Aaro Leikari

Reunalaskennan hyödyntäminen liukkaiden olosuhteiden havaitsemisessa

Tietotekniikan pro gradu -tutkielma

6. kesäkuuta 2022

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Aaro Leikari

Yhteystiedot: aaro.leikari@gmail.com

Ohjaaja: Timo Hämäläinen

Työn nimi: Reunalaskennan hyödyntäminen liukkaiden olosuhteiden havaitsemisessa

Title in English: Edge computing in monitoring the formation of slippery conditions

Työ: Pro gradu -tutkielma

Opintosuunta: IoT ja tietoliikenne

Sivumäärä: 61+9

Tiivistelmä: IoT on mullistanut maailman jossa nykyään elämme ja se tuo valtavasti uusia mahdollisuuksia ja tapoja kanssakäydä ympäristömme kanssa. Ympäristöstä kerättävä datan määrä on kasvanut valtavasti, sillä sensoreita voidaan upottaa lähes mihin tahansa ja tämän datan prosessoimiseen on olemassa tehokkaita työkaluja. Pilvipalvelut ovat pitkään olleet suosittu tapa varastoida ja prosessoida kerättyä dataa, mutta koska datan määrä on niin valtava, ei ole kovin mielekästä lähettää tällaista suurtaa määrää dataa pitkiä matkoja. Jossakin tilanteissa saattaa olla tarkoituksen mukaisempaa ja jopa välttämätöntä, että data prosessoidaan osittain tai kokonaan lähellä sen lähdettä ja että kyetään nopeasti vastaamaan muuttuviin olosuhteisiin.

Onkin varsin ilmiselvää, että pelkästään pilvipalveluita käyttämällä ei tällaisiin vaatimuksiin voida päästä. Siksi tähän rinnalle on noussut uusi paradigma jota nimitetään reunalaskennaksi. Mikrokontrollerien kehitys muistin, laskentatehon ja koon pienentymisen puolesta on mahdollistanut sen, että kerättyä dataa voidaan prosessoida jo hyvin aikaisessa vaiheessa lähellä datan lähdettä. Tämä avaa uusia sovelluskohteita, sekä hyötyjä esimerkiksi tietoturvan ja datan perusteella tehtävien toimenpiteiden suorittamisen suhteen. Kuitenkin tällä saralla on vielä paljon tutkimusta tehtävänä. Tässä tutkielmassa tutkittiin reunalaskennan soveltamista liukkauden havainnoinnissa. Tutkielmassa rakennettiin yksinkertainen sovellus, joka kykenee sensoreilla keräämään dataa ympäristöstä, prosessoimaan dataa, syöttämään prosessoidun datan tekoälymallille ja näin tuottamaan tuloksia datan pohjalta. Sovellusta testattiin

viikon ajan ja tulokset kirjattiin lokitiedostoon. Tuloksista selviää, että vaikka itse tutkimus ei onnistunutkaan täysin kuten oltiin alkuun suunniteltu, on reunalaskennalle silti nähtävissä potentiaalia valitun skenaarion tarkkailussa.

Avainsanat: Reunalaskenta, reunaäly, IoT, tekoäly, koneoppiminen, langaton kommunikaatio, älykäs ympäristö

Abstract: IoT has revolutionized the world we live in today and it brings enormous possibilities and ways we can interact with the world around us. The amount of data gathered from our environment has increased enormously because now a days sensors can be embedded almost everywhere. On the other hand the tools used to process this gathered data have become more and more effective, too. For a long time cloud computing has been the most widely used method to process this data and cloud services are used to store data. However, the amount of data has become so massive that it is not very efficient to move such big loads of data for long distances. In some cases it might be more practical or even essential to process the data on the location where it is generated and in turn to take action in different situations.

Given this statement, it is quite obvious that cloud computing alone can not meet these requirements. The pressure to meet these requirements has created a new computing paradigm called edge computing. The advancements in microcontroller techonologies, specifically size, computing capacity and increased memory, are the key features making edge computing possible. Edge computing opens up a whole new world of possibilities with applications and benefits for example with information security and with the speed that a certain action can be taken based on data from environment. In this research an application was built that uses edge computing. The goal was to examine, how could an application using edge computing be used to monitor the formation of slippery conditions. The application gathers data from the environment using sensors, processes the gathered data and feeds that processed data to an ai model which then makes an inference and that inference is stored. Even though the research itself did not go as was planned, it is safe to say that edge computing could still be used in the scenarion examined in this research.

Keywords: Edge computing, edge ai, IoT,artificial intelligence, machine learning, wireless

communication, smart environment

Termiluettelo

IoT	Internet of Things, esineiden internet, tarkoittaa fyysisiä laitteita, jotka kykenevät kommunikoimaan toisten laitteiden kanssa esimerkiksi internetin välityksellä.
sensori	Pieni systeemi jolla voi mitata dataa ympäristöstä.
mikrokontrolleri	Pieni mikropiiri, joka sisältää oman prosessorinsa ja muistinsa.

Kuviot

Sisällys

1 Johdanto

Viime vuosina erilaiset mikrokontrollerit ja sensorit ovat kehittyneet ja niiden käyttö erilaisissa ympäristöissä on yleistynyt merkittävästi. Nämä ympäristöt ulottuvat teollisuuden haastavista olosuhteista aina urbaaniin kaupunkiympäristöön ja lähes kaikkeen siltä väliltä. Mikrokontrollereita ja sensoreita on mahdollista upottaa lähes mihin tahansa kuten esimerkiksi autoihin tai kodinkoneisiin. Lisäksi niiden koko on pienentynyt, jonka ansiosta niitä voidaan lisätä esimerkiksi jopa vaatteisiin tai rannekelloihin. Näillä mikrokontrollereilla ja sensoreilla voidaan kerätä suuret määrät arvokasta dataa, jota jatkojalostamalla kyetään tuottamaan ihmisille yleishyödyllisiä palveluita, kuten ilmanlaadun seurantaa tai esimerkiksi urheilusuorituksen aikana tehtävää sykkeen mittausta tai kalorien kulutusta.

Ympäristöstä kerättävän datan määrä on kasvanut valtavasti ja mitä todennäköisimmin tulevaisuudessa tämän datan määrä kasvaa vielä entisestään. Tällainen kehitys luo paineen, kuinka dataa voidaan prosessoida mahdollisimman tehokkaasti ja kuinka sen varastointi onnistuu mahdollisimman järkevästi. Fyysisen tietoteknisen laitteiston ylläpito datan prosessointiin ja varastointiin saattaa käydä pitkällä aikavälillä kalliiksi, sillä tällainen muoto aiheuttaa kuluja esimerkiksi huoltotoimenpiteiden ja ohjelmistopäivitysten muodossa, joista laitteiston omistava taho joutuu vastaamaan itse. Tämän takia pilvipalveluiden suosio on kasvanut, sillä tällaisten palveluiden käyttö siirtää, joko osittain tai kokonaan vastuun fyysisten laitteiden ylläpitämisestä toisille tahoille. Pilvipalvelut tarjoavat joustavan alustan lukuisine palveluineen datan prosessointiin ja varastointiin. Tämä ei kuitenkaan ratkaise koko ongelmaa. Koska kerättävän datan määrä on valtava, ei enää ole kovin mielekästä siirtää suuria datamääriä tietoverkkoja pitkin sen lähteeltä palvelimille. Tällainen toiminta muun muassa kuormittaa tiedonsiirtokanavia, eikä se ole kovin tietoturvallista. Tämän takia datan prosessointia ollaan alettu siirtää lähemmäksi datan lähdettä.

Reunalaskenta on paradigma, jossa data käsitellään lähempänä sen lähdettä. Tällainen toiminta on mahdollista, sillä mikrokontrollerit ovat parantuneet prosessointitehossa ja niiden muistia on voitu kasvattaa, vaikka se ei edellenkään ole kovinkaan suuri. Myös tekoäly on kehittynyt valtavasti viimeisimpien vuosien aikana ja tekoälymalleja voidaan muuntaa kevyempään muotoon, jolloin niitä voidaan ajaa mikrokontrollereilla. Tällaista toimintaa, jos-

sa tekoälymallia ajetaan mikrokontrollerilla kutsutaan reunaälyksi. Reunalaskennan avulla on mahdollista prosessoida dataa jo etukäteen lähellä datan lähdettä ja näin siirtää taakkaa pois tiedonsiirtokanavilta valtavien datamäärien lähettämisessä ja pilvipalveluilta valtavien datamäärien käsittelyssä.

Yksi reunlaskennasta saatava merkittävä hyöty on kyky regoida datasta saatuihin tuloksiin varsin nopeastikin. Tämä on erittäin hyödyllistä jos mietitään tilanteita, jotka vaativat nopeaa reagointia muuttuviin olosuhteisiin kuten esimerkiksi ajoneuvoissa ajonhallintajärjestelmissä tai teollisuusympäristössä ongelmatilanteiden sattuessa. Yksi mahdollinen reunalaskennan sovelluskohde on talvella esiintyvien liukkaiden olosuhteiden torjunnassa. Tässä tutkimuksessa on tarkoituksena tutkia, miten reunlaskenta voisi soveltua tällaiseen. Tutkimuksessa rakennetaan yksinkertainen hiekkalaatikkoversio reunaälyä hyödyntävästä sovelluksesta, joka kerää sensoreilla dataa ympäristöstään, tekee tarvittavan muokkauksen datalle ja tämän jälkeen syöttää datan tekoälymallille, joka tekee datasta päätelmän, miten todennäköisesti liukkautta saataa esiintyä. Reunaälysovellus kerää mallin päätelmät sille syötettyine parametreineen lokitiedostoon, josta sen toimintaa on helppo seurata.

Tutkielman toisessa luvussa käsitellään älykästä ympäristöä, luvussa käydään läpi, mikä älykäs ympäristö on, älykkään ympäristön sovelluskohteita, älykkäitä kaupunkeja, sekä sensoriverkkoja ja pilvipalveluita, jotka olennaisesti liittyvät älykkääseen ympäristöön. Kolmannessa luvussa käsitellään reunalaskentaa, mitä se on, siitä saatavat hyödyt, sekä käydään läpi, mitä on reunaäly. Neljännessä luvussa tarkastellaan reunalaskennan ja reunaälyn tutkimuksia ja sovelluksia. Viidennessä luvussa käsitellään langatonta kommunikaatiota, mitä se on, miten se tapahtuu, sekä joitakin teknologioita, joita hyödynnetään langattomassa kommunikaatiossa. Kuudennessa luvussa käsitellään tutkimuksen suorittaminen, mitä tutkimuksella tavoiteltiin ja mitä tutkimusmenetelmää käytettiin, kuvaillaan sovellus, miten aineisto kerättiin ja muokattiin, sekä millä tavalla tekoälymalli tehtiin. Seitsemännessä luvussa käsitellään järjestelmän toimintaan laittaminen. Kahdeksannessa luvussa käsitellään sovelluksen testausta ja sen tuottamia tuloksia. Yhdeksäs luku sisältää tutkimuksen pohdinnan ja johtopäätökset.

2 Älykäs ympäristö

Tässä luvussa käsitellään älykästä ympäristöä, sen sovelluskohteita, sekä sensoriverkkoja ja pilvipalveluita.

2.1 Mitä älykäs ympäristö tarkoittaa?

IoT:n kehittymisen myötä meitä ympäröivä maailma ja siinä olevat esineet ovat yhä enemmän kiinni internetissä. Tämä mahdollistaa näiden laitteiden välisen kommunikaation, sekä uusia tapoja, joilla ihmiset voivat kontrolloida näitä laitteita. Tämän myötä on kehittynyt käsite "älykäs ympäristö". Älykäs ympäristö mahdollistaa saumattoman ihmisten ja ympäristön vuorovaikutuksen (Nugent ym. 2014).

Karkeasti ottaen älykkäästä ympäristöstä on tunnistettavissa kolme selkeää komponenttia:

- sensorit
- datan prosessointi ja
- ympäristön hallitseminen.

Sensorit ovat niitä laitteita, jotka ovat upotettuna ympäristöön. Sensoreiden tehtävä on kerätä tietoa ympäristöstä, jossa ihmiset suorittavat erilaisia tehtäviä. Datan prosessointi on keskeinen komponentti. Siinä pyritään tekemään havaintoja ympäristöstä, esimerkiksi muutosten seuranta tai poikkeuksien havainnnointi. Ympäristön hallitsemisen tarkoituksena on tehdä tarvittavat toimenpiteet prosessoidun datan perusteella, ihmisen tai esimerkiksi jonkin toisen laitteen toimesta. Esimerkiksi jos jostakin huoneesta saadut lämpötilamittaukset osoittavat kyseisen huoneen lämpötilan olevan liian korkea tai liian matala, voidaan toisen laitteen tai ihmisen toimesta tehdä toimenpiteitä tämän korjaamiseksi. (Nugent ym. 2014)

2.2 Älykkään ympäristön sovelluskohteita

Älykkyyttä voidaan tuoda moniin eri skenaarioihin. Käytännöllisimmät esimerkit tällaisista skenaarioista lienevät kodit, terveydenhuolto, teollisuus ja kaupungit. Kodit ovat varsin sopi-

va ympäristö älykkyyden soveltamiseen. Yleensä tällaisissa ympäristöissä on jo valmiina iso määrä teknologisia laitteita. Kodit ovat kontrolloituja ympäristöjä (tarkoittaen, että perusasiat kuten esimerkiksi ympäristön lämpötila ja ilmankosteus on säädelty vastaamaan toimintatarpeita ja että ympärstö on eristetty muista ympäristöistä ("Liberty" 2020)) ja kodin omistajat kykenevät tarjoamaan vaadittavan pääoman teknologisten ratkaisujen rakentamiselle ja ylläpidolle. Lisäksi älyratkaisut voivat tarjota kodin asuttajille laajan valikoiman erilaisia, yleistä elämistä hyödyttäviä ratkaisuja. Älykodin palvelut voidaan karkeasti ottaen jakaa kahteen kategoiriaan: avustavat palvelut, sekä ylläpitävät palvelut. Avustavat palvelut pyrkivät tarjoamaan suoraa tukea kodin asukkaille jokapäivisissä toimissa. Avustavat palvelut voidaan myös räätälöidä asukkaan tarpeisiin. Ylläpitävät palvelut tarjoavat tukea ylläpitoon liittyvissä tehtävissä, kuten esimerkiksi kodin turvallisuudessa tai energian säästämisessä. (Gomez ym. 2019)

Sensoreiden halventuminen sekä langattomien kommunikaatiomenetelmien parantuminen on tuonut mahdollisuuden lisätä älykkyyttä myös terveydenhuoltoon. Tyypillisin keino lisätä älyä terveydenhuollon ympäristöön on liittää sensoreita esimerkiksi ympäristöön jossa henkilö on, liittää nämä sensorit henkilöön itseensä tai käyttää näitä molempia keinoja. Näistä sensoreista voidaan jatkuvasti tai tietyin väliajoin kerätä dataa, prosessoida tämä data ja näin antaa palautetta esimerkiksi henkilölle itselleen, sekä muulle terveydenhuollon henkilökunnalle sen hetkisestä tilasta. Älyn avulla voidaan myös parantaa hätätilojen havaitsemista, henkilön kunnon tarkkailua, sekä yleistä henkilön avustamista ja hätätapausten ehkäisyä. Hätätilojen havaitsemisesta yksi merkittävä esimerkki on kaatumisen havaitseminen, joka on etenkin vanhuksilla johtava syy monille hätätiloille. Henkilön kuntoa voidaan jatkuvasti tarkkailla ja näin havaita laskua esimerkiksi terveydentilassa. (Gomez ym. 2019)

2.3 Älykkäät kaupungit

Kaupungit ovat alati muuttuvassa tilassa. Mitä todennäköisimmin kaupungistuminen tulee tulevaisuudessa kiihtymään tai vähintäänkin jatkumaan samalla tavalla. Tämä on luonnollinen kulku, sillä kaupungit pystyvät tarjoamaan kattavan valikoiman palveluja ihmisille, jotka tulevat eri taustoista. Ihmiset muuttavat syrjäseuduilta kaupunkeihin, joiden koko tulee kasvamaan ja joiden täytyy yhä paremmin pystyä sopeutumaan ihmisten tarpeisiin. Lisäksi kau-

pungit vaikuttavat suurissa määrin ympäpristöönsä luomalla esimerkiksi melua ja päästöjä. Siksi on varsin luonnollista, että myös kaupunkiympäristöön tullaan yhä enenevissä määrin lisäämään älykkyyttä.

Termiä "älykaupunki"käytettiin ensimmäisen kerran 1990. Tuohon aikaan termin ensisijainen merkitys oli nousussa olevan uuden tietotekniikan ja modernien kaupunki-infrastruktuurien kontekstissa. Termille ei kuitenkaan ole mitään yksittäistä määritettä. "Älykaupunki"termiä käytetään tavoilla, jotka eivät aina ole kovin yhdenmukaisia, eikä ole olemassa yhtä viitekehystä, johon kyseinen määritelmä voitaisiin täysin sovittaa. (Albino, Berardi ja Dangelico 2015)

Joitakin määritelmiä kuitenkin ollaan esitetty ja jokainen yksittäinen taho määrittelee sen parhaaksi näkemällään tavalla. Bakıcı, Almirall ja Wareham (2013) mukaan älykaupunki on teknologisesti kehittynyt kaupunki, joka yhdistää ihmiset, informaation ja kaupunkielementit uusien teknologioiden avulla, jotta voitaisiin luoda kestävä, vihreämpi kaupunki, joka pystyy käymään kauppaa kilpailukykyisesti ja innovatiivisesti ja joka takaa hyvän elinlaadun sen asukkaille. Barrionuevo, Berrone ja Ricart (2012) mukaan taas älykäs kaupunki tarkoittaa kaiken saatavilla olevan teknologian ja resurssien hyödyntämisen älykkäällä ja järjestelmällisellä tavalla, jotta voidaan kehittää urbaaneja, asutettavia ja kestäviä kaupunkikeskuksia. Kolmannen määritelmän mukaan, jonka esittää Caragliu, Del Bo ja Nijkamp (2011), kaupunki on älykäs, kun panostukset esimerkiksi ihmisiin, sosiaaliseen pääomaan ja moderniin teknologiaan ruokkivat kestävää ekonomista kehitystä ja korkeaa elintasoa, samalla hoitaen viisaasti luonnonvaroja. Lisää määritelmiä löytyy mm. Cretu (2012), Giffinger ym. (2007) ja Harrison ym. (2010).

Kuten jo aikaisemmin mainittiin, älykaupungille ei ole mitään yhtä ja yhtenäistä määrittelyä. Yllä lainattujen määritysten perusteella voidaan kuitenkin sanoa joitakin yhtymäkohtia olevan. Keskeisimpiä käsitteitä älykaupungeissa näyttäisivät olevan kestävän kaupunkiympäristön luominen, vahva teknologian hyödyntäminen, sekä elämänlaadun parantaminen kaupungin asukkaille.

