TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Olev Abel

Audiovisuaalse mooduli programmeerimine “Minu loodusheli” Android rakenduse näitel

Bakalaureusetöö (9 EAP)

Juhendaja: Marko Peterson
Kaasjuhendaja: Neeme Kahusk

Tartu 2016

Audiovisuaalse mooduli programmeerimine “Minu loodusheli” Android rakenduse näitel

Lühikokkuvõte:

Käesoleva bakalaureusetöö raames valmis Android rakenduse “Minu loodusheli” näitel heli salvestamise ja visualiseerimise moodul. Mooduli arendamiseks kasutati väle tarkvaraarendus meetodit Kanban ning mitmeid arendusvahendeid. Mooduli peamisteks funktsionaalsusteks on helitugevuse testimise näidik, helitugevuse muutmine käsitsi ja automaatselt, heliformaadi valik ning salvestise helitugevuse visualiseerimine.

Võtmesõnad:

Audiovisuaalne moodul, Android, Kanban

CERCS: P175 – Informaatika, süsteemiteooria

Audiovisual module programming for “Minu loodusheli” Android application

Abstract:

As a result of this paper audio visual module for Android applications have been developed. The module consists of two different functional views: audio recording view and audio visualization view. Agile software development method called Kanban was used during the development of the module. Core functionalities of this module include audio volume meter, volume control by end user, audio format selection and recorded audio visualization.

Keywords:

Audiovisual module, Android, Kanban

CERCS: P175 – Informatics, systems theory

Sisukord

1. Sissejuhatus

Loodusvaatlused on saamas järjest populaarsemaks ning seda just tavainimeste seas. Loodusvaatluste tulemused on väga otseselt mõõdetavad andmete hulga näol. Kui seni on tehtud loodusvaatlusi välitöödel peamiselt kolme vahendiga: diktofon, fotokaamera ja vaatlusraport, siis nutiseadmete arenguga on nüüd võimalik kõik vajaminev ära teha ühe seadmega, tehes nii loodusvaatluse palju mugavamaks ning mobiilsemaks ja seeläbi ka lihtsaks, väljaspool eriala töötavatele inimestele.

Tehniline võimekus pakkuda kvaliteetset heli- ja pildimaterjali ning samuti salvestada kirjalike andmed on nutiseadmetel küll olemas, kuid mugavaks ja lihtsaks loodusvaatluseks on vaja ka tarkvara, mis kogu andmete voo ühtseks ning kergesti hallatavaks tervikuks seoks.

Tartu Ülikooli loodusmuuseum ja Eesti Loodusmuuseum on arendanud esimese versiooni omanäoliseimast mobiilirakendusest maailmas, mis võimaldab teha audiosalvestuse-põhiseid loodusvaatluseid, eelkõige loomariigi esindajate häälte ja helide püüdmiseks ning automaatseks andmebaasi salvestamiseks. Tarkvara loomise on enda kanda võtnud Tartu Ülikooli loodusmuuseumi elurikkuse informaatika ja digiarhiivide töörühm. Rakendus on ühendatud PlutoF infosüsteemiga. Rakenduse esmasest versioonist on puudu moodul, millega oleks võimalik suurem kontroll mobiilse seadme salvestusvahendite üle ning samuti visualiseerida salvestatud helifaili.

Käesoleva töö eesmärgiks on analüüsida erinevaid heli salvestamise võimalusi ning valida loodusvaatluste seisukohalt sobivaim. Samuti uurida, milline on optimaalne salvestusformaat mobiilsel seadmel heli salvestamiseks ning serverisse saatmiseks. Lisaks analüüsile on eesmärgiks programmeerida erinevaid lisavõimalusi pakkuv heli salvestamise moodul, järgimaks ülal mainitud rakenduse eripärasid. Lisaks on võimalik valmivat moodulit kasutada erinevate Android platvormi rakenduste juures. Veel kuulub eesmärkide juurde heli visualiseerimiseks mõeldud teegi ühildamine valmiva salvestusmooduliga, võimaldades salvestist visuaalselt analüüsida, tehes nii salvestise kasutajale mugavamini hallatavaks. Kogu moodul programmeeritakse loodusvaatluse vajadusi ja eripärasid silmas pidades. Koos mooduli programmeerimisega on eesmärgiks tutvustada väle tarkvara arendusmeetodit Kanban ning mooduli programmeerimisel lähtuda sellest arendusmeetodist.

2. Mikrofonid

Antud peatükis tuuakse välja mikrofonide liigitus. Samuti kirjeldatakse lühidalt mobiilsetes seadmetes kasutavate mikrofonide tööpõhimõtet ja tuuakse välja, mis on mobiilsetes seadmetes kasutatavate mikrofonide eelised ning miks kasutatakse just seda tüüpi mikrofone.

Erki Suurjaak [1] on oma referaadis (Abo 1997 [2] järgi) välja toonud mikrofoni lühitutvustuse, mis kirjeldab käesolevas töös käsi mobiilsetes seadmetes olevaid mikrofone. Mikrofon on oma olemuselt elektroakustiline seade, mille ülesandeks on muuta helisignaal samaväärseks elektrisignaaliks.

Kuna antud töö keskendub peamiselt mobiilsetes seadmetes kasutatavatele mikrofonidele, siis räägitakse pikemalt vaid elektrostaatilistest mikrofonidest. Elektrostaatilistel mikrofonidel, mis on kasutusel mobiiltelefonides, on membraan avatud helilainele vaid ühest küljest. Seega moodustab membraan suletud anuma ning sellised mikrofonid on õhurõhule tundlikud. Elektrostaatilistel ehk kondensaatormikrofonidel moodustab membraan ühe kondensaatori plaadi [3]. Helilained põhjustavad membraani vibratsiooni ning seetõttu muutub kondensaatori plaatide vaheline kaugus, mis tingib kondensaatori mahtuvuse muutuse [3]. Kondensaatori plaatidele antakse läbi takisti pinge, umbes 60 V. Kondensaatori mahtuvus arvutatakse valemiga $C = Q/U$, kus Q on elektrilaeng kulonites, U juhi potentsiaal ehk pinge voltides ja C kondensaatori mahtuvus faradites [4]. Sellest valemist saab leida pinge, mis on arvutatav valemiga $U = Q/C$ [4]. Kuna laeng jääb konstantseks, põhjustavad pinge muutusi kondensaatori mahutavuse muutused. Need pinge muutused salvestatakse ja need ongi helisignaali väljendus elektrisignaali näol.

2.1 Mobiilseadmetes kasutatavad mikrofonid

Mobiiltelefonides ja teistes mobiilsetes seadmetes kasutatavatest mikrofonidest on enamik elektrostaatilised mikrofonid. Täpsemalt on tegu nn. elektreet mikrofonidega. Oma tööpõhimõttelt on sellist tüüpi mikrofonid väga sarnased klassikalisele kondensaatortüüpi mikrofonile. Vastupidiselt klassikalisele kondensaatormikrofonile, kus kondensaatorit pidevalt pingestatakse, kasutab elektreetmikrofon tootmisel membraanile elektrostaatilise välja abil antud laengut. Kui välja mõju eemaldatakse, jääb laeng membraanile alles, ning normaalsetel tingimustel (õhuniiskus alla 90%, toatemperatuur), püsib laeng väga kaua ning mikrofoni kasutamiseks pole vaja lisaks polariseerivat pinget. Pinget vajab vaid väljatransistor ning seegi pinge on nii madal, et 1.5 V patarei suudab väljatransistori laenguga varustada tuhandeteks tundideks. Taoline mikrofon on valitud mobiilsetesse seadmetesse peamiselt kahe aspekti tõttu. Ühelt poolt on selline mikrofon väga autonoomne, see ei vaja lisa pingestamist. Samuti on see oma ehituse ja eelpingestatuse tõttu gabariitidelt väga väike. Teisalt on elektreettüüpi mikrofon komponentide vähesuse ja odavuse tõttu suhteliselt odav ning seega on seda masstoodangus väga hea kasutada.

