Teemu Gerasimoff

Mobiilirobotiikan hyödyntäminen katutaiteessa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Konetekniikka

Insinöörityö

23.11.2022

Tiivistelmä

Tekijä:	Teemu Gerasimoff
Otsikko:	Mobiilirobotiikan hyödyntäminen urbaanissa katutaiteessa
Sivumäärä:	40 sivua + 1 liite
Aika:	23.11.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Konetekniikka
Ammatillinen pääaine:	Koneautomaatio
Ohjaajat:	Lehtori Antti Liljaniemi

Insinöörityön tarkoituksena oli selvittää, kuinka robotiikkaa voidaan hyödyntää katutaiteessa. Asfalttiteiden maalaus ja ihmisen artistinen näkemys haluttiin yhdistää mobiilirobotiikalla. Työssä rakennettua mobiilirobottia ajettiin kesällä 2022 nuorisopajoissa ympäri Vantaata. Työ tehtiin projektina Metropolia Ammattikorkeakoulun BigFlash -hankkeelle. Toisena asiakkaana toimi SAV (Street Art Vantaa) Taidekollektiivi. Työssä myös tutustutaan tarkemmin erilaisiin mobiilirobotiikan liikkumistoteutuksiin ja erikoispyöriin.

Ohjaavana periaatteena projektissa oli luoda nuorille tarkoitettu käyttäjäystävällinen ohjausjärjestelmä. Robotin kauko-ohjaus sidottiin käytettyyn peliohjaimeen, joka on nuorisolle ennestään tuttu. Rakennettu maalausrobotti kehitettiin Raspberry Piin pienoistietokoneen ja vanhasta puretusta robotista saatujen sähkömoottorien ympärille.

Työn tuloksena saatiin helppokäyttöinen maalausrobotti, jolla voidaan käyttää kuutta erilaista maalia samanaikaisesti. Robotin ohjauksesta tehtiin maalaamiselle sopiva ja helposti opittava. Herkimmät elektroniset laitteet suojattiin 3D-tulostetuilla suojalaatikoilla. Maalausmekanismin kiinnitys suunniteltiin irrotettavaksi, minkä takia robotin käytön jälkeinen huolto oli helppoa.

Insinöörityötä voidaan käyttää yleishyödyllisenä pohjana kauko-ohjattavien mobiilirobottiprojektien suunnittelussa.

Avainsanat: Robotiikka, Maalausrobotti, Mobiilirobotiikka, omni-pyörä.

Abstract

Author:	Teemu Gerasimoff
Title:	Using Mobile Robotics in Urban Street Art
Number of Pages:	40 pages + 1 appendix
Date:	23 November 2022
Degree:	Bachelor of Engineering
Degree Programme:	Mechanical Engineering
Professional Major:	Machine Automation
Supervisors:	Antti Liljaniemi, Senior Lecturer

The purpose of this thesis work was to determine the viability of robotics in urban street art. This thesis work was commissioned by Metropolia University of Applied Sciences' Big-Flash project in which the customer was SAV (Street Art Vantaa) art collective. The designed mobile robot was used in youth workshops around Vantaa during the summer of 2022. This thesis also examines different types of movement configurations for mobile robotics, which also includes the inner workings of special type of wheels e.g., omni-wheels.

The guiding principle in the project was to create a user-friendly mobile robot. The main target group for the created robot was the youth of Vantaa. A general controller was used to make the controls of the robot more familiar for the youth. The created robot was built around a small computer called Raspberry Pi. In addition, an old robot of Metropolia was disassembled for the project, and its parts could be used for the new robot's electric motors and omni-wheels.

As a result, a mobile robot for painting streets was successfully designed and assembled. The robot has six canisters which can be simultaneously filled with different paints. The steering of the mobile robot was made to fit the streets that were to be painted and it was easy for the youth to learn to use it. The most delicate electronics were hidden inside of a 3D-printed protective cases in order to protect them from paint splatters. Finally, the painting mechanism was made to be modular, which made the maintenance and cleaning of the mobile robot effortless.

This thesis can be used as a general guide for the development of remote-controlled mobile robot projects.

Keywords: Robotics, Mobile Robotics, Omni-wheel, Painting Robot.

Sisällys

Lyhenteet

AM: Additive Manufacturing tarkoittaa materiaalia lisäävää valmistusmenetelmää.

AUV: Autonomous underwater vehicle eli autonominen vedenalainen kulkuneuvo.

CAD: Computer-aided Design tarkoittaa tietokoneavusteista suunnittelua.

DC: Direct current, eli tasavirta.

DMA: Direct Memory Access, eli oikosiirto tarkoittaa, että tietoja voidaan kopioida tietokoneen sisällä käyttämättä sen suoritinta.

GPIO: General Purpose Input/Output, yleiseen lähetettäviin tai vastaanotettuihin signaaleihin käytetty pinni.

pHAT: Pi Hardware Attached on Top tarkoittaa pinottavaa lisävarustusta, joka tuo helposti lisätoimintoja Raspberry Pi:n.

PWM: Pulse-width modulation on pulssinleveysmodulaatio, jossa kuormaan menevää jännitettä säädetään muuttamalla pulssisuhdetta.

Python: Monipuolinen ohjelmointikieli, jota ei tarvitse kääntää kirjoitetun ohjelman ajamiseksi.

RPI: Raspberry Pi 4 on pienoistietokone, jota käytetään erilaisten kevyempien ohjelmien ja/tai ohjeiden pyörittämiseen.

UAV: Unmanned aerial vehicle eli miehittämätön ilmakulkuneuvo.

WMR: Wheeled mobile robot eli raajallinen mobiilirobotti.

1 Johdanto

Insinöörityön tarkoituksena oli suunnitella ja toteuttaa käyttäjäystävällinen kauko-ohjattava maalausrobotti. Työssä tutkittiin, kuinka robotiikkaa voidaan käyttää ihmisen jatkeena katutaiteessa. Teiden maalaus ja ihmisen artistinen näkemys haluttiin yhdistää mobiilirobotiikalla.

Työ tehtiin Suvituuli- ja Salamaprojektina Metropolia Ammattikorkeakoulun Big-Flash -hankkeelle, joka kehittää yritysten kanssa uusia nousevia teknologisia ratkaisuja. Suvituuli- ja Salamaprojektit kuvaavat tuotteen eri vaiheita kehitetystä ideasta konkreettiseen tuotteeseen asti. Toisena asiakkaana toimi SAV (Street Art Vantaa) Taidekollektiivi, joka luo seinämaalausteoksia ja suunnittelee kokonaisia kaupunginosia käsittäviä taidekokonaisuuksia.

Maalausrobotti suunniteltiin ja rakennettiin Vantaan nuorisopajojen aktiviteetiksi kesälle 2022, missä sitä käytettiin ennalta sovittujen suojateiden maalaamiseen. Pajat, joissa maalausrobottia ajettiin, pidettiin Myyrmäen urheilupuistossa, Tikkurilan kirkon edessä ja Havukosken nuorisotalolla.

Aluksi insinöörityössä käydään läpi robotiikkaa ja sen yleisempiä muotoja. Tämän jälkeen käydään syvemmin erikoisia pyörätoteutuksia mobiilirobotiikassa, mitä itse työssä käytettiin hyväksi. Seuraavaksi käsitellään robotin kehikon suunnittelua ja maalausmekanismin toteutusta. Myös maalausrobotin elektroniikan valinta ja sähkösuunnittelu käydään läpi. Työssä hyödynnettiin myös 3D-tulostus teknologiaa, millä suojattiin herkemmät robotin elektroniset komponentit. Testiajossa ilmenneet ajatukset raportoitiin ja työpajoilta saadut käyttökokemukset ja mielipiteet otettiin myöhemmin huomioon robotin kehityksessä. Insinöörityön loppuun on kirjoitettu yhteenveto, missä pohditaan ja käydään läpi insinöörityön tuottamia tuloksia.

2 Mobiilirobotiikan yleisimmät muodot

2.1 Robotiikka

Nykyään robotiikkaa hyödynnetään yhä useammalla alalla. Robotteja voidaan hyödyntää yksinkertaisissa toistuvissa liikkeissä tai monimutkaisemmissa kokonaisuuksissa. Itsenäiset robotit ovat ohjelmoitu toimimaan ilman ihmisen väliintuloa. Tekoälyn avulla se voi työskennellä, sekä selviytyä itsenäisesti ympäristössään. (Tzafestas 2013: 2.)

Antureiden käyttö robotiikassa on hyvin tärkeää takaisin kytketyiden ohjaussilmukoiden sulkemisessa, millä varmistetaan robottien automatisoitu ja autonominen toiminta tosielämän sovelluksissa. Roboteille suunnitellut anturit muistuttavat hyvin paljon ihmisen aistijärjestelmää, kuten: näkö-, kuulo- ja asentoaisti. Nämä anturit antavat robotille syöttösignaaleja, joita käytetään ulkoisen tiedon prosessointiin ja robotin fyysisten liikkeiden luomiseen. Tunnistusmenetelmät tarjoavat robotille kyvyn hienostuneempaan ongelmanratkaisuun, millä saadaan aikaan paranneltu versio perinteisestä esiohjelmoidusta toistuvasta suorituksesta. Robotti kykenee siis itsenäisesti reagoimaan ulkoisiin muuttujiin. (Tzafestas 2013: 101.)

Robotin anturit jaetaan yleisesti analogisiin- ja digitaalisiin antureihin. Analogiset anturit tarjoavat robotille analogisia lähtösignaaleja, jotka tarvitsevat analogia-digitaalimuunnoksen, jotta käytetty tietokone pystyy ymmärtämään sisään tulevan signaalin. Esimerkkejä analogisista antureista ovat: analoginen infrapunaetäisyysanturi, mikrofoni ja analoginen kompassi. Digitaaliset anturit ovat tarkempia kuin analogiset anturit, ja niiden datamuodot voivat olla eri muotoisia. Niillä voi esimerkiksi olla synkroninen sarjarakenne, eli data luetaan bitti bitiltä, tai rinnakkaisrakenne. Kaikissa tapauksissa anturien ominaisuuksiksi halutaan:

- korkea resoluutio,
- laaja toiminta-alue,
- nopea vasteaika,
- helppo kalibrointi,
- korkea luotettavuus.

Sensoreihin kuuluvat myös visuaaliset kamerat ja niiden toimintaan tarvittavat apuvälineet. (Tzafestas 2013: 101.)

Roboteissa käytetyt anturit voidaan ryhmitellä seuraaviin luokkiin:

- mekaaninen rajakytkin,
- akustinen anturi,
- sähkömagneettinen anturi,
- magneettinen anturi,
- optinen anturi.

Mekaaniset-, sähkömagneettiset- ja magneettiset anturit soveltuvat parhaiten lähellä olevien kappaleiden tunnistamiseen, kun taas robotin liikkeen aikana tapahtuvaan paikan muutoksen mittaukseen paremmin soveltuvat akustiset- ja optiset anturit, niiden pidemmän kantaman takia. Mekaaniset rajakytkimet vaativat fyysisen kontaktin robotin ja anturin välillä, minkä takia ne ovat usein integroitu robotin runkoon. Akustiset ja sähkömagneettiset anturit käyttävät lähetettyjen ja vastaanotettujen signaalien suuntaa ja lentoaikaa laskeakseen kohteen kulman ja lineaarisen sijainnin. Akustiset järjestelmät käyttävät hyväksi ultraäänitaajuuksia robotin ympäristön hahmottamiseen. Sähkömagneettisiin anturijärjestelmiin kuuluvat optiset, laser- ja tutkalaitteet. Molemmissa tapauksissa lähettimen ja vastaanottimen välillä vaaditaan selvä näköyhteys, jotta tarkka mittaus voidaan suorittaa. Magneettiset anturit käyttävät kohteen ja anturin välisiä magneettikenttiä sijainnin laskemiseen, kun taas optiset anturit käyttävät tilanteeseen sopivia näkökameroita. (Tzafestas 2013: 102–103.)

2.1.1 Yhteistyörobotiikka

Yhteistyörobotti eli kobotti on robotti, joka voi turvallisesti ja tehokkaasti olla vuorovaikutuksessa ihmisten kanssa erilaisissa yhteisissä tehtävissä. Yhteistyöllä voi olla monia muotoja, kuten rakennusmateriaalin siirtäminen ihmiselle tai robottiavusteinen komponenttien rakennus. Robotin tehtävissä voidaan hyödyntää sen voimaa tai ulottuvuutta, millä täydennetään ihmisen fyysisiä kykyjä. Tärkeänä tekijänä missä tahansa tilanteessa, jossa kobotti on osallisena, on ihmisen ja robotin välisen työskentelytilan käyttö. Toisin kuin teollisuusrobotit, kobottia ei ole aidattu tai sijoitettu erilliseen tilaan, joten ihmisten ja robottien on toimittava turvallisesti yhdessä. (Greenspan & Matthews 2020: 51–52.)

Ihmisten ja robottien välinen yhteistyö on yksi vaikeimmista pulmista ratkaista, mutta sillä on potentiaalia parantaa jaettuja tiloja ja yhteisien tehtävien tehokkuutta. Tämä edellyttää osapuolilta toimivaa viestintää ja yhteisymmärrystä, sekä hienostuneita ohjelmallisia käytäntöjä ja erilaisia mittauksia. Toinen yhteistyörobottien tärkeä ominaisuus on sen kyky soveltaa ohjeiden tulkintaa. Robotin odotetaan selviytyvän yksinkertaisista ohjeista, kuten: "Siirrä tuoli pois tieltä" tai "Siivoa tuo sotku". Näiden ohjeiden tulkinta ja turvallinen toimeenpano asettaa ylimääräisiä vaatimuksia kunkin toimialueen kokonaisuuden käsittely-, viestintä- ja riskienhallintakapasiteetilta. (Greenspan & Matthews 2020: 59.)

2.1.2 Mobiilirobotiikka

Mobiilirobotit ovat robotteja, jotka voivat liikkua paikasta toiseen itsenäisesti eli ilman ulkopuolisten ihmisten apua. Toisin kuin useimmat teollisuusrobotit, mobiilirobottien erityispiirteenä on se, että ne voivat liikkua vapaasti ennalta määritetyissä työtiloissa. Tämä liikkuvuus tekee niistä sopivia monenlaisille työympäristöille, jotka voivat olla, joko järjestelmällisiä, tai aktiivisesti muuttuvia. Maalla liikkuvat robotit yleensä erotetaan pyörillä liikkuvista roboteista (engl. wheeled mobile robots, WMR) jalallisiin robotteihin (engl. legged mobile robots, LMR). Mobiilirobotteihin sisältyy myös miehittämättömät ilma-alukset (engl. unmanned aerial vehicles, UAV) ja autonomiset vedenalaiset ajoneuvot (engl. autonomous underwater vehicles, AUV). (Tzafestas 2013: 101–102.)

Nykypäivän mobiilirobotit voivat:

- liikkua turvallisesti sotkuisessa ympäristössä,
- ymmärtää luonnollista puhetta,
- tunnistaa todellisia esineitä,
- paikantaa itsensä,
- suunnitella polkuja,
- yleensä ajatella itsenäisesti.

