OULUN AMMATTIKORKEAKOULU

Eino-Pekka Vanttaja

ROBOTTIKÄSIVARREN LIIKEOHJAUKSEN INTEGROINTI MOBIILIROBOTTIALUSTAAN PERUSTUEN TAITOPOHJAISEEN OHJAUSRAKENTEESEEN

ROBOTTIKÄSIVARREN LIIKEOHJAUKSEN INTEGROINTI MOBIILIROBOTTIALUSTAAN PERUSTUEN TAITOPOHJAISEEN OHJAUSRAKENTEESEEN

Eino-Pekka Vanttaja
Opinnäytetyö
Syksy 2019
Sähkö- ja automaatiotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Sähkö- ja automaatiotekniikan tutkinto-ohjelma, automaatiotekniikka

Tekijä: Eino-Pekka Vanttaja
Opinnäytetyön nimi suomeksi: Robottikäsivarren liikeohjauksen integrointi mobiilirobottialustaan perustuen taitopohjaiseen ohjausrakenteeseen
Opinnäytetyön nimi englanniksi: Integration of Robot Arm to Mobile Robot Platform Using Skill-Based Control Structure
Työn ohjaajat: Tapio Heikkilä (VTT), Tero Hietanen (Oamk)
Työn valmistumislukukausi ja -vuosi: Syksy 2019
Sivumäärä: 30 + 1 liite

Tässä opinnäytetyössä oli tarkoituksena toteuttaa ja integroida robottikäsivarren liikeohjaus mobiilirobottialustaan käyttäen taitopohjaista ohjausrakennetta. Taitopohjaista ohjausrakennetta käyttäen mobiilirobotti osaa missiosta riippuen hakea ja kuljettaa kappaleen paikasta A paikkaan B, hakea kappaleen paikasta A tai kuljettaa kappaleen paikkaan B. Taito on esimerkiksi ”poimi kappale”. Työ on osa pidempiaikaista mobiilirobotin kehitystyötä ja se tehtiin täysin käyttäen VTT:n tiloja.

Mobiilirobottia on pystytty ajamaan aikaisemmin käyttäen käsiajoa ohjaustietokoneelta tai etänä WLAN-yhteydellä olevalta koneelta. Navigointiajolla mobiilirobotti on osannut seurata sille tallennettua reittiä. Robotille kehitettiin robottikäden ja tarttujan ohjaukset sekä lisättiin taitopohjainen alustan ohjaus. Reitin mobiilirobotin tulisi osata valita vertailemalla omaa paikkaansa reitin aloituspaikkaan ja reitin lopetuspaikkaa kappaleen poiminnan tai asetuksen paikkaan. Ohjaukset on kirjoitettu C++-kielellä käyttäen Qt Creator -ohjelmointiympäristöä.

Lopputuloksena saatiin mobiilirobotille toimiva taitopohjainen ohjaus, jonka mukaan mobiilirobotti osaa ajaa reitin aloituspaikasta kappaleen luokse, tunnistaa sen konenäöllä ja nostaa sen kyytiin ja ajaa paikkaan, jossa tapahtuu kappaleen asettaminen tasolle.

Asiasanat: taito, robotiikka, mobiilirobotti, konenäkö

ABSTRACT

Oulu University of Applied Sciences
Degree Programme of Electrical and Automation Engineering, Automation

Author: Eino-Pekka Vanttaja
Title of thesis: Integration of Robot Arm to Mobile Robot Platform Using Skill-Based Control Structure
Supervisors: Tapio Heikkilä (VTT), Tero Hietanen (Oamk)
Term and year when the thesis was submitted: Fall 2019
Pages: 30 + 1 appendices

The goal of this thesis was to engineer and integrate a robot arm control to a robot platform using a skill based control structure. A skill is for example “pick object”. The mobile robot can transport an object from A to B, pick an object from A or place an object to B based on selected mission. The whole work was done in premises of VTT and it is a part of long-term development of mobile robot.

User can control the mobile robot with arrow keys from a control computer or with remote control computer, which uses WLAN connection. Navigation drive where the robot follows a saved route has been also featured. Control programs were written in C++ language using Qt Creator development environment.

The result was a working skill based control. The mobile robot can perform a mission for example “transport” where it executes a task “goto” and drive from starting location to location A. On location A, the mobile robot uses its machine vision to detect the object and after that executes a “pick” task. After “pick” the mobile robot execute “goto” task again and drives to location B. On location B, the mobile robot executes “place” task.

Keywords: skill, machine vision, robotics, mobile robot

ALKULAUSE

Olen ollut kiinnostunut roboteista pitkään ja on ollut hienoa päästä työskentelemään aiheen parissa opinnäytetyön merkeissä. Suuret kiitokset menevät Oamk:n tutkintovastaava Tero Hietaselle tästä hienosta mahdollisuudesta työskennellä näin mielenkiintoisen vaikkakin haastavan aiheen parissa.

Kiitokseni haluan osoittaa myös VTT:n työntekijöille, jotka ovat tavalla tai toisella auttaneet minua työn edistymisessä. Erityisesti haluan kiittää VTT:n johtava tutkija Tapio Heikkilää sekä insinööri Jarkko Kotaniemeä. Heidän osaamisensa ja asiantuntemuksensa toivat valtavan avun tähän työhön.

Oulussa 9.9.2019

Eino-Pekka Vanttaja

SISÄLLYS

TIIVISTELMÄ 3
ABSTRACT 4
ALKULAUSE 5
SISÄLLYS 6
SANASTO 7
1 JOHDANTO 8
2 ROBOTTIKÄDEN OHJAUKSEEN KÄYTETYT TYÖKALUT 9
2.1 Qt 9
2.2 KEBA 9
2.2.1 KEBA KeTop T55 pendantti ja KeMotion KAIRO 10
2.2.2 KEBA KeMotion CP 242/A PLC 10
2.3 SCHUNK PowerBall LWA 4P -robottikäsivarsi ja WSG 25 -tarttuja 11
2.4 Intel RealSense D435 3D-kamera 13
3 TAITOPOHJAINEN OHJAUSRAKENNE 14
3.1 Taitopohjaisuuden hierarkia 14
3.2 Taitorakenteen käyttö 15
4 TYÖN SUORITUS 17
4.1 Robottikäsivarren ja tarttujan integrointi ja ohjauksen suunnittelu 17
4.2 Missio-ohjauksen luonti ja integrointi mobiilirobotin ohjelmistoon sekä alustan ohjelman kehittäminen 19
4.3 3D-kameran asennus ja koordinaattimuutokset 20
5 TULOSTEN TARKASTELU 22
6 POHDINTA 27
LÄHTEET 28
LIITTEET 31

