Eetu Seppänen

Kaivosveden puhdistuskontin ohjausjärjestelmän suunnittelu

Insinööri

Konetekniikka

Kevät 2024

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Seppänen Eetu

Työn nimi: Kaivosveden puhdistuskontin ohjausjärjestelmän suunnittelu

Tutkintonimike: Insinööri (AMK), Konetekniikka

Asiasanat: ohjausjärjestelmä, automaatio, vedenpuhdistus

Kaivosveden puhdistuskontti on osa EAKR-rahoitteista (Euroopan aluekehitysrahasto) WaterPro-hanketta. KAMK on osana hanketta muun muassa Kokkolan yliopistokeskuksen sekä Oulun yliopiston kanssa. Opinnäytetyökseni teen hankkeen jäteveden puhdistuskonttiin ohjausjärjestelmän. Jäteveden puhdistuskontissa jätevettä puhdistetaan sekä pussisuodattimella että puhdistuskolonneilla. Pussisuodatin toimii jatkuvatoimisen painesuodattimen periaatteella, kun taas puhdistuskolonnit toimivat sekvenssiajoisella adsorptioperiaatteella. Yhteensä puhdistuskonttiin kuuluu noin 30 kenttälaitetta, joita joko ohjelmallisesti luetaan, pyöritetään tai säädetään. Kenttälaitteilla tarkoitetaan tässä kohdassa antureita, venttiilejä jne.

Itse ohjausjärjestelmän tarkoituksena on ohjata kontin prosessia automaattisesti, eli kääntää venttiilejä, ohjata pumppuja ja lukea anturidataa ilman ihmisen vaikutusta muuten kuin Start-nappia painettaessa. Ohjausjärjestelmän ohjelmointi suoritettiin Beckoff Automationin TwinCat 3- suunnitteluohjelmalla, käyttäen pääasiallisesti Structure Text- ohjelmointikieltä.

Ohjausjärjestelmän käyttöliittymä on paneeli-PC:n näytöllä, jonka muistissa myös itse ohjausjärjestelmä tulee olemaan tallennettuna ajon aikana. Paneeli-PCltä järjestelmää voidaan ajaa sekä säätää tarvittaessa. Itse ohjausjärjestelmä rakentuu lukuisista eri askelista (myöhemmin stepeistä), laskureista (nCountereista), ajastimista (timers) sekä sisäisistä muuttujista (variables), joille on määritelty jokin fyysinen suure tai kenttälaite. Sisäisillä muuttujilla, laskimilla sekä muilla parametreillä ohjelma saadaan siirtymään askeleesta toiseen halutulla tavalla.

Itse ohjausjärjestelmää ei saatu koeponnistettua aikataulun puitteissa. Keskeisimpänä tuotoksena opinnäytetyön aikana on itse ohjausjärjestelmä, sekä tietenkin tämä kirjallinen tuotos. Itse hanke on jo loppunut, joten jatkokehityskohteita tuotokselle on hankala määrittää.

KAMK • University of Applied Sciences

Abstract

Author(s): Seppänen Eetu

Title of The Publication: Designing a control system for the container of wastewater purification

Degree Title: Bachelor of Engineering, Mechanical and production engineering

Keywords: control system, automation, water purification

Container for wastewater purification is a part of an ERDF-funded project called WaterPRO (European Regional Development Fund). KAMK is a part of the project along with the university Center of Kokkola and the university of Oulu. For my thesis I'm doing the automated control system for said container. In the container the wastewater is purified by a bag-filter and purification columns. The bag filter operates on the continuous pressure filter principle, whereas the purification in columns is adsorption based and sequenced. All in all, there are about 30 fields devices in the container, which are either read, rotated, or adjusted by the control system. Field devices, meaning in this case sensors and valves etc.

The purpose of the control system is to operate the process within the said container automatically. This includes turning valves, directing pumps, and reading sensor data without any human interaction other than pressing the Start-button. The control system is written/programmed mainly in Structure Text, which is a programming language developed by Beckoff Automation.

The control system operates on a closed control system principle which means that the program diagnoses itself and works fully autonomically. The Human-Machine-Interface aka HMI is on the screen of a panel PC. From the panel-PC you can run and adjust the control system when needed. The control system itself is built out of numerous steps, counters (nCounters), and internal variables. For each variable a certain physical quantity or field device is defined. With said variables, counters, and other parameters the program moves from one step to another in a desired way.

The control system we couldn't test drive in time because of the restraints set by the timetable. The main product of my thesis was the control system and obviously this written part. The project itself has ended, so targets for further development are difficult to define.

Sisällysluettelo

1. Termiluettelo

Adsorbentti= kemikaali, jolla prosessiaineesta sidotaan joitain aineita

Flokki= saostuma, johon on tarkoituksella saostettu esim. metalleja

Flokkulaatio= hienoaineiden saostuminen flokeiksi

Flokkulantti= hienoaineita saostava aine

Function Block= toimilohko eli ohjausjärjestelmän toiminnallinen yksikkö

GVL= Global Variable List eli globaali muuttujalista TwinCat-ohjelmassa

HMI= Human Machine Interface eli käyttöliittymä

Kenttälaite= anturi tai toimilaite, jonka toimintaa ohjataan yleensä logiikalla

Koagulantti= hienoaineiden saostumista edistävä aine

ModBus TCP= Tiedonsiirtoprotokolla, joka kulkee Ethernet-kaapelia pitkin

MTBF= aika laitteen vikaantumiseen sen edellisestä alkuperäiseen kuntoon saattamisesta

pH= happamuus eli positiivisten vetyionien aktiivisuus liuoksessa

PLC= Programmable logic controller eli ohjelmoitava logiikka

pmy/ml= pesäkettä muodostavaa yksikköä per millilitra

Polymeeri= orgaaninen, huonosti liukeneva suurimolekyylinen aine

RTU= Remote Terminal Unit eli etäterminaaliyksikkö

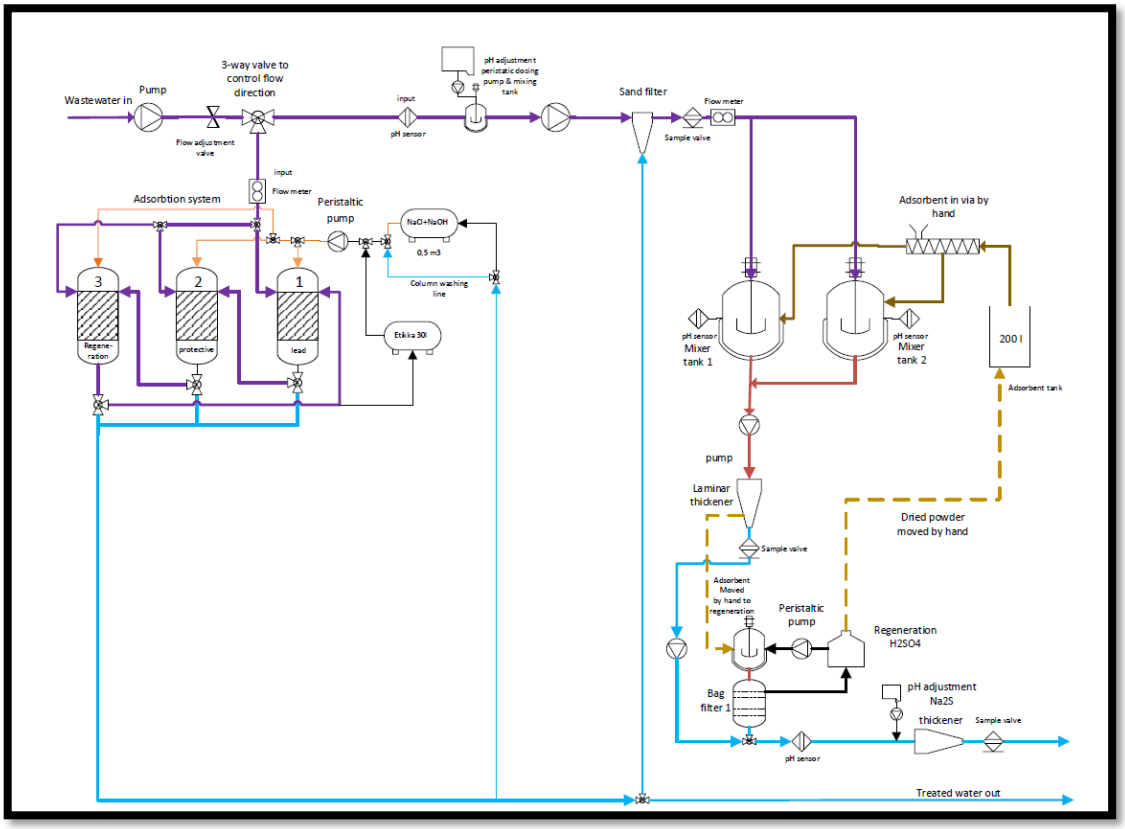
TCP= Transmission Control Protocol eli lähetyksen tiedonsiirtoprotokolla

2. Johdanto

WaterPro on EAKR-rahoitteinen hanke (Euroopan Aluekehitysrahasto), joka tutkii teollisuuden jätemateriaalien hyödyntämiseen liittyviä ratkaisuja. Hanke on alkanut vuonna 2018 ja KAMK on osallistunut siihen jo edellisvuosinakin. Hankkeessa pyritään ottamaan arvoaineita talteen jätevesistä muun muassa käyttämällä teollisia sivuvirtoja absorbentteina ja saostuskemikaaleina sekä käyttämällä hyväksi sähkökemiallista saostamista. Hankkeeseen kuuluu useamman yliopiston ja ammattikorkeakoulun tutkimusryhmiä, muun muassa Kajaanin AMK, jonne teen opinnäytetyöni.

Kajaanin AMK:n projektiryhmä on tehnyt osanaan hanketta kaivosveden puhdistuskontin, jonka ohjausjärjestelmän suunnittelu on opinnäytetyöni aiheena. Itse ohjausjärjestelmä tehdään Beckoff-nimisen yrityksen kehittämällä TwinCat-suunnitteluohjelmalla, tarkemmin TwinCat 3 4024.12:lla. Ohjausjärjestelmän fyysisten komponenttien toimittaja eli Beckoff myös tarjosi meille kevään 2021 aikana ohjelmiston käyttöön koulutuksia, jotka tulivat juuri hyvään aikaan opinnäytetyöprosessini kannalta. Aikaisemmin opinnoissani on hieman käyty ohjausjärjestelmiä läpi, joten tämä opinnäytetyö ei itselleni ollut täydellisesti uuden opettelua kuitenkaan.

Itse puhdistuskontti on melko yksinkertainen kiertojärjestelmä, jossa hyödynnetään useampaa eri suodatusmenetelmää, kuten mekaanista sekä kemiallista suodatusta. Eli jätevesi ajetaan useamman suodattimen läpi alla olevan suuntaa antavan prosessikaavion mukaisesti (kaavio 1), kunnes se on tarpeeksi puhdasta. Prosessilla suodatetaan jätevedestä pois mm. erilaisia metalleja, kuten kuparia. Puhdistusprosessin osana on myös alhainen pH, jota säädetään mm. etikkahapolla. pH:n pitäminen tarpeeksi alhaalla estää eri metallien saostumisen paikkoihin, jonne niiden ei haluta saostuvan. Prosessia tulkitaan eri kenttälaitteilla eli antureilla ja ohjausjärjestelmällä ohjataan pumppuja yms. antureilta saatavan datan mukaisesti.



Kaavio 1. WaterPro –pilot container schematic V4.7 [2]

3. Yleistä vedenpuhdistuksen kemiasta

Kaivoksien jätevesissä on prosessin laadusta, lämpötilasta ja pH:sta riippuen erilaisia epäpuhtauksia. Veden epäpuhtaudet voidaan karkeasti jakaa kuuteen eri osaan, vaikka kaivosvesissä esimerkiksi orgaanisia epäpuhtauksia yleensä onkin vähemmän, koska pH on normaalisti melko kaukana 7:stä eli neutraalista. Tässä osiossa käsittelen tarkemmin tyypillisesti kaivosvesissä esiintyviä epäpuhtauksia ja jätän muut epäpuhtaudet pelkkään listaukseen. Alla oleva listaus ei tietenkään ole absoluuttinen kaikkia epäpuhtauksia kattava listaus, vaan suuntaa antava karkea jako.

1. Liukeneva materiaali (turpiditeetti/sameus=liuenneiden aineiden pitoisuus mg/l) eli esim. liukenevat metallit sekä jotkin alkuaineet sekä suolat [2, 5.6].

2. Liukenematon materiaali eli erilaiset kiintoaineet, kuten hiekka, maa-aines sekä saostuneet metallit. [2, 5.20]

3. Orgaaniset epäpuhtaudet, kuten liuenneet hiiliyhdisteet (DOC=dissolved organic carbon) [2, 5.21]

4. Biologiset epäpuhtaudet, kuten mikro-organismit sekä bakteerit [2, 5.25].

5. Liuenneet kaasut, kuten liuennut happi sekä hiilidioksidi [2, 5.28].

6. Radioaktiiviset materiaalit, kuten radioaktiivinen uraani [2, 5.6].

Seuraavissa osioissa kerron lyhyesti erilaisista kemiallisista reaktioista, joita käytetään ylhäällä mainittujen epäpuhtauksien poistamiseen/vähentämiseen jätevedestä. Keskityn pääasiassa reaktioihin, joita kyseisessä prosessissa käytetään hyväksi, enkä kerro esimerkiksi radioaktiivisten materiaalien poistosta.

Vedenpuhdistuksen kemiasta en kerro tässä yhteydessä kovin laajasti, vaan opinnäytetyössä keskityn enemmän itse ohjausjärjestelmän rakenteeseen. Kuitenkin vedenpuhdistuksen kemiaa on hyvä hieman avata näin opinnäytetyön alussa, koska se osaltaan selventää, miksi tällainen puhdistuskontti on rakennettu, ja miksi teen tämän kyseisen opinnäytetyön.

