Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2019

Ilkka Kananen

MUSTAN JÄÄN HAVAITSEMISJÄRJESTELMÄ

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

2019 | 32 sivua

Ilkka Kananen

MUSTAN JÄÄN HAVAITSEMISJÄRJESTELMÄ

Opinnäytetyön tavoitteena oli selvittää, pystyykö mustaa jäätä havaitsemaan mittaamalla kosteutta ja tienpinnan lämpötilaa. Kosteuden mittaamisessa käytettiin sadeanturia ja lämpötilan mittaamisessa käytettiin infrapuna-anturia, joka kohdennettiin mitattavaan tienpintaan. Nämä kaksi anturia yhdistettiin Raspberry Pi -mikrotietokoneeseen, joka sijoitettiin ulos tekemään mittauksia. Mittalaitteen tarkoitus on havaita jään muodostumista yhdessä mittauspisteessä kerrallaan. Tämä laite voidaan asentaa mihin vain, mistä se saa yhteyden internetiin.

Yksi mittauspiste ei välttämättä riitä, sillä olosuhteet voivat vaihdella pitkillä välimatkoilla tai mittauspisteeseen voi vaikuttaa jokin mittauksiin kuulumaton tekijä. Laajemman alueen havaitsemiseen päätettiin käyttää Ilmatieteen laitoksen avointa dataa. Tallennettuja kyselyitä käyttämällä voidaan pyytää esimerkiksi havaintoasemien säähavaintoja avoimen datan palvelimelta. Parametrit, jotka olivat tämän työn kannalta oleellisia, olivat ilman lämpötila, sateen määrä ja kastepistelämpötila eli lämpötila, jossa kosteuden tiivistyminen alkaa.

Työn tuloksena voidaan todeta, että tähän tarkoitukseen rakennettu mittalaite pystyy havaitsemaan mustan jään muodostumista. Ilmatieteen laitoksen datasta oli vaikea päätellä mustan jään muodostumista, koska ei voinut tietää, kuinka paljon kosteutta on tienpinnassa. Myös tienpinnan lämpötilan mittaaminen on oleellista, ja avoimen datan palvelusta sai haettua ainoastaan ilman lämpötilan. Jos järjestelmää laajentaisi suuremmalle alueelle, useamman mittalaitteen rakentaminen ja sijoittaminen sopiviin paikkoihin tuottaisi huomattavasti tarkempia tuloksia. Ilmatieteen laitoksen dataa voisi kuitenkin hyödyntää pyrkimällä ennustamaan mustan jään muodostumista ja varmistamaan tämä mittalaitteella.

ASIASANAT:

sääilmiöt, ohjelmointi, Raspberry Pi, Python, Javascript

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communication Technology

2019 | 32 pages

Ilkka Kananen

BLACK ICE DETECTION SYSTEM

The objective of this thesis was to find out if it's possible to detect the formation of black ice by measuring moisture and the temperature of a road surface. A rain sensor was used to measure moisture and an infrared sensor was used to measure the road surface's temperature. These two sensors were connected to a Raspberry Pi which was placed outside to perform the measurements. The purpose of the measuring device is to detect the formation of ice in a single measurement point at a time and it can be placed anywhere it connect to the internet.

One measurement point may not be enough though as the conditions can change over long distances or there may be some external factor influencing the measurements. The Finnish Meteorological Institute's open data service was used to measure a wider area. Stored queries can be used to fetch data from observation stations. Relevant parameters for this thesis were air temperature, precipitation amount and dew point temperature, which is the temperate at which moisture begins to condense.

The result of this thesis is that the measuring device can detect the formation of black ice. It was difficult to estimate the formation of ice based on the data from the Finnish Meteorological Institute as there was no way to know the amount of moisture on the ground. Also, knowing the road surface's temperature is important and it was only possible to get the air temperature from the open data service. If this system were to be expanded it would produce much better results to build several measuring devices than to use the open data service. One way to make use of the open data service would be to try to forecast the formation of black ice and then use the measuring device to confirm this.

KEYWORDS:

weather phenomena, programming, Raspberry Pi, Python, Javascript

SISÄLTÖ

KÄYTETYT LYHENTEET JA SANASTO

ADC	muuntaa analogisen signaalin digitaaliseksi signaaliksi (analog-to-digital converter)
AJAX	tekniikka, jolla voidaan hakea ja pyytää dataa palvelimelta asynkronisesti (Asynchronous Javascript and XML)
GPIO	pinni piirilevyssä, joka voi toimia lähtönä tai tulona ja sen käyttötarkoitus voidaan vapaasti määritellä (general-purpose input/output)
HTML	merkintäkieli, jolla kuvataan internetsivujen rakennetta (hypertext markup langueage)
I2C	mikropiirien ja mikrokontrollerien väliseen kommunikointiin tarkoitettu väylä (inter-integrated circuit)
Javascript	internetsivuilla käytettävä ohjelmointikieli, jota voidaan käyttää HTML-tiedoston sisällä lisätäkseen siihen toiminnallisuutta
Node.js	ympäristö, jossa voidaan suorittaa Javascript-koodia palvelimella, toisin kuin yleensä selaimen sisällä
PWM	menetelmä, jossa signaali voidaan esittää digitaalisessa muodossa muuttamalla aikaa, jolloin se on päällä suhteessa aikaan, jolloin se on pois päältä (pulse-width modulation)
Python	tulkattava ohjelmointikieli, jolla on monta eri käyttökohdetta
SPI	kahden laitteen väliseen kommunikointiin tarkoitettu synkroninen sarjaliitäntä (serial peripheral interface)
SSH	protokolla, jolla voidaan muodostaa suojattu yhteys asiakkaan ja palvelimen välille (Secure Shell)

1 JOHDANTO

Syksyisin, yöllisten lämpötilojen laskiessa pakkaselle voi kohdata vaarallisen ilmiön: mustan jään. Mustalla jäällä tarkoitetaan ohutta kerrosta jäätä, joka muodostuu tienpintaan. Koska jääpeite on ohut ja läpinäkyvä, se näyttää mustalta jääpeitteen alla olevan tien mustan värin takia. Tienpinta, johon on muodostunut mustaa jäätä, on vaikea erottaa tienpinnasta, joka ei ole jäässä. [1.] Jäätyneellä tiellä on hyvin vähän pitoa ja jalankulkijan tai pyöräilijän on helppo liukastua tai autoilijan menettää auton hallinta, ja koska mustaa jäätä on vaikea havaita, se voi yllättää tiellä liikkujan. Tämän takia musta jää on vaarallinen ilmiö.

Tämän opinnäytetyön tavoitteena on rakentaa laite, joka pystyy havaitsemaan mustan jään muodostumista mittaamalla kosteuden määrää ja tien lämpötilaa. Laite päätettiin rakentaa Raspberry Pi -mikrotietokonetta käyttäen. Sen hyötyjä ovat monipuolinen Linux-käyttöjärjestelmä ja mahdollisuus internetyhteyteen. Laitteen keräämää dataa voi selata mittauksien aikana ottamalla etäyhteys laitteeseen. Etäyhteyden muodostamiseen voi käyttää esimerkiksi SSH:ta ja PuTTY-ohjelmaa.

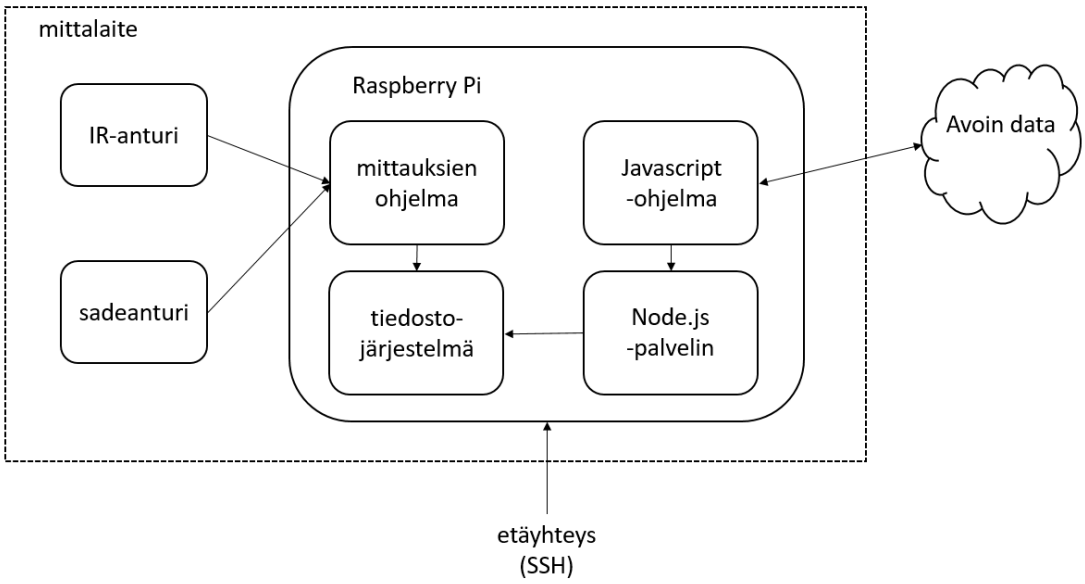
Pelkästään yhden laitteen keräämä data ei välttämättä riitä. Olosuhteet voivat vaihdella muutamankin kilometrin päästä mittauspisteestä. Esimerkiksi sadepilvi voi kulkea juuri mittauspisteen ohi ja lämpötilan pienet erot voivat olla merkittäviä pakkasrajan tuntumassa. On myös mahdollista, että mittauspisteeseen vaikuttaa jokin ulkopuolinen tekijä. Jokin ylimääräinen lämmönlähde voi vaikuttaa lämpötilan mittaamiseen ja vääristää tuloksia. Tämän takia olisi hyvä olla useampi mittauspiste.

Dataa pitäisi siis pystyä keräämään useasta eri paikasta. Useamman mittalaitteen rakentaminen ja niiden asentaminen sopiviin paikkoihin olisi yksi ratkaisu. Tässä työssä päätettiin kuitenkin hyödyntää Ilmatieteen laitoksen avoimen datan rajapintaa. Avoimen datan palvelimelta voi hakea säähavaintoja ja ennusteita Javascript-kirjaston avulla. Ilmatieteen laitoksen dataa verrataan mittalaitteen keräämään dataan. Tästä pyritään päättelemään, olisiko toimivampi ratkaisu luoda ohjelma, joka hakee dataa usealta eri havaintoasemalta, vai rakentaa useampi mittalaite ja kerätä niistä saatu data samaan paikkaan.

