Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Kasimir Luotojärvi

Uuden robottisolun ohjelmointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatiotekniikka

Insinöörityö

04.05.2020

Tekijä Otsikko Sivumäärä Aika	Kasimir Luotojärvi Uuden robottisolun ohjelmointi 25 sivua + 7 liitettä 04.05.2020
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	sähkö- ja automaatiotekniikka
Ammatillinen pääaine	automaatiotekniikka
Ohjaajat	lehtori Timo Tuominen sr. process development engineer Juha-Pekka Larramäki

Insinöörityö tehtiin Murata Electronicsille. Tarkoituksena oli suunnitella robottisolu uuniprosessille, jossa keskeneräisiä antureita sisältävät makasiinit kulkevat uunin läpi.

Ensin robottisolusta ja nykyisestä prosessista tehtiin 3D-mallinnukset, joissa niiden toimintaa simuloitiin ja niiden teoreettista suorituskykyä vertailtiin toisiinsa. Seuraavaksi varastossa olevaan nivelvarsirobottiin hankittiin robottisolun edellyttämät komponentit. Tämän jälkeen komponenttien välille muodostettiin sähköiset yhteydet ja lopuksi robotille luotiin solun mukainen ohjelma.

Robottisolun toimintaedellytyksenä oli, että robottia toimii ohjelmoitavan logiikan kautta. Logiikan tehtävänä on antaa robotille numeroituja käskyjä, jotka robotti kuittaa logiikalle takaisin. Robotin tarttujaksi kaavailtiin ensin sähkötoimista tarttujaa, mutta aikaisempien kokemusten perusteella se päätettiin vaihtaa pneumaattistoimiseksi. Tätä varten logiikkaan tuli liittää paineilma-mittari, jonka tarkoituksena on estää robottia toimimasta, jos paine on liian alhainen. Logiikkaan liitettiin myös viivakoodinlukija, joka lukee makasiinin kyljessä olevan viivakoodin.

Insinöörityön suorittamiseen liittyi muutamia ongelmia, kuten hitaudet komponenttien toimituksissa ja logiikan ohjelmointityökalun lisenssin väliaikainen vanhentuminen. Ongelmista huolimatta kaikki komponentit saatiin hankittua ja insinöörityön lopputuloksena luotiin robotille toimiva ohjelma. Kun robotti päätetään ottaa tuotantokäyttöön, komponentteja ja ohjelmaa on tarkoitus hyödyntää siihen liittyvässä käyttöönottoprojektissa.

Avainsanat	robotiikka, 3D-mallinnus, logiikka, konfiguraatio, ohjelmointi

Author Title Number of Pages Date	Kasimir Luotojärvi Programming the logic for a new robot cell 25 pages + 7 appendices 4 May 2020
Degree	Bachelor of Engineering
Degree Programme	Electrical and Automation Engineering
Professional Major	Automation Engineering
Instructors	Timo Tuominen, Senior Lecturer Juha-Pekka Larramäki, Sr. Process Development Engineer

This final year project was made for Murata Electronics. The purpose of this project was to design a robot cell for oven process where magazines containing unfinished sensors are passing through the oven.

At first 3D simulation models were created from robot cell and the current process. After that, graphs presenting their theoretical efficiency were created from those models and were compared. Next step was to acquire parts for the articulated arm robot, located in warehouse, so it functions according to the designed cell. After this step, electronic connections between components were established. Finally, robot was programmed according to the robot cell.

A precondition of the robot cell was that it must act via logic. The purpose of the logic was to give numeric commands to robot from which robot acknowledges them back to the logic. An electric gripper was first planned for the gripping tool of the robot, but based on previous experience it was decided to replace it with a pneumatic gripper. For this kind of gripper also air gauge had to be connected to the logic, because it prevents robot from acting if the pressure is too low. A barcode reader that reads the barcode on the side of magazine was also attached to the logic.

There were few problems with the project, such as delays in component deliveries and temporary expiration of the license for the logic-programming tool. Despite the problems, all components were procured and as the result of the project, working program was created for the robot. When the robot is decided to be put into production, components and the program are to be utilized in the deployment project.

Keywords	Robotics, 3D-modeling, Logic, Configuration, Programming

Sisällys

Lyhenteet ja käsitteet

GSD	*General Station Description*. Ohjelmointityökaluun ladattava laitteen valmistajan kuvaus I/O-laitteesta.
GUI	*Graphical User Interface*. Graafinen käyttöliittymä.
GVL	*Global Variable List*. Globaalien muuttujien lista.
HMI	*Human Machine Interface*. Graafinen käyttöliittymä.
Layout	Esimerkiksi teollisen prosessin pohjapiirroksesta luotu tilallinen hahmotelma.
MEMS	*Micro Electrical Mechanical System*. Komponentti, jossa mekaaniset muutokset muutetaan sähköisiksi signaaleiksi.
PLC	*Programmable Logic Computer*. Ohjelmoitava logiikka.
Profinet	Avoin teollisuus-Ethernet standardi, jota käytetään esimerkiksi automaatiolaitteiden väliseen kommunikointiin.
TIA-Portal	*Totally Integrated Automation-Portal*. Siemensin valmistamien automaatiolaitteiden ohjelmointityökalu.
Toistotarkkuus	Tilastollinen tarkkuus, jolla robotti palaa sille ohjelmoituun pisteeseen.

1 Johdanto

Insinöörityö tehdään Murata Electronics Oy:lle ja sen tarkoituksena on kehittää anturien Reflow-uuniprosessointivaiheeseen automaatioratkaisu. Työn tavoitteena on suunnitella robottisolu, joka toimii ohjelmoitavan logiikan välityksellä. Suunnitteluvaiheessa mallinnettaan nykyinen ja uusi layout simulaatio-ohjelmistolla ja mallinnuksien ideaalista suorituskykyä vertaillaan toisiinsa. Suunnitteluvaihe sisältää myös robottiin tarvittavien mahdollisten lisäosien suunnittelun. Kun suunnitteluvaihe on valmis, robottiin hankitaan tarvittavat osat. Tämän jälkeen robotti kasataan testipaikassa ja ohjelmoidaan valmiiksi siellä. Insinöörityön tuloksena syntynyttä robotin logiikkaohjelmaa on tarkoitus hyödyntää siinä vaiheessa, kun robotti päätetään ottaa tuotantokäyttöön.

Normaalisti uutta robottisolua suunniteltaessa tulisi sitä varten laatia myös turvallisuusselvitys, jossa määriteltäisiin robotin ympärille rakennettavat turvallisuusmekanismit työtapaturmien välttämiseksi. Opinnäytetyö ei sisällä turvallisuusselvitystä, koska robottia ei olla vielä ottamassa tuotantokäyttöön. Päätavoitteena on suunnitella vain robotin toiminta logiikka, mutta sitä suunniteltaessa otetaan myös mahdolliset turvallisuusnäkökohdat huomioon.

Anturikotelot kulkevat uunin läpi (kuva 1), jonka tarkoitus on ”vanhentaa” antureita. Vanhennus paljastaa niissä mahdollisesti piileviä vikoja, kuten esimerkiksi halkeamia ja ilmakuplia. Anturit ovat sijoitettu makasiineihin, ja ne ovat asetettu uunijigeihin. Uunijigit ovat metallikehyksiä, jotka toimivat kuljetusalustana makasiineille uunin liukuhihnalla. Operaattori asettaa makasiinin sisältämän uunijigin liukuhihnan alkupäähän kaksi kerrallaan, ja ne kulkevat uunin läpi noin kahdeksassa minuutissa. Kun molemmat ovat kulkeneet uunin läpi, operaattori ottaa makasiinin liukuhihnalta lämpöeristävillä hanskoilla ja asettaa sen jäähytelineeseen noin 15 minuutiksi. Uunijigit puolestaan asetetaan jäähtymään pöydälle. Kun makasiini on jäähtynyt, operaattori laittaa makasiinit tuotekohtaisesti kuljetuslaatikkoon. Siihen mahtuu kahdeksan makasiinia kerrallaan. Laatikon täytyttyä se viedään sen läpivientikaappiin edelleen prosessointia varten.

Kuva 1. Reflow-uuni.

Murata Electronics Oy

Murata Electronics perustettiin vuonna 1944 Kiotossa Japanissa Akira Muratan toimesta. Aluksi yritys valmisti vain keraamisia eristeitä, mutta myöhemmin yritys erikoistui elektroniikassa tarvittaviin komponentteihin. Ensimmäisiä tällaisia komponentteja olivat esimerkiksi kondensaattorit, joiden kysyntä kasvoi mm. radioiden yleistyttyä. Ajan kuluessa yritys erikoistui yhä enemmän erilaisiin elektronisiin komponentteihin ja nykypäivään mennessä Murata Electronicsin päätuotteita ovat elektroniset anturit. (1.)

Suomeen Murata saapui vuonna 2012, kun konserni päätti ostaa Vantaalla sijaitsevan VTI-technologies Oy: n. Toimipiste valmistaa antureita autoteollisuuden ja terveydenhuoltoalan tarpeisiin. Nämä ovat ns. MEMS-antureita, jotka mittaavat erilaisia fysikaalisia suureita kuten kiihtyvyyttä, painetta, kallistusta ja kulmanopeutta. Tällä hetkellä Murata työllistää Suomessa on noin 1000 henkilöä, mutta määrä kasvaa koko ajan uuden tehdaslaajennuksen myötä.

Antureiden suurin tilaaja on autoteollisuus, joka tarvitsee kiihtyvyys- ja kallistusantureita autojen sähköisiin järjestelmiin (esimerkiksi ESP ja ABS). Toinen tilaaja on terveydenhuoltoala, joka tarvitsee antureita mm. sydämentahdistimiin. Muita antureiden tilaajia ovat esimerkiksi maatalouskonevalmistajat, kaivosteollisuus ja ilmailuala. (2.)

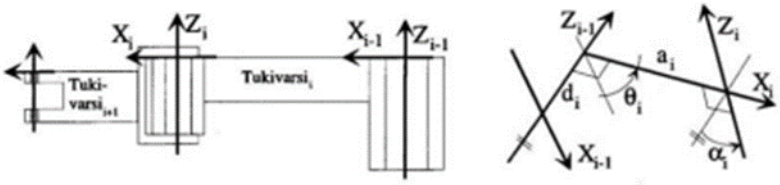
2 Nivelvarsirobotit, robottitarttujat ja ohjelmoitavat logiikat

2.1 Nivelvarsirobottien yleinen kehitys

Sana robotti tulee tšekin kielestä robota, joka viittaa mm. maaorjien palkatta tekemiin raskaisiin töihin. Sanan otti ensimmäisen kerran käyttöön näytelmäkirjailija Karel Čapek vuonna 1920 ilmestyneessä näytelmässään Rossum`s Universal Robots. (3; 4, s.2; 6; 7 s.11.) Robotti sanana vakiintui myöhemmin tarkoittamaan suurinta osaa ihmisien tieteellisissä tarkoituksissa luomia itsenäisesti toimivia mekaanisia laitteistoja, jotka toimivat ympäristön ärsykkeiden mukaisesti (4, s.2).

Robotiikan kehitys otti askeleen eteenpäin 1950-luvulla, kun Jacques Denavit ja Richard S. Hartenberg määrittelivät tavan laskea kinemaattisia liikesarjoja matriisien avulla (4, s.5). Näitä kutsutaan yleisesti Denavit-Hartenberg-parametreiksi (kuva 2). Niissä parametrit ovat määritelty seuraavasti (5, s.24):

- αi on kahden peräkkäisten akselien välinen etäisyys
- αi on kahden perättäisen akselien välinen kulma
- θi on tukivarren kiertymäkulma
- di on nivelen akselin suuntainen siirtymä.



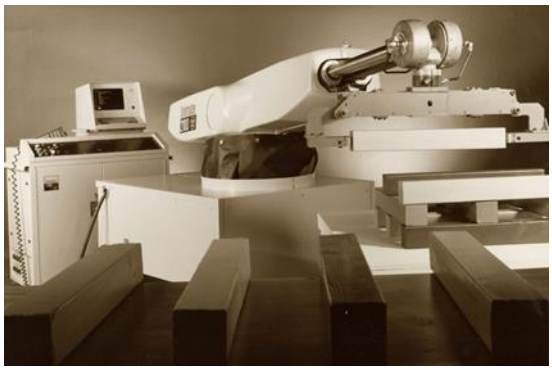
Kuva 2. Robotin tukivarsistoihin lisätyt koordinaatit ja niiden perusteella nimetyt parametrit Denavit-Hartenberg-menetelmällä (5 s.24).

$$T = \begin{bmatrix} cos\theta & -sin\theta cos\alpha & sin\theta sin\alpha & acos\alpha \\ sin\theta & cos\theta cos\alpha & -cos\theta sin\alpha & asin\alpha \\ 0 & sin\alpha & cos\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parametrien avulla voidaan määrittää yläpuolella oleva homogeeninen muunnosmatriisi, jonka avulla tietyn pisteen koordinaatit voidaan määrittää jossain toisessa koordinaatistossa. Esimerkiksi robotin työkalun koordinaattipisteet voidaan matriisin avulla viedä maailmankoordinaatistoon. Matriisi saattaa yksinkertaistua, koska esimerkiksi α-kulma on usein 0 tai ±90°. Tämä on kuitenkin robotin rakenteesta riippuvainen. (5, s.24, 25.) Parametrit mahdollistivat käyttämään robottikäsivartta, joka koostuu kahdesta tai useammasta mekaanisesta komponentista. Nämä komponentit ovat yhdistetty käyttäen joko liuku- tai kiertoniveliä. (4, s.5.)

Robotiikassa kinematiikka voidaan jakaa pääosin kahteen eri kategoriaan: suoraan ja käänteiseen kinematiikkaan. Suoran kinematiikan periaatteessa jonkin pisteen sijainti lasketaan käyttäen hyödyksi muiden komponenttien nivelten kulmia. Epäsuorassa kinematiikassa puolestaan pisteen sijainti lasketaan robotin työkalun aseman ja asennon perusteella ja otetaan huomioon muiden nivelten vapausasteet. (4, s.5.)

Transistorit nopeuttivat 1960-luvulla robotiikan kehitystä, koska ne mahdollistivat tietokoneiden yhdistämisen robotteihin. Tietokoneet alkoivat hoitaa tietojenkäsittelyä automaattisesti. Ensimmäinen sähkötoiminen teollinen robotti oli Unimate (kuva 3), jonka General Motors otti käyttöön autotehtaassa vuonna 1961 painevalukoneen käyttöön. Robotin toiminta perustui rumpumuistiin, johon oli tallennettu noin 180 liikesarjaa. (4, s.5.)



Kuva 3. Unimate-robotti (6).

Teollisuusrobotit yleistyivät kaupallisesti vasta 1980-luvulla (7, s.11). Robotit alkoivat lisääntyvissä määrin korvata ihmistä raskaissa teollisissa töissä kuten tavaroiden lavauksissa ja pakkaamisissa. Niiden käyttökohteet laajenivat myös tarkkuutta vaativiin töihin

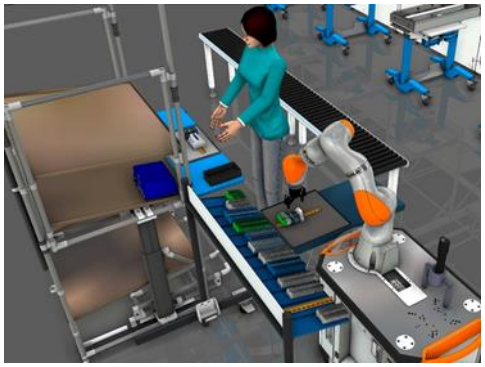
kuten hitsaamiseen ja maalaamiseen. Näiden edellä mainittujen työtehtävien automatisointi lisäsi tehtaiden tuotantokapasiteettia merkittävästi. Vuosikymmenen lopulla teollisuusrobotteihin oli myös yleistynyt nivelten ohjaus servomoottoreilla, joka mahdollisti niiden entistä tarkemman ohjauksen ja toistotarkkuuden (8). 1990-luvun alkuun mennessä robottien perustekniikka vakiintui ja on kehittynyt siitä lähtien laaja-alaisesti asteittain nykypäivään (8; 9).

Teollisuusrobotit ovat yleensä turvallisuussyistä eristetty omaan tilaansa, jotta ihmiset eivät pääsisi niiden läheisyyteen mahdollisten työtapaturmien vuoksi. Näitä työtapaturman riskejä ovat esimerkiksi puristumis- ja törmäysvaarat. Kuitenkin yksi 2020-luvun kehityksen kohteista on ollut yhteistyö ihmisten kanssa (kuva 4). Näitä niin sanottuja yhteistyörobotteja on tarkoitus käyttää tuotannon prosesseissa yhdistämällä ihmisen joustavuutta ja robotin tarkkuutta ja tarvittavaa voimaa. (9.)

Robotin toimintaa voidaan myös simuloida ja ohjelmoida etukäteen erilaisilla tietokoneohjelmoistoilla. Tätä kutsutaan etäohjelmoinniksi. Sen hyötyjä ovat esimerkiksi:

- tuotantoseisokkien häviäminen
- ohjelmoinnin ja suunnittelun nopeutuminen
- erilaisten layout-ratkaisujen suunnittelu koko tuotannon parhaaksi.

Edellä mainittujen hyötyjen kautta etäohjelmoinnilla saavutetaan kustannussäästöjä ja sitä voidaan hyödyntää myös työntekijöiden perehdytyksessä tai mainostamistarkoituksissa. (10, s.21.)



Kuva 4. Yhteistyörobotin visuaalinen mallinnus simulaatio-ohjelmiston avulla (11).

2.2 Robottitarttujat

Robotin työkaluna käytetään useimmiten jotakin tarttujaa. Niitä ovat

- mekaaniset tarttujat
- alipaine/Imukuppitarttujat
- magneettiset tarttujat
- universaalit tarttujat
- erikoistarttujat.

Yleisin tarttuja on mekaaninen. Mekaaninen tarttuja saa käyttövoimansa joko sähköstä, hydrauliikasta tai paineilmasta. Niissä on yleensä kaksi leukaa, mutta niitä voi olla tarvittaessa enemmänkin. Paineilmatarttujat ovat halvimpia, kun taas puolestaan sähköiset tarttujat voivat olla moninkertaisesti kalliimpia. Leuat voivat liikkua joko rotaatio- tai lineaariliikkeillä. Tarttujan suunnittelussa otetaan huomioon mm. käsiteltävä kappale ja toimintaympäristö. Oleellisinta kuitenkin on, miten ja mistä suunnasta kappaletta halutaan käsitellä. (12, s.6.)

Alipainetarttujia käytetään silloin kun mekaanisen tarttujan käyttö todetaan hankalammaksi esimerkiksi kappaleen muotojen vuoksi. Näiden tarttujien toiminta perustuu alipaineistettuun imukuppiin, joita voi olla useampi paremman tartuntavoiman takaamiseksi. Tartuntapinnan tulee olla tasainen ja puhdas, jotta imukuppi pystyy tarttumaan kappaleeseen tiiviisti. (12, s.7.)

Magneettisille aineille voidaan käyttää magneettitarttujia, jotka tarttuvat kappaleeseen magneettikenttien avulla. Tartuntapinta-ala tulee olla mahdollisimman suuri ja mahdollisen ilmaraon tulee olla vastaavasti pieni, jotta ote olisi pitävä. Magneetti myös lämpenee käytettäessä, jolloin työnkiertoa tulee ajatella sen myös sen kannalta. Kappaleen irrotus tarttujasta tapahtuu kääntämällä magneettikentän suunta ja kääntönopeudella on suoravaikutus kappaleen irrotusaikaan. (12, s.8.)

Universaalien tarttujien kehityksen lähtökohta on ollut ihmiskäden toiminnan mallintaminen. Versioita on tehty vetopyörien ja vaijerien avulla liikuteltavista sormista aina ilmalla

täytettäviin kumipalloihin, jotka mukautuvat kappaleen muotoihin tyhjiön avulla. Universaalit tarttujat ovat kuitenkin kalliita verrattuna tavanomaisiin tarttujiin ja siksi niiden käyttö on harvinaisempaa. (12, s.9.)

Erikoistarttujat voivat olla mitä tahansa edellä mainittujen yhdistelmiä. Niiden rakenne riippuu käyttökohteesta tai ympäristön erityisvaatimuksista (12, s.10).

2.3 Ohjelmoitavat logiikat ja niiden historiaa

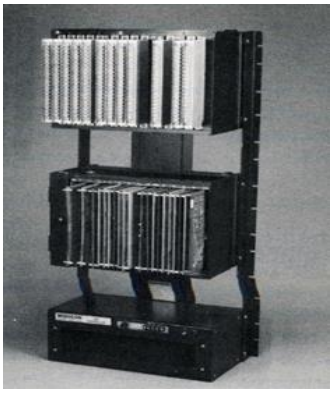
Ohjelmoitava logiikka lyhenteeltään PLC (Programmable Logic Computer) on mikroprosessorilla toimiva tietokone, joka on tarkoitettu ensisijaisesti teollisten automaatiojärjestelmien ohjaukseen. PLC:t korvaavat monimutkaiset ja tilaa vievät relepohjaiset koneohjausjärjestelmät, joiden muutostyöt voivat olla hankalia esimerkiksi uudelleen johdotusten takia. PLC:llä muutostyöt voidaan tehdä pelkästään sen uudelleen ohjelmoinnin kautta. (13, s.9.)

Logiikassa on tulo- ja lähtöportit, joita kutsutaan myös I/O:iksi (input/output). Tarvittaessa niiden määrää voidaan kasvattaa lisäämällä logiikkaan I/O-kortteja. Niihin on kytketty järjestelmässä olevat toimilaitteet kuten esimerkiksi anturit, venttiilit, sähkömoottorit ja valot. Anturit ja venttiilit ovat kytketty tulopuolelle, kun taas puolestaan sähkömoottorit ja valot kytketään lähtöpuolelle. Tulo- ja lähtöporttien käsittelemät signaalit voivat olla tyypiltään joko digitaalisia tai analogisia. (13, s.9, s.10.)

Digitaaliset signaalit toimivat tasajännitteellä (12 V ja 24 V) tai vaihtojännitteellä (110 V ja 240 V). Digitaalisignaalien tila voi olla vain joko 0 tai 1. Digitaalisilla signaaleilla kerrotaan vain toimilaitteentila, eli onko se aktiivinen vai ei. Logiikat pystyivät aluksi vain käsittelemään digitaalisia signaaleja. (13, s.9.)

Analogisten signaalien tila voi puolestaan olla mitä tahansa mittausalueen välistä. Ne käsitellään yleensä kokonaislukuina ja logiikkaan tarvittava muistimäärä kasvaa käsiteltäessä yhä tarkempia lukuja. Analogiviestit luetaan joko 4–20 mA tai 0–20 mA sähkövirrasta tai jännitealueilta jotka ovat 0–10 V, +/-10 V tai +/- 5 V. Analogisia signaaleja käytetään esimerkiksi paineen, lämpötilan tai virtausnopeuksien mittaamiseen. (13, s.9–10.)

Ohjelmoitavien logiikoiden kehitystyö alkoi vuonna 1968. Ensimmäisen logiikan toi markkinoille Bedford Associates, jonka mallinimi oli 084 (kuva 5). Patentin mallille saatuaan, Bedford Associates perusti logiikkaa jälleenmyyvän ja kehittävän tytäryrityksen Modiconin, joka on lyhenne sanoista Modular Digital Controller. Myöhemmin logiikoita alettiin kutsua nimellä Programmable Logic Controller, josta tuli lyhenne PLC ja suomenkielinen nimi logiikka. (13;14.)



Kuva 5. Modicon-084 (15).

1970-luvulla logiikoiden viestintämahdollisuuksia kehitetiin. Ensimmäinen tämänlainen järjestelmä oli Modiconin Modbus. Tämän mahdollisti kommunikoinnin toisten logiikoiden kanssa ja mahdollisti ohjausjärjestelmien hallitsemisen etäämmältä. (16.) Pääasiallisesti niiden ohjelmointi tapahtui käyttäen tikapuukaavioita (Ladder Diagram), joita laitosinsinöörien ja sähkömiesten oli työnsä puolesta helpompi lukea. (14.)

Logiikoiden fyysinen koko pieneni ja laskentateho lisääntyi 1980-luvulla ja symbolikieli yleistyi ohjelmoinnissa (16). Logiikat liitettiin tietokoneisiin yhteensopivien ohjelmistojen avulla ja ne tekivät ohjausjärjestelmien valvomisen helpommaksi (16).

1990-luvulla logiikkoihin liitettiin graafiset käyttöliittymät (kuva 6), joita kutsutaan lyhenteillä HMI (Human Machine Interface) tai GUI (Graphical User Interface). Ne yhdistettiin tehtaiden keskustietokoneisiin, joista sai tietoa esimerkiksi tuotantomääristä ja laitteiden käyntiajoista. Huoltomiesten ei välttämättä tarvinnut katsoa logiikkaa ollenkaan, koska käyttöliittymät saattoivat näyttää jo itsessään tarpeelliset tiedot. (14.)

Kuva 6. Ohjelmoitava käyttöliittymä (14).

Nykyään logiikoita voi ohjelmoida haluamallaan tyylillä. Yleisin tapa on tehdä ohjelmatiedosto etukäteen tietokoneohjelmalla ja sen jälkeen ladata se logiikalle. Ohjelman toimintaa voi myös simuloida pelkällä tietokoneohjelmalla seuraamalla muuttujien tiloja GVL:stä (Global Variable List) tai simuloidusta graafisesta käyttöliittymästä.

2.4 Logiikoiden ohjelmointitavat

Logiikoiden yleiset ohjelmointitavat perustuvat IEC-61131-3 standardiin, jossa on määritelty viisi eri ohjelmointikieltä (13, s.11; 17, s.2–4). Nämä ovat

- FBD (Function Block Diagram)
- LD (Ladder Diagram)
- ST (Structured Text)
- IL (Instruction List)

- SFC (Sequential Function Chart)

FBD on graafinen toimintalohkokaavio, joka sisältää erilaisia funktiopalikoita. Funktiopalikoita on perinteisten loogisten operaatioiden lisäksi myös monimutkaisempia funktioita, kuten esimerkiksi ajastimia, laskureita ja set/reset-piirejä. Kielellä on kuitenkin vaikea tehdä monimutkaisia ohjelmia, koska se menettää selkeyttään graafisuuden vuoksi. (17, s.3.)

Tikapuuohjelmointi eli LD on mahdollisesti suosituin ohjelmointikieli, koska se muistuttaa sähkörelekaavioita. Tällöin myös ohjelmointia vähän tuntevat saavat tuntuman siihen yleensä nopeammin. LD:ssä vasemmalla puolella on sisääntulot ja oikeapuolella ulostulot, ja signaalien kulkua säädellään niiden väliin asetetuilla kytkimillä. Kieli kuitenkin kärsii samasta ongelmasta kuin FBD, ja siksi sitä käytetään yleensä vain vähän muuttujia ja toimintoja sisältävissä ohjelmissa. (17, s. 2–3.)

ST-kieli on tekstipohjainen ohjelmointikieli, joka sisältää samoja rakenteita kuin C-kieli. Näitä ovat esimerkiksi if-lauseet ja while-silmukat, jotka helpottavat ohjelmoida monimutkaisempia järjestelmiä. Graafisen luonteen puuttumisen vuoksi, ohjelmaa vaikeampi lukea kuin esimerkiksi FBD:tä, mutta koodirivejä voi kuitenkin kommentoida, jolloin sen luettavuutta voi halutessaan helpottaa. (17, s. 3–4.)

Instruction List -ohjelmoinnissa luodaan yksinkertaisia käskyjä, jotka suorittavat koodia yksi komento kerrallaan ja se muistuttaa konekieltä. IL on myös tekstipohjainen kuten ST, mutta se on sitä yksinkertaisempi ja sisältää vähemmän toimintoja. IL soveltuu huonommin monimutkaisempien laitteiden ohjelmointiin ja sen ohjelmointi voi olla haastavampaa kuin esimerkiksi FBD:n. Sen saa kuitenkin ladattua laitteistoihin yleensä helposti ja nopeasti, mutta nykyaikaisten tehdasjärjestelmien muistikapasiteetteihin ja suoritustehoihin verrattuna, näistä ominaisuuksista ei ole enää merkittävää etua. (17, s.3.)

Sekvenssikaaviodiagrammi eli SFC on vuokaavion tapainen ohjelmointikieli, jossa funktiolaatikot kytkeytyvät toisiinsa pystysuorasti. Funktiolaatikot sisältävät ohjelmakoodin, joka voidaan kirjoittaa millä tahansa ohjelmointikielellä, jonka logiikka tuntee. Laatikko aktivoituu, kun siihen menevä siirtymäehto, eli transitio toteutuu. Ohjelma kulkee ylhäältä alas järjestyksessä laatikko kerrallaan, joka helpottaa sen suorituksen ja mahdollisesti

koodissa olevien virheiden seurantaa. Kielen huonona puolena voidaan pitää sen muunnettavuutta kokonaan joksikin edellä mainituksi ohjelmointikieleksi. (17, s.3.)

