Roope From

Sakeuttimen haran korkeuden automatisointi

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Konetekniikka
Opinnäytetyö
23.2.2024

Tiivistelmä

Tekijä:	Roope From
Otsikko:	Sakeuttimen haran korkeuden automatisointi
Sivumäärä:	60 sivua
Aika:	20.2.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Konetekniikka
Ammatillinen pääaine:	Koneautomaatio
Ohjaajat:	Chief Engineer, Jesper Setälä Lehtori, Maria Sjöholm

---

Opinnäytetyön tilaajana toimi Metso. Opinnäytetyön aiheena on uuden lisäosan suunnittelu Metson sakeuttimiin. Lisäosa tuo uusiin sakeuttimiin mahdollisuuden ohjata sakeuttimen sylintereitä tarkemmin, laskien laitteen mekaanista rasitusta. Ohjelma luotiin Siemensin tarjoamalla ohjelmointityökalulla TIA Portalilla.

Opinnäytetyö rajattiin koskemaan pääosin ohjelmallista osuutta, joten visuaalisuudesta ei ole raportoitu yhtä kattavasti. Opinnäytetyössä käsitellään myös kaivosteollisuuden vaiheet lyhyesti, sakeuttimen toiminta, Metson sakeutinmallit ja ohjelmoinnin perusteet.

Työn pääasiallinen tavoite oli luoda täysin toimiva ohjelma sakeuttimen haran ohjaamiseen toiminnankuvauksen mukaan. Ohjelman tuli toimia simuloinnissa täydellisesti. Opinnäytetyön projekti tullaan luonnollisesti myöhemmin FAT-testaamaan, mutta mahdolliset mekaaniset ongelmat eivät liity opinnäytetyön aiheeseen. Ohjelman toimintaa ei voida opinnäytetyössä täysin raportoida, mutta ohjelman käyttötarkoitus selviää opinnäytetyöstä. Yhteenvetona opinnäytetyö kuvaa suunnitteluprosessia ja lopullista tuotetta, jonka Metso toimiessaan saa sakeuttimiinsa.

Avainsanat: Metso, Sakeutin, PLC, Siemens, TIA Portal, Ohjelmointi

Abstract

Author:	Roope From
Title:	Automation of Thickener Rake Height
Number of pages:	60 Pages
Date:	23.2.2024
Degree:	Bachelor of Engineering
Degree Programme:	Degree Programme in Mechanical Engineering
Professional Major:	Machine Automation
Supervisors:	Chief Engineer, Jesper Setälä
	Senior Lecturer, Maria Sjöholm

The thesis was commissioned by Metso. The topic of the thesis was to design and create a new add-on to a master thickener program. This new addition provides the ability to control cylinders more accurately and individually, reducing mechanical stress on the equipment. The program was designed using the TIA Portal software tool provided by Siemens.

The main objective of the thesis is to design and create a fully functional program according to the given functional description. The program should perform flawlessly in simulation. Real physical possible limitations are not considered in the designing process. The thesis consists mainly of code-related topics, so visualization on the HMI panel is not covered as much as the program. In addition, the thesis covers basics of the mining process, thickener models offered by Metso and general information about the PLC programming.

The function of the program is described, but in such a way that the operating principles are not presented in detail. The program will be FAT-tested in near future and the work continues with the program to make it more similar to the thickener master program.

Keywords: Metso, thickener, PLC, Siemens, TIA Portal, programming

Sisällys

Lyhenteet

PLC:	Programmable Logic Controller, suomeksi ohjelmoitava logiikka, on laite, jolla ohjataan prosessia tai yksittäistä laitetta.
HMI:	Human Machine Interface. Kosketusnäyttö, joka toimii rajapintana käyttäjän ja logiikan välillä.
Tia Portal:	Totally Integrated Automation. Siemensin tarjoama ohjelmointityökalu.
I/O:	Input / Output, Sisään- ja ulostulo tieto.
Runtime:	Ohjelman kiertoaika. Yleisnimitys ohjelmakierrolle.
FAT:	Factory Acceptance Test. Tämä testi varmistaa, että tuote vastaa asiakkaan määrittelyjä ja vaatimuksia ennen kuin se lähtee asiakkaalle.
CCD:	Counter Current Decantation circuits. Vastavirtadekantointi, mineraalien erottelussa.
SFC:	Sequential Function Chart. Graafinen ohjelmointikieli.
LAD:	Ladder, Graafinen ohjelmointikieli.
FBD:	Function Block Chart. Graafinen ohjelmointikieli
ST:	Structured Text. Tekstipohjainen ohjelmointikieli.
IL:	Instruction List. Tekstipohjainen ohjelmointikieli.
CPU:	Central Processing Unit. Keskususyksikkö.

ALU:	Arithmetic Logic Unit. Keskusyksikön osa, joka suorittaa loogisia toimintoja.
CU:	Control Unit. Keskusyksikön osa, joka hallitsee ohjaustoimintoja.

1 Johdanto

1.1 Työn aihe ja rajaus

Opinnäytetyön aiheena on luoda PLC-ohjelma sakeuttimen haran automatisoinnille vääntömomentin mukaisesti. Ohjelma luodaan Siemens TIA Portal -ohjelmointiympäristössä toiminnallisen kuvauksen kuvaamalla tavalla. Sakeuttimelle (Kuva 1) on olemassa valmis pääohjelma, mutta valmiissa ohjelmassa ei ole mahdollisuutta ohjata haran korkeutta ohjaavien sylintereiden asemaa tarkasti. Harat liikkuvat pystysuunnassa usean sylinterin varassa, joiden iskunpituus on säädeltävissä käytössä olevan sovelluksen mukaan. Työssä tutustutaan ohjelman toiminallisuuteen, logiikkaohjelmoinnin perusteisiin ja suunnittelutyön etenemiseen.

Kuva 1. High-Rate Thickener (Metso sakeutinesite 2022).

Valmis lisäosa tulee olemaan Simatic ET 200SP etä-I/O:na, mutta simulointiin tarvitaan runtime, jonka vuoksi ohjelmoinnissa tulee näkymään 1200-tuoteperheen CPU. Laitteiston konfiguraatiota ei ole tehty todellisen laitteen mukaisesti, eli ohjelmaan ei ole luotu oikeita ohjauskortteja. Esimerkiksi sisääntuloina näkyvät muuttujat ovat simuloitavia muuttujia, jotka vastaavat oikeita kenttälaitteita. Työssä käsitellään kuinka sylintereiden asemaa säädellään vääntömomentin mukaan ja kuinka se on toteutettu ohjelmassa. Yrityksen edun vuoksi ohjelmasta voidaan näyttää vain pieniä osia. Siemensin ohjelmointikirjaston lohkoja voidaan tosin näyttää vapaasti, mutta luomiani aliohjelmia ei kokonaisuudessaan voida näyttää. Valmiin Metson pääohjelman lohkoista ei voida näyttää mitään.

Olen työskennellyt kyseisen ohjelman parissa kesäkuusta asti, joten esihenkilön kanssa sovimme, että ohjelman raportoiminen opinnäytetyöksi on mahdollista. Samalla sovittiin, että opinnäytetyö rajoitetaan uuden ohjelman luontiin ja simulointiin Siemensin ohjelmointiympäristössä. Opinnäytetyöhön ei kuulu rakentamani ohjelman yhdistäminen sakeuttimen pääohjelmaan, eikä käyttöönoton raportointi. Mahdolliset muutokset ohjelmaan opinnäytetyön aloittamisen jälkeen, eivät kuulu opinnäytetyöhön.

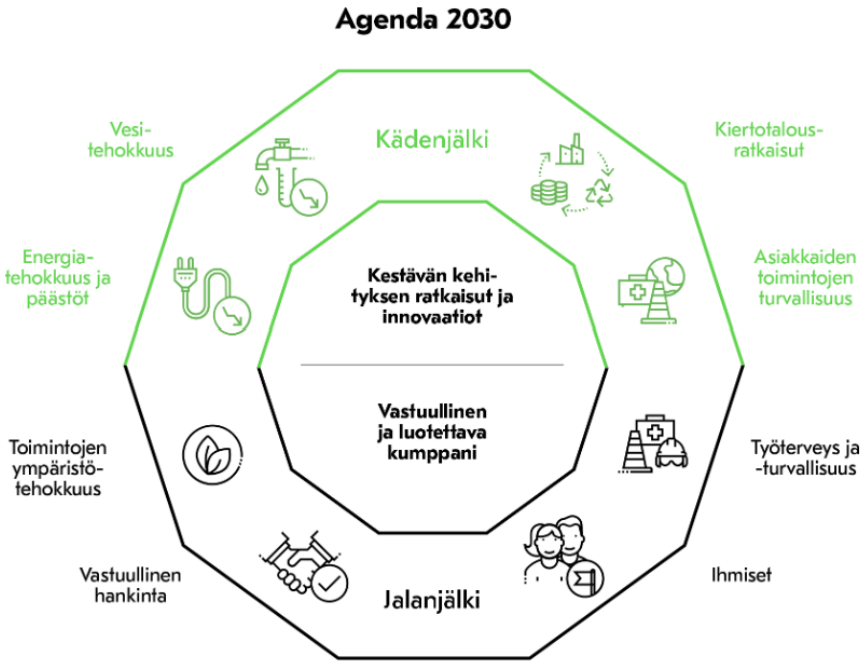
1.2 Yrityksestä

Opinnäytetyö tehdään Metso Oyj:lle. Metso on suomalainen pörssiyhtiö, joka toimii koneiden ja laitteiden valmistajana kiviaines-, kaivos- ja metallinjalostusteollisuuden aloilla. Aiemmin Metso tunnettiin nimellä Metso Outotec, kun Metso ja Outotec fuusioituivat vuonna 2020. Keväällä 2023 yrityksen nimi vaihdettiin jälleen Metsoksi.

Metsolla on toimintaa 45 maassa ja yli 16 000 työntekijää. Metson liikevaihto on noin 5,3 miljardia euroa, josta liikevoittoa oli 0,5 miljardia vuonna 2022. Yrityksen kokoa voidaan tarkastella esimerkiksi liikevaihdon, henkilöstön, liikevoiton ja markkina-arvon mukaan. Markkina-arvon mukaan vuoden 2023 lokakuussa

Metso on 9,54 miljardilla eurolla Suomessa sijalla 8 ja maailmanlaajuisesti sijalla 1471. [companiesmarketcap 2023.]

Metso on sitoutunut vastuulliseen toimintaan, joka toimii Metson strategisena prioriteettina. Metso pyrkii jatkuvasti parantamaan laitteiden tehokkuutta ja hyötysuhteita prosessien parantamiseksi, mikä osaltaan on osa ympäristöystävällisempää tulevaisuutta. Metso jakaa vastuullisuusstrategiansa kahteen osaan, jotka ovat kädenjälki ja jalanjälki. Kädenjälki kuvaa myönteistä vaikutusta, joka Metsolla on ihmisiin ja ympäristöön, asiakkaiden käyttäessä Metson teknologiaa. Jalanjälkenä ovat vaikutukset, jotka syntyvät laitteiden valmistuksessa ja kuljetuksessa asiakkaille. Metso pyrkii nettonollapäästöihin vuoteen 2030 mennessä, ja yritys on sitoutunut omilla teoillaan rajoittamaan ilmaston lämpenemisen 1,5 asteeseen. [Metso -a.]



Kuva 2. Agenda 2023 (Metso -a).

2 Projektin tavoitteet

Opinnäytetyön tavoite on saada ohjelma toimimaan täysin simuloinnissa. Käyttöönotossa tai FAT-testissä mahdollisia ongelmia ei tarvitse ottaa huomioon virtuaalisessa simuloinnissa. Esimerkiksi hydrauliikkakoneikon pumpun kokoa ei tarvitse miettiä, koska simuloinnissa on käytössä rajaton määrä tehoa sylintereille. Projektissa luodaan myös luonnostelma käyttöliittymästä, eli visualisointi HMI-paneelille.

2.1 Ohjelman käyttötarkoitus

Ohjelma toimii uutena lisäosana, esimerkiksi Metson tahnasakeuttimiin (paste Thickener), joka voidaan ottaa käyttöön HMI-paneelilta. Ohjelma kehitetään pääosin sen vuoksi, että ohjelma on sakeuttimen pääohjelmaan verraten tarkempi ohjaamaan sylintereitä ja sitä kautta haran korkeutta. Sisääntulodatana saadaan sylintereiden asema, jolloin sylintereiden asemaa voidaan korjata jatkuvasti.

2.2 Lisäarvo Metsolle

Ohjelman toimiessa täydellisesti Metso voi myydä asiakkailleen sakeuttimiin uutta lisäosaa, joka tehostaa erotteluprosessia ja pidentää laitteen käyttöikää. Metso voi myydä laitetta suuremmalla hinnalla tämän uuden lisäosan avulla. Lisäkustannuksia tulee pääosin vain suunnitteluvaiheen henkilöstökuluista. Laitteen mekaaninen rasitus vähenee, kun haraa voidaan nostaa ylöspäin automaattisesti, jolloin huoltoväliä voidaan mahdollisesti kasvattaa. Samoin hara pysyy stabiilina pitäessään sylintereitä samalla tasolla.

3 Sakeutin

3.1 Sakeutin yleisesti

Jotta saadaan kattava kokonaiskuva ohjelman toiminnasta, on hyvä ymmärtää, minkälaiseen laitteeseen ohjelma tullaan lataamaan. Sakeutin on osa kaivosteollisuuden erotteluprosessia. Sakeuttimet perustuvat laskeutukseen, jossa kiinteä aines valuu pohjalle ja neste jää pinnalle. Talteen otettava aines on tiheää lietettä. Sakeuttimien kokoluokka on muutamasta metristä satoihin metreihin. Sakeuttimia on Metson tarjonnassa 5 erilaista laiteperhettä.

3.1.1 Paste Thickener

Tahnasakeuttimia (Paste Thickener) käytetään, jos sovelluksissa vaaditaan runsaasti vedenpoistoa. Tahnasakeuttimet ovat usein korkeampia ja halkaisijaltaan kapeampia kuin muut sakeutintyypit. Nämä sakeuttimet saavuttavat maksimaalisen alivuototiheyden. Alivuotoaines on nimensä mukaisesti hyvin tahnamaista. Näiden sakeuttimen etuna on jopa 40 % vedensäästö, Metson yleisimpään sakeutinmalliin verrattuna, joka on korkean nopeuden sakeuttimet. Tahnasakeuttimisaa voidaan käyttää opinnäytetyön lisäosaa, joka saa sisääntulodatana sylintereiden aseman ja paineen, jonka avulla asemaa korjataan. Tahnasakeuttimet sopivat hyvin sovelluksiin, joissa käsitellään erotteluprosessin jäännöksiä, halutaan täyttää louhinnan jälkeiset ontelot lietteellä tai varastoida vaaralliset ylijäämäaineet maan alle. Tahnasakeuttimia käytetään erityisesti prosesseissa, jossa on käytössä CCD-piiri eli vastavirtadekantointi. CCD on sarja useita tahnasakeuttimia, jossa pyritään erottelemaan lähes täysin kaikki mineraalit lietteestä usean sakeuttimen voimin. Metson tahnasakeuttimia on asennettu maailmalle jo yli 100 kappaletta. [Metso -b.]

Kuva 3. Metso 2nd Paste Thickener (Mining Techonology).

3.1.2 Inclined Plate Settler

IPS-sarjan sakeuttimet hyödyntävät lamellisuutta mineraalien erotteluun. Toisin sanoen laite hyödyntää ohuita levyjä erottamaan veden lietteestä. IPS-sarjan laitteet ovat osoittaneet, että ne tarjoavat tehokkaan veden talteenottonopeuden ja energiahyötysuhde on erinomainen. Näin ollen laitoksen hiilijalanjälkeä saadaan karsittua. Etuna IPS-sarjan laitteissa on kemikaalien säästö. Jopa 10 % kemikaaleista voidaan säästää käyttämällä IPS-sarjan laitteistoa. Sakeuttimien osalta verrattuna konventionaaliseen sakeuttimeen, laitoksen hiilijalanjälki laskee 85 %. Käyttöikä IPS-sarjan tuotteilla on suuri, käyttökulut pienemmät ja huoltaminen on helpompaa. Kokonsa puolesta IPS-sarjan laitteet voidaan asentaa kompaktiin tilaan, jopa päällekkäin. Asennusaika on myös pienempi kuin muilla sakeutinmalleilla. [Metso -c.]

3.1.3 High Rate Thickener

Korkean nopeuden sakeuttimet (High Rate Thickeners) ovat Metson toimitetuin sakeutinmalli. Tätä sakeutinmallia on kehitetty 35 vuotta ja sitä on toimitettu 85 maahan, yli 2000 kappaletta. Tähän sakeutinmalliin voidaan lisäosana lisätä Reactorwell, joka parantaa sakeuttimen elinikää, suurentaa alivuototiheyttä ja selkiyttää ylivuotoa. Reactorwell antaa suurimman lisäarvon siinä, että flokkulanttien määrää voidaan laskea. Tässä sakeutinmallissa on myös ”Directional autodil™ feed dilution system”, joka vastaa automaattisesta laimennuksesta. Systeemi varmistaa, että syöttökaivossa vallitseva tiheys on optimaalinen flokkulaatiolle, vaikka syöttövirtaus tai syöttötiheys muuttuisi. Tämä mahdollistetaan siten, että syöttökaivoon syötetään supernatanttia vettä, eli laskeutuneen lietteen yläpuolelle jäävää vettä, missä lietteen laimennus ja sekoitus suoritetaan. Nämä sakeutinmallit ovat hyvin automatisoituja, kuten yllä olevista esimerkeistä huomataan, mahdollistaen tehokkaan erotteluprosessin instrumenttien antamien arvojen avulla. Kyseisiä sakeuttimia voidaan käyttää lähes kaikkiin prosesseihin, joihin sakeuttimet ovat tarkoitettu. Esimerkiksi CCD-piiriin tai jätteenkäsittelyyn. [Metso -d.]

