VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Toni Mahla

KOKOONPANON AUTOMATISOINTI

Tekniikan yksikkö
2017

VAASAN AMMATTIKORKEAKOULU
Kone- ja tuotantotekniikka

TIIVISTELMÄ

Tekijä	Toni Mahla
Opinnäytetyön nimi	Kokoonpanon automatisointi
Vuosi	2017
Kieli	suomi
Sivumäärä	50 + 3 liitettä
Ohjaaja	Lehtori, DI Mika Billing

Opinnäytetyön tarkoituksena oli tutkia kokoonpanon automatisointia Suomessa toimivalle yritykselle nimeltä Nordic Lights. Tavoite oli automatisoida 5-osaisen LED-valaisimen kokoonpano robottia käyttäen. Työn kuvaan kuului robottisolun suunnitteleminen, robotin ohjelmointi ja simulointi, robotin työkaluihin tulevien osien, jigien, palettien ja tasojen suunnittelu sekä CAD-mallien luominen.

Opinnäytetyössä suunniteltiin robottisolu, johon valittiin robotti ja kokoonpanoon tarvittavat robotin työkalut. Robottisolusta luotiin virtuaalinen malli, jota käyttäen luotiin ja simuloitiin ohjelma robotille. Tehtävä vaati perehtymistä erityyppisiin robotteihin, robotin työkaluihin sekä robotin ohjelmoimiseen ja simuloimiseen. Suurin osa opinnäytetyöstä osoittautui robotin ohjelman luomiseksi ja simuloinniksi. Kokoonpano oli tarkoitus testata koulussa olevalla robotilla ja robotin työkaluilla.

Opinnäytetyön tuloksena saatiin selvitettyä, miten ja millä komponenteilla LED-valaisimen kokoonpano on mahdollista automatisoida. Projekti jäi kesken johtuen jigien, palettien ja tasojen CAD-mallien viivästymisestä. Täten lopulliset ohjelmat oikeine liikeratoineen ja koulussa suoritettava kokoonpanon testaus jäi suorittamatta. Opinnäytetyö antoi yritykselle mallin, miten kokoonpanon automatisointi on mahdollista suorittaa, opetti paljon robottisolun suunnittelusta ja robotin ohjelmoinnista sekä antoi hyvin kokemusta työelämää varten.

Avainsanat automatisointi, kokoonpano, ohjelmointi, simulointi

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Mechanical engineering

ABSTRACT

Author	Toni Mahla
Title	Automatization of Part Assembly
Year	2017
Language	Finnish
Pages	50 + 3 Appendices
Name of Supervisor	Lecturer, M.Sc. Mika Billing

The purpose of this Bachelor's thesis was to investigate automatization of an assembly to a company based in Finland, named Nordic Lights. The task was to automate an assembly of a 5-part LED light using a robot. The objectives were designing a robot cell, robot programming and simulation, designing and creating CAD-models for parts to robot tools, jigs, palettes and levels.

A robot cell was designed where a robot and robot tools were chosen. From the robot cell was created a virtual model and it was used to create and simulate a program to robot. Thesis required comprehension to different type robots, robot tools and to robot programming and simulation. Thesis consisted mainly from programming and simulation. The purpose was to test the assembly with school's robot and robot tools.

As the results of thesis, it was resolved how and which parts to use for automatization of the LED light assembly. Thesis left unfinished due to retardation of jigs, palettes and levels CAD-models which were needed to finalize the program and manufacturing the parts used in testing the assembly. For the company, thesis gave a model how to automate the assembly, taught many aspects of robot cell designing and robot programming and gave valuable experience for labor tasks.

Keywords Assembly, automatization, programming, simulation

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

LIITELUETTELO

1 JOHDANTO

Opinnäytetyö käsittelee valaisimen kokoonpanon automatisointia Suomessa toimivalle yritykselle nimeltä Nordic Lights, joka suunnittelee ja valmistaa valaisimia vaativaan käyttöön, kuten maantie- ja maastoaloille. Opinnäytetyön tavoitteena on automatisoida 5-osaisen LED-valaisimen kokoonpano robottia käyttäen. Työn kuvaan kuuluu robottisolun suunnitteleminen, robotin ohjelmointi ja simulointi ABB RobotStudiolla, robotin työkaluihin tulevien osien, jigien, palettien ja tasojen suunnittelu sekä CAD-mallien luominen.

LED-valaisin koostuu piirikortista, piirikorttiin kiinnitettävästä liittimestä, kotelosta ja kahdesta lasista. Piirikorttia ja liitintä pitää paikallaan sisempi lasi, joka kiinnitetään kuudella ruuvilla koteloon. Ulompi lasi kiinnitetään koteloon liimalla ja neljällä lasissa olevalla lukituskoukulla.

Kokoonpanon kulku menee siten, että työntekijä asettaa osat kääntyville tasoille solun ulkopuolella, josta ne käännetään soluun. Tämän jälkeen robotti kokoonpanee valaisimen ja asettaa sen hihnakuljettimelle, josta se kuljetetaan solun ulkopuolelle valmiina työntekijän testattavaksi.

Robottisolun suunnittelussa keskeisimpiä vaiheita ovat robotin ja työkalujen valitseminen sekä palettien, jigien ja työkaluun kiinnitettävien osien CAD-mallien luominen. Kokoonpanossa käytettävät jigit ja paletit suunnittelee toinen opiskelija, joten niiden suunnittelua ei käsitellä tässä opinnäytetyössä.

Robotin ohjelmoinnissa keskeisimpiä vaiheita ovat virtuaalisen mallin luominen robottisolusta, jonka avulla luodaan kulkureitti robotille ja ehtojen määrittäminen käyttäen RAPID -ohjelmointikieltä. Virtuaalisen mallin avulla simuloidaan robotin liikettä ja toimintaa mahdollisissa poikkeavissa tilanteissa, kuten kappaleen puuttuessa paletilta.

Sovellusta testataan koulussa olevalla robotilla ja työkaluilla, jota varten mallinnetaan erilliset CAD-mallit ja tehdään erillinen ohjelma. Työkaluihin kiinnitettävät osat, paletti ja jigi valmistetaan koulun 3D-tulostimella.

2 TYÖN TAUSTA, TARKOITUS JA TAVOITTEET

Työ suoritetaan Nordic Lights nimiselle yritykselle ja tavoitteena on automatisoida ihmistyövoimalla tehtävä kokoonpano käyttäen robottia. Opinnäytetyön keskeisimpiä vaiheita ovat robottisolun suunnittelu ja robotin ohjelmointi.

2.1 Yritys ja kokoonpantava LED-valaisin

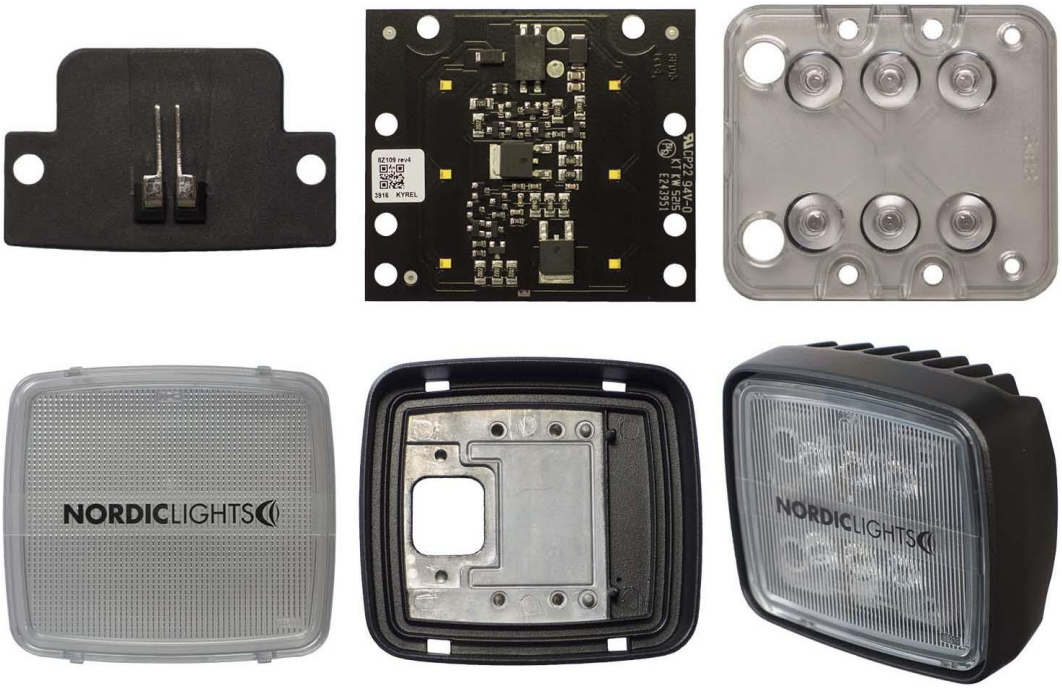
Nordic Lights on Suomessa toimiva yritys, joka suunnittelee ja valmistaa LED-, HID-, Xenon- ja halogeenityövaloja sekä Xenon -ajovaloja vaativaan maantie- ja maastoalan käyttöön. Yritys on vaativaan käyttöön tarkoitettujen varusteiden valaistusteknologian yksi johtavista maailmanlaajuisista toimittajista. /1/

Pääkonttori sekä tutkimus ja tuotekehitys sijaitsevat Suomessa. Lisäksi yrityksellä on konttoreita Kiinassa, Yhdysvalloissa, Saksassa ja Brasiliassa sekä laaja maailmanlaajuinen jälleenmyyntiverkosto. /1/

Tuotteita käytetään kaivos-, rakennus-, metsätalous-, maanviljely- ja kuormaussekä trukki-, perävaunu- ja pelastusalan koneissa ja ajoneuvoissa. Yritys toimittaa valaistusratkaisuja monille maailman johtaville raskaaseen käyttöön tarkoitettujen koneiden valmistajille. /1/

Kokoonpantavan LED-valaisimen malli on KL2001 ja se on erityisesti tarkoitettu käytettäväksi vaativissa ympäristöissä ja tehtävissä. Yleisin käyttökohde on työalueen valaistus koneen ympärillä, johon lamppu on kiinnitettynä. LED-valaisinta käytetään työajoneuvoissa kaivos-, rakennus-, metsä-, maanviljelys- ja pelastusteollisuudessa sekä rekoissa ja perävaunuissa. /2/

LED-valaisin koostuu piirikortista, piirikorttiin kiinnitettävästä liittimestä, kotelosta ja kahdesta lasista. Piirikorttia ja liitintä pitää paikallaan sisempi lasi, joka kiinnitetään kuudella ruuvilla koteloon. Ulompi lasi kiinnitetään koteloon liimalla ja neljällä lasissa olevalla lukituskoukulla.



Kuva 1. Kokoonpantava LED-valaisin.

2.2 Tavoitteet ja tarkoitus

Opinnäytetyön tavoitteena on automatisoida 5-osaisen LED-valaisimen kokoonpano robottia käyttäen. Työn kuvaan kuuluu robottisolun suunnitteleminen, robotin ohjelmointi ja simulointi ABB RobotStudiolla sekä palettien ja tasojen CAD-mallien mallintaminen.

Tarkoituksena on, että LED-valaisimen kokoonpanon kulku menee siten, että työntekijä asettaa osat kääntyville tasoille solun ulkopuolella, josta ne käännetään soluun. Tämän jälkeen robotti kokoonpanee valaisimen ja asettaa sen hihnakuljettimelle josta se kuljetetaan solun ulkopuolelle valmiina työntekijän testattavaksi.

3 ROBOTTISOLUN SUUNNITTELU

Robottisolua suunniteltaessa tulee ottaa huomioon tila, johon solun tulee mahtua, sijainti, josta osia syötetään soluun, robotin ulottuvuus sekä valmiina olevien laitteiden koko ja sijainti. /3-5/

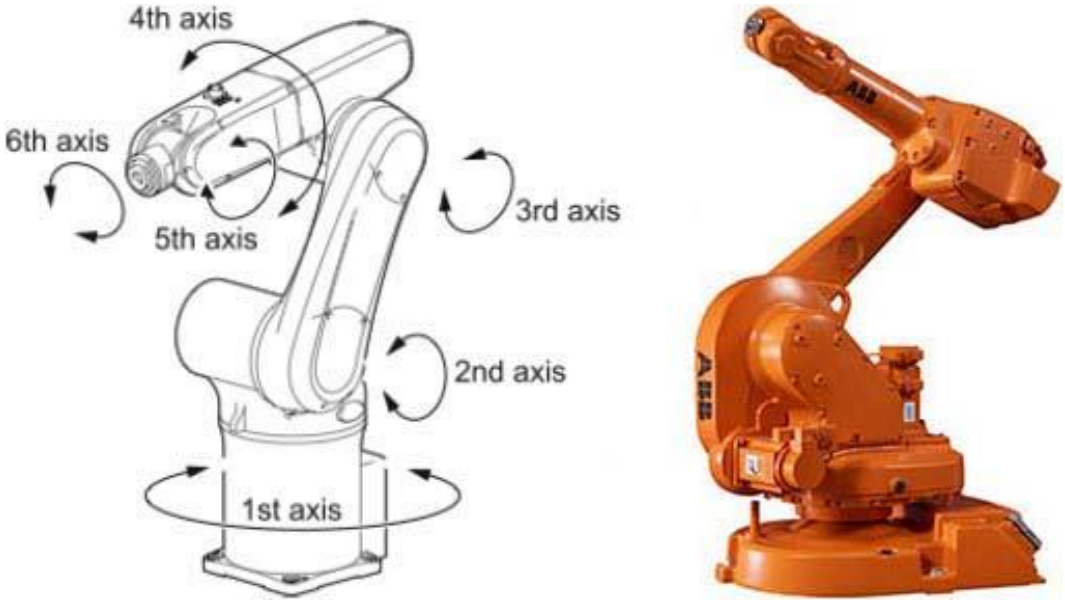
Kun nämä vaatimukset ovat saavutettu, tulisi pyrkiä tekemään kokoonpanosyklistä mahdollisimman sujuvan ja robottisolun viemä tila tulisi olla mahdollisimman pieni valitsemalla kuljettimet palettien koon perusteella sekä minimoimalla robotin ja palettien sekä jigien välissä oleva tyhjä tila. /3-5/

Robottisolua suunniteltaessa ensimmäisenä valittiin kokoonpanoon soveltuva robotti, tämän jälkeen valittiin tehtävään soveltuvat työkalut. Jigien- ja palettien mallintaminen sekä paleteille soveltuvien käännettävien tasojen valitseminen oli annettu tehtäväksi toiselle opiskelijalle, joten niiden suunnittelua ei käydä tässä opinnäytetyössä läpi. Kun robotti, työkalut osineen, jigit, paletit ja kuljettimet oli valittu/mallinnettu, määritettiin tasojen korkeudet, koot ja sijainnit, jolla kokoonpano onnistuu siten, että robotin ulottuvuus riittää ja että robotti työkaluineen pääsee liikkumaan vapaasti ilman törmäyksiä.

3.1 Robotin valitseminen

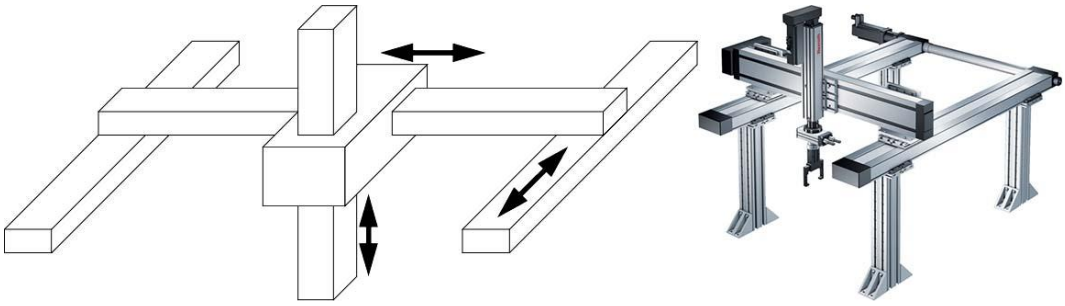
Robottia valittaessa ensimmäisenä tulee valita minkä tyyppinen robotti soveltuu tehtävään parhaiten. Yleisimpiä teollisuusrobotteja ovat kiertyvänivelinen robotti, napakoordinaatistorobotti, rinnakkaisrakenteinen robotti, SCARA-robotti, suorakulmainen robotti, sylinterirobotti ja yhteistoimintarobotti. /6/

Kiertyvänivelisessä robotissa on ihmiskäsivartta muistuttava rakenne. Vapausasteita on yleensä neljä tai kuusi ja ne ovat kiertyviä. Tämä rakenne mahdollistaa työkalun kääntyvyyden kaikkiin mahdollisiin kulmiin. Kiertyvänivelisessä robotissa on suuri ulottuvuus, mutta pieni kuormankantokyky. Kiertyvänivelinen robotti on soveltuvuudeltaan monipuolisin robottirakenne. /7-8/



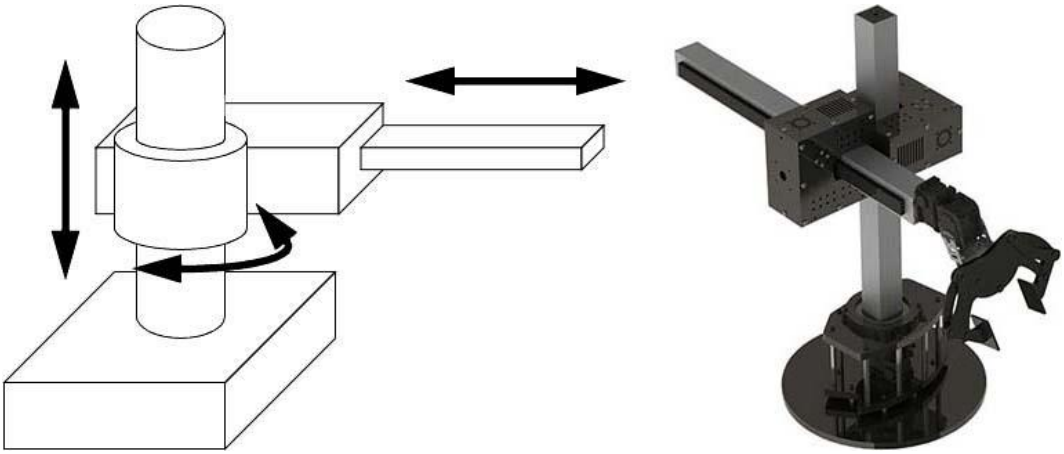
Kuva 2. Kiertyvänivelinen robotti.

Suorakulmaisessa robotissa kolme ensimmäistä vapausastetta ovat lineaarisia ja seuraavat vapausasteet ovat kiertyviä. Tyypillisintä kolmen lineaarisen vapausasteen robottia kutsutaan portaalirobotiksi ja sen rakenne on tuettu nurkista palkeilla. /7-8/



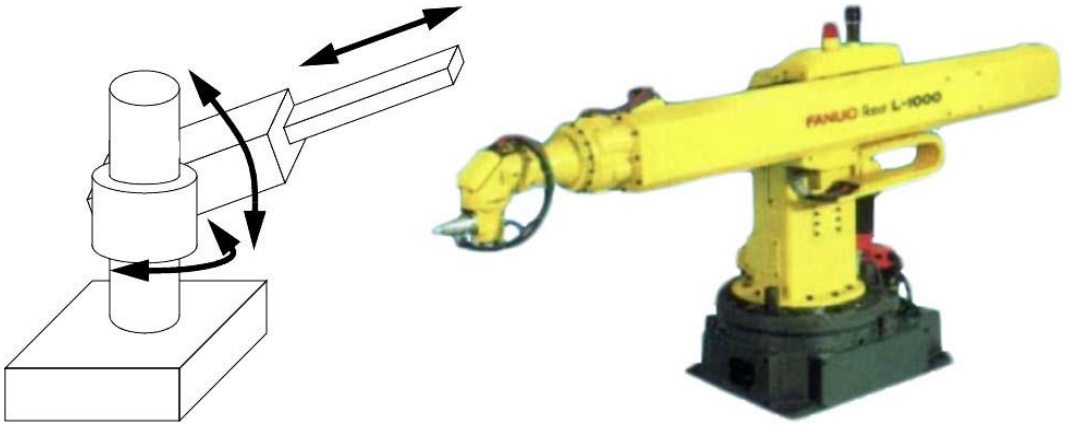
Kuva 3. Portaalirobotti.