3. Elektrisignaali töötlus kahendarvuks

Käesolevas peatükis antakse lühiülevaate elektrisignaali töötlusest arvutile arusaavaks kahendarvuks. Veel kirjeldatakse selle protsessi käigus tekkivaid probleeme ja erinevaid karakteristikuid, mis antud protsessi tulemust määravad. Lisaks antakse ülevaade erinevatest heli formaatidest, millesse on võimalik ja sobilik heli salvestada.

3.1 Signaali töötlus

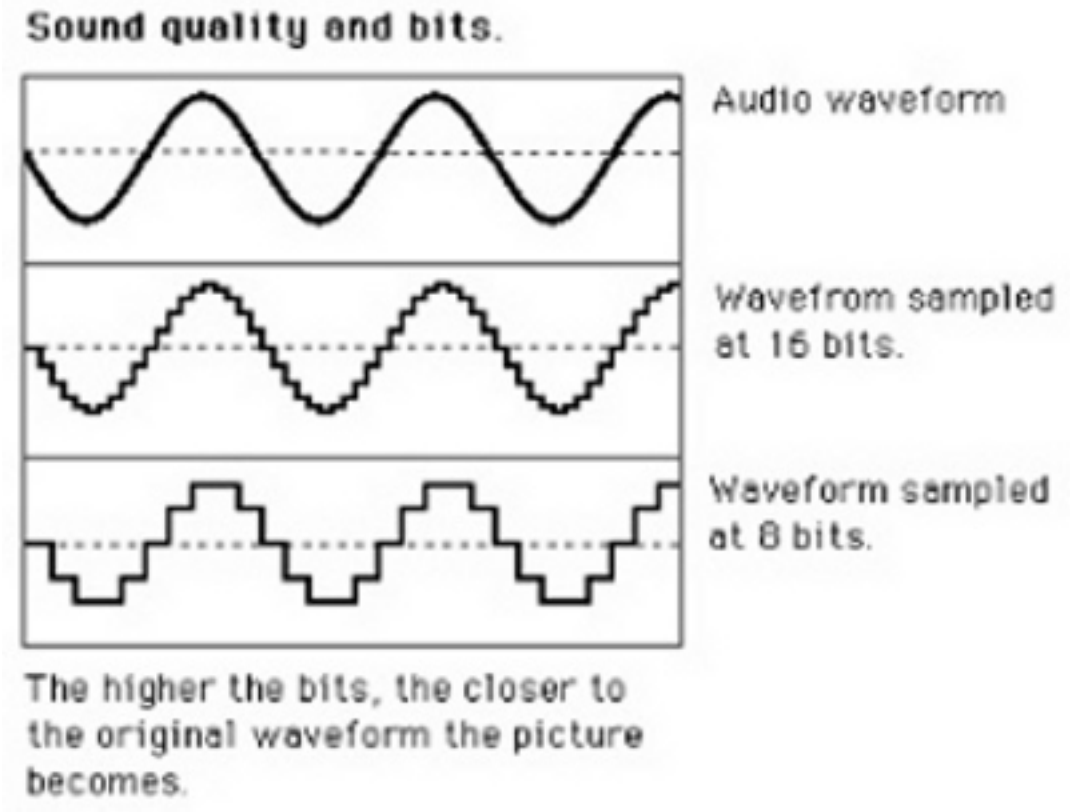
Tänapäeva arvuti on ehitatud nii, et selles olev mikroprotsessor töötleb andmeid kahendarvu esituses. Seega on vajalik, et mikrofonilt väljastatav elektrisignaal töödeldaks ümber kahendarvuks. Töötlemiseks on vajalik lisaseade mikrofoni ja arvuti vahel. Selleks seadmeks on analoog-digitaalmuundur. Analoog-digitaalmuundur on seade, mis võtab vastu analoogsignaali ja muundab selle digitaalsignaaliks, mida esitatakse kahendarvuna. Teisisõnu, mõõdab analoog-digitaalmuundur sisendsignaali väärtusi ja muudab need kahendarvuks. Siinjuures, on sisendväärtus ja väljundiks antav arv võrdelises seoses.

Kogu protsessi peamine probleem seisneb selles, et mikrofonilt tulev elektrisignaal on pidev, seega on võimalik seda signaali üksühele edasi anda kasutades kontiinumi võimsusega arvuhulki. Arvuti opereerib aga loenduvatel hulkadel. On aga teada, et kontiinumi võimsusega hulgad pole loenduvatega ekvivalentsed. Loenduv võimsus on kontiinumi võimsusest nõrgem. Seega pole analoogsignaali üksühene esitus kahendarvu kujul võimalik.

Walt Kester [5] kirjeldab oma raamatus protsessi, kuidas toimub elektrisignaali töötlus kahendarvuks. Pidevuse probleemi lahendamiseks kasutatakse pideva analoogisignaali diskreetimist. Selleks fikseeritakse ajavahemik ehk periood, mille tagant analoogsignaali väärtust mõõdetakse. Praktikas on väga levinud kasutada diskreetimise perioodi kujutamiseks hoopis selle pöördväärtust ehk sagedust. Diskreetimise sagedus määrab ära selle, kui hea lähend sisendsignaalile saadakse. Mida suurem on diskreetimissagedus, seda täpsem on väljund sisendsignaali suhtes. Lisaks arvestatakse diskreetimissagedusel Nyquist-Shannoni diskreetimisteoreemi [6], mis väidab, et diskreetimissagedus peab olema vähemalt kaks korda suurem sisendsignaali maksimum väärtusest. Sellega tagatakse, et salvestise taasesitamisel tehtav digitaalsignaali teisendus analoogsignaaliks, ei kaotaks ühtegi originaalsageduse väärtust. Tüüpiliselt kasutatakse diskreetimissageduseks 44 100 Hz, sest inimkõrvale kuuldav heli jääb vahemikku 20 – 20 000 Hz. Seega on mainitud sagedus inimkõrvale kuuldavast maksimumsagedusest natuke üle kahe korra suurem. Selline diskreetimissagedus on piisav, sest sellest alast välja jäävaid helisid inimene ei kuule ning, seega nende mitte salvestamisel hoitakse kokku salvestusruumi. Samuti parandatakse sellega heli

kvaliteeti. Nimelt tekivad kuulmisläve ülemisel piiri olevatel sagedustel inimesele valuaistingut tekitavad helid ning neid soovitakse vältida.

Teine oluline parameeter analoogsignaali diskreetimisel on kvantitatsiooniastmete arv [7]. Kvantitatsiooniastmete arv näitab ära iga mõõdetud väärtuse bitisügavuse [8]. Teisisõnu, kvantitatsiooniastmete arv näitab, kui mitmeks erinevaks väärtuseks on protsessi käigus võimalik analoogsignaali väärtus salvestada. Seega, mida suurem on kvantatsiooniastmete arv, seda täpsemalt on võimalik mõõdetav signaal salvestada. Kvantitatsiooniastmete arvu ja heli signaali diskreetimise täpsuse korrelatsiooni illustreerimiseks kasutatakse joonist 1.



