TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutisüsteemide instituut

Rain Kilkson 155682IASB

NUTISEADME KASUTUSLAIENDUSED(SÕIDUKAAMERA NÄITEL)

Bakalaureusetöö

Juhendaja: Vladimir Viies
Phd

Tallinn 2019

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: Rain Kilkson

27.05.2019

Annotatsioon

Töö eesmärgiks on tutvustada nutiseadmete kasutusvõimalusi, Android operatsioonisüsteemi arengut ja autori poolt loodud kaamerarakendust ja selle loomise protsessi rakenduse loomise selgitamiseks Android Studio arenduskeskkonnas.

Töö jaguneb kolmeks loogiliseks peatükiks, esimeses peatükis käsitletakse üldiselt mobiiltelefonide nutirakendusi ja tutvustatakse Android Operatsioonisüsteemi ja selle arengut. Teises peatükis käsitletakse Rakenduse loomise protsessi Android Studio arenduskeskonnas ning kolmas peatükk tutvustab autori loodud rakendust ja selle ülesehitust.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 34 leheküljel, 5 peatükki, 4 joonist, 1 tabelit.

Abstract

Smartphone possibilities (based on example of a car camera)

The goal of given Thesis is to give an overview of Smartphone possibilities, development of Android Operation System and how to create an application for Android based on an application created by the author, which gives a possibility to stream Live video from a camera to Smartphone. The purpose of developed application was to create an additional purpose for the Smartphone by using the Screen of a Smartphone to view a live stream from the camera.

Application was created for the idea of using as a alternative car camera giving the potentiality to use a camera mounted in car without an additional screen for it, but the application can be used additionally for different purposes, for example home security system.

The thesis is in Estonian and contains 34 pages of text, 5 chapters, 4 figures, 1 tables.

Lühendite ja mõistete sõnastik

RTSP	Real Time Streaming Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
RTP	Real Time Protocol
RTCP	Real Time Control Protocol
JIT	Just-in-time compilation

Sisukord

Jooniste Loetelu

Tabelite Loetelu

1 Sissejuhatus

Nutiseadmed on arenenud viimaste aastatega lihtsast seadmest millega on võimalik helistada, seadmeks mille kasutusvõimalused on peaaegu samaväärsed sülearvutiga. Nutiseadmete eeliseks on just selle kompaktsus ja tööaeg, ühe laadimisega on võimalik kasutada seadet keskmiselt 1-2 päeva ja kanda seda endaga alati kaasas.

Käesolev bakalaureusetöö on edasiarendus Tarkvara projektist, mille eesmärgiks oli kaardistada sõiduteel asuvad teedefektid salvestades kaameraga ülesse teepind, ning lisades sinna juurde ka defekti täpne asukoht defektide hilisema analüüsi kergendamiseks. Töö käigus tekkis vajadus filmitava video reaalajas vaatamiseks, et vajadusel korrigeerida kaamera asukohta ja filmitavat ala. Antud töö eesmärk on analüüsida Android nutitelefoni rakendusvõimalusi sõidukaamera näitel , kus kasutajal on võimalik kasutada eraldiseisva ekraani asemel kaamerapildi vaatamiseks isiklikku Android nutitelefoni.

Bakalaureusetöö praktiline osa on loodud programmeerimiskeeles java, kasutades rakenduse arendamiseks Android Studio arenduskeskkonda. Autor tutvustab töös Android operatsioonisüsteemi, selle arengut , erinevaid kasutusvõimalusi ja rakenduse loomist sõidukaamera näitel. Samuti on töös välja toodud protokollid, mida tavalised kaamerad kasutavad pildi reaalajas edastamiseks.

2 Mobiiltelefonide nutirakenduste kasutamine ja võimalused

Tänapäeval on igal inimesel taskus nutitelefon , nii mõnelgi inimesel on neid isegi mitu. Telefoni algne idee, kasutada telefoni helistamiseks, on ajaga vaikselt muutunud mõnede jaoks isegi teisejärguliseks. Aastal 1969 jõudis inimene esimest korda kuu peale. Reisi ajal oli nende kosmoselaeva peamine arvuti vahemälu vaid 64 Kbit ja protsessori kiiruseks 0.043 MHz, samas kui tänapäevane keskklassi nutitelefon(nt Oneplus 5) omab 6 Gb vahemälu ning

protsessori kiiruseks on kuni 2.45 GHz, mis on eelnevast kordades suurema jõudlusega.

See tähendab, et isegi nõrgema nutitelefoni abil on võimalik seda arvutusjõudlust õigesti kasutades teha peaaegu kõike.[1], [2]

Androidi operatsioonisüsteem jõudis esmakordselt turule 2009 aasta lõpus vabavarana ja hetkel on jõutud versioon 8.1 juurde, mis kannab nime Oreo. Androidi operatsioonisüsteemi arendab Google ning 2017 aastal kasutas seda vähemalt 2 miljardit seadet.

Peamine võimalus Androidi nutitelefonis rakenduste ja erinevate võimaluste loomiseks on Android Applications ehk rakendused mida on võimalik arendada Android Software development Kit-is (SDK) kus peamiseks programmeerimis keeleks on Java. SDK-s on võimalik Java kombineerida ka C ja C++ keeles. Alates mai 2017 on võimalik Androidi rakendusi arendada ka programmeerimiskeeles Kotlin. Android SDK arenduskeskkond võimaldab arendatavaid rakendusi hõlpsalt korrigeerida ja analüüsida. Lisaks on võimalik arendatavat programmi käivitada Android Simulaatoris, mis tähendab, et arendatavat programmi ei pea arendamisel katsetama füüsilise seadme peal. Algselt toetas Google ka arendamist Eclipse-i integreeritud arenduskeskkonnas, kuid 2014 väljastas Google Android Stuudio, mis on põhiliseks Androidi rakenduste arenduskeskkonnaks. Androidil on üha suurenev kolmandate osapoolte rakenduste arv, mida kasutajad saavad enda nutitelefoni alla laadida Google Play Store rakenduse kaudu. Sama keskkonna kaudu on võimalik alla laadida ka arendaja poolt väljastatud uuendusi rakendusele. Enamus olemasolevatest rakendustest on tasuta kättesaadavad, kuid sisaldavad endas reklaame või on tasuta kättesaadav vaid teatud osa rakenduse sisust. Suurim osa rakendustest on nutitelefonidele

mõeldud mängud, kuid samuti on olemas ka pangarakendused, mis võimaldavad teha enamus vajalikke toiminguid mobiilis mugavamalt kui vastava panga veebileheküljel.

Android rakenduste loomisel on oluline rakenduste minimaalne energiakasutus, sest nutitelefonid on akutoitega ning mida väiksem on rakenduste optimaalne energiakasutus, seda kauem on võimalik seadet ühe laadimiskorraga kasutada. Selle saavutamiseks piiratakse rakenduste, mida hetkel ei kasutata juurdepääsu internetile kui ka jõudlust.

Erinevatel Androidi nutitelefonidel on erinev hulk riistvara komponente, tavaliselt on nendeks kaamera telefoni tagaküljel, aina rohkem on turule jõudmas ka kahe tagumise kaameraga nutitelefone, mis pildi kvaliteedi ja detailsuse tagamiseks kasutavad mõlemat kaamerat, kaamera esiküljel videokõnede tegemiseks, GPS, Bluetooth, NFC, termomeetrid, kiirendusandurid, kõlarid, mikrofon, güroskoop ja puutetundlik ekraan. Lisaks on olemas nutitelefone millel on lisaks eelnevale ka termokaamera, baromeeter ja muud vajalikud spetsiaalsed riistvarakomponendid. Algselt nõudis Android seadmelt näiteks helistamiseks vajalikku riistvara(mikrofon, kõlar) kuid nüüdseks on nende olemasolu valikuline, sest Androidi operatsioonisüsteemi saab kasutada lisaks nutitelefonidele ka tahvelarvutites, nutikellades ja teistes seadmetes.