2.4 Sensoriverkot

Elektronisten komponenttien pienentyessä avautui mahdollisuuksia rakentaa yhä pienempiä ja pienempiä tietoteknisiä järjestelmiä. Yksi tällainen pienten tietoteknisten järjestelmien rakennussuuntaus oli mikrosysteemien (*MEM, microelectromechanical systems; MST, microsystems technology*) rakentaminen. Nykyään tällaisia mikrosysteemejä nimitetään ehkä kaikille hieman tutummalla nimellä sensori. Sensorit ovat pieniä upotettuja systeemejä, jotka yleensä liittyvät muihin komponentteihin ja rakenteisiin (Maluf ja Williams 2004). Sensoreiden avulla on mahdollista muuntaa oikean maailman signaaleja yhdestä muodosta toiseen (Judy 2001).

On varsin ilmiselvää, että älykäs ympäristö on riippuvainen sensoreista ja sensorit ylipäänsä mahdollistavat älykkään ympäristön olemassa olon. Jotta ympäristön kanssa voidaan olla missään vuorovaikutuksessa, tulee siitä voida kerätä tietoa (Lewis ym. 2004). Sensoreiden ja langattoman kommunikaation kehityksen myötä pystyttiin yhdistämään useampia sensoreita solmuiksi (eng. node). Näistä yksittäisistä solmuista ollaan siten pystytty muodostamaan kokonaisia sensoriverkkoja. Sensoriverkon solmut voidaan sijoittaa joko samaan ympäristöön mitattavan asian kanssa, tai sen välittömään läheisyyteen. Sensorisolmuja voidaan esimerkiksi käyttää jatkuvaan havainnointiin, tapahtumien tarkkailuun tai sijainnin havainnointiin. Sensoriverkot voivat itsessään koostua monenlaisista eri sensoreista, esimerkiksi lämpötila-, kosteus-, valoisuus-, tai melusensoreista. (Akyildiz ym. 2002)

Kuten missä tahansa muussakin verkossa, myös sensoriverkoissa perusvaatimuksina on tietyn suoritustehon ja laadun varmistaminen kommunikaation suhteen. Riippuen ympäristöstä tai esimerkiksi sovelluskohteesta vaihtoehdot eri topologioille verkon rakentamisen suhteen ovat esimerkiksi seuraavat:

- tähti
- rengas
- väylä
- täysin yhdistetty ja
- verkko.

Tähtimallissa kaikki verkon solmut ovat yhteydessä yhteen keskitettyyn hubiin. Hubin täy-

tyy käsitellä kaikki kommunikaatio ja se vaatii hyvän laskentatehon. Jos yksi verkon solmuista kaatuu, se ei vaikuta muun verkon toimintaan, mutta jos hubi kaatuu, niin koko verkko on käyttökelvoton. Rengasmallissa ei ole yhtä johtavaa solmua, vaan kaikki solmut hoitavat samoja tehtäviä tasavertaisesti. Viestit kulkevat yhteen suuntaan ja jos yksikin solmu verkossa tippuu pois, katkeaa koko verkon kommunikaatio. Väylämallissa viestit välitetään väylää pitkin kaikille verkon solmuille. Välyämallissa jokainen solmu kuuntelee viestejä ja yksikään solmu ei ole vastuussa viestin uudelleen lähettämisestä. Täysin yhdistetyt verkot kärsivät NP-kompleksisuudesta (*nondeterministic polynomial time*). Kompleksisuusteoriassa NP-kompleksisuus ongelmat ovat ongelmia, joihin ei löydy ratkaisua polynomisessa ajassa, ehkäpä tunnetuin tällainen ongelma on kauppamatkustajan ongelma (De Jong, Spears ym. 1989). Kun täysin yhdistettyyn verkkoon lisätään uusia solmuja, yhteyksien määrä kasvaa eksponentiaalisesti. Verkkomallissa tavanomaisesti lähetys onnistuu vain lähimmälle naapurisolmulle. Verkkomallit ovat hyviä suuren skaalan verkoille, jotka levittäytyvät isolle alueelle. (Lewis ym. 2004)

Kuten muissakin tietotekniikan sovelluksissa, on sensoriverkoissakin tärkeää huolehtia hyvästä tietoturvasta. Koska pääsääntöisesti sensoriverkkojen kommunikaatio tapahtuu verkkoyhteyden yli, ovat nekin alttiita hyökkäyksille.

Palvelunestohyökkäyksessä (DoS, Denial of Service) jokin sensoriverkon solmuista pyritään tekemään saavuttamattomaksi käyttäjille esimerkiksi lähettämällä jatkuvalla syötöllä paketteja solmulta toiselle, jolloin vastaanottava solmu ei kykene palvelemaan muita. Palvelunestohyökkäys on mahdollista myös toteuttaa, jos päästään käsiksi fyysiseen kerrokseen, jumittamalla tai peukaloimalla paketteja. Palvelunestohyökkäyksiä voidaan estää vahvalla autentikaatiolla ja identifikaatiolla. Myös lähetettyyn dataan voidaan kohdistaa hyökkäyksiä esimerkiksi väärentämällä tai muuntamalla sitä. Hyökkääjä voi myös saada laittomasti käsiinsä useamman yhtä solmua koskevan identiteetin (Sybil attack). Hyökkääjä voi näin vaikuttaa esimerkiksi reititysmekanismeihin. (Pandey ja Tripathi 2010)

Sensoriverkoille on toteutettu joitakin arkkitehtuureja tietoturvan parantamiseksi. SPINS on yksi tällainen. SPINS koostuu kahdesta palasesta, jotka ovat SNEP (Security Network Encryption Protocol) ja µTESLA. Molemmat toimivat TinyOS nimisen käyttöjärjestelmän päällä. SNEP tarjoaa salauksen ja autentikoinnin. µTESLAa käytetään lähetetyn datan autenti-

koinnissa. TinySec on SNEPin pohjalta rakennettu linkkikerroksen tietoturva-arkkitehtuuri sensoriverkoille. TinySecistä on saatavilla kahta eri variaatiota, TinySec-Auth ja TinySec-AE. Ensimmäinen tarjoaa vain autentikoinnin, kun taas jälkimmäinen tarjoaa sekä autentikoinnin, että salauksen. LEAP (Localized Encryption and Authentication Protocol) on salausavainten hallintaan tarkoitettu protokolla sensoriverkoille. LEAP tarjoaa neljää eri ratkaisua eri turvallisuusvaatimuksille. Eri avaima on mahdollista luoda yksilöllisesti, ryhmän kesken, suurempien rykelmien kesken, sekä pareittain jaetusti. Jokainen sensoriverkon solmu luo oman avainketjunsa ja lähettää tämän ketjun ensimmäisen avaimen naapurisolmuille. Tämä avain on nyt AUTH-avain ja joka kerta kun solmu lähettää viestin, se kiinnittää viestiin seuraavan avainketjussa olevan avaimen. Avaimet ovat luettavissa käänteisessä järjestyksessä niiden luontiin nähden ja vastaanottaja voi näin varmistaa vastanotetun viestin luotettavuuden. (Boyle ja Newe 2008)

2.5 Pilvipalvelut

Käyttöomaisuusinvestoinnit (Capital Expenditure, CapEx) ovat niitä investointeja, joita yritys käyttää fyysisten varojensa päivittämiseen ja ylläpitämiseen (“Capital Expenditure (CapEx)” 2021). IT-alan yrityksissä tällaiset investoinnit ja ylläpitotehtävät liittyvät esimerkiksi palvelinkoneisiin. Kun sovellus tai verkkosivu saa pienessä ajassa suuremman määrän uusia käyttäjiä, saatetaan joutua investoimeen uusiin palvelimiin laskentakapasiteetin kasvattamiseksi. Jos taas jokin palvelinkoneista kaatuu tai pahimmassa tapauksessa hajoaa täysin, täytyy se korjata tai hankkia tilalle uusi. Tällaiset sijoitukset it-laitteisiin eivät välttämättä ole yritykselle aina hyväksi, koska it-laitteiden arvo laskee nopeasti ja tällaisten laitteiden hankkiminen ja ylläpito saattaa olla kallistakin.

Pilvipalvelut tai pilvilaskenta on paradigma, joka on noussut hyvin vahvaan suosioon. Pilvipalveluille on olemassa monenlaisia määritelmiä, mutta ydinominaisuutena on IT-infrastruktuurin ja palveluiden tarjoaminen helposti skaalautuvassa muodossa (Stanoevska-Slabeva ja Wozniak 2010). Laitteisto ja ohjelmisto on abstrahoitu virtualisoinnin avulla ja nämä tarjotaan jonkin määritellyn rajapinnan kautta. Näin voidaan varmistaa tehokas joustavuus ja skaalautuvuus ja loppukäyttäjän on helppo muokata resurssejaan (Stanoevska-Slabeva ja Wozniak 2010).

Datan tuottajan ja loppukäyttäjän näkökulmasta merkittäviä hyötyjä ovat esimerkiksi kulujen laskeminen ja maksaminen vain siitä mitä käyttää, ohjelmistojen päivittämisen tarpeen laskeminen, tarve palvelin- ja datanvarastointitiloille laskee tai katoaa kokonaan, sekä skaalautuvuus. Kulujen laskeminen on varsin ilmeistä, sillä kun laitteisto, tallennustila ja laskentateho itsessään tarjotaan jostakin muualta, ei investointeja tarvitse enää tehdä laitteistoon paikanpäällä, esimerkiksi yrityksen omissa tiloissa. Maksaminen vain siitä mitä käyttää on myös yksi tekijä, joka laskee kuluja. Kun esimerkiksi yritys ostaa lisää tallennuslaitteistoa, sen lisäksi, että maksetaan käytetystä tallennustilasta, maksetaan myös siitä tilasta, joka on käyttämättömänä. Ohjelmistojen päivitys on tehtävä, jota täytyy suorittaa säännöllisesti. Pilvipalveluissa ohjelmistojen päivitys ja muu ylläpito on pilvipalvelun resurssientarjoajan vastuulla ja näin ollen se siirtyy pois esimerkiksi yritykseltä. Palvelin- ja datanvarastointitilojen tarve laskee, sillä nämä tarjotaan pilvipalvelun palveluntarjoajan puolesta ja tästä seurauksena taas on kulujen laskeminen entisestään, sillä rahaa ei tarvitse enää käyttää suurempien toimitilojen vuokraamiseen, eikä esimerkiksi palvelinlaitteiston ylläpitämiseen. Skaalautuvuus tarkoittaa järjestelmän kykyä suoriutua hyvin erilaisten rasitustilanteiden alla. Pilvipalvelujen yksi etu on skaalautuvuuden parantaminen. Pilvipalvelujen avulla esimerkiksi palvelinten suoritustehoa on helppo skaalata esimerkiksi äkkiä kasvavan käyttäjämäärän mukaan.(Prince 2011)

Pilvipalvelujen tarkoituksena on siis tarjota erilaisia resursseja liittyen esimerkiksi inrastruktuuriin, erilaisiin alustoihin tai ohjelmistoihin, palveluina (X-as-a-Service). IaaS (Infrastructure as a Service) tarjoaa laskentatehoa tai tallennustilaa palveluna. Palvelua ei tarjota itse laitteistona, vaan yleensä tarjotaan virtualisoitu infrastruktuuri. PaaS (Platform as a Service) tarjoaa kehitysalustan esimerkiksi ohjelmistokehittäjille. PaaS mahdollistaa sovellusten rakentamisen ilman tarvetta huolehtia alla olevan laitteiston infrastruktuurista. PaaS yleensä myös huolehtii "itsenäisesti"esimerkiksi tarvittavasta skaalautuvuudesta. SaaS (Software as a Service) on ohjelmisto, jonka tarjoaa ja jota hoitaa pilvipalveluntarjoaja. SaaSin käyttö vaatii vähän, jos ollenkaan, tietoa alla toimivasta alustasta tai infrastruktuurista ja sitä varten tulee ainoastaan tarjota haluttu data. (Stanoevska-Slabeva ja Wozniak 2010)

Rimal, Choi ja Lumb (2009) ovat määritelleet pilvipalveluille taksonomian, joka muodostuu seuraavanlaisista osista:

- pilviarkkitehtuuri
- virtualisoinnin hoitaminen
- palvelut
- vikasieto
- turvallisuus ja
- haasteet

Kun yritys tai muu organisaatio valitsee itselleen parhaiten sopivimman pilviarkkitehtuurin, valittavana on kolme eri vaihtoehtoa. Yksityisessä mallissa organisaatio hallitsee itse datansa ja muut prosessit. Julkisessa mallissa jokin kolmas osapuoli tarjoaa resurssit verkon kautta jonkin palvelun tai muun sovelluksen avulla. Hybridimalli yhdistää kaksi aiemmin mainittua. Virtualisoinnin tarkoituksena on abstrahoida fyysinen laitteisto ja käyttöjärjestelmä toisistaan. Kun nämä resurssit erotetaan toisistaan, mahdollistetaan kulujen pienentäminen, ketteryys ja joustavuus. Palvelut, joita voidaan tarjota pilvipalvelujen kautta ovat SaaS (Software as a Service), PaaS (Platform as a Service) ja IaaS (Infrastructure as a Service). Vikasietoon liittyy toimenpiteet, miten häiriötilanteet hoidetaan. Yleensä häiriöin sattuessa esimerkiksi sovelluksen suoritus voidaan siirtää jollekin varainstanssille. Näin pystytään varmistamaan palvelujen saatavuus myös poikkeustiloissa. Turvallisuus on tärkeä prioriteetti pilvipalveluiden kohdalla. Pilvipalvelut sisältävät yleensä sensitiivistäkin dataa yrityksistä, organisaatioista tai näiden asiakkaista. Haasteista esiin nousee kuorman jakaminen, yhteensopivuus ja skaalautuva datan varastointi. Kuorman jakamisen järkevä toteuttaminen on tärkeää, sillä se vähentää resurssien kulutusta, vähentää stressiä itse laitteistoon ja kokonaisuudessa voi mahdollistaa laitteiston pidemmän toimintaiän. Yhteensopivuus tarkoittaa sitä, että esimerkiksi eri sovellukset voidaan helposti yhdistää eri pilviympäristöjen välillä. Skaalautuva ja turvallinen datan varastointi on tärkeä seikka. Yleensä pilvipalveluiden asiakkaat antavat datansa pilvipalvelun käyttöön juurikaan murehtimatta, minne ja miten data varastoidaan.

Dash, Mohapatra ja Pattnaik (2010) mukaan sensoriverkkojen ja pilvilaskennan yhdistäminen helpottaa sensoriverkoista kerätyn datan analysointia. Yhdistelmän avulla kyetään myös tarjoamaan dataa tai tapahtumien hallintaa pilvipalveluille tuttuun tapaan palveluna verkon yli. Esimerkiksi liikenteen seurannassa, sään ennustamisessa tai terveyden hoidossa tuotetaan valtavat määrät jopa sensitiivistäkin dataa, jonka prosessointi vaatii hyvän turvallisuu-

den ja laskenta tehon ja nämä elementit pystytään tarjoamaan pilvipalveluiden avulla.

Kolme ehkäpä tunnetuinta pilvipalvelua ovat Googlen Google App Engine (GAE), Amazonin Amazon Web Service (AWS) ja Microsoftin Microsoft Azure. GAE julkaistiin 2008, AWS 2006-2007 ja Azure 2008. Pilvipalveluissa käytettävät tekniikat ovat virtualisointi ja niin sanottu hiekkalaatikkoistaminen. Virtualisointi tarjoaa enemmän joustavuutta ja yhteensopivuutta eri ohjelmistojen ja sovellusten välillä, kun taas hiekkalaatikkoistaminen asettaa enemmän rajoituksia ohjelmointikielille, mutta vähentää abstrahoimisen tarvetta. AWS käyttää puhdasta virtualisointia pilvipalvelunsa kanssa, GAE käyttää puhdasta hiekkalaatikkoistamista ja Azure käyttää näiden kahden yhdistelmää. (Qian ym. 2009)

3 Reunalaskenta

Tässä luvussa käsitellään reunalaskentaa, mitä se on ja mitä hyötyjä sillä voidaan saavuttaa. Lisäksi käydään läpi tutkimuksia reunalaskennasta, sekä mitä on reunaäly.

3.1 Mitä reunalaskenta on?

Niin sanottu flash crowd -ongelma on ongelma, jossa jatkuvat pyynnöt esimerkiksi jollekin tietylle verkkosivulle ruuhkauttavat sen ja näin ollen verkkosivun kyky palvella jokaista pyyntöä laskee. Tämä ilmenee esimerkiksi viivästyneenä vastausaikana tai pahimmillaan koko sivun kaatumisena (Dilley ym. 2002). Tämän ongelman ehkäisemiseksi, Akamai Technologies kehitti 1990 -luvun loppupuolella sisällönjakeluverkon (Content Delivery Network). Sisällönjakeluverkon ideana on siirtyä pois keskitetystä verkkopalvelujen tarjoamisesta ja hajauttaa palvelujen tarjonta useammalle, verkon "reunalla"sijaitevalle palvelimelle. Näin toimittaessa pystytään vähentämään rasitusta verkkosivun varsinaiselle infrastruktuurille ja palvelemaan jokaista pyyntöä nopeammin (Dilley ym. 2002).

IoT-laitteiden kehitys on mullistanut kykymme tarkkailla ympäristöä ja kerätä siitä tietoa. Meitä ympäröi alati kasvava määrä sensoreita, jotka keräävät suuret määrät dataa. Pilvipalveluiden käyttämisestä on kiistämättä tullut yksi suosituimmista keinoista varastoida ja prosessoida tätä dataa, eikä tämä trendi todennäköisesti tule enää laantumaan. Pilvipalveluiden ominaisuuksien, kuten skaalautuvuuden ja saatavuuden vuoksi niiden käyttö on järkevä ratkaisu kerätyn datan varastointiin ja prosessointiin (Premsankar, Di Francesco ja Taleb 2018). Tähän kuitenkin liittyy hyvin samankaltainen ongelma, joka sivuaa flash crowd -ongelmaa. Pilvipalveluresurssit sijaitsevat kaukana datakeskuksissa, josta aiheutuu varsinkin datamäärien yhä kasvaessa viivettä ja rasitusta datansiirtokanaville (Premsankar, Di Francesco ja Taleb 2018).

Reunalaskennan juuret ovat sisällönjakeluverkoissa ja pähkinän kuoressa sen idea on varsin samanlainen, laajentamalla sisällönjakeluverkko koskemaan myös pilvipalvelimia (Satyanarayanan 2017). Reunalaskenta on paradigma, jonka ideana on sijoittaa datan prosessoinnin resursseja lähemmäs käyttäjiä tai dataa tuottavia sensoreita (Satyanarayanan 2017). Reuna-

laskentaa suorittavaa laitetta nimitetään reunalaitteeksi (edge device) ja tällainen reunalaite voi olla esimerkiksi taskussa mukana kulkeva älypuhelin tai jokin paikallaan pysyvä mikrokontrolleri sijoitettuna jonnekin ympäristöön (Shi ja Dustdar 2016). Näin ollen reunalaskentaa on mahdollista upottaa minne tahansa ja missä muodossa tahansa. Reunalaskenta ei kuitenkaan poista pilvipalveluiden tarvetta, vaan lähinnä vain lisää yhden kerroksen lähemmäksi datan lähdettä (Reiter, Prünster ja Zefferer 2017). Reunalaskennan arkkitehtuurin puolesta Caprolu ym. (2019) mukaan ei ole olemassa mitään universaalia paradigmaa tai standardia, jolla se oltaisiin määritelty vaan lähinnä vain useita erilaisia selityksiä, jotka sivuavat toisiaan hyvin paljon.

