TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Karl-Friedrich Lind 186105IACB

Projekt ADMS graafilise kasutajaliidese loomine

Bakalaureusetöö

Juhendaja: Peeter Ellervee
PhD

Tallinn 2021

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: Karl-Friedrich Lind

21.03.2021

Annotatsioon

Project ADMS graafilise kasutaja liidese loomine

Antud lõputöö eesmärgiks on luua graafiline kasutajaliides, mis abistab seadme nimega Projekt ADMS kasutajat. Tegemist on programmeerimiskeele Python 3-l põhineva heli esitamise seadmega, mille graafilise kasutajaliidese loomine lihtsustaks märgatavalt seadme kasutamist. Kõigepealt selgitan töös, mida seade teeb ja mida täpsemalt graafiline kasutajaliides kuvab. Seejärel esitan tarkvara analüüsi ja disaini tulemused. Analüüsi eesmärgiks on selgitada välja nõuded graafilisele kasutajaliidesele ning disaini eesmärgiks on koostada nõuete, võimaluste ja piirangutega arvestav tehniline kavand.

Töö lõplikuks tulemuseks on Python 3-s kirjutatud graafiline kasutajaliides, kuhu kuvatakse seadme kasutamise ajal vajalik informatsiooni. Kasutades Tkinter Python 3 raamistikku kuvatakse Raspberry Pi 4 küljes olevale ekraanile info seadme töö kohta. Sarnaseid seadmeid on varem loodud. Kuid ei leidnud ühtegi varem loodud seadet, mis on sarnase arhitektuuriga ehitatud.

Töö valideerimiseks kasutan seadet ja võrdlen, kas heli, mida seade väljastab, on korrelatsioonis sellega, mida ekraan kuvab. Töö viimases osas hindan graafilise kasutajaliidese täpsust ja kasutusmugavust.

Tarkvara on avatud lähtekoodiga. Tarkvara lähtekood on avaldatud aadressil: https://github.com/MASTERLLAMACHEESE/project_ADMS

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 36 leheküljel, 9 peatükki, 14 joonist, 2 tabelit.

Abstract

Graphical User Interface for Project ADMS

The aim of this thesis is to create a graphical user interface that assists the user of a device called Project ADMS. It is an audio playback device based on the programming language Python 3, the creation of which would significantly simplify the use of the device. First, I will explain in the work what the device does and what the graphical user interface displays in more detail. I then present the results of the software analysis and design. The aim of the analysis is to find out the requirements for the graphical user interface and the aim of the design is to prepare a technical design that takes into account the requirements, possibilities and limitations of the hardware.

The end result is a graphical user interface written in Python 3 that displays the information needed to use the device. Using the Tkinter Python 3 framework, information about the operation of the device is displayed on the screen attached to the Raspberry Pi 4. Similar devices have been created before. However, no previously created device with a similar architecture was found.

To validate the job, I use the device and compare whether the sound the device emits correlates with what the screen displays. In the last part of the thesis I evaluate the accuracy and ease of use of the graphical user interface.

The software is open source. The source code of the software is published at: https://github.com/MASTERLLAMACHEESE/project_ADMS .

The thesis is in Estonian and contains 36 pages of text, 9 chapters, 14 figures, 2 tables.

Lühendite ja mõistete sõnastik

IA	Arvutisüsteemide instituut
ADMS	Analoog Digitaalne Modulaarne Süntesaator
LED	*Light-emitting diode*, valgusdiood
Sine-wave	Y=sin x geomeetriline laine
CLI	*Command Line Interface*, Käsurida
GPIO	*General-Purpose input/output*, Mitme otstarbeline sisend/väljund
MVC	*Model-View-Controller*, Mudel vaade kontroller
CPU	*Central Processing Unit*, Kesktöötlusseade

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tänapäeva maailmas, kus tehnoloogia areneb niivõrd kiirelt, on loomelisus, eriti muusikaline loome, väga kõrgel kohal. Enda muusikaliseks väljendamiseks on mitmeid erinevaid võimalusi. Otsitakse pidevalt uusi võimalusi selleks, seal hulgas ka mina. Luuakse uusi seadmeid ja tarkvara, mis annaks uusi võimalusi, kiirendaks ja muudaks mugavamaks loome protsessi. Aine IAS1420 Arvutite ja süsteemide projekt [1] raames loodi grupitööna, koostöös kaastudengiga Kaupo Sinimaa, projekt ADMS (joonis 1), mille lõpptulemus oli lihtne digitaalne süntesaator. Süntesaator on muusikainstrument, mille eesmärk on luua elektroonilisest signaalist looduses eksisteeriva akustilise heli imitatsiooni. Selle töö põhineb helisünteesil. Süntesaatoreid saab jagada riistvaralisteks ja tarkvaralisteks, virtuaalseteks. Riistvaralised jagunevad omakorda analoog- ja digitaalsüntesaatoriteks. Projekt ADMS liigitub riistvaraliste digitaalsete süntesaatorite alla. Tavaliselt kasutakse süntesaatoritel klahvistikku, mis sarnaneb klassikalisele klaverile. Projekt ADMS on eriline nuppude poolest, mis on kasutusel klahvide asemel ning töötavad OFF-ON põhimõtet järgides. Süntesaatori töö põhineb helisünteesil, mis on elektroonilisest signaalist heli loomine. Samamoodi saab Projekt ADMS elektrilise signaali ning mängib seejärel vastava heli. Lisaks sellele on võimalik seadmes olevad helisid moduleerida ja esitada.

Lõputöö eesmärk on luua Project ADMS-ile graafiline kasutajaliides. Valmiva tarkvara puhul on tegemist lisafunktsionaalsusega, mis muudab Projekt ADMS kasutajakogemust mugavamaks.

Lõputöö on jagatud alljärgnevateks etappideks:

- Analüüsida, mida kuvada graafilises kasutajaliideses;
- Disainida vajaminevale funktsionaalsusele vastav graafiline kasutajaliides;
- Analüüsida, kui suur on ressursikasutus graafilisel kasutajaliidesel;
- Ühildada uus kood varasemalt kirjutatud koodi stiiliga;

- Valminud graafilise kasutajaliidese testimine.

Algselt pidin analüüsima, mis osa seadme tööst tuleks kajastada graafilises kasutajaliideses. Leidsin, et oluline on näidata kasutajale, kuidas tarkvara kasutab helisid ja kuidas need on seoses füüsilises kasutajaliideses olevate nuppudega. Kasutades olemasolevaid funktsioone, mis on heli mängimine ja heli taasesitamine, otsustasin, et nende kasutamise juurestuleb näidata, mis nupul mis helile vastab. Järgnevalt tuli mõelda, milline peab graafiline kasutajaliides välja nägema ning mis on parim viis kasutajale seadme tööd näidata. Disainimise ja loomise käigus tuli jälgida, et graafiline kasutajaliides ei oleks liiga kompleksne, ning et ressursikasutus oleks minimaalne. Seade peab täitma põhiülesannet, milleks on helide mängimine ja taasesitamine.

Töös kasutasin ainult Python 3 programmeerimiskeelt ja Tkinter raamistikku [2]. Tkinter raamistik olles väga laialdaselt kasutusel on võimalik leida juhendeid, kuidas luua graafilise elemente. Programmeerimiskeele Python 3 kasutamine oli eelistatud varasema kogemuse tõttu.

2 Projekt ADMS olemus

Projekt ADMS eesmärk on luua üks multifunktsionaalne seade, mis võimaldab kasutajal esitada helisid, neid omavahel siduda taasesitamiseks ja jooksvalt manipuleerida. Aine IAS1420 raames loodi helide taasesitamise, helikõrguste manipuleerimise ja dünaamilise tempo muutmise funktsionaalsus [3]. Füüsiline kasutajaliides koosneb 38 nupust, mis on 7x7 maatriksis. Maatriksit on kasutatud, et vähendada vajalikke ühenduste kogust. Kasutusel olev riistavara Raspberry Pi on piiratud GPIO ühendustega, mille kaudu on ühendatud kõik füüsilise liidese nupud.

Kuusteistkümnele nupule on eelnevalt peale laetud helid. Need helid on valitud nii, et kasutajal oleks võimalik seadmega simuleerida terviklikku trummikomplekti. Neid on näha joonisel 1 punase värviga ümbritsetud alal.

Taasesitaja funktsionaalsusele on mõeldud teine komplekt nuppe ning neid on samuti kuusteist, joonisel 1 on nupud sinise värviga ümbritsetud alal. Need on jagatud neljaks erinevaks taktiks, kus iga nupp tähistab ühte lööki. Musta värviga nupud on paarituarvuga löögid ja punased on paarisarvuga löögid Iga lööki tähistava nupu peale on võimalik salvestada kuni neli erinevat heli. Taasesitamise jaoks on määratud eraldi nupp, mis käivitab või lõpetab esituse. Joonisel 1 on näha kahte sinist värvi nuppu, ühel neist on roheline tähistus ümber, teisel tähistus puudub. Puuduva tähistusega sinine nupp on mõeldud taasesitajat käivitama ja peatama.

