Tampereen ammattikorkeakoulu

Konttinosturin ohjausjärjestelmän suunnittelu

Simuloidun konttinosturin pienoismallin ohjaus PLC:n avulla

Toni Valkama

OPINNÄYTETYÖ
Marraskuu 2020

Konetekniikka
Koneautomaatio

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Konetekniikka
Koneautomaatio

VALKAMA, TONI:
Konttinosturin ohjausjärjestelmän suunnittelu
Simuloidun konttinosturin pienoismallin ohjaus PLC:n avulla

Opinnäytetyö 42 sivua, joista liitteitä 2 sivua
Marraskuu 2020

Opinnäytetyön tarkoituksena oli rakentaa automatisoitu ohjausjärjestelmä konttinosturin pienoismalliin PLC-ohjelmaa käyttäen. Konttinosturin oli tarkoitus tulla opetuskäyttöön Tampereen ammattikorkeakoululle, ja työn rajaukset tulivat koulun opettajilta. Koodi kirjoitettiin Siemensin Tia Portal -ohjelmalla FBD-kielellä. Kieli sekä ohjelma valittiin siksi, että ne ovat yksinkertaisia sekä jo opetuskäytössä.

Työ oli jaettu mekaaniseen ja ohjelmalliseen osaan, ja töiden oli tarkoitus edetä yhdessä. Koodi täytyi testata simuloimalla, sillä konttinosturin rakennus ei edennyt kuten oli arvioitu. Ilman fyysistä rakennelmaa työ vaihdettiin teoreettiseksi. Koodin kirjoittaminen alkoi jaottelemalla koodi pienempiin osa-alueisiin, mikä selkeytti projektia huomattavasti, ja mahdollisti keskittymisen yhteen alueeseen kerrallaan. Tuloksena on valmis konttinosturin koodi sekä käyttöliittymä, jotka simuloinnin jälkeen todistettiin toimiviksi.

Koodi on hyvin kommentoitu ja yksinkertainen. Automaattisykli onnistui erinomaisesti, ja käyttöliittymä toimii sen kanssa hyvin yhteen. Jatkotutkimukseksi ehdotetaan konttinosturin käyttöönottoa sen rakennuttua. Käyttöönotossa voi vertailla toteutuvatko tehdyt oletukset, ja kuinka koodi toimii todellisuudessa. Koodia voi kehittää lisäämällä uusia virhetilanteita, joita voi ilmetä käyttöönoton yhteydessä.

Asiasanat: programmable logic controller, konttinosturi, siemens, tia portal, s7-1500

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in Mechanical Engineering
Machine Automation

VALKAMA, TONI:
Designing a Control System for a Miniature Container Crane
Controlling a Simulated Miniature Container Crane Using PLC
Bachelor's thesis 42 pages, appendices 2 pages
November 2020

The purpose of this thesis was to build an automated control system for a miniature container crane using a Programmable Logic Controller. The container crane was aimed to be a part of teaching material for Tampere University of Applied Sciences and the guidelines were given by the teachers. The code was written with Siemens Tia Portal using Function Block Diagram as the coding language. The language and the program were chosen due to their simplicity and their existing usage in teaching.

The project was split into building the physical crane and programming of the control system which were intended to progress simultaneously. The code had to be tested via simulation since the physical project advanced slower than anticipated. Without the physical crane this thesis was changed to a theoretical thesis. Writing of the code began by breaking it down into smaller sections which clarified the process and allowed focusing on one section at a time. As a result, working code and a user interface were achieved and verified via simulation.

The code is well commented and simple. The automatic cycle was a success and the user interface communicated well with the code. For follow-up research, commissioning of the container crane after it is complete is suggested. Commissioning allows empirical observation of whether the assumptions made in this study hold true and how the code works in reality. The code can be developed further by adding new error situations that may occur during the process.

Key words: programmable logic controller, container crane, siemens, tia portal, s7-1500

SISÄLLYS

ERITYISSANASTO tai LYHENTEET JA TERMIT (valitse jompikumpi)

CPU	Central Processing Unit
DB	Data Block
FC	Function
FB	Function Block
FBD	Function Block Diagram
LAD	Ladder Logic
HMI	Human-Machine Interface
OB	Organization Block
OCC	One Cable Connection
SCL	Structured Control Language
STL	Statement List
PLC	Programmable Logic Controller

1 Johdanto

Tämän opinnäytetyön tarkoituksena on luoda Tampereen ammattikorkeakoululle konttinosturin pienoismalliin PLC-ohjelma (Programmable Logic Controller), joka tulee opetuskäyttöön. Prosessin alussa konttinosturiprojekti jaettiin mekaaniseen- ja ohjelmalliseen osaan. Tämä työ keskittyy toimilaitteiden valintaan, sekä ohjelman luomiseen määriteltyjen rajauksien puitteissa, kun taas mekaaninen osuus toteutetaan erillisenä työnä.

Ohjelma suunnitellaan siten, että nosturin ympärillä on kaksi lastausaluetta, sekä yksi pinoamisalue, joiden välillä kontteja siirretään käyttäjän ohjeiden mukaisesti. Konttinosturi on kolmiakselinen laite, jota ohjataan manuaalisesti tai automaattisesti PLC:n avulla. Ohjelmassa otetaan huomioon, että kontteja on kahta eri kokoa. PLC-ohjelman lisäksi konttinosturille tehdään käyttöliittymä HMI-paneelille. Paneelin avulla nosturia saa ohjattua käyttäjän haluamalla tavalla.

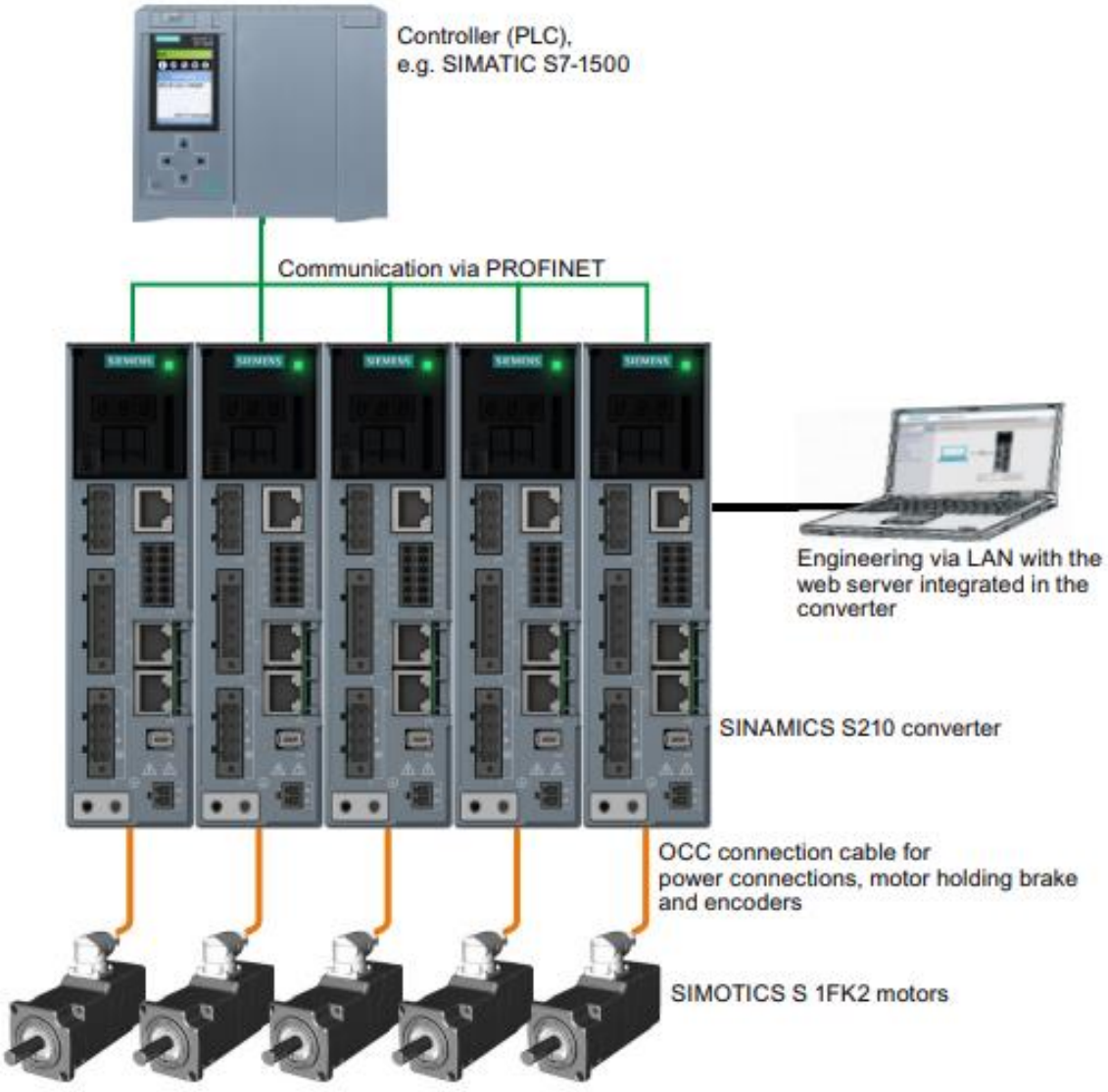
1.1 Mikä on konttinosturi?

Nosturit ovat koneita, joita käytetään raskaiden kuormien tai vaarallisten materiaalien kuljettamiseen paikasta toiseen. Nämä toimet tapahtuvat yleensä teollisuudessa, kuten tehtaissa, rakennusalalla, meriteollisuudessa ja satama-alueilla. Lukuisat nosturityypit ovat laajalti käytössä monilla teollisuuden aloilla, kuten ylä- ja siltanosturit, pukkinosturit, puominosturit ja pyörivät torninosturit. Ne voidaan luokitella sen vapausasteen perusteella, jota tukimekanismi tarjoaa ripustuskohdassa. (Ramil, Mohamed, Abdullahi, Jaafar & Lazim 2017, 2)

Liitteessä yksi on erilaisia esimerkkejä nostureita. RTG-nosturi (Rubber Tired Gantry crane) muistuttaa eniten projektissa rakennettavaa nosturia. Erona on, ettei pienoismalli liiku renkaiden avulla, vaan alumiinikiskojen avulla.

2 Teoria

Järjestelmä, jota PLC ohjaa, voi esimerkiksi koostua logiikasta, ohjattavista laitteista, sensoreista sekä käyttäjän kirjoittamasta koodista. Koodilla ohjataan asennettuja laitteita, kun ne on kytketty toisiinsa kiinni. Ennen kytkemistä toimintoja on mahdollista simuloida ilman fyysisiä laitteita. Kuvassa 1 näkyy esimerkki järjestelmäkokonaisuudesta, jossa on PLC, vahvistimet ja servomoottorit. Laitteet on liitetty toisiinsa PROFINET:n avulla sekä OCC-kaapeleilla (One Cable Connection). PROFINET on ensimmäinen teollisuuteen integroitu standardi Ethernet-lähiverkkoratkaisulle. Se käyttää hyväkseen Ethernetin avoimia viestintäkanavia yrityksen johtamistasolta itse prosessin tasolle asti (Metter & Pigan 2015).



Kuva 1. Järjestelmän yleiskatsaus (Siemens 2019, 30)

2.1 Logiikka

PLC (Programmable Logic Controller) eli ohjelmoitava logiikka on pieni tietokone, jolla voi ohjata kokonaisia järjestelmiä. PLC:n liitännät on jaettu tuloihin ja lähtöihin (Input/Output). Tulojen avulla PLC saa tietoa, kuinka järjestelmä toimii. Esimerkkejä tuloista ovat anturit tai kytkimet. Kytkimien avulla saadaan esimerkiksi On/Off-tieto tietyltä laitteelta, kun taas antureilla voi mitata eri suureita. PLC:n lähtöihin yhdistetään ohjattavat laitteet kuten moottorit ja venttiilit. Tulojen ja logiikan sisällä olevan ohjelman avulla saa ohjattua toimilaitteita haluamalla tavalla.

Simatic S7 on ohjelmoitavaan logiikkaohjaimeen perustuva järjestelmä. Ohjaustehtävän ratkaisu tallennetaan CPU:n (Central Processing Unit) käyttömuistiin ohjelmaohjeiden muodossa. Prosessori lukee yksittäiset ohjeet peräkkäin, tulkitsee niiden sisällön ja suorittaa ohjelmoidun toiminnon. CPU sisältää myös käyttöjärjestelmän. Se varmistaa laitteen sisäisten käyttötoimintojen suorittamisen, kuten yhteydenpidon ohjelmointilaitteen kanssa ja tietojen varmuuskopioinnin sähkökatkon sattuessa. Käyttöjärjestelmä myös käynnistää käyttäjäohjelman, joko toistuvasti syklisesti tai riippuen liipaisutapahtumasta, kuten hälytyksestä. (Berger 2012, 14)