Älykäs mobiilirobotti pystyy hyödyntämään kognitiiviseen ja älykkääseen käyttäytymiseen perustuvia ohjauksenmenetelmiä ja -tekniikoita, kuten tekoälyä ja koneoppia. Tehokkaiden mobiilirobottien on maksimoitava suorituskyvyn joustavuus ulkoisten ohjeiden minimaalisella määrällä ja vähentää käytettyjen laskennallisten ohjeiden monimutkaisuutta. (Tzafestas 2013: 2.)

2.2 Mobiilirobotin liikunta konfiguraatioita

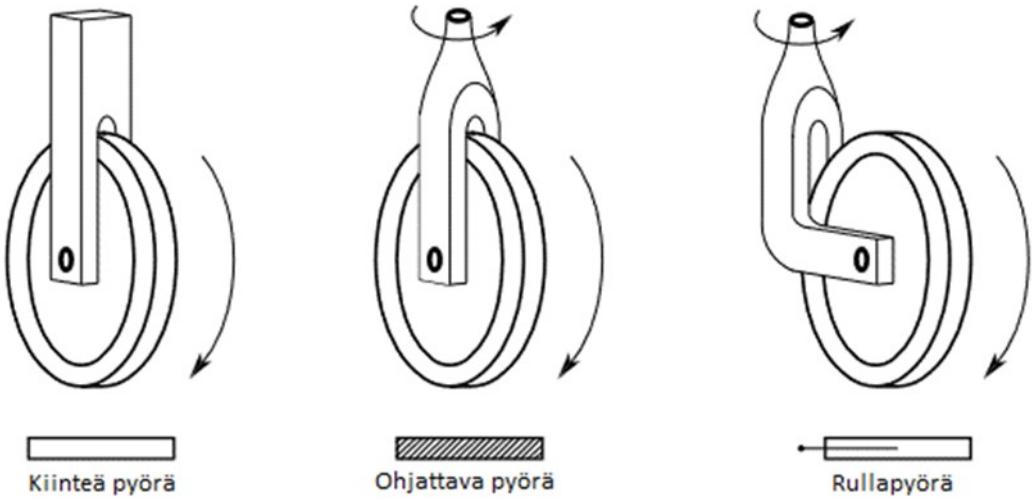
Mobiilirobotiikassa käytetyt pyörät ovat yleisesti yksinkertaisia mekanismeja, jotka koostuvat pyörivällä nivelellä varustetuista levyistä, kun taas robotin jalat koostuvat useammasta nivelestä ja niiden yhdistyksistä. Toisaalta hyvin yksin-

kertainen jaloilla varustettu robotti tarvitsee vähintään kolmen vapausasteen liikkeen. Mikä tarkoittaa, että mobiilirobotti tarvitsee vähintään kolme eri toimielintä toimivan pidon ja suunnan aikaansaamiseksi. Staattisesti vakaalle jalkarobotille taas tarvitaan vähintään neljä nivellettyä jalkaa, mikä nostaa jalkarobotin toimilaitteiden lukumäärän vähintään 12:sta. Tämän takia sähkömekaaninen järjestelmä on monimutkaisempi ja kalliimpi toteuttaa jalkarobotille kuin pyörillä varustetulle robotille. (Gonzalez de Santos ym. 2006: 17–18.)

Jalkarobottien liikkuvuus on yleisesti parempi kuin pyörillä varustettujen robottien, koska ne ovat luonnostaan monisuuntaisia järjestelmiä. Jalkarobotti voi muuttaa suuntaansa vaihtamalla jalkojensa paikkaa riippumatta päärungon akselin suunnasta. Toisaalta tavanomaisilla pyörillä varustetun robotin pitäisi liikuttaa jonkin verran sen kehoa voidakseen muuttaa suuntaa. Jalkarobotti voi myös liikkua ja muuttaa kehonsa suuntausta pidentämällä vain sen jalkoja. Tämä ominaisuus tarjoaa robotille luonnollisen jousituksen mukauttamalla jalkojen pituuksia maaston epätasaisuuksiin. Jalkarobotti voi myös kattaa erittäin epäsäännöllisiä maastoja sen kehon pysyessä vaakatasossa. Se voi myös ylittää robotin ja maan väliin jäävän maavaran alittavat esteet pelkästään astumalla niiden päälle. (Gonzalez de Santos ym. 2006: 18–22.)

Jalkarobotin jokaisella toimilaitteella on siihen liittyvä oma teho-ohjain, minkä takia ne vaativat enemmän elektronisia järjestelmiä kuin pyörillä varustetut robotit. Toisena ongelmana on robotin nivelten ohjattavuus. Pyörät ovat aina kosketuksissa maahan, kun taas jalat vuorottelevat asento- ja liikevaiheiden välillä. Tämän takia ohjausjärjestelmä vaatii robotilta ylimääräisiä antureita määrittääkseen, milloin eri jalat koskettavat maata. Kosketusanturin tai jonkinlaisen vastaavan anturin sisällyttäminen jokaiseen jalkaan lisää anturitietojen käsittelyyn tarvittavien elektronisten korttien määrää. Nelijalkaisen robotin on kuitenkin koordinoitava samanaikaisesti kaikkien kahdentoista nivelensä liikettä sekä jalka-antureitaan, jotta liike saataisiin vakaaksi. Joten voidaan sanoa, että jalkaisten robottien ohjausalgoritmit ovat monimutkaisempia kuin pyörillä varustettujen robottien algoritmit. (Gonzalez de Santos ym. 2006: 20–21.)

Pyörillä varustetut mobiilirobotit edustavat suurinta osaa käytetyistä liikuntamalleista. Tällaisten robottien mekaaninen peruselementti on todellakin pyörä. Perinteisiä pyöriä on kolmea eri tyyppiä, jotka ovat esitetty kuvassa 1. (Oriolo ym. 2010: 10–11).

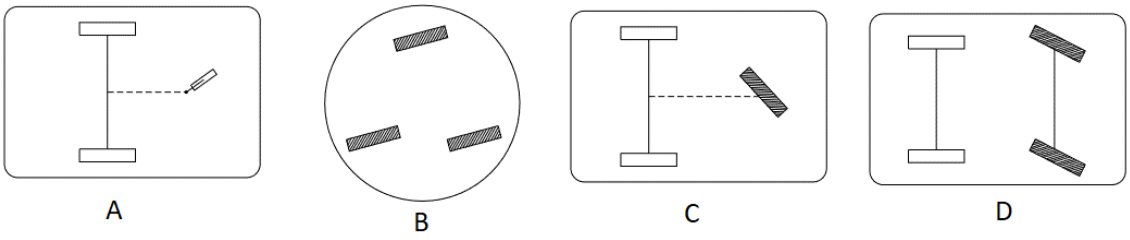


Kuva 1. Mobiilirobotiikassa käytettyjä pyörämalleja (Oriolo ym. 2010: 11).

- Kiinteä pyörä voi pyöriä akselinsa ympäri, joka kulkee pyörän keskustan läpi ja on kohtisuorassa sen tasoon nähden. Pyörä on kiinnitetty jäykästi alustaan, minkä takia pyörimissuunta pyörään nähden pysyy vakiona.

- Ohjattavassa pyörässä on kaksi pyörimisakselia. Ensimmäinen akseli on sama kuin kiinteässä pyörässä, kun taas toinen akseli mahdollistaa pyörän kääntymisen sivusuunnassa. Tämä antaa pyörän muuttaa pyörimissuuntaansa suhteessa alustaan.

- Rullapyörässä on myös kaksi pyörimisakselia, mutta pystyakseli ei kulje pyörän keskipisteen läpi, vaan se on tasaisessa siirtymässä. Tällainen järjestely saa pyörän kääntymään automaattisesti ja nopeasti linjaamaan itsensä alustan liikesuuntaan. Rullapyörää käytetään, jotta tukipisteen staattisen tasapainon muodostaminen ei vaikuttaisi alustan liikkuvuuteen. Rullapyöriä käytetään esimerkiksi ostoskärryissä, sekä pyörillä varustetuissa tuoleissa. (Oriolo ym. 2010: 11.)



Kuva 2. Pyörä konfiguraatioita (Oriolo ym. 2010: 12–13).

Tasauspyörästö vetoisessa mobiilirobotissa on kaksi kiinteää pyörää, joilla on yhteinen pyörimisakseli. Yhden tai useamman rullapyörän tehtävänä on pitää robotti staattisessa tasapainossa (kuva 2 A). Kahta kiinteätä pyörää ohjataan erikseen. Erilaisia kulmanopeuden arvoja takapyörille voidaan asettaa ohjelmallisesti, kun taas kääntöpyörä on passiivinen. Tällainen robotti voi pyöriä paikallaan, jos kahden pyörän kulmanopeudet ovat samat mutta vastakkaiset. (Oriolo ym. 2010: 12.)

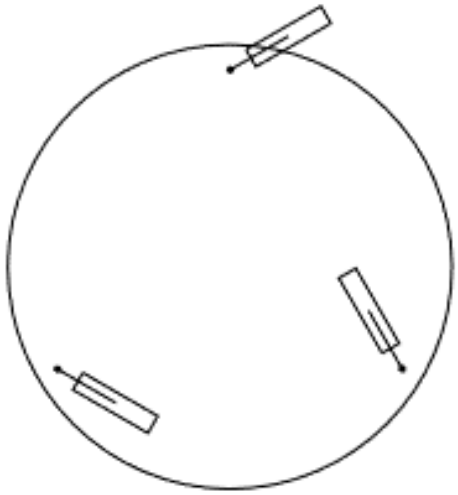
Mobiilirobotti, jolla on samanlainen liikkuvuus kuin tasauspyörästövetoisella, saadaan käyttämällä kolmen ohjattavan pyörän järjestelyä (kuva 2 B). Robotissa on kolme toisiinsa yhdistettyä ohjattavaa pyörää, joita ajetaan synkronisesti kahdella moottorilla. Pyörät ovat yleensä mekaanisesti kytketty toisiinsa esimerkiksi ketjun avulla. Ensimmäinen moottori ohjaa pyörien pyörimistä vaaka-akselin ympäri ja tarjoaa siten pitoa mobiilirobotille. Toinen moottori ohjaa pyörien pyörimistä pystyakselin ympäri, mikä vaikuttaa niiden suunnanvaihtoon. Alustan suunta ei kuitenkaan muutu liikkeen aikana. (Oriolo ym. 2010: 12.)

Kolmipyöräisessä mobiilirobotissa (kuva 2 C) on kaksi kiinteää pyörää asennettuna taka-akselille ja ohjattava pyörä sijaitsee edessä. Kiinteitä pyöriä ajetaan yhdellä moottorilla, joka ohjaa niiden pyörimistä. Kun taas ohjattavaa pyörää ohjaa toinen moottori, joka muuttaa suuntaa ja toimii siten ohjauslaitteena. Vaihtoehtoisesti kaksi takapyörää voivat olla passiivisia ja etupyörä voi tarjota robotille vedon ja ohjauksen. (Oriolo ym. 2010: 12–13.)

Ajoneuvomaisessa mobiilirobotissa on kaksi kiinteää pyörää kiinnitettynä taka-akselille ja kaksi ohjattavaa pyörää etuakselilla, mikä voidaan nähdä kuvassa 2 D. Kuten edellisessä tapauksessa, yksi moottoreista tarjoaa laitteen edessä tai takana vetovoimaa, kun taas toinen moottori vaihtaa etupyörien suuntaa. Pitää kuitenkin muistaa, että liukumisen välttämiseksi molempien etupyörien on oltava eri suunnissa robotin liikkuessa kaarteessa. Erityisesti kaarteessa oleva sisäpyörä tarvitsee enemmän ohjausta kuin ulkoinen pyörä. (Oriolo ym. 2010: 13.)

Yleisten pyörien lisäksi mobiilirobotteja voidaan ohjata tela-ajolla, mikä on erittäin tehokas epätasaisessa maastossa. Lainehtivan käärmeen liikkumisesta inspiroitu aaltoileva liike voidaan myös saavuttaa ilman erityisiä laitteita. Erilaisia ilma- ja vedenalaisia liikkumismuotoja on myös kehitelty.

Viimeiseksi kuvan 3 robotti, jossa on yleensä symmetrisesti paikoitettu kolme tai neljä erikoispyörää, joita ajetaan itsenäisesti, mahdollistaa kaiken suuntaisen liikkeen. Toisin kuin aikaisemmissa tapauksissa, robotti voi liikkua hetkessä mihin tahansa karteesiseen suuntaan. Se myös pystyy paikoillaan suuntautua uudelleen haluttuun suuntaan. (Oriolo ym. 2010: 11.)

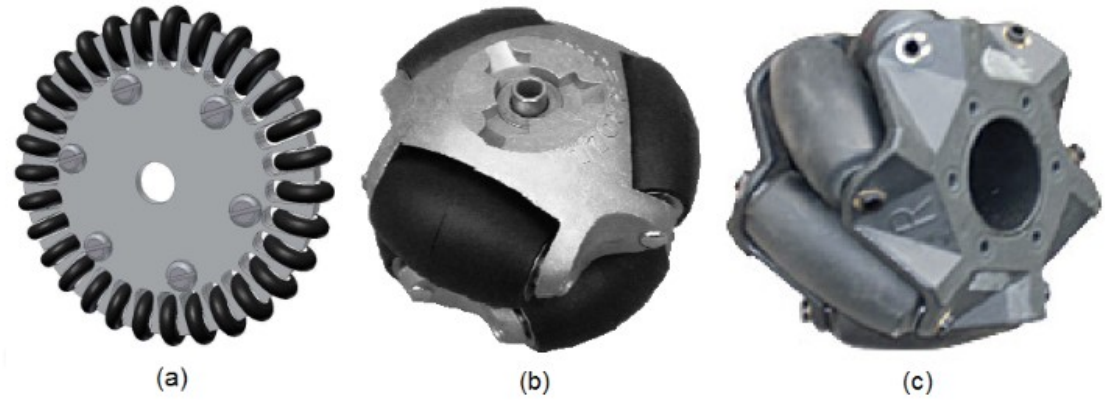


Kuva 3. Suuntaukseton pyöräjärjestelmä (Oriolo ym. 2010: 14).

2.3 Suuntauksettomat pyörät

Vaikka omni-pyöriä (suuntaukseton pyörä) on monia erilaisia, niiden toimintaperiaate on yleensä sama. Pyörän pyöriessä ilmenevässä maan ja pyörän välisessä kosketuspisteessä sen on mahdollista vieriä kahteen eri suuntaan samanaikaisesti. Jos omni-pyörä on ajossa, eli kytkettynä pyörivään moottoriin, sen moottorin suuntainen pyörimisakseli on ensisijainen akseli, kun taas toissijainen akseli pääsisi pyörimään vapaasti. Kuitenkaan vapaasti pyörivän akselin suunta ei ole samansuuntainen ajetun käyttösuunnan kanssa. Jotta voitaisiin saavuttaa todellinen vapaaliike tasossa, on kokoonpanossa käytettävä vähintään kolme pyörää, jotka ovat aseteltu asentoon. Tämä mahdollistaa kolmen vapausasteen liikkeen. Ajoneuvo ei pysty ainoastaan ajamaan eteen- ja taaksepäin, vaan myös vaakasuunnassa, vinottain, pyörimään akselinsa ympäri tai jopa yhdistää siirtymisen ja pyörimisen. (Bemis 2007: 10.)

