LAPIN AMK
Lapland University of Applied Sciences

# AJONEUVOJEN SENSOREIDEN ULKOTESTAUS- JA DATANKERÄYSJÄRJESTELMÄ

WinterSim-projekti

Piitsalo Miko

Opinnäytetyö

Tieto- ja viestintätekniikka
Insinööri (AMK)

2021

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tieto- ja viestintätekniikka
Insinööri (AMK)

---


| | | | |
|---|---|---|---|
| **Tekijä** | Miko Piitsalo | Vuosi | 2021 |
| **Ohjaaja** | Erkki Mattila | | |
| **Toimeksiantaja** | FrostBit-ohjelmistolaboratorio | | |
| **Työn nimi** | Ajoneuvojen sensoreiden ulkotestaus- ja datankeräysjärjestelmä | | |
| **Sivumäärä** | 51 | | |


---


Tässä opinnäytetyössä selvitettiin, millaisilla laitteisto- ja sovellusratkaisuilla voidaan toteuttaa ajoneuvojen sensoreiden ulkotestaukseen ja sensoridatan keräämiseen tarkoitettu järjestelmä.

WinterSim-projektissa selvitetään, miten autonomisten ajoneuvojen sensoreiden ja järjestelmien toimivuutta talviolosuhteissa voidaan simuloida peliteknologioita hyödyntämällä. Realistisen simulaation tuottamiseksi projektissa kerätään runsaasti oikean elämän sensoridataa eri sääolosuhteista, jotta nähdään, miten ne vaikuttavat sensoreiden toimintaan. Lisäksi opinnäytetyössä luotiin kokoelma sovelluksia, jotka mahdollistavat datankeräyksen seuraamisen sekä halutun datan etsimisen ja purkamisen analyysiä varten. Järjestelmä toteutettiin talvella 2020 – 2021 Lapin AMK:n FrostBit-ohjelmistolaboratorion WinterSim-projektia varten.

Opinnäytetyön tuloksena syntyi järjestelmä, joka kykenee tallentamaan ja jäsentämään suuria määriä sää- ja sensoridataa hyödyntämällä tietokantaa ja objektintallennuspalvelua.



Avainsanat autonomiset järjestelmät, mittausasemat, tiedonkeräys, tietokantasuunnittelu, verkkosovellukset, WinterSim

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Degree Programme in Information and Communications Technology
Bachelor of Engineering

---

**Author** Miko Piitsalo Year 2021
**Supervisor** Erkki Mattila
**Commissioned by** FrostBit Software Lab
**Subject of thesis** Vehicle sensor outdoor testing and data collection system
**Number of pages** 51

---

The aim of this thesis study was to describe what kind of hardware and software solutions can be utilized in creation of a system for vehicle sensor outdoor testing and data collection.

The objective of WinterSim project is to produce data and knowledge on how autonomous vehicle sensors and systems performance in winter conditions can be simulated using game technologies. To create a realistic simulation, vast amounts of real sensor data was collected in various weather conditions to determine what impact, for example, a heavy snowfall has on sensor performance. In addition, a collection of applications was created, which allow monitoring data collection, as well as searching and downloading data for analysis. The system was built during winter 2020 - 2021 for use in WinterSim project at FrostBit software laboratory of Lapland UAS.

The result of the thesis study is a system which can store and structure great amounts of weather and sensor data by utilizing a database and an object storage service.

Key words autonomous systems, database design, data collection, measuring stations, web applications, WinterSim

SISÄLLYS

# ALKUSANAT

Haluan kiittää Lapin ammattikorkeakoulua ja erityisesti sen FrostBit-sovelluslaboratoriota tarjotusta työtehtävästä WinterSim-projektissa, josta sain opinnäytetyölleni paitsi yleisesti mielenkiintoisen myös minulle itselleni mielekkään aiheen.

Iso kiitos erikseen myös WinterSim-projektiryhmälleni, jonka kanssa on ollut mukava työskennellä projektin parissa. Ryhmän jäsenet ovat myös auttaneet ja neuvoneet opintojen loppuun saattamista ja opinnäytetyötä koskevien käytännön seikkojen kanssa. Lisäksi projektia varten kerätty tutkimustieto ja projektiryhmäläisten työtehtävistään kirjoittamat raportit ovat helpottaneet opinnäytetyöhön liittyvää tutkimista ja kirjoittamista.

Kiitos myös kaikille muille ystäville sekä perheenjäsenille, jotka ovat auttaneet ja kannustaneet opintoihin ja opinnäytetyöhön liittyen.

# KÄYTETYT LYHENTEET JA TERMIT

| | |
|---|---|
| ACID | Atomicity, Consistency, Isolation, Durability |
| API | Application Interface, rajapinta |
| ASCII | American Standard Code for Information Interchange |
| AWS | Amazon Web Services |
| CSS | Cascading Style Sheets, verkkosivuihin käytetty tyylikieli |
| Docker | virtualisointiohjelmisto |
| Framework | ohjelmointikehys |
| GUI | Graphical User Interface |
| JSON | JavaScript Object Notation |
| JWT | JSON Web Tokens |
| LIDAR | Light detection and Ranging, lasertutka |
| NoSQL | relaatiomallista poikkeava tietokantamalli |
| PyPI | Python Package Index |
| REST | Representational State Transfer |
| ROI | Regions of Interest (Zhao, Sun, Xu, Min & Yu 2020, 4901 – 4902) |
| ROS | Robot Operating System (Open Robotics 2014) |
| SAE | Society of Automotive Engineers |
| SCSS | Sassy CSS, CSS-kielen jatke |
| SQL | Structured Query Language |
| Tkinter | käyttöliittymäkirjasto Pythonille |
| Ubuntu | Linux-pohjainen käyttöjärjestelmä |
| UMB | Universal Measurement Bus, protokolla (Lufft 2019, 4) |
| Unreal | pelimoottori |

# 1 JOHDANTO

Opinnäytetyön aiheena on selvittää, miten luodaan autonomisten ajoneuvojen sensoreiden ulkotestaukseen tarvittava datankeruujärjestelmä. Opinnäytetyö on osa Lapin AMK:n FrostBit-ohjelmistolaboratorion WinterSim-projektia ja FrostBit toimii opinnäytetyön toimeksiantajana.

Kun syksyllä 2020 yritin keksiä opinnäytetyölleni aihetta, kertoi luokkakaverini mielenkiintoisesta projektista, jonka parissa hän työskentelee FrostBit-ohjelmistolaboratoriossa. Lähetinkin pikimmiten sähköpostia asiasta ja ei aikaakaan, kun olin löytänyt opinnäytetyölleni mitä ajankohtaisimman ja mielenkiintoisimman aiheen.

Mitä pidemmälle tulevaisuutta kohti mennään, sitä enemmän kasvaa automaation määrä ja tarve maailmanlaajuisesti kaikissa liikennemuodoissa. Jotta itseajavat ajoneuvot olisivat luotettavia ja ennen kaikkea turvallisia, tarvitaan niiden hyödyntämien tekoälyjen ja algoritmien pohjaksi paljon tutkittua tietoa ja dataa. Yksi tapa tehostaa tällaisen tiedon hankkimista ja ajoneuvojen testaamista on hyödyntää erilaisia simulaatioratkaisuja. Jotta sensoreiden simulointi olisi mahdollisimman realistista, täytyy simulaation perustua oikeaan dataan. Siksi WinterSim-projektissa kerätäänkin iso otanta tosielämän sensoridataa eri säätiloista.

Tällaisen datan kerääminen vaatii mittausaseman rakentamista, johon tulee valikoitujen sensoreiden lisäksi sääasema. Mittausasemalle ja sen ulkopuolelle ohjelmoitu sovelluskokonaisuus kerää sensoridatan, yhdistää sen käsitteellisesti nykyiseen säätilaan ja varastoi sen myöhempää analyysiä varten. Prosessin seuraamista ja jälkianalysointia varten tarvitaan useita tukisovelluksia.

Tässä opinnäytetyössä kerron, mistä laitteisto- ja ohjelmistokomponenteista tällainen mittausasema ja datankeräysjärjestelmä rakennetaan, millaisia haasteita tällaiseen moniosaiseen järjestelmään liittyy ja minkälaisia varotoimenpiteitä voi tehdä erilaisten ongelmatilanteiden välttämiseksi. Lisäksi perustelen tekemiäni suunnitteluun liittyviä ratkaisuja ja pohdin jälkeenpäin, oliko ratkaisu toimiva. Toteutusten kuvausten painopiste on sovelluspuolella, koska siihen työtehtäväni projektissa pääosin liittyivät.

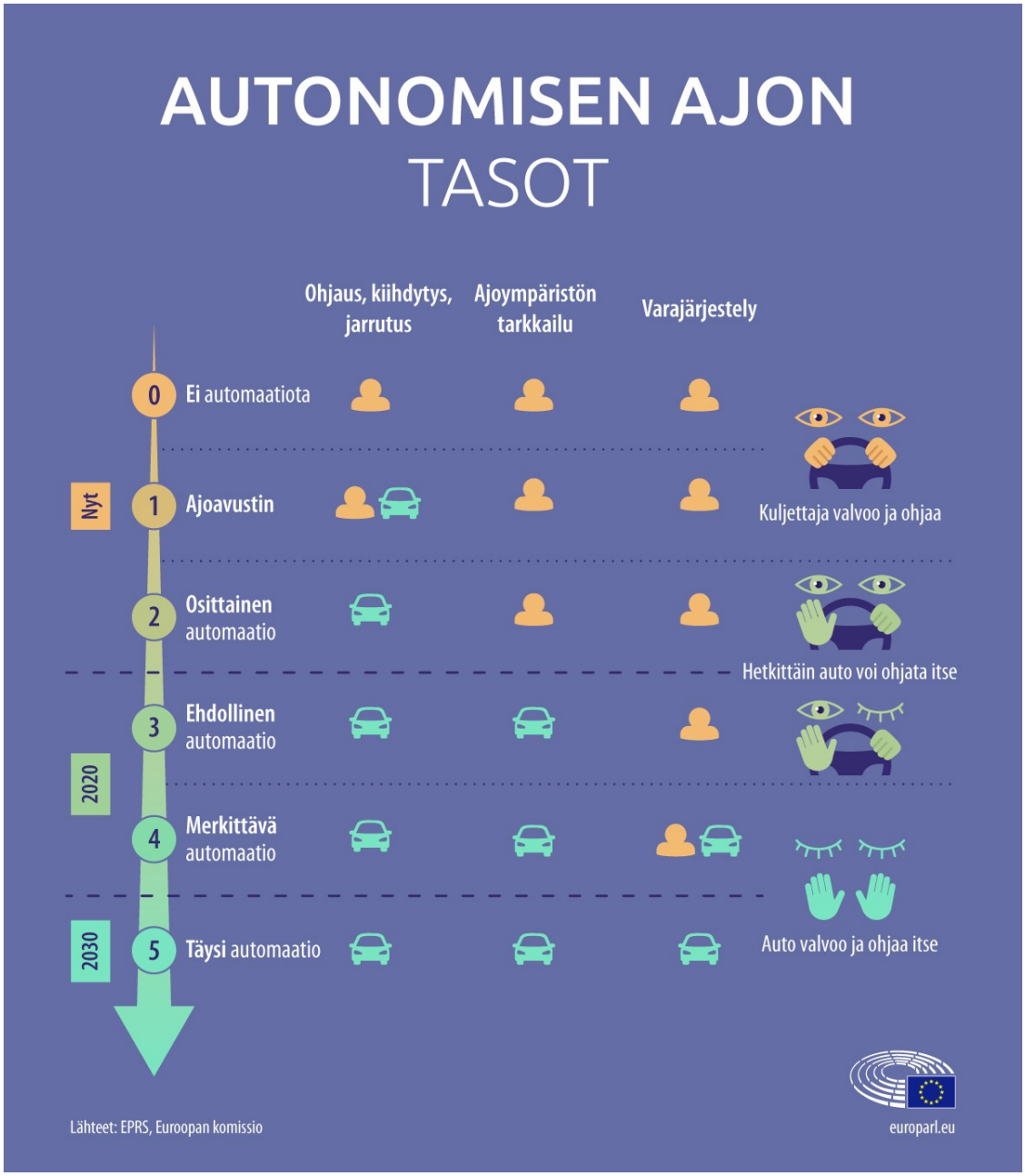
# 2 PROJEKTIN TAUSTATIETOA

## 2.1 Autonomiset ajoneuvot ja niiden sensorit

Autonominen eli itseajava ajoneuvo on kulkuväline, joka hyödyntää sensoreita ja tekoälyä matkustamiseen osittain tai kokonaan ilman ihmisen ohjausta. Joitain huomattavia autonomisten ajoneuvojen parissa työskenteleviä yrityksiä ovat esimerkiksi Audi, BMW, Ford, Google, General Motors, Tesla, Volkswagen ja Volvo. (Lutkevich 2019.)

Ajoneuvon itsenäisyyden määrä kuljettajasta ei kuitenkaan ole mustavalkoista. Eri automaation tasoja on pyritty kuvaamaan esimerkiksi SAE:n standardilla J3016. Standardi jaottelee ajoneuvot kuuteen eri luokkaan (numeroilla 0 – 5) riippuen automaation tasosta. Tasolla 0 ajaminen on täysin manuaalista, kun taas tasolla 5 ajaminen on täysin automaattista. SAE:n standardi on laajimmin hyväksytty ja käytössä oleva luokittelu autonomisille ajoneuvoille. (Kahala 2020, 4 – 5.)

Nykyään tasojen 1 ja 2 mukaisia ajoneuvoja on jo markkinoilla Euroopassa. Euroopan parlamentin vuonna 2019 julkaiseman strategian (Kuvio 1) mukaan tasojen 3 ja 4 mukaisia autoja testataan parhaillaan ja niiden arvioidaan yleistyvän 2020-luvun aikana, kun taas tason 5 mukaisia, täysin automaattisia robottiautoja saa kuitenkin odottaa vuoteen 2030 asti. (Euroopan parlamentti 2019, 2.)

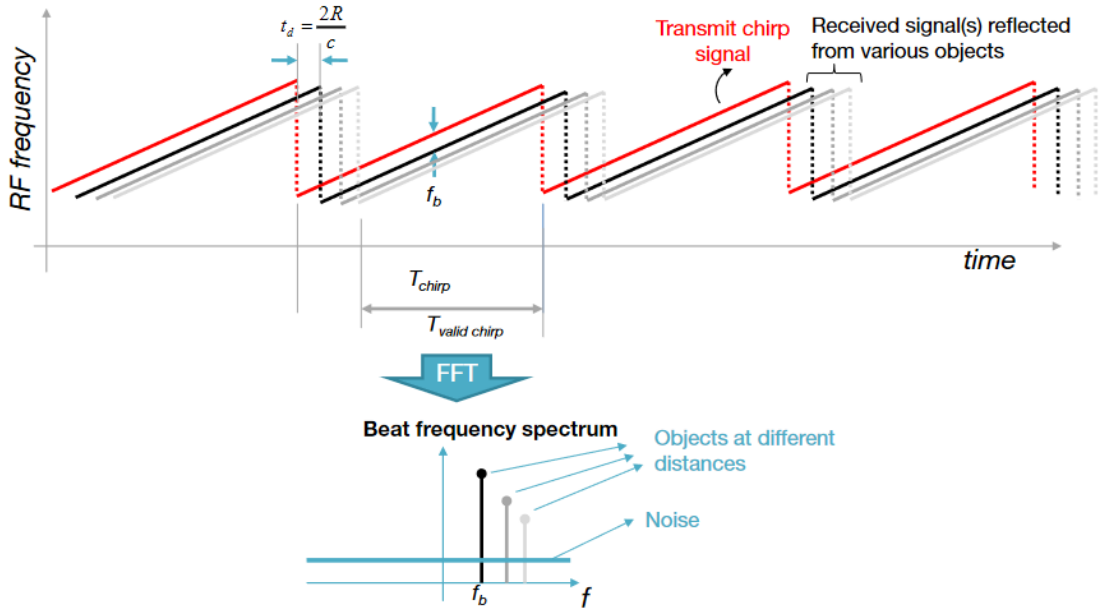



Kuvio 1. Autonomisen ajon tasot ja kehitys (Euroopan parlamentti 2019, 4)

On kuitenkin huomioitava, että vuonna 2020 alkanut koronaviruspandemia ja siitä johtuvat ilmiöt ja rajoitukset ovat osaltaan hidastaneet autonomisten autojen kehitys- ja testaustyötä sekä tuoneet epävarmuutta autoalalle ja maailmantalouteen yleensä (Metz & Griffith 2020). Liikenteen kokonaisvaltainen automaatio on kuitenkin matkalla, ja siihen ollaan valmistautumassa myös poliittisilla ja juridisilla muutoksilla. Esimerkiksi Suomessa lakia liikennejärjestelmästä ja maanteistä päivitettiin vuonna 2018 sisällyttämään liikenteen automaatioon sekä päästöjen

vähentämiseen liittyviä seikkoja (Sorokin 2020, 20). Lisäksi kesällä 2020 Suomessa vaihdettiin teiden sulkuviivojen väri keltaisesta valkoiseksi, muun muassa juuri siksi, että robottiautot ja toisaalta myös ihmisetkin erottavat valkoisen värin paremmin (Koskinen 2020).

