SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Toni Kataja

Robottisolun suunnittelu

Opinnäytetyö
Kevät 2024
Insinööri (AMK), Automaatiotekniikka

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Tutkinto-ohjelma: Insinööri (AMK), Automaatiotekniikka

Suuntautumisvaihtoehto: Sähköautomaatio

Tekijä: Toni Kataja

Työn nimi alaotsikoineen: Robottisolun suunnittelu

Ohjaaja: Jarkko Pakkanen

Vuosi: 2024 Sivumäärä: 46

Tämän opinnäytetyön tavoitteena oli suunnitella ja ohjelmoida Core Link Oy:lle asiakasprojektiin robottisolu automatisoidulle rullakäsittelylinjastolle. Robotin tehtävänä oli noutaa kartonkinen päätylappu ja toimittaa se paperirullan päätyyn muovikäärinnän aikana. Robotiksi oli esisuunnitteluvaiheessa valittu ABB:n IRB – 6700 ja robottisolun suunnittelu sekä simulointi toteutettiin ABB:n RobotStudio-ohjelmistolla.

Opinnäytetyön alussa esitellään työn tavoitteet ja toimeksiantaja, sekä käytetyt ohjelmistot. Teoriaosuudessa käsitellään teollisuusrobotiikkaa, simulointia, tiedonsiirtoväyliä robotin ja ohjelmoitavien logiikoiden välillä, sekä robottisolusuunnittelua, kuten layoutsuunnittelua ja offline-ohjelmointia. Toteutusosuudessa käsitellään robottiohjelman rakennetta, layoutsuunnittelua, robotin sijoittelua ulottuvuustarkasteluineen, sekä robottisolusuunnittelua käytännössä.

Työn tuloksena saatiin simulointien ja läpikäyntien perusteella käyttökelpoinen ohjelma robotille. Simulointien perusteella robotti pystyy suorittamaan tehtävänsä kolmella työskentely pisteellä. Tehdyn työn tuloksia voidaan hyödyntää jatkossa robottisuunnittelussa yrityksen tulevissa asiakasprojekteissa.

[1] Asiasanat: RobotStudio, robottisolu, simulointi, offline-ohjelmointi, layout-suunnittelu

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Degree programme: Bachelor of Engineering, Automation Engineering

Specialisation: Electric Automation

Author: Toni Kataja

Title of thesis: Robotic cell design

Supervisor: Jarkko Pakkanen

Year: 2024 Number of pages: 46

The purpose of the thesis was to design and program a robotic cell for an automated roll handling line for a client project at Core Link Oy. The task of the robot was to pick up a cardboard header and deliver it to the end of the paper roll during plastic wrapping. The ABB IRB – 6700 robot had been chosen for this in the pre-design phase, and the design and simulation of the robotic cell were carried out using the RobotStudio software of ABB.

In the beginning of the thesis, the objectives of the client and the thesis were defined, and the used software was presented. The theoretical part studied industrial robotics, simulation, data transfer interfaces between the robot and programmable logics, and robotic cell design, such as layout design and offline programming. The implementation section discussed the structure of the robot program, layout design, the placement of the robot with reach analysis, and a practical robotic cell design.

As the result of the simulations and walkthroughs, a usable program for the robot was obtained. According to the simulations, the robot can perform its task at three working stations. In the future, the results of the thesis project can be utilized in robot design in the upcoming client projects of Core Link Oy.

[1] Keywords: RobotStudio, robotical cell, simulation, offline-programming, layout design

SISÄLTÖ

Kuvioluettelo

Käytetyt termit ja lyhenteet

HMI	Lyhenne sanoista Human-Machine Interface, eli käyttöliittymä, joka mahdollistaa ihmisen ja koneen vuorovaikutuksen.
I/O	Input/Output. Inputeilla tarkoitetaan signaalia, jota lähetetään laitteelle. Outputeilla tarkoitetaan laitteesta lähtevää signaalia.
Layout	Layout on termi, jolla tarkoitetaan koneiden, laitteiden, varastopaikkojen ja kulkureittien sijoittelua suunnitteluohjelmassa.
PLC	Programmable logic controller. Ohjelmoitava logiikka, jota käytetään automaatioprosessien ohjaamiseen.

1 Johdanto

1.1 Työn tausta

Opinnäytetyö tehtiin Core Link Oy:lle asiakasprojektiin. Yritys suunnittelee, ja käyttöönottaa asiakkaalle paperitehtaaseen uuden automaattisen paperirullien käsittelylinjaston, jonka yhteydessä tässä työssä suunniteltu robottisolu työskentelee.

1.2 Työn tavoite

Tavoitteena on suunnitella ja ohjelmoida mahdollisimman tehokkaasti toimiva robottisolu. Robotin tarkoituksena on asettaa paperirullan muovikäärinnän aikana rullan päätyyn kartonkinen päätylappu suojaamaan rullaa varastoinnissa sekä kuljetuksissa. Robotin yhteydessä on linjaston ja käärintälaitteiston ohella myös PLC-ohjattu päätylappukaruselli, eli pyörivä alusta erikokoisille päätylapuille, sekä keskitysasema.

1.3 Työn rakenne

Työ koostuu seitsemästä luvusta. Ensimmäisessä luvussa esitellään työn taustat ja tavoitteet, ja kerrotaan taustatietoa yrityksestä, sekä esitellään työssä käytetyt ohjelmistot. Kolmessa seuraavassa luvussa keskitytään robottisolusuunnittelun teoriaan, teollisuusrobotiikkaan, offline-ohjelmointiin ja ohjelman rakenteeseen, sekä simuloinnin teoriaan ja sen hyötyihin. Viidennessä luvussa perehdytään tarkemmin rullankäsittelylinjaston ja robottisolun toimintaan. Luvussa kuusi käsitellään ulottuvuustarkastelua, ja robotin ohjelmointia käytännössä. Viimeisessä luvussa pohditaan työn toteutusta ja sen haasteita, sekä tehdyn työn tuloksia.

1.4 Yritysesittely

Core Link Oy toimii osana laajempaa Core Link -yritysryhmää, jolla on toimipaikat Suomessa, Saksassa, Ruotsissa ja Yhdysvalloissa (Core Link AB, i.a.). Yrityksen päätoimisto sijaitsee Ruotsissa. Core Link on osa Mustad United C.O. AB:tä, joka on Ruotsissa

vuonna 1913 perustettu kansainvälinen perheomisteinen teollisuuskonserni. Konsernin liikevaihto on noin 90 miljoonaa euroa.

Core Link on paperiteolisuuden rullan- ja hylsynkäsittelyn markkinajohtaja (Core Link AB, i.a.). Core Link Oy suunnittelee ja valmistaa rullankäsittelyjärjestelmiä teollisuuteen maailmanlaajuisesti. Yrityksen palveluihin kuuluvat suunnittelu, asennukset, käyttöönotto, koulutus ja after sales -palvelut. Suomessa yrityksellä on toimipisteet Seinäjoella, Helsingissä, sekä Kauhajoella, jossa sijaitsee myös yrityksen kokoonpano ja testaushalli.

1.5 Työssä käytettävät ohjelmistot

Tässä luvussa käydään läpi työssä käytettyihin ohjelmistoihin liittyvää teoriaa.

1.5.1 RobotStudio

ABB RobotStudio on ohjelmisto, joka on suunniteltu ABB:n teollisuusrobottien ohjelmointiin, simulointiin ja offline-ohjelmointiin (ABB, 2024). ABB on maailmanlaajuinen teknologiayritys, joka valmistaa teollisuusrobottijärjestelmiä erilaisiin sovelluksiin, kuten valmistus, logistiikka ja automaatio.

RobotStudio mahdollistaa robotin ohjelmoinnin ja simuloinnin tietokoneella ennen kuin ohjelma siirretään varsinaiseen robottisoluun (ABB, 2024). Tämä vähentää käyttökatkoja ja parantaa ohjelman laatua, kun ohjelmointia voidaan testata ja optimoida virtuaalisessa ympäristössä ennen varsinaista käyttöä.

Ohjelmiston avulla käyttäjät voivat luoda, muokata ja simuloida robotin liikeratoja, ohjelmoita logiikkaa ja tarkistaa, miten robotti toimii erilaisissa skenaarioissa (ABB, 2024). Tämä auttaa optimoimaan prosesseja ja välttämään mahdolliset ongelmat ennen kuin robotti otetaan käyttöön tehtaalla tai tuotantolaitoksessa.

1.5.2 Studio 5000 Logix Designer

Studio 5000 Logix Designer on ohjelmistoalusta, joka on kehitetty Rockwell Automationin valmistamien ohjelmoitavien logiikkaohjainten (PLC) ohjelmointiin (Rockwell Automation, i.a.). Rockwell Automation on yksi johtavista teollisuusautomaation ja ohjausjärjestelmien toimittajista.

Studio 5000 Logix Designer on osa Studio 5000 -ohjelmistoperhettä, joka tarjoaa integroidun ympäristön teollisuusautomaation suunnitteluun, ohjelmointiin ja käyttöönottoon (Rockwell Automation, i.a.). Studio 5000 Logix Designerilla käyttäjät voivat ohjelmoida ja konfiguroida Logix-perheen ohjainalustoja, kuten ControlLogix ja CompactLogix, jotka ovat Rockwell Automationin valmistamia ohjelmoitavia logiikkaohjaimia.

Ohjelmistolla voidaan suunnitella ja toteuttaa ohjelmalogiikkaa, joka ohjaa erilaisia teollisuusprosesseja (Rockwell Automation, i.a.). Se mahdollistaa myös ohjelmoitavien logiikkaohjainten konfiguroinnin ja valvonnan. Studio 5000 Logix Designer sisältää useita ominaisuuksia, kuten graafisen ohjelmoinnin työkalut, simulointimahdollisuudet ja diagnostiikkatoiminnot, jotka auttavat automaatiojärjestelmän suunnittelussa ja ylläpidossa.

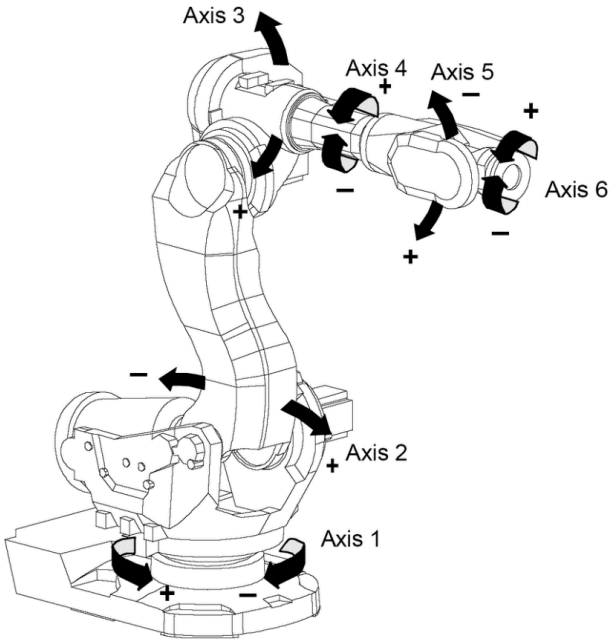
2 Teollisuusrobotiikka

Tässä luvussa käydään läpi teollisuusrobotiikkaa, ohjelmoitavia logiikoita, sekä robotin valintaan liittyvää teoriaa.

2.1 Teollisuusrobotti

Teollisuusrobotit ovat uudelleenohjelmoitavia työkappaleita käsitteleviä yleiskäyttöisiä koneita eli manipulaattoreita, joissa on yleensä kolme vapaasti ohjelmoitavaa akselia ja työkalu (Ahonen ym., 2023, 17). Tyypillisimmät rakenteet ovat käsivarsityyppisiä robotteja, joissa on viidestä seitsemään vapausastetta (degree of freedom). Määritelmissä määritellään robottien työskentelytilaksi kolme tai useampi vapausaste ja työkappaletta käsittelevän työkalun liikkeet. Loppuasiakas on useimmiten kiinnostunut suunniteltuun tehtävään valitun robotin työkappaleenkäsittely kapasiteetista, tehtävän toistettavuudesta ja työkierron nopeudesta.