2.2 Ohjelmointi

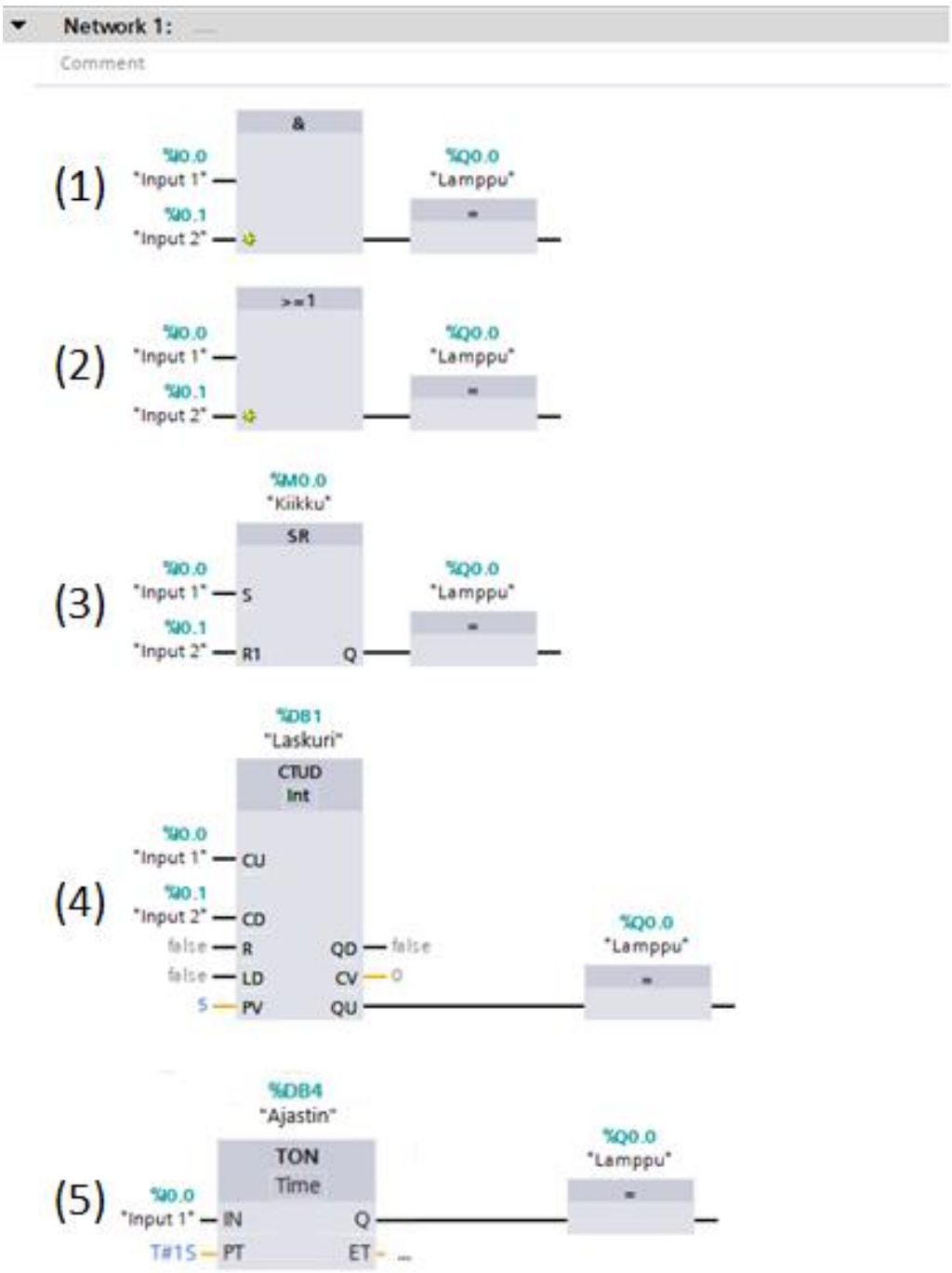
Siemens Tia Portal käyttää erilaisia lohkoja ja operaatioita bittien liikuttamiseen. Lohkoiksi kutsutaan kokonaisuuksia, jotka koostuvat operaatioista, ja ne suorittavat ohjelmaa sykleittäin. Näitä kutsutaan eli käsketään yleensä päälohkolla, joka toimii organisaatiolohkona (katso 4.3). Operaatiot ovat pienempiä toiminnallisia lohkoja, jotka toimivat rajauksina ohjelmalle.

Usein käytössä olevia perusoperaatiota ovat JA-portti, TAI-portti, kiikku, laskuri sekä ajastin, esiteltynä kuvassa 2. JA-portissa kaikkien sisääntulojen tarvitsee olla aktiivisena, jotta ulostulo on positiivinen. TAI-portissa vain yhden sisääntulon aktiivisuus riittää. Kiikku säilyttää tilansa sisääntulojen Set ja Reset avulla. Set

aktivoi kiikun, ja Reset nollaa sen. Kiikuissa sisääntulot on asetettu tärkeysjärjestykseen. Jos molemmat sisääntulot ovat aktiivisena samaan aikaan, kiikun lähtö asettuu tärkeysjärjestyksen mukaan.

Laskureilla voidaan lisätä tai vähentää arvoa riippuen mitä käyttäjä haluaa. Sisääntulot CU (Count Up) ja CD (Count Down) nostavat tai laskevat arvoa yhdellä aktivoituessaan. R (Reset) nollaa laskurin arvon. LD (Load Input) saadessa pulssin, se asettaa sisääntulon PV arvon ulostuloon CV (Current Value). PV on käyttäjän asettama raja-arvo, jossa ulostulo QU muuttuu aktiiviseksi. Jos arvo on alle asetetun rajan, ulostulo on negatiivinen. CV näyttää laskurin tämänhetkisen arvon.

Ajastimilla voidaan pitkittää signaalin etenemistä, tai katkaista se tietyn aikamääreen kuluttua. Esimerkissä on käytössä viiveajastin (ON-delay), joka avaa piirin määritellyn ajan kuluttua. Ajastimen sisääntulo IN aktivoi lohkon, jolloin ajastin aktivoituu. PT (Preset Time) sisääntuloon määritellään aika, jonka kuluttua aktivoitu lohko avaa piirin. ET (Elapsed Time) näyttää kuluneen ajan.

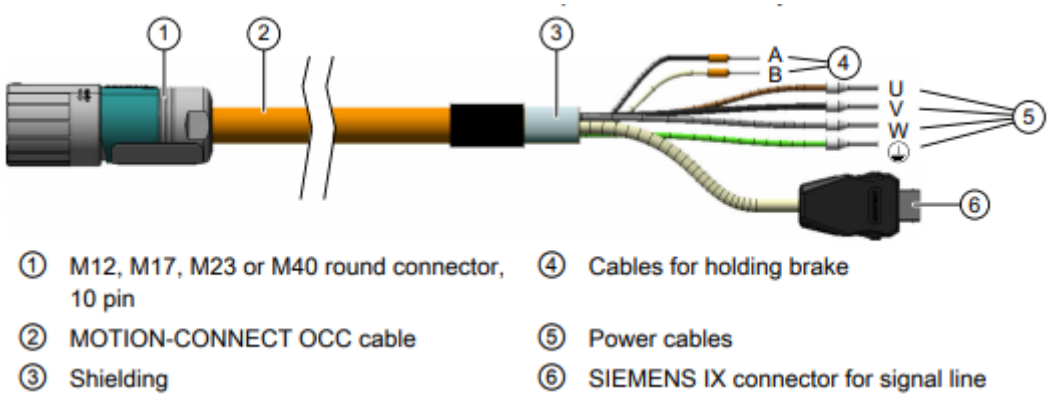


Kuva 2. Perusoperaatiot: (1) JA-portti, (2) TAI-portti, (3) Kiikku, (4) Laskuri ja (5) Ajastin

2.3 Servomoottorit

Servotekniikassa järjestelmän tehtävänä on ohjata takaisinkytkennän tai -kytkentöjen avulla haluttu suure järjestelmän ohjauksen antamaan arvoon. Takaisinkytkennästä mitattua lähtöarvoa verrataan järjestelmän ohjauksen arvoon, jonka perusteella moottoria ajetaan kohti uutta arvoa. Servot jaetaan säädettävän suureen mukaan asema-, nopeus-, voima- ja momenttiservoihin. (Fonselius, Rinkinen & Vilenius 1998, 7)

Servomoottorit ovat joko tasavirta- tai vaihtovirtamoottoreita. Ne koostuvat kolmesta eri osasta: moottori, asentoanturi ja takaisinkytkentä. Asentoanturin avulla saadaan tieto moottorin pyörimisnopeudesta sekä kulmasta. Nämä tiedot välitetään servovahvistimelle, jonka avulla käyttäjä voi muokata moottorin liikehdintää. Tiedonvälitys tapahtuu OCC:n avulla, joka mahdollistaa vähäisen johdotuksen, sekä helpottaa kommunikointia laitteiden välillä. Kuvassa 3 kaapelin toimintaperiaate.



Kuva 3. OCC:n toimintaperiaate (Siemens 2019, 150)

2.4 Servovahvistin

Servovahvistin on osa Servo Drive -järjestelmää, jonka avulla kontrolloidaan servomoottoria. Servovahvistimet on mahdollista käyttöönottaa joko PLC:tä tai verkkoselainta hyödyntäen. Ohjaussignaalin avulla vahvistin ohjaa moottoria, ja säätää moottorin pyörimisnopeutta suljetun takaisinkytkennän avulla. Vahvistin on yksiakselinen laite, jolle on ominaista kompakti muotoilu, vierekkäin tapahtuva asennus sekä korkea ylikuormituskyky. (Siemens 2019, 38)

3 Suunnittelu

3.1 Vaatimukset

Työ rajattiin yhdessä Tampereen ammattikorkeakoulun opettajien kanssa. Taulukossa 1 näkyy vaatimuslista, jonka sisältö määriteltiin sen mukaan, miten konttinosturin tulisi toimia. Vaatimukset on rajattu toiminnallisiin- ja ohjelmallisiin osioihin. Taulukon rajausten lisäksi tila konttinosturille oli rajallinen. Pienoismalli mitoitetaan 75cm x 180cm kokoiselle pöydälle.

Taulukko 1. Vaatimusmäärittely

Vaatimusmäärittely				
#	Ohjelmalliset vaatimukset	Vaatimuksen esittäjä	Tärkeys	Perustelu
1	Manuaali- ja automaattiajo	Asiakas	1	
2	Ohjelman soveltaminen tuleviin versioi	Asiakas	1	Ohjelman pystyttävä toimimaan myös useamman kerroksen kanssa
3	Ohjelman järkevä kommentointi	Asiakas	1	Muiden käyttäjien on ymmärrettävä kuinka koodi toimii
#	Toiminnalliset vaatimukset	Vaatimuksen esittäjä	Tärkeys	Perustelu
1	Hätäseis	Työntekijä	1	Turvallisen käyttämisen varmistaminen
2	Moottoreille rajakytkimet	Työntekijä	1	Moottoreille asetettava turvarajat, joiden yli ei pysty ajamaan
3	Konttinosturin mahduttava pöydälle	Asiakas / Ympäristö	1	Asiakas on määrittänyt käytettävissä olevan alueen
4	Kaksi eri konttikokoa	Asiakas	1	Kouran tulee havaita kumpi kontti on kyseessä
5	Lastauspaikkojen anturointi	Työntekijä	2	Ohjelman on saatava havaittua onko alueella konttia
6	HMI-paneeli	Asiakas / Työntekijä	2	Nosturia tulee pystyä ohjaamaan ja seuraamaan
	1-Pakollinen, 2-Hyödyllinen			

Päätavoitteena oli nosturin ylemmän tason ohjauksen suunnittelu, johon kuului nosturin liikkuminen, konttien siirto-operaatiot sekä käyttöliittymän suunnittelu. Käyttäjä voi kuljettaa kontteja lastausalueen ja pinoamisalueen välillä, tai pinoamisalueen sisällä manuaaliohjauksen avulla. Ehtoina ovat rajakytkimet, joiden avulla nosturia ei ajeta asetettujen rajojen yli, sekä hätäseis-painike, jolla käyttäjä voi pysäyttää konttinosturin kokonaan. Automaattiohjauksella käyttäjä valitsee vain siirrettävän kontin sekä ruudun, johon kontti kuljetetaan. Valintojen jälkeen ohjelman tulee liikkua itsenäisesti. Valinnat tehdään käyttöliittymältä. Ohjelman järkevä kommentointi, eli selitys koodin toiminnoista sen sisällä, on myös tärkeää. Kommentoinnilla varmistetaan, että koodin toimintojen suhteen ei tule sekaannuksia.

Toiminnalliset rajaukset liittyvät konttinosturin liikkumiseen sekä käytettävyyteen. Turvallinen ympäristö oli yksi tärkeimmistä rajauksista, ja se saavutettiin hätä-

seis-painikkeen ja rajakytkimien avulla. Konttinosturi on saatava pysäytettyä, mikäli vaaratilanne on syntymässä. Muut rajaukset eivät olleet pakollisia, mutta helpottivat suunnittelua ja työn tekemistä. Suunnitteluvaiheessa hahmoteltiin malliratkaisu, jonka perusteella kontit sijoitettaisiin pöydälle (liite 2). Käyttöliittymän Konttipaikkojen tila -näkymästä huomaa, että malli on pysynyt pitkälti samana (kuva 26).

3.2 Toimilaitteiden valinta

Toimilaitteiden valinta ei tuottanut vaikeuksia, sillä osa laitteistosta löytyi valmiiksi koululta. Moottorit sekä servovahvistimet, joilla moottoreita ohjataan, oli valmiiksi valittu. Logiikkana tuli aluksi käyttää Siemens Simaticin S7-300-sarjan logiikkaa, mutta työn edetessä huomattiin, ettei sen käyttö onnistunut servovahvistimien kanssa. Logiikka vaihdettiin uudempaan 1500-sarjan logiikkaan. Uuden logiikan malli on 1512SP F-1 PN.
Nosturia ohjataan KTP-600 paneelin avulla, johon luodaan käyttöliittymä. Paneeli kiinnitetään Ethernetin avulla ohjelmoitavaan logiikkaan. KTP-600 on kosketusnäytöllinen ohjauspaneeli, johon voi ohjelmoida eri painikkeita sekä laskureita.

Näiden lisäksi toimilaitteisiin kuuluu servovahvistin, jolla ohjataan servomoottoria. Servovahvistin näkyy kuvassa 4. Kuvan oikeassa laidassa on työssä käytettävä servomoottori Siemens Simotics 1FK2103-4AG00-1MAO. Koko järjestelmä on nimeltään Sinamics S210 Servo Drive.

Kuva 4. Servovahvistin ja servomoottori (Siemens Oy)

Antureita päätettiin käyttää molemmissa lastausalueissa, sekä radoilla, joilla konttinosturi kulkee. Lastausalueilla käytetään kapasitiivisia CAP-18NPN12 antureita, joita tulee yhteensä kuusi kappaletta. Kapasitiivisen anturit aktivoituvat, kun ne havaitsevat liikettä niiden tuntoetäisyyden edessä. Näiden avulla ohjelma saa tiedon lastausalueen konttien tiloista. Anturit saadaan kiinnitettyä M18 kierrekiinnityksen avulla lastausalueen pohjaan. Pinoamisalueelle ei asenneta antureita, sillä niiden konttimäärät nähdään ohjelman laskureista.

Liikeradan rajaukseen käytetään D2SW-3L2M rajakytkimiä, jotka asennetaan ratojen päähän. Kytkimet ovat mekaanisia, ja lähettävät signaalin logiikalle, kun konttinosturi on ajettu radan päätyyn asti. Rullavipu vaatii 0,59 Newtonia aktivoituakseen, joka vastaa noin 60 gramman painon kohdistumista vivulle.