Kuva 4. Metso High-Rate Thickener ja Reactorwell (Metso -d).

3.1.4 High Compression Thickener

Korkean paineen sakeuttimet ovat hyvin samankaltaisia korkean nopeuden sakeuttimien kanssa. Korkean paineen sakeuttimet ovat hieman halkaisijaltaan pienempiä, mutta korkeampia, aiheuttaen suurempaa painetta säiliön pohjalle. Alivuototiheys on hyvin suuri tämän mallin sakeuttimissa. Metso on toimittanut yli 200 korkean paineen sakeutinta. Korkean paineen sakeuttimiin on myös tarjolla Reactorwell-, Turbodil- ja Directional autodil lisäosat. Kyseistä sakeutintyyppiä käytetään erottelemaan mineraaleja, esimerkiksi savimaisesta materiaalista. [Metso -e.]

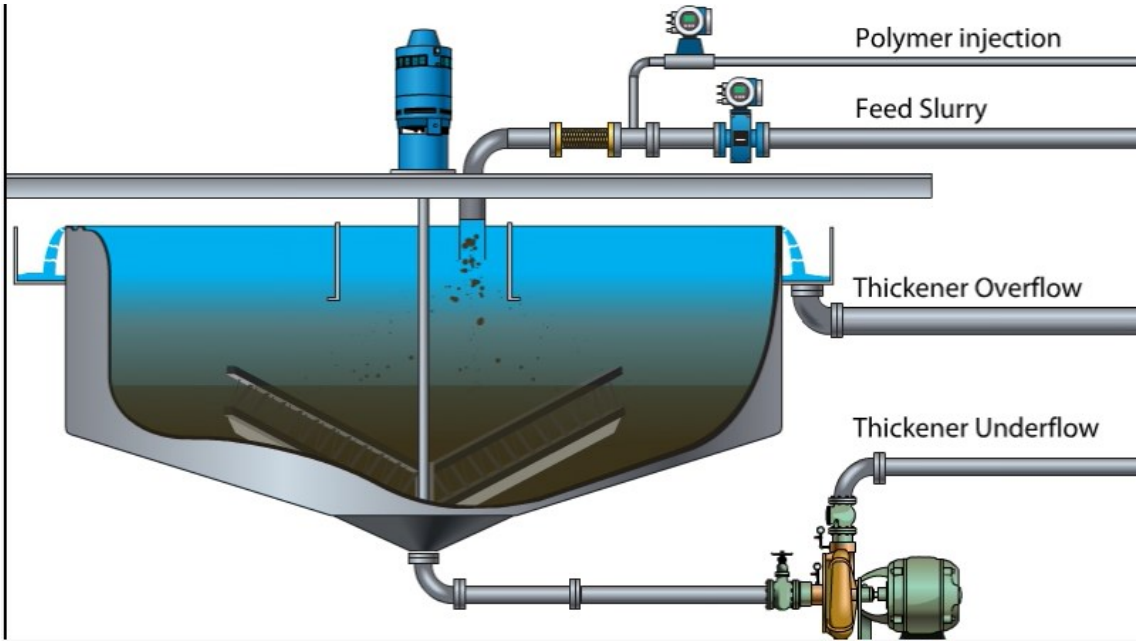
3.2 Toimintaperiaate

Sakeuttimia tai selkeyttimiä käytetään joko veden erottelemiseen tai kiinteän aineen talteenottoon. Sakeuttimen tarkoitus on lisätä nesteen kiintoainepitoisuutta, jotta saadaan suurempi osuus kiintoainesta lietteestä talteen. Selkeyttimen tarkoitus on poistaa nesteestä kiintoaineita ja tehdä nesteestä mahdollisimman puhdasta. Selkeyttimissä usein käytetään erilaisia suodattimia erottelemaan pienimmätkin partikkelit nesteestä. [Savona Equipment 2018.]

Sakeuttimen toimintaperiaate perustuu laskeutukseen painovoiman avulla, jossa kiinteä aine erotetaan nestelietemäisestä aineesta. Yleisin sakeutintyyppi on sylinterimäinen suuri laite, jossa on allasmainen tankki. Sakeuttimet ovat usein valmistettu teräksestä tai raudasta. Sakeuttimen yli kulkee huoltosilta, jossa on ohjauskeskus, sekä ajoyksikkö. Sakeuttimen altaassa pyörii hitaasti hara, jonka tehtävänä on aluksi tiivistää kiintoainetta ja lopulta ohjata kiintoainetta pois laitteesta. [Savona Equipment 2018.]

Sakeuttimen toiminnan tärkeimmät elementit ovat syöttökaivo, haramekanismi ja alivuotopumppu. Syöttökaivon pääasiallinen idea on eliminoida kineettinen energia, joka vaikuttaa sisääntuloyhteen lietteessä. Syöttökaivossa voidaan sisääntulolietteeseen sekoittaa flokkulantteja, jotka edesauttavat partikkeleiden liittymistä suuremmiksi kokonaisuuksiksi, joten ne laskeutavat nopeammin

sakeuttimen pohjalle, sekä suurempi määrä kiinteää materiaalia saadaan eroteltua. Prosessi tunnetaan nimellä flokkulaatio. Sakeuttimen hara on eräänlainen harava, joka pyörii hitaasti sakeuttimen pohjalla. Hara ohjaa kiinteää ainesta alivuotopumpulle ja kiinteyttää ainesta. Alivuotopumppu on vastuussa, että kiinteä aines saadaan talteen ja sakeuttimen pohjalla pysyy oikea määrä lietettä, jota kutsutaan mudaksi. Tavoite on maksimoida kiinteän aineen tiheys alivuotopumpulle, jotta ylivuotovesi olisi mahdollisimman puhdasta ja sitä voidaan uudelleen käyttää. Suuri mudan määrä kasvattaa kiintoaineen talteenottoa, mutta laitteen vääntömomentti ja pumpun kyky pumpata korkeatiheyksistä ainesta ovat rajoittava tekijöitä. [Knight 2017.]



Kuva 5. Kuvakaappaus pääyhteistä. (Knight 2017).

3.3 Sakeuttimen instrumentointi

Sakeuttimen instrumentoinnilla seurataan erilaisia parametrejä, joilla mitataan pääosin sakeuttimen suorituskykyä ja tarkkaillaan mahdollisia vikatiloja. Sakeuttimessa on useita mittalaitteita, joilla voidaan säätää prosessia tehokkaammaksi. Instrumentit lähettävät tietoa logiikalle, joka toteuttaa sisääntulodatan perusteella ennalta ohjelmoidun ohjelman mukaan sakeuttimen ohjaamista

ulostulosignaalien avulla. Opinnäytetyössä käytetyistä IO:sta kerrotaan lisää ohjelman toteutusluvun alla. [Diemme Filtration 2023.]

3.4 Kaivosprosessin vaiheet lyhyesti

Sakeutin on tyypillisesti prosessin häntäpäässä. Sakeuttimia käytetään pääosin veden poistossa. Kaivoskokonaisuudessa on 4 pääasiallista prosessivaihetta, jotka ovat murskaus ja jauhatus, materiaalin analysointi, erottelu ja veden poisto. Eri prosesseille on erilaisia sovelluksia, jotta mahdollisestaan tehokas mineraalien talteenotto.

3.4.1 Murskaus ja jauhatus

Aluksi sisään tuleva kiviaines murskataan ja jauhetaan pienikokoseiksi jauheeksi. Murskaus tapahtuu tavallisesti kuivissa olosuhteissa, mutta jauhatusmyllyjä käytetään kuivina tai märkinä. Suurin osa malmeista koostuu kovista ja sitkeistä kivimassoista, joita tulee murskata, jotta arvokkaat mineraalit voidaan ottaa talteen. Murskaus on usein toteutettu osissa. Ensimmäisessä vaiheessa kiviaines murskataan halkaisijaltaan noin 150 mm kokoisiksi kappaleiksi leukamurskaimella ja toisessa vaiheessa kiviaines murskataan halkaisijaltaan noin 10–15 mm kokoiseksi, jotta kiviaines voidaan syöttää jauhatusmyllyyn. Jauhatusmylly on sylinterin muotoinen laite, joka pyörii akselinsa ympäri. Myllyn sisällä on metallisia palloja, sauvoja tai murskaimia, jotka jauhavat kiviainesta pienemmäksi. Jauhetta käsitellään usein mikrometrien kokoluokassa. [Lorig & Gruner 2006.]

3.4.2 Analysointi

Seuraavaksi mineraaleja analysoidaan, jotta prosessi voidaan optimoida aineksen mukaan tehokkaaksi. Useimmissa kaivoksissa on jo kehittyneet automaatiojärjestelmät, jotka muokkaavat laitteiden parametreja automaattisesti mineraalien ominaisuuksien mukaan, jotta kiviaineksesta saadaan eroteltua mahdollisimman paljon arvokkaita mineraaleja. Kiviaineksesta otetaan näytteitä, jotka

edustavat yleistä keskiarvoa kokoluokassaan. Näytteitä analysoidaan kemiallisesti, röntgenillä ja mineralogisesti raskasainetestauksella. Myös koko on määräävä tekijä prosessin optimoimisessa. [Lorig & Gruner 2006.]

3.4.3 Erottelu

Jauhetusta kiviaineksesta erotellaan arvokkaita mineraaleja erilaisilla tavoilla. Optinen lajittelu, painovoimaerotus, magneettinen- tai sähköinen erottelu ja flotaatioerottelu ovat yleisimpiä tapoja erotella kiviaineksesta arvokkaat mineraalit. Optisessa erottelussa erotellaan nimensä mukaisesti mineraaleja niiden värin perusteella. Apuna voidaan käyttää ultravioletti- ja infrapunavaloa. Paras kontrasti on usein valkoisen ja mustan välillä.

Painovoimaerottelussa apuna käytetään kiviaineksen tiheyseroja. Yksi painovoimaerottelun malli on erotella materiaaleja virtaavasta vedestä. Tänä päivänä suosiota ovat lisänneet keskipakoisvoimaan perustuvat spiraalinmuotoiset laitteet. Näissä laitteissa massa virtaa siis ylhäältä alaspäin, jolloin raskaammat hiukkaset keskittyvät virran sisäpuolelle, josta ne voidaan erotella erotteluaukkojen avulla. Tällainen sovellus toimii erinomaisesti raskaiden mineraalihiekkojen käsittelyssä.

Magneettisessa erottelussa käytetään hyväksi magneettikenttiä, joiden avulla mineraalit saadaan eroteltua kiviaineksesta niiden vetovoiman avulla. Erityisesti materiaalit kuten frankliniitti ja pyrrotiittia saadaan eroteltua tehokkaasti matalan itensiteetin erottimilla. Korkean intensiteetin erottimilla saadaan puolestaan eroteltua rautaa sisältäviä titaani- ja voflramimalmeja. Sähköisessä erottelussa käytetään hyväksi mineraalien sähkövarauksia. Sähköstaattista erotusta käytetään laitoksissa, joissa erotellaan raskasta zirkonia tai monatsiittia sisältävää ainesta.

Flotaatioerottelu on suosituin erottelumuoto nykyaikaisissa kaivosprosesseissa. Flotaatioerottelussa hyödynnetään fysikaaliskemiallisia pintaominaisuuksia, erityisesti koostumusta. Hiukkasia voidaan muokata joko vettä hylkiviksi (hydrofobisiksi) tai vettä houkuttelevaksi (hydrofiilisiksi). Vaahdotuskennossa on vettä ja

vaahtoa. Partikkelit voidaan joko sitoa vaahdotuskennon läpi kulkeviin ilmakupliin tai jäämään vesimassaan. Tällä erotusmenetelmällä mahdollistetaan mineraalien kuten kuparin, lyijyn ja sinkin erottelu, mikä oli ennen mahdollista vain metallurgisilla prosesseilla. [Lorig & Gruner 2006.]

3.4.4 Veden poisto

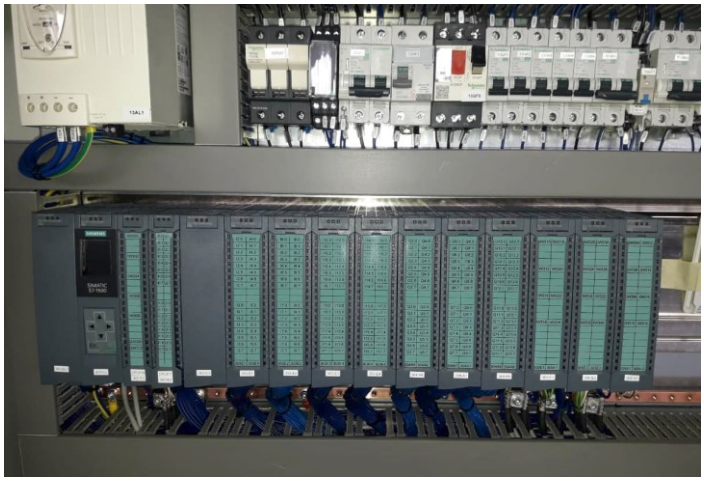
Erottelusta syntyneestä rikasteesta tulee poistaa vesi, jotta massat saadaan kuljetettua pois kaivoksilta. Vettä voidaan myös kierrättää uuteen prosessikiertoon. Edellä mainituista prosessivaiheista tuotetutuista rikasteista poistetaan vettä, esimerkiksi sakeuttimien avulla. Vettä voidaan poistaa myös suodattamalla tai kuivaamalla rikasteita. Suodatuksessa rikasteet työnnetään suodatinmateriaalin läpi, jolloin rikasteen nestepitoisuudeksi jää noin 8 – 15 %. Kiinteistä rikasteista poistetaan vettä nykyaikana usein kuivaamalla. Kuivaus tapahtuu usein konvektiokuivaamalla, jossa käytetään kuumia palamiskaasuja kosteuden poistamiseen massavirrasta. [Lorig & Gruner 2006.]

4 Ohjausjärjestelmistä yleisesti

Ohjausjärjestelmä on yleisnimitys laitteille tai prosessilaitoksille, joita ohjataan automaattisesti. Ohjausjärjestelmän pääasiallinen tehtävä on ohjata laitteita sisään tulevan signaalitiedon avulla. Tavallisimmin ohjausjärjestelmillä ei ole omaa tekoälyä, vaan ihmisen tekemä koodi ohjaa laitekokonaisuutta. Automaattisesti toimivasta järjestelmästä käytetään nimitystä SCADA (Supervisory Control And Data Systems). Erona automaatiojärjestelmän ja ohjausjärjestelmän välillä on se, että automaatiojärjestelmässä kyse on ohjauskokonaisuudesta, kun taas ohjausjärjestelmä on yhden laitteen tai prosessin ohjaus. Sen lisäksi, että ohjausjärjestelmät ohjaavat laitteita, ne keräävät ja tallentavat tietoa laitteilta. Näin voidaan ennakoida laitteen elinikää ja mahdollisia huoltokohteita. Ohjausjärjestelmät jakautuvat avoimeen- ja suljettuun järjestelmään. Avoimessa järjestelmässä ei ole takaisinkytkentää, eli ohjausjärjestelmä ei tiedä esimerkiksi moottorin pyörimisnopeutta. Suljetussa järjestelmässä saadaan moottorin toiminnasta dataa. [Keinänen ym. 2008: 209 - 210.]

4.1.1 PLC-ohjelmointi yleisesti

PLC (programmable logic controller), eli ohjelmoitavat logiikat ovat suuressa roolissa nykyaikaisissa automaatiojärjestelmissä. Ohjelmoitava logiikka on pienikokoinen tietokone, joka on varustettu mikroprosessorilla. Ohjelmoitavat logiikat syrjäyttivät releohjauksen, sillä yhdellä logiikalla voidaan korvata, jopa tuhansia releitä. Samoin kaapelointi on hyvin paljon yksinkertaisempaa kenttäväylillä, ohjelmoitavaa logiikkaa käyttämällä. Ensimmäisenä logiikat otettiin käyttöön autoteollisuudessa.



Kuva 6 Siemens 1500-sarjan logiikka ja ohjauskortteja (Automazioni industriali 2023).

Ohjelmoitava logiikka suorittaa ohjausta ennalta luodun koodin mukaisesti sisääntulodatan mukaan. Ohjelmoitavassa logiikassa on tulo- ja lähtöportteja ja näihin portteihin yhdistetään kenttälaitteet. Käsiteltävä data voi olla digitaalista tai analogista. Logiikka pyörittää ohjelmakiertoa jatkuvasti ja tyypillisesti ohjelmakierron kesto on 0–20 ms. Ohjelmantestauksessa voidaan käyttää myös soft-PLC sovellusta, joka on tietokoneessa pyörivä ohjelmakierto, josta voidaan monitoroida koko ohjelman toimintaa. Tämä on hyödyllistä etenkin vianetsinnässä. [Kippo & Tikka 2007: 54 - 58.]

4.1.2 PLC-ohjelmoinnin perusteet

Logiikkaohjelmointi noudattaa standardia IEC 61131-3 ja on eräänlainen ohjelmoinnin suuntaus. IEC 61131-3 -standardi sisältää rungon pääasiassa vakio-ohjelmointikieliin ja tietotyyppeihin. Alkuperäinen IEC 61131 julkistettiin vuonna 1993. Nykyinen versio on kolmas ja tämä versio julkistettiin vuonna 2013. Kaikki laitevalmistajat ovat sitoutuneet noudattamaan kyseistä standardia omassa ohjelmointiympäristössään, mutta pieniä eroja standardiin saattaa löytyä. Esimerkiksi, Siemens kutsuu ohjelmointikieliä hieman eri nimillä. Kuten todettua, harva laitevalmistaja noudattaa standardia kuitenkaan täydellisesti, mutta runko on hyvin samanlainen. Joten, jos ohjelmoija osaa yhden laitevalmistajan syntaksin, on uusi ohjelmointiympäristö nopea oppia. Ohjelmointikielet on pyritty suunnittelemaan niin, että ohjelmoinnista saa selvää ihminen, jolla ei ole aikaisempaa ohjelmointikokemusta. Ainoastaan ST (Structured Text) on haastavampi kieli, sillä se muistuttaa ulkoasultaan muun muassa Python-ohjelmointikieltä. Käytetty ohjelmointikieli kannattaa valita ohjelmoitavan sovelluksen mukaan, joten kaikkien kielien opetteleminen on olennaista. [Kippo & Tikka 2007: 54 - 58.]

