jamk.fi

Peruspiirien parantaminen Siemens TIA Portal -ympäristössä kunnossapidon tarpeet huomioiden

Jani Ervalahti

Opinnäytetyöraportti
Elokuu 2018
Tekniikan ja liikenteen ala
Insinööri (AMK), sähkö- ja automaatiotekniikan tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Ervalahti, Jani	Opinnäytetyö, AMK	Joulukuu 2018
	Sivumäärä 30	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Peruspiirien parantaminen Siemens TIA Portal -ympäristössä kunnossapidon tarpeet huomioiden

Tutkinto-ohjelma

Insinööri (AMK), sähkö- ja automaatiotekniikan tutkinto-ohjelma

Työn ohjaaja(t)

Veli-Matti Häkkinen, Markku Ström

Toimeksiantaja(t)

JEEC Oy

Tiivistelmä

Automaatiosuunnittelun sovellussuunnittelu on yksi suurimpia työvaiheita. Ohjelmoinnissa voidaan säästää aikaa luomalla valmiita pohjia toistuvista asioista. Näitä ovat esimerkiksi ohjelmointipohjat mittauksien skaalaamiseen, vaadittavat laskukaavat tai moottorien ohjaukseen luodut lohkot. Ohjelmointipohjien ajan tasalla pitäminen on tärkeää, ja siksi niitä tulisi päivittää aika ajoin.

Tavoitteena oli parantaa Siemens TIA Portal -ympäristöön luotua PLC-ohjelmointipohjaa, jossa otettaisiin huomioon kunnossapidon näkökulma. Ohjelmointipohjan tavoitteena on yrityksen näkökulmasta säästää aikaa PLC-ohjelmoinnissa ja asiakkaiden päässä selkeyttää näyttökuvien käytettävyyttä.

Työn tuloksena on neljä paranneltua peruspiiriä, joissa on huomiotu kunnossapitotarpeet. Ohjelmointipohja on valmis käytettäväksi sellaisenaan, ja siinä on korjattu kaikki alkuperäisistä lohkoista havaitut toiminnallisuuteen vaikuttavat virheet. Piirit ovat kuitenkin muokattavissa projektista riippuen, ja niitä voidaan muutella asiakkaan tarpeen mukaan. Käyttöohjeita piirien käytölle ei tarvinnut tehdä, koska toimeksiantaja ei kokenut niitä tarvitsevansa.

Ohjelmointipohjan hyöty tulee esiin ohjelmoinnissa ja käyttöönotossa nopeuttavana tekijänä. Pohjaan tehtyjen muutosten vaikutukset näkyvät jopa operaattorille asti. Kunnossapitoa hyödyntämään tehdyt ominaisuudet auttavat operaattoria ennakoimaan laitteiden huoltoajankohdat ja seuraamaan laitteiden käyttöaikoja realistisemmin. Opinnäytetyö on jatkoa toimeksiantajan aiemmille opinnäytetöille.

Avainsanat (asiasanat)

TIA Portal -ohjelmointityökalu, ohjelmoitava logiikka, ohjelmointi

Liitteenä neljä kappaletta kuvia piirinäytöistä, 4 sivua. Liitteet 1-4 ovat salassa pidettäviä, joten ne on poistettu julkisesta työstä. Salassapidon peruste Julkisuuslain 621/1999 24§, kohta 17, yrityksen liike- tai ammattisalaisuus. Salassapitoaika viisi (5) vuotta, salassapito päättyy 4.12.2023

jamk.fi

Description

Author(s)	Type of publication	Date
Ervalahti, Jani	Bachelor's thesis	December 2018
		Language of publication: Finnish
	Number of pages: 30	Permission for web publication: x

Title of publication

Siemens TIA Portal blocks upgrading with adding maintenance perspective

Degree programme

Electrical and Automation Engineering

Supervisor(s)

Häkkinen Veli-Matti, Strömm Markku

Assigned by

JEEC Oy

Abstract

Application designing is one of the largest parts of automation system design. Creating templates, it is possible save huge amount on programming of repeatable actions. These are example programming templates for scaling measurements, possible function blocks and motor controls. Updating those templates is important part of application designing.

The goal was to upgrade PLC programming template for Siemens TIA Portal environment, where maintenance perspective would be included. For client goal was to create template which save time on PLC-programming and for client create HMI-screens easier to operate.

Thesis result was four upgraded function blocks where maintenance needs were included. Programming template is ready for use as it is now, and all previously noticed problems are solved. Blocks are still editable depend on current project and customer needs. Manuals for blocks was not made because client didn't see need for those.

Benefit of programming template is speeding-up programming and commissioning. Changes to template influence even operator. Programming template changes help operator predict maintenance needs and devices operating times. Thesis is continuation of previous client thesis.

Keywords/tags (subjects)

TIA-Portal software, programmable logic controller, programming

In appendixes there are four photos of HMI-screens, 4 pages. Appendixes 1-4 are confidential which have been removed from the public thesis. Grounds for secrecy: Act on the Openness of Government Activities 621/1999, Section 24, 17: business or professional secret. Period of secrecy is five years and it ends 4.12.2023.

Sisältö

Kuviot

1 Johdanto

1.1 Opinnäytetyön tausta

Työelämässä pyritään monessa tapauksessa järjestelmälliseen työtulokseen ja tätä myötä myös optimoimaan työvaihden toteutukset. Järjestelmällisyys auttaa virheiden minimoinnissa, ja työvaiheiden ennalta valmiiksi tekeminen puolestaan puhtaasti työtehtävissä säästyvässä työajassa. Tämän työn tarkoituksena oli parantaa molempia.

Opinnäytetyön tarkoitus oli kehittää projektipohjaa Siemens TIA Portal -ympäristössä toteutettaviin projekteihin. Kattavalla projektipohjalla monet työvaiheet jäävät kokonaan pois suunnittelijan harteilta projektia tehdessä, joten tämä lisää järjestelmällisyyttä ja säästää valtavasti aikaa. Projektipohjan toiminta perustuu siihen, että tietyt tiedot ovat miltei samanlaisia projektista riippumatta. Projektipohjaa voisi kuvastaa jäätelövohveli, joka on aina sama, mutta siihen valkattava jäätelö on verrattavissa projektikohtaisiin tietoihin.

Toimeksiantaja JEEC Oy on kehitellyt vuosien varrella hurjaa kyytiä logiikkaohjelmointiaan. Yrityksellä on lukuisia pohjia tehty moneen eri tarkoitukseen ohjelmoinnin nopeuttamiseksi. Pohjia löytyy sekä uusia, että vanhoja, joita päivitellään tasaisin väliajoin tarpeen mukaan.

1.2 Opinnäytetyön tehtävä ja tavoitteet

Tässä opinnäytetyössä tehtävänä oli koota aiemmista ohjelmalohkoista valmis ohjelmointipohja Siemens TIA Portal -ympäristöön ja parannella sitä toimeksiantajan toiveiden mukaan. Ohjelmointipohjan tarkoitus on helpottaa ja nopeuttaa tulevien projektien ohjelmointiosuuksia. Ohjelmalohkojen parantamisella tavoitellaan kunnossapidolle tarpeellisia ominaisuuksia, mutta samaiset ominaisuudet eli laskurit mahdollistavat vaihtoautomaation toteuttamisen.

Opinnäytetyöhön tietoperustaa hankin kirjajulkaisuista, internet lähteistä, valmistajien sivuilta ja muista opinnäytetöistä.