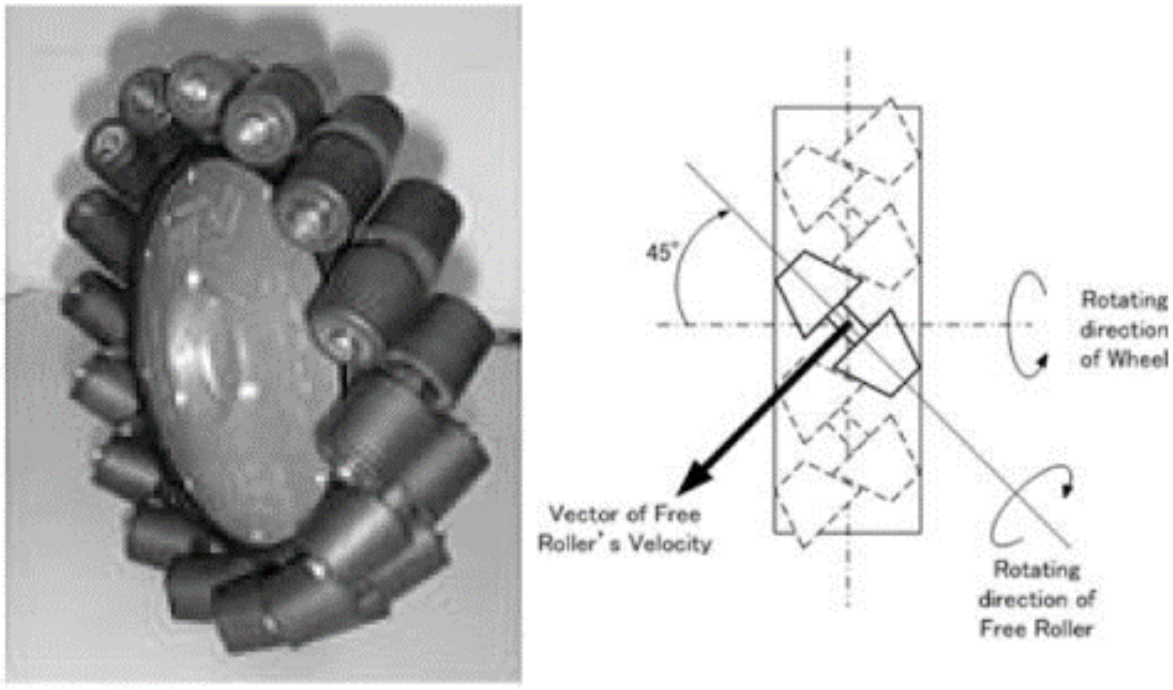
On olemassa kolme päätyyppistä omni-pyörää, joita käytetään ensisijaisesti monisuuntaisilla alustoilla, kuten roboteilla. Ne ovat: segmentoitu omni-pyörä, kaksinkertainen omni-pyörä ja mecanum-pyörä (tai ruotsalainen pyörä). Jokaisella näistä pyöristä on omat edut ja haitat. Segmentoidussa monisuuntaisessa pyörässä on sarja rullia, jotka on asennettu suuremman pyörän kehän ympärille, kuten kuvassa 4a näkyy. Telat on asennettu siten, että ne rullaavat kohtisuoraan pääakselia nähden. Koska on mahdotonta saada tarpeeksi pyöriviä putkia kattamaan pyörän koko kehän, niiden väliin jätetään rako. Tästä kuitenkin aiheutuu tärinää ja naksahtelua, kun pyörä pyörii alustalla. (Bemis 2007: 10.)



Kuva 4. Kolme käytetyintä omni-pyörä tyyliä (Bemis 2007: 11).

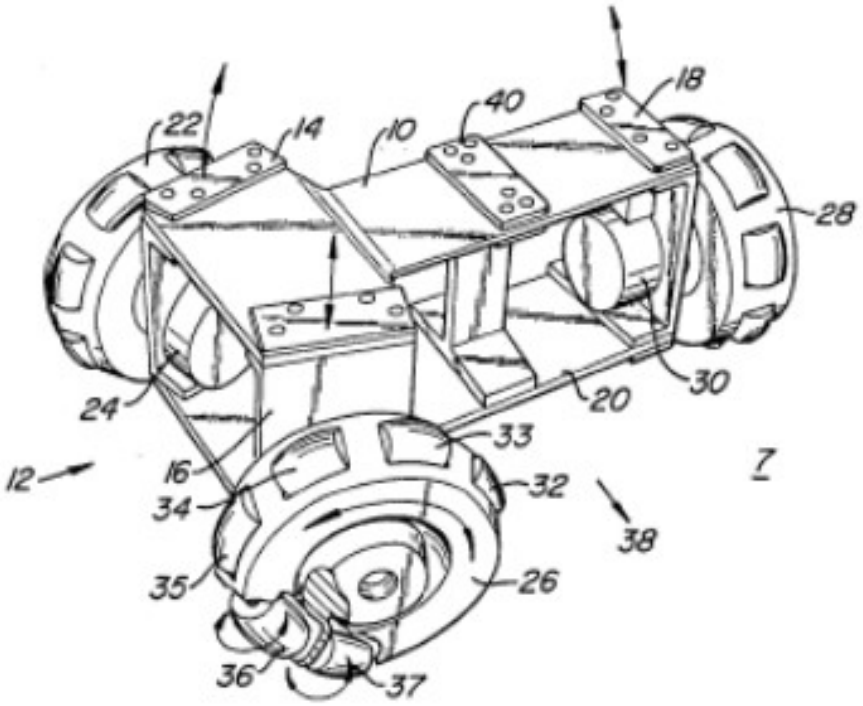
Kaksinkertainen monisuuntainen pyörä koostuu kahdesta toisiinsa lukitusta segmentoidusta omni-pyörästä kuvan 4b mukaisesti. Tämä pyörä ratkaisee naksahdusongelman, kun pyörä kulkee pintaa vuorotellen pyörän ja putken välillä. Siksi pyörässä on aina yksi pyörivä kappale, joka on kosketuksissa lattiaan. Pyörään kohdistuva voima vaihtelee vapaasti pyörivän putken ja pyörän välillä, koska ne vaihtelevat kosketusta liikkuvaan tasoon vuorotellen. Tämä voi kuitenkin aiheuttaa ylimääräistä melua ja tärinää pyörän liikkuessa epätasaisilla pinnoilla. (Bemis 2007: 10–11.)

Mecanum-pyörä on samanlainen kuin segmentoitu omni-pyörä sillä poikkeuksella, että tynnyrit on asennettu 45 asteeseen sen sijaan, että ne olisivat kohtisuorassa suuremman pyörän kehään nähden. Pyörien on aina toimittava toisiaan vasten halutun liikkeen saavuttamiseksi putkien sijoittelun vuoksi. Tämä aiheuttaa voimia runkoon, joka yhdistää pyörät toisiinsa. (Bemis 2007: 11.)



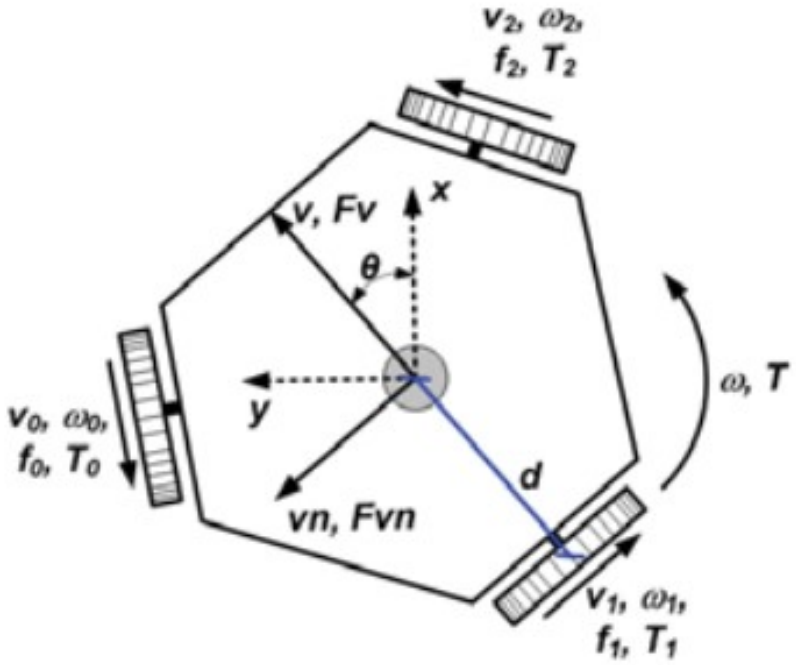
Kuva 5. Mecanum pyörä ja sen profiili (Bemis 2007: 11).

On myös kehitelty muita ainutlaatuisia omni-pyöriä, mitkä kaikki toimivat sillä periaatteella, että jokainen pyörä voi kulkea useampaan kuin yhteen suuntaan kerrallaan. On myös muita alustoja, jotka voivat tuottaa vapaata liikettä käyttämällä pyöriviä pyöriä, joiden kulkusuuntaa ohjataan. Tätä ei kuitenkaan pidetä todellisena monisuuntaisena ajona, koska pyöriä on käännettävä kulkusuunnan vaihtamiseksi. (Bemis 2007: 11–12.)



Kuva 6. Esimerkki omni-pyörien asettelusta (Bemis 2007: 12).

Omni-pyöriä käyttävien robottien dynaamiset mallit ovat harvinaisia, koska pyörien useamman sisäisen kitkan mallintaminen on ilmentynyt ongelmaksi. Mallit rakennetaan yleensä tiettyjen valmiiksi tehtyjen pyörien ympärille. Yleisimmät kokoonpanot ovat, joko kolme- tai neljäpyöräisiä. Kolmipyöräiset järjestelmät ovat mekaanisesti yksinkertaisempia, mutta nelipyöräisillä roboteilla on enemmän kiihtyvyyttä samanlaisilla moottoreilla. Nelipyöräisillä roboteilla voidaan yleisesti olettaa vähemmän pyörien luistamista olettaen, että kaikki pyörät painetaan lattiaa vasten tasaisesti. (Costa ym. 2008:189.)



Kuva 7. Kolmepyöräisen robotin kinematiikka (Costa ym. 2008:190).

Kolmipyöräisessä järjestelmässä pyörät on erotettu 120 asteella (Costa ym. 2008:190). Kuvassa 7 näytetään kaikki kolmepyöräiseen robotin pyöriin kohdistuvat voimat ja vektorit. Omni-pyöriin liittyvissä matemaattisissa kaavoissa käytetään ainoastaan kuvan 7 näyttämiä muuttujia.

x,y,θ ovat robotin paikka (x,y) ja kulma θ robotin keulasta,
d [m] on pyörien ja robotin keskipisteen välinen matka,
v_0, v_1, v_2, [m/s] ovat pyörien lineaarinopeus,
ω_0, ω_1, ω_2, [rad /s] ovat pyörien kulmanopeus,
f_0, f_1, f_2, [N] ovat pyöriin kohdistuvat voimat,
T_0, T_1, T_2, [N ·m] pyörien vääntömomentit,
v, vn, [m/s] ovat robotin lineaarinopeus,
ω [rad/s] on robotin pyörimisnopeus,
F_v, F_{vn} [N] ovat robottiin kohdistuvat v ja v_n suuntaiset voimat,
T [N ·m] on robotin vääntömomentti sen pyörimisnopeuteen suhteutettuna.

Edellä mainituista muuttujista voidaan laskea omni-pyörien kinemaattiset ominaisuudet. Yleisesti tunnettu kinemaattinen malli suuntauksettomalle robotille paikalla (x,y,θ) saadaan kaavoista (1,2,3). (Costa ym. 2008:190).

$v_x(t)=dx(t)/dt$ (1)

$v_y(t)=dy(t)/dt$ (2)

$\omega(t) = d\theta(t)/dt$ (3)

Seuraavaksi kaavalla 4 voidaan muuttaa pysyvältä akselilta F_v ja F_{vn} lineaariset nopeudet v_x ja v_y robotin v ja vn -akseleille (Costa ym. 2008:190).

$$X_R = \begin{bmatrix} v(t) \\ vn(t) \\ \omega(t) \end{bmatrix}; X_0 = \begin{bmatrix} v_x(t) \\ v_y(t) \\ \omega(t) \end{bmatrix}$$

$$X_R = \begin{bmatrix} \cos(\theta(t)) & \sin(\theta(t)) & 0 \\ -\sin(\theta(t)) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} * X_0 \quad (4)$$

Kolmepyöräisen robotin pyörien nopeudet v0, v1 ja v2 tulevat robotin nopeuksista v, v_n ja ω kaavan 5 mukaisesti (Costa ym. 2008:190).

$$\begin{bmatrix} v_0(t) \\ v_1(t) \\ v_2(t) \end{bmatrix} = \begin{bmatrix} -\sin(\frac{\pi}{3}) & \cos(\pi/3) & d \\ 0 & -1 & d \\ \sin(\pi/3) & \cos(\pi/3) & d \end{bmatrix} * \begin{bmatrix} v(t) \\ vn(t) \\ \omega(t) \end{bmatrix} \quad (5)$$

Jos käytetään käänteiskinematiikkaa, voidaan saada yhtälöitä, jotka määrittävät robotin nopeuden pyörien nopeuden suhteen. Ratkaistut yhtälöt v, vn and ω suhteen näyttävät seuraavilta: (Costa ym. 2008:190).