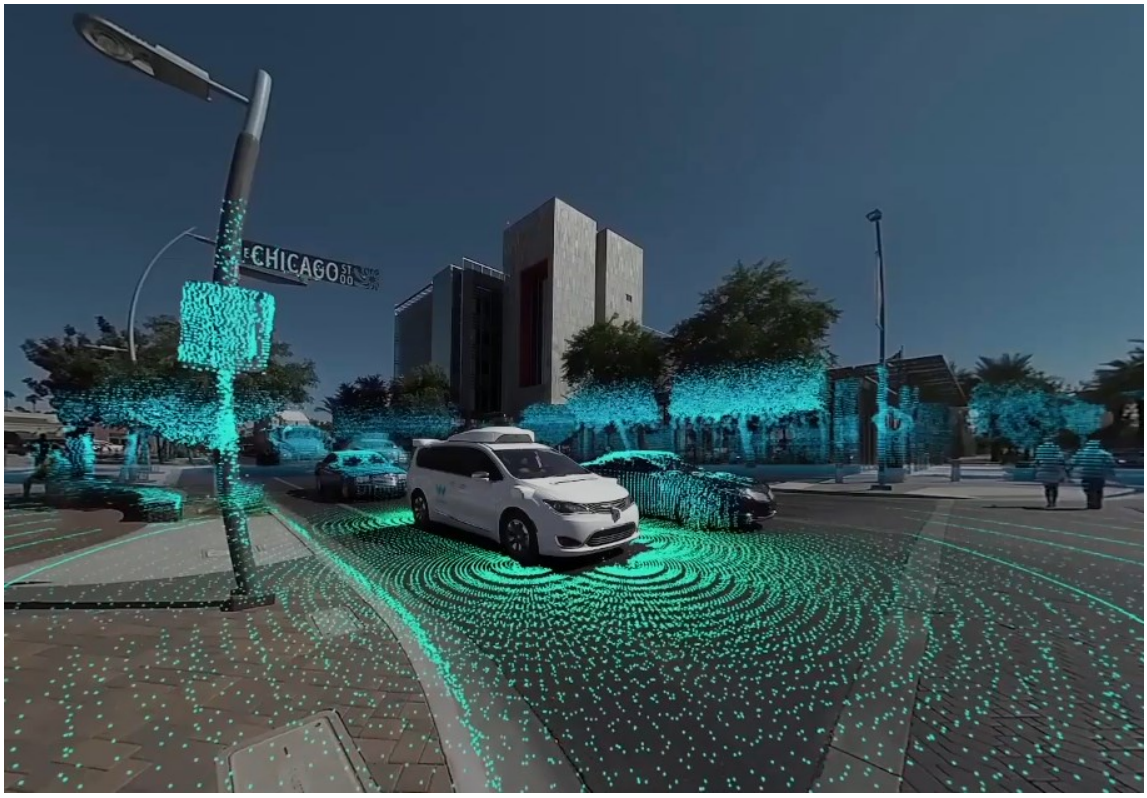
Autonomisissa ajoneuvoissa käytetään pääasiallisesti kolmea sensorityyppiä: tutkaa, lasertutkaa (josta käytetään myös lyhennettä LIDAR) sekä kameraa (Burke 2019). Tutkan toiminta perustuu sen tuottaman elektromagneettisen säteilyn heijastumiseen kohteista. Lähtevän säteilyn takaisin heijastumiseen vaaditusta ajasta voidaan suoraan laskea kohteen etäisyys ja kulma tutkaan nähden (Kuvio 2). Lisäksi antamalla tutkalle parametreina ajoneuvon nykyinen nopeus ja suunta, voi se hahmottaa myös kohteen nopeutta ja suuntaa, joiden avulla voidaan ennustaa kohteen tulevaa liikerataa sekä erottaa liikkuvat kohteet paikallaan olevasta ympäristöstä. (Händel, Konttaniemi & Autioniemi 2018, 18 – 19.)




Kuvio 2. Tutkan toimintaperiaate kohteiden havaitsemiseen (Ramasubramanian & Ginsburg 2017, 2)

Lasertutka perustuu pitkälti samaan toimintaperiaatteeseen kuin perinteinenkin tutka, paitsi että elektromagneettisen säteilyn sijasta vertaillaankin valonsäteiden

lähtö- ja saapumisparametreja. Takaisin heijastuneiden säteiden avulla rakennetaan ympäristöstä kolmiulotteinen pistepilvi (Kuvio 3). Lasertutkan etuina on kuvan resoluutio verrattuna perinteiseen tutkaan ja usein myös laajempi näkökenttä. Etenkin mekaanisesti pyörivä lasertutka kykenee näkemään täydet 360-astetta ympärilleen. Mekaanisesti pyörivät lasertutkat ovat kuitenkin kalliita ja isokokoisia, minkä takia autoala on alkanut käyttämään kiinteitä lasertutkia yhä enemmän. (Khader & Cherian 2017, 3.)




Kuvio 3. Havainnekuva ajoneuvossa käytettävän lasertutkan toiminnasta (Cooper Hewitt 2018)

Kamera on yksi ensimmäisistä sensoreista, jota on käytetty itseajavissa ajoneuvoissa. Verrattuna muihin sensoreihin, kamerassa on etuna sen resoluutio, saatavuus ja edullisuus. Kameran huono puoli on kuitenkin se, että datan käsittelyyn tarvitsee reilusti enemmän laskentatehoa, jotta siitä voidaan tuottaa ajoneuvon ohjaamisen kannalta hyödyllistä tietoa riittävän nopeasti. (Kocić, Jovičić & Drndarević 2018, 3.)

Modernit datan käsittelymenetelmät pyrkivät yhdistelemään sensoreiden vahvuuksia. Esimerkiksi yksi tapa on ensin etsiä lasertutkan antamasta pistepilvestä niin sanottuja ”kiinnostavia kohtia” (Regions of Interest), joissa mahdolliset ympäristön esteet ja muut liikenteenkäyttäjät todennäköisimmin sijaitsevat. Kun kohteet näin ensin paikannetaan lasertutkalla ja sitten koneoppimis- ja kuvantunnistusalgoritmeja hyväksi käyttäen tunnistetaan kameran kuvasta, on käsittelyprosessi luotettavampi ja nopeampi. (Zhao ym. 2020, 4901 – 4902.)

Autonomisten ajoneuvojen hyödyntämien sensoreiden käyttöön tieliikenteessä liittyy kuitenkin vielä huomattavasti haasteita. Yksi näistä haasteista on erilaisten säätilaan liittyvien seikkojen, kuten sateen, sumun, lumisateen tai jäisen tienpinnan vaikutus sensoreiden toimintaan. Tätä haastetta WinterSim-projektissa tutkitaan.

## 2.2 WinterSim-projekti

WinterSim-projektin tavoitteena on tuottaa tutkittua tietoa ja dataa autonomisten ajoneuvojen yleisimpien sensoreiden ja järjestelmien simuloinnista talviolosuhteissa pelimoottoreita hyödyntämällä. Projektin päätuotoksina syntyy: nykytilakatsaus sensoreiden ja järjestelmien simulointiin ajoneuvoteollisuudessa, kenttäolosuhteissa kerättyä referenssidataa, avoimen lähdekoodin simulaatioympäristöön toteutetut ohjelmointiratkaisut sekä raportteja ja tieteellisiä julkaisuja aiheista. (Lapin AMK 2020a.)

WinterSim-projekti on aloitettu kesällä 2020, ja se kestää kaksi vuotta. Projektin kokonaisbudjetti on 571 200 euroa, ja sen rahoituslähteenä on Business Finland. (Lapin AMK 2020b.)

Projektin referenssidatan keräämistä kenttäolosuhteissa koskeva osuus on päätetty jakaa kahteen jaksoon: stationaariseen ja liikkuvaan mittaukseen. Ensimmäisenä mitataan sensorien toimintaa stationaarisesti talvikaudella 2020 – 2021. Liikkuva mittaus toteutetaan jonkinlaista testiajoneuvoa käyttäen talvikaudella 2021 – 2022. (Lapin AMK 2020a.)

Referenssidatan kerääminen ei käy tuosta vain. Erilaisten sensoreiden järjestelmälliseen ja automatisoituun testaamiseen tarvitaan väkisin jonkinlainen prosessia tukeva järjestelmä. Kenttäolosuhteissa testaamiseen täytyy rakentaa asema, johon testattavat sensorit kiinnitetään. Aseman täytyy hoitaa virransyöttö kaikille komponenteille, tiedon vastaanottaminen sensoreilta, tiedon siirtäminen tallennuspaikkaan sekä säätä huonosti kestävien komponenttien eristäminen ja lämmittäminen. Tällaista mittausasemaa projektiryhmä lähti suunnittelemaan ja toteuttamaan syksyllä 2020.

# 3 MITTAUSASEMA

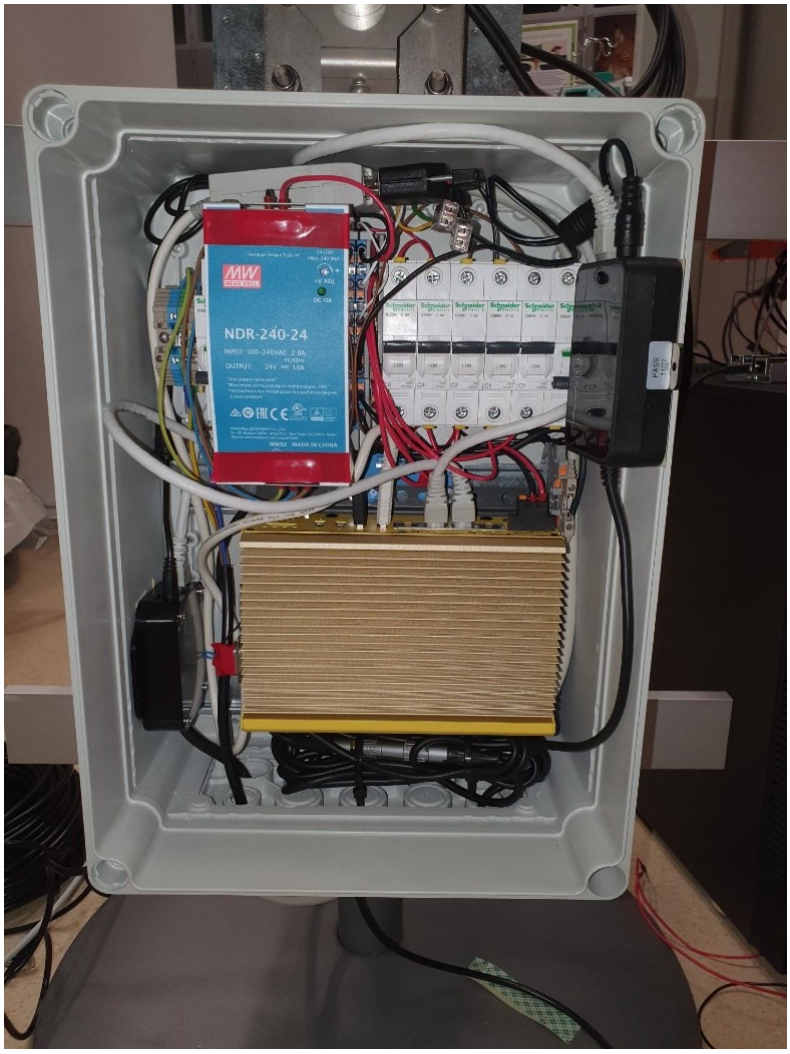
## 3.1 Rakenne ja kytkennät

Mittausaseman rakenteen tulisi olla tukeva ja talvisään kestävä. Johdotukset ja kytkennät tulisi tehdä turvallisiksi ja siisteiksi tilankäytön minimoimiseksi. Nämä olivat mittausaseman suunnittelun lähtökohdat.

Mittausaseman runkona käytettiin raskasta valurautaista jalkaa (Kuvio 4). Jalkaan kiinnitettiin U-kiinnikkeitä, joiden varaan mittausaseman muut osat olisi helppo asentaa.

Kuvio 4. Mittausaseman runko

Mittausaseman juurelle kiinnitettiin Fibox-merkkinen muovinen asennuskotelo, johon mittausaseman sähkönjakelu rakennettiin. Kuviossa 5 kotelon ylävasemmalla näkyy virtalähde, ylhäällä taustalla sulaketaulu ja yläoikealla yhden laser-tutkan tarvitsema välikappale. Alhaalla näkyy teollisuustietokoneen passiivinen jäähdytyssiili, jonka takana on itse tietokone.




Kuvio 5. Mittausaseman sähkönjakelun ja teollisuustietokoneen sisältävä kotelo

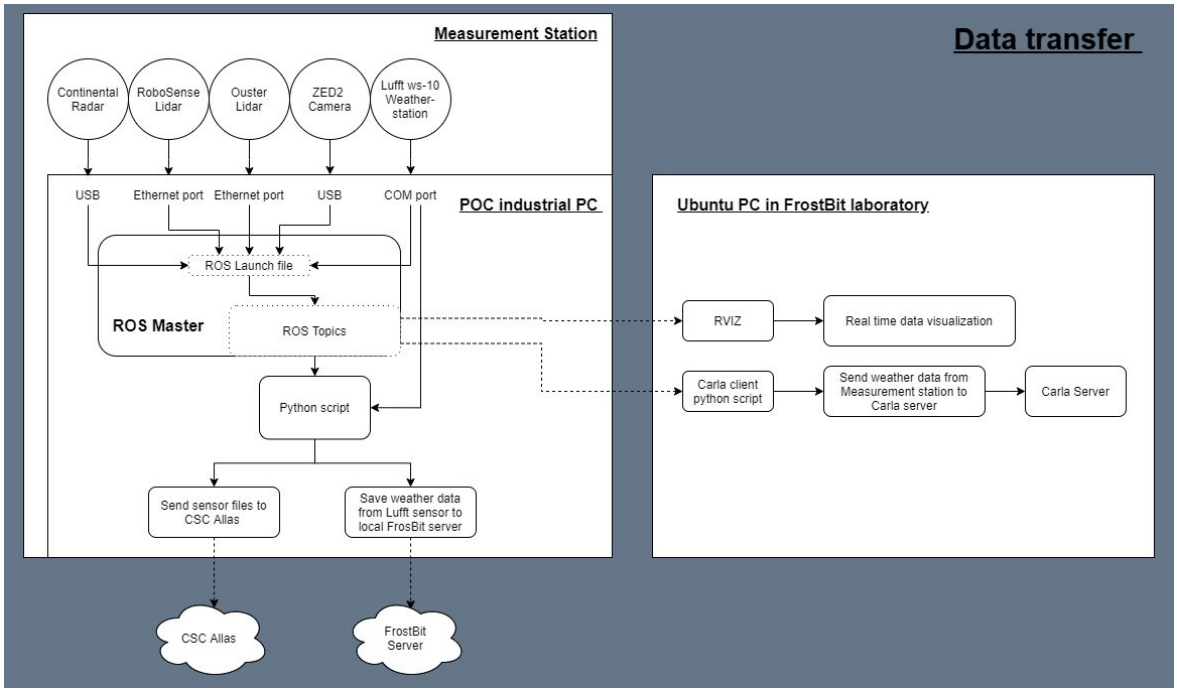
Ennen teollisuustietokoneen valitsemista mittausjärjestelyt suunniteltiin siten, että tarvittava tietotekniikka tulisi osittain aseman alla sijaitsevaan lämmitettyyn konttiin. Valittu teollisuustietokone osoittautui kuitenkin tarpeeksi pienikokoiseksi, että sen pystyi sovittamaan suoraan mittausaseman koteloon muiden laitteiden kanssa.

Kuviossa 6 näkyvät mittausaseman yläosassa lasertutkat, keskellä tutka ja alhaalla pienempi asennuskotelo, joka sisältää kameran. Kameran sisältävä asennuskotelo on lämmitetty pienikokoisella lämpövastuksella.