Sylinterirobotissa kolmen lineaarisen vapausasteen lisäksi on yksi kiertyvä vapausaste, joka kääntää koko rakennetta. Suorakulmaisia robotteja käytetään logistiikka- ja varastosovelluksissa, kevyessä työstössä, työstökoneiden panostuksessa sekä 3D-tulostimissa. /7-8/



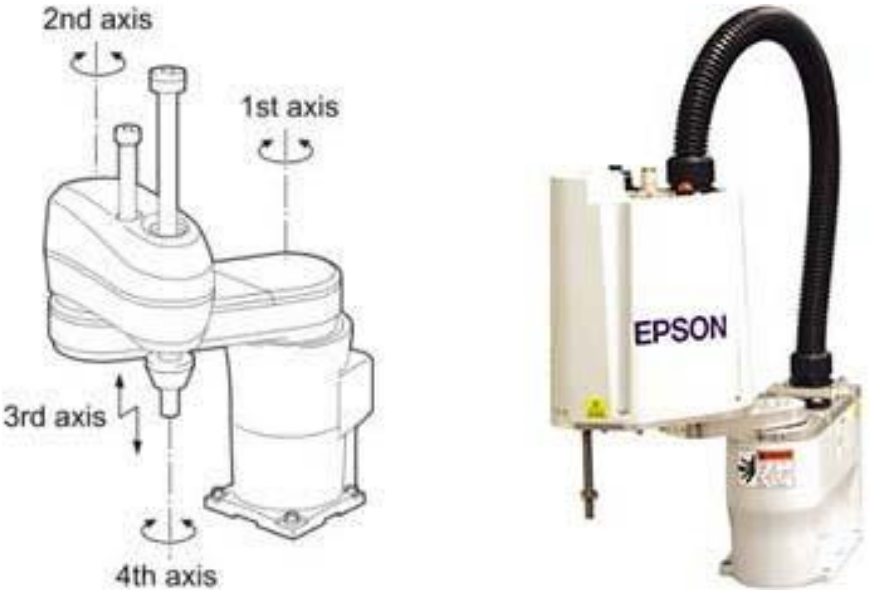
Kuva 4. Sylinterirobotti.

Napakoordinaatistorobotissa on kaksi kiertyvää vapausastetta muiden vapausasteiden ollessa lineaarisia. Ensimmäinen kiertyvä vapausaste kääntää koko rakennetta ja toinen vapausaste kääntää käsivartta pystysuunnassa. Napakoordinaatistorobottia käytetään erikoissovelluksiin. /7-8/



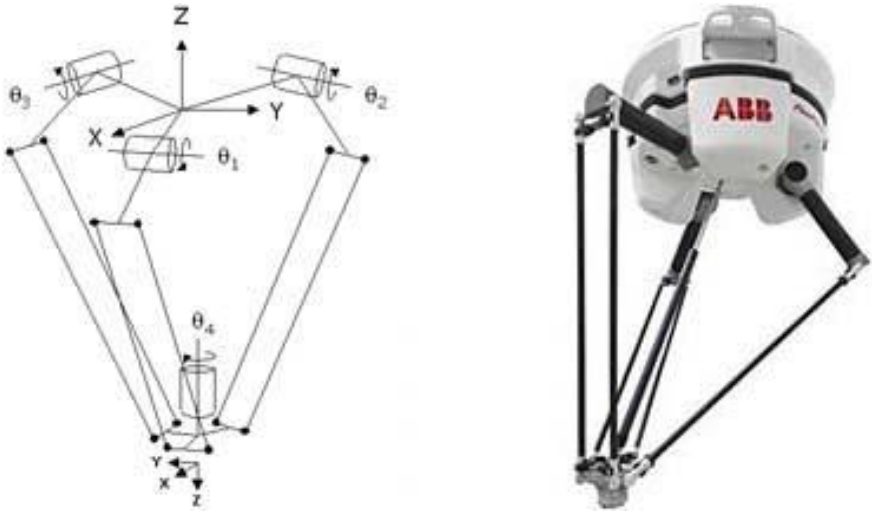
Kuva 5. Napakoordinaatistorobotti.

SCARA-robotissa (Selective Compliance Assembly Robot Arm) on useimmiten neljä vapausastetta, joista yksi on lineaarinen ja loput ovat kiertyviä. Tämä rakenne mahdollistaa suuren pystysuuntaisen jäykkyyden. Rakenne voidaan toteuttaa myös siten, että kaikki käyttömoottorit ovat jalustassa. SCARA-robotteja käytetään pienikokoisen mekaniikan kokoonpanossa, ruuvauksessa ja piikiekkojen käsittelyssä. /7-8/



Kuva 6. SCARA-robotti.

Rinnakkaisrakenteisessa robotissa on suljettu kinemaattinen rakenne. Työkalu on kolmen lineaarisen vapausasteen varassa, jolloin rakenteesta tulee tukeva. Rinnakkaisrakenteinen robotti on asennettu roikkumaan telineestä, liikenopeus on suuri, mutta ulottuvuus on rajoittunut. Tämän tyyppisiä robotteja käytetään poimintasovelluksissa ja työstötehtävissä. /7-8/



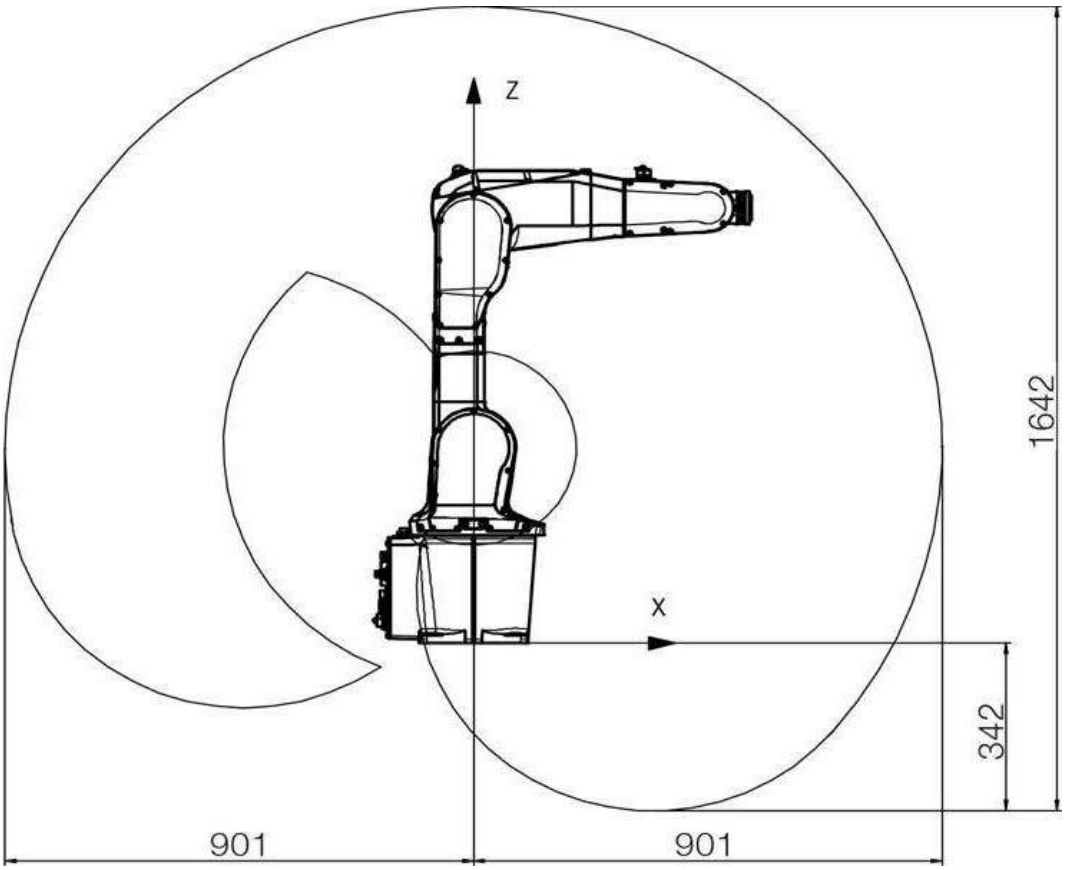
Kuva 7. Rinnakkaisrakenteinen robotti.

Yhteistoiminta robotti eroaa tavanomaisesta teollisuusrobotista siten, että voiman mittaus ja muotoilu on toteutettu siten, että se on ihmiselle turvallisempi, se pystyy toimimaan ihmisen kanssa samassa työtilassa, ohjelmointi on yksinkertaisempaa ja nopeampaa, asennustilavaatimukset ovat pienempiä, eivätkä useimmiten vaadi turva-aitoja. Yhteistoiminta robotit ovat vastaavan rakenteen ja koon omaavia teollisuusrobotteja hitaampia. Niillä on pienempi toistotarkkuus ja kantokyky. Yhteistoiminta robotteja käytetään toimistosovelluksista teollisuussovelluksiin. Eräitä teollisuussovelluksia ovat poimintasovellukset, kokoonpanotehtävät ja osien paikoittaminen konepalvelusovelluksissa. /7-9/

Toisena tulee määrittää kuorma, joka robotin tulee jaksaa kantaa. Kantokyvyn (payload) on oltava suurempi kuin kannettavan kuorman, joka koostuu kappaleesta ja robotin kärkeen kiinnitetystä työkalusta osineen. Kantokyky on määritetty robotin viimeisen akselin päästä ja se on useimmiten annettu massan yksikkönä (kg/lbs). /6/

Kolmantena tulee määrittää vapausaste, jolla robotin on kyettävä liikkumaan. Vapausaste on riippuvainen robotin akselien lukumäärästä ja tyypistä (kiertyvä, lineaarinen). Jos robottia tullaan myöhemmin käyttämään muissa tehtävissä joita ei valintahetkellä tiedetä, on suotavaa valita robotti, jossa on akseleita tarpeeksi suuri määrä joka mahdollistaa laajan soveltuvuuden eri tehtäviin. Valmistajat nimeävät useimmiten nivelet nousevassa järjestyksessä robotin kannasta lähtien. /6/

Neljäntenä tulee määrittää tarvittava ulottuvuus robotille. Maksimaalinen vertikaalinen ulottuvuus on matka alimmasta korkeimpaan pisteeseen, johon robotti yltää. Maksimaalinen horisontaalinen ulottuvuus on matka robotin kannan keskipisteestä kauimpaan pisteeseen horisontaalisessa suunnassa, johon robotti yltää. /6/



Kuva 8. Robotin ulottuvuus. Kuvassa ABB IRB 1200-5/0.9.

Viidentenä tulee määrittää vaadittava liikkeen toistotarkkuus. Liikkeen toistotarkkuudella tarkoitetaan robotin kykyä toistaa sama sijainti jokaisella syklillä, jonka robotti suorittaa. Virheen suunta määrittyy robotin vapausasteiden perusteella, esim. kaksiulotteisen liikeradan omaavan robotin virhe toistojen välillä on kaksiulotteisessa suunnassa, kun taas kolmiulotteisessa robotissa virhe toistojen välillä voi olla mihin tahansa suuntaan. /6/

Kuudentena tulee määrittää aika, jona robotin tulee saada sykli suoritettua eli kuinka nopea robotin tulee olla. Määritettäessä robotin nopeutta tulee ottaa huomioon robotin kiihtyvyys ja maksiminopeus. Valmistaja ilmoittaa maksiminopeuden nopeuden yksikössä (m/s), liikeyksikön (motion unit) useimmiten astetta/sekunti ja maksimikiihtyvyyden kiihtyvyyden yksikössä (m/s^2). /6/

Seitsemäntenä tulee määrittää massa, jonka robotti saa korkeintaan painaa. On otettava huomioon mihin robotti istutetaan ja kuinka suuri on alustan kuormankantokyky. /6/

Viimeisenä tulee määrittää tiiviys (IP-luokitus), joka vaaditaan robotilta. IP-luokitus on kansainvälinen standardi joka määrittää sähkölaitteiden ja laitekoteloiden tiiviyden. Vaadittava tiiviys on riippuvainen ympäristöstä, jossa robotti toimii. Valmistaja usein tarjoaa samaa mallia robotista eri IP-luokituksilla. /6, 10/

Kyseisessä sovelluksessa robotilta vaaditaan kolmiulotteista kiertoa, joten suotavin rakenne robotille on kiertyvänivelinen. Kokoonpanossa tehtävä ruuvaus on mahdollista tehdä samalla kiertyvänivelisellä robotilla vaihtamalla työkalua kesken syklin tai erillisellä robotilla, esim. SCARA-robotilla. Erillisen työkalun vahvuuksia ovat edullisuus ja yksinkertaisempi ohjelmointi. Erillisen ruuvaukseen tarkoitetun robotin vahvuuksia taas ovat nopeus ja kompakti koko. Asiaa yritykseltä kysyttäessä he valitsivat erillisen työkalun.

Toinen merkittävä tekijä robottia valitessa on kokoonpanoon tarvittava ulottuvuus, jonka pääasiassa määrittää komponentteja paikalla pitävät palettien koot. Kääntyvien tasojen leveydet valitaan palettien leveyden perusteella. Kääntyvien tasojen leveys määrittää minimietäisyyden jigin sekä paletin- että palettien välillä. Palettien ja jigien korkeus suhteessa robottiin määrittää minimietäisyyden robotin ja palettien välillä, koska robotin säteensuuntainen ulottuvuus muuttuu korkeuden muuttuessa. Kappaleiden määrät paleteilla tulevat olemaan 15kpl (kotelo, ulkolasi) ja 30kpl (liitin, piirilevy ja sisälasi), täten palettien koot tulevat olemaan arviolta 30cm leveitä. Ottaen huomioon nämä seikat, robotilta vaadittu ulottuvuus on arviolta 0,6-0,8m.

Kolmas merkittävä seikka LED-valaisimen kokoonpanossa on toistotarkkuus. Kokoonpanossa suurinta tarkkuutta vaatii liittimen liittäminen piirilevylle. Ottamalla huomioon piirilevyssä olevan vastakappaleen viisteiden koon, toistotarkkuus tulisi olla arviolta ±0,2mm.

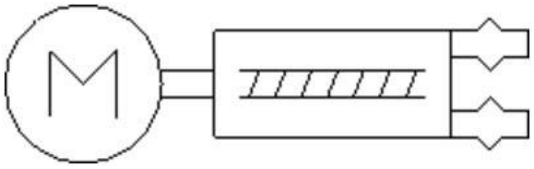
LED-valaisimen paino on n. 382g ilman liimaa ja ulkolasin painaminen ei vaadi suurta voimaa. Tehdas ei asettanut tarkkaa vaatimusta sykliajalle ja robotti tullaan asentamaan tehtaan betoniselle lattialle, jossa ei vaadita suurta tiiviyttä robotilta. Nämä seikat huomioiden robottia valitessa sen kantokyky, nopeus, paino ja IP-luokitus eivät ole robottia valittaessa ensisijaisia ominaisuuksia. Kyseiset kriteerit täyttävä robotti löytyi valmiina koululta, ABB IRB 1200-5/0.9 /11/.

Kuva 9. ABB IRB 1200-5/0.9.

3.2 Työkalun valitseminen

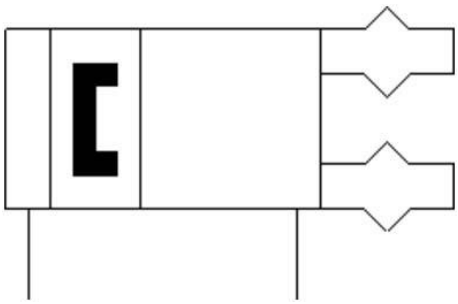
Työkalua valittaessa ensin tulee määrittää työkalun tyyppi, joka soveltuu parhaiten suoritettavaan tehtävään. Teollisuusroboteissa käytettäviä työkalujen eri tyyppejä ovat mm. tarttujat, joita on elektronisia, pneumaattisia ja magneettisia sekä hitsaukseen, maalaukseen ja ruuvaukseen käytettävät työkalut. /12/

Elektronisissa tarttujissa ote kappaleeseen toteutetaan sormien avulla. Sormien liikettä ohjataan sähkömoottoreilla, joita ohjataan robotin kontrollerilla. Sormien liike voi olla lineaarinen tai kiertyvä. Elektronisen tarttujan etuja ovat mm. sormen aseman portaaton ohjaus, otteen tunnistus, otteen voiman ja nopeuden ohjaus, huoltovapaus, matala energiankulutus ja puhtaus. Elektronisia tarttujia käytetään mm. kokoonpano-, poiminta- ja käsittelysovelluksissa sekä konepalveluissa. /12/



Kuva 10. Elektronisen tarttujan toimintaperiaate yksinkertaistettuna.

Pneumaattisissa tarttujissa sormien liikettä ohjataan paineilman avulla. Robotin kontrolleri ohjaa solenoidiventtiiliä liikuttaen paineilman avulla tarttujassa olevaa mäntää, joka on yhdistetty tarttujan sormiin. Sormien liike voi olla lineaarinen tai kiertyvä. Pneumaattisessa tarttujassa sormet ovat joko täysin auki tai täysin kiinni, otteen voimaa ja nopeutta voidaan säätää rajoitetusti paineilman voimakkuutta muuttamalla. Paineilmajärjestelmä, jolla liikutetaan tarttujan sormia, vaatii jatkuvaa huoltoa, eikä useimmissa pneumaattisissa tarttujissa ole sisäänrakennettua otteen tunnistusta. Pneumaattista tarttujaa käytetään elektronisen tarttujan sijasta, kun vaaditaan edullista hintaa suhteessa tartuntavoimaan. /12/



Kuva 11. Pneumaattisen tarttujan toimintaperiaate yksinkertaistettuna.

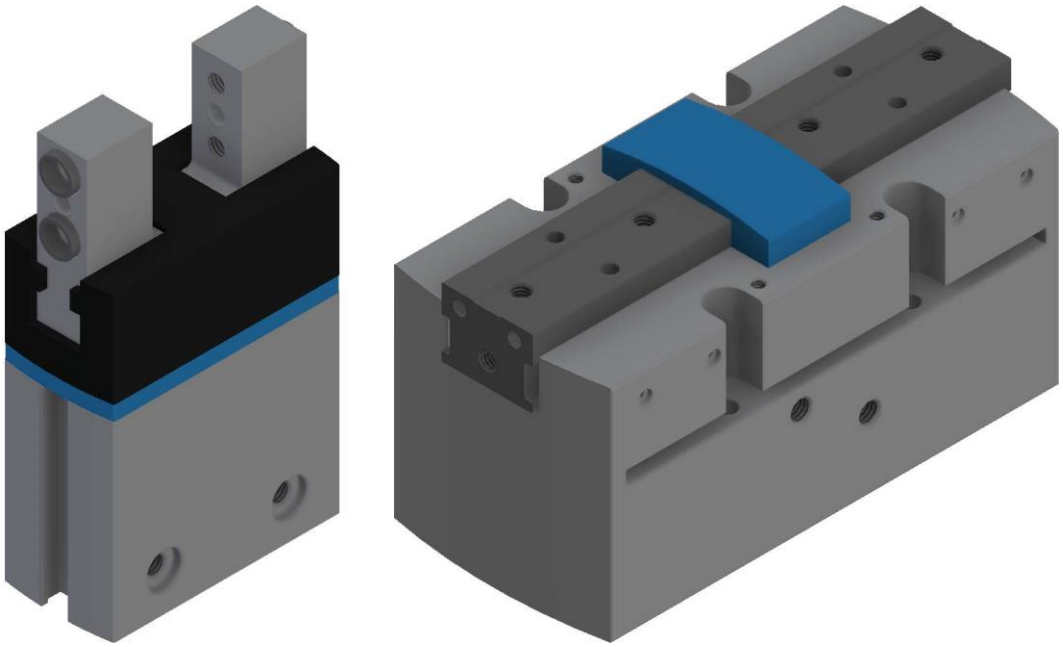
Imukuppitarttujissa ote kappaleeseen tapahtuu imukupin ja kappaleen väliseen tilaan syntyvän alipaineen avulla. Alipaineen synnyttää solenoidiventtiili, joka kytketään paineilmajärjestelmään. Imukuppitarttujat ovat anteeksiantavia pystysuuntaisessa tarkkuudessa joustavan materiaalinsa ja mahdollisen pystykompensaation ansiosta jättävät jälkiä tiettyihin pintoihin, ovat hinnaltaan edullisia, vaativat jatkuvaa huoltamista paineilmajärjestelmän vuoksi ja tartuttavan materiaalin on oltava sileä ja reiätön. Tyypillisiä käyttökohteita ovat mm. kokoonpano ja poimintasovellukset. /12/

Magneettisissa tarttujissa ote kappaleeseen toteutetaan magneettisen voiman avulla, joka syntyy magneetin ja kappaleen välille. Magneettisia tarttujia on kahta tyyppiä: sähkö- ja kestomagneettinen. Sähkömagneettisessa tarttujassa voima magneetin ja kappaleen välille synnytetään sähkövirran avulla. Kestomagneettisessa magneettikentän aiheuttama voima on päällä aina, kappale irrotetaan käyttäen työkalussa sijaitsevaa tappia, joka työntää kappaleen irti magneetista. Magneettisen tarttujan etuja ovat: ne tarvitsevat vain yhden pinnan tarttuakseen, tarttumisnopeus on suuri, hyvä soveltuvuus erimuotoisille- ja kokoisille kappaleille, tartuttavan pinnan ei tarvitse olla reiätön ja huoltovapaus. Huonoja puolia ovat heikko pito nopeissa liikkeissä, pienikin määrä öljyä heikentää pitoa merkittävästi, kestomagneetti kerää metallijäystettä, kappaleet voivat pysyä magnetisoituneena ja voidaan käyttää ainoastaan ferromagneettisiin materiaaleihin. /12/

Työkalun tyypin valintaan vaikuttavat käsiteltävän kappaleen koko, -muoto, -paino ja -pinnan tyyppi sekä vaadittu syklin nopeus, toistotarkkuus ja ympäristölliset rajoitteet. Kappaleen koko, muoto, paino ja pinta tulee ottaa huomioon valittaessa elektronisen/pneumaattisen tarttujan avautumaa sekä imukupin/magneetin halkaisijaa. /12/

Sykliaika määrittää kiihtyvyydet, joilla robottia ajetaan ja nopeuden jolla tarttujan on saatava ote kappaleesta. Työkalu tulee valita siten, että robotin voima riittää tuottamamaan tarpeeksi suuren kiihtyvyyden käsiteltävän kappaleen ja työkalun yhteismassalla ($F_{robotti} \geq F_{kappale} + F_{työkalu}$). Sovelluksen tyyppi määrittää tarkkuuden, jota työkalulta vaaditaan ja ympäristö, missä robotti operoi määrittää vaadittavan puhtauden. /12/

LED-valaisimen kokoonpanossa liittimen kiinnittämiseen soveltuu parhaiten tarttujatyyppinen työkalu, muiden osien kokoonpanemiseen imukuppitarttuja ja ruuvaukseen ruuvaustyökalu. Ottaen huomioon liittimen koon ja muodon, tarttujan leukojen avautuman ei tarvitse olla suuri ja liittimen paino, pinta, syklin nopeus sekä ympäristö eivät aseta rajoitteita tarttujalle. Yritykselle tulevaan layoutiin tarttujatyyppiseksi työkaluksi valittiin Festo DHPS-16-A ja liiketunnistimiksi SMT-8G. Koulussa tehtävässä simulaatiossa työkaluna käytetty malli on HGPP-20-A /13-15/.