Lähes 75 prosenttia teollisuusroboteista on antropomorfisia nivelrobotteja, joissa on tyypillisesti kuusi vapausastetta (kuvio 1) (Ahonen ym., 2023, 17). Ne ovat varsin yleiskäyttöisiä laajan työskentelytilansa ansiosta, ja ne voidaan asentaa lattialle, seinälle tai kattoon. Nivelrobottien käsittelykapasiteetti vaihtelee suuresti, tyypillisesti 0,5 kg:sta jopa 2 300 kg:aan, ja nivelrobottien volyymit ovat 6–30 kg:n välillä.



Kuvio 1. Kuusiakselinen robotti (ABB, 2004).

2.1.1 Robotin valinta

Kun valitaan robottia teolliseen käyttöön, on useita tekijöitä, jotka tulee ottaa huomioon, että valittu robotti vastaa parhaiten käyttötarpeita ja -vaatimuksia (Robots Done Right, i.a.). Näiden tekijöiden ymmärtäminen auttaa valitsemaan sopivan robotin, joka lisää tuotannon tehokkuutta, tarkkuutta ja laatua.

1. Kuormituskyvyn arviointi. On tärkeää tietää kuinka paljon painoa robotti voi käsitellä. Jos työkappaleet ovat raskaita, tarvitaan robotti suuremmalla kuormituskyvyllä.

2. Ulottuvuus. Robotin ulottuvuus pysty- ja vaakasuunnassa. Ulottuvuus määrittää kuinka kauas tai korkealle robotti voi ulottua suorittaessaan tehtäviään.

3. Akselien määrä. Robotin akselien määrä määrittää robotin liikkumisvapauden. Useimmat teollisuusrobotit ovat kuusiakselisia, mikä jäljittelee ihmisen käsivarren liikkuvuutta, tämä mahdollistaa kääntymisen ja kiertämisen, mikä laajentaa työskentelyaluetta.

4. Työympäristö. Robotin toimintaympäristö vaikuttaa sen sopivuuteen tehtäviin. Esimerkiksi tietyt robotit on suunniteltu toimimaan puhtaissa olosuhteissa, kun taas toiset kestävät paremmin pölyisiä tai kosteita olosuhteita.

Lisäksi on muitakin tekijöitä, kuten toistettavuus, ylikuormituskapasiteetti, tarkkuus, ja ohjelmoinnin joustavuus, jotka ovat keskeisiä robottia valittaessa (Baiju, 2021). Nämä tekijät auttavat varmistamaan, että valittu robotti ei ainoastaan vastaa nykyisiä tarpeita, vaan on myös joustava mahdollisten tulevien sovellusten suhteen.

2.2 Ohjelmoitava logiikka

Ohjelmoitava logiikka (Programmable Logic Controller, PLC) on digitaalinen tietokone, jota käytetään teollisuusympäristöissä automaation ohjaamiseen (Bolton, 2019). PLC:t on suunniteltu kestämään ankaria teollisuusympäristöjä, kuten korkeita lämpötiloja, kosteutta, pölyä ja mekaanista rasitusta. Niitä ohjataan erilaisilla ohjelmointikielillä, kuten ladderkoodilla, ja niitä käytetään monenlaisissa sovelluksissa, kuten kokoonpanolinjojen ja koneiden ohjauksessa sekä prosessien hallinnassa.

Yksi PLC:n keskeisistä eduista on sen kyky toimia reaaliajassa, joka on olennainen vaatimus monissa teollisuuden prosessien ohjauksissa (Hughes, 2020). PLC:t pystyvät lukemaan antureilta tulevat signaalit ja toteuttamaan määriteltyjä toimintoja lähes välittömästi. Tämä ominaisuus tekee niistä erinomaisen valinnan sovelluksiin, joissa ajoituksen tarkkuus on kriittinen.

3 Robottisolun suunnittelu

Tässä luvussa käydään läpi robottisolun suunnitteluun liittyvää teoriaa. Luvussa tutustutaan simulointiin ja sen hyötyihin, erilaisiin robotin työkaluihin, layout-suunnitteluun, ulottuvuustarkasteluun sekä PLC-kommunikointiin.

3.1 Simulointi suunnittelun tukena

Simuloinnin avulla robotisointiprojekteissa tarkastellaan robotin, robottisolujen ja koko tuotantojärjestelmän toiminnallisuutta (Ahonen ym., 2023, s 66). Sen avulla voidaan visualisoida ja analysoida suunnitellun prosessin yksittäisiä osia, lähtien itse prosessin ydinkohdista. Esimerkkeinä simuloinnista ovat robotin kappaleeseen tarttumisen, työkalun liikeradan, nopeuden ja lähestymiskulman simulointi työkappaleen suhteen, sekä robotin kinematiikan ja robottisolun kokonaisvaltaisen toiminnan tutkiminen. Robotti kykenee liikuttamaan työkalua tarkasti ja kontrolloidulla nopeudella.

Esimerkiksi, robotin on kyettävä navigoimaan tai siirtämään työkalu monimutkaisiin asentoihin, välttäen samalla esteitä ja vahinkoja (Ahonen ym., 2023, s 66). Simuloinnin kautta analysoidaan, miten robotin tulisi kappaleeseen tarttua ja millainen olisi työkalun ihanteellinen liikerata, nopeus ja lähestymiskulma työkappaleen kohdalla. Tämä soveltuu erinomaisesti robotin liikeratojen ennakkotarkasteluun. Robottisolun toiminnan simulointi puolestaan varmistaa, että robotti kykenee suorittamaan suunnitellut tehtävät. Simuloinnilla arvioidaan työkalujen soveltuvuus ja testataan robotin aikataulutus ja vaiheistus. Lisäksi simuloinnin avulla voidaan ennakoida ja tutkia koko tuotantojärjestelmän toimintaa.

Simuloinnin keskeinen tavoite ei ole ohjelmointi sinänsä, vaan liikeratojen ja muiden toiminnallisten aspektien tehokkuuden varmistaminen (Ahonen ym., 2023, s 67). Kuitenkin simulointimallia voidaan käyttää ohjelmoinnin perustana, erityisesti etäohjelmoinnissa. Koska robotit toimivat kolmiulotteisessa tilassa, simuloinnissa on tärkeää käyttää 3D-malleja. Tämän vuoksi 2D-työkalut eivät ole riittäviä, ja robotisoinnin suunnittelussa käytetään erityisesti suunniteltuja simulointityökaluja.

Simulointi on jatkuva prosessi, joka jatkuu koko projektin ajan, aina sen alkuvaiheista loppuun saakka (Ahonen ym., 2023, s 67). Jokaisen suunnitelmamuutoksen yhteydessä on suositeltavaa suorittaa simulointi uudelleen ja varmistaa muutosten toimivuus.

3.1.1 Simuloinnin hyödyt

Simuloinnin avulla on mahdollista testata robotin toimintaa virtuaalisessa ympäristössä ennen sen toteuttamista todellisessa tuotantoympäristössä (ABB, i.a.-a). Tämä vähentää virheiden ja laitteistovaurioiden riskiä sekä parantaa turvallisuutta. Toinen etu on nopeampi käyttöönotto. Simuloinnin avulla suunnittelijat voivat kokeilla ja hienosäätää robottien toimintaa ilman, että se häiritsee olemassa olevaa tuotantoa. Tämä nopeuttaa uusien tuotantolinjojen tai -prosessien käyttöönottoa ja lyhentää käynnistysaikoja. Kolmantena etuna on tuottavuuden kasvu. Simuloinnin avulla voidaan optimoida robotin liikeradat ja prosessit, mikä parantaa tuotannon tehokkuutta ja vähentää aikaa, joka kuluu tuotannon suunnitteluun ja toteutukseen.

3.2 Työkalut

Robotin tarttujat ovat monipuolisia laitteita, jotka mahdollistavat robottikäsivarsien vuorovaikutuksen fyysisen maailman kanssa (Universal Robots, i.a.). Tarttujat, joita myös kutsutaan end-of-arm toolingiksi (EOAT), ovat keskeisiä komponentteja teollisuusroboteissa ja ne määrittävät pitkälti robottikäsivarsien käyttötarkoitukset ja -sovellukset.

Tarttujien tyyppien kirjo on laaja, ja ne eroavat toisistaan käyttötarkoituksen, rakenteen ja käyttövoiman perusteella (Dorna Robotics, i.a.). Yleisimpiä tarttujatyyppejä ovat:

1. **Vakuumitarttujat:** Nämä tarttujat käyttävät ilmanpaine-eroa ja vakuumitekniikkaa esineiden nostamiseen, pitelemiseen ja liikuttamiseen. Ne ovat erityisen tehokkaita tasaisilla ja sileillä pinnoilla, kuten lasilla, muovilla ja metallilevyillä.

2. **Pneumaattiset tarttujat:** Tarttujat käyttävät puristettua ilmaa ja mäntiä "leukojen" tai "sormien" ohjaamiseen. Ne ovat monipuolisia ja soveltuvat monenlaisiin

sovelluksiin. Pneumaattiset tarttujat ovat tunnettuja nopeudestaan ja suuresta tartuntavoimastaan.

3. **Hydrauliset tarttujat:** Nämä tarttujat käyttävät hydraulisia nesteitä ja tarjoavat suurempaa tartuntavoimaa kuin pneumaattiset tarttujat, mikä tekee niistä ihanteellisia raskaisiin tehtäviin.

4. **Sähköiset tarttujat:** Nämä sopivat hyvin moniin yhteistyörobotin sovelluksiin, kuten koneen käsittelyyn ja poiminta- ja sijoittelutehtäviin. Ne eivät tarjoa samaa tartuntavoimaa kuin hydrauliset tarttujat, mutta sopivat kevyempiin ja keskiraskaisiin sovelluksiin.

5. **Magneettiset tarttujat:** Tarttujat käyttävät magneettikenttää rautapitoisten esineiden tarttumiseen ja käsittelyyn. Ne ovat tehokkaita, kun tarvitaan vahvaa ja vakaata otetta metalliesineistä.

6. **Sormitarttujat:** Nämä tarttujat on varustettu useilla sormilla tai nystyröillä, jotka voivat mukautua tartuttavan esineen muotoon. Ne ovat erittäin monipuolisia ja sopivat monenlaisiin automaatiotehtäviin, jotka vaativat suurta sorminäppäryyttä.

3.3 Layoutsuunnittelu

Layoutsuunnittelussa hyödynnetään usein 3D-mallinnusta ja simulaatioita, jotka mahdollistavat erilaisten asetelmien ja työskentelytapojen testaamisen virtuaaliympäristössä ennen niiden toteuttamista (Groover, 2020, s. 17). Tämä auttaa tunnistamaan mahdollisia ongelmia, kuten törmäysriskit tai tehottomat työpisteiden sijoittelut, joita voidaan sitten korjata ennen fyysisten muutosten tekemistä. Lisäksi layoutsuunnittelu tukee joustavuutta, ja mahdollistaa nopean mukautumisen tuotannon tarpeiden muuttuessa, mikä on erityisen tärkeää nopeasti muuttuvassa ja kilpailukykyisessä teollisessa ympäristössä.