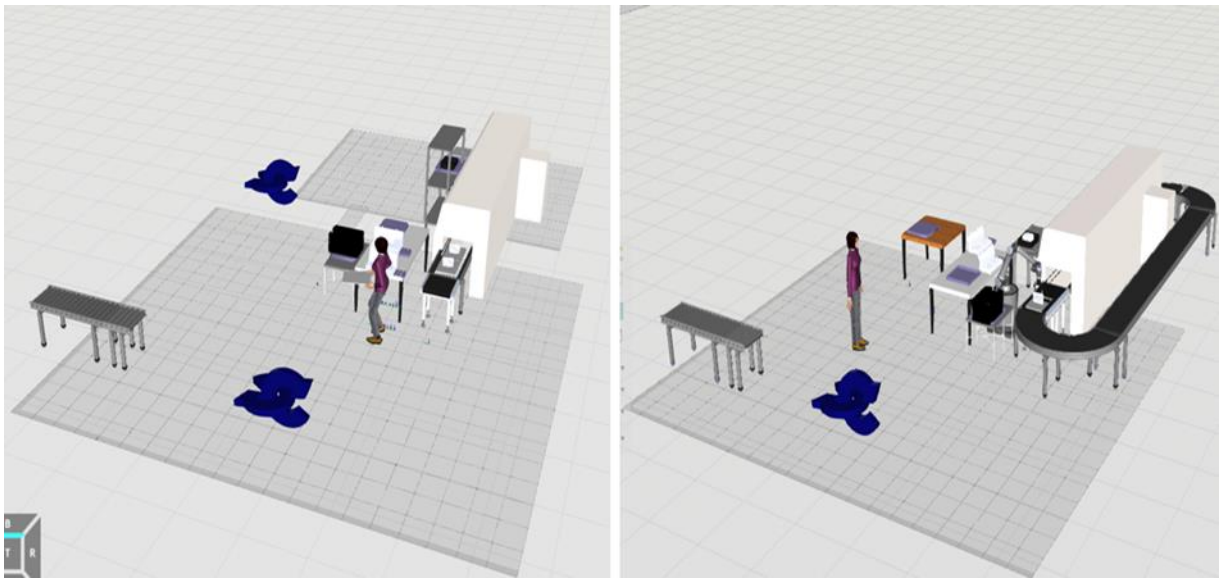
2.5 Simulaatio-ohjelmistot

Robotti voidaan etäohjelmoida simulointiohjelmiston avulla. Se voidaan suorittaa joko 2D- tai 3D mallinnuksena. Sillä nähdään laitteiden sijoittuminen pohjapiirrokseen ja havainnollistetaan niiden toiminta. Simulaation avulla voidaan nähdä ongelmia tuotannossa kuten esimerkiksi pullonkaulat. Myös ihmisille vaaralliset paikat voidaan havaita simuloinnissa. 2D-mallissa voidaan nähdä tuotantolinjaston perustoiminta ja kerätä tarvittava data siitä. 3D-mallissa voidaan näiden lisäksi myös mallintaa laitteiden toiminta, jolloin nähtäviä ongelmia voivat olla mm. robotin liikeratojen fysikaaliset mahdottomuudet. (18, s.16.)

3 Projektin työvaiheet

3.1 Robottisolun mallintaminen

Uuden robottisolun mallintamiseen käytettiin Visual Components -tietokoneohjelmistoa. Ohjelmistolla luotiin kaksi simulaatiota (kuva 7). Ensimmäisessä mallinnettiin nykyinen layout, jossa vain operaattori hoiti koko prosessia. Toisessa puolestaan simuloitiin suunniteltua muutosta layoutiin, jossa robotti on korvannut operaattorin lukuun ottamatta kuljetuslaatikon viemistä läpivientikaappiin. Uuteen layout-kuvaan on myös lisätty hihnakuljetinrata, jossa uunijigit kiertävät takaisin uunin suuaukolle. Kuljettimien nopeudeksi oli määritelty 200 mm/s. Uuden layoutin visuaalinen ilme ei kuitenkaan aivan vastannut todellisuutta. Esimerkiksi makasiinit olivat liukuhihnalla pystyasennossa ja ohjelmiston omasta kirjastosta löytyneet tarjottimet korvasivat uunijigit. Makasiineja myöskään ei oltu jaoteltu tuotekohtaisesti.



Kuva 7. Nykyinen ja suunniteltu layout Visual Componentsilla hahmoteltuina.

Molemmista simulaatioista laadittiin kuvaajat, jotka näyttivät läpivientikaappiin vietyjen kuljetuslaatikoiden (sisältää kahdeksan makasiinia) lukumäärän ajan funktiona (kts. liite 1). Tämä kuvastaa makasiinien kokonaisläpimenoaikaa. Simulointiajoiksi asetettiin 12 tuntia. Ohjelmistossa on mahdollisuus ajan nopeutukseen aina tuhat kertaiseksi ja simulointiaikoja nopeutettiin reaaliajan säästämiseksi.

Kuvaajista voitiin nähdä ero noin kolmen tunnin kohdalla. Tällöin suunniteltu layout muutos ohitti nykyisen layoutin läpivientikaappiin vietyjen kuljetuslaatikoiden määrissä yhdellä kappaleella. Tämä ero kasvoi suoraan verrannollisesti, eli kuuden tunnin kohdalla ero oli kaksi laatikkoa ja yhdeksän tunnin kohdalla kolme laatikkoa. Lopulta 12 tunnin kohdalla ero oli neljä laatikkoa. Uuniin menneiden makasiinien lukumäärä myös kasvoi, jolloin uunin käyttöaste oli parempi robottisolun kanssa.

Simulointiohjelmia vertaillessa tulee kuitenkin muistaa, että molempien layouttien toiminta on ideaalista, eikä siinä otettu huomioon esimerkiksi tuotannon tai laitteistojen toimintaan liittyviä mahdollisia häiriöitä. Tuotannonvirtaus oli asetettu ideaaliseksi, 40 makasiinia tunnissa. Tämä saatiin asettamalla viive makasiinien syöttöasemalle, joka oli 900 sekuntia aina neljän makasiinin jälkeen. Oikeassa tuotannossa makasiinien virtaus on epätasaisempi. Vuorokaudessa makasiineja menee uunin läpi noin 20–50 kappaletta. Simulointi on siis suuntaa antava robottisolun tehokkuudesta.

3.2 Laitteiston valinta

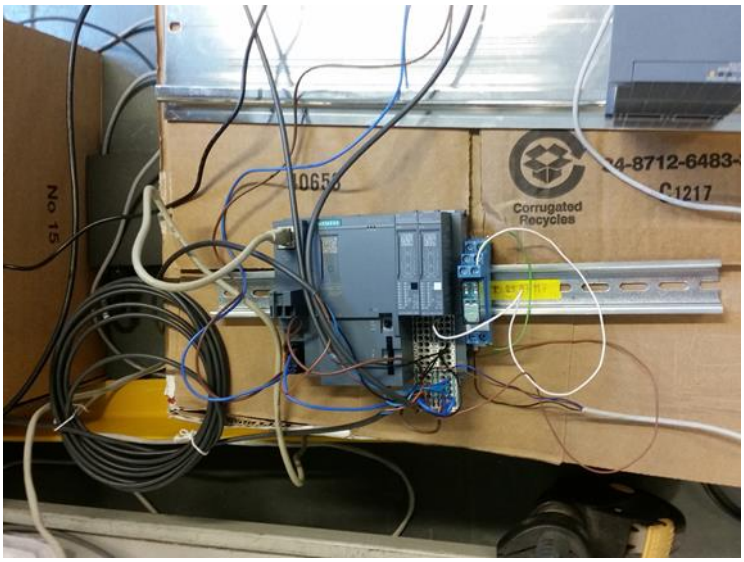
Robottisolu tullaan rakentamaan seuraavalla periaatteella: logiikka ohjaa robottia antamalla sille numeroituja käskyjä, joiden perusteella robotti suorittaa käskyn. Näitä käskyjä voivat olla esimerkiksi työkalun vieminen tiettyyn pisteeseen tai robottitarttujan aukaiseminen ja sulkeminen. Kun käsky on suoritettu, robotin tulee lähettää kuittausviesti takaisin logiikalle, koska halutaan varmistua logiikan ja robotin olevan tietoisia toistensa aikeista. Logiikkaa tarvitaan siksi, että robotin tulee kommunikoida tehtaan tietokannan kanssa ja vastaan ottaa sieltä tieto tuotteen viivakoodista.

3.2.1 Robotti

Työssä käytettävä nivelvarsirobotti on tanskalaisen Universal Robotsin valmistama UR10e. Se on suunniteltu yhteistyörobotiksi, eli ihmiset voivat työskennellä sen kanssa samoissa tiloissa. Robottiin voidaan määritellä turvarajat nivelien törmäysvoimille. Rajojen ylittyessä robotti pysähtyy välittömästi ja menee vikatilaan. Robotti toimii verkkovirralla ja sen kantama maksimikuorma on 10 kg. (19, s. II-18, I-71.)

3.2.2 Logiikka

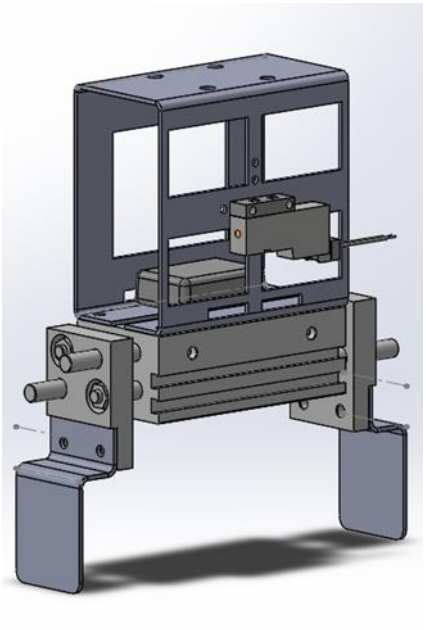
Logiikaksi valikoitui Siemensin Simatic ET 200SP (kuva 8), koska sellainen on jo käytössä samanlaisessa robotissa eräässä toisessa tehtaan tuotantoprosessissa. Logiikka ja sen ohjelmoimiseen vaadittava TIA-portal-ohjelmisto piti tilata erikseen. Logiikan valinnalla halutaan varmistaa, että laitteistot eri puolilla tehdasta olisivat mahdollisimman saman-laisia keskenään. Logiikassa on erillinen jännitelähde ja yksi irroitettava I/O-moduuli.



Kuva 8. ET 200 SP-logiikka.

3.2.3 Tarttuja

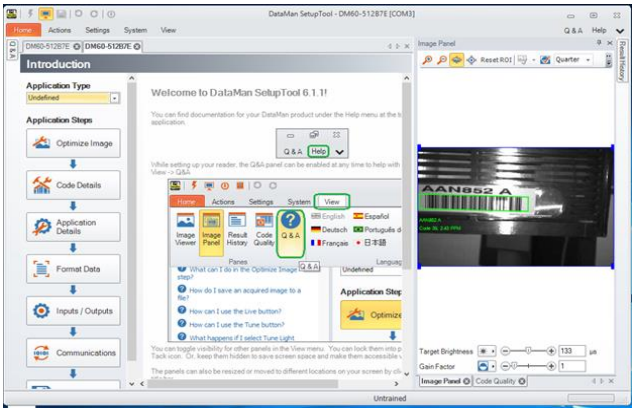
Tarttujaksi valikoitui SMC:n valmistama paineilmatoiminen MHL2-16D2. Aluksi tarttujaksi harkittiin sähköllä toimivaa tarttujaa, mutta paineilmalla toimiva tarttuja on mekaanisesti yksikertaisempi ja siksi helpommin huollettava. Paineilmatoimiset tarttujat olivat myös halvempihintaisia verrattuna sähköisiin tarttujiin. Tarttuja ei ollut kuitenkaan suoraan yhteensopiva robotin kanssa, joten sitä varten piti teettää erillinen piirilevy ja metallisia kiinnikkeitä. Nämä olivat tarttujan kiinnike robotin niveleen ja tarttujan leuat, jolla makasiineista saa otteen. Tarttujan vaatimat metalliset kiinnikkeet suunniteltiin ja mittakuvat piirrettiin käyttäen Solidworks-ohjelmistoa (kts. kuva 9, liite 2 ja liite 3).



Kuva 9. Koko tarttuja hahmoteltuna SolidWorks-ohjelmistolla.

3.2.4 Viivakoodinlukija

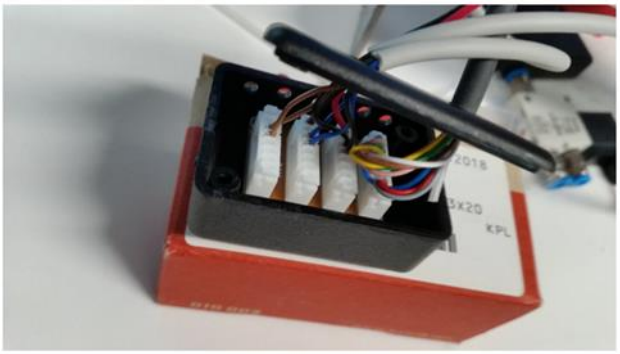
Viivakoodinlukija on Cognexin valmistama Dataman 60. Jotta viivakoodin asetuksia voitiin säätää, tuli Cognexin www-sivuilta ladata Dataman Setup tool-niminen konenäköohjelma (kuva 10). Setup toolilla voidaan määrittää esimerkiksi luettava viivakoodityyppi, viivakoodinlukijan kommunikointitapa ja säätää kameran valoasetuksia.



Kuva 10. Näkymä Dataman 6.1.1 Setup toolista.

3.3 Asennukset

Ensimmäiset sähköasennukset tehtiin tarttujalle (kuva 11). Tarttujan leukoja ohjataan sähkötoimisella paineilmaventtiilillä. Venttiili yhdistettiin robotin sähköjärjestelmään käyttäen erillistä piirilevyä. Tarttujassa on myös magneettianturit, jotka antavat tarttujan auki/kiinni-tilatiedot. Piirilevyssä on paikat robotin kaapelille ja niihin kytketyille I/Oille. Piirilevyn sähköjohdot liitetään siihen liittimien kautta, joten asennuksia voidaan helpommin muuttaa jälkikäteen.



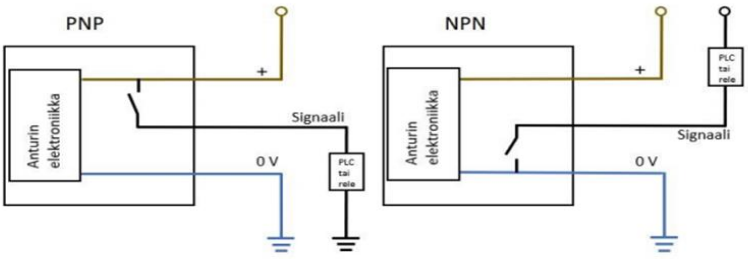
	Digital Outputs 0	Digital Outputs 1	Digital Input 1	Digital Inputs 0	Analog 3 or RS485	Analog 2 or RS485	POWER	Ground
Robot	BLUE	PINK	GREEN	YELLOW	BROWN	WHITE	GRAY	RED
	RKMV 8-354	RKMV 8-354	RKMV 8-354	RKMV 8-354	RKMV 8-354	RKMV 8-354	RKMV 8-354	RKMV 8-354
	BLACK VQD1121W (Venttiili)		BLACK D-Y7P (auki/kiinni) tilatiedot	BLACK D-Y7P (auki/kiinni) tilatiedot				
0V			BLUE D-Y7P (auki/kiinni) tilatiedot	BLUE D-Y7P (auki/kiinni) tilatiedot				
24V	RED VQD1121W (Venttiili)		BROWN D-Y7P (auki/kiinni) tilatiedot	BROWN D-Y7P (auki/kiinni) tilatiedot				

Kuva 11. Tarttujan sähköliitännät ja sitä kuvaava kytkentätaulukko.

Koska robottitarttuja oli paineilmatoiminen, siihen piti saada paineilmaa. Valitettavasti kiinteistön oma kompressori ei ollut käytettävissä, joten paineilman tuottamiseen päätettiin hankkia uusi kompressori (kuva 12). Kompressoriin asennettiin 3 mm halkaisijaltaan oleva letku, jossa oli myös liitin paineilmamittarille. Letkua varattiin noin 6 metriä, jotta sen pituus riittäisi kaikkiin robotin liikkeisiin.

Kuva 12. Paineilman tuottava kompressori.

Toiset sähköasennukset tehtiin logiikalle, johon kytkettiin valokennot ja paineilmamittari. Valokennoja on neljä kappaletta ja niiden avulla ohjataan robotin toimintaa. Jokainen valokenno vastaa yhtä robottisolun toimintapistettä ja robotti toimii valokennojen antamien tilatietojen mukaisesti. Paineilmamittaria varten tarvittiin erillinen sähkörele, jonka kautta logiikka pystyi lukemaan mittarin antamat boolean-tyyppiset tilatiedot, koska mittari käyttää toimiakseen niin sanottua NPN-tekniikkaa (kuva 13), jossa rele katkoo mittarille tulevan jännitteen nollatasoa (20 s.16).



Kuva 13. PNP- ja NPN-tekniikan ero (20 s.16).

3.4 Robottisolun mallinnus

Robottisolun layout on mallinnettu kuvassa 14. Robotti asennettiin 0,6 metriä korkealle jalustalle ja se oli asetettu kuormalavan päälle, jotta robottia on helpompi siirrellä tarvittaessa. Robotti myös sijoitettiin niin, että se ei ylettyisi sitä ympäröiviin seiniin tai ikkunoihin. Robotin ympärille tehtiin neljä pylvästä pahvilaatikoista, joiden jokaisen päälle oli lisätty valokenno. Pylväät päätettiin tehdä pahvista, koska esimerkiksi robotin mahdollisesti tehdessä virheliikkeitä pahvi antaisi periksi törmäystilanteissa ja todennäköisyys jonkin komponentin hajoamiselle olisi mahdollisimman pieni. Nämä pylväät simuloivat makasiinin syöttöpistettä, uunin liukuhihnaa, jäähypaikkaa ja kuljetuslaatikon paikkaa. Robottisolun suunniteltu toimintalogiikka on seuraava:

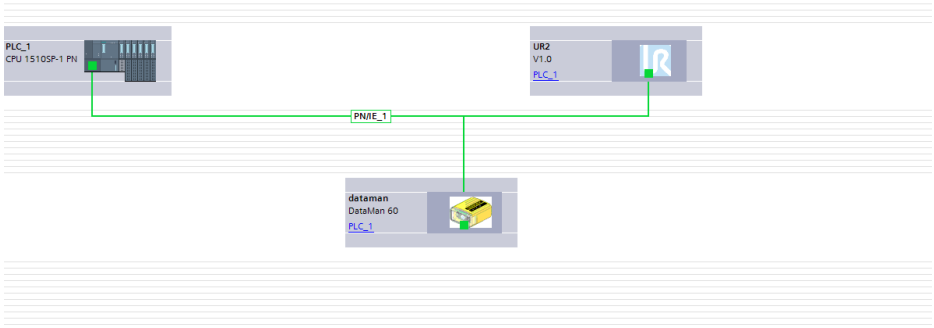
- Robotti hakee makasiinin syöttöpisteeltä.
- Robotti vie makasiinin uuninkuljetinradalle.
- Robotti hakee makasiinin uunikuljetinradalta jäähyjigille.
- Robotti lukee makasiinin viivakoodin ja asettaa makasiinin kuljetuslaatikkoon.

Kuva 14. Testipaikan layout.

3.5 Konfiguraatiot

Jotta ohjelmointi sujuisi mahdollisimman vaivattomasti, robotti, viivakoodinlukija ja logiikka piti saada ensin muodostamaan yhteyden toisiinsa TIA-Portal-ohjelmiston kautta (kuva 15). Ohjelmistolla tehtiin muutama projekti ihan vain yhteyksien muodostamista varten. Aluksi logiikkaa konfiguroidessa ongelmia aiheuttivat mm. sen I/O-kortit, joista jompikumpi oli yleensä virhetilassa. Ongelma poistui, kun joka ohjelmointikerran päätteeksi robotti sammutettiin ennen logiikkaa ja logiikka asetettiin Stop-tilaan ennen sen sammuttamista. Kun paineilma oli lisätty, huomattiin että tarttujan venttiili ei toiminut. Vika havaittiin sähköasennuksissa, jossa venttiili oli kytketty väärinpäin. Oletuksena oli, että robotti katkoo venttiilin 24 V:n jännitettä eli käyttää PNP-tekniikkaa, mutta oikeasti se toimii päinvastoin, eli käyttää NPN-tekniikkaa kuten paineilmamittarikin. Robotin ohjekirjassa se mainitaan työkalun digitaalisten ulostulojen ohjaustapana (19, s. I-43).

Robotti, viivakoodin ja logiikka yhdistettiin toisiinsa modeemin välityksellä. Pystyväkseen kommunikoimaan keskenään, on TIA-Portal-ohjelmistoon ladattava näiden laitteiden tukitiedostot, eli GSD-filet (General Station Description). GSD-tiedostojen avulla logiikka tunnistaa laitteet Profinet-kommunikaatioväylän kautta (21). Näille laitteille on määritelty muuttujarekisterit, joiden arvoja logiikka voi lukea tai kirjoittaa. Kun GSD-tiedostot ovat asennettu TIA-Portaaliin, on määriteltävä näiden muuttujien rekisteriosoitteet logiikan I/O:issa. Viivakoodinlukijaa konfiguroidessa näiden osoitteiden määrittelyn helpottamiseksi internetistä voi ladata sitä varten erillisen Excel-taulukon. Siihen voi täyttää viivakoodin laitetiedoista löytyvät muuttujat järjestyksessä ja taulukko määrittelee laitteen I/O-osoitteet automaattisesti. Lopuksi osoitelista voidaan kopioida exelistä PLC-muuttujataulukkoon. TIA-Portalin ollessa online-tilassa logiikan kanssa, pystyy tarkistamaan muuttujien toimivuuden. Esimerkiksi vaihtamalla jonkin boolean-tyyppisen muuttujan tiloja. Kun yhteydet logiikan ja robotin välillä todettiin toimiviksi, ohjelmointi voitiin aloittaa.



Kuva 15. Näkymä TIA-Portalin laitetopologiasta.

Yhdessä vaiheessa ongelmia ilmeni TIA-Portal-ohjelmiston lisensseissä. Kun ohjelma oli asennettu, se toimi ongelmitta noin kolme viikkoa. Tämän jälkeen lisenssit vanhenivat ja TIA-Portal eväsi pääsyn mm. sen ohjelmointi- ja laitekirjastoihin, jolloin ohjelmointia oli mahdotonta jatkaa. Kävi ilmi, että käytössä oli vain demo-lisenssit ja oikea lisenssi oli kadonnut ohjelman mukana tulleelta muistitikulta. Tämä ongelma ratkesi ottamalla yhteyttä Siemensin asiakaspalveluun, josta myönnettiin uusi lisenssiavain kadonneen tilalle.

3.6 Ohjelmointi

3.6.1 Työnkierron ohjelmointi

Ohjelma päätettiin tehdä logiikalle FBD-kielellä, koska se sisältää ohjelman toteuttamiseen sopivat datablokit, kuten ajastimet ja move-operaatiot, joilla voidaan yli kirjoittaa muuttujien integer-arvoja. Robotilla puolestaan ohjelma pyörii SFC-tyyppisesti.

Robotin suorittama työnkierto jaettiin järjestysnumeroihin, jotka se suorittaa järjestyksessä numerosta 1 alkaen (kts. liite 4). Logiikka kirjoittaa ensimmäisen järjestysnumeron robotin input-integer-rekisteriin. Robotti lukee arvon ja kirjoittaa seuraavan järjestysnumeron output-integer-rekisteriin logiikan luettavaksi. Tällä metodilla varmistetaan, että logiikka ja robotti ovat koko ajan tietoisia toisistaan, eli logiikka tietää mitä vaihetta robotti on suorittamassa. Robotti ei myöskään lähde suorittamaan tiettyä työvaihetta, ellei se

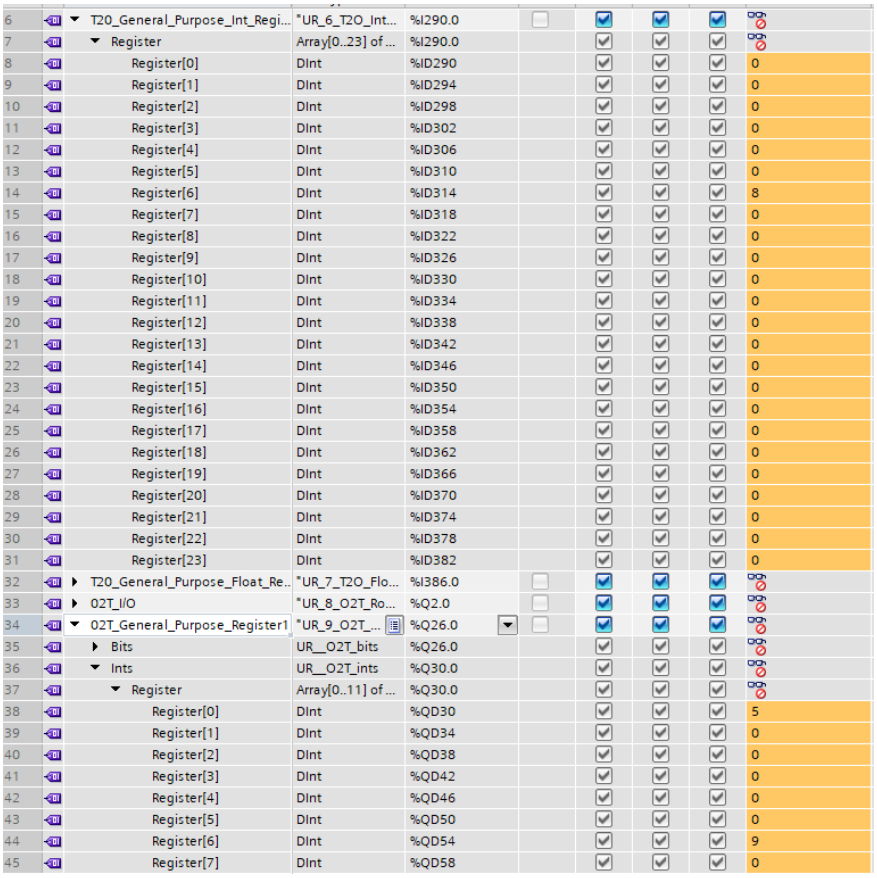
ole saanut logiikalta siihen tarvittavaa numeroa. Ainoan poikkeuksen numeroiden kuittausjärjestykseen tekee viivakoodinlukija, jolle logiikka lähettää käskyn viivakoodin lukuun numerolla 49 ja saa siitä kuittauksen numerolla 50. Tämän jälkeen logiikka ja robotti jatkavat kommunikointia normaalisti numerojärjestyksessä. Kun työnkierto saavuttaa viimeisen numeronsa, robotti palaa takaisin alkupisteeseensä ja nollaa integer-rekisterinsä ja logiikka myös nollaa omansa. Tällä halutaan varmistaa, että jos robotti on toimettomana pitkiäkin aikoja, mitkään numerot eivät jää sotkemaan seuraavan työnkierron alkua. Robotti kuitenkin kirjoittaa aina oman output-integer-rekisterinsä nollaksi alkupisteeseen saavuttuaan, vaikka koko työnkiertoa ei olisikaan saavutettu loppuun.

3.6.2 PLC-muuttujat

Logiikalle luotiin kolme eri muuttujalistaa (kts. liite 5). Ensimmäinen sisälsi logiikkaan kytketyt komponentit, eli anturit ja paineilmamittarin. Näiden lisäksi muuttujalistassa oli logiikan aloituspainike, joka on nimeltään ”Start/StopKey” ja viivakoodin luvun epäonnistumista kuvaavan muuttujan aktivoiva datablokki. Toinen muuttujalista sisälsi viivakoodinlukijan muuttujat ja kolmas robottiin kuuluvat muuttujat.

Robotin input integer-rekisteripaikka 0 on varattu Start/StopKeylle. Ollessaan aktivoituna sen arvo on 5. Start/StopKeyn oli tarkoitus toimia ohjelmointia helpottavana apumuuttujana, jolla annettiin lupa robotille lähteä suorittamaan ohjelmasekvenssiä ja nollattiin tarvittaessa työnkierron input-integer-rekisteri. Tämä muuttuja poistettiin, kun robotin ohjelma todettiin toimivaksi ja se korvattiin robotin omalla ” PR: Is program running”-muuttujalla.