Kuva 12. Festo- DHPS-16-A (vas.), HGPP-20-A (oik.).

Imukuppitarttujiksi valittiin Festo ESG-10-SNA ja ESG-15-SNA antistaattisilla imukupeilla. Tässä ensimmäisen kriteerin asettivat piirilevyn- ja sisälasin tartuntakelpoisen pinnan koko, joiden perusteella imukuppien koko valittiin. Toisen kriteerin asetti kootun valaisimen paino (n. 382g ilman liimaa), imukuppien yhteenlaskettu voima tulee olla tarpeeksi suuri, jotta LED-valaisinta voidaan liikuttaa tarpeeksi suurella kiihtyvyydellä ilman, että se irtoaa. Imukuppien pitovoimat nimellispaineella (-0,7 bar) ovat 3,9N (Ø 10mm) ja 8,5N (Ø 15mm). Maksimaalinen pystysuuntainen kiihtyvyys, jolla kappale pysyy imukupissa kiinni, saadaan laskettua kaavalla (1).

$F_{10} + F_{15} = F_a + F_g$, jossa F_{10} = Ø 10mm imukupin pitovoima (1)

F_{15} = Ø 15mm imukupin pitovoima

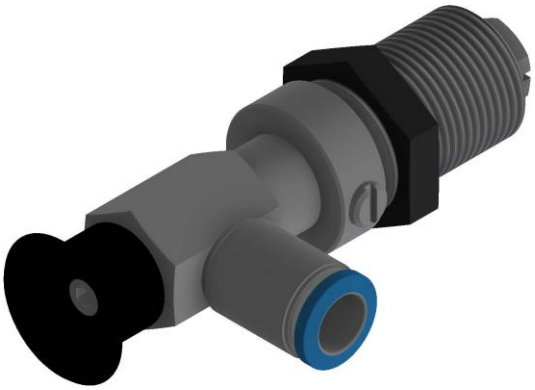
F_a = robotin kiihtyvyyden aiheuttama voima

F_g = putoamiskiihtyvyyden aiheuttama voima

Täten kiihtyvyys

$a = (F_{i10} + F_{i15} - F_g)/m = (3,9 + 8,5 – 0,382*9,81)(kg*m/s^2)/(0,382kg) \approx 22,7m/s^2$

Saatu kiihtyvyys riittää mainiosti kyseiseen tehtävään ja muut seikat eivät aseta lisää vaatimuksia imukuppitarttujille. /16-17/

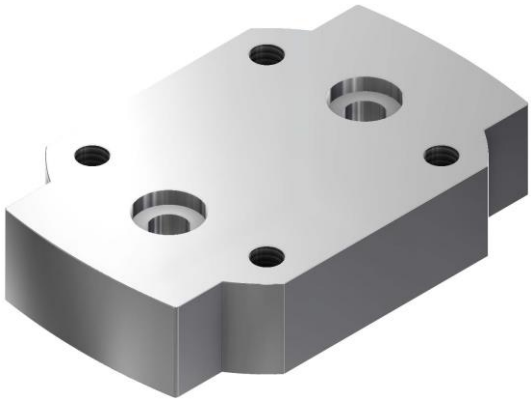


Kuva 13. Festo ESG-15-SNA.

3.2.1 Työkaluun tulevien osien suunnittelu

Työkaluun on suunniteltava kiinnityslevy robotin ja työkalun välille, sormet joilla liitetään liitin piirilevyyn, kappale, jolla painetaan lasi paikalleen ja imukuppiteline, johon imukuppitarttujat kiinnitetään. Kiinnityslevy tullaan koneistamaan alumiinista ja loput osat 3D-tulostamaan ABS-muovista. CAD-mallit luodaan Autodesk Inventor -ohjelmalla.

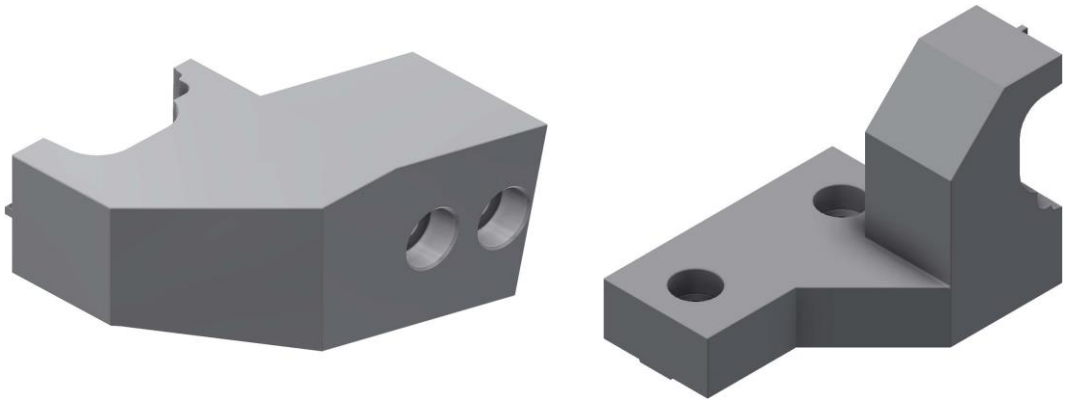
Kiinnityslevyä suunniteltaessa on otettava huomioon käsiteltävien kappaleiden massa, sovelluksessa vaadittava tarkkuus ja työkalun asento suhteessa robotin kärkeen, jolla tehtävä onnistuu optimaalisimmin. Yritykselle tulevassa layoutissa robotin ja työkalun välille tuleva kiinnityslevy suunniteltiin siten, että työkalu tulee robotin kärkeen keskitetysti ja työkalun leuat robotin kärjen kanssa samassa linjassa. Materiaaliksi valittiin alumiini jäykkyytensä ja riittävän keveytensä vuoksi. Kappale valmistetaan koneistamalla. Kiinnitys robotin kärkeen tehdään neljällä M4-ruuvilla, työkalu kiinnitetään välipalaan kahdella M4-ruuvilla ja ulkopinnan muoto projisoitiin työkalusta.



Kuva 14. Kiinnityslevy.

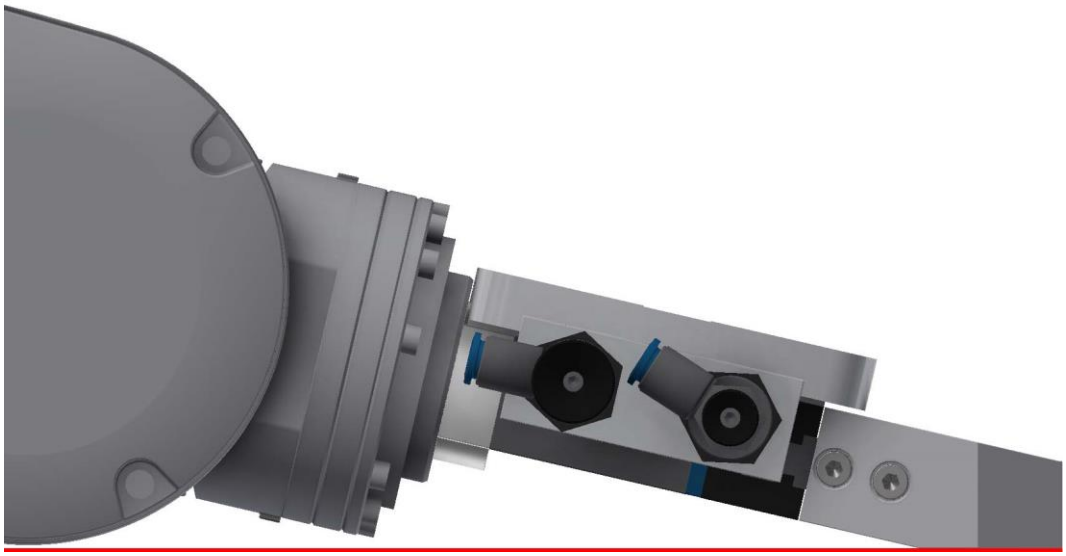
Robotin leukoihin kiinnitettävien sormien kärkiä suunniteltaessa tulee ottaa huomioon käsiteltävän kappaleen muoto, koko, massa ja sovelluksessa vaadittava tarkkuus. Sormien muoto projisoitiin liittimen pinnasta, johon ote muodostetaan. Materiaaliksi valittiin ABS-muovi, koska se ei naarmuta pintoja pehmeytensä vuoksi, mahdollistaa valmistamisen 3D-tulostamalla ja se on riittävän jäykkää kyseiseen tehtävään. Sormet kiinnitetään työkalun leukoihin kahdella M4-ruuvilla. Yritykselle tulevat sormet kohdistetaan kohdistussokilla ja koulussa käytettävissä sormissa kohdistus tapahtuu työkalun leuoissa olevien urien avulla.

Sormien päihin mallinnettiin suorakulmaisen särmiön muotoiset kohoumat, joiden avulla työkalukoordinaatisto saadaan paikoitettua tarkasti robottia ohjelmoitaessa. Nämä kohoumat poistetaan paikoittamisen jälkeen, mikäli halutaan, että tarttujan sormet pääsevät sulkeutumaan kokonaan mahdollistaen sormien välissä olevan kappaleen tunnistuksen. Sormet suunniteltiin siten, että liittimestä saa otteen kohdistustappien ollessa paikoillaan, otteen muodostuttua liittimeen kohoumien välille jää n. 0,2mm välys riippuen 3D-tulostimen tarkkuudesta.



Kuva 15. Vasemmalla yrityksen layoutissa käytettävä sormi ja oikealla koulussa käytettävä.

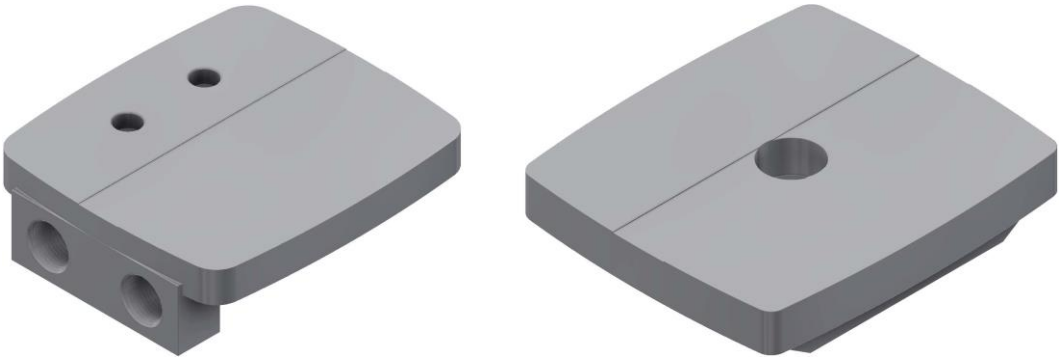
Yritykselle tulevassa layoutissa sormien kärjet suhteessa robotin kärkeen on mallinnettu siten, että liitintä poimittaessa sormet ovat alempana kuin robotin ylempi käsivarsi ja työkalu (Kuva 16). Tämä mahdollistaa osien poimimisen paletin perältä ilman, että robotin käsivarsi törmää siihen.



Kuva 16. Sormien kärkien kulma. Poimittaessa liittimen pohja on samansuuntainen punaisen viivan kanssa.

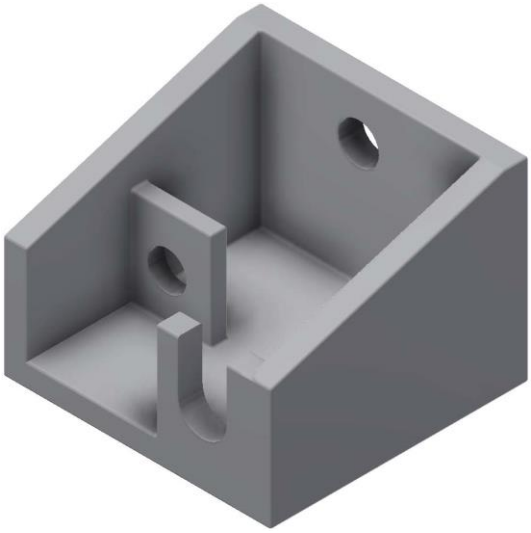
LED-valaisimen kokoonpanossa ulkolasi painetaan koteloon. Tämä vaatii erillisen kappaleen, joka kiinnitetään tarttujatyökalun runkoon. Tästä eteenpäin kyseistä

kappaletta kutsutaan lasityökaluksi. Suunnittelussa on otettava huomioon, että painettaessa lasi ei naarmuunnu ja että paine jakautuu tasaisesti lasin pinnalle. Materiaaliksi valittiin ABS-muovi, koska se ei naarmuta pintoja pehmeytensä vuoksi, mahdollistaa valmistamisen 3D-tulostamalla ja se on riittävän jäykkää kyseiseen tehtävään.



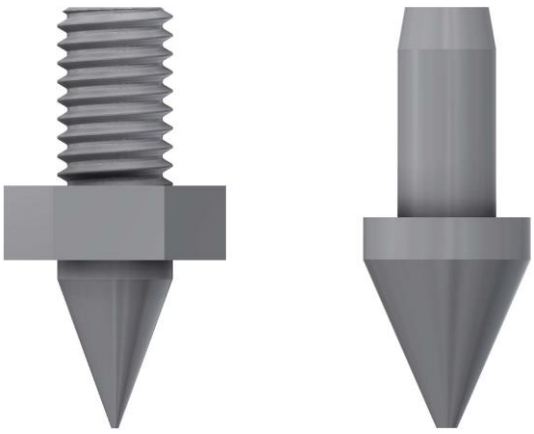
Kuva 17. Yrityksessä käytettävä kappale vasemmalla ja koulussa käytettävä oikealla.

Teline, johon imukuppitarttujat kiinnitetään, tulee suunnitella siten, että tarttujatyökalun leuat eivät ota kiinni telineeseen avautuessaan ja että imukuppitarttujat eivät ole tiellä käytettäessä tarttujatyökalua tai painettaessa lasia paikalleen. Suunnattaessa imukuppitarttujia on huomioitava robotin liikkuvuus ja solussa olevien objektien sijainti, jotta vältytään törmäyksiltä. Yritykselle tuleva teline mallinnettiin lasityökaluun (Kuva 17) ja koulussa käytettävä teline kiinnitetään robotin ja tarttujatyökalun välissä olevaan piirtimelle tarkoitettuun telineeseen. Materiaaliksi valittiin ABS-muovi, koska se ei naarmuta pintoja pehmeytensä vuoksi, mahdollistaa valmistamisen 3D-tulostamalla ja se on riittävän jäykkää kyseiseen tehtävään.

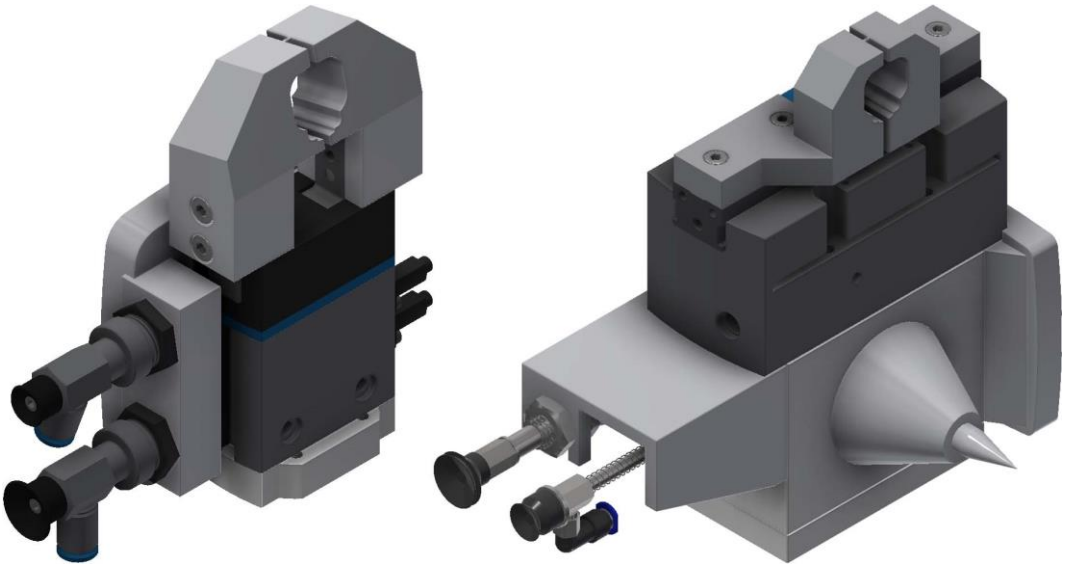


Kuva 18. Koulussa käytettävä teline imukuppitarttujille.

Lasityökalulle ja imukuppitarttujille luotiin työkalukoordinaatiston paikoitusta helpottavat tapit, jotka valmistetaan ABS-muovista 3D-tulostamalla. Paikoitettaessa työkalukoordinaatistoja lasityökalun paikoitustappi kiinnitetään samoihin kierteisiin kuin lasityökalu, joten siihen mallinnettiin kierteiden geometria kolmiulotteisena. Tämä mahdollistaa kierteiden 3D-tulostamisen. Imukuppitarttujien paikoitustappi kiinnitetään imukuppitarttujan päässä painelinjassa olevaan putkeen.



Kuva 19. Lasityökalun paikoitustappi vasemmalla ja imukuppitarttujien oikealla (tapit eivät ole samassa mittasuhteessa).



Kuva 20. Yritykselle tulevat työkalut osineen vasemmalla ja koulussa käytettävät oikealla.

4 ROBOTIN OHJELMOINTI

Robotin ohjelmointi tehtiin ABB RobotStudiolla, joka mahdollistaa robotin ohjaamisen ja ohjelmoinnin PC:llä sekä simuloinnin virtuaalikontrollerilla, joka on identtinen kopio robotin oikean kontrollerin ohjelmistosta /18/. Simuloinnin avulla on mahdollista analysoida toteutettavan sovelluksen käyttäytymistä virtuaalisesti ennen laitteiston hankkimista, mikä mahdollistaa sulavan siirtymisen konseptista toteutukseen. Suunnitteluvaiheessa simuloinnin ansiosta on mahdollista tarkastaa robotin ulottuvuuden riittävyys, joka helpottaa sovelluksessa käytettävän robotin valitsemista työkaluineen. Ohjelmoitaessa simulointi nopeuttaa reitin ja siinä käytettävän robotin konfiguraation määrittämistä sekä auttaa törmäyksien ehkäisemisessä. Täten robottisolun valmistukseen menevää aikaa, kustannuksia ja onnettomuusriskiä saadaan alennettua. /19-20/

Tässä luvussa käsitellään ohjelmoinnin jokainen päävaihe läpi kommentoiden mitä on tehty ja mitä seikkoja tulee ottaa huomioon ohjelmoinnissa. Robottisolun suunnitteluvaiheessa tehtävää sovellusta voidaan simuloida eri roboteilla ja työkaluilla helpottaen valitsemaan mahdollisimman optimaaliset komponentit kyseiseen sovellukseen. Ennen lopullisen ohjelman luomista sovelluksessa käytettävä robotti työkaluineen, käsiteltävät osat jigeineen ja paletteineen tulee olla asennettuna paikalleen ja niistä on oltava CAD-mallit, joita käytetään ohjelmoinnissa. Ohjelma robotille tehdään neljässä eri päävaiheessa alkaen koordinaatistojen paikoituksesta robotilla ja niiden tuomisesta RobotStudioon, jota seuraa aseman- ja ohjelman luominen ja simulointi sekä viimeisenä ohjelman siirto robotille.