PLCopen on vastuussa standardissa määriteltyjen kielien ja tietotyyppien totutuksesta. PLCopen on eurooppalainen standardiorganisaatio, joka pyrkii edesauttamaan yhtenäisyyttä eri laitevalmistajien yhteensopivuutta. Tämä standardiorganisaatio edesauttaa laitteiden käyttämistä ristiin. [PLCopen.]

Ohjelmointi koostuu ehto- ja toteamuslausekkeista, sekä ali- ja pääohjelmista. Ehtolause vaatii nimensä mukaan jonkin ehdon. Esimerkiksi, jos A ja B on tosi, suorita aliohjelma. Toteamuslause on yksinkertaisesti vain A on tosi. Aliohjelmia käytetään ohjelman selkeyttämiseksi ja samaa aliohjelmaa voidaan kopioida ohjelmaan eri muuttujilla. Pääohjelmaan lisätään aliohjelmia ja usein pääohjelmalle tuodaan arvot, joita halutaan monitoroida.

4.2 Keskusyksikkö CPU

Central processing unit, suomeksi prosessori tai keskusyksikkö, on vastuussa ohjelman suorittamisesta ja toimii ohjelman aivoina. Keskusyksikkö pyörittää ohjelmakiertoa toimintaohjeidensa mukaan, jotka on ladattu keskusyksikölle etukäteen. Keskusyksikön rakenteeseen kuuluu ohjelman muisti, rekisterit, ohjelman laskentayksikkö, ohjelman siirtoyksikkö ja väylät. Virtalähde on usein ulkoisesti sähkökaapissa. [[Kippo & Tikka 2007: 54 - 58.]

Ohjelman muisti (Instruction memory) säilyttää ohjelmakoodia, jonka mukaan keskusyksikkö toimii. Keskusyksikkö hakee tästä muistista ohjeita suorittaakseen tehtävät oikein. Rekisterit (Registers) toimivat pikamuistina, toisin sanoen välimuistina tallentamaan väliaikaisia tietoja ohjelmakierron aikana. Ohjelman laskentayksikkö (ALU- Arithmetic Logic Unit) suorittaa loogisia toimintoja, sekä matemaattisia laskutoimituksia. Ohjelman siirtoyksikkö (CU- Control Unit) hallitsee logiikan ohjausohjeita. Siirtoyksikkö hakee tietoa ohjelman muistista, tulkitsee sitä ja ohjaa laskentayksikköä ja muita mahdollisia komponentteja. Väylät ovat yleisnimitys tiedonsiirtokaapeleille ja nykyaikaiset logiikat tukevat usein kaikkia väyläprotokollia. [Computer Science Wiki 2023.]

4.3 IEC 61131-3 Ohjelmointikielet

Siemens tukee kaikkia IEC 61131-3 ohjelmointikieliä, mutta Siemens on nimennyt kielet eri tavalla. Ohjelmointikieliä on 3 graafista ja 2 tekstipohjaista. IEC 61131-3 sanelee myös käytettävät tietotyypit.

4.3.1 Tietotyypit

Jokaiselle muuttujalle, jota ohjelmassa käytetään, on annettava tietotyyppi (Data Type). Tietotyypit ovat oleellinen osa logiikkaohjelmointia ja niiden oikea käyttö on tärkeää ohjelman suorituskyvyn ja virheiden vähentämisen kannalta. Nykyaikaiset ohjelmointiympäristöt eivät käännä ohjelmaa, mikäli väärää tietotyyppiä käytetään epähuomiossa. Esimerkiksi, jos BOOL-tietotyyppiä, johon

tallennetaan digitaalista tietoa, yrittää käyttää analogisen tiedon käsittelyyn, ohjelma antaa virheilmoituksen. Sen sijaan analogisen tiedon käsittelyyn tulee varata muistia tarvittava määrä, jotta ei varata turhaan ylimääräistä muistia. Jos sovelluksessa on laskuri, jonka maksimiarvo on 10, voidaan käyttää SINT eli Short Integer -tietotyyppiä, joka vie 8 bittiä muistia. Tietotyypin alue on -128–127, joten se riittää tähän sovellukseen. Sen sijaan käyttämällä samaiseen laskuriin tietotyyppiä DINT eli Double Integer, joka on kooltaan 32 bittiä ja alue on -2147483648–2147483647, varataan turhaan muistista 24 bittiä, jota ei ohjelmassa käytetä.

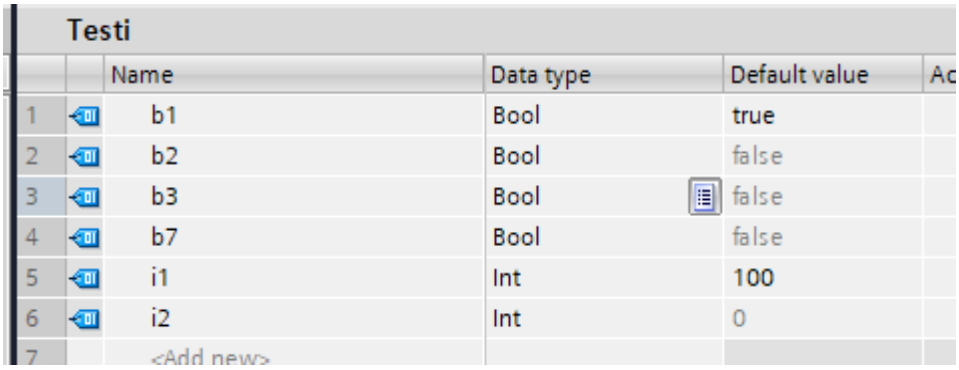
UDT (Used-Defined Data Type) on ohjelmoijan itse luoma tietotyyppi. Hämäävästä nimestä huolimatta UDT on tietotyyppi, joka sisältää usein yhdistelmän standardin mukaisia tietotyyppejä, joko STRUCT- tai ARRAY -muodossa.

UDT:n käyttäminen on tehokasta ohjelmointia, koska samaa tietotyyppiä voidaan käyttää useasti ohjelmassa, ettei yksittäisiä muuttujia tarvitse luoda jokaiseen aliohjelmaan tai TAG-taulukkoon. Näin ollen ohjelma on kompaktimpi ja helpompilukuinen. Samoin muutoksien tekeminen on helpompaa. Tarvitsee vain muokata UDT, niin ohjelmointiympäristössä kaikki tietotyypit päivittyvät päivitettyyn tietotyyppiin. Logiikkaohjelmoijien keskuudessa ei ole yhteisymmärrystä, mikä on paras tapa käyttää UDT:tä. Suurin osa logiikoita vaatii latauksen, jos UDT:tä muutetaan. Tämä aiheuttaa ongelmia etenkin silloin, jos tietotyyppiä muutetaan laitteen ollessa käynnissä. Näin ollen nyrkkisääntönä voidaan pitää, että UDT kannattaa luoda muuttujille, jotka tuskin tulevat muuttumaan käyttöönoton jälkeen. [Dietrich 2023.]

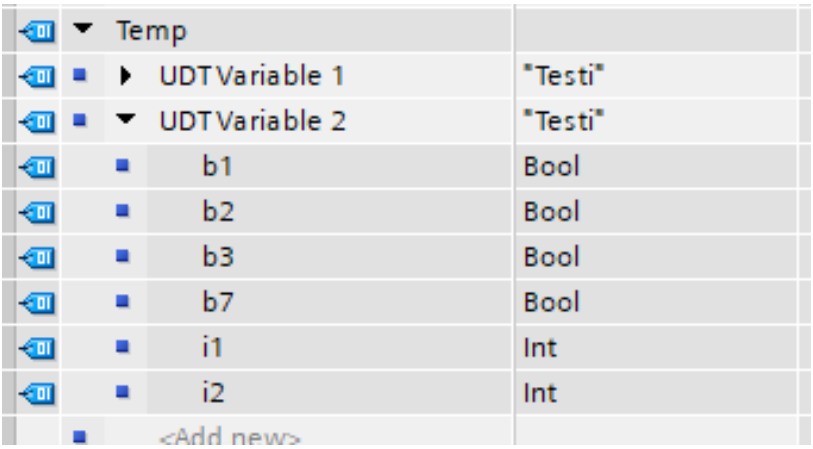
Esimerkkinä UDT:n käytöstä voidaan pitää lämpötilan tallennusta vuorokauden aikana. Samaa UDT:tä voidaan käyttää tallentamaan lämpötiloja lukuisilta antureilta. Luodaan UDT, jossa on 24 REAL-muuttujaa taulukossa ja lämpötila otetaan tunnin välein. Näin ollen samaa UDT:tä voidaan käyttää eri antureiden sisääntulotiedon käsittelyyn, eikä jokaiselle mittaustulokselle tarvitse luoda omaa muuttujaa. Samoin UDT:n kautta on helppo tallentaa lämpötilat päivän päätteeksi lokiin, josta voidaan tarkastella lämpötilahistoriaa. Vuorokauden kuluttua

nollataan UDT -muistipaikat ja tallennetaan vuorokauden ensimmäinen lämpötila jälleen UDT-taulukon ensimmäiseen muuttujaan.

Alla kuvissa 7 ja 8 nähdään, kuinka UDT:tä voidaan käyttää Siemensin ohjelmointityökalulla. Kuvassa 7 luodaan UDT ja asetetaan lähtöarvot muuttujille. Kuvassa 8 luodaan kaksi muuttujaa, jotka saavat tietotyypikseen juuri luodun tietotyypin ’Testi’.



Kuva 7. UDT luonti.

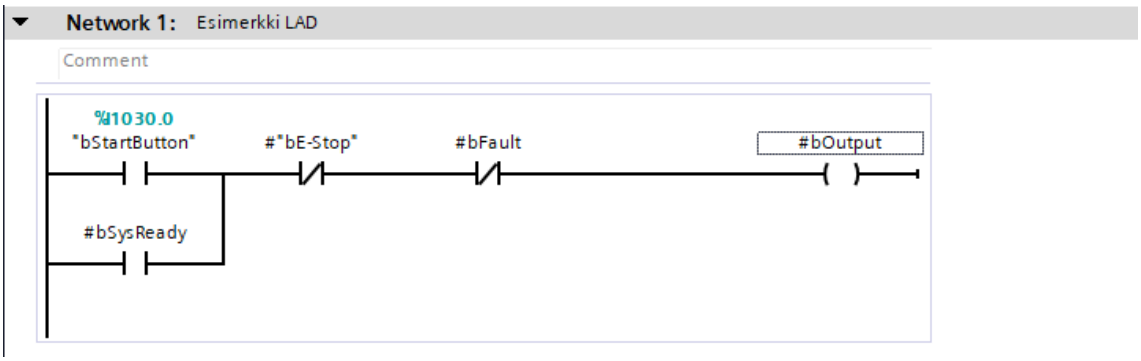


Kuva 8. UDT käyttö.

4.3.2 Ladder Diagram

Ladder Diagram (LD), joka on nimetty Siemensin järjestelmissä Ladder Logic (LAD), on graafinen ohjelmointikieli, joka tunnetaan suomeksi tikapuukaaviona.

Kyseinen kieli muistuttaa jonkin verran teollisuuden piirikaavioita. Tikapuukaavion oikeassa reunassa on virtakiskoa muistuttava viiva ja oikealla nollakiskoa muistuttava viiva. Tikapuukaavion symbolit muistuttavat sähkökuvista tuttuja relekoskettimia, jotka voivat olla joko normaalisti, suljettu tai auki. Samoin ehtolausekkeen aktivoiva muuttuja muistuttaa käämiä. Tikapuukaavio on yksi suosituimmista kielistä vielä tänäkin päivänä, koska useat ohjelmoijat ovat edistyneitä lukemaan sähkökuvia. Tikapuukaavio sopii hyvin käytettäväksi, jos käsitellään digitaalista tietoa. [Keinänen ym. 2008: 224.]

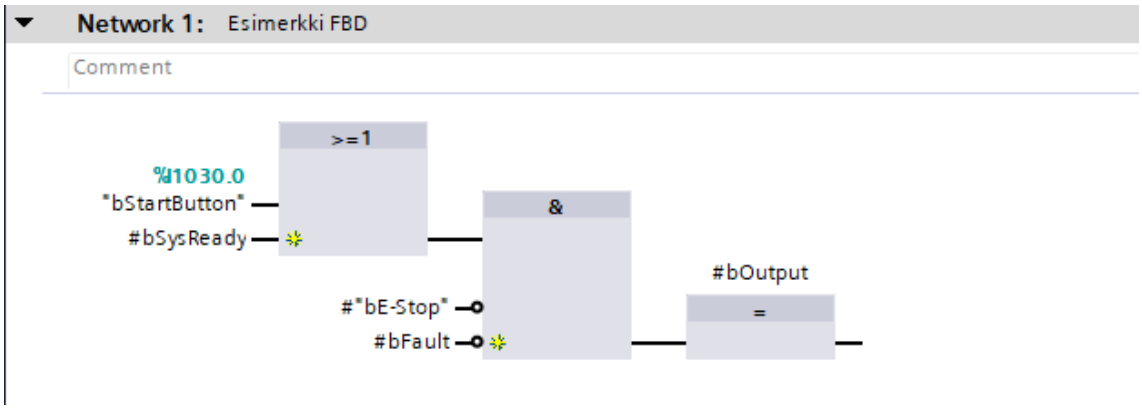


Kuva 9. Esimerkki LAD-ohjelmointikielestä.

Esimerkissä nähdään, kuinka LAD-ohjelmointikielellä toteutetaan TAI-ehto, sekä JA-ehto. Asettamalla muuttujat päällekkäin, saadaan TAI-ehto ja asettamalla muuttujat peräkkäin saadaan JA-ehto. Ohjelmassa voidaan muuttujat asettaa normaalisti auki tai normaalisti suljettu -tilaan. Esimerkissä ’bStartButton’ on normaalisti auki ja ’bFault’ on normaalisti suljettu. Ohjelmointikieltä luetaan vasemmalta oikealla, kunnes saavutetaan aktivoitava muuttuja. Tämän jälkeen siirrytään tarkastelemaan seuraavaa verkkoa (network). Esimerkissä ’bOutput’ saa arvon TRUE, jos ’bE-stop’ ja ’bFault’ eivät ole aktiivisia. Samalla joko ’bStartButton’ tai ’bSysReady’ tulee olla aktiivinen. Esimerkissä ’bStartButton’ esittää manuaalista prosessikierron käynnistystä, kun taas ’bSysReady’ aktivoituu, kun edellinen prosessikierto on päättynyt ja uusi voidaan aloittaa.

4.3.3 Function Block Diagram

Function Block Diagram (FBD), suomeksi toimintalohkokaavio, rakentuu toisistaan yhdistetyistä toimilohkoista. Toimintalohkokaavio on toinen kolmesta graafisesta ohjelmointikielestä IEC 61131-3 -standardissa. Toimintalohkokaavio muistuttaa ohjainkorttikaaviota, joita käytetään mikropiireissä. Käytettävyydeltään toimintalohkokaavio on hyvin samanlainen tikapuukaavioon nähden, mutta toimintalohkokaavioissa on käytettävissä EN/ENO portit. Toimintalohkokaavio ei toimi, jos EN-porttiin ei anneta TOSI-signaalia. Jos yksittäinen lohko on aktiivinen, ENO-portti on myös aktiivinen, joten lohkoja voidaan rakentaa ketjuun. [Keinänen ym. 2008: 224.]



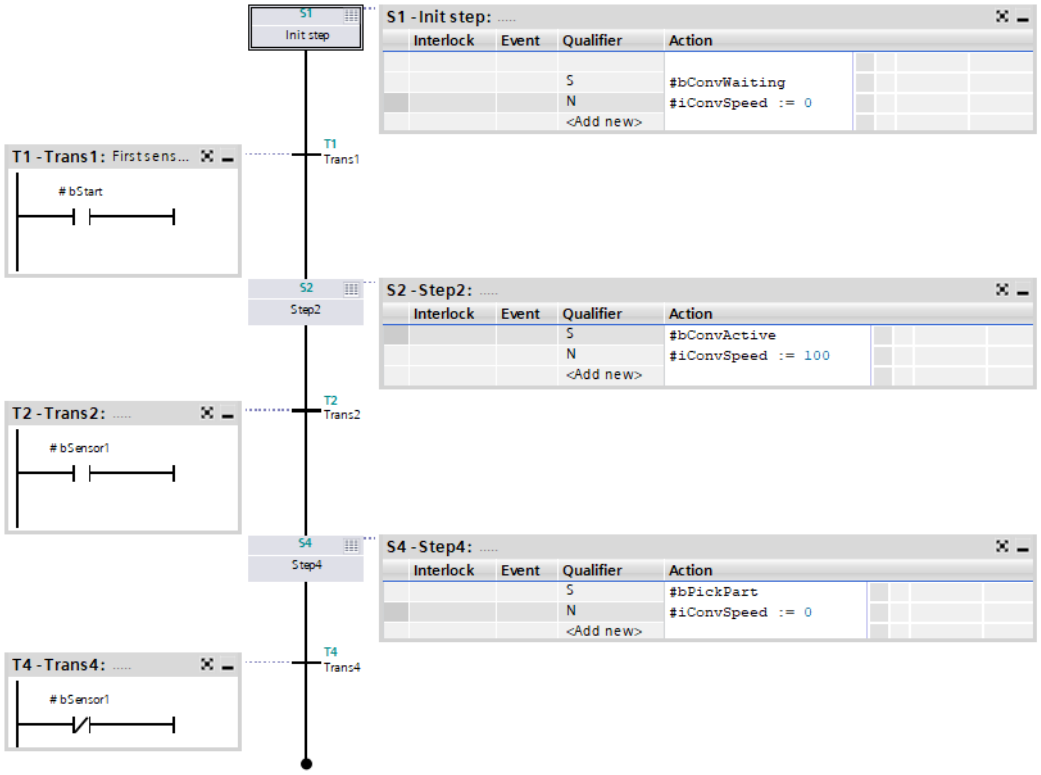
Kuva 10. Esimerkki FBD-ohjelmointikielestä.