2.1 Mobiilseadmete operatsioonisüsteemid

Mobiilseadmete operatsioonisüsteemide puhul on populaarseim android, mida kasutab umbes 80% kõikidest seadmetest, selle tingib peamiselt suur arv erinevaid mitte mobiilseid seadmeid mis kasutavad android operatsioonisüsteemi, näiteks nuti kellad, televiisorid, külmkapid ja hulgaliselt muid seadmeid. Lisaks android operatsioonisüsteemile on olemas ka Apple poolt arendatav IOS, mida kasutavad kõik Apple poolt loodud tooted alates Iphone-de ja lõpetades Iwatch-idega. Need kaks on kõige tuntumad operatsioonisüsteemid ning moodustavad kahekesi umbes 95% kõikidel seadmetel jooksvatest operatsioonisüsteemidest, kuid lisaks nendele eksisteerivad näiteks ka Blackberry OS, Windows Phone, Symbian. Mobiilseadmete operatsioonisüsteemi puhul on oluline kasutajale olemasolevate rakenduste arv ning kuna rakendusi luuakse keskkondadesse kus on kõige suurem võimalik kasutajate hulk , tähendab see, et mida populaarsem on antud süsteem, seda rohkem loovad arendajad antud süsteemile rakendusi ja see omakorda tõmbab juurde veelgi rohkem kasutajaid.

Selles töös keskendub autor peamiselt Android operatsioonisüsteemiga nutitelefonidele jättes kõrvale Apple IOS, Nokia Symbian, BlackBerry ja teised nutitelefonidele mõeldud operatsioonisüsteemid.

2.2 Androidi versioonid

Versioon	Nimetus	API	Jaotus
2.33-2.3.7	Gingerbread	10	0,30%
4.0.3-4.0.4	Ice Cream Sandwich	15	0,40%
4.1.x	Jelly Bean	16	1,50%
4.2.x		17	2,20%
4.3.x		18	0,60%
4.4	KitKat	19	10,30%
5.0	Lollipop	21	4,80%
5.1		22	17,60%
6.0	Marshmallow	23	25,50%
7.0	Nougat	24	22,90%
7.1		25	8,20%
8.0	Oreo	26	4,90%
8.1		27	0,80%
9.0	Pie	28	

Tabel 1. Androidi versioonide kasutus

Androidi platvormi uus versioon tuleb tavaliselt välja iga aastaselt kuid seadmeteni jõuab uus versioon tavaliselt paari kuu pärast. See on tingitud erinevate seadmete riistvara erinevustest ja tootjatel kulub aega uue versiooni oma süsteemile kohandamiseks. Tootjad uuendavad tavaliselt vaid oma kõige uuemaid telefone uue Androidi versiooniga, mistõttu jäävad vanemad nutitelefonid tihti vanema platvormi peale. Lisa 1 on näha, et kõige rohkem seadmeid kasutab Androidi versioone Marshmallow ja Nougat mis tulid välja vastavalt 2015 ja 2016 aastal. Seda on eriti oluline jälgida rakenduste arendamisel, sest rakendus mis on arendatud uuemale versioonile, ei pruugi olenevalt oma lahendustest töötada

madalamatel versioonidel, samas kui rakendused mis on arendatud vanemal versioonil töötavad ka uuematel versioonidel.

2.2.1 Olulised erinevused

Maailmas konkurentsis püsimiseks infotehnoloogia valdkonnas tuleb pidevalt kaasas käia erinevate uuendustega, seetõttu uuendab Android oma tarkvara regulaarselt, tavaliselt iga aastaselt. Lisaks iga-aastastele uuendustele uuendab Android oma versiooni ka suuremate turvavigade ja muude vigade ilmnemisel, mis võivad segada seadme toimimist, tööjõudlust või aku kasutust nende tekkimisel. Sellised uuendused kasutavad Androidi versiooni numbris üldjuhul kolmandat numbrit , nt. 4.0.2, 4.1.1 ja 5.1.1 .

Iga Androidi uus versioon pakub kasutajale uusi võimalusi ning samuti võimaldab arendajal kasutajale pakutavaid võimalusi rakendada, ning parandab arendusvõimalusi ja ligipääsu erinevatele uutele komponentidele, mille riistvara uus Androidi versioon võimaldab. Nagu eelnevalt mainitud ei kasuta kõik Androidi seadmed kõige uuemat versiooni, vaid see oleneb nutiseadme tootjast ning kas tootja on seadme vastavale versioonile modifitseerinud. Alljärgnevas loetelus ei ole mainitud kõiki Androidi versioone, välja on toodud versioonid, mis tõid kaasa suuremad uuendused ja võimalused seadmetele.

Android 1.0
Android 1.0 oli esimene avalik Androidi versioon mis tuli välja aastal 2008 ja mis andis seadmele tänapäeval nutiseadme jaoks elementaarsed võimalused. Versioon andis võimaluse kasutada nutiseadme kaamerat, kuid kaamera seadete muutmise võimalus puudus. Lisaks implementeeriti Wi-Fi ja Bluetoothi kasutusvõimalus, seadmele tekkisid juba algselt olemasolevad rakendused, sealhulgas Android Market(praeguse nimega Play Store), kalkulaator, äratuskell, Google Maps, rakendus piltide vaatamiseks(Photos) ja helistamise ning kontaktide rakendus.

Android 1.5 (Cupcake)

Android 1.5 oli esimene versioon, millele anti koodnimi mõne magusa toote või üldlevinud maiustuse järgi. See versioon võimaldas kasutada kolmandate osapoolte loodud klaviatuure, samuti võimaldas klaviatuur teksti ennustamist, samuti võimaldati

stereo kasutamine Bluetoothi kaudu ja ekraani pööramine, mis võimaldas piltide ja videote vaatamiseks kasutada kogu ekraani.

Android 1.6 Donut

Versioon Donut võimaldas Androidi kasutada ka WVGA ekraani resolutsiooniga seadmetega, mis lubas kasutada Androidi kasutada suuremal hulgal erineva suurusega nutiseadmetel. Samuti võimaldati ühendada erinevaid rakendusi erinevate ekraanil tehtavate liigutustega, näiteks joonistades seadme ekraanile näpuga ringi oli võimalik avada selle liigutusega seotud rakendust. Lisaks tekkis võimalus ekraanil kuvatavat teksti lasta seadmel üldlevinumates keeltes ette lugeda, mis on kasuks eelkõige vaegnägijatele.

Android 2.0 Eclair

Kui tavaliselt on Androidi suuremate versioonide vahe keskmiselt aasta, siis Android 2.0 väljastati vaid poolteist kuud peale eelmist suuremat versiooni uuendust. Antud versioonis suurendati kasutaja konto sünkroniseerimist, mis võimaldas lisada seadmele mitu kasutajakontot erinevate meiliaadresside ja kontaktide kasutamiseks. Suurendati kaamera seadistamis võimalusi, lisati võimalus kasutada välku, kasutada digitaalset suurendamist ja muuta valge tasakaalu(White Balance). Samuti lisati Androidile HTML5 tugi ning samuti Bluetooth 2.1 tugi. Uuendati ka kasutajaliidest, millel muudeti välimust. Lisaks parandati võimekust jälgida mitut ekraani puudutust üheaegselt.

Android 2.2 Froyo

Antud versioon tuli välja 2010 aasta kevadel ja tõi kaasa olulise paranduse seadme jõudluse ja mälu optimiseerimisel, rakenduste kiiruse tõstmiseks implementeeriti ka JIT, mis võimaldab kompileerida programmi koodi programmi käitamise ajal, mitte sellele eelnevalt, mis mõnes olukorras parandab rakenduse võimekust. Samuti lisandus võimalus vahetada otse klaviatuuril klaviatuuri keelt, mis võimaldab näiteks kirjutada teatud osa tekstist teises tähestikus nagu näiteks kirillitsa. Samuti lisandus võimalus kasutada nutiseadet ruuterina, ehk kasutada telefoni Wi-fi leviala loomiseks. Lisaks võimaldati rakenduste salvestamine välisele mälukaardile, enne oli võimalik hoida välisel mälukaardil vaid erinevaid faile. Loodi ka toetus Adobe Flashile veebibrowseris, mis oli tol ajahetkel laialdaselt kasutusel ning mis tekitas vastandlikkust Apple ja Google vahel, nimelt oli Apple kindlalt oma nutiseadmetes Adobe Flashi toetamise vastu.