Opinnäytetyön tavoitteena on nopeuttaa entisestään Siemens TIA Portaalilla tehtävien projektien ohjelmointiosuuksia, mutta myös täyttää asiakkaiden toivomien ominaisuuksien toteutumista. Toisena tavoitteena on helpottaa ja selkeyttää piirinäyttöjen käytettävyyttä.

Henkilökohtaisena tavoitteena opinnäytetyötä tehdessä oli oppia työskentelemään itsenäisemmin ja oppia tuntemaan suunnittelutoimiston käytänteet. Nämä edistäisivät tulevissa projekteissa itsenäiseen työskentelyyn kannustamista ja antaisi itseluottamusta asiantuntijana toimimisessa.

Opinnäytetyö oli kehitystutkimus, jonka tuloksia vertailtiin ja analysoitiin uudistetun ja alkuperäisen toimintatapojen välillä

1.3 JEEC Oy

Toimeksiantajana opinnäytetyössä toimi JEEC Oy. JEEC Oy on perustettu vuonna 2009 ja toimii Jyväskylässä. Yritys tarjoaa korkealaatuisia automaatio- ja sähköalojen suunnittelu- ja konsultointipalveluja. Yrityksen pääasiallinen ydinosaaminen keskittyy automaatiopuolella prosessiautomaatiojärjestelmiin ja käyttöönottoihin. (JEEC Oy, 2018)

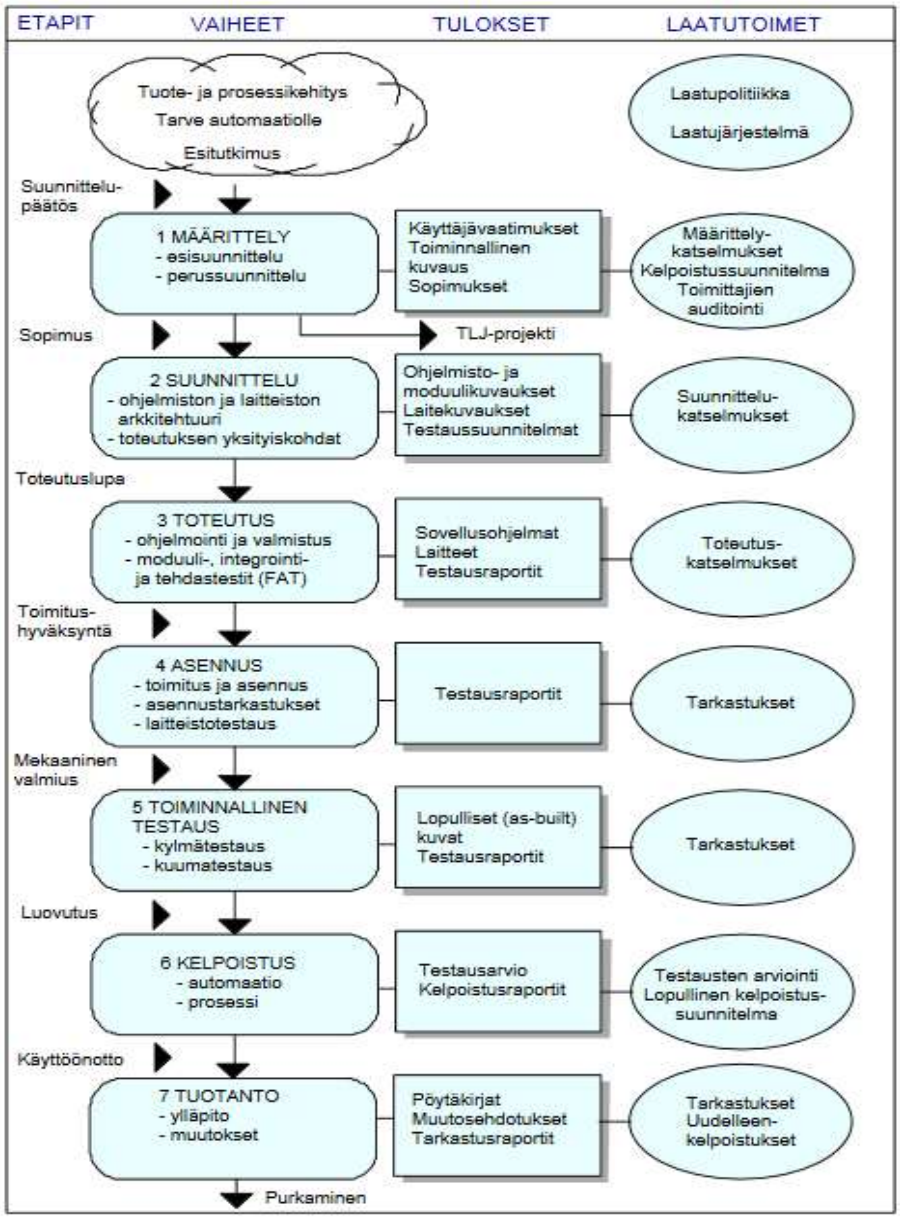
Yrityksessä työskentelee tällä hetkellä 17 vakituista työntekijää, joista kaksi sähkö- ja instrumentointisuunnittelijoita ja loput automaatiosuunnittelijoita. Yrityksen toiminta perustuu avainhenkilöiden pitkäaikaiseen ja monipuoliseen toimintaan. Yrityksen arvot perustuvat laatuun osaamiseen, asiakaslähtöisyyteen, luottamukseen ja jatkuvaan parantamisen periaatteiden mukaiseen toimintaan. (JEEC Oy, 2018)

Opinnäytetyössä käytettävät laitteistot ja ohjelmistot tulivat toimeksiantajalta. Työskentelytilana toimi työpiste toimeksiantajan toimistotiloista.

2 Automaatiosuunnittelun standardointi

2.1 Sovellussuunnittelu automaatioprojektissa

Automaatioprojektin eri vaiheita voidaan kuvailla elinkaarilla (ks Kuvio 1). Sovellussuunnittelun osuus elinkaarimallista toteutetaan suunnittelu- ja toteusvaiheissa. Sovellussuunnittelu pitää sisällään mm. ohjausohjelmiston rungon, käyttöliittymäsuunnittelun, valvomon ja rajapintojen välisien yhteyksien suunnittelemisen. (Suomen Automaatioseura Ry 2001, 50.)



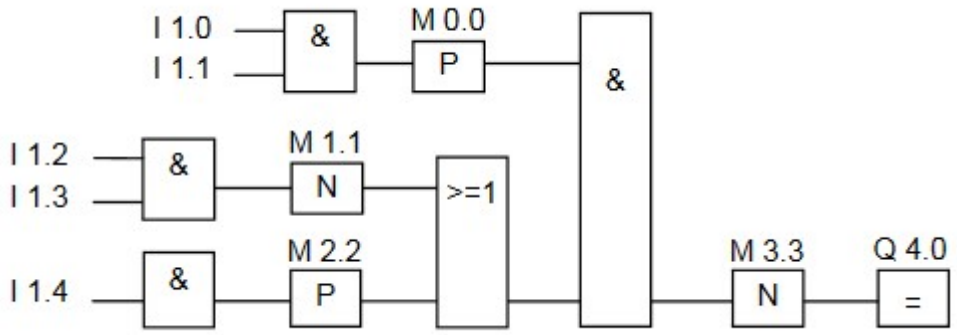
Kuvio 1. Automaatiosuunnittelun elinkaarimalli (Suomen Automaatioseura 2001, 17)

2.2 Ohjelmointikielet automaatiosovellussuunnittelussa

TIA-Portaali pohjautuu SFS-EN 61131-3 automaatiosuunnittelustandardiin, joka määrittelee ohjelmoitavien logiikoiden käytettävissä olevat ohjelmointikielet tekstipohjaisina ja graafisina esitysmuotoina. Standardi on tarkoitettu ohjelmistovalmistajille, jotka nimenomaan valmistavat ohjelmointikieliä omiin ohjausjärjestelmiinsä. Standardissa käsiteltyjä ohjelmointikieliä ovat LAD (Ladder Diagram), ST(Structure Text), IL (Instruction List), SFC (Sequential Function Chart) ja FBD (Function Block Diagram), joka on mainituista ohjelmointikielistä tavanomaisin. (SFS 175-2, 2006, 132-152.)

