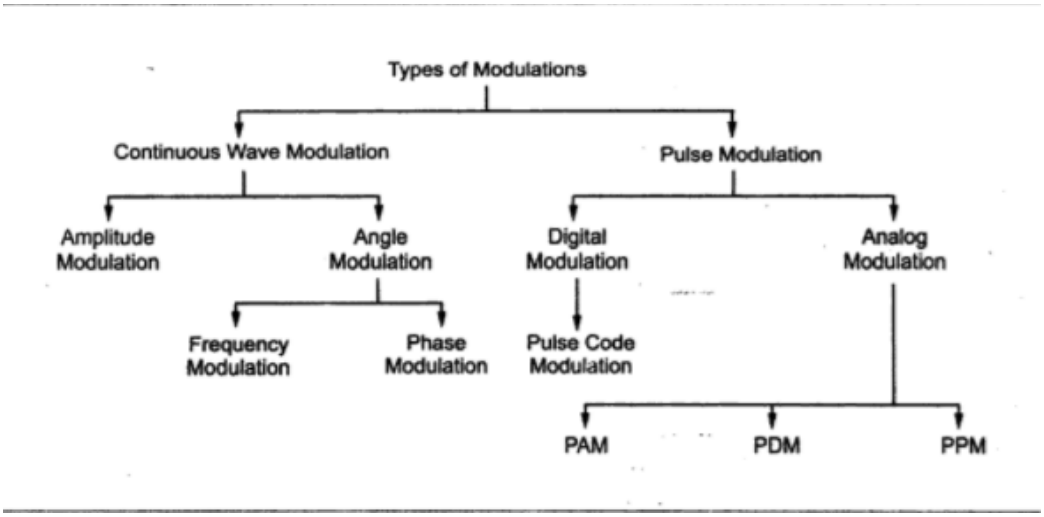
... kasutab tarkvara, mille abil moduleeritakse ja demoduleeritakse signaale ning kontrollitakse modemi tegevust. Kasutusel on minimaalselt riistvara, mis on arvutites standardselt olemas. Antud modemeid on kahte erinevat liiki: ilma kontrollerita, kuid riistvara sisaldavad ja ilma riistvarata, edaspidi HSP, kus kogu tegevus on kontrollitud tarkvara poolt. Juhtseadmeta modemite puhul tegeleb modemi kontrollimisega tarkvara, kuid andmete töötlemisega tegeleb riistvara. HSP seadmete puhul kasutatakse kogu tegevuse haldamiseks tarkvaralist lahendust, välja arvatud põhiliste baasfunktsioonide täitmist. [2, p. 683]. Töös kasutatakse tarkvaralise modemina igas kasutatavas arvutis olevat helikaarti, läbi mille teostatakse andmevahetust.

2.2 Riistvaraline modem

... kasutab riistvaralisi kiipe, mille abil muundatakse signaale ühest liigist teise ning kontrollitakse modemi tegevust teatud sisendite korral. Üldjuhul ühendatakse riistvaraline modem arvutiga läbi USB, RS-232 või mõne muu pordi ning tegemist on välise seadmega. Ühendatava seadme töötamiseks on kõik vajalik antud seadmel sisemiselt olemas [2, p. 683].

3 Modulatsioon

Modulatsiooniks sidetehnika mõistes on kandesignaali parameetrite muutmine vastavalt ülekantava signaali muutumisele. Raadio- ja sidetehnikas nimetatakse modulatsiooniks kõrgsageduslike elektrivõnkumiste või impulsijadade mingi parameetri muutmist tunduvalt madalama sagedusega signaali rütmis. Modulatsiooni eesmärgiks on tagada signaali edastamine ülekandekanalis, mis töötab piiratud sagedusalas. [3, pp. 5-6] Modulatsiooni võib klassifitseerida mitmeti, kuid järgnevalt on välja toodud analoog ehk pidev ning digitaalne ehk mittepidev modulatsiooni. Analoogmodulatsiooni korral muutub kõrgsageduslike võnkumiste mingi parameeter pidevalt. Digitaalse modulatsiooni korral on sellel parameetril lõplik arv lubatud ja fikseeritud väärtuseid. Raadiosides kasutatakse kas ühe või mitme harmoonilise (siinussignaali) kandesignaali moduleerimist. Modulatsiooniviisid jagunevad amplituud ning nurkmodulatsiooniks. Viimase alla kuuluvad sagedus ning faasmodulatsioon, kus faas muutub vastavalt põhiriba signaali väärtusele [4, pp. 7-8].

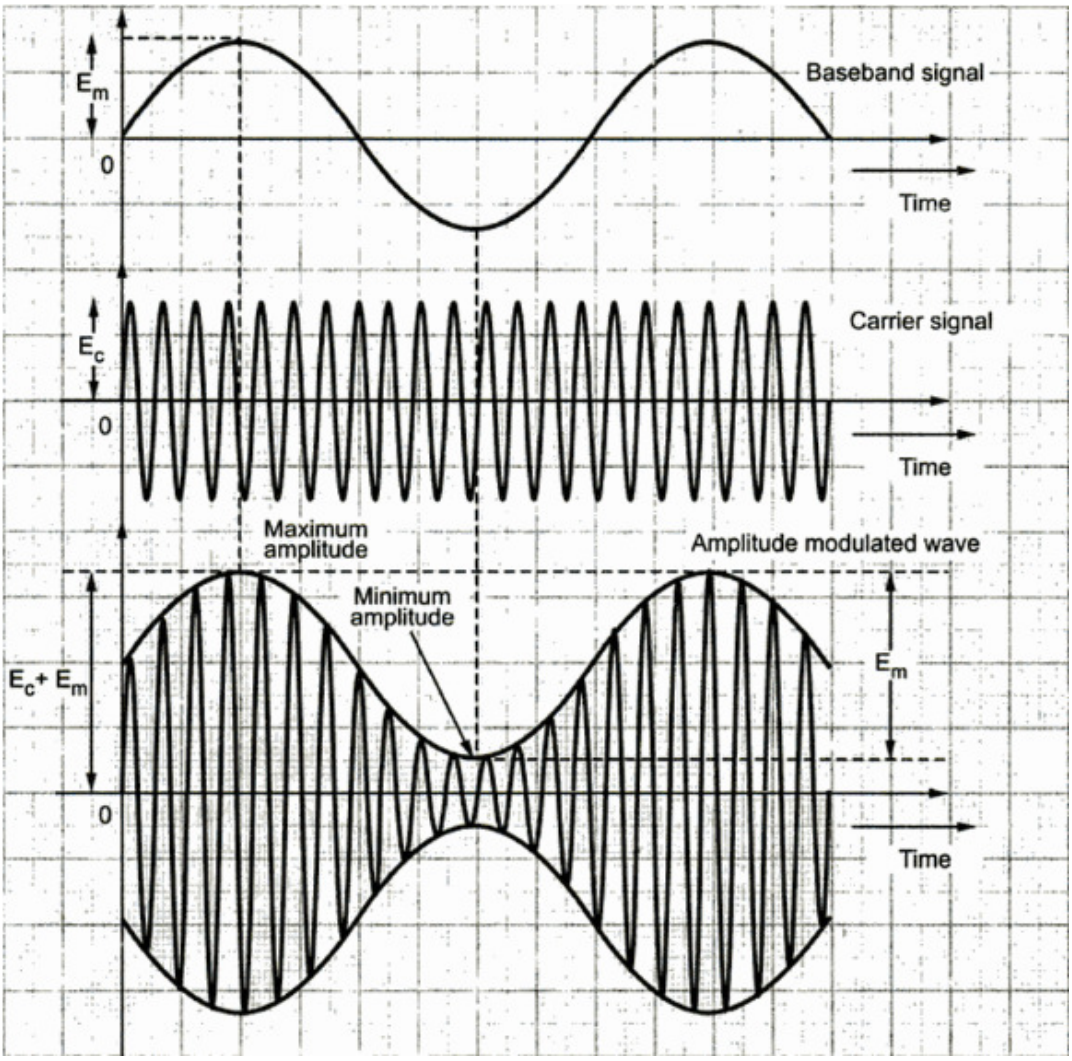


Joonis 1. Modulatsiooniviisid [4, p. 8]

3.1 Amplituudmodulatsioon

Amplituudmodulatsiooni, edaspidi AM, puhul on kandesignaali amplituud võrdeline moduleeriva signaali hetkväärtusega. AM-i iseloomustab kandesignaali olemasolu ka moduleeriva signaali puudumisel ning moduleeritud signaali mähisjoone sarnasus moduleeriva signaaliga. Klassikalise kandesignaaliga AM juhtumil on moduleeritud signaal $S_{AM}(t) = A\,[1 + m\,f(t)]\cos(\omega_C t)$, kus f(t) on moduleeriv signaal, ω_C on

faasinurk ning m on modulatsioonitegur, mis jääb vahemiku $0 \leq m \leq 1$. Signaali keskmine amplituud on võrdne suurusega A, mis on signaali suuruseks ka modulatsiooni puudumisel [3, p. 9]. Antud joonisel (Joonis 2) on kuvatud, kuidas madalama sagedusega siinussignaal moduleerib kõrgema sagedusega kandesignaali. Jooniselt on näha olukord, kus kandesignaali sagedus jääb modulatsiooni käigus konstantseks, kuid amplituud varieerub vastavalt moduleerivale signaalile [4, pp. 8-9].

