Lappeenrannan teknillinen yliopisto
School of Engineering Science
Tietotekniikan koulutusohjelma
Master's Programme in Software Engineering

Diplomityö

Anton Simola

SENSORIDATAN HYÖDYNTÄMINEN VARASTONHALLINNASSA SAHAN TUKKIKENTÄLLÄ

Työn tarkastaja(t): Professori Jari Porras
Tutkijatohtori Ari Happonen

Työn ohjaaja(t): Tutkijatohtori Ari Happonen
Diplomi-insinööri Juha Alatalo

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
School of Engineering Science
Tietotekniikan koulutusohjelma
Master's Programme in Software Engineering

Anton Simola

Sensoridatan hyödyntäminen varastonhallinnassa sahan tukkikentällä

Diplomityö

2019

53 sivua, 7 kuvaa

Työn tarkastajat: Professori Jari Porras
Tutkijatohtori Ari Happonen

Hakusanat: esineiden internet, satelliittipaikannus, varastonhallinta, mikropalvelut
Keywords: Internet of Things, satellite navigation, inventory management, microservices

Työssä esitellään sensoridataa hyödyntävän sahan varastonhallintasovelluksen määrittely-, suunnittelu- ja toteutustyö. Työssä perehdyttiin varastonhallinnassa hyödynnettävien sensoriteknologioiden taustoihin, käyttöliittymäsuunnitteluun, työjärjestysoptimointiin sekä ohjelmistoarkkitehtuuriin. Teoriatiedon ja haastatteluiden pohjalta toteutettiin ensimmäinen versio tukkikentän varastonhallintasovelluksesta. Sovelluksen kenttätestaus asiakassahalla toteutettiin loppukäyttäjien kanssa. Lopputuloksena oli asiakaslähtöisesti kehitetty ensimmäinen versio suurilta osin automatisoidusta tukkikentän hallintajärjestelmästä, jota voidaan kehittää kerättyjen jatkokehitysideoiden pohjalta.

ABSTRACT

Lappeenranta University of Technology
School of Engineering Science
Software Engineering
Master's Programme in Software Engineering

Anton Simola

Utilizing sensor data in log yard inventory management

Master's Thesis

53 pages, 7 figures

Examiners: Professor Jari Porras
D.Sc. (Tech.) Ari Happonen

Keywords: Internet of Things, satellite navigation, inventory management, microservices

This master's thesis presents requirements, design and implementation work done for a sawmill inventory management software utilizing sensor data. The theory part focuses on the background of different sensor technologies, user interface design, work order optimization and software architecture. First version of a log yard inventory management system was developed based on the theory part and end-user interviews. A field test with the prototype was conducted with the end-users on a client sawmill. The result was a largely automated log yard inventory management system developed with customer-oriented approach that can be developed further with the gathered feedback from the users.

ALKUSANAT

Kiitos Henkalle, Nuutille, Juhalle, Tomille, Terolle, Tonille, Petrille, Jerelle, Marille ja muulle Finnoksen porukalle tuesta ja avusta työn aikana. Kiitos mielenkiintoisesta diplomityön aiheesta. Työn parissa olen saanut käyttää vahvuuksiani, mutta mikä tärkeämpää, olen saanut myös arvokasta kokemusta asioista, joiden koin olleen sopivasti oman mukavuusalueeni ulkopuolelta. Työhön on sisältynyt ohjelmoinnin lisäksi laitteiden kanssa näpräilyä, haastatteluja, esityksiä, kenttätestausta sekä paljon kommunikointia eri osapuolien kesken. Työ on avartanut käsitystäni siitä, mitä kaikkea uuden tuotteen kehitykseen oikeastaan kuuluukaan. Työtä tehdessä kertyneestä kokemuksesta on korvaamaton hyöty jatkossa elämässäni, joten olen suunnattoman kiitollinen annetusta mahdollisuudesta tehdä diplomityö Finnokselle tästä aiheesta.

Erityiskiitos Versowoodille, eritoten Samuli Salmiselle ideoista, palautteesta ja testausmahdollisuudesta. Suuret kiitokset myös muille sahoille haastatteluista.

Paljon kiitoksia LUT:n Ari Happoselle ja Jari Portaalle avusta työni kanssa. Kiitän suuresti myös perhettäni ja ystäviäni tuesta työn aikana.

SISÄLLYSLUETTELO

1 JOHDANTO 4

1.1 TAVOITTEET JA RAJAUKSET 5

1.2 TYÖN RAKENNE 6

2 TAUSTA 7

2.1 TOIMEKSIANTAJAN TAUSTA 7

2.2 TYYPILLISEN SAHAN PROSESSIKUVAUS 8

2.3 KONEKUSKIN TYÖSKENTELYN NYKYTILANNE 9

2.4 KILPAILIJAT JA TARVE MARKKINOILLA 11

3 TEORIA 12

3.1 PAIKANNUS 12

3.2 AJONEUVON SUUNTA 15

3.3 KÄYTTÖLIITTYMÄSUUNNITTELU AJONEUVOON 16

3.4 TYÖN- JA REITINOPTIMOINTI 18

4 SUUNNITTELU JA TOTEUTUS 21

4.1 HAASTATTELUT 21

4.2 TEKNOLOGIOIDEN VALINTA 24

4.2.1 Web vs. native 24
4.2.2 Ohjelmointikielet, ohjelmistokehykset ja tietokanta 25
4.2.3 Karttakirjasto 25
4.2.4 GeoJSON-standardi 27
4.2.5 Reitinoptimointikirjasto 28

4.3 LAITTEISTO 30

4.4 ARKKITEHTUURI 31

4.5 PALVELUIDEN VÄLINEN KOMMUNIKAATIO 33

4.6 SOVELLUKSEN TOIMINNALLISUUKSIEN KUVAUS 35

4.6.1 Alkuasetukset 35
4.6.2 Tietojen tarkastelu kartalla 36
4.6.3 Varastonsiirto 37
4.6.4 Varoitukset 39

5 TULOKSET 40

6 POHDINTA JA TULEVAISUUS 43

7 YHTEENVETO 45

LÄHTEET 46

SYMBOLI- JA LYHENNELUETTELO

GPS	Global Positioning System
GNSS	Global Navigation Satellite System
IMU	Inertial Measurement Unit
IoT	Internet of Things
LoRa	Long Range
MVP	Minimum Viable Product
NMEA	National Marine Electronics Association
NTRIP	Networked Transport of RTCM via Internet Protocol
RFID	Radio Frequency Identification
RTK	Real Time Kinematic
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem

1 JOHDANTO

Viime vuosien tietotekniikan trendisanastoon ovat kuuluneet vahvasti muun muassa esineiden internet (engl. Internet of Things, IoT), koneoppiminen (engl. Machine Learning, ML) ja tekoäly (engl. Artificial Intelligence, AI). Aikakauttamme on jopa nimitetty neljänneksi teolliseksi vallankumoukseksi (engl. Industry 4.0), jonka ytimessä on optimoidun teollisen tuotannon yhdistäminen verkottuneisiin älykkäisiin laitteisiin sekä tästä syntyvän datan analysointi [1]. Myös hieman hitaammin liikkuvat, suuret teollisuudenalat – kuten sahateollisuus – ovat havahtuneet esineiden internetin ja tekoälyn mahdollistamaan tuotannon optimointiin, ja ryhtyneet investointeihin. Esimerkiksi entistä tarkemmilla mittalaitteilla pyritään minimoimaan raaka-ainehukkaa, ja tietojen välitystä eri järjestelmien välillä automatisoidaan. Tekoälyä on ryhdytty hyödyntämään esimerkiksi konenäön muodossa, kun laatupoikkeamia etsitään raaka-aineista ja lopputuotteista. Laitteiden verkottumista hyödynnetään enenevässä määrin, ja tietokoneita nähdään paikoissa, joissa niitä ei ole ennen hyödynnetty. Materiaalivirran seurantaan panostetaan yhä enemmän. Koko tuotantoprosessin läpi jatkuvan seurannan avulla pyritään ennustamaan ja optimoimaan tuotantoa tulevaisuudessa. Kokonaisvaltaisen seurannan hyödyt nähdään yhä useammin.

Sahan tukkikentällä ei ole aikaisemmin nähty juurikaan optimointi- tai seurantatyökaluja, mutta viime aikoina paikannuslaitteiden ja muiden sensorien kustannusten pienentyessä ja tarkkuuden parantuessa on ryhdytty visioimaan erilaisia sovelluksia. Tukkikentän tilannetta visualisoimalla pyritään tehostamaan työskentelyä. Virheitä voitaisiin vähentää hälytyksillä vääristä varastonsiirroista. Tukkikentän seuranta on yksi askel laajamittaisessa materiaalinseurannassa. Tukista laskettava tieto hävisi suurilta osin, kun se lajitellaan ja siirretään pinoon, sillä ilman seurantaa pinosta tiedetään lopulta vain tukkiluokka. Esittelemällä sovellus tähän tarkoitukseen voidaan seurata esimerkiksi pinojen ikäjakaumaa, mikä on tärkeä tieto tuotannonsuunnittelussa. Lisäksi voidaan tehdä arviota sahalle päätyvistä tukeista, kun pinoissa säilytetään puiden lasku -ja nostopaikat. Pienilläkin parannuksilla tukkikentän työskentelyprosessiin voi olla yllättävänkin suuri hyöty toiminnan tehostamiselle, mutta toisaalta toteutuksessa on helppo myös epäonnistua. Jos käytettävyydessä tai luotettavuudessa ilmenee ongelmia, jää sovellus helposti täysin

käyttämättä. Niinpä sovellus on toteutettava harkitusti loppukäyttäjien palautteen perusteella.

1.1 Tavoitteet ja rajaukset

Työn tavoitteena on suunnitella ja toteuttaa ensimmäinen versio, niin sanottu MVP (Minimum Viable Product) sensoridataa hyödyntävästä sovelluksesta tukkikentän varastonhallinnan ja operoinnin tueksi. Sovelluksen tarkoituksena on vähentää virheitä tukkien siirtämisessä (esimerkiksi tukkien laskeminen väärään varastopaikkaan), lisätä työn tehokkuutta antamalla optimoituja työohjeita, sekä helpottaa kommunikaatiota konekuskien, tukkilajittelun ja sahan välillä. Työssä keskitytään sovelluksen kehittämiseen nimenomaan konekuskien näkökulmasta, kuitenkin niin että sovellusta voidaan jatkossa laajentaa muille käyttäjäryhmille, esimerkiksi tuotannonsuunnittelijoille. Näin ollen esimerkiksi raportoinnin kehitys jätetään tässä vaiheessa työn ulkopuolelle, ja sen sijaan keskitytään konekuskien näkökulmasta tärkeisiin elementteihin, kuten käyttöliittymäsuunnitteluun niin että sovelluksen käyttö on mahdollisimman helppoa ajon aikana. Teoreettisemmat tietojenkäsittelytieteen seikat kuten esimerkiksi erilaiset reitinoptimointialgoritmit ja niiden taustat jätetään tarkoituksella pintapuolisiksi, ja keskitytään käytännön tuotteen kehittämiseen.

Tutkimuskysymyksiä ovat:

- Millainen paikannusteknologia on sopiva tukkikentän kaltaisessa varastointiympäristössä?
- Miten mallintaa tukkikenttä ja erityisesti sen varastopaikat kartalla niin, että se hyödyttää konekuskeja heidän työssään?
- Miten paikkatiedon ja muun sensoridatan avulla voitaisiin automatisoida varastonhallintaan liittyviä operaatioita?
- Miten varastonhallintaa voidaan optimoida tällaisen järjestelmän tuottaman datan perusteella?

Teoriaosuudessa esitellään perustiedot eri satelliittipaikannusteknologioista, muista hyödynnettävistä sensoritekniikoista, käyttöliittymän suunnittelusta ajoneuvoon, sekä

etsitään kirjallisuudesta mahdollisia aiempia vastaavanlaisia sovelluksia ja pyritään oppimaan niistä. Teoriaosuudessa tutkitaan myös, miten tukkikentän kaltaisessa dynaamisessa ympäristössä voitaisiin toteuttaa reitti- ja työjärjestysoptimointia. Vaatimusmäärittelyn tukena avoimia haastatteluja käydään erilaisten loppukäyttäjäryhmien välillä, ja tulokset esitellään haastatteluosiossa. Teoriatutkimuksen ja haastatteluiden pohjalta kerätään ja priorisoidaan vaatimukset. Vaatimuksia käytetään suunnittelun tukena laite-, ohjelmisto- ja arkkitehtuurivalinnoissa. Tavoitteena onkin kuvata laaja-alaisesti ohjelmistokehitysprosessin eri vaiheita ja sitä, miten asiakaslähtöinen kehitys vaikuttaa valintoihin prosessin aikana. Työ on soveltava tutkimus, jossa ei pyritä luomaan uutta tietoa vaan hyödyntämään olemassa olevaa uuden tuotteen kehityksen tukena.

1.2 Työn rakenne

Luvussa 2 kuvataan taustatietoja lukijan aiheeseen johdattelemiseksi. Luvussa 3 esitellään teoriatietoa sensoriteknologioista, perehdytään käyttöliittymäsuunnitteluun heuristiikkojen pohjalta, sekä esitellään lyhyesti mahdollisia optimointitekniikoita. Luvussa 4 esitellään haastattelujen tulokset, teknologioiden valintoihin vaikuttaneet seikat sekä kuvataan toteutuksen arkkitehtuuria ja toiminnallisuuksia. Luvussa 5 kuvataan kenttätestien tuloksia. Luvussa 6 pohditaan, miten toteutettua ensimmäistä versiota voitaisiin jatkokehittää työn aikana saadun palautteen pohjalta. Luku 7 on yhteenveto työstä.

2 TAUSTA

Aliluvuissa kuvataan taustatietoja toimeksiantajasta, sahausprosessista yleisesti, tutustutaan konekuskin nykyiseen työskentelymalliin, sekä suoritetaan lyhyt katsaus muihin vastaaviin tuotteisiin.

2.1 Toimeksiantajan tausta

Työn toimeksiantaja on Lappeenrantalainen Finnos Oy, joka toimittaa laitteita ja ohjelmistoja ensisijaisesti sahateollisuuteen. Yritys on perustettu vuonna 2016. Yrityksen päätuotteena on röntgen- ja lasermittaukseen perustuva tukkimittari tukkilajitteluun sekä siihen liittyvät ohjelmistot. Tavoitteena on laajentaa tuote- ja palveluvalikoimaa lopulta koko sahausprosessin eri vaiheisiin, jotta tarkka tukeista kerättävä data saadaan hyötykäyttöön mahdollisimman monipuolisesti.

Tukkilajittelua seuraava tukkikentän hallinta on luonnollinen seuraava vaihe sahan prosessissa ja näin ollen sopiva alue tuote- ja palveluperheen laajennuksessa. Tukkimittarilla mitattua tietoa voidaan käyttää tukkikentällä varastotilanteen seurantaan. Tarkka lajittelu menee osittain hukkaan, jos varastoinnissa sattuu virheitä, ja tukkeja päätyy esimerkiksi vääriin varastopaikkoihin. Lisäksi mittausprosessista saatava tilastotieto tukkimääristä voi auttaa ennustamaan varastopaikkojen täyttymistä tulevaisuudessa. Näin voitaisiin luoda tehtävälista sahan tilausten, lokeroiden täyttymisennusteen ja lajittelun syöttötarpeiden pohjalta. Optimoidulla tehtävälistalla saataisiin aikaan kustannussäästöjä, kun kuljettua matkaa minimoitaisiin ja turhaa liikkumista vähennettäisiin. Lopullinen päämäärä olisi kuitenkin se, että tukista kerättävä data olisi kytköksissä kyseiseen yksilöön koko tuotantoprosessin ajan aina lopputuotteeksi asti niin, ettei tietoa häviäisi prosessin eri vaiheissa. Tätä päämäärää voitaisiin tavoitella esimerkiksi niin, että tukkilajittelumittari, tukkikenttäsovellus sekä sahan eteen asennettava mittari voisivat yhteistyössä yksilöidä tukkeja ja ylläpitää tiedon säilymistä.