SANASTO

C++	c-kieleen perustuva ohjelmointikieli, johon on lisätty ominaisuuksia
KEBA	valmistaa ohjelmoitavia logiikoita sekä robottikäsivarsien ohjaukseen käytettäviä pendantteja eli käsiohjaimia
PLC	ohjelmoitava logiikka, jota käytetään reaaliaikaisten automaattiprosessien ohjaukseen
Qt	alustariippumaton ohjelmistojen ja graafisten käyttöliittymien kehitysympäristö, lausutaan samoin kuin englannin kielen sana ”cute”
Schunk	valmistaa robotiikkatuotteita muun muassa tarttujia ja robottikäsivarsia

1 JOHDANTO

Työn tarkoituksena oli tehdä robottikäsivarren ja tarttujan ohjaukset sekä käyttöönotto Probot Oy:n valmistamaan mobiilirobottialustaan käyttäen konenäköä ja taitopohjaista ohjausrakennetta. Työ on tehty täysin VTT:n tiloissa. Teknologian tutkimuskeskus VTT Oy on kansallisella statuksella toimiva yksi Euroopan johtavista tutkimus- ja teknologiaorganisaatioista (1).

Mobiilirobottialustalle on aikaisemmin tehty koordinaattiohjaus (2), navigointimenetelmät (3), navigointimenetelmän parantaminen odometrian avulla (4) sekä paikannuksen ja ympäristön mallinnus etäohjauksen avulla (5). Koordinaattiohjauksen on tehnyt Jussi Pulkkinen, navigointimenetelmät ovat Taavi Oksasen tekemiä, navigoinnin parannukset odometrian avulla on toteuttanut Jarkko Kotaniemi ja viimeisimpänä Juho Kotkaranta on saanut aikaan paikannuksen ja ympäristön mallinnuksen etäohjauksen avulla.

Aiemmin robottia on ajettu nuolinäppäimillä robotin kyydissä olevalta tietokoneelta, etäohjauksena toiselta tietokoneelta sekä automaattiajolla konenäöllä tai ”sokea-ajolla”.

Tässä työssä mobiilirobottialustalle toteutettiin ohjelma Qt Creatorilla, jotta mobiilirobotti osaisi verrata omaa paikkaansa listassa olevan reitin aloituspaikkaan, valitsemaan sieltä lähin aloituspaikka ja ajamaan reitin lopetuspaikkaan, jossa mobiilirobotti konenäkönsä avulla osaisi määrätyn tehtävän mukaan tehdä joko tavaran poiminnan ja asettamisen tai pelkän poiminnan tai asettamisen.

Robotin kappaleentunnistus toimii ROS:n (Robot Operating System) pohjalle tehdyllä konenäköohjelmalla. Ohjaukselle ja säädölle on tehty C++-ohjelma Qt Creatorilla. Qt Creatorilla toteutettiin myös ohjelmisto mobiilikäsivarren ja tarttujan ohjaukselle, jossa UDP/IP-yhteydellä lähetetään koordinaattiarvoja PLC:n kautta robottikäsivarrelle sekä ohjausarvot ohjaustietokoneelta suoraan tarttujalle.

2 ROBOTTIKÄDEN OHJAUKSEEN KÄYTETYT TYÖKALUT

Tässä luvussa kerrotaan, mitä ohjelmistotyökaluja on käytetty robottikäsivarren ja tarttujan ohjaukseen. Pääasiallisena ohjelmointiympäristönä toimi Qt Creator, myös robottikäden ohjaukseen käytetyn KEBA:n pendantin käyttämään KAIRO ohjelmointiin sekä KEBA:n valmistaman PLC:n ohjelmointiin tutustuttiin.

2.1 Qt

Qt on alustariippumaton ohjelmistojen ja graafisten käyttöliittymien kehitysympäristö. Pääosin Qt käyttää C++-kieltä. Lisäksi Qt sisältää hyödyllisiä laajennuksia esimerkiksi tapahtumien käsittelyä helpottamaan käytettävät signaalit ja slotit.

Qt:a käyttää hyväksi Qt Creator -ohjelmistoympäristö. Qt Creatorissa on koodieditori ja graafisten käyttöliittymien suunnittelutyökalu sekä se tukee useita eri debuggereita ja versionhallintaohjelmistoja. Pääosin Qt Creator on suunniteltu C++-kielelle, mutta erinäisten kirjastojen avulla se tukee myös muita ohjelmointikieliä. (6.)

Tässä työssä käytettiin hyväksi Qt:n verkkoyhteysominaisuuksia. Verkkoyhteydellä lähetetään mobiilirobotin ohjaustietokoneelta liikekäskyjä PLC:lle, joka ohjaa käskyjen sisältämät x-, y- ja z-arvot robottikäsivarrelle. Tarttujalle kulkevat käskyt tulevat suoraan ohjaustietokoneelta UDP/IP-verkkoyhteyttä käyttäen.

2.2 KEBA

KEBA on kansainvälinen automaatioratkaisuja tuottava ja kehittävä yhtiö. Se toimii teollisuus-, pankkitoiminta-, kunnossapito- ja energia-automaatioiden alueilla. (7.) Tässä työssä käytettiin KEBA:n valmistamaa PLC:tä sekä robottikädenohjaukseen tarkoitettua pendanttia eli käsiohjainta.

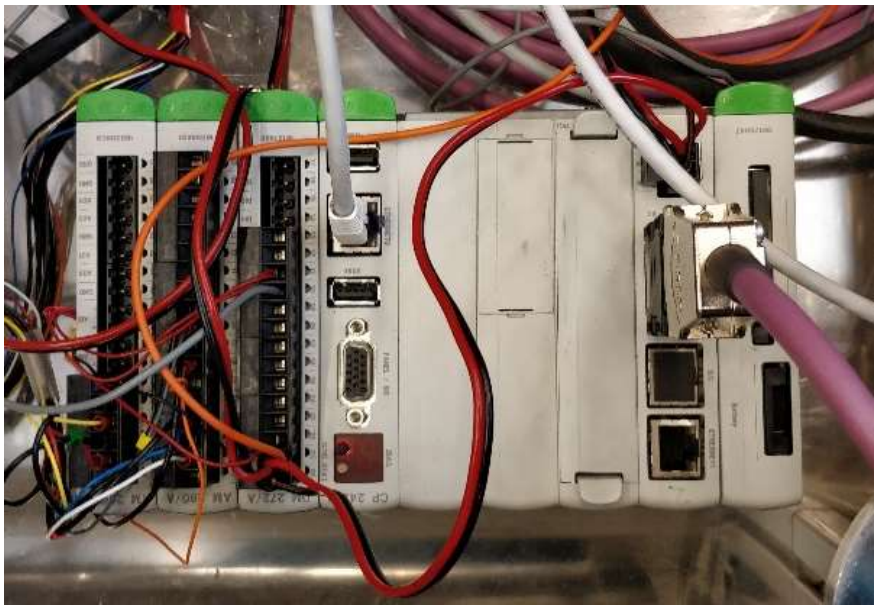
2.2.1 KEBA KeTop T55 pendantti ja KeMotion KAIRO

Robottikäsivartta voidaan ohjata manuaalisesti ja automaattisesti KEBA:n valmistaman pendantin (kuva 1) avulla (8). Manuaaliajolla käyttäjä voi itse ajaa akseleita haluamaansa suuntaan. Automaattiajoa varten pendanttiin täytyy luoda KAIRO makro-ohjelmointikielellä ohjelmisto.