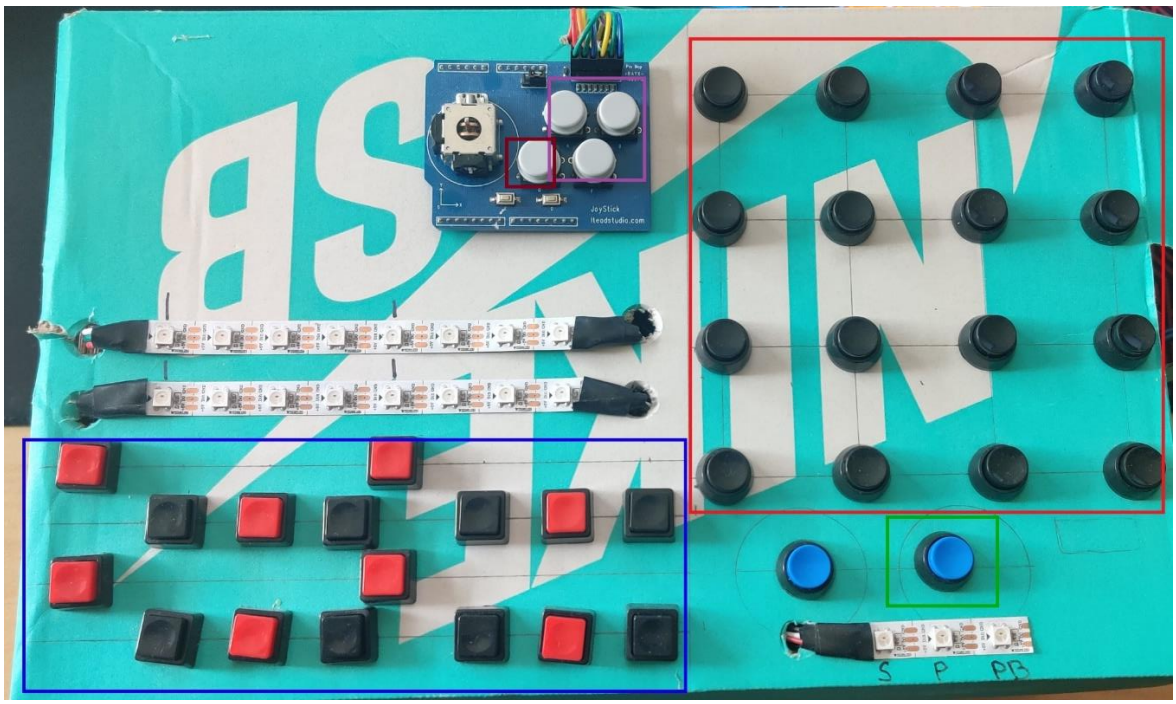
Joonisel 1 tähistatud rohelise kastiga nupp on määratud helikõrguse manipuleerimiseks. Sobiva heli valimisel saab seadme panna vastavasse helimanipuleerimise olekusse. Sellesse olekusse võetakse viimasena valitud heli ja muudetakse selle heli kõrgust kuueteistkümnel erineval viisil pool tooni alusel. Uued saadud helid paigutatakse kuueteistkümnele nupule, joonisel 1 punane ala, kus algselt asusid eellaetud helid. Olekust väljudes kustutakse tekitatud helid mälust ja taastatakse algolek, enne heli manipuleerimis olekut olevad helid on mängitavad..

Joonisel pruuniga tähistatud nupu eesmärk on vaigistada eelsalvestatud helisid olukorraks, kus taasesitaja töötab ja sealt helisid mängitakse, kui soovitakse helisid taasesitajasse lisada. Helides valimise käigus mängitakse see heli, kuid olles vaigistatud olekus siis nupu vajutsel ei väljastata heli kui saab teha aktiivseks, et lisada taasesitajasse.

Dünaamilise tempo muutmiseks on kolm lilla värviga ümbritsetud nuppu. Neist üks muudab tempot kiiremaks, üks vähendab ja üks taastab tempo eelnevalt ettemääratud tempoks, mis on 124 lööki minutis. Võimalik vahemik tempo muutmiseks on 70 kuni 200 lööki minutis.

Lisaks on olemas LED indikaator tulukesed, mis näitavad kasutajale, millises olekus taasesitaja on, kas on aktiivne ja kus kohas taasesituse hetkel valitud heli asub, kui üldse asub. Samut saab LED indikaatoritega näha, kas seade on vaigistamis oleksus, manipuleerimis olekus või millises.

Füüsiline liides võimaldab kasutajal seadet juhtida vastavalt enda soovidele ja esitada helisid täpselt sellisel viisil nagu soovitakse. LED indikaator tulukesed annavad tagasisidet nuppu vajutuste.



Joonis 1 Project ADMS prototüüp ilma ekraanita, koos värviliste märgistustega

2.1 Graafilise kasutajaliidese töö

Graafilise kasutajaliidese eesmärk on lihtsustada kasutajakogemust, näidates ekraani peal infot seadme kohta. Käesoleva töö raames graafiline liides näitab helide paiknemist vastavalt nuppudele.

Projekt ADMS töötab Raspberry Pi 4 platvormi peal ja sel põhjusel võtsin kasutusele Raspberry enda toodetud ekraani, *Official Raspberry Pi 7" Touchscreen*. Tegemist on Raspberry poolt toodetud ametlikult toetatud ekraaniga, mille suurus on seitse tolli ja millel on puutetundlik omadus. See ekraan sobis ka suurepäraselt, sest graafilises kasutajaliideses navigeerimiseks on kasutusel just ekraani puutetundlikkus.

Graafilisel liidesel on kuvatud *pad*, joonisel 1 punasega tähistatud ala, millised helid kus paiknevad, mis on nende nimetus ning kus asuvad eellaetud helid. Samuti on reaalajas kuvatud, taasesitaja vaade, joonisel 1 sinisega tähistatud ala, samuti näitamaks, mis helid, mis nupuga kokku lähevad. See eemaldab kasutajal vajaduse helisid ükshaaval läbi proovida ja pidevalt jälgida LED-indikaator tulesid, mis näitavad helide asukohta. Samuti eemaldab vajaduse seade kõik nupud pähe õppida ja terve kasutamis aja liigtused meelde jätta. Parim lahendus selleks on kasutada kahte erinevat vaadet, kus ühes on kuvatud kuueteistkümnel nupul, *pad*, olevad helid ja teises taasesituses, *sequencer*, olevad helid.

2.2 Võimalikud muudatused

Võimalikud muudatused oleks graafilise kasutajaliidese uuendamine ja väljanägemuse muutmine. Põhiline oleks seejuures värvide lisamine. Kuueteistkümnel nupul, pad, olevatel helidel on kõigil oma värv, mida saaks kasutada ka taasesitaja vaates, et kasutaja saaks värvide kaudu helisid ja nuppe seostada. Samuti oleks muudatuste seas kasutajaliidese atraktiivsuse tõstmine. Seda oleks võimalik teha, tuues juurde erinevaid värve ja muutes lihtsad tabelid ruumilisemaks ja luues erinevaid kujundeid.

3 Sarnased seadmed

Inimesed on sarnaseid seadmeid teinud omaalgatuslikult, kuid on ka olemas firmad, mis müüvad süntesaatoreid Raspberry platvormi peal. Tuntuim neist on Zynthian [4] (joonis 2). Zynthian on loonud seadme, mille tarkvara on vabavaraliselt kättesaadav ning igaüks saab sarnase seadme ka ise luua. Zynthian valmistab enda klientidele kvaliteetse ja nõutele vastava süntesaatori, kuid nende toode on rohkem suunatud olemasolevate instrumentide emuleerimiseks ja mängitava heli manipuleerimiseks. Heli manipuleerimise eesmärgil on Raspberry Pi peale vabavaras olemas mitmeid seadmeid, mis on saanud inspiratsiooni suur firmade toodetest. Samuti on loodud mitmeid seadeid, mis on mõeldud Sine-wave heli esitama ja seda moonutama. Vabavaras on leitav ka noLoop seade [5] (joonis 3), millel on lisaks olemas juhend, kuidas seadet kokku ja tööle panna. Seade on mõeldud Sine-wave moonutama, et tekitada uusi helisid, sarnaselt traditsioonilisele süntesaatorile. Teine võimalus on peale laadida helid ning seejärel moonutada ja taasesitada sellest tulnud uusi helisid.

