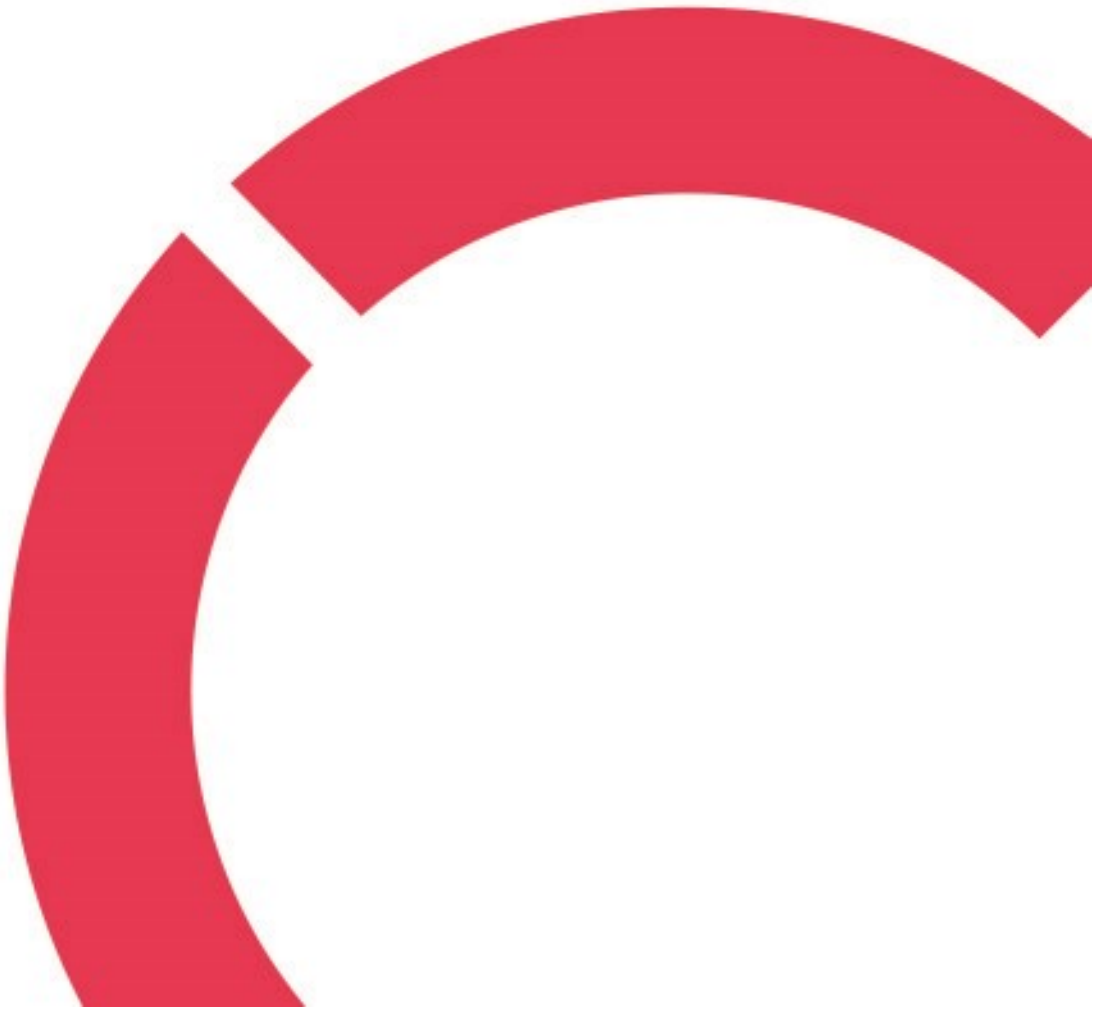
Tomi Laakso

AUTOMAATION SUUNNITTELU JÄRJESTELMÄÄ VAIHTAESSA

Vanha Siemens-järjestelmä vaihdetaan ABB:n 800xA-järjestelmään

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Sähkö- ja automaatiotekniikan koulutus
Syyskuu 2023



TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Syyskuu 2023	**Tekijä/tekijät** Tomi Laakso
Koulutus Sähkö- ja automaatiotekniikka		☒ AMK ☐ YAMK
Työn nimi AUTOMAATION SUUNNITTELU JÄRJESTELMÄÄ VAIHTAESSA		
Työn ohjaaja Hannu Ala-Pöntiö		**Sivumäärä** 56 + 2
Työelämäohjaaja Taneli Känsäkangas		

Opinnäytetyön tavoitteena oli siirtää kemian prosessilaitoksen typellä toimiva jäähdytyslaitteisto Cumulus px100 vanhasta Siemens S7-CPU 313C-järjestelmästä yritykselle räätälöityyn ABB 800xA-järjestelmään. Työntoimeksiantajana oli CABB Kokkola, joka pääasiallisesti valmistaa kasviensuojeluaineita.

Cumulus px100-järjestelmän siirrossa käydään läpi, millaista laitteistoa ja instrumentointia on käytetty, mitä komponentteja vanhasta järjestelmästä pystytään hyödyntämään uudelleen ja millaisia turvatoimia pitää ottaa huomioon laitteiston uudelleen ohjelmoimisessa 800xA-järjestelmään. Riskien arviointiin ja turvallisuuden määrittelyyn on käytetty HAZOP-analyysia.

Opinnäytetyön teoriaosuudessa käydään läpi automaatiosuunnittelun eri vaiheita, tarkastelemme erilaisia riskien arviointimenetelmiä ja arvioimme, miksi HAZOP-menetelmä sopii hyvin prosessi teollisuuteen. Työssä tutkitaan myös 800xA-järjestelmän ohjelmointirakennetta, laitteistoa ja työssä esiintyviä ohjelmointikieliä ja malleja.

Projektin lopputuloksena saatiin toiminnallinen ja siistiltä näyttävä graafinen käyttöliittymä. Ohjelman graafisiin elementteihin tehtiin helppokäyttötoimintoja antamaan operaattorille mahdollisimman paljon tietoa laitteen eri tiloista ja tilojen vaiheista. Ohjelman toiminnallisuutta ei päästy kokeilemaan käytännössä, koska projekti otetaan vasta myöhemmin käyttöön. Tästä syystä ohjelman toiminnallisuus on voitu todeta toimivaksi pelkästään simuloinnin kautta.

Kokonaisuudessaan opinnäytetyöprosessi oli mielenkiintoinen. Vaikka alustava tietopohja CABB:n 800xA-järjestelmästä oli vähän hutera, sain nopeasti kiinni työssä tarvittavista tekniikoista ja pääsin etenemään suhteellisen jouhevasti eteenpäin ohjelmointivaiheessa. Sain kerättyä paljon tietoa tehdessäni ohjelmointia CABB:lle, mikä helpotti opinnäytetyön teoriaosuuden kehittämistä. Teoria-osuus kokoaa hyvin yhteen tutkimuksen tulokset ja esittää työssä käytettyjä ohjelmointi tekniikoita.

Asiasanat
ABB 800xA, automaatiosuunnittelu, järjestelmäsuunnittelu, ohjelmointi, prosessilaitos, prosessinohjaus, riskinarviointi

ABSTRACT

Centria University of Applied Sciences	Date September 2023	Author Tomi Laakso
Degree programme Electrical and automation engineering		
Name of thesis AUTOMATION PLANNING WHEN CHANGING THE SYSTEM		
Centria supervisor Hannu Ala-Pöntiö		**Pages** 56 + 2
Instructor representing commissioning institution or company Taneli Känsäkangas		

The aim of the thesis work was to transfer the chemical process plant's nitrogen-powered cooling equipment Cumulus px100 from the old Siemens S7-CPU 313C system to an ABB-800xA system. This specific version of ABB-800xA programming tool is custom-made for the company. The client was CABB Kokkola, whose main product is a crop protection agent.

When transferring the Cumulus px100 system, the type of equipment and instrumentation that have been used was reviewed, which components are reusable, and what kind of safety measures must be considered when reprogramming the current cooling solution into the 800xA system. The HAZOP method was used for risk assessment and safety definition.

The theory part of the thesis discusses the different phases of automation, looks at possible risk assessment methods, and evaluates why the HAZOP method is well suited to process engineering. In addition, the operations of the 800xA systems program structure, equipment, and programming models and languages will be discussed.

As a result of the project, the program achieved a functional and visually appealing graphical user interface. Easy-to-use functionalities were implemented into the graphical elements of the program to provide as much information as possible to the operators about the device's different states and phases. Due to the project being implemented at a later stage, the program's functionality in practice were unable to be tested. Therefore, the program's functionality has been verified solely through simulation.

Overall, the thesis process was intriguing. Even though having a limited initial knowledge about CABB's 800xA system, the necessary techniques were quickly learned, and smooth progress was made during the programming phase. While programming for CABB, a substantial amount of information was gathered, contributing to the development of the theoretical part of the thesis. The theory section effectively unites the acquired knowledge and presents the techniques employed in programming.

Key words
ABB 800xA, automation design, process control, process plant, programming, system design, risk assessment

KÄSITTEIDEN MÄÄRITTELY

barg	Mittaripaine/hyötypaine (ei ilmoita absoluuttista ilmanpainetta)
CCM	Ohjelmoitava säätömoduuli (Custom Control Module)
CPU	Prosessori/suoritin, suorittaa ohjelmointi käskyjä (Central Processing Unit)
EN IEC	Eurooppalainen sovitus, kansainvälinen sähköalan standardointiorganisaatio (International Electrotechnical Commission)
FD	Toimintakaavio, ohjelmointikieli (Function Diagram)
HAZOP	Riskianalyysimenetelmä/poikkeamatarkastelu (Hazard and Operability Study)
IED	Älykäs elektroninen laite (Intelligent Electronic Device)
ISO	kansainvälinen standardisoimisjärjestö (International Organization for Standardization)
I/O	Siirräntä, tiedonsiirto (Input/Output)
LOPA	Semikvantitatiivinen riskinarviointimenetelmä (Layer Of Protection Analysis)
EU-OSHA	Euroopan työterveys- ja työturvallisuusvirasto (Occupational Safety and Health Administration)
PC	Tietokone (Personal Computer)
PFH_d	Vaarallisen vian todennäköisyys tunnissa, laskeminen (probability of dangerous failure per hour)
PHA	Prosessin vaarojen arviointi, analyysi (Process hazard analysis)
PI	Putkitus- ja instrumentointikaavio
PL	Suorituskykytaso, laskeminen (Performance Level)
POU	Ohjelman organisointiyksikkö (Program Organization Unit)
SFC	Toiminnallinen sekvenssikaavio, ohjelmointi (Sequential function chart)
SIF	Turvatoiminnon hallintalaite (Safety Instrumented Function)
SIL	Turvallisuuden eheystaso (Safety Integrity Level)
SIS	Turvatoimintojen hallintajärjestelmä (safety instrumented system)
ST	Ohjelmointikieli (Structured Text)

TIIVISTELMÄ
ABSTRACT
KÄSITTEIDEN MÄÄRITTELY
SISÄLLYS

1 JOHDANTO 1

2 AUTOMAATION SUUNNITTELU JA PROSESSIRISKIEN HALLINTA 2
2.1 Automaation suunnittelu ja toimintaa esittävät kaaviot 2
2.1.1 PI-kaavio 4
2.1.2 Lukituskaavio 5
2.2 PHA & HAZOP 6
2.3 Turvasuunnittelu standardeilla PL & SIL 11

3 JÄRJESTELMÄT JA LAITTEISTO 15
3.1 ABB 800xA-laitteisto 15
3.2 ABB 800xA-ohjelmisto 17
3.2.1 Ohjelmointirakenne 20
3.2.2 Function Diagram 21
3.2.3 Structured Text & Sequential Function Chart 23
3.2.4 Graafinen suunnittelu 28
3.3 PHA-Pro & LOPA 32

4 CABB 34
4.1 CABB Kokkola 34
4.2 CABB:n asettamat tavoitteet opinnäytetyölle 35

5 CUMULUS PX100 OHJELMOINTI 36
5.1 Ohjelmassa käytettyjen raja-arvojen määrittely 36
5.2 Graafinen käyttöliittymä 40
5.3 Toiminnallisuuden ohjelmoiminen 46
5.3.1 Toimilaitteiden lisääminen ohjelmistoon 48
5.3.2 Lukitusten ohjelmointi 49
5.3.3 Ohjattu käynnistyminen ja pysyttäminen käyttäen SFC:tä 53
5.4 Ohjelmoinnin ulkopuolelle jääviä asioita 55

6 POHDINTA 56

LÄHTEET .....
LIITTEET

KUVIOT

KUVIO 1. Automaatio suunnittelun eteneminen 3
KUVIO 2. Automaation toimintaa kuvaavat piirustukset 3
KUVIO 3. Instrumentoinnin tunnuskirjainten käyttö SFS 14617-6 4
KUVIO 4. Työssä esiintyviä PI-kaavio merkintöjä 5
KUVIO 5. PHA-analyysin eteneminen 7

KUVIO 6. Esimerkki PI-kaavio 9
KUVIO 7. Toiminta kuvaukset ja vaaratilanteiden arviointi 10
KUVIO 8. HAZOP-analyysin vaiheet 10
KUVIO 9. ABB-riskienarviointi menetelmä ohjelmoitaessa 11
KUVIO 10. Riskin arviointi (PL_r) 12
KUVIO 11. SIL-luokitukset 13
KUVIO 12. ABB 800xA-verkkorakenne 15
KUVIO 13. AC 800M ohjausjärjestelmä ja S800 I/O yksikkö 16
KUVIO 14. Esimerkki ”Aspect Object” toimintamallista 19
KUVIO 15. Ohjausmoduulien ja funktiolohkojen yhteys käyttölaitteisiin 20
KUVIO 16. Esimerkki liitäntäpisteistä, lohkojen graafisista yhteyksistä ja yhteyden invertoinnista 22
KUVIO 17. järjestelmien välisiä eroja 23
KUVIO 18. Esimerkki ST-koodista 24
KUVIO 19. Esimerkki sekvenssi rakenteesta “Init” P1 sisältää koodia N ja P0 ovat tyhjiä 24
KUVIO 20. Sekvenssikaavion tulkinta 26
KUVIO 21. Sekvenssin valinta ja samanaikainen sekvenssi 28
KUVIO 22. Grafiikan suunnittelutyökalu Graphics Builder 29
KUVIO 23. PID-ohjaimen Faceplate 30
KUVIO 24. Määritelmä editori 31
KUVIO 25. CABB Kokkolan tehtaat 35
KUVIO 26. Cumulus laitteiston vanhat ja uudet positio tunnukset, ohjelmointirajat ja selvennykset 37
KUVIO 27. Sulkuventtiilin MA-1XV-3405 sulkeminen 38
KUVIO 28. Säätöventtiilin MA-1TV-3406 sulkeminen 39
KUVIO 29. Moottorin MA3171 lukitus 40
KUVIO 30. PI-kaavio, jonka pohjalta luodaan käyttöliittymän grafiikka 41
KUVIO 31. Graafisen liittymän alku 41
KUVIO 32. Viimeistelty graafinen käyttöliittymä eli ohjauksen perusnäkymä 42
KUVIO 33. Tag napin tuomat positiot ja muut lisätiedot graafiseen näkymään 43
KUVIO 34. Linkin MA3728 avaama tuotantolinjan ohjausnäkymä 43
KUVIO 35. Ohjauspaneelin ikkuna ja CCM ikkuna 44
KUVIO 36. Properties lista ”Tag” painikkeen funktionaalisuudelle 45
KUVIO 37. Expression Editor ja pieni tosi vai epätosi ohjelmakoodi ”Tag” napille 46
KUVIO 38. Applikaation alle luotuja toiminnallisia taulukoita 47
KUVIO 39. uCumulus (Batch unit) sisältämät taulukkotyypit 48

KUVIO 40. EQ-sivu 3 Analogimittaukset 48
KUVIO 41. EQ-sivu 4 (PID-säätö) säätöventtiilin kontrollointi suhteessa prosessin lämpötilaan 49
KUVIO 42. Parametrointi lista PID-säädölle MA-1TC-3406 48 49
KUVIO 43. Säätöventtiilin turvalukitus 50
KUVIO 44. Moottorin turvalukitus 51
KUVIO 45. Sulkuventtiilin MA_1XV_3405 turvalukitus 52
KUVIO 46. Set Point Value asetus ja käynnistys tekstin kirjoittaminen 54
KUVIO 47. Ohjelmakoodin suorittaminen stepistä poistuttaessa 54
KUVIO 48. Siirtymä vaiheelle asetettu ehto 55

1 JOHDANTO

Opinnäytetyön aiheen ja siihen liittyvän toimeksiannon sain CABB:lta. CABB valmistaa useita erilaisia kemiallista prosessia vaativia tuotteita; näistä valmistetuista tuotteista suurin osa on kasvinsuojeluaineita. CABB uudisti prosessitiloja, mikä mahdollisti typellä toimivan Cumulus px100 -järjestelmän päivittämisen. Prosessitilojen uudistuksen aikana voidaan myös tarvittaessa päivittää mittareita ja muuta laitteistoa järjestelmän tarkkuuden ja eliniän lisäämiseksi.

Päivitettävä Cumulus px100-järjestelmä toimi Siemensin Simatic S7-CPU 313C-logiikalla, joka on liitetty Siemensin ohjauspaneeliin Simatic OP170B. Siemensillä toteutettu järjestelmä on noin 20 vuotta vanha. Tavoitteena on saada Cumulus px100 ohjelmoitua alusta asti uudestaan CABB:n omaan ABB 800xA-järjestelmään ja saada se toimimaan vähintään yhtä hyvin kuin aiemminkin.

Koska nykyiseen järjestelmään tehdään muutoksia, meidän pitää myös selvittää mahdolliset riskitekijät. Riskien selvittämiseen on käytetty HAZOP-riskienarviointimenetelmää. Työn kannalta tärkeää on se, että HAZOP:ssa todettujen riskien pohjalta tehdään myös ohjelmointia. Riskien arviointi on tärkeää, jotta mahdolliset vaaratilanteet voidaan välttää ennen kuin ne pääsevät edes syntymään.

Opinnäytetyön teoriaosuudessa syvennytään miettimään, millaisia menetelmiä on hyvä käyttää työtä suunnitellessa ja mitä keinoja meillä on arvioida mahdollisia riskitekijöitä liittyen prosessilaitoksilla tehtäviin muutoksiin. Teoriaosuuden viimeisessä vaiheessa käydään läpi, mitä laitteita ja ohjelmistoja meillä on käytettävissä 800xA-järjestelmässä. Työssä tutustutaan myös tarkemmin ohjelmistojen toimintaan ja avataan sitä, kuinka kyseisiä ohjelmistoja käytetään ja mitä mahdollisia rajoituksia ohjelmien käyttöön voi liittyä.

Lopuksi esitellään tilattuun työhön liittyneitä erilaisia työvaiheita. Käydään läpi ohjelmointia ja sitä, millaisia ratkaisuja pitää tehdä, jotta työ saadaan sopimaan HAZOP:n ohjaamiin määritteisiin. Lopuksi pohditaan, päästinkö tavoitteisiin ja olisiko jotakin voinut tehdä toisin tai paremmin.

On myös hyvä mainita se että, työtä ei päästy kokeilemaan käytännössä. Kaikki työhön liittyvä ohjelmointi ja testaus suoritettiin simuloimalla. Tämä johtuu siitä, että projekti otetaan käyttöön vasta vuoden 2024 aikana. Toimeksiantaja on tarkistanut ja hyväksynyt työn laadun.

2 AUTOMAATION SUUNNITTELU JA PROSESSIRISKIEN HALLINTA

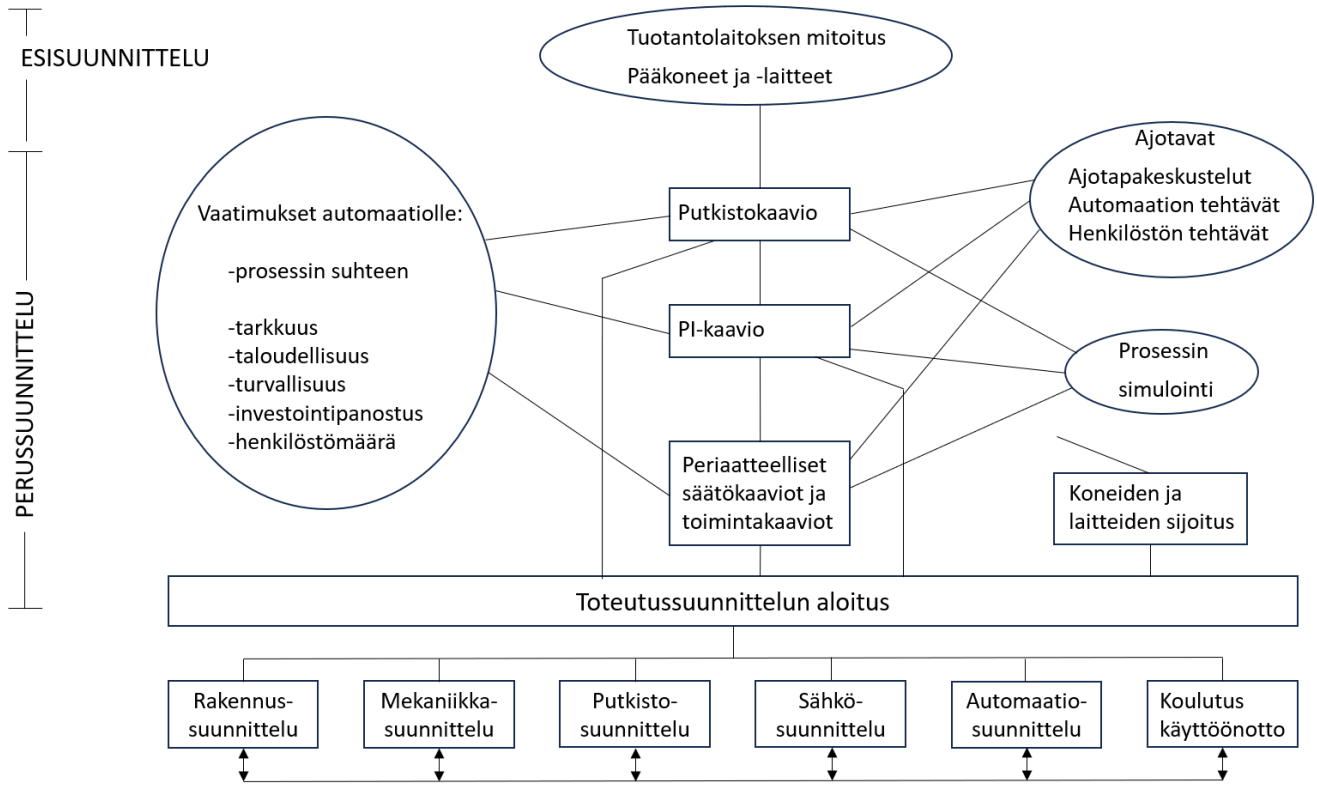
Kun suunnitellaan prosessiteollisuuteen sopivaa automaatioratkaisua, pitää huomioida monia erilaisia asioita ja hankkia tietoa monista eri lähteistä, jotta voidaan saavuttaa paras mahdollinen lopputulos. Kerättyä tietoa pitää myös osata soveltaa oikein, jotta vältytään erinäisiltä virheiltä. Prosessiteollisuudessa pienelläkin virheellä saattaa olla todella merkittävä ja pitkäaikainen vaikutus. Tästä syystä prosessiteollisuuteen on kehitetty erilaisia analysointikeinoja arvioida prosessin ajamiseen liittyviä riskejä. Hyvin suunniteltu työ vähentää huomattavasti riskiä ajautua vaaratilanteisiin, joissa pahimpana vaarana saattaa olla ihmishengen menetys.

Tämä osio käy läpi, millaista tietoa tarvitaan automaatioratkaisuiden suunnitteluun. Tässä osiossa käsitellään myös erilaisia riskienarviointi menetelmiä, koska monet erilaiset riskitekijät ovat usein läsnä, kun suunnitellaan automaatiota. Riskien arviointiprosessi kuului merkittävänä osana opinnäytetyöhön.

2.1 Automaation suunnittelu ja toimintaa esittävät kaaviot

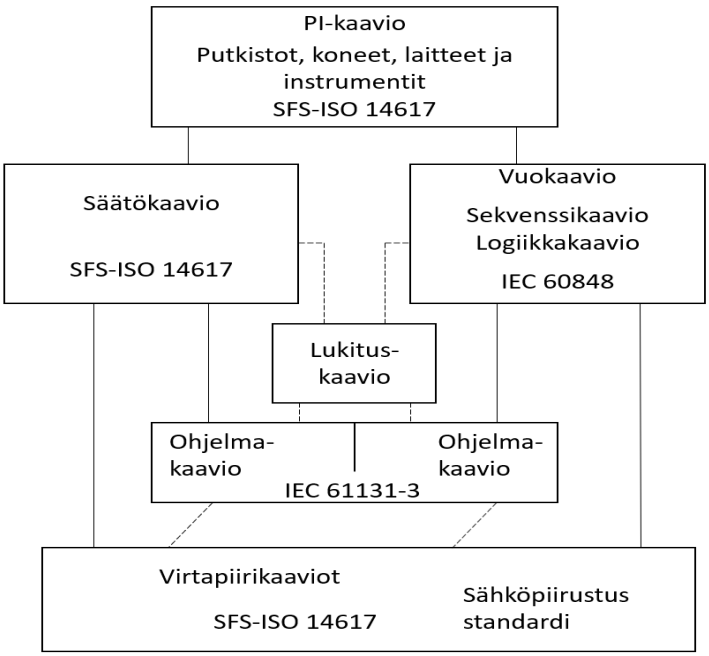
Kun suunnitellaan automaatioratkaisuja nämä yleensä, aloitetaan miettimällä sopivia laitteita ja säätöpiirejä. Ratkaisut tukeutuvat helposti vanhoihin ja entuudestaan hyväksi todettuihin malleihin. Vanhojen mallien käytössä on pienemmät riskit, mutta silloin ei saada hyödynnettyä uuden teknologian tuomia mahdollisuuksia. Uusi teknologia mahdollistaa uusien prosessien automatisoinnin, joita on aiemmin ajettu manuaalisesti. Vastaavasti uudella teknologialla voidaan tehostaa jo olemassa olevia automaatioratkaisuja. (Sivonen 2008, 228.)

Määriteltäessä, millaista automaatiota kohteessa tarvitaan, täytyy selvittää olemassa oleva prosessi, mikä on prosessin tarkoitus, millaisia laatutavoitteita halutaan saavuttaa ja miten nämä päätökset tulevat vaikuttamaan taloudellisuuteen. Automaatiossa on usein erilaisia toimintoja, joita pitää ottaa huomioon kuten esimerkiksi, miten prosessia ajetaan tavallisesti, miten hallitaan tavallisesta toiminnasta poikkeavia tilanteita, miten prosessien ylös- ja alasajot toteutetaan, tuotantolajin tai tyypin erilaiset muutokset, tai muita vastaavia toimintoja, joilla saadaan myös optimoitua tuotantoa. Tarvittava henkilöstön määrä mukautuu automaatioasteen mukaan. Automaatioratkaisuja suunniteltaessa usein mukana on myös käyttö- ja ylläpitohenkilöstöä, koska he tietävät eniten, miten prosessi käyttäytyy normaalitilassa tai poikkeustilanteissa. (Sivonen 2008, 228.)



KUVIO 1. Automaatiosuunnittelun eteneminen (mukaillen Sivonen 2008, 229)

Seuraavaksi tutkitaan kaavioita, joilla kuvataan automaation toimintaa (kuvio 2). Opinnäytetyön kannalta oleellisia kaavioita ovat PI-kaaviot ja lukituskaaviot. Tästä huolimatta on hyvä nähdä miten eri kaaviot linkittyvät toisiinsa, koska jokaisella kaaviolla on tärkeä merkitys automaatiota suunnitellessa.