Jotta Ilmatieteenlaitokselta saatu data pystytään tallentamaan tiedostojärjestelmään, täytyy käynnistää palvelin ja lähettää sille data http-pyynnöillä. Tämä toteutettiin node.js-

ympäristöllä, jossa Raspberry Pi toimii palvelimena. Palvelimen käyttämisellä on myös toinen hyöty: tulokset voidaan näyttää verkkosivulla, jos on yhdistetty samaan verkkoon. Node.js voidaan määrittää kuuntelemaan Raspberry Piin IP-osoitetta, jolloin suoritetaan http-pyyntö, aina kun selaimeen kirjoitetaan osoitteeksi määritelty IP-osoite ja portin numero. Http-pyyntö määritetään lukemaan ja lähettämään sivulle tuloksien näyttämiseen tarkoitettu HTML-tiedosto. Näin voidaan helposti selata avoimen datan ja mittalaitteen tuloksia esimerkiksi älypuhelimella, kun on käynnistänyt node.js-palvelimen ja on yhdistänyt samaan verkkoon, kuin Raspberry Pi.

Yleiskaavio koko järjestelmästä on esitetty kuvassa 1. Siitä näkee järjestelmän kaksi osaa: mittalaitteen ja Ilmatieteenlaitoksen avoimen datan. Mittalaite muodostuu Raspberry Piistä ja kahdesta anturista. Raspberry Pi myös suorittaa mittauksia tekevää ohjelmaa sekä avoimen datan Javascript-ohjelmaa.



Kuva 1. Yleiskaavio järjestelmästä, johon sisältyy mittalaite ja Ilmatieteen laitoksen avoin data.

Opinnäytetyössä käydään ensin läpi, miten mustaa jäätä muodostuu ja tutkitaan olemassa olevia ratkaisuja mustan jään muodostumiseen. Myös laitteessa käytetyt komponentit ja ohjelmat esitellään. Laitteen rakentaminen koekytkentälevylle ja mittauksia tekevän ohjelman jälkeen esitetään Ilmatieteen laitoksen avointa dataa hakeva ohjelma. Lopuksi verrataan mittalaitteen ja avoimen datan tuloksia keskenään ja arvioidaan niiden soveltuvuutta tämän työn tavoitteeseen.

2 TAUSTATIETOA

2.1 Musta jää

Mustaa jäätä voi muodostua erilaisissa olosuhteissa. Yleisin tapaus on, kun sateen jälkeen maahan on jäänyt vettä ja ilman lämpötila laskee nollaan tai sen alapuolelle. Esimerkiksi jos päivällä on satanut vettä ja illalla tai yöllä lämpötila laskee pakkaselle, maahan jäänyt vesi jäätyy. Mustaa jäätä voi kuitenkin muodostua ilman sadettakin. Rännän tai sulaneen lumen jäätyminen on toinen tapaus. Aina ei tarvita sadetta eikä edes maassa olevaa räntää tai lunta, jos ilmassa oleva kosteus tiivistyy ja muodostaa kastetta ja tämä kaste jäätyy. Jotta näin tapahtuu, on tienpinnan lämpötila ja kastepiste oltava pakkasen puolella. [2.]

Kosteuden tiivistymisessä on kaksi tärkeää käsitettä: kastepiste ja suhteellinen kosteus. Kastepiste on laskennallinen lämpötila, johon nykyisen lämpötilan tulee laskea, jotta kosteus alkaisi tiivistyä. Mitä kosteampi ilma on, sitä lähempänä ilman lämpötila ja kastepiste ovat toisiaan. Kun ilman lämpötila laskee kastepisteeseen tai sen alapuolelle, ilman suhteellinen kosteusprosentti on 100 ja tiivistyminen alkaa. [3.]

Suhteellinen kosteus kertoo, kuinka paljon ilmassa on kosteutta suhteessa nykyisen lämpötilan suurimmasta mahdollisesta kosteudesta. Ilman kosteus siis riippuu lämpötilasta. Mitä kylmempi ilma on, sitä vähemmän se pystyy pidättämään kosteutta. Tämän takia talvella suhteellinen kosteus on suurempi kuin kesällä alemman lämpötilan vuoksi. [4.] Kun suhteellinen kosteusprosentti on 100 ja ilma jatkaa jäähtymistä, kosteus tiivistyy, koska suhteellinen kosteusprosentti ei voi nousta yli 100:n.

2.2 Olemassa olevat ratkaisut mustan jään havaitsemiseen

Aihetta tutkiessa kävi ilmi, että on jo olemassa järjestelmiä mustan jään tai ylipäätään liukkaan tien havaitsemiseen. Nämä kuitenkin eroavat tässä työssä tehdystä laitteesta. Seuraavaksi käydään läpi kaksi esimerkkijärjestelmää.

Ensimmäisessä esimerkissä mitataan pinnan lähellä olevaa resistanssia, joka muuttuu, kun siinä on kosteutta tai jäätä. Työssä upotettiin kaksi paria metallisia pylväitä betonista

valmistettuun sylinteriin. Näiden pylväiden välistä resistanssia ja pinnan lämpötilaa mittaamalla voidaan havaita jään muodostumista. [5.] Työssä valmistettu laite vastaa tarkoitukseltaan tässä työssä suunniteltua laitetta. Kummassakin on tavoitteena rakentaa laite, joka mittaa kosteuden muodostumista ja lämpötilaa, ja pyrkii näistä päättelemään jään muodostumista. Kosteuden mittaaminen on esimerkkityössä suoritettu eri tavalla: resistanssin muutosta mittaamalla. Tässä työssä käytetyt anturit ovat siis erilaisia, sekä Raspberry Piin käyttäminen mahdollistaa datan tallentamisen tiedostojärjestelmään.

On olemassa myös menetelmiä, jotka mittaavat reaaliajassa liikkuvan auton ympärillä olevia olosuhteita ja pyrkivät varoittamaan kuljettajaa liukkaan tien havaitessaan. Esimerkiksi VTT:n kehittämä ja patentoima liukkaudentunnistin ”arvioi vetoakselin ja vapaasti pyörivien akselien nopeuseroja erilaisissa ajotilanteissa ja päättelee kitkatason”. Tiedot teiden liukkaudesta lähetetään langattomasti taustajärjestelmään, jossa muodostetaan liukkauskartasto kerätystä datasta. Tästä kartastosta voidaan lähettää tietoja muille samassa järjestelmässä mukana oleville ajoneuvoille. Näin voidaan varoittaa etukäteen muita kuljettajia aikaisemmin tehdyistä havainnoista. [6.]

VTT:n menetelmä on tarkoitettu lähinnä autoilijoille ja se käyttää datan keräämiseen autojen omia antureita. Tässä työssä rakennetaan erillinen laite ja asennetaan se kiinteään pisteeseen suorittamaan mittauksia. Siitä saatua dataa voi hyödyntää muutkin kuin autoilijat, koska se voidaan asentaa melkein mihin vain.

2.3 Tässä työssä suunniteltu ratkaisu

Tässä työssä suunniteltu laite käyttää Raspberry Pi -mikrotietokonetta, jonka GPIO-pinneihin on yhdistetty sadeanturi ja infrapunalämpöanturi. Sadeanturissa on muutaman senttimetrin kokoinen levy, joka havaitsee siihen kertyneet vesipisarat tai kosteuden. Kosteutta tai vettä voi muodostua sateesta, ilman kosteuden tiivistymisestä tai sulaneesta lumesta. Infrapunalämpöanturi mittaa tien tai jalkakäytävän lämpötilaa ilman kontaktia mitattavan kohteen kanssa. Näin saadaan tarkka lukema kohteen lämpötilasta vaikuttamatta siihen, esimerkiksi laitteen tuottamalla lämmöllä. Valitusta lämpöanturista saa luettua myös anturin ympäröivän ilman lämpötilan.

Näitä kahta anturia käyttämällä voidaan arvioida jään muodostumisen mahdollisuutta tarkastelemalla kahta ehtoa. Jos sadeanturiin on kertynyt kosteutta ja jos lämpöanturin

mittaama tienpinta on pakkasella samaan aikaan tai hieman myöhemmin, on todennäköistä, että on muodostunut mustaa jäätä.

Laitetta ei ole tarkoitettu tekemään reaaliaikaisia mittauksia esimerkiksi liikenteessä. Ensisijainen käyttökohde on jättää laite ulos ja myöhemmin lukea mitattuja arvoja. Olennaista on, että anturit ovat ulkoilmassa ja niihin ei kohdistu ylimääräistä lämpöä mistään lähellä olevasta lähteestä, jolloin mittaukset vääristyisivät. Esimerkiksi aamulla voidaan arvioida, onko yön tai illan aikana mahdollisesti muodostunut mustaa jäätä.

3 KÄYTETYT KOMPONENTIT JA OHJELMAT

3.1 Raspberry Pi

Raspberry Pi on Raspberry Pi Foundationin luoma mikrotietokone, jonka käyttöjärjestelmänä toimii Linux. Suositeltu distribuutio on Raspbian, joka perustuu Debianiin ja on optimoitu Raspberry Piille ja sen laitteistolle. Käyttöjärjestelmä ladataan SD-kortille ja liitetään kiinni Raspberry Pihin. Kaikki tiedostot tallentuvat myös SD-kortille. Raspberry Piissä on USB Micro B -portti, josta se saa virtaa. [7.] Virtalähteenä voi siis käyttää esimerkiksi tavallista puhelinlaturia tai varavirtalähdettä.

Raspberry Piissä on myös useita muita liitäntöjä. Siinä on paikka HDMI-kaapelille, jotta siihen saa kytkettyä näytön. Sen USB portteihin voi liittää hiiren ja näppäimistön tai muita oheislaitteita. Piissä on myös ethernet-portti, josta se saa verkkoyhteyden. Uudemmissa malleissa on myös WiFi-yhteyden mahdollisuus. Nämä liitännät ja ominaisuudet antavat Raspberry Piille melkein samat perustoiminnot kuin tavallisella tietokoneella pienessä ja helposti siirrettävässä muodossa. Tämän lisäksi Raspberry Piissä on useita GPIO-pinnejä, joihin voi liittää antureita tai joista voi syöttää ulos 3,3 V:n tai 5 V:n jännitteitä.