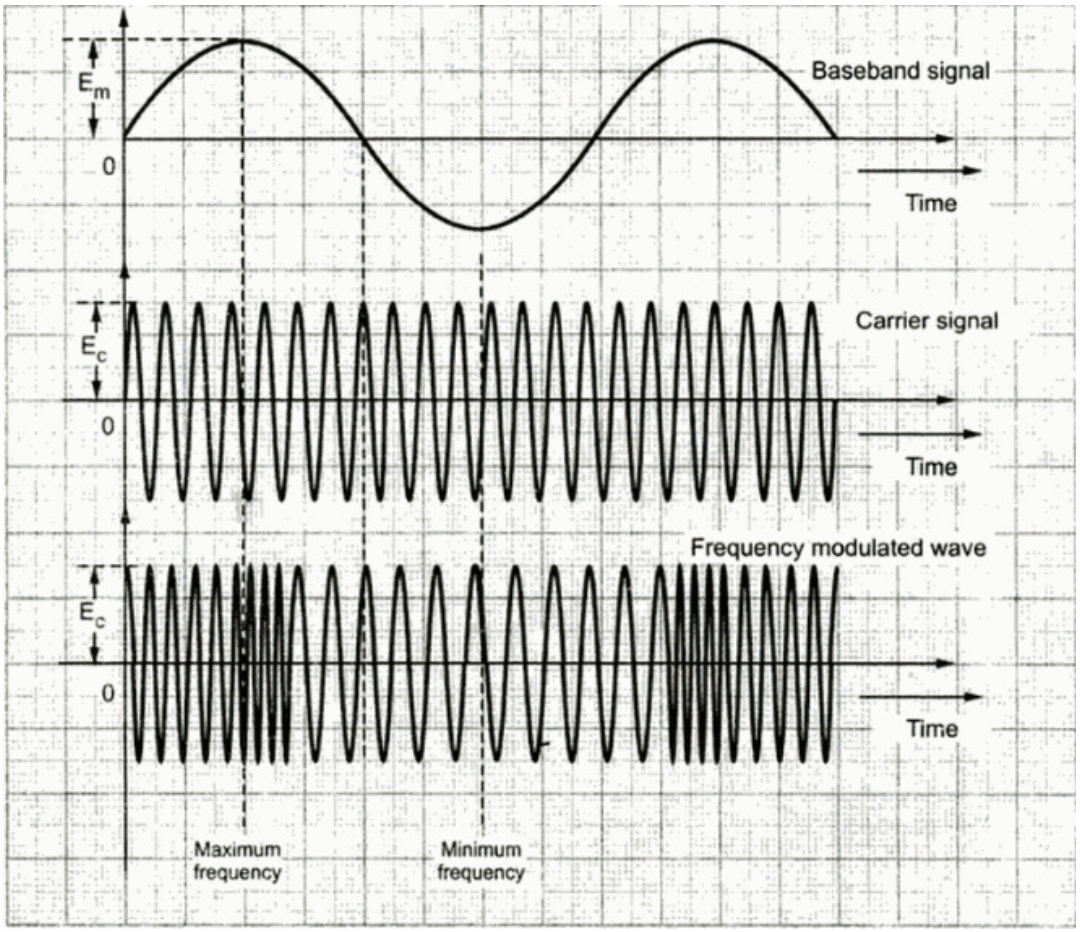


Joonis 2. Amplituudmodulatsioon [4, p. 17]

3.2 Sagedusmodulatsioon

Sagedusmodulatsiooni korral kandva signaali sagedus kasvab ning kahaneb vastavalt moduleeriva signaali väärtustele. Moduleeriva signaali maksimaalsele väärtusele vastab kõrgem sagedus ning minimaalse väärtuse korral on sagedus väikseim. Vastavalt

moduleeriva signaali amplituudi muutusele, varieerub kandesignaali sagedus üles ning allapoole kesksagedust f_C. Kandesignaali sageduse maksimaalset hälvet nimetatakse sagedusdeviatsiooniks [4, pp. 9-10]. Moduleerivast signaalist olenevalt kõigub hetkeline sagedus sagedusribas $f_C \pm f_\Delta$, kus f_Δ on deviatsioon. Sisendsignaali võimsus on alati võrdne $S_i = A^2/2$ [3, p. 58].



Joonis 3. Sagedusmodulatsioon [4, p. 10]

3.3 Sagedus- ja amplituudmodulatsiooni võrdlus

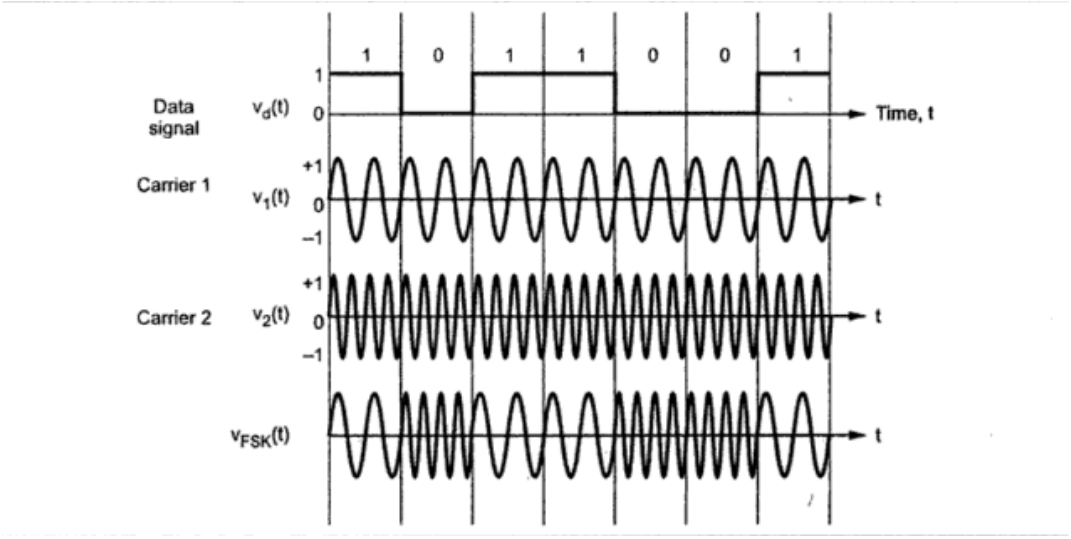
FM signaali amplituud on konstantne, mistõttu on ülekande võimsus samuti konstantne ning sõltumatu modulatsiooni sügavusest. AM korral modulatsiooni sügavus mõjutab ülekande võimsust. FM vastuvõtjad kasutavad amplituudi piirajaid, millega elimineeritakse mürast tingitud amplituudi varieerumist, mistõttu on FM signaali vastuvõtt müra suhtes immuunsem kui AM signaali vastuvõtmine. Sagedusmodulatsioon nõuab aga tunduvalt suuremat ribalaiust kui amplituudmodulatsioon [4, pp. 190-191].

Table 1. Sagedus- ja amplituudmodulatsiooni erinevused [4, p. 191]

Nr	FM	AM
1	FM võrrand: $v(t) = A\sin[\omega_C t + \omega\Delta f(t)]$	AM võrrand: $v(t) = E_C[1 + m\,f(t)]\sin\omega_C t$
2	Modulatsiooni indeks võib olla suurem või väiksem ühest	Modulatsiooni indeks on alati vahemikus 0 - 1
3	Kandva signaali amplituud ja võimsus on konstantsed ning sõltumatud modulatsiooni indeksist.	Ülekandevõimsus sõltub modulatsiooniindeksist $P_T = \mathrm{P_C}\left[1 + \frac{\mathrm{m}^2}{2}\right]$
4	Modulatsiooni indeks piiritleb külgriba paaride arvu FM signaalis	Kaks külgriba
5	Külgribade ja kandesignaali amplituud varieerub vastavalt modulatsiooni indeksile	Külgriba amplituud sõltub modulatsiooniindeksist ja on alati väiksem kui kandesignaali oma
6	Kandesignaali või külgriba amplituud võib olla 0	Külgriba amplituud ei ole kunagi 0, välja arvatud erijuhul kui moduleeriv signaal puudub
7	FM signaali ribalaius on võrdeline modulatsiooni indeksiga	AM signaali ribalaius on kaks korda suurem moduleeriva signaali omast

3.4 Sagedusmanipulatsioon

Sagedusmanipulatsiooni, järgneva peatüki jooksul FSK, korral muundatakse binaarse sisendsignaali $x(t)$ väärtused 0 ja 1 vastavalt signaalideks $s_0(t) = Acos(\omega_0 t)$ ja $s_1(t) = Acos(\omega_1 t)$. Sageduste erinevus $\omega_1 - \omega_0$ peab tagama nende eristatavuse vastuvõtul. Sageduse muutmiseks on kaks erinevat moodust, kus esimesel juhul kasutatakse kahte sõltumatut generaatorit ning toimub nende ümberlülitamine signaali väärtuste põhjal. Teise viisina juhitakse generaatorit sagedusmodulatsiooniga, mis tagab faasi pidevuse ning katkestuskohtade puudumise väljundsignaalis [3, pp. 106-107]. FSK meetodit kasutatakse aeglastes modemites, kus kandevõnkumise sagedust muudetakse erinevate sageduste vahel [5, pp. 14-15,20]. Autori poolt antud töö raames on kindlaks määratud sagedused, kus 1200 Hz sagedusele vastab biti väärtus 1 ning sagedusega 2200 Hz signaaliga edastatakse binaarset 0. Antud sageduste kasutamine võimaldab andmeid edastada läbi kõnekanali. FSK modulatsiooni puhul on müraga seotud probleeme vähem kui AM korral kuna vastuvõtja otsib kindlaid sageduse muutusi etteantud perioodi vältel [5, pp. 14-15,20].

