OULUN AMMATTIKORKEAKOULU

AUTONOMISEN AJOJÄRJESTELMÄN KEHITTÄMINEN

Asiakas-palvelinarkkitehtuuria hyödyntäen

Aleksi Koberg, Jukka Loppukaarre & Saku Suorajärvi
Opinnäytetyö AMK
Kevät 2025
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma
Ohjelmistokehitys

Tekijä(t): Aleksi Koberg, Jukka Loppukaarre ja Saku Suorajärvi
Opinnäytetyön otsikko: Autonomisen ajojärjestelmän kehittäminen
Työn ohjaaja(t): Eero Nousiainen
Työn valmistumislukukausi ja -vuosi: kevät 2025
Sivumäärä: 72

Autonomiset ajoneuvot ovat jo osa jokapäiväistä elämää ja niiden rooli yhteiskunnassa kasvaa jatkuvasti. Tällä hetkellä niitä hyödynnetään esimerkiksi autonomisissa takseissa ja varastojen logistiikassa sekä kuluttajakäytössä, kuten robottiimureina ja robottiruohonleikkureina.

Opinnäytetyön tavoitteena oli tutustua autonomisten ajoneuvojen nykytilaan, ja toteuttaa autonomisesti liikkuva robotti. Tavoitteena oli myös opetella työskentelemään kehittäjätiimin kanssa hyödyntämällä Scrum-menetelmää.

Tietoperustassa perehdyttiin opinnäytetyössä käytettyihin menetelmiin, kuten autonomisiin ajoneuvoihin, ohjelmoinnin työkaluihin, asiakaspalvelimeen, GoPiGo-robottiin ja siinä käytettäviin antureihin sekä toimintoihin. Scrum-menetelmä oli myös osa tietoperustaa. Toteutuksessa käytettiin GoPiGo-robottia ja projektinhallintamenetelmänä käytettiin Scrum-menetelmää. Robotin ohjelma toimi Python-koodien avulla käyttäen asiakas-palvelinarkkitehtuuria.

Tulokseksi tuli kaksi autonomista robottia, jotka pystyvät liikkumaan kahdestaan samalla radalla autonomisesti. Suurin osa tavoitteista saavutettiin ja product backlogista tekemättömiksi jääneitä kohtia ei tehty ajan tai niiden tarpeettomuuden vuoksi.

Johtopäätöksenä voidaan todeta, että autonomisen ajoneuvon kehitys onnistuu hyvin käyttämällä Scrum-menetelmää. Scrum-menetelmä mahdollisti nopean reagoinnin muutoksiin, mitä kehitysvaiheessa tapahtui. Tämän ansiosta saatiin toimiva tuote ajoissa tehtyä.

ABSTRACT

Oulu University of Applied Sciences
Degree Program in Information Technology
Option of Software Development

Author(s): Aleksi Koberg, Jukka Loppukaarre & Saku Suorajärvi
Title of thesis: Developing an Autonomous Driving System
Supervisor(s): Eero Nousiainen
Term and year when the thesis was submitted: Spring 2025
Number of pages: 72

Autonomous vehicles are becoming an integral part of our daily lives, and their role continues to grow across various sectors, from individual users to large companies. They are used as autonomous taxis, in warehouse logistics to deliver packages, and in many other applications. The development of autonomous vehicles aims to make roads safer and optimize company productivity.

The main goal of this thesis was to explore autonomous vehicles and create a real-life example of an autonomous robot capable of moving without human intervention. Various solutions were researched, including the types of sensors and algorithms that could be used to make a robot autonomous.

The development process followed an agile Scrum methodology, which allowed us to divide the work into smaller, manageable parts. The project was completed in nine one-week sprints, forming the empirical section of this thesis.

In the final version of the project, we successfully built an autonomous robot that could navigate along predefined paths. While the robot had additional functions, its primary goal of autonomous navigation was successfully achieved.

SISÄLLYS

1 JOHDANTO

Robotiikan nopea kehitys on tuonut autonomiset laitteet osaksi arkipäivää, ja niiden merkitys kasvaa jatkuvasti. Erilaiset robotit ja ajoneuvot hoitavat jo monia tehtäviä niin teollisuudessa kuin kotitalouksissa. Teollisuudessa robotteja otetaan käyttöön entistä enemmän vuosittain, mikä kuvastaa automaation kiihtyvää kehitystä. Samaan aikaan autonomisia järjestelmiä hyödynnetään arkielämässä muun muassa itseajavien taksien, varastojen automatisoitujen kuljetuslaitteiden sekä robotti-imurien ja -ruohonleikkureiden muodossa. Tämä kehityssuunta muodostaa lähtökohdan tälle opinnäytetyölle.

Työn aiheena on autonomisen robotin ohjausjärjestelmän ja käyttöliittymän toteutus. Käytännössä opinnäytetyössä suunniteltiin ja rakennettiin pienimuotoinen autonominen robottijärjestelmä, johon kuuluu sekä robottien liikkumista ohjaava palvelin ja käyttäjälle tarkoitettu hallintapaneeli, että robotin toiminnallisuuksista vastaava asiakasohjelma.

Lähtökohtana on soveltaa nykyaikaisia robotiikan menetelmiä rakentamalla järjestelmä, joka hyödyntäisi uudella tavalla olemassa olevia teknologioita. Samalla projekti tarjoaa mahdollisuuden syventää tekijöiden osaamista ohjelmistokehityksessä, nopeasti kehittyvällä robotiikan alalla ja vastata kasvavaan autonomisten järjestelmien osaajatarpeeseen.

Työn tavoitteena on luoda itsenäisesti liikkuva robotti sekä sille keskeiset ohjelmistokomponentit. Palvelimen tulee huolehtia robotin suunnistuksesta annetussa ympäristössä ja käyttöliittymän kautta käyttäjä voi seurata robotin tilaa sekä antaa komentoja. Tavoitteena on toteuttaa järjestelmä asiakas-palvelin-arkkitehtuurilla siten, että yhtä palvelinta ja käyttöliittymää hyödyntämällä voidaan hallita useita robotteja samanaikaisesti. Lisäksi opinnäytetyössä pyritään kehittämään tekijöiden projektityöskentelytaitoja: kehitystyössä sovelletaan Scrum-projektinhallintamenetelmää, mikä auttaa harjoittelemaan tiimityöskentelyä ja projektinhallintaa käytännössä.

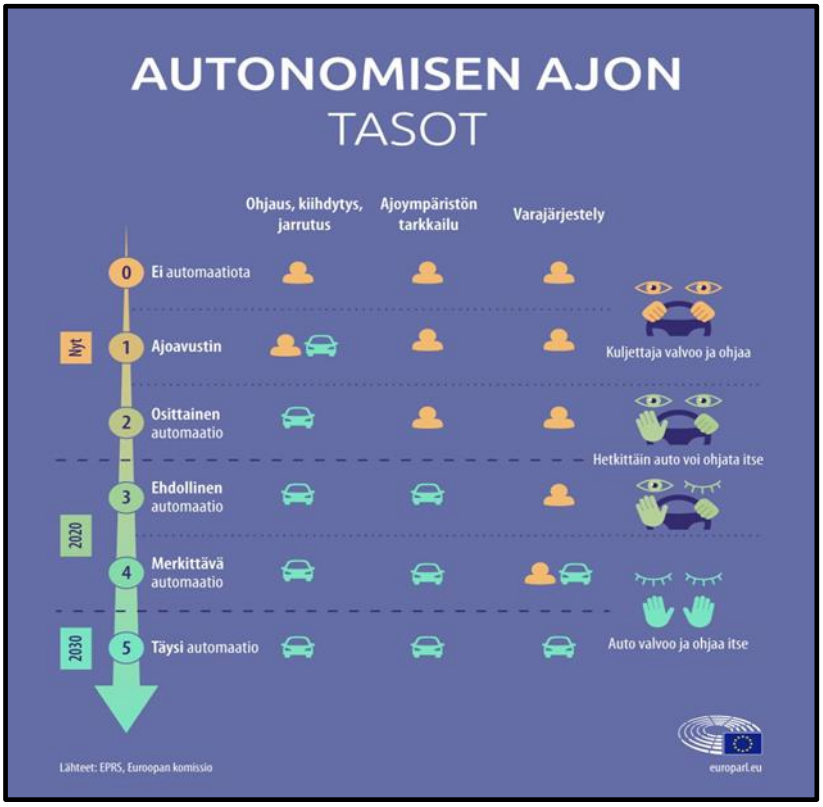
Työn merkitys korostuu sekä teknisessä että pedagogisessa mielessä. Työssä toteutettu järjestelmä demonstroi, kuinka autonomisen robotin ohjaus ja ympäristön havainnointi voidaan toteuttaa jo olemassa olevilla laitteilla sekä avoimen lähdekoodin ohjelmistoilla. Palvelimen rakenteessa otettiin huomioon laajennettavuus ja käyttöliittymän suunnittelussa painotettiin helppokäyttöisyyttä, koska intuitiivinen käyttöliittymä parantaa järjestelmän selkeyttä ja vähentää virheiden mahdollisuutta. Työn tuloksena syntynyttä robottialustaa voidaan mahdollisesti hyödyntää oppimisympäristönä tuleville robotiikan projekteille. Projekti antoi myös tekijöille arvokasta kokemusta tosielämän tuotekehityksestä tiimissä.

Yhteenvetona, opinnäytetyö tarjoaa konkreettisen esimerkin autonomisen robotin ohjausjärjestelmän ja käyttöliittymän onnistuneesta toteutuksesta, ja se luo pohjaa sekä jatkokehitykselle, että tekijöiden ammatilliselle kehittymiselle.

2 TIETOPERUSTA

2.1 Autonomiset ajoneuvot

Autonomiset ajoneuvot toimivat itseohjautuvasti tai siellä on antureita, jotka avustavat kuljettajaa ajamisessa, esimerkiksi kaistavahdit ja adaptiiviset nopeudensäätimet. Ajoneuvojen autonomisuus voidaan jakaa viiteen tasoon (kuva 1).



KUVA 1. Ajoneuvojen autonomisuuden tasot. (Euroopan parlamentti 14.1.2019.)

Autonomisten ajoneuvojen tarkoituksena on lisätä turvallisuutta liikenteeseen, sillä 95 %:a liikenne onnettomuuksista tapahtuu inhimillisistä virheistä. Autonomiset ajoneuvot eivät kuitenkaan ole vain liikenteeseen käytettäviä ajoneuvoja, vaan niitä voidaan käyttää teollisuudessakin. Esimerkiksi varastokeskukset ja satamat ovat tällaisia ympäristöjä, missä niitä hyödynnetään jo nykypäivänä erittäin paljon. Myös puolustusteollisuus on alkanut kehittämään autonomisia ajoneuvoja.

2.2 Autonomisten ajoneuvojen turvallisuus, eettisyys sekä laillisuus

Turvallisuus

Autonomisten ajoneuvojen yleistyminen tuo merkittäviä parannuksia liikenneturvallisuuteen, sillä optimistisimmat arviot sanovat, että onnettomuudet vähenevät jopa 90%-yksikön verran. Tästä huolimatta hengenvaaralliset tilanteet eivät katoa. Ne vähenevät autonomisten ajoneuvojen käyttöönoton myötä. Tosin autonomiset ajoneuvot joutuvat lähitulevaisuudessa liikenteeseen ihmiskuskien kanssa, jolloin niitä ei voida pitää harmittomina tienkäyttäjinä.

Hengenvaarallisia tilanteita ja ongelmia tulee olemaan jatkossa, eikä niitä voi täysin eliminoida, kun autonomiset ajoneuvot kehittyvät. Ongelmien olemassaolon seurauksena on kehkeytynyt eri ajatusmalleja näiden tilanteiden hoitamiseksi: valitaan kahden huonon vaihtoehdon välillä, ja erinäisten toimintatapojen sovellutuksia. Ideaalisesta eettisestä käytännöstä käydään avointa ja kiivasta keskustelua, jossa asiaa tarkastellaan metaetiikalla ja monitieteellisestä näkökulmasta. On hankala päättää yhtä turvallista toimintatapaa, koska monilla eri ratkaisuilla on perustelunsa. Sopivia eettisiä käytänteitä miettiessä pitäisi tasapainottaa julkista hyväksyntää ja käytänteen toteuttamiseen tarvittavia moraalisia vaatimuksia. Niiden täytyisi olla tarpeeksi hyväksyttäviä kerätäkseen luottamusta ja välttääkseen julkisen tuomitsevuuden. (Evans, de Moura, Chauvier, Chatila & Dogan 2020, 3287-3288.)

Kehittyvissä maissa liikenneonnettomuuksissa tapahtuneiden kuolemien määrä on vähentynyt kulkuneuvoissa käytettävän teknologian paranemisen myötä. Tähän on vaikuttanut muun muassa ajo- ja turvallisuusjärjestelmien käyttö. Teknologian kehittymisen myötä on mahdollista mitätöidä juopumisen, ylinopeuden ja mobiililaitteiden käytön aiheuttamat onnettomuudet. Autonomisia ajoneuvoja on tärkeä pyrkiä suojaamaan kyberhyökkäyksiltä ja haittaohjelmilta. (Martinez-Diaz & Soriguera 2018, 279–280.)

Eettisyys

Autonomiset ajoneuvot tuovat mukanaan eettisesti monimutkaisia vaaratilanteita, joissa onnettomuus on väistämätön. Näissä tilanteissa ohjelmistokehittäjien täytyisi valita, kenen henki tulisi priorisoida, jolloin voidaan pohtia erilaisia kriteerejä kohteen valitsemiselle. Tässä on esteenä esimerkiksi Yhdysvaltojen lakimuutos ja Saksan perustuslaki, joissa katsotaan, että kaikilla ihmisillä on oikeus elää. Äkkiä tapahtuvaan vaaratilanteeseen tarvitaan valinnantekoskripti, joka tekee järkevimmän päätöksen riittävällä datalla. (Maurer, Gerdes, Lenz & Winner 2015, 70–71, 74.)

Yksi keskeisimmistä autonomisiin ajoneuvoihin liittyvistä eettisistä kysymyksistä on vastuun ja velvollisuuden siirtyminen. Mitä enemmän automatisoituja ominaisuuksia ajoneuvossa on, sitä olennaisempaa on siirtyä tarkastelemaan, miten autonomisten ajoneuvojen toimintalogiikka on suunniteltu, kuin taas, miten ihmisen tulisi toimia näissä tilanteissa. Esimerkiksi törmäysalgoritmien suunnittelussa valmistajien tulee tasapainotella eettisyyden ja laillisten seuraamusten välillä. On vaikeaa löytää ratkaisu, jossa turvallisuus ja lailliset kysymykset kohtaisivat ja kaikki voisivat olla tyytyväisiä. Vastuu ja epävarmuus autonomisten ajoneuvojen turvallisuudesta voi lannistaa valmistajia kehittämästä ja tuottamasta kyseisiä teknologioita. (Martinho, Herber, Kroesen & Chorus. 568–570.)

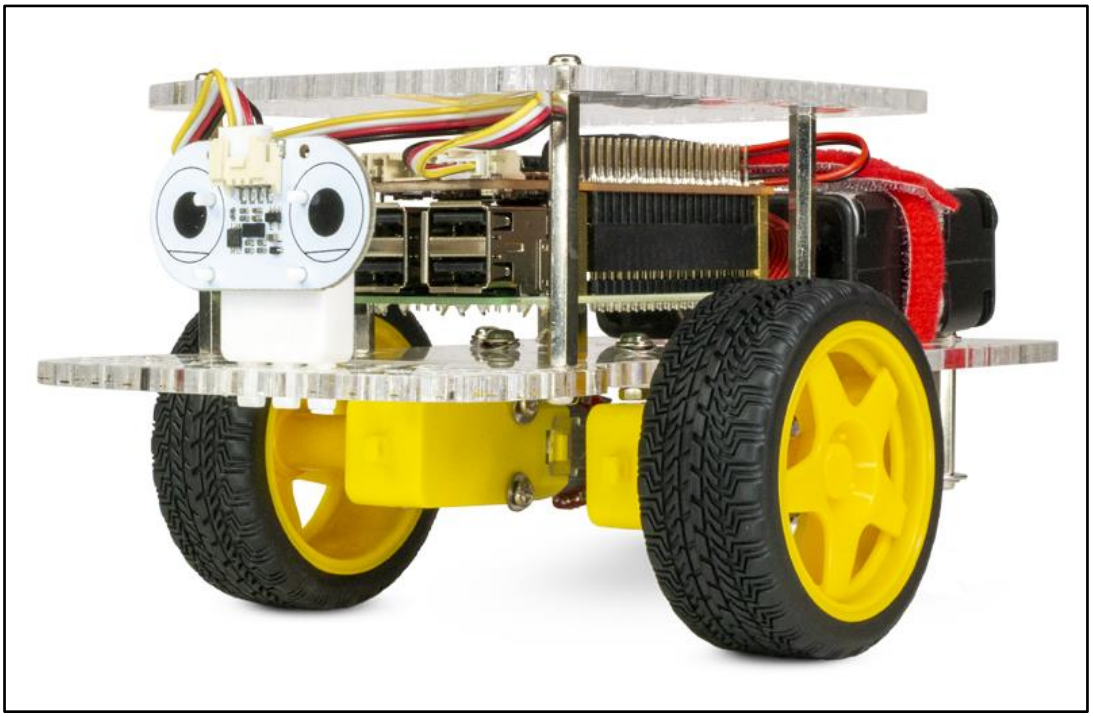
Laki

Euroopan unionin komission täytäntöönpanoasetuksen (EU) 2022/1426 mukaan täysin automatisoitujen ajoneuvojen on pysyttävä suunnitellulla toiminta-alueella ja noudatettava suunniteltua reittiä tarkasti. Tämä varmistetaan ajoneuvon automatisoidun ajojärjestelmän (ADS) avulla, joka on ohjelmoitu suorittamaan dynaamisen ajotehtävän tietty osa. ADS-järjestelmä huolehtii sekä ajoneuvon pituus- että sivusuuntaisista liikkeistä. Sen lisäksi järjestelmä havaitsee ympäristön esineet ja tapahtumat, ja osaa reagoida valmistelun avulla. (Euroopan komission täytäntöönpanoasetus 2022/1426. 3.)

Lisäksi järjestelmän on kyettävä tunnistamaan toiminta-alue ja mahdollisuus toimia alueella sekä toimintaa liittyvät olosuhteet. Onnettomuustilanteissa ajoneuvon on kyettävä tekemään ohjausliike, jolla pyritään vaaran minimoivaan tilaan.

Järjestelmä saa jatkaa normaalia toimintaa vasta, kun vahvistetaan, että toimintaa on turvallista jatkaa. Tämä ohjelmointiin perustuva hallinta korostaa järjestelmän merkitystä toimintaympäristön hallinnassa. (Euroopan komission täytäntöönpanoasetus 2022/1426. 12-13.)

2.3 Robottiauto GoPiGo – Raspberry Pi robot



KUVA 2. Robottiauto GoPiGo (Dexter Industries 2025a.)

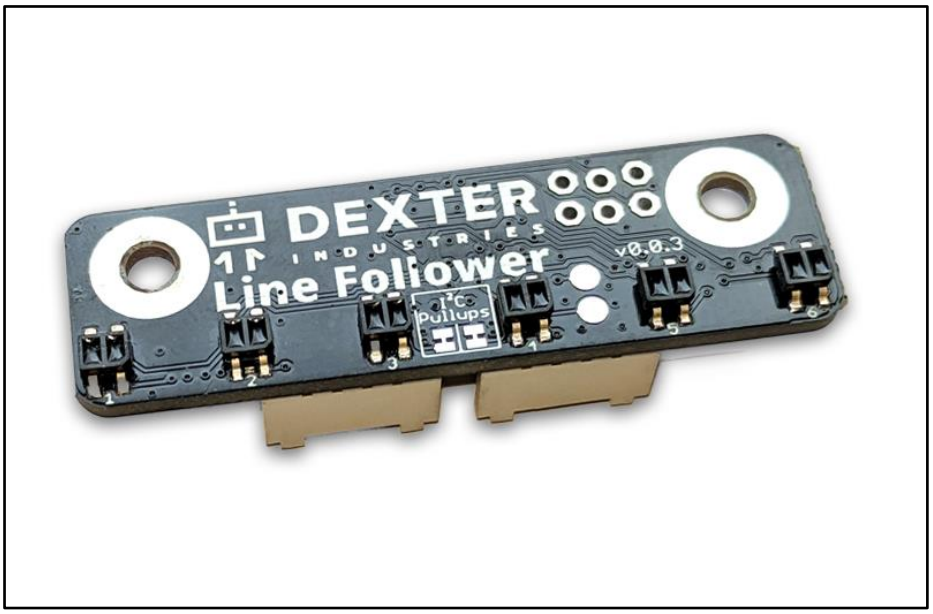
GoPiGo on Dexter Industries:n vuonna 2014 kehittämä ohjelmoitava robottialusta, jonka toiminta pohjautuu Raspberry Pi -pienoistietokoneen ja lukuisten anturien yhteistoimintaan. Dexter Industries:n luoma käyttöjärjestelmä GoPiGo OS tekee robotin ohjelmoimisesta äärimmäisen helppoa, tarjoten aloittelijaystävällisen graafisen Bloxter-ohjelmointikielen sekä edistyneille kehittäjille Python-ohjelmointikielen. GoPiGo on suunniteltu erityisesti opetus- ja harrastuskäyttöön, ja GoPiGo tarjoaakin useita valmiita projekteja, joiden avulla on helppo lähteä tutustumaan robottiin. (Kuva 2.) (GoPiGo 2024a.)