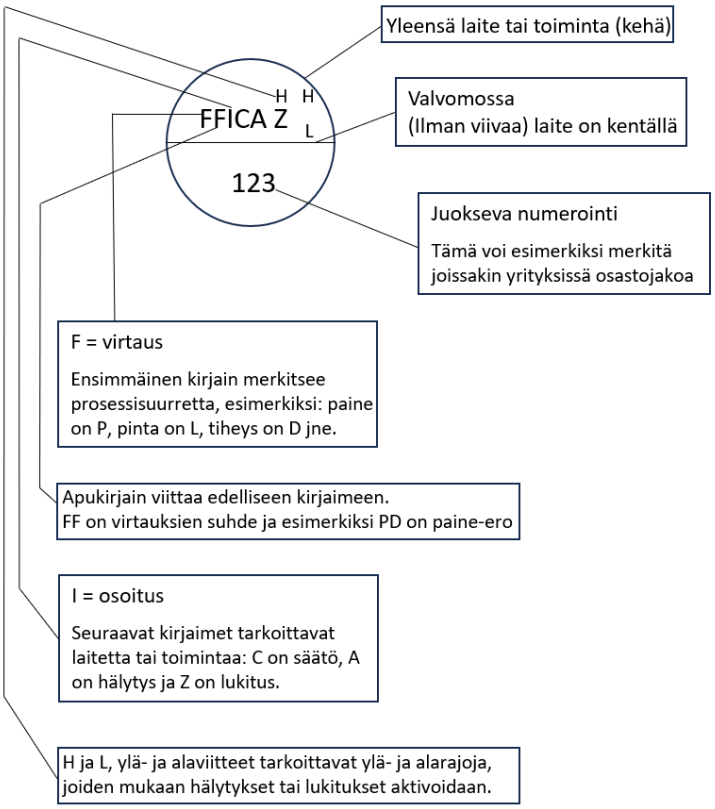
KUVIO 2. Automaation toimintaa kuvaavat piirustukset (mukaillen Sivonen 2008, 243)

2.1.1 PI-kaavio

PI-kaavio esittää kaikki oleelliset tiedot järjestelmään liittyvistä ratkaisuista ja toiminnoista. PI-kaavioihin saadaan tiivistettyä paljon informaatiota käyttämällä erilaisia kuvioita ja kirjainyhdistelmiä. Kirjainten tulkintaan käytetään standardia SFS-ISO 14617-6, esimerkki tulkinnasta (kuviossa 3).

Tarkoitus:
- antaa tiedot prosessin teknillisestä ratkaisusta.
- esittää putkien ja muiden kuljetusteiden yksityiskohtainen kulku
- antaa perustiedot putki-, instrumentointi- ja asennuspiirustusten laatimista varten
- antaa tiedot materiaaliluettelon ja kustannusarvion laatimista varten
- ym. (OAMK 2009, luku 4.2.)

Sisältö:
- kaikki laitteet
- putket ja muut kuljetustiet
- kaikki venttiilit
- mittauspisteet ja säätöpiirit yleispiirrosmerkkejä käyttäen
- laitenumerot
- putkitunnukset
- venttiilien tunnukset
- tulevien ja lähtevien virtojen osoitteet
- ym. (OAMK 2009, luku 4.2.)

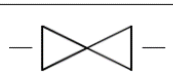
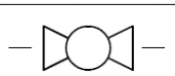
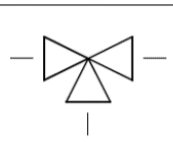
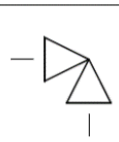
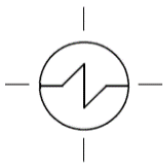
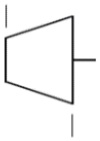
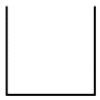


Instrumentoinnin tunnuskirjaimet

	MITTASUURE (alkuperä)	LISÄMÄÄRITE	TOIMINTA
A			*hälytys*
B			eri tilojen näyttö
C			ohjaus
D	tiheys	ero	
E	sähkösuureet		anturitoiminta
F	virtaama	suhde, murtoluku	
G	suhde, asento, pituus		tarkastelu
H	käsiohjaus		
I			osoitus
J	voima	pyyhkäisy, jasottainen toiminta	
K	aika	muutosnopeus	
L	pinnankorkeus		
M	kosteus	hetkellisesti	
N	käyttäjän valittavissa		käyttäjän valittavissa
O	käyttäjän valittavissa		
P	paine, alipaine		testauskohdan yhteys
Q	laatu	yhtenäinen, kokonainen	yhdistäminen, summa
R	säteily		rekisteröinti, tallennus
S	nopeus, taajuus		kytkentä
T	lämpötila		lähettäminen
U	monimuuttuja		monitoiminta
V	käyttäjän valittavissa		vaikuttaminen toimilaitteella
W	paino, voima		
X	määrittelemätön		määrittelemätön
Y	käyttäjän valittavissa		muuntaminen, laskenta
Z	tapahtumien lukumäärä		hätä- tai turvatoiminta

KUVIO 3. Instrumentoinnin tunnuskirjainten käyttö SFS 14617-6 (Mukaillen OAMK 2009, luku 4.2)

Seuraavaksi käydään lyhyesti läpi tässä työssä esiintyviä PI-kaavio merkintöjä (Kuvio 4). Merkinnät on valikoitu ja koottu tähän työhön sopivaksi nipuksi standartista SFS-EN ISO 10628-2. Mainitaan myös se, että tämä ei ole täydellinen lista PI-kaavio merkinnöistä. Tämä ei myöskään ole täydellinen lista työssä esiintyvien PI-merkintöjen osalta, mutta työn ymmärtämisen kannalta oleellisimpia merkintöjä on tuotu esille.

21.1	2101		Venttiili	21.7	X8071		Pallo venttiili
21.3	2103		Kolmitie venttiili	21.2	2102		Kulma venttiili
3.1	X8079		Lämmön vaihdin	20.1	2571		Turbiini
15.1	2301		Neste pumppu	1.2	2061		Säiliö
24.18	511		Letku	24.16	2571		Laippa

KUVIO 4. Työssä esiintyviä PI-kaavio merkintöjä (mukaillen SFS-EN ISO 10628-2, 3–41)

2.1.2 Lukituskaavio

Lukituskaavioilla esitetään automaatiojärjestelmän erilaisia hälytyksiä ja lukituksia. Nämä kaaviot esittävät laitteen tai laitoksen tilaa, kun jokin osa prosessista ylittää asetetun hälytys kynnyksen. Tarkoituksena on estää virheellisiä toimintoja ja antaa käyttäjille tarpeellista informaatiota järjestelmästä. Lukituskaavioiden toiminnat voidaan piirtää omille kaavioille tai niistä voidaan tehdä erillisiä luetteloita. (Sivonen 2008, 244.)

Opinnäytetyöhön tehdyt lukituskaaviot toimivat esimerkkinä sille miltä lukituskaavio näyttää ja ne antavat myös esimerkkiä siitä, kuinka lukituskaavion voi tehdä. Mainitaan myös se, että lukituskaavion voi esittää eri tavoilla. Lukituskaaviot löytyvät luvusta 5.1: kuviot 27–29.

2.2 PHA & HAZOP

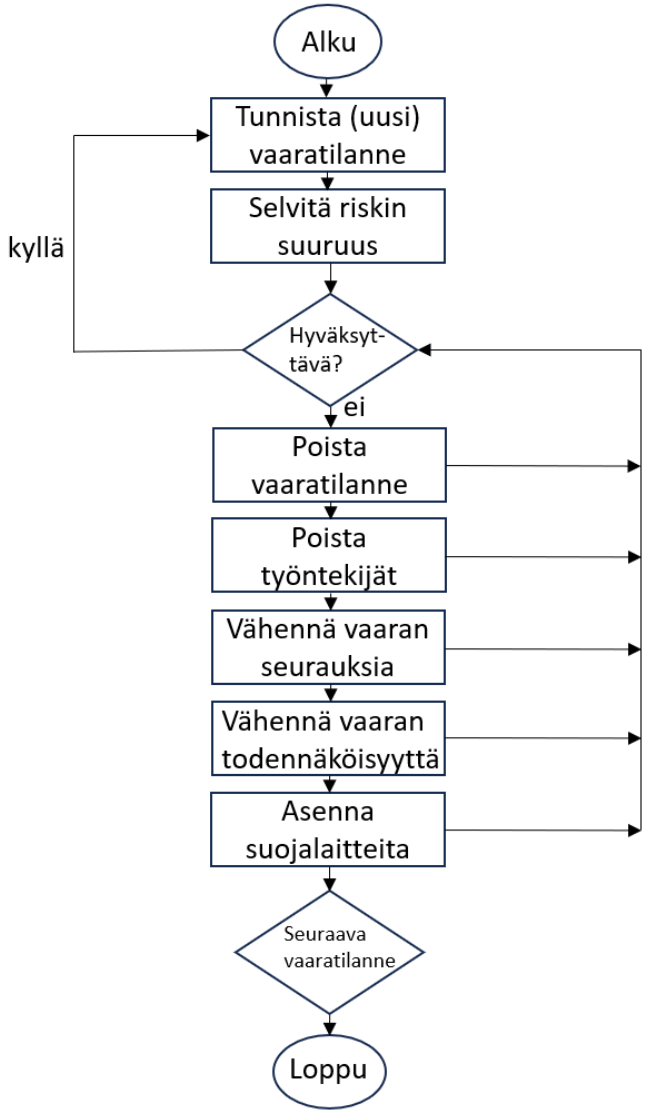
Käytännössä PHA (Process Hazard Analysis) on yksityiskohtainen ja vaiheittain suoritettava katsaus kemiallisten tuotantolaitosten tuotantoprosesseihin ja menettelytapoihin. PHA analysoi tietoja laitteista, instrumenteista, hyödykkeistä ja ihmisten käyttäytymisestä, korostaen potentiaalisia syitä vaarallisten kemikaalien vapautumiselle ja arvioi tapahtumien seurauksia. OSHA (Occupational Safety and Health Administration) ja Euroopan työturvallisuus- ja työterveysvirasto (EU-OSHA) vaativat PHA:ta teollisuuden prosessille, jotka käyttävät vaarallisia kemikaaleja. Vakuutusyhtiöt ja lupaviranomaiset edellyttävät myös PHA:n laatimista. (What is the Difference Between PHA, HAZOP & LOPA? 2022, luku ”What is PHA?”.)

PHA keskittyy tunnistamaan vaaratilanteita prosessiteollisuudessa hyödyntämällä erilaisia riskienarviointimenetelmiä. Riskien arviointimenetelmien tarkoituksena on helpottaa prosessiteollisuuden vaaratilanteiden tunnistamista eri näkökulmista, kuten esimerkiksi mitkä ovat vammautumisen tai kuolemisen vaara, ympäristölliset tai taloudelliset haitat, kuinka todennäköisesti tai usein vaaratilanteet toteutuvat, mitä voidaan tehdä vaaratilanteiden poistamiseksi tai haittojen minimoimiseksi. PHA-analyysin tarkoituksena on myös auttaa työyhteisöä ajattelemaan vaaratilanteita ennaltaehkäisevästi ja miettimään mitä seurauksia heidän toimintansa saattaisi aiheuttaa prosessia operoitaessa. Prosessiteollisuudessa käsitellään usein vaarallisia aineita vaarallisissa prosesseissa, jossa paineet saattavat olla todella suuria tai vastaavasti lämpötilan muutokset ovat suuria. (Sutton 2010, 80.)

Pääsääntöisesti PHA-analyysit suoritetaan ryhmissä. Nämä edellä mainitut ryhmät usein pitävät sisällään erilaisia henkilöitä kuten insinöörejä, operaattoreita, prosessi suunnittelijoita ja huoltoa suorittavia henkilöitä. Kokoukseen halutaan erilaisia työyhteisön jäseniä, koska silloin saadaan mahdollisimman paljon hyödyllistä tietoa useilta eri henkilöiltä mahdollisesti syntyvistä vaaratilanteista ja kehitetään keskustelua. (Sutton 2010, 83.)

Kokouksen vetäjällä on tärkeä rooli keskittää keskustelua oikeisiin aiheisiin, koska PHA-analyysin tarkoituksena on huomioida pelkästään ne vaaratilanteet, jotka liittyvät prosessin ajamiseen ja siitä mahdollisesti syntyviin vaaratilanteisiin. Kokouksessa selvitetään vaaratilanteiden riskialttius, todennäköisyys ja vaarallisuuden aste. Paras tapa saada vaaratilanteet hallintaan on poistaa vaaraa aiheuttava tekijä kokonaisuudessaan, mutta tämä ei ole aina mahdollista, joten pitää selvittää miten vaaratilannetta saataisiin pienennettyä mahdollisimman paljon. (Sutton 2010, 80–83.)

Kuvio 5 kiteyttää hyvin sen, kuinka PHA-analyysin arviointi prosessi edistyy.



KUVIO 5. PHA-analyysin eteneminen (mukaillen Sutton 2010, 82)

PHA-analyysissä käytettyihin arviointimenetelmiin kuuluvat

- Hazard and operability study (HAZOP) analyysi
- What-if (Mitä jos) analyysi
- Failure mode and effects analysis (FMEA) vikatila ja vaikutusten analyysi
- Fault-tree analysis (FTA) vikapuuanalyysi
- Tarkistuslistat (Turton & Shaeiwitz & Bhattacharyya & Whiting, 1145–1146.)
- Major Hazards Screening (suurten vaarojen seulonta)
- Monte Carlo simulaatio ja Markov analyysi (Sutton 2010, 84.)

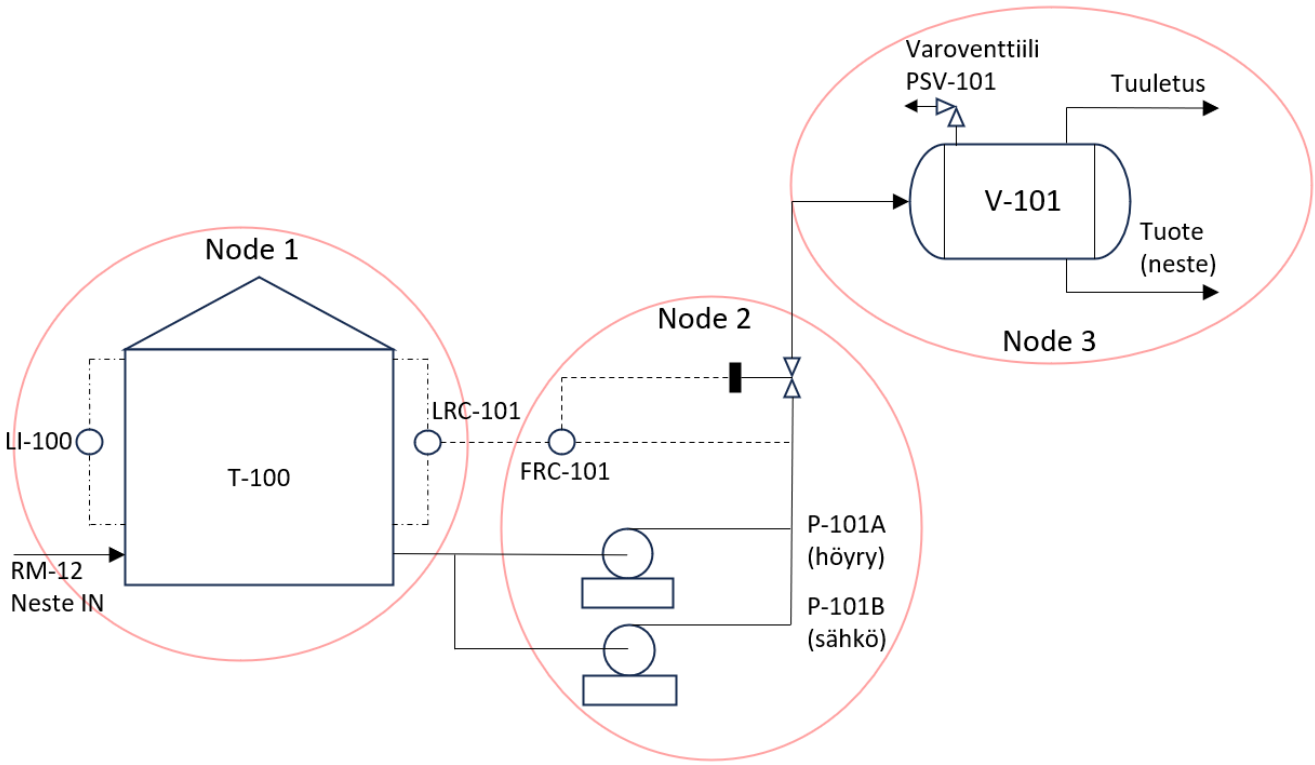
Työn kannalta oleellisin menetelmä on HAZOP-analyysi. Tästä syystä HAZOP-analyysia tutkitaan tarkemmin.

HAZOP

HAZOP (Hazard and Operability Analysis) on oletettavasti kaikista suosituin riskienarviointimenetelmä. Useat työntekijät, jotka työskentelevät prosessiteollisuudessa ovat ainakin jossakin yhteydessä kuulleet termin HAZOP ja he osaavat yhdistää tämän termin prosessiturvallisuuteen. Monet yritykset pitävät HAZOP-menetelmästä, koska HAZOP-menetelmällä on hyvä maine, HAZOP-menetelmä vähentää riskiä vaaratilanteiden syntymiselle ja tarvittaessa sitä on helppo puolustaa oikeuden edessä. HAZOP-menetelmää on helppo puolustaa oikeudessa, koska kyseinen menetelmä on niin perusteellisesti toteutettu. (Sutton 2010, 132.)

Yleisesti ottaen HAZOP-menetelmässä systeemi/yksikkö hajautetaan solmukohdista (node) tarkempaa tarkastelua varten. Solmukohdat keskitetään niihin alueisiin, jotka muuttavat merkittävästi systeemissä ajettavaa prosessia. Esimerkiksi jos systeemissä on pumppu, joka siirtää tuotetta tankista toiseen tätä kyseistä pumppua voidaan pitää solmukohtana, koska tämä kyseinen pumppu nostaa prosessissa olevan virtauksen ja paineen määrää. Muita yleisiä solmukohtia ovat esimerkiksi lämmönvaihtimet ja reaktorit. Lämmönvaihdin muuttaa prosessissa olevan tuotteen lämpötilaa ja vastaavasti reaktorissa oleva tuote muuttaa kemiallista muotoaan. Todellisuudessa solmukohdat saattavat sisältää useamman laitteen samanaikaisesti, jos niistä aiheutuvat muutokset ja vaaratilanteet ovat toistuvia, tai varsinkin silloin kun käytetään samalla periaatteella toimivia laitteita lähekkäin. (Sutton 2010, 133–134.)

Kuviot 6, 7 ja 8 esittävät miten HAZOP etenee yleisellä tasolla.



KUVIO 6. PI-kaavio esimerkki solmukohtien määrittelystä (mukaillen Sutton 2010, 134)

- Solmukohta 1 (node 1) kuvaa säiliötä T-100, siihen liittyvää laitteistoa ja instrumentointia. (Prosessiin vaikuttava tekijä on säiliön pinnankorkeus)
- Solmukohta 2 (node 2) kuvaa pumppuja P-101 A/B ja virtauksensäätöventtiiliä FRC-101. (Prosessiin vaikuttavat tekijät ovat virtauksen määrä ja nesteen paine)
- Solmukohta 3 (node 3) kuvaa painesäiliötä V-101, varoventtiiliä PSV-101 ja muuta instrumentaatiota. (Prosessiin vaikuttavat tekijät ovat paine, pinnankorkeus ja kemiallinen koostumus) (Sutton 2010, 134.)

Kun prosessin solmukohdat ovat määritetty luodaan sopivat toimintakuvaukset ja tutkitaan mahdollisia vaaratilanteiden aiheuttajia (KUVIO 7).

Solmukohdan (node) toiminta ja kuvaus		
Node:n Numero	Nimi	Toiminta
1	Säiliö T-100 ja siihen liittyvä instrumentaatio.	Säiliö T-100 toimii nesteen RM-12 varastona, säiliön neste tulee ulkopuolisilta toimittajilta säiliö autoissa. Solmukohdassa ei huomioida säiliön täyttöjärjestelmää.
2	Pumput P-101 A/B ja virtauksensäätöventtiili FRC-101	Pumput P-101 A/B siirtävät nestettä RM-12 säiliöstä T-100 säiliöön V-101. Virtausta ohjaa FRC-101, säädön asetusarvo tulee ohjaimelta LRC-101 (node1). Pumput eivät toimi samanaikaisesti. Pumppu A toimii höyryllä ja pumppu B toimii sähköllä. Yleensä pumppu B on lepotilassa.
3	Painesäiliö V-101 ja varoventtiili PSV-101	Neste RM-12 virtaa tähän säiliöön useista eri lähteistä. Säiliö V-101 toimii virtauksen tasaajana vähentäen muutoksia virtaus nopeuksissa. Tuuletus aukko poistaa säiliöön jääneet kaasut.

Vaaratilanteiden syyt			
Node	Prosessin muuttuja	Poikkeama	Syy
1	Pinta	Korkea	1. Korkea virtaus säiliöön T-100 2. Häiriö säiliön T-100 pinnansäätimessä 3. Pumput P-101A ja P-101B pysähtyvät
		Matala	1. Matala/heikko virtaus säiliöön T-100 2. Häiriö säiliön T-100 pinnansäätimessä
2	Virtaus	Korkea	1. Häiriö säiliön T-100 pinnansäätimessä 2. Liian nopea pumpun toiminta
		Matala/Ei	1. Häiriö säiliön T-100 pinnansäätimessä 2. Pumpun mekaaniset ongelmat
		Käänteinen	1. Pumpun toimintahäiriö (viallinen takaiskuventtiili)

KUVIO 7. Toimintakuvaukset ja vaaratilanteiden arviointi (mukaillen Sutton 2010, 135–138)

Kuviosta 7 huomataan, että mahdollisia vaaratilanteita aiheuttavat säiliön T-100 pinnankorkeus ja virtaus säiliöstä T-100 säiliöön V-101. Jotkin vaaratilanteet aiheutuvat useammasta kuin yhdestä syystä. Esimerkiksi säiliössä T-100 on kolme erillistä vaaratilanteen aiheuttajaa, jotka johtavat liialliseen pinnankorkeuteen. (Sutton 2010, 138.)

Viimeinen esimerkki käsittelee HAZOP-analyysin eri vaiheita (KUVIO 8). Nämä vaiheet toistuvat jatkuvasti HAZOP-analyysin edetessä. Tämä vaiheiden jatkuva toistaminen johtaa siihen, että työyhteisö alkaa ajattelemaan vaaratilanteita ennaltaehkäisevästi.

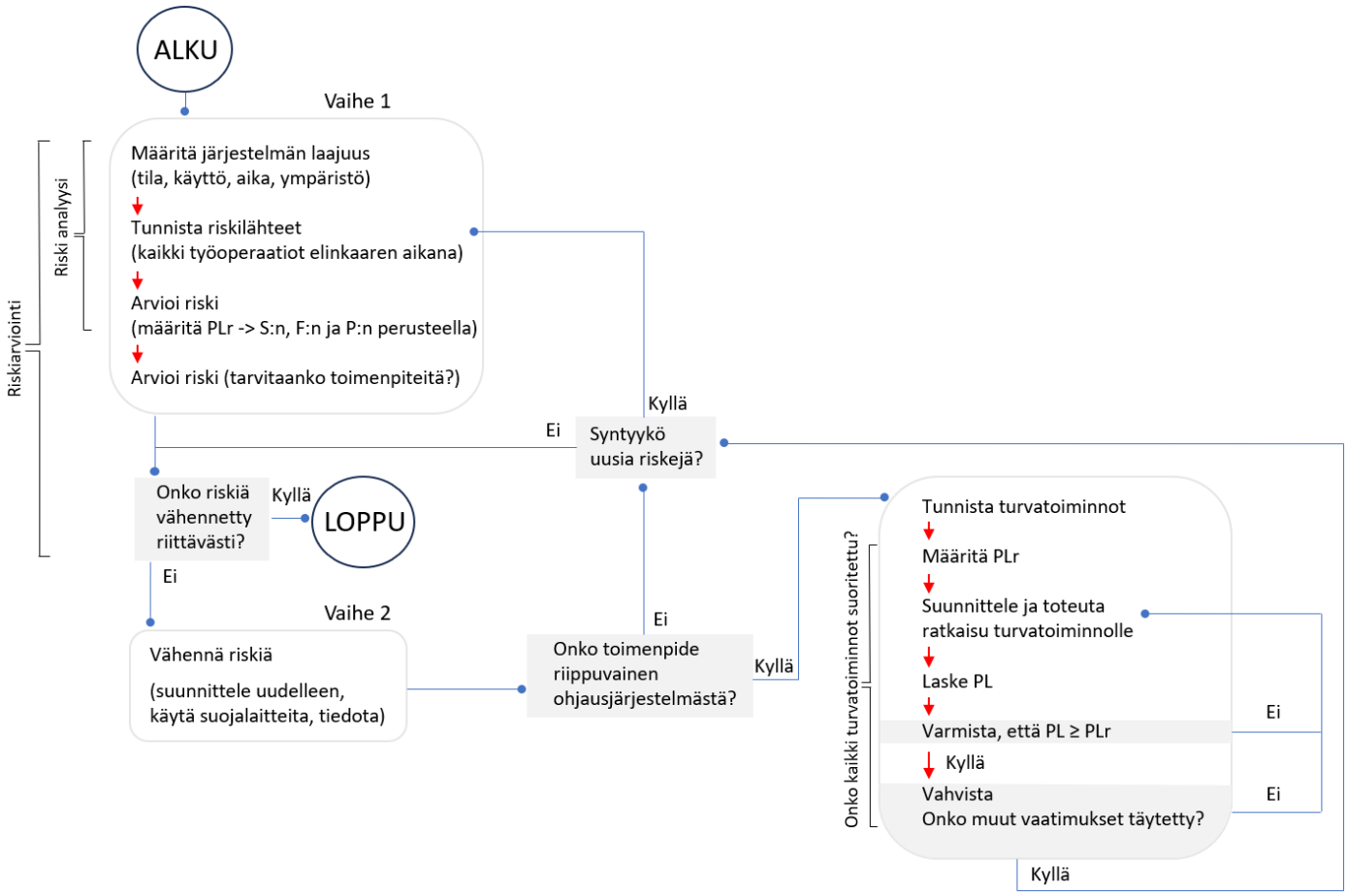
HAZOP analyysin vaiheet
1. Valitse solmukohta (node), määrittele sen tarkoitus ja määritä prosessin turvarajat.
2. Valitse keskustelua ohjeistavat kuvaukset solmukohdille.
3. tunnistaa vaarat ja niiden syyt hyödyntäen solmukohdan kuvauksia.
4. Selvitä miten vaaratilanne ilmoitetaan, eli miten operaattori saa tiedon siitä kun turvaraja ylittyy.
5. Arvioi jokaisesta vaaratilanteesta aiheutuvat seuraukset.
6. Tunnista tarvittavat suojavälineet.
7. Arvioi kuinka usein vaaratilanne toistuu.
8. Arvioi vaarallisuuden aste suojalaitteilla ja ilman suojalaitteita.
9. Tee havaintoja ja ilmoita potentiaalisista tutkimustuloksista.
10. Siirry seuraavaan solmukohtaan, kun solmukohdan kaikki kuvaukset on käyty läpi.

Esimerkki HAZOP analyysin vaiheista kuvauksella ”korkea virtaus”
1. mikä on ”korkean virtauksen" määrällinen määritelmä (eli mikä on virtausnopeuden turvallinen yläraja kyseissä solmukohdassa)?
3. Mistä ”korkea virtaus” johtuu?
4. Millä keinoin ilmoitetaan operaattorille, että virtaus on ”liian korkealla”?
5. Millaisia (turvallisuus, taloudellisia, ympäristö) seurauksia syntyy ”korkeasta virtauksesta”?
6. Mitä turvalaitteita on asennettu estämään ”korkeaa virtausta”?
7/8. Kuinka usein ”korkea virtaus” syntyy suojalaitteilla ja ilman suojalaitteita?
9. Onko tiimi saanut tutkimustuloksia, pystyykö tiimi antamaan suosituksia?

KUVIO 8. HAZOP-analyysin vaiheet (mukaillen Sutton 2010, 133–139)

2.3 Turvasuunnittelu standardeilla PL & SIL

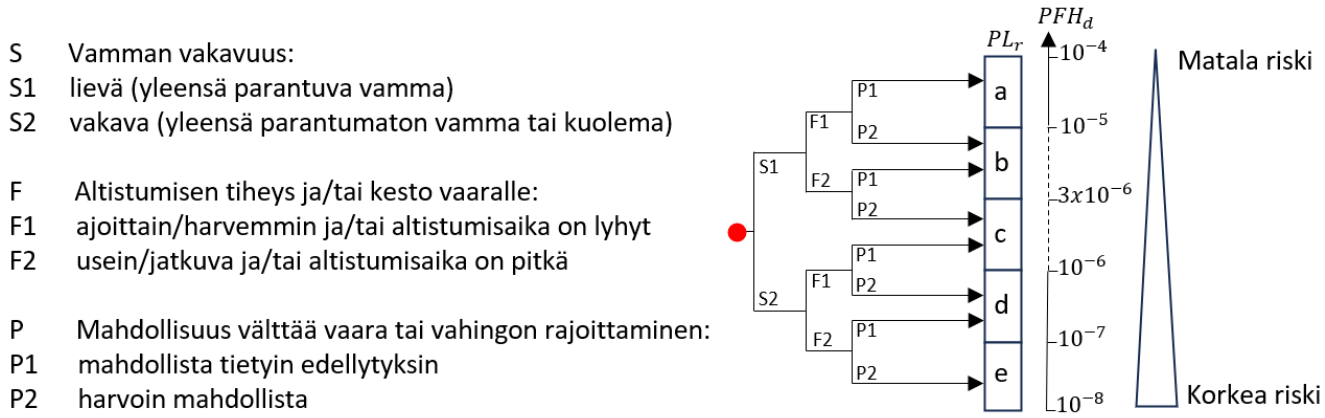
Ohjaamaan turvallista laitesuunnittelua käytetään standardia EN ISO 13849-1 (PL, Performance Level) ja ohjausjärjestelmiä ohjaa standardi EN IEC 62061 (SIL, Safety Integrity Level). Ensiksi tutkimme standardia ISO 13849-1 ja tämän jälkeen standardia IEC 62061. Mainitut standardit ovat osittain verrattavissa keskenään. Kuvio 9 esittää yleisellä tasolla, miten suunnittelu etenee.



