Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

OLLI LAIHONEN

Laatikkomanipulaattorin ohjelmointi ja käyttöönotto

SÄHKÖ- JA AUTOMAATIOTEKNIIKAN
KOULUTUSOHJELMA
2020

Tekijä(t)
Laihonen, Olli

Julkaisun laji
Opinnäytetyö, AMK

Päivämäärä
Syyskuu 2020

Sivumäärä
66

Julkaisun kieli
Suomi

Julkaisun nimi
Laatikkomanipulaattorin ohjelmointi ja käyttöönotto

Tutkinto-ohjelma
Sähkö- ja automaatiotekniikan koulutusohjelma

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa opinnäytetyön toimeksiantajalta, AMH-Systems Oy:ltä tilatulle manipulaattorille automaatio-ohjelma ja käyttöliittymä, kirjoittaa laitteelle toimintaseloste ja käyttöohjekirja sekä tehdä sille käyttöönotto Eckes-Granini Finland Oy Ab:n Turun tehtaalle.

Automaattiohjelman tarkoituksena oli ohjata manipulaattoria niin, että se ottaa kuormalavalta laatikoita kerroksittain pois ja siirtää ne vieressä olevalle rullapöydälle työntekijöiden käsiteltäväksi. Jokaisen viedyn kerroksen jälkeen työntekijän on kuitattava laite uudelleen käyntiin.

Ohjelmointi pyrittiin toteuttamaan mahdollisimman yksinkertaisesti ja selkeästi noudattaen IEC 61131 standardia sekä AMH-Systems Oy:n aiempia ohjelmointikäytäntöjä. Ohjelma rakennettiin pohtimalla omatoimisesti ratkaisuja haluttujen toimintojen saavuttamiseksi sekä tutkimalla ja hyödyntämällä verkosta löytyvää materiaalia ja AMH-Systemsin aiempaa ohjelmointiprojektia.

Ohjelmoitavana logiikkana toimi Siemensin ET200SP sarjan 1510SP F-1 PN ja sen ohjelmointiin käytettiin TIA Portal -ohjelmointiympäristöä. Tästä syystä opinnäytetyön tutkimusosiossa tutustuttiin ohjelmoitaviin logiikoihin, niissä käytettäviin ohjelmointikieliin sekä vallitseviin ohjelmointikäytäntöihin.

Lopputuloksena asiakkaalle saatiin toimitettua ja asennettua toimivaksi testattu manipulaattori asiaankuuluvine dokumentteineen.

Asiasanat
TIA Portal, HMI, PLC, Movimot

Author(s)
Laihonen, Olli

Type of Publication
Bachelor's thesis

Date
September 2020

Number of pages
66

Language of publication:
Finnish

Title of publication
Programming and Commissioning a Box Manipulator

Degree program
Electrical and Automation Engineering

The purpose of this thesis was to build a PLC program for a customer ordered manipulator, make operating instructions for it and commission the machine at the Eckes-Granini Finland Oy Ab's Turku's premises. The mandator of this thesis was AMH-Systems Finland Oy.

The PLC program's purpose was to automate the manipulator's movements so that it could move a layer of cardboard boxes from a pallet to the table adjacent to it for further processing. After each transferred layer the program would be stopped until an acknowledgement was made to pick another layer.

IEC 61131 standard and AMH-Systems' previous programming methods were followed to make the code as simple and explicit as possible. The program was built by considering solutions to the problems independently and by researching and utilizing materials from the internet and AMH's previous projects.

The programmable logic used in the project was Siemens' ET200SP series 1510SP F-1 PN and it was programmed using the TIA Portal programming tool. Different web- and literature sources were used for this thesis's theoretical part, which was focused on PLC's overall and the current programming customs and languages used in automation engineering.

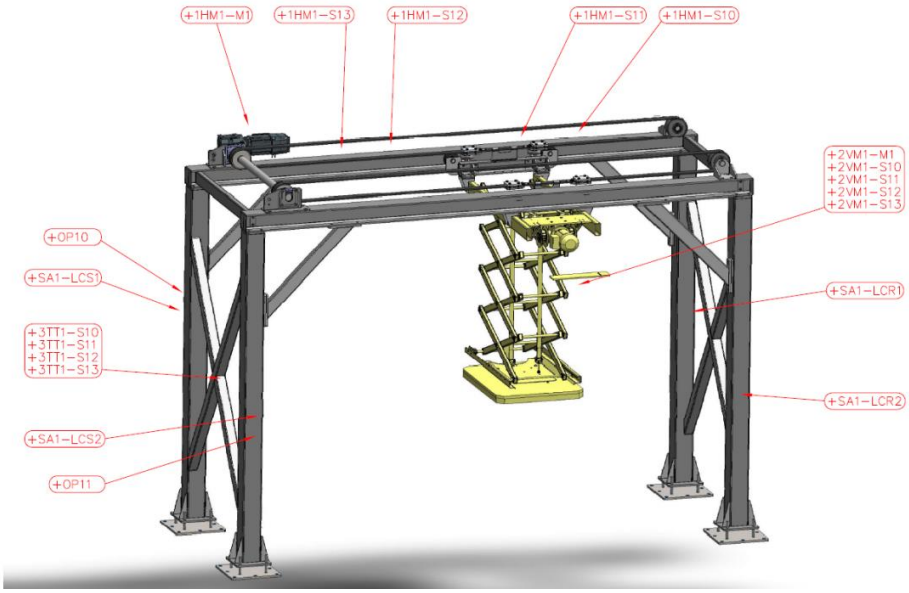
As a result, the manipulator was delivered, installed and fully tested to the client with its appropriate documentation.

Key words
TIA Portal, HMI, PLC, Movimot

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön tavoitteena on toimeksiantajayritykseltä tilatun laatikkomanipulaattorin logiikkaohjelman suunnitteleminen ja rakentaminen sekä laitteen toiminnankuvauksen, käyttöohjeiden ja käyttöönoton tekeminen. Työn toimeksiantaja on AMH-Systems Oy.



Kuva 1. Manipulaattorin 3D-mallinnos

Manipulaattorille rakennetaan automaattiohjelma, jonka avulla laite siirtää käynnistyskuittauksen jälkeen kuormalavalta kerroksen laatikoita viereiselle rullapöydälle työntekijöiden käsiteltäväksi. Käynnistyksen ehtoina on turvapiirin eheys sekä se, että rullapöytä on tyhjä. Manipulaattori muodostaa kokonaisuudessaan yhden automaattialueen ja sen turvalaitteina toimivat kaksi valoverhoparia sekä hätäseispainikkeet. Automaattiohjelman lisäksi laitteelle tehdään käyttöliittymä sekä turva- ja käsiajotoiminnallisuudet.

Opinnäytetyö on toiminnallinen ja siinä käsitellään ohjelmointiprojektia kokonaisuutena. Työn teoriaosuudessa perehdytään ohjelmoitaviin logiikoihin sekä niihin liittyviin standardeihin ja käytäntöihin.

Käyttöönoton arvioidaan olevan vuoden 2020 juhannusviikolla, joten ohjelman ja laitteen testien on oltava valmiina ennen juhannusta. Manipulaattori on kokonaisuudessaan jo suunniteltu valmiiksi sekä sen komponentit on tilattu ja mekaaniset osat ovat viimeisteltävänä. Laitteen koeajojen on tarkoitus alkaa toukokuun lopussa.

1.1 AMH-Systems Oy

AMH-Systems Oy on vuonna 2008 perustettu pyhärantalainen yritys. Vuonna 2019 AMH-Systems työllisti 24 henkilöä ja sen liikevaihto oli 7 miljoonaa euroa. Yrityksen liiketoimintaan kuuluu materiaalinkäsittelyyn liittyvien järjestelmäratkaisuiden suunnittelu ja toteuttaminen. AMH toteuttaa asiakkaan tarpeiden mukaisesti kokonaisia järjestelmäratkaisuja sekä yksittäisiä koneita ja laitteita. Pyhärannan lisäksi yrityksellä on toimipiste myös Loimaalla. (AMH-Systems Oy:n www-sivut 2020; Kauppalehden www-sivut 2020)

2 OHJELMOITAVAT LOGIIKAT

2.1 Historia

Ohjelmoitavat logiikat eli PLC:t (Programmable Logic Controller) ovat alkujaan lähtöisin releohjatuista järjestelmistä. Ennen ohjelmoitavia logiikoita automaatiojärjestelmät toteutettiin sähkömiesten toimesta releistä, kytkimistä, ajastimista ja laskureista koostuvista virtapiireistä, jotka kasattiin joko omatoimisesti tai sähkökuvien perusteella. Relejärjestelmien sähkökuvista nähtiin, kuinka laitteet ja niiden toiminnot olivat yhteydessä toisiinsa ja niitä pidetään Ladder -ohjelmoinnin esiasteena. (Hanssen 2015, 24.)

Mekaanisessa automaatiossa oli paljon haittapuolia ja ongelmia. Suurten tilavaatimusten lisäksi niiden alkuperäinen rakentaminen sekä järjestelmään myöhemmin tehtävät muutokset olivat työläitä. Releohjattu järjestelmä koostuu usein sadoista toisiinsa kytketyistä releistä ja jos sen toimintoja halutaan muuttaa jälkikäteen, koko fyysinen järjestelmä tulee johdottaa uudelleen. Lisäksi laitteilla on rajallinen käyttöikä, joka johtaa vikaantumisiin ja tuotannon häiriöihin. Relejärjestelmien laitteita ei myöskään voitu testata ennen kuin koko järjestelmä oli valmis, joten pienetkin virheet piirustuksissa tai asennuksissa huomattiin vasta lopuksi ja niiden seuraukset saattoivat olla suuria järjestelmän toiminnan kannalta. (Hanssen 2015, 24.)

Ohjelmoitavia logiikoita alettiin kehittämään, kun General Motors alkoi kasvavan kilpailun ja asiakkaiden vaatimusten luoman paineen alla etsimään korvaavaa teknologiaa releohjaukselle. Vaatimuksina järjestelmälle olivat:

- Kilpailukykyinen hinta releohjaukseen verrattuna
- Järjestelmän joustavuus
- Toimintakyky haastavissa ympäristöissä
- Modulaarinen sisään- ja ulostulojen määrä
- Helppo ohjelmoida ja uudelleen ohjelmoida

Useissa yrityksissä alettiin kehittämään ratkaisua ongelmaan ja ensimmäisen kaupallisesti valmistetun PLC:n onnistui kehittämään Bedford Associates Inc. Laitetta kutsuttiin alun perin modulaariseksi digitaaliohjaimeksi (Modular Digital Controller),

josta se sai nimensä MODICON 064. MODICON:issa käytettiin Ladder-ohjelmointikieltä, jota pidetään syynä sen suureen suosioon. (Hanssen 2015, 25-26.)

Aluksi ohjelmoitavien logiikoiden ainoana tarkoituksena oli korvata releet, mutta ajan kuluessa niihin sisällytettiin entistä enemmän toimintoja esimerkiksi ajastimien, laskureiden, vertailuoperaattoreiden sekä laskutoimitusten muodossa. PLC:t tulivat laajasti teollisuuden käyttöön puolijohteiden ja integroitujen virtapiirien kehityksen myötä ja etenkin, kun mikroprosessorit tulivat markkinoille 1970-luvulla. (Hanssen 2015, 25-26.)

Kehityksen myötä myös logiikoiden kommunikaatiotarpeet kasvoivat. Haluttiin logiikoita, jotka voisivat keskustella keskenään ja joita ei tarvitsisi sijoittaa tuotantolinjalle. Vuonna 1973 Modicon kehitti logiikoiden välisen kommunikointiprotokollan nimeltään Modbus. Modbus:n avulla logiikat voitiin asentaa entistä kauemmaksi tuotannosta. (Hanssen 2015, 27-28.)

Ohjelmoitavien logiikoiden ja niihin liittyvien tarvikkeiden valmistajien lisääntyessä myös kommunikointiprotokollia tuli lisää, jotka olivat valmistajasta riippuen joko yleiskäyttöisiä tai laitteisiin sidottuja. Standardoimattomuus ja teknologian kehittyminen johti useisiin erilaisiin keskenään yhteensopimattomiin protokolliin sekä fyysisiin ratkaisuihin. Ennen standardeja ja pitkään sen jälkeen logiikoiden kesken oli suuria eroja valmistajittain. Erot pystyttiin huomaamaan ohjelmointikielien tarjonnassa sekä siitä, kuinka niitä voitiin käyttää PLC:n ohjelmointiympäristössä. Viime aikoina valmistajat ovat kuitenkin alkaneet noudattamaan standardia entistä paremmin. (Hanssen 2015, 27-28.)

2.2 PLC:n toiminta

Ohjelmoitavat logiikat toimivat samalla periaatteella kuin tietokoneet. Jos PLC:llä halutaan toteuttaa tehtäviä, kuten prosessin hallintaa ja monitorointia, se tulee ensin ohjelmoida. Ohjelmoitavan prosessin perusteella päätetään, minkälaisia antureita ja toimilaitteita prosessin ohjauksessa tarvitaan. Fyysiset anturit ja toimilaitteet toimivat PLC:n I/O-moduulissa, josta logiikka saa tarvittavan informaation prosessin

seuraamiseen ja toimilaitteiden ohjaamiseen. Sisääntulojen hyödyntämiseksi kuitenkin tarvitaan logiikkaohjelma, joka käsittelee dataa ja ohjaa laitteita. (Hanssen 2015, 36-37.)

Logiikkaohjelmia rakennetaan usein käytössä olevan PLC:n valmistajan tarjoaman ohjelmointiympäristön avulla. Useissa ohjelmointiympäristöissä on sisäänrakennettuja virheentunnistusmenetelmiä ja mahdollisuus simuloida ohjelman toimintoja. Nämä ominaisuudet helpottavat huomattavasti ohjelman rakentamista ja testaamista. Kun ohjelma saadaan valmiiksi, se voidaan siirtää ohjelmointikaapelilla tietokoneelta logiikalle, jonka jälkeen kaapeli voidaan irrottaa ja PLC on valmis käytettäväksi. (Hanssen 2015, 36-37.)

3 IEC 61131 STANDARDI

IEC 61131 standardi käsittelee PLC-järjestelmien laitteiston sekä niiden ohjelmoinnin vaatimuksia. Standardin loivat SC65BWG7-niminen ryhmä kansainvälisen standardointiorganisaation IEC:n (International Electrotechnical Comission) toimesta. Ryhmä koostui eri PLC-valmistajien, ohjelmistoyritysten sekä niiden käyttäjien edustajista. (Karl-Heinz & Tiegelkamp 2010, 12.)

Standardi on niin laaja, ettei järjestelmien odoteta täyttävän kaikkia vaatimuksia. Sen tarkoituksena onkin olla suuntaa antava eikä asettaa PLC-ohjelmoinnille ehdottomia sääntöjä. Standardi koostuu kahdeksasta osasta, jotka ovat:

1. Määritelmät ja perusominaisuudet
2. Laitevaatimukset ja niihin liittyvät testit
3. Ohjelmointikielet
4. Käyttäjän ohjeistus
5. Laitteiden välinen kommunikaatio
6. Toiminnallinen turvallisuus
7. Ohjelmointikieli sumeaan logiikkaan (Fuzzy Control Language)
8. Ohjeita ohjelmointikielten soveltamiseen ja käyttöönottoon

Standardoinnin saavuttamiseksi PLC -valmistajien tulee dokumentoida, mitkä osat ne täyttävät standardista ja mitä eivät. (Karl-Heinz & Tiegelkamp 2010, 12.)

Standardin tavoitteena on, että ohjelmistot eivät olisi laitevalmistajista riippuvaisia, jolloin esimerkiksi Siemensin logiikkaan tehty ohjelma olisi mahdollista siirtää suoraan jonkun toisen valmistajan, kuten Omronin logiikkaan. (Suvela 2011, 3.)

Standardin laajuuden vuoksi opinnäytetyössä ei käyty sitä kokonaisuudessaan läpi. Sen sijaan standardista on poimittu opinnäytetyön tekijän mielestä työn kannalta olennaisimpia osia, kuten ohjelmointikielet ja ohjeet ohjelman rakentamiseen.

3.1 Ohjelmointikielet

IEC 61131-3 standardissa on määritelty kaksi tekstipohjaista- ja kolme graafista ohjelmointikieltä. Tekstipohjaiset kielet ovat Instruction List (IL) ja Structured Text (ST), kun taas graafisiin kieliin kuuluvat Ladder Diagram (LD), Function Block Diagram (FBD) sekä Sequential Function Chart (SFC). (Karl-Heinz & Tiegelkamp 2010, 97.)

Graafiset kielet käyttävät PLC:n toimintojen muodostamiseen graafisia elementtejä. Kaikkien graafisten kielten yhdistävänä tekijänä on ohjelmointiympäristössä käytettävät viivat, jotka yhdistävät funktioita ja toimintoja keskenään näyttäen, kuinka tieto kulkee ohjelmassa. (Karl-Heinz & Tiegelkamp 2010, 97.)

Graafisiin kieliin verrattuna tekstipohjaiset kielet vievät huomattavasti vähemmän tilaa ja niillä toteutettujen ohjelmien rakennetta ja kulkua pidetään yleisesti helpommin luettavana ja ymmärrettävänä. Lisäksi tekstipohjaisilla ohjelmointikielillä voidaan yhdistää eri ohjelmointikieliä, sillä esimerkiksi toimilohkoilla voidaan suorittaa ST:llä kirjoitettuja funktioita. (PLC Academyn www-sivut 2020)

Logiikkaohjelmaan voidaan sisällyttää useita eri ohjelmointikieliä, jolloin ohjelman eri osissa voidaan käyttää toiminnallisuuksista riippuen parhaiten soveltuvaa ohjelmointikieltä. (Asmala n.da., 3.)

3.1.1 Structured Text (ST)

Structured Text eli lausekieli on teollisuusautomaation tarpeisiin kehitetty korkean tason tekstipohjainen ohjelmointikieli. ST:llä on samoja piirteitä Pascal-ohjelmointikielen kanssa, mutta yhteneväisyyksistä huolimatta ST on oma kielensä. Lausekieli on erittäin käyttökelpoinen monimutkaisten aritmeettisten operaatioiden ratkaisemiseen sekä sen avulla voidaan toteuttaa IEC 61131 -standardin määrittelemiä funktioita ja toimilohkoja. Siemens on kehittänyt standardin pohjalta oman lausekielensä Structured Control Language (SCL). (Asmala n.da., 2.; Siemens 1998. 18.)

Structured Text on IEC 61131-3 standardissa määritelty tekstipohjainen ohjelmointikieli. ST:n syntaksi on samankaltainen kuin muidenkin korkean tason ohjelmointikielien pitäen sisällään silmukoita, muuttujia, ehtoja ja operaattoreita. ST:n syntaksissa kaikki lauseet on eroteltu toisistaan puolipisteillä. Lauseilla tarkoitetaan kaikkea, mitä ST-ohjelmassa tehdään, kuten muuttujien luomista tai niiden arvojen muuttamista. (PLC Academyn www-sivut 2020)

```
PROGRAM stexample
  VAR
    x : BOOL;
  END_VAR
  x := TRUE;
  REPEAT
    x := FALSE;
  UNTIL x := FALSE;
  END_REPEAT;
END_PROGRAM;
```

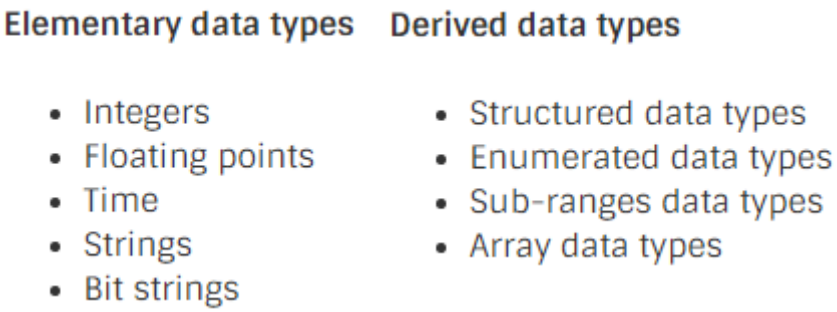
Kuva 2. Structured Text koodia.

ST:llä kirjoitettujen ohjelmien kehys on aina sama, se alkaa sanalla PROGRAM ja loppuu sanaan END_PROGRAM. Kaikki niiden väliin kirjoitettu kuuluu PLC:n ohjelmaan ja näin määritellään ST-ohjelman ohjelmakierto. PROGRAM ja END_PROGRAM -komentoja ei kuitenkaan tarvitse erikseen kirjoittaa useimpien logiikkavalmistajien ohjelmointialustoilla, sillä ne on usein sisällytetty koodiin valmistajan toimesta, jolloin ohjelmaan tarvitsee kirjoittaa ainoastaan suoritettavat toiminnot. (PLC Academyn www-sivut 2020)

PROGRAM ja END_PROGRAM, VAR ja END_VAR sekä TYPE ja END_TYPE ovat ST-ohjelmissa rakenteita, jotka varaavat niiden välissä olevan osan ohjelmaa tietyn tehtävän suorittamiseen, kuten muuttujien ja tietotyyppien määrittelyyn. ST:n ja PLC:n ohjelmakiertoja ei tule kuitenkaan sekoittaa keskenään, sillä ST:n loppuessa logiikka tekee muut mahdolliset toimintansa loppuun ja kutsuu seuraavalla kierrolla ST-ohjelmaa uudelleen. ST toimii siis samalla periaatteella kuin muutkin ohjelmoitavien logiikoiden ohjelmointikielet suorittaen ohjelmansa rivin kerrallaan koodin alusta loppuun. (PLC Academyn www-sivut 2020)

Kuva 3. ST ohjelmakierto

ST:llä voidaan käyttää IEC 61131-3 standardin mukaisesti kaikkia tavallisia ja johdettuja tietotyyppejä riippuen käytössä olevasta logiikasta. Lisäksi käytettävissä on tavalliset aritmeettiset-, vertailu-, loogiset- ja bittioperaattorit sekä valintarakenteet ja toistolauseet. (PLC Academyn www-sivut 2020)



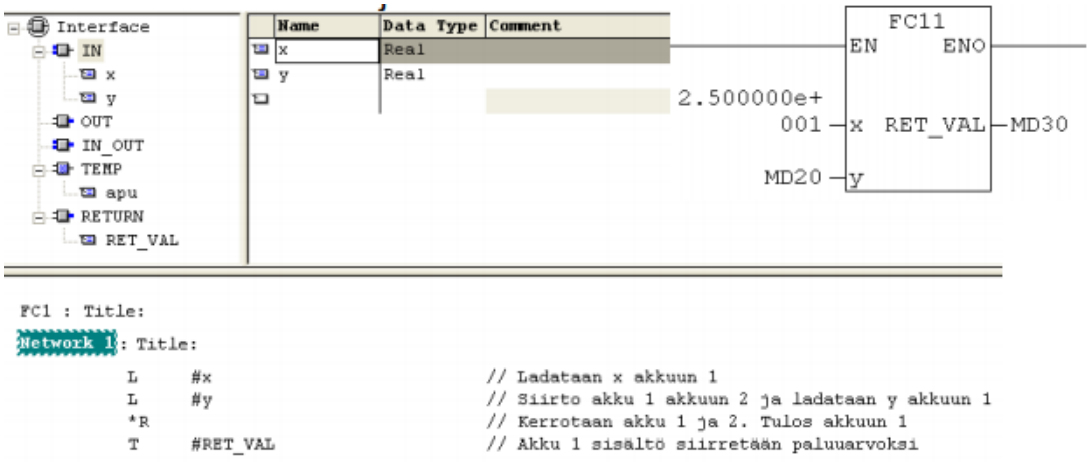
Kuva 4. Tietotyypit

- + (add)
- – (subtract/negate)
- * (multiply)
- ** (exponent)
- / (divide)
- MOD (modulo divide)
- = (equal)
- < (less than)
- <= (less than or equal)
- > (greater than)
- >= (greater than or equal)
- <> (not equal)
- & or AND
- OR
- XOR
- NOT

Kuva 5. Operaattorit

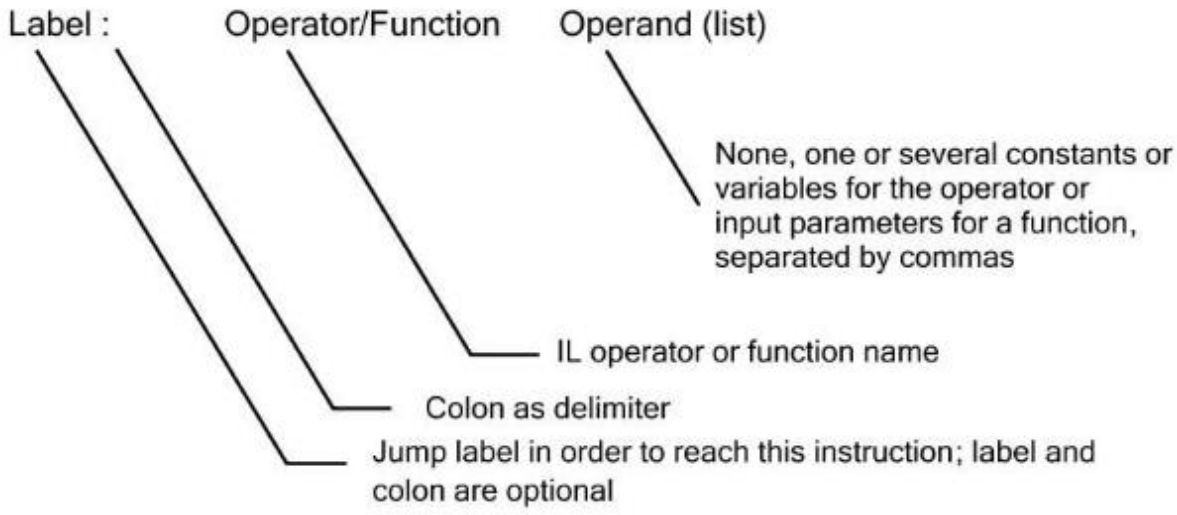
3.1.2 Instruction List (IL)

Käskylista eli Instruction List on tekstipohjainen ohjelmointikieli, joka muistuttaa Assembler-ohjelmointikieltä. Ohjelmointiympäristöstä riippuen käskylistalla kirjoitettua koodia voidaan liittää esimerkiksi rele- ja logiikkakaavioissa omiin virtapiireihinsä. (Asmala n.db., 2.)