2.3.1 GoPiGo:n anturit ja kamera

GoPiGo robotin toiminta perustuu suurilta osin erilaisten anturien tuottaman tiedon puitteissa toimimiseen. Anturien keräämän tiedon perusteella voidaan ohjelmoida robotti toimimaan halutulla tavalla. GoPiGolle on saatavilla kymmeniä erilaisia antureita, joista jokainen tuo uuden ulottuvuuden robotin kehittämiseen. (GoPiGo 2024 a.)

Line Follower

Anturi, jonka avulla GoPiGo-robotti tunnistaa linjoja. Anturi toimii parhaiten tilanteessa, jossa musta linja on vedetty valkoiselle pohjalle. Linjaseuraaja toimii kuuden infrapuna lähettimen ja vastaanottimen havaitseman valon heijastuman määrän perusteella. Näiden arvojen perusteella robotti osaa päätellä, onko se viivan päällä, vai täytyykö sen kääntyä. (Kuva 3). (Dexter Industries 2024.)

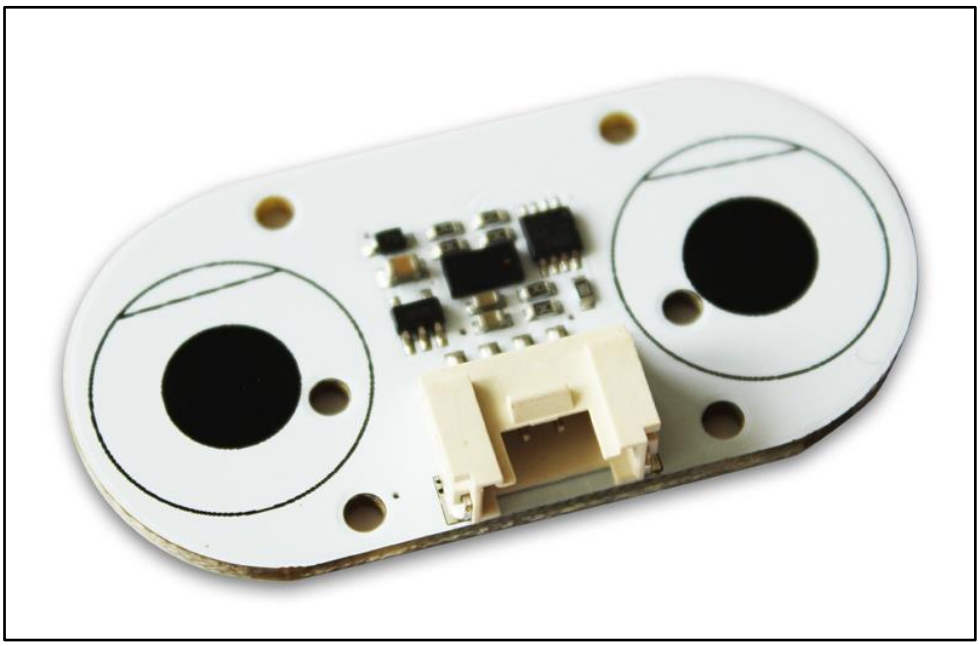


KUVA 3. Line Follower -anturi (Dexter Industries 2025b.)

Distance Sensor

GoPiGon etäisyysanturin toiminta perustuu laser-teknologiaan. Anturi laskee etäisyyden Time-of-Flight (ToF) menetelmällä, eli se lähettää laservalon pulsseja ja laskee etäisyyden niiden palautumisajan perusteella. Tällä tekniikalla saadaan

tarkempia tuloksia, kuin tutulla ultraäänimittauksella. Anturi pystyy mittaamaan etäisyyden tarkasti jopa kahteen metriin. (Kuva 4.) (GoPiGo 2024b.)



KUVA 4. Distance Sensor -anturi (Dexter Industries 2025c.)

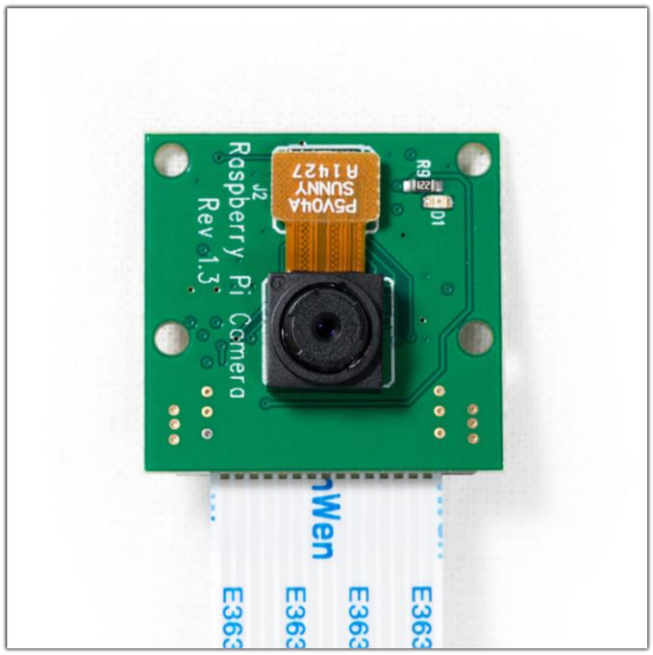
Light & Color Sensor

Valo- ja värianturi toimii RGB-anturin avulla, joka mittaa näkyvän valon kolmen päävärin punaisen, vihreän ja sinisen voimakkuutta. Laskemalla valon värien heijastuman määrän ympäristöstä ja kohteesta, se pystyy päättelemään edessään olevan värin. Anturia pystyy myös käyttämään valon kirkkauden analysoimiseen. (Kuva 5.) (GoPiGo 2024c.)

KUVA 5. Light & Color Sensor -anturi (Dexter Industries 2025d.)

Raspberry Pi Camera

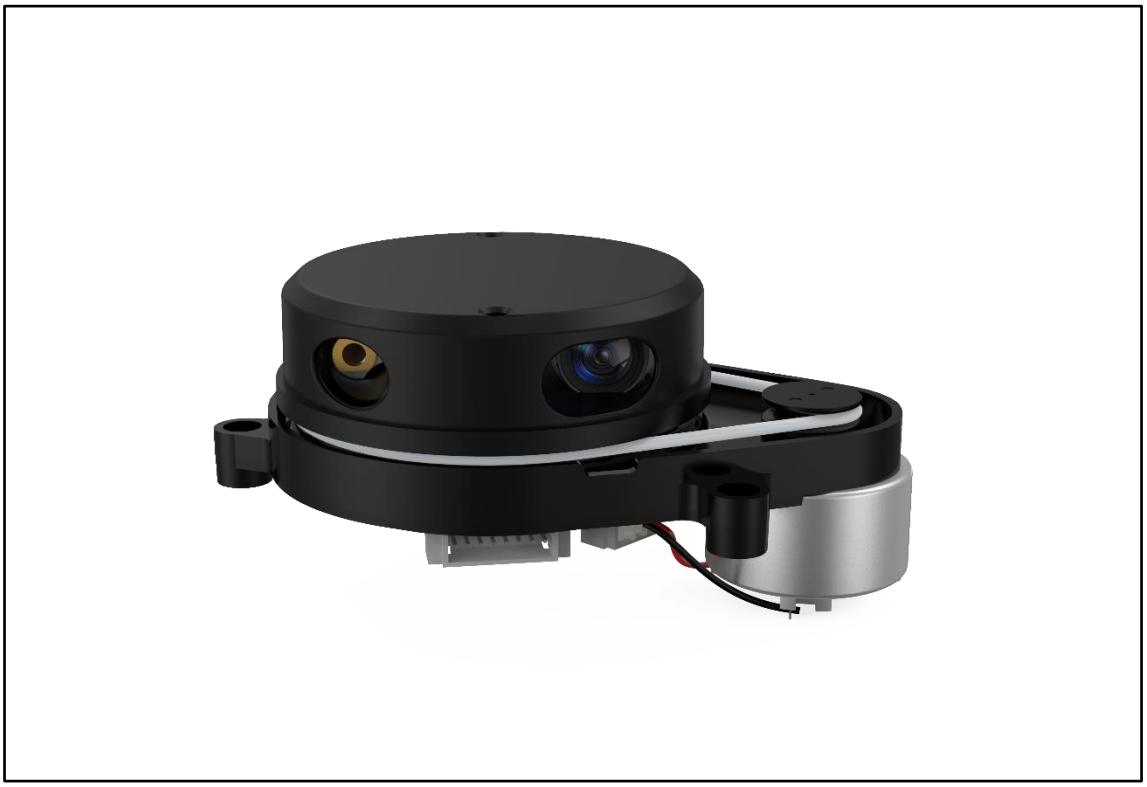
Kameramoduuli, jolla voi ottaa korkealaatuisia videoita ja kuvia. Tukee 1080p30fps, 720p60fps ja VGA90 video formaatteja. Lisäominaisuutena myös mahdollista kuvata hidastettua videokuvaa sekä ajastettua kuvausta (timelapse). (Kuva 6.) (GoPiGo 2024d.)



KUVA 6. Raspberry Pi Camera –komponentti (Dexter Industries 2025e.)

YDLidar X4 Pro

YDLidar X4Pro -valotutka on 360 asteen kaksiulotteinen laseretäisyysmittari, joka on suunniteltu tuottamaan tarkkoja ja luotettavia etäisyysmittauksia erilaisiin käyttötarkoituksiin. Sen toiminta perustuu kolmikulmamittaustekniikkaan ja se pystyy mittaamaan etäisyyksiä välillä 0,12–10 metriä. (Kuva 7.) (Generation Robots 2024.)



KUVA 7. YDLidar X4 Pro -valotutka (Shenzhen EAI Technology Co.,Ltd.)

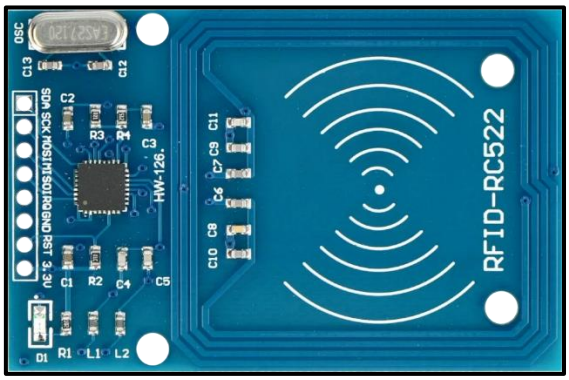
YDLidar:n keskeisimpiä ominaisuuksia ovat mittausnopeus, skannausnopeus ja sen monipuoliset käyttökohteet. Se pystyy suorittamaan 5000 mittaus sekunnissa välillä 6–12 hertsiä, joka mahdollistaa todella tarkan ja reaaliaikaisen ympäristön kartoituksen. Tarkkojen ja nopeiden mittausten ansiosta sitä loistava soveltaa robottien navigointiin ja esteiden välttämiseen, ympäristön skannaukseen sekä 3D-mallinnukseen. (Generation Robots 2024.)

RFID-522

RC522 on edullinen ja hyvin suosittu RFID-lukijamoduuli, joka käyttää 13,56 MHz taajuutta ja tukee ISO 14443A -standardin mukaisia tunnisteita. Sen toiminta

perustuu MFRC522 -piiriin ja se kommunikoi mikrokontrollerien (esimerkiksi Arduino) kanssa SPI-väylän kautta. (Kuva 8.) (Last Minute Engineers 2024.)

Sen lukuetäisyys on noin 2–5 cm ja se soveltuu erinomaisesti erilaisiin kulunvalvonta- ja tunnistussovelluksiin. Sen helppokäyttöisyyden, hinnan ja laajan tuen ansiosta RC522 on suosittu valinta erilaisiin harrasteprojekteihin. (Last Minute Engineers 2024.)



KUVA 8. RC522 RFID-lukija (Cirkit Designer.)

2.3.2 Linux ja Raspberry Pi

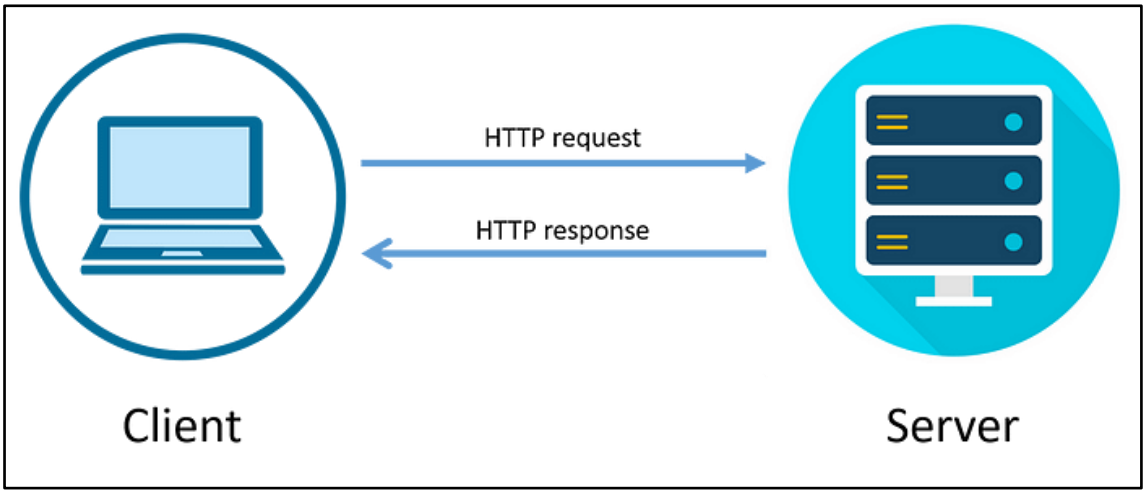
Linux on Linus Torvaldsin kehittämä käyttöjärjestelmäydin, jonka pohjalta tehdään erilaisia Linux-pohjaisia käyttöjärjestelmiä, kuten projektissa käytetty GoPiGo OS. GoPiGo OS antaa robottien ohjelmoinnille kevyen ja vakaan pohjan.

Käyttöjärjestelmä pyörii Raspberry 3b+ yhden piirilevyn pienoistietokoneella, mikä sisältää 64-bittisen neliydin prosessorin ja 1 GB väliaikaista muistia. Koneen käyttämistä varten se on varustelu 4 USB-portilla, 1 micro-USB portilla, Ethernet-portilla sekä Wi-Fi vastaanottimella. Raspberry Pi on täydellinen alusta GoPiGo:n kaltaisten robottien kehitykseen, koska sen laskentateho on riittävä erilaisten anturien datan käsittelyyn ja se kykenee myös käyttämään tekoälyä tehokkaasti. (Kuva 9.) (Raspberry Pi 2024.)

KUVA 9. Raspberry Pi 3 Model B+ (RaspberryPi.)

2.4 Asiakaspalvelin

Asiakas-palvelin-arkkitehtuuri on ohjelmistoarkkitehtuuri, jossa sovelluksen toiminnallisuudet jaetaan kahdelle tai useammalle eri fyysiselle laitteelle hyödyntäen internetin mahdollistamaa kommunikointia. Arkkitehtuurin asiakas (client) lähettää pyyntöjä ja palvelin (server) käsittelee ne ja palauttaa vastaukset. Esimerkiksi asiakas voi lähettää palvelimelle dataa käsiteltäväksi, ja palvelin voi puolestaan käsitellä dataa eri prosesseilla ja lähettää asiakkaalle komentoja, jotka ohjaavat sovelluksen toimintaa aiheuttamatta paljoa kuormitusta. (Kuva 10.) (Haroon 2014, 67–68.)



KUVA 10. Asiakas-palvelin-arkkitehtuuri. (Nath 3.10.2023.)

2.5 Python-ohjelmistokirjastot

Ohjelmistokirjasto antaa ohjelmistokehittäjille käyttöön valmiiksi määriteltyjä tarvittavia ja yleisiä toiminnallisuuksia. Eri kirjastot helpottavat koodin kirjoittamista jakamalla sen toiminnallisuuksia pienempiin osiin. Tämän ansiosta koodin kokonaispituus lyhenee ja muokattavuus paranee. (Simsek 2004.)

Threading

Threading-kirjasto sallii useamman tehtävän suorittamisen rinnakkain Python-ohjelmissa. Threading kirjastoa käytetään, kun halutaan esimerkiksi muodostaa useampia yhteyksiä palvelimeen. (Codecademy 2022.)

Time

Time on Python-ohjelmointikieleen sisältyvä kirjasto, jota ei tarvitse erikseen asentaa. Se mahdollistaa aikaan liittyvät funktiot. Sen avulla pystytään esimerkiksi viivästyttämään toimintoja, tulostamaan aikaa eri muodoissa, tekemään aikaleimoja ja monia muita aikaan liittyviä asioita. (influxdata 2023.)

Asyncio

Asyncio-kirjasto pystyy tekemään funktioista asynkronisia. Sen avulla funktiot saadaan toimimaan ilman kuittauksen odottamista, toisin kuin synkronisissa funktioissa. (Python Software Foundation 2025.)

Queue

Queue-kirjasto pystyy tekemään listan asioista, joista haetaan sinne syötetyt arvot siinä järjestyksessä missä ne on sinne annettu, eli ensimmäisenä tullut haetaan jonosta ensimmäisenä ja niin edelleen.

dotenv

Dotenv-kirjastoon pystytään tallentamaan muuttujia, joita haetaan käyttämällä load_dotenv funktiota. Dotenv:n avulla voidaan myös estää salaisten muuttujien joutumista esimerkiksi GitHub-alustalle. (Python Software Foundation 2025.)

Multiprocessing

Multiprocessin-kirjasto mahdollistaa rinnakkaisen suorituksen luomalla erillisiä prosesseja, mikä kiertää Global Interpreter Lockin (GIL) rajoitukset. Tällä mahdollistetaan moniydinsuorittimen hyödyntämisen prosessori intensiivisissä tehtävissä. (Python 18.4.2025.)

Traceback

Traceback-kirjasto tarjoaa työkalut virheiden jäljitystietojen käsittelyyn ja tulostamiseen. Se mahdollistaa monimutkaisempien virhetilanteiden debuggauksen ja antaa käyttäjälle mahdollisuuden muotoilla virheilmoitukset selkeämmiksi. (Python 18.4.2025.)

Tkinter

Tkinter on Pythonin mukana tuleva graafisen käyttöliittymän kirjasto. Se mahdollistaa ikkunoiden, painikkeiden, syöttökenttien, valikoiden ja muiden peruskomponenttien luomisen, sekä niiden yhdistämisen eri toiminnallisuuksiin.

Base64

Base64-kirjaston avulla mahdollistetaan datan koodaaminen ja purkaminen Base64-muotoon, jota käytetään binääridatan (esim. kuvat) esittämiseen tekstimuodossa. Näin voidaan välittää tiedostoja verkon yli tekstipohjaisessa muodossa. (Python 18.4.2025.)

Struct

Struct-kirjasto mahdollistaa binääridatan pakkaamisen ja purkamisen C-tyylisiin rakenteisiin. Sitä käytetään, kun Pythonin tietoja pitää kirjoittaa tai lukea binäärimuodossa, kuten tiedostoista tai verkkoprotokollista. (Python 18.4.2025.)

Copy

Copy-kirjastosta saadaan työkalut objektien kopiointiin Pythonissa. Kopiot voidaan tehdä joko pinnallisesti (shallow copy) tai syväkopiona (deep copy). Pinnallinen kopio kopioi itse olion, mutta ei sen sisäisiä rakenteita. Syväkopiossa kopioidaan myös kaikki olion sisäiset objektit rekursiivisesti. Päätarkoitus luoda turvallisia kopioita, joiden alkuperäistä tietoa ei muuteta vahingossa. (Python 18.4.2025.)

EasyGoPiGo3

Easygopigo3-kirjasto tarjoaa yksinkertaisen tavan ohjata GoPiGo3-robottia. Kirjasto on suunniteltu aloittelijoille ja opetuskäyttöön, tarjoten selkeän rajapinnan robotin perustoimintojen hallintaan. Kirjastosta löytyy valmiita funktioita robotin liikuttamiseen ja nopeuden säätöön, sekä valojen ja antureiden käyttöön.

MFRC522

Mfrc522 on Python kirjasto, joka mahdollistaa RFID-lukijoiden ohjaamisen Raspberry Pi:llä. Se tukee 13,56 MHz taajuudella toimivia RFID-lukijoita ja mahdollistaa niitä vastaavien tunnisteiden lukemisen sekä uudelleen kirjoittamisen.

Paramiko

Paramiko on Pythonilla toteutettu SSHv2-protokollan toteutus, joka mahdollistaa SSH-toiminnallisuuden sekä asiakkaan, että palvelimen osalta. Se tarjoaa ohjelmallisen tavan muodostaa suojattuja yhteyksiä etäpalvelimiin, suorittaa komentoja sekä siirtää tiedostoja turvallisesti. (Paramiko 2025.)

OpenAI

Openai on OpenAI:n virallinen Python kirjasto, joka mahdollistaa pääsyn OpenAI:n REST API:iin. Se tukee synkronisia sekä asynkronisia pyyntöjä ja sisältää tyypityksen kaikille parametreille ja vastauksille. Keskeisimpiä ominaisuuksia ovat tekstin generointi, kuvien analysointi, tiedostojen hallinta sekä Azure OpenAI-tuki. (openai-python 16.4.2025.)