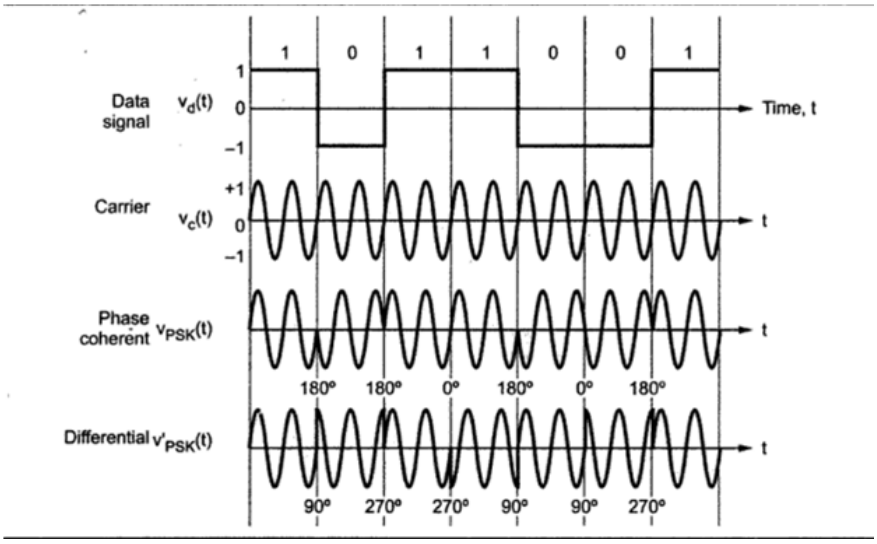


Joonis 4. FSK ajaline kuju [5, p. 14]

3.5 Faasmanipulatsioon

Faasmanipulatsiooni, edaspidi PSK, korral kasutatakse meetodit, kus signaali sagedus ning amplituud on konstantsed, kuid kandesignaali faasi muudetakse vastavalt ülekantavatele andmetele [5, p. 15]. Selline manipulatsioon on teostatav kandja A $\cos(\omega_C t)$ korrutamise teel polaarses formaadis sisendsignaaliga ±1, mis on kergesti

realiseeritav lülitireziimis seadmega [3, p. 103]. Joonisel 5 on kirjeldatud kahte erinevat faasinihet. Esimesel juhul kandev signaal edastab binaarset 0 ja 1, kuid faas erineb nende vahel 180° võrra [5, p. 15].



Joonis 5. Näide faasmanipulatsioonist [5, p. 15]

4 Signaal

Signaal on andmete esitluseks kasutatav füüsikaline suurus, mis võib olla looduslik või tehislik [6]. Antud töö raames käsitletakse analoog- ja digitaalsignaali. Signaali mõjutavad mürad, mis on juhuslik ja looduslik signaal, häired ning moonutused. Häireks peetakse tehisliku ning ebasoovitavat signaali ning moonutus näitab kuidas edastuskanal signaali muudab. Müra on defineeritav kui tahtmatu elektriline või magnetiline signaal, mis kahjustab esialgseid andmeid [7]. Vastuvõetud signaal koosneb mürast ning saatja poolt edastatud signaalist. Analoogseks võib lugeda kõiki pidevaid signaale, mis on mõõdetavad igal ajahetkel [8, p. 2].

4.1 Analoogsignaal

... on pidev nii ajas kui amplituudis ning siiamaani laialdaselt kasutusel raadios. Tehnikad, mida kasutatakse modulatsiooniks, muudavad signaali pealiskaudselt, et edastada andmeid läbi kanali. Analoogsignaali töötlemiseks ei ole vajalik kasutada spetsiaalset tarkvara, algoritme ega muundureid, vaid muundamine põhineb elektrilistele ning elektroonilistele seadmetel nagu takistid, transistorid ning mikroskeemid. Analoogsignaaliks reaalses maailmas on näiteks pinge, temperatuur, surve ning valguse intensiivsus [8, pp. 2-3].

4.2 Digitaalsignaal

Digitaalsignaaliks nimetatakse analoogsignaali, mis on konverteeritud digitaalsele kujule, kasutades arvkoodi [2, p. 135] . Digitaalsignaal on diskreetne nii ajas kui amplituudis ning edastatakse signaali, millel on piiratud arv erinevaid väärtusi [3, p. 7].

4.3 Nyquist´i kriteerium

Näitab kui suur peab olema minimaalne diskreetimissagedus, mis näitab diskreetide arvu sekundis. Kõige üldisemal kujul peab diskreetimissagedus f_s olema vähemalt kaks korda suurem kui suurema sagedusega signaalikomponendi sagedus f_{max}. Kui aeg on antud sekundites siis ribalaius on väljendatud hertsides [9, p. 187]. Kanalis võimaliku

maksimaalse edastuskiiruse arvutamiseks võib kasutada Shanoni valemit $C = B\ \log_2(1 + \frac{S}{N})$, kus B on kanali ribalaius ning $\frac{S}{N}$ on signaal-müra suhe [10].

Bell 202

Bell 202 kommunikatsiooni standardi kohaselt luuakse digitaalne signaal kahe sageduse abil. Sagedusele 1200 Hz vastab binaarne 1 ning sagedusele 2200 Hz vastab binaarne 0. Nimetatud standardi kohaselt on andmeedastus kiirus 1200 bitti sekundis [11, p. 243]. Autor on oma töös lähtunud samuti antud standardist ning kasutab vastavaid sagedusi andmete edastamisel. Järgnevas tabelis (Tabel 2) on välja toodud olemasolevad Bell standardid.

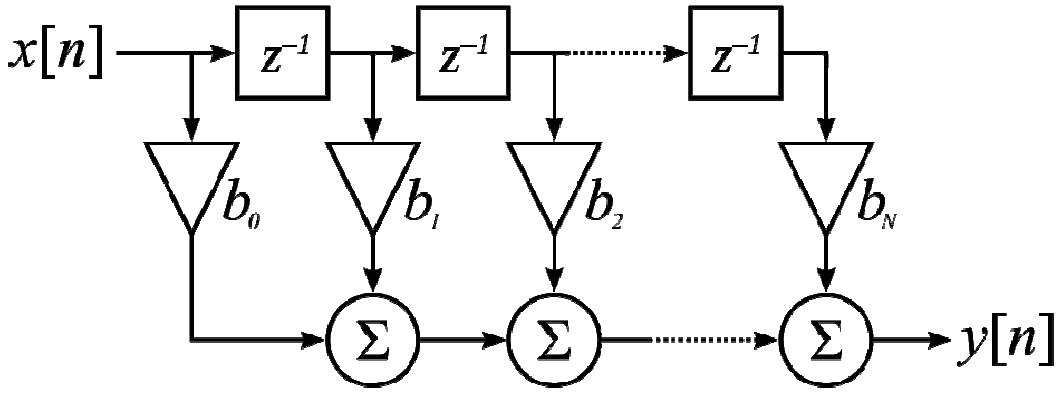
Table 2. Modemite standardid [11, p. 129]

CCITT	**Bell**	**Kiirus (BPS)**	**Formaat**	**Modulatsioon**	**Ühilduvus**
v.21	103/113	Kuni 300	Async	FSK	-
v.22	212 A	1200/600	Async/sync	PSK	Jah 1200 bps
v.23	202	1200/600/75*	Async	FSK	Tavaliselt
v.26	201 B	2400/75*	Async/sync	PSK	Jah
v.26 b	201 C	2400/1200/75*	Sync	PSK	Jah 2400 bps
v. 27	208 B	4800/75*	Sync	PSK	-
v.27 b	208 B	4800/2400/75*	Sync	PSK	-
v. 27 t	208 A	4800/2400/75*	Sync	PSK	-
v. 29	209	9600/7200/4800	Sync	QAM	-

*75 bps tagasiside kanal kasutab asünkroonseid andmeid ja FSK modulatsiooni

4.4 Lõpliku impulsskajaga filter

FIR filtrid on kasutusel digitaalse signaali töötlemisel. Disainitava filtri amplituudikarakteristik on tavaliselt kirjeldatud sagedusvahemike abil, kus teatud sagedusvahemikus olevad signaali komponendid peavad alles jääma muutmata kujul ning teatud signaalid tuleb eemaldada. Ideaalse madalpääsfiltri sageduskarakteristik D(ω) on määratletud $D = \begin{cases} 1, kui\ \omega \in [0, \omega_c) \\ 0, kui\ \omega \in (\omega_c, \pi] \end{cases}$, kus ω_c on filtri lõikesagedus. Seosest on näha, et filter laseb läbi kõik sagedused, mis on väiksemad lõikesagedusest [12]. Antud meetod on kasutusel paljudes mikroprotsessorites, kus ressursid on piiratud ning töökiirus oluline [13].



Joonis 6. FIR filtri plokkskeem [14]

5 Rakendus

Autor valis antud rakenduse loomise vastavalt tekkinud ideele koostada prototüüp tarkvaralisele modemile, millele leida tulevikus kasutust. Antud tarkvara mõte on testida, kas kasutaja poolt sisestatud andmed on võimalik kuvada vastuvõtjas viisil, et sisend edastatakse esmalt läbi audiokaabli raadiojaama, mis omakorda edastab andmed läbi raadiokanali teisele käsijaamale. Vastuvõtvast jaamast liigub sisendi põhjal genereeritud signaal läbi audiokaabli arvutisse, mis kuvab sisestatud andmeid. Antud rakendus ei sisalda veatuvastus algoritme, mistõttu on esialgu tegemist prototüübiga ning sobilike vahendite ja meetodite valimise ning katsetamisega.