Kuva 6. IL Step 7 ohjelmassa

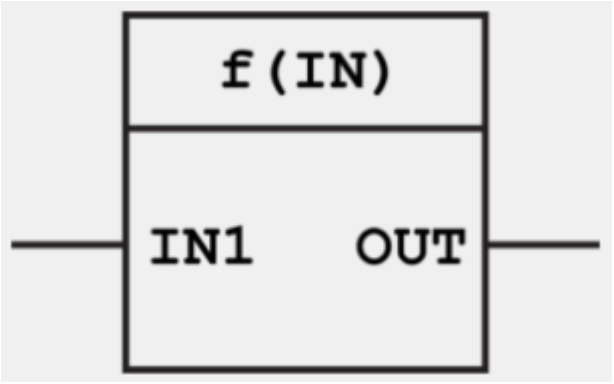
IL on yleisesti ohjelmoitavien logiikoiden käyttämä suorituskieli, johon graafiset tai tekstikieliset ohjelmat käännetään ennen niiden varsinaista suoritusta. Käskylistalla toteutettavat toiminnot on määritelty yhdellä ainoalla rivillä eli lauseella, joita lukemalla PLC toteuttaa toimintoja. IL:n lause koostuu mahdollisesta nimestä, operaattorista tai funktiosta sekä toiminnosta riippuen muuttujista tai vakioista. (Karl-Heinz & Tiegelkamp 2010, 98.)



Kuva 7. Käskylistan lause

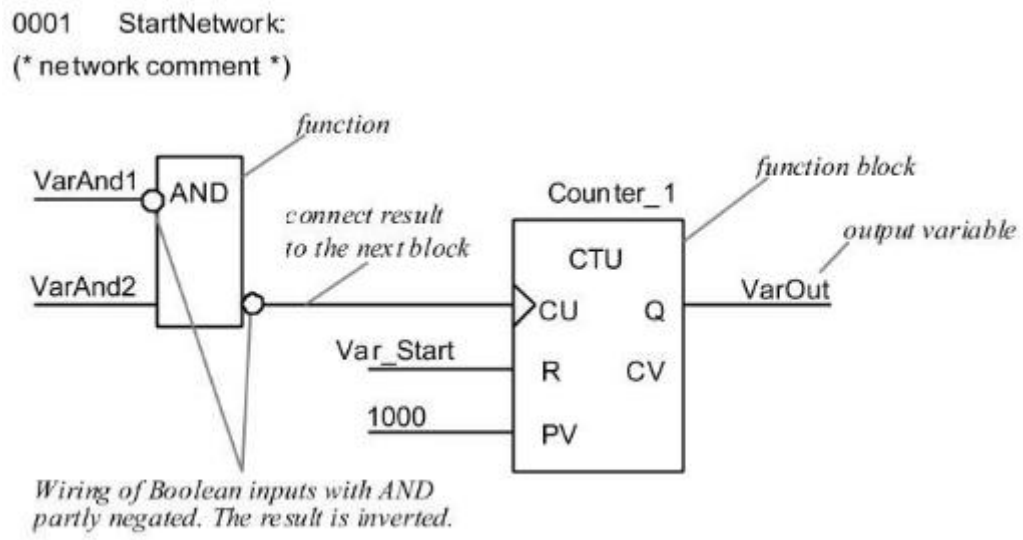
3.1.3 Function Block Diagram (FBD)

Function Block Diagram (FBD) eli toimilohkokaavio on graafinen ohjelmointikieli, jolla saadaan useita koodirivejä vaativat funktiot toteutettua yksittäisten toimilohkojen avulla. FBD on laajasti käytössä oleva kieli ja sen avulla saadaan toteutettua hyvin kattavasti erilaisia logiikkaohjelmissa käytettäviä toimintoja. (PLC Academyn www-sivut 2020)



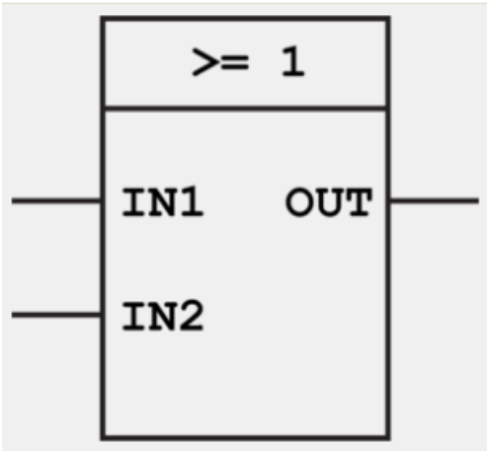
Kuva 8. Yksittäinen toimilohko

Toimilohko esitetään ohjelmointiympäristöissä laatikkona. Laatikon keskellä on usein symboli tai teksti, jonka avulla kerrotaan, mikä toiminnallisuus lohkoon on liitetty. Toiminnallisuuden mukaan toimilohkolla voi olla useita sisään- tai ulostuloja. Toimilohkon ulostulolla voidaan kirjoittaa tietoa suoraan erilliseen muuttujaan tai sitten se voidaan liittää toisen toimilohkon sisääntuloon, jolloin yhdistettyjen lohkojen kokonaisuudesta muodostuu toimilohkokaavio. (PLC Academyn www-sivut 2020)



Kuva 9. Toimilohkokaavio

IEC 61131-3 standardissa on määritelty useita FBD:n valmiita toimilohkoja. Tärkeimpiä standardoituja lohkoja ovat esimerkiksi binääriset vertailuoperaattorit AND- ja OR-lohkot sekä niiden eri variaatiot eli poissulkeva XOR sekä käänteiset NAND ja NOR. Numeerisessa vertailussa taas voidaan esimerkiksi tarkkailla annettuja arvoja muodoissa suurempi-, pienempi-, yhtä suuri kuin tai niiden yhdistelmillä. Numeerisia arvoja voidaan myös siirtää muuttujasta toiseen ja niiden arvoja voidaan muuttaa ja rajoittaa erilaisten laskutoimitusten avulla. Lisäksi standardiin on sisällytetty useita erilaisia laskuri- ja ajastintoimintoja. (PLC Academyn www-sivut 2020)

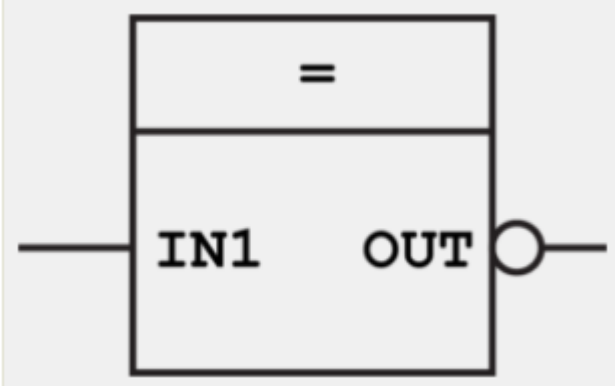


Kuva 10. OR-lohko

FBD-ohjelmoinnissa ulostuloja tulisi ohjata kirjoituslohkon avulla. Kirjoituslohkossa ulostulo asetetaan päälle, jos sen sisääntuloon tulee signaali ja heti signaalin katketessa ulostulo ohjataan pois. Ulostulojen ohjauksessa tärkeitä toimintoja ovat myös SR- ja

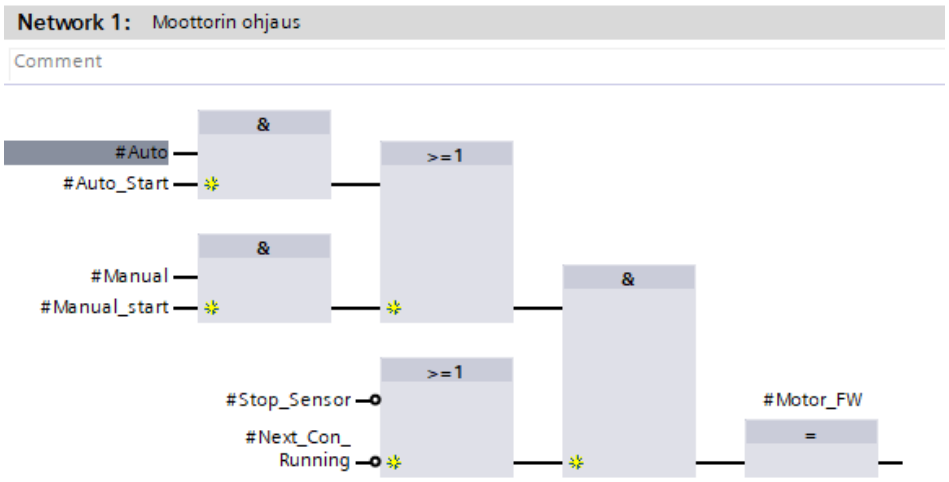
RS-kiikut sekä nousevaa ja laskevaa reunaa tulkitsevat toimilohkot. (PLC Academyn www-sivut 2020)

Set/Reset- ja Reset/Set -kiikkujen avulla lohkon ulostulo asetetaan päälle tiettyjen ehtojen täyttyessä ja se pidetään päällä, kunnes nollauksen ehdot täyttyvät. Usein kiikkujen yhteydessä käytettävien reunoja tarkkailevien toimilohkojen ulostulo taas asetetaan hetkellisesti päälle sisääntulon joko mennessä päälle tai pois, jolloin sisääntulo huomioidaan ainoastaan silloin, kun sen tila muuttuu. Toimilohkojen sisään- ja ulostulojen bitit voidaan myös kääntää negaatiolla, jolloin niiden tilatietoja tulkitaan käänteisesti. (PLC Academyn www-sivut 2020)

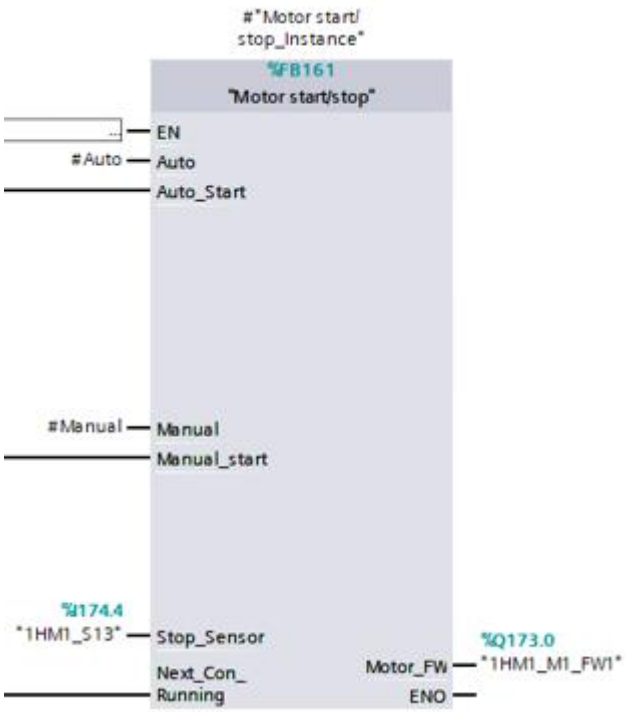


Kuva 11. Kirjoituslohko negaatiolla

Standardoitujen toimilohkojen lisäksi FBD:lla on mahdollista rakentaa omia toiminnallisuuksia toimilohkojen avulla. Usein logiikkaohjelmissa tulee vastaan tilanteita, joissa samaa koodia voitaisiin hyödyntää ohjelman muiden samankaltaisten toimintojen toteuttamisessa, kuten moottoreiden tai venttiilien ohjauksissa. Toimilohkoista voidaan tehdä uudelleenkäytettäviä, jolloin niitä on mahdollista kutsua ohjelmassa useita kertoja tarpeen mukaan. Yhteen ohjelmalohkoon voidaan esimerkiksi rakentaa moottorin ohjauksen toiminnallisuudet, joita taas kutsutaan ohjelmassa kunkin ohjattavan moottorin kohdalla erikseen. (PLC Academyn www-sivut 2020)



Kuva 12. Moottorin ohjauslohko

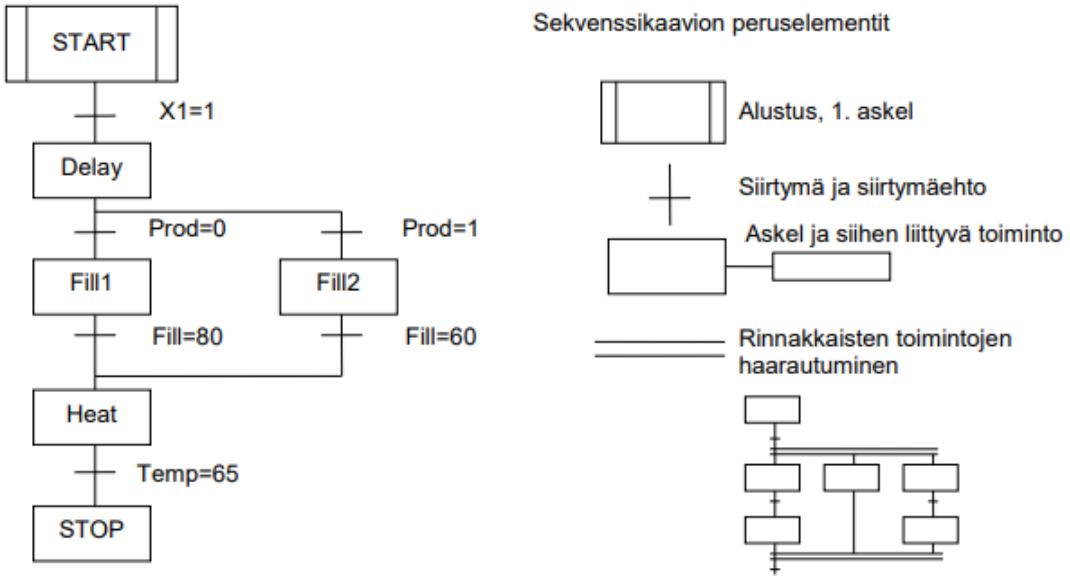


Kuva 13. Moottorin ohjauslohko kutsuttuna ohjelmassa

3.1.4 Sequential Function Chart (SFC)

Sequential Function Chart eli sekvenssikaavio kuvaa graafisesti ohjelman käyttäytymistä ohjelmasekvensseissä. Kaaviosta voi nähdä helposti järjestelmän toimintaperiaatteen varsinkin, jos askeleet on nimetty toimintoja kuvaavilla nimillä. Sekvenssikaavio sopii hyvin prosessien tai laitteiden ohjaamiseen, joiden toimintojen vaiheet seuraavat tarkasti toisiaan, mutta se ei kuitenkaan sovellu hyvin prosesseihin, jotka eivät

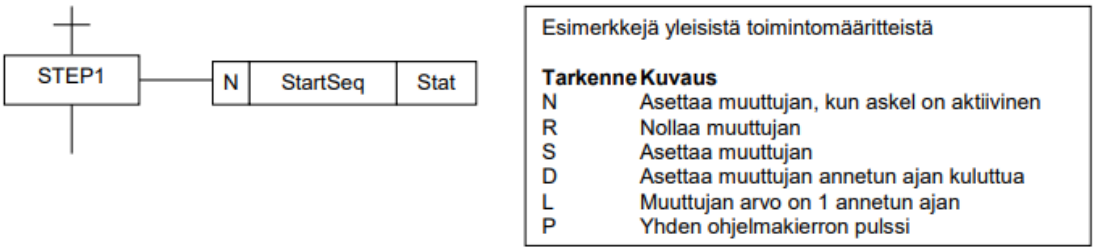
ole luonteeltaan askelittaisia. Sekvenssikaaviossa voidaan kutsua muita sekvenssejä ja niitä voidaan suorittaa rinnakkain toisistaan rippumatta. SFC:lla voidaan myös valita suoritettava työvaihe vaihtoehtoisista askeleista. Siemensin laitteistoissa sekvenssikaavioita voidaan rakentaa Siemensin omalla GRAPH-ohjelmointikielellä.
(Asmala n.dc., 2.; Satakunnan Ammattikorkeakoulu n.d., 2.; Siemensin www-sivut 2020)



Kuva 14. SFC

Sekvenssikaavio koostuu askeleista, toiminnoista ja siirtymistä. Siirtymillä määritellään, milloin yksi askel päätetään ja seuraava aloitetaan. Siirtymäehtojen täytyttyä ohjelman kontrollointi siirretään seuraavalle askeleelle ja siihen yhdistetyt toiminnot suoritetaan. Kun askeleen vaikutus on loppunut, toiminnot päätetään ja siirrytään seuraavaan askeleeseen. Askeleiden toiminnot voidaan kuitenkin jättää muistiin, jolloin ne voivat olla voimassa useammankin askeleen ajan. (Asmala n.dc., 3.)

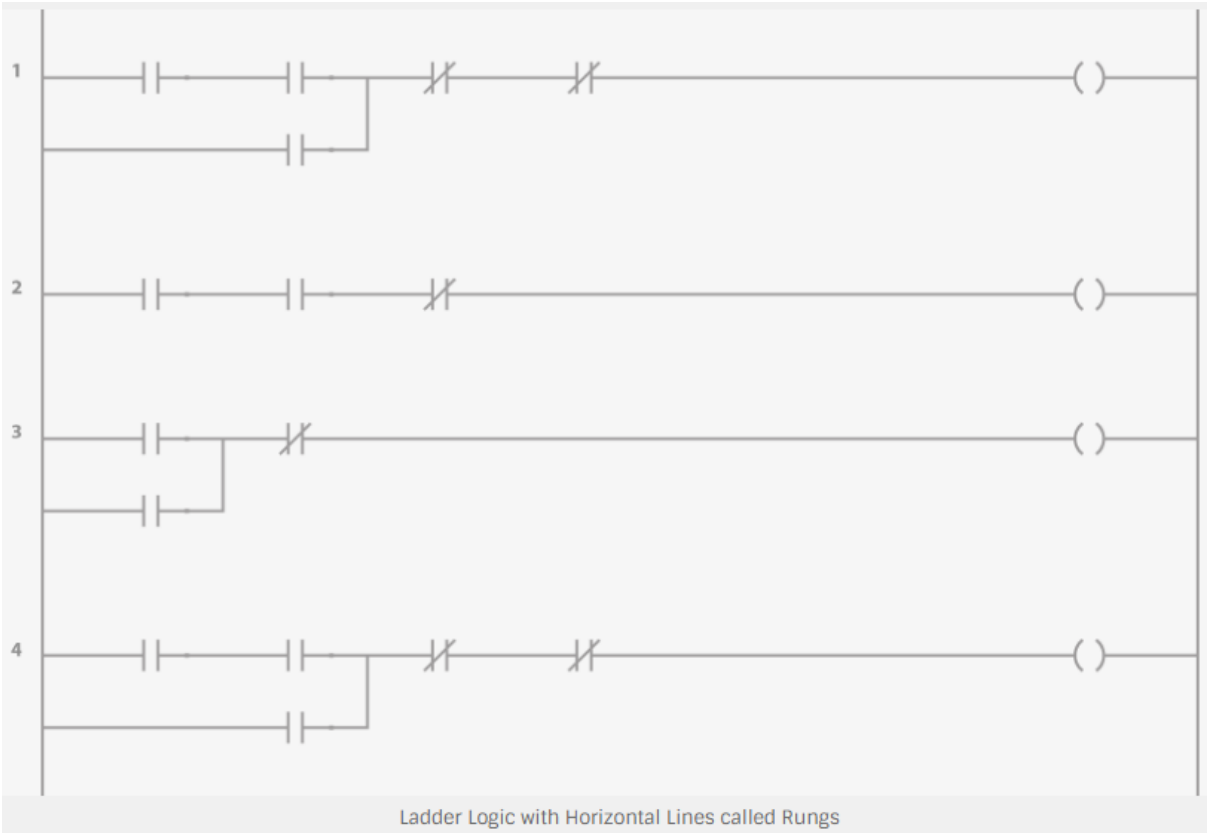
SFC:n toiminnot ovat vähintään kaksiosaisia. Toiminnolle määritellään ensin tarkenne, jolla voidaan vaikuttaa toiminnon kohteeseen halutulla tavalla. Kohteena voi olla esimerkiksi muuttuja, jonka sisältöä halutaan muuttaa. Toiminnon viimeiseen osaan voidaan haluttaessa liittää muuttuja, johon kirjoitetaan tieto siitä, onko toiminto suoritettu. Askeleisiin voidaan liittää yksinkertaisia toimintoja, kuten SET ja RESET, mutta monimutkaisemmat toiminnot vaativat ohjelmointia jollain muulla ohjelmointikielellä, kuten Ladderilla. (Satakunnan Ammattikorkeakoulu n.d., 13.)



