Topi Purhonen

Vihivaunun lopputestauksen öljynpuhdistuksen automatisointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

Opinnäytetyö

23.1.2023

Tiivistelmä

Tekijä:	Topi Purhonen
Otsikko:	Vihivaunun lopputestauksen öljynpuhdistuksen automatisointi
Sivumäärä:	25 sivua + 1 liite
Aika:	23.1.2023
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikka
Ammatillinen pääaine:	Automaatiotekniikka
Ohjaajat:	Test Manager Petri Soikkeli Lehtori Timo Tuominen

Opinnäytetyön tavoitteena oli automatisoida vihivaunun lopputestauksessa tapahtuva hydrauliikkaöljyn pudistus. Pudistuksen automatisoinnin tarkoituksena on nopeuttaa testausvaihetta ja vapauttaa vaunutestaaja muihin testaustehtäviin.

Mitsubishi Logisnextille tehtävässä työssä on tarkoitus löytää ja luoda kommunikaatioyhteys vihivaunun ja analysaattorin välille. Kommunikaatioyhteyden löydyttyä vaunun on tarkoitus aloittaa ohjelma, jossa se suorittaa erilaisia liikesarjoja, jotta hydrauliikkaöljy lähtee liikkumaan vihivaunussa. Liikesarjojen jälkeen nostetaan masto ylös ja annetaan sen valua alaspäin, jolloin analysaattori analysoi ohivirtauksen avulla, kuinka suuria likapartikkeleita hydrauliikkaöljyssä on. Kun hydrauliikkaöljy on tarpeeksi puhdasta vihivaunu ilmoittaa testaajalle olevansa valmis.

Ohjelman luontia vaikeutti työn nopea aikataulu sekä käytettävissä olevien vihivaunujen yksipuolisuus. Toimiva aliohjelma saatiin kuitenkin luotua, jonka pohjalta on hyvä lähteä rakentamaan lopullista ratkaisua. Ohjelmalla pystytään vapauttamaan vaunutestaaja muihin työtehtäviin, kun on tietynlainen vaunutyyppi puhdistuksessa.

Avainsanat: Automaatio, Vihivaunu, Hydrauliikka, Sisäänrakennettu hiukkasmonitori, Ohjelmointi

Abstract

Author:	Topi Purhonen
Title:	Automation of Oil Cleaning in the End Testing of AGV
Number of Pages:	25 pages + 1 appendix
Date:	23 January 2023
Degree:	Bachelor of Engineering
Degree Programme:	Electrical and Automation Engineering
Professional Major:	Automation Engineering
Supervisors:	Petri Soikkeli, Test Manager
	Timo Tuominen, Senior Lecturer

The aim of the thesis work was to automate the cleaning of the hydraulic oil during the final testing of an AGV (Automated Guided Vehicle). The purpose of automating the cleaning is to speed up the testing phase and free the vehicle tester for other testing tasks.

The work was carried out for Mitsubishi Logisnext, and the aim was to find and create a communication connection between the AGV and the analyzer. After the communication connection is found, the AGV is supposed to start a program, in which it does all the movement sequences so that the hydraulic oil starts moving in the AGV. After the series of movements, the mast is lifted up and allowed to flow down, whereupon the analyzer analyzes the size of dirt particles in the hydraulic oil using the bypass flow. When the hydraulic oil is clean enough, the AGV notifies the tester that it is ready.

The creation of the program was complicated by the tight work schedule, as well as the paucity of the available AGVs. Nonetheless, a functioning sub-programme was created, on the basis of which building the final solution can be started. With the program, it is possible to release the vehicle tester for all work tasks when there is a certain type of AGV being cleaned.

Keywords: Automation, Automated guided vehicle, Programming

Sisällys

Lyhenteet

AGV: Automated Guided Vehicle. Vihivaunu.

ATX: Automatic Truck X. Vihivaunumalli.

ART: Automated Reach Truck. Vihivaunumalli.

AWT: Automated Warehouse Truck. Vihivaunumalli.

PLC: Programmable Logic Controller. Ohjelmoitava logiikka.

VAD: Vehicle Application Designer. Kollmorgenin luoma työkalusovellus.

CVC: Compact Vehicle Controller. Vihivaunuille tarkoitettu ohjainyksikkö, valmistaja Kollmorgen.

1 Johdanto

Mitsubishi Logisnext on kansainvälinen konserni, joka on keskittynyt luomaan älykkäitä logistiikkaratkaisuja toimittajille. Osa toiminnasta on sijoitettu Suomeen, jossa se toimii nimellä Mitsubishi Logisnext Europe Oy, joka on aikaisemmin tunnettu nimellä Rocla Oy. Yrityksen päätoiminen tehtävä on suunnitella ja valmistaa vihivaunujärjestelmiä sekä trukkeja. Yritys on tehnyt yli 1000 asiakasprojektia, joissa se on toimittanut yli 7000 vihivaunua. [1.]

Työn tarkoituksena on kehittää vihivaunun lopputestaukseen uusi työkalu, joka vapauttaisi testaajan muihin tehtäviin sekä mahdollisesti nopeuttaisi vaunun testauksen läpivienti aikoja.

Hydrauliikkaöljyn puhdistus on osa lopputestausta, jonka automatisointia on pohdittu pitkään. Prosessissa vihivaunu kytketään kiinni mittauslaitteiseen, joka tulkitsee lasertunnistimen avulla likapartikkeleiden suuruutta hydrauliikkaöljyn seasta. Tunnistaminen tapahtuu ohivirtauksen avulla, joten vaunun eri hydrauliikkaominaisuuksia on välillä liikuteltava, jotta saadaan kaikki mahdollinen öljy analysoitua. Öljyä puhdistetaan samaan aikaan vihivaunun öljytankkiin kiinnitetyn pumpun avulla.

Tällä hetkellä hydrauliikkaominaisuuksien liikuttelu tapahtuu manuaalisesti, jolloin testaajan täytyy olla läsnä, kun liikkeet suoritetaan. Automatisointia varten täytyy suunnitella ohjelma, joka tekisi liikkeet itsenäisesti sekä ilmoittaisi testaajalle, milloin ohjelma on päättynyt onnistuneesti.

2 Vihivaunut

Automated Guided Vehicle (lyhent. AGV) tarkoittaa suomeksi vihivaunua. Vihivaunu on puolestaan käsite automaattitrukille, joka osaa liikkua ja suorittaa sille annetut tehtävät itsenäisesti. Vihivaunut on luotu optimoimaan sisälogistiikan turvallisuutta ja tehokkuutta.

Mitsubishi Logisnext tuottaa tällä hetkellä kolmea erityyppistä vihivaunua, jotka ovat ATX, ART ja AWT. Malli määrittää millainen vihivaunun rakenne on, mutta lastinkäsittelylaite on täysin kustomoitavissa asiakkaan tarpeiden mukaan. Yleisimpiä lastinkäsittelylaitteita ovat haarukat (fork) ja pihdit (clamp).

2.1 ATX (Automatic Truck X)

ATX on kooltaan pienen vaunumalli (kuva 1). Malli on suunniteltu matalille nostokorkeuksille, ja sen pääasialliset lastit ovat lavoja. ATX voi nostaa lavoja maksimissaan 1,7 metrin korkeuteen, sen nimelliskuormitus on 1250 kg. ATX-mallista on tulossa ISO-standardin mukainen versio, jonka tarkat mitat eivät ole vielä julkista tietoa.

ATX-vaunua pystytään hyödyntämään esimerkiksi varastotyöntekijän apuvälineenä. Esimerkiksi työntekijä poimii tavarat keräilyalueelta ja lastaa ne automaattiseen trukkiin jatkokuljetuksia varten. [2.]

Kuva 1. 3D-mallinnus ATX-vaunusta.

2.2 AWT (Automated Warehouse Truck)

AWT puolestaan on isoin vaunumalli (kuva 2). Malli on suunniteltu varastokäyttöön, jossa vaaditaan enemmän nostokorkeutta ja -voimaa. Riippuen lastinkäsittelylaitteesta AWT-malli pystyy nostamaan parhaimmillaan jopa 8,5 metrin korkeuteen sekä nostamaan 8000 kg painavia lasteja.

Esimerkiksi vastapaino-AWT on suunniteltu tilaan, jossa lavakuljetukset tai varastot edellyttävät työntö- ja sisäänajohyllyjä. Sen nostokorkeus on 8,4 metriä ja sallittu kuorma 2 500 kilogrammaa. [3.]

Kuva 2. AWT-vastapainovihivaunu

2.3 ART (Automated Reach Truck)

ART on vaunumalleista tuorein, sillä se tuli markkinoille vuonna 2019 (kuva 3). ART poikkeaa muista vaunumalleista työntömastonsa ansiosta. Työntömasto ja uusi sensoriteknologia yhdessä mahdollistavat vaunun nopean toiminnan ahtaissa tiloissa, jopa vain 3 metriä kapealla käytävällä. ART pystyy nostamaan jopa 10 metrin korkeuteen dynaamisen ajonvakautusjärjestelmänsä ansiosta. Järjestelmä valvoo jatkuvasti ajonopeutta, kuorman painoa, korkeutta sekä työntömaston positiota. [4.]

Kuva 3. 3D-mallinnus ART-vaunusta.

3 Laitteisto ja ohjelmistot

Opinnäytetyön kannalta tärkein laitteisto koostuu pääasiassa vihivaunun pääohjaimesta CVC-700 ja IPD Counter -analysaattorista. Lisäksi laitteistoon kuuluu

kommunikaatiokaapeli, joka kootaan itse. Ohjelmat, joita työssä käytetään pääasiassa, ovat Infoteamin OpenPCS ja Kollmorgenin Vehicle App Designer.