KUVIO 9. ABB-riskienarviointimenetelmä ohjelmoitaessa (mukaillen ABB Jokab Safety 2019, 12)

PL (Performance Level) on teknologianeutraali käsite, jota voidaan käyttää sähköisissä, mekaanisissa, pneumaattisissa ja hydraulisissa turvallisuusratkaisuissa. PL tai PFH_d (probability of dangerous failure per hour) laskelma mittaa turvatoimintojen luotettavuutta asteikolla (a-e). Esimerkiksi PL=e antaa parhaan luotettavuusluokituksen, jota vaaditaan, kun toimitaan korkeimmalla riskitasolla. PL:n laskemiseen on olemassa ohjelmia, kuten esimerkiksi FSDT tai SISTEMA. Tietyt laitevalmistajat ilmoittavat laitetiedoissaan valmiiksi lasketut arvot PL:lle tai PFH_d:lle, kuten esimerkiksi ABB. (ABB Jokab Safety 2019, 11.)

Suorittaessa riskiarviointia (PL_r) ensin määritellään, kuinka työkone tulee toimimaan. Työkoneen toimintaa määriteltäessä meidän pitää huomioida työtilan koko ja arvioida työkoneen kaikki operatiiviset toiminnot tämän elinkaaren aikana. Työkoneen elinkaaren aikana kaikki käyttöön liittyvät riskilähteet pitää arvioida. Kuvio 10 toimii apuna eri riskilähteitä arvioidessa. (ABB Jokab Safety 2019, 13.)



KUVIO 10. Riskinarviointi (PL_r) ja riskitason määrittäminen (ABB Jokab Safety 2019, 13–14)

Jokainen riskilähde pitää arvioida erikseen ja antaa arvio riskin suuruudesta. Standardin EN ISO 13849-1 mukaisesti riskit arvioidaan käyttämällä kolmea tekijää: vamman vakavuus (S), kuinka usein vaara toistuu (F) ja mahdollisuudet vamman välttämiseen tai rajoittamiseen (P). Jokaiselle tekijälle annetaan kaksi vaihtoehtoa. Standardi ei kuitenkaan anna selvää rajaa kahden vaihtoehdon välille, mutta seuraavat tulkinnat ja suositukset ovat yleisiä. (ABB Jokab Safety 2019, 13.)

- S1 mustelmat, naarmut, puhkaisuhaavat ja lievät murskavammat
- S2 luustovammat, amputaatiot ja kuolema
- F1 harvemmin kuin kerran viikossa
- F2 kerran viikossa tai useammin
- P1 hitaat koneliikkeet, riittävästi tilaa, alhainen teho
- P2 nopeat koneliikkeet, ahdas tila, suuri teho (ABB Jokab Safety 2019, 13.)

Kun S, F ja P on valikoitu, saadaan arvo PL_r, jota käytetään riskilähde arvioinnissa (KUVIO 10). Kun kaikki riskit ovat tiedossa voidaan suorittaa lopullinen riskinarviointi ja päättää pitääkö riskiä vähentää vai onko riittävä turvallisuuden aste saavutettu. (ABB Jokab Safety 2019, 13.)

SIL

SIL (Safety Integrity Level) riskinarviointimenetelmää käytetään lähestulkoon ainoastaan prosessiteollisuudessa. Koska SIL-menetelmä soveltuu pelkästään käytettäväksi sähköisiin, elektronisiin tai ohjelmoitaviin turvallisuusratkaisuihin (ABB Jokab Safety 2019, 11.). SIL-arviointiasteikko toimii neljällä tasolla standardin IEC 61508-1 mukaan, mutta standardissa IEC 62061 maksimitaso on laskettu kolmeen (SFS-EN IEC 62061, 102). Esimerkki havainnollistaa sen, miten SIL-luokitukset arvioidaan ja mikä yhteys SIL-luokituksella on PL:n (KUVIO 11).

<table>
<tr><th rowspan="2">Seuraukset</th><th rowspan="2">Vakavuus (Se)</th><th colspan="13">Luokka Cl = Fr + Pr + Av</th></tr>
<tr><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td rowspan="2">Kuolema, silmän tai käden menetys</td><td rowspan="2">4</td><td colspan="2">SIL 1</td><td colspan="3">SIL 2</td><td colspan="3">SIL 2</td><td colspan="3">SIL 3</td><td colspan="2">SIL 3</td></tr>
<tr><td>$PL_r b$</td><td>$PL_r c$</td><td colspan="3">$PL_r d$</td><td colspan="3">$PL_r d$</td><td colspan="3">$PL_r e$</td><td colspan="2">$PL_r e$</td></tr>
<tr><td rowspan="2">Pysyvä vamma, sormien menetys</td><td rowspan="2">3</td><td colspan="5" rowspan="3">Ei SIL (tai PL) vaatimuksia</td><td colspan="3">SIL 1</td><td colspan="3">SIL 2</td><td colspan="2">SIL 3</td></tr>
<tr><td colspan="3">OM
$PL_r a$</td><td>$PL_r b$</td><td colspan="2">$PL_r c$</td><td colspan="3">$PL_r d$</td><td colspan="2">$PL_r e$</td></tr>
<tr><td rowspan="2">Tilapäinen vamma, tarvitaan terveydenhuollon apua</td><td rowspan="2">2</td><td colspan="5" rowspan="2"></td><td colspan="3">OM</td><td colspan="2">SIL 1</td><td colspan="2">SIL 2</td></tr>
<tr><td colspan="3">$PL_r a$</td><td>$PL_r b$</td><td>$PL_r c$</td><td colspan="2">$PL_r d$</td></tr>
<tr><td rowspan="2">Tilapäinen vamma, ensiapu</td><td rowspan="2">1</td><td colspan="8" rowspan="2"></td><td colspan="3">OM</td><td colspan="2">SIL 1</td></tr>
<tr><td colspan="3">$PL_r a$</td><td>$PL_r b$</td><td>$PL_r c$</td></tr>
</table>

KUVIO 11. SIL-luokitukset ja seurausten arviointi (mukaillen SFS-EN IEC 62061, 102)

Jotta voimme laskea luokan Cl (3–15), pitää ensiksi selvittää arvot muuttujille Fr, Pr ja Av (KUVIO 11). Muuttujien arvot vaihtelevat pääosin välillä 1–5, jossa ykkönen kuvastaa pienentä turvallisuusriskiä.

Fr arvioi altistumistaajuutta, eli kuinka usein vaara toistuu ja kuinka kauan kyseinen vaaratilanne kestää. Jos altistumisen kesto on vähemmän kuin 10 minuuttia, käytetään arviointiasteikkoa 1–5, muuten käytetään tiukempaa arviointiasteikkoa, jossa arviointiasteikko esitetään näin: 2, 3, 4, 5 ja 5. (SFS-EN IEC 62061, 99.)

Pr arvioi, kuinka todennäköisesti vaaratilanne tapahtuu. Tämä arviointi perustuu koneenosien käyttäytymiseen ja ennustettavuuteen eri toimintatiloissa. Arvioinnissa otetaan myös huomioon ihmisten määritelty tai ennakoivissa oleva käyttäytyminen vaaran lähteenä, kun he ovat tekemisessä koneenosien kanssa. Arviointi suoritetaan asteikolla 1–5. (SFS-EN IEC 62061, 100.)

Av koittaa arvioida, voiko vahingon välttää, tai voidaanko sitä ainakin rajoittaa vaarallisen tapahtuman sattuessa. Vaaratilanteita voidaan esimerkiksi tunnistaa fysikaalisten ominaisuuksien perusteella, tai vaaratilanteita voidaan havaita pelkästään käyttämällä teknisiä ilmaisimia. Vahingon hallittavuutta (rajoittamista) määrittää enimmäkseen ihmisen puuttuminen tilanteeseen, ja puuttumisesta saatavat tulokset riippuvat aika lailla pelkästään yksittäisten ihmisten kyvyistä. Arviointi suoritetaan asteikolla 1, 3 ja 5. (SFS-EN IEC 62061, 101.)

Tärkeä osa prosessiturvallisuuden hallintaa on toiminnallisen turvallisuuden hallinta. Standardien IEC 61508 ja IEC 61511 mukaisesti toiminnallinen turvallisuuden hallinta varmistaa sen, että laitoksen kriittiset suojakerrokset ovat jatkuvassa toiminnassa ja valmiina suojelemaan henkilöstöä vaaratilanteiden sattuessa. (What is the Difference Between PHA, HAZOP & LOPA? 2022, luku ”The Role of the Safety Integrity Level (SIL)”.)

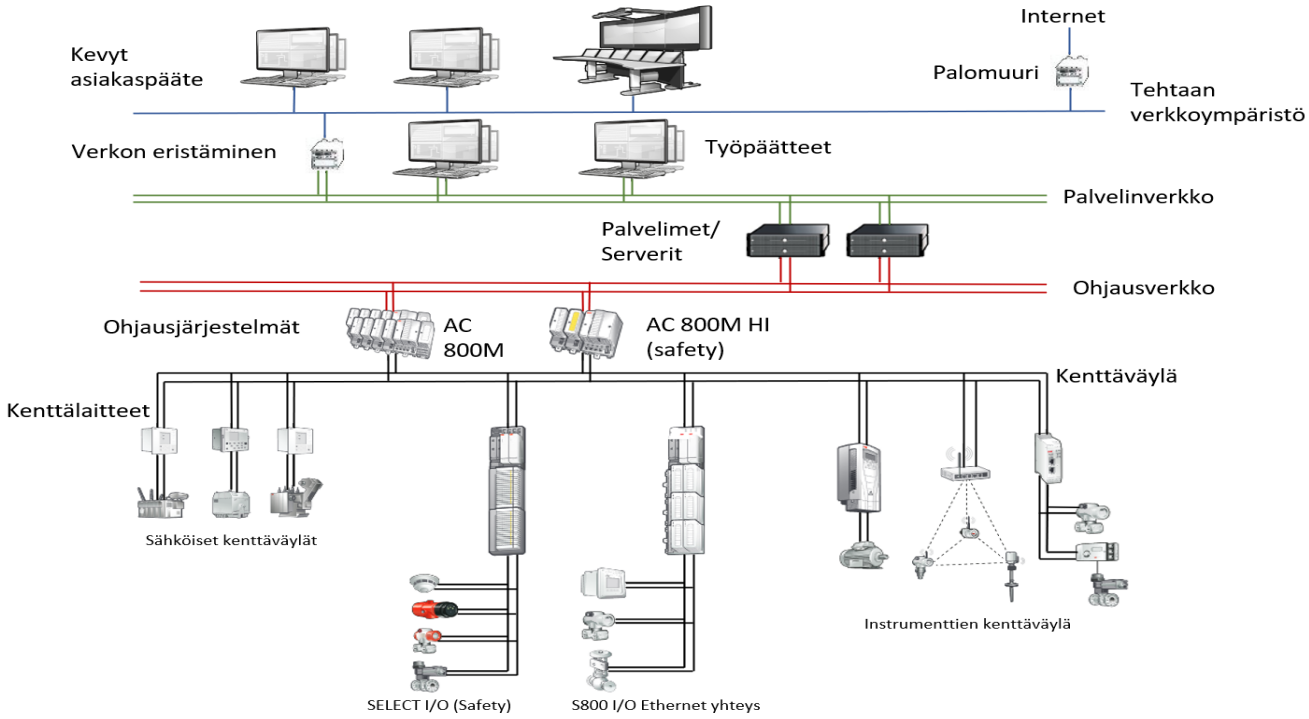
Osana toiminnallista turvallisuuden hallintaa jokaisella kriittisellä toiminolla pitää olla myös turvatoiminto, tätä toimintoa kutsutaan SIF:ksi (Safety Instrumented Function). SIF:llä pitää olla SIL-taso, joka määrittää kuinka paljon riskiä voidaan vähentää. SIL-laskelman pitää määrittää vaaran aiheuttama riski ilman SIS-järjestelmän (Safety Instrumented System) tuomaa riskin vähennystä. Laskelmaa kutsutaan ”vähentämättömäksi” riskiksi. Vähentämättömän riskin esittämää lukua verrataan tavoiteltavaan siedettävän riskintasoon. Jos vähentämätön riski on suurempi kuin siedettävä riski, se on käsiteltävä SIF:in kautta, joka on osa SIS-järjestelmää. Kun pidetään huolta siitä, että jokaisella SIF:llä on aktiivisesti hallittu SIL, saavutetaan kokonaisvaltainen prosessiturvallisuuden hallinta. (What is the Difference Between PHA, HAZOP & LOPA? 2022, luku ”The Role of the Safety Integrity Level (SIL)”.)

3 JÄRJESTELMÄT JA LAITTEISTO

Vaikka CABB käyttääkin omaa versiota ABB 800xA-järjestelmästä, suurimmaksi osaksi heidän järjestelmänsä toimii täysin samalla tavalla kuin tavallinenkin versio. Tästä syystä voimme käyttää tavallisia ABB:n tarjoamia manuaaleja havainnollistamaan ohjelmiston toimintaa yleisellä tasolla. Opinnäytetyönkannalta tärkeä osio on ABB 800xA-ohjelmisto, koska se liittyi hyvin oleellisesti opinnäytetyön tekemiseen. Samalla käydään tarkemmin läpi työssä käytettyjä ohjelmointikieliä. Tässä osiossa sivutaan myös hiukan sitä, miltä 800xA-laitteisto näyttää, ja käydään lyhyesti läpi toiminnan perusperiaatteet. Lisäksi tässä osiossa käsitellään lyhyesti myös opinnäytetyössä käytettyä ohjelmistoa PHA-pro.

3.1 ABB 800xA-laitteisto

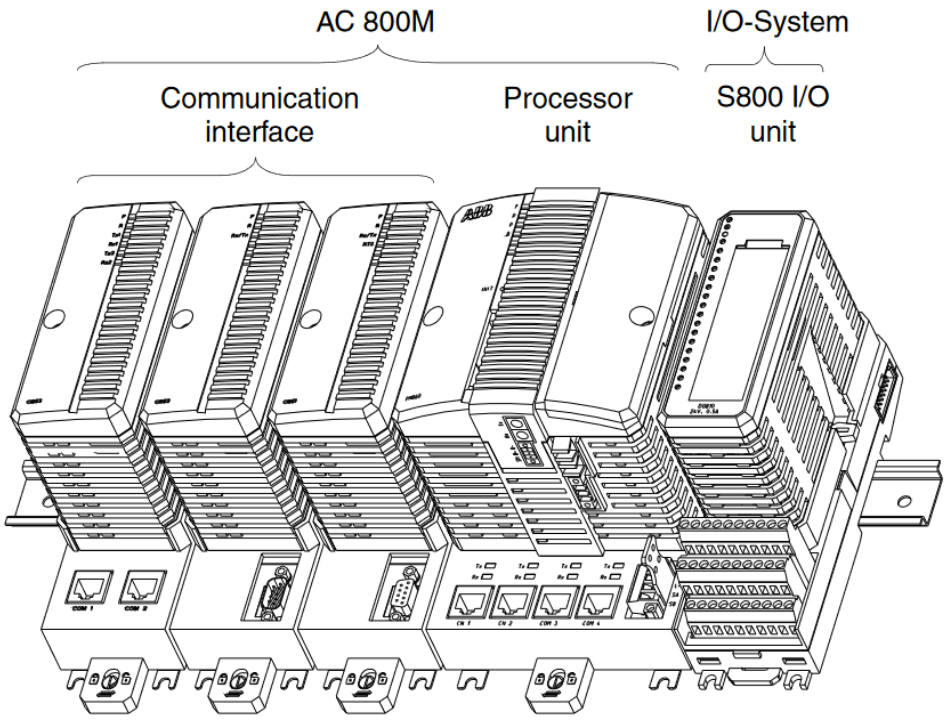
ABB 800xA on hajautettu ohjausjärjestelmä, joka toimii myös sähköisenä ohjausjärjestelmänä, turvajärjestelmänä ja yhteistyön mahdollistajana. Tämä yhdistetty ympäristö mahdollistaa tehtaiden yksinkertaistetun ohjelmistoesityksen. Tämä ohjelmistoesitys kattaa on/off tyyppiset kytkimet ja venttiilit, älykkäät kenttälaitteet, yksilöidyt ohjausjärjestelmät, taajuusmuuttajat, älykkäät kytkentälaitteet, suojareleet (IED) ja PC-pohjaiset valvontajärjestelmät. (Control and I/O Overview 2019, 4.) Seuraava esimerkki havainnollistaa sen, miten kaikki laitteet ja toiminnot yhdistyvät toisiinsa hyvin suunnitellun arkkitehtuurin ansiosta (KUVIO 12).



KUVIO 12. ABB 800xA-järjestelmän verkkorakenne (mukaillen System Planning 2016, 53)

Kuten kuviosta 12 huomataan, järjestelmä hoitaa kaikki vaaditut toiminnalliset vaiheet alusta loppuun asti saman verkkojärjestelmän alla. Kuvio 12 on tehty kahdentamalla, koska tämä on tehokas toiminta-tapa paikoissa, joissa prosessien seisokit tulevat hintaviksi. Kahdentaminen varmistaa sen, että meillä on ympäristö, jossa voidaan suorittaa ohjelmointia ja simulointia prosessin ulkopuolella. Ohjelmalliset virheet ovat helposti korjattavissa, koska meillä on ohjelmiston edellinen versio tallessa toisella palvelimella. Vikojen paikantaminen helpottuu ja saadaan minimoitua ajattavan prosessin seisokki aika.

AC 800M voidaan määritellä laitteistoalustaksi, joka koostuu erillisistä laiteyksiköistä. AC 800M ohjainjärjestelmä koostuu seuraavista laiteyksiköistä: prosessori (CPU), kommunikointirajapinnasta eri protokollien välille, virtalähdeyksiköstä eri tehoisille ulostuloille ja varavirtalähteestä (KUVIO 13). Kun ohjainjärjestelmään liitetään sopiva ohjelmisto, sitä voidaan käyttää monenlaisissa prosesseissa ja automaatioratkaisuissa. Ohjainohjelmistolla AC 800M sopii vaikka prosessin ohjaimeksi tai sillä voidaan esimerkiksi suorittaa paikallisia säätötehtäviä, jotka sitten yhdistyvät ohjausverkon kautta useisiin työpäätteisiin. (AC 800M Controller Hardware Product Guide 2019, 17.)



KUVIO 13. AC 800M-ohjausjärjestelmä ja S800 I/O-yksikkö (AC 800M Controller Hardware Product Guide 2019, 18)

Seuraavaksi tutkitaan muutamia AC 800M-yksikön tärkeimpiä ominaisuuksia. Näitä ominaisuuksia ovat: korkea suorituskyky ja suuri sovellusmuisti, AC 800M on SIL2 sertifioitu, AC 800M HI on SIL3 sertifioitu, sisäiset Ethernet kanavat, sisäiset RS-232C kanavat, jaettu CEX-väylä käyttäen BC810/BC820, sisäinen virtalähde muistille, ulkoinen varavirtalähde ja virtalähde 24 v DC teollisuuden standardien mukaisesti. (AC 800M Controller Hardware Product Guide 2019, 20.)

Seuraavaksi katsotaan tuettuja viestintäprotokollia, joihin kuuluu: PROFIBUS DP, FOUNDATITION Fieldbus, Advant Fieldbus 100, Modbus TCP, Field Device Tool (FDT), Device Type Manger (DTM), INSUM yhteys porttien (Ethernet/LON) kautta, standardi IEC 61850, PROFINET IO, EtherNet/IP, S100 I/O, Satt I/O, TRIO/Genius ja MasterBus 300. (AC 800M Controller Hardware Product Guide 2019, 21.)

3.2 ABB 800xA-ohjelmisto

Käyttäessä ABB 800xA-ohjelmistoa on hyvä huomata se, että kyseinen järjestelmä toimii yhdistelemällä erilaisia ”ohjelmointimalleja” ohjelmoinnin eri vaiheissa. Tämä tarkoittaa sitä, että ohjelmointitapa poikkeaa merkittävästi perinteisestä tekstipohjaisesta ohjelmoinnista. ABB:n ohjelmointimalli perustuu olio-ohjelmointiin, jossa ohjelmoitava kokonaisuus muodostuu ”Aspect Object” ajattelutavan kautta (KUVIO14). Tuettuja ohjelmointikieliä 800xA-järjestelmässä (standardi IEC 61131-3): Instruction List, Structured Text, Function Block Diagram, Sequential Function Chart ja Ladder Diagram.

Olio-ohjelmointi eroaa perinteisestä ohjelmoinnista, koska sen rakenne ja toteutustapa ovat erilaisia. Perinteistä ohjelmointia suunnitellessa pitää ottaa huomioon se, että ohjelma lukee ja suorittaa koodia ylhäältä alaspäin. Jos ohjelmaan tehdään merkittäviä muutoksia tämä yleensä, meinaa sitä, että koodia on kirjoitettava uudelleen. Ohjelmointia suorittaessa pitää myös ymmärtää toiminnon tyyppi ja suoritussykli, jotta oikea toiminto tapahtuu oikealla hetkellä. (AC 800M Planning 2016, 16.)

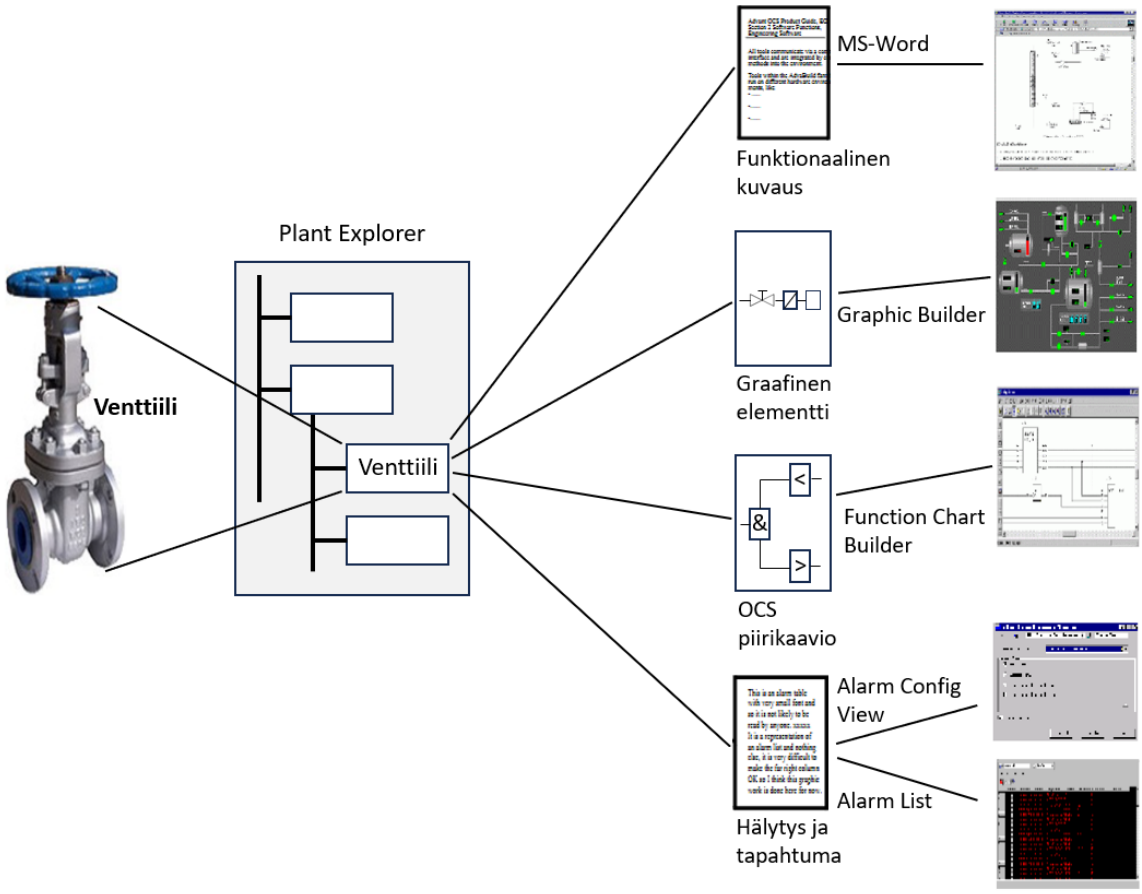
Perinteisessä ohjelmoinnissa keskitytään hahmottamaan ohjelmoitava kokonaisuus, kuten esimerkiksi ”tuota sementtiä”. Olio-ohjelmointi vastaavasti jakaa tehtävään liittyvät ohjelmointivaiheet erillisiksi toiminnallisiksi lohkoiksi, joka helpottaa isojen kokonaisuuksien hallitsemista. Näitä toiminnallisia lohkoja on esimerkiksi toiminnot ”säädä moottorin nopeutta” tai ”avaa vesiventtiili”, näitä lohkoja kutsutaan objekteiksi (object). Kun moottorin nopeutta säädetään, moottorista itsestään ei tarvita tietoja,

sama pätee venttiiliin. Jokaisella objektilla on ulostulo ja sisääntulo, joita objekti analysoi, jonka jälkeen objekti tekee tarvittavat päätökset, joka johtaa joko toimintaan tai ulostuloon. Objektin tehdessä analyysiä objekti ei tarvitse järjestelmän ulkopuolista syötettä. (AC 800M Planning 2016, 17.)

Kun suunnitellaan oliopohjaisia ratkaisuja, on hyvä huomata, että objektien väliset riippuvuudet sijoitetaan objektien ulkopuolelle. Esimerkiksi jos moottori saa käskyn käynnistyä ”säätö” -objekti ei pysty käynnistämään moottoria, mutta ”säätö” -objekti pystyy pysäyttämään moottorin, jos nopeus nousee liian suureksi. Moottorin tavallinen pysäytyskäsky on myös ”säätö” -objektin ulkopuolella, koska kyseinen objekti huolehtii vain moottorin nopeuden säätämisestä ja moottorin pysäyttämisestä vikatilanteissa tai hätätapauksissa. (AC 800M Planning 2016, 17.)

Oliopohjaisessa suunnittelussa mallintamisympäristöstä on tehty yksinkertainen käyttää. Objektien suunnittelu kasvattaa jatkuvasti uudelleen käytettävien työkalujen määrää. Uudet työkalut helpottavat luomaan suurempia ja entistäkin monimutkaisempia kokonaisuuksia tehokkaasti. Kun käytettäviä työkaluja on kasaantunut tarpeeksi, voidaan nämä kehitetyt työkalut muuttaa helposti käytettäväksi kirjastoksi, joka tehostaa suunnittelua entisestään. Oliopohjainen suunnittelu on aluksi todella työlästä ja vie paljon aikaa, koska jokainen objekti on analysoitava huolellisesti ja suunniteltava siten, että se voi pysyä erillisenä, itsenäisenä ja uudelleenkäytettävänä objektina. (AC 800M Planning 2016, 17.)

Olio-ohjelmointia käytetään ainoastaan toimintolohkoissa, ohjausmoduuleissa ja kaavioissa. Tietyt tekniset seikat johtavat siihen, että jokin toinen tekniikka toimii ohjelmoitaessa toista tekniikkaa paremmin. Esimerkiksi ohjausmoduuli ja toimintolohko eroavat jonkin verran toisistaan, joka johtuu siitä, että ohjausmoduulin koodilohkot järjestetään automaattisesti optimaalisimpaan koodin suoritus järjestykseen, joka luonnollisesti johtaa parempaan suorituskykyyn. Jos käytetään kaavioita, on mahdollista saada koko projekti suunniteltua yhteen kaavioon, koska kaaviomalli mahdollistaa ohjausmoduulien ja toimintolohkojen sekoittamisen keskenään graafisten yhteyksien avulla. (AC 800M Planning 2016, 18.)



Käyttölaite	Object (Aspect Object)		Aspects	Views (näkymät)

KUVIO 14. Esimerkki ”Aspect Object” toimintamallista (mukaillen System Planning 2016, 310)

Kuvio 14 antaa hyvän yleiskäsityksen siitä, miten 800xA-järjestelmä toimii, ja esittää selvästi miten eri ohjelmointivaiheet linkittyvät toisiinsa projektin edetessä. Seuraavaksi käydään lyhyesti läpi mitä kuvion 14 esittämät objektit, aspektit ja näkymät tarkoittavat käytännössä.

