

Karelia-ammattikorkeakoulu
Tradenomi (AMK)

Reittikartta-ohjelmiston kehittäminen

Avoimiin lähdekoodeihin ja aineistoihin perustuvan Reittikartta-ohjelmiston toteuttaminen

Jari Keskisalo

Opinnäytetyö, toukokuu 2024



OPINNÄYTETYÖ
Toukokuu 2024
Tietojenkäsittelyn koulutus

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Tekijä(t)
Jari Keskisalo

Nimeke
Reittikartta-ohjelmiston kehittäminen

Tiivistelmä

Opinnäytetyön tavoitteena oli kehittää taitojani full stack -kehittäjänä. Taitojen kehittäminen tapahtui jatkokehittämällä jo työharjoittelussa tekemääni projektia omatoimisesti. Opinnäytetyössä toteutettavan karttaohjelman lähtökohtana toimi raportin kaltainen verkkosivu, joka muodostettiin ajoneuvon ajotietokoneen ja liikeantureiden välittämästä datasta.

Opinnäytetyössä käsitellään projektissa käytettyjen antureiden ja mikrokontrollereiden käyttämistä niille tehtyjen dokumenttien ja esimerkkien avulla. Lisäksi opinnäytetyössä tutustutaan karttaohjelmien tekemisessä käytettyihin avoimiin ohjelmointikirjastoihin ja aineistoihin niitä hyödyntämällä. Opinnäytetyössä tarkastellaan myös webkehittämisessä eteen tulevia ongelmia ja niiden ratkaisuja.

Oikeiden tietojen löytäminen ja niiden oikealla tavalla hyödyntäminen osoittautuivat tärkeiksi taidoiksi avoimiin lähdekoodeihin perustuvaa karttaohjelmaa kehittäessä. Reittikartta-ohjelmiston avulla oli mahdollista seurata liikkumista kartalla reaaliajassa ja tallenteelta. Liikeantureiden keräämä data välitettiin myös lähes reaaliajassa eri laitteella toimivalle Reittikartta-ohjelmalle. Ohjelman avulla antureiden keräämästä datasta tehtyjä laskelmia oli mahdollista siirtää myös toisiin ohjelmiin.

Kieli
suomi

Sivuja 63

Asiasanat
Chart.JS, karttaohjelma, Leaflet, OpenStreetMaps, verkkosovellus



THESIS
May 2024
Degree Programme in Business Information Technology

Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600

Author (s)
Jari Keskisalo

Title
Developing a Route-map Program

Abstract

The goal of this thesis was to improve my own skills as a full stack developer. The development of skills took place by further developing the project I had already started during the internship independently. The starting point for the map program development was a report-like web page which was formed from data transmitted by the vehicle's on-board computer and motion sensors.

The thesis discusses the use of sensors and microcontrollers used in the project with the help of documents and examples made for them. In addition, the thesis explores the open programming libraries and datasets used in creating map programs by utilising them. The thesis also examines problems that arise during web development and their solutions.

Finding truthful data and utilising it in the right way proved to be important skills when developing an open-source based map software. The Route-map program made it possible to monitor movement on a map in real time and from records. The data collected by the motion sensors was also transmitted almost in real time to the Route-map program operating on a different device. Calculations made from the data collected by the sensors could also be exported to other programs.

Language
Finnish

Pages 63

Keywords
Chart.JS, Leaflet, Map application, OpenStreetMap, Web application

Sisältö

1	Johdanto	5
2	Tietoperusta	6
2.1	Frontend	6
2.1.1	HTML ja CSS	6
2.1.2	Same-origin policy ja paikalliset tiedostot	7
2.1.3	JavaScript	8
2.1.4	jQuery ja jQuery UI	9
2.1.5	Leaflet	10
2.1.6	Kartan pyörittäminen Leafletissä	11
2.1.7	Leaflet Routing Machine	12
2.1.8	Chart.JS	13
2.2	Backend	15
2.2.1	Arduino IDE	15
2.2.2	Python	16
2.2.3	CircuitPython	17
2.2.4	Flask	18
2.3	Avoimet ohjelmat ja aineistot	19
2.3.1	Project OSRM	19
2.3.2	OpenStreetMap	20
2.3.3	Maanmittauslaitoksen avoimet aineistot	21
2.3.4	Ilmatieteen laitoksen avoin data	23
2.4	Laitteet	24
2.4.1	Adafruit ESP32 Feather V2/Esspressif ESP32	24
2.4.2	Raspberry Pi 4 Model B	26
2.4.3	Adafruit Ultimate GPS/CDtop technology PA1616D	27
2.4.4	Adafruit ICM20948 IMU/Invensense ICM20948	29
2.5	Datansiirto	32
2.5.1	Muistikortti	32
2.5.2	USB-Sarja-yhteys	34
2.5.3	Bluetooth Low Energy	34
3	Laitekokonaisuuksien kokoaminen	36
3.1	Raspberry Pi -laitekokonaisuus	36
3.2	ESP32-laitekokonaisuus	37
4	Toteuttaminen	39
4.1	Ohjelmiston toimintaympäristö	39
4.2	Käyttöliittymän ja skriptipuolen toteuttaminen	40
4.3	Rajapinnan ja datankeräyksen toteuttaminen	41
5	Toteutetut toiminnot	46
5.1	Käyttöliittymä	46
5.2	Tallenne-toimintatila	49
5.3	Live- ja Serial-toimintatilat	49
5.4	Karttamerkit	50
6	Pohdinta	52
6.1	Menetelmälliset valinnat	52
6.2	Luotettavuus ja eettisyys	53
6.3	Hyödyntäminen ja jatkokehittäminen	54
	Lähteet	56

1 Johdanto

Työharjoittelussa kehitin ammatillisessa koulutuksessa käytettävän sovelluksen osaa, jonka avulla kerättiin kuorma-auton kuljettajan ajotavasta tietoa mahdollisten epätaloudellisten ajotapojen löytämistä varten. Ajosuorituksen aikana kuorma-auton ajotietokoneesta kerätyn tiedon lisäksi kerättiin myös paikannustietoja. Näiden paikannustietojen avulla ajotietokoneen data pystyttiin liittämään tiettyyn hetkeen ajetulla reitillä mikä mahdollisti ajosuorituksen tarkemman analyysin. Sovelluksen osaa tehdessä kehittyi ajatus siitä, missä muualla samankaltaista liikkumisen aikana kerättyä dataa voisi hyödyntää.

Ensimmäisenä ajatuksenani tässä projektissa oli hyödyntää liikkumisen aikana kerättyä dataa tekemieni pyöräily- ja kävelylenkkien suoritusten analysointiin. Suoritusten esittämistavan idean sain televisioiduista kilpasuunnistuskisoista, joissa sen hetkinen kilpailijoiden sijainti merkittiin reaaliajassa päivittyvään karttaan hännällä, jonka avulla kuvattiin kuljettua reittiä kisan edetessä. Samantapaisen reaaliaikaisen sijainnin kartalle piirtäminen mahdollisti osittain projektin karttaohjelman käyttämisen myös navigoinnin apuvälineenä. Avoimien reitinselvitys-kirjastojen avulla opinnäytetyön karttaohjelmaa oli mahdollista jatkokehittää edelleen, jotta sitä olisi voinut käyttää erilaisissa ajoneuvossa ja jalkaisin liikkeessä navigoinnin apuvälineenä.

Opinnäytetyön tavoitteena oli verkkoselaimessa toimivan Reittikartta-ohjelmiston toteuttaminen, jonka avulla olisi mahdollista seurata liikkumista kartalla reaaliajassa ja myös tallenteelta. Karttaohjelman avulla piti olla mahdollista käsitellä karttamerkkejä. Liikettä mittaavista antureista saatu data esitettäisiin käytölliittymässä olevan kaavion avulla. Reittikartta-ohjelmiston piti toimia sitä käyttötarkoitusta varten hankitulla kosketusnäytöllisellä laitteella. Datan keräämisen piti olla myös mahdollista myös pienemmällä laitteella. Pienemmän laitteen voisi liittää tietokoneeseen, jolloin tietokoneella olisi mahdollista seurata reaaliaikaista liikkeestä kerättyä dataa Reittikartta-ohjelmiston avulla.

Päätavoitteena tässä projektissa oli oman osaamisen lisääminen full stack -kehittäjänä. Osaamisen lisäämisen lisäksi pyrkimyksenä oli luoda perusversio omaan käyttöön tulevasta lähes kaiken kattavasta karttaohjelmasta. Hyvän tietopohja hankkiminen oli myös osana karttaohjelman toteuttamisessa, koska se myös auttaisi projektin jatkokehityksessä. Opinnäytetyötä tehdessä tutustuin myös useampiin avoimiin lähdekoodikirjastoihin ja aineistoihin ja niiden käyttämiseen.

2 Tietoperusta

2.1 Frontend

2.1.1 HTML ja CSS

HTML on merkintäkieli, jonka avulla voi tehdä staattista sisältöä sisältäviä verkkosivuja (MDN contributors 2023a). HTML-merkintäkielen avulla verkkosivun rakenne määritellään elementteihin, jotka luodaan kulmasulkeiden sisään sijoituilla HTML-tageilla (MDN contributors 2023b). Tagi määrittää elementin tyylin ja attribuuttien arvot (MDN contributors 2023b). Kaikissa tageissa on myös globaaleja attribuutteja, joiden avulla voi esimerkiksi asettaa elementin tyylin ja sen tunnisteet (MDN contributors 2023c). Tagien ja tunnisteiden avulla elementin tyylin voi määritellä myös luokissa CSS-tyylikielen avulla (MDN contributors 2023d). HTML-tyyleillä vaikutetaan HTML-elementin ulkoasuun, eli miten kyseinen elementti piirretään verkkosivulle (MDN contributors 2023e). HTML-elementtien sisältöä on mahdollista muokata dynaamisesti verkkoselaimessa toimivan skriptikielen avulla (MDN contributors 2023a). Skriptikielen avulla muokataan verkkoselaimen muistissa olevaa dataa DOM-rajapintaa käyttämällä (MDN contributors 2023f).

HTML- ja DOM-standardia nykyisin kehittää WHATWG-ryhmä, jonka kehittämät standardit ovat eläviä standardeja (W3C & WHATWG 2021). Standardeissa kuvataan, kuinka toimintojen pitäisi toimia (MDN contributors 2023g). Olemassa on myös tietoturvakäytäntösuosituksia (policy), jotka kuvaavat kuinka tietoturva haavoittuvaisuuksia voi vähentää verkkoselaimissa (W3C 2016).

2.1.2 Same-origin policy ja paikalliset tiedostot

Same-origin policyn tavoitteena on verkkosivun eristäminen, jotta ulkopuoliset verkkosivut eivät pääsisi käsiksi sen sisältöön. Tämä tapahtuu siten, että verkkoselaimelle tulevassa viestissä pitää olla origin-tunniste. Yhteyspyyntöä ei hyväksytä, jos origin-tunniste ei ole sama. Origin-tunnisteen määrittelyä varten on olemassa CORS-protokolla, jonka avulla myös verkkotunnuksen ulkopuolisia yhteyksiä voidaan käyttää. Origin-tunnistetta ei kuitenkaan käytetä kaikissa viesteissä. Esimerkiksi HTTP-metodit GET ja POST toimivat myös ilman CORS-protokollan käyttöä. (WHATWG 2023a.) Verkkoselaimissa JavaScript-kielen avulla tehtävä origin-tunnisteen määrittely on estetty, joten paikallisen tiedoston käsittely JavaScriptin file-rajapinnan kautta aiheuttaa virheen verkkoselaimissa (MDN contributors 2023h).

Paikallisten tiedostojen käsitteleminen JavaScriptillä oli siis tehty hankalaksi verkkoselaimissa verkkosivujen verkkotunnusten tietoturvan vuoksi. Näiden rajoitteiden takia karttaohjelmassa karttaan lisättävien merkkien tallentaminen JavaScriptillä täytyi toteuttaa muuten kuin suoraan käsittelemällä paikallista tiedostoa. Tiedostoon tallentaminen ja tiedostosta lukeminen olisi Baillyn (2019) mukaan mahdollista paikallisen HTTP-palvelimen avulla. Jos HTTP-palvelinta ei ole mahdollista käyttää, silloin JavaScriptissä olevan datan tallentaminen tiedostoon olisi mahdollista HTTP-sivulla olevan tiedostolinkin kautta, jossa käytetään download-attribuuttia (Firdaus 2023). Ladattava tiedoston datan kuitenkin piti olla HTML-sivulla olevan elementin määrittelyssä. Download-attribuutin käyttäminen vaatii uuden HTML-elementin luomista JavaScript-kielen avulla, johon suorituksen aikana käsitelty data oli mahdollista lisätä.

Kartta-ohjelmassa käytettävän karttamerkkien tiedot sisältävän tiedoston lataaminen JavaScript-koodissa käytettäväksi oli mahdollista, kun sen lisäsi HTML-sivun lähdekoodiin omassa JavaScript-tiedostossa olevana muuttujana. Toinen vaihtoehto olisi ollut käyttäjän aloittama tiedoston avaustoiminto. Tätä varten on olemassa HTML-elementti nimeltään `input`, jota voidaan tarkkailla JavaScript-kielen avulla (MDN contributors 2023i). Opinnäytetyössä `input`-elementin avulla ladattiin verkkosivun muistiin laitteilla tallennettu data.

2.1.3 JavaScript

JavaScript on ohjelmointikieli, jonka avulla staattisen verkkosivun sisältöä ja rakennetta pystytään käsittelemään suorituksen aikana DOM-rajapintojen avulla. JavaScript-koodi sijaitsee verkkosivun palvelimella erillisissä tiedostoissa tai sisällettynä palvelimella olevan verkkosivun koodiin. Verkkosivua ladatessa siihen liitätettyjen JavaScript-koodien suoritus aloitetaan verkkoselaimessa olevan JavaScript-tulkin avulla, jos sen toimintaa ei ole estetty verkkoselaimen asetuksissa. JavaScriptiä voidaan käyttää myös palvelimen puolella, esimerkiksi Node.js -suoritusympäristössä toimivan JavaScript-tulkin avulla. (MDN contributors 2023j.)

JavaScript-ohjelmakoodin suoritus tapahtuu järjestyksessä. JavaScriptin suorituserjestyksestä voidaan muokata epäsynkronisten `async`-funktioiden avulla, jotka jäävät odottamaan vapaata suoritusaikaa. Epäsynkronisen funktion palautusarvoa kuvaa lupausobjekti, jonka tehtävänä on palauttaa kyseisen funktion palauttama arvo, jos funktio suoriutuu onnistuneesti. (MDN contributors 2023k.)

Huomioitavaa on myös JavaScriptissä muuttujien määrittelyssä käytettävien `var`- ja `let`-sanojen erot. `let`-sana rajoittaa muuttujan arvon hakasulkujen tasoon eikä sitä voi uudelleen määrittellä samassa tasossa. `Var`-sanalla määriteltynä muuttujan arvo on käsiteltävissä kaikkialla hakasulkujen sisässä ja se on uudelleen määriteltävissä. `Var`-sanon käytössä pitää olla huolellinen, koska sen avulla määriteltäviä muuttujia on mahdollista kutsua ennen sen arvon

asettamista, kun taas let-sanalla määritelty muuttujan kutsuminen ennen sen määrittelyä palauttaa virheen. (W3School 2023.)

JavaScriptiä käyttää paljon eri tasoisia ohjelmoijia, joten ohjeita sen käyttöön on saatavilla paljon, kuten kirjoja, videoita ja keskustelupalsta viesteinä (Atrawalkar 2022). Kuitenkin JavaScriptin pääasiallinen toimintaympäristö, eli HTML-standardi kehittyy koko ajan (WHATWG 2023b). Saatavilla olevat ohjeet eivät tästä syystä aina välttämättä ole ajan tasalla. Tästä syystä suosin kehityksen aikana eteen tulevien ongelmien ratkaisussa mahdollisimman lähellä alkuperäisiä ohjeita olevia lähteitä, kuten Mozillan ylläpitämää MDM-verkkosivustoa. Tosin MDM-verkkosivustolla olevat ohjeet olivat välillä kovin epäselviä, joten muutkin lähteet tulivat tarpeellisiksi JavaScript-kieleen liittyviä ohjelmointiongelmia ratkaistaessa.

Opinnäytetyössä JavaScriptin avulla jäsenneltiin ja analysoitiin laitteisiin kiinnitetyiltä antureilta saatua dataa, joka käsittelyn jälkeen esitettiin interaktiivisesti verkkosivulla lähes reaaliajassa avoimien JavaScript-kirjastojen avulla. Hankalinta kehityksen alussa oli verkkoselaimen epämääräiset virheilmoitukset, jotka eivät auttaneet tarkentamaan JavaScript-kielisessä koodissa olevien virheiden perimmäisiä syitä. Koko kehityksen ajan hyödynsinkin JavaScriptistä verkkoselaimen konsoliin lähetettäviä viestejä. Näiden konsoliviestien avulla JavaScriptin suorituksen aikaisten tapahtumien selvittäminen oli mahdollista.

2.1.4 jQuery ja jQuery UI

jQuery on avoimen lähdekoodin Javascript-kirjasto. JQuery käyttää MIT-lisenssiä. (jQuery Foundation 2023.) JQueryn avulla on mahdollista yksinkertaistaa verkkosivujen JavaScript-toteutusta, oli sitten kyseessä verkkosivun DOM-rajapinnan sisällön manipuloinnista tai web-palvelimelle lähtevistä Ajax-metodikutsuista (jQuery 2023a).

jQueryn Ajax-metodikutsujen käsittelyä varten on olemassa lyhenteitä, joiden avulla HTTP-metodien GET ja POST suorittaminen on vielä yksinkertaisempaa (jQuery 2023b). Opinnäytetyössä hyödynsin jQueryn ajax-metodikutsun määrittelyä helpottamaan tehtyä getJSON -metodia. GetJSON-metodi välitti GET-pyyynnön rajapinnassa toimivalle web-palvelimelle, otti palvelimen palauttaman JSON-muotoisen datan vastaan ja muutti sen merkkijonoksi.

jQuery UI on jQueryyn perustuva JavaScript-kirjasto. jQuery UI lisää jQueryyn funktioita, joiden avulla käyttöliittymä elementtien lisääminen verkkosivulle on mahdollista. (jQuery User Interface 2023a.) jQuery UI myös muuttaa joidenkin jQuery-komentojen toimintaa (jQuery User Interface 2023b). jQuery UI vaikutti käyttävän MIT-lisenssiä, ainakin sen GitHub-sivustolla olevan lisenssimäärittelyn sisältö vastasi sanasta sanaan MIT-lisenssiksi kutsutun käyttöehtolauselman sisältöä (jQuery UI 2022; Opensource.org 2023).

Opinnäytetyössä jQuery UI:n avulla tuotiin ponnahdusikkuna näkyviin verkkosivulla. Ponnahdusikkunan sisältö koostui verkkosivun lähdekoodissa olevasta piilotetusta lomakkeesta. Ponnahdusikkunan näyttämisen lisäksi jQuery UI:n avulla oli mahdollista lisätä painikkeita dynaamisesti erilaisten käyttötarpeiden mukaan. Eli samaa lomaketta oli mahdollista käyttää karttamerkkien lisäämiseen, muokkaamiseen ja poistamiseen. Vain ponnahdusikkunassa olevat painikkeet niihin liitettyine toimintoineen muuttuivat käyttötarpeen mukaan.

2.1.5 Leaflet

Kartanpiirtäjäksi verkkosivulle valitsin avoimen lähdekoodin Leaflet JavaScript-kirjaston, johon tutustuin jo työharjoittelussa tekemässäni projektissa. Leafletin käyttäminen työharjoittelun aikana oli varsin selkeää, kun sitä käytettiin kartan ja yhden reitin piirtämiseen. Opinnäytetyössä kartan ja kuljetun reitin piirtämisen lisäksi hyödynnettiin Leafletin karttamerkkejä. Leaflet käyttää avointa BSD-2-Clause-lisenssiä (Leaflet 2023).

Geoapifyn tekemässä vertailussa Leaflettiä vertailtiin toiseen kartanpiirto-kirjastoon nimeltään Open Layers. Vertailussa tuli esiin Leafletin parhaat puolet, joita olivat helppokäyttöisyys, käyttäjämäärät ja sille tehdyt liitännäiset. Vertailussa mukana ollut Open Layers olisi ilmeisesti ollut jonkin verran hankalampi käyttöisempi kuin Leaflet. (Geoapify 2019.) Täysin tuntemattoman ja monimutkikkaaman rajapinnan opiskelu olisi tosin vienyt liikaa aikaa tässä opinnäytetyössä, mutta mahdollisesti projektin jatkokehityksessä Open Layersiin tutustuminen tulee ajankohtaiseksi.