KUVA 1. KEBA KeTop T55 pendantti

2.2.2 KEBA KeMotion CP 242/A PLC

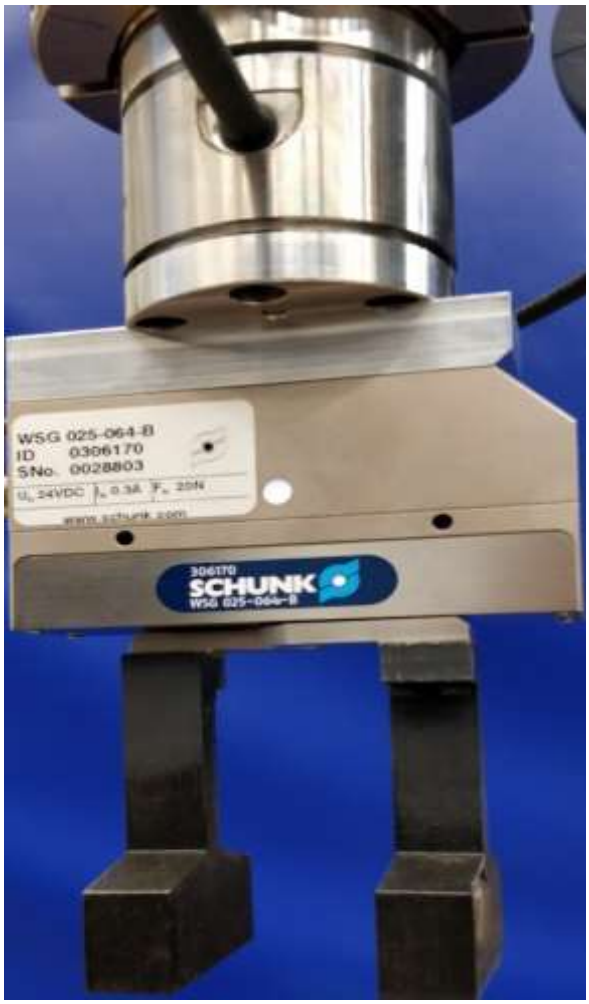
Robottikäsivarteen lähetetään ohjausdataa ohjaustietokoneelta PLC:n (kuva 2) kautta. PLC:lle on aikaisemmassa projektissa luotu ohjelmisto, jota käytettiin tässä työssä hyväksi. Käytettyyn ohjelmistoon lisättiin funktiolohko, joka vastaanottaa ohjauskoneelta lähetettäviä x-, y- ja z-arvoja. Lisätyn funktiolohkon avulla käsivarren liikkeet saatiin hitaista askelmaisista liikkeistä sulavammiksi ja nopeammiksi liikkeiksi.



KUVA 2. KEBA CP 242/A PLC, jossa kiinni I/O-kortteja

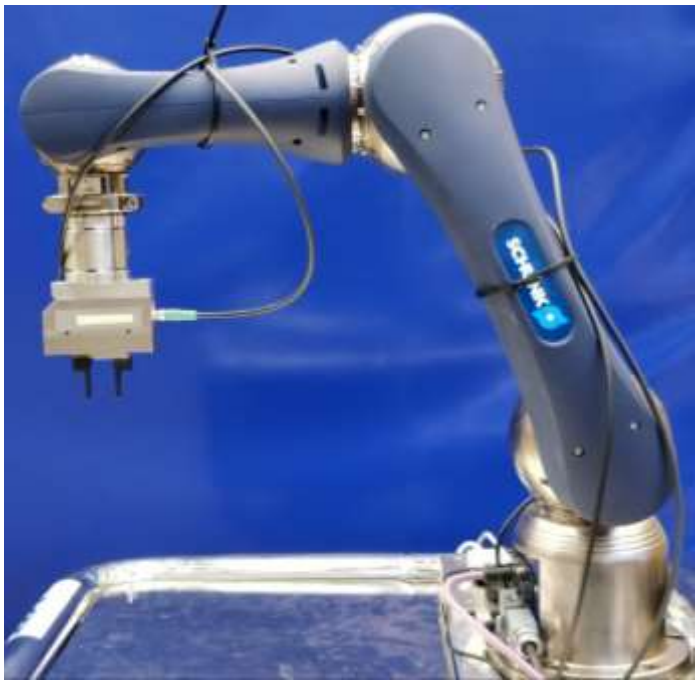
2.3 SCHUNK PowerBall LWA 4P -robottikäsivarsi ja WSG 25-tarttuja

Tässä työssä käytettiin Schunkin valmistamia SCHUNK WSG 25-tarttujaa (kuva 3), sekä SCHUNK PowerBall LWA 4P -robottikäsivartta (kuva 4).
Tarttuja on kaksisorminen ja sormien liikealue on 64 mm. Tarttumisvoima on säädettävissä 5 – 20 N. Todellinen tarttumisvoima riippuu kuitenkin sormien pituudesta sekä sormien ja tartuttavan kappaleen materiaaleista. Tarttumisnopeus on säädettävissä 5 – 300 mm/s-. (9.)



KUVA 3. SCHUNK WSG-25-tarttuja

Robottikäsivarrella on 6 vapausastetta. 7. vapausaste saataisiin Schunkin kolmesormisella SDH-tarttujalla. Akseleiden 1, 2, 4, 5 ja 6 toiminta-alueet ovat ±170° sekä akselin 3 toiminta-alue on ±155,5°. Akseleiden nimellisnopeudet ovat 72°/s. Käsivartta voidaan ohjata joko ROS:lla tai työssä käytetyllä KEBA:n PLC:llä. (10.)



KUVA 4. SCHUNK PowerBall LWA 4P -robottikäsivarsi

2.4 Intel RealSense D435 3D-kamera

Työssä käytetty 3D-kamera (kuva 5) on Intelin valmistama laajakuvalinssillä varustettu syvyyskamera. Se on valmistajansa mukaan täydellinen suurta näkökenttää vaativissa käyttökohteissa, kuten robotiikassa sekä lisätyssä ja virtuaalisessa todellisuudessa. Kamera pystyy havaitsemaan kohteita noin 10 metrin päähän, mutta kalibrointi, paikka ja valoisuus vaikuttavat maksimietäisyyteen. (11.)