Android 2.3 Gingerbread

Modifitseeriti kasutajaliidest, mis parandas selle lihtsust, disaini ja ka kiirust. Versioonile lisati NFC toetus, kasutajate jaoks oli oluliseks lisaks võimekus toetada nutiseadmeid mitme kaameraga, mis võimaldas seadmetele lisada ka kaamera nutiseadme ekraanipoolsele osale. Kaameraga lisaks loodi võimalus kasutada lisakaamerat videokõnede tegemiseks . Lisati ka toetus erinevatele riistvara komponentidele nagu , baromeeter ja güroskoop. Seadme aku kasutamise optimeerimiseks loodi võimalus vähendada rakenduste energiakasutust, mis hoidsid seadet ka rakenduse mitte kasutamisel ärkvel ehk aktiivses tööreziimis. Rakenduste arendajatele loodi võimalus implementeerida osa rakendusest programmeerimiskeeltes c ja C++. See andis võimaluse just mängurakenduste loojatele luua rakendusi mis võimaldavad paremini kasutada seadme jõudlust kuna sellisel lähenemisel on võimalik kood kompileerida otse masinkoodi, mitte nagu Java puhul.

Android 3.0-3.2 Honeycomb

Lisati võimekus toetada mitmetuumalisi nutiseadmeid, samuti võimalus krüpteerida kõik kasutaja andmed. Oluliseks turvalisuse parandamiseks keelati rakendustel võimalus kirjutada andmeid välisel mälul väljaspool neile võimaldatud mälu, eelnevalt võis iga rakendus kirjutada ümber andmeid tervel välisel mälul. Võeti kasutusele MTP ja PTP protokollid, mis lihtsustasid failidele ligipääsu seadme ühendamisel arvutiga. Kui siiani oli võimalik kopeerida andmeid vaid rakenduse siseselt, siis nüüd tekkis võimalus teksti kopeerimisele terves süsteemis erinevate rakenduste vahel. Võimaldati ka HTTP reaalajas videopildi ja heli edastamine(HTTP Live Streaming protocol). Lisati ka toetus RTP protokollile. Versiooni täiustati hilisemate uuendustega 3.1 ja 3.2 mis võimaldasid lisada seadmele väliste seadmetena hiirt ja klaviatuuri. Arendajatele loodi võimalus paremini luua rakendusi, mis võimaldas kergemini luua rakendus erinevate suurustega seadmetele, jäädes visuaalselt viisakaks kõikidel ekraanisuurustel.

Android 4.0-4.0.4 Ice Cream Sandwich

Android 4.0 tuli välja 2011 aastal ja lõi kasutajate jaoks võimaluse pääseda ligi osadele rakendustele ilma seadet lahti lukustamata. Selliseks rakenduseks on näiteks kaamera, mis võimaldab vajadusel kiiresti pilti teha, kuid tehtud piltidele ei ole võimalik siiski ilma seadet lahti lukustamata ligi pääseda. Seadme kaameratele võimaldati kasutada

näotuvastustarkvara mis andis võimaluse alternatiivsele seadme lahti lukustamisele. Samuti tekkis võimalus kontrollida mobiilse interneti kasutust, ning vajadusel andmemahtu teatud mahuni jõudes piirata. Lisati võimalusi NFC kasutamiseks kahe seadme vahel failide, veebilinkide ja muu informatsiooni jagamiseks. Moderniseeriti Androidi kaamera rakendust, lisati panoraamvõte ja parandati olemasolevaid funktsioone, võimaldati ka seadmele lisada temperatuuri ja niiskusandureid, ning parandati juba olemasolevate andurite kasutamist.

Android 4.1 -4.3 Jelly Bean

Androidi versioon Jelly Bean koosneb kokku 7 alamversioonist. Versioon võimaldas kasutajale jagada oma seadme pilti Miracast abil juhtmevabalt teistele ekraanidele, kasutajale anti võimalus rakenduste edastatud teateid kas lubada või keelata, võimaldati toetus 4K resolutsioonile, samuti loodi võimalus kirjutada heebrea ja araabia keeltes paremalt vasakule, võimaldati BLE toetus, ka kaamera rakendus sai uuendusi kasutajaliidesele ning lisati võimalus luua 360 kraadiseid fotosi. Arendajatele loodi võimalus turvaliselt vaid autenteeritud füüsilisel seadmel USB ühenduse kaudu programmi vigade silumiseks. Lisati võimalus lisada vaid tahvelarvutitele mitu erinevat kasutajat, kes saavad kasutada vaid nende kasutajaga seotud seadme sisu.

Android 4.4-4.4.4 Kitkat

Android Kitkat võimaldas kasutajale salvestada ekraanile kuvatavat sisu, parandati süsteemi jõudlust ja akukasutust. Rakenduste arendajatele loodi printimise ja mälu kasutamiseks raamistik, mis lihtsustas erinevate võimaluste kasutamist, ning lisati tööriistad mälu kasutamise analüüsimiseks, rakenduse nõrkade kohtade avastamiseks.

Android 5.0 Lollipop

Android 5.0 parandas süsteemi jõudlust ja akukasutust, võimaldati nutiseadmele kahe SIM kaardi toetus, loodi võimalus kadunud või varastatud nutiseadme lukustamiseks, vältimaks isikliku informatsiooni sattumist võõraste inimeste kätte. Loodi suutlikkus kõrglahutusega telefonikõnede tegemiseks. Loodi võimalus NFC ja Bluetoothi abil kogu seadme info edastamine teise telefoni, mis kergendas üleminekut vanalt nutiseadmelt uuele. Pidevaks probleemiks olnud rakenduste suure akukasutuse leevendamiseks loodi

võimalus jälgida spetsiifiliste rakenduste akukasutust, lootes, et arendajad saavad antud informatsiooni põhjal vähendada rakenduste ebavajalikku akukasutust.

Android 6.0 Marshmallow

Android Marshmallow on hetkel kõige vanem Androidi versioon, millel Google endiselt turvaauke lapib. Marshmallow toetab ka USB-C ühendus- ja laadimiskaablit, mis alustas seadmete üleminekut Micro USB ühenduselt. Sellega loodeti minna üle uuele standardile, mis võimaldab suuremat andmeedastus kiirust, ning võimaldab laadida ka sülearvuteid. Kasutajatele anti võimalus määrata, millised rakendused millisele nutiseadme sisule ligi pääsevad, näiteks kas seade võib kasutada kaamerat ja mikrofoni, salvestada ja lugeda rakenduse mälu väliseid faile või ligi pääseda helistamis funktsioonile. Loodi toetus sõrmejälje lugejale, mis võimaldavad seadmeid millel on sõrmejälje lugeja lahti lukustada oma sõrmejäljega.

Android 7.0 Nougat

Nougat tuli välja 2016 aasta sügisel ja tõi endaga kaasa Multi-Window mode ehk võimalus jagada ekraani kahe rakenduse vahel, võimaldades neid üheaegselt kasutada. Suureks uuenduseks oli ka Daydream virtuaalreaalsuse platvorm, mis võimaldas seadmetel, mis omasid platvormile vajalikku riistvara ja tarkvara mis andis võimaluse antud seadmeid kasutada Daydream View seadmes ekraanina. 2018 Mai seisuga toetab platvormi vaid üksteist 7 erineva tootja nutiseadet. Lisati ka hoiatused, mis annavad kasutajale teada rakendustest, mis kasutavad palju akut.

Android 8.0-8.1 Oreo

Android Oreo tuli välja 2017 augustis ning 8.1 on hetkel kõige uuem saadaolev versioon Androidile. Versioon tõi endaga kaasa võimaluse näha Bluetoothi kaudu ühendatud seadmete akutaset, muuta kahe rakenduse üheaegsel kasutamisel akende suurust, ning loodi Go Edition, mis võimaldab kasutada Androidi seadmetel millel on vähem kui 1GB muutmälu.