3.1 CVC-700

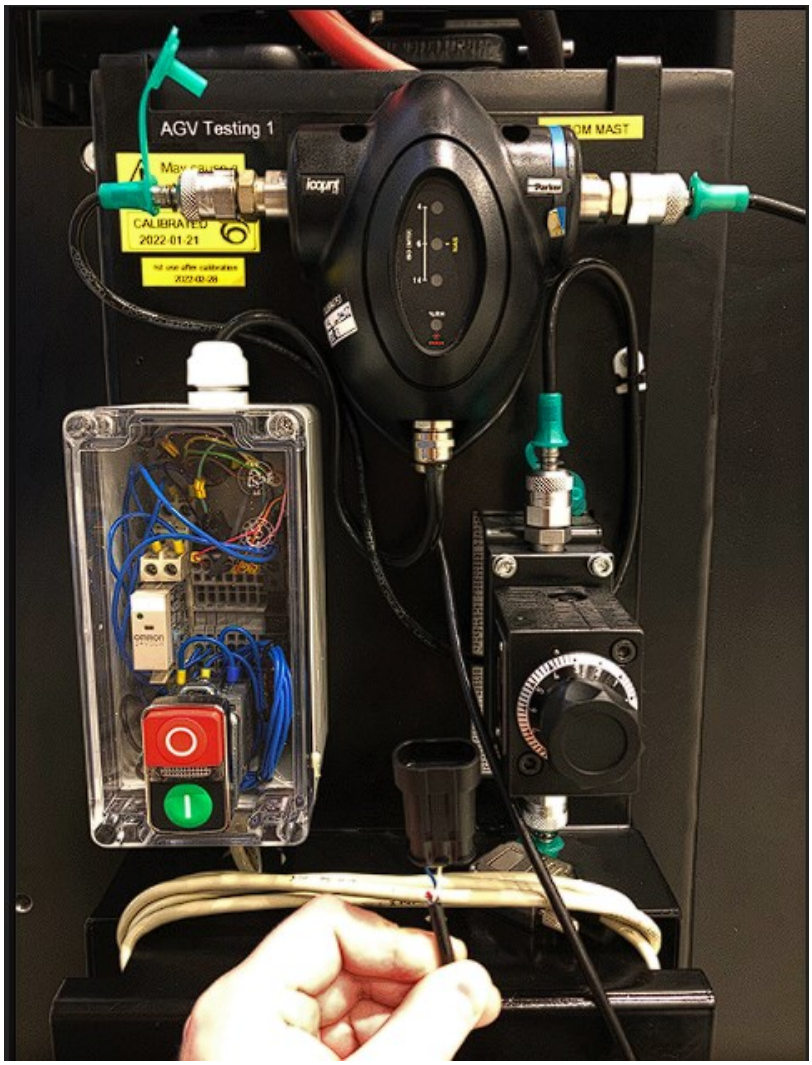
CVC-700 (Compact Vehicle Controller) on Kollmorgenin valmistama pääohjain, jota Mitsubishi Logisnext käyttää vihivaunuissaan (kuva 4). Pääohjain soveltuu hyvin vihivaunun käyttöön, sillä se tukee kaikkia mahdollisia navigointitekniikoita sekä se on helposti muokattavissa avoimen arkkitehtuurinsa ansiosta. Inputeja löytyy yhteensä 23, joista 11 on analogiasia ja loput digitaalisia. Kommunikointia varten CVC:ssä on 3 kpl CAN-väyliä, 2 kpl LAN-portteja, RS232/422/485-portit sekä se tukee Bluetooth ja Wi-Fi-yhteyttä. [5.]

Kuva 4. CVC-700.

3.2 iCountPD-analysaattori

Parkerin luoma iCountPD on hiukkasmonitori, joka toimii lasertunnistuksella. Laser tunnistaa ohivirtauksen avulla hydrauliikkaöljyssä olevien epäpuhtauksien koon (kuva 5). Analysaattori pystyy analysoimaan jopa 60 litraa öljyä minuutissa, joista se muodostaa raportin 30 sekunnin välein. [6.] 30 sekunnin välein

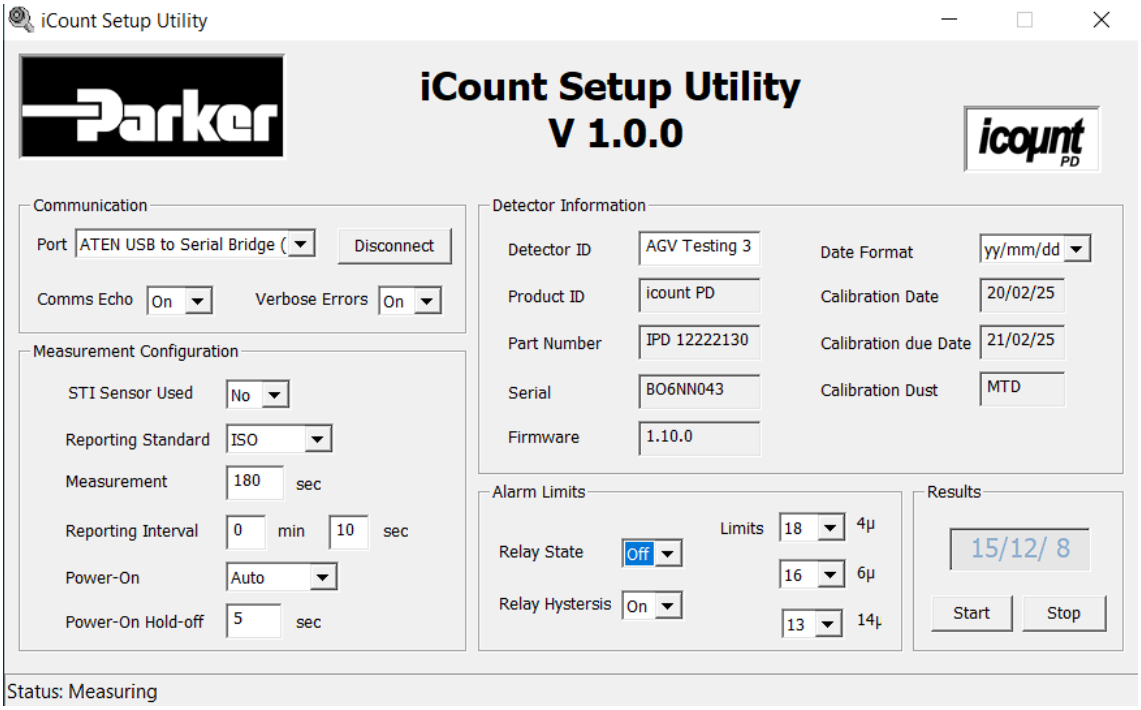
raportti ilmoitetaan käyttäjälle LED-valojen avulla. LED-valot palavat punaisena, jos likapartikkelien koko on liian suuri. Valot palavat puolestaan vihreänä, jos likapartikkelit ovat tarpeeksi pieniä. Käyttämässämme analysaattorissa on lisäominaisuutena relelähtö, jonka avulla pystymme tuomaan tiedon öljyn puhtaudesta vaunun PLC:lle. Analysaattoriin on saatavilla lisäoptiona myös kosteusanturi.



Kuva 5. iCountPD:n analysaattori. Kädessä näkyy relelähdön liitin.

Analysaattorilla on myös oma tietokoneohjelma, jonka avulla pystytään muokkaamaan käytetyn analysaattorin asetuksia. Asetuksista pystyy muokkaamaan mittauksen raportointistandardia ja mittauksen ajanottoväliä. Asetuksista pystyy

myös muokkaamaan releen statusta sekä muokkaamaan likapartikkelien koon raja-arvoja (kuva 6).



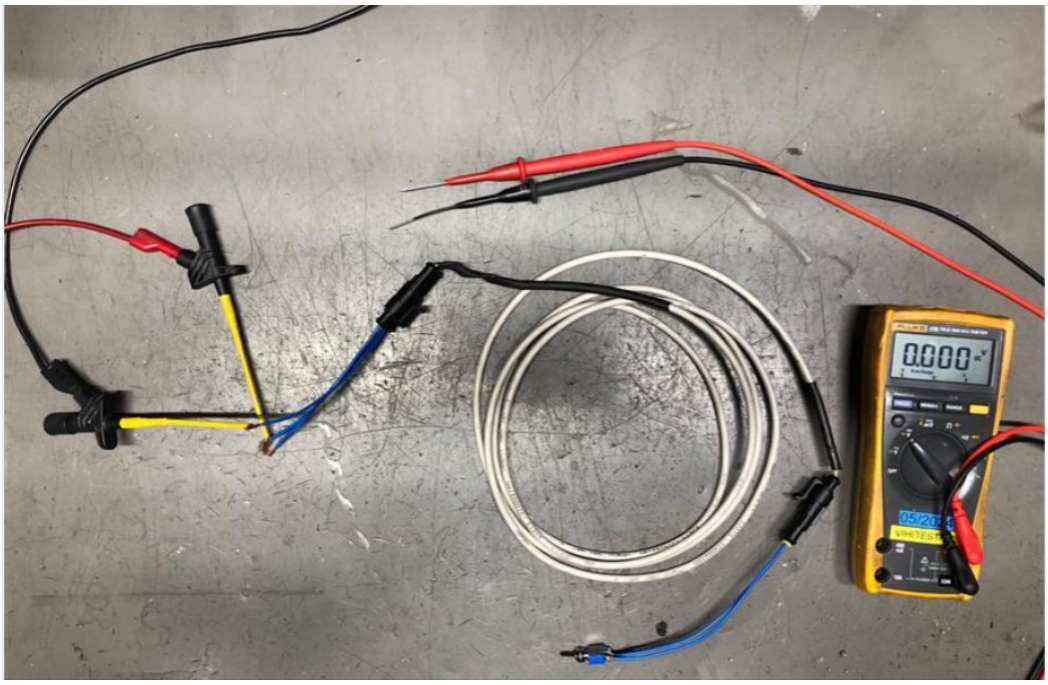
Kuva 6. Kuvakaappaus Parker iCountPD:n omasta ohjelmasta.