Näiden edellä mainittujen ominaisuuksien takia Raspberry Pi valikoitui tämän työn kehitysalustaksi. GPIO-pinnejä pystyy lukemaan anturien arvoja ja niistä saatu data voidaan tallentaa SD-kortille. Koska Raspberry Piillä on oma käyttöjärjestelmä, se pystyy suorittamaan mittauksia itsenäisesti. Virtalähteenä voi käyttää puhelimille tarkoitettuja varavirtalähteitä, joilla Piitä voi pitää käynnissä useita tunteja. Jos Piille rakentaa säiliön, jossa se pysyy kuivana ja suojattuna, sen voi jättää ulos tekemään mittauksia. Tässä työssä Pi, varavirtalähde ja pieni koekytkentälevy laitettiin muovirasian sisälle ja rasian ulkopuolella olevat anturit kytkettiin koekytkentälevyyn rasiaan porattujen reikien kautta. Näin mittauksia pystyy tekemään melkein missä vain.

3.2 Sadeanturi

Sadeanturiksi tähän työhön valikoitui alun perin Arduinolle tarkoitettu FC-37 / YL-83. Aluksi kosteuden havaitsemiseen ajateltiin käyttää kosteusanturia, mutta kosteusantureiden epätarkkuus on suurta, kun suhteellinen kosteus lähestyy 100 %. Tästä syystä

kosteuden tiivistymistä olisi hankala arvioida kosteusanturilla. Sadeanturi havaitsee sateen lisäksi ilmasta tiivistyneen kosteuden.

Sadeanturissa on kaksi osaa: elektroniikkamoduuli ja sateelle altistuva nikkelipäällysteinen levy, jossa on toisista erillään olevia johtoja. Kun vettä kertyy levyn pinnalle, johtojen välinen resistanssi muuttuu. Kun levy on melkein kuiva, sen resistanssi on 2M ohmia ja sen lähtöjännite on 3,3 V tai 5 V riippuen siitä, mikä on valittu käyttöjännitteeksi. Tässä työssä käytetään 3,3 V käyttöjännitteenä. Kun levy kastuu, sen resistanssi ja lähtöjännite laskevat.

Elektroniikkamoduulissa on kaksi lähtöä: analoginen- ja digitaalinen lähtö. Digitaalilähtö kertoo, onko sadetta havaittu. Digitaalilähdön herkkyyttä (eli kuinka paljon vettä pitää kertyä levylle, jotta saadaan lähdön arvoksi 1) voidaan säätää elektroniikkamoduulin sisältämällä potentiometrillä. Analoginen lähtö kertoo, kuinka paljon kosteutta on kertynyt levylle. Tässä työssä analogilähdön tarkkaileminen on järkevämpää, sillä kosteuden tiivistyessä vettä ei kerry kovin paljon pinnalle. Ja koska pienikin määrä kosteutta voi jäätyä, pitää pystyä havaitsemaan pienetkin erot kosteudessa.

3.3 Infrapunalämpöanturi

Tässä työssä käytetään Melexiksen valmistamaa MLX90614ACF-anturia mittaamaan tien lämpötilaa. Lämpötilan mittaamiseen valikoitui infrapunalla toimiva anturi, jotta anturin tai laitteen muiden osien ei tarvitsisi olla kosketuksissa mitattavan kohteen kanssa. Näin voidaan olla varmoja, että mittalaite itsessään ei vaikuta mittaustuloksiin.

MLX90614ACF:n ominaisuudet sopivat tähän työhön hyvin. Valintakoodi ACF:n A viittaa 5 V:n käyttöjännitteeseen, joka voidaan antaa suoraan Raspberry Piin 5 V:n pinneistä. Sillä saadaan mitattua kahta eri lämpötilaa: anturin ympäröivää ilmaa ja mitattavaa kohdetta. Sen lämpötila-alue ulottuu –40 °C:seen anturin ympäröivän ilman mittaamisessa ja –70 °C:seen kohteen lämpötilan mittaamisessa. Sen mittausresoluutio on 0,02 °C ja mittaustarkkuus ±0,5 °C. [8.]

Anturin antama lämpötila mitattavalle kohteelle on keskiarvo kaikista kohteista sen näkökentässä. Valintakoodin viimeinen kirjain ilmaisee anturin näkökentän laajuuden. Tässä tapauksessa ACF:n viimeinen kirjain F tarkoittaa, että anturin näkökenttä on 10 °. [8.] Anturi siis havaitsee kaikki kohteet ja esineet 10 ° kulmassa ja laskee näiden lämpö-

tilojen keskiarvon. Kapean näkökentän ansiosta mittausalue saadaan kohdistettua tiettyyn kohteeseen ilman, että mittauksiin tulee mukaan epäolennaisten kohteiden lämpötiloja. Tämän mallin kapea näkökenttä yhdistettynä hyvään mittaustarkkuuteen takaa sen, että anturilla saadaan mitattua tarkkoja tuloksia.

Mitattuja lämpötiloja voidaan lukea joko I2C-väylän kautta tai 10-bittisenä PWM-lähtönä. PWM-tilassa oletusasetuksien lämpötila-alue ulottuu –20 °C:sta 120 °C:seen ja mittausresoluutio on 0,14 °C [8]. Lämpötila-aluetta voi rajata, jotta saadaan tarkempi resoluutio, mutta I2C-väylän antamaan 0,02 °C:n resoluutioon päästäkseen on lämpötila-alue rajattava noin 20 °C:seen ($\frac{20}{2^{10}} \approx 0{,}02$). Tämä lämpötila-alue on todennäköisesti liian kapea, joten PWM-tilaa käyttäessä pitäisi tehdä kompromissi lämpötila-alueen ja resoluution välillä. I2C-väylää käyttäessä sen sijaan saadaan oletusasetuksilla laaja lämpötila-alue ja tarkka resoluutio. Tästä syystä lämpötila arvoja päätettiin lukea I2C-väylän kautta.

3.4 ADC-muunnin

Infrapunalämpöanturin arvoja voidaan lukea suoraan Raspberry Piillä I2C-väylän kautta, mutta sadeanturin analogisten arvojen lukemiseen tarvitaan ADC-muunnin, koska Raspberry Piillä ei voi lukea analogisia arvoja. Tässä työssä käytetään ADC-muuntimena MCP 3008 -mikropiiriä. Sen resoluutio on 10 bittiä, siinä on 8 sisääntulo kanavaa ja se käyttää SPI-väylää kommunikointiin [9].

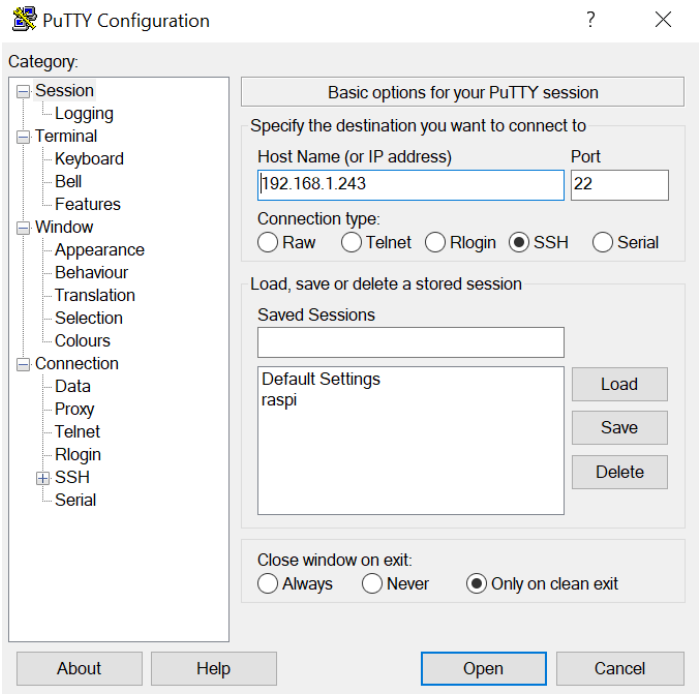
SPI-väylän arvoja voi lukea käyttämällä joko laitteistopohjaista- tai ohjelmistopohjaista SPI-väylää. Ohjelmistopohjaista SPI:tä käyttäessä voidaan käyttää mitä tahansa neljää GPIO-pinniä kommunikoimaan SPI-väylän kautta ja laitteistopohjaista SPI:tä käyttäessä täytyy käyttää neljää tiettyä GPIO-pinniä, joissa on sisäänrakennettu tuki SPI:lle. Näiden kahden merkittävin ero on väylän nopeudessa. Sadeanturin arvoja luetaan minuutin välein, joten väylän nopeudella ei tässä tapauksessa ole merkitystä. Silti tässä työssä päädyttiin käyttämään laitteistopohjaista SPI:tä, koska Raspberry Piissä on siihen sisäänrakennettu tuki ja siihen tarvittavia tiettyjä GPIO-pinnejä ei tarvittu mihinkään muuhunkaan.

3.5 SSH

SSH eli Secure Shell on protokolla, jonka avulla voidaan muodostaa suojattu yhteys asiakaslaitteen ja palvelinlaitteen välille. Kaikki asiakkaan ja palvelimen välillä liikkuva

data salataan. Yhteyden muodostus alkaa, kun asiakas ottaa yhteyden palvelimeen ja palvelin lähettää asiakkaalle julkisen avaimen. Tämän jälkeen suojattu kanava avataan ja käyttäjä kirjautuu sisään palvelimen käyttöjärjestelmään. Yhteyden muodostamisen jälkeen SSH käyttää vahvaa synkronista salausta ja tiivistysalgoritmeja varmistaakseen yksityisyyden ja eheyden kaikelle asiakkaan ja palvelimen välillä liikkuvalle datalle. [10.]

Tässä työssä SSH:ta käytetään ottamaan etäyhteys Raspberry Pihin Windows-tietokoneelta. Tässä tapauksessa Raspberry Pi toimii palvelimena ja Windows-tietokoneella käytetään PuTTY-ohjelmaa yhteyden muodostamiseen. Kuvassa 2 näkyy PuTTYn oletussivu, jossa määritetään palvelimen nimi tai IP-osoite ja portti. Nämä tiedot voi tallentaa, jotta niitä ei tarvitse kirjoittaa joka kerta uudelleen. Raspberry Piin IP-osoitteen saa tietää käyttämällä esimerkiksi ifconfig-komentoa. IP-osoite kannattaa muuttaa staattiseksi, jolloin se ei muutu. Siten SSH istunnon asetuksissa voi käyttää aina samaa osoitetta ottamaan etäyhteys. Tämä onnistuu lisäämällä staattisen osoitteen asetukset dhcpcd.conf-tiedoston loppuun, kuten esimerkiksi kuvassa 3.