Kuva 15. Sekvenssikaavion toiminnot

3.1.5 Ladder Diagram (LD)

Ladder Diagram (LD) eli tikapuukaaviota pidetään yhtenä parhaista ja helpoiten opittavista graafisista ohjelmointikielistä. Tikapuukaavioissa ohjelman toiminnot rakennetaan ohjelmointiympäristössä vaakaviivoille käyttämällä erilaisia symboleja. Ohjelmointikieli on alun perin luotu henkilöille, joilla on taustaa sähkötöistä, joten kielessä käytettävät symbolit on luotu muistuttamaan sähkötekniikassa käytettäviä symboleita. (PLC Academyn www-sivut 2020)



Kuva 16. Ladder-ohjelma

Tikapuukaavio on pääsääntöisesti tarkoitettu binääristen toimintojen suorittamiseen. Symboleihin voidaan liittää fyysisten- tai logiikan sisäisten muuttujien osoitteita, joita

ohjelmassa luetaan tai kirjoitetaan. Kielen käytetyimmät symbolit sisääntulojen lukemiseen ovat normaalisti auki (NO) ja normaalisti kiinni (NC). Sisääntulon ollessa päällä NO ohjaa virtaa eteenpäin ja NC-symbolilla tuloa tulkitaan päinvastoin. Ulostuloja tikapuukaaviossa ohjataan käämisymbolilla. (PLC Academyn www-sivut 2020)

Kuva 17. NO, NC ja käämi

3.2 Ohjelmarakenne

Logiikkaohjelma rakennettiin AMH-Systems Oy:n aiemman projektin pohjalle, joten sen rakennetta ei tarvinnut juurikaan muuttaa. IEC 61131 standardin osilta perehdyttiinkin siis lähinnä nimeämiseen ja kommentointiin, sillä muilla osa-alueilla noudatettiin AMH:n käytäntöjä. Seuraavassa kappaleessa on kerrottu tarkemmin standardissa olleista ohjeista, joita hyödynnettiin ohjelmaa rakentaessa.

3.2.1 Nimeäminen

Ohjelmaa tehdessä tulee määritellä ja tiedostaa nimet, joita ei saa käyttää minkään ohjelman osa-alueen nimeämisessä. Standardissa ohjeistetaan välttämään:

- IEC tietotyyppejä ja standardikirjaston objektien nimiä
- ST-kielen avainsanoja
- Omalle projektille varattuja sanoja
- Erikoisia, ei käytettyjä tai tunnettuja lyhenteitä
- Merkityksettömiä nimiä, kuten info, data, temp, str, buf

Varattuja- ja avainsanoja tulee välttää, koska ne aiheuttavat usein ohjelmaan virheitä sekä hankaloittavat ohjelman luettavuutta ja huollettavuutta. Myös sanojen turhaa lyhentämistä ja varsinkin liian lyhyitä lyhenteitä tulisi välttää sillä ne voivat olla harhaanjohtavia. (PLCopen 2016, 20.)

Ohjelman samankaltaisille objekteille tulisi määritellä kirjoitusasu, jota noudatetaan kauttaaltaan koko projektissa. Vaihtoehtoisia kirjoitusasuja on esimerkiksi:

- alllowercase
- underscore_separated
- lowerCamelCase

- UpperCamelCase
- ALLUPPERCASE
- UPPER_SNAKE_CASE
- OTHER_style

Projektin ohjelmalohkot voidaan esimerkiksi kirjoittaa muotoon UPPER_SNAKE_CASE ja muuttujat muotoon UpperCamelCase. Kirjoitustyyli ei vaikuta ohjelman toimintaan, mutta siihen on hyvä kiinnittää huomiota ohjelman luettavuuden parantamiseksi. (PLCopen 2016, 22.)

Muuttujille tulisi antaa ainoastaan yksilöityjä ja merkityksellisiä nimiä, jotka kertovat muuttujan käyttökohteesta tai tarkoituksesta. Fyysisten osoitteiden, kuten ”Q130.0”, ”I130.1” ja ”M130.2” käyttöä tulisi välttää, sillä ne hankaloittavat ohjelman luettavuutta ja niiden tulkitsemiseen tarvittaisiin todennäköisesti sähkökuvia tai osoitetaulua. Lisäksi ohjelman virhealttius kasvaa, jos logiikkaohjelman tai fyysisen laitteiston I/O-avaruutta muokataan jälkeenpäin. (PLCopen 2016, 16.)

Koodin luettavuuden vuoksi kaikkien ohjelmassa olevien nimien tulee olla yksilöllisiä. Samojen nimien lisäksi myös samankaltaisia nimiä tulee välttää ja tarpeen mukaan nimeämisessä voidaan käyttää etuliitteitä. (PLCopen 2016, 25.)

Nimien tukena voidaan käyttää etuliitteitä, joiden avulla esimerkiksi samankaltaisia muuttujia saadaan ryhmiteltyä. Etuliitteiden muodon voi määritellä halutulla tavalla esimerkiksi tietotyypin (bool, int), laajuuden (globaali, lokaali, parametri), kontrollin (input, output) tai toiminta-alueen mukaan. Etuliitteestä ei saa kuitenkaan tehdä liian pitkää tai monimutkaista vaan se tulisi pitää sopivan lyhyenä. Yleisenä käytäntönä on antaa etuliite tietotyypin mukaan. Etuliitteiden käytössä tärkeintä on noudattaa valittua nimeämissääntöä projektin alusta loppuun. Jos kaikkia muuttujia ei nimetä yhdenmukaisesti, niiden yhteistä tekijää ei voida tunnistaa enää helposti ja ryhmittely epäonnistuu. (PLCopen 2016, 17.)

Ohjelmassa tulisi käyttää kaikkia siihen luotuja muuttujia. Jos ohjelmaan jätetään käyttämättömiä muuttujia, sen luettavuus ja huollettavuus kärsii. Tämän lisäksi muuttujat vievät hieman PLC:n muistia. Poikkeuksena on kuitenkin varamuuttujat, joilla varataan muistia tulevia muutoksia sekä online-muokkauksia varten. Jossain tapauksissa

muuttujia ei voida luoda pysäyttämättä logiikan toimintoja, joten ne tulee luoda ohjelmaan valmiiksi. (PLCopen 2016, 94.)

Datan eheyden edellytyksenä on, että kuhunkin muuttujaan kirjoitetaan aina ainoastaan yhdessä ohjelman osassa, vaikka sitä käytettäisiinkin ohjelman useissa eri paikoissa. Jos muuttujaa kirjoitetaan useassa ohjelman osassa, on mahdollista, että erilliset tehtävät yrittävät kirjoittaa samaan aikaan muuttujalle arvoa, joka taas johtaa väärään dataan tai sen häviämiseen. Muuttujaan tulisi siis kirjoittaa ainoastaan kerran koko ohjelmassa, mutta poikkeuksia voidaan kuitenkin tehdä, jos toiminnot ovat hyvin samankaltaisia tai ne liittyvät toisiinsa. (PLCopen 2016, 63.)

Lähtökohtaisesti kaikki ohjelmassa oleva koodi on kommentoitava. Kommentoinnin tarkoituksena on kertoa, mitä ohjelmoija on pyrkinyt saavuttamaan ohjelmassa ja mitä kyseisellä koodilla on tarkoitus tehdä. Hyvän ohjelmoinnin käytännöt edellyttävät kuitenkin, että hyvin kirjoitettu ja nimetty koodi kertoo riittävästi toiminnoistaan, joten jos koodin rakenne on riittävän yksinkertaista ja selkeää, sitä ei ole välttämätöntä kommentoida. Jos koodia kuitenkin kommentoidaan, sillä ei saa olla kommentista poikkeavia toimintoja. Ohjelmakoodin lisäksi ohjelman muutkin elementit, kuten ohjelmalohkot ja muuttujat tulisi kommentoida. Kommentoinnilla voidaan parantaa huomattavasti ohjelman ymmärrettävyyttä. (PLCopen 2016, 36-37.)

Ohjelmaan ei saa jättää käyttämätöntä koodia. Kuollut koodi vaikuttaa ohjelman huollettavuuteen ja luettavuuteen sekä voi joissain tapauksissa aiheuttaa toimintaan häiriöitä. Joissain tapauksissa kuollutta koodia voidaan kuitenkin jättää ohjelmaan, jos kommenteissa kerrotaan selkeästi sen olevan käyttämätöntä sekä syy, miksi se on päätetty jättää ohjelmaan. (PLCopen 2016, 45.)

4 KÄYTETYT OHJELMAT JA LAITTEISTO

Opinnäytetyössä oleellisin ohjelma oli Siemensin TIA Portal -ohjelmointiympäristö, jossa toteutettiin manipulaattorin logiikkaohjelman toiminnot. TIA Portalin lisäksi

työtä tehdessä tutustuttiin TIA Selection Tooliin, jonka avulla manipulaattorin laitekonfiguraatio ladattiin TIA-projektiin.

4.1 TIA Portal

TIA Portal (Totally Integrated Automation Portal) on Siemensin kehittämä automaatiosovellusten suunnitteluohjelmisto. TIA Portalissa on yhdistetty Siemensin STEP 7, WinCC, SINAMICS Startdrive, SIMOCODE ES ja SIMOTION SCOUT TIA -ohjelmistojen toiminnallisuudet, joka tarkoittaa sitä, että ohjelmistolla voidaan tehdä PLC-ohjelmointia, käyttöliittymiä, liikkeenohjausta, säätää laitteiden virranjakoa sekä parametroida ohjaimia ja ohjelmistoja. (Siemensin www-sivut 2020)

4.2 TIA Selection Tool

TIA Selection Tool on Siemensin luoma automaatiosovellusten laitekonfiguraatioiden rakentamiseen tarkoitettu ohjelmisto. Selection Tooliin on sisällytetty kaikki Siemensiltä tilattavissa olevat komponentit, joita voidaan hyödyntää oman laitteiston rakentamisessa, jonka lisäksi ohjelma osaa tarjota korvaavia tuotteita poistuneiden mallien tilalle. (Siemensin www-sivut 2020)

Selection Tool muodostaa tehdystä laitekonfiguraatiosta laitelistan ja kirjautumalla Siemensin Industry Mall -palveluun, sillä pystytään tarkastelemaan myös erillisten laitteiden hintoja sekä laitteiston kokonaishintaa. Ohjelmalla voidaan myös vertailla vaihtoehtoisten konfiguraatioiden hintoja, sillä yhteen projektiin voidaan luoda useita erillisiä laitekokoonpanoja. Ohjelmalla tehtyjä projekteja voidaan jakaa käyttäjien kesken linkkeinä tai tiedostoina. (Siemensin www-sivut 2020)

Projektitiedostojen avulla laitekonfiguraatio voidaan ladata kokonaan tai osittain Siemensin ohjelmointiympäristöihin, kuten TIA Portaliin. Projektin viemisen yhteydessä laitteita voidaan vielä muokata ja päättää, pidetäänkö Selection Toolissa määritellyt laiteyhteydet. Siemensin ohjelmointiympäristöistä on myös mahdollista tuoda laitekonfiguraatioita TIA Selection Tooliin. (Siemensin www-sivut 2020)

4.3 Komponentit

Manipulaattori oli suunniteltu sähköisesti ja mekaanisesti valmiiksi opinnäytetyötä aloittaessa. Opinnäytetyössä perehdyttiin erityisesti laitteen sähkökuviin. Ohjelmoinnin kannalta manipulaattorin olennaisimmat laitteet olivat sijoitettu manipulaattorin keskukseen, jotka olivat:

- ET 200SP CPU 1510SP F-1 PN x1 (PLC)
- F-DI8x24VDCHF x2 (Turvatulot)
- F-DQ4x24VDC/2APMHF x1 (Turvalähdöt)
- DI8x24VDCHF x7 (Digitaaliset tulot)
- DQ8x24VDC/0.5AHF x5 (Digitaaliset lähdöt)
- KTP900 Basic PN x1 (Ohjauspaneeli)

Edellä mainittujen lisäksi manipulaattorin laitteistoon kuului erilaisia toimi- ja turvalaitteita, releitä, painikkeita ja kytkimiä sekä paineilmakomponentteja.

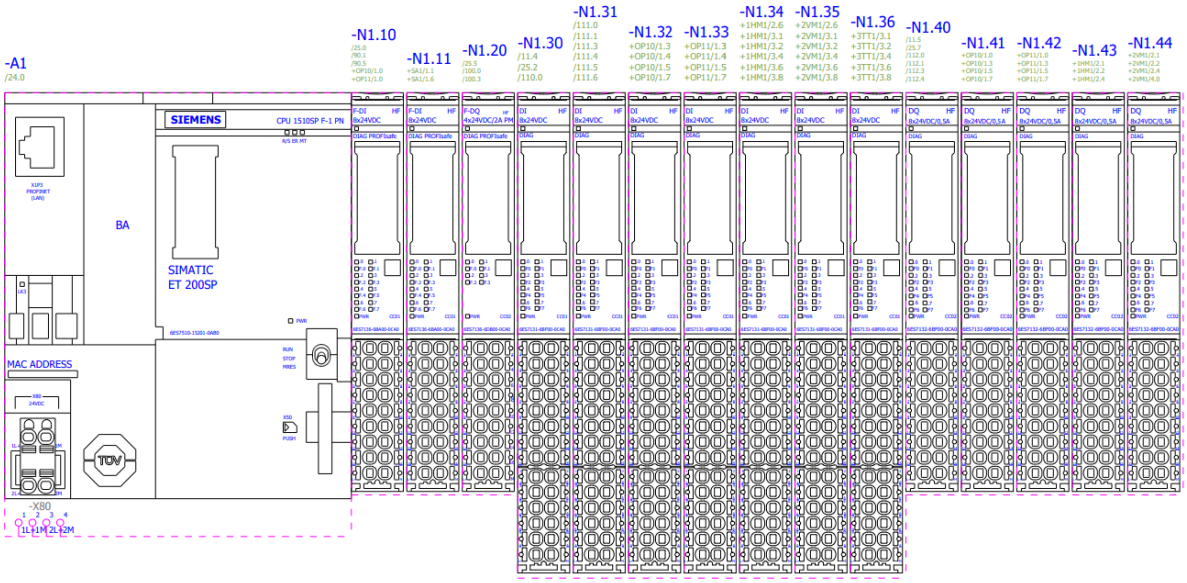
4.3.1 PLC

Manipulaattorin logiikkana toimi Siemensin hajautetun I/O:n ET 200SP -sarjaan kuuluva 1510SP F-1 PN turvalogiikka. Hajautetulla I/O:lla tarkoitetaan järjestelmää, jossa tulo- ja lähtöpiirejä on sijoitettu prosessiaseman sijasta lähemmäs toimilaitteita. Siemensin hajautetun I/O:n tuotteet kulkevat nimellä SIMATIC ET 200. (Sitek-Palvelu Oy:n www-sivut 2020)

ET 200 -sarjan logiikoilla on käytännössä samat toiminnallisuudet sekä keskusyksikkö kuin S7-1500 tuoteperheen 1511- ja 1513 logiikoilla. Logiikoissa kuitenkin käytetään sisään- ja ulostulojen moduuleita, jotka kiinnitetään suoraan PLC:n kylkeen. I/O-moduulit voivat olla joko digitaalisia, analogisia tai turvamalleja (Fail-Safe). ET 200 sarjasta on saatavilla neljää eri logiikkamallia; tavalliset PLC:t 1510SP-1 PN ja 1512SP sekä turvalogiikat 1510SP F-1 PN ja 1512SP F-1 PN. Tavallisen ja turvalogiikan pystyy erottamaan nimessä olevan F-kirjaimen perusteella. (Siemensin www-sivut 2020)

Turvalogiikka eroaa tavallisesta PLC:stä niin, että siinä on tavallisten ohjelmointitoimintojen lisäksi mahdollisuus ohjelmoida erillinen turvaohjelma. Logiikat on sertifioitu EN 61508 ”Toiminnallinen turvallisuus” -standardin mukaisesti ja niitä voidaan

käyttää turvatoimintojen ohjaamisessa SIL 3 -tasoon asti. Logiikoihin voidaan yhdistää ET 200SP:n tavallisia- ja turva I/O-moduuleja. (Siemensin www-sivut 2020)



Kuva 18. PLC sähkökuvissa

IEC 61508 standardissa on määritelty turvajärjestelmille toteutettavien turvatoimintojen vaatimustasot. Turvallisuuden eheyden tasot (SIL, Safety Integrity Level) on määritelty asteikolle yhdestä neljään, joista ensimmäiselle on asetettu pienimmät vaatimukset ja neljännelle suurimmat. (IEC 2004, 8.)

4.3.2 Moottorit

Manipulaattorin moottoreina käytettiin SEW-EURODRIVE:n MOVIMOT MM03D-503-00 moottoreita. MOVIMOT on hajautettuun käyttölaitetekniikkaan tarkoitettu vaihdemoottorin ja digitaalisen taajuusmuuttajan yhdistelmä, joten laitteen moottoreita ohjaavat taajuusmuuttajat oli integroitu moottoreihin. (SEW-EURODRIVE:n www-sivut 2020)

Moottoreita ohjattiin binäärisesti logiikan ulostulojen kautta, mutta niitä olisi ollut mahdollista ohjata myös jonkin kenttäväylän, kuten PROFINET:in avulla (SEW-EURODRIVE 2014, 15). Väyläohjauksella taajuusmuuttajaa olisi voitu ohjata PLC:n kautta prosessidatasanojen avulla. Väyläohjauksen ohjaus koostuu ohjaussanasta ja pyörimisnopeuden prosentista sekä haluttaessa rampista, joiden avulla on mahdollista

esimerkiksi vaihtaa moottorin pyörimissuuntaa, käynnistää laite, kuitata häiriöitä, vapauttaa tai lukita laitteen jarrut sekä säätää käyntinopeutta ja ramppiaikaa. (SEW-EURODRIVE 2014, 119-121.)

6.6.2 DIP-kytkin S2/2

Jarrun vapautus ilman aktivointia

Kun DIP-kytkimen S2/2 asetuksena = "ON", jarrun vapautus on mahdollista myös silloin, kun käyttölaitetta ei ole aktivoitu.

Toiminnot binääriohjauksessa

Jarru voidaan vapauttaa binääriohjauksessa asettamalla liittimen f1/f2 X6:7,8 signaali seuraavien edellytysten täyttyessä:

Liittimen tila			Aktivointi-tila	Vikatila	Jarrutoiminto
R ↷ X6:11,12	L ↶ X6:9,10	f1/f2 X6:7,8			
"1" "0"	"0" "1"	"0"	Laite aktivoitu	Ei laite-virhettä	MOVIMOT®-taajuusmuuttaja ohjaa jarrua. Ohjearvo f1
"1" "0"	"0" "1"	"1"	Laite aktivoitu	Ei laite-virhettä	MOVIMOT®-taajuusmuuttaja ohjaa jarrua. Ohjearvo f2
"1" "0"	"1" "0"	"0"	Laitetta ei ole aktivoitu	Ei laite-virhettä	Jarru on kiinni.
"1"	"1"	"1"	Laitetta ei ole aktivoitu	Ei laite-virhettä	Jarru on kiinni.
"0"	**"0"**	**"1"**	**Laitetta ei aktivoitu**	**Ei laite-virhettä**	**Jarru vapautetaan manuaalista ajoa varten.**[1)]
Kaikki tilat mahdollisia			Laitetta ei ole aktivoitu	Laitevi-ka	Jarru on kiinni.

1) Expert-tilassa parametrin P600 (liitinten konfigurointi) on lisäksi oltava = "0" (oletus) => "Ohjearvon vaihto vasen/seis - oikea/seis".

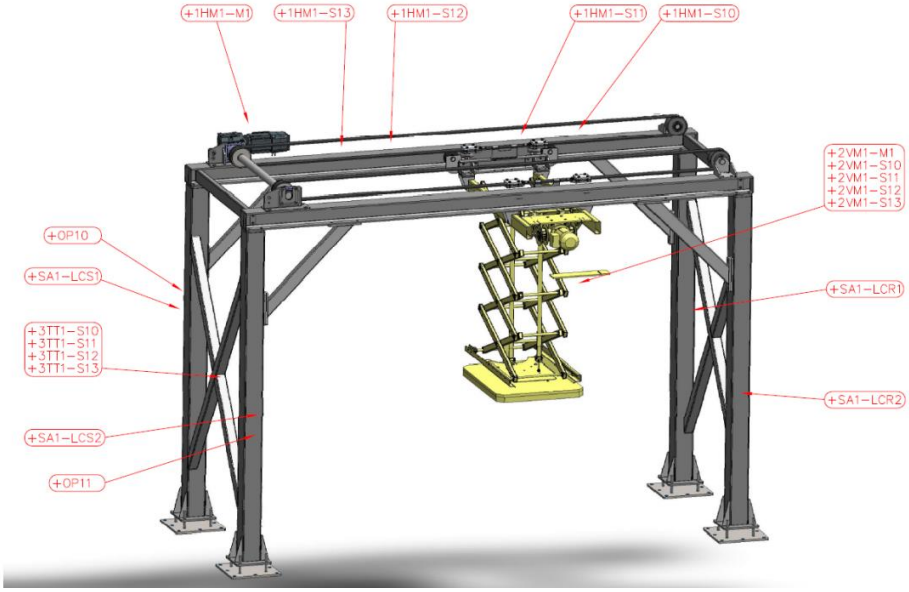
Kuva 19. Moottorin ohjauksen ohjearvot

5 TOIMINNANKUVAUS

Manipulaattorissa on kaksi moottoria – 1HM1-M1 vaakaliikkeelle ja 2VM1-M1 nostoliikkeelle. Moottoreita käytetään laitteen automaattiohjelmassa, jonka lisäksi niitä voidaan ohjata manuaalisesti ohjauspaneelin kautta valitsemalla ajettava moottori ja vääntämällä paneelin alla olevaa vääntökytkintä haluttuun suuntaan.

Manipulaattori muodostaa yhden automaattialueen ja sen turvalaitteina toimivat valoverhoparit LC1 ja LC2 sekä nappikoteloista ja ohjauspaneelista löytyvät hätäseispainikkeet. Jos yhdenkään turvalaitteen piiri katkeaa, molempien moottoreiden liikkeet

pysäytetään. Turvapiiri ei kuitenkaan vaikuta nostimen tarttujaan, jotta mahdollisesti kyydissä oleva taakka ei tippuisi kenenkään tai minkään päälle.



Kuva 20. Manipulaattorin 3D-mallinnos

Automaattiohjelman tarkoituksena on ohjata manipulaattoria niin, että se ottaa kuormalavalta laatikoita kerroksittain pois ja siirtää ne vieressä olevalle rullapöydälle työntekijöiden käsiteltäväksi. Laitteen yksi ohjelmakierto tulee valmiiksi, kun laatikot on jätetty rullapöydälle ja nostin on ajettu pöydän ylle odottamaan ohjelman seuraavaa käynnistyskuittausta. Automaattiohjelma voidaan kuitata käyntiin, kun turvapiiri on ehjä ja rullapöydän anturit eivät havaitse laatikoita. Manipulaattorin automaattiohjelma on toteutettu kahdessa eri toimintalohkossa, joissa ohjataan erikseen moottoreiden 1HM1 ja 2VM1 toimintoja.