Android Pie

Android Pie on Androidi kõige uuem versioon mille kõige olulisemad uuendused on seotud seadme kasutajale mugavdamisega, ehk lihtsustati võimalusi rakenduste

vahetamiseks, samuti parandati seadme akukasutust. Uus versioon võimaldab seadet ühendada kokku kuni 5 seadmega läbi Bluetooth ühenduse, mis võimaldab kasutajal lihtsalt lülituda ühelt Bluetooth seadmelt teisele. Lisaks on võimalik uue versiooniga ühendada USB kaudu väline kaamera juhul kui seade seda võimaldab. [4]

2.2.2 Järjepidevus versioonides

Kuigi erinevates versioonides on ajajooksul esinenud nii kasutajaliidese, turvalisuse kui ka jõudluse erinevusi, on uuemad versioonid loodud selliselt, et rakendused, mis on loodud vanemale versioonile, töötavad tõrgeteta ka kõige uuematel versioonidel, kuid need rakendused kasutavad tavaliselt liigselt ressurssi kui kaasajastatud rakendused.[5]

2.3 Nutirakendused

2.3.1 Kasutamisvaldkonnad

Nutiseadme rakendusi on võimalik kasutada kõikides elu valdkondades, Android operatsioonisüsteemile loodud rakenduste arv ulatub umbes 3,5 miljoni rakenduseni 2017 aasta detsembri seisuga ja see arv kasvab igapäevaselt. [6]

Kõige populaarsem valdkond nutirakendustel on erinevad mängud, need võivad olla nii mitme mängijaga veebipõhised mängud kui ka vaid ühele mängijale mõeldud rakendused, mis ei vaja kasutamiseks interneti ühendust.

Suur hulk olemasolevatest rakendustest on veebilehe põhised rakendused, mis muudavad kasutajale sisu vaatamise mugavamaks väiksemal ekraanil. Sellisteks rakendusteks on näiteks erinevad veebipoed ja uudisteportaalid.

Suur kasutajaskond on ka sotsiaalmeedia rakendustel, näiteks Facebook, Twitter ja Instagram, nende rakenduste abil on kasutajal võimalik olla ühenduses oma sõpradega, nendega suhelda, jagada pilte, arvamusi ja muu info kohta, mida peetakse oluliseks teiste inimestega jagada.

3 Rakenduse loomine Android operatsioonisüsteemile

Rakendust Android operatsioonisüsteemile on võimalik luua selleks loodud Android Studio tarkvaras. Enne rakenduse loomist on vajalik teada, milliseid lahendusi ning riistvarakomponente loodav rakendus nõuab, ning kui oluline on rakenduse loomisel kasutajate hulk, näiteks luues sotsiaalmeedia rakendusi, mis võimaldavad kasutajatel üksteisega suhelda või informatsiooni jagada, on edu eelduseks võimalikult lai nutiseadmete valik, et lihtsustada uute kasutajate lisandumist. Sellisel juhul peab soovitud rakendust arendama vanemale versioonile, mille abil saaks antud rakendust kasutada potentsiaalselt maksimaalne hulk nutiseadmete omanikke(tabel 3). Kui aga soovitakse kasutada teatud riistvara või lahendusi, tuleb uurida, alates millisest versioonist on vastavad lahendused võimaldatud.

Esimese sammuna tuleb alla laadida endale android Studio, tarkvara on võimalik installida nii Windows, Mac kui ka Linuxi operatsioonisüsteemile. Tarkvara avamisel tuleb luua uus projekt, kirjutada loodava rakenduse esialgne nimi, samuti tuleb otsustada, kas loodav rakendus arendatakse ainult Java programmeerimiskeeles, või soovitakse lisada ka C++ ning Kotlin programmeerimiskeele toetus. Kotlin on avatud lähtekoodiga programmeerimiskeel, mis baseerub Java virtuaalmasinal.

Seejärel tuleb otsustada, millise API(Application Program Interface), ehk versiooni arendamisel kasutatakse. Peale valiku tegemist pakutakse arendajale valida, millise struktuuriga saab olema loodava rakenduse peamine toimingu leht, selleks on võimalik valida erinevate mallide vahel või valida tühi mall. Malle on võimalik hilisemalt vastavalt oma soovile modifitseerida, lisaks on hilisemalt võimalik toimingu lehekülgi lisada, kui soovitakse näiteks lisada toiminguid, mida kasutaja teeb ühekordselt või harva, ei peaks need olema koheselt kättesaadavad. Peale esmaste valikute tegemist, genereeritakse loodavale projektile algsed failid.

Failide genereerimisel luuakse projekti kaust, kus peamiseks failis on Android manifest, mis on xml fail, kus on kirjeldatud rakenduse nimi ja peamised rakenduse seadistused. Juhul kui loodav rakendus nõuab ligipääsu teatud parameetritele nagu internet või telefoni kaamera, tuleb kõik vajalikud parameetrid antud failis kirjeldada. Algses projekti failis on ka olemas ka tühi Java klass, mis on seotud algselt loodud toimingu lehega, kuhu on

võimalik luua antud toimingu lehel toimuvate funktsioonide rakendamiseks koodi. Teine fail mis on seotud peamise toimingu lehega on antud toimingu lehekülge kirjeldav xml, kus on võimalik modifitseerida rakenduse lehekülje välimust. Seda on võimalik teha manuaalselt parameetreid kirjutades ehk tekstipõhiselt, kuid olemas on ka kasutajale tihti mugavam visuaalne võimalus, kus on võimalik lisada atribuute lehekülje visuaalsele kujundusele, mis annab parema ülevaate lehe välimusest. Kuigi on võimalik lisada teksti erinevate nuppudele ja väljadele selles samas xml failis, on mõistlikum lisada tekstid eraldi automaatselt genereeritud strings.xml faili ning antud parameeter edastada selle id abil. Selline lähenemine võimaldab hoida loodavat rakendust paremini struktureeritud, ning vajadusel kasutada ühte teksti väärtust mitmekordselt, ei teki vajadus seda kirjutada mitmekordselt. Samuti soovitatakse antud lähenemist, sest kui hiljem soovitakse antud rakendus teha võimalikuks mitmes erinevas keeles, on kogu rakenduses sisalduv tekst ühes failis ning seda on võimalik kergemini tõlkida, ning erinevate keelte kasutamisel rakendada.

3.1 Nõuded riistvarale

Android operatsioonisüsteem töötab põhiliselt ARM, ehk reduced instruction set computer arhitektuuriga protsessoritel. Uuemate Androidi versioonide mugavaks kasutamiseks peaks seadmel olema vähemalt 2GB muutmälu. Android Oreo-ga tuli välja ka Android Go mis on spetsiaalselt mõeldud seadmetele, millel on 512MB-1GB muutmälu. Android Go on niiöelda kergekaaluline Androidi operatsioonisüsteem, mis võtab vähem seadme põhimälu, et töötada nõrgemate riistvarakomponentidega seadmetel. Lisaks operatsioonisüsteemile, on ka antud süsteemile loodud spetsiaalsed Google rakendused, mis vajavad töötamiseks vähem jõudlust. Kuigi antud versioon on kiirem ja lihtsam, on selle saavutamiseks vähendatud teisi võimalusi, et jõudlust parandada. Näitena võib tuua rakenduse Facebook Messenger, millest on olemas nii tavaversioon kui ka Go edition, kui tavalisele Androidile arendatud rakenduses on võimalik saata nii pilte, teha videokõnesi kui ka jagada faile, on selle rakenduse GO editionis võimalused nagu kaamera kasutus ja failide ning piltide jagamine ning videokõned eemaldatud, et hoida rakendust võimalikult lihtsana ja vähem seadme jõudlust vajavaks.

Androidi operatsioonisüsteem oli algselt loodud ainult nutitelefonidele, ning seetõttu oli esimestes versioonides nõue, et olemas oleksid riistvara komponendid kõnede tegemiseks nagu näiteks mikrofon. Aja möödudes selliseid nõudeid riistvara komponentidele leevendati, sest Android levis ka tahvelarvutitele ja muudele nutiseadmetele mille funktsioonidele polnud ettenähtud vajadus helistamiseks.

Lisanduvad riistvaralised nõuded sõltuvad erinevatest rakendustest, mis neid komponente kasutada soovivad. Seetõttu ei pea seadmetel olema näiteks GPS toetust, küll aga ei saa sellisel juhul kasutada navigeerimis rakendusi mis vajavad selleks piisavalt täpset positsioneerimis võimekust.