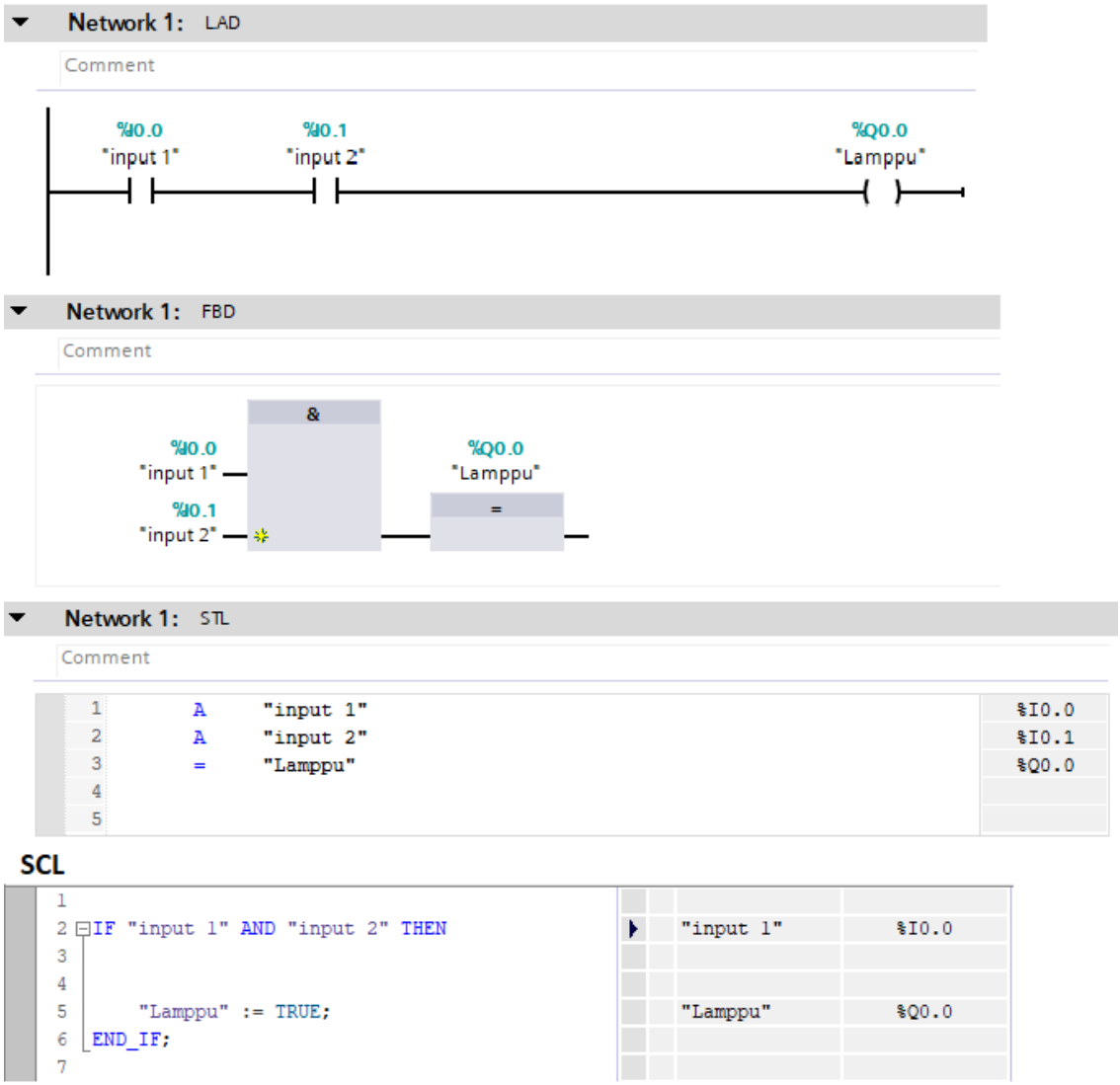
Tarttujiksi valittiin kaksi Feston EHPS tarttujaa (kuva 5). Kahden tarttujan avulla saadaan helpommin tieto nostettavan kontin koosta. Kontin koko eri ruudussa näkyy myös ohjelmasta.

Kuva 5. Festo EHPS tarttuja (Festo Oy)

Kaikkien yllä mainittujen toimilaitteiden käyttöjännite on 20-30Vdc. Muuntajan on katettava kaikkien laitteiden virrantarve, vaikka laitteita lisättäisiin projektiin myöhemmin. Muuntajaksi valikoitui MEAN WELL NDR-240-24, joka muuntaa 230 voltin verkkojännitteen 24 voltin tasasähköksi.

3.3 Ohjelmointikielen valinta

Ohjelmointikielen valinta riippuu yleensä tehtävästä, jonka ohjelman tulee suorittaa. LAD (Ladder Logic) tai FBD (Function Block Diagram) ovat hyviä ratkaisuja, jos ohjelma koostuu suurimmaksi osaksi binääreistä. Vaikeampiin tehtäviin kuten kompleksien muuttujien käsittelyyn ja niiden epäsuoraan osoittamiseen on vaihtoehtona STL (Statement List). SCL (Structured Control Language) on paras valinta heille, jotka ovat jo perehtyneet toiseen korkean tason ohjelmointikieleen. SCL:n avulla kirjoitetut ohjelmat voivat prosessoida suuria määriä dataa. (Berger 2012, 12)



Kuva 6. JA-portti eri kielillä: LAD, FBD, STL ja SCL

Kuvassa 6 näkyy yksinkertainen JA-portti, joka kuvastaa lampun syttymistä kahden sisääntulon seurauksena. Sisääntulojen Input 1 ja Input 2 on oltava samaan aikaan aktiivisia, jotta ulostulo Lamppu syttyy. Kuva demonstroi kielten eroavaisuuksia. FBD ja LAD muistuttavat enemmän tikapuukaavioita, kun taas STL ja SCL muistuttavat korkeamman tason ohjelmointikieliä. Työn ohjelmointikieleksi valikoitui FBD sen yksinkertaisuuden takia. Kieltä opetetaan jo muilla Tampereen Ammattikorkeakoulun kursseilla, joten konttinosturin voi ottaa opetuskäyttöön ilman uuden kielen opettamista.

4 PLC

4.1 I/O-luettelo ja muuttujat

Aluksi ohjelmaan määriteltiin PLC-tagit eli muuttujat. Muuttujille annetaan ainutlaatuinen absoluuttinen osoite, johon voidaan viitata koodin sisällä. Kaikkia muuttujia ei kuitenkaan tarvitse lisätä heti, sillä niitä voi ilmentyä lisää projektin edetessä.

Uutta muuttujaa lisätessä sille tulee määrittää nimi, datatyyppi ja osoite (kuva 7). Ensimmäinen muuttuja on nimeltään Lastausanturi_1. Se on data tyypiltään Bool eli Boolean, joten sillä voi olla kaksi eri arvoa, tosi tai epätosi. Muuttuja saa arvon tosi sen havaitessa kontin, muulloin se pysyy epätotena. Muuttujan osoite on %I136.0, joka näkyy Address-palkista.

PLC tags

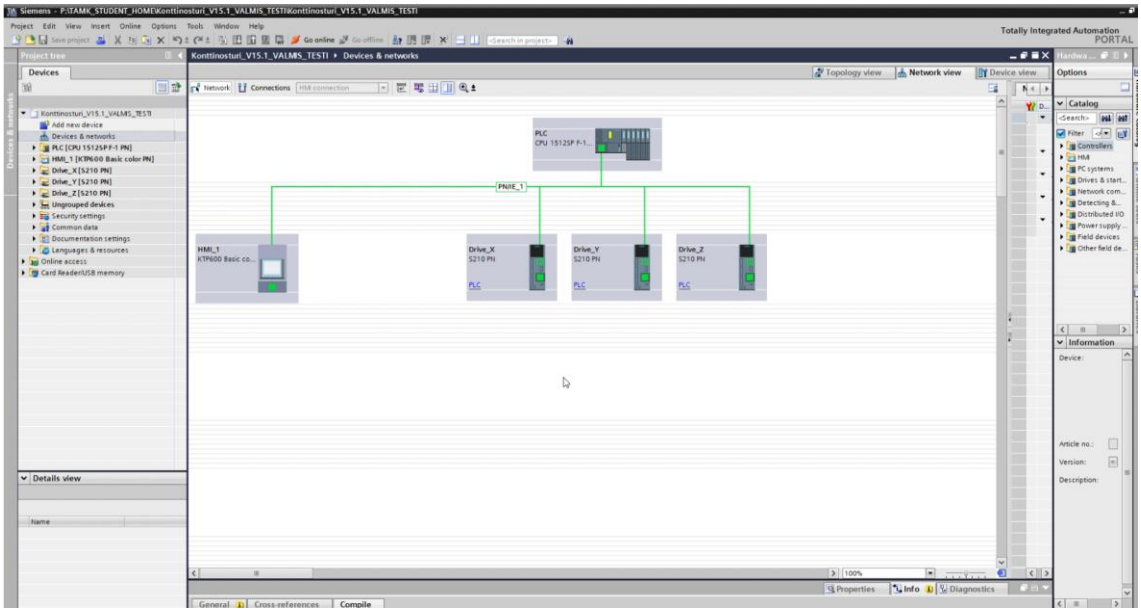
	Name	Tag table	Data type	Address	Retain	Acces...	Writa...	Visibl...	Supervis...	Comment
1	Lastausanturi_1	PLC Tags	Bool	%I136.0		☑	☑	☑		Tarkkailee lastausalueen tilaa
2	Lastausanturi_2	PLC Tags	Bool	%I136.1		☑	☑	☑		Tarkkailee lastausalueen tilaa
3	Lastausanturi_3	PLC Tags	Bool	%I136.2		☑	☑	☑		Tarkkailee lastausalueen tilaa
4	Lastausanturi_4	PLC Tags	Bool	%I136.3		☑	☑	☑		Tarkkailee lastausalueen tilaa
5	Lastausanturi_5	PLC Tags	Bool	%I136.4		☑	☑	☑		Tarkkailee lastausalueen tilaa
6	Lastausanturi_6	PLC Tags	Bool	%I136.5		☑	☑	☑		Tarkkailee lastausalueen tilaa
7	Kontin_Tunnistus	PLC Tags	Bool	%I136.6		☑	☑	☑		Tunnistetaan onko kontti gripperien kohd...
8	Kontin_Tunnistus_Koko	PLC Tags	Bool	%I136.7		☑	☑	☑		
9	Rajakytkin_1	PLC Tags	Bool	%I137.0		☑	☑	☑		Raja moottorin X miinus-liikkeelle
10	Rajakytkin_2	PLC Tags	Bool	%I137.1		☑	☑	☑		Raja moottorin X plus-liikkeelle
11	Rajakytkin_3	PLC Tags	Bool	%I137.2		☑	☑	☑		Raja moottorin Y miinus-liikkeelle
12	Rajakytkin_4	PLC Tags	Bool	%I137.3		☑	☑	☑		Raja moottorin Y plus-liikkeelle
13	Rajakytkin_5	PLC Tags	Bool	%I137.4		☑	☑	☑		Raja moottorin Z miinus-liikkeelle
14	Rajakytkin_6	PLC Tags	Bool	%I137.5		☑	☑	☑		Raja moottorin Z plus-liikkeelle
15	Vapaa(2)	PLC Tags	Bool	%I137.6		☑	☑	☑		
16	Hätäseis	PLC Tags	Bool	%I137.7		☑	☑	☑		
17	PositioningAxis_1_Actor_Interf...	Default tag table	"PD_TEL105_IN"	%I256.0		☑	☑	☑		
18	PositioningAxis_Y_Actor_Interf...	Default tag table	"PD_TEL105_IN"	%I276.0		☑	☑	☑		
19	PositioningAxis_Z_Actor_Interf...	Default tag table	"PD_TEL105_IN"	%I296.0		☑	☑	☑		
20	Kontin_lukitus	PLC Tags	Bool	%Q136.6		☑	☑	☑		Gripperit lukitsevat kontin paikoilleen
21	PositioningAxis_1_Actor_Interf...	Default tag table	"PD_TEL105_O...	%Q256.0		☑	☑	☑		
22	PositioningAxis_Y_Actor_Interf...	Default tag table	"PD_TEL105_O...	%Q276.0		☑	☑	☑		
23	PositioningAxis_Z_Actor_Interf...	Default tag table	"PD_TEL105_O...	%Q296.0		☑	☑	☑		
24	Moottori_X_-	Default tag table	Bool	%M0.0		☑	☑	☑		
25	Moottori_X_+	Default tag table	Bool	%M0.1		☑	☑	☑		
26	Moottori_Y_-	Default tag table	Bool	%M0.2		☑	☑	☑		
27	Moottori_Y_+	Default tag table	Bool	%M0.3		☑	☑	☑		
28	Moottori_Z_-	Default tag table	Bool	%M0.4		☑	☑	☑		
29	Moottori_Z_+	Default tag table	Bool	%M0.5		☑	☑	☑		

Kuva 7. Muuttujat

Ohjelmassa käytettiin Booleanin lisäksi myös INT ja TIME datatyyppejä. INT eli Integer antaa syöttää arvokseen kokonaislukuja. TIME datatyyppiä käytetään laskureissa ajan mittaamiseen. Aika on mahdollista esittää eri muodoissa, kuten päivissä, tunneissa tai sekunneissa.

4.2 Laitteisto

Laitteiston määrittelyyn pääsee vasemmalta löytyvästä valikosta (kuva 8). Lisättävä laite löytyy ohjelman katalogista, jonka jälkeen laite nimetään ja valitaan käytettävä versio. Laitteiden lisäyksen jälkeen ne yhdistetään toisiinsa PN/IE:n avulla (PROFINET/Industrial Ethernet). Yhteys saadaan klikkaamalla laitteen Ethernet porttia ja raahaamalla se toisen Ethernet portin päälle. Servovahvistimien asetuksista saa lisättyä oikean moottorin taulukon avulla. Kun moottori on valittu, ohjelma vaihtaa valitun moottorin arvot ohjelmassa olevaan teknologiaobjektiin.

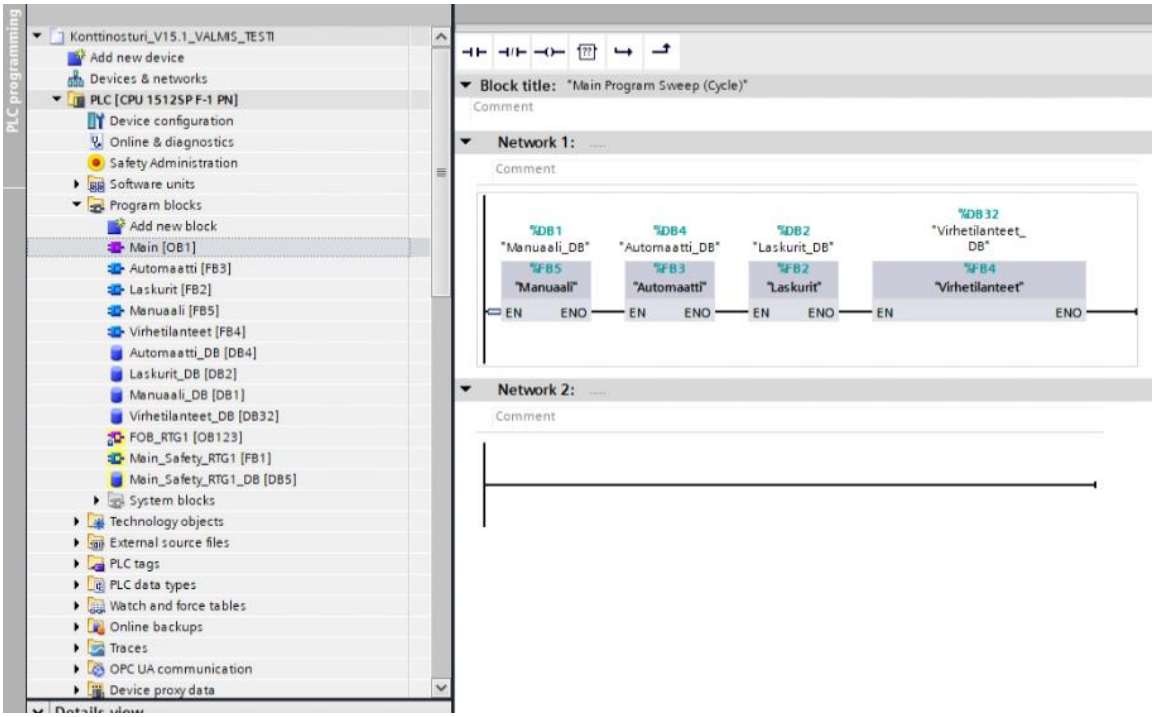


Kuva 8. Devices & Networks

Rakennetussa konttinosturissa laitteet ovat kytketty toisiinsa Ethernet kaapeleiden avulla. PLC on tilanteessa päälaite (sync master), jolla ohjataan servovahvistimia (slave). KTP-600 paneeli liitetään myös Ethernetin avulla logiikkaan, jonka kautta käyttöliittymän ohjelma ladataan paneeliin. Konttipaikkojen anturit ja mekaaniset rajakytkimet kytketään logiikan sisääntuloihin.