Esimerkin toiminta on täysin sama, mutta esitysmuoto on erilainen. JA-ehto, TAI-ehto ja aktivoitava muuttuja ovat esitetty omina lohkoinaan. Normaalisti suljetut muuttujat ovat FBD-ohjelmointikielessä negaatioitu, joka näkyy kuvassa 10 mustana pienenä pallona. Samoin LAD-ohjelmointikielen kanssa, FBD-ohjelmointikieltä luetaan vasemmalta oikealle ja ylhäältä alas.

4.3.4 Sequential Function Chart

Sequential Function Chart (SFC), Siemensin mukaan Graph, on usein käytännöllinen ylemmän tason ohjelmointiin. Suomeksi tämä tunnetaan nimellä sekvenssikaavio. SFC on IEC 61131-3 -standardin kolmas graafinen

ohjelmointikieli. Sekvenssikaavio toimii askelten ja siirtymien avulla. Ehdon täyttyessä ohjelma siirtyy seuraavaan askeleeseen. Tällä ohjelmointikielellä tilakoneen luominen on vaivatonta ja toimintamoodin valinta on helppoa. Sekvenssikaavio ei kuitenkaan tarjoa monipuolisia työkaluja loogiseen ohjelmointiin. Sekvenssikaavio on usein käytössä ohjelman runkona, johon liitetään aliohjelmia, jotka on luotu muilla ohjelmointikielillä. Joitain sekvenssisiä toimintoja voidaan kuitenkin ohjelmoida kyseisellä ohjelmointikielellä. Esimerkiksi liikennevalojen kierto on vaivatonta ohjelmoida, käyttämällä SFC-ohjelmointikieltä. [Keinänen ym. 2008: 224 - 225.]



Kuva 11. Esimerkki GRAPH-ohjelmointikielestä.

Esimerkissä on yksinkertainen sekvenssi, jossa kappale liikkuu sensorille, josta robotti nostaa kappaleen pois. GRAPH-toiminta perustuu askelmien (step) ja siirtymien (transition) käyttöön. Askelissa määritellään mitä kyseisen askeleen aikana tapahtuu ja ehdon avulla hypätään seuraavaan askeleeseen. Samanaikaisesti voi olla useita polkuja. Esimerkin alkutilanteessa muuttuja

’bConvWaiting’ on aktiivinen, joka indikoi, että kuljetin on paikallaan ja odottaa start-komentoa. Samoin nopeus asetetaan muutujalle ’ConvSpeed’ nollaksi. Jos ensimmäinen ehto, eli start-nappia painetaan kuljetin liikkuu nopeudella 100 RPM, kunnes sensori 1 havaitsee kappaleen. ’bPickPart’ aktivoituu, joka on signaali esimerkiksi robotille, joka nappaa kappaleen kuljettimelta. Kun sensori ei havaitse kappaletta, palataan takaisin alkupisteeseen, jolloin odotetaan start-nappia.

4.3.5 Structured Text

Structured Text, Siemensin nimeämänä Structured Control Language (SCL), pidetään haastavimpana logiikkaohjelmointikielenä, sillä se vaatii hieman syvempää ohjelmointiosaamista. Suomeksi käskylistaohjelmointi, muistuttaa syntaksiltaan Pascal-ohjelmointikieltä. Siemensin ohjelmointiympäristö mahdollistaa SCL:n käyttämistä, joko omana aliohjelmanaan tai luomalla yksittäisen SCL-komentolohkon (Network). Yksittäistä komentolohkoa voidaan käyttää, jos tulee tarve silmukalle tai halutaan pakottaa kaikki muuttujat FALSE-tilaan. Tämä on oiva työkalu ohjelmoijalle, sillä ohjelmasta voi tehdä helpompilukuista jakamalla ohjelmaa useampaan komentolohkoon. [Keinänen ym. 2008: 224.]

Käskylistaohjelmointi on ohjelmointikielistä kaikista monikäyttöisin. Taitava ohjelmoija voisi kirjoittaa kokonaisen logiikkaohjelman ainoastaan SFC-kielellä. SFC on kuitenkin vaikealukuisin kieli, jos toinen ohjelmoija lukee koodia ensimmäistä kertaa. Näin ollen usein aliohjelmasta tuodaan monitoroitavat muuttujat pääkiertoon, joka on esimerkiksi toimintalohkokaavio-kielellä. Käskylistaohjelmointi perustuu pääosin IF-THEN-ELSE-rakenteeseen. Käskylistaohjelmointi kielellä on etuja, mitä muilla kielillä ei ole. Esimerkiksi LOOP-, sekä CASE-rakenteet ovat vain käskylistaohjelmoinnilla mahdollista toteuttaa. Samoin koodin jäsentely tietynlaiseen sisällysluetteloon onnistuu REGION ja END_REGION -väliin.

Ohjelmakierron manipulointi on yksi ehdoton etu SCL-ohjelmointikielessä. Esimerkiksi ohjelmakierto aliohjelmassa voidaan lopettaa RETURN-käskyllä, jolloin

ohjelma jatkaa kiertoa pääohjelmassa seuraavaan aliohjelmaan. Samoin silmukoita voidaan hyödyntää, jolloin ohjelma ei etene, ellei silmukka ole suoritettu. Silmukasta voidaan ehdon mukaan poistua EXIT-käskyllä tai GOTO-käskyllä.

```
// Jos Fault on aktiiviinen poistu aliohjelmasta ja pakota arvot 0-tilaan.
WHILE #bFault DO
    #bMoveUp := FALSE;
    #bMoveDown := FALSE;
    #iVelocity := 0;
    RETURN;
END_WHILE;

//Valitse kuljetin. 1 = vasen & 2 = oikea.
CASE #iConvPick OF
    0:
        GOTO Here;
    1:
        IF #bMoveDown THEN
            #iVelocity := 10;
        ELSIF #bMoveUp THEN
            #iVelocity := -10;
        ELSE
            #iVelocity := 0;
        END_IF;
    2:
        IF #bMoveDown THEN
            #iVelocity := 10;
        ELSIF #bMoveUp THEN
            #iVelocity := -10;
        ELSE
            #iVelocity := 0;
        END_IF;
END_CASE;

//Jump Label
Here:
```

Kuva 12. Esimerkki SCL-ohjelmointikielestä.

Esimerkissä on yksinkertainen ohjelma, joka kuvastaa kahden kuljettimen toimintaa. Esimerkissä on käytössä silmukka-, tila- ja ehtorakenne. Aliohjelma tarkistaa onko ohjelma havainnut vikaa. Mikäli vika on aktiivinen ohjelmassa, silmukka käy muuttujat läpi ja asettaa ne epätosi-tilaan ja nopeuden varmistukseksi 0. Tällöin nappia painamalla ohjelma ei liikuta kuljettimia. RETURN-käsky palauttaa kierron pääohjelmaan. Jos vikaa ei havaita, ohjelma siirtyy tilakoneeseen. Jos tilakoneen muuttuja ’iConvPick’ on 1, käytössä on vasen kuljetin, Muuttujan ollessa 2, käytössä on oikea kuljetin. Muuttujat ’bMoveDown’ ja ’bMoveUp’ simuloivat oikeita painonappeja. ’iConvPick’ ollessa 1 ja

’bMoveDown’ ollessa aktiivisena vasen kuljetin liikkuu taaksepäin nopeudella 10. Jos painonappeja ei ole painettu ohjelmakierto, hyppää tilarakenteen yli.

4.3.6 Instruction List

Instruction List (IL), Siemensin mukaan Statement List (STL), on yksi vanhimmista PLC-ohjelmoinnissa käytetyistä kielistä. IL on toinen IEC 61131-3 tekstipohjaisista kielistä. IL on matalan tason ohjelmointikieli, joka koostuu allekkaisista ohjeista, jotka suoritetaan yksi kerrallaan. IL muistuttaa etäisesti binääristä konekieltä, jota esimerkiksi CPU suorittaa. Ohjeet ovat loogisia toimintoja tai laskutoimituksia. Matalan tason ohjelmointikielenä IL:n etuna on nopeus, tehokkuus ja alhainen muistinkäyttö. Kuitenkin IL:n syntaksi on ensisilmäyksellä hankalalukuista, joten vianetsintä on usein kentällä vaativaa. Täten työntekijän, joka on kentällä, olisi helpompi etsiä vikaa graafisista ohjelmointikielistä, jotka ovat helpompilukuisia. [Collins 2019.]

IL on nykyaikaisessa ohjelmoinnissa melko harvoin käytetty kieli, koska kieli on merkitty standardissa marraskuussa 2017 vanhentuneeksi ja sen käyttöä ei suositella. Standardiorganisaation mukaan kyseinen kieli ei ole ajantasainen ja muut ohjelmointikielet ovat parempia käytettäväksi nykyaikaisessa ohjelmointiympäristössä. Näin ollen kieltä saattaa nähdä pääosin vanhoissa PLC-ohjelmissa. Kielen ollessa hyvin vähän käytetty, on oletettavissa, että laitevalmistajat lopettavat kielen tukemisen lähiaikoina. Metsolla ei ole lisenssiä kyseiseen ohjelmointikieleen, joten esimerkkiä ei ole saatavilla. [Collins 2019.]

5 Siemens

Koska Metso käyttää pääosin Siemensin logiikoita, niin ohessa käydään lyhyesti Siemensin logiikat, sekä oleelliset moduulit. Ohessa on myös lyhyt tiivistys Siemensin tarjoamasta ohjelmointiympäristöstä. Siemens on tunnettu kansainvälinen yritys, joka tarjoaa laitteita teolliseen automaation, terveydenhuoltoon, logistiikkaan ja kuluttajille kodinkoneiden muodossa.

5.1 Katsaus Siemensin logiikoihin

Metsolla on käytössä pääosin Simatic S7-1200- ja Simatic S7-1500 -tuoteperheiden logiikat, joten muita sarjoja ei käsitellä. Siemens tunnetaan logiikkamarkkinoilla laadukkaista tuotteista ja laajasta valikoimasta logiikoita. Siemens on myös yksi ensimmäisistä logiikkavalmistajista, joten tuotekehitys on tänä päivänä ensiluokkaista. Molemmat sarjat noudattavat IEC 61131-3 -standardia ja tukevat kaikkia nykyaikaisia kenttäväyliä.

Oleellisin ero Simatic S7-1200 ja Simatic S7-1500 välillä on suoritusteho ja muistin määrä. 1200-sarjan logiikat ovat hinnaltaan huokeampia ja ovat tarkoitettu pieniin-, sekä keskisuuriin sovelluksiin. 1200-sarjan logiikat ovat erinomainen valinta yksinkertaisiin sovelluksiin tai yhden laitteen ohjaukseen. 1500-sarjan logiikat ovat sen sijaan kalliimpia, mutta ne voivat ohjata komplekseja automaatiojärjestelmiä. 1500-sarjan logiikka mahdollistaa keskisuuren- ja suuren järjestelmän ohjauksen yhdellä logiikalla. 1500-sarja mahdollistaa myös kehittyneen liikeohjauksen, parannetun tietoturvan ja kehittyneet turvallisuusominaisuudet. Alla tiivis vertailu oleellisista teknisistä tiedoista kahden logiikan välillä, jotta hahmotetaan käytännössä sarjojen erot. [Salama.]

5.1.1 Tiivistys Simatic S7-1200 teknisistä tiedoista

Valitsin esimerkiksi Simatic S7-1200 sarjan CPU 1212C DC/DC/DC logiikan. Artikkelinumero on kyseisellä logiikalla 6ES7212-1AE40-0XB0. Hintaa logiikalle tulee suomalaisen toimittaja Soneparin mukaan 400,15 €. Kooltaan logiikka on kompakti, joka painaa vain 370 grammaa ja ulkoiset mitat ovat 90 x 100 x 75 mm. Logiikan muistit jakautuvat seuraavasti: työmuistia (Work memory) on 50 kilotavua, varmuuskopioon (Retentive memory) on varattu 10 kilotavua ja lataukseen (Load memory) löytyy valmiina megatavun verran muistia, mutta lisää on mahdollista saada ulkoisen SD-muistikortin avulla. Valmiina logiikassa on 8 digitaalista sisääntuloa ja 2 analogista sisääntuloa, sekä 6 digitaalista ulostuloa. Suorituskykyä Siemens mittaa ajanyksikkönä suorittaa operaatio, esimerkiksi matemaattinen operaatio. Aikaa kuluu seuraavasti, Boolean tietotyypin

käsittelyyn 0,08 ms, Word tietotyypin käsittelyyn 1,7 ms ja Real tietotyypin käsittelyyn 2,3 ms. [Siemens -a.]

5.1.2 Tiivistys Simatic S7-1500 teknisistä tiedoista

Siemens Simatic S7-1500 sarjan logiikaksi vertailuun valitsin 1518–4 PN/DP, jonka artikkelinumero on 6ES7518-4AP00-0AB0. Hintaa tällä logiikalla on noin 9066 €. Kooltaan logiikka on suurempi 175 x 147 x 129 mm ja logiikka on varustettu pienellä näytöllä, josta saadaan oleellista tietoa ilman ohjelmointiympäristöön liittämistä. Painoa logiikalla on 1988 grammaa. Muistia logiikassa on seuraavasti: Työmuistia 4 megatavua ohjelman suoritukseen ja 20 megatavua tiedon väliaikaiseen säilytykseen. Lataukseen tarvitaan erillinen muistikortti joka voimaksimissaan kooltaan olla 32 gigatavua. Logiikka käsittelee boolean tietotyypin ohjeen 1 nanosekunnissa, Word tietotyypin 2 nanosekunnissa ja Real tietotyypin laskutoimituksen 6 nanosekunnissa. [Siemens -b.]

5.2 Siemensin ohjelmointiympäristö

Simatic S7-1200- ja Simatic S7-1500 -sarjan logiikoita on mahdollista ohjelmoida vain Siemens Totally Integrated Automation Portal-ohjelmointityökalulla. TIA Portalilla suoritetaan logiikan ohjelman ohjelmointi, käyttöliittymän suunnittelu, turvallisuustoimintojen konfigurointi ja käyttöönotto. TIA Portalissa on mahdollista simuloida logiikan toimintaa, sekä virtuaalinen käyttöönotto voidaan suorittaa yhdistämällä TIA Portal Siemens NX-sovellukseen S7-PLCSIM Advanced avulla.

5.3 Siemensin paneelit

Siemens tarjoaa logiikoiden ohelle omia HMI-paneeleita (Human-Machine Interface). Näyttöpaneeleilta voidaan ohjata prosessia prosessin ollessa käynnissä ja operaattori saa tärkeää parametritietoa näytöltä. Näytöltä voidaan myös tarkastella prosessin historiaa, esimerkiksi trendianalyysistä. Näytöltä saadaan myös kriittistä tietoa mahdollisista hälytyksistä, samalla hälytykset voidaan

kuitata näytöltä. Nykypäivänä HMI-paneelit suunnitellaan hyvin käyttäjäystävälliseksi, eli operaattoreiden on helppo ymmärtää suunniteltu käyttöliittymä. Samalla osa prosessin ohjauksesta voidaan piilottaa salasanan taakse.

5.4 Siemens etä-IO

Ohjelma on suunniteltu käytettäväksi Simatic ET 200SP etä-IO:lla, joten on hyvä ymmärtää, miksi sakeuttimelle ei ole omaa logiikkaa. Suurin syy on kustannustehokkuus. Etä-IO on huomattavasti halvempi, jos logiikassa riittää suoritusteho pyörittämään useaa laitetta etä-IO avulla. Etä-IO mahdollistaa myös ohjauslaitteen sijoittamisen laitteen viereen. [Siemens c.]

SIMATIC ET 200SP on kompakti modulaarinen etä-IO. Lisämoduuleja voidaan lisätä jopa 64 kappaletta. Moduuleja voidaan vaihtaa myös laitteen ollessa käynnissä. ET 200SP tukee kaikkia nykyaikaisia kenttäväyläprotokollia. Lisäosana voidaan liittää esimerkiksi turvallisuusmoduuleja, moottorin käynnistysmoduuli ja energiamittari. ET 200SP toimii -30 asteen ja 60 asteen välissä ja jopa 5000 m korkeudessa. [Siemens -c.]

6 Ohjelman suunnittelu

6.1 Tutustuminen malliohjelmaan

Alkuperäisen tiedon mukaan ohjelmaa ei tarvitse suunnitella alusta asti, vaan valmis ohjelmarunko on jo olemassa, jota muokkaamalla saadaan toimiva ohjelma. Tähän aikaansaamieni tietojen mukaan ohjelma toimii halutulla tavalla.

Ohjelma oli luotu täysin espanjaksi, joka aiheutti ensimmäisen haasteen syventyä ohjelman toimintaan. Aluksi koko ohjelma tuli kääntää englanniksi. Kääntäminen osoittautui hyvin hankalaksi, sillä ohjelmassa oli käytetty erittäin paljon lyhenteitä. Myös toiminnankuvaus oli täysin espanjaksi, joten käännösvirheitä ilmeni aika ajoin, joka luonnollisesti sekoitti ajatusprosessia. Ohjelmaa ei ole kommentoitu juuri ollenkaan, joten joidenkin muuttujien tarkoitusta ohjelmassa

oli miltei mahdotonta selvittää. Oman haasteensa loi myös ohjelmaan luodut aliohjelmat, jotka eivät kuitenkaan olleet käytössä sakeuttimessa. Esimerkiksi analogiatulon käsittelyyn oli luotu 3 aliohjelmaa, joista yksi oli ohjelmakierrossa. Muut oli poissuljettu negaation avulla, jota en aluksi huomannut.