Tämä muuttuja aktivoituu, kun ohjelman suoritus käynnistetään robotilta. Ohjelman suorituksen keskeytyessä muuttuja menee false-tilaan ja siten nollaa työnkierron integer-rekisterin. Jos ohjelman suoritus halutaan aloittaa uudestaan, robotti on aina ajettava aloituspisteeseensä. Tämä tarkoittaa sitä, että työnkierron molemmat rekisterit nollautuvat, jolloin työnkierron uudelleen aloittaminen on turvallista. Seuraavat input-integer rekisteripaikat 1–4 on varattu anturien tilatiedoille. Vaikka anturit antavat tilatietonsa boolean-tyyppisesti, niin ohjelmoinnin kannalta oli helpompaa antamaan ne tilatiedot integer-muodossa. Työnkierron input- ja output-integer-rekisteripaikoiksi oli määritelty molemmille paikat numero 6 (kuva 16).



6	T20_General_Purpose_Int_Regi...	"UR_6_T2O_Int...	%I290.0		☑	☑	☑		
7	Register	Array[0..23] of ...	%I290.0		☑	☑	☑		
8	Register[0]	DInt	%ID290		☑	☑	☑	0	
9	Register[1]	DInt	%ID294		☑	☑	☑	0	
10	Register[2]	DInt	%ID298		☑	☑	☑	0	
11	Register[3]	DInt	%ID302		☑	☑	☑	0	
12	Register[4]	DInt	%ID306		☑	☑	☑	0	
13	Register[5]	DInt	%ID310		☑	☑	☑	0	
14	Register[6]	DInt	%ID314		☑	☑	☑	8	
15	Register[7]	DInt	%ID318		☑	☑	☑	0	
16	Register[8]	DInt	%ID322		☑	☑	☑	0	
17	Register[9]	DInt	%ID326		☑	☑	☑	0	
18	Register[10]	DInt	%ID330		☑	☑	☑	0	
19	Register[11]	DInt	%ID334		☑	☑	☑	0	
20	Register[12]	DInt	%ID338		☑	☑	☑	0	
21	Register[13]	DInt	%ID342		☑	☑	☑	0	
22	Register[14]	DInt	%ID346		☑	☑	☑	0	
23	Register[15]	DInt	%ID350		☑	☑	☑	0	
24	Register[16]	DInt	%ID354		☑	☑	☑	0	
25	Register[17]	DInt	%ID358		☑	☑	☑	0	
26	Register[18]	DInt	%ID362		☑	☑	☑	0	
27	Register[19]	DInt	%ID366		☑	☑	☑	0	
28	Register[20]	DInt	%ID370		☑	☑	☑	0	
29	Register[21]	DInt	%ID374		☑	☑	☑	0	
30	Register[22]	DInt	%ID378		☑	☑	☑	0	
31	Register[23]	DInt	%ID382		☑	☑	☑	0	
32	T20_General_Purpose_Float_Re...	"UR_7_T2O_Flo...	%I386.0		☑	☑	☑		
33	O2T_I/O	"UR_8_O2T_Ro...	%Q2.0		☑	☑	☑		
34	O2T_General_Purpose_Register1	"UR_9_O2T_...	%Q26.0		☑	☑	☑		
35	Bits	UR__O2T_bits	%Q26.0		☑	☑	☑		
36	Ints	UR__O2T_ints	%Q30.0		☑	☑	☑		
37	Register	Array[0..11] of ...	%Q30.0		☑	☑	☑		
38	Register[0]	DInt	%QD30		☑	☑	☑	5	
39	Register[1]	DInt	%QD34		☑	☑	☑	0	
40	Register[2]	DInt	%QD38		☑	☑	☑	0	
41	Register[3]	DInt	%QD42		☑	☑	☑	0	
42	Register[4]	DInt	%QD46		☑	☑	☑	0	
43	Register[5]	DInt	%QD50		☑	☑	☑	0	
44	Register[6]	DInt	%QD54		☑	☑	☑	9	
45	Register[7]	DInt	%QD58		☑	☑	☑	0	

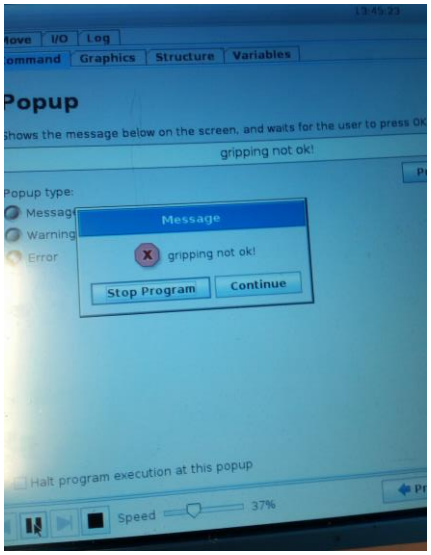
Kuva 16. Työnkierron seurantaa robotin integer-rekistereistä.

3.7 Testaus

Ennen lopullisten ohjelmaversioiden syntymistä logiikalle ja robotille (kts. liite 6 ja liite 7), niistä tehtiin eri versiota. Ensimmäinen versio tehtiin vain toiminnan peruslogiikan määrittämiseksi. Robotti toimi pelkästään anturitietojen mukaisesti ja viivakoodinlukija jätettiin pois. Myöskään tässä versiossa mitään virhetilanteita ei otettu huomioon ja pääohjelmaa ei oltu pilkottu aliohjelmiin. Kun tämä peruslogiikkaohjelma oli saatu valmiiksi, tästä oli hyvä kehittää aina parempia versioita, johon liitettiin enemmän muuttujia ja uusia mahdollisia virhetilanteita. Tällä tavalla ohjelmointia helpotettiin ja nopeutettiin. Lopullisessa ohjelmaversiossa virhetilanteet on otettu huomioon keskeyttämällä robotin toiminta seuraavissa tilanteissa:

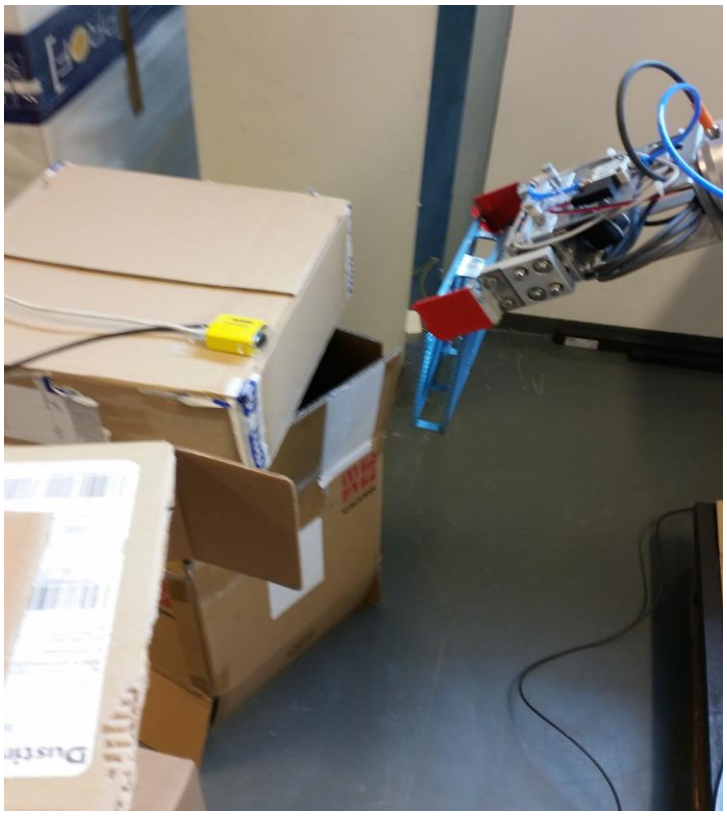
- Paineilmaa ei ole tarpeeksi.
- Tarttuja on täysin kiinni (makasiinia ei ole tarttujassa).
- Viivakoodi ei ole luettavissa.
- Ohjelman suoritus on keskeytetty robotilta.
- Seuraavalla työasemalla on makasiini poimimatta.

Jos paineilmaa ei ole tarpeeksi robotti keskeyttää ohjelman suorittamisen ja antaa virhetilatiedon paineilman puuttumisesta ponnahdusikkunan kautta (kuva 17). Samoin tapahtuu myös silloin kun robotin tarttuja on täysin kiinni, eli makasiinia ei ole tarttujassa. Jos logiikka ei saa viivakoodinlukijalta tiettyä attribuutin arvoa, robotti siirtää makasiinin syrjään ja antaa tästä virheilmoituksen. Jos robotilta painetaan kesken ohjelmasuorituksen stop-painiketta, logiikka nollaa robotin input integer-rekisterin, jotta robotti ei lähde ohjelman uudelleen käynnistyessä suorittamaan keskeytynyttä työvaihetta. Robotti ei myöskään lähde suorittamaan työvaihetta, mikäli seuraavalla työasemalla on edelleen makasiini. Makasiinin jäädessä tarttujaan ohjelman keskeydyttyä, robotti tunnistaa tämän ja laskee makasiinin syöttöasemalle päästäen siitä irti ja palaamalla sitten takaisin alkupisteeseensä.



Kuva 17. Robotin antama ilmoitus tartuntavirheestä.

Kun ohjelma todettiin toimivaksi myös näiden virhetilanteiden kanssa, logiikan apumuuttuja Start/StopKey-aloituspainike vaihdettiin robotin ” PR: Is program running”-muuttujalla, jolloin integer-rekisterien nollaaminen tapahtui automaattisesti. Tämän jälkeen TIA-Portal ohjelmisto suljettiin tietokoneelta ja työnkiertoa testattiin muutaman kokonaisen työnkierron ajan logiikan toimiessa itsenäisesti (kuva 18).



Kuva 18. Makasiini vietynä viivakoodinlukijalle testausvaiheessa.

4 Yhteenveto

Insinöörityön tavoitteena oli ensin 3D-mallintaa robottisolu uunikuljettimelle ja sen jälkeen hankkia jo valmiina varastossa olevalle nivelvarsirobotille solun toimintaan vaadittavat komponentit. Tämän jälkeen komponentit tuli yhdistää toisiinsa ohjelmoitavan logiikan välityksellä ja lopuksi luoda ohjelma, joka vastaa robottisolun toimintaa.

3D-mallinnus suoritettiin Visual Components-tietokoneohjelmistolla ja robottisolun toimintaa verrattiin nykyiseen layoutiin laatimalla kuvaajat molempien toiminnasta 12 tunnin

ajalta, jolloin voitiin havaita, voisiko suunniteltu muutos olla tehokkaampi kuin nykyinen prosessi.

Nivelvarsi robotilla ei ollut aluksi sopivaa tarttujaa, joten sellainen piti tilata erikseen ja tarttujan kiinnittämiseksi robottiin piti suunnitella erillinen kiinnike. Myös tarttujan leuat piti teettää erikseen. Näiden suunnitteluun käytettiin Solidworks-3D suunnitteluohjelmistoa ja malleista laadittiin mittakuvat. Tarttujaa varten teetätettiin piirilevy, johon kytkettiin robotilta tuleva kaapeli, tarttujan venttiili ja tarttujan tilatietoja kuvaavat anturit.

Robotin ohjaaminen päätettiin suorittaa ohjelmoitavan logiikan välityksellä ja logiikaksi valikoitui Siemensin valmistama ET 200 SP, koska eräs toinen samanlainen robotti myös käyttää sellaista. Logiikan ohjelmointia varten piti myös tilata TIA-portal ohjelmointityökalu ja siihen toimiva lisenssi. Logiikan ja ohjelmointityökalun toimitusajoissa ilmeni pientä viivettä, mutta onneksi ohjelmointi pystyttiin aloittamaan ajoissa. Myös TIA-Portalin lisenssit vanhenivat eräässä vaiheessa, mutta ne saatiin palautetuksi kahden päivän kuluessa. Tämä toisaalta vei hieman aikaa ohjelmointiosuudesta.

Kun robotin ja logiikan välille oli saatu muodostettua yhteys, ohjelmointi aloitettiin. Robottisolun toiminnasta luotiin ensin perusohjelmisto, jossa ei oltu otettu huomioon poikkeavia tilanteita ohjelmasuorituksessa. Kun perusohjelma oli valmis, ohjelmaan liitettiin viivakoodinlukija ja poikkeustilanteet huomioitiin. Lopuksi TIA-Portal ohjelmisto suljettiin ja robotin toimintaa testattiin itsenäisesti logiikan kanssa. Ohjelmaan olisi vielä voinut lisätä testitietokannan, joka olisi lajitellut makasiinit erillisiin kuljetuslaatikoihin viivakoodista luettujen tuotetietojen perusteella. Tämä kuitenkin jätettiin pois kiireisen aikataulun vuoksi.

Opinnäytetyötä tehtäessä pääsin hyödyntämään lähes kaikkia ammattikorkeakoulun tarjoamia automaatioalan opintoja. Näitä olivat mm. ohjelmointi ja TIA-portal-ohjelmiston käyttö, konenäkösovelluksen hyödyntäminen ja metalliosien suunnittelu Solidworks-ohjelmistolla. Tarvitsin myös sähkötekniikan alan tuntemusta esimerkiksi tarttujan venttiilin ja paineilmamittarin toiminnassa, jotka toimivat NPN-tekniikalla. Ohjaajani Juha-Pekka Larramäki auttoi minua näissä asioissa ja hän myös suunnitteli robottitarttujaan tarvittavan piirilevyn.

Lähteet

1 History. Decade after decade of Innovation. 2020. Murata Electronics. Verkkoaineisto. <https://www.murata.com/en-us/about/company/history?intcid5=com_xxx_xxx_cmn_nv_xxx>. Luettu 4.5.2020.

2 Yrityksenä. Verkkoaineisto. 2020. Murata Electronics. Verkkoaineisto <https://muratafinland.com/murata-electronics-yrityksena/>. Luettu 4.5.2020.

3 Lankinen Pasi. 2019. Synnyttävätkö robotteja unelmat, osingot vai ihmisten tarpeet. Verkkoaineisto. Robologi. Metropolia. <https://blogit.metropolia.fi/robologi/avainsana/robotiikan-historia/>. Luettu 4.5.2020.

4 Huttunen Jyri. 2004. Robotiikan historia. Verkkoaineisto. Tietojen käsittelytieteen historia-seminaarinesitelmä. Helsingin yliopisto. <https://www.cs.helsinki.fi/u/kerola/tkhist/k2004/alustukset/robotiikka/roboalus.pdf>. Luettu 4.5.2020.

5 Aalto, H. Heilala, J. Hirvelä, T. Kuivanen, R. Laitinen, M. Lehtinen, H. Lempiäinen, J. Lylynoja, A. Renfors, J. Selin, K. Siintoharju, T. Temmes, J. Tuovila, T. Veikkolainen, M. Vihinen, J. Virtanen, A. 1999. Robotiikka. Suomen Robotiikkayhdistys Ry. Vantaa: Talentum Oyj.

6 Malone Bob. 2011. George Devol: A life Devoted to Invention, and Robots. Verkkoaineisto. IEE Spectrum. < https://spectrum.ieee.org/automaton/robotics/industrial-robots/george-devol-a-life-devoted-to-invention-and-robots>. Luettu 4.5.2020.

7 Haapalainen Tomi. 2011. ABB robotin käyttöönotto ja ohjelmointi konenäkösovelluksessa. Verkkoaineisto. Opinnäytetyö. Satakunnan ammattikorkeakoulu. Theseus-tietokanta. <https://www.theseus.fi/bitstream/handle/10024/33889/Haapalainen_Tomi.pdf?sequence=1&isAllowed=y>. Verkkoaineisto. Luettu 4.5.2020.

8 Teollisuusrobotti. 2020. Verkkoaineisto. Wikipedia. <https://fi.wikipedia.org/wiki/Teollisuusrobotti>. Luettu 4.5.2020.

9 Salmi Timo. 2014. Robotiikka–monien mahdollisuuksien tekniikkaa. Verkkoaineisto. VTT. <https://www.vtt.fi/Impulssi/Pages/Robotiikka-%E2%80%93-monien-mahdollisuuksien-tekniikkaa.aspx. >Luettu 16.3.2020.

10 Pukkinen Veli-Matti. 2011. Robottisolun simulointi. Verkkoaineisto. Opinnäytetyö. Seinäjoen Ammattikorkeakoulu. Theseus-tietokanta. Verkkoaineisto <https://www.theseus.fi/bitstream/handle/10024/28312/Pukkinen_V-M.pdf?sequence=1&isAllowed=y.> Luettu 4.5.2020.

11 Tervola Janne. 2020. Yhteistyörobotit eivät väsy tuotannossa – näin ne otetaan tehokkaasti käyttöön. Verkkoaineisto. Tekniikka & Talous. < https://www.tekniikkatalous.fi/uutiset/yhteistyorobotit-eivat-vasy-tuotannossa-nain-ne-otetaan-tehokkaasti-kayttoon/72a195d9-6f34-4764-84fd-894ab2f0b48b>. Luettu 4.5.2020.

12 Helminen Jesse. 2015. Pakkaussovelluksessa käytettävän tarttujan suunnittelu. Opinnäytetyö. Verkkoaineisto. Lahden ammattikorkeakoulu. Theseus-tietokanta. <https://www.theseus.fi/bitstream/handle/10024/87178/Helminen_Jesse.pdf?sequence=1.> Luettu 4.5.2020.

13 Kuusisto Ari. 2008. Ohjelmoitavien logiikoiden kartoitus. Opinnäytetyö. Verkkoaineisto. Satakunnan ammattikorkeakoulu. Theseus-tietokanta. <https://www.theseus.fi/bitstream/handle/10024/735/Kuusisto%20Ari.pdf?sequence=1&isAllowed=y.> Luettu 4.5.2020.

14 Lander Arnold. 2019. Programmable Logic Controllers: The Evolution of a Disruptive Technology. Verkkoaineisto. Engineering.com <https://new.engineering.com/story/programmable-logic-controllers-the-evolution-of-a-disruptive-technology.> Luettu 4.5.2020.

15 DaveM580. 2017. Stepwise approach to upgrading your automation control systems. Case example: Modicon. Designspark. Verkkoaineisto. < https://www.rs-online.com/designspark/stepwise-approach-to-upgrading-your-old-control-system>. Luettu 4.5.2020.

16 PLC history. 2011. Verkkoaineisto. PLCS.net <http://www.plcs.net/chapters/history2.htm.> Luettu 4.5.2020.

17 Heinimäki Juho. 2018. Simuloitua tuotantolinjastoa ohjaavan logiikan toteuttaminen Rasberry Pillä. Kanditaatin työ. Verkkoaineisto. Tampereen teknillinen yliopisto. <https://trepo.tuni.fi/bitstream/handle/123456789/26995/Heinim%C3%A4ki.pdf?sequence=4&isAllowed=y.> Luettu 4.5.2020.

18 Lehtinen Tommi. 2019. Simuloinnilla ja virtuaalitodellisuudella tehokkuutta tuotannon kehittämiseen. Opinnäytetyö. Verkkoaineisto. Satakunnan ammattikorkeakoulu. Theseus-tietokanta. <https://www.theseus.fi/bitstream/handle/10024/265735/Lehtinen_Tommi.pdf?sequence=2&isAllowed=y.> Luettu 4.5.2020.

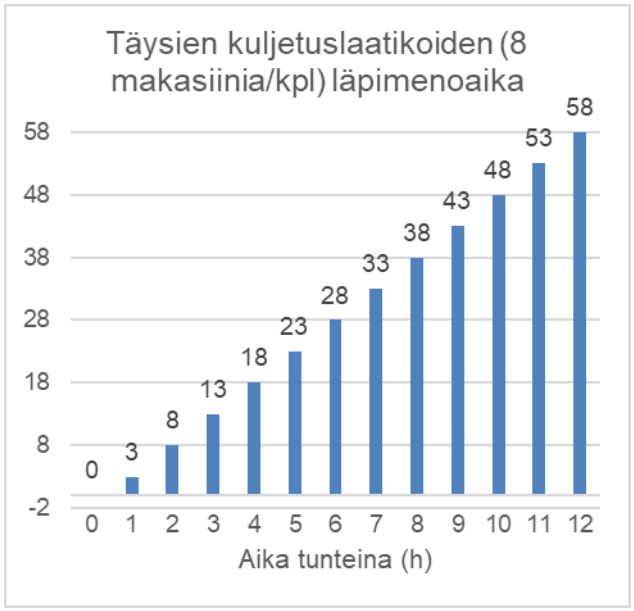
19 Universal Robots e-Series User Manual UR10e Version 5.0.2 Original instructions. 2018. Universal Robots. Käyttöohjedokumentti.

20 Huhtanen Tomi. 2019. Anturitekniikan harjoitustyö. Opinnäytetyö. Verkkoaineisto. Seinäjoen ammattikorkeakoulu. Theseus-tietokanta. <https://www.theseus.fi/bitstream/handle/10024/166328/Huhtanen_Tomi.pdf?sequence=2&isAllowed=y> Luettu 4.5.2020.

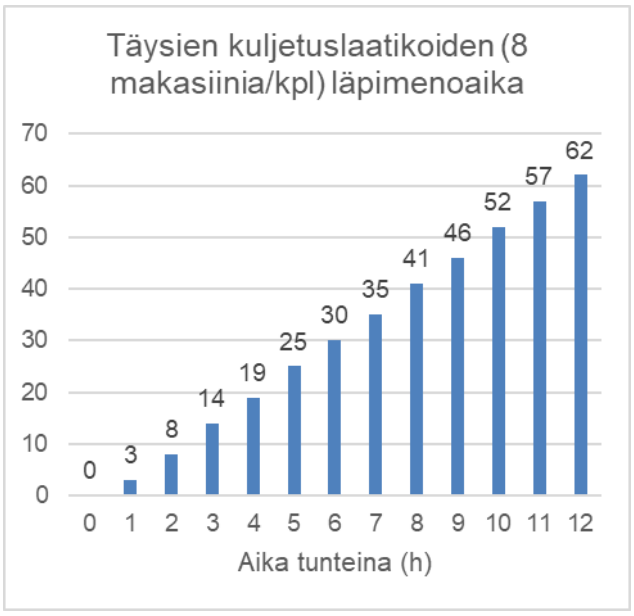
21 What is GSD-file? 2019. Profinet GSD-file basics. Verkkoaineisto. Profinet University. <https://profinetuniversity.com/profinet-basics/profinet-gsd-file-basics/>. Luettu 4.5.2020.