Saatja moduleerib andmed vastavalt Bell 202 standardile, et andmeedastus oleks võimalik läbi kõneedastuseks mõeldud raadiokanali. Antud standardi kasutamine tagab töökindluse, et andmed mahuvad ka raadiokanalisse. Olemasolevatele standarditele toetumine võimaldab tulevikus erinevaid seadmeid ühildada. Töö käigus kasutati Lisas 3 olevat lähtekoodi kontrollimaks genereeritud helifaile graafiliselt. Antud lähtekoodi kasutamise eesmärgiks oli veenduda sageduste korrektses vaheldumises.

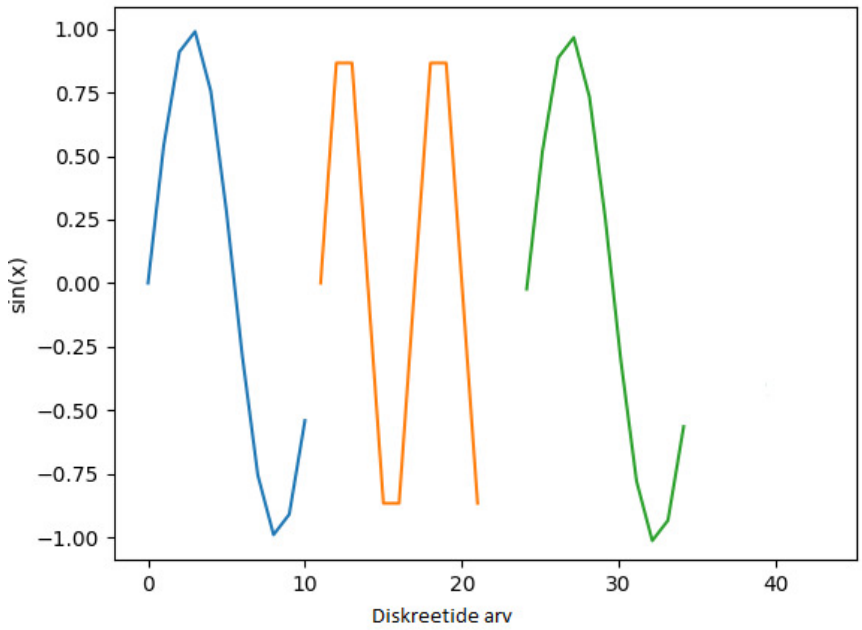
5.1 Saatja

Saatjapoolne modem (Lisa 1) küsib esmalt kasutajalt sisendandmeid. Kui sisendit ei ole ei toimu ka andmevahetust. Kasutaja sisestab andmed kasutades klaviatuuri. Andmed konverteeritakse vastavalt ASCII tabelile (Joonis 7), mis on standardiks andmete digitaalsel kujul esitamiseks arvutile mõistetavas formaadis, ning lisatakse listi. Seejärel luuakse kaks erinevat signaali vastavalt Bell 202 standardile, kus sagedusele 1200 Hz vastab binaarne 1 ning 2200 Hz vastab 0 [10]. Joonisel 8 on kujutatud näidis moduleeritud signaal, mis loodi digitaalsignaali „101“ põhjal. Signaalide jaoks loob tarkvara kaks massiivi vastavalt valemitele $s_1(\mathrm{t}) = \sin(2\,\pi \frac{1200}{13200} t)$ ja $s_0(t) = \sin(2\,\pi\ \frac{2200}{13200}\ t)$, kus t on diskreetne aeg. Järgnevalt loeb tarkvara eelpool genereeritud listi, mis koostati sisendi põhjal ning vastavalt sisule edastab massiivis oleva signaali heliväljundisse. Jooniselt 9 on näha signaali ajaline kuju, millel seisneb, et antud 3 biti edastamiseks kulus ligikaudu 0,2 millisekundit.

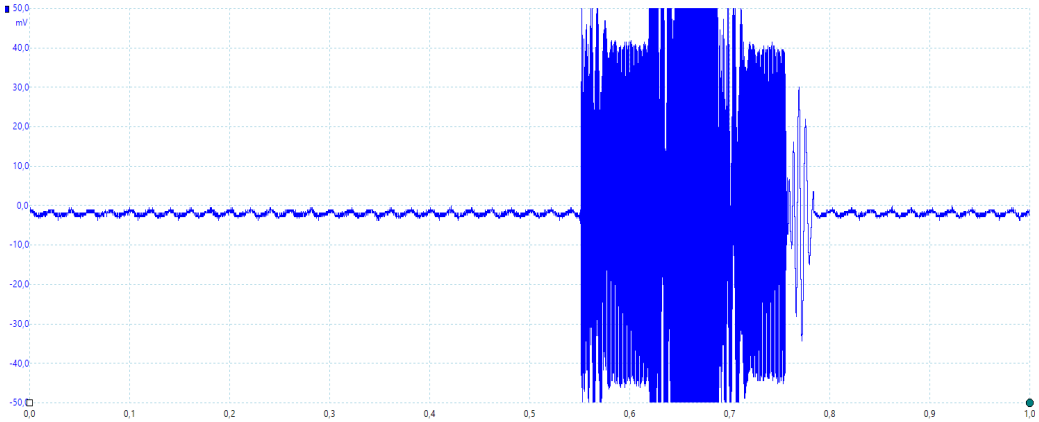
ASCII Code: Character to Binary

0	0011 0000	O	0100 1111	m	0110 1101
1	0011 0001	P	0101 0000	n	0110 1110
2	0011 0010	Q	0101 0001	o	0110 1111
3	0011 0011	R	0101 0010	p	0111 0000
4	0011 0100	S	0101 0011	q	0111 0001
5	0011 0101	T	0101 0100	r	0111 0010
6	0011 0110	U	0101 0101	s	0111 0011
7	0011 0111	V	0101 0110	t	0111 0100
8	0011 1000	W	0101 0111	u	0111 0101
9	0011 1001	X	0101 1000	v	0111 0110
A	0100 0001	Y	0101 1001	w	0111 0111
B	0100 0010	Z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	.	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	;	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	I	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(	0010 1000
N	0100 1110	l	0110 1100	)	0010 1001
				space	0010 0000

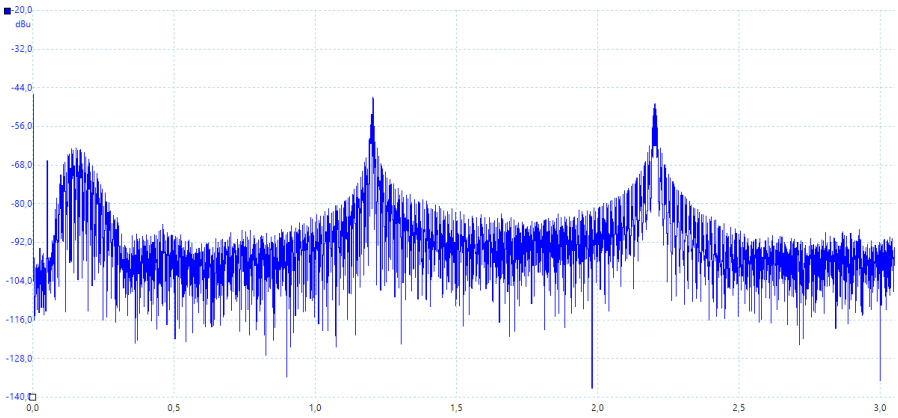
Joonis 7. ASCII tabel [15]



Joonis 8 Moduleeritud signaali graafiline kuju



Joonis 9 Moduleeritud signaali ajaline kuju



Joonis 10 Teoreetilise signaali spektripilt

5.2 Vastuvõtja

Vastuvõtja poolel kuulab programm (Lisa 2) helisisendit ning kontrollib, kas andmeedastus on alanud. Kui edastus toimub, kuid tegemist on vaikusega ehk signaali maksimaalne väärtus ei ületa etteantud väärtust, klassifitseeritakse signaal mitte vajalikuks. Kui andmeedastus teatud kindla aja jooksul ei alga, lõpetatakse kuulamine. Antud lahendus sobib prototüübi korral, kuid tulevikus reaalselt töötava seadme korral peaks antud lahendust modifitseerima viisil, et kuulamine toimuks pidevalt. Andmeid loetakse teatud pikkusega segmentide kaupa ning lisatakse listi. Antud listi ei lisata müra, kuid enne ning peale andmeid lisatakse heli mõistes vaikust selleks, et erinevad meediaedastus programmid suudaksid faili ette mängida. Antud programm genereerib ka helifaili vastavalt etteantud nimetusega, kuhu lisab varasemalt loodud listi ja paneb paika kanalite arvu ning ribalaiuse.