2.2.1 Function block diagram

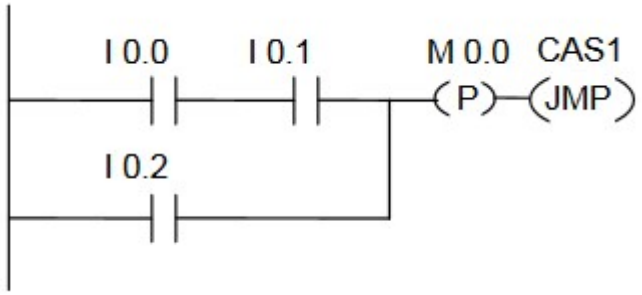
Function block diagram on tämän hetken käytetyin ohjelmointikieli. Kyseinen ohjelmointikieli koostuu ulkomuodoltaan erikokoisista suorakaiteen muotoisista toiminnoista, joiden toiminta pohjautuu niiden sisällä olevien symbolien merkitykseen. FBD-kielen etuna ilmenee sen selkeys ja tiivis havannoillistaminen (ks Kuvio 2).



Kuvio 2. Esimerkki FBD-kielestä (Function Block Diagram (FBD) for S7-300 and S7-400 Programming, 29.)

2.2.2 Ladder diagram

Ladder diagram muistuttaa tikapuumaista ohjelmointikielirakennetta. Tämä on toinen grafiikkapohjainen ohjelmointikieli IEC 61131-3 standardista. Kyseinen ohjelmointikieli on varsin pätevä työkalu yksinkertaisten sovellusten toteutukseen. Sitä on helppo lukea, joten myöhemmät muokkaukset koodiin onnistuu vaivattomasti (ks. Kuvio 3.)



Kuvio 3. Esimerkki Ladder Diagram -koodista (Ladder Logic (LAD) for S7-300 and S7-400 Programming, 29.)

2.2.3 Structured text

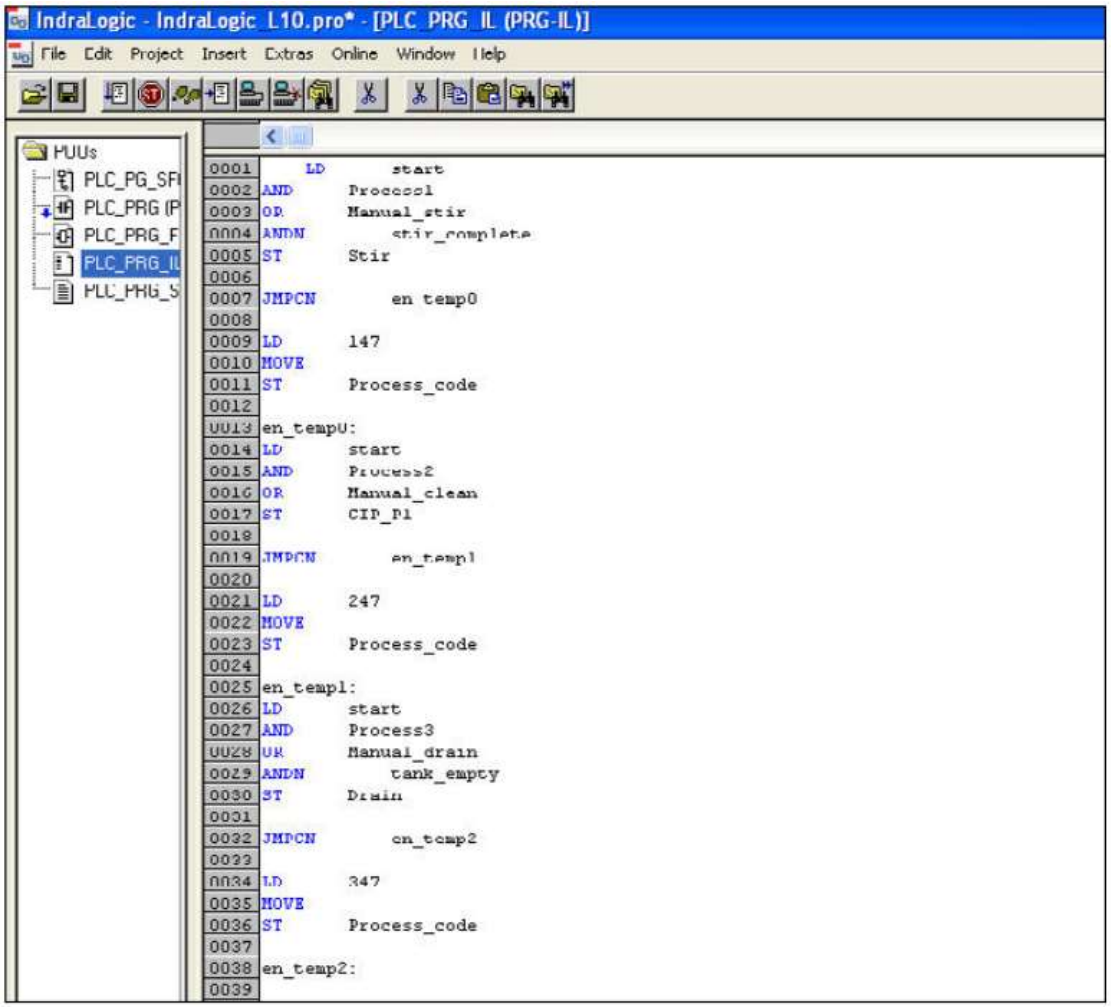
Structured text on tekstipohjainen ohjelmointikieli. ST-ohjelmointikieli vaikuttaa pitkälti Pascal- tai C-ohjelmointikieleltä. Ohjelmointikieli muodostuu lauseista, jotka erotetaan toisistaan puolipilkuilla (ks. Kuvio 4).

```
CMD := AUTO_CMD & AUTO_MODE
          OR MAN_CMD & NOT MAN_CMD_CHK & NOT AUTO_MODE ;
CMD_TMR (IN := CMD, PT := T_CMD_MAX);
ALRM_FF (S1 := CMD_TMR.Q & NOT FDBK, R := ACK);
ALRM := ALRM_FF.Q1;
```

Kuvio 4. Esimerkki ST-ohjelmointikielen koodista (Saari 2015, 8.)

