Tampereen yliopisto

Santeri Puola

ROBOTTIKÄSIVARREN INTEGROINTI AUTONOMISEEN MAA-AJONEUVOON

Kandidaatintyö
Tekniikan ja luonnontieteiden tiedekunta
Huhtikuu 2020

TIIVISTELMÄ

Santeri Puola: Robottikäsivarren integrointi autonomiseen maa-ajoneuvoon
Kandidaatintyö
Tampereen yliopisto
Tekniikan kandidaatin tutkinto-ohjelma
Huhtikuu 2020

Työn tarkoituksena on toteuttaa robottikäsivarrelle ohjaus ja liittää käsivarsi itseohjautuvaan ajoneuvoon. Liittämällä robottikäsivarsi ajoneuvoon saadaan ajoneuvolle useampia käyttötarkoituksia. Ajoneuvo ei enää vain kulje paikasta toiseen, vaan se voi myös esimerkiksi siirrellä kappaleita käsivarrellaan.

Tässä työssä toteutetaan toimiva ohjaus robottikäsivarteen, joka liitetään osaksi ajoneuvoa. Ohjauksen valinnassa pohditaan robottikäden valmistajan tarjoaman valmiin työkalun ja itse kirjoitetun ohjauskoodin välillä. Valmiin työkalun etuja ovat sen käyttövarmuus ja luotettavuus. Lisäksi se on erittäin helppo saada toimimaan. Kirjoittamalla oman ohjauksen robotille saan ohjauksesta muokattavamman. Päädyin toteuttamaan oman ohjauksen, jolloin robottia pystyy ohjaamaan samalla ohjaimella kuin ajoneuvoakin. Se ei olisi ollut mahdollista valmiilla työkalulla. Ohjauksessa hyödynnetään robotinohjausjärjestelmää (*engl.* Robot Operating System, ROS), jota myös ajoneuvon toiminnot käyttävät. Tämä helpottaa robottikäsivarren liittämisessä ajoneuvoon.

Vaikka työ painottuu ohjauksen toteuttamiselle, tarkastellaan työssä myös robottikäsivarren liittämistä ajoneuvoon. Käsivartta ei vielä kiinnitetä ajoneuvoon, vaikka liitäminen onkin lopulta hyvin yksinkertaista. Työssä käydään kuitenkin läpi liittämisessä huomioitavat asiat. Kun robottikäsivarsi korvaa ajoneuvossa tällä hetkellä olevan kameran, hankaloittaa se käytännön ohjaamista. Käyttäjä ei pysty enää seuraamaan ajoneuvon ja robotin liikkeitä. Tarvittaessa ajoneuvoon voisi kiinnittää pienemmän kameran, jolla ongelma ratkaistaisiin.

Avainsanat: robotin ohjaus, ROS, WidowX, Summit-XL

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla.

SISÄLLYSLUETTELO

1. JOHDANTO 1
2. TEOREETTISTA TAUSTAA JÄRJESTELMISTÄ 2
2.1 WidowX Robot Arm 2
2.2 Inverse kinematics 4
2.3 Robotnik SUMMIT-XL 5
2.4 Robot Operating System 6
3. KÄYTETTÄVÄT MENETELMÄT 7
3.1 Vaatimukset robottikäsivarren ohjaukselle 7
3.2 Vaihtoehdot robottikäsivarren ohjaamiseen 7
3.2.1 Vaihtoehto 1: RVIZ 8
3.2.2 Vaihtoehto 2: Toisen mallin ohjauspaketti 9
3.2.3 Vaihtoehto 3: Sarjaportti 9
3.3 Työn vaiheet 10
4. ROBOTTIKÄDEN OHJAAMINEN JA LIITTÄMINEN AJONEUVOON 11
4.1 Robottikäsivarren ohjauksen valmistelut 11
4.2 Ohjauksen toteuttaminen 13
4.3 Robottikäsivarren liittäminen ajoneuvoon 16
5. PÄÄTELMÄT 18
LÄHTEET 19
LIITE A: ROBOTIN OHJAUKSESSA KÄYTETTY OHJELMAKOODI 21

LYHENTEET JA TERMIT

GPS	engl. Global Positioning System, paikannusjärjestelmä
IK	engl. Inverse kinematics, käänteinen kinematiikka
PID	engl. Proportional-Integral-Derivative, servomoottorin säädin
ROS	engl. Robot Operating System, robotinohjausjärjestelmä
RVIZ	engl. ROS visualization, 3D-visualisointityökalu
Arbotix	Arbotix-ohjauslevyn Arduino-koodikirjasto
Arduino	Arbotix-M-ohjauslevyn käyttämä ohjelmointikieli
ArmLink	Trossen Robotics -robottien käyttämä ohjausohjelmisto
Cartesian	suorakulmainen koordinaattijärjestelmä
Compliance	servomoottorin ohjaustyyppi
Cylindrical	sylinterikoordinaatisto
Middleware	väliohjelmisto, joka huolehtii ohjelmien välisestä tiedonsiirrosta
Node	prosessi, joka ohjaa robotin toimintoja
pySerial	Python-kirjasto, joka tarjoaa tiedonsiirron sarjaportin kautta
Python	ohjelmointikieli
rospy	Python-kirjasto, jolla saa yhteyden ROS-järjestelmään
Unix	avoimen lähdekoodin käyttöjärjestelmä

1. JOHDANTO

Robottien sovelluksia hyödynnetään yhä useammissa arkipäivissä asioissa. Robotit ovat levinneet teollisuudesta, sairaanhoidosta ja viranomaisilta aina yksityisten ihmisten koteihin. Nykyään monelta kuluttajalta löytyy jo robottipölynimuri tai -ruohonleikkuri omasta taloudestaan. Robotit hoitavat asioita, joita ihminen ei pysty tai halua tehdä.

Monet autonvalmistajat kilpailevat siitä, kuka saa ensimmäisenä oman itseohjautuvan autonsa kuluttajamarkkinoille. Anturiteknologiat ja autojen välinen viestintä kehittyy jatkuvasti tämän kilpailun tuloksena. Itseohjautuvuus helpottaa ajoneuvon liikuttamista, kun ihmisen ei tarvitse määrätä jokaista liikettä.

Tässä työssä tehtävänäni on liittää robottikäsivarsi osaksi itseohjautuvaa autonomista ajoneuvoa ja saada käsivarsi toimimaan ajoneuvon ohjausjärjestelmän kanssa. Tarkoituksena on saada ohjattua käsivartta PlayStation Dualshock4 -ohjaimella Robot Operating System -robotinohjausjärjestelmän kautta. ROS-ympäristö mahdollistaa ohjaamisen lähiverkon kautta etänä, ja PlayStation-ohjaimen käyttö helpottaa käsivarren ohjaamisesta. Kun ajoneuvoon liitetään robottikäsivarsi, saadaan ajoneuvolle monipuolisempia ominaisuuksia ja käyttötarkoituksia. Ajoneuvolle voidaan esimerkiksi asettaa päämäärä ja sen saavutettuaan robottikädellä lastata tai siirtää kappaleita.

Työni koostuu johdannosta, teoriaosuudesta, työn menetelmien erittelystä ja päätelmistä. Toisessa luvussa perehdytään työssä käytettävien järjestelmien teoriataustaan. Kolmannessa luvussa esitellään työssä käytettävät menetelmät, eli miten työ toteutetaan. Neljännessä luvussa kuvataan työn yksityiskohdat ja miten käsivarsi liitetään ajoneuvoon. Lopussa on päätelmiä työstä ja siitä selvinneistä asioista.