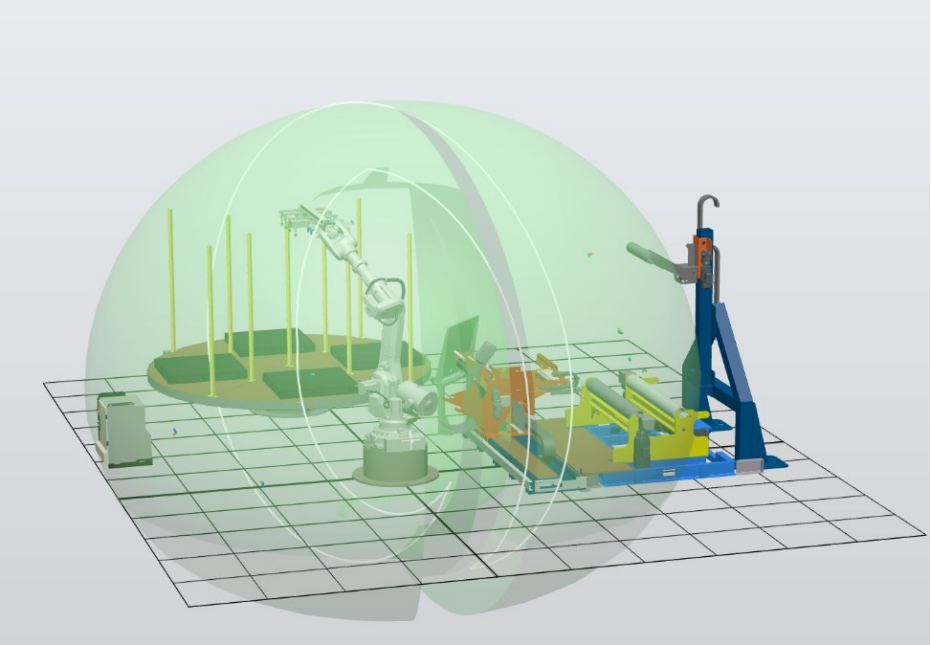
Parhaimmillaan layoutsuunnittelu voi auttaa lopullisissa laitevalinnoissa, esimerkiksi valitsemaan robotin, jolla on riittävä ulottuvuus, tai oikean korkuisen robottijalustan (Ahonen ym., 2023, s. 190). Robottisolun layoutsuunnittelun osalta on oleellista varmistaa, että

kaikki tarvittavat työkappaleet, kiinnittimet, syöttölaitteet ja muut tarvittavat komponentit ovat robotin ulottuvilla. Yksi tärkeimmistä tehtävistä layoutsuunnittelussa on varmistaa, että robotti ei törmää mihinkään, että sillä on riittävästi tilaa liikkua eikä se törmää ympäröiviin koneisiin tai muihin komponentteihin. Simulointiohjelmistoissa voidaan määritellä törmäyspintoja, jotka aktivoituvat törmäyksen tai läheltä piti -tilanteen yhteydessä ja pysäyttävät simuloinnin törmäyksen merkiksi.

3.4 Ulottuvuustarkastelu

Ulottuvuustarkastelu (kuvio 2) sisältää robottien ja muiden automaattisten laitteiden toiminta-alueen analysoinnin, jotta varmistetaan niiden kyky saavuttaa ja käsitellä eri työkappaleita ja komponentteja tehokkaasti (Shimon, 2020, s. 82). Ulottuvuustarkastelun tavoitteena on optimoida robottien sijoittelu ja liikeradat siten, että ne voivat suorittaa tehtävänsä ilman esteitä ja mahdollisimman vähällä liikkeellä, mikä lisää tuotantolinjan tehokkuutta ja vähentää ajanhukkaa.

- Ulottuvuustarkastelun tarkoituksena on varmistaa, että robotti pystyy saavuttamaan ja käsittelemään tarvittavia työkappaleita ja komponentteja tehokkaasti, ja että se mahtuu työskentelemään annetussa tilassa ilman esteitä (Makhal & Goins, 2017, s. 5).
- Ulottuvuustarkastelussa käytetään tyypillisesti 2D- ja 3D-simulointityökaluja, jotka auttavat visualisoimaan robotin liikeratoja ja ulottuvuuksia (Makhal & Goins, 2017, s. 7).
- Prosessi auttaa tunnistamaan ja ennalta ehkäisemään potentiaalisia ongelmia, kuten nivelkonfiguraatio- ja singulariteettiongelmia, jotka voivat syntyä tietyissä työasennoissa tai -liikkeissä (Makhal & Goins, 2017, s. 2).
- Lopullinen ulottuvuusanalyysi tehdään yleensä osana robottiohjelman kokonaissimulointia, mikä auttaa varmistamaan sujuvan ja tehokkaan toiminnan (Makhal & Goins, 2017, s. 7).



Kuvio 2. Robotin ulottuvuus visuaalisesti.

3.5 PLC- kommunikointi

Robottien ja ohjelmoitavien logiikkaohjainten (PLC) välisessä kommunikaatiossa yleisesti käytettävät väylät ja protokollat ovat Ethernet/IP, PROFINET, Modbus, DeviceNet ja EtherCAT (FS Community, 2023). Nämä protokollat mahdollistavat eri laitteiden, kuten robottien ja PLC-ohjainten, välisen tehokkaan tiedonsiirron ja kommunikaation erilaisissa teollisuusympäristöissä.

1. **Ethernet/IP** on laajalti hyväksytty teollisuusstandardi, joka hyödyntää Ethernet-verkkoa tarjotakseen luotettavan kommunikaation laitteiden välillä (FS Community, 2023). Ethernet/IP-standardia hallinnoi ODVA (Open DeviceNet Vendor Association) ja se tukee sekä reaaliaikaista että ei-reaaliaikaista kommunikaatiota.

2. **PROFINET** on Siemensin kehittämä Ethernet-pohjainen protokolla, joka tarjoaa reaaliaikaisen kommunikaation ja tukee joustavia verkkotopologioita (FS Community, 2023). PROFINET on suunniteltu integroitumaan saumattomasti olemassa oleviin järjestelmiin ja tukee laajaa valikoimaa laitteita.

3. **Modbus**, joka kehitettiin alun perin Schneider Electricin toimesta, on yksinkertainen ja joustava protokolla, joka tukee sekä sarja- että TCP/IP-yhteyksiä (FS Community, 2023). Sen yksinkertaisuus ja helppo toteutus tekevät siitä suositun valinnan monenlaisiin sovelluksiin.

4. **EtherCAT** on Beckhoff Automationin kehittämä protokolla, joka on erikoistunut tarjoamaan erittäin nopeaa ja reaaliaikaista tiedonsiirtoa käyttäen master-slave-arkkitehtuuria (FS Community, 2023).

5. **DeviceNet** on avoin protokolla, joka keskittyy teollisuusympäristöjen laitteiden välisten yhteyksien luomiseen, ja se on suunniteltu helpottamaan laitteiden integrointia ja kommunikaatiota verkossa (Encoder.com, 2023).

4 Offline-ohjelmointi

Offline-ohjelmointi on keskeinen osa nykyaikaista teollisuusrobotiikkaa ja tarjoaa merkittäviä etuja verrattuna perinteiseen online-ohjelmointiin (Siciliano ym., 2009. s. 34). Tämä ohjelmointitapa mahdollistaa robotin ohjelmien kehittämisen ilman, että itse robottia tarvitsee käyttää fyysisesti, mikä säästää aikaa ja vähentää tuotannon keskeytyksiä. Opinnäytetyössä käytetty ABB RobotStudio on kehittynyt työkalu, joka mahdollistaa offline-ohjelmoinnin käytön monipuolisesti. Sen avulla voidaan simuloida robotin toimintaa virtuaalisesti, jolloin ohjelmointi, testaus ja optimointi voidaan suorittaa ilman fyysistä robottia. Tämä ei ainoastaan paranna ohjelmointiprosessin tehokkuutta, vaan myös mahdollistaa monimutkaisten ja tarkkuutta vaativien tehtävien suorittamisen vähemmillä virheillä ja suuremmalla tarkkuudella.

4.1 Ohjelman rakenne

Kun simulaatiomalli valitulla robottityypillä ulottuvuustarkasteluineen on testattu, voidaan aloittaa ohjelman laatiminen (Ahonen ym., 2023, s 172).

Robotin ohjelmoinnin tehtävälista:

1. Pisteiden, signaalien, muuttujien ja muiden tietojen määrittely ja nimeäminen.
2. Tiedon dokumentointi, nimi, tarkoitus, (logiikkaosoite) jne.
3. Virheenkäsittelijän määrittely, mitä tehdään virhetilanteessa?
4. Taustatehtävien, kuljettimien, viestinnän jne. määrittäminen.
5. Ohjelmakoodin kirjoittaminen, alusta alkaen kirjoitettuna tai valmiita moduuleja kierrättämällä.
6. Ohjelman kommentointi, jotta muut käyttäjät ymmärtävät, miten se toimii.
7. Ohjelman ja oheislaitteiden ja niiden ohjauksen dokumentointi.

4.2 World Zones

World zonet ovat määriteltyjä alueita robotin toiminta-alueella, jotka mahdollistavat tiettyjen toimintojen suorittamisen tai estävät törmäykset laitteiston tai muiden robottien kanssa, kun robotti saavuttaa nämä alueet (ABB, i.a.-c). Ne voidaan määritellä joko koordinaattien (robtarget-tyyppisen datan) tai nivelkulmien perusteella. Geometrisesti world zonet voivat olla laatikon (WZBoxDef), sylinterin (WZCylDef) tai pallon (WZSphDef) muotoisia. World zonet määritellään RAPID-ohjelmointikielessä, ja ne voivat kytkeä päälle digitaalisen ulos-tulon tai pysäyttää liikkeen, kun robotin työkalun keskipiste (TCP) saavuttaa määritellyn alueen. Ne toimivat sekä ohjelman suorituksen aikana että manuaalisessa ohjauksessa.

World zonet ovat hyödyllisiä ilmoittamaan ohjelmoitavalle logiikkasäätimelle (PLC) robotin sijainnista ohjelman suorituksen aikana (ABB, i.a.-c). Nämä määrittelyt voivat myös auttaa palauttamaan robotin tunnettuun turvalliseen asentoon. Jokainen world zone vaatii kaksi datatyyppiä: wzstationary (työalueen määrittely robotille) ja shapedata (alueen koko).

Turvallisuuden kannalta world zonet voidaan myös määrittää osana Safemove-toimintoa, joka mahdollistaa turvallisuusvyöhykkeiden luomisen robotin ympärille ja rajoituksia tietyille nivelkulmille (Control Automation, 2022).

4.3 Natiiviympäristö

Natiivissa ohjelmointiympäristössä kehitetyt robottiohjelmistot on suunniteltu tietylle laitteistolle tai käyttöjärjestelmälle, kuten ABB:n RobotStudio, mikä mahdollistaa paremman integraation ja optimoinnin kehitysympäristön ja robotin välille (Raygun Blog, 2023; Uptech, 2023). Tämä tarkoittaa, että ohjelmistot voivat hyödyntää laitteiston täyttä kapasiteettia ja tarjota nopeampaa suorituskykyä sekä parempaa vakautta. Natiivi kehitys on erityisen hyödyllistä robottiohjelmoinnissa, sillä se mahdollistaa tarkan hallinnan robotin liikkeistä ja toiminnoista, jotka ovat kriittisiä monimutkaisten ja tarkkuutta vaativien tehtävien suorittamisessa. Lisäksi natiivi kehitys tukee paremmin erikoistuneita toimintoja, kuten sensorien tarkkaa lukemista ja reaaliaikaista käsittelyä, mikä on olennaista robotiikassa.

4.4 Universaali ympäristö

Universaali ohjelmointiympäristö tarjoaa laajan yhteensopivuuden ja joustavuuden, mikä on hyödyllistä robottiohjelmoinnissa, kun on tarve tukea erilaisia laitteistoja ja käyttöjärjestelmiä (Built In, 2023). Tämä mahdollistaa sovellusten kehittämisen, mitkä voivat toimia useilla eri alustoilla, mikä vähentää kehityskustannuksia ja nopeuttaa kehitysprosessia. Universaali ympäristö on erityisen hyödyllinen tilanteissa, joissa on tarve integroida robotteja erilaisiin järjestelmiin tai kun halutaan varmistaa ohjelmiston tuleva yhteensopivuus uusien teknologioiden kanssa. Tämä joustavuus on tärkeää erityisesti dynaamisissa ja nopeasti kehittyvissä teollisuusympäristöissä, joissa teknologiat ja vaatimukset voivat nopeasti muuttua.

5 Yleiskuvaus linjastosta

Työ käsittelee rullankäsittelylinjaston yhteydessä toimivan robottisolun suunnittelua. Robottisolun on tarkoitus asettaa paperirulliin kartonkinen päätylappu automaattisesti. Seuraavassa on kuvattuna rullankäsittelylinjaston toimintaa tarkemmin.