3.1 Liukenematon materiaali

Kiintoaineiden poistoon käytetään erilaisia suodattimia, joista kyseisessä kohteessa käytetään kahta eri tyyppiä [2, 6.33]:

1. Painovoimaan perustuviin eli. rae- ja hiekkasuodattimet yms.
2. Paineeseen perustuviin eli erilaiset painesuodattimet, suotimet sekä filtterit

3.2 Raesuodattimet

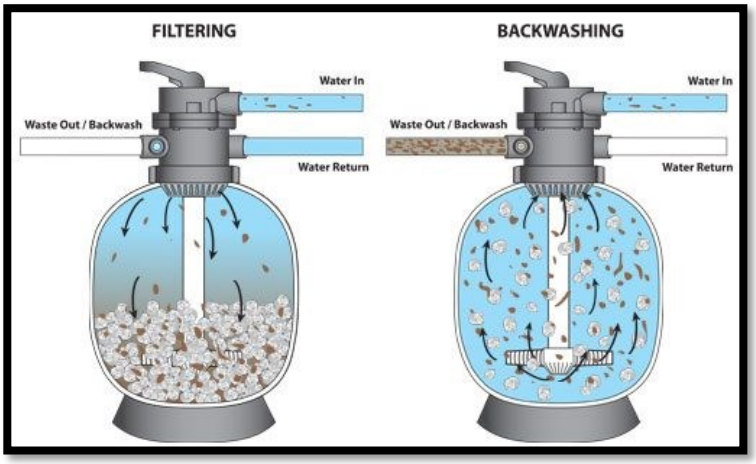
Painovoimaan perustuvia hiekkasuodattimia on kahdenlaisia riippuen siitä, onko suodattimessa pelkkää hiekkaa, jolloin sitä kutsutaan ”single media”-suodattimeksi, vai onko siinä esim. antrasiitti-hiekkaa, jolloin sitä kutsutaan ”dual-media” -suodattimeksi. Kiintoaineiden suodatus tapahtuu näissä suodattimissa samoin kuin luonnollisessa vesikierrossa (sadevesi valuu maa-aineksen läpi pohjaveteen) eli veden kiintoaineet jäävät kiinni hiekkaan samalla, kun itse vesi valuu suodattimen läpi. Raesuodattimien toiminta perustuu ionivaihtoon: riippuen siitä, mitä ainetta halutaan irrottaa, käytetään joko anioni-ionirakeita (negatiivisesti varautuneita) tai kationi-ionirakeita (positiivinen varaus). Tämän takia rae- ja hiekkasuodattimilla poistetaan nesteestä sekä kiinteitä aineita että jo siihen liuenneita aineita. Yleisesti raesuodattimissa voidaan käyttää itse suodatinhiekan sekaan lisättävää flokkulanttia tai koagulanttia suodatuksen parantamiseksi. Nämä aineet siis aiheuttavat flokkulaatiota vedessä, jolloin flokit eli saostumat jäävät helpommin kiinni suodatusmediaan. [2, 6.38.]

3.3 Vastavirtahuuhtelu-metodi

Raesuodattimia täytyy välillä huuhdella esimerkiksi vastavirtahuuhtelu-metodilla. Kuten nimi jo kertookin, kyseisessä huuhtelumetodissa vesi kulkee toiseen suuntaan suodattimessa kuin itse suodatusprosessin aikana. Huuhtelua tehdään sen takia, että raesuodattimen rakeiden pinta ei menisi tukkoon, jonka takia suodatus ei olisi enää tarpeeksi tehokasta. Tämä metodi myös pidentää itse suo-

datusmedian elinikää eli sillä saadaan käsiteltyä enemmän prosessinestettä ennen kuin suodatusmedia joudutaan vaihtamaan. Itse huuhtelun aikana suodattimen suodatusmedia saadaan leijumaan itse suodattimen sisällä, jolloin epäpuhtaudet saadaan tehokkaammin huuhdeltua pois huuhtelumediasta eli tavallisimmin hienojakoisesta hiekasta, johon on sekoitettu suodatusta edistävää lisäainetta. Huuhtelumetodeita on toki vastavirtahuuhtelun lisäksi muitakin, mutta niihin en tässä osiossa perehdy, vaan keskityn itse vastavirtahuuhteluun, koska se on ainoa huuhtelumetodi, jota kyseisessä puhdistuskontissa käytetään. [2, 6.39.]

Vastavirtahuuhteluun käytetään joko pelkkää vettä; vettä, jossa on ilmakuplia; tai seosta, jossa on sekä vettä että ilmaa. Veden ja ilman seos saadaan aikaan johtamalla ilmaa ylipaineella tuloveteen. Kyseisessä kohteessa käytetään pelkkää vettä vastavirtahuuhteluun. Alla näkyvä kaaviokuva (kuva 1) havainnollistaa vastavirtahuuhtelua, eli vasemmalla oleva kuva kuvastaa normaalia suodatustilannetta, jolloin likainen vesi tulee oikealta ylhäältä sisään suodattimeen, poistuu oikealta alhaalta suodattimesta sekä epäpuhtaudet jäävät itse suodattimeen. Vastavirtahuuhtelun aikana jätevedellä huuhdellaan suodatinta, kuten kuvasta näkyy, mutta oleellisena erona on veden kiertosuunta. Veden kiertosuunnan muutoksella suodatinrakeet saadaan leijumaan vedessä. Täten veteen liukenee ionivaihdon avulla suodatusmediasta kiintoaineita, jotka sitten pumpataan niille tarkoitettuun astiaan, jossa kiintoaineet yleensä kuivatetaan jatkokäsittelyä varten. Itse puhdistuskontin vastavirtahuuhtelu ei tietenkään ole täysin samanlainen, vaan kuva on suuntaa antava periaatekuva vastavirtahuuhtelusta.



Kuva 1. vastavirtahuuhtelun periaatekuva [3]

Vastavirtahuuhtelua voidaan tehostaa lisäämällä tuloveteen esimerkiksi polymeeriä, joka toimii joko koagulanttina tai flokkulanttina. Tehostamisen pääasiallinen hyöty on itse huuhtelun nopeuttaminen sekä epäpuhtauksien nopeampi sitoutuminen polymeeriin eli siis rakeiden pinnan nopeampi puhdistuminen.

3.4 Painesuodattimet

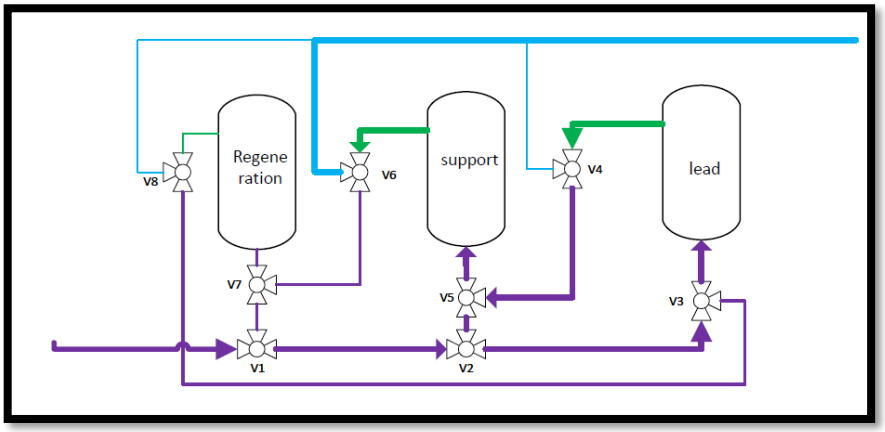
Painesuodattimien sekä -suotimien pääasiallinen ero raesuodattimiin on luonnollisesti puhdistettavan aineen suurempi tulopaine sekä tilavuusvirta. Painesuodattimissa ei myöskään yleensä ole väliainetta, vaan yleensä pelkkä useampi sarjassa oleva suodatin [2, 6.46]. Itse suodattimen suodattavat osiot voivat olla esimerkiksi kumimattoja tai hienoja siivilöitä.

Painesuodattimilla on myös se etu raesuodattimiin nähden, että kiintoaineen talteenotto on yleensä helpompaa. Painesuodattimia käytetäänkin tämän takia myös esimerkiksi prosessinesteiden kuivatuksessa, kun prosessinesteestä halutaan poistaa vesi ja saada kiintoaine talteen.

3.5 Suodatus puhdistuskontissa (adsorptio kolonneissa)

Adsorptio tarkoittaa sitä, että epäpuhtaudet sitoutuvat suodatinrakeiden pintaan, kun vesi ajetaan sen läpi. Kolonneille otetaan vedestä talteen ammoniayhdisteitä, kun taas pussisuotimella otetaan vedestä talteen metalleja. Ammoniayhdisteillä tarkoitetaan tässä yhteydessä typpipohjaisia ioneja. Pussisuodatin toimii jatkuvatoimisen raesuodattimen periaatteella, kun taas puhdistuskolonnit toimivat sekvenssiajoisen raesuodattimen periaatteella.

Sekvenssiajossa yksi suodatin on aina regeneraatiotilassa ja kaksi muuta kolonnia ovat ”ajossa”, alla olevan kaaviokuvan mukaisesti (kuva 2). Sekvenssin mukaan jätevesi ajetaan ajossa olevien kolonnien läpi vaihtelevassa järjestyksessä, jolloin regeneraatiossa olevan kolonni saa ”levätä”. Sekvenssiajossa jätevesi ajetaan ensin Lead- eli johtokolonnin läpi ja sen jälkeen Support- eli tukikolonnin läpi. Regeneraatio-tilassa olevan kolonnin läpi ajetaan suolaliuosta, jolloin ammonia-yhdisteet adsorptoituvat rakeiden pinnasta suolaliuokseen. Kuvassa 2 paksumman violetit viivat kuvaavat jäteveden kiertosuuntaa ja vihreät suodatetun veden kiertoa.



Kaavio 2. Venttiilien ohjaus kolonneissa [4]

Itse ohjausjärjestelmän suunnittelussa priorisoin pussisuodattimen ”puolta” eli prosessikaaviossa (Kaavio 3) näkyvää oikeaa puolta. Tämä sen takia, koska kyseinen ”puoli” on kriittisempi pilotoinnin kannalta. Myös aikataulutuksen takia priorisointi on tärkeää, jotta projektin aikataulu ei veny minun osaltani liian pitkälle. Kolonnipuolen ohjelma on myös paljon yksinkertaisempi kuin pussisuodattimen puolen ohjelma, joten en sitä lisännyt osion 5: Ohjausjärjestelmän rakenne -kaavioon. Tämä sen takia, että ammonia-puolen ohjelma on komponenteiltaan lähes samanlainen kuin pussisuotimen puolen ohjelman osio. Jokaisen sekvenssin ohjelma on myös melkein samanlainen, paitsi venttiilien numerot vaihtuvat.

3.6 Suodatus puhdistuskontissa (metallien talteenotto pussisuotimella)

Pussisuodattimen toiminta on jatkuvatoimista toisin kuin kolonneissa. Tämä johtuu siitä, että kontissa on ainoastaan yksi pussisuodatin, kun taas kolonneja on kolme. Pussisuodattimen toiminta perustuu siihen, että jätevesi ajetaan kyseisen suodattimen läpi, jolloin jäteveden metallit sekä muut kiintoaineet jäävät suodattimen suodatinpussiin ”talteen”. Kun prosessi saadaan valmiiksi, pussisuodattimen sisällä oleva pussi kuivatetaan sekä siihen tarttuneet kiintoaineet, kuten metallit, otetaan talteen. Pussisuotimella on tarkoitus ottaa vedestä talteen metalleja, kuten rautaa ja kuparia, kun taas kolonneilla vedestä puhdistetaan ammoniayhdisteitä.

3.7 Orgaaniset sekä biologiset epäpuhtaudet

Erilaisia orgaanisia epäpuhtauksia on suuri määrä, ja niitä on tietenkin myös ”puhtaassa vedessä”. Orgaanisten epäpuhtauksien laatu ja määrä riippuvat veden alkuperästä, eli onko se yhdyskuntajätevettä vai esimerkiksi jokivettä. Biologisilla epäpuhtauksilla tarkoitetaan esimerkiksi mikro-organismeja sekä erilaisia bakteereja, joita toki esiintyy puhtaassakin vedessä. Jokivedessä on yleensä enemmän maaperästä lähtöisin olevia epäpuhtauksia, jotka saadaan jo hyvin suodatettua pois muun muassa edellä mainituilla suodatusmetodeilla, kun jokivedestä tehdään juomakelpoista [2, 5.21.].

Tässäkin tapauksessa veteen pitää kuitenkin lisätä neutralointikemikaaleja, jotta epäpuhtauksien pitoisuudet saadaan tarpeeksi matalalle tasolle, kuten jos vedestä olisi saatava ihmisravinnoksi kelpaavaa. Näille pitoisuuksille on useita eri turvarajoja, jotka riippuvat esimerkiksi veden laadusta, veden alkuperästä sekä geologisesta sijainnista. Tästä esimerkkinä Suomen terveysministeriön asetus, jonka mukaan talousvedessä saa olla esim. entrokokkeja 0 pmy/250 ml [1].

Muitakin rajoituksia toki on muidenkin pitoisuuksien suhteen, sekä pH:n ja sameuden suhteen mutten niistä nyt seikkaperäisesti kerro, koska ne eivät kuulu tähän aiheeseen. Tämän esimerkin tarkoituksena oli lähinnä havainnollistaa sallittuja turvarajoja yleisesti, eli kuinka matalat pitoisuudet vedenpuhdistuksella on tarkoitus saavuttaa, ja miten ne riippuvat erittäin paljon käyttökohteesta.

4 Yleistä ohjausjärjestelmästä

Ohjausjärjestelmä on ohjelmoitavan logiikan muistissa oleva ohjelma, jonka tehtävänä on saada itse prosessi/laite toimimaan itsenäisesti halutulla tavalla ja tarpeeksi luotettavasti ilman jatkuvaa ihmisen vaikutusta. NEMA (National Electrical Manufacturers Associaton) määrittelee ohjelmoitavan logiikan seuraavasti:

Digitaalisesti toimiva elektroninen laite, joka käyttää ohjelmoitavaa muistia käskyjen sisäiseen tallentamiseen sellaisten erikoistoimintojen toteuttamiksesi, kuten logiikka, sekvenssi, ajastus, laskuri ja aritmetiikka, jotta voitaisiin ohjata digitaalisilla tai analogisilla tulo/lähtömoduuleilla erilaisia koneita tai prosesseja. Digitaalinen tietokone, jota käytetään suorittamaan ohjelmoitavan logiikan tehtäviä, luetaan kuuluvaksi tähän ryhmään. Pois on luettava rumputyyppiset ja muut vastaavat mekaaniset sekvenssiohjelmat. [3, s. 95.]

Ohjausjärjestelmät voidaan jakaa karkeasti avoimiin ja suljettuihin ohjausjärjestelmiin. Näiden erona on se, että suljetussa ohjausjärjestelmässä on käytössä takaisinkytkentä eli järjestelmä ”tulkitsee” koko ajan itse itseään [4, s. 4]. Avoimia ohjausjärjestelmiä käytetään enemmän laitteissa ja koneissa, jotka eivät ole jatkuvatoimisia eli ohjelma keskeytetään työkierron jälkeen esimerkiksi työkappaleiden lisäyksen tai poiston ajaksi. Puhdistuskontin ohjausjärjestelmä on toteutettu suljetun ohjausjärjestelmän periaatteella eli se on pääosiltaan jatkuvatoiminen, kuten myöhemmin tulee selville.

Ohjausjärjestelmän rakennetta voi lähestyä myös Master-Slave-ajattelun kautta, eli tässä tapauksessa PLC on Master-komponentti eli se antaa käskyjä Slave-komponenteille eli yksittäisille kenttälaitteille. Tämän hierarkian takia yksittäinen kenttälaite ei voi vaikuttaa Master-komponentin toimintaan, jollei sitä ole erikseen ohjelmoitu niin tekemään. Tästä esimerkkinä hätäpysäytys -nappi tai joku muu turvalaite. Tätä hierarkiaa voi myös selventää siten, että Slave-laitteilla tulkitaan itse prosessia, ja Master-komponentti ”päättää”, mitä kyseisellä anturidatalla tehdään.