$$v(t) = \left(\sqrt{3}/3\right) * (v_2(t) - v_0(t)) \quad (6)$$

$$vn(t) = \left(\frac{1}{3}\right) * \left(v_2(t) + v_0(t)\right) - \left(\frac{2}{3}\right) * v_1(t) \quad (7)$$

$$\omega(t) = (1/(3 * d)) * (v_0(t) + v_1(t) + v_2(t)) \quad (8)$$

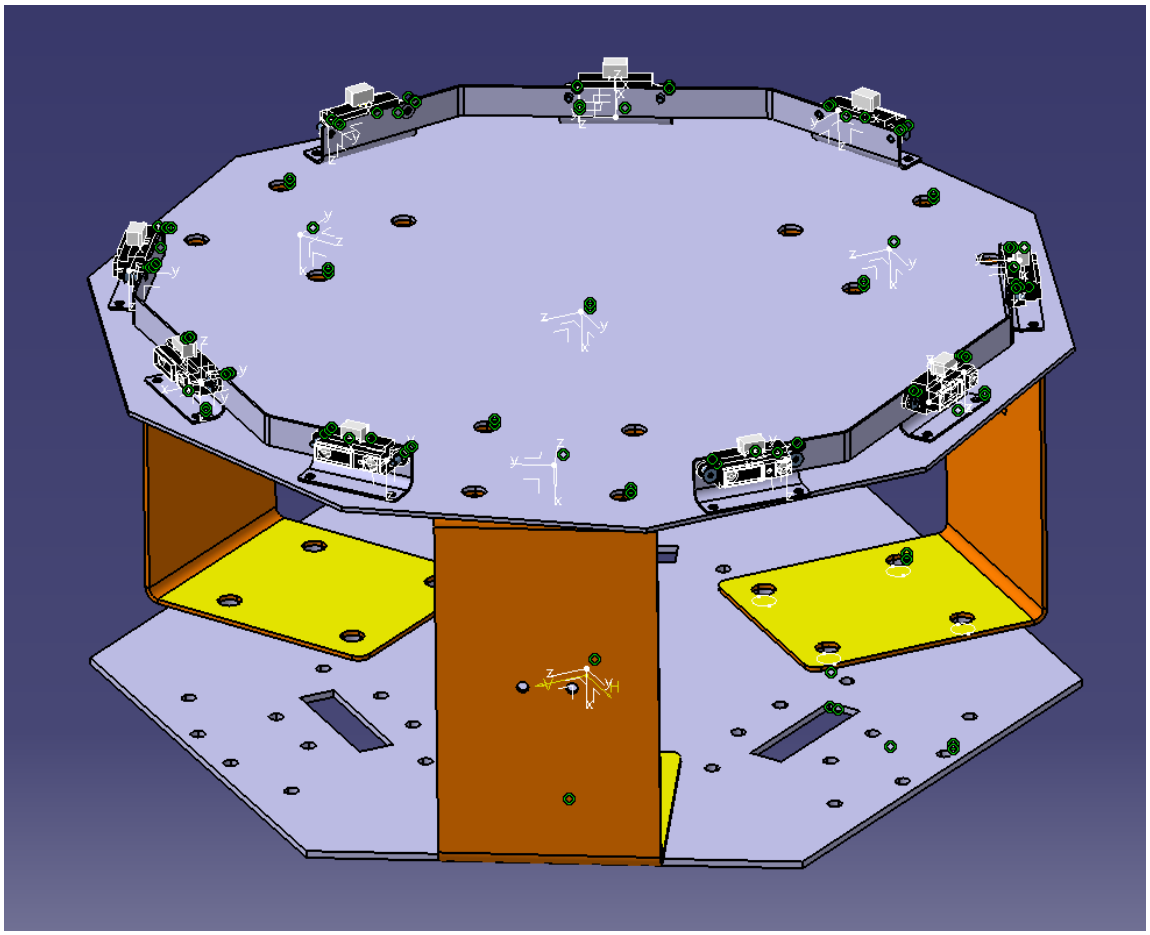
3 Maalausrobotin suunnittelu ja toteutus

3.1 Rungon suunnittelu

Robotin rungon suunnittelu aloitettiin käyttäen CATIA V5 -mallinnusohjelmistoa. Inspiraationa haettiin toisessa projektissa käytettyä vanhan robotin kehikkoa, joka purettiin uudelleen käyttöä varten, koska siitä löytyvät sähkömoottorit, vaihteistot ja anturit haluttiin hyödyntää. Maalausrobotin alatasossa käytettiin 3 mm paksua kuusikulmaista laserleikattua teräslevyä, millä saatiin sijoitettua moottorit 120 asteen kulmaan, jotta käytetyistä omni-pyöristä saatiin mahdollisimman tasainen liike. Työn alussa ei pystytty arvioimaan, minkälainen maalausmekanismi robotille tehtäisiin niin robotin pohja ylimitoitettiin, jotta mahdolliset tulevat rasitukset maastossa liikkumisesta ja mekanismin käytöstä eivät vaikuttaisi robotin jämäkkyyteen. Ylätasossa hyödynnettiin vanhasta robotista saatuja anturei den kiinnikkeitä, joista saatiin kymmenkulmainen muoto. Maalausrobotin sivutuet ja yläosa tehtiin 2 mm paksusta teräslevystä. Sisätiloihin piilotettiin turvaan robotissa käytetyt:

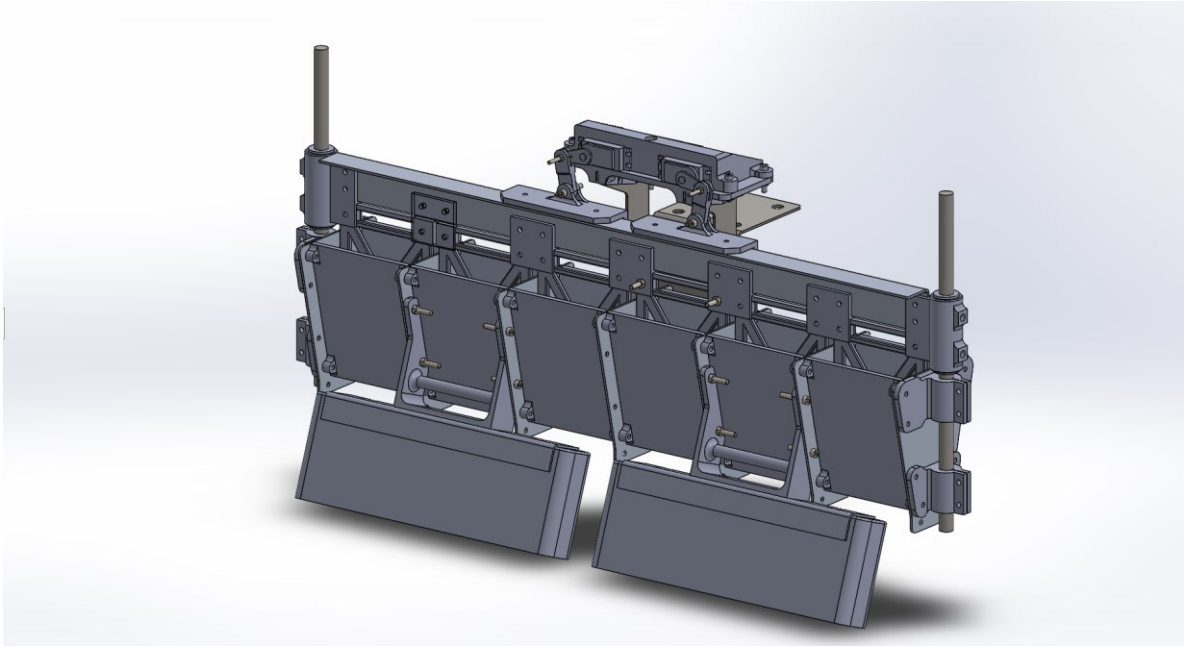
- vaihteistot,
- suuri osa virransiirrosta,
- moottorien ohjauskortit,
- logiikan signaalijohdot.

Elektroniikalle luotiin oma erillinen 3D-printattu ympyrän muotoinen musta laatikko, joka ruuvattiin kiinni robotin ylätasoon.



Kuva 8. Ensimmäinen konseptikehikko.

Maalausmekanismi toteutettiin projektiassistentin toimesta. Maalausmekanismista löytyy maalisäiliöt ja liikkuva osa, millä tukitaan ja avataan säiliöissä olevat valutusraot. Tukkiva liike saadaan ohjaamalla liikkuvaan osaan kiinnitetyillä servomoottoreilla. Mahdollinen nostosta ilmenevä sivuittainen liike estetiin ohjaavilla terästangoilla, joihin liikkuva osa kiinnitettiin laakereilla. Maalausmekanismi kiinnitettiin robotin takaosassa olevaan sivutukeen. Mekanismi voitiin purkaa kolmeen eri osaa, mikä mahdollisti laitteiston helpon pesun käytön jälkeen. Laitteessa hyödynnettiin paljon 3D-tulostamista, mikä mahdollisti uusien osien nopean tuotannon ja vaihtamisen.



Kuva 9. Mallinnettu maalausmekanismi.

Robotin päälle laitettiin Rapsberry Pi 4 B, servomoottorien jännitteen laskemiseen käytetyt jännitteenmuuntimet ja akku asennettiin robotin etuosioon, millä voitiin kertoa mihin suuntaan maalausrobotin keula osoitti. Tämä auttoi ajossa ohjauksen suunnan hahmottamisen. Takamoottorit asennettiin vinoon, mikä antoi lisää vakautta robotille, koska suurin osa laitteen massasta keskittyi robotin takaosaan. Maalausmekanismissa olevat pensselit laskettiin maahan asti, jotta saatiin hyvä kontakti asfaltin ja pensseleiden väliin. Tämä myös tarjosi hyvän takatuen robotille ajon aikana.

Kuva 10. Rakennettu maalausrobotti testiajoissa.

3.2 Elektroniikan valinta

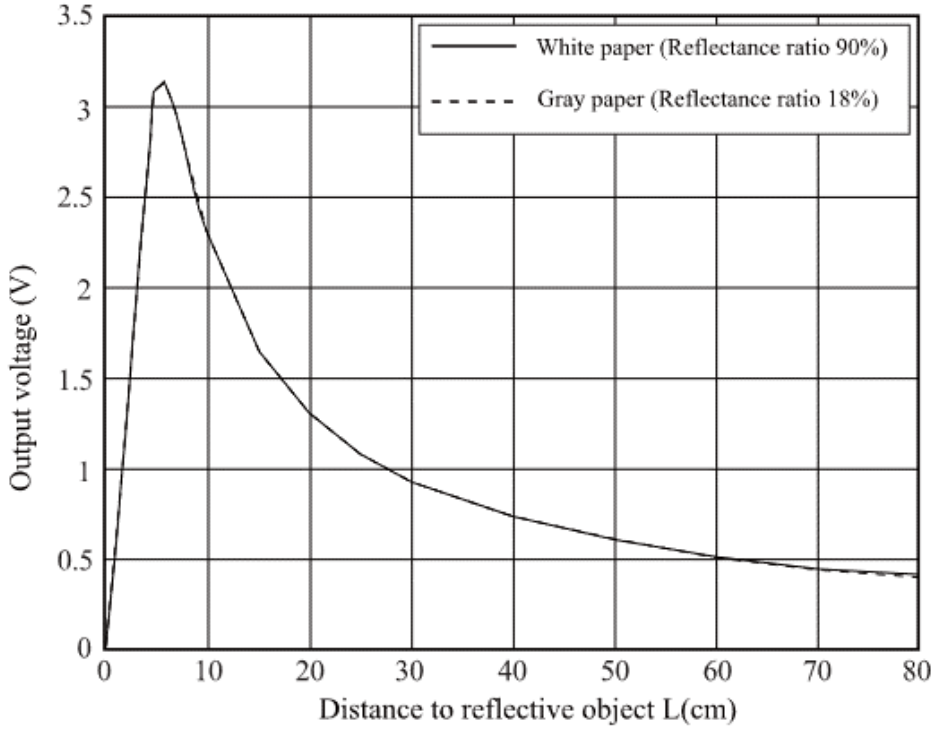
Käytetty elektroniikka rakennettiin Dundermotors GR42x25 24V -sähkömoottorien ympärille. Moottoreita ohjattiin EM-324C 12-24V -moottorinohjauskortilla, joilla pystyttiin säätämään sähkömoottorien pyörintänopeutta ja -suuntaa. Kortti tarjosi hyvän määrän erilisiä ominaisuuksia moottoreiden ohjaukseen, kuten:

- suoja mahdolliselta ylijännitteeltä,
- fault-ongelma signaalin ohjelmointia varten,
- moottorin pyörintäsuunnan vaihdon,
- kiihdytyksen ja hidastuksen nopeuden säädön,
- analogisen pyörintänopeuden säädön.

Ohjauskortti liitettiin maalausrobotissa käytettyyn 24V-lithiumakkuun, mikä sijaitsi robotin etukeulassa. Moottorienohjauskortteja hallinnointiin Raspberry Pi 4 B -nimisellä pienoistietokoneella, josta tulevat ulostulosignaalit olivat 3.3V. Ongelmana ilmeni, että ohjauskortin signaalien piti olla vähintään 4V, jotta kortti pystyi aktivoimaan halutut portit. Tähän käytettiin kahdensuuntaista logiikkatason jännitteennostajaa, joiden toiminta perustui neljään BSS138-kanavatransistoriin, jotka pystyivät nostamaan ohjejännitteen avulla Rasberry Pi 4 B:stä tulevan 3.3V signaalin 5 volttiin.

Maalausmekanismin nostamiseen käytettiin Hitec:n HS-8360TH-servomoottoreita, jotka toimivat jännitealueella 6–7.4V. Akusta tullut 24V jännite laskettiin 7.4V:ksi käyttämällä Elecrow PCH2596M - DC/DC (engl. Direct Current) jännitemuunninta, joka antoi maksimissaan 3A-sähkövirran. Servomoottoreita voitiin ohjata suoraan Raspberry Pi 4 B:stä tulevalla 3.3V PWM-signaalilla (engl. pulse-width modulation).

Robottiin lisättiin myös GP2Y0A21YK0F-matka-antureita, joilla pystytiin tunnistamaan robotin lähiympäristöä. Kuitenkin ilmeni, että antureiden aktivointi alue 10–80 cm ei antanut tarpeeksi korkeaa jännitettä vasta, kuin ulkoinen kappale oli 20 sentin päässä robotista. Kuvassa 11 kuvaaja kertoo anturista uloslähtevän jännitteen kappaleen ja anturin väliseen matkaan verrattuna.



Kuva 11. Antureista uloslähtevä jännite matkaan verrattuna (Sharp).

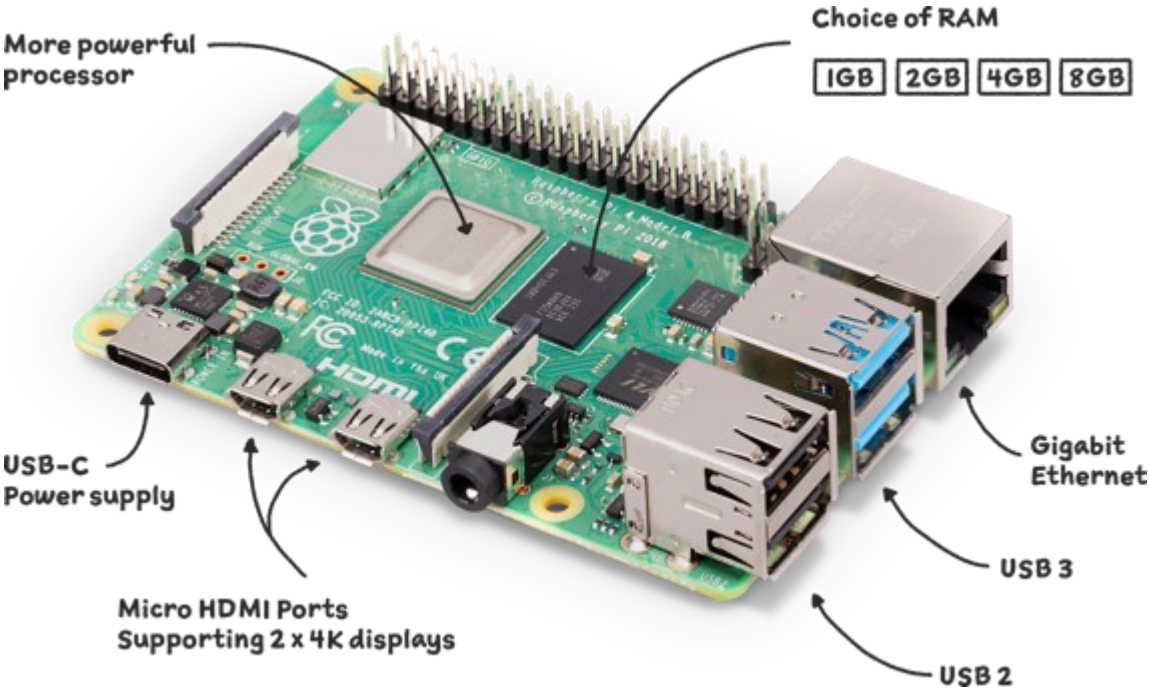
Tehdyissä testeissä ilmeni, että Raspberry Pi 4 B pystyy tunnistamaan anturista tulevan signaalin vasta, kun sitä lähestyvä kappale on noin 15 cm päässä. Antureille robotin kehikossa tarkoitetulla paikalla ihmisen ja robotin väliin jäi alle 10 cm, millä ei olisi ollut suurta vaikutusta käyttäjän turvallisuuden kanssa. Maalausrobotin maksimi etenemisnopeus rajoitettiin ohjelmallisesti sellaiseksi, että se ei pystynyt vahingoittamaan ihmistä törmäystilanteessa.