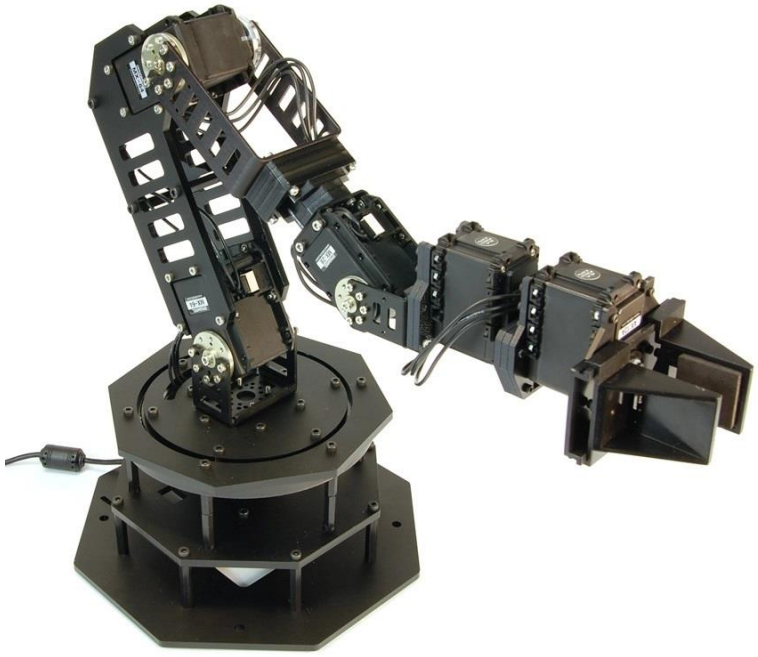
2. TEOREETTISTA TAUSTAA JÄRJESTELMISTÄ

Työn tarkoituksena on saada kaksi järjestelmää toimimaan yhdessä. Käytettävät järjestelmät ovat robottikäsivarsi ja ajoneuvo, johon käsivarsi liitetään. Ajoneuvoon on valmiiksi toteutettu ohjaus, mutta robottikäsivarren ohjaaminen täytyy toteuttaa itse. Ajoneuvo tarjoaa valmiina myös ROS-robotinohjausjärjestelmän.

Tässä luvussa tarkastellaan käytettävien järjestelmien teoreettista taustaa. Luvussa tutustutaan robottikäsivarren ja ajoneuvon ominaisuuksiin ja ohjaamiseen. Lisäksi perehdytään ROS-ympäristöön ja sen suomiin mahdollisuuksiin.

2.1 WidowX Robot Arm

Työssä käytetään Trossen Roboticsin valmistamaa WidowX Robot Arm Mark II -robottikäsivartta (kuva 1). WidowX-mallin robotissa on kaikkiaan kuusi servomoottoria, joilla säädetään robotin pyörivää alustaa, kolmea akselia sekä tarttujaa. Käsivarrella on siis kaikkiaan viisi vapausastetta, mikä mahdollistaa robotin monipuolisen liikuttamisen 41 cm ulottumalla. (WidowX Robot Arm Kit 2020)



Kuva 1: *WidowX-robottikäsivarsi (WidowX Robot Arm Kit 2020).*

Valmistajalla on WidowX-robotin lisäksi useita muita robottikäsimalleja. Malleissa on eroja muun muassa servomoottoreissa, vapausasteissa ja ulottumilla. Taulukossa 1 on esitetty työssä käytetyn robottikäsivarren lisäksi muutaman muun Trossen Roboticsin valmistaman robottikäsivarren ominaisuuksia havainnollistamaan mallien välisiä eroja. Kantokyvyllä tarkoitetaan tässä tapauksessa maksimikuormaa, jonka robotti pystyy kannattelemaan ääriasennossa. Osassa malleista valmistajan tarjoama ROS-paketti on kattavampi kuin työssä käytettävässä WidowX-robotissa. Kattavampi paketti tarjoaa valmiina esimerkiksi tuen PlayStation-ohjaimelle.

Taulukko 1: *Trossen Roboticsin robottikäsivarsien vertailu (PhantomX Reactor Robot Arm Kit 2020; PincherX 150 Robot Arm 2020; ViperX 300 Robot Arm 6DOF 2020; WidowX Robot Arm Kit 2020).*

Malli	Kantokyky (g)	Vapaus-asteet	Ulottuma (cm)	Servomoottorien määrä
WidowX	400	5	41	6
PincherX 150	50	5	45	8
ViperX 300 6DOF	750	6	82	9
PhantomX Reactor	150	5	43	7

WidowX-robottikäsivarren liikkeet luodaan DYNAMIXEL-servomoottoreilla. Robotissa on kolmenlaisia servomoottorimalleja: MX-64, MX-28 ja AX-12A. Jokaista näitä servomoottorimallia on kaksi kappaletta kyseisessä robotissa. Eri servomoottorien tärkeimmät ominaisuudet on esitetty taulukossa 2. Resoluutiolla tarkoitetaan pienintä kulmaa, jonka servomoottori pystyy kääntymään. MX-64- ja MX28-servomoottorit ovat siis liikkeiltään AX-12A-servomoottoria tarkempia, koska niissä on pienempi resoluutio. MX-sarjan servomoottorit käyttävät PID-säädintä (*engl.* Proportional-Integral-Derivative) ja magneettista asentoanturia, kun AX-12A-servossa on Compliance-säädin ja potentiometri. Compliance-ohjaus säätää servoa asennon lisäksi myös siihen vaikuttavan voiman perusteella. Jokaiselta servolta on mahdollisuus saada tieto sen asennosta, nopeudesta, lämpötilasta, kuormasta ja syöttöjännitteestä. (DYNAMIXEL MX-64T 2020) Nämä tiedot saadaan välitettyä tietokoneelle ja syntynyttä tietoa pystytään hyödyntämään servomoottorien ohjaamisessa.

Taulukko 2: *WidowX-robottikäsivarren servomoottorien ominaisuuksia (WidowX Robot Arm Kit 2020).*

Malli	Toimintakulma (°)	Resoluutio (°)	Ohjaus	Asentoanturi
MX-64	360	0,088	PID	Magneettinen
MX-28	360	0,088	PID	Magneettinen
AX-12A	300	0,29	Compliance	Potentiometri

WidowX-robotissa on Interbotixin Arbotix-M-ohjauslevy. Se on robotin ydin, joka ohjaa robotin servomoottoreita. Valmistajan verkkosivuilta on saatavilla ohjauslevylle asennettava Arduino-työkalu. Arduinolla voidaan syöttää ohjauslevylle ohjelmakoodi, joka määrittää servomoottorien tarkat liikkeet.

2.2 Inverse kinematics

Inverse kinematics lyhennetään usein IK, ja sen voisi suomentaa käänteiseksi kinematiikaksi. IK on matemaattinen menetelmä, jossa nivelten asennot määrittyvät tarkasteltavan pisteen koordinaattien mukaan. Robotin tapauksessa niveliä taivutetaan servomoottoreilla ja tarkasteltava piste on tartuntakouran sijainti. Inverse kinematicsin vastakohta on forward kinematics, jossa jokaista niveltä taivutetaan erikseen saavuttaakseen lopullisen asennon. IK on robottien ohjauksessa hyvin yleistä. On helpompaa syöttää robotille tartuntakouran sijainti kuin ohjata yhtä servomoottoria kerrallaan.