Joonis 2 Zynthian toote pilt [6]

Joonis 3 noLoop vabavara kasutades loodud seade [7]

3.1 Inspiratsiooniallikad

Projekt ADMS on saanud inspiratsiooni ka suurte firmade töödest. Mitte küll ühest konkreetsest aga mitmest, mille olemust on omavahel kombineeritud. Nimelt ADMS on oma olemuselt sarnane *drumpad* [8] ja *sequencer* [9] seadmetega, võttes mõlemast osakese ja kasutades neid koos. ADMS küll ei ole trummipatjadega, vaid nuppudega, mis eemaldab temast rütmilist dünaamikat, kuid siiski saab sarnaseid jooni tõmmata seadmega nagu näiteks Roland SPD-SX [10]. Ühtlasi on mitmeid sarnaseid tooteid taasesitajale. Olemasolevad variandid on põhiliselt mõeldud stuudiokasutuseks, ent leidub ka esinemiseks mõeldud mudeleid. ADMS algse visioonina peaks mõlemas keskkonnas otstarbekas olema, kuid rohkem siiski suunatud esinemiseks. Oluline osa elektroonilise muusikaseadme juures on tema otstarve. Seadmed, mis on mõeldud muusika produtseerimiseks, ja seadmed, mis on etteasteteks, erinevad üksteisest mitme elemendi poolest. Esinemiseks mõeldud seadmed on tihti lihtsamad ja väiksema funktsionaalsusega, et tagada järjepidev ja kindel töö. Samuti on ka Projekt ADMS küllaltki lihtsa funktsionaalsusega, et tagada prototüüpseadme töökindlus.

4 Nõuded ja eesmärgid

Antud peatükis tuuakse välja loodava tarkvara graafilise kasutajaliidese eesmärgid ja mitte funktsionaalsed nõuded.

4.1 Graafilise liidese eesmärgid

- Näidata visuaalselt, kus eellaetud helid paiknevad kuuteistkümne nupu suhtes.

  Selle eesmärk on eemaldada kasutajal vajadus katsetada nuppe, et leida soovitav heli. Tänu visuaalsele kujutamisele saab kasutaja kohe õiget heli mängida.

- Näidata visuaalselt helide paiknemist taasesitaja vaates.

  See funktsionaalsus lihtsustab ülevaadet helidest taasesitajas. Koostöös LED-indikaatoritega saab kasutaja selge ülevaate, kus paiknevad helid seoses löökidega taasesitajas.

- Uuendada graafilist kasutajaliidest reaalaajas.

  Graafilist kasutajaliides uuendub reaalajas ning iga uue heli lisamisega on seda kohe võimalik näha. See välistab käsitsi graafilist liidese uuendamise vajaduse.

Nende kolme punkti täitmine on oluline, et luua kasutajasõbralik graafiline kasutajaliides ja parandada kasutaja kasutamiskogemust.

4.2 Mittefunktsionaalsed nõuded

- Lihtsasti arusaadav graafiline liides.

  Visuaalne näide sellest on peatükis 7 ja joonisel 10. Põhifookus on seejuures eelkõige funktsionaalsusel. Sellest tulenevalt on graafiline kasutajaliides üsna lihtne ning vähem on keskendutud välimuse atraktiivsusele. Peamine eesmärk on kasutajale anda selget infot.

- Kirjutada PEP 8 [11] stiili järgi kood.

Eesmärk on lihtsustada tuleviku arendust, kui keegi teine soovib luua lisafunktsionaalsust või koodi muuta. Järgitakse laialdaselt kasutusel olevat koodi stiili standardit.

5 Disain

Disainimisel ning programmeerimisel arvestasin põhiliselt puhta koodi [12] põhimõtetega. Raamistike valikul oli oluline, et arendamisel kasutatav tarkvara oleks vaba tarkvara ja ajakohane.

5.1 Kasutatava vahendid ja valiku põhjendused

Esimene samm õige tarkvara leidmiseks oli välja uurida, milline on kõige õigem Python raamistik graafilise kasutajaliidese loomiseks antud tarkvara juures. Selleks kasutasin Google otsingumootorit ning kasutasin märksõnu: *Python GUI creation, GUI with Python*.

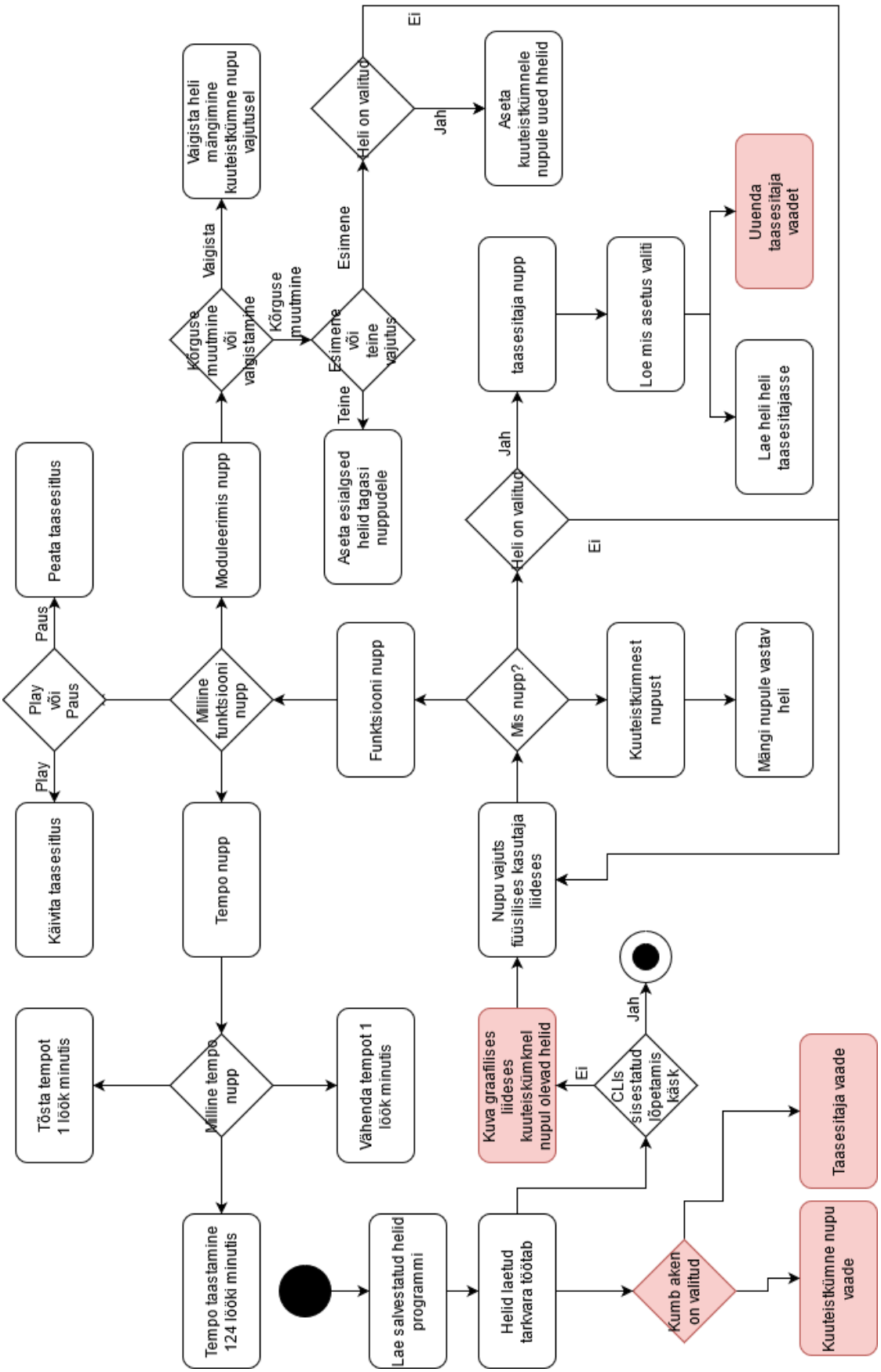
Läbi selle otsingu leidsin mitmeid erinevaid raamistikke. Laialdasemalt levinute seas oli Tkinter [2], PyQt [13], Kivy [14]. Kõik kolm on väga head raamistikud, milles luua graafilist kasutajaliidest.

Otsustasin kasutusele võtta Tkinter raamistiku. Eelis teiste ees on see, et Tkinter on osa python.org süsteemist ja on vaba tarkvara. PyQt on professionaalse taseme tööriist, mille eest tuleks maksta, kui realiseeritav rakendus ei ole vabavara. Hetkel hoidusin selle kasutamisest, sest lõplikud tulevikuplaanid ADMSile ei ole veel selged. Kivy ei ole ühegi organisatsiooni oma, vaid on erinevate inimeste poolt hallatud kogukonnaprojekt. See osutus negatiivse aspektina, sest ei leidnud enda jaoks piisavalt selgelt sõnastatud dokumentatsiooni. Tkinteri suur eelis oli ka selgesti arusaadav ja lihtsasti kättesaadav dokumentatsioon [2], kus on selgitatud, mida erinevad funktsioonid raamistiku sees teevad.