3.3 Kommunikaatiokaapeli

Kommunikaatiokaapeli tarvitaan välittämään analysaattorin tuottamia jännite-viestejä eteenpäin vihivaunun PLC:lle. Kaapeliin tulee 2-pinninen liitin analysaattorin relelähdölle, sekä toiselle puolelle 3-pinninen liitin, joka kytketään paineanturin tilalle. Paineanturia ei tarvita öljynpuhdistuksen aikana, ja se on helppo tapa saada jänniteviesti läpi vaunun PLC:lle, joten se korvataan prosessin ajaksi. Paineanturi on myös kaikille vaunuille yhteinen liitäntäkohta, jota ei tarvita hydrauliikkaöljyn puhdistamisessa. ATX-vaunumallilla on erilainen liitin paineanturissa kuin AWT- ja ART-malleista, joten sitä varten oli myös luotava adapteri (kuva 7).

Kuva 7. Vasemmalla kommunikaatiokaapeli ja oikealla adapteri ATX-vihivaunulle.

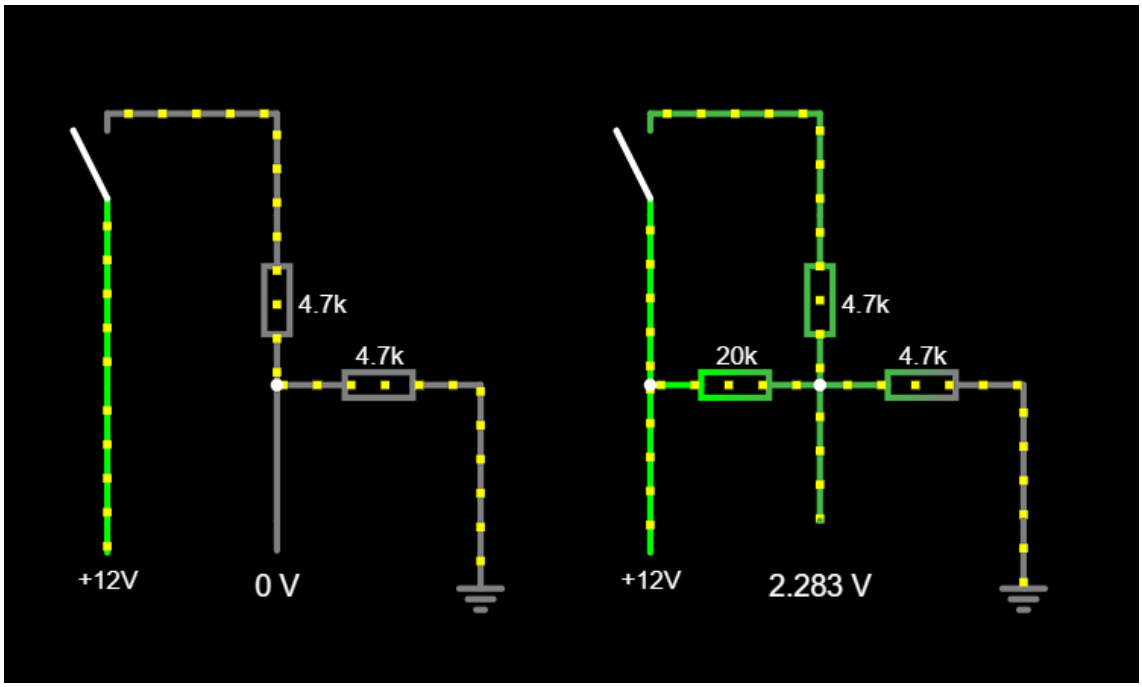
Ensimmäiseksi ajateltiin, että kytkentä tulee tehdä niin, että kun analysaattorin rele on OFF-tilassa, sen jänniteviestin tulee olla 0 V ja ON-tilassa puolestaan 6 V. Alkuperäisessä ideassa oli 2 kappaletta 4.7 kΩ vastusta. Kytkentää testattiin ensiksi asettamalla virransyöttö vaunun puoleiseen liittimeen ja releen puoleiseen liittimeen vipukytkin, joiden avulla pystyttiin demonstroimaan kytkennän toimivuutta (kuva 8).



Kuva 8. Kytkennän toimivuuden testausta varten koottu laitteisto.

Kun kytkentää testattiin itse vaunussa, todettiin, että kytkentä on virheellinen, sillä vihivaunu meni heti vikatilaan, jonka syyksi todettiin alijännite. Alijännite lamaannutti moottoriohjain AZIO:n, joka puolestaan esti vihivaunua liikkumasta.

Moottoriohjain vaatii vähintään 0.5 V syötön koko ajan toimiakseen, joten kytkentää muokattiin sen mukaisesti. Kytkentään lisättiin yksi 20 kΩ vastus ohittamaan releen kytkin, jolloin jänniteviesti pääsee läpi, vaikka kytkin olisi auki. Nyt releen ollessa OFF-asennossa AZIO:lle syötetään noin 2 V ja ON-tilassa puolestaan noin 6 V. Tämä kytkentä todettiin toimivaksi tähän sovellukseen.



Kuva 9. Vasemmalla alkuperäinen kytkentäsuunnitelma ja oikealla lopullinen.

3.4 Työkaluohjelmat

Vaunuohjelman luontiin käytetään kahta ohjelmaa: Infoteamin OpenPCS:ää ja Kollmorgonenin Vehicle App Designeria.

OpenPCS on ohjelmointiympäristö, joka tukee IEC 61131-3 -standardin ohjelmointikieliä. Vaunuohjelmien ohjelmoinnissa kielenä käytetään pääsääntöisesti Structured textiä (ST). OpenPCS on optimoitu mikrokontrollereille, joilla on rajoitettu tallennustila ja korkea suorituskyky. [7.]

Kuva 10. OpenPCS -ohjelman aloitusnäkymä.

Vehicle App Designer on ohjelmatyökalu, jonka avulla voidaan kääntää ja siirtämää OpenPCS:llä luodut ohjelmat vihivaunun CVC:lle. Pääsääntöisesti vaunutestaaja käyttää VAD:ia seuratakseen ja muokatakseen vihivaunun parametrejä. Ohjelmalla pystytään myös luomaan COM-porttien avulla LAN-yhteys laitteiden ja CVC:n välille.

Kuva 11. Vehicle App Designer -ohjelman aloitusnäkymä.

4 Hydraulic Flushing -aliohjelma

Luotavan ohjelman on tarkoitus olla aliohjelmana pääohjelmassa, joka voidaan käynnistää tiettyjen ehtojen täyttyessä. Aliohjelman ideana on suorittaa hydrauliikkaöljyn puhdistus automaattiajolla niin, että vaunutestaajan läsnäoloa ei vaadita ohjelman ollessa käynnissä. Ohjelman olennaisiin ominaisuuksiin kuuluu tunnistaa, minkälainen lastinkäsittelylaite on käytössä, mitä liikkeitä sen huuhteleminen vaatii sekä öljyn kriittinen analysointi.

4.1 State Machine

Ohjelmaa lähdettiin suunnittelemaan jo valmiina olevaan TestingMast-Function Blockiin, joka pitää sisällään kaksi valmista aliohjelmaa: Mast Adjustin ja Leakage Testin.

Mast Adjustissa komennetaan vaunun masto tiettyyn korkeuteen, johon masto ajaa, jonka jälkeen katsotaan paljon on eroa komennetun ja mitatun korkeuden välillä. Ohjelman avulla pystytään säätämään nostonopeutta haluttaessa, jonka avulla pystytään määrittämään maston liike niin, että komennettu korkeus ja mitattu korkeus ovat lähes identtiset aina.

Leakage Testissä ajetaan mastoa ylös ja alas painolastin kera niin pitkään kuin moottoriöljy on yli 40°C lämmintä. Kun haluttu lämpötila on saavutettu, masto ajetaan ylös ja annetaan sen seisoa siellä kymmenen minuuttia. Seisonnan aikana seurataan, ettei hydrauliikkajärjestelmässä ole sisäisiä tai ulkoisia vuotoja. [8.]

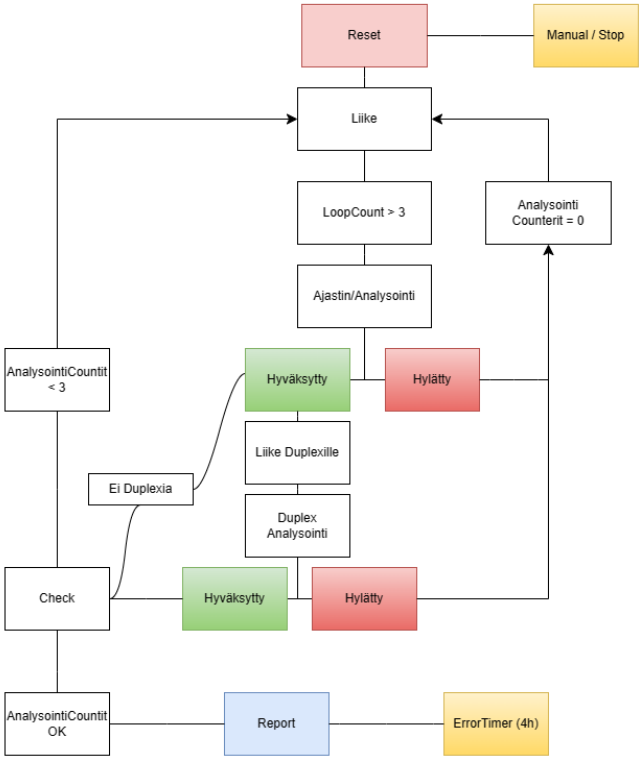
Kyseiset ohjelmat sisältävät paljon koodia, jota voidaan hyödyntää puhdistusohjelmassa. Esimerkiksi molemmat ohjelmat on luotu State Machine -malliin. Lisäksi niissä on monia komentoja, joita tullaan soveltamaan.