2.2.4 Instruction list

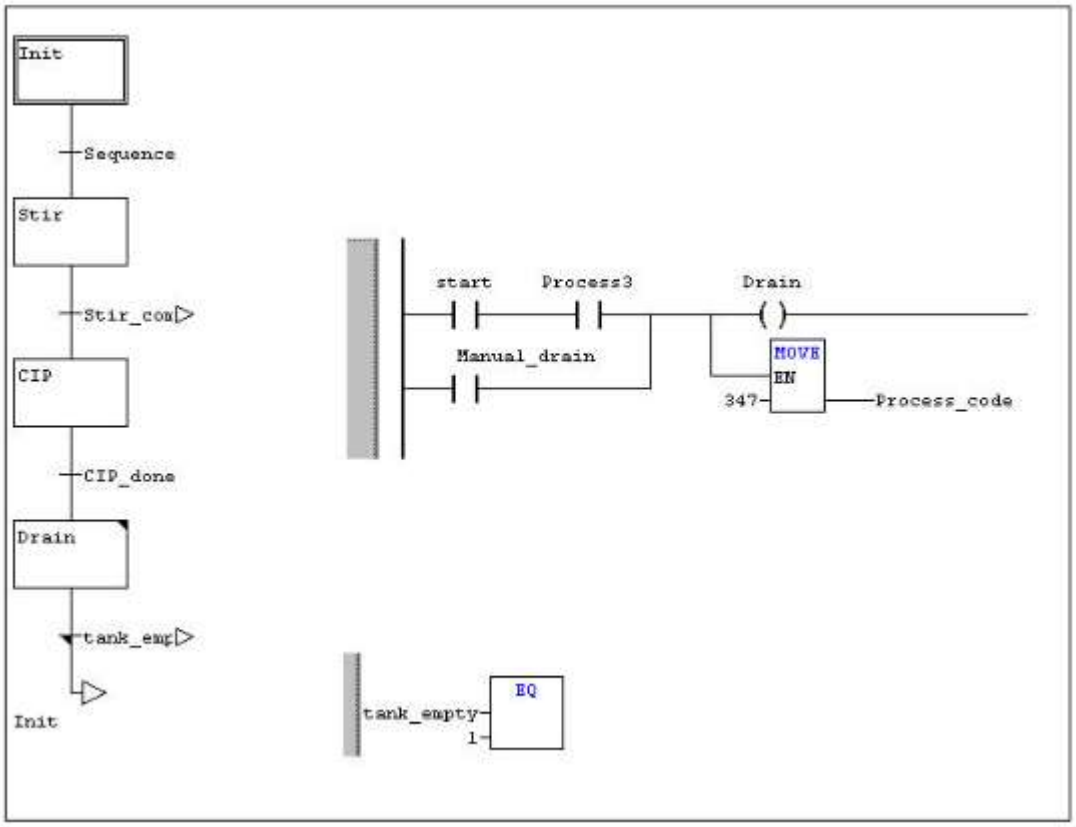
Instruction list eli suomeksi ohjelista on nimensä mukaan lista ohjeita. Il-ohjelmointikieli tekstipohjainen ja se vastaa hieman sekvenssin toimintaperiateetta. Instruction List -ohjelmointikielen kanssa on samankaltainen Assembler-ohjelmointikieli. Tämä ohjelmointikieli sisältää useita rivejä koodia, siten että jokaisella rivillä on yksi komento (ks. Kuvio 5). Lisäykseksi, jos ohjelmaa koodatessa tällä kielelellä noudatetaan IEC-standardia, niin ohjelman siirtely alustalta toiselle on tehty erittäin helpoksi (Rexroth Bosch Group 2009.)



Kuvio 5. Esimerkki Instruction list -ohjelmointikielestä (Rexroth 2009.)

2.2.5 Sequentil function chart

Sequential function chart on luettavuudeltaan simppeliä. Ohjelmointikieli etenee ylhäältä alaspäin, ensin on mainittu tehtävä ja sitten kysely, että onko kyseisen tehtävän tarvittavat asiat tehty (ks. Kuvio 6). Tehtävän toteutuksen jälkeen siirrytään eteenpäin. Tämä helpottaa valtavasti käyttäjää ongelmatilanteiden ratkaisussa, koska operaattori näkee missä vaiheessa prosessia annetut tehtävät ei täyty. Huonona puolena ohjelmointikielessä on se, että sitä ei ole käännettävissä suoraan toiseksi standardiksi automaatio-ohjelmointikieleksi. SFC ei sovi muutenkaan ihan kaikkiin käyttökohteisiin, mutta suurien kokonaisuuksien hahmottamiseen ja hallintaan se on varsin pätevä (Rexroth 2009.)



Kuvio 6. Esimerkki SFC-ohjelmointikielestä (Rexroth 2009.)

2.3 Toimilohkot

Toimilohko on valmis logiikkasovelluksen osa, joka suorittaa tietyn toiminnon, esimerkiksi moottorilähdön ohjaamisen ja valvonnan tai säätöalgoritmin (Rajala 2010, 13.)

3 Siemens logiikat

3.1 S7-1200-Sarja

Simatic S7-1200 on ohjelmoitava PLC-logiikka (Programmable Logic Controller). Se on tarkoitettu mekaniikan ohjaustehtäviin ja on suorituskykyynsä nähden varsin pienikokoinen fyysisiltä mitoiltaan. Tänä päivänä monet erilaiset automaatio-ohjausta vaativat toiminnot ovat toteuttu PLC-pohjaisilla ratkaisuilla. (Helppoa automaatio-ohjelmointia S7-1200-logiikalla N.d.)

Kuvio 7. Siemens S7-1200-sarjan PLC

3.2 S7-1500-Sarja

Simatic S7-1500 -logiikkaohjain on uusi innovaatio automaatiolle. Siihen on lisätty valtavasti ominaisuuksia edeltäjiinsä nähden. Logiikka vastaa useamman aiemman

sukupolven eri logiikoiden, lisälaitteiden ja ohjelmistojen yhdistelmää. Myös aiemmasta mallista poiketen, tämän hetkisessä lippulaivamallissa on erillinen näyttö, joka nopeuttaa huomattavasti käyttöönottoa ja vikatilanteiden ratkaisua. (Tehokasta automaatio-ohjelmointia S7-1500-logiikalla N.d.)

Kuvio 8. Siemens S7-1500 tuoteperheen eri logiikoita

3.3 Siemens TIA Portal -ohjelmointityökalu

TIA Portal on ohjelmointityökalu, joka on tarkoitettu nimenomaan PLC-ohjelmointiin. Sana TIA (Totally Integrated Automation) on lyhenne ja suomeksi tarkoittaa täysin intergroitua automaatiota. TIA Portal -ohjelmointityökalussa on panostettu selkeästi käytettävyyten ja helppokäyttöisyyteen verraten aiempiin Siemensin ohjelmistoihin. Se mikä tekee TIA Portalista tällä hetkellä muita työkaluja edistyksellisemmän on se, että kaikki ohjelmointi ja suunnittelu tapahtuvat yhdellä ohjelmalla ohjelmakoodista HMI-näyttökuvien tekoon. (TIA Portal (Simatic STEP 7) N.d.)

3.4 Siemens näyttöpaneelit

Monien prosessien hallinnan helpottamiseksi niille on monesti luotu käyttöliittymä. Käyttöliittymä on prosessin hallintaan suunniteltu kommunikointi välinen prosessin ja operaattorin välillä. Useimissa tapauksissa ohjelma on toteuttettu visuaalisesti ja

se suunnitellaan Siemensin tapauksessa WinCC-ohjelmalla. Visuaalisesti selkeä ja havainnollinen käyttöliittymä helpottaa sen käyttäjää huomattavasti. Suunniteltu ohjelma ladataan näyttöpaneelin, josta sitä voidaan operoida (ks. Kuvio 9). Näyttöpaneeleja on kahta eri tyyppiä. Siemenssillä on sekä napillisia, että kosketusnäytöllä varustettuja operointipaneeleja. (SIMATIC HMI -kosketusnäytöt N.d.)