Socket

Pythonin Socket-kirjasto tarjoaa pääsyn nettiverkon kommunikointiin tarkoitettuihin toiminnallisuuksiin, jolloin tietokoneet voivat kommunikoida. Kirjasto antaa ominaisuudet lähettää ja ottaa vastaan dataa verkkoyhteyden välityksellä. Monet modernit tietokoneiden käyttöjärjestelmät, kuten Unix-pohjaiset käyttöjärjestelmät, Windows ja macOS tukevat kirjaston käyttöä. (Python Software Foundation 22.4.2025.)

String

String-kirjastoa käytetään apuna tekstinkäsittelyyn. Kirjaston tarjoamat toiminnallisuudet tähtäävät merkkijonojen läpi käymiseen ja muokkaamiseen. Esimerkiksi kirjastosta on mahdollista ottaa käyttöön kaikki aakkoset sisältävä merkkijono, jota voi käyttää vertailukohteena tarkistamaan sisältyykö käsiteltävä merkki aakkosiin. (Python Software Foundation 22.4.2025.)

Json

Json-kirjastolla käsitellään dataa, joka on JSON- eli JavaScript Object Notation -muodossa. Kirjasto tarjoaa mahdollisuuden muokata dataa JSON-muotoon ja tästä muodosta muun muassa tekstiksi. Muuntamiseen tarkoitetut

toiminnallisuudet ovat avuksi ohjelmoijille, sillä ne osaavat muuntaa erilaisia tietotyyppejä JSON- muodon ja Python-ohjelmointikielen välillä oikeaoppisesti. (Python Software Foundation 22.4.2025.)

Re

Re-kirjasto on tarkoitettu löytämään ja muokkaamaan kuvioita eli samanlaisia merkkiyhdistelmiä tekstin seasta. Pääkäyttötarkoitus ohjelmoijalle on löytää tiettyjä tiedonpätkiä suuresta määrästä kirjoitettua tietoa. Merkkiyhdistelmien löytämiseen käytetään omaa regular expression -järjestelmää, joka on toimintaperiaatteiltaan kuin ohjelmointikieli. (Python Software Foundation 22.4.2025.)

NetworkX

NetworkX-kirjasto on avoimenlähteen ohjelmistokirjasto, jonka avulla pystytään luomaan tietorakenteita graafeille. Kirjastossa monia standardisoituja graaffisia algoritmeja. sillä myös pystytään luomaan tavallisia graaffeja, satunnaisia graaffeja ja synteettisiä verkkoja. Kirjasto antaa myös määritellä minkälaisia solmut ja reunat ovat. (NetworkX 2025.)

OS

Os-kirjastoa tarjoaa siirrettävän tavan käyttää käyttöjärjestelmäriippuvaisia toiminnallisuuksia. Eli koodi toimii mahdollisimman monella alustalla samalla tavalla. Ominaisuuksia os-kirjastolla on esimerkiksi tiedoston luominen ja katsominen, tiedostopolkujen käsittely, kaikkien rivien lukeminen komentorivillä annetuista tiedostoista ja väliaikaisten tiedostojen luonti. (Python Software Foundation 2025.)

Matplotlib

Matplotlib on kattava kirjasto, jolla voidaan luoda staattisia, animoituja ja interaktiivisia visualisointeja. Matplotlib hyödyntää kolmasien osapuolten laajaa valikoimaa, jotka on rakennettu matplotlibi:n päälle. (Matplotlib 2025.)

Typing

Typing-kirjastolla luodaan staattinen tyypitys Python-ohjelmointikieleen, joka on perinteisesti dynaamisesti tyypitetty. Se mahdollistaa tyyppi vihjeiden käytön, jonka avulla koodista tehdään luotettavampaa ja ymmärrettävämpää. Sen avulla koodista tehdään helpommin ymmärrettävää ja koodi selittää itseään, kun funktioiden tyypit näkyvät suoraan. (Medium 2024.)

2.6 Ohjelmointityökalut

Ohjelmistokehitys vaatii monenlaisia ohjelmia, jotka helpottavat koodin kirjoittamista, projektin eri versioiden hallinnoimista ja eri laitteiden saumatonta hallinnointia. Näiden ohjelmien avulla nopeutetaan ja helpotetaan tiimien, sekä yksilöiden ohjelmistokehitystä.

2.6.1 Versiohallinta

Versionhallintajärjestelmät ovat ohjelmistoja, jotka seuraavat ja hallitsevat erilaisten tiedostojen muutoksia. Tällaiset ohjelmat mahdollistavat useiden kehittäjien yhtäaikaisen työskentelyn, kooditiedostojen aiempien versioiden tarkastelun sekä mahdolliset tiedostojen palauttamiset aiempiin versioihin. Versiohallinnan suuren etuna on se, että tiedostot ovat helposti kaikkien määrätyn ryhmän jäsenten saatavilla, eikä yksittäisen laitteen rikkoutuminen hävitä tiedostoja.

Git

Git on ohjelmistokehityksen tarpeisiin suunniteltu versionhallintajärjestelmä. Sen avulla kehittäjät voivat helposti seurata kooditiedostoissa tapahtuvia muutoksia, hallita eri versioita sekä työskennellä tehokkaasti muiden kehittäjien kanssa. Git tallentaa muutokset paikallisesti, mikä mahdollistaa offline -työskentelyn ja nopean versionhallinnan. Git:n perusominaisuudet ovat haarojen (branch) luominen ja käyttö sekä kooditiedostojen yhdistäminen (merge). Haaroilla mahdollistetaan saman kooditiedoston kehittäminen rinnakkaisissa versioissa ja yhdistämisellä sulautetaan rinnakkaiset versiot yhdeksi kooditiedostoksi. (Git 2024.)

GitHub

GitHub on Git-versionhallintajärjestelmään perustuva verkkopalvelu, joka tarjoaa pilvipohjaisen ympäristön kooditiedostojen hallintaan, jakamiseen ja yhteistyöhön. Se toimii Git:n kanssa ja mahdollistaa muun muassa etäsäilöjen(repository) hallinnan, ongelmien seurannan (issue labels) ja muutospyynnöt (pull request). GitHub mahdollistaa myös sosiaalisen kanssakäymisen kehittäjien kesken tarjoamalla mahdollisuuden tutustua muiden projekteihin ja tehdä yhteistyötä maailmanlaajuisesti. Palvelu on suosittu niin yksilöiden, kuin isojen yritysten keskuudessa. (Brown 21.9.2016.)

2.6.2 Etähallinta

Etähallintatyökalut (Remote Access Tools, RAT) ovat ohjelmia, jotka mahdollistavat etäyhteyden muodostamisen erilaisiin laitteisiin, verkkoihin ja palvelimiin. Niiden avulla käyttäjä voi käyttää toista laitetta ilman fyysistä pääsyä laitteeseen. Tämä helpottaa työskentelyä useampien laitteiden parissa kerralla, koska useampaa laitetta voi hallita yhdeltä koneelta. (Ballejos 25.10.2024.)

SSH

SSH (Secure Shell) on yleisesti tietokoneilta valmiiksi esiasennettuna löytyvä protokolla (Windowsilla 10 ja 11), jonka avulla mahdollistetaan turvallisen etäyhteyden muodostaminen tietokoneiden välillä salaamalla tiedonsiirron. Sitä käytetään yleisesti palvelimien etähallintaan ja tiedostojen siirtämiseen. SSH perustuu hallinta komentojen lähettämiseen etälaitteelle turvallisesti. SSH:n turvallisuus perustuu kryptografiaan, jonka avulla varmistetaan yhteyden osapuolten aitous, sekä tiedonsiirron luottamuksellisuus. Windows-käyttöjärjestelmällä SSH-yhteys on helppo muodostaa käyttämällä komentokehotetta (cmd). (Cloudflare 2024.)

WinSCP

WinSCP on Windows-käyttöjärjestelmille kehitetty ohjelma, joka mahdollistaa turvallisen tiedonsiirron kahden tietokoneen välillä. WinSCP tukee monia salaus- ja todennusmenetelmiä. Se hyödyntää SSH-protokollaan perustuvia SFTP- ja SCP-

protokollia salatun yhteyden muodostamiseen, mutta myös salaamaton yhteys on mahdollista FTP sekä WebDAV -protokollilla.

WinSCP on todella käyttäjäystävällinen sen selkeän graafisen käyttöliittymän ansiosta. Se tarjoaa Commander-näkymän, jossa yhteydessä olevien koneiden kansiot ja tiedostot näkyvät selkeästi rinnakkain sekä Explorer-näkymän, joka muistuttaa normaalia Windowsin resurssienhallintaa.

Tiedostojen siirto tapahtuu kätevästi vetämällä ja pudottamalla tiedostot kansiosta toiseen. WinSCP sisältää myös integroidun tekstieditorin, jonka avulla käyttäjä voi muokata etätiedostoja suoraan ohjelman sisällä. (IONOS editorial team 12.5.2023.)

2.6.3 Ohjelmointiympäristöt ja koodieditorit

Koodieditorit ja ohjelmointiympäristöt ovat ohjelmistokehityksen keskeisiä työkaluja, mutta ne eroavat toisistaan käyttötarkoituksensa ja ominaisuuksiensa perusteella. Koodieditorit ovat tarkoitettu kevyempiin ja joustavimpiin tarpeisiin, kun taas ohjelmointiympäristöt ovat tarkoitettu monipuolisempiin ja laajempiin projekteihin.

Visual Studio Code

Visual Studio Code (VS Code) on Microsoftin kehittämä kevyt, mutta tehokas koodieditori. Se tukee monia ohjelmointikieliä ja kehitystyökaluja. VS Code on avoimen lähdekoodin ohjelma ja on saatavilla Windows-, macOS- ja Linux-käyttöjärjestelmille. VS Coden keskeisiä ominaisuuksia ovat:

- **Älykäs koodinmuokkaus**: Editori tarjoaa automaattisen täydennyksen (IntelliSense) ja virheiden tarkistuksen.
- **Integroitu terminaali**: Mahdollisuus käyttää komentoriviä suoraan editorissa.
- **Laajennettavuus**: VS Code tarjoaa sisäisen kaupan, josta voi ladata haluamiaan lisäominaisuuksia editoriin, kuten erilaisia kehitystyökaluja.

Kevyet järjestelmävaatimukset, ääretön laajennettavuus ja monipuoliset ominaisuudet tekevät VS Codesta yhden suosituimmista koodieditoreista. (Visual Studio Code 2024.)

Thonny IDE

Thonny on integroitu ohjelmointiympäristö (IDE), joka on suunniteltu Python-ohjelmoinnin aloittelijoille. Sen tarjoama selkeä ja yksinkertainen käyttöliittymä auttaa käyttäjää keskittymään ohjelmointiin, ilman monimutkaisia asetuksia. Thonnyn keskeiset ominaisuudet ovat:

- **Sisäänrakennettu Python-tuki**: Ei tarvetta asentaa Pythonia erikseen.
- **Portaittain etenevä debuggaus**: Koodi voidaan käydä läpi rivi kerrallaan, mikä helpottaa sen ymmärtämistä.
- **Helppo pakettien asentaminen**: Erilaisten moduulien ja kirjastojen asentaminen on tehty erittäin helpoksi.

Thonny on hyvä valinta uusille Python-ohjelmoijille, sen helposti lähestyttävän käyttöliittymän ansiosta. Thonny on myös oletuksena Raspberry Pi -laitteissa, joka tekee siitä ideaalin Python editorin Raspberry Pi -projekteissa. (Thonny 2024.)

2.6.4 Ohjelmointikielet

Ohjelmointi tarkoittaa tietotekniikassa prosessia, jossa tietokoneelle annetaan toimintaohjeita ohjelmointikielen avulla. Eri ohjelmointikielet on kehitetty helpottamaan näiden ohjeiden kirjoittamista ja ymmärrettävyyttä. On olemassa monia ohjelmointikieliä ja niitä kehitetään jatkuvasti uusien toimintojen tai tekniikoiden takia. Ohjelmointikielten valinta perustuu siihen, mihin niitä halutaan käyttää, sillä jotkut tukevat joitakin toimintoja paremmin kuin muut. Alla olevassa kuvassa on esitetty, kuinka kolmella eri ohjelmointikielellä voidaan toteuttaa sama tehtävä (Kuva 11). (TIM 2024.)

KUVA 11. Ohjelmointikieliesimerkkejä. (Python4dev 14.9.2024.)

Python ohjelmointikieli

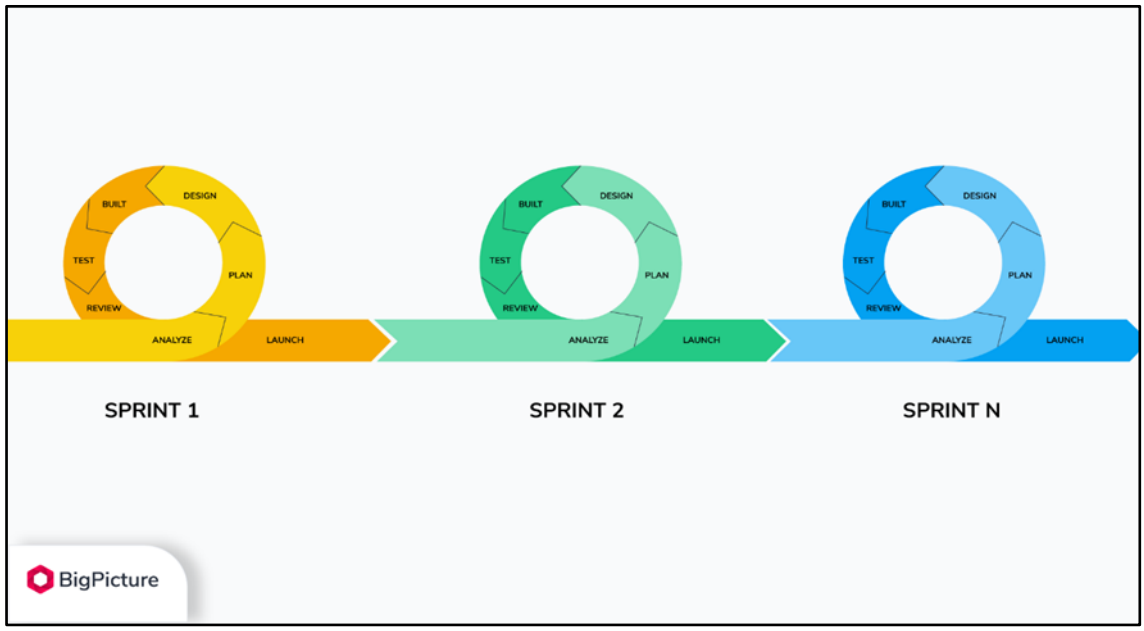
Python on vuonna 1991 kehitetty ohjelmointikieli, jota voidaan käyttää moniin asioihin, kuten verkkosivujen palvelinten kehittämiseen, ohjelmistokehitykseen, matemaattisiin laskelmiin sekä järjestelmän komentosarjakielten suorittamiseen. Python-ohjelmointikielen etuina on, että se toimii monilla eri käyttöjärjestelmillä ja sen yksinkertainen syntaksi eli lauseoppi helpottaa ohjelmien kehittämistä, sillä se ei vaadi niin pitkiä koodirivejä kuin muut ohjelmointikielet. (W3schools 2025.)

2.7 Scrum-projektinhallintamenetelmä

Scrum menetelmä on vuonna 1995 maailmalle tuotu vaativien ongelmien ratkaisuun tarkoitettu kehitysmenetelmä. Scrum-tiimi koostuu kolmesta ryhmästä, johon kuuluu tuoteomistaja, Scrum master ja sovelluskehittäjät.

Scrum ei ole lyhenne mistään vaan se juontaa juurensa rugbyyn, jossa pelaajat käyvät keskustelua pelin etenemisestä ja he käyttävät Scrumia nimityksenä tälle keskustelulle.

Scrum menetelmän päätarkoituksena on tehdä työtä pieni osa kerrallaan tiiminä ja kun tiimi saavuttaa sprintin eli määrätyn ajanjakson tavoitteet, voidaan aloittaa suunnittelemaan seuraavan sprintin tavoitteita. Scrum on siis käytännössä silmukka, jota toistetaan, kunnes se saavuttaa projektin halutun lopputuloksen. (Scrum 2025.) Kuvassa 12 on esiteltynä yksi tapa toteuttaa Scrum-projektin sprintit. Tämä ei ole ainut tapa, vaan Scrum-menetelmää voidaan muokata tarpeiden mukaan ja kokeneemmat Scrum-tiimit voivat jättää tiettyjä vaiheita pois.



KUVA 12. Sprintien kulku Scrumissa (BigPicture 23.5.2022.)

2.7.0 Tuotteen ominaisuuslista – Product Backlog

Product Backlog on lista asioista, mikä kehittyy koko ajan. Listaan määritellään asioita, joita tarvitaan tuotteen parantamiseksi. Product backlogista on ainoa työnlähde, mistä Scrum-tiimi saa itsellensä tehtävät, joita he lähtevät tekemään.

Backlogin kohteet, jotka voidaan toteuttaa Scrum-tiimin toimesta yhden päivän aikana ovat valmiita valittavaksi sprint planning-tapahtumassa. Kohteiden kuvaus tarkennetaan tarkennusprosessin kautta, jolloin kohteet voidaan pilkkoa pienemmiksi ja selkeämmiksi tehtäviksi. Tämä on jatkuva prosessi yksityiskohtien lisäämiseksi, kuten kuvaus kohteesta, järjestys ja koko. Tuoteomistaja voi auttaa kehittäjiä ymmärtämään ja valitsemaan kohteita, mutta vastuu niistä on loppukädessä kehittäjillä.

Scrum-tiimi työskentelee saavuttaakseen tuotteen tavoitteen, joka sisältyy product backlog:iin. Backlog rakentuu tämän tavoitteen ympärille, jotta pystytään määrittelemään, mikä täyttää tuotteen tavoitteen. Se on pitkän aikavälin tavoite ja scrum-tiimin on saavutettava tai hylättävä tavoite, ennen kuin voidaan aloittaa seuraava tavoite. (Schwaber & Sutherland 2020.)

2.7.1 Kehityssyklin ominaisuuslista – Sprint Backlog

Sprint backlog koostuu sprintin tavoitteista, valituista product backlog -kohteista ja toteutussuunnitelmasta. Eli Miksi sprintti tehdään, mitä sprintissä tehdään ja miten sprintti toteutetaan.

Sprint backlog-kehittäjille tarkoitettu suunnitelma, jotta sprintin tavoitteet saadaan tehtyä. Sprint baclog:n avulla saadaan reaaliaikainen ja näkyvä kuva siitä, mitä kehittäjät suunnittelevat saavuttaakseen sprintin tavoitteet. Sprint backlog:ia päivitetään sprinttien aikana sitä mukaan, kun opitaan lisää. Siinä tulisi olla tarpeeksi yksityiskohtia, jotta edistystä pystyttäisiin tarkastelemaan päivittäisten Scrum tapaamisten aikana.

Sprintin tavoitteet on lista tehtäviä, joita sprinttien aikana pyritään toteuttamaan. Sprintin tavoitteet antavat kehittäjille joustavuutta, kunhan sprintin tavoitteet saavutetaan. Sprinttien tavoitteet luovat myös johdonmukaisuutta ja rohkaisee Scrum-tiimiä toimimaan yhdessä.

Sprint baclogiin kuuluu increment eli lisäys, joka on konkreettinen askel kohti tuotteen tavoitetta. Increment on arvokas tuotteelle vain, jos se on käyttökelpoinen. Sprintin aikana voidaan luoda useita incrementtejä. Valmiit incrementit esitellään Sprint review tapahtumassa. Incrementeiksi ei voida hyväksyä lisäyksiä, jotka eivät tavoita Definition of Done:ssa määriteltyjä kriteereitä.

Definition of Done on muodollinen kuvaus siitä, milloin increment täyttää tuottee laadulliset vaatimukset. Incrementit ovat Product Backlogin kohteita, jotka ovat täyttäneet Definition of Donen. Definition of Done selkeyttää kaikille, mitä tarkoitetaan valmiilla työllä. Jos kohde ei täytä Definition of Donea, se palautetaan takaisin Product Backlogiin. Definition of Done voi olla yrityksen tai Scrum-tiimin

itse määrittelemä. Kehittäjien on noudatettava Definition of Donea. (Schwaber & Sutherland *2020.*)

2.7.2 Kehityssyklien suunnittelu – Sprint Planning

Sprint planning on sprinttien ensimmäinen vaihe, jossa käynnistetään sprintti. Siinä määritellään, mitä sprintin aikana tehdään. Sprint planningiin osallistuu koko Scrum-tiimi. Product Owner katsoo, että kyseiseen sprinttiin valitut Product Backlogin kohteet ovat valmiina käsiteltäväksi ja ovatko ne olennaisia tuotteen tavoitteisiin nähden.

Sprint planningissä on kolme aihetta: Miksi tämä Sprintti on hyödyllinen, mitä tämän sprintin aikana voidaan tehdä ja miten työt toteutetaan.

Ensimmäisen aiheen kohdalla Product Owner ehdottaa, kuinka tuotteen arvoa ja hyödyllisyyttä pystyisi nostamaan sen hetkisen sprintin aikana. Sen jälkeen Scrum-tiimi määrittelee sprintin tavoitteet, jotka kertovat osakkaille, miksi sprintti on hyödyllinen heille. Sen jälkeen tavoite lukitaan.