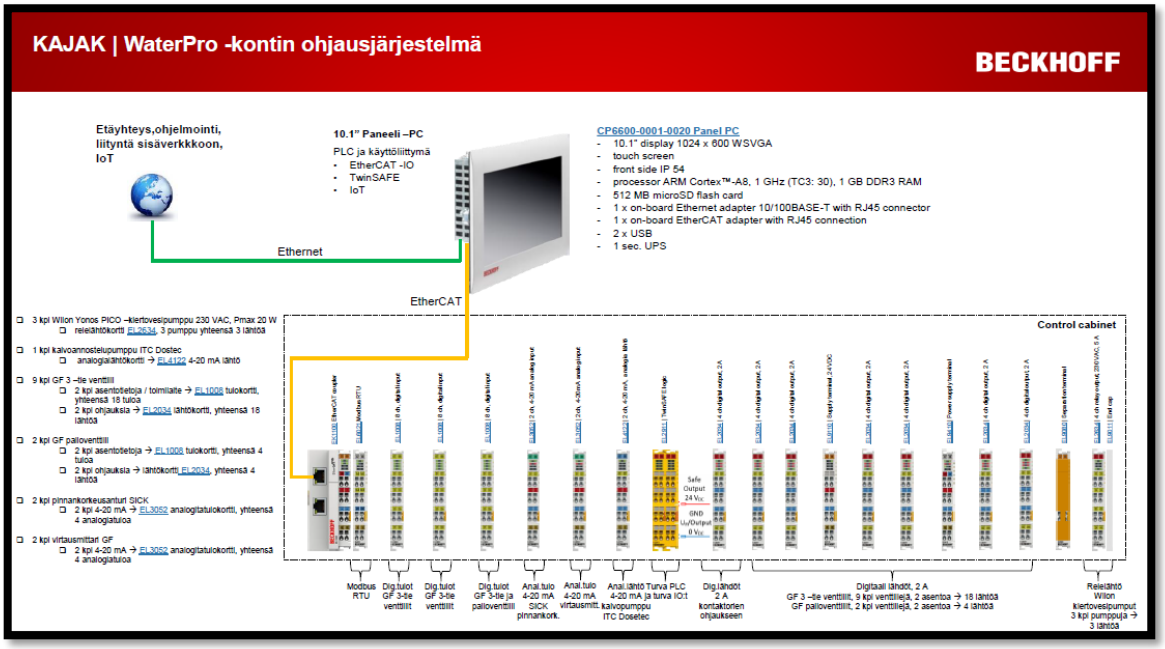
Itse PLC-kortteja puhdistuskontissa on 11 eri mallia (kuva 2) ja jokaisella kortilla on oma tehtävänsä, eli niillä luetaan joko anturidataa tai ohjataan tiettyä moottoria tai venttiiliä. Logiikassa on kaksi turvakorttia, joista toinen (EL9080), valvoo logiikan sisäistä tietoliikennettä. Toiseen turvakorttiin (El2911) on kytketty fyysinen hätäpysäytys -nappi, josta koko prosessin saa tarvittaessa pysäytettyä.

Tähän turvakorttiin on liitetty myös hätäpysäytyksen kuittausnappi, jolla prosessin saa takaisin päälle Step_INITistä eli ohjelman alusta. Tai varsinaisesti itse hätäpysäytyksen kuittausnappi ei ohjelmaa resetoi, vaan se resetoi pelkän hätäpysäytys-toimintalohkon. Hätäpysäytys -toimintalohkosta kerron tarkemmin kohdassa 5.7 TwinSAFE.

Itse ohjausjärjestelmä on CP-6600-0001-0020 paneeli PC:n muistissa, jonka kautta ohjausjärjestelmään voi tarvittaessa tehdä muutoksia ja josta näkee myös koko prosessin tilan. Ajon aikana itse ohjausjärjestelmään ei voi tehdä muita muutoksia kuin esim. parametrien säätöä tai ohjelman aloitusta. Itse ohjausjärjestelmä kuitenkin suunnitellaan erillisellä koneella, joka voidaan tarvittaessa liittää suoraan PLChen Ethernet-kaapelilla itse ohjelmoinnin helpottamiseksi. Ohjausjärjestelmän perimmäinen tarkoitus on, että prosessia voi ajaa täysin automaattisesti käyttöliittymän kautta, joka tulee näkymään kuvassa näkyvän (kuva 2. keskellä ylhäällä) paneeli-PC:n näytölle.

Itse ohjelma on kirjoitettu/ohjelmoitu Structure Text- ohjelmointikielellä, jota Bechkoffin kehittämä TwinCat 3 4042.2- ohjelmointiohjelmisto tukee. Ohjelma tukee toki muitakin ohjelmointikieliä, mutta tässä opinnäytetyössäni käytän Structure Textiä, koska se on tekstipohjainen, korkean tason ohjelmointikieli, jota TwinCat-ohjelmisto myös tukee. TwinCat-ohjelmistolla ohjelmaa voi muun muassa simuloida, mikä helpottaa ohjelman virheenjäljitystä. Ohjelmassa on myös käytetty toimilohko-ohjelmointia (Function Block-programming) joissakin kohdissa itse ohjelmoinnin helpottamiseksi. Toimilohkoja on ohjausjärjestelmässä useampia, muun muassa TwinSAFE-toimilohko eli toimilohko, jolla valvotaan hätäpysäytys-nappia sekä fbGetLocalSystemTime-toimilohko, jolla ohjausjärjestelmä lukee tietokoneen kellonaikaa sekä päivää.

Alla näkyy Beckhoff Automationin tekemä kaavio ohjausjärjestelmän ja logiikan fyysisestä rakenteesta (kuva 2), eli miten ohjausjärjestelmä kytketään itse prosessiin ja eri kenttälaitteisiin. Prosessin sisäisestä tietoliikenteestä sekä sen vaikutuksesta itse ohjausjärjestelmän rakenteeseen kerron tarkemmin kohdassa 7. Ohjausjärjestelmän integrointi.



Kuva 2. WaterPro järjestelmäkaavio [5]

5 Ohjausjärjestelmän rakenne

Tässä osiossa esittelen yleisesti ohjausjärjestelmän ohjelmallista rakennetta sekä itse ohjelman liittämistä käyttöliittymään ja itse puhdistusprosessin integrointia ohjausjärjestelmään.

Ohjausjärjestelmä voidaan ajatella mustana laatikkona, jolla on jokin tulo ja lähtö. Laatikko on musta, koska emme ole kiinnostuneet siitä, mitä laatikon sisällä tapahtuu, vaan pelkästään tulon ja lähdön relaatiosta. Ohjausjärjestelmällä lähtöä säädetään, eli sille annetaan tietty arvo tai sitä muutetaan tietyllä tavalla. Esimerkiksi lämmitysjärjestelmällä on jokin tietty tavoitelämpötila, kun taas työstökoneen työkalu voi seurata tiettyä työstörataa. [4, s. 1.]

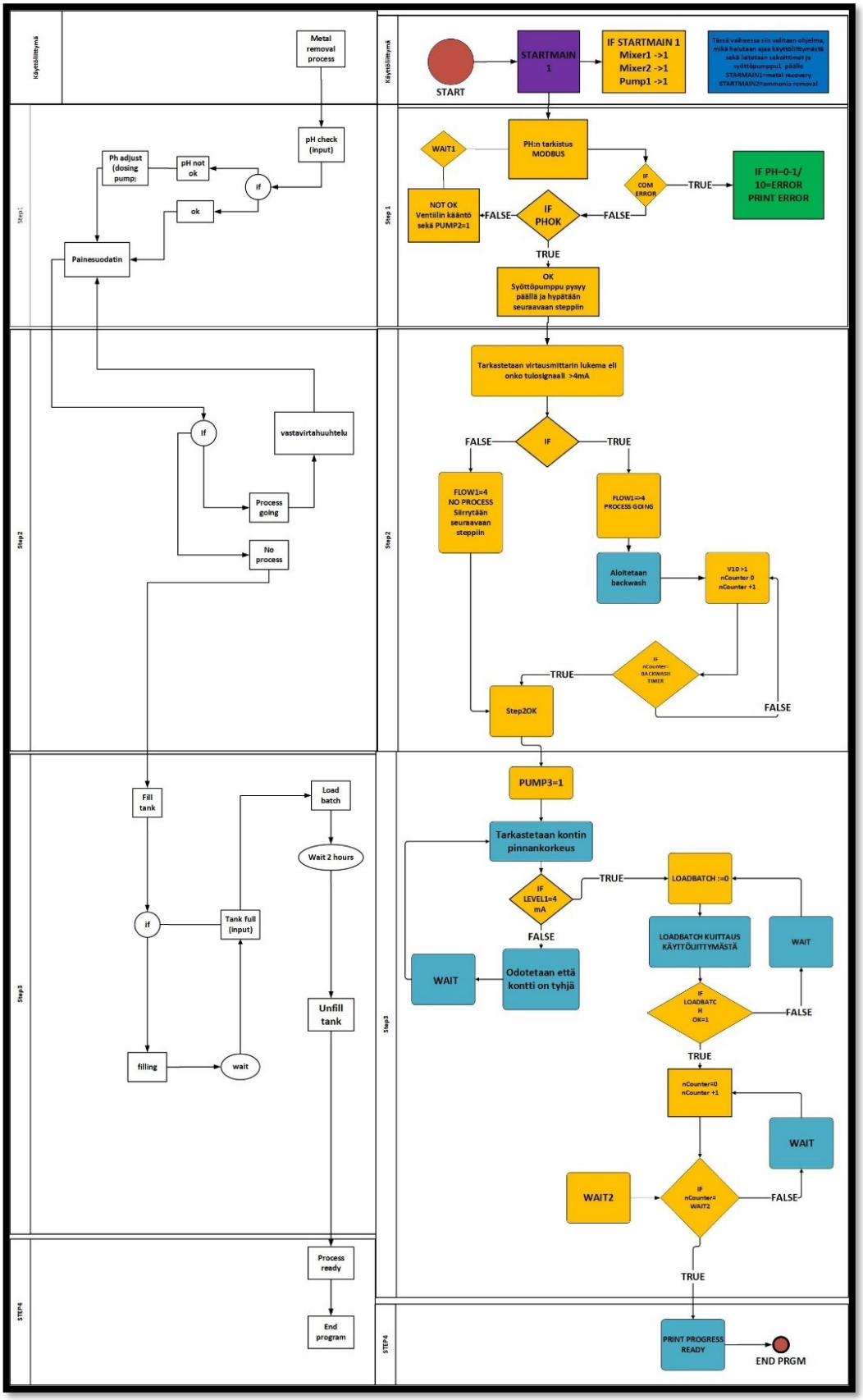
5.1 Ohjelman yleinen rakenne

Oheisessa kaaviossa (kaavio 3) näkyy vasemmassa laidassa prosessin kulku pussisuotimen osalta ja oikealla puolella näkyy itse ohjelman karkea rakenne. Kaavio ei ole koko ohjelman kattava, vaan se kattaa vain osan itse kontin prosessista sekä toiminnoista. Alakohdissa 7.2–7.7 kerron seikkaperäisemmin kunkin Stepin eli askeleen rakenteessa ja tässä kohdassa rakenteesta yleisemmällä tasolla. Structure Textillä tehty ohjelma on jaettu askeliin ohjelmoinnin helpottamiseksi. Alla oleva kaavio (kaavio 3) kattaa pelkästään pussisuotimen ”puolen” ohjelman, koska muuten kaavion luettavuus kärsisi, ja kolonnipuolen ohjelma on rakenteeltaan lähes samanlainen kuin pussisuotimen ohjelma sekä paljon yksinkertaisempi.

Ohjelman askelrakenne perustuu siihen, että jokaisen askeleen sisällä on omat funktionsa, joiden suorittamisen jälkeen siirrytään seuraavaan askeleeseen. Tässä askelrakenteessa on sekin hyvä puoli, että näin on helppoa varmistaa ennen seuraavaan askeleeseen siirtymistä, että prosessi on halutussa vaiheessa, joten prosessi voidaan tarvittaessa pysäyttää esim. vikatilanteessa haluttuun askeleeseen. Askelrakenne takaa myös sen, että ohjelma ei mene itsestään solmuun tai jää junnaamaan paikallaan, kun logiikka ajaa koko ohjelman läpi ensimmäiseltä riviltä viimeiselle alle sekunnissa. Myös uusien komentojen lisääminen ohjelman tiettyyn askeleeseen on tällä rakenteella tehty mahdollisimman yksinkertaiseksi.

Alla olevan kaavion (kaavio 3) ylin osa ”käyttöliittymä” tarkoittaa paneeli-PC:n näytöllä näkyvää visuaalista ohjausjärjestelmän osaa, josta näkee prosessin tilan ja josta prosessi saadaan tarpeen tullessa käyntiin sekä myös pysäytettyä. Käyttöliittymästä käytetään myös nimitystä HMI eli Human Machine Interface. Käyttöliittymästä saadaan myös valittua haluttu aliohjelma eli sekvenssi sekä säädettyä joitain ohjelman parametrejä, kuten sekvenssiaikoja jne. Käyttöliittymän yhdistämisestä varsinaiseen koodiin kerron tarkemmin kohdassa 6.

Kaavioiden 3–8 keltaiset laatikot indikoivat ohjelman sisäisiä funktioita ja vihreät ovat käyttöliittymän osia. Tumman- ja vaaleansiniset laatikot ovat pelkkiä havainnollistavia kommentteja lukijalle, joilla ei ole mitään ohjelmallista vaikutusta.