Ohjelmaa lähdettiin suunnittelemaan aikaisemmin mainittuun State Machine -malliin. State Machine eli tila-automaatti toimii niin, että ohjelma lukee aina yh-

den tilan kerrallaan ja sen sisältämän koodin. Koodin luettuaan se tekee käskettyt komennot, jonka jälkeen ehtojen täytyttyä se siirtyy seuraavaan tilaan. Tiloja voidaan edetä systemaattisesti tai niihin voidaan palata myöhemminkin.

Tiloja ohjelmassa tulee olemaan useita, mutta ne voidaan jakaa toimintaperiaatteensa vuoksi viiteen eri kategoriaan:

- resetointitila
- liiketila
- ajastin/analysointitila
- tarkistustila
- raportointitila.



Kuva 12. Hydraulic Flush -aliohjelman kierto.

4.1.1 Resetointitila

Resetointitila on erittäin yksinkertainen, mutta todennäköisesti tärkein tila. Tilan tarkoituksena on nollata kaikki mahdolliset arvot ja laskurit, sekä palauttaa vaunu ns. normaaliin tilaan.

Resetointitila menee päälle, kun käyttäjä painaa Stop-painiketta tai kääntää avaimesta vihivaunun manuaaliajotilaan. Tilasta poistutaan asettamalla avaimella ensin vihivaunu automaattiajolle, jonka jälkeen aloitetaan puhdistus-ohjelma Start-painikkeella.

4.1.2 Liiketila

Ohjelmakierrossa resetointitilan jälkeen siirrytään liiketilaan, jossa vaunu suorittaa liikkeen, joka on ohjelmoitu siihen tilaan.

Tilassa ensimmäiseksi kutsutaan Main-koodia, jonka avulla vaunu tunnistaa, onko kyseistä lastinkäsittelylaitetta asennettu ja näin mahdollista tehdä liike. Esimerkiksi koodissa käytetään ehtolausetta, jos työntömasto on asennettu, niin suoritetaan seuraavat komennot. Jos työntömastoa ei ole asennettu, siirrytään seuraavaan tilaan.

Tilasta siirrytään seuraavaan, kun liikekomento saavuttaa halutut raja-arvot. Esimerkiksi masto on komennettu parametrin ”HydraulicFlush.LiftMax-Height_INT” arvon mukaiseen korkeuteen. Mikäli maston mitattu korkeus on komennon jälkeen raja-arvojen sisällä, voidaan siirtyä seuraavaan tilaan.

4.1.3 Ajastin/analysointitila

Ajastin- ja analysointitiloissa seurataan analysaattorin antamaa informaatiota. Tilan alussa aloitetaan ajastin, joka tässä sovelluksessa on 10 minuuttia. Seuraavaan tilaan voidaan siirtyä, kun ajastimen aikana rele on ollut ON-asennossa koko ajan. Tällöin voidaan todeta, että likapartikkelit ovat olleet tarpeeksi pieniä

kyseisen ajan. Tällöin lisätään laskuriin yksi puhdas analysointi lisää ja siirrytään tarkistustilaan.

Jos rele käy OFF-tilassa, on hydrauliikkaöljyssä ollut liikaa epäpuhtauksia. Tällöin keskeytetään ajastin ja nollataan puhtaiden analysointien laskurit, jonka jälkeen siirrytään takaisin liiketiloihin.

4.1.4 Tarkistustila

Tarkistustila on yksinkertainen tila, jossa katsotaan, mihin tilaan liikutaan seuraavaksi. Tilaan saavutaan aina Ajastin/analysointitilasta ja katsotaan, montako puhdasta analysointia on tehty. Jos maston maksimiasennosta ja vapaanostoalueelta on molemmista suoritettu kolme puhdasta analysointia, niin siirrytään raportointitilaan. Mikäli puhtaita analysointeja on vähemmän, siirrytään takaisin liiketiloihin.

4.1.5 Raportointitila

Raportointitilat ovat ns. umpikujia, sillä niistä ei liikuta enää muihin tiloihin. Tilan tarkoitus on ilmoittaa vaunutestaajalle, että vaunu tarvitsee käyttäjän huomiota. Vaunun saavuttua raportointitilaan, se alkaa väläyttelemään punaisia valoja testaajan huomion herättämiseksi. Vaunun näytöltä voidaan lukea tapahtumalokista, mikä vaunun tila on ja miksi se tarvitsee käyttäjää. Lokitiedoissa voi lukea, että vaunu on suorittanut ohjelman onnistuneesti tai vaihtoehtoisesti se on mennyt vikatilaan.

4.2 Raja-arvot

Vaunumalleissa on hieman poikkeavuuksia painoanturin kytkennässä, joka vaikuttaa kytketyn analysaattorin antamiin jänniteviesteihin, jotka puolestaan vaikuttavat painon raaka-arvojen suuruuteen. Esimerkiksi releen ollessa OFF-asennossa ATX:n antama raaka-arvo on 94, AWT:n 405 ja ART:in 850. ON-

asennossa puolestaan ATX:n antama raaka-arvo on 168, AWT:n 1023 ja ART:in 1023.

Raaka-arvojen erilaisuus on pakottanut tekemään koodiin useita ehtolauseita, jotka ovat riippuvaisia ”vaunubooleanista”. Vaunuboolean on parametri, joka kertoo, mikä vaunumalli on puhdistusohjelmassa. Esimerkiksi HF_ART_B-parametri on tarkoitettu ART-malleille. Jos parametrin arvo on 1, käytettävä vaunumalli on ART. Arvon ollessa 0 kyseessä on silloin toinen vaunumalli, sekä toinen vaunuboolean on tällöin arvoltaan 1, esimerkiksi HF_AWT_B = 1.

Boolean määrittää, mitä arvoja ohjelman koodissa luetaan. Esimerkkinä tilan siirtymisehtona voidaan käyttää ehtolausetta, jossa käytetään vaunubooleania ja raaka-arvojen tulkintaa: ”If HF_ATX_B = 1 AND PressureInput_INT > 100 THEN State_STR := ’Hyde_Mast_Low’;”. Toisin sanoen: ”Jos ATX:n vaunubooleanin ollessa 1 ja PressureInput_INT-parametrin ollessa suurempi kuin 100, siirytään liike-tilaan: Hyde_Mast_Low”.

4.3 TestingMast-blokin ulkopuoliset koodit

Vihivaunun ohjelma koostuu useasta Function Blockista, jotka saattavat tuottaa ongelmia ohjelman luonnissa. Esimerkiksi valojen ohjelmointi oli tehty lähes mahdottomaksi, jos kyseisiä ”blokkeja” ei käy peukaloimassa.

Valo-ongelma oli vielä suhteellisen helposti ratkaistavissa, tarvitsi lisätä vain ehtolause SignalsState-Function Blockiin, jossa komennettiin valot ”ReducedSafety”-tilaan, kun HF_LampSwitch_B = 1 (kuva 13). Tällöin vaunun valot alkavat vilkkumaan punaisena, kun LampSwitch_Booleanin arvo muuttuu ykköseksi.

```
if Tractor.EnergySystem.Command.ShutDownSignals_B then
    Features.Signals.Normal.State.Name_UT := PowerOff; DebugName_STR := 'PowerOff';
elsif Objects.Option.TestingMast.Runtime.HF_LampSwitch_B = true then
        Features.Signals.Normal.State.Name_UT := ReducedSafety; DebugName_STR := 'ReducedSafety'; (*Hydraulic Flushing Demo*)
elsif MI.UserControl.State.Tr.Auto_B then
    if MI.NavSystem.VehicleControl.Estop_B and not UserStopActive_NoEMS_B then
        Features.Signals.Normal.State.Name_UT := EMS; DebugName_STR := 'EMS';
```

Kuva 13. Kuvakaappaus SignalsState-Function Blockissa, johon on lisätty ehtolauseke valotilan muuttamiselle.

Vastaavanlaisia ehtolauseita jouduttiin tekemään myös SwayMonitor-Function Blockiin, koska siellä oli koodilausekkeita, jotka rajoittivat nostokorkeutta, kun painoarvot ovat liian suuria. Koska kommunikaatioyhteys on luotu paineanturin tilalle, jänniteviestiä releeltä tulkitaan painon raaka-arvona, joka tuottaa ongelmia etenkin releen ollessa ON-asennossa. ON-asennossa raaka-arvo on AWT:lla ja ART:lla 1023, joka skaalauksen jälkeen on kilogrammoissa noin 5500. Tämä tulkitaan ohjelmistossa helposti ylipainoksi, jolloin nostokorkeus rajoitetaan metrin korkeuteen. Ehtolauseen avulla pystytään ohittamaan nostokorkeutta rajoittavat koodin pätkät, jolloin mastoa pystytään ajamaan haluttuun korkeuteen, vaikka paino tulkittaisiinkin ylipainoksi. Ohjelman loputtua ylipainon tunnistus palaa normaaliksi.

```
(* Over weight *)
if Tractor.Com.Controllers.State.Rx.PumpOn_B then
    OverWeight_B := false;
elsif Objects.Option.TestingMast.Runtime.HF_SwayOff_B = true then OverWeight_B := false;
elsif MI.Mast.State.Tr.DuplexMastInChangingArea_B then
    OverWeight_B := MI.Mast.Measure.Tr.Weight_INT > (Tractor_Settings.Mast.MaxLoadingWeight_INT+MI.Mast.State.Tr.DuplexMastWeight_INT);
else
    OverWeight_B := MI.Mast.Measure.Tr.Weight_INT > Tractor_Settings.Mast.MaxLoadingWeight_INT;
end_if;
```

Kuva 14. Koodiin lisätty ehtolause, joka ohittaa maston korkeuden rajoittamisen.