Kuvio 9. Esimerkki napillisesta ja kosketusnäytöllä varustetusta näyttöpaneelista

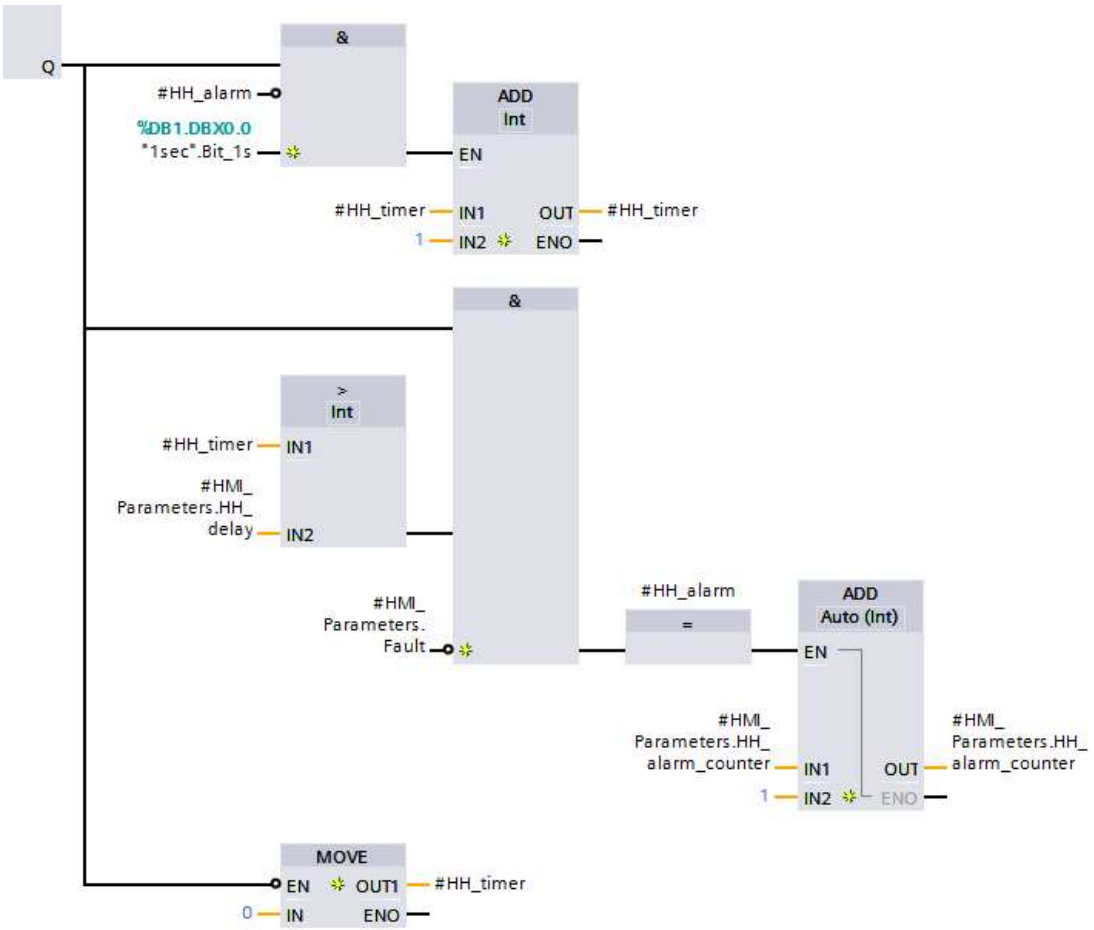
4 Ohjelmalohkotyypit

Tässä opinnäytetyössä haluttiin jokaiseen ohjelmalohkotyyppiin lisäyksiä kunnossapidon näkökulmaa ajatellen. Opinnäytetyön lisäyksillä tavoiteltiin operaattoreille parempia prosessin tarkkailumahdollisuuksista kunnossapidon näkökulmaa tarkastellen. Kunnossapitoa varten lisättävät laskurit ja toiminnot ohjelmalohkoihin mahdollistavat myös vaihtoautomaation toteuttamisen.

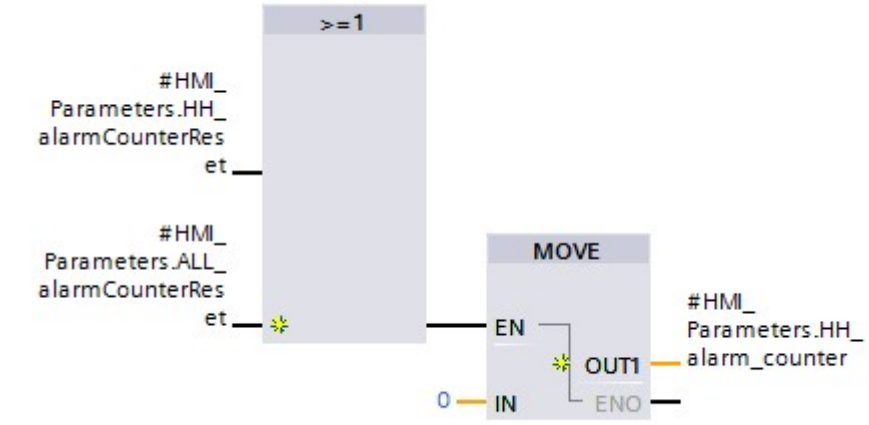
4.1 Mittauslohko

Mittauslohkon tehtävä on vastaanottaa ja prosessoida signaaleja antureilta. Antureilta tulevat signaalit skaalataan lohkossa ymmärrettäviin yksiköihin. Skaalauksen jälkeen arvot ovat vertailukelpoisia määrättyihin rajoihin. Monessa prosesissa kukin anturilta tuleva mittaus on asetettu rajoihin, jossa sen olisi tarkoitus normaalitilanteessa pysyä. Näin ei kuitenkaan aina ole, vaan saattaa käydä niin, että syystä tai toisesta prosessin arvot ovat muuta kuin niiden toivottaisiin olevan esimerkiksi laiterikon tai muun syyn johdosta.

Tämän opinnäytetyön tarkoituksena oli mittauslohkon osalta lisäillä hälytysrajoille laskurit. Laskureista esimerkkinä toimii ylempi ylärajahälytyslaskuri HH-alarms -laskuri (ks. Kuvio 10). Laskurin resetointi eli nollaus on toteutettu kaikissa lohkoissa täysin samalla tavalla (ks. Kuvio 11). Laskureista haluttiin operaattorille mahdollisimman selkeät, jotta niistä olisi käytännön tasollakin hyötyä. Laskurien ideana tässä työssä oli selvittää, kuinka monta kertaa mittausarvot ylittävät hälytysrajat. Tällä pyrittiin helpottamaan mahdollisen vian löytämistä. Esmerkiksi, jos alempi ylähälytysraja H-alarms laskuri olisi selkeästi muita laskureita suurempi, niin pystyttäisiin heti rajaamaan mistä vikaa voisi lähteä etsimään (ks. Liite 1).



Kuvio 10. Mittauspiirin ylempi ylähälytysrajan toteutus koodissa



Kuvio 11. HH-alarms eli ylemmän ylähälytysrajan nollaus koodissa.