Käyttöliittymä
Metal removal process
START
STARTMAIN 1
IF STARTMAIN 1
Mixer1 ->1
Mixer2 ->1
Pump1 ->1
Tässä vaiheessa siis valitaan ohjelma, mikä halutaan ajaa käyttöliittymästä sekä laitetaan sakottimet ja syöttöpumppu1 päälle
STARMAIN1=metal recovery
STARTMAIN2=ammonia removal
Step1
pH check (input)
Ph adjust (dosing pump)
pH not ok
if
ok
Painesuodatin
WAIT1
PH:n tarkistus MODBUS
IF COM ERROR
TRUE
IF PH=0-1/ 10=ERROR PRINT ERROR
NOT OK Ventillin kääntö sekä PUMP2=1
FALSE
IF PHOK
FALSE
TRUE
OK Syöttöpumppu pysyy päällä ja hypätään seuraavaan stepplin
Step2
Tarkastetaan virtausmittarin lukema eli onko tulosignaali >4mA
FALSE
IF
TRUE
FLOW1<4 NO PROCESS Siirrytään seuraavaan stepplin
FLOW1=>4 PROCESS GOING
Aloitetaan backwash
V10 >1 nCounter 0 nCounter +1
IF nCounter= BACKWASH TIMER
TRUE
FALSE
Step2OK
if
vastavirtahuuhtelu
Process going
No process
Step3
PUMP3=1
Tarkastetaan kontin pinnankorkeus
IF LEVEL1=4 mA
TRUE
FALSE
LOADBATCH :=0
LOADBATCH KUITTAUS KÄYTTÖLIITTYMÄSTÄ
WAIT
WAIT
Odotetaan että kontti on tyhjä
IF LOADBATCH OK=1
FALSE
TRUE
nCounter=0 nCounter +1
WAIT
WAIT2
IF nCounter= WAIT2
FALSE
TRUE
Fill tank
Load batch
Wait 2 hours
if
Tank full (input)
filling
wait
Unfill tank
STEP4
Process ready
End program
PRINT PROGRESS READY
END PRGM

Kaavio 3. Ohjausjärjestelmän yleinen rakenne [6]

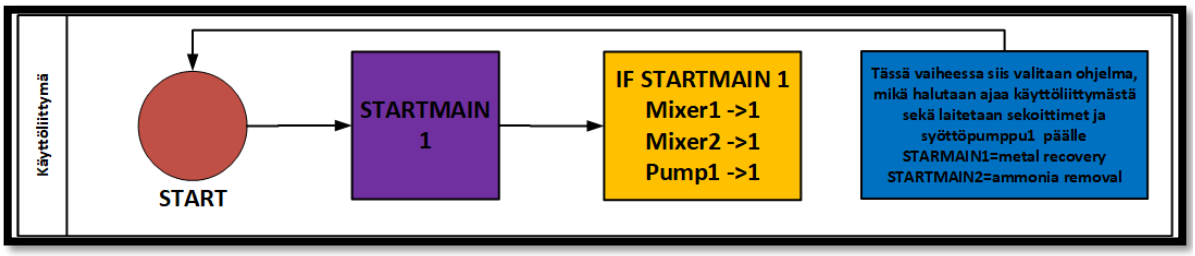
5.2 Step_INIT sekä Step0

```
E_STATE.STEPINIT:
//tässä stepissä valitaan ohjelma sekä piilotetaan program-ready popup-ikkuna
    //STARTMAIN1=pussisuotimen puolen ohjelma
    //STARTMAIN2=kolonnipuolen ohjelma, regeneraatiokolumni vasemmalla
    //STARTMAIN3=kolonnipuolen ohjelma, regeneraatiokolumni oikealla
    //STARTMAIN4=kolonnipuolen ohjelma, regeneraatiokolumni keskellä
    Running := FALSE;
    Error   := FALSE;
    GVL.ProgramReady :=1;
    IF GVL.STARTMAIN1=1 THEN
        Running := TRUE;
        nStep := E_STATE.Step0;
        Dummyvar1 :=1;
        ELSE
        Running := FALSE;
    END_IF
    IF GVL.STARTMAIN2=TRUE THEN
        Running :=TRUE;
        nStep := E_STATE.Step7;
        DummyVar3 :=1;  //Tämä on step 7:n  toinen käynnistysehto
        ELSE
        Running := FALSE;
    END_IF
    IF GVL.STARTMAIN3=TRUE THEN
        Running :=TRUE;
        nStep := E_STATE.Step9;
        DummyVar4 :=1; //Tämä on step 9:n toinen käynnistysehto
        ELSE
        Running := FALSE;
    END_IF
    IF GVL.STARTMAIN4=TRUE THEN
        Running :=TRUE;
        nStep :=E_STATE.Step8;
        DummyVar5 :=1; //Tämä on step 8:n toinen käynnistysehto
        ELSE
        Running := FALSE;
    END_IF
```

Kuva 3. Step_INITin rakenne [7]

Step1:stä ennen ohjelmassa on kaksi askelta, eli Step_INIT (Kuva 3) sekä Step0 (Kaavio 4). Step_INITin käynnistysehtona on joku STARTMAIN-napeista, jotka saadaan todeksi käyttöliittymästä. START-MAIN1 on pussisuotimen ohjelman aloitusnappi, STARTMAIN2, -3- ja -4 puolestaan ovat kolonnipuo-len aloitusnappeja. STARTMAIN-napeista kerron tarkemmin kohdassa 6.3. Vain yksi kolonnipuolen aliohjelma voi olla käynnissä kerrallaan putkiston asettamien rajoitusten vuoksi. Step0:aa tarvitaan pelkästään pussisuotimen aliohjelmassa, kolonnipuolen aliohjelmiin on sisällytetty Step0n komen-not. Puolestaan, kun joku kyseisistä aliohjelmista on suoritettu, ”palaa” ohjelma automaattisesti Step_INITiin odottamaan uuden ohjelman aloituskäskyä.

Step 0:ssa vuorostaan käynnistetään sekoittimet sekä osa pumpuista. Eli siis Step_INITissä valitaan ohjelma, ja sen jälkeen siirrytään Step0aan, jonka jälkeen alkaa Step1, Step7, Step8 tai Step9, riippuen käynnistysehdoista. Eli kuten Kuva 4sta näkee, Step1 on pussisuotimen ohjelman ensimmäinen askel, ja Step7, -8 ja -9 ovat kolonnipuolen sekvenssien askelia. ”GVL.PROGRAMREADY_A :=1” komennolla saadaan ProgramReady Pop-up-ikkuna piiloon ja ”Running := TRUE/FALSE” -komennolla vuorostaan saadaan indikoitua, onko prosessi käynnissä vai ei.



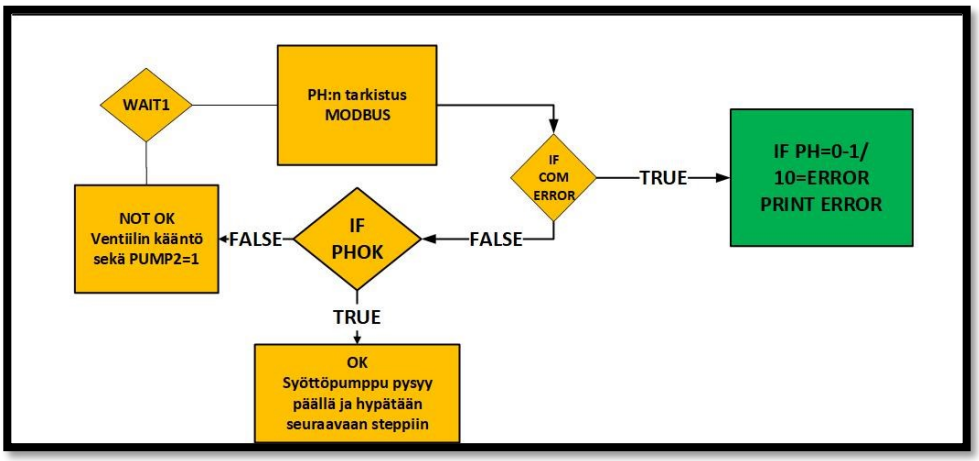
Kaavio 4. Käyttöliittymän rakenne [8]

5.3 Askel 1 eli Step1

Ennen tätä askelta (Kaavio 5) molemmissa IBC-konteissa olevat sekoittimet ovat jo käynnissä ja käyttöliittymän pussisuotimen Start-nappia on painettu eli itse pussisuotimen prosessi on jo käynnissä. Tämän ensimmäisen askeleen (Step1) tarkoituksena on tarkistaa prosessiin saapuvan jäteveden pH, eli onko se annettujen parametrien rajoissa, jotta esimerkiksi kuparia ei pääse saostumaan putkistoon tai IBC-kontteihin. Tämä sen takia, että mm. kuparia tällä kyseisellä prosessilla on tarkoitus ottaa talteen. Jos pH on ok, eli sen arvo on käyttöliittymässä määritettyjen arvojen välissä (kuva 4), tällöin pumput pysyvät päällä ja siirrytään seuraavaan askeleeseen. Vastaavasti jos pH ei ole ok käynnistetään syöttöpumppu, joka syöttää etikkahappoa veteen, joka laskee jäteveden pH:ta. Pumpun käynnistämisen jälkeen odotetaan Wai1:n ajan, kunnes pumppu pysäytetään, venttiili suljetaan sekä pH tarkistetaan uudestaan. Tämän ”loopin” jälkeen palataan pH:n tarkastukseen ja sen jälkeen hypätään seuraavaan askeleeseen.

Ohjelmassa on useita tällaisia odotuspiirejä, jotka ovat pääasiallisesti toteutettuja nCounter-nimisellä laskimella, joka lähtee laskemaan millisekunteja nollasta ylöspäin, ja kun se saavuttaa tietyn lukeman (joko kovakoodatun tai muuttujilla säädettävän arvon), IF-lause tulee todeksi ja siirrytään

seuraavaan toimintoon. Muuttamalla Wait1:n aikaa käyttöliittymästä, saadaan odotusaikaa muutettua helposti. Lähes kaikki ohjelman nCounterit on toteutettu niin, että itse nCounterin arvoa verrataan johonkin muuttujaan, jonka arvoa säädetään käyttöliittymästä. Tässä toimintatavassa on se etu, että näin prosessiaikoja voidaan säätää prosessin ollessa käynnissä. Jos arvo olisi kovakoodattu (eli nCounter := 600'00 tms.), jouduttaisiin ohjelma ja koko prosessi käyttämään alhaalla ennen kuin odotusaikoja voitaisiin muuttaa.



Kaavio 5. Step 1:n rakenne [9]

Kaaviossa 5, vihreässä laatikossa oleva teksti tarkoittaa sitä, että jos esim. pH-anturissa on jokin häiriö eli sen antama arvo on pienempi kuin yksi tai suurempi kuin kymmenen, niin silloin käyttöjärjestelmään tulee PopUp-ilmoitus kyseisen anturin häiriötilasta. Tämä pätee muihinkin antureihin, kuten virtausmittareihin sekä joihinkin logiikkakortteihin. Riippuen häiriön vakavuudesta koko prosessi voidaan tarvittaessa pysäyttää tai häiriön voi vain kuitata pois ja jatkaa prosessia. Error-PopUpeista kerron tarkemmin kohdassa 8.3.4. Prosessi voidaan tietysti pysäyttää hätäpysäytys -napilla tarvittaessa eli kaikki vaaraa tuottavat liikkeet pysähtyvät nappia painettaessa. Hätäpysäytyksestä kerrotaan tarkemmin kohdassa 5.7 TwinSAFE.

```
IF PH1 >=GVL.PhLOW AND PH1 <=GVL.PhHIGH THEN
PUMP1 :=1;
Step1OK :=TRUE;
END_IF
ELSE
PUMP2 :=1;
nCounter := nCOunter +1;
IF nCounter = GVL.WAIT1 THEN
    Step1OK := TRUE;
END_IF
```

Kuva 4. Step 1:n IF-lauseen rakenne [10]

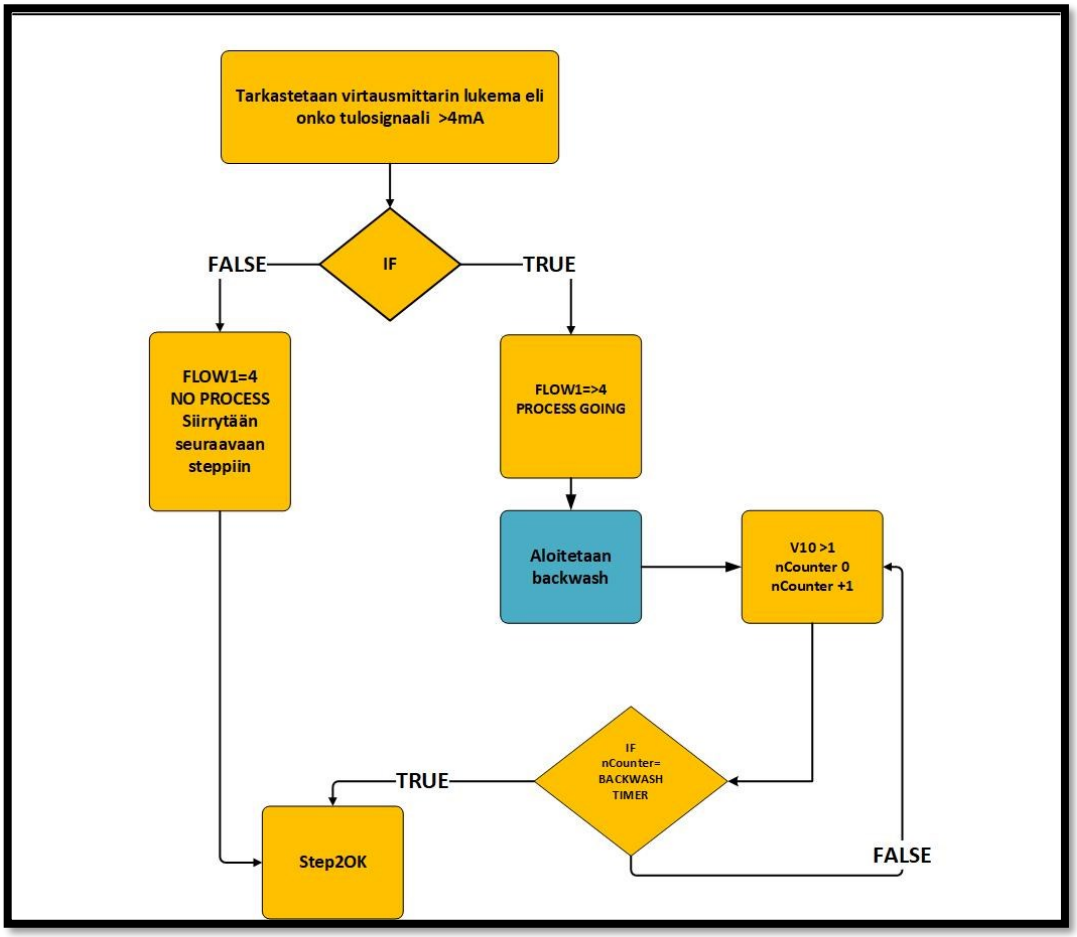
Kaaviossa 3 on useita eri countereita, vaikka käytännössähän kaikki ohjelman counterit ovat yksi ja sama nCounter, poislukien kolonnipuolen sekvenssi-ajastimet. nCounter:it nollataan ennekuin niitä uudelleen tarvitaan. nCountereilla ohjelmaan saadaan aikaan viiveitä sekä odotuspiirejä, joilla itse prosessia saadaan säädettyä ajallisesti, kun itse ohjelma ajaa itsensä läpi alle sekunnissa. Tässä ohjelmassa käytetään Rising Edge-muotoista counteria, eli se laskee 0:sta ylöspäin toisinkuin Falling Edge, joka laskee kohti nollaa. Kuten kuva 4:sta näkee sen seitsemännellä rivillä nCounter laitetaan laskemaan 0:sta ylöspäin. Tämän jälkeen asetetaan IF-lause, jolla saadaan määritettyä, kuinka pitkälle nCounterin annetaan laskea ennen kuin siirrytään seuraavaan funktioon. nCounterin arvon ollessa yhtäsuuri kuin GVL.WAIT1:n arvo, saa Step1OK arvon tosi. GVL.WAIT1 on käyttöliittymästä säädettävä muuttuja, jonka etuliite GVL tarkoittaa globaalia muuttujalistaa. Näistä säädettävistä muuttujista kerron enemmän kohdassa 6.3. nCountereita käytettäessä on myös otettava huomioon sekin, että ohjelman siirtyessä lukemaan seuraavaa riviä nCounter jatkaa laskemistaan ”hamaan” loppuun asti, eli se pitää nollata ennekuin sitä seuraavan kerran tarvitaan. Käytännössä tämä nCounter nollataan jokaisen askeleen viimeisellä rivillä ennen siirtymistä seuraavaan.