KUVA 5. Intel RealSense D435 3D-kamera

3 TAITOPOHJAINEN OHJAUSRAKENNE

”Taito” on korkeatasoinen kokonaisuus, anturimotorinen erittelymalli, jossa korkeampitasoiset operaatiot hajotetaan toiminnoiksi ja ratatason esityksiksi (12 s. 157-170). Turvautumalla ”taito”-lähestymistapaan antureita ja muunneltavia työkaluja voidaan käyttää helpommin. Robottien operaatioiden taitopohjainen ohjelmointi on saanut toistaiseksi vain vähän huomiota (13).

3.1 Taitopohjaisuuden hierarkia

Robotin taitoja voidaan luokitella monin eri tavoin. Puhtaasti liikkeiden näkökulmasta keskeinen tapa on huomioida näiden suhteellisten liikkeiden tyypit kahden vapausasteen välillä. On olemassa 64 mahdollista suhteellista liikeluokkaa kahden kehyksen välillä, esimerkit sisältävät:

- vapaat liikkeet, kuten pallon sisällä oleva pallo (6 vapausastetta)
- piste vasten pintaa tai raja vasten rajaa (5 vapausastetta)
- raja vasten pintaa (4 vapausastetta)
- taso vasten tasoa tai pyöreäpäinen tappi reiässä tai pallo istukassa (3 vapausastetta)
- lieriönmuotoisessa reiässä oleva lieriönmuotoinen tappi (2 vapausastetta)
- ei-sylinterinen tappi ei-sylinterisessä reiässä. (14.)

Taidot voidaan luokitella myös kokonaisvaltaisemmin riippuen niiden yhteydestä ympäristöönsä:

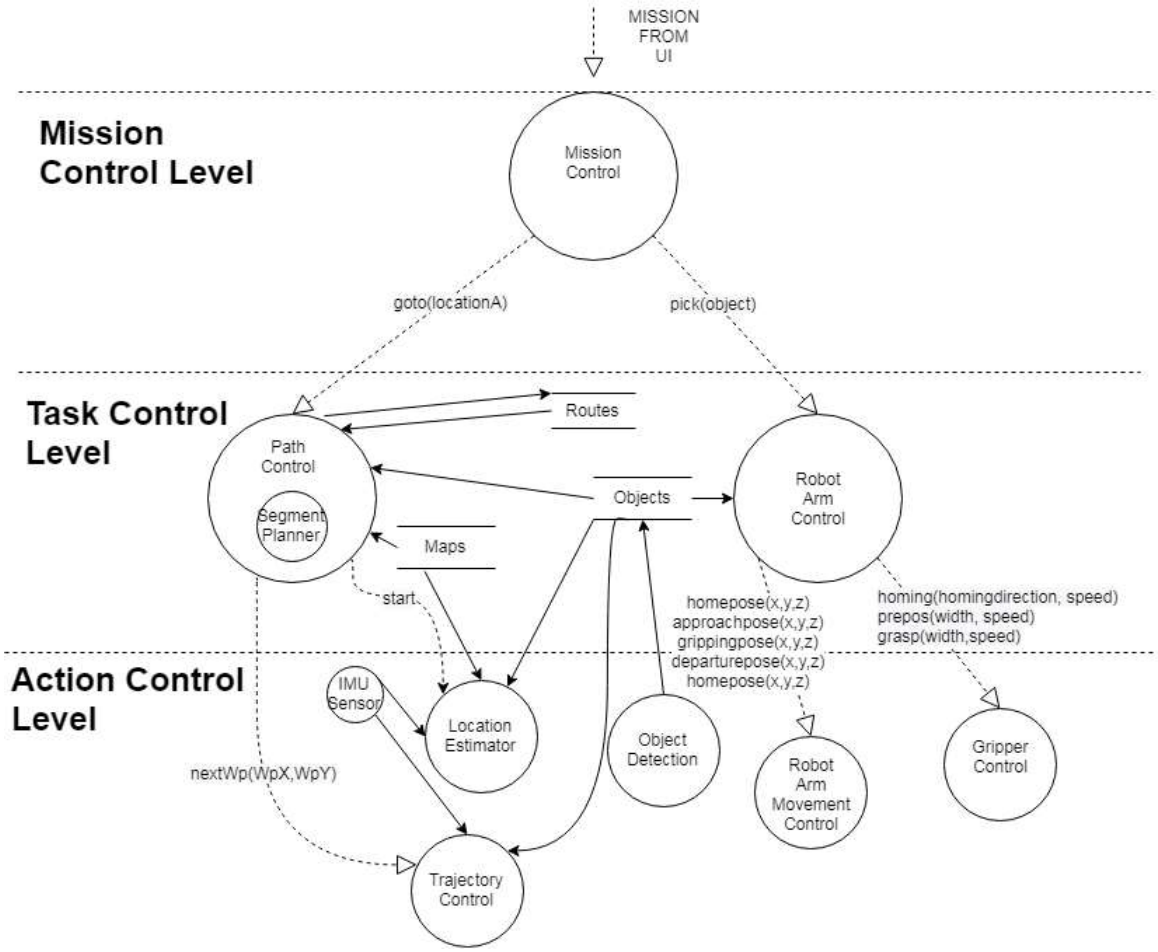
- kohdekappale tarvitaan tai sitä työstetään
- kohdekappaletta siirretään tai se pysyy paikallaan
- kohdekappaleen ominaisuudet ovat muuttuneet
- kohdekappale on liikkunut
- robotti liikkuu
- robotin toimielimet liikkuvat. (15.)

Näitä kahta lähestymistapaa yhdistämällä päästään hierarkkiseen lähestymiseen, jossa korkeamman tason tehtävät ja abstraktimmat taidot on hajotettu liikepareiksi sekä anturi, työkalu ja laite operaatioiksi.

3.2 Taitorakenteen käyttö

Taitopohjaisuus tarkoittaa tämän työn tapauksessa sitä, kun robotti saa mission ”PICK”, joka ohjelman missio-ohjaustasolla hajotetaan missio taskeiksi, jotka ovat ns. taitoja. Taidot ovat tässä missiossa ”siirry paikkaan” ja ”poimi kappale”. Taidot hajotetaan toiminnoiksi task-ohjaustasolla. Toiminnat ovat mobiilialustalle ”etsi reitti” ja ”aja ratapisteeseen”. Toiminnat tarttujalle ja käsivarrelle ovat ”tarttuja koti asentoon”, ”käsivarsi koti positioon”, ”tarttuja esiasentoon”, ”käsivarsi lähestymispositioon”, ”käsivarsi tartuntapositioon”, ”tartu tarttujalla”, ”käsivarsi poistumispositioon” ja ”käsivarsi kotipositioon”. (Kuva 6.)