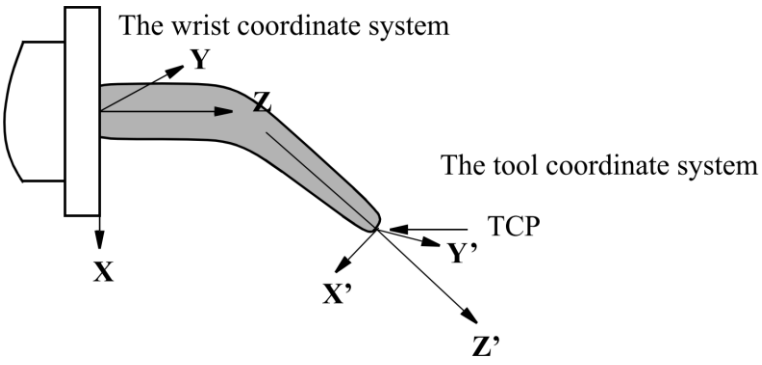
4.1 Koordinaatistojen paikoitus ja tuominen

Ohjelmaa luotaessa ensimmäisenä on suotavaa paikoittaa koordinaatistot työkaluille ja kappaleille, joille kappalekoordinaatistot luodaan. Täten virtuaaliasemassa olevien työkalujen- ja kappaleiden CAD-mallit saadaan sijoitettua siten kuin ne ovat fyysisessä robottisolussa suhteessa robottiin. Tämä mahdollistaa robotin kulkureitin paikoituspisteiden luomisen virtuaaliasemassa siten, että niitten sijainti

vastaa sijaintia fyysisessä robottisolussa. Koordinaatistoja on mahdollista uudelleen paikoittaa myöhemmin, mikäli robottisoluun tehdään muutoksia. Tällöin kyseisille koordinaatistoille luotujen robotin kulkureitin paikoituspisteitä ei tarvitse määrittää uudelleen.

4.1.1 Työkalukoordinaatiston luominen

Työkalukoordinaatistoa käytetään luodessa paikoituspisteitä robotille ja ne mahdollistavat eri työkalujen käytön ilman robotin kulkureitin paikoituspisteiden muuttamista /21, s. 101/. Työkalukoordinaatiston origo paikoitetaan työkalun kärkeen siten, että z-akseli tulee kärjen suuntaisesti.



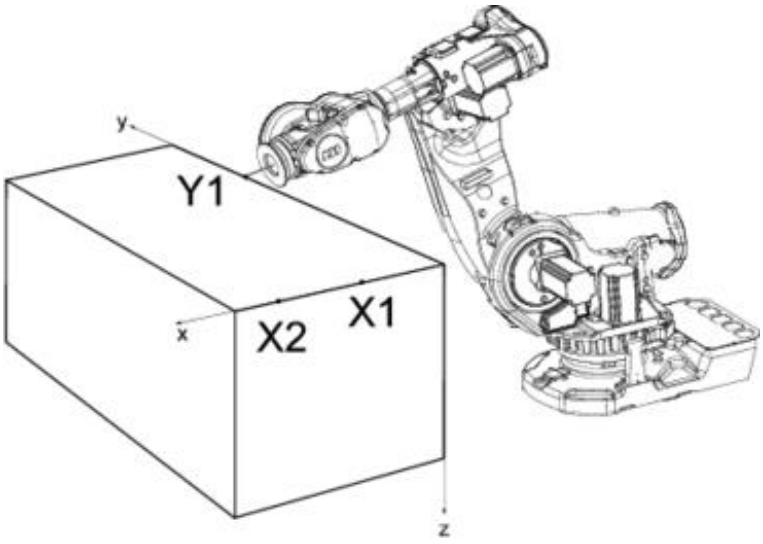
Kuva 21. Työkalukoordinaatisto sijoitetaan työkalun keskipisteeseen (TCP).

Työkalukoordinaatisto määritetään TCP & Z, X -metodilla siten, että otetaan kolme pistettä työkalun ollessa kallistettuna eri suuntiin työkalun kärjen pysyessä samassa pisteessä ja kaksi pistettä, joilla määritetään x- ja z-akseli. Työkalukoordinaatiston paikoituksessa helpottaa ohutkärkinen tappi, jonka kärkeen määritetään origo. Mikäli työkalussa ei ole pintaa, johon koordinaatiston saa paikoitettua tarkasti, on suotavaa tehdä työkaluun kiinnitettävä kohdistukseen soveltuva geometria. Tässä työssä työkalukoordinaatistot määritettiin tarttujatyökalun sormien kärkien keskipisteeseen, imukuppitarttujien imukuppien keskelle 0,2mm tartuntapinnasta imukuppiin päin ja lasityökaluun siten, että sen origo tulee lasia painettaessa keskelle lasin pintaa. /21, s. 238/

4.1.2 Kappalekoordinaatiston luominen

Kappalekoordinaatiston avulla on mahdollista tehdä korjauksia ohjelmaan ilman paikoituspisteiden muuttamista. Tämä nopeuttaa ohjelmointia kappaletta siirrettäessä. Kun sovellukseen kuuluu useampi identtinen kappale, voidaan käyttää samoja paikoituspisteitä ainoastaan siirtämällä kappalekoordinaatistoa. Kappalekoordinaatisto paikoitetaan useimmiten fyysiseen kappaleeseen mahdollistaen kappaleen paikoittamisen virtuaaliasemassa, kuten se on oikeassa robottisolussa. Kappalekoordinaatiston avulla on myös mahdollista työstää kuljettimilla liikkuvia kappaleita; paikoituspisteet liikkuvat kappalekoordinaatiston mukana.

Kappalekoordinaatisto määritetään 3-points -metodilla (Kuva 22) ottamalla kaksi pistettä, jotka määrittävät koordinaatiston x-akselin ja yksi piste, joka määrittää koordinaatiston origon sekä z- ja y-akselin /21, s. 237/. Tässä työssä työkalukoordinaatistot määritettiin siten, että niiden origot tulevat jigien ja palettien nurkkiin, joihin ne on helppo kohdistaa tarkasti.



Kuva 22. Kappalekoordinaatiston määrittäminen.

4.2 Aseman luominen

Ennen varsinaisen robotilla ajettavan ohjelman luomista on luotava virtuaalinen asema oikeassa robottisolussa käytettävällä robotilla. Asemalle luodaan virtuaalinen kontrolleri, joka on identtinen kopio robotin oikean kontrollerin ohjelmistosta.

Useampaa robottia käytettäessä asemalle on mahdollista luoda useampi kontrolleri. Tämän jälkeen asemaan tuodaan robotin kontrollerilta paikoitetut koordinaatistot, joiden avulla luodaan robotilla käytettävät työkalut ja paikoitetaan kappaleet siten, että niiden sijainti suhteessa robottiin virtuaaliasemassa vastaa sijaintia oikeassa robottisolussa suhteessa robottiin. Antureille ja liikkuville objekteille on luotava logiikat (SmartComponent), mikäli niiden käyttäytymistä halutaan simuloida virtuaalisesti. Kun asemaan on tuotu koordinaatistot, 3D-geometriat ja luotu työkalut sekä logiikat, on mahdollista aloittaa ohjelman luominen. /21, s. 83-84/

4.2.1 Koordinaatistojen tuominen

Koordinaatistoja robotilla luotaessa ne tallentuvat robotin kontrollerin fyysiseen muistiin, josta ne tuodaan PC:lle RobotStudioon. Yhteys PC:n ja kontrollerin välille on mahdollista muodostaa suoraan RJ45 -verkkokaapelilla tai verkon välityksellä. /21, s. 155/

Kun asema on luotu ja yhteys muodostettu PC:n ja robotin välille, synkronoidaan RAPID -välilehdeltä robotin kontrollerilta luodut työkalu- ja kappalekoordinaatistot asemalle.

4.2.2 3D-geometrioiden tuominen

Työkalujen ja kappaleiden 3D-geometriat tuodaan joko RobotStudion omasta kirjastosta tai erillisistä CAD-tiedostoista /21, s. 225/. RobotStudion vakioformaatti on SAT ja maksullisilla lisäosilla yleisimmät CAD-formaatit ovat tuettuja (Taulukko 1) /21, s. 42/.

Taulukko 2. Tuetut formaatit.

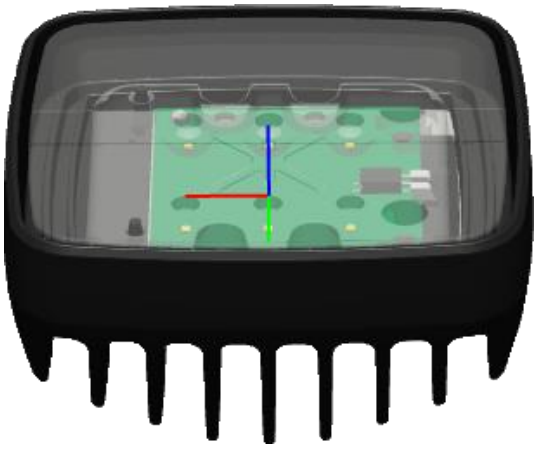
Format	File extensions	Option required
3DStudio	.3ds	-
ACIS, reads versions R1 - 2017 1.0, writes versions R18 - 2017 1.0	.sat, .sab, .asat, .asab	-
CATIA V4, reads versions 4.1.9 to 4.2.4	.model, .exp, .session	CATIA V4
CATIA V5/V6, reads versions V5R8 – V5/V6R2016 writes V5R15 – V5/V6R2016	.CATPart, .CATProduct, .CGR, .3DXML	CATIA V5
COLLADA 1.4.1	.dae	-
DirectX writes 2.0	.x	-
DXF/DWG, reads versions 2.5 - 2016	.dxf, .dwg	AutoCAD
FBX writes version 7.5	.fbx	-
IGES, reads up to version 5.3, writes version 5.3	.igs, .iges	IGES
Inventor, reads V6 – 2016	.ipt, .iam	Inventor
JT, reads versions 8.x and 9.x	.jt	JT
LDraw, reads version 1.0.2	.ldr, .ldraw, .mpd	-
NX, reads versions NX 1 – NX 10	.prt	NX
OBJ	.obj	-
Parasolid, reads versions 9.0.* – 29.0.*	.x_t, .xmt_txt, .x_b, .xmt_bin	Parasolid
Pro/E / Creo, reads versions 16 – Creo 3.0	.prt, .prt.*, .asm, .asm.*	Pro/ENGINEER
Solid Edge, reads versions V18 – ST8	.par, .asm, .psm	SolidEdge

Tuodut 3D-geometriat asettuvat aseman origoon, jonka jälkeen niille uudelleen määritetään paikalliset origot, mikäli se on tarpeellista. Paikallinen origo on suotavaa määrittää siten, että se tulee kappaleessa pisteeseen, josta se on helppo paikoittaa. Työkaluissa paikallinen origo määritetään pisteeseen, joka tulee robotin kärjen origoon. Kappaleissa, joille on luotu oma kappalekoordinaatisto, määritetään paikallinen origo pisteeseen, joka tulee kappalekoordinaatiston origoon. RobotStudiossa on mahdollista myös mallintaa 3D-geometrioita. Tosin ominaisuudet eivät ole yhtä laajat kuin erillisissä mallintamiseen tarkoitetuissa ohjelmissa. Tuo-

dut geometriat voidaan tallentaa RobotStudion kirjastoon sekä linkittää valittuun tiedostoon, jolloin geometriaan tehtävät muutokset päivittyvät asemalle automaattisesti.

Tarttujatyökalu, imukuppitarttujat ja niiden teline sekä lasityökalu tuotiin yhtenä geometriana. Tarttujatyökalun leuat sormineen tuotiin omina geometrioina, koska niiden tulee liikkua simulaatiossa. Edellä mainittujen geometrioiden origo määritettiin kohtaan, joka tulee robotin kärjen origoon mallinnettaessa työkalun kokoonpanoa. Tässä vaiheessa työkalujen geometriat jätettiin vielä aseman origoon työkalun luomista varten.

Jigien ja palettien paikallinen origo määritettiin kohtaan, joka tulee kappalekoordinaatiston origoon. LED-valaisimen komponenttien paikalliset origot määritettiin siten, että ne ovat samassa pisteessä LED-valaisin koottuna (Kuva 23). Seuraavaksi paletit ja jigit paikannettiin niille luotuihin kappalekoordinaatistoihin ja niihin luotiin koordinaatistot pisteisiin, jotka ovat niihin asetettavan kappaleen origossa. Paletteihin luotuja koordinaatistoja käytettiin myöhemmin logiikassa, joka luo komponentteja paleteille. Lisäksi luotuja koordinaatistoja käytettiin komponenttien siirtämiseen paikoituspisteitä luodessa.

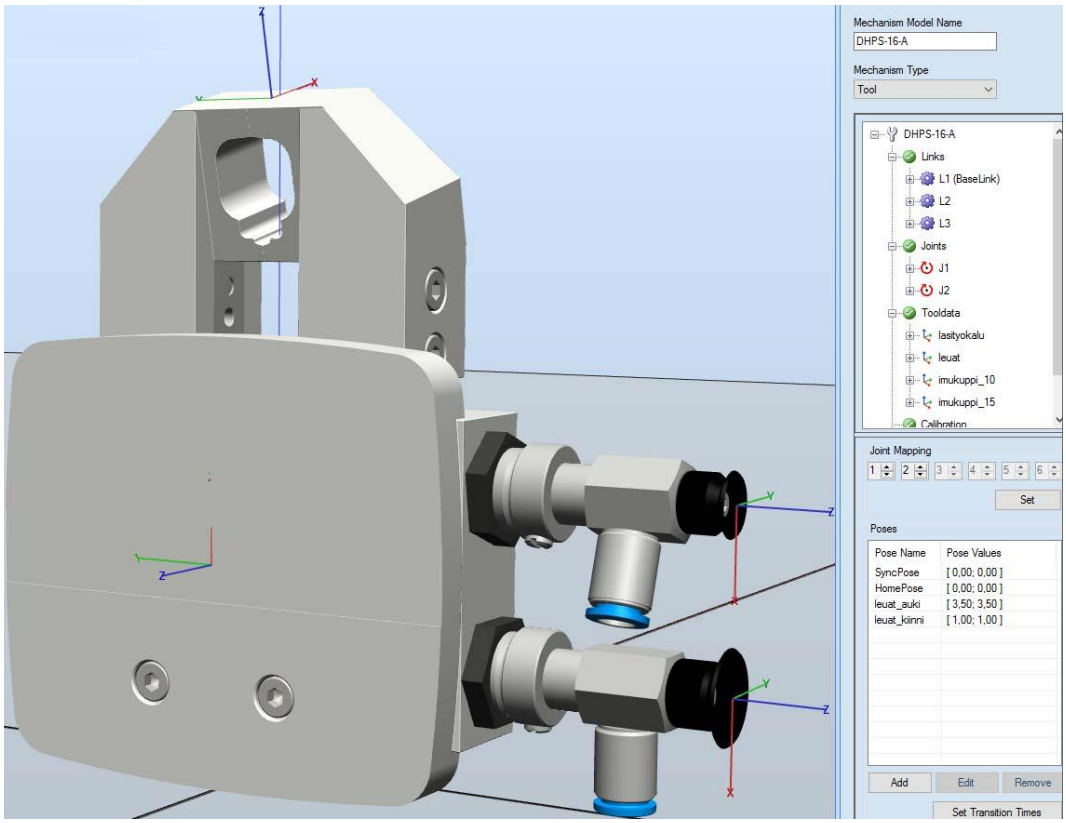


Kuva 23. Valaisimen komponenttien paikallinen origo.

4.2.3 Työkalun luominen

Työkalulle määritetään työkalukoordinaatistot, geometriat, paino ja miten se on jakautunut sekä mahdolliset lineaarisella- ja/tai kiertyvällä liikkeellä liikuteltavat osat. Työkalua luotaessa siihen tulevat koordinaatistot ja geometriat sijoitetaan siten, että niiden sijainti suhteessa aseman origoon on sama kuin niiden sijainti suhteessa robotin kärjen origoon työkalun ollessa kiinnitettynä robottiin. /21, s. 238/

Työkaluun määritettiin työkalukoordinaatistot tarttujatyökalulle, imukuppitarttujille ja lasityökalulle. Tarttujatyökalun leuoille luotiin lineaariliikkeiset nivelet, jotka mahdollistavat niiden liikuttamisen simuloinnissa.



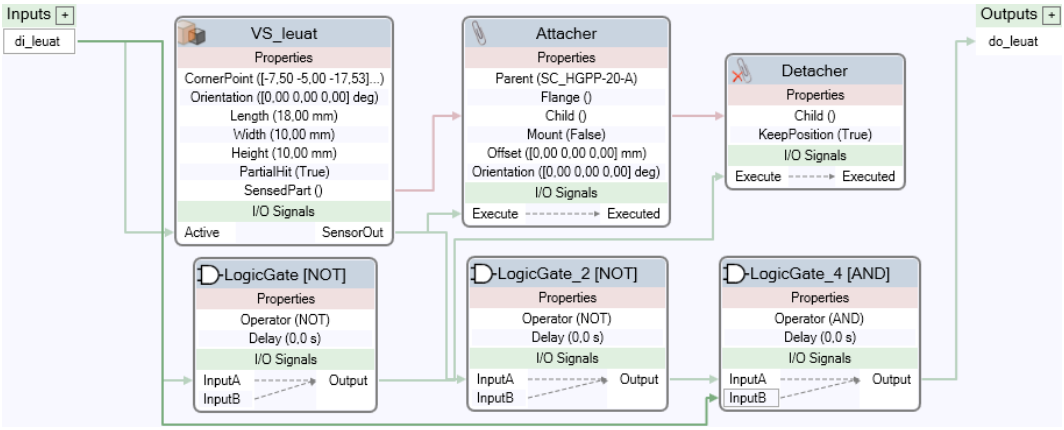
Kuva 24. Työkalun luominen.

4.2.4 Logiikan luominen

Antureille, joiden toimintaa- ja kappaleille, joiden liikettä simuloidaan, on luotava logiikka, joka määrittää säännöt objektin toiminnalle. Tämä logiikka on mahdollista luoda Smart Component -objektin avulla. Kappaleiden liikkeitä on mahdollista simuloida myös Collision Set -toiminnolla, joka soveltuu simulaatioihin, joissa liikuteltavien osien määrä on vähäinen. Smart Component -objekti on mahdollista kytkeä virtuaalikontrolleriin ja toisiin Smart Component -objekteihin.

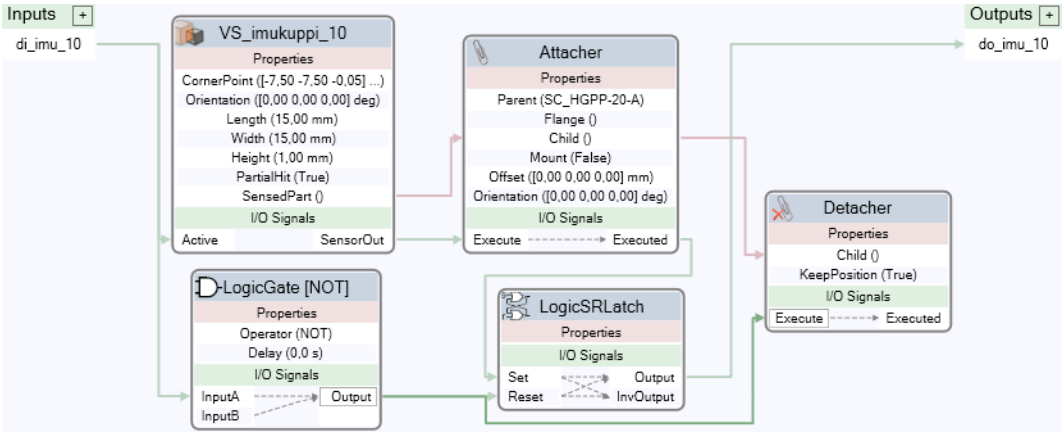
Työssä antureita simuloivat ja komponentteja manipuloivat logiikat luotiin Smart Component -objektien avulla. Manipuloitavien kappaleen määrittäminen tehtiin antureiden avulla. Valaisimen kulkiessa hihnakuljettimen päätyyn siinä olevat komponentit poistetaan ja vastaavasti kun asemalle tuodaan uusi paletti, luodaan uudet kopiot kappaleista. Logiikka luotiin tarttujatyökalulle, imukuppitarttujille, lasityökalulle, jigeille, paleteille ja hihnakuljettimelle.