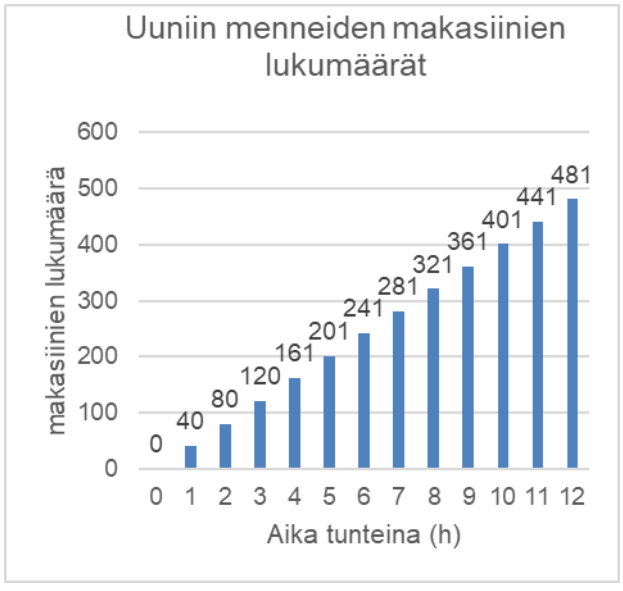
Visual Components-simuloinnit

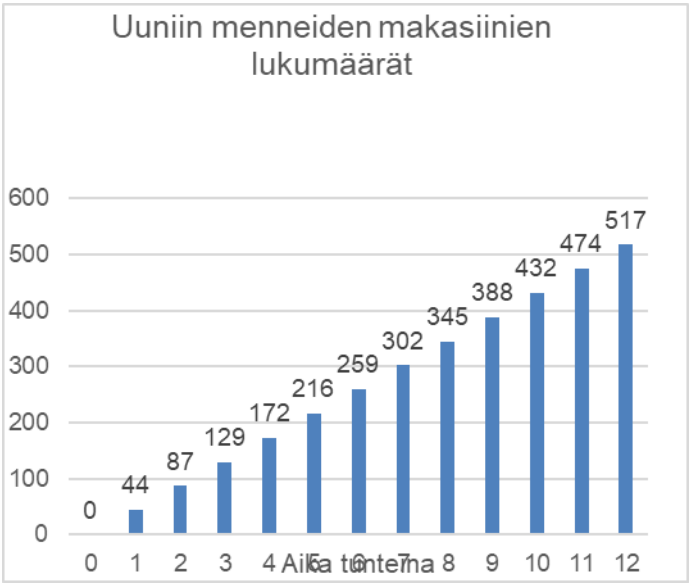
Nykyinen layout



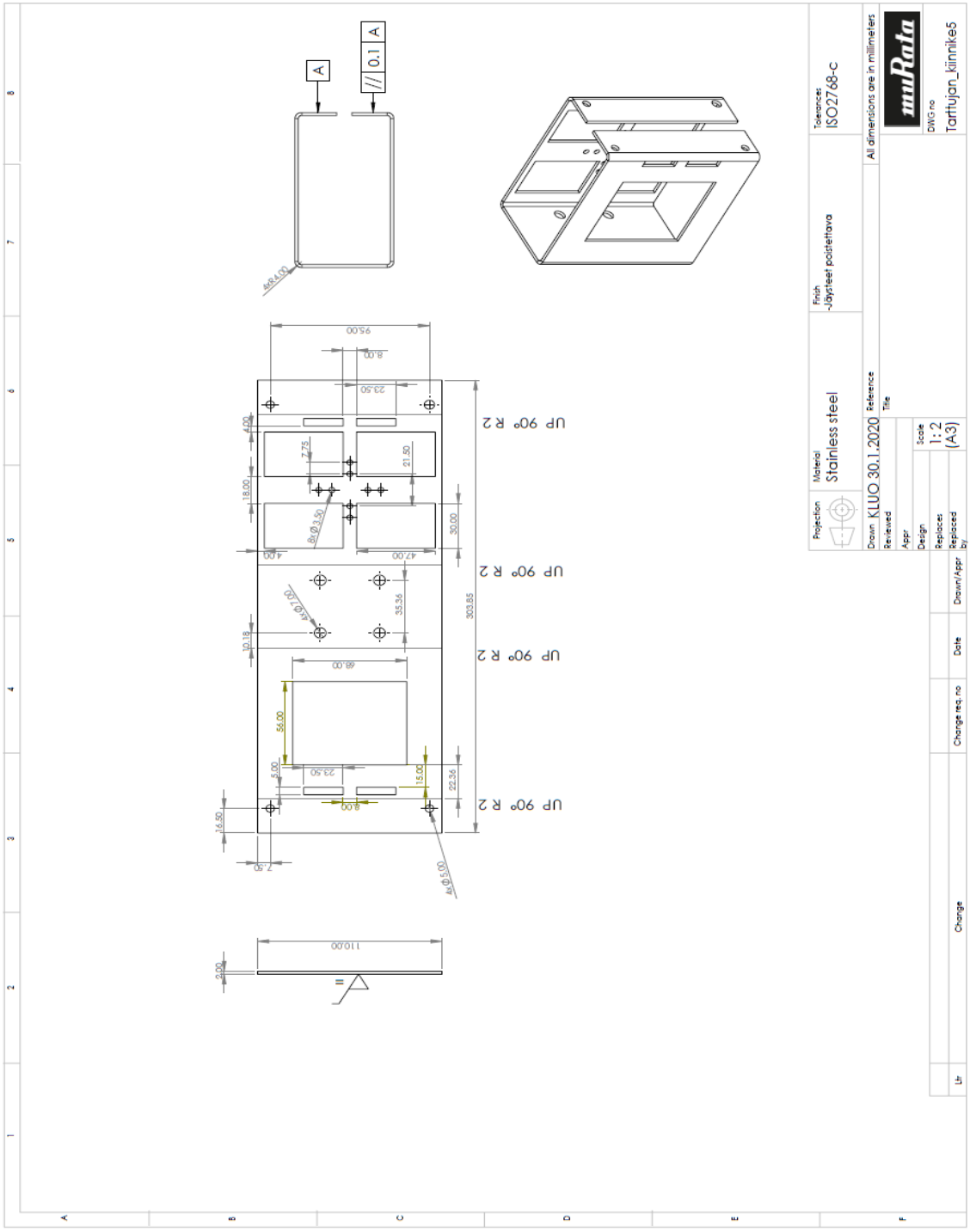
Suunniteltu layout robotin kanssa



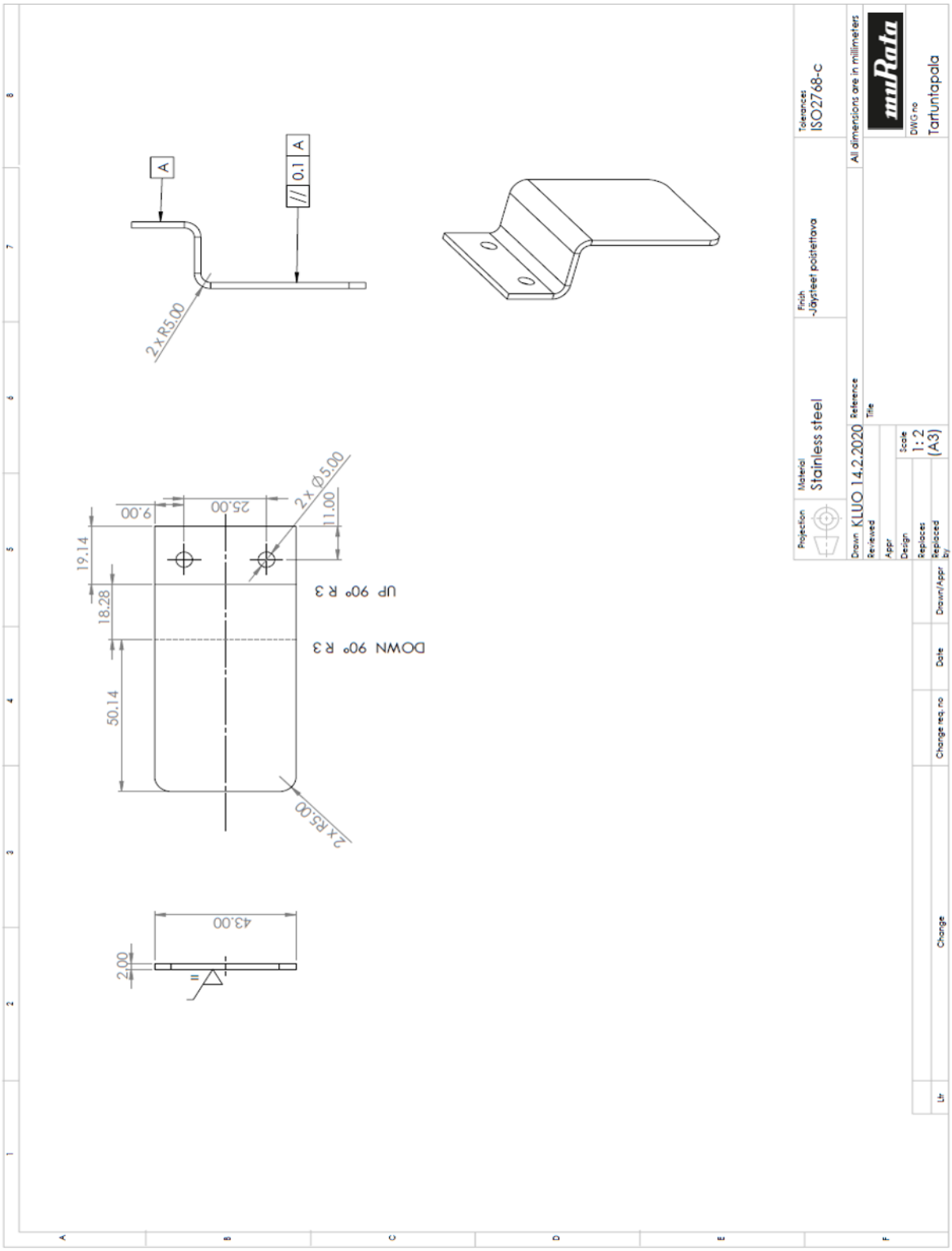




Tarttujan kiinnikkeen mittakuva



Tartuntapalan mittakuva



Työnkierto robotin ohjelmalle

Logiikka			
Robotti			
Viivakoodinlukija			
Toiminto	Kuittausnumerot	Rekisteriosoite johon kirjoitetaan	Ohjelmasekvenssit
Starttia painettu logiikasta	5	input integer register(0)	Makasiini haetaan syöttöasemalta ja viedään hihnakuljettimelle
Aloituspisteessä	0	output integer register(6)	
Lähesty syöttöasemaa	1	input integer register(6)	
Olen lähestymispisteessä	2	output integer register(6)	
Ota tartuntaetäisyys makasiiniiin	3	input integer register(6)	
Olen tartunta etäisyydellä	4	output integer register(6)	
Ota kiinni makasiinista	5	input integer register(6)	
Makasiini tarttujassa	6	output integer register(6)	
Lähesty uunikuljetinta piste1	7	input integer register(6)	
Olen uunikuljettimen lähestymispisteessä1	8	output integer register(6)	
Lähesty uunikuljetinta piste2	9	input integer register(6)	
Olen uunikuljettimen lähestymispisteessä2	10	output integer register(6)	
Laske makasiini uunikuljettimelle	11	input integer register(6)	
makasiini laskettu uunikuljettimelle	12	output integer register(6)	
Päästä irti makasiinista	13	input integer register(6)	
Makasiini laskettu irti	14	output integer register(6)	
Nosta tarttuja ylös kuljettimelta	15	input integer register(6)	
Tarttuja nostettu ylös	16	output integer register(6)	
Palaa alkupisteeseen kuljettimelta	17	input integer register(6)	
Aloituspisteessä	0	output integer register(6)	
Lähesty uunikuljettimen hakupistettä 1	18	input integer register(6)	Uunitettu makasiini viedään jäähylle
Olen uunikuljettimen hakupisteessä1	19	output integer register(6)	
Laske tarttuja alas	20	input integer register(6)	
Tarttuja laskettu alas	21	output integer register(6)	
Ota kiinni makasiinista	22	input integer register(6)	
Makasiini tarttujassa	23	output integer register(6)	
Nosta makasiini ylös	24	input integer register(6)	
makasiini nostettu ylös	25	output integer register(6)	
Lähesty jäähyjigiä piste1	26	input integer register(6)	
Olen jäähyjigin lähestymispisteessä 1	27	output integer register(6)	
Lähesty jäähyjigiä piste2	28	input integer register(6)	
Olen jäähyjigin lähestymispisteessä 2	29	output integer register(6)	
Laske tarttuja alas	30	input integer register(6)	
Tarttuja laskettu alas	31	output integer register(6)	
Päästä irti makasiinista	32	input integer register(6)	
Makasiini laskettu irti	33	output integer register(6)	
Nosta tarttuja ylös jäähyjigiltä	34	input integer register(6)	
Tarttuja nostettu ylös	35	output integer register(6)	
Palaa alkupisteeseen jäähyjigiltä	36	input integer register(6)	
Aloituspisteessä	0	output integer register(6)	

Lähesty jäähyjigiä piste1		37		input integer register(6)	
Olen jäähyjigin lähestymispisteessä 1		38		output integer register(6)	
Laske tarttuja alas		39		input integer register(6)	
Tarttuja laskettu alas		40		output integer register(6)	
Ota kiinni makasiinista		41		input integer register(6)	
Makasiini tarttujassa		42		output integer register(6)	
Nosta makasiini ylös		43		input integer register(6)	
makasiini nostettu ylös		44		output integer register(6)	Jäähtynyt makasiini viedään viivakoodin lukijalle ja sen jälkeen poisvientipisteeseen
Lähesty viivakoodinlukijaa 1		45		input integer register(6)	
Olen viivakoodin lähestymispisteessä1		46		output integer register(6)	
Vie makasiini viivakoodin lukijalle		47		input integer register(6)	
Makasiini viety viivakoodin lukijalle		48		output integer register(6)	
Lue viivakoodi (käsky datamanille)		49		input integer register(6)	
Viivakoodi luettu onnistuneesti		50		output integer register(6)	
nosta tarrtuja ylös		51		input integer register(6)	
tarttuja nostettu ylös		52		output integer register(6)	
Lähesty noutopistettä piste1		53		input integer register(6)	
noutopisteen pisteessä 1		54		output integer register(6)	
Laske makasiini noutopisteelle		55		input integer register(6)	
Makasiini laskettu noutopisteelle		56		output integer register(6)	
Päästä irti makasiinista		57		input integer register(6)	
makasiini laskettu irti		58		output integer register(6)	
Nosta tarttuja ylös		59		input integer register(6)	
Tarttuja ylhäällä		60		output integer register(6)	
Palaa alkupisteeseen		61		input integer register(6)	
alkupisteessä		0		output integer register(6)	

PLC-muuttujat

Totally Integrated Automation Portal		

PLC tags / Default tag table [83]

PLC tags

PLC tags

	Name	Data type	Address	Retain	Accessible from HMI/OPC UA/Web API	Writable from HMI/OPC UA/Web API	Visible in HMI engineering	Supervision	Comment
	FeedPointSensor	Bool	%I1.1	False	True	True	True		Syöttöpisteen anturi
	PickUpPointSensor	Bool	%I1.2	False	True	True	True		Noutopisteen anturi
	CoolJigSensor(1)	Bool	%I1.3	False	True	True	True		Jäähyjigin anturi
	ConveyorSensor	Bool	%I1.4	False	True	True	True		Kuljettimen anturi
	Start/StopKey	Bool	%M0.1	False	True	True	True		Start/Stop apumuuttuja
	PressureMeter	Bool	%I0.0	False	True	True	True		Painemittarin tilatiedot
	Bar_code_cannot_read_block	Bool	%M0.2	False	True	True	True		Viivakoodin luku epäonnistui-datablokki SR-piirille

Totally Integrated Automation Portal		

PLC tags / Dataman_Tags [66]

PLC tags

PLC tags

	Name	Data type	Address	Retain	Accessible from HMI/OPC UA/Web API	Writable from HMI/OPC UA/Web API	Visible in HMI engineering	Supervision	Comment
	DataMan-TriggerEnable	Bool	%Q226.0	False	True	True	True		
	DataMan-Trigger	Bool	%Q226.1	False	True	True	True		
	DataMan-TriggerReady	Bool	%I482.0	False	True	True	True		
	DataMan-TriggerAck	Bool	%I482.1	False	True	True	True		
	DataMan-Acquiring	Bool	%I482.2	False	True	True	True		
	DataMan-MissedAcq	Bool	%I482.3	False	True	True	True		
	DataMan-TriggerID	Word	%IW483	False	True	True	True		
	DataMan-BufferResultsEnable	Bool	%Q227.0	False	True	True	True		
	DataMan-ResultsAck	Bool	%Q227.1	False	True	True	True		
	DataMan-Decoding	Bool	%I485.0	False	True	True	True		
	DataMan-Decode Complete	Bool	%I485.1	False	True	True	True		
	DataMan-Results Buffer Overrun	Bool	%I485.2	False	True	True	True		
	DataMan-Results Available	Bool	%I485.3	False	True	True	True		
	DataMan-General Fault	Bool	%I485.7	False	True	True	True		
	DataMan-Train Code	Bool	%Q228.0	False	True	True	True		
	DataMan-Train Match String	Bool	%Q228.1	False	True	True	True		
	DataMan-Train Focus	Bool	%Q228.2	False	True	True	True		
	DataMan-Train Brightness	Bool	%Q228.3	False	True	True	True		
	DataMan-Untrain	Bool	%Q228.4	False	True	True	True		
	DataMan-Execute DMCC	Bool	%Q228.6	False	True	True	True		
	DataMan-Set Match String	Bool	%Q228.7	False	True	True	True		
	DataMan-Train Code Ack	Bool	%I486.0	False	True	True	True		
	DataMan-Train Match String Ack	Bool	%I486.1	False	True	True	True		
	DataMan-Train Focus Ack	Bool	%I486.2	False	True	True	True		
	DataMan-Train Brightness Ack	Bool	%I486.3	False	True	True	True		
	DataMan-Untrain Ack	Bool	%I486.4	False	True	True	True		
	DataMan-Execute DMCC Ack	Bool	%I486.6	False	True	True	True		

Totally Integrated Automation Portal

	Name	Data type	Address	Retain	Accessible from HMI/OPC UA/Web API	Writable from HMI/OPC UA/Web API	Visible in HMI engineering	Supervision	Comment
	DataMan-Set Match String Ack	Bool	%I486.7	False	True	True	True		
	DataMan-UserData Option	Word	%QW229	False	True	True	True		
	DataMan-UserData Length	Word	%QW231	False	True	True	True		
	DataMan-UserData[0]	Char	%QB233	False	True	True	True		
	DataMan-UserData[1]	Char	%QB234	False	True	True	True		
	DataMan-UserData[2]	Char	%QB235	False	True	True	True		
	DataMan-UserData[3]	Char	%QB236	False	True	True	True		
	DataMan-UserData[4]	Char	%QB237	False	True	True	True		
	DataMan-UserData[5]	Char	%QB238	False	True	True	True		
	DataMan-UserData[6]	Char	%QB239	False	True	True	True		
	DataMan-UserData[7]	Char	%QB240	False	True	True	True		
	DataMan-UserData[8]	Char	%QB241	False	True	True	True		
	DataMan-UserData[9]	Char	%QB242	False	True	True	True		
	DataMan-UserData[10]	Char	%QB243	False	True	True	True		
	DataMan-UserData[11]	Char	%QB244	False	True	True	True		
	DataMan-UserData[12]	Char	%QB245	False	True	True	True		
	DataMan-UserData[13]	Char	%QB246	False	True	True	True		
	DataMan-UserData[14]	Char	%QB247	False	True	True	True		
	DataMan-UserData[15]	Char	%QB248	False	True	True	True		
	DataMan-ResultID	Char	%IB487	False	True	True	True		
	DataMan-Result Code	Char	%IB489	False	True	True	True		
	DataMan-Result Extended	Char	%IB491	False	True	True	True		
	DataMan-Result Length	Char	%IB493	False	True	True	True		
	DataMan-InspectionResults[0]	Char	%IB495	False	True	True	True		
	DataMan-InspectionResults[1]	Char	%IB496	False	True	True	True		
	DataMan-InspectionResults[2]	Char	%IB497	False	True	True	True		
	DataMan-InspectionResults[3]	Char	%IB498	False	True	True	True		
	DataMan-InspectionResults[4]	Char	%IB499	False	True	True	True		
	DataMan-InspectionResults[5]	Char	%IB500	False	True	True	True		
	DataMan-InspectionResults[6]	Char	%IB501	False	True	True	True		

Totally Integrated Automation Portal

	Name	Data type	Address	Retain	Accessible from HMI/OPC UA/Web API	Writable from HMI/OPC UA/Web API	Visible in HMI engineering	Supervision	Comment
	DataMan-Inspection-Results[7]	Char	%IB502	False	True	True	True		
	DataMan-Inspection-Results[8]	Char	%IB503	False	True	True	True		
	DataMan-Inspection-Results[9]	Char	%IB504	False	True	True	True		
	DataMan-Inspection-Results[10]	Char	%IB505	False	True	True	True		
	DataMan-Inspection-Results[11]	Char	%IB506	False	True	True	True		
	DataMan-Inspection-Results[12]	Char	%IB507	False	True	True	True		
	DataMan-Inspection-Results[13]	Char	%IB508	False	True	True	True		
	DataMan-Inspection-Results[14]	Char	%IB509	False	True	True	True		
	DataMan-Inspection-Results[15]	Char	%IB510	False	True	True	True		

Totally Integrated Automation Portal

PLC tags / RoboTags [10]

PLC tags

PLC tags

	Name	Data type	Address	Retain	Accessible from HMI/OPC UA/Web API	Writable from HMI/OPC UA/Web API	Visible in HMI engineering	Supervision	Comment
	T20_State	"UR_1_T20_State"	%I2.0	False	True	True	True		
	Robot	UR__T20_Robot	%I2.0		Accessible	Accessible	Accessible		
	Controller major version	USInt	%IB2		Accessible	Accessible	Accessible		
	Controller minor version	USInt	%IB3		Accessible	Accessible	Accessible		
	Reserved	UInt	%IW4		None	None	None		
	Robot mode	USInt	%IB6		Accessible	Accessible	Accessible		
	Real time machine seconds	USInt	%IB7		Accessible	Accessible	Accessible		
	Real time machine milliseconds	UInt	%IW8		Accessible	Accessible	Accessible		
	Real time machine minutes	USInt	%IB10		Accessible	Accessible	Accessible		
	Real time machine hours	USInt	%IB11		Accessible	Accessible	Accessible		
	Real time machine days	UInt	%IW12		Accessible	Accessible	Accessible		
	Robot current [A]	Real	%ID14		Accessible	Accessible	Accessible		
	PW: Is power on	Bool	%I18.0		Accessible	Accessible	Accessible		
	PR: Is program running	Bool	%I18.1		Accessible	Accessible	Accessible		
	TB: Is teach button pressed	Bool	%I18.2		Accessible	Accessible	Accessible		
	PB: Is power button pressed	Bool	%I18.3		Accessible	Accessible	Accessible		
	Reserved_4	Bool	%I18.4		None	None	None		
	Reserved_5	Bool	%I18.5		None	None	None		
	Reserved_6	Bool	%I18.6		None	None	None		
	Reserved_7	Bool	%I18.7		None	None	None		
	Reserved_8_15	USInt	%IB19		None	None	None		
	Reserved_16_31	UInt	%IW20		None	None	None		
	Speed slider fraction	Real	%ID22		Accessible	Accessible	Accessible		
	Safety	UR__T20_Safety	%I26.0		Accessible	Accessible	Accessible		
	Safety mode	USInt	%IB26		Accessible	Accessible	Accessible		

	T20_General_Purpose_Int_Register	"UR_6_T2O_IntRegisters"	%I290.0	False	True	True	True		
	Register	Array[0..23] of DInt	%I290.0		Accessible	Accessible	Accessible		
	Register[0]	DInt	%ID290		Accessible	Accessible	Accessible		
	Register[1]	DInt	%ID294		Accessible	Accessible	Accessible		
	Register[2]	DInt	%ID298		Accessible	Accessible	Accessible		
	Register[3]	DInt	%ID302		Accessible	Accessible	Accessible		
	Register[4]	DInt	%ID306		Accessible	Accessible	Accessible		
	Register[5]	DInt	%ID310		Accessible	Accessible	Accessible		
	Register[6]	DInt	%ID314		Accessible	Accessible	Accessible		
	Register[7]	DInt	%ID318		Accessible	Accessible	Accessible		
	Register[8]	DInt	%ID322		Accessible	Accessible	Accessible		
	Register[9]	DInt	%ID326		Accessible	Accessible	Accessible		
	Register[10]	DInt	%ID330		Accessible	Accessible	Accessible		
	Register[11]	DInt	%ID334		Accessible	Accessible	Accessible		
	Register[12]	DInt	%ID338		Accessible	Accessible	Accessible		
	Register[13]	DInt	%ID342		Accessible	Accessible	Accessible		
	Register[14]	DInt	%ID346		Accessible	Accessible	Accessible		

	02T_General_Purpose_Register1	"UR_9_O2T_Registers"	%Q26.0	False	True	True	True		
	Bits	UR__O2T_bits	%Q26.0		Accessible	Accessible	Accessible		
	Register	Array[0..31] of Bool	%Q26.0		Accessible	Accessible	Accessible		
	Register[0]	Bool	%Q26.0		Accessible	Accessible	Accessible		
	Register[1]	Bool	%Q26.1		Accessible	Accessible	Accessible		
	Register[2]	Bool	%Q26.2		Accessible	Accessible	Accessible		
	Register[3]	Bool	%Q26.3		Accessible	Accessible	Accessible		
	Register[4]	Bool	%Q26.4		Accessible	Accessible	Accessible		
	Register[5]	Bool	%Q26.5		Accessible	Accessible	Accessible		
	Register[6]	Bool	%Q26.6		Accessible	Accessible	Accessible		
	Register[7]	Bool	%Q26.7		Accessible	Accessible	Accessible		
	Register[8]	Bool	%Q27.0		Accessible	Accessible	Accessible		
	Register[9]	Bool	%Q27.1		Accessible	Accessible	Accessible		
	Register[10]	Bool	%Q27.2		Accessible	Accessible	Accessible		
	Register[11]	Bool	%Q27.3		Accessible	Accessible	Accessible		
	Register[12]	Bool	%Q27.4		Accessible	Accessible	Accessible		
	Register[13]	Bool	%Q27.5		Accessible	Accessible	Accessible		
	Register[14]	Bool	%Q27.6		Accessible	Accessible	Accessible		
	Register[15]	Bool	%Q27.7		Accessible	Accessible	Accessible		
	Register[16]	Bool	%Q28.0		Accessible	Accessible	Accessible		
	Register[17]	Bool	%Q28.1		Accessible	Accessible	Accessible		

					sible	sible	sible		
	Ints	UR__O2T_ints	%Q130.0		Acces-sible	Acces-sible	Acces-sible		
	Register	Array[0..11] of DInt	%Q130.0		Acces-sible	Acces-sible	Acces-sible		
	Register[0]	DInt	%QD130		Acces-sible	Acces-sible	Acces-sible		
	Register[1]	DInt	%QD134		Acces-sible	Acces-sible	Acces-sible		

Totally Integrated Automation Portal		

	Name	Data type	Address	Retain	Acces-sible from HMI/OPC UA/Web API	Writa-ble from HMI/OPC UA/Web API	Visi-ble in HMI engi-neer-ing	Supervision	Comment
	Register[2]	DInt	%QD138		Acces-sible	Acces-sible	Acces-sible		
	Register[3]	DInt	%QD142		Acces-sible	Acces-sible	Acces-sible		
	Register[4]	DInt	%QD146		Acces-sible	Acces-sible	Acces-sible		
	Register[5]	DInt	%QD150		Acces-sible	Acces-sible	Acces-sible		
	Register[6]	DInt	%QD154		Acces-sible	Acces-sible	Acces-sible		
	Register[7]	DInt	%QD158		Acces-sible	Acces-sible	Acces-sible		
	Register[8]	DInt	%QD162		Acces-sible	Acces-sible	Acces-sible		
	Register[9]	DInt	%QD166		Acces-sible	Acces-sible	Acces-sible		
	Register[10]	DInt	%QD170		Acces-sible	Acces-sible	Acces-sible		
	Register[11]	DInt	%QD174		Acces-sible	Acces-sible	Acces-sible		

Logiikan ohjelmakoodi

Program blocks

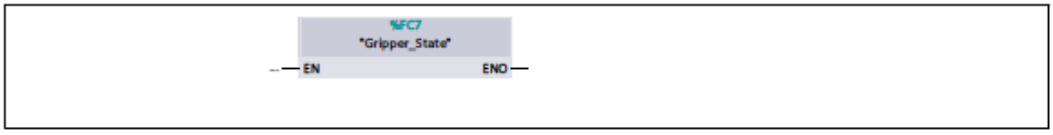
Main [OB1]

Main Properties					
General					
Name	Main	**Number**	1	**Type**	OB
Language	FBD	**Numbering**	Automatic		
Information					
Title	"Main Program Sweep (Cycle)"	**Author**		**Comment**	Pääohjelma ja siinä olevat aliohjelmakutsut
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Comment
Input			
Initial_Call	Bool		Initial call of this OB
Remanence	Bool		=True, if remanent data are available
Temp			
Constant			

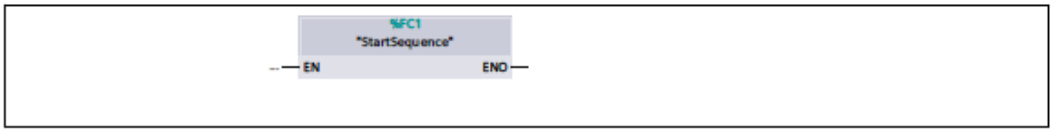
Network 1:

Katsotaan painemittarin tilatiedot



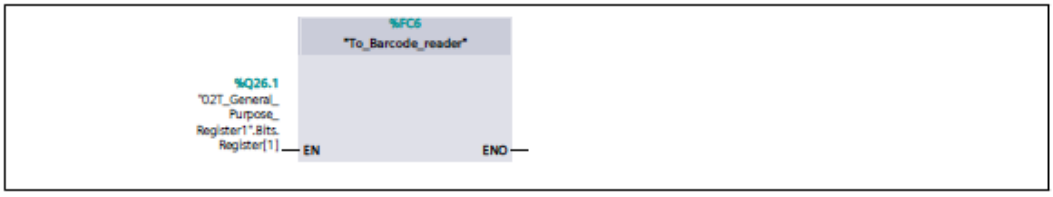
Network 2:

Aloitussekvenssi joka katsoo ohjelman olevan käynnissä robotilla ja antaa robotille luvan ohjelman suorittamiseen



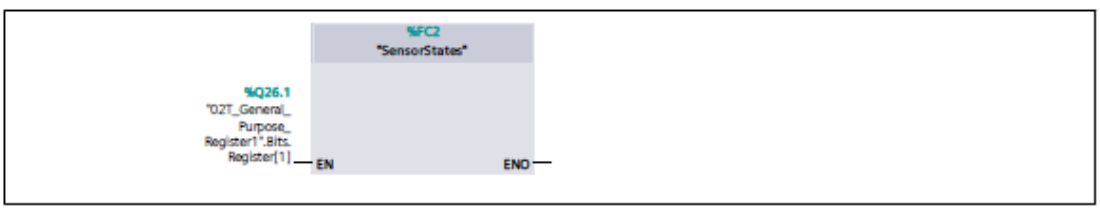
Network 3:

Aliohjelma robotin ohjaamiseen viivakoodinlukijan luo



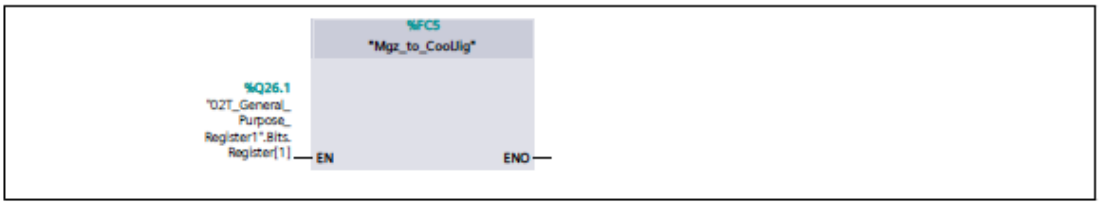
Network 4:

Sensorien tilatiedot, jotka kirjoitetaan integer-rekistereihin



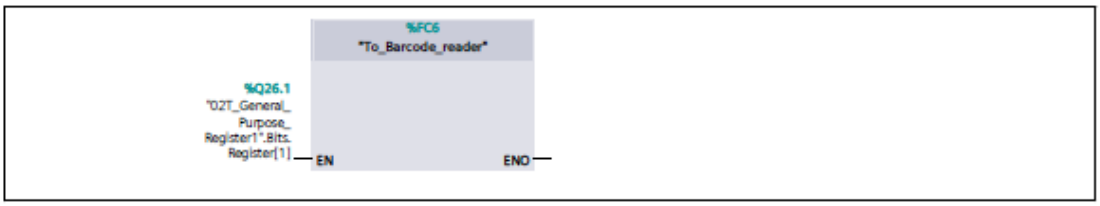
Network 5:

Aliohjelma makasiinin viemiselle jäähyjigin luo



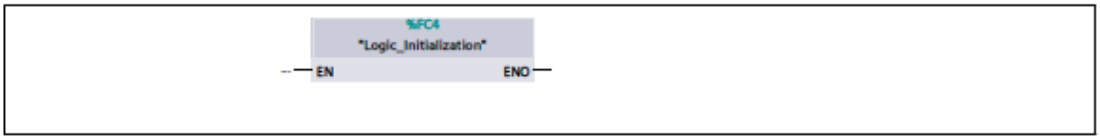
Network 6:

Aliohjelma robotin ohjaamiseen viivakoodinlukijan luo



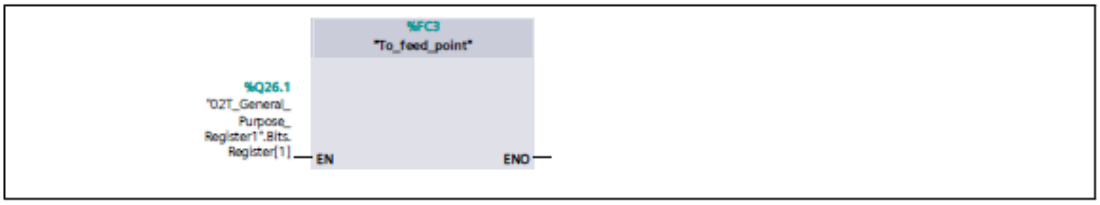
Network 7:

Logiikan initalisointi sekvenssi, jonka ehdot löytyvät blockin sisältä. Logiikalle kirjoitetaan arvo 0.