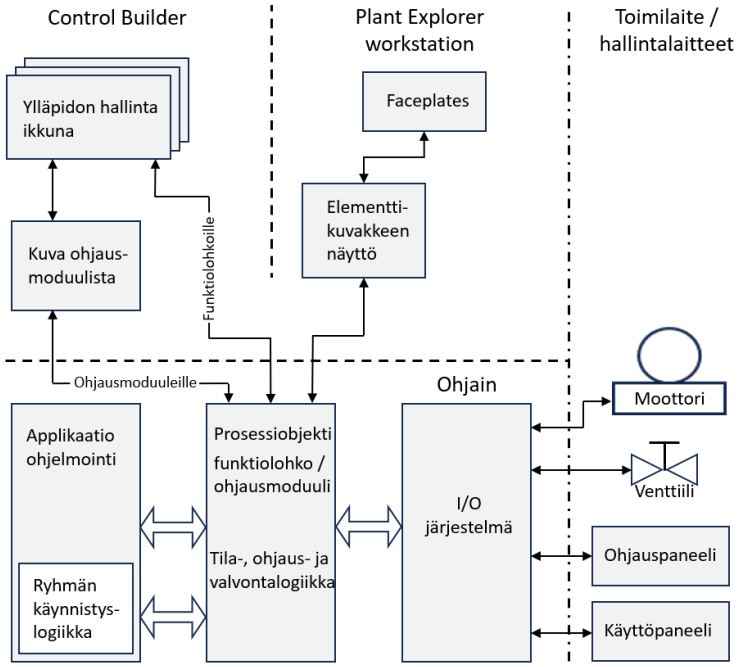
ABB:n 800xA-järjestelmässä ”Aspect Object” edustaa fyysisiä käyttölaiteita eli objekteja (object). Objektit voivat olla esimerkiksi fyysisiä ohjaimia tai I/O-kortteja. Meillä on myös virtuaalisia objekteja, kuten esimerkiksi erilaiset toiminnot ja prosessiobjektit. (System Planning 2016, 307.)

Aspekti (aspect) on tietojoukko, joka kuvaa tiettyjä objektin ominaisuuksia. Objekteilla on useita erilaisia aspekteja. Objektien ominaisuudet (joissa on suunnittelun tai käyttöajan tietoja) määräytyvät aspektin ominaisuuksien perusteella. Aspektien ominaisuuksiin kuuluu esimerkiksi toiminnalliset ominaisuudet, fyysisen rakenteen ominaisuudet, sijaintiominaisuudet ja ohjausominaisuudet. (System Planning 2016, 309.)

Näkymät (views) esittää aspektitietojen työkaluja omissa ikkunoissaan. Yhdellä aspektilla voi olla useita näkymiä. Kun aspektinäkymä avataan, pääsee käsiksi aspektijärjestelmään. Aspektijärjestelmässä olevilla työkaluilla voidaan tarkastella, luoda, muokata tai poistaa aspektiin liittyviä tietoja. Tietoja voidaan esittää esimerkiksi: listoina, taulukkoina, kaavioina (piirikaavio), piirustuksina, grafiikkana tai toimintakaaviona. (System Planning 2016, 309.)

3.2.1 Ohjelmointirakenne

Tässä vaiheessa on hyvä tutkia myös CABB:n omia dokumentteja liittyen ABB 800xA-järjestelmän ohjelmointiin, koska heidän järjestelmänsä poikkeaa jonkin verran esimerkiksi applikaatio ohjelmointiin liittyvissä rakenteissa. Kuvio 15 esittää tyypillisen 800xA-järjestelmän rakenteellisia tiedonsiirto yhteyksiä.



KUVIO 15. Ohjausmoduulien ja funktiolohkojen yhteys käyttölaitteisiin (mukaillen AC 800M Binary and Analog Handling 2016, 294)

Suurin ero CABB:n ja tavallisen 800xA-järjestelmän välillä on kirjastot. CABB:n käytössä on kirjastoja, jotka on kehitetty vastaamaan yrityksen tarpeita. Kirjastot määrittelevät prosessin hallintaan tarkoitettuja työkaluja, joita voidaan soveltaa erinäisten toimilaitteiden hallinnassa. Suurimmaksi osaksi

kaikki prosessin hallintaan liittyvät toiminnot määritellään kaavioihin. Luodut kaaviot ovat yhteydessä toisiinsa ohjelmassa määritettyjen tietoyhteyksien avulla. (Engineering Guidelines 2017)

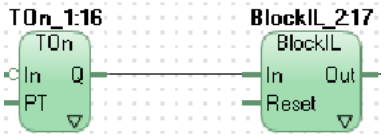
CABB:lle räätälöity 800xA-ohjelmisto toimii hyödyntämällä FD:tä (Function Diagram), SFC:tä (Sequential Function Chart) sekä ST:tä (Structured Text). 800xA-järjestelmä tukee viittä ohjelmointikieltä standardin IEC 61131-3 mukaisesti, mutta CABB:n käyttämä ohjelmistomalli pystyy hyödyntämään vain kahta viidestä ohjelmointikielestä, koska käytetty FD-malli tukee vain SFC:tä ja ST:tä. Tämä FD-mallin luoma rajoite on mainittu epäsuorasti CABB:n manuaalissa. (Engineering Guidelines 2017, 12.) Suorempi vastaus aiheeseen löytyy ABB:n omasta manuaalista (AC 800M Planning 2016, 120). On silti hyvä huomata, että CABB:n ohjelmisto pystyy edelleen tukemaan viittä ohjelmointikieltä FD-mallin lisäksi, mutta silloin ohjelmat pitää rakentaa eri tavalla (Engineering Guidelines 2017, 12). Toisaalta CABB:n käytettävissä olevat ohjelmointikielet ovat ABB:n mielestä voimakkaimmat ohjelmointikielet (pois lukien SFC), joten rajoituksista huolimatta funktionaalisuus jää hyvälle tasolle (AC 800M Planning 2016, 57).

CABB:n FD-mallia käytetään prosessin ohjauslogiikan ja turvalogiikan ohjelmointiin. Ohjauslogiikkaa ja turvalogiikkaa tuetaan projektikohtaisesti ST:n avulla. Eräajo ohjelmointi on toteutettu hyödyntämällä SFC ohjelmointikieltä. (Engineering Guidelines 2017, 12.)

3.2.2 Function Diagram

Funktiokaavio (Function Diagram (FD)) on graafinen kieli, joka mahdollistaa erilaisten funktioiden, toimintolohkojen, ohjausmoduulien ja kaavioiden yhdistämisen yhdeksi koodilohkoksi ja niiden välille voidaan luoda graafisia yhteyksiä. FD soveltuu ohjelmointityökaluksi SIL-tasoille (1–3) ja ei SIL-luokitusta vaativiin töihin. FD hyödyntää pääasiallisesti kolmea elementtiä, joita käytetään logiikan luomisessa: sivut (pages), kutsulohkot (objects) ja tietoyhteydet (Data connections). Funktiokaaviot jaetaan kahteen kaaviotyyppiin ”Diagram ja Diagram Type” (KUVIO 17). Kaavio (diagram) voi sisältää useita erilaisia kaaviotyyppejä (diagram type), mutta kaaviotyyppi voi sisältää pelkästään muita kaaviotyyppejä. (AC 800M Planning 2016, 58, 60, 31.) Esimerkiksi CABB:n järjestelmässä kaavio ”uCumulus (Batch unit)” sisältää kaikki kyseisen applikaation liittyvät kaaviotyypit (KUVIO 17) (Engineering Guidelines 2017, 22).

FD:n sivuelementti esittää yhtä sivua kaaviosta tai kaaviotyypistä. Nämä sivut sisältävät objekteja ja tietoyhteys elementtejä. Sivut numeroidaan alkaen ykkösestä. Sivunumerot ovat yksilöllisiä yksittäisissä kaavioissa ja kaaviotyypeissä. Kaavioissa käytetään kutsulohkoja esittämään suoritettavan yksikön kutsua, kuten esimerkiksi koodilohkoa. Lohkon nimi esitetään ennen ”:” ja sen perään tulee datavirran järjestysnumero. Lohkoissa esiintyvät liitäntäpisteet toimivat tietoyhteyksien päätepisteinä. Lohkoissa esiintyvät liitäntäpisteet riippuvat lohkon tyypistä. Lohkon liitäntäpisteet esittävät: vastaavan parametrin nimen, vastaavan parametrin tietotyypin, suunnan (In, Out, In_Out tai Unspecified) ja onko liitäntäpiste invertoitu vai ei. (AC 800M Planning 2016, 60–61.)

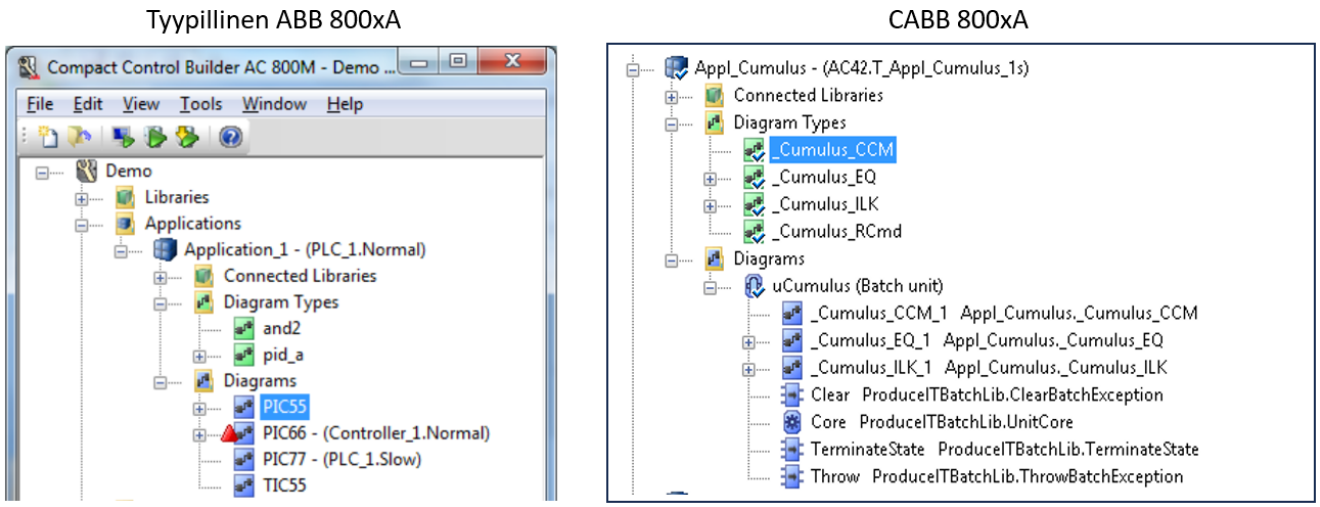


KUVIO 16. Esimerkki liitäntäpisteistä, lohkojen graafisista yhteyksistä ja yhteyden invertoinnista (TOn)

Kaaviotyyppi luokitellaan POU:ksi (Program Organization Unit), joka sisältää parametrejä, muuttujia ja koodia. Tämä perustuu objektin yksi tyyppisuunnitteluun, joka tarkoittaa sitä, että niitä kuvataan tyyppien mukaan. Objektityypistä luodaan tapahtumia, jotka toimivat täysin samalla tavalla kuin itse tyyppi. Jokaisella instanssilla (objektilla) on oma muistiesitys muuttujille ja parametreille. Kun objektia kutsutaan, kutsutaan juuri itse objektia eikä sen perustyyppiä. Kun koodia suoritetaan, se on jaettu kaikkien objektien kesken ja kuuluu objektityyppiin. Muistin säästämiseksi olisi hyvä käyttää objektityyppejä useiden objektien lähteenä. (AC 800M Planning 2016, 120.)

Jokaisella objektilla on uniikki datavirran järjestysnumero (Data Flow Order), joka annetaan ja esitetään automaattisesti. Datavirran numero määräytyy objektien järjestyksen mukaan vasemmalta oikealle. Koodia ajettaessa seurataan datavirran numeroa pienimmästä suurimpaan. Jos objektiin on tehty graafinen yhteys ja tämän jälkeen sijaintia muutetaan, niin yhteydet analysoidaan uudelleen, ja lähdeobjekti saa alhaisimman datavirta numeron. Tämän vuoksi olisi hyvä sijoittaa ensimmäiseksi suoritettavat objektit ikkunan vasempaan laitaan. Jos koodissa on liikaa graafisia yhteyksiä ja kuvaa on vaikea ymmärtää, koodia voidaan selventää käyttämällä ST ja SFC koodilohkoja. (AC 800M Planning 2016, 121.)

Lopuksi tarkastellaan eroja tavallisen 800xA järjestelmän ja CABB:n järjestelmän välillä.



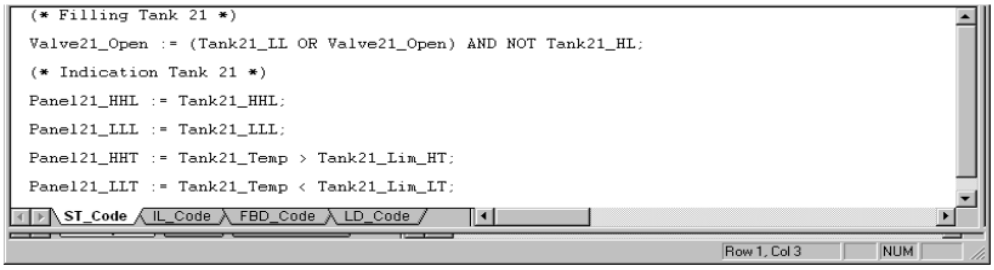
KUVIO 17. järjestelmien välisiä eroja (Steiner 2013, 9)

Suurin ero tyypillisen- ja CABB:n 800xA-järjestelmän välillä on kirjastoissa. Kuviossa 17 ”uCumulus” kaavion alla näkyy pätkiä kirjastoista (Lib.), jotka on automaattisesti liitetty osaksi ohjelmointia CABB:n järjestelmässä, jotta järjestelmässä saadaan käyttöön CABB:n vaatimaa funktionaalisuutta.

Muita havaittavia eroja järjestelmien välillä on esimerkiksi rajoitusten määrissä. Tyypillisessä järjestelmässä kaaviot on rajoitettu 200 sivuun ja 50 objektiin. Kaaviotyypeille ei ole asetettu numeraalisia rajoituksia (AC 800M Planning 2016, 30, 32.). CABB:n järjestelmässä kaaviot on rajoitettu 100 sivuun/kaavio, 30 objektiin/sivu ja 32 viestintämuuttujaan/kaavio. Kaaviotyypeille on asetettu seuraavat sivukohtaiset rajoitukset: venttiilit 10/sivu, moottorit 10/sivu, digitaaliset tulot 20/sivu, analogiset tulot 20/sivu ja yksi analoginen silmukka per sivu. (Engineering Guidelines 2017, 22, 14.)

3.2.3 Structured Text & Sequential Function Chart

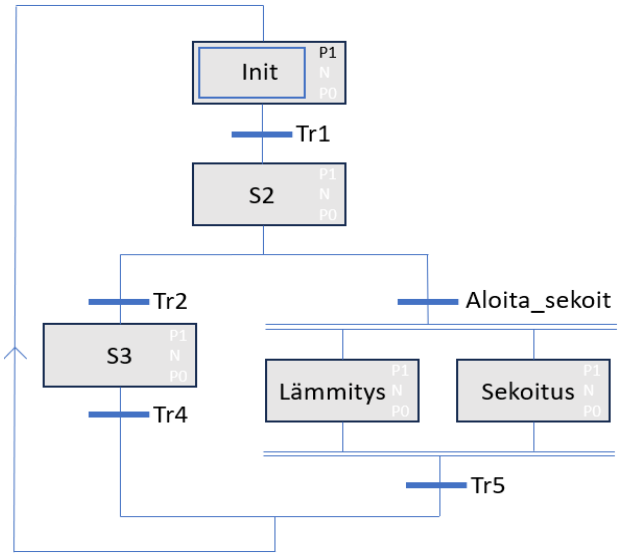
Structured text (ST) on korkean tason ohjelmointikieli, joka muistuttaa Pascalia ja C:tä ja se on erityisesti suunniteltu ohjaimien ohjelmoimiseen. ST on kompakti ohjelmointikieli, sillä on hyvä arkkitehtuuri ja se sisältää laajan valikoiman erilaisia rakenteita, kuten esimerkiksi muuttuja arvojen asettaminen, funktioiden/funktiolohkojen kutsuminen, lausekkeet, ehto lausekkeet ja monia muita ominaisuuksia. ST kieli sopii hyvin esimeriksi monimutkaisiin laskelmiin ja silmukoiden ohjelmointiin. Koodia on helppo lukea ja kirjoittaa, koska ST:n rakenteet on jäsennelty hyvin ja loogisesti. Kielen tiiviyden vuoksi koodista saa paremman yleiskuvan ja vierittämisen tarve pienenee. ST:tä voi käyttää turvaluokitukseen SIL-3 asti. (AC 800M Planning 2016, 65, 58.)



KUVIO 18. Esimerkki ST-koodista (AC 800M Planning 2016, 66)

Sequential Function Chart (SFC) on ohjelmointikieli, joka antaa käyttäjän kuvata sekvenssilogiikalla ohjattavia laitteita graafisesti. Graafisella esitystavalla saadaan kuvattua kaikki prosessin ohjaustoiminnot yhdessä sekvenssi rakenteessa, vaikka se sisältäisikin useita rinnakkaisia toimintaketjuja. Tämän lisäksi sekvenssit voivat olla hierarkkisia, eli toimintaketjut voidaan ryhmitellä selkeäksi, korkean tason esitykseksi prosessin ohjausyksikössä. SFC soveltuu korkeimmillaan turvaluokituksen SIL-2 ohjelmointiin ja tästä alaspäin. (AC 800M Planning 2016, 86, 58.)

Sekvenssi voidaan ajatella ”yksikkönä”, joka sisältää täydellisen sekvenssin. Sekvenssi on ympäröity ehdottomalla ja suljetulla silmukalla, kun sekvenssi on valmis, ensimmäinen vaihe aloitetaan uudelleen. Sekvenssit voidaan myös jakaa erilaisiin rakenteisiin. On olemassa kahden tyyppisiä sekvenssi rakenteita, jotka ovat sekvenssin valinta ja samanaikainen sekvenssi. Sekvenssin näkymä voidaan järjestää useille hierarkkisille tasoille hyödyntämällä alisekvenssi funktiota. (AC 800M Planning 2016, 86.)



KUVIO 19. Esimerkki sekvenssi rakenteesta “Init” P1 sisältää koodia N ja P0 ovat tyhjiä (mukaillen AC 800M Planning 2016, 87)

SFC on tehokas työkalu, joka soveltuu hyvin projektien suunnitteluun heti alusta asti. SCF:tä voidaan myös käyttää kuvaamaan yksityiskohtaisemmin prosessiobjektien käyttäytymistä. SFC:n graafisia kuvia voidaan hyödyntää esittämään järjestelmän käyttäytymistä. Koska SFC:n esitystapa on, selkeä se soveltuu hyvin asiakkaan ja ohjelmoijan väliseen kommunikointiin. Yleensä projektien alkuvaiheissa on monia järjestelmän käyttäytymiseen liittyviä näkökulmia, joita ei olla tarkkaan määritelty. Esimerkiksi jos SFC-kaaviot on luotu projektin ensimmäisessä vaiheessa, voidaan niitä tarvittaessa tarkentaa ja jalostaa uuden tiedon tultua saataville. (AC 800M Planning 2016, 87–88.)

Vaikka SFC:llä on monia etuja suunnittelu- ja rakennetyökaluna, se ei silti ole ”täydellinen” ohjelmointikieli. Siksi siirtymien tilat ja toimintojen kuvaukset on ohjelmoitava vähintään yhdellä tai useammalla standardin IEC 61131-3 määrittämistä ohjelmointikielistä. Usein kokeneemmat ohjelmoijat hyödyntävät ST:tä sekvenssilogiikkaa ohjelmoidessa. (AC 800M Planning 2016, 88.)

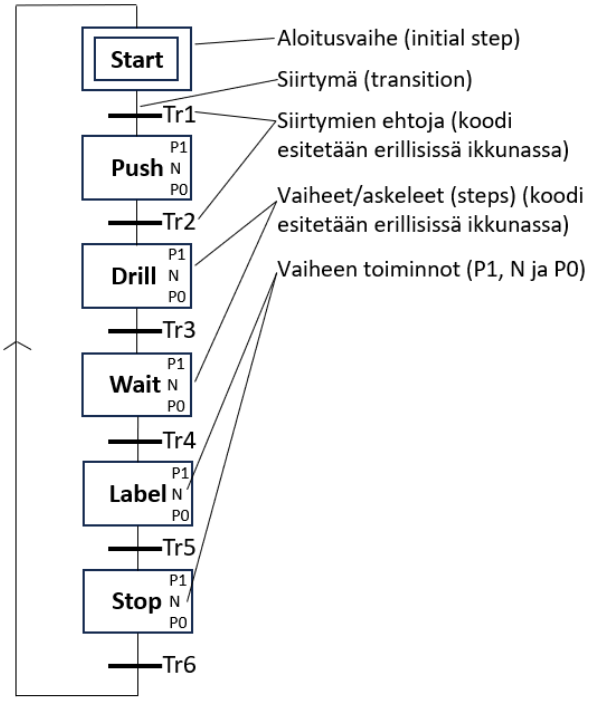
SFC-editorissa on useita komentoja, joilla luodaan vaiheita, siirtymiä, sekvenssin valintoja, ussia haaroja, hyppyjä, alisekvenssejä ja niin edelleen. Sekvenssin peruselementtejä ovat vaiheet ja siirtymät (KUVIO 20). Jokaiseen siirtymään on liitetty looginen siirtymäehto. Vaiheiden toiminnot ohjelmoidaan käyttäen ST ohjelmointikieltä. Vaiheiden toimintoja (P1, N tai P0) silmäilemällä voidaan tehdä havaintoja sen suhteen, onko vaiheisiin kirjoitettu koodia vai ei (KUVIO 19). Koodien tilaa havainnollistetaan väritetyllä tekstillä laatikon oikeassa laidassa. Tekstin eri värit ilmaisevat seuraavaa: valkoinen teksti kertoo, että lohko on olemassa mutta se on tyhjä, musta teksti kertoo, että lohko on olemassa ja se sisältää koodia ja väritön tarkoittaa sitä, että toimintolohkoa ei ole olemassa. (AC 800M Planning 2016, 88.)

Sekvenssin koodia tehdessä on hyvä huomata, että meidän pitää seurata muutamia sääntöjä. Näihin sekvenssi sääntöihin kuuluu: sekvenssisilmukka on aina suljettu, viimeinen siirtymä on yhdistetty ensimmäiseen vaiheeseen ja suoritus jatkuu viimeisestä vaiheesta ensimmäiseen vaiheeseen, kun siirtymäehto on totta. Siirtyminen vaiheiden välillä suoritetaan siirtymäehtojen avulla, jotka ovat loogisia lausekkeita ja sisältävät prosessisignaaleja. (AC 800M Planning 2016, 89.)

SFC:n voi ajatella menetelmänä, jolla ohjaustoiminnot jaetaan sarjaksi vaiheita, joita esitetään suorakulmaisina laatikoina ja yhdistetään toisiinsa pystysuuntaisilla viivoilla. Jokainen vaihe edustaa ohjattavan järjestelmän fyysistä tilaa. Jokaisella yhdistävällä viivalla on vaakasuuntainen palkki, joka edustaa siirtymää. Vaiheiden välillä siirtymiseen liittyy siirtymäehto, kun siirtymäehto muuttuu todeksi,

vaihe deaktivoidaan ennen siirtymää ja siirtymän jälkeen uusi vaihe aktivoidaan. (AC 800M Planning 2016, 90.)

Kaikki SFC-sekvenssit vaativat aloitusvaiheen, joka määrittää ohjelman suorittamisen sen jälkeen, kun järjestelmä on käynnistetty. Aloitusvaihe piirretään suorakulmaisena laatikkona, jossa on kaksinkertaiset reunaviivat. Aloitusvaihe pysyy aktiivisena siihen asti, kunnes seuraava siirtymä on totta, tämän jälkeen ohjelman on mahdollista siirtyä aloitusvaiheesta seuraavaan vaiheeseen. Jokaiseen vaiheeseen liittyy yleensä yksi tai useampi toiminto. Toiminnot kuvaavat laitteiden fyysistä toimintaa tehtaalla, kuten esimerkiksi venttiilin avaamista tai moottorin käynnistämistä. On hyvä huomata, että SFC-ohjelma suorittaa pelkästään niitä vaiheita, jotka on aktivoitu. (AC 800M Planning 2016, 90.)



KUVIO 20. Sekvenssikaavion tulkinta (mukaillen AC 800M Planning 2016, 91)

Kaikki SFC:n vaiheet on nimettävä uniikisti ja on hyvä huomata, että samoja nimiä ei voida käyttää myöhemmin uudelleen sekvenssissä. Jokainen vaihe aktivoituu automaattisesti, vaihe pysyy aktiivisena ja suorittaa koodia niin pitkään kunnes siirtymäehto muuttuu todeksi ja voidaan siirtyä seuraavaan vaiheeseen. Peräkkäiset vaiheet erotetaan aina siirtymäehdolla, jossa siirtymäehdon tuloksena pitää olla totuusarvomuuttuja. Siirtymäehto voi pitää sisällään mitä tahansa koodia, muuttujia ja kompleksisia lausekkeita, kunhan lopputuloksena on totuusarvomuuttuja. (AC 800M Planning 2016, 91.)

SFC:n vaiheet kuvaavat ohjattavien laitteiden tai prosessien tiloja. Ohjaimen suorittaessa SFC ohjelmaa, tilamalli toimii pelkästään ohjaimen sisäisenä muistiesityksenä ohjaustoiminnoille. Ohjaustoiminnon aktivoimiseksi tarvitaan vähintään yksi toiminnankuvaus, joka sisältää ohjelmakoodia toimilaitteen ohjaamiseksi. Toimintojen ohjelmoimiseksi käyttäjä voi valita neljästä jäljelle jäävästä IEC ohjelmointikielestä mieluisimman. (AC 800M Planning 2016, 92.)

Vaiheiden toiminnot voidaan määritellä erikseen, jotta saadaan tarkennettua miten ja milloin toiminto tulisi suorittaa. Useimmat SFC editorit tukevat kolmea toiminnankuvausta P1, N ja P0. Ohjelmoitaessa ei ole kuitenkaan pakko käyttää kaikkia toiminnankuvauksia. Useimmat sekvenssit käyttävät vähintään toimintokuvasta N, mutta vaihtoehtoisesti tämänkin voi jättää tyhjäksi, joka johtaa vaiheeseen, jossa ei suoriteta toimenpiteitä. (AC 800M Planning 2016, 92.)

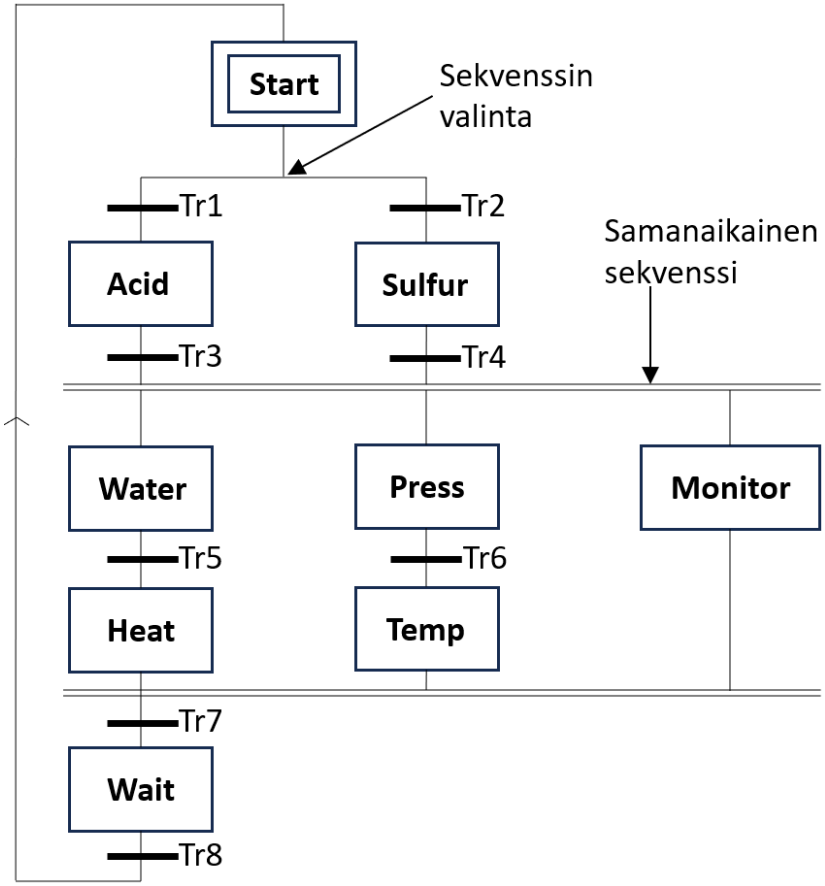
- Toiminnankuvaus N (Non-stored) suorittaa toimintokoodia jatkuvasti, kun vaihe on aktiivisena.
- Toiminnankuvaus P1 (pulssi nouseva reuna) suorittaa toimintokoodin kerran, vaiheen aktivoituessa.
- Toiminnankuvaus P0 (pulssi laskeva reuna) suorittaa toimintokoodin kerran, vaiheesta poistuessa. (AC 800M Planning 2016, 92.)