Joonis 1. Helisignaali kujutamine diagrammil [8]. y-teljel on salvestise helisagedus ja x-teljel aeg. Ülevalt alla – originaalsignaal, 16 bitine digitaalsignaal, 8 bitine digitaalsignaal.

Jooniselt on selgelt näha, et suurem bitisügavus tagab analoogsignaali täpsema muunduse. Termin bitisügavus tuleneb sellest, et peamiselt määratakse analoog-digitaalmuundurite kvantitatsiooniastmete arv just bittides. Bittide teisendamisel kvantitatsiooniastmeteks kehtib võrdus $N = 2^n$, kus n on analoog-digitaalmuunduri bittide arv. Seega, kui tegu on näiteks 16-bitise analoog-digitaalmuunduriga, siis on diskreetimisel kasutada $2^{16} = 65536$ erinevat kvantitatsiooniastet. Tänapäeval kasutaksegi heli salvestamisel peamiselt 16-bitiseid muundureid. Väiksemate bittide arvu korral on salvestisel selgelt kuulda diskreetimisest tekkinud nn. “valge müra” (*white*

noise). Suuremat bitisügavust on ebapraktiline kasutada seetõttu, et inimesekõrv ei suuda eristada helis nüansse, mis jäävad 16-bitisest spektrist välja. Tihti pole nende nüansside salvestamine ka eesmärk. Samuti kasvatab bitisügavuse tõstmine salvestatava faili mahtu lineaarselt. Teisisõnu, tõstes bitisügavust kaks korda, suureneb ka salvestatava faili maht kaks korda.

Lisaks kahele eelpool kirjeldatud parameetrile, kasutatakse diskreetimisel ka kõrg- ja madalpääsu filterid [9]. Nende ülesanne on teatud sagedusest alla või ülespoole jäävate helide ära lõikamine, vältimaks ülalpool mainitud valuaistingut kõrgete sageduste puhul ning samuti tervist kahjustavat toimet infraheli sageduste puhul. Lisaks negatiivsetele mõjudele inimese tervisele, soovitakse kõrg- ja madalpääsu filtrite kasutamisega vältida heli moonutusi.

3.2 Salvestise formaadid

Heli salvestamisel kasutatakse peamiselt kahte tüüpi formaate: pakitud (*compressed)* ja pakkimata (*uncompressed*). Pakkimata formaadi puhul on tegemist salvestise formaadi muutusega. Teisisõnu, on pakkimata formaadis salvestised lihtsalt muutnud heli seadmele arusaadavaks, muutmata sealjuures heli salvestamisel kasutatud karateristikuid. Seega on pakkimata formaadis olevad helifailid küllaltki suured. Tuntumateks pakkimata salvestusformaatideks on WAV ja AIFF [10].

Pakitud formaadid jagunevad omakorda kaheks: kadudeta (*lossless*) ja kadudega (*lossy*) formaatideks. Kadudeta formaadid on pakkimata formaatide teisendus, mis ei kaota originaaliga võrreldes oma kvaliteedis. Siiski on võimalik kadudeta formaatide puhul märkimisväärselt salvestusruumi kokku hoida. Seda tingib asjaolu, et kadudeta formaatidesse salvestades kasutatakse täpselt nii mitut bitti, kui see on salvestatava heli jaoks vajalik. Teisisõnu, kui pakkimata formaatide puhul kulub ühe sekundi heli ja ühe sekundi vaikuse salvestamiseks sama palju salvestusruumi, siis kadudeta formaatide puhul ei kulu vaikuse salvestamiseks peaaegu üldse salvestusruumi. Seega võib väita, et kadudeta salvestusformaadid kasutavad salvestusruumi optimaalselt ja failid on seetõttu oluliselt väiksemad kui sama heli sisaldavad pakkimata formaadiga failid. Tuntumateks kadudeta formaatideks on Apple Lossless ja FLAC (*Free Lossless Audio Codec*) [10].

Kadudega formaadid on, nagu nimigi ütleb, kvaliteedi kadudega formaadid. Seega, kui viia originaal salvestis kadudega formaati ja sellest formaadist see tagasi teisendada pakkimata formaati, on tulemuseks kvaliteedi kadu, sest osa heli läheb kaotsi [10]. Peamiselt tingib heli kvaliteedi kao asjaolu, et kadudega formaati salvestamise käigus langetatakse salvestise bitisügavust, seega on vaja vähem salvestusruumi, kuid nagu ülalpool esitatud jooniselt nähtub, kaotab salvestis sellega osad sagedused. Peamiseks kadudega formaatide eeliseks on väga suur salvestusruumi kokkuhoid. Seega on kadudega formaadid peamisteks heli vorminguteks mobiilsetes seadmetes. Lisaks tingib kadudega formaatide kasutuse ka asjaolu, et mobiilsete seadmete kõlarid ei ole

võimelised edasi andma kõiki pakkimata või kadudeta vormingutes olevaid heli nüansse ning seega puudub praktiline mõte salvestada mobiilsetes seadmetes heli salvestusruumi rohkem kasutavatesse formaatidesse. Tuntumateks kadudega helisalvestus formaatideks on MP3 ja MP4.

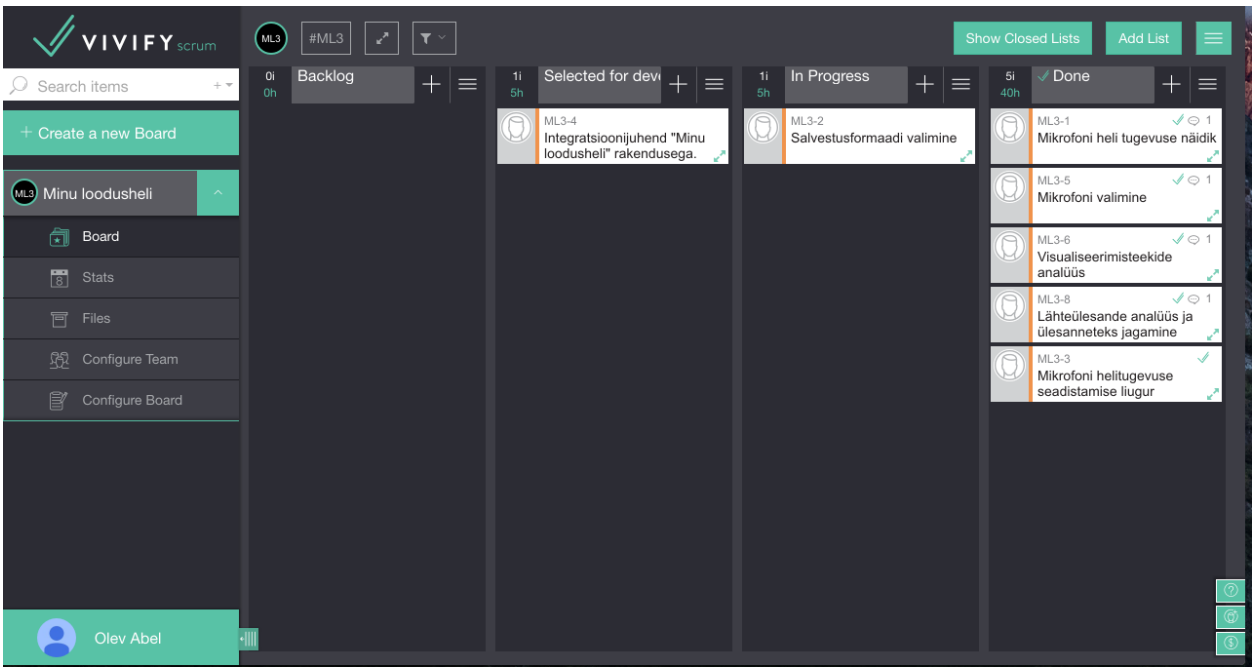
Praktilises osas tehtud rakenduses kasutatakse salvestusformaatidena ühte pakkimata formaati (WAV) ja ühte pakitud kadudega formaati (MP4). Sellise valiku tingis asjaolu, et kasutaja saaks valida kvaliteetse ja vähem kvaliteetse formaadi vahel. Kui kasutajal on suure salvestusruumiga seade ning ta soovib head salvestuskvaliteeti, on tal see võimalus olemas. Samas on mõeldud ka keskmisele kasutajale, kellel on suhteliselt piiratud salvestusruum ning seetõttu on talle oluline failide väike suurus.