Ohjelman rakenne oli myös todella hämmentävä. Ohjelmasta löytyi hyvin paljon käyttämättömiä muuttujia, eikä aliohjelman sisäisiä muuttujia juurikaan käytetä. Suurin osa muuttujista löytyy Tag-taulukosta. Tag-taulukossa on noin 350 kappaletta muuttujia, joista noin 150 on käytössä ohjelmassa. Samoin ohjelman rakenteessa on epäjohdonmukaisuuksia ja ohjelma ei ole optimaalisesti rakennettu. Muistinkäyttö ja ohjelman luettavuus oli alkuperäisessä ohjelmassa puutteellista ja ohjelmaa oli jaettu kymmeniin verkkoihin.

Projektin edetessä kävi lopulta ilmi, että ohjelma ei toimi toiminnankuvauksen mukaisesti, eli täysin automaattisesti. Ohjelma ajaa sakeutinta alarajalla ellei haraa nosteta manuaalisesti. Ohjelmassa on ajateltu, että korkeutta määritellään setpoint-integer muuttujalla, mutta muuttujaan kirjoitetaan vain HMI-paneelilta. Näin ollen aloin luomaan uutta ohjelmaa alkuperäisen toiminnankuvauksen mukaan. Ohjelmani noudatti edelleen etäisesti alkuperäistä ohjelmarunkoa, mutta ohjelman toiminallisuutta alettiin automatisoimaan.

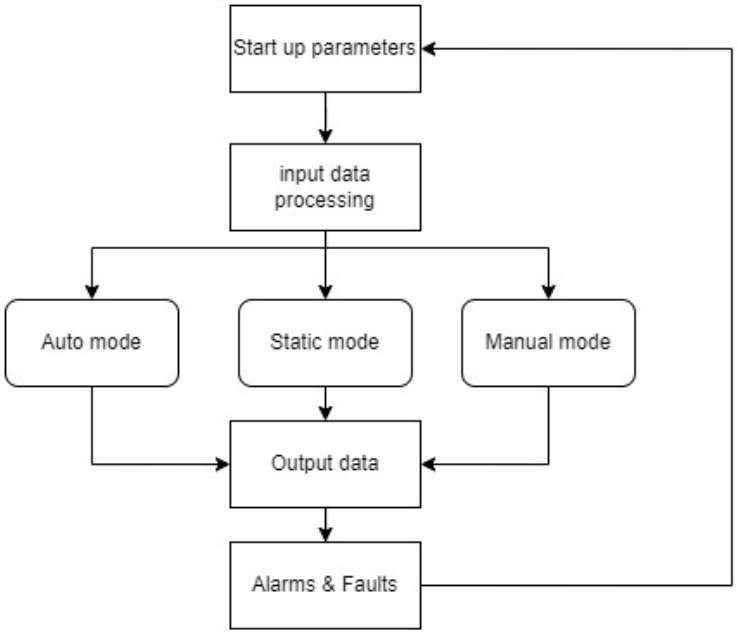
6.2 Toiminnankuvaus

Elokuun lopulla oli palaveri esihenkilöiden kanssa, joilta sain lopullisen toiminnankuvauksen. Samoin sähkökuvat valmistuivat, joten minulla on lopulta hyvä käsitys sisään- ja ulostulo tiedosta. Lopulliseen ohjelmaan tulee 3 moodia, jotka ovat manuaali-, staattinen- ja automaattinen ajo. Manuaalinen – ja staattinen moodi ovat käytössä vain käyttöönotossa ja vianetsinnässä. Automaattisen ajon on tarkoitus olla käytössä, ellei vikatiloja ilmene. Toiminnankuvauksessa ei oteta kantaa ohjelmallisiin hälytyksiin, joten ne tuli minun selvittää itse. Sakeuttimen pääohjelmassa on valmiina tiettyjä hälytyksiä.

6.3 Ohjelman rungon suunnittelu

Ensimmäinen versio ohjelmasta toimii alkuperäisen toiminnankuvauksen mukaan, eikä vastaa uutta toiminnankuvausta, minkä sain elokuun lopulla. Näin ollen ohjelma tuli suunnitella uudestaan. En kuitenkaan halunnut tyhjentää projektia täysin, vaan loin suunnitelman vanhan ohjelman pohjalle.

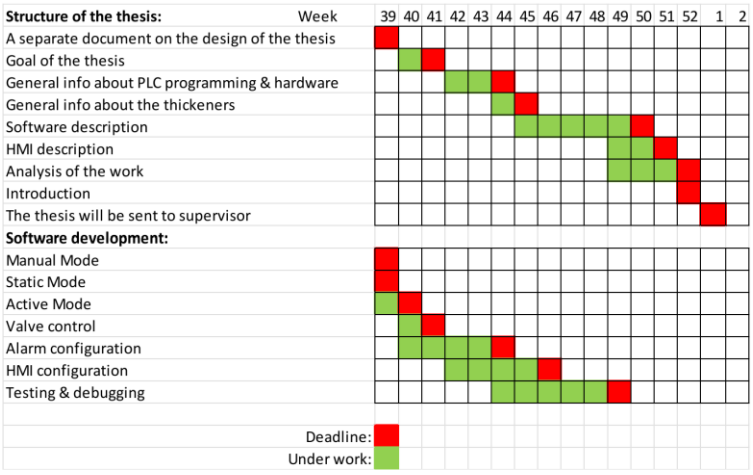
Koska ohjelma on tarkoitus luoda yhden kansion alle Tia Portalissa, ajattelin luoda koko ohjelman yhteen pääohjelmaan, johon tuodaan oleelliset muuttujat. Tästä pääohjelmasta voidaan monitoroida koko ohjelman kulkua. Pyrin ohjelman suunnittelussa ottamaan huomioon, että ohjelmaa tulee käyttämään usea ihminen, joten ohjelman kulku on yksinkertainen ja ohjelmaa on kommentoitu kattavasti. Myös omia muuttujia (UDT) on luotu oleellisille toiminnoille, jotta mahdolliset muutokset on helppo muuttaa koko ohjelmaan. UDT:n avulla HMI-paneelille yhdistettyjä muuttujia on helpompi muokata. Kielinä käytetään pääosin SCL- ja FBD-ohjelmointikieliä, sillä Metsolla on nämä kaksi kieltä aikaisemmissa ohjelmissa vahvasti käytössä. Alla on havainnollistava kaavio ohjelman kierrosta.



Kuva 12. Pääasiallinen ohjelmakierto.

6.4 Aikataulu

Työn etenemisessä on 2 aikataulua, jotka liittyvät opinnäytetyön etenemiseen ja ohjelman rakennukseen. Muuten sain vapaat kädet luoda aikataulut, mutta ohjelma tuli olla valmis ennen vuodenvaihetta. Ohjelman yhdistäminen ei kuulu aikatauluni alle. Aikataulussa on välitavoitteita, jotka on pyritty luomaan realistisesti ja aikatauluun on jätetty aikaa ohjelman kokonaisvaltaiselle testaamiselle. Vaikka aikataulussa lukee, että opinnäytetyö olisi valmis viikolla 1, vuonna 2024, opinnäytetyön viimeinen palautuspäivä on sovittu Metson kanssa helmikuun lopulle.



Kuva 13. Aikataulut.

7 Lopullinen toteutus

7.1 Käytetty laitteisto

Vanhassa ohjelmassa jokaiselle sakeuttimelle oli oma logiikka ohjaamassa laitetta. Jatkossa laitteisiin, joihin toimitetaan uusi lisäosa, tullaan pääosin toimittamaan etä I/O:na toimiva ohjelma. Ohjelma suunniteltiin käytettäväksi SIMATIC ET 200SP etä I/O:ssa. Ohjelma suunniteltiin täysin Siemensin

ohjelmointityökalulla, eli TIA Portalissa. Apuna suunnittelussa käytettiin Microsoft 365 pakettia, josta tärkeimpänä Excel.

7.2 Sisään- ja ulostulot

Sisään-ohjelmaan saadaan tietoa haran vääntömomentista, joka skaalataan 0–100 % alueelle. Sisääntulona saadaan sylintereiltä takaisinkytkennän kautta tietoa. Proportionaaliventtiilit ovat myös varustettu takaisinkytkennällä. Lähes kaikki sisääntulotieto on analogista tässä ohjelmassa. Ulostulona ohjataan proportionaaliventtiileitä ja solenoideja. Proportionaaliventtiilin asennolla saadaan laskettua tai nostettua hydraulisen sylinterin asemaa. Käytännössä siis analogisella tiedolla proportionaaliventtiilille saadaan määriteltyä nousu- ja laskunopeus. Solenoidit avaavat proportionaalisen venttiilin, sen mukaan, miten HMI-paneelilta on nopeus asetettu.

7.3 Ohjelma

Ohjelman pääkierrossa on 11 Networkkia, joista voidaan ohjelmaa monitoroida yhdellä silmäyksellä. Muuten ohjelmaa on pakattu aliohjelmiin, jotta ohjelmaan tehdyt muutokset muuttuvat yhtenäisesti. Samoin esimerkiksi automaattisen toiminnon aliohjelmaa on helppo monitoroida, koska aliohjelma palauttaa kiertoon vain oleelliset muuttujat. Ohjelmaa pyritään selostamaan opinnäytetyössä niin, että toiminta selviää, mutta ohjelmallisesti voidaan näyttää vain pieniä osia.

7.3.1 Sakeuttimen parametrit

Sakeuttimen parametrit asetetaan ennen sakeuttimen käyttöönottoa HMI-paneelilta. Sakeuttimen parametreja ei voi muokata ilman salasanaa, joten operaattori ei voi päästä käsiksi sakeuttimen parametreihin kesken sakeuttimen ajon. Sakeuttimen ohjelma toimii asetettujen parametrien mukaan automaattisesti, samoin HMI-paneelilla näkyy parametreja vain asetettujen parametrien mukaan. Ohjelmallisesti parametrien konfigurointi on toteutettu kahdella Int-muuttujalla, joihin siirretään arvo sen mukaan, mitä oikeassa sovelluksessa on

käytössä. Salasanan takaa saadaan asetettua sylinterien lukumäärä, sekä iskunpituus. Jos epähuomiossa on painettu samanaikaisesti esimerkiksi useampi iskunpituus aktiiviseksi, ohjelma ei anna ajaa sakeutinta. Lisäosa voidaan myös asettaa kokonaan pois päältä parametrien konfiguroinnista. Jos lisäosa on pois päältä, ohjelma ei suorita ohjelmakiertoa ja sakeuttimen pääohjelma ottaa sakeuttimen hallintaan.

7.3.2 Sisääntulotiedon käsittely

Sisään tuleva tieto muokataan aliohjelmassa luettavaan muotoon. Siemensin logiikoissa sisään tulevan analogisen tiedon alue on 0–27648, joten sitä on hankala käsitellä ohjelmassa. Esimerkiksi, jos maksimipaine on 100 baaria, niin skaalataan 0–27648, muotoon 0–100. Näin ollen painetta on helpompi käsitellä ohjelmallisesti, sekä HMI-paneelille saadaan suoraan sylinterin oikea paine. Samassa aliohjelmassa asetetaan ohjelmalliset rajakytkimet sylinterien liikkeelle, jotta sylinteriä ei voida ajaa päätyyn asti. Alla käsitellään muutama laskutoimitus, joilla ohjataan sakeutinta.

Ohjelmassa on toteutettu Case-rakenteella sylinterien määrä ja iskunpituus laskutoimituksissa. Parametrien konfiguroinnista tuodun muuttujan mukaan ohjelma käyttää oikeita laskuja. Norm_X-funktiolla saadaan muutettua Int-arvo Real-arvo muotoon 0–100.0. Tämä arvo skaalataan todellisella arvolla, esimerkiksi todellisella paineella. Kuten kuvasta 14 nähdään oikealla ’CylinderCount’ -arvolla päästään Case-rakenteen sisään. Sovelluksen paineet asetetaan HMI-paneelilta. ’ActPress’ on sovelluksen todellinen paine, jonka mukaan toimii muun muassa hälytyslohko. Arvo muutetaan vielä Int-muotoon, sillä ohjelmassa ei ole tarpeellista tarkastella desimaalien tarkkuudella painetta.

```
CASE #CylinderCount OF
    1:
        //**   CYLINDERS **//
        //Hydraulic pressure of cylinders 1-
        #rPressCyl1 := NORM_X(MIN := 0, VALUE := #PressCyl1, MAX := 27648);
        #rPressCyl1 := SCALE_X(MIN := #MinCylPress, VALUE := #rPressCyl1, MAX := #MaxCylPress);
        #ActPress1 := REAL_TO_INT(#rPressCyl1);
```

Kuva 14. Todellisen paineen skaalaus

Koska sisään tulevana tietona ei saada kenttälaitteilta rajakytkindataa, on ohjelmaan rakennettava niin sanotut pehmeät rajat eli ohjelmalliset rajat. Ohjelmallisesti tämä on yksinkertaisesti vain yksi IF-lause. Ehtolauseella tarkastellaan onko, sylinterin asema yli 599 mm, 600 mm iskunpituudella, jolloin ’UpperLimit’ -muuttuja aktivoituu ja sakeuttimen sylintereitä ei voida ajaa ylemmäs. Kyseinen rajakytkin toimii luonnollisesti jokaisessa moodissa.

```
//**LIMIT SWITCHES**//

//Upper//
IF #rPosCyll >=                                                    THEN
    #UpperLimit := TRUE;
ELSE
    #UpperLimit := FALSE;
END_IF;
```

Kuva 15. Ohjelmallinen rajakytkin.

Myös pääsisääntulopainetta mittaava anturitieto tulee skaalata, jotta sitä voidaan käyttää ohjelmassa. Haran paine skaalataan prosenttimuotoon, joten normioinnin jälkeen riittää, että arvoa kerrotaan sadalla. Käyttöönotossa voi paine heitellä jonkin verran, joten ehtolauseella arvo pidetään minimissään 0, jotta ohjelma toimii oikein.

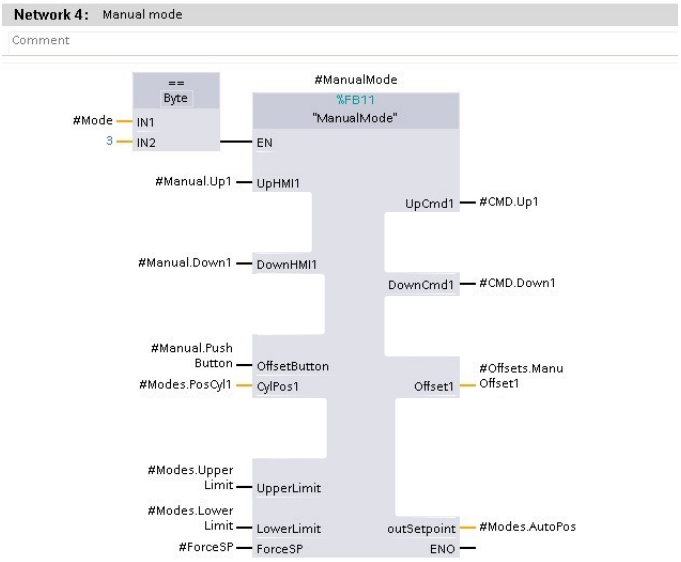
```
//Rake turning pressure
#rTurnPressure := NORM_X(MIN := 0, VALUE := #TurnPressure, MAX := 27648);
#rTorque := #rTurnPressure;
#ActRakePress := #rTurnPressure * 100;

//Turn pressure can't be under 0
//Force to 0
IF #ActRakePress < 0 THEN
    #ActRakePress := 0;
END_IF;
```

Kuva 16. Pääpaineen skaalaus.

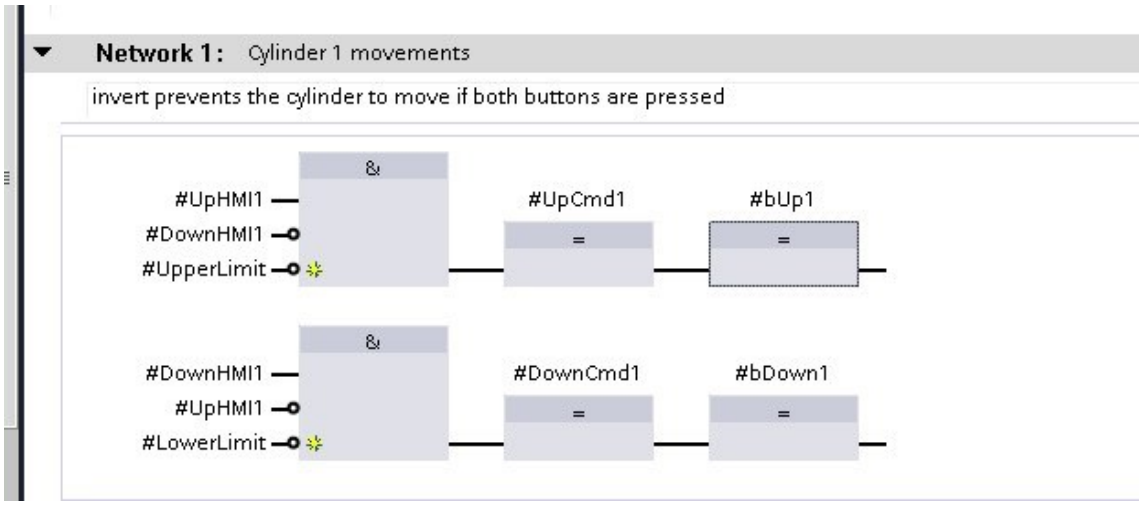
7.3.3 Manuaalinen moodi

Manuaalisella moodilla voidaan sylintereitä ajaa yksi kerrallaan tai kaikkia sylintereitä yhdessä. HMI-paneelilla on jokaiselle sylinterille oma ohjausnappinsa. Monitoroitavassa lohkosta kuvassa 17 nähdään kaikki muuttujat, joita manuaalinen moodi laitteen ajossa tarvitsee. Huomioitavaa on se, että manuaalinen moodi on tarkoitettu lähtökohtaisesti vianetsintään, joten mahdolliset viat eivät estä laitteen ajamista. Ajoa estävät ainoastaan rajakytkimet tai, jos sylintereiden asemaa ajetaan yli 10 mm epätasapainoon. Vianetsinnän vuoksi manuaaliselle moodille on luotu muuttuja ’ForceSP’, jota painamalla HMI-paneelilta, saadaan asetettua uusi asemaohjearvo automaattiselle moodille. Manuaalimoodi on luotu pääosin FBD-kielellä, sillä ohjelmallisesti manuaalimoodi on hyvin yksinkertainen toteuttaa.