Manipulaattorin nostimessa on pulssipyörä, jonka avulla voidaan tarkkailla nostimen asentoa. Pulssipyörän ylittäessä ohjelmaan annetun raja-arvon, ohjelmassa asetetaan lavan vaihto päälle, jolloin nostin jätetään rullapöydän ylle odottamaan ohjauspaneelista tehtävää kuittausta. Lava on kuitattava vaihdetuksi, jotta automaattiohjelmaa voidaan jatkaa. Vaihto voidaan kytkeä päälle myös ohjauspaneelin ”Lavan vaihto” -painikkeesta, jos kuormalava pitää vaihtaa ennen kuin se tyhjenee.

Laitteen ohjauspaneelissa on myös painike, jota painamalla manipulaattori ajetaan siirron rajojen ohi laitteen huoltoasemaan. Nostin ajetaan automaattisesti painikkeen aktivoiduttua huoltorajalle, jos kyydissä ei ole laatikoita. Huollon valmistuttua, laite tulee

kuitata käyntiin paneelin ”Huolto Valmis” -painikkeesta, jonka jälkeen jatketaan automaattiohjelman toimintoja. Nostin voidaan ajaa huoltorajalle myös käsiajolla.

Kuitatun huollon tai lavan vaihdon jälkeen nostimen hidastusrajalle etsitään uusi arvo. Kummassakin tapauksessa kuittauksen jälkeisellä nostolla nostinta ajetaan alas hitaalla nopeudella niin kauan, kunnes laatikko on tunnistettu ja imu aktivoitu. Imun aktivoituessa pulssipyörän sen hetkisestä arvosta vähennetään 10 ja se siirretään hidastusrajana toimivan muuttujan arvoksi. Myöhempien nostojen yhteydessä hidastusrajalle annetaan uusi arvo aina silloin, kun nostimen valokennot tunnistavat laatikoiden pinnan. Hidastusrajalle oli myös mahdollista syöttää haluttu arvo ohjauspaneelin kautta.

Hidastusrajalle on etsittävä uusi arvo huollon ja lavan vaihdon jälkeen, koska rajaa kasvatetaan ohjelmassa jokaisen noston yhteydessä. Ilman hidastuksen hakua nostin siis ajettaisiin nopealla nopeudella laatikoita päin. Ensimmäiselle hidastusrajalle ei voitu myöskään antaa kiinteätä arvoa, sillä manipulaattorilla tyhjennettävät lavat eivät aina olleet saman korkuisia.

Manipulaattorin ohjauspaneelin kautta on myös mahdollista nollata laite sekä seurata sen toimintojen ja laitteiston tilaa. Laitteen mukana toimitettiin asiakkaalle käyttöohjekirja, jossa sen kuvataan sen toimintoja ja niiden ehtoja yksityiskohtaisemmin.

6 LOGIIKKAOHJELMA

Ohjelmointi aloitettiin luomalla TIA Portal -projekti, johon ryhdyttiin rakentamaan manipulaattorin laitekonfiguraatiota sekä ohjelmakoodia. Projekti nimettiin AMH:n käytäntöjen mukaan projektinumerolla 20390.

Ohjelmoinnissa hyödynnettiin AMH:n vuonna 2019 tekemää ohjelmaprojektia, jota käytettiin mallina useiden manipulaattorin toimintojen toteuttamisessa.

Logiikkaohjelma pyrittiin toteuttamaan IEC 61131 standardia sekä AMH:n aiempia ohjelmointimenetelmiä noudattaen.

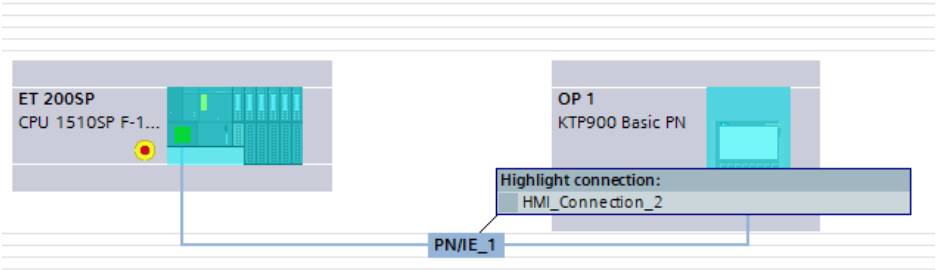
6.1 Laitekonfiguraatio

Suuri osa projektin laitekonfiguraatiosta saatiin ladattua TIA Portal -ohjelmointiympäristöön valmiiksi luodun TIA Selection Tool -tiedoston avulla. Kun laitteisto saatiin ladattua ohjelmaan, komponentteja piti vielä konfiguroida I/O-osoiteavaruuden, turvatietojen, virransyöttöjen ja IP-osoitteiden osilta. Käytännössä I/O-moduuleille annettiin oikeat osoitealueet ja niiden virransyöttö muokattiin fyysisten laitteiden mukaisiksi. Lisäksi ohjelmaan lisättiin manuaalisesti laitteen ohjauspaneeli, jota ei ollut Selection Tool -tiedostossa.

TIA Portal -projektiin tuodut laitteet olivat:

- ET 200SP CPU 1510SP F-1 PN x1 (PLC)
- F-DI8x24VDCHF x2 (Turva sisääntulokortit)
- F-DQ4x24VDC/2APMHF x1 (Turva ulostulokortti)
- DI8x24VDCHF x7 (Sisääntulokortit)
- DQ8x24VDC/0.5AHF x5 (Ulostulokortit)
- KTP900 Basic PN x1 (Ohjauspaneeli)

Lopuksi PLC ja HMI kytkettiin samaan IP-avaruuteen ja niiden välille luotiin yhteys virtuaalisella ethernet-kaapelilla.



Kuva 21. PLC:n ja HMI:n yhteys

6.2 I/O-Lista

TIA projektin tunnistetauluun (Default tag table) luotiin sähkökuvien perusteella laitteen sisään- ja ulostulot. Tageja eli tunnisteita käytettiin ohjelmassa fyysisten

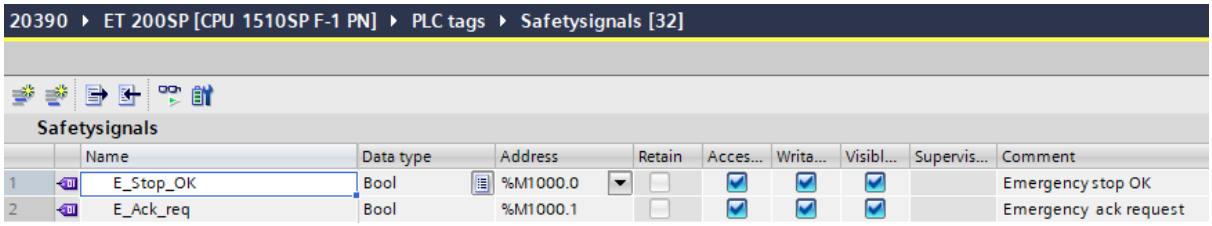
toimilaitteiden ja antureiden hallintaan ja seurantaan. Lisäksi tunnistetauluun luotiin logiikan sisäisiä globaaleja muuttujia.

Tauluun tuotiin käytännössä laitteen kaikkien korttien koko osoiteavaruus. Kaikkia osoitteita ei kuitenkaan otettu suoraan käyttöön, vaan ne luotiin varalle odottamaan mahdollisia laitelisäyksiä ja muutoksia. Varamuuttujat nimettiin tauluun nimellä ”SPARE”. Osa fyysisistä osoitteista oli liitetty laitteen turvatoimintoihin, jotka voitiin tunnistaa ohjelmointiympäristössä tunnisteen keltaisesta pohjasta.

75	C1_ESK1	Bool	%Q130.0	☐	☑	☑	☑		PÄÄKONTAKTORI SA1 TURVAKONTAKTORI...
76	C1_ESK2	Bool	%Q130.1	☐	☑	☑	☑		TURVAPIIRI VARALLA

Kuva 22. Turvamuuttujia

Fyysisten osoitteiden lisäksi tunnistetauluun luotiin globaaleja muuttujia, joita käytettiin ohjelman turva-, nollaus-, seuranta- ja kuittaustoiminnoissa. Ohjelman luettavuuden parantamiseksi fyysisille osoitteille, turvatoimintojen signaaleille ja ohjelman muistibiteille luotiin omat tunnistetaulunsa: Default_Tag_Table, Safetysignals ja M_Bits.



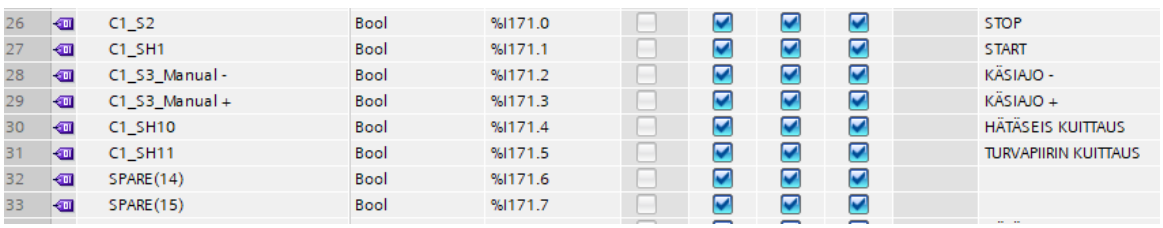
20390 ▸ ET 200SP [CPU 1510SP F-1 PN] ▸ PLC tags ▸ Safetysignals [32]

Safetysignals

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Supervis...	Comment
1	E_Stop_OK	Bool	%M1000.0	☐	☑	☑	☑		Emergency stop OK
2	E_Ack_req	Bool	%M1000.1	☐	☑	☑	☑		Emergency ack request

Kuva 23. Turvasignaalien taulu

Ohjelman tunnisteet pyrittiin nimeämään mahdollisimman selkeästi ja yhtenäisesti. Ohjelman sisäiset muuttujat luotiin muotoon Ensimmäinen_Kirjain_Isolla ja laitteen fyysiset tunnisteet nimettiin samalla nimellä kuin ne olivat sähkökuvissa. Nimeämisessä käytettiin etumerkkejä tarpeen mukaan muuttujaan liittyvän laitteen tunnistamiseen eli esimerkiksi LC1_Fault ja LC2_Fault. Kaikkien fyysisten osoitteiden nimet eivät kuvanneet erityisen hyvin niiden toimintoja, joten niistä kerrottiin tarkemmin kommenttien avulla.



26	C1_S2	Bool	%I171.0	☐	☑	☑	☑		STOP
27	C1_SH1	Bool	%I171.1	☐	☑	☑	☑		START
28	C1_S3_Manual -	Bool	%I171.2	☐	☑	☑	☑		KÄSIAJO -
29	C1_S3_Manual +	Bool	%I171.3	☐	☑	☑	☑		KÄSIAJO +
30	C1_SH10	Bool	%I171.4	☐	☑	☑	☑		HÄTÄSEIS KUITTAUS
31	C1_SH11	Bool	%I171.5	☐	☑	☑	☑		TURVAPIIRIN KUITTAUS
32	SPARE(14)	Bool	%I171.6	☐	☑	☑	☑		
33	SPARE(15)	Bool	%I171.7	☐	☑	☑	☑		

Kuva 24. Tunnistetaulu

6.3 Manipulaattorin ohjelmakierto

Manipulaattorin ohjelmakierto toteutettiin vuorottelemalla toimilohkojen 1HM1 ja 2VM1 välillä. Laitteen ohjelmakierto oli kokonaisuudessaan valmis, kun moottorin 1HM1 sekvenssi oli suoritettu alusta loppuun. 1HM1 kutsui 2VM1-moottorin ohjausta aina nostojen ja laskujen yhteydessä ja jäi odottamaan sen suorituksen valmistumista. Nostimen ohjelmassa tarkasteltiin tulotietoja, joiden perusteella voitiin päätellä, oltiinko laatikoita nostamassa tai laskemassa. Sekvenssien rakennetta ja ehtoja on kuvattu tarkemmin laitteen käyttöohjeiden toimintaselosteessa.

Laitteen ohjelmakierto toteutettiin seuraavasti:

1. 1HM1
   - Manipulaattorin ohjelmakierto alkaa lohkosta 1HM1, kun nostin on rullapöydän yllä odottamassa käynnistyksen kuittausta. Laite voidaan käynnistää, jos rullapöydällä ei ole laatikoita.
   - Käynnistyksen jälkeen nostinta aletaan ajamaan kuormalavan ylle ja sinne saavuttuaan 1HM1-ohjaus asetetaan odotusaskeleeseen ja ohjelmassa siirrytään ohjaamaan nostomoottoria.

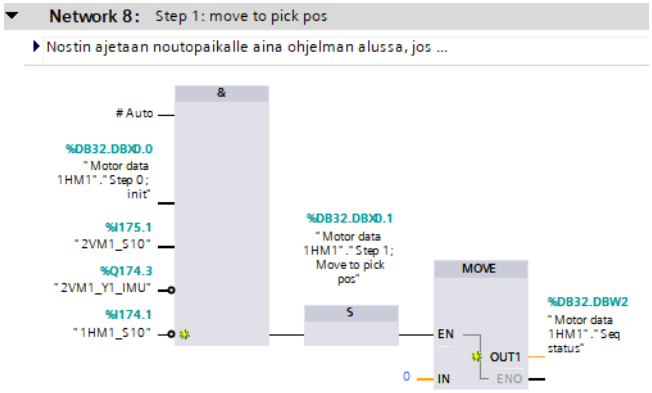
2. 2VM1
   - Nostinta aletaan ajamaan alas heti, kun se saapuu kuormalavan ylle.
   - Kerroksen tunnistusanturin tunnistaessa laatikoiden pinnan, nostin pysäytetään ja imu asetetaan päälle.
     - Nostimeen on asennettu yksi ylimääräinen imukuppi, joka nousee muiden mukana ylös niiden osuessa laatikoiden pintaan. Imukupin varsi aktivoi kerroksen tunnistusanturin sen noustessa riittävän korkealle.
   - Tarttujan imun mentyä päälle, odotetaan hetki ja nostinta aletaan ajamaan ylös.

3. 1HM1
   - Kun nostin saapuu ylärajalle, 1HM1-moottorin ohjaus siirtyy seuraavaan vaiheeseen, jossa se ajetaan luovutusasemalle. Nostimen saapuessa luovutusasemalle 1HM1-ohjaus asetetaan odotusaskeleeseen ja ohjelmassa siirrytään ohjaamaan nostomoottoria.

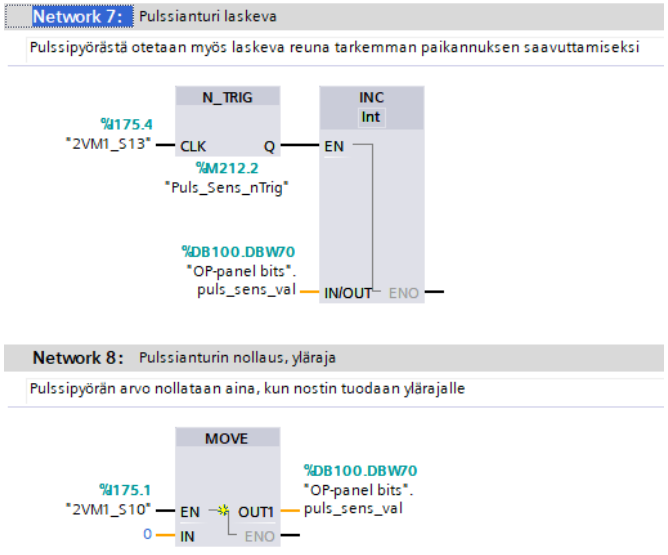
4. 2VM1
   - Nostimen saavuttua luovutusasemalle 2VM1-moottorin ohjaus alkaa uudelleen alusta. Nostinta aletaan ajamaan alas ja tarttujan imu kytketään pois, kun kerroksen tunnistus aktivoituu.
     - Nostimen ohjauksessa käytetään sekä 1HM1-moottorin että antureiden ja toimilaitteiden tilatietoja, joilla päätellään, ollaanko laatikoita viemässä vai hakemassa.
   - Kun laatikot on jätetty pöydälle, nostinta aletaan ajamaan takaisin ylös.

5. 1HM1
    - Kun nostin saapuu ylärajalle, 1HM1-ohjauksessa jäädään odottamaan käynnistyksen kuittausta. Kuittauksen jälkeen ohjelmakierto aloitetaan alusta.

Laiteen sekvenssien vaiheita sekä tietoa laitteen suorittamista toiminnoista, kuten nosteuista kerroksista, tyhjennetyistä lavoista ja nostimen asennosta esitettiin käyttäjille käyttöliittymän välityksellä. Nostimen asentoa tarkkailtiin ohjelmassa integer-muuttujalla, jonka arvoa kasvatettiin pulssipyörän pyörimisliikkeestä saatavien signaalien nousevien ja laskevien reunojen avulla. Muuttujan arvo nollattiin aina, kun nostin tuotiin ylärajalle.



Kuva 25. Sekvenssin vaiheen kirjoittamien paneeliin



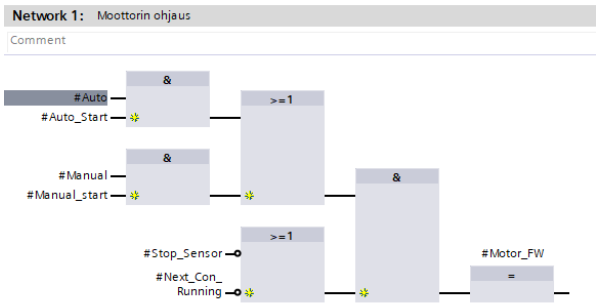
Kuva 26. Pulssipyörän nollaus

6.4 Moottoreiden ohjaus

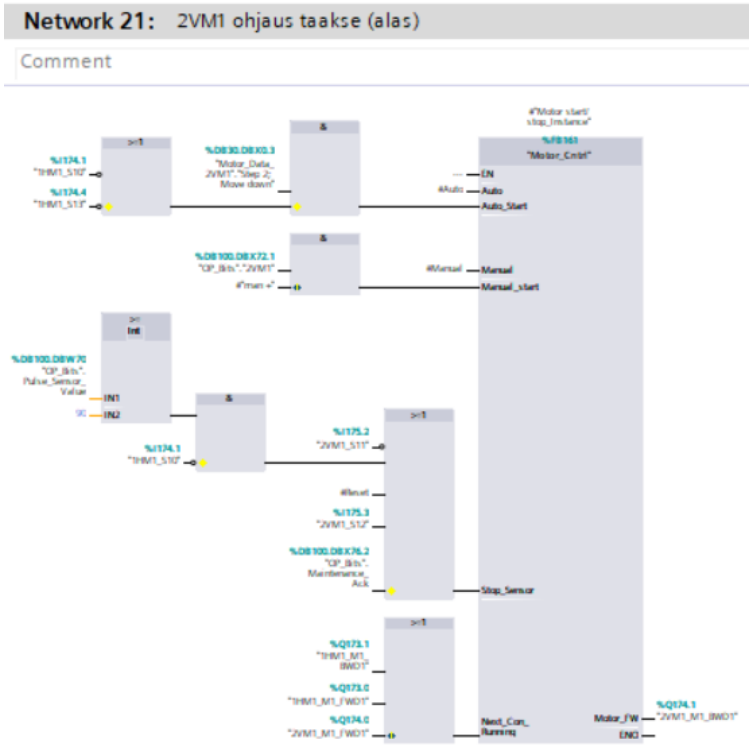
Manipulaattorin ohjelmakiertoa ja muita laitteen toimintoja ohjattiin moottoreiden omista ohjelmalohkoista 1HM1 ja 2VM1. 2VM1-moottorin lohkossa ohjattiin liikkeiden ja ohjelmakierron lisäksi tarttujan toimintaa.

6.4.1 Liikkeenohjaus

Moottoreiden liikkeenohjaus toteutettiin luomalla Motor_Cntrl -toimilohko, jossa määriteltiin ehdot moottoreiden ulostulojen ohjaamiseen. Lohkoa kutsuttiin kummankin moottorin ohjauksessa kaksi kertaa, jotta niitä voitiin ajaa eteen ja taakse.



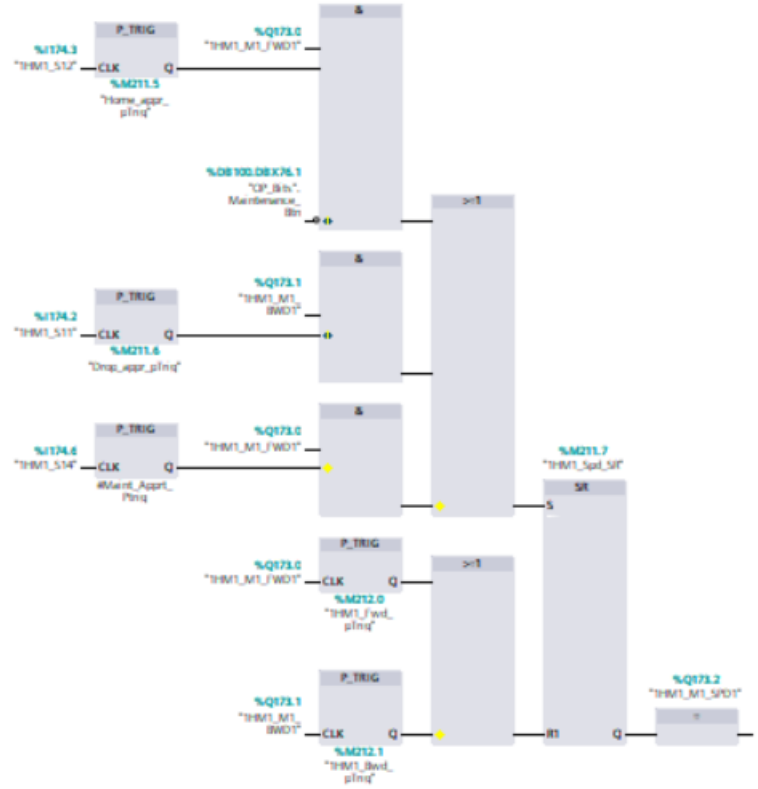
Kuva 27. Motor_Cntrl



Kuva 28. Motor_Cntrl instanssikutsu

6.4.2 Nopeuden säätö

Moottoreiden nopeutta ohjattiin omissa virtapiireissään kummankin moottorin ohjelmalohkoissa. Moottoreita alettiin aina ajamaan nopealla nopeudella noston aloituksen hidastusta lukuun ottamatta. Hidastukset kytkettiin pääsääntöisesti päälle aina, kun laite oli hidastausrajalla sekä silloin, kun pulssipyörän arvo ylitti annetut raja-arvot.



Kuva 29. 1HM1 moottorin nopeuden säätö

Moottoreiden nopeutta ohjattiin binäärisesti, joka käytettyjen moottoreiden kohdalla tarkoitti sitä, että nopeuden ulostulon ollessa päällä moottori kävi hitaalla nopeudella ja sen ollessa pois päältä moottoria ajettiin nopealla nopeudella. Moottorin varsinaista nopeutta pystyttiin säätämään ruuvaamalla moottorin kyljessä olevaa nopeuden ohjearvon säätöruuvia f2. (SEW-EURODRIVE 2014, 64.)