2.2 Tyypillisen sahan prosessikuvaus

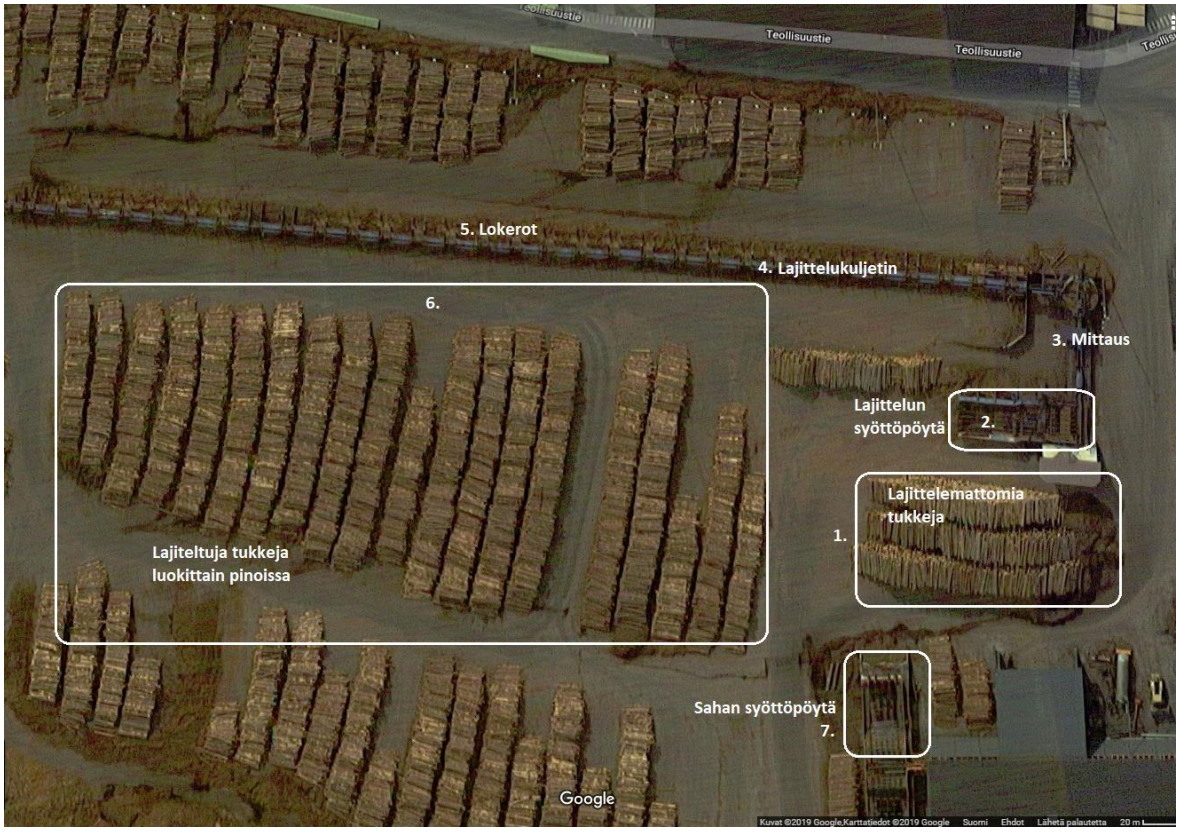
Seuraavaksi esitellään sahan korkean tason prosessi, jotta ymmärretään kokonaiskuva ympäristöstä, missä konekuskit operoivat, ja jotta sovellusta kehittäessä voidaan pitää mielessä laajennusmahdollisuudet. Kuvauksessa keskitytään erityisesti tukkilajitteluun ennen sahausta.

Puiden kaatovaiheessa metsässä metsätyökone katkoo puun. Metsätyökone laskee muun muassa puun halkaisija- ja pituustietoja, ja suorittaa katkomisen metsäyhtiön ohjeiden mukaan. Katkottu puu jää odottamaan kuljetusta tien laitaan, tai se voidaan kuljettaa muualle välivarastointiin. Puut kuljetetaan sahalle, jossa puut siirretään joko suoraan tukkilajitteluun tai mahdollisesti hetkeksi varastoon odottamaan lajittelua. Tässä vaiheessa on huomattava, että puun katkaisusta voi olla jo tässä vaiheessa kulunut viikkoja. Puutavarassa voi alkaa esiintymään laatupoikkeamia pitkän varastoinnin seurauksena esimerkiksi bakteerien tai sienien takia [2]. Sahan konekuskit tai tukkirekan kuljettajat suorittavat kuorman purun tukkilajitteluun tai tukkikentälle.

Tukkilajittelussa tukin ominaisuudet selvitetään laser- ja röntgenmittauksen avulla. Tukki luokitellaan ominaisuuksien (ainakin läpimitta, mutta usein myös laatu ja pituus) perusteella luokkiin, jotka on määritelty niin, että tukki voidaan sahata optimaalisesti lopputuotteiksi sivutuotteet minimoiden. Tukit kulkevat kuljettimella tukkimittarin läpi, jonka jälkeen tukit ohjataan lokeroihin luokittelun mukaisesti. Tarkan luokittelun takia tukkilokeroita on paljon, yleensä n. 60-80, joskus jopa yli sata.

Konekuskit siirtävät tukkeja täyttyvistä lokeroista väliaikaisvarastointiin tukkipinoihin (kutsutaan joskus myös teloiksi). Myös tukkipinot on yleensä myös määritelty lajitteluluokan mukaisesti. Tukkipinoista tukit siirretään sahaa syöttävälle syöttöpöydälle sahausohjelman mukaisesti. Kuvassa 1 havainnollisestaan tähän asti kuvattu lajitteluprosessi järjestysnumeroimalla eri vaiheet. Tukit kuoritaan, sahataan ja trimmataan asetteen mukaisesti. Valmis sahatavara lajitellaan edelleen lokeroihin dimensioiden mukaan. Laudat rimoitetaan kuivaamista varten. Kuivauksen jälkeen sahatavara pakataan ja varastoidaan toimitusta odottamaan niin, että paketti yksilöityy dimension, puulajin, laadun ja kuivausasteen mukaan. [3], [4] Lopputuotteen varastointipaikka voi olla tukkien

tavoin myös ulkona, jolloin tukkikenttäsovellusta olisi mahdollisuus hyödyntää myös tähän tarkoitukseen.



Kuva 1: Tukkilajittelu ja varastointi

2.3 Konekuskin työskentelyn nykytilanne

Konekuskin työskentely on usein kiireistä ja henkisesti kuormittavaa. Konekuskin täytyy kiinnittää tiukasti huomiota ympäristöönsä muiden ajoneuvojen, ihmisten sekä alati muuttuvan ympäristön takia. Keskittymisen herpaantuminen voi johtaa materiaali- ja henkilövahinkoihin. Konekuskien on erityisesti pidettävä huolta siitä, ettei sahausprosessissa käytettävän raaka-aineen syöttö pääse keskeytymään, mutta myös siitä, etteivät lajittelulokerot täyty sekä siitä, etteivät raaka-ainetta toimittavat ajoneuvot joudu odottamaan kuorman purkua pitkään. Työ vaatii runsaasti kokemusta, jotta se voidaan tehdä tehokkaasti.

Konekuski tarvitsee jatkuvasti informaatiota muuttuvasta työympäristöstään. Esimerkiksi lokeroihin päätyvät tukkiluokat voivat hetkellisesti tai pysyvästi vaihtaa järjestystään.

Lajiteltua puulajia voidaan vaihtaa, mikä vaikuttaa varastonsiirtoihin, sillä puulajit eivät saa mennä sekaisin. Tukkikentälle saatetaan perustaa väliaikaisia varastopaikkoja, jos esimerkiksi tietystä laadusta on hetkellisesti ylitarjontaa. Konekuskin täytyy myös jatkuvasti olla tietoinen tällä hetkellä sahatun puun luokasta, jotta hän tietää toimittaa oikeaa luokkaa sahaukseen. Lisäksi konekuskin olisi hyvä olla tietoinen saapuvista tukkitoimituksista. Nämä tiedot on ylläpidettävä useiden konekuskien välillä, sillä kentällä voi toimia useita konekuskeja samanaikaisesti. Vaikka konekuskeilla on usein tietynlainen tehtävänjako (esimerkiksi: yksi syöttää raaka-ainetta sahalle, toinen hoitaa toimitusten vastaanoton, ja lokeroiden tyhjennys jaetaan kuskien kesken), saatetaan toista auttaa väliaikaisesti, jos esimerkiksi sahan raaka-aineen syötössä on kiire. On siis tärkeää, että kaikilla on oikea informaatio tukkikentän tilanteesta reaaliaikaisesti.

Haastattelujen perusteella voidaan sanoa, että useissa paikoissa tietotekniikkaa ei hyödynnetä konekuskin työssä niin paljon kuin ehkä olisi mahdollista. Konekuskit saattavat esimerkiksi ylläpitää jokainen omaa paperista dokumenttiaan tukkikentän tilasta. Esimerkiksi kun perustetaan väliaikainen varastopaikka, konekuski saattaa ensin merkitä varastopaikan omaan dokumenttiin ja tämän jälkeen kommunikoida muutoksen radiopuhelimella muille kuskeille, jolloin muut päivittävät omaa dokumenttiaan. Myös menossa olevan sahauserään tarvittava tukkiluokka kommunikoidaan usein radioteitse. Tällainen verbaali kommunikointi jättää useita mahdollisuuksia erehdyksille ja unohduksille, sekä luo päällekkäistä työtä, kun kaikkien on replikoitava muutokset omaan dokumenttiinsa tai järjestelmäänsä.

Vaikka konekuskien prosessissa nähdään selviä tilanteita, jossa tietotekniikka voisi olla hyödyksi, ei tietotekniikan esitteleminen prosessiin ole aivan yksinkertaista. Konekuskilla ei ole aikaa operoida mitään ylimääräistä laitetta. Jos kommunikaatiota ja varastonhallintaa varten suunnitellaan ohjelmistoa, on sen toiminta oltava konekuskin kannalta suurilta osin automaattista. Konekuskin kannalta ei ole mahdollista esimerkiksi valita aina varastopaikkaa manuaalisesti, vaan varastonsiirtojen on tapahduttava suurilta osin automaattisesti sensoridataan perustuen. Kommunikaatiovirheitä voidaan kuitenkin vähentää tietotekniikalla kohtalaisen helposti, jos varastotilanteet ja erityisesti varastopaikkojen sisältämät luokat visualisoidaan reaaliajassa kuskien kesken. Erityisesti

väliaikaiset varastopaikat olisi hyvä visualisoida, ja varastopaikkojen lisääminen ja muokkaaminen olisi tehtävä helpoksi muokkaustyökaluilla, sillä kentän tilanne muuttuu jatkuvasti. Lisäksi jos sahaustilaus saataisiin viestinä kolmannen osapuolen järjestelmästä, voitaisiin se välittää automaattisesti kaikille kuskeille nähtäväksi, jolloin kommunikaatiovirheitä ja unohduksia syntyisi oletetusti vähemmän.

2.4 Kilpailijat ja tarve markkinoilla

Työn aihepiiristä tehtiin samanaikaisesti myös toinen opinnäytetyö myynnin ja markkinoinnin näkökulmasta. Työssä selvisi, että varsinkin Suomen markkinoilla löytyy tilaa uudelle tuotteelle, sillä useat kilpailevat tuotteet ovat edelleen prototyyppiasteella, tai niiden asiakaskanta on pieni. Suurin kilpailija maailmalla on tutkimuksen mukaan GPSTimber, joka niin ikään visualisoi tukkikentän tilannetta karttanäkymänä. Tiettävästi GPSTimberillä ei kuitenkaan ole Suomessa asiakkaita. Tutkimuksen mukaan Finnos Oy:n visioima tukkilajittelun, tukkikentän ja sahaa edeltävän mittarin kokonaisuus on jatkossa vahva kilpailuvaltti. [5] Osa töiden haastattelusessioista tehtiin yhteistyössä, kuitenkin eri näkökulmia painottaen. Tästä syystä töiden tausta- ja haastatteluosioissa esiintynee yhtäläisyyksiä. Työn lopputuloksen onnistumisen kannalta on tärkeää, että tuotekehitys, myynti ja markkinointi ovat kaikki perillä siitä, mitä asiakkaan vaatimukset ja tarpeet ovat ja mitä lopullinen palvelu tarjoaa.

3 TEORIA

Työn teoriaosuudessa keskityttiin erityisesti tutkimaan niiden sensoriteknologioiden perusteita, mitkä nähdään mahdollisesti hyödyllisinä työn toteutuksen kannalta. Tiettyjen teknologioiden poisrajaaminen pyritään perustelemaan aiempia tutkimuksiin perustuen. Lisäksi tutkittiin käyttöliittymäsuunnittelun heuristiikkoja sekä pohdittiin, miten reitinoptimointialgoritmeja voitaisiin hyödyntää työssä.

3.1 Paikannus

Sovellusta varten on saatava tietoa pyöräkoneiden reaaliaikaisesta sijainnista. Yhdistämällä paikkatiedon muuhun sensoridataan voidaan esimerkiksi automatisoida varaston siirtojen tarkkailu niin, ettei käyttäjän syötettä tarvittaisi normaalitilanteessa lainkaan. Paikkatiedosta voidaan päätellä, minkä varastopaikan läheisyydessä tai päällä pyöräkone milloinkin on, ja sen jälkeen pyöräkoneen kouraan liitettävistä sensoreista voidaan tarkastella, kerättiinkö tai laskettiinko kuormaa. Paikannuksen tulee olla tarkkaa, sillä varastopaikat ovat verrattain pieniä: esimerkiksi tukkipinon leveys on noin 6-7 metriä, samoin kuin vierekkäisten lokeroiden keskipisteiden välimatka. Epätarkalla paikannuksella syntyisi useammin tilanteita, joissa oltaisiin järjestelmän näkökulmasta väärällä varastopaikalla, ja näin käytettävyys heikkenisi. Tämän takia erilaisiin paikannusmenetelmiin tutustuttiin, erityisesti kiinnittäen huomiota niiden tarkkuuteen, kustannustehokkuuteen ja toimintavarmuuteen.

Kun ajatellaan yleisesti varastonhallintaa, niin huomataan, että tukkikentän varastonhallinta on luontaisinta toteuttaa satelliittipaikannuksen avulla. Satelliittipaikannus ei toimisi yhtä hyvin siinä yleisessä tilanteessa, jossa varasto sijaitsee sisätiloissa. Sisätiloissa varastonhallintaan hyödynnetään usein jotain muuta teknologiaa materiaalinseurantaan, kuten esimerkiksi RFID:tä (Radio Frequency Identification) tai viivakoodeja, ja esimerkiksi Wi-Fiin tai RFID:in perustuvaa paikannusta sisätiloissa [6]–[8]. Tukkikohtaiset RFID- tai viivakooditunnisteet loisivat kuitenkin tässä sovelluksessa lisäkustannuksia, ja niiden toimintavarmuus olisi heikko tukkikentän ankarassa ympäristössä. Satelliittipaikannustietoon ja kuormien approksimaatioon perustuvan

varastotilanteen seurannan nähdään olevan riittävä varsinkin konekuskien näkökulmasta. Niinpä tukkikohtaiset identifiointijärjestelmät jätetään tässä vaiheessa huomiotta ja keskitytään satelliittipaikannukseen perustuvan järjestelmän kehittämiseen.

GNSS (Global Navigation Satellite System) on yleisnimitys koko maapallon kattaville satelliittipaikannusjärjestelmille. Tällaisia järjestelmiä ovat esimerkiksi yhdysvaltalainen NavStar GPS (Global Positioning System), eurooppalainen Galileo, ja venäläinen GLONASS (Global'naya Navigatsionnaya Sputnikovaya Sistema). [9] Nykyään GNSS-järjestelmien paikkatietoa ei yleensä hyödynnetä sellaisenaan, vaan tietoa tarkennetaan erinäisin keinoin. Tällaiset parannusjärjestelmät usein hyödyntävät esimerkiksi tunnettujen kiintopisteiden sijaintia tai lisäsatelliittien tai muiden laitteiden sijaintia korjauslaskennassaan. SBAS (Satellite-based Augmentation Systems) on yleisnimike satelliittipohjaisille parannusjärjestelmille. Ne tuottavat korjaussignaalia perustuen tarkkaan tiedettyjen kiintopisteiden ja ylimääräisten satelliittien tuottamaan dataan, erityisesti näiden pisteiden havaitsemaan virheeseen. Kuten GNSS, SBAS on yleisnimike, jonka alle paikalliset toteutukset – kuten yhdysvaltalainen WAAS **(**Wide Area Augmentation System) ja eurooppalainen EGNOS (European Geostationary Navigation Overlay Service) – kuuluvat. [10]

RTK (Real-Time Kinematic) on tekniikka reaaliaikaisen korjausdatan tuottamiseen ja hyödyntämiseen paikannuksen tarkentamisessa. Siinä hyödynnetään kiintoaseman (engl. base station) tarkalleen tunnettua sijaintia korjausdatan laskemiseksi. Kiintoasema laskee sijaintinsa satelliittidatan perusteella, ja vertaa sitä omaan oikeaan sijaintiinsa, joka on tunnettu tai määritelty. Tämän perusteella lasketaan korjausarvoja perustuen mitatun ja tiedetyn sijainnin eroon. Korjausarvot voidaan välittää liikkuville vastaanottimille (engl. rover station). Ne korjaavat satelliittidatasta mittaamaansa sijaintia korjausdatalla. Näin saadaan mahdollisesti senttimetriluokan tarkkuudella oleva sijainti. [11] Erityisesti suhteellinen sijainti kiintoaseman ja liikkuvan vastaanottimen välillä voidaan määritellä tarkasti. Korjausdataa tulee välittää jatkuvasti liikkuvalle vastaanottimelle. Tiedon välitys voidaan hoitaa esimerkiksi radiolla tai Internetin yli (4G tai Wi-Fi).