Kuva 2. PuTTYn SSH istunnon määrityssivu. IP-osoitteeksi laitetaan Raspberry Piin staattiseksi määritelty osoite. SSH:ssa käytetään oletuksena porttia 22, jota ei ole tarvetta muuttaa. IP-osoitteen ja portin voi tallentaa alempana olevasta valikosta. Siitä voi myös ladata aiemmin tallennetut asetukset.

```
interface wlan0

static ip_address=192.168.1.243/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

Kuva 3. dhcpcd.conf-tiedoston loppuun voi määrittää staattisen IP-osoitteen. Tässä tapauksessa määritetään wlan-yhteydelle staattinen osoite.

Yhteys voidaan muodostaa, jos Raspberry Pi on yhdistetty samaan verkkoon kuin siihen yhteyden ottava tietokone. Sisäänkirjautumiseen voidaan käyttää oletuskäyttäjä ”pi”:n käyttäjänimeä ja salasanaa. Oletussalasana ”raspberry” kannattaa kuitenkin muuttaa johonkin turvallisempaan salasanaan ennen SSH:n käyttöönottoa. Jotta Raspberry Piitä voi käyttää etänä, on SSH otettava ensin käyttöön. Sen voi tehdä työpöydältä avaamalla ”Raspberry Pi Configuration” tai komentolinjalta komennolla raspi-config. SSH ja muut käyttöliittymät löytyvät ”Interfaces”-välilehdeltä. SSH:ta käyttäessä on käytössä vain komentolinja, joten työpöytäohjelmia tai internetselainta ei voi käyttää etänä. Tämä ei kuitenkaan ole ongelma, sillä melkein kaikki tässä työssä tehdyt toimenpiteet voi suorittaa komentolinjaa käyttämällä.

Etäyhteyden käyttäminen Raspberry Piin ohjelmointiin ja käyttöön on kätevämpää, koska siten ei tarvitse liittää Pihin näyttöä, näppäimistöä ja hiirtä aina kun halutaan tehdä muutoksia ohjelmakoodiin, käynnistää ohjelma tai tarkkailla tuloksia. Käyttämällä SSH:n etäyhteyttä voidaan asettaa laite tekemään mittauksia, tehdä muutoksia ohjelmakoodiin ja tarkkailla mittaustuloksia ilman, että laite pitäisi viedä muutosten yhteydessä takaisin sisälle.

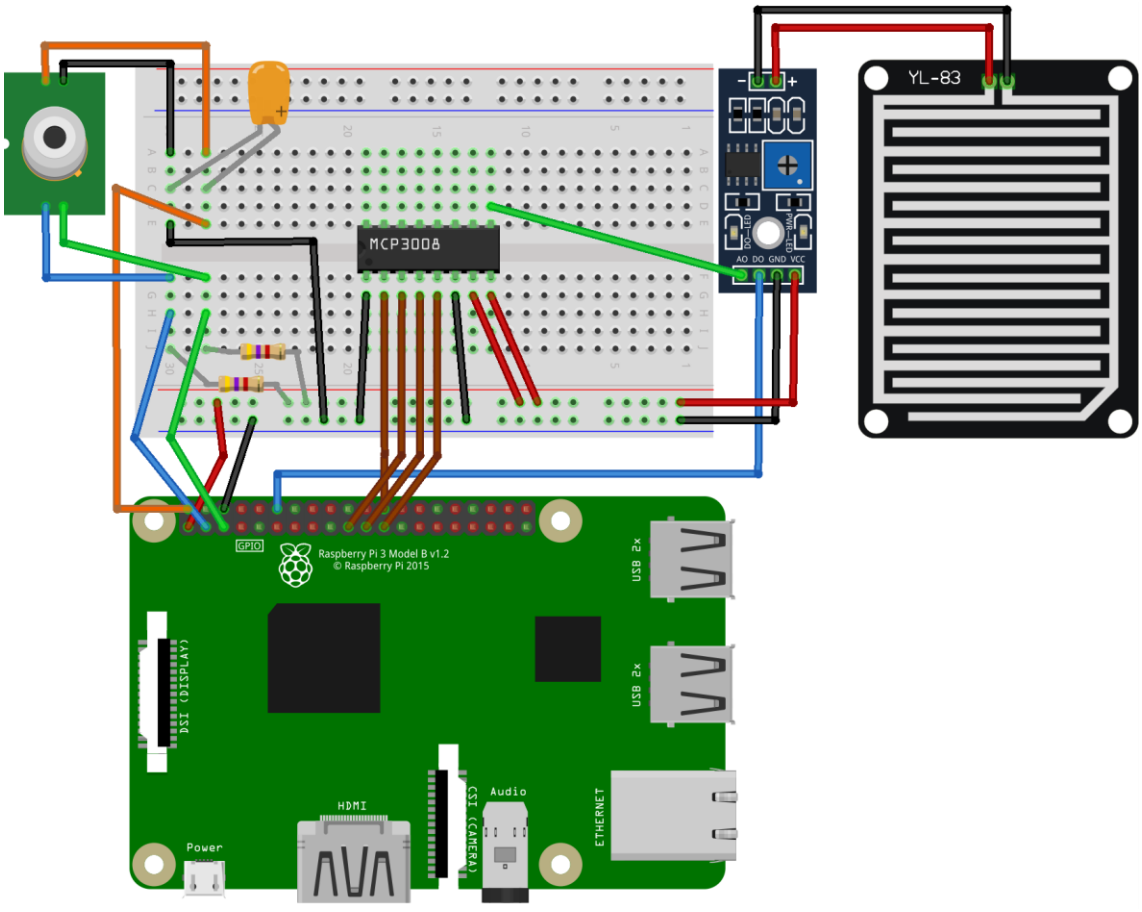
3.6 Python

Python on korkean tason, tulkattava, ohjelmointikieli. Pythonilla tehtyä lähdekoodia ei siis käännetä erikseen suoritettavaksi ohjelmatiedostoksi. Se soveltuu moneen eri tarkoitukseen monipuolisten ominaisuuksiensa ansiosta, mutta tämän työn tarkoituksiin nähden toinen vaihtoehto sille olisi ollut C-kieli. C on huomattavasti suositumpi kieli sulautettujen järjestelmien ympäristöissä, mutta Python on nopeimmin kasvava kieli sulautetuissa järjestelmissä. [11.]

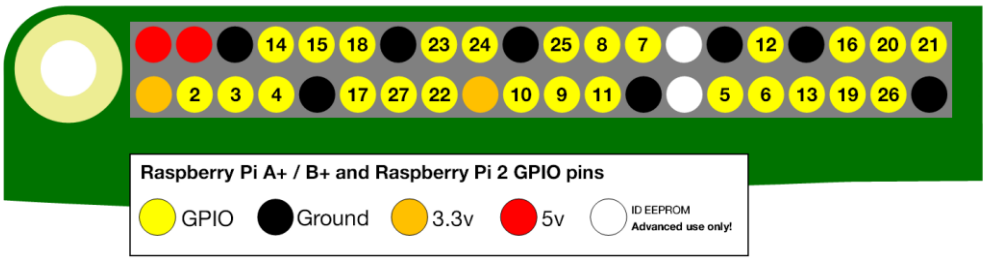
Verrattuna C:hen, Pythonia on nopeampi ja helpompi kehittää, koska jokaisen muutoksen jälkeen ei tarvitse kääntää suoritettavaa tiedostoa. Siihen on myös saatavilla paljon kirjastoja, joita hyödyntämällä ohjelmointi nopeutuu entisestään. Toisaalta Python-koodin suorittaminen on hitaampaa ja vie enemmän muistia kuin käännetty C:llä tehty ohjelma. [11.] Tässä työssä ohjelman suoritusnopeus ei kuitenkaan ole merkittävä tekijä, sillä anturien arvoja luetaan minuutin välein ja järjestelmällä ei ole mitään reaaliaikaisia vaatimuksia. Muistin määräkään ei ole erityisen rajoitettu. Tämän, ja kasvavan suosionsa takia, Python valikoitui pääohjelman kieleksi.

4 KOMPONENTTIEN KYTKENNÄT

Kuvassa 4 on Fritzing-ohjelmalla tehty graafinen esitys kytkentäkaaviosta. Kuvan oikealla laidalla oleva sadeanturi on yhdistetty sen mukana tulleeseen elektroniikkamoduuliin, joka sisältää potentiometrin, jolla säädetään digitaalilähdön herkkyyttä. Tämä moduuli on yhdistetty 3,3 V:n käyttöjännitteeseen ja sen digitaalilähtö on yhdistetty suoraan Raspberry Piin pinniin 18. Pinnien numerointi on esitetty kuvassa 5. Siitä selviää myös, mitkä pinnit ovat määritelty jännitelähdöiksi ja mitkä maadoituspinneiksi. Koekytkentälevyn alaosassa alin rivi on siis määritelty maaksi ja toiseksi alin 3,3 V:n lähtöjännitteeksi.



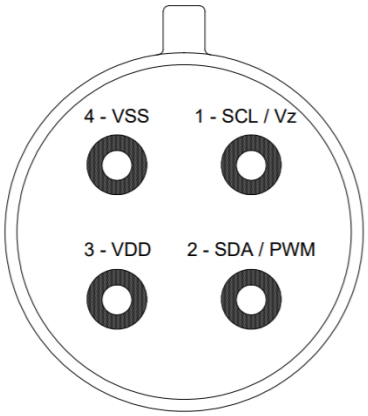
Kuva 4. Raspberry Piin ja komponenttien väliset kytkennät koekytkentälevyllä.



Kuva 5. Raspberry Piin GPIO-pinnit ja niiden numeroinnit (kuva Raspberry Pi Foundation).

Kuten aiemmin todettiin, Piihin ei voi suoraan yhdistää analogisia lähtöjä. Väliin tarvitaan ADC-muunnin, joka on tässä tapauksessa koekytkentälevyn keskellä oleva MCP3008. Sen alaosassa olevat mustat johdot ovat kytketty maahan ja punaiset johdot 3,3 V:iin. Ruskeat johdot ovat kytketty Raspberry Piin SPI-väylän lukemiseen tarkoitettuihin pinneihin. Raspberry Piin dokumentaatiossa kerrottaan, että pinnit 8–11 ovat määritelty SPI-väylän lukemiseen [12]. MCP3008:n ylemmällä rivillä on sen 8 kanavaa, joihin voi yhdistää antureita tai laitteita. Sadeanturin analogilähtö on yhdistetty ensimmäiseen kanavaan vihreällä johdolla. Muille kanaville ei ole tarvetta, joten niihin ei ole kytketty mitään.

Infrapuna-anturi tarvitsee toimiakseen I2C-protokollan kello- ja datalinjat sekä näiden lisäksi 5V syöttöjännitteen ja maan, kuten käy ilmi kuvasta 6. Raspberry Piin pinnejä 2 ja 3 voidaan käyttää I2C-väylän lukemiseen. Pinni 2 on määritelty datalinjaksi ja pinni 3 kellolinjaksi. [12.] Kuvan 4 vasemmalla laidalla oleva sininen johto on siis datalinja ja vihreä johto on kellolinja. Syöttöjännitteen ja maan välissä on 0,1 F kondensaattori mahdollisten häiriöiden poistamiseen.



Kuva 6. Infrapuna-anturin pinnit. SCL- ja SDA-pinnit ovat I2C-protokollan kello- ja datalinjat. VDD on syöttöjännite ja VSS on maa (kuva Melexis).

5 MITTALAITTEEN OHJELMOINTI

5.1 Tuloksien tallentaminen ja lukeminen

Mittausdata tallennetaan seuraavasti: oletuskäyttäjä pi:n kotikansiossa on kansio mittaus_data, joka sisältää Python-tiedoston ja mittaustuloksia tekstitiedostoina. Mittaustuloksia sisältävät tekstitiedostot saavat nimensä sen päivämäärän mukaan, jolloin ohjelmaa on alettu suorittamaan. Esimerkiksi, jos mittauksia tekevää ohjelmaa aletaan suorittaa 5.3.2019, tiedosto saa nimekseen 5.3.2019.txt. Jos päivä vaihtuu mittauksien aikana, tulokset tallennetaan silti samaan tiedostoon. Uusi tiedosto luodaan vain, kun ohjelma käynnistetään uudelleen.

Tekstitiedostoihin tallennetaan minuutin välein mittaustulos ja nykyinen aika. Aika saadaan käyttöjärjestelmän ajasta ja tallennetaan muodossa, jossa näkyy pelkkä kellonaika, sillä päivämäärän näkee tiedoston nimestä. Aika kirjoitetaan ensin, jonka jälkeen kirjoitetaan sadeanturin havaitsema määrä kosteutta. Näiden jälkeen kirjoitetaan infrapunaanturin kohteen lämpötila. Lopuksi, jos kosteutta on havaittu riittävä määrä ja kohteen lämpötila on pakkasella, kirjoitetaan loppuun 1 ja sana jää, kuvastamaan jään havaitsemista. Muulloin kirjoitetaan loppuun 0. Sanallisen kuvauksen lisäksi lisätään numeerinen arvo 0 tai 1 helpottamaan datasta tehdyn kaavion piirtämistä. Nämä kaikki kirjoitetaan samalle riville. Yksi rivi edustaa siis yhtä mittaustulosta.

Koska vain ne tulokset, joissa on havaittu jäätä sisältävät sanan ”jää”, voidaan etsiä tiedostosta kaikki rivit, jotka sisältävät kyseisen sanan. Tämä voidaan tehdä lukemalla koko tiedosto cat-komennolla ja siirtämällä sen ruudulle kirjoittama teksti grep-komennolle, joka etsii tekstistä sille määritetyn sanan: ”cat 5.3.2019.txt | grep jää”. Näin voidaan etsiä koko tiedostosta kaikki ne rivit, jotka sisältävät etsityn sanan. Nämä rivit tulostetaan ruudulle ja koska jokaisella rivillä on aikaleima, voidaan helposti nähdä, milloin havainto on tehty.

5.2 Mittauksien pääohjelma

Kuvassa 7 pääohjelma aloitetaan määrittelemällä merkistökoodaukseksi UTF-8. Ilman tätä lisäystä, Python käyttää oletuskoodauksena ASCII-merkistöä, jolloin esimerkiksi ä-kirjainta ei voi tulostaa ruudulle tai tiedostoon. Tämän jälkeen sisällytetään ohjelmaan

tarvittavat moduulit. Näistä tärkeimpiä ovat smbus, jota tarvitaan infrapuna-anturin lukemiseen ja MCP3008, jolla luetaan sadeanturin analogisia arvoja.

Seraavaksi määritetään tekstitiedoston nimi. Tämänhetkinen aika saadaan käyttämällä time-moduulin time()-funktiota, joka palauttaa Unix-järjestelmässä sen määrän sekunteja, joka on kulunut 1. tammikuuta 1970 klo 00:00:00 UTC lähtien. Tämä annetaan parametrina localtime()-funktiolle, joka palauttaa struct_time-tyyppisen olion. Struct_time-tyypillä on eri elementtejä, jotka sisältävät tietoa tämänhetkisestä ajasta, mm. vuosi, kuukausi, kuukauden päivä. [13.] Näitä käytetään seuraavalla rivillä, jossa *tiedostopvm:n* päivä (tm_mday), kuukausi (tm_mon) ja vuosi (tm_year) tallennetaan string-tyyppisenä *tiedostonimi*-muuttujaan. %d-notaatio tarkoittaa kokonaisluvun ilmaisua string muodossa. *Tiedostonimi* saa siis arvokseen kuluvan päivän, kuukauden ja vuoden, joiden perään lisätään loppuliite ”.txt”, joka ilmaisee, että kyseessä on tekstitiedosto. Kaikki tämä suoritetaan vain kerran ohjelman käynnistyessä, koska nämä rivit ovat pääsilmukan ulkopuolella.

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3  import smbus
4  import time
5  from datetime import datetime
6  from gpiozero import InputDevice
7  from gpiozero import MCP3008
8
9  tiedostopvm = time.localtime(time.time())
10 tiedostonimi = "%d.%d.%d.txt" % (tiedostopvm.tm_mday, tiedostopvm.tm_mon, tiedostopvm.tm_year)
```