3.2 Reunalaskennan hyödyt

Gedeon ym. (2019) mukaan reunalaskennasta saatavia hyötyjä ovat mm. pienempi viive, pienempi kaistanleveyden kulutus, peinempi energian kulutus ja energian säästäminen, sekä parantunut tietoturva. Monilla verkkoyhteyteen nojaavilla sovelluksilla on tarkat määritteet viiveen suhteen. Pilvilaskennan resurssit ovat yleensä maantieteellisesti hajautettuja ja saattavat sijaita kaukanakin datan lähteestä ja tämä kontribuoi kasvaneeseen viiveeseen. Tämä lisää viivettä, joka puolestaan luo tarpeen datan prosessoinnille lähempänä sen lähdettä.

Kun osa kerätystä datasta prosessoidaan lähellä sen lähdettä, pystytään vähentämään lähetettävän datan määrää ja näin säästää kaistanleveyttä. Gedeon ym. (2019) mukaan kerätyllä raa'alla datalla on oikeastaan hyvin vähän merkitystä. Yksittäiset sensoreiden lukemat eivät yleensä ole kovinkaan mielenkiintoisia, vaan sensoridataa prosessoimalla kerätyt tulokset. Tyypillisessä skenaariossa kaikki data lähetettäisiin sellaisenaan pilveen. Tämä kuitenkin rasittaa verkkoa turhan paljon.

Gedeon ym. (2019) mukaan pienempi energian kulutus ja energiatehokkuus ovat seikkoja, joiden parantaminen on hyvin tärkeää niin mobiililaitteissa, kuin sensorijärjestelmissäkin. Laskennallisesti intensiivisten tehtävien suorittaminen laitteilla on hyvin kuluttavaa laitteiden akuille. Lisäksi pieni koko on yksi akun tehoa pienentävä tekijä. Laskennan siirtäminen reunalle on yksi ratkaiseva tekijä näiden ongelmien ratkaisemisessa.

Gedeon ym. (2019) pilvipalveluita käytettäessä datan prosessointiin liittyen on yleensä var-

sin vähän valinnanvaraa ja tämä luo huolen riittävästä tietoturvasta. Reunalaskentaa suorittavien reunalaitteiden on mahdollista toimia eräänlaisina välikappaleina datan lähteen ja pilvipalvelun välillä. Reunalaitteisiin voidaan lisätä tietoturvaa tehostavia mekanismeja datan prosessointiketjun alkuvaiheessa. Tietoturvaa voidaan reunalaitteilla lisätä esimerkiksi poistamalla datasta yksilöiviä tietoja.

3.3 Reunaäly

Viime vuosina tekoälyn merkitys eri toimialoilla on kasvanut merkittävästi. Erilaisiin tekoälyn sovelluskohteisiin törmää lähes kaikkialla, esimerkiksi ajoneuvoissa ajamisen avustukseen tarkoitetuissa laitteistossa ja teollisuus 4.0:ssa (Industry 4.0) (Brandalero ym. 2021). Kyberturvallisuuden alalla tekoälyä voidaan hyödyntää tekemään älykkäitä ratkaisuja tietoturvaan liittyen ja sen avulla voidaan rakentaa älykkäitä ja automatisoituja tietoturvajärjestelmiä (Sarker, Furhad ja Nowrozy 2021). Lääketieteen saralla tekoälyä voidaan hyödyntää muun muassa uusien lääkkeiden etsimisessä, lääkeaineiden testauksessa, potilaiden hoidossa, sairaalahenkilökuntaa ja potilaita avustavissa roboteissa ja esimerkiksi äitiyskuolemien ja synnytyksen jälkeisten ongelmien ehkäisemisessä (Shaheen 2021). Tekoälylle on löydetty myös sovelluskohteita COVID-19 pandemiaan liittyen muun muassa tartuntojen aikaisessa havaitsemisessa ja diagnosoinnissa, lääkkeiden ja rokotteiden kehittämisessä, itse taudin ehkäisyssä ja esimerkiksi lääkintähenkilökunnan työmäärän vähentämisessä (Vaishya ym. 2020). Sovelluskohteita siis riittää paljon.

Reunaäly on jälleen yksi alue, jossa tekoälyä on mahdollista hyödyntää. Sovelluksissa, joiden toiminta vaatii älyä, on tekoäly puoli yleisimmin hoidettu pilvialustalla, mutta joissakin tapauksissa tämä kuitenkin tuottaa ongelmia. Datan lähettäminen prosessoitavaksi pilvipalvelulle on liian hidasta sovelluksissa, jotka vaativat nopeaa datan prosessointia ja sen perusteella toimintojen tekemistä. Tietoturva tulee myöskin kysymykseen, kun lähetetään suuria määriä dataa mahdollisesti pitkänkin matkan päähän. Lisäksi kaikkialle ei välttämättä ole edes mahdollista saada vakaata internet yhteyttä. Nämä kaikki seikat huomioon ottaen on oleellista tuoda älykkyyttä reunalaitteisiin. (Lee, Tsung ja Wu 2018)

Jotta älyä voidaan lisätä yhtään mihinkään sovellukseen, tulee sitä varten ensin kouluttaa

tekoälymalli. Tämän mallin kouluttaminen on erittäin aikaa vievää ja vaatii paljon laskentatehoa. Yleensä mallin kouluttamiseen vaaditaan runsaat määrät dataa, jotta mallista saadaan mahdollisimman tarkka ja hyvin soveltuva sille tarkoitettuun tehtävään. Parhaimmillaan tekoälymallilla voi olla syötteenään jopa miljoonia eri arvoja (Shi ym. 2020). Reunalaitteiden kanssa ilmenevät ongelmat liittyvät suoritustehoon, puutteisiin yhdenmukaisuudessa esimerkiksi reunalaitteiden laitteistossa, verkoissa ja virrankulutuksessa, sekä yksityisyys- ja tietoturvarajoituksissa (Shi ym. 2020).

Mahdollisista haasteista huolimatta tekoälyn tuominen reunalle on silti tavoiteltava päämäärä. Reunaälyn avulla on mahdollista saada kerätystä datasta varsin tehokkaasti kaikki mahdollinen hyöty irti havaitsemalla yhdenmukaisuuksia tai anomalioita datasta. Reunalaskenta tuo mukanaan reunaälyn mahdollistamisen ja näiden kahden "yhteistyö"tulee korostumaan yhä enemmän tulevaisuudessa, kun ympäristöstämme kerätyn datan määrä kasvaa ja sen lähetys ja prosessointi pilvipalvelimilla käy yhä raskaammaksi vaatien paljon kaistanleveyttä ja suoritustehoa. (Zhou ym. 2019)

Keskittyneisyys on klassinen tapa ajatella tekoälyä. Tekoälyn "koulutusprosessi"pidetään lokaalina ja erillään tekoälyn päättelyprosessista. Tämänlaisessa menettelyssä on yleensä ongelmana se, että koulutettu tekoälymalli on taipuvaisempi virhepäätelmiin, koska se saattaa vääristyä koulutusdatan luonteen takia. Oikean maailman (sensoreiden keräämä ja ihmisten tuottama) data on monitasoista ja hyvin vaihtelevaa ja olisi toivottavaa, että tekoälymalli kykenisi tekemään oikeat päättelyt juuri tämän kaltaisen datan kanssa. Tekoäly saattaa hyvin usein olla vääristynyt koulutusdatasetin suhteen, jonka vuoksi se saattaa tuottaa virhepäätelmiä oikean maailman datan kanssa, joka on hyvin monimuotoista. Reunalaskennan hajautuneisuuden ansiosta näihin ongelmiin voidaan pystyä vastaamaan. (Park ym. 2019)

4 Reunaälyn ja reunalaskennan tutkimuksia ja sovelluksia

Tässä luvussa käsitellään reunalaskentaan ja reunaälyyn liittyviä tutkimuksia ja sovelluksia.

4.1 Kuvantunnistus reunalaskentasovelluksissa

Artikkelissaan *Deep learning-based multiple object visual tracking on embedded system for iot and mobile edge computing applications*, Blanco-Filgueira ym. (2019) tutkivat tapaa, jolla voidaan implementoida alhaisen virran kuvantunnistusjärjestelmä NVIDIA Jetson TX2 kehitysalustalle. Kuvan tunnistus ja esimerkiksi tunnistuksen perusteella tehtävä luokittelu ympäristöstä, on monelle sovellukselle tärkeä toimenpide. Tällaisessa toiminnassa datamäärät ovat yleensä varsin suuria ja toimenpiteiden suorittaminen tulee olla reaaliaikaista. Pelkän pilvipalvelimen käyttö tähän ei siis yksinään riitä. Kuitenkin käytettäessä reunalaitteita, on virrankulutus yksi seikka, joka tulee ottaa huomioon.

Tutkimuksessa yhdistettiin keksi eri algoritmia. HO-PBAS (Hardware-Oriented PBAS) algoritmi liikkuvien objektien havaitsemiseen ja GOTURN algoritmi objektien seurantaan. Tutkimuksen mukaan kumpikin algoritmi yksinään toimii varsin moitteettomasti, mutta yhdessä toimiakseen tuli tehdä hieman jatkokehitystä. Tutkimuksen tuloksena saatiin kehitettyä kuvantunnistus- ja seurantajärjestelmä NVIDIA Jetson TX2 alustalle, joka kulutti virtaa parhaimmillaankin vain noin 12 wattia.

4.2 Reunalaskenta mobiililaitteissa

Tutkimuksessaan *Hybrid Mobile Edge Computing: Unleashing the Full Potential of Edge Computing in Mobile Device Use Cases*, Reiter, Prünster ja Zefferer (2017) ehdottavat mobiililaitteille uudenlaista teknologiaa, HMEC (Hybrid Mobile Edge Computing), joka hyödyntää reunalaskentaa. Tutkimuksessa kerrotaan tämän ehdotetun teknologian sallivan mm. joustavan lähellä ja kaukana olevien resurssien käytön ja yhteentoimivuuden erilaisten ympäristöjen välillä samalla säilyttäen käyttäjien yksityisyyden ja datan turvallisen säilyttämisen.

Tutkimuksessa kerrotaan, että mobiililaitteiden prosessori on yksi suurimmista mobiililaitteiden virtaa syövistä komponenteista. Virran kulutuksen vähentämiseksi mobiililaitteiden laskentataakkaa pyritään helpottamaan siirtämällä datan prosessointia pilvipalvelimille. Tämä kuitenkin johtaa viivästyneeseen kommunikaatioon mobiililaitteen ja pilvipalvelimen välillä. Tutkimuksessa kerrotaan, että mobiililaitteiden yleisimmin käyttämä yhteys on UMTS-verkko (Universal Mobile Telecommunication System). Tässä verkossa pahimpina ruuhkaaikoina viive saattaa olla jopa yli 100 millisekuntia.

HMEC teknologiaa varten luotu malli kattaa tutkimuksen mukaan parannellun ja joustavamman arkkitehtuurin, sekä useamman laskentalaitteen integroinnin. Arkkitehtuurin joustavuuden mahdollistaa muun muassa se, että reunalaitteet ja mobiililaitteet pystyvät tehokkaasti kommunikoimaan keskenään, joka taas osaltaan poistaa staattisen pilvipalvelinyhteyden tarpeen. Useiden reunalaitteiden integrointiin HMEC mallissa käytetään vertaisverkkomallia (peer-to-peer). Vertaisverkon tehtävänä on toimia palvelunjakajana sen sisällä oleville laitteille, reunalaitteiden tarjotessa laskentatehoaan ja mobiililaitteiden käyttäessä sitä. Tutkimuksen mukaan tuloksena ehdotetulla teknologialla saatiin laskettua virrankulutusta ja kasvatettua suorituskykyä.

4.3 Reunalaskenta kaupunkiympäristössä

Tutkimuksessaan *Demo Abstract: Crowd analysis with infrared sensor arrays on the smart city edge*, Bock ym. (2019) tutkivat reunalaskennan soveltamista kaupunkiympäristössä. Tutkimuksessa esitellään sovellus, jolla voidaan nopeuttaa datan prosessointia, vähentää viivettä, sekä pienentää kaistanleveyden kulutusta. Sovellus toteuttiin infrapunasensoreiden avulla, jotka analysoivat ihmisjoukkoja. Data lähetetään sensroreilta yhdyskäytävälle, joka prosessoi datan ja antaa arvion ihmisjoukon koosta.

Tutkimuksessa käytettiin CityLab testausalustaa, joka on osa City of Things -ohjelmaa. CityLab testausalusta sijaitsee Antwerpenin kaupungissa, Pohjois-Belgiassa. CityLab koostuu yhdyskäytävistä, jotka on luokiteltu kolmeen kategoriaan: sisäyksikkö, aktiivinen ulkoyksikkö, sekä passiivinen ulkoyksikkö.

Tutkimuksessa siis luotiin sovellus, jolla on mahdollista estimoida väkijoukon suuruutta.

Sovelluksessa käytettiin infrapunasensoreita. Sensorit järjestettiin riviin, joista jokainen rivi sisältää kolme sensoriruudukkoa. Nämä sensoriruudukot kommunikoivat yhdyskäytävän kanssa, joka prosessoi datan. Käyttämällä reunalaskentaa datan prosessoinnissa, pystyttiin tutkimuksen mukaan vähentämään viivettä, sekä datan määrää lähetyksessä ja säästämään kaistanleveyttä.

4.4 Reunalskenta IIoT ympäristössä

Artikkelissaan *Edge Computing in Industrial Internet of Things: Architecture, Advances and Challenges*, Qiu ym. (2020) tutkivat reunalaskennan käyttöä IIoT-ympäristössä (Industrial Internet of Things, esineiden internet teollisuudessa). Artikkelissa kerrotaan, että tuomalla reunalaskentaa mukaan IIoT-ympäristöön voidaan merkittävästi lyhentää datan prosessointiaikoija, säästää kaistan käyttöä langattomassa kommunikaatiossa, sekä mahdollisesti lisätä tietoturvallisuutta. IIoT-ympäristö sisältää paljon mm. pikaista reagoimista vaativia toimia, joihin yksinään datan vieminen pilvipalvelimelle ja sen prosessointi siellä eivät riitä. Siirtämällä osan datasta reunalaskennan piiriin, voidaan tehokkaasti ehkäistä tällaisia ongelmia.

Reunalaskennan lisääminen IIoT-ympäristöön antaa mm. seuraavia hyötyjä:

- vähentää kuluja
- parantaa järjestelmien suorituskykyä
- parantaa tietoturvaa.

Kun suuret määrät dataa lähetetään suoraan pilvipalvelimelle mm. nopea tiedonsiirto ja yleisesti datan siirtäminen paikasta toiseen käyvät haastavaksi. Reunalaskennan avulla voidaan merkittävästi pienentää datamääriä, jolloin ei vaadita niin paljon esimerkiksi kaistanleveyttä ja näin ollen myös kulut laskevat. Pienentämällä vaadittavaa kaistanleveyttä pystytään myös parantamaan yleisesti järjestelmien suorituskykyä. Kun data prosessoidaan lähellä datan lähdettä, pienenee riski suurien datamäärien vuotamiselle lähetyksen aikana ja näin reunalaskennalla on mahdollista parantaa myös tietoturvaa.

Artikkelin mukaan reunalaskentaan IIoT-ympäristössä liittyy myös paljon haasteita. IIoT-verkot ovat tällä hetkellä kahdentyyppisiä, langattomia ja langallisia. Langallinen vaihtoeh-

to tarjoaa vakaata yhteyttä, kun taas langaton vaihtoehto on edelleen kehittymässä, antaen huonomman tiedonsiirtonopeuden ja olemalla epävakaampi. IIoT-ympäristöön lisätty reunalaskenta vaatii nopeaa ja vakaata tiedonsiirtoa. Nykyiset teknologiat saattavat olla vielä puutteellisia, mutta 5G yhteyksien yleistyessä ja parantuessa tämän todennäköisesti tulee muuttumaan.

Mitä todennäköisimmin IIoT-ympäristöissä on useampi reunalaskentaa suorittava alusta ja on erityisen tärkeä harkita tarkkaan, miten näiden alustojen välille jaetaan järkevästi mm. datan prosessointi. Kun tämä seikka on hyvin suunniteltu, saadaan paras hyöty irti. Jos tätä seikkaa ei oteta kunnolla huomioon, saatta seurauksena reunalaitteiden rajallisesta muistista ja laskenta tehosta johtuen olla kasvanut viive, turha kaistan käyttö ja alentunut suoritusteho.

Artikkelin mukaan reunalskennalle on jo olemassa muutamia sovelluksia. PHM on uudenlainen ratkaisu, jossa jonkin järjestelmän eri komponenteista kerättyä dataa käytetään analysoimaan järjestelmän kokonaistoimintakuntoa. Reunalaskennan ja PHM ratkaisun yhteiskäyttöä hyödynnetään mm. rautateiden kunnon tarkkailussa. Reunalaskentaa voidaan hyödyntää myös älykkäissä sähköverkoissa. Myös logistiikan alalla reunalaskennan avulla on saatavissa hyötyjä. Tuotteita kuljettavan ajoneuvon kulkureitistä, kunnosta tai hätätilanteessa tarvittavista toimenpiteistä kerättävä data voidaan kerätä ja prosessoida jo esimerkiksi itse ajoneuvossa reunalaskennan avulla.

4.5 Reunalaskennan hyödyntäminen MQTT pohjaisessa väliohjelmistossa

Tutkimuksessaan *Emma: Distributed qos-aware mqtt middleware for edge computing applications*, Rausch, Nastic ja Dustdar (2018) tutkivat verkon palvelun laadun (Quality of Service, QoS) parantamista reunalskennan avulla. Tutkimuksessa toteutettiin "EMMA"niminen väliohjelmista, joka monitoroi jatkuvasti verkon palvelun laatua ja orkestroi MQTT-protokollalla toimivia välittäjiä. Verkon käyttäjät ohjataan näille välittäjille ja näin saadaan parannettua palvelun laatua kyseisessä verkossa.

Tutkimuksessa kerrotaan monien nykypäivän IoT-ratkaisujen olevan tiukan laadunvalvonnan alla hyvän palveluntarjonnan suhteen. Tyypillistä on, että laitteiden kommunikaatio no-

jaa paljon pilvipalveluiden varaan. Pilvipalvelut eivät kuitenkaan kaikissa skenaarioissa kykene kattamaan tehokasta palveluntarjontaa, joten on järkevämpää siirtää laskentaa lähemmäksi käyttäjiä ja datan lähdettä. IoT:n ja reunalskennan myötä tulee myös haasteita. Näissä sovelluksissa verkon topologia on hyvin dynaaminen, käyttäjät ja laitteet ovat liikkuvia ja liittyvät ja lähtevät verkosta satunnaisesti ja miten tahansa. Lisäksi reunalaitteiden hajanaisuus lisää vielä ennestään haasteita. Tutkimuksessa kehitetty EMMA-väliohjelmisto pyrkii osaltaan vastaamaan näihin haasteisiin. EMMAn avulle on mahdollista tarkkailla jatkuvasti verkon palvelunlaatua ja tarvittaessa orkestroida joukko MQTT pohjaisia yhdyskäytäviä ja välittäjiä. Tutkimuksen mukaan näin pystyttiin parantamaan palvelun laatua verkossa.