Rullankäsittelylinjasto sisältää useita eri laitteita ja työvaiheita. Rulla saapuu valmistuksesta yksittäisenä rullana tai useamman rullan nipuissa. Rullat keskitetään keskitysasemalla, jonka jälkeen rullat mitataan ja punnitaan. Mittauksien jälkeen rullat kääritään kahteen kertaan, ensin aksiaalisesti ja tämän jälkeen radiaalisuuntaisesti, tämän yhteydessä tässä työssä suunnitellun robottisolun on tarkoitus asettaa päätylappu rullan toiseen päätyyn. Käärintöjen jälkeen rullien kylkeen asetetaan etiketit etiketöintiasemalla. Etiketöinnin jälkeen rullat käännetään pystyasentoon, jolloin asennettu päätylappu jää alapuolelle suojaamaan rullanpäätyä. Pystyyn noston jälkeen rulla siirretään kuljettimia pitkin linjaston loppuun, josta rullat noudetaan trukilla varastoitavaksi.

5.1 Robotin toimintakuvaus

Robotti poimii oikean kokoisen kartonkisen päätylapun päätylappukarusellista. Poiminta tapahtuu robotin alipainetarttujalla. Tämän jälkeen robotti vie päätylapun keskitysasemalle, jossa se keskitetään tarttujan keskelle. Kun radiaalikäärinnän sekvenssi on oikeassa vaiheessa robotti toimittaa päätylapun käärintäasemalle ja asettaa sen rullaa vasten toiseen päätyyn, imu katkaistaan ja rullassa pidin työntyy pitämään lapun tukevasti rullan ja tarttujan välissä. Kun päätylappu on kertaalleen koko kehän pituudelta käärityn muovikääreeseen, robotti poistuu takaisin kotiasemaan odottamaan seuraavaa tehtävää.

Robottisolun tärkeimmät osat ovat:

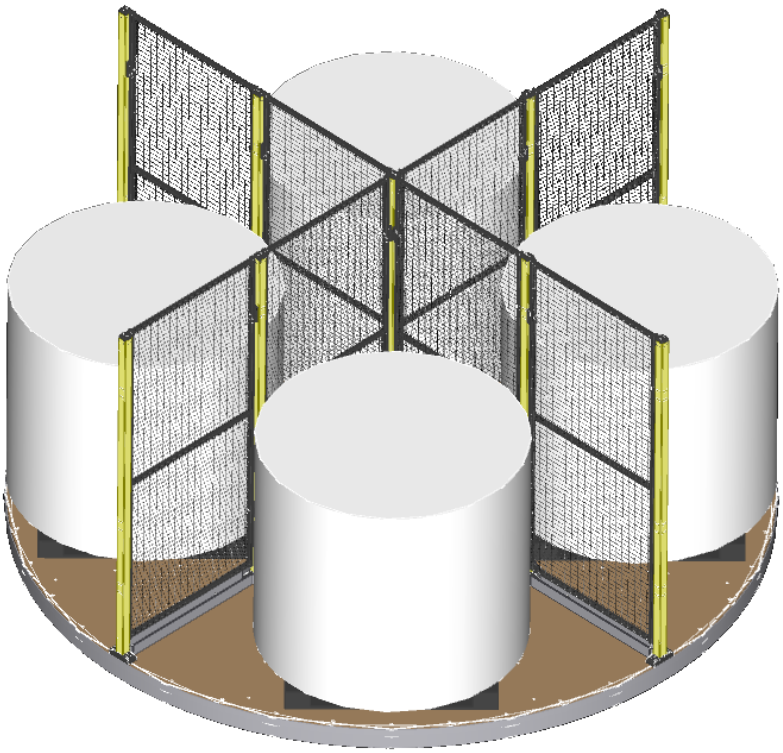
1. Robotti päätylappujen käsittelyyn
2. Yksipuolinen alipainetarttuja
3. Päätylappukaruselli neljälle päätylappupinolle

4. Keskitysasema

5. Käärintälaitteisto

5.2 Päätylappukaruselli

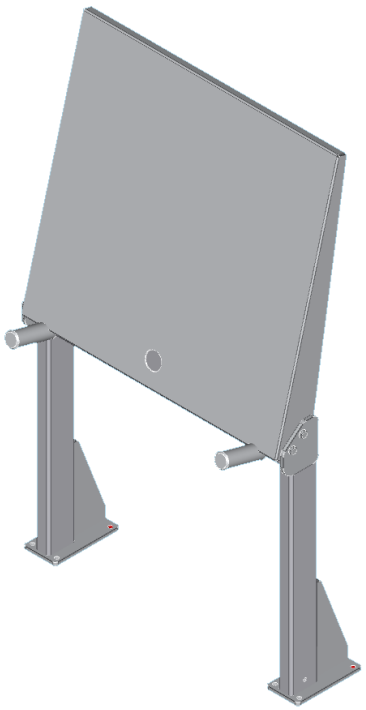
Päätylappukarusellissa (kuvio 3) on neljä paikkaa erikokoisille päätylapuille. Kullekin paikalle on oma muistipaikkansa PLC-ohjelmassa. Täyttäessään karusellia operaattori tallentaa HMI-paneelilta päätylapun halkaisijan. Kun robotti on suorittanut edellisen tehtävän robotti ilmoittaa PLC:lle olevansa valmiina uuteen tehtävään. PLC lukee seuraavan rullan tiedot ja laskee minkä kokoisen lapun rulla tarvitsee, ja jos sellainen löytyy karusellista, karuselli pyörähtää oikeaan asentoon ja antaa robotille luvan hakea lapun. Mikäli oikean kokoista päätylappua ei löydy, antaa ohjelma hälytyksen paneelille, että operaattori tietää lastata oikean kokoisen päätylappupinon karuselliin ja toiminta voi jatkua.



Kuvio 3. Päätylappukaruselli.

5.3 Keskitysasema

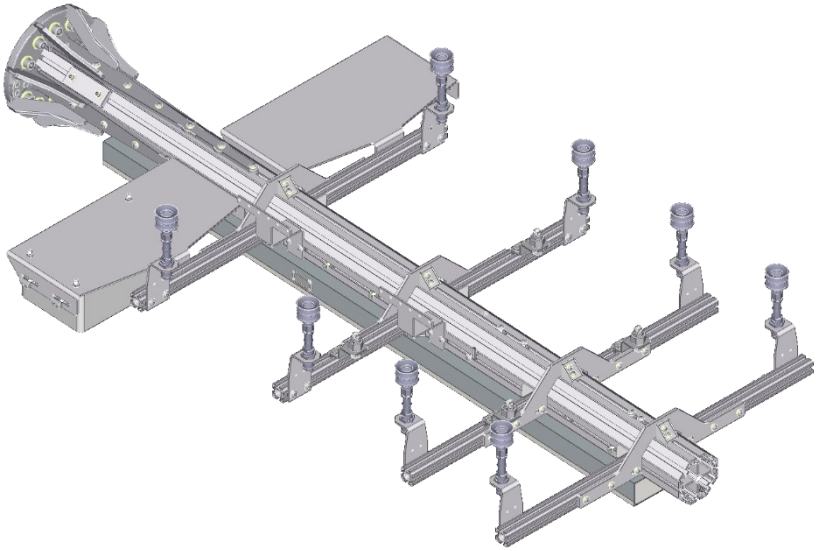
Keskitysaseman (kuvio 4) tarkoituksena on keskittää poimittu päätylappu, sillä karusellissa olevat päätylaput eivät aina täsmälleen oikeassa paikassa. Robotti toimittaa päätylapun keskitysasemalle, laskee keskikohdan halkaisijan perusteella ja poimii lapun lasketusta keskikohdasta, jotta se asettuisi rullan päätyyn keskelle. Keskitysasema on varustettu valokennolla, että saadaan tieto, onko päätylappu oikealla paikalla, tai tippunut keskityksen aikana.



Kuvio 4. Keskitysasema.

5.4 Työkalu

Robotin työkaluna toimii tähän tarkoitukseen suunniteltu tarttuja (kuvio 5). Tarttujassa on kahdeksan imupistettä, joilla päätylappu saadaan kiinni tarttujaan. Tarttujassa on myös pinonlähestymisanturi, törmäystunnistimet, imukuppien painumista valvovat induktiiviset anturit sekä pneumaattinen sylinteri päätylapun pitämiseksi rullaa vasten.



Kuvio 5. Robotin tarttuja.

5.5 Robotin valinta

Tässä työssä käytetään ABB:n robottia. Robotiksi valittiin IRB 6700–155/2.85-mallin robotti. Robotti on käsivarsirobotti (ks. s. 11-12), jonka työkalupisteen ulottuvuus on 2,85 metriä ja maksimikuormitus työkalun kanssa 155 kilogrammaa. ABB:n (i.a.-b) mukaan IRB 6700-tuoteperhe on 150–300 kilogramman luokan suorituskykyisin robotti (kuva 1).

Kuva 1. IRB-6700 (ABB, i.a.-b).

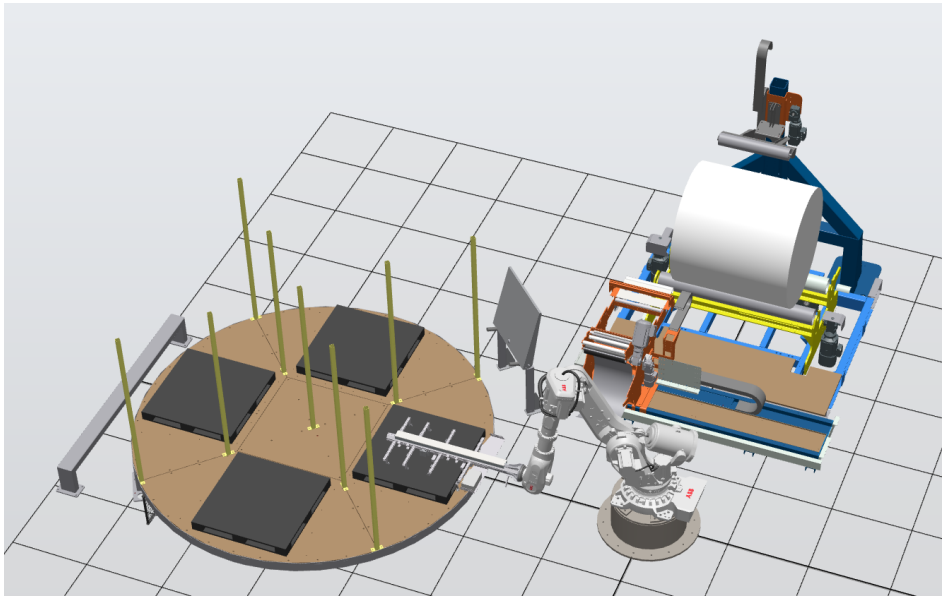
6 Työn toteutus

Työn toteutus alkaa layoutin tuonnilla RobotStudioon, työkalun luonnilla, työkalupisteen määrittelyllä ja käyttäjäkoordinaatistojen luonnilla. Tämän jälkeen suoritettiin ulottuvuustarkastelu. Ulottuvuustarkastelun jälkeen aloitettiin robotin ohjelman suunnittelu sekvenssikaavion luomisella (kuvio 6), jonka jälkeen aloitettiin I/O-määrittely ja ohjelmointi, sekä lopuksi suoritettiin simulointi. Tässä luvussa kerrotaan tarkemmin työvaiheista.

Kuvio 6. Robotin sekvenssikaavio.