Kuva 7. Python-ohjelmassa käytetyt moduulit ja tiedoston nimeäminen.

Kuvan 8 Main-funktiossa alustetaan sadeanturin analogisen arvon sisältävä muuttuja ja objekti infrapuna-anturin luokasta. Digitaaliarvoa ei ole tarvetta huomioida, sillä analoginen arvo kertoo kosteuden määrän ja tässä tapauksessa halutaan havaita pienikin määrä kosteutta. Digitaaliarvo saataisiin kuitenkin käyttämällä gpiozero-moduulin InputDevice-funktiota. Sille annetaan parametriksi se pinni, johon laite tai anturi on yhdistetty. Alustamisen jälkeen sen arvon, eli onko se aktiivinen vai ei, voidaan lukea is_active-attribuutilla, joka palauttaa joko Truen tai Falsen. [14.]

Sadeanturin analogisen arvon sisältämä muuttuja alustetaan gpiozeron MCP3008 -funktiolla, joka lukee ADC-muuntimesta määritetyn kanavan arvon. MCP3008:ssa on 8 kanavaa ja sadeanturin analogilähtö on kytketty ensimmäiseen kanavaan, joten funktiolle annetaan parametriksi 0, eli ensimmäinen. ADC-muuntimen arvo voidaan lukea value -attribuutilla, joka palauttaa siihen yhdistetyn anturin arvon skaalattuna nollasta yhteen.

Koska kyseessä on 10-bittinen ADC-muunnin, luetun arvon resoluutio on noin 0,001 ($\frac{1}{2^{10}}$).

```
60  def main():
61      sadeanalogi = MCP3008(0)
62      ir_anturi = MLX90614()
63
64      while True:
65      fo = open(tiedostonimi, "a+")
66
67      aika = (time.localtime(time.time()))
68      kellonaika = "%d:%d:%d" % (aika.tm_hour, aika.tm_min, aika.tm_sec)
69      sademaara = sadeanalogi.value
70      T_kohde = ir_anturi.get_obj_temp()
71      data_teksti = "%s, %f, %.2f" % (kellonaika, sademaara, T_kohde)
72
73      if sademaara < 0.997 and T_kohde < 0:
74          data_teksti += ", 1, jää"
75      else:
76          data_teksti += ", 0"
77      fo.write("%s\n" % (data_teksti))
78      fo.close()
79      time.sleep(60)
```

Kuva 8. Python-ohjelman main-funktio ja sen silmukka, jossa tehdään mittaukset ja tallennetaan tulokset tiedostoon.

Riviltä 64 alkavassa pääsilmukassa luetaan anturien arvot ja tallennetaan ne tekstitiedostoon. Ensin avataan tekstitiedosto, johon kirjoitetaan append tilassa, jolloin siihen kirjoitettu teksti lisätään tiedoston loppuun, eikä ylikirjoiteta olemassa olevaa tekstiä. Sitten alustetaan tiedostoon kirjoitettavien muuttujien arvot.

Aikaleima tallennetaan muuttujaan käyttämällä asctime-funktiota, joka muuntaa parametriksi annetun ajan helpommin luettavaan muotoon. Tämänhetkinen aika saadaan samalla tavalla, kuin tiedoston nimessä aiemmin. Kahdella seuraavalla rivillä luetaan sadeanturin analoginen arvo ja infrapuna-anturin kohteen lämpötila.

Tämänhetkinen aika, sateen määrä ja lämpötila tallennetaan *data_teksti*-muuttujaan. ”%s” kirjoittaa string-tyyppisen muuttujan arvon ja ”%f” float-tyyppisen muuttujan arvon, eli liukuluvun. ”%.2f” tarkoittaa, että otetaan mukaan vain kaksi ensimmäistä desimaalia. Tietojen väliin lisätään pilkku, jotta myöhemmin kaaviota tehdessä olisi helpompi erottaa tiedot eri soluihin. Tietoja tuodessa esimerkiksi Exceliin voidaan käyttää pilkkua erottimena.

Jään havaitsemisessa käytetään sadeanturin analogista arvoa ja kohteen lämpötilaa. Sadeanturi havaitsee isot vesitipat hyvin, mutta tasaisen kosteuden tai tihkusateen huonommin. Sen takia käytetään raja-arvona 0,997 eli melkein kuiva. Testien perusteella tämän alittava arvo tarkoittaa, että anturissa on jo hieman kosteutta. Tarkistuksessa ei voida katsoa, onko arvo alle yhden, sillä täysin kuivana anturi antaa joskus arvoksi 0,999.

Tämän takia 0,997 valittiin raja-arvoksi, jolloin kyseessä ei ole virheellinen arvo ja anturissa on ainakin jonkin verran kosteutta.

Jäätä on todennäköisesti muodostunut, jos sadeanturi havaitsee tarpeeksi kosteutta ja samaan aikaan kohteen lämpötila on pakkasella. Tällöin lisätään *data_tekstin* loppuun 1 ja jään havaitsemista kuvaava teksti. Lopuksi kirjoitetaan *data_tekstin* sisältämät tiedot tiedostoon ja lisätään sen perään rivivaihto ”\n”-merkinnällä. Sitten tiedosto suljetaan ja odotetaan minuutti, jonka jälkeen tehdään seuraavat mittaukset.