EMMA on rakennettu MQTT -protokollan pohjalle. MQTT soveltuu erinomaisen hyvin alhaisen virran ja kapean kaistanleveyden ympäristöihin, ja siksi se on saavuttanut suosiota IoT-sovelluksissa. Yksinkertaistettuna EMMA kostuu neljästä osasta:

- yhdyskäytävästä
- välittäjistä
- kontrollerista ja
- verkontarkkailuprotokollasta.

Käyttäjät yhdistyvät yhdyskäytävään. Yhdyskäytävä toimii käänteisenä proxyna, ja yhdistää käyttäjät välittäjille. Kontrolleri monitoroi verkon tilaa, toimii rekisterinä ja järjestelmän orkestroijana.

Tutkimuksessa suoritettiin testiskenaario EMMAlle oikean maailman ympärstössä. Tuloksena saatiin pienempi viive kommunikaatiossa toisiaan lähellä olevien laitteiden välillä, samalla kyeten kuitenkin tehokkaasti levittämään kommunikaatiota laajemmalle alueelle maantieteelliesti hajallaan olevien sijaintien välillä. Järjestelmän avulle pystyttiin vastaamaan käyttäjien liikkuvuuden luomiin haasteisiin samalla varmistaen dynaaminen välittäjien varaaminen käyttäjille ja rasituksen tasaaminen.

5 Langaton kommunikaatio

Tässä luvussa käsitellään langattomia kommunikaatiomenetelmiä.

5.1 Mitä on langaton kommunikaatio?

Hyvin usein langaton kommunikaatio käsitetään sen nykyaikaisessa muodossaan koskettamaan tietotekniikkaa ja tiedonsiirtoa esimerkiksi radioaalloilla tai internetin välityksellä. Kuitenkin langaton kommunikaatio käsittää paljon laajemman skaalan kommunikaatiomenetelmiä.

Jo enne teollisen vallankumouksen aikaa ihmiset ovat lähettäneet tietoa toisilleen langattomasti. Menetelminä käytettiin esimerkiksi savumerkkejä, peilejä, semaforeja ja tulimerkkejä. Näitä näköetäisyydeltä käytettäviä menetelmiä edelleen kehittämällä pystyttiin lähettämään varsin monimutkaistakin informaatiota. Myöhemmin nämä menetelmät korvatiin lennättimillä ja tästä vielä eteenpäin pystyttiin lähettämään ensimmäistä kertaa informaatiota langattomasti siinä muodossa, jossa se nykyään käsitetään, eli elektromagneettista säteilyä hyväksikäyttäen. (Goldsmith 2005)

Langaton kommunikaatio on viime vuosikymmeninä saavuttanut suurta kehitystä. Esimerkiksi mobiilitekniikassa on kehitetty useita eri generaatiota, joista tällä hetkellä mennään viidennessä generaatiossa (5G). 2000-luvulle tultaessa mobiililaitteiden määrä on kokenut valtavan kasvun siinä määrin, että ne ovat tällä hetkellä maailman yleisimpiä elektronisia laitteita. Langaton kommunikaatio on myös siirtynyt kattamaan monia jokapäiväisen elämämme osa-alueita, kuten kodeissa ja terveyden huollossa. (Du ja Swamy 2010)

5.2 Elektromagneettiset aallot

Langattomassa kommunikaatiossa informaatiota siirretään avoimessa tilassa käyttäen hyödyksi elektromagneettista säteilyä tai elektromagneettisia aaltoja. Elektromagneettinen säteily on spektri, jonka langattomassa kommunikaatiossa käytetyt osa-alueet jaetaan taajuuskanaviin korkea- tai matalataajuisuuden perusteella. Matalemmilla taajuuksilla aallot pyr-

kivät seuraamaan maanpinnan muotoja, kun taas korkeammilla taajuuksilla aallot kulkevat lähes suorana viivana. Tällä hetkellä korkeimman käytettävä taajuuden raja kulkee 30-300 gigahertsissä (EHF, Extreme high frequency) ja jo tällaisten taajuuksien kohdalla esimerkiksi signaalin lähetys ja kuuluvuus on teknisesti haastavaa. Tämä johtuu siitä, että tällaisilla taajuusalueilla jo pelkästään maapallon ilmakehä riittää aiheuttamaan häiriötä signaalissa. Lisäksi haasteita on aallon muodostamisessa, voimistamisessa, havaitsemisessa ja modulaatiossa. (Du ja Swamy 2010)

Langattomassa kommunikaatiossa signaali siis lähetetään kulkemaan jotakin tiettyä kanavaa pitkin. Käytetyn kanavan kuuluvuus vaihtelee ajan ja taajuusalueen mukaan. Tämä vaihtelu voidaan karkeasti jakaa kahteen luokkaan:

- suuren luokan kuuluvuuden huonontuminen ja
- pienen luokan kuuluvuuden huonontuminen.

Suuren luokan kuuluvuuden huonontumisessa signaalikato johtuu etäisyyden ja suurten esteiden kuten rakennusten tai maanpinnan muotojen aiheuttamasta häiriöstä. Tällainen kuuluvuuden huonontuminen ilmenee, kun signaali kulkee eteenpäin avoimessa tilassa ja on yleensä riippumaton taajuudesta. Pienen luokan kuuluvuuden huonontuminen johtuu muiden samassa tilassa kulkevien signaalien häiritsevästä vaikutuksesta. Tällainen kuuluvuuden heikentyminen on yleensä riippuvaista käytetystä taajuudesta. Mitä korkeataajuisempi signaali, sitä herkempi se on ympäristöstä tuleville häiriötekijöille. (Tse ja Viswanath 2005)

5.3 Langattoman kommunikaation teknologioita

Langatonta kommunikaatiota varten on kehitetty lukuisia eri teknologioita ja niille on eri käyttötapauksia. Tässä alaluvussa käydään läpi osa näistä teknologioista.

2000-lukua lähestyttäessä ja tietotekniikan kehittyessä esimerkiksi PC:t ja kannettavat tietokoneet alkoivat yleistyä voimakkaasti tavallisten ihmisten käytössä. Jo pitkään kuitenkin eri tietoteknisten laitteiden välinen kommunikaatio oli ollut pitkään vaivalloista toteuttaa hyvin pitkälti vain langallisten yhteyksien kautta (Haartsen 2000). Vuonna 1998 viisi sen ajan suurinta teknologiayritystä, Ericsson, Nokia, IBM, Toshiba ja Intel ryhtyivät yhdessä tutkimaan

lisenssi-vapaan langattoman tiedonsiirtomenetelmän kehittämistä ja näin syntyi Bluetooth, lyhyen kantaman langattoman kommunikaation menetelmä (Bhagwat 2001).

Bluetooth on järjestelmänä luonteelta puhtaasti ad hoc (latinen kielestä "tätä tehtävää/tarkoitusta varten) tyyppinen järjestelmä. Ennen Bluetoothin kehittämistä käsitys ad hoc järjestelmistä oli hieman erilainen, käsittämällä se lähinnä kiinteästi tarjotun verkon kautta, johon laitteet voivat yhdistyä ja katkaista yhteyden vapaata tahtia. Bluetooth eroaa tästä siinä mielessä, että kaksi eri laitetta kykenevät ottamaan yhteyden toisiinsa ilman esimerkiksi mitään erillistä verkkoa. Yhteys voidaan muodostaa ilman mitään kiinteää tukiasemaa, ilman mitään infrastruktuuria joka tukisi kahden laitteen välistä yhteyttä tai esimerkiksi ilman kommunikaation koordinoimista. Lisäksi Bluetoothin kanssa samassa tilassa saattaa olla useampi päällekkäinen yhteys. (Haartsen 2000)

Koska Bluetooth on langaton kommunikaatio menetelmä, tulee sen pystyä käyttämään jotakin taajuutta tiedonsiirtoon. Mahdollisia käytettäviä taajuuksia on paljon, mutta erityispiirteitä Bluetoothin kohdalla ovat vapaa julkinen käyttö ja saatavuus maailmanlaajuisesti. Nämä kaksi kriteeriä rajaavat taajuutta, jolla Bluetoothin on mahdollista toimia. Suurin osa olemassa olevista taajuuksista vaatii lisenssin ja eri taajuudet ovat käytössä eri toimijoilla eri puolilla maailmaa. Tähän tarkoitukseen on onneksi yksi taajuus, joka on hyvin pitkälti saatavilla ympäri maailman ilman lisenssiä. Tämä taajuusalue on noin 2.45 gigahertsiä.

Bluetoothlaitteella on käytössään kaksi parametria, jotka ovat mukana käytännössä kaikessa Bluetoothkommunikaatiossa. Ensimmäinen parametri on uniikki 48 bittinen IEEE osoite, joka määrätään jokaiselle Bluetoothlaitteelle erikseen. Toinen parametri on 28 bittinen kellosignaali. Bluetoothlaitteet voivat kommunikoida keskenään hakemalla nämä kaksi parametriä toisiltaan. (Bisdikian 2001)

Bluetooth käyttää tiedustelua ja lajittelua yhdistettävissä olevien laitteiden hakemiseen, sekä niiden lajitteluun. Tiedustelu ja lajittelu ovat molemmat asymmetrisia tapahtumia, mikä tarkoittaa sitä, että tiedustelijalaitteen ja teidusteltavan laitteen tulee suorittaa erilaisia toimintoja. Tämä tarkoittaa esimerkiksi sitä, että molempien laitteiden tulee aloittaa toimintansa eri tiloista, koska muuten kaksi eri laitetta eivät ikinä voisi löytää toisiaan. (Bhagwat 2001)

Bluetoothlaitteita voidaan autentikoida ja yhteyksiä voidaan salata. Koska Bluetooth on luon-

teelta täysin ad hoc tyyppinen, tavanomaiset tietoturvamenetelmät, kuten sertifikaatit ja julkisen avaimen järjestelmä (PKI, Public Key Infrastructure), eivät täysin suorasti päde. Näiden sijaan Bluetoothautentikaatio perustuu haaste/vastaus -mekanismiin (challenge/response mechanism). Autentikaatio voidaan suorittaa missä vain kohdassa yhteyden elinkaarta. Karkeasti ottaen autentikaatio suoritetaan siten, että laite, joka tahtoo autentikoida toisen laitteen, lähettää tälle laitteelle "haastepaketin". Tämä paketti sisältää satunnaisesti generoidun numeron, jolle autentikoitava laite suorittaa omat toimenpiteensä käyttäen 128 bittistä autentikaatioavainta. Autentikoitava laite palauttaa vastauksensa autentikaatiota pyytänelle laitteelle, joka voi tarkistaa vastauksen oikeellisuuden ja näin autentikoida laitteen. Kahden laitteen välinen yhteys on myös mahdollista salata. Käyttäen niin kutsuttua linkkiavainta, voivat toisiinsa yhteydessä olevat laitteet generoida joukon salausavaimia, joilla on mahdollista salata laitteiden välinen lähetys. (Bisdikian 2001)

WiFi (Wireless Fidelity) on langattoman lähiverkon standardi, joka on noussut varsin suosituksi. WiFin helppous näkyy siinä, että se toimii linsensoimattomalla taajuusalueella ja kuka vain voi pystyttää oman WiFiverkkonsa kattaen laajan alueen nopealla internetyhteydellä ja näin antaa pääsyn verkkoon. (Al-Alawi 2006)

WiFi on suosittu nimi IEEE:n (Institute of Electrical and Electronics Engineering) määrittämälle 802.11b mukaiselle standardille koskien lähiverkkoja. Yksi WiFin eduista on siihen liittyvän laitteiston laaja saatavuus. Monet teknologia-alan yritykset pyrkivät varmistamaan uusimpien ohjelmistojen ja laitteistojen saatavuuden, jotta käyttäjät pääsevät vaivattomasti WiFin avulla verkkoon. (Al-Alawi 2006)

IoT sovelluksia on löydettävissä nykyään lähes mistä tahansa. Näillä sovelluksilla on yleensä tarkat määritelmät datan siirrossa etäisyyksien sekä määrien suhteen, sekä energina kulutuksen suhteen. Laajasti käytössä olevat lyhyen matkan langattomat kommunikaatiomenetelmät, kuten Bluetooth eivät kuitenkaan kaikissa tilanteissa kykene vastaamaan näihin haasteisiin, jos esimerkiksi dataa täytyy siirtää suuria välimatkoja. Mobiiliteknologiaan perustuvat kommunikaatiomenetelmät (esimerkiksi 2G, 3G tai 4G) tarjoavat hyvän kattavuuden, mutta yleensä näiden kanssa haasteeksi nousee suuri energina kulutus. Tämän vuoksi on syntynyt tarve kehittää uusi IoT sovelluksille sopiva langattoman kommunikaation menetelmä, LPWAN (Low Power Wide Are Network). LPWAN teknologiat on kehitetty käyttämään

linsensoituja, sekä lisensoimattomia taajuuskanavia. Näistä teknologioista Sigfox, LoRa ja NB-IoT ovat johtavassa asemassa. (Mekki ym. 2019)

Sigfox käyttää sen omaa patentoitua UNB teknologiaa ja jakaa tukiasemansa moniin eri maihin lisensoimattomalla sub-GHz ISM taajudella (868MHz Euroopassa, 915MHz Pohjois-Amerikassa ja 433MHz Aasiassa). Laitteet voivat yhdistyä näihin tukiasemiin käyttäen BPSK modulaatiota hyvin kapealla, sadan hertsin kaistalla, jonka suurin datansiirtonopeus on 100 bittiä sekunnissa. Hyödyntämällä erittäin kapeaa kaistanleveyttä, Sigfoxilla voidaan saavuttaa tehokas kaistan käyttö, sekä erittäin alhaiset melutasot. Tämä taas johtaa alhaiseen virran kulutukseen, korkeaan vastaanottoherkkyyteen, sekä antenniteknologian kehityksen alhaisiin kuluihin. Alunperin Sigfox suuniteltiin vain UL (Uplink) lähetyksiin, mutta myöhemmin se päivitettiin tukemaan myös DL (Downlink) lähetyksiä. Kuitenkin niin, että ensin suoritetaan UL-lähetys. Suurin määrä lähetettävissä olevia viestejä UL-lähetyksessä on 140 kappaletta päivässä ja jokaisen viestin pituus voi olla suurimmillaan 12 tavua. DL-lähetyksissä suurin määrä viestejä on rajoitettu vain neljään kappaleeseen päivässä ja näissä jokaisen viestin suurin pituus voi olla 8 tavua.(Mekki ym. 2018)

LoRaWAN standardoitiin 2015 LoRa-Alliancen toimesta. LoRa on patentoitu hajaspektriteknologia, joka käyttää lisensoimatonta sub-GHZ taajuutta. LoRa mahdollistaa täyden kaksisuuntaisen kommunikaation ja sen avulla muodostutettu signaali sietää hyvin melua ja muuta häiriötä, sekä sitä on hankala jumittaa tai havaita. Jokaisen LoRaWANin kanssa lähetetyn viestin vastaanottaa jokainen kuuluvuusetäisyydellä oleva tukiasema, koska tällä tavalla parannetaan kommunikaation tehokkuutta. Tällainen menettely vaatii kuitenkin usen tukiaseman sijoittamista jokaiselle alueelle, joka puolestaan kasvattaa verkon infrastruktuuria ja näin ollen kuluja. LoRaWAN määrittelee kommunikaatioluokat kommunikaation latenssin mukaan. A-luokka on vähiten virtaa kuluttava luokka, jossa loppulaitteet sallivat molempiin suuntiin kulkevan liikenteen siten, että jokaista UL-lähetystä seuraa lyhyellä aikaikkunalla kaksi DL-lähetystä. B-luokka on muuten samanlainen kuin A-luokka, mutta siihen on lisätty mahdollisuus muutamalle lisälähetykselle. C-luokassa laitteilla on lähes rajattomasti aikaa lähettää dataa. Tämä luokka syö enemmän virtaa ja se onkin suunniteltu IoT sovelluksille, joilla on jatkuvasti virtaa saatavilla. (Mekki ym. 2018)

NB-IoT on kapean kaistan LPWAN teknologia, joka voi olla olemassa rinnakkain LTE:n

tai GSM:n kanssa linsesoiduilla taajuuskanavilla. NB-IoTn kaistanleveys on 200 kilohertsiä. NB-IoT:n kommunikaatiprotokolla perustuu LTE protokollaan, mutta se karsii monia LTE protokollan toiminnallisuuksia minimiin ja lisää niitä vain, jos IoT sovellus sitä vaatii. NB-IoT onkin optimoitu pieniin ja harvakseltaan liikkuviin datamääriin ja pidättymään ominaisuuksista, jotka eivät ole oleellisia IoT-sovelluksissa. Tällä tavalla laitteiden virrankulutus voidaan pitää alhaisena ja kuluja säästävänä. NB-IoT sallii solmuittain lähes satatuhatta laitetta ja tätäkin määrää on mahdollista kasvattaa. Modulaatiotekniikkana toimii QPSK ja se tukee sekä UL-, että DL-lähetyksiä. DL-lähetyksissä suurin suoritusteho on 200 kilobittiä sekunnissa ja UL-lähetyksissä vastaava on 20 kilobittiä. Jokainen viesti voi olla 1600 tavua pitkä. (Mekki ym. 2018)

6 Tutkimus

Tässä tutkimuksessa toteutetaan IoT-sovellus, joka hyödyntää reunalaskentaa ja tarkemmin ottaen reunaälyä. Sovelluksella on tarkoitus havainnoida ympäristössä tapahtuvia muutoksia. Tutkimuskohteeksi valikoitui talvikunnossapitoon liittyvien hoitotoimenpiteiden tarpeen arviointi. Sovelluksella on tarkoitus havainnoida, mahdollisen liukkauden esiintyvyyttä.

6.1 Tutkimusmenetelmä ja tutkimuksen tavoitteet

Tämän tutkimuksen tutkimusmenetelmäksi on valittu konstruktiivinen tutkimus. “Kari Lukka: Konstruktiivinen tutkimusote” (2001) mukaan konstruktiivinen tutkimusote on konstruktioita tuottava metodologia, jolla pyritään ratkaisemaan reaalimaailman ongelmia ja näin tuottamaan kontribuutiota. Konstruktiivisen tutkimusotteen ydinpiirteitä ovat

- keskittyminen tosielämän ongelmiin, jotka koetaan käytännössä tarpeellisiksi ratkaista,
- innovatiivisen konstruktion tuottaminen, joka on tarkoitettu ratkaisemaan alkuperäinen tosielämän ongelma,
- tutkijan, sekä käytännön edustajien hyvin läheistä tiimimäistä yhteistyötä,
- huolellinen kytkeytyminen olemassa olevaan teoreettiseen tietämykseen ja
- empiiristen löydösten reflektoiminen takaisin teoriaan.