Kuvio 6. Valmiin mittausaseman etupuoli

Mittausaseman teollisuustietokoneeseen kytketyt sensori ja sääasema vaativat joukon erilaisia liittimiä ja adaptereita. Näille tarvittavan lisätilan vuoksi tilava asennuskotelo oli hyvä hankinta. Kun suurin osa laitteistosta ja johdoista on tiiviiden ja lämmitettyjen koteloiden sisällä, säälle altistumisen aiheuttamien ongel-

mien riski luonnollisesti pienenee. Johdotuksien haarautuminen vasta mittausasemalla, eikä sitä aiemmin tekee kokonaisuudesta siistin ja suoraviivaisen. Kuviossa 7 näkyy mittausaseman lopullinen datansiirtorakenne.

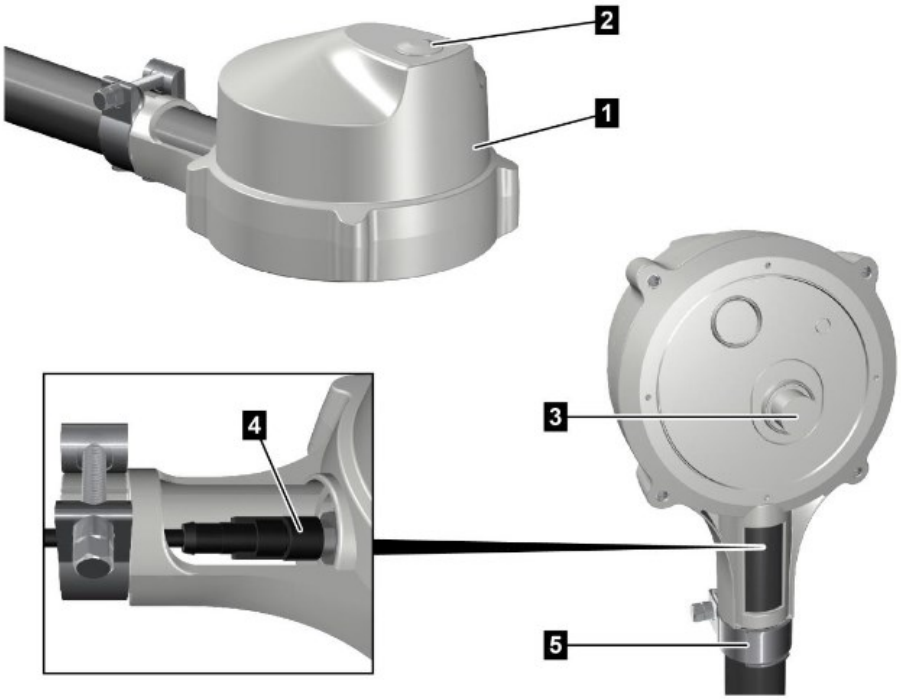



Kuvio 7. Valmiin mittausaseman datansiirtokaavio

## 3.2 Laitteisto

### 3.2.1 Sääasema

Sääasemaksi valittiin Lufft WS10 -sääasema (Kuvio 8). Tässä sääasemassa hyviä puolia ovat muun muassa sen kompakti koko, lämmitys ja se, että asemaa ei tarvitse erikseen huoltaa (Lufft 2021).




| | | | |
|---|---|---|---|
| 1 | Weather station WS10 | 4 | Connector |
| 2 | Transparent glass | 5 | Mounting bracket |
| 3 | Wind measurement dome | | |

Kuvio 8. WS10-sääaseman esittely (Lufft 2019, 5)

Sääaseman kanssa kommunikoidaan käyttäen tekstipohjaista UMB-ASCII 2.0 -protokollaa, jota käyttäen sääasemalle voi asettaa yksinkertaisia asetuksia sekä tietysti hakea säätiedot (Taulukko 1). Yksi näistä asetuksista on ”standardijoukon” (Standard Set) lähetys. Jos se on päällä, lähettää sääasema tietyin väliajoin joukon sääarvoja yhdessä viestissä, josta ne on helppo koodillisesti erotella. Kirjatut sääarvot ovat ilman lämpötila, suhteellinen ilmanpaine, suhteellinen kosteus, kastepiste, tuulen nopeus, tuulen suunta, sateen tyyppi, sateen määrä, globaali säteily, UV-indeksi ja kirkkaus. (Lufft 2019, 32.)

Taulukko 1. Sääasemalta saatavissa olevat tiedot, niiden ääriarvot ja yksiköt (Lufft 2019, 45)

| | | |
|---|---|---|
| Air Temperature | Range | -40 – 60 °C |
| | Accuracy | ±1,0 °C (+5…+60 °C), otherwise < +2,0 °C |
| Relative Humidity | Range | 0 – 100 % |
| | Accuracy | ±5 % (at 20°C and < 80 % rH) |
| Air Pressure | Range | 300 – 1100 hPa |
| | Accuracy | ±0,5 hPa (at room temperature 25 °C) |
| Wind Speed | Range | 0 – 40 m/s |
| | Accuracy | ±1 m/s or 5 %, the larger value is valid |
| Wind Direction | Range | 0 – 359° |
| | Accuracy | ±10° |
| Precipitation | Range | 0 – 100 mm/h |
| | Accuracy | 20 % under laboratory environment |
| Precipitation Type | Range | Rain, Snow, Sleet, Freezing Rain, Hail |
| Global Radiation | Range | 0 – 1500 W/m² |
| | Accuracy | 10% or ±120 W/m², larger value is valid |
| UV-Index | Range | 1 – 15 |
| Brightness | Range | 0 – 167 klx |
| | Accuracy | ±5 % of measured value |
| Twilight | Range | 0 – 500 lx |
| | Accuracy | ± 10 lx |

Sääaseman käyttöönoton jälkeen huomattiin, että sattumoisin täysin samanmerkkinen ja -mallinen sääasema on jo käytössä koulun rakennustekniikan laboratorion toimesta. Projektiryhmä sai luvan käyttää sitä, jolloin mittausaseman toimintaa voitiin varmentaa. Jos mittausasemaan kytketty sääasema lakkaisi toimimasta, voitaisiin mittaukseen käytettävän säädatan lähde vaihtaa pienellä vaivalla rakennuslaboratorion sääasemaan.

### 3.2.2 Tutka

Mittausasemaan kytkettäväksi tutkaksi valittiin Continental ARS 408 -tutka (Kuvio 9), koska sellainen oli jäänyt koululle aikaisemmasta tutkimuksesta. Saatuani tutkan aloin tutkimaan sen toimintaa ja yritin saada sitä toimimaan ensin paikallisesti omalla koneella.

Kuvio 9. ARS 408 -tutka

Koska tutka toimii CAN-väylän kautta, eikä minulla ollut CAN-liitäntää koneessani, jouduin käyttämään adapteria. Adapteri vaati kuitenkin toimiakseen valmistajan tekemiä ajureita, joiden nettisivut olivat salasanan takana. Salasanaa pitäisi kysyä sähköpostitse valmistajalta. Sain kuitenkin lopulta sovellustiedostot helpommin haltuuni kysyttyäni niitä tutkan ja adapterin aiemmalta käyttäjältä.

Valmistajan tuottamat sovellukset ja kehitystyökalut CAN-datan käsittelyä varten olivat melko yksinkertaisia ja osittain jo vanhentuneita. Kehitysympäristönä toimi vanha C++-ympäristö ja dokumentaatio kehityksestä ei ollut kovin perusteellista. Siirryin siis etsimään vaihtoehtoisia ratkaisuja tutkan käyttämiseksi.

Sain lopulta tutkan edelliseltä käyttäjältä PCAN-USB-merkkisen CAN-USB-adapterin. Tämän adapterin ajuri on sisällytettynä Linux-kerneliin, jolloin se toimii lähes kaikilla Linux-pohjaisilla järjestelmillä ilman eri asennuksia. Asensin työkoneelle

Ubuntun ja sainkin näin luettua raakadataa adapterilta. Siirryin seuraavaksi kehittämään Python-ohjelmointikielellä sovellusta raakadatan muuttamiseksi paremmin käsiteltävään ja luettavaan muotoon. Tuloksena tästä syntyi Tkinter-käyttöliittymään pohjautuva sovellus, joka pystyi lukemaan muutamia viestityyppejä tutkalta. Koko tutkan käyttämän dataformaatin parsimiseen tarkoitetun työkalun kehitykseen olisi kuitenkin uponnut liikaa aikaa, joten tämäkään ratkaisumalli ei ollut pidemmän päälle hyvä.

Löysin lopulta erään yksityishenkilön tekemän ROS-pohjaisen kirjaston tutkan käyttämistä varten. Kirjasto ei kuitenkaan antanut dataa halutussa muodossa, vaan lopulta jouduttiin tallentamaan tutkan datan pohjalta tehdyn visualisaation pisteitä. Vaikka käsitellyssä muodossa tallennetun datan myöhempi analysointi saattaisi olla monimutkaisempaa, vaihtoehtoisia ratkaisuja ei alettu enää etsimään, koska tutkimuksen pääpaino on muutenkin lasertutkien toiminnassa.

### 3.2.3 Lasertutkat

Lasertutkiksi valittiin RoboSense RS-LiDAR-16 ja Ouster OS1-32 lasertutkat (Kuvio 10). Lasertutkat ruuvattiin kulmarautoihin, jotka kiinnitettiin rinnakkain mittausaseman runkoon samalle tasolle, jotta korkeuseron vaikutus lasertutkien näkökenttään saataisiin minimoitua.


Kuvio 10. Ouster ja RoboSense lasertutkat asennettuna mittausasemaan

Lasertutkia on tehty eri määrillä keilaavia lasersäteitä. Pienellä lasersädemäärällä varustetut sensorit ovat edullisempia, kun taas suuren lasersädemäärän sensorit tuottavat tiheämpää pistepilveä. RS-LiDAR-16 sisältää nimensä mukaisesti 16 sädettä, ja OS1-32 taas 32 sädettä. Lasertutkat valittiin tarkoituksellisesti eri määrällä säteitä, jotta testissä voitaisiin arvioida myös säteiden määrän vaikutusta sensoreiden suoriutumiseen.

Molemmat käytetyistä lasertutkista ovat mekaanisia, ja niiden näkökentän laajuus on täydet 360 astetta. Mekaanisesti pyörivät lasertutkat valittiin, koska niitä olisi helppo uusiokäyttää seuraavan talvikauden liikkuvissa testeissä.

### 3.2.4 Kamera

Kameraksi mittausasemaan valittiin Stereolabs ZED 2 (Kuvio 11), koska sellainen oli laboratoriossa valmiiksi saatavilla. Koska kameraa käytetään ajoneuvoissa yleensä kohteentunnistukseen tai muihin vastaaviin toimintoihin, ei tyhjän parkkipaikan kuvaaminen stationäärisessä testissä toisi juurikaan lisäarvoa tutkimukselle. Tämän vuoksi kameraa käytettiin lähinnä säätilojen ja vuorokauden aikojen todentamiseen. ZED 2 -kamera sisältää kohteentunnistuksen lisäksi myös syvyysnäön ja muita edistyneitä ominaisuuksia (Stereolabs 2019), mutta niillekään ei löytynyt käyttöä tyhjän parkkialueen kuvaamisesta.


Kuvio 11. ZED 2 -kamera

## 3.3 Ohjelmisto

### 3.3.1 Datan vastaanottaminen sensoreilta

Sensoreiden hankkimisen jälkeen niiden toimintaa ja käyttöä opeteltiin paikallisilla työpisteillä ennen niiden asentamista mittausasemaan. Kun sensoreiden datansiirtoon käytettäviä standardeja selvitettiin, huomattiin että kaikilla valituilla sensoreilla on joko valmistajan tai kolmannen osapuolen tekemänä ROS-tuki.

ROS eli Robotic Operating System on järjestelmällinen kommunikaatiokerros robottien ohjelmointia varten. Vaikka ROS:ia sanotaankin käyttöjärjestelmäksi (engl. Operating System), se ei ole käyttöjärjestelmä sanan perinteisessä merkityksessä. Sen sijaan ROS toimii sovellustasolla, ja sillä voidaan muodostaa vertaisverkko (Peer-to-Peer Network) laitteiden välille, joilla ROS-instanssi on käynnissä. Tällöin verkon laitteita voi käyttää yhtenä kokonaisuutena eli klusterina. (Quigley ym. 2009, 1.)

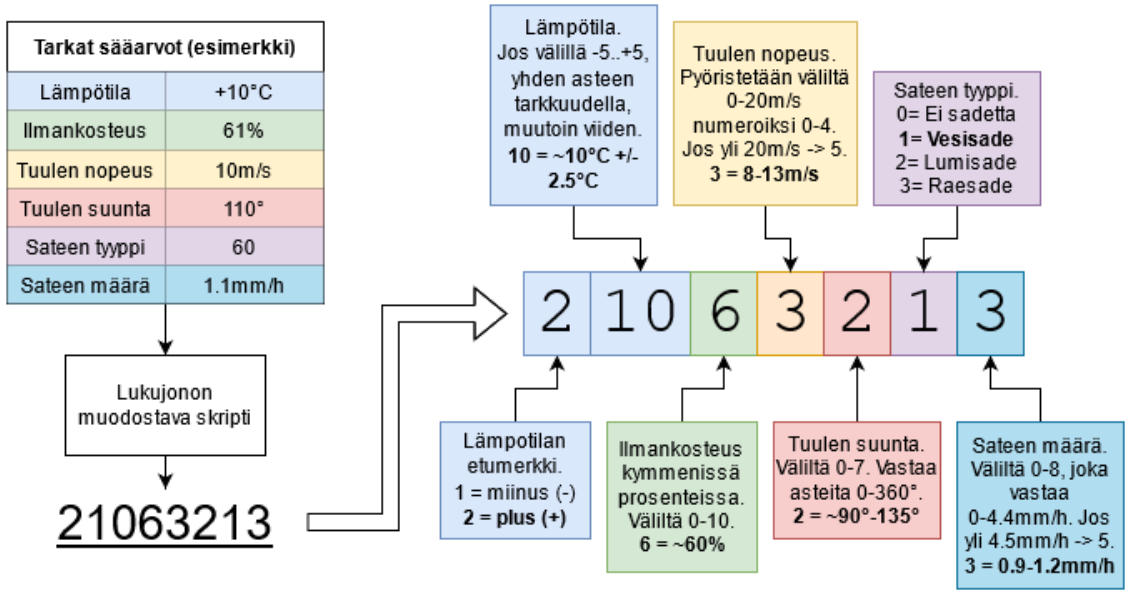
ROS-vertaisverkon logiikka perustuu julkaisija-tilaaja-malliin (Publisher-Subscriber Model). Yksittäiset laitteet (Nodes) julkaisevat viestejä (Messages) aiheisiin (Topics). Laitteet voivat myös tilata aiheita, jolloin ne alkavat vastaanottaa siihen aiheeseen julkaistuja viestejä. Jotta laitteet löytäisivät toisensa verkosta sekä verkkoon julkaistut aiheet, toimii yksi koneista isäntänä (Master), jonka tehtävänä on koordinoida vertaisverkossa tapahtuvaa toimintaa. Reaaliajassa tapahtuvan viesteihin reagoinnin lisäksi ROS mahdollistaa aiheista tulevien viestien tallentamisen BAG-muotoiseen tiedostoon myöhempää käyttöä varten. (Open Robotics 2014.)

Mittausaseman käyttötapauksessa tehtiin asemaan asennetusta teollisuus-PC-koneesta verkon isäntä. Koneessa oleva ROS-asennus tilaa sensoreiden aiheet, ja kun yksittäisen sensorin ”mittausvuoro” tulee, tallennetaan sen data BAG-tiedostoon.

### 3.3.2 Säätilan muutokseen reagointi

Jo projektin alkuvaiheessa oli jo selvää, että sensoridataa ei kannattaisi tallentaa jatkuvalla syötöllä, koska tällöin kerätty aineisto voisi potentiaalisesti kasvaa todella isoksi. Jatkuvasti kerätyllä datalla ei olisi tutkimuksen kannalta muutenkaan suurta arvoa, koska säätila muuttuu melko hitaasti. Suurien mittausmäärien ottaminen samasta säätilasta luonnollisesti parantaisi otantaa, mutta suurta hyötyä siitä tulosten kannalta tuskin olisi. Datankeräyksen prioriteetti tulisi siis olla datan monipuolisuudessa eli siinä, että dataa kerätään mahdollisimman erilaisista sääoloista. Sääasemalta saatavat sääparametrit ovat kuitenkin melko tarkkoja. Jos mittausasema tallentaisi dataa joka kerta, kun vaikka ilman lämpötila nousisi asteen kymmenesosalla, tallentuisi dataa jatkuvalla syötöllä.