5.2 Tehniline arhitektuur

Realiseerisin programmile graafilise kasutajaliidese, mis käivitub üheaegselt programmiga., avades kasutaja ekraanil akna, kuhu kuvatakse vastav informatsioon. Järgnevas plokkskeemis (joonis 4) on näha, kuidas tarkvara töötab. Punasega on märgistatud tarkvarast see osa, mis on antud töö puhul tehtud. Näidates ära kuidas graafiline kasutajaliides on seotud ülejäänud programmi töösse. Peale tarkvara käivitamist ja helide laadimist, käib tarkvaras pidev kontroll, mis nupuvajutust tehakse

füüsilisel kasutajaliidesel ja milline aken on valitud graafilises kasutajaliideses. Vastavalt sisendile jätkatakse tööga nii nagu on plokkskeemil kuvatud.



Joonis 4 Tarkvara plokkskeem

5.3 Tarkvara disain

Tarkvara disainimisel oli oluline lähtuda MVC mudelist, sest varem loodud kood oli kirjutatud vastavalt sellele. MVC [15] on rakenduse disainimuster, kus komponentidel on oma eesmärk ja ühenduvad omavahel vastavate eesmärkide täitmiseks. MVC eesmärk on jagada programm kolmeks osaks, et eemaldada tarkvarasisese info ristumine. Projektis ADMS väljendub MVC selles, et kasutaja manipuleerib seadet füüsilise kasutajaliidesega, mida jälgib alamprogramm tarkvarast. Nupuvajutus edastab info põhiprogrammi, mis omakorda saadab vastava signaali alamprogrammi. See võimaldab esitada heli, muuta väärtust taasesitajas ja vastavalt nupuvajutusele saata graafilise kasutajaliidese alamprogrammi, et kuvada muudatused ka ekraanil. Põhiprogramm ei tee ühtegi vajalikku arvutust või mäluruumi muutust, vaid edastab alamprogrammidesse signaali, mis tuli kasutaja poolt antud sisendist.

6 Tarkvara

Seadme kasutamiseks on kasutajal vaja Raspberry Pi, soovitatavalt neljanda generatsiooni oma või mõnda uuemat ja võimalikult kiire protsessorikiirusega varianti. Lisaks tuleb sellele installeerida Headless Rasbian, mis on Raspberry enda hallatav Linux ning on spetsiaalselt mõeldud ARM protsessorile ja sellele seadmele. Operatsioonisüsteemi tuleb kergelt modifitseerida, lisades sellele ilma lisadeta töölaua keskkonna ehk ainult süsteemi navigeerija ja terminali. Selle eesmärk on hoida süsteemi ressursikasutust võimalikult madalal. Samuti tuleb eelnevalt installeerida vajalikud raamistikud ja teegid Python3 vajaliku programmi jooksutamiseks. Python3-pip on vajalik erinevate raamistike installeerimiseks. FFMPEG on vajalik helifailide tüüpide muutmiseks esitamisel, teha helifailist, helifaili objekt. Pynput on vajalik ainult siis, kui on soov arendada, et programmi arvutis jooksutada. Pynput on kasutusel Projekt ADMS füüsilise kasutajaliidese simuleerimiseks arvutis. Pydub on vajalik helide esitamiseks. Pygame on vajalik helide moduleerimiseks. Kõikide vajalike raamistike installeerimiseks on olemas fail, mille jooksutamisel otsitakse kõikide vajalike raamistike andmebaasid ja installeeritakse seadmesse õigel kujul, õigete jooksmis õigustega.

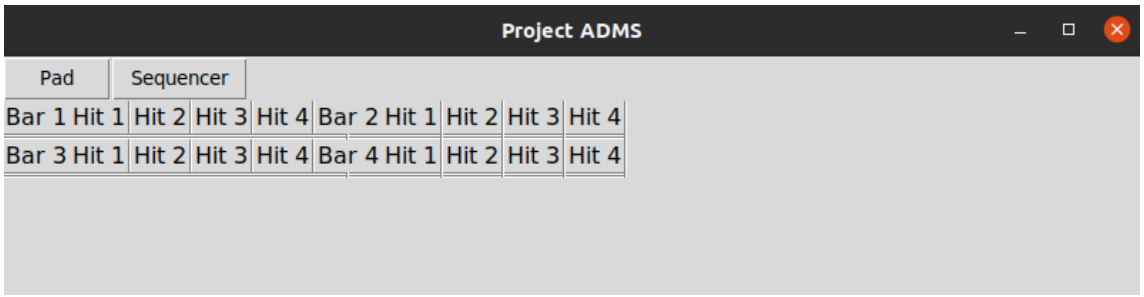
Tarkvara on kättesaadaval GitHub'is:

https://github.com/MASTERLLAMACHEESE/project_ADMS

6.1 Tarkvara kasutamine

Tarkvara kasutamine nõuab terminali ehk teisisõnu CLI kasutamist. CLI-d kasutades tuleb liikuda projet_adms kausta ja seal see käivitada. Kasutades *sudo python3 adms.py rasp või pc*. Sudo õigusi on tarvis kasutada, kuna vastasel juhul ei suuda programm näha heli väljundit ja heli ei kostu. Python3 on kasutusel, sest koodi ülesehitus kasutab just selle omapärasid ja meetodeid. Rasp või pc lisa näitab, millisel platvormil seadet kasutatakse. Rasp annab teada, et programm pannakse Raspberry Pi peal tööle ning programm teab edasi kuulata GPIO-d. Pc lühend ütleb, et programm aktiveeritakse arvutis, misjärel tuleb GPIOd ignoreerida ja kuulata hoopis klaviatuurilt sisestusi, mis simuleerib nuppe seadme peal. Nii pea kui programm käivitada, avaneb graafiline kasutajaliides ja aktiveerub indikaator LED seadme peal. Esimesena on graafilises liideses kohe näha helide paiknevust kuueteist nupu peal (joonis 7). Samuti on üleval

vasakul näha menüüd, kus on kaks valikut: helide asukoht, pad, ja taasesitaja vaade, sequencer (joonis 5). Taasesitaja vaade täitub jooksvalt helidega, mis pannakse vastavate löökide peale.



Joonis 5 Taasesitaja vaade

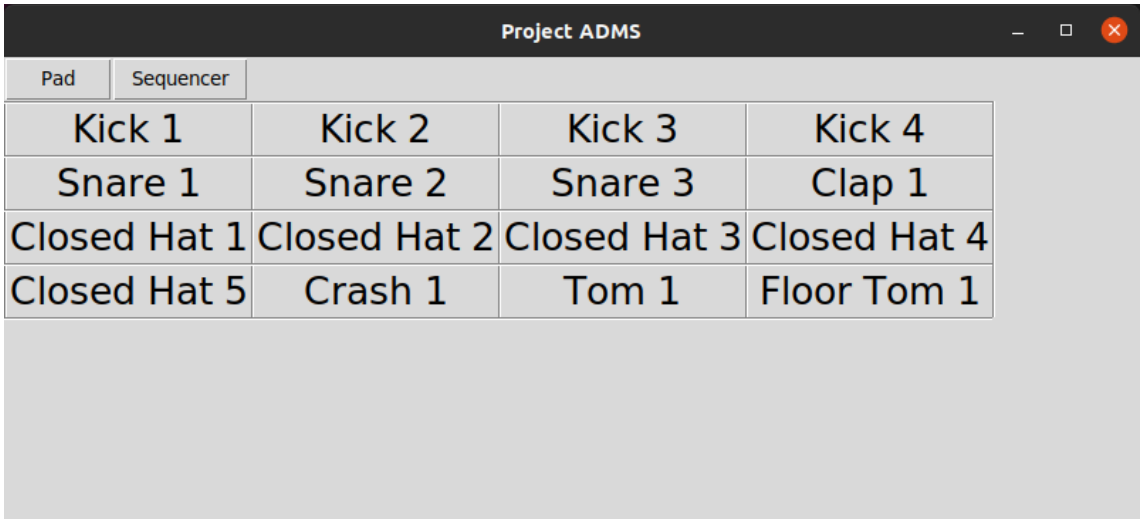
Graafiline kasutajaliides on loodud üsna lihtsakoelist välimust silmas pidades. Peamine fookus oli suunatud just funktsionaalsusele, jättes atraktiivsuse teisejärguliseks. Sellest olenemata on selgelt arusaadav, mida üks või teine nupp tähistab ja kus heli paikneb nuppude suhtes.