Luotavan ohjelman yksi tavoitteista oli nopeuttaa hydrauliikkapuhdistuksen läpimenoaikoja. Normaalisti putsaukseen kului arviolta noin 4 tuntia, joten luotiin ajastin, joka siirtää vaunun vikatilaan, jos ohjelma on kestänyt 4 tuntia. Tarkoituksena on ilmoittaa testaajalle, että vaunussa tai ohjelmassa saattaa olla jotakin pielessä. Tällöin testaaja pystyy manuaalisesti tarkastamaan, onko puhdistus tapahtunut hyväksytysti tai päättämään mahdollisesta putsaamisesta uudelleen.

Koodissa täytyy muistaa myös määrittää, mitä analogista tuloa parametri HF_PressureInput_INT tulkitsee. Jokaisella vaunutyypillä painosensori on määritetty eri analogiatuloon, joten tämän määrittäminen oikein on todella tärkeää ohjelman toimivuuden kannalta. Ohjelmaan on asetettu ehto, jossa liian pieni HF_PressureInput_INT-arvo aiheuttaa vikatilan.

```
(*TestingMast Parameters*)

Objects.Option.TestingMast.Runtime.HF_AWT_B := true;
Objects.Option.TestingMast.Runtime.HF_PressureInput_INT  := UINT_TO_INT(NDC8.AZIO_1_22.AnalogInput02_UINT);
```

Kuva 15. Kuvakaappaus HardwareModels-Function Blockissa, jossa on ensin määritelty AWT-vaunun vaunuboolean. Alimmalla rivillä on määritelty, mitä analogiatuloa parametri HF_PressureInput_INT tulkitsee.

4.4 Turvallisuus

Turvallisuusseikat tuli ottaa huomioon myös ohjelmaa tehtäessä. Ohjelmaa testatessa huomasin, että vihivaunun ollessa automaattiajolla vaunun tehdessä liikkeitä se antaa äänimerkin ja väläyttelee keltaisia valoja. Analysointitilassa vaunu ei anna minkäänlaisia signaaleja, että se on käynnissä, joten on hyvä eristää maston nostoalue huomioita herättävillä kartioilla. Kartioiden tehtävä on huomauttaa alueella liikkuvalle, että alueen sisälle ei mennä turvallisuussyistä.

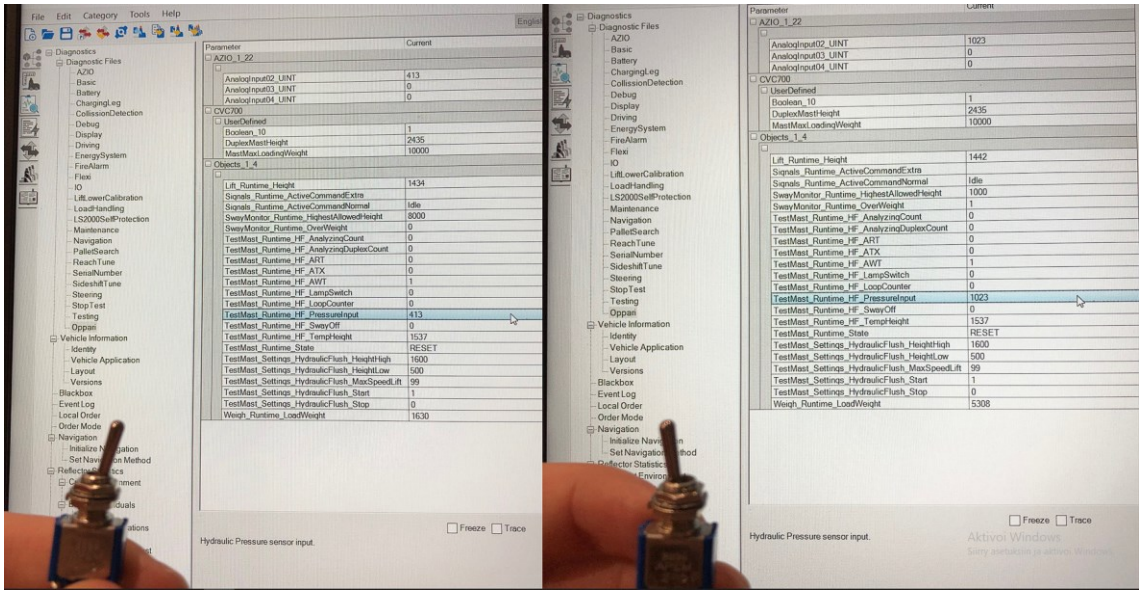
Vaunussa olevat SICK-laserskannerit toimivat yhä normaalisti, vaikka aliohjelma olisikin päällä. Laserskannerit estävät vaunun liikkeen, jos se havaitsee esteen omassa suojakentässään. Liike jatkuu, kun este on poistunut suojakentästä.

5 Testaus

Ohjelmaa testattiin vastapaino AWT:lla, jossa ainoa hydrauliikkaa vaativa liike oli maston ajaminen. Testauskäyttöön ei valitettavasti kerennyt saapua vihivaunua, jossa olisi ollut enemmän hydrauliikkaa vaativia ominaisuuksia, joten suurin osa liiketilojen koodista jäi testaamatta.

Ensimmäinen testausvaihe oli ohjelmakäännöksen teko, jossa kokeiltiin aluksi vain komentoja, jotka käskivät mastoa liikkumaan haluttuun positioon. Seuraavaksi testaukseen lisättiin ajastintilat, jotta saatiin käsitys niiden toiminnallisuudesta. Viimeiseksi lisättiin ns. laskurit, jotta saatiin perusta ohjelmakierrolle luotua.

Seuraavassa testausvaiheessa kiinnitimme kommunikaatiokaapelin vihivaunuun. Kaapelin toisessa päässä oli vipukytkin demonstroimassa analysaattorin relelähdön toimintaa (kuva 16). Vipukytkimen avulla pystyttiin testaamaan eri skenaarioita siitä, mitä tapahtuu, kun relelähdön tila muuttuu kesken ohjelman. Lisäksi vipukytkimen avulla saatiin selville raaka-arvot, joiden avulla pystytään määrittämään ohjelman kulkua sekä määrittämään, millä arvoilla vaunu menee virhetilaan.



Kuva 16. Ohjelma testausta vipukytkimen avulla. Vipukytkimellä demonstroitiin releen asento. Vasemmalla olevassa kuvassa rele OFF-asennossa, jolloin painon raaka-arvo on 413. Oikealla kytkimen asennon muuttuessa, releen ollessa ON-asennossa, raaka-arvo on 1023.

Vipukytkintestauksen ansiosta ohjelman perustoiminnot olivat hyvällä mallilla, joten oli aikaa tehdä pientä hienosäätöä. Tässä vaiheessa tutkittiin, miten saadaan viestittyä käyttäjälle, että ohjelma on päättynyt tai mennyt vikatilaan. Lopputuloksena lisättiin tilatiedon lähettäminen vihivaunun näytölle ja punaisten valojen vilkkuminen.

Viimeisessä testausvaiheessa ohjelmaa kokeiltiin tositoimissa. Analysaattori ja pumppu liitettiin vihivaunuun kiinni (kuva 17), jonka jälkeen ohjelma laitettiin käyntiin.

Kuva 17. Pumppu ja analysaattori kiinnitettynä vihivaunuun.

Ohjelma suoriutui kohtalaisesti ensimmäisestä ns. tulikokeesta, mutta todettiin, että analysoinnin täytyy olla luotettavampi. Luotettavuutta saatiin lisäämällä toinen näytteenotto kohta mastonvaihtokohtaan ja vaatimalla useampia puhtaita analysointeja. Seuraava koe suoritettiin eri vaunulla, jota ei myöskään ollut puhdistettu. Tämän kertainen tulos oli huomattavasti luotettavampi kuin edellisen vihivaunun.

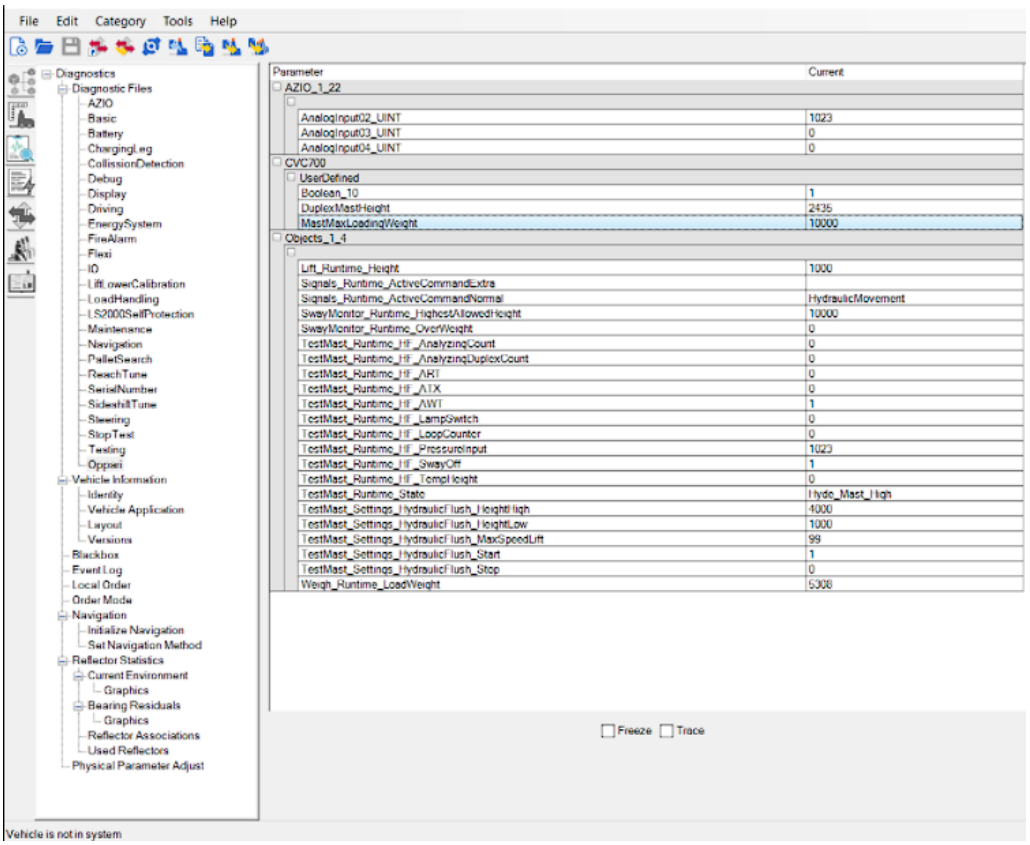
Kuva 18. AWT-vihivaunu puhdistusohjelmassa. Punaiset valot palaa, joten vaunu pyytää vaunutestaajan huomiota.