Useimmat järjestelmät vaativat useita haaroja sekvenssirakenteissaan. Haaroittumista hyödynnetään eniten eräajo sovelluksia suunnitellessa. Jos ohjelmassa on, useampi haara meillä on myös useampi siirtymäehto voimassa samanaikaisesti. Saman arvoisissa haaroissa vain yksi haara voi olla suorituksessa kerrallaan. Siirtymäehtojen aktivointijärjestys määräytyy sen mukaan mikä siirtymäehto on tullut todeksi ensimmäisenä, jos ehdot tulevat todeksi samanaikaisesti silloin vasemmanpuoleisin haara saa korkeimman suoritusprioriteetin. (AC 800M Planning 2016, 93.)

Eräajo prosessien suunnittelussa tulee usein tarve haaroille, joita voidaan suorittaa samanaikaisesti. Pääsekvenssiä käytetään ensisijaiseen prosessin ohjaukseen, kun taas toissijaisia rinnakkais- sekvenssejä käytetään valvomaan prosessin normaalia toimintaa. Rinnakkais- sekvenssit pitävät huolta esimerkiksi siitä, että laitoksen lämpötila ja paineet ovat vaaditulla tasolla, muussa tapauksessa ohjausjärjestelmä saattaa pysäyttää prosessin. (AC 800M Planning 2016, 93.)

Seuraavassa esimerkissä, kun jompikumpi siirtymäehdoista Tr1 tai Tr2 tulee todeksi, valitaan vastaava haara ja jatketaan ohjelman suorittamaista kyseistä haaraa pitkin. Seuraavat kolme haarautuvaa haaraa

alkavat joko siirtymäehdolla Tr3 tai Tr4. Tämän jälkeen ohjelman suorittaminen jatkuu samanaikaisesti ja itsenäisesti kolmea polkua pitkin, kunnes kaikki polut yhtyvät jälleen. Samanaikaisen sekvenssin haarautuva ja yhtyvä polku piirretään kahdella viivalla, jotta rakenne saadaan erotettua sekvenssin valinnasta. Samanaikaisen sekvenssirakenteen siirtymäehtoa, ei tarkisteta ennen kuin kaikki haarat ovat suoritettuina, eli tarkistus suoritetaan vasta silloin kun jokaisen haaran viimeinen vaihe on aktivoitunut. (AC 800M Planning 2016, 93–94.)



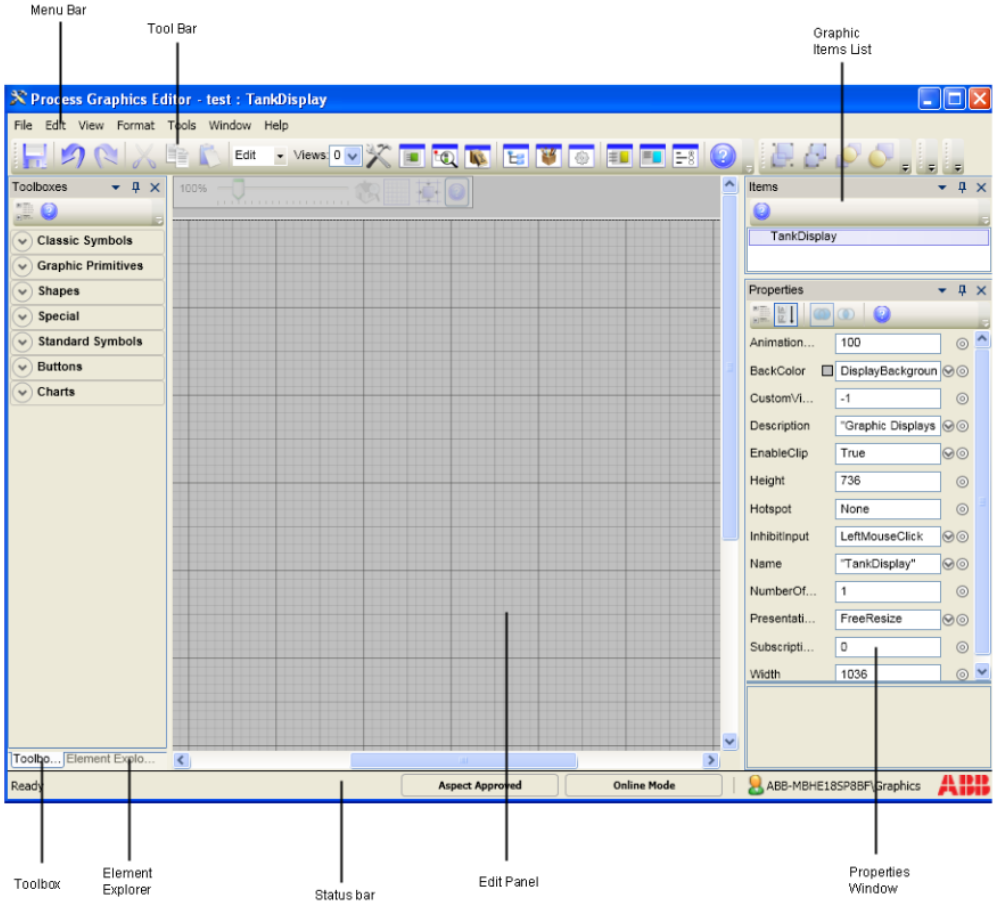
KUVIO 21. Sekvenssin valinta ja samanaikainen sekvenssi (mukaillen AC 800M Planning 2016, 94)

3.2.4 Graafinen suunnittelu

ABB 800xA järjestelmässä graafinen suunnittelu toteutetaan ”Graphics Builder” ohjelmalla (KUVIO 22). Grafiikkaohjelman avulla voidaan: rakentaa graafisia elementtejä, määrittää graafisia näyttöjä, katsella grafiikkaa tai esittää prosessin ohjaukseen liittyvää tietoa ”Faceplate” ikkunan kautta (System Planning 2016, 244).

Seuraavat ominaisuudet kuvaavat Graphics Builder ohjelmaa:

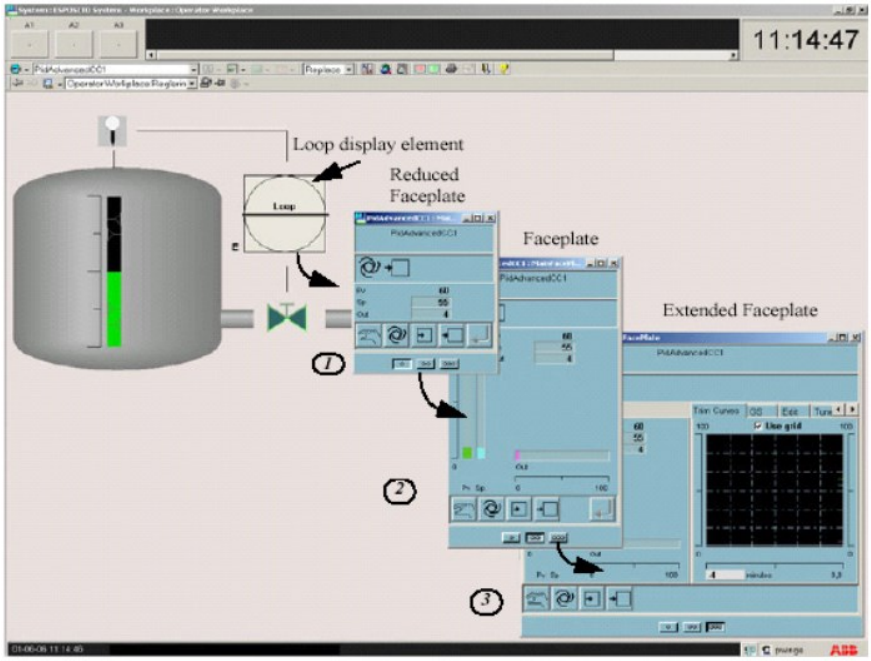
- ”Properties Window” määrittää graafisten kohteiden ja syötteiden ominaisuudet.
- ”Expression Variables” määrittää ehdot ohjelmoitaville lausekkeille.
- ”Expression Editor” on laajamittainen muokkaustyökalu, joka soveltuu ominaisuusarvojen tarkkaan määrittämiseen (KUVIO 24).
- ”Toolbox” esittää järjestelmään määritettyjä graafisen rakenteen rakennuspalikoita.
- ”Element Explorer” auttaa hakemaan ja lisäämään graafisia elementtejä muokattavaan kuvaan.
- ”Input Properties” työkalulla määritetään ja muokataan käyttäjän määrittelemiä ominaisuuksia.
- ”Solution Library” määrittelee graafiset elementit uudelleenkäytettävinä ratkaisuina. (System Planning 2016, 244.)



KUVIO 22. Grafiikan suunnittelutyökalu Graphics Builder (Process Graphics 2016, 31)

Graphic Builder tarjoaa neljä erilaista aspektityyppiä ”Process Graphics 2” aspektijärjestelmän kautta. Näitä aspekteja ovat: ”Graphic Display PG2” joka on tarkoitettu prosessioperaattorien käyttöön, ”Graphic Element PG2” on objektitietoinen rakennuspalikka, ”Generic Element PG2” on yleiskäyttöinen rakennuspalikka ja ”Solution Library PG2” on graafinen aspekti, joka tukee ratkaisujen kopiointia ja liittämistä graafisiin näyttöihin Graphics Builder ohjelmassa. (Process Graphics 2016, 124–127.)

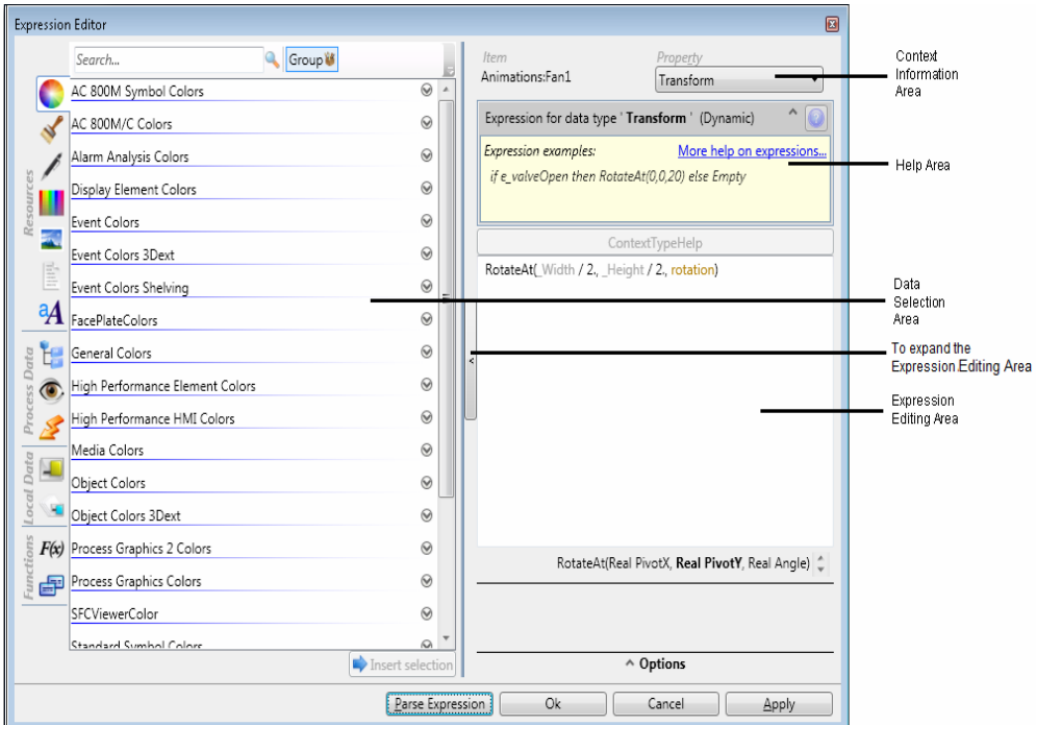
Kun aspektityypeistä luodaan aspekteja, ne määritellään Graphics Builder ohjelmassa. Prosessioperaattorit kutsuvat graafisia näyttöjä, joita he käyttävät prosessin valvomiseen ja ohjaamiseen. Yleisiä ja graafisia elementtejä käytetään rakennuspalikoina, joilla määritetään graafisia näyttöjä, muita graafisia elementtejä tai Faceplate näytön toiminta. (Process Graphics 2016, 125.) Seuraavaksi kuvio 23 havainnollistaa sen, miltä Faceplate näyttää ja kuinka se linkittyy grafiikassa käytettävään elementtiin. Faceplate toimii kolmella eri tasolla, mitä suurempi taso, sitä enemmän annetaan tietoa graafisesta elementistä.



KUVIO 23. PID-ohjaimen Faceplate (System Planning 2016, 252)

Määritelmät (expression) ovat yksinkertaisia vakioita tai monimutkaisia lausekkeita, jotka sisältävät viittauksia määritelmä muuttujiin, syöttö -ominaisuuksiin, objektien ominaisuuksiin, tietoihin tai resurssiviittauksiin. Määritelmien kautta päästään käsiksi järjestelmätietoihin. Määritelmäeditori (expression editor) auttaa käyttäjiä luomaan ja määrittelemään elementtien toimintaa erilaisten muuttujien tai

valinta toimintojen kautta (KUVIO 24). Editoriin sisältyy myös perus- ja monimutkaiset tyyppimuokkausohjelmat, jotka tarjoavat laajemman valikoiman muokkaustoimintoja, kuin perus tyyppimuokkausohjelma ”properties window” (KUVIO 22). (Process Graphics 2016, 77–78.)



KUVIO 24. Määritelmä editori (Process Graphics 2016, 66)

Määritelmämuuttujia käytetään väliaikaisten määritelmälausekkeiden tulosten tallentamiseen. Määritelmämuuttuja säilyttää määrittelyjen tulokset ja näin vähentävät määritelmien monimutkaisuutta mahdollistamalla laskutoimitusten suorittamisen välivaiheittain. Määritelmämuuttuja mahdollistaa usein käytettyjen määritelmien uudelleen käyttämisen. Määritelmämuuttujia käytetään myös paikallisten tilojen tallentamiseen. Esimerkiksi jos työtekijä painaa kuvaketta tietokoneella, tietyt graafiset kohteet tulevat käyttäjän nähtäville. Kuvakkeen painalluksen aikana arvo voidaan kirjoittaa määritelmämuuttujaan. Muiden graafisten kohteiden ”visible” ominaisuus voi sisältää määritelmän, joka viittaa tähän määritelmämuuttujaan. (Process Graphics 2016, 77–78.)

Määritelmät voivat sisältää operaattoreita, operandeja ja funktioita. Määritelmät suorittavat laskelmia, jotka perustuvat parametrien arvoihin ja tämän jälkeen palauttaa yhden arvon, joka on määritelmä laskennan tulos. Tulos kuvataan ominaisuuden määrittelemässä ” Property Expression” (ominaisuus ilmaisimessa). Jokainen Property Expression tietotyyppi on määritelty selkeästi. Muutamia esimerkkejä

Property Expression tietotyypeistä, joihin kuuluu reaaliluvut, kokonaisluvut, totuusarvot, värit ja merkkijonot. (Process Graphics 2016, 200.)

Ohjelmointia ohjaava syntaksi määritteitä määriteltäessä:

- Käytä ”if-then-else” (jos-sitten-muuten) lausekkeita ehtojen ohjelmoimiseen.
- Käytä operaattoreita laskutoimituksiin.
- Käytä määritelmissä funktioita laskutoimitusten suorittamiseen. (Process Graphics 2016, 200.)

3.3 PHA-Pro & LOPA

PHA-Pro on työkalu, jota käytetään prosessiteollisuudessa, koska se on helppo käyttöinen ja monipuolinen ohjelma, joka tukee useita prosessiteollisuuteen liittyviä standardeja. PHA-Pro antaa paljon tietoa erilaisista riskeistä, auttaa järjestelmien suunnittelussa ja antaa mahdollisuuden käyttää valmiita mallipohjia riskienarviointiin.

PHA-Pro tarjoaa käyttäjäystävällisen, joustavan ja tietopohjaisen ratkaisun, joka on kehittynyt kaupallisen käytön myötä vastaamaan maailman suurimpien yritysten vaatimuksia. PHA-Pro on todettu toimivaksi ohjelmistoratkaisuksi, jossa on valmiiksi muotoiltuja vakiotyyppisiä PHA-pohjia ja useita ominaisuuksia, jotka ovat helppokäyttöisiä ja intuitiivisia. (Minimize Risk Exposure 2016, 1.)

PHA-Pro:n tukemat lakisääteiset standardit

• OSHA 1910.119 PSM
• EPA 40 CFR Part 68 RMP
• Seveso II Directive
• CCPS LOPA
• IEC 61882
• IEC 61511, IEC 61508, ANSI/ISA, S84.00.01
• ISO 31000, AS/NZS 4360
• CSA Standardi Z1000 (Minimize Risk Exposure 2016, 1.)

PHA-Pro:n ominaisuuksia

• Dynamic link of diagrams with worksheets
• Professional reports exportable in HTML, MS Word, MS Excel
• International support such as multi-language and right to left data entry
• Enhanced AutoType and Copy features
• Enhanced Release Management
• Criticality Matrix
• Linked HAZOP and LOPA templates
• Recommendations Manager Identify risk (Minimize Risk Exposure 2016, 2.)

LOPA

Yritykset käyttävät myös ”Layer of Protection Analysis” (LOPA) -menetelmää, joka on määritelty järjestön ”Center for Chemical Process Safety” toimesta. LOPA analysoi erillisiä tapahtumaskenaarioita ja vertaa niiden riskiarviota riskikriteereihin. LOPA auttaa yrityksiä selvittämään, kuinka monta riippumatonta suojakerrosta tarvitaan ja kuinka paljon kunkin kerroksen pitää vähentää riskiä, jotta mahdollinen vaaratilanne sopii yrityksen riskinsietokykyyn. (What is the Difference Between PHA, HAZOP & LOPA? 2022, luku ”What is LOPA? How does it differ from HAZOP?”.)

LOPA ja HAZOP suoritetaan erikseen, mutta ne täydentävät toisiaan tarjoten kattavan riskiarvioinnin. HAZOP auttaa yrityksiä ymmärtämään nykyiset riskit esittämällä kaikki mahdolliset vaihtoehdot. LOPA puolestaan esittää käytettävissä olevat suojakerrokset HAZOP:ssa todettuihin riskeihin. LOPA auttaa myös tunnistamaan järjestelmän mahdollisia heikkouksia, jotta niihin voidaan puuttua. (What is the Difference Between PHA, HAZOP & LOPA? 2022, luku ”What is LOPA? How does it differ from HAZOP?”.)

HAZOP ja PHA arvioivat prosessiturvallisuuteen liittyviä ongelmia. Tämän jälkeen käytetään LOPA-menetelmää tunnistamaan mahdollisia aukkoja järjestelmässä ja esittämään missä mahdollisesti tarvittaisiin SIF:iä. LOPA määrittää myös sen kuinka korkea SIL-taso vaaditaan SIF:lle. Nämä SIF:it ovat osana SIS-järjestelmää. Kaikki nämä menetelmät liittyvät toisiinsa, jossa jokainen elementti vaikuttaa seuraavaan käsiteltävään elementtiin. (What is the Difference Between PHA, HAZOP & LOPA? 2022, luku ”Process Safety Management Tools: A Summary”.)

4 CABB

CABB on kansainvälinen yhtiö, joka valmistaa hieno- ja erikoiskemikaaleja. CABB työllistää tällä hetkellä noin 1100 työntekijää 5 eri maassa: Suomessa, Saksassa, Yhdysvalloissa, Sveitsissä ja Kiinassa. Yritykset ympäri maailmaa voivat tilata CABB:ltä tuotteita, joissa on useita erilaisia kemiallisia vaiheita. Kemiallisten tuotteiden tuotantomäärät ovat kymmenistä kiloista satoihin tuhansiin kiloihin asti. Tyypillisesti tuotantoa tuotetaan panosprosesseina. CABB:n prosessituotanto pystyy tuottamaan aineita, joita käytetään esimerkiksi kasvien suojelussa, lääkkeissä, kosmetiikassa, hajusteissa tai puhdistusaineissa. (CABB Oy 2023.)

4.1 CABB Kokkola

CABB Oy Kokkola sijaitsee KIP (Kokkola Industrial Park) alueella. CABB:n Kokkolan toimipiste sai alkunsa 1940 luvulla Kemiran toimipisteenä. Vuonna 1996 Kemira muutti liiketoimintamallikseen hienokemikaalit ja yrityksen nimi vaihdettiin Kemira Fine Chemicals Oy:ksi. Seuraavan kerran omistaja vaihtui 3i:ksi ja nimi muutettiin KemFine:ksi. Vuonna 2005 Avecia Fine Chemicals ostettiin Skotlannista ja nimi vaihtui KemFine UK Ltd:ksi. Seuraavan kerran yritys vaihtoi omistajaa vuonna 2010, kun KemFine UK Ltd myytiin Aurelius AG:lle. Viimeisin myyntitapahtuma suoritettiin vuonna 2011, jolloin KemFine myytiin CABB AG:lle. (CABB Oy 2023.)

CABB Oy Kokkola keskittyy pääasiallisesti rahtivalmistukseen, kasviensuojeluaineisiin sekä niiden välituotteisiin ja erikoiskemikaaleihin. Näitä kemikaaleja valmistetaan synteesilaitoksella, MAP-tehtaalla ja Moni-1 tehtaalla. Tuotteista syntyvä orgaaninen jäte käsitellään CABB Kokkolan omalla polttolaitoksella. Kokkolan toimipiste työllistää tällä hetkellä noin 250 henkilöä. (CABB Oy 2023.)

KUVIO 25. CABB Kokkolan tehtaat (CABB Oy 2023.)

4.2 CABB:n asettamat tavoitteet opinnäytetyölle

Opinnäytetyön tavoitteena on saada siirrettyä Cumulus px100 järjestelmä erillisestä Siemensillä toimivasta järjestelmästä CABB:n omaan ABB-järjestelmään. Työn tavoitteena on saada ohjelmoitua Cumulus px100 järjestelmä mukaillen alkuperäisiä ohjelmointi ehtoja. Työhön tehdään lukituskaaviot, käydään läpi HAZOP-tarkastelu ja luodaan operaattorille sopiva graafinen käyttöliittymä.

5 CUMULUS PX100 OHJELMOINTI

Työn suunnittelu aloitettiin sopimalla siitä, miten työ tulisi valmistaa CABB:in toimintatapojen mukaisesti. Ensimmäisenä asiana nousi ylös se, että grafiikan pitää muistuttaa mahdollisimman paljon PI-kaaviota, jotta operaattorin on helppo tulkita sitä missä tahansa tilanteessa. Seuraavaksi tutkittiin mitkä kaikki laitteet ja instrumentit kuuluvat Cumulus px100 kokonaisuuteen, jotta voimme täsmentää ohjelmoinnin tarpeet niille tarkoitetuille kohteille. Suunnittelua jatkettiin HAZOP-kokouksessa, jossa käytiin läpi mahdolliset riskitekijät, varotoimenpiteet ja miten niihin tulisi reagoida fyysisesti ja ohjelmallisesti. HAZOP-kokouksen pohjalta kehiteltiin lukituskaavio, jota käytettiin koodattavassa ohjelmistossa.

5.1 Ohjelmassa käytettyjen raja-arvojen määrittely

Cumulus järjestelmän riskien ja rajojen määrittelyyn käytettiin HAZOP-analyysia. Kun kaikki riskit ja tarvittavat parametrit on selvitetty, voidaan jatkaa uuden Cumulus ohjelmiston kehittämistä. Salassapitosyistä ei voida esittää esimerkkiä HAZOP-arvioinnissa käytetystä työkalusta.

HAZOP-kokouksessa todettujen riskien huomioiminen ja arviointi.

1. Paine-eromittaus 1PD-3406 lukitsee typen säätöventtiilin ja sulkuventtiilin HV-020
2. Matala lämpötila kierrossa -40C (1TI-3178 tai 1TI-11) lukitsee typen XV-venttiilin HV-020
3. HAZOP:ssa määriteltyjen toimenpiteiden lisäksi kaikki alun perin logiikassa olleet lukitukset mukaan
4. Mikäli 1FI-3178 virtaus jää alle 2m3h lukitaan typpi XV
5. Mikäli Cumuluksen yli vaikuttava paine-ero on alle 0,05 bar lukitaan typpi XV
6. HV-20 lukitaan korkeasta paineesta MA3170 vaippakierron paisuntasäiliössä (lähellä 5,5barg)
7. Mikäli virtaus 1FI-3178 jää pienellä viiveellä alle 2 m3/h, lukitaan pumppu MA3171
8. Mikäli ulostulevan typen lämpötila on alle -60C, niin lukitaan typen HV kiinni, jotta typpi ei pääse nestemäisenä ulos.
9. Matala pinta MA3171 kierron paisarissa lukitsee pumpun

Työn selkeyttämiseksi olen kirjannut ylös kaikki positiot, jotka liittyvät Cumulus laitteistoon. Samalla on hyvä kirjata ylös positioihin liittyvät toiminnot ja raja-arvot. HAZOP-keskustelussa todettiin, että

kaikki entiset lukitukset pitää löytyä myös uudesta järjestelmästä. Lista on tehty käyttäen PI-kuvaa (KUVIO 30), HAZOP-analyysin raja-arvoja, operaatio manuaalissa esiintyvillä raja-arvoilla ja positio-tunnuksilla. Uudet positiotunnukset on merkitty heti alkuperäisten positiotunnusten viereen.