6.2 Tarkvara kood

Esimene samm graafilise kasutajaliidese loomise juures on vajaliku vaateakna loomine. Ühtlasi on olulisel kohal navigeerimine erinevate vaadete vahel. Joonis 6 esitatud koodilõigus on näha, kuidas on loodud nupud ja neile vastav tähistus vaadete vahetamiseks. Pad, Page1 vaadet on näha joonisel 5, mis näitab nuppudel asuvaid helisid. Sequencer, Page2 on vaade, mis näitab helide asetust taasesitaja vaates (joonis 10). Nupuvajutusega kutsutakse välja alamfunktsioon vastava vaate kuvamiseks.

```
def __init__(self, *args, **kwargs):
    tk.Frame.__init__(self, *args, **kwargs) #create gui frame
    p1 = Page1(self) #set pad view function callout
    p2 = Page2(self) #set seq view function callout
    buttonframe = tk.Frame(self) #create button frame
    container = tk.Frame(self) #create container for buttons
    #create button frame
    buttonframe.pack(side="top", fill="x", expand=False)
    #make it possible for buttons to be placed in container
    container.pack(side="top", fill="both", expand=True)
    #place button, pad view
    p1.place(in_=container, x=0, y=0, relwidth=1, relheight=1)
    #place button, seq view
    p2.place(in_=container, x=0, y=0, relwidth=1, relheight=1)
    #name button, make text appear in gui
    b1 = tk.Button(buttonframe, text="   Pad   ", command=p1.lift)
    b2 = tk.Button(buttonframe, text="Sequencer", command=p2.lift)
    #push buttons to left side of gui
    b1.pack(side="left")
    b2.pack(side="left")
    #show pad view on startup
    p1.show()
```

Joonis 6 Akende vahetamise nuppude kood



Joonis 7 Pad vaade

Joonis 8 on programmikood, mis loob taasesitaja vaate kujunduse. Joonisel näha olev raamistik on loodud vertikaalsetest ja horisontaalsetest joontest, kus iga takt on tähistatud sõnaga *Bar* ja numbriga on näidatud, mitmes takt järjekorras on. Sama lahendust on kasutatud taktides olevate löökidega, mis on tähistatud kui *hit* ja seejärel vastava numbriga. Kasutatud on ruudustiku süsteemi, ridu ja tulpasid. Igal joonel on oma rida või tulp olenevalt sellest, kas joon on horisontaalne või vertikaalne. Hetkel on kokku 21 rida ja 18 tulpa. Ridade arv on valitud selliselt, et iga löögi alla mahuks neli rida teksti – igal

löögil saab olla kuni neli heli. Kui jooned on pandud ridadele üks ja kuus, siis read kaks kuni viis on mõeldud tekstile, kus igal real on heli tähistav tekst. Sarnane olukord on vertikaalsete joontega. Igale joonele on antud oma tulp, et saaks eraldada tekste. Näidet sellest, milline on tavakasutus olukorras taasesitaja vaates, näeb peatükis 7 joonisel 14.

Lisaks kujundusele on oluline saada heliga seonduv tekst õigesse kohta graafilises kasutajaliideses ja luua võimalus sama tekst eemalda, kui kasutaja seda soovib. Selleks on kasutatud loendeid. Loend *module.gui_list* uuendatakse iga nupu vajutusega füüsilises kasutajaliideses. Loend *module.gui_list* sisu võrreldakse loendiga *self.active_list*. Kontrollitakse, kas löögi koha peal on vähem kui viis heli. Kui need kontrollid on edukalt läbitud, siis võrreldakse eeldefineeritud loend *self.position*, loendi sisu on tähistused vastava nupu vajutuse ja heli nimetusega. Järgnevalt lisatakse tekst loendist *module.gui_list* määratud kohale graafilises kasutajaliideses. Sarnane lahendus on kasutusel teksti eemaldamisel. Kuid sel puhul vaadeldakse, kas *self.active_list* loendi sisu vastab *module.gui_list* loendile. Kui ei, siis eemaldatakse raamistikus tekst. Kirjeldatud funktsioonide koodi saab näha joonis 9.

```
#set name for bar 1 and its hits
tk.Label(self, text="Bar 1", font=self.fontstyle).grid(row=0, column=0)
tk.Label(self, text="Hit 1", font=self.fontstyle).grid(row=0, column=1)
tk.Label(self, text="Hit 2", font=self.fontstyle).grid(row=0, column=3)
tk.Label(self, text="Hit 3", font=self.fontstyle).grid(row=0, column=5)
tk.Label(self, text="Hit 4", font=self.fontstyle).grid(row=0, column=7)

#draw vertical lines for gui
tkinter.ttk.Separator(self, orient=VERTICAL).grid(column=0, row=1, rowspan=21, sticky='ns')
tkinter.ttk.Separator(self, orient=VERTICAL).grid(column=2, row=0, rowspan=21, sticky='ns')
tkinter.ttk.Separator(self, orient=VERTICAL).grid(column=4, row=0, rowspan=21, sticky='ns')
tkinter.ttk.Separator(self, orient=VERTICAL).grid(column=6, row=0, rowspan=21, sticky='ns')
tkinter.ttk.Separator(self, orient=VERTICAL).grid(column=8, row=0, rowspan=21, sticky='ns')

#draw horizontal lines for gui
tkinter.ttk.Separator(self, orient=HORIZONTAL).grid(column=0, row=1, columnspan=18, sticky='ew')
tkinter.ttk.Separator(self, orient=HORIZONTAL).grid(column=0, row=6, columnspan=18, sticky='ew')
tkinter.ttk.Separator(self, orient=HORIZONTAL).grid(column=0, row=11, columnspan=18, sticky='ew')
tkinter.ttk.Separator(self, orient=HORIZONTAL).grid(column=0, row=16, columnspan=18, sticky='ew')
tkinter.ttk.Separator(self, orient=HORIZONTAL).grid(column=0, row=21, columnspan=18, sticky='ew')
```

Joonis 8 Taasesitaja vaate kujunduse loomine

```
def gui_places(self):
#get text insert and set it to the correct text field in grid
  #check list created when buttons pressed
  for e in module.gui_list:
    #compare previously created with active gui
    if e not in self.active_list[module.beat_in_seq]:
      for x in range (16):
        if module.beat_in_seq == x:
          amount = len(self.active_list[module.beat_in_seq])
          #check if there are less than 5 sounds on beat
          if amount < 5:
            if amount >= 1 and (amount + 1) < 5:
              amount =+ 1
            #check with tuple to get positon of sound
            for key, value in self.position:
              if module.beat_in_seq in value:
                 column_space = key
              #add new sound to gui
            insert = module.gui_list[module.beat_in_seq]
            col = column= column_space
            row = =amount
            tk.Label(self,text=insert,font=self.fontstyle).grid(col, row)

#get text remove and remove it from the correct place in grid
  for y in self.active_list: # loop active gui list
    #check if it is in button press created list
    if y not in module.gui_list[module.beat_in_seq]:
      for x in range (16):
        if module.beat_in_seq == x:
           amount = len(self.active_list[module.beat_in_seq])
           #check if there are less than 5 sounds on beat
           if amount < 5:
             if amount >= 1 and (amount + 1) < 5:
               amount =+ 1
             #get position of sound to be removed
             for key, value in self.position:
               if module.beat_in_seq in value:
                  column_space = key
             #replace removed sound text with blank space
             col = column= column_space
             row = =amount
             tk.Label(self, text="", font=self.fontstyle).grid(col,row)
```

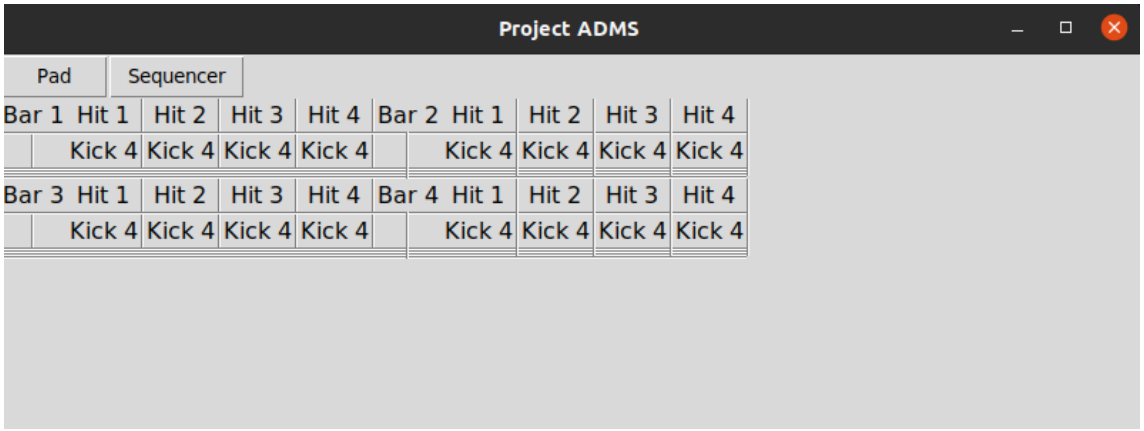
Joonis 9 Taasesitaja vaatesse teksti lisamine ja eemaldamine