Haluttujen lisäyksien jälkeen haluttiin kehittää vielä käytettävyysominaisuuksia, jos se vain olisi mahdollista. Kyseinen mittauspiiri pääsi prototyypiksi. Peruspiirit haluttiin päänäkymään Popup-ikkunoiksi, jotta prosessia olisi helpompi seurata kokonaisuutena samanaikaisesti. Ongelmaksi tuli se, että vaikka TIA Portalissa oli valmiit Popup-ikkunaohjelmointimahdollisuudet, ne olivat erittäin kömpelöitä käyttää. Popup-ikkunoissa ei ollut mahdollisuutta liikutella niitä vapaasti näytöllä vaan ne ilmeistyivät tiettyyn ennalta määrättyyn kohtaan. Tähän haluttiin keksiä toisenlaista toteutustapaa. Tähän löytyikin muutama tutkimisen arvoinen kehitysidea, mutta ajanpuutteen vuoksi päätin toteuttaa opinnäytetyön aikataulussa ja selvitellä paremmalla ajalla löytyisikö ongelmaan järkevää ratkaisua.

4.2 Säätölohko

Säätölohkon tehtävä on toimia säätimenä. Säädin on toteuttu samalla tavalla kuin muutkin tyypilliset säätimet. Säädin tarkastelee kyseiseen säätimeen asetetun mittauksen arvoa ja pyrkii ohjausalgoritmillaan ohjaamaan prosessia mahdollisimman nopeasti asetusarvoon ja pysymään siinä mahdollisimman stabiilisti.

Säätöpiiriin muutoksia tuli eniten. Säätöpiirissä korjailtavaa riitti DataBlockin muuttujissa, HMI-tag luettelossa ja itse piirinäytön piirtämisessä. Kaiken perusasioiden kuntoon laatimisen jälkeen, aloitin vasta itse varsinaisen opinnäytetyön tekemisen. Säätöpiiriin haluttiin pääasiallisesti vain mittapoikkeaman hälytyksestä tieto ja siitä laskuri. Toteuttamani mittapoikkeamalaskuriin haluttiin On/Off –toiminto, jotta sen saisi pois päältä tarvittaessa. Tällä annettaisiin mahdollisuus ajaa prosessia, vaikka hälytys olisikin aktiivinen. Laskuriin olisi voinut hyvin lisätä ominaisuuden määritellä käyttöliittymästä kuinka suuri säätöpoikkeama saa olla, mutta toimeksiantaja oli jo tyytyväinen tähän ratkaisuun (ks. Liite 2).

4.3 Venttiililohko

Venttiililohkon toiminta rajoittuu On/Off-venttiileihin. Piirin idea on huomattavasti yksinkertaisempi mitä se olisi säätöventtiilin kohdalla. Venttiilipiirin tehtävänä on siis vain availla ja sulkea venttiiliä laitetasolla, mutta lohkolla tehdään paljon muutakin.

Piiriin on upotettu valtava määrä informaatiota. Viimeisimpänä lisäyksenä minun tehtäväni oli kehittää piirille kunnossapitoa ajatellen hyödyllisiä lisäominaisuuksia.

Työtovereideni haastattelun jälkeen selvisi, että asiakkailla on ollut jo pitkään toiveena saada mahdollisuus seurata venttiilien käyttöä. Tämä helpottaa puolestaan heitä ennakoimaan huolto- tai jopa vaihtotarpeet. Lisäsin siis laskurin avauskerroille ja käyttöajalle. Käyttöaika mittaa venttiilin aukiolo aikaa. Tämä antaa mahdollisuuden selvittää huoltoajankohdat (ks. Liite 3).

4.4 Moottorilohko

Moottorilohkon tehtävä on viestittää taajuusmuuttajalle tarvittavat tiedot ja lukea taajuusmuuttajalta tarvittavat tiedot halutulta moottorilta. Moottorille tapahtuvat ohjaustiedot ovat päälle tai pois päältä ja kierrosnopeus. Taajuusmuuttajalta luettavat tiedot voivat olla paljon kattavampi kokonaisuus mitä moottoripiirin HMI-piirinäyttökuvassa on tuotu esiin (ks. Liite 4).

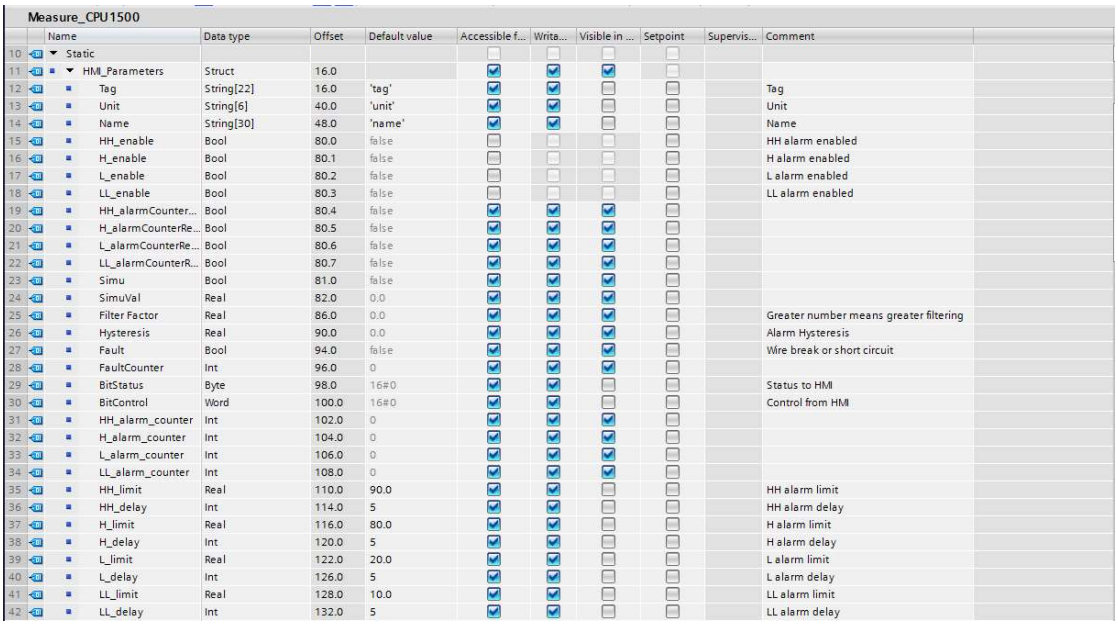
Tässä opinnäytetyössä keskityttiin miettimään mitkä ominaisuuslisäykset parantaisivat kunnossapidon näkökulmaa. Tilanne oli sama mitä venttiililohkossa. Työkavereita haastatellessani tulin siihen tulokseen, että samaiset käynnistyskertojen ja päälläolo-laskurit haluttiin lisäykseksi siihen mitä oli valmiina. Nämä ominaisuudet auttasivat huoltotoimenpiteiden ennakoinnissa, esimerkiksi moottoreiden huollot tiettyjen käyttötuntien jälkeen. Näistä ominaisuuksista olisi myös hyötyä vaihtoautomaation toteutuksessa. Monesti prosesseissa on kahdennettu laitteet, jotta huoltotilanteissa ei tarvitse seisauttaa prosessia täysin. Näissä tilanteissa toinen moottori voi olla päällä kun toista huolletaan. Moottoreita halutaan myös usein käyttää tasaisesti, jotta ne eivät jämähtäisi käytön puutteesta. Useissa tapauksissa moottoreiden käyttöä vaihdellaan esimerkiksi sadan käyttötunnin välein.