3.2 Nõuded tarkvarale

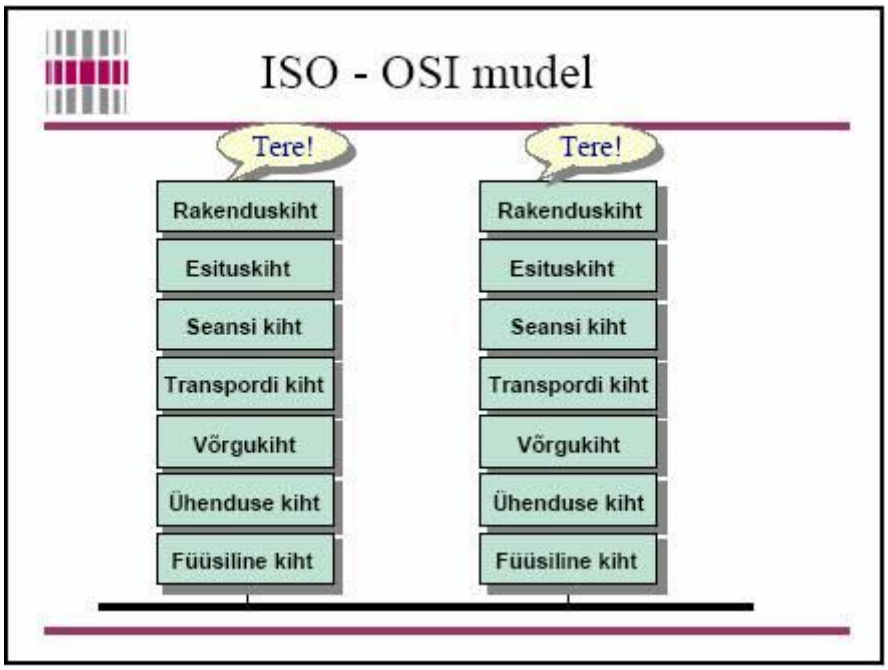
Nutirakenduse arendamiseks on vaja arenduskeskkonda milleks võib olla nii autori poolt kasutatud Android Studio, kuid rakendusi on võimalik arendada ka Visual Studio, IntelliJ IDEA ja mitmetes teistes arenduskeskkondades. Rakenduse testimiseks on vajalik Android nutitelefon või Android Emulator mis võimaldab lihtsasti emuleerida suurt hulka erinevaid nutitelefone ilma neid reaalselt omamata. Android Emulator lihtsustab rakenduse testimist erinevatel telefonidel millel on erisuurusega ekraan ning samuti saab selle abil kontrollida kuidas töötab rakendus ka vanemate Android versioonidega.

4 Nutirakendus videoga

Antud töö on edasiarendus Teede Tehnokeskuse projektist, mille eesmärgiks oli maapinnal asuvate objektide positsioneerimine videol. Selle projekti puhul video salvestati, et seda oleks võimalik hiljem analüüsida ning töödelda. Projektiga tegeles autor koos kaastudengi Marko Ratasega, kes antud teemal ka oma lõputöö kirjutas. Projekti käigus tekkis testimisel vajadus näha kaamerapilti koheselt, et saada ülevaade kaamera positsioonist ning nurgast teepinna suhtes. Sealt arenes välja idee, võimekuse loomiseks, millega saaks erinevatest kaameratest saadavat pilti otse nutitelefoni ekraanilt vaadata, mis on tänapäeval kõigil juba olemas kuid mida võiks edukalt kasutada ka muude seadmete ekraanina.

Nutirakenduse, mis võimaldab video kasutajale edastamist loomine on Androidis tehtud kergesti implementeeritavaks ning arendaja jaoks on erinevaid võimalusi, samuti on olemas juba mitmeid valmis rakendusi mida on võimalik kasutada. Sellisteks rakendusteks on näiteks VLC for Android, VLC player on vabavara, mis võimaldab esitada enamtuntud video ja audio formaate, samuti on VLC player olemas kõikidele tuntud operatsioonisüsteemidele.[7]

4.1 RTSP Protokoll



Joonis 1. Arvutivõrgu ISO OSI mudeli füüsiline ja ühenduskihid.[8]

RTSP(Real Time Streaming Protocol) on rakendustaseme kontrollprotokoll reaalaja andmete edastuse võrgus juhtimiseks. RTSP töötab ISO- OSI mudelil rakenduskihis ning kasutab transpordi kihis UDP(User Datagram Protocol) ja TCP(Transmission Control Protocol) ühendust.

4.1.1 Real Time Transport Protocol

Reaalaja transpordi protokoll(RTP) on võrgu protokoll audio ja video edastamiseks võrgus. Edastus toimub tavaliselt üle UDP protokolli. Protokolli eesmärgiks on reaalajas heli ja video edastamiseks rohkem kui ühele kliendile. RTP protokoll võimaldab ka avastada pakettide kaduma mineku mida võib sageli ette tulla UDP kasutamisel. Kuigi RTP võimaldab ka TCP kasutamist, eelistatakse siiski UDP kasutamist just tema kiiruse tõttu andmete edastamisel, mis reaalajas meedia edastamisel on tavaliselt olulisem, kui kõikide pakettide kohalejõudmises kindel olemine.

4.1.2 Real Time Control Protocol

RTCP protokoll võimaldab analüüsida edastuse kvaliteeti. Protokoll on seotud RTP protokolliga ja iga klient kes on edastusega ühendunud, saadab automaatselt edastuse kohta tagasisidet.[9]

4.1.3 User Datagram Protocol

Protokoll kasutab UDP transpordiprotokolli pideva ja kiire ühenduse tagamiseks. UDP protokollis ei toimu ühenduse kontrolli, mis tähendab, et kõik paketid ei pea vastuvõtjani jõudma, see lähenemine tagab kiirema pakettide edastamise, mis on video otseedastusel olulisem kui on seda kõikide pakettide garanteeritud kohale jõudmine.

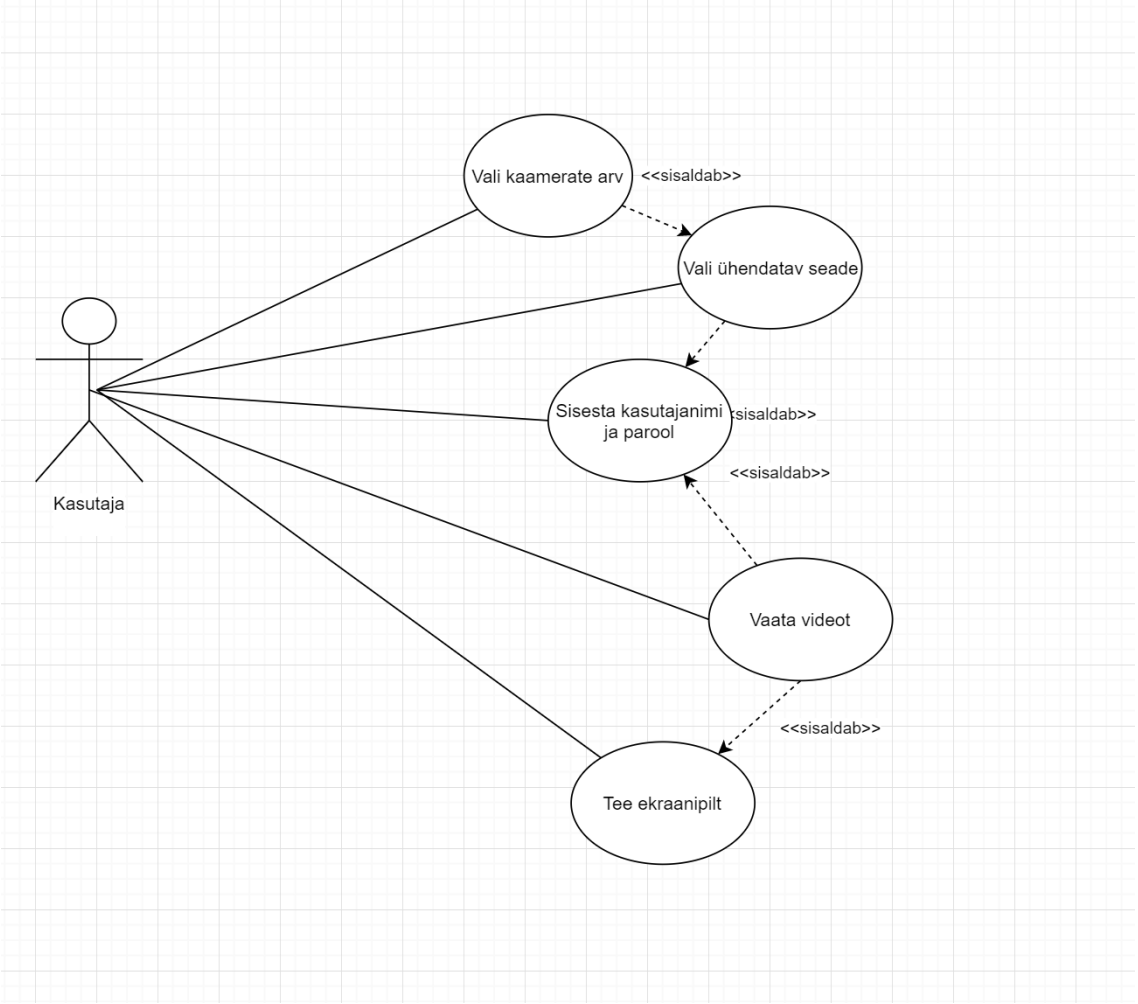
4.1.4 Transmission Control Protocol

TCP protokoll on transpordi kihi protokoll mis erineb UDP protokollist kontrollmehhanismide tõttu. TCP protokollis toimub andmevahetuse kontroll juba protokolli kihis.

4.2 Kaamera konfiguratsioon ja nõuded

Kaamera konfiguratsiooni puhul on oluliseks faktoriks edastatava pildi formaat ja protokollid mida kaamera võimaldab.

4.3 Nutirakendus



Joonis 2. Rakenduse kasutajaloo diagramm

Antud rakendus on loodud versioonile API 27, ehk Android Oreo 8.1. Kuigi rakenduse kõige olulisemad komponendid, sealhulgas video esitamise võimalus ning video edastus RTSP protokolli abil on toetatud juba alates Android Honeycomb 3.0 versioonist, kasutas autor rakenduse loomisel hilisemat versiooni, mille kasutajaskond on väike, kuid selle lähenemise tõttu oli autoril võimalik olla kindel, et töö käigus ei esine probleeme erinevate võimaluste kasutamisel, teades, et antud versioonis on kõik hetkel olemasolevad lahendused kasutatavad ning implementeeritud.

Samuti võib rakenduse loomisega vanemale versioonile kui Marshmallow tekkida konflikt rakenduse kasutajatega. Rakendused mis on vanemad kui Marshmallow nõuavad automaatselt ligipääsu kogu nutiseadmele, sealhulgas mälule, kaamerale, mikrofonile,

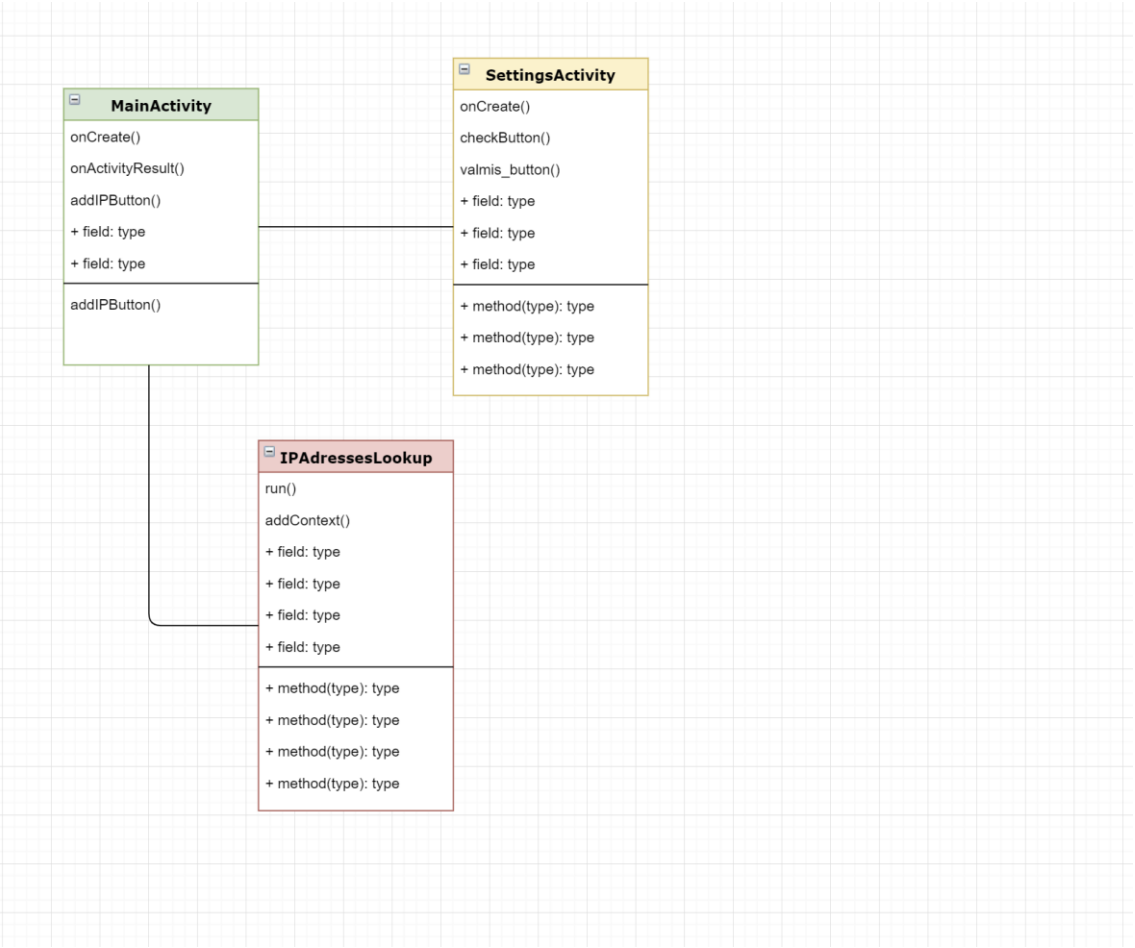
kontaktidele, ja muule privaatsele sisule. See võib tekitada olukorra, kus kasutaja ei ole nõus antud rakendust kasutama, sest kardab, et nende isikliku informatsiooni võidakse ära kasutada kolmandate osapoolte huvides. Testimise käigus avastasin, et antud rakendust on võimalik kasutada ka API 23 ehk seadmetel, mille operatsioonisüsteem on vähemalt Android Marshmallow.

Rakenduse peamine eesmärk on luua autosse võimalus kasutada pardakaamera asemel lahendust, mis võimaldab kasutada tavalist traadita ühendusega kaamerat ning ekraanina Android nutitelefoni, mis on tänapäeval üldlevinud ning suure kasutajaskonnaga. See ei ole rakenduse ainuke võimalik rakendusmeetod, lahendus on mõeldud üldakasutatavana ka muudel eesmärkidel kus selline lahendus võib praktilist otstarvet leida. Rakendus võimaldab pilti vaadelda reaalajas, samuti on võimalik rakendusele implementeerida salvestusvõimalus, mis aitab fikseerida näiteks liiklusõnnetusi ja pärast kasutada antud videot tõestusmaterjalina. Kui kaamera on paigaldatud auto tahaossa, saab kasutada pildi reaalajas edastamise võimalust kasutada ka vaatevälja suurendamiseks auto tagaosas pimedate nurkade vältimiseks.

Rakendus omab kahte toimingu lehekülge, Main Activity võimaldab vaadata antud videot ning settings võimaldab kasutajal valida, kas soovitakse pilti edastada ühelt või kahelt kaameralt, ning sisestada kaamera või kaamerate URL. Peale valikut satub kasutaja automaatselt tagasi eelnevale leheküljele, kus vastavalt kasutaja valikule on ekraanil üks või 2 videomängjiat võimaldamaks esitada videot.

Rakenduse testimiseks kasutasin Android Emulatorit, mis imiteeris nutiseadet Pixel XL, mis on arendatud Google poolt ning mille tootjaks on HTC. Emulaator kasutas töötamiseks Android Oreo 8.1 versiooni. Emulaatori kasutamine võimaldas rakenduse koodi kergemini siluda ning polnud vajalik antud rakendust reaalses füüsilises seadmes katsetada, mis vähendas ajakulu.

4.3.1 Nutirakenduse struktuur



Joonis 3. Nutirakenduse üldstruktuur

Nutirakendus koosneb kolmest põhifailist : MainActivity.java, SettingsActivity.java ja IPAdressesLookup.java. Lisaks on rakendusel ka rakendusel ka kaks XML faili activity_settings.xml ja activity_main.xml mis kirjeldavad rakenduse toimingulehtede visuaalse kujunduse. Igal Android rakendusel peab olema ka AndroidManifest.xml mis kirjeldab üldist rakenduse struktuuri ja rakenduse poolt nõutavad ligipääsud seadmele ja kasutajaandmetele(ligipääs internetile, kaamerale, kontaktidele jne.). Kogu rakenduses kuvatav tekst, näiteks nuppudel kuvatav tekst deklareeritakse eraldi failis strings.xml. Selline lähenemine võimaldab rakenduses kuvatavat teksti hiljem kergemini muuta, ning samuti annab see võimaluse vastavalt vajadusele hilisemalt tõlkida rakendus ka teistesse keeltesse kui kasutajaskond peaks suurenema ja laienema.

MainActivity.java

MainActivity.java sisaldab video esitamiseks vajalikku koodi ning kutsub esile järgnevad vaated ning funktsioonid.

onCreate :

Rakenduse avamisel genereerib rakenduse avalehe ning käivitab kohe rakenduse avamisel ka hargtöötlust kasutava IPAddressLookup-i, mis paralleelset tööd alustab.

addIPButton :

Meetod seotud rakenduse pealehel oleva nupuga „Seaded“ ning käivitub nupule vajutamisel. meetod kutsub esile rakenduse teise toimingu lehe „SettingsActivity, annab sellele kaasa hargprogrammist saadud IP- aadressid.

onActivityResult :

Alustab tööd juhul kui toimingu lehekülg SettingsActivity on oma töö lõpetanud. meetod kontrollib kaasa antud video aadressi ning juhul kui ühendus on edukas, edastab video.

Joonis 4. Pilt rakendusest

SettingsActivity.java

SettingsActivity peamiseks funktsiooniks on kasutaja sisestatud aadressi sisselugemine ja selle edastamine MainActivity toimingulehele video edastamiseks.

onCreate :

Genereerib toimingu lehekülje visuaalse disaini

checkButton:

loeb kasutaja sisestatud valiku raadio nuppude valikust, kas üks või kaks kaamerat ning vastavalt sellele muudab antud arvu tekstisisestuseks mõeldud lünki kasutaja jaoks nähtavaks.

valmis_button:

loeb sisestatud kasutajanime aadressi ja parooli ning pakib need kokku edastamiseks MainActivity toimingulehele ning sulgeb SettingsActivity toimingu.

IPAdressesLookup:

Klassi eesmärgiks on leida ühendatud võrgust millega seade on ühendatud ülejäänud seadmed, et kasutajal oleks lihtsam ühendada soovitud kaameraga.

run:

kontrollib läbi kõik kohaliku võrgu aadressid, et leida kaamera. Antud ülesanne on ajamahukas ja et mitte luua situatsiooni kus kasutaja peab rakenduse järgi ootama, töötab see osa programmist paralleelselt.

getIPaddresses:

võimaldab tagastada leitud seadmete aadressi rakenduse põhiprotsessile, juhul kui kõrvalprotsess pole selleks ajaks oma tööd lõpetanud, antakse tagasi vaid need seadmete aadressid mis on selleks hetkeks leitud.

4.4 Edasiarengu võimalused

Rakenduse edasiarendamisel oleks järgmisteks lisavõimalusteks võimalik lisada vaadatava video salvestamine ja ka ekraanipildi tegemine samal ajahetkel esitatud videost. Video salvestamisel saab lisada ka võimeluise salvestatud video uuesti esitamiseks.

Rakenduse loomisel ja katsetamisel erinevate kaameratega leidis töö autor, et kasutatud teek VideoView võimaldab esitada vaid teatud videoformaati edastavate kaamerate videot. Kui uuemad kaamerad võimaldavad muuta edastatava video formaati, siis vanemad kaamerad võivad kasutada edastamiseks antud rakendusele mittesobivaid videoformaate. Selle probleemi lahendamiseks võib vahetada hetkel kasutusel oleva videomängija mõne vabavarana kasutatava videomängijaga või luua ise võimekus esitada hetkel mittesobivaid videoformaate neid ümber kodeerides.

5 Kokkuvõte

Antud töö teemaks oli nutiseadme kasutusvõimalused sõidukaamera näitel. Töös tutvustati nutiseadmete arengut keskendudes Android operatsioonisüsteemiga seadmetele ja selgitati nutiseadmete kasutamisvõimalusi erinevate ülesannete täitmisel. Nutirakenduste paremaks selgitamiseks tutvustati töös autori poolt loodud rakendust kaamerapildi vaatamiseks ja rakenduse loomise protsessi loodud rakenduse näitel.

Töö tutvustab autori arvates põhjalikult Android süsteemi kasutavate seadmete võimalusi ning annab põhjaliku ülevaate rakenduse loomise protsessist ja võimalusest kasutada nutitelefoni ka väliste seadmetega ühendamiseks ja nende edastatava info kasutamise võimaldamise.

Töö käigus tutvus autor programmeerimiskeelega Java, nutirakenduste loomisega Android süsteemile ja reaalajas pildi vaatamiseks vajalike protokollidega mida erinevad kaamerad on võimelised esitama.

6 Kasutatud kirjandus

[1] https://www.computerweekly.com/feature/Apollo-11-The-computers-that-put-man-on-the-moon

[2] https://www.oneplus.com/ee/5/specs

[3] http://socialcompare.com/en/comparison/android-versions-comparison

[4] https://www.android.com/versions/pie-9-0/

[5] https://developer.android.com/about/dashboards/

[6] https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[7] https://developer.android.com/studio/intro

[8] http://veeremaa.tpt.edu.ee/net/Arvutivorgud.htm

[9] https://tools.ietf.org/html/rfc3550

Lisad

MainActivity.java

```
package cameraviewer_rain_kilkson.cameraviewer;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;

import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.view.View;
import android.widget.VideoView;


import java.lang.ref.WeakReference;
import java.util.ArrayList;
import java.util.Date;

import static android.text.format.Formatter.formatIpAddress;


public class MainActivity extends Activity {

    private IPAddressesLookup getIpaddresses = new IPAddressesLookup();


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Context context= getApplicationContext();
        setContentView(R.layout.activity_main);
        getIpaddresses.addContext(context);
        getIpaddresses.start();
    }

    @Override
    public void onActivityResult(int requestCode,
                                 int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        Context context= getApplicationContext();
        Bundle bundle = data.getExtras();
        if(data.hasExtra("ip_address2_data")&& getIntent().hasExtra("ip_ad-
dress1_data")){
            String value2 = bundle.getString("ip_address2_data");
            Uri videoUri2 = Uri.parse(value2);
            VideoView videoView2 = (VideoView) findViewById(R.id.sec-
ond_video);
            videoView2.setVisibility(View.VISIBLE);
            videoView2.setVideoURI(videoUri2);
            videoView2.requestFocus();
            videoView2.start();
            String value = bundle.getString("ip_address1_data");
            video1(value);
        }else if(data.hasExtra("ip_address1_data")){
            String value = bundle.getString("ip_address1_data");
            video1(value);
        }
    }
```

```
    public void addIPButton(View view) {
        Intent intent = new Intent(this, SettingsActivity.class);
        ArrayList  iPaddressesList =   getIpaddresses.getIPaddressesList();
        intent.putStringArrayListExtra("ipAdressesFound",iPaddressesList);
        startActivityForResult(intent,1);
    }

    public void video1(String value) {
        VideoView videoView1 = (VideoView) findViewById(R.id.first_video);
        Uri videoUri = Uri.parse(value);
        videoView1.setVideoURI(videoUri);
        videoView1.requestFocus();
        videoView1.start();
    }
}
```

SettingsActivity.java

```
package cameraviewer_rain_kilkson.cameraviewer;

import android.content.Intent;
import android.graphics.Color;
import android.provider.SyncStateContract;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioGroup;

import java.util.ArrayList;

public class SettingsActivity extends AppCompatActivity {

    RadioGroup radioGroup;
    Button btnDisplay;
    AutoCompleteTextView autoCompleteTextView;
    AutoCompleteTextView autoCompleteTextView2;
    ArrayAdapter<String> adapter;
    ArrayAdapter<String> adapter2;
    EditText userName;
    EditText userName2;
    EditText userPassword;
    EditText userPassword2;
    private static final String TAG = SyncStateContract.Constants.DATA;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings);
        autoCompleteTextView = (AutoCompleteTextView) findViewById(R.id.auto-
CompleteIpAdressTextView);
        autoCompleteTextView2 = (AutoCompleteTextView)  findViewById(R.id.au-
toCompleteIpAdressTextView2);
    }

        protected void checkButton(View view) {
            Bundle bundle = getIntent().getExtras();

            radioGroup = (RadioGroup) findViewById(R.id.RadioIPGroup);
            btnDisplay = (Button) findViewById(R.id.btnDisplay);
            userName = (EditText) findViewById(R.id.userName);
            userName2 = (EditText) findViewById(R.id.userName2);
```

```
            userPassword = (EditText) findViewById(R.id.userPassword);
            userPassword2 = (EditText) findViewById(R.id.userPassword2);
            int selectedId = radioGroup.getCheckedRadioButtonId();
            // find the radiobutton by returned id
            ArrayList <String> ipAdressesFoundList = new ArrayList<>();
            if(getIntent().hasExtra("ipAdressesFound")) {
                ipAdressesFoundList = bundle.getStringArrayList("ipAdress-
esFound");
            }
            if (selectedId == R.id.RadioButton) {
                userName.setVisibility(View.VISIBLE);
                userPassword.setVisibility(View.VISIBLE);
                adapter = new ArrayAdapter<String>(this,
                        android.R.layout.simple_dropdown_item_1line, ipAdress-
esFoundList);
                autoCompleteTextView.setAdapter(adapter);
                autoCompleteTextView.setVisibility(View.VISIBLE);

                userName2.setVisibility(View.INVISIBLE);
                userPassword2.setVisibility(View.INVISIBLE);
                autoCompleteTextView2.setVisibility(View.INVISIBLE);

            } else if (selectedId == R.id.RadioButton2) {
                autoCompleteTextView.setVisibility(View.VISIBLE);
                userName.setVisibility(View.VISIBLE);
                userPassword.setVisibility(View.VISIBLE);
                autoCompleteTextView.setVisibility(View.VISIBLE);
                adapter = new ArrayAdapter<>(this,
                        android.R.layout.simple_dropdown_item_1line, ipAdress-
esFoundList);
                autoCompleteTextView.setAdapter(adapter);



                userName2.setVisibility(View.VISIBLE);
                userPassword2.setVisibility(View.VISIBLE);
                autoCompleteTextView2.setVisibility(View.VISIBLE);
                adapter2 = new ArrayAdapter<String>(this,
                        android.R.layout.simple_dropdown_item_1line, ipAdress-
esFoundList);
                autoCompleteTextView2.setAdapter(adapter2);
            }

        }


    protected void valmis_button(View view){//if host is given controls the
host and leads to connecting client with the host
        int selectedRadioButton = radioGroup.getCheckedRadioButtonId();
        String completeIpAddress ="";
        String completeIpAddress2 ="";
        if(radioGroup.getCheckedRadioButtonId() == R.id.RadioButton ||  radi-
oGroup.getCheckedRadioButtonId() == R.id.RadioButton2){
            if(autoCompleteTextView.getText().toString().isEmpty() ||
userName.getText().toString().isEmpty() || userPass-
word.getText().toString().isEmpty()){
                if(autoCompleteTextView.getText().toString().isEmpty()){
                    setInputMissing(autoCompleteTextView);
                }
                if(userName.getText().toString().isEmpty()){
                    setInputMissing(userName);
                }
                if(userPassword.getText().toString().isEmpty()){
                    setInputMissing(userPassword);
                }
```

```
                return;
            }
            String iPString = correctIPAdress(autoComplete-
TextView.getText().toString(),autoCompleteTextView);
            //"rtsp://admin:Suurvend1234@192.168.0.23:554/h264");
            completeIpAddress = "rtsp://" + userName.getText().toString() +
":" + userPassword.getText().toString() +"@" + iPString + ":554/h264";
            Log.i(TAG,"Host address: " + completeIpAddress);
        }if(radioGroup.getCheckedRadioButtonId() == R.id.RadioButton2) {
            if (autoCompleteTextView2.getText().toString().isEmpty() ||
userName2.getText().toString().isEmpty() || userPass-
word2.getText().toString().isEmpty()) {
                if(autoCompleteTextView2.getText().toString().isEmpty()){
                    setInputMissing(autoCompleteTextView2);
                }
                if(userName2.getText().toString().isEmpty()){
                    setInputMissing(userName2);
                }
                if(userPassword2.getText().toString().isEmpty()){
                    setInputMissing(userPassword2);
                }
                return;
            }
            completeIpAddress2 = "rtsp://" + userName2.getText().toString() +
":" + userPassword2.getText().toString() +autoComplete-
TextView2.getText().toString()+ ":554/h264";
            Log.i(TAG,"Host address: " + completeIpAddress2);
        }
        Intent intent = new Intent(this, MainActivity.class);
        Boolean ipAddressValid = false;
        intent.putExtra("ip_address1_data", completeIpAddress);

        if(!completeIpAddress2.isEmpty()) {
            intent.putExtra("ip_address2_data", completeIpAddress2);
        }
         setResult(0,intent);
         finish();
    }

    private void setInputMissing(EditText emptyText) {
        emptyText.setHint("Sisestus puudub");
        emptyText.setHintTextColor(Color.RED);
    }

    private String correctIPAdress(String toString, AutoCompleteTextView auto-
CompleteTextView) {
          String nonCorrectedIPAddress = autoComplete-
TextView.getText().toString();
            String correctedIPAdress = nonCorrectedIPAddress.substring(0,non-
CorrectedIPAddress.indexOf(" "));
        return correctedIPAdress;
    }
}
```

IPAddressesLookup

```
package cameraviewer_rain_kilkson.cameraviewer;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.provider.SyncStateContract;
import android.support.annotation.NonNull;
```

```
import android.util.Log;

import java.io.IOException;
import java.net.ConnectException;
import java.net.InetAddress;
import java.net.InetSocketAddress;
import java.net.NetworkInterface;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Date;
import java.util.Iterator;
import java.util.List;
import java.util.ListIterator;

import static android.text.format.Formatter.formatIpAddress;

class IPAddressesLookup extends Thread {
    private Context context;


    public ArrayList<String> ipaddresses = new ArrayList<String>();
    private static final String TAG = SyncStateContract.Constants.DATA;

    public void addContext(Context contextIncoming) {
        context = contextIncoming;
    }


    public void run(){
        try {
            if (context != null) {
                ConnectivityManager cm = (ConnectivityManager)context.get-
SystemService(Context.CONNECTIVITY_SERVICE);
                NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
                WifiManager wm = (WifiManager)context.getSystemService(Con-
text.WIFI_SERVICE);

                WifiInfo connectionInfo = wm.getConnectionInfo();
                int ipAddress = connectionInfo.getIpAddress();
                String ipString = formatIpAddress(ipAddress);
                InetAddress ipString2 = InetAddress.getLocalHost();

                Log.d(TAG, "activeNetwork: " + String.valueOf(activeNetwork));
                Log.d(TAG, "ipString: " + String.valueOf(ipString));

                String prefix = ipString.substring(0, ip-
String.lastIndexOf(".") + 1);
                Log.d(TAG, "prefix: " + prefix);
                for (int i =0 ; i < 255; i++) {
                    String testIp = prefix + String.valueOf(i);
                    InetAddress address = InetAddress.getByName(testIp);
                    boolean reachable = address.isReachable(10);
                    String hostName = address.getCanonicalHostName();
                    if (reachable && isRTSPPortOpen(testIp,554)){
                        Log.i(TAG, "Host: " + String.valueOf(hostName) + "(" +
String.valueOf(testIp) + ") is reachable!");
                        ipaddresses.add(testIp + " (" + hostName + ")");
                    }else if (!hostName.equals(testIp)){
                        ipaddresses.add(testIp + " (" + hostName + ")");
                    }
                }
            }
        } catch (Throwable t) {
        }
    }
```

```
    public ArrayList getIPaddressesList() {
        return ipaddresses;
    }

    private static boolean isRTSPPortOpen(final String ip, final int port) {
        try {
            Socket socket = new Socket();
            socket.connect(new InetSocketAddress(ip, port), 100);
            socket.close();
            return true;
        }catch(ConnectException ce){
            return false;
        }catch (Exception ex) {
            return false;
        }
    }
}
```