Network 8:

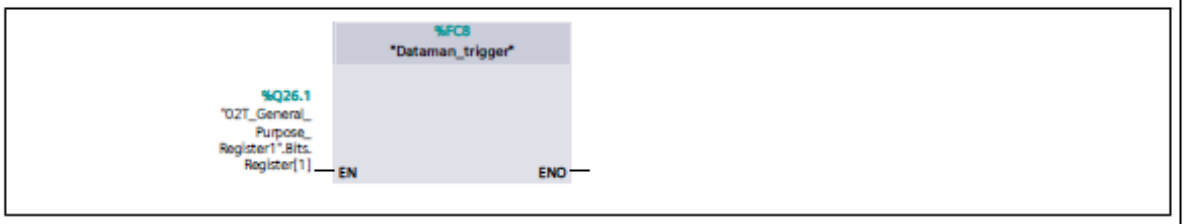
Robotti ohjataan hakemaan makasiini syöttöpisteeltä ja viemään se hihnakuljettimelle



Network 9:

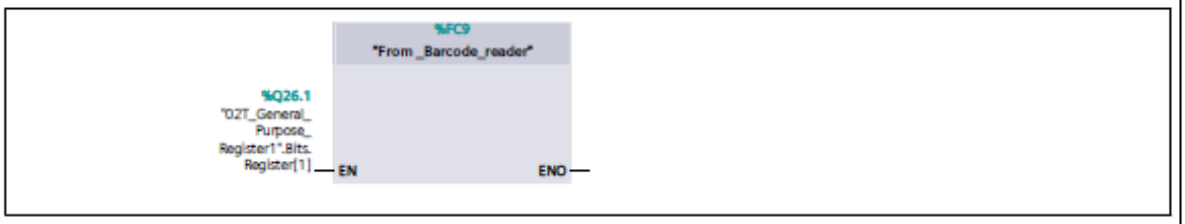
Viivakoodin lukijan triggaussekvenssi, joka laukaisee viivakoodinlukijan.

Totally Integrated Automation Portal		



Network 10:

Aliohjelma joka ohjaa robotin viivakoodinlukijalta makasiinin noutopisteeseen



Program blocks

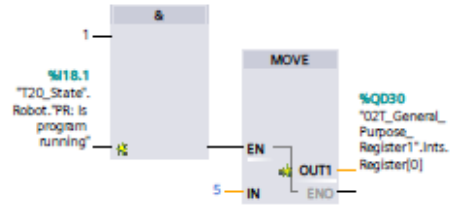
StartSequence [FC1]

StartSequence Properties					
General					
Name	StartSequence	**Number**	1	**Type**	FC
Language	FBD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	Aloitussekvenssi joka katsoo ohjelman olevan käynnissä robotilla ja antaa robotille luvan ohjelman suorittamiseen
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Comment
Input			
Output			
InOut			
Temp			
Constant			
▼ Return			
StartSequence	Void		

Network 1:

Jos robotin ohjelma pyörii, niin aloitusehto kirjoitetaan numeroksi 5 ja robotti lähtee suorittamaan ohjelmaa.



Network 2:

Jos robotin ohjelma ei pyöri. logiikka kirjoittaa arvon 0 integer rekisteriin eli initalisoi itsensä

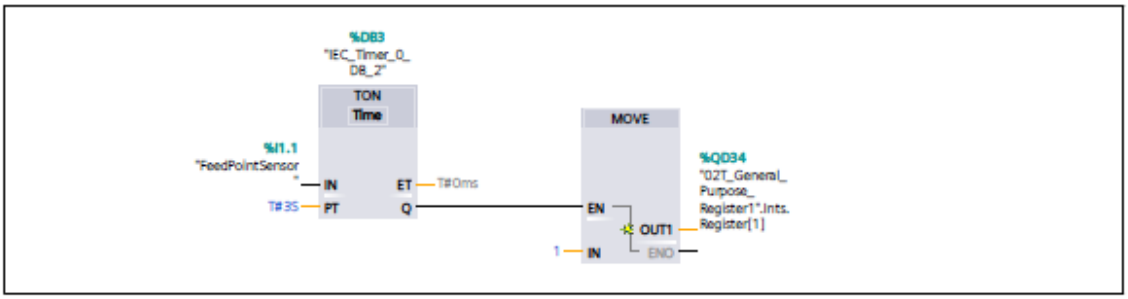
SensorStates [FC2]

SensorStates Properties					
General					
Name	SensorStates	**Number**	2	**Type**	FC
Language	FBD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	Sensorien tilatiedot, jotka kirjoitetaan integer-rekis-tereihin
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Comment
Input			
Output			
InOut			
Temp			
Constant			
▼ Return			
SensorStates	Void		

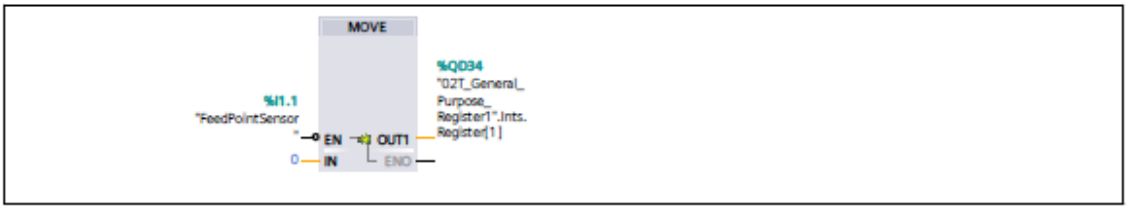
Network 1:

Syöttöaseman anturin ollessa true tilassa 3 sekuntia, kirjoitetaan integer-rekisteriin arvo 1.



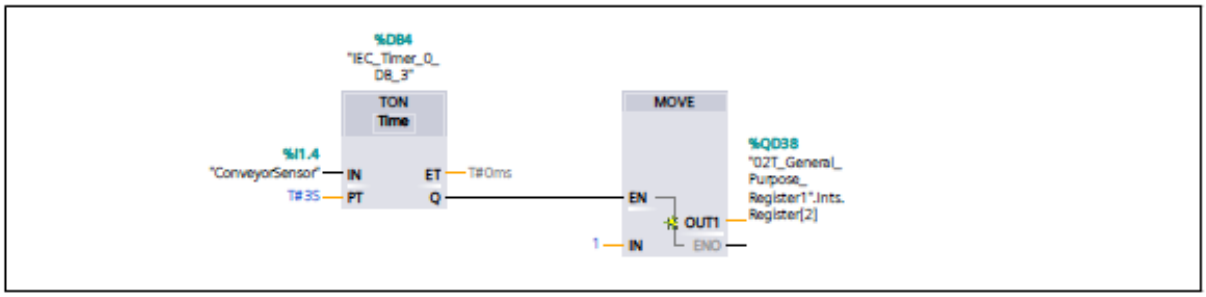
Network 2:

Syöttöaseman anturin ollessa false-tilassa kirjoitetaan välittömästi arvo 0 integer rekisteriin



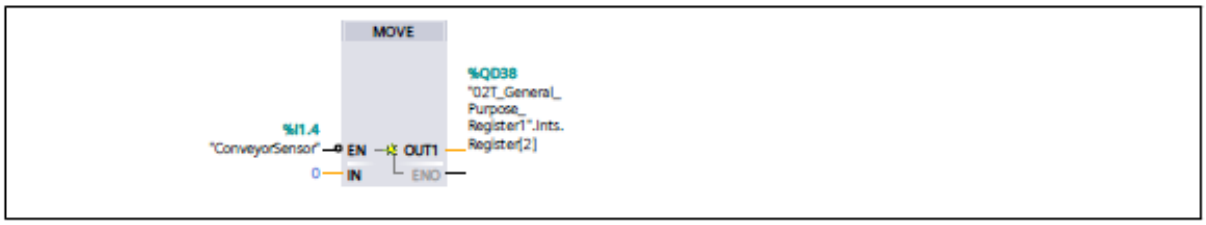
Network 3:

Hihnakuljettimen anturin ollessa true tilassa 3 sekuntia, kirjoitetaan integer-rekisteriin arvo 1.



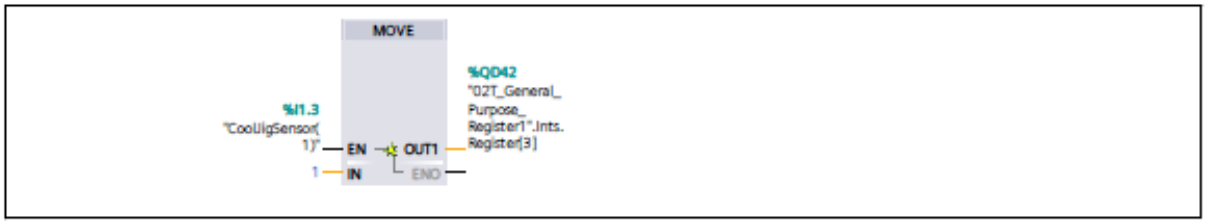
Network 4:

Hihnakuljettimen anturin ollessa false-tilassa kirjoitetaan välittömästi arvo 0 integer rekisteriin



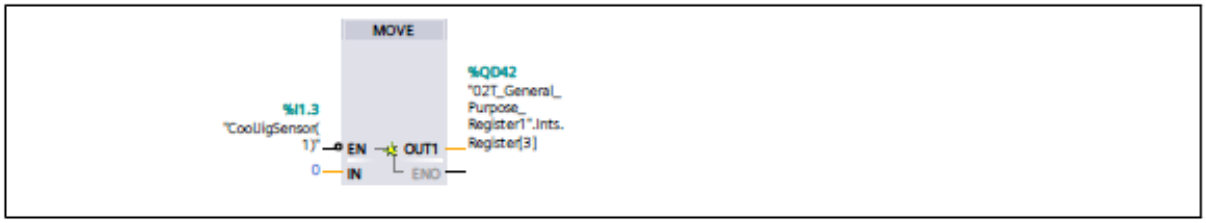
Network 5:

Jäähyjigin anturin ollessa true tilassa, kirjoitetaan välittömästi integer-rekisteriin arvo 1. Viivettä ei tässä tarvita koska se on jo laitettu toiseen aliohjelmaan (jäähyaika).



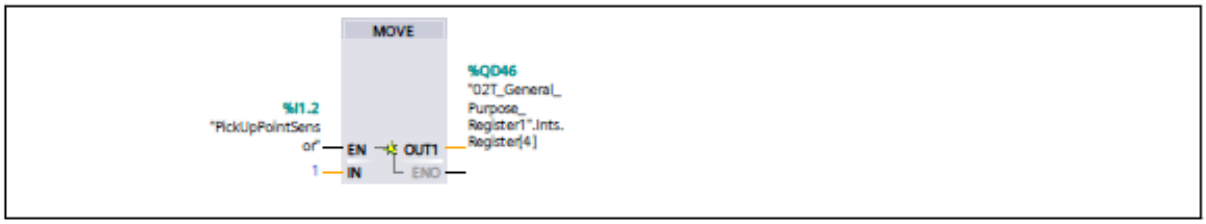
Network 6:

Jäähyjigin anturin ollessa false-tilassa kirjoitetaan välittömästi arvo 0 integer rekisteriin



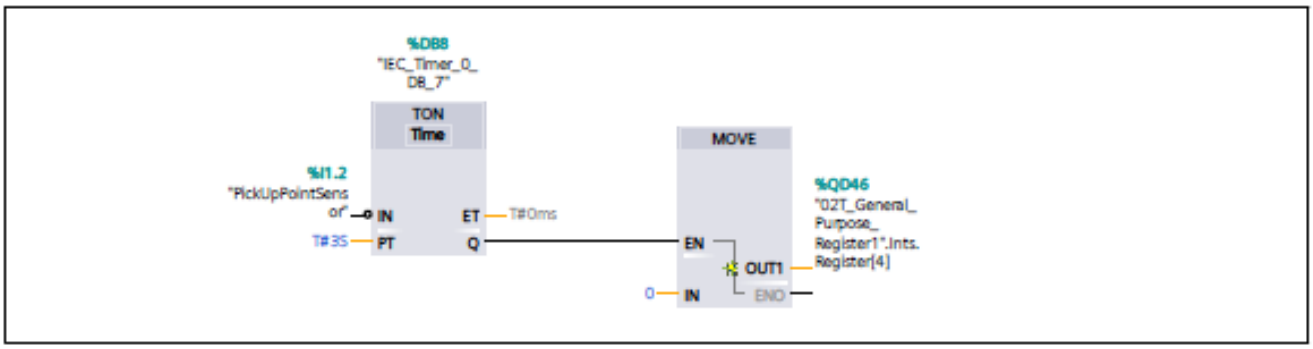
Network 7:

Noutopisteen anturin ollessa true-tilassa kirjoitetaan välittömästi arvo 1 integer rekisteriin



Network 8:

Noutopisteen anturin ollessa false tilassa 3 sekuntia, kirjoitetaan integer-rekisteriin arvo 0. tämä siksi jos makasiini kaatuu, niin robotti ei heti reagoi false tilaan.



Program blocks

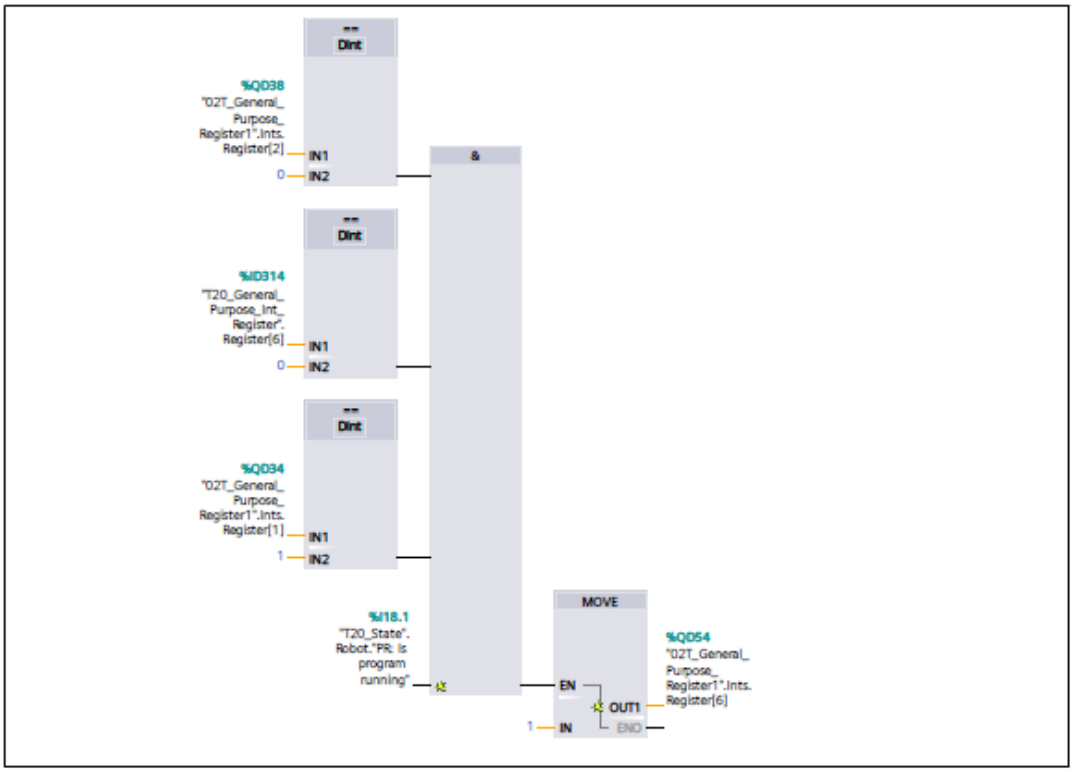
To_feed_point [FC3]

To_feed_point Properties					
General					
Name	To_feed_point	**Number**	3	**Type**	FC
Language	FBD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	Robotti ohjataan hakemaan makasiini syöttöpisteeltä ja viemään se hihnakuljettimelle
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Comment
Input			
Output			
InOut			
Temp			
Constant			
▼ Return			
To_feed_point	Void		

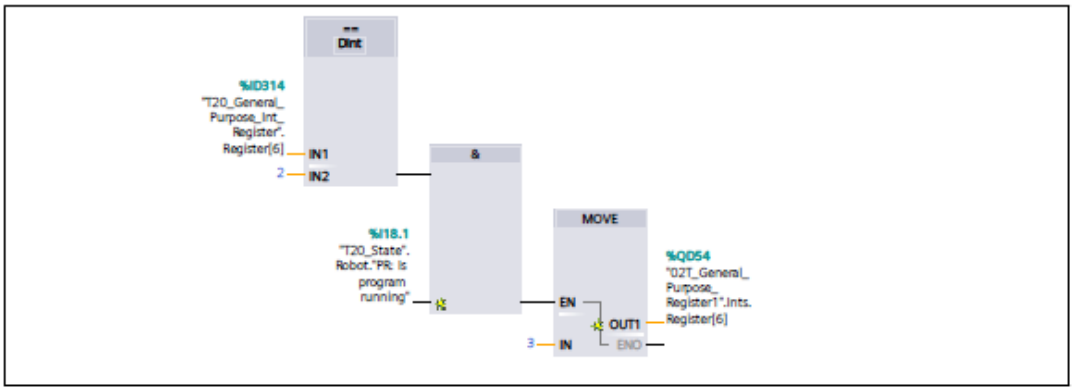
Network 1:

Logiikka kirjoittaa integer rekisteriin arvon 1 jos hihnakuljettimen arvo 0 ja robotilla on arvo nolla ja syöttöaseman anturi tunnistaa ja robotin ohjelma pyörii.



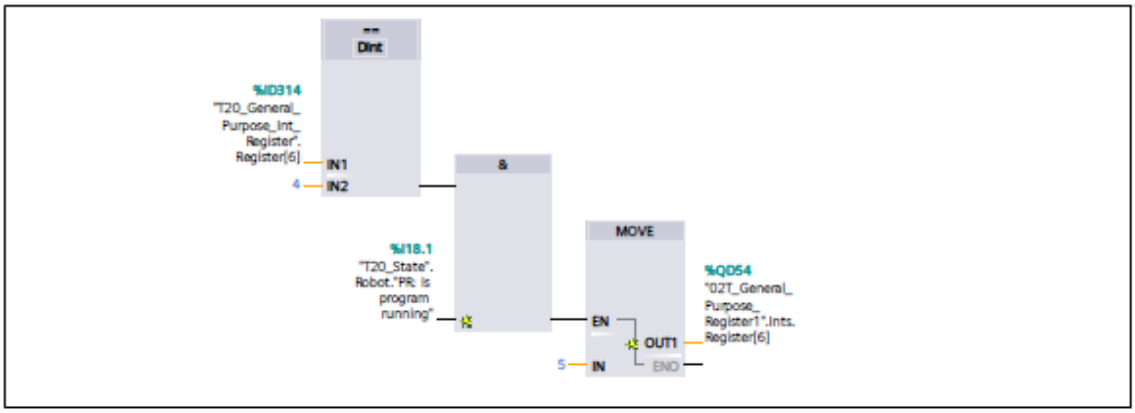
Network 2:

Logiikka kirjoittaa integer rekisteriin arvon 3 jos robotilla on arvo 2 ja robotin ohjelma pyörii.



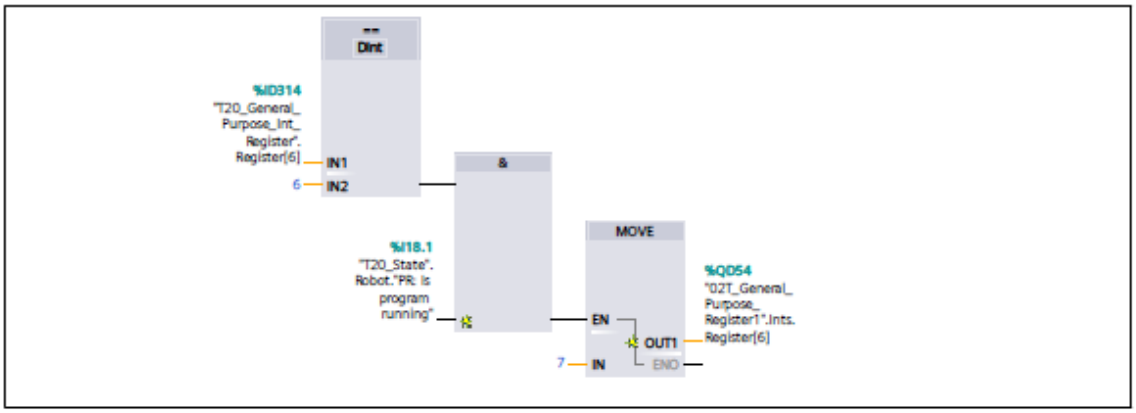
Network 3:

Logiikka kirjoittaa integer rekisteriin arvon 5 jos robotilla on arvo 4 ja robotin ohjelma pyörii.



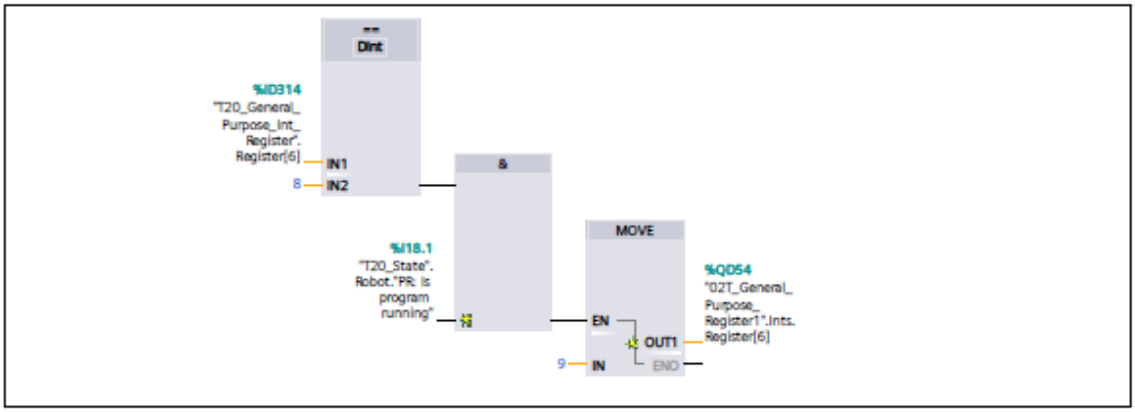
Network 4:

Logiikka kirjoittaa integer rekisteriin arvon 7 jos robotilla on arvo 6 ja robotin ohjelma pyörii.



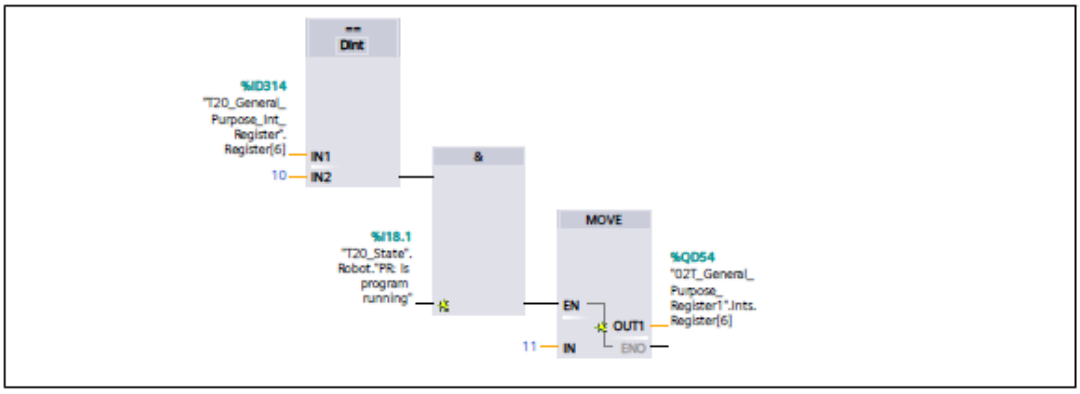
Network 5:

Logiikka kirjoittaa integer rekisteriin arvon 9 jos robotilla on arvo 8 ja robotin ohjelma pyörii.



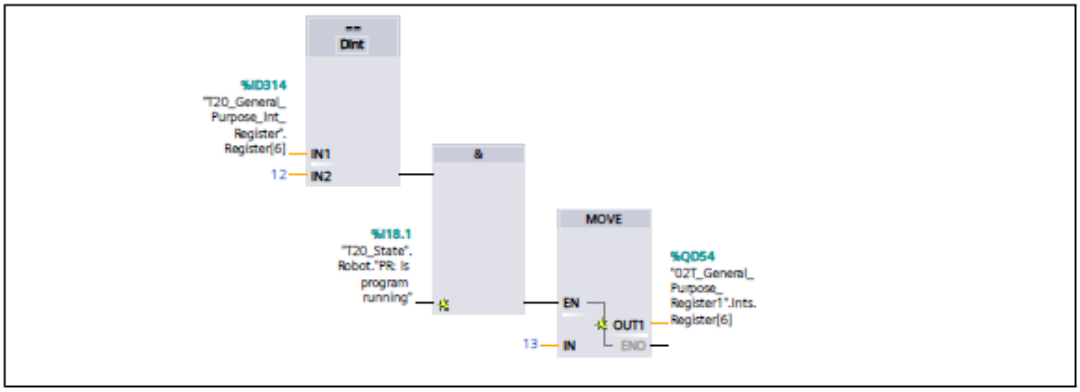
Network 6:

Logiikka kirjoittaa integer rekisteriin arvon 11 jos robotilla on arvo 10 ja robotin ohjelma pyörii.



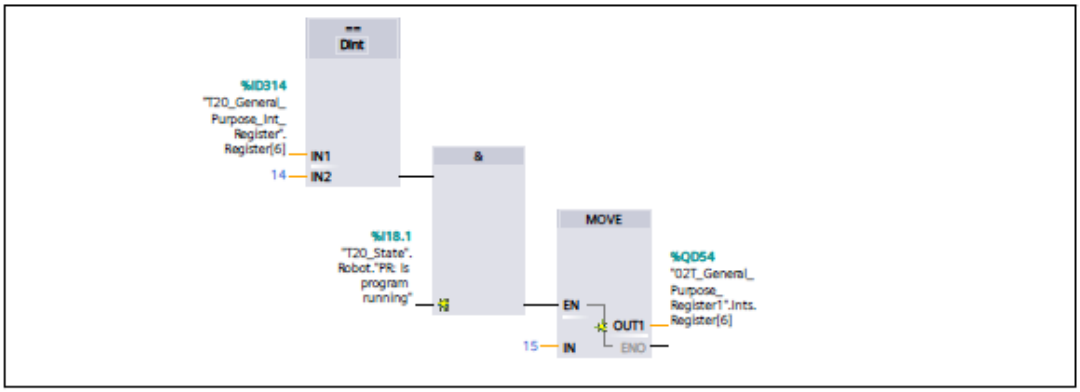
Network 7:

Logiikka kirjoittaa integer rekisteriin arvon 13 jos robotilla on arvo 12 ja robotin ohjelma pyörii.



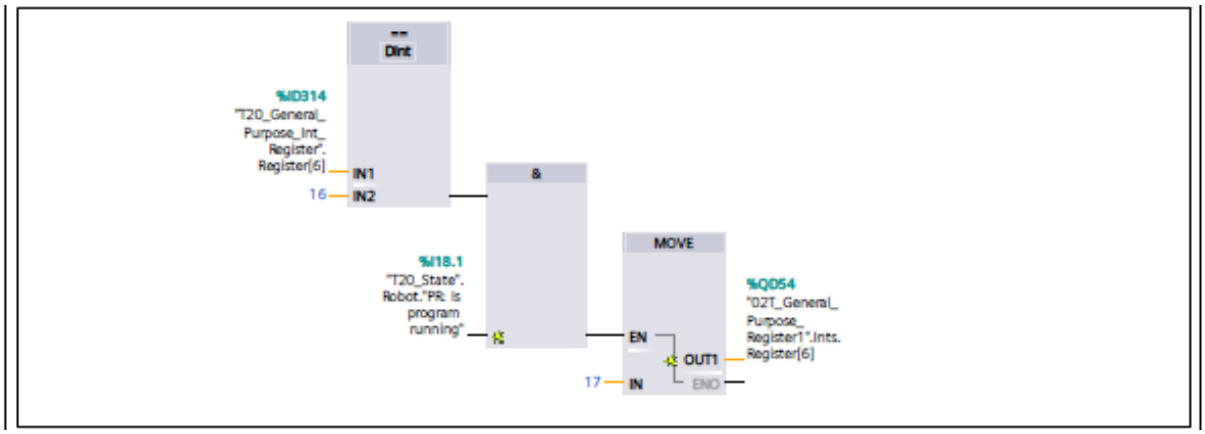
Network 8:

Logiikka kirjoittaa integer rekisteriin arvon 15 jos robotilla on arvo 14 ja robotin ohjelma pyörii.



Network 9:

Logiikka kirjoittaa integer rekisteriin arvon 17 jos robotilla on arvo 16 ja robotin ohjelma pyörii.