Inverse kinematicsin toiminta perustuu matemaattisiin kaavoihin, jotka laskevat jokaiselle servomoottorille oikean asennon. Kaavat ovat robottikohtaisia, sillä roboteilla ei eri mitat ja eri määrä niveliä. Koska robotilla on monta niveltä, ovat kaavat usein monimutkaisia. Yleensä robotin valmistaja tarjoaa kuitenkin valmiina ohjaustyökaluja, joissa IK on jo toteutettu.

Tässä työssä yksi tehtävistäni on toteuttaa robottikäsivarrelle ohjaus. Tavoitteena on saada robottikäsivartta ohjattua PlayStation Dualshock4 -ohjaimella. Käsivarren valmistajan tarjoamat ohjaustyökalut eivät suoraan mahdollista tätä. WidowX-robotin valmistajan työkaluissa on kuitenkin IK jo valmiina, joten voin käyttää niitä oman ohjaukseni pohjana.

2.3 Robotnik SUMMIT-XL

Työssä käytetty ajoneuvo on Robotnikin kehittämä SUMMIT-XL (kuva 2). Ajoneuvossa on sisäänrakennettu tietokone, joka ohjaa ajoneuvon toimintoja. Tietokoneeseen on kytketty reititin, jonka avulla ajoneuvoon saadaan yhteys myös etänä ROS-järjestelmän kautta.

Kuva 2: *Työssä käytetty ajoneuvo ennen robottikäsivarren liittämistä.*

SUMMIT-XL on autonominen ajoneuvo eli se pystyy myös kulkemaan itseohjautuvasti. Siihen on kiinnitettynä GPS-paikannin (*engl.* Global Positioning System), joka syöttää tietokoneelle ajoneuvon sijaintitietoa. Ajoneuvolle on mahdollista antaa karttapalvelussa reitti, jonka se kulkee itsenäisesti. GPS-paikantimen lisäksi ajoneuvoon on kiinnitetty antureita, jotka havainnoivat ajoneuvon etäisyyttä ympäristön esteisiin törmäyksen varalta. Ajoneuvoa voidaan myös ohjata ilman paikannusominaisuutta PlayStation Dualshock4 -ohjaimella. Tällöin ohjaimen analogisilla sauvoilla pystytään ohjaamaan ajoneuvon kulkua. Ohjain on yhteydessä ajoneuvoon Bluetooth-yhteydellä, joten sen kantama on vain muutamia metrejä.

Tällä hetkellä ajoneuvon etuosaan on kytkettynä 360-kamera. Ajoneuvon käynnistyessä kamera osoittaa suoraan eteenpäin, mutta sen näkökenttää on mahdollista kääntää ohjaimen suuntanäppäimiä käyttämällä. Antureiden tavoin myös kameran lähettämä kuva

saadaan näkyviin ROS-ympäristössä. Ajoneuvon liikkeitä on siis mahdollista seurata etäyhteydellä. WidowX-robottikäsivarsi tulee kuitenkin korvaamaan tämän kameran.

2.4 Robot Operating System

Robot Operating System, eli lyhyesti ROS, on Unix-pohjalla toimiva avoimen lähdekoodin väliohjelmisto. Väliohjelmisto (*engl.* middleware) huolehtii ohjelmien välisestä tiedonsiirrosta. ROS tarjoaa muun muassa työkaluja, kirjastoja ja algoritmeja robottien ohjaamiseen. (ROS Documentation – Introduction 2020) ROS-järjestelmän tarkoituksena on luoda ohjelmistoja roboteille. Ohjelmistot pyritään suunnittelemaan siten, että ne olisivat hyvin pienillä muutoksilla käytettävissä myös muissa roboteissa. (Fernández 2015) Tällä idealla syntyy yhteisö, jossa ohjelmistoja ja koodeja pystytään jakamaan keskenään.

ROS-järjestelmän toiminta perustuu prosesseihin (*engl.* node), jotka ohjaavat robotin eri toimintoja. Robotin ohjausjärjestelmä sisältää yleensä useita prosesseja, esimerkiksi yksi prosessi ohjaa renkaiden moottoreita, toinen käsittelee anturien dataa ja kolmas sisältää graafisen esityksen järjestelmästä. Prosessit pystyvät kommunikoimaan keskenään lähettämällä ja vastaanottamalla viestejä. Viestit ovat yksinkertaisia tietorakenteita, jotka sisältävät tietyntyyppistä dataa, kuten lukuarvon, liukuluvun tai totuusarvon. (ROS Documentation – Concepts 2020)

3. KÄYTETTÄVÄT MENETELMÄT

Työssä tutkitaan, miten robottikäsivarsi liitetään osaksi itseohjautuvaa ajoneuvoa ja miten käsivarren ohjaus toteutetaan. Tässä luvussa esitellään menetelmät, joilla tätä ongelmaa aletaan ratkaisemaan. Ensin selvitetään, mitä vaatimuksia käytettävät järjestelmät asettavat työlle. Seuraavaksi eritellään vaihtoehdot, joita robotin ohjaamiseen on. Tämän jälkeen käydään läpi suoritettavan työn vaiheet.

3.1 Vaatimukset robottikäsivarren ohjaukselle

Ajoneuvo asettaa ehtoja sille, miten robottikäsivarren ohjaus toteutetaan. Esimerkiksi ajoneuvon ollessa kauko-ohjattava, tulee robottikädenkin olla kauko-ohjattava. Tämä vaatimus on oleellinen, vaikka tuntuukin itsestäänselvyydeltä. Jotta ajoneuvon ja robottikäden kauko-ohjaaminen olisi mahdollisimman luontevaa, olisi yksinkertaisinta ohjata molempia järjestelmiä samalla ohjaimella. Suotavaa siis olisi, että robottikäteen toteutettava ohjaus toimisi ajoneuvon tavoin PlayStation-ohjaimella.

Ajoneuvo käyttää toimintojensa ohjaamiseen ROS-robotinohjausjärjestelmää. Jotta robottikäsivarren saa toimimaan osana ajoneuvoa, tulee myös käsivarren ohjauksen olla yhteydessä ROS-järjestelmään. Tämä asettaa rajoituksia ohjaukselle, mutta toisaalta helpottaa kauko-ohjaamisen toteuttamista. ROS-ympäristöä käytettäessä nimittäin riittää, että käytettävät järjestelmät ovat samassa lähiverkossa. Tämä mahdollistaa robottikäsivarren ohjaamisen esimerkiksi toisella tietokoneella, joka on yhteydessä ajoneuvon tietokoneeseen ROS-järjestelmän kautta.