	(OLD) Name	(NEW) Name	Desc	Action	Limits	Type	Control?	Description	
1	TICZA-011	MA-1TI-3406	LOW LOW	SHUTDOWN	> -40C°	Lämmönmittaus	KYLLÄ		
			LOW	ALARM	> -39C°				
			HIGH	ALARM	< 10C°				
	TE-011					Lämmönmittaus	EI		
2	TICZA-015	(Ei käytetä)	LOW LOW	SHUTDOWN	> -55C°	Lämmönmittaus	EI KÄYTETÄ	(external&optional)	
			LOW	ALARM	> -45C°			Ei ohjelmoida	
			HIGH	ALARM	< 5C°				
	TE-015					Lämmönmittaus	EI		
3	TICZA-022	MA-2TI-3406	LOW LOW	SHUTDOWN	> -60C°	Lämmönmittaus	KYLLÄ		
			LOW	ALARM	> -48C°				
	TE-022					Lämmönmittaus	EI		
4	1TI-3178	MA-1TI-3178	LOW LOW	SHUTDOWN	> -40C°	Lämmönmittaus	KYLLÄ		
			LOW	ALARM	> -39C°				
			HIGH	ALARM	< 10C°				
5	PIZA-012	MA-1PD-3406	LOW LOW	SHUTDOWN	> 0.05 bar	Painemittaus	KYLLÄ		
			LOW	ALARM	> 0.2 bar				
			HIGH	ALARM	< 2.5 bar				
			HIGH HIGH	SHUTDOWN	< 4 bar				
	PDT-012					Painemittaus	EI		
6	PZA-031	(Ei käytetä)	LOW	SHUTDOWN	Instrumentin ilman syöttö häiriö	Painemittaus	EI	Ei ohjelmoida	
	PSLL-031					Painemittaus	EI		
7	1PI-3170	MA-1PI-3170	HIGH HIGH	SHUTDOWN	< 5.3 barg	Painemittaus			
8	1LS-3170	MA-1LS-3170	LOW LEVEL	SHUTDOWN	LS KYTKIN MUUTOS	Kytkin			
9	1FI-3178	MA-1FI-3178	LOW LOW	SHUTDOWN	> 2 m3/h	Virranmittaus			
10	EMERGENCY STOP								
	HZ-016	MA-1HZ-3406		SHUTDOWN		Kytkin	KYLLÄ		
11	HV-020	MA-1XV-3405				Sulkuventtiili	KYLLÄ		
12	TV-021	MA-1TV-3406			Säätö --> TICZA-011	Säätöventtiili	KYLLÄ		
13	MA3171	MA-M-3171				Kiertopumppu	KYLLÄ		

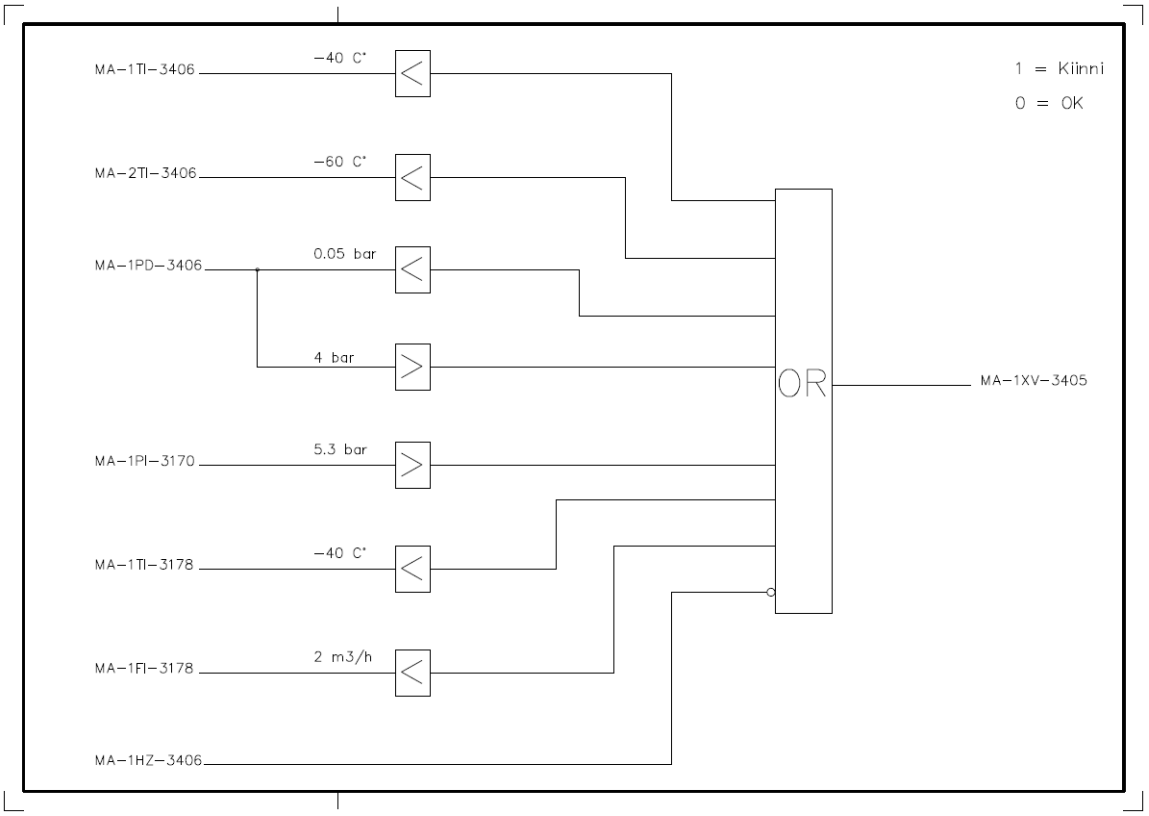
KUVIO 26. Cumulus laitteiston vanhat ja uudet positio tunnukset, ohjelmointirajat ja selvennykset

Seuraavaksi tehtiin lukituskaaviot, koska niiden pohjalta on helppo todeta, millaisia raja-arvoja on käytetty laitteiston turvalogiikan suunnittelussa ja samalla voimme myös päätellä sen, kuinka ohjelmiston logiikka toimii riskitilanteissa. Hyödyntämällä Excel taulukkoa, HAZOP-keskustelua ja Cumuluksen operaattori manuaaleja voimme luoda tehtävänkuvaan sopivat lukituskaaviot. Lukituskaaviot tehdään laitekohtaisesti eli yksi lukituskaavio vastaa yhden määritellyn laitteen sulkemisen asetettujen ehtojen täyttyessä.

Typen sulkuventtiilin MA-1XV-3405 lukitus. Venttiili lukitaan kun:

1. MA-1TI-3406 mittaama arvo on pienempi kuin - 40 c°
2. MA-2TI-3406 mittaama arvo on pienempi kuin - 60 c°
3. MA-1PD-3406 mittaama arvo on pienempi kuin 0,05 bar tai isompi kuin 4 bar

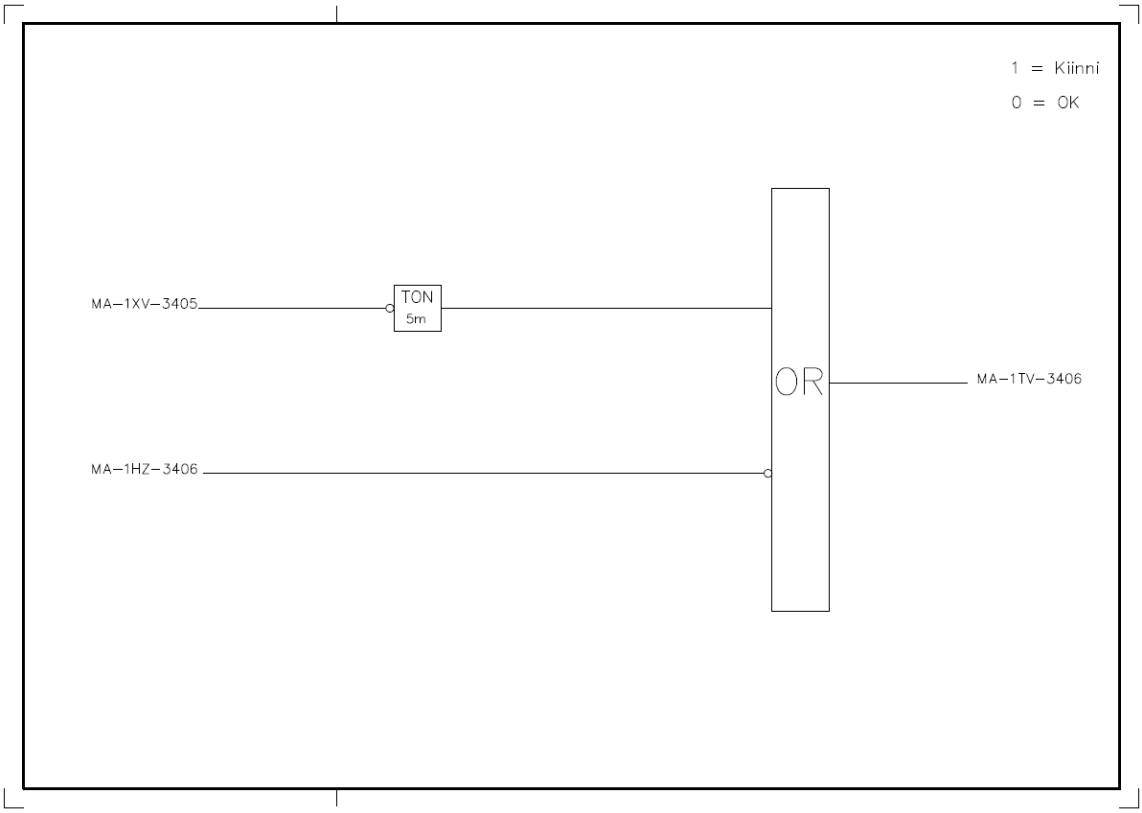
4. MA-1PI-3170 mittaama arvo on pienempi kuin 5,3 barg
5. MA-1TI-3178 mittaama arvo on pienempi kuin - 40 c°
6. MA-1FI-3178 mittaama arvo on pienempi kuin 2 m³/h
7. Hätäseispainike MA-1HZ-3406 aktivoituu



KUVIO 27. Sulkuventtiilin MA-1XV-3405 sulkeminen

Typen säätöventtiilin MA-1TV-3406 lukitus. Venttiili lukitaan kun:

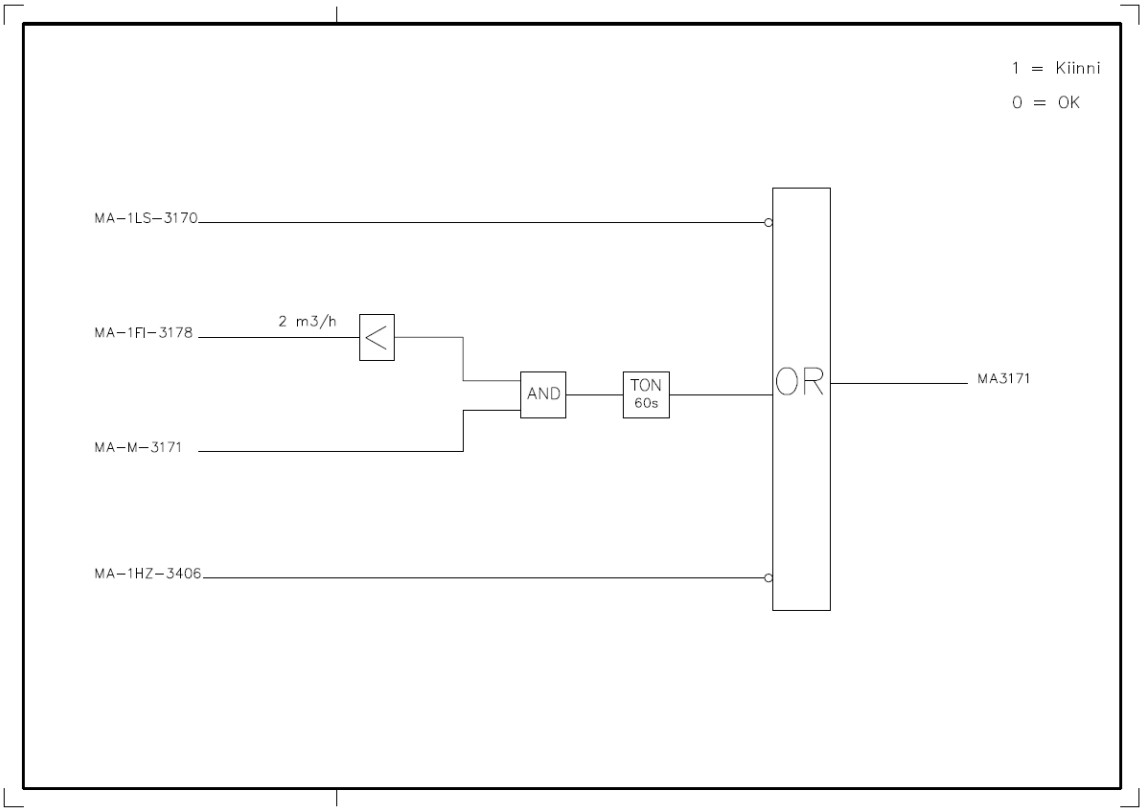
1. Sulkuventtiili MA_1XV_3405 on ollut suljettuna vähintään 5 minuuttia
2. Hätäseispainike MA-1HZ-3406 aktivoituu



KUVIO 28. Säätöventtiilin MA-1TV-3406 sulkeminen

Moottorin MA-M-3171 (MA3171) lukitus. Moottori lukitaan kun:

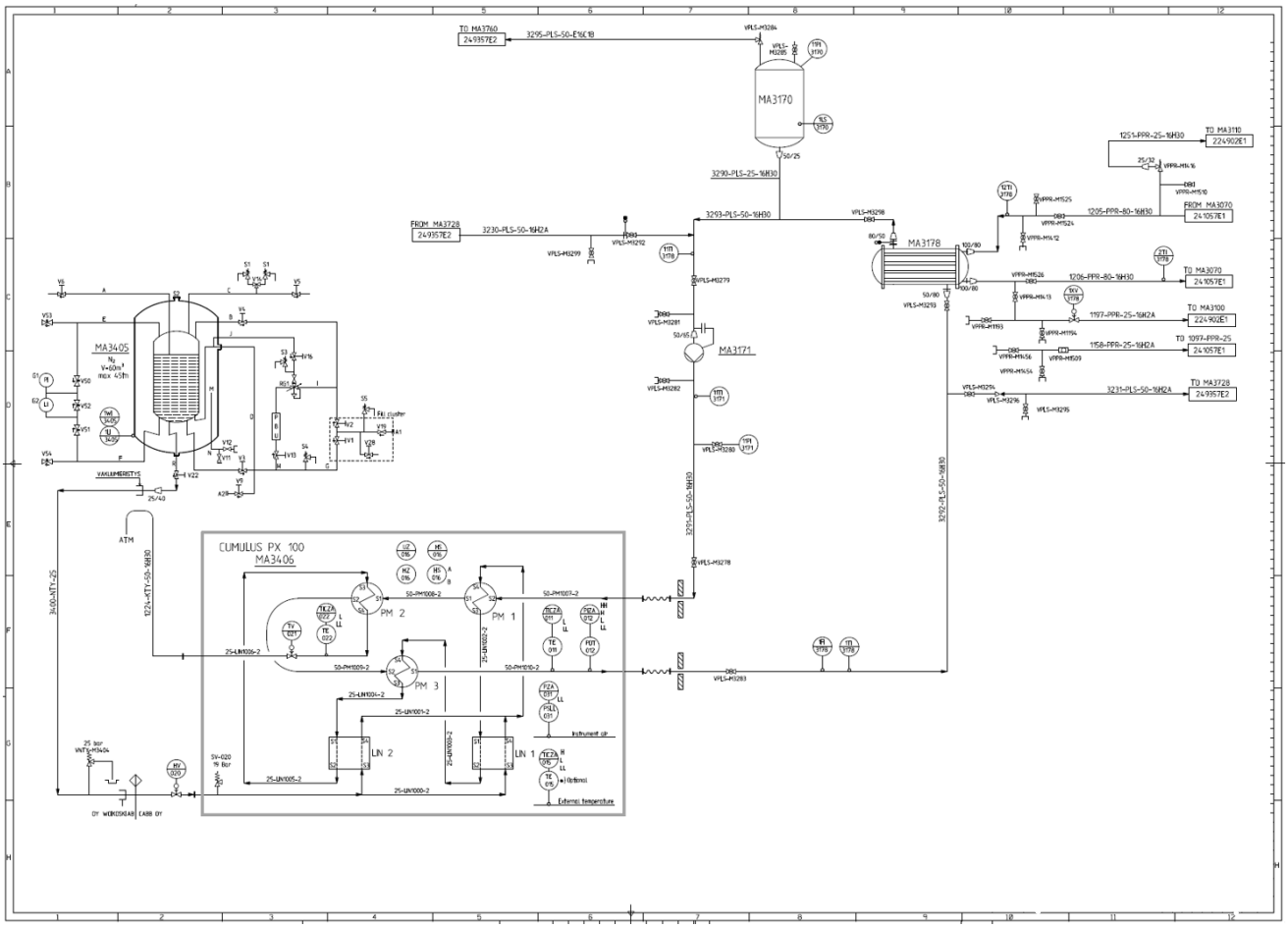
1. Paisuntasäiliön MA_1LS_3170 alarajakytkin vaihtaa tilaa
2. Moottori MA-M-3171 on päällä ja virtaus on ollut vähintään 60 s vähemmän kuin 2 m³/h
3. Hätäseispainike MA-1HZ-3406 aktivoituu



KUVIO 29. Moottorin MA3171 lukitus

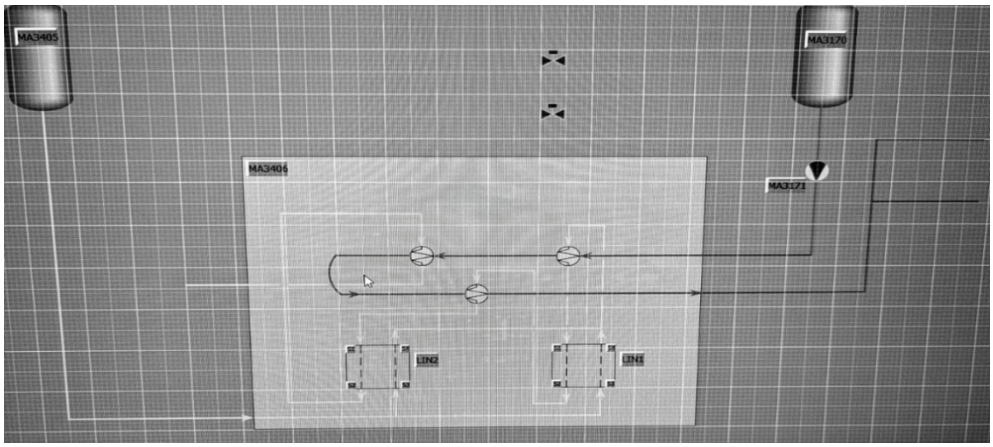
5.2 Graafinen käyttöliittymä

Ohjelman kehittäminen aloitettiin alustamalla ohjelmiston grafiikkaa, joka seuraa annettua PI-kaaviota mahdollisimman tarkasti. Grafiikkaan on otettu mukaan pelkästään Cumulus px100:an liittyvä prosessinkierto, instrumentit, moottorit ja säiliöt.



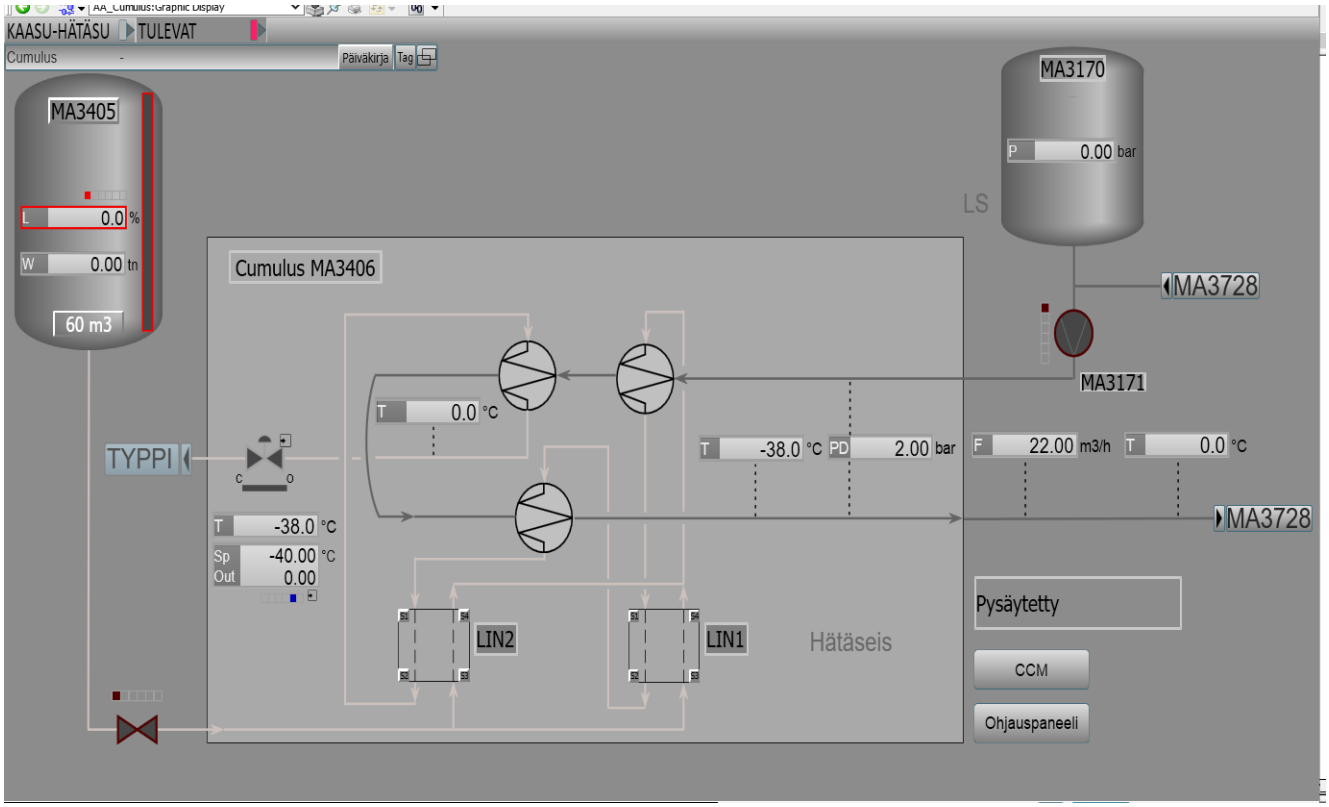
KUVIO 30. PI-kaavio, jonka pohjalta luodaan käyttöliittymän grafiikka

Grafiikan ensimmäisessä iteraatiossa on piirretty Cumulus laitteisto positiolla MA3406, typpisäiliö MA3405, paisuntasäiliö MA3170, moottori MA3171, lämmönvaihtimet ja typen höyrystimet LIN1 ja LIN2. Typellä toimiva kierto on piirretty valkoisin viivoin ja tuotteen kierto on piirretty tummanharmailla viivoilla.



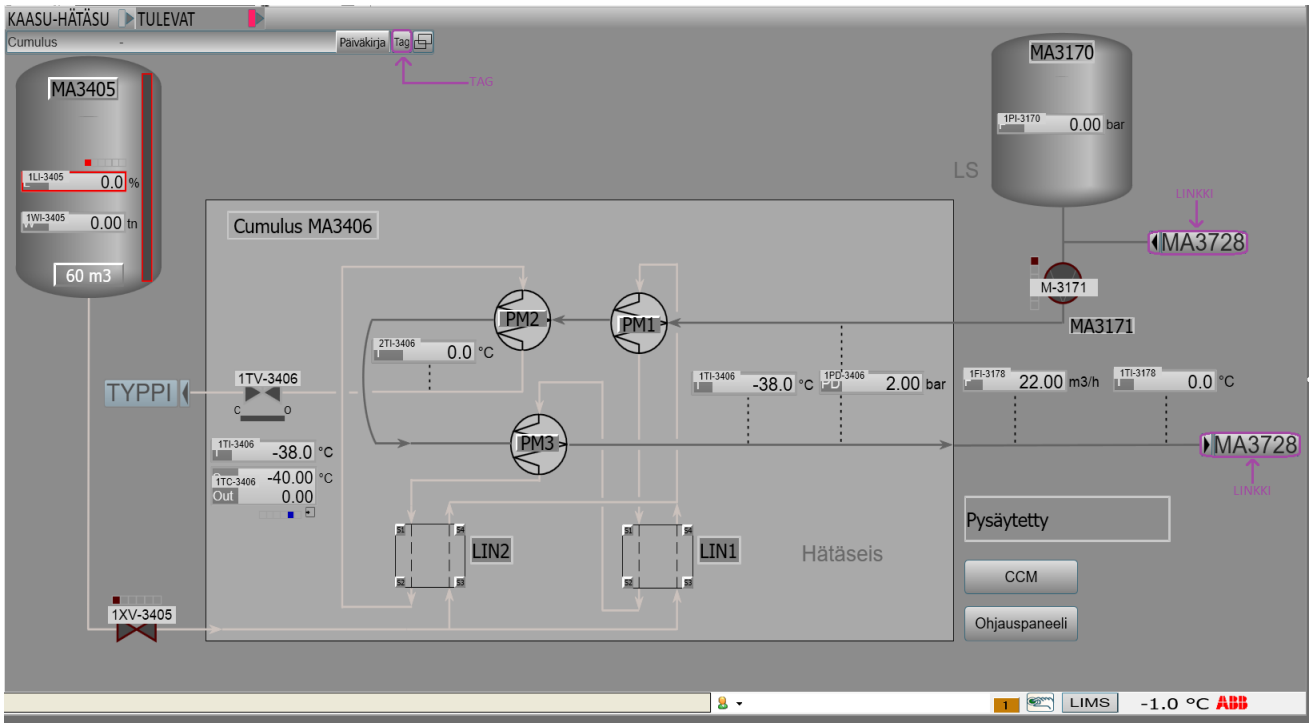
KUVIO 31. Graafisen liittymän alku

Graafisen osuuden kehittämistä jatkettiin lisäämällä kaikki tarvittavat hälytykset, mittaukset ja mittauksiin liittyvät mittausarvot tuotiin esille selvästi ja helposti luettavaksi. Grafiikkaan lisättiin myös sulkuventtiili, säätöventtiili ja tuotiin esille säätöventtiiliin liittyvät ohjausarvot. Kuvakkeita suurennettiin, jotta operaattorin on helpompi lukea graafista käyttöliittymää. Grafiikkaan lisättiin linkkejä (MA3728), jotka ohjaavat käyttäjän kyseiseen tuotantolinja näkymään, josta esimerkiksi voidaan havaita reaktori, jonka vaippakiertoa Cumulus järjestelmä jäähdyttää. Grafiikasta löytyy myös nappi ohjauspaneelille ja sekvenssin tarkkailulle, joka on nimellä ”CCM” (Custom Control Module). Grafiikkaan liittyvät lisätiedot saadaan esille painamalla ”Tag” nappia. Seuraavat neljä kuviota esittävät graafisen käyttöliittymän ulkonäköä ja funktionaalisuutta.



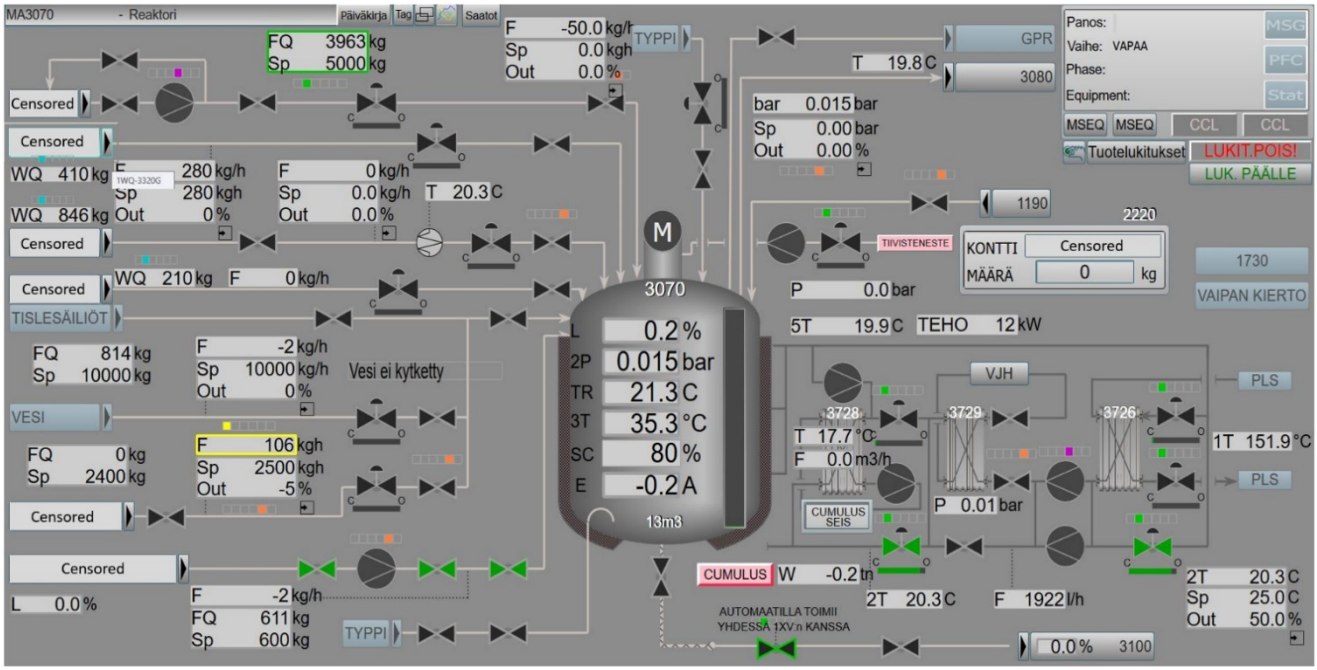
KUVIO 32. Viimeistelty graafinen käyttöliittymä eli ohjauksen perusnäkymä

Seuraavaksi kuvio 33 esittää grafiikan funktionaalisuutta ja näyttää mitä tapahtuu, kun painetaan painiketta ”Tag”. Painike tag esittää laitteisiin liittyvät positiotunnukset, joita ei voida havaita kuviossa 32. Kuvioon on myös lisätty näkyvyyttä painikkeelle tag ja linkeille ”MA3728”, jotta viitteet löytyvät kätevämmin ja on helpompi hahmottaa miltä grafiikassa esiintyvät painikkeet näyttävät.