Järgnevalt arvutab programm (Lisa 2) signaali sagedused, mille põhjal tehakse järeldus, milline andmebitt saadeti. Sageduste leidmiseks teostab tarkvara Fourier teisenduse vastava Pythoni funktsiooniga ning võtab sellest absoluutväärtuse ja leiab tulemuse ruudu. Fourier teisendus teisendab funktsiooni sagedusteks mida on võimalik esitada siinuste ja koosinuste summana [16]. Järgnevalt arvutatakse eelneva tulemuse maksimaalne väärtus ning interpolatsioon, mis korrutatakse diskreetimissagedusega ja jagatakse segmendi pikkusega. Sageduste arvutamisel püütakse järgida FIR põhimõtet ning andmebitt otsustatakse vastavalt Bell 202 standardile. Kui sagedus on vahemikus 1000 Hz – 1400 Hz lisatakse listi väärtus „1“, sagedusvahemiku 2000 Hz – 2400 Hz

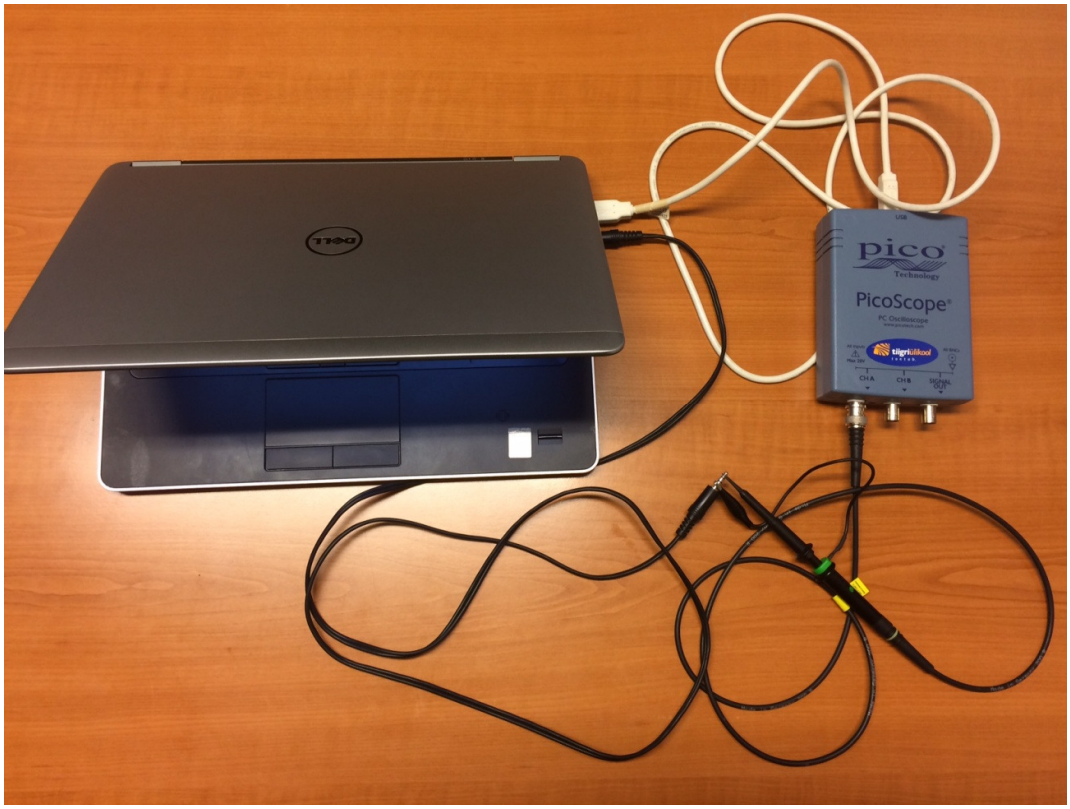
korral aga väärtus „0“. Järgnevalt dekodeeritakse genereeritud listi ning vastavalt ASCII tabelile (Joonis 7) kuvatakse andmed ekraanile.

5.3 Katsed / mõõtmised

Autor teostas mõõtmisi selleks, et veenduda püstitatud eesmärgi täitmises ning loodud tarkvara töötamises. Katsetes kasutati PicoScope 2002 ostsilloskoopi, mis oli ühendatud arvutiga. Graafikute saamiseks oli tarkvara PicoScope 6, millega kontrolliti sageduste vastavust standarditele. Kasutusel oli 2 käsiraadiojaama Nissei N888U ning tarkvara, mis loodi Python keeles. Ühendus arvuti ning raadiojaamade vahel loodi audiokaabliga.

5.3.1 Katse I

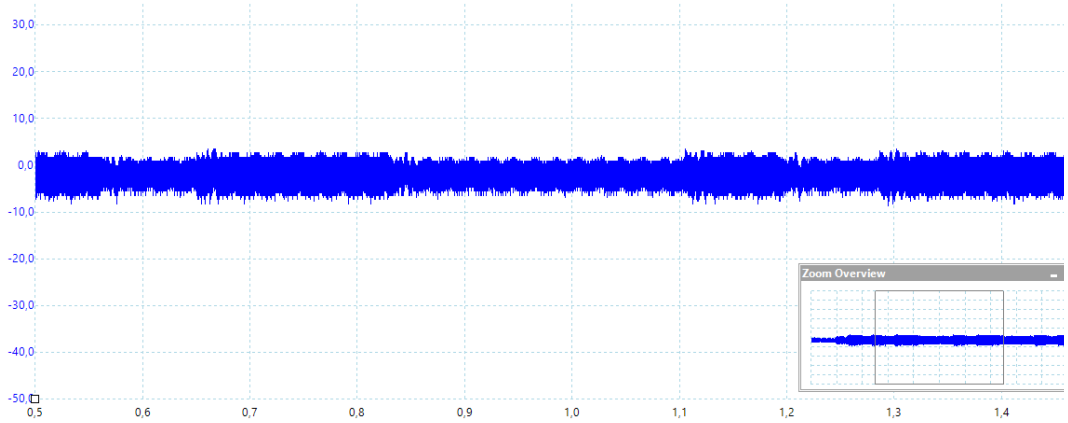
Loodud süsteemi testimiseks koostati skeem (Joonis 11), millesse kuulus autori arvuti ja ostsilloskoop, mis olid ühendatud audiokaabliga. Katseliselt saadeti programmi (Lisa 1) tekst „TtuTallinn“, mis vastab ASCII tabelis (Joonis 7) koodidele, mis on kuvatud joonisel 12. Kasutades PicoScope 2205 mõõteseadet on mõõtetulemuseks spektripilt (Joonis 14), millelt on loetavad sagedused 1200 Hz ning 2200 Hz. Joonisel 13 on näha segmentide edastamiseks kuluv aeg ning võib lugeda, et ühe segmendi edastamiseks kulub ligikaudu 0,1 millisekundit.



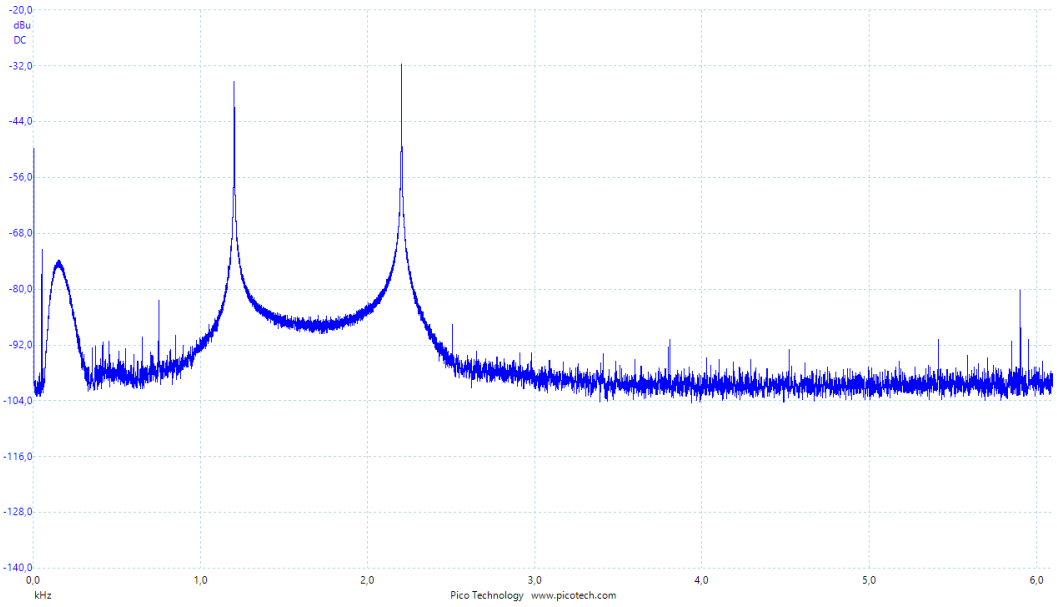
Joonis 11. Katse I skeem

```
>>> ========================= RESTART ========================
>>>
Insert data: TtuTallinn
1010100
1110100
1110101
1010100
1100001
1101100
1101100
1101001
1101110
1101110
Insert filename: Ttu.wav
>>>
```