Opinnäytetyötä tehdessä Leafletin JavaScript-kirjaston käyttämisen kanssa oli ongelmia kartan piirtämisen kuluva ajan kanssa, kun kartassa näkyvään kuljetuun reittiin lisättiin useampia polygon-viivoja kuvaamaan nopeudessa tapahtuvia muutoksia. Toisien viivojen lisäyksen jälkeen kartan päivittämiseen kuluva aika alkoi kasvamaan sitä enemmän mitä pidempi piirretty reitti oli. Epäilykseni hidastumisen syystä kohdentui tapaan, jolla kasvatin piirrettyjen yhtenäisten viivojen kokoa yksi GPS-sijainti kerrallaan. Samanlaista normaalista poikkeavaa hidastumista ei tapahtunut, kun kaikki viivat piirrettiin uudestaan kokonaisina aina kartan päivityshetkellä. Piirtämistavan muutoksen jälkeen reitin päivitykseen kuluva aika ei enää kasvanut suhteettomasti piirrettävän reitin pituuden suhteen, kun aikaisemmin se kasvoi käytössä ollutta datan keräysväliä suuremmaksi, josta seurasi datan reaaliaikaisen toiston hidastuminen.

2.1.6 Kartan pyörittäminen Leafletissä

Leafletistä puuttui metodi, jonka avulla kartan pyörittäminen olisi ollut mahdollista etenemissuunnan mukaan. Leafletin (2016) GitHub-sivuilta kuitenkin löytyi rotate-branch, jota ei ollut päivitetty moneen vuoteen. Ilmeisesti kartan pyörittämistä oli jossain vaiheessa kehitetty, mutta branchin kehitystä ei jostain tuntemattomasta syystä jatkettu julkaisukelpoiseen versioon asti. Leafletin main ja rotate -branchista tehty yhdistelmä löytyi kuitenkin Ronikarin ylläpitämältä GitHub-sivustolta, joka käytti BSD-2-Clause-lisenssiä (Ronikar 2023a; Ronikar

2023b). Pelkästä rotate-branchista löytyi myös liitännäisversio Raruton ylläpitä-miltä GitHub-sivuilla, joka käytti GPL-3.0 - lisenssiä (Raruto 2023a).

Näiden epävirallisten Leaflet-kirjastojen yhdistelmien käyttämisen ongelmana oli ohjeiden vähyys. Kuitenkin Raruton GitHub -sivuilla olevasta esimerkki verkkosivun lähdekoodista oli mahdollista selvittää, kuinka karttaa pyöritettiin yhden lisätyn parametrin ja metodin avulla (Raruto 2023b). Ongelmana voi myös olla kirjastojen päivittyminen ja yhteensopivuus uudempien Leaflet versioiden kanssa.

Kartan pyörittämistä testatessa Ronikarin ylläpitämän yhdistelmäkirjaston avulla ilmeni vika, jossa Leafletin karttanäkymä siirtyi karttamerkkiä valittaessa pois sen hetkisestä sijainnista. Aivan kuin valinnan sijainti kartalla ei olisi vastannut hiiren sijaintia hiirenpainiketta painaessa. Lisäksi kartassa näkyvät karttamerkkien tekstimuotoiset kuvaukset eivät pyörineet kartan ja siinä olevien merkkien mukaisesti. Olisiko karttanäkymän siirtymä vika karttamerkkiä valittaessa kuitenkin peräisin verkkoselaimien viasta/ominaisuudesta, ainakin saman tapaisiin vikoihin liittyviä keskusteluja löytyi Leafletin (2012) GitHub-sivuston viat-osioista. Keskusteluissa oli joitakin ehdotuksia myös kyseisen vian korjaamiseksi. Ilmeisesti kohdistuksen pitäminen kartan HTML-elementissä auttaisi ainakin osittain kartan hyppimiseen siinä olevia karttamerkkejä valittaessa (Leaflet 2012).

2.1.7 Leaflet Routing Machine

Leafletin alkuperäinen tekijä Liedman on tehnyt myös JavaScriptillä toteutetun käyttöliittymän nimeltään Leaflet Routing Machine, joka käyttää avointa ISC-lisenssiä. Käyttöliittymän avulla voidaan hakea reittiä kahden tai useamman osoitteen välillä ja esittää saadut ajo-ohjeet samassa käyttöliittymässä. Liedmanin mukaan Leaflet Routing Machinen avulla pystyy käyttämään useita eri reitin-hakupalveluita, joista Project OSRM on oletusarvona. (Liedman 2015a.)

Leafletin ja Leaflet Routing Machinen tekijä tarjoaa myös liitännäistä nimeltä Leaflet Control Geocoder, joka käyttää avointa BSD-2-Clause-lisenssiä. Sen

avulla on mahdollista käyttää kolmansien osapuolien tarjoamia osoitteenhakupalveluja Leaflet Routing Machinessa. (Liedman 2023.) Osoitteenhakupalvelun avulla koordinaatit oli mahdollista muuttaa osoitteiksi ja kaduiksi.

Leaflet Routing Machinen kotisivuilla olevien esimerkkien avulla sen käyttöönotto oli varsin yksinkertaista (Liedman 2015b). Lisäksi kotisivuilta löytyi myös rajapinnan käyttöohjeet tarkempaan tarkasteluun, missä metodit ja niissä käytettyjen parametrien tarkoitus oli selvitettävissä (Liedman 2015c). Karttaohjelman reitinselvitystä varten Leaflet Routing Machinen JavaScript-kirjasto tarvitsi pientä muokkausta, jotta sen avulla oli mahdollista käyttää paikallisesti toimivaa OSRM-reitinhakupalvelinta. Liedmanin verkkosivuilta löytyikin hyvät ohjeet, kuinka tämä muutos tehdään (Liedman 2015d).

Leaflet Routing Machinen käyttöliittymä ja Leaflet Control Geocoderin toiminta tarvitsi pieniä muutoksia, jotta se kävi paremmin tässä opinnäytetyössä käytettäväksi. Karttaohjelmassa tarkoituksena oli reitin selvittäminen aloittaminen aina sen hetkisestä sijainnista. Lähtöpisteen valinta ja sen merkitsemiseen käytetty karttamerkki olisi pitänyt tämän takia saada piilotetuksi, jotta lähtösijaintia ei olisi ollut mahdollista muuttaa. Ilmeisesti pelkästään lähtöpisteen valintaa ei voinut kuitenkaan helposti piilottaa, koska jostain syystä Leaflet Routing Machine loi sijainnin syöttämiseen käytetyt input-elementit käännetyssä järjestyksessä. Kyseistä input-elementtiä ei siis voinut merkitä luotettavasti, koska input-elementtien määrää oli mahdollista muuttaa ohjelman suorituksen aikana välipisteitä lisäämällä. Ainoana vaihtoehtona olikin kaikkien input-elementtien piilottaminen, kun mitattu nopeus ylitti karttaohjelmassa asetetun raja-arvon. Sitä varten elementille mihin input-elementit luotiin piti antaa oma tunniste. Tämän tunnisteiden avulla kaikkien elementtiin luotujen lapsi elementtien piilottaminen onnistui elementin tyyliä muuttamalla.

2.1.8 Chart.JS

Zingchartin (2023) ylläpitämän listan mukaan JavaScriptillä toteutettuja avoimen lähdekoodin kaavionpiirto kirjastoja oli kymmeniä. Niistä kolmea suosituinta

Chart.JS, Plotly ja ECharts vertailtiin Derflingerin (2021) tekemässä vertailussa. Derflingerin (2021) vertailussa mainittiin myös, että kyseiset kaavionpiirto kirjastot käyttivät kahta erilaista HTML-piirtotapaa.

HTML-piirtotapa vaikuttikin olevan ainoa suurempi ero erilaisten kaavionpiirto-ohjelmien välillä (Puszynski 2019). HTML-Canvasilla piirretään pikseleillä, kun taas HTML-SVG:llä piirtäminen tehdään vektoreilla (MDN contributors 2023I). Suorituskyvyn suhteen HTML-SVG sopii paremmin ison alan kattaville yksinkertaisille kuvioille, kun taas HTML-Canvas sopisi paremmin pienemmille ja monimutkaisimmille kuvioille (Padamkar 2023).

Opinnäytetyöhön valitsin vertailussa olleen Chart.JS:n, joka käyttää HTML-Canvasista kaavion piirtämiseen verkkosivulle (Chart.JS 2023a). Chart.JS:n kotisivuilla olevien esimerkkien perusteella kaavion piirtäminen JSON-muuttujien avulla olisi varsin yksinkertaista (Chart.JS 2023b). Esimerkkejä yksityiskohtaisempaa lisätietoa oli mahdollista etsiä Chart.JS:n verkkosivuilla olevista rajapinnan metodien kuvauksista (Chart.JS 2023c).

Chart.JS kaavion piirtämiseen kuluvan ajan kanssa ilmeni ongelmia karttaohjelman kehityksen aikana. Reittikartta-ohjelmassa kaavion koko sisältö piirrettiin uudestaan jokaisen datapisteen kohdalla, koska kartassa käytettävän kaavion uudelleen piirtäminen oli tekemieni testien perustella nopeampaa kuin kaaviossa olevan datan päivittäminen uusilla arvoilla. Lisäksi kaavion pystyakselin koon muuttuminen isompien arvojen päivittyessä kaavioon toimi mielestäni selkeämmin, kun data piirrettiin kokonaan uudestaan eikä Chart.JS-kirjaston animaatioita käyttämällä. Silti kaavion piirtäminen hidastui mitä pidempään se piirsi dataa, vaikka kaaviossa näytetty datan määrä ei kasvanut.

2.2 Backend

2.2.1 Arduino IDE

Arduino IDE on Arduinon kehittämä avoimen lähdekoodin kehitysympäristö, jossa C++-ohjelmointikielellä ohjelmoidaan Arduino laitteille toimintoja (Arduino 2023a). Myös toiset laitekehittäjät voivat hyödyntää Arduino IDEä omien laitteidensa ohjelmointiin, kunhan huolehtivat laitteen komentojen yhteensovittamisesta Arduino Corejen avulla (Arduino 2023b). Opinnäytetyössä käytettyjen antureiden valmistaja Adafruit tarjoaa Arduino IDE -kehitysympäristöön omille laitteillaan esimerkkejä ja kirjastoja, joiden avulla laitteiden käyttämisen aloittaminen on helppoa (Rembor, Herrada & Rubell 2023).

Arduino IDE:n avulla opinnäytetyössä käytetyn Adafruitin ESP32 V2 Featherin ja siihen lisättyjen antureiden toiminnan ohjelmointi oli varsin yksinkertaista. Varsinkin Adafruitin esimerkeistä oli suurta apua kehityksen alkuvaiheessa. Reaaliaikaista ohjelmointikielen kieliovin tarkistusta tosin jäin kaipaamaan Arduino IDE:ssä, ettei koodissa mahdollisesti olevia syntaksi virheitä olisi tarvinnut tarkistaa hitaalla koodin kääntämisellä. Käyttämäni Arduino IDE:n versio toimi välillä myös virheellisesti, koska se ei jokaisen käynnistyksen yhteydessä ladannut kolmansien osapuolien tekemiä esimerkkejä IDE:ssä olevaan valikkoon, vaikka ESP32-laite oli yhdistettynä tietokoneeseen.

Opinnäytetyössä ESP32-laitteelle tehdyssä koodissa liikettä mittaavilta antureilta kerättiin dataa, joka tallennetaan microSD-muistikortille ja välitettiin USB-Serial porttiin. Kerätystä datasta laskettuja arvoja näytettiin myös laitteeseen liitettyllä pienellä näytöllä, jotta datasta tehtyjen laskelmien toimivuutta oli mahdollista arvioida reaaliajassa. Näytössä olevien nappien avulla datankeräyksen toimintaa pystyi hallinnoimaan. Laitteessa olevan ledin avulla ilmoitettiin tallennustilan päällä olosta.

Opinnäytetyötä tehdessä huomasin, että Adafruitin yleiset esimerkit Arduino IDE:ssä eivät saattaneet toimia uudemmilla Adafruitin antureilla. Kuitenkin

Adafruitin tarjoamia Arduino IDE:n sketchejä oli suhteellisen helppoa muokata toimiviksi myös uudempien antureiden kanssa, koska niiden muokkaamiseen pystyi hyödyntämään Adafruitin antureille tekemiä esimerkkejä. Esimerkiksi tässä projektissa käytetyn magnetismia mittaava magnetometrin kalibrointi MotionCal-ohjelman avulla ei toiminut Adafruitin tarjoamien ohjeiden mukaisesti. Syyksi toimimattomuudelle paljastui se, että Adafruitin kalibrointi esimerkissä käyttämä Arduino-sketchi ei pystynyt lukemaan projektissa käytettyä Adafruitin IMU-anturia. Kyseisen IMU-anturin lukemisen lisäämisen jälkeen MotionCal-ohjelmaa pystyi käyttämään Adafruitin ohjeiden mukaan.

2.2.2 Python

Python on korkeamman tason ohjelmointikieli, missä tekstin sisennykset ja kaksoispisteet ohjaavat koodin osien suorittamista muissa ohjelmointikielissä käytettyjen hakasulkujen ja puolipisteiden sijaan. Python muuttujille ei myöskään tarvitse määritellä tyyppejä (Volkman 2020). Pythonissa suoritussäikeet eivät toimi rinnakkain, koska niiden toimintaa rajoitetaan muistin säieturvallisuuden vuoksi (Python Wiki 2020). Yhtäaikaista suorittamista Pythonissa on kuitenkin mahdollista käyttämällä aliprosessia, koska se ohittaa yhtäaikaista suoritusta rajoittavan globaalin Python-tulkin lukon (Python 2023). TIOBEN (2023) tekemän vertailun perusteella Python on kasvattanut suosiotaan viime vuosien aikana. Python-ohjelmointikielen opiskeluun löytyykin runsaasti ohjeita ja esimerkkejä monesta eri lähteestä (Wdzięczna 2019).

Pythonin valitseminen Raspberry Pillä toteutettavan datankeruun kehityksessä käytettäväksi ohjelmointikieleksi oli varsin selkeää. Suurimmalle osalle Adafruitin antureista on saatavilla Python-ohjelmointikieleen perustuvat kirjastot, joiden avulla antureita on helppo käyttää (Rembor 2017). Lisäksi Python on myös osa Raspberry PI -käyttöjärjestelmää (Raspberry Pi 2023).

Opinnäytetyössä Python-ohjelmointikielellä toteutettiin Raspberry Pillä tapahtuva datan keräys siihen liitetyiltä antureilta. Kerätty data tallennetaan

tiedostoon ja välitetään samanaikaisesti Python-kielellä toteutetulle web-rajapinnalle. Web-rajapinnan avulla sen aliprosessissa toimivan datan keräyksen toimintaa oli mahdollista hallita. Opinnäytetyössä käytettyjen Docker-kuvien käynnistämisen toteutin myös Python-skriptin avulla. Pythonin avulla kehittämisessä ei varsinaisesti suurempia ongelmia tullut esiin, ainoastaan pelkällä tekstieditorilla koodia kirjoittaessa aikaa kului paljon koodin sisennysten muuttamiseen koodissa tapahtuneiden rakenteellisten muutosten jälkeen.

2.2.3 CircuitPython

Adafruit sponsoroi mikrokontrollereiden ohjelmointiin tarkoitettua Python-kieleen perustuvaa CircuitPythonin ohjelmointikielen kehittämistä. CircuitPythonista löytyy tuki useimmille mikrokontrollereille. Se on hyvin samanlainen MicroPython-ohjelmointikielen kanssa, mutta eroaa ilmeisesti joidenkin osien osalta. (CircuitPython 2023.) CircuitPython-kirjastoihin perustuvia esimerkki ohjeita löytyykin useimpien Adafruitin laitteiden käyttöohjeissa (LeBlanc-Williams 2023a).

Jotta CircuitPython-kirjastojen avulla voitaisiin kommunikoida Raspberry Piin liitettyjen lisälaitteiden kanssa Raspberry Pi -käyttöjärjestelmässä, pitää CircuitPythonin mikrokontrolleri käskyt muuttaa yhteensopiviksi Raspberry Pi -käyttöjärjestelmän käskyjen kanssa. Tätä varten Adafruit tarjoaa Blinka-yhteensopivuuskirjastoa, jonka avulla CircuitPython-kirjastojen funktio kutsut välittyvät Raspberry Pin GPIO-pinneihin. (LeBlanc-Williams 2023b.) Raspberry Pi:llä tarvittavien kirjastojen asentamista varten LeBlanc-Williams (2023c) oli kirjoittanut kattavat ohjeet. Kirjastojen asennuksen jälkeen muutamalla import-komennolla saa CircuitPython-kirjastojen komennot käytettäväksi Pythonin työpöytä versiossa (LeBlanc-Williams 2023c).

Adafruitin tuki Python-kielen puolella oli joidenkin laitteiden osalta hiukan heikompi kuin Arduino IDE:ssä. Esimerkiksi tässä opinnäytetyössä käytetyn ICM-20948 IMU:ssa olevan lämpötila-anturin tuloksien palautusta ei ollut toteutettu Adafruitin ICM- laiteperheelle tarkoitettussa CircuitPython-kirjastossa (Siepert

2020a). Siepertin (2020a) lähdekoodissa olevista kommentteista kuitenkin selvisi, että hän oli hyödyntänyt SparkFun Electronicsin vastaavaa Python-kirjastoja, jossa lämpötila tuloksien palautus oli toteutettu.

2.2.4 Flask

Koska verkkoselaimien same-origin policy -tietoturva rajoitteet rajoittivat tiedoston käsittelyä JavaScriptillä, täytyi Rasperryllä Pillä toimivalle ohjelmalle löytyä jokin HTTP-protokollaa käyttävä palvelin ratkaisu (MDN contributors 2023m). Web-palvelimen piti olla mahdollisimman vähän resursseja käyttävä. Palvelimen piti olla myös Python-ohjelmointikielellä toteutettu, jotta antureiden keräämän datan siirto Python-skriptistä kyseisen palvelimen välitettäväksi oli mahdollisimman yksinkertaista toteuttaa.

JetBrainsin Python-kehittäjä kyselyn vastausten perusteella suosituimpia avoimia web-kehys ratkaisuja olisivat olleet Django, Flask ja FastApi (JetBrains 2022). Django vaikutti sen kotisivuilla olevien esimerkkien perusteella hiukan toisia web-kehyskiä vaikeampi käyttöisemmältä (Django 2023). Flaskin kotisivuilla olevat esimerkit olivat mielestäni selkeämpiä, kuin Django esimerkit. (Flask 2023a). Työharjoittelussa tekemässäni projektissa käytin JetBrainsin kyselyn kolmanneksi suosituinta FastAPI-frameworkia (JetBrains 2022). Rajapinnan tekeminen FastAPI:n avulla olisi ollut vielä yksinkertaisempaa kuin Flaskilla, mutta se aiheutti ongelmia työharjoittelussa tekemässäni projektissa. Jokin FastAPI:n toteutuksen käyttämä osa aiheutti ongelmia Pythonin aliprosessien suoritukseen ja siitä seurasi ongelmia itse FastAPI:n suoritukseen. FastAPI ei siis ollut vaihtoehtona tähän opinnäytetyöhön.

Projektissa käytettäväksi web-palvelin ratkaisuksi valitsin Flask-kehiksen, joka on kokoelma web-palvelimille tehtyjä työkaluja web-rajapinnan toteuttamiseksi. Flaskin lähdekoodi käyttää BSD-3-Clause-lisenssiä (Flask 2010). Flaskin kotisivuilta löytyi esimerkkejä, kuinka web-rajapinta voidaan luoda. (Flask 2023a.)

Yksi Flaskin mukana tulevista työkaluista oli Jinja, joka vastasi Flaskin avulla palveltavien HTML-sivujen malleista (Flask 2023b). Jinjan avulla HTML-sivun lähdekoodin sisältöä pystyisi rakentamaan dynaamisesti Python-ohjelmointikielen kaltaisen kielen avulla (Jinja 2023). Jinjan avulla on myös mahdollista estää ohjelmoinnissa käytettävien merkkien käyttäminen verkkosivun käyttäjiltä, eli parantaa verkkosivun tietoturvaa (Flask 2023c). Jinja ilmeisesti määritteli myös Flaskin vaatiman verkkosivun hakemisto rakenteen.

Flaskin työkalupakissa on myös Werkzeug WSGI-kirjasto, joka välittää viestejä Flaskin Python-rajapinnan ja palvelimen välillä (Flask 2023d). Werkzeugin (2023) mukana tulee muitakin web-palvelimen yhteyksiin liittyviä apuohjelmia. Projektin kannalta tärkein osa Werkzeugin (2023) apuohjelmista oli pieni testi palvelin, joka voi ottaa vastaan ja välittää HTTP-protokolla pyyntöjä testausta varten. Tämä pieni testi palvelin mahdollisti opinnäytetyössä datan keräyksessä käytettävien HTTP-metodi kutsujen vastaan ottamisen Flaskilla toteutetun rajapinnan avulla.