6.4.2 Kytkin f2

Kytkimellä f2 on käyttötavasta riippuen erilaisia toimintoja:

- Binääriohjaus: Ohjearvon asetus f2 (f2 valitaan liittimestä f1/f2 X6:7,8 = "1")
- Ohjaus RS485:n kautta: Minimitaajuuden f_{min} asetus

Kytkin f2											
Asento	**0**	1	2	3	4	5	6	7	8	9	10
Ohjearvo f2 [Hz]	**5**	7	10	15	20	25	35	50	60	70	100
Minimitaajuus [Hz]	**2**	5	7	10	12	15	20	25	30	35	40

6.4.3 Kytkin t1

Kytkintä t1 käytetään MOVIMOT®-käyttölaitteen kiihdytyksen säätämiseen. Ramppiaika perustuu ohjearvon askelmuutokseen, jonka suuruus on 1500 1/min (50 Hz).

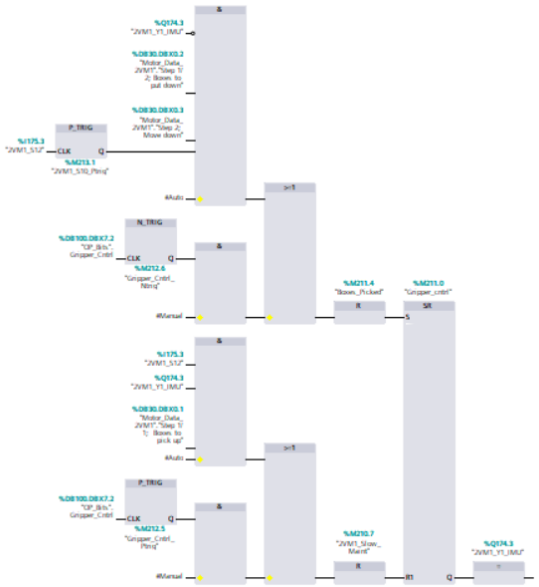
Kytkin t1											
Asento	0	1	2	3	4	**5**	6	7	8	9	10
Ramppiaika t1 [s]	0,1	0,2	0,3	0,5	0,7	**1**	2	3	5	7	10

Kuva 30. Moottorin f2/t1 paremetrointi

6.4.3 Tarttuja

Nostimen imukuppitarttujaa ohjattiin 2VM1-lohkossa. Tarttujan ulostulon Q174.3 ollessa pois päältä tarttujan imu oli päällä ja päinvastoin. Ohjaus toteutettiin tällä tavalla, jotta esimerkiksi turvapiirin rikkoutuessa kyydissä olevaa kuormaa ei irrotettaisi tarttujasta.

Tarttujaa ohjattiin automaattiohjelmassa päälle tai pois tietyissä ohjelman askeleissa tiettyjen ehtojen täytyttyä. Tarttujan ohjauksessa kirjoitettiin myös apumuuttujiin arvoja, joita käytettiin joidenkin sekvenssien ja nollausten ehtoina. Tarttujaa oli mahdollista ohjata myös ohjauspaneeliin tehdyn painikkeen kautta, jos nostomoottori oli asetettu käsiajotilaan.

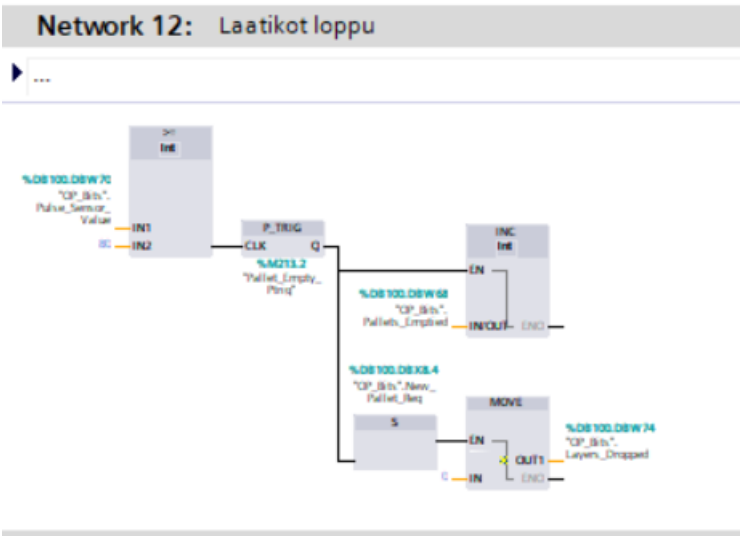


Kuva 31. Tarttujan ohjaus

6.4.4 Huoltoajo ja lavanvaihto

1HM1-lohkoon tehtiin toiminnallisuudet lavan vaihtoa ja huoltoajoa varten. Automaattiohjelmassa voitiin tunnistaa tyhjä lava laatikoiden noston yhteydessä, jonka lisäksi lavan vaihto voitiin asettaa aktiiviseksi ohjauspaneelin painikkeesta. Huoltoajo pystyttiin kytkemään päälle ainoastaan paneelin painikkeesta.

Ohjelmassa seurattiin pulssipyörän arvoa, jonka ylittäessä ennalta määritellyn raja-arvon lavan vaihdon muuttuja kytkettiin päälle. Raja-arvon ylittyessä voitiin olla varmoja, että lavalla ei ollut enempää laatikoita.



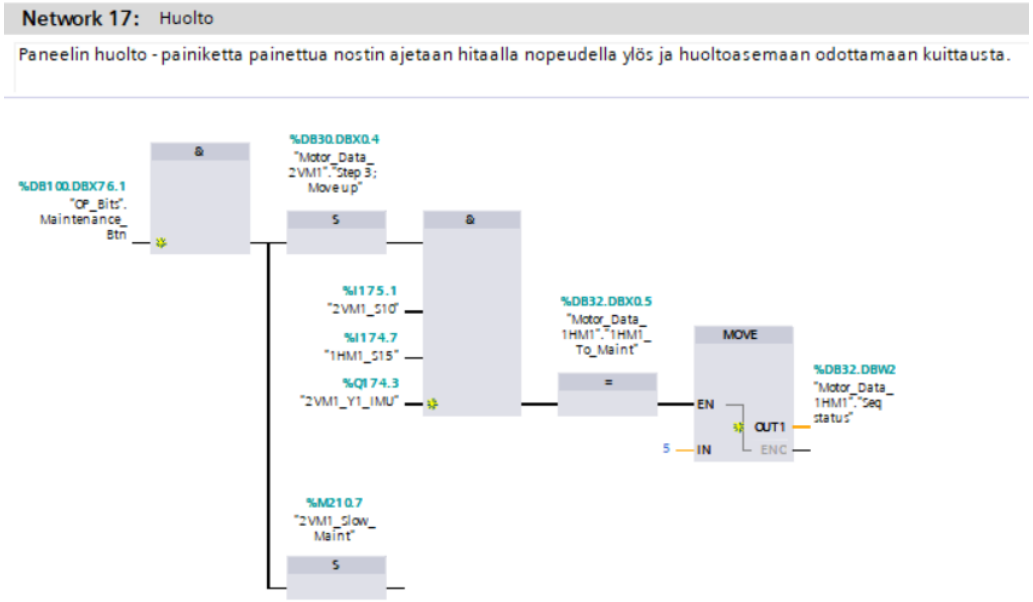
Kuva 32. Laatikot loppu

Lavan vaihdon aktivoiduttua mahdollisesti käynnissä ollut ohjelmakierto ajettiin loppuun, jonka jälkeen manipulaattori jätettiin rullapöydän ylle odottamaan, että lava kuitattaisiin vaihdetuksi. Vaihto voitiin kuitata ohjauspaneeliin tehdystä painikkeesta. Automaattiohjelman käynnistäminen oli estetty, jos kuittausta ei ollut tehty.

Lavan vaihdon jälkeen ensimmäinen laatikoiden nostaminen toteutettiin niin, että nostinta ajettiin koko matkan alas ainoastaan hitaalla nopeudella. Hidastus piti asettaa päälle, koska vanhan hidastusrajan arvo ei pitäisi paikkaansa enää uuden lavan yhteydessä. Hidastukselle annettiin uusi arvo kerroksen tunnistuksen yhteydessä, jolloin pulssipyörän arvosta miinustettiin kymmenen ja se siirrettiin hidastuksen raja-arvoksi.

Huoltoajossa nostin ajettiin siirron rajojen ohi huoltoasemaan. Automaattiohjelmassa ei ollut mahdollista ajaa huoltorajalle, ellei huoltoajoa ollut aktivoitu. Rajalle oli kuitenkin mahdollista ajaa käsiajolla.

Huoltoajo voitiin kytkeä päälle ainoastaan ohjauspaneeliin tehdystä painikkeesta. Huoltoajon aktivoiduttua nostin ajettiin huoltorajoille sillä edellytyksellä, että nostimen imu ei ollut päällä. Imun ja huollon ollessa päällä ohjelmaan muodostettiin häiriö ja painike nollattiin. Kun laite oli ajettu huoltorajalle ja huolto oli saatu valmiiksi, ohjelma tuli kuitata käyntiin ohjauspaneelin painikkeesta ”Huolto Valmis”, jonka jälkeen ohjelma voitiin kuitata käyntiin. Huollon jälkeisellä ensimmäisellä nostolla kuormalavan pintakerros haettiin samalla tavalla kuin lavanvaihdon jälkeen.



Kuva 33. Huoltoajo

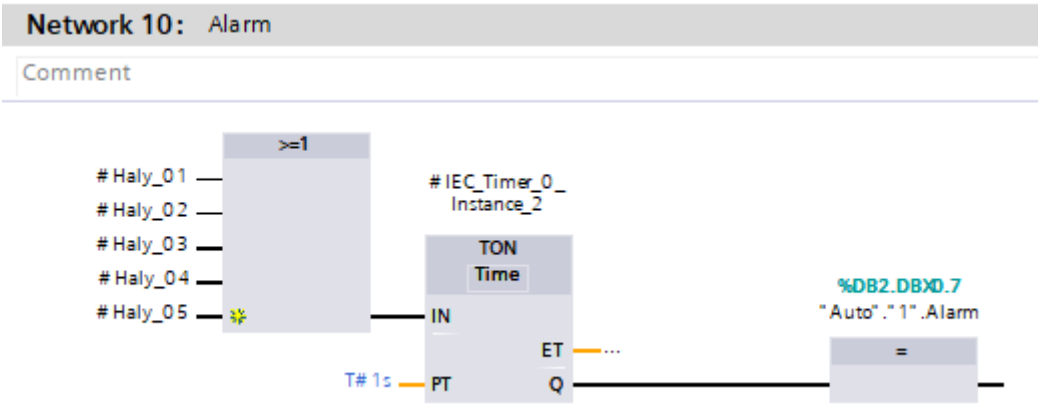
6.5 Ajotavan valinta

Manipulaattorin ajotavan valinta toteutettiin erillisessä funktiossa, jota kutsuttiin muiden lohkojen tapaan ohjelman Main [OB1] -lohkossa. Automaattiryhmällä oli neljä mahdollista tilaa: seis-, käsiajo-, automaatti- ja häiriötila.

Laite voitiin kytkeä automaatille painamalla paneelissa olevaa automaattialueen painiketta tai laitteen ”Start”-painonappeja. Automaattiajon käynnistämisen edellytyksenä oli, että manipulaattorin turva-alue oli kunnossa eikä laitteella ollut aktiivisia

hälytyksiä. Onnistunut automaatin kytkeminen nollasi käsiajon ja asetti kahden sekunnin viiveen kuluttua laitteen automaattiohjaukselle. Käsiajo taas voitiin kytkeä päälle painamalla paneelin ”1HM1 Vaakaliike” tai ”2VM1 Nostin” -painikkeita. Paneelin painikkeet nollasivat toinen toisensa, joten käsiajolla voitiin ajaa ainoastaan yhtä moottoria kerrallaan. Automaatti- ja käsiajot nollattiin, jos nappikoteloiden tai paneelin stop-painikkeita painettiin ja samalla kytkettiin seis-tila päälle.

Ohjelmassa kerättiin kaikkien järjestelmän hälytysten tilatietoja yhden muuttujan taakse, jonka aktivoiduttua häiriötila asetettiin päälle. Samalla automaatti- ja käsiajotilat nollattiin. Automaattialueen tilasta välitettiin jatkuvasti tietoa ohjauspaneeliin.



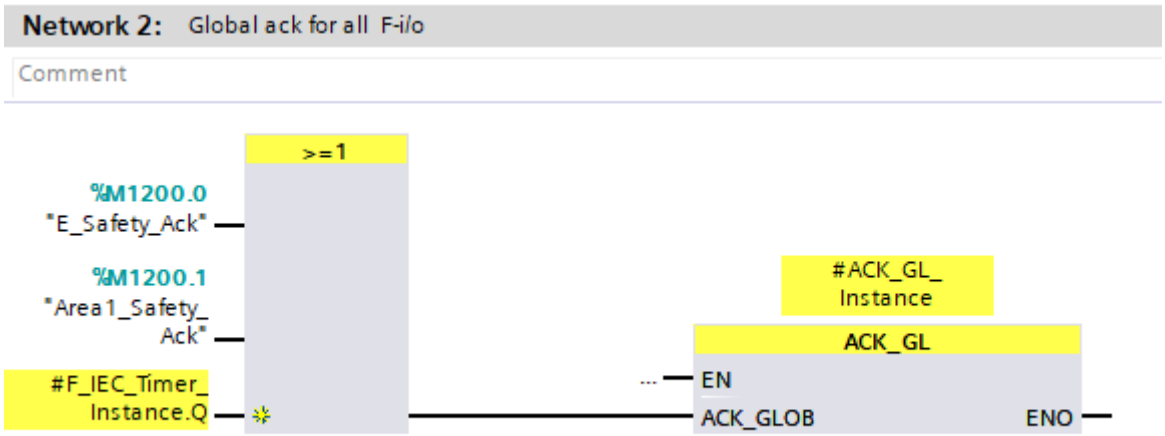
Kuva 34. Hälytykset

Funktiossa toteutettiin myös ohjelmassa muodostettujen hälytysten kuittaus. Laitteen kuittauspainikkeita painamalla ohjelmassa kirjoitettiin Data Blockien DB400 ja DB401 kaikkien hälytysten muuttujien arvoksi 0. Hälytykset tulivat kuitenkin takaisin, jos varsinaista vikaa ei korjattu.

6.6 Turvatoiminnot

Ohjelman turvatoiminnot toteutettiin TIA Portalin Main_Safety_RTG1 turvaohjelmassa. Turvalohkossa toteutettiin fyysisten turvatulojen valvonta, hälytysten varsinaiset kuittaukset, hätäseis- ja valoverhojen toiminnallisuudet ja niistä johdetut takaisinkytkennät. Turvalaitteiden tilatietoja ohjattiin erillisiin muuttujiin, joita käytettiin pääkontaktorin ohjaamisessa, jonka avulla taas voitiin kytkeä manipulaattorin laitteet virrattomiksi.

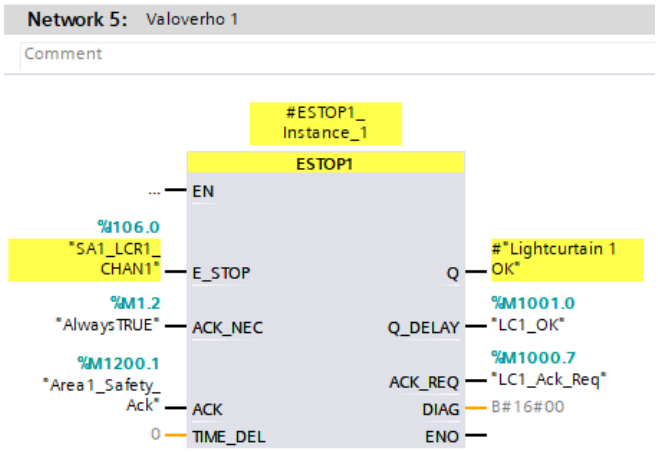
Hälytysten kuittaukseen käytettiin ACK_GL-instanssia, joka kuittasi kaikki turvaohjelmaan liitetyt turva-I/O-kortit. Kuittaukseen käytettiin muistibittejä E_Safety_Ack ja Area1_Safety_Ack, joiden toiminnallisuudet toteutettiin ohjelman Safety_Ack -ohjelmalohkossa. (TIA Information System, ACK_GL)



Kuva 35. Hälytysten kuittaus

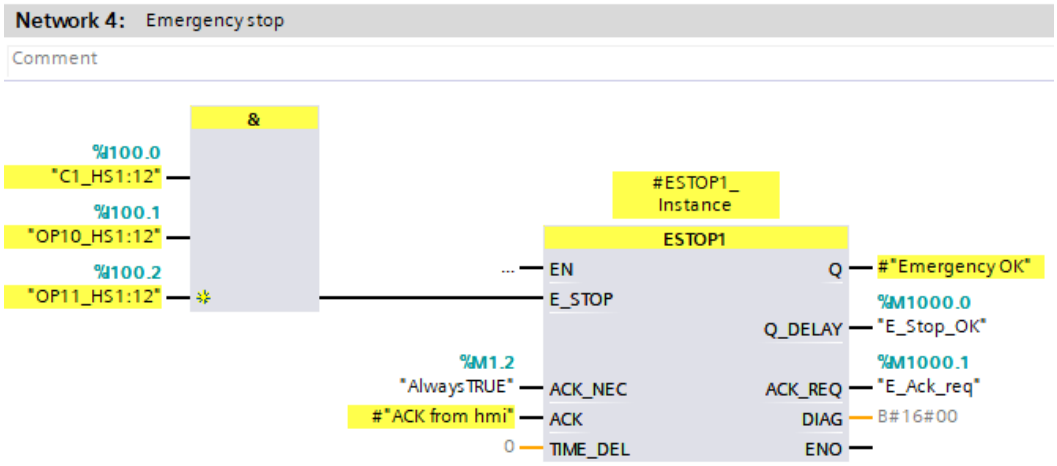
6.7 Valoverhot ja hätäseispiiri

Manipulaattorissa oli kaksi valoverhoa, joilla valvottiin laitteen turva-alueen eheyttä. Valoverhoja ohjattiin TIA Portalin ESTOP-lohkoilla. Lohkon toiminnallisuudet sopivat hyvin valoverhojen ohjaukseen, sillä turvatoimintojen tuli laueta aina, kun valoverhossa ilmeni häiriö. Toiminnallisuuksia rakennettaessa pohdittiin, olisiko valoverhojen ohjaukseen voitu käyttää valoverhojen ohitusta silloin, kun manipulaattori oli odottamassa käynnistyksen kuittausta. Ohitusten seurauksena laite olisi voitu kuitata käyntiin ilman erillistä turva-alueen kuittausta, joka taas olisi johtanut turvallisuusriskiin.



Kuva 36. Valoverojen ohjaus

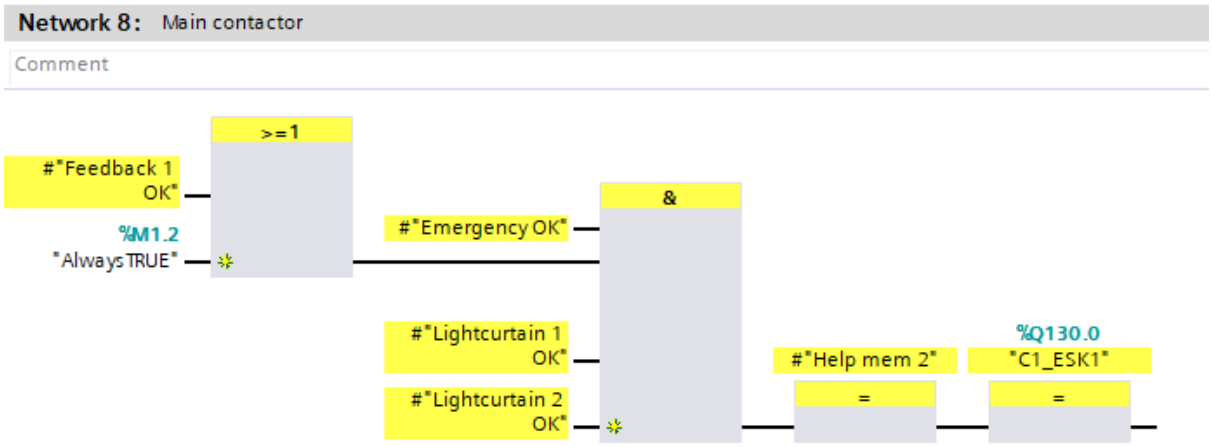
Myös laitteen hätäseis-toimintojen ohjaus oli toteutettu ESTOP-lohkolla. Manipulaattorissa oli kolme hätäseispainiketta, joita painamalla voitiin laukaista hätäseispiiri.



Kuva 37. Hätäseisvalvonta

Turvaohjelmassa seurattiin turvalaitteiden tilatietoja. Turvaohjelman lohkojen ulostuloista kerättiin tietoja erillisiin tunnisteisiin muodossa M1000.X, joita käytettiin ohjelman muissa osissa hälytysten luomiseen ja merkkivalojen ohjaamiseen.

Varsinaisia turvamuuttujia käytettiin ainoastaan pääkontaktorin ohjaamiseen, jonka syöttö asetettiin päälle, jos takaisinkytkentä, hätäseis, ja valoverhot olivat kunnossa. Turvamuuttujat, turvatulot ja turvalähdöt voi tunnistaa ohjelmasta niiden keltaisesta taustasta. Pääkontaktorilla ohjattiin tarttujan imua lukuun ottamatta kaikki muut manipulaattorin laitteet virrattomiksi, jotta laitteen liikkeet saataisiin pysäytettyä turvatoimintojen aktivoituessa niin, että mahdollisesti kyydissä oleva kuorma ei kuitenkaan tippuisi kenenkään tai minkään päälle.



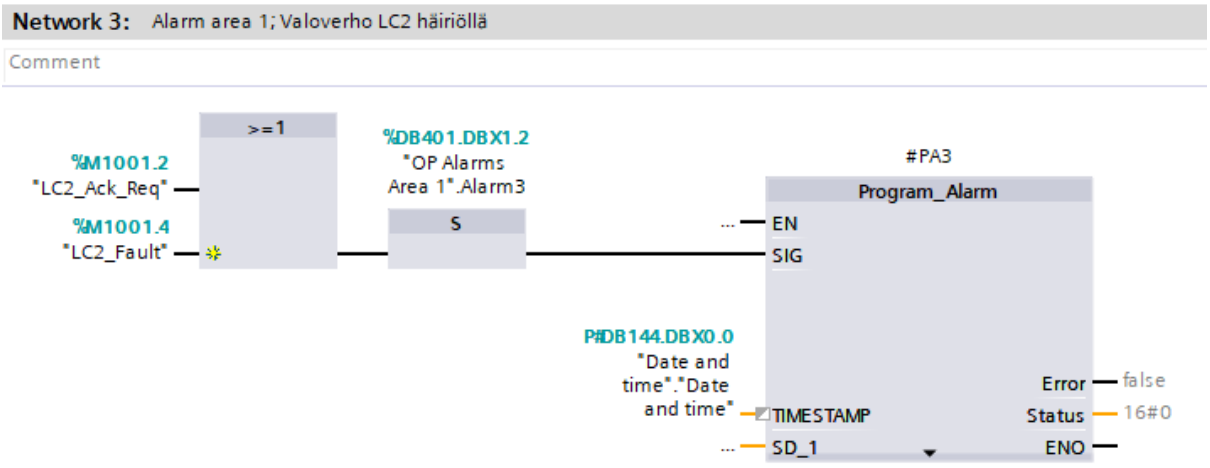
Kuva 38. Pääkontaktorin ohjaus

6.8 Hälytykset

Järjestelmän hälytykset muodostettiin kahdessa eri toimilohkossa. Alarm_Area_1 -lohkossa muodostettiin automaattialueen yleiset hälytykset ja Alarm_E-stop -lohkossa hätäseis -hälytykset.