6 Ohjelman käynnistysohjeet

Ensimmäiseksi tarkistetaan analysaattorin kytkennät, jotta ei tapahdu öljyvuotoja. Seuraavaksi tarkistetaan kommunikaatiokaapelin kytkennät, jonka jälkeen voidaan käynnistää vihivaunu ja analysaattori.

Seuraavaksi tarkastetaan ja muokataan VAD:sta parametrit haluttuihin arvoihin (kuva 19):

- HeightHigh = maston nostokorkeuden yläraja.
- HeightLow = maston nostokorkeuden alaraja.
- MaxMastLoadingWeight = 10000.
- Vaunuboolean vastaa vaunumallia.
- PressureInput_INT antaa oikeanlaisia arvoja.



Kuva 19. Kuva VAD:ista, jossa on lähtöparametrit on säädetty ohjelman aloituksen mukaisiksi.

Parametrien ollessa oikein ajetaan masto alle metrin korkeuteen, jonka jälkeen avataan analysaattorin venttiiliä hieman. Venttiiliä avataan sen verran, että maston korkeus putoaa hieman hitaammin kuin 1 mm/s. Tämän jälkeen voidaan

kääntää avaimella vihivaunu automaattiajolle, jonka jälkeen ohjelma voidaan aloittaa.

7 Yhteenveto

Projektin tavoitteena oli automatisoida vihivaunun lopputestauksessa tapahtuva hydrauliikkaöljyn puhdistaminen. Tavoitteena oli nopeuttaa vaiheen läpimenoaikoja ja vapauttaa vaunutestaaja muihin työtehtäviin. Työ edellytti kommunikaatioyhteyden löytämistä puhdistuksessa käytettävän analysaattorin ja vihivaunun välillä, sekä aliohjelman luontia. Työn tavoitteissa onnistuttiin, sillä puhdistus saatiin automatisoitua, sekä vaunutestaaja saatiin vapautettua muihin työtehtäviin.

Työssä on vielä kuitenkin parantamisen varaa, sillä ohjelma toimii tällä hetkellä vain tietyntyyppisissä vaunumalleissa. Esimerkiksi vaunumalleissa, joissa on monimutkaisempi lastinkäsittelylaite, kyseinen ohjelma ei ole vielä käyttökelpoinen. Kaikkia ohjelmaan suunniteltuja liiketiloja ei ole testattu, koska testauksen käytettävissä ei ollut vaunumalleja, joissa olisi ollut vaadittavia lastinkäsittelylaitteita.

Työn olisi tarkoitus seuraavaksi mennä tuotekehityksen käsittelyyn, jossa ohjelman toimintoja ja ominaisuuksia viilattaisiin vielä hieman enemmän. Tämän jälkeen aliohjelma lisättäisiin base-vaunuohjelmiin, jonka jälkeen siitä tulisi olennainen työkalu testausryhmälle. Base-vaunuohjelman ympärille luodaan projektikohtaiset ohjelmat, joten työkaluohjelma olisi aina saatavilla.

Kehitysehdotuksia tuotekehitykselle voisi olla muiden hydrauliikkaa vaativien liikkeiden automatisointi, analysoinnissa olevien arvojen/aikojen optimoiminen, sekä onko vaunutyypeillä eroavaisuuksia ajallisesti. Myös valoille voisi kehittää oman tilan SignalsState-Function Blockiin. Tämänhetkinen järjestely voi tuottaa myöhemmin ongelmia, jos on tarvetta tehdä muutoksia ylipäätänsä valoihin.

Tuotekehitykseltä itseltään tuli pari kehitysideaa. Ensimmäisenä vaunun tulisi ilmoittaa käyttäjälle matalasta akun tilasta, jotta vältyttäisiin akun loppumiselta. Akun loppuminen on täysin mahdollista, koska tuotannosta tulleiden vihivaunujen akkujen tasot ovat yleensä matalia. Vaunut on myös tapana laittaa suoraan puhdistukseen tuotannosta tultuaan. Toinen kehitysideana vaunun tulisi seurata korkeuden muutosta analysointitiloissa. Tällöin voitaisiin varmistaa, että hydrauliikkaöljy liikkuu maston sylintereissä. Kyseisiä kehityskohteita kerettiin hieman visioimaan, mutta konkreettisia muutoksia ohjelmaan ei keretty tekemään.

Yllätyksenä työn aikana tuli myös, että käytettävissä oli vain yksi analysaattori, jossa oli relelähtö. Testaukselta löytyy kolme analysaattoria, jossa releoptiota ei ole, joten useampaa vihivaunua ei voida laittaa samaan aikaan automaattipuhdistukseen. Jos useamman vaunun tahtoo saada samaan aikaan puhdistukseen, tulisi muihin analysaattoreihin kehitellä jonkinlainen kommunikaatioyhteys. Vaihtoehtona on myös hankkia uusia releillä olevia analysaattoreita, mutta se ei välttämättä ole kustannustehokkain ratkaisu.

Kehitysehdotukset ovat mielestäni helposti toteuttavissa. Arvojen ja muiden muuttujien optimoiminen ei vaadi kuin enemmän puhdistuskertoja ja niiden seuraamista. Isommalla otannalla pystytään selvittämään, mitkä arvot soveltuvat parhaiten nopeaan läpimenoaikaan.

Lähteet

1 Rocla. MAIN PAGE. Verkkoaineisto. <https://rocla-agv.com/>. Luettu 8.12.2022.

2 Rocla. PALLET MOVER ATX. Verkkoaineisto. <https://rocla-agv.com/agv-solution/automated-guided-vehicles/pallet-mover-atx/>. Luettu 8.12.2022.

3 Rocla. COUNTERBALANCE AWT. Verkkoaineisto. <https://rocla-agv.com/agv-solution/automated-guided-vehicles/counterbalance-awt/>. Luettu 8.12.2022.

4 Rocla. REACH TRUCK ART. Verkkoaineisto. <https://rocla-agv.com/agv-solution/automated-guided-vehicles/reach-truck-art/>. Luettu 9.12.2022.

5 Kollmorgen. CVC700 – Vehicle Controller. Verkkoaineisto. <https://ndcsolutions.com/cvc700/>. Luettu 23.12.2022.

6 Parker. Inline Particle Monitor – iCountPD. Verkkoaineisto. <https://ph.parker.com/us/en/icountpd-online-particle-detector>. Luettu 23.12.2022.

7 Infoteam Software. PLC programming systems. Verkkoaineisto. <https://infoteam.de/en/our-know-how/plc-programming-systems>. Luettu 19.1.2023.

8 Koivula, Mikko. 2019. Opinnäytetyö. Metropolia Ammattikorkeakoulu. Theseus-tietokanta. <https://www.theseus.fi/bitstream/handle/10024/262829/Koivula_Mikko.pdf?sequence=2&isAllowed=y>. Luettu 19.1.2023.