5.4 Askel 2 eli Step2

Askel 2:n pääasiallisena tehtävänä on tarkistaa, onko prosessi käynnissä vai ei. Kaavio 6 havainnollistaa tätä askelta. Jos prosessi ei ole käynnissä, voidaan aloittaa vastavirtahuuhtelu, jonka kemiasta kerroin jo aiemmin. Tässä kohdassa vastavirtahuuhtelun käynnistysehto on virtausmittarin lukema, jonka on 0:aa suurempi, jos prosessi on käynnissä, vaikka kaaviossa onkin numero 4. Tämä sen takia,

koska mittarin tulojännite on 4–24 mA. Tulojännitteen ollessa 0 voidaan olettaa, että itse anturi on vikaantunut, tai että johdotuksessa on jokin häiriö. Ohjausjännitteen ollessa 0 käyttöjärjestelmään ilmestyy ponnahdusikkuna, joka kertoo kyseisen virtausmittarin olevan häiriössä. vastavirtahuuhtelun alkamisen jälkeen odotetaan BackWashTimerin verran, jonka jälkeen vastavirtahuuhtelun oletetaan olevan valmis. Tätä aikaa voidaan säätää käyttöliittymästä, ja BackWashTimerin aikaa verrataan nCounterin arvoon, eli kun ne ovat yhtä suuret, askel kuitataan valmiiksi. Periaate tällä ajastimella on siis täysin sama kuin edellisen askeleenkin ajastimella.

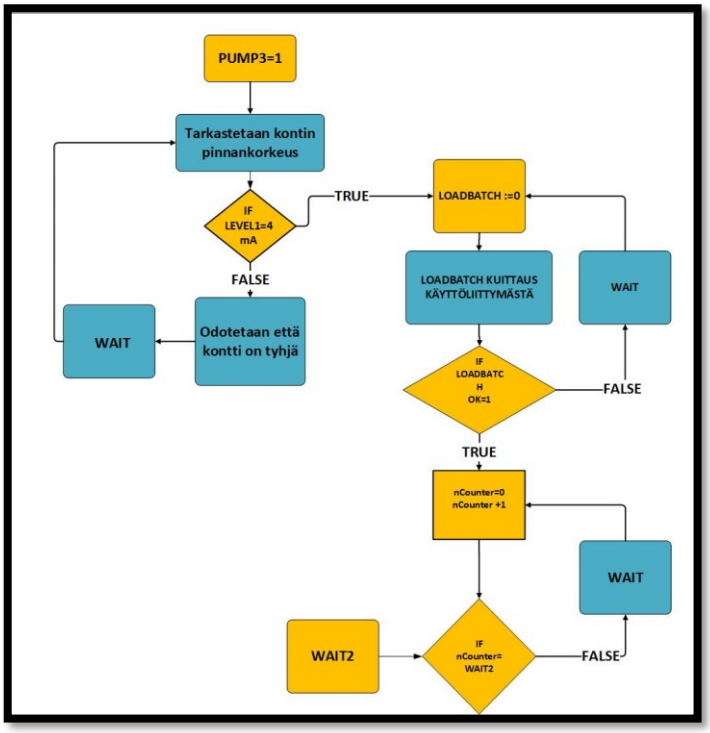
Tässä askeleessa ei tapahdu juurikaan muuta, vaan vastavirtahuuhtelun jälkeen siirrytään seuraavaan askeleeseen. Tämä askel on kuitenkin itse prosessin kannalta tärkeä pussisuodattimen tukkoon menemisen estämiseksi ja suodatuksen laadun ylläpitämiseksi. Askeleen lopussa on ”Step2OK-”palikka”, jollainen on jokaisessa muussakin askeleessa (toki tietenkin kyseessä olevan askeleen numerolla). Tämän funktion tarkoituksena on se, että tähän funktioon kytketään IF-lause, jolla tarkistetaan, että tarpeelliset tehtävät on suoritettu ennen seuraavaan askeleeseen siirtymistä. Step2OK:n saatua arvon tosi tilakoneen seuraava askel eli Step3:n tulee aktiiviseksi.



Kaavio 6. Step 2:n rakenne [11]

5.5 Askel 3 eli Step3

Tässä askeleessa tärkeimpänä komponenttina on LOADBATCH-vaatimus eli polymeeriin lisäys, jota syötetään käsin jäteveteen, jolloin se sitoo itseensä epäpuhtauksia. Vaatimus toteutetaan pop-up-ikkunalla, jonka vaatimuksena on PRINTLOADBATCH-muuttuja, jonka saadessa arvon 0 eli epätosi, pop-up-ikkuna muuttuu näkyväksi käyttöliittymäruudulla. Kyseisessä PopUp-ikkunassa on ok-nappi, jolla kuitataan polymeerin käsin syöttämisen jälkeen, että syöttäminen on tehty. Kuittauksen jälkeen LOADBATCHOK-vaatimus muuttuu TRUEksi, jolloin seuraava IF-lauseen ehdot täyttyvät. LOADBATCHOKn saadessa arvon tosi PopUp-ikkuna myös muuttuu takaisin näkymättömäksi.



Kaavio 7. Step 3:n rakenne [12]

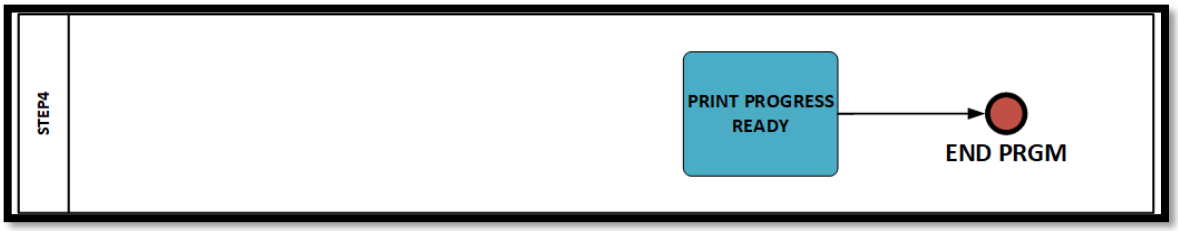
Jo edellisten askelten kohdalla esitetty IF-lause on tässäkin kohdassa tarpeen. Sen ideana on se, että ensin asetetaan ehdot, jotka tarvitaan ennen kuin IF-lause saa arvon tosi eli tosi (tässä askeleessa ehtona on, että virtausmittarin läpi ei kulje virtausta). Tämän jälkeen määritellään arvot ja muuttujat, jotka saavat arvon 1 sen jälkeen, kun IF-lause on tosi. Kun nämä arvot ovat tosia, siirrytään seuraavaan askeleeseen eli askel 4:een ”nStep := E_STATE.Step4” -komennolla. Kuvassa 5 on esimerkkinä askeleen 3 IF-lauseiden rakenne.

```
E_STATE.Step3:
    //Ohjataan pumppu päälle
    //Kun pumppu päällä vaaditaan polymeerin syötön kuittaus
    //Tämän jälkeen odotetaan Wait2:n ajan ja hypätään seuraavaan askeleeseen
    //Wait2 säädetään F08-visusta
    GVL.PUMP3 :=1;
    IF GVL.LEVEL1HIGH =1 THEN
    GVL.LOADBATCH :=0; //Tällä saadaan LOADBATCH-ikkuna näkyväksi
    nCounter := nCounter :=0;
    END_IF
    IF GVL.LOADBATCHOK=1 THEN
        nCounter := nCounter +1;
    END_IF
    IF nCounter = GVL.WAIT2 THEN
        Step3OK :=1;
    END_IF
    IF STEP3OK=1 THEN
        nStep := E_STATE.Step4;
        GVL.LOADBATCH :=1; //Tällä saadaan loadbatch-ikkuna piiloon
    END_IF
```

Kuva 5. Step 3:n IF-lauseiden rakenne [13]

5.6 Askel 4 eli Step4

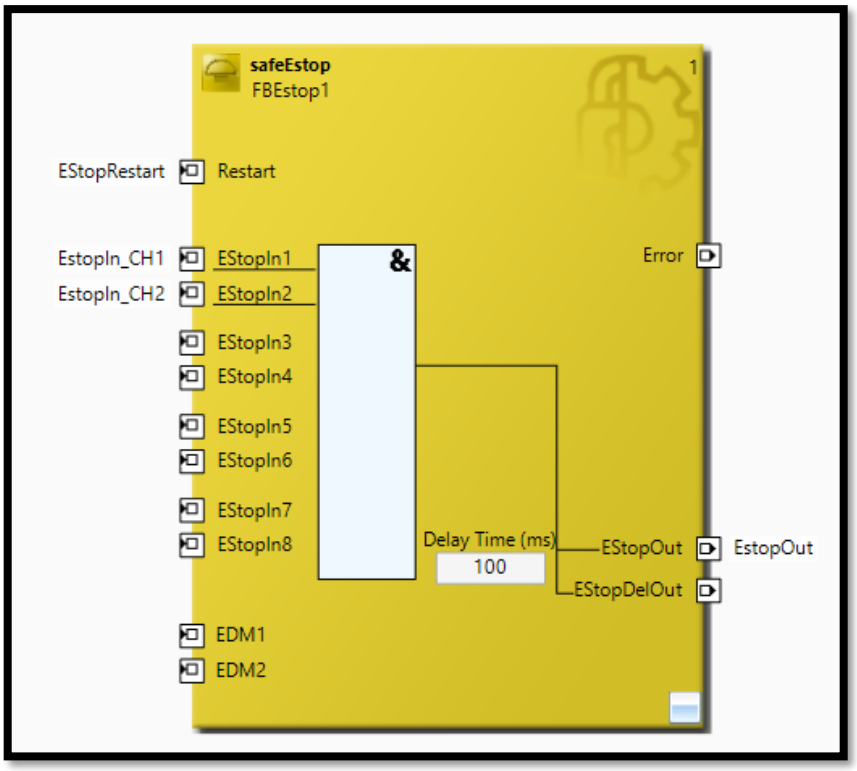
Tässä askeleessa (Kaavio 8) ei ole muita komentoja kuin vain pelkkä printtaus, eli käyttöliittymään ilmestyy pop-up-ikkuna, jossa lukee prosessin olevan valmis. Tämän jälkeen prosessi pysäytetään pussisuotimen osalta. Pop-up-ikkunasta voidaan valita jatkotoimenpiteet, eli jatketaanko saman ohjelman ajoa, vai käynnistetäänkö joku muu ohjelma. END PRGRM -pallon ”sisällä” on myös Askel 5, mutta siinä on pelkästään prosessin eli pumppujen jne. pysäytys sekä ohjelman resetointi alkuun.



Kaavio 8. Step 4:n rakenne [14]

5.7 TwinSAFE eli turvapuoli

Tämä askel pyörii koko ajan muun ohjelman rinnalla, ja sillä seurataan koko prosessin toimintaa turvallisuuden kannalta. Tämän turvapuolen tarkoituksena on siis valvoa ohjelman turvallisen suorittamisen edellytyksiä eli mahdollisien vikojen aiheutumista mm. kommunikaatiohäiriöiden ja muiden vastaavien varalta. TwinSAFEn osana on tässä yhteydessä myös fyysinen hätäpysäytys-nappi, jolla koko prosessin vaaraa tuottavat toiminnot pysäytetään. Turvapuoli valvoo myös mahdollisia logiikkakorttien vikaantumista sekä siitä johtuvia kommunikaatiohäiriöitä. Alapuolella on itse TwinSAFEn turvapuolen function block- eli toimilohkokaavio (Kaavio 9), johon on kytketty sekä fyysinen hätäpysäytys-nappi sekä myös hätäpysäytyksen kuittausnappi. Itse hätäpysäytys on ajon aikana arvona 0, ja kun nappi on vaikutettuna eli pohjassa, sille tulee arvo 1. Tämän jälkeen koko prosessi pysähtyy eli jokainen pumppu ja sekoitin pysähtyy. Tämän jälkeen tehdään mahdolliset korjaustoimenpiteet eli paikataan vuotava putkiliitos tms.



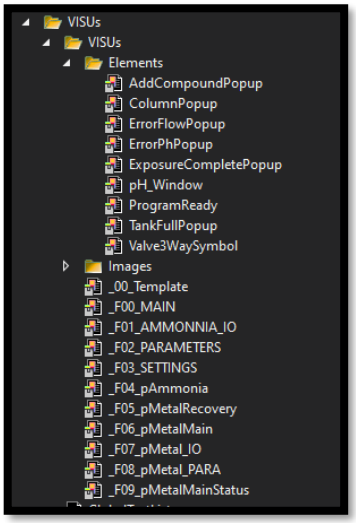
Kaavio 9. TwinSAFE rakenne [15]

Korjaustoimenpiteiden jälkeen hätäpysäytys kuitataan fyysisellä kuittausnapilla, jonka jälkeen ohjelma jatkaa Step INITistä eli ohjelman alusta. Tämä toimilohkohan ei varsinaisesti ohjelmaa resetoi, vaan kuittausnappi on kytketty koodinpätkään, joka resetoi ohjelman Step_INITiin. Hätäpysäytysnappi on fyysinen eikä ohjelmallinen sen takia, ettei se ohjelmistohäiriön takia kuittaannu esimerkiksi, kun käyttäjä on vaaravyöhykkeellä tekemässä korjaustoimenpiteitä. TwinSAFEn osana on myös EL9080-turvakortti (kuva 2), joka valvoo logiikkakortilta toiselle tapahtuvaa tietoliikennettä. Itse hätäpysäytysnappi sekä sen kuittausnappi vuorostaan ovat kytkettyinä EL2911-mallin turvakorttiin.

Toimilohkokaavion (kaavio 9) vasemmassa laidassa on useampi Inputkanava, joista EStopRestart-laatikkoon on kytketty hätäpysäytyksen kuittausnappi, sekä EStopIn_CH1een hätäpysäytys-nappi. Muut EStopIn kanavat ovat tyhjiä, koska kohteessa ei ole kuin yksi hätäpysäytys-nappi. EStopOut puolestaan kuvaa kuittaussignaalia, joka toimilohkosta lähtee sitten, kun hätäpysäytys on ollut aktiivinen ja se on kuitattu eli prosessi voi näin ollen jatkua TwinSAFE- toimilohkon puolesta normaalisti. Myös vasteaikaa kuittausnappiin vaikuttamisesta toimilohkon kuittaantumiseen voidaan säätää, kuten kaaviosta 8. näkyy. Vasteajanhan pitää tietysti olla mahdollisimman lyhyt, mutta käyttökohteen mukaan vasteajan pituudelle voi olla hyvinkin tarkat kriteerit.

6 Käyttöliittymän rakenne

Käyttöliittymä rakentuu useita eri ikkunoista, joista näkee prosessin tilan sekä joista voi valita tiettyjä ohjausparametrejä sekä suorittaa aliohjelmia. Aliohjelmilla tarkoitan puhdistuskolonnien eri sekvenssejä. Ohjausparametrien säädöstä kerron tarkemmin kohdassa 6.3.3. Aliohjelmien suorituksen käynnistysehdoista vuorostaan kerron tarkemmin kohdissa 6.3.1 ja 6.3.2. Itse käyttöliittymäikkunoita on 9 kappaletta (Kuva 7), joista esittelen tässä pari keskeisintä. Itse käyttöliittymä tai HMI on ohjausjärjestelmän graafinen osa, joka näkyy paneeli-PC:n näytöllä. Kuvassa 6 näkyy listaus käyttöliittymän ikkunoista sekä elementeistä. Elementit ovat objekteja, jotka tulevat käyttöliittymän päälle ja ne ovat oletuksena näkymättömiä, kunnes ne ohjelmallisesti kutsutaan näkyviksi. Elementeissä on siis aina ”käynnistysehtona” joku muuttuja, jolla se kutsutaan käyttöliittymän päälle näkyviin. Vastaavasti kun PopUp kuitataan pois ok-napista, saadaan kuitattu elementti piiloon.