Täydelliselle web-palvelimelle ei tässä projektissa ollut vielä käyttöä, koska datan keräyksen hallinnointiin käytetty rajapinta ei ollut Internetin kautta käytettävissä. Lisäksi rajapinta toimi vain paikallisesti ja palveltavana oli vain yksi käyttäjä. Raspberry Pin suoritus kykykin aiheutti epäilyksiä siitä voisiko opinnäytetyössä käyttää kokonaisempaa web-palvelin ratkaisua, kaikkien muiden käytettyjen ohjelmien rinnalla. Toimintojen päällekkäisyyksiäkin syntyi lopulta, kun opinnäytetyössä otin käyttöön tuli Docker-kuvia, missä jokaisessa oli oma web-palvelin.

2.3 Avoimet ohjelmat ja aineistot

2.3.1 Project OSRM

Project OSRM on BSD-2-Clause lisenssiä käyttävä avoimen lähdekoodin ohjelmisto, jonka avulla pystyy selvittämään lyhyimmät kulkureitit OpenStreetMapin

tarjoaman datan avulla. Ohjelmisto on saatavilla myös valmiina Docker-paketina, jossa oli myös mukana web-palvelin rajapintaa varten. (Project-OSMR 2022.) Web-rajapinnan käyttöön Project OSRM -kotisivuilla oli varsin kattavilta vaikuttavat ohjeet, missä rajapinnan komentojen käyttöä oli kuvailtu esimerkkien ja palautusarvojen avulla (OSMR 2023). Lisäksi Docker-paketin käyttöönotto oli ohjeistettu Project OSRM:n GitHub Packages -sivuilla (Project-OSMR 2022).

Opinnäytetyössä käytin Project OSRM -backendiä saatavilla olleen Docker-kuvan avulla, jotta karttaan liitettävän reitinhaun käyttäminen oli mahdollista myös ilman verkkoyhteyttä. OSRM-testipalvelin oli myös kehityksen aikana testattavissa internetin välityksellä, mutta sille tehtävien kyselyjen määrää oli rajoitettu (Project-OSMR 2020). Kehityksen aikana hiukan ongelmia aiheutti Docker-kuvien määrittely niiden käyttöönoton yhteydessä. Jonkin verran joutui miettimään ja kokeilemaan mitä kukin syötettävä komento tarkoitti, koska joitakin niiden asetuksista piti muuttaa ennen niiden käyttämistä. Muutokset koskivat käytettäviä portteja ja käytetyn datan sijaintia.

2.3.2 OpenStreetMap

OpenStreetMap on suosittu Open Database -lisenssiä käyttävä käyttäjien rakentama karttakokoelma, jota kuka tahansa voi käyttää ja muokata (OpenStreetMaps 2023a). Karttakokoelmaan pystyy myös tekemään karttoja olemassa olevien karttojen pohjalta, mutta niiden lisenssien pitää vähintään vastata Open Database -lisenssin ehtoja, tai niiden käyttämiseen pitää saada lupa lisenssin todelliselta haltijalta (The OpenStreetMap Foundation 2023). Kokoelmassa oleviin karttoihin merkataan kulkureitit, katujen nimiä ja osoitteita, joita voi hyödyntää niiden sijaintien selvittämisessä (OpenStreetMaps 2023b). Osoitteiden sijaintien etsimiseen kartta-aineistosta OpenStreetMap tarjoaa Nominatim-rajapintapalvelua (OpenStreetMaps 2023c).

OpenStreetMap Foundation varaa lahjoitusvaroin ylläpitämiensä servereiden suorituskykyä karttojen tekijöille. OpenStreetMap-karttakuvien latausrajapinnan käyttämiseen onkin asetettu vaatimuksia millä rajapintakutsujen määrää ja data-liikennettä pyritään vähentämään. Kuvien raskas käyttö, eli rajapintakutsujen jatkuva käyttäminen jaossa olevissa ohjelmissa on kielletty ilman lupaa. Rajapintaa käyttävien ohjelmien vaatimuksina on kuvien tallentaminen välimuistiin ja rajapintakutsuissa oleva yksilöllinen tunniste, jonka avulla mahdollinen väärinkäyttö voidaan estää. (OSMF Operations Working Group 2023.)

Käytettävissä oli myös kolmansien osapuolien ylläpitämiä ilmaisia ja maksullisia OpenStreetMapin kartta-aineistoja jakavia tile-palvelimia (OpenStreetMaps 2023d). Mahdollisuutena oli myös oman OpenStreetMaps aineiston kuvarajapinnan ylläpito, mitä varten oli saatavilla valmiita palvelinohjelmistoja eri ohjelmointikielellä toteutettuna (OpenStreetMaps 2023e). OpenStreetMapsin tile-palvelimesta ja Nominatim-rajapintapalvelusta oli saatavilla myös valmiit Docker-kuvat (Overvoorde 2023; Mediagis 2024).

Opinnäytetyössä käytettiin OpenStreetMapsin tile-karttapalvelinta ja Nominatim-rajapintapalvelua niistä tehtyjen valmiiden Docker-kuvien avulla. Paikallisesti käyttäessä OpenStreetMapsin tile-palvelimelle menevien pyyntöjen määrää ei tarvinnut rajata ja niiden käyttäminen oli mahdollista ilman internet-yhteyttä. Varsinkin Nominatim-rajapintapalvelun käyttö navigointi käytössä vaati sen paikallista käyttämistä, koska rajapintapalveluun tehtävien kyselyjen määrä oli erittäin rajattu.

2.3.3 Maanmittauslaitoksen avoimet aineistot

Maanmittauslaitoksen (2023a) mukaan se avasi aineistonsa vapaaseen käyttöön 2012. Nykyisin maanmittauslaitoksen avoimia aineistoja on mahdollista ladata maanmittauslaitoksen tarjoaman karttapaikka-palvelun avulla (Maanmittauslaitos 2023a). Karttapaikka-palvelun kautta on mahdollista ladata kuvia,

rasteroituja karttoja sekä vektori muotoisia aineistoja eri formaateissa (Maanmittauslaitos 2023b.) Maanmittauslaitos tarjoaa myös rajapintapalvelua, jonka käyttö vaatii vähintäänkin rekisteröitymistä API-avaimen saamiseksi (Maanmittauslaitos 2023c). Rajapinnan kautta avoimien aineistojen lisäksi on mahdollista hakea muun muassa osoitteiden koordinaatteja (Maanmittauslaitos 2023d). Maanmittauslaitoksen avoimet aineistot käyttävät Nimeä 4.0 Kansainvälinen -lisenssiä (Maanmittauslaitos 2023e).

Opinnäytetyössä maanmittauslaitoksen avoimien aineistojen hyödyntäminen kohdistui valmiiksi rasteroituihin maastokarttoihin ja ilmakuviin, jotka olivat ladattavissa karttapaikka-palvelun kautta. Tekemieni lataustestien perusteella maastokartat olivat saatavilla PNG-tiedostoformaattiin tallennettuina kuvina ja kuvan paikannukseen tarvittavat tiedot olivat PGW-tiedostoformaattissa. Ilmakuvat olivat saatavilla JPEG2000-tiedostoformaatin kuvatiedostoina, joiden lähdekoodissa oli kuvan paikannukseen tarvittavat tiedot. (Maanmittauslaitos 2023f.)

Maanmittauslaitoksen rasterikartat ovat Suomessa käytetyn ETRS35-FIN-tasokoordinaatisto systeemin mukaisia (Häkli, Puupponen, Koivula & Poutanen 2009, 19–20, 29–30). Häklin ym. (2009, 38) mukaan muunnokset ja käytetyt koordinaattijärjestelmät eivät ole täydellisiä, joten karttojen sijainneissa voi olla jonkin verran epätarkkuutta. Karttapaikasta ladatut kartat voi muuntaa muiden karttaohjelmien kanssa yhteensopiviksi, GDAL-muunnoskirjaston ja sitä käyttävien apuohjelmien avulla (Maanmittauslaitos 2023g). Yksi apuohjelmista on gdal2tiles, jonka avulla kartat voi pilkkoa pienemmiksi kuviksi ja tallentaa ne Leaflet-kirjaston käyttämän hakemisto rakenteen mukaisesti (GDAL 2023).

Maastokarttojen rasterikuvien PNG-tiedostojen väripaletti piti muuttaa RGB/RGBA muotoon, jotta niitä pystyi käsittelemään gdal2tiles avulla. Maastokuvien JPEG2000-tiedostot oli mahdollista pilkkoa gdal2tilesin avulla ilman muunnoksia. Ilmeisesti gdal2tilesin avulla pilkkoessa kuvien peittämän alueen ulkoreunat eivät ole pilkkomisen jälkeen suoria, joten kartta-alueiden liittäminen kokonaisuudeksi olisi tehtävä ennen niiden pilkkomista. Toinen mahdollisuus

olisi limittäisten lähde kuvien käyttö, mutta Maamittauslaitoksen Karttapaikka-palvelun kautta niiden lataaminen ei ollut mahdollista.

2.3.4 Ilmatieteen laitoksen avoin data

Ilmatieteen laitos on avannut osan aineistoistaan ilmaiseen käyttöön (Ilmatieteen laitos 2023a). Ilmatieteen verkkopalvelussa jaossa olevat avoimet aineistot käyttävät Nimeä 4.0 kansainvälinen -lisenssiä (Ilmatieteen laitos 2023b). Ilmatieteen laitoksen avoin aineisto on käytettävissä Ilmatieteen laitoksen tarjoamien lataus- ja katselupalvelun kautta (Ilmatieteen laitos 2023a).

Latauspalvelu on toteutettu WFS 2.0 -standardin mukaisesti, jonka kautta saa ladattua yksityiskohtaisempaa tietoa XML-muotoisena datana valmiiksi määritellyillä komennoilla (Ilmatieteen laitos 2023c). Katselupalvelu on toteutettu WMS-rajapinta standardin mukaisesti, jonka kautta saa ladattua kuvatiedostoja. Katselupalvelun käyttöä rajoitetaan, jos palvelin on suuren kuormituksen alla (Ilmatieteen laitos 2023d). Jos Ilmatieteen laitoksen WMS-rajapinnan kautta saatavia kuvia aikoo jakaa toisille, täytyy silloin kuvat ladata GeoTIFF-muotoisina ja jakaa se oman palvelimen kautta (ilmatieteen laitos 2023c).

Ilmatieteen laitos päivittää avoimia aineistojaan tietyin aikavälein (Ilmatieteen laitos 2023d). Opinnäytetyössä käyttämäni sade ja salama aineistot päivittyvät Ilmatieteen laitoksen (2023d) mukaan viiden minuutin ja yhden sekunnin välein. Kaikki ilmatieteen avoimet aineistot käyttävät UTC-aikaa (Ilmatieteen laitos 2023f). Ilmatieteen laitoksen rajapintoihin tekemiäni kyselyjen perusteella avoimet aineistot olivat EPSG:4326-koordinaattijärjestelmässä. EPSG:4326 on WGS84-ellipsoidiin perustuva kaksiulotteinen koordinaattijärjestelmä (Jurkowska 2023). Ilmatieteen laitokselta saatava data oli siis yhteen sopiva GPS-paikannustietojen ja Leaflet-kirjaston kanssa.

Opinnäytetyötä tehdessä hieman sekavalta vaikutti Ilmatieteen laitoksen verkkosivujen Avoid data -osuus. Tiedot olivat saatavilla kahdelle eri kielellä omissa osioissaan, jotka eivät sisältäneet täysin samoja asioita. WFS-palvelun valmiit kyselyt olivat nähtävissä verkkosivuilla ja kyselyn avulla saatavissa latauspalvelusta, mutta esimerkkejä kyselyjen palauttamista arvoista ei ollut saatavilla. Muutenkin osa rajapinnan tiedoista vaikutti olevan ripoteltuna useammille verkkosivuille, vaikka ne mielestäni olisivat olleet selkeämmin käytettävissä yhdeltä sivulta.

Opinnäytetyössä tarkoituksena oli hyödyntää Ilmatieteen laitoksen WMS- ja WFS- rajapintoja, kun internet-yhteys olisi saatavilla. Reittikartta-ohjelma hakee WMS-rajapinnasta Suomen päällä olevien sadealueiden kuvan viiden minuutin välein, kun kyseinen kuva on näkyvissä ohjelman kartassa. Salama kyselyn toteuttaminen ei opinnäytetyötä tehdessä ollut mahdollista, koska esimerkki dataa ei ollut saatavilla ja salamoita ei esiintynyt kehityksen aikana. Ilmatieteen laitoksen datan avulla opinnäytetyön Reittikartta-ohjelmassa olisi mahdollista tarkistaa sade ja ukkospilvien läheisyys, mutta sen lopullinen toteuttaminen jäi myöhemmäksi.

2.4 Laitteet

2.4.1 Adafruit ESP32 Feather V2/Espressif ESP32

Pelkkää dataa keräävän laitteen rooliin etsin mahdollisimman tehokasta, mutta samalla mahdollisimman vähän virtaa kuluttavaa edullista laitekokonaisuutta. Laitteen oli oltava kooltaan pieni ja toimittava pienen ladattavan akun avulla, jotta sitä olisi helppo kuljettaa mukana. Laitteen kasaamisen piti olla myös mahdollista vähemmän juottamista harjoitelleelle, joten piirilevyjen suunnittelu ja pienten komponenttien juottaminen ei tullut kyseeseen. Lisäksi laitteen ohjelmointiin piti olla saatavilla helposti luettavat ohjeet.

Vaatimukset pienestä koosta ja toimimisesta yhden ladattavan litiumionimakun jännitteellä (3,7V) karsi mahdollisten laitteiden valikoiman mikrokontrollereihin. Raspberryn ja Adafruitin tuotteet tulivat tutuiksi jo työharjoittelussa, joten tiesin näiden valmistuttajien tarjoavan myös ohjeita laitteidensa ohjelmointiin. Laitteiden ohjelmointiohjeiden tarjonnan puolesta AdaFruit oli mielestäni hiukan parempi, kuin Raspberry Pi. Varsinkin Adafruitin lisälaitteiden puolella Adafruitin ohjelmointikoodi esimerkit nopeuttivat kehityksen aloittamista. Raspberry Pin omien lisälaitteiden valikoima oli myös rajatumpi kuin Adafruitin valikoima (Raspberry Pi 2023a; Adafruit 2023a).

Raspberry Pi -tuoteperheeseen kuuluva Raspberry Pi Pico -mikrokontrolleri vaikutti täyttävän asettamani tavoitteet, hinnasta, virran kulutuksesta ja koosta. Ainoastaan vähäinen keskusmuistin määrä ja massatallennustilan pienuus aiheutti epäilyksiä soveltuvuudesta projektiin. (Raspberry Pi 2023b.) Etsiessäni lisätietoa Picosta löysin vertailun, jossa Picoa vertailtiin ESP32-järjestelmäpiirillä varustettuun mikrokontrolleriin. Olujinmin (2022) tekemässä vertailussa mukana ollut ESP32-järjestelmäpiirillä varustettu mikrokontrolleri vaikutti paremmin soveltuvulta projektiin kuin Pico. Vertailussa mukana olleella ESP32-järjestelmäpiirillä oli korkeamman kellotaajuuden omaava suoritin sekä enemmän keskusmuistia ja tallennustilaa (Olujinmin 2022). Adafruitin tuotevalikoimasta löytyi samantapaisella ESP32-järjestelmäpiirillä varustettu Feather-mikrokontrolleri (Rembor 2023a).

Adafruitin ESP32 Feather V2 -piirilevyllä olevassa ESP32-järjestelmäpiirin moduulissa suorittimen lisäksi on 2MB PSRAM-muistia ja 8MB Flash-muistia, joiden avulla datan keräyksessä käyttämäni ohjelma pystyi toimimaan hyvin. Lisäksi järjestelmäpiirin moduulissa on Bluetooth- ja Wifi-valmius. (Rembor 2023a.) Piirilevyllä oleva kaksinapainen JST-liitin mahdollisti virran syöttämisen Litiumioniakun avulla (Rembor 2023b). Adafruit (2023a) Feather-tuoteperheeseen kuuluvia lisälaitteita käyttämällä olisi laitteet mahdollista liittää toisiinsa pinnottavien piikkirimojen avulla.

Adafruitin ESP32 Featherin Power Management -ohjeessa mainittiin, että USB-yhteyden pitäisi ottaa virran antajan rooli, kun se liitetään ja silloin JST-liitimeen liitetty akku toimisi varavirtalähteenä (Rembor 2022a). USB-varavirtalähdettä käyttäessä huomasin, että Adafruitin ESP32 Feather sammui, jos ensimmäisenä kytketty akku irrotettiin, vaikka toisena kytketty varavirtalähde olisi edelleen ollut yhdistettynä USB-porttiin. Ilmeisesti kyseisen varavirtalähteen tarvitsemaa virran kulutuksen vähimmäismäärää ei tapahtunut USB-kaapelia yhdistäessä ESP32 Featherin USB-porttiin, jos akku oli jo kytkettynä (Anker 2023). Sama ongelma toistui myös käyttäessä joitakin verkkovirralla toimivia USB-latureita. Käyttämällä jatkuvasti virtaa ulos antavaa USB-yhteyttä, kuten tietokoneen USB-porttia, tätä virranmenetys ongelmaa ei esiintynyt. Tämän ongelman pystyi myös huomaan piirilevyllä olevasta latauksen tilaa ilmoittavasta ledistä. Lataustilan oranssi ledi ei syttynyt, kun akku oli kytkettynä ennen USB-yhteyden muodostusta.

ESP32 Featherin piirilevyllä oleva STEMMA QT -liitin oli sijoitettu mielestäni huonoon paikkaan. Siihen ei voinut liittää I2C-väylää käyttäviä antureita, jos Feather-lisälaite oli kiinnitettynä ESP32 Featherin piirilevyn päälle. Samalla piirilevyllä olevien painikkeiden käyttö estyi. Adafruitin Feather tuoteperheen näytöissä on varmaankin juuri tästä syystä STEMMA QT -liitin ja painikkeita (Adafruit 2023b).

2.4.2 Raspberry Pi 4 Model B

Työharjoittelussa tutustuin Raspberry Pi 4 -tietokoneen käyttämiseen jo varsin kattavasti, joten sen valinta opinnäytetyössä käytettäväksi oli varsin helppo. Työharjoittelussa saamani kokemuksen perusteella Raspberry Pi 4 toimisi tarvittavien antureiden ja kosketusnäytön kanssa viiden voltin jännitteellä ja noin kahden ampeerin sähkövirralla, joten sen käyttäminen onnistuisi USB-varavirtalähteiden ja auton tupakansytyttimeen liitettävän USB-adapterin avulla.

Raspberryn Pi 4 Model B:n suorittimen suorituskyvynkin piti olla riittävä tarvittavien ohjelmien ajamiseen. Lisäksi työharjoittelun aikana käyttämäni kehitysvälineetkin toimisivat ilman odottamattomia ongelmia.

Suuren suosion saavuttaneita Raspberry Pi-laitteita oli hankala ostaa, maailmaa vaivanneen sirupulan takia (Cunningham 2023). Loppu kesästä 2023 saatavuus oli sen verran parantunut, että yhdestä suuresta elektronisia komponentteja myyvistä kaupasta Raspberry Pi:n sai ostettua ilman ylimääräistä odottelua. Edullisemmat versiot vähemmällä järjestelmämuistilla oli vielä silloinkin loppuun myytyjä, mutta Raspberry Pi 4 Model B 8GB -versioita oli saatavilla.

8GB järjestelmämuistin yhtenäinen käyttäminen vaatii 64-bittisen Raspberry Pi -käyttöjärjestelmän. 32-bittisessä Raspberry Pi -käyttöjärjestelmässä ohjelma voisi käyttää korkeintaan 3GB:iä muistia prosessia kohden. (Hollingworth 2022.) Yli 3GB:iä järjestelmämuistia tässä projektissa mahdollisesti olisi tarvinnut Project OSMR -reitinselvityspalvelin, joka latsi käytetyn alueen kaikki kulkureitit järjestelmämuistiin (Project-OSMR 2021). OpenStreetMapsin data Suomesta ei tosin ollut vielä kovin suuri opinnäytetyön tekemisen aikaan.