5 Opinnäytetyön toteutus

Opinnäytetyön toteutus lähti liikkeelle tutustumalla Siemens TIA Portal -ohjelmistoon tehtyihin valmiisiin lohkoihin, joita työssä oli tarkoitus korjailla ja parannella ominaisuuksia lisäilemällä toimeksiantajan antamien kriteerien merkeissä. Toimeksiantaja oli etukäteen määritellyt opinnäytetyössä kehitettävät ohjelmalohkot. Ohjelmalohkotyyppejä oli 4 erilaista (mittaus, säätö, venttiili ja moottori). Toimeksiantaja antoi listauksen lisättävistä ominaisuuksista toimeksiannon yhteydessä. Pääasialliset ominaisuudet, joita työnantaja halusi lisättävän lohkoihin, olivat kaiken näköiset laskurit. Laskureiden lisäämisellä lohkoihin lähdettiin hakemaan huoltotoimenpiteiden ennakointia ja vaihtoautomaation mahdollistamista ilman operaattoria. Lisäyksien lisäksi toimeksiantaja toivoi, että lohkoissa ilmenneet ristiriidat ohjelmalohkon ja HMI-piirinäyttöjen välillä myös korjattaisiin. Näitä olivat esimerkiksi HMI-näyttökuvissa olevien muuttujien väärät tagit eli tunnisteet jollekin muuttujalle.

Työ lähti liikkeelle jo valmiina olevien lohkojen toiminnan selvittämisestä. Selvittämisessä kävin läpi lohkot vilkuillen, millä tavalla mitkäkin valmiit toiminnot lohkoihin oli toteutettu. Tähän kului aikaa suhteellisen paljon, sillä itse työskentelyalustan käyttö vaati alussa omaa opetteluaan. Selvitettyäni pääpiirtein ohjelmalohkojen toiminnan pääsin suunnitteluvaiheeseen.

Päädyttyäni järkevään ratkaisuun toteutuksesta, siirryin toteuttamaan ohjelmalohkoihin haluttuja lisäyksiä. Lisäykset pyrin toteuttamaan samalla tyylillä, kuin ohjelmalohkoissa valmiina olleet ominaisuudet. Lisäykset aloitin ohjelmalohkojen koodia muokkaamalla. Koodia muokatessa lisäilin tarvittavat muuttujat datablockeihin. Ohjelmakoodin valmistuttua järjestelin lohkon muuttujat omaan struct rakenteeseen offsetin eli osoiteavaruuden alkupäähän, jotta näyttökuvien tageihin liitetyt offset arvot pysyisivät helpommin hallittavassa järjestyksessä lohkoa myöhemmin päiviteltäessä tai tulevista projekteista riippumatta (ks. Kuvio 11).

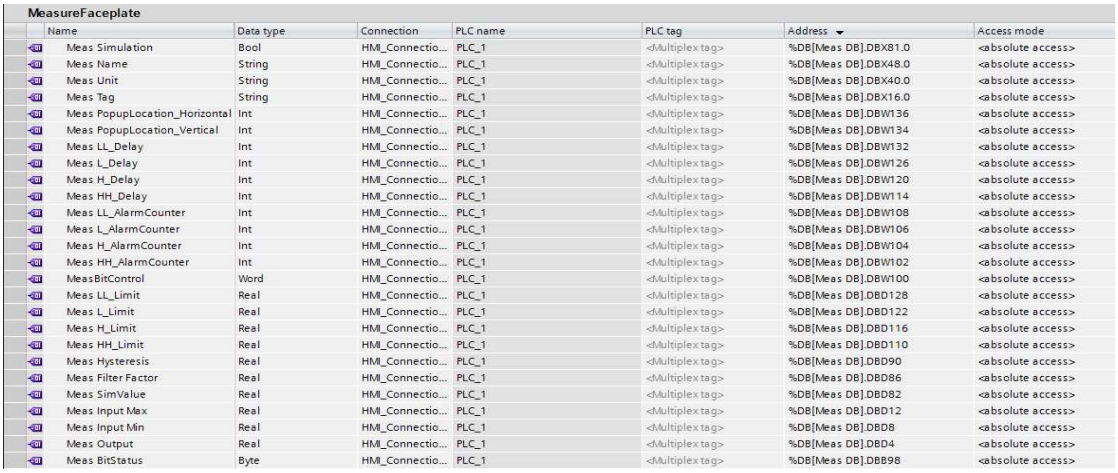


Measure_CPU1500

	Name	Data type	Offset	Default value	Accessible f...	Writa...	Visible in ...	Setpoint	Supervis...	Comment
10	Static									
11	HMI_Parameters	Struct	16.0							
12	Tag	String[22]	16.0	'tag'						Tag
13	Unit	String[6]	40.0	'unit'						Unit
14	Name	String[30]	48.0	'name'						Name
15	HH_enable	Bool	80.0	false						HH alarm enabled
16	H_enable	Bool	80.1	false						H alarm enabled
17	L_enable	Bool	80.2	false						L alarm enabled
18	LL_enable	Bool	80.3	false						LL alarm enabled
19	HH_alarmCounter...	Bool	80.4	false						
20	H_alarmCounterRe...	Bool	80.5	false						
21	L_alarmCounterRe...	Bool	80.6	false						
22	LL_alarmCounterR...	Bool	80.7	false						
23	Simu	Bool	81.0	false						
24	SimuVal	Real	82.0	0.0						
25	Filter Factor	Real	86.0	0.0						Greater number means greater filtering
26	Hysteresis	Real	90.0	0.0						Alarm Hysteresis
27	Fault	Bool	94.0	false						Wire break or short circuit
28	FaultCounter	Int	96.0	0						
29	BitStatus	Byte	98.0	16#0						Status to HMI
30	BitControl	Word	100.0	16#0						Control from HMI
31	HH_alarm_counter	Int	102.0	0						
32	H_alarm_counter	Int	104.0	0						
33	L_alarm_counter	Int	106.0	0						
34	LL_alarm_counter	Int	108.0	0						
35	HH_limit	Real	110.0	90.0						HH alarm limit
36	HH_delay	Int	114.0	5						HH alarm delay
37	H_limit	Real	116.0	80.0						H alarm limit
38	H_delay	Int	120.0	5						H alarm delay
39	L_limit	Real	122.0	20.0						L alarm limit
40	L_delay	Int	126.0	5						L alarm delay
41	LL_limit	Real	128.0	10.0						LL alarm limit
42	LL_delay	Int	132.0	5						LL alarm delay

Kuvio 12. HMI-näyttökuvassa käytetyt Mittauspiirin Datablockin muuttujat.

Ennen kuin pystyin siirtymään muokkailemaan näyttökuvia, täytyi tagi-listat korjata ajan tasalle. Datablockien muuttujiin tuli väkisinkin muutoksia, jo pelkästään lisättävien signaalien viemisestä näyttökuviin (ks. Kuvio 13). Tagi-listoihin lisättiin kaikki uudet signaalit, jotka haluttiin viedä näyttökuviin, mutta myös muokkailtiin tiettyjä signaalin vientimuotoja. Boolean tyyppiset muuttujat vietiin pakattuina esimerkiksi koottuna yhteen tavuun tai jopa sanaan.