4.3 Ohjelma

Ohjelman rakentaminen alkoi jakamalle se pienempiin osa-alueisiin (kuva 9). Automaatti- ja manuaali-funktiolohkoissa ohjelmoidaan automaatti- ja manuaaliajo. Laskurit-funktiolohkossa on laskurit jokaiselle pinoamisruudulle. Kyseinen lohko muistaa myös minkä kokoiset kontit ruuduissa on. Virhetilanteet-funktiolohkossa on ratkaisut mahdollisille virhetilanteille konttinosturia ajettaessa. Main-organisaatiolohkossa kutsutaan kaikkia muita lohkoja.

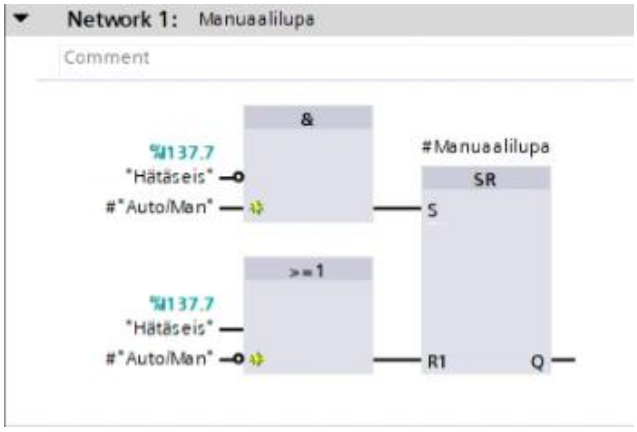


Kuva 9. Ohjelman rakenne

Main on organisaatiolohko (OB, Organization Block), joka muodostaa käyttöliittymän käyttöjärjestelmän ja ohjelman välille. Näiden avulla kutsutaan muita ohjelman lohkoja. Organisaatiolohkoilla voi suorittaa esimerkiksi ohjelmointi syklin tai käsitellä virheitä. Funktio (FC, Function) on lohko ilman muistia. Se suorittaa annetun tehtävän ilman parametrien tallentamista. Funktiolohko (FB, Function Block) on muistillinen lohko. Datalohko (DB, Data Block) toimii varastona ohjelman datalle. Globaali datalohkojen tietoja voi käyttää missä tahansa lohkossa. Lohkot koostuvat verkoista (network), joihin ohjelma rakennetaan operaatioiden avulla. Verkkojen avulla ohjelman saa jaettua vielä pienempiin osa-alueisiin, kuin lohkoilla.

4.3.1 Manuaaliajo

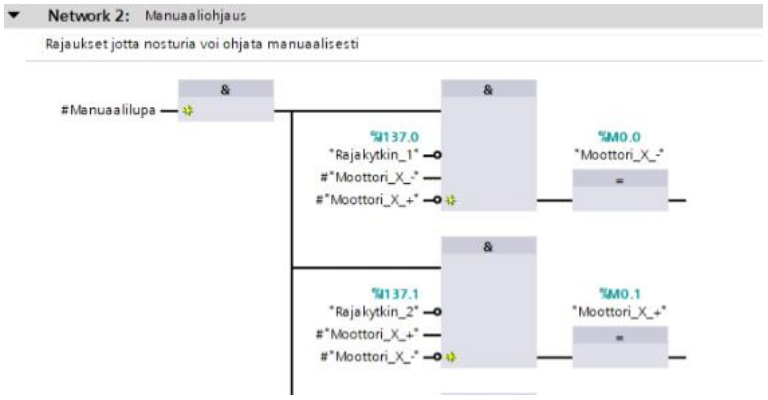
Manuaalilohkon ensimmäisessä verkossa tarkastetaan, onko käyttäjällä lupa käyttää manuaaliajoa (kuva 10). Manuaalilupa saadaan, kun hätäseis ei ole aktiivisena, ja tilaksi on vaihdettu manuaali. Auto/Man-bitti on Boolean-datatyyppi, joten sillä on kaksi tilaa. Epätosi on automaatti, ja tosi on manuaali. Bitin tilaa vaihdetaan HMI-paneelin painikkeiden avulla. Jos tila vaihtuu automaattiin tai hätäseis aktivoituu, manuaalilupa nollaantuu.



Kuva 10. Manuaalilupa

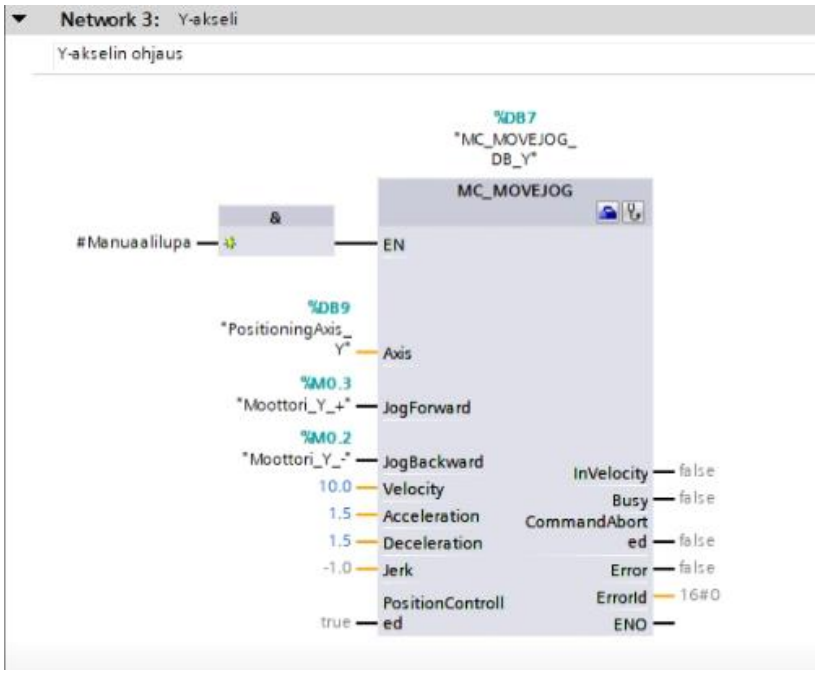
Toisessa verkossa varmistetaan ylimääräiset rajat jokaiselle liikkeelle (kuva 11). Moottoria X voi liikuttaa miinussuuntaan, kun seuraavat rajoitukset täyttyvät:

- Manuaalilupa on aktiivinen (Manuaalilupa)
- Rajakytkin ei ole aktiivinen (Rajakytkin_1)
- Moottori X saa miinusliikkeen HMI paneelilta (Moottori_X_-)
- Moottori X ei saa plusliikettä HMI paneelilta (Moottori_X_+)



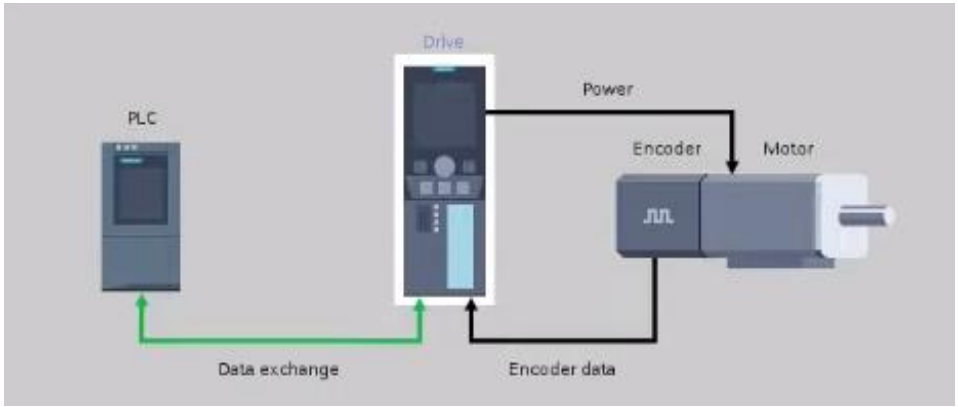
Kuva 11. Manuaaliohjaus

Kolmannessa verkossa ohjataan akselia MC_Movejog -ohjauslohkon avulla (kuva 12). Lohko aktivoituu (EN, Enable), kun manuaalilupa on aktiivinen. Axis-kohdassa valitaan liikutettava akseli. JogForward- ja JogBackward-sisääntulot ovat liikkeet eteen ja taakse, ja niihin valitaan moottorin plus- ja miinusliikkeet. Lohkossa voi myös valita nopeuden, kiihtyvyyden sekä hidastuvuuden. Moottoreiden ohjaus on samanlainen kaikille suunnille.



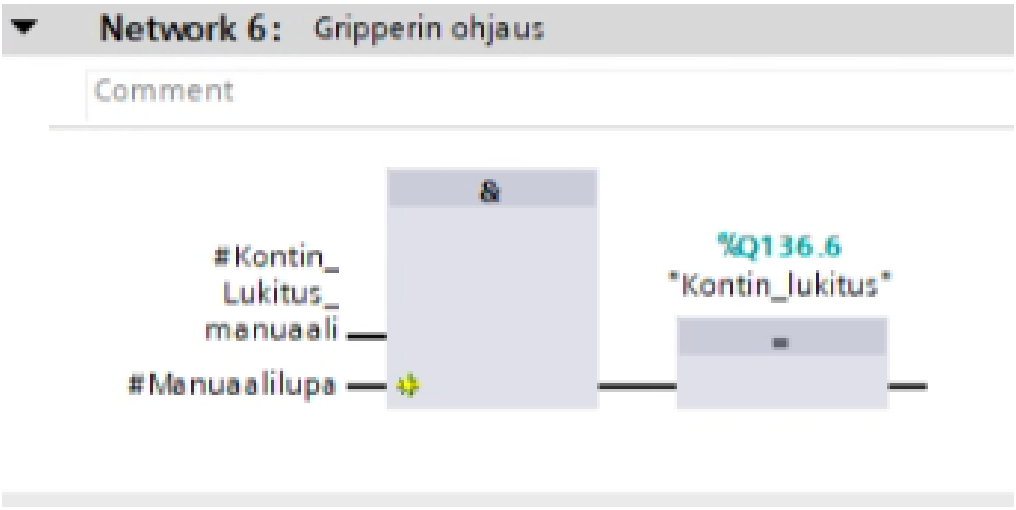
Kuva 12. Y-akselin ohjaus

Kun lohko aloittaa moottorin liikuttamisen, sen näkee akselin käyttöönotto -sivulta (comissioning). Käytännössä servovahvistimet saavat tiedon ohjelmalta, joiden avulla moottoria liikutetaan haluttuun suuntaan. Kuva 13 on havainnollistaa, kuinka tieto liikkuu laitteelta toiselle. Lohkon tehdessä liikkeen, moottori liikuttaa nosturin yhtä akselia valittuun suuntaan.



Kuva 13. Tietoliikenne laitteiden välillä

Viimeisenä verkkona on tarttujan ohjaus (kuva 14). Manuaaliluvan ja kontin lukitus on oltava aktiivisena, jotta tarttuja sulkeutuu. Lukitusta ohjataan HMI-paneelin painikkeella, joka vaihtaa Kontin_Lukitus_manuaali -bitin tilaa.



Kuva 14. Tarttujien manuaaliohjaus

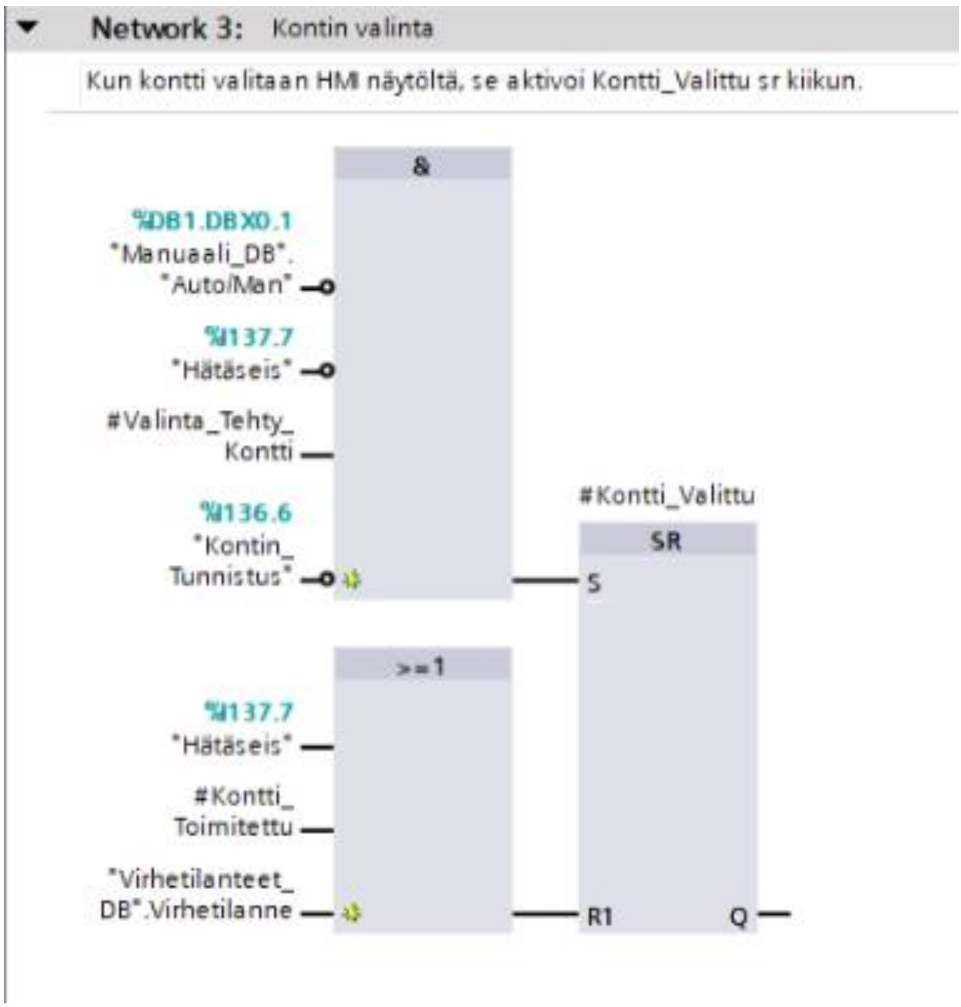
4.3.2 Automaattiajo

Automaattisykli luotiin teknologiaobjekteja käyttäen. Ensimmäisessä verkossa annetaan virta jokaiselle moottorille MC_Power -lohkolla. Moottorit liikkuvat vain niiden ollessa aktiivisena. Samaan verkkoon on lisätty teknologiaobjektien resetointilohko, jonka avulla voi nollata objektien hälytykset.