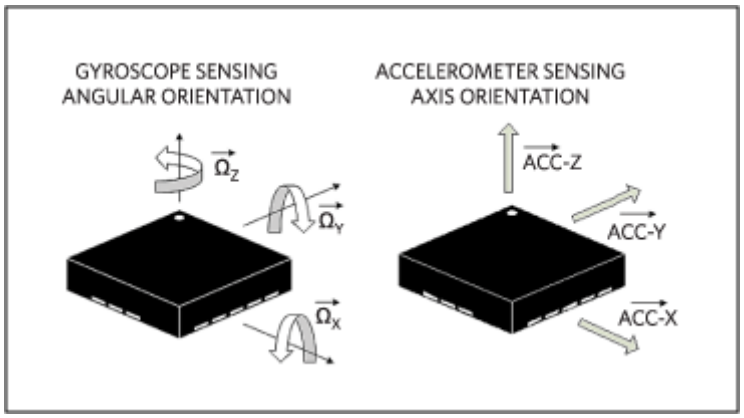
RTK-korjausdataa voidaan myös saada Internetin yli valtakunnallisista kiintopisteistä, hyödyntämällä NTRIP-protokollaa (Networked Transport of RTCM via Internet Protocol). NTRIP-korjausdatan hyödyntäminen laskee kustannuksia, kun omaa erillistä kiintoasemaa ei tarvitse hankkia. NTRIP vaatii kuitenkin Internet-yhteyden. [12], [13] Se asettaa vaatimuksia tehtaan verkkoratkaisulle ja tietoturvakäytännöille. NTRIP-korjausdatan avulla saatava tarkkuus riippuu kiintopisteen ja liikkuvan pisteen välimatkasta, minkä takia NTRIP:n avulla tehty paikannus ei ole aina yhtä tarkka kuin lähellä sijaitsevan, itse asennetun kiintoaseman avulla tehtävä paikannus [14]. Suomessa korjausdataa voi saada ilmaiseksi Maanmittauslaitoksen ylläpitämästä palvelusta [15].

Paikannuksen tarkkuutta on vaikea arvioida objektiivisesti. Kaupallisten paikannuslaitteiden tarkkuudet ilmoitetaan usein markkinointimielessä optimiolosuhteissa mitattuna. Nyrkkisääntönä voidaan kuitenkin pitää, että kuluttajakäyttöön suunnitelluilla paikannuslaitteilla, esimerkiksi älypuhelinten paikannustekniikoilla, voidaan päästä n. 1-20 metrin tarkkuuteen normaaliolosuhteissa, [16], [17], ja ulkotiloissa käyttötarkkuus voi olla 1-5 metriä. Tähänkin arvioon on useita heikentäviä tekijöitä, kuten laitteen käyttö sisällä, signaalin heijastuminen ja kahdentuminen, tai satelliitin signaalin katoaminen näköesteiden vuoksi. Myös kartta, johon signaalia verrataan, voi olla esimerkiksi puutteellisesti mallinnettu tai vanhentunut. Tällöin tuntemus paikannuksen epätarkkuudesta syntyy, vaikkei syy olisikaan itse paikanninlaitteessa. [18] Tukkikentän tapauksessa tarkkuutta heikentäviä tekijöitä ovat muun muassa ympäröivät rakennukset, joista heijastuksia ja katvealueita voi syntyä. Lisäksi liikkuva pyöräkone, johon paikannuslaite on sijoitettu, asettaa paikanninlaitteelle tiettyjä vaatimuksia. Ensinnäkin tarkkuustestit tehdään usein staattisissa pisteissä keskiarvoihin perustuen, mutta tässä tapauksessa pitää tarkastella liikkuvan vastaanottimen tarkkuutta. Toiseksi, kentällä operointi aiheuttaa tärinää ja heilahduksia, mitkä paikannuslaitteen pitäisi pystyä suodattamaan. Lisäksi rankat sääolosuhteet luovat lisävaatimuksia laitteen keston suhteen. Näin ollen esimerkiksi pelkästään tabletin sisäisen paikannuslaitteen tarkkuus ei luultavasti riitä tämän sovelluksen puitteissa, vaan on tutkittava lisälaitteita, jotka pystyvät hyödyntämään esimerkiksi RTK- tai SBAS-korjausta paikannustarkkuuden parantamiseksi. Lisäksi erillisen GPS-laitteen hyötynä olisi se, että niiden kanssa voidaan usein vaikuttaa paremmin antennin sijaintiin erillisellä antennimoduulilla.

3.2 Ajoneuvon suunta

Ajoneuvon suuntatieto nähdään hyödylliseksi sovelluksen puitteissa. Suuntatiedolla voidaan visualisoida käyttäjän ajoneuvo esimerkiksi nuolena pelkän pisteen sijasta. Suuntatieto auttaisi myös kouran sijaintitiedon määrittämisessä. Pyöräkoneet ovat suuria ajoneuvoja, mikä asettaa haasteita satelliittipaikannukselle. GPS-paikannin saatetaan esimerkiksi joutua sijoittamaan pyöräkoneen ohjaamon katolle vastaanottimen rikkoutumisen riskin pienentämiseksi, mutta varastojen siirtojen tapauksessa ohjaamon sijainti ei ole yhtä kiinnostava kuin koneen kouran sijainti. Ajoneuvon suunnan ja sijaintitiedon perusteella voitaisiin esimerkiksi laskea korjattu piste ohjaamosta, jota voitaisiin käyttää arviona kouran sijainnista. Vaihtoehtona olisi myös asentaa paikannin mahdollisimman lähelle kouraa, mutta tämä asettaa paikantimen kestolle suuremmat vaatimukset.

Älypuhelimissa ja tableteissa on usein sensorit laitteen asennon määrittelemiseksi. Asentotieto lasketaan yleensä kiihtyvyysanturin, gyroskoopin ja magnetometrin antaman tiedon perusteella. Kiihtyvyysanturin toiminta perustuu sensorin havaitseman lineaarisen kiihtyvyyden ja maan painovoiman aiheuttaman kiihtyvyyden eroon [19]. Kiihtyvyysanturilla voidaan siis mitata laitteen liikkeen suuntaa. Gyroskoopilla voidaan lisäksi mitata paremmin laitteen pyörimisliikettä ja kulmanopeutta [20]. Yhteiskäytössä nämä kaksi anturia siis täydentävät toisiaan. Kuvassa 2 on havainnollistettu sensorien erot. Asentotietoa voidaan hyödyntää sovelluksissa ohjelmointirajapintojen (engl. Application Programming Interface, API) kautta. Web-sovellukset voivat hyödyntää esimerkiksi DeviceOrientation Event Specification -määrittelyn tarjoamia tapahtumia laitteen liikkeen ja pyörimisen havainnointiin, mikäli käyttäjän selain implementoi kyseisen rajapinnan [21] [22].



Kuva 2: Gyroskoopin ja kiihtyvyysanturin erot [23]

Orientaatiotiedon hyödyntämisessä on kuitenkin myös huomioitava se, että asiakkaalla saattaa olla esimerkiksi kannettava tietokone tai tabletti valmiiksi asennettuna pyöräkoneeseen. Varsinkaan vanhemmissa kannettavissa tietokoneissa ei välttämättä ole sisäänrakennettua inertiamittausyksiköitä (engl. IMU, Inertial Measurement Unit). Tällaisessa tilanteessa olisi hyvä olla vaihtoehtoinen ratkaisu suuntatiedon selvittämiseksi. Niinpä myös erillisiä, päätelaiteriippumattomia inertiamittausyksiköitä olisi tutkittava.

3.3 Käyttöliittymäsuunnittelu ajoneuvoon

Ohjelman käyttöliittymän suunnitteluun tulee kiinnittää erityistä huomiota. Ohjelman käyttökonteksti eroaa merkittävästi normaalin web-sovelluksen käyttökontekstista, mikä täytyy ottaa huomioon käyttöliittymäsuunnittelussa. Käyttöliittymäsuunnitteluun liittyvät heuristiikat saattavat usein olla todettu toimivaksi siinä yleisessä tilanteessa, jossa sovellusta käytetään toimistokontekstissa. Sovelluksen käyttäminen ajon aikana luo useita lisätarpeita käyttöliittymän suunnitteluun, eivätkä kaikki yleiset käytettävyysheuristiikat välttämättä päde tässä ympäristössä. On siis tutkittava ohjeistuksia, jotka on todettu toimivaksi erityisesti ajoneuvokontekstissa.

Mirnig et al. (2016) ovat suorittaneet useita tutkimuksia ja luoneet niiden pohjalta useita yksinkertaisia sääntöjä, joita tulisi noudattaa ajoneuvoon suunnitellun käyttöliittymän toteutuksessa. Tutkimusten tulokset sisältävät ohjeita valikoiden monimutkaisuuden vähentämiseen (syvyys, leveys), kosketusnäytön komponenttien kosketusalaan, äänipalautteen sävyyn, järjestelmän vasteaikaan, ikonien kokoon ja värimaailmaan liittyen.

Tutkimuksen tulokset antavat ohjeistuksia myös siihen, että milloin ja miten paljon kosketusnäyttöinteraktioita ja fyysisiä kontrolleja tulisi hyödyntää. [24] Tukkikenttäsovelluksen tapauksessa kiinnitetään huomiota erityisesti värimaailmaan, kontrollien karsimiseen ja helppokäyttöisyyteen sekä vaihtoehtoisiin ohjaus- ja palautekeinoihin, erityisesti ääniohjaukseen ja -palautteeseen. Tukkikenttäsovelluksen yleisimpänä vuorovaikutuksena on kuorman nostaminen ja laskeminen varastopaikasta toiseen (esimerkiksi lokerosta pinoon, tai pinosta sahalle). Käyttäjä voisi suorittaa vuorovaikutuksen manuaalisesti, kosketusnäyttöä tai fyysistä painiketta painamalla, tai esimerkiksi ääni- tai eleohjauksella. Toinen vaihtoehto olisi automatisoida interaktio sensoridataan perustuen. Esimerkiksi jos kouran paineanturien välittämästä kuormadatasta olisi pääteltävissä, onko kourassa tällä hetkellä kuormaa vai ei, järjestelmä voisi arvioida, että varastonsiirto-operaatio on tapahtunut tietyllä hetkellä tietyssä paikassa.
Järjestelmän olisi hyvä antaa palautetta tilamuutoksista (esimerkiksi: ”kuorma laskettu pinoon 26”). Tämä voitaisiin toteuttaa esimerkiksi äänimerkein tai puhesynteesillä. On huolehdittava, että äänipalaute on miellyttävää [24], jotta käyttäjä ei ärsyynny ja eikä keskittyminen näin herpaannu. Parasta olisi, jos käyttäjä voisi vaikuttaa äänipalautteen tyyliin, tai ottaa sen pois päältä tarvittaessa.

Nielsenin (1995) tunnetussa kymmenen käytettävyysheuristiikan listassa on ohjeistuksia, joita voidaan hyödyntää pitkälti käyttökontekstiriippumattomasti. Nielsenin mukaan järjestelmän tulisi muun muassa ilmoittaa tilastaan ja sen muutoksista käyttäjälle. [25] Tukkikenttäsovelluksen tapauksessa olisi esimerkiksi annettava palautetta, kun järjestelmä uskoo kuorman noston tai laskun tapahtuneen. Nielsenin mukaan myös termien ja konseptien tulisi olla yhteneväisiä ja tuttuja käyttäjälle oikeasta maailmasta. Virheiden sattuminen on pyrittävä estämään, ja virheistä pitäisi pystyä palautumaan helposti. [25] Esimerkiksi jos järjestelmä paikannuksen epätarkkuuden vuoksi laskee kuorman väärään pinoon, niin tilanne olisi oltava helposti käyttäjän korjattavissa. Lisäksi minimalistisuus on suositeltua käyttöliittymissä [25], mutta se korostuu eritoten ajoneuvoihin tarkoitetuissa käyttöliittymissä.

Autojen käyttöliittymät perustuvat yhä enenevässä määrin kosketusnäyttöihin. Kosketusnäytöt mahdollistavat intuitiivisen ja monipuolisen käyttökokemuksen.

Kosketusnäyttö ei kuitenkaan yleensä anna tuntoaistiin perustuvaa palautetta käyttäjän toiminnoista, minkä takia kosketusnäyttökäyttöliittymää ei voida helposti operoida menettämättä keskittymistä tiehen. [24] Keskeiset toiminnot on usein toteutettu fyysisillä ohjaimilla. Lisäksi useita vaihtoehtoisia tapoja ohjata ajoneuvon toimintoja on esitelty, kuten esimerkiksi puhe- ja eleohjaus. Riener ja Rossbory (2011) havaitsivat, että Kinect-pohjainen hahmontunnistukseen (käden asento) perustuva ohjaus oli testikäyttäjien mielestä luontevaa ja he haluisivat hyödyntää sitä jatkossa vaihtoehtoja toimintojen suorittamiseen [26]. Useat modernit ajoneuvot tarjoavat nykyään myös rajallisia puheohjaustoimintoja. Puheohjauksen tehokkuutta ajoneuvoissa on tutkittu vaihtelevin lopputuloksin. Strayer et al. (2016) toteavat tutkimuksessaan, että kognitiivinen kuorma puheohjaustoimintoja suorittaessa on verrattain korkea, varsinkin vanhempien käyttäjien keskuudessa. Väärin ymmärretyt puhekomennot luovat pitkiä ajanjaksoja, jolloin kuljettaja keskittyy korjaamaan tilannetta. [27]

3.4 Työn- ja reitinoptimointi

Työn yhtenä tavoitteena on tutkia, miten varastonhallintaan liittyviä operaatioita voisi optimoida kustannuksia minimoiden. Yhtenä tapana olisi minimoida pyöräkoneiden kulkemaa matkaa työpäivän aikana. Turhia liikkeitä olisi minimoitava, jotta sahan syöttö ja muut tehtävät pysyisivät aikataulussa. Kartalla näytettävä visualisointi pinojen pituuksista ja lokeroiden täyttöasteista voisi vähentää turhaa liikkumista, sillä konekuskin ei välttämättä tarvitsee ajaa esimerkiksi toiselle puolelle lajittelukuljetinta vain nähdäkseen lokeroiden täyttöasteet. Kuskeille voitaisiin myös näyttää ohjeita siitä, mitkä lokerot ovat täyttymäisillään, ja limittää tehtävät sahan syötön kanssa. Ohjeet voitaisiin järjestää niin, että kuljettu matka minimoituu kuitenkin niin että kaikki tärkeät tehtävät tulevat tehdyksi.

Reitinoptimointi on yksi tutkituimmista tietojenkäsittelytieteen ongelmista. Tunnetuin erikoistapaus reitinoptimoinnista on niin sanottu kauppamatkustajaongelma (engl. Traveling Salesman Problem, TSP). [28], [29] Tehtävänä TSP:ssä on suunnitella reitti myyjälle niin, että kaikkien myyntikaupunkien läpi kuljetaan kerran ja palataan aloituspaikkaan niin, että kuljettu matka minimoituu. Tämän ongelman yleistys on ns. ajoneuvojen reititysongelma (engl. Vehicle Routing Problem, VRP). Tehtävänä VRP:ssä

on löytää optimaalinen tapa palvella asiakkaita (toimittaa diskreetti määrä paketteja) keskusvarastolta joukolla ajoneuvoja niin, että kustannukset minimoituvat. Lisäksi VRP-ongelmiin usein liittyy reaalimaailman tilanteissa esimerkiksi kapasiteetti- ja aikarajoituksia. [30] VRP- ja TSP-ongelmat ovat NP-vaikeita. NP-vaikeat tehtävät muuttuvat nopeasti tehtävän kasvaessa laskennallisesti erittäin raskaiksi, eikä tehtävän kasvaessa ole enää mahdollista käydä kaikkia vaihtoehtoja läpi. Vain pienet tehtävät voidaan ratkaista järkevässä ajassa optimaalisesti. Suuremmissa tehtävissä joudutaan useimmiten tyytymään hyvään arvioon, ja käyttämään esimerkiksi heuristiikkaan perustuvia algoritmeja, jotka tuottavat ei-optimaalisen mutta hyvän tuloksen rajallisessa ajassa [30].

Tukkikentän reitinoptimointi voidaan nähdä VRP-ongelmana, jossa on tiettyjä erityisvaatimuksia. Jokainen tukkiluokka on erilainen tuote. Tehtävässä ei ole varsinaisia asiakkaita, mutta esimerkiksi tukkipino ja sahan syöttöpöytä voidaan nähdä samassa roolissa: ne ikään kuin tilaavat tiettyä tuotetta. Kuljetusajoneuvoja voi olla useita, ja niillä voi olla erilaisia rooleja ja rajoitteita sille, mitä mikin ajoneuvo pystyy tekemään. Kuljetuksilla ei ole keskusvarastoa kuten klassisessa VRP:ssä, vaan tuotteet täytyy noutaa ensin noutopisteistä ja toimittaa yksi tilaus kerrallaan. Ajoneuvoilla ja varastopaikoilla on maksimikapasiteetit. Varastonsiirrolla on myös aikaikkuna, joissa tehtävä tulee suorittaa: lokero täytyy tyhjentää ennen kuin se täyttyy kokonaan ja syöttöpöytiä täytyy syöttää niin, ettei raaka-aine pääse loppumaan kesken. Näille vaatimuksille voitaisiin ainakin ennustaa arvioitu aikaikkuna perustuen historiadataan. Lisäksi ongelma on jatkuvasti muuttuva. Ei välttämättä riitä, että lasketaan optimoitu kulkemisjärjestys koko päiväksi kerran aamulla. Tilanne saattaa muuttua useita kertoja päivän aikana riippuen päivän aikana sattuvista erikoistilanteista.

Reittioptimointia varten on luotava verkko, joka kuvastaa etäisyyksiä eri varastopaikkojen ja ajoneuvojen välillä. Usein tämä verkko on avoimesti valmiiksi saatavilla, sillä reitinoptimointi halutaan usein esimerkiksi tehdä tieverkossa kaupunkien välillä. Tässä tapauksessa tilanne on toisenlainen: tukkikenttä on avoin alue, jossa on tiettyjä esteitä, ja verkko täytyy muodostaa dynaamisesti objektien välille esteet huomioiden. On siis muodostettava niin sanottu navigointiverkko (engl. navigation mesh) tukkikentälle. Tämän

jälkeen lyhin reitti navigointiverkossa kahden pisteen välillä voidaan hakea esimerkiksi Dijkstran [31, s. 638] algoritmilla tai A*-algoritmilla [32]. Navigointiverkon muodostaminen on yleinen ongelma esimerkiksi robotiikassa ja videopeleissä, joten valmiita kirjastoja, joissa optimointiin on panostettu, voidaan olettaa löytyvän.

Työn pääpaino ei kuitenkaan ole reitinoptimoinnissa. Haastatteluiden perusteella työjärjestyksen optimoinnin painoarvoa on laskettu. Haastatteluissa kävi ilmi, että konekuskit tuskin noudattaisivat ohjeita, sillä he luottavat mieluummin omaan intuitioonsa. Työohjeisiin menetettäisiin helposti luottamus, mikäli niissä olisi vähänkään virheitä tai mikäli ne sotisivat käyttäjän intuitiota vastaan. Erilaisia algoritmeja VRP-ongelman ratkaisuun ei tässä esitellä, sillä algoritmia ei ole tarkoitus ohjelmoida työssä itse. Tämän työn puitteissa tavoitteena on löytää kirjasto, joka tarjoaisi mahdollisimman hyvät työkalut ongelman ratkaisuun niin, että se tukisi yllämainittuja erityisvaatimuksia parhaiten. Ensimmäisen prototyypin on tarkoitus tuottaa dataa analysoitavaksi siitä, olisiko algoritmin suunnittelema reitti voinut parantaa työskentelyä, kun sitä verrataan ihmisen intuitioon perustuvaan työskentelyyn. Jos hyötyjä löydetään, voidaan ohjeet asettaa kuskille näkyville.

4 SUUNNITTELU JA TOTEUTUS

Aliluvuissa kuvataan teoriaan ja oletuksiin perustunut ohjelmistokehitystyö. Aivan aluksi työssä koostettiin vaatimusmäärittely, joka perustui hyvin pitkälti olettamuksiin asiakkaiden tarpeista perustuen pohjatietoihin. Tämän jälkeen oletuksia vahvistettiin ja korjattiin haastatteluilla. Haastatteluiden pohjalta ilmeni uusia vaatimuksia, joita ei osattu olettaa alun perin. Lisäksi aliluvuissa perustellaan teoriaan ja dokumentaatiokatselmointiin liittyviä teknologia- ja laitevalintoja sekä arkkitehtuurillisia valintoja, jotka myös tukisivat mahdollisimman hyvin vaatimusten toteuttamista. Lopuksi toteutetun ensimmäisen version toimintoja havainnollistetaan käyttötapauskuvausten tavoin.

4.1 Haastattelut

Suunnittelun tukena haastateltiin kuuden sahan tukkilajittelussa ja tukkikentällä toimivaa henkilökuntaa. Eri asemissa olevia henkilöitä haastateltiin, jotta saataisiin parempi käsitys erilaisista vaatimuksista eri käyttötarkoituksiin. Joukossa oli tuotannonsuunnittelijoita, johtohenkilökuntaa sekä konekuskeja. Haastatteluja varten laadittiin ohjaavia kysymyksiä, mutta sessiot pidettiin vapaamuotoisina. Näin haastateltavat pääsivät helpommin kertomaan asioita, joita he pitäisivät tärkeinä ominaisuuksina sovelluksessa. Haastatteluissa pyrittiin selvittämään työskentelyn nykytilanne, eli miten tukkikentän tilannetta ylläpidetään nykyisessä mallissa. Selvitettiin, mitä hyviä puolia ja toisaalta mitä parannettavaa nykyisessä mallissa on. Erityisesti, mikäli saha käytti jotain ohjelmistoa tukkikentän hallintaan, keskityttiin sen hyvin ja huonoihin puoliin, jottei toisteta samoja virheitä ja jotta pystytään tarjoamaan vähintään samat ominaisuudet. Lisäksi pyrittiin selvittämään, miten eri sahojen toimintamallit eroavat, jotta ohjelmiston suunnittelussa voidaan välttää vääränlaiset yleistykset. Haastattelut konekuskien kanssa tehtiin usein työn ohessa koneen kyydissä istuessa, jolloin saatiin myös kuvaa siitä, millaisessa ympäristössä sovellusta tullaan lopulta käyttämään.

Ensimmäisenä haastatellulla sahalla oli ollut testikäytössä Tieto Oyj:n ROCS-järjestelmä, mutta se oli otettu pois käytöstä järjestelmän kaatuilemisen ja virheellisten varastosiirtojen korjailun tuottaman lisätyön takia. Järjestelmän hyvinä puolina pidettiin pinojen

ikäjakaumatietoa, ja ROCS:sta oli saatavilla myös tietoa siitä, milloin puu oli kaadettu. Lisäksi videokuvaa syöttöpöydistä pidettiin tärkeänä. Mikäli uutta järjestelmää harkittaisiin, olisi sen paikkatietoratkaisun oltava tarkka, ja sen tulisi toimia myös tilanteessa, jossa pinoja täytetään sivusta tukkikurottajalla. Haastateltavat konekuskit kertoivat työskentelevänsä kokemuksen tuottaman tiedon perusteella. Nykyään tietoa varastopaikoista pidetään yllä paperidokumentissa, ja tiedonsiirto kommunikoidaan radiopuhelimella. Kuskeilla on selvä työnjako, jossa toinen syöttää sahaa ja hoitaa toisen puolen lokerot, ja toinen tyhjentää autokuormia ja tyhjentää toisen puolen lokerot. Käytössä olevissa pyöräkoneissa ei ole vaakaa asennettuna. Tiedot sahan tilauksista saadaan SAP-toiminnanohjausjärjestelmästä, jota voidaan tarvittaessa käyttää pyöräkoneessa olevalta tietokoneelta.

Toinen haastateltava saha käytti ROCS-järjestelmää aktiivisesti. Se koettiin hyödylliseksi ohjeiden antajaksi, varsinkin jos konekuski on uusi, eikä tämä muista pinojen paikkoja ulkoa. Sovellus ei näyttää pinojen täyttöasteet, muttei lokeroiden täyttöasteita. Videokuvaa syöttöpöydistä pidettiin myös tärkeänä. ROCS tunnistaa varastonsiirrot automaattisesti kuormatietoon perustuen, sillä tällä sahalla pyöräkoneisiin oli asennettu lisävarustevaa'at. Kuormantunnistuksessa oli ilmennyt pieniä ongelmia, sillä järjestelmä saattaa päätellä virheellisen varastonsiirron esimerkiksi töyssyyn ajaessa. Konekuskien mielestä optimoituja reittiohjeita olisi turha antaa, sillä he luottavat mieluummin omaan kokemukseensa. Järjestelmän käyttöönottovaiheessa ei saatu riittävästi tukea. Kuskeilla oli samanlainen työnjako kuin ensimmäisellä sahalla.

Kolmas haastateltava saha ei ole aikaisemmin käyttänyt sovellusta tukkikentän hallintaan, vaan tiedot ylläpidetään paperidokumentilla ja muutokset kommunikoidaan suullisesti. Vaikka malli on toimiva, kommunikaatiovirheitä tapahtuu ja tiedon on joskus puutteellista tai se ei ole ajan tasalla. Mikäli tukkikenttäsovellusta harkittaisiin käyttöönotettavaksi, tulisi sen olla helposti kustomoitavissa. Esimerkiksi telojen paikat voivat siirtyä, jolloin käyttäjän on helposti pystyttävä muokkaamaan niitä. Haastattelussa tuli myös ilmi ideoita tukkikentän sijoittelun optimoinnista. Tilastotiedon perusteella voitaisiin ehdottaa varastopaikkojen perustamista niin, että usein sahattavat tukkiluokat sijaitsisivat lähimpänä

sahauspöytää ottaen huomioon myös mahdollisesti tiedossa olevan lähitulevaisuuden sahaussuunnitelman.

Neljäs haastateltava saha käytti tiedon ylläpitoon paperidokumenttia. Tällä sahalla töitä tehdään vain yhdessä vuorossa, joten tiedonsiirrolle kuskien kesken ei ole varsinaista tarvetta, jos työtehtävät pidetään samana. Yleinen näkemys oli, että näin pienellä sahalla kentän hallinta onnistuu hyvin manuaalisestikin, mutta hyötyjä nähtiin isommille sahoille.

Viides haastateltava saha käytti paperidokumentteja, jotka synkronoitiin kuskien kesken kommunikoimalla. Sahalla oli käytössä yksi kurottaja, joten sivuttaistyöskentelyn huomioon ottaminen olisi tälläkin sahalla välttämätöntä. Pyöräkoneissa ei ole vaakaa vakiona. Työnjako on vastaavanlainen kuin muilla sahoilla. Erityisesti lajinvaihdossa mahdollisien jaettujen pinojen tärkeyttä korostettiin. Saha on myös kiinnostunut kuljetun matkan optimoinnista ja kartan helposta muokattavuudesta ja dynaamisuudesta.

Kuudennen sahan vaatimukset vastasivat paljolti viidennettä. Pyöräkoneissa ei ole tällä sahalla vaakaa asennettuna. Myös tämän sahan konekuskit kommentoivat, että optimoiduista työohjeista ei olisi paljoakaan hyötyä. Puulajin vaihto lajittelussa on tällä sahalla samanlainen haaste kuin edellisellä. Edellisistä haastatteluista poiketen oli tässä haastattelussa mukana myös useita tuotannonsuunnittelijoita. Erityisesti kiinnostuksen kohteena olivat erilaiset varastoraportit ja puiden ikäjakauma. Ideoita tuli myös turvallisuuden lisäämiseen niin, että kartalla näkyisi myös tukkikentän läheisyydessä olevat henkilöt. Haastattelussa oli myös henkilöitä, jotka ovat aiemmin käyttäneet ROCS-järjestelmää, ja he näkivät paikkatiedon tuoman lisähyödyn, koska kartalta pystyy nopeasti tarkistamaan sijaintinsa epävarmoissa tilanteissa. Sahalla oli käytössä taulukkolaskentapohjainen järjestelmä pinojen kartoittamiseksi. Nykyinen järjestelmä koettiin toimivaksi.

Haastatteluita analysoimalla päätettiin korostaa nosto- ja laskutoimenpiteiden helppoutta ja tarkkuutta. Selvisi, että useassa paikassa pyöräkoneesta ei ole saatavissa vaakatietoa vakiona, joten olisi tarjottava toisenlainen helppo tapa havaita tai syöttää varastonsiirrot, esimerkiksi painonapin avulla. Oli myös suunniteltava keino tukea tukkikurottajilla

mahdollista sivuttaistyöskentelyä (tukkien nostaminen pinon keskelle sivusta), joko GPS-antennin sijoittamisella kouraan, tai laskemalla korjattu kouran sijainti hyödyntäen suuntatietoa. Haastatteluiden pohjalta päätettiin myös laskea työjärjestysoptimoinnin prioriteettia. Huomattiin, että jotkut sahat käyttävät pinojen määritykseen tukkiluokkaa, mutta toiset taas puhuvat siitä, mitä lokeroa on viety mihinkin pinoon. Tätä varten suunniteltiin vaihtoehtoiset nimeämiskäytännöt pinoille. Videokuvan tärkeyden takia IP-kameranäkymä lisättiin vaatimuksiin. Pinojen dynaamisuus ja muokattavuus sai paljon erityisvaatimuksia, jotka eivät olleet tiedossa ennen haastatteluja. Niinpä esimerkiksi pinon jakaminen useaan osaan tukkiluokittain mahdollistettiin.

4.2 Teknologioiden valinta

Alikappaleissa perustellaan sovellusta varten tehtyjä teknologiavalintoja.

4.2.1 Web vs. native

Konekuskit tulevat käyttämään sovellusta mobiilisti, joten projektin alussa oli valittava mobiilikäyttöliittymän toteutustekniikka. Vaihtoehtoja on nykyään monia. Ensinnäkin voidaan luoda natiivi applikaatio jokaista laitetta kohden. Tämä tarkoittaa erillisen sovelluksen luomista sekä Android-, iOS- sekä Windows-ympäristöihin. Tämä mahdollistaa laitteen ominaisuuksien täyden hyödyntämisen, mutta haittapuolena on se, ettei koodia voida helposti jakaa eri alustojen välillä. Niinpä sama sovellus voidaan joutua käytännössä ohjelmoimaan jokaiselle alustalle erikseen, vaikka niiden toiminnallisuudet olisivat suurilta osin samat. Täysin natiivit applikaatiot ohjelmoidaan Androidille Javalla ja iOS:lle Swiftillä.

Natiivin sovelluksen täysi vastakohta on luoda web-sovellus, jolloin tuettujen laitteiden kirjo on suurin mahdollinen (mikä tahansa laite, joka vain tukee web-selainta), ja sama koodipohja toimii lähes poikkeuksetta jokaisella laitealustalla. Web-ohjelmoinnilla ei kuitenkaan päästä käsiksi laitteiden erikoisominaisuuksiin yhtä hyvin kuin natiivilla applikaatiolla. Nykyiset web-standardit kuitenkin mahdollistavat esimerkiksi kameran, paikkatiedon, laitteen orientaation käyttämisen alustalla kuin alustalla, mikäli kyseiset sensorit tai laitteet vain löytyvät laitteesta [33].

Toteutustavaksi valittiin täysin web-pohjainen käyttöliittymä. On totta, että varastonhallintasovellus tarvitsee laitetietoa, eritoten tarkkaa paikkatietoa varastonsiirtojen havaitsemiseksi. Nykyaikaiset web-standardit kuitenkin mahdollistavat pääsyn kaikkeen tässä kontekstissa tarvittavaan laitetietoon. Web-sovellus ei sido käyttäjää tiettyyn laitevalmistajaan tai käyttöjärjestelmään, jolloin voidaan hyödyntää pyöräkoneissa jo mahdollisesti olemassa olevia laitteita. Myös web-käyttöliittymän mahdollistama nopea kehitys ja yrityksestä löytyvä aiempi osaaminen web-kehityksestä olivat painottavia tekijöitä valintaa tehdessä. Finnos Oy:n tukkilajittelun käyttöliittymä on toteutettu myös selainpohjaisena, joten koodia pystyttiin jakamaan myös joiltain osin eri sovellusten käyttöliittymien välillä.

4.2.2 Ohjelmointikielet, ohjelmistokehykset ja tietokanta

Kun päätös toteuttaa sovellus selainpohjaisena oli tehty, oli päätettävä seuraavaksi mitä apuvälineitä, kuten ohjelmistokehyksiä, kirjastoja ja ohjelmoinnin työkaluja käytettäisiin. Hyvin nopeasti front-end-kehykseksi valikoitui Angular [34], jolloin edelleen mahdollistettiin aiemman web-pohjaisen sovelluksen käyttöliittymäkomponenttien osittainen uudelleenkäyttö. Muita suosittuja kehyksiä ja kirjastoja ovat muun muassa React [35] ja VueJS [36], mutta näitä ei pidetty tässä tapauksessa vaihtoehtoina. Haluttiin, että kuka tahansa, joka yrityksessä on aikaisemmin työskennellyt tukkilajittelukäyttöliittymän kanssa voisi myös jatkossa helposti ylläpitää tukkikenttäsovellusta. Samoin perustein jatkettiin C# Web API:n käyttöä palvelimella ja MongoDB:ta tietokantana.

4.2.3 Karttakirjasto

Seuraavaksi tutkittiin erilaisia vaihtoehtoja karttakirjastolle, jota käytettäisiin sovelluksen päänäkymäksi visioidun karttanäkymän kehittämiseen. Tämä valinta ei ollut yhtä suoraviivainen kuin ylempänä mainitut alustan ja kehyksen valinnat, sillä tukkilajittelusovelluksessa ei ollut työn aloitushetkellä varsinaisia karttatoimintoja.

Google Maps JavaScript API on Googlen tarjoama karttakirjasto. Sen vahvuuksina ovat muun muassa laajat geolokaatio- ja navigointiominaisuudet, mutta niiden hyödyntäminen tuskin olisi tukkikenttäsovelluksen puitteissa mahdollista, erityistarpeiden takia. Google

Mapsin räätälöinti ja verkottoman käytön mahdollisuudet olivat dokumentaation perusteella puutteelliset. Lisäksi palvelun maksulliset ominaisuudet nähtiin heikkoutena. Google Maps:n tarkastelu jätettiin pois dokumentaatiokatsauksen perusteella.

Leaflet [37] on suosittu avoimen lähdekoodin karttakirjasto selainsovelluksiin. Sen hyviksi puoliksi nähtiin kirjaston helppokäyttöinen ohjelmointirajapinta, helppo räätälöitävyys sekä hyvä Angular-tuki. Heikkoutena ilmeni esimerkiksi kartan selaamisen rajoitteet, sillä karttaa ei pystynyt pyörittämään tai kallistamaan. Leaflet ei myöskään tue vektorikarttoja vakiona [37].

Mapbox-kartat pohjautuvat rasterikuvien sijaan vektoripohjaisten karttakuvien renderöintiin, mikä tekee kartan operoinnista sulavaa. Vektorikarttojen renderöinti tapahtuu WebGL-kirjastolla, mikä mahdollistaa näytönohjainkiihdytetyn grafiikan piirron. Mapbox tarjoaa eri alustoille natiivikirjastoja JavaScript-pohjaisen (Mapbox GL JS) kirjaston lisäksi. Mapbox GL JS -kirjaston käyttö itsessään on ilmaista (lisenssi BSD 3-Clause), mutta Mapbox:n karttadatan käyttö on maksullista. Tätä varten oli tutkittava miten tarjota karttapohjat kirjastolle niin, että niitä voitaisiin käyttää myös verkottomassa tilassa, samalla mahdollistaen ilmaisten tai kertamaksullisten karttapohjien käytön.

Mapbox GL JS -kirjaston käyttö Angularin kanssa oli helppoa kolmannen osapuolen kirjaston avulla [38], joka käärii Mapbox GL JS:n puhtaat JavaScript-toiminnallisuudet helppokäyttöisiksi Angular-komponenteiksi. Lisäksi hyviksi puoliksi katsottiin sen käytön sulavuus ja helppo kustomointi. Esimerkiksi kartan värit ovat täysin kustomoitavissa, ja kuvioiden piirtäminen kartan päälle oli helppoa kerrosten (engl. map layer) avulla. Mapbox mahdollisti myös kartan pyörittämisen ja katselukulman säätämisen, joka nähtiin tarpeelliseksi navigointitilanteessa. Tällöin voitaisiin toteuttaa tila, jossa kamera seuraa käyttäjän orientaatiota yläviistosta, monien muiden navigointisovellusten tapaan.

Dokumentaatiota tutkimalla päätyttiin kokeilemaan Leaflet- ja Mapbox-kirjastoja. Leaflet-kirjasto oli monilta osin sopiva tarkoitukseen, mutta esimerkiksi puuttuvien kartan pyöritys- ja kallistusominaisuuksien ja rasterikarttojen heikon suorituskyvyn takia

päädyttiin lopulta Mapbox GL JS -kirjastoon. Lisäksi otettiin käyttöön ngx-mapbox-gl-kirjasto [38], joka helpotti kirjaston käyttöä entisestään Angularin kanssa.

4.2.4 GeoJSON-standardi

Käyttämällä karttakirjastoa pystytään visualisoimaan räätälöityjä objekteja kuvioiden avulla karttapohjan päälle. Kuviodatan tallennukseen ja esittämiseen käytettiin GeoJSON-formaattia. Se on avoin standardi maantieteellisen datan tietorakenteelle. [39]

Esimerkki GeoJSON-datasta [39]:

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {"name": "Ajoneuvo 1"},
      "geometry": {
        "type": "Point",
        "coordinates": [28.167341351509094, 61.06607355274095]
      }
    },
    {
      "type": "Feature",
      "properties": {"name": "Pino 1"},,
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [28.167357444763183, 61.06604889729469],
            [28.16745400428772, 61.06602164651599],
            [28.16745400428772, 61.06608004101309],
            [28.167357444763183, 61.06604889729469]
          ]
        ]
      }
    }
  ]
}
```

Feature-objekti sisältää geometriatiedon lisäksi vapaavalintaista metadataa properties-attribuutissa. FeatureCollection sisältää kokoelman Feature-objekteja. Geometrialla on

tyyppi, joka voi olla Point, LineString, Polygon, MultiPoint, MultiLineString tai MultiPolygon. Tyypin mukaan coordinates-attribuutin rakenne vaihtelee. Esimerkiksi pisteellä koordinaatit esitetään kaksialkioisena listana, jossa esiintyy ensin longitudi ja toisena latitudi. LineString on lista tällaisia pisteitä. Polygon on kuin LineString, mutta viimeinen piste on sama kuin aloituspiste, ja kuvio voi sisältää myös reikiä. Tukkikenttäsovelluksessa Point voisi esimerkiksi olla ajoneuvon tai lokeron sijainti, LineString reitti lokerolta pinoon, ja Polygon tukkipinon alue.

GeoJSON-formaattiin päädyttiin, koska se on tuettu monissa kolmannen osapuolen kirjastoissa. Monet JavaScript-karttakirjastot, kuten Leaflet ja Mapbox GL JS, tukevat kuvioiden piirtoa karttakerroksiin GeoJSON-muodossa. Lisäksi MongoDB, jota tässä ratkaisussa käytettiin tietokantana, tukee indeksoituja paikkatietohakuja, jos tieto on tallennettu GeoJSON-muodossa. Tällaisia hakuja voivat olla esimerkiksi:

- Etsi pisteet tietyllä etäisyydellä halutusta pisteestä
- Järjestä dokumentit etäisyyden mukaan halutusta pisteestä
- Etsi pisteet, jotka ovat monikulmion sisällä

Lisäksi Turf.js -kirjastoa [40] käytettiin GeoJSON-datan manipulointiin ja erilaisiin laskutoimenpiteisiin, kuten esimerkiksi monikulmioiden massakeskipisteiden hakuun ja etäisyyslaskentaan.

4.2.5 Reitinoptimointikirjasto

Googlen OR-tools on tarkoitettu yleisesti erilaisten kombinatoristen optimointitehtävien ratkaisuun. Sen tarjonta ei rajoitu pelkästään ajoneuvojen reittiongelmiin, vaan se tarjoaa myös työkaluja lineaari- ja kokonaislukuoptimointiin, pakkaustehtäviin sekä muihin graafitehtäviin reititystehtävien lisäksi, kuten esimerkiksi minimi- ja maksimivirtausten laskemiseen. Kirjasto on kirjoitettu C++:lla, mutta se tarjoaa käärityt versiot (engl. wrapper) Pythonille, Javalle ja C#:lle. [41] Dokumentaatiossa oli useita laajoja esimerkkejä kirjaston käytöstä, mutta kaikki esimerkit olivat Pythonille. Kirjaston käyttöönotto oli sulavaa Pythonilla. Käyttöönotto C#:lla oli vaikeampaa ja vaati ylimääräisten ohjelmistojen asentamista. Positiivisena seikkana nähtiin oletetusti hyvä suorituskyky, sillä kirjasto on ohjelmoitu C++:lla. Koska kirjasto on tarkoitettu yleisesti optimointiongelmiin, oli kirjaston käyttö ja sen käyttämä termistö enemmänkin matemaattinen eikä niinkään käytännönläheinen. Vaikka kirjasto esimerkiksi tukee reittioptimoinnin aikaikkunoita,

niiden implementointi vaati verrattain paljon töitä käsin. Tämä lisäsi kirjaston käytön kompleksisuutta mutta se antoi toisaalta mahdollisuuksia vaikuttaa ja tehdä tehtävän ratkaisuun muutoksia.

Jsprit on ainoastaan reittioptimointiin suunnattu kirjasto. Kirjasto on kirjoitettu Javalla. Se tukee esimerkiksi aikaikkunoita, nouda-ja-toimita-ongelmia sekä erilaisia ajoneuvoja [42]. Se ratkaisee monimutkaisia VRP-ongelmia ns. ruin-and-recreate-periaatteella [43]. Ohjelmointirajapinta Jspritissä on hyvin intuitiivinen: Vehicle-luokan instanssille voidaan antaa muun muassa sijainti, tyyppi ja maksimikapasiteetti. Tehtävät luodaan Service-luokan instansseina, joille voidaan antaa kapasiteettivaatimus ja sijainti. Heikkoutena kirjastossa nähtiin kirjaston toteutus Javalla. Koska koodipohja palvelimella on muuten toteutettu C#:lla, tulisi Jspritin integroiminen vaatimaan lisätyötä.

Esimerkki triviaalin VRP-ongelman ratkaisusta Jspritillä [42]:

```
VehicleTypeImpl.Builder vehicleTypeBuilder =
VehicleTypeImpl.Builder.newInstance("vehicleType").addCapacityDimension(WEIG
HT_INDEX, 2);
VehicleType vehicleType = vehicleTypeBuilder.build();
Builder vehicleBuilder = VehicleImpl.Builder.newInstance("vehicle");
vehicleBuilder.setStartLocation(Location.newInstance(10, 10));
vehicleBuilder.setType(vehicleType);
VehicleImpl vehicle = vehicleBuilder.build();
Service service1 =
Service.Builder.newInstance("1").addSizeDimension(WEIGHT_INDEX,
1).setLocation(Location.newInstance(5, 7)).build();
VehicleRoutingProblem.Builder vrpBuilder =
VehicleRoutingProblem.Builder.newInstance();
VehicleRoutingProblem problem =
vrpBuilder.addVehicle(vehicle).addJob(service1).build();
VehicleRoutingAlgorithm algorithm = Jsprit.createAlgorithm(problem);
Collection<VehicleRoutingProblemSolution> solutions =
algorithm.searchSolutions();
VehicleRoutingProblemSolution bestSolution = Solutions.bestOf(solutions);
new Plotter(problem, bestSolution).plot("output/plot.png","simple example");
```

Optaplanner on Jspritin tavoin Javalla toteutettu optimointikirjasto. Sen tarjoamat työkalut eivät rajoitu pelkästään VRP-ongelmiin, vaan se tarjoaa työkaluja muun muassa

työtehtävien toimeksiantoon ja aikataulutusongelmiin. Optaplanner asettuu siis monipuolisuudessaan Jspritin ja Google OR-toolsin välimaastoon. Dokumentaatio on koodiesimerkkien kannalta puutteellista, ja dokumentaatiota katselmoimalla nähtiin, että kirjaston käyttö olisi työläämpää kuin esimerkiksi Jspritin.

Tämän työn puitteissa päädyttiin käyttämään Jsprit-kirjastoa sen helppokäyttöisyyden ja nopean käyttöönoton takia. Muut tutkitut kirjastot olivat monipuolisempia tarjonnaltaan, mutta niillä alkuun pääseminen oli suuremman työn takana. Myös dokumentaatiot OR-toolsissa ja Optaplannerissa nähtiin osittain puutteellisina tai ainakin vaikeaselkoisimpina. Monipuolisemmat kirjastot saattavat osoittautua hyödylliseksi jatkossa, mikäli reitin- ja tehtävienoptimointiin päädytään panostamaan laajemmin. Niinpä kirjaston käyttö suunniteltiin niin, että se voidaan jatkossa vaihtaa toiseen mahdollisimman vaivattomasti. Mikropalvelu sopi tähän tarkoitukseen hyvin, sillä mikropalvelu voidaan vaihtaa käyttämään jopa toista ohjelmointikieltä ilman että kutsuvaan palveluun täytyy tehdä muutoksia.

4.3 Laitteisto

Teoriaosuuden satelliittipaikannuksen katselmoinnin pohjalta päädyttiin tutkimaan RTK-korjauksia tukevia laitteita. Perinteisesti RTK-laitteistoa on käytetty lähinnä suurta tarkkuutta vaativaan maanmittaukseen. Yksi tunnetuimmista ammattikäyttöön tarkoitettujen GPS-vastaanottimien valmistajista on Trimble. Trimblen halvimmat RTK-korjauksia tukevat laitteet maksavat noin 3500€ (R1) [44], ja astetta monipuolisemmat mallit (R2) $10000 [45]. On huomattava myös, että jokaisen koneeseen asennettavan vastaanottimen lisäksi RTK-korjaus vaatii yhden kiintoaseman per toimipiste, tai esimerkiksi Trimnet-korjausdatapalvelun lisenssin.

Uudempi tulokas RTK-laitteiden valmistajana on Emlid, joka suunnittelee RTK-laitteita erityisesti droneissa käytettäväksi. Edullisin RTK-korjauksia tukeva laite Emlidiltä on Reach M+, jonka hinta on 265$. Hinnat on saatu alhaisiksi käyttämällä avoimen lähdekoodin RTKLIB-kirjastoa korjauksien laskennassa [46], [47]. Huomattavaa on, että laitteet tarvitsevat lisäksi esimerkiksi LoRa-radion (Long Range) kiintoaseman ja liikkuvan vastaanottimen väliseen kommunikointiin, sekä mahdollisesti erillisen GNSS-antennin.

Ensimmäisessä testissä päädyttiin kokeilemaan Emlid Reach M+ ja Reach RS+ vastaanottimia. Mikäli niiden laatu ja tarkkuus osoittautuisi riittäväksi, voitaisiin asiakkaalle tarjota houkuttelevalla hinnalla tarkkaa paikannusteknologiaa. Trimble, Emlid ja useat muut valmistajat tarjoavat sijaintitiedon ulostulon NMEA-formaatissa, joten laitteet voidaan lopulta valita mielivaltaisesti asiakkaan vaatimusten mukaan, ilman ohjelmistomuutoksia.

Myös inertiamittausyksiköiden tarjontaa tutkittiin. Ensimmäisissä testeissä käytettiin Getac F110:n sisäisiä sensoreita suunnan määrittelemiseksi, mutta ne todettiin nopeasti epätarkoiksi. Ulkoista inertiamittausyksikköä haluttiin tutkia myös sen takia, että voitaisiin tarjota ratkaisu myös silloin, kun ajoneuvoissa on jo olemassa tietokoneet, jotka eivät sisällä antureita. Tämän työn aikataulussa ei kuitenkaan keretty testaamaan useampia IMU-yksiköitä, vaan päädyttiin nopealla katselmoinnilla Bosch BNO055 inertiamittausyksikköön. Valinta oli aikataulujen puitteissa helppo, sillä BNO055 oli ensimmäinen vastaantuleva sensori, joka suoritti valmiiksi mikroprosessorilla magnetometrin, kiihtyvyysanturin ja gyroskoopin datan fuusiota. Anturi antaakin ulostulonaan absoluuttisen orientaation asteina, jolloin ohjelmistokehitystyö sensorin hyödyntämiseksi jäi kevyeksi.

Lisäksi työn loppupuolella ilmeni tarve ulkoiselle painonapille. Vaatimuksina oli pieni koko ja helppo sijoitettavuus ohjauslaitteisiin. Erilaisia älynappeja katselmoitiin. Flic-älypainike valittiin pienen kokonsa ja monipuolisten kirjastojensa takia. Nappi voidaan kiinnittää liimapinnalla ohjauslaitteisiin ja yhdistää Bluetoothilla päätelaitteeseen. Muita vastaavia älynappeja ovat mm. Logitech POP ja Amazon Echo Button, jotka suljettiin pois vaihtoehdoista kokonsa takia.

4.4 Arkkitehtuuri

Sovelluksen arkkitehtuurin voidaan sanoa noudattelevan mikropalveluarkkitehtuuria (engl. microservices). Mikropalveluarkkitehtuurissa ohjelmisto koostuu pienemmistä, itsenäisistä, modulaarisista komponenteista, jotka koostavat yhdessä applikaation [48], [49]. Mikropalvelut julkaista jokainen erikseen omana prosessinaan, mikä tekee ohjelmistojen

päivityksestä ketterää, sillä käyttökatkot voidaan rajata vain pieneen osaan sovellusta tai ne voidaan eliminoida kokonaan. Mikropalvelut keskustelevat usein jonkin kevyen protokollan avulla, kuten HTTP:lla. Koska kommunikaatio toteutetaan HTTP:llä, palvelut voidaan toteuttaa tehtävään sopivalla ohjelmointikielellä. Lisäksi kommunikaatio asiakasohjelman ja palvelinohjelman välillä kuin myös palvelinten keskinäinen kommunikaatio voidaan toteuttaa helposti samoja tekniikoita käyttäen. Mikropalveluiden kehittäminen voidaan jakaa helposti ryhmän jäsenten kesken niin, että jokainen voi kehittää omaa osa-aluettaan riippumatta muista, kunhan ulkoinen ohjelmointirajapinta säilytetään. Mikropalveluarkkitehtuuri kehottaa myös käyttämään palvelulle sopivaa tietokantaratkaisua. Jokaisella palvelulla voi siis ääritapauksessa olla oma tietokantansa. Mikropalveluarkkitehtuuria on myös helpompi skaalata. Esimerkiksi jos mikropalvelu on tilaton, niin palvelua voidaan skaalata horisontaalisesti julkaisemalla palvelusta useita instansseja ja tasaamalla kuorma (engl. load balance) instanssien välillä [48], [49]

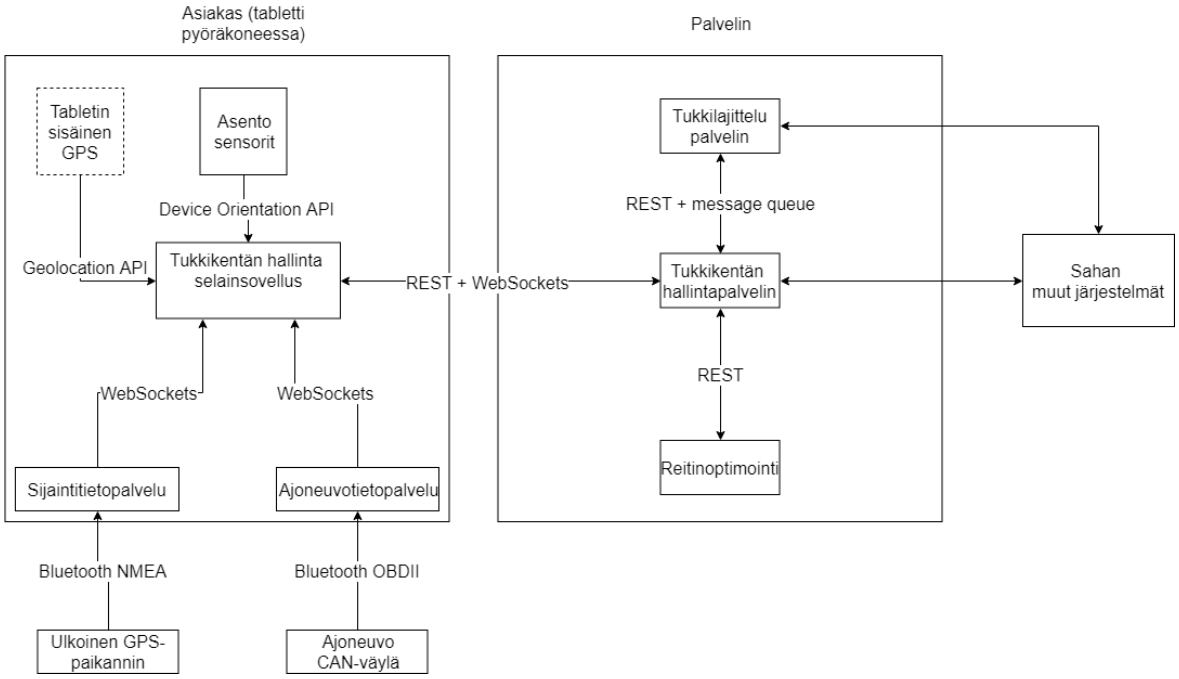
Vaikka mikropalveluarkkitehtuuri ratkaisee useita monoliittiarkkitehtuurin heikkouksia, se myös tuo uusia haasteita. Kun palveluiden määrä kasvaa, on syytä suunnitella mahdollisimman automatisoitu testaus- ja julkaisuprosessi. Muutokset ulkoisiin rajapintoihin on suunniteltava huolellisesti niin, ettei vanhaa rajapintaa käyttävä palvelu hajoa muokkauksen seurauksena. Mikropalveluarkkitehtuuri antaa paljon vapautta esimerkiksi siihen, millä työkaluilla mikäkin palvelu toteutetaan. Tämä voi kontrolloimattomana muodostua ongelmaksi. Työtehtävien vaihtaminen esimerkiksi yrityksen sisällä voi hankaloitua, jos ryhmän jäsenillä ei ole laajaa osaamista eri ohjelmointikielistä ja työkaluista. [49] Mikropalveluarkkitehtuuriin kannattaakin siis suhtautua kriittisesti ja hyödyntää sitä mittakaavassa mikä on kyseiselle projektille järkevää. Pienempi projekti kannattaa esimerkiksi jakaa vain muutamaan palveluun niin, ettei ylimääräistä monimutkaisuutta synny.

Tukkikentän hallintasovellus eriytettiin varsinaisesta tukkilajittelusovelluksesta. Tämä mahdollistaa tukkikenttäsovelluksen kehittämisen erikseen, kunhan pieni määrä rajapintoja näiden kahden sovelluksen välillä säilyy samana. Näin palvelut voidaan myös tarjota mahdollisuuksien mukaan toisistaan riippumatta: asiakkaalle voidaan tarjota tukkikenttäsovellus lisäpalveluna, ja toisaalta tukkikenttäsovellus voidaan tarjota

asiakkaalle ilman tukkilajittelukokonaisuutta, mikäli integraatio kolmannen osapuolen järjestelmään voidaan toteuttaa.

4.5 Palveluiden välinen kommunikaatio

Palveluiden ja laitteiden välinen kommunikointi on kuvattu korkealla abstraktiotasolla kuvassa 3.



Kuva 3: Palvelut, laitteet ja kommunikointi

Tukkilajittelusovelluksen palvelin julkaisee viestin tukin mittaustapahtumasta viestijonoon (engl. message queue). Tukkikenttäsovelluksen palvelin rekisteröityy viestijonoon kuuntelemaan mittaustapahtumaviestiä. Viesti sisältää tukin avainominaisuudet, kuten dimensiot, lokeron ja tukkiluokan, jolloin varastonhallinta voi suorittaa tukin varastonsiirron lajittelupöydästä oikeaan lokeroon. Tukkikenttäsovellus voi myös kutsua tukkilajittelusovelluksen REST-rajapintaa. Lajittelusovellukselta voidaan esimerkiksi pyytää tietoa tukkiluokkajakaumista, ja tätä tietoa voidaan hyödyntää tehtävien optimoinnissa.

Tukkikenttäsovelluksen palvelin tarjoaa Angular-sovelluksen asiakastablettien selaimille. Angular-sovellus kommunikoi palvelimen kanssa REST-kutsuilla ja WebSocketseilla. Esimerkiksi varastonsiirron yhteydessä palvelin julkaisee WebSocket-viestin, jolloin kaikkien asiakasohjelmien karttanäkymät päivittyvät reaaliajassa visualisoiden reaaliaikaisen varastotilanteen. Samoin, kun asiakasohjelma havaitsee sijaintitiedon muutoksen, se kommunikoi uuden sijaintitiedon ensin palvelimelle, joka puolestaan julkaisee WebSocket-viestin muille asiakasohjelmille ajoneuvon sijaintimuutoksesta. Näin kartalla voidaan seurata kaikkien ajoneuvojen liikettä reaaliajassa.

Paikkatietoa sovellusta varten voidaan saada joko tabletin omasta sisäänrakennetusta paikantimesta (mikäli tarkkuus todetaan riittäväksi) tai erillisestä vastaanottimesta. Paikkatieto voidaan saada sisäänrakennetun paikantimen tapauksessa suoraan selaimen Geolocation API:sta. Erillistä vastaanotinta varten luotiin mikropalvelu. Palvelu toteutettiin NodeJS:llä käyttöjärjestelmäriippumattomuuden takia. Palvelu asennetaan tausta-ajettavaksi tablettiin. Palvelu lukee COM-portista NMEA 0183 -standardin [50] mukaista dataa. NMEA 0183 (National Marine Electronics Association) on yleinen viestirakenne paikkatiedon välittämiseen, ja lähes jokainen ammattilaiskäyttöön suunniteltu satelliittipaikannin tukee tätä viestimuotoa dataviennissään. Palvelu jäsentää NMEA-viestejä ja tarvittavat tiedot lähetetään WebSocketsilla edelleen Angular-asiakasohjelmaan. Näin esimerkiksi Bluetoothilla tai USB:lla yhdistettyä ulkoista satelliittipaikanninta voidaan hyödyntää navigoinnissa.

Myös ajoneuvon kuormatiedon lukeminen suunniteltiin toteutettavaksi samalla tavalla kuin ulkoisen paikannuslaitteen kommunikointi. Ajoneuvon CAN-väylään on usein kytkettävissä Bluetoothia tukeva lukijalaite, joka voisi edelleen lähettää tiedot tablettiin. CAN-väylän datasta voitaisiin jäsentää kuormatieto ajoneuvokohtaisesti, jonka jälkeen kuorman muutos voidaan kommunikoida WebSocketsilla asiakasohjelmaan.

Koska .NET-ympäristöön ei löydetty riittävän kattavaa reitinoptimointikirjastoa, päädyttiin luomaan mikropalvelu Javalla, jotta päästäisiin hyödyntämään Jsprit-kirjastoa. Palveluun luotiin yksinkertainen REST-rajapinta, joka ottaa parametrina tiedot tukkikentän tilanteesta (varastopaikkojen ja ajoneuvojen sijainnit, kapasiteetit, täyttöasteet, oletetut

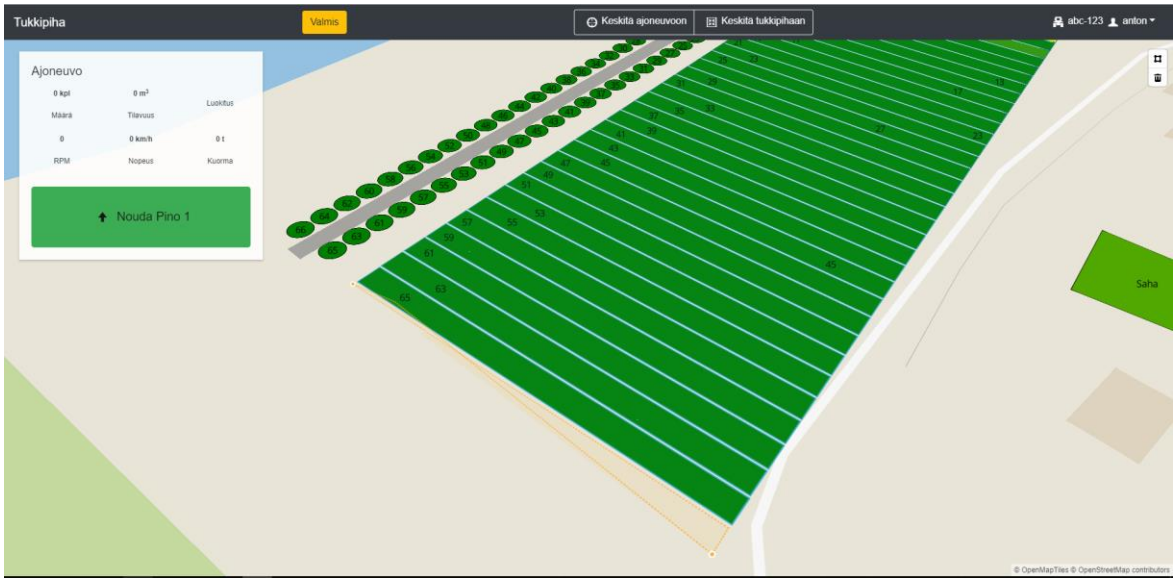
täyttymisnopeudet) ja palauttaa optimoinnin jälkeen listan työtehtävistä per ajoneuvo. Palvelua kutsutaan tukkikenttäsovelluksen pääpalvelimelta.

4.6 Sovelluksen toiminnallisuuksien kuvaus

Seuraavaksi esitellään prototyypin keskeisimmät toiminnot kuvaten yleisimmät käyttäjän toiminnot käyttötapausten muodossa.

4.6.1 Alkuasetukset

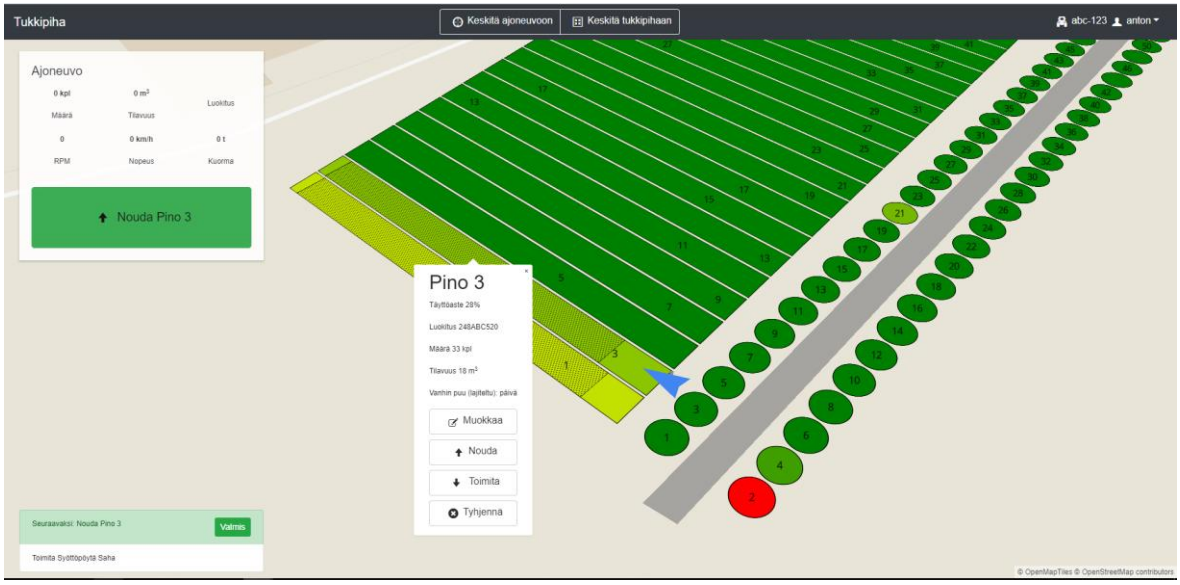
Käyttäjä kirjautuu sovellukseen samoilla tunnuksilla, joita käytetään tukkilajittelusovelluksessa. Kirjautumisen jälkeen käyttäjä ohjataan karttanäkymään. Aluksi kartta on täysin tyhjä. Varastopaikat voidaan määritellä kartalle navigoimalla ylävalikosta asetuksiin, ja valitsemalla *Muokkaa karttaa*. Tämä asettaa kartan editointimoodiin, jolloin lokeroita, tukkipinoja, syöttöpöytiä ja esteitä voidaan lisätä, muokata ja poistaa kartalta. Karttaobjekteja voidaan muokata myös yhtäaikaisesti laatikkovalinnalla. Kartan määrittelyn tekee kuitenkin aluksi järjestelmän toimittaja, eikä käyttäjän oleteta muokkaavan karttaa jatkuvasti. Esimerkiksi varatukkipinopohjat kannattaa määritellä jo heti käyttöönotossa, sillä niiden sisältämät tai hyväksymät tuotteet määritellään dynaamisesti ajon aikana, jolloin lisäpaikkoja ei tarvitse määritellä jatkuvasti tuotannon aikana. Ohjelmassa on myös aputoimintoja, joilla samankokoisia tukkipinoja voidaan määrittää piirtämällä vain yksi isompi nelikulmio halutulle alueelle (kuten kuvassa 4). Lokerot kuljettimen vieressä voidaan määrittää piirtämällä viiva ensimmäisen ja viimeisen lokeron välillä ja määrittämällä lokeroväli. Asetuksista määritellään myös olemassa olevat ajoneuvot ja niiden parametrit, kuten maksimikuorma ja GPS-paikannuksen korjausparametrit. Ajoneuvovalinta on sidoksissa päätelaitteeseen, ei käyttäjään, joten asetuksista voidaan valita tämänhetkinen ajoneuvo (kuvassa 4 yläpalkissa oikealla) tai jättää valinta tyhjäksi, jolloin karttaa voi katsella ilman konekuskin toimintoja. Asetuksista voidaan valita myös tumma värimaailma helpottamaan työskentelyä iltaisin, vaihtaa pinojen nimeämiskäytäntöä, ja kytkeä päälle äänimerkit nosto- ja laskutapahtumista.



Kuva 4: esimerkki uuden pinon määrittämisestä karttamuokkaustilassa käsin piirtämällä

4.6.2 Tietojen tarkastelu kartalla

Varastopaikkojen ja muiden alkuasetusten määrittämisen jälkeen lokerot täyttyvät tukkilajitteluohjelman tarjoaman tiedon perusteella reaaliaikaisesti kaikkiin asiakaskäyttöliittymiin WebSocketsin avulla. Varastopaikan väri kuvastaa varastopaikan täyttöastetta. Lisäksi tukkipinoissa arvioidaan varastopaikan pituutta kentällä. Pituus visualisoidaan varjostuksella. Jokaisen varastopaikan sisältöä voi tarkastella klikkaamalla kyseistä objektia kartalla. Aukeavassa ikkunassa on tietoa siitä, mikä on kyseisen varastopaikan puuluokka, paljonko puita on varastopaikassa (kappalemäärä ja tilavuus) sekä mikä varaston ikä on perustuen vanhimpaan puuhun. Ikkunasta voi myös suorittaa manuaalisia nosto- lasku- ja tyhjennysoperaatioita, ilman että käyttäjän tarvitsee olla fyysisesti varastopaikan lähellä. Varastopaikan tietoja – kuten maksimikapasiteettia – voi myös muokata aukeavassa ikkunassa.



Kuva 5: Tietojen tarkastelu karttanäkymässä

4.6.3 Varastonsiirto

Normaalitilanteessa varastonsiirto on suunniteltu tapahtuvaksi niin, että käyttäjän tarvitsee painaa yhtä painiketta (*nouda tai laske varastopaikkaan*, kuvassa 5 vasemmalla keskellä), kun varastoa noudetaan tai toimitetaan. Jatkossa tämä pikapainike pyritään automatisoimaan ajoneuvon vaakatiedon perusteella. Tällöin konekuskin ei parhaassa tapauksessa normaalitilanteessa tarvitsisi operoida käyttöliittymää laisinkaan. Lisäksi pikapainikkeen toimintoa voidaan myös täydentää fyysisellä painonapilla.

1. Konekuski siirtyy noudettavan varastopaikan läheisyyteen. Sininen nuoli (eli tämänhetkinen ajoneuvo) liikkuu ja kiertyy satelliittipaikannuksen ja tabletin orientaatiotiedon perusteella kartalla. Nouto- ja laskupainikkeen teksti kertoo lähimmän varastopaikan tiedon.
2. Konekuski nostaa kuorman ja painaa *Nouda varastopaikka*, jolloin varastopaikasta vähennetään koneen maksimikapasiteetin verran tuotteita, ja ne siirretään koneen varastoon. Tukit järjestetään myös niin, että kouraa lähinnä olevat tukit otetaan kuljetukseen siinä tapauksessa, että tukkien määrä varastopaikassa ylittää ajoneuvon maksimikapasiteetin. Tämä auttaa esimerkiksi tukkipinon pituuden määrityksessä.

3. Kun koneella on kuormaa, ohjelma korostaa niitä varastopaikkoja, johon kuorma voidaan laskea (kuva 6). Lisäksi ajoneuvokortti vasemmalla näyttää avaintietoja kuormasta. Myös *Laske varastopaikkaan* -painikkeen väri (vihreä tai harmaa) kertoo, onko kone tällä hetkellä oikean laskupaikan läheisyydessä.
4. Kuski siirtää kuorman varastopaikalle, laskee kuorman ja painaa kuorman laskupainiketta. Kuorma siirretään ajoneuvon varastosta uuteen varastopaikkaan. Kuorman laskupaikka otetaan myös talteen, sillä tukkipinon tapauksessa tukkipinon pituus arvioidaan laskupaikoista.

Kuva 6: Varastonsiirto. Kuorma on nostettu lokerosta 4 ja se toimitetaan pinoon 7

Varastonsiirto pinosta sahan syöttöpöydälle toimii samankaltaisesti. Syöttöpöydät ovat varastopaikkoja, joista vähennetään automaattisesti varastoa sahan järjestelmän lähettämien viestien perusteella. Kun saha on sahannut puun, voidaan tuotetta vähentää sahapöydän varastosta. Tämä viesti on kuitenkin aina sahakohtainen, eikä sitä ole aina saatavilla. Tämä viesti voi tulla myös jatkossa Finnos Oy:n sahan eteen sijoitetusta toisesta mittarista.

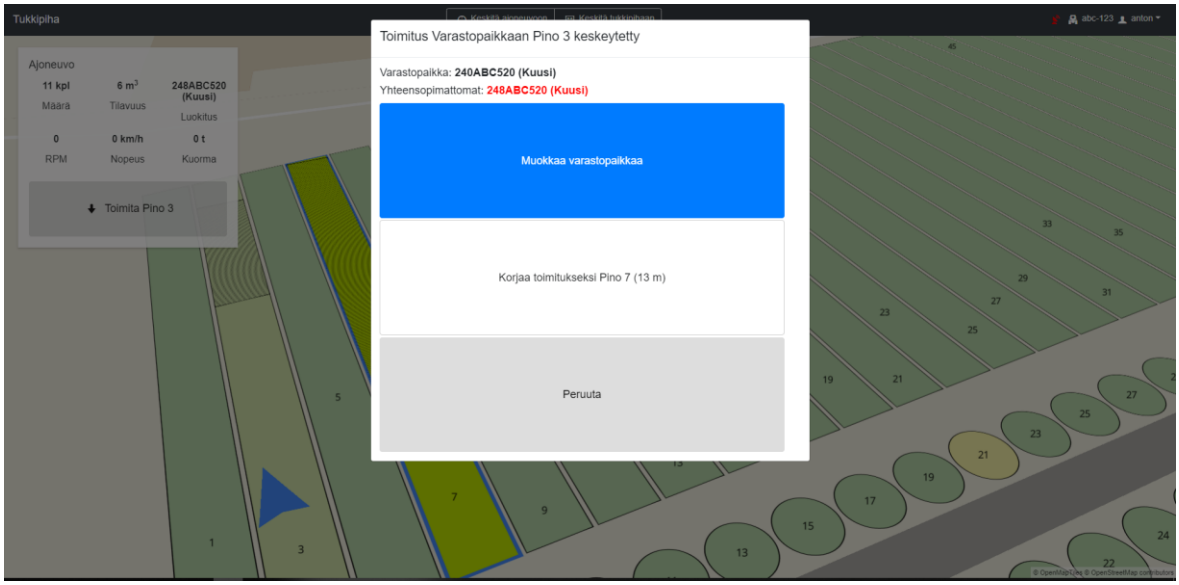
Myös lajittelun syöttämistä voidaan mallintaa hieman samankaltaisesti. Haaste syntyy kuitenkin siitä, että toimitettujen kuormien sijaintia ei voida tarkalleen tietää. Yleensä kuormille on kuitenkin varattu lajittelupöydän läheisyydestä alue. Puiden vastaanottojärjestelmästä on usein myös saatavilla viestinä tieto kuormista, jotka ovat

lähitulevaisuudessa saapumassa tai juuri saapuneet. Tukkien vastaanoton kannalta tukkikenttäsovellukseen jää tämän työn jälkeen vielä jatkokehitettävää.

4.6.4 Varoitukset

Yksi ohjelman höydyistä on, että virheellisiä varastonsiirtoja voidaan huomata, ehkäistä ja korjata. Kun kuormaa ei ole, käyttäjä voi mielivaltaisesti aloittaa uuden varastonsiirron. Nosto- ja laskupikapainike on siis aina vihreä, kun kuormaa ei ole. Kun varastonsiirto on aloitettu ja ajoneuvolla on kuormaa, voi käyttäjä pikapainikkeen väristä katsomalla todeta ennen kuorman laskemista, onko hän toimittamassa oikeaan varastopaikkaan. Mikäli käyttäjä kuitenkin laskee kuorman ohjelman näkökulmasta väärään varastopaikkaan, antaa ohjelma varoituksen ponnahdusikkunana (kuva 7). Ponnahdusikkunasta voidaan:

1. Pakottaa siirto, eli muokata varastopaikkaa niin, että se on jatkossa kahden tai useamman tukkiluokan sekoitus. Jos pino on täysin tyhjä, niin painike luo uuden varastopaikan kyseiselle tukkiluokalle.
2. Valita oikea, lähimpänä oleva varastopaikka kuorman vastaanottajaksi. Painikkeesta voi olla hyötyä esimerkiksi siinä tilanteessa, että sijaintitieto on ollut hetkellisesti epätarkka, ja nykyinen sijaintitieto sattuu osumaan esimerkiksi viereiseen varastopaikkaan.
3. Peruuttaa varastonsiirto, jolloin voidaan ajaa oikeaan varastopaikkaan ja suorittaa toiminto uudelleen.



Kuva 7: Varoitusikkuna, kun käyttäjä on laskenut kuorman väärään pinoon

5 TULOKSET

Testejä suoritettiin kolmena eri päivänä Versowood Vierumäen sahalla.

Testiä varten RTK-korjauksia tarjoava kiintoasema (Emlid Reach RS+) kalibroitiin kiintopisteeseen 15 minuutin ajan keskiarvottamalla tänä aikana saadut staattisen pisteen koordinaatit. Toinen vastaavanlainen satelliittipaikannin asennettiin tukkikurottajaan. Vastaanottimet konfiguroitiin käyttämään RTK-korjauksia LoRa-radion välittämänä.

Tukkikenttä mallinnettiin nauhoittamalla pisteitä vastaanottimen tarjoamaa tutkimustyökalua käyttäen. Pisteet ensimmäisestä ja viimeisestä tukkilokerosta nauhoitettiin noin kahden metrin etäisyydeltä lokerosta, ja ensimmäisen ja viimeisen lokeron välille interpoloitiin loput lokerot lokeroiden välimatkaa hyödyntäen. Tämä toistettiin kuljettimen kummallekin puolelle. Tukkipinot mallinnettiin niin, että nelikulmion mallinen alue nauhoitettiin tukkipinoille varatun alueen ääripisteistä, ja sen jälkeen alueen sisälle jäävä osa jaettiin ohjelmallisesti tasaisesti tukkipinon levyisiin kaistaleisiin. tutkimustyökalusta nauhoitetut pisteet saatiin vietyä GeoJSON-muotoisena tiedostoon. Sovellukseen rakennettiin lisäksi aputoiminnallisuus, joka pystyi tuomaan GeoJSON-vientitiedoston ja luomaan lokeroita ja pinoja tutkimustyökalun GeoJSON-pisteistä. Lokeroiden ja tukkipinojen maksimikapasiteetit määriteltiin konekuskin kanssa arvioimalla tukkipinon maksimikorkeus sekä lokeron maksimikapasiteetti (lokeron poikkipinta-ala, ja tukkipinon sivuprofiilipinta-ala pituus-korkeussuunnassa).

Testeistä saatiin runsaasti käyttöliittymään ja toimintavarmuuteen liittyvää palautetta. Koska testisahalla lajitellaan vuorotellen sekä kuusta että mäntyä, pidettiin tärkeänä näkymää, jossa eri lajia olevat pinot visualisoidaan eri värein, täyttöasteen sijaan. Sovellukseen kehitettiinkin palautteen perusteella värikoodauksen tilavaihto, jossa voidaan visualisoida varastopaikat täyttöasteen (vihreän sävy: tyhjä, punaisen sävy: täynnä), lajin (vihreä: mänty, oranssi: kuusi) tai ikäjakauman mukaan (vihreän sävy: uusi, punaisen sävy: vanha). Lisäksi käyttöliittymään luotiin mahdollisuus sulauttaa IP-kamerakuvaa sahan tai lajittelun syöttöpöydältä. Sovelluksen operoimisen helpottamiseksi ideoitiin erillinen painonappi, jolla pikatoimintoja, kuten nosto- ja laskutoimintoja ja hälytysten kuittauksia

pystyisi suorittamaan ilman tietokoneeseen koskemista. Langaton painonappi kiinnitettiin kurottajan ohjauslaitteeseen, jotta käyttäjän ei tarvitse irrottaa otettaan siitä. Painonappia varten kehitettiin palvelu, joka kuuntelee napinpainalluksia Bluetoothin välityksellä, ja välittää tiedot WebSocketsilla selaimelle. Pinojen nimitietoja varten kehitettiin valintamahdollisuus näyttää pinon varsinaisen nimen sijaan esimerkiksi pinoa vastaavan lokeron numero, tai vastaava tukkiluokka, sillä sahojen käyttämä toimintamalli pinojen nimeämisen suhteen on vaihtelevaa. Lisäksi huomattiin, että käyttäessä kameran *seuraa*-tilaa, olisi käyttöä helpottavaa, jos kamera zoomaisi hetkeksi ulospäin varaston nostotilanteessa, jotta vastaanottavat varastopaikat saataisiin nopeammin selville. Tätä varten kehitettiin asetuksiin valinta *Automaattinen kameran ohjaus*, joka mahdollistaa kyseisen käyttäjän toiveen.

Kiintoaseman koordinaatteja nauhoitettiin vain 15 minuutin ajan. Tuotantokäytössä kiintoasema tulisi kalibroida pitemmällä aikavälillä, ja absoluuttista tarkkuutta voidaan parantaa lähettämällä vastaanottimen nauhoittama raakadata jälkiprosessointiin PPP-palveluun (Precise Point Positioning). Tämän takia liikkuvan vastaanottimen absoluuttinen tarkkuus heikkenee, mutta suhteellinen sijainti kiintoasemaan nähden on silti edelleen hyvin tarkka. Testeissä huomattiin GPS-antennin sijoituspaikan tärkeys. Antennin sijoituspaikkaa vaihdettiin hytin, katon ja kuormaajan puomin välillä. Sijainnin äkillisiä vaihteluita syntyi vähemmän, kun antennilla on selvä taivasnäkymä, jolloin heijastuneita signaaleita syntyy vähemmän.

Testiä varten hankittiin myös erillinen inertiamittausyksikkö (Bosch BNO055), joka suorittaa mikroprosessorilla sensorifuusiota magnetometrin, kiihtyvyysanturin ja gyroskoopin välillä tarjoten absoluuttista suuntatietoa. Anturi toimi ideaaliolosuhteissa toimistossa hyvin, mutta testitilanteessa kurottajan ferromagneettiset metallit sekä elektroniikka aiheuttivat suuntatiedon ajelehtimista. Pienikin suuntatiedon virheellisyys voi tässä tilanteessa aiheuttaa sen, että oletettu ajoneuvon kouran piste – eli piste sijaintitiedon sijainnista ajoneuvon suuntavektorin suuntaan, kouran ja antennin etäisyyden verran – siirtyy sen verran, että noudettaisiin väärä lokero. Anturin kalibrointiin oikeassa ympäristössä on tutustuttava tarkemmin. Lisäksi paikkatiedon ja orientaatiotiedon fuusiota olisi tutkittava, sillä nyt kyseiset anturit toimivat täysin itsenäisesti.

Lopputuloksena voidaan testikäyttäjän kommenttien perusteella pitää, että sovelluksesta olisi hyötyä konekuskin työskentelyssä. Kuitenkin sensorien toimintavarmuutta olisi vielä paranneltava, jotta etenkin varaston nosto-operaatiot olisivat luotettavampia. Erityisesti testeissä jouduttiin korjaamaan inertiamittausyksikön suuntatiedon ajelehtimista korjausvakiolla, mikä teki operoinnista paikoin vaivalloista. Suuntatiedosta voidaan kuitenkin luopua, jos GNSS-antenni sijoitetaan koneen kouraan. Sijaintitiedon tarkkuutta pidettiin riittävänä suurimman osan ajasta. Vähäistä signaalin heikkoutta ilmeni, jos GNSS-antenni oli sijoitettu heikosti (hyttiin tai katolle kouran puomin alle). Sovelluksen käytettävyyttä saatiin paranneltua testipäivien välillä painonapin, kameran automaattisen zoomaustoiminnon ja uusien pikakomentojen avulla. Testejä suoritettiin yhden konekuskin kanssa yhdessä vuorossa. Testejä suoritetaan jatkossa useammassa ajoneuvossa yhtäaikaisesti, jotta saadaan parempi käsitys siitä, millaiset hyödyt ovat realistisemmassa työtilanteessa. Koska testitilanteessa ei ollut muita käyttäjiä, kentän tilanne jää suurilta osin päivittymättä, kun esimerkiksi puolia lokeroista ei tyhjennetä.

6 POHDINTA JA TULEVAISUUS

Työn käytännön osuudessa syntynyttä sovellusta tullaan jatkokehittämään loppukäyttäjien antaman palautteen perusteella. Sovellukseen kaavailtu automaattinen kuormantunnistus osoittautui haastavaksi toteuttaa kustannustehokkaasti, sillä pyöräkuormaajien valmistajien dokumentaatiota ei ollut vapaasti saatavilla, ja selvisi, että useimmissa pyöräkoneissa ei ole tarvittavia antureita vakiovarusteena. Kuormantunnistuksen vaihtoehtoisiin tapoihin tarvitaan jatkoselvitystyötä. Koneiden reitinoptimointiin jää myös lisäkehitettävää, sillä ensimmäisessä versiossa pyrittiin vain kuljetun matkan minimointiin. Reitinoptimoinnilla saatavia hyötyjä olisi myös tutkittava analyyttisesti. Olisi siis vertailtava ihmisen intuitioon perustuvaa työskentelyä siihen, miten reitinoptimointialgoritmi olisi suunnitellut työjärjestyksen. Tällaisen vertailun pohjalta voitaisiin todeta, voiko reitinoptimointi tuoda merkittäviä kustannussäästöjä. Tulosten perusteella voitaisiin päättää jatkopanostuksesta sovelluksen reitinoptimointiominaisuuksiin ja käyttää tuloksia myynnin tukena.

Useita lisäominaisuuksia rajattiin käytännön työn ensimmäisen version ulkopuolelle. Esimerkiksi vaihtoehtoisia käyttöliittymäinteraktioita kuten puhe- tai eleohjausta ei tässä versiossa ole toteutettu. Lisäksi pyöräkoneiden diagnostiikkaan, kuskien työskentelytehokkuuden seuraamiseen ja työskentelyn pelillistämiseen liittyviä ideoita ja ominaisuuksia on kertynyt työn aikana jatkokehityslistalle. Näiden tuomaa lisäarvoa olisi tutkittava esimerkiksi lisähaastatteluilla.

Sensoriteknologioiden kehitys on edelleen nopeaa, ja uusia tekniikoita varsinkin paikannuksen tarkkuuden parantamiseksi kehitetään jatkuvasti. Työn jatkuvuuden kannalta olisi seurattava varsinkin kustannustehokkaiden, tablettiin integroitujen satelliittipaikantimien kehitystä. Esimerkiksi L1- ja L5-signaaleja hyödyntävän Broadcom BCM47755-sirun odotetaan mahdollistavan alle metrin paikannustarkkuuden tietyissä uusissa kuluttajaälypuhelimissa ja -tableteissa vuoden 2018 aikana [16].

Sovellus suunniteltiin ensisijaisesti tukkikentän hallintaan. Sovelluksen tietomalli pyrittiin pitämään kuitenkin mahdollisimman yleistettävänä niin, että sovellusta voitaisiin jatkossa hyödyntää muihin vastaaviin varastointitoimenpiteisiin. Sahalaitokset voivat varastoida

lopputuotteina syntyvät lautapaketit hieman tukkikentän kaltaisesti, joten ohjelman soveltuvuutta tähän tarkoitukseen voitaisiin selvittää. Sovelluksen käyttömahdollisuudet eivät rajoitu vain sahateollisuuteen: esimerkiksi satamakonttien varastoinnissa nähdään samoja piirteitä. Lisäksi luodusta varastonhallintasovelluksesta voisi olla hyötyä esimerkiksi puupylvästuotannossa, jossa tolppien varastointi toteutetaan hyvin samankaltaisesti kuin tukkikentällä.

7 YHTEENVETO

Tässä työssä on kuvattu uuden ohjelmistotuotteen ensimmäisen version kehityksen kulku. Teoriaan ja haastatteluihin perustuen kehitettiin vaatimukset tukkikentän hallintajärjestelmän toteuttamiseksi. Vaatimuksia tukevia teknisiä ratkaisuja tehtiin suunnittelun pohjalta. Lopuksi toteutusta testattiin yhdessä loppukäyttäjien kanssa useaan kertaan iteroiden. Kaikkiin alussa mainittuihin tutkimuskysymyksiin on esitetty vastauksia ja ratkaisuja tässä työssä. Lopputuloksena on pohja uudelle sovellukselle, jonka loppukäyttäjä näkisi hyödylliseksi työssään. Lisäksi työn aikana on kertynyt runsaasti kehitysehdotuksia, joita ei varsinaisesti voida laskea MVP:n piiriin. Työ jatkuu kehitysehdotuksien toteuttamisella, painottaen varsinkin sensoridatan tarkkuudesta ilmenneitä kehityskohteita.

LÄHTEET

[1] R. Schmidt, M. Möhring, R.-C. Härting, C. Reichstein, P. Neumaier, ja P. Jozinović, ”Industry 4.0 - Potentials for Creating Smart Products: Empirical Research Results”, teoksessa *Business Information Systems*, vsk. 208, W. Abramowicz, Toim. Cham: Springer International Publishing, 2015, ss. 16–27.

[2] J. Lukkari *ym.*, ”Puun laadun säilyttäminen -opas”, s. 27.

[3] O. Laiho, ”Sahalaitoksen tuotantoprosessi”, s. 21.

[4] ”Sahatavaran valmistus”. [Verkossa]. Saatavissa: http://www03.edu.fi/oppimateriaalit/puutuoteteollisuus/ensijalostus/sahatavaratuotanto/sahatavaran_valmistus.html. [Viitattu: 22-joulu-2018].

[5] M. Varpenius, ”Uuden tuotteen soveltuvuus markkinoille, case: tukkikentän hallintajärjestelmä”, s. 48.

[6] M. Li, S. Gu, G. Chen, ja Z. Zhu, ”A RFID-based Intelligent Warehouse Management System Design and Implementation”, teoksessa *2011 IEEE 8th International Conference on e-Business Engineering*, 2011, ss. 178–184.

[7] ”Application of RTLS in Warehouse Management Based on RFID and Wi-Fi - IEEE Conference Publication”. [Verkossa]. Saatavissa: https://ieeexplore.ieee.org/abstract/document/4679157. [Viitattu: 26-syys-2018].

[8] ”(PDF) Design of An Intelligent Warehouse Management System”, *ResearchGate*. [Verkossa]. Saatavissa: https://www.researchgate.net/publication/259558571_Design_of_An_Intelligent_Warehouse_Management_System. [Viitattu: 21-loka-2018].

[9] ”What is GNSS?”, 01-maalis-2016. [Verkossa]. Saatavissa: https://www.gsa.europa.eu/european-gnss/what-gnss. [Viitattu: 20-loka-2018].

[10] ”What is SBAS?”, 01-maalis-2016. [Verkossa]. Saatavissa: https://www.gsa.europa.eu/european-gnss/what-gnss/what-sbas. [Viitattu: 20-loka-2018].

[11] ”Real-Time Kinematic and Differential GPS | GEOG 862: GPS and GNSS for Geospatial Professionals”. [Verkossa]. Saatavissa: https://www.e-education.psu.edu/geog862/node/1828. [Viitattu: 11-marras-2018].

[12] ”What is NTRIP?” [Verkossa]. Saatavissa: https://www.anatumfieldsolutions.com/What-is-NTRIP_b_42.html. [Viitattu: 11-marras-2018].

[13] G. Weber, D. Dettmering, ja H. Gebhard, ”Networked Transport of RTCM via Internet Protocol (NTRIP)”, teoksessa *A Window on the Future of Geodesy*, vsk. 128, F. Sansò, Toim. Berlin/Heidelberg: Springer-Verlag, 2005, ss. 60–64.

[14] J. P. Cintra, M. A. Nero, ja D. Rodrigues, ”GNSS/NTRIP service and technique: accuracy tests”, *Bol. Ciênc. Geodésicas*, vsk. 17, nro 2, ss. 257–271, kesä 2011.

[15] ”DGNSS-paikannuspalvelu | Maanmittauslaitos”. [Verkossa]. Saatavissa: https://www.maanmittauslaitos.fi/kartat-ja-paikkatieto/paikannuspalvelu/dgnss-palvelu. [Viitattu: 17-touko-2018].

[16] S. K. Moore, ”Super-accurate GPS coming to smartphones in 2018 [News]”, *IEEE Spectr.*, vsk. 54, nro 11, ss. 10–11, marras 2017.

[17] L. Wang, Z. Li, J. Zhao, K. Zhou, Z. Wang, ja H. Yuan, ”Smart Device-Supported BDS/GNSS Real-Time Kinematic Positioning for Sub-Meter-Level Accuracy in Urban Location-Based Services”, *Sensors*, vsk. 16, nro 12, joulu 2016.
[18] ”GPS.gov: GPS Accuracy”. [Verkossa]. Saatavissa: https://www.gps.gov/systems/gps/performance/accuracy/. [Viitattu: 20-loka-2018].
[19] M. Pedley, ”Tilt Sensing Using a Three-Axis Accelerometer”, s. 22.
[20] V. M. N. Passaro, A. Cuccovillo, L. Vaiani, M. De Carlo, ja C. E. Campanella, ”Gyroscope Technology and Applications: A Review in the Industrial Perspective”, *Sensors*, vsk. 17, nro 10, s. 2284, loka 2017.
[21] ”DeviceOrientation Event Specification”. [Verkossa]. Saatavissa: https://w3c.github.io/deviceorientation/. [Viitattu: 15-joulu-2018].
[22] ”Detecting device orientation”, *MDN Web Docs*. [Verkossa]. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/API/Detecting_device_orientation. [Viitattu: 15-joulu-2018].
[23] ”arduino uno - Should I use an accelerometer or gyroscope in my project?”, *Arduino Stack Exchange*. [Verkossa]. Saatavissa: https://arduino.stackexchange.com/questions/47483/should-i-use-an-accelerometer-or-gyroscope-in-my-project. [Viitattu: 15-joulu-2018].
[24] A. G. Mirnig *ym.*, ”Automotive User Experience Design Patterns: An Approach and Pattern Examples”, s. 13, 2016.
[25] J. Nielsen, ”10 Heuristics for User Interface Design: Article by Jakob Nielsen”. [Verkossa]. Saatavissa: http://www.nngroup.com/articles/ten-usability-heuristics/. [Viitattu: 13-joulu-2015].
[26] A. Riener ja M. Rossbory, ”Natural and Intuitive Hand Gestures: A Substitute for Traditional Vehicle Control?”, s. 2.
[27] D. L. Strayer, J. M. Cooper, J. Turrill, J. R. Coleman, ja R. J. Hopman, ”Talking to your car can drive you to distraction”, *Cogn. Res. Princ. Implic.*, vsk. 1, nro 1, joulu 2016.
[28] ”Traveling Salesman Problem | Optimization”, *Google Developers*. [Verkossa]. Saatavissa: https://developers.google.com/optimization/routing/tsp/tsp. [Viitattu: 16-huhti-2018].
[29] ”Traveling Salesman Problem”. [Verkossa]. Saatavissa: http://www.math.uwaterloo.ca/tsp/index.html. [Viitattu: 15-joulu-2018].
[30] Tonci Caric ja Hrvoje Gold, *Vehicle Routing Problem*. INTECH Open Access Publisher, 2008.
[31] R. Sedgewick ja K. Wayne, *Algorithms*, 4th p. Addison-Wesley Professional, 2011.
[32] W. Zeng ja R. L. Church, ”Finding shortest paths on real road networks: the case for A*”, *Int. J. Geogr. Inf. Sci.*, vsk. 23, nro 4, ss. 531–543, huhti 2009.
[33] ”What Web Can Do Today”, *WhatWebCanDo*. [Verkossa]. Saatavissa: https://whatwebcando.today/. [Viitattu: 16-joulu-2018].
[34] ”Angular Docs”. [Verkossa]. Saatavissa: https://angular.io/docs. [Viitattu: 16-joulu-2018].
[35] ”React Docs”. [Verkossa]. Saatavissa: https://reactjs.org/docs/getting-started.html. [Viitattu: 16-joulu-2018].
[36] ”VueJS Docs”. [Verkossa]. Saatavissa: https://vuejs.org/v2/guide/. [Viitattu: 16-joulu-2018].
[37] ”Leaflet - a JavaScript library for interactive maps”. [Verkossa]. Saatavissa: https://leafletjs.com/. [Viitattu: 13-loka-2018].

[38] ”Wykks/ngx-mapbox-gl: Angular binding of mapbox-gl-js”. [Verkossa]. Saatavissa: https://github.com/Wykks/ngx-mapbox-gl. [Viitattu: 10-marras-2018].
[39] ”GeoJSON”. [Verkossa]. Saatavissa: http://geojson.org/. [Viitattu: 21-loka-2018].
[40] ”Turf.js | Advanced geospatial analysis”. [Verkossa]. Saatavissa: http://turfjs.org/. [Viitattu: 21-loka-2018].
[41] ”About Combinatorial Optimization | OR-Tools | Google Developers”. [Verkossa]. Saatavissa: https://developers.google.com/optimization/introduction/overview. [Viitattu: 23-joulu-2018].
[42] *jsprit is a java based, open source toolkit for solving rich vehicle routing problems: graphhopper/jsprit*. GraphHopper, 2019.
[43] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, ja G. Dueck, ”Record Breaking Optimization Results Using the Ruin and Recreate Principle”, *J. Comput. Phys.*, vsk. 159, nro 2, ss. 139–171, huhti 2000.
[44] ”Trimble R1 GNSS”, *Default Store View*. [Verkossa]. Saatavissa: https://shop.geotrim.fi/trimble-r1-gnss.html. [Viitattu: 24-helmi-2019].
[45] ”Trimble Store : GNSS Receivers”. [Verkossa]. Saatavissa: https://store.trimble.com/OA_HTML/ibeCCtpSctDspRte.jsp?section=10063&sitex=10140:22372:US. [Viitattu: 24-helmi-2019].
[46] ”RTKLIB: An Open Source Program Package for GNSS Positioning”. [Verkossa]. Saatavissa: http://www.rtklib.com/. [Viitattu: 24-helmi-2019].
[47] ”How RTK works - Reach M+ docs”. [Verkossa]. Saatavissa: https://docs.emlid.com/reachm-plus/common/tutorials/rtk-introduction/. [Viitattu: 24-helmi-2019].
[48] ”What is microservices? - Definition from WhatIs.com”. [Verkossa]. Saatavissa: https://searchmicroservices.techtarget.com/definition/microservices. [Viitattu: 10-marras-2018].
[49] L. Chen, ”Microservices: Architecting for Continuous Delivery and DevOps”, teoksessa *2018 IEEE International Conference on Software Architecture (ICSA)*, Seattle, WA, 2018, ss. 39–397.
[50] ”NMEA data”. [Verkossa]. Saatavissa: http://www.gpsinformation.org/dale/nmea.htm. [Viitattu: 21-joulu-2018].