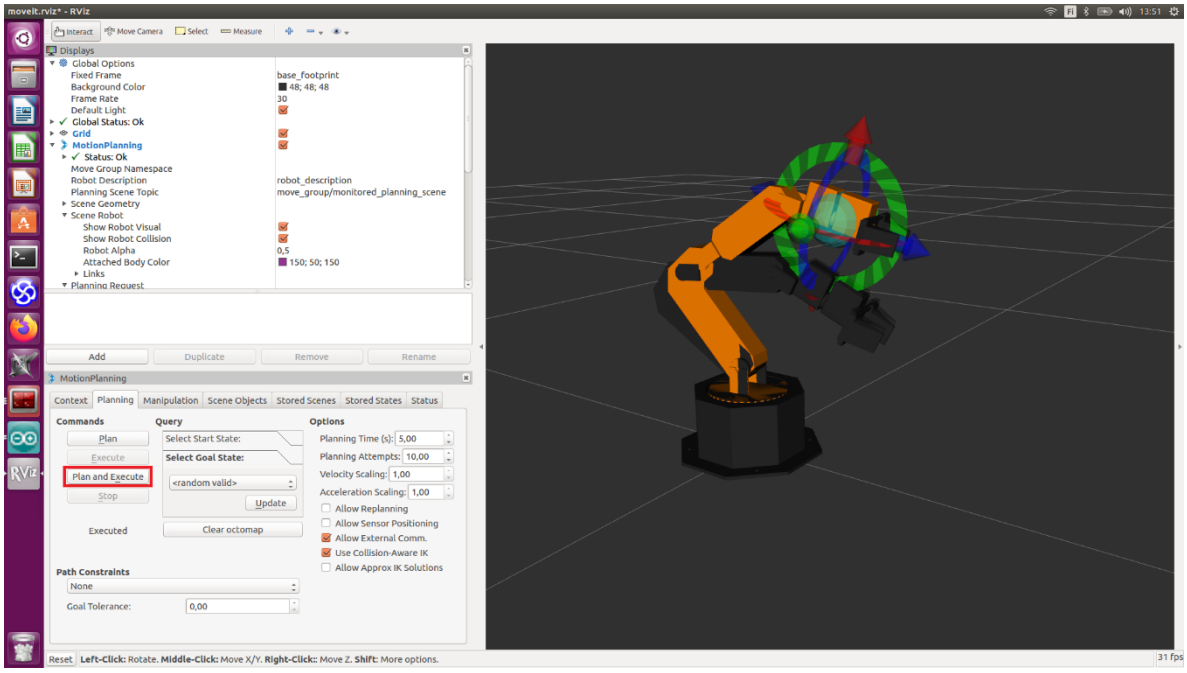
3.2 Vaihtoehdot robottikäsivarren ohjaamiseen

Edellisessä luvussa mainittujen vaatimusten perusteella robottikäsivarren tulee olla kauko-ohjattava ja yhteydessä ajoneuvon ROS-järjestelmään. Pohdin näitä ehtoja täyttäviä ratkaisuja käsivarren ohjaukselle, ja päädyin kolmeen mahdolliseen vaihtoehtoon. Ensimmäinen vaihtoehto on käyttää käsivarren valmistajan tukemaa RVIZ-työkalua (*engl.* ROS visualization), jolla robottia pystyy liikuttamaan ROS-järjestelmän välityksellä. Toinen vaihtoehto on muokata toiselle mallille tarkoitettua ohjauspakettia tälle mallille sopivaksi. Kolmas vaihtoehto on kirjoittaa oma ohjelmakoodi, joka lähettää komennot sarjaportin kautta robotille.

3.2.1 Vaihtoehto 1: RVIZ

RVIZ on 3D-visualisointityökalu, joka toimii ROS-ympäristössä ja soveltuu robotin ohjaamiseen. RVIZ vaatii tarkat tiedot robotin mitoista ja toimintasäteestä. Lisäksi työkalu tarvitsee 3D-mallin robotin osista, jotta se pystyy visualisoimaan robotin asennot. Trossen Robotics kuitenkin tarjoaa robottiensa ohjaamiseen valmiin paketin, joka hyödyntää RVIZ-työkalua ja näin ollen sisältää mallikohtaiset tiedot (ROS packages to work with the WidowX Arm 2020). RVIZ-työkalua käytettäessä tarvitsee robottikäsivarren ohjauslevylle ladata Arduino-koodi, joka sisältää inverse kinematics -kaavat ja huolehtii tiedonsiirrosta robotin ja ROS-ympäristön välillä.

Kuvassa 3 on kuvakaappaus WidowX-robotista RVIZ-työkalussa. Työkalun käyttönäkymässä on 3D-malli robotin senhetkisestä asennosta (kuvassa tummalla) ja asennosta, johon robotin halutaan liikkuvan (kuvassa keltaisella). Haluttu asento määritellään liikuttamalla hiirellä tarttujan kohdalla olevaa pistettä. Painamalla ”Plan and Execute” -nappia (kuvassa punaisella) robotti liikkuu tavoiteasentoon. RVIZ-työkaluun on mahdollista liittää myös erillinen ohjain, jolla tarttujaa voi liikuttaa tietokonehiiren sijaan.



Kuva 3: *Kuvakaappaus RVIZ-työkalusta.*

Huono puoli tässä vaihtoehdossa on ohjaamiseen tarvittava erillinen tietokone, jolla RVIZ-työkalua käytetään. Tietokone olisi ROS-järjestelmän kautta yhteydessä ajoneuvon tietokoneeseen ja robottikäteen, joten ohjaaminen olisi edelleen langatonta. Ohjaaminen ei kuitenkaan toiminut PlayStation-ohjaimella vaan ohjaamiseen piti käyttää erillistä langallista ohjainta. Tämä vaihtoehto siis toimisi käden ohjaamisessa, mutta ei olisi kovin käytännöllinen.

3.2.2 Vaihtoehto 2: Toisen mallin ohjauspaketti

Robottikäsivarren ohjaamiseen voisi hyödyntää myös saman valmistajan toiselle mallille tarkoitettua ohjausta. Trossen Robotics tarjoaa lähes kaikille muille robottikäsimalleilleen ROS-paketin, jossa on WidowX-mallin pakettiin verrattuna kattavammin ominaisuuksia (InterbotiX X-Series Arm & Turret ROS Packages 2020). ROS-paketti sisältäisi RVIZ-työkalun lisäksi muun muassa valmiina tuen PlayStation Dualshock4 -ohjaimelle. Työssä käytettävälle vanhemmalle WidowX-mallille kattavampaa ROS-pakettia ei kuitenkaan ole saatavilla.

Yksi vaihtoehto robottikäsivarren ohjauksen toteuttamiselle olisi tutustua kattavamman ROS-paketin ohjelmakoodiin ja muokata se WidowX-mallille sopivaksi. Tämä kuitenkin vaatisi suureen ohjelmakokonaisuuteen perehtymistä ja sen toiminnan sisäistämistä. Muokatun ohjelmakoodin toimimisesta ei myöskään ole mitään varmuutta.

3.2.3 Vaihtoehto 3: Sarjaportti

Robottikäsivarren ohjaukselle on mahdollista myös kirjoittaa oma ohjelmakoodi, joka muokkaa PlayStation-ohjaimen antamat arvot ja lähettää ne komentona eteenpäin robottikädelle. Ohjauksessa hyödynnettäisiin ohjelmakoodia ”ArmLink”-nimisestä ohjelmasta, jolla pystyy RVIZ-työkalun tavoin ohjaamaan robottikäsivartta (Arm Link Reference 2020). ArmLink tarjoaa valmiina komennot, joita robottikäsivarsi ymmärtää.

Tässä toteutuksessa PlayStation-ohjain on yhteydessä ajoneuvon tietokoneeseen, jonka kautta ohjaimen arvot lähetetään ROS-järjestelmään. Kirjoittamani ohjelmakoodi lukee ohjaimen arvot ROS-järjestelmästä ja määrittää robottikäden tarttujan uudet koordinaatit arvojen perusteella. Tämän jälkeen ohjelmakoodi muuttaa koordinaatin robottikäden ymmärtämäksi komennoksi ja lähettää sen sarjaportin kautta robotille.