Ennen syklin aloittamista nosturi tulee olla kotiutettuna. MC_Home -lohkon avulla nosturi ajetaan ennalta määriteltyyn kotipisteeseen, jonka jälkeen moottorit tietävät fyysisen olinpaikkansa. Kotiutuksen jälkeen moottoreilla pystyy suorittamaan absoluuttista paikoitusajoa. Kotiutus_suoritettu -bitti aktivoituu, kun viimeinen liike kotiutuksesta on tehty.

Seuraava vaihe on liikutettavan kontin valinta (kuva 15). Kontti on valittuna, kun seuraavat rajaukset ovat täyttyneet:

- Automaattitila on aktiivinen (Auto/Man)
- Hätäseis ei ole aktiivinen (Hätäseis)
- Valinta on tehty HMI paneelilta (Valinta_Tehty_Kontti)
- Tarttujalla ei ole konttia kyydissä (Kontin_Tunnistus)



Kuva 15. Kontin valinta

Kontin valinta nollaantuu, jos jokin seuraavista rajauksista aktivoituu:

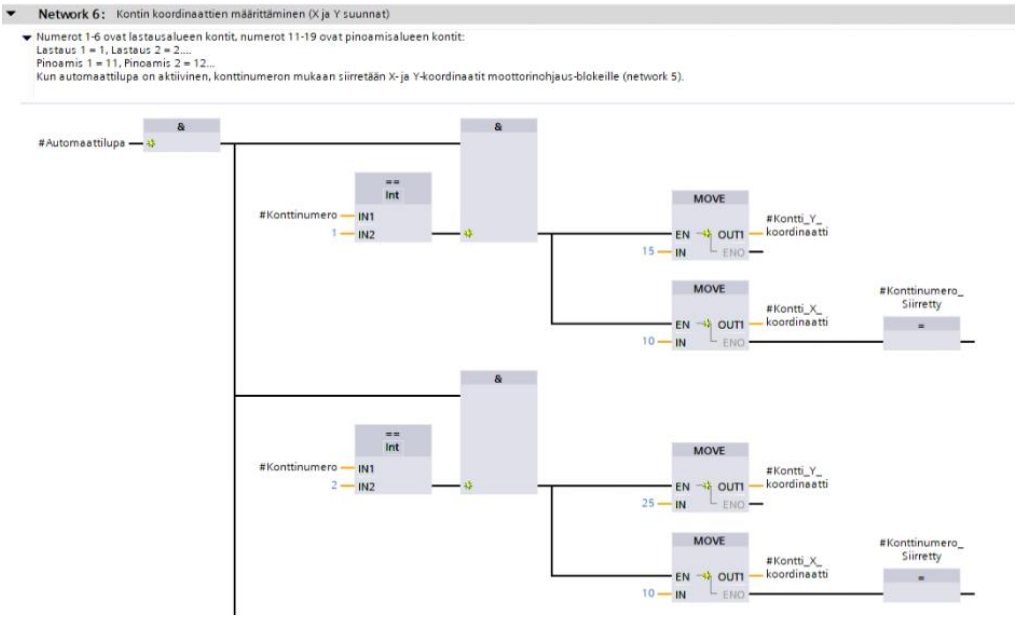
- Hätäseis aktivoituu (Hätäseis)
- Kontti toimitetaan paikoilleen (Kontti_Toimitettu)
- Syntyy virhetilanne (Virhetilanteet_DB.Virhetilanne)

Virhetilanteen nollauksessa suoritetaan tarvittavat korjausliikkeet, ennen kuin nosturia voi käyttää uudelleen. Nollaus tapahtuu myös syklin jälkeen.

Kontin valinnan jälkeen valitaan ruutu, johon kontti siirretään. Ruudun valinta suoritetaan samalla tavalla kuin kontin valinta. Kun valinnat on tehty, bitit muuttuvat aktiiviseksi ja antavat käyttäjälle automaattiluvan. Automaattiluvan aktivoiduttua siirretään oikeat koordinaatit teknologiaobjekteille. Kuvassa 16 on esimerkki, kuinka koordinaatit siirretään. Konttinumero on muuttuja, jonka avulla tiedetään mikä kontti on valittuna. Käyttäjän valitsema kontti HMI-paneelilta muuttaa bitin

arvon oikeaksi ruudun sijainnin perusteella. Esimerkiksi, kun siirrettävä kontti sijaitsee lastausalueen toisessa ruudussa, konttinumeroksi muuttuu kaksi. Koordinaatit siirretään eteenpäin ohjelmassa, kun automaattilupa sekä oikea konttinumero ovat molemmat aktiivisena. Valitun ruudun koordinaatit siirretään vastaavalla tavalla.

Siirto tapahtuu MOVE-lohkolla, joka siirtää sisääntulossa IN sijaitsevan luvun ulostulossa OUT1 olevaan bittiin. Koska konttipaikkoja on useampi, muuttujan datatyyppi on integer.



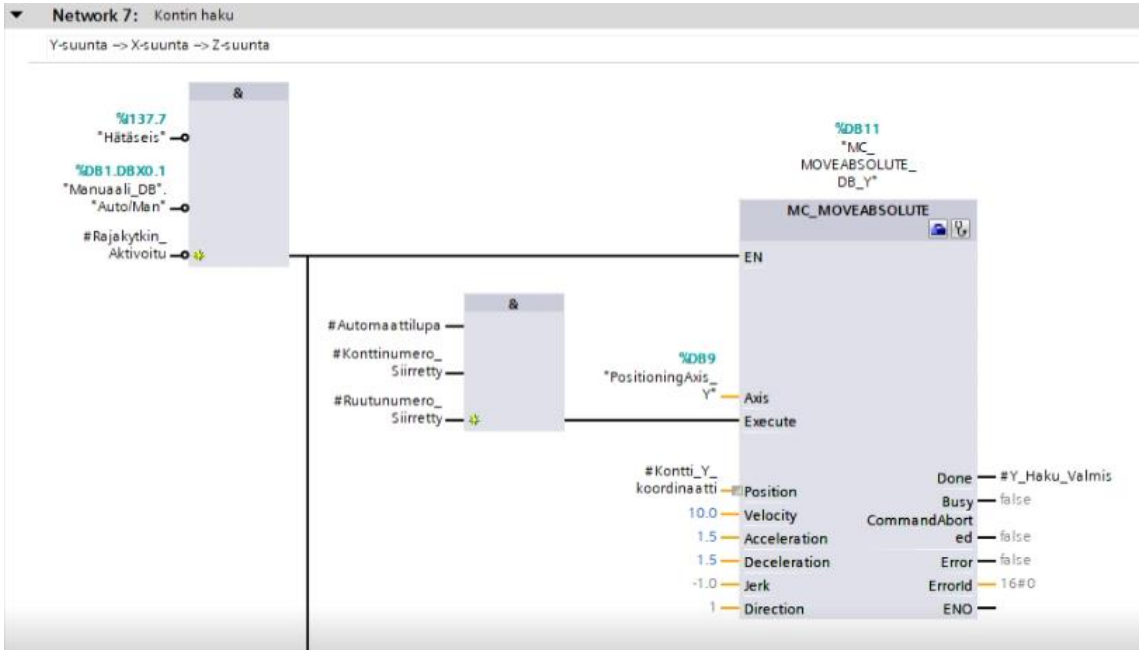
Kuva 16. Koordinaattien määrittely

Koordinaattien määrittelyn jälkeen aloitetaan kontin haku (kuva 17). Liikkeisiin käytetään MC_MoveAbsolute -lohkoja, joiden avulla moottoreita voidaan liikuttaa tiettyyn pisteeseen. Lohkot aktivoituvat, kun tilaksi on valittu automaatti, hätäseis ei ole painettuna ja rajakytkimet eivät ole lauenneet. Akseleita liikutetaan yksi kerrallaan.

Lohkon sisääntulo Execute antaa käskyn suorittaa liike, joka toteutuu, kun

- Automaattilupa on aktiivinen (Automaattilupa)
- Konttinumero on siirretty (Konttinumero_Siirretty)
- Ruutunumero on siirretty (Ruutunumero_Siirretty)

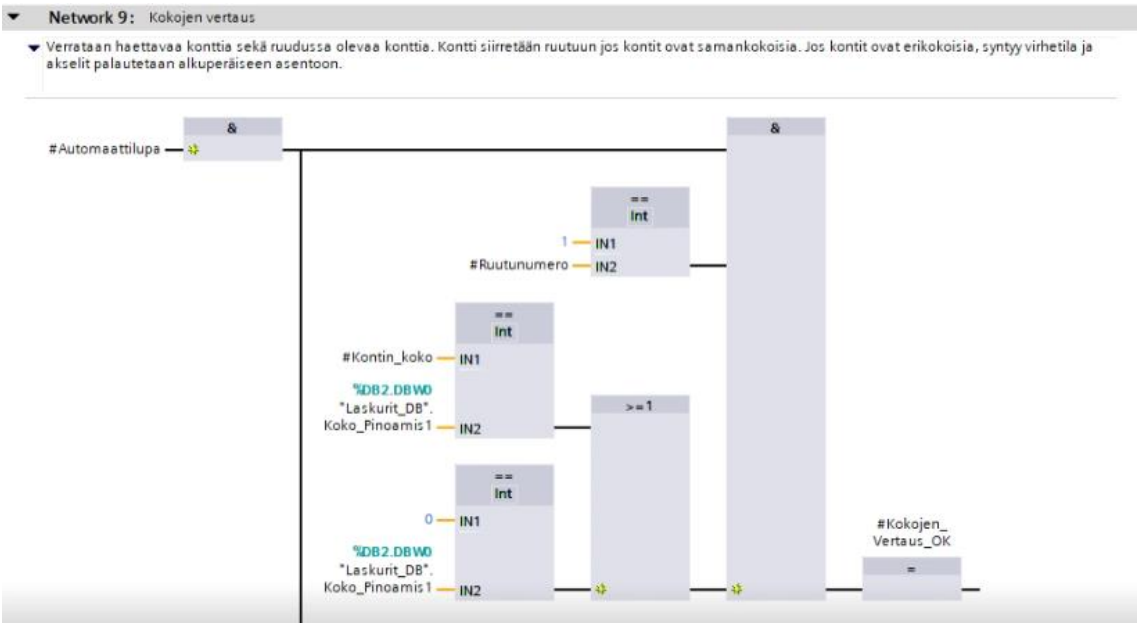
Position-sisääntulo määrittää mihin pisteeseen moottori ajetaan. Lohkon muihin sisääntuloihin saa määritettyä moottorin liikehdintään olennaisia arvoja. Kun valittu piste on saavutettu, lohkon ulostulo Done aktivoituu. Tämä ulostulo aktivoi seuraavan akselin liikkeen.



Kuva 17. Kontin haku

Kontin haun jälkeen suoritetaan kontin tunnistus. Kahden tarttujan avulla saadaan tieto kontin koosta. Pitkään konttiin osuvat molemmat tarttujat, ja pieneen vain toinen. Kontin_koko -muuttujaan siirretään kokoa vastaava numero. Koko on pieni muuttujan arvon ollessa yksi, ja iso sen ollessa kaksi.

Ennen kontin nostoa kokoa tulee verrata ruudussa olevaan konttiin (kuva 18). Kokojen vertaus on onnistunut, kun kontin koko on sama kuin ruudussa tai ruudussa ei ole vielä konttia. Jos kumpikaan rajauksista ei toteudu, syntyy virhetilanne ja akselit palautetaan alkuperäiseen asentoon, ja valinnat nollataan.

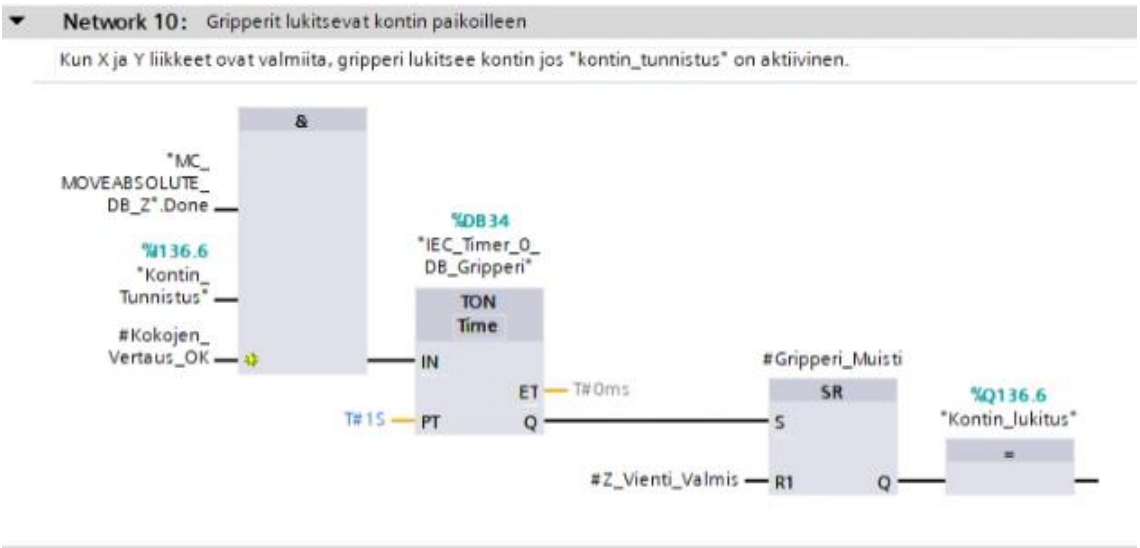


Kuva 18. Kokojen vertaus

Tarttujat toimivat ajastimen avulla (kuva 19). Ne lukitsevat kontin paikoilleen vasta, kun:

- Kontin haku -liikkeet ovat valmiita (MC_MoveAbsolute_DB_Z.Done)
- Tarttuja tunnistaa kontin (Kontin_Tunnistus)
- Konttikokojen vertaus on onnistunut (Kokojen_Vertaus_OK)

Kun rajaukset ovat täyttyneet, ajastin käynnistyy. Ajastimen tarkoitus on hidastaa sykliä, jotta tarttujat ovat oikealla kohdalla konttiin nähden. Ajastin on ON-delay ajastin, joka sisääntuloon PT asetetun ajan jälkeen aktivoi ulostulon Q. Tarttujat pysyvät kiinni, kunnes kontin vienti on valmis, ja tarttujat saavat nollaamiskäskyn.