4. Arendusmeetodite valik

Käesolevas peatükis antakse ülevaate antud bakalaureusetöö raames tehtud audiovisuaalse mooduli programmeerimisega seotud arendusmeetodite valikust. Peatükis käsitletakse lisaks programmeerimises kasutatud tehnoloogiatele ka lähteülesande analüütilist osa.

4.1 Väle tarkvaraarenduse meetod Kanban

Osama Al-Baik ja James Miller kirjeldavad oma artiklis Kanban meetodi põhiprintsiipe [11]. Kanban on väle tarkvaraarenduse meetod, mis on välja kasvanud logistilisest tarneahela meetodist, mis võeti esmakordselt kasutusele autotootja Toyota kontsernis. See meetod tagab tarneahela tõrgeteta töö, sest kogu meetod on ülesehitatud nn täppisajastusele. Teisisõnu, tähendab see, et järgmine tootmise etapp võetakse ette juhul, kui eelnev, millest ettevõetav etapp sõltub, on edukalt läbitud. Samuti minimeerib selline protsess tootmisel tekkiva praagi jõudmist kliendini. Kõik tootmises tekkinud defektiga tooted praagitakse järgmisel etapil välja. Tarkvara arendusse tõi Kanban meetodi David Anderson. Anderson on üritanud alates 2003. aastat muuta Kanbani meetodit tarkvaraarenduses kasutatavaks [11]. Järgnev joonis illustreerib Kanban töölauda



Joonis 2. Kanban töölaud keskkonnas Vivify Scrum [12].

Kanban töölaud (*board*) koosneb neljast tulbast. Teoorias liigub iga alamülesanne protsessi käigus tulpades ainult vasakult paremale. Selline ühesuunaline liikumine tagab töö voo ning on alus täppisajastusele. Vasakult paremale liikumine välistab ülesannete

kuhjumise ja hoiab ära olukorra, kus kõik alles jäänud alamülesanded sõltuvad pooleli olevatest.

Kõige vasakpoolsemas tulbas on lahendamata ülesanded (*backlog)*. Sinna kuuluvad ülesanded, mille kohta pole tehtud veel analüüsi või puudub neil ajaline hinnang. Kõik uued alamülesanded paigutatakse loomise käigus sellesse tulpa.
Vasakult teise tulpa kuuluvad ülesanded, millele on tehtud piisav analüüs ja määratud ajaline hinnang ning mis ootavad töösse minekut (*selected for development)*. Siia tulpa jõuavad ka need alamülesanded, mille kohta on analüüs tehtud, kuid mis ootavad kolmandate osapoolte ressursside taga.

Vasakult kolmandasse tulpa liiguvad ülesanded, mida hakatakse lahendama (*In progress)*. Selle tulba moodustavad pooleli olevad alamülesanded. Selles tulbas logitakse ka ülesande lahendamisele kulunud aeg.

Viimane tulp koosneb ülesannetest, mis on arendajate poolt lahendatud ja ootavad kvaliteedikontrolli poolt kinnitamist (*Done*). Kui kvaliteedikontroll ei leia lahenduses puudujääke märgitakse ülesanne lahendatuks. Vastasel korral tekitatakse puudujääkide kohta uus alamülesanne ning originaalse alamülesande juurde märgitakse, et tegu on veel lahendamata ülesandega.

Autor valis antud bakalaureusetöö raames valmistatava projekti teostamiseks Kanban meetodi, sest on tööalaselt sellega varem kokku puutunud. Samuti on Kanban meetod antud töö raames hea lahendus seetõttu, et võimaldab paindliku tööaega ning ei järgi nii kindlalt erinevaid tähtaegu, nagu seda teeb Scrum meetod. Sprintide puudumine küll raskendab üldist ajaplaneerimist, kuid vähendab ka planeerimisele kuluvat aega. Lisaks eelpool mainitule, on Kanban meetod hea kasutada projektides, mille korral tuleb tööle hakata funktsionaalsustega, mille kohta on vähe infot või nende funktsionaalsuste lahendamine on arendusmeeskonnale võõras. Nii ka käesoleva töö praktiline osa oli autorile tööle hakates võõras ning paikapidavaid ajahinnanguid oli seetõttu väga raske anda.

4.2 Platvormi valik

Autor otsustas audiovisuaalse mooduli programmeerida “Minu loodusheli” rakenduse Android versiooni prototüübile. Seda tingis asjaolu, et autor on varem kokku puutunud Andoridi arendusega ning samuti puudus autoril eelnev kogemus iOS platvormiga, millel on samuti “Minu loodusheli” rakendus olemas. Lisaks eelnevale kogemusele Andoridi vallas, valis autor Androidi platvormi ka seetõttu, et see on hetkel maailmas enimkasutatav mobiilne operatsioonisüsteem, moodustades ligikaudu 60 % kogu maailma mobiilsete seadmete operatsioonisüsteemides [13]. See asjaolu tingib väga laia kasutajaskonna ja väga suure hulga seadmeid. Seetõttu saab töö praktilise osa käigus

valminud moodulit testida erinevate tootjate seadmetel. Erinevatel seadmetel testimine on vajalik, sest erinevad tootjad kasutavad erinevat riistvara ning seetõttu on võimalik, et mõnel mobiilsel seadmel on mitu mikrofoni, mõnel vaid üks. Lisaks pole ühtset standardit erinevate failiformaatide toetamiseks, mistõttu on vajalik läbi testida erinevate tootjate seadmed, veendumaks, et soovitud failiformaadid on nendes seadmetes toetatud.

Lisaks riistvaralisele erinevusele, erineb tootjatel ka operatsioonisüsteem. “Puhas” Android (*Vanilla Android*) on vaid Google telefonidel. Teistel tootjatel on Google poolt arendatud Androidile peale pandud tootjaspetsiifiline kiht, mis enamasti muudab graafilist poolt, aga võib ka lisada teatavaid funktsionaalsusi. Taoline operatsioonisüsteemile lisakihtide juurde liitmine, võib aga tekitada operatsioonisüsteemis soovimatut käitumist. See on veel üks põhjus, miks tuleb Androidile arendatavat rakendust põhjalikult erinevate tootjate seadmetega testida.