Automaattialueen hälytyksissä seurattiin valoverhojen, pääkontaktorin, sulakkeen, turvakytkimien ja moottoreiden tiloja, kun taas hätäseishälytyksissä seurattiin ainoastaan hätäseispiirin laitteiden tilaa. Kummallekin toimilohkolle luotiin oma Data Block, johon tallennettiin ainoastaan kyseisen FB:n hälytyksiä. Erilliset Data Blockit tarvittiin, jotta hälytykset pystyttiin kuittaamaan erikseen.

Ohjelman hälytykset muodostettiin kirjoittamalla kunkin hälytyksen yksilölliseen muuttujaan arvo tiettyjen ehtojen täyttyessä SET-lohkolla. Hälytysten muuttujia luettiin esimerkiksi ajotavan valinnan funktiossa, jossa hälytyksistä muodostettiin häiriöitä ohjelmaan.



Kuva 39. Valoverhon virheen muodostus

Hälytysten muuttujia luettiin myös ohjauspaneelin ohjelmassa, jotta aktiiviset hälytykset saatiin kirjoitettua hälytyssivuille. Muuttujat tuotiin ohjauspaneelin HMI Alarms -valikkoon, jossa jokaiselle hälytykselle määriteltiin niiden yhteydessä esitettävät tekstit. Hälytykset itsessään esitettiin ohjauspaneelin Hälytykset-sivulla Alarm View -objektin avulla.

Discrete alarms

	ID	Name	Alarm text	Alar...	Trigger ..	Trigge..	Trigger address
	1	Discrete_alarm_1	Common alarm; Hätäseis lauennut	Errors	Alarm c...	0	%DB400.DBX1.0
	2	Discrete_alarm_2	Common alarm; Hätäseis painettu C1-HS1	Errors	Alarm c...	1	%DB400.DBX1.1
	3	Discrete_alarm_3	Common alarm; Hätäseis painettu OP10-HS1	Errors	Alarm c...	2	%DB400.DBX1.2
	4	Discrete_alarm_4	Common alarm; Hätäseis painettu OP11-HS1	Errors	Alarm c...	3	%DB400.DBX1.3

Kuva 40. HMI Alarms

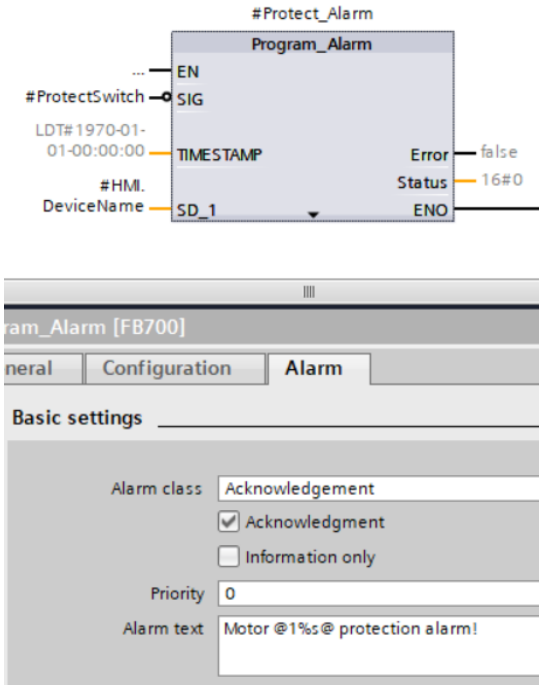
Virhetilanteiden sattuessa ohjelmassa luotiin myös Program Alarm -lohkoilla järjestelmähälytyksiä. Järjestelmähälytyksiä ei kuitenkaan käytetty esimerkkiprojektissa eikä niitä päätetty ottaa käyttöön tässäkään projektissa.

Järjestelmähälytysten avulla hälytykset olisi saatu siirrettyä ohjauspaneelille suoraan ilman erillisiä muuttujia ja Data Blockeja. Järjestelmähälytyksille ei myöskään olisi tarvinnut tehdä erillisiä hälytystekstejä ohjauspaneelin ohjelmaan sekä niillä luodut hälytykset olisi saatu kuitattua helposti yksi kerrallaan Alarm View -objektin omasta kuittauspainikkeesta. (TIA Information System, Program_Alarm)

Järjestelmähälytyksen yhteydessä esitettävä teksti kirjoitetaan suoraan lohkon sisään, joka esitetään hälytyslistoilla sen aktivoiduttua. Yksi lohko muodostaa siis aina yhden hälytyksen järjestelmään. Järjestelmähälytyksistä voidaan tehdä myös monikäyttöisiä,

jos hälytyslohko luodaan esimerkiksi moottorin ohjauksen ohjelmalohkoon, jota kutsutaan pääohjelmassa jokaisen moottorin kohdalla erikseen. (TIA Information System, Program_Alarm)

Program Alarm -lohkon SD_1-sisääntuloon voidaan halutessa liittää arvo tai muuttuja, joka sisällytetään hälytyksen yhteydessä esitettävään hälytystekstiin. Esimerkiksi string-muotoinen SD_1-tuloon liitetty muuttuja saadaan tuotua hälytystekstiin kirjoittamalla lohkon sisäiseen hälytystekstiin merkkijono @1%s@, jonka tilalle hälytyksen sattuessa kopioidaan SD_1-tulon arvo. SD_1:lle annetulla arvolla saadaan siis esimerkiksi yksilöityä kunkin moottorin hälytyksen teksti moottorin varsinaisen kutsun yhteydessä. (TIA Information System, Program_Alarm)

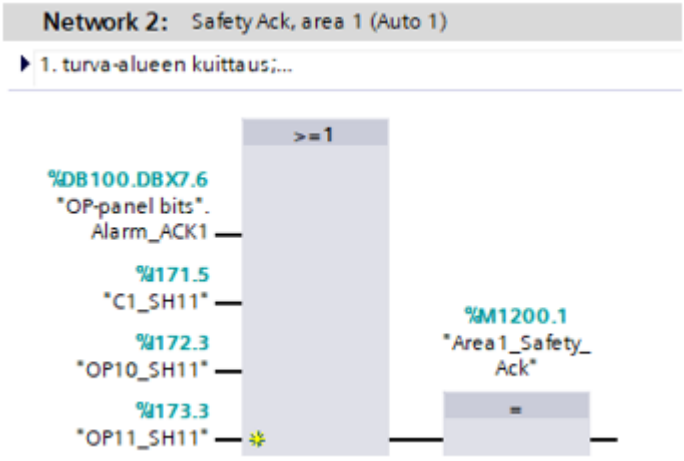


Kuva 41. Program_Alarm-lohko

Järjestelmähälytyksiä ei päätetty ottaa projektissa käyttöön, sillä manipulaattoriin tehtiin suhteellisen vähän hälytyksiä ja niistä ainoastaan kahden moottorin hälytykset olisi voitu yhdistää samaan järjestelmähälytykseen. Lisäksi laitteen häiriöiden kuittaukset haluttiin toteuttaa niin, että kaikki hälytykset pystyttiin kuittaamaan yhdellä kerralla millä tahansa laitteen kuittauspainikkeella.

6.9 Kuittaukset

Laitteen hätäseis- ja turva-alueen kuittauspainikkeiden tilatiedot ohjattiin kahteen erilliseen muuttujaan, joita käytettiin varsinaisten kuittausten toteuttamiseen. Muuttujiin liitetyt toiminnallisuudet toteutettiin erillisessä funktiossa, jotta niitä pystyttäisiin tarpeen tullen muuttamaan ohjelman käydessä. Tavallisia ohjelmalohkoja pystytään muokkaamaan ajon yhteydessä ilman logiikan pysäyttämistä. Turvaohjelma sen sijaan pysäyttää aina ohjelman, jos siihen tehdään muutoksia.

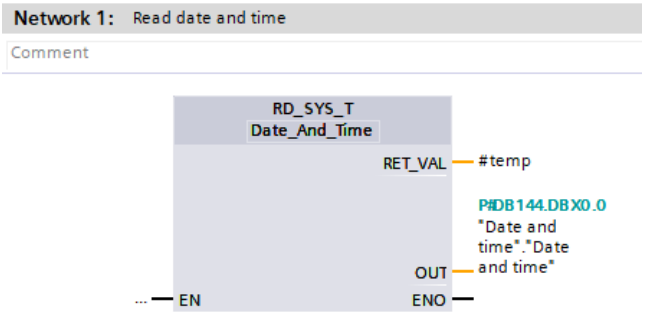


Kuva 42. Kuittaukset

Painikkeista johdetut muuttujat vietiin edelleen turvaohjelmaan, jossa niitä käytettiin ACK_GL-lohkon kuittauksessa sekä valoverhojen ja hätäseispiirin kuittaukseen.

6.10 Aika ja päivämäärä

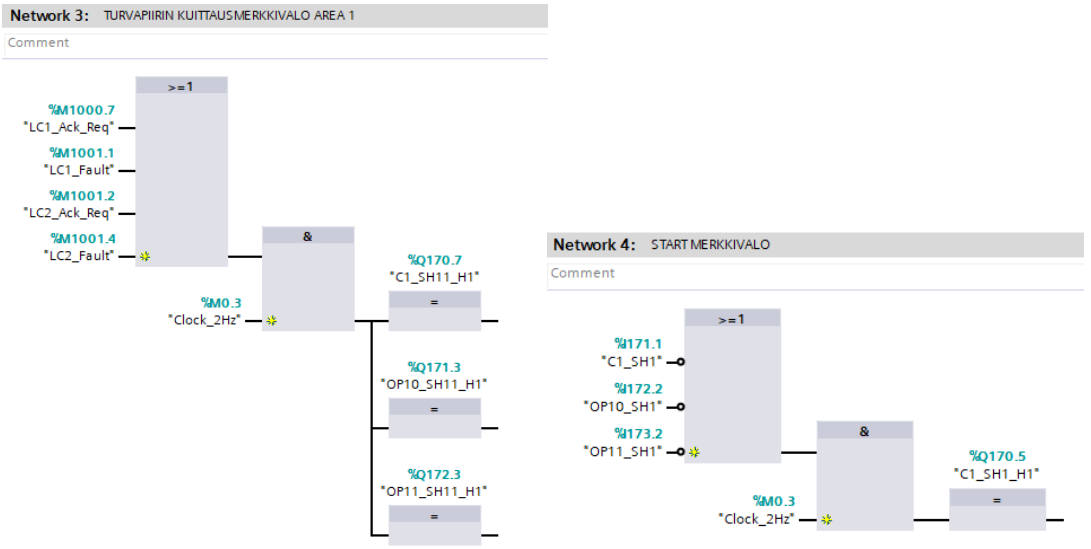
Ohjelmaan kopioitiin malliprojektista funktio, jossa muuttujaan kirjoitettiin jatkuvasti järjestelmän kellonaikaa ja päivämäärä. Muuttujaa ei kuitenkaan varsinaisesti käytetty projektissa, sillä se oli liitetty ainoastaan järjestelmähälytysten sisääntuloksi. Funktio jätettiin kuitenkin ohjelmaan, jos esimerkiksi hälytysten muodostamista haluttaisiin muuttaa jälkikäteen.



Kuva 43. Kellonaika ja päivämäärä

6.11 Merkkivalojen ohjaus

Laitteen kaikkien merkkivalojen ohjaukset kerättiin yhteen funktioon, jossa niitä ohjattiin joko turvatunnisteiden tai suoraan sisääntulojen tilatiedoilla. Funktiolla ohjattiin käynnistys- ja hätäseispainikkeiden sekä hätäseis- ja turvapiirin kuittauksen merkkivaloja.

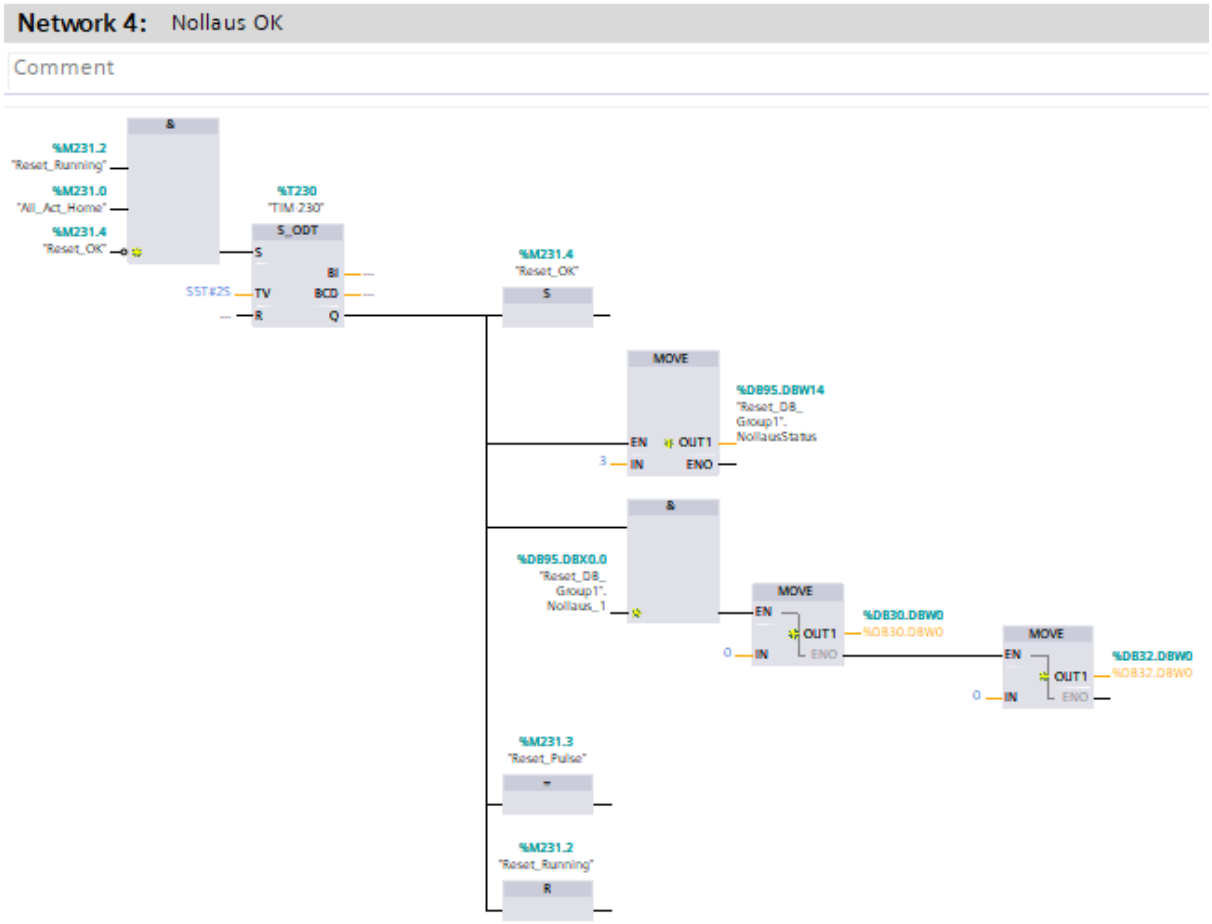


Kuva 44. Merkkivalojen ohjaus

6.12 Nollaus

Ohjelmaan tehtiin nollaustoiminto tilanteita varten, joissa laitteen ohjelmakierto jää jumiin tai se tekee muuten virheellisiä toimintoja. Toimintoa voitiin käyttää ohjauspaneelin Nollaus-sivulla. Nollaus oli mahdollista suorittaa ainoastaan, kun manipulaattorin ohjelma oli pysäytetty.

Nollauspainiketta painamalla ohjelmassa alettiin suorittamaan funktiota Reset_Group1, joka onnistuessaan asetti moottoreiden sekvenssien ohjauksen Data Blockien 1HM1_DB ja 2VM1_DB kaikkiin muuttujiin arvon 0. Nollaus oli monivaiheinen ja osaan vaiheista oli lisätty S_ODT-ajastimia. Ajastinten avulla varmistettiin, että paneelin nollauspainike ja vaiheen muut sisääntulot olivat aktiivisina riittävän kauan ennen kuin nollauksessa siirryttiin seuraavaan vaiheeseen.



Kuva 45. Nollaus

Data Blockien tyhjentämisen jälkeen ohjelmassa asetettiin vaikuttuneiden antureiden, apumuuttujien ja toimilaitteiden perusteella automaattiohjelmaan oikea askel aktiiviseksi. Ohjelman jokaiseen askeleeseen oli asetettu tietyt ehdot, joiden perusteella automaattiohjelma osasi nollauksen jälkeen jatkaa kesken jäänyttä toimintoa. Onnistuneen nollauksen jälkeen laite tuli kuitata uudelleen käyntiin.

Kuva 46. Nollaus moottorien ohjauksessa

6.13 I/O-tiedot ohjauspaneeliin

Laitteen sisään- ja ulostulojen tilatietoja välitettiin ohjauspaneeliin funktion IO-Status_To_Operator avulla. Kyseessä on Main [OB1] -lohkon ohjelmakutsuja lukuun ottamatta ohjelman ainoa lohko, joka on tehty tekstikielellä ja siinä käytettiin Instruction List -ohjelmointikieltä.

Funktioon tehtiin neljä erillistä virtapiiriä, joissa haettiin sisääntulot alueilta 100.0 – 105.7 ja 170.0 – 177.7 sekä ulostulot alueilta 130.0 – 135.7 ja 170.0 – 175.7. Kullekin virtapiirille ladattiin haettavat I/O-alueet Data Blockin IO-status ensimmäisistä muuttujista, joille oli määritelty arvoiksi 100, 130 ja 170.

Jokaisen virtapiirin toimintaperiaate oli sama. Kunkin piirin alussa ladattiin tulojen hakemisessa käytettävän muuttujan arvo logiikan ensimmäiseen rekisteriin L-komennolla, jonka jälkeen se siirrettiin T-komennolla lohkon sisäisen muistin osoitteeseen #alkuosoite, josta se ladattiin takaisin logiikan 1. rekisteriin. (TIA Information System, L)

Tämän jälkeen muuttujalle tehtiin tietotyypin muunnos SLW-komennolla integeristä 32-bittiseksi kohdistimeksi muotoon P#X.0, jolla saatu arvo ladattiin 1. osoiterekisteriin LAR1-komennolla. Tietotyyppi oli muunnettava kohdistimeksi, jotta funktion myöhemmissä osissa voitiin viitata tulojen osoitteisiin ja hakea niiden arvot. Kohdistin ei pidä sisällään erillistä arvoa, vaan osoitteen, johon sillä halutaan viitata. Kohdistimelle voidaan antaa parametrinä esimerkiksi osoite P#I170.0, jolloin sillä haetaan dataa sanasta I170 aloittaen osoitteesta I170.0. (Siemensin www-sivut 2020: Siemens 2006, 552.)

```
L     "IO-status".dbw0
T     #alkuosoite
L     #alkuosoite
SLW   3
LAR1
```

Kuva 47. Valmistelu

Seuraavaksi ohjelmassa alettiin lukemaan ja lataamaan rekisteriin tulojen tilatietoja. Kuvassa 49 esitetyssä virtapiirissä ensimmäisten sisääntulojen tilatiedot luetaan rivillä 15. Kohdistin rakennettiin L-komennon yhteydessä kertomalla, että haettavat kohteet ovat sisääntuloja (IW). Hakasulkujen sisällä viitattiin osoiterekisteriin tallennettuun arvoon, joka oli tässä tapauksessa 170. AR1:n jälkeisellä komennolla ”P#0.0” tarkennettiin, että haettavan sanan arvo alkaa luvusta 170.0.

```
L IW [ AR1 , P#0.0 ]
T     "IO-status".dbw10
```

Kuva 48. Kohdistimen rakentaminen

Rekisteriin ladattiin kohdistimen P#IW170.0 avulla sisääntulon sana alkaen osoitteesta 170.0 ja päättyen osoitteeseen 171.7. Kun tilatiedot oli ladattu rekisteriin, ne siirrettiin T-komennolla IO-status DB:n integer tyyppiseen muuttujaan dbw10, jota taas käytettiin ohjauspaneelissa I/O-tilatietojen esittämiseen. (TIA Information System, +AR1)

Samaa toimintoa sovellettiin kaikkien sisään- ja ulostulojen lukemiseen. Jokaiselle sisään- ja ulostulon sanalle luotiin Data Blockiin oma muuttuja, johon se tallennettiin.

Network 1: Inputit alueelta 170

Comment

```
1    L     "IO-status".dbw0                    %DB220.DBW0
2
3
4
5
6    T     #alkuosoite
7
8
9    L     #alkuosoite
10
11
12   SLW   3                                   3
13   LAR1
14
15   L IW [ AR1 , P#0.0 ]
16   T     "IO-status".dbw10                   %DB220.DBW10
17
18   L IW [ AR1 , P#2.0 ]
19   T     "IO-status".dbw12                   %DB220.DBW12
20
21   L IW [ AR1 , P#4.0 ]
22   T     "IO-status".dbw14                   %DB220.DBW14
23
24   L IW [ AR1 , P#6.0 ]
25   T     "IO-status".dbw34                   %DB220.DBW34
26
```

Kuva 49. Sisääntulojen haku

Funktio kopioitiin malliprojektista, jossa I/O-haku oli toteutettu niin, että ohjauspaneelin sisään- ja ulostulojen näyttösivuilta syötettiin logiikalle haettavan osoitealueen arvo. Ohjelmassa oli siis vain kaksi virtapiiriä, joista toinen haki sisään- ja toinen ulostuloja. Manipulaattorin ohjelmaan päätettiin tehdä kuitenkin erilliset virtapiirit jokaisen I/O-alueen samanaikaiseen lukemiseen, sillä laitteessa oli vähän luettavia tuloja, jotka mahtuivat hyvin ohjauspaneelin kahdelle erilliselle sivulle. Lisäksi tulojen näyttösivut haluttiin toteuttaa niin, että esitettävien tulojen yhteydessä kerrottaisiin myös niiden nimet, mikä ei olisi onnistunut, jos tarkasteltavaa osoitealuetta olisi voitu muuttaa näyttösivun kautta. Virtapiireille olisi voitu todennäköisesti antaa suoraan haettavat osoitteet viittaamatta DB:n muuttujiin, mutta malliprojektin toiminnallisuudet jätettiin ohjelmaan mahdollisten muutosten varalta.