Ongelman ratkaisemiseksi kehitettiin mittausaseman skriptiin sääparametreihin pohjautuva lukujono (Kuvio 12). Yksittäiset parametrit pyöristetään ennalta päätettyihin ”lokeroihin”, ja näin saatu arvo liitetään lukujonoon. Lukujono on siis yksinkertaisesti vain kokonaisluku, joka muodostetaan liittämällä yhteen pyöristettyjä sääparametrejä. Kun säätila muuttuu tarpeeksi, lukujononkin arvo muuttuu. Tällöin säätilojen vertailu koodissa on helppoa, kun monien sääparametrien sijaan voi vertailla vain kahta kokonaislukua.




Kuvio 12. Lukujonon muodostaminen

Mittausasema lukee ja tallentaa säätilan 15 minuutin välein. Tämän jälkeen mittausasema vertaa säätilan lukujonoa edellisen säätilan lukujonoon. Jos lukujono on eri, asema aloittaa tallentamaan vuorotellen jokaiselta sensorilta tulevaa dataa noin kymmenen sekunnin ajan sensoria kohden. Muussa tapauksessa vain säätilan tiedot tallennetaan.

### 3.3.3 Datan lähetys

Datan lähetys ja tallennus tapahtuvat molemmat mittauslaitteen tietokoneessa. Niihin käytetyt skriptit päätettiin kuitenkin erottaa toisistaan koodin selkeyttämiseksi ja siksi, että eri henkilöt voisivat koodata niitä helposti yhtäaikaisesti. Luvuissa 3.3.1 ja 3.3.2 esitetyt toiminnot tapahtuvat siis mittausaseman ”pääskriptissä”, kun datan lähetys taas on erotettuna omaan koodikirjastoonsa. Pääskripti kutsuu lähetyskirjastoa tarvittaessa eri vaiheissa prosessia, kun dataa valmistellaan lähetettäväksi ja kun se lopulta lähetetään.

Datan lähetyksen kirjasto on toteutettu olio-ohjelmoinnin tyylillä. Säädata, sensoridata ja lähetin ovat omia luokkiansa. Pääskripti luo aluksi uuden olion eli instanssin lähetinluokasta. Lähetinluokka tunnistautuu aluksi objektintallennuspalveluun käyttäen erillisessä tiedostossa määriteltyjä tunnuksia, koska tunnuksien kirjoittaminen suoraan koodiin on turvallisuusriski (Pedro 2017). Kun pääskripti on tallentanut joko sää- tai sensoridataa, luo se datan pohjalta vastaavan olion ja antaa sen lähettimelle. Sensoridataolion luominen vaatii säädataolion parametrinä. Tällöin viimeisin säädata on aina ”kytkettynä” lähetettävään sensoridataan. Lisäksi lähetin odottaa, että säädata on lähetetty onnistuneesti, koska sen vastauksen mukana rajapinta ilmoittaa säädatan ID:n tietokannassa. Säätilan ID on pakko lähettää sensoridatan mukana, jotta sensoridata on mahdollista yhdistää säätilaan tietokannassa.

Vaikka sensorimittauksen tiedot sekä itse sensoridatan sisältävän tiedoston polku on sisällytettynä samaan sensoridataolioon, osaa lähetin erotella tiedot lähetysvaiheessa ja varmistaa niiden lähetyksen oikeisiin kohteisiin (ks. luvut 4.1 ja 4.2).

Lähettimeen on lisäksi rakennettu jonotusjärjestelmä sen varalle, että datan lähetys epäonnistuu esimerkiksi nettiyhteyden katkeamisen vuoksi. Tällöin lähetin lisää dataolion jonotuslistalle. Seuraavalla kerralla kun dataa lähetetään, yritetään lähettää myös listalla olevat datat. Listan datatiedostoille on määritetty varmuuden vuoksi myös kokorajoitus. Jos listan koko yrittää raja-arvon, joka on oletuksena kymmenen gigatavua, aletaan listan olioita ja niihin liitettyjä datatiedostoja poistamaan vanhimmasta päästä.

## 3.4 Mittausjärjestelyt

Mittausjärjestelyjä suunniteltaessa tultiin siihen tulokseen, että sensoreilta saatu data olisi sitä vertailukelpoisempaa, mitä vähemmän ympäristö muuttuisi. Ympäristöksi haluttiin siis iso, tyhjä alue. Alue tulisi olla myös mahdollisimman lähellä laboratoriota; mielellään saman rakennuksen piirissä. Koulun kattoa mietittiin yhtenä vaihtoehtona, mutta sen todettiin olevan epäkäytännöllistä muun muassa työturvallisuuteen liittyvien ylimääräisten asennusten takia. Lopulta alueeksi valikoitui koulun takana sijaitseva parkkialue. Parkkialueelle saatiin projektin käyttöön kaksi konttia, joista toisessa on lisäksi lämmitys. Kun mittausaseman asentaa kontin päälle, olisi sen näkökenttä riittävän suuri, koska tällöin sensorit näkisivät parkkipaikalla olevien autojen ylitse (Kuvio 13).


Kuvio 13. Konttien päälle asennettu mittausasema sekä vasemmalla taustalla valotolpassa yksi heijastinlevyistä

Mittausalueelle asennettiin valotolppiin tasavälein heijastinlevyjä (Kuvio 13), joiden tarkoituksena olisi toimia optimaalisina referenssipisteinä sensoreille. Jos jokin sensori esimerkiksi menettäisi näkyvyyden kauimmaiseen levyyn säätilan muuttuessa, olisi siitä helppoa päätellä karkeasti käytetyn sensorin kantama senhetkisessä säätilassa. Heijastinlevyt olivat alun perin tehty alumiinista, mutta ne eivät jostain selvittämättömästä syystä heijastaneetkaan lasertutkien säteitä kuin hyvin heikosti. Kun levyjen materiaali vaihdettiin vaneriin, sensorit havaitsivat ne huomattavasti paremmin.

Lopullinen mittausasema (Kuvio 14) on toimiva kokonaisuus, joka on suoriutunut tehtävästään mallikkaasti. Etenkin sen eristetyt osiot ovat kestäneet säätä hyvin. Mittausasema on ollut yhtäjaksoisesti ulkona toiminnassa lähes koko talvikauden 2020 – 2021. Asema jäänee keräämään sensoridataa toistaiseksi, kunnes sensorit tarvitaan muuhun käyttöön tai projektiryhmä toteaa, että dataa on kerätty tarpeeksi.


Kuvio 14. Valmis mittausasema

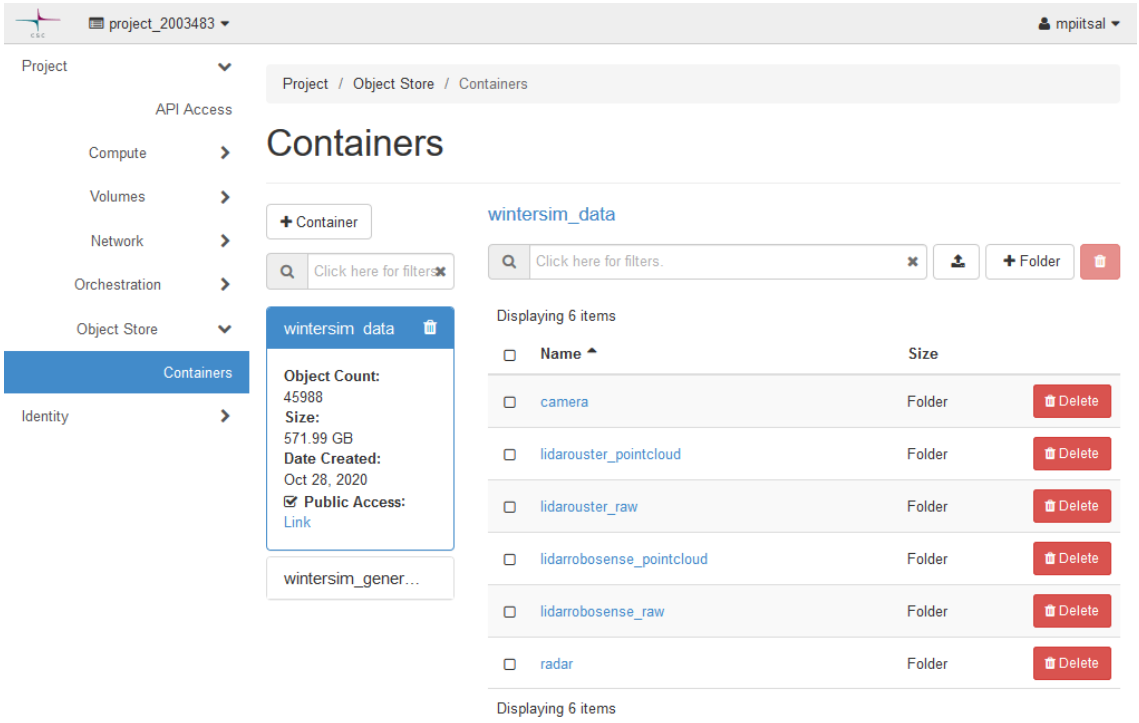
# 4 DATAN TALLENNUS

## 4.1 Sensoridatatiedostojen verkkotallennus

Koska sensoridatatiedostot voivat sensorista riippuen kasvaa melko isoiksi, niiden tallentaminen esimerkiksi suoraan tietokantaan tai mittausasemalle koko talvikauden ajalla ei tuntunut mielekkäältä ratkaisulta. Kerättyjen datatiedostojen pitäminen ulkoisella palvelimella helpottaisi myös tiedostojen varmuuskopiointia ja järjestelmällistä hakemista.

Projektiryhmä harkitsi erilaisia tiedostopalvelimia ja tallennuspalveluita, kuten omaa sisäistä tiedostopalvelinta tai Amazonin AWS:ää. Lopulta kuitenkin päädyttiin käyttämään CSC:n eli tieteen tietotekniikan keskuksen Allas-palvelua, koska se on ilmainen tutkimuskäyttöön. Tilaa on oletuksena kymmenen teratavua, ja sitä voi tarvittaessa pyytää lisää. (CSC 2021.)

Allas-palveluun on saatavilla monta eri työkalua datan käsittelyyn. Tiedostojen tallentamiseen ja lataamiseen käytettävien työkalujen kehittämiseen käytettiin projektissa Allas-palvelun Python-kirjastoa. Tarvittaessa käytettiin myös Allas-palvelun Pouta-nettikäyttöliittymää tallennuksen seuraamiseen ja muuhun manuaaliseen ylläpitoon (Kuvio 15).




Kuvio 15. Kuvankaappaus Allas-palvelun Pouta-nettikäyttöliittymästä

Allas-palveluun tallennettuja tiedostoja ei varmuuskopioida (CSC 2021). Tämän vuoksi toteutin Pythonilla yksinkertaisen komentorivisovelluksen, joka lataa kaikki tiedostot paikalliseen kansioon. Sovellus näyttää latauksen edistymisen latauspalkkina. Lisäksi sovelluksen voi pysäyttää ja latausta voi jatkaa myöhemmin. Tällöin sovellus ohittaa jo ladatut tiedostot. Suoritin maaliskuussa 2021 täyden varmuuskopion lataamisen. Tiedostojen yhteiskoko oli tuolloin noin 450 gigatavua, ja niiden lataamisessa kesti noin kolme tuntia.

## 4.2 Tietokanta datan jäsentämiseen

Sensoridatan mittaustapahtumien ja säätiedon kirjaamista varten tarvittiin tallennuspaikka. Jo alusta asti oli selvää, että jonkinlainen tietokantajärjestelmä olisi paras tähän käyttötarkoitukseen, koska tietoa kirjattaisiin jatkuvalla syötöllä ja sen rakenne olisi aina sama. Tietokannan malliksi harkittiin joko perinteistä relaatiotietokantaa, kuten MySQL ja PostgreSQL tai niin sanottua NoSQL-tietokantaa, kuten MongoDB:tä.

Relaatiotietokannat ovat olleet jo vuosikymmeniä yleisessä käytössä, minkä vuoksi ne ovat pitkälle kehitettyjä ja toimivaksi todettuja. Relaatiotietokannat perustuvat tietojen tallentamiseen taulukoihin, joiden väliin voi lisätä yhteyksiä (Relations). Taulukoiden rakennemalli on lähtökohtaisesti muuttumaton. Datan lisäämiseen, poistamiseen, päivittämiseen ja hakemiseen käytetään SQL-kieltä (Structured Query Language). Relaatiotietokannat tukevat niin sanottua ACID-periaatetta (Atomicity, Consistency, Isolation, Durability), jonka tarkoituksena on varmistaa datan oikeellisuus sähkökatkoista tai muista virheistä aiheutuvista ongelmista huolimatta. (Microsoft 2021.)

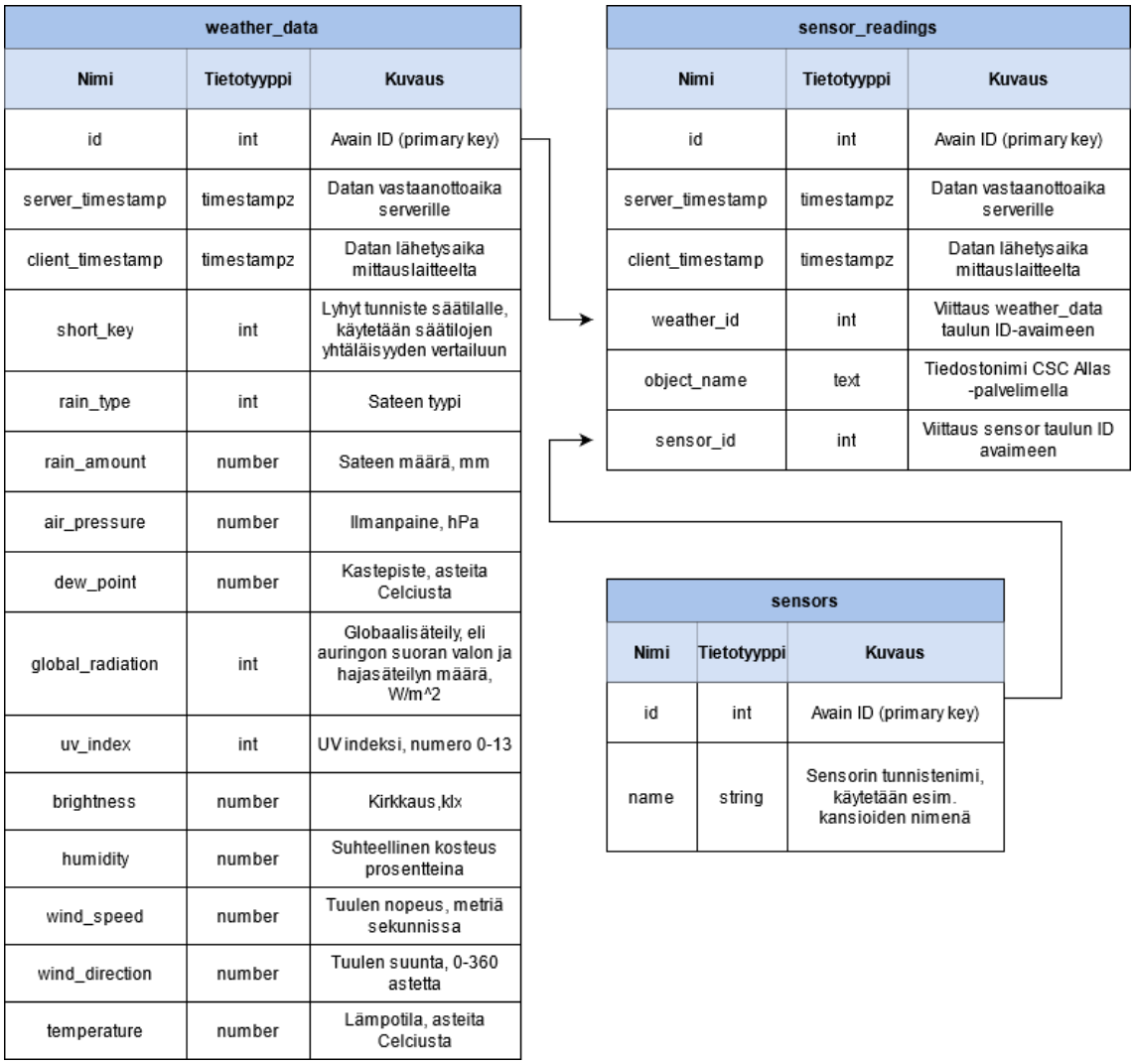
NoSQL-pohjaiset tietokannat on kehitetty vastaamaan relaatiotietokannan käytöstä johtuviin haasteisiin. Relaatiotietokannan jäykkä ja muuttumaton datan rakenne sekä tehottomuus suuria määriä jäsentämätöntä dataa käsiteltäessä aiheuttavat ongelmia esimerkiksi suurien sosiaalisten medioiden käyttötapauksissa. NoSQL-tietokannat sen sijaan tallentavat datan yleensä avainarvopareilla (Key-Value Pairs) tai JSON-dokumentteina. Tiedon käsittely on tällöin vähemmän turvattua, mutta tehokkaampaa ja skaalautuvampaa, jolloin se käy hyvin tapauksiin, jossa käsitellään suuria määriä vähemmän kriittistä dataa. (Microsoft 2021.)