6.1 Layoutin muodostus

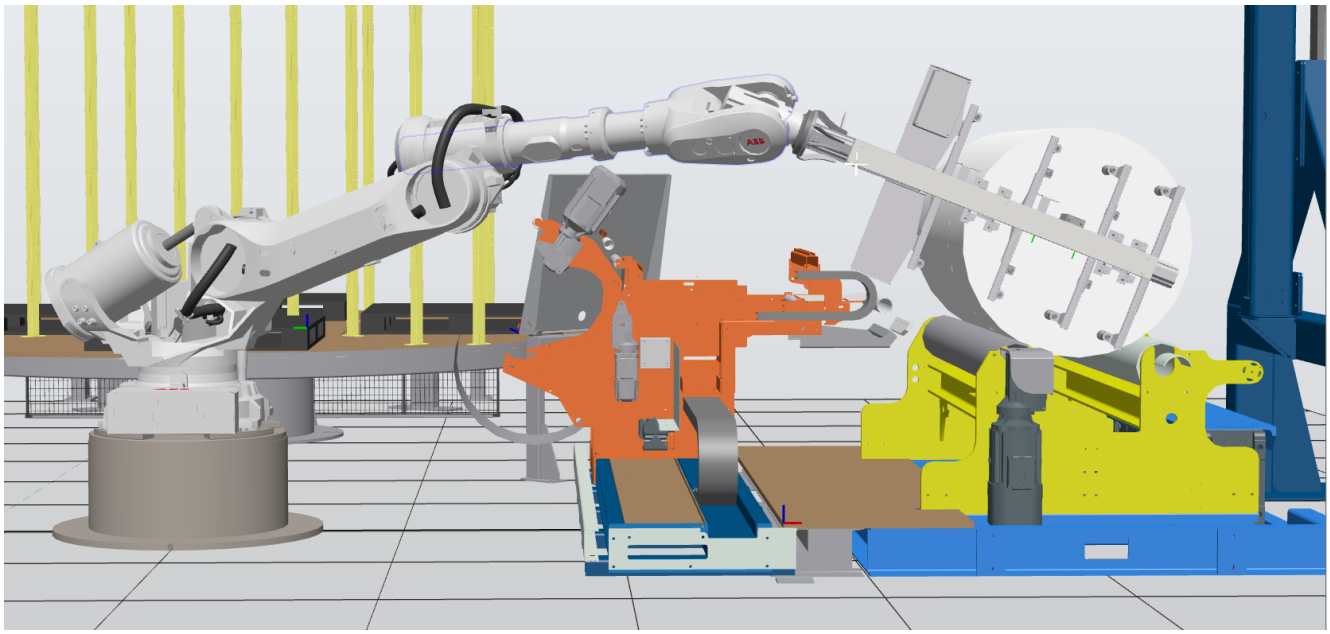
Kaikista robottisolun laitteista oli olemassa valmiit mekaniikkasuunnittelijoiden suunnittelemat 3D-mallit. Robottisolun laitteistoista tuotiin RobotStudioon kokonaisena mallina, näin mittasuhteet säilyvät oikeina, eikä laitteita tarvitse sijoitella tarkasti omille paikoilleen. Tämä säästää aikaa huomattavasti, sekä vähentää virheiden riskiä sijoittelussa. Kuviossa 7 on RobotStudioon tuotu layout robottisolusta.



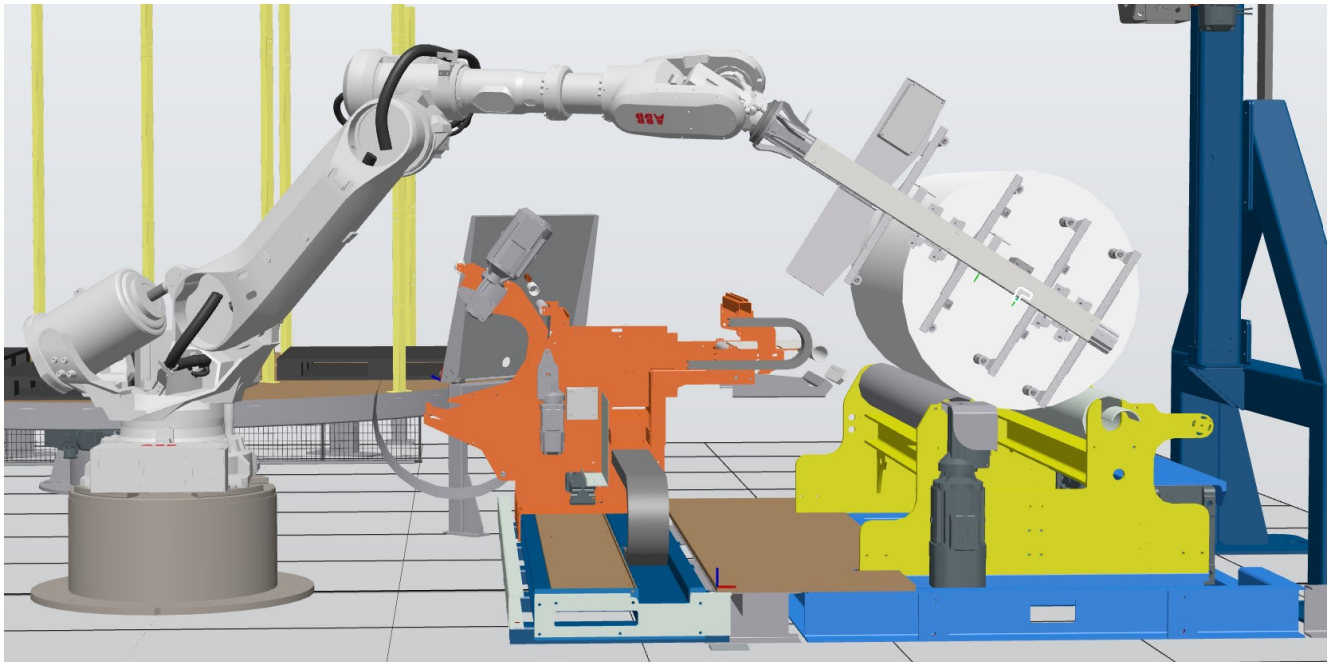
Kuvio 7. Robotin layout.

6.2 Ulottuvuustarkastelu

Robotin ulottuvuustarkastelussa (ks. s. 17) havaittiin ongelma (kuvio 8). Robotti joutui kurkottamaan ylettyäkseen rullan päätyyn ja osui käärintälaitteiston sähkömoottorin koppaan tietyissä tilanteissa. Robottia kokeiltiin siirtää useisiin eri pisteisiin, joista robotti yltäisi kaikkiin työpisteisiin ilman törmäämisriskiä eikä haittaisi tuotannon toimintaa. Tällainen löytyi ja robottia päädyttiin siirtämään 350 mm lähemmäksi käärintälaitteistoa (kuvio 9). Tältä kohdalta robotti pystyi suorittamaan tehtävänsä karusellilla, keskitysasemalla ja rullan päädyssä. Haluttiin myös varmistaa, että jos asiakkaan tarvitsee tulevaisuudessa asentaa päätylaput molempiin rullanpäätyihin, on myös tämä mahdollista suorittaa siirretystä paikasta. Ulottuvuustarkastelua helpotti käyttäjäkoordinaatistojen luonti robotin työskentelypisteille. Koordinaatistot voidaan sitoa suoraan layoutiin kiinni, ja niiden koordinaatiopisteet siirtyvät layoutin siirron mukana. Tällöin ei tarvitse siirtojen yhteydessä muokata myös robotin paikkapisteitä.



Kuvio 8. Ulottuvuustarkastelussa havaittu ongelmakohta.



Kuvio 9. Ulottuvuustarkastelu robotin uudelta positiolta.

6.3 Robotin I/O-signaalit

Robotille tulee tarttujalta ja keskitysasemalta yhteensä 8 input-tietoa:

1. Collision detector 1 ja 2. Nämä ovat pinon lähestymisantureita, jotka tunnistavat, jos pino on väärässä paikassa.

2. Board head detection. Tämä tunnistaa, onko lappu tarttujassa, anturi säädetään tunnistamaan imukuppien tasalle.

3. Stack detection. Pinon lähestymisanturi. Säädetään tunnistamaan noin 300 mm päästä tarttujasta. Tällä voidaan esimerkiksi asettaa robotti hidastamaan vauhtia ennen pinoa.

4. Delivery stop. Tällä saadaan tieto, että imukuppi on painunut.

5. Header holder front & rear limit. Lapun rullassapitimien päätyanturit. Näillä saadaan tieto, onko lapun päätylapun pidin edessä vai takana.

6. Board is at stand. Tämä signaali kertoo, onko keskitysasemalla lappua vai ei.

Robotilta tarttujalle lähtee myös kolme output-tietoa:

1. Suction area on. Asettaa robotin tarttujan imun päälle.

2. Header holder, reverse. Tarttujan päätylapun pidin taka-asentoon.

3. Header holder, forward. Tarttujan päätylapun pidin etuasentoon.

Kuviossa 10 esitellään robotille logiikalta ja tarttujalta tulevat input-tiedot.

	Name	Type	Value	Min Value	Max Value	Simulated	Network	Device	Device Mapping	Category
1	AutomaticMode	DI	1	0	1	No	SC_Feedback_Net	SC_Feedback_Dev	1	SC_Feedback
0	di1_SysStartAtMain	DI	0	0	1	No	EtherNetIP	PLC	0	
0	di2_SysStart	DI	0	0	1	No	EtherNetIP	PLC	1	
0	di3_SysStop	DI	0	0	1	No	EtherNetIP	PLC	2	
0	di4_SysMotorsOn	DI	0	0	1	No	EtherNetIP	PLC	3	
0	di5_SysMotorsOff	DI	0	0	1	No	EtherNetIP	PLC	4	
0	di6_SysResetTaskErr	DI	0	0	1	No	EtherNetIP	PLC	5	
0	di7_SysResetEStop	DI	0	0	1	No	EtherNetIP	PLC	6	
0	di8_SysReserve	DI	0	0	1	No	EtherNetIP	PLC	7	
0	di9_AlarmReset	DI	0	0	1	No	EtherNetIP	PLC	8	
0	di10_NewValuesReady	DI	0	0	1	No	EtherNetIP	PLC	9	
0	di11_PermFromRevolver	DI	0	0	1	No	EtherNetIP	PLC	10	
0	di12_PermToDeliver	DI	0	0	1	No	EtherNetIP	PLC	11	
0	di13_PileFilled	DI	0	0	1	No	EtherNetIP	PLC	12	
0	di14_DeliveredAck	DI	0	0	1	No	EtherNetIP	PLC	13	
0	di15_ProductionStop	DI	0	0	1	No	EtherNetIP	PLC	14	
0	di16_Reserve	DI	0	0	1	No	EtherNetIP	PLC	15	
0	di17_RejectHead	DI	0	0	1	No	EtherNetIP	PLC	16	HMI
0	di18_ToServicePos	DI	0	0	1	No	EtherNetIP	PLC	17	HMI
0	di19_SuctionOn	DI	0	0	1	No	EtherNetIP	PLC	18	HMI
0	di20_HolderReverse	DI	0	0	1	No	EtherNetIP	PLC	19	HMI
0	di21_HolderForward	DI	0	0	1	No	EtherNetIP	PLC	20	HMI
0	dir1_ColDetect1	DI	0	0	1	No	EtherNetIP	Local_IO	0	
0	dir2_ColDetect2	DI	0	0	1	No	EtherNetIP	Local_IO	1	
0	dir3_HeadDetection	DI	0	0	1	No	EtherNetIP	Local_IO	2	
0	dir4_StackDetection	DI	0	0	1	No	EtherNetIP	Local_IO	3	
0	dir5_DeliveryStop	DI	0	0	1	No	EtherNetIP	Local_IO	4	
0	dir6_Reserve	DI	0	0	1	No	EtherNetIP	Local_IO	5	
0	dir7_Reserve	DI	0	0	1	No	EtherNetIP	Local_IO	6	
0	dir8_Reserve	DI	0	0	1	No	EtherNetIP	Local_IO	7	
0	dir9_Reserve	DI	0	0	1	No	EtherNetIP	Local_IO	8	
0	dir10_HolderFrontLimit	DI	0	0	1	No	EtherNetIP	Local_IO	9	
0	dir11_HolderRearLimit	DI	0	0	1	No	EtherNetIP	Local_IO	10	
0	dir12_Reserve	DI	0	0	1	No	EtherNetIP	Local_IO	11	
0	dir13_Reserve	DI	0	0	1	No	EtherNetIP	Local_IO	12	
0	dir14_Reserve	DI	0	0	1	No	EtherNetIP	Local_IO	13	
0	dir15_Reserve	DI	0	0	1	No	EtherNetIP	Local_IO	14	
0	dir16_HeadAtStand	DI	0	0	1	No	EtherNetIP	Local_IO	15	

Kuvio 10. Robotin inputit.