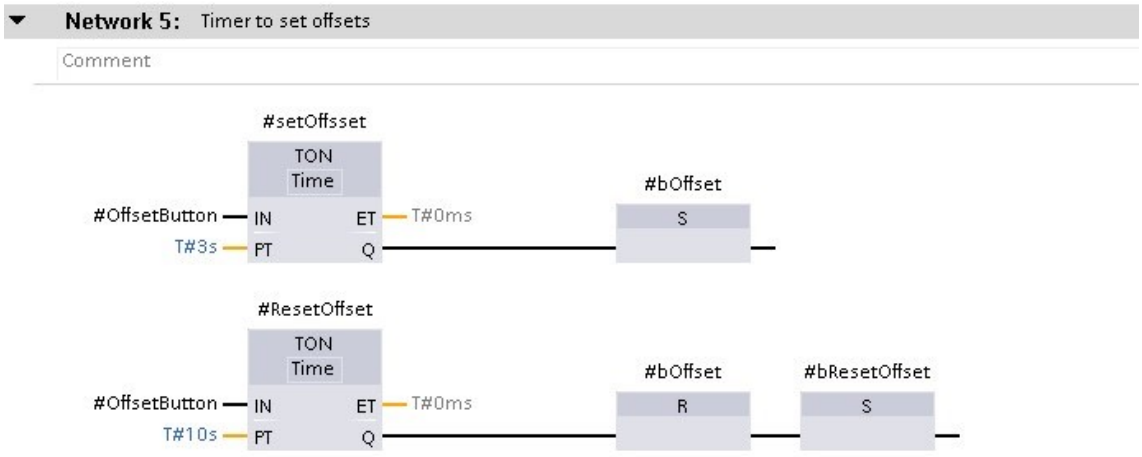
Kuva 17. Pääohjelman manuaalilohko.

Kuvassa 18 nähdään, että sylinterin 1 ajoa estää ainoastaan rajakytkin, joka konfiguroitiin aiemmin. Muuten ohjelmallisesti manuaaliajo ei vaadi hirveästi muistia tai suorituskykyä ollessaan todella yksinkertainen.

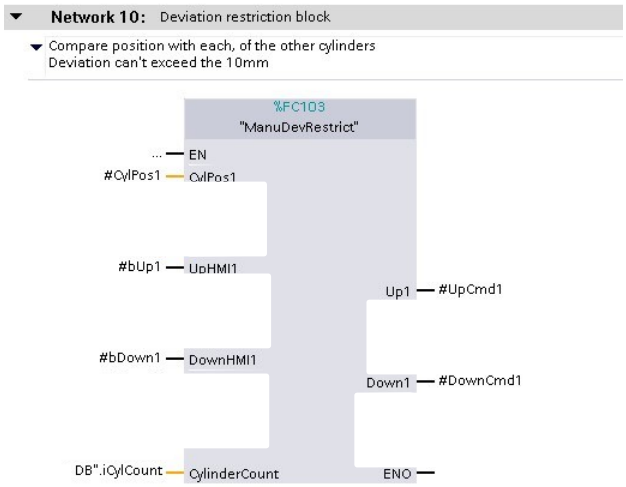


Kuva 18. Manuaalin ajoblokit.

Toiminannankuvauksessa oli myös painonappi, jota painamalla voidaan manuaalimoodista asettaa offsetit sylintereille nykyisen aseman mukaan. Painonappi ei kuitenkaan sähkökuvien mukaan ollut fyysinen painonappi, vaan HMI-paneelin luodaan oma painonappi offseteille. Painamalla 3 sekuntia offsetit asetetaan ohjelmaan ja painamalla 10 sekuntia offsetit nollataan. Tämä on toteutettu Calculate- ja Timer -funktioilla. Ajastimet näkyvät kuvassa 19. Sylintereiden asemissa voi olla vain 10 mm ero toisiinsa nähden, jotta sakeutin ei mene mittavaan epätasapainoon. Tämän vuoksi tuli rakentaa aliohjelma manuaalimoodin alle, joka estää ajon, mikäli asemaero sylintereiden välillä on 10 mm. Aliohjelma siis vertailee sylintereiden asemaa keskenään ja keskeyttää sylinterien ajot, mikäli ollaan yli 10 mm päässä toisistaan. Kuvassa 20 nähdään aliohjelmalle tulevaa tietoa. Aliohjelmalta saadaan lupa ajaa sylinteriä.



Kuva 19. Ajastimet



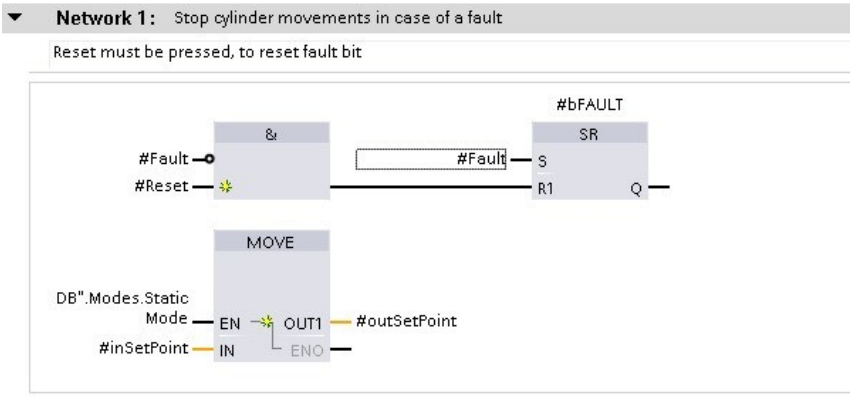
Kuva 20. Aliohjelma, joka tarkkailee poikkeamaa.

7.3.4 Staattinen moodi

Staattinen moodi on puoliautomaattinen moodi, jolla sylintereitä voidaan ajaa yhtä aikaa. Mahdollinen hälytys pysäyttää ajon ja operaattorin tulee palata manuaalisen moodiin ja selvittää vikaa tätä kautta. Staattisella moodilla ajetaan kaikkia sylintereitä samanaikaisesti. Ajon jälkeen sylinterit hakeutuvat samalle tasolle. Sylintereillä on 30 sekuntia aikaa löytää oikea asema ajon jälkeen tai ohjelma aiheuttaa hälytyksen, sillä sylinterit ovat epätasapainossa. Sylintereillä on 2 mm toleranssi, jonka sisällä jokaisen sylinterin tulee olla ohjearvoa.

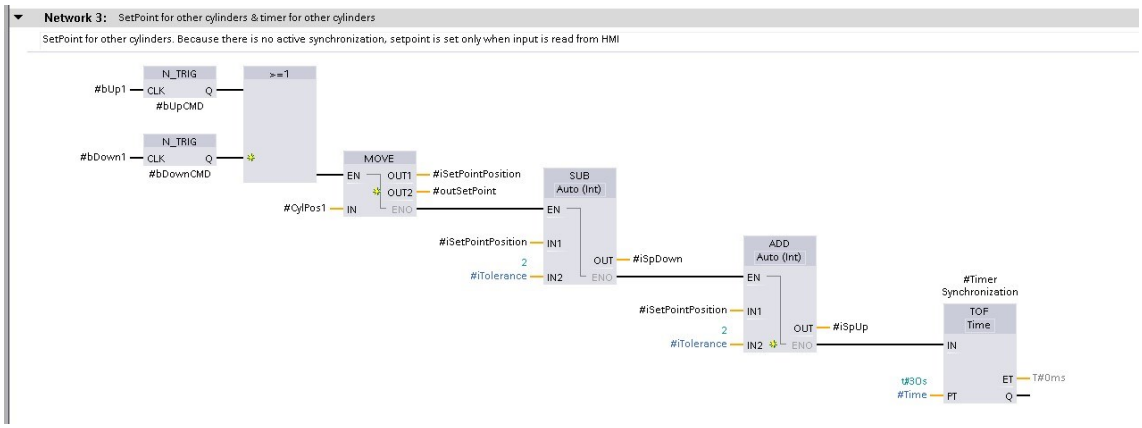
Staattinen moodi ottaa huomioon mahdolliset offsetit, mitä asemalle on asetettu, joko manuaaliselta moodilta tai HMI-paneelilta. Staattinen moodi on myös luotu lähtökohtaisesti FBD-kielellä.

Hälytys aktivoi staattisen moodin sisällä muuttujan #bFault, joka estää sylintereiden ajon. ’bFault’ tulee resetoida HMI-paneelilta, jotta ajoa voidaan jatkaa. Mikäli ohjelma tunnistaa edelleen konfiguroidun vian, hälytystä ei saada kuitattua ja staattinen moodi ei toimi. Ensimmäiseen verkkoon on myös luotu Move-funktio, jolla saadaan siirrettyä ohjearvoa eteenpäin, muuten ohjearvo olisi 0, jos staattisessa moodissa käydään, mutta sylintereitä ei ajeta.



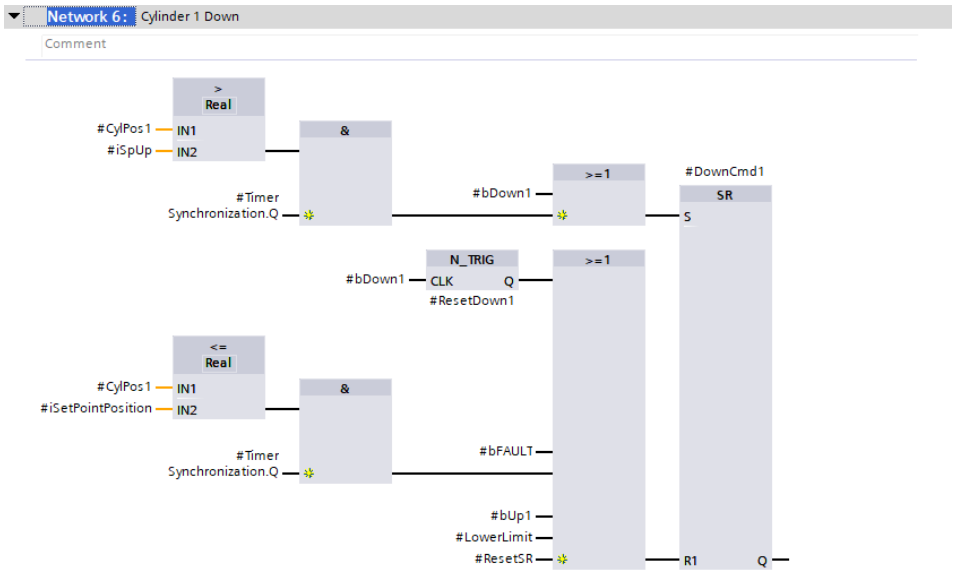
Kuva 21. Staattisen moodin SR-kiikku.

Staattisella moodilla on jonkin verran ehtoja, jotka tulee toteutua, jotta sylinterit voivat ajaa. Sylintereitä ei voi ajaa samanaikaisesti ylös ja alas. Jos sylintereiden ajastin tasapainotukselle on kesken, sylintereitä ei voi tällöinkään ajaa. Ohjelman tila näkyy HMI-paneelilla, jotta operaattori ei yritä turhaan ajaa sylintereitä. Samat rajakytkimet toimivat myös staattisella moodilla. Kuvasta 22. nähdään, kun operaattori nostaa sormen napilta, sylinterin 1 asema asetetaan ohjearvoksi. Samalla tämän ohjearvon mukaan asetetaan toleranssi, joka on 2 mm suuntaansa ohjearvosta ja ajastin sylintereiden aseman korjaamiseksi aktivoituu.



Kuva 22. Ajastin ja toleranssit.

Sylintereiden ajo napista ja sylinterien aseman korjaus on mahdollistettu samalla SR-kiikulla. Alla olevasta kuvasta 23. nähdään, että staattinen moodi palauttaa pääohjelmaan käskyn laskea sylinteriä 1, jos ’bDown1’ on aktiivinen, eli nappia painetaan HMI-paneelilta tai ajastin on aktiivinen ja sylinteri on toleranssin ulkopuolella. Sen sijaan SR-kiikkua resetoi kohtuullisen suuri määrä muuttujia. SR-kiikun resetoi napin nostaminen HMI-paneelilta, hälytys, ajastimen päättyminen, alaraja tai se, kun sylinteri on saavuttanut ohjearvoaseman.

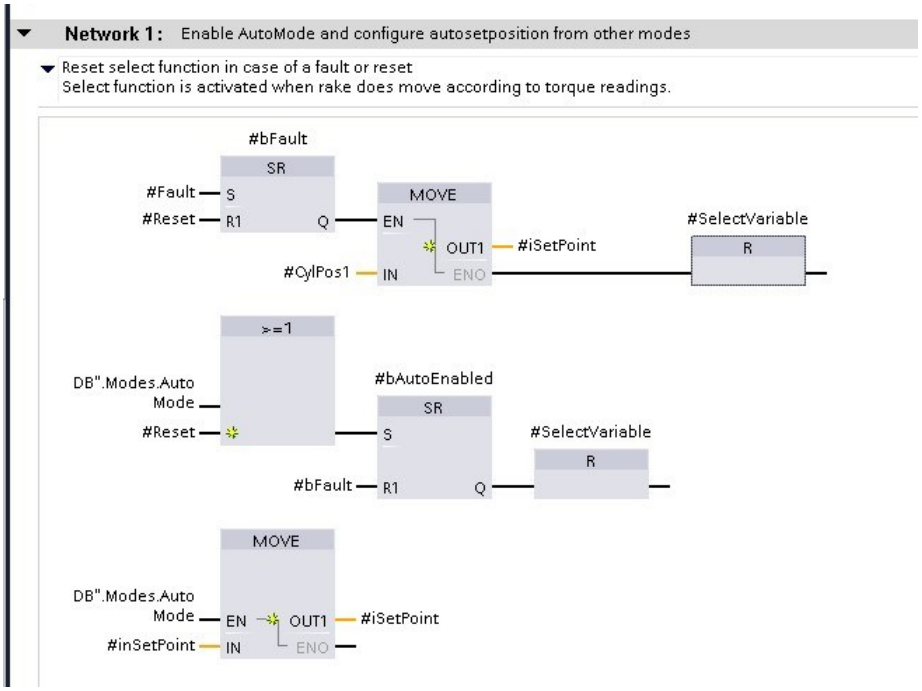


Kuva 23. Sylinterin 1 ohjauksen ehdot.

7.3.5 Automaattinen moodi

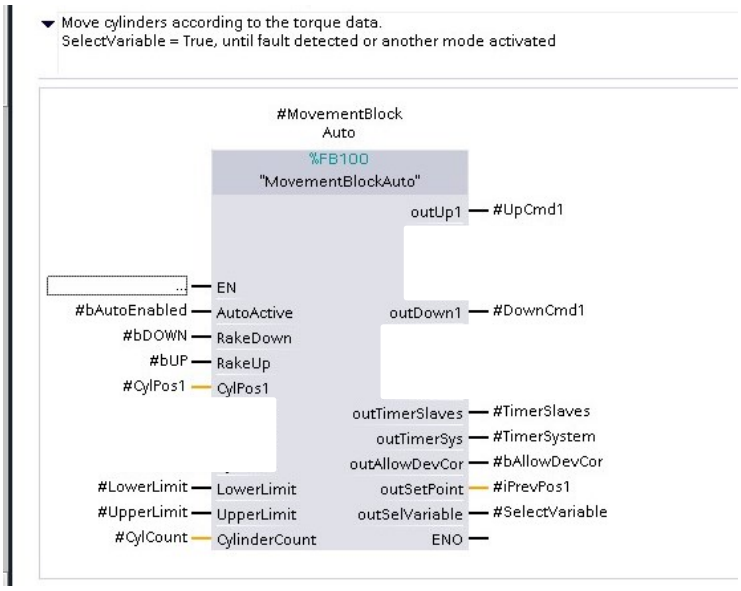
Automaattinen moodi on moodi, joka on lisäosan ydin. Lisäosa ohjaa haraa automaattisesti vääntömomentin mukaan ylös ja alas, samalla korjaten jokaisen sylinterin asemaa. Automaattinen moodi on normaalitilassa kokoaikaisesti päällä ja muita moodeja käytetään vain käyttöönotossa tai mahdollisessa vikatilassa. Kuten kuvasta 24 nähdään manuaalisesti,- tai staattisesta moodista tuotu aseman ohjearvo tuodaan automaattiselle moodille käyttöönotossa, jotta vältytään hälytykseltä. Automaattisessa moodissa ei ole mahdollista ohjata sylintereitä käsin, vaan laite toimii täysin automaattisesti.

Samoin kuten staattinen-, automaattinen moodi ei toimi, jos ohjelma tunnistaa vian. Vikatilan sattuessa sylinterin 1 asema tallennetaan ohjearvoksi. Samoin ’SelectVariable’ resetoidaan. Kyseinen muuttuja päättää otetaanko ohjearvo muilta moodeilta vai automaattiselta moodilta. Jos ’SelectVariable’ on epätosi, käytetään ohjearvoa muilta moodeilta. Ensimmäinen verkko asettaa ’SelectVariable’ -muuttujan arvoksi jatkuvasti epätosi, mutta vääntömomentin korjatessa asemaa, ohjearvo muokataan automaattiselta moodilta ja muuttujan arvoksi pakotetaan tosi. Kuvassa 24 nähdään automaattiajon alustus.