6.14 Käyttöliittymä / HMI

Manipulaattorin ohjauspaneelina toimi Siemensin KTP 900 Basic PN -paneeli. Laitteen käyttöliittymä rakennettiin AMH-Systemsin aiemman projektin pohjalle, jossa oli

käytetty samaa paneelimallia. Käyttöliittymä suunniteltiin AMH:n käytäntöjä noudattaen ja siitä pyrittiin tekemään mahdollisimman yksinkertainen ja selkeä. Ohjauspaneeliin luotiin toiminnot käsiajolle sekä erilaisia näkymiä prosessin ja logiikan toimintojen seurantaan. Lisäksi käyttöliittymään tehtiin kaksi kielivalintaa: suomi ja englanti. Ohjauspaneeliin tehtiin 10 erillistä sivua, jotka olivat:

- Pääsivu
- Käsiajo
- Nollaus
- Diagnostiikka
    - Sisääntulot
    - Ulostulot
    - PLC
    - Asetukset
- Hälytykset
- Häiriöloki

Erillisten näyttösivujen lisäksi paneelille tehtiin pohjasivu eli template, johon luodut objektit esitettiin jokaisella näyttösivulla. Pohjasivun alareunaan tehtiin tekstikentät, joilla kerrottiin paneelin painikkeiden toiminnoista ja sen yläreunaan luotiin objekteja esimerkiksi pääsivulle siirtymiseen sekä kellonajan ja näyttösivun nimen esittämiseen.

Kuva 50. Pohjasivu

6.14.1 Käyttöliittymän suunnittelu

Käyttöliittymän toteuttaminen ei vaatinut juurikaan suunnittelua, sillä kaikki tarvittavat toiminnallisuudet saatiin kopioitua malliprojektista. Käyttöliittymän rakentaminen vaati pääasiassa HMI:n tunnistetaulun päivittämistä sekä sivujen ulkoasujen muokkaamista. Valmiista pohjasta huolimatta toimintoja ja ulkoasua pyrittiin parantamaan niiltä osin, kuin se oli mahdollista. Eniten käyttöliittymää rakentaessa jouduttiin pohtimaan, kuinka paneelin sivujen välinen navigointi tulisi toteuttaa. Vaihtoehtoina olivat:

- Pääsivu

Pääsivumallia käytettiin AMH:n esimerkkiprojektissa. Sen käyttöliittymään oli tehty pääsivu, johon oli luotu painikkeita paneelin muille sivuille siirtymiseksi. Ainoastaan pääsivulta päästiin siis siirtymään muille sivuille, mutta kaikilta muilta sivuilta päästiin palaamaan pääsivulle ohjauspaneelin pohjaan luodun painikkeen avulla.

- Pudotusvalikko

Pudotusvalikolla sivuvalinnat olisi toteutettu painikkeella, jota painamalla paneelin eri sivujen valintapainikkeet olisi saatu näkyviin tai piiloon. Ratkaisun avulla erillistä valintasivua ei olisi tarvinnut tehdä ja paneelilta olisi tarvittu hyvin vähän tilaa navigoinnin toteuttamiseen. Toiminnallisuus oli tarkoitus toteuttaa paneelin pohjaan liitetyllä popup-ikkunalla, mutta käytössä olleeseen paneelimalliin ei saanut luotua niitä, joten ratkaisuna koitettiin tavallista pudotusvalikkoa. Pudotusvalikon valintaruudut olivat sidoksissa fonttikokoon, joten valikoista tuli joko liian pieniä painettavaksi tai fonttia suurentaessa niiden tekstit eivät enää mahtuneet ruutuun.

- Pohjaan liitetyt painikkeet

Käyttöliittymän pohjasivun alareunaan lisättiin samat painikkeet, kuin käyttöliittymän pääsivulla oli, jolloin paneelin jokaiselta sivulta voitiin siirtyä muille sivuille. Käytössä ollut paneeli oli pienikokoinen, joten tilaa koitettiin säästää tekemällä painikkeista mahdollisimman pienet ja liittämällä niiden toiminnot paneelissa olleisiin fyysisiin painikkeisiin. Paneelin painikkeista tehtiin lopulta niin pieniä, että niiden painamisesta tuli hyvin hankalaa ja lopuksi painikkeiden tilalle tehtiin tekstikenttiä, joilla kerrottiin, mitä toimintoja paneelin fyysisiin painikkeisiin oli liitetty.

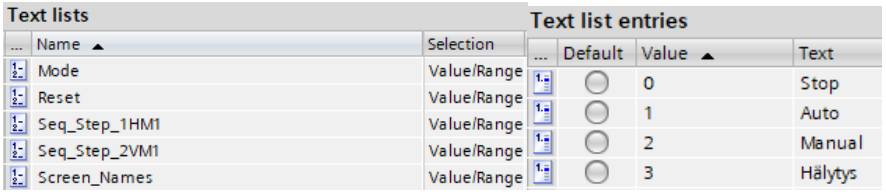
Lopulta paneeliin päätettiin tehdä pääsivu ja sen tueksi myös paneelin fyysiset painikkeet otettiin käyttöön. Pääsivulla kaikki sivuvalinnat saatiin esitettyä selkeästi ja alareunan painikkeilla sivujen vaihtaminen kävi nopeammin.

6.14.2 Yhteys logiikkaan

Paneelissa esitettiin manipulaattorin antureiden, ohjelmakiertojen, nollaustoiminnon sekä toimilaitteiden tiloja, jonka lisäksi sen kautta oli mahdollista kirjoittaa joillekin logiikan muuttujille arvoja. Tietojen esittämistä ja kirjoittamista varten paneelin omaan tunnistetauluun luotiin tageja, jotka liitettiin logiikan vastaaviin muuttujiin.

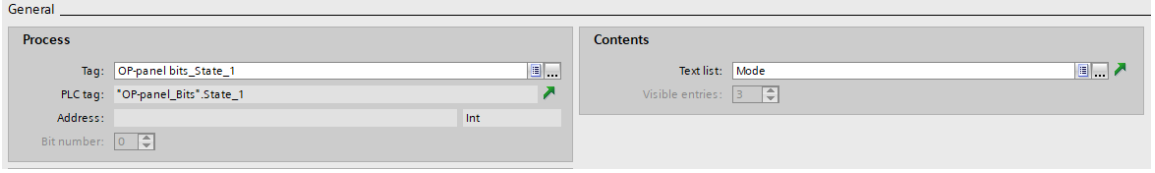
Logiikasta tuodut muuttujat olivat tietotyypeiltään joko integer- tai binäärimuotoisia. Binäärimuuttujia pystyttiin käyttämään suoraan antureiden ja toimilaitteiden tilojen esittämisissä, mutta integer-muuttujien arvo piti joissain tapauksissa kääntää tekstimuotoon, sillä niillä esitettiin esimerkiksi nollauksen ja sekvenssien tiloja.

Jotta muuttujien arvot saatiin käännettyä merkkijonoiksi, paneeliin tuli tehdä tekstilistoja, joissa numeroarvoille määriteltiin niitä vastaavat merkkijonot. Paneeliin luotiin myös PLC:stä riippumaton tekstilista, jonka avulla HMI:n pohjassa olevaan I/O-kenttään päivitettiin aktiivista sivua vastaava teksti.



Kuva 51. Tekstilistat ja niiden tekstien määrittely

Tekstejä esitettiin paneelissa liittämällä I/O-kenttään seurattava tagi ja sen sisällöksi asetettiin haluttu tekstilista. Tekstilistasta etsittiin siis tagin arvoa vastaava merkkijono, joka kirjoitettiin I/O-kenttään.

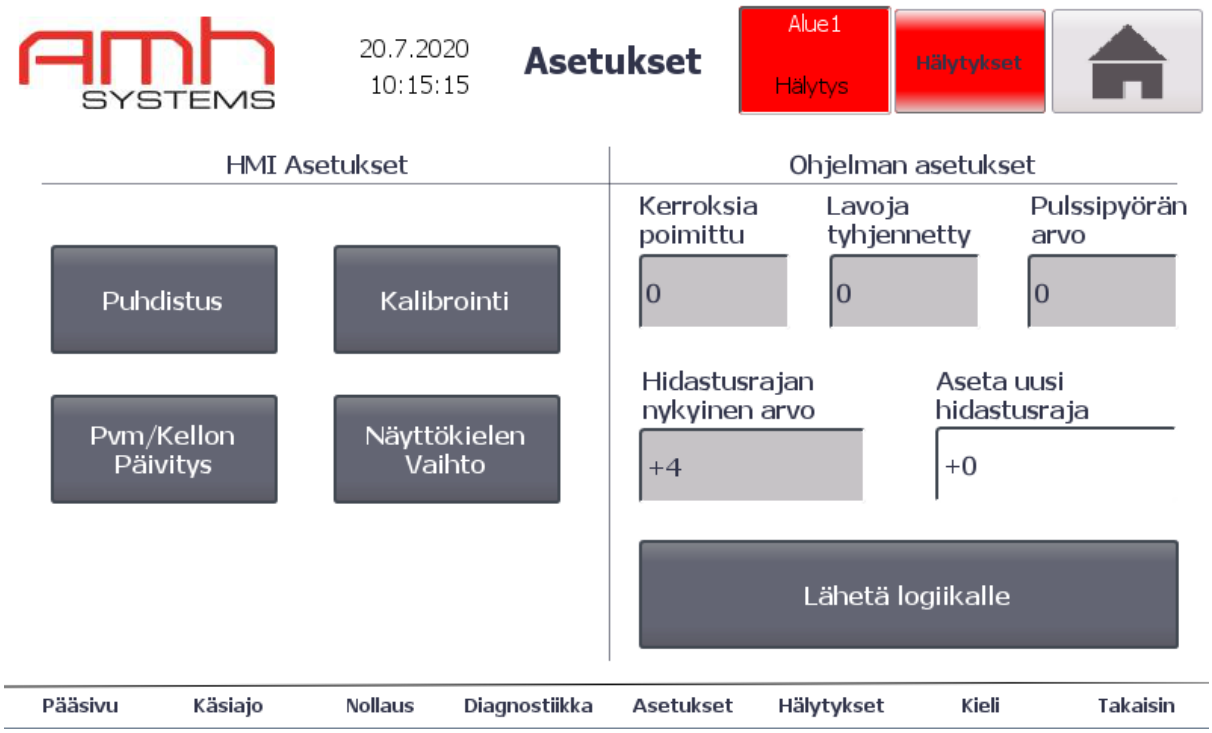


Kuva 52. Paneelissa esiteltävän tiedon käsittely

6.14.3 Asetukset

Paneelin Asetukset-sivulla oli mahdollista tehdä paneeliin liittyviä toimintoja, kuten päivittää kellonajan ja päivämäärän, kalibroida näytön sekä vaihtaa käyttöliittymän kielen. Lisäksi sivulla esitettiin ohjelman ja sen suorittamien toimintojen tilatietoja sekä sivulla oli myös mahdollista asettaa nostimelle uusi hidastusraja.

Hidastusrajan arvo tuli syöttää valkoisella pohjalla olevaan I/O-kenttään, joka siirrettiin ”Lähetä logiikalle” -painiketta painamalla logiikan hidastusrajan muuttujaan. Muiden muuttujien arvot esitettiin selkeyden vuoksi harmaalla pohjavärillä olevilla I/O-kentillä, joilla osoitettiin, että niihin ei voida kirjoittaa.



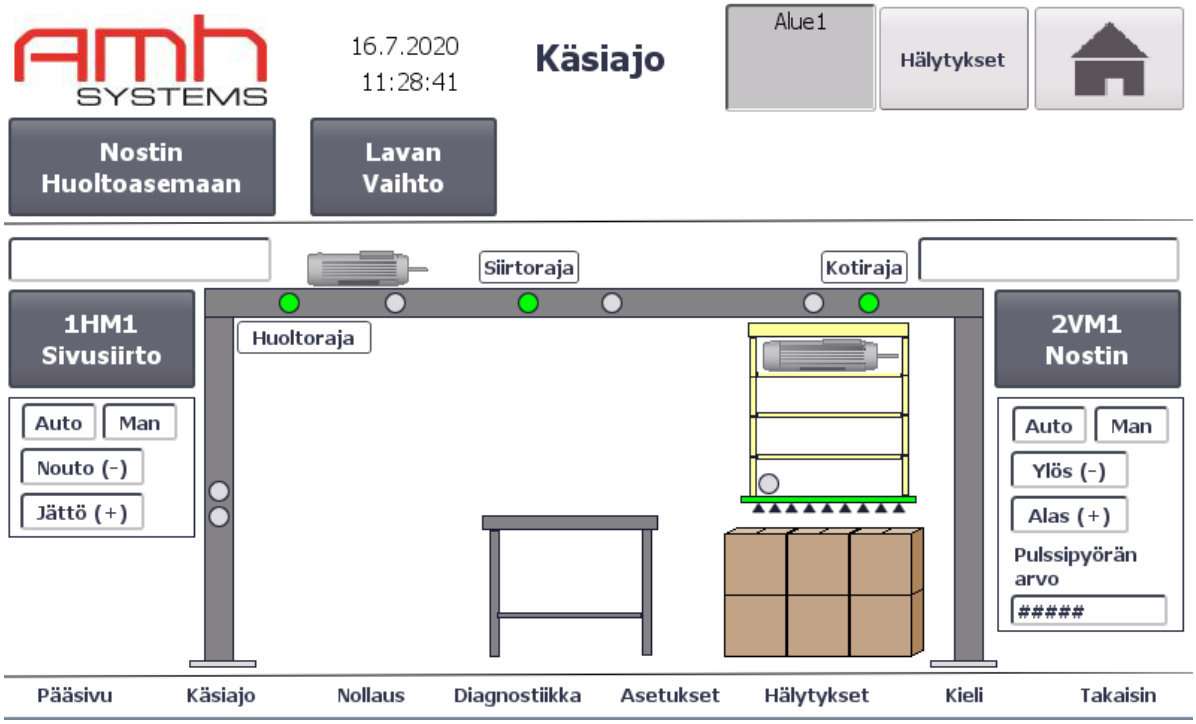
Kuva 53. Asetukset

6.14.4 Käsiajo

Manipulaattorin 1HM1- ja 2VM1-moottoreita pystyttiin ohjaamaan paneelin Käsiajo-sivulla. Käsiajolla pystyttiin ohjaamaan ainoastaan yhtä moottoria kerrallaan, sillä painamalla jommankumman moottorin laitepainiketta, toinen nollattiin. Paneeliin ei tehty moottoreiden ohjaamiseen erillistä painiketta vaan niitä pystyttiin ohjaamaan ainoastaan ohjauspaneelin alla olevalla vääntökytkimellä.

Laitepainikkeiden yllä oleviin I/O-kenttiin tuotiin tekstilistojen avulla tieto kummankin moottorin ohjauslohkon käynnissä olevasta vaiheesta. Sivulle tehtiin myös painikkeita ja tekstejä, jotka näytettiin ainoastaan tiettyjen ehtojen täyttyessä.

Nostimen 2VM1 laitepainikkeen aktivoiduttua sivulla asetettiin näkyväksi tarttujan ohjauksen painike. Painiketta ei näytetty jatkuvasti, sillä sitä voitiin käyttää ainoastaan nostinta ajettaessa. Lisäksi sivulla oli mahdollista asettaa logiikkaohjelmaan lavanvaihto tai huoltoajo aktiiviseksi. Kun painikkeita painettiin, laite ajettiin automaattisesti asiaankuuluville rajoille ja paneelissa asetettiin näkyväksi painikkeet lavan vaihdon ja huollon kuittauksiin. Sivulla esitettiin myös laitteen antureiden ja toimilaitteiden tilatietoja sivulle sijoitetuilla ympyröillä. Anturin aktivoiduttua ympyrän väri muuttui vihreäksi ja siihen liittyvä teksti tuli näkyviin.

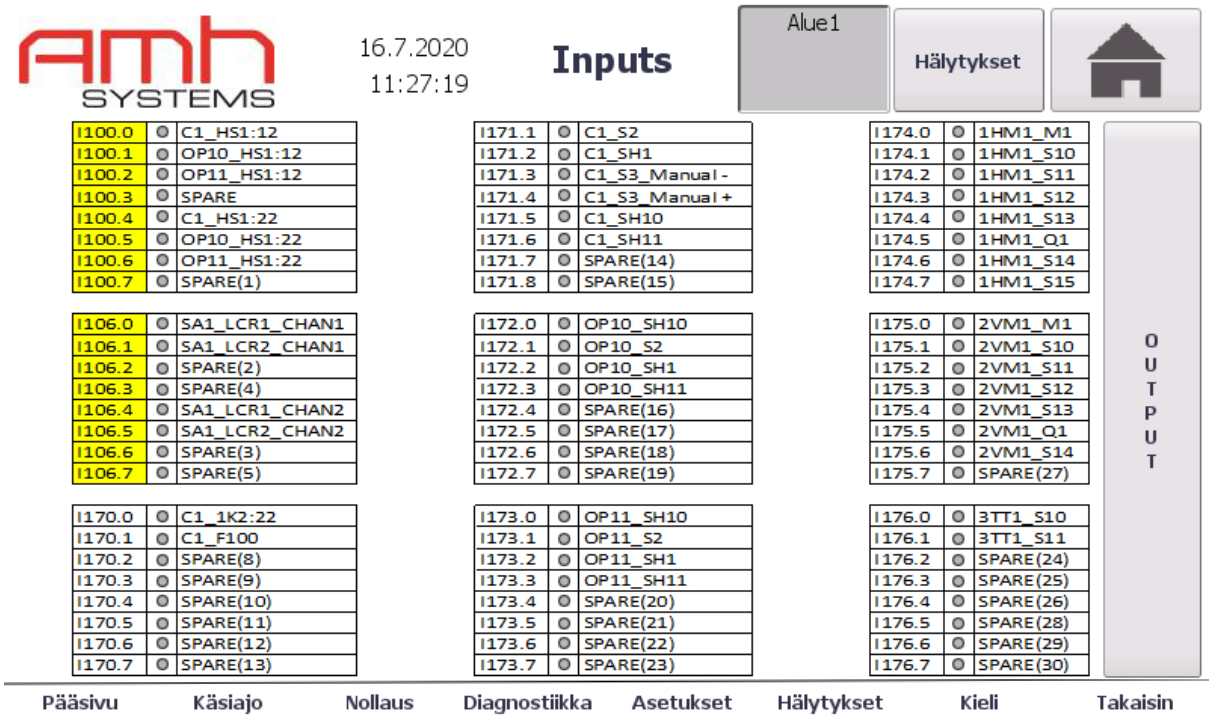


Kuva 54. Käsiajo

6.14.5 I/O Diagnostiikka

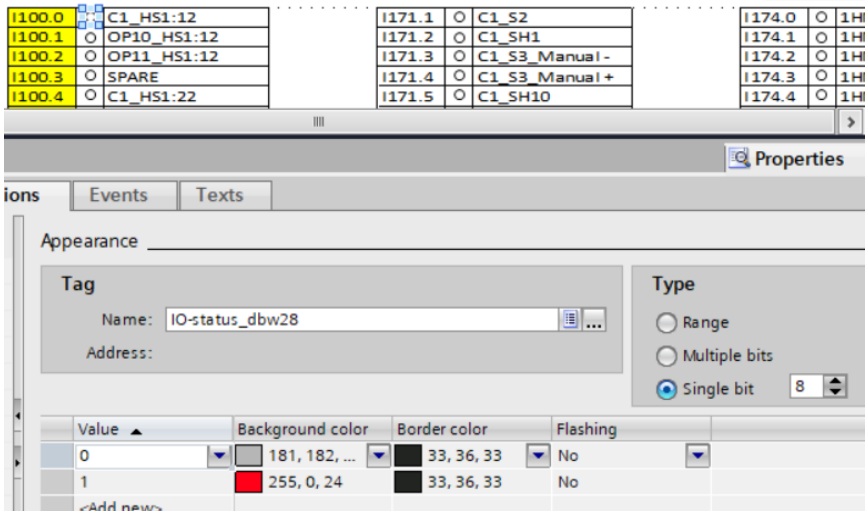
Ohjauspaneeliin diagnostiikkaryhmään tehtiin erilliset sivut, joissa esitettiin manipulaattorin sisään- ja ulostulojen tilatietoja. Tulojen tiedot siirrettiin paneelin käytettäväksi IO-Status_To_Operator -funktion Data Blockin muuttujista. Kunkin tulon tilaa

esitettiin sivulle luotujen ympyröiden avulla, joilla kullakin seurattiin tietyn osoitealueen tagia ja siitä tiettyä bittiä. Käytetyt muuttujat pitivät sisällään sanan verran tietoa, joka tarkoitti sitä, että esimerkiksi sisääntulojen 170.0 – 171.7 alue oli tallennettuna muuttujaan dbw10.



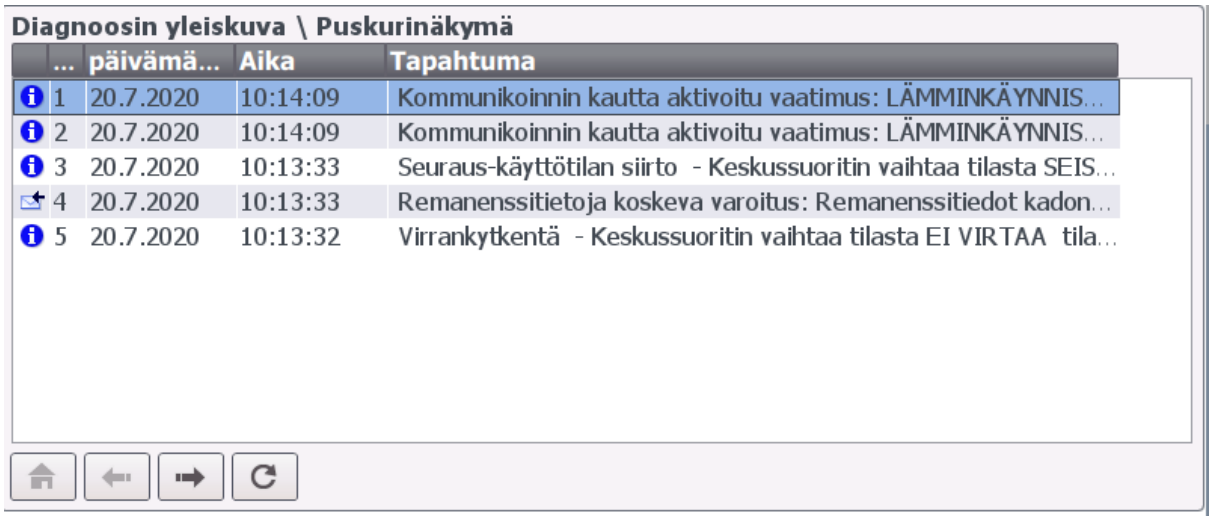
Kuva 55. Output sivu

Kuvasta 56 nähdään, kuinka keskuksen hätäseispainikkeen tilan esittäminen toteutettiin käyttöliittymässä. Sisääntulon seurantaan tehtiin objekti, joka seurasi HMI:n tunnistetauluun tuodun tagin kautta IO-Status DB:n muuttujaa dbw28. Muuttujasta seurattiin sen kahdeksatta bittiä, jonka mennessä päälle, myös objektin väri vaihdettiin punaiseksi. Tulotietoja seurattiin siis samalla periaatteella, kuin esimerkiksi hälytyksiä, jolloin biteillä 8-15 tarkastellaan sanan aluetta 0.0-0.7 ja biteillä 0-7 aluetta 1.0-1.7.