Joonis 12. Katse I sisendandmed



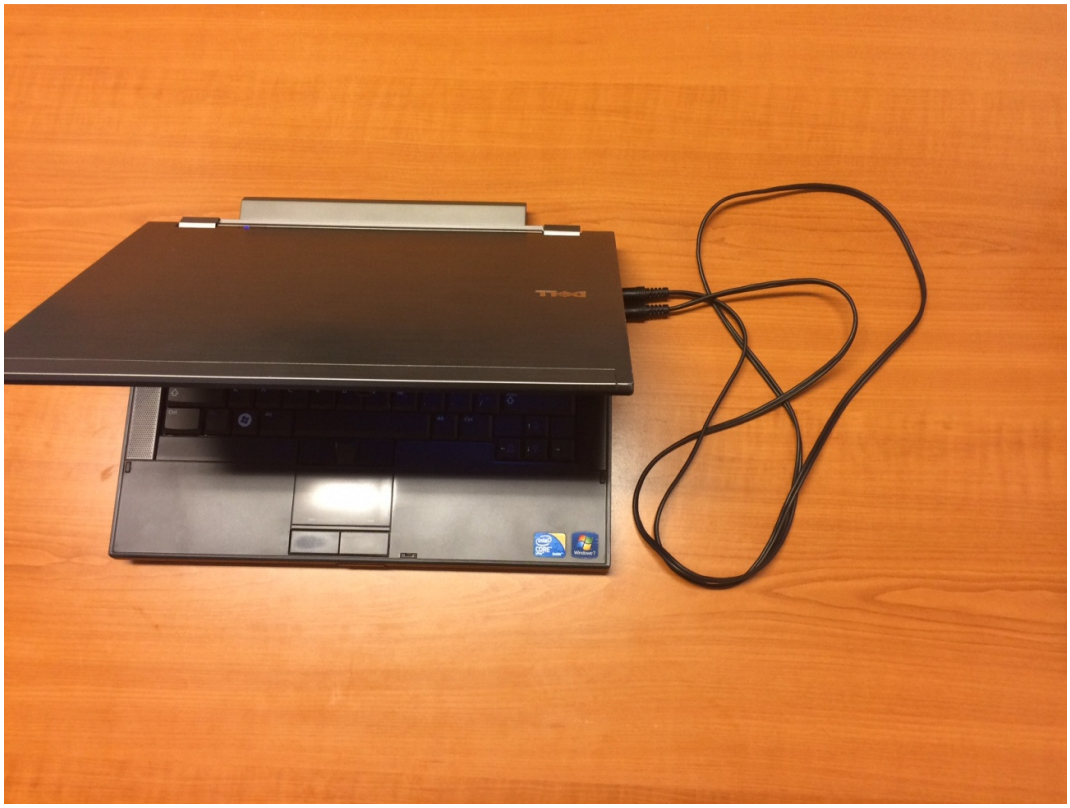
Joonis 13. Katse I signaal



Joonis 14. Katse I ostsilloskoobi spektripilt

5.3.2 Katse II

Katse II teostati skeemi (Joonis 15) põhjal, kus on ühendatud arvuti helisisend ja –väljund audiokaabliga. Katseliselt saadeti programmi (Lisa 1) andmed „TtuTallinn“ ja „ttu.wav“, mis on kujutatud Joonisel 16. Antud sisendite põhjal loodud helifail saadeti heliväljundisse ning kasutades programmi (Lisa 2) kuulati helisisendit ning loodi helifail „ttu.wav“. Eelnevalt nimetatud lähtekood avas faili ning tulemuseks saadi andmed, mis on kujutatud Joonisel 17. Antud helifaili graafiliselt vaadates, lähtekoodiga (Lisa 3), on näha Jooniselt 18, et toimub üleminek ühelt sageduselt teisele. Vastavalt väljundandmete õigsusele võib järeldada, et signaal vastab sagedustele 1200 Hz ning 2200 Hz.



Joonis 15. Katse II skeem

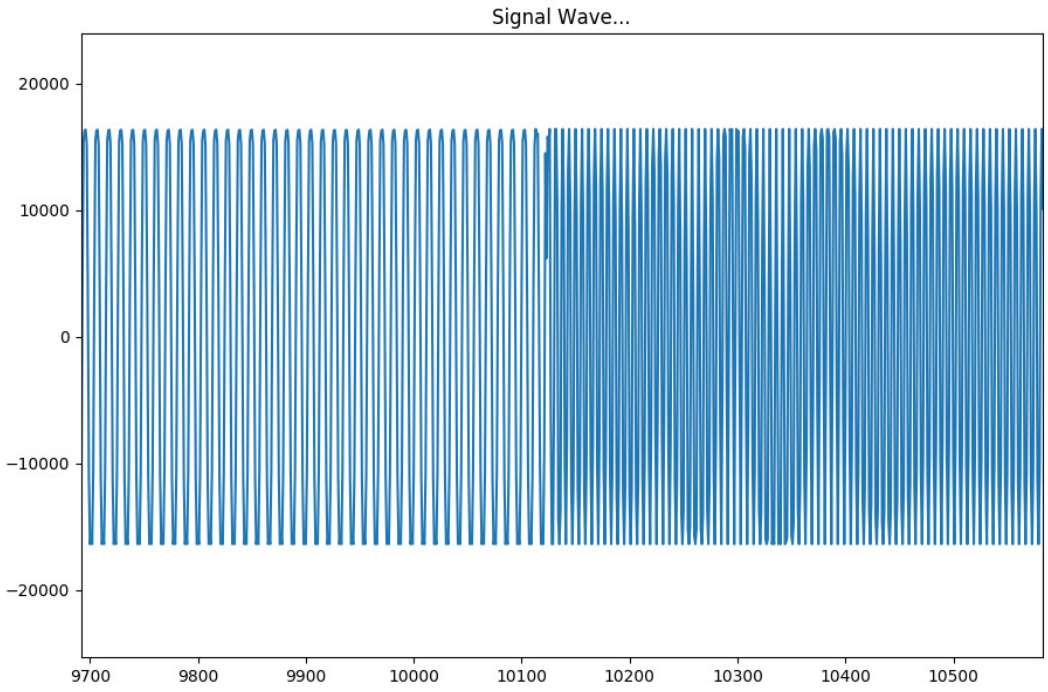
```
>>> ====================== RESTART ======================
>>>
Insert data: TtuTallinn
1010100
1110100
1110101
1010100
1100001
1101100
1101100
1101001
1101110
1101110
Insert filename: ttu.wav
>>>
```

Joonis 16. Katse II sisendandmed

```
>>> ====================== RESTART ======================
>>>
Insert filename: ttu.wav

1010100
1110100
1110101
1010100
1100001
1101100
1101100
1101001
1101110
1101110
['T', 't', 'u', 'T', 'a', 'l', 'l', 'i', 'n', 'n']
```

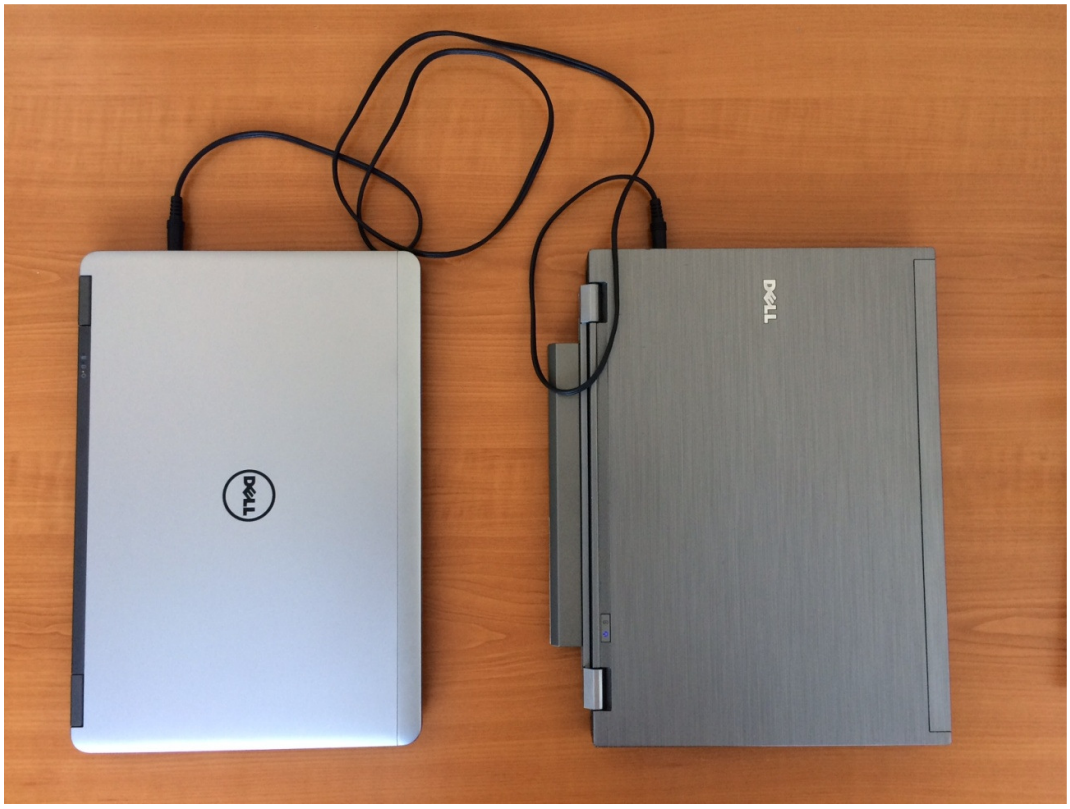
Joonis 17. Katse II väljundandmed



Joonis 18. Katse II signaal graafiliselt

5.3.3 Katse III

Katse III jaoks koostas autor ühenduse skeemi (Joonis 19) põhjal, kus kasutades audiokaablit ühendati kokku kaks arvutit. Programmi (Lisa 1) edastati andmed, mis on kuvatud Joonisel 20, millelt on näha, et sisendiks valiti „TtyTallinn“ ning faili nimeks valiti „tty.wav“. Joonisel 21 on kirjeldatud väljundandmed, mis saadi kasutades lähtekoodi Lisas 2. Antud jooniste põhjal saab teha järelduse, et andmevahetus kahe masina vahel oli edukas ning sisendina edastatud andmed jõudsid ilma kadudeta väljundisse. Joonisel 22 on kuvatud edastatud signaal graafiliselt, kasutades Lisas 3 olevat lähtekoodi, millelt on loetav müra olemasolu kanalis ja edastatav signaal.