2.4.3 Adafruit Ultimate GPS/CDtop technology PA1616D

Työharjoittelussa saamani kokemuksen perusteella tiesin, että paikannustietoa laskevassa GPS-moduulissa piti olla muisti, eli patteri minkä avulla se pystyisi säilyttämään tietoa edellisistä paikannussatelliiteista. GPS-moduulissa piti myös olla paikka mihin oli mahdollista liittää ulkoinen GPS-antenni. GPS-moduulin käynnistyksen yhteydessä tarpeeksi monen paikannussatelliitin signaalin vastaanottaminen saattoi kestää lähes puoli tuntia, jos GPS-moduuli ei muistanut edellisiä signaaleja. Ilman ulkoista GPS-antennia signaalin saaminen tarpeeksi monesta paikannussatelliitista saattoi myös olla hidasta, kun laite oli auton tai rakennuksen sisällä. Lisäksi GPS-moduulin piti tukea mahdollisimman montaa GPS-satelliittijärjestelmää, tarkkuuden parantamiseksi.

Adafruitin tuote valikoimasta oli yksi ehdot täyttävä laite, jossa oli CDtop technologyn MTK3333 paikannus -moduuli (Adafruit 2023c). MTK3333 perustuu Mediatekin MT3333 GPS-mikrosiruun (CDtop technology 2023, 3). Mediatekin (2016, 2) dokumentin mukaan MT3333-mikrosiru tukisi eurooppalaisen Galileo-satelliittijärjestelmän lisäksi myös muita suuria satelliittipaikannusjärjestelmiä (GPS, GLONASS, BeiDou). CDtop technologyn (2023, 14) dokumentissa kuitenkin mainitaan, että vain GPS ja GLONASS olisi tuettuna MTK3333-moduulin laiteohjaimen palauttamissa viesteissä. Dokumentissa mainitaan, että satelliitti pohjaisia avustusjärjestelmiä (SBAS) on mahdollista käyttää vain 5Hz tai sitä hitaammilla päivitysnopeuksilla (CDtop technology 2023, 4). Dokumentissa mainittiin myös, että moduulista olisi mahdollista saada myös suuntakulma ja muita laskettuja arvoja sen hetkisen nopeuden lisäksi (CDtop technology 2023, 21).

Satelliiteista saadaan hyvin tarkka UTC-aika käyttöön, sillä satelliitit käyttävät atomikelloja. Tosin huomioon on otettava se, että kaikki satelliittijärjestelmät (GLONASS) eivät korjaa atomiaikaa karkaussekunneilla, kuten normaalissa UTC-ajassa tehdään (Sanz Subirana, Juan Zornoza & Hernández-Pajares 2011). Jotta GPS-moduuli voisi arvioida sijainnin tarkasti pitää sen ottaa huomiiin eri satelliittijärjestelmien tavat mitata aikaa, koska satelliitti paikannus perustuu aikaan ja satelliittien sen hetkiseen sijaintiin (Maanmittauslaitos 2023h).

Paikannussatelliiteista saatavien sijaintitietojen tarkkuus vaihtelee eri satelliittijärjestelmien välillä (SpaceFinland 2023). Paikannustietojen tarkkuutta voidaan parantaa satelliitti avustejärjestelmien korjaus viestien avulla, jos GPS-moduuli tukee niitä (Maanmittauslaitos 2023h). Kuten myös käyttämällä GPS-moduulia, jonka avulla voidaan käyttää useampia järjestelmiä yhtä aikaa (GPS.gov 2023). Mitä useampaa paikannussatelliittia voidaan käyttää paikanlaskemiseen, sitä tarkempi saatu tulos on (Maanmittauslaitos 2023h). GPS-moduulin ympärillä olevat esteet voivat huonontaa paikannuksen tarkkuutta paikannussatelliittien lähettämien signaalien heijastumien takia (SpaceFinland 2023).

GPS-moduulista oli mahdollista saada GPS-moduulin antennin korkeus WGS 84 -ellipsoidista johdetusta arvosta, missä on huomioitu maan vetovoiman

vaikutus. Eli korkeus lasketusta meren pinnasta sekä sen ja WGS 84 -ellipsoidin välinen korkeus ero. (Fraczek 2003, 1.) Adafruitin GPS-laitekirjaston palauttama antennin korkeus arvo vaikutti olevan korkeus WGS 84 -ellipsoidista, koska sen palauttaman arvot vastasivat paremmin maanmittauslaitoksen maastokartassa olevia korkeusarvoja, kun siitä oli vähennetty WGS 84 -ellipsoidin ja lasketun merenpinnan välinen ero. GPS-moduulista saadut sijainti koordinaatit oli desimaaliminuutteina (ddmm.mmmm) (CDtop technology 2023, 15). Adafruitin GPS-laitekirjaston avulla koordinaatit oli mahdollista saada desimaaliasteina, jolloin niitä pystyi käyttämään ilman muunnoksia Leaflet-kirjaston metodeissa.

GPS-paikannuksesta saatua dataa tässä opinnäytetyössä käytettiin ajan, sijainnin ja nopeuden mittaamiseen. Aikaa käytettiin merkitsemään kerätyn datan keräyksen ajankohtaa ja samalla datan keräyksen kestoa. Sijainnista saatujen arvojen avulla saatiin haettua oikea paikka kartasta ja merkattua siihen sen hetkisen sijainti. Nopeus arvoja käytettiin sen hetkisen nopeuden näyttämiseen sekä nopeuden keskiarvon laskemiseen. Nopeuden keskiarvon ja ajan avulla kuljettu matkakin oli mahdollista selvittää suhteellisen tarkasti.

2.4.4 Adafruit ICM20948 IMU/Invensense ICM20948

Opinnäytetyössä inertiaa mittaavan anturin avulla pyrittiin selvittämään datan keräyksessä käytetyn laitteen osoittama ilmansuunta. Mitatun ilmansuunnan perusteella karttaa oli tarkoitus kääntää laitteen osoittaman suunnan mukaan. Liikkeessä etenemissuunnan selvitys onnistui myös paikannustietojen avulla, mutta paikannustietojen vaihteleva tarkkuus aiheutti lasketussa suunnassa liian paljon heittelyä.

Tavallisen kompassin tapaan IMU-moduulissa olevan magnetometrin avulla pystytään selvittämään mihin ilmansuuntaan laite osoittaa. Magnetometri palauttaa suuntavektorin, joka osoittaa magneettisen kentän mukaisesti kohti magneettista pohjoisnapaa, joka poikkeaa karttojen pohjoisnavasta. Mittaustavan takia IMU-moduulin asento vaikuttaa magnetometrin tuloksiin, joten tuloksien korjaamiseksi laskuissa pitää huomioida moduulin orientaatio. Lisäksi

magnetometrin kalibrointi käyttökohteessa on tärkeää, sillä kaikki läheiset magneettiset kohteet vaikuttavat mittaustuloksiin. (Caruso 2020.)

Laiteen ollessa paikallaan IMU-moduulin orientaation selvittäminen onnistuu siinä olevan kiihtyvyysanturin avulla. Maan painovoimasta kiihtyvyysanturilla saadaan suunta vektori, josta IMU-moduulin orientaatio voidaan laskea. (Pedley 2013, 2.) Liikkeessä kiihtyvyysanturilla lasketun orientaation tarkkuus heikkenee anturin toisten akseleiden suuntaisten liikkeiden lisääntyessä, joten laskelmiin pitää ottaa mukaan kulmakiihtyvyysanturi (McWorther 2019).

IMU-moduulissa oleva kulmakiihtyvyysanturi mittaa siihen kohdistuvaa pyörimisnopeutta. Kulmakiihtyvyysanturin mittaustulokset ovat vakaita sen akselien suuntaisissa liikkeissä, koska anturi mittaa vain akselien pyörimissuuntaisia muutoksia. Kulmakiihtyvyysanturin mittaaman kulmanmuutoksen ja muutokseen kuluneen ajan avulla kiihtyvyysanturin palauttaman suuntavektorin arvojen muuttumista voidaan tasoittaa. (McWorther 2019.)

Opinnäytetyössä käytetyn kulmakiihtyvyysanturin ongelmana oli mitattujen arvojen muuttuminen, vaikka liikettä ei tapahtunut. Kulmakiihtyvyysanturin kalibrointi auttoi jonkin verran arvojen ajautumiseen. Kalibrointikaan ei estänyt arvojen itsestään ajautumista pidempiaikaisessa mittaamisessa, joten arvoa olisi pitänyt kompensoida aika ajoin. Yhtenä syynä näiden MEMS-tyypisten kulmakiihtyvyysantureiden mittauservojen ajautumiseen on lämpötilan vaihtelu (Prikhodko, Trusov & Shkel 2012, 517). Ilmeisesti tästä syystä opinnäytetyössä käytetyssä Adafruitin IMU-moduulissa oli myös lämpötilamittari.

Opinnäytetyössä käytettävässä Adafruit ICM20948 IMU:ssa kiihtyvyys- ja kulmakiihtyvyysanturina oli TDK:n InvenSense ICM 20486 (Siepert 2020b). Magneettikentän mittauksesta vastasi Siepertin (2020b) mukaan AK09916-magnetometer. Kyseinen AK09916-magnetometer on sisällettynä ICM 20486-mikrosiruun ja se pystyy mittaamaan magnetismia ± 4900 mikrotleslaan asti (TDK 2021, 9–10). Kiihtyvyysanturi pystyy mittaamaan kiihtyvyyttä 16 kertaa maan painovoiman verran ja kulmakiihtyvyysanturi kulmakiihtyvyyttä 2000

astetta sekunnissa (TDK 2021, 10). BitSmashedin mukaan (2023) suurempi kulman asteväli heikentää kulmakiihtyvyyssantureiden tarkkuutta. Sama varmaan-kin pätee myös kiihtyvyyssanturiin, joten tarkempaa tulosta halutessa piti nämä anturit määritellä käyttämään pienempiä mittausta arvoja. TDK:n dokumentista ilmenee joidenkin antureiden akselien suunnat mikrosiruun nähden, jonka suuntaus selviää mikrosirussa olevan pienen pisteen avulla (TDK 2021, 89). Adafruitin piirilevyllä oli ilmeisesti merkitty magnetismia mittaavan anturin X- ja Y-akselien suunnat. TDK:n (2021) tekemässä dokumentissa kuitenkin on erikseen merkitty toisetkin akselit, joista oli mahdollista selvittää kulmakiihtyvyyttä mittaavan anturin akselien pyörimissuuntaa. Dokumentissa ei kuitenkaan mainittu kiihtyvyyttä mittaavan anturin akselien suuntia.

Nämä ICM20948-mikrosirussa olevien anturien akselien suunnat osoittautuivat projektia tehdessä ongelmallisiksi. Ilmeisesti kaikissa saatavilla olevissa asento kompensoiduissa kompassi esimerkeissä käytettiin antureita, joiden akselit osoittivat eri suuntiin. Eri suuntaisen akselit aiheuttivat laskukaavan toimimattomuuden laitteen asennon muuttuessa, joten laskukaavat eivät toimineet halutulla tavalla ICM20948-mikrosirun kanssa.

ICM20948-mikrosirussa ilmeisesti on myös pieni ohjelmoitava FPGA-piiri, jonka avulla on mahdollista käsitellä antureista saatavia arvoja (ZaneL 2021). FPGA-piirin ohjelmointia varten on olemassa laiteohjelma, tosin sen käyttäminen on SparkFun Electronicsin (2021) GitHub-sivuilla olevan kuvauksen mukaan hankalaa. Arduino-laitteille tosin on saatavilla ZaneLin tekemä rajapinta, jonka avulla kyseistä laiteohjelmaa olisi mahdollista käyttää (ZaneL 2021).

Remingtonin (2024) ylläpitämältä Github-sivustolta oli saatavilla ICM20948-mikrosirulle toteutettu ICM_20948-AHRS-ohjelmakirjasto Arduino-laitteille. Ohjelmakirjasto laski kallistuskompensoidun kompassi suunnan. Antureiden kalibrointia varten Remingtonin (2024) GitHub-sivulla oli ohjelmia ja linkkejä ohjeisiin antureiden kalibrointia varten. Kalibroinnista saatuja arvoja käytettiin Remingtonin ohjelmakirjastossa antureiden palauttamien arvojen virheiden korjaamiseen (Remington 2024). Ilman kalibrointia oikean kompassi suunnan laskeminen ei ollut mahdollista.

Ohjelmankirjaston avulla toteutetun kallistuskompensoidun kompassin tarkkuus vaikutti olevan jonkin verran vakaampi kuin mitä GPS-paikannuksen sijainneista oli mahdollista laskea liikkeessä. Kalibroinnin eli laitteen pyörittely sen oman keskipisteen ympäri kalibroinnissa käytettävää dataa varten tosin piti onnistua suhteellisen hyvin, mikä saattoi olla vaikeaa ilman apuvälineitä. Käytetyn magnetometrin mittausta arvot vaihtelivat paljon, kun laitetta ei ollut käytetty vähään aikaan. Projektissa käytetyn magnetometrin kalibrointi pitikin uusia melkein joka kerta kun laite käynnistettiin, jotta kompassi toimi normaalisti.

Opinnäytetyötä varten käänsin Remingtonin ICM_20948-AHRS-ohjelmakirjaston myös Python-kielelle, jotta sitä oli mahdollista käyttää Raspberry Pillä. Kyseinen ohjelmakirjasto käytti SparkFun Electronicsin laitekirjastoa, joka oli saatavilla myös Python-kielellä. SparkFun Electronicsin laitekirjaston käyttäminen tosin aiheutti ongelman missä datan lukeminen sensoreista pysäytti ohjelman suorituksen muutaman sekunnin välein melkein sekunniksi. Python-kielen avulla sensorien lukeminen oli mahdollista eristää omaksi aliprosessiksi. Arduino IDE:llä tehdyssä ohjelmassa se ei kuitenkaan vaikuttanut olevan mahdollista, vaikka sensorien lukeminen olisi suoritettu omassa suoritus silmukassa ja omassa prosessissaan toisella ESP32:ssa olevalla prosessorilla.

2.5 Datansiirto

2.5.1 Muistikortti

Raspberry Pillä toimiva Reittikartta-ohjelman backend tallensi keräämänsä datan reaaliajassa tekstitiedostoon microSD-muistikortille. Muistikortti oli mahdollista tarvittaessa myös irrottaa Raspberry Pistä, mutta se oli varsin suuritöistä käytetyn suojakotelon takia. Tämän takia tallennettua dataa pääasiassa luettiin samalla Raspberry Pillä, jolla data myös tallennettiin.

Muistikorttia Raspberry Pi 4:lle rasitti käyttöjärjestelmä ja Reittikartta-ohjelmisto, joka tallensi suorituksen aikana kerätyn datan tiedostoon. Jos samaan muisti-paikkaan tallentaa usein dataa, kuluttaa se muistikorteissa käytettyä NAND flash -muistia (Micron 2008). Raspberry Pi -käyttöjärjestelmän uudemmat versiot sisältävät valmiiksi asennettuna muistikortin testaus ohjelman, jonka avulla voi testata onko käytetyn muistikortin lukunopeus tarpeeksi nopea Raspberry Pi -käyttöjärjestelmän asemaksi (Brown 2023). Raspberry Pi 4:hin oli myös mahdollista kytkeä ulkoinen SSD-levy Raspberry Pi 4:n USB 3.0 -porttiin, tosin silloin piti myös huolehtia Raspberry Pi 4:n piirilevyllä olevan USB-ohjaimen aktiivisesta jäähdytyksestä (Geerling 2019).

Kojelautakameroille tarkoitettuihin muistikortteihin yleensä luvataan parempaa datan ylikirjoituksen kestämistä, joten sellaisen valinta Raspberry Pi 4:lle käytettäväksi oli selkeä. Raspberry Pi 4:ssä olevan muistikortinlukijan nopeus tosin ei aivan riittänyt projektin tarpeisiin. Varsinkin karttalohkojen piirtäminen ja reittien selvittäminen Suomen kattavasta OpenStreetMaps aineistosta aiheutti karttakuvien lataamisessa hitautta. Siirrettyäni kaikki käyttämäni Docker-kuvat ulkoiselle SSD-asemalle paransi se karttakuvien lukunopeutta riittävästi.

Adafruitin ESP32 Feather V2 -mikrokontrollerissa ei ollut muistikortinlukijaa, joten sen lisääminen oli tarpeellista. Adafruitin Feather-tuotteiden joukosta löytyi microSD-muistikortinlukija, jonka piirilevyllä oli myös paikka patterille ja ki-deoskillaaattori, jotka yhdessä mahdollistivat kellonajan säilyttämisen (Adafruit 2022). Tätä kellonaikaa tosin ei opinnäytetyössä käytetty, koska käytettävissä oli GPS-moduulista saatava aika.

ESP32-laitteessa muistikorttia käytettiin vain tekstimuotoisen datan tallentamiseen, joten myös hidas muistikortti toimi hyvin. Dataa Reittikartta-ohjelman backendissä tallentui noin 5MB 60 minuutin mittaisen datankeräyksen aikana, joten kirjoitusnopeudella ei ollut suurta merkitystä. Valitun muistikortin pidempiaikainen kestäminen tosin jäi vielä tuntemattomaksi. Datatallennus tapahtui aina uuteen tiedostoon, joten tallennuksen pitäisi tapahtua suurimmalta osalta eri kohtaan microSD-muistikortin muistia.

2.5.2 USB-Sarja-yhteys

Projektissa käytetty Adafruitin ESP32 Feather V2 -mikrokontrolleri tukee USB-yhteyden kautta toimivaa sarjaliikennettä, jonka kautta mikrokontrolleri ohjelmoidaan (Rembor 2022b). Samaa USB-yhteyden kautta toimivaa sarjaliikennettä oli myös mahdollista käyttää toiseen suuntaan. Tämän yhteyden avulla Reittikartta-ohjelmiston reaaliaikaisten toimintojen käyttäminen oli mahdollista myös tietokoneella.

Adafruitin ESP32 Feather V2:ssa USB-yhteyttä hoitaa CH9102F-mikropiiri (Rembor 2023a). CH9102F muuttaa mikrokontrollerin sarjaliikenteen USB-yhteyden kautta kulkevaksi (WCH 2023a). CH9102F välittämän sarjaliikenne datan käyttäminen vaatii kohde laitteelle virtuaalisen COM-portti ajurin (WCH 2023b). Ajurin luoman COM-portin avulla sitä tukevat verkkoselaimet pystyvät aukaisemaan yhteyden COM-portissa olevaan laitteeseen.

Chromiumiin perustuvien verkkoselaimien avulla sarjaväylän avaaminen on mahdollista kaikilla pöytätietokoneiden käyttöjärjestelmillä. Android-laitteilla väylän muodostuksen pitäisi onnistua Serial API polyfill -kirjaston avulla. Beauafortin mukaan Applen puhelimissa ja tableteissa sarjaväylän luominen USB:n kautta ei toimi, koska USB-portti ei ole vapaasti käytettävissä. (Beauafortin 2020.) Opinnäytetyössä keskityin tukemaan vain Chromium-pohjaisia verkkoselaimia pöytätietokoneiden käyttöjärjestelmissä. Tosin tallenteen lukemisen pitäisi toimia lähes kaikissa verkkoselaimissa lähes kaikilla laitteilla.

2.5.3 Bluetooth Low Energy

Adafruitin ESP32 Feather V2:ssa olevassa ESP32-suoritimessa oli tuki Bluetooth-yhteyksien muodostamiseksi (Rembor 2023a). Tuettuina olivat Bluetooth Classic ja Bluetooth Low Energy (Rembor 2023a). Opinnäytetyössä käytettävä Polarin H10 -sykeanturi tuki Bluetooth Low Energy ja Ant+ -yhteyksiä (Polar 2024).

Bluetooth Low Energy -yhteyksien muodostamiseen Raspberry Pillä tutustuin jo työharjoittelussa. Bluetooth Low Energy -yhteyden muodostamiseksi piti selvittää yhdistettävän laitteen ja sen dataa välittävän majakan UUID-tunniste, joka silloin oli suhteellisen yksinkertaista Python-kielisten Bluetooth-kirjaston avulla. Raspberry Pi -käyttöliittymä tosin aiheutti silloin ongelmia Bluetooth Low Energy laitteen löytymisen ja yhdistämisen kanssa, koska kyseessä ei ollut tavallinen human interface laite.

Espressif Systemsin (2023) Arduino IDE:lle tekemästä Core:sta löytyi tuntemattoman henkilön tekemä esimerkki lähdekoodi Bluetooth Low Energy Client -yhteyden muodostukseen. Esimerkki lähdekoodissa oli funktioiden kuvaukset, mutta itse esimerkki koodin toiminta silmukkaa ei ollut kuvattu mitenkään. Toiminta silmukan selvittämiseen kuluikin jonkin verran aikaa, mutta lopulta suoritus järjestys selkeni. Sen jälkeen lähdekoodin muokkaaminen oli mahdollista, jotta sitä pystyi hyödyntämään Arduino IDE:llä tehdyssä Reittikartta-ohjelmiston backendissä. Alkuperäisessä lähdekoodissa Bluetooth LE -yhteyden etsiminen käytti suorituksen pysäyttävää muotoa. Muuttamalla yhtä parametriä yhteyden hakemisen aloittavassa funktiossa sai sen muutettua ei pysäyttävään muotoon. Tämän jälkeen Bluetooth LE -yhteyden etsiminen, yhdistäminen, yhteyden menettäminen ja yhteyden uudelleen etsiminen toimi ilman suorituksen hidastumista.