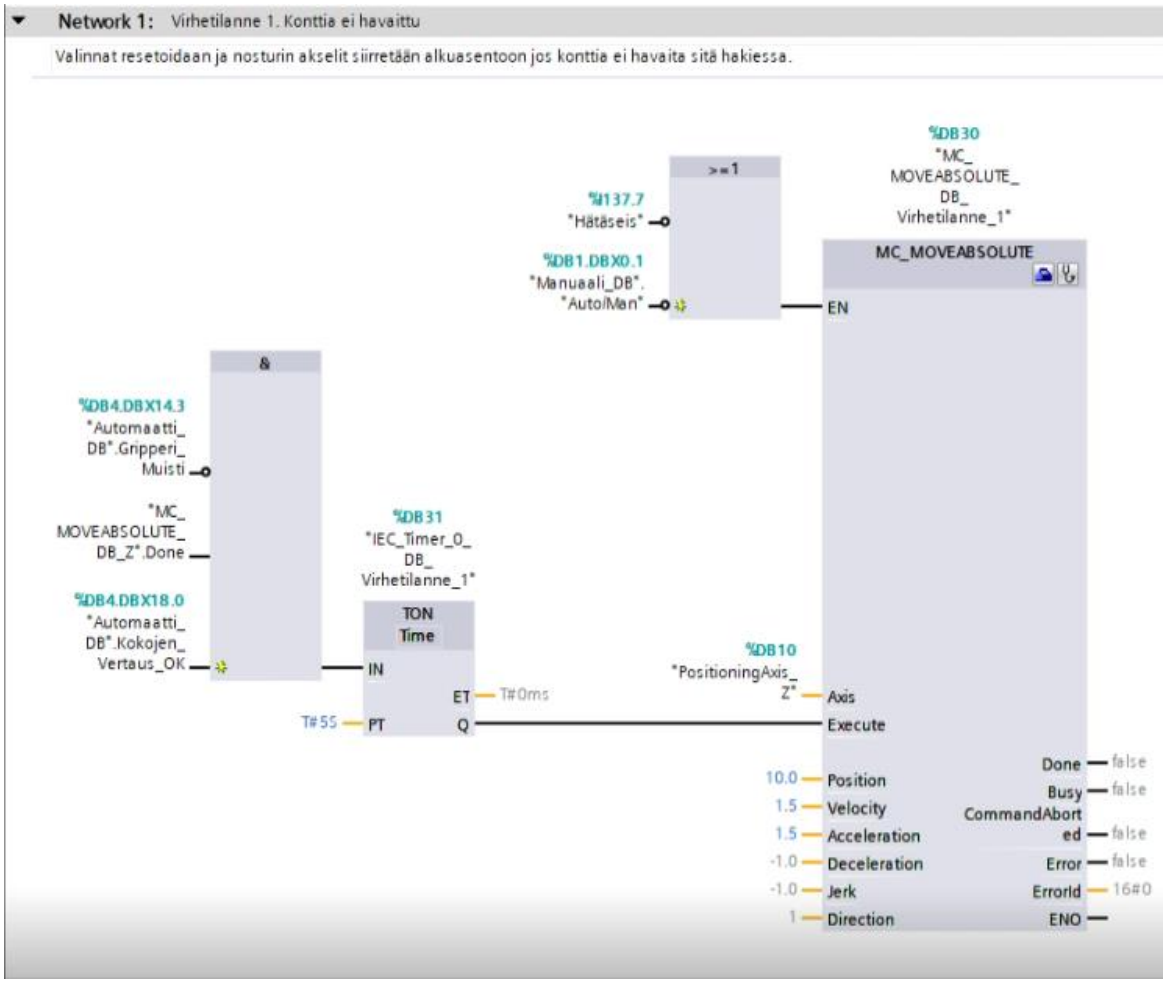
Kuva 19. Tarttujien automaattiohjaus

Kontin kuljetus valittuun ruutuun alkaa, kun tarttujat ovat kiinnittäneet kontin paikoilleen. Kontin kuljetus tapahtuu samalla periaatteella kuin kontin haku. Akseleita liikutetaan yksi kerralla, ja seuraava aktivoituu edellisen oltua valmis. Koko syklin valmistuttua Kontti_Toimitettu -bitti saa positiivisen reunan, joka nollaa kontin ja ruudun valinnat.

4.3.3 Virhetilanteet

Virhetilanteet-lohkossa on erilaisia skenaarioita, jotka voivat aiheuttaa konttinosturin toimintahäiriöitä. Ensimmäisessä verkossa nosturi ei havaitse konttia, ja palauttaa nosturin takaisin ylös (kuva 20). MC_Moveabsolute -lohko aktivoituu, kun automaattitila on valittu, ja kun hätäseis ei ole aktiivinen. Automaattitilan lisäys aktivoimiseen varmistaa, että manuaalitilaa voi käyttää ilman virhetilojen aktivoitumista. Lohko suorittaa korjausliikkeen, jos seuraavat rajaukset täyttyvät viiden sekunnin ajan:

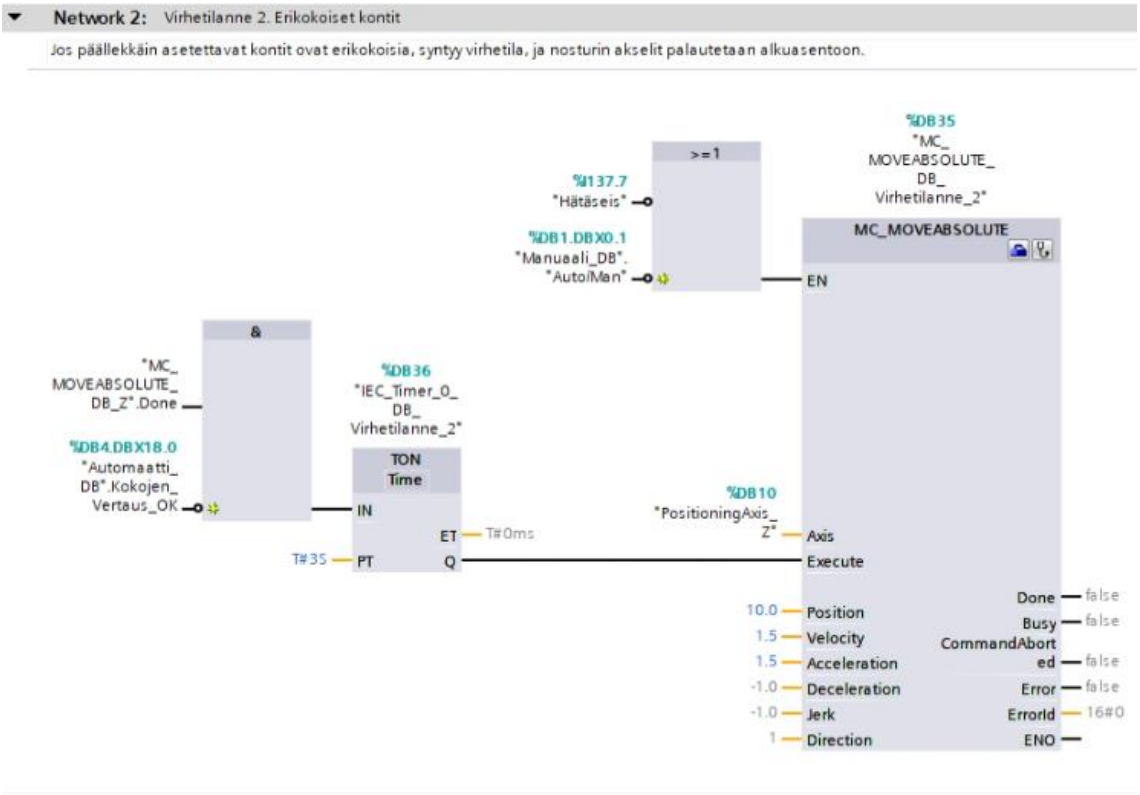
- Tarttuja ei ole mennyt kiinni (Gripperi_Muisti)
- Kontin hakuliike on suoritettu (MC_Moveabsolute_DB.Z.Done)
- Konttikoko oli oikea (Kokojen_Vertaus_OK)



Kuva 20. Virhetilanne 1

Toisessa virhetilanteessa suoritetaan vertailua konttikokojen kesken (kuva 21). Lohko aktivoidaan samalla tavalla kuin edellisessä virhetilanteessa, mutta korjausliikkeen aktivoimiseen on eri rajaukset. Korjausliike suoritetaan, kun seuraavat rajaukset täyttyvät:

- Kontin hakuliike on suoritettu (MC_Moveabsolute_DB.Z.Done)
- Konttikoko ei ollut oikea (Kokojen_Vertaus_OK)
- Yllä listatut rajaukset ovat aktiivisena 3 sekuntia

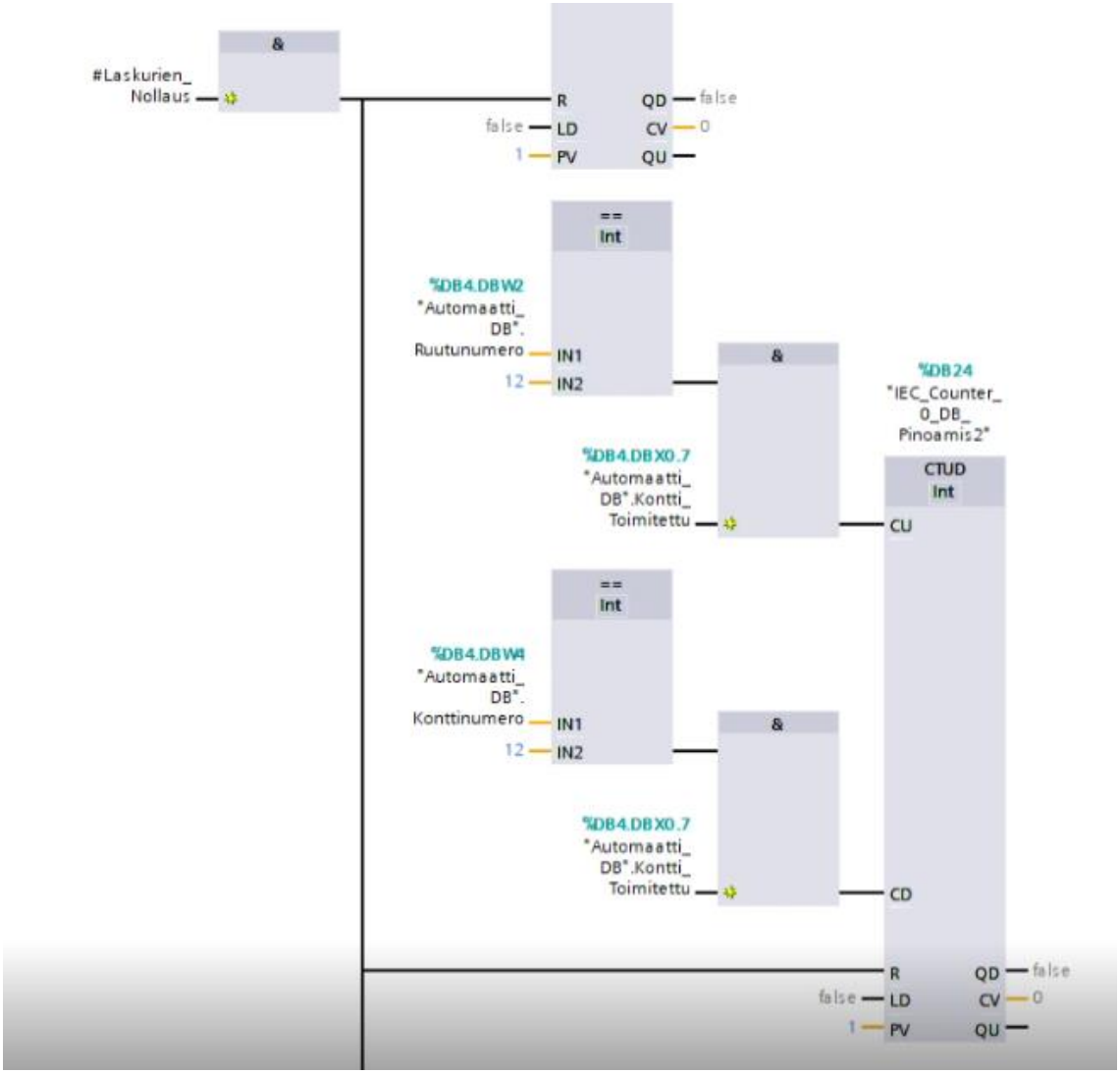


Kuva 21. Virhetilanne 2

Virhetilanteisiin on lisätty myös rajakytkimien aktivoituminen. Jos automaattiajon aikana jokin kytkimistä aktivoituu, liikkeet jätetään kesken, ja kahden sekunnin kuluttua kaikki akselit ajetaan keskipisteeseen. Nosturi täytyy kotiuttaa uudestaan ennen käyttöä, mikäli tämä virhetila aktivoituu.

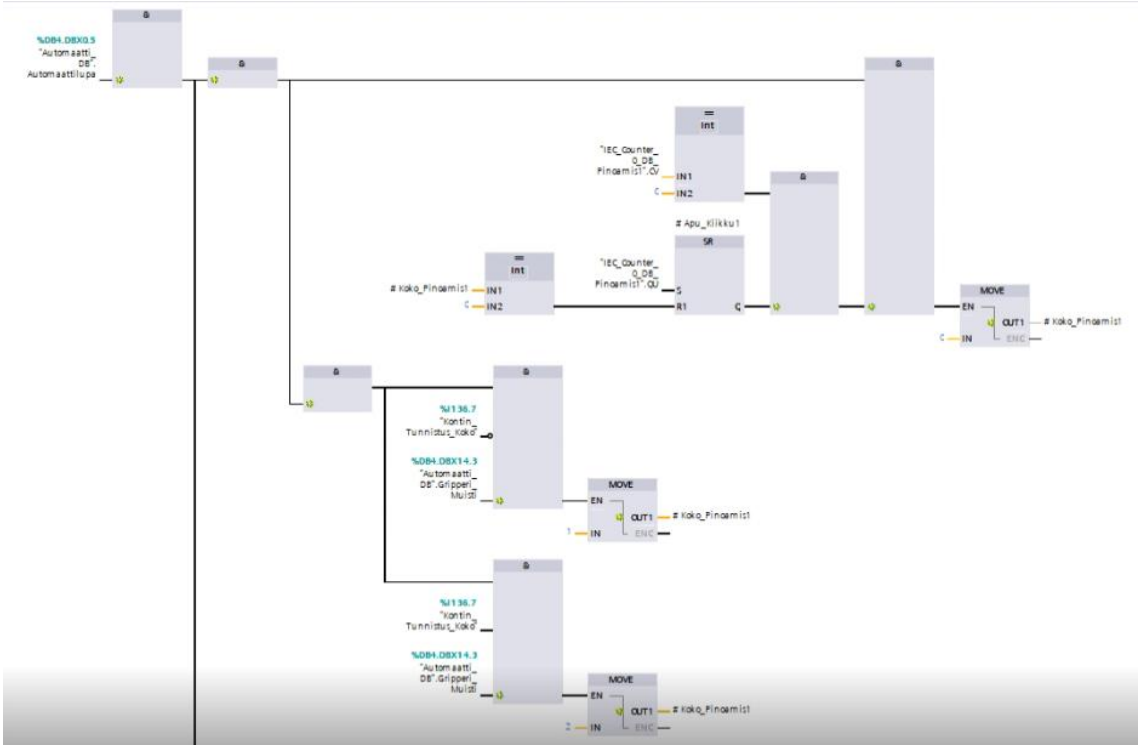
4.3.4 Laskurit

Laskurit-funktiolohkossa ohjelma pitää muistissa konttien määrät eri ruuduissa (kuva 22). Laskurin arvo nousee yhdellä, jos ruudusta valitaan kontti lähteväksi, ja kontti on toimitettu ruudusta pois. Arvo laskee, jos kontti valitaan vietäväksi ruutuun, ja kontti toimitettiin sinne. Laskureiden nollaus tapahtuu HMI-paneelilta.