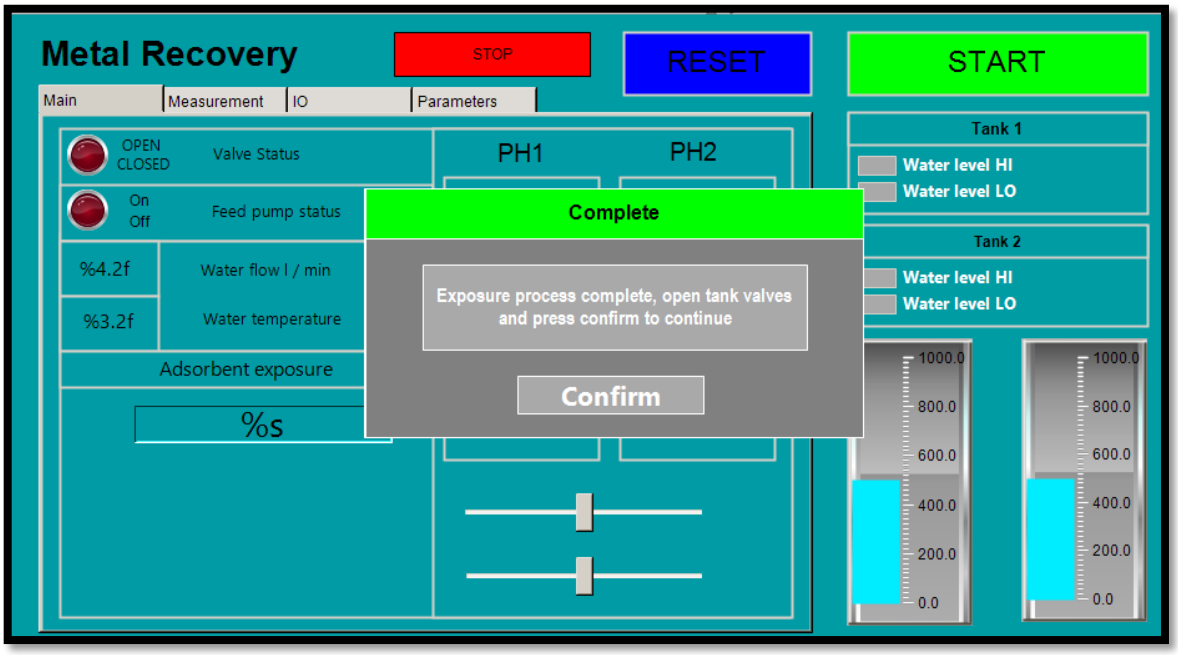


Kuva 6. Lista käyttöliittymän elementeistä [16]

6.1 Metallien talteenoton käyttöliittymä

Tästä näkymästä (kuva 7) näkee metallien talteenoton tilan eli pussisuotimen puolen ohjelman sekä prosessin tilan. Tälle näkymälle tulee näkyviin mm. LOADBATCH-PopUp-ikkuna sekä ProgramReady-PopUp-ikkuna. PopUp-ikkunoista sekä niiden piilotuksen konfiguroinnista kerron tarkemmin kohdassa 6.3.4 ErrorFlowPopUp sekä ErrorPhPopUp. Tästä ikkunasta (kuva 8) näkee myös syöttöpumpun tilan sekä mm. pinnankorkeudet IBC-konteissa. Tässä ikkunassa ei ole muita toiminnallisia nappeja kuin START-, Stop- sekä Reset-nappi. Muut elementit ovat vain indikoivia eli ne kertovat prosessin tilasta käyttäjälle. Kyseisen ruudun syöttöpumpun (Feedpump) ja venttiilin lamppuja on kaksi kappaletta eli vihreä ja punainen, jotka muuttujan tilan mukaan joko palavat tai ovat piilotettuna. Eli kuten kuvasta näkyy, kun syöttöpumppu ei ole päällä, vihreä indikaattorivalo on piilotettuna, ja punainen valo on näkyvä. Pumpun ollessa päällä vihreä valo palaa, ja punainen valo on piilotettuna. %4.2f lamppujen alla olevassa laatikossa indikoi sitä, montako desimaalia näytetään virtausmittarin arvolle, ja %s vuorostaan kuvastaa string-muuttujaa. String-muuttujia ohjelmassa ei ole kovinkaan montaa, mutta se eroaa boolean-muuttujista ja integreistä muuttujista siten, että string-muuttujalle on annettu merkitsevien merkkien pituusehto. Eli siis 4.2 kertoo sen, että ennen desimaalipilkkua on 4 numeroa ja desimaalipilkun jälkeen 2 numeroa, eli arvo on välillä 0000.0 -> 9999.99. Tällä ruudulla

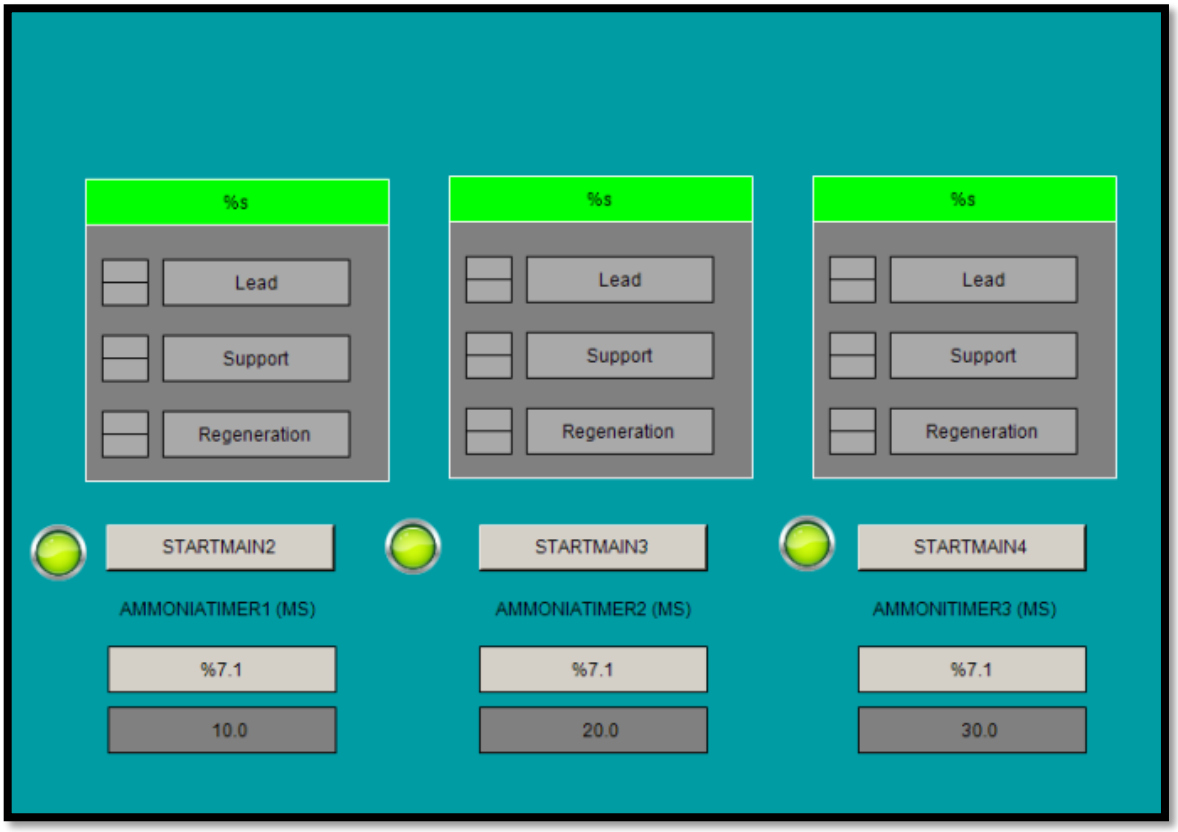
on myös navigointipalkki, eli siitä säädetään, mikä ikkuna näytöllä kulloinkin näkyy, ja josta navigoidaan eri käyttöliittymäikkunoiden välillä.



Kuva 7. Metallien talteenoton käyttöliittymä [17]

6.2 Ammoniapuolen käyttöliittymä

Ammonia- eli kolonnipuolen käyttöliittymäruutu näkyy kuvassa 8. Käyttöliittymästä näkee sekä prosessin tilan että pystyy säätämään parametrejä ja käynnistämään jonkin aliohjelmista eli sekvenseistä. Ylimmät objektit indikoivat sitä, onko kyseinen kolonni Lead-, Support- vaiko Regeneration-tilassa. STARTMAIN-napeista vuorostaan saadaan yksi kolmesta eri kolonniohjelmasta käyntiin. AMMONIATIMER-laatikoista saadaan säädettyä prosessiaikaa eli kuinka pitkään sekvenssi kestää. Alemmat tummanharmaat laatikot ovat pelkkiä indikaattoreita eli ne kertovat, kuinka pitkä prosessiaika on syötetty kuhunkin AMMONIATIMERiin. Vihreät valot vuorostaan kuvastavat sitä, mikä kolonni on milloinkin regeneraatiotilassa. Kun kolumni on regeneraatiotilassa, sen läpi ei ajeta jätevettä, vaan sitä huuhdellaan suolaliuoksella.



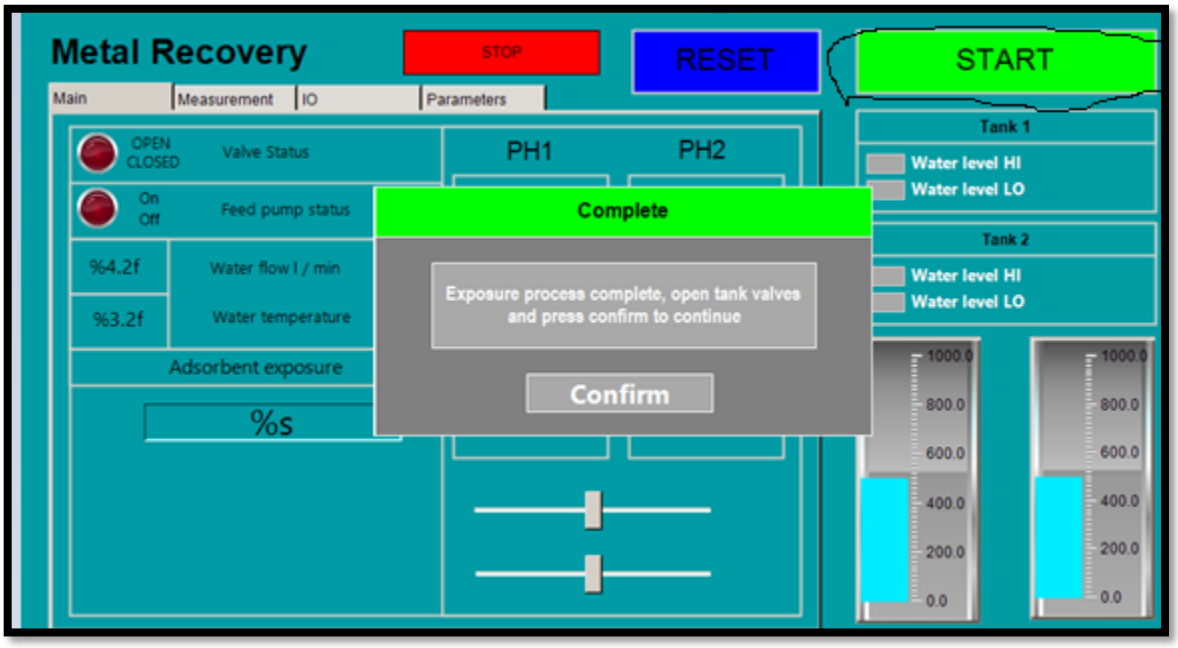
Kuva 8. Ammonia-puolen käyttöliittymä [18]

6.3 Käyttöliittymän liittäminen itse ohjelmaan

6.3.1 Startmain1

Kuten jo aikaisemmin mainitaan, STARTMAIN1 (Kuvassa 9 ympyröitynä) on pussisuotimen puolen ohjelman käynnistysehto tai oikeammin käynnistysnappi. Kuvassa 10 näkyy metallien talteenoton eli pussisuotimen puolen käyttöliittymä, jonka START-nappi on yhdistetty STARTMAIN1-muuttujaan. Muuttujia voi yhdistää käyttöliittymään monilla eri tavoilla, riippuen muuttujan muodosta eli onko se Boolean-muuttuja tai Integreri-muuttuja, vaiko joku muu muuttuja. Tässä on käytetty Toggle-nappia, koska muuttuja on boolean-muuttuja, joka on joko tosi tai epätosi. Eli kun nappiin vaikutetaan, muuttuja saa arvon TRUE eli tosi. Vastaavasti kun nappiin vaikutetaan uudestaan, muuttuja saa arvon

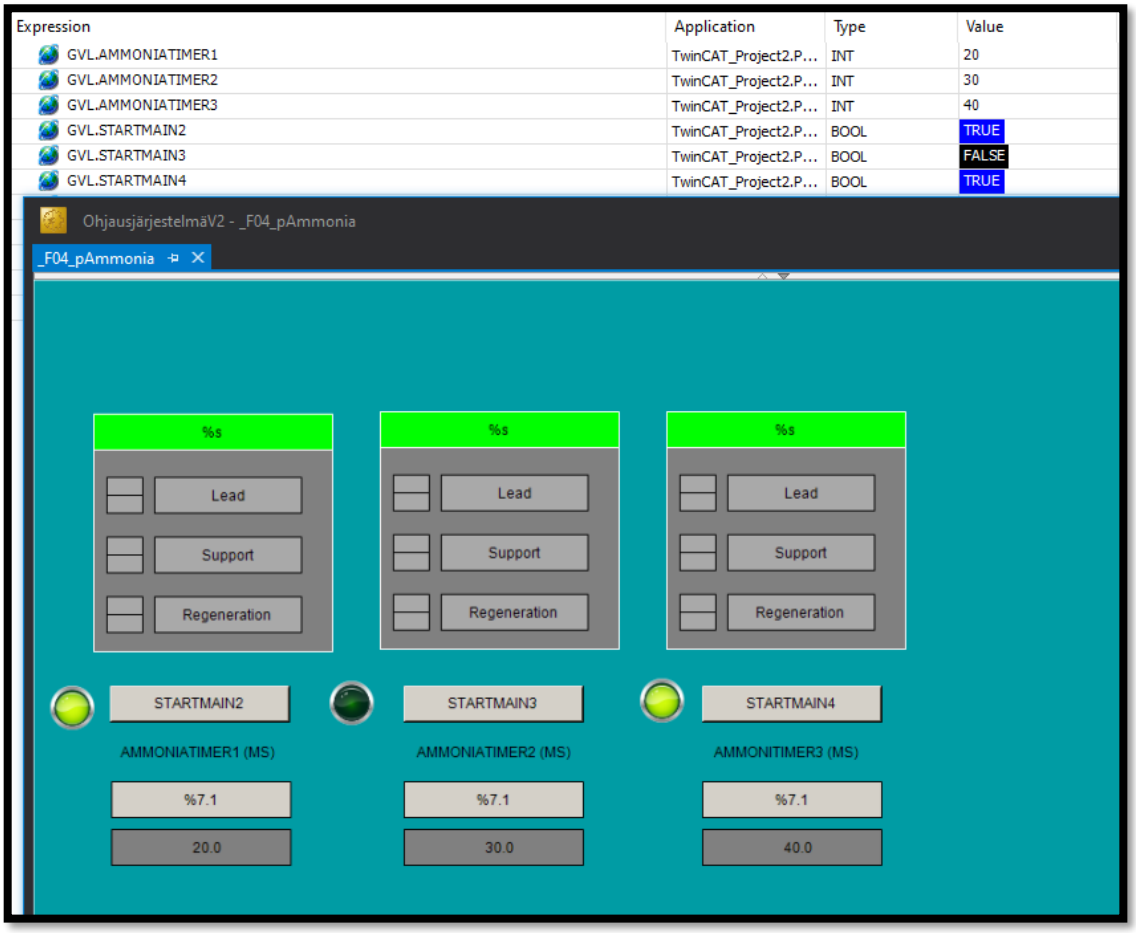
FALSE eli epätosi. Tässä ohjelmassa itse nappia ei resetoida, vaan STARTMAIN1-muuttuja resetoidaan ohjelmallisesti, mikä vuorostaan resetoi käyttöliittymän Start-napin.