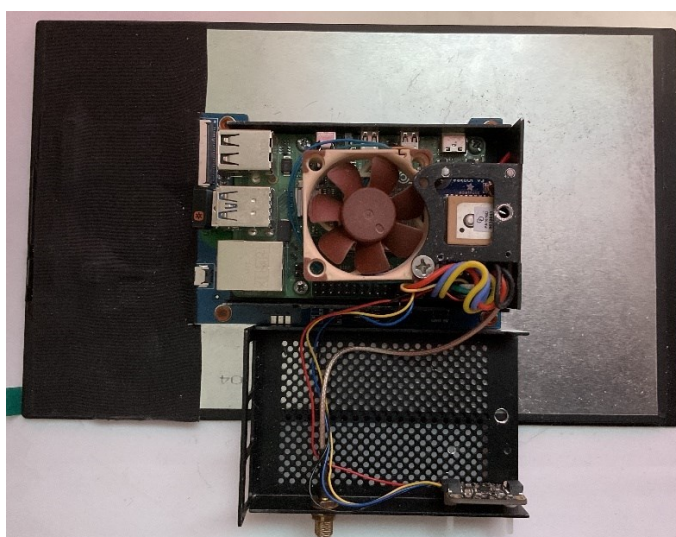
Käytetyn esimerkki lähdekoodin avulla oli mahdollista selvittää Polar H10:n Bluetooth Low Energy -palvelimen uniikki tunniste (UUID). Sykedataa välittävän majakkapalvelun UUID-tunnistetta ei esimerkki koodin avulla saanut selvitettyä. Bluetooth SIG julkaisee kuitenkin listan jokaisesta ominaisuudesta, jolle UUID-tunniste on myönnetty (Bluetooth SIG Inc. 2024). Listassa datatyyppien UUID-tunnisteet ovat 16-bittisinä lukuina. Lisäämällä Bluetoothin listasta saatu sykedatan 16-bittinen tunniste (0xFFFF) Polarin H10:stä saadun 128-bittisen UUID-tunnisteen vasemmaisesta oktettiin (0000XXXX-0000-0000-0000-000000000000) saadaan selville sykedataa välittävän majakkapalvelun UUID-tunniste (Community wiki 2016). Näiden kahden UUID-tunnisteen avulla oli

mahdollista kuunnella sykedataa Bluetooth Low Energy -yhteyden välityksellä Polar H10:stä.

3 Laitekokonaisuuksien kokoaminen

3.1 Raspberry Pi -laitekokonaisuus

Raspberry Pihin (kuva 1) liitettävät laitteet tarvitsivat kaksi 5 voltin pinniä sekä kaksi 3,3 voltin pinniä, joten kaikki Raspberryn Pissä olevat GPIO-virtapinnit tulivat käytetyiksi. GPS-moduuli käytti UART-sarjaväylän pinnejä (RX ja TX) ja IMU-anturin yhdistin Raspberry Pin SPI-sarjaväylän pinneihin. Raspberry Pi 4 Model B:hen liitettävien lisälaitteiden ja antureiden liittäminen tapahtui GPIO-pinneihin, joten se vaatii siihen tarkoitukseen sopivien johtojen tekemistä ja muokkaamista. Opinnäytetyössä käytetyn Raspberry Pille tarkoitetun kotelon sisällä oleva tila ei mahdollistanut valmiiden GPIO-pinneihin käyvien johtojen käyttämistä. Tilan puutteen takia käytettyjen johtojen suojavaipan piti olla silikonia, jotta ne sai taivuteltua mahdollisimman pieneen tilaan.



Kuva 1. Raspberry Pi (Kuva: Jari Keskisalo).

Raspberry Pi:tä varten hankin 5V:n virralla toimivan kosketusnäytön. Näyttö toimi Raspberry Pi:n 5V:n virtapinnistä tulevan virran avulla. Lisäksi se käytti Raspberry Pi:n DSI-liitintä ja I2C-sarjavyöhyksen pinnejä. Käyttämällä GPIO-pinnejä ja DSI-liitintä Raspberry Pi:n USB- ja HDMI-portit jäivät vapaaksi muuhun käyttöön. Näytössä olevaan virtaliittimeen sopivaa johtoa ei tullut näytön mukana, joten valmistin sen itse. Näytönvalmistajan verkkosivuilta en löytänyt kyseisen liittimen tyyppiä, joten tiedustelin sitä näytönvalmistajan tarjoamasta tuesta. Tuki kutsui liittintä joksikin todella harvinaiseksi, mutta se osoittautui lopulta 4-pinnisen JST-liittimen klooniksi.

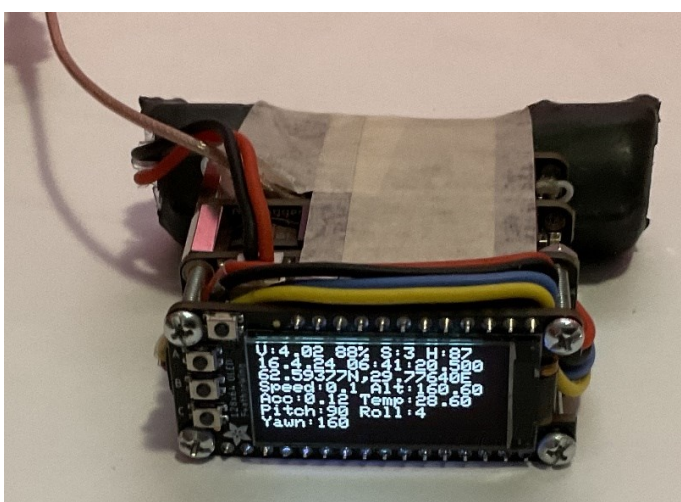
Raspberry Pi:n jäähdytyksessä käytettävän tuulettimen johdon muokkasin GPIO-pinneihin sopivaksi, jotta sen pystyin liittämään Raspberry Pi:n GPIO-virtapinneihin. Käytetyn Noctua 5V PWM -tuulettimen pyörimisnopeutta olisi lisäksi voinut säädellä Raspberry Pi:n PWM-signaalia tukevien GPIO-pinnien kautta (Pinout.xyz 2023). Kyseinen Noctuan tuuletin toimi hiljaisesti ilman sen pyörimisnopeuden säätelyä, joten sen ei ollut tarpeellista. Raspberry Pi -laitetekonaisuudessa käytetyn GPS-moduulin piirilevyyn jouduin juottamaan johdot kiinni tilan puutteen takia. IMU-moduulin QT-liittimeen sopivaa johtoa piti muokata, jotta sen pystyi liittämään Raspberry Pi:n GPIO-pinneihin.

Raspberry Pi:tä varten hankkimaani alumiiniseen koteloon porasin reikiä ja tein niihin kierteitä adapterin ja antureiden kiinnitystä varten. Tosin kotelon ohuessa rakenteessa kierteet eivät kestäneet ruuvien toistuvaa kiristämistä. Vaihdoin käyttämäni teräsruuvit nylonista tehtyihin ruuveihin kierteiden paremman keston toivossa. Lisäksi kotelon alkuperäinen kannen kiinnityksessä käytetty ruuvinkierre irtosi melkein heti, joten päädyin suurentamaan kierteenreikää ja asentamaan siihen kierteenkorjaussarjasta teräksisen helicoilin. Kierteenkorjauksen jälkeen kotelon kansi kesti hyvin ruuvien kiristämistä.

3.2 ESP32-laitetekonaisuus

Adafruitin ESP32 V2 Featheriin ((kuva 2) kiinnitin Feather-tuoteperheen musta-valkoisen OLED-näytön, microSD-muistikortinlukijan ja GPS-anturin toisiinsa

kasattavien pinnirivien avulla. Ensimmäisten pinnirivien juottamiseen ehdoton apuväline oli koekytkentälevy, jonka avulla pinnirivit sai juotettua suorakulmaan piirilevyyn nähden. Loput pinnirivit olivat sen jälkeen juotettavissa jo kiinnitettyjen pinnirivien avulla. IMU-moduuli oli mahdollista yhdistää valmiilla QT-johdolla näytössä olevaan liittimeen. Virtaa laitekokonaisuus sai kokoamastani Li-ion-paristosta. Virtajohdot juotin kiinni irrallisiin paristopidikkeisiin, jonka jälkeen suljin kaikki kuristeputken sisälle. Pienen JST-liittimen kiinnittäminen silikonisella suojavaipalla oleviin virtajohtoihin osoittautui varsin hankalaksi. Liittimen kiinnittäminen vaati useamman yrityksen ennen kuin johdot pysyivät tukevasti kiinni liittimen pinneissä.



Kuva 2. ESP32-laite (Kuva: Jari Keskisalo).

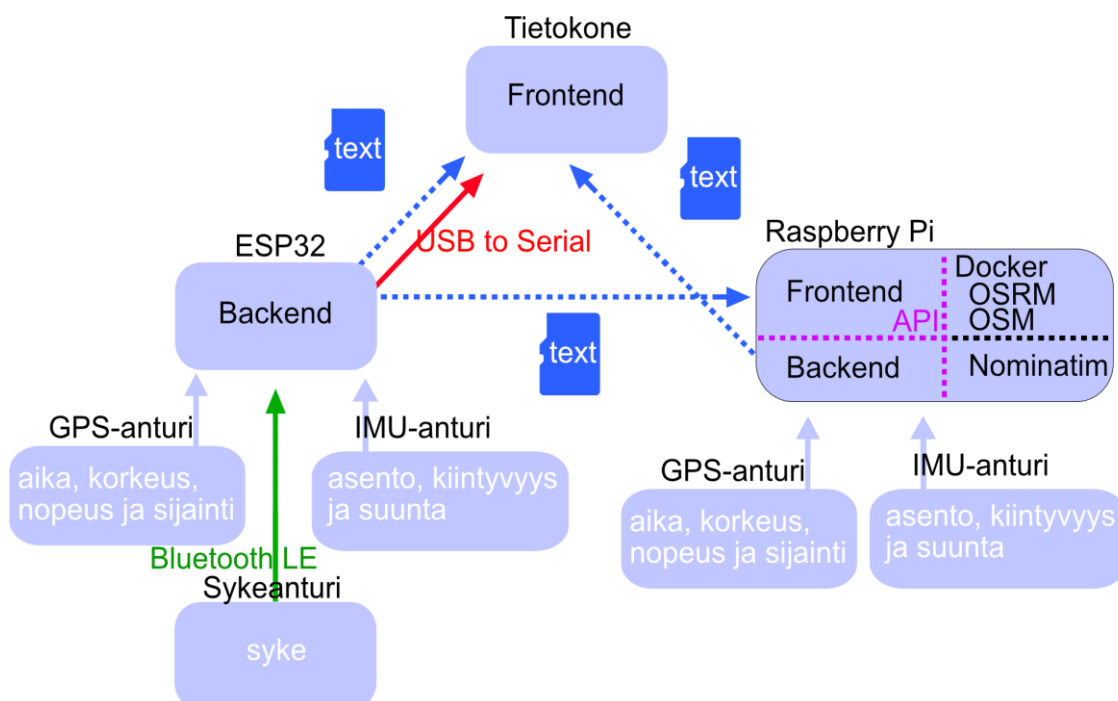
Adafruit Ultimate GPS Featherwingin piirilevyllä oleva antenniliitin (pieni kierre-jousi) ei kestänyt antenniadapterin irrottamista ja liittämistä, joten se hajosi muutamien toistojen jälkeen. Juotoskolvilla antenniliittimestä jäljelle jääneet osat sai irrotettua helposti. Katkaistun antenniadapterin johdon juottaminen piirilevyllä olleisiin pieniin juotoskohtiin oli jo hiukan vaikeampaa. Myös GPS-antenninjohdossa oleva SMA-urosliitin irtosi opinnäytetyön aikana. Kyseinen SMA-liitin oli puristettavaa mallia lukuun ottamatta keskipinniä. Liittimen takaisin kiinnittäminen vaati ohutta poranterää, millä keskipinnistä sai poistettua jäljelle jääneen juotostinan. Juotostinan poistamisen jälkeen johdossa olevan sisäjohtimen kiinni juottaminen oli mahdollista. Johdossa oleva ulkojohdin piti juottaa liittimen ulkokuoreen kiinni, koska kiinni puristettua liittimen osaa ei voinut aukaista rikkomatta liittintä.

Kasattavien pinnirivien avulla toisiinsa kiinnitetyt Feather-piirilevyt liikkuivat vielä jonkin verran, koska pinnirivit olivat eri levyisiä. Liikkumisen estämiseksi liitin piirilevyt toisiinsa vielä korotepaloilla kolmesta eri nurkasta. Ongelmia korotepaloilla kiinnittämiseen aiheutti kiinnitykseen käytetyt pinnirivit ja piirilevyillä olevat komponentit, jotka olivat liian lähellä piirilevyjen nurkissa olevia reikiä. Tästä syystä jouduin vielä hiomaan korokepaloja ohuemmiksi, jotta ne sai kiinnitettyä paikoilleen.

4 Toteuttaminen

4.1 Ohjelmiston toimintaympäristö

Reittikartta-ohjelmiston toimintaympäristö koostui useasta (kuvio 1) eri osasta. Verkkoselaimessa toimiva frontend toimi Raspberry Pillä ja tietokoneella. Tietokoneella frontend hyödynsi virtuaalisen COM-portin (USB to Serial) kautta tulevaa dataa. Raspberry Pillä frontend käytti sekä backendin rajapintaa (API), että Docker-kuvissa olevia OpenStreetMapsien tile- (OSM) ja OSRM-palvelimen (OSRM) rajapintoja. ESP32:ssa backend keräsi dataa myös Bluetooth LE -yhteyden välityksellä Polarin H10-sykeanturista (sykeanturi). GPS- ja IMU-moduulit molemmissa laitteissa (ESP32 ja Raspberry Pi) olivat samat, mutta niitä käytettiin eri ohjelmointikielillä toteutettujen laitekirjastojen avulla.



Kuvio 1. Ohjelmiston toimintaympäristö.

Raspberry Pillä backend toimi myös välityspalvelimena frontendin ja Nominatimin Docker-kuvassa olevan rajapinnan välillä. Nominatimin Docker-kuvassa ollut web-palvelin ei käyttänyt HTTP orgin -otsikon määrittelyä, joten sen käyttäminen vaati välityspalvelimen käyttämistä. Toisiin opinnäytetyössä käytettyihin Docker-kuviin sai muodostettua yhteyden ilman välityspalvelimen käyttöä.

Molemmilla laitteilla (ESP32 ja Raspberry Pi) backendit tallensivat datan microSD-muistikortille tekstitiedostona. Tiedostot nimettiin GPS-moduulista saatavan ajan perusteella. Molemmat frontendit pystyivät lukemaan molempien backendien tallentamaa dataa.

4.2 Käyttöliittymän ja skriptipuolen toteuttaminen

Frontendin toteutin HTML-, CSS- ja JavaScript-kielten avulla. Työharjoittelussa tekemäni HTML-sivu toimi pohjana frontendin toteutuksessa tässä opinnäytetyössä. Tähän pohjaan tein joitakin muutoksia kehityksen aikana. Käytössä oleva CSS-tyylimäärittely oli myös pääosin työharjoittelun aikana tekemäni. CSS-tyylimäärittelyssä oli otettu huomioon eri laitteiden ruutukoot. CSS-tyylimäärittelyssä muutin eri elementtien piirtämisjärjestystä, jotta sain

karttamerkkien käsittelyssä käytettävät ponnahdusikkunat näkyviin toisten verkkosivulla olevien elementtien päälle. Sen lisäksi ryhmittelin käyttöliittymään lisäämiäni painikkeita ja tekstimuotoista dataa sisältäviä elementtejä CSS-tyylimäärittelyn ruudukon avulla.

JavaScriptin ja jQuery:n avulla karttaohjelmassa luettiin dataa tekstitiedostosta ja ohjelmiston backendeistä. Backendeistä tuleva data piti aluksi jäsenellä, jotta jatkokäsittelyyn välittyi vain kokonaisia data-elementtejä. Jäsennelly data jaettiin osiin ja sen jälkeen osaa niistä käytettiin laskelmissa. Lopulta data esitettiin verkkosivulla Chart.JS:n ja Leafletin avulla. JavaScriptin ja jQueryUI:n avulla toteutin Leafletin karttamerkkien lisäämisen ja muokkaamisessa käytettävän käyttöliittymän. Kartassa ja taulukossa näkyvää data oli mahdollista toistaa automaattisesti. Dataa pystyi myös selaamaan manuaalisesti käyttöliittymässä olevan vierintäpalkin avulla. Varsinkin jo esitetyn ajan hetken uudelleen näyttäminen lähes reaaliajassa osoittautui projektia tehdessä haasteelliseksi. Ajassa eteenpäin mennessä vastaavaa ongelmaa ei ollut koska näytettävää dataa oli mahdollista lisätä käyttöliittymään pieninä osina. Taaksepäin ajassa siirtyessä käyttöliittymässä näkyvän datan uudelleen laskeminen osoittautui raskaaksi toimenpiteeksi, joten sen optimointi oli tarpeellista.

Reittikartta-ohjelmiston frontendin toteuttamisessa käytin avoimen lähdekoodin JavaScript-kirjastoja, joista suurin osa oli käytettävissä Internetissä olevilta palvelimilta. Raspberry Pillä toimivan frontendin piti myös toimia ilman internet-yhteyttä, joten käytettyjen aineistojen, ohjelmien ja JavaScript-kirjastojen asentaminen niiden ohjeiden mukaisesti oli myös tarpeellista. Suurin osa JavaScript-kirjastojen asentamisesta tapahtui npm-pakettivaraston ja Node.js:n eli JavaScript-suoritusympäristön avulla.

4.3 Rajapinnan ja datankeräyksen toteuttaminen

Raspberry Pillä datan keräyksen hoiti kehittämäni Python-moduuli, jossa käytettiin Adafruitin Circuitpython-laitekirjastoja. Kompassisuunnan laskemisessa käytetyt metodit käyttivät SparkFun Electronicsin laitekirjastoa. Kompassisuunnan

laskuissa käytetyt metodit olivat peräisin Remingtonin Arduinolle tekemästä ICM_20948-AHRS-ohjelmakirjastosta, jonka käänsin Python-ohjelmointikielelle.

Ohjelmiston käynnistäminen tapahtui Shell-skriptien avulla. Ensimmäinen skripteistä käynnisti ensiksi Docker-palvelun, minkä jälkeen se suoritti Docker-kuvien käynnistuksen Python-skriptin avulla. Docker-kuvien käynnistämässä käytetyn Python-skriptin toteutuksessa käytin Docker Enginen Python-kirjastoa. Toinen Shell-skripteistä suoritti web-palvelimen Python-skriptin, joka samalla aloitti datankeruun aliprosessissaan. Kolmas Shell-skripteistä avasi Chromium-verkkoselaimen web-palvelimen käyttämään IP-osoitteeseen kiosk-tilassa.

Datankeräys aliprosessissa ei alkanut ennen kuin GPS-moduuli oli saanut signaalin vähintään neljältä paikannussatelliitilta. Tallennuksen alussa microSD-kortille avattiin uusi tiedosto, johon dataa kirjoitettiin yksi datankeräyskierros kerrallaan. Samanaikaisesti datankeräyksen aliprosessin aliprosessissa alkoi kompassisuunnan laskeminen. Data eri prosessien välillä liikkui moniajota varten tehtyjen jonojen avulla. Datankeräys aloitettiin ja lopetettiin rajapinnan kautta asetettavalla prosessien välisellä Python-eventillä. Datankeräämisen loppuessa datankeräämiseen ja tallentamiseen käytetyn silmukan osion suorittaminen loppui ja samalla sulkeutui tallentamiseen käytetty tiedosto. Aliprosessin silmukan suorittaminen jatkui ilman datan keräämistä aina web-palvelimen Python-skriptin lopettamiseen asti, jotta datankeräämisen uudelleen aloittaminen oli mahdollista ilman koko ohjelmiston uudelleen käynnistämistä.