I/O-signaalien avulla robotti kykenee toimimaan solun muiden laitteiden kanssa synkronoidusti. Robotin ja PLC:n välinen kommunikointi tapahtuu Ethernet/IP-yhteydellä (ks. s. 18-19). Kuviossa 11 määritellyiden system inputtien avulla voidaan esimerkiksi käynnistää ja sammuttaa robotti myös PLC-ohjelmassa. System inputit ovat ABB-robotin järjestelmä tuloja.

Kuviossa 11 on esitetty robotilta logiikalle sekä tarttujalle lähtevät output-tiedot.

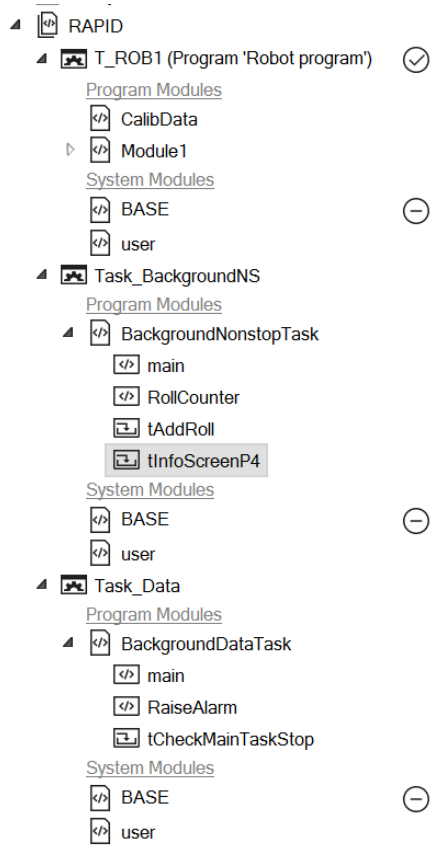
	Name	Type	Value	Min Value	Max Value	Simulated	Network	Device	Device Mapping	Category
0	do1_SysEStop	DO	0	0	1	No	EtherNetIP	PLC	0	
1	do2_SysMotOffState	DO	1	0	1	No	EtherNetIP	PLC	1	
0	do3_SysMotOnState	DO	0	0	1	No	EtherNetIP	PLC	2	
1	do4_SysRunChainOK	DO	1	0	1	No	EtherNetIP	PLC	3	
1	do5_SysAutoON	DO	1	0	1	No	EtherNetIP	PLC	4	
0	do6_SysMotSupTrig	DO	0	0	1	No	EtherNetIP	PLC	5	
0	do7_SysMainTaskRun	DO	0	0	1	No	EtherNetIP	PLC	6	
0	do8_SysMainTaskErr	DO	0	0	1	No	EtherNetIP	PLC	7	
1	do9_SysDataTaskRun	DO	1	0	1	No	EtherNetIP	PLC	8	
0	do10_SysDataTaskErr	DO	0	0	1	No	EtherNetIP	PLC	9	
0	do11_BGTaskRun	DO	0	0	1	No	EtherNetIP	PLC	10	
0	do12_BGTaskErr	DO	0	0	1	No	EtherNetIP	PLC	11	
0	do13_SysCycleON	DO	0	0	1	No	EtherNetIP	PLC	12	
0	do14_SysTempWarning	DO	0	0	1	No	EtherNetIP	PLC	13	
0	do15_SysReserve	DO	0	0	1	No	EtherNetIP	PLC	14	
0	do16_HeadDelivered	DO	0	0	1	No	EtherNetIP	PLC	15	
0	do17_PermissionToRoll	DO	0	0	1	No	EtherNetIP	PLC	16	
0	do18_PermToMoveRevolver	DO	0	0	1	No	EtherNetIP	PLC	17	
0	do19_PileLevelUpdated	DO	0	0	1	No	EtherNetIP	PLC	18	
0	do20_ProductionStopOn	DO	0	0	1	No	EtherNetIP	PLC	19	
0	do21_ReadyForNewValues	DO	0	0	1	No	EtherNetIP	PLC	20	
0	do22_ReadyToDeliver	DO	0	0	1	No	EtherNetIP	PLC	21	
0	do23_ResetAlarmAck	DO	0	0	1	No	EtherNetIP	PLC	22	
0	do24_TaskInProgress	DO	0	0	1	No	EtherNetIP	PLC	23	
0	do25_RobAtRollArea	DO	0	0	1	No	EtherNetIP	PLC	24	
0	do26_HeadDropped	DO	0	0	1	No	EtherNetIP	PLC	25	
0	do41_ColDetect1	DO	0	0	1	No	EtherNetIP	PLC	40	HMI
0	do42_ColDetect2	DO	0	0	1	No	EtherNetIP	PLC	41	HMI
0	do43_HeadDetection	DO	0	0	1	No	EtherNetIP	PLC	42	HMI
0	do44_StackDetection	DO	0	0	1	No	EtherNetIP	PLC	43	HMI
0	do45_DeliveryStop	DO	0	0	1	No	EtherNetIP	PLC	44	HMI
0	do46_HolderFrontLimit	DO	0	0	1	No	EtherNetIP	PLC	45	HMI
0	do47_HolderRearLimit	DO	0	0	1	No	EtherNetIP	PLC	46	HMI
0	do48_HeadAtStand	DO	0	0	1	No	EtherNetIP	PLC	47	HMI
0	do_wzAboveRevolverArea	DO	0	0	1	No	EtherNetIP	PLC	26	Worldzone
0	do_wzAboveRoll	DO	0	0	1	No	EtherNetIP	PLC	27	Worldzone
0	do_wzAboveStandArea	DO	0	0	1	No	EtherNetIP	PLC	28	Worldzone
0	do_wzCenteringArea	DO	0	0	1	No	EtherNetIP	PLC	29	Worldzone
0	do_wzHeadDropArea	DO	0	0	1	No	EtherNetIP	PLC	30	Worldzone
0	do_wzPileArea	DO	0	0	1	No	EtherNetIP	PLC	31	Worldzone
0	do_wzRevolverArea	DO	0	0	1	No	EtherNetIP	PLC	32	Worldzone
0	do_wzRobotAtHome	DO	0	0	1	No	EtherNetIP	PLC	33	Worldzone
0	do_wzRollArea	DO	0	0	1	No	EtherNetIP	PLC	34	Worldzone
1	do_wzServiceArea	DO	1	0	1	No	EtherNetIP	PLC	35	Worldzone
0	do_wzWrapArea	DO	0	0	1	No	EtherNetIP	PLC	36	Worldzone
0	dor1_SuctionArea1	DO	0	0	1	No	EtherNetIP	Local_IO	0	
0	dor2_HolderReverse	DO	0	0	1	No	EtherNetIP	Local_IO	1	
0	dor3_HolderForward	DO	0	0	1	No	EtherNetIP	Local_IO	2	

Kuvio 11. Robotin outputit.

Output-tietojen avulla robotti keskustelee PLC-logiikan kanssa ja käskyttää tarttujan laitteita, kuten imua ja rullassa pitimen sylinteriä.

6.4 Robotin ohjelmointi

Robotin ohjelma jaettiin kolmeen eri tehtävään eli ”taskiin” (kuvio 12). Tehtävät jaettiin seuraavasti: Yksi suorittamaan robotin ohjelmaa, ja kaksi semistaattista tausta-tehtävää suorittamaan taustalla hälytyksien valvontaa, sekä robotin laskureita (kuvio 13) jotka laskevat käärittyjen rullien lukumäärää, sekä robotin työkierron suorittamiseen kuluneen ajan keskiarvoa. Nämä ovat asiakasta kiinnostavia tietoja, joita halutaan kerätä ja näyttää. Taustataskit suorittavat tauotta ohjelmaansa taustalla häiritsemättä robotin pääohjelmaa, näiden avulla robotti voi tehdä useita tehtäviä samanaikaisesti. Kuviossa (kuvio 12) on esitetty robotin tehtäväpuu.



Kuvio 12. Robotin tehtäväpuu.

```
IF nInfoScreen=0 THEN
    TPErase;
    TPWrite "Core Link Oy " + CDate() +" "+ CTime();
    TPWrite "Info Screen 1/4";
    TPWrite "Last Roll Done ["+ strLastRollClocktime+"]";
    TPWrite "Roll Count Total:............" \Num := nRollCount;
    TPWrite "Roll Count 24h:..............." \Num := nRollCountLast24h;
    TPWrite "Roll Count Within An Hour:..." \Num := nRollCountWithinAnHour;
    TPWrite "Last Cycletime:........." \Num := nCycletime;
    TPWrite "AVG Cycletime:..........." \Num := nAvgCycleTime;
    TPWrite "Best Count Per In An Hour:..." \Num := nBestRollCountPerHour;

ELSEIF ninfoscreen=1 THEN
    TPErase;
    TPWrite "Info Screen 2/4";
    TPWrite "Roll Count Today............." \Num := nRollCountTodaY;
    TPWrite "Roll Count Yesterday........." \Num := nRollCountYesterday;
    TPWrite "Roll Count In This Week......" \Num := nRollCountThisWeek;
    TPWrite "Roll Count In Last Week......" \Num := nRollCountLastWeek;
    TPWrite "Roll Count In last 7 days...." \Num := nRollCountWeekTotal;
    TPWrite "Roll Count In This Hour......" \Num := nRollCountThisHour;
    TPWrite "Roll Count In Last Hour......" \Num := nRollCountlastHour;
ELSEIF ninfoscreen=2 THEN
    TPErase;
    TPWrite "Info Screen 3/4";
    TPWrite "Average Count Per Day:......." \Num := nAVGRollCountDay;
    TPWrite "Average Count Per Hour:......" \Num := nAVGRollCounthour;
    TPWrite "Best Count Per Hour:  ......." \Num := nBestRollCountHour;
    TPWrite "Best Count Per Day:..........." \Num := nBestRollCountDay;
    TPWrite "Best Count Per Week:........." \Num := nBestRollCountWeek;

ELSEIF ninfoscreen=3 THEN
    TPErase;
    TPWrite"Info Screen 4/4";
    TPWrite "Best Roll Count Day was ["+ strBestRollCountDay+"]";
    TPWrite ValToStr(nBestRollCountday)+" Rolls";
    TPWrite".............................................";
    TPWrite "Best Roll Count Hour was ["+ strBestRollCountHourClocktime+"]";
    TPWrite ValToStr(nBestRollCountHour)+" Rolls";
    TPWrite".............................................";
    TPWrite "Best Roll Cycletime:........." \Num := nBestCycleTime;
```

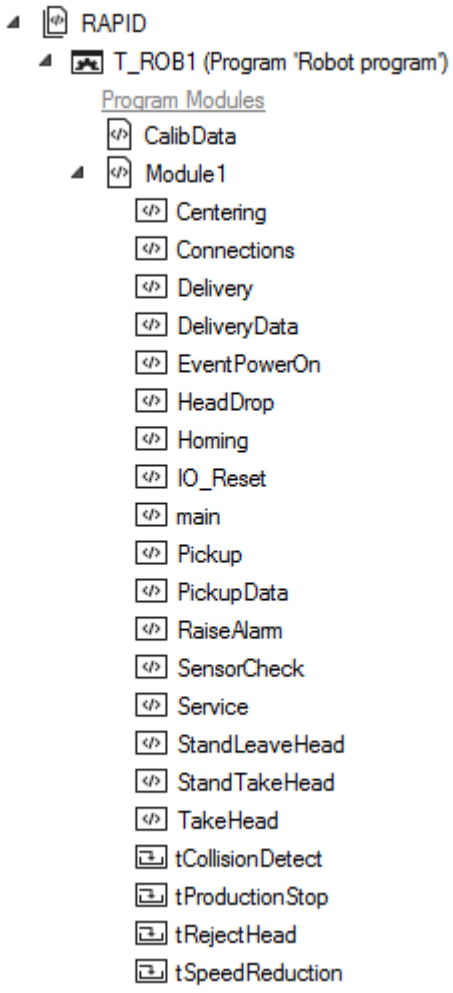
Kuvio 13. Robotin laskurit.