Tämän vaihtoehdon hyvänä puolena on ohjelmakoodin muokattavuus. Pystyn itse määrittämään, miten robottia ohjataan. Toisaalta huono puoli toteutuksessa on, että osan ohjauksesta joutuu kirjoittamaan käsin. Kuitenkin ArmLink-ohjelmakoodin puolella tapahtuu osa ohjauksesta, kuten inverese kinematics -laskelmat.

3.3 Työn vaiheet

Tässä luvussa esitellään lyhyesti työn vaiheet. Työ alkaa valmistelulla, jossa valitaan käytettävä toteutus robottikäsivarren ohjaamiselle. Valmistelua seuraa ohjauksen toteuttaminen. Kun käsivarren ohjaus on valmis, tarvitsee se enää liittää ajoneuvoon.

Valmisteluvaiheessa kokeillaan kaikkia kolmea vaihtoehtoa, jolla ohjauksen voi toteuttaa. Tarkoitus karsia huonommat vaihtoehdot pois ja valita paras ratkaisu robottikäsivarren ohjaukselle. Samalla varmistetaan, että vaihtoehto on mahdollista tehdä toimivaksi. Näin selviää jo valmisteluvaiheessa, onko vaihtoehto liian hankala toteuttaa. Valmistelun jälkeen on selvillä, minkälaista ohjausta käsivarrelle aletaan työstämään. Toteuttamisvaiheessa ohjaus tehdään toimivaksi. Jos ohjauksen työstämisessä ilmenee ongelmia, voidaan palata valmisteluvaiheeseen ja harkita muita vaihtoehtoja uudelleen. Kun robottikäsivarren ohjaus on valmis, se liitetään ajoneuvoon. Tarvittaessa ohjaukseen voi tehdä hienosäätöä vielä liittämisen jälkeenkin.

4. ROBOTTIKÄDEN OHJAAMINEN JA LIITTÄMINEN AJONEUVOON

Tämä luku käsittelee työn suorittamista. Työn tavoitteena on saada robottikäsivarsi toimimaan PlayStation Dualshock4 -ohjaimella ja liittää käsivarsi ajoneuvoon. Työ suoritetaan kolmessa vaiheessa: valmistelu, ohjauksen toteuttaminen ja käsivarren liittäminen ajoneuvoon.

4.1 Robottikäsivarren ohjauksen valmistelut

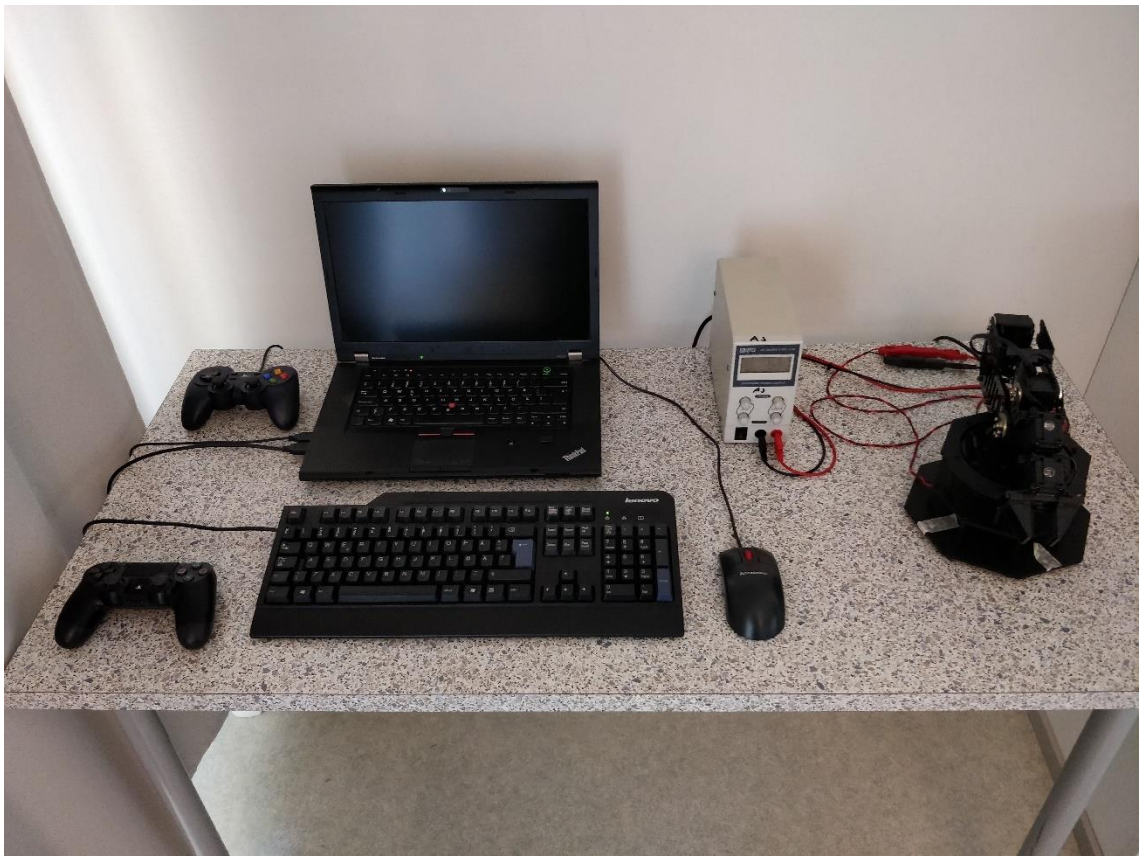
Robotin ohjaamista varten robotin ohjauslevylle pitää ladata ohjauskoodi. Työssä käytettävä WidowX-robottikäsivarsi sisältää Arbotix-M-ohjauslevyn, joka käyttää Arduino-tyyppistä ohjelmakoodia. Ohjauslevyn valmistajan sivuilta on ladattavissa Arduino-ohjelma, jolla Arduino-koodin lataaminen robotin ohjauslevylle tapahtuu (Arduino Software 2020). Arduino tarjoaa Arbotix-ohjauslevyille tarkoitetun Arbotix-kirjaston, jossa on valmiita ohjelmakoodeja eri robottimalleille ja eri käyttötarkoituksiin. Ohjelmakoodi voi sisältää muun muassa servomoottorien ohjauksen ja inverse kinematics -laskukaavat. Arbotix-kirjasto tarjoaa esimerkiksi WidowX-mallilla toimivan ”ros”-koodin, joka mahdollistaa robotin ohjaamisen ROS-järjestelmän kautta.

Valmisteluvaiheessa kokeilin kolmea vaihtoehtoa robottikäsivarren ohjauksesta ja valitsin näistä ajoneuvoon sopivimman. Ensimmäinen vaihtoehto oli käyttää valmistajan tarjoamaa RVIZ-työkalua. Valmistajan verkkosivuilta löytyi selkeät ohjeet työkalun käyttöönottoon, joten käsivarren saaminen toimintakuntoon oli helppoa. RVIZ-työkalun käyttöä varten robotin ohjauslevylle ladattiin Arbotix-kirjastosta ROS-järjestelmää käyttävä Arduino-koodi. Koodi lukee robotin ohjauskomennot suoraan ROS-järjestelmästä, jonne käyttäjä syöttää ne tässä tapauksessa RVIZ-työkalulla.