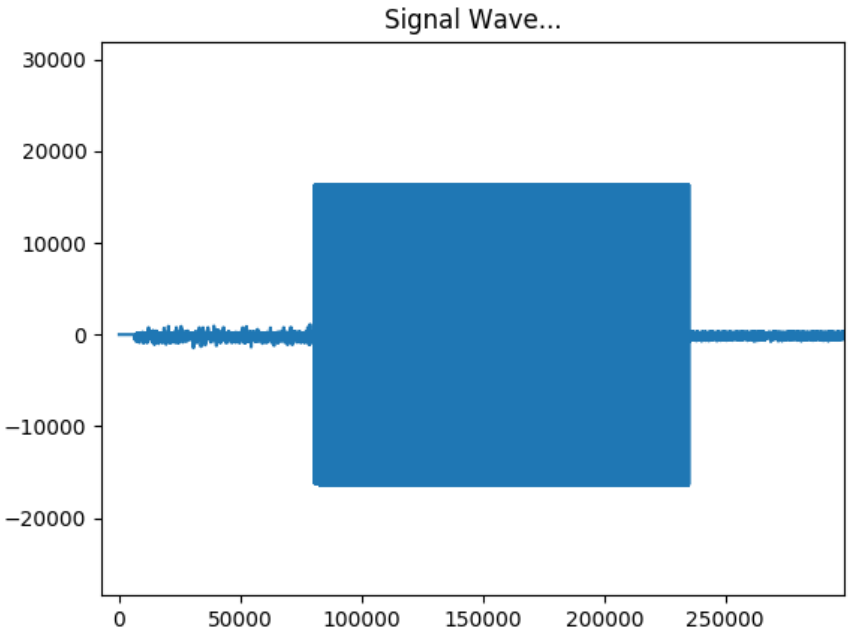
Joonis 19. Katse III skeem

```
>>> ================================ RESTART ================================
>>>
Insert data: TtyTallinn
1010100
1110100
1111001
1010100
1100001
1101100
1101100
1101001
1101110
1101110
Insert filename: tty.wav
>>>
```

Joonis 20. Katse III sisendandmed

```
>>> ======================== RESTART ========================
>>>
Start recording...
Recording done, start opening file
1010100
1110100
1111001
1010100
1100001
1101100
1101100
1101001
1101110
1101110
['T', 't', 'y', 'T', 'a', 'l', 'l', 'i', 'n', 'n']
File opening done
>>>
```

Joonis 21. Katse III väljundandmed



Joonis 22. Katse III signaal graafiliselt

5.3.4 Katse IV

Viimase katsena koostas autor skeemi (Joonis 23) kuhu kuulus kaks käsiraadiojaama, kaks audiokaablit ning kaks arvutit. Katse eesmärgiks oli sisestada ühte arvutisse andmed, mis edastatakse käsijaamale. Raadio omakorda edastab saadud informatsiooni läbi raadiokanali teisele jaamale, mis võtab andmed vastu ning saadab need arvutisse. Vastuvõtja poolses arvutis olev programm (Lisa 2) kuulab helisisendit ning salvestab saadud signaali põhjal helifaili. Programmiga (Lisa 2) loetakse antud faili ning kuvatakse sisu arvutiekraanile. Sisendiks valiti „TtyTallinn“, mis on kuvatud Joonisel 24. Raadiojaamad olid seadistatud kanalile „CH12“. Andmed saadeti läbi raadiojaamade ning väljundiks teises arvutis oli „TtyTallinn“, mis on nähtav Jooniselt 25. Andmevahetuse signaal on esitatud Joonisel 26, kus on selgelt eristatav edastatav signaal ning kanalis tekkiv müra. Antud sisendi ning väljundi võrdlemisel jõutakse tulemuseni, mis täidab püstitatud eesmärki, kus andmevahetus on toimunud läbi tarkvaralise modemi ning raadiokanali.

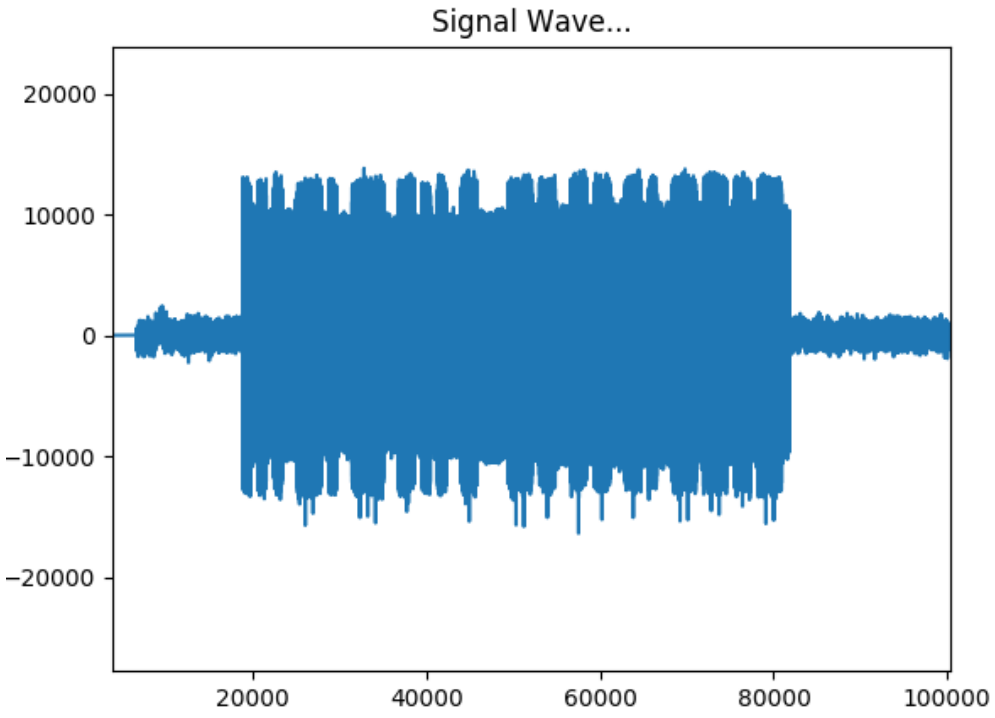
Joonis 23. Katse IV skeem

```
>>> ================================ RESTART ================================
>>>
Insert data: TtyTallinn
1010100
1110100
1111001
1010100
1100001
1101100
1101100
1101001
1101110
1101110
```

Joonis 24. Katse IV sisendandmed

```
>>> ========== RESTART ==========
>>>
Start recording...
Recording done, start opening file
1010100
1110100
1111001
1010100
1100001
1101100
1101100
1101001
1101110
1101110
['T', 't', 'y', 'T', 'a', 'l', 'l', 'i', 'n', 'n']
File opening done
```

Joonis 25. Katse IV väljundandmed



Joonis 26. Katse IV signaal graafiliselt

Kokkuvõte

Bakalaureusetöö eesmärgiks oli edastada kasutaja poolt sisestatud andmed läbi tarkvaralise modemi vastuvõtjale. Lisaks soovis autor välja selgitada, millisel viisil on edastus võimalik ning milliseid vahendeid kasutada.

Töö esimeses etapis uuriti teoreetilisi võimalusi ning standardeid, millega tuleks arvestada tarkvara loomisel, et kindlustada andmevahetus. Olemasolevate reeglite järgimine tagab tarkvara ning seadmete ühilduvuse.

Töö teises etapis loodi tarkvara, millega pidada andmevahetust saatja ja vastuvõtja vahel ning teostati mõõtmisi ja katseid, mille tulemused on antud projektis kajastatud. Autori hinnangul võib katse IV tulemusi pidada antud töö kõige märkimisväärsemateks kuna andmeid edastati läbi erinevate kanalite ning sisendina antud „TtyTallinn“ jõudis ilma veatuvastuse ja tagasisideta korrektselt väljundisse.

Tulemite analüüsimine kinnitab andmete eduka edastamise läbi tarkvaralise modemi. Lisaks modemile vahendati informatsiooni läbi audiokaabli ja raadiokanali, mida võib autor, lähtudes katsetulemustest, pidada edukaks.

Antud tööd oleks võimalik edasi arendada luues saatjas andmete krüpteerimise ning vastuvõtjas dekrüpteerimise. Lisades tarkvarale veatuvastuse või kõneedastuse. Pythoni asendamine arvuti ja nutiseadme rakendusega.

Saavutatud resultaatide põhjal, võib autor järeldada, et esialgselt püstitatud eesmärk on täidetud ning katsete ja mõõtmiste analüüsimine kinnitavad andmevahetuse edukust, mille tulemusena jõuti töötava tarkvaralise prototüübini.

Kasutatud kirjandus

[1] C. Georgopoulos, „Interface Fundamentals in Microprocessor-Controlled Systems,“ D. Reidel Publishing Company, Dordrecht, 1985.

[2] Atlantic, „Encyclopedia Of Information Technology,“ Atlantic publishers & distributors LTD, 2007.

[3] A. Meister, Modulatsioon, Tallinn: TTÜ kirjastus, 1999.

[4] A.P.Godse ja U.A.Bakshi, „Communication Engineering,“ Technical Publications Pune, 2008.

[5] I.A.Dhotre ja V.S.Bagad, „Data Communication And Networking,“ Technical Publications Pune, 2005.

[6] „Wikipedia,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Signal. [Kasutatud 25 03 2017].

[7] „Omega,“ [Võrgumaterjal]. Available: https://www.omega.com/literature/transactions/volume2/analogsignal2.html. [Kasutatud 20 3 2017].