Androidi operatsioonisüsteemi laialdase leviku tõttu on ka Android operatsioonisüsteemi kasutavate seadmete jõudlused väga erinevad. Selle asjaolu tõttu tuleb Androidi rakenduse programmeerimisel pöörata rohkem tähelepanu rakenduse koodi optimeerimisele. Hästi optimeeritud kood tagab rakenduse tõrgeteta töö ka väiksema jõudlusega seadmetel.

4.3 Arendustööriistad

Autor kasutas käesoleva bakalaureusetöö raames tehtava praktilise töö programmeerimisel mitmeid arendustööriistu. Arendustööriistade eesmärk on teha arendaja töö kiiremaks, efektiivsemaks ja lihtsamaks. Selle saavutamiseks on kasutusel mitmed erinevad tehnoloogiad.

Käesoleva töö raames on integreeritud arenduskeskkonnana kasutatud (*IDE – Itegrated development environment*) Andorid Studiot. Anrdoid Studio pakub arendajale võimalust arendada mugavalt nii rakenduse äriloogika poolt kui ka graafilist liidest. Nimelt võimaldab Android Studio näha graafilise liidese objektide paigutust ilma, et arendajal oleks vaja kogu rakendus ära kompileerida ning seadmele paigaldada. Lisaks efektiivsetele ja mugavatele graafilise liidese arendusvõimalustele, pakub Android Studio uusim versioon ka tehnoloogiat *Instant Run.* Tegu on Java baitkoodis töötava mehhanismiga, mis võimaldab programmikoodis teha muudatusi ja ilma uuesti kompileerimata neid muudatusi näha. See tehnoloogia teeb väikeste paranduste sisseviimise kiiremaks ning minimeerib arendaja nö “tühja“ aega ehk aega, kus arendaja ootab kompilaatori järel. Siiski ei ole võimalik kõiki muudatusi ilma uuesti kompileerimata sisse viia. Seega uute funktsionaalsuste või suuremate loogikamuudatuste korral, tuleb programmi kood uuesti täielikult kompileerida. Lisaks *Instant Runile* on Android Studio näol tegemist nö “targa” kompilaatoriga arenduskeskkonnaga. Nimelt kompileeritakse programmikoodi muutumisel võimalikult väike osa kogu programmi koodist. Teisisõnu, kui muudetud kood ei mõjuta mingilgi viisil mingit varasemat funktsionaalsust, siis selle varasema

funktsionaalsuse koodi uuesti ei kompileerita. Sellega hoitakse sammuti kokku kompileerimisele kuluvat aega.

Teiseks oluliseks arendustööriistaks, mida praktilises osas kasutatakse, on Vivify Scrum [12]. Tegu on vabavaralise Kanbani töölauaga. Selle töölaua abil märkis autor üles analüüsil tekkinud alamülesanded. Lisaks ülesannete detailsele kirjeldusele, on Vivify Scrumis võimalik muuta alamülesande prioriteeti, mis liigutab alamülesanded tehtava tööde nimekirjas ülespoole. Samuti on võimalik iga alamülesande juurde lisada kommentaare. Üks Kanban meetodi seisukohalt oluline võimalus on Vivify Scrumis veel. Nimelt on võimalik igale alamülesandele määrata loeteluna täitmistingimused (*definition of done)*. Selle funktsionaalsuse eesmärk on avastada puudujääk arenduse käigus ning samuti, on täitmistingimuste kirjapanek kvaliteedikontrollile väga heaks lähtekohaks. Lisaks eelpool mainitud võimalustele, saab Vivify Scrum keskkonnas lisada alamülesannetele ajalisi hinnanguid. Samuti on võimalik muuta ajalise hinnangu ühikuid ning logida tegelikkuses kulunud aega.

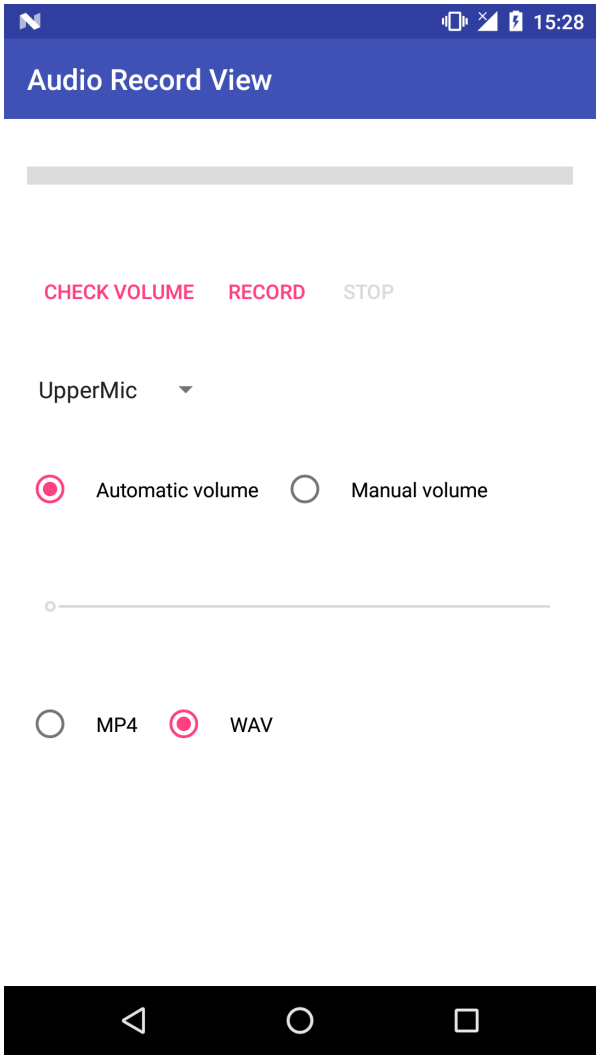
Kolmandaks arendustööriistaks kasutatakse koodihoidlat (*code repository*) GitHub [14]. Koodihoidla võimaldab arendajal hoida programmikoodi serveris, tehes nii mitme arendajaga projektide töökorralduse lihtsaks ja mugavaks. Samuti täidab koodihoidla varundamise eesmärki. Kui arendaja lokaalne programmi kood kaob või arendaja viib sisse muudatusi, mis lõhub mingi funktsionaalsuse, on arendajal võimalik koodihoidlast võtta stabiilne programmikoodi seis. Samuti on Git koodihoidlates võimalik tekitada puukujuline struktuur, mille ühes harus hoitakse näiteks kliendile minevat stabiilset ja versioneeritud programmikoodi ning teises harus on arendusel kasutatav ning pidevalt muutuv kood.

5. Audiovisuaalse mooduli programmeerimine

Selles peatükis antakse ülevaate bakalaureusetöö praktilises osas tehtu kohta. Lisaks üldisele kirjeldusele, põhjendatakse ka mõnede tehnoloogiate ja meetodite kasutust ning avatakse tähtsamate programmikoodi osade sisu lugejale. Lisaks kirjeldatakse käesolevas peatükis praktilise töö kodeerimisprotsessi, kasutades Kanban meetodit.

5.1 Tähtsamad autori poolt kirjutatud mooduli osad

Audiovisuaalse mooduli tähtsamad komponendid, mida tuli programmeerida olid: salvestatava heli tugevuse näidik, salvestatava heli tugevuse muutmise liugur, salvestatava faili formaadi valik, mikrofoni valik, kui seadmel on neid mitu ja nende komponentide integreerimine kolmanda osapoole teegiga, mis kuvab salvestise helitugevused ekraanile diagrammi kujul.