Sain RVIZ-työkalun toimimaan moitteettomasti. Robottikäsivarren ohjaaminen onnistui sekä tietokoneella 3D-mallia liikuttamalla että erillisellä ohjaimella. Tosin ohjaaminen ei toiminut haluamallani PlayStation-ohjaimella, vaan ohjaamiseen piti käyttää langallista Logitech Gamepad F310 -ohjainta. Totesin tämän vaihtoehdon kuitenkin toimivaksi, vaikka ohjaamiseen tarvitaan toinen tietokone ja toinen ohjain.

Toinen vaihtoehto ohjauksen toteuttamiseen oli käyttää toiselle robottimallille tarkoitettua ohjaustyökalua. Ideana tässä oli tutustua kattavamman ROS-paketin ohjelmakoodiin ja selvittää, mitä vaatisi saada se toimimaan WidowX-robotilla. Huomasin kuitenkin nopeasti paketin olevan hyvin suuri ja monimutkainen, joten näin parhaaksi hylätä tämän vaihtoehdon. Paketin muokkaaminen olisi vaatinut monia muutoksia ja paljon aikaa, enkä voinut olla varma ohjauksen toimimisesta.

Kolmas vaihtoehto oli kirjoittaa Python-ohjelmointikielellä oma ohjelma. Python-ohjelma lukee ROS-järjestelmästä PlayStation-ohjaimen arvot, muuttaa arvot ohjauskomennoksi ja lähetetään komennon sarjaportin kautta robotille. Jotta komennot sai välitettyä robottikädelle, piti robotin ohjauslevylle ladata ”ros”-koodin sijaan ”ArmLink”-koodi, joka lukee ohjauskomennot sarjaportin kautta. Tässä toteutuksessa käytetään siis kahta eri koodia, joista Python-koodi ajetaan ajoneuvon tietokoneella ja Arduino-koodi robotin ohjauslevyllä. Ohjaimen arvojen lähettäminen ROS-järjestelmään on jo toteutettuna ajoneuvossa. Kuvassa 4 on ohjauksen toteuttamisessa käyttämäni laitteisto.



Kuva 4: *Järjestelyt ohjauksen toteuttamiseksi*

Hylättyäni yhden vaihtoehdoista jäljelle jäi vain kaksi vaihtoehtoa. Koska ensimmäinen vaihtoehdoista oli jo kokeilun jälkeen toimiva, päätin valita kolmannen vaihtoehdon toteutettavaksi ohjaukseksi. Kirjoittamalla oman ohjelmakoodin saisin ohjattua robottia PlayStation-ohjaimella, mikä vastaa työn tavoitteita ensimmäistä vaihtoehtoa paremmin. Jos oma ohjelmakoodi ei toimisi, pystyisin palaamaan ensimmäiseen vaihtoehtoon.

4.2 Ohjauksen toteuttaminen

Valmisteluvaiheessa valitsin robottikäsivarren ohjaukseksi oman Python-ohjelman kirjoittamisen. Valmisteluihin kuului myös ”ArmLink”-koodin lataaminen WidowX-robottikäsivarren ohjauslevylle. Tässä alaluvussa perehdytään yksityiskohtaisesti ohjauksen toteuttamiseen ja ohjelmakoodiin. Valmis ohjelmakoodi on liitteessä A.

Aloitin ohjauksen toteuttamisen kirjoittamalla ensin ohjelmakoodin, joka lukee ROS-järjestelmästä PlayStation-ohjaimen lähettämät arvot. Kuvassa 5 on ohjaimen painikkeet ja niiden lähettämä informaatio. Tiedonsiirto ROS-järjestelmän ja Python-koodin välillä onnistui helposti Pythonin ”rospy”-kirjastolla, joka tarjoaa tähän sopivat toiminnot. Ohjaimen jokainen painike antaa arvon nolla ollessaan vapautettuna ja arvon yksi ollessaan painettuna. Osa painikkeista antaa arvoja nollan ja yhden välillä riippuen painikkeen asennosta. Liukuvia arvoja antavat esimerkiksi ohjaimen takapainikkeet (kuvassa L2 ja R2) ja analogiset sauvat.

Kuva 5: *PlayStation Dualshock4 -ohjaimen painikkeet (InterbotiX X-Series Arm & Turret ROS Packages 2020)*

Kun ohjaimen arvot saadaan luettua, muutetaan ne koordinaatiksi. Käytettävä ”Arm-Link”-koodi tarjoaa IK-laskukaavat kahdelle eri koordinaattijärjestelmälle: cartesian ja cylindrical. Cartesian-koordinaatisto käyttää x-, y- ja z-koordinaatteja sijainnin määrittämiseen, kun cylindrical-koordinaatistolle syötetään vain y- ja z-koordinaatti. Cylindrical-koordinaatiston x-koordinaatti korvataan alustan servoa ohjaamalla, jolloin robotti pyörii pystyakselin ympäri. Tässä työssä on valittu käytettäväksi cartesian-koordinaatistoa.

Koordinaatit määrittävät robottikäsivarren tarttujan sijainnin. Uusi koordinaatti saadaan lisäämällä tarttujan tämänhetkiseen sijaintiin ohjaimen painikkeen arvo kerrottuna sopivalla skaalauskertoimella. Kerroin asetetaan, jotta tarttuja liikkeet olisivat tarkempia painallukseen nähden. Asetin tarttujan liikkumisen x- ja y-akselilla vasempaan analogiseen sauvaan ja z-akselissa oikeaan analogiseen sauvaan. Tarttujan koordinaattien muuttamisen lisäksi asetin ohjaimen painikkeisiin muun muassa tarttujan kiertämisen, taivuttamisen, avautumisen ja sulkeutumisen.

Robottikäsivarrelle lähetettävät tiedot pitää muuttaa robotin ymmärtämäksi komentoviestiksi ennen lähettämistä. Robotin valmistajalla on selkeä dokumentti ArmLink-komentoviestin muodostamisesta (Arm Link Reference 2020). Komentoviesti sisältää, tarttujan x-, y- ja z-koordinaattien ja asennon tietojen lisäksi, myös niin sanotun nopeusarvon, laajennetun ohjeen ja ohjauslevyn painikkeiden arvot. Nopeusarvolla määritetään, kuinka kauan tarttujalla kestää liikkua asetettuun koordinaattiin. Viestissä voi lähettää myös valmiita komentoja, jotka syötetään laajennettuna ohjeena. Laajennettu ohje jättää syötetyn koordinaatin ja muut tiedot huomiotta ja toimii komennon määrittämällä tavalla. Laajennetun ohjeen komento voi esimerkiksi asettaa robotin lepoasentoon tai vaihtaa käytettävää koordinaatistoa. Viestiin voi sisällyttää myös robotin ohjauslevyn painikkeiden antamat arvot, mutta tätä ominaisuutta ei ole tarvetta hyödyntää tässä työssä.