Tutkimuksen tavoitteena on tutkia talvikunnossapidon tarpeen ennakointia reunalaskentaa hyödyntävän sovelluksen avulla. Tarpeen arviointia miittaaviksi parametreiksi valittiin mahdollisen liukkauden esiintymisen ennustaminen. Liukkaat tiet ovat varsin yleinen ilmiö Suomen kaltaisissa valtioissa, jotka sijaitsevat korkeammilla leveysasteilla. Tiet ja muut kulkuväylät saattavat Suomessa olla jopa kuusi kuukautta lumen ja jään peitossa. Tilastojen mukaan Suomessa sattuu vuosittain lähes 50 000 sairaalahoitoa vaativaa liukastumisonnettomuutta ja kaikkien liukastumisonnettomuuksien määrä yhteensä on vielä tätäkin suurempi. Tällaiset onnettomuudet aiheuttavat taloudellisia kuluja noin 2 400 miljoonaa euroa vuosittain, mukaan lukien terveydenhuollon kustannukset, sekä menetykset työpäiviin ja hyvinvointiin. Etenkin vanhemmalle väestölle liukastumiset saattaavat olla kohtalokkaitakin.

(Hippi 2012)

Liukkauteen liittyvät riskit ovat hyvin erilaisia, mitä tulee jalankulkijoihin ja ajoneuvoliikenteeseen. Näihin liittyvät onnettomuudet tapahtuvat yleensä eri päivinä ja erilaisissa olosuhteissa. Kova lumisade, jäätävä sade ja pöllyävä lumi ovat ajoneuvoliikenteen näkökulmasta tekijöitä, jotka aiheuttavat hankalat ajo-olosuhteet. Jalankulkijoiden kohdalla taas onnettomuuksia aihettavat vesi tai kuiva lumi jääkerroksen päällä. Myös pakkautunut lumi saattaa muodostua varsin liukkaaksi. (Hippi 2012) Tutkimuksen tavoitteena on rakentaa sovellus, joka pystyy keräämään dataa ympäristöstään ja tästä datasta tekemään päätelmän, onko mahdollista, että tietyillä ympäristöstä saaduilla arvoilla saattaa esiintyä liukkautta.

Liukkauden muodostumisen ennustaminen on haastava prosessi, joka yleensä vaatii useamman parametrin tuottaakseen luotettavia tuloksia. Esimerkiksi Suomen Ilmatieteenlaitoksella käytössä oleva RoadSurf versio 6.60b tiesäämalli vaatii ennustuksien tuottamiseen useamman eri syötteen muun muassa pinnanlämpötilan, tuulen nopeuden, pinnan lähellä olevan suhteellisen kosteuden, sekä sadanta määrän (Toivonen ym. 2019). Tutkimuksen kontekstin huomioon ottaen, resurssit eivät riitä näin monen eri parametrin prosessoimiseen, joten näistä on täytynyt karsia joitakin siten, että pystytään pitämään kuitenkin ne parametrit, joilla pystyttäisiin tuottamaan mahdollisimman luotettavaa tietoa.

Tutkimuksessa rakennettu järjestelmä koostuu kahdesta mikrokontrollerista, sekä lämpötilaa ja ilmankosteutta mittaavista sensoreista. Sensoreilta tullut data on tarkoitus antaa syötteenä tekoälymallille, joka tekee datasta päätelmiä, onko mahdollista, että kyseisillä ympäristöstä kerätyillä parametreilla esiintyy liukkautta.

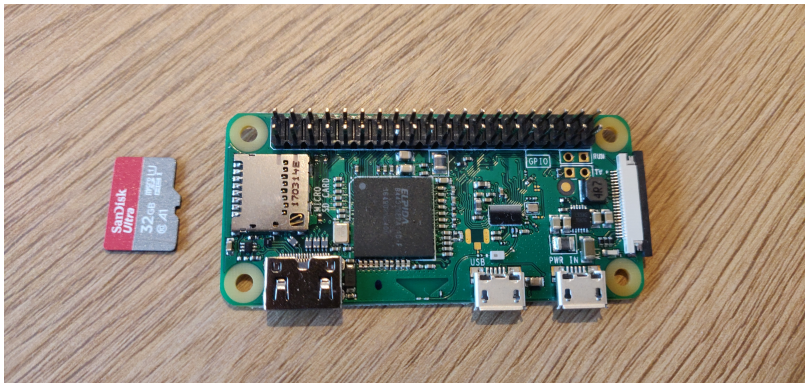
6.2 Alusta

Järjestelmän mikrokontrollereiksi valittiin Raspberry Pi Zeroon perustuva Raspberry Pi Zero W -mikrokontrolleri, sekä Arduino Nano 33 BLE Sense. Raspberry Pi Zero W on pieni alusta, johon voi kytkeä lisälaitteita televisiosta monitoreihin, sekä tärkeimpänä sen avulla voi muodostaa langattomia yhteyksiä. Tutkimuksen kannalta oleellisin on mahdollisuus muodostaa Bluetooth yhteys (Tzivaras 2017). Arduino Nano 33 BLE Sense on pieni ohjelmoitava mikrokontrolleri, jolla on mahdollista ajaa tekoälyä. Lisäksi alusta tarjoaa mahdol-

lisuuden käyttää BLE tekniikkaa langattomassa kommunikaatiossa. (Arduino, n.d.)

Arduino Nano 33 BLE Sensen mitat ovat leveydeltä 18mm ja pituudelta 45mm ja se toimii muista Arduino alustoista poiketen 3.3 voltin virralla. Digitaalisia I/O pinnejä Nano sisältää 14 kappaletta ja analogisia 8 kappaletta. Nano sisältää myös valmiiksi jo levyyn asennettuja sensoreita, nämä sensorit ovat ele-, valoisuus-, ja etäisyyssensori, värähtelysensori, mikrofoni, painesensori, sekä lämpötila- ja ilmankosteussensori. Keskusmuistia Nanossa on 256 kilobittiä, sekä yhden megabitin suuruinen flashmuisti, jonne ajettava ohjelma voidaan ladata. Nano on helppo ohjelmoida Arduino IDE -kehitysympäristössä, Arduino CLI -työkaluilla tai käyttämällä Arduinon omaa verkossa toimivaa editoria. (Arduino, n.d.)

Raspberry Pi Zero W:n mitat ovat 65mm x 30mm x 5mm, joten se on varsin pienikokoinen. MicroSD -muistikorttipaikan ansiosta Raspberry Pi Zero W:n tallennustila on helppo kasvattaa tai supistaa tarpeen mukaan. Raspberry Pi Zero W sisältää kaksi micro-USB -porttia, joista toinen on dataa varten ja toinen 5 voltin virransyöttöä varten. Alusta mahdollistaa lisäksi 2.4 gigahertsin, 802.11n standardin mukaisen langattoman LAN yhteyden, normaalin Bluetoothyhteyden, sekä BLE (Bluetooth Low Energy) yhteyden. Lisäksi Raspberry PI Zero W sisältää 40 kappaletta GPIO (General Purpose Input/Output) -pinnejä, joiden avulla siihen on mahdollista liittää haluamiaan lisälaitteita. (Tzivaras 2017)



Kuvio 1. Raspberry Pi Zero W V1.1 -mikrokontrolleri, sekä muistikortti jolle asennetaan käyttöjärjestelmä.

Jotta Raspberry Pi Zero W olisi käyttövalmis, tulee sille asentaa käyttöjärjestelmä. Käyttöjärjestelmän voi asentaa alustalle liitettävään muistikorttiin. Raspbian on suosituin käyttöjärjestelmä Raspberry Pi laitteille (Harrington 2015). Raspbian on avoimen lähdekoodin De-

Kuvio 2. Arduino Nano 33 BLE Sense

bian perusteinen käyttöjärjestelmä, joka on ladattavissa ilmaiseksi (Harrington 2015). Raspbianista on saatavilla kahta eri versiota. Täysi versio sisältää enemmän ominaisuuksia, sekä graafisen käyttöliittymän. Lite versio on kevyempi, siinä ei ole kaikkia samoja ominaisuuksia kuin täydessä versiossa ja sen mukana tulee ainoastaan komentoliittymä. Tätä tutkimusta varten Raspberry Pi Zero W alustalle asennetaan Raspbianista lite versio. (Tzivaras 2017)

6.3 Sensorit

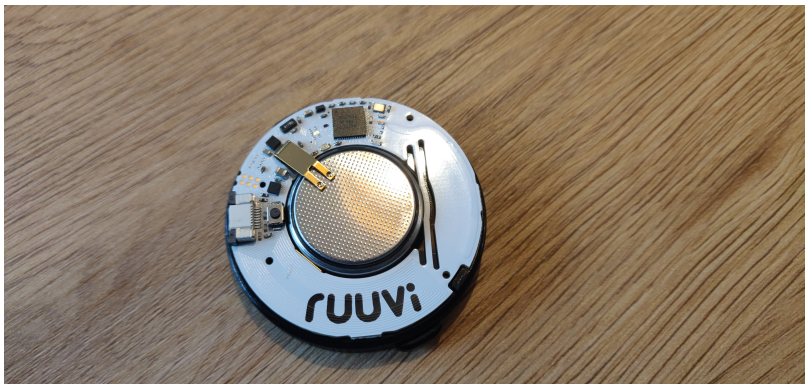
Datan kerääminen ympäristöstä päätettiin hoitaa käyttämällä RuuviTag ja RuuviTag Pro sensoreita. Nämä sensorit ovat vuonna 2016 perustetun Ruuvi Innovations Oy nimisen suomalaisen startup-yrityksen kehittämiä.

Sensoreita oli käytössä kahta mallia, RuuviTag ja RuuviTag Pro. RuuviTag on halkaisijaltaan 52mm ja korkeudeltaan 12,55mm. Sen avulla voidaan mitata neljää arvoa ympäristöstä, ilmanpainetta, lämpötilaa, lmankosteutta, sekä liikettä. RuuviTag Pro on suunniteltu hieman kovempaan käyttöön ja se kestääkin muun muassa korkeampia ja matalempia lämpötiloja. RuuviTag Prota on olemassa kahta eri mallia, hengittävä ja vedenpitävä. Hengittävä malli mittaa lämpötilaa, liikettä ja ilmankosteutta, kun taas vedenpitävä mittaa vain lämpötilaa ja liikettä. RuuviTag Pro tarjoaa myös Ruuvi Connector -laajennusportin, jonka avulla sensoriin voidaan liittää ulkoisia sensoreita.

Ruuvi tarjoaa myös Ruuvi Station nimisen mobiilisovelluksen, jonka avulla sensoreista saa-

tuja lukemia on mahdollista seurata. Sovellus on saatavilla sekä Androidille, että iOSille. Kerättyä dataa on mahdollista seurata sovelluksessa reaaliaikaisesti, sekä historiallisina kaavioina. Sovelluksen kautta sensoreita voidaan myös hallita, sieltä voidaan asettaa hälytyksiä, tarkastella säätietoja, sekä jakaa sensorien lähettämää dataa pilvipalveluun. Tämän tutkimuksen kontekstissa Ruuvi Station sovellusta ei kuitenkaan käytetty. (Ruuvi 2016)

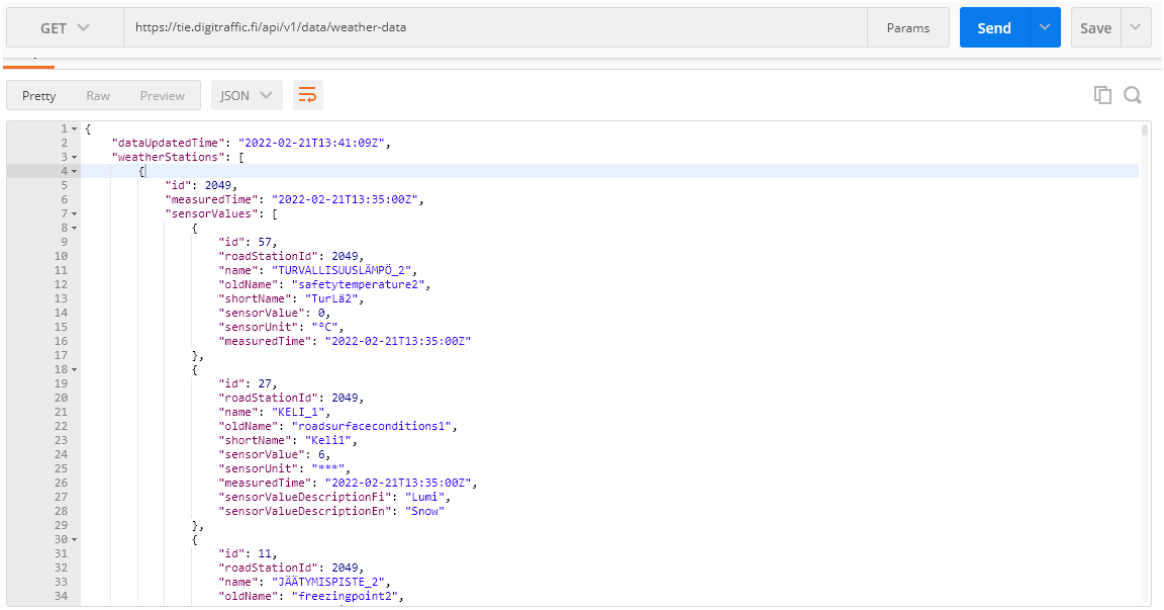
Kuvio 3. RuuviTag Pro ja RuuviTag sensorit, vasemmalla liikettä ja lämpötilaa mittaava sensori, oikealla lämpötilaa, ilmankosteutta, ilmanpainetta ja liikettä mittaava sensorii.



Kuvio 4. RuuviTag Pro sensori avattuna.

6.4 Aineiston keruu ja muokkaus

Tutkimusta varten tuli hankkia aineisto, jonka avulla olisi mahdollista ennustaa tarpeeksi luotettavasti liukkauden esiintymistä. Liukkauden esiintymisen ennustamiseen päädyttiin käyttämään ilman kastepistettä, sekä tien lämpötilaa. Kyseessä olevia vaatimuksia vastaava aineisto löytyi Fintrafficin tarjoaman Digitraffic Road API -rajapinnan kautta (Fintraffic, n.d.).



Kuvio 5. Kuva Postman sovelluksesta, Digitraffic palveluun tehdyn GET pyynnön jälkeen.

Digitraffic Road palveluun kerätään dataa ajoneuvoliikenteestä ITM Finlandin tarjoamista liikenteenhallinnointi järjestelmistä. Rajapinnan kautta saatiin haettua dataa tekemällä Postman sovelluksen avulla GET pyyntö osoitteeseen *https://tie.digitraffic.fi/api/v1/data/weather-data*. Pyyntö palauttaa json-objektin, joka sisältää sääasemittain mittaustulokset sekä tiedon, milloin mittaustulokset on päivitetty. Rajapinnasta saadusta datasta eristettiin tutkimusta varten oleelliset parametrit, jotka olivat tienpinnan lämpötila, ilmanlämpötila, sekä ilmankosteus. Parametrit eristettiin koko datasta yksinkertaisella JavaScript-skriptillä.

Tarvittavien parametrin eristämisen jälkeen datan muokkausta jatkettiin excelissä. Ilmanlämpötilan ja ilmankosteuden perusteella jokaiselle riville laskettiin kastepiste. Liukkauden ennustamisen kannalta mielekkäintä oli seurata sitä, miten nopeasti liukkautta muodostuu ja tämän määrittelyyn jokaiselle mittausriville lisättiin kulmakerroin, jonka jyrkkyys kertoo liukkauden muodostumisesta. Kulmakertoimen ollessa kauempana nollasta, voidaan olettaa liukkauden muodostuvan nopeasti ja kun taas se lähestyy nollaa, niin voidaan olettaa liukkauden muodostumisen olevan hitaampaa. Kulmakertoimen perusteella määriteltiin vielä asteikko nollasta kahteen. Asteikossa nolla tarkoittaa, että liukkautta ei juuri ole, yksi tarkoittaa, että liukkautta saataa muodostua ja kaksi tarkoittaa, että liukkautta on muodostunut nopeastikin.

6.5 Tekoälymallin ohjelmointi

Kun datasetti oltiin saatu haluttuun muotoon, aloitettiin tekoälymallin ohjelmointi. Itse tekoälymalli tehtiin TensorFlowta käyttäen. TensorFlow on syväoppimiseen ja tekoälyyn erikoistuneen tutkimusryhmän, Google Brainsin, kehittämä avoimen lähdekoodin ohjelmistokirjasto. Google tarjoaa ilmaisen Google Colab ympäristön, jossa on mahdollista kirjoittaa Jupyter Notebook -tyyppisiä "muistiinpanoja". Kyseisessä ympäristössä on valmiina tuki Pythonille, jota käytetään TensorFlown kanssa ohjelmoidessa.

Mallin tekeminen aloitettiin asentamalla tarvittavat kirjastot Pythonin pip nimisen paketinhallintajärjestelmän avulla. Asennettavat kirjastot olivat:

- pandas,
- numpy,
- matplotlib ja
- scikit-learn

Pandas on datan manipulointiin ja analysointiin tarkoitettu kirjasto, Numpy on hyödyllinen Python kirjasto taulukkomuotoisen datan kanssa työskentelyssä, Matplotlib on datan visualisointiin tarkoitettu kirjasto ja Scikit-learn on koneoppimiskirjasto, josta löytyy tarvittavia työkaluja tekoälymallin luomiseksi.

Aiemmin luotu datasetti otettiin käyttöön Google Colab -ympäristössä ja se jaettiin kahteen ryhmään, jotka nimettiin X ja y. Ryhmä X sisältää datasetistä ominaisuudet, joiden perusteella ennustaminen haluttiin tehdä, eli kastepisteen, tien lämpötilan ja kulmakertoimen. Ryhmään y datasetistä jätettiin ominaisuus, jota haluttiin ennustaa eli liukkausasteikko. Tämän jälkeen ryhmät X ja y jaettiin koulutus ja testaus ryhmiin suhteessa 20/80, eli 80 prosenttia datapisteistä jaettiin koulutusryhmään ja 20 prosenttia testausryhmään. Tämän jälkeen koulutusryhmiä muokattiin, jotta poikkeavia havaintoja saatiin hieman karsittua.

Koska datasetin data oli luonteeltaan lineaarista, päädyttiin tekoälyalgoritmina käyttämään Sequental -algoritmia, joka soveltuu parhaiten lineaaristen datapisteiden kanssa toimimiseen. Malli ajettiin lukuisia kierroksia, jotta sille saatiin mahdollisimman hyvä tarkkuus. Kun haluttu tarkkuus oltiin saavutettu, pakattiin valmis malli Tensorflow lite malliksi, joka sopii pa-

remmin IoT laitteiden kanssa yhteen, sillä se ei ole yhtä raskas, kuin perinteinen Tensorflow malli.