Kuva 22. Laskurit

Verkossa kaksi muistetaan konttien koot eri ruuduissa (kuva 23). Jokaisella pinoamisruudulla on oma muuttuja, jonka avulla muistaminen tapahtuu. Nolla ilmoittaa, ettei ruudussa ole konttia, yksi kertoo kontin olevan pieni, ja kaksi kertoo kontin olevan suuri. Koon vaihtuminen yhdeksi tai kahdeksi vaatii tarttujien kiinniasennon, sekä tunnistuksen tarttujilta kumpi kontti on kyseessä.



Kuva 23. Konttikoon muuttaminen

Konttikoon muuttuminen nollaksi vaatii enemmän rajauksia. Muuttamisessa käytetään apuna kiikkua, jonka avulla estetään muuttujan turha vaihtuminen. Tässä osassa koodia käytetään ylimääräisiä JA-portteja, jotka eivät ole olennaisia koodin kannalta. Portteja käytetään koodin erotteluun helppolukuisuuden vuoksi.

5 Käyttöliittymä

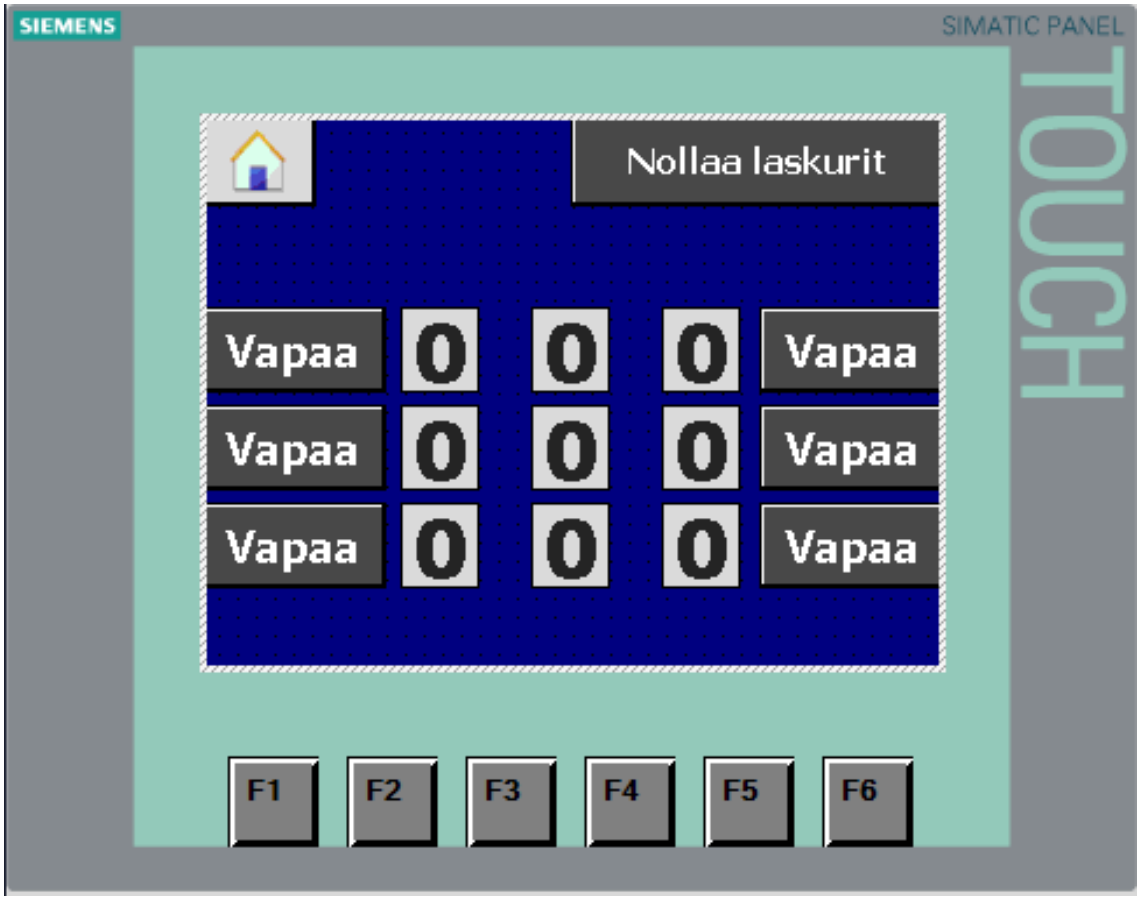
Konttinosturin ohjaus hoidetaan HMI-paneelin avulla. Paneeli on Siemensin KTP-600, johon ohjelmoidaan useampi näyttö. Näyttöjä on kuusi erilaista, joilla jokaisella on oma tehtävänsä. Kuvassa 26 näkyy kotinäyttö, joka on alkunäkymä laitetta käynnistäessä. Jokaisen näytön vasempaan yläkulmaan on sijoitettu Kotinäyttö-painike, jonka avulla pääsee takaisin aloitusnäyttöön. Kotinäytöltä valitaan joko automaatti- tai manuaalitila, joista pääsee ohjaamaan konttinosturia.

Kuva 24. Kotinäyttö

Näytön oikeassa ylänurkassa on keltainen virhetilanteiden Reset-painike, joka nollaa lohkoissa olevat viat ja palauttaa ne alkuperäisiin asetuksiin. Vasemmalla reunalla on Kotiutus-painike, jonka avulla konttinosturin akseleille määritellään kotipiste. Ilman kotiuttamista automaattitilaa ei voi käyttää.

Kotinäytöllä on myös Konttipaikkojen tila -painike, jonka avulla voidaan seurata, kuinka monta konttia tietylle ruudulle on kasattu (kuva 27). Ruudun reunoilla näkyy myös lastausalueiden tila. Jos lastausalueella ei ole konttia, käytön aikana

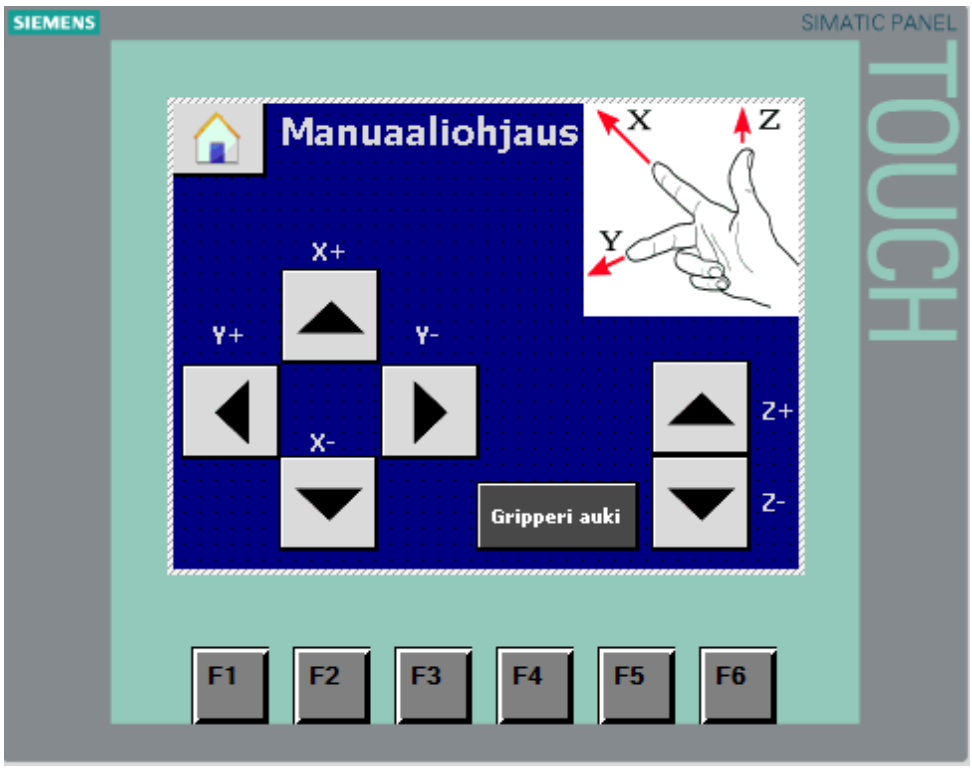
näytöllä näkyy vihreä Vapaa-ruutu. Jos lastausalueella on kontti, ruutu muuttuu punaiseksi ja tekstikenttä vaihtuu varatuksi.



Kuva 25. Konttipaikkojen tila

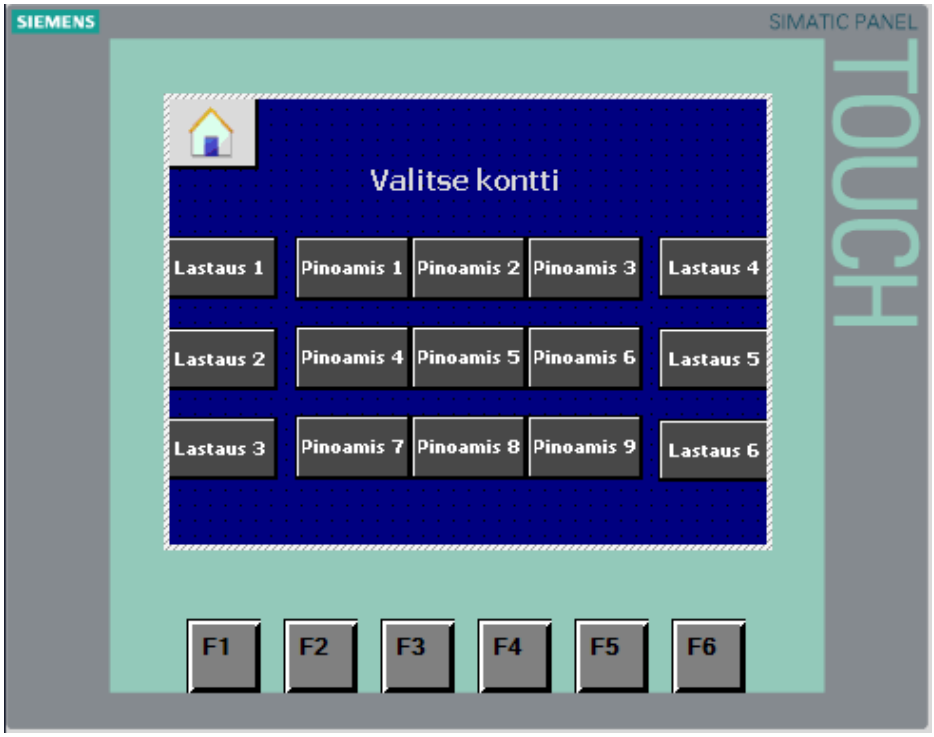
Lastausalueen tiedot saadaan antureiden avulla, mutta pinoamisalueen tiedot muistetaan koodissa olevien laskureiden avulla. Laskureiden resetointi tapahtuu Nollaa laskurit-painikkeella näytön oikeassa ylänurkassa.

Kuva 28 on näkymä manuaaliohjauksesta. Manuaaliohjaus-näytölle pääsee painamalla kotinäytön Manuaaliohjaus-painiketta. Konttinosturia voi liikuttaa painikkeiden avulla X-, Y- tai Z-akselien suuntaan. Painikkeiden vieressä on tekstiruudut, jotka indikoivat mihin suuntaan nosturi liikkuu painiketta painettaessa. Painikkeiden välissä on myös tarttujien ohjaus. Tarttujan ohjaaminen on toteutettu katkaisijan avulla, joka ilmoittaa tarttujan olevan kiinni tai auki värien sekä tekstin avulla. Oikeaan ylänurkkaan on sijoitettu havainnollistava kuva akseleista, joka helpottaa konttinosturin ohjaamista.



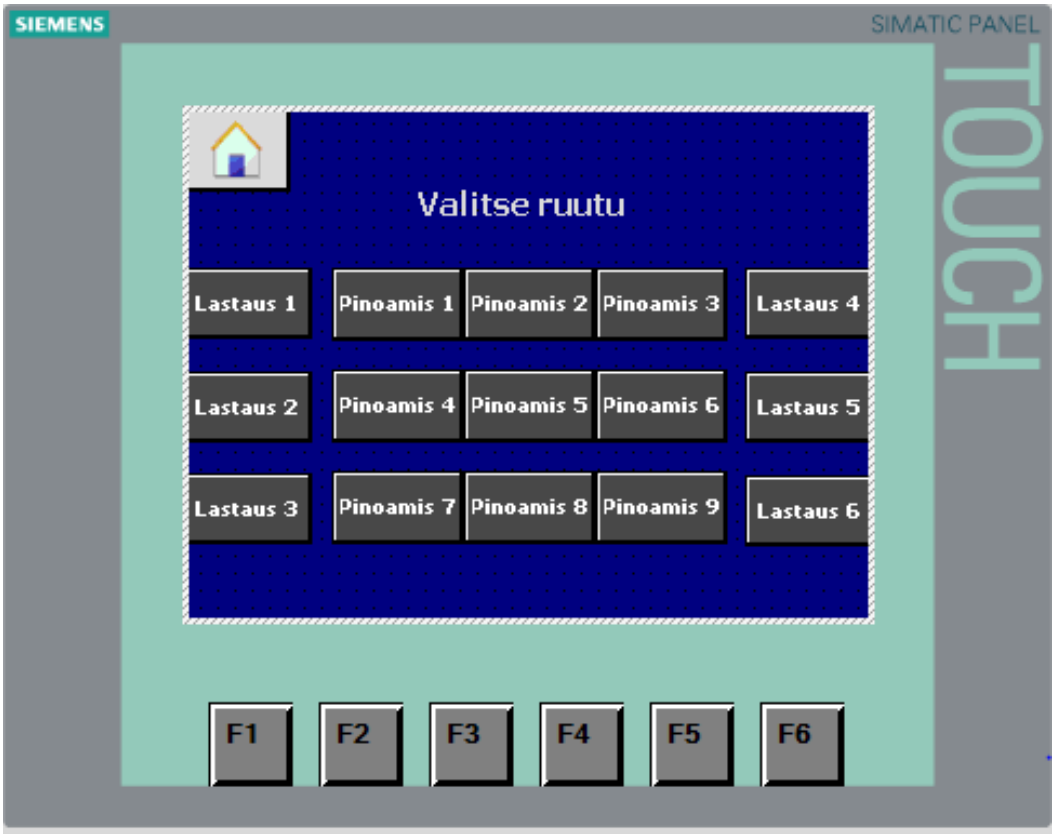
Kuva 26. Manuaaliohjaus

Kotinäytön Automaatti-painikkeella pääsee ohjaamaan konttinosturia automaattisesti. Automaattitilassa valitaan siirrettävä kontti (kuva 29) sekä ruutu (kuva 30) johon kontti siirretään. Valintojen jälkeen näyttö ilmoittaa kuljetuksen olevan käynnissä (kuva 31).