Viestin sisältämät tiedot ovat vielä kokonaislukuja, jotka pitää muuttaa tavuiksi. Lopullinen viesti koostuu yhteensä 17 tavusta. Ensimmäinen tavu on otsikkotavu, joka on jokaisessa viestissä sama. Tavut 2–13 sisältävät tarttujan tiedot. Tarttujan x-, y- ja z-koordinaatit sekä asennon tiedot muutetaan jokainen kahdeksi tavuksi, koska niiden arvot eivät mahdu yhteen tavuun. Nopeusarvo, ohjauslevyn painikkeiden tiedot ja laajennettu ohje muutetaan omiksi tavuiksi. Viestin loppuun pitää lisätä myös tarkistussummatavu, josta robottikäsi tietää viestin tulleen ehjänä perille. Tarkistussumma lasketaan erillisellä laskukaavalla viestin tavujen summasta. (Arm Link Reference 2020) Lopuksi tavut yhdistetään tavujonoksi.

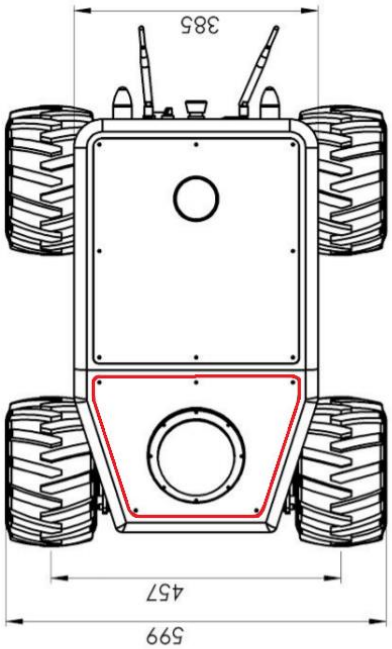
Komentoviestin lähettämiseen sarjaportin kautta käytetään Pythonin pySerial-kirjastoa (pySerial 3.0 documentation – Short introduction 2020). Ensin sarjaportille määritetään oikeat asetukset. Asetukset sisältävät muun muassa käytettävän portin, tiedonsiirtonopeuden, tavukoon ja aikakatkaisun. Sarjaportin asetusten ollessa kunnossa, pystytään komentoviesti lähettämään robottikädelle. Kun Python-ohjelma käynnistetään, robotti siirtyy määrittämääni aloitusasentoon. Tämän jälkeen ohjelma lähettää ohjauskomentoja robotille, kunnes ohjelma suljetaan. Ohjelma lähettää komennot myös ROS-järjestelmään, josta komentoja pystyy tarkastelemaan. Ohjelman päätyttyä robotti siirtyy lepoasentoon.

Sain tavoittelemani ohjauksen toimimaan. Ohjaimen arvojen lukeminen onnistui muutamalla rivillä ohjelmakoodia ja komentoviestin muodostamiseen oli selkeät ohjeet ArmLink-dokumentaatiossa. Eniten vaikeuksia ohjelmakoodin kirjoittamisessa tuotti komentorivin lähettäminen robotille ja pySerial-kirjaston käyttö. ArmLink-dokumentti neuvoi

muuttamaan lähetettävät tiedot tavuiksi, mikä onnistui ongelmitta. Jouduin kuitenkin erikseen selvittämään, että robotille pitääkin lähettää näistä tavuista muodostettu tavujono. PySerial-kirjaston kanssa minulla oli hankaluuksia saada luettua robotin takaisinlähettämää dataa. En kuitenkaan osannut helposti muuttaa robotin lähettämä dataa tavuista luettavaan muotoon. Olisin hyödyntänyt dataa selvittääkseni, mitä komentoa robotti toteuttaa sillä hetkellä. Tätä tietoa ei tarvittu lopullisessa toteutuksessa, joten en perehtynyt siihen enempää.

4.3 Robottikäsivarren liittäminen ajoneuvoon

Kun robottikäsivarren ohjaus on saatu toimimaan, voidaan se kytkeä ajoneuvoon. Robotti kiinnitetään akryylilevyyn, joka ruuvataan ajoneuvon keulaan kameran tilalle. Kuvaan 6 on merkitty keulan levy, jolle robotti kiinnitetään. Kiinnityksessä tulee huomioida, ettei robotti haittaa muita ajoneuvon komponentteja, kuten antureita. Robotti tarvitsee toimiakseen 12 V tasajännitteen. Jännite on sama kuin ajoneuvon kamerassa, jonka tilalle robotti kiinnitetään. Robotti voidaan siis kytkeä ajoneuvon sulakerasian ulostuloon samaan kohtaan, johon kamera oli kytketty.



Kuva 6: *Robotin kiinnityskohta. Perustuu lähteeseen (Robotnik SUMMIT XL System Maintenance Manual 2020).*

Ajoneuvon tietokoneeseen robotti yhdistetään USB-kaapelilla. Lisäksi tietokoneelle pitää ladata käytettävä ohjausohjelma, joka on tässä tapauksessa itse kirjoittamani Python-koodi. Ohjelma myös asetetaan käynnistymään samalla, kun ajoneuvo käynnistetään. Tämä onnistuu ROS-järjestelmän tarjoamalla ”roslauch”-komennolla, joka ajaa erillisessä ”launch”-tiedostoon merkityt ohjelmat. Ajoneuvon tietokoneella on muitakin käynnistämisen yhteydessä ajettavia ohjelmia, joten voin lisätä oman ohjelmani olemassa olevaan ”launch”-tiedostoon.

5. PÄÄTELMÄT

Sain robottikäden kolmesta ohjausvaihtoehdoista toimimaan kaksi toteutusta, joten lasken työn onnistuneeksi. Pystyn nyt valitsemaan kahdesta toimivasta ohjauksesta, kumpaa haluan käyttää robotissa. Valmiin RVIZ-työkalun etuna on sen käyttövarmuus ja luotettavuus. Valmistaja on jo tehnyt ohjauksesta toimivan. Omassa ohjauskoodissani etuna on, että robottia pystyy ohjaamaan samalla PlayStation-ohjaimella kuin ajoneuvoakin. Mielestäni oleellista robotin ohjauksessa on helppokäyttöisyys, joten valitsin PlayStation-ohjainta tukevan oman ohjelmakoodini robotin ohjaamiseen.

Työn suorittamisen aikana ei ilmennyt ongelmia. Vaikeuksia tuotti ainoastaan pySerial-kirjaston käyttö Python-koodia kirjoittaessa. Muuten työ eteni suunnitelman mukaan. Lopullinen robottikäden ohjaus on toimiva, mutta ei kuitenkaan aivan vastaa odotuksiani. Ohjaus toimii nimittäin pienellä viiveellä. Viiveeseen vaikuttaa etenkin taajuus, jolla ohjelma lähettää komentoja. Viivettä kasvattaa myös se, että robotti vastaanottaa koko ajan uutta komentoa. Robotti toteuttaa komennoista tuoreimman, joten se joutuu liikkumaan aina kohti uutta pistettä. Ohjausta voisi hienosäätää siten, että robotti liikkuisi vasta tavoitekoordinaatin muuttuessa. Tällöin lähetettävien komentojen määrä vähenisi. Olen viiveestä huolimatta tyytyväinen saavuttamaani lopputulokseen. Ohjausta voisi myöhemmin kehittää asettamalla robotille tarkemmat toiminta-alueen rajat. Rajat estäisivät käyttäjää ohjaamasta tarttujaa esimerkiksi ajoneuvon renkaiden alle. Näin vältyttäisiin tuottamasta turhaa vahinkoa. Rajojen määrittäminen onnistuisi mahdollisesti IK-laskukaavoja muokkaamalla.