Joonis 3. Salvestusvaate ekraanitõmmis.

5.1.1 Salvestatava heli tugevuse näidik

Autori ülesanne oli lõppkasutajale audiomooduli vaatesse navigeerimisel kuvada heli tugevust, mis jõuab valitud mikrofonini. Selle saavutamiseks kasutatakse Androidi spetsiaalset heli salvestamiseks mõeldud salvestaja klassi AudioRecord [15]. See salvestaja klass võimaldab salvestada andmed puhvrisse ning sealt need kirjutada sobivasse faili ümber. Just kergesti ligipääsetav puhver oli asjaolu, mille tõttu otsustas autor kasutada AudioRecord klassi ja mitte universaalsemat MediaRecorder klassi [16]. Puhver võimaldas näidata helitugevuse muutust sujuvalt ning samuti oli puhvri olemasolu abiks animatsiooni loomisel. Nimelt oli võimalik puhvrist järjest väärtusi lugeda ning neid kasutajale kuvada. Siiski pidi arvestama kahe asjaoluga. Esiteks, kogu heli salvestamine ja puhvri täitmine toimub taustalõimes (*background thread)*, välistamaks graafilise liidese hangumist. Teiseks, tänapäeva mobiiltelefonid on varustatud võimsate protsessoritega, mistõttu puhvri läbimine tavaolukorras toimub millisekunditega. Seega oli vaja animatsiooni korralikuks toimimiseks programmeerida puhvri väärtuse kuvamine kasutajale mõistliku viitega. See tagab sujuva ja inimesele jälgitava animatsiooni.

5.1.2 Salvestatava heli tugevuse muutmise liugur

Lisaks eelpool mainitud salvestatava heli tugevuse näidikule, on töös valmis programmeeritud ka salvestatava heli tugevuse muutmiseks liugur. Kasutaja saab selle liuguri abil valida, kui tugevat heli ta salvestada soovib. Antud liuguri kasutamise tingib asjaolu, et tihti ei ole looduses võimalik heliallikale piisavalt lähedale minna ning salvestis jääb liiga vaikne, et see oleks normaalsetes tingimustes ja ilma eri vahenditeta kasutatav. Seega on liuguri abil kasutajal võimalik muuta salvestatava heli tugevust. Tegu on salvestise töötlusega. Nimelt pole sisendheli tugevust võimalik enne selle digitaliseerimist kuidagi teada. Seega pole võimalik enne salvestamist heli tugevusega manipuleerida. Alles salvestamise hetkel on võimalik üheselt öelda puhvris oleva digitaliseeritud andmete baitide väärtused ning neid vastavalt manipuleerida. Heli tugevuse muutus on saavutatud lihtsa matemaatilise tehte abil. Nagu diskreetimise peatükist selgus, on salvestatava heli bait esituses olevate andmete väärtus seda suurem, mida suurem on selle heli tugevus. Seega piisab helitugevuse tõstmiseks või langetamiseks baidi väärtuse muutmisest. Nimelt on jagatud liugur sajaks osaks ning antud igale osale väärtus lõigust [0.5,2]. Saadakse valem 0.5 + 0,015 * N, kus N on liuguri väärtus. Selline implementatsioon tagab, et kui kasutaja on liuguri liigutanud miinimumini, on salvestatava heli tugevus normaaltugevusest kaks korda vaiksem. Kui aga kasutaja on liuguri liigutanud maksimumini, on salvestatava heli tugevus normaaltugevusest kaks korda valjem. Siin tekib piirang, sest ühes baidis on võimalik salvestada 2^8 erinevat väärtust. Seega on võimalik ühte baiti salvestada väärtuseid 0..255. Seega on vajalik lisada baidi väärtuse suurendamise funktsiooni ka tingimus, et kui muudetava baidi väärtus läheb suuremaks kui 255, siis tuleb see fikseerida 255 peale.

Peale manuaalse helitugevuse muutmise, on kasutajal võimalik valida ka seadme enda poolt diskreeditud helitugevus. Selleks ei pea kasutaja tegema muud, kui vajutama valikunuppu “Automatic volume”. Selline funktsionaalsus on lisatud selleks, et kasutajad ei peaks iga kord helitugevuse muutmise liugurit kasutama. Näiteks juhul, kui salvestatava heli originaaltugevus on piisav ning ei vaja manipuleerimist.

5.1.3 Salvestise formaadi valik.

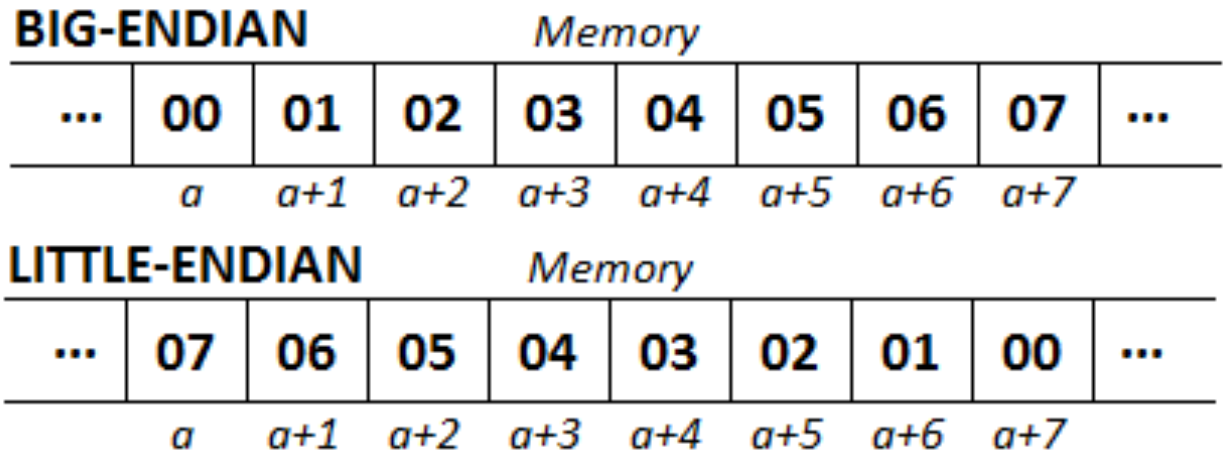
Formaadi valik on lahendatud tavalise rippmenüüga (*Spinner*). Valikus on nii WAV formaat kui ka MP4 formaat. Kuna Androidi salvestaja klass AudioRecord antud formaate ei toeta, siis tuli formaatidesse teisendamine ise kirjutada. WAV faili korral tuli lisada faili algusesse RIFF päis. RIFF päis on 44 baidist koosnev päis, mis hoiab informatsiooni nii salvestise suuruse kui ka muude karakteristikute kohta. Nimelt sisaldub selles päises nii kanalite arv, diskreetimissagedus kui ka bitisügavus. Järgnev tabel annab ülevaate RIFF päises olevatest baitidest:

Positsioon	Näidis väärtus	Kirjeldus
1-4	“RIFF”	Märgib, mis tüüpi failiga on tegu. Kõik sümbolid on ühe baidi suurused
5-8	Faili suurus	4 baidine(*int*)täisarv. Salvestise suurus peab olema sellel hetkel teada.
9-12	“WAVE”	Faili päise tüüp.
13-16	“fmt ”	Marker. Sisaldab endas kolme ühe baidi suurust sümbolit ja nende lõpus tühisümbolit.
17-20	16	
21-22	1	Formaadi tüüp (1 vastab PCM formaadile). Kahebaidine(*short*) täisarv
23-24	1	Kanalite arv. 1- mono, 2- stereo. Kahebaidine täisarv.
25-28	44100	Diskreetimissagedus hertsides. 4 baidine täisarv
29-32	176400	(Diskreetimissagedus * bitisügavus * kanalite arv) / 8. 4 baidine täisarv.
33-34	4	(Bitisügavus * kanalite arv) / 8. Kahebaidine täisarv

35-36	16	Bitisügevus. Kahebaidine täisarv
37-40	“data”	Marker. Märgib salvestise algust. Neli ühebaidist sümbolit.
41-44	882000	Salvestise osa suurus

Tabel 1. RIFF päise detailne kirjeldus [17].