Kuva 9. STARTMAIN1-muuttuja käyttöliittymässä [19]

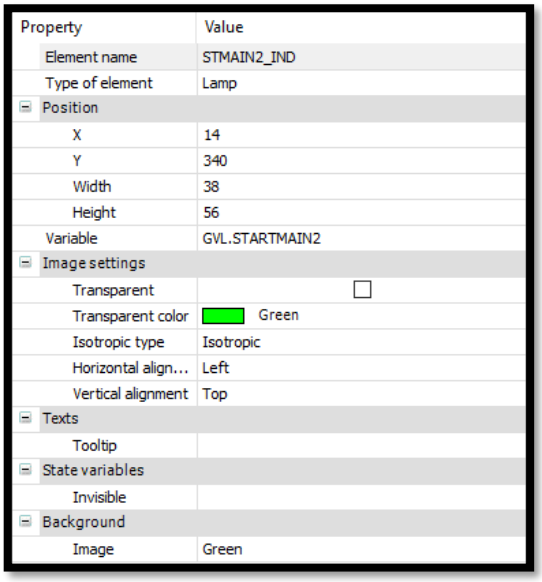
6.3.2 Startmain2, -3 ja -4 sekä Ammoniatimer1, -2 ja -3

Kuvassa 10 on havainnollistava esimerkki itse muuttujien kytkennästä kyseisiin nappeihin. STARTMAIN2, STARTMAIN3, sekä STARTMAIN4 ovat Toggle-nappeja, kuten STARTMAIN1. Mustalla näkyvät AMMONIATIMER (MS) ovat pelkkiä tageja, joilla ei ole mitään muuta tarkoitusta kuin helpottaa käyttöliittymän käyttöä. Vastaavasti vaaleamman harmaat laatikot, joissa lukee %7.1 ovat laatikoita, joista GVL.AMMONIATIMER-muuttujille saadaan kirjoitettua jokin arvo, jolla säädetään sekvenssiaikaa. AMMONIATIMER-muuttujat ovat aikamuuttujia, ja niillä säädetään kunkin sekvenssin ajastimen aikaa. Eli ajastintoimilohkon PT-aikaa (time to pass) säädetään AMMONIATIMERillä ja ET-ajan (Elapsed time) saavutettua AMMONIATIMERin arvon laskuri nollaantuu ja sekvenssi on valmis. %7.1 kuvassa 10 tarkoittaa sitä, että laatikkoon syötetään reaaliluku, jossa voi olla 7 lukua ennen desimaalipistettä sekä yksi luku desimaalipisteen jälkeen. Eli luku on välillä 0–9999999.9 millisekuntia eli 0–2.8 tuntia.



Kuva 10. STARTMAIN-muuttujien liittäminen käyttöliittymään [20]

Muuttujan kirjoituskonfigurointia avataan tarkemmin kohdassa 6.3.3. Tummemman harmaat laatikot kuvassa 10 ovat pelkkiä indikaattoreita siitä, mikä arvo AMMONIATIMERille on syötetty. Ylimpänä oleva lista on lista muuttujista, joka havainnollistaa muuttujien kirjoittamista sekä käyttöliittymänappien vaikutusta itse ohjelmaan. Eli, kun STARTMAIN2 on totena, merkkivalo palaa vaaleamman vihreänä, ja täten kyseisen sekvenssin käynnistysehto on tosi, eli sekvenssi voi alkaa. Vastaavasti AMMONIATIMER-muuttujat vaikuttavat itse sekvenssin kestoon samalla tapaa kuin jo edellisissä kohdissa mainitut BACKWASHTIMER ja WAIT1. Vaikka kuvassa 10 on kaksi käynnistysehtoa totena, näin ei kuitenkaan voi prosessin ollessa käynnissä tapahtua, koska yhtä kolonneista huuhdellaan jokaisessa sekvenssissä, ja se ei voi olla samaan aikaan huuhdeltavana sekä käytössä.



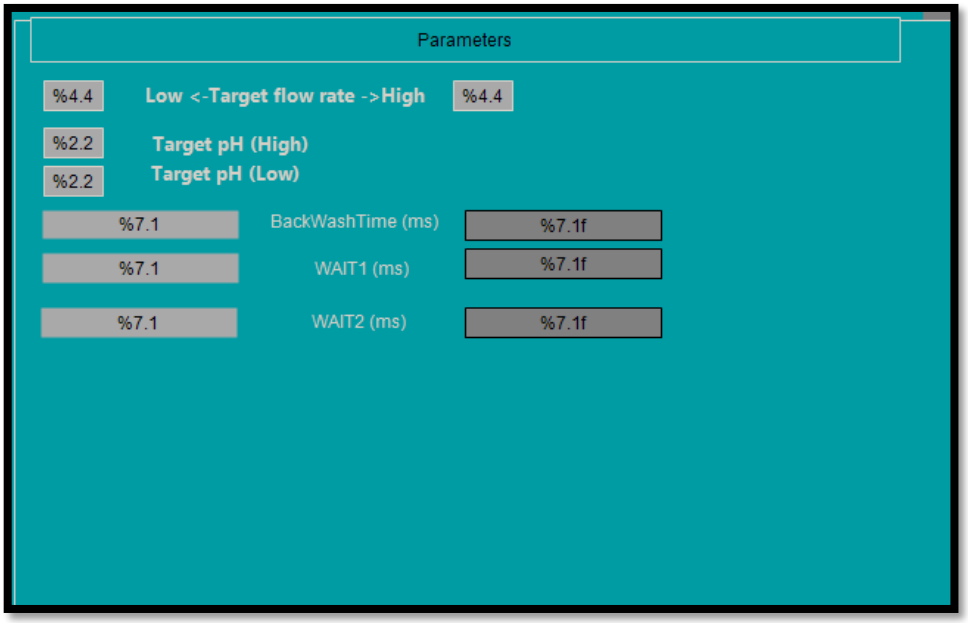
Kuva 11. Indikaattorilampun konfigurointi [21]

Kuvassa 11 näkyy tarkemmin lampun konfigurointi. Lamppuja on kuvan 10 käyttöliittymäruudussa 3 kappaletta, joista jokainen on yhdistetty luonnollisesti omaan muuttujaansa. Kuten kuvasta 11 näkee, kun muuttujan arvo on TRUE eli tosi, lamppu palaa vaaleana ja muulloin se on tummempi. Samaa lamppujen konfigurointia on käytetty myös muissa käyttöliittymän osissa, mutta periaate on näissäkin täysin sama, kuten kuvassa 11, riippuen toki muuttujasta, jota lampulla halutaan indikoida. Joissain kohdissa on käytetty myös State Variablea, eli muuttujan arvon mukaan objekti saadaan näkymättömäksi tai näkyväksi. Samaa State Variablea käytetään myös PopUp-ikkunoissa, eli ne pysyvät piilossa tai siis oikeammin näkymättöminä, kunnes tietty muuttuja saa arvon 0 eli epätosi.

6.3.3 F08_pMetal_PARA sekä muuttujien kirjoitus NumPadilla

Tässä osiossa kerron tarkemmin BackWashTimerin käytöstä sekä yleisesti muuttujien kirjoittamisesta sekä kyseisten lukujen syötön konfiguroinnista itse käyttöjärjestelmän objekteihin. Vastavirtahuuhtelun tarkemmasta vaikutuksesta on kerrottu kohdissa 5.1.2 ja7.4, eli tätä kyseistä BackWashTimerin arvoa verrataan askeleen nCounterin arvoon. Kuva 13 on otettu käyttöliittymän pMetal_PARA-välilehdeltä, jolta voidaan määrittää muitakin parametrejä kuten pH:n ja virtauksen target-arvoja. Kuvassa 12 vaalean harmaalla olevat tekstit kertovat, mitä laatikkoon pitää syöttää millisekunteina ja mihin muuttujaan kyseinen objekti on kytketty. Maksimiarvot ovat liian suuria, mutta

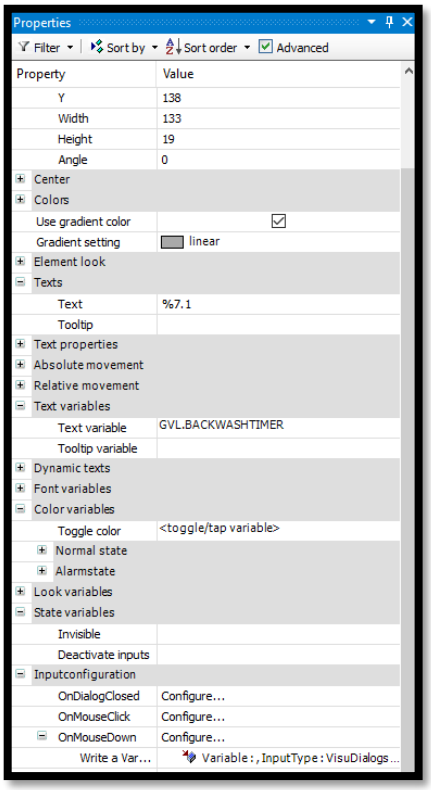
näin vältytään ns. turhilta ohjelman pysäytyksiltä, koska itse syötettävän arvon maksimia ei voi ohjelman käynnissä olon aikana muuttaa. Itse arvo muuttujalle voidaan toki kirjoittaa prosessin ollessa käynnissä, jollei muuttuja ole juuri sillä hetkellä käytössä. Target pH:n arvoja tarvitaan askeleessa 1 sekä virtauksen Target-arvoja askeleessa 2. Wait1- ja Wait2-arvot puolestaan ovat tarpeen askelissa 2 ja 3.



Kuva 12. pMetal_PARA-välilehden rakenne [22]

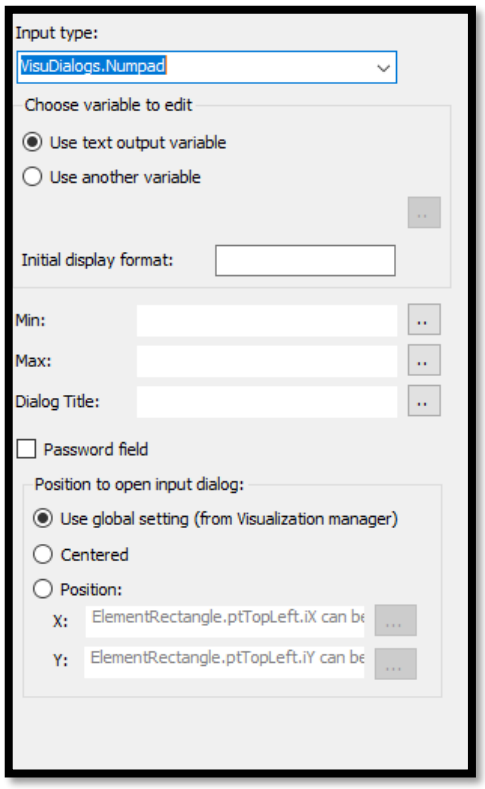
Kuvassa 13 on BackWashTimerin Properties-ikkuna, josta saadaan konfiguroitua laatikon parametrejä. Tämä siis tarkoittaa sitä, että tästä ikkunasta määritetään, mikä muuttuja yhdistetään kyseiseen objektiin ja millä tavalla. Koska GVL.BACKWASHTIMER on integreri-muuttuja, valitaan tulokonfiguraatioksi tässä OnMouseDown. Tämä tarkoittaa sitä, että hiiren vasenta painiketta painetaan kyseisen objektin päällä. Tähän ”tapahtumaan” voitaisiin toki lisätä muitakin toimintoja, kuten ennalta määritetyn Structure Textin ajamista, mutta tässä tapauksessa valitaan ”Write a Variable” eli kirjoitetaan muuttujalle lukuarvo. Säätämällä laatikon Text Inputia voitaisiin muuttujalle antaa muukin arvo kuin lukuarvo kuten vaikka kirjallinen teksti, mutta tässä tapauksessa lukuarvo on oikea vaihtoehto. Tekstiarvo voi olla oikea vaihtoehto, jos muuttujalla säädetään esimerkiksi PopUp-ikkunan viestiä tms. Tämä luku on maksimiarvo, joka määritellään text-osiossa, kuten aikaisemmin mainitsin. Jos muuttuja vuorostaan olisi boolean-muuttuja (tosiarvomuuttuja) eli tosi tai epätosi, voitaisiin käyttää

OnMOuseClickiä ja Toggle a Variable-konfigurointia eli nappia painamalla muuttuja saa arvon tosi ja vastaavasti, kun nappiin vaikutetaan uudestaan, saa se arvon epätosi.



Kuva 13.

BackWashTimerin Properties-ikkuna [23]

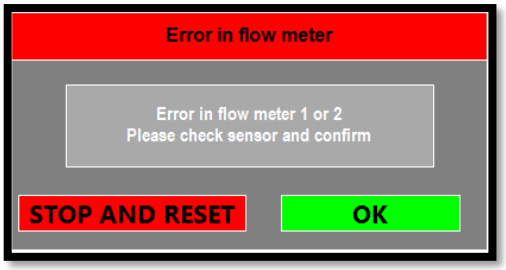


Kuva 14.

Muuttujan input konfigurointi [24]

Kuvassa 14 vuorostaan valitaan Input type eli se, syötetäänkö muuttujalle arvo tekstimuodossa vai tuleeko käyttöliittymään näppäimistö tai NumPad-PopUp-ikkuna. Tässä tapauksessa valitaan NumPad, koska käyttöliittymä on kosketusnäytöllä eli erillistä näppäimistöä ei ole. Muuttujaksi valitaan Text-OutPut-variable, eli kuvassa 13 valittu GVL.BACKWASHTIMER. Kuvan 14 Properties-ruudusta voitaisiin valita myös esimerkiksi laatikon värin muutos muuttujan arvon suhteen sekä paljon muitakin asetuksia, mutta tässä tapauksessa edellä mainitut konfiguroinnit riittävät.