Taito-ohjaustasolla tapahtuu myös robotin paikoitus karttaan, kartan tallennus, reitin valinta ja reitin tallennus. Lisäksi poimittavan kappaleen tiedot saadaan tällä tasolla.



KUVA 6. Mobiilirobotin ohjausrakenteen tiedonkulku

4 TYÖN SUORITUS

Työn toteuttaminen alkoi tutustumalla mobiilirobotin olemassa oleviin ohjelmistoihin ja dokumentaatioihin. Ohjauksen ohjelmistot on kirjoitettu C++-ohjelmointikielellä käyttäen Qt Creator -editoria. C++ ja Qt eivät olleet tekijälle tuttuja, joten näihin oli perehdyttävä tarkemmin. Robottialustan lisäksi robottikäsivarren ja tarttujan toimintaan tuli paneutua.

4.1 Robottikäsivarren ja tarttujan integrointi ja ohjauksen suunnittelu

Robotin liikeavaruuteen perehdyttiin käyttämällä KEBA:n pendanttia (luku 2.2.1). KEBA:n KAIRO makro-ohjelmointiin sekä PLC:n ohjelmointiin liittyviä dokumentteja käytiin läpi, jotta nämäkin tulisivat tutuiksi. Lisäksi tarttuja asennettiin käsivarteen. Seuraavaksi testattiin valmista, vanhan projektin KAIRO- ja PLC-ohjelmaa.

Heti alusta lähtien ohjauksia lähdettiin suunnittelemaan taitopohjaisiksi. Samalla päätettiin mobiilirobottialustan ohjauksen muokkauksesta niin, että sekin saataisiin toimimaan taitopohjaisen ohjauksen avulla.

Qt Creatorilla tehtiin ensin robottikäsivarren ohjaukseen testiohjelmisto, jossa avataan UDP/IP-yhteys PLC:lle. Samalla käytettiin KeMotion-ohjelmistoa, josta pystytään seuraamaan PLC:n ohjelman tilaa ja näkemään, muuttuvatko halutut arvot. Yhteyden avaamisen jälkeen Qt:n ohjelmasta lähetettiin aluksi siniaalto x-arvoja. Ohjaus toimi, mutta liikkeet olivat hitaita ja hyvin askelmaisia, koska vanhassa projektissa käytetty KAIRO-ohjelma ei sopinut tähän työhön.

Seuraavaksi tutustuttiin tarttujan toimintaan. Tehtiin jälleen testiohjelmisto, jossa avattiin UDP/IP-yhteys tarttujalle ja käytettiin KeMotion ohjelmistoa seuraamaan PLC:n tilaa ja muuttujien arvoja. Tarttujalle on aina käynnistettäessä ensimmäisenä käskynä lähetettävä ”koti”-käsky, jossa tarttuja ajaa sormensa joko täysin auki tai täysin kiinni asentoon, riippuen mikä heksadesimaaliluku ”koti”-käskyssä on annettu. Toisena käskynä voidaan antaa ”esipaikka”-käsky, riippuen tarttujan

”koti”-asennosta ja tartuttavasta kappaleesta. Tässä työssä ”esipaikka”-käskyä ei tarvittaisi, sillä ”koti”-asento on asetettu tarttujan sormien ollessa täysin auki ja tarttuminen tapahtuu kappaleeseen sormien liikkuessa lähemmäksi toisiaan. ”Esipaikka”-käskyssä tulee antaa nopeus- ja leveysarvot. Seuraavat käskyt tarttujalle ovat ”tartu” ja ”vapauta”. Käskyt ovat periaatteeltaan identtisiä, vain sormien liikesuunta muuttuu. Myös näissä käskyissä on annettava leveys- ja nopeusarvot. Kun tarttujan ohjelma todettiin toimivaksi, integroitiin se käsivarren ohjelmaan.

Tarttujan ohjelmiston integroinnin jälkeen tutkittiin KEBA:n PLC:n dokumentaatioita ja etsittiin ratkaisua hitaisiin ja askelmaisiin käsivarren liikkeisiin. Päädyttiin suunnitelmaan, jossa muuttujat lähetettäisiin analogisen sisääntulon kautta RcInputs-nimiseen lohkoon. Löytyikin parempi vaihtoehto: fb_SetCartPos funktiolohko (16 s. 67), jonka sisääntuloon saataisiin suoraan x-, y- ja z-arvot ulkopuolella olevasta lähteestä, tässä tapauksessa ohjaustietokoneelta.

Seuraavaksi KEBA:an tehtiin makro-ohjelma (kuva 7). KAIRO-ohjelman alussa määritetää dynamiikka, referenssijärjestelmä ja liiketyyppi, joka tässä työssä on pisteestä pisteeseen tyyppinen. Dynamiikassa määritetään käsivarren nopeus ja kiihtyvyysarvo sekä lisäksi voidaan antaa päällekkäisiä dynamiikkoja. Referenssijärjestelmässä määrätään käsivarren viitearvot eli nollapiste ja käsivarren koordinaatisto. Tässä työssä nollakohta on tarttujassa. Seuraavalla rivillä annetaan otsikko ”test” sen takia, että ohjelman loppua voidaan pitää silmukassa. Seuraavaksi annetaan 1000 ms viive, jotta liikkeet ehditään tehdä, eikä päällekkäisiä käskyjä anneta. Seuraavana on jälleen pisteestä pisteeseen tyyppinen liikemakro, jonka jälkeen odotetaan, että liike on valmis ja voidaan antaa mene kohtaan ”test”-käsky. Ohjelma on silmukassa niin kauan, kuin ohjelma on käynnissä.

KUVA 7. KAIRO makro-ohjelma

4.2 Missio-ohjauksen luonti ja integrointi mobiilirobotin ohjelmistoon sekä alustan ohjelman kehittäminen

Missio-ohjauksen tekeminen aloitettiin tekemällä käyttöliittymä, josta saa valittua halutun tehtävän. Kun käyttäjä on valinnut tehtävän, aukeaa uusi missioparametri-ikkuna, josta valitaan lokaatiot sekä kohdekappale, joka tässä työssä oli 3D-tulostettu ympyrälieriö. Robottikäden ja tarttujan ohjaukset tuli kuitenkin muokata tilakoneeksi, tässä työssä tilakoneena toimii C++-kielen switch-case rakenne. Tässä vaiheessa testattiin robottikäden ja tarttujan ohjausta sekä etsittiin robottikädelle ihanteellinen kappaleen tartuntakohta. Seuraavaksi missio-ohjaustasolle tehtiin switch-case tilakone, jolta käskyt etenevät tilakoneen mukaan joko robottikäden ohjaukseen tai robottialustan ohjaukseen. Alustan ohjaus muutettiin tilakone ohjaukseksi, jotta taitopohjainen ohjaus oli helppo toteuttaa. Alustan ohjaukseen lisättiin reitin valinta reittilistasta.