Kuvan 9 koodissa suoritetaan main-funktio, jos tätä moduulia suoritetaan pääohjelmana. Tämä tehdään tarkistuksella, jossa katsotaan kyseisen moduulin *__name__*-muuttujan arvoa. Tämä muuttuja saa arvokseen *__main__*, jos moduulia suoritetaan pääohjelmana. Jos moduuli tuodaan toiseen moduuliin, *__name__* saa arvokseen sen moduulin nimen. Näin tätä tiedostoa voitaisiin käyttää moduulina jossain toisessa Python tiedostossa ilman, että suoritetaan kaikki koodi tässä tiedostossa. [15.]

Tällä on myös toinen hyöty. Kun main-funktio määritellään erikseen ja kutsutaan sitä, voidaan lopettaa ohjelman suoritus huomioimalla käyttäjän tekemät keskeytykset Ctrl + C -näppäinyhdistelmällä. Ohjelma voidaan pysäyttää myös ilman tätä koodipätkää, mutta tällä tavalla voidaan tehdä viimeistelyjä ennen kuin lopetetaan ohjelma. Esimerkiksi voitaisiin sulkea tiedosto tai kutsua funktioita, ennen kuin poistutaan ohjelman suorituksesta. Tässä tapauksessa ei ole tarvetta tehdä mitään erityistä, mutta on hyvä tapa huomioida ohjelman mahdolliset pysäytykset tai muut virhetilat. Koska tässä tarkistetaan ohjelman keskeytyksiä, tämän kohdan koodi suoritettaisiin myös silloin, kun ohjelmaa suorittavalle prosessille lähetetään SIGINT-komento. Tämän komennon lähettäminen vastaisi siis näppäinyhdistelmällä tehtyä keskeytystä.

```
81 if __name__ == "__main__":
82     try:
83     main()
84     except KeyboardInterrupt:
85     pass
```

Kuva 9. Python-ohjelman main-funktion käynnistys ja tarkistetaan käyttäjän tekemät keskeytykset.

6 ILMATIETEEN LAITOKSEN AVOIN DATA

6.1 Avoimen datan hyödyntäminen

Koska tämän työn mittalaite mittaa vain yhtä mittauspistettä kerrallaan, on tärkeää tietää, millaiset olosuhteet ovat kauempana mittauspisteestä. Lämpötila ja sateen määrä saattavat vaihdella usean kilometrin päässä mittauspisteestä. On myös mahdollista, että infrapuna-anturin mittaamaan kohteeseen vaikuttaa jokin lämmönlähde ja sen antamat arvot eivät pidä paikkaansa. Laitteen mittaamille arvoille olisi siis hyvä saada jokin vertailukohde.

Yksi ratkaisu olisi rakentaa useampi mittalaite ja sijoittaa ne erilleen toisistaan. Helpompi ratkaisu on hyödyntää Ilmatieteen laitoksen julkaisemaa avoimen datan rajapintaa. Avoimen datan palvelusta saa haettua paljon erilaista tietoa, kuten havaintoasemien mittaustuloksia. Havaintoasemien tuloksia voidaan verrata mittalaitteen tuloksiin, jotta saadaan tehtyä vertailuja mittausten välillä.

Kosteuden havaitsemisessa joudutaan käyttämään hieman erilaista tapaa avoimen datan kanssa kuin mittalaitteella. Mittalaitteen sadeanturi havaitsee kaiken siihen kertyneen kosteuden sateesta tai tiivistyneestä kosteudesta. Ilmatieteen laitoksen havaintoasemilta saa sateen määrän helposti, mutta kosteuden tiivistymistä ei suoraan raportoida. Havaintoasemat mittaavat suhteellista kosteutta ja tämän perusteella lasketaan kastepistelämpötila, johon siis nykyisen lämpötilan tulee laskea, jotta kosteus tiivistyisi. Edellisten havaintojen kastepistelämpötilaa käyttämällä voidaan arvioida, onko kosteutta mahdollisesti tiivistynyt. Tosin tämäkään ei kerro, kuinka paljon kosteutta on mahdollisesti kertynyt.

Ilmatieteen laitos on julkaissut Javascript-kirjaston ja esimerkkitiedoston, joiden avulla dataa saa haettua helposti. Tätä esimerkkitiedostoa piti muokata, jotta saadaan haettua tämän työn kannalta oleellisia tietoja. Tiedot pitää myös saada tallennettua jonnekin. Koska esimerkki oli HTML-tiedosto, jossa dataa haetaan Javascriptin avulla, piti käyttää palvelinta, jonne data lähetetään ja jossa se tallennetaan tekstitiedostoon. Palvelin toteutettiin Node.js-ympäristöllä. Palvelin ohjelmoitiin myös näyttämään mittalaitteen ja avoimen datan tulokset Raspberry Piin IP-osoitteeksi määritetyllä sivulla. Aina kun on yhdistetty samaan verkkoon kuin Pi, voidaan kirjoittaa selaimeen osoitteeksi sen IP-

osoite ja määritetyn portin numero, jolloin palvelin tulostaa sivulle kaikki siihen mennessä saadut tulokset.

6.2 Avoimen datan ohjelmointi

Avoimen datan palvelusta saa haettua paljon erilaista dataa. Kaikki mahdolliset kyselyt ja mitä tietoja ne palauttavat ovat lueteltu Ilmatieteen laitoksen sivuilla. Tässä työssä käytetään hetkellisten säähavaintoarvojen kyselyä, joka palauttaa havaintoasemien sää-havaintoja [16]. Tämän työn kannalta oleellisia tietoja ovat ilman lämpötila, kastepiste ja sateen määrä.

Kuvassa 10 yhdistetään avoimen datan palvelimelle halutuilla parametreilla. Kyselyksi on valittu siis havaintoasemien palauttama data multipointcoverage-muodossa. On myös mahdollista käyttää ennusteita, mutta niiden palauttaman datan tarkkuus ei ole yhtä hyvä kuin havainnoilla. Yhdistämispyynnön parametreiksi annetaan ilman lämpötila, kastepis-telämpötila, suhteellinen kosteusprosentti, sateen määrä ja sateen intensiteetti. Kaikki mahdolliset kyselyn palauttamat parametrit saa haettua kirjoittamalla selaimeen osoit-teeksi kooditiedostosta löytyvän URL-osoitteen, kyselyn ja havaintoaseman tunnistusnu-meron (http://opendata.fmi.fi/wfs?request=getFeature&storedquery_id=fmi::observati-ons::weather::multipointcoverage&fmisid=100949). Havaintoaseman numero löytyy http://catalog.fmi.fi/-osoitteesta ja hakemalla sieltä haluttua havaintoasemaa. Parametrit löytyvät observedProperty-tagin sisältämän linkin kautta.

```
22          var STORED_QUERY_OBSERVATION = "fmi::observations::weather::multipointcoverage";
23          var STORED_QUERY_FORECAST =
            "fmi::forecast::harmonie::surface::point::multipointcoverage";
24          var SERVER_URL = "http://opendata.fmi.fi/wfs";
25          var connection = new Metolib.WfsConnection();
26          function getTurkuData() {
27              if (connection.connect(SERVER_URL, STORED_QUERY_OBSERVATION)) {
28                  connection.getData({
29                      requestParameter : "t2m,td,rh,r_1h,ri_10min",
30                      begin : new Date().setMinutes(new Date().getMinutes()-60),
31                      end : new Date(),
32                      timestep : 60 * 60 * 1000,
33                      sites : "Turku",
34                      callback : function(data, errors) {
35                          handleCallback(data, errors, "Turku");
36                          haeKaste();
37                          connection.disconnect();
38                      }
39                  });
40              }
41          }
```

Kuva 10. Yhteyden muodostamiseen ja datan hakuun käytetty funktio.

Kun yhteys on muodostettu halutuilla parametreilla, pitää vielä määritellä miltä aikaväliltä dataa haetaan. Tässä kohtaa kyselyn palauttaman datan kanssa oli ongelmia. Koodissa rivillä 32 oleva *timestep* määrittää kuinka usein dataa pyydetään alun ja lopun välillä. Esimerkissä pyydetään dataa tunnin välein, jolloin kysely palauttaa tasatunnein dataa. Jos dataa pyytää liian nopeasti havainnon jälkeen, esimerkiksi minuutin tai kaksi pyydettävän ajan jälkeen, kyselyn palauttaman data ei välttämättä ole vielä valmis eikä se palauta mitään. Tämä on suurempi ongelma, jos aikaväliä pienentää.

Ongelman voi kiertää pyytämällä dataa aiemmilta havainnoilta, jolloin data on varmasti valmis joka kerta, kun kysely tehdään. Tämän takia rivillä 30 asetetaan aloitusajaksi tämänhetkisestä ajasta tunti aikaisemmaksi. Kyselyn palauttama data on siis koodin suoritushetkellä 1–2 tuntia vanhaa, mutta näin voidaan varmistaa, että kysely palauttaa aina jotain dataa.

Kuvan 10 rivillä 35 kutsutaan *handleCallback*-funktiota, joka puolestaan kutsuu kuvan 11 funktiota. Siinä käydään läpi saatu data ja tulostetaan se sivulle. Riviltä 114 alkavassa silmukassa pitää tehdä tarkistuksia, jotta saadaan haluttu data tallennettua. Kyselyt palauttavat paljon erilaista tietoa ja ne ovat eroteltu elementteihin. Tiedot tulostetaan sivulle niin, että yksi elementti vastaa yhtä riviä.

Jokaisella elementillä on arvo (value) ja avain (key). Kumpikaan näistä ei suoraan kerro, mikä tieto on kyseessä. Eli onko kyseessä ilman lämpötila, havaintoaseman nimi vai jokin muu tieto. Varsinaisilla tuloksilla on kuitenkin *time*- ja *value* -attribuutit arvon sisällä, joka kertoo, että kyseessä on havainto. Pelkästään näiden attribuuttien etsiminen ei riitä, sillä kysely palauttaa aina useamman kuin yhden havainnon. Vaikka aloitus- ja lopetusajat olisivat samat, kysely palauttaa silti kaksi havaintoa aikaväliksi määritetyn ajan mukaan. Tämän takia pitää vielä tarkistaa, että otetaan huomioon vain ensimmäinen havainto, jos halutaan hakea vain yksi havainto kerrallaan.