Toisen aiheen kohdalla Scrum-tiimin kehittäjät valitsevat Product Backlogista kohteita sprinttiin tuoteomistajan kanssa. Kehittäjät voivat tämän prosessin aikana tarkentaa kohteita. Aiempien suoritusten perusteella kehittäjät voivat miettiä, montako kohdetta voidaan realistisesti toteuttaa sprintin aikana.

Kolmannessa aiheessa suunnitellaan, mitä työvaiheita eri kohteiden kanssa tarvitaan, jotta ne täyttävät Definition and Done:n, joka on muodollinen kuvaus laatuvaatimukset täyttävästä tuotteesta. Product Baclogin-kohteet pilkotaan pienemmiksi osiksi; yleensä yhden työpäivän mittaiseksi tai lyhyemmiksi. Kehittäjät itse päättävät työtavan kunkin kohteen kohdalla. Muut eivät ole ohjaamassa kehitystiimin työskentelytapoja.(Schwaber & Sutherland *2020.*)

2.7.3 Kehityssyklien tarkastelu - Sprint Review

Sprint Reviewissä tarkastellaan sprintin tuloksista ja siellä päätetään tulevista muutoksista. Scrum-tiimi esittelee sidosryhmille sprintin tulokset ja jatkavat keskustelemaan tuotteen tavoitteista.

Tapahtuman aikana Scrum-tiimi ja sidosryhmä keskustelevat, mitä saavutettiin sprintissä ja mikä on muuttunut heidän ympäristössään. Tämän perusteella osallistuja päättävät, mitä tehdään seuraavaksi. Product Backlogia voidaan muuttaa uusien mahdollisuuksien saavuttamiseksi. Sprint Review on työtilaisuus eikä pelkkä esitys, joten tiimin tulisi välttää sitä tilaisuuden muuttumista esitykseksi.

Sprint Review on toiseksi viimeinen vaihe ja sen pituus tulisi olla mukautettu siihen, kuinka pitkiä sprintit ovat. Yhden kuukauden sprintti voi olla neljän tunnin mittainen ja lyhyemmissä aikaa käytetään vähemmän. (Schwaber & Sutherland 2020.)

2.7.4 Kehityssyklien arviointi – Sprint Retrospect

Sprint Retrospect -tapahtumassa suunnitellaan keinoja laadun ja tehokkuuden parantamiseksi. Scrum-tiimi tarkastelee edellistä sprinttiä ja kuinka se sujui eri osa-alueilla. Osa-alueisiin kuuluu yksilö, vuorovaikutus, prosessit, työkalut ja Definition of Done. Se mitä osa-alueita tarkastellaan, liittyy siihen, minkälainen työn luonne on ollut. Oletukset, jotka johtivat harhaan, tunnistetaan ja niiden alkuperät tutkitaan. Scrum-tiimi keskustelee, mikä meni hyvin ja mitä ongelmia tuli vastaan ja kuinka ongelmat ratkaistiin sprintin aikana.

Scrum-tiimi tunnistaa parhaimmat muutokset, parantaakseen tuotteliaisuutta. Merkityksellisimmät muutokset toteutetaan mahdollisimman nopeasti. Ne voidaan lisätä jopa Sprint Backlogiin seuraavaan sprinttiin. Sprint Retrospect on rajattu kolmeen tuntiin kuukauden mittaisissa sprinteissä ja lyhyemmissä sprinteissä aika on lyhyempi riippuen sprintin pituudesta. (Schwaber & Sutherland 2020.)

3 KEHITYSTYÖ SPRINTEITTÄIN

Tässä osiossa esitellään, miten opinnäytetyön projektiosuus eteni sprinteittäin. Projekti koostui yhteensä 12 sprintistä, joista kaksi ensimmäistä (-1 ja 0) olivat projektiin valmistelevia sprinttejä. Sprintit on esitetty selkeällä kolmijakoisella rakenteella: tavoitteet, tehdyt asiat ja tulokset.

3.1 Sprint -1 (20.1.2025–26.1.2025)

Tavoitteet

Ennen opinnäytetyön virallista aloittamista määriteltiin projektin keskeiset tavoitteet, tarvittavat hankinnat sekä suunnitellut ominaisuudet. Tavoitteena oli kehittää osittain autonomisesti toimiva robottiauto, jonka autonominen toiminta perustuisi linjaseuraajaan, etäisyysanturiin, kameraan sekä värianturiin. Näiden antureiden välittämän datan perusteella robotin tuli pystyä toimimaan halutulla tavalla erilaisissa tilanteissa.

Tehdyt asiat

Projektin pohjaksi hyödynnettiin aiemmassa yritysprojektissa hankittua kokemusta GoPiGo-robotista. Näin ollen projekti ei alkanut täysin tyhjästä, vaan käytettävissä oli jo valmiiksi testattuja ominaisuuksia. Projektin alussa kartoitettiin tarvittavat lisähankinnat. Mahdollisiksi hankinnoiksi tunnistettiin väri- ja valoanturi sekä IMU (kiihtyvyys)-anturi. IMU-anturista kuitenkin luovuttiin melko varhaisessa vaiheessa, sillä sen ei katsottu olevan tarpeellinen projektin tavoitteiden kannalta. Linjaseuraaja, etäisyysanturi ja kamera löytyivät jo entuudestaan käytettävistä laitteista.

Tulokset

Projektin lähtökohtiin ja olemassa oleviin komponentteihin perustuen saatiin muodostettua selkeä kokonaiskuva robotin tulevista ominaisuuksista. Osaa ominaisuuksista oli jo testattu ja validoitu aikaisemmassa projektissa, mikä

mahdollisti tehokkaan etenemisen suunnitteluvaiheessa. Projektin alkuvaiheen suunnittelussa luotiin vahva perusta robottiauton jatkokehitykselle ja myöhemmälle toteutukselle.

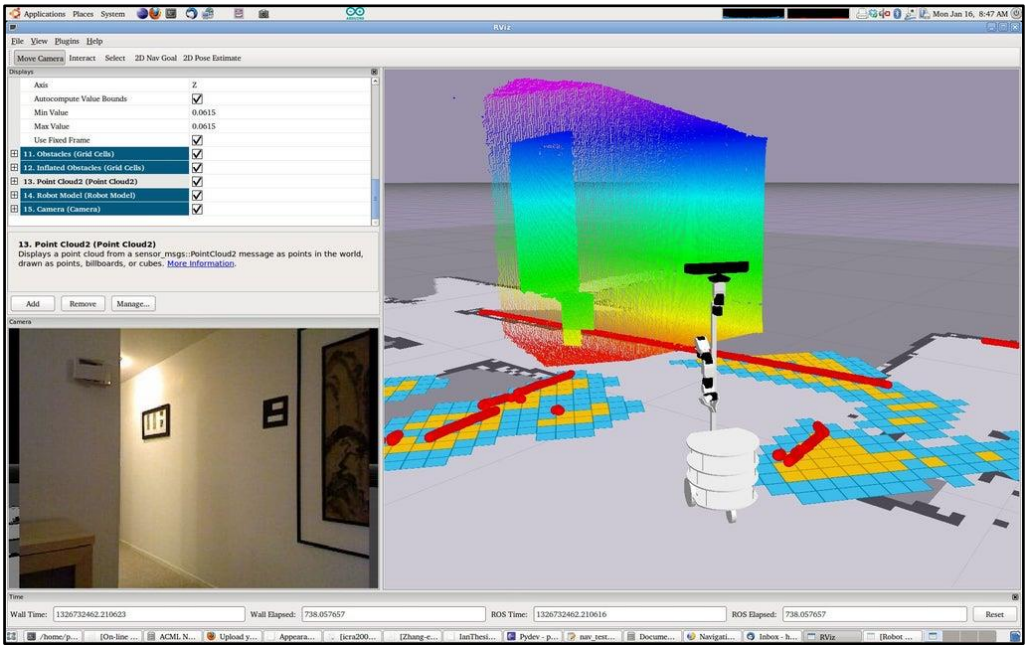
3.2 Sprint 0 (27.1.2025–2.2.2025)

Tavoitteet

Projektin virallisen aloituksen yhteydessä asetettiin tavoitteeksi selvittää mahdollisia toteutustapoja robotin autonomisen toiminnan mahdollistamiseksi. Tärkeänä osana oli kehittää menetelmä, jolla robotti voisi hyödyntää antureiden tuottamaa dataa älykkääseen ohjaamiseen. Lähtökohtana oli rakentaa rata, jota pitkin robotti voisi kulkea linjaseuraajaa hyödyntäen. Radan varrella tapahtuvaa kääntymistä suunniteltiin ohjattavan värianturin avulla ja esteiden tunnistamista varten aiottiin käyttää etäisyysanturia sekä kameraa. Robotin oli tarkoitus olla yhteydessä palvelimeen, jonka kautta sitä ohjattaisiin.

Tehdyt asiat

Sprintin aikana aloitettiin tiedonhaku eri teknisistä ratkaisuista autonomisuuden toteuttamiseksi. Aluksi tutkittiin mahdollisuutta käyttää valotutkaa ja ROS-ohjelmistokirjastoa ratakartoitukseen (kuva 13). Tämän ratkaisun nähtiin tarjoavan tarkkaa dataa, jonka avulla robotti voisi tehdä oikea-aikaisia käännöksiä ja pysähdyksiä. Valotutkan käyttöönoton arvioitiin kuitenkin vaativan robotin käyttöjärjestelmän vaihtamista, mikä todettiin liian työlääksi ja epäkäytännölliseksi projektin aikataulun ja laajuuden kannalta. Kun valotutkasta luovuttiin, ryhdyttiin kartoittamaan vaihtoehtoisia menetelmiä. Tiedonhaun aikana löydettiin Dijkstran algoritmi, joka arvioitiin soveltuvaksi ratkaisuksi robotin reittilogiikan muodostamiseen. Lisäksi sprintin aikana määriteltiin projektin aikataulu, johon sisällytettiin työn aloitus- ja palautuspäivämäärät sekä sprinttien kestot.



KUVA 13. Esimerkki ympäristön 3D-visualisoinnista ROS-kirjastolla. (ZRob314 2025.)

Tulokset

Sprintin aikana muodostettiin kokonaisvaltainen käsitys robotin älykkään ohjauksen vaatimuksista ja teknisistä haasteista. Vaikka alkuperäinen ajatus valotutkan hyödyntämisestä osoittautui liian monimutkaiseksi, löydettiin sen tilalle lupaava vaihtoehto Dijkstran algoritmista. Näin pystyttiin etenemään ratkaisun suuntaan, jossa robotin liikkumista voidaan ohjata selkeän algoritmipohjaisen logiikan avulla ilman käyttöjärjestelmän vaihtoa tai raskasta lisälaitteistoa.

3.3 Sprint 1 (3.2.2025–9.2.2025)

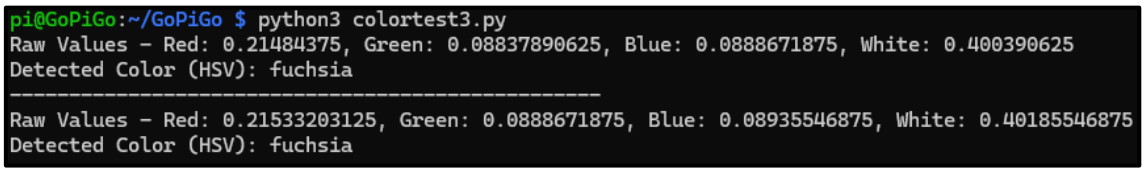
Tavoitteet

Sprintin tavoitteena oli syventää ymmärrystä Dijkstran algoritmin toiminnasta sekä värianturin käytöstä. Tavoitteena oli luoda algoritmista sellainen versio, joka pystyisi tukemaan radan kartoitusta projektin tarpeisiin. Värianturin osalta pyrittiin selvittämään, miten sitä voitaisiin hyödyntää robotin sijainnin määrittämiseen radalla, ja miten luettavat värit saadaan ohjelmallisesti tunnistettua luotettavasti.

Tehdyt asiat

Sprintin aikana perehdyttiin Dijkstran algoritmin perusteisiin. Algoritmin toiminta-logiikkaan kuuluu lähimmän solmun valinta lähtösolmusta ja etäisyyksien päivitys naapurisolmuille. Tämä prosessi toistetaan, kunnes lyhin reitti määränpäähän on selvitetty. Dijkstran algoritmia testattiin hyödyntämällä GitHubista löydettyä demonstroivaa ohjelmaa sekä Datacamp-verkkosivuston tarjoamia tutoriaaleja. Näiden pohjalta kehitettiin ensimmäinen demoversio, jossa käyttäjä pystyi syöttämään alku- ja loppusolmut, joiden perusteella ohjelma laski lyhimmän reitin solmujen välillä ja tulosti sen näytölle.

Värianturin osalta aloitettiin ohjelmistokirjaston tutkiminen ja sen tarjoamien funktioiden testaaminen. Anturin todettiin kykenevän tunnistamaan joko valmiiksi määritettyjä värejä tai mukautettuja värejä, jotka voitiin määritellä lukuarvojen perusteella (kuva 14). Projektin vaatimukset mahdollistivat valmiiden värien käytön, jolloin mukauttamiseen ei nähty tarvetta. Testien perusteella värianturi tunnisti värit kohtuullisella tarkkuudella. Anturin käyttötarkoitukseksi hahmoteltiin solmujen tunnistaminen, jonka avulla voitaisiin pitää kirjaa robotin sijainnista.



KUVA 14. Värianturin havaitsema väri.

Tulokset

Sprintin lopputuloksena saatiin toimiva demoversio Dijkstran algoritmista, joka pystyi laskemaan lyhyimmän reitin annetulla solmuverkostolla. Tämä versio loi perustan projektin radan logiikan jatkokehitykselle. Värianturin osalta saatiin varmistettua sen perustoimivuus, mutta haasteeksi nousi sensorin tarkkuus käytännön tilanteissa. Värillisten merkkien lukeminen vaatii robotilta täsmällistä sijaintia ja suuntausta, jota ei voida taata, koska robotti ei liiku täydellisen suoraviivaisesti. Näin ollen värianturin käyttö vaatii jatkokehitystä ja lisää testausta luotettavan toiminnan varmistamiseksi.

3.4 Sprint 2 (10.2.2025–16.2.2025)

Tavoitteet

Tämän sprintin tavoitteena oli suunnitella lopullinen fyysinen rata robotille, kehittää robotin kääntymislogiikka risteystilanteita varten sekä ratkaista robotin sijainnin kirjaamiseen liittyvät haasteet. Tarkoituksena oli myös luoda toimiva demoversio, joka pystyy kulkemaan annetun reitin aloitus- ja lopetussolmun välillä oikeita reittivalintoja tehden.

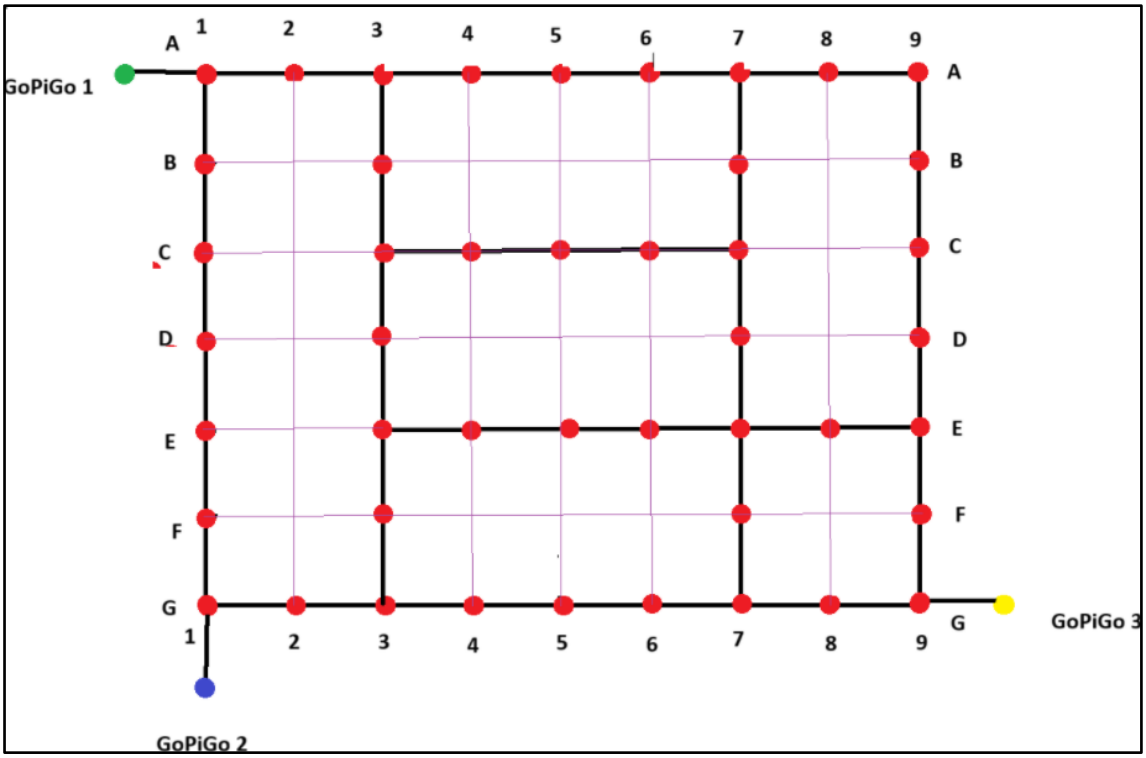
Tehdyt asiat

Radan suunnittelussa päädyttiin suorakulmaiseen muotoon, jotta siihen voitiin luoda mahdollisimman selkeä ja yksinkertainen koordinaatisto. Rata rakennettiin 45 pisteestä (solmusta) ja niiden välisistä suorista (reunoista). Fyysinen toteutus tehtiin paperille: reitit merkittiin mustalla sähköteipillä ja solmukohdat värillisillä tarroilla (kuva 15).

KUVA 15. Rata ilman väritarroja.

Robotin kääntymislogiikka risteyksissä osoittautui haastavaksi, sillä robotin tuli jokaisessa risteyksessä tietää, mistä se on tullut, ja mihin suuntaan sen tulee jatkaa. Ongelma ratkaistiin siten, että robotti laski jokaisen ylittämänsä solmun. Tämän avulla sen sijainti radalla saatiin määritettyä. Kääntymissuunnan

määrittämisessä siirryttiin vertaamaan solmujen kirjaimia ja numeroita keskenään (kuva 16). Näin pystyttiin toteuttamaan kääntyminen yksinkertaisilla vertailuoperaatioilla (esimerkiksi suurempi/pienempi), jolloin robotti osasi valita oikean suunnan.



KUVA 16. Rata koordinaatistona.

Robotin sijainnin kirjaamisessa hyödynnettiin värianturia. Solmukohtiin asetettiin yksi, helposti tunnistettava väri, jonka perusteella robotti havaitsi ylittäneensä solmun. Värianturin kanssa kuitenkin ilmeni edelleen ongelmia; värit joko tunnistettiin virheellisesti tai ei tunnistettu lainkaan, mikä johtui lukunopeudesta ja robotin liikkeestä. Ongelman ratkaisemiseksi päätettiin kokeilla selkeämpiä ja kontrastiltaan voimakkaampia värejä, jotka todennäköisesti parantaisivat anturin lukutarkkuutta.

Tulokset

Sprintin päätteeksi saatiin käyttöön demoversio, jossa robotti pystyi ajamaan annetun reitin aloitus- ja lopetussolmujen välillä. Robotti liikkui kiinteissä 12 cm pituisissa matkoissa solmulta toiselle ja suoritti oikeat käännökset perustuen sijaintiinsa ja reittitietoon. Vaikka värianturin toiminnassa ilmeni edelleen haasteita,

saavutettiin tärkeä välitavoite, sillä robotti pystyi jo seuraamaan suunniteltua reittiä oikein.

3.5 Sprint 3 (17.2.2025–23.2.2025)

Tavoitteet

Sprintin tavoitteena oli kehittää ensimmäinen raakaversio asiakas–palvelinarkkitehtuurista, määrittää robotissa käytettävät anturit robotin porttien puitteissa sekä suunnitella, miten radan eri tilanteet merkitään värein. Radan merkkivärien oli tarkoitus kattaa aloituspiste, risteykset ja pysähdyspaikat. Tavoitteisiin kuului myös rakentaa yhteys palvelimen ja käyttäjän välille siten, että yhteys voitaisiin todeta toimivaksi molempien puolien osalta.

Tehdyt asiat

Projektissa hyödynnettiin aiemmassa Yritysprojekti 3:ssa toteutettua asiakas–palvelinmallia, minkä ansiosta työtä ei tarvinnut aloittaa täysin alusta. Ensimmäisessä palvelinversiossa keskityttiin ainoastaan yhteyden muodostamiseen käyttäjään, ja tämä saatiin toimimaan nopeasti. Tämän jälkeen palvelinta ja asiakasta alettiin muokata niin, että palvelin lähettäisi komentoja asiakkaalle liikkumista varten, ja asiakas toimisi ainoastaan näiden komentojen vastaanottajana ilman omaa ohjelmallista logiikkaa.