Vaikka NoSQL-tietokanta voi joskus olla relaatiotietokantaa tehokkaampi, erot suorituskyvyssä alkavat näkyä huomattavasti vasta kun käyttäjämäärä nousee kymmeniintuhansiin. Projektin käyttötapauksessa tietokannan mallilla ei siis suorituskyvyn kannalta ole todennäköisesti juurikaan merkitystä.

Päädyin lopulta käyttämään PostgreSQL-relaatiotietokantamoottoria, koska se on jo FrostBit-laboratorion palvelimilla asennettuna ja muiden projektien toimesta käytössä. Tiedon rakenne on projektissa myöskin melko tiukasti jäsennettyä ja muuttumatonta, mikä sopii relaatiotietokannalle.

Tietokannan lopullinen rakenne koostuu kolmesta taulukosta (Kuvio 16). ”Sensors”-taulukossa on listattuna yksinkertaisesti sensoreiden nimet, jotta sensoreita voi tarvittaessa lisätä ja poistaa joustavasti. ”Weather_data”-taulukossa on tallennettuna yksittäisen hetken säätilaa määrittävät parametrit. ”Sensors_readings”-

taulukko sisältää tiedon yhdellä sensorilla suoritetusta mittauksesta. Mittauksen datan sisältävä tiedosto on tallennettu ”object_name”-kentässä olevalla nimellä erilliselle palvelimelle (ks. luku 4.1). Tietueeseen tallentuvat myös aikaleimat sekä viitteet säätilaan ja sensoriin, millä mittaus suoritettiin.



**weather_data**

| Nimi | Tietotyyppi | Kuvaus |
|---|---|---|
| id | int | Avain ID (primary key) |
| server_timestamp | timestampz | Datan vastaanottoaika serverille |
| client_timestamp | timestampz | Datan lähetysaika mittauslaitteelta |
| short_key | int | Lyhyt tunniste säätilalle, käytetään säätilojen yhtäläisyyden vertailuun |
| rain_type | int | Sateen tyypi |
| rain_amount | number | Sateen määrä, mm |
| air_pressure | number | Ilmanpaine, hPa |
| dew_point | number | Kastepiste, asteita Celciusta |
| global_radiation | int | Globaalisäteily, eli auringon suoran valon ja hajasäteilyn määrä, W/m^2 |
| uv_index | int | UV indeksi, numero 0-13 |
| brightness | number | Kirkkaus,klx |
| humidity | number | Suhteellinen kosteus prosentteina |
| wind_speed | number | Tuulen nopeus, metriä sekunnissa |
| wind_direction | number | Tuulen suunta, 0-360 astetta |
| temperature | number | Lämpotila, asteita Celciusta |

**sensor_readings**

| Nimi | Tietotyyppi | Kuvaus |
|---|---|---|
| id | int | Avain ID (primary key) |
| server_timestamp | timestampz | Datan vastaanottoaika serverille |
| client_timestamp | timestampz | Datan lähetysaika mittauslaitteelta |
| weather_id | int | Viittaus weather_data taulun ID-avaimeen |
| object_name | text | Tiedostonimi CSC Allas -palvelimella |
| sensor_id | int | Viittaus sensor taulun ID avaimeen |

**sensors**

| Nimi | Tietotyyppi | Kuvaus |
|---|---|---|
| id | int | Avain ID (primary key) |
| name | string | Sensorin tunnistenimi, käytetään esim. kansioiden nimenä |

Kuvio 16. Tietokannan taulukkorakenne

## 4.3 Tietokannan rajapinta

Tietokannan käyttöä varten tarvittiin rajapinta eli API, jonka kautta tiedon kulku toimisi tehokkaasti, loogisesti ja turvallisesti. Yleinen tapa palvelinpuolen sovelluskokonaisuuden ja näin ollen myös sen rajapinnan kehitykseen on käyttäen jotain valmista kehystä (Framework). Tällaisia kehyksiä ovat esimerkiksi Python-pohjainen Django, PHP-pohjaiset kehykset kuten Laravel ja CakepPHP, tai JavaScript-pohjaiset ratkaisut kuten Express.js. (Clark 2020.)

Sen sijaan, että olisin itse lähtenyt kehittämään rajapintaa käyttäen jotain palvelinpuolen ohjelmointikehystä, valitsin valmiin rajapintaratkaisun nimeltä PostgREST. Kuten sen nimestä voi päätellä, on kyseessä PostgreSQL-tietokannan kanssa käytettäväksi tehty, REST-arkkitehtuurityyliä käyttävä valmis rajapintaratkaisu.

REST tulee sanoista Representational State Transfer. Sillä tarkoitetaan arkkitehtuurista tyyliä, joka on suunniteltu hajautetuille hypertekstijärjestelmille, joihin myös rajapinta käsitteenä liittyy. Jotta järjestelmää voisi kutsua REST-tyyliseksi, täytyy sen täyttää kuusi ohjaavaa periaatetta: asiakas-palvelin-arkkitehtuuri, tilattomuus, välimuistin hyödyntäminen, yhdenmukainen rajapinta, kerrostettu järjestelmä ja valinnaisesti koodin lataaminen tarvittaessa. (RedHat 2021.)

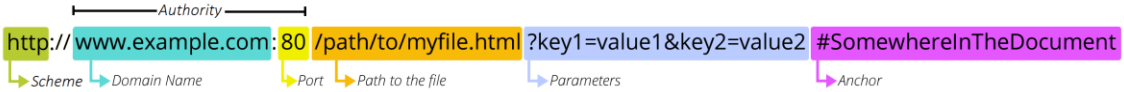
REST-tyylissä keskeisenä käsitteenä on resurssi, joka tarkoittaa mitä tahansa nimettyä tietoa, kuten kuvaa, palvelua tai vaikkapa henkilötietoa. Resursseilla on tunniste, joilla sen voi löytää. Kun resurssin tilaa halutaan muuttaa, pyynnön mukana esitetään metodi. Esimerkiksi kun käyttäjä poistaa netin kuvapalveluun lataamansa kuvan painamalla verkkosivulla olevaa poistonappia, selain lähettää pyynnön, jossa tunnisteena on kuvan osoite ja metodina http-protokollan mukaisesti DELETE. Vaikka REST-tyyliä käytetäänkin yleensä juuri http-protokollan kautta, mikä voi aiheuttaa sekaannusta näiden kahden määritteen välillä, ei REST-tyylin alkuperäisessä määrittelyssä nimetä mitään tiettyä protokollaa käytettäväksi. (REST API Tutorial 2020.)

PostgREST-rajapinnan käyttö osoittautui melko helposti, etenkin jos tietokantoihin käytettävän SQL-kielen periaatteet ovat hallussa. Testattuani ensin rajapintaa paikallisesti omalla koneellani, tapahtui asennus laboratorion palvelimelle tietokannan rinnalle käyttäen valmista Docker-imagea.

PostgREST toimii itsenäisenä palvelinsovelluksena. Ennen sovelluksen käynnistämistä on kuitenkin määritettävä konfiguraatiotiedostoon tietokannan osoite, käyttäjätunnukset ja JWT-salausavain. JWT eli JSON Web Token on JSON-ob-

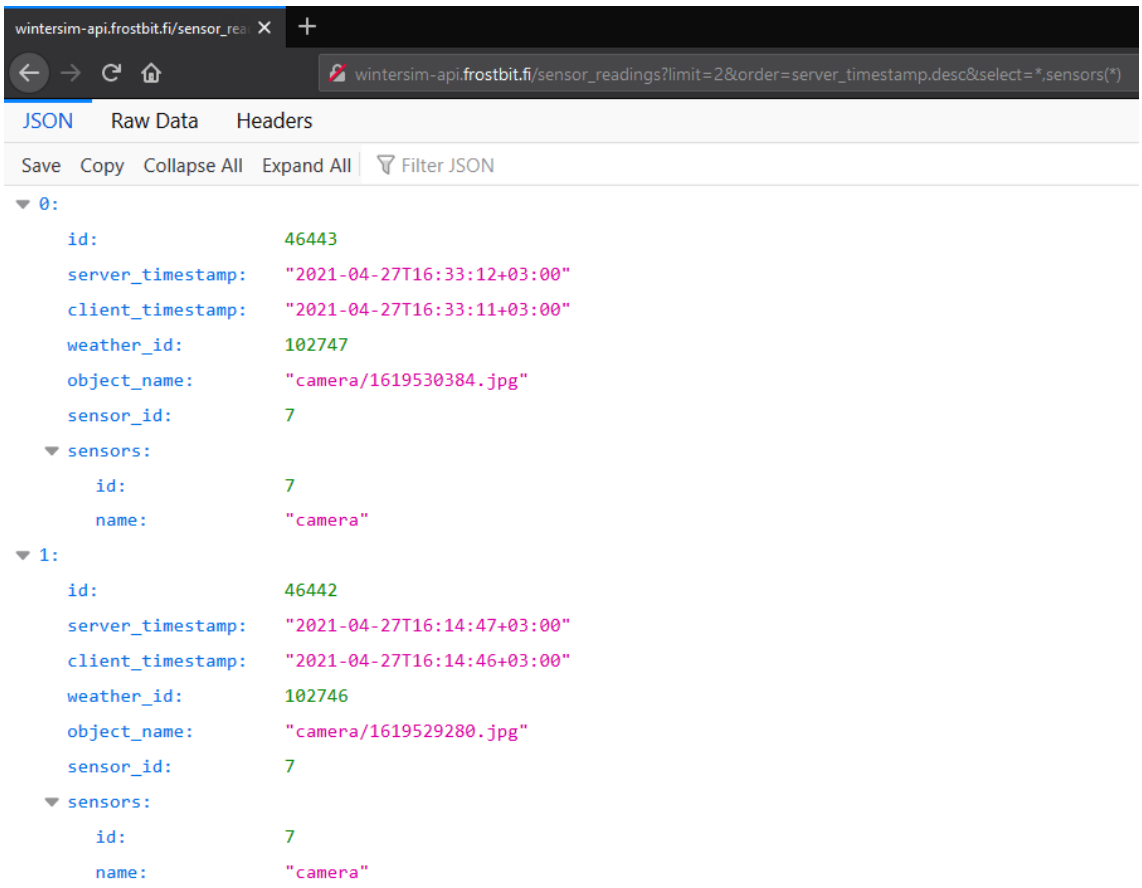
jekti, joka salataan käyttäen salausavainta. Kun rajapintaa käyttävä sovellus tarvitsee erityisoikeuksia, kuten oikeuden lisätä tietoa, lähettää se pyynnön mukana kyseisen JWT-objektin. PostgREST varmistaa tokenin aitouden konfiguraatiossa olevalla salausavaimella, minkä jälkeen se vaihtaa konfiguraatiossa asetetusta vieraskäyttäjästä tietokannan pääkäyttäjään, jolla on oikeus tehdä halutut muutokset. (Nelson & Chavez 2017a.)

Rajapinnan itsensä käyttäminen tapahtuu täysin URL-osoitepohjaisesti. Verkkoa selatessakin käytettävät URL-osoitteet koostuvat useasta osasta (Kuvio 17). URL-osoitteen domain-kohtaan tulee luonnollisesti rajapintasovelluksen sisältämän palvelimen osoite. Tiedostopolun kohdalle tulee PostgREST-rajapintaa käyttäessä halutun tietokantataulun nimi. Parametrit-osioon voi halutessaan lisätä SQL-tyylisiä kyselyitä tai suodattimia &-merkillä eroteltuna. (Nelson & Chavez 2017b.)




Kuvio 17. URL-osoitteen eri osat (MDN Web Docs 2021b)

Kuviossa 18 on tehty esimerkkinä pyyntö, jossa tiedostopolkuna on ”sensor_readings”-taulukko. Parametrit osiossa on rajoitettu tulosten määrä kahteen, lajiteltu tulokset palvelinajan mukaan laskevasti ja liitetty tulosten mukaan tarkat tiedot sensoreista pelkän ID:n sijasta.



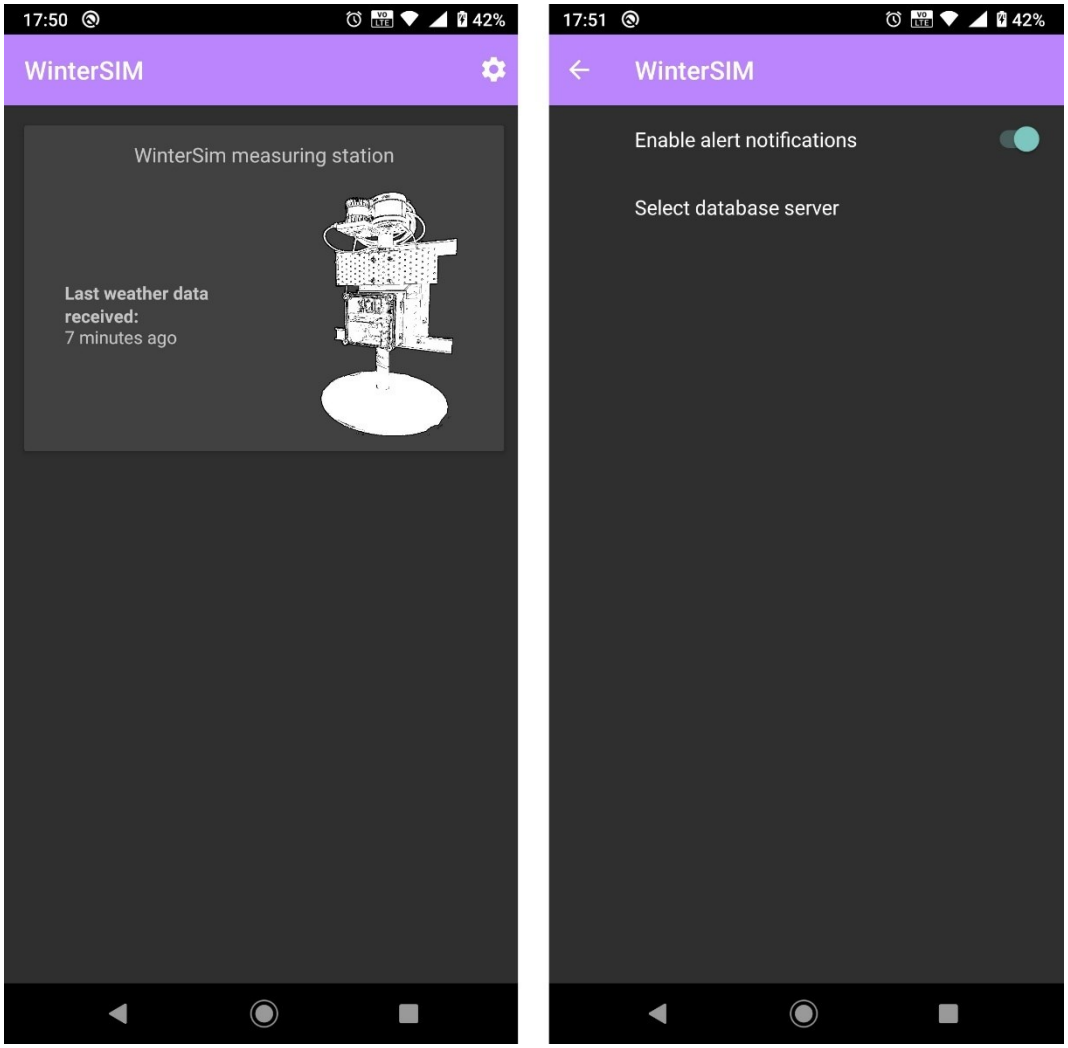

Kuvio 18. Tiedot kahdesta viimeisimmästä sensorimittauksesta selaimen kautta pyydettynä

Valmiin rajapintaratkaisun käyttäminen oli jälkeenpäin ajateltuna projektin tarpeisiin erittäin hyvin sopiva ratkaisu, joka säästi paljon kehitykseen käytettävää aikaa. PostgREST-rajapintaratkaisun rajoitteet, jotka liittyvät lähinnä monimutkaisten kyselyjen tekemiseen, eivät projektin käyttötapauksessa muodostuneet missään vaiheessa ongelmaksi, sillä tietokannan rakenne on suhteellisen yksinkertainen. Rajapinnan REST-tyylinen, looginen, minimaalinen ja avoin malli tekivät rajapintaa käyttävien työkalujen kehittämisestä helppoa.