Hydraulic Flush-ohjelman koodi

```
FUNCTION_BLOCK TESTINGMAST_FB_S
TYPE
END_TYPE

VAR_EXTERNAL
    Options : Options_T;
    Features : Features_T;
    MI : MI_T;
    MI_Settings : MI_Settings_T;
    Tractor_Settings : Tractor_Settings_T;
    Events : Events_T;
    Objects : Objects_T;
    NDC8 : NDC8_T;
    Tractor : Tractor_T;
END_VAR

VAR_INPUT
END_VAR

VAR_OUTPUT
END_VAR

VAR
    HeightReady_B : BOOL;
    HeightReady_TON : TON;
    AnalyzingReady_TON : TON;
    AnalyzingHeight_B : BOOL;
    AnalyzingDuplexReady_TON : TON;
    HydeErrorTimer_TON : TON;
    HydeErrorTimer_B : BOOL;
    DuplexTimer_B : BOOL;
    DuplexHeight_B : BOOL;
    HydeTimer_B : BOOL;
    Report_B : BOOL;
    OldState_STR : STRING;
    State_STR : STRING := 'RESET';
    DebugState_STR : STRING;
    SendEvent_FB : Event_ToBuffers_FB_S;
    ForceReset_B : BOOL;
END_VAR
(*********************************************************************************************
Module:     TestingMast_FB_S
*********************************************************************************************)
```

```
OldState_STR := State_STR;
ForceReset_B := false;

if MI.Drive.State.Tr.Moving_B
or Features.UserControl.IO.Input.FastButton_B
or Features.UserControl.IO.Input.SlowButton_B
or MI.Mast.State.Tr.Error_B
or not Features.Lift.State.Installed_B then
    ForceReset_B := true;
    State_STR := 'RESET';
end_if;

HeightReady_B := Features.Lift.State.Standstill_B and
           ABS(Features.Lift.Measure.Height_INT - Features.Lift.Command.Testing.Height_INT) < 50
           and not Features.Lift.State.Active_B
           and not StateChanged_B ;

AnalyzingReady_TON(IN := HydeTimer_B, PT:= T#10m);
AnalyzingDuplexReady_TON(IN := DuplexTimer_B, PT:= T#10m);
HydeErrorTimer_TON(IN := HydeErrorTimer_B, PT := T#240m);
WarmUpTimer_TON(IN := WarmUpTimer_TON_IN, PT := Objects.Option.TestingMast.Settings.Leak-
ageTest.WarmUpTimeSec_TIMEs);



(* STATE = RESET *)
if State_STR = 'RESET' then
    if Options.TestingMast.State.MastAdjustActive_B or Options.TestingMast.State.LeakageTestAc-
    tive_B or Options.TestingMast.State.HydraulicFlushActive_B then
        SendEvent_FB(EventCode_DINT := Events.Options.TestingMast.TestModeEnded,
                    Parameter1_DINT := 0,
                    Parameter2_DINT := 0);
    end_if;

    Features.Lift.Command.Testing.Control_UT := NoControl;
    Options.TestingMast.State.Idle_B := true;
    Options.TestingMast.State.MastAdjustActive_B := false;
    Options.TestingMast.State.LeakageTestActive_B := false;
    Options.TestingMast.State.HydraulicFlushActive_B := false;
    Objects.Option.TestingMast.Runtime.HF_LoopCounter_INT := 0;
    Report_B := true;
    AnalyzingHeight_B := false;
    Objects.Option.TestingMast.Runtime.HF_SwayOff_B := false;
    Objects.Option.TestingMast.Runtime.HF_LampSwitch_B := false;
    HydeTimer_B := false;
    DuplexTimer_B := false;
    Objects.Option.TestingMast.Runtime.HF_AnalyzingCount_INT := 0;
    Objects.Option.TestingMast.Runtime.HF_AnalyzingDuplexCount_INT := 0;
```

```
    HydeErrorTimer_B := false;

(* Start and stop buttons are defined. Values can be changed in VAD. *)
    Options.TestingMast.Command.StartHydraulicFlush_B := Objects.Option.TestingMast.Settings.Hy-
    draulicFlush.Start_B;
    Options.TestingMast.Command.StopHydraulicFlush_B := Objects.Option.TestingMast.Settings.Hy-
    draulicFlush.Stop_B;

(* Conditional statements to start subroutines *)
    if Options.TestingMast.Command.StartMastAdjust_B then
        State_STR := 'WAITING_ADJUST_COMMAND';
    elsif Options.TestingMast.Command.StartLeakageTest_B then
        State_STR := 'WARMUP_HIGH';
    elsif Options.TestingMast.Command.StartHydraulicFlush_B
    and MI.UserControl.State.Tr.Auto_B
    and not ForceReset_B then
        SendEvent_FB(EventCode_DINT := Events.Options.TestingMast.HydraulicFlushStart,
                    Parameter1_DINT := 0,
                    Parameter2_DINT := 0);
        State_STR := 'Hyde_Mast_Low';
    end_if;

(* STATE = Hyde_Mast_Low, The mast moves to MinHeight position *)
elsif State_STR = 'Hyde_Mast_Low' then

    Options.TestingMast.State.Idle_B := false;
    Options.TestingMast.State.MastAdjustActive_B := false;
    Options.TestingMast.State.LeakageTestActive_B := false;
    Options.TestingMast.State.HydraulicFlushActive_B := true;
    Objects.Option.TestingMast.Runtime.HF_SwayOff_B := true;
    Features.Lift.Restrictors.TestingMast.StopMovements_B := false;
    HydeErrorTimer_B := true;
    HydeTimer_B := false;
    AnalyzingHeight_B := false;
    DuplexTimer_B := false;
    DuplexHeight_B := false;


    if Options.TestingMast.Command.StopHydraulicFlush_B or MI.UserControl.State.Tr.Manual_B then
        State_STR := 'RESET';

    elsif Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 70 or HydeErrorTimer_TON.Q
    then State_STR := 'Hyde_Error';
    elsif Tractor.EnergySystem.State.LowBattery_B = true then State_STR := 'Hyde_Error';
    else
        Features.Lift.Command.Testing.Control_UT := Automatic;
        if Objects.Option.TestingMast.Settings.HydraulicFlush.HeightLow_INT >= MI_Settings.Min
Height_INT then
```

```
            Features.Lift.Command.Testing.Height_INT := Objects.Option.TestingMast.Set-
tings.HydraulicFlush.HeightLow_INT;
        else
            Features.Lift.Command.Testing.Height_INT := MI_Settings.MinHeight_INT + 100;
        end_if;
        Features.Lift.Command.Testing.Speed_USINT := Objects.Option.TestingMast.Settings.Hydrau-
licFlush.MaxSpeedLift_USINT;

        if Features.Lift.Measure.Height_INT < Features.Lift.Command.Testing.Height_INT + 10 AND
Features.Lift.Measure.Height_INT > Features.Lift.Command.Testing.Height_INT - 10 then State_STR
:= 'Reach_Out';
        elsif Features.Lift.Measure.Height_INT = Features.Lift.Command.Testing.Height_INT then
State_STR := 'Reach_Out';
        end_if;
    end_if;

(* STATE = Reach_Out, Reach mast moves to out limit position *)
(* This state needs more testing *)

elsif State_STR = 'Reach_Out' then
    if Features.Reach.State.Installed_B = true then
      if Options.TestingMast.Command.StopHydraulicFlush_B or MI.UserControl.State.Tr.Manual_B
then State_STR := 'RESET';
      elsif Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 70 or HydeErrorTimer_TON.Q
then State_STR := 'Hyde_Error';
      elsif Tractor.EnergySystem.State.LowBattery_B = true then State_STR := 'Hyde_Error';
      else
            Features.Reach.Command.Testing.Control_UT := Automatic;
            if Objects.Features.Reach.Settings.EndTargetLimitOut_INT > Objects.Fea-
tures.Reach.Settings.LimitOut_INT then
                Features.Reach.Command.LoadHandling.Position_INT := Objects.Features.Reach.Set-
tings.LimitOut_INT;
            else Features.Reach.Command.LoadHandling.Position_INT := Objects.Features.Reach.Set-
tings.LimitOut_INT - 10;
            end_if;

        Features.Reach.Command.LoadHandling.MaxSpeed_INT := MIN(Features.Reach.Command.LoadHan-
dling.MaxSpeed_INT, Objects.Features.Reach.Settings.HighSpeed_INT);

            if Objects.Features.Reach.Runtime.Position_INT < Features.Reach.Command.LoadHan-
dling.Position_INT + 3 AND Objects.Features.Reach.Runtime.Position_INT > Features.Reach.Com-
mand.LoadHandling.Position_INT - 3 then State_STR := 'Reach_In';
            elsif Objects.Features.Reach.Runtime.Position_INT = Features.Reach.Command.LoadHan-
dling.Position_INT then State_STR := 'Reach_In';
            end_if;
        end_if;

    else State_STR := 'Reach_In';
```

```
    end_if;

(* STATE = Reach_In, Reach mast moves to in limit position *)
(* This state needs more testing *)

elsif State_STR = 'Reach_In' then

    if Features.Reach.State.Installed_B = true then
        if Options.TestingMast.Command.StopHydraulicFlush_B or MI.UserControl.State.Tr.Manual_B
then State_STR := 'RESET';
        elsif Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 70 or HydeErrorTi-
mer_TON.Q then State_STR := 'Hyde_Error';
        elsif Tractor.EnergySystem.State.LowBattery_B = true then State_STR := 'Hyde_Error';
        else
            if Objects.Features.Reach.Settings.EndTargetLimitIn_INT < Objects.Fea-
tures.Reach.Settings.LimitIn_INT then
                Features.Reach.Command.LoadHandling.Position_INT := Objects.Features.Reach.Set-
tings.LimitIn_INT;
            else
                Features.Reach.Command.LoadHandling.Position_INT := Objects.Features.Reach.Set-
tings.LimitIn_INT + 10;
            end_if;

        Features.Reach.Command.LoadHandling.MaxSpeed_INT := MIN(Features.Reach.Command.LoadHan-
dling.MaxSpeed_INT, Objects.Features.Reach.Settings.HighSpeed_INT);

            if Objects.Features.Reach.Runtime.Position_INT < Features.Reach.Command.LoadHan-
dling.Position_INT + 3 AND Objects.Features.Reach.Runtime.Position_INT > Features.Reach.Com-
mand.LoadHandling.Position_INT - 3 then State_STR := 'Sideshift_Right';
            elsif Objects.Features.Reach.Runtime.Position_INT = Features.Reach.Command.LoadHan-
dling.Position_INT then State_STR := 'Sideshift_Right';
            end_if;
        end_if;

(* STATE = Sideshift_Right, Sideshift moves to right limit position *)
(* This state needs more testing *)

    else State_STR := 'Sideshift_Right';
    end_if;

elsif State_STR = 'Sideshift_Right' then

    if Features.Sideshift.State.Installed_B = true then
        if Options.TestingMast.Command.StopHydraulicFlush_B or MI.UserControl.State.Tr.Manual_B
then State_STR := 'RESET';
        elsif Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 70 then State_STR :=
'Hyde_Error';
        else
```

```
            Features.Sideshift.Command.Testing.Control_UT := Automatic;
            if Objects.Features.Sideshift.Settings.LimitRight_INT < Objects.Fea-
tures.Sideshift.Settings.EndTargetLimitRight_INT then
                Features.Sideshift.Command.LoadHandling.Position_INT := Objects.Fea-
tures.Sideshift.Settings.LimitRight_INT;
            else
                Features.Sideshift.Command.LoadHandling.Position_INT := Objects.Fea-
tures.Sideshift.Settings.LimitRight_INT - 5;
            end_if;

        Features.Sideshift.Command.LoadHandling.MaxSpeed_INT := MIN(Features.Sideshift.Com-
mand.LoadHandling.MaxSpeed_INT, Objects.Features.Sideshift.Settings.HighSpeed_INT);

            if Objects.Features.Sideshift.Runtime.Position_INT < Features.Sideshift.Com-
mand.LoadHandling.Position_INT + 3 AND Objects.Features.Reach.Runtime.Position_INT > Fea-
tures.Sideshift.Command.LoadHandling.Position_INT - 3 then State_STR :=
'Sideshift_Left';
            elsif Objects.Features.Sideshift.Runtime.Position_INT = Features.Sideshift.Com-
mand.LoadHandling.Position_INT then State_STR := 'Sideshift_Left';
            end_if;
        end_if;

    else State_STR := 'Sideshift_Left';
    end_if;

(* STATE = Sideshift_Left, Sideshift moves to left limit position *)
(* This state needs more testing *)

elsif State_STR = 'Sideshift_Left' then

    if Features.Sideshift.State.Installed_B = true then
        if Options.TestingMast.Command.StopHydraulicFlush_B or MI.UserControl.State.Tr.Manual_B
then State_STR := 'RESET';
        elsif Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 70 then State_STR :=
'Hyde_Error';
        else
            if Objects.Features.Sideshift.Settings.LimitLeft_INT > Objects.Fea-
tures.Sideshift.Settings.EndTargetLimitLeft_INT then
                Features.Sideshift.Command.LoadHandling.Position_INT := Objects.Fea-
tures.Sideshift.Settings.LimitLeft_INT;
            else
                Features.Sideshift.Command.LoadHandling.Position_INT := Objects.Fea-
tures.Sideshift.Settings.LimitLeft_INT + 5;
            end_if;

        Features.Sideshift.Command.LoadHandling.MaxSpeed_INT := MIN(Features.Sideshift.Com-
mand.LoadHandling.MaxSpeed_INT, Objects.Features.Sideshift.Settings.HighSpeed_INT);
```

```
            if Objects.Features.Sideshift.Runtime.Position_INT < Features.Sideshift.Com-
mand.LoadHandling.Position_INT + 3 AND Objects.Features.Reach.Runtime.Position_INT > Fea-
tures.Sideshift.Command.LoadHandling.Position_INT - 3 then State_STR :=
'Hyde_Mast_High';
            elsif Objects.Features.Sideshift.Runtime.Position_INT = Features.Sideshift.Com-
mand.LoadHandling.Position_INT then State_STR := 'Hyde_Mast_High';
            end_if;
        end_if;

    else State_STR := 'Hyde_Mast_High';
    end_if;
```