Ohjelmasta saatiin rakennettua versio, jossa yhtä robottia voitiin ohjata palvelimelta käsin. Samalla havaittiin kaksi merkittävää ongelmaa:

1. Mikäli useampaa robottia halutaan ohjata samanaikaisesti, täytyy palvelimeen lisätä säikeistys (threads). Ilman tätä palvelin ei kykene kommunikoimaan usean robotin kanssa yhtä aikaa.
2. Linjaseuraajaan liittyi luotettavuusongelmia. Robotin ohjaaminen palvelimelta lähetettävien komentojen avulla yhdistettynä sen omaan autonomiseen liikkumiseen aiheutti kuormitusta, mikä heikensi linjaseuraajan toimintakykyä.

Antureiden osalta päätettiin, että keskitytään saamaan värianturi ja linjaseuraaja toimimaan yhdessä, ennen kuin lisätään muita antureita. Robotissa on

yhteysportit vain kahdelle anturille, joten jotain oli keksittävä useamman anturin lisäämiseksi. Radan merkkivärit päätettiin, mutta niitä ei pystytty vielä testaamaan.

Tulokset

Sprintin aikana saatiin rakennettua toimiva asiakas–palvelin-yhteys, joka mahdollisti yksittäisen robotin ohjauksen palvelimelta. Samalla tunnistettiin keskeiset haasteet, jotka tulee ratkaista ennen järjestelmän laajentamista useamman robotin ohjaukseen. Säikeistämisen tarve palvelimessa sekä linjaseuraajan heikentynyt suorituskyky osoittautuivat jatkokehityksen kannalta kriittisiksi. Lisäksi edistyttiin radan visuaalisessa suunnittelussa ja robotin antureiden valinnoissa. Näiden tulosten pohjalta seuraavat vaiheet voitiin suunnitella tarkemmin.

3.6 Sprint 4–5 (24.2.2025–16.3.2025)

Tavoitteet

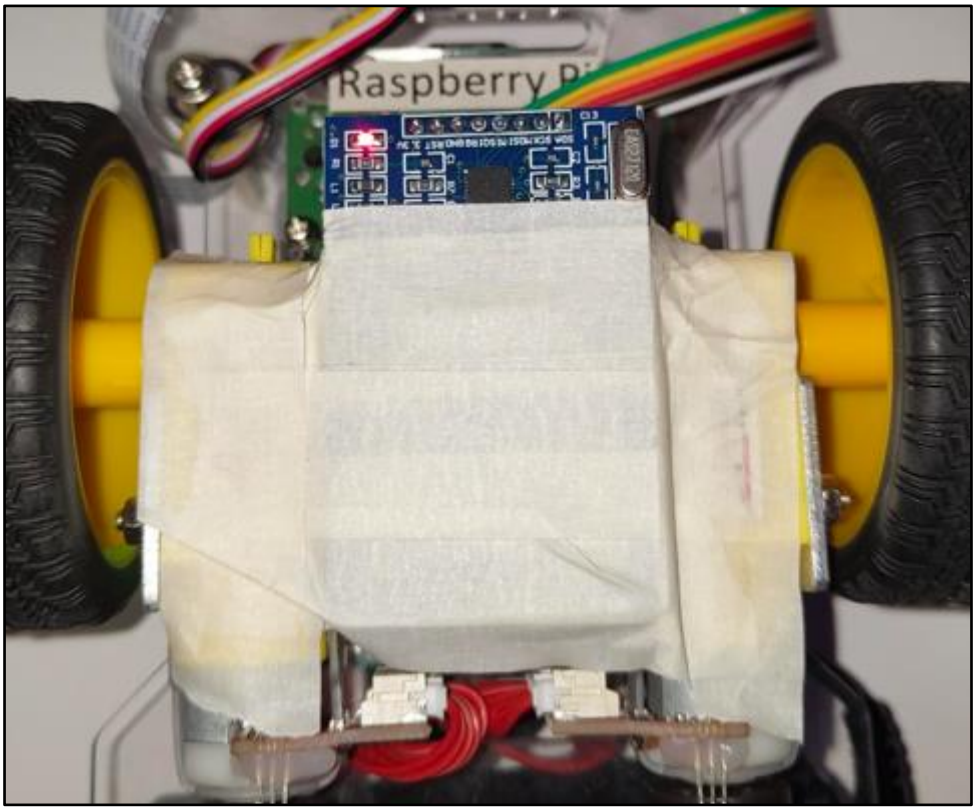
Näiden sprinttien tavoitteena oli kehittää palvelinta merkittävästi eteenpäin. Tavoitteisiin kuului säikeiden, prosessien, tkinter-kirjaston ja socket-yhteyksien toiminnan ymmärtäminen, esityskelpoisen demon toteuttaminen palvelimen ja robotin välisestä kommunikaatiosta sekä reittiä kuvaavan karttaikkunan luominen. Samalla pyrittiin parantamaan robottien liikkumisen tarkkuutta sekä ratkaisemaan edelleen vaivanneet värianturiin ja linjaseuraajaan liittyvät ongelmat.

Tehdyt asiat

Sprintin alussa keskityttiin uusien teknologioiden haltuunottoon. Erilaisiin esimerkkeihin ja dokumentaatioon perehtymällä saatiin nopeasti toimiva säikeistetty palvelin, jossa kaksi robottia pystyi liikkumaan yhtä aikaa. Reittien jakaminen robottikohtaisesti ei kuitenkaan vielä tässä vaiheessa onnistunut. Palvelimen käytettävyyttä tulisi parantaa rakentamalla ohjauspaneeli, jonka kautta robottien käynnistäminen ja reittien määrittely helpottuisi huomattavasti.

Palvelimen kehitys eteni hyvin, mutta robottien linjaseuraaja ja värianturi aiheuttivat edelleen merkittäviä ongelmia. Linjaseuraaja kuormittui ja kaatui toistuvasti,

eikä värianturi toiminut luotettavasti solmujen havaitsemisessa. Värianturista päätettiin luopua ja tilalle tuotiin huomattavasti luotettavampi ratkaisu. Värianturi korvattiin RFID-lukijalla ja väritarrat NFC-tarroilla. RFID-lukija asennettiin robotin pohjaan (kuva 17), ja NFC-tarrat sijoitettiin fyysisen radan alle (kuva 18). Tämä mahdollisti solmujen tunnistamisen ilman värien epävarmuustekijöitä.



KUVA 17. RFID-lukija robotin pohjaan kiinnitettynä.

KUVA 18. NFC-tarrat sijoitettuna radan linjojen alle.

RFID:n käyttöönoton myötä pystyttiin aloittamaan karttaikkunan toteuttaminen. Karttaikkunaan oli tarkoitus piirtää radalla liikkuvat robotit, kullakin oma värinsä. Kartan kehitystyössä kohdattiin kuitenkin haasteita säikeiden kanssa: karttaa ei saatu päivittymään reaaliaikaisesti ja sen sisältävän ikkunan liikuttaminen aiheutti palvelimen kaatumisen. Karttaa paranneltiin viikon aikana, mutta toimivaa versiota ei ehditty esitellä.

Tulokset

Sprintin lopputuloksena saatiin toimiva asiakas–palvelinyhteys, jota voitiin käyttää yksittäisen robotin ohjaukseen. Uusi RFID-tekniikka mahdollisti robotin sijainnin luotettavan seurannan, mutta linjaseuraajan ongelmat sen sijaan jatkuivat: robotti ei pysynyt reitillä, sillä linjaseuraajan suorituskyky ei ollut riittävä komentosyötön ja itsenäisen ohjauksen yhdistelmään. Vaikka palvelinta kehitettiin merkittävästi aiemmasta versiosta, kahden robotin yhtäaikaista toimintaa ei ehditty integroida uuteen versioon ajoissa, joten se jäi pois esittelystä. Karttaikkunan toteutus jäi osittain kesken, mutta tärkeät tekniset ratkaisut sen taustalla saatiin pohjustettua tulevaa varten.

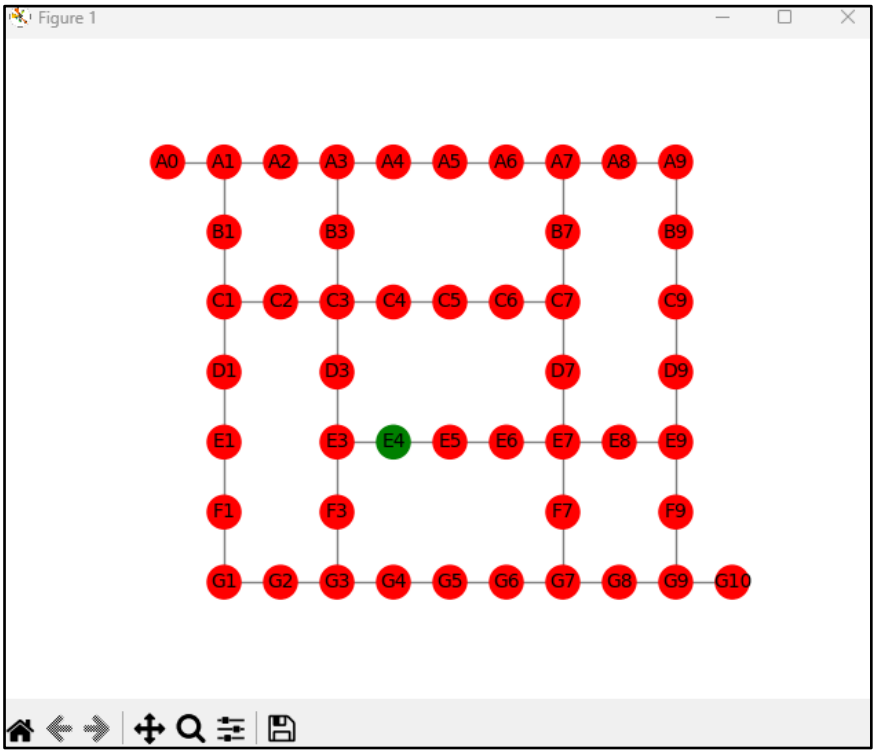
3.7 Sprint 6 (17.3.2025–23.3.2025)

Tavoitteet

Sprintin tavoitteena oli saada robotin sijainti päivittymään reaaliajassa sille suunnitellulle kartalle. Tarkoituksena oli hyödyntää NetworkX-kirjastoa kartan toteutuksessa, testata sen integraatiota projektin muihin komponentteihin sekä korjata aiemmin esiintyneet ongelmat kartan käytettävyydessä. Lisäksi haluttiin selvittää, miten useampi robotti saataisiin näkymään ja toimimaan kartalla samanaikaisesti.

Tehdyt asiat

Kartta toteutettiin NetworkX-kirjaston avulla siten, että sen solmut ja reunat vastasivat fyysisen radan rakennetta. Tämä mahdollisti kartan toiminnan tarkistamisen ohjelman muissa osissa ja helpotti virheiden havaitsemista solmu- ja reittimäärityksissä. Robotin sijaintitiedon perusteella ohjelmassa muutettiin sen kulloisenkin solmun väri vihreäksi, jolloin robotin sijainti näkyi reaaliaikaisesti kartalla. (Kuva 19.)



KUVA 19. Robotti E4 solmun päällä.

Kun sijaintipäivitykset saatiin toimimaan yhdelle robotille, alettiin kartoittaa, miten toinen robotti voitaisiin lisätä järjestelmään. Aluksi molemmat robotit näkyivät kartalla, mutta sijaintipäivityksissä ilmeni ongelmia: kun toinen robotti palasi aloituspisteeseen, päivitykset loppuivat molempien osalta. Tämä ongelma paikallistettiin ja korjattiin onnistuneesti.

Samassa sprintissä tutkittiin myös mahdollisuuksia kahden robotin samanaikaiseen käyttöön radalla. Tavoitteena oli varmistaa kartan toiminta reaaliaikaisessa monirobottiympäristössä ja valmistautua seuraavan sprintin kehitystehtäviin.

Tulokset

Sprintin aikana kartta saatiin toimimaan suunnitellulla tavalla. Karttaan liittyneet virheet, kuten sen kaatuminen ikkunaa liikuttaessa tai suljettaessa, korjattiin onnistuneesti. Robotin sijaintitiedot pystyttiin näyttämään reaaliaikaisesti kartalla, ja useamman robotin sijaintien näyttämisessä ilmenneet ongelmat ratkaistiin. Kartasta muodostui toimiva työkalu robottien liikkeiden seuraamiseen ja sijainnin hallintaan. Näin asetettiin vankka pohja seuraavalle sprintille, jossa keskityttiin robottien samanaikaiseen liikkumiseen ja toiminnan parantamiseen.

3.8 Sprint 7 (24.3.2025–30.3.2025)

Tavoitteet

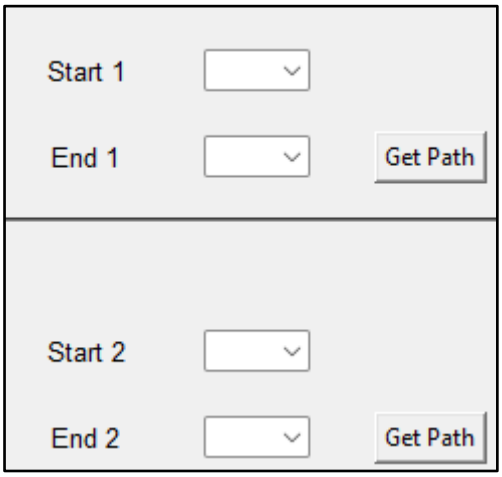
Tämän sprintin tavoitteena oli kehittää edelleen robottien hallintaa, erityisesti ohjauspaneelin näkökulmasta. Tarkoituksena oli toteuttaa käyttöliittymä, josta voitaisiin valita roboteille aloitus- ja lopetussolmut. Lisäksi tavoitteena oli ratkaista ongelmat, jotka liittyivät kahden robotin samanaikaiseen toimintaan radalla, erityisesti linjaseuraajaan liittyvät haasteet. Myös kartan sijaintipäivitysongelman ratkaiseminen tunnistettiin tarpeelliseksi, mutta sen korjaus päätettiin siirtää seuraavaan sprinttiin.

Tehdyt asiat

Sprintin alussa havaittiin kartan sijaintipäivitykseen liittyvä virhe, jossa toisen robotin sijaintipäivitys lakkasi toimimasta, kun toinen robotti palasi

aloituspisteeseen. Ongelman analysointia lykättiin, jotta voitaisiin keskittyä muihin kiireellisempiin tehtäviin.

Tämän sprintin pääpaino oli ohjauspaneelin kehittämisessä. Aloitettiin paneelin ominaisuuksien ja rakenteen suunnittelu tulevia käyttötarpeita silmällä pitäen. Toteutettiin toiminnallisuus, jossa käyttäjä voi valita kummalle tahansa robotille aloitus- ja lopetussolmut, joiden perusteella palvelin ohjaa robotin reitille. (Kuva 20).



KUVA 20. Robottien reittien määritys ohjauspaneelista

Lisäksi tutkittiin mahdollisuuksia saada kaksi robottia toimimaan radalla samanaikaisesti. Ongelmia aiheutti edelleen linjaseuraaja, jonka toiminta ei ollut luotettavaa kahden robotin liikkuessa yhtä aikaa. Ratkaisuun ei vielä päästy tämän sprintin aikana.

Tulokset

Sprintin aikana saatiin kehitettyä käyttökelpoinen versio ohjauspaneelista, jonka kautta roboteille voidaan määrittää yksilölliset reitit. Kartan päivitysvirhe tunnistettiin, mutta sen korjaus jäi seuraavaan sprinttiin. Kahden robotin yhtäaikainen käyttö ei vielä onnistunut toivotulla tavalla, sillä linjaseuraajan epävakaus aiheutti toimintahäiriöitä. Näin ollen sprintin lopputuloksena syntyi toimiva pohja ohjauspaneelille, mutta robottien käytettävyyden osalta tavoitteisiin ei vielä täysin päästy.

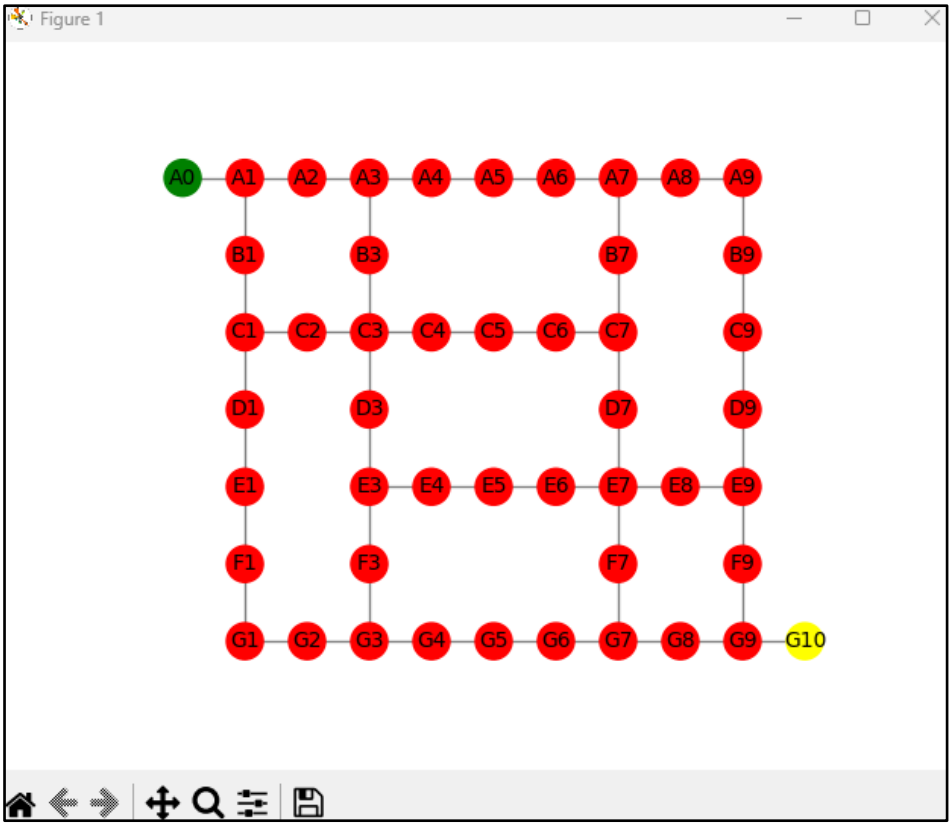
3.9 Sprint 8 (31.3.2025–6.4.2025)

Tavoitteet

Sprintin tavoitteena oli saattaa kartan toiminta vastaamaan haluttua toimintalogiikkaa, lisätä mahdollisuus reunan poistamiseen ja palauttamiseen kahden solmun väliltä, sekä ratkaista linjaseuraajan ja RFID-lukijan yhteistoiminnasta johtuvat ongelmat. Lisäksi pyrittiin varmistamaan, että kartan päivitys toimii oikein myös silloin, kun robotit palaavat aloituspisteeseen.

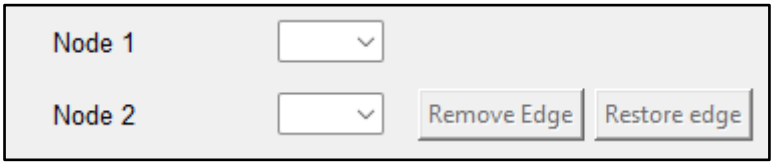
Tehdyt asiat

Sprintin alussa keskityttiin korjaamaan edellisessä sprintissä havaittu kartan päivitysongelma. Aiemmin kartan päivittyminen lakkasi, kun robotti palasi aloituspisteeseen, ja viimeinen sijaintimerkintä jäi todellista paikkaa edeltävään solmuun. Tämä ratkaistiin muokkaamalla robottien seurannan lopetusehtoja siten, että sijaintipäivitys toimii loppuun asti. (Kuva 21.)

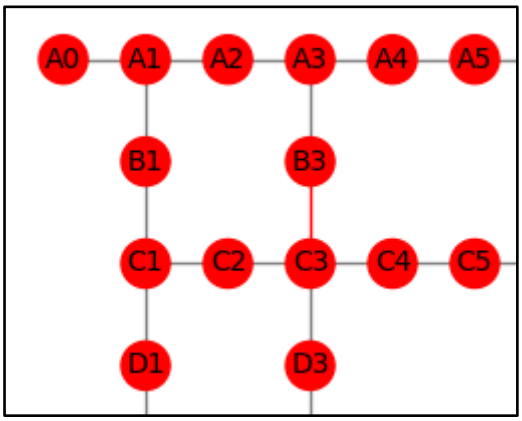


KUVA 21. Kaksi robottia korostettuna kartalla aloitussolmuissa A0 ja G10.

Ohjauspaneeliin lisättiin uusi ominaisuus, jonka avulla käyttäjä voi poistaa tai palauttaa reunoja kartalta antamalla kaksi niitä yhdistävää solmua (kuva 22 ja 23). Tämä lisäsi joustavuutta kartan hallintaan ja mahdollisti dynaamisen reittisuunnittelun. Tämän seurauksena robottien ohjaukseen kehitettiin logiikka, jonka avulla ne osaavat reitittää itsensä uudelleen, mikäli valittua reittiä ei voida käyttää poistettujen reunojen vuoksi.



KUVA 22. Reunan poistaminen radalta.



KUVA 23. B3 ja C3 solmujen välistä poistettu reuna korostettuna punaisella värillä kartalla.