Kuva 27. Kontin valinta

Konttia valittaessa näyttö näyttää ruudut eri väreillä. Värit määräytyvät konttimäärien mukaan. Painike on vihreä jos ruudussa ei ole konttia, keltainen jos kontteja on yksi tai kaksi, ja punainen jos kontteja on kolme. Käyttäjä ei voi valita ruutua jossa on jo kolme konttia.



Kuva 28. Ruudun valinta

Kun kontti on kuljetettu paikoilleen, ponnahtaa näytölle Valmis-painike, jota painamalla pääsee takaisin Kontin valinta -näytölle. Virhetilojen aktivoituessa näytölle avautuu virhetila ilmoitus. Virhetilan voi resetoida korjausliikkeen oltua valmis. Tällöin paneelille ilmestyy Palaa takaisin -painike. Painikkeet ovat esitettyinä kuvassa 31.

Kuva 29. Kuljetus käynnissä

Hätäseis-painikkeen aktivoituessa paneelille tulee punainen ilmoitus. Painike resetoi virhetilan, jos hätäseis ei ole enää aktiivisena. Sama ilmoitus näkyy jokaisella näytöllä, eikä se poistu ennen nollaamista (kuva 32).

Kuva 30. Hätäseis-ilmoitus

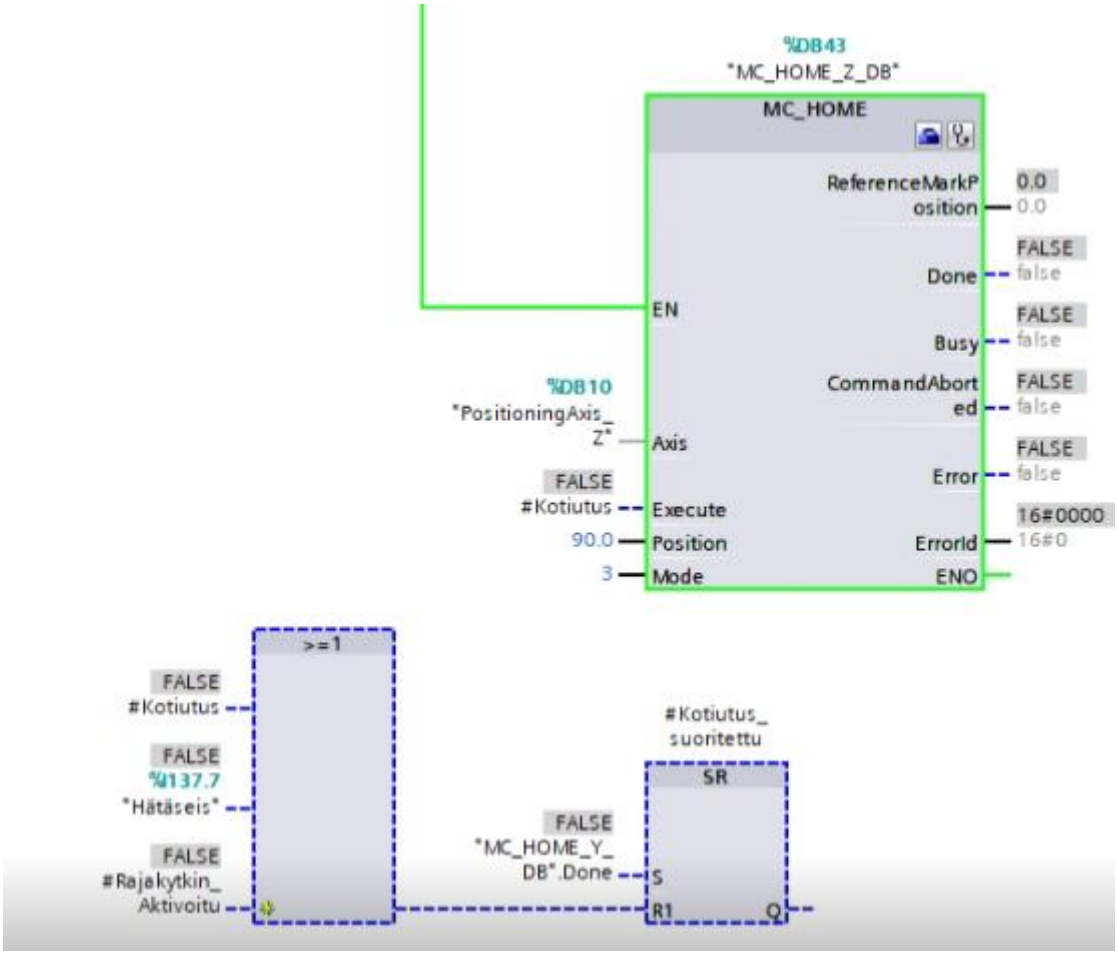
6 Simulointi

6.1 Teoria

Simulointi on ohjelman testausta ilman fyysisiä laitteita, eli virtuaalisesti. Ohjelma ladataan virtuaaliselle logiikalle, josta voi ohjata ohjelman muuttujia. Virtuaalisella PLC:llä voi myös suorittaa sekvenssejä, joiden avulla voidaan simuloida oikeaa tilannetta. Kuvassa 24 on alkunäkymä Siemensin Tia Portaalin virtuaalilogiikasta. Suurennetun näkymän saa esille, kun painaa logiikan oikeassa yläreunassa olevaa laajenna-painiketta.

Kuva 31. Virtuaali-PLC

Työtä simuloidessa kytketään päälle monitorointi. Monitoroinnin ollessa aktiivinen, ohjelman yläreuna muuttuu oranssiksi. Tämän avulla on helpompi tulkita, mitä koodin sisällä tapahtuu. Aktiiviset lohkot ja muuttujat saavat tällöin vihreän värin ympärilleen. Kuvasta 25 näkyy kotiutuslohkon olevan aktiivinen, kun hätäseis ei ole päällä. Lohko ei kuitenkaan ole vielä liikuttanut moottoria, joten kaikki ulostulot ovat epätosia. Kun lohko ei ole aktiivinen, sen ympärillä on siniset katkoviivat.



Kuva 32. Simulointi

6.2 Työn simulointi

Ohjelmaa päätetiin testata simuloinnin avulla, koska konttinosturia ei ollut vielä rakennettu. Tämä hankaloitti koodin toimivuuden toteamista. Simuloinnin ajaksi koodiin lisättiin erilaisia apumuuttujia, joiden avulla päästiin haluttuun lopputulokseen. Simuloidessa huomattiin, ettei koodi joissain tapauksissa toiminut halutulla tavalla, ja tarvittavat muutokset lisättiin. Osassa ongelmia riitti negaation lisääminen, jolloin bitti toimii päinvastaisesti. Osaan koodista jouduttiin tekemään isompia muutoksia, kuten konttikokojen seurantaan. Koodiin lisättiin apukiikkuja, jotka varmistavat, ettei koodi vaihda konttikokoa väärään aikaan.

Simuloinnissa moottoreita pystyttiin ajamaan manuaali- sekä automaattitilassa, ja sykli saatiin suoritettua loppuun. Testauksen lopuksi voidaan todeta, että koodi toimii teoriassa, mutta täyttä varmuutta siitä, etteikö muutoksia joutuisi tekemään vielä lisää konttinosturin valmistuttua ei luonnollisesti saada.

7 Pohdinta

Tavoitteena oli luoda toimiva konttinosturin pienoismalli. Työn alussa aihe jaettiin mekaaniseen ja ohjelmalliseen osaan. Konttinosturi ei kuitenkaan valmistunut sovitussa ajassa, joten ohjausohjelmatyö muutettiin teoreettiseksi. Työn muutos tuli ilmi myöhään, joka hankaloitti koodin testaamista, mutta lopuksi testaus saatiin suoritettua simuloinnilla. Vaikkei fyysistä rakennelmaa ole, simuloinnin avulla pystyttiin todistamaan koodin toiminta. Automaattisykli toimi odotetusti, joten työn tavoite saavutettiin. Työstä tuli myös laaja teoreettiselta kannalta.

Koodikieleksi valittu FBD osoittautui toimivaksi, kun huomioon otetaan työn alussa asetetut vaatimukset. Valinta perusteltiin opetuskäytön kannalta, mutta työn muuttuessa teoreettiseksi voidaan argumentoida, että jokin toinen koodikieli olisi ollut parempi vaihtoehto. Esimerkiksi SCL-kielellä koodista olisi voitu tehdä helppolukuisempi Case of -rakenteen avulla.

Työstä saadaan jatkotutkimusaihe, joka toteuttaa konttinosturin käyttöönoton sen valmistuessa, ja yhdistää töiden havainnot kokonaisuudeksi. Käyttöönotossa voidaan hyödyntää tästä työstä opittuja konsepteja, sekä vertailla tehtyjä oletuksia koodin toimivuuden suhteen todellisuudessa. Koodiin tulee luultavasti pieniä muutoksia käyttöönoton yhteydessä, esimerkiksi IO-listan muutoksia, tai uusien laitteiden lisäyksiä. Tarttujien ohjaukseen on todennäköisesti lisättävä vielä yksi verkko, jossa tarttujilta tulevat tiedot muutetaan sellaiseen muotoon, että logiikka ymmärtää ne.

LÄHTEET

Berger, H. 2012. Automating with SIMATIC: Controllers, Software, Programming, Data Communication, Operator Control and Process Monitoring. 5th edition. Publicis MCD Werbeagentur GmbH.

Carlo Gavazzi Automation, 1999. Proximity Sensors Capacitive Thermoplastic Polyester Housing Type CA, M18, DC. Luettu 13.11.2020. https://www.starelec.fi/UserFiles/File/PDF-liitteet/CAP-18DC.pdf

Festo, 2006. Standard grippers HGP/HGD/HGR/HGW. Micro grippers HGPM/HGWM. Luettu 1.11.2020. https://www.festo.com/net/SupportPortal/Files/26905/info_116_en.pdf

Fonselius, J., Rinkinen, J. & Vilenius, M. 1998. Servotekniikka. 1. painos. Helsinki: Opetushallitus.

Mean Well, 2015. 240W Single Output Industrial DIN RAIL. NDR-240 series. Luettu 1.11.2020. https://www.mouser.com/datasheet/2/260/NDR-240-spec-1109663.pdf

Metter, M., Pigan, R. 2015. Automating with PROFINET: Industrial Communication Based on Industrial Ethernet. Wiley-Blackwell.

Omron, n.d. D2SW Sealed Subminiature Basic Switch. Luettu 13.11.2020. https://www.starelec.fi/UserFiles/File/PDF-liitteet2/D2SW-OMRON.pdf

Ramil, L., Mohamed, Z., Abdullahi, A., Jaafar, H., Lazim, I. 2017. Control strategies for crane systems: A comprehensive review. Mechanical systems and signal processing, Vol.95, p.1-23.

Siemens, 2019. Servo drive system SINAMICS S210. Luettu 19.9.2020. https://support.industry.siemens.com/cs/document/109763297/sinamics-s210-simotics-s-1fk2?dti=0&lc=en-MX

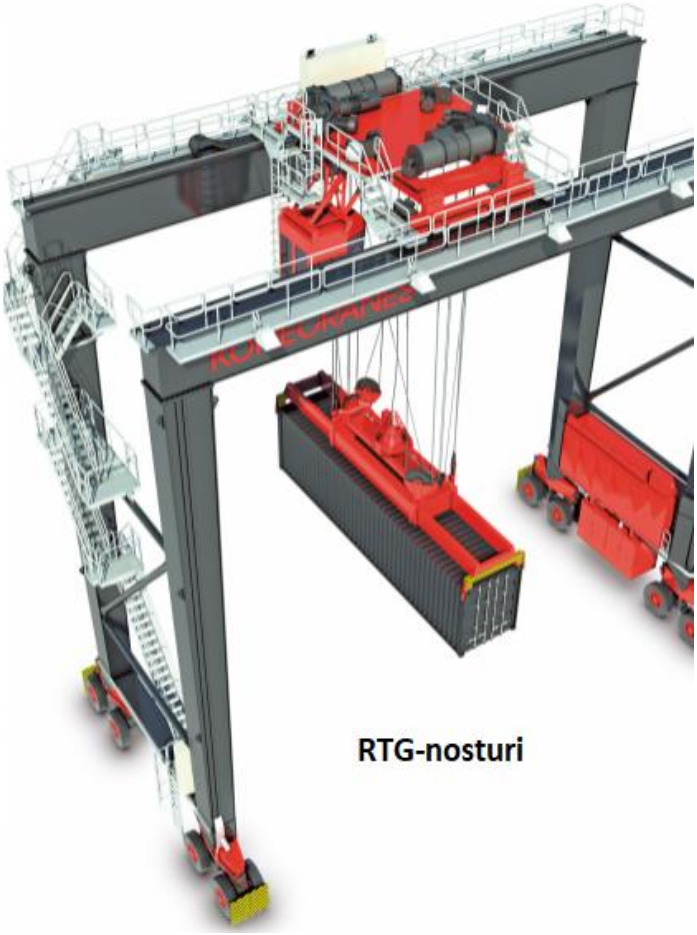
Siemens, 2020. Data sheet for SIMOTICS S-1FK2. Luettu 8.10.2020. https://mall.industry.siemens.com/mall/en/se/Catalog/Product/1FK2103-4AG00-1MA0

LIITTEET

Liite 1. Nosturiesimerkkejä (Konecranes)

Köysinostinnosturi

Kääntöpuominosturi



RTG-nosturi

Liite 2. Konttipaikkojen sijoitus