Tarttujatyökalulle luotiin logiikka, joka sulkee leuat ja kiinnittää sormien leukojen välissä olevan kappaleen työkaluun saadessaan signaalin leukojen sulkemiseen robotin kontrollerilta ja sormien välissä olevalta anturilta. Kun kontrollerilta tuleva signaali muuttuu, kappale irtoaa ja leuat aukeavat. Jos välissä ei ole kappaletta ja kontrollerilta tuleva signaali on päällä, sulkeutuvat leuat siihen saakka, että sormet ottavat kiinni toisiinsa. Kun kontrollerilta tuleva leuat sulkeva signaali on päällä ja sormien välissä olevalta anturilta ei tule signaalia, kytkeytyy kontrollerille menevä signaali päälle, joka ilmaisee tarttujan sormien olevan kiinni toisissaan. Täten saadaan tieto, onko ote kappaleeseen muodostunut.



Kuva 25. Tarttujatyökalun logiikka.

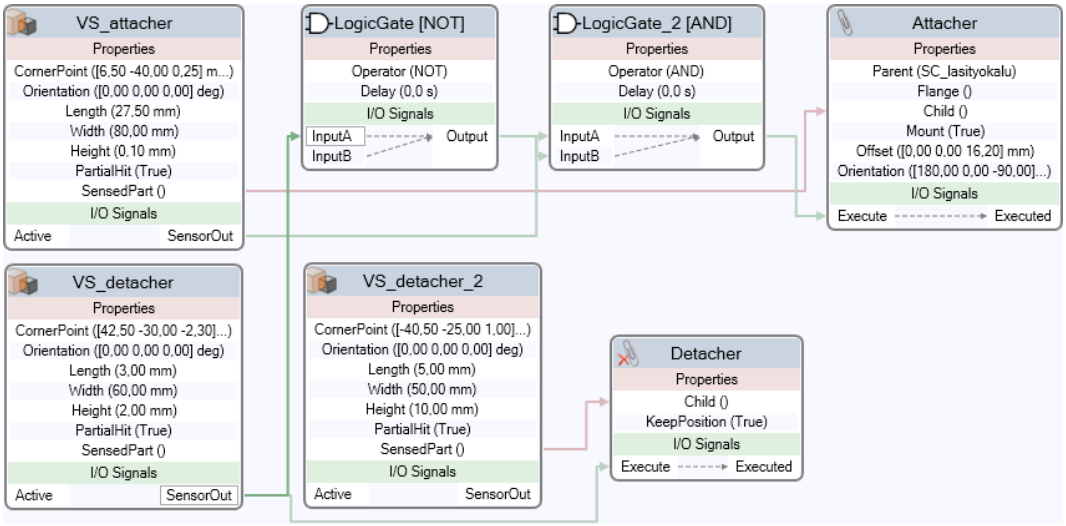
Imukuppitarttujille luotiin logiikka, joka kiinnittää imukupin tartuntapinnassa olevan kappaleen työkaluun saadessaan imun päälle kytkevän signaalin kontrollerilta ja imukupin tartuntapinnassa olevalta anturilta. Tämän jälkeen logiikka antaa signaalin kontrollerille, joka ilmaisee kappaleen olevan kiinni imukupissa. Kontrollerilta tulevan signaalin muuttuessa kappale irtoaa ja kontrollerille menevä signaali kytkeytyy pois päältä.



Kuva 26. Imukuppitarttujan logiikka.

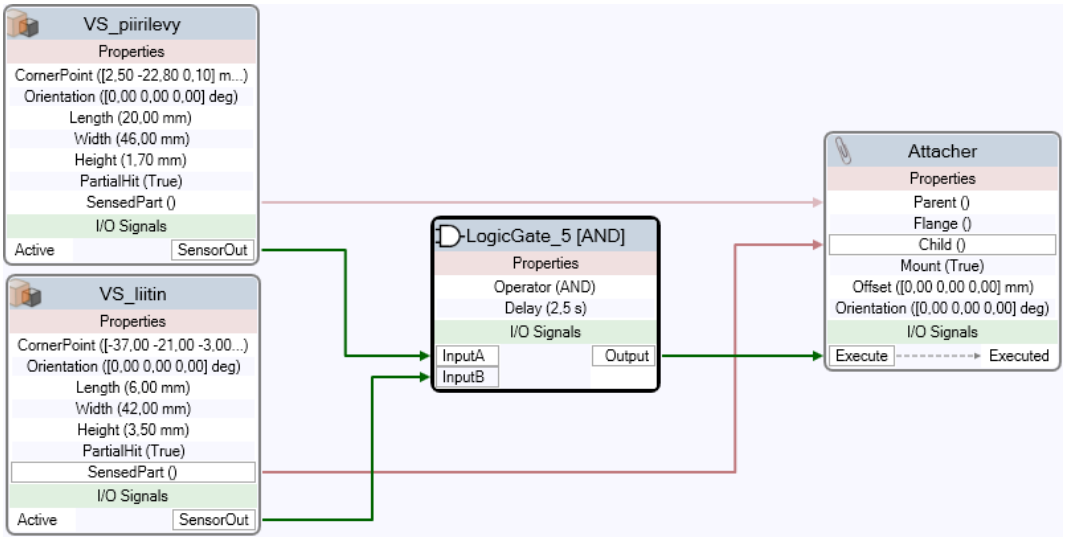
Lasityökalulle luotiin logiikka, joka kiinnittää lasityökalun pintaan osuvan kappaleen ja irrottaa kappaleen sen painamisen päättyessä. Lasityökalun pinnassa olevan sensori avulla kappale tunnistetaan ja kiinnitetään. Kappale irrotetaan toisen anturin signaalista, joka on asetettu siten, että se osuu kotelon reunaan painamisen

päättyessä. Toimivarmuuden vuoksi luotiin kolmas anturi, joka määrittää irrotettavan kappaleen.



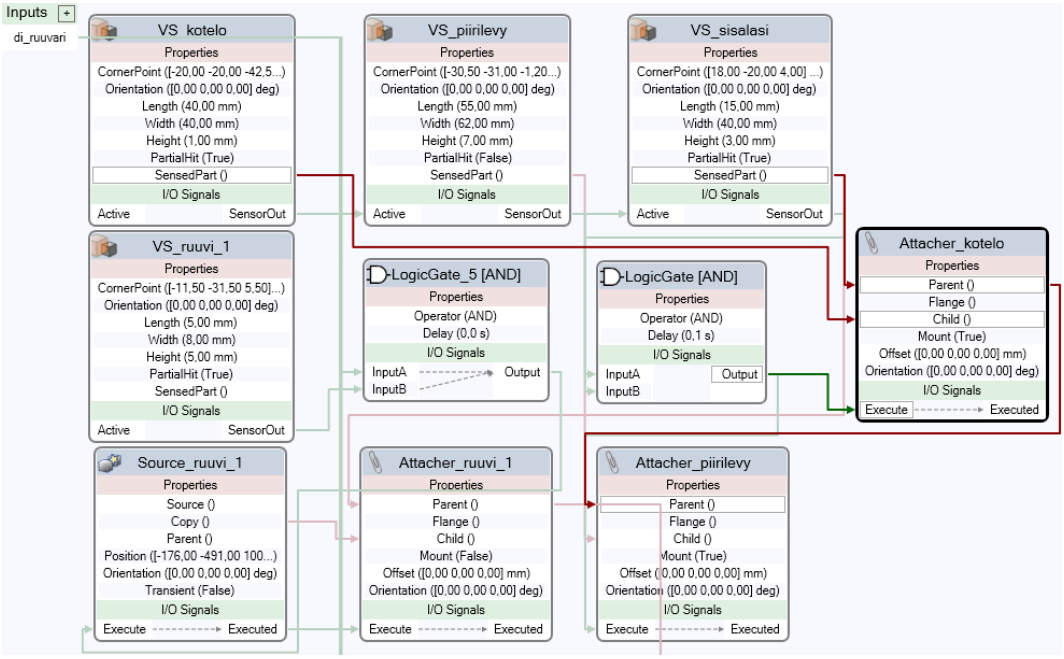
Kuva 27. Lasityökalun logiikka.

Jigiin, jossa liitetään liitin piirilevyyn, luotiin logiikka, joka kiinnittää liittimen piirilevyyn, kun molemmat komponentit ovat paikallaan. Kun molempien komponenttien tunnistavat anturit lähettävät signaalin, tapahtuu kiinnitys viiveellä sen vuoksi, että robotti ehtii asettamaan liittimen kokonaan paikalleen.



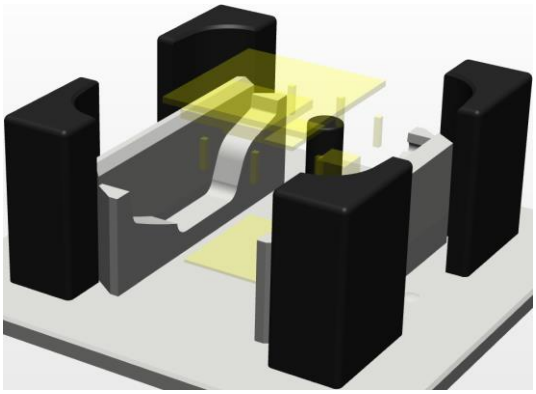
Kuva 28. Liitosjigin logiikka.

Jigiin, jossa asetetaan ja ruuvataan piirilevy ja sisempi lasi koteloon, luotiin logiikka, joka luo ruuvin sisälasiin aina, kun ruuvaustyökalu suorittaa ruuvauksen sekä kiinnittää ruuvit, piirilevyn ja kotelon sisempään lasiin. Kiinnitettäville komponenteille luotiin omat anturit ja kun kaikki anturit antavat signaalin, kiinnitetään ne sisempään lasiin. Ruuvit kiinnitetään, kun signaali kytkeytyy päälle ruuvaustyökalulta ja ruuvien reikien yläpuolella olevista antureista.



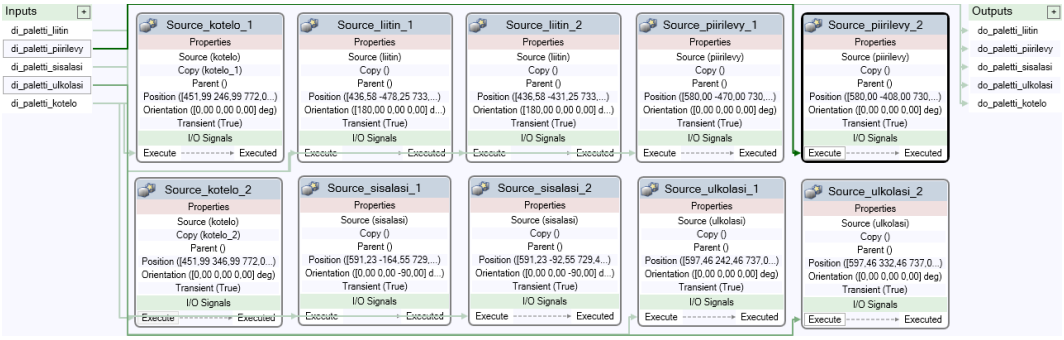
Kuva 29. Ruuvausjigin logiikka (kuvasta rajattu pois ruuvien 2-6 logiikat).

Jigiin (Kuva 30) jossa ulompi lasi liimataan ja painetaan koteloon, luotiin logiikka, (Liite 1) joka ensin poistaa valaisimen sekä jigin näkyvistä sen mennessä liiman levittävälle koneelle ja kun ulompi lasi on painettu paikalleen, kiinnittää se sisemmän lasin ulompaan lasiin. Jokaisella komponentilla on oma anturi, joka määrittää kiinnitettävät ja näkyvistä poistettavat kappaleet.



Kuva 30. Suorakulmaiset keltaiset särmiöt ovat antureita, joita käytettiin simulointiin liimaukseen käytettävän jigin logiikassa.

Paleteille luotiin logiikka, joka saadessaan signaalin luo kyseiselle paletille sille kuuluvat komponentit ja antaa signaalin kontrollerille lisätyistä komponenteista.



Kuva 31. Koulun simulaatiossa käytettävä palettien logiikka.

Hihnakuljettimelle luotiin logiikka (Liite 2), joka kuljettaa ja poistaa kootun valaisimen. RobotStudiossa on kuljettimen simulointiin tarkoitettu toiminto (Conveyor), jossa tulee määrittää liikutettava osa etukäteen; täten se ei sovellu tämän työn simulointiin, koska kyseisessä sovelluksessa on liikutettava useita etukäteen määrittämättömiä kappaleita samanaikaisesti. Logiikka toteutettiin siten, että kotelon osuessa kuljettimeen se irrotetaan ulkolasista, jonka jälkeen ulkolasi kiinnitetään koteloon. Tämä tehdään siksi, koska liikutettava kappale ei saa olla kiinnitettynä toiseen kappaleeseen, kun sitä liikutetaan, mutta liikutettavassa kappaleessa saa olla muita kappaleita kiinnitettynä. Kiinnityksen suunnalla on siis merkitystä. Tämän jälkeen LED-valaisin kuljetetaan pois solusta kotelon ollessa kuljetettava

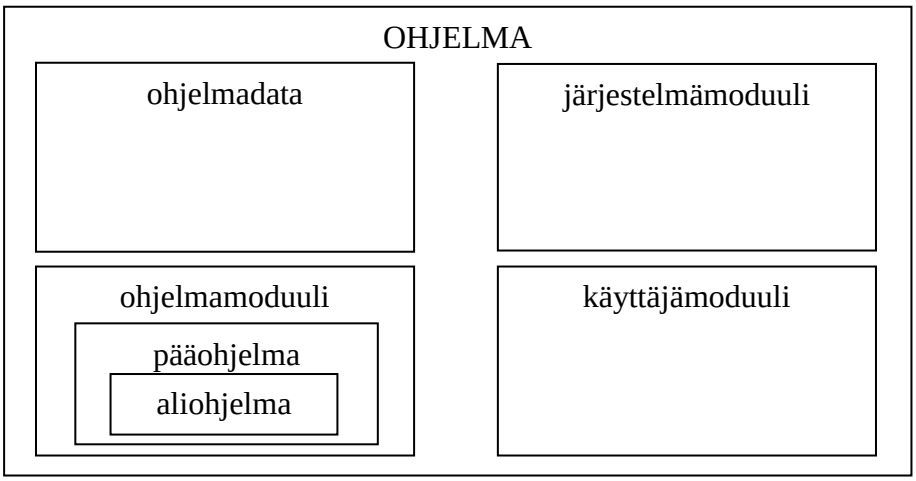
osa. Kun LED-valaisin saavuttaa hihnakuljettimen päädyn pysäytetään kappale ja poistetaan siinä olevat komponentit niille tarkoitetuilla antureilla. Logiikkaan jouduttiin lisäämään useita ylimääräisiä ehtoja ja viiveitä, jotta hihnakuljettimen simulointi saatiin sujumaan luotettavasti ilman virheitä. Osittain tästä syystä logiikasta kasvoi massiivinen suhteessa simuloitavaan toimintoon.

Kuva 32. Suorakulmaiset keltaiset särmiöt ovat antureita, joita käytettiin simulointiin hihnakuljettimen logiikassa.

4.3 Ohjelman luominen

Tässä luvussa käsitellään ohjelmoinnin pääkohdat, joita käytettiin kyseiseen sovellukseen. RobotStudiossa käytetään RAPID -ohjelmointikieltä, jolla kirjoitetaan ohjelma, joka ohjaa kontrolleriin kytkettyä robottia ja manipuloitavia laitteita. Ohjelma koostuu ohjelmadatasta (CalibData), ohjelmamoduulista (Module), järjestelmämoduulista (BASE) ja käyttäjämoduulista (user).

Taulukko 2. Ohjelman rakenne.



Ohjelmadata sisältää kyseiseen ohjelmaan määritetyt työkalu- ja kappalekoordinaatistot. Ohjelmamoduuli sisältää muuttujat (VAR), pysyvät muuttujat (PERS) kuten paikoituspisteet (robtarget), pääohjelman ja aliohjelmat. Järjestelmämoduuli on tallennettu robotin sisäiseen muistiin ja se sisältää mm. työkalujen ja koordinaatistojen määrittelyt. Käyttäjämoduuliin on mahdollista tallentaa työkalu- ja kappalekoordinaatistoja sekä numeerisia muuttujia.

```
MODULE CalibData
    PERS tooldata leuat:=[TRUE,[[17,0,160.145],[1,0,0,0]],[1,[1,1,1],[1,0,0,0],0,0,0]];
    PERS tooldata lasityokalu:=[TRUE,[[0,56.3,40],[0.5,-0.5,0.5,0.5]],[1,[1,1,1],[1,0,0,0],0,0,0]];
    PERS tooldata imukuppi_10:=[TRUE,[[12.5,-134.2,55.5],[0.5,0.5,0.5,-0.5]],[1,[1,1,1],[1,0,0,0],0,0,0]];
    PERS tooldata imukuppi_15:=[TRUE,[[-16,-134.2,55.5],[0.5,0.5,0.5,-0.5]],[1,[1,1,1],[1,0,0,0],0,0,0]];
    PERS wobjdata poyta:=[FALSE,TRUE,"",[[400,-500,84],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    TASK PERS wobjdata paletti_liitin:=[FALSE,TRUE,"",[[400,-500,84],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    TASK PERS wobjdata paletti_piirilevy:=[FALSE,TRUE,"",[[250,-500,84],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    TASK PERS wobjdata paletti_kotelo:=[FALSE,TRUE,"",[[400,-500,84],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    TASK PERS wobjdata paletti_sisalasi:=[FALSE,TRUE,"",[[400,-500,84],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    TASK PERS wobjdata paletti_ulkolasi:=[FALSE,TRUE,"",[[250,-500,84],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    TASK PERS wobjdata jigi_ruuvaus:=[FALSE,TRUE,"",[[400,-500,84],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    TASK PERS wobjdata jigi_liitos:=[FALSE,TRUE,"",[[400,-500,84],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    TASK PERS wobjdata jigi_liittaminen:=[FALSE,TRUE,"",[[400,-500,84],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
ENDMODULE
```

Kuva 33. Ohjelmadata.