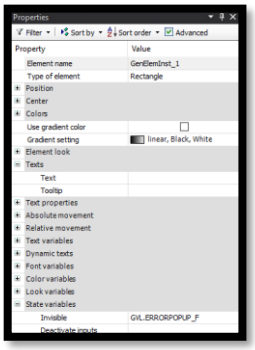
6.3.4 ErrorFlowPopUp, ErrorPhPopUp ja ErrorReset



Kuva 15. ErrorFlow-ikkuna [25]

Nämä PopUpit ovat anturihäiriötä indikoivia ikkunoita, jotka tulevat käyttöjärjestelmän päälle. Näiden kahden ikkunan toimintaperiaate on käytännössä täysin sama, joten esittelen tarkemmin tässä kohdassa pelkästään ErrorFlowPopUpin. Kuvassa 15 näkyy ErrorFlowPopUp elementin ulkoasu. Kyseisessä elementissä vain ”STOP AND RESET”- sekä ”OK”-nappeihin on liitetty jokin muuttuja. OK-nappiin on liitetty GVL.ERRORAKNOWLEDGEMENT-muuttuja, joka ei resetoi ohjelmaa, vaan jättää vikatilan päälle ja antaa prosessin jatkua. GVL.ErrorReset on muuttuja, joka on yhdistetty ”STOP AND RESET”-nappiin, ja johon vaikutettaessa prosessi pysähtyy ja palaa Step_INITiin. Itse PopUp-ikkunan näkymättömyyskonfigurointi on tehty kuvan 16 osoittamalla tavalla.

Eli virtausmittarin vikatilan tullessa päälle, GVL.ERROROPOPUP_F muuttuu FALSEksi, jolloin itse ErrorFlowPopUp-ikkuna tulee näkyväksi. Vastaavasti GVL.ERRORPOPUP_P-muuttujalla säädetään ErrorPhPopUp-ikkunan näkömättömyyttä. Tätä samanlaista konfigurointia käytetään myös muissa käyttöliittymän PopUp-ikkunoissa, erona toki muuttujan nimi.



Kuva 16. PopUp-ikkunan näkymättömyyskonfigurointi [26]

7 Ohjausjärjestelmän integrointi

Tässä osiossa kerron hieman ohjausjärjestelmän integroinnista itse prosessiin, eli miten 1:t ja 0:t ohjelmassa vaikuttavat itse prosessiin. Kerron myös erilaisista signaaleista sekä tiedonsiirrosta kenttälaitteilta logiikalle sekä logiikalta näyttöpäätteelle, koska näissä on kuitenkin hyvinkin suuria eroavaisuuksia. Logiikalla tarkoitetaan PLCtä, johon on lisätty lisäkortteja, joihin prosessin kenttälaitteet on kytketty. Käyttöliittymällä vuorostaan tarkoitetaan paneeli-PC:n näytöllä olevaa näkymää, jolla helpotetaan prosessin seurantaa.

7.1 Yleistä ohjausjärjestelmän integroinnista

Tässä jaossa pätee hierarkia, jossa PLCltä voidaan säätää kenttälaitteita, eli se toimii Master-komponenttina. Kenttälaitteet taas ovat Slave-komponentteja. Eli Master-laite kertoo Slave-komponenteille mitä tehdä, kun taas Slave-komponentit suorittavat halutut tehtävät sekä viestivät tilastaan Master-komponentille.

7.2 Automaatiojärjestelmän tietoliikenteen vaikutus ohjausjärjestelmän rakenteeseen

Tietoliikenne kenttälaitteelta logiikalle on pääasiallisesti analogista 4–24 milliampeerin virtaa, vaikkakin esim. pH-anturilta on suora ModBus-yhteys (tarkemmin ModBus RTU eli Ethernet-yhteys, jonka kautta ModBus RTU-tiedonsiirtoprotokolla kulkee) itse logiikkaan. Signaali logiikalta paneeli-PC:lle kulkee yleensä Ethernet-kaapelia pitkin, kuten tässäkin tapauksessa (sama tiedonsiirtoprotokolla kuin pH-anturilta logiikalle). Tämä tiedonsiirto voi olla langatontakin (yleensä WLAN tai radioaallot) esimerkiksi pitkän välimatkan takia. Tämä ModBus RTU-yhteys eroaa analogisesta signaalista siten, että se on digitaalista signaalia eli välillä 0–1, kun taas analoginen signaali on väliltä 4–20 mA ja kulkee yleensä kierrettyä kuparijohtoja pitkin. Näistä siis esimerkkinä pH-anturin digitaalinen Ethernet-signaali vs. venttiilin asennosta kertova analoginen signaali.

Yleisesti analoginen signaali on hankalampi lukea ohjelmallisesti koska ennen signaalin lukua on suoritettava niin sanottu A/D-muunnos (analogisesta digitaaliseen), eli 4–24 mA:n välillä oleva arvo on muutettava ymmärrettävämpään muotoon, kun luetaan esim. pinnankorkeusanturin lukemaa.

8 Riskien arviointi sekä vikaantuminen

Tässä osiossa kerron yleisellä tasolla riskien arvioinnista sekä mahdollisista vikatilanteista, sekä niitä vastaan suojautumisesta ohjelmallisesti. Riskien arviointia en käsittele tässä opinnäytetyössä kovin syvällisesti riskien laadun vuoksi. Riskin arvioinnin ensimmäinen vaihe on riskianalyysi, johon kuuluu järjestelmän ja sen käytön määrittely, vaarojen tunnistaminen sekä vaaroihin liittyvien riskien laskennallinen määrittäminen. Toisena riskin arvioinnin vaiheena on riskien toteaminen tarkastamalla, onko toivottu turvallisuustaso saavutettu. [3, s. 281]

8.1 Riskien arviointi

Riskien arviointihan alkaa mahdollisten riskien kartoituksella sekä jäännösriskien tutkimisella. Itse riskien kartoituksen alkuvaiheessa lasketaan MTBF, joka kertoo ajan laitteen vikaantumiseen sen edellisestä alkuperäiseen kuntoon saattamisesta. Tämän lasketaan jokaiselle kohteen komponentille, jonka jälkeen arvioidaan, onko riski tarpeeksi tiheään sattuva ja vahingon sattuessa tarpeeksi vaarallinen eli kriittinen, jotta sitä pitäisi tutkia tarkemmin. Kriittisyydelle on useita eri mittareita, kuten tuotannonmenetyksen laajuus, henkilövahingot tai suuret laitteistolle aiheutuvat vauriot ja tietenkin välillisesti myös rahalliset tappiot. Koska kyseessä on ”pelkkä” pilottiprosessi, siihen ei täten liity sen yksinkertaisuuden sekä koon vuoksi kovinkaan suuria riskejä. Tämän takia laajaa riskianalyysiä ei ole tarpeen tehdä, koska riskianalyysi on pitkä ja monimutkainen prosessi.

Erilaisia riskejä ovat esim. henkilövahingot, ympäristövahingot sekä itse laitteiston itselleen aiheuttamat vauriot. Ympäristövahingon riskejä tulee käsitellä kaivosteollisuudessa yleisesti enemmän ja tarkemmin kuin tässä projektissa, koska volyymi vedenpuhdistuksessa on huomattavasti suurempi sekä nesteet ovat huomattavasti haitallisempia ympäristölle kuin tässä kyseisessä kohteessa. Tämä johtuu muun muassa alhaisesta pH:sta sekä ympäristölle enemmän haitallisista aineista kuten. rikistä, uraanista sekä raskasmetalleista. Henkilövahingot sekä laitteiston itselleen aiheuttamat vauriot ovat todennäköisempiä silloin, kun ohjausjärjestelmä ohjaa raskaita tai nopeasti liikkuvia laitteistoja tai käytössä ovat hyvin suuret paineet tai muuta vastaavaa.

8.2 Vikaantuminen sekä sen eri muodot

Tässä osiossa kerron tarkemmin prosessin yleisimmistä vikatilanteista sekä siitä, miten niitä vastaan suojaudutaan ohjausjärjestelmätasolla. Vikatilanteen laadun mukaan koko prosessi voidaan ajaa alas tai se voi jatkua normaalisti, jos vika ei ole tarpeeksi kriittinen. Mahdollisia vikoja ei ole tässä prosessissa kovin montaa erilaista tyyppiä prosessin yksinkertaisuuden vuoksi. Näistä vikatilanteista muutamia esimerkkejä ovat muun muassa jännitepiikit, kenttälaitteiston vikaantuminen (eli anturi, venttiili, tai pumppu/sekoitin) tai itse ohjausjärjestelmää käyttävän koneen käyttöjärjestelmän kaatuminen. Mahdollisia ulkoisia vikatilanteita ovat myös. sähkökatkot, salamaniskut sekä muut ulkoiset vaarat, kuten fyysiset iskut jne., vaikka nämä vaarat/vikatilanteet nyt ovatkin melko epätodennäköisiä. Tämä tietenkin sen takia, että prosessi on erotettu ympäristöstään merikontin sisään. Seuraavassa listauksessa kerron muutaman esimerkin siitä, miten eri vikaantumisen muotoja sekä riskejä on ohjelmallisesti pyritty vähentämään.

Sekä riskit että vikaantumiset ovat tässä yhteydessä samassa listassa, koska vikaantumisesta syntyy normaalia korkeampi riski vahingosta itse laitteistolle tai käyttäjälle.

1. Jännitepiikkejä vastaan on suojauduttu fyysisillä vikavirtasuojilla sekä sulakkeilla.

2. Kenttälaitteen kriittisen vikaantumisen tapahtuessa ohjelma pysähtyy siihen kohtaan, jolloin vikaantuminen on havaittu, ettei esim. jätevettä pääse väärään paikkaan kuten maahan.

3. Ulkoisia riskejä vastaan ohjelmallisesti ei varsinaisesti voi suojautua, vaan näitä riskejä vastaan suojaudutaan itse fyysisen laitteistorakenteen avulla esimerkiksi suojamaadoituksilla ja koteloinnilla. Ulkoisilla riskeillä tarkoitan tässä kohdassa ulkoisia fyysisiä iskuja, salamaniskuja, maanjäristyksiä jne.

4. Ohjelmallisten häiriöiden aiheutumista vastaan on suojauduttu esim. fyysisellä hätäpysäytys-napilla, joka pysäyttää koko prosessin muutamassa millisekunnissa. Kaikkia ohjelmallisia häiriöitä ei voi tietenkään vielä suunnitteluvaiheessa tunnistaa, joten prosessin valvonnan tärkeys korostuu tässäkin aspektissa.

5. Odottamaton ohjelman käynnistyminen eli ohjelma käynnistyy odottamattomasti esim. virtapiikin, muun sähköisen vian takia tai ristiriitaisen hätäpysäytyskäskyn takia. Tätä vastaan on ohjelmallisesti vaikea suojautua, mutta tämän takia kohteessa on jo mainittu hätäpysäytys -nappi.

6. Itse prosessista aiheutuva haitta kuten esimerkiksi liian alhaisesta pHsta aiheutuva putkiston syöpyminen. Vaikkei tämä riski kovin suuri olekaan, ollaan sitä vastaan suojauduttu mm. pH-anturoinnilla ja pH:n säätölaitteistolla, joiden vikaantuessa koko prosessi pysähtyy.

7. Jonkin osan sinkoutuminen tai muu vastaava mekaaninen riski. Tätä vastaan on suojauduttu esim. anturoimalla sekoittimet tärinäantureilla.

9 Yhteenveto

Kaivosveden puhdistuskontti on KAMKn osa WaterPRO-hanketta, jonka alaisena opinnäytetyöni myös oli. Puhdistusprosessi on merikontin sisällä. Opinnäytetyöni aiheena on ohjausjärjestelmä, jonka ohjelmointi tehtiin TwinCat-ohjelmalla, ohjelmointikielellä nimeltä Structure Text. Itse logiikkaohjelman perimmäisenä tarkoituksena on ajaa prosessia halutulla tavalla täysin automaattisesti.

Itse ohjelmointi oli melko haastavaa, koska tämänlaista ohjelmointia ei oikein opeteta oppilaitoksessani tai koulutusalallani. Minulla ei myöskään ollut käytännössä minkäänlaista ohjelmointitaustaa, joten tämä loi oman haasteensa itse opinnäytetyöprosessin ohjelmointiosioon. Myös projektin minusta riippumattomat asianhaarat toivat oman haasteensa opinnäytetyöprosessiin. Opinnäytetyöprosessin aikana opin paljonkin ohjelmoinnista, sekä muun muassa oman tekemisen aikataulutuksesta.

Opinnäytetyöprosessin alkuvaiheen tärkein osa oli vesikemiaan perehtyminen, koska minulla ei ollut juurikaan lähtötietoja vedenpuhdistuksen kemiasta. Tämä osaltaan helpotti hahmottamaan itse puhdistuskontin prosessia, ja selkeytti erikoisvaatimuksia, joita itse prosessi asettaa ohjausjärjestelmälle.

Opinnäytetyöni haasteina olivat itse prosessin venyminen, aikataulutus sekä erinäiset tekniset ongelmat, jotka kuitenkin saatiin lopussa ratkaistua. Aikataulun takia sekä muista syistä itse ohjausjärjestelmän koeponnistusta ei ehditty tekemään opinnäytetyöprosessin aikana.

Haluaisin tässä opinnäytetyön lopussa kiittää kaikkia työkavereita, opettajia sekä muita tukijoukkoja, joita ilman en olisi onnistunut näinkään mallikkaasti.

10 Lähteet

E-artikkeli

1. Sosiaali- ja terveysministeriö. Liite I, Talousveden laatuvaatimukset ja -suositukset (enimmäisarvot). [Internet]. [Viitattu 10.1.2024]. Saatavilla: https://stm.fi/documents/1271139/5001813/Liitteet+I-III+Talousveden+laatuvaatimukset+ja+-suositukset.pdf

Kirjat:

2. Nalco Company. The Nalco Water Handbook Third Edition. 2009

3. Aaltonen Kalevi – Torvinen Seppo. Konepaja-automaatio. WSOY Konepajan Tuotantotekniikka. 1997

4. Bolton W. Control engineering. Longman Pub Group. 1998

Kuvat sekä kaaviot:

1. Heikkinen J. WATERPRO_RENDER.9.2020. [Kansikuva]. 2020 [10.1.2024]

2. Heikkinen J. WaterPro-pilot container schematic V4.7. [Kaavio]. 2021

   [10.1.2024]

3. AQUA TECH. backwash-pool-filter.jpg. [Kuva]. 2019. [viitattu 1.4.2021] Saatavilla: https://aqua-tech.ca/wp-content/uploads/2019/05/backwash-pool-filter.jpg

4. Heikkinen J, KAMK. Venttiilien ohjaukset. [Kaavio]. 2021 [10.1.2024]

5. Beckoff Automation Oy. WaterPro järjestelmäkaavio. [Kuva]. 2021 [10.1.2024]