Kuva 24. Automaattisen moodin alustus.

Automaattinen moodi siis nostaa tai laskee haraa vääntömomentin mukaan. Jos vääntömomentti esimerkiksi kasvaa raja-arvon yli, hara lähtee nousemaan sylintereiden avulla. Raja-arvon ylitys aktivoi muuttujan ’bUp’, joka käynnistää haran nostosekvenssin. Kuvasta 25 nähdään muuttujat, joita tarvitaan haran ohjaukseen haran ohjauslohkossa. Ensimmäisenä tarkistetaan, että ei ole havaittua vikatilaa. Ohjauslohkon sisään tulevat muuttujat ’bUp’ ja ’bDown’, jotka aktivoituvat vääntömomentin mukaan. Ohjauslohkoon tulevaa tietoa ovat vielä sylintereiden asema ja rajakytkimet. Ohjauslohkosta ulostuleva tieto ohjaa sylintereitä. Jokaiselle sylinterin liikkeelle on oma muuttuja, jonka aktivoituessa solenoidi aktivoituu. ’bAllowDevCor’ ollessa tosi, automaattinen moodi jatkuvasti korjaa sylintereiden asemaa, jos vääntömomentti on halutulla alueella. Jokaisen vääntömomentin korjauksen jälkeen uusi ohjearvo kirjoitetaan muuttujaan ’iSetPoint’.



Kuva 25. automaattisen ajon lohko

Haran korkeutta säädetään sekvenssisesti. GRAPH-ohjelmointikieli ei ole käytössä 1200-sarjan logiikoissa, joten sekvenssi ohjelmoidaan SCL-kielellä. Sekvenssi voi käynnistyä, mikäli aikaisempi sekvenssi ei ole käytössä tai ohjelma ei tunnista vikatilaa. Sekvenssi koostuu pääosin kolmesta vaiheesta, joita ovat

vääntömomentin korjaus, mahdollinen aseman epätasapainon korjaus ja systeemin lepo. Aluksi ehtolauseella tarkistetaan, onko sekvenssi kesken. Jos sekvenssi ei ole kesken, muuttuja ’bMotionAllowed’ on tosi ja hara liikkuu vääntömomentin mukaan, joka näkyy kuvasta 26.

```
2:  //   cylinders activated //
    //** UP **//
    IF #RakeUp AND #bMotionAllowed AND NOT #UpperLimit THEN
        #outUp1 := TRUE;
```

Kuva 26. 4 sylinterin liike ylös.

Vääntömomentin korjauksen jälkeen aktivoidaan ajastin, jonka aikana mahdollinen sylintereiden epätasapaino korjataan. Ohjelma tunnistaa F_TRIG-funktion avulla, kun haran vääntömomentti siirtyy tosi-tilasta, epätosi-tilaan ja ajastin sylintereiden korjaukselle käynnistyy. Kuvasta 27 nähdään ehdot ajastimen alustukselle. Oikean vääntömomentin saavuttamisen jälkeen tallennetaan uusi ohjearvo, johon sylintereiden asemaa verrataan. Sylintereillä on 20 sekuntia aikaa asettua ohjearvon mukaiseen asemaan.

```
65  //** Set up timer for deviation correction**//
66  #SetUpSlaveTimer(CLK:=(#RakeDown AND NOT #TimerForSystem.Q AND NOT #TimerForSlaves.Q) OR (#RakeUp AND NOT #TimerForSystem.Q AND NOT #TimerForSlaves.Q),
67                   Q=>#bTorqReached);
68
69  //**Timer after torque signal, to correct possible deviation**//
70  #TimerForSlaves(IN:=#bTorqReached,
71                  PT:=#"Timer 20s");
```

Kuva 27. Ajastimen alustus ja initialisointi.

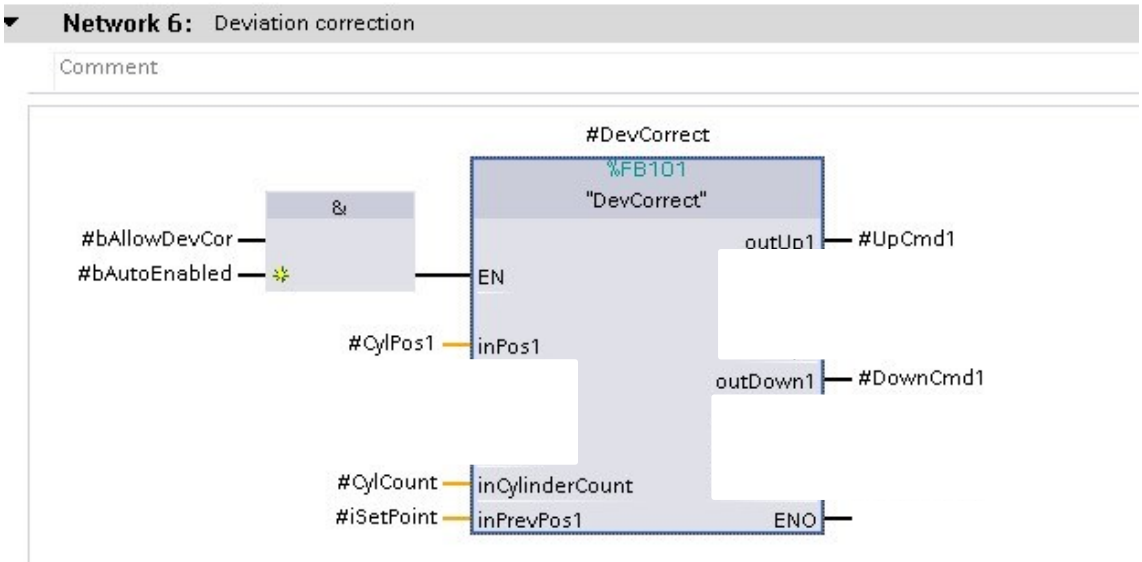
Kun sekvenssi on valmis, muuttuja ’outAllowDevCor’ on tosi ja ohjelma korjaa aktiivisesti sylintereiden asemaa, jos jokin sylinteri on epätasapainossa. Tämä on toteutettu yksinkertaisesti ehtolauseella, joka näkyy kuvassa 28. Samoin ensimmäisen vääntömomentin korjauksen jälkeen uusi ohjearvo kirjoitetaan muuttujalle, koska haran mukaan sylinterien asema on muuttunut. Alemmassa ehtolauseessa kuvassa 28, pakotetaan aiemmin mainittu ’SelectVariable’ tosi-tilaan,

kunnes automaattinen moodi menee pois päältä. Näin automaattinen moodi ottaa ohjearvonsa kyseiseltä aliohjelmalta, eikä muilta moodeilta.

```
118  //** For deviation correction **//
119 IF #TimerForSystem.Q OR #TimerForSlaves.Q OR #RakeUp OR #RakeDown THEN
120      #outAllowDevCor := FALSE;
121  ELSE
122      #outAllowDevCor := TRUE;
123  END_IF;
124
125
126  //** To set correct setpoint**//
127 IF #bTorqReached THEN
128      #outSelVariable := true;
129  ELSIF NOT #AutoActive THEN
130      #outSelVariable := false;
131  END_IF;
```

Kuva 28. Ehdot aseman korjaukselle ja ohjearvolle.

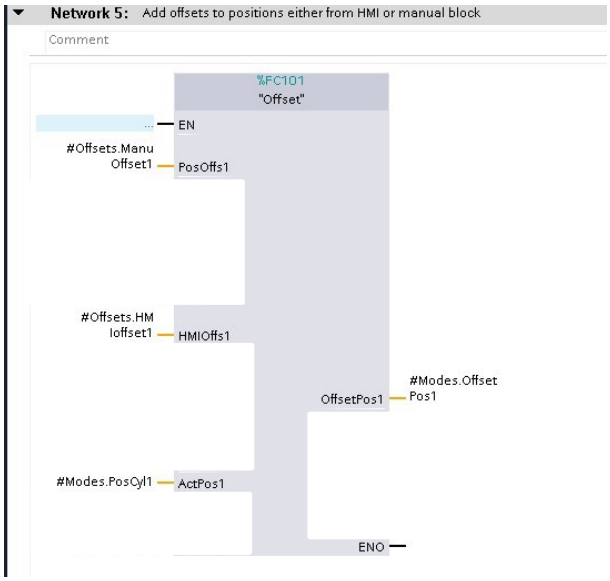
Ohjelma korjaa siis asemaa, jos vääntömomentin sekvenssi ei ole käynnissä. Kuvassa 29 nähdään, että aliohjelma tarvitsee toimiakseen, että ’bAllowDevCor’ on tosi. Samoin automaattinen moodi tulee olla aktiivinen. Asemaa korjataan ohjearvon mukaisesti, joten jokainen sylinteri reagoi välittömästi mahdolliseen poikkeamaan asemassa.



Kuva 29. Poikkeaman korjauslohko.

7.3.6 Offsetit

Staattisella- ja automaattisella moodilla sylintereiden asemalle voidaan asettaa offset, joka on maksimissaan 10 mm todellisesta arvosta. Manuaalisessa moodissa offsetit eivät vaikuta. Offsetit voidaan asettaa joko painamalla manuaalisessa moodissa painiketta tai erilliseltä näytöltä jokaiselle sylinterille erikseen. Offsettien asettamiselle on luotu oma aliohjelmansa, jotta pääkierrosta voidaan monitoroida onko, vahingossa asetettu offsettejä kahdesta paikasta. Kuvasta 30 nähdään, että aliohjelmalle tuodaan todellinen sylinterin asema anturilta ja offset -arvot. Aliohjelma laskee offsetit yhteen ja lisää ne todelliseen aseman arvoon.

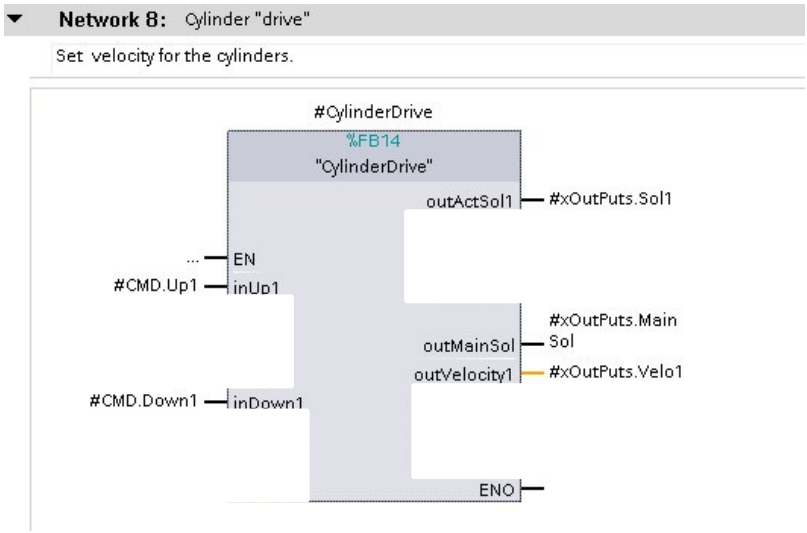


Kuva 30. aliohjelma offseteille.

7.3.7 Sylinterien ohjaus

Kaikki moodit palauttavat aliohjelmistaan pääkiertoon tietoa sylintereiden liikkeestä. Esimerkiksi muuttuja ’CMD.Up1’ on tosi, kun sylinteriä 1 halutaan nostaa. Sylinterien ohjauslohko aktivoi halutun liikkeen mukaan solenoidit proportionaaliventtiilille. Jokaiselle venttiilille on siis luonnollisesti oma solenoidi, joka aktivoituu. Samalla aktivoituu pääsolenoidi, joka avaa pääsyöttöventtiilin. Proportionaaliventtiilille kerrotaan myös haluttu nopeus prosenttiarvona.

Ohjaussignaali lähetetään venttiilille, kuinka paljon venttiiliä halutaan avata. Nopeutta voidaan säädellä jokaiselle sylinterille erikseen HMI-paneelilta. Kuvasta 31 nähdään ohjauslohkolle tuotavat muuttujat. Lähtevät muuttujat ovat kuvassa 31 vielä simulointivaiheessa, joten todellisia ulostuloja ei ole asetettu aliohjelmalle.



Kuva 31. Sylinterien ohjauslohko.

Sylinterin liikkumisnopeutta säädellään proportionaaliventtiilin asennolla. Esimerkiksi sylinteri 1 saa komennon liikkua ylöspäin nopeudella 50 %. HMI-paneelilta tuleva prosenttitieto tulee muuttaa proportionaaliventtiilille tietoon 0–27648, jolloin 50 % on 20601, eli proportionaaliventtiili aukeaa puoleenväliin. Ohjelmassa siis ’CMD.Up1’ on tosi. Tämä aktivoi aliohjelman ’ValveCalcUp’, joka saa HMI-paneelilta nopeustiedon ja aliohjelma avaa proportionaaliventtiiliä halutun verran. Kuvassa 32 aliohjelmalle viedyt tiedot.

```
IF #bUpCMD THEN
    //Cylinder 1//
    "ValveCalcUP"(inVelocity := #rVelocity1,
                  inVelocityOffset := #rVelOffset1,
                  inCMD := #inMoveCMD1,
                  outVelocity => #outVelocity1);
```

Kuva 32. Sylinterin 1 nopeuden säätö.

Mikäli sylintereitä ei tarvitse ajaa, proportionaaliventtiili sulkeutuu, eli ulostulo muuttujalle annetaan arvo 13824. Kuvassa 33 nähdään ehtolause, jolla tarkistetaan, onko jokin komento tosi.

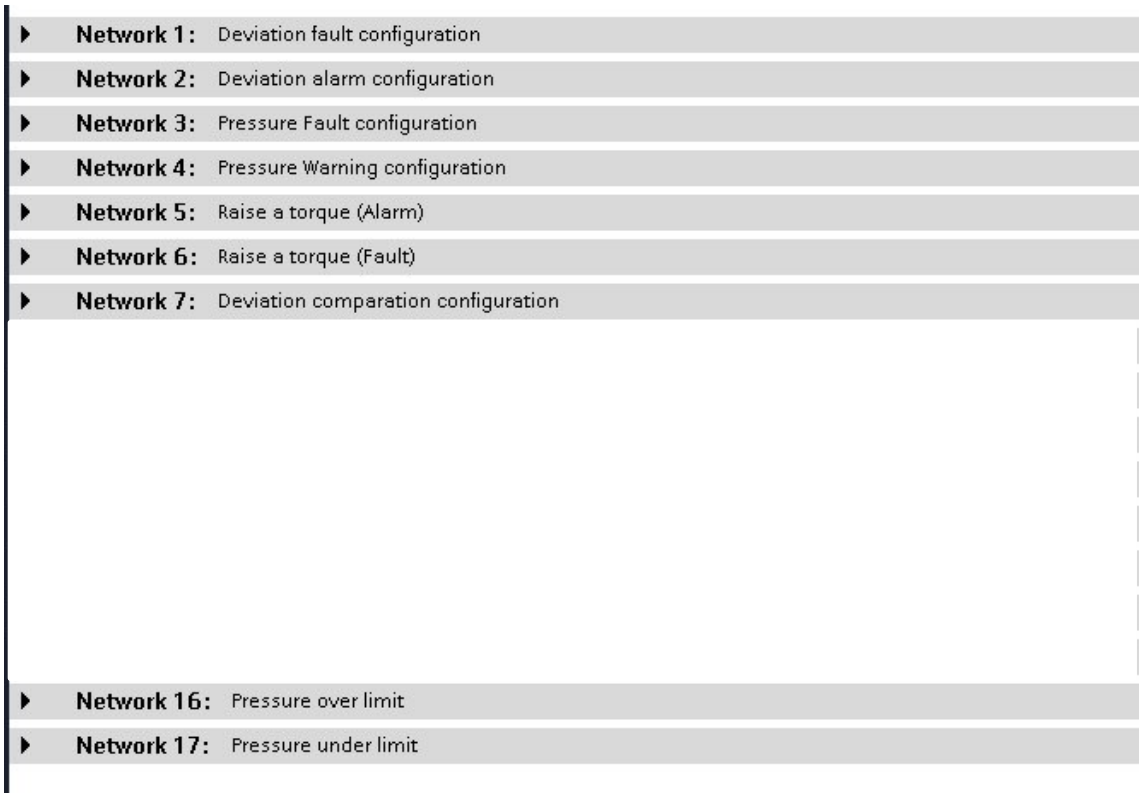
```
64
65  //IF not move command, close the proportional valve//
66 IF NOT #inMoveCMD1 THEN
67     #outVelocity1 := 13824;
68 END_IF;
```

Kuva 33. Venttiilin pakotus suljettuun asentoon.

7.3.8 Hälytyksien alustus

Ohjelmassa ei ole kenttälaitteita, esimerkiksi rajakytkintä, jotka aiheuttaisivat suoraan hälytyksen. Sen takia kaikki hälytykset tulee luoda ohjelmallisesti. Ohjelmalliset hälytykset ovat luonnollisesti alttiimpia ohjelmointivirheille, joten turvallisen ajon kannalta ne tulee testata täydellisesti. Ohjelmassa on varoituksia (Warnings) ja vikoja (Faults) konfiguroituna, joista hälytys ei estä ohjelman ajoa, sen sijaan vikatila lopettaa ohjelman ajon ja ohjelma menee vikatilaan. Hälytyksiä ohjelma antaa aseman liian suuresta poikkeamasta ohjearvoon nähden, paine sylinterissä ylittää raja-arvon ja vääntömomentti on liian suuri.