[8] J. J. Li Tan, „Fundamentals of Analog and Digital Signal Processing,“ AuthorHouse, Bloomington, 2007.

[9] M. H. Weik, „Fiber Optics Standard Dictionary,“ Van Nostrand Reinhold, Virginia, 1989.

[10] [Võrgumaterjal]. Available: http://www.gaussianwaves.com/2008/04/channel-capacity/. [Kasutatud 11 04 2017].

[11] W. Boyes, „Instrumentation Reference Book,“ Elsevier Inc, Burlington, 2010.

[12] „Digitaalne signaalitöötlus: valitud FIR filtrite disainimine,“ [Võrgumaterjal]. Available: https://www.ttu.ee/public/t/Tehnomeedikum/Instituudid/Biomeditsiinitehnika_instituut/yhisope/dsp.pdf. [Kasutatud 05 04 2017].

[13] „dspGuru,“ [Võrgumaterjal]. Available: https://dspguru.com/dsp/faqs/fir/basics/. [Kasutatud 20 03 2017].

[14] „Wikipedia,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Finite_impulse_response. [Kasutatud 15 03 2017].

[15] [Võrgumaterjal]. Available: http://inf14lorenzoms.weebly.com/uploads/3/8/1/5/38154377/3634458_orig.gif 16.05.2017. [Kasutatud 08 04 2017].

[16] „Wikipedia,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Fourier_transform. [Kasutatud 18 03 2017].

[17] „Stackoverflow,“ [Võrgumaterjal]. Available: http://stackoverflow.com/questions/2648151/python-frequency-detection. [Kasutatud 22 03 2017].

[18] „Stackoverflow,“ [Võrgumaterjal]. Available:

http://stackoverflow.com/questions/892199/detect-record-audio-in-python. [Kasutatud 22 03 2017].

[19] „Stackoverflow,“ [Võrgumaterjal]. Available: http://stackoverflow.com/questions/18625085/how-to-plot-a-wav-file. [Kasutatud 18 03 2017].

[20] „Wikipedia,“ [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/Modulatsioon_(%C3%BClekandetehnika). [Kasutatud 16 3 2017].

[21] R. E. Blahut, „Modem Theory: An Introduction to Telecommunications,“ Cambridge University Press, New York, 2010.

Lisa 1 – Saatjas kasutatav lähtekood

```
import winsound,threading, wave, pyaudio, time, math, struct
import numpy as np

binaryData=[]
wavBinaryData=[]
CHUNK = 1024
Channels = 1
Rate = 1200
Format = 2
Frequency = 13200
volume=0.2
pyAudio = pyaudio.PyAudio()
start = 0
frames = []


inputText = input(“Insert data: “)

for i in range(0,len(inputText)):
    eachWord = inputText[i]

    for i in range(0, len(eachWord)):
        eachLetter = bin(ord(eachWord[i])).replace(“0b”, “”)
        for i in range(0,len(eachLetter)):
            binaryData.append(int(eachLetter[i]))


samplesForOne =
(np.sin(2*np.pi*np.arange(13200)*1200/Frequency)).astype(np.float32)
samplesForZero =
(np.sin(2*np.pi*np.arange(13200)*2200/Frequency)).astype(np.float32)

for i in range(0,9600):
    samplesForOne = np.delete(samplesForOne, (0), 0)

for i in range(0,9600):
    samplesForZero = np.delete(samplesForZero, (0), 0)


stream = pyAudio.open(format=pyaudio.paFloat32,
                channels=1,
                rate=13200,
                output=True)
```

```
for i in range(0, len(binaryData)):
    if int(binaryData[i]) == 1:
        threading.Thread(target=stream.write(volume*samplesForOne)).start()
    else:
        threading.Thread(target=stream.write(volume*samplesForZero)).start()

stream.stop_stream()
stream.close()
pyAudio.terminate()

##GENERATE WAV
def generateSoundFile():
    frameRate = 13200
    Type = “NONE”
    Compress = “not compressed”

    for i in range(0,len(binaryData)):
        if int(binaryData[i]) == 0:
            for x in range(2200):
                wavBinaryData.append(math.sin(2*math.pi*2200*(x/Frequency)))
        if int(binaryData[i]) == 1:
            for x in range(1200):
                wavBinaryData.append(math.sin(2*math.pi*1200*(x/Frequency)))
    file = input(‘Insert filename: ‘)
    wavFile = wave.open(file, ‘wb’)
    wavFile.setparams((Channels,Format,frameRate,Rate,Type,Compress))

    for sample in wavBinaryData:
        amplitude = 1200
        wavFile.writeframes((struct.pack(‘h’, int(sample * amplitude / 2))))

    wavFile.close()
    playWav(file)
## WAV END

def playWav(file):
    winsound.PlaySound(file, winsound.SND_FILENAME)

generateSoundFile()
```

Lisa 2 – Vastuvõtjas kasutatav lähtekood

```
from sys import byteorder
from array import array
from struct import pack
import pyaudio, binascii, wave
import numpy as np

THRESHOLD = 500
CHUNK_SIZE = 1024
FORMAT = pyaudio.paInt16
RATE = 13200
chunk = 2200
stringData =  []
readData = []

def is_silent(sound_data):
    return max(sound_data) < THRESHOLD

def normalize(sound_data):
    MAXIMUM = 16384
    times = float(MAXIMUM)/max(abs(i) for i in sound_data)
    r = array('h')
    for i in sound_data:
        r.append(int(i*times))
    return r

def trim(sound_data):
    "Trim the blank spots at the start and end"
    def _trim(sound_data):
        snd_started = False
        r = array('h')

        for i in sound_data:
            if not sound_data and abs(i)>THRESHOLD:
                sound_data = True
                r.append(i)

            elif sound_data:
                r.append(i)
        return r

    sound_data = _trim(sound_data)
    sound_data.reverse()
    sound_data = _trim(sound_data)
    sound_data.reverse()
    return sound_data

def add_silence(sound_data, seconds):
```

```
    r = array('h', [0 for i in range(int(seconds*RATE))])
    r.extend(sound_data)
    r.extend([0 for i in range(int(seconds*RATE))])
    return r

def record():
    p = pyaudio.PyAudio()
    stream = p.open(format=FORMAT, channels=1, rate=RATE,
        input=True, output=True,
        frames_per_buffer=CHUNK_SIZE)
    num_silent = 0
    snd_started = False
    r = array('h')

    while 1:
        sound_data = array('h', stream.read(CHUNK_SIZE))
        if byteorder == 'big':
            sound_data.byteswap()
        r.extend(sound_data)
        silent = is_silent(sound_data)
        if silent and snd_started:
            num_silent += 1
        elif not silent and not snd_started:
            snd_started = True
        if snd_started and num_silent > 100:
            break

    sample_width = p.get_sample_size(FORMAT)
    stream.stop_stream()
    stream.close()
    p.terminate()

    r = normalize(r)
    r = trim(r)
    r = add_silence(r, 0.5)
    return sample_width, r

def record_to_file(path):
    sample_width, data = record()
    data = pack('<' + ('h'*len(data)), *data)
    wf = wave.open(path, 'wb')
    wf.setnchannels(1)
    wf.setsampwidth(sample_width)
    wf.setframerate(RATE)
    wf.writeframes(data)
    wf.close()

def openFile(filename):
    wf = wave.open(filename, 'rb')
    swidth = wf.getsampwidth()
```

```
    RATE = wf.getframerate()
    window = np.blackman(chunk)
    p = pyaudio.PyAudio()
    stream = p.open(format =
                    p.get_format_from_width(wf.getsampwidth()),
                    channels = wf.getnchannels(),
                    rate = RATE,
                    output = True)
    data = wf.readframes(chunk)

    while len(data) == chunk*swidth:
        stream.write(data)
        indata = np.array(wave.struct.unpack("%dh"%(len(data)/swidth),\
                                             data))*window
        fftData=abs(np.fft.rfft(indata))**2
        which = fftData[1:].argmax() + 1
        if which != len(fftData)-1:
            y0,y1,y2 = np.log(fftData[which-1:which+2:])
            x1 = (y2 - y0) * .5 / (2 * y1 - y2 - y0)
            thefreq = (which+x1)*RATE/chunk
            if thefreq < 1400 and thefreq > 1000:
                readData.append(1)
            if thefreq > 2000:
                readData.append(0)
        else:
            thefreq = which*RATE/chunk
        data = wf.readframes(chunk)
    if data:
        stream.write(data)
    stream.close()
    p.terminate()

    for i in range (0,int((len(readData)-1)/7)):
        letter = ""
        for binary in range(0,7):
            letter = letter + str(readData[0])
            readData.pop(0)
        print(letter)
        letter = int(letter,2)
        stringData.append(letter.to_bytes((letter.bit_length() + 7) // 8,
'big').decode())


if __name__ == '__main__':
    print("Start recording...")
    record_to_file('Ttu.wav')
    print("Recording done, start reading")
    openFile('Ttu.wav')
    print("Reading done")
```

[17] [18]

Lisa 3 – Lähtekood signaali graafiliseks kujutamiseks

```
import matplotlib.pyplot as plot
import numpy
import wave,sys

file = input("Insert filename: ")
openFile = wave.open(file,'r')

#Extract Raw Audio from Wav File
signal = openFile.readframes(-1)
signal = numpy.fromstring(signal, 'Int16')


#If Stereo
if openFile.getnchannels() == 2:
    print ('Just mono files')
    plot.plot(signal)
    plot.show()
    sys.exit(0)

plot.figure(1)
plot.title('Signal Wave…')
plot.plot(signal)
plot.show()
```

[19]