3.3 Raspberry Pi 4 B

Maalausrobotissa käytettiin Raspberry Pi:ä (RPI), joka on luottokortin kokoinen pienoistietokone. Uusin versio Raspberry Pi 4 B tulee sisään rakennetulla langattomalla internetillä ja Bluetooth-ominaisuuksilla, joita käytettiin muiden laitteiden, kuten Playstation 4 DualShock-ohjaimen yhdistämisessä. RPI:stä löytyvät liitännät ovat:

- Micro-SD-korttipaikka käyttöjärjestelmän lataamiseen ja tietojen tallentamista varten,
- neljä USB-porttia,
- kaksi micro-HDMI-porttia,
- 40-nastainen GPIO (engl. General Purpose Input/Output) yleisten IO-yhteyksien tekemiseen.

RPI antaa laajan valikoiman myös ohjelmakoodin kehitykseen sen avoimen tietokoneohjelmiston takia. Samankaltaiset pienoistietokoneet ovat hyvin suosittuja harrastelijoiden ja prototyyppivalmistajien parissa, mitkä antavat kehittäjille vapaat kädet työstää haluamiansa projekteja.



Kuva 12. Raspberry Pi 4 Model B.

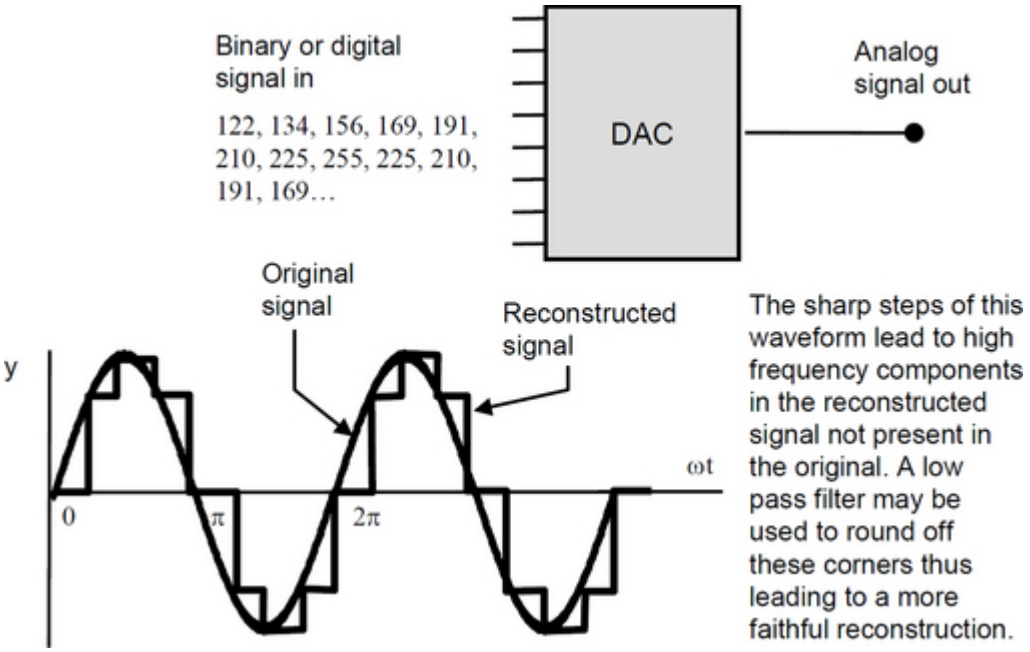
RPI:n mukana tullutta Thonny Python IDE-nimistä ohjelmaa käytettiin projektin ohjelmointiympäristönä. IDE (engl. Integrated development environment) tarkoittaa ohjelmointiympäristöä, missä käyttäjä pystyy rakentaa ja kehittää omaa ohjelmistokoodiansa. Se myös antaa käyttäjän ajaa luotua ohjelmakoodia ilman erillistä komentokehotetta, joka on käyttöjärjestelmien oma ohjelmakoodin suorittaja. Myös ohjelmointivirheiden etsiminen on osana IDE:tä, jolla voidaan optimoida luodun ohjelmakoodin suorituskykyä.

Maalausrobotissa käytetyn ohjelman ohjelmointikielenä toimi monipuolinen mutta yksinkertainen Python. Sen hyvänä puolena on järjestelmäriippumattomuus, eli kirjoitettua ohjelmaa voidaan ajaa useammalla kuin yhdellä käyttöjärjestelmällä. Python tarjoaa hyvin kokonaisen sisäisen kirjaston, mihin voidaan helposti asentaa Internetistä haluttuja ominaisuuksia ja integroida ne ohjelmaan. Koska Python on laajasti käytetty ohjelmointikielin, niin sille myös löytyy suuri ulkopuolinen kirjastotuki. Vaikka Python onkin luotu olio-ohjelmointikieleksi, sillä voidaan kuitenkin myös ohjelmoida menetelmällisesti. (Summerfield 2010: 1–3.)

3.4 Moottorien ohjausten toteutus

Maalausrobotissa käytettyjen sähkömoottorien ohjaus tehtiin EM-324 DC-moottorin ohjauskortilla. Jotta ohjauskorttia pystyttiin käyttää, piti RPI:stä tuleva digitaalinen signaali kääntää analogiseksi. Tähän käytettiin digitaalianalogista muunninta, mikä pystyi simuloimaan analogista signaalia. Kuva 13 näyttää, kuinka askelmaiset digitaaliset signaalit ovat, kun taas analoginen signaali on enemmänkin sinifunktion kuvaajan.

Digitaalinen signaali voidaan kääntää analogiseksi käyttäen suodattajia, millä käännös tapahtuu nanosekunneissa. Koska digitaalinen tieto on askelapproksimaatio sisään tulevasta signaalista, sen ulos lähtevä analogisen signaalin kuvaaja muistuttaa askelmaista luonnetta. (Fischer-Cripps 2002.)



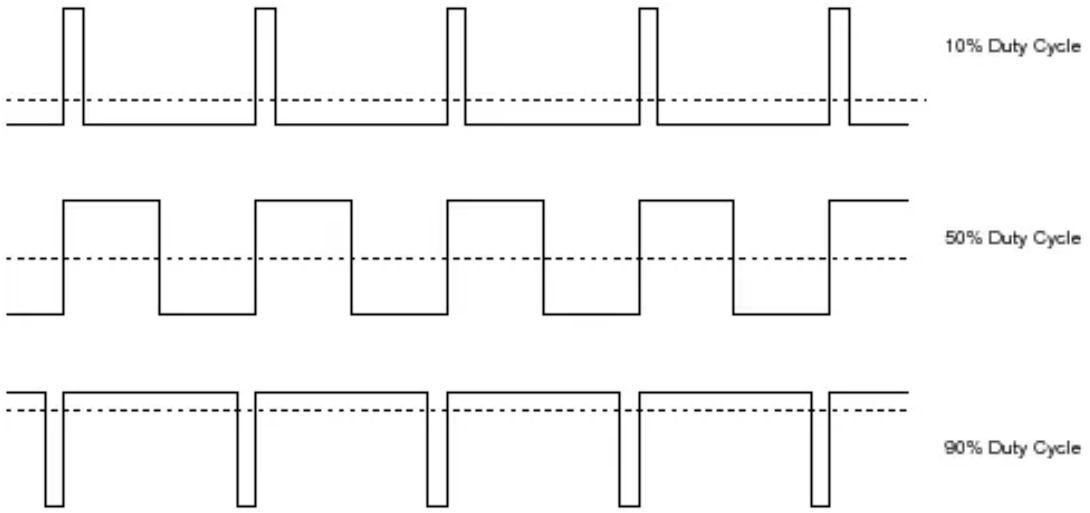
Kuva 13. Digitaalisen signaalin muuntaminen analogisignaaliksi (Fischer-Cripps 2002).

PWM eli pulssinleveysmodulaatio on tekniikka, millä rajoitetaan syötetyn virran määrää vastaanottavalle elektroniselle laitteelle. Virtaa säädetään katkaisemalla virta ja kytkemällä se takaisin nopealla aikavälillä. Tätä voidaan tehdä jopa kymmeniätuhansia kertoja sekunnissa. On- ja off-jaksojen väliin jäävä suhde määrää lopullisen sähkövirran määrän, mikä laitteeseen päästetään. Tätä kutsutaan pulssisuhteeksi. Saatu pulssisuhde ilmoitetaan prosentuaalisena lukuna, esimerkiksi 50 % tarkoittaa, että puolet maksimi virtasta pääsee laitteelle. Laskettu pulssisuhde saadaan yhtälöstä:

$$D = \frac{\mathrm{t}}{\mathrm{T}} \tag{9}$$

D on pulssisuhde,
t on jaksonpituus,
T on on- ja off-jaksojen yhteenlaskettu aika.

Signaalin luomiseksi käytetään laskuria, jota kasvatetaan haluttuun lukuarvoon saakka, minkä jälkeen se nollataan ja aloitetaan alusta. (Lunkka 2011: 2–3). Kuvassa 14 havainnollistetaan, miten pulssin pituus vaikuttaa PWM-signaalin pulssisuhteeseen.



Kuva 14. Esimerkki PWM-signaalista (PWM PIC 2022).

Maalausrobotissa käytettiin RPI:stä saatavia PWM-signaaleita sähkömoottorien ohjaamiseen, mutta RPI:stä löytyi vain kaksi käytettävää PWM-signaalia. Jotta saatiin aikaan ohjaukseen tarvittavat neljä erillistä ohjaussignaalia, joita moottorit ymmärtäisivät, jouduttiin myös käyttämään RPI:stä lähtevää digitaalista signaalia. Tämä digitaalinen signaali oli kuitenkin muutettava analogiseksi signaaliksi, mikä tehtiin PRI:hin lisätyllä ADCDAC Pi Zero pHAT (pi Hardware Attached on Top) pinottavalla ja juovuttamattomalla lisäosalla. Tämä yhdistetiin RPI:stä löytyvään 40 pinniseen GPIO-liittimeen. pHAT lisäsi RPI:lle kaksi 12-bittistä digitaalianalogista signaalin muuttajaa, jotka muuttivat RPI:stä tulevan digitaalisensignaalin analogiseksi ohjesignaaliksi, mitä sähkömoottorien ohjauskortit käyttivät sähkömoottorien virran säätöön.

3.4.1 Sähkömoottorien ohjaus

Playstation DualShock 4 -ohjainta käytettiin maalausrobotin ohjaamiseen, joista vasen sauvaohjain hoiti robotin kääntymisen ja oikeaa alankytkintä käytettiin kiihdytykseen. Ohjaimen sauvaohjaimista saatiin minimiarvo 337 ja maksimiarvo 32767. Näistä luvuista maksimi haluttiin jakaa minimi-luvulla, mikä antoi desimaaliluvun nollasta yhteen, mikä kerrottiin ADCDACPi:n kirjastosta löytyvän vahvistusluvun kanssa (Kaava 10).

$$x = (\frac{a}{32767} * 4095) \quad (10)$$

x on saatu ohjausjännitearvo
a on ohjaimesta saatu ohjearvo
4095 on kirjastosta löytyvä vahvistusluku
32676 on ohjaimesta saatu maksimiarvo

Saatu luku syötettiin sitten pHAT:lle, joka antoi lasketun luvun mukaan ulosläh-tevän jännitteen 0–3.3V väliltä. Digitaalinen signaali muutettiin pHAT:ssa ole-valla MCP4822 mikrosirulla analogiseksi ja nostettiin 0-5V käyttäen logiikkata-sonmuunninta. Ohjausjännite vietiin sähkömoottoria ohjaavaan ohjauskorttiin, joka hallinnoi moottoriin menevää sähkövirtausta.

Moottorin ohjauskortin nostaessa moottorille menevää sähkövirtaa johti sähkö-moottorin pyörintänopeuden kasvamiseen. Kuitenkin lopullinen moottorin pyö-rintänopeus säätyy kuormituksen mukaan, joten kuormituksen kasvaessa pyö-rintänopeus laskee ja kuormituksen laskiessa pyörintänopeus kasvaa (Kajula 2013: 9–10). Koska maalausrobotin moottoreille kohdistuvalla kuormituksella ei ollut suuria vaihteluita, robotin vakionopeus kasvoi moottoreille menevän sähkö-virran mukaisesti. Tällä voitiin hienosäätää robotille tienmaalaukselle sopiva etenemisnopeus.

Omni-pyöristä saatavasta suuntauksettomasta liikkeestä luovuttiin, jotta pystyt-tiin saamaan aikaan haluttu maalausjälki. Robottiin takaosaan kiinnitetyn maa-lausmekanismin käyttö ei ollut järkevää, mikäli robotin liikkeet eivät olleet en-nalta arvattavia, mikä ei ollut aina mahdollista suuntauksettomalla liikkeellä.

Sähkömoottorit jaettiin kahteen eri ryhmään: robotin edessä sijaitseva kääntymisestä huolehtiva moottori ja takana olevat moottorit hoitivat robotin eteenpäin ajon ja peruuttamisen.

3.4.2 Maalausmekanismin servomoottorien ohjaus

Maalausmekanismin maalin syötön määrää ohjattiin kahdella Hitec:n HS-8360TH-servomoottorilla, joilla nostettiin maalin vapaata valumista estävä mekanismi. Sulavan pystysuuntainen liike saatiin aikaan kääntämällä toinen servomoottoreista ylösalaisin, jotta moottorien nostoliikeradat olisivat mahdollisimman symmetriset mekanismia kohden. Tällä varmistettiin tasaisesti levitetty voima, millä puristettiin mekanismin ja maalisäiliöiden väliin jäävät tiivististeet tiukasti yhteen. Servomoottorien liike haluttiin asettaa 25 asteesta 40 asteeseen, mikä nosti mekanismia noin puoli senttiä ja antoi laitteiston tiputtaa maalia maalauspinnalle. Tarkat servomoottorien kulmat saatiin käyttämällä suoran yhtälön kaavaa:

$$y = kx + b \tag{11}$$

y on suora
k on kulmakerroin
x on muuttuja
b on vakiotermi

Servomoottorin liikettä ohjattiin for-silmukkalauseella, jossa moottorille annettiin kymmenen eri asentoarvoa 0.1 sekunnin välein. Ohjaukseen lisättiin pieni viive, koska moottorien liian rajut liikkeet olisivat aiheuttaneet säiliöiden maalin roiskumisen.