7 Testimine

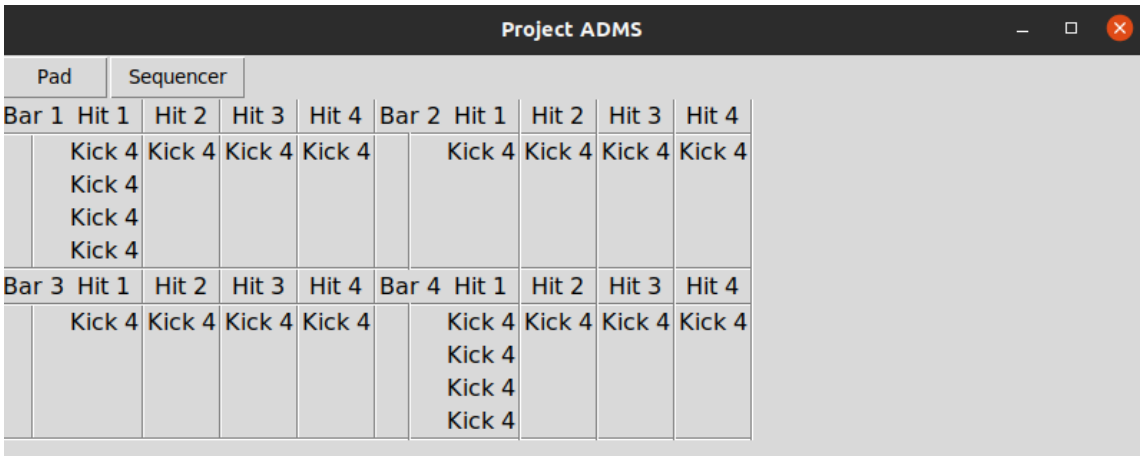
Kui graafiline kasutajaliides sai disainitud ja valmis programmeeritud tuli läbi viia selle testimine. Eelnevalt valminud füüsiline kasutajaliides aitas testimist läbi viia, graafilise kasutajaliidese tööd oli võimalik kontrollida füüsilises liideses indikaator LED näiduga. Esmalt sooviti vaadelda, kas kuueteistkümne nupu peal on kõik helid õigesti tähistatud, joonis 1 punasega tähistatud nupud. Katse seisnes selles, et vajutati nuppu ja kuulati, mis heli mängitakse. Seejärel võeti ette samanimeline helifail, mängiti seda väljaspool ADMS tarkvara ja vaadeldi, kas kõlab sama heli.

Taasesituse katsetamine sõltus rohkem tarkvarast. Graafilist liidest tuli pidevalt uuendada vastavalt sellele, millised helid millistele löökidele on asetatud taasesituse juures. Kõigepealt katsetasin, kas igale löögile ilmus suvaliselt valitud heli (joonis 10). Kui graafilises kasutajaliidesse suvaliselt valitud heli ei ilmu, siis teised funktsionaalsused ka ei tööta nii nagu peaks. Seda sai kontrollida kasutades füüsilisel kasutajaliidesel olevaid LED indikaatoreid. Suvaliselt valitud heli oli Kick 4, see ilmus igale löögile.

Järgmine katse seisnes selles, kas igale löögile on võimalik asetada maksimaalselt neli heli (joonis 11). Eesmärk on tagada, et graafilises kasutajaliideses ei ilmuks rohkem helisid kui neli, sest tarkvaras on piiratud nii, et saab lisada kuni neli heli igale löögile. Seda oli võimalik kontrollida kasutades LED indikaatoreid, kuigi olles kohati tülikas. Iga heli tuli eraldi kontrollida vastavalt LED indikaatoritele. Valides heli aktiveerida taasesitajas soovitud löök ja see järel kontrollida LED indikaator tuledelt, kas heli on õiges kohas ja kuulata heli väljundit, kas heli mängib õigel ajal.

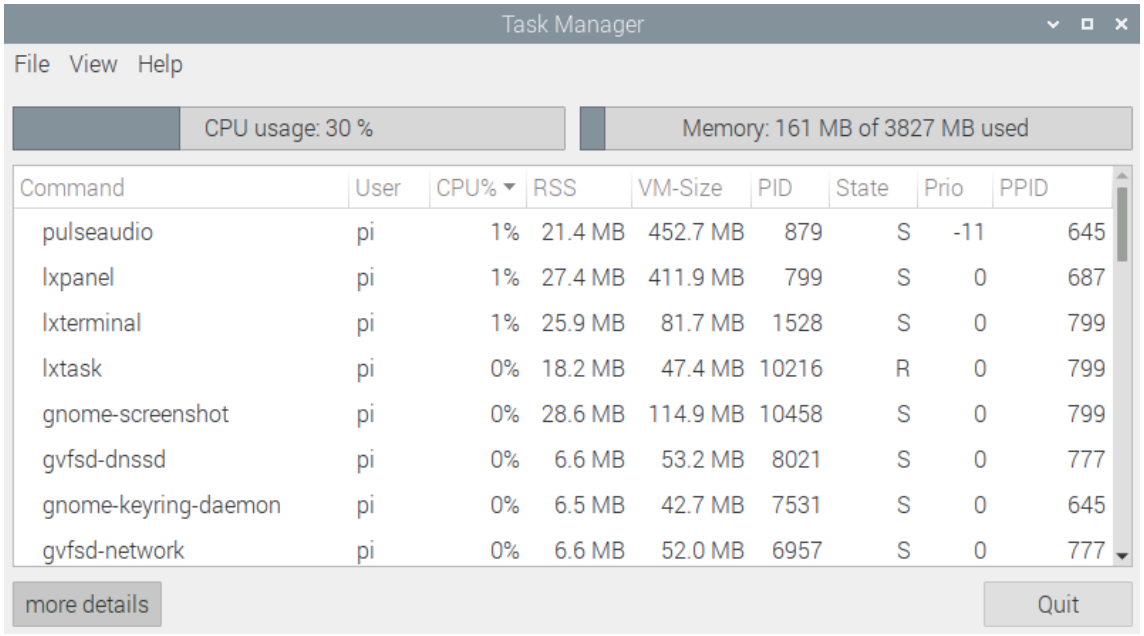


Joonis 10 Igale löögile on asetatud heli „Kick 4“

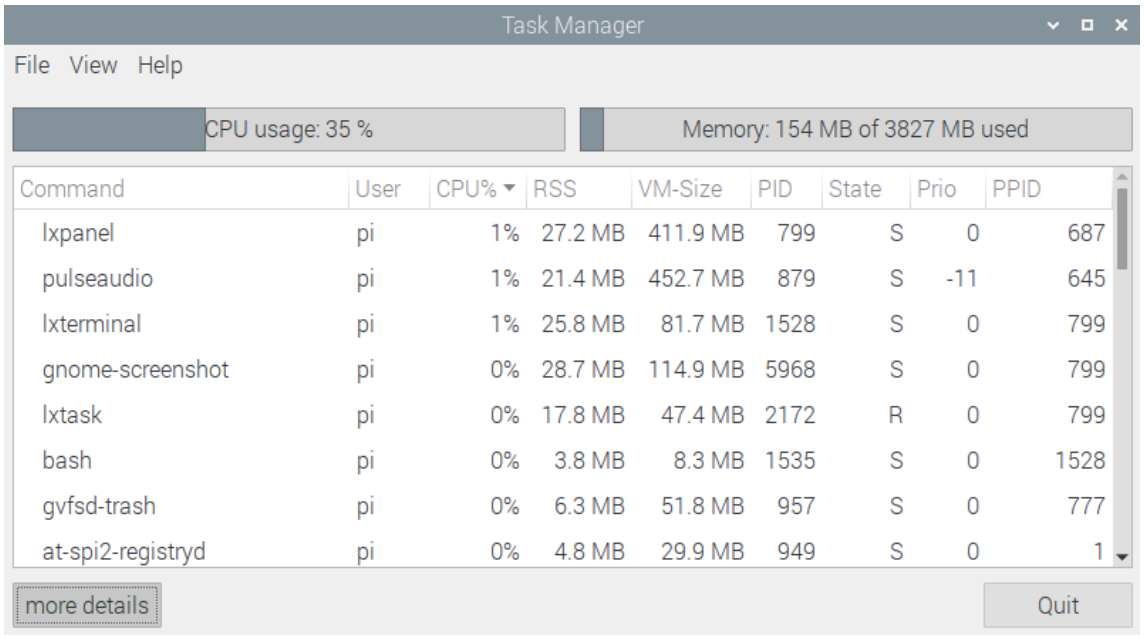


Joonis 11 Takt 1 ja Takt 4 esimesele löögile on pandud maksimaalne kogus helisid

Oluline oli katsetada, kuidas graafilise kasutaja liidese jooksutamine mõjutab Raspberry Pi võimekust enda ülesandeid täita. Selleks kasutasin Raspbian [16] sisse ehitatud *taskmanager* aplikatsiooni. Esimese sammuna oli tarvis uurida, kui koormatud süsteem on olukorras, kus graafiline kasutajaliides ei ole kasutuses (joonis 12). Seejärel aktiveeriti graafiline kasutajaliides ja võrreldi uuesti koormuseid (joonis 13). Tulemuste paremaks võrdlemiseks koostati tabel (tabel 1). Tabelis 1 olevate numbrite põhjal on näha, et süsteem on suurema koormuse all, kui graafiline kasutajaliides on aktiivne. Seda on näha CPU kasutusest, mis on loodud tarkvara puhul väga oluline näitaja. Tarkvara peab korraga täitma mitut ülesannet: heli tagasi mängima, jälgima taasesitajat, kontrollima nupuvajutusi ja graafilise liidese kasutamisel pilti uuendama.