RIFF päise kokkupanekul tuleb silmas pidada baitide järjekorda. Nimelt on võimalik esitada baidijada kahes järjekorras: Kas mälus on baidijada salvestatud vasakult paremale (*big-endian*) või paremalt vasakule *(little-endian)* Järgnev joonis demonstreerib nende kahe erinevust:



Joonis 4. Baidijadade erinev järjestus seadme mälus [18].

RIFF päisesse on vaja baidid salvestada pööratud järjestuses. Selle saavutamiseks on rakenduses kasutatud Javasse sisse ehitatud ByteBuffer [19] konstruktsiooni. Sellele on võimalik mugavalt ette anda baitide järjekord ning samuti mitme baidist arvu on vaja salvestada. Järgnev ekraanitõmmis näitab meetodite kasutust:

```
private static byte[] intToByteArray(int i)
{
    return ByteBuffer.allocate(4).order(ByteOrder.LITTLE_ENDIAN).putInt(i).array();
}

// convert a short to a byte array
public static byte[] shortToByteArray(short data)
{
    return ByteBuffer.allocate(2).order(ByteOrder.LITTLE_ENDIAN).putShort(data).array();
}
```

Joonis 5. Erineva suurusega arvude teisendamine pööratud baidijadaks.

5.1.4 Seadme mikrofonide valik

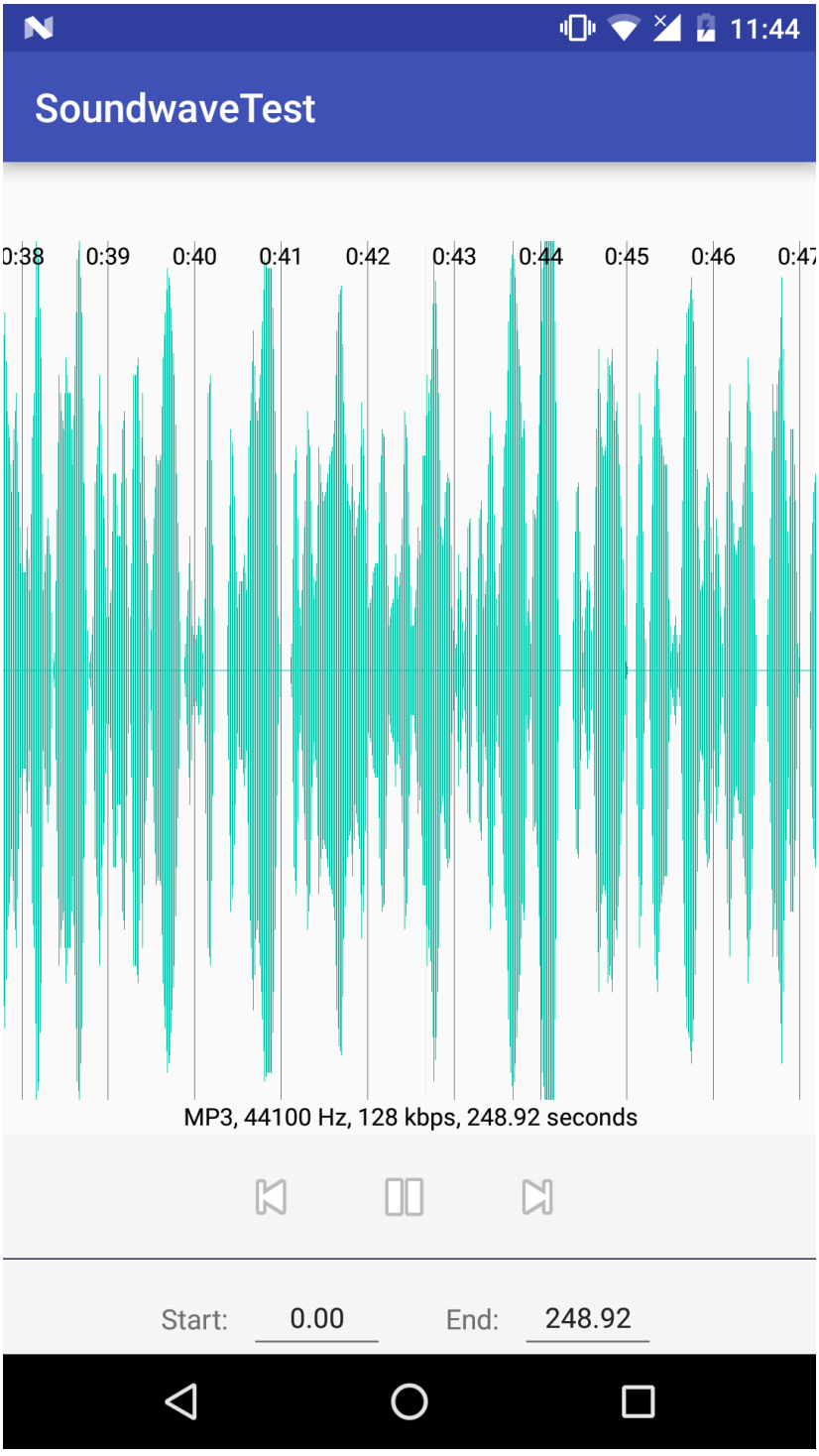
Mikrofoni valik seadmes tehti sarnaselt salvestusformaadi valikule tavalise rippmenüüga. Kuna Android seadmes on võimalik mitme mikrofoni olemasolu, siis on vajalik kasutajale kuvada valikut, millist mikrofoni ta soovib kasutada. Valikus kaks mikrofoni: kõne mikrofon ja kaamera mikrofon. Kõne mikrofon asub seadme allosas ja ongi mõeldud kõnedes kasutamiseks. Üldjuhul on selle mikrofoni kvaliteet parem kui kaamera mikrofonil, kuid vahe ei ole enamasti väga märkimisväärne. Peamine erinevus seisneb tavaliselt mikrofoni tundlikkuses. Kaamera mikrofoni kasutatakse videote salvestamisel, kus seadet hoitakse erinevates asendites, ning võib juhtuda, et hoidmisega kaetakse kinni kõne mikrofon. Kaamera mikrofon asub enamasti seadme ülaosas.

Mikrofoni valik Androidi rakenduses on tehtud väga mugavaks. AudioRecord salvestaja klass võimaldab salvestaja initsialiseerimisel määrata sisendi tüüpi. Sisenditüübid on operatsioonisüsteemi poolt ette antud nii, et õige sisendi kasutamiseks on vaja teada vaid sisendile määratud nime.