Switch-case rakenteessa casesta eli tapauksesta seuraavaan päästään, kun määrätyt ehdot täyttyvät. Tässä työssä esimerkiksi missio ”Transport” alkaa missio-ohjaustasolla switch-casen tapauksesta 1 ”aja paikkaan A”. Ehdot täyttyvät, kun taito-ohjaustasolla alustan ohjauksessa on suoritettu switch-case listassa tapaus 1 ”löydä reitti” ja reitin löydyttyä päästään tapaukseen 2 ”aja seuraavaan

ratapisteeseen”. Tapausta 2 toistetaan niin monta kertaa, että alusta on ajanut määrätyn reitin loppupisteeseen. Kun paikkaan A on päästy, lähetetään ”alustan taito valmis”-viesti missio-ohjaustasolle, jolloin edetään tapaukseen 2. Missio-ohjaustasolla tapaus 2 on ”Pick”, joka suoritetaan taito-ohjaustasolla käsivarrenohjauksella. (Liite 1.) Käsivarrenohjauksen switch-case listassa on 8 toiminnallista tapausta, joissa tapahtuu tarttujan ja käsivarren liikkeet. Kun nämä on suoritettu listan mukaisessa järjestyksessä, lähetetään missio-ohjaukseen jälleen ”robottikäden taito valmis”-viesti. (Liite 2.)

4.3 3D-kameran asennus ja koordinaattimuutokset

Kamera asennettiin mobiilirobotin vasempaan etunurkkaan kuvaamaan hieman etu- ja alaviistoon (kuva 8), jotta tartuttava kappale pystytään havaitsemaan ja tunnistamaan. Asennusta luultiin parhaaksi mahdolliseksi. Kappaleentunnistus tapahtuu ROS-pohjaisella kappaleentunnistusohjelmalla.

KUVA 8. Intel RealSense D435 3D-kamera kiinnitettynä mobiilirobottialustaan

Kameran ja tarttujan akselistot ovat toisiinsa nähden väärät, joten joudutaan käyttämään rotaatiomatriiseja (17 s. 67), jotka lasketaan kaavoilla 1 ja 2. Kameran havaitsemat objektit saadaan tarttujan koordinaatistoon kaavalla 3.

$$ {}^{A}_{B}R_Z(\theta) = \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \textit{KAAVA 1} $$

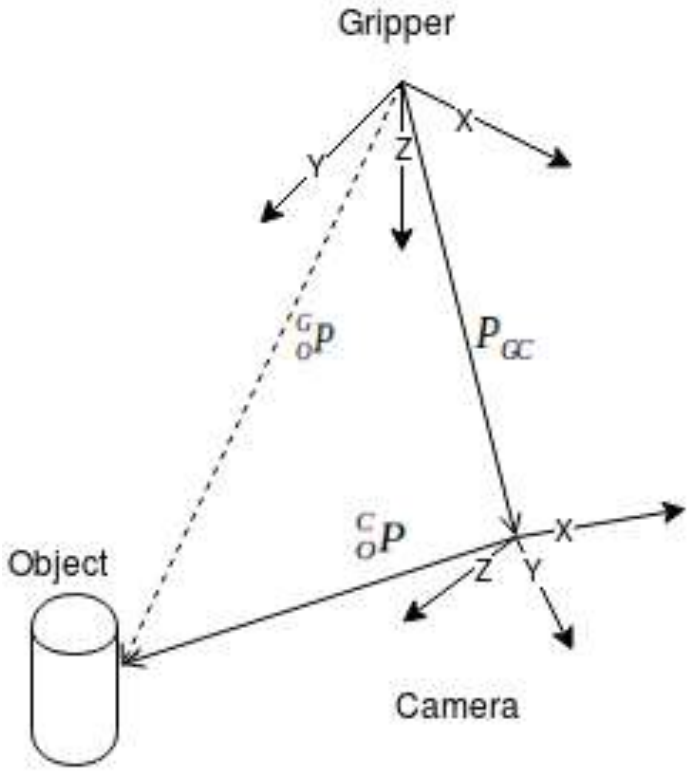
R on rotaatiomatriisi koordinaatistojen A ja B välillä Z-akselin suhteen, sekä θ on koordinaatistojen välinen kulma

$$ {}^{A}_{B}R_X(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{pmatrix} \qquad \textit{KAAVA 2} $$

R on rotaatiomatriisi koordinaatistojen A ja B välillä X-akselin suhteen, sekä θ on koordinaatistojen välinen kulma

$$ {}^{G}_{O}P = R_Z(\theta) * (R_X(\theta) * {}^{C}_{O}P) + P_{GC} \qquad \textit{KAAVA 3} $$

${}^{G}_{O}P$ on kappaleen paikka tarttujan näkökulmasta, ${}^{C}_{O}P$ on kappaleen paikka kameran näkökulmasta ja P_{GC} on tarttujan ja kameran välinen vektori (kuva 9)



KUVA 9. Tarttujan ja kameran akselistot

5 TULOSTEN TARKASTELU

Testaaminen aloitettiin mahdollisimman pienillä liikkeillä, joten testaamista varten missiotason tilakonetta muutettiin niin, että vain robottikäden ja tarttujan ohjausta käytettiin. 3D-kameraa ei tässä vielä käytetty, joten robottikädelle ei saatu kappaleentunnistuksesta arvoja, jolloin tarttuja oli ainoa testattava osa.

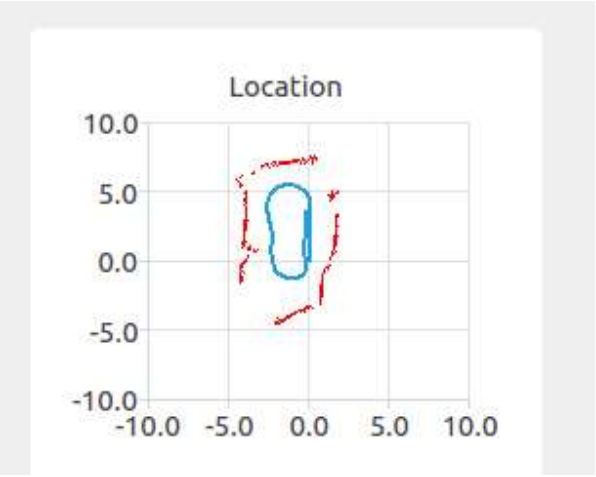
Kun tarttuja toimi oikein, annettiin robottikäden ohjaukseen kovakoodattuja x-, y- ja z-arvoja, sillä 3D-kameraa ei vielä tässä vaiheessa testejä käytetty. Näiden testien perusteella käsivarren ja tarttujan ohjaukset toimivat kuten kuuluikin.