Kun havainnon aika on löydetty rivin 115 tarkistuksessa, muutetaan se luettavaan muotoon. Ajat palautetaan oletuksena Unix-aikana, eli sekunteina 1970 lähtien. Tämä muutetaan muotoon, jossa on tunnit, minuutit ja sekunnit sekä tallennetaan muuttujaan. Rivillä 126 tarkistetaan, että on löydetty havainto, sen aika ja, että sillä on numeerinen arvo. Aika halutaan tallentaa vain kerran, koska se on sama kaikissa havainnoissa. Se tallennetaan taulukon ensimmäiseksi alkioksi ja sitten tallennetaan havaintojen varsinaiset arvot.

```
function recursiveBrowse(container, data, indentStr) {
    if (_.isArray(data) || _.isObject(data)) {
        indentStr += ">";
        _.each(data, function(value, key) {
            if (value.time != null && key == 0) {
                var pvm = new Date(value.time);
                var h = pvm.getHours();
                var m = pvm.getMinutes()<10 ? '0' + pvm.getMinutes():pvm.getMinutes();
                var s = pvm.getSeconds()<10 ? '0' + pvm.getSeconds():pvm.getSeconds();
                aika = h+':'+m+':'+s;
                value.time = aika;
            }
            if (value.value != null && !isNaN(value.value) && key == 0) {
                havaintoArvo = value.value;
            }
            if (aika != null && havaintoArvo != null) {
                if (aikaAsetettu == 0) {
                    tulokset[tulokset.length] = aika;
                    aikaAsetettu = 1;
                }
                tulokset[tulokset.length] = (havaintoArvo.toString());
                aika = null;
                havaintoArvo = null;
            }
            container.append("<br>" + indentStr + " [" + key + "]");
            recursiveBrowse(container, value, indentStr);
        });
```

Kuva 11. Funktio, joka käy läpi saatua dataa rekursiivisesti.

Kun haluttu data on haettu ja tallennettu taulukkoon, se pitää lähettää palvelimelle, jossa se tallennetaan Raspberry Piin tiedostojärjestelmään. Ennen tätä pitää kuitenkin hakea edellisten tuloksien kastepistelämpötila, johon verrataan tämänhetkistä lämpötilaa. Se tehdään kuvassa 12. Palvelimen osoitteeksi on määritelty Raspberry Piin IP-osoite ja portti 8080. Jos edellinen kastepiste saadaan haettua, kutsutaan *callback*-funktiota edellinen kastepiste parametrinaan. Voi kuitenkin olla, että ollaan suorittamassa koodia päivän ensimmäistä kertaa, jolloin edellistä kastepistettä ei vielä ole. Tässä tapauksessa kutsutaan *callbackia* parametrilla null.

```
function haeKaste() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            callback(this.responseText);
        }
        else if (this.readyState == 4 && this.status == 500) {
            callback(null);
        }
    };
    xhttp.open("GET", "http://192.168.1.243:8080/haekaste", true);
    xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhttp.send();
}
```

Kuva 12. Funktio, jossa tehdään AJAX-pyyntö palvelimelle.

Callback-funktio on esitetty kuvassa 13. Siinä tallennetaan edellinen kastepiste muuttujaan ja asetetaan taulukon arvot string-muuttujaan. Tässä kohtaa myös tarkistetaan, onko jäätä mahdollisesti muodostunut. Se tehdään tarkistamalla, onko lämpötila ollut pakkasella ja sateen määrä on suurempi kuin nolla tai nykyinen lämpötila on alempi kuin

edellinen kastepistelämpötila. Lopuksi tallennetaan erikseen nykyinen kastepistelämpötila seuraavaa kertaa varten ja tallennetaan kaikki saatu data samanlaisilla AJAX-pyynnöillä kuin kuvassa 12.

Muut AJAX-pyynnöt tehdään *callbackin* kautta, koska ne ovat asynkronisia, joten niiden lähetysjärjestys voi vaihdella. *Callbackia* hyödyntämällä voidaan varmistaa, että haetaan edellinen kastepiste ja tehdään tarkistukset ennen kuin tallennetaan uutta dataa.

```
94          this.callback = function(responseText) {
95              edellinenKaste = responseText;
96              console.log("edellinen kaste: " + responseText);
97              asetaTuloksetString();
98              console.log(tuloksetString);
99              tallennaKaste();
100             tallennaData();
101         }
```

Kuva 13. *callback*-funktio, jossa tallennetaan edellinen kastepiste ja tallennetaan data.

Mittauksien ja avoimen datan Shell-skripti

Avoimen datan haku suoritetaan siis HTML-tiedoston sisällä olevalla Javascriptillä, joka tarvitsee Github-projektin mukana tulleen Javascript-kirjaston toimiakseen. Mittaukset tekevä ohjelma voidaan käynnistää komentoriviltä ja samoin Node.js-palvelin, mutta HTML-tiedoston toistuvaan suorittamiseen tarvitaan Shell-skripti. Tämä on esitetty kuvassa 14, jossa rivillä 8 suoritetaan HTML-tiedosto käyttäen Chromium-selainta. Timeout-komento lopettaa käynnistetyn prosessin halutun ajan kuluttua. Tässä tapauksessa selaimen avaama sivu suljetaan 60 sekunnin kuluttua. Seuraavaksi määritetään millä ruudulla selain käynnistetään. Tämä pitää olla erikseen määritetty, jotta skripti toimii myös SSH:n kautta. Lopuksi odotetaan tunti, jonka jälkeen haetaan dataa uudestaan.

```
1   #!/bin/bash
2
3   python mittaukset.py &
4   nodejs avoin_data/metolib/examples/main.js &
5   sleep 5
6   while [ 1 ]
7   do
8       timeout 60 env DISPLAY=:0.0 chromium-browser avoin_data/metolib/examples/haedata.html &
9       sleep 3600
10  done
```

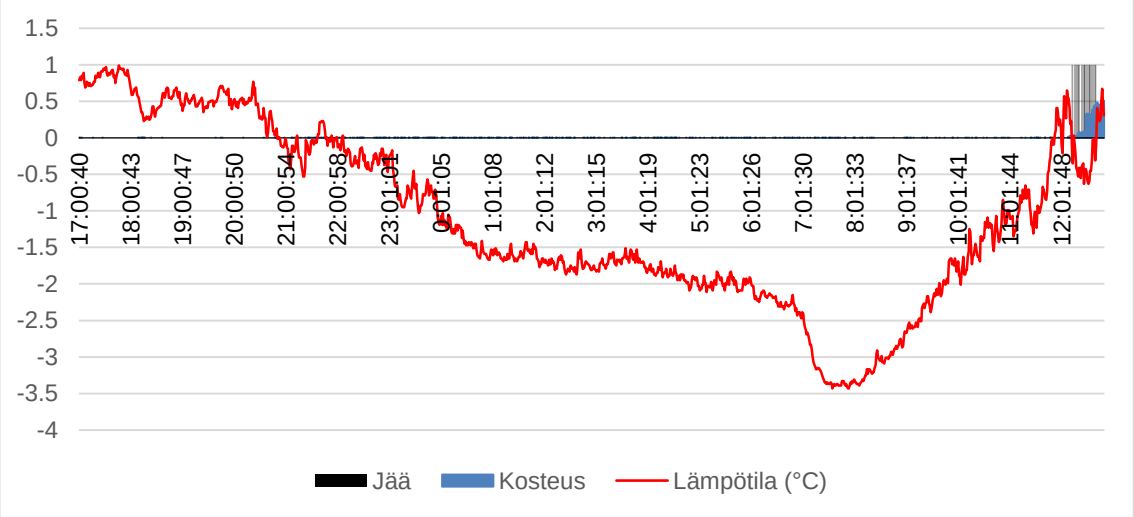
Kuva 14. Shell-skripti, joka käynnistää mittaukset ja palvelimen sekä suorittaa avoimen datan haun selaimella.

7 TULOKSET

7.1 Mustan jään havaitseminen mittalaitteella

Mittalaitteen testien perusteella voidaan todeta, että sillä pystytään havaitsemaan mustan jään muodostumista. Esimerkkinä tästä on kuvassa 15 esitetty mittaustulos, joka suoritettiin 9.3.2019. Siitä nähdään, että mittauksien lopussa sadeanturille kertyi kostetta, joka jäätyi, kun lämpötila laski pakkasen puolelle. Tämä varmistettiin tarkastelemalla sadeanturin levyä, johon oli muodostunut jäätä. Myös muut testit osoittivat, että sadeanturi pystyi havaitsemaan siihen kertyneen kosteuden, jos sitä on riittävä määrä.

Tasaisesti levinnyt tihkusade tai pieni määrä tiivistynyttä kosteutta sadeanturin levyllä antoi hyvin pieniä arvoja. Yksikin iso vesipisara saattoi tuottaa suuremman arvon anturin kosteuden määräksi. Tämä johtuu sadeanturin toimintaperiaatteesta. Sen sateelle altistuvaan levyyn on painettu johtoja, jotka eivät kosketa toisiaan. Niiden välistä resistanssia mittaamalla voidaan arvioida veden määrää, sillä levylle kertyneet vesipisarat alkavat muodostaa yhteyksiä johtojen välillä ja levyn resistanssi muuttuu. Pienet vesipisarat yhdistävät johtoja huonommin. Yksi iso pisara voi helposti yhdistää kaksi erillistä johtoa, jos se on kontaktissa molempien kanssa samaan aikaan. Tämä piti ottaa huomioon mittauksia tekevässä ohjelmassa, jossa kosteuden raja-arvoksi valittiin 0,997. Täysin kuiva levy tuottaa arvoksi 1. Tämän raja-arvon alittaessa voidaan olla melko varmoja, että levylle on kertynyt ainakin hieman kosteutta.