5.2 Visualiseerimisteegi ühendamine mooduliga

Programmeeritava audiovisuaalse mooduli üks osa on visualiseerimisteek. Tegu on kolmanda osapoole poolt kirjutatud teegiga [20], mis võimaldab salvestise sisse lugemist ja selle graafilist kujutamist. Samuti on võimalik graafilikult valida erinevaid salvestise osi ning neid ette mängida. Lisaks sellele on võimalik kerida graafikul salvestist ning suumida, et näha täpsemalt salvestise osa helitugevust. Tegu on eraldi projektiga, mistõttu oli mooduli loomisel lihtsam võtta esialgne visualiseerimisteek ning seda täiendada vajaliku äriloogikaga. Samuti oli vaja kirjutada navigatsioon

visualiseerimisvaate ja salvestamisvaate vahel ning anda salvestaja vaatest kaasa salvestatud faili nimi, et seda saaks visualiseerimisvaates sisse lugeda ning kuvada.



Joonis 6. Visualiseerimisteegi [18] ekraanitõmmis

6. Kokkuvõte

Käeoleva töö tulemusena loodi Android moodul, millega on võimalik kasutajal saavutada suurem kontroll heli salvestamise üle kui Androidi sisse ehitatud vahendid seda võimaldavad. Sealjuures oli oluline osa võimalusel muuta heli salvestise formaati ja heli tugevust. Lisaks uutele funktsionaalsustele, on loodud rakenduses võimalik visualiseerida salvestis ning seda lihtsalt ja mugavalt hallata: kuulata valitud osi salvestusest, vaadata salvestuse heli tugevust kujutatuna ekraanil ning soovi korral kerida salvestust kasutades selleks visuaalset graafikut. Peale praktilisele rakendusele analüüsiti töö tulemusena erinevaid salvestusvõimalusi ja salvestusformaate Androidi platvormil ning toodi välja sobilikud, lähtudes töö eesmärgist ehk loodusvaatluse eripärast. Veel anti töö tulemusel ülevaade väledast tarkvara arenduse meetodist Kanban ning toodi välja selle sobilikud küljed arendamaks Androidi mobiilirakendust.

7. Kasutatud materjalid

[1] Erki Suurjaak, “Mikrofon” referaat, 3-7 (1999).
[2] Lembit Abo, “Elektroonikakomponendid”, 203-205 (1997).
[3] http://www.stokowski.org/Development_of_Electrical_Recording.htm, (11.04.2016).
[4] Elektrimahtuvus, http://www.ene.ttu.ee/leonardo/elektro_alused/5Elektrimahtuvus.pdf, (15.04.2016).
[5] Walt Kester, “The Data Conversion Handbook”, 3.39 – 3.41 (2005).
[6] Why do we need Analog to Digital converters, http://courses.me.berkeley.edu/ME102B/Past_Proj/f03/Proj6/TMS320LF2407A_Documents/Intro-ADC.pdf (12.04.2016).
[7] Signaali muundamine, http://www.e-uni.ee/e-kursused/eucip/haldus/313_signaali_muundamine.html, (12.04.2016).
[8] Sound quality and bits, http://blogs.heraldo.es/ciencia/files/2012/06/quantization-0-400.jpg, (19.04.2016).
[9] Low-pass Filters, http://www.allaboutcircuits.com/textbook/alternating-current/chpt-8/low-pass-filters/, (15.04.2016).
[10] Kirk McElhearn, “Everything you need to know about digital audio files”, Macworld, March 2016, Vol. 33, Issue 3
[11] Osama al-Baik, James Miller, “The kanban approach, between agility and leanness: a systematic review”, Empirical Software Engineering, December 2015,Vol. 20,Issue 6, pp 1861-1897.
[12] Vivify Scrum tööriist, https://www.vivifyscrum.com/feature/how-it-works,(29.04.2016).
[13] Mobile/Tablet Operating System Market Share https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1, (19.04.2016).
[14] GitHub koodihoidla, https://github.com/, (10.05.2016).
[15] AudioRecord, http://developer.android.com/reference/android/media/AudioRecord.html, (22.04.2016).
[16] MediaRecorder, http://developer.android.com/reference/android/media/MediaRecorder.html, (22.04.2016).
[17] WAV faili päise loomine, http://www.topherlee.com/software/pcm-tut-wavformat.html, (28.04.2016),
[17] Erinevad baidijadade järjestused, http://www.ibm.com/developerworks/library/l-ibm-xl-c-cpp-compiler/image001.png, (28.04.2016).
[19] ByteBuffer, https://docs.oracle.com/javase/7/docs/api/java/nio/ByteBuffer.html, (28.04.2016).
[20] Waveform view, https://github.com/Semantive/waveform-android,(10.05.2016).

Lisad

1. Funktsionaalsed nõuded

- Kasutajal peab olema võimalik käivitada ja peatada salvestamist.
- Kasutajal peab olema võimalik muuta salvestatava heli tugevust.
    - Muutmisel peab olema võimalik valida käitsi ja automaatse variandi vahel.
- Kasutajal peab olema võimalik testida helitugevust näidiku abil.
    - Näidik peab kuvama helitugevusi.
    - Näidik peab reageerima helitugevuse muutusele sujuvalt.
- Kasutajal peab olema võimalik salvestada heli kahte erinevasse formaati.
    - WAV formaat
    - MP4 formaat
- Kasutajal peab olema pärast salvestamist võimalik oma salvestist näha graafikul.
    - Graafik peab olema keritav.
    - Graafikul peab olema võimalik valida lõike.
    - Graafik peab olema suumitav.
- Kasutajal peab olema võimalik valida erinevate seadme mikrofonide vahel.
    - Valik peab olema võimalik vaid siis, kui seadmel on mitu mikrofoni.

2. Mittefunktsionaalsed nõuded

- Moodul ei tohi kokku joosta ega kasutaja rakendust kokku jooksutada.
- Mooduli arendaja peab mooduliga kaasa panama integratsiooni dokumendi.
- Moodul peab olema kasutatav eraldi rakendusena.
- Moodul peab töötama alates Andorid versioonist 4.3

3. Lähtekoodi hoidla aadress

https://github.com/olevabel/WaveFormAndMic/tree/master

4. Audiovisuaalse mooduli integreerimisjuhend

Töö tulemusena tekkinud audiovisuaalset moodulit on lihtne integreerida ükskõik millisesse Andorid projekti. Järgnevalt toodud juhised eeldavad, et arendaja kasutab Android Studio arenduskeskkonda:

1. Tõmba koodihoidlast alla audiovisuaalse mooduli lähtekood
2. Lisa moodul oma projekti.
    a. Vali menüüst File-> New -> Import Module-> Vali terve alla laetud projekti AudioVisualizationModule kaust
    b. Oota kuni Gradle on sünkroniseerimise lõpetanud
3. Mooduli kasutamiseks kutsu soovitud kohas välja AudioRecordActivity

a. Activity väljakutsel tekitatakse AudioRecordFragment, mida saad oma soovi järgi disainida
b. Faili salvestamise kaust on kirjutatud staatilisse välja DIR_NAME
c. Salvestist sisaldava faili nimi on kirjutatud staatilisse välja FILENAME

5. Lihtlitsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Olev Abel**,
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
Audiovisuaalse mooduli programmeerimine "Minu loodusheli" Android rakenduse näitel,
(*lõputöö pealkiri*)

mille juhendajaks on Marko Peterson ja kaasjuhendajaks Neeme Kahusk ,
(*juhendaja nimi*)

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **12.05.2016**