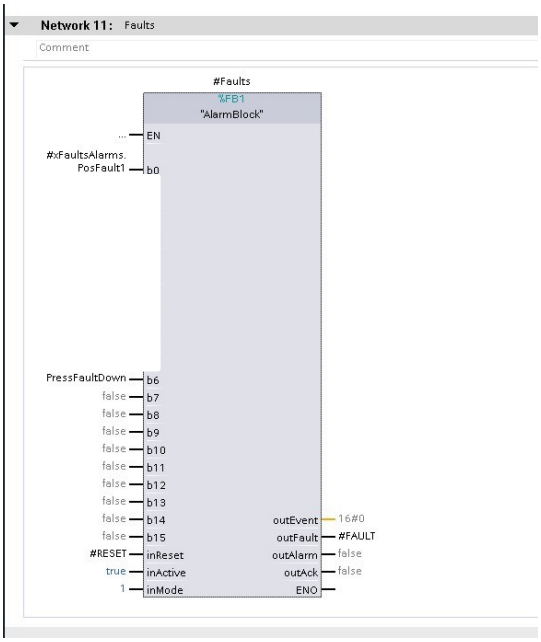
Aliohjelmassa on yksinkertaisia vertailufunktioita ja alustuksia. Suurimmassa osassa hälytyksiä, raja-arvot asetetaan HMI-paneelilta. Kuvassa 34 nähdään yhteenveto kaikista konfiguroiduista hälytyksistä.



Kuva 34, yleiskuva hälytyksien alustuksesta

7.3.9 Hälytykset ja vikatilat

Hälytyksiin ja vikatiloihin on luotu aliohjelma, joka palauttaa Word -tietotyypin arvon. Jokaiselle bitille on konfiguroitu oma hälytyksensä, joka näkyy HMI-paneelilla hälytyslokissa. Kuvasta 35 nähdään kaikki bitit, jotka aiheuttavat vian. ’RESET’ -muuttuja resetoi vikatilan, jos yksikään sisääntulo muuttuja ei ole tosi. Mode -sisääntuloportilla valitaan onko, kyseessä vika vai varoitus. Kuvassa näkyvä numero 1 asettaa aliohjelman tilaan, että se palauttaa vikoja ohjelman pääkiertoon ja HMI-paneelille. Vikatila siirtää muuttujan ’FAULT’ aktiiviseksi. Aliohjelma voidaan nopeasti asettaa pois päältä vaihtamalla epätosi, toden tilalle sisääntuloon #inActive Jos käyttöönotossa ohjelmassa on ongelmia, tämä voi olla tarpeellista.



Kuva 35. Vikalohko.

Vikatilalohkossa on käytetty tekniikkaa eng. overlaying, jonka avulla voidaan kirjoittaa yksittäinen bitti johonkin Word-tietotyypin. Tässä tapauksessa yksittäiset bitit kirjoitetaan muuttujaan #wFault. Kuvasta 36 nähdään, että ’wFault’ liitetään yksittäisiin bitteihin AT-funktion avulla. Samoin muuttuja ’bFault’ asettuu aktiiviseksi, joka estää laitteen ajon.

Kuva 36. Overlaying.

Kaikki viat näkyvät HMI-paneelilla tekstinä ja vilkkuvana hälytysvalona. Siksi kaikkien vikojen tulee erottua toisistaan ja ne tulee resetoida. Ilman resetointia vika on aktiivinen ja vikoja voi samanaikaisesti olla useita. Word -tietotyyppiä voidaan ajatella muistina, jossa on 16 erillistä muistipaikkaa. Jokaiselle vialle on varattu oma muistipaikkansa, joka liitetään HMI hälytyksiin. Kuvassa 37 nähdään kuinka HMI -hälytyksiin liitetään Fault, jonka trigger bit on 0.

ID	Name	Alarm text	Alarm class	Trigger tag	Trigge..
1	Discrete alarm_1	Fault in the position of cylinder 1	FAULT	FAULTS	0

Kuva 37. Discrete Alarm.

Ohjelmassa on käytetty yhtä ehtolausetta, jolla tarkistetaan jatkuvasti, onko uutta vikaa havaittu. Uusi vika ei korvaa vanhaa vikaa, vaan kaikki viat tallennetaan muuttujaan #LOG.wTotalFaults. Uusi vika tallennetaan, jos uusi vika on eri, kuin jo todettu vika.

```
//Store current fault in a static variable if new event is detected
#LOG.wNewEvent := #wFault;
//store previous fault in a static variable for the next cycle, in case of the fault disappears
IF #LOG.wNewEvent <> #LOG.wPrevFault THEN
    #LOG.wPrevFault := #LOG.wPrevFault OR #wFault;
END_IF;
//total faults
#LOG.wTotalFaults := #LOG.wPrevFault;
```

Kuva 38. Vikojen tallennus muuttujaan.

7.4 Visualisointi

Visualisointi HMI-näytöllä noudattaa pääosin Metson pohjaa. Visualisoinnissa on otettu mallia alkuperäisestä ohjelmasta, mutta sitä on muokattu Metson standardin mukaiseksi. Jokaiselle moodille on luotu oma näyttönsä, joista monitoroidaan ohjelman parametreja. Visualisoinnille Metson TIA Portal -pohjaan luodaan nappi, josta avautuu lisäosan etusivu. Visualisoinnissa on pyritty pitämään graafinen ilme selkeänä, jotta operaattorin on helppo ymmärtää käyttöliittymän toiminnot.

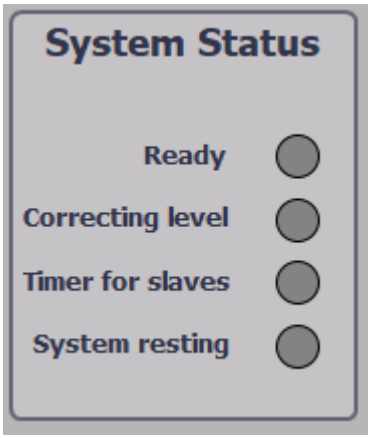
7.4.1 Etusivu

Lisäosan etusivua ei ole viimeistelty. Se tullaan viimeistelemään, kun ohjelma saadaan integroitua sakeuttimen pääohjelmaan. Etusivulta siis konfiguroidaan sylintereiden lukumäärä ja sylintereiden iskunpituus. Samoin valitaan, mitä moodia halutaan käyttää. Jokaiselta moodilta on palattava aina etusivulle vaihtamaan moodi. Etusivulta voidaan myös asettaa lisäosa täysin pois päältä, jolloin sakeuttimen pääohjelma ottaa täyden kontrollin sakeuttimen ajamisesta.

7.4.2 Moodien visualisointi

Jokainen moodi toimii omalla tavallaan, joten visualisoinnissa tuli ottaa huomioon, että tähän moodiin liittyvää tärkeä tieto välittyy operaattorille. Oman haasteensa loi se, että riittävä määrä tietoa tulee saada samanaikaisesti esitettyä Siemens TP700 Comfort Outdoor -näytöltä. Näytön koko on 7 tuumaa, joten teksti tai painonapit eivät voi olla liian pienellä fontilla. Jossain moodeissa on hyvä, että operaattori tietää, missä vaiheessa sekvenssiä sakeuttimen ajo on.

Operaattorin on hyvä tietää Automaattisessa moodissa ohjelman status. Kuvassa 39 nähdään onko, jokin sylinteri liikkeessä ja mikä vaihe sekvenssistä on menossa. Kaikki tieto, mitä kenttälaitteilta saadaan, esitetään myös HMI-näytöllä operaattorille.

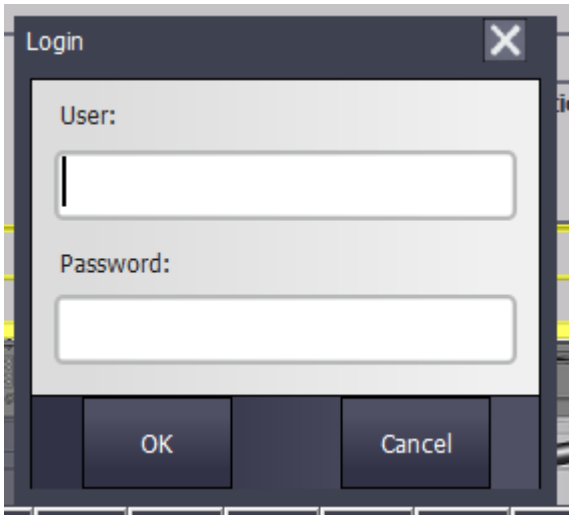


Kuva 39. Automaattiajon status.

7.4.3 Salattavat sivut

Osa lisäosan ominaisuuksista tulee piilottaa salasanan taakse. Esimerkiksi, jos kesken ajon muokattaisiin sylinterien iskunpituutta, laitteen turvaominaisuudet eivät toimisi oikein, joka saattaisi aiheuttaa laitteen rikkoutumisen. TIA Portalissa on valmiina ominaisuus, jolla voidaan luoda HMI-näytölle käyttäjä ja salasana, jotta joitain ponnahdusikkunoita ei voida avata ilman salasanaa. Kuvassa

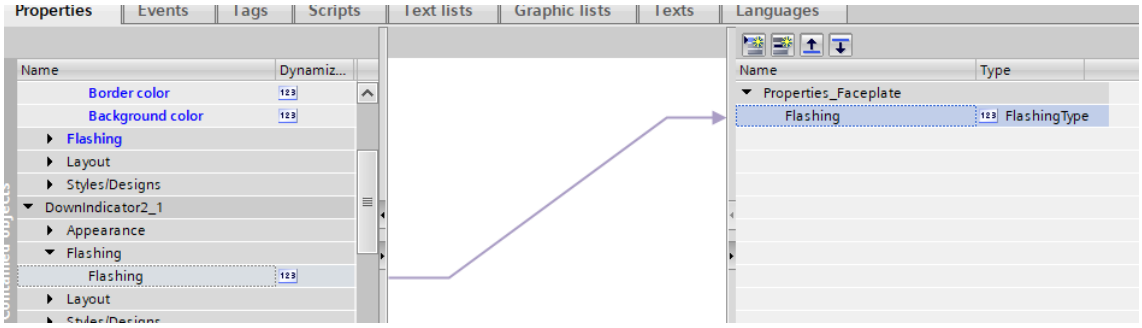
40 nähdään, mitä käy, kun pyritään avaamaan ponnahdusikkuna, joka on salasanasuojattu.



Kuva 40. Kirjautumissivu.

7.4.4 Faceplaten käyttö

Visualisoinnissa käytetään kirjasto-ominaisuutta. TIA Portalin kirjasto mahdollistaa visualisoinnissa kattavamman versiohallinnan. Kirjastossa voidaan luoda oma Faceplate, joka toimii eräänlaisena aliohjelmana. Samaa faceplatea voidaan monistaa ohjelmaan, mutta dynamization-lehdeltä asetetaan oikeat parametrit. Näin ollen visualisointi on nopempaa ja mahdolliset muutokset saadaan vietyä ohjelmaan kerralla.



Kuva 41. Faceplaten konfigurointi.

8 Yhteenveto

8.1 Pohdintaa lopputuloksesta

Opinnäytetyön tavoitteena oli saada ohjelma toimimaan TIA Portal -ohjelmointiympäristössä. Kesätyö Metsolla ohjelmoinnin parissa helpotti huomattavasti, kun ohjelma tuli luoda uudelleen lähes tyhjästä. Olen saanut tietyt ohjelmointirutiinit ja tavat, joka helpottaa suurten kokonaisuuksien hallintaa. Samoin laite, johon ohjelma ladataan tuli kesän aikana tutuksi. Opinnäytetyö rajattiin vain uuden ohjelman luomiseen toiminnankuvauksen mukaisesti, joten uudet muutokset eivät kuulu opinnäytetyön aiheeseen. Samoin integroiminen pääohjelmaan ei kuulu työn aiheeseen.

Tämä oli ensimmäinen suurempi logiikkaohjelmointikokonaisuus, jota pääsin ohjelmoimaan, joten ongelmiakin esiintyi suunnittelutyössä, jota etukäteen en osannut ottaa huomioon. Suurimpana ongelmana pidin sitä, että välillä jouduin pohtimaan, mikä olisi paras tapa toteuttaa jokin funktio. Välillä myöhemmin selvisi, että olisi olemassa parempikin tapa toteuttaa jokin funktio, joka aiheutti turhaa työtä jonkin verran, sillä funktiota tuli muokata jälkeenpäin. Toimin pääosin yksin kyseisen ohjelman suunnittelijana, joten ohjelmallisiin kysymyksiin ratkaisut tuli löytää itse, koska ohjelma ei ole tuttu Metson automaatioinsinööreille. Yleisiin ongelmiin ohjelmoinnissa sain vaivatta apua Metson automaatioinsinööreiltä.

Tavoitteet saavutettiin ja ohjelma toimii halutulla tavalla. Ohjelma toimii simuloinnissa täysin ja ohjelman toiminnallisuus saatiin valmiiksi aikataulussa. Simuloinnissa testattiin ohjelman toiminnallisuus, HMI-paneelin nappien toiminnot ja kaikki vikatilat.

8.2 Sakeuttimen pääohjelmaan yhdistäminen

Lisäosa tullaan tulevien kuukausien aikana yhdistämään sakeuttimen pääohjelmaan. Samoin tuotteen omistajalta on tullut lisää toivomuksia, kuinka laitteen tulisi toimia. Ohjelmaa myös siistitään ja ohjelmaa pyritään muokkaan sakeuttimen pääohjelman kaltaiseksi. Ohjelmaa tullaan kehittämään jatkuvasti käyttöönottoon asti ja käyttöönotossa tuskin ongelmilta vältytään.

Lähteet

Automazioni industriali. 2023. Verkkoaineisto. <https://ebattocchio.it/en/services/plc/#>. Luettu 20.12.2023.

Companiesmarketcap. Verkkoaineisto. Largest companies in Finland by market capitalization. <https://companiesmarketcap.com/finland/largest-companies-in-finland-by-market-cap/>. Luettu 22.11.2023.

Collins, Danielle. 2019. Why is the instruction list (IL) language for PLCs falling out of favor? Verkkoaineisto. Motioncontroltips. <https://www.motioncontroltips.com/why-is-the-instruction-list-il-language-for-plcs-falling-out-of-favor/>. 16.7.2019. Luettu 20.12.2023.

Computer Science Wiki. Verkkoaineisto. Architecture of the central processing unit (CPU). <https://computersciencewiki.org/index.php/Architecture_of_the_central_processing_unit_(CPU)>. Luettu 15.12.2023

Diemme Filtration. 2023. Verkkoaineisto. <https://www.diemmefiltration.com/sludge-thickener/>. Luettu 5.12.2023.

Dietrich, Shawn. 2023. Verkkoaineisto. Explaining User-Defined Data Types for PLC Programming.<https://control.com/technical-articles/explaining-user-defined-data-types-for-plc-programming/>. Luettu 18.12.2023.

Keinänen, Toimi; Kärkkäinen, Pentti; Lähetkangas, Markku & Sumujärvi, Matti. 2007. Automaatiojärjestelmien logiikat ja ohjaustekniikat. Helsinki: WSOY Oppimateriaalit Oy. Luettu 17.12.2023.

Kippo, Asko & Tikka, Aimo. 2008. Automaatiotekniikan perusteet. Helsinki: EDITA. Luettu 17.12.2023.

Knight, Ken. 2017. THICKENERS: How they operate. Verkkoaineisto. Mainland Machinery. <https://mainlandmachinery.com/thickeners-how-they-operate/>. Luettu 5.12.2023.

Lorig, Clarence & Gruner, Holger. 2006. Mineral processing. Verkkoaineisto. Britannica. <https://www.britannica.com/technology/mineral-processing>. Luettu 7.12.2023.

Mining technology. 2023. Verkkoaineisto. 2nd generation paste thickener for consistent, efficient dewatering performance. <https://www.mining-technology.com/contractors/data/nd-gen-paste-thickener/pressreleases/2nd-gen-paste-thickener/>. Luettu 25.11.2023.

Metso sakeutinesite. 2023. Sakeutinesite. Combining legacy and innovation HRT. Luettu 22.11.2023.

Metso. -a. Verkkoaineisto. Vastuullisuus. <https://www.metso.com/fi/yritys/vastuullisuus/>. Luettu 24.11.2023.

Metso. -b. Verkkoaineisto. Paste Thickener. <https://www.metso.com/portfolio/paste-thickener/>. Luettu 25.11.2023.

Metso. -c. Verkkoaineisto. Inclined Plate Settler. <https://www.metso.com/portfolio/ips-series/>. Luettu 27.11.2023.

Metso. -d. Verkkoaineisto. High Rate Thickeners. <https://www.metso.com/portfolio/high-rate-thickener/>. Luettu 28.11.2023.

Metso. -e. Verkkoaineisto. High Compression Thickener. <https://www.metso.com/portfolio/high-compression-thickener/>. Luettu 1.12.2023.

PLCopen. 2023. Verkkoaineisto. <https://plcopen.org/>. Luettu 10.12.2023.

Salama, Mahmoud. Overview of SIEMENS PLC – S7-1500, S7-1200, S7-400, S7-300. Verkkoaineisto. Inst Tools. <https://instrumentationtools.com/overview-of-siemens-plc/>- Luettu 21.12.2023.

Savona Equipment (nimimerkki). 2018. Verkkoaineisto. Thickeners — Types, Working Principle & Applications. Medium. <https://medium.com/@mari-anna_56839/thickeners-types-working-principle-applications-3d92a3725e8a>. Luettu 3.12.2023.

Siemens -a. 6ES7212-1AE40-0XB0 tuotesivu. <https://mall.industry.siemens.com/mall/en/WW/Catalog/Product/6ES7212-1AE40-0XB0>. Luettu 26.12.2023.

Siemens -b. 6ES7518-4AP00-0AB0 tuotesivu. <https://mall.industry.siemens.com/mall/en/WW/Catalog/Product/6ES7518-4AP00-0AB0>. Luettu 26.12.2023.

Siemens -c. SIMATIC ET 200SP – the compact I/O system for the control cabinet. Tuotesivu. <https://www.siemens.com/global/en/products/automation/systems/industrial/io-systems/et-200sp.html>. Luettu 28.12.2023.