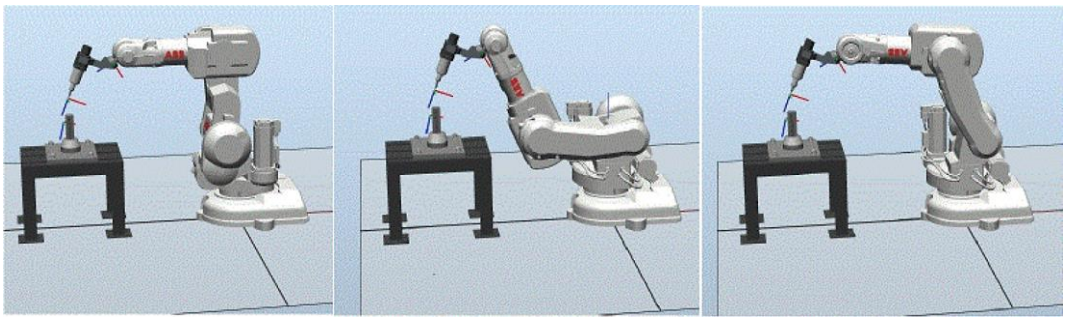
Ennen ohjelman luomista on oltava valmiina asema, joka sisältää robotin, työkalut, käsiteltävät kappaleet ja kaikki objektit, joiden avulla halutaan luoda paikoituspisteitä tai havaita mahdollisia törmäyksiä. Useimmiten on suotavaa luoda ensimmäisenä reitit paikoituspisteineen ja jälkimmäisenä ehdot, koska robotin liikerataa voidaan simuloida ilman ehtoja, mutta usein ehtoja ei voida simuloida ilman robotin liikettä. /21, s. 109/

4.3.1 Paikoituspisteiden ja reittien luominen

Paikoituspisteet ovat sijainteja, joiden kautta robotin työkalu kulkee ajettaessa ohjelmaa ja ne voidaan luoda graafisessa näkymässä tai kirjoittamalla käyttäen RAPID-ohjelmointikieltä. Paikoituspisteet ovat mahdollista paikantaa manuaalisesti antamalla koordinaatit ja asemassa olevien geometrioiden-, koordinaatistojen- sekä robotin työkalun avulla. Paikoituspisteille määritetään kappalekoordinaatisto, johon ne luodaan. Kappalekoordinaatistoa siirrettäessä sille luodut paikoituspisteet siirtyvät sen mukana. Tämä nopeuttaa ohjelmointia, kun käsiteltävää kappa-

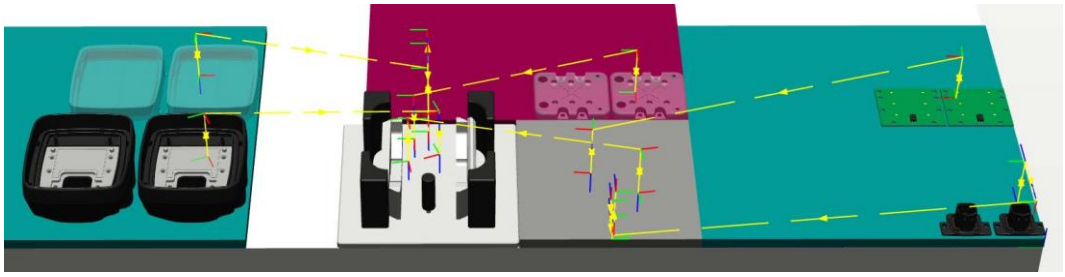
letta siirretään ja kun sovelluksessa käsitellään useampaa identtistä kappaletta voidaan ohjelmaan luoda käsky, joka siirtää kappalekoordinaatistoa seuraavan kappaleen kohdalle tehtäessä useamman kappaleen sarjaa. Tällöin vältytään kappalekoordinaatistoon luotujen paikoituspisteiden siirtämiseltä erikseen tai erillisten paikoituspisteiden luomiselta. /21, s. 112-116/

Paikoituspisteitä on kahdentyyppisiä; target ja jointtarget. Target -toiminto määrittää työkalukoordinaatiston sijainnin ja kiertymän, jolle voidaan määrittää yksi tai useampi konfiguraatio (Kuva 34) riippuen robotin liikkuvuudesta. Jointtarget -toiminto määrittää robotin jokaisen nivelen kiertymän, täten sillä on ainoastaan yksi konfiguraatio. Konfiguraatiolla tässä asiayhteydessä tarkoitetaan robotin asentoa, jonka määrittävät robotin nivelten kiertymät (kiertyvänivelisellä robotilla). Täten paikoituspisteellä ollessa useita konfiguraatioita robotti ulottuu siihen useassa eri asennossa. /21, s. 112-116/



Kuva 34. Robotin konfiguraatio.

Reitti on sekvenssi, joka koostuu useasta liikekomennosta paikoituspisteineen ja se voidaan luoda graafisessa näkymässä tai kirjoittamalla RAPID-ohjelmointikieltä. Liikekomento on mahdollista luoda lineaari- ja nivelliikkeenä. Reitti voidaan luoda jo olevista paikoituspisteistä, opettamalla liikuttamalla robottia ja geometrian muodoista. RobotStudioon on mahdollista ostaa lisäosia, jotka tuovat lisää tapoja reitin luomiselle. Reitti voidaan luoda suoraan pääohjelmaan ja yhteen- tai useampaan aliohjelmaan jota/joita kutsutaan pääohjelmassa. /21, s. 112-116/



Kuva 35. Robotin kulkurata keltaisella katkoviivalla (keskeneräinen simulaatio koulussa tehtävästä kokoonpanosta).

Ennen reitin luomista on syytä asettaa robotin liikkeiden oletusnopeudet (speed) ja oletusarvot aluedatoille (zone). Aluedata määrittää kaaren koon, joka syntyy robotin työkalun liikkuessa paikoituspisteelle. Näin ollen, mitä pienempi aluedatan arvo on, sitä kulmikkaampi liikkeestä tulee. Kun reitti on luotu, tulisi määrittää robotin konfiguraatiot paikoituspisteille. Se on suotavaa tehdä tässä vaiheessa, koska tällöin on helppoa valita konfiguraatiot, jotka ovat mahdollisimman lähellä toisiaan perättäisten paikoituspiteiden välillä. Tällöin robotin liikeradasta saadaan mahdollisimman sujuva. Luotuja reittejä on mahdollista siirtää, kääntää ja interpoloida sekä reittien sisältämien liikekomentojen järjestystä on mahdollista muuttaa jälkikäteen. /21, s. 112-116/

4.3.2 Komentojen ja ehtojen luominen

Komennoilla ohjataan robotin lisäksi muita kontrolleriin kytkettyjä laitteita ja ne voidaan luoda graafisessa näkymässä tai kirjoittaen RAPID-ohjelmointikielen avulla. Komennot jakautuvat kahteen eri kategoriaan; liikekomentoihin ja toimintakomentoihin. Liikekomennoiksi lukeutuvat komennot, joilla liikutetaan robottia. Kaikki loput komennot lukeutuvat toimintokomennoiksi, esim. komennot joilla ohjataan kontrolleriin kytkettyjä laitteita ja komennot, joilla asetetaan viiveet ja koordinaatistojen siirrot ohjelmassa. /21, s. 121-127, 22/

Ohjelmaan tehtävät ehdot kirjoitetaan RAPID -ohjelmointikielen avulla. Ehdoilla määritetään, miten kontrolleriin kytketty robotti ja laitteet käyttäytyvät määritetyissä tilanteissa. Ehtoja käytettiin suoritettavan sarjakoon määrittämiseksi, robo-

tin käyttäytymisen määrittämiseksi kappaleen puuttuessa paletilta, -paletin tyhjentyessä ja -uuden paletin tullessa soluun.

Seuraava selostus komentojen ja ehtojen määrittämisestä on koulussa käytettävästä ohjelmasta (Liite 3), jossa sarjakoko on kaksi ja kappaleet ovat aseman y-akselin suuntaisesti.

Valmistettavien kappaleiden sarjakoko ja tehtyjen kappaleiden laskuri ohjelmoitiin seuraavasti:

VAR num counter:=0;

Määrittää laskurille numeerisen muuttujan, joka alkaa asetetusta arvosta.

```
FOR i FROM 1 TO 2 DO
Path_10;
Path_20;
Path_30;
Path_40;
Path_50;
Path_60;
Incr counter;
ENDFOR
```

Määrittää mistä arvosta mihin arvoon FOR ja ENDFOR -komentojen välillä olevaa koodia ajetaan. Incr counter -komento lisää laskurin arvoa yhdellä ja loput ylläolevat komennot kutsuvat aliohjelmaa.

Kotelon haku paletilta ja sen vienti jigille, paletin kappalekoordinaatiston siirto ja robotin toiminta kappaleen puuttuessa paletilta määritettiin seuraavasti:

VAR num iy_kotelo:=1;

Määrittää numeerisen muuttujan, jonka avulla määritetään, monesko y-akselin suuntainen kappale haetaan paletilta.

PROC Path_10()

Määrittää aliohjelman alun.

```
MoveJ Target_10,v100,fine,imukuppi_15\WObj:=paletti_kotelo;
WaitDI DI_paletti_kotelo,1;
MoveL Target_10_2,v50,fine,imukuppi_15\WObj:=paletti_kotelo;
SetDO DO_imu_10,1;
SetDO DO_imu_15,1;
WaitTime 0.3;
```

Ajetaan robotti paletin yläpuolelle ja tarvittaessa odotetaan, että paikallaan on paletti, jossa on kappaleita. Tämän jälkeen ajetaan robotti haettavan kotelon kohdalle ja laitetaan imukuppitarttujiin imu päälle.

```
IF DI_imu_10=1 AND DI_imu_15=1 THEN
    MoveL Target_10,v50,fine,imukuppi_15\WObj:=paletti_kotelo;
    MoveJ Target_20,v100,fine,imukuppi_15\WObj:=jigi_ruuvaus;
    MoveL Target_20_2,v50,fine,imukuppi_15\WObj:=jigi_ruuvaus;
    SetDO DO_imu_10,0;
    SetDO DO_imu_15,0;
    WaitTime 0.3;
    MoveJ Target_20,v100,z100,imukuppi_15\WObj:=jigi_ruuvaus;
```

Jos molempiin imukuppitarttujiin syntyy alipaine, kuljetetaan kotelo jigille, jossa tehdään ruuvaus ja asetetaan imukuppitarttujista imu pois päältä.

```
IF iy_kotelo<2 THEN
  paletti_kotelo.uframe.trans.y:=paletti_kotelo.uframe.trans.y+100;
  iy_kotelo:=iy_kotelo+1;
ELSE
  SetDO DO_paletti_kotelo,0;
  paletti_kotelo.uframe.trans.y:=paletti_kotelo.uframe.trans.y-100;
  iy_kotelo:=1;
ENDIF
```

Jos haettavan kotelon määrittävän muuttujan arvo on alle kaksi, lisätään sen arvoa yhdellä ja tehdään y-akselin suuntainen kappalekoordinaatiston siirto positiiviseen suuntaan (toisen kotelon kohdalle) sekä lisätään haettavan kotelon muuttujan arvoa yhdellä. Muulloin asetetaan signaali, joka ilmaisee, että paletilla on kappaleita pois päältä ja tehdään y-akselin suuntainen kappalekoordinaatiston siirto negatiiviseen suuntaan (ensimmäisen kotelon kohdalle) sekä asetetaan haettavan kotelon arvoksi yksi.

```
ELSE
  IF iy_kotelo<2 THEN
    paletti_kotelo.uframe.trans.y:=paletti_kotelo.uframe.trans.y+100;
```

```
      iy_kotelo:=iy_kotelo+1;
      Path_10;
    ELSE
      SetDO DO_paletti_kotelo,0;
      paletti_kotelo.uframe.trans.y:=paletti_kotelo.uframe.trans.y-100;
      iy_kotelo:=1;
      Path_10;
    ENDIF
  ENDIF
```

Jos imukuppitarttujiin ei synny alipainetta ja jos haettavan kotelon määrittävä muuttujan arvo on alle kaksi, lisätään sen arvoa yhdellä, tehdään y-akselin suuntainen kappalekoordinaatiston siirto positiiviseen suuntaan (toisen kotelon kohdalle) ja lisätään haettavan kotelon muuttujan arvoa yhdellä sekä ajetaan aliohjelma alusta. Muulloin asetetaan signaali joka ilmaisee, että paletilla on kappaleita pois päältä, tehdään y-akselin suuntainen kappalekoordinaatiston siirto negatiiviseen suuntaan (ensimmäisen kotelon kohdalle) ja asetetaan haettavan kotelon arvoksi yksi sekä ajetaan aliohjelma alusta.

```
ENDPROC
```

Määrittää aliohjelman lopun.

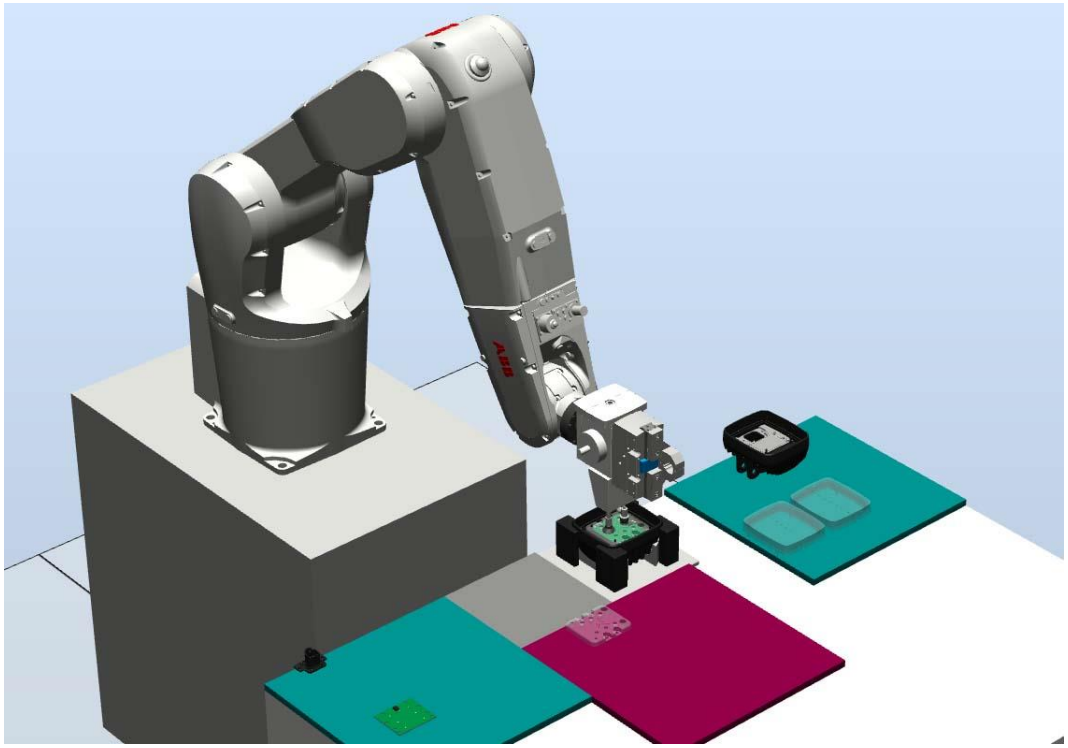
Muiden komponenttien haku toteutettiin komennoilla ja ehdoilla, joissa on sama periaate kuin kotelon haussa, joten niitä ei käydä erikseen läpi. Yritykselle tulevassa ohjelmassa muuttujia, jotka määrittävät haettavan komponentin, on kaksi kutakin komponenttia kohden, koska kappaleita on paletilla vierekkäin (y-akselin suuntaisesti) ja peräkkäin (x-akselin suuntaisesti). Täten myös koordinaatiston siirrot tehdään x- ja y-akselin suuntaisesti.

4.3.3 Ohjelman simulointi

Simulaatio on prosessi, jossa luodaan toteutettavaa sovellusta jäljittelevä virtuaalinen malli. Simuloinnin avulla on mahdollista analysoida toteutettavan sovelluksen käyttäytymistä virtuaalisesti ennen laitteiston hankkimista, mikä mahdollistaa sulavan siirtymisen konseptista toteutukseen. Suunnitteluvaiheessa simuloinnin ansiosta on mahdollista tarkastaa robotin ulottuvuuden riittävyys, joka helpottaa sovelluksessa käytettävän robotin valitsemista työkaluineen. Ohjelmoitaessa si-

mulointi nopeuttaa reitin ja siinä käytettävän robotin konfiguraation määrittämistä sekä auttaa törmäyksien ehkäisemisessä. Täten robottisolun valmistukseen menevää aikaa, kustannuksia ja onnettomuusriskiä saadaan alennettua. /19-20/

Simulaatiossa ohjelmaa ajetaan virtuaalikontrollerilla, joka on identtinen kopio robotin oikean kontrollerin ohjelmistosta /18/. Useamman robotin konfiguraatiota on myös mahdollista simuloida, jolloin jokaista robottia ohjaa oma virtuaalikontrolleri. Simulaatiossa pystytään havaitsemaan törmäyksiä etukäteen määritettyjen kappaleiden välillä.



Kuva 36. Simulaatio koulussa tehtävästä kokoonpanosta (keskeneräinen).

5 YHTEENVETO

Opinnäytetyön tavoitteena oli automatisoida 5-osaisen LED-valaisimen kokoonpano robottia käyttäen. Työn kuvaan kuului robottisolun suunnitteleminen, robotin ohjelmointi ja simulointi sekä jigien, palettien ja tasojen CAD-mallien luominen. Tehtävä vaati perehtymistä erityyppisiin robotteihin, robotin työkaluihin sekä robotin ohjelmointiin ja simulointiin. Suurin osa opinnäytetyöstä osoittautui robotin ohjelman luomiseksi ja simuloinniksi. Kokoonpano oli tarkoitus testata koulussa olevalla robotilla ja robotin työkaluilla.

Opinnäytetyössä suunniteltiin robottisolu, johon valittiin robotti ja kokoonpanoon tarvittavat robotin työkalut. Robotin työkaluihin suunniteltiin ja mallinnettiin tarvittavat osat. Kokoonpanossa tarvittavien jigien, palettien sekä tasojen suunnittelu ja mallintaminen oli annettu tehtäväksi toiselle opiskelijalle. Mallinnetut robotin työkalut, jigit ja paletit oli tarkoitus valmistaa 3D-tulostamalla koulun simulaatiota varten. Robottisolusta luotiin virtuaalinen malli, jota käyttäen luotiin ja simuloitiin ohjelma robotille. Koulussa tehtävään kokoonpanon testaukseen luotiin erillinen virtuaalinen malli osineen, johon ohjelmoitiin erillinen ohjelma.

Opinnäytetyön tuloksena saatiin selvitettyä, miten ja millä komponenteilla LED-valaisimen kokoonpano on mahdollista automatisoida. Projekti jäi kesken jigien, palettien ja tasojen CAD-mallien viivästymisestä johtuen. Niitä tarvitaan lopullisen ohjelman luomiseen ja edellä mainittujen osien valmistamiseen. Täten lopulliset ohjelmat oikeine liikeratoineen ja koulussa suoritettava kokoonpanon testaus jäi suorittamatta. Opinnäytetyö antoi yritykselle mallin, miten ja millä osilla kokoonpanon automatisointi on mahdollista suorittaa, opetti paljon robottisolun suunnittelusta ja robotin ohjelmoinnista sekä antoi hyvin kokemusta työelämää varten, huolimatta siitä, että jäi kesken.

LÄHTEET

/1/ Tietoa meistä. Viitattu 1.3.2017. Herrmans Oy Ab. http://www.nordiclights.com/start-finnish/yritys/tietoa-meista

/2/ 2016. EU Declaration of Conformity. Viitattu 1.3.2017. Herrmans Oy Ab. EU Declaration of Conformity - KL range_B_April 2016.pdf

/3/ Martin, G. Robotic Cell Layout Considerations. Viitattu 1.4.2017. Bastian Solutions, Inc. https://www.bastiansolutions.com/blog/index.php/2015/10/09/robotic-cell-layout-considerations/#.WQrNcp9E6Uk

/4/ Pare, A. 2015. Q&As When Developing a Robotic Manipulation Cell (Part 1of 2). Viitattu 1.4.2017. Robotiq. http://blog.robotiq.com/qas-when-developing-a-manipulation-cell

/5/ Pare, A. 2015. Q&As When Developing a Robotic Manipulation Cell (Part 2/2). Viitattu 1.4.2017. Robotiq. http://blog.robotiq.com/qas-when-developing-a-robotic-manipulation-cell-part-2

/6/ Bélanger-Barrette, M. How to Choose the Right Industrial Robot?. Viitattu 1.4.2017. Robotiq. http://blog.robotiq.com/bid/70408/How-to-Choose-the-Right-Industrial-Robot

/7/ 2017. Kuivanen, R. 1999. Robotiikka. ISBN 951-9438-58-0. Talentum Oyj.