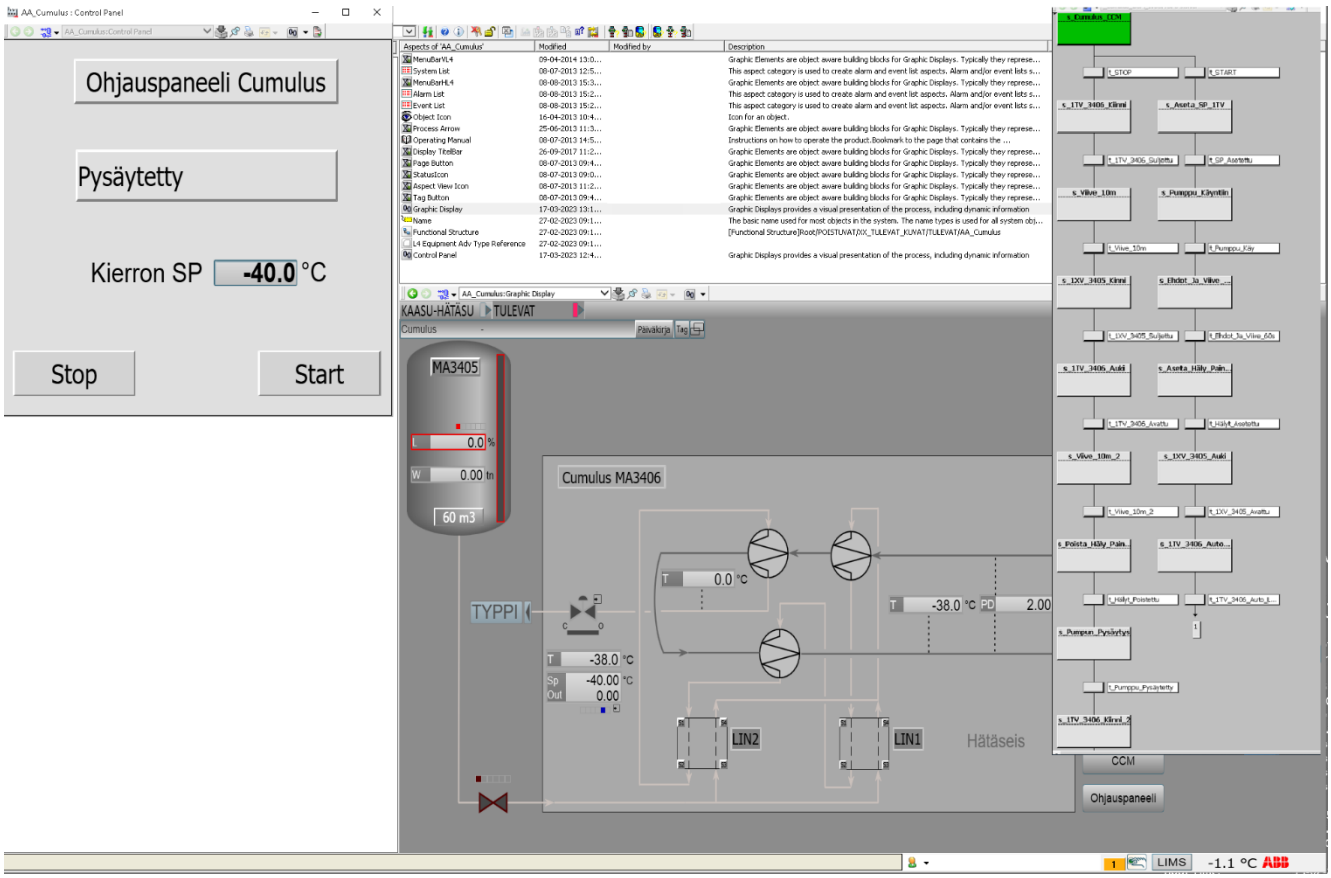
KUVIO 33. Tag napin tuomat positiot ja muut lisätiedot graafiseen näkymään

Kuvio 34 esittää mitä tapahtuu, kun käyttäjä painaa linkkiä MA3728. Linkki johtaa toiseen esitykseen, jossa havaitaan Cumulus laitteiston jäähdyttämä säiliö ja muita säiliöön liittyviä ominaisuuksia.



KUVIO 34. Linkin MA3728 avaama tuotantolinjan ohjausnäkymä

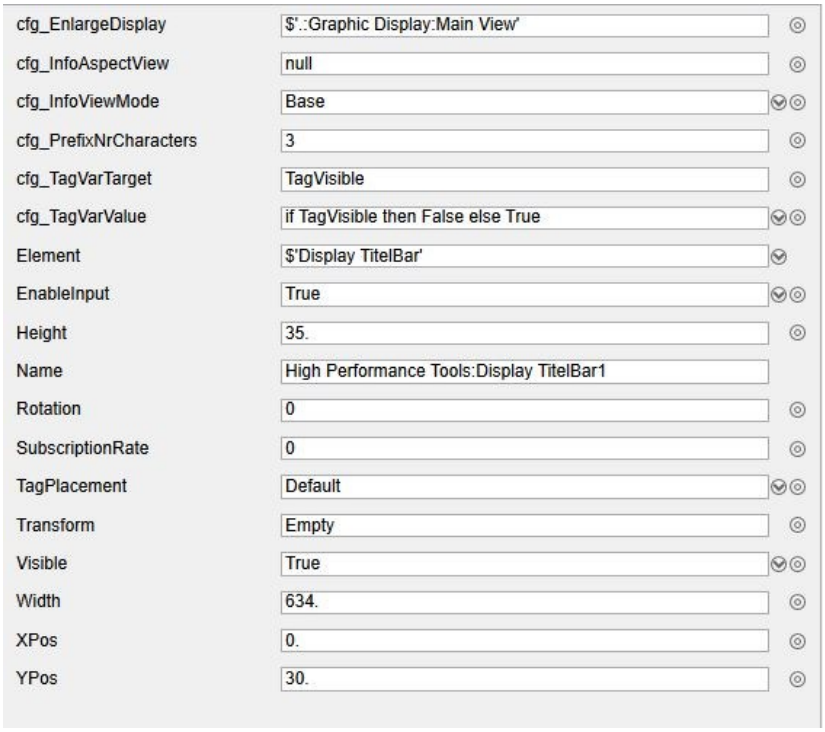
Kuvio 35 esittää mitä tapahtuu, kun painetaan painiketta ”CCM” tai ”Ohjauspaneeli”. Ohjauspaneeli esittää kierron nykyisen SetPoint asetusarvon, pysäytys funktion ja käynnistys funktion. CCM painike avaa ikkunan, joka esittää missä vaiheessa sekvenssiä pysäytys tai käynnistys funktio etenee.



KUVIO 35. Ohjauspaneelin ikkuna ja CCM ikkuna

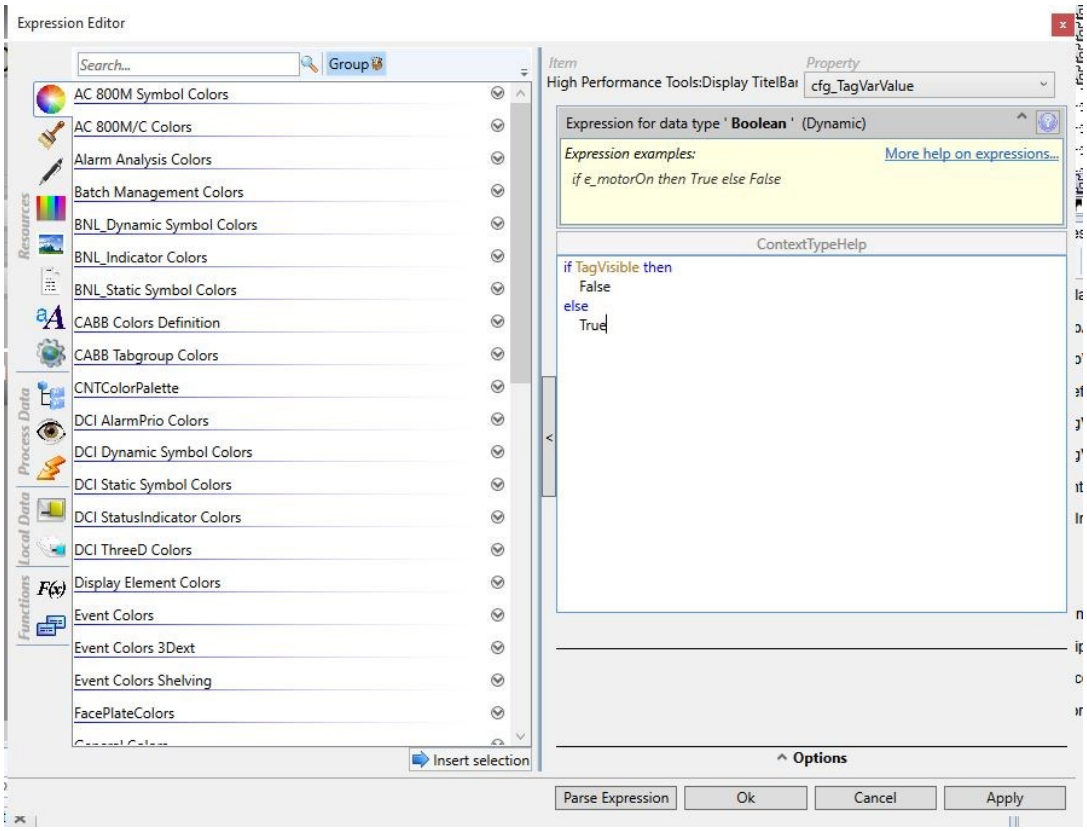
Ohjelmoitaessa graafisille elementeille toiminnallisia funktioita, meillä on karkeasti sanottuna noin kaksi päätoimista vaihtoehtoa toteuttaa funktionaalista ohjelmointia. Ensimmäisenä ohjelmallisena vaihtoehtona meille tarjotaan ”Properties” taulukkoa, josta voimme valita ennalta määritettyjä funktioita (KUVIO 36). Toisena vaihtoehtona meillä on avata properties ikkunasta laajamittainen editointityökalu nimeltä ”Expression Editor” (KUVIO 37). Expression editorissa voidaan tehdä laajamittaista funktionaalisuuden määrittämistä, joko valmiiksi määritellyillä funktioilla tai kirjoittaa itse ohjelmoitavaan painikkeeseen sopivaa ohjelmakoodia. Itse kirjoitettu ohjelmakoodi kirjoitetaan hyödyntäen Visual Basic -ohjelmointikieltä. Ohjelmoitaessa tag napille sopivaa funktionaalisuutta se toteutettiin hyödyntäen molempia ohjelmointi tapoja.

Kuvio 36 esittää ”Tag” painikkeen ”Properties” ikkunan ja millaisia asetusarvoja voidaan käyttää yksinkertaistetussa esitysmuodossa.



KUVIO 36. Properties lista ”Tag” painikkeen funktionaalisuudelle

Kuvio 37 esittää miltä laajamittainen editointityökalu ”Expression Editor” näyttää. Editointi työkalua käyttämällä ohjelmoitiin painikkeelle ”Tag” toiminto, joka esittää laitteiden positiotunnukset, kun painiketta ”Tag” on painettu (KUVIO 33).



KUVIO 37. Expression Editor ja pieni tosi vai epätosi ohjelmakoodi ”Tag” painikkeelle

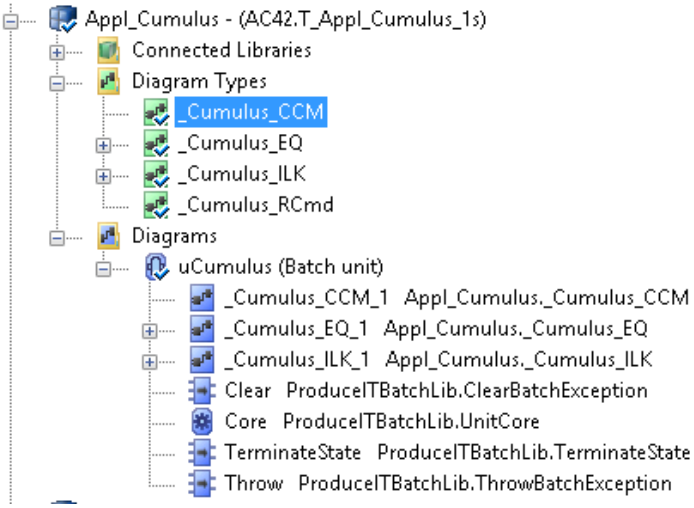
Jotta grafiikkaan saadaan tuotua kaikki oleelliset mittausarvot näkyviin, niille pitää ensin luoda omat muuttujat järjestelmässä, jonka jälkeen on mahdollista liittää kaikki tarvittavat mittaukset ja ohjaukset grafiikan elementteihin. Seuraavaksi tarkastellaan missä näitä muuttujia tehdään, miten ne tehdään, millaista automatiikkaa ne pitävät sisällään ja millaisia raja-arvoja meidän pitää huomioida ohjelmointia tehtäessä.

5.3 Toiminnallisuuden ohjelmoiminen

Laitteiston ohjelmointi aloitetaan luomalla uusi applikaatio ohjelmakirjastoon. Luodun applikaation alle kerätään ensin ”diagram types” toisin sanoen luodaan taulukkotyyppejä (KUVIO 38). Nämä taulukkotyypit pitävät sisällään erilaisia laitekohtaisia tietoja, ja nämä taulukkotyypit pystyvät myös samanaikaisesti keskustelemaan keskenään. Se että taulukkotyypit pystyvät keskustelemaan keskenään vaatii sen, että taulukkotyypissä oleva tietotyyppi on oikeassa muodossa ja kaikki taulukkotyypit on yhdistetty toisiinsa taulukon (diagram) kautta, koska taulukkotyyppi ei voi sisältää toisia taulukkotyyppejä. CABB:n tapauksessa kaikki taulukkotyypit yhdistyvät (Batch unit) taulukon kautta (KUVIO 39).

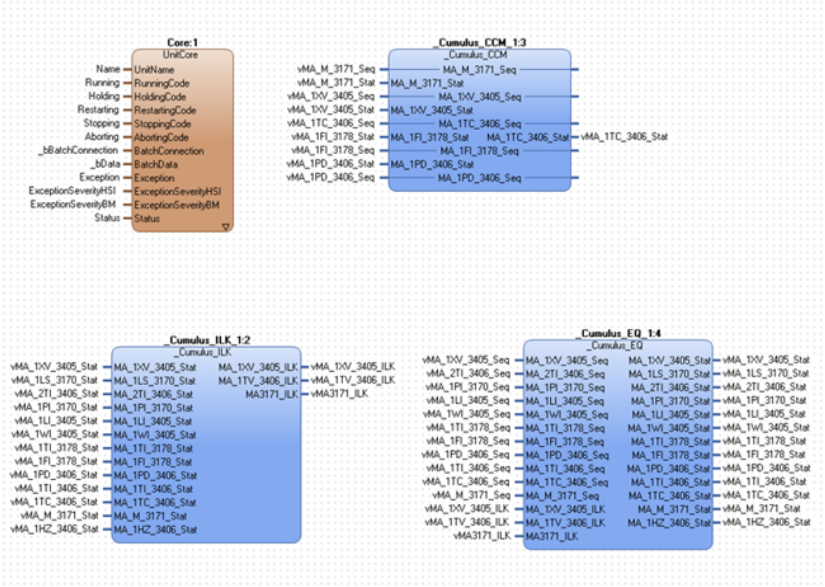
Kaaviotyypeissä esiintyvät muuttujien suunnat ovat yleisimmin joko IN tai OUT tyyppisiä. Tämä määrittelee sen, mihin suuntaan tietoa siirretään (KUVIO 42). IN tyyppiseen muuttujaan tuodaan tietoa jostakin taulukon ulkopuolelta, ja vastaavasti OUT tyyppinen muuttuja soveltuu taulukossa olevan tiedon siirtämisen taulukosta toiseen.

Meillä on myös tietotyypin suunta In_Out, joka käyttäytyy identtisesti parametrin by_ref kanssa, kun 800xA generoi yhteyksien välistä koodia. In_Out ja by_ref eroavat siinä, miten niitä käytetään funktiolohkoissa. Poikkeuksetta In-parametrista saa pelkästään lukea tietoa. 800xA-ohjelmisto havaitsee tietotyypin automaattisesti ja päättelee, saako tietotyyppiä muokata vai ei. Siksi on suositeltavaa kuvata tietotyypin suunta oikein ohjelmoitaessa. In_Out tyyppiset parametrit välitetään viitteinä eteenpäin. Joten toimenpiteitä suorittaessa In_Out parametreilla funktiolohkoissa tarkoittaisi toiminnon suorittamista suoraan lohkoon liitettyyn todelliseen muuttujaan. (AC 800M Configuration 2016, 72.) Tästä syystä In_Out tyyppinen muuttuja sopii esimerkiksi HSI-ohjelmointiin, jossa muuttujan arvo voidaan esittää suoraan käyttäjälle. Kuvio 42 sisältää muuttujan MA_1TC_3406_HSI, jonka suunta on määritelty tyypiksi In_Out, koska kyseistä muuttujaa käytetään HSI-ohjelmointiin.



KUVIO 38. Applikaation alle luotuja toiminnallisia taulukoita

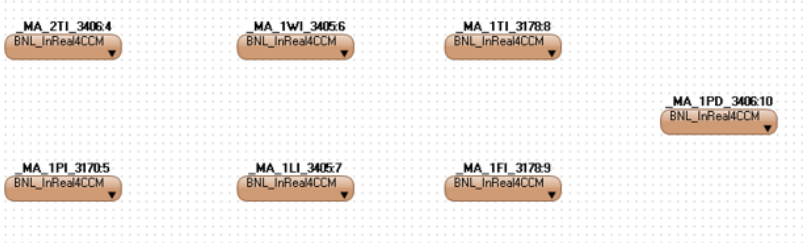
Kuvio 39 esittää miten kaikki taulukkotyypit ovat yhteydessä toisiinsa hyödyntämällä taulukkoa uCumulus.



KUVIO 39. uCumulus (Batch unit) sisältämät taulukkotyypit

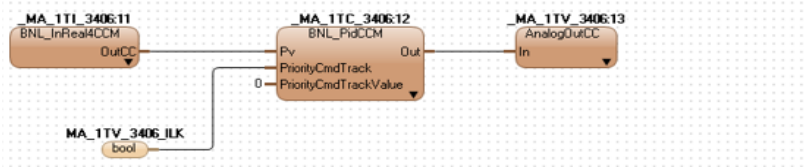
5.3.1 Toimilaitteiden lisääminen ohjelmistoon

Ensimmäisenä tarkastelemme taulukkotyyppiä nimeltä Cumulus_EQ. EQ-taulukko (Equipment) sisältää pääasiassa käytettävään laitteistoon liittyvää tietoa (KUVIO 40). Laitteistoa lajitellaan sopivasti sivuttain rajoitusten, ymmärrettävyyden ja selkeyttämisen vuoksi. Eli kaikki instrumentit, venttiilit, moottorit ja muu laitteisto kirjataan tänne, ja niille määritellään parametrit (KUVIO 42). Näiden parametrien avulla määritellään tiedon tyyppi ja suunta. Parametreillä saadaan myös parametroituja laitekohtaisia hälytyksiä ja raja-arvoja. Parametrilistat muuttuvat sen mukaan mitä parametroidaan.



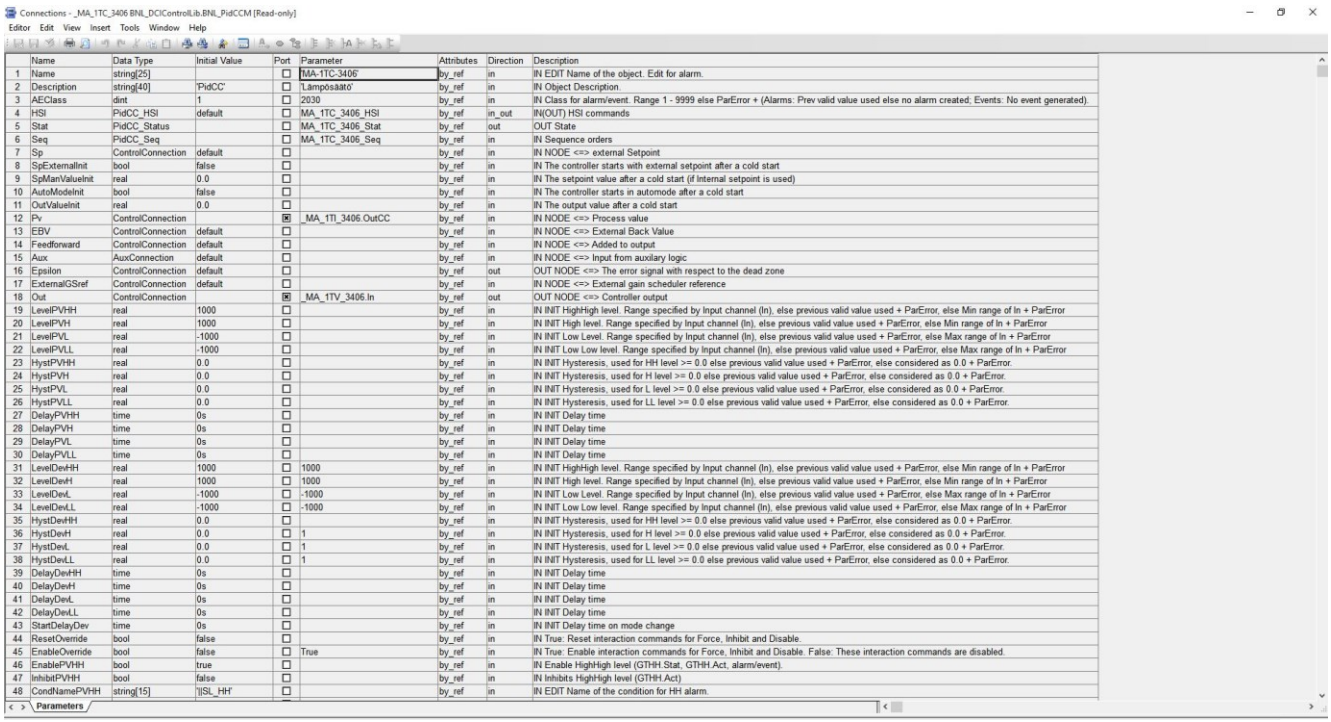
KUVIO 40. EQ-sivu 3 Analogimittaukset

Cumuluksen PID-säätö toteutettiin täysin ohjelmallisesti EQ-taulukkoon sivulle neljä (KUVIO 41).



KUVIO 41. EQ-sivu 4 (PID-säätö) säätöventtiilin kontrollointi suhteessa prosessin lämpötilaan

Seuraava esimerkki havainnollistaa sen millaista tietoa parametrilista voi pitää sisällään. Esitetty parametrilista antaa tietoja esimerkiksi oletusarvoista, esittää valittujen arvojen vaikutuksen ohjelmointiin, esitetäänkö tietoväyläyhteys funktiolohkossa (port), tietotyypin ja tietotyypin suunnan.



Connections - _MA_1TC_3406 BNL_DCIControlLib.BNL_PidCCM [Read-only]

Editor Edit View Insert Tools Window Help

	Name	Data Type	Initial Value	Port	Parameter	Attributes	Direction	Description
1	Name	string[25]		☐	'MA-1TC-3406'	by_ref	in	IN EDIT Name of the object. Edit for alarm.
2	Description	string[40]	'PidCC'	☐	'Lämpösäätö'	by_ref	in	IN Object Description.
3	AEClass	dint	1	☐	2030	by_ref	in	IN Class for alarm/event. Range 1 - 9999 else ParError + (Alarms: Prev valid value used else no alarm created; Events: No event generated).
4	HSI	PidCC_HSI	default	☐	MA_1TC_3406_HSI	by_ref	in_out	IN(OUT) HSI commands
5	Stat	PidCC_Status		☐	MA_1TC_3406_Stat	by_ref	out	OUT State
6	Seq	PidCC_Seq		☐	MA_1TC_3406_Seq	by_ref	in	IN Sequence orders
7	Sp	ControlConnection	default	☐		by_ref	in	IN NODE <=> external Setpoint
8	SpExternalInit	bool	false	☐		by_ref	in	IN The controller starts with external setpoint after a cold start
9	SpManValueInit	real	0.0	☐		by_ref	in	IN The setpoint value after a cold start (if Internal setpoint is used)
10	AutoModeInit	bool	false	☐		by_ref	in	IN The controller starts in automode after a cold start
11	OutValueInit	real	0.0	☐		by_ref	in	IN The output value after a cold start
12	Pv	ControlConnection		☒	_MA_1TI_3406.OutCC	by_ref	in	IN NODE <=> Process value
13	EBV	ControlConnection	default	☐		by_ref	in	IN NODE <=> External Back Value
14	Feedforward	ControlConnection	default	☐		by_ref	in	IN NODE <=> Added to output
15	Aux	AuxConnection	default	☐		by_ref	in	IN NODE <=> Input from auxilary logic
16	Epsilon	ControlConnection	default	☐		by_ref	out	OUT NODE <=> The error signal with respect to the dead zone
17	ExternalGSref	ControlConnection	default	☐		by_ref	in	IN NODE <=> External gain scheduler reference
18	Out	ControlConnection		☒	_MA_1TV_3406.In	by_ref	out	OUT NODE <=> Controller output
19	LevelPVHH	real	1000	☐		by_ref	in	IN INIT HighHigh level. Range specified by Input channel (In), else previous valid value used + ParError, else Min range of In + ParError
20	LevelPVH	real	1000	☐		by_ref	in	IN INIT High level. Range specified by Input channel (In), else previous valid value used + ParError, else Min range of In + ParError
21	LevelPVL	real	-1000	☐		by_ref	in	IN INIT Low Level. Range specified by Input channel (In), else previous valid value used + ParError, else Max range of In + ParError
22	LevelPVLL	real	-1000	☐		by_ref	in	IN INIT Low Low level. Range specified by Input channel (In), else previous valid value used + ParError, else Max range of In + ParError
23	HystPVHH	real	0.0	☐		by_ref	in	IN INIT Hysteresis, used for HH level >= 0.0 else previous valid value used + ParError, else considered as 0.0 + ParError.
24	HystPVH	real	0.0	☐		by_ref	in	IN INIT Hysteresis, used for H level >= 0.0 else previous valid value used + ParError, else considered as 0.0 + ParError.
25	HystPVL	real	0.0	☐		by_ref	in	IN INIT Hysteresis, used for L level >= 0.0 else previous valid value used + ParError, else considered as 0.0 + ParError.
26	HystPVLL	real	0.0	☐		by_ref	in	IN INIT Hysteresis, used for LL level >= 0.0 else previous valid value used + ParError, else considered as 0.0 + ParError.
27	DelayPVHH	time	0s	☐		by_ref	in	IN INIT Delay time
28	DelayPVH	time	0s	☐		by_ref	in	IN INIT Delay time
29	DelayPVL	time	0s	☐		by_ref	in	IN INIT Delay time
30	DelayPVLL	time	0s	☐		by_ref	in	IN INIT Delay time
31	LevelDevHH	real	1000	☐	1000	by_ref	in	IN INIT HighHigh level. Range specified by Input channel (In), else previous valid value used + ParError, else Min range of In + ParError
32	LevelDevH	real	1000	☐	1000	by_ref	in	IN INIT High level. Range specified by Input channel (In), else previous valid value used + ParError, else Min range of In + ParError
33	LevelDevL	real	-1000	☐	-1000	by_ref	in	IN INIT Low Level. Range specified by Input channel (In), else previous valid value used + ParError, else Max range of In + ParError
34	LevelDevLL	real	-1000	☐	-1000	by_ref	in	IN INIT Low Low level. Range specified by Input channel (In), else previous valid value used + ParError, else Max range of In + ParError
35	HystDevHH	real	0.0	☐		by_ref	in	IN INIT Hysteresis, used for HH level >= 0.0 else previous valid value used + ParError, else considered as 0.0 + ParError.
36	HystDevH	real	0.0	☐	1	by_ref	in	IN INIT Hysteresis, used for H level >= 0.0 else previous valid value used + ParError, else considered as 0.0 + ParError.
37	HystDevL	real	0.0	☐	1	by_ref	in	IN INIT Hysteresis, used for L level >= 0.0 else previous valid value used + ParError, else considered as 0.0 + ParError.
38	HystDevLL	real	0.0	☐	1	by_ref	in	IN INIT Hysteresis, used for LL level >= 0.0 else previous valid value used + ParError, else considered as 0.0 + ParError.
39	DelayDevHH	time	0s	☐		by_ref	in	IN INIT Delay time
40	DelayDevH	time	0s	☐		by_ref	in	IN INIT Delay time
41	DelayDevL	time	0s	☐		by_ref	in	IN INIT Delay time
42	DelayDevLL	time	0s	☐		by_ref	in	IN INIT Delay time
43	StartDelayDev	time	0s	☐		by_ref	in	IN INIT Delay time on mode change
44	ResetOverride	bool	false	☐		by_ref	in	IN True: Reset interaction commands for Force, Inhibit and Disable.
45	EnableOverride	bool	false	☐	True	by_ref	in	IN True: Enable interaction commands for Force, Inhibit and Disable. False: These interaction commands are disabled.
46	EnablePVHH	bool	true	☐		by_ref	in	IN Enable HighHigh level (GTHH.Stat, GTHH.Act, alarm/event).
47	InhibitPVHH	bool	false	☐		by_ref	in	IN Inhibits HighHigh level (GTHH.Act)
48	CondNamePVHH	string[15]	'||SL_HH'	☐		by_ref	in	IN EDIT Name of the condition for HH alarm.