3D-kameran asennuksen jälkeen testauksissa otettiin ensin huomioon, olivatko koordinaatistomuutokset onnistuneet. Vertailtiin debuggaamalla tulostettuja kameran havaitsemia kappaleen x-, y- ja z-arvoja pendantilta luettuihin robottikäden posen arvoihin. Kun arvot olivat tarpeeksi lähellä saatuja arvoja, aloitettiin kappaleen poiminnan testaukset konenäöllä. Ensimmäisissä testeissä huomattiin, että koordinaatistomuutoksissa käytetyt kulmat olivat hieman väärät. Kulmaa muuttamalla kappale saatiin tason tietyssä paikassa poimittua.

Poimintaa testattaessa kävi ilmi, että tarttujan sormet ovat liian ohuet, sillä kappaleen ihanteellinen tartuntakohta osoittautui hyvin kapeaksi. Leveämmät sormet tulostettiin 3D-tulostimella. Uusien sormien asennuksen jälkeen testattiin poimintaa uudelleen ja todettiin, että koordinaattikulmat olivat vieläkin väärät, joten kulmat mitattiin uudestaan. Tarkemmin mitatut kulmat olivat nyt tarpeeksi tarkat ja kappale pystyttiin poimimaan tason eri pisteistä yhtä tarkasti. Kun poiminta oli tarkka, otettiin testiin mukaan myös kappaleen asettaminen tasolle. Tämä toimi moitteetta.

Kun kappaleen poiminta konenäöllä oli saatu toimimaan, oli vuorossa alustan ajon testien suorittaminen. Testit aloitettiin ajamalla käsiajolla reitti, jotta reitin pisteitä saataisiin asetettua kappaleen poimintaa varten olevaan paikkalistaan. Nyt voitiin testata taitopohjaista alustan ohjausta. Tässä vaiheessa käytettiin vain mobiilirobotin ohjaukseen tarkoitettua tietokonetta, eikä siis alustan oikealle kuvaavaa seinäntunnistuksiin tarkoitettua kameraa käytetty ollenkaan. Tämän johdosta ajettu rata oli kuitenkin kuin päädyistä typistetty versio halutusta reitistä, joten

testiajoja jatkettiin lisäämällä testeihin seinientunnistuksia varten oleva oikealle kuvaava kamera parantamaan robotin paikannusta. Tämä ei tuonut haluttua tulosta, joten päädyttiin olemaan käyttämättä kameraa paikan tarkennuksessa. Pääteltiin, että typistynyt reitti johtui nopeuden kalibroinnissa olevasta virheestä, joten nopeus kalibroitiin uudestaan. Kalibroinnin jälkeen ajettiin käyttäjäajolla reitti, joka tallennettiin. Kartan tallennusta varten käytettiin reitin ajossa oikealle sivulle kuvaavaa kameraa, jotta käyttöliittymään saatiin kartan kuvaajaan kameralla havaitut seinäpinnat. (Kuva 10.) Reitin aloituspaikka merkattiin lattiaan, jotta testejä tehdessä reitin aloituspaikka ei muuttuisi vaan pysyisi vakiona (Kuva 11). Tämän jälkeen ajettiin tallennettu reitti käyttäen tässä työssä tehtyä tilakoneella toteutettua navigointiajoa. Reitin ajo onnistui, eikä reitti ollut enää typistetty versio alkuperäisestä reitistä.



KUVA 10. Mobiilirobotilla ajettu reitti sinisellä ja havaitut seinäpinnat punaisella

KUVA 11. Testeissä käytetty mobiilirobotin aloituspaikka

Päästiin tekemään lisää testiajoja. Navigoinnin heikon tarkkuuden takia jouduttiin säätämään reittilistan arvoja useaan otteeseen, jotta alustalla päästiin tarpeeksi lähelle tasoa, josta kappaleenpoiminta ja jolle kappaleen asetus tapahtui. Kun reitin ajo saatiin säädettyä niin, että pysähdykset tason luona tapahtuivat sopivalla etäisyydellä, asetettiin poimittava kappale tasolle ja otettiin testeihin mukaan käsivarrenohjaus. Robotti ajoi lähdöstä paikkaan A (kuva 12) ja poiminta onnistui ja alusta jatkoi ajoa paikkaan B, jossa navigoinnin tarkkuuden takia alusta ajautui hieman kauemmaksi tasosta kuin paikassa A. Kappaleen asetus tasolle kuitenkin onnistui ja voitiin todeta, että robottikäden integrointi ja sen ohjaus olivat onnistuneet.

KUVA 12. Tilannekuva kappaleen poiminnasta

KUVA 13. Mission suoritettuaan mobiilirobotti jää kappaleen asetuspaikkaan B.

6 POHDINTA

Tämän opinnäytetyön tavoitteena oli toteuttaa ja integroida robottikäsivarren ja tarttujan taitopohjaiset liikeohjaukset sekä muuttaa Probot Oy:n valmistaman mobiilirobottialustan ohjaus taitopohjaiseksi. Robottikäden ja tarttujan ohjausarvot kulkevat UDP/IP-verkkoyhteysominaisuuksia käyttäen. Mobiilirobottiin lisättiin ROS-pohjaista konenäköä käyttävä 3D-kamera tunnistamaan poimittava kappale.

Työ oli tekijälle mieluinen ja antoi hienon mahdollisuuden päästä tutustumaan robotiikkaan ja mobiilirobotin kehitystyöhön. Tekijälle karttui kokemusta Qt Creatorin käytöstä ja C++-kielen ohjelmoinnista.

Työtä suorittaessa huomattiin muutamia kehitystä vaativia kohteita. Robottikäsivarsi on pieni, jolloin on ajettava hyvin lähelle tasoa, jolla poimittava kappale on. Tämä toi testeissä oman haastavuutensa reittien suunnitteluun. Toinen kehityskohde huomattiin 3D-kameran asennuksen jälkeen. Kamera asennettiin juuri käsivarren kulkulinjalle, joten käsivarren lähestymis- ja poistumispositioon tapahtuvissa liikkeissä tarttujan sormet käyvät aika lähellä 3D-kameraa. Tämä voitaisiin mahdollisesti korjata pelkästään kameraa hieman siirtämällä tai kääntämällä.

Navigoinnin todettiin olevan myös epätarkkaa, varsinkin akkujen varauksen laskiessa. Navigointia saatiin hieman tarkemmaksi kalibroimalla nopeus uudestaan. Työn suorituksessa haasteita aiheutti virtalähde, joka olisi hyvä korvata johdottomalla ratkaisulla.

LÄHTEET

1. Tietoa meistä. 2019. VTT. Saatavissa: https://www.vtt.fi/tietoa-meist%C3%A4 . Hakupäivä 5.11.2019.

2. Pulkkinen, Jussi 2016. Mobiilirobotin koordinaattiohjaus. Saatavissa: http://urn.fi/URN:NBN:fi:amk-2016110315720. Hakupäivä 5.11.2019.