Kiinnittämällä robottikäsivarsi autonomiseen ajoneuvoon saadaan ajoneuvolle monipuolisempia käyttötarkoituksia. Sen lisäksi että ajoneuvo kulkee sille asetettuun päämäärään, se voisi kohteessaan liikuttaa kappaleita robottikäsivarrellaan. Ajoneuvo voisi toimia apuna esimerkiksi ahtaissa tai likaisissa paikoissa, joihin ihminen ei pääse tai halua mennä. Tosin robottikäsivarren korvatessa ajoneuvon kameran ei ajoneuvon liikkumista pysty enää seuraamaan. Tämän ongelman voisi tarvittaessa ratkaista liittämällä ajoneuvoon pienemmän kameran.

LÄHTEET

Arduino Software, Arduino, verkkosivu. Saatavissa (viitattu 27.4.2020): https://www.arduino.cc/en/main/software

Arm Link Reference, Trossen Robotics, verkkodokumentti. Saatavissa (viitattu 25.4.2020): https://learn.trossenrobotics.com/arbotix/arbotix-communication-controllers/31-arm-link-reference.html

DYNAMIXEL MX-64T, Robotis, verkkosivu. Saatavissa (viitattu 2.3.2020): http://www.robotis.us/dynamixel-mx-64t/

Fernández, E. (2015) Learning ROS for robotics programming: Your one-stop guide to the Robot Operating System. 2nd ed. Packt Publishing.

InterbotiX X-Series Arm & Turret ROS Packages, GitHub, lähdekoodi. Saatavissa (viitattu 26.4.2020): https://github.com/Interbotix/interbotix_ros_arms

PhantomX Reactor Robot Arm Kit, Trossen Robotics, verkkosivu. Saatavissa (viitattu 26.3.2020): https://www.trossenrobotics.com/p/phantomx-ax-12-reactor-robot-arm.aspx

PincherX 150 Robot Arm, Trossen Robotics, verkkosivu. Saatavissa (viitattu 26.3.2020): https://www.trossenrobotics.com/pincherx-150-robot-arm.aspx

pySerial 3.0 documentation – Short introduction, Python, verkkodokumentti. Saatavissa (viitattu 27.4.2020): https://pyserial.readthedocs.io/en/latest/shortintro.html#

Robotnik SUMMIT XL System Maintenance Manual, Robotnik Automation SLL, verkkodokumentti, 29 p. Saatavissa (viitattu 22.4.2020): https://www.manualslib.com/manual/1438389/Robotnik-Summit-Xl.html

ROS Documentation – Concepts, verkkodokumentti. Saatavissa (viitattu 2.3.2020): http://wiki.ros.org/ROS/Concepts

ROS Documentation – Introduction, verkkodokumentti. Saatavissa (viitattu 2.3.2020): http://wiki.ros.org/ROS/Introduction

ROS packages to work with the WidowX Arm, GitHub, lähdekoodi. Saatavissa (viitattu 26.4.2020): https://github.com/Interbotix/widowx_arm

ViperX 300 Robot Arm 6DOF, Trossen Robotics, verkkosivu. Saatavissa (viitattu 26.3.2020): https://www.trossenrobotics.com/viperx-300-robot-arm-6dof.aspx

WidowX Robot Arm Kit, Trossen Robotics, verkkosivu. Saatavissa (viitattu 2.3.2020): https://www.trossenrobotics.com/widowxrobotarm

LIITE A: ROBOTIN OHJAUKSESSA KÄYTETTY OHJELMAKOODI

```
1   import serial
2   import struct
3   import rospy
4   from std_msgs.msg import String
5   from sensor_msgs.msg import Joy
6
7
8   def joy_callback(data):
9       global axes
10      global buttons
11
12      axes = data.axes
13      buttons = data.buttons
14
15
16  def funktio():
17      r = rospy.Rate(10);
18
19      # X
20      data[0] += axes[0] * -0.25
21
22      # Y
23      data[1] += axes[1] * 0.25
24
25      # Z
26      data[2] += axes[5] * 0.25
27
28      # W_ANGLE
29      data[3] -= buttons[6] * 0.1
30      data[3] += buttons[7] * 0.1
31
32      # W_ROTATE
33      data[4] += axes[9] * 0.5
34
35      # GRIPPER
36      data[5] += axes[10] * 0.5
37
38
39      # Limits:
40
41      # X
42      if data[0] < -300 : data[0] = -300
43      if data[0] > 300 : data[0] = 300
44
```

```
        # Y
        if data[1] < 50 : data[1] = 50
        if data[1] > 400 : data[1] = 400

        # Z
        if data[2] < 20 : data[2] = 20
        if data[2] > 350 : data[2] = 350

        # W_ANGLE
        if data[3] < -30 : data[3] = -30
        if data[3] > 30 : data[3] = 30

        #W_ROTATE
        if data[4] < 0 : data[4] = 0
        if data[4] > 1023 : data[4] = 1023

        # GRIPPER
        if data[5] < 0 : data[5] = 0
        if data[5] > 512 : data[5] = 512

        send(data[0], data[1], data[2], data[3], data[4], data[5], data[6],
data[7], data[8])

        # Int to bytes:

    def i2b(num):
        bytes = [int( int(num) >> i & 0xff) for i in (8, 0)]
        return bytes

    def i2bb(num):
        bytes = [int(num) % 256]
        return bytes


    def send(x, y, z, w_angle, w_rotate, gripper, delta, button, extended):

        # Create a list of message informations

        vector = [0xff]
        vector.extend(i2b(x + 512))
        vector.extend(i2b(y))
        vector.extend(i2b(z))
        vector.extend(i2b(w_angle + 90))
        vector.extend(i2b(w_rotate))
        vector.extend(i2b(gripper))
        vector.extend(i2bb(delta))
        vector.extend(i2bb(button))
        vector.extend(i2bb(extended))

```

```
        # Calculate checksum
        byte_sum = 0
        for i in range(1,16):
            byte_sum = byte_sum + vector[i]

        inverted_checksum = byte_sum % 256
        checksum = 255 - inverted_checksum

        vector.append(checksum)

        # Send message to robot
        ser.write(bytearray(vector))
        message.publish(', '.join([str(elem) for elem in vector]))

    def main():

        global ser
        ser = serial.Serial()
        ser.port = '/dev/ttyUSB0'
        ser.baudrate = 38400
        ser.bytesize = serial.EIGHTBITS
        ser.parity = serial.PARITY_NONE
        ser.stopbits = serial.STOPBITS_ONE
        ser.timeout = 1

        ser.open()

        rospy.init_node('read')
        r = rospy.Rate(10);

        #rospy.Subscriber("/summit_xl/joy", Joy, joy_callback)
        rospy.Subscriber("/joy", Joy, joy_callback)
        r.sleep()

        global message
        message = rospy.Publisher('widowx', String, queue_size=10)

        send(0, 0, 0, 0, 0 ,0, 0, 0, 32) # Change Mode to 3D Cartesian

        send(0, 250, 225, 0, 512, 256, 128, 0, 0) # Go to start position

        r.sleep()

        global data
        data = [0, 250, 225, 0, 512, 256, 125, 0, 0]


        while not rospy.is_shutdown():
             funktio()

```

```
        send(0, 0, 0, 0, 0 ,0, 0, 0, 96) # Sleep mode

        rospy.spin()

        ser.close()


    if __name__ == '__main__':

       try:
       main()
       except rospy.ROSInterruptException:
             pass

```