Flashin avulla toteutettu web-palvelin käynnisti datankeräysmoduulin aliprosessina palvelimen käynnistuksen yhteydessä. Palvelimen HTTP-rajapinnan kautta aliprosessissa oleva datankeräys oli mahdollista aloittaa ja lopettaa GET-pyyntöillä. Palvelin palautti GET-pyyntöillä prosessien väliseen jonoon viimeisenä lisätyn arvon. Palvelimeen lähetetyllä POST-pyyntöillä oli mahdollista tallentaa kartassa olevat karttamerkit niille varattuun tiedostoon. Näiden toimintojen lisäksi web-palvelimeen täytyi toteuttaa välityspalvelin toiminto Nominatim-ohjelman ja Frontending välille. Missä frontendiltä tuleva GET-pyyntö välitettiin Nominatimin Docker-kuvassa olleelle web-palvelimelle. Docker-kuvassa olevan

web-palvelimen palauttaman datan otsikkoon lisättiin origin-määrittely ja sen jälkeen se palautettiin Reittikartta-ohjelmiston frontendiin.

Raspberry Pillä IMU-moduulissa olevien antureiden kalibrointi oli varsin yksinkertaista toteuttaa Remingtonin GitHub-sivulla olevan Python-skriptin ansiosta. Skripti laski kaikkien IMU-moduulissa olleiden antureiden kalibrointi arvot tiedostossa olevista tallennetuista arvoista. Ainoastaan epäselväksi jäi miten kalibroinnin osana oleva laitteen pyörittäminen keskipisteensä ympäri käynnistyksen yhteydessä onnistuisi, jos laite olisi kiinnitettynä ajoneuvoon. Tosin varsin pienellä määrällä arvoja oli mahdollista saada kohtalaisesti toimivat kalibrointi arvot.

Datankeräyksen suorittava ohjelma ESP32:lle oli toteutettu Arduino IDE:llä, jossa käytettiin Adafruitin ja SparkFun Electronicsin julkaisemia laitekirjastoja. Laitekirjastojen lisäksi ESP32:lle toteuttamassani datankeruu ohjelmassa käytettiin Espressif Systemsin Arduino Coressa olevaa avoimen lähdekoodin Bluetooth-ohjelmakirjastoa sekä kompassisuunnan selvittämisessä Remingtonin ICM_20948-AHRS-ohjelmakirjastoa.

ESP32-laitteen käynnistyksen yhteydessä ohjelma tarkisti, oliko kalibrointi-arvot sisältävä tiedosto olemassa microSD-muistikortilla. Mikäli tiedosto oli jo olemassa, se luettiin ja luetut kalibrointi-arvot otettiin käyttöön. Mikäli tiedostoa ei ollut olemassa silloin käynnistyksen yhteydessä suoritettiin magnetometrin Hard Iron -kalibrointi. Hard Iron -kalibroinnissa mitattiin anturin antamia arvoja 10 sekuntia, jonka jälkeen laskettiin jokaisen akselin osalta suurimman ja pienimmän mitatun arvon keskiarvo. Laskuista saadut kalibrointi-arvot otettiin käyttöön ohjelmassa ja samalla ne tallennettiin tiedostoon. Kalibroinnin käyttöönoton jälkeen datan kerääminen ja tallentaminen alkoi automaattisesti, kun GPS-moduuli sai signaalin neljältä paikannussatelliitilta. Tallentamisen pystyi lopettamaan painamalla ESP32-laitteen näytössä olevaa A-nappia. A-napin painaminen poisti myös tallennetun kalibrointitiedoston, jolloin kalibrointi piti uusia uudelleenkäynnistyksen yhteydessä. Datatallentaminen alkoi automaattisesti uudestaan 10 sekunnin kuluttua, mikäli laitteesta ei katkaistu virtaa tai poistettu microSD-muistikorttia.

Hard Iron -kalibrointiarvojen avulla ohjelmassa korjattiin mittaustuloksien suuret epätasapainot ja mahdollisten magneettisten esineiden aiheuttamat vääristymät magnetometrin mittaus arvoissa. Keskiarvon laskeminen jokaiselle akselille oli ilmeisesti yksinkertaisin tapa laskea Hard Iron -kalibrointiarvot. Hyödyntämällä valmiita Python-kirjastoja ja matriisilaskuja olisi ollut mahdollista saada tarkemmat kalibrointiarvot. Vastaavien laskukaavojen toteuttaminen ESP32:lla ja Arduino IDE:n avulla vaikutti olevan hankalaa ja paljon aikaa vievää. Tarkemmat kalibrointi arvot piti laskea tietokoneella ajoittain, jolloin myös visuaalisen avun käyttäminen kalibroinnissa oli mahdollista. Tätä varten muokkasinkin Remingtonin GitHub-sivuilla olevaa kalibrointidatan keräyksessä käytettävää Arduino-sketchiä helppokäyttöisemmäksi. Soft Iron -kalibroinnin laskemat arvot muuttivat antureiden mittaamia arvoja vain vähän, joten niiden muuttuminen eri käynnistyskertojen välillä ei aiheuttanut kompassisuunnan selvittämisessä ongelmia. Hard Iron -kalibroinnin antamat arvot taas saattoivat muuttaa mitattuja arvoja hyvinkin paljon, mikä pahimmillaan aiheutti kompassinsuunnan asteikon pienemisen alle puoleen.

ESP32-laitteen näytössä olevien tietojen avulla varmistin ohjelman koodin ja laskukaavojen toimiminen oletetusti. Satelliittiyhteyksien määrä ja paikannuskoordinaatit ilmaisivat tallennuksen käynnistymistä. Käytetyn akun varaustason näyttäminen tuli tarpeelliseksi, kun testatessa ilmeni eri antureiden toimintaongelmat ennen koko laitteen sammumista vähäisen jännitteen takia. Opinnäytetyön kehityksen aikana tehdyissä testeissä laitteen näyttö ei ollut näkyvissä, joten muiden tietojen lisääminen ei vielä ollut tarpeellista.

ESP32-laitteella kerätyn datan tallennus aloitettiin aina uuteen tiedostoon, joka nimettiin GPS-ajan mukaan. Dataa kerättiin ja tallennettiin 500ms välein toistuvassa suoritus silmukassa. Kompassisuunnan laskeminen tapahtui nopeamassa tahdissa, jotta kompassisuunnasta sai laskettua keskiarvon edellisen arvon kanssa. Samalla vähentyi kompassisuunnan laskemisen aiheuttamat viivästykset muun koodin suorittamisessa. Antureiden keräämä data oli myös mahdollista lähettää sarjadatana laitteen USB-porttiin painamalla näytössä

olevaa C-painiketta. B-painiketta painamalla oli mahdollista lisätä karttamerkki sen hetkiseen kohtaan.

Koska ESP32-laitetta kuljetettiin mukana harjoituksia tehdessä, sen virran saannissa saattoi tapahtua katkoksia. Näiden virtakatkoksien takia datankeräys antureilta alkoi automaattisesti, kalibrointidata ladattiin tiedostosta ja tiedostoa tallennettiin yksi rivi kerrallaan tekstimuotoisena. Virtakatkokset johtuivat pääasiassa 18650-paristosta kokoamastani akusta. Akun päissä olevilla irrallisilla paristopidikkeillä oli heikko kontakti paristoon pelkän kuristeputken avulla kiinnitetynä, vaikka paristopidikkeiden muoto kiinnitti ne tiukasti kuristeputkeen.

Antureista kerättiin lähes kaikki data molemmilla laitteilla mitä Adafruitin laitekirjastojen avulla oli mahdollista saada. Kaikki arvot eivät olleet saatavilla molemmissa käytetyissä ohjelmointi ympäristöissä. Sparkfun Electronicsin samalle IMU-moduulille tekemän laitekirjaston avulla puuttuvat arvot olisi ollut mahdollista saada toteutetun kompassisuunnan laskemisen yhteydessä. Sparkfun Electronicsin laitekirjaston palauttamat arvot olivat konvertoimattomassa muodossa, joten niitä olisi pitänyt muuttaa laskukaavoilla ennen niiden käyttämistä ohjelmassa. Lasketut arvot eivät vastanneet Adafruitin laitekirjaston avulla saatuja arvoja muunnosten jälkeen.

GPS-moduulin laitekirjaston palauttamasta datasta saattoi puuttua aika ja paikannustiedot, tai pelkästään toinen paikannustiedoista. Laitekirjasto palautti myös joskus saman arvon molemmissa paikannusarvoissa. ICM-moduulin osalta datan saamisen kanssa ei kehityksen aikana esiintynyt ongelmia. Mikäli kerätyistä datasta puuttui arvo tai se oli selvästi väärä, korvattiin se edellisellä hyväksi todetulla arvolla frontendissä. Datankeräyksen alussa esiintyviä datansaanti ongelmia estettiin myöhemmällä datankeräyksen aloittamisella.

Ohjelmassa käytetty datanrakenne noudatti kuvassa olevaa formaattia (kuva 3) ilman rivin vaihtoja. Cycle-elementti lapsi elementteineen toistui data-elementin sisässä tallennetussa datassa. GPS-moduulista (<gps>) kerättiin ajan (<time>) ja paikannus tietojen (<lat><lon>) lisäksi, nopeus (<gpsspeed>), satelliitin kulma (<angle>), antennin korkeus (<altitude>) ja horisontaalinen tarkkuus

(<accuracy>). Aika oli gps-elementin ulkopuolella, koska GPS-moduuli saattoi palauttaa ajan jo yhdestä satelliittisignaalista, kun neljä satelliittia vaadittiin sijaintien ja muiden arvojen saamiseen. Aika oli UTC-ajassa, joka muutettiin frontendissä JavaScriptin avulla paikalliseen aikavyöhykkeeseen. IMU-moduulin kiihtyvyysanturista (<accelerometer>) tallennettiin akselien arvot, mutta myös kymmenen valmiiksi laskettua kiihtyvyyttä (<a1>...<a10>), jotka oli tallennettu 50ms:n välein. Tarkoituksena valmiiksi lasketuilla kiihtyvyyksillä oli saada selville tekemieni kävelyharjoitusten tarkempi askelmien määrä askelmien pituuden laskemiseksi. Kulmakiihtyvyysanturista (<gyro>), sekä magnetismia mittaavasta anturista (<magnetometer>) tallennettiin mitattujen akselien antamat arvot. Suunta (<bearing>), kulma (<pitch>) ja kallistus (<roll>) saadaan Remingtonin kompassisuunnan laskevan ohjelmistokirjaston avulla. Syke-anturi lähetti sydämen sykkeen sekunnin välein (<heartrate>).

```
<data>
<cycle>
<time>2024-04-15 04:11:10:0</time>
<gps>
<lat>62.595878601</lat><lon>29.777395248</lon><gpsspeed>4.50</gpsspeed><angle>252.57</angle>
<altitude>111.00</altitude><accuracy>1.18</accuracy>
</gps>
<accelerometer><x>-0.17</x><y>-9.83</y><z>0.09</z>
<a1>0.31</a1><a2>-4.63</a2><a3>- .13</a3><a4>5.53</a4><a5>1.54</a5>
<a6>1.82</a6><a7>4.19</a7><a8>10.17</a8><a9>- .94</a9><a10>3.78</a10>
</accelerometer>
<magnetometer><x>0.00</x><y>0.00</y><z>0.00</z></magnetometer>
<gyro><x>-1.72</x><y>1.10</y><z>1.11</z></gyro>
<bearing>216.00</bearing>
<pitch>-11.00</pitch>
<roll>3.00</roll>
<heartrate>159</heartrate>
</cycle>
</data>
```

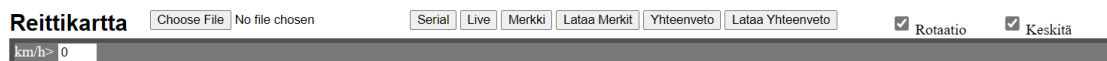
Kuva 3. Datanrakenne (Kuva: Jari Keskisalo).

5 Toteutetut toiminnot

5.1 Käyttöliittymä

Reittikartta-ohjelmiston käyttöliittymässä (kuva 4) oli kaksi toisistaan eroavaa toimintatilaa: tallenteen lukeminen- ja reaaliaikainen-tila. Tallenteen lukeminen -tilassa (Choose File-painike) avattiin käyttäjän valitsemana tiedosto, jonka

sisällöstä käyttöliittymässä esitettävä data luettiin. Reaaliaikaisesta-tilasta oli kaksi toisistaan hiukan eroavaa versiota. Raspberry Pillä Live-tila (Live-painike) sai käytettävän datan Raspberry Pillä toimivan rajapinnan kautta. Tietokoneella Serial-tila (Serial-painike) aukaisi ensiksi COM-portin valinnan, jonka jälkeen virtuaaliseen COM-porttiin liitetystä ESP32-laitteesta tuleva data käsiteltiin lähes reaaliajassa.



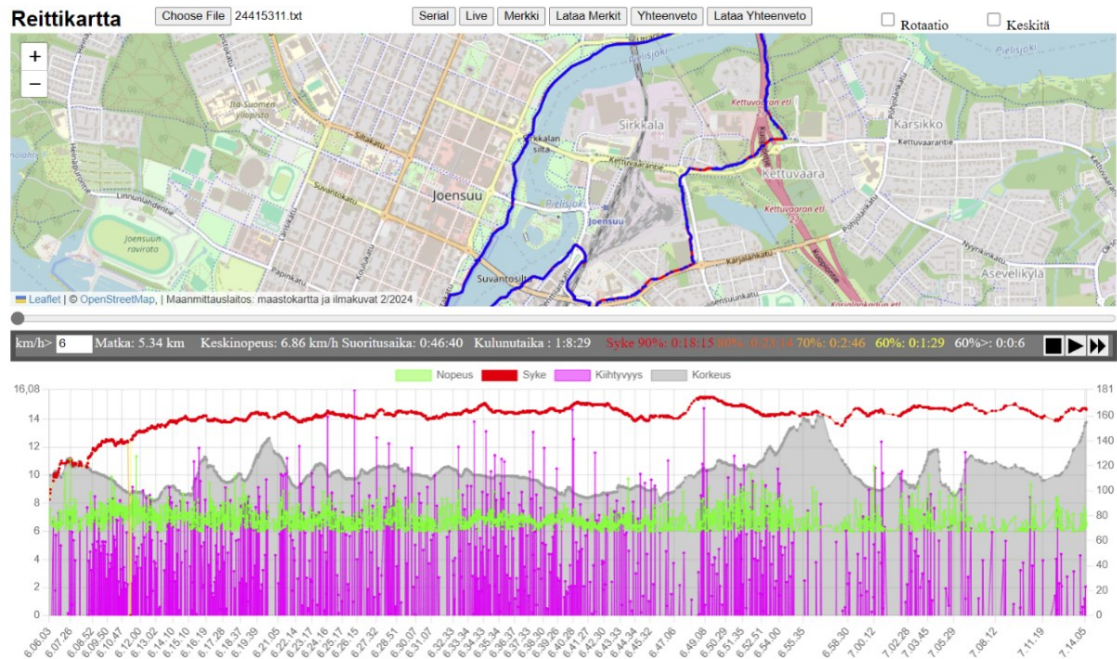
Kuva 4. Aloitusnäkö (Kuva: Jari Keskisalo).

Käyttöliittymässä olevaa Merkit-painiketta painamalla aukesi karttamerkkien lisäämisessä käytetty käyttöliittymä ponnahdus ikkunassa. Lataa Merkit-painikkeesta kartassa olevat karttamerkit oli mahdollista ladata JavaScript-tiedostona. Yhteenvedo-painiketta painamalla käyttöliittymässä näytettiin tallennettu data kokonaisuutena. Kokonaisesta datasta tehtyjä laskelmia pystyi lataamaan CSV-tiedostona Lataa Yhteenvedo -painiketta painamalla. Kyseisen CSV-tiedoston sisältämä data oli sen jälkeen mahdollista hyödyntää taulukkolaskentaohjelmassa. Rotaatio-valintalaatikon avulla kartan pyörimisen käyttämistä pystyi hallitsemaan. Kartan automaattinen keskittäminen sen hetkiseen sijaintiin oli hallittavissa Keskitä-valintalaatikon avulla.

Käyttöliittymässä olevan input-tekstikentän avulla (km/h>) oli mahdollista määrittellä nopeuden alaraja-arvo. Nopeuden alaraja-arvo rajasi nopeudesta ja kulu-neesta ajasta (datankeräyksessä käytetyn syklin pituus) laskettavien matkan, keskinopeuden ja suorituksenkeston ajan. Nopeuden alaraja-arvo määritteli myös mikä aika otettiin huomioon käyttöliittymässä esitettyjen sykealueiden aikojen mittauksessa. Mikäli data näytettiin yhteenvedona, silloin nopeuden alaraja-arvo määritteli myös mitä/miten dataa näytettiin käyttöliittymän kartassa ja kaaviossa ja mitä dataa yhteenvedon latauksessa oli mukana. Jos alaraja-arvoa muutettiin, päivittyi silloin kaikki käyttöliittymässä näytetyt arvot sen mukaan.

Painikkeiden alapuolella näkyvässä kartassa (kuva 5) näytettiin OpenStreetMapin kartta Suomesta ja joissakin sijainneissa Maanmittauslaitoksen rasteroituja maastokarttoja ja ilmakuvia. Karttakuvien päälle piirrettiin sen hetkinen sijainti ja

sitä edeltävät sijainnit karttaohjelman toimintatilan mukaan. Lisäksi kartassa näkyi lisätyt karttamerkit ja Maanmittauslaitoksen henkilökohtaisesta palvelusta saatavia tonttien ja metsäpalstojen rajamerkkejä ja niiden tiedot. Rajamerkkien rajaama alue oli merkitty viivoilla silloin kun raja oli suora, eli ei silloin kun tie toimi rajana.



Kuva 5. Yhteenvedo (Kuva: Jari Keskisalo).

Tekstimuotoisen datan esityksessä käytetyn palkin alapuolella olevassa kaaviossa esitettiin nopeus, korkeus, syke ja kiihtyvyys. Kaavion korkeus akselit kuvasivat vasemmalla puolella alempia mitattuja arvoja ja oikealla puolella ylempiä arvoja, riippuen siitä mitkä arvot olivat datassa suurempia. Kaaviossa esitetävä sykedatan viiva vaihtoi väriä käyttöliittymässä olleiden teksti muotoisten sykealueiden värien mukaan. Jatkossa kaaviossa on tarkoituksena esittää vielä mitatut kallistukset (pitch ja roll), jolloin mitattujen harjoitusreittien korkeus erot tulisivat paremmin esiin hitaissa nopeuksissa. Jalkaisin liikkuesssa kallistuskulmista ei vielä ollut hyötyä.

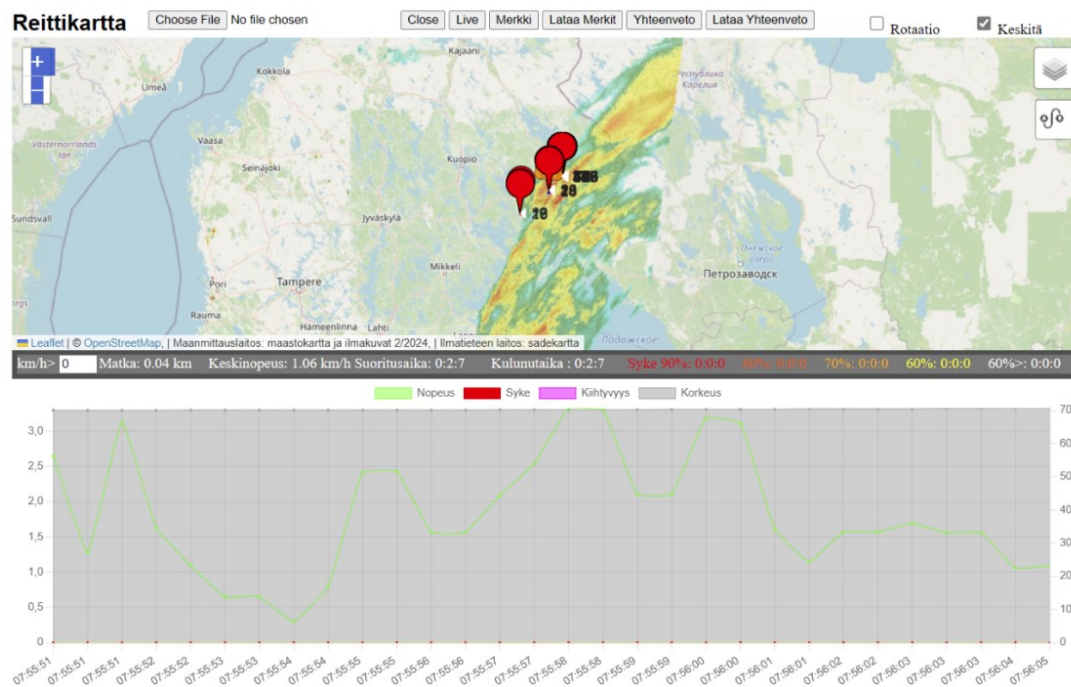
5.2 Tallenne-toimintatila

Tallenne-toimintatilassa oli mahdollista tarkkailla koko reitin pituudelta tallennettua dataa automaattisen toiston avulla, jonka nopeutta oli mahdollista säätää kartan oikeassa alanurkassa olevien painikkeiden avulla. Käyttöliittymän kartalle tuli näkyviin yhtenäinen kuljettu reitti (punainen viiva) samaan tahtiin datan käsittelyn kanssa. Punaisen viivan päälle piirrettiin sininen viiva, mikäli sen hetkinen nopeus ylitti käyttäjän asettaman alaraja-arvon. Kartassa esitettävän sinisen viivan pituus pystyi kehityksen alussa olemaan kokoreitin pituinen, mutta sitä piti lyhentää ohjelman suorituskyvyn takia. Samanaikaisesti käyttöliittymässä olevaan viivakaavioon piirrettiin antureiden tallentamat arvot viimeisen 15 sekunnin ajalta. Kartan alapuolella oleva vierityspalkin koko vastasi koko reitin pituutta, ja siinä oleva valitsin liikkui samaan tahtiin datan näyttämisen kanssa.