Joonis 12 Tarkvara kasutamine ilma kasutajaliideseta

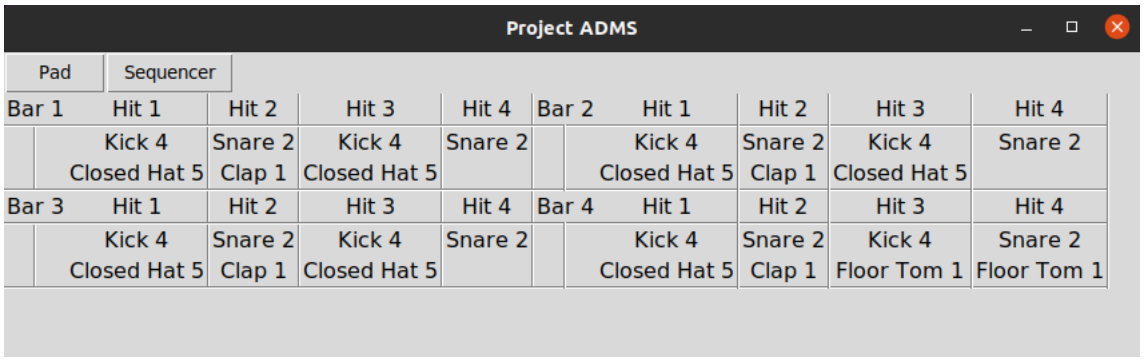


Joonis 13 Tarkvara kasutus koos graafilise kasutaja liidesega

Tabel 1 Tarkvara ressursi kasutus

	Graafilise liideseta	Graafilise liidesega
CPU	30%	35%
Memory	161 MB	154 MB

Edukalt sooritanud katsed, mille eesmärk oli kontrollida, kas graafiline kasutajaliides töötab eesmärgikohaselt, tuli soorida lisa katsetus. Testida olukorda, kus simuleeritakse tavakasutust. (joonis 14). Eesmärk on vaadelda, kas tavakasutuse ajal kuvatakse kõik vajalik ja kui selgesti see loetav on loodud graafilises kasutajaliideses.



Joonis 14 Ettenähtud tavakasutus olukord

Katsest saan järeldada, et kõik töötab nii nagu algselt oli plaanitud ja disainitud. Eesmärk oli luua graafiline kasutajaliides, mis hõlpsustaks kasutajakogemust. Eemaldades vajaduse kasutajal meelde jätta iga nupuvajutust ja liigutust, mis Projektiga ADMS on tehtud. Samas kogu info peab olema selgesti ja lihtsalt kätte saadavad graafiliselt kasutajaliideselt, eriti oluline on see taasesitaja kasutamise korral. Nähtav on, et soovitud kasutajakogemuse parandamine on saavutatud, lihtsustades protsessi, milles kontrollitakse, kus kohas asuvad taasesituses helid.

8 Tagasivaade tehtud tööle

Käesoleva peatüki eesmärk on hinnata tehtud tööd ja sellest tulenevalt sõnastada parendamisvõimalused projektile. Selleks, et tehtud tööd analüüsida, tuleb esmalt sõnastada, mis sooviti saavutada (Tabel 2) ning seejärel hinnata, kuidas tehtud töö seda väljendab.

Hindan tehtud tööd kolme palli süsteemis, kus 3 tähistab kõige paremat tulemust ja sooritus oli selline nagu algselt plaanitud. 2 tähistab ,et soovitud funktsionaalsus on saavutatud kuid saaks muuta paremaks. 1 on kõige halvem ja täiesti puudulik. Samuti on iga punkti kohta lisatud kommentaar, mis selgitab antud hinnangut.

Tabel 2 Hinnang funktsionaalsusele

Funktsioon	Hinnang	Kommentaar
Kuueteistkümne nupu kuvamine	3	Selgelt ja lihtsalt arusaadav, millise nupuga milline heli kokku läheb.
Taasesitaja kuvamine	3	Saab aru, millisele kasutaja valitud takti löögile, kasutaja poolt valitud heli on lisatud ja näeb reaalajas.
Funktsionaalsus	2	Kõik töötab vastavalt disainile. Lisada saaks veel selle, et igale helile on omistatud oma värv ja neid saab kasutada ka taasesituse kuvas, et mugavamalt jälgida helisid.
Atraktiivsus	1	Kuna põhiliselt fokuseerisin funktsionaalsuse loomisele, siis on graafiline kasutajaliides lihtsa välimusega. Edasiarenduse vaates on see hea aspekt, mida uurida.
Ressursikasutus	3	Peatükis 7 esitatud tabel 1 näitab selgelt, kui hästi on tarkvara optimeeritud.

Funktsionaalsused, mis pandi paika täitmiseks täideti. Kasutaja kasutamis kogemust parandati. Kuid selle käigu oli põhifookus funktsionaalsusel, loodud graafiline kasutaja liides, ei ole kuigi atraktiivse välimusega. See polnud ka ühtlasi eesmärk mida täitma pidi. Projekt ADMS on veel küllaltki algelises seisus ja arendus ruumi on veel palju. Kuid antud töö käigus loodud tarkvara on küllaltki suur saam terviklikkuse poole.

9 Kokkuvõte

Käesoleva töö eesmärgiks oli luua tarkvara, mis abistaks Projekt ADMS kasutajat. Selle saavutamiseks loodi graafiline kasutajaliides. Loodud tarkvara sisaldab graafilist kasutajaliidest, mis näitab kuueteistkümnele nupule eellaetud helide asukohta seoses nuppudega füüsilises kasutajaliideses, lisaks taasesitajas asuvate helide asukohta seoses taktis olevate löökidega ja nuppudega füüsilises kasutajaliideses. Tarkvara on kirjutatud kasutades Python 3-e.

Tarkvara on avatud lähtekoodiga ja avaldatud aadressil: https://github.com/MASTERLLAMACHEESE/project_ADMS.

Kirjeldasin töös Projekt ADMS olemust ja mis seade tegema peab. Samuti kirjeldasin graafilise kasutajaliidese eesmärki ja mida ta kuvab. Põhjendasin ka enda raamistiku valikut. Katsetasin loodud tarkvara ja analüüsisin loodud tarkvara ressursikasutust.

Tarkvaraks on rakendus, mis kuvab kasutajale Projekt ADMS kasutamiseks vajalikku infot. Graafiline kasutajaliides kasutab Python 3 Tkinter raamistikus loodud aknaid ja menüüd, mis kuvavad kasutajale soovitud infot. Peale Projekt ADMS tarkvara käivitamist avaneb graafilise kasutajaliidese aken, mille kasutamine on lihtne. Näha on vaid kahe valikuga menüü, mis muudab vaateid. Vaadelda saab korraga kas ainult kuueteistkümne nupu helisid või taasesitajat.

Kasutajaliides on hetkeseisuga suunatud funktsionaalsusele, Olgugi et liidese välimuse juures pole keskendutud tema välimusele, täidab ta edukalt oma primaarseid ülesandeid ja funktsionaalsusi. Samuti on kasutajaliides loodud inglise keeles eesmärgiga eemalda võimalikud keelepiirangud kasutamisel.

Töö käigus said edukalt realiseeritud kõik funktsionaalsused, välja arvatud osad, mille eesmärk oli kasutusmugavus. Kasutaja saab vaadata kergelt, millised helid asuvad kuueteistkümne nupu peal ja millised taasesitajas. Saab kindlalt väita, et soovitud tarkvara on realiseeritud edukalt.

10 Kasutatud kirjandus

[1] TalTech. [Võrgumaterjal]. Available: https://ois2.ttu.ee/uusois/aine/IAS1420.

[2] „tkinter,“ Python Software Foundation, [Võrgumaterjal]. Available: https://docs.python.org/3/library/tkinter.html.