ABB-robottien ohjelmointi tapahtuu RAPID-ohjelmointi kielellä (ABB, i.a.-c). RAPID on ABB:n kehittämä robottien ohjelmointikieli, joka on käytössä kaikissa ABB:n roboteissa. Robotin ohjelmointia voi suorittaa joko RobotStudiossa tai suoraan ABB:n ohjauskapulasta Flexpendantilta.

Tehokas ohjelmointi edellytti, että jokainen robotin tehtävä jaettiin omiksi aliohjelmiksi eli alirutiineiksi. Tämän jälkeen ohjelmat ketjutettiin toteutusvaiheen mukaan. Main-ohjelma suorittaa aliohjelmien ketjua PLC:ltä tulevien signaalien perusteella. Se on perusta, jolle robotin ohjelmointilogiikka rakentuu, ja se koordinoi robotin toimintaa yhdessä muiden

järjestelmän osien kanssa. Main-ohjelma on hyvä pitää mahdollisimman yksinkertaisena, helposti luettavana. Tässä auttaa ohjelman työvaiheiden jaottelu omiksi aliohjelmiksi, joita sitten kutsutaan Main-ohjelmassa.

Ohjelma toteutettiin Main-ohjelmalla ja 17 eri aliohjelmalla, jotka on nimetty työvaiheiden mukaan (kuvio 14). Osaa aliohjelmista kutsutaan toisten aliohjelmien sisällä.



Kuvio 14. Robotin ohjelmamoduulit.

Seuraavassa käydään läpi työssä käytetyt aliohjelmat sekä niiden tarkoitus:

1. **Connections.** Connections-rutiinia kutsutaan main ohjelman alussa. Tässä rutiinissa määritellään I/O-signaalit työssä käytettyihin trap-funkitioihin.

2. **Centering.** Centering-rutiinissa suoritetaan robotin työvaihe keskitysasemalla. Työvaiheet on jaettu vielä kahteen pienempään aliohjelmaan: StandLeaveHead ja StandTakeHead, joita kutsutaan Centering-rutiinissa.

3. **Delivery.** Delivery-rutiinissa suoritetaan siirto-ohjelma keskitysasemalta rullalle päätylapun pudotukseen asti. Tässä rutiinissa kutsutaan myös aliohjelmia DeliveryData ja HeadDrop. DeliveryDatassa sijaitsevat laskentakaavat päätylapun viemiseksi oikeaan paikkaan keskelle rullan toista päätyä. HeadDrop-rutiinissa suoritetaan lapun tiputus ohjelmakoodi, sekä lapun pitäminen rullassa käärinnän aikana.

4. **EventPowerOn**. Tässä rutiinissa on määritelty World Zone-alueiden koordinaatiopisteet. Näiden avulla voidaan ohjelmaan määritellä robotille toiminta-alueita, sekä turvallisia poistumisreittejä, esimerkiksi virhetilanteessa törmäämättä laitteisiin.

5. **Homing**. Robotin kotiinajorutiini. Rutiinissa ohjataan robotti turvallisesti takaisin kotiasemaan. Tässä rutiinissa on hyödynnetty World Zone-alueita, että kotiinajo tapahtuisi turvallisesti. Kotiinajo on mahdollista suorittaa myös HMI-paneelille luodusta painikkeesta.

6. **IO-Reset.** Rutiinia kutsutaan Main-ohjelman alussa, ja ohjelma nollaa robotin tarttujan lähdöt sekä tiettyjä logiikalle meneviä lähtöjä (kuvio 15).

```
PROC IO_Reset()
Reset dor1_SuctionArea1;
Reset dor2_HolderReverse;
Reset dor3_HolderForward;

Reset do16_HeadDelivered;
Reset do19_PileLevelUpdated;
Reset do20_ProductionStopOn;
Reset do21_ReadyForNewValues;
Reset do22_ReadyToDeliver;
Reset do18_PermToMoveRevolver;
Reset do23_ResetAlarmAck;
Reset do17_PermissionToRoll;

bBGDataReady:=FALSE;

ENDPROC
```

Kuvio 15. IO-Reset-rutiinin suorittama ohjelmakoodi.

7. **PickUp.** Lapun nouto päätylappukarusellista. Ohjelmassa kutsutaan myös PickUpData- ja TakeHead-rutiinit. Kun karuselli on kääntänyt oikean pinon robotin käytettäväksi, suoritetaan PickUp-rutiini, ohjelmassa robotti lähestyy pinoa, kunnes StackDetection-input aktivoituu. Robotti pudottaa vauhtia tSpeedReduction trap-funktion avulla. Kun imukupit ovat painuneet tarpeeksi, induktiiviset anturit vaikuttuvat ja liike pysähtyy, imu lähtee päälle ja lappu nostetaan karusellista pois. PickUpData-rutiinissa suoritetaan laskenta keskikohdan määrittämiselle ja sitä kutsutaan PickUp-rutiinin alussa.

8. **RaiseAlarm.** Hälytyksien valvontaan käytettävä rutiini. Ohjelman suorittaminen pysähtyy hälytyksen aktivoituessa.

9. **SensorCheck.** Suoritetaan Main-ohjelman alussa. Tarkistaa tarttujan anturien tilat ja antaa hälytyksen, jos anturitilat eivät ole kunnossa.

10. **Service.** Paneelilta ja FlexPendantilta voidaan ajaa robotti service-asentoon eli huoltoasemaan. Robotti ajaa tarttujan paikkaan, jossa tarttuja on helppo huoltaa tarvittaessa.

6.5 Trapit

Ohjelma sisältää myös trap-funktioita, joiden avulla saadaan ohjelmaan vaikutettua kesken ohjelmakierron. Trap-funktioiden toteutus on määritelty kuvion 16 mukaisesti Connections-rutiinissa, trap-funktioita voidaan asettaa ohjelmassa joko IWatch-komennolla tarkkailuun tai komennolla ISleep, joka poistaa tarkkailusta, tällöin trap ei vaikuta ohjelman kulkuun. Trapit sidotaan keskeytyksiin intnum-muuttujien avulla.

```
PROC Connections()
IDelete intProductionStop;
CONNECT intProductionStop WITH tProductionStop;
ISignalDI di15_ProductionStop,1,intProductionStop;
ISleep intProductionStop;

IDelete intSpeedReduction;
CONNECT intSpeedReduction WITH tSpeedReduction;
ISignalDI dir4_StackDetection,0,intSpeedReduction;
ISleep intSpeedReduction;

IDelete intCollisionDetect;
CONNECT intCollisionDetect WITH tcollisionDetect;
ISignalDI dis_ColDetect,1,intCollisionDetect;
ISleep intCollisionDetect;

IDelete intRejectHead;
CONNECT intRejectHead WITH tRejectHead;
ISignalDI di17_RejectHead,1,intRejectHead;
ISleep intRejectHead;

IDelete intStandSensor;
CONNECT intStandSensor WITH tStandsensor;
ISignalDI dir16_HeadAtStand,1,intStandSensor;
ISleep intStandSensor;
```

Kuvio 16. Connections-rutiinin ohjelmakoodi.

Seuraavassa käydään läpi työssä käytetyt trapit:

tProductionStop. Tämä trap valvoo PLC.ltä tulevaa input-tietoa ”di_ProductionStop”. Kun tämä tulo menee päälle, aktivoituu tCollisionDetect-trap. Tämä trap keskeyttää ohjelman ja suorittaa kuvion 17 mukaiset komennot.

```
TRAP tProductionStop
    IF do5_SysAutoON=0 RETURN;
    Set do20_ProductionStopOn;
    stop;
    Reset do20_ProductionStopOn;
ENDTRAP
```

Kuvio 17. ProductionStop-trap.

ProductionStop on tuotannosta tuleva käsky tuotannon keskeyttämisestä. Kaikissa ohjelman vaiheissa tätä trappia ei valvota. Esimerkiksi, jos robotti on suorittamassa kriittistä työvaihetta, kuten lapun rullalle vientiä, tällöin trap poistetaan valvonnasta ISleep-komennolla ja robotti suorittaa tehtävän loppuun ja pysähtyy.

tSpeedReduction. Tämä trap on tarkoitettu robotin vauhdin hiljentämiselle lähestyttäessä pinoa karusellissa. Se aktivoituu kun tarttujan anturilta tuleva tulo dir4_StackDetection menee päälle (kuvio 18).

```
TRAP tSpeedReduction            !Trap for speed reduction on pile approaching
    ISleep intCollisionDetect;
    nSpeed_Correction := 30;
    SpeedRefresh nSpeed_Correction;
ENDTRAP
```

Kuvio 18. Speed reduction trap.

tCollisionDetect. Tämä trap valvoo tarttujan Collision Detect-antureilta tulevaa input-tietoa. Tämä trap aktivoituu, mikäli robotti on lähellä törmätä tai pino on väärässä paikassa (kuvio 19).

```
    TRAP tCollisionDetect               !Trap for collision detect while seaching pile
        RaiseAlarm 4;   !Collision Detect
        TPWrite "Either the robot was close to the collision or";
        TPwrite "the collision sensor hit the edge of the pile!!";
        Stopmove;
        ClearPath;
        WaitTime\inpos, 1;
        StartMove;
        ExitCycle;
        Stop;
ENDTRAP
```

Kuvio 19. Collision detect-trap.

tRejectHead. Operaattori voi tarvittaessa hylätä lapun. Tällöin operaattori painaa HMI-paneelilla olevaa painiketta ja robotti vie lapun kotiaseman kautta hylkäyspisteelle, ja aloittaa tehtävän alusta (kuvio 20).

```
TRAP tRejectHead
    TPWrite "Head rejection started";
    Homing;
    ExitCycle;
ENDTRAP
```

Kuvio 20. Reject head-trap.

tStandSensor. Tämä trap valvoo keskitysaseman anturia dir16_HeadAtStand. Mikäli anturi on epänormaalissa kohdassa ohjelmaa vaikuttuneena antaa ohjelma hälytyksen ”Head stuck in stand or sensor not working”. Tämän avulla operaattori tietää tarkistaa anturin ja poistaa keskitysasemalla mahdollisesti olevan päätylapun ennen kuin robotti tuo uuden (kuvio 21).

```
TRAP tStandSensor
    RaiseAlarm 24;   !Head stuck in stand or sensor not working
    TPWrite "Head stuck in stand or sensor not working";
ENDTRAP
```

Kuvio 21. Stand sensor-trap.