7 Järjestelmän toimintaan laittaminen

Järjestelmän toiminnassa käytettiin Arduino Nano 33 BLE Senseä, sekä Raspberry Pi Zero W:tä, sekä RuuviTag ja RuuviTag Pro sensoreita. Raspberry Pi:n tehtävänä oli toimia päälaitteena kommunikoiden niin sensorien, kuin Arduinonkin kanssa. Arduinon tehtävänä oli vastaanottaa Raspberry Pi:n sensoreilta keräämä data, syöttää data tekoälymallille ja lähettää tekoälymallin antama tulos takaisin Raspberry Pi:lle. sensoreiden käyttöönotto oli varsin yksinkertaista. Sensorit sisälsivät pariston valmiina ja niistä tuli ainoastaan poistaa muoviliuska, jotta patteri pääsi koskettamaan kontaktipintaa. Tämän jälkeen sensorit olivat käyttövalmiita.

7.1 Raspberry Pi

Raspberry Pi:lle luotiin kansio edge-ai-app, jonne sovelluksen koodi sijoitettiin, ohjelmointikielenä käytettiin Pythonia. Kansioon luotiin skripti `run.py`, jonne tuli sovelluksen niin kutsuttu pääohjelma. Lisäksi luotiin erillinen skripti `get_ruuvitag_data.py`, jonka tehtävänä oli hoitaa kommunikaatio RuuviTag sensorien kanssa, sekä `calc_dewpoint.py` ja `calc_slope.py`, joiden avulla laskettin kastepiste ja kulmakerroin. RuuviTag sensoreille on luotu oma pythonkirjastonsa ja tämän lisäksi myös Raspberry Pi:lle oma versionsa tästä kirjastosta ja tämä kirjasto asennettiin Raspberry Pi:lle. Kirjaston asennusohjeet löytyivät Ruuvin GitHub sivulta.

Kirjaston asentamisen jälkeen kirjoitettiin koodi kommunikaatiota varten. `get_ruuvitag_data.py` skriptissä otettiin käyttöön RuuviTag pythonkirjasto, josta käytettiin `get_datas_for_sensors()` -metodia datan hakemiseksi. `get_datas_for_sensors()` -metodille tulee antaa parametreina RuuviTag sensoreiden MAC-osoitteet, sekä aika sekunteina, kuinka pitkään odotetaan sensoreilta vastausta. Skripti palauttaa RuuviTageilta saadut lukemat.

Kun kommunikaatio RuuviTagien kanssa oltiin saatu toimintaan, kirjoitettiin skripti kastepisteen laskemiselle. Kastepisteelle saadaan laskettua yksinkertainen approksimaatio kaavasta

$$Td = T - ((100 - RH)/5$$

, jossa Td on kastepisteen approksimaatio, T on havaittu lämpötila ja RH on ilmankosteuden prosentti. Yllä olevaa kaavaa käyttämällä laskettiin kastepiste. Kulmakerroin laskettiin kastepisteestä, sekä lämpötilasta lähellä maantasoa kaavasta

$$(y_2 - y_1)/(x_2 - x_1)$$

, jossa y_2 on kastepiste, y_1 on 0, x_2 on lämpötila ja x_1 on 0. Skripti palauttaa kulmakertoimen.

Pääohjelmana toimii `run.py`, jossa suoritetaan kaikki kutsut. Koko ohjelmaa suoritetaan ikuisessa silmukassa. Ensin kutsutaan RuuviTagien kanssa kommunikoivasta skriptistä funktiota `get_data()`, joka palauttaa RuuviTageilta saadut lukemat. Tämän jälkeen lasketaan kastepiste kutsumalla kastepisteen laskevasta skriptistä `calculate()` funktiota, jolla annetaan paramtreiksi ilmankosteus ja lämpötila. Kun kastepiste ollaan saatu, lasketaan viimeinen arvo eli kulmakerroin kutsumalla kulmakertoimen laskevasta skriptistä funktiota `calc_slope()`, jolle annetaan parametreiksi kastepiste ja lämpötila. Kun kaikki tarvittavat arvot ollaan saatu, ne enkoodataan tavutaulukoiksi.

Seuraava askel on muodostaa BLE yhteys Arduinoon. Tämä tapahtuu käyttämällä pythonin Bleak kirjastoa. Bleak on GATT asiakasohjelma, joka mahdollistaa eri laitteiden välisten yhteyksien muodostamisen käyttäen BLE:tä. Bleakista otetaan käyttöön `BleakClient()` metodi, jolla annetaan parametrina Arduinon MAC-osoite, joka tässä tapauksessa on 93:3E:66:B9:26:FB. Kun yhteys ollaan muodostettu, voidaan lähettää data Arduinolle. Datan lähettäminen tapahtuu kirjoittamalla data BLE:ssä käytössä oleviin tunnuspiirteisiin (eng. Charasteristic). Jokaiselle datapisteelle määriteltiin oma tunnuspiirteensä, jonne tätä vastaava arvo kirjoitettiin. Tunnuspiirteille tulee määrittää yksilöllinen UUID, jolla ne voidaan yksilöidä. Kun tunnuspiirteisiin ollaan kirjoitettu data, odotetaan viisi sekuntia, jotta Arduinon päässä tekoälymalli ehtii suorittaa päätelmän ja jotta se saadaan kirjoitettua sitä vastaavaan tunnuspiirteeseen. Kun tarvittava aika ollaan odotettu, käydään lukemassa päätelmä. Koska kyseessä on kokeellinen järjestelmä, ei päätelmän kanssa tehdä muuta kuin kirjoiteta se vain lokitiedostoon parametrien kanssa, jotka tuottivat kyseisen päätelmän. Koko koodi on luettavista liitteestä C.

7.2 Arduino Nano 33 BLE Sense

Arduinolla koodi muodostuu yhdestä skirptistä, jonne lisättiin kaikki toiminnallisuudet. Ensimmäiseksi skriptissä otetaan käyttöön Arduino kirjasto ArduinoBLE, sekä TensorFlow-Lite kirjasto. Tämän jälkeen skriptiin liitetään tiedosto model.h, joka sisältää tekoälymallin heksalukumuodossa. Seuraavat askeleet sisältävät paljon alustamista liittyen bluetoothyhteyteen, sekä tekoälymalliin. Tekoälymallille määritellään muun muassa syötteet ja ulostulot, sille varataan tarvittava määrä muistia, sekä määritellään syötteiden nimet. Bluetoothin alustuksessa muun muassa luodaan tunnuspiirteet, määritetään niille UUID:eet, määritellään niille metodit joita niille voidaan suorittaa ja asetetaan Arduino vastaanottamaan yhteyksiä.

Yhteyksien vastaanottamista, tunnuspiirteisiin kirjoitetun datan lukemista ja tunnuspiirteisiin kirjoittamista varten luotiin omat funktiot. Kun Arduinolle luodaan uusi yhteys kutsutaan `onBLEConnected()` funktiota, joka vain ilmoittaa uudesta yhteydestä ja tulostaa yhdistyneen laitteen MAC-osoitteen. Kun tunnuspiirteisiin kohdistuu kirjoitustapahtuma, luotiin funktiot näiden tapahtumien hoitamista varten. Nämä funktiot vain tarkkailevat mahdollisia kirjoitustapahtumia, lukevat tunnuspiirteisiin kirjoitetun datan ja lisäävät tämän datan tekoälymallin syötteisiin. Kun kaikki tarvittava data ollaan saatu, kutsutaan `makeInference()` funktiota, joka lukee tekoälymallin päätelmän ja kirjoittaa sen päätelmälle tarkoitettuun tunnuspiirteeseen. Kun BLE yhteys katkaistaan, kutsutaan `onBLEDisconnected()` funktiota, joka vain ilmoittaa yhteyden katkaisusta ja taas tulostaa yhteyden katkaisseen laitteen MAC-osoitteen.

8 Järjestelmän testaus ja tulokset

Kun järjestelmä oltiin saatu toimimaan, sen toimintaa testattiin viikon ajan. Tässä luvussa kerrotaan järjestelmän testauksesta, järjestelmän toiminnasta testauksen aikana, sekä saaduista mittaustuloksista.

8.1 Mittausolosuhteet

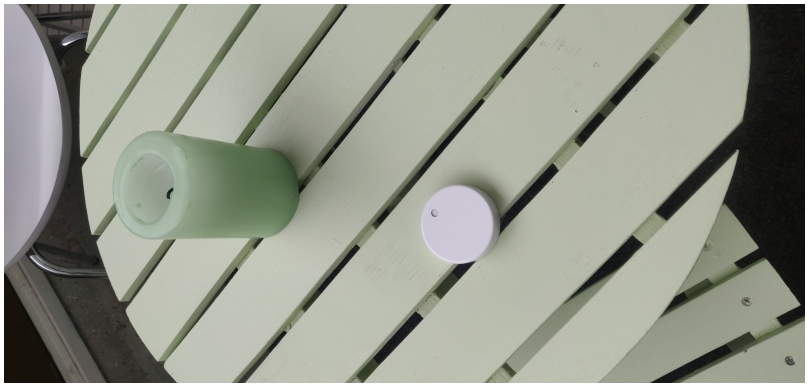
Koska järjestelmän toimintaan laittaminen pitkittyi melko pitkälle keväälle, oli varsin haastavaa saada optimaaliset mittausolosuhteet. Tämän vuoksi itse konkreettisen liukkauden mittaaminen piti jättää pois ja testaus päätettiin suorittaa itse järjestelmän toiminnan näkökulmasta. Toimintaa testattiin pitämällä silmällä, miten järjestelmän eri palaset toimivat yhdessä, sekä sitä, miten mittausten tekeminen datasta ylipäänsä onnistuu.

RuuviTag sensorit sijoitettiin tutkimuksen tekijän kodissa sijaitsevalle parvekkeelle. Parveke on noin $6.6m^2$, lasitettu ja sijaitsee toisessa kerroksessa. RuuviTag Pro sensori sijoitettiin parvekkeelle lattiatasoon simuloimaan mitattavaa tienpinnan lämpötilaa ja RuuviTag sensori sijoitettiin parvekkeella olevalle pöydälle mittaamaan ilmanlämpötilaa, sekä ilmankosteutta. Parvekkeen lattia on betonia, joka pysyi koko mittausajan suhteellisen viileänä. Parvekkeen lasit olivat koko mittausajan kiinni, jonka vuoksi ilmankierto parvekkeen sisällä oli varsin heikko. Aurinkoisina päivinä aurinko paistaa suoraan parvekkeella pitkälle päivään saakka, jonka takia lämpötila parvekkeella yleensä nousee melko korkeaksikin, varsinkin kevätauringon lämmittäessä.

Mittauksia suoritettiin viikon ajan aikavälillä 28.3. - 4.4., jonka ajan sensorit olivat sijoitettuna parvekkelle. Mittausaikavälille osui lämmin ajanjakso ja lämpötilavaihtelut olivat kohtuullisen suuria. Useampana päivänä lämpötila parvekkeella kohosi lähes kolmeenkymmeneen asteeseen. Lämpötila alkoi kohota varsin nopeasti jo heti aamulla, noin kello 9:00 alkaen, kun aamuaurinko alkoi paistaa suoraan parvekkeelle. Lämpötilahuippu saavutettiin mittausajanjaksolla päivittäin noin kello 13:00 - 14:00, jonka jälkeen se alkoi tippua varsin nopeasti. Parvekkeen sisällä lämpötila pysyi plussan puolella pitkälle yöhön ja ainoastaan kahtena yönä lämpötila kävi nollan alapuolella. Ensimmäisellä kerralla nollan alapuolelle

tippuminen tapahtui aamuyölle mentäessä ja toisella kerralla vasta aamun varhaisina tunteina ja kello yhdeksään lähestyttäessä lämpötila lähti taas melko nopeasti nousuun.

Myös ilmankosteus heitteli parvekkeella suhteellisen paljon. Mittausten ensimmäisinä päivinä, noin 28.3. - 1.4., lämpötila oli hieman alhaisempi, jonka johdosta ilmankosteus pääsi kohoamaan noin neljäänkymmeneen prosenttiin. Kuvaajaa seuraamalla on nähtävissä selvä trendi ilmankosteuden vaihtelussa ilmanlämpötilan suhteen. Ilmankosteus oli korkeimmillaan yöllä ja aamuisin, mutta päivää lähestyttäessä kuvaajasta huomaa pienen pudotuksen ilmankosteudessa. Kosteus ei kuitenkaan pudonnut juurikaan alle kolmenkymmenen prosentin, koska lämpötila ei päässyt nousemaan kovin paljon. 1.4. lämpötila parvekkeella alkoi kohoamaan reilusti enemmän, jonka johdosta myös ilmankosteus koki rajun pudotuksen. Raju pudotus lämpötilassa aiheutti sen, että ilmankosteus putosi alkuun reilusti alle kahdenkymmenen prosentin keskipäivällä noin kello 9:30 alkaen ja pysyi alhaisena aina kello 17:00 saakka, jonka jälkeen lämpötilan laskiessa ilmankosetus lähti taas nousuun.

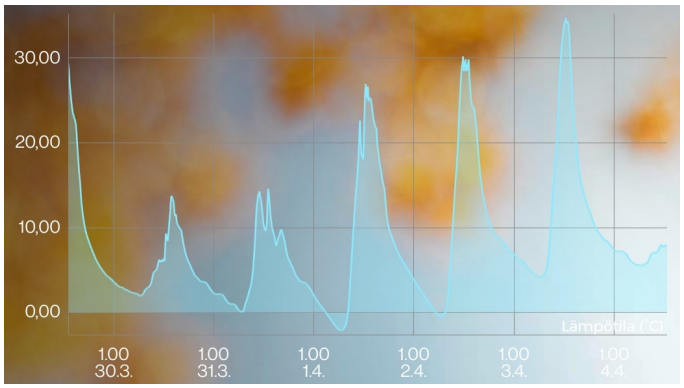


Kuvio 6. Parvekkeen pöydällä oleva RuuviTag sensori, joka mittaa ilmanlämpötilaa ja ilmankosteutta.

8.2 Järjestelmän toiminta

Tutkimuksen tarkoituksena oli tarkkailla rakennetun järjestelmän toimintaa, sekä sen kykyä suorittaa mittausta. Mittaustulokset kerättiin lokitiedostoon. Kokonaisuutena katsoen järjestelmä toimi kuten sen oli ollut tarkoituskin, mutta samalla sen toiminnassa oli huomattavissa tiettyjä epäluotettavuuksia.

Kuvio 7. Parvekkeen lattialla oleva RuuviTag Pro sensori, joka mittaa lämpötilaa lattian tasolla.



Kuvio 8. RuuviTag sensorin mittaamat lämpötilalukemat mittausajanjaksolta.

Kulmakertoimen ja kastepisteen laskutoimitukset suorittavat funktiot toimivat kuten pitikin. Mittausympäristöstä kerätty data oli jokseenkin erilaista kuin oltiin tarkoitettu ja tämän johdosta saadut tulokset erosivat jonkin verran siitä, millaista dataa tekoälymallin koulutusvaiheessa oltiin käytetty. Lasketoimituksen suorittaminen oli kuitenkin varsin mekaaninen tehtävä, joten tämän suhteen järjestelmässä ei ollut puutteita.

Bluetoothyhteyden suhteen järjestelmässä ilmeni joitakin epävarmuuksia etenkin Raspberry Pi:n puolella. Alkuun jostain syystä datan hakeminen RuuviTageilta ei aina onnistunu, vaan yhteys aikakatkaistiin. Tämä saatiin korjattua kasvattamalla odotusaikaa, pitämällä yhteyttä auki, mutta tämä toisaalta hidasti koko järjestelmän kokonaistoiminta-aikaa. Bluetoothyhteyden muodostaminen Arduinon kanssa ei myöskään onnistunut aina täydellisesti. Vaikka Arduinon koodissa asetettiin laitteelle nimi jolla se itseään mainostaa, ei tätä nimeä näkynyt

laitteiden skannauksen yhteydessä. Tämä ei tosin mitenkään varsinaisesti vaikuttanut yhteyden muodostamiseen, muuta kuin siltä osin, että se teki siitä työläämpää, kun Arduino piti etsiä MAC-osoitteen perusteella. Alkuun Arduinolta saatuja tekoälyn tuottamia tuloksia ei tallennettu mihinkään, vaan ne ainoastaan tulostettiin komentoriville. Kun mukaan tuotiin tulosten tallentaminen, ilmeni ongelmia yhteyden muodostamisessa Arduinolle. Jostain syystä yhteyttä ei aina pystytty muodostamaan, jonka seurauksena lokitiedostoon kirjattiin vain virheilmoitus. Usein tämä ei jäänyt vain satunnaiseksi, vaan se tapahtui lähes varmuudella jossakin vaiheessa ohjelman suorittamista. Kun yhteyden muodostuksessa oltiin kerran epäonnistuttu, ei tätä seuravillakaan kerroilla yhteyden muodostaminen enää onnistunut. Tämä johti siihen, että järjestelmän toimintaa piti tarkkailla tietyin väliajoin ja kun yhteyden muodostaminen epäonnistui, piti sekä Raspberry Pi, että Arduino käynnistää uudelleen, jonka jälkeen järjestelmä toimi taas normaalisti. Kokonaisuutena katsoen tällainen toiminta teki järjestelmän toiminnasta varsin epäluotettavan ja esimerkiksi ei voinut luottaa, että jos järjestelmän jättää esimerkiksi yöksi päälle, on se tallentanut mittaustuloksia tasaisesti.

Arduinon puolella ilmeni myös ongelmia. Jo heti alkuun mainittiin laitteen nimen näkymättömyys skannauksen yhteydessä, mutta tämä ei tosin ollut kovin merkittävä. Ongelmat ilmenivät tekoälyn kanssa siinä yhteydessä, kun tehtiin ennustuksia. Arduinolle alettiin rakentaa koodia siten, että ensin otettiin tekoälymalli käyttöön ja tämän jälkeen otettiin käyttöön Bluetooth. Ennen kuin kuin Bluetooth otettiin käyttöön, ei tekoälyn kanssa ilmennyt mitään ongelmia. Arduinolla on koodissaan kaksi pääfunktiota, setup ja loop. Setup funktioon kirjoitetaan kaikki tarvittavat alustukset, kuten esimerkiksi kirjastojen käyttöönotto. Loop funktio on itse halutun asian suorittamista varten ja käytännössä katsoen tätä funktiota suoritetaan silmukassa niin nopeasti kuin mahdollista. Kun Bluetoothia ei ollut käytössä, pystyi loop funktiota käyttämään päätelmien tekemiseen tekoälyn kanssa. Kun Bluetooth otettin mukaan, ilmeni suorituksen kanssa kummallista toimintaa. Jostain syystä funktiossa alettiin suorittaa itse päätelmien tekemistä niin nopeasti kuin mahdollista jättäen täysin huomiotta kaikki siihen asetetut viivästykset ja tämä taas johti virheellisiin päätelmiin tekoälymallilta. Tämä osaltaan myös johti toisinaan Bluetooth yhteyden katkaisuihin tai siihen, että yhteyden muodostamistakaan ei jääty odottamaan ollenkaan. Nämä ongelmat saatiin ratkaistua osaltaan hoitamalla yhteyden muodostaminen, paramtrien syöttäminen, sekä päätelmien teko omissa funktioissaan. Näin pystyttiin varmistamaan, että tekoäly teki päätelmän vain kerran

ja yhteys pidettiin päällä niin pitkään, kuin oli tarpeellista.