Linjaseuraajan ja RFID-lukijan yhteistoimintaa yritettiin optimoida karsimalla koodista ylimääräisiä ajon keskeytyksiä ja muita mahdollisesti häiritseviä toimintoja. Koska tämä ei tuottanut toivottua parannusta, kokeiltiin vaihtoehtoisia ohjelmistokirjastoja sekä linjaseuraajalle että RFID-lukijalle. Uusien kirjastojen käyttöönotto ei kuitenkaan ratkaissut ongelmia, minkä vuoksi päätettiin palata alkuperäisiin ohjelmistokirjastoihin.

Tulokset

Sprintin aikana kartan toiminta saatiin vastaamaan odotuksia, ja kartan päivittymiseen liittyneet ongelmat korjattiin onnistuneesti. Ohjauspaneelin uusi reunan

poisto- ja palautustoiminto otettiin käyttöön, ja sen pohjalta toteutettiin myös robottien uudelleenreititys. Linjaseuraajan ja RFID-lukijan yhteistoiminnan ongelmat jäivät kuitenkin edelleen ratkaisematta, ja niiden selvittäminen päätettiin siirtää seuraavaan sprinttiin. Kokonaisuudessaan sprintin tavoitteet saavutettiin teknisen kehityksen osalta hyvin, vaikka kaikki toiminnalliset haasteet eivät vielä ratkenneet.

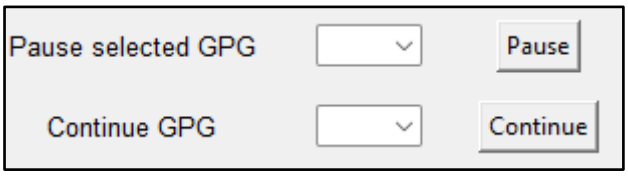
3.10 Sprint 9 (7.4.2025–13.4.2025)

Tavoitteet

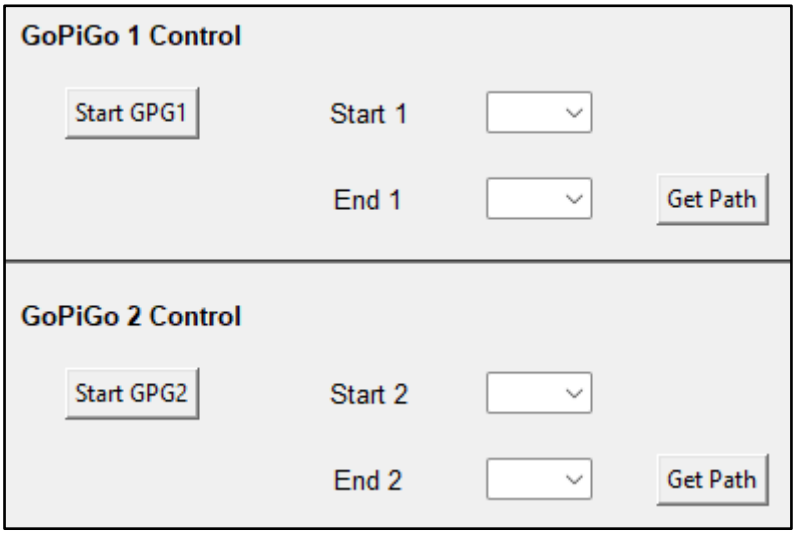
Sprintin tavoitteena oli lisätä ohjauspaneeliin robottien pysäyttämiseen liittyvä toiminto, jonka avulla sekä yksittäinen että kaikki robotit voitaisiin pysäyttää suoraan käyttöliittymästä. Ohjauspaneeliin haluttiin lisätä myös ominaisuus, jonka avulla robotteihin voidaan ottaa yhteys ja käynnistää ne suoraan paneelista. Tämän lisäksi pyrittiin ratkaisemaan pitkään vaivannut linjaseuraajan jumiutumisongelma. Product backlogista nostettiin toteutettavaksi myös viimeisiä haluttuja toiminnallisuuksia, kuten reaaliaikainen uudelleenreititys, sekä kameran lisääminen roboteille esteiden analysointia varten.

Tehdyt asiat

Ohjauspaneeliin lisättiin toiminto, jolla robottien toiminta voidaan pysäyttää joko yksittäin tai kaikkien osalta samanaikaisesti (kuva 24). Tämä mahdollisti sen, että robotteja voidaan ohjata ja hallita täysin paneelin kautta ilman ulkoisia komentoja. Paneeliin lisättiin myös ominaisuus, joka ottaa etäyhteyden robottiin ja käynnistää ohjelman automaattisesti (kuva 25). Tähän asti robotti oli käynnistetty etäyhteydellä palvelimen ulkopuolelta.



KUVA 24. Robottien pysäyttäminen ohjauspaneelista.



KUVA 25. Robottien reittien määrityksen yhteyteen lisätty käynnistäminen.

Linjaseuraajan koodiin tehtiin korjaus, joka poisti aiemmin esiintyneen jumiutumisongelman. Ongelma ratkesi lopulta, kun RFID-lukeminen ja linjan seuranta siirrettiin omiin säikeisiinsä, mikä jakoi kuormituksen tasaisemmin ja mahdollisti molempien toimintojen yhtäaikaisen suorituksen.

Tämän jälkeen aloitettiin product backlogissa määriteltyjen ominaisuuksien toteutus. Ensimmäisenä kehitettiin uudelleenreititys (kuva 26), jonka avulla robotti osaa reagoida tilanteeseen, jossa reitiltä on poistettu reuna. Robotin ohjelma tunnistaa muutoksen ja laskee uuden reitin ilman poistettua osuutta.

```
if self.rerouting_check[self.ID]["event"].is_set():
    print("rerouting_check is set")
    self.rerouting_check[self.ID]["event"].clear()
    break_from_logic_loop = False
    for i in range(len(self.path) - 1):
        edge = (self.path[i], self.path[i + 1])
        reversed_edge = (self.path[i + 1], self.path[i])
        if edge in PathFinding.removed_edges or reversed_edge in PathFinding.removed_edges:
            break_from_logic_loop = True
            print("Removed edge found in path:", edge)
            self.state = "REROUTED_FROM_CURRENT_TO_DESTINATION"
            break
    if break_from_logic_loop == True:
        break
    if break_from_logic_loop == True:
        break
```

KUVA 26. Osa uudelleenreititys logiikasta.

Lisäksi robottiin asennettiin kamera, jonka oli tarkoitus analysoida esteitä, joita etäisyysanturi havaitsee. Kameran integroinnin myötä tunnistettiin uusia haasteita, kuten se, että robotin toiminta jäätyy, kun kamera ja etäisyysanturi toimivat samanaikaisesti. Kuvan lähettämistä robotilta palvelimelle ei vielä ehditty ratkaista, ja sen selvittäminen päätettiin siirtää seuraavaan ylimääräiseen sprinttiin.

Tulokset

Sprintin aikana saatiin käyttöön yhteydenmuodostaminen ohjauspaneelista, joka mahdollistaa robottien hallinnan täysin paneelista käsin. Linjaseuraajan luotettavuus parani merkittävästi säikeistämisen myötä, ja robotit saatiin toimimaan tasaisemmin. Reaaliaikainen uudelleenreititys saatiin onnistuneesti käyttöön, ja robotti osaa nyt vaihtaa reittiä ilman manuaalista väliintuloa. Robottien pysäyttäminen kesken niiden ohjelman suorittamisen oli nyt myös mahdollista. Kameran integroinnin myötä ilmeni uusia haasteita, joita ei vielä ehditty ratkaista, mutta pohjatyö seuraavaa sprinttiä varten saatiin tehtyä.

3.11 Sprint 10 (14.4.2025–16.4.2025)

Tavoitteet

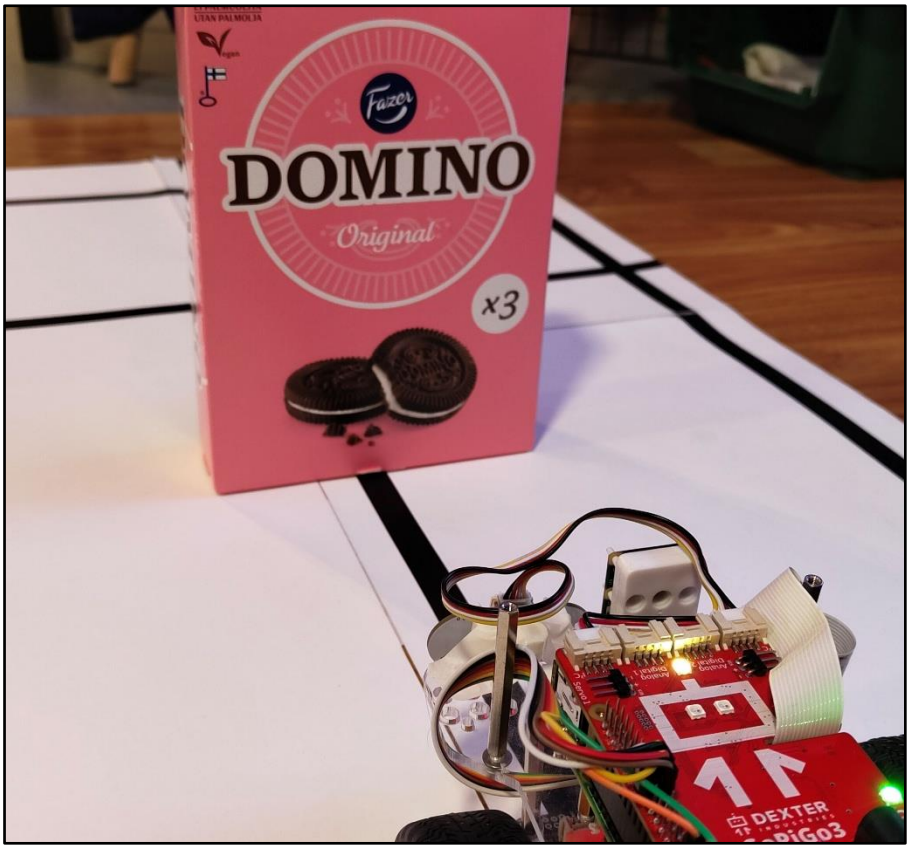
Viimeisen, lyhyemmän sprintin tavoitteena oli saattaa loppuun edellisessä sprintissä kesken jäänyt esteestä kuvan ottaminen ja sen välittäminen tekoälylle analysoitavaksi. Lisäksi pyrittiin ratkaisemaan ongelmat, joita syntyi eri antureiden ja kameran yhteistoiminnasta, sekä viimeistellä tekoälyn viestien esitystapa ohjauspaneelissa.

Tehdyt asiat

Sprintin alussa keskityttiin antureiden ja kameran aiheuttaman viiveen selvittämiseen. Näiden yhteiskäyttö aiheutti robotin hallintaan merkittävää hidastumista, minkä vuoksi robotti ei pystynyt enää seuraamaan viivaa luotettavasti. Ongelmaa lähdettiin ratkaisemaan siirtämällä etäisyysanturin ja kameran toiminta omiin säikeisiinsä. Tämä mahdollisti samanaikaisen toiminnan, mutta aiheutti uusia

ongelmia, kun useita päällekkäisiä säikeitä muodostui. Ongelma ratkaistiin estämällä usean samaa funktiota suorittavan säikeen käynnistyminen.

Kuvan tallentaminen onnistuttiin toteuttamaan niin, että robotti ottaa kuvan tietyllä etäisyydellä havaitsemastaan esteestä ja tallentaa sen (kuva 27). Tämän jälkeen täytyi enää kehittää menetelmä, jolla kuva välitetään palvelimelle ja sieltä tekoälylle analysoitavaksi. Toteutuksessa kuva päätettiin lähettää biteiksi muunnettuna, ja palvelimelle lisättiin logiikka, jolla se osaa vastaanottaa kuvan määriteltyjen ehtojen mukaisesti. Kun kuva on vastaanotettu, palvelin lähettää sen edelleen tekoälyrajapinnalle, joka palauttaa analyysin kuvasta.



KUVA 27. Robotti ottamassa kuvaa esteestä.

Lopuksi ohjauspaneeliin lisättiin tekstikenttä, johon tekoälyrajapinnan analyysit voidaan tulostaa (kuva 28). Näin varmistettiin, että käyttäjä näkee tekoälyn palauttaman tiedon suoraan ohjausnäkymästä.

KUVA 28. Tekoälyn analyysi esteestä.

Tulokset

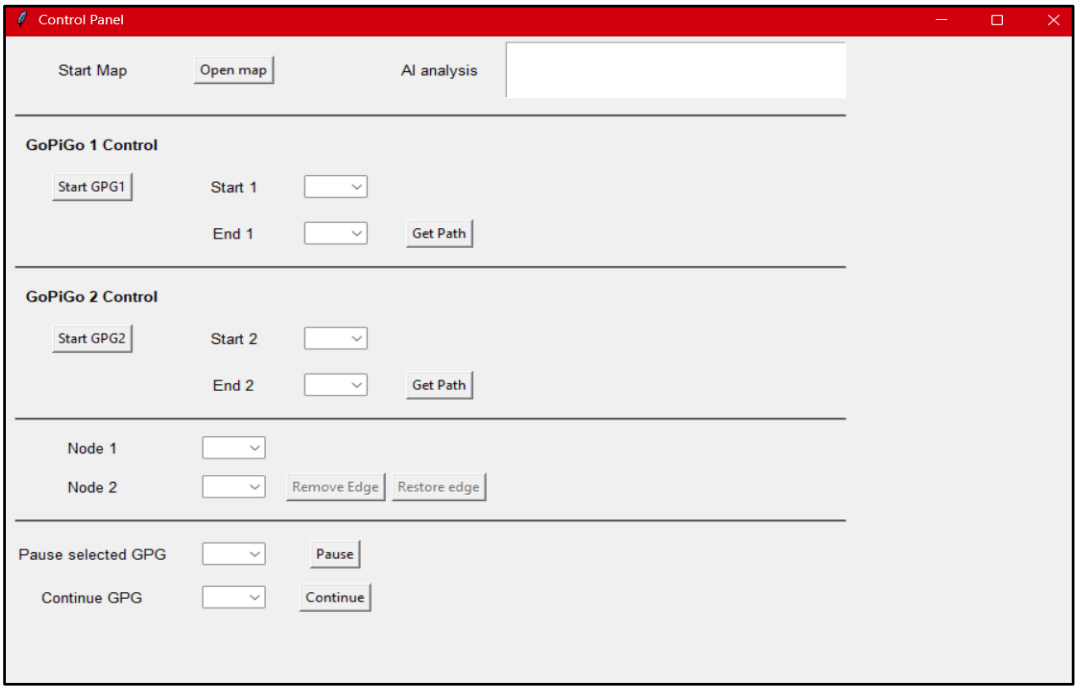
Sprintin aikana saatiin onnistuneesti toteutettua kameran ja tekoälyn välinen tiedonsiirto sekä sen integrointi palvelimen kautta toimivaksi kokonaisuudeksi. Robotti kykenee nyt ottamaan kuvan, lähettämään sen palvelimelle ja sieltä tekoälylle analysoitavaksi. Tekoälyn vastaus näkyy selkeästi ohjauspaneeliin lisätyssä tekstikentässä. Lisäksi ratkaistiin anturien ja kameran yhteistoimintaan liittyneet suorituskykyongelmat säikeistämällä ne erikseen ja estämällä turhat säikeiden monistukset. Näin saatiin päätökseen viimeinen sprintti, joka viimeisteli projektin lopulliset toiminnallisuudet.

4 TEKNINEN TOTEUTUS

4.1 Ohjauspaneeli

Robottien hallintaan käytetään Pythonilla toteutettua ohjauspaneelia (kuva 29), joka kokoaa yhteen kaikki keskeiset toiminnot robottien operointia varten. Ohjauspaneeli sijaitsee palvelimessa. Ohjauspaneelin päätoimintoihin kuuluvat robottien käynnistys, reittitietojen syöttäminen, karttamoduulin avaaminen, robottien pysäytys sekä ajastettu käynnistys. Lisäksi paneelista on mahdollista suorittaa tekoälyanalyysiä hyödyntäen OpenAI API:n kuvantunnistusominaisuuksia sekä hallita toiminta-alueen reunojen poistamista ja palauttamista.

Käyttöliittymässä on omat graafiset elementit molemmille roboteille. Käyttäjä voi syöttää aloitus- ja lopetussolmut, joiden perusteella ohjelma luo reitin. Paneelissa on painikkeet robottien käynnistämiseen ja pysäyttämiseen, karttanäkymän avaamiseen, reunan poistamiseen ja palauttamiseen sekä tekoälyanalyysin tulosten näyttämiseen.



Kuva 29. Käyttäjälle aukeava ohjauspaneeli näkymä.

Karttanäkymä voidaan avata ohjauspaneelista, ja se avautuu erilliseen ikkunaan. Kartta luodaan map.py-tiedostossa hyödyntäen NetworkX-kirjastoa. Kartan muodostamisessa käytetään hyväksi Get_coordinates_and_edges.py-tiedostossa olevia globaaleja muuttujia, jotka sisältävät kartan koordinaatit ja reunat. Näiden tietojen perusteella rakennetaan visuaalinen kuvaus siitä, miltä kartta näyttää.

Reunan poisto tapahtuu syöttämällä kaksi solmua, joiden välillä oleva reuna halutaan poistaa. Kyseinen reuna poistetaan reunalistasta, jolloin se myös päivittyy globaaliin reunalistaan Get_coordinates_and_edges.py-tiedostossa. Reunan palautus kumoaa tämän muutoksen, jolloin lista palautuu alkuperäiseen tilaansa. Karttanäkymässä poistettu reuna esitetään punaisena, kun normaalit reunat näkyvät harmaana.

Robottien käynnistystä varten toteutettiin socket-yhteydet asiakkaan ja palvelimen välille. Molemmille roboteille luotiin oma socket-yhteytensä, mikä mahdollistaa niiden hallinnan erikseen. Robottien koodi ajetaan etänä SSH-yhteyden kautta, jolloin voidaan suorittaa tarvittava skripti robotin tiedostopolussa sijaitsevien asiakaskoodien avulla. (Kuva 30.)

```
 17  class Control_Panel:
148      def start_gpg1_and_ssh(self):
149          threading.Thread(
150              target=self.ssh_start_script_on_gopigo,
151              args=(
152                  os.getenv("GPG1_IP"),
153                  os.getenv("GPG1_USER"),
154                  os.getenv("GPG1_PASS"),
155                  "/home/pi/GoPiGo/22222/main.py"
156              ),
157              daemon=True
158          ).start()
159
160          self.handle_button_press("GPG1")
```

KUVA 30. SSH-yhteyden muodostaminen palvelinohjelman sisällä

4.2 Asiakasohjelma

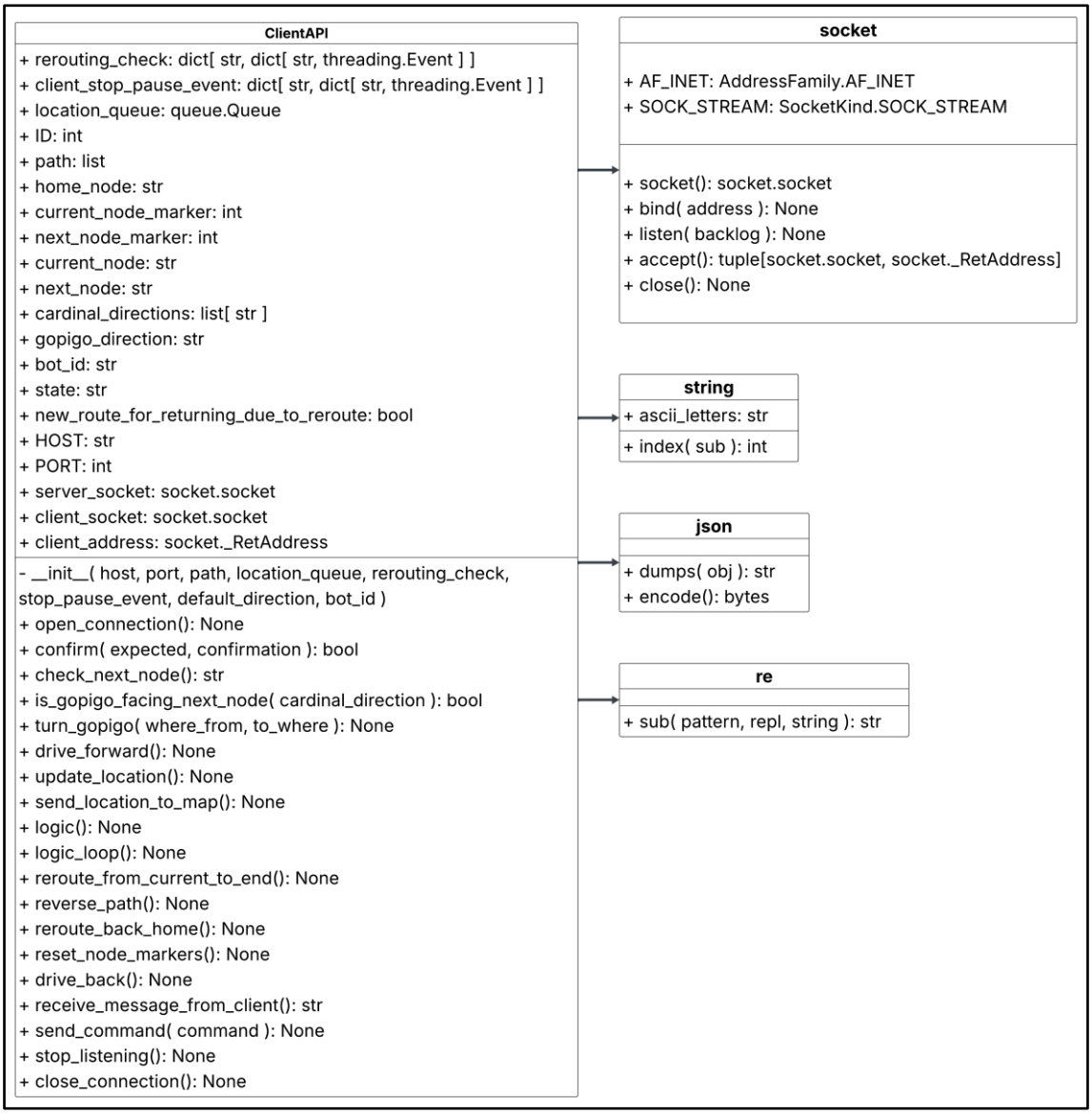
Asiakkaan (robotin) koodit sijaitsevat Raspberry Pi:ssä, ja niitä ajetaan palvelimen kautta. Koodin tärkeimpiin ominaisuuksiin kuuluu erilaisten anturien alustaminen, jotta robotti pystyy lukemaan dataa esimerkiksi esteistä, linjan seuraamisesta sekä RFID-tarroista, joiden avulla se saa sijaintitietonsa.