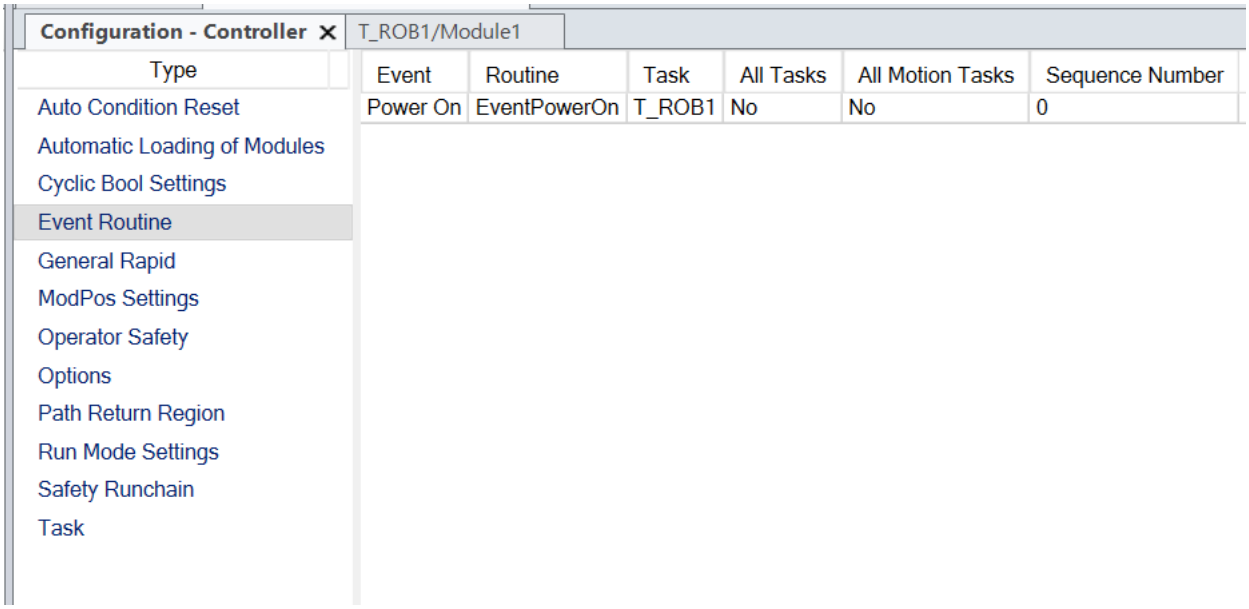
6.6 World Zones

Robotin ohjelmaan määriteltiin yhteensä kuvion 22 mukaisesti 11 world zonea. Näiden avulla saadaan robotti tarvittaessa palaamaan turvallisesti kotiasentoon lähes mistä tahansa, näiden avulla voidaan myös estää kuljettimien ja karusellin käyttö PLC:lla, kun robotti on näillä alueilla. World Zone määrittely tapahtui EventPowerOn-rutiiniin, jossa määriteltiin alueen tyyppi, alueen raja-arvot, sekä toiminnallisuus robotin ollessa alueella.

```
PROC EventPowerOn()
    WZHomeJointDef\Inside,shdHomePos,jHomePos,delta_pos;
    WZDOSet\Stat,wzsHomeArea\Inside,shdHomepos,do_wzRobotAtHome,1;

    WZHomeJointDef\Inside,shdServicePos,jServicePos,delta_pos;
    WZDOSet\Stat,wzsServiceArea\Inside,shdServicepos,do_wzServiceArea,1;

    wzcyldef\Inside,shdRevolver,[4193,580,0],2286,3050;
    WZDOSet\Stat,wzsRevolverArea\Inside,shdRevolver,do_wzRevolverArea,1;

    wzcyldef\Inside,shdPileArea,[2720,583,700],950,2300;
    WZDOSet\Stat,wzsPileArea\Inside,shdPileArea,do_wzPileArea,1;

    WZBoxDef\Inside,shdRollArea,[-4387,-4707,0],[3614,-2607,3500];
    WZDOSet\Stat,wzsRollArea\Inside,shdRollArea,do_wzRollArea,1;

    WZBoxDef\Inside,shdWrapArea,[-1769,-2607,0],[1731,-1107,1720];
    WZDOSet\Stat,wzsWrapArea\Inside,shdWrapArea,do_wzWrapArea,1;

    WZBoxDef\Inside,shdAboveStandArea,[1577,-2729,2100],[2777,-1079,3500];
    WZDOSet\Stat,wzsAboveStandArea\Inside,shdAboveStandArea,do_wzAboveStandArea,1;

    WZBoxDef\Inside,shdCenteringArea,[2088,-1174,798],[2782,-1824,1841];
    WZDOSet\Stat,wzsCenteringArea\Inside,shdCenteringArea,do_wzCenteringArea,1;

    WZBoxDef\Inside,shdHeadDropArea,[-1050,-4424,900],[0,-2849,2470];
    WZDOSet\Stat,wzsHeadDropArea\Inside,shdHeadDropArea,do_wzHeadDropArea,1;

    WZBoxDef\Inside,shdAboveRollArea,[-1923,-3629,2600],[1577,-1029,3500];
    WZDOSet\Stat,wzsAboveRollArea\Inside,shdAboveRollArea,do_wzAboveRoll,1;

    WZBoxDef\Inside,shdAboveRevolver,[1577,-1079,3052],[2777,2221,3502];
    WZDOSet\Stat,wzsAboveRevolver\Inside,shdAboveRevolver,do_wzAboveRevolverArea,1;
```

Kuvio 22. World Zone määrittely ohjelmassa.

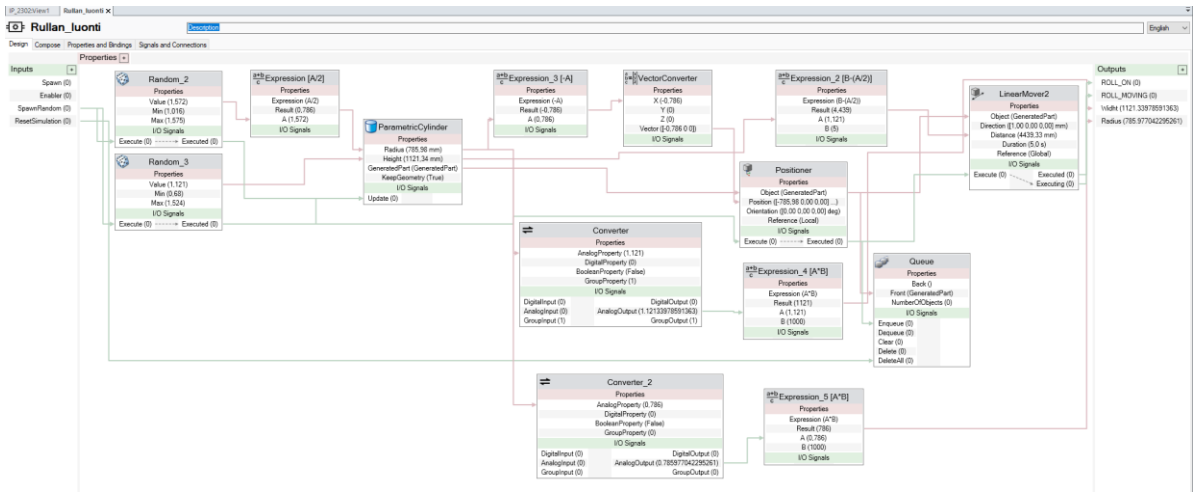
EventPowerOn-rutiini määriteltiin Event-rutiiniksi (kuvio 23), joka suoritetaan, kun robotti käynnistetään. Tämän määrittelyn avulla varmistetaan robotin sijainti heti robotin käynnistyksessä.



Kuvio 23. Event-rutiinin määrittely.

6.7 Simulointi

Rullalle viennin simulointia varten luotiin kuvion 24 mukainen Smart Component. Smart Componentin tarkoituksena oli luoda rulla käärintäaseman keskelle satunnaisilla mitoilla. Tällä voitiin simuloida laskentojen toimivuus päätylapun toimituksessa rullanpäätyyn.



Kuvio 24. Rullan luonti Smart Component.

Smart Component luo sallituissa rajoissa rullalle satunnaisen halkaisijan ja pituuden. Tällä voitiin todentaa laskentojen toimivuus ja se, että päätylappu toimitetaan keskelle rullan päätyyn. Simuloinnista oli hyötyä, sillä aikaisempi suunnitelma ei toiminut oikealla tavalla ja sitä täytyi korjata. Uusien laskentojen lopputuloksena saatiin toimiva ohjelma ja päätylappu sijoittuu simuloinnin perusteella oikein.

7 Yhteenveto ja pohdinta

Opinnäytetyön tavoitteena oli automatisoida robottisolu. Työn alussa tutustuttiin robottisuunnittelun teoriaan ja työvaiheisiin, jonka kautta edettiin robottisolun suunnitteluvaiheeseen ja toteutukseen.

Simulointien ja yrityksessä käytyjen läpikäyntien perusteella saatiin käyttökelpoinen ohjelma robotille. Tämä osuus asiakasprojektista onnistui suunnitellulla tavalla tavoitteiden mukaisessa aikataulussa. Robotin fyysinen testaaminen ei projektin aikataulujen vuoksi ehtinyt tähän työhön mukaan. Tehdastestien tarkoituksena on testata linjaston toimivuus ennen lopullista käyntiinajoa asiakkaan tehtaassa. Tällöin testataan myös robotin ohjelman toiminta kokonaisuudessaan. Tehty ohjelma toimii kuitenkin hyvänä pohjana varsinaisille tehdastesteille, vaikkakin pieniä ja isompiakin korjauksia mahdollisesti tarvitsee tehdä.

Haasteita riitti erityisesti työn laajuuden takia ohjelman rakenteen muodostamisessa ja kokonaisuuden hallinnassa. Tutkimustyötä täytyi tehdä koko suunnitteluprosessin ajan.

Tehdyn työn tuloksia voidaan ainakin joiltain osin hyödyntää jatkossakin robottisuunnittelussa tulevissa asiakasprojekteissa. Projekti oli mielenkiintoinen ja sopivan laajuinen sekä haastava, ja antoi hyvän kuvan robottisuunnittelijan työstä.

LÄHTEET

ABB. (i.a.-a). *Installation and Commissioning - Service.* https://new.abb.com/products/robotics/robotstudio/installation-and-commissioning-service

ABB. (i.a.-b). *IRB 6700*. https://new.abb.com/products/robotics/robots/articulated-robots/irb-6700

ABB. (i.a.-c). *Technical reference manual*. https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf

ABB. (2004). *Operating manual – IRC5 with FlexPendant*. https://icdn.tradew.com/file/201606/1569362/pdf/7083424.pdf

ABB. (2024). *RobotStudio*. https://new.abb.com/products/robotics/robotstudio

Ahonen, T., Latokartano, J., Liuha, A., Lempiäinen, J., Billing, M., O'Connor, J., & Maylett, R. (2023). *The industrial robot book. The Robotics Society in Finland*.

Baiju, N.T. (2021). *Factors to consider while selecting appropriate industrial robots*. https://roboticsbiz.com/factors-to-consider-while-selecting-appropriate-industrial-robots/

Bolton, W. (2019). *Programmable Logic Controllers*. Newnes.

Built In. (2023). *Native vs. Cross-Platform App Development*. https://builtin.com/software-engineering-perspectives/native-vs-cross-platform-app-development

Control Automation. (2022). *ABB robot example programming tutorial*. https://control.com/technical-articles/abb-robot-example-programming-tutorial/

Core Link AB. (i.a.). *Sales offices*. https://www.corelink.se/sales-offices/

Dorna Robotics. (i.a.). *Robot grippers*. https://dorna.ai/grippers/

Encoder.com. (2023). *White Paper - Industrial Ethernet Communication Protocols*. https://www.encoder.com/wp2019-industrial-ethernet-communication-protocols

FS Community. (2023). *Comparing Industrial Ethernet Protocols*. https://community.fs.com/article/comparing-industrial-ethernet-protocolschoosing-the-right-solution-for-your-network.html

Groover, P. (2020). *Fundamentals of Modern Manufacturing: Materials, Processes, and Systems*. John Wiley & Sons.

Hughes, E. (2020). *Electrical and Electronic Technology. Pearson Education*.

Makhal, A., & Goins, A. (2017). *Reuleaux: Robot Base Placement by Reachability Analysis*. https://arxiv.org/pdf/1710.01328.pdf

Raygun Blog. (2023). *The 2023 guide to native app development*. https://raygun.com/blog/native-app-development-guide/

Robots Done Right. (i.a.). *Considerations for selecting the right industrial robot*. https://robotsdoneright.com/Articles/considerations-for-selecting-the-right-industrial-robot.html

Rockwell Automation. (i.a.). *Studio 5000 Logix Designer*. https://www.rockwellautomation.com/en-us/products/software/factorytalk/designsuite/studio-5000/studio-5000-logix-designer.html

Shimon, Y. (2020). *Handbook of Industrial Robotics*. John Wiley & Sons.

Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics: Modelling, Planning and Control*. Springer.

Universal Robots. (i.a.). *End of arm tooling (EOAT)*. https://www.plus.universal-robots.com/blog/tags/end-of-arm-tooling/

Uptech. (2023). *Native vs Cross-Platform Development*. https://www.uptech.team/blog/native-vs-cross-platform-development