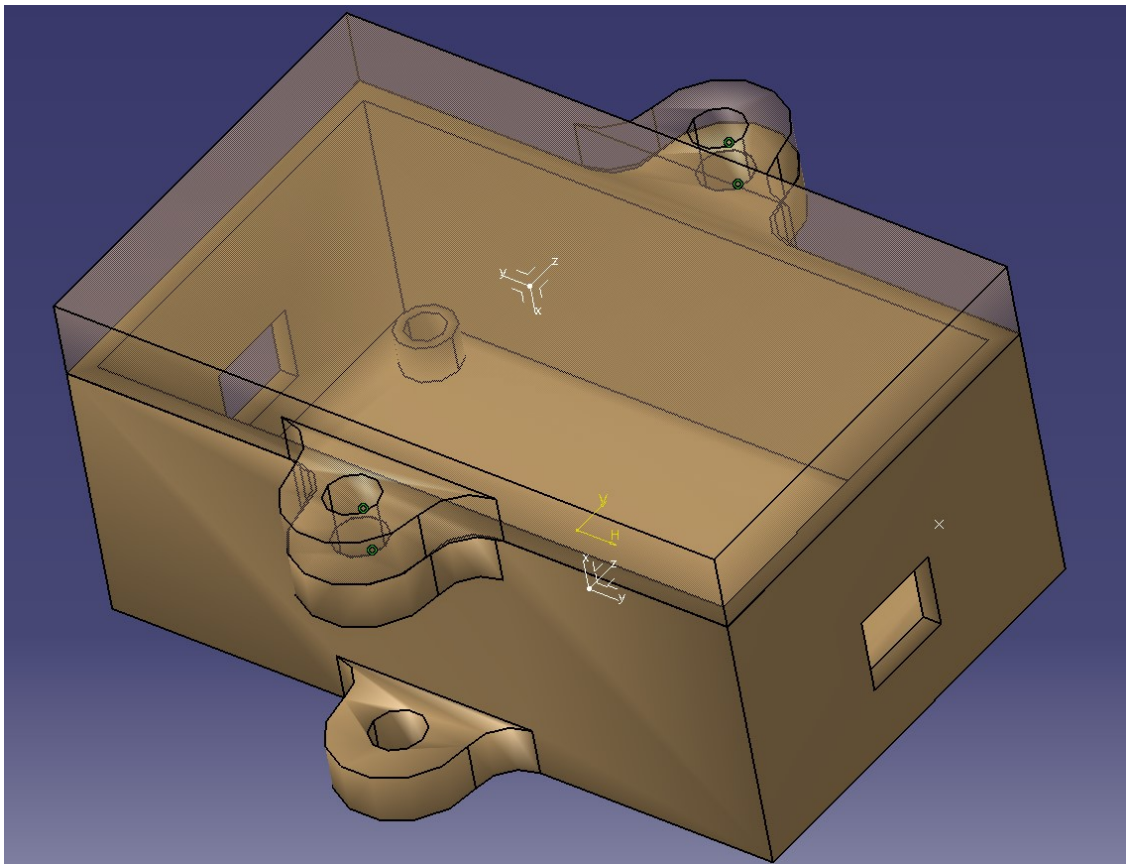
Maalausmekanismi pystyi valuttamaan maalia ainoastaan, jos käyttäjä piti kauko-ohjaimesta laadittua näppäintä manuaalisesti alhaalla, mikä nosti maalia sisällä pitävän mekanismin. Kun käyttäjä päästi irti kyseisestä nappulasta, maalin virtaus loppui, koska mekanismi laskettiin ja se tukki valumiseen tarkoitetut reiät.

Käytetyn servomoottorin liikeohjeet luotiin käyttämällä GPIO Zero -nimistä kirjastoa. Se luo RPI:lle taustaprosessin, joka tukee laitteiston oikeaa PWM-signaalia, eikä simuloitua sellaista. RPI:n omasta kirjastosta löytyvä PWM-signaali on ohjelmallisesti tehty ja se ei luo yhtä tarkkaa signaalia kuin kirjastossa käytettävän laitteistossa luotu PWM, joka myös käyttää hyväksi oikosiirtoa. Laitteistolla luotu PWM-signaali on paljon vakaampi ja tarkempi kuin simuloitu, minkä seurauksena servomoottorilta saatiin ylimääräiset epävakaat liikkeet poistettua. (Jones & Nuttall 2021.)

3.5 3D-tulostamisen käyttö elektroniikan suojaamisessa

Kosteusvaurioiden pelon takia, robotin elektroniikkaa haluttiin suojata mahdollisilta maali- tai nesteroiskeilta. Tähän käytettiin hyväksi 3D-tulostamista. 3D-tulostuksella tarkoitetaan materiaalia lisäävää teknologiaa (Additive Manufacturing, AM), missä materiaalia lisätään tulostukseen ennalta määriteltyjen liikeratojen avulla kerros kerrokselta. Liikeradat luodaan käyttäen kolmiulotteista tietokoneella avustettua mallia (3D CAD). Kappaleen laatu määräytyy kerrosten paksuudesta. Mitä ohuemmat kerrokset, sitä lähempänä tulostettu kappale on alkuperäistä mallinnusta. (Brent ym. 2015: 1–3.)

Elektroniikan suojausten mallinnus toteutettiin CAD-tiedoston tekemisellä CATIA V-5 -sovelluksella (Kuva 15.). Mallinnettu kappale muutettiin STL-formaattiin (Standard Tessellation Language), minkä data luodaan laskemalla erikokoisia kolmioita 3D-mallin pinnalle. Luotujen kolmioiden kärjet ja vektorit yhdistetään ja lopuksi muutetaan koordinaatioiksi, joita voidaan käyttää hyväksi liikeratojen luomisessa. STL-tiedostossa ei säilytetä erikseen koordinaatistojen välisiä matkoja millimetreissä tai tuumissa, vaan 3D-tulostimen ohjelman pitää muuttaa matkat haluttuun muotoon. (Brent ym. 2015: 352–353.)

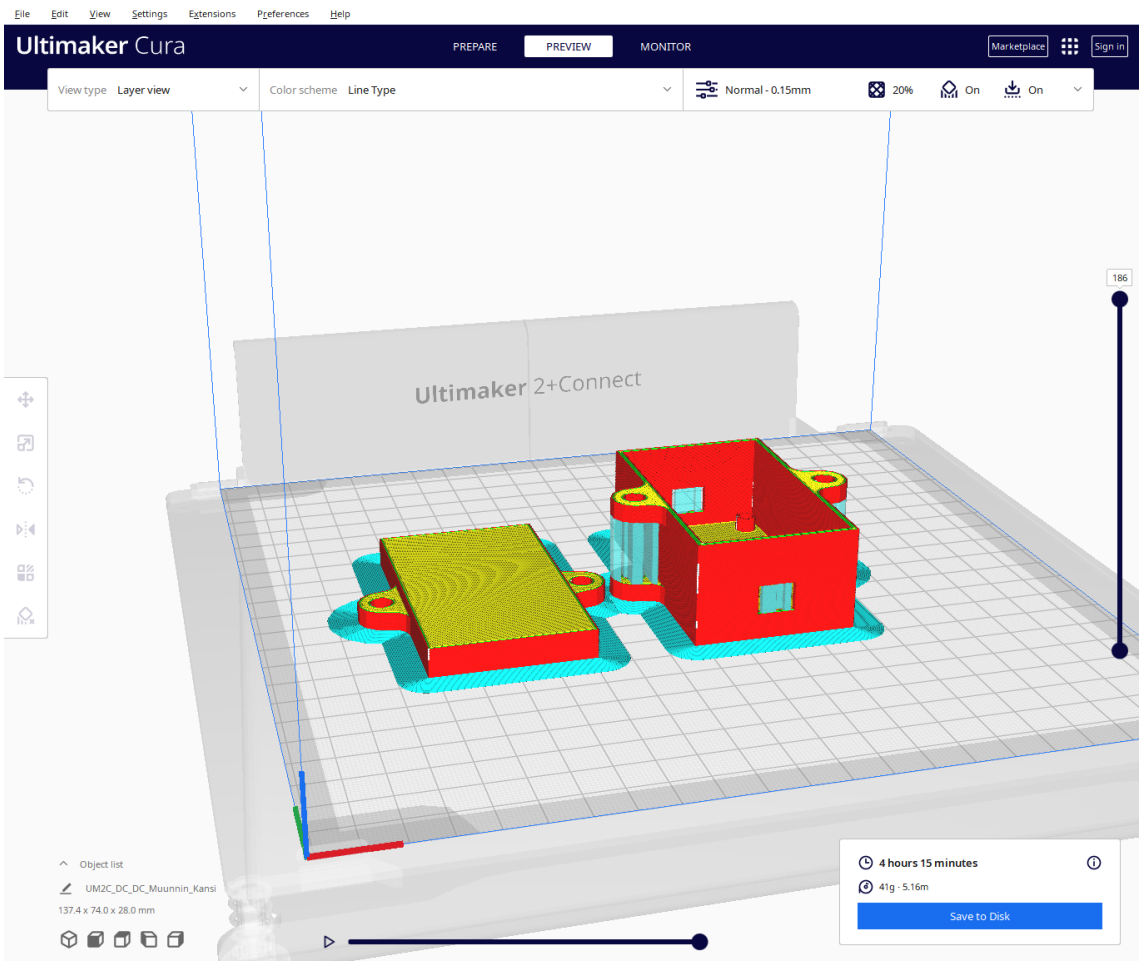


Kuva 15. STL-muotoon muunnettu CAD-malli.

Kappaleiden tulostamiseen käytettiin ammattilaiskäyttöön tarkoitettua Ultimaker 2+ Connect -nimistä 3D-tulostinta. Tulostin käyttää omaa ohjelmistoa nimeltä Ultimaker Cura (Kuva 15.), jolla voidaan säätää tulostuksen ominaisuuksia kuten:

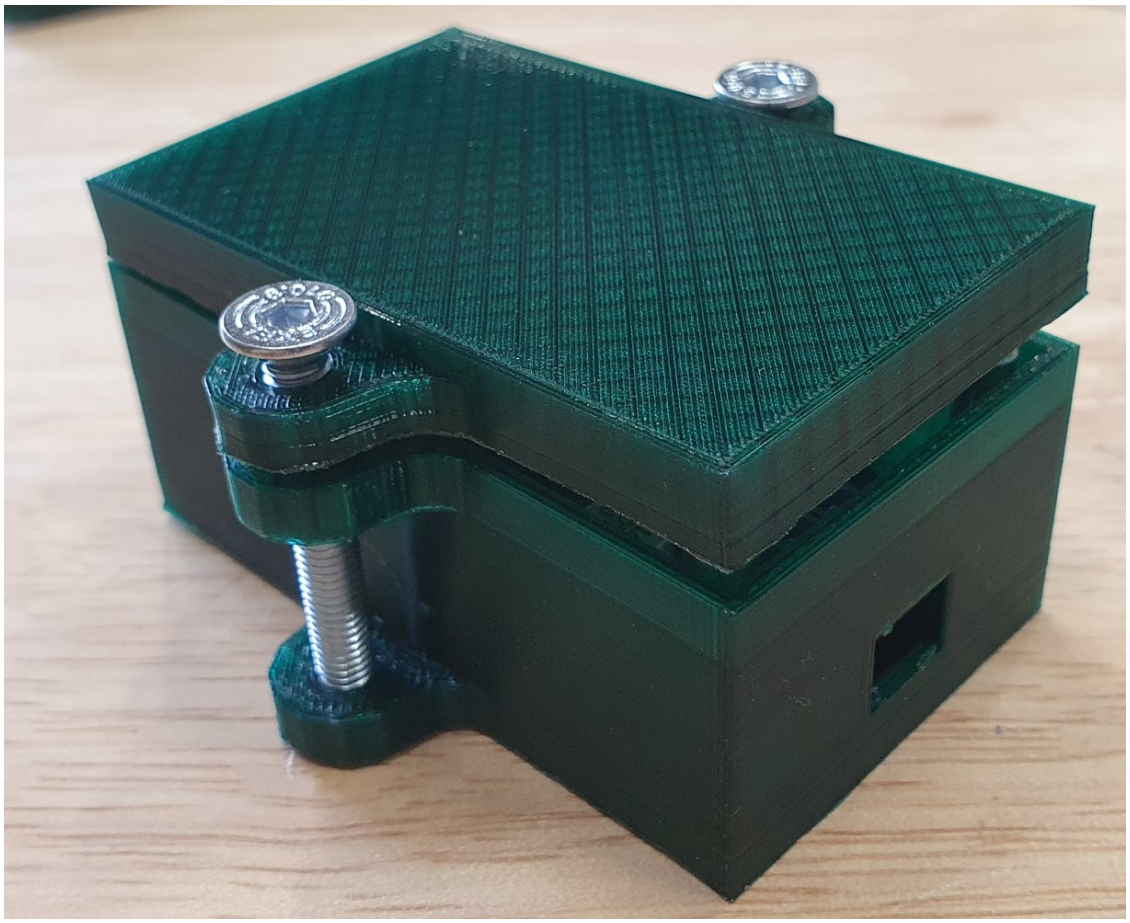
- kerrosten paksuutta,
- materiaalin määrän käyttöä,
- tulostuksen tiheyttä,
- tulostuksen nopeutta.

Aiemmin luotu STL-tiedosto avattiin 3D-tulostimen ohjelmalla ja kappale asetettiin simuloidun tulostimen työstöalueelle.



Kuva 16. 3D-tulostuksen liikeradat Ultimaker Cura -ohjelmassa.