# 5 SEURANTA JA JÄLKIKÄSITTELY

## 5.1 Mobiilisovellus datankeräyksen seurantaan

Yksi huolenaihe, joka nousi esiin projektipalavereissa, oli se, miten voitaisiin varmistaa, että datan keruu ei keskeytyisi pitkäksi aikaa projektiryhmän tietämättä. Tämä voisi aiheutua esimerkiksi sähkökatkoksen tai ohjelmointivirheen seurauksena. Yhtenä ratkaisuna ongelmaan projektiryhmä päätti, että minä koodaisin Android-sovelluksen (Kuvio 19) joka ilmoittaisi, jos tiedonkulku keskeytyisi.

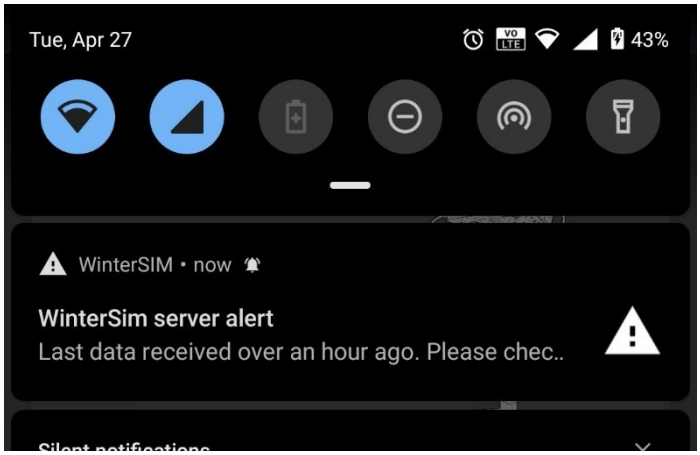



Kuvio 19. Sovelluksen päänäkymä ja asetukset

Valmis sovellus on melko yksinkertainen. Sovelluksen avauduttua päänäkymässä on tieto siitä, milloin viimeksi tietokanta on vastaanottanut tietoa mittaus-

asemalta. Oikealta yläkulmasta sovelluksen käyttäjä pääsee sovelluksen asetuksiin, josta voi kytkeä ilmoitukset päälle tai pois päältä sekä vaihtaa rajapintaa testirajapinnan ja oikean rajapinnan välillä (Kuvio 19).

Sovelluksen päätoiminto on kuviossa 20 näkyvä järjestelmätason ilmoituksen antaminen, jos dataa ei ole vastaanotettu tietokantaan viimeisen tunnin aikana. Koska tarkistus on ajastettu käyttäen Androidin WorkManageria, se suoritetaan, vaikka itse sovellus olisi kiinni (Google Developers 2021).




Kuvio 20. Sovelluksen antama järjestelmätason ilmoitus

## 5.2 Verkkosivu datan keräyksen esittelyyn

Jonkin aikaa sen jälkeen, kun datan kerääminen oli aloitettu, keksittiin, että olisi mukava seurata mittausaseman tilaa ja datan keräyksen etenemistä jonkinlaisesta käyttöliittymästä, esimerkiksi laboratoriossa sijaitsevasta TV-ruudusta. Jo olemassa olevien, rajapinnan ja datan hakemisen toteutusten vuoksi selkeästi helpoin tapa tällaisen käyttöliittymän toteuttamiseen oli verkkopohjainen sovellus. Verkkosivulla oleva sovellus olisi myös helppo avata ja esittää millä tahansa nettiin kytketyllä laitteella.

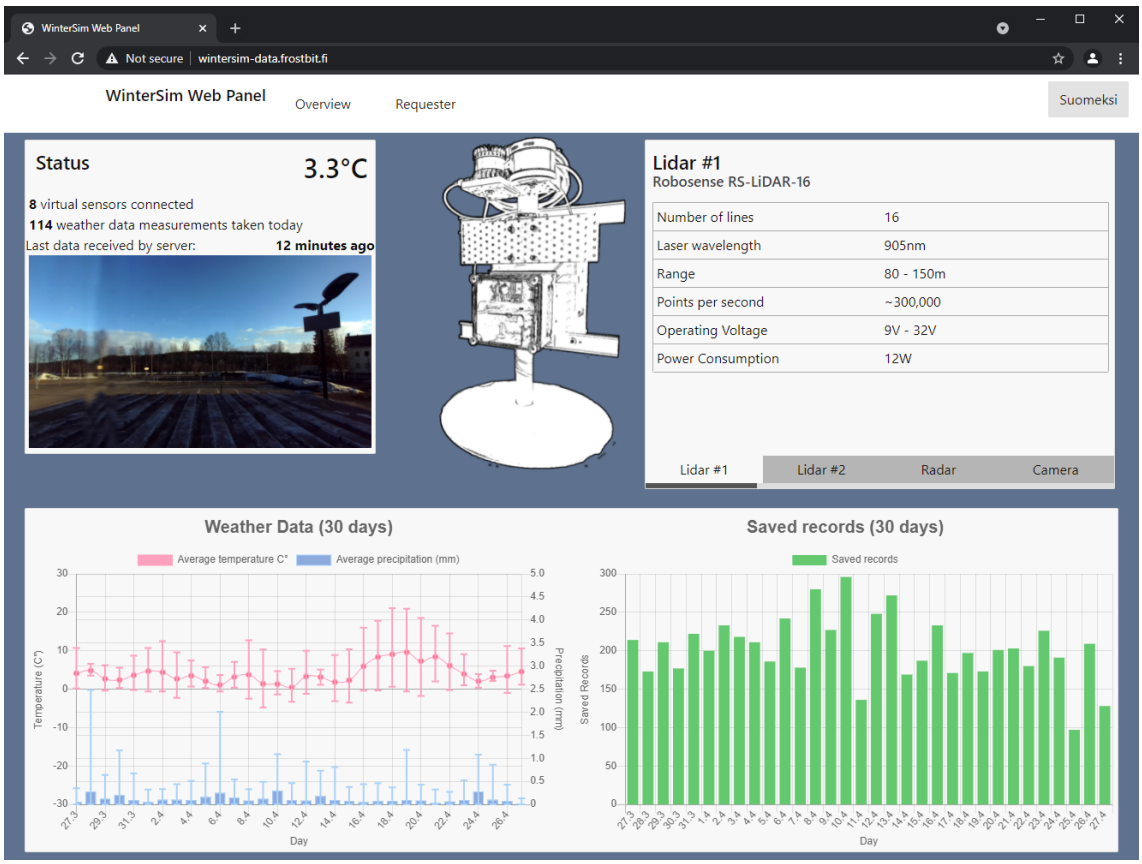
Lähdin toteuttamaan verkkosovellusta käyttäen perinteisiä HTML-, CSS- ja JavaScript-teknologioita. Sivuston eri dynaamisten elementtien rakenteen luomiseen käytin Handlebars-nimistä kirjastoa. Sivuston tilanhallintaan ja sisällön reaktiiviseen päivitykseen hyödynsin MobX-nimistä kirjastoa. Lisäksi käytin

JQuery-kirjastoa koodin yksinkertaistamiseksi. Sivuston ulkoasun toteutin käyttäen mini.css-nimistä tyylikirjastoa, jonka päälle tein myös omia tyylejä käyttäen SCSS-kieltä.

SCSS kieli on CSS-kielen jatke, joka käännetään kehitystyökaluilla normaaliksi CSS-tiedostoksi ennen käyttöä. SCSS-kieli sisältää monia hyödyllisiä lisäominaisuuksia CSS:ään nähden, kuten sisäkkäiset tyylimääritykset ja muuttujat. (Wickramarachchi 2021.)

Edellä mainitut teknologiat ovat hyviä etenkin yksinkertaisten, yhden sivun verkkosovellusten luomiseen. Sivustoon lisättiin kuitenkin eteenpäin mennessä jatkuvasti uusia ominaisuuksia, kuten luvussa 5.3 kuvattu hakutyökalu. Tämän vuoksi olisi jälkeenpäin ajateltuna ollut järkevämpää toteuttaa sivusto käyttäen pohjana jotain tiiviimpää sovelluskehystä (Framework). Suosituimpia verkkosovellusten rakentamiseen käytettyjä kehyksiä JQuery:n lisäksi ovat esimerkiksi React, Vue.js ja Angular (Stack Overflow 2020). Isompiin projekteihin suunnitellun kehyksen käyttö olisi aluksi lisännyt hieman työtaakkaa, mutta se olisi voinut pidemmän päälle pitää sekä rakenteen että koodin selkeämpänä sivuston kasvaessa.

Sivustosta oli tarkoitus myös hioa jossain vaiheessa visuaalisesti näyttävämpi, mutta sitä ei tapahtunut, koska projektista löytyi jatkuvasti tärkeämpiäkin tehtäviä. Kuviossa 21 näkyvä lopputulos on kuitenkin melko selkeä, informatiivinen ja interaktiivinen kokonaisuus, joka näin toteuttaa tarpeeksi hyvin alkuperäistä tarkoitustaan.




Kuvio 21. Kuvankaappaus verkkoseurantapaneelista

Sivulla näkyy vasemmassa yläkulmassa olevassa laatikossa nykyinen lämpötila, kytkettyjen sensoridatalähteiden määrä, päivän aikana mitattujen sääarvojen lukumäärä, datan viimeisin vastaanottamisaika ja viimeisin kameran ottama kuva. Oikealla ylhäällä on esiteltynä sensoreiden ominaisuuksia neljällä välilehdellä. Välilehteä vaihdetaan automaattisesti seuraavaan tietyn ajan kuluttua, jotta tietoa voisi tarkastella näytöltä ilman interaktiota. Alhaalla näkyvät ChartJS-kirjastolla toteutetut kaaviot. Vasemmassa kaaviossa on viimeisen 30 päivän lämpötilan ja sademäärän keskiarvot. Keskiarvon kohdalla oleva pystysuora palkki merkitsee arvon vaihteluväliä kyseisenä päivänä. Oikealla näkyvät lukumäärät tallennetuista sää- ja sensoriarvoista päiväkohtaisesti.

## 5.3 Hakutyökalu datan lataamiseen

Kun keväällä 2021 dataa oli kerätty kuluneen talven ajalta jo monen kuukauden verran, sen jälkikäsittely ja analysointi alkoivat tulemaan ajankohtaiseksi. Datan jälkikäsittelyn prosessin voi ajatella koostuvan kolmesta vaiheesta: tiedostojen

suodattamisesta hakuparametrien perusteella, niiden lataamisesta ja niiden purkamisesta analyysiä varten. Ideaalia olisi ollut se, jos olisin pystynyt toteuttamaan yhden työkalun, joka pystyisi hoitamaan kaikki nämä tehtävät. Harkitsin joko täysin selainpohjaisen tai komentorivipohjaisen työkalun tekemistä.

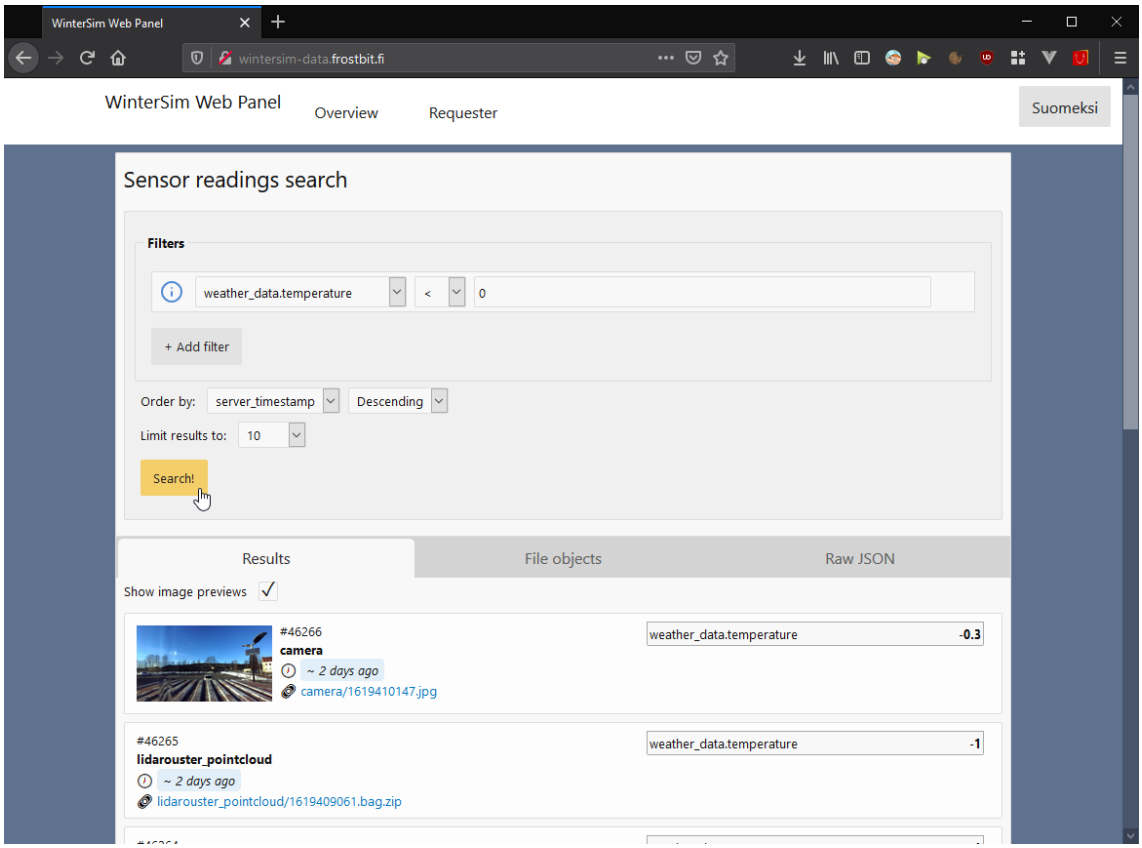
Tutkittuani asiaa tajusin kuitenkin, että tietojen haun ja suodatuksen tekeminen komentorivipohjaiseksi olisi verrattain työlästä, siinä missä selainpohjaiseen ratkaisuun löytyy valmiita lomake-elementtejä. Lisäksi selainpohjainen ratkaisu sisältäisi graafisen käyttöliittymän, mikä olisi tyylikäs, mutta myös käyttäjälle intuitiivinen verrattuna komentorivisovellukseen.

Tätä toteuttaessani huomasin kuitenkin myös, että verkkoselaimen turvallisuuskäytäntö ei salli tiedostojen lataamista taustalla, jolloin täysin selainpohjaisen ratkaisun tekemisen ei olisi mahdollista. Kyseinen turvallisuuskäytäntö on niin sanottu ”saman lähteen käytäntö” (Same-Origin Policy), eli selain ei välttämättä salli käytettävän eri lähteestä saatua resurssia, kuin missä sivu itse on. Esimerkiksi, jos käyttäjä menee selaimella osoitteeseen http://foo.example ja sivulla oleva JavaScript-koodi lähettää pyynnön osoitteeseen http://other.example, lähettää selain ensin esipyynnön (Pre-flight Request). Kun http://other.example -osoitteessa sijaitseva palvelin vastaa esipyyntöön, voi se laittaa vastauksen mukaan otsikkotiedon (Header), joka sallii resurssin käytön. Jos otsikkotietoa ei ole, selain estää varsinaisen pyynnön. Tällä pyritään estämään mahdollista haitallista verkkosivua lähettämästä pyyntöjä taustalla jollekin toiselle sivustolle (W3 2021), mille käyttäjä voisi olla kirjautuneena selaimellaan, kuten vaikkapa yhteisöpalveluun tai verkkopankkiin. (MDN Web Docs 2021a.)

CSC Allas -palvelun tiedostojen lataamiseen käytettävä palvelin ei ilmeisesti sisällytä tarvittavaa otsikkoa. Näin ollen siellä olevia tiedostoja ei voi ladata tai käsitellä selainpohjaisesti taustalla, ainakaan sillä tavalla millä minä sitä yritin. Mietin yhteyden ottamista CSC:een tukeen asiaan liittyen, mutta minun olisi tällöin pitänyt odottaa vastausta ennen työn jatkamista. Siirryin siksi varasuunnitelmaan, eli tekemään erillistä selainpohjaista työkalua tiedostojen hakemiseen ja sen jälkeen komentorivityökalua tiedostojen lataamiseen ja purkamiseen. Haun suori-

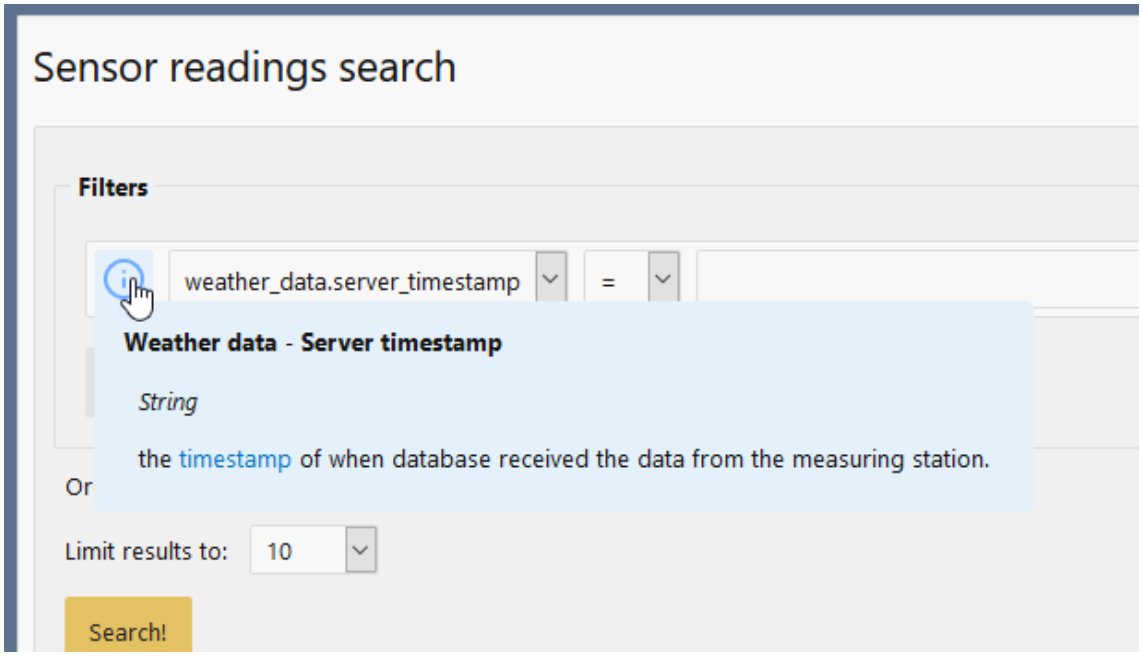
tettuaan käyttäjä voisi ladata hakutulokset tekstitiedostona, ja viedä sen manuaalisesti komentorivisovellukseen. Vaikka tällöin koko prosessi olisi loppukäyttäjälle melko kömpelö, yhdistäisi se sekä selain- että komentorivipohjaisten ratkaisujen hyvät puolet ja pienentäisi kehitykseen vaadittavaa aikaa.

Olin aluksi laittamassa jokaisen hakuparametrin omana kenttänään hakulomakkeeseen. Koska erilaisia hakuparametrejä on paljon, olisi lomakkeesta tällöin tullut kuitenkin iso ja vaikeakäyttöinen. Yritin keksiä parempaa ratkaisua ja lopputuloksena syntyi dynaamiseen suodatinlistaan perustuva rakenne (Kuvio 22). Lista on aluksi tyhjä, jolloin hakupainiketta painamalla saa kaikki tulokset suodattamattomana. Käyttäjä voi halutessaan lisätä tai poistaa rivejä suodatinlistasta. Yksittäinen suodatin muodostaa ikään kuin matemaattisen lausekkeen, johon käyttäjä määrittää halutun hakuparametrin, vertailuoperaattorin ja kohdearvon. Kuviossa 22 on esimerkiksi suodatettu tulokset, joissa ilman lämpötila on pienempi kuin nolla. Suodattimien välillä käytetään AND-logiikkaa, eli hakutulosten täytyy täyttää kaikkien suodattimien ehdot.



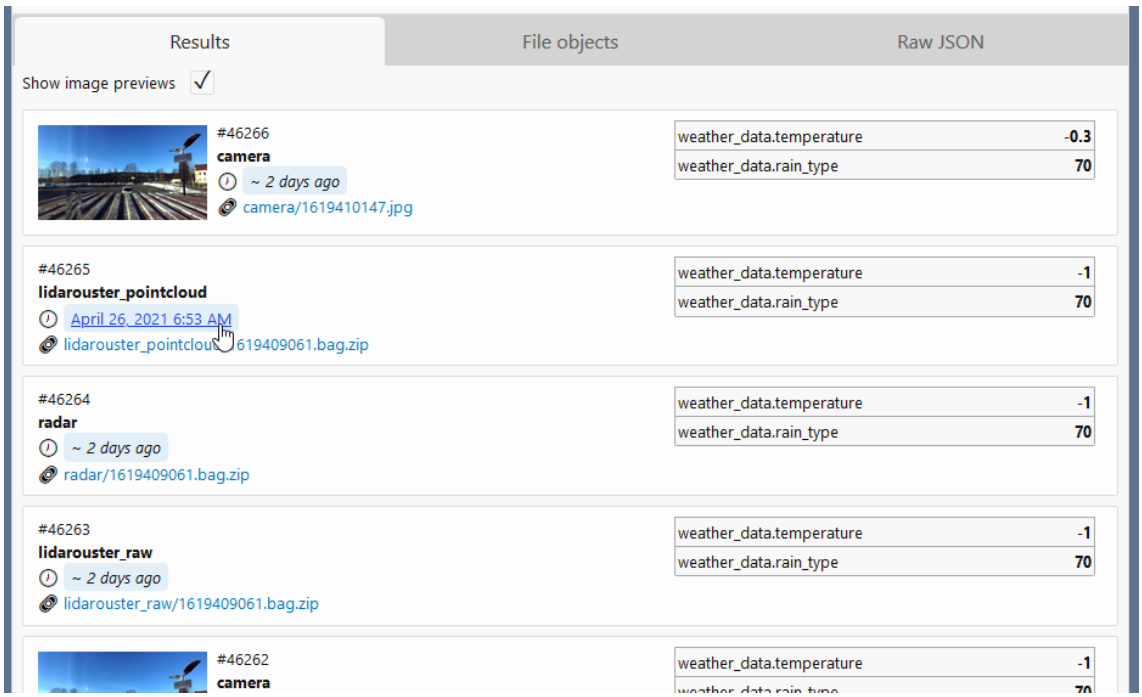

Kuvio 22. Sensoridatan hakutyökalu

Suodatinrivillä on lisäksi infonappi, jota painamalla ilmestyy ponnahdusikkuna, joka sisältää lisätietoa sillä hetkellä valitusta parametrista (Kuvio 23). Suodatinpohjainen haku oli jälkeenpäin ajatellen todella hyvä keksintö, joka teki hakusivusta erittäin helppokäyttöisen verrattuna siihen, mitä se olisi voinut olla muilla tavoilla toteutettuna.



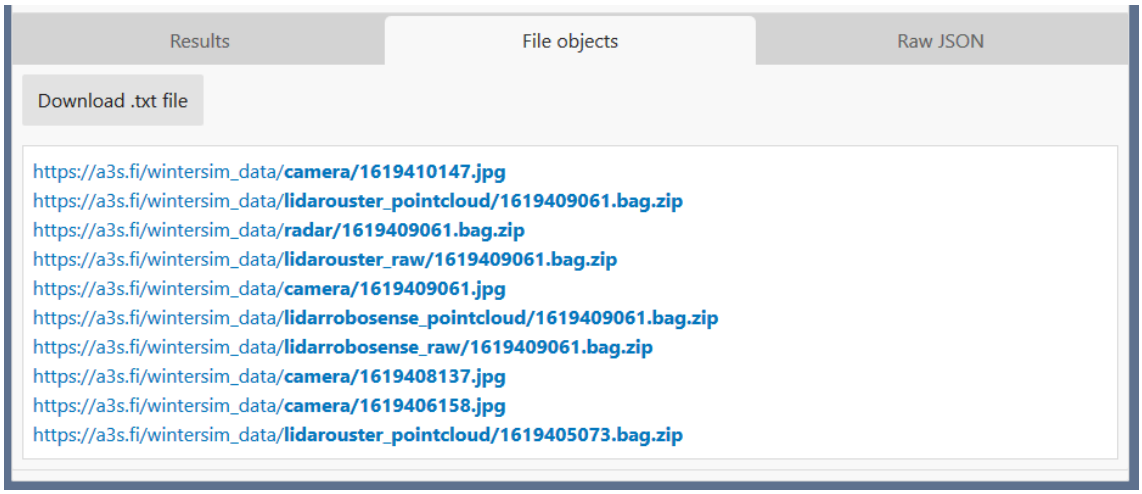

Kuvio 23. Suodatinriviin sisäänrakennettu ohjetoiminto

Hakutulokset on esitetty kolmessa eri muodossa. Eri muodot on eroteltu välilehtiä käyttäen. Oletusarvoisesti valitulla ensimmäisellä välilehdellä hakutulokset esitetään graafisesti (Kuvio 24). Hakutuloksesta löytyvät aina sensorin nimi, aikaleima ja suora linkki yksittäiseen sensoridatatiedostoon. Aikaleimassa näkyy oletuksena suhteellinen aika, paitsi jos käyttäjä liikuttaa kursorin sen päälle, jolloin aikaleima näytetään kokonaisena. Jos kyseessä olevan hakutuloksen sensori on kamera, näytetään kameran ottama kuva suoraan hakutulokseen liitettynä. Käyttäjän on näin helppo tarkastella hakutuloksien säätilaa, vuodenaikaa ja vuorokaudenaikaa visualisesti. Lisäksi hakutuloksessa näytetään valikoidusti ne parametrit, joiden perusteellä käyttäjä on suodattanut hakutulokset. Käyttäjä voi näin kätevästi tarkastella suodattimen läpi päässeitä arvoja ja halutessaan hienosäätää suodattimia. Kuvion 24 esimerkissä suodatettuna, ja tällöin myös hakutuloksissa näkyvänä, on lämpötila ja sateen tyyppitunniste.




Kuvio 24. Hakutulosten ensimmäinen välilehti

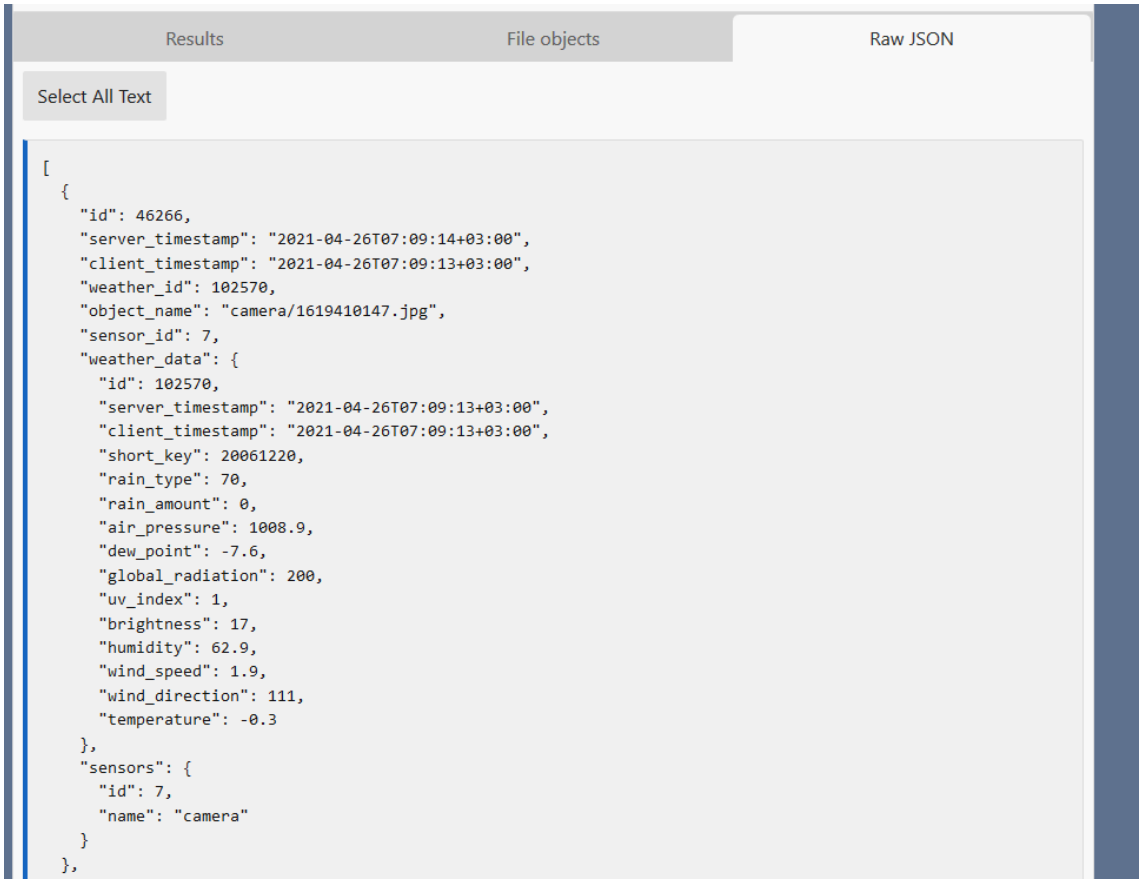
Toisella välilehdellä esitetään ainoastaan linkit hakutuloksiin liitettyihin sensoridatatiedostoihin (Kuvio 25). Välilehdellä on nappi, jota painamalla käyttäjä voi ladata linkit yhtenä tekstitiedostona komentorivisovellukseen vietäväksi (ks. luku 5.4).




Kuvio 25. Hakutulosten toinen välilehti

Kolmannella välilehdellä on esitettynä raaka, JSON-muodossa oleva data sellaisena, kun se on saatu rajapinnasta (Kuvio 26). Tämä näkymä on tehty alun perin testausta varten, mutta siitä käyttäjä voi myös nähdä kaikki parametrit, siinä

missä ensimmäisellä välilehdellä näkyy vain suodatukseen vaikuttavat parametrit.




Kuvio 26. Hakutulosten kolmas välilehti

## 5.4 Komentorivityökalu datan purkamiseen

Luvussa 5.3 kerrottujen syiden vuoksi oli tarpeellista erottaa sensoridatatiedostojen lataamiseen ja purkamiseen käytettävä toiminnallisuus erilliseen sovellukseen. Lähdin toteuttamaan sovellusta Pythonilla, koska se oli tuttu mittausasemaan tehtyjen ratkaisujen koodaamisesta ja minulla oli kaikki Python-kehitykseen tarvittava asennettuna.

Sovelluksen toiminta on todella suoraviivaista. Sovellukselle annetaan polku, jossa sijaitsee tekstitiedosto, joka sisältää listan sensoridatatiedostojen linkeistä (ks. luku 5.3). Tämän jälkeen sovellus yksitellen lataa tiedostot, purkaa ne, lukee niiden sisältämät ROS-datat ja tulostaa ne CSV-tiedostoon. Käyttäjät voivat tämän jälkeen analysoida CSV-muodossa olevaa dataa omilla työkaluillaan.

Sovelluksen ohjelmoinnissa ei ollut juurikaan hankaluuksia, mutta valmiin sovelluksen jakelussa niitä kyllä oli. Python-koodin jakaminen sellaisenaan ei osoittautunut hyväksi tavaksi, koska asennusprosessi oli tällöin käyttäjälle monimutkainen ja Pythonin asennus vaati järjestelmänvalvojan oikeuksia, joita käyttäjillä ei työkoneillansa ollut. Lisäksi pari ROS-datan käsittelyyn tarvittavaa pakettia ei tullut Pythonin omasta pakettien jakelupaikasta eli PyPI:stä, vaan GitHubista, mikä monimutkaisti asennusprosessin automatisointia. Vaikka sain tarvittavien pakettien automaattisen asennuksen lopulta toimimaan omalla koneella, se ei toiminut sellaisenaan käyttäjien koneilla. Asennuksesta oli siis tarpeen tehdä täysin riippumaton Pythonin tai sen pakettien versioista tai muista koneiden välillä muuttuvista tekijöistä.

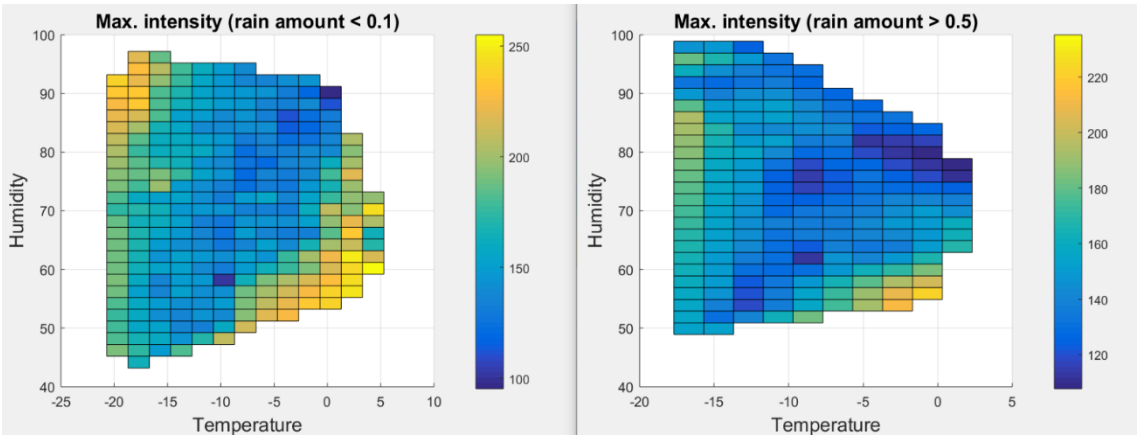
Lopulta sain asennusprosessin yksinkertaistettua PyInstaller-sovelluksen avulla. PyInstaller pakkaa Pythonin ja kaikki tarvittavat paketit yhdeksi suoritettavaksi tiedostoksi, joka Windowsilla on EXE-muotoinen (PyInstaller Development Team 2021). EXE-tiedoston voi käynnistää lataamisen jälkeen, eikä sitä tarvitse erikseen asentaa. Tämä osoittautui paljon paremmaksi tavaksi Python-sovelluksen jakamiseen ja käyttäjätkin olivat tyytyväisiä.

# 6 TULOKSET

## 6.1 Alustavia tutkimustuloksia

Keväällä 2021 tehtiin alustavia laskelmia kaikista Robosense-lasertutkan tekemistä mittauksista helmikuun ajalta. Mittauksia laskelmaan tuli tällöin yhteensä 989 ja jokaiseen mittaukseen kuuluu noin 97 lasertutkan skannausta. Jokaisesta mittauksesta valittiin yksi satunnainen skannaus prosessin nopeuttamiseksi.

Alustavien laskelmien perusteella sade näyttäisi haittaavan lasertutkien toimintaa, mikä on linjassa vastaavien tieteellisten tutkimusten kanssa (Khader & Cherian 2017, 3). Sääasemalta saatu sadearvo on tosin lähinnä vain suuntaa antava, koska aseman sadetta mittaava sensori ei välttämättä toimi samalla lailla lumi- ja vesisateessa. Seuraavassa kuviossa 27 on esitetty sademäärän vaikutus lasertutkan intensiteettiin visuaalisesti.




Kuvio 27. Sademäärän vaikutus yhden heijastinlevyn heijastuksen maksimi-intensiteettiin (Korhonen 2021)

## 6.2 Datan hyödyntäminen simulaatiossa

Alustaviin laskelmiin (Luku 6.1) perustuen laskettiin havaintoja mukaileva matemaattinen kaava. Kaava ohjelmoitiin käyttöön WinterSim-projektiryhmän tekemään omaan versioon Carla-simulaattorista, ja sillä pystytään onnistuneesti toistamaan tosielämän lasertutkan toimintaa.

Carla on autonomisten ajoneuvojen kehitystä varten tehty, avoimen lähdekoodin simulaatioalusta. Carla on rakennettu Unreal-pelimoottorin päälle, mutta vaikka tällöin sen ohjelmointikieli onkin C++, löytyy Carlasta myös modulaarinen Python-rajapinta, mikä antaa käyttäjien ohjelmoida omia simulaatiotilanteitaan todella vaivattomasti. Koska Python-skriptit toimivat jokainen omina moduuleinaan, voi niiden käyttöä yhdistellä rinnakkain vapaasti. Tämä tekee Carlasta helppokäyttöisen ja joustavan myös peruskäyttäjille autoalan ammattilaisten lisäksi. (Dosovitsky, Ros, Codevilla, López & Koltun 2017, 1 – 3.)

Carlassa ei oletuksena ole kovin monipuoliset ympäristöt. Talvimaisemaa ei ole ollenkaan, ja eri säätiloja on toteutettu vain visuaalisina efekteinä. Tämän vuoksi WinterSim-projektissa luodaankin täysin alusta asti uudet, pohjoiseen sijoittuvat kartat Carlaan. Ensimmäisen kartan sijaintina toimii Muoniossa sijaitseva Aurora-älytie ja sen ympäristö. Muonion mallintamista varten projektiryhmä on kerännyt referenssidataa 360-asteen videokameraa hyödyntäen (Kuvio 28). Toinen kartta sijoittuu Rovaniemen keskustaan.

Kuvio 28. 360-asteinen videokuvaus simulaation mallinnuksia varten

Projektiryhmä on laajentanut Carlan säätiloja ja ympäristöelementtejä sisältämään talvisään ja lumihanget (Kuvio 29). Myös autokantaa ja jalankulkijoiden pukeutumista muokataan sopimaan paremmin pohjoiseen miljööseen.

Kuvio 29. Muonio-kartta ja talvisää Carla-simulaatiossa

Carlaan löytyy myös ROS:ia tukevia lisäosia, mikä mahdollistaa sensoridatan suoran lähettämisen mittausasemalta simulaatioon. Staattisen mittausaseman tapauksessa sille ei tosin ole löytynyt mitään varsinaista käyttöä, muuten kuin mittauksen esittelyssä. Tulevaisuudessa tapahtuvassa liikkuvassa testauksessa se voi tosin olla hyödyllinen ominaisuus.

WinterSim-projektin omasta Carla-versiosta on tehty projektin kuluessa visuaalisesti näyttävä kokonaisuus, jota kelpaa esitellä projektista kiinnostuneille. Simulaation uskottavuutta lisää kuitenkin huomattavasti se seikka, että sen toiminta perustuu oikeaan tutkimustietoon, vaikka itse datan merkitys simulaation käyttäjäkokemukseen onkin melko pieni. Simulaatioalusta on tällöin potentiaalisesti houkuttelevampi yritysten silmissä ja näin hyödyksi autonomisten ajoneuvojen kehitystyölle.

# 7 POHDINTA

Näin laajan datankeräysjärjestelmän suunnittelu ja toteuttaminen oli projektiryhmälle melkoinen työmaa, mutta toisaalta näin jälkeenpäin ajatellen myös hieno saavutus. Useiden eri laitteiden, tekniikoiden ja ohjelmointiratkaisujen saaminen toimimaan luotettavasti ja yhtäaikaisesti ei tosiaankaan ole itsestään selvää. Siksi arvioisinkin, että ongelmia oli loppujen lopuksi suhteellisen vähän järjestelmän laajuuteen nähden.

Kaikki esiin tulevat ongelmatilanteet ratkaistiin ja haasteet ylitettiin tavalla tai toisella. Projektiryhmäläisten monimuotoinen tietotekniikan osaaminen pääsi oikeuksiinsa, kun jokainen löysi laajasta projektista oman osaamisalueensa työtehtävät. Joku teki sähkötöitä, joku toinen koodasi ja kolmas 3D-mallinsi. Projektiryhmäläisten ahkeruus ja tiimityötaidot tekivät työnteosta rattoisaa.

Kokonaisuudessaan opinnäytetyö ja sen pohjana oleva WinterSim-projekti tarjosivat alusta lähtien todella mielenkiintoisia ja opettavaisia työhetkiä. Opin paljon uutta ajoneuvoista, sensoreiden toiminnasta, simuloinnista ja tietysti datan käsittelyyn liittyvistä tekniikoista.

On kiinnostavaa seurata liikenteen automaation ja sensoreiden kehitystä tulevaisuutta kohti mentäessä. Mikä vielä innoittavampaa, sain tämän projektin kautta olla omalta pieneltä osaltani edistämässä alan kehitysaskelia. Uskon, että projektin tuotoksilla on paljon potentiaalia edistää ajoneuvoja koskevaa kehitystyötä. Joka tapauksessa se on ainakin antanut minulle henkilökohtaisesti runsaasti käytännön kokemusta, josta on varmasti hyötyä jatkossa.

LÄHTEET

Burke, K. 2019. How Does a Self-Driving Car See? Camera, radar and lidar sensors give autonomous vehicles superhuman vision. Viitattu 29.4.2021 https://blogs.nvidia.com/blog/2019/04/15/how-does-a-self-driving-car-see/.

Clark, J. 2020. Top 10 backend frameworks. Viitattu 29.4.2021 https://blog.back4app.com/backend-frameworks/.

Cooper Hewitt 2018. Waymo, Sensor Visualization | The Road Ahead: Reimagining Mobility. Viitattu 29.4.2021 https://www.youtube.com/watch?v=wKNvzLgTYhQ.

CSC 2021. Introduction to the Allas storage service. Viitattu 29.4.2021 https://docs.csc.fi/data/Allas/introduction/.

Dosovitsky A., Ros G., Codevilla F., López A. & Koltun V. 2017. CARLA: An Open Urban Driving Simulator. Proceedings of the 1st Annual Conference on Robot Learning, 1–16. Viitattu 29.4.2021 http://proceedings.mlr.press/v78/dosovitskiy17a/dosovitskiy17a.pdf.

Euroopan parlamentti 2019. Itseohjautuvat autot pian todellisuutta EU:ssa. Viitattu 29.4.2021 https://www.europarl.europa.eu/pdfs/news/expert/2019/1/story/20190110STO23102/20190110STO23102_fi.pdf.

Google Developers 2021. Schedule tasks with WorkManager. Viitattu 29.4.2021 https://developer.android.com/topic/libraries/architecture/workmanager.

Händel, C., Konttaniemi, H. & Autioniemi, M. 2018. State-of-the-Art Review on Automotive Radars and Passive Radar Reflectors. Arctic Challenge research project. Series B. Research reports and compilations 6/2018. Viitattu 29.4.2021 https://www.lapinamk.fi/loader.aspx?id=983f39aa-c97f-4f2d-bb39-abe006c6bad3.

Kahala, V. 2020. Autonomisten ajoneuvojen regulaation haasteet vastuun, eettisyyden ja datan näkökulmasta. Tampereen yliopisto. Teknis-taloudellinen tutkinto-ohjelma. Kandidaatintyö. Viitattu 14.5.2021 https://trepo.tuni.fi/bitstream/handle/10024/122369/KahalaV%C3%A4in%C3%B6.pdf?sequence=2&isAllowed=y.

Khader, M. & Cherian, S. 2017. An Introduction to Automotive LIDAR. Dallas: Texas Instruments. Viitattu 29.4.2021 https://www.ti.com/lit/wp/slyy150a/slyy150a.pdf?ts=1616761714486.

Kocić, J., Jovičić, N. & Drndarević, V. 2018. Sensors and Sensor Fusion in Autonomous Vehicles. Belgrad: University of Belgrade. Viitattu 29.4.2021 https://www.researchgate.net/profile/Jelena-Kocic/publication/329153240_Sensors_and_Sensor_Fusion_in_Autonomous_Vehicles/links/5c6c65c692851c1c9dee9030/Sensors-and-Sensor-Fusion-in-Autonomous-Vehicles.pdf.

Korhonen, M. 2021. Data-analyysiä viikolta 9. WinterSim-projektin tiedostot. Rovaniemi: LapinAMK.

Koskinen, A.-L. 2020. Keltaiset viivat poistuvat kohta Suomen maanteiltä – syynä insinöörien näkemykset väristä, robottiautot ja laki. Viitattu 29.4.2021 https://yle.fi/uutiset/3-11539676.

Lapin AMK 2020a. WinterSim-projektisuunnitelma. Rovaniemi: Lapin AMK.

Lapin AMK 2020b. WinterSim. Lapin AMK Hanketietojen haku. Viitattu 29.4.2021 https://www.lapinamk.fi/fi/Yrityksille-ja-yhteisoille/Lapin-AMKin-hankkeet?RepoProject=4205700060.

Lufft 2019. Manual. Weather Station WS10. Viitattu 29.4.2021 https://www.lufft.com/fileadmin/lufft.com/07_downloads/1_manuals/Manual_WS_10_EN_V1-5_01.pdf.

Lufft 2021. WS10 Smart Weather Sensor – discontinued. Viitattu 29.4.2021 https://www.lufft.com/products/accessories-310/lufft-ws10-smart-weather-sensor-2421/.

Lutkevich, B. 2019. Self-driving car (autonomous car or driverless car). Techtarget 10/2019. Viitattu 29.4.2021 https://searchenterpriseai.techtarget.com/definition/driverless-car.

MDN Web Docs 2021a. Cross-Origin Resource Sharing (CORS). Viitattu 29.4.2021 https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS.

MDN Web Docs 2021b. What is a URL? Viitattu 14.5.2021 https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL.

Metz, C. & Griffith, E. 2020. This Was Supposed to Be the Year Driverless Cars Went Mainstream. The New York Times 12.5.2020. Viitattu 29.4.2021 https://www.nytimes.com/2020/05/12/technology/self-driving-cars-coronavirus.html.

Microsoft 2021. Relational vs. NoSQL data. Viitattu 29.4.2021 https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/relational-vs-nosql-data.

Nelson, J. & Chavez, S. 2017a. Tutorial 1 – The Golden Key. Viitattu 29.4.2021 https://postgrest.org/en/stable/tutorials/tut1.html.

Nelson, J. & Chavez, S. 2017b. Tables and Views. Viitattu 29.4.2021 https://postgrest.org/en/stable/api.html.

Open Robotics 2014. ROS/Concepts. Viitattu 29.4.2021 http://wiki.ros.org/ROS/Concepts.

Pedro, B. 2017. Best practices for securely storing API keys. Viitattu 29.4.2021 https://www.freecodecamp.org/news/how-to-securely-store-api-keys-4ff3ea19ebda/.

PyInstaller Development Team 2021. PyInstaller Quickstart – PyInstaller bundles Python applications. Viitattu 29.4.2021 https://www.pyinstaller.org/.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R. & Ng, A. 2009. ROS: an open-source Robot Operating System. ICRA workshop on open source software. 3/2009. Viitattu 29.4.2021 http://www.robotics.stanford.edu/~ang/papers/icraoss09-ROS.pdf.

Ramasubramanian, K. & Ginsburg, B. 2017. AWR1243 sensor: Highly integrated 76–81-GHz radar front-end for emerging ADAS applications. Dallas: Texas Instruments. Viitattu 29.4.2021 https://www.ti.com/lit/wp/spyy003/spyy003.pdf.

RedHat 2021. What is a REST API? Viitattu 29.4.2021 https://www.redhat.com/en/topics/api/what-is-a-rest-api.

REST API Tutorial 2020. What is REST? Viitattu 29.4.2021 https://restfulapi.net/.

Sorokin, H. 2020. Autonomisen ajamisen nykytila ja tulevaisuuden näkymät. Metropolia Ammattikorkeakoulu. Auto- ja kuljetustekniikka. Insinöörityö. Viitattu 14.5.2021 https://www.theseus.fi/bitstream/handle/10024/279684/sorokin_henri.pdf?sequence=2&isAllowed=y.

Stack Overflow 2020. Web Frameworks. 2020 Developer Survey. Viitattu 29.4.2021 https://insights.stackoverflow.com/survey/2020#technology-web-frameworks.

Stereolabs 2019. ZED 2 Camera and SDK Overview. Viitattu 29.4.2021 https://www.stereolabs.com/assets/datasheets/zed2-camera-datasheet.pdf.

W3 2021. Same Origin Policy. Viitattu 29.4.2021 https://www.w3.org/Security/wiki/Same_Origin_Policy.

Wickramarachchi, V. 2021. Should We Use CSS or SCSS in 2021? With the continuously improving CSS over the years, is SCSS relevant anymore? Let’s find out. Viitattu 29.4.2021 https://blog.bitsrc.io/will-scss-be-replaced-by-css3-754842d6b681.

Zhao, X., Sun, P., Xu, Z., Min, H. & Yu H. 2020. Fusion of 3D LIDAR and Camera Data for Object Detection in Autonomous Vehicle Applications. Ieee Sensors Journal. 9/2020. 4901–4913. Viitattu 14.5.2021 https://doi.org/10.1109/JSEN.2020.2966034.