==
Dint
%ID314
"T20_General_
Purpose_Int_
Register".
Register[6]
IN1
16
IN2
&
%I18.1
"T20_State".
Robot."PR: Is
program
running"
MOVE
EN
OUT1
17
IN
ENO
%Q054
"02T_General_
Purpose_
Register1".Ints.
Register[6]

Program blocks

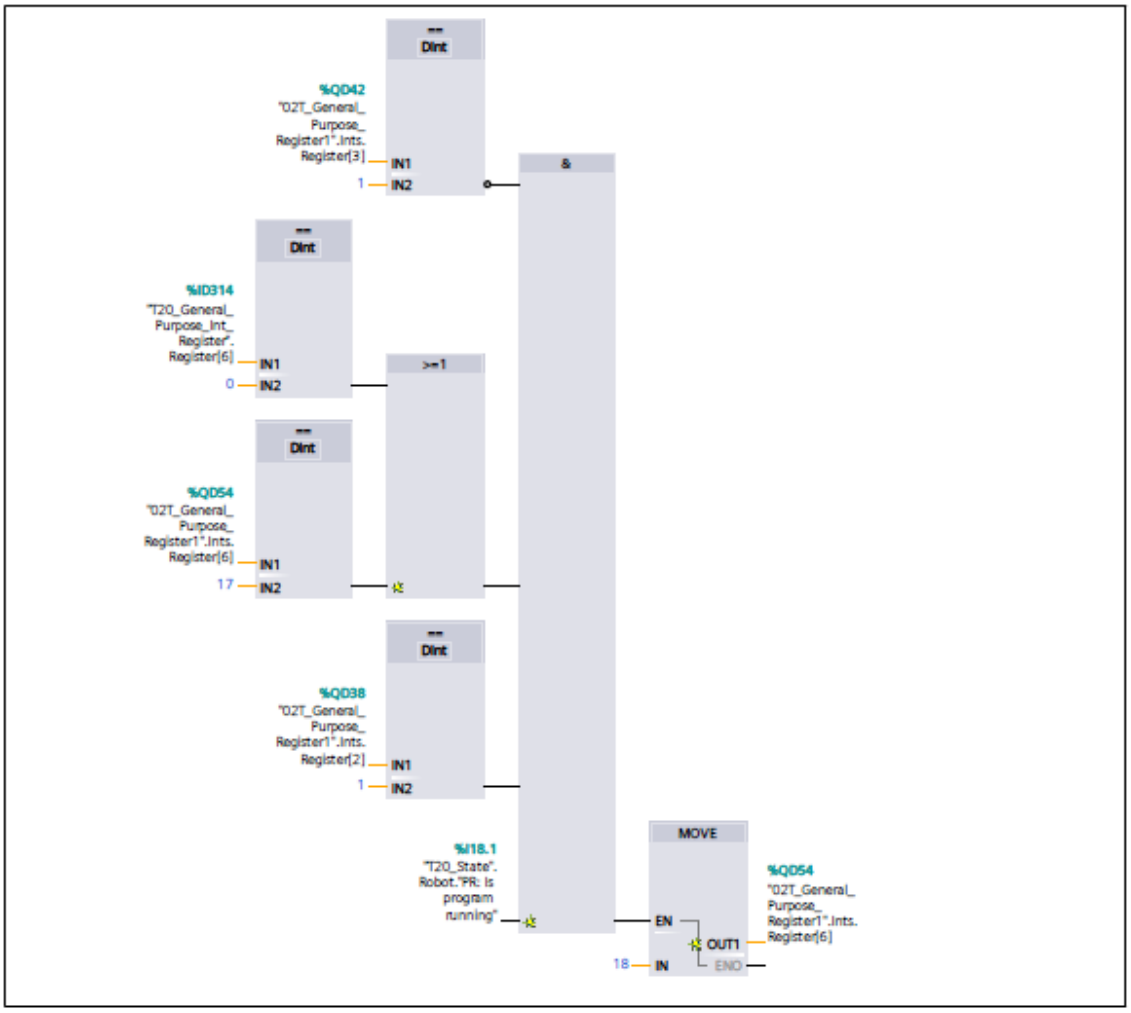
Mgz_to_CoolJig [FC5]

Mgz_to_CoolJig Properties					
General					
Name	Mgz_to_CoolJig	**Number**	5	**Type**	FC
Language	FBD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	Aliohjelma makasiinin viemiselle jäähyjigin luo
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Comment
Input			
Output			
InOut			
Temp			
Constant			
▼ Return			
Mgz_to_CoolJig	Void		

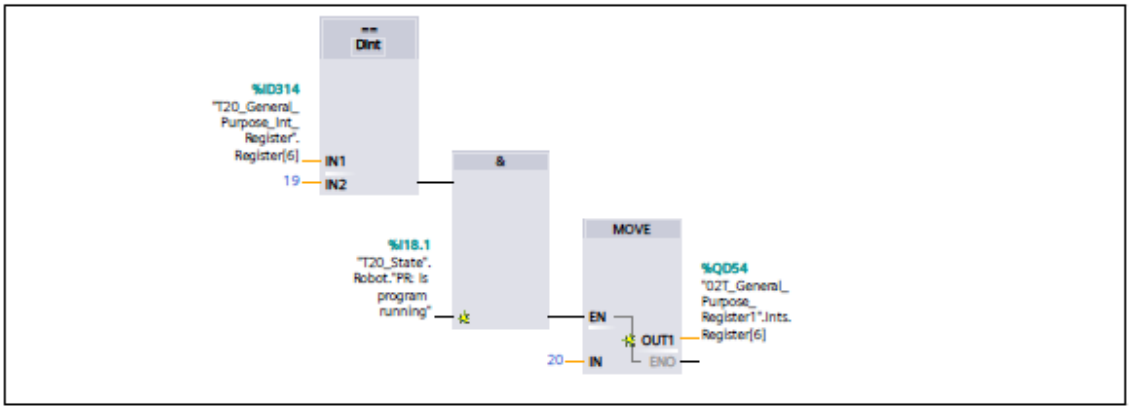
Network 1:

Robotti kirjoittaa integer-rekisteriin arvon nolla jos jäähyjigin anturi ei tunnista ja hihnakuljettimen anturi tunnistaa ja robotin ohjelma pyörii. Lisäehtoina on myös että robotin integer-rekisterissä pitää olla 0 tai 17.



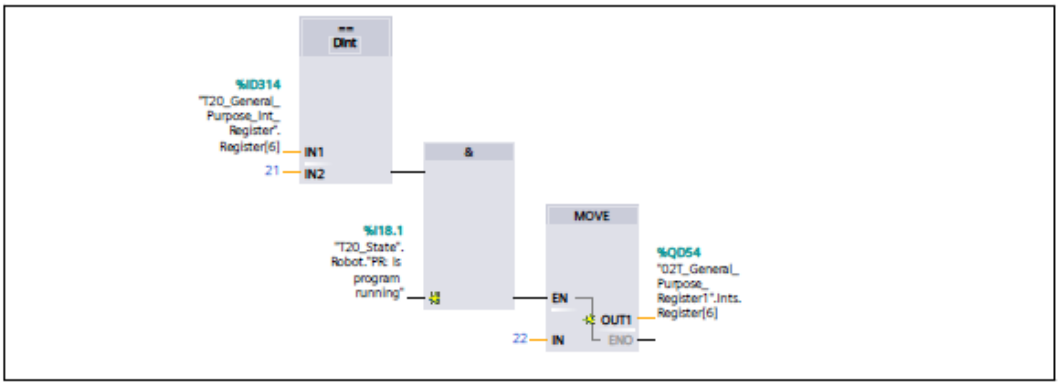
Network 2:

Logiikka kirjoittaa integer rekisteriin arvon 20 jos robotilla on arvo 19 ja robotin ohjelma pyörii.



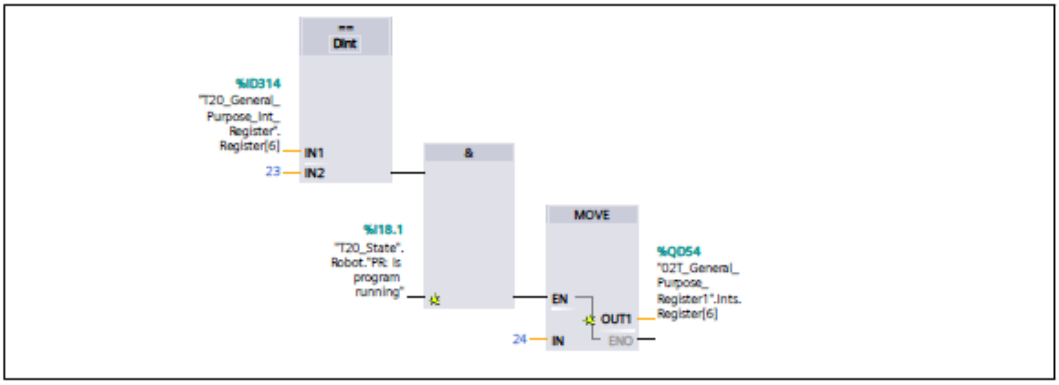
Network 3:

Logiikka kirjoittaa integer rekisteriin arvon 22 jos robotilla on arvo 21 ja robotin ohjelma pyörii.



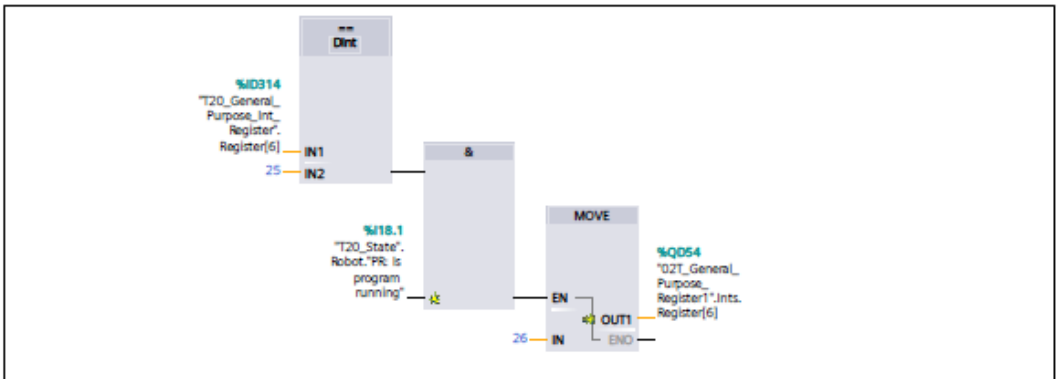
Network 4:

Logiikka kirjoittaa integer rekisteriin arvon 24 jos robotilla on arvo 23 ja robotin ohjelma pyörii.



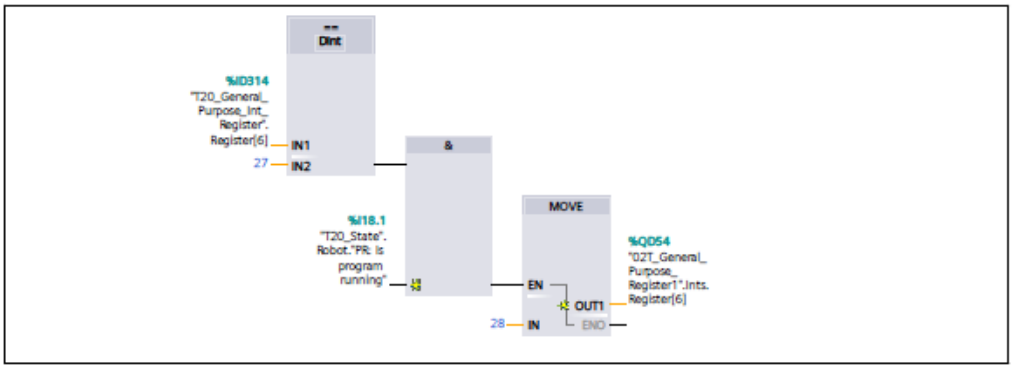
Network 5:

Logiikka kirjoittaa integer rekisteriin arvon 26 jos robotilla on arvo 25 ja robotin ohjelma pyörii.



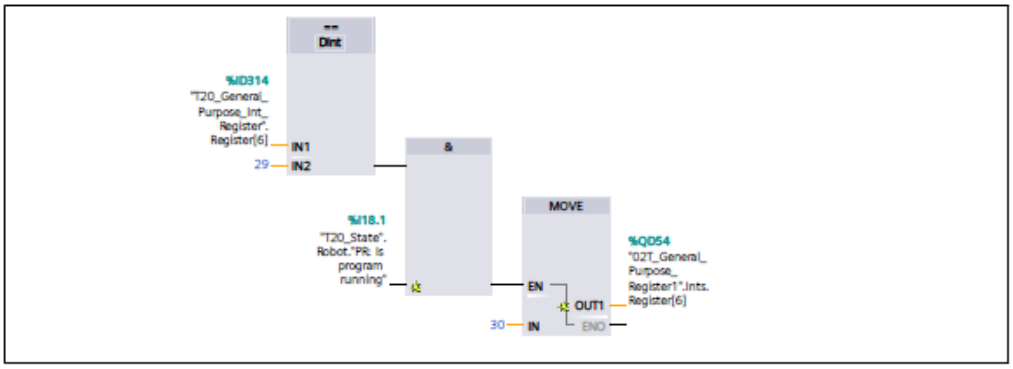
Network 6:

Logiikka kirjoittaa integer rekisteriin arvon 28 jos robotilla on arvo 27 ja robotin ohjelma pyörii.



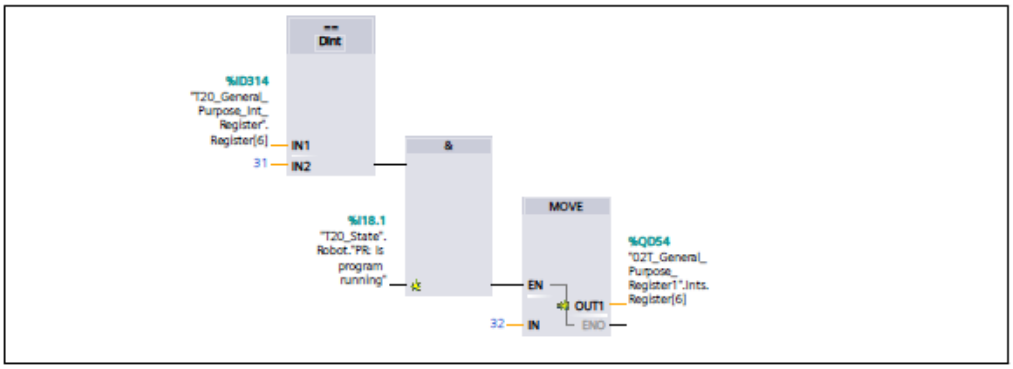
Network 7:

Logiikka kirjoittaa integer rekisteriin arvon 29 jos robotilla on arvo 30 ja robotin ohjelma pyörii.



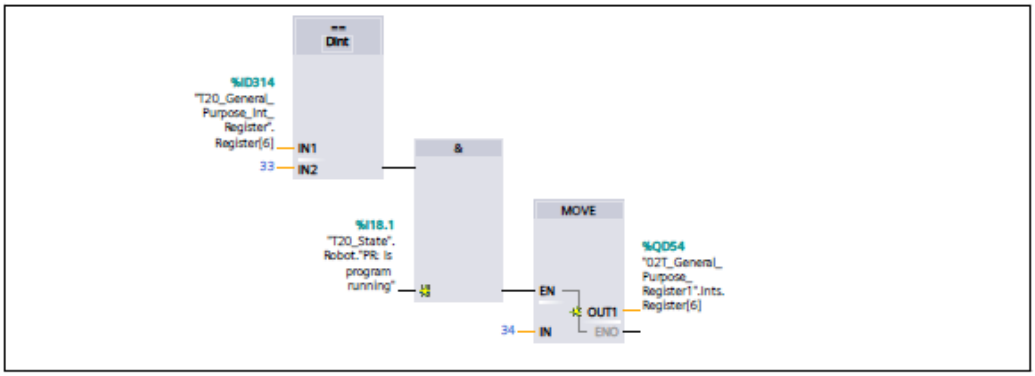
Network 8:

Logiikka kirjoittaa integer rekisteriin arvon 32 jos robotilla on arvo 31 ja robotin ohjelma pyörii.



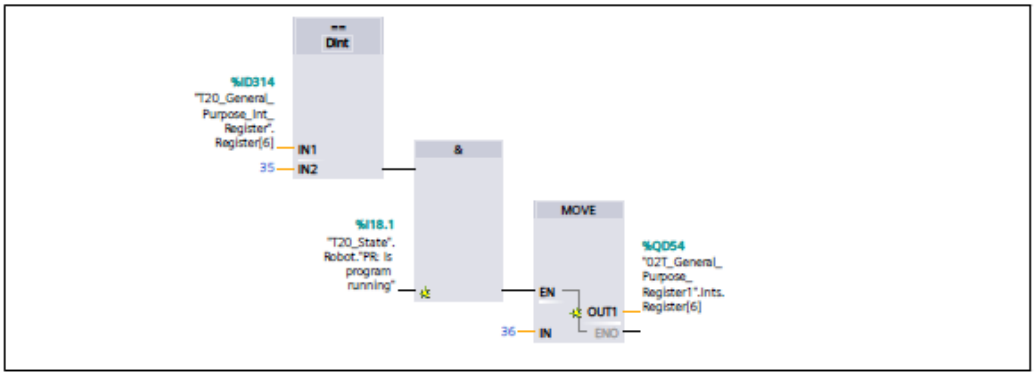
Network 9:

Logiikka kirjoittaa integer rekisteriin arvon 34 jos robotilla on arvo 33 ja robotin ohjelma pyörii.



Network 10:

Logiikka kirjoittaa integer rekisteriin arvon 36 jos robotilla on arvo 35 ja robotin ohjelma pyörii.



Program blocks

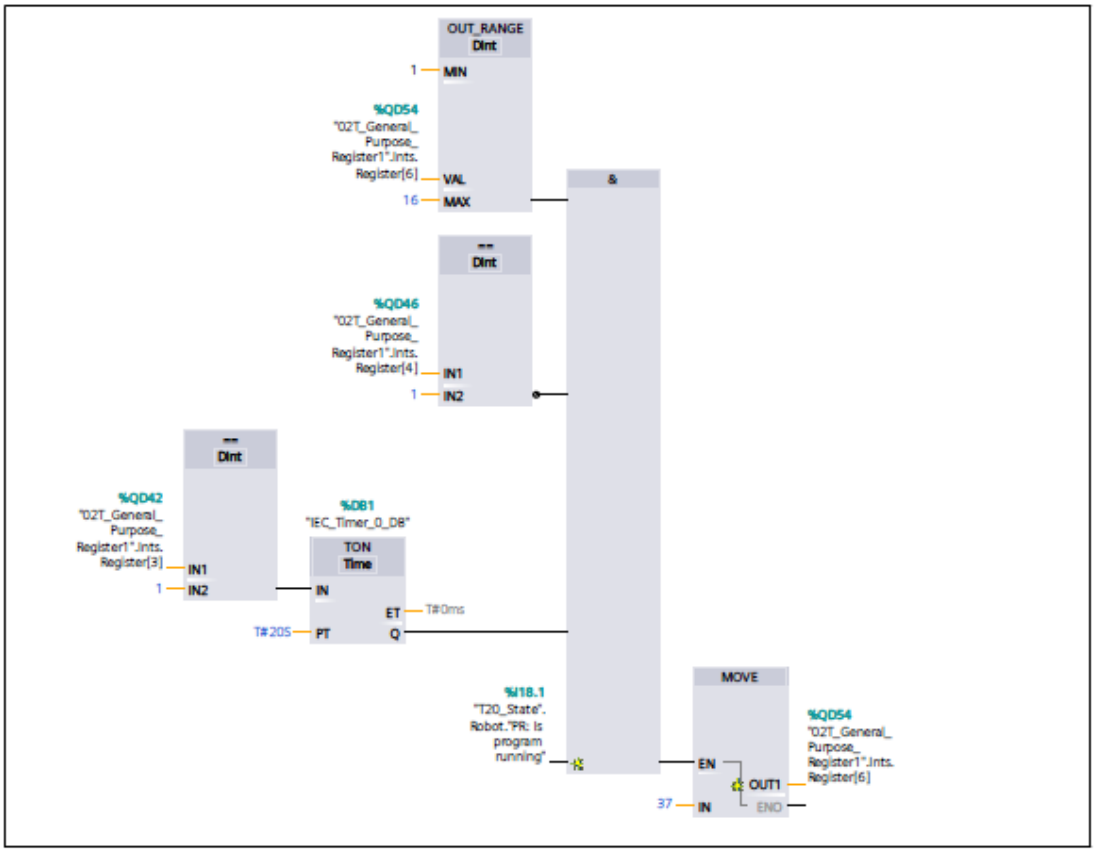
To_Barcode_reader [FC6]

To_Barcode_reader Properties					
General					
Name	To_Barcode_reader	**Number**	6	**Type**	FC
Language	FBD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	Aliohjelma robotin ohjaamiseen viivakoodinlukijan luo
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Comment
Input			
Output			
InOut			
Temp			
Constant			
▼ Return			
To_Barcode_reader	Void		

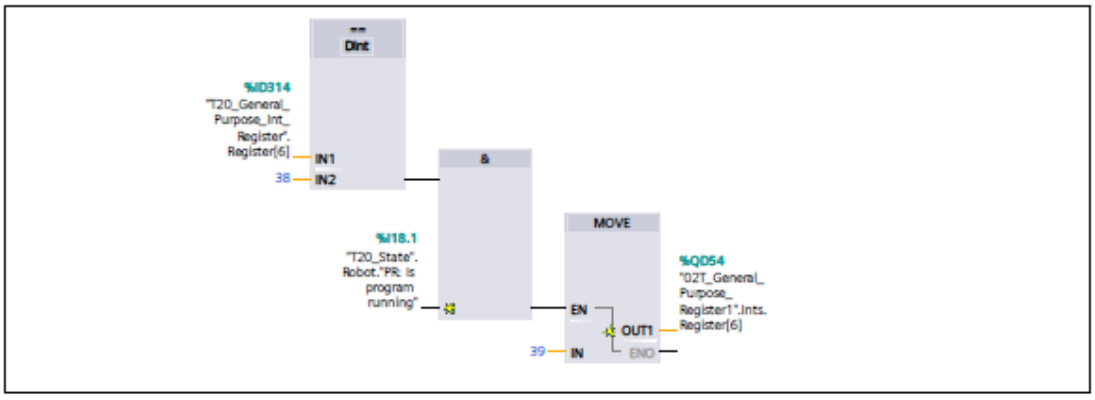
Network 1:

Robotti ei lähde suorittamaan sekvenssiä, jos logiikan integer-rekisterin arvot ovat 1-16 välissä, joka tarkoittaa että robotti on hakemassa syöttöpisteeltä makasiinia. Myöskään noutopisteen anturi ei saa tunnistaa ja jäähyjigin anturin on pitänyt olla true-tilassa 30s, jolloin makasiini on jäähtynyt. Myös robotin ohjelma pitää pyöriä. Jos &-piiri toteutuu logiikka kirjoittaa arvon 37 integer-rekisteriin.



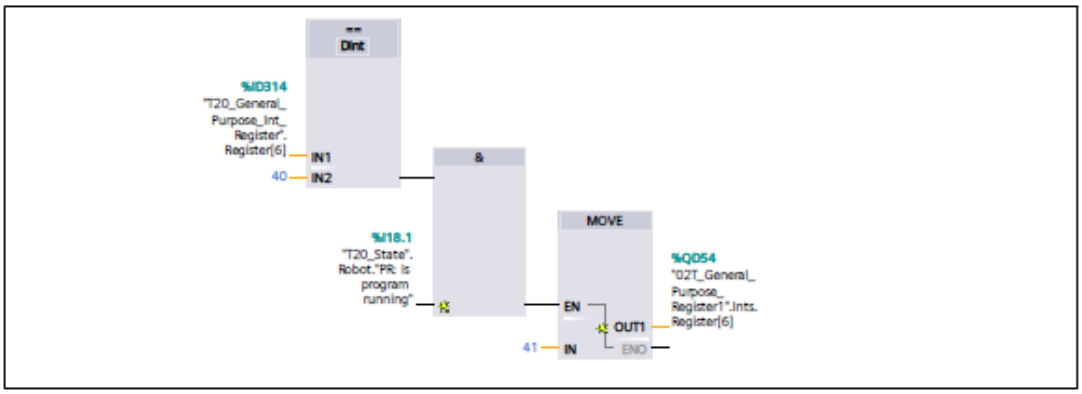
Network 2:

Logiikka kirjoittaa integer rekisteriin arvon 39 jos robotilla on arvo 38 ja robotin ohjelma pyörii.



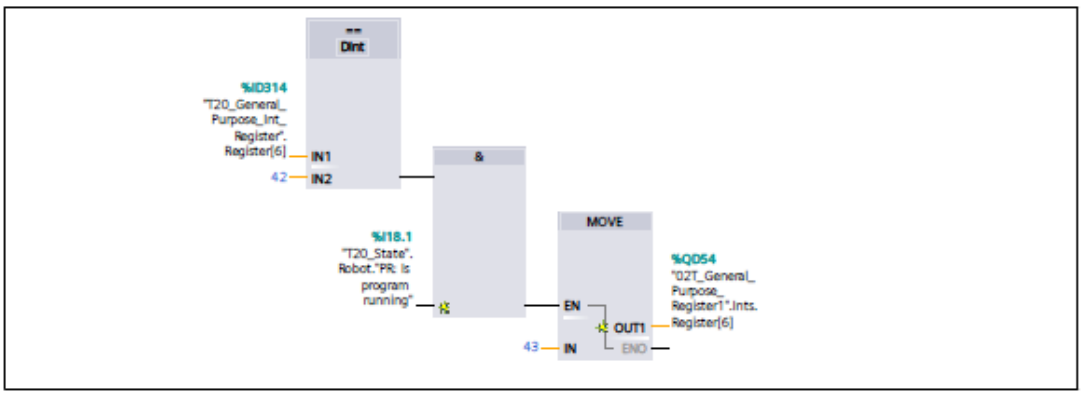
Network 3:

Logiikka kirjoittaa integer rekisteriin arvon 41 jos robotilla on arvo 40 ja robotin ohjelma pyörii.



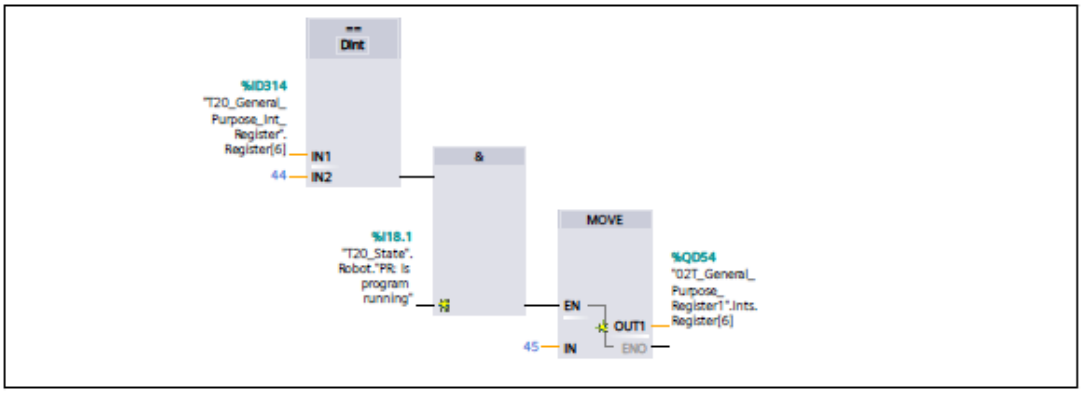
Network 4:

Logiikka kirjoittaa integer rekisteriin arvon 43 jos robotilla on arvo 42 ja robotin ohjelma pyörii.



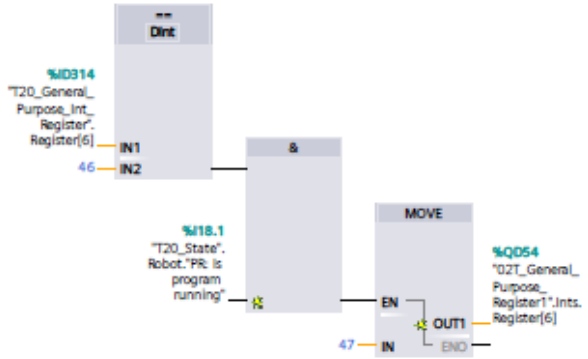
Network 5:

Logiikka kirjoittaa integer rekisteriin arvon 45 jos robotilla on arvo 44 ja robotin ohjelma pyörii.



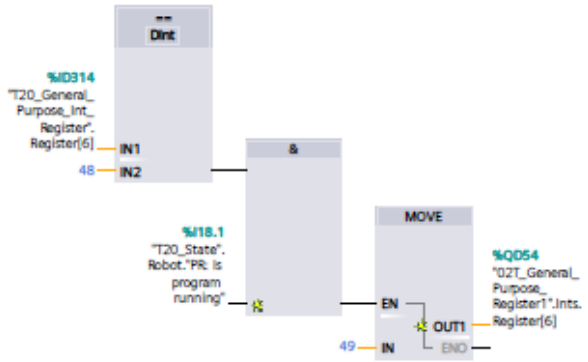
Network 6:

Logiikka kirjoittaa integer rekisteriin arvon 47 jos robotilla on arvo 46 ja robotin ohjelma pyörii.