[3] K. F. Lind & K. Sinimaa, *Projekt ADMS,* Tallinn: Taltech IAS1420, 2020.

[4] Zynthian, „Zynthian,“ Zynthian, [Võrgumaterjal]. Available: https://zynthian.org/.

[5] tom'sHARDWARE, „noLoop,“ [Võrgumaterjal]. Available: https://www.tomshardware.com/news/diy-synthesizer-raspberry-pi-project-granular.

[6] Zynthian. [Võrgumaterjal]. Available: https://shop.zynthian.org/web/image/11911.

[7] N. Fromm, „Tom's Hardware,“ [Võrgumaterjal]. Available: https://cdn.mos.cms.futurecdn.net/PjbqHqot9adv4mzmQi8GH-970-80.jpg.

[8] Oxford, „lexico,“ [Võrgumaterjal]. Available: https://www.lexico.com/definition/drum_pad.

[9] Oxford, „lexico,“ [Võrgumaterjal]. Available: https://www.lexico.com/definition/sequencer.

[10] Roland, „Roland.com,“ [Võrgumaterjal]. Available: https://www.roland.com/global/products/spd-sx/.

[11] Python.org, „Pytrhon.org PEP 8,“ Python, [Võrgumaterjal]. Available: https://www.python.org/dev/peps/pep-0008/.

[12] R. C. Martin, Clean Code: A Handbook of Agile Software Craftmanship, 2008: Prentice Hall.

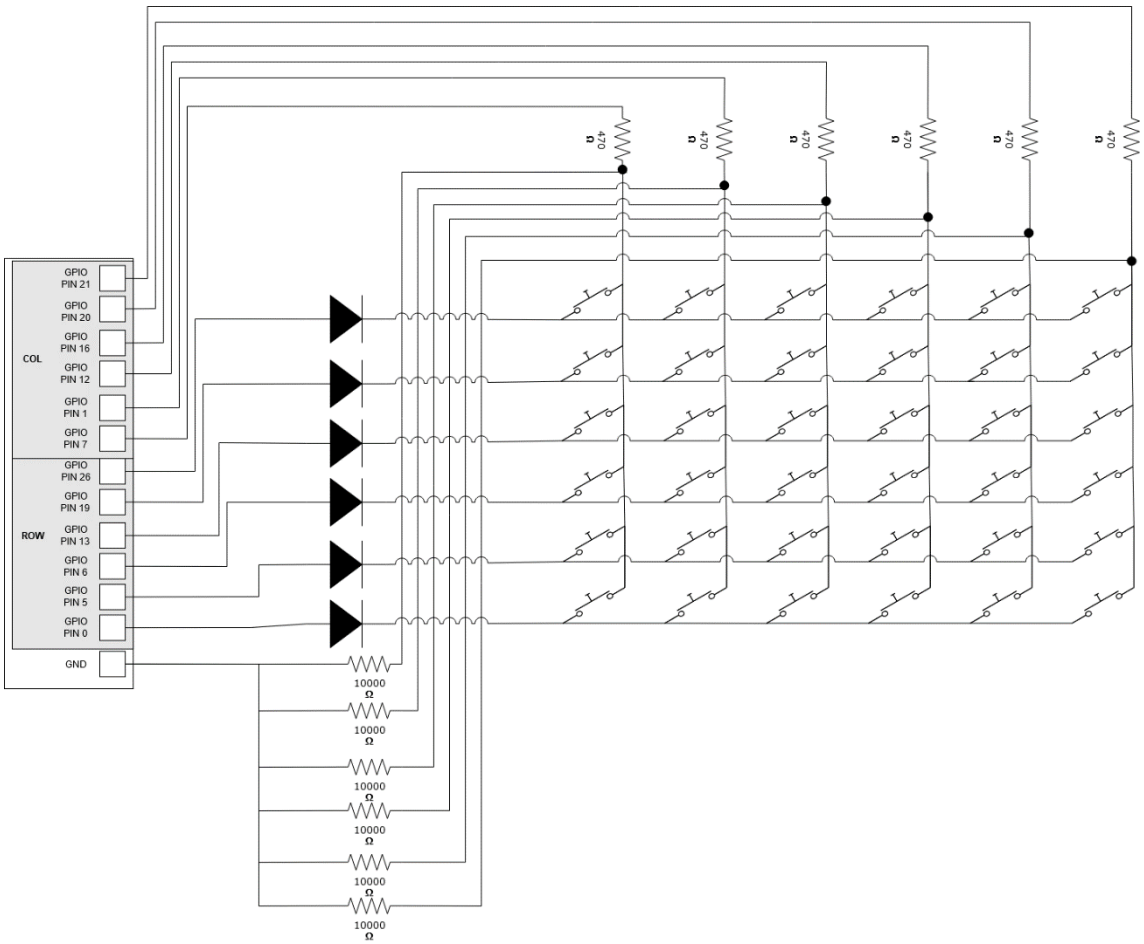
[13] Qt, „qt.io,“ The Qt Company, [Võrgumaterjal]. Available: https://www.qt.io/.

[14] Kivy, „kivy.org,“ [Võrgumaterjal]. Available: https://kivy.org/#home.

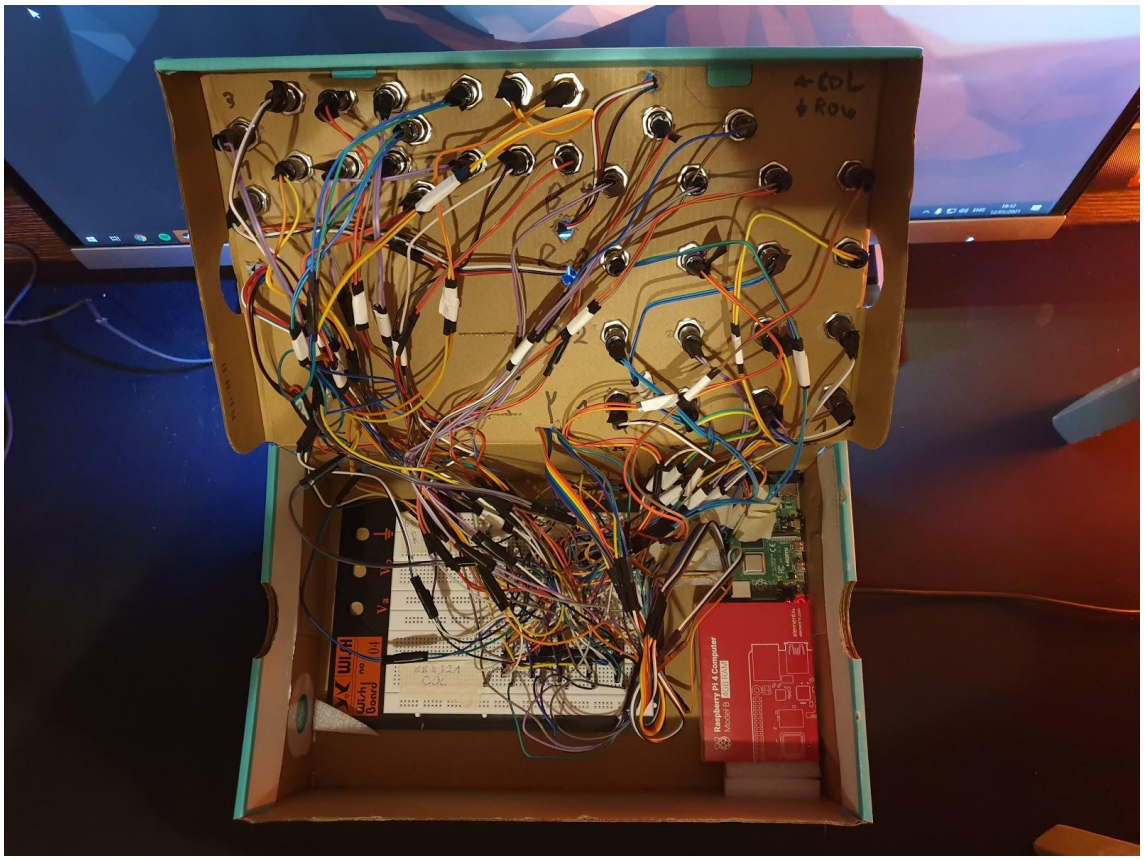
[15] T. R. j. J. O. Coplie, The DCI Architecture: A New Vision of Object-Oriented Programming, 2009.

[16] R. P. Foundation. [Võrgumaterjal]. Available: https://www.raspberrypi.org/software/operating-systems/.

Lisa 1 – Projekt ADMS nuppude ühenduse skeem



Lisa 2 – Projekt ADMS sisemus ilma ekraanita



Lisa 3 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Karl-Friedrich Lind

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Projekt ADMS graafilise kasutajaliidese loomine“, mille juhendaja on Peeter Ellervee
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

13.05.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.