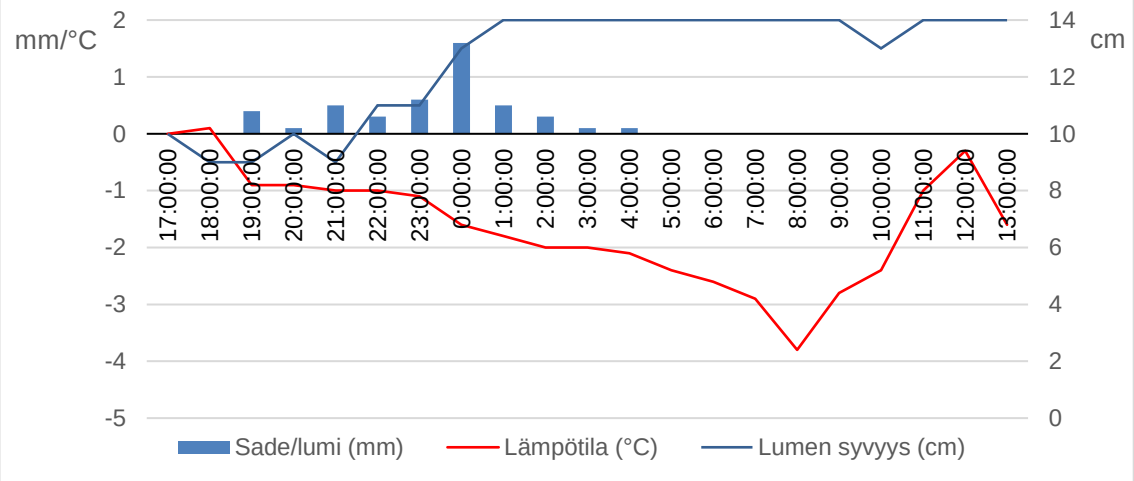


Kuva 15. Mittaustulos 9.3.2019, jossa jäätä havaittiin mittauksen lopussa. Kosteus vaihtelee välillä 0–1, jossa 0 tai melkein 0 on kuiva.

7.2 Vertailu Ilmatieteen laitoksen havaintoihin

Testeissä huomattiin, että sadeanturi ei havaitse lunta ollenkaan. Se siis tuottaa saman arvon kuivana, kuin jos sen päälle sataa lunta. Tämä ei ole kovin suuri ongelma, sillä lumesta voi syntyä mustaa jäätä vain, jos se sulaa vedeksi ja sitten jäätyy. Tästä tilanteesta syntyneen jään sadeanturi kuitenkin pystyy havaitsemaan. Kuvan 15 mittauksissa kävi juuri näin. Yön aikana oli satanut lunta ja iltapäivällä lämpötila kävi nollan yläpuolella, jolloin lumesta muodostui kosteutta. Melkein heti tämän jälkeen, lämpötila laski taas pakkaselle ja sulanut lumi jäätyi. Ilmatieteen laitoksen data Artukaisten havaintoasemalta vahvistaa tämän kuvassa 16.

Avoimen datan kyselyiden palauttama sateen määrä ei kerro, onko sade ollut vettä vai lunta. Kuvassa 16 kyseessä on ollut lumisade, sillä sateen jälkeen lumen syvyys on kasvanut. Tätä lumisadetta mittalaite ei havainnut ollenkaan, mutta se havaitsi lumen sulamisen, joka on tärkeämpää tämän työn kannalta. Kuvassa 16 lämpötila ei myöhemmin nouse nollan yläpuolelle, toisin kuten mittalaitteen mittauksissa. Eroa mittauspisteen ja havaintoaseman välillä oli tässä tapauksessa noin 5 km.



Kuva 16. Artukaisten havaintoaseman säähavainnot 9.3.2019.

Avoimen datan havainnoista ei pysty tarkasti arvioimaan, kuinka paljon maassa on vettä. Sateen määrä kertoo sataneen veden määrän millimetrien tarkkuudella, mutta on vaikeaa arvioida, kuinka nopeasti se haihtuu pois. Varsinkin kosteuden tiivistymistä on vaikea arvioida pelkästään Ilmatieteen laitoksen datan perusteella. Edellisten havaintojen kastepistelämpötilaa voidaan verrata tämänhetkiseen lämpötilaan ja kosteusprosenttia voidaan myös tarkastella, mutta nämä eivät kerro, kuinka paljon kosteutta on tiivistynyt,

jos ollenkaan. Nämä ovat enemmänkin arvioita, kun taas mittalaitteen sadeanturilla pystyy mittaamaan maahan kertyneen kosteuden ja sen määrän.

Mustan jään havaitsemiseen hetkellisten havaintojen kautta mittalaite soveltuu huomattavasti paremmin kuin Ilmatieteenlaitoksen havaintoasemien mittaukset. Mittalaitteen sadeanturin levy voidaan asettaa lähelle maata, jolloin se edustaa maan tai tien pintaa. Näin saadaan mitattua maassa oleva veden määrä. Myös lämpötilan havaitsemisessa mittalaitteen infrapuna-anturi on toimiva ratkaisu. Mustan jään havaitsemisessa halutaan tietää tien tai maan lämpötila. On mahdollista, että ilman lämpötila on hieman nollan yläpuolella, mutta maan lämpötila on joissain kohdissa pakkasella. Infrapuna-anturi soveltuu tähän hyvin. Se voidaan kohdistaa mitattavaan alueeseen, jolloin saadaan mitattua maan lämpötila ja voidaan olla varmoja, että mittalaite itsessään ei ainakaan vaikuta mittaustuloksiin.

8 POHDINTA

Tämän opinnäytetyön tavoitteena oli rakentaa laite, joka pystyy havaitsemaan mustan jään muodostumista sekä käyttää Ilmatieteen laitoksen avointa dataa ja päätellä, kumpi menetelmä olisi parempi laajemman alueen mittaamiseen. Avoimen datan palvelusta haettiin hetkellisiä säähavaintoja, joita vertailtiin mittalaitteen tekemiin mittauksiin.

Tuloksena voidaan todeta, että mittalaitteeseen valitut anturit ovat parempi ratkaisu kuin Ilmatieteen laitoksen data mittauksia tehdessä. Jos mittauksia haluttaisiin tehdä laajemmalta alueelta, pitäisi rakentaa useampi mittalaite ja asentaa ne haluttuihin mittauspisteisiin. Niistä saatu data pitäisi saada tallennettua ja kerättyä samaan paikkaan, jotta tuloksia pystyy arvioimaan. Helpoin ratkaisu tähän olisi asentaa mittalaitteet paikkoihin, joissa ne voidaan yhdistää WiFi-verkkoon. Mittausdata voitaisiin sitten tallentaa esimerkiksi Git-repositorioon, josta näkisi kaikkien laitteiden mittaustulokset. Suurimmat ongelmat tässä olisivat todennäköisesti sopivien mittauspaikkojen löytäminen ja useiden laitteiden sekä anturien kustannukset.

Myös sadeanturin puhtaana pysyminen voisi muodostua ongelmaksi pidemmällä aikavälillä. Tämän työn testeissä sadeanturi pysyi puhtaana, mutta epäpuhtauksien kertyessä anturin levylle, mittaustulokset voivat muuttua epäluotettaviksi. Varsinkin tiesuolasta voi olla haittaa sadeanturille. Jatkon kannalta pitäisi selvittää, kuinka paljon epäpuhtaudet vaikuttavat mittauksiin ja miten niitä voisi ehkäistä.

Yksi tapa hyödyntää Ilmatieteen laitoksen dataa olisi käyttää ennusteita ja pyrkiä ennakoimaan mustan jään muodostumista. Toisaalta kuten aikaisemmin todettiin, on vaikea arvioida maassa olevan kosteuden määrää pelkästään kastepisteellä ja sateen määrällä. Tästä syystä olisi hyvä käyttää myös mittalaitetta vahvistamaan jään muodostuminen. Ideaali ratkaisu voisi siis olla, että käyttäisi ennusteita arvioimaan mustan jään muodostumista ja tehdä myöhemmin mittauksia varmistamaan arviot.

LÄHTEET

[1] K. Rodman, A. Williams. *Black ice: How to spot this winter driving danger.* Saatavissa: https://www.accuweather.com/en/weather-news/black-ice-driving-dangers/22052530. Viitattu 8.1.2019.

[2] T. Liu, Q. Pan, J. Sanchez, S. Sun, N. Wang, H. Yu. Prototype Decision Support System for Black Ice Detection and Road Closure Control. *IEEE Intelligent Transportation Systems Magazine*, 2017. Vol. 9:2. S. 91-102. ISSN 1939-1390.

[3] Ilmatieteen laitos. Saatavissa: https://ilmatieteenlaitos.fi/lampotila-ja-kosteus. Viitattu 12.1.2019.

[4] Ilmatieteen laitos. Saatavissa: https://ilmatieteenlaitos.fi/ilman-kosteus. Viitattu 12.1.2019.

[5] Tabatabai H, Aljuboori M. A Novel Concrete-Based Sensor for Detection of Ice and Water on Roads and Bridges. *Sensors (Basel, Switzerland)*, 2017. Vol. 17:12. S. 2912. ISSN 1424-8220.

[6] *VTT:n liukkauden tunnistin varoittaa etukäteen kuljettajaa.* VTT 2013; Saatavissa: https://www.vtt.fi/medialle/uutiset/vtt-n-liukkauden-tunnistin-varoittaa-etuk%C3%A4teen-kuljettajaa. Viitattu 22.1.2019.

[7] *FAQs – Raspberry Pi Documentation*. Raspberry Pi; Saatavissa: https://www.raspberrypi.org/documentation/faqs/#introduction. Viitattu 27.5.2019.

[8] *MLX90614 tekniset tiedot.* Melexis 2017; Saatavissa: https://www.melexis.com/-/media/files/documents/datasheets/mlx90614-datasheet-melexis.pdf. Viitattu 19.2.2019.

[9] *MCP3008 tekniset tiedot.* Microchip 2007; Saatavissa: http://ww1.microchip.com/downloads/en/devicedoc/21295c.pdf. Viitattu 19.2.2019.

[10] *SSH Protocol.* SSH Communications Security; Saatavissa: https://www.ssh.com/ssh/protocol/. Viitattu 25.2.2019.

[11] Radcliffe T. *Python Vs. C/C++ In Embedded Systems.* ActiveState; Saatavissa: https://www.activestate.com/blog/python-vs-cc-embedded-systems/. Viitattu 26.2.2019.

[12] *GPIO - Raspberry Pi Documentation.* Raspberry Pi; Saatavissa: https://www.raspberrypi.org/documentation/usage/gpio/. Viitattu 22.3.2019.

[13] *Time access and conversions.* Python Software Foundation 2019; Saatavissa: https://docs.python.org/3/library/time.html. Viitattu 11.3.2019.

[14] *Input Devices — Gpiozero 1.5.0 Documentation.* Saatavissa: https://gpiozero.readthedocs.io/en/stable/api_input.html#gpiozero.InputDevice. Viitattu 12.3.2019.

[15] *__main__ — Top-level script environment.* Python Software Foundation 2019; Saatavissa: https://docs.python.org/3/library/__main__.html. Viitattu 18.3.2019.

[16] *Tallennetut kyselyt.* Ilmatieteen laitos; Saatavissa: https://ilmatieteenlaitos.fi/tallennetut-kyselyt. Viitattu 10.4.2019.