Mikäli kartan alapuolella olevaa vierityspalkkia liikutti, päivittyi kartalla oleva kuljettureitti ja kaavio valittuun hetkeen asti. Vierityspalkkia ei voinut siirtää viiden toista sekunnin sisään alusta. Tallennuksen toistamisen pystyi lopettamaan stop-painikkeesta ja käynnistämään uudelleen play-painikkeesta. Tallennuksen toisto nopeutta pystyi myös kasvattamaan fastforward-painikkeesta. Tallenteen toisto loppui automaattisesti, kun avatun tiedoston data oli esitetty kokonaisuudessaan.

5.3 Live- ja Serial-toimintatilat

Live- ja Serial-toimintatilat olivat toiminnoiltaan lähes samanlaisia tallenne-toimintatilan kanssa. Näissä tiloissa ei ollut tallenteen katsomiseen liittyviä toimintoja. Näissä toimintatiloissa karttaan oli mahdollista lisätä reaaliaikaista dataa (kuva 6). Reaaliaikaisen datan lähteenä toimi Ilmatieteen laitoksen avoimen datan rajapinnasta saatu sadealueiden kuva, mikäli internet-yhteys oli käytettävissä. Sade -aluetta esittävä kuva päivittyi viiden minuutin välein.



Kuva 6. Serial-toimintatila (Kuva: Jari Keskisalo).

Kartassa olevasta toisesta kuvakkeesta oli mahdollista aukaista reitinsuunnittelu. Reitinsuunnittelussa sen hetkinen sijainti oli asetettuna lähtöpisteeksi. Kohteen hakeminen vaati kohde osoitteen/kadun kirjoittamista kokonaisuudessaan, ennen kuin reitinhaku palautti mahdolliset vaihtoehdot. Lähtö-, kohde- ja välipisteitä oli mahdollista lisätä/muuttaa raahaamalla kartassa näkyvää reittiä. Reitinsuunnittelu tässä opinnäytetyössä palautti vain autolla kuljettavat reitit. Reitinsuunnittelun käyttöliittymässä olevat osoitekentät (lähtöpiste, välipisteet ja kohdepiste) hävisivät näkyvistä, kun mitattu nopeus kasvoi yli ennalta määritellyn arvon. Reitinhaun avulla määritelty reitti ja kulkuohjeet pysyivät näkyvissä ja päivittyi sitä mukaa kun sijainti kartalla muuttui, kunnes reitinsuunnittelun sulki sen kuvaketta klikkaamalla.

5.4 Karttamerkit

Käyttöliittymässä olevaa Merkki-painiketta painamalla oli mahdollista lisätä karttamerkki sen hetkiseen sijaintiin. Painikkeen painaminen aukaisi ponnahdusikkunan (kuva 7), jossa karttamerkin kuvakkeen pystyi valitsemaan tekemistäni vektorikuvista. Karttamerkille oli mahdollista kirjoittaa tekstimuotoinen nimi ja kuvaus. Ponnahdusikkunassa näkyi myös sijainnin tarkkuus, joka saatiin GPS-

moduulilta saadusta arvosta. Tallenna-painike tallensi merkin karttaan ja välitti sen ohjelmiston rajapinnalle tallennettavaksi. Palaa-painike sulki ponnahtusikkunan tallentamatta.



 Lakka ▼

Nimi:

Kuvaus:

Tarkkuus:

Kuva 7. Merkin lisäämisessä käytetty ponnahtusikkuna (Kuva: Jari Keski-salo).

Kartassa olevaa karttamerkkiä painamalla avautui karttamerkin muokkauksessa käytetty ponnahtusikkuna. Ponnahtusikkuna oli muuten sama lisäyksessä käytetyn ikkunan kanssa siihen lisättyä Poista-painiketta lukuun ottamatta. Poista-painikkeen avulla karttamerkin pystyi poistamaan kartasta ja tallennetuista merkeistä rajapinnan avulla. Karttamerkin kuvake ja tekstimuotoiset kuvaukset oli mahdollista muuttaa ja tehdyt muutokset oli mahdollista tallentaa. Käyttöliitymässä oli myös Lataa merkit -painike. Lataa merkit -painikkeen avulla karttamerkit ja niihin tehdyt muutokset oli mahdollista tallentaa ilman yhteyttä rajapintaan. Mikäli käsiteltävässä datassa oli karttamerkin tiedot, tarkistettiin kyseisen karttamerkin sijainti. Jos sijaintia ei löytynyt olemassa olevista karttamerkeistä, silloin karttamerkki lisättiin karttaan ja tallennettiin rajapinnan avulla.

6 Pohdinta

6.1 Menetelmälliset valinnat

Laitteiden hankinnassa etukäteen tehtävä selvitystyötä pidin erityisen tärkeänä. Laitteiden ominaisuuksien yhteen sovittaminen asetettujen tavoitteiden kanssa jo ennen niiden hankintaa mahdollisti tavoitteiden saavuttamisen ja vähensi odottamattomien viivytysten riskiä. Tosin opinnäytetyössä käytetyssä IMU-moduulissa olevien antureiden akselien poikkeukselliset suunnat jäivät silti huomaamatta laitevalintaa tehdessä. Tällainen mitättömältä tuntuva tieto aiheutti projektin aikataulun venymistä viikoilla. Laitteiden hankintaan varasin riittävästi aikaa, koska tiesin toimitusaikojen olevan epävarmoja, eikä komponenttien toimivuudesta ollut takeita. Osa tilatuista laitteista/komponenteista olikin jo epäkunnossa saapuessaan ja osa meni epäkuntoon vähäisen käytön jälkeen. Jos vain mahdollista, kannattaisi hankkia tarvittavia komponentteja kaksinkertainen määrä.

Opinnäytetyössä käytettävien laitteiden kokoaminen vaati alkeellisia juottamistaitoja ja välineitä antureiden ja laitteiden yhdistämiseksi. Laitteiden kasaamisen lisäksi oikean pituisten ja oikeanlaisilla liittimillä varustettujen johtojen valmistaminen oli myös tarpeellista. Johtojen liittimien kiinnittämisen helpottamiseksi siihen soveltuvien puristimien käyttö olisi ollut kannattavaa, mutta useamman yrityksen jälkeen pienten JST-liitinten naaraspinnien kiinnittäminen johtoihin onnistui pelkkien kärkipihtien avulla. Käyttötarkoitukseen soveltuvan suojakotelon löytäminen oli myös vaikeaa, koska suojakotelo joutui muokkamaan, jotta siihen pystyi kiinnittämään käytettävät anturit. Opinnäytetyötä varten hankittu kotelo ei siinä käytetyn materiaalin paksuuden takia parhaimmaksi valinnaksi osoittautunut.

Avoimia lähdekoodeja ja aineistoja valittaessa niiden dokumentaatio ja käyttäjämäärät olivat tärkeässä roolissa mahdollisia ongelmatilanteita ratkaistaessa. Mikäli käytettävän kirjaston ohjeista ei ollut apua ongelmatilanteissa hyödynsin

muista lähteistä saatavia tietoja. Muualta saatujen tietojen oikeellisuuden pyrin tarkistamaan vielä toisista lähteistä.

Opinnäytetyössä oli tarkoituksena saada nopeasti valmiiksi toimiva ohjelman pohja, johon myöhemmissä vaiheissa oli mahdollista lisätä ominaisuuksia. Uusia ominaisuuksia lisättiin karttaohjelmaan sitä mukaa, kun näin ne tarpeelliseksi ja niitä oli mahdollista testata. Ohjelman kehitysversioita testasin paljon suunnitelluissa käyttöympäristöissä. Näissä pidempiaikaisissa ja toistuvissa testeissä ilmenikin ongelmia, joita ei voinut havaita ohjelman lähdekoodista tai yksittäisten testien avulla. Esiin tulevat ongelmat pyrin korjaamaan sitä mukaa kun niitä tuli ilmi.

Ohjelmointi tapahtui Arduino IDE:n, tekstieditorien ja Visual Studio Coden avulla. Ohjelmoinnin apuna käytin alkuperäisistä lähteistä peräisin olevia ohjeita. Niiden lisäksi käytettävissä oli ohjelmointikielten omat ohjeet ja Mozillan MDN Web Docs -sivusto. Ohjelmakoodin testaus tapahtui Chromiumiin perustuvilla verkkoselaimilla, jotka toimivat pöytätietokoneella ja Raspberry Pi 4 – tietokoneella.

Karttaohjelman osat oli kehitysvaiheessa varmuuskopioitu kahdelle eri kovalevyille ja itse laiteisiin. Lisäksi säilytin versioita verkossa olevassa OneDrive-kansiossa siltä varalta, että jokin laiteista olisi mennyt epäkuntoon. Varsinkinkin kehitysvaiheessa mukana kuljettamani ESP32-laite oli vaarassa rikkoutua sekä siinä käytetyn microSD-muistikortin kestävyys ei ollut tiedossa. Lisäksi version hallinnan vuoksi tallensin ohjelman kehitysversioita eri nimillä, ennen niissä tapahtuvia suuria muutoksia. Version hallintaa jouduinkin hyödyntämään muutama kerran, kun ohjelman suorituskyvyn parannukset onnistuivat vaihtelevin tuloksin.

6.2 Luotettavuus ja eettisyys

Karttaohjelmaa tehdessä pyrkimyksenä oli käyttää mahdollisimman paljon tietoa alkuperäisistä lähteistä. Täten tiedon tulkinnasta syntyneet erot jäivät

mahdollisimman pieniksi. Tosin suurin osa tiedosta oli saatavilla vain vieraalla kielellä, joten kielen käännöksessä aiheutuneet tulkintaerot olivat silti mahdollisia. Tieto oli myös joissakin tapauksissa eroteltu alkuperäisestä kontekstista, silloin tiedon ymmärtäminen saattoi olla vaikeaa, varsinkin kun tiedon sisältö poikkesi suuresti omasta tietämyksestäni. Tiedon laaja-alainen tutkiminen ja mahdollisuuksien mukaan sen testaaminen lisäsi luottamustani tiedon luotettavuudesta. Joissakin tapauksissa tietoa etsiessä jouduin arvioimaan oman tietämyksen pohjalta tiedon oikeellisuutta, koska muuta materiaalia ei ollut saatavilla. Tällöin yhteneväisyys asiaa sivuavien tietojen kanssa auttoi niiden tietojen luotettavuuden arvioinnissa.

Projektissa käytetyissä ohjelmissa, kirjastoissa ja aineistoissa käytettiin erilaisia avoimia lisenssejä, joten niiden ehtojen noudattaminen oli ehdotonta. Samoin vain Internetissä toimivien aineistojen rajapintojen käyttöehtoja noudatettiin projektin karttaohjelmassa. Lisenssien noudattamine lähinnä koostui lisenssi ehtojen mukana kulkemisesta ja joissakin tapauksissa käyttämisen mainitsemisesta itse ohjelmassa.

6.3 Hyödyntäminen ja jatkokehittäminen

Opinnäytetyön projektissa tuotettu ohjelmisto tuli omaan käyttöön ja sen kehitys jatkuu vielä opinnäytetyön tekemisen jälkeen. Opinnäytetyön dokumenttia on mahdollista hyödyntää samankaltaisen verkkoselaimessa toimivan ohjelman toteuttamisessa. Projektin toteuttaminen ja tiedonkerääminen lisäsi osaamistani full stack -kehittäjänä. Projektia tehdessä opittuja mikrokontrollerin ohjelmointi taitoja hyödynnän myös muissa vielä suunnitteluvaiheessa olevissa projekteissa.

Reittikartta-ohjelmiston perustoiminnot valmistuivat, eli sitä pystyi käyttämään kuljettujen reittien analysointiin tallenteelta, merkkien lisäämiseen ja liikkumisen reaaliaikaiseen seuraamiseen. Sydämen sykkeen seurannan lisääminen mahdollisti tallennettujen harjoitusten tehokkuuden ja kehittymisen tarkemman analysoinnin. Datan siirtämisen mahdollisuus taulukkolaskentaohjelmistoon

mahdollisesti useampien harjoitusten vertaamisen toisiinsa. Kallistuskompensoidun kompassin tekemisen kanssa eteen tulleiden ongelmien kanssa kului paljon aikaa. Kompassisuunnan paremmassa hyödyntämisessä, eli kartan turhan pyörimisen poistamisessa jäi vielä paranneltavaa, jotta sitä voisi käyttää kompassin tapaan navigoidessa.

JavaScriptillä toteutettu lähdekoodi koki opinnäytetyötä tehdessä kaksi suurta uudistusta. Tehdyt muutokset kohdentuivat siihen, miten tallennettua dataa käsiteltiin ja missä muodossa käsitelty data oli verkkoselaimen muistissa. Kaikki verkkoselaimen tarjoamat datan tallennuskeinotkin tuli testattua opinnäytetyötä tehdessä. Lähdekoodiin jäikin paljon toimivia, mutta ei käytössä olevia ratkaisuja.

ESP32:lla Arduino-sketchi ei kehityksen aikana suurempia uudelleen kirjoituksia kokenut. Yhdessä vaiheessa yritin muuttaa kaikki datan keruussa käytetyt merkkijonot (String) merkkitaulukoksi (character array) paremman muistin hallinnan vuoksi. Merkkitaulukon puskurinmuistin hallinta kuitenkin osoittautui hiukan liian hankalaksi ensimmäisellä yrityksellä, eikä tapani käyttää merkkijonoa aiheuttanut muistin loppumista edes monta tuntia kestäneissä datan keruissa. Projektin jatkokehityksessä kyseisen buffer overrunin syyn tulen selvittämään, eli missä kohdassa koodia merkkitaulukon koko kasvaa määrittelyä puskuria suuremmaksi.

Ohjelman jatkokehityksessä karttaohjelmaan lisään toimintoja sen käyttötarkoitusten mukaan. Mahdollisia käyttökohteita olisi metsänhoitotyön edistymisen seuraaminen eli kuljetun reitin alueen merkitseminen sekä auton ajo-ohjeiden antaminen. Ajo-ohjeiden antaminen vaatisi tekstimuotoisten ohjeiden muuttamisen ääniohjeiksi eli niiden nauhoittamisen ja liittämisen karttaohjelmaan. Antureiden lisääminenkin olisi mahdollista opinnäytetyössä käytettyyn Raspberry Pi:hin. Kalastusta ajatellen hyödyllinen tieto voisi esimerkiksi olla pintaveden lämpötilan seuraaminen. Ohjelman kehittäminen toimi ja tulee toimimaan niin sanotussa kehitä ja testaa -kehässä, joten eri vuodenajat rajoittavat eri toimintojen kehittämistä. Karttaan lisättävien salamatietojen toteuttaminen jäi odottamaan salamoiden esiintymistä.

Lähteet

- Adafruit. 2022. Adalogger FeatherWing - RTC + SD Add-on For All Feather Boards. <https://www.adafruit.com/product/2922>. 15.11.2023.
- Adafruit. 2023a. Feather. <https://www.adafruit.com/category/943>. 20.11.2023.
- Adafruit. 2023b. Adafruit FeatherWing OLED - 128x64 OLED Add-on For Feather - STEMMA QT / Qwiic. <https://www.adafruit.com/product/4650>. 20.11.2023.
- Adafruit. 2023c. Adafruit Ultimate GPS Breakout with GLONASS + GPS - PA1616D - 99 channel w/10 Hz updates. <https://www.adafruit.com/product/5440>. 18.11.2023.
- Agafonkin, V. 2023a. Leaflet. <https://leafletjs.com/>. 12.11.2023.
- Agafonkin, V. 2023b. Leaflet API reference. <https://leafletjs.com/reference.html>. 12.11.2023.
- Anker. 2023. What is Trickle Charging Mode?. [https://support.anker.com/s/article/What-is-Trickle-Charging-Mode#:~:text=The%20minimum%20current%20requirement%20\(the,is%20about%2030mA%20to%2090mA](https://support.anker.com/s/article/What-is-Trickle-Charging-Mode#:~:text=The%20minimum%20current%20requirement%20(the,is%20about%2030mA%20to%2090mA). 15.11.2023.
- Arduino. 2023a. Getting Started with Arduino IDE 2. <https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2>. 22.11.2023.
- Arduino. 2023b. Installing additional cores. <https://docs.arduino.cc/learn/start-ing-guide/cores>. 22.11.2023.
- Atrawalkar, C. 2022. Ultimate List Of Javascript Learning Resources [Free]. DEV. 22.1.2022. Blogi. https://dev.to/chetan_atrawalkar/ultimate-list-of-javascript-learning-resources-free-37c2. 26.11.2023.
- Bailly, N. 2019. What you should know about CORS. <https://dev.to/nicolus/what-you-should-know-about-cors-48d6>. 18.11.2023.
- Beaufort, F. 2020. Read from and write to a serial port. <https://developer.chrome.com/articles/serial/>. 20.11.2023.
- Bitsmashed. Gyroscope. SparkFun Electronics. <https://learn.sparkfun.com/tutorials/gyroscope/all>. 3.12.2023.
- Bluetooth SIG Inc. 2024. characteristic_uuids.yaml. Bitbucket-sivusto. https://bitbucket.org/bluetooth-SIG/public/src/main/assigned_numbers/uuids/characteristic_uuids.yaml. 20.02.2024.
- Brown, K. .2023. How to test SD card speed on Raspberry Pi. LinuxConfig. <https://linuxconfig.org/how-to-test-sd-card-speed-on-raspberry-pi>. 23.11.2023.
- Caruso, M.J. 2020. Applications of Magnetoresistive Sensors in Navigation System. HoneyWell. https://aerospace.honeywell.com/content/dam/aer-obt/en/documents/learn/products/sensors/technical-articles/ApplicationsofMagnetoresistiveSensorsinNavigationSystems_ta.pdf. 28.11.2023.
- CDtop technology. 2023. CD-PA1616D GNSS patch antenna module. <https://drive.google.com/file/d/1kfhYX9w9B1p3TzluX87McfV2YKG-fgQ/view>. 30.11.2023.
- Chart.JS. 2023a. Chart.js. <https://www.chartjs.org/docs/latest>. 12.11.2023. (chart.js)
- Chart.JS. 2023b. Multi Axis Line Chart. <https://www.chartjs.org/docs/latest/samples/line/multi-axis.html>. 20.11.2023.

- Chart.JS. 2023c. Interface: LineHoverOptions. <https://www.chartjs.org/docs/latest/api/interfaces/LineHoverOptions.html>. 20.11.2023.
- Chart.JS. 2023d. Performance. <https://www.chartjs.org/docs/latest/general/performance.html>. 20.11.2023.
- CircuitPython. 2023. CircuitPython. GitHub-sivut. <https://github.com/adafruit/circuitpython>. 15.11.2022.
- Community wiki. 2016. How can I convert a Bluetooth 16 bit service UUID into a 128 bit UUID?. Stack Overflow-sivusto. <https://stackoverflow.com/questions/36212020/how-can-i-convert-a-bluetooth-16-bit-service-uuid-into-a-128-bit-uuid>. 20.02.2024.
- Cunningham, A. 2023. Raspberry Pi availability is visibly improving after years of shortages. <https://urly.fi/3iSB>. 15.11.2023.
- Derflinger, T. 2021. Comparing 3 Popular JavaScript Charting Libraries. Thoughts by Thomas Derflinger. 23.5.2022. Blogi. <https://www.tderflinger.com/comparing-3-popular-javascript-charting-libraries>. 20.11.2023. (chart.js)
- Django. 2023. Django at a glance. <https://docs.djangoproject.com/en/4.2/intro/overview/>. 17.11.2023.
- Espressif Systems. 2023. Client.ino. GitHub-sivusto. <https://github.com/espressif/arduino-esp32/blob/master/libraries/BLE/examples/Client/Client.ino>. 20.2.2023.
- European Space Agency. 2023. Galileo and EGNOS. https://www.esa.int/Applications/Navigation/Galileo_and_EGNOS 18.11.2023.
- EUSPA. 2022. What is SBAS?. <https://www.euspa.europa.eu/european-space/eu-space-programme/what-sbas>. 18.11.2023.
- Firdaus, T. 2023. Quick Tip: Using the HTML5 Download Attribute. <https://web-design.tutsplus.com/quick-tip-using-the-html5-download-attribute--cms-23880t>. 18.11.2023.
- Flask. 2010. BSD-3-Clause License. <https://flask.palletsprojects.com/en/2.3.x/license/>. 3.12.2023.
- Flask. 2023a. Quickstart. <https://flask.palletsprojects.com/en/3.0.x/quickstart/>. 17.11.2022.
- Flask. 2023b. Templates. <https://flask.palletsprojects.com/en/3.0.x/tutorial/templates/>. 26.11.2023.
- Flask. 2023c. Jinja Setup. <https://flask.palletsprojects.com/en/3.0.x/templating/#jinja-setup>. 26.11.2023.
- Flask. 2023d. Dependencies. <https://flask.palletsprojects.com/en/3.0.x/installation/#dependencies>. 26.11.2023.
- Fraczek, W. 2003. Mean Sea Level, GPS, and the Geoid. Esri Applications Prototype Lab. <https://www.esri.com/news/arcuser/0703/geoid1of3.html>. 18.11.2023.
- GDAL. 2023. gdal2tiles.py. <https://gdal.org/programs/gdal2tiles.html#gdal2tiles>. 25.11.2023.
- Geerling, J. 2019. Raspberry Pi microSD card performance comparison - 2019. Blog. 10.6.2019. Blogi. <https://www.jeffgeerling.com/blog/2019/raspberry-pi-microsd-card-performance-comparison-2019>. 12.12.2023.
- Geoapify. 2019. Leaflet vs OpenLayers. What to choose?. <https://www.geoapify.com/leaflet-vs-openlayers>. 19.11.2023.
- GPS.gov. 2023. GPS Accuracy. the National Coordination Office for Space-Based Positioning, Navigation, and Timing. <https://www.gps.gov/systems/gps/performance/accuracy/>. 18.11.2023.

- Häkli, P., Puupponen, J., Koivula, H. & Poutanen, M. 2009. Suomen geodeettiset koordinaatistot ja niiden väliset muunnokset. Geodeettinen laitos. <https://www.maanmittauslaitos.fi/sites/maanmittauslaitos.fi/files/fgi/GLtiedote30.pdf>. 15.11.2023.
- Hollingworth, G. 2022. Raspberry Pi OS (64-bit). Raspberry Pi. <https://www.raspberrypi.com/news/raspberry-pi-os-64-bit/>. 3.12.2023.
- JetBrains. 2022. Python Developers Survey 2022 Results. <https://lp.jetbrains.com/python-developers-survey-2022/#FrameworksLibraries>. 17.11.2023.
- jQuery. 2023a. What is jQuery?. OpenJS Foundation. <https://jquery.com>. 12.11.2023.
- jQuery. 2023b. Category: Shorthand Methods. OpenJS Foundation. <https://api.jquery.com/category/ajax/shorthand-methods/>. 20.11.2023.
- jQuery Foundation. 2023. License. <https://jquery.org/license/>. 13.12.2023.
- jQuery UI. 2022. LICENSE.txt. GitHub-sivut. <https://github.com/jquery/jquery-ui/blob/main/LICENSE.txt>. 13.12.2023.
- jQuery User Interface. 2023a. jQuery UI 1.13 API Documentation. OpenJS Foundation. <https://api.jqueryui.com>. 12.11.2023.
- jQuery User Interface. 2023b. OpenJS. Category: Method Overrides. <https://api.jqueryui.com/category/overrides/>. 20.11.2023.
- Lady ada. 2023. Adafruit. Magnetic Calibration with MotionCal. <https://learn.adafruit.com/adafruit-sensorlab-magnetometer-calibration/magnetic-calibration-with-motioncal>. 20.11.2023.
- Leaflet. 2016. Leaflet. GitHub-sivut. <https://github.com/Leaflet/Leaflet/tree/rotate>. 19.11.2023.
- Leaflet. 2023. LICENSE. GitHub-sivut. <https://github.com/Leaflet/Leaflet/blob/main/LICENSE>. 13.12.2023.
- Leaflet. 2012. Window scrolls on first click. GitHub-sivusto. <https://github.com/Leaflet/Leaflet/issues/1228>. 12.1.2024.
- LeBlanc-Williams, M. 2023a. CircuitPython Libraries on Linux & Raspberry Pi. Adafruit. <https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/circuitpython-raspi>. 17.11.2023.
- LeBlanc-Williams, M. 2023b. Overview. Adafruit. <https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/overview>. 17.11.2023.
- LeBlanc-Williams, M. 2023c. Installing Blinka on Raspberry Pi. Adafruit. <https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/installing-circuitpython-on-raspberry-pi>. 17.11.2022.
- Liedman, P. 2015a. About. <https://www.liedman.net/leaflet-routing-machine/#about>. 12.11.2023.
- Liedman, P. 2015b. Tutorials. <http://www.liedman.net/leaflet-routing-machine/tutorials/>. 19.11.2023.
- Liedman, P. 2015c. Leaflet Routing Machine API. <https://www.liedman.net/leaflet-routing-machine/api/>. 12.11.2023. (leaflet routing)
- Liedman, P. 2015d. GraphHopper, Mapbox, Valhalla and other routing software. <http://www.liedman.net/leaflet-routing-machine/tutorials/alternative-routers/>. 19.11.2023.
- Liedman, P. 2023. leaflet-control-geocoder. GitHub-sivut. <https://github.com/perliedman/leaflet-control-geocoder>. 13.12.2023.
- Long, S. 2022. An update to Raspberry Pi OS Bullseye. <https://www.raspberrypi.com/news/raspberry-pi-bullseye-update-april-2022/>. 3.12.2023.

- Maanmittauslaitos. 2023a. Maanmittauslaitoksen avoimien aineistojen jakelu avattu Karttapaikka-palveluun. <https://www.maanmittauslaitos.fi/ajan-kohtaista/maanmittauslaitoksen-avoimien-aineistojen-jakelu-avattu-karttapaikka-palveluun>. 25.11.2023.
- Maanmittauslaitos. 2023b. Lataa paikkatietoaineistoja. <https://asiointi.maanmittauslaitos.fi/karttapaikka/tiedostopalvelu>. 25.11.2023.
- Maanmittauslaitos. 2023c. Ohje API-avaimen käyttöön. <https://www.maanmittauslaitos.fi/rajapinnat/api-avaimen-ohje>. 25.11.2023.
- Maanmittauslaitos. 2023d. Paikkatietojen rajapintapalvelut. <https://www.maanmittauslaitos.fi/rajapinnat/paikkatiedot>. 25.11.2023.
- Maanmittauslaitos. 2023e. Maanmittauslaitoksen avoimen tietoaaineiston Nimeä CC 4.0 -lisenssi. <https://www.maanmittauslaitos.fi/avoindata-lisenssi-cc40>. 25.11.2023.
- Maanmittauslaitos. 2023f. Maanmittauslaitoksen paikkatietoaineistojen formaatit. <https://www.maanmittauslaitos.fi/paikkatietoaineistojen-formaatit>. 25.11.2023.
- Maanmittauslaitos. 2023g. Vihjeitä Maanmittauslaitoksen Maastotietokannan OGC API Features - palvelun käyttäjille <https://www.maanmittauslaitos.fi/sites/maanmittauslaitos.fi/files/attachments/2020/12/Maastotietokannan%20OGC%20API%20Features%20-%20vihjeit%C3%A4%20valmisohjelmistojen%20k%C3%A4ytt%C3%A4jille.pdf>. 25.11.2023.
- Maanmittauslaitos. 2023h. Satelliittipaikannus. <https://www.maanmittauslaitos.fi/tutkimus/teematietoa/satelliittipaikannus>. 18.11.2023.
- McWhorter, P. 2019. 9-Axis IMU LESSON 9: Accurate and Stable Tilt Using Accelerometers, Gyros and a Complimentary Filter. YouTube-video. <https://www.youtube.com/watch?v=cUFqSGYZO5k>. 28.11.2023.
- MDN contributors. 2023a. JavaScript — Dynamic client-side scripting. Mozilla. <https://developer.mozilla.org/en-US/docs/Learn/JavaScript>. 8.12.2023.
- MDN contributors. 2023b. HTML basics. Mozilla. https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics. 8.12.2023.
- MDN contributors. 2023c. Global attributes. Mozilla. https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes. 8.12.2023.
- MDN contributors. 2023d. CSS selectors. Mozilla. https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors. 8.12.2023.
- MDN contributors. 2023e. <style>: The Style Information element. Mozilla. <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/style>. 8.12.2023.
- MDN contributors. 2023f. Document Object Model (DOM). Mozilla. https://developer.mozilla.org/en-US/docs/Web/API/HTML_DOM_API. 8.12.2023.
- MDN contributors. 2023g. The web and web standards. Mozilla. https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/The_web_and_web_standards. 8.12.2023.
- MDN contributors. 2023h. Origin. Mozilla. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Origin>. 29.11.2023.

- MDN contributors. 2023i. <input type="file">. Mozilla. <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/file>. 18.11.2023.
- MDN contributors. 2023j. What is JavaScript?. Mozilla. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. 24.11.2023.
- MDN contributors. 2023k. How to use promises. Mozilla. <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Promises>. 24.11.2023.
- MDN contributors. 2023l. Drawing shapes with canvas. MDM. https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Drawing_shapes. 20.11.2023.
- MDN contributors. 2023m. Reason: CORS request not HTTP. Mozilla. <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS/Errors/CORSRequestNotHttp>. 22.11.2023.
- Mediagis. 2024. Nominatim Docker. GitHub-sivusto. <https://github.com/mediagis/nominatim-docker>. 20.04.2024.
- MediaTek. 2016. MT3333 All-in-One GNSS Datasheet. https://d86o2zu8ugzlg.cloudfront.net/mediatek-craft/documents/mt3333/MT3333_Datasheet.pdf. 18.11.2023.
- Micron. 2008. Wear-Leveling Techniques in NAND Flash Devices. https://www.micron.com/-/media/client/global/documents/products/technical-note/nand-flash/tn2942_nand_wear_leveling.pdf. 23.11.2023.
- Olujinmi, T. 2022. Raspberry Pi Pico vs. ESP32: Which Microcontroller Is Right for You?. <https://www.makeuseof.com/raspberry-pi-pico-vs-esp32-microcontroller/>. 15.11.2023.
- Opensource.org. 2023. The MIT License. <https://opensource.org/license/mit/>. 13.12.2023.
- OpenStreetMaps. 2023a. Legal FAQ. OpenStreetMaps Wiki-sivut. https://wiki.openstreetmap.org/wiki/Legal_FAQ. 3.12.2023.
- OpenStreetMaps. 2023b. *Routing. OpenStreetMaps Wiki-sivut. <https://wiki.openstreetmap.org/wiki/Routing>. 3.12.2023.
- OpenStreetMaps. 2023c. Nominatim. OpenStreetMaps Wiki-sivut. <https://wiki.openstreetmap.org/wiki/Nominatim>. 3.12.2023.
- OpenStreetMaps. 2023d. Raster tile providers. https://wiki.openstreetmap.org/wiki/Raster_tile_providers. 3.12.2023.
- OpenStreetMaps. 2023e. Tile servers. OpenStreetMaps Wiki-sivut. https://wiki.openstreetmap.org/wiki/Tile_servers. 3.12.2023.
- OSMF Operations Working Group. 2023. Tile Usage Policy. <https://operations.osmfoundation.org/policies/tiles/>. 3.12.2023.
- OSMR. 2023. OSMR API Documentation. <https://project-osrm.org/docs/v5.24.0/api/#>. 2.12.2023.
- Overvoorde. A. 2023. openstreetmap-tile-server. GitHub-sivusto. <https://github.com/Overv/openstreetmap-tile-server>. 20.04.2024.
- Patterson, D. 2016. [OSRM-talk] The future of Windows support. <https://lists.openstreetmap.org/pipermail/osrm-talk/2016-May/001214.html>. 26.11.2023.
- Pedamkar, P. 2023. SVG vs Canvas. <https://www.educba.com/svg-vs-canvas/>. 20.11.2023.

- Pedley, M. 2013. Tilt Sensing Using a Three-Axis Accelerometer. NXP/ Freescale Semiconductor, Inc. <https://www.nxp.com/docs/en/application-note/AN3461.pdf>. 28.11.2023.
- Pinout.xyz. 2023. Raspberry Pi Pinout. Gadgetoid. https://pinout.xyz/pinout/pin33_gpio13/. 6.12.2023.
- Polar. 2024. Polar H10. <https://www.polar.com/en/sensors/h10-heart-rate-sensor>. 20.1.2024.
- Prikhodko, I.P., Trusov, A.A. & Shkel, A.M. 2013. Compensation of drifts in high-Q MEMS gyroscopes using temperature self-sensing. Teoksessa Sensors and Actuators A: Physical. Volume 201. Hollanti: Elsevier B.V. ScienceDirect. 12.12.2023.
- Project-OSRM. 2020. Demo Server. GitHub-sivu. <https://github.com/Project-OSRM/osrm-backend/wiki/Demo-server>. 26.11.2023.
- Project-OSRM. 2022. osmr-backend. GitHub-sivu. <https://github.com/Project-OSRM/osrm-backend/pkgs/container/osrm-backend>. 2.12.2023.
- Project-OSRM. 2021. Disk and Memory Requirements. GitHub-sivu. <https://github.com/Project-OSRM/osrm-backend/wiki/Disk-and-Memory-Requirements>. 3.12.2023.
- Puszynski, A. 2019. These are the best JavaScript chart libraries for 2019. <https://www.freecodecamp.org/news/these-are-the-best-javascript-chart-libraries-for-2019-29782f5e1dc2/>. 20.11.2023.
- Python Wiki. 2020. GlobalInterpreterLock. Python Software Foundation. <https://wiki.python.org/moin/GlobalInterpreterLock>. 1.12.2023.
- Python. 2023. multiprocessing — Process-based parallelism. Python Software Foundation. <https://docs.python.org/fr/3/library/multiprocessing.html>. 1.12.2023.
- Raruto. 2023a. leaflet-rotate. GitHub-sivu. <https://github.com/Raruto/leaflet-rotate>. 19.11.2023.
- Raruto. 2023b. leaflet-rotate.spec.js. GitHub-sivu. <https://github.com/Raruto/leaflet-rotate/blob/master/examples/leaflet-rotate.spec.js>. 20.11.2023.
- Raspberry Pi. 2023. Raspberry Pi OS. <https://www.raspberrypi.com/documentation/computers/os.html#python-on-raspberry-pi>. 23.11.2023.
- Raspberry Pi. 2023a. Raspberry Pi Pico. <https://www.raspberrypi.com/products/raspberry-pi-pico/>. 15.11.2023.
- Raspberry Pi. 2023b. Products. <https://www.raspberrypi.com/products/>. 15.11.2023.
- Rembor, K. 2017. CircuitPython Libraries. Adafruit. <https://learn.adafruit.com/welcome-to-circuitpython/circuitpython-libraries>. 23.11.2023.
- Rembor, K. 2022a. Power Management. Adafruit. <https://learn.adafruit.com/adafruit-esp32-feather-v2/power-management-2>. 15.11.2023.
- Rembor, K. 2022b. CircuitPython. Adafruit. <https://learn.adafruit.com/adafruit-esp32-feather-v2/circuitpython>. 3.12.2023.
- Rembor, K. 2023a. Overview. Adafruit. <https://learn.adafruit.com/adafruit-esp32-feather-v2>. 15.11.2023.
- Rembor, K. 2023b. Pinouts. Adafruit. <https://learn.adafruit.com/adafruit-esp32-feather-v2/pinouts>. 15.11.2023.

- Rembor, K., Herrada, E. & Rubell, B. 2023. Arduino IDE Setup. Adafruit. <https://learn.adafruit.com/adafruit-esp32-feather-v2/arduino-ide-setup>. 20.11.2023.
- Remington, S.J. 2024. ICM_20948-AHRS. GitHub-sivusto. https://github.com/jremington/ICM_20948-AHRS. 9.2.2024.
- Ronikar. 2023a. Leaflet. GitHub-sivut. <https://github.com/ronikar/Leaflet/tree/master>. 13.11.2023.
- Ronikar. 2023b. Rotate Leaflet Map. NPM. <https://www.npmjs.com/package/leaflet-rotate-map>. 19.11.2023.
- Sanz Subirana, J., Juan Zornoza, J.M. & Hernández-Pajares, M. 2011. Time References in GNSS. https://gssc.esa.int/navipedia/index.php/Time_References_in_GNSS. 18.11.2023.
- Schwenk, J., Niemietz, M. & Mainka, C. 2017. Same-Origin Policy: Evaluation in Modern Browsers. Teoksessa Kirda, E. & Ristenpart, T. (toim.). Proceedings of the 26th USENIX Security Symposium 7. Berkeley, CA: USENIX Association. 713–727. Usenix. <https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-schwenk.pdf>. 8.12.2023.
- Siepert, B. 2020a. adafruit_icm20x.py. GitHub-sivut. https://github.com/adafruit/Adafruit_CircuitPython_ICM20X/blob/main/adafruit_icm20x.py. 23.11.2023.
- Siepert, B. 2020b. Overview. <https://learn.adafruit.com/adafruit-tdk-invensense-icm-20948-9-dof-imu>. 3.12.2023.
- SpaceFinland. 2023. What is the accuracy of GNSS location data? Työ- ja elinkeinoministeriö. <https://spacefinland.fi/en/accuracy-of-gnss-location-data>. 18.11.2023.
- SparkFun Electronics. 2021. InvenSense Digital Motion Processor (DMP™). GitHub-sivusto. https://github.com/sparkfun/SparkFun_ICM-20948_ArduinoLibrary/blob/main/DMP.md. 9.2.2024.
- TDK. 2021. <https://invensense.tdk.com/wp-content/uploads/2021/10/DS-000189-ICM-20948-v1.5.pdf>. 28.11.2023.
- The OpenStreetMap Foundation. 2023. Licence/Licence and Legal FAQ. https://osmfoundation.org/wiki/Licence/Licence_and_Legal_FAQ. 3.12.2023.
- TIOBE. 2023. TIOBE Index for November 2023. <https://www.tiobe.com/tiobe-index/>. 23.11.2023.
- Volkman, M. 2020. Comparing Python and JavaScript: a guide for developers. Object Computing. <https://objectcomputing.com/resources/publications/sett/december-2020-comparing-python-and-javascript>. 23.11.2023.
- W3C & WHATWG. 2021. W3C/WHATWG Relationship update. https://www.w3.org/2021/06/WHATWG-W3C-MOU_2021_update.html. 8.12.2023.
- W3C. 2016. Content Security Policy Level 2. <https://www.w3.org/TR/CSP2/>. 8.12.2023.
- W3School. 2023. JavaScript Variables. Refsnes Data. https://www.w3schools.com/js/js_variables.asp. 11.12.2023.
- WCH. 2023a. USB to High Speed Serial Port Chip CH9102. NanjingQinhengMicroelectronic. <https://www.wch-ic.com/products/CH9102.html>. 20.11.2023.

- WCH. 2023b. CH343SER.ZIP. NanjingQinhengMicroelectronic.
https://www.wch-ic.com/downloads/CH343SER_ZIP.html. 3.12.2023.
- Wdzięczna, D. 2019. 9 Best Online Resources to Start Learning Python Today.
<https://learnpython.com/blog/9-best-python-online-resources-start-learning/>. 23.11.2023.
- Werkzeug. 2023. Werkzeug Tutorial. <https://werkzeug.palletsprojects.com/en/3.0.x/tutorial/#step-0-a-basic-wsgi-introduction>.
26.11.2023.
- WHATWG. 2023a. Fetch. <https://fetch.spec.whatwg.org/#http-cors-protocol>.
24.11.2023.
- WHATWG. 2023b. HTML: The Living Standard.
<https://html.spec.whatwg.org/dev/introduction.html#history-2>.
24.11.2023.
- ZaneL. 2021. Arduino library for the ICM-20948 motion tracking sensor -- with DMP support. GitHub-sivusto. <https://github.com/ZaneL/Teensy-ICM-20948>. 9.2.2024.
- Zingchart. 2023. Awesome Charting. GitHub-sivut. <https://github.com/zing-chart/awesome-charting>. 20.11.2023.