(* STATE = Hyde_Mast_High, The mast moves to MaxHeight position *)

```
elsif State_STR = 'Hyde_Mast_High' then
    if Options.TestingMast.Command.StopHydraulicFlush_B or MI.UserControl.State.Tr.Manual_B then
        State_STR := 'RESET';

    elsif Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 70 or HydeErrorTimer_TON.Q
then State_STR := 'Hyde_Error';
    elsif Tractor.EnergySystem.State.LowBattery_B = true then State_STR := 'Hyde_Error';
    else
        if Objects.Option.TestingMast.Settings.HydraulicFlush.HeightHigh_INT <= MI_Set-
tings.MaxHeight_INT then
            Features.Lift.Command.Testing.Height_INT := Objects.Option.TestingMast.Settings.Hy-
draulicFlush.HeightHigh_INT;
        else
            Features.Lift.Command.Testing.Height_INT := MI_Settings.MaxHeight_INT - 100;
        end_if;
        Features.Lift.Command.Testing.Speed_USINT := Objects.Option.TestingMast.Settings.Hydrau-
licFlush.MaxSpeedLift_USINT;

        if Features.Lift.Measure.Height_INT < Features.Lift.Command.Testing.Height_INT + 10 AND
Features.Lift.Measure.Height_INT > Features.Lift.Command.Testing.Height_INT - 10 then
            Objects.Option.TestingMast.Runtime.HF_LoopCounter_INT := Objects.Option.Testing-
Mast.Runtime.HF_LoopCounter_INT + 1;

            if Objects.Option.TestingMast.Runtime.HF_LoopCounter_INT >= 3 then State_STR := 'An-
alyzing';
            elsif Features.Lift.Measure.Height_INT < Features.Lift.Command.Testing.Height_INT +
10 AND Features.Lift.Measure.Height_INT > Features.Lift.Command.Testing.Height_INT - 10 then
State_STR := 'Hyde_Mast_Low';
            elsif Features.Lift.Measure.Height_INT = Features.Lift.Command.Testing.Height_INT
then State_STR := 'Hyde_Mast_Low';
            end_if;
        end_if;
    end_if;
```

(* STATE = Analyzing, The 10 minute timer starts. The analyzer analyzes the cleanliness of the hydraulic oil in the maximum position of the mast.*)

```
elsif State_STR = 'Analyzing'
then
    AnalyzingHeight_B := true;
    if Options.TestingMast.Command.StopHydraulicFlush_B or MI.UserControl.State.Tr.Manual_B
then
        State_STR := 'RESET';

    elsif Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 70 or HydeErrorTimer_TON.Q
then State_STR := 'Hyde_Error';

    elsif Tractor.EnergySystem.State.LowBattery_B = true then State_STR := 'Hyde_Error';


    elsif AnalyzingHeight_B = true then

        HydeTimer_B := true;
        Features.Lift.Restrictors.TestingMast.StopMovements_B := true; (* Make sure mast does
not accept new commands *)

        if Objects.Option.TestingMast.Runtime.HF_AWT_B = true then
            if Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 600 then
            Objects.Option.TestingMast.Runtime.HF_AnalyzingCount_INT := 0;
            Objects.Option.TestingMast.Runtime.HF_AnalyzingDuplexCount_INT := 0;
            HydeTimer_B := false;
            State_STR := 'Hyde_Mast_Low';
            end_if;
        elsif Objects.Option.TestingMast.Runtime.HF_ATX_B = true then
            if Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 110 then
            Objects.Option.TestingMast.Runtime.HF_AnalyzingCount_INT := 0;
            Objects.Option.TestingMast.Runtime.HF_AnalyzingDuplexCount_INT := 0;
            HydeTimer_B := false;
            State_STR := 'Hyde_Mast_Low';
            end_if;
        elsif Objects.Option.TestingMast.Runtime.HF_ART_B = true then
            if Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 600 then
            Objects.Option.TestingMast.Runtime.HF_AnalyzingCount_INT := 0;
            Objects.Option.TestingMast.Runtime.HF_AnalyzingDuplexCount_INT := 0;
            HydeTimer_B := false;
            State_STR := 'Hyde_Mast_Low';
            end_if;
        end_if;

        if AnalyzingReady_TON.Q then
            Objects.Option.TestingMast.Runtime.HF_AnalyzingCount_INT := Objects.Option.Testing-
Mast.Runtime.HF_AnalyzingCount_INT + 1;
```

```
            if  Objects.Option.TestingMast.Runtime.HF_ATX_B = true AND Objects.Option.Testing-
Mast.Runtime.HF_PressureInput_INT >= 150 then State_STR := 'Duplex';
            elsif Objects.Option.TestingMast.Runtime.HF_ART_B = true AND Objects.Option.Testing-
Mast.Runtime.HF_PressureInput_INT >= 800 then State_STR := 'Duplex';
            elsif Objects.Option.TestingMast.Runtime.HF_AWT_B = true AND Objects.Option.Testing-
Mast.Runtime.HF_PressureInput_INT >= 900 then State_STR := 'Duplex';
            else State_STR := 'Hyde_Mast_Low';
            end_if;
        end_if;
    end_if;
```

(* STATE = Duplex, mast moves to duplex changing position. *)

```
elsif State_STR = 'Duplex' then

    Features.Lift.Restrictors.TestingMast.StopMovements_B := false;
    HydeTimer_B := false;

    if Options.TestingMast.Command.StopHydraulicFlush_B or MI.UserControl.State.Tr.Manual_B then
        State_STR := 'RESET';
    elsif Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 70 or HydeErrorTimer_TON.Q
then State_STR := 'Hyde_Error';
    elsif Tractor.EnergySystem.State.LowBattery_B = true then State_STR := 'Hyde_Error';
    else
        if Tractor_Settings.Mast.DuplexMastHeight_INT > 5000 then
            State_STR := 'AnalyzingDuplex';
        else
            Features.Lift.Command.Testing.Height_INT := Tractor_Settings.Mast.Du-
plexMastHeight_INT;
        end_if;
        Features.Lift.Command.Testing.Speed_USINT := Objects.Option.TestingMast.Settings.Hydrau-
licFlush.MaxSpeedLift_USINT;

        if Features.Lift.Measure.Height_INT < Features.Lift.Command.Testing.Height_INT + 10 AND
Features.Lift.Measure.Height_INT > Features.Lift.Command.Testing.Height_INT - 10 then State_STR
:= 'AnalyzingDuplex';
        elsif Features.Lift.Measure.Height_INT = Features.Lift.Command.Testing.Height_INT then
State_STR := 'AnalyzingDuplex';
        end_if;
    end_if;
```

(* STATE = AnalyzingDuplex, The 10 minute timer starts. The analyzer analyzes the cleanliness of the hydraulic oil in duplex mast postion.*)

```
elsif State_STR = 'AnalyzingDuplex'
then
```

```
    if Tractor_Settings.Mast.DuplexMastHeight_INT > 5000 then
            Objects.Option.TestingMast.Runtime.HF_AnalyzingDuplexCount_INT := 3;
            State_STR := 'LoopCheck';
    else

        DuplexHeight_B := true;

        if Options.TestingMast.Command.StopHydraulicFlush_B or MI.UserControl.State.Tr.Manual_B
then
        State_STR := 'RESET';

        elsif Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 70 or HydeErrorTi-
mer_TON.Q then State_STR := 'Hyde_Error';

        elsif Tractor.EnergySystem.State.LowBattery_B = true then State_STR := 'Hyde_Error';

         Features.Lift.Restrictors.TestingMast.StopMovements_B := true;

            (*AnalyzingDuplexