DTALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Artur Lusmägi 155550IAPB

TRAADITA SENSORVÕRGUST KOGUTUD ANDMETE JA NENDE HILISTUMISTE VISUALISEERIMINE

Bakalaureusetöö

Juhendaja: Jaanus Kaugerand
MSc

Tallinn 2018

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: Artur Lusmägi

17.05.2020

Annotatsioon

Käesolevas töö põhieesmärgiks on luua rakendus, mis kuvab traadita sensorvõrgu helisensorite saadetud andmeid ning visualiseerib andmete koguteekonna käigus tekkinud hilistumist. Muuhulgas antakse ülevaade sensorandmete visualiseerimise rakenduse loomisest, loodud rakenduse funktsionaalsestest ja mittefunktsionaalsetest nõuetest ning tutvustatakse valitud lahendusi. Rakenduse loomiseks kasutati Angular ja Node.js raamistikku.

Töö tulemusena valmis rakendus, mis võimaldab jälgida reaalajas helisensorite saadetud andmeid ning tekkivad hilistumist.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 21 leheküljel, 5 peatükki, 9 joonist.

Abstract

Visualizing Data and Delay from a Wireless Sensor Network.

The main aim of this thesis is to create an application, which displays data sent by wireless sensor networks and also visualizes the end-to-end delay of data created by the sensor network with the requirement, that the data is visualized in real-time.

The thesis begins by examining different commercial applications, which are capable of real-time visualization of data from IoT devices. By analyzing these products, the author concluded that they don't satisfy the requirements needed for the application. After that the thesis describes the functional and non functional requirements, which apply to the resulting application. The author then proceeds to give an overview of wireless sensor networks in general, but also describes different wireless sensor networks by their topologies and the delays which occur.

The second part of the thesis is focused on the application itself. It describes the technologies which were used, the format of input data and gives a detailed explanation as to how the program works. It concludes by describing different directions of how the application could be further developed.

The result was an application, which allows the user to observe the data sent by accoustic array sensors in real time and displays the computed end-to-end delays.

The thesis is in Estonian and contains 21 pages of text, 5 chapters, 9 figures.

Lühendite ja mõistete sõnastik

REST	*Representational State Transfer,* tarkvaraarhitektuuri laad veebirakendustes
API	*Application Programming Interface,* rakendusliides
TCP	*Transmission Control Protocol,* edastusohje protokoll
IP	*Internet Protocol,* internetiprotokoll
IoT	*Internet of Things,* asjade internet
WSN	*Wireless Sensor Network,* traadita sensorvõrk
CSS	*Cascading Style Sheets,* kaskaadlaadistik
UTM	*Universal Transverse Mercator,* Mercatori universaalne põikprojektsioon
JSON	*JavaScript Object Notation,* lihtsustatud andmevahetusvorming
LIFO	*Last In First Out,* viimasena sisse, esimesena välja. Põhimõte, kus viimasena sisestatud infoüksus töödeldatakse esimesena
GUI	*Graphical User Interface,* graafiline kasutajaliides

Sisukord

Jooniste loetelu

1 Sissejuhatus

Tallinna Tehnikaülikooli Proaktiivtehnoloogiate Teaduslaboris arendatakse traadita sensorvõrgu tehnoloogiaid. Üks labori projektidest oli Smenete (*Smart Environment Networking Technologies*), mis oli oma olemuselt „Targa Linna“ projekt, ning mille raames paigaldati üle Tallinna sensorvõrk erinevate andmete kogumiseks. Sensorandmeid kogutakse erinevatelt sensoritelt, on sündmuspõhiseid andmeid, perioodiliselt kogutavaid andmeid ja komposiitandmeid. Kõikides andmetes leidub alati ka meta-andmeid, mis sisaldavad informatsiooni mõõtmise aja, koha, täpsuse ja usaldusväärsuse kohta.

Arenduse käigus on vajalik teostada mõõtmisi andmete kogumiseks, et edasi arendada mõnda andurit või algoritmi. See protsess jaguneb kahte etappi. Alguses teostatakse mõõtmised ja seejärel hiljem analüüsitakse mõõtmistulemusi. Prototüübi tasemel seadmetega juhtub tihti vigu, näiteks polnud andur õigesti seadistatud või läks midagi käsitlemise käigus katki, mis mõõtmistel silma ei paistnud. Kuna andmete analüüs on mõõtmistest erinev, siis pannakse selliseid vigu alles analüüsi etapis tähele. Seetõttu kujuneb mõõtmiste protsess aeganõudvamaks ja kulukamaks. Seda protsessi annab teha kiiremaks, kui mõõtetulemusi näeks jooksvalt mõõtmiste ajal.

Lõputöö raames analüüsitakse erinevaid reaal-ajas andmete visualiseerimise platvorme, tutvustakse traadita sensorvõrke ning luuakse rakendus, mis saab voogedastuse kaudu helisensoritelt andmeid ning töötleb neid reaalajas, s.t et andmeid ei võeta andmebaasist. Samuti peab antud rakendus tuvastama andmete hilistumist, mis antud kontekstis tähendab seda, et andmed ei pruugi jõuda kogumispunkti mõõtmise järjekorras vaid läbisegi. Hilistumist tuleb samuti rakenduses kasutajale kuvada.

2 Analüüs

Autor otsis välja lõputöö nõuetega enim sobivaid IoT sensorandmete visualiseerimise rakendusi, mida analüüsitakse eesmärgiga selgitada, kas toodud rakendused võimaldavad lõputöö ülesandepüstituteses nõutud põhifunktsionaalsust saavutada. Analüüsi tulemusena saab lahti kirjutada mõningad arenduse funktsionaalsed ning mittefunktsionaalsed nõuded.

2.1 Olemasolevate rakenduste analüüs

Autor otsis erinevaid reaalajas sensorandmete visualiseerimist võimaldavaid programme ning tõi nende seast välja lõputöö nõuetega enim sobivamad rakendused.

2.1.1 Cumulocity IoT

Cumulocity on asjade interneti platvorm, mida kasutades on võimalik oma seadmeid üle pilve hallata ning saadud andmeid visualiseerida [1].

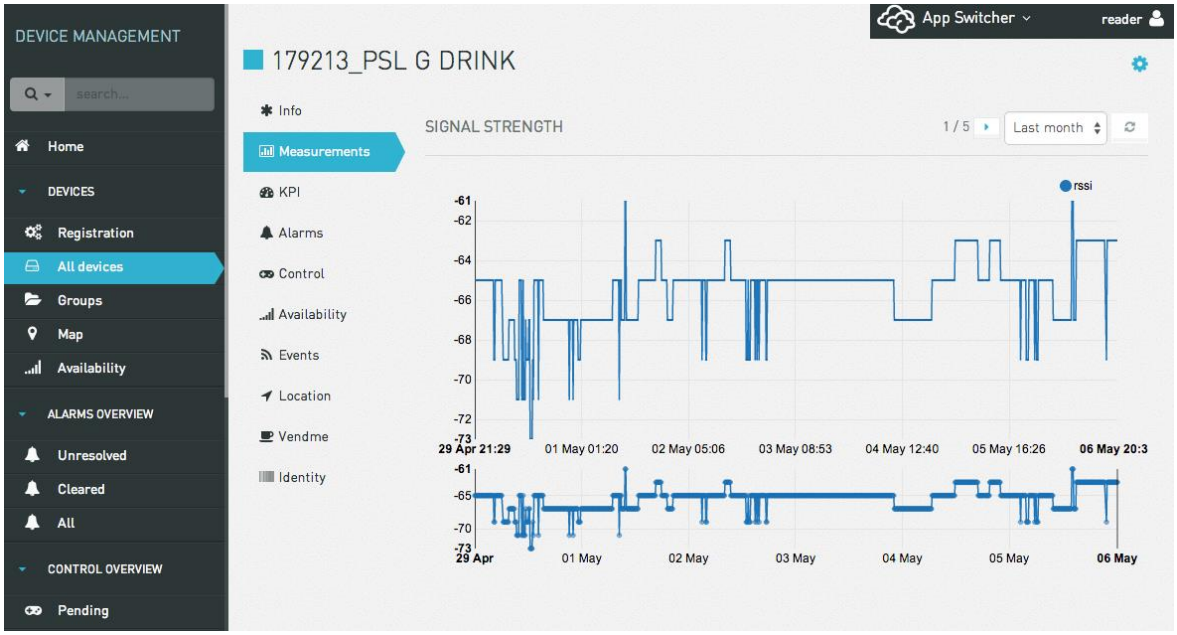
Cumulocity platvorm pakub erinevaid valmiskomponente andmete visualiseerimiseks nagu graafikud ja kaardirakendused. Samuti on võimalus Cumulocity arendusteeke kasutades luua enda pistikmoodul AngularJS raamistikus, mis vastab kasutaja seatud vajadustele. Samuti on Cumulocity'l olemas võimalus luua ühe projekti raames erinevaid kasutajaõiguste vaateid.

Andmete vastuvõtmiseks on REST API, mille jaoks on suhtleval seadmel vaja TCP/IP protokollistiku tugi ning ligipääs on määratud kasutajapõhiselt. Võimalus on ka vähendada vahendatavate andmete mahtu, eeldefineerides teatud väljad ning saates hiljem ainult väljade sisu.

Peale pilve platvormi, on ka võimalik paigaldada Cumulocity kohalikku serverisse, mis võimaldab kasutajal tarkvara enda sisevõrgus hallata. Selline võimalus vähendab turvalisusega seotud probleeme [2].

Cumulosity puudusteks on võimaluse puudumine, et süsteemis visualiseerida tuvastatud objekte ning metaandmeid nende objektide kohta. Cumulosity's on andmed alati seotud

mõõtmisi teinud seadmega. Ühte sellist näidet demonstreerib Joonis 1.



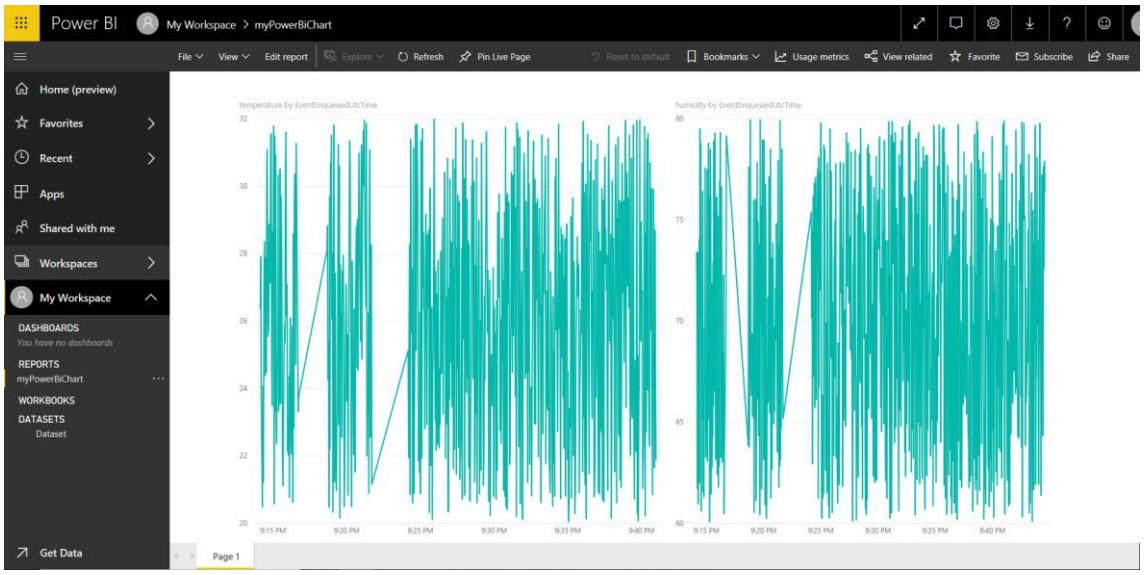
Joonis 1 Cumulocity kasutajaliides.

2.1.2 Power BI / Azure IoT hub

Power BI on Microsofti loodud ärianalüüsi teenus, mis võimaldab kasutajal visualiseerida ja analüüsida erinevaid andmeid läbi pilve, kui ka oma serveris. Power BI on kergesti integreeruv teiste Microsofti toodetega, nagu näiteks Microsoft Excel [3].

Azure IoT hub, mis on samuti Microsofti poolt väljaarendatud teenus, lubab kasutajal ühendada asjade interneti seadmeid ja pakub nendest ülevaadet ning erinevaid kontrollmeetmeid nende seadete üle. Funktsionaalsus sisaldab võimalust seadmete metaandmeid ja hetkeandmeid talletada, seadmetega erinevaid operatsioone läbiviia – taaskäivitada, seadeid lähtestada, konfigureerida, uuendada ja seadme hetkeseisukorrast teavitusi saada [4].

Mõlemad on Microsofti loodud teenused, mis on seatud omavahel ühilduma. Ning kasutades Power BI'd, et visualiseerida Azure IoT hub'st saadud sensorinformatsiooni on võimalik reaalajas visualiseerida seadmetelt tulevat teavet. Sarnaselt Cumulocity platvormile, ei ole võimalik visualiseerida tuvastatud objekte, Joonis 2.

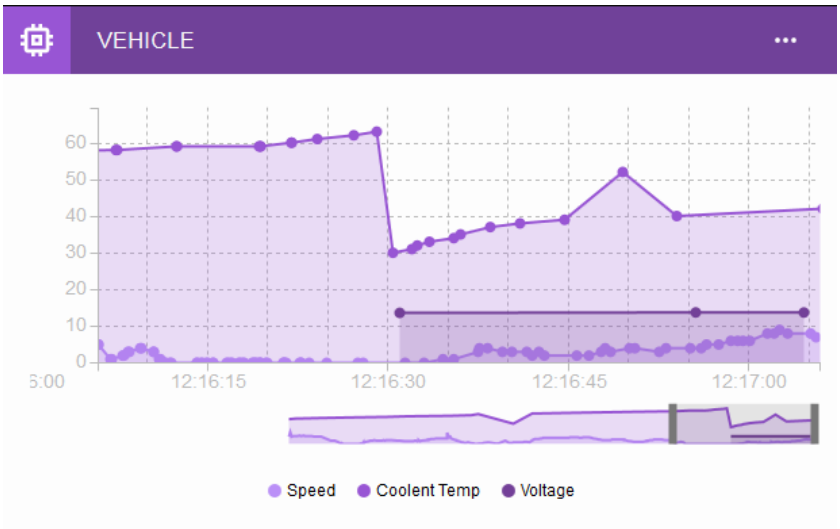


Joonis 2 Power BI kasutajaliides.

2.1.3 IBM Watson IoT Platform

Platvorm (vt. Joonis 3 IBM Watson IoT Platform kasutajaliides) pakub võimalust ühendada kasutaja IoT seadmed platvormi, neid haldada ja muuhulgas pakub ka andmete visualiseerimist brauserirakendusena ja võimaldab läbi REST API ehitada ja kohandada rakendusi, mis suhtlevad läbi Watson platvormi kasutaja seadmetega.

Watson IoT Platform aitab kasutajal luua kergelt kohandatavaid minimalistlikke diagramme, graafikuid ja tabeleid, mis kuvavad staatilisi ja dünaamilisi andmeid. Andmeid hoitakse IBM poolt hallatavas pilves [5].

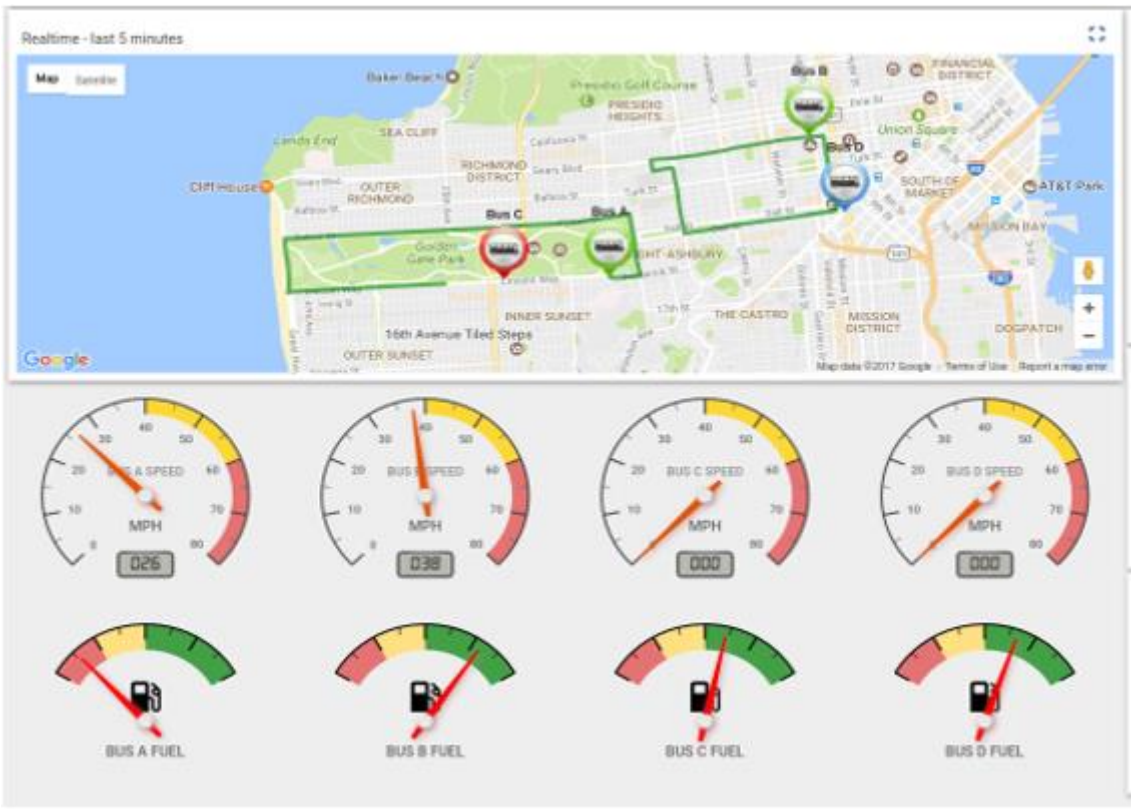


Joonis 3 IBM Watson IoT Platform kasutajaliides.

2.1.4 ThingsBoard

ThingsBoard (vt. Joonis 4) on vabavara IoT platvorm andmete kogumiseks, nende töötlemiseks, visualiseerimiseks ja seadmete halduseks. ThingsBoard pakub reaalajas andmeanalüüsi ning võimaldab kasutajal luua kohandatavaid IoT paneele, kuhu saab lisada erinevaid vidinaid, mis visualiseerivad erinevalt sensoritelt saadud andmeid.

Muuhulgas on olemas joon- ja tulpdiagrammid, mis suudavad visualiseerida nii tavaandmeid kui ka reaalajas saabuvaid andmeid [6].



Joonis 4 ThingsBoard kasutajaliides.

2.2 Nõuded

Eelnevalt tutvustatud platvormide korral on küll võimalik visualiseerida sensoritel saadud andmeid, kuid puudub nõutud põhifunktsionaalsus, milleks on, et süsteemid ei suuda

visualiseerida sündmuseid, mis ei ole otseselt seotud sensori asukohaga. Näiteks kuvada nurka müraallikani ja nurkade ristumispunktides ilmnevaid helisündmuse asukohti.

Rakendusele kehtivaid nõudeid on kahte liiki – funktsionaalsed nõuded ning mittefunktsionaalsed nõuded. Funktsionaalsed nõuded määravad ära süsteemi funktsiooni või komponendi käitumise sisendi muutmisel väljundiks [7].

Mittefunktsionaalsed nõuded on nõuded, mille järgi hinnatakse süsteemi töökäiku üldisemalt, mitte spetsiifilisi funktsioone või käitumist [8].

2.2.1 Funktsionaalsed nõuded

1. Rakendus saab andmeid voogedastuse kaudu
2. Rakendus visualiseerib helisensoritest saadud andmeid
3. Rakendus kuvab andmepakettide hilistumisi
4. Rakendus salvestab andmeid andmebaasi

2.2.2 Mittefunktsionaalsed nõuded

1. Rakendus töötab Windows operatsioonisüsteemil
2. Rakendus peab suutma kuvada korraga vähemalt 10 sensori andmeid
3. Lähtekood on inglisekeelne

3 Traadita sensorvõrgud

Traadita sensorvõrgud koosnevad üldjuhul tihedalt paigutatud sensorseadmetest, mis mitmekesi koos töötades moodustavad traadita sensorvõrgu. Iga sensorsõlm WSN sees on varustatud mõningase arvutusvõimsusega, vooluallika, transiiveri ning üldiselt vähemalt ühe ümbruskonda jälgiva sensorseadmega [9]. Kuna selliste sensorvõrgu seadmete energia on piiratud, siis on üldjuhul nende raadiolevi ulatus väga piiratud. See omakorda võib kaasa tuua vajaduse pakettide marsruutimiseks üle mitme sensorsõlme [9]. Sensorsõlmed võivad koordineerida oma tegevust teiste võrgusolevate sõlmedega, et sooritada erinevaid ülesandeid.

Need sensorseadmed mõõdavad väliskeskkonnas toimunud muutusi ning muundavad leitud andmed sensorlugemiteks, mille läbi võrgu liikumiseks kulunud aega on võimalik nii arvutada kui ka hinnata [10]. Toimunud muutuste mõõtmine võib olla kas perioodiline või sündmuspõhine. Mõlemal juhul võimaldab sensorlugemi vanus tagasiulatuvalt arvutada hinnang millal mõõdetud sündmus toimus [11]. Saadud andmete sihtpunktiks on vastuvõtja, mis käitub nagu liides kasutaja ja sensorvõrgu vahel. Vastuvõtjas on võimalus toodetud andmeid jälgida ja analüüsida [9].

3.1 Traadita sensorvõrkude topoloogiad

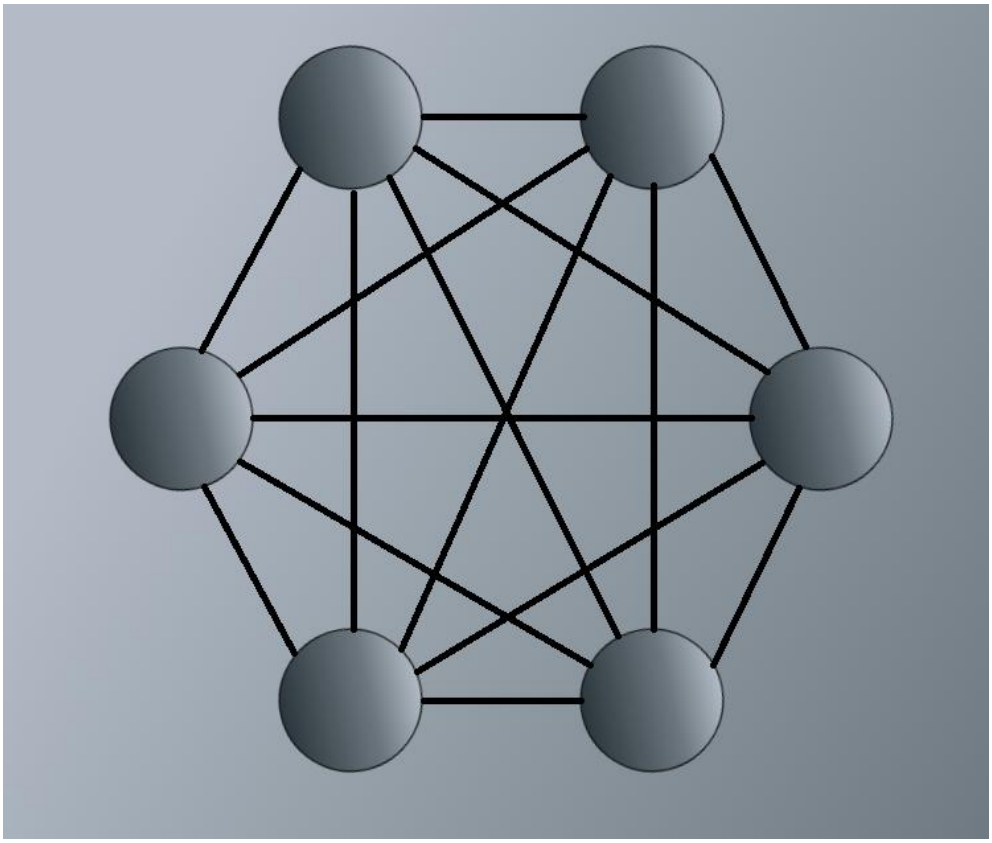
Sensorvõrkude struktuure on erisuguseid, tulenevalt võrgu topoloogiast on võrgul omad eelised ja puudused, mille olulisus oleneb sensorvõrgule seatud ülesannetest ja võimalustest.

3.1.1 Puuvõrgud

Puuvõrk on disainilt võrdlemisi lihtsakoeline. Puuvõrgu on loodud madala kuluga, madala võimsusega traadita sensorvõrgu liigiks. Sellise struktuuri korral välditakse võrgu ülekoormamist teekonna otsimise ja uuendussõnumitega, eesmärgiga vähendada võrgukasutust ja energiakulu, kasutades pakettide vahendamisel ainult vanem-laps sõlmi. Puuvõrgu puuduseks on aga vajalike tehtavate „hüpete“ arv, et andmepakett jõuaks lõpp-punkti. See tuleneb asjaolust, et pakette saab vahendada vaid vanem-laps teekondi pidi [12].

3.1.2 Mesh võrgud

Traadita mesh võrkudel on kaks põhiprintsiipi. Esiteks ei ole võrgu peale ühtegi ülem sõlme. See tähendab, et ei ole sellist sõlme, kelle kaudu peaksid andmepaketid ilmtingimata vastuvõtjasse saadetama. Seda ülesannet võib läbiviia ükskõik milline sõlm. Teiseks põhiprintsiibiks on, et iga sõlm peab olema saadaval ühenduseks teiste raadiolevi ulatuses olevate sõlmede poolt. See võimaldab andmete saatmiseks mitmeid erinevaid teekondi, ning andmed ei pea liikuma vaid vanem-laps sõlmi mööda nagu puu topoloogia korral [13]. See võib aga endaga kaasatuua liialt pikki andmepakettide liikumise ahelaid, mistõttu süsteemi koguhilistumine suureneb [9]. Mesh võrgu korral sõlmede protsessormaht, mälumaht ja võrgukasutuse maht on reeglina suurem, kui muude tutvustatud võrkude korral. Kuid andmepakettide kohalejõudmine on tõenäolisem võrreldes teiste topoloogiatega. Samuti on selline struktuur töökindlam, sest mitte töötava sensorsõlme ülesandeid võib mõni teine sõlm asendada [14]. Mesh võrgu struktuuri demonstreerib Joonis 5.



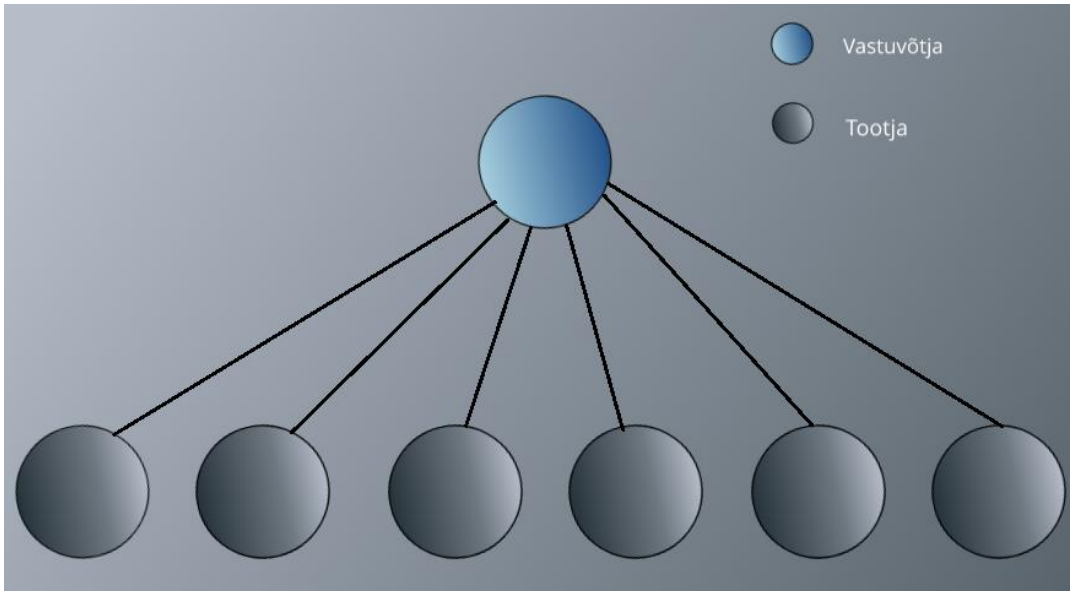
Joonis 5 Mesh topoloogial põhinev võrk.

3.1.3 Tähtvõrgud

Tähtvõrgu topoloogia kujutab endast struktuuri (vt. Joonis 6) , kus ühe vastuvõtja peale on mitmeid sensoreid, kes ilma andmeid teistele sõlmedele vahendamata, vastuvõtjale otse andmepakette saadavad [14]. Sellise võrgustruktuuri eeliseks on vähene energiakulu ja potensiaal vähesteks hilistumiseks, kuna paketid ei kuluta aega teekonna leidmisele ning on vastuvõtjale lähedal. Puudusteks on tähtvõrgu korral vastuvõtja asukoht, mis peab olema raadiolevi piires kõigist individuaalsetest sõlmedest. Ka ei ole selline topoloogia üldjuhul olla nii robustne, kui teiste topoloogiate korral, kuna töökindlus sõltub vaid ainult antud sõlmest [9].

3.1.4 Lõputöös kasutatud sensorvõrgu lühikirjeldus

Antud lõputöös kasutatava rakenduse jaoks kasutatakse tähttopoloogial põhinevad sensorvõrku. Sensorvõrku kuulub 7 sensorsõlme, mis signaalitöötluseks ja arvutamiseks kasutavad silabs EFR32 platvormi koos Cortex M4 32-bitise protsessoriga. Iga sensorsõlm kasutab sensorina akustilist massiivi mis koosneb kahest mikrofonist, mille abil määratakse mürasündmuse suunda [11]. Vastuvõtjaks on Atmel Atmega 256RFR2, mis kasutab 8-bitist protsessorit. Vastuvõtja on ühendatud Raspberry Pi'ga ning paketi saabumisel kirjutab vastuvõtja paketi serial porti. Muuhulgas on Raspberry Pi peal tarkvara, mis serial porti kuulab ning saadab kõik paketid loodud rakenduse sensorandmete vastuvõtmise mooduli. Ning nii sensorid kui vastuvõtjad kasutavad 2.4Ghz transiiverit, mis vastab IEEE 802.15.4 raadioside standardile.



Joonis 6 Tähttopoloogial põhinev võrk.

3.2 Hilistumised traadita sensorvõrgus

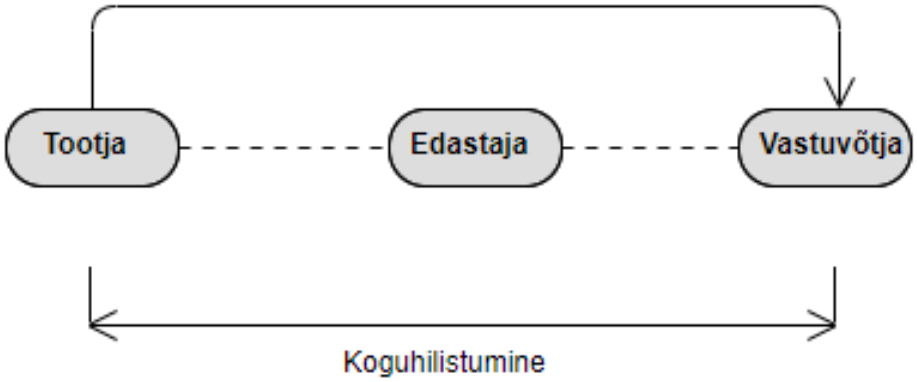
Olulisemad hilistumiste allikad on informatsiooni töötlemine, puhverdamine enne sensorlugemi võrgusõlme raadiostackile andmist [10], raadiostacki mittedeteministlikust välja saatmisest [16] ja puhverdamine võrgusõlmedes kui kasutusel on multi-hop ruutimine [15].

Hilistumiste tuvastamiseks kasutatakse End-To-End hilistumise hindamise mehhanismi [17]. Selleks oletatakse, et võrgus on kolm eri osapoolt – tootja, edastaja ja vastuvõtja (vt. Joonis 7). Vastuvõtja on kõikide andmepakettide sihtpunkt, edastaja saadab teistelt sõlmetelt saadud pakette edasi, kuid ka loob neid ise. Tootja loob pakette ning saadab neid edastajale edasi. Ehituselt on tootja ja edastaja identsed, ning iga tootja võib olla ka edastaja ning vastupidi. See millist rolli sensorsõlm täidab, oleneb võrgu topoloogiast ning hetkeolukorrast. Lõputöö kirjutamise hetkel on paigas tähttopoloogial põhinev sensorvõrk, mis tähendab, et edastajat ei kasutata kuna iga sõlm saadab enda paketid otse vastuvõtjasse.

Igal sõlmel on kaks sisemist hilistumiskomponenti – üks, mis seisneb paketi saatmisel tekkivas hilistumises ja teine, mis tekib andmete töötlemisel sõlmes. Paketi saatmisel tekkiv hilistumine jaguneb omakorda kaheks, kus üks osa on ajavahemik paketi lisamise raadiopuhvrisse ja selle paketi väljavõtmise vahel ning teine osa on ajavahemik, mis kulub andmepaketi saatmiseks [15]. Nendele hilistumisele lisandub veel hilistumine, mis tuleneb sensorsõlmede asünkroonsest ja perioodilisest käitumisest. See tähendab, et hoolimata sellest, et andmed on valmis transpordiks, ei ole näitaks sama sensorsõlme transpordikiht veel valmis sensorlugemit edastama või edastaja sensorsõlm on parasjagu energia kokkuhoidmiseks unerežiimis. Seega on sensorvõrgus kuluv paketi teekonna aeg ajas muutuv. Mis tähendab, et sensorlugemid isegi samalt sensorsõlmelt hilistuvad tarbijani jõudes iga kord erinevalt [17].

2013. aasta simulatsioonil põhineval uuringul [10] võrreldi erinevate topoloogiate korral tekkivaid koguhilistumisi ja andmepakettide kohaletoimetamise suhteid olenevalt saadetud andmepakettide arvust sekundis. Tähttopoloogia korral saatmiskiirusega 7 andmepaketti sekundis, jäi koguhilistumine 700 millisekundi juurde ning hilistumist oli

vähem, kui näiteks lineaartopoloogial põhineva sensorvõrgu korral, kus tekib lisahilistumine, kuna sensorsõlm otsib parimat teekonda läbi teiste sensorite, et oma andmed vastuvõtjale saata [10].



Joonis 7 Sensorvõrgu koguhilistumine.

4 Rakenduse arendus

Lõputöö raames valmis Angular platvormil, Node.js serveri poolena kasutav veebirakendus, kus andmebaasina kasutatakse MongoDB pilves paiknevat andmebaasi.

4.1 Rakenduse ülesehitus

Rakendus on jaotatud kaheks osaks – kliendi pool ning serveri pool. Kliendi poolel kasutatakse Angulari, mis on peamiselt Typescripti kasutav veebiraamistik.

Serveri poolel on kasutusel Node.js, mis on Javascripti käitussüsteem.

4.1.1 Kliendi pool

Kliendipoolne osa on loodud kasutades Angular veebiraamistikku, mida tavaliselt kasutatakse veebilehtede loomiseks.

Kliendipoolne osa on jaotatud kaheks põhikomponendiks.

- Display – Sisaldab endas kaardirakenduse vaadet.
- Latency-Chart – Sisaldab hilistumisvaadet.

Vastavalt Angulari ülesehitusele, tekib iga komponendi jaoks neli erinevat faili. Kus html failis asub lehe üldine ülesehitus, .less failis asuvad komponendi stiiliseaded, .spec.ts failis testid ning .ts failis asub typescripti kompileeritav kood.

Display.component.ts fail sisaldab endas kaardirakenduse kuvamist ja selleks vajalikke meetodeid. Kaart on loodud kasutades vabavaralist Mapbox API. Algselt luuaksegi kaardiinstants ning seejärel käivitatakse WebSocket, mis hakkab kuulama serveri saadetuid JSON andmepakette nurkade visualiseerimiseks ning hilistumise kuvamiseks vajaliku infoga. Websocketi eeliseks on, et mõlemad pooled on koguaeg ühenduses, ning võivad infot levitada ilma, et üks pool peaks seda eraldi teiselt pärima. See võimaldab ka kiirema ning tõhusama andmevahetuse, mis on antud ülesande puhul peamise tähtsusega.

Hilistumiskomponent tegeleb vastuvõetud hilistumiste kuvamisega. X-telg on ajatelg, ning Y-telg on hilistumise suurus millisekundites. Vaikimisi on graafikult esindatud kõikide sensorite andmed, kuid tahtmise korral on võimalik filtreerida nähtavaks ainult valitud sensori hilistumised. Hilistumiskomponendis on graafikuna kasutusel Chart.js teek.

4.1.2 Serveri pool

Serveri pool on loodud kasutades Node.js. Sensorite identifikaatorid ja neile vastavad asukohad, loetakse sisse tekstifailist. Serveri poolel võetakse vastu andmepakett ning andmetes olevate identifikaatori järgi sobitatakse see kokku failist sisseloetud andmetega. Seejärel konverteeritakse kogutud informatsioon messageObject tüüpi objektiks. See sisaldab endas sensori asukoha koordinaate, sensori saadetud nurka, sensori identifikaatorit, sensori suunda ning mõttelisi lõppkoordinaate kuuldud suunas, et oleks kliendi pool võimalik joonestada selle informatsiooni põhjal sirglõik.

4.1.3 Andmebaas

Andmebaasiks valiti MongoDB Atlas. MongoDB on No-SQL, JSON dokumendi põhine, mitterelatsiooniline andmebaas. Selline andmebaas sobib antud ülesandega paremini, kui SQL'il põhinev andmebaas, kuna on kiirem ning puudub vajadus relatsioonilise andmebaasi järele. Samuti on MongoDB JSON-laadne andmetestruktuur paremini kokkusobiv javascripti poolse Node.js serveripoolega.

4.2 Andmete formaat

Andmed saadetakse rakendusse JSON formaadis, kus nad muudetakse MessageObject objektiks. Alljärgnev nimekiri kirjeldab seal objektis sisalduvaid muutujaid.

- StringAngle – sensorilt saadud nurk kraadides sõne andmetüübina.
- id – sensori identifikaator.
- messageAge – andmete vanus millisekundites sellel hetkel, kui need vastuvõtjale saadeti.

- longitude – sensori paiknemise asukoha pikkuskraad.
- latitude – sensori paiknemise asukoha laiuskraad.
- direction – näitab sensori suuna asimuuti.

4.3 Andmete vastuvõtmine

Esimese asjana serveripoole käivitamisel luuakse side MongoDB andmebaasiga, ning loetakse tekstifailist sisse sensorite identifikaatorid, asukohad ning suunaasimuut.

Andmete vastuvõtmiseks luuakse seejärel WebSocket server, mis hakkab vastuvõtma saabuvaid helisensorite andmepakette.

Andmepaketi kohalejõudmisel filtreeritakse esimese asjana välja sensorite näidud, mis ei sisalda endas korrektset nurka. Nende näitude ’stringAngle’ väärtuseks on seatud 307, mis tähendab et sensor ei suutnud heli asukohta positsioneerida.

Filtreerimisest edasipääsenud paketi andmetest luuakse MessageObject objekt. Selle jaoks sobitatakse andmepakett kokku eelnevalt tekstifailist sisseloetud andmetega. Identifikaatori kaudu andmed kokku viies, saab määrata saabunud andmetele laiuskraadi, pikkuskraadi ning suunaasimuudi.

Peale andmete kokkuklapitamist arvutatakse välja tekkiva lõigu lõpp-koordinaadid, et andmeid saaks kuvada kaardi peal. Lõigu pikkuseks on määratud 60 meetrit. Kuna asukoha koordinaadid on UTM formaadis aga visualiseerimiseks on tarvilikud WGS84 formaadis koordinaadid, siis muundatakse nad kasutades Pro4js teeki korrektsesse formaati. Peale seda salvestatakse saadud objekt andmebaasi ning tagastatakse, misjärel saadetakse objekt WebSocket’it kasutades Angulari rakendusse.

Rakendusse saabub sekundis ligikaudu 35 sõnumit, tulenevalt sellest, et üksik sensor saadab umbkaudu iga 205 millisekundi tagant ühe andmepaketi. Antud töö piires oli sensorvõrgus 7 sensorit.

4.4 Andmete visualiseermine

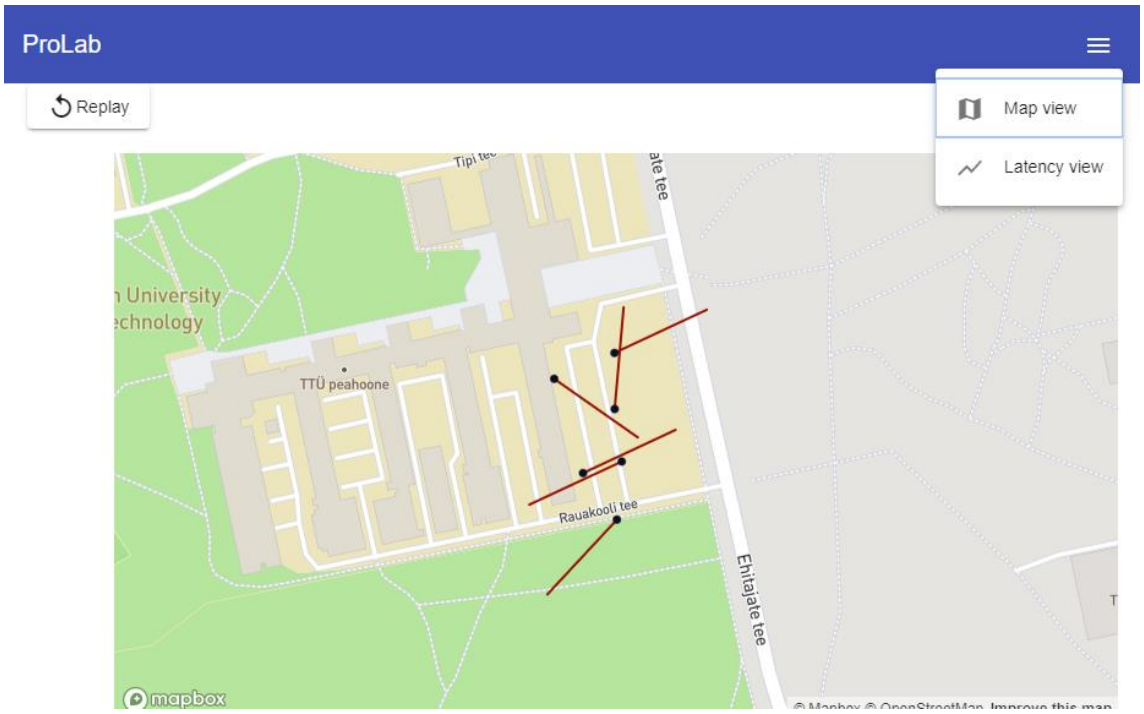
Andmete visualiseerimine toimub kliendipoolses osas eelnevalt serveri poolel töödeldud andmetega.

4.4.1 Nurkade visualiseerimine

Helisensorite saadetud nurkade visualiseerimine (vt. Joonis 7) toimub display moodulis. Esmalt luuakse MapBox kaardiplatvormi kasutades uus kaart. Seejärel luuakse ühendus serveri poolega ning hakatakse vastuvõtma andmepakette. Iga sensori asukohale märgitakse tumesinine ring, millele pealevajutades näeb valitud sensori identifikaatorit.

Edasi luuakse Turf.js teeki kasutades vastuvõetud andmepaketist saadud informatsioonist sirglõigud, mis algavad sensori asukohast ning lõppevad kuuldud müra-allika suunas 60 meetri kaugusel. Kui kaardi peal on juba samalt sensorilt kuvatud nurk, siis eemaldatakse see. Samuti kontrollitakse, kas kaardi peal leidub nurki, mis on vanemad kui 5 sekundit. Kui leidub, siis eemaldatakse need.

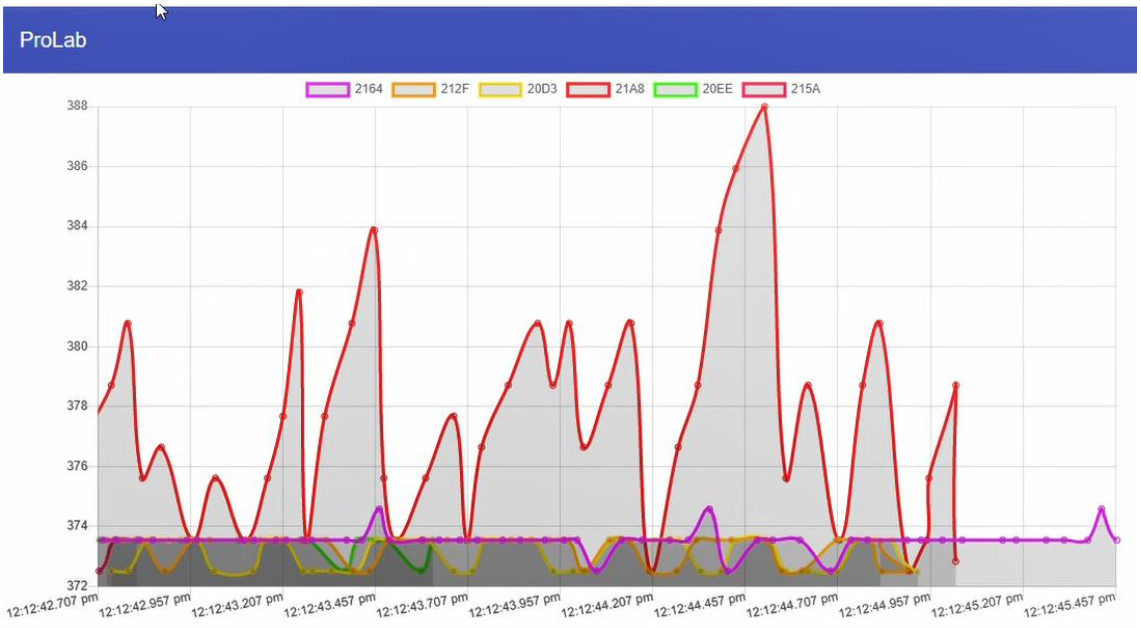
Nurkade visualiseerimist saab ka järelevaadata. Selleks on vaja vajutada „Replay“ nuppu ning sisestada alg- ja lõppaeg. Selle tulemusena puhatatakse kaart ning saadetakse serverisse sõnum antud aegadega. Seepeale saadab server päringu andmebaasi ning hakkab samasuguse ajalise vahega, nagu andmepaketid algselt saabusid, nurki ettemängima. Selleks ajaks peatatakse reaalajas saabuvate andmepakettide edasisaatmine, et omavahel ei läheks segamine järelvaadatavad ja reaalajas saabuvad andmed. Järelvaatamise peatamiseks tuleb vajutada uuesti „Replay“ nuppu ning seejärel „Stop replay“. Selle tegevuse järel puhastatakse kaart eelnevatest andmetest ning saadetakse serverisse sõnum, mille tagajärel hakkab server uuesti vastuvõtma ja edastama reaalajas saabuvaid andmeid.



Joonis 8 Nurkade visualiseermine.

4.4.2 Hilistumiste visualiseerimine

Hilistumiste visualiseerimine (vt. Joonis 7) toimub ’latency-chart’ moodulis. Sensoritelt saadetud andmetega on alati kaasas sensorlugemi vanus – messageAge. See vanus on sensori poolt arvutatud vahetult enne paketi lisamist raadiopuhvrisse. Paketi saabumisel sensorinfot vastuvõtvasse klassi valitakse talle üks värv ning lisatakse graafikule. Kui graafik hakkab täis saama eemaldatakse graafikult esimene punkt, et ruumi teha järgmistele.



Joonis 9 Hilistumiste visualiseermine.

5 Võimalikud edasiarendused

Lõputöö käigus loodud rakendus on prototüübi tasemel abitööriist Proaktiivtehnoloogiate Teaduslaborile. Sellest tulenevalt on edasiarendamise võimalusi mitmeid.

Üks edasiarenduse suund võiks olla analüüsi tegemine hoitud andmetega. Hetkel salvestab rakendus saadud andmed andmebaasi, kuid juba salvestatud andmeid rakenduse töös, peale nurkade taaskuvamise, uuesti ei kasutata. Üks võimalus oleks kasutada salvestatud andmeid, et näidata ajaskaalal erinevate sensorite tööparameetreid

Teine edasiarenduse suund võiks olla suurem rakenduse integreerimine sensoritega ja veelgi detailsem metainfo kuvamise võimalus. Oleks võimalus saata sensoritelt rohkem informatsiooni rakendusse, et näiteks jälgida sensori aku taset või muid parameetreid.

6 Kokkuvõte

Töö põhieesmärgiks oli luua rakendus, mis kiirendaks Proaktiivtehnoloogiate Teaduslabori läbiviidavaid traadita sensorvõrgu arendusi, võimaldades reaalajas näha töötavaid helisensoreid, nende toodetavaid andmeid ning andmepakettide hilistumisi unikaalsete sensorite lõikes.

Lõputöö esimeses pooles tutvustati lõputöö nõuetega enim kokkulangevaid kommertsrakendusi ning tuvastati, et uuritud rakendused ei rahulda lõputöö ülesandepüstituse nõudeid. a määrati lõputöö rakendusele käivad funktsionaalsed ja mittefunktsionaalsed nõuded. Samuti anti lühike ülevaade erinevatest traadita sensorvõrkudest ning nendes tekkivatest hilistumistest.

Lõputöö teises pooles kirjeldati valitud tehnoloogiaid ning seletati lahti rakenduse töötamise põhimõtted.

Selle tööga sai valmis esialgne prototüüp, millega on võimalik jälgida sensorite tööd reaalajas ning tuvastada tekkivaid hilistumisi. Autorile andis töö tegemine juurde teadmisi sensorvõrkude ülesehituse ja reaalajas tarkvaraga suhtlemise poolest, ning samuti teadmisi erinevate teekide ja raamistikega töötamisest.

Kasutatud kirjandus

[1] „Introduction to Cumulocity,“ [Võrgumaterjal]. Available: https://www.cumulocity.com/guides/concepts/introduction/. [Kasutatud 18 03 2019].

[2] „Cumulocity IoT platvorm liidab kõik seadmed ühtsesse süsteemi,“ [Võrgumaterjal]. Available: https://iot.ttu.ee/et/cumulocity-iot-platvorm-liidab-koik-seadmed-uhtsesse-susteemi/. [Kasutatud 19 3 2019].

[3] „What is Power BI?,“ [Võrgumaterjal]. Available: https://powerbi.microsoft.com/en-us/what-is-power-bi/. [Kasutatud 20 3 2019].

[4] „What is Azure IoT Hub,“ [Võrgumaterjal]. Available: https://www.c-sharpcorner.com/article/what-is-azure-iot-hub/. [Kasutatud 20 3 2019].

[5] „Securely connect, manage and analyze IoT data with Watson IoT Platform,“ [Võrgumaterjal]. Available: https://www.ibm.com/internet-of-things/solutions/iot-platform/watson-iot-platform. [Kasutatud 20 03 2019].

[6] „How to Unlock the Power of IoT Data with Visualization Tools,“ [Võrgumaterjal]. Available: https://www.iotforall.com/iot-data-visualization-tools/. [Kasutatud 20 03 2019].

[7] R. Fulton ja R. Vandermolen, „Requirements - Writing Requirements,“ %1 *Airborne Electronic Hardware Design Assurance: A Practitioner's Guide to RTCA/DO-254*, CRC Press, 2014, pp. 89-93.

[8] L. Chen, A. B. Muhammad ja N. Bashar, „Characterizing Architecturally Significant Requirements,“ *IEEE Software,* kd. 30, nr 2, p. 38–45, 2013.

[9] M. A. Matin ja N. Islam, „Overview of Wireless Sensor Network,“ %1 *Wireless Sensor Networks – Technology and Protocols*, INTECH, 2012, pp. 1-22.

[10] M. F. Khan, E. A. Felemban, S. Qaisar ja S. Ali, „Performance Analysis on Packet Delivery Ratio and End-to-End Delay of Different Network Topologies in Wireless Sensor Networks (WSNs),“ *IEEE Xplore,* pp. 324-325,327-328, 2013.

[11] J. Ehala, J. Kaugerand, R. Pahtma , S. Astapov, A. Riid, J.-S. Preden ja L. Mõtus, „Situation awareness via Internet of things and in-network data processing,“ *International Journal of Distributed Sensor Networks,* kd. 13, nr 1, 2017.

[12] W. Qiu, S. Efstratios ja H. Peng, „Enhanced tree routing for wireless sensor networks,“ *Ad Hoc Networks,* kd. 7, nr 3, p. 638, 2009.

[13] S. D. Odabasi ja A. H. Zaim, „Wireless Mesh Sensor Networks: Advantages, Difficulties and Interconnection Methods.,“ %1 *International Conference on Networking and Future Internet,* Istanbul, 2012.

[14] S. Sharma, D. Kumar ja K. Kishore, „Wireless Sensor Networks-A Review on Topologies and Node Architecture,“ *International Journal of Computer Sciences and Engineering,* nr 1, pp. 19-25, 2013.

[15] P. Pinto, A. Pinto ja M. Ricardo, „End-to-end delay estimation using RPL metrics in WSN,“ *IEEE Xplore,* pp. 1-2, 2013.

[16] K. Herman, Real-time systems: design principles for distributed embedded applications., Springer Science & Business Media, 2011.

[17] J. Kaugerand, J. Ehala, L. Mõtus ja J.-S. Preden, „Time-selective data fusion for in-network processing in ad hoc wireless sensor networks,“ *International Journal of Distributed Sensor Networks,* kd. 14, nr 11, 2018.

[18] „BlockingQueue,“ [Võrgumaterjal]. Available: https://docs.oracle.com/javase/8/docs/api/?java/util/concurrent/BlockingQueue.html. [Kasutatud 20 04 2019].

[19] „Class Task<V>,“ [Võrgumaterjal]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/concurrent/Task.html. [Kasutatud 24 04 2019].

[20] „Platform,“ [Võrgumaterjal]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/application/Platform.html. [Kasutatud 16 04 2019].

[21] „Licensing and Pricing,“ [Võrgumaterjal]. Available: https://www.teamdev.com/jxmaps#licensing-pricing. [Kasutatud 20 05 2019].