8.3 Mittaustulokset

Raspberry Pi keräsi mittautulokset `data.log` nimiseen lokitiedostoon. Lokitiedostoon tulostui informaatiota RuuviTageilta haettujen lukemian aloittamisesta ja vastaanottamisesta, sekä näiden MAC-osoitteet. Kun RuuviTageilta oltiin vastaanotettu data, otettiin yhteys Arduinoon, jolle RuuviTageilta saatu data lähetettiin. Kun data oltiin lähetetty ja tekoälymallin päätelmä vastaanotettu, yhteys suljettiin. Tämän jälkeen lokitiedostoon tulostettiin kastepisteen, lämpötilan ja kulmakertoimen arvot ja loppuun vielä tekoälymallin antama päätelmä näiden arvojen pohjalta. Tulostetut rivit ovat nähtävissä alla olevassa lyhyessä osassa koko lokitiedosta. Rivit 1-3 liittyvät keskusteluun RuuviTagien kanssa, rivit 4-8 liittyvät keskusteluun Arduinon kanssa ja loput rivit 9-13 ovat tulosteet arvoista.

```
...
1. 04/04/2022 11:43:15 AM Get latest data for sensors. Stop with
   Ctrl+C.
2. 04/04/2022 11:43:15 AM Stops automatically in 20s
3. 04/04/2022 11:43:15 AM MACs: ['EA:33:0E:59:73:83',
   'C5:74:D0:24:B4:77']
4. 04/04/2022 11:43:15 AM Start receiving broadcasts (device hci0)
5. 04/04/2022 11:43:15 AM FYI: Calling a process with sudo:
   hciconfig hci0 reset
6. 04/04/2022 11:43:15 AM FYI: Spawning process with sudo: hcitool
   -i hci0 lescan2 --duplicates
7. 04/04/2022 11:43:15 AM FYI: Spawning process with sudo: hcidump
   -i hci0 --raw
8. 04/04/2022 11:43:35 AM Stop receiving broadcasts
9. 04/04/2022 11:43:46 AM Dewpoint value: -3.15
10. 04/04/2022 11:43:46 AM Temp value: 6.22
11. 04/04/2022 11:43:46 AM Slope value: 0.51
12. 04/04/2022 11:43:46 AM Inference value: -0.15
13. 04/04/2022 11:43:48 AM Going to sleep for 1 hour
```

```
...
```

Lokitiedostoon kirjoitetuista tuloksista voidaan päätellä, että järjestelmä toimi pääasiassa niin kuin pitikin, vaikkakin tulokset eivät olleet sellaisia, joita olisi voinut odottaa. Lokitiedostosta on huomattavissa, että tekoälymallin tarkkudessa olisi ollut vielä melko paljon säätämiseen varaa, sillä sen antamat tulokset olivat joiltakin osin melko epätarkkoja. Tekoälymallin tulosten oli tarkoitus kertoa liukkauden muodostumisen todennäköisyydestä, mitä enemmän tulos oli nollasta poikkeava, sitä todennäköisemmin liukkautta saattaisi muodostua. Lokitiedostosta oli nähtävissä, että monissa tapauksissa tekoälymallin antamat tulokset poikkesivat huomattavasti nollasta, jopa sellaisissa tapauksissa kun näin ei olisi pitänyt olla. Tekoälymallille syötettävistä parametreista kulmakerroin on se, jonka perusteella voidaan hieman päätellä mahdollista tulosta, jonka tekoälymalli saattaisi antaa. Tämä johtuu siitä, että kulmakerroin kertoo kastepisteen ja pinnanlämpötilan erosta, mitä jyrkempi kerroin on, sitä suurempaa on lämpötila ero ja näin ollen herkemmin kosteus tiivistyy tien pintaan ja näin muodostuu liukkautta.

Kun lokitiedostoa tutkittiin huomattiin, että tekoälyn antamat tulokset poikkesivat paljonkin nollasta jopa niissä tapauksissa, joissa kulmakertoimen perusteella ei olisi pitänyt. Esimerkiksi alla olevassa tapauksessa,

```
...
1. 30/03/2022 12:01:47 PM Dewpoint value: -4.26
2. 30/03/2022 12:01:47 PM Temp value: 4.50
3. 30/03/2022 12:01:47 PM Slope value: 0.95
4. 30/03/2022 12:01:47 PM Inference value: -4.26
...
```

jossa kulmakerroin on varsin loiva, olisi voinut odottaa tekoälyn tekemän päätelmän olevan lähempänä nollaa. Tuloksista on havaittavissa, että päätelmä on itseasiassa täysin sama arvo kastepisteen kanssa ja olisi mielenkiintoista selvittää, millä tavalla tällaiseen arvoon ollaan päädytty.

Tiedostosta on myös nähtävissä useaan otteeseen tapauksia, joissa yhteyden muodostaminen

Arduinoon on epäonnistunut. Tämä epäonnistuminen on ilmaistu lokitiedostoon kirjoitetulla "Error occurred"viestillä, joka on nähtävissä alla rivillä 9

```
...
1.  30/03/2022 09:24:17 AM Get latest data for sensors. Stop with
    Ctrl+C.
2.  30/03/2022 09:24:17 AM Stops automatically in 20s
3.  30/03/2022 09:24:17 AM MACs: ['EA:33:0E:59:73:83',
    'C5:74:D0:24:B4:77']
4.  30/03/2022 09:24:17 AM Start receiving broadcasts (device hci0)
5.  30/03/2022 09:24:17 AM FYI: Calling a process with sudo:
    hciconfig hci0 reset
6.  30/03/2022 09:24:17 AM FYI: Spawning process with sudo:
    hcitool -i hci0 lescan2 --duplicates
7.  30/03/2022 09:24:17 AM FYI: Spawning process with sudo:
    hcidump -i hci0 --raw
8.  30/03/2022 09:24:37 AM Stop receiving broadcasts
9.  30/03/2022 09:24:43 AM Error occurred
10. 30/03/2022 09:24:43 AM Going to sleep for 1 hour
...
```

Lokitiedostosta on nähtävissä, että kun tällainen virhetilanne sattui yhdenkin kerran, jäi koko sovellus herkästi jumiin tähän samaan virheeseen kanssa ja tunnin kuluttua kun yhteyttä koitettiin muodostaa uudelleen, sattui edelleen tämä sama virhe. Todennäköisesti virhe johtui jostakin yhteyden muodostamiseen liittyvästä asiasta. Valitettavasti Arduinolta ei otettu talteen mitään lokeja, joten varsinaista syytä yhteyden muodostuksen epäonnistumiselle on vaikea sanoa. Mitä todennäköisimmin vika ei voi olla itse Arduinon piirilevyssä, sillä yhteys saatiin kuitenkin muodostuettua. Ongelma saattoi siis johtua jostakin ulkopuolisesta tekijästä. Yksi mahdollinen syy saattoi olla ympäristössä oleva "melu". Sekä Arduino, että Raspeberry Pi olivat paikassa, jossa lähes vieressä oli WiFi-reititin. BLE toimii 2.4 gigahertsin taajuudella ja tämä taajuusalue on sama, kuin WiFi-yhteyden hitaampi kaista. Tämä on saattanut sekoittaa yhteyden muodostamisen Raspberry Pi:ltä Arduinoon ja näin ollen yhteyttä ei ole ollut mahdollsita muodostaa. Lokeista päätellen virhe näyttäisi ilmaantuvan niinä aikoi-

na, kun WiFi:n käyttö on ollut suurta. Nämä ajankohdat ovat suurinpiirtein aamulla, päivällä ja illalla. Lisäksi jokaisessa tapauksessa kun virhe huomattiin, saatiin se korjattua käynnistämällä Arduino ja Rapsberry Pi uudestaan. Nämä seikat tukevat arveluita siitä, että ongelman aiheuttaja olisi ympäristön melu. Ongelma olisi saattanut ratketa, jos Arduino ja Raspberry Pi oltaisiin siirretty kauemmaksi reitittimestä. Kaikesta huolimatta mittaustuloksia saatiin kerättyä kattava määrä ja näiden perusteella on mahdollista todentaa, että sovellus kykenee keräämään dataa ympäristöstä ja tekemään tämän datan pohjalta päätelmiä olosuhteista.

9 Yhteenveto

Tässä Pro Gradu -tutkimuksessa oli tarkoituksena tutkia reunaälyn hyödyntämistä liukkauden torjunnassa. Gradun tuloksena rakennettiin kokeellinen sovellus, joka pystyy keräämään dataa ympäristöstä, muokkaamaan kerättyä dataa, syöttämään datan tekoälymallille ja näin tuottamaan tuloksia. Sovellusta testattiin viikon ajan tutkimuksen tekijän parvekkeella ja sen avulla saatiin tuotettua tuloksia, vaikka nämä tulokset eivät olleet kovinkaan tarkkoja.

Tutkimuksen tuloksena voidaan sanoa, että reunaälyä on mahdollista hyödyntää tutkimuksen aiheen kaltaisessa skenaariossa. Sovelluksen toiminnassa oli kuitenkin merkittäviä puutteita, eikä se sellaisenaan sovelutuisi varsinaiseen käyttöön. Alunperin testaus oli tarkoitus suorittaa oikeassa ympäristössä esmierkiksi jossakin paikassa, jossa ihmiset liikkuvat paljon ja jonne liukkautta saattaa muodostua. Tämä kuitenkin epäonnistui, sillä tutkimusvaiheeseen päästiin vasta siinä kohtaa, kun liukkauden muodostuminen alkoi olla varsin epätodennäköistä, eikä järjestelmä olisi välttämättä kestänyt sääolosuhteiden vaihteluita. Tästä syystä tutkimuskysymystä piti hieman muuttaa suuntaa, jossa pohdittiin yleisellä tasolla tutkimuksen aiheena olleen järjestelmän toimivuutta.

Järjestelmän rakentamisessa ehkäpä suurin heikkous oli järjestelmän tekijän hieman puutteelliset taidot, etenkin tekoälyn tekemisen saralla ja tämä osaltaan myös vaikutti siihen, miksi tekoälymallista ei tullut niin tarkka kuin olisi ehkä voinut olla mahdollista. Etenkin heikkoudet liittyivät mallin tarkkuuden säätämiseen. Mallin tekeminen oli käytännössä katsoen jatkuvasti uusien asioiden opettelua ja tämän johdosta puutteita oli havaittavissa esimerkiksi työskentelytavoissa, datan muokkaamisessa, mallin säätämisessä oikeanlaiseksi ja esimerkiksi arvioinnissa siitä, tuliko mallista tarpeeksi tarkka. Toisaalta tämä tuskin heikentää argumenttia järjestelmän toimivuudesta, sillä oikein tehtynä, kunnollisen ohjelmistokehitysprosessin tuloksena sen toiminta pystyttäisiin varmasti tekemään hyvinkin luotettavaksi. Puutteita oli myös datan kanssa, jota käytettiin mallin kouluttamiseen. Fintrafficilta saatu data ei koskenut kevyenliikenteen tietolosuhteita, vaan lähinnä ajoneuvoliikenteen. Tämän vuoksi datassa olevat parametrit saattoivat poiketa suurestikin siitä, millaiset olosuhteet ovat normaalisti kevyenliikenteenväylillä. Paras skenaario olisi ollut jos kevyenliikenteen tieolosuhteista olisi ollut valmista dataa tarjolla tarvittavine parametreineen, mutta tällaista data-

settiä ei ollut mistään saatavilla. Tästä syystä dataa piti muokata sopivampaan muotoon ja sen vuoksi sen autenttisuus saattoi kärsiä. Myöskin yleisesti katsoen olisi todennäköisesti ollut parasta, jos datasetti olisi koskenut suoraan sitä asiaa, jota varten malli haluttiin rakentaa. Lisäksi olisi ehkä ollut tarkoituksen mukaista kirjata ylös suuntaa antavia arvoja siitä, millaisia lukemia tekoälymallin olisi voinut odottaa antavan. Näin olisi ehkä entisestään helpottunut mallin tarkkuuden arviointi.

Ongelmista huolimatta ja osaksi myös niiden takia on tämän tutkimuksen pohjalta nähtävissä hyvinkin paljon potentiaalia jatkotutkimukselle monelta eri kannalta. Dataan ja tekoälyyn liittyen tutkimusta voisi jatkaa siltä näkökannalta, että mitkä tekijät erityisesti vaikuttavat liukkauden muodostumiseen keyvenliikenteenväylillä ja mitä parametreja käyttäen tekoälyn päätelmät saataisiin mahdollisimman tarkoiksi. Liukkaiden olosuhteiden muodostuminen riippuu hyvin monesta tekijästä, eikä aina ole kovinkaan mielekästä ehkä kerätä dataa näistä kaikista tekijöistä. Voi myös olla tilanne, jossa ei yksinkertaisesti ole mahdollista kerätä dataa näistä kaikista tekijöistä. Olisi siis mielenkiintoista selvittää, mitkä olisivat ne optimaalisimmat parametrit, joilla liukkautta voitaisiin ennustaa mahdollisimman tarkasti. Jatkotutkimusta voisi myös suorittaa siltä osin, että mihin kohtiin dataa keräävät sensorit olisi järkevintä asetella, jotta niillä saataisiin parhaimmat mahdolliset tulokset. Esimerkiksi, mikä on optimaalisin etäisyys maanpinnasta maanpinnan lämpötilaa mittaavalla sensorilla. Toinen vaihtoehto toki olisi vaihtaa tämä sensori kokonaan ja korvata esimerkiksi maanpinnan lämpötilaa mittaava sensori maahan upotettavalla sensorilla, jolla voitaisiin todennäköisesti saada vieläkin tarkempia tuloksia. Myöskin ilmankosteutta ja ilmanlämpötilaa mittaavan sensorin optimaalista sijaintia olisi mielenkiintoista selvittää. Vielä yksi mahdollinen jatkotutkimusaihe voisi olla järjestelmän toimintakyvyn varmistaminen niin monelta osin kuin vain mahdollista. Koska järjestelmä tulisi varsinaisessa käytössä todennäköisesti altistumaan luonnonvoimille, olisi hyvä selvittää miten sen voisi suojata parhaiten erilaisilta sääolosuhteilta. Lisäksi virran kulutusta olisi myös hyvä selvittää, millainen virtalähde toimisi parhaiten. Kaikesta huolimatta voidaan kuitenkin sanoa, että tällä tutkimuksella saatiin aikaan tuloksia. Tutkimuksen hiekkalaatikkoversio sovelluksesta onnistui keräämään dataa ympäristöstä ja prosessoimaan sitä niin kuin oli ollut tarkoituskin. Näin ollen, kun tässä tutkimuksessa rakennettua sovellusta hiottaisiin vielä hyvän ohjelmistokehitysprosessin kautta, voitaisiin sellainen varmasti saada käyttöön myös oikean elämän ympäristöön ja sen avulla

voitaisiin varmasti tuottaa hyödyllistä tietoa eri tahojen käyttöön esimerkiksi tämän tutkimuksen käyttötarkoituksen tiimoilta, tai jossakin muussa kontekstissa.

Lähteet

Akyildiz, Ian F, Weilian Su, Yogesh Sankarasubramaniam ja Erdal Cayirci. 2002. “Wireless sensor networks: a survey”. *Computer networks* 38 (4): 393–422.

Al-Alawi, Adel Ismail. 2006. “WiFi technology: Future market challenges and opportunities”. *Journal of computer science* 2 (1): 13–18.

Albino, Vito, Umberto Berardi ja Rosa Maria Dangelico. 2015. “Smart cities: Definitions, dimensions, performance, and initiatives”. *Journal of urban technology* 22 (1): 3–21.

Arduino. n.d. “Arduino Nano 33 BLE Sense”. Viitattu 22. maaliskuuta 2022. https://docs.arduino.cc/hardware/nano-33-ble-sense.

Bakıcı, Tuba, Esteve Almirall ja Jonathan Wareham. 2013. “A smart city initiative: the case of Barcelona”. *Journal of the knowledge economy* 4 (2): 135–148.

Barrionuevo, Juan M, Pascual Berrone ja Joan E Ricart. 2012. “Smart cities, sustainable progress”. *Iese Insight* 14 (14): 50–57.

Bhagwat, Pravin. 2001. “Bluetooth: technology for short-range wireless apps”. *IEEE Internet Computing* 5 (3): 96–103.

Bisdikian, Chatschik. 2001. “An overview of the Bluetooth wireless technology”. *IEEE Communications magazine* 39 (12): 86–94.

Blanco-Filgueira, Beatriz, Daniel Garcia-Lesta, Mauro Fernández-Sanjurjo, Victor Manuel Brea ja Paula Lopez. 2019. “Deep learning-based multiple object visual tracking on embedded system for iot and mobile edge computing applications”. *IEEE Internet of Things Journal* 6 (3): 5423–5431.

Bock, Yorick de, Bart Braem, Dragan Subotic, Maarten Weyn ja Johann M. Marquez-Barja. 2019. “Demo Abstract: Crowd analysis with infrared sensor arrays on the smart city edge”. Teoksessa *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS),* 983–984. https://doi.org/10.1109/INFCOMW.2019.8845075.

Boyle, David, ja Thomas Newe. 2008. “Securing Wireless Sensor Networks: Security Architectures.” *J. Networks* 3 (1): 65–77.

Brandalero, Marcelo, Mitko Veleski, Hector Gerardo Muñoz Hernandez, Muhammad Ali, Laurens Le Jeune, Toon Goedemé, Nele Mentens ym. 2021. “AITIA: Embedded AI Techniques for Industrial Applications”. Teoksessa *2021 31st International Conference on Field-Programmable Logic and Applications (FPL),* 374–375. https://doi.org/10.1109/FPL53798.2021.00071.

“Capital Expenditure (CapEx)”. 2021. https://www.investopedia.com/terms/c/capitalexpenditure.asp.

Caprolu, Maurantonio, Roberto Di Pietro, Flavio Lombardi ja Simone Raponi. 2019. “Edge Computing Perspectives: Architectures, Technologies, and Open Security Issues”. Teoksessa *2019 IEEE International Conference on Edge Computing (EDGE),* 116–123. https://doi.org/10.1109/EDGE.2019.00035.

Caragliu, Andrea, Chiara Del Bo ja Peter Nijkamp. 2011. “Smart cities in Europe”. *Journal of urban technology* 18 (2): 65–82.

Cretu, Liviu-Gabriel. 2012. “Smart cities design using event-driven paradigm and semantic web”. *Informatica Economica* 16 (4): 57.

Dash, Sanjit Kumar, Subasish Mohapatra ja Prasant Kumar Pattnaik. 2010. “A survey on applications of wireless sensor network using cloud computing”. *International Journal of Computer Science & Emerging Technologies* 1 (4): 50–55.

De Jong, Kenneth A, William M Spears ym. 1989. “Using genetic algorithms to solve NP-complete problems.” Teoksessa *ICGA,* 124–132.

Dilley, John, Bruce Maggs, Jay Parikh, Harald Prokop, Ramesh Sitaraman ja Bill Weihl. 2002. “Globally distributed content delivery”. *IEEE Internet Computing* 6 (5): 50–58.