Kuva 56. Tulotietojen esittäminen paneelissa

Tulojen seurannan lisäksi paneeliin tehtiin PLC diagnostiikka -sivu, jonka System diagnostics view -objektissa esitettiin järjestelmään konfiguroitujen laitteiden tilatietoja. (Siemensin www-sivut 2020)
https://support.industry.siemens.com/cs/document/61910954/system-diagnostics-view-using-wincc-(tia-portal)-and-simatic-comfort-panels?dti=0&lc=en-DZ



Kuva 57. PLC diagnostiikka

6.14.6 Hälytykset

Manipulaattorin hälytyksiä esitettiin kahdella sivulla: Hälytykset ja Häiriöloki. Kummankin sivun toiminnot toteutettiin Alarm view -objektin avulla. Aktiiviset hälytykset saatiin näkymään asettamalla objektiin ”Current alarm states” päälle ja loki tehtiin asettamalla ”Alarm buffer” päälle. Hälytyssivuilla esitettiin kaikki hälytykset heti

niiden aktivoiduttua. Kaikki aktiiviset hälytykset pystyttiin kuittaamaan kerralla painamalla alareunan kuittauspainiketta.

Kuva 58. Hälytykset

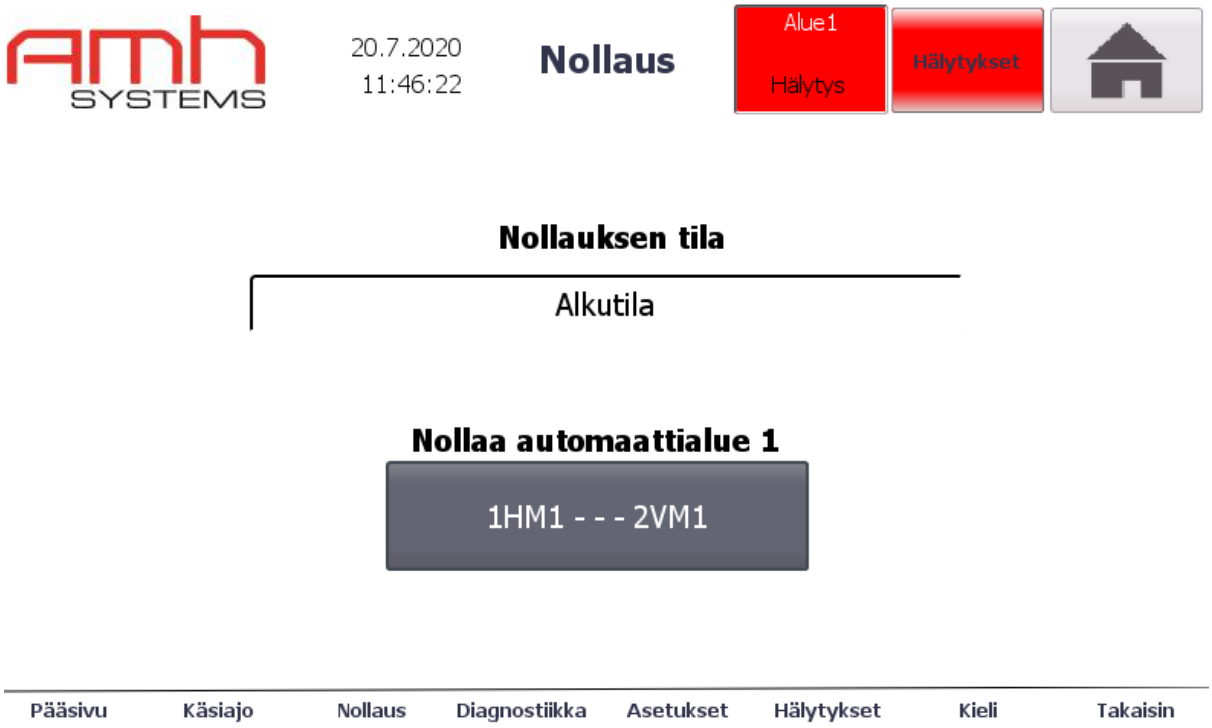
Hälytykset saatiin esitettyä hälytyssivuilla tekemällä HMI Alarms -sivulle viittaukset muuttujiin, joihin logiikkaohjelmassa tallennettiin hälytysten tietoja. Kullekin hälytykselle määriteltiin samalla englannin- ja suomenkielinen teksti, jotka esitettiin hälytyssivuilla hälytyksen sattuessa.

1	Discrete_alarm_1	Common alarm; Hätäseis lauennut	Errors	Alarm common	0	%DB400.DBX1.0
2	Discrete_alarm_2	Common alarm; Hätäseis painettu C1-HS1	Errors	Alarm common	1	%DB400.DBX1.1
3	Discrete_alarm_3	Common alarm; Hätäseis painettu OP10-HS1	Errors	Alarm common	2	%DB400.DBX1.2
4	Discrete_alarm_4	Common alarm; Hätäseis painettu OP11-HS1	Errors	Alarm common	3	%DB400.DBX1.3

Kuva 59. Hälytekstit

6.14.7 Nollaus

Manipulaattorin nollaustoimintoa ohjattiin näyttösivulla Nollaus. Sivulla olevaa painiketta painamalla ohjelmassa alettiin suorittamaan ohjelman Reset_Group_1 -funktiota. Painikkeen yllä olevassa I/O-kentässä esitettiin nollauksen tilaa vastaava teksti, jonka avulla käyttäjä saattoi tietää, mitä nollauksessa tapahtui. Lisäksi nollauspainikkeen väriä muutettiin tilan mukaan.



Kuva 60. Nollaus

7 OHJELMAN TESTAUS

Manipulaattori kasattiin AMH-Systems Oy:n konepajalle 9.6.2020, jolloin ohjelmaa päästiin testaamaan fyysisillä laitteilla. Ohjelma vaati hieman muutoksia sisääntulojen osalta sekä projektin laitekonfiguraatio ja turvaparametrit piti päivittää vastaamaan fyysisiä laitteita, mutta projekti saatiin kuitenkin siirrettyä logiikalle melko vaivattomasti. Manipulaattorin fyysiseen kokoonpanoon oli myös tehty muutamia muutoksia, jotka piti päivittää ohjelmaan. Nostimeen oli lisätty alipaineanturi, jolla tarkistettiin, onko tarttujassa laatikoita sekä sen lähestymisraja oli muutettu hihnarajaksi, jota käytettiin nostomoottorin pysäyttämiseen alas ajettaessa. Hihnarajana oli anturi, joka oli vaikuttuneena aina, kun hihna oli kireällä ja haittalevy oli anturin edessä. Jos nostin ajettiin liian pitkälle, levy pääsi pois anturin tieltä ja signaali katkesi.

Ohjelman toimintoja testattiin aluksi siirtelemällä konepajalta löytynyttä kovalevyä käsiajolla. Käsiajon toiminnallisuudet oli saatu suoraan toimimaan kotona tehdyssä

ohjelmassa ja seuraavaksi alettiin testaamaan tehtyä automaattiohjelmaa sekä turvalaitteiden toimivuutta ja niiden vaikutusta ohjelman suoritukseen.

Ohjelmaa piti muokata laitteeseen tehtyjen muutosten vuoksi sekä askelten ja moottoreiden ehtoja piti muuttaa, sillä esimerkiksi pysäytysrajoja tulkittiin ohjelmassa väärin. Automaattiohjelma saatiin kuitenkin melko nopeasti toimimaan, jonka jälkeen alettiin rakentamaan nollaustoimintoja.

Nollauksessa ohjelman askeleille tehtiin ehtoja, joiden perusteella nollausta ajettaessa ohjelmassa löydettiin kesken jäänyt askel ja se asetettiin aktiiviseksi. Nollaus tuli uutena asiana opinnäytetyön tekijälle, jonka vuoksi toiminnallisuuden rakentaminen oli hyvin työlästä. Toimivan nollauksen saavuttamiseksi valmiiksi tehtyä ohjelmaa sekä tarttujan ohjausta muutettiin suurilta osin ja ohjelmaan luotiin useita uusia muuttujia. Automaattiohjelman ja nollauksen toimintoja kehitettiin ja testattiin jatkuvasti työn edetessä laitteen käyttöönottoon asti. Testejä tehtiin niin kauan, kunnes laitteen toiminnasta voitiin olla täysin varmoja.

Kun lopulta manipulaattorin todettiin toimivan halutulla, tavalla siirryttiin nostamaan laatikoita, joita laitteella oli tarkoitus siirtää. Tässä kohtaa ilmeni ongelmia, sillä tarttujan imukupit eivät pitäneet laatikoiden pinnassa.

Ongelmaa koitettiin korjata erilaisilla imukupeilla, suuremmalla ilmanpaineella sekä lisäämällä tarttujaan puristin, jonka avulla saatiin tartuttua myös laatikoiden kyljistä kiinni. Kun puristimen avulla laatikot saatiin pysymään tarttujassa, huomattiin, että nostomoottori ei jaksa nostaa laatikoita ylös lavalta. Tämän jälkeen nostimeen vaihdettiin tehokkaampi moottori sekä suuremmat imukupit ja jokaiselle imukupille asennettiin oma ejektori. Laitteessa aiemmin ollutta alipaineanturia ei voitu enää käyttää usean ejektorin vuoksi ja se jouduttiin korvaamaan kohdekennoilla, jotka asennettiin vahtimaan jokaista nostettavaa laatikkoa. Lisäksi laitteesta otettiin lopulta puristin pois, sillä suuremmilla imukupeilla saatiin riittävän tiukka ote laatikoista. Kohdekennot tuotiin sarjassa samaan sisääntuloporttiin, johon alipaineanturi aiemmin oli kytketty, joten muutosten vuoksi ohjelmaa ei tarvinnut muokata.

Manipulaattorin koeajot saatiin valmiiksi 2.7.2020, jolloin laite ja sen toiminnot esiteltiin sen tilaajille AMH-Systems Oy:n konepajalla. Laitteeseen haluttiin kuitenkin tehdä muutamia muutoksia, joten niiden parissa jatkettiin heti seuraavana päivänä.

Manipulaattoriin haluttiin huoltoa varten toiminto, jolla se ajettaisiin mahdollisimman kauas nostoasemasta. Tätä varten laitteeseen asennettiin työn tekijän toimesta huollon hidastus- (2VM1-S14) ja pysäytysanturit (2VM1-S15), joiden avulla laite saataisiin pysäytettyä käsiajolla tai paneeliin lisätyllä huoltoajopainikkeella käynnistettävän siirtymisen huoltorajalle.

Huoltoajo toteutettiin niin, että painikkeen aktivoituessa nostin ajettiin ensin yläasentoon ja sinne saavuttuaan se ajettiin huoltorajalle. Nostin pidettiin huoltorajalla niin kauan, kunnes paneelista kuitattiin huolto valmiiksi.

Lisäksi lavan vaihdon yhteydessä haluttiin, että nostin ajettaisiin siirron rajalle odottamaan lavan vaihtoa, jotta lavaa vaihtava työntekijä ei löisi päätään nostomoottoriin tai trukilla ei ajettaisi sitä päin. Lavan vaihtoa varten ohjauspaneeliin oli jo luotu painike, jota painamalla nostin osasi hakea seuraavan lavan pinnan hitaalla nopeudella. Lavanvaihtoon piti siis ainoastaan lisätä painikkeen painamisen tai lavan tyhjentymisen jälkeen ajo siirron rajalle, kunhan pöytä olisi ensin tyhjennetty ja laite kuitattu käyntiin.

Huollolle ja lavanvaihdolle tehtiin ohjauspaneeliin kuittauspainikkeet, joita painamalla manipulaattori palautettiin ohjelmakierron alkuun.

8 KÄYTTÖÖNOTTO

Manipulaattori otettiin käyttöön Eckes-Granini Finland Oy Ab:n Turun tehtaalla 12.8.2020. Käyttöönottoa varten tehtaalle mentiin päivystämään mahdollisten ohjelmointivirheiden varalta sekä perehdyttämään laitteen käyttäjiä. Käyttöönotto onnistui suhteellisen vaivattomasti ja ilman suurempia ongelmia. Tehtaalla jouduttiin käymään kaiken kaikkiaan kolme kertaa käyttöönoton jälkeen.

Ensimmäisenä päivänä tarkkailtiin laitteen toimintaa ja samalla viimeisteltiin sen käyttöohjeita. Ongelmia ei ilmennyt ja ainoastaan laitteen ohjauspaneeliin tehtiin muutos, sillä sen Asetukset-sivu haluttiin suojata salasanalla.

Seuraavana aamuna soitettiin, että tarttujasta oli tippunut laatikoita kesken noston. Vikaa lähdettiin selvittämään paikan päälle. Ohjelmaa tutkiessa huomattiin, että sivuttaisliikkeiden hidastus asetettiin päälle hidastusantureiden nousevasta reunasta, joka tarkoitti sitä, että hidastus ei mennyt päälle, jos laite oli pysähtynyt hidastusrajalle. Ohjelmaa muutettiin niin, että hidastukset olivat aina päällä hidastusrajan ollessa vaikuttuneena, jos moottoria ajettiin tiettyyn suuntaan. Tämän lisäksi sivuttaisliikkeen nopea nopeus asetettiin päälle hidastusrajan mentäessä pois päältä, kun laitetta ajetaan huoltoasemaan. Tehtaalle jäätiin kuitenkin vielä päivystämään, jos muita vikoja sattuisi.

Samana päivänä ei tullut muita vikatilanteita, mutta pyynnöstä ohjelmakiertoa muutettiin niin, että se jäi odottamaan käynnistyskuittausta rullapöydän päälle. Alun perin ohjelmakierto alkoi ja loppui askeleeseen, jossa nostin oli kuormalavan päällä. Paikkaa haluttiin vaihtaa, koska lavanvaihdon kuittaukset olivat työläitä, hitaita ja sekavia. Kotiasema oli alun perin kuormalavan päällä sen takia, ettei laitteen käyttäjät löisi päätään nostimeen heidän ottaessa laatikoita pöydältä, mutta nostin oli kuitenkin niin korkealla, että varsinaista riskiä ei ollut.

Kotiasema saatiin muutettua niin, että nostoasemaan siirtymisen ehdoksi asetettiin käynnistyspainikkeen painaminen ja samalla se otettiin pois nostomoottorin ohjauksesta. Lisäksi käynnistyspainikkeeseen liittyvä muuttuja nollattiin aina, kun nostin ajettiin ylös siirron rajoilla. Ongelmia ilmeni, kun nostin ajettiin laatikoiden kanssa siirron rajalle, jolloin se alkoi heti ajamaan takaisin kotiasemaan. Vika saatiin korjattua lisäämällä sivuttaissiirtymiin ehtoja, joissa tarkasteltiin, onko tarttujan imu päällä vai ei.

Viimeisellä käynnillä 17.8.2020 tarkoituksena oli pitää perehdytys laitteen käytöstä viimeiselle työvuorolle. Tehtaalle päästyä kävi kuitenkin ilmi, että nostimen liike oli pysähtynyt heti, kun ohjelmassa oli huomattu nostettavan laatikon olevan viimeinen.

Pysäytys johtui siitä, että 2VM1 ohjauksessa nostimen alas ajaminen estettiin, jos lavan vaihto oli aktiivinen ja lavaa ei ollut kuitattu vaihdetuksi. Ehto luotiin kotiaseman muutoksen yhteydessä ja sen tarkoituksena oli estää laatikoiden hakeminen, jos lavaa ei ollut kuitattu vaihdetuksi. Vika korjattiin asettamalla ajon estolle uudet ehdot, joissa nostimen oli oltava yläasennossa sekä noutoasemassa.

9 YHTEENVETO

Opinnäytetyön tekeminen edellytti paljon itseopiskelua manipulaattorin laitteiston ja käytettävien ohjelma-alustojen sekä niiden toimintojen osilta. AMH-Systems Oy:n tarjoaman esimerkkiprojektin avulla ohjelmointityöstä saatiin säästettyä huomattavasti aikaa, sillä koko ohjelmaa ei tarvinnut rakentaa alusta asti uudelleen.

TIA Portal -ohjelmointiympäristö ja sillä tehtävä logiikkaohjelmointi olivat opinnäytteen tekijälle tuttuja aiheita koulun kurssien kautta. Työn edetessä vastaan tuli kuitenkin uusia asioita, joihin perehdyttiin joko etsimällä tietoa itsenäisesti tai kysymällä neuvoa työn ohjaajilta.

Manipulaattorin logiikkaohjelma päätettiin kirjoittaa pääosin toimilohkokaavioita käyttäen. Työn alkuvaiheessa vaihtoehtoisena ohjelmointitapana mietittiin sekvenssikaaviota, koska koko ohjelmakierto olisi mahtunut yhteen kaavioon ja SFC olisi tukenut ohjelman haarautuvaa rakennetta. Toimilohkoihin päädyttiin kuitenkin siitä syystä, että niiden käyttäminen tuntui opinnäytetyön tekijälle luontevammalta ohjelmointitavalta sekä sen takia, että AMH-Systemsillä on ollut yleisesti tapana käyttää toimilohkoja ohjelmointiprojekteissa.

Logiikkaohjelma rakennettiin pitkälti malliprojektin mukaisesti noudattaen AMH:n ohjelmointitapoja. Malliprojektin lisäksi ohjelmoinnin tukena käytettiin koulusta saatavilla ollutta opetusmateriaalia sekä siellä aiemmin tehtyjä ohjelmaprojekteja. Koulun projekteja tutkiessa kävi ilmi, että manipulaattorin ohjelmassa useiden toimilohkojen ohjauksissa oli käytetty turhaan tunnistetauluun luotuja tageja apubitteinä. Kyseessä

oli siis muuttujia, joita käytettiin ainoastaan yksittäisten toimilohkojen toiminnoissa. Muuttujat olisi pitänyt sijoittaa toimilohkon sisäiseen muistiin, jolloin tunnistetaulusta olisi saatu huomattavasti lyhempi ja selkeämpi. Jo tehdyt muuttujat jätettiin kuitenkin alkuperäiseen muotoonsa, sillä ne eivät varsinaisesti vaikuttaneet ohjelman toimintaan millään tavalla.

Ohjelmasta tuli työn tekijän odotuksiin verrattuna melko monimutkainen ottaen huomioon laitteen toiminnot. Ohjelmoinnin työläin ja haastavin osuus oli nollaustoimintojen rakentaminen. Nollauksen ehdot piti määritellä tarkkaan jokaiselle askeleelle, sillä jotkin ohjelman askeleet olivat hyvin samankaltaisia keskenään. Projekti oli kuitenkin kokonaisuudessaan sopivan haastava, mutta silti riittävän yksinkertainen ensimmäiseksi ohjelmointiprojektiksi, sillä työstä selvittiin suurilta osin omatoimisesti. Työtä helpotti myös se, että manipulaattori oli itsenäinen laite eikä sen tarvinnut olla yhteydessä tehtaan muihin laitteisiin ja järjestelmiin.

Projektissa korostui, kuinka tärkeitä fyysisellä laitteella tehtävät testit olivat. Ilman logiikkaohjelman kattavia testejä ja sitä kautta tehtyjä muutoksia, ohjelmaa olisi tarvinnut muokata huomattavasti enemmän käyttöönoton yhteydessä.

Työn lopputulokseen voidaan olla tyytyväisiä, sillä asiakkaalle saatiin toimitettua toimivaksi testattu laite asiaankuuluvine dokumentteineen. Työn valmistuminen venyi useista eri syistä alkuperäisestä kahdesta kuukaudesta melkein neljään kuukauteen, mutta laitteen asennuksella ei ollut tiukkaa aikarajaa, joten se ei varsinaisesti haitannut työn osapuolia.

LÄHTEET

Hanssen, D. 2015. Programmable Logic Controllers: A Practical Approach to IEC 61131-3 Using CoDeSys. Chichester: John Wiley & Sons, Ltd. Viitattu 22.5.2020. https://ebookcentral.proquest.com/lib/samk/reader.action?docID=4040112

Karl-Heinz, J. & Tiegelkamp, M. 2010. Programming Industrial Automation Systems: Concepts and Programming Languages, Requirements for Programming Systems, Decision-Making Aids. Heidelberg: Springer Publishing. Viitattu 18.5.2020.

Suvela, T. 2011. AU080401 Automaatiotekniikka, IEC 61131-3 Ohjelmointikielet. Satakunnan Ammattikorkeakoulu. Viitattu 29.7.2020. https://moodle3.samk.fi/pluginfile.php/137948/mod_resource/content/1/AU080402_IEC_61131_3.pdf

Asmala, H. n.da. Ohjelmointikielet, Structured Text, Automaatiotekniikka. Satakunnan Ammattikorkeakoulu. Viitattu 29.7.2020. https://moodle3.samk.fi/pluginfile.php/137974/mod_resource/content/2/Ohjelmointi/Kielet_ST.pdf

Asmala, H. n.db. Ohjelmointikielet, Instruction List, Automaatiotekniikka. Satakunnan Ammattikorkeakoulu. Viitattu 30.7.2020. https://moodle3.samk.fi/pluginfile.php/137980/mod_resource/content/2/Kielet_IL.pdf

Asmala, H. n.dc. Ohjelmointikielet, SFC, Sequential Function Chart, Automaatiotekniikka. Satakunnan Ammattikorkeakoulu. Viitattu 30.7.2020. https://moodle3.samk.fi/pluginfile.php/137977/mod_resource/content/2/Kielet_SFC.pdf

PLC Academyn www-sivut. Viitattu 25.6.2020. https://www.plcacademy.com/

Automaatiotekniikka, Tila- ja sekvenssikaaviot. Satakunnan Ammattikorkeakoulu n.d.. Viitattu 26.6.2020. https://moodle3.samk.fi/pluginfile.php/137920/mod_resource/content/4/Tila-%20ja%20sekvenssikaaviot.pdf

Coding Guidelines. PLCopen 2016. Viitattu 20.6.2020. https://plcopen.org/system/files/downloads/plcopen_coding_guidelines_version_1.0.pdf

Siemensin www-sivut. Viitattu 29.7.2020. https://new.siemens.com/fi/fi.html

Sitek-Palvelu Oy:n www-sivut. Viitattu 29.7.2020. https://www.sitek.fi/
http://www.sitek.fi/tuotteetsiemens/hajautettu-io-et200

Functional safety and IEC 61508 A basic guide. IEC 2004. Viitattu 29.7.2020. https://www.ida.liu.se/~simna73/teaching/SCRTS/IEC61508_Guide.pdf

SEW-EURODRIVE:n www-sivut. Viitattu 29.7.2020. https://www.sew-eurodrive.fi/koti.html

Hajautetut käyttölaitejärjestelmät MOVIMOT MM..D. SEW-EURODRIVE 2014. Viitattu 30.7.2020. https://download.sew-eurodrive.com/download/pdf/21214328.pdf

TIA Information System. Siemens (TIA Portal -ohjelmisto) n.d. Viitattu 16.7.2020.

Structured Control Language (SCL) for S7-300/S7-400 Programming. Siemens. 1998. Viitattu 18.8.2020. https://cache.industry.siemens.com/dl/files/188/1137188/att_27471/v1/SCLV4_e.pdf

AMH-Systems Oy:n www-sivut. Viitattu 20.8.2020. http://amh.fi/

Kauppalehden www-sivut. Viitattu 29.7.2020. https://www.kauppalehti.fi/yritykset/yritys/amhsystems+oy/22322051

SIMATIC Programming with STEP 7. Siemens 2006. Viitattu 21.8.2020. https://cache.industry.siemens.com/dl/files/056/18652056/att_70829/v1/S7prv54_e.pdf