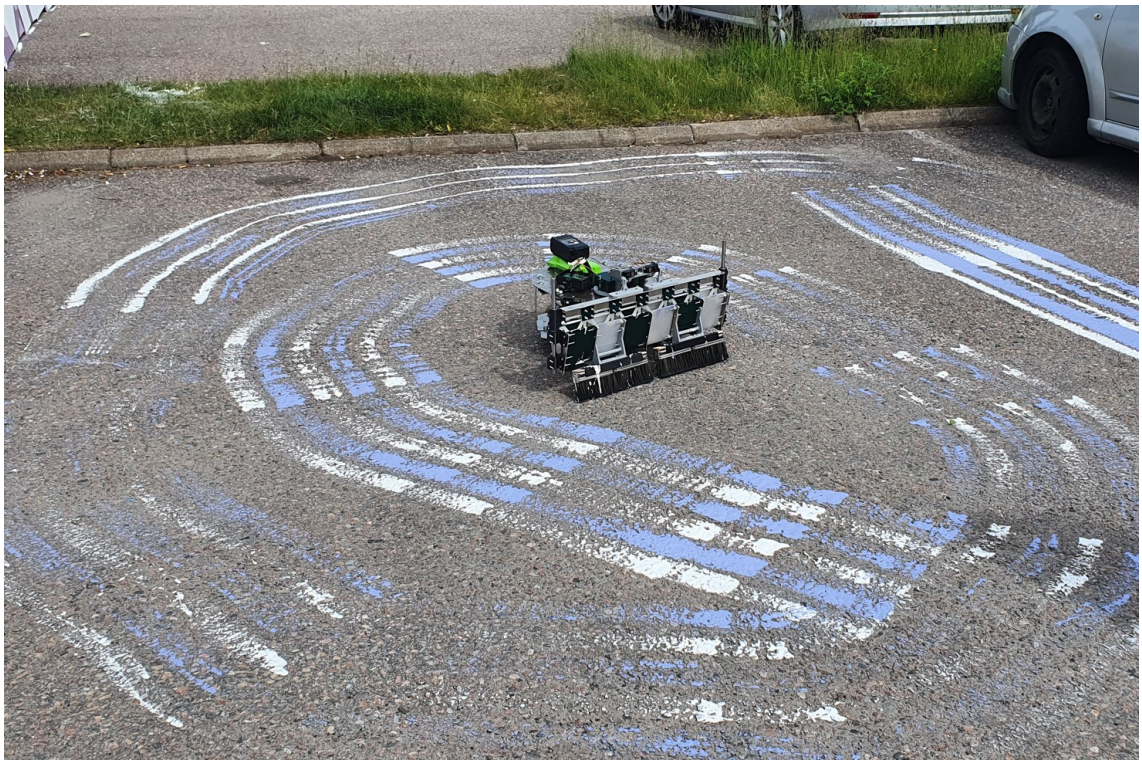
Kuvassa 16 sinisellä olevat tukialustat luotiin kappaleelle tulostamisen lopputuloksen parantamiseksi. Seuraavaksi tulostuksien kerrosten paksuudeksi valittiin 0.15 mm. Kappaleen täytöksi laitettiin 20 prosenttia materiaalia, mikä tarkoittaa, että kappale koostumus on 80 prosenttisesti ilmaa ja 20 prosenttisesti käytettyä materiaalia. Kappaleen materiaaliksi valittiin vihreän värinen BASF:n Ultrafuse PLA, joka on helposti tulostuva ja kestää pientä pintahiontaa. Jokaiselle elektroniikkaosalle tehtiin omat suojalaatikot.



Kuva 17. Valmis tulostettu kappale.

4 Maalausrobotin käyttökokemukset

Maalausrobotin ensimmäinen käyttökoe suoritettiin Street Art Vantaan parkkipaikalla. Kokeessa käytiin läpi robotin ohjauksen ja maalausmekanismin käyttö. Paikan päällä ilmeni, että signaalijohdot olivat irronneet RPI:stä kuljetuksen aikana. Kun robotti saatiin takaisin liikkeelle, sillä tehtiin kuiva-ajo, eli robottia ajettiin parkkipaikalla edestakaisin ja maalausmekanismin nostoa testattiin. Kun robotin käyttäytymiseen oltiin tyytyväisiä, maalausmekanismin kanisterit täytettiin Teknopark 3085 -maalilla. Kuvassa 18 näkyy testeistä tehty maalausjälki. Kokeessa huomattiin, että robotti valutti liian nopeasti maalia, mistä seurasi sen lyhyt aktiivinen käyttöaika. Servomoottorien liikerataa pienennettiin, jotta maalia pääsisi vähemmän ulos kanistereista.



Kuva 18. Maalausrobotin ensimmäinen testikäyttö.

Koetestin lyhyyden takia akkujen todellista kestoa ei pystytty testaamaan. Todelliseen käyttöön kuitenkin varattiin vara-akku, mikäli yksi ei olisi riittänyt koko tapahtuman ajaksi. Kokeen jälkeen maalausmekanismi pestiin, jotta maali ei pinttyisi kanisterien seinillä ja tukkisi valutusreikiä.

4.1 Käyttökokemukset Vantaan nuortentapahtumissa

Vantaan kaupunki järjesti yhteistyössä Street Art Vantaa kollektiivin kanssa kesällä 2022 nuorille tarkoitettuja työpajoja, joiden teemana oli Pride-viikko. Pajoja pidettiin seuraavissa kohteissa:

- Myyrmäen urheilupuistossa
- Tikkurilan kirkon edessä
- Havukosken nuorisotalolla.

Työpajat aloitettiin tutustumalla nuoret maalausrobottiin, jolla maalattaisiin ennalta päätettyjä asfalttiteitä, jonka jälkeen heille opetettiin robotin ohjaus. Maalausrobottia ajettiin aluksi ilman maalia, jotta osallistujat tottuisivat sen käyttöön ja maalausjälki olisi paremman laatuinen. Nuoret oppivat robotin ohjausjärjestelmän hyvin nopeasti, koska se suunniteltiin samankaltaiseksi kuin ohjaimella pelattavat ajovideopelit. Osallistujat saivat maalata robotilla, kunnes maali loppui kanistereista, minkä jälkeen ne täytettiin uudelleen. Ohjaajan vaihto sujui ilman ongelmia ja nuoret opettivat toisiaan ajamaan maalausrobottia, mikäli minkäänlaista epävarmuutta ohjauksesta ilmeni. Kuvassa 19 nähdään robotin tekemää jälkeä.

Kuva 19. Myyrmäen urheilupuiston työpajalla aikaan saatu jälki.

Työpajat kestivät noin kaksi tuntia, minkä aikana robotti oli keskimäärin käytössä 70 prosenttia ajasta. Maalausrobotin akku pystyi pitämään robotin käyttökelpoisena noin 45 minuuttia, mikäli käytön aikana ei ilmennyt taukoja. Litiumakun tyhjentyessä jännitevaraus tippui alle 23V, minkä takia robotin ja ohjaimen välinen kommunikointi heikkeni. Tämä johti siihen, että viimeinen lähetetty komento robotille jäi voimaan ja robotti ei vastannut ohjaimen lähettämiin komentoihin. Tyhjentynyt akku irrotettiin nopeasti robotista ja uusi vaihdettiin tilalle ja robotti saatiin uudelleen käynnistyksen jälkeen takaisin käyttökuntoon. RPI:ssa käytetyt signaalijohtojenliitäntä heikkeni huonolaatuisen asfaltin aiheuttaman värähtelyn takia. Pajapäivien aikana olleet poikkeavat 30 asteen hellepäivät myös vähensivät robotin käyttöaikaa, minkä takia RPI ylikuumeni auringonvalossa kahden työpajan lopuksi.

Kuva 20. Maalausrobotti käytössä.

Ongelmista huolimatta maalausrobotilla saatiin maalattua koko varattu alue ja kaikki haluavat osallistujat pääsivät käyttämään robottia. Vaikka maalausmekanismin täyttö tehtiin käsin, ylimääräisistä roiskeita säästyttiin. Pajoilla käytetyt värit pidettiin robotin lähettyvillä olevassa kärryssä, jotta uudelleen täyttö olisi nopeaa.

4.2 Käytössä ilmenneiden ongelmien kehittäminen

Työpajoilla ilmenneistä ongelmista haluttiin päästä eroon, jos maalausrobottia tultaisiin käyttämään uudelleen tulevaisuudessa. Maalausrobotille haluttiin tehdä uusi pinnoitus, koska kehikossa käytetty teräs alkoi ruostua ja maalinsyöttöjärjestelmänohjausta haluttiin päivittää. Signaalijohtojen liitosten löystyminen haluttiin myös ratkaista, jotta robotti ei lopettaisi ohjauskomentojen seuraamista kesken ajon. Myös akkujen lautauksessa ilmentynyt ylijännite haluttiin korjata.

Maalausrobotti purettiin kokonaan ja teräsosat eriteltiin hiekkapuhallusta varten. Hiekkapuhalluksessa käytettiin keskihienoa hiekkaa, millä irrotettiin teräkseen syntynyt ruoste. Seuraavaksi teräsosat käsiteltiin teräkselle sopivalla spray-pinnoitusaineella ja annettiin kuivua, kunnes kappaleet voitiin pinnoittaa toisella kerroksella halutun paksumman suojakerroksen takia.

Robottia uudelleen kootessa käytetyt matka-anturit poistettiin, koska niistä saatava turva-alue ei suojannut käyttäjää tai muita mahdolliselta törmäykseltä käytön aikana. Robotin huippunopeus oli sen verran pieni, ettei se itsestään voinut aiheuttaa vahinkoa ihmisille. Myös virransiirrossa käytettyjä johtoja siistittiin antureiden poiston takia.

Työpajoissa maalin asfaltille syöttö tapahtui manuaalisena näppäimen painalluksena. Käyttäjän luoma satunnainen maalin käyttö haluttiin muuttaa ennalta arvattavaksi muuttamalla ohjauskomentoa. Jotta mekanismi saatiin aina valuttamaan tietyn verran maalia, ohjauskomentoa muutettiin automaattiseksi liikkeeksi, mikä tapahtuu käyttäjän painaessa ohjausnäppäintä. Komento nostaa maalausmekanismin 0.1 sekunnin ajaksi ja laskee sen tasaisesti takaisin paikalleen tukkimaan maalin virtausta. Ongelmaksi osoittautui, ettei ohjelma osannut suorittaa useampaa erillistä komentoa samanaikaisesti, joten liikkeen toteutus tehtiin muuttamalla käytetty ohjelmakoodi säkeiseksi.

Säkeisyydellä voidaan suorittaa useampaa osaa ohjelmassa samanaikaisesti. Yleisesti Python-ohjelmointikielessä komennot tapahtuvat toinen toisensa jälkeen, eikä samanaikaisesti, vaikka se siltä voisikin vaikuttaa. Mikäli ohjelmassa on useampi säe, tietokoneen prosessori ajaa säkeet järjestyksessä, eikä kaikkia samanaikaisesti. (Anderson 2021.)

Koska maalausrobotti tarvitsee vain yhden ylimääräisen säkeen, voidaan haluttua ohjelmakoodia ja ylimääräistä silmukkaa ajaa samanaikaisesti luomalla yksi säe, minne kyseinen silmukka on ohjelmoitu. Silmukka suoritetaan aina, kun siihen yhdistettyä nappulaa painetaan pohjaan. Silmukan pyöriessä, robottia voitiin ohjata muilla ohjekomennoilla, mikä mahdollistaa robotin samanaikaisen ohjauksen ja automaattisen maalinsyötön käytön.

5 Yhteenveto

Insinöörityössä suunniteltiin ja toteutettiin onnistuneesti käyttäjäystävällinen maalausrobotti. Työllä haluttiin kartoittaa yhteistyörobotiikan hyödyntämistä urbaanikatutaitteessa ja Vantaan nuorisotapahtumissa. Tutkimuskohteina olivat robotin fyysisten rakenteiden suunnittelu, elektroniikan valinta ja käyttöönotto, laitteen ohjelmointiohjelman luominen ja ohjauksen ratkaisut. Työn aikana ilmeni, että robotiikan ja taiteen yhteistyöt ovat vielä hyvin alkuvaiheessa ja kehitysmahdollisuuksia löytyy useammalta alueelta.

Lähestymistapana oli käyttäjäystävällisyys, joka näkyy robotin ohjauksessa ja myös sen käytössä. Maalausrobotille tarvitaan vain antaa virtaa ja se itsenäisesti osaa käynnistää ohjelmistonsa, minkä jälkeen käyttäjän tarvitsee vain yhdistää kauko-ohjaimensa robottiin yhdistysnappulaa painamalla. Sitten voidaan käynnistää maalausmekanismi ja kaataa halutut maalit säiliöihinsä, minkä jälkeen maalausrobotti on käyttövalmis.

Projektin toteutusta suunnitellessa Big-Flash -hankkeen henkilökunta osoittautui arvokkaaksi avuksi. Ongelmien ilmetessä aina löytyi henkilö, jolta pystyi kysymään asioista tai sitten mietittiin asiaa yhdessä. Maalausmekanismin pääosin suunnitteli ja toteutti projektiin myöhemmin liittynyt projektiassistentti. Hänen kanssaan pystyttiin puntaroimaan useampaa ideaa robotin kokonaisuudesta ja valitsemaan projektiin sopivat toteutukset.

Kaikkia työpajoissa ilmenneitä kehitysideoita ei kyetty toteuttamaan opinnäytetyön aikana. Akkujen lataamisesta ilmenneistä ylijännitteistä ei päästy eroon. Mahdolliset ratkaisut olisivat olleet akkujen vaihto, jännitteen tarkastelu ja latauksen aktiivinen hallinnointi. Ylimääräistä jännitettä yritettiin alentaa asentamalla virransiirtoon diodi, joka olisi vähentänyt ylimääräistä jännitettä, mutta robotin käyttämän virran määrä ilmeni liian suureksi ja se paloi.

Maalausrobotin muotoilua voitaisiin kehittää, jotta se muistuttaisi enemmän massatuotettua laitetta kuin prototyyppiä. Veden kestävyyttä voitaisiin parantaa

luomalla robotille ulkokuoret ja piilottamalla elektroniikka kuorien sisälle suojaan. Mikäli halutaan parantaa käyttäjien ja lähellä olevien turvallisuutta, voitaisiin robottiin asentaa sensorit, joilla tunnistettaisiin lähellä olevat ihmiset. Myös maalin automaattista täyttöä voitaisiin alkaa kehittämään, jotta pidempi käyttökokemus parantuisi.

Insinöörityö osoitti, että katutaiteen ja robotiikan yhteistyö on toteutettavissa. Rakennetulla robotilla saatiin aikaan katutaidetta Vantaan kaduille nuorten voimin. Työssä luotu maalausrobotti on hyvä alku taiteessa käytettävään yhteistyörobottiin ja tilaajan toiveisiin päästiin.

Lähteet

ADCDAC Pi with Python. 2015. Verkkoaineisto. AB Electronics UK. <https://www.abelectronics.co.uk/kb/article/15/adcdac-pi-with-python>. Luettu 29.9.2022.

Anderson Jim. 2021. An Intro to Threading in Python. Artikkeli. < https://realpython.com/intro-to-python-threading/>. Luettu 10.10.2022.

Bemis Steven. Design And Development of a Novel Omni-Directional Platform. 2007. Opinnäytetyö. University of Ontario Institute of Technology. <https://www.collectionscanada.gc.ca/obj/thesescanada/vol1/OOSHDU/TC-OOSHDU-29.pdf>.

Brent Stucker, David Rosen, Gibson Ian. 2015. Additive Manufacturing Technologies 3D printing, Rapid Prototyping, and Direct Digital Manufacturing. 2nd ed. Springer New York Heidelberg Dordrecht London. ISBN 978-1-4939-2112-6.

Costa J. Paulo, Moreira Paulo A., Oliveira P. Hélder & Sousa J. Armando. Dynamical Models For Omni-directional Robots With 3 And 4 Wheels. 2008. Verkkoaineisto. <https://www.researchgate.net/publication/256089847_Dynamical_Models_for_Omni-directional_Robots_with_3_and_4_Wheels>. Luettu 19.10.2022.