MeasureFaceplate

Name	Data type	Connection	PLC name	PLC tag	Address	Access mode
Meas Simulation	Bool	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBX81.0	<absolute access>
Meas Name	String	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBX48.0	<absolute access>
Meas Unit	String	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBX40.0	<absolute access>
Meas Tag	String	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBX16.0	<absolute access>
Meas PopupLocation_Horizontal	Int	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBW136	<absolute access>
Meas PopupLocation_Vertical	Int	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBW134	<absolute access>
Meas LL_Delay	Int	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBW132	<absolute access>
Meas L_Delay	Int	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBW126	<absolute access>
Meas H_Delay	Int	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBW120	<absolute access>
Meas HH_Delay	Int	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBW114	<absolute access>
Meas LL_AlarmCounter	Int	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBW108	<absolute access>
Meas L_AlarmCounter	Int	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBW106	<absolute access>
Meas H_AlarmCounter	Int	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBW104	<absolute access>
Meas HH_AlarmCounter	Int	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBW102	<absolute access>
MeasBitControl	Word	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBW100	<absolute access>
Meas LL_Limit	Real	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBD128	<absolute access>
Meas L_Limit	Real	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBD122	<absolute access>
Meas H_Limit	Real	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBD116	<absolute access>
Meas HH_Limit	Real	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBD110	<absolute access>
Meas Hysteresis	Real	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBD90	<absolute access>
Meas Filter Factor	Real	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBD86	<absolute access>
Meas SimValue	Real	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBD82	<absolute access>
Meas Input Max	Real	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBD12	<absolute access>
Meas Input Min	Real	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBD8	<absolute access>
Meas Output	Real	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBD4	<absolute access>
Meas BitStatus	Byte	HMI_Connectio...	PLC_1	<Multiplex tag>	%DB[Meas DB].DBB98	<absolute access>

Kuvio 13. Mittauspiirin HMI-näytön tagi-lista

Lopuksi näyttöihin tehtiin tarvittavat indikaattorit näyttämään tageihin liitetyt signaalit. Myös muista indikaattoreista tarkastettiin samassa yhteydessä, että niihin oli liitetty oikeat tagit indikoimaan niille tarkoitettuja oikeita muuttujia (ks. Liitteet 1-4). Viimeiseksi kokeilin PLCSIM-lisäosalla simuloimalla, kuinka lohkot pääpiirtein toimivat. Oletus kuitenkin on se, että koska ne on tehty samoilla tekniikoilla kuin aiemmat ominaisuudet, että niiden kuuluisi toimia oikeissa prosesseissa niin kuin ne on suunniteltukin toimivan.

6 Pohdinta

Opinnäytteen tavoitteena oli parannella valmiit ohjelmalohkoja kooditasolta piirinäyttökuviin ja lisäillä niihin mittareita helpottamaan operaattoria seuraamaan prosessin tarpeita kunnossapidon näkökulmasta. Opinnäytetyön tavoitteena oli myös tehdä ohjelmointipohjan neljästä peruspiiristä sellaiset, että niitä voitaisiin käyttää sellaisenaan tulevissa projekteissa. Tällä säästettäisiin aikaa automaatio-ohjelmointiosuuksissa.

Työn tuloksena oli neljä paranneltua peruspiiriä. Peruspiirit ovat nyt sillä tasolla, että ne ovat suoraan valmiita käytettäväksi tuleviin projekteihin. Ohjelmointiosuuden konkreettista ajan säästöä on vaikea arvioida vielä, koska sitä ei olla keretty käyttämään yhdessäkään projektissa. Opinnäytetyö pääsee kuitenkin välittömästi koetukselle, sillä se otetaan käyttöön nyt alkavassa projektissa ja tätä myötä nähdään miten suuri konkreettinen hyöty siitä on ohjelmointiosuudelle. Peruspiireihin lisätyt ominaisuudet eli käyttöaikoihin ja kertoihin liittyvät laskurit mahdollistavat myös vaihtoautomaatiolta vaadittavat ominaisuudet ohjelmointipohjalta. Tämä näkyy kunnossapidon tarpeissa operaattorin apuna.

Työssä saavutettiin halutut tavoitteet toimeksiantajan ja minun osalta. Peruspiirit saatiin korjattua sille tasolla, että ne ovat valmiita suoraan käytettäviksi sellaisinaan. Piireihin saatiin lisättyä halutut kunnossapitoa helpottavat ominaisuudet. Simuloitaessa piirit toimivat ongelmitta, mutta testaus oikeassa projektissa jäi tekemättä ja

sen tuloksen näkee vasta tulevina kuukausina. Omalta osaltani pääsin paljon työskentelemään TIA Portalilla ja tätä myötä sain ohjelmointikokemusta mitä lähdin itselleni hakemaan.

Työ onnistui pääpiirteiltään suunnitellusti. Työn edetessä huomattiin, että käytettävyyttä olisi voinut HMI-piirinäyttöjen osalta voinut hieman parantaa. Siispä ne jääköön kehitysideoiksi. Käytettävyyteen oli toivottu ratkaisun löytämistä popup-ikkunoiden luomiseksi. TIA Portalin oma popup-ikkuna funktio tuntui kovin kömpelöltä käyttää ja siksi siihen toivottiin käytännöllisempää ratkaisua. Itse piirien HMI-näyttökuviin olisi voinut lisäillä vaikka ja mitä, mutta tällä hetkellä niissä on kaikki välttämätön prosessin tarkkailun ja ohjauksen kannalta.

Lähteet

Function Block Diagram (FBD) for S7-300 and S7-400 Programming. 2010. Reference Manual. Siemens. Viitattu 19.11.2018.
https://cache.industry.siemens.com/dl/files/487/45522487/att_61497/v1/s7fup__b.pdf

Helppoa automaatio-ohjelmointia S7-1200-logiikalla. N.d. Siemens. Viitattu 20.11.2018.
http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automaatiotekniikka/ohjelmoitavat_logiikat_simatic/s7_1200.htm

Ladder Logic (LAD) for S7-300 and S7-400 Programming. 2010. Reference Manual. Siemens. Viitattu 19.11.2018.
https://cache.industry.siemens.com/dl/files/822/45523822/att_82001/v1/s7kop__b.pdf

Rajala-Rahko, V. 2010. Toimilohkokirjasto kattilalaitoksen ohjaamiseen. Opinnäytetyö. Vaasan Ammattikorkeakoulu. Tekniikka ja liikenne. Viitattu 19.11.2018.
http://www.theseus.fi/bitstream/handle/10024/16766/Rajala-Rahko_Ville.pdf?sequence=1

Saari, S. 2015. PLCOpen XML -esitystapa sovelluster siirrossa. Metropolia Ammattikorkeakoulu. Automaatiotekniikka. Viitattu 19.11.2018.
https://www.theseus.fi/bitstream/handle/10024/92974/PLCopen%20XML%20-%20Sami%20Saari.pdf?sequence=1

SFS 175-2. 2006. 1. painos. Automaatio. Osa 2: Ohjelmointi ja teollisuusprosessien valvonta. Helsinki. Suomen standardisoimisliitto SFS RY. Viitattu 19.11.2018.

SIMATIC HMI -kosketusnäytöt. N.d. Siemens. Viitattu 20.11.2018
www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automaatiotekniikka/ohjelmoitavat_logiikat_simatic/ohjelmistot/tia_portal_step7.htm

Tehokasta automaatio-ohjelmointia S7-1500-logiikalla. N.d. Siemens. Viitattu 19.11.2018.
http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automaatiotekniikka/ohjelmoitavat_logiikat_simatic/s7_1500.php

TIA Portal (SIMATIC STEP 7). N.d. Siemens. Viitattu 20.11.2018
www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automaatiotekniikka/ohjelmoitavat_logiikat_simatic/ohjelmistot/tia_portal_step7.htm

Understanding the IEC61131-3 Programming Languages. 2009. Rexroth. Viitattu 20.11.2018.
https://www.automation.com/pdf_articles/IEC_Programming_Thayer_L.pdf

Liitteet

Liite 1. Mittauspiirin HMI-näyttökuva

Liite 2. Säätöpiirin HMI-näyttökuva

Liite 3. Venttiilipiirin HMI-näyttökuva

Liite 4. Moottoripiirin HMI-näyttökuva