Parameters

KUVIO 42. Parametrointilista PID-säädölle MA-1TC-3406

5.3.2 Lukitusten ohjelmointi

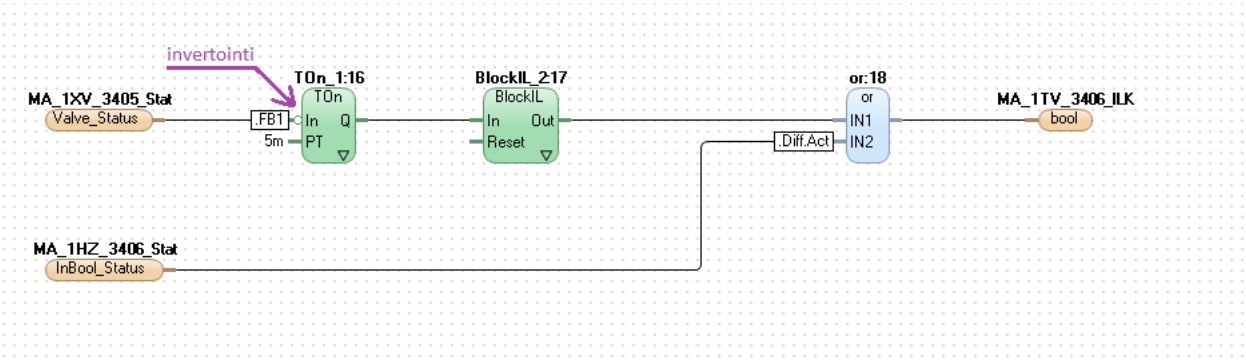
Seuraava taulukkotyyppi, jota tarkastellaan, on Cumulus_ILK (InterLock). Cumulus_ILK pitää sisällään turvallisuuteen liittyvän ohjelmoinnin. Jokainen lukitusta tarvitseva kohde luodaan omaksi kokonaisuudeksi omalle sivulleen. Näitä sivuille kasattuja kokonaisuuksia määrittää tietyt rajoitukset.

800xA-ohjelmisto toteuttaa tämän vaiheen hyödyntämällä FD ohjelmointia eli ”Function Diagram” tyyliä. FD ohjelmoinnissa käytetään niin sanottuja funktiolohkoja, joita yhdistelemällä saadaan luotua

ohjelmoitavaan ohjelmaan sopivia ehtoja. CABB:n järjestelmässä funktiolohkoille on asetettu 30 funktion maksimi määrä per sivu, joten meidän ei kannata käyttää funktiolohkoja joka vaiheessa. Esimerkiksi invertointia varten on olemassa erillinen funktio, mutta se söisi nopeasti meidän käytettävissä olevat funktiolohkot, jos ohjelmassa on paljon invertoitavaa. Tässä tapauksessa invertointi kannattaisi tehdä suoraan valittuun haaraan. Invertointi jättää pienen ”pallon” merkiksi siitä, että kyseinen haara on invertoitu (KUVIO 43).

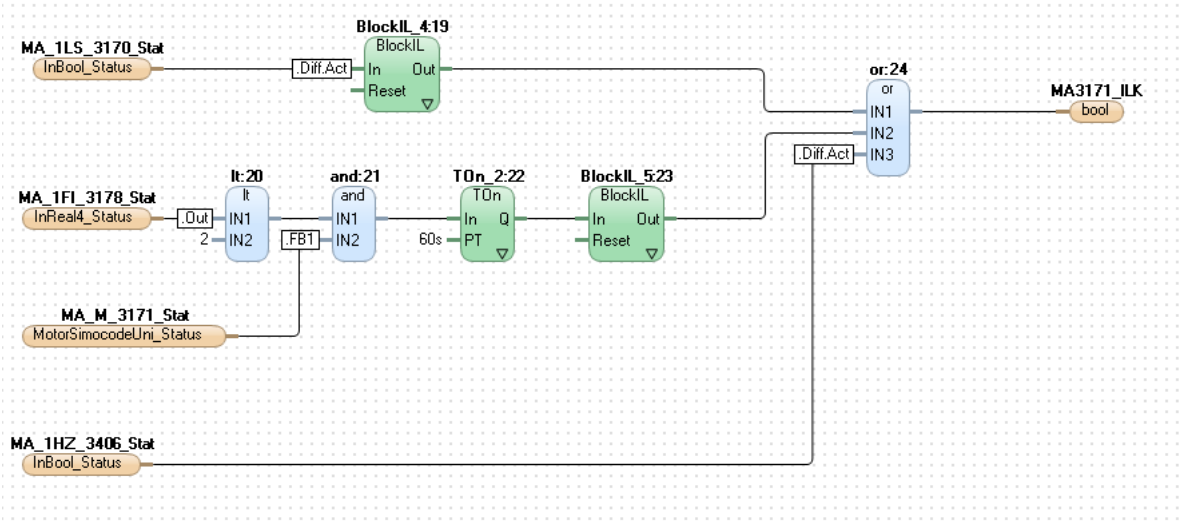
Binääritulojen tapauksessa on valittu funktio ”Diff.Act”, joka tutkii tilan muutosta suhteessa normaalitilaan 800xA-järjestelmässä. Tämä on erittäin tärkeää, koska meidän pitää tietää, jos kytkin tai muu turvatoimintoja sisältävä laite vaurioituu, jotta se saadaan vaihdettua. Näin varmistetaan, että hätätapauksissa meillä on aina toimiva turvajärjestelmä (teoriassa). Seuraavaksi käydään läpi lukituksiin liittyvää ohjelmointia ja esitetään ohjelman toiminto.

1. Säätöventtiili (MA_1TV_3406) suljetaan, jos hätäseis-painike (MA_1HZ_3406) muuttaa tilaansa tai sulkuventtiili (MA_1XV_3405) on ollut suljettuna vähintään 5 minuuttia
2. Funktio BlockIL antaa operaattorille mahdollisuuden yliajaa/kirjoittaa lukituskäskyn



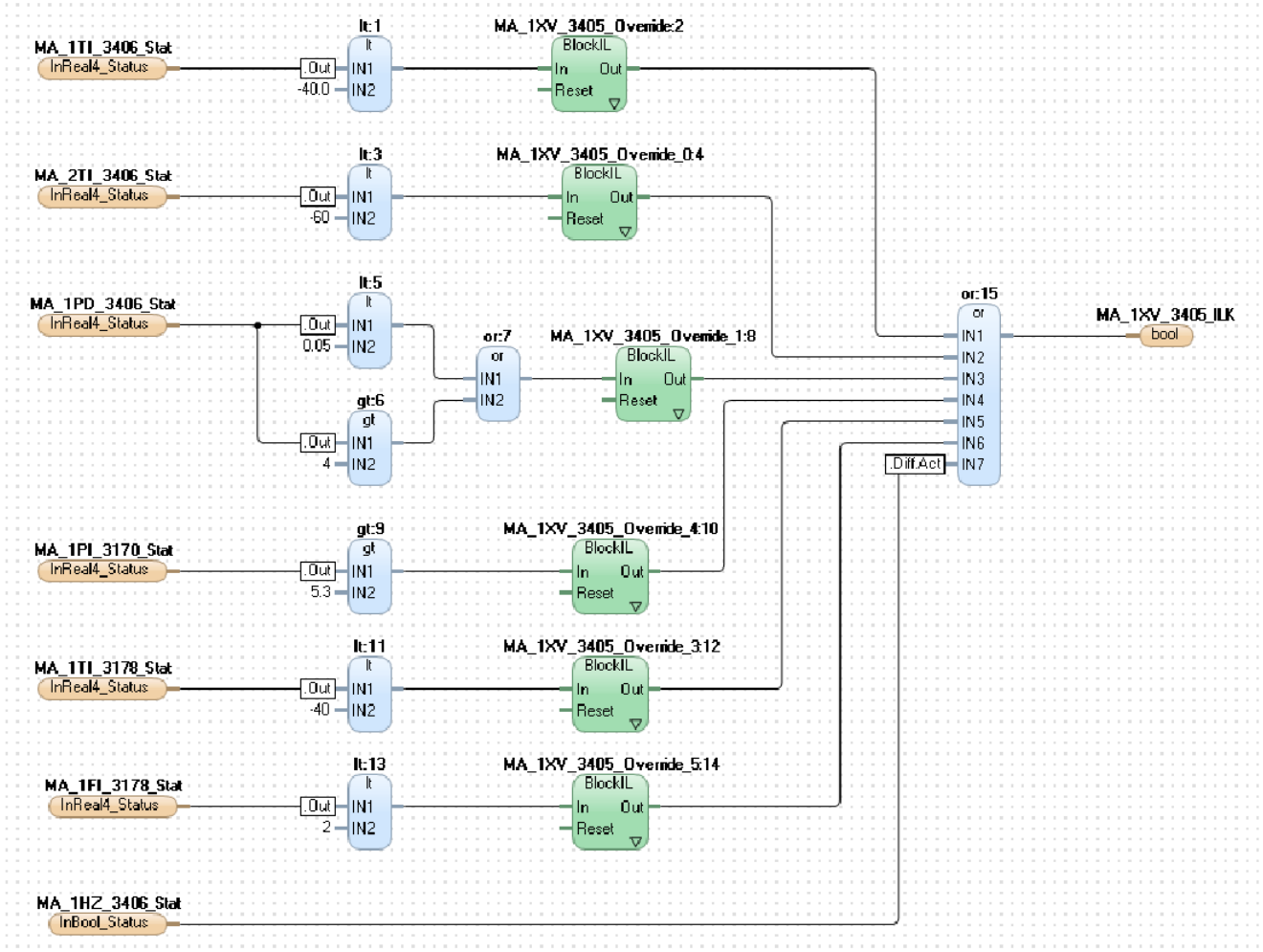
KUVIO 43. Säätöventtiilin turvalukitus

1. Moottori (MA3171) lukitaan, jos hätäseis-painike (MA_1HZ_3406) muuttaa tilaansa tai jos paisuntasäiliön (MA_1LS_3170) alarajakytkin muuttaa tilaansa
2. Moottori (MA3171) lukitaan, jos moottori on päällä ja virtaus on ollut 60 s pienempi kuin 2 m³/h



KUVIO 44. Moottorin turvalukitus

1. Sulkuventtiili MA_1XV_3405 suljetaan jos hätäseis-painike (MA_1HZ_3406) muuttaa tilaansa
2. Sulkuventtiili MA_1XV_3405 suljetaan jos (MA_1TI_3406) on vähemmän kuin - 40 c°
3. Sulkuventtiili MA_1XV_3405 suljetaan jos (MA_2TI_3406) on vähemmän kuin - 60 c°
4. Sulkuventtiili MA_1XV_3405 suljetaan jos (MA_1PD_3406) on vähemmän kuin 0,05 bar tai enemmän kuin 4 bar
5. Sulkuventtiili MA_1XV_3405 suljetaan jos (MA_1PI_3170) on vähemmän kuin 5,3 barg
6. Sulkuventtiili MA_1XV_3405 suljetaan jos (MA_1TI_3178) on vähemmän kuin - 40 c°
7. Sulkuventtiili MA_1XV_3405 suljetaan jos (MA_1FI_3178) on vähemmän kuin 2 m³/h



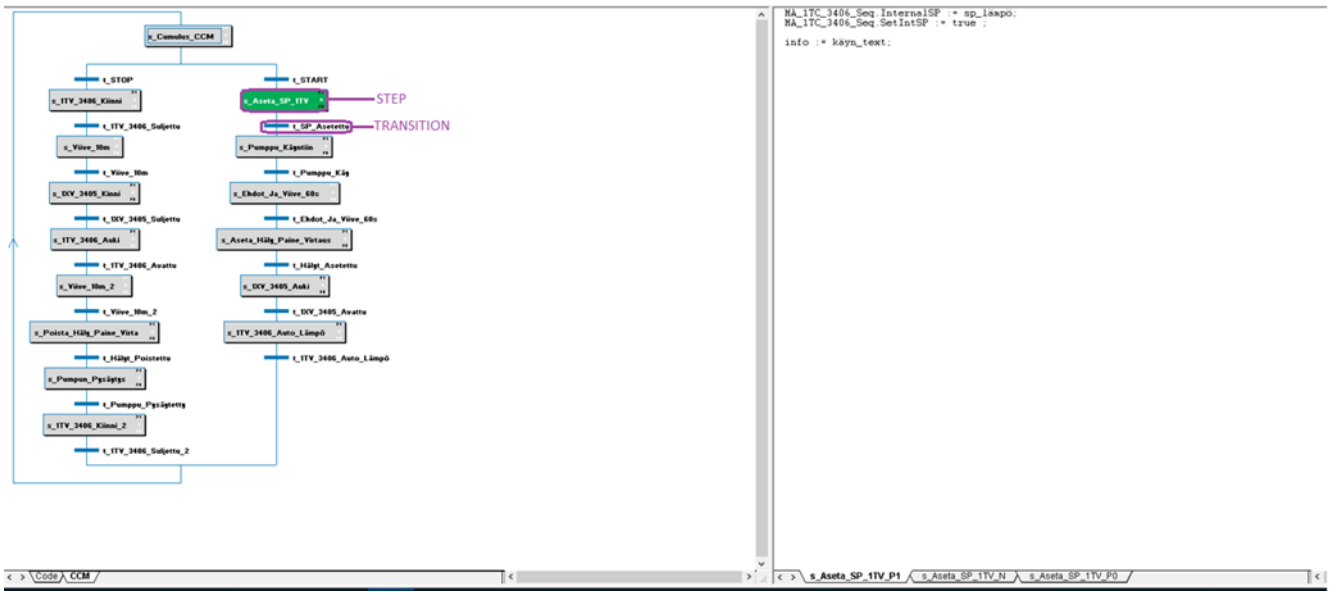
KUVIO 45. Sulkuventtiilin MA_1XV_3405 turvalukitus

5.3.3 Ohjattu käynnistyminen ja pysyttäminen käyttäen SFC:tä

Jäljelle jäävä taulukkotyyppi on Cumulus_CCM. Tähän SFC (Sequential Function Chart) taulukkoon luodaan sekvenssillä toimivia funktioita. Tässä taulukossa määritellään tarkasti erilaiset toiminta vaiheet, kuten funktion aktivointi, aktivoinnin pysäytys ja tarkistaminen, ennen kuin siirrytään seuraavaan ohjelmalohkoon. Ohjelmakoodin toiminnallisuus kirjoitetaan steppiin (Step). Siirtymäehtoon (Transition) kirjoitettu ohjelmakoodi odottaa ehdon täyttymistä, jotta voidaan siirtyä seuraavaan steppiin. SFC:ssä voidaan käyttää ST-ohjelmointikieltä.

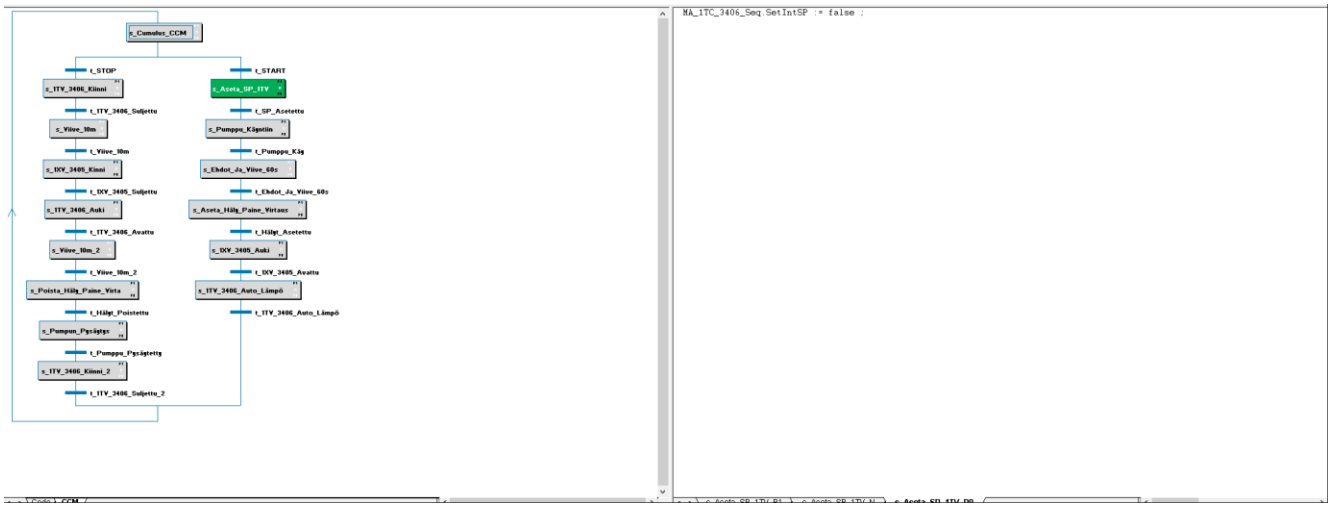
Meidän tapauksessamme ohjelmaan luodun SFC osion tarkoituksena on käynnistää järjestelmä hallitusti ja järjestelmällisesti. Järjestelmän pysäyttäminen toimii samalla periaatteella, eli Cumulus järjestelmä halutaan pysäyttää hallitusti ja järjestelmällisesti, käynnistämättä turhia hälytyksiä pysäytysprosessin aikana. Järjestelmällinen pysäytys pitää järjestelmän käyttökelpoisena pidemmän aikaa, koska järjestelmälle ei aiheudu turhaa rasitusta liian nopeista pysäytyksistä. Jokaiselle stepille on koitettu antaa mahdollisimman kuvaava nimi, jotta operaattorin olisi mahdollisimman helppo päätellä mitä kyseisen stepin aikana tapahtuu. Tämän työn osalta steppien nimistä on todella helppo päätellä, mitä kyseinen steppi tekee kussakin kohdassa, koska stepeissä ei ollut useita erilaisia muuttujia, mikä helpotti lyhyiden ja kuvaavien nimien luomista. Seuraavaksi käydään läpi lyhyt esimerkki siitä, kuinka SFC ohjelmointi toimii käytännössä.

1. Aseta muuttujan ”MA_1TC_3406” tavoite arvoksi muuttujan ”sp_lämpö” arvo (Set Point Value)
2. Hyväksy muuttujaan ”MA_1TC_3406” asetettu tavoitearvo
3. Kirjoita muuttujaan ”info” muuttujan ”käyn_text” sisältö



KUVIO 46. Set Point Value asetus ja käynnistys tekstin kirjoittaminen

1. Lopeta Set Point Valuen asettaminen, kun poistutaan stepistä



KUVIO 47. Ohjelmakoodin suorittaminen stepistä poistuttaessa

1. Odota kunnes muuttujan ”MA_1TC_3406” arvo on yhtä kuin ”sp_lämpö”

KUVIO 48. Siirtymä vaiheelle asetettu ehto

5.4 Ohjelmoinnin ulkopuolelle jääviä asioita

Lähestulkoon kaikki Cumuluksen entinen laitteisto ja instrumentaatio saatiin lisättyä uuteen ohjelmaan. Ohjelmoinnin ulkopuolelle jäi PZA-031 instrumentille menevän ilmanpaineen määrän mittaaminen ja ulkoista lämpötilaa mittaava TICZA-015. PZA-031 mittaus ei ole järjestelmälle kriittinen, koska jos ilmansyötössä on jotain vikaa säätöventtiili ei pääse ohjautumaan oikein ja tämä aiheuttaa hälytyksen typen tai kierrossa olevan tuotteen lämpötilan liiallisesta laskemisesta, joka johtaa Cumulus laitteiston lukkiutumiseen. Kierron liian korkea lämpötila vastaavasti aiheuttaa hälytyksen mutta ei sulje Cumulus järjestelmää.

TICZA-015 tapauksessa meidän ongelmaksemme muodostuu se, että tähän laitteiseen ei päästä ollenkaan käsiksi koska instrumentti on oletettavasti betonikuution sisällä, joten se joudutaan tämän vuoksi jättämään pois ohjelmoinnista. Instrumentin TICZA-015 tarkoituksena oli suorittaa ulkoista lämpötilan mittausta mittaustarkkuuden lisäämiseksi. Tämä on kuitenkin merkitty operaatio manuaalissa ja PI-kaaviossa valinnaiseksi vaihtoehdoksi, joten TICZA-015 instrumentin pois jättäminen ei vaikuta merkittävällä tavalla järjestelmän normaaliin toimintaan.

6 POHDINTA

Opinnäytetyön tavoitteena oli oppia mahdollisimman paljon uutta automatisoinnista ja ABB 800xA-järjestelmästä sekä suorittaa laadukas Cumulus-laitteiston uudelleenohjelmointi 800xA-järjestelmään. Pääsin omasta mielestäni hyvin asettamiini tavoitteisiin ja sain paljon uutta tietoa lyhyellä aikavälillä.

Graafisesta käyttöliittymästä tuli hyvän näköinen ja selkeä. Kaikki vaadittavat ohjelmalliset toiminnot saatiin myös ohjelmoitua ja testattua toimiviksi. Cumulus-projektin aikana sain paljon uuttaa tietoa 800xA-järjestelmästä ja Function Diagram -mallista. Opin kuinka tärkeää on tehdä selviä ja oikean kokoisia graafisia näyttöjä, jotta operaattorit pystyvät tekemään työnsä huolella. Pääsin näkemään miten ohjelmoinnin eri vaiheet liittyvät toisiinsa ja miltä ohjelmoitava laitteisto näyttää. Sain myös paljon suosituksia ja ehdotuksia hyvistä ohjelmointikäytännöistä, kun työskennellään 800xA-järjestelmän kanssa.

Opinnäytetyön teoriaosuus nitoi hyvin yhteen kaiken sen mitä olin oppinut CABB:llä käytännön työtä tehdessäni. Samalla ymmärrykseni kyseistä järjestelmää kohtaa syventyi huomattavasti. Ymmärrän eri ohjelmointimallien väliset rajoitukset, heikkoudet ja vahvuudet. Opin miten turvaluokitukset vaikuttavat eri ohjelmointimallien valintaan, kun tarvitaan tiukempia turvaluokituksia. Samalla olen saanut tietoa siitä miksi turvaluokitukset ovat tärkeitä, mistä ne koostuvat ja kuinka turvaluokitusten eri tasoja voidaan arvioida. Nyt tiedän myös konkreettisesti, miten PHA ja HAZOP liittyvät prosessiteollisuuteen ja miten nämä menetelmät ohjaavat prosessiteollisuutta ja sen kehittämistä.

Ohjelmointi sujui enimmäkseen ilman sen suurempia ongelmia, kunhan perusasiat järjestelmästä saatiin hallintaan. Tietopohjassa oli puutteita, mikä hankaloitti asiaan perehtymistä etukäteen, mutta keräämäni tiedot CABB:ltä helpotti tutkimuksen tekemistä ja tietojen yhdistämistä oikeisiin asioihin.

Työssä ei varsinaisesti ollut aikarajaa, koska järjestelmä otetaan käyttöön vasta vuonna 2024. Tästä huolimatta työn käytännön osuus suoritettiin kahden viikon sisällä. Tämä tarkoittaa myös sitä, että projektia ei päästy kokeilemaan käytännössä ja kaikki vaiheet piti testata simuloimalla. Valitettavasti en pystynyt ennakoivasti tutustumaan projektissa tarvittavaan materiaaliin, koska alustava tietopohja projektista oli rajallinen ja en osannut sanoa kuinka paljon tai miten CABB:n järjestelmä eroaa ”tyypilli-

sestä” 800xA-järjestelmästä. Koulussa olin käynyt kurssin liittyen 800xA-järjestelmän ohjelmoimiseen, mutta kyseinen järjestelmä ei hyödyntänyt Function Diagram -ohjelmointimallia, joten FD:n käyttö piti opetella täysin alusta asti.

HAZOP-kokous oli aluksi vähän kankea, koska läpi käytävään materiaaliin ei tutustuttu ennen kokousta. Opinnäytetyötä tehdessä sain tietää, että olisi ”suositeltavaa” antaa vähintään puoli päivää aikaa kokoukseen osallistuville henkilöille tutustua läpikäytävään materiaaliin, jotta HAZOP on tehokas, kaikki tietävät mistä puhutaan ja vältytään mahdollisilta virheiltä. Onneksi alun jälkeen kokous palautui oikeille raiteille ja päästiin hyvään lopputulokseen.

Kaiken kaikkiaan olen todella tyytyväinen CABB:n tarjoamaan toimeksiantoon ja saavutettuihin tuloksiin.

LÄHTEET

ABB Jokab Safety. Safety Product Handbook. 2019. Saatavissa: https://library.e.abb.com/public/ac78ec67405e42889f1a4003dd111af3/ABB%20Jokab%20Safety_LR.pdf. Viitattu 04.07.2023.

AC 800M Controller Hardware Product Guide. 2019. SYSTEM VERSION 6.1. ABB Ability System 800xA. Saatavissa: https://search.abb.com/library/Download.aspx?DocumentID=3BSE036352-610&LanguageCode=en&DocumentPartId=&Action=Launch. Viitattu 12.07.2023.

AC 800M Binary and Analog Handling. 2016. SYSTEM VERSION 6.0. System 800xA Control. Saatavissa: https://library.e.abb.com/public/01044bb4edcc4608b3a61391722b72fe/3BSE035981-600_A_en_System_800xA_Control_6.0_AC_800M_Binary_and_Analog_Handling.pdf. Viitattu 17.07.2023.

AC 800M Configuration. 2016. SYSTEM VERSION 6.0. System 800xA Control. Saatavissa: https://library.e.abb.com/public/0a5240681b35410d8e24ac55c75ee89c/3BSE035980-600_A_en_System_800xA_Control_6.0_AC_800M_Configuration.pdf. Viitattu 31.07.2023.

AC 800M Planning. 2016. SYSTEM VERSION 6.0. System 800xA Control. Saatavissa: https://library.e.abb.com/public/8cd9c7fa0157456295182124ba53fe19/3BSE043732-600_A_en_System_800xA_Control_6.0_AC_800M_Planning.pdf. Viitattu 13.07.2023.

CABB Oy. 2023. CABB. Esitys. Saatavissa: https://cabb-chemicals.com/. Viitattu. 01.06.2023.

Control and I/O Overview. 2019. ABB Ability System 800xA. Saatavissa: https://library.e.abb.com/public/1a8bb0db5cf4474a95a113f96c18a5c2/3BSE047351%20en%20K%20ABB%20Ability%20System%20800xA%20Control%20and%20IO%20Overview.pdf. Viitattu 12.07.2023.

Engineering Guidelines. 2017. SYSTEM VERSION 6.0. CABB. Saatavissa: https://cabb-chemicals.com/. Viitattu. 17.07.2023.

Minimize Risk Exposure. 2016. PHA-Pro. Esite. Saatavissa: https://pdf4pro.com/cdn/pha-pro-sphera-3a71d5.pdf. Viitattu 28.07.2023.

OAMK. 2009. AUTOMAATIOTEKNIIKKA I. Kurssimateriaali. Saatavissa: http://www.tek-niikka.oamk.fi/~terohi/auto1_s2009u.htm#_Toc147132883. Viitattu 08.06.2023.

Process Graphics. 2016. SYSTEM VERSION 6.0. System 800xA Engineering. Saatavissa: https://library.e.abb.com/public/1bfa63553a6b4640a5a521685cf3036b/3BSE049230-600_B_en_System_800xA_Engineering_6.0_Process_Graphics.pdf. Viitattu 26.07.2023.

SFS-EN ISO 10628-2. DIAGRAMS FOR THE CHEMICAL AND PETROCHEMICAL INDUSTRY. PART 2: GRAPHICAL SYMBOLS. 2012. Helsinki: Suomen standardisoimisliitto SFS. Viitattu. 12.06.2023

SFS-EN IEC 62061. Koneturvallisuus. Turvallisuuteen liittyvien ohjausjärjestelmien toiminnallinen turvallisuus. 2021. Helsinki: Suomen standardisoimisliitto SFS. Viitattu. 06.07.2023.

Sivonen, M. 2008. Teollisuuden instrumentointi – Rakenne ja suunnittelu. Helsinki: AEL.

Steiner, M. 2013. Control Logic at your fingertips. The Control Builder Diagram Editor. Esitys. Saatavissa: https://library.e.abb.com/public/47797d997bcc4dcb91a5bebae8475eaf/3BSE074022_en_2013-Apr_NEU_Partner_Day_-_Diagram_Editor_oversikt.pdf?x-sign=gVxaRGl5FvMDIQ46BEgbN+Iig0Ek+agGIxIB6F8CQXMIO+oGsQKaAEICrcHmQo12. Viitattu 19.07.2023.

Sutton, I. 2010. Process Risk and Reliability Management – Operational Integrity Management. USA: Elsevier.

System Planning. 2016. SYSTEM VERSION 6.0. System 800xA. Saatavissa: https://library.e.abb.com/public/fc7813f0a7c647599e33e9e6c7b930b2/3BSE041389-600_B_en_System_800xA_6.0_System_Planning.pdf. Viitattu 25.07.2023.

Turton, R., Shaeiwitz, J., Bhattacharyya, D & Whiting, W. 2018. Analysis, Synthesis, and Design of Chemical Processes. 5. painos. Pearson.

What is the Difference Between PHA, HAZOP & LOPA? 2022. Sanasto. Saatavissa: https://sphera.com/glossary/what-is-the-difference-between-pha-hazop-lopa/. Viitattu 29.07.2023.