3. Oksanen, Taavi 2017. Mobiilirobotin navigointimenetelmä. Saatavissa: http://urn.fi/URN:NBN:fi:amk-2017052910821 . Hakupäivä 5.11.2019.

4. Kotaniemi, Jarkko 2018. Mobiilirobotin navigointimenetelmän parantaminen odometrian avulla. Saatavissa: http://urn.fi/URN:NBN:fi:amk-201802122312 . Hakupäivä 5.11.2019.

5. Kotkaranta, Juho 2018. Mobiilirobotin paikannus ja ympäristönmallinnus etäohjauksen avulla. Saatavissa: http://urn.fi/URN:NBN:fi:amk-2018120319638 . Hakupäivä 5.11.2019.

6. About Qt. 2019. Qt Wiki. Saatavissa: https://wiki.qt.io/About_Qt . Hakupäivä 5.11.2019.

7. About KEBA. 2019. Saatavissa: https://www.keba.com/en/corporate/about/overview/overview . Hakupäivä 5.11.2019

8. KEBA KeTop T55 pendantti. 2019. Saatavissa: http://www.otomasyonline.com/Data/Keba/KeTop%20T55.pdf . Hakupäivä 5.11.2019.

9. Assembly and Operating Manual WSG 25. s. 14. 2019. Saatavissa: https://schunk.com/fileadmin/pim/docs/IM0017886.PDF . Hakupäivä 5.11.2019.

10. Modular and Mobile Gripping Systems. s.20-21. 2019. Saatavissa: https://schunk.com/fileadmin/pim/docs/IM0019885.PDF . Hakupäivä 5.11.2019

11. Intel RealSense D435. 2019. Saatavissa: https://www.intelrealsense.com/depth-camera-d435/ . Hakupäivä 5.11.2019.

12. Boada, Maria – Barber, Ramon – Salichs, Miguel 2002. Visual approach skill for a mobile robot using learning and fusion of simple skills. Madrid: Carlos III:n yliopisto.

13. Heikkilä, Tapio – Saukkoriipi, Janne – Ahola, Jari - Maiju, Seppälä 2019. On-line programming of robot skills. Anaheim: 2019 International Design Engineering Technical Conferences and Computer and Information in Engineering Conference IDETC/CIE2019.

14. Morrow, Daniel – Khosla, Pradeep 1997. Manipulation Task Primitives for Composing Robot Skills. Albuquerque: Proc of the 1997 IEEE Int'l Conf on Robotics and Automation. Saatavissa: https://www.ri.cmu.edu/pub_files/pub2/morrow_james_1997_1/morrow_james_1997_1.pdf . Hakupäivä 6.11.2019.

15. Wang, Wei – Johnston, Benjamin – Williams, Mary-Anne 2013. Recognition and Representation of Robot Skills in Real Time: A Theoretical Analysis. Berlin Heidelberg: Springer-Verlag GmbH.

16. Kemro K2 KeMotion PLC Robotic Programming manual V2.62.

17. Siciliano, Bruno - Khatib, Oussama 2008. Springer Handbook of Robotics. Berlin Heidelberg: Springer-Verlag.

LIITTEET

Liite 1: Robottikäden taidon, ”pick”, tilakoneen ohjelma.

robotarmcontrol.cpp <Select Symbol>

```
void RobotArmControl::robotArmControlCycle()
{
    if(m_task == "pick"){
        m_taskDone = false;
        //qDebug() << m_state;
        switch(m_state) {

        case 1:     //homing

            if(m_gripperControl->m_done && !m_wasFalse){
                m_gripperControl->homing(m_homingDirection);    //homing direction set to default value
                m_wasFalse = true;
            }
            if(m_wasFalse && m_gripperControl->m_done){
                m_state = 2;
                m_wasFalse = false;
            }
            break;

        case 2:     //homepose

            if(m_robotArmMovementControl->m_done && !m_wasFalse){
                m_robotArmMovementControl->moveRobotArm(0, 0, 0, 0, 0, 0);        //set coordinates to 0
                m_wasFalse = true;
            }
            if(m_wasFalse && m_robotArmMovementControl->m_done){
                m_state = 3;
                m_wasFalse = false;
            }
            break;

        case 3:     //prepos

            if(m_gripperControl->m_done && !m_wasFalse){
                m_gripperControl->prepos(60, 300);     //add 5mm to width and set speed to 300
                m_wasFalse = true;
            }
            if(m_wasFalse && m_gripperControl->m_done){
                m_state = 4;
                m_wasFalse = false;
            }
            break;

        case 4:     //approachpose

            if(m_robotArmMovementControl->m_done && !m_wasFalse){
                m_robotArmMovementControl->moveRobotArm(m_x, m_y, 120, m_a, m_b, m_c);      //set z to value 0
                m_wasFalse = true;
            }
            if(m_wasFalse && m_robotArmMovementControl->m_done){
                m_state = 5;
                m_wasFalse = false;
            }
            break;
```

2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Contr

robotarmcontrol.cpp <Select Symbol>

```
        case 5:     //grippingpose

            if(m_robotArmMovementControl->m_done && !m_wasFalse){
                m_robotArmMovementControl->moveRobotArm(m_x, m_y, m_z, m_a, m_b, m_c);
                m_wasFalse = true;
            }
            if(m_wasFalse && m_robotArmMovementControl->m_done){
                m_state = 6;
                m_wasFalse = false;
            }
            break;

        case 6:     //grasp

            if(m_gripperControl->m_done && !m_wasFalse){
                m_gripperControl->grasp(m_width, 50);       //set speed to 10
                m_wasFalse = true;
            }
            if(m_wasFalse && m_gripperControl->m_done){
                m_state = 7;
                m_wasFalse = false;
            }
            break;

        case 7:     //departurepose

            if(m_robotArmMovementControl->m_done && !m_wasFalse){
                m_robotArmMovementControl->moveRobotArm(m_x, m_y, 120, m_a, m_b, m_c);      //set z to value 0
                m_wasFalse = true;
            }
            if(m_wasFalse && m_robotArmMovementControl->m_done){
                m_state = 8;
                m_wasFalse = false;
            }
            break;

        case 8:     //homepose

            if(m_robotArmMovementControl->m_done && !m_wasFalse){
                m_robotArmMovementControl->moveRobotArm(0, 0, 0, 0, 0, 0);        //set coordinates to 0
                m_wasFalse = true;
            }
            if(m_wasFalse && m_robotArmMovementControl->m_done){
                m_state = 100;
                //m_taskDone = true;
                m_wasFalse = false;
            }
            break;

        case 100:
            m_task = "";
            m_state = 1;
            m_taskDone = true;
            break;
        }
    }
```

2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Contro

Liite 2: Vuokaavio