Du, Ke-Lin, ja Madisetti NS Swamy. 2010. *Wireless communication systems: from RF subsystems to 4G enabling technologies.* Cambridge University Press.

Fintraffic. n.d. “Digitraffic Road API”. Viitattu 21. helmikuuta 2022. https://www.digitraffic.fi/en/.

Gedeon, Julien, Florian Brandherm, Rolf Egert, Tim Grube ja Max Mühlhäuser. 2019. “What the fog? edge computing revisited: Promises, applications and future challenges”. *IEEE Access* 7:152847–152878.

Giffinger, Rudolf, Christian Fertner, Hans Kramar, Evert Meijers ym. 2007. “City-ranking of European medium-sized cities”. *Cent. Reg. Sci. Vienna UT,* 1–12.

Goldsmith, Andrea. 2005. *Wireless communications.* Cambridge university press.

Gomez, Carles, Stefano Chessa, Anthony Fleury, George Roussos ja Davy Preuveneers. 2019. “Internet of Things for enabling smart environments: A technology-centric perspective”. *Journal of Ambient Intelligence and Smart Environments* 11 (1): 23–43.

Haartsen, J.C. 2000. “The Bluetooth radio system”. *IEEE Personal Communications* 7 (1): 28–36. https://doi.org/10.1109/98.824570.

Harrington, William. 2015. *Learning raspbian.* Packt Publishing Ltd.

Harrison, Colin, Barbara Eckman, Rick Hamilton, Perry Hartswick, Jayant Kalagnanam, Jurij Paraszczak ja Peter Williams. 2010. “Foundations for smarter cities”. *IBM Journal of research and development* 54 (4): 1–16.

Hippi, M. 2012. “Monitoring slipperiness on walkways”. Teoksessa *16th International Road Weather Conference, SIRWEC, Helsinki, Finland,* 23–25.

Hippi, Marjo, Markku Kangas, Reija Ruuhela, Johanna Ruotsalainen ja Sari Hartonen. 2020. “RoadSurf-Pedestrian: a sidewalk condition model to predict risk for wintertime slipping injuries”. *Meteorological Applications* 27 (5): e1955.

Judy, Jack W. 2001. “Microelectromechanical systems (MEMS): fabrication, design and applications”. *Smart materials and Structures* 10 (6): 1115.

“Kari Lukka: Konstruktiivinen tutkimusote”. 2001. Viitattu 12. tammikuuta 2022. https://metodix.fi/2014/05/19/lukka-konstruktiivinen-tutkimusote/.

Lee, Yen-Lin, Pei-Kuei Tsung ja Max Wu. 2018. “Techology trend of edge AI”. Teoksessa *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT),* 1–2. https://doi.org/10.1109/VLSI-DAT.2018.8373244.

Lépy, Élise, Sinikka Rantala, Antti Huusko, Pentti Nieminen, Marjo Hippi ja Arja Rautio. 2016. “Role of winter weather conditions and slipperiness on tourists’ accidents in finland”. *International journal of environmental research and public health* 13 (8): 822.

Lewis, Franck L, ym. 2004. “Wireless sensor networks”. *Smart environments: technologies, protocols, and applications* 11:46.

“Liberty”. 2020. Viitattu 26. lokakuuta 2021. https://www.liberty-ind.com/blog/controlled-environment-vs-cleanroom/.

Maluf, Nadim, ja Kirt Williams. 2004. *An introduction to microelectromechanical systems engineering.* Artech House.

Mekki, Kais, Eddy Bajic, Frederic Chaxel ja Fernand Meyer. 2018. “Overview of cellular LPWAN technologies for IoT deployment: Sigfox, LoRaWAN, and NB-IoT”. Teoksessa *2018 ieee international conference on pervasive computing and communications workshops (percom workshops),* 197–202. IEEE.

———. 2019. “A comparative study of LPWAN technologies for large-scale IoT deployment”. *ICT express* 5 (1): 1–7.

Nugent, C.D., S.I. McClean, I. Cleland ja W. Burns. 2014. “13.18 - Sensor Technology for a Safe and Smart Living Environment for the Aged and Infirm at Home”. Teoksessa *Comprehensive Materials Processing,* toimittanut Saleem Hashmi, Gilmar Ferreira Batalha, Chester J. Van Tyne ja Bekir Yilbas, 459–472. Oxford: Elsevier. ISBN: 978-0-08-096533-8. https://doi.org/https://doi.org/10.1016/B978-0-08-096532-1.01319-4. https://www.sciencedirect.com/science/article/pii/B9780080965321013194.

Pandey, Abhishek, ja RC Tripathi. 2010. “A survey on wireless sensor networks security”. *International Journal of Computer Applications* 3 (2): 43–49.

Park, Jihong, Sumudu Samarakoon, Mehdi Bennis ja Mérouane Debbah. 2019. “Wireless network intelligence at the edge”. *Proceedings of the IEEE* 107 (11): 2204–2239.

Powell, Alison. 2008. “WiFi publics: producing community and technology”. *Information, communication & society* 11 (8): 1068–1088.

Premsankar, Gopika, Mario Di Francesco ja Tarik Taleb. 2018. “Edge computing for the Internet of Things: A case study”. *IEEE Internet of Things Journal* 5 (2): 1275–1284.

Prince, J Dale. 2011. “Introduction to cloud computing”. *Journal of Electronic Resources in Medical Libraries* 8 (4): 449–458.

Qian, Ling, Zhiguo Luo, Yujian Du ja Leitao Guo. 2009. “Cloud computing: An overview”. Teoksessa *IEEE International Conference on Cloud Computing,* 626–631. Springer.

Qiu, Tie, Jiancheng Chi, Xiaobo Zhou, Zhaolong Ning, Mohammed Atiquzzaman ja Dapeng Oliver Wu. 2020. “Edge Computing in Industrial Internet of Things: Architecture, Advances and Challenges”. *IEEE Communications Surveys Tutorials* 22 (4): 2462–2488. https://doi.org/10.1109/COMST.2020.3009103.

Rausch, Thomas, Stefan Nastic ja Schahram Dustdar. 2018. “Emma: Distributed qos-aware mqtt middleware for edge computing applications”. Teoksessa *2018 IEEE International Conference on Cloud Engineering (IC2E),* 191–197. IEEE.

Reiter, Andreas, Bernd Prünster ja Thomas Zefferer. 2017. “Hybrid Mobile Edge Computing: Unleashing the Full Potential of Edge Computing in Mobile Device Use Cases”. Teoksessa *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID),* 935–944. https://doi.org/10.1109/CCGRID.2017.125.

Rimal, Bhaskar Prasad, Eunmi Choi ja Ian Lumb. 2009. “A taxonomy and survey of cloud computing systems”. Teoksessa *2009 Fifth International Joint Conference on INC, IMS and IDC,* 44–51. Ieee.

Ruuvi. 2016. “Ruuvi”. Viitattu 23. maaliskuuta 2022. https://ruuvi.com/fi/.

Sarker, Iqbal H, Md Hasan Furhad ja Raza Nowrozy. 2021. “Ai-driven cybersecurity: an overview, security intelligence modeling and research directions”. *SN Computer Science* 2 (3): 1–18.

Satyanarayanan, Mahadev. 2017. “The emergence of edge computing”. *Computer* 50 (1): 30–39.

Shaheen, Mohammed Yousef. 2021. “Applications of Artificial Intelligence (AI) in healthcare: A review”. *ScienceOpen Preprints.*

Shi, Weisong, ja Schahram Dustdar. 2016. “The promise of edge computing”. *Computer* 49 (5): 78–81.

Shi, Yuanming, Kai Yang, Tao Jiang, Jun Zhang ja Khaled B. Letaief. 2020. “Communication-Efficient Edge AI: Algorithms and Systems”. *IEEE Communications Surveys Tutorials* 22 (4): 2167–2191. https://doi.org/10.1109/COMST.2020.3007787.

Stanoevska-Slabeva, Katarina, ja Thomas Wozniak. 2010. “Cloud basics–an introduction to cloud computing”. Teoksessa *Grid and cloud computing,* 47–61. Springer.

Toivonen, Erika, Marjo Hippi, Hannele Korhonen, Ari Laaksonen, Markku Kangas ja Joni-Pekka Pietik"ainen. 2019. “The road weather model RoadSurf (v6. 60b) driven by the regional climate model HCLIM38: evaluation over Finland”. *Geoscientific Model Development* 12 (8): 3481–3501.

Tse, David, ja Pramod Viswanath. 2005. *Fundamentals of wireless communication.* Cambridge university press.

Tzivaras, Vasilis. 2017. *Raspberry Pi Zero W Wireless Projects.* Packt Publishing Ltd.

Vaishya, Raju, Mohd Javaid, Ibrahim Haleem Khan ja Abid Haleem. 2020. “Artificial Intelligence (AI) applications for COVID-19 pandemic”. *Diabetes & Metabolic Syndrome: Clinical Research & Reviews* 14 (4): 337–339.

Zhou, Zhi, Xu Chen, En Li, Liekang Zeng, Ke Luo ja Junshan Zhang. 2019. “Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing”. *Proceedings of the IEEE* 107 (8): 1738–1762. https://doi.org/10.1109/JPROC.2019.2918951.

Liitteet

A Skripti paramtrien eristämiseksi datasta

```
const raw_data = require('./data.json')
const weather_stations = raw_data.weatherStations

const values = weather_stations.map(x => x.sensorValues)

const dewpoints = values.map(value => {
  return value.filter(x => x.name == "KASTEPISTE")
})

const groundtemp = values.map(value => {
  return value.filter(x => x.name == "MAA_1"
})

for(let value in dewpoints) {
  try {
    console.log(dewpoints[value][0].sensorvalue)
  } catch (error) {
    continue
  }
}

for(let value in groundtemp) {
  try {
    console.log(groundtemp[value][0].sensorvalue)
  } catch (error) {
    continue
  }
}
```

B Raspberry PI:n run.py

```
#Startpoint of the python program
import logging
import struct
import time
import asyncio
from bleak import BleakScanner
from bleak import BleakClient

#function imports
from get_ruuvitag_data import get_data
from utils.calc_dew_point import calculate
from utils.calc_slope import calc_slope

#MAC addresses of the Ruuvitag sensors
macs = ['EA:33:0E:59:73:83', 'C5:74:D0:24:B4:77']
async def main():
    logging.basicConfig(filename='data.log', format='%(asctime)s
        %(message)s', datefmt='%d/%m/%Y %I:%M:%S %p',
        encoding='utf-8', level=logging.INFO)
    while True:
        readings = get_data(macs)
        try:
            dewpoint =
                calculate(readings['EA:33:0E:59:73:83']['humidity'],
                readings['EA:33:0E:59:73:83']['temperature'])
            slope = calc_slope(dewpoint,
                readings['C5:74:D0:24:B4:77']['temperature'])

            dewpoint_to_send = bytearray(struct.pack("f", dewpoint))
            slope_to_send = bytearray(struct.pack("f", slope))
            temp_to_send = bytearray(struct.pack("f", dewpoint,
                readings['C5:74:D0:24:B4:77']['temperature']))
```

```
            address = "93:3e:66:b9:26:fb"
            async with BleakClient(address) as client:

                await
                    client.write_gatt_char("19b10001-e8f2-537e-4f6c-d104768a1215"
                time.sleep(2)
                await
                    client.write_gatt_char("19b10001-e8f2-537e-4f6c-d104768a1214"
                time.sleep(2)
                await
                    client.write_gatt_char("19b10001-e8f2-537e-4f6c-d104768a1216"

                time.sleep(5)

                raw_inference = await
                    client.read_gatt_char("19b10001-e8f2-537e-4f6c-d104768a1217")
                inference = struct.unpack('f', raw_inference)

                logging.info("Dewpoint value: %.2f", dewpoint)
                logging.info("Temp value: %.2f",
                    readings['C5:74:D0:24:B4:77']['temperature'])
                logging.info("Slope value: %.2f", slope)
                logging.info("Inference value: %.2f", inference[0])
        except:
            logging.info("Error occurred")

        logging.info("Going to sleep for one hour")
        time.sleep(3600)
asyncio.run(main)
```

C Raspberry PI:n koodi datan hakemiseksi RuuviTageilta

```
from ruuvitag_sensor.ruuvi import RuuviTagSensor

def get_data(macs):
   timout_in_sec = 20
   readings = RuuviTagSensor.get_data_for_sensors(macs,
      timeout_in_sec)
   return readings
```

D Raspberry PI:n koodi kastepisteen laskemiseksi

```
def calculate(humidity, temperature):
   return temperature - ((100- humidity) / 5)
```

E Raspberry PI:n koodi kulmakertoimen laskemiseksi

```
def calc_slope(dewpoint, temperature):
   return (dewpoint - 0) / (0 - (temperature))
```

F Arduinon koodi

```
#include <ArduinoBLE.h>

#include <TensorFlowLite.h>
#include <tensorflow/lite/micro/all_ops_resolver.h>
#include <tensorflow/lite/micro/micro_error_report.h>
#include <tensorflow/lite/micro/micro_interpreter.h>
#include <tensorflow/lite/schema/schema_generated.h>
#include <tensorflow/lite/version.h>

#include "model.h"
```

```
tflite::MicroErrorReporter tflErrorReporter;
tflite::AllOpsResolver tflOpsResolver;

const tflite::Model* tflModel = nullptr;
tflite::MicroInterpreter* tflInterpreter = nullptr;
TfLiteTensor* tflInputTensor = nullptr;
TfLiteTensor* tflOutputTensor = nullprt;

constexpr int tensorArenaSize = 8 * 2949;
byte tensorArena[tensorArenaSize] __attribute__((aligned(16)));

const char* FEATURES[] = {
   "dewpoint",
   "ground_temp",
   "slope"
};

float dewpoint;
float temp;
float slope;
float inference;

#define NUM_FEATURES (sizeof(FEATURES) / sizeof(FEATURES[0]))

const char* deviceServiceUuid =
   "19b10000-e8f2-537e-4f6c-d104768a1213";
const char* temperatureServiceCharacteristicUuid =
   "19b10000-e8f2-537e-4f6c-d104768a1214";
const char* dewpointServiceCharacteristicUuid =
   "19b10000-e8f2-537e-4f6c-d104768a1215";
const char* slopeServiceCharacteristicUuid =
   "19b10000-e8f2-537e-4f6c-d104768a1216";
```

```
const char* inferenceServiceCharacteristicUuid =
   "19b10000-e8f2-537e-4f6c-d104768a1217";

BLEService* slipperyService = nullptr;
BLEFloatCharacteristic* rxTemperatureSlipperyCharacteristic =
   nullptr;
BLEFloatCharacteristic* rxDewpointSlipperyCharacteristic = nullptr;
BLEFloatCharacteristic* rxSlopeSlipperyCharacteristic = nullptr;
BLEFloatCharacteristic* txInferenceSlipperyServiceCharacteristic =
   nullptr;

void setup() {
   slipperyService = new BLEService(deviceServiceUuid);
   rxTemperatureSlipperyCharacteristic =
      newBLEFloatCharacteristic(temperatureServiceCharacteristicUuid,
      BLERead | BLEWrite | BLEWriteWithoutResponse);
   rxDewpointSlipperyServiceCharacteristic = new
      BLEFloatCharacteristic(dewpointServiceCharacteristicUuid,
      BLERead | BLEWrite | BLEWriteWithoutResponse);
   rxSlopeSlipperyServiceCharacteristic = new
      BLEFloatCharacteristic(slopeServiceCharacteristicUuid,
      BLERead | BLEWrite | BLEWriteWithoutResponse);
   rxInferenceSlipperyServiceCharacteristic = new
      BLEFloatCharacteristic(inferenceServiceCharacteristicUuid,
      BLERead | BLEWrite | BLEWriteWithoutResponse);

   Serial.begin(9600);
   while(!Serial);

   Serial.println("+++ Initializing TFLite model +++");
   tflModel = tflite::GetModel(model);
   if(tflModel->version() != TFLITE_SCHEMA_VERSION) {
      Serial.println("Model schema mismatch!");
```

```
    while(1);
}
Serial.println("+++ Success! +++");

tflInterpreter = new tflite::MicroInterpreter(tflModel,
    tflOpsResolver, tensorArena, tensorArenaSize,
    &tflErrorReporter);

tflInterpreter->AllocateTensors();
tflInputTensor = tflInterpreter->input(0);
tflOutputTensor = tflInterpreter->output(0);

if(!BLE.begin()) {
    Serial.println("- Starting Bluetooth Low Energy module
        failed");
    while(1);
}

BLE.setAdvertisedService(*slipperyService);
slipperyService->addCharacteristic(*rxTemperatureSlipperyCharacteristic);
slipperyService->addCharacteristic(*rxDewpointSlipperyServiceCharacterist
slipperyService->addCharacteristic(*rxSlopeSlipperyServiceCharacteristic)
slipperyService->addCharacteristic(*rxInferenceSlipperyServiceCharacteris
BLE.addService(*slipperyService);
BLE.setLocalName("Arduino Nano 33 BLE Sense (Peripheral)");

BLE.setEventhandler(BLEConnected, onBLEConnected);
BLE.setEventHandler(BLEDisconnected, onBLEDisconnected);

rxDewpointSlipperyServiceCharacteristic->setEventHandler(BLEWritte,
    onDewpointCharacteristicWrite);
rxTemperatureSlipperyCharacteristic->setEventHandler(BLEWritten,
    onTempCharacteristicWrite);
```

```
    rxSlopeSlipperyServiceCharacteristic->setEventHandler(BLEWritten,
        onSlopeCharacteristicWritten);

    BLE.advertise();

    Serial.println("Peripheral advertising info: ");
    Serial.print("MAC: ");
    Serial.println(BLE.address());
    Serial.print("Service UUID: ");
    Serial.println(slipperyService->uuid());
    Serial.println();
    Serial.println("+++ Discovering central device... +++");
}

void loop() {
    BLEDevice central = BLE.central();
}

void onDewpointCharacteristicWrite(BLEDevice central,
    BLECharacteristic characteristic) {
    dewpoint = rxDewpointSlipperyServiceCharacteristic->value();
    tflInputTensor->data.f[0] = dewpoint;
}

void onTempCharacteristicWrite(BLEDevice central,
    BLECharacteristic characteristic) {
    temp = rxTemperatureSlipperyCharacteristic->value();
    tflInputTensor->data.f[1] = temp;
}

void onSlopeCharacteristicWrite(BLEDevice central,
    BLECharacteristic characteristic) {
    slope = rxSlopeSlipperyServiceCharacteristic->value();
```

```
    tflInputTensor->data.f[2] = slope;
    makeInference();
}

void makeInference() {
    Serial.println("Performing inference");
    inference = tflOutputTensor->data.f[0];
    txInferenceSlipperyServiceCharacteristic->writeValue(inference);
}

void onBLEConnected(BLEDevice central) {
    Serial.println("* Connected to central device!");
    Serial.print("* Device MAC address: ");
    Serial.println(central.address());
}

void onBLEDisconnected(BLEDevice central) {
    Serial.println("* Disconnected from central device!");
    Serial.print("* Device MAC address: ");
    Serial.println(central.address());
}
```