/8/ 2016. Robotiikka. Viitattu 15.4.2017. Lahden ammattikorkeakoulu. http://miniweb.lpt.fi/automaatio/opetus/luennot/pdf_tiedostot/Robotiikka_yleinen.pdf, 12-18

/9/ 2017. Cobot. Viitattu 15.4.2017. Wikipedia. https://en.wikipedia.org/wiki/Cobot

/10/ 2016. IP-luokitus. Viitattu 15.4.2017. Wikipedia. https://fi.wikipedia.org/wiki/IP-luokitus

/11/ 2014. IRB 1200, A compact, flexible, fast and functional small industrial robot. Viitattu 15.4.2017. ABB AB. https://library.e.abb.com/public/6f866d55894a49a7b44384db76e01b4b/IRB1200_ROB0275EN_A.pdf

/12/ 2015. How To Choose The Right End Effector For Your Application. Viitattu 15.4.2017. Robotiq. http://robotiq.com/wp-content/uploads/2015/06/How-to-Choose-the-Right-End-Effector-F.pdf%3FsubmissionGuid%3De9cff220-d1e3-444c-9f69-e8d4ec716fff

/13/ 2017. Parallel grippers DHPS. Viitattu 15.4.2017. Festo KG. https://www.festo.com/cat/en-gb_gb/data/doc_ENGB/PDF/EN/DHPS_EN.PDF

/14/ 2017. Proximity sensors SMT/SME-8, for T-slot. Viitattu 15.4.2017. Festo KG. https://www.festo.com/cat/en-gb_gb/data/doc_ENGB/PDF/EN/SMX8_EN.PDF

/15/ 2017. Parallel grippers HGPP, precision. Viitattu 15.4.2017. Festo KG. https://www.festo.com/cat/en-gb_gb/data/doc_ENGB/PDF/EN/HGPP_EN.PDF

/16/ 2017. Vacuum suction grippers ESG. Viitattu 15.4.2017. Festo KG. https://www.festo.com/cat/en-gb_gb/data/doc_ENGB/PDF/EN/ESG_EN.PDF

/17/ 2017. Suction cups, complete ESS and suction cups ESV. Viitattu 15.4.2017. Festo KG. https://www.festo.com/net/SupportPortal/Files/10584/ESS-ESV_ENUS.pdf

/18/ RobotStudio®. Viitattu 15.4.2017. ABB AB. http://new.abb.com/products/robotics/RobotStudio

/19/ Brumson, B. 2009. Robotic Simulation and Off-line Programming: From Academia to Industry. Viitattu. 1.5.2017. Robotic Industries Association. http://www.robotics.org/content-detail.cfm/Industrial-Robotics-Industry-Insights/Robotic-Simulation-and-Off-line-Programming-From-Academia-to-Industry/content_id/1825

/20/ 2013. Robotics Simulation Softwares With 3D Modeling and Programming Support. Viitattu 1.5.2017. Into Robotics. https://www.intorobotics.com/robotics-simulation-softwares-with-3d-modeling-and-programming-support/

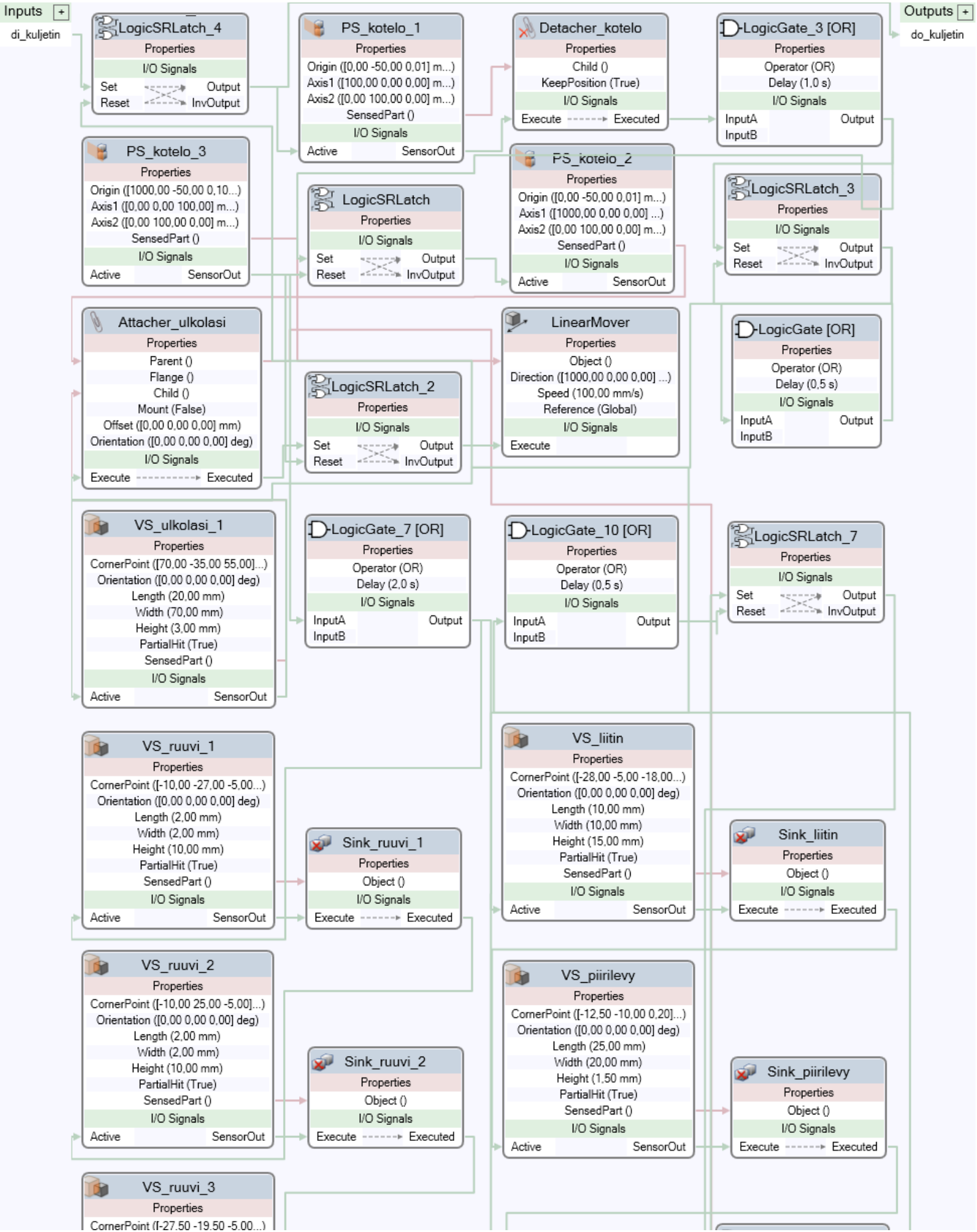
/21/ 2016. Operating manual – RobotStudio. Viitattu 15.4.2017. ABB AB. Operating manual RobotStudio 6.04.pdf

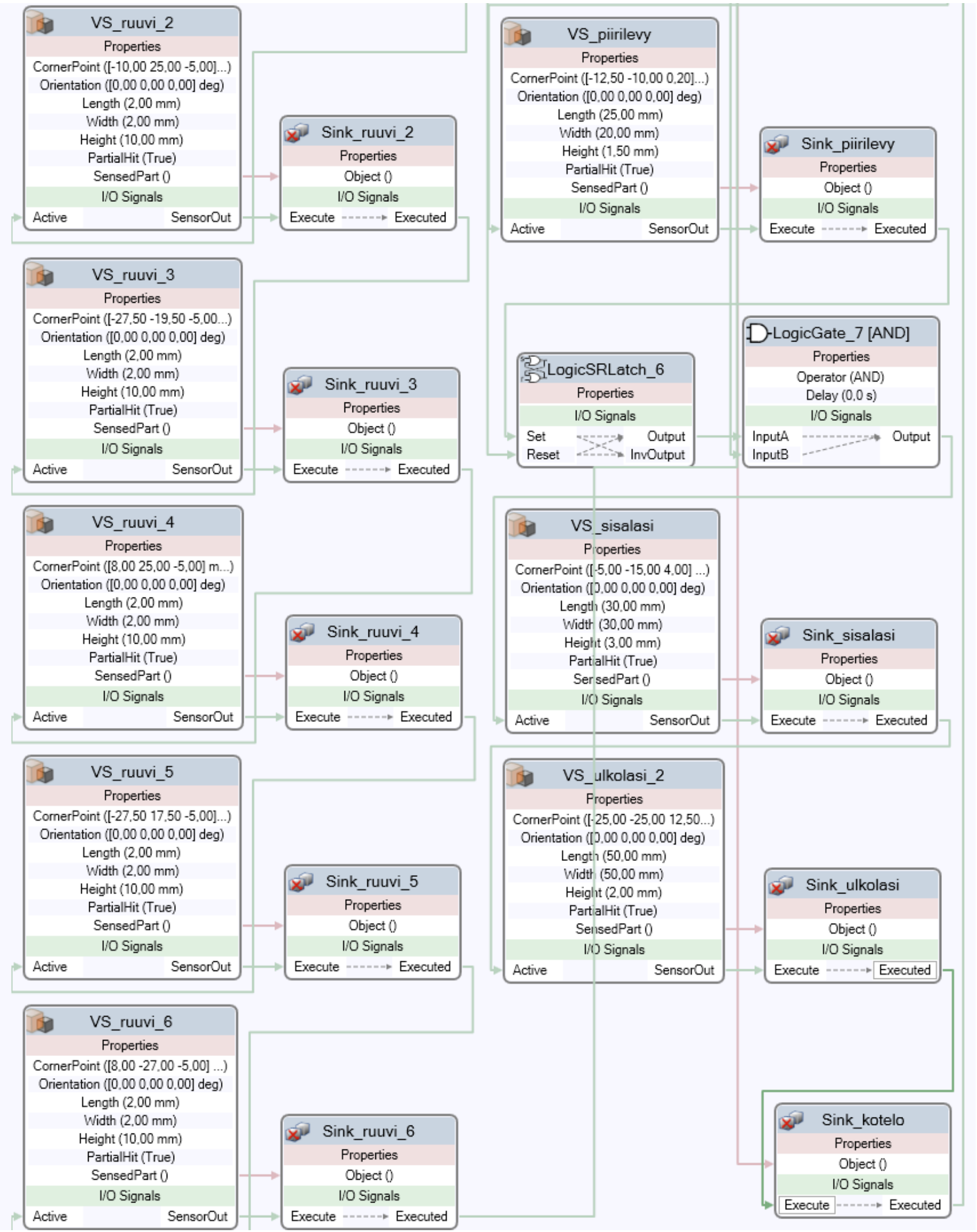
/22/ Technical reference manual, RAPID Instructions, Functions and Data types 2010. Viitattu 15.4.2017. ABB AB. https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf


Inputs
di_liimaus
Outputs
do_liimaus
VS_kotelo
Properties
CornerPoint ([-20,00 -20,00 -42,5...)
Orientation ([0,00 0,00 0,00] deg)
Length (40,00 mm)
Width (40,00 mm)
Height (1,00 mm)
PartialHit (True)
SensedPart (liimausjigi)
I/O Signals
Active
SensorOut
VS_liitin
Properties
CornerPoint ([-28,00 -5,00 -18,00...)
Orientation ([0,00 0,00 0,00] deg)
Length (10,00 mm)
Width (10,00 mm)
Height (15,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
VS_piirilevy
Properties
CornerPoint ([-12,50 -10,00 0,20]...)
Orientation ([0,00 0,00 0,00] deg)
Length (25,00 mm)
Width (20,00 mm)
Height (1,50 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
VS_sisalasi
Properties
CornerPoint ([-5,00 -15,00 4,00] ...)
Orientation ([0,00 0,00 0,00] deg)
Length (30,00 mm)
Width (30,00 mm)
Height (3,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
VS_ulkolasi
Properties
CornerPoint ([-25,00 -25,00 12,50...)
Orientation ([0,00 0,00 0,00] deg)
Length (50,00 mm)
Width (50,00 mm)
Height (2,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
LogicGate [OR]
Properties
Operator (OR)
Delay (0,2 s)
I/O Signals
InputA
InputB
Output
Attacher
Properties
Parent ()
Flange ()
Child ()
Mount (True)
Offset ([0,00 0,00 0,00] mm)
Orientation ([0,00 0,00 0,00] deg)
I/O Signals
Execute
Executed
LogicGate_2 [OR]
Properties
Operator (OR)
Delay (6,0 s)
I/O Signals
InputA
InputB
Output
LogicGate_3 [AND]
Properties
Operator (AND)
Delay (0,0 s)
I/O Signals
InputA
InputB
Output
Hide_kotelo
Properties
Object (liimausjigi)
I/O Signals
Execute
Executed
Hide_liitin
Properties
Object ()
I/O Signals
Execute
Executed
Hide_piirilevy
Properties
Object ()
I/O Signals
Execute
Executed
Hide_sisalasi
Properties
Object ()
I/O Signals
Execute
Executed
Hide_liimausjigi
Properties
Object (liimausjigi)
I/O Signals
Execute
Executed
Show_kotelo
Properties
Object (liimausjigi)
I/O Signals
Execute
Executed
Show_liitin
Properties
Object ()
I/O Signals
Execute
Executed
Show_piirilevy
Properties
Object ()
I/O Signals
Execute
Executed
Show_sisalasi
Properties
Object ()
I/O Signals
Execute
Executed
Show_liimausjigi
Properties
Object (liimausjigi)
I/O Signals
Execute
Executed
VS_ruuvi_1
Properties
CornerPoint ([-10,00 -27,00 -5,00...)
Orientation ([0,00 0,00 0,00] deg)
Length (2,00 mm)
Width (2,00 mm)
Height (10,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Hide_ruuvi_1
Properties
Object ()
I/O Signals
Execute
Executed
Show_ruuvi_1
Properties
Object ()
I/O Signals
Execute
Executed
VS_ruuvi_2
Properties
CornerPoint ([-10,00 25,00 -5,00]...)
Orientation ([0,00 0,00 0,00] deg)
Length (2,00 mm)
Width (2,00 mm)
Height (10,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Hide_ruuvi_2
Properties
Object ()
I/O Signals
Execute
Executed
Show_ruuvi_2
Properties
Object ()
I/O Signals
Execute
Executed
VS_ruuvi_3
Properties
CornerPoint ([-27,50 -19,50 -5,00...)
Orientation ([0,00 0,00 0,00] deg)
Length (2,00 mm)
Width (2,00 mm)
Height (10,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Hide_ruuvi_3
Properties
Object ()
I/O Signals
Execute
Executed
Show_ruuvi_3
Properties
Object ()
I/O Signals
Execute
Executed
VS_ruuvi_4
Properties
CornerPoint ([8,00 25,00 -5,00] m...)
Orientation ([0,00 0,00 0,00] deg)
Length (2,00 mm)
Width (2,00 mm)
Height (10,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Hide_ruuvi_4
Properties
Object ()
I/O Signals
Execute
Executed
Show_ruuvi_4
Properties
Object ()
I/O Signals
Execute
Executed
VS_ruuvi_5
Properties
CornerPoint ([-27,50 17,50 -5,00]...)
Orientation ([0,00 0,00 0,00] deg)
Length (2,00 mm)
Width (2,00 mm)
Height (10,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Hide_ruuvi_5
Properties
Object ()
I/O Signals
Execute
Executed
Show_ruuvi_5
Properties
Object ()
I/O Signals
Execute
Executed
VS_ruuvi_6
Properties
CornerPoint ([8,00 -27,00 -5,00] ...)
Orientation ([0,00 0,00 0,00] deg)
Length (2,00 mm)
Width (2,00 mm)
Height (10,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Hide_ruuvi_6
Properties
Object ()
I/O Signals
Execute
Executed
Show_ruuvi_6
Properties
Object ()
I/O Signals
Execute
Executed

I/O Connections

Source Object	Source Signal	Target Object	Target Signal
VS_ulkolasi	SensorOut	LogicGate [OR]	InputA
VS_liitin	SensorOut	VS_piirilevy	Active
VS_piirilevy	SensorOut	VS_sisalasi	Active
SC_liimausjigi	di_liimaus	Hide_kotelo	Execute
SC_liimausjigi	di_liimaus	Hide_piirilevy	Execute
SC_liimausjigi	di_liimaus	Hide_sisalasi	Execute
SC_liimausjigi	di_liimaus	Hide_liimausjigi	Execute
SC_liimausjigi	di_liimaus	Hide_liitin	Execute
SC_liimausjigi	di_liimaus	LogicGate_2 [OR]	InputA
LogicGate_2 [OR]	Output	Show_liitin	Execute
LogicGate_2 [OR]	Output	Show_piirilevy	Execute
LogicGate_2 [OR]	Output	Show_sisalasi	Execute
LogicGate_2 [OR]	Output	Show_kotelo	Execute
LogicGate_2 [OR]	Output	Show_liimausjigi	Execute
LogicGate_2 [OR]	Output	SC_liimausjigi	do_liimaus
VS_kotelo	SensorOut	VS_liitin	Active
LogicGate [OR]	Output	Attacher	Execute
VS_piirilevy	SensorOut	LogicGate_3 [AND]	InputA
LogicGate_2 [OR]	Output	LogicGate_3 [AND]	InputB
LogicGate_3 [AND]	Output	VS_ulkolasi	Active
SC_liimausjigi	di_liimaus	Hide_ruuvi_1	Execute
SC_liimausjigi	di_liimaus	Hide_ruuvi_2	Execute
SC_liimausjigi	di_liimaus	Hide_ruuvi_3	Execute
SC_liimausjigi	di_liimaus	Hide_ruuvi_4	Execute
LogicGate_2 [OR]	Output	Show_ruuvi_1	Execute
LogicGate_2 [OR]	Output	Show_ruuvi_2	Execute
LogicGate_2 [OR]	Output	Show_ruuvi_3	Execute
LogicGate_2 [OR]	Output	Show_ruuvi_4	Execute
LogicGate_2 [OR]	Output	Show_ruuvi_5	Execute
LogicGate_2 [OR]	Output	Show_ruuvi_6	Execute
SC_liimausjigi	di_liimaus	Hide_ruuvi_5	Execute
SC_liimausjigi	di_liimaus	Hide_ruuvi_6	Execute


Inputs
di_kuljetin
Outputs
do_kuljetin
LogicSRLatch_4
Properties
I/O Signals
Set
Reset
Output
InvOutput
PS_kotelo_1
Properties
Origin ([0,00 -50,00 0,01] m...)
Axis1 ([100,00 0,00 0,00] m...)
Axis2 ([0,00 100,00 0,00] m...)
SensedPart ()
I/O Signals
Active
SensorOut
Detacher_kotelo
Properties
Child ()
KeepPosition (True)
I/O Signals
Execute
Executed
LogicGate_3 [OR]
Properties
Operator (OR)
Delay (1,0 s)
I/O Signals
InputA
InputB
Output
PS_kotelo_3
Properties
Origin ([1000,00 -50,00 0,10...)
Axis1 ([0,00 0,00 100,00] m...)
Axis2 ([0,00 100,00 0,00] m...)
SensedPart ()
I/O Signals
Active
SensorOut
LogicSRLatch
Properties
I/O Signals
Set
Reset
Output
InvOutput
PS_kotelo_2
Properties
Origin ([0,00 -50,00 0,01] m...)
Axis1 ([1000,00 0,00 0,00] ...)
Axis2 ([0,00 100,00 0,00] m...)
SensedPart ()
I/O Signals
Active
SensorOut
LogicSRLatch_3
Properties
I/O Signals
Set
Reset
Output
InvOutput
Attacher_ulkolasi
Properties
Parent ()
Flange ()
Child ()
Mount (False)
Offset ([0,00 0,00 0,00] mm)
Orientation ([0,00 0,00 0,00] deg)
I/O Signals
Execute
Executed
LinearMover
Properties
Object ()
Direction ([1000,00 0,00 0,00] ...)
Speed (100,00 mm/s)
Reference (Global)
I/O Signals
Execute
LogicGate [OR]
Properties
Operator (OR)
Delay (0,5 s)
I/O Signals
InputA
InputB
Output
LogicSRLatch_2
Properties
I/O Signals
Set
Reset
Output
InvOutput
VS_ulkolasi_1
Properties
CornerPoint ([70,00 -35,00 55,00]...)
Orientation ([0,00 0,00 0,00] deg)
Length (20,00 mm)
Width (70,00 mm)
Height (3,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
LogicGate_7 [OR]
Properties
Operator (OR)
Delay (2,0 s)
I/O Signals
InputA
InputB
Output
LogicGate_10 [OR]
Properties
Operator (OR)
Delay (0,5 s)
I/O Signals
InputA
InputB
Output
LogicSRLatch_7
Properties
I/O Signals
Set
Reset
Output
InvOutput
VS_ruuvi_1
Properties
CornerPoint ([-10,00 -27,00 -5,00...)
Orientation ([0,00 0,00 0,00] deg)
Length (2,00 mm)
Width (2,00 mm)
Height (10,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Sink_ruuvi_1
Properties
Object ()
I/O Signals
Execute
Executed
VS_liitin
Properties
CornerPoint ([-28,00 -5,00 -18,00...)
Orientation ([0,00 0,00 0,00] deg)
Length (10,00 mm)
Width (10,00 mm)
Height (15,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Sink_liitin
Properties
Object ()
I/O Signals
Execute
Executed
VS_ruuvi_2
Properties
CornerPoint ([-10,00 25,00 -5,00]...)
Orientation ([0,00 0,00 0,00] deg)
Length (2,00 mm)
Width (2,00 mm)
Height (10,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Sink_ruuvi_2
Properties
Object ()
I/O Signals
Execute
Executed
VS_piirilevy
Properties
CornerPoint ([-12,50 -10,00 0,20]...)
Orientation ([0,00 0,00 0,00] deg)
Length (25,00 mm)
Width (20,00 mm)
Height (1,50 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Sink_piirilevy
Properties
Object ()
I/O Signals
Execute
Executed
VS_ruuvi_3
Properties
CornerPoint ([-27,50 -19,50 -5,00...)