Network 7:

Logiikka kirjoittaa integer rekisteriin arvon 49 jos robotilla on arvo 48 ja robotin ohjelma pyörii.



Program blocks

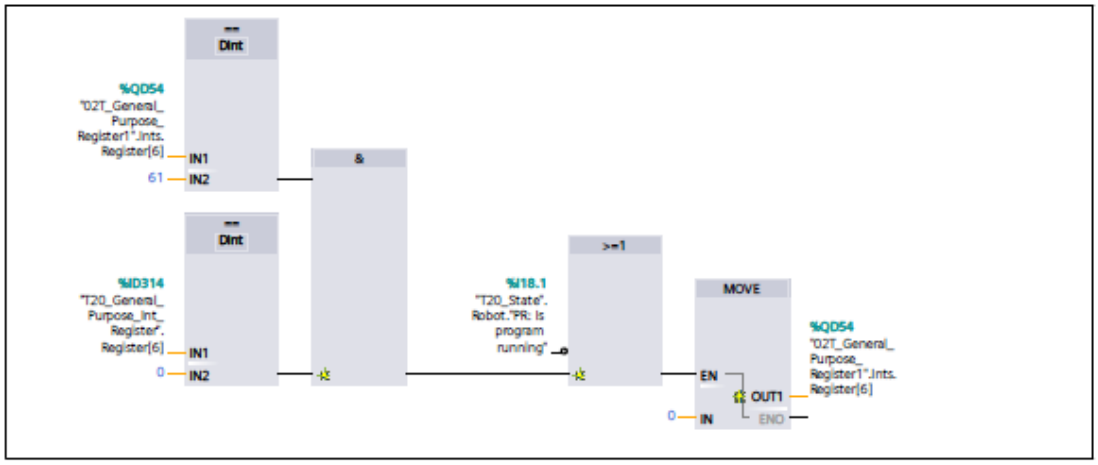
Logic_Initialization [FC4]

Logic_Initialization Properties					
General					
Name	Logic_Initialization	**Number**	4	**Type**	FC
Language	FBD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	Logiikan initalisointi sekvenssi, jonka ehdot löytyvät blockin sisältä. Logiikalle kirjoitetaan arvo 0.
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Comment
Input			
Output			
InOut			
Temp			
Constant			
▼ Return			
Logic_Initialization	Void		

Network 1:

Logiikka kirjoittaa omaan integer-rekisteriinsä arvon nolla, jos koko työnkierto on suoritettu arvolla 61 ja robotti on initialisointi-tilassa. Logiikka myös initialisoi itsensä siinä vaiheessa kun robotin ohjelma ei pyöri.



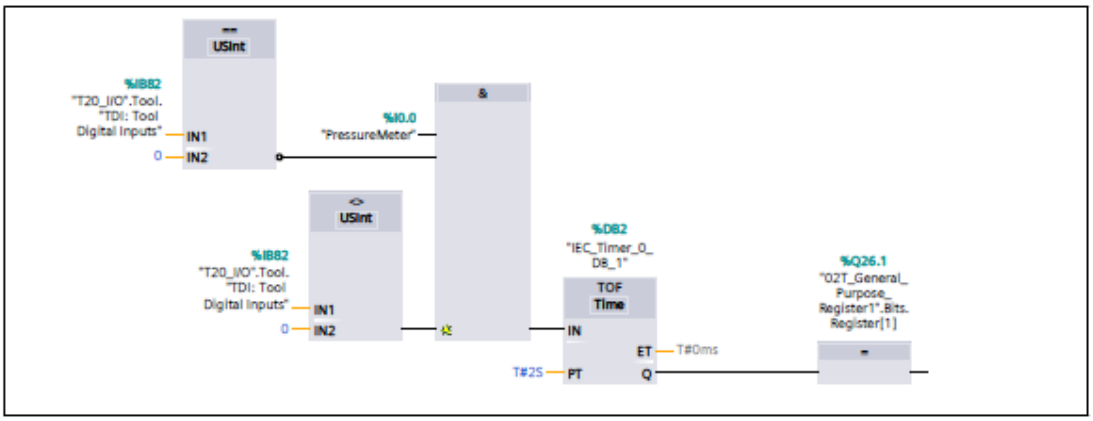
Gripper_State [FC7]

Gripper_State Properties					
General					
Name	Gripper_State	**Number**	7	**Type**	FC
Language	FBD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	Katsotaan painemittarin ja tarttujan tilatiedot
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Comment
Input			
Output			
InOut			
Temp			
Constant			
▼ Return			
Gripper_State	Void		

Network 1:

Tässä tarkistetaan onko painemittarin ulostulo päällä ja onko tarttuja kokonaan kiinni enemmän kuin 2s. Jos molemmat ehdot pitävät paikkansa, tämän jälkeen "gripping not ok!" muuttuja aktivoituu. Sama muuttuja on Main-ohjelmassa jokaisen ohjelman kutsuehtona. Eli jos muuttuja ei ole päällä, silloin aliohjelmia ei kutsuta ja robotti keskeyttää ohjelman suorittamisen.



Program blocks

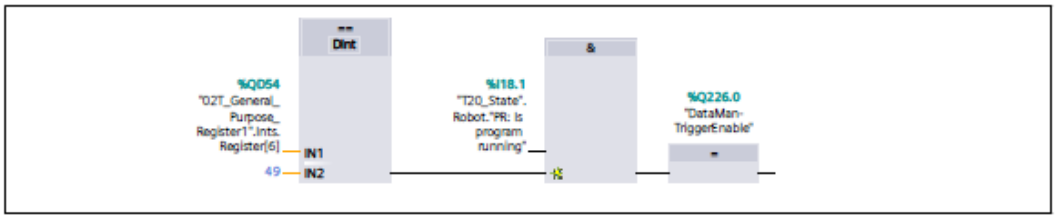
Dataman_trigger [FC8]

Dataman_trigger Properties					
General					
Name	Dataman_trigger	**Number**	8	**Type**	FC
Language	FBD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	Viivakoodin lukijan triggaussekvenssi, joka laukaisee viivakoodinlukijan.
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Comment
Input			
Output			
InOut			
Temp			
Constant			
▼ Return			
Dataman_trigger	Void		

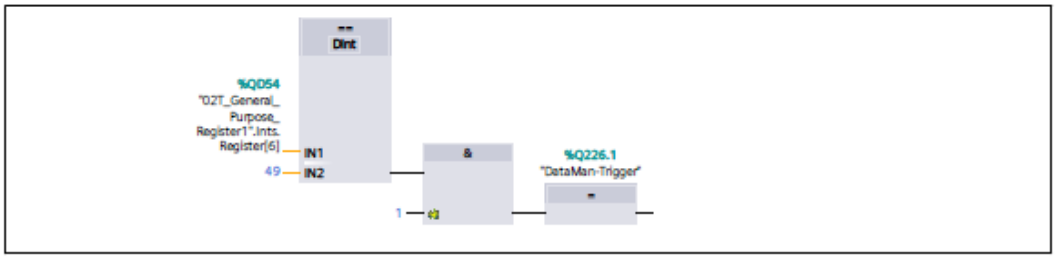
Network 1:

Logiikka katsoo, onko sen viimeisin kirjoitettu arvo 49 ja jos ehto toteutuu, niin se antaa luvan viivakoodinlukijan triggaukseen.



Network 2:

Tämä ehto laukaisee viivakoodinlukijan

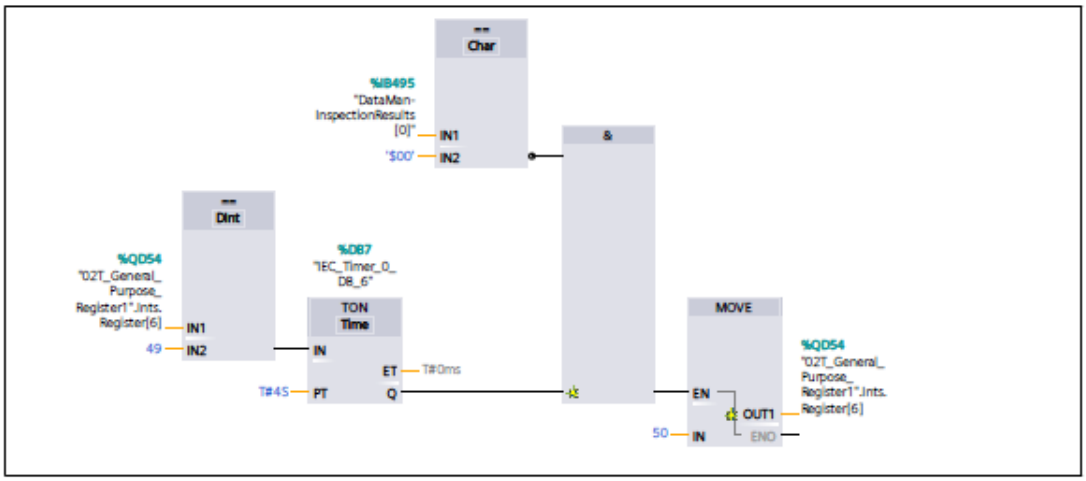


Network 3:

Tämä ehto varmistaa viivakoodinlukijan laukaisun

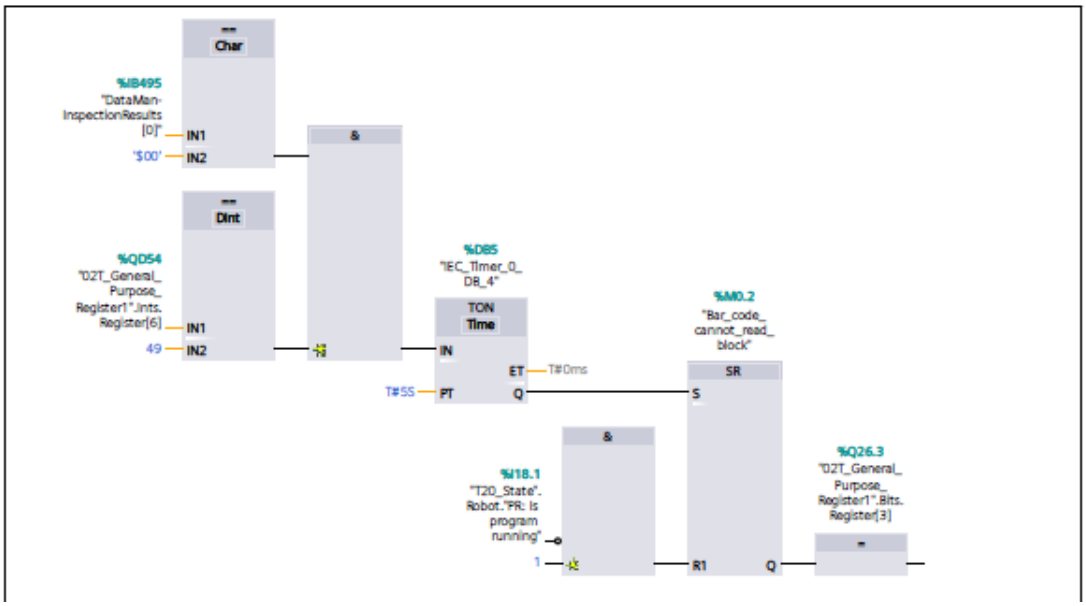
Network 4:

Tällä varmistetaan että viivakoodi on luettu onnistuneesti ja sen jälkeen logiikka kirjoittaa integer-rekisteriin arvon 50.



Network 5:

Tämä ehto on sitä varten, jos viivakoodin luku ei onnistu ja aktivoi epäonnistumista vastaavan muuttujan. Tämä muuttuja resetoituu kun robotin ohjelma ei pyöri.



Program blocks

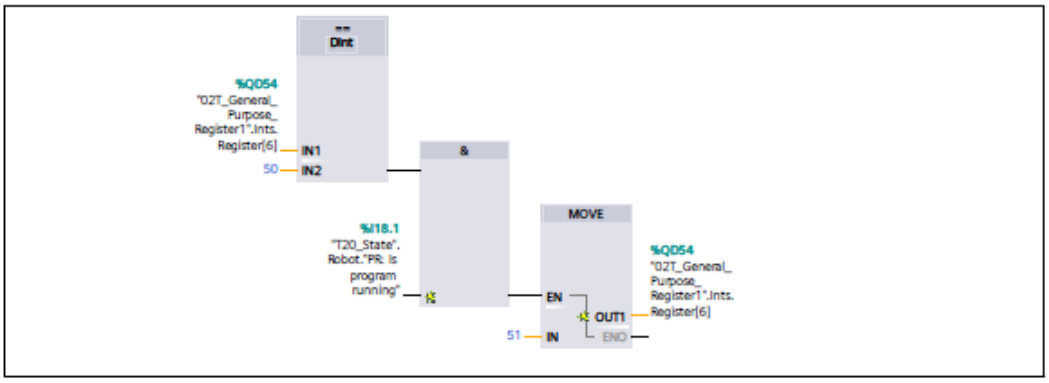
From _Barcode_reader [FC9]

From _Barcode_reader Properties					
General					
Name	From _Barcode_reader	**Number**	9	**Type**	FC
Language	FBD	**Numbering**	Automatic		
Information					
Title		**Author**		**Comment**	Aliohjelma joka ohjaa robotin viivakoodinlukijalta makasiinin noutopisteeseen
Family		**Version**	0.1	**User-defined ID**	

Name	Data type	Default value	Comment
Input			
Output			
InOut			
Temp			
Constant			
Return			
From _Barcode_reader	Void		

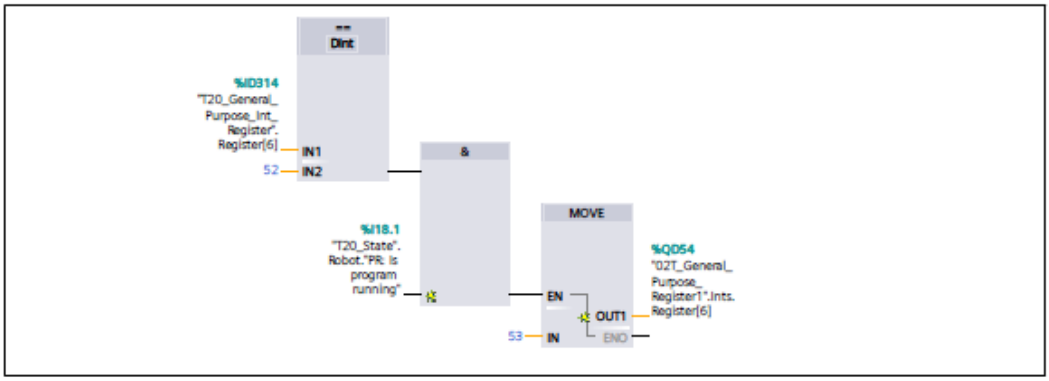
Network 1:

Logiikka tarkistaa onko robotti kirjoittanut arvon 50 ja jos ehto toteutuu niin logiikka kirjoittaa arvon 51.



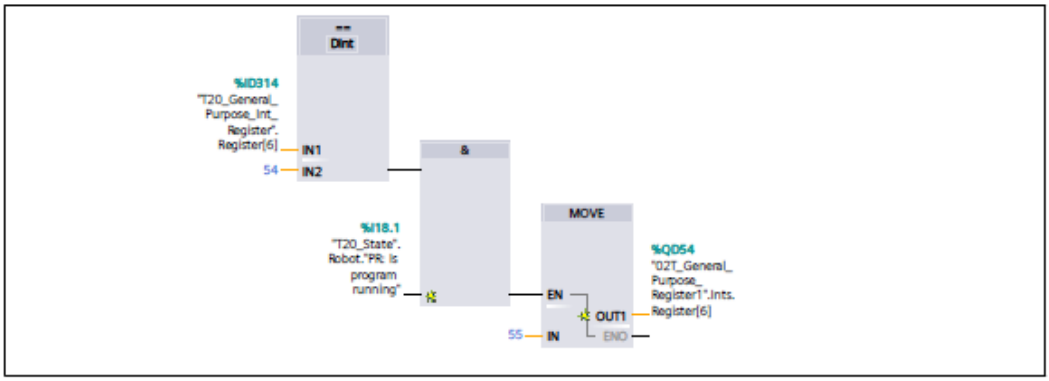
Network 2:

Logiikka tarkistaa onko robotti kirjoittanut arvon 52 ja jos ehto toteutuu niin logiikka kirjoittaa arvon 53.



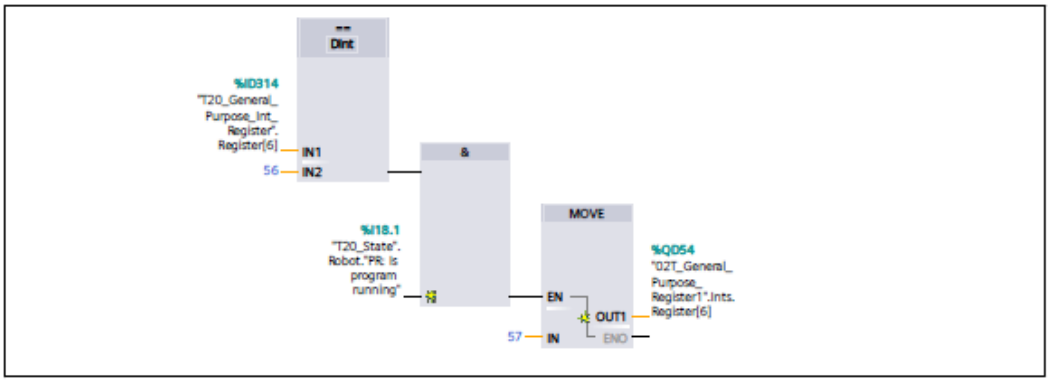
Network 3:

Logiikka tarkistaa onko robotti kirjoittanut arvon 54 ja jos ehto toteutuu niin logiikka kirjoittaa arvon 55.



Network 4:

Logiikka tarkistaa onko robotti kirjoittanut arvon 56 ja jos ehto toteutuu niin logiikka kirjoittaa arvon 57.

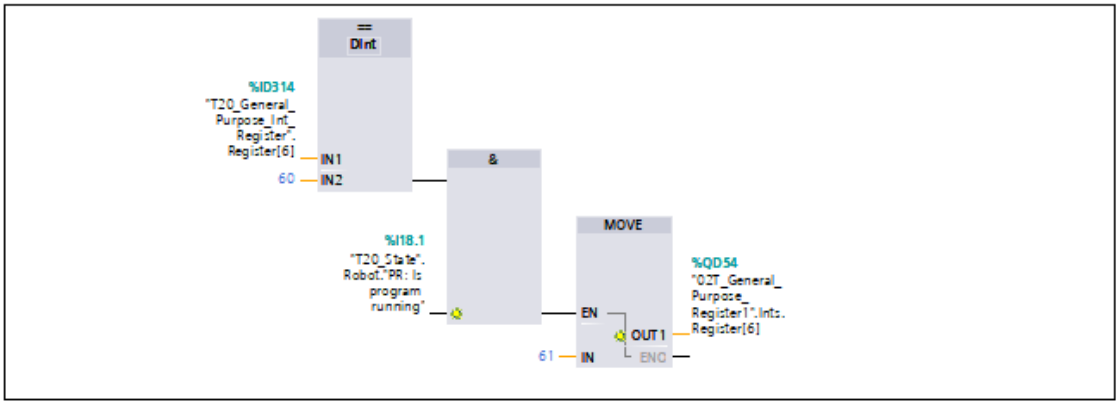


Network 5:

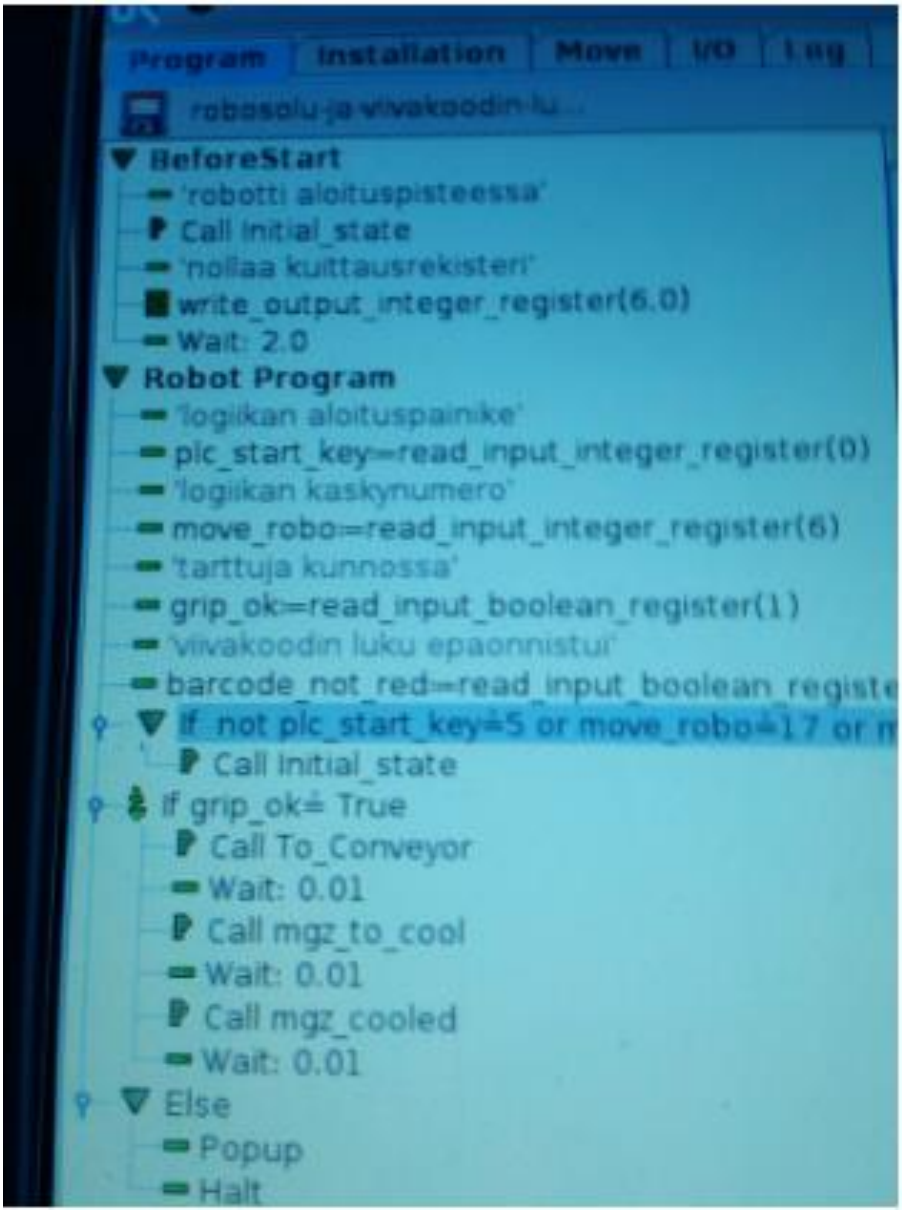
Logiikka tarkistaa onko robotti kirjoittanut arvon 58 ja jos ehto toteutuu niin logiikka kirjoittaa arvon 59.

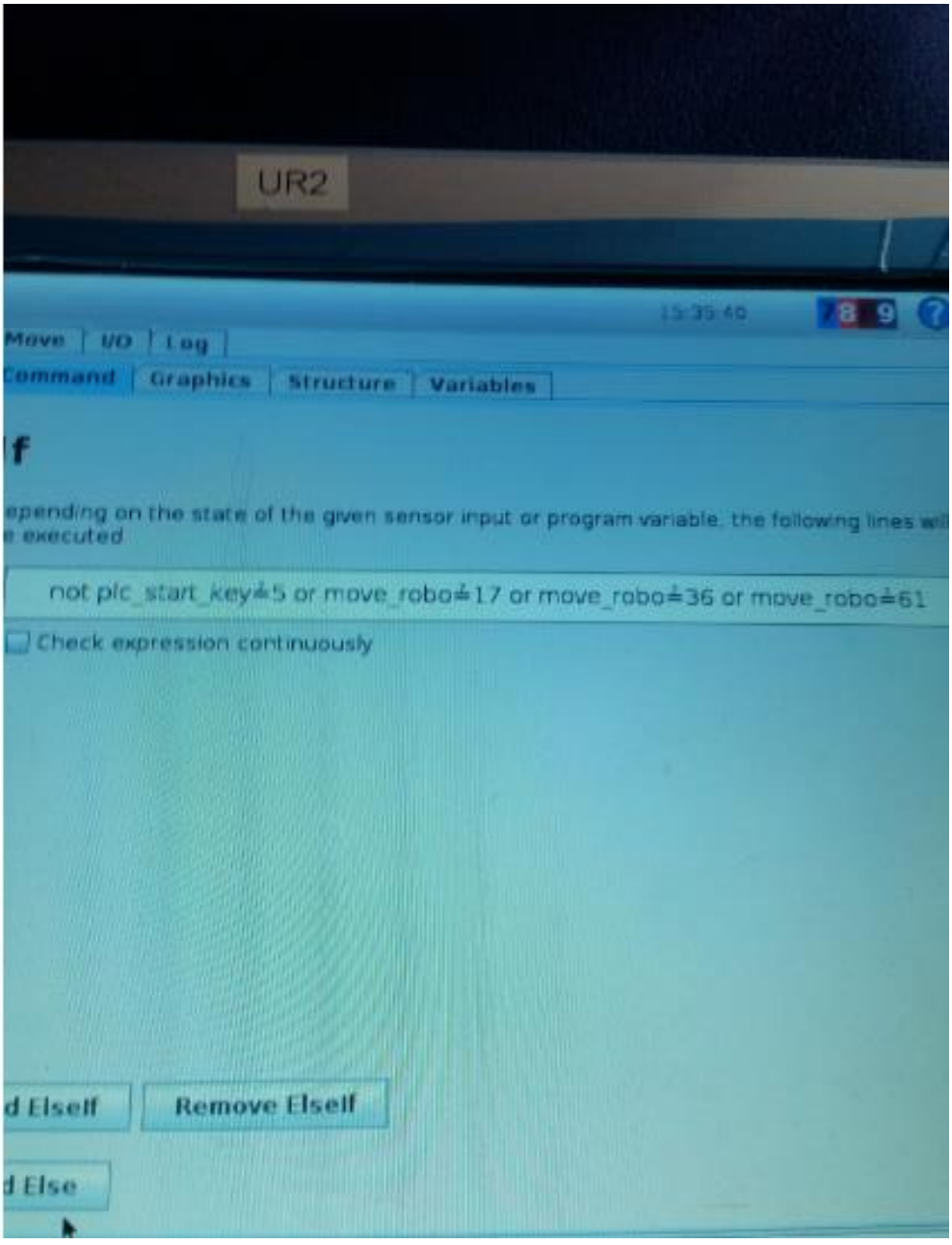
Network 6:

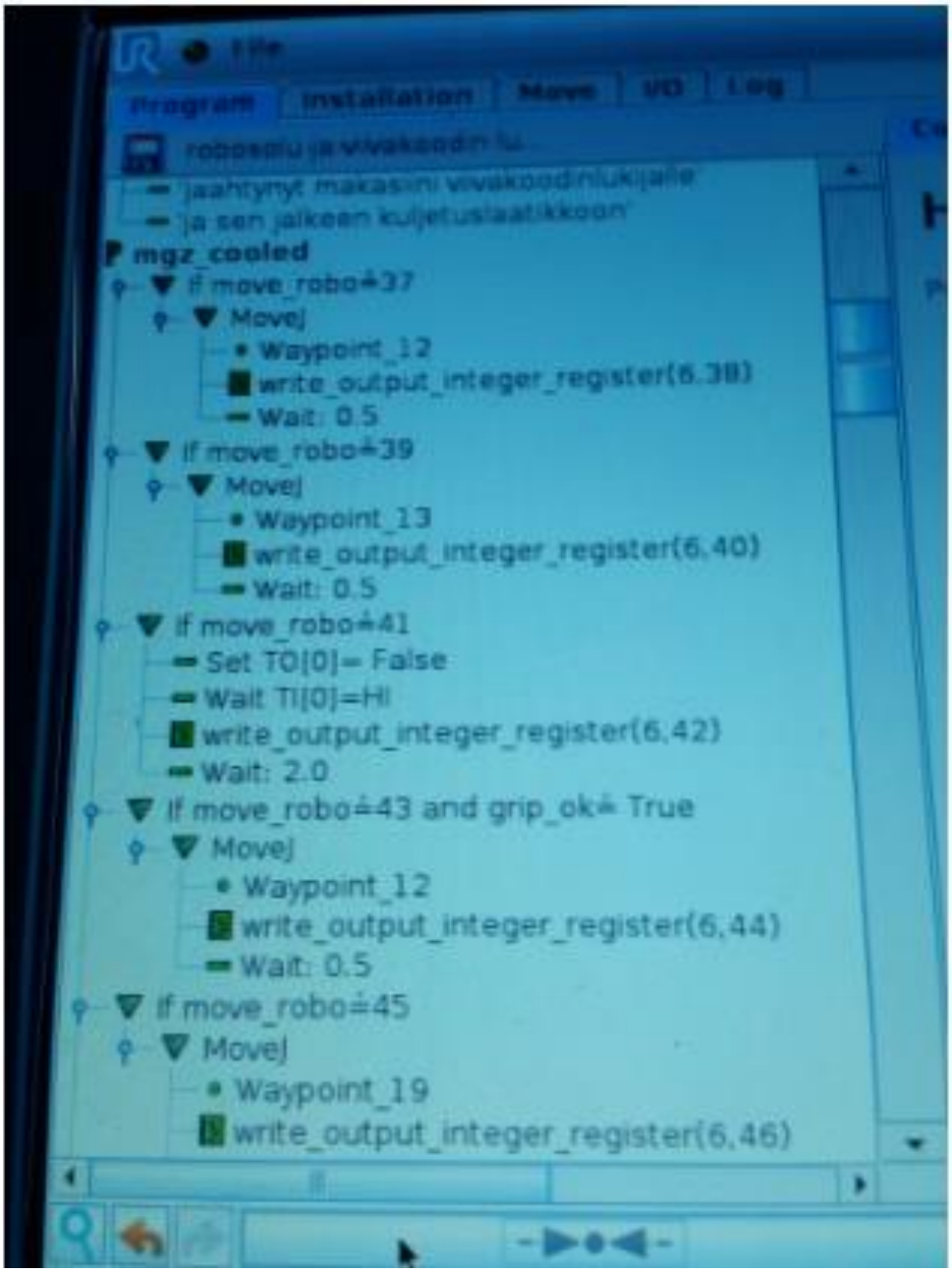
Logiikka tarkistaa onko robotti kirjoittanut arvon 60 ja jos ehto toteutuu niin logiikka kirjoittaa arvon 61. Tämä jälkeen logiikka antaa robotille luvan palata aloituspisteeseen.

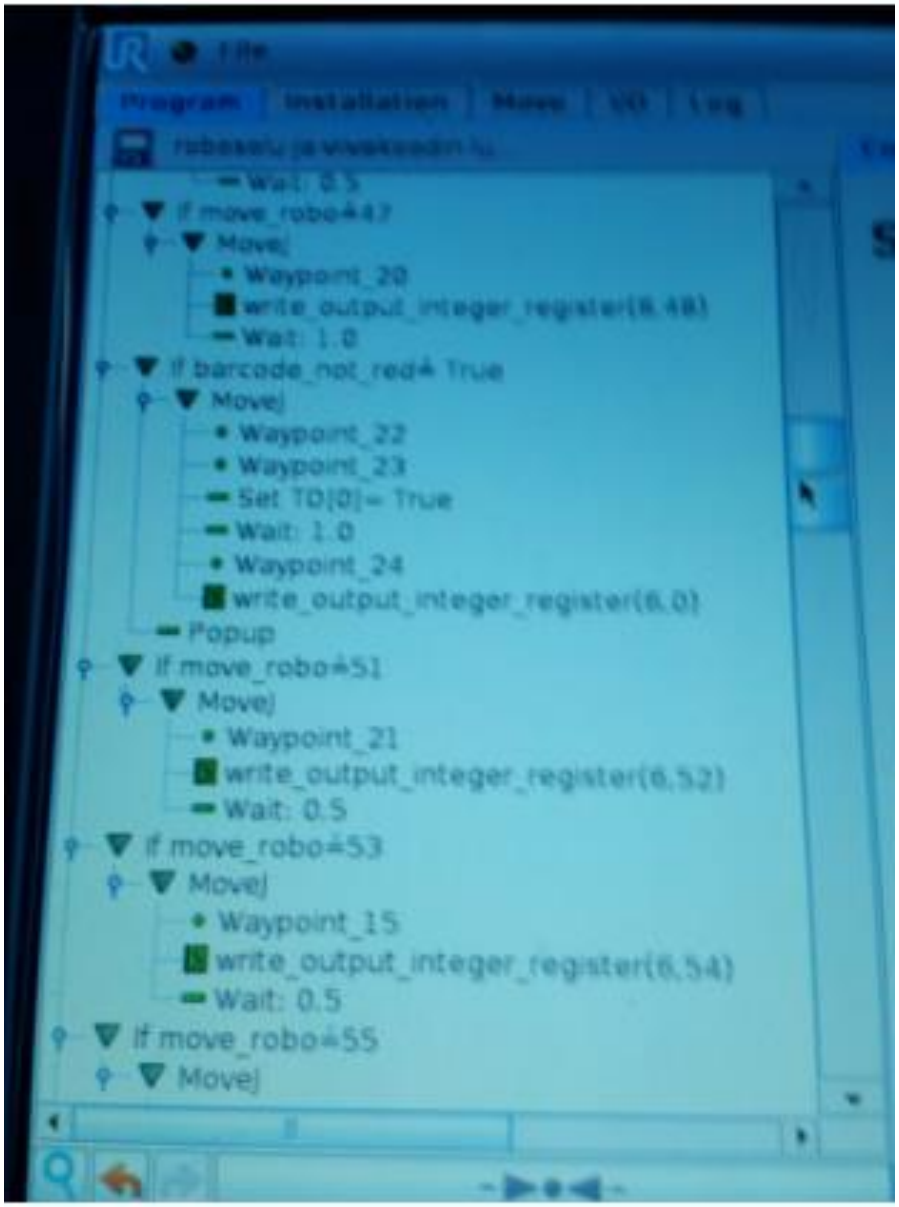


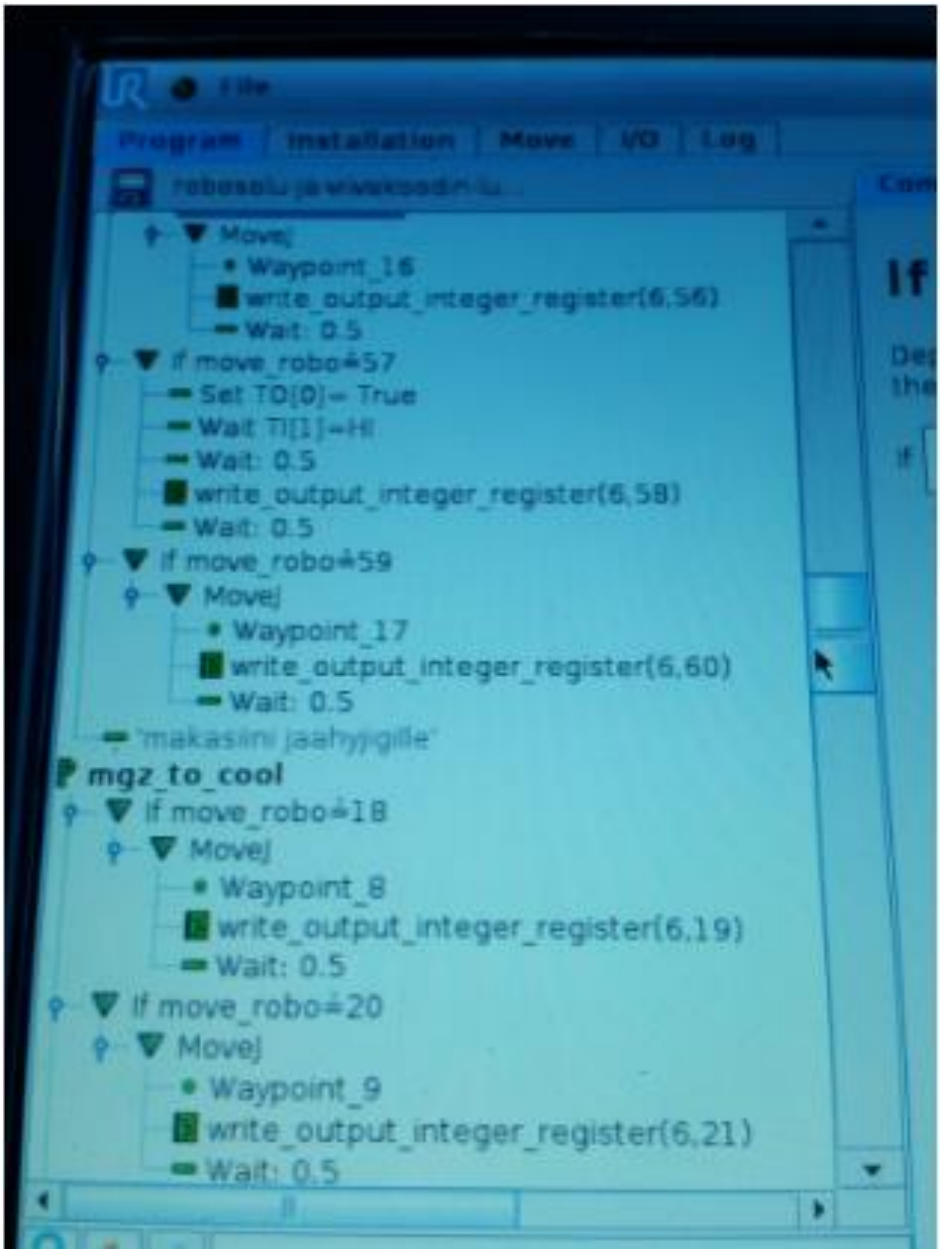
Robotin ohjelmakoodi

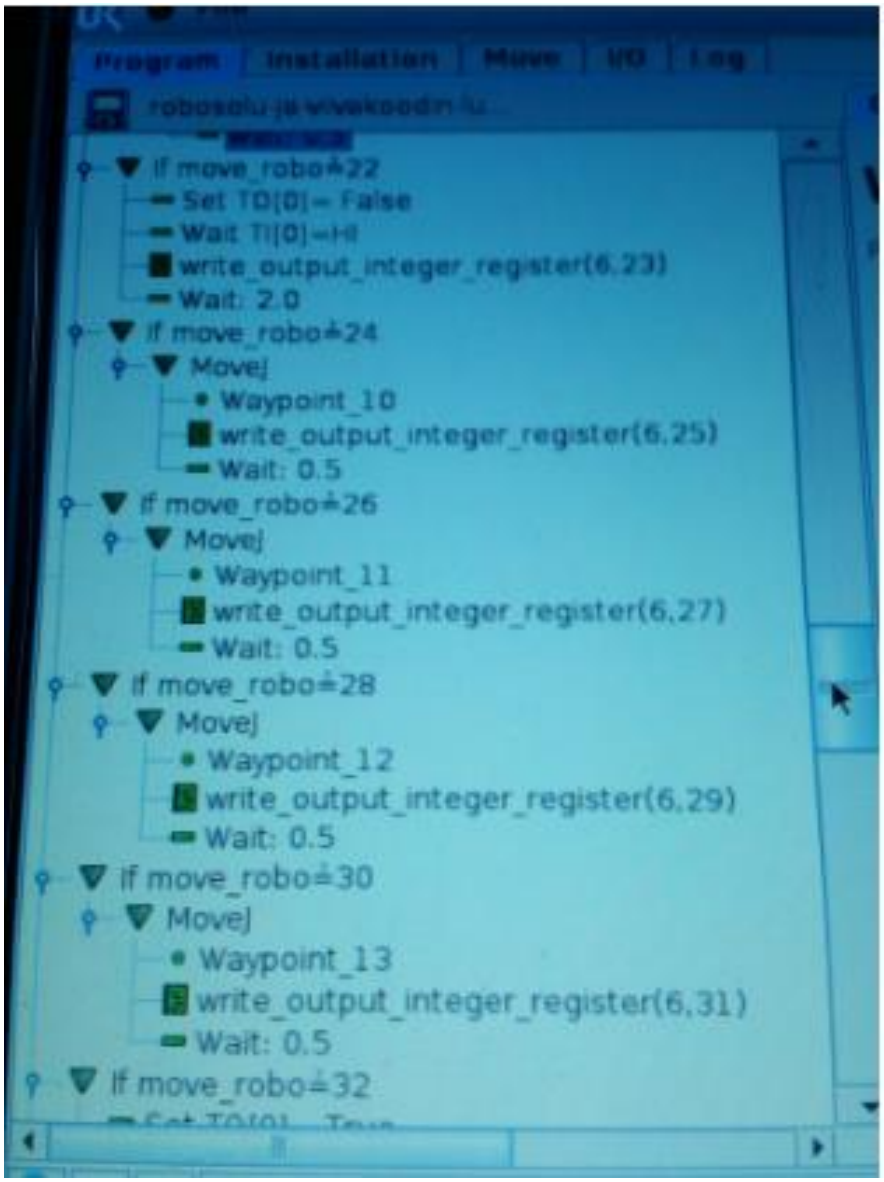


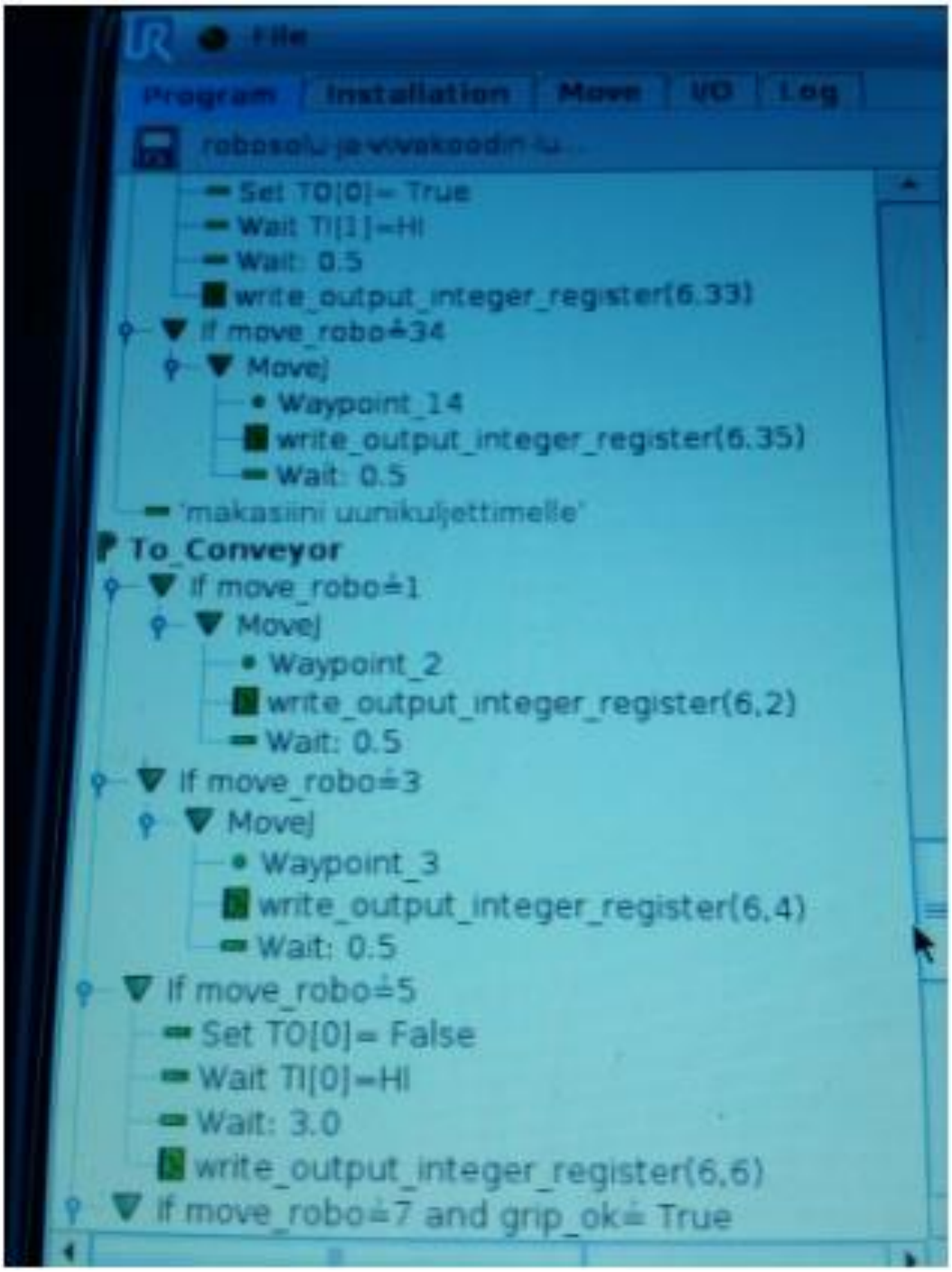

UR2
Move
I/O
Log
Command
Graphics
Structure
Variables
f
epending on the state of the given sensor input or program variable, the following lines wi
e executed
not plc_start_key≟5 or move_robo≟17 or move_robo≟36 or move_robo≟61
Check expression continuously
d Elseif
Remove Elseif
d Else

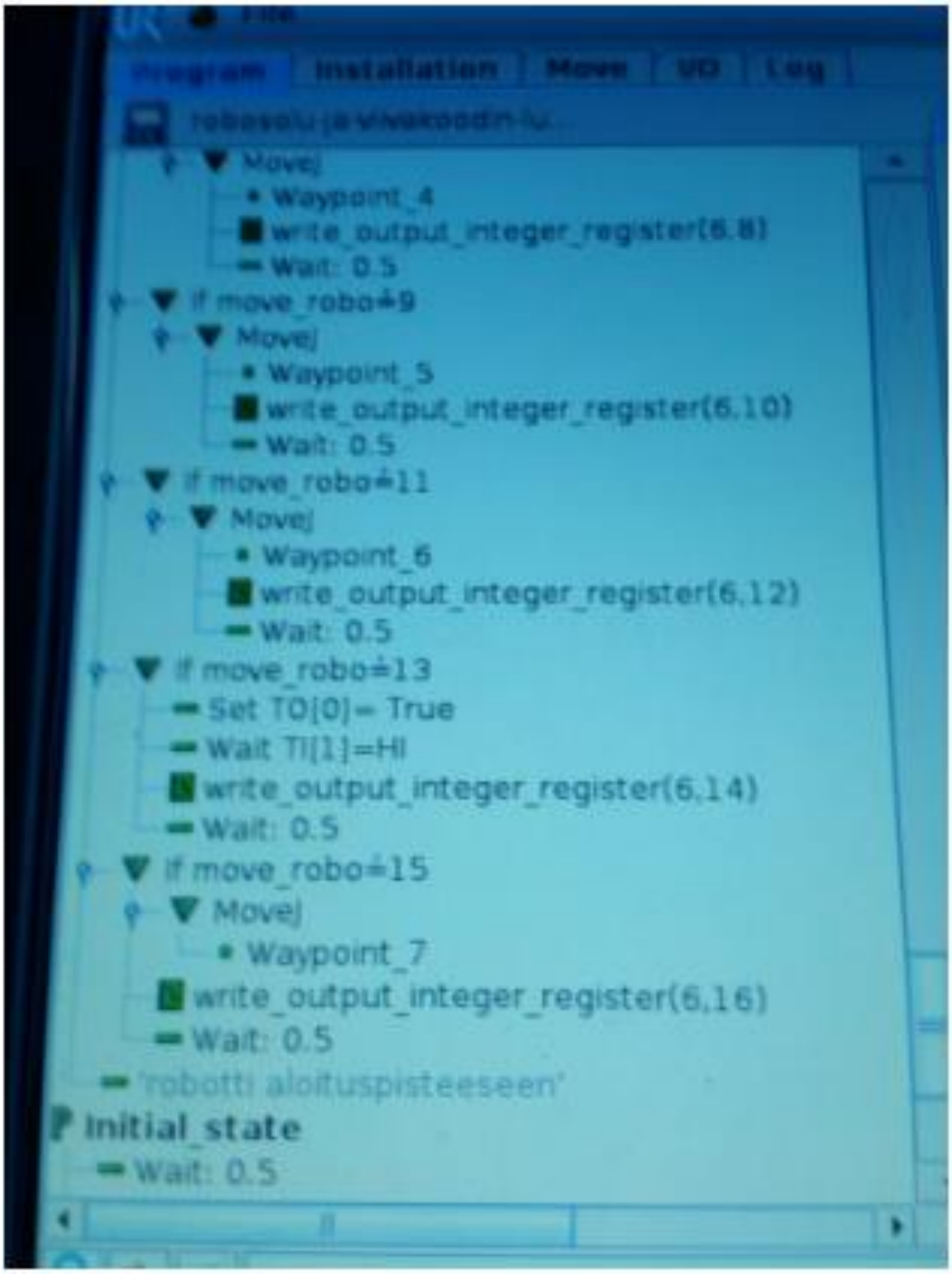

Program
Installation
Move
I/O
Log
mgz_cooled
If move_robo=37
MoveJ
Waypoint_12
write_output_integer_register(6,38)
Wait: 0.5
If move_robo=39
MoveJ
Waypoint_13
write_output_integer_register(6,40)
Wait: 0.5
If move_robo=41
Set TO[0]= False
Wait TI[0]=HI
write_output_integer_register(6,42)
Wait: 2.0
If move_robo=43 and grip_ok= True
MoveJ
Waypoint_12
write_output_integer_register(6,44)
Wait: 0.5
If move_robo=45
MoveJ
Waypoint_19
write_output_integer_register(6,46)

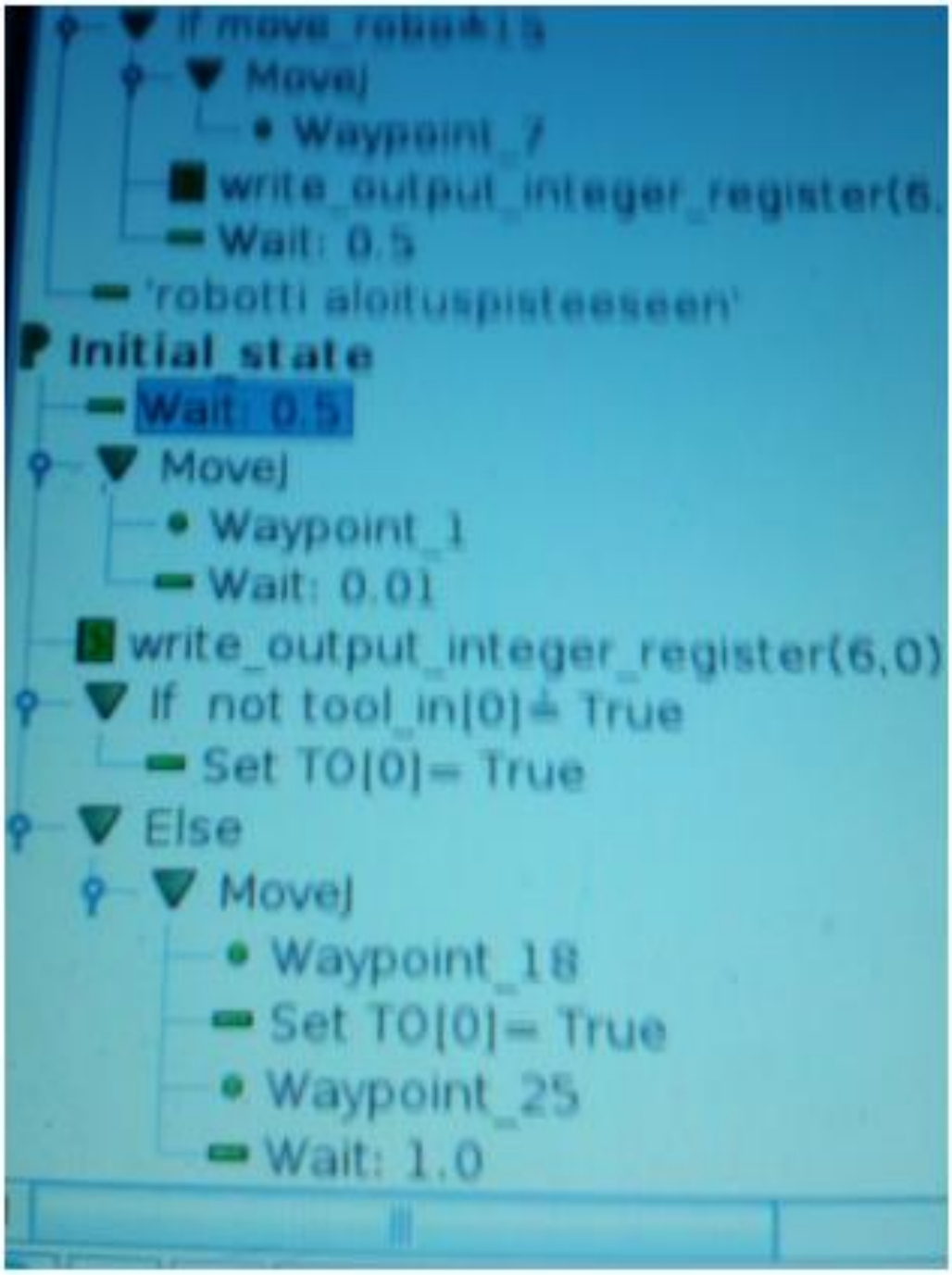

Program
Installation
Move
I/O
Log
Wait: 0.5
If move_robo=47
MoveJ
Waypoint_20
write_output_integer_register(6,48)
Wait: 1.0
If barcode_not_read= True
MoveJ
Waypoint_22
Waypoint_23
Set TO[0]= True
Wait: 1.0
Waypoint_24
write_output_integer_register(6,0)
Popup
If move_robo=51
MoveJ
Waypoint_21
write_output_integer_register(6,52)
Wait: 0.5
If move_robo=53
MoveJ
Waypoint_15
write_output_integer_register(6,54)
Wait: 0.5
If move_robo=55
MoveJ


File
Program
Installation
Move
I/O
Log
MoveJ
Waypoint_16
write_output_integer_register(6,56)
Wait: 0.5
If move_robo≠57
Set TO[0]= True
Wait TI[1]=HI
Wait: 0.5
write_output_integer_register(6,58)
Wait: 0.5
If move_robo≠59
MoveJ
Waypoint_17
write_output_integer_register(6,60)
Wait: 0.5
"makasiini jaahyjgille"
mgz_to_cool
If move_robo≠18
MoveJ
Waypoint_8
write_output_integer_register(6,19)
Wait: 0.5
If move_robo≠20
MoveJ
Waypoint_9
write_output_integer_register(6,21)
Wait: 0.5


Program
Installation
Move
I/O
Log
If move_robo≟22
Set TO[0]:= False
Wait TI[0]=H
write_output_integer_register(6,23)
Wait: 2.0
If move_robo≟24
MoveJ
Waypoint_10
write_output_integer_register(6,25)
Wait: 0.5
If move_robo≟26
MoveJ
Waypoint_11
write_output_integer_register(6,27)
Wait: 0.5
If move_robo≟28
MoveJ
Waypoint_12
write_output_integer_register(6,29)
Wait: 0.5
If move_robo≟30
MoveJ
Waypoint_13
write_output_integer_register(6,31)
Wait: 0.5
If move_robo≟32


File
Program
Installation
Move
I/O
Log
Set TO[0]= True
Wait TI[1]=HI
Wait: 0.5
write_output_integer_register(6,33)
If move_robo=34
MoveJ
Waypoint_14
write_output_integer_register(6,35)
Wait: 0.5
'makasiini uunikuljettimelle'
To_Conveyor
If move_robo=1
MoveJ
Waypoint_2
write_output_integer_register(6,2)
Wait: 0.5
If move_robo=3
MoveJ
Waypoint_3
write_output_integer_register(6,4)
Wait: 0.5
If move_robo=5
Set TO[0]= False
Wait TI[0]=HI
Wait: 3.0
write_output_integer_register(6,6)
If move_robo=7 and grip_ok= True


Program
Installation
Move
I/O
Log
MoveJ
Waypoint_4
write_output_integer_register(6,8)
Wait: 0.5
If move_robo=9
MoveJ
Waypoint_5
write_output_integer_register(6,10)
Wait: 0.5
If move_robo=11
MoveJ
Waypoint_6
write_output_integer_register(6,12)
Wait: 0.5
If move_robo=13
Set TO[0]= True
Wait TI[1]=HI
write_output_integer_register(6,14)
Wait: 0.5
If move_robo=15
MoveJ
Waypoint_7
write_output_integer_register(6,16)
Wait: 0.5
"robotti aloituspisteeseen"
Initial_state
Wait: 0.5


MoveJ
Waypoint_7
write_output_integer_register(6,
Wait: 0.5
'robotti aloituspisteeseen'
Initial_state
Wait: 0.5
MoveJ
Waypoint_1
Wait: 0.01
write_output_integer_register(6,0)
If not tool_in[0]= True
Set TO[0]= True
Else
MoveJ
Waypoint_18
Set TO[0]= True
Waypoint_25
Wait: 1.0