GoPiGo-alusta täytyy alustaa, jotta robotti pystyy liikkumaan halutulla tavalla palvelimelta tulevien komentojen perusteella. Koodissa määritellään myös, milloin ja miten robotti kääntyy eri tilanteissa, sekä miten se toimii, jos se ei enää löydä seurattavaa mustaa linjaa.

RFID-lukijan tehtävänä on lukea kartalle asetetut RFID-tarrat, jotka sijaitsevat solmupisteissä. Näiden avulla robotti pystyy matemaattisesti määrittämään, minkä solmun kohdalla se on. Linjaseuraaja lukee reitillä kulkevaa mustaa linjaa, joka toimii robotin kulkureittinä. Etäisyysanturi puolestaan pysäyttää robotin, jos este havaitaan, ja ottaa siitä kuvan tekoälyn analysoitavaksi.

4.3 ClientAPI-komponentin luokkakaavio

Luokkakaaviossa (kuva 31) on avattuna tiimin luomasta sovelluksen yksi merkittävä osa, ClientAPI-komponentti. Komponentin tarkoituksena on käsitellä robotin liikkumisen ohjausta. Jokainen erillinen kaavion komponentti on kuvattu neliönä, joka on jaettu kolmeen osaan, joista ensimmäisessä eli ylimmässä on komponentin nimi. Isoin komponentti on nimensä mukaisesti tiimin oma kehittämä ClientAPI-komponentti. Pienemmät laatikot ovat Pythonin sisäisiä ohjelmistokirjastoja, joihin on lueteltuna muuttujat ja metodit, joita käytetään ClientAPI:ssa. (Kuva 31.)



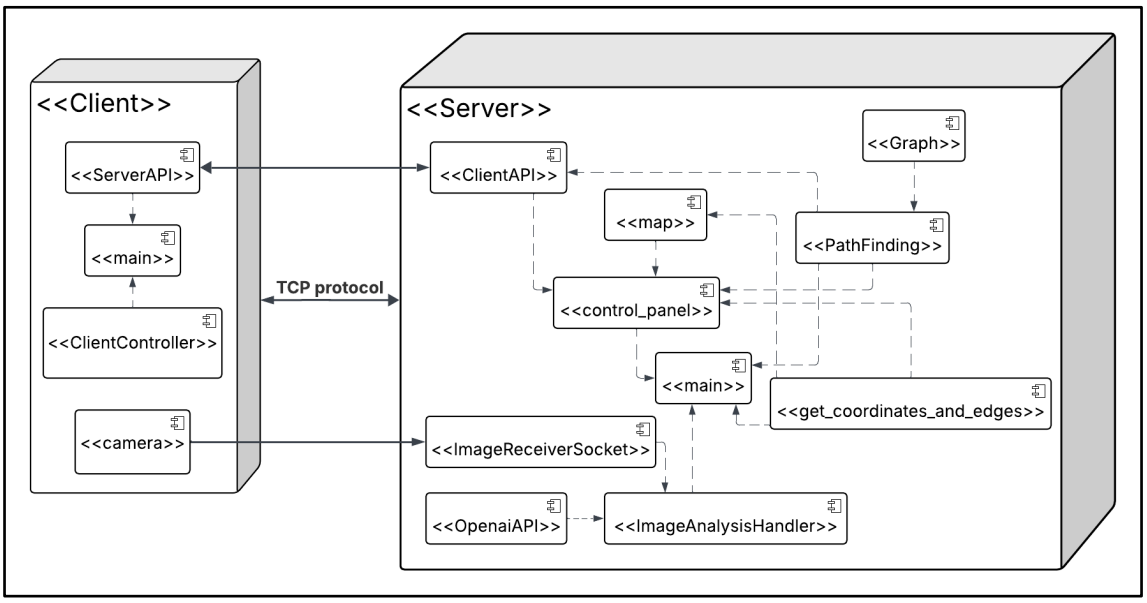
KUVA 31. Luokkakaavio ClientAPI.py tiedostosta

4.4 Arkkitehtuurikuvaus

Arkkitehtuurikuvauksessa (kuva 32) tiimin luoma projekti on jaettu kahteen tietokoneohjelmaan: asiakasohjelma eli client ja palvelinohjelma eli server. Näiden ohjelmien sisälle tiimi kehitti erinimisiä komponentteja, jotka ovat käytännössä ohjelman eri osia, joilla on jokaisella oma käyttötarkoitus.

Komponenttien väliset katkoviivat osoittavat nuolensuuntaisesti, mitkä ovat komponenttien väliset suhteet. Katkoviivoissa nuolensuunta näyttää, mikä komponentti käyttää mitäkin. Kiinteät viivat tarkoittavat tiedonsiirron ja suuntaa. Asiakas-

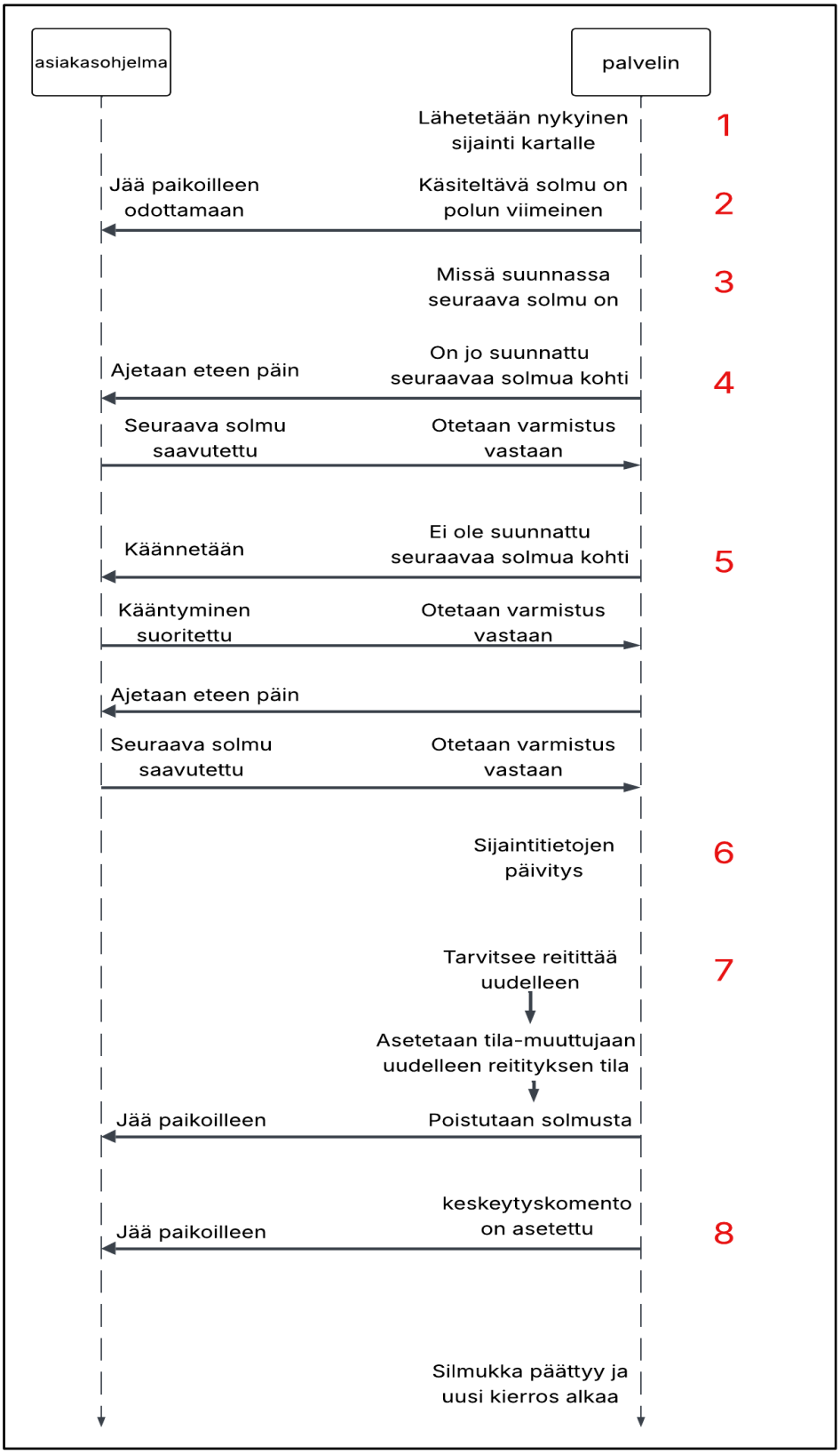
ja palvelinohjelma kommunikoivat käyttäen TCP-tiedonsiirtomenetelmää. (Kuva 32.)



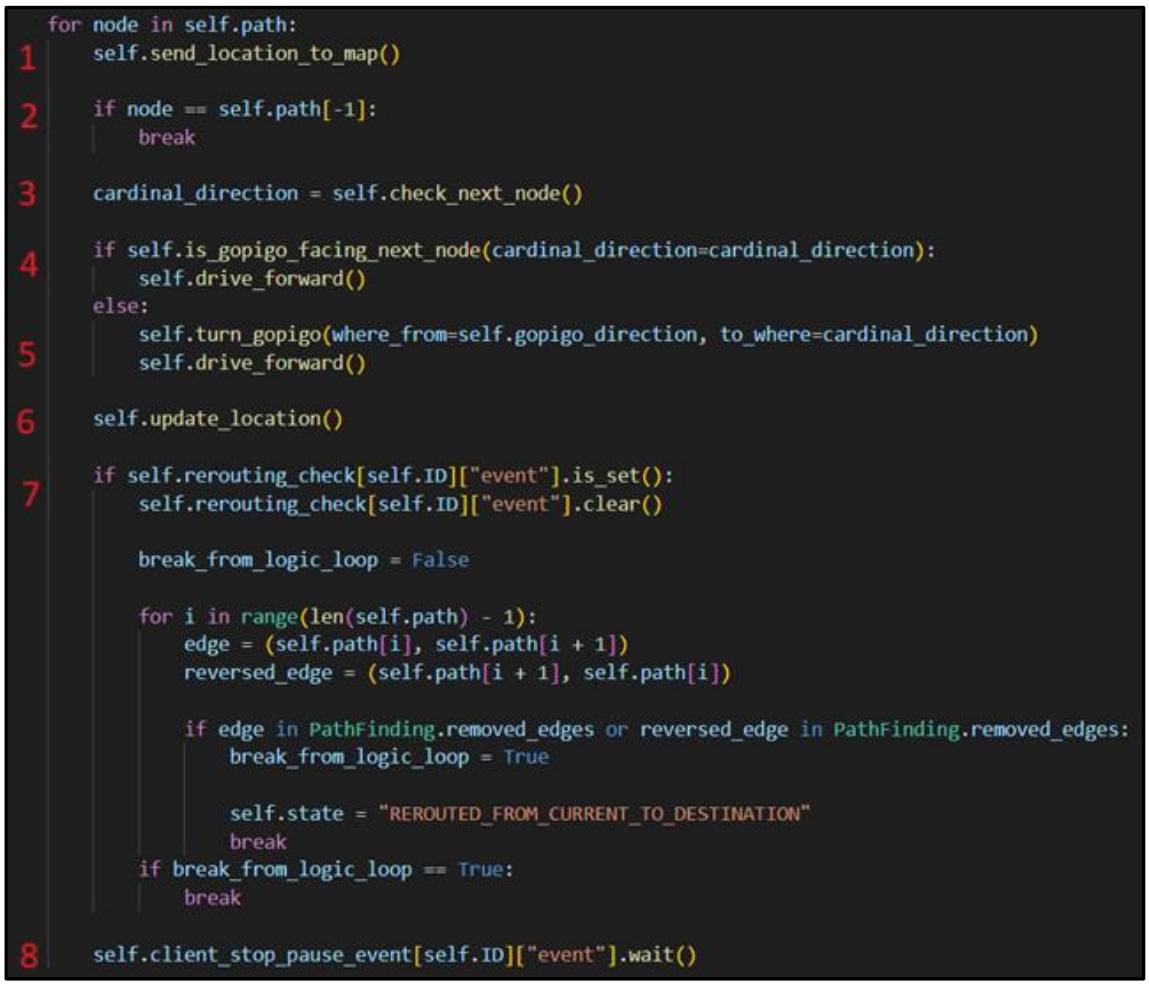
KUVA 32. Järjestelmän arkkitehtuurikaavio.

4.5 Ajologiikka

Reitin jokainen solmu käydään läpi Pythonin sisäänrakennetulla for loop -toiminnolla, jonka sisässä on johonkin tarkoitukseen luotuja komentoja. for loopissa käydään läpi yhtä monta kierrosta kuin on reitillä solmuja. Jokaisen kierroksen aikana käydään läpi samat ennalta määrätyt komennot, if-osioiden ehdot, ja ehtojen täyttyessä niiden alaiset komennot. Suoritettavilla komennoilla pyritään joko ohjaamaan robottia tai päivittämään muistiin ohjaamiseen liittyvää dataa.



KUVA 33. Ajologiikan kulkukaavio.



KUVA 34. Ajologiikka koodissa.

Ajologiikka on avattu vaiheittain sekä toimintaa selkeyttävässä kuvaajassa, (kuva 33) että koodipätkässä (kuva 34). Vaiheet ovat numeroitu siten, että numeroinnin mukaiset vaiheet molemmissa kuvissa vastaavat toisiaan. Vaiheet ovat seuraavat:

1. Kierros aloitetaan siirtämällä muistiin tallennettu robotin sijaintitieto kyseisestä ohjaamista käsittelevästä luokasta toiseen luokkaan, jossa käsitellään karttanäkymään. Tämä toiminto muodostaa perustan sijaintitiedon kuvantamiselle. Sijaintitiedon siirtäminen katsottiin parhaaksi toteutettavan tässä vaiheessa, koska tämän vaiheen kohdalla robotin fyysinen sijainti ja looginen sijainti täsmäävät.
2. Sijaintitiedon siirtämisen jälkeen tarkistetaan Pythonin if-lausekkeella, onko käsiteltävä solmu polun viimeinen solmu, Jos on, niin silmukasta poistutaan. Pyrkimyksenä on lopettaa silmukan

sisäisten koodirivien suoritus aikaisin, jolloin robotti ei ala tekemään ylimääräisiä liikkeitä päätepisteeseen saavuttuaan. Tällainen yksittäinen if-lauseke tarkistus logiikan alussa vähentää suoritettavan koodin kuormittavuutta.

3. Jos käsiteltävä solmu ei olekaan polun viimeinen, niin silmukasta ei poistutakaan vaan sisäisten koodirivien suorittamista jatketaan. Seuraavana vaiheena on funktion suorittaminen, jonka sisällä verrataan nykyisen solmun nimenä käytettävää kirjain-numero-yhdistelmää seuraavaan solmuun. Esimerkiksi A1, josta seuraavana voisi olla A2; tässä tapauksessa kirjain on sama, joten verrataan numeroita. Luku kasvaa, joten suunta on itä. Vastaavasti, jos luku vähenee, niin suunta on länsi. Kirjaimia vertaillaan aakkosten järjestyksellä. Kirjaimen ollessa aakkosissa seuraava, suunta on etelä, ja toisin päin pohjoinen. funktiokutsusta saadaan palautusarvona suunta.
4. Saatua suuntaa verrataan, vastaako se muistissa olevaa robotin suuntaa. Tässä käytetään yksinkertaista totuusehtoa Pythonin if-else-lausekkeella, jolla voidaan tarkistaa ehtojen varassa, kumpi kahdesta vaihtoehdosta suoritetaan. Eli jos robotti on suunnattu seuraavaa solmua kohti, niin ehto on tosi ja asiakasohjelmalle lähetetään komento, että se ajaisi eteen päin. Asiakas-ohjelma lähettää varmistuksen, jonka saapumista odotetaan ennen kuin jatketaan seuraavien koodirivien suorittamista. Täten varmistetaan eri osien yhtäaikaisen toiminnan sujuvuus.
5. Jos ehto on epätosi, niin suoritetaan erillinen kääntymiskomento. Kääntyminen tapahtuu siis tarpeen vaatiessa, kun laskelmoitu suunta ei vastaa robotin suuntaa. Taas odotetaan varmistusta toiminnan onnistuneesta suorittamisesta. Kääntymisen jälkeen suoritetaan samanlainen eteen päin ajo –komento kuin aiemmassa vaiheessa.
6. Seuraavaksi päivitetään muistissa pidettäviä sijaintitietoja, joista käy ilmi nykyinen ja sitä seuraava solmu. Sijaintitiedon päivitys hoidetaan nyt, koska robotti on edellisessä vaiheessa päässyt reitillä

uuden solmun kohdalle. Näin varmistetaan sijainnin seurannan pitäminen johdonmukaisena.

7. Robotin ollessa uudessa kohteessa paikoillaan tarkistetaan, onko komentopaneelissa poistettu reuna reitiltä. Tarkistus hoidetaan threading-kirjaston Event-ominaisuudella. Komentopaneelissa napin painalluksella asetetaan Event päälle. Jos Event on päällä siirrytään suorittamaan if-lausekkeen sisäistä koodia. Aluksi laitetaan Event pois päältä, jotta sitä voidaan käyttää tulevaisuudessa uudestaan, sitten tarkistetaan, vastaako reitin yksikään solmu luokkamuuttujaan talletettua solmua. Jos vastaa, niin se tarkoittaa, että poistettu solmu on robotin reitillä, jolloin asetetaan state-muuttujaan uudelleen reitityksen tila, ja silmukasta poistutaan.
8. Silmukan lopussa viimeisenä toimintona tarkistetaan jälleen Eventiä käyttäen, tarvitseeko ohjelmaa keskeyttää väliaikaisesti. Tällä kertaa toimintaperiaatteena on, että jos Event on laitettu päälle, niin tämä rivi ohitetaan, jolloin silmukan kierros päättyy ja uusi kierros alkaa. Komentopaneelissa nappia painamalla Event voidaan laittaa pois päältä, jolloin pysähdytään tälle riville odottamaan, että Event on laitettu uudestaan päälle.

4.6 Dijkstran algoritmi

Dijkstran algoritmi on graafialgoritmi, jonka avulla etsitään lyhin polku lähdesolmusta haluttuun solmuun, tai kaikkiin muihin solmuihin painotetussa graafissa, jossa kaikkien reunojen painot ovat ei-negatiiviset. (Geeksforgeeks 18.4.2025.)

Projektissa Dijkstran algoritmi toimii seuraavalla tavalla (kuva 35):

1. Alustukset:
   - Kaikki etäisyydet alustetaan äärettömiksi (INFINITY), paitsi aloitussolmulle asetetaan arvo 0.
   - Luodaan muuttuja ”distance_from_start”, johon tallennetaan lyhin tunnettu etäisyys lähtösolmusta jokaiseen solmuun.

- Luodaan muuttuja ”previous_node”, joka pitää kirjaa polun edellisistä solmuista.

2. Iterointi:
   - Ajetaan silmukkaa, jossa valitaan käsittelemättömistä solmuista se, jonka etäisyys on pienin aloitussolmuun.
   - Päivitetään sen naapureiden etäisyydet, jos uusi reitti on aiemmin tunnettuja lyhyempi.
   - Prosessia jatketaan, kunnes kohdesolmu saavutetaan.
3. Lopetus:
   - Kun lyhin reitti on löydetty, algoritmi rakentaa varsinaisen reitin seuraamalla muuttujan ”previous_node” tietoja kohdesolmusta takaisin aloitussolmun.
   - Funktio palauttaa listan, johon reitti on tallennettuna.

```
def shortest_path(self, start_node, end_node):
    unvisited_nodes = self.nodes.copy()
    distance_from_start = {node: (0 if node == start_node else self.INFINITY) for node in self.nodes}
    previous_node = {node: None for node in self.nodes}
    while unvisited_nodes:
        current_node = min(unvisited_nodes, key=lambda node: distance_from_start[node])
        unvisited_nodes.remove(current_node)

        if distance_from_start[current_node] == self.INFINITY:
            break
        for neighbor, distance in self.adjacency_list[current_node]:
            new_path = distance_from_start[current_node] + distance
            if new_path < distance_from_start[neighbor]:
                distance_from_start[neighbor] = new_path
                previous_node[neighbor] = current_node
        if current_node == end_node:
            break
    path = deque()
    current_node = end_node
    while previous_node[current_node] is not None:
        path.appendleft(current_node)
        current_node = previous_node[current_node]
    path.appendleft(start_node)
    return list(path), distance_from_start[end_node]
```

KUVA 35. Projektissa käytetty Dijkstran algoritmi.