Estremera Joaquin, Garcia Elena, Gonzalez de Santos Pablo. Quadrupedal Locomotion An Introduction to the Control of Four-legged Robots. 2006. Springer-Verlag London Limited. e-ISBN 1-84628-307-8.

Fischer-Cripps A. C.. 2002. Newnes Interfacing Companion. Linacre House, Jordan Hill, Oxford OX2 8DP. ISBN 0-750-65720-0

Greenspan Steven, Matthews Peter. Automation and Collaborative Robotics. 2020. Springer Science+Business Media New York, 1New York Plaza, New York, NY 10004. e-ISBN 978-1-4842-5964-1.

Herath Damith, Kroos Christian, Editors Stelarc. Robots and Art. 2016. Springer Nature. ISBN 978-981-10-0321-9.

Jones Dave, Nuttall Ben. 2021. API - Output Devices. Verkkoaineisto. <https://gpiozero.readthedocs.io/en/stable/api_output.html?highlight=servo>. Luettu 10.10.2022.

Jones Dave, Nuttall Ben. 2021. API - Pins. Verkkoaineisto. <https://gpiozero.readthedocs.io/en/stable/api_pins.html?highlight=factory#module-gpiozero.pins.pigpio>. Luettu 10.10.2022.

Kajula Miika. 2013. Sähkötyökoneen moottorinohjausjärjestelmän suunnittelu ja toteutus. Opinnäytetyö. Oulun seudun ammattikorkeakoulu. Theseus-tietokanta.

Lunkka Kalle. 2011. Pulssinleveysmodulaation käyttö moottoriohjauksessa. Opinnäytetyö. Lahden Ammattikorkeakoulu. Theseus-tietokanta.

Oriolo Giuseppe, Sciavicco Lorenzo, Siciliano Bruno, Villani Luigi. Robotics Modelling, Planning and Control. 2010. Springer-Verlag London Limited. e-ISBN 978-1-84628-642-1.

Pearson Chris. 2012. High Speed, Digital to Analog Converters Basics. Verkkoaineisto. <https://www.ti.com/lit/an/slaa523a/slaa523a.pdf?ts=1664458914848&ref_url=https%253A%252F%252Fwww.google.com%252F>. Luettu 29.9.2022.

PWM PIC: Pulse Width Modulation. 2011. PWM for the PIC Microcontroller (or any microcontroller). Verkkoaineisto. <http://www.best-microcontroller-projects.com/pwm-pic.html>. Luettu 29.9.2022.

Satish Kumar Peddapelli. 2017. Pulse Width Modulation: Analysis and Performance in Multilevel Inverters. Walter de Gruyter GmbH, Berlin/Boston. ISBN 978-3-11-046817-5.

Servomoottorin ohjaus. 2021. Verkkoaineisto. Hutasu. <https://www.hutasu.net/mikrokontrollerit/arduino/9-servomoottorin-ohjaus/>. Luettu 10.10.2022.

Sharp. Distance Measuring Sensor Unit Measuring distance: 10 to 80 cm Analog output type. Verkkoaineisto. <https://global.sharp/products/device/lineup/data/pdf/datasheet/gp2y0a21yk_e.pdf>. Luettu 18.10.2022.

Spirin Artur. pyPS4Controller. 2016. Verkkoaineisto. <https://pypi.org/project/pyPS4Controller/>. Luettu 29.9.2022.

Summerfield Mark. 2010. Programming in Python 3: A Complete Introduction to the Python Language. 2nd ed. RR Donnelley Crawfordsville in Crawfordsville, Indiana. ISBN 978-0-321-68056-3.

Tzafestas S. G., Introduction to Mobile Robot Control. 2013. Elsevier. ISBN 978-0-12417-103-9.

Käytetty ohjelmakoodi

```
import RPi.GPIO as GPIO             #Raspberry's own library
from time import sleep
from pyPS4Controller.controller import Controller  #pyPS4Controller
library import to use ps4 dualshock controller
from gpiozero import Servo                         #GPIOzero library
for giving cleaner signal to servo motors.
from gpiozero import AngularServo
from gpiozero.pins.pigpio import PiGPIOFactory     #Changes PWM-signal
from software generated to hardware generated one.
from ADCDACPi import ADCDACPi                      #ADCDAC library for
the extra daugtherboard, which is used to control motors 1-3. Chips in
the board generate true analog signal from a digital one, in an area
of 0-3.3v
import threading

global pause                 #Variable for controlling the pausing of
motors
pause = 0                    # 0 = off, 1 = on
global park                  #Variable to check if the robot is moving
or not (stops paint from stacking under it)
park = 0                     # 0 = off, 1 = on
global adjust_up             #Used to manually adjust the degree of
movement of the servomotors.
adjust_up = 0
global adjust_down           #Used to manually adjust the degree of
movement of the servomotors.
adjust_down = 0
global stop_threads          #Stops For-loop used to move Servomotors
stop_threads = True

#------ GPIO ADDRESSES -----#
adcdac1 = ADCDACPi(2) # Gain = 2 -> area is 0...3.3 V  MOTOR 3, which
is used for rotational movement.
adcdac2 = ADCDACPi(2) # Gain = 2 -> area is 0...3.3 V  MOTOR 1 & 2,
which are used for backward and fortward movement.

adcdac1.set_dac_raw(1, 0)    #Turns off motor 3 by giving a value of
0v
adcdac2.set_dac_raw(2, 0)    #Turns off motor 1 & 2 by giving a value
of 0v

pause_GPIO = 11                  #GPIO 11 pin for stopping motors
duty=0                         # Global variable to change the value
of PWM (Used to run and stop motors)
FWD2 = 15                    #GPIO 16 pin for reversing the rotational
direction of motor3 (making it turn the other way)
FWD1 = 16                    #GPIO 15 pin for reversing the rotational
direction of motors 1 & 2 (making them turn the other way)

#----- servo -----#          #NOTE PiGPIOFactory uses GPIO.BCM not
GPIO.BOARD to refer pins. (Which is why there are 2x 13 pins)
factory = PiGPIOFactory()   #Selects factory as a PiGPIOFactory(),
which is the gpiozero library's way of converting a GPIO-pin from
software pwm to hardware one.
servo1 = AngularServo(13, min_pulse_width=0.00078,
max_pulse_width=0.0021, pin_factory=factory) # Tested pulse width for
```

```
Servomotors, They aren't exact but are good enough to give around 130*
degree movement. 7.8mm formax and 210mm for min positions.
servo1.value = None;         #removes any value that may be going out
to servo motors.

servo2 = AngularServo(12, min_pulse_width=0.00078,
max_pulse_width=0.0021, pin_factory=factory) # Tested pulse width for
Servomotors, They aren't exact but are good enough to give around 130*
degree movement. 7.8mm formax and 210mm for min positions.
servo2.value = None;         #removes any value that may be going out
to servo motors.

#------ GPIO SETUPS ------#
GPIO.setwarnings(False)           #disable warnings
GPIO.setmode(GPIO.BOARD)          #set pin numbering system

GPIO.setup(FWD1,GPIO.OUT)         #set FWD1 for motor12 GPIO pin active
and as output
GPIO.setup(FWD2,GPIO.OUT)         #set FWD2 for motor3 GPIO pin active
and as output

GPIO.output(FWD1,GPIO.LOW)       #Set FWD 1 and 2 low at the start of
the program
GPIO.output(FWD2,GPIO.LOW)

GPIO.setup(pause_GPIO,GPIO.OUT)       #set pause_GPIO for all the mo-
tors GPIO pin active and as output
GPIO.output(pause_GPIO,GPIO.LOW)
#-----------------------------Functions--------------------------------
---#

def stop():
    GPIO.output(FWD1,GPIO.LOW)   #Writes GPIO.Board pin 13 (FWD1) out-
put as low gate ( 0 )
    GPIO.output(FWD2,GPIO.LOW)  #Writes GPIO.Board pin 16 (FWD2) out-
put as low gate ( 0 )
    adcdac1.set_dac_raw(1, 0)    #Changes voltage value to 0 ( 0% sig-
nal )
    adcdac2.set_dac_raw(2, 0)
    servo1.value = None;
    servo2.value = None;

def ServoMovement():             #Creates a up-and-down-motion for ser-
vos. Function runs in a thread
    global x
    global stop_threads
    while True:                  #if stop_threads is true -> checks the
state of stop_threads again untill it is turned False
        if stop_threads:
            sleep(0.1)
            continue
        else:                    #Gives servos posiotions from 25 de-
grees to 40 degress in 10 steps
            for x in range(1,10,1):
                servo1.angle = (-15/9)*x+(125/3)     #-40 to -25 de-
grees
                servo2.angle = (15/9)*x+(-125/3)     #40 to 25 degrees
                print("x-up: ",x,"Servo1 asento : ",servo1.an-
gle,"Servo1 asento : ",servo2.angle)
                sleep(0.1)
```

```
            for x in range(10,0,-1):
                servo1.angle = (-15/9)*x+(125/3)     #-25 to -40 de-
grees
                servo2.angle = (15/9)*x+(-125/3)     #25 to 40 degrees
                print("x-down: ",x,"Servo1 asento : ",servo1.angle,
"Servo2 asento : ",servo2.angle)
                sleep(0.1)

SmoothServo = threading.Thread(target=ServoMovement) #Create a thread
called SmoothSerco to run ServoMovement in it
SmoothServo.start()                                  #Start the thread

#----------------------------PS4 Controller--------------------------
----------#

class MyController(Controller):

    def __init__(self, **kwargs):              #pyPS4Controller li-
brary init
        Controller.__init__(self, **kwargs)

#------- Buttons -------#


    def on_options_press(self):                        #Stop-Signal to
the motors
     global pause                              #Global pause variable to
manage which functions can be ran when the value of pause is 1.
     pause += 1
     if pause == 1:                             #When options-button is
pressed, give pause- +1 in value. When pause = 1, send signal to all
motors to stop. When pause does not equal 1, turn off the pause signal
and change the value of pause back to 0.
          GPIO.output(pause_GPIO,GPIO.HIGH)
          adcdac1.set_dac_raw(1, 0)
          adcdac2.set_dac_raw(2, 0)
          servo1.value = None;
          servo2.value = None;
          print("Paused")
     else:                                    #When pause = 2, change it
back to 0 so everything can be ran again.
          GPIO.output(pause_GPIO,GPIO.LOW)
          pause = 0
          print("Unpaused")

    def on_x_press(self):
        global pause
        global stop_threads


        if pause != 1:                             #If pause is off and
park (robot is moving) is on, the bot may paint
            stop_threads = False                   #Runs For-loop used to
move Servomotors
        else:
            stop_threads = True                    #Stops For-loop used
to move Servomotors

    def on_x_release(self):                        # Puts servo back to
its' resting position.
```

```
        global stop_threads
        stop_threads = True

# ------ Joy sticks ---------#

    def on_L3_right(self, value):                           # Motor 3
(Motor for turning right)
        global pause
        global park

        if pause != 1:
            park = 1                                        #park = 1,
which means the robot is moving.
            Controller.on_L3_right(self, value)             #337 lowest,
32767 highest.
            L3_right_DACValue = int(value / 32767 * 3900) #DAC-values
are defined in 0-4095.
            adcdac1.set_dac_raw(1, L3_right_DACValue)     #rewrite new
dac value.
            print(L3_right_DACValue)
        else:
            adcdac1.set_dac_raw(1, 0)                       #if pause =
1, shut down motor and make park = 0 (robot has stopped moving)
            print("Paused ", pause)
            park = 0

    def on_L3_left(self, value):                            #Motor 3
(Motor for turning left)
        GPIO.output(FWD2, GPIO.HIGH)                        #Change the
rotational direction of motor 3
        global pause
        global park


        if pause != 1:

            park = 1                                        #park = 1,
which means the robot is moving.
            Controller.on_L3_left(self, value)              #-337 low-
est, -32767 highest.
            L3_left_DACValue = int(value / -32767 * 3900) #DAC-value
0-4095 = 0-3V
            adcdac1.set_dac_raw(1, L3_left_DACValue)      #rewrite new
dac value.
            print(L3_left_DACValue)
        else:
            adcdac1.set_dac_raw(1, 0)                       #if pause is
on, stop motor 3 and reset its' values
            GPIO.output(FWD2,GPIO.LOW)
            park = 0
            print("Paused ", pause)

    def on_L3_x_at_rest(self):                              #Resets mo-
tor 3's values when L3 is at a resting position.
        global park
        adcdac1.set_dac_raw(1, 0)
        GPIO.output(FWD2,GPIO.LOW)
        park = 0

#----- Bumpers -----#
```

```
    def on_R2_press(self,value):                                #Motor 1
& 2 foward movement with bumpers. Largely works the same as functions
before it.
        global pause
        global park

        if pause != 1:
            park = 1
            GPIO.output(FWD1,GPIO.HIGH)
            Controller.on_R2_press(self, value)                 #-32767
Lowest , 32767 Highest
            R2_Value = int( (585 / 9362) * value + (3900 / 2))# Maths
based on (y=kx*b)
            adcdac2.set_dac_raw(2, R2_Value)
            print(R2_Value)
        else:
            park = 0
            adcdac2.set_dac_raw(2, 0)
            GPIO.output(FWD1,GPIO.LOW)
            print("Paused ", pause)

    def on_R2_release(self):                                    #Resets
motors 1 & 2's values
        global park
        park = 0
        adcdac2.set_dac_raw(2, 0)
        GPIO.output(FWD1,GPIO.LOW)

    def on_L2_press(self,value):                                #Motor 1
& 2 reverse movement with bumpers.
        global pause
        global park
        if pause != 1:
            park = 1
            Controller.on_L2_press(self, value)                 #-32767
Lowest , 32767 Highest
            L2_Value = int( (585 / 9362) * value + (3900 / 2))# Maths
based on (y=kx*b)
            adcdac2.set_dac_raw(2, L2_Value)
            print(L2_Value)
            print("park = ", park)
        else:
            park = 0
            adcdac2.set_dac_raw(2, 0)
            print("Paused ", pause)

    def on_L2_release(self):                                    #Resets
motors 1 & 2's values
        global park
        park = 0
        adcdac2.set_dac_raw(2, 0)

# --------------------------------Safety-------------------------------
---------#

def disconnect():                       #IF Controller disconnects it
changes all runnable values to 0 or low gate ( 0 )
    # any code you want to run during loss of connection with the con-
troller or keyboard interrupt
```

```
    GPIO.output(FWD1,GPIO.LOW)
    GPIO.output(FWD2,GPIO.LOW)
    GPIO.output(pause_GPIO,GPIO.HIGH)
    adcdac1.set_dac_raw(1, 0)
    adcdac2.set_dac_raw(2, 0)
    GPIO.cleanup()
    print("Goodbye!")
    pass


controller = MyController(interface="/dev/input/js0", connecting_us-
ing_ds4drv=False)
# you can start listening before controller is paired, as long as you
pair it within the timeout window
controller.listen(on_disconnect=disconnect) #Listens to controller and
runs disconnect() -funtion if it is disconnected
```