VS_ruuvi_2
Properties
CornerPoint ([-10,00 25,00 -5,00]...)
Orientation ([0,00 0,00 0,00] deg)
Length (2,00 mm)
Width (2,00 mm)
Height (10,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Sink_ruuvi_2
Properties
Object ()
I/O Signals
Execute
Executed
VS_piirilevy
Properties
CornerPoint ([-12,50 -10,00 0,20]...)
Orientation ([0,00 0,00 0,00] deg)
Length (25,00 mm)
Width (20,00 mm)
Height (1,50 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Sink_piirilevy
Properties
Object ()
I/O Signals
Execute
Executed
VS_ruuvi_3
Properties
CornerPoint ([-27,50 -19,50 -5,00...)
Orientation ([0,00 0,00 0,00] deg)
Length (2,00 mm)
Width (2,00 mm)
Height (10,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Sink_ruuvi_3
Properties
Object ()
I/O Signals
Execute
Executed
LogicSRLatch_6
Properties
I/O Signals
Set
Reset
Output
InvOutput
LogicGate_7 [AND]
Properties
Operator (AND)
Delay (0,0 s)
I/O Signals
InputA
InputB
Output
VS_ruuvi_4
Properties
CornerPoint ([8,00 25,00 -5,00] m...)
Orientation ([0,00 0,00 0,00] deg)
Length (2,00 mm)
Width (2,00 mm)
Height (10,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Sink_ruuvi_4
Properties
Object ()
I/O Signals
Execute
Executed
VS_sisalasi
Properties
CornerPoint ([-5,00 -15,00 4,00] ...)
Orientation ([0,00 0,00 0,00] deg)
Length (30,00 mm)
Width (30,00 mm)
Height (3,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Sink_sisalasi
Properties
Object ()
I/O Signals
Execute
Executed
VS_ruuvi_5
Properties
CornerPoint ([-27,50 17,50 -5,00]...)
Orientation ([0,00 0,00 0,00] deg)
Length (2,00 mm)
Width (2,00 mm)
Height (10,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Sink_ruuvi_5
Properties
Object ()
I/O Signals
Execute
Executed
VS_ulkolasi_2
Properties
CornerPoint ([-25,00 -25,00 12,50...)
Orientation ([0,00 0,00 0,00] deg)
Length (50,00 mm)
Width (50,00 mm)
Height (2,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Sink_ulkolasi
Properties
Object ()
I/O Signals
Execute
Executed
VS_ruuvi_6
Properties
CornerPoint ([8,00 -27,00 -5,00] ...)
Orientation ([0,00 0,00 0,00] deg)
Length (2,00 mm)
Width (2,00 mm)
Height (10,00 mm)
PartialHit (True)
SensedPart ()
I/O Signals
Active
SensorOut
Sink_ruuvi_6
Properties
Object ()
I/O Signals
Execute
Executed
Sink_kotelo
Properties
Object ()
I/O Signals
Execute
Executed

I/O Connections

Source Object	Source Signal	Target Object	Target Signal
Attacher_ulkolasi	Executed	LogicSRLatch_2	Set
LogicSRLatch_2	Output	LinearMover	Execute
PS_kotelo_3	SensorOut	LogicSRLatch	Reset
PS_kotelo_3	SensorOut	LogicSRLatch_2	Reset
LogicSRLatch	Output	PS_kotelo_2	Active
PS_kotelo_1	SensorOut	Detacher_kotelo	Execute
VS_ulkolasi_1	SensorOut	Attacher_ulkolasi	Execute
Detacher_kotelo	Executed	LogicGate_3 [OR]	InputA
LogicGate_3 [OR]	Output	LogicSRLatch	Set
LogicGate_3 [OR]	Output	LogicSRLatch_3	Set
LogicSRLatch_3	Output	VS_ulkolasi_1	Active
LogicSRLatch_3	Output	LogicGate [OR]	InputA
LogicGate [OR]	Output	LogicSRLatch_3	Reset
SC_kuljetin	di_kuljetin	LogicSRLatch_4	Set
LogicSRLatch_4	Output	PS_kotelo_1	Active
Sink_ulkolasi	Executed	Sink_kotelo	Execute
PS_kotelo_3	SensorOut	LogicGate_7 [OR]	InputA
LogicGate_7 [OR]	Output	VS_liitin	Active
VS_liitin	SensorOut	Sink_liitin	Execute
Sink_liitin	Executed	VS_piirilevy	Active
VS_piirilevy	SensorOut	Sink_piirilevy	Execute
VS_sisalasi	SensorOut	Sink_sisalasi	Execute
Sink_sisalasi	Executed	VS_ulkolasi_2	Active
VS_ulkolasi_2	SensorOut	Sink_ulkolasi	Execute
LogicSRLatch_4	Output	SC_kuljetin	do_kuljetin
LogicGate_7 [OR]	Output	VS_ruuvi_1	Active
VS_ruuvi_1	SensorOut	Sink_ruuvi_1	Execute
LogicGate_7 [AND]	Output	VS_sisalasi	Active
Sink_piirilevy	Executed	LogicSRLatch_6	Set
LogicSRLatch_6	Output	LogicGate_7 [AND]	InputA
VS_ruuvi_2	SensorOut	Sink_ruuvi_2	Execute
VS_ruuvi_3	SensorOut	Sink_ruuvi_3	Execute
VS_ruuvi_4	SensorOut	Sink_ruuvi_4	Execute
VS_ruuvi_6	SensorOut	Sink_ruuvi_6	Execute
VS_ruuvi_5	SensorOut	Sink_ruuvi_5	Execute
LogicSRLatch_7	Output	LogicGate_7 [AND]	InputB
Sink_ruuvi_1	Executed	VS_ruuvi_2	Active
Sink_ruuvi_2	Executed	VS_ruuvi_3	Active
Sink_ruuvi_3	Executed	VS_ruuvi_4	Active
Sink_ruuvi_4	Executed	VS_ruuvi_5	Active
Sink_ruuvi_5	Executed	VS_ruuvi_6	Active
Sink_ruuvi_6	Executed	LogicSRLatch_7	Set
Sink_kotelo	Executed	LogicGate_10 [OR]	InputA
LogicGate_10 [OR]	Output	LogicSRLatch_7	Reset
LogicGate_10 [OR]	Output	LogicSRLatch_6	Reset
LogicGate_10 [OR]	Output	LogicSRLatch_4	Reset

```
MODULE Module1
    CONST jointtarget JointTarget_1:=[[0,0,0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10_2:=[[61.838560884,735.091643859,52],[0,-0.190808995,0.981627184,0],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10:=[[61.838560884,735.091643859,102],[0,-0.190808995,0.981627184,0],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20_2:=[[68.10556888,530.148060139,49],[0,0.559192904,0.829037572,0],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20:=[[68.10556888,530.148060139,99],[0,0.559192904,0.829037572,0],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_30_2:=[[338.033,47.34,11.6],[0,0.736864505,0.676040459,0],[-1,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_30:=[[338.033,47.34,61.6],[0,0.736864505,0.676040459,0],[-1,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_40_2:=[[38.03,397.337,53.33],[0,0.736864505,0.676040459,0],[0,1,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_40:=[[38.03,397.337,103.33],[0,0.736864505,0.676040459,0],[0,1,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_50_2:=[[26.077,21.75,26.965],[0.707106781,0,0.707106781,0],[-1,-1,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_50:=[[26.077,21.75,76.965],[0.707106781,0,0.707106781,0],[-1,-1,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_60_2:=[[19.497,380,38.465],[0,0.707106781,0,0.707106781],[-1,-1,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_60:=[[-25,380,38.465],[0,0.707106781,0,0.707106781],[-1,-1,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_60_3:=[[19.497,380,20.265],[0,0.707106781,0,0.707106781],[-1,-1,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_60_4:=[[-10.503,380,20.265],[0,0.707106781,0,0.707106781],[-1,-1,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_60_5:=[[-25,380,88.465],[0,0.707106781,0,0.707106781],[-1,-1,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_50_3:=[[-25,21.75,76.965],[0.707106781,0,0.707106781,0],[-1,-1,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_70:=[[57.217991633,557.574496727,100.6],[0,-0.079068064,0.99686922,0],[0,-1,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_70_2:=[[57.217991633,557.574496727,50.6],[0,-0.079068064,0.99686922,0],[0,-1,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_80:=[[191.23,347.95,67.49],[0,0,1,0],[-1,-1,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_80_2:=[[191.23,347.95,17.49],[0,0,1,0],[-1,-1,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_90:=[[80,552.5,107],[0,0,1,0],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_90_2:=[[80,552.5,57],[0,0,1,0],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_100:=[[347.368,755.462,83],[0,-0.707106781,0.707106781,0],[0,-1,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_100_2:=[[347.368,755.462,33],[0,-0.707106781,0.707106781,0],[0,-1,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_110:=[[96.008,540,124.53],[0.100336231,0.00002078,-0.994948703,-0.003117633],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_110_2:=[[96.008,540,74.53],[0.100336231,0.00002078,-0.994948703,-0.003117633],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_110_3:=[[96.008,540,124.53],[0.100336231,0.00002078,-0.994948703,-0.003117633],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_120:=[[83.313,540,177.32],[0.1003015,0.000021767,-0.994952231,-0.003108974],[0,0,-2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_120_2:=[[83.313,540,77.32],[0.1003015,0.000021767,-0.994952231,-0.003108974],[0,0,-1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_120_3:=[[80,540,65.2],[0,0,1,0],[0,0,-1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_120_4:=[[80,540,165.2],[0,0,1,0],[0,0,-2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_130:=[[12.425634486,357.217571437,103.33],[0,0.760802549,-0.648983422,0],[-1,-2,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_130_2:=[[12.425634486,357.217571437,53.33],[0,0.760802549,-0.648983422,0],[-1,-2,1,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    VAR num iy_kotelo:=1;
    VAR num iy_liitin:=1;
    VAR num iy_piirilevy:=1;
    VAR num iy_sisalasi:=1;
    VAR num iy_ulkolasi:=1;
    VAR num counter:=0;
        PROC DO_reset()
        SetDO DO_imu_10,0;
        SetDO DO_imu_15,0;
        SetDO DO_kuljetin,0;
        SetDO DO_leuat,1;
        SetDO DO_liimaus,0;
        SetDO DO_paletti_liitin,1;
        SetDO DO_paletti_piirilevy,1;
        SetDO DO_paletti_kotelo,1;
        SetDO DO_paletti_sisalasi,1;
        SetDO DO_paletti_ulkolasi,1;
        SetDO DO_ruuvaus,0;
    ENDPROC
    PROC wobj_reset()
    paletti_kotelo:=[FALSE,TRUE,"",[[400,-500,84],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    paletti_liitin:=[FALSE,TRUE,"",[[400,-500,84],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    paletti_piirilevy:=[FALSE,TRUE,"",[[250,-500,84],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    paletti_sisalasi:=[FALSE,TRUE,"",[[400,-500,84],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    paletti_ulkolasi:=[FALSE,TRUE,"",[[250,-500,84],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
    ENDPROC
    PROC main()
        DO_reset;
        wobj_reset;
        FOR i FROM 1 TO 2 DO
        Path_10;
        Path_20;
        Path_30;
        Path_40;
        Path_50;
        Path_60;
        Incr counter;
        ENDFOR
        MoveAbsJ JointTarget_1,v200,z100,imukuppi_15\WObj:=paletti_liitin;
```

```
ENDPROC
PROC Path_10()
    MoveJ Target_10,v100,fine,imukuppi_15\WObj:=paletti_kotelo;
    WaitDI DI_paletti_kotelo,1;
    MoveL Target_10_2,v50,fine,imukuppi_15\WObj:=paletti_kotelo;
    SetDO DO_imu_10,1;
    SetDO DO_imu_15,1;
    WaitTime 0.3;
    IF DI_imu_10=1 AND DI_imu_15=1 THEN
    MoveL Target_10,v50,fine,imukuppi_15\WObj:=paletti_kotelo;
    MoveJ Target_20,v100,fine,imukuppi_15\WObj:=jigi_ruuvaus;
    MoveL Target_20_2,v50,fine,imukuppi_15\WObj:=jigi_ruuvaus;
    SetDO DO_imu_10,0;
    SetDO DO_imu_15,0;
    WaitTime 0.3;
    MoveJ Target_20,v100,z100,imukuppi_15\WObj:=jigi_ruuvaus;
      IF iy_kotelo<2 THEN
        paletti_kotelo.uframe.trans.y:=paletti_kotelo.uframe.trans.y+100;
        iy_kotelo:=iy_kotelo+1;
      ELSE
        SetDO DO_paletti_kotelo,0;
        paletti_kotelo.uframe.trans.y:=paletti_kotelo.uframe.trans.y-100;
        iy_kotelo:=1;
      ENDIF
    ELSE
      IF iy_kotelo<2 THEN
        paletti_kotelo.uframe.trans.y:=paletti_kotelo.uframe.trans.y+100;
        iy_kotelo:=iy_kotelo+1;
        Path_10;
      ELSE
        SetDO DO_paletti_kotelo,0;
        paletti_kotelo.uframe.trans.y:=paletti_kotelo.uframe.trans.y-100;
        iy_kotelo:=1;
        Path_10;
      ENDIF
    ENDIF
ENDPROC
PROC Path_20()
    MoveJ Target_30,v100,z100,imukuppi_10\WObj:=paletti_piirilevy;
    WaitDI DI_paletti_piirilevy,1;
    MoveL Target_30_2,v50,fine,imukuppi_10\WObj:=paletti_piirilevy;
    SetDO DO_imu_10,1;
    WaitTime 0.3;
    IF DI_imu_10=1 THEN
    MoveL Target_30,v50,fine,imukuppi_10\WObj:=paletti_piirilevy;
    MoveJ Target_40,v100,z100,imukuppi_10\WObj:=jigi_liittaminen;
    MoveL Target_40_2,v50,fine,imukuppi_10\WObj:=jigi_liittaminen;
    SetDO DO_imu_10,0;
    WaitTime 0.3;
    MoveJ Target_40,v100,z100,imukuppi_10\WObj:=jigi_liittaminen;
      IF iy_piirilevy<2 THEN
        paletti_piirilevy.uframe.trans.y:=paletti_piirilevy.uframe.trans.y+62;
        iy_piirilevy:=iy_piirilevy+1;
      ELSE
        SetDO DO_paletti_piirilevy,0;
        paletti_piirilevy.uframe.trans.y:=paletti_piirilevy.uframe.trans.y-62;
        iy_piirilevy:=1;
      ENDIF
    ELSE
      IF iy_piirilevy<2 THEN
        paletti_piirilevy.uframe.trans.y:=paletti_piirilevy.uframe.trans.y+62;
        iy_piirilevy:=iy_piirilevy+1;
        Path_20;
      ELSE
        SetDO DO_paletti_piirilevy,0;
        paletti_piirilevy.uframe.trans.y:=paletti_piirilevy.uframe.trans.y-62;
        iy_piirilevy:=1;
        Path_20;
      ENDIF
    ENDIF
ENDPROC
PROC Path_30()
    MoveJ Target_50,v100,fine,leuat\WObj:=paletti_liitin;
    SetDO DO_leuat,0;
```

```
  WaitDI DI_paletti_liitin,1;
  MoveL Target_50_2,v50,fine,leuat\WObj:=paletti_liitin;
  SetDO DO_leuat,1;
  WaitTime 0.3;
  IF DI_leuat=0 THEN
  MoveL Target_50,v50,fine,leuat\WObj:=paletti_liitin;
  MoveJ Target_50_3,v100,z100,leuat\WObj:=paletti_liitin;
  MoveJ Target_60,v100,fine,leuat\WObj:=jigi_liittaminen;
  MoveL Target_60_2,v50,fine,leuat\WObj:=jigi_liittaminen;
  SetDO DO_leuat,0;
  WaitTime 0.3;
  MoveL Target_60_3,v50,fine,leuat\WObj:=jigi_liittaminen;
  MoveL Target_60_4,v100,fine,leuat\WObj:=jigi_liittaminen;
  SetDO DO_leuat,1;
  MoveJ Target_60_5,v100,z100,leuat\WObj:=jigi_liittaminen;
    IF iy_liitin<2 THEN
      paletti_liitin.uframe.trans.y:=paletti_liitin.uframe.trans.y+47;
      iy_liitin:=iy_liitin+1;
    ELSE
      SetDO DO_paletti_liitin,0;
      paletti_liitin.uframe.trans.y:=paletti_liitin.uframe.trans.y-47;
      iy_liitin:=1;
    ENDIF
  ELSE
    IF iy_liitin<2 THEN
      paletti_liitin.uframe.trans.y:=paletti_liitin.uframe.trans.y+47;
      iy_liitin:=iy_liitin+1;
      Path_30;
    ELSE
      SetDO DO_paletti_liitin,0;
      paletti_liitin.uframe.trans.y:=paletti_liitin.uframe.trans.y-47;
      iy_liitin:=1;
      Path_30;
    ENDIF
  ENDIF
ENDPROC
PROC Path_40()
  MoveL Target_130,v100,z100,imukuppi_10\WObj:=jigi_liittaminen;
  MoveL Target_130_2,v50,fine,imukuppi_10\WObj:=jigi_liittaminen;
  SetDO DO_imu_10,1;
  WaitTime 0.3;
  MoveL Target_130,v50,fine,imukuppi_10\WObj:=jigi_liittaminen;
  MoveJ Target_70,v100,z100,imukuppi_10\WObj:=jigi_ruuvaus;
  MoveL Target_70_2,v50,fine,imukuppi_10\WObj:=jigi_ruuvaus;
  SetDO DO_imu_10,0;
  WaitTime 0.3;
  MoveJ Target_70,v100,z100,imukuppi_10\WObj:=jigi_ruuvaus;
ENDPROC
PROC Path_50()
  MoveJ Target_80,v100,z100,imukuppi_10\WObj:=paletti_sisalasi;
  WaitDI DI_paletti_sisalasi,1;
  MoveL Target_80_2,v50,fine,imukuppi_10\WObj:=paletti_sisalasi;
  SetDO DO_imu_10,1;
  WaitTime 0.3;
  IF DI_imu_10=1 THEN
  MoveL Target_80,v100,z100,imukuppi_10\WObj:=paletti_sisalasi;
  MoveJ Target_90,v100,z100,imukuppi_10\WObj:=jigi_ruuvaus;
  MoveL Target_90_2,v50,fine,imukuppi_10\WObj:=jigi_ruuvaus;
  SetDO DO_imu_10,0;
  WaitTime 0.3;
  MoveL Target_90,v100,z100,imukuppi_10\WObj:=jigi_ruuvaus;
    IF iy_sisalasi<2 THEN
      paletti_sisalasi.uframe.trans.y:=paletti_sisalasi.uframe.trans.y+72;
      iy_sisalasi:=iy_sisalasi+1;
    ELSE
      SetDO DO_paletti_sisalasi,0;
      paletti_sisalasi.uframe.trans.y:=paletti_sisalasi.uframe.trans.y-72;
      iy_sisalasi:=1;
    ENDIF
  ELSE
    IF iy_sisalasi<2 THEN
      paletti_sisalasi.uframe.trans.y:=paletti_sisalasi.uframe.trans.y+72;
      iy_sisalasi:=iy_sisalasi+1;
      Path_50;
```

```
      ELSE
        SetDO DO_paletti_sisalasi,0;
        paletti_sisalasi.uframe.trans.y:=paletti_sisalasi.uframe.trans.y-72;
        iy_sisalasi:=1;
        Path_50;
      ENDIF
    ENDIF
  ENDPROC
  PROC Path_60()
    MoveJ Target_100,v100,z100,imukuppi_10\WObj:=paletti_ulkolasi;
    WaitDI DI_paletti_ulkolasi,1;
    MoveL Target_100_2,v50,fine,imukuppi_10\WObj:=paletti_ulkolasi;
    SetDO DO_imu_10,1;
    WaitTime 0.3;
    IF DI_imu_10=1 THEN
    MoveL Target_100,v50,fine,imukuppi_10\WObj:=paletti_ulkolasi;
    MoveJ Target_110,v100,z100,imukuppi_10\WObj:=jigi_ruuvaus;
    MoveL Target_110_2,v50,fine,imukuppi_10\WObj:=jigi_ruuvaus;
    SetDO DO_imu_10,0;
    WaitTime 0.3;
    MoveL Target_110_3,v100,z100,imukuppi_10\WObj:=jigi_ruuvaus;
    MoveJ Target_120,v100,z100,lasityokalu\WObj:=jigi_ruuvaus;
    MoveL Target_120_2,v50,fine,lasityokalu\WObj:=jigi_ruuvaus;
    MoveL Target_120_3,v5,fine,lasityokalu\WObj:=jigi_ruuvaus;
    MoveL Target_120_4,v100,fine,lasityokalu\WObj:=jigi_ruuvaus;
      IF iy_ulkolasi<2 THEN
        paletti_ulkolasi.uframe.trans.y:=paletti_ulkolasi.uframe.trans.y+90;
        iy_ulkolasi:=iy_ulkolasi+1;
      ELSE
        SetDO DO_paletti_ulkolasi,0;
        paletti_ulkolasi.uframe.trans.y:=paletti_ulkolasi.uframe.trans.y-90;
        iy_ulkolasi:=1;
      ENDIF
    ELSE
      IF iy_ulkolasi<2 THEN
        paletti_ulkolasi.uframe.trans.y:=paletti_ulkolasi.uframe.trans.y+90;
        iy_ulkolasi:=iy_ulkolasi+1;
        Path_60;
      ELSE
        SetDO DO_paletti_ulkolasi,0;
        paletti_ulkolasi.uframe.trans.y:=paletti_ulkolasi.uframe.trans.y-90;
        iy_ulkolasi:=1;
        Path_60;
      ENDIF
    ENDIF
  ENDPROC
ENDMODULE
```