4.7 Säikeet

Säikeet ovat ohjelman itsenäisiä suorituspolkuja ja niiden toiminta perustuu siihen, että ne jakavat saman prosessin muistin ja resurssit. Säikeistämisen avulla voidaan suorittaa useita tehtäviä rinnakkain samassa ohjelmassa. Sen avulla

voidaan säilyttää ohjelmien responsiivisuus, sekä suorittaa toistuvia tehtäviä taustalla ilman, että päälogiikka pysähtyy. (Python 18.4.2025.)

Palvelimen puolella säikeistystä hyödynnettiin mm. useamman robotin ohjaamisessa, useamman SSH-yhteyden muodostamisessa, kartan päivittämisessä sekä ohjauspaneelissa. Alla olevassa kuvassa (kuva 36) on esimerkki funktiosta, jonka avulla käynnistetään asiakkaan (robotin) socket-yhteys ja suoritetaan se omassa säikeessä. Jos funktiota ei suoritettaisi omassa säikeessä, useamman socket-yhteyden muodostaminen lopettaisi aina edellisen suorittamisen. Tällöin aiempi robotti lopettaisi ohjelmansa suorittamisen.

```
def open_and_run_socket(self, port, the_id):
    location_queue = self.location_queue_1 if the_id == "gopigo_1" else self.location_queue_2

    client_api = ClientAPI(host=os.getenv("IP_ADDRESS"), port=port, path=self.path, location_queue=location_queue,
                           default_direction="east", bot_id=the_id, rerouting_check=self.rerouting_check,
                           stop_pause_event=self.socket_logic_execution_pause)
    run_server = lambda client_api: asyncio.run(client_api.open_connection())
    (threading.Thread(target=run_server, args=(client_api,), daemon=True)).start()
```

KUVA 36. Yhteyden luominen asiakkaaseen.

Asiakkaan (robotin) puolella säikeistystä hyödynnettiin solmujen havaitsemisessa, sekä kuvan ottamisessa. Alla esimerkki solmujen havaitsemisen funktiosta, jossa säikeistämisen avulla voidaan suorittaa RFID-tunnisteiden seurantaa ilman, että se häiritsee robotin linjaseuraajaan toimintaa. (Kuva 37.)

```
def start_detecting_nodes(self):

    def node_detecting_logic():
        time.sleep(1.5)
        while True:
            if self.detect_rfid_node():
                self.node_detected_event.set()
                break
            else:
                time.sleep(0.25)

    threading.Thread(target=node_detecting_logic, daemon=True).start()
```

KUVA 37. Solmujen havaitsemisen aloittamisen funktio.

4.8 Tilanhallinta

Tilanhallinta (State management) on ohjelmoinnissa käytetty menetelmä, jossa sovelluksen tai olion toimintaa ohjataan hetkellisen tilan perusteella. Jokaiselle tilalle on omat ehdot ja toiminnallisuudet. Projektissa tilanhallintaa käytettiin robotin liikkumisen ohjaamiseen palvelimen puolella ”self.state”-muuttuja avulla. Robotit toimivat oman tilansa mukaan ja siirtyvät aina seuraavaan loogiseen tilaan polun edetessä tai poikkeustilanteen sattuessa. (Kuva 38.)

```
def logic(self):
    self.state = "STARTED"
    while self.listening:

        if self.state == "REROUTED_FROM_CURRENT_TO_DESTINATION":
            self.reroute_from_current_to_end()
            self.new_route_for_returning_due_to_reroute = True
            self.state = "STARTED"

        if self.state == "STARTED":
            self.logic_loop()

        if self.state == "DRIVE_BACK":
            self.drive_back()

        if self.state == "RETURNED_HOME":
            self.send_command(command="TURN_TWICE_RIGHT")
            confirmation = self.receive_message_from_client()
            self.confirm(expected="TURN_OK", confirmation=confirmation)
            self.close_connection()
```

KUVA 38. Tilanhallintafunktio palvelimessa.

Tilanhallintafunktio koostuu neljästä tilasta:

1. **STARTED**
   - Robotti aloittaa liikkumisen käyden läpi sille annetun reitin solmu kerrallaan.
2. **(REROUTED_FROM_CURRENT_TO_DESTINATION)**
   - Robotin tila vaihdetaan tähän, jos sen lasketulta reitiltä poistetaan reuna käytöstä, jolloin se ei voi ajaa loppuun alussa määritettyä reittiä.
   - Robotti laskee uuden reitin viimeisimmän ylitetyn solmun kohdalta, ja jatkaa uuden lasketun reitin perusteella alussa määritettyyn kohdesolmuun.

3. **DRIVE_BACK**
    - Robotti ajaa reitin käänteisessä järjestyksessä takaisin aloitussolmuun, ottaen huomioon mahdolliset poistetut reunat.
4. **RETURNED_HOME**
    - Kun robotti on palannut takaisin aloitussolmuun, tehdään käännös takaisin alkuperäiseen lähtöasentoon ja sammutetaan yhteys.

5 YHTEENVETO

Tämän opinnäytetyön päätarkoituksena oli kehittää itsenäisesti toimiva robotti, joka kykenee liikkumaan annetussa ympäristössä sille suunniteltujen algoritmien avulla. Tavoitteet saavutettiin pääosin, ja lopputuloksena syntyi osittain autonominen robotti, joka liikkui luotettavasti ja noudatti sille annettuja komentoja. Kokonaisuuteen kuului toimiva ohjausjärjestelmä sekä etäisyysanturi ja kamera, jotka mahdollistavat esteiden havainnoinnin. Uudelleenreitityksen toteutus esteen tai toisen robotin kohdatessa jäi kuitenkin ajanpuutteen vuoksi lisäämättä. Kyseinen toiminnallisuus olisi ollut teknisesti toteutettavissa, mutta projektin venyessä ominaisuuden lisääminen päätettiin jättää tulevaisuuden jatkokehitykseen.

Työskentelyn aikana kehitettiin vahvasti osaamista Python-ohjelmointikielessä sekä sen kirjastoissa ja rajapinnoissa. OpenAI API:n hyödyntäminen tarjosi mahdollisuuden perehtyä myös ohjelmointirajapintojen käsittelyyn. Hankitun osaamisen uskotaan tukevan tulevaa työllistymistä ohjelmistoalalla. Lisäksi projekti antoi kokemusta Scrum-menetelmästä ja sen käytännön soveltamisesta. Roolien selkeys ja säännöllinen tiimityöskentely edesauttoivat projektin hallintaa. Kuitenkin aikarajojen vuoksi osa ideoista jouduttiin hylkäämään, mikä osoittautui haasteelliseksi päätöksenteoksi.

Projektinhallinnan näkökulmasta opittiin erityisesti suunnittelun ja aikatauluttamisen tärkeys. Jo pienet viivästykset hidastivat koko kehitystä, koska ne vaikuttivat myös muiden tiimin jäsenten työskentelyaikatauluihin. Kommunikaation merkitys korostui tilanteissa, joissa oli tarve jakaa tietoa ongelmista ja päätöksistä. Mikäli viestintä ei ollut riittävän selkeää tai tehtävänjako epäselvää, syntyi helposti turhaa ajankäyttöä. Avoimella ja aktiivisella viestinnällä näitä ongelmia onnistuttiin kuitenkin korjaamaan. Merkittävin projektia hidastanut tekijä oli lopulta tekninen kokemattomuus, joka näkyi erityisesti ongelmatilanteiden ratkaisuissa.

Yksi konkreettinen esimerkki oli valotutkaan liittyvien teknisten vaatimusten määrittely, joka vei kohtuuttomasti aikaa suhteessa sen hyötyyn. Jälkikäteen arvioituna, vaatimusten tarkempi arviointi projektin alkuvaiheessa olisi mahdollistanut

yksinkertaisemman ratkaisun valinnan. Tämän perusteella opittiin, että teknisesti optimaalinen ratkaisu ei aina ole paras vaihtoehto, etenkin kun aika ja resurssit ovat rajallisia.

Projektin aikana kohdattiin myös muita teknisiä haasteita, joiden ratkaiseminen edellytti perehtymistä autonomisten ajoneuvojen yleisiin käytäntöihin. Valotutkan ja värianturin sijaan päädyttiin käyttämään RFID-lukijaa, joka osoittautui toimivaksi ja yksinkertaisemmaksi ratkaisuksi. Jatkossa vastaavissa tilanteissa vaihtoehtoisia ratkaisuja kannattaa arvioida etukäteen, jotta päätöksenteko olisi sujuvampaa ongelmien ilmaantuessa.

Ammatillisesti opinnäytetyö tarjosi monipuolisia oppimiskokemuksia. Ohjelmistokehittäjänä kehittyminen näkyi sekä teknisen osaamisen että projektityöskentelytaitojen kehittymisenä. Scrum-menetelmän käyttäminen antoi realistisen kuvan siitä, kuinka ohjelmistokehitys toteutuu oikeassa työympäristössä. Projekti lisäsi motivaatiota kehittyä alalla ja loi vahvan pohjan tulevalle osaamiselle.

Jatkokehityksen kannalta projektia voitaisiin täydentää muun muassa uudelleenreititystoiminnolla esteen tai toisen robotin kohdatessa. Robottiin voitaisiin lisätä myös muita antureita, jotka mahdollistaisivat uusia toimintoja. Lisäksi palvelin- ja asiakaspuolen virheenkäsittelyssä on tilanteita, joihin olisi hyödyllistä kehittää tarkempi toimintalogiikka – esimerkiksi tilanteessa, jossa palvelin kaatuu, robotti voisi palata itsenäisesti aloituspaikkaan.

Projektin käyttöarvo on selvä erityisesti oppimisnäkökulmasta. Työn tuloksia voidaan hyödyntää jatkossa sekä robotiikkaan tutustuvien opiskelijoiden että harrastajien projekteissa. Kokonaisuutena työ toimii selkeänä esimerkkinä siitä, miten autonominen robotti voidaan rakentaa itse rajallisilla resursseilla ja moderneja teknologioita hyödyntäen.

LÄHTEET

Ballejos, L. 25.10.2024. What Is a Remote Access Tool (RAT)? ninjaOne. Luettavissa: https://www.ninjaone.com/blog/what-is-a-remote-access-tool-rat/. Luettu: 30.1.2025.

BigPicture 23.5.2022. Iteration vs Sprint vs cadence in Agile: meaning and difference. Luettavissa: https://bigpicture.one/blog/sprint-cadence-iteration/. Luettu: 10.3.2025.

Brown, K. 21.9.2016. What Is Github, and What Is It Used For? How To Geek. Luettavissa: https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/. Luettu: 16.1.2025.

Cirkit Designer 2025. How to Use RFID-RC522: Examples, Pinouts, and Specs. Luettavissa: https://docs.cirkitdesigner.com/component/1a67bee7-de27-46be-8922-740cef7ebaec/rfid-rc522. Luettu: 23.4.2025.

Cloudflare 2024. What is the Secure Shell (SSH) protocol? Luettavissa: https://www.cloudflare.com/learning/access-management/what-is-ssh/. Luettu: 30.1.2025.

Codecademy 2022. Threading. Luettavissa https://www.codecademy.com/resources/docs/python/threading/. Luettu: 27.3.2025.

Dexter Industries 2024. Line Follower for Robots. Luettavissa: https://www.dexterindustries.com/store/line-follower-for-robots/. Luettu: 10.1.2025.

Dexter Industries 2025a. GoPiGo3 Starter Kit. Luettavissa: https://www.dexterindustries.com/store/gopigo3-starter-kit/. Luettu: 23.4.2025.

Dexter Industries 2025b. Line Follower for Robots. Luettavissa: https://www.dexterindustries.com/store/line-follower-for-robots/. Luettu: 23.4.2025.

Dexter Industries 2025c. Distance Sensor. Luettavissa: https://www.dexterindustries.com/store/distance-sensor/. Luettu: 23.4.2025.

Dexter Industries 2025d. Light & Color Sensor. Luettavissa: https://www.dexter-industries.com/store/light-color-sensor/. Luettu: 23.4.2025.

Dexter Industries 2025e. Raspberry Pi Camera. Luettavissa: https://www.dexterindustries.com/site/?product=raspberry-pi-camera. Luettu: 23.4.2025.

Euroopan komission täytäntöönpanoasetus 2022/1426. Itseohjautuvat autot pian todellisuutta EU:ssa. Luettavissa: https://eur-lex.europa.eu/legal-content/FI/TXT/PDF/?uri=CELEX:32022R1426&qid=1737102099088. Luettu: 14.1.2025.

Euroopan parlamentti 14.1.2019. Luettavissa: https://www.europarl.europa.eu/topics/fi/article/20190110STO23102/itseohjautuvat-autot-pian-todellisuutta-eu-ssa. Luettu: 11.2.2025.

Evans, K., de Moura, N., Chauvier, S., Chatila, R. & Dogan, E. 2020. Ethical Decision Making in Autonomous Vehicles: The AV Ethics Project. Sci Eng Ethics 26, 3285–3312 (2020). Luettavissa: https://doi.org/10.1007/s11948-020-00272-8. Luettu: 13.1.2025.

Geeksforgeeks 9.4.2025. What is Dijkstra's Algorithm? Luettavissa: https://www.geeksforgeeks.org/introduction-to-dijkstras-shortest-path-algorithm/. Luettu: 18.4.2025.

Generation Robots 2024. YDLIDAR X4PRO: Precise 360-Degree Distance Measurements. Luettavissa: https://www.generationrobots.com/en/404203-ydlidar-x4pro-360-laser.html. Luettu: 18.2.2025.

Git 2024. 1.3 Getting Started – What is Git? Luettavissa: https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F. Luettu: 16.1.2025.

GoPiGo 2024a. GoPiGo Go further. Luettavissa: https://gopigo.io/gopigo/ Luettu: 9.1.2025.

GoPiGo 2024b. Distance Sensor for GoPiGo. Luettavissa: https://gopigo.io/distance-sensor/. Luettu: 10.1.2025.

GoPiGo 2024c. Light & Color Sensor for GoPiGo. Luettavissa: https://gopigo.io/light-and-color-sensor/. Luettu: 11.1.2025.

GoPiGo 2024d. Raspberry Pi Camera for GoPiGo. Luettavissa: https://gopigo.io/raspberry-pi-camera/. Luettu: 11.1.2025.

Haroon, S. O. 2014. Client-Server Model. IOSR Journal of Computer Engineering. 16. vuosikerta. School of Computing Universiti Utara Malaysia Kedah, Malesia. Luettavissa: https://www.researchgate.net/publication/271295146_Client-Server_Model. Luettu: 28.1.2025.

influxdata 2023. What is the Time Library in Python? A Helpful Guide. Luettavissa: https://www.influxdata.com/blog/what-is-time-library-in-python-helpful-guide/. Luettu: 27.3.2025.

IONOS editorial team 12.5.2023. WinSCP. IONOS. Luettavissa: https://www.ionos.com/digitalguide/hosting/technical-matters/encrypted-data-transfer-with-winscp/. Luettu: 1.2.2025.

Last Minute Engineers 2024. What is RFID? How It Works? Luettavissa: https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/. Luettu: 19.2.2025.

Martinez-Diaz, M. & Soriguera, F. 2018. Autonomous vehicles: theoretical and practical challenges. ScienceDirect. Transportation Research Procedia. 33. vuosikerta. 275-282. Luettavissa: https://doi.org/10.1016/j.trpro.2018.10.103. Luettu: 21.1.2025.

Martinho, A., Herber, N., Kroesen, M. & Chorus, C. 2021. Ethical issues in focus by the autonomous vehicles industry. Transport Reviews. 41. vuosikerta. Numero 5. 556-577. Luettavissa: https://doi.org/10.1080/01441647.2020.1862355. Luettu: 27.1.2025

Matplotlib 2025. Matplotlib: Visualization with Python. Luettavissa: https://matplotlib.org/. Luettu: 18.4.2025.

Maurer, M., Gerdes, J., Lenz, B. & Winner, H. 2015. Autonomes Fahren: Technische, rechtliche und gesellschaftliche Aspekte. First Edition. Springer Vieweg Berlin. Heidelberg. E-kirja. Luettavissa: https://link.springer.com/chapter/10.1007/978-3-662-45854-9_4. Luettu: 15.1.2025.

Medium 2024. Exploring the Power of Python's typing Library. Luettavissa: https://medium.com/@moraneus/exploring-the-power-of-pythons-typing-library-ff32cec44981. Luettu: 18.4.2025.

Nath S 3.10.2025. Request Response Model, Usages, Anatomy and Drawbacks. Medium 2023. Luettavissa: https://medium.com/@sujoy.swe/request-response-model-usages-anatomy-and-drawbacks-42464e475cf5. Luettu 10.4.2025.

NetworkX 2025. Software for complex networks. Luettavissa: https://networkx.org/. Luettu: 18.4.2025.

Openai-python 16.4.2025. OpenAI Python API library. Github. Luettavissa: https://github.com/openai/openai-python. Luettu: 18.4.2025.

Paramiko 2025. Welcome to Paramiko! Luettavissa: https://www.paramiko.org/. Luettu: 18.4.2025.

Python 18.4.2025. base64 – Base16, Base32, Base64, Base85 Data Encodings. Luettavissa: https://docs.python.org/3/library/base64.html. Luettu 18.4.2025.

Python 18.4.2025. copy – Shallow and deep copy operations. Luettavissa: https://docs.python.org/3/library/copy.html. Luettu: 18.4.2025.

Python 18.4.2025. multiprocessing – Process-based parallelism. Luettavissa: https://docs.python.org/3/library/multiprocessing.html. Luettu: 18.4.2025.

Python 18.4.2025. struct – Interpret bytes as packed binary data. Luettavissa: https://docs.python.org/3/library/struct.html. Luettu: 18.4.2025.

Python 18.4.2025. traceback – Print or retrieve a stack traceback. Luettavissa: https://docs.python.org/3/library/traceback.html. Luettu: 18.4.2025.

Python Software Foundation 2025. asyncio — Asynchronous I/O. Luettavissa: https://docs.python.org/3/library/asyncio.html. Luettu: 27.3.2025.

Python Software Foundation 2025. os — Miscellaneo. us operating system interfaces. Luettavissa: https://docs.python.org/3/library/os.html. Luettu: 18.4.2025.

Python Software Foundation 2025. python-dotenv 1.1.0. Luettavissa: https://pypi.org/project/python-dotenv/. Luettu: 27.3.2025.

Python Software Foundation 22.4.2025. json — JSON encoder and decoder. Luettavissa: https://docs.python.org/3/library/json.html. Luettu: 22.4.2025.

Python Software Foundation 22.4.2025. re — Regular expression operations. Luettavissa: https://docs.python.org/3/library/re.html. Luettu: 22.4.2025.

Python Software Foundation 22.4.2025. socket — Low-level networking interface. Luettavissa: https://docs.python.org/3/library/socket.html. Luettu: 22.4.2025.

Python Software Foundation 22.4.2025. string — Common string operations. Luettavissa: https://docs.python.org/3/library/string.html. Luettu: 22.4.2025.

Python4dev 14.9.2024. Valokuva. Instagram. Luettavissa: https://www.instagram.com/python4dev/p/C_4rEO8tjdI/. Luettu: 20.2.2025.

Raspberry Pi 2024. Raspberry Pi 3 Model B+. Luettavissa: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/. Luettu: 21.1.2025.

Raspberry Pi. Raspberry Pi 3 Model B+. Luettavissa: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/. Luettu: 23.4.2025.

Schwaber, K. & Sutherland, J. 2020. The 2020 Scrum Guide. Luettavissa: https://scrumguides.org/scrum-guide.html. Luettu 20.4.2025.

Scrum 2025. What is Scrum? Luettavissa: https://www.scrum.org/resources/what-scrum-module. Luettu: 8.1.2025.

Shenzhen EAI Technology Co.,Ltd. 2025. YDLIDAR X4PRO. Luettavissa: https://www.ydlidar.com/products/view/21.html. Luettu: 23.4.2025.

Simsek B. 2004. EnderUNIX Software Development Team. Artikkeli. Luettavissa: https://web.archive.org/web/20060628062553/http://www.enderunix.org/simsek/articles/libraries.pdf. Luettu: 13.2.2025.

Thonny 2024. Python IDE for beginners. Luettavissa: https://thonny.org/. Luettu: 25.1.2025.

TIM 2024. Mitä ohjelmointi on? Luettavissa: https://tim.jyu.fi/view/kurssit/tie/ohj1/materiaali/mita-ohjelmointi-on#mit%C3%A4-ohjelmointi-on. Luettu: 20.2.2025.

Visual Studio Code 2024. Why did we build Visual Studio Code? Luettavissa: https://code.visualstudio.com/Docs/editor/whyvscode. Luettu: 24.1.2025.

W3schools 2025. Python Introduction. Luettavissa: https://www.w3schools.com/python/python_intro.asp. Luettu: 13.02.2025.

ZRob314 2025. Getting Started With ROS. AUTODESK, Inc. Luettavissa: https://www.instructables.com/Getting-Started-with-ROS-Robotic-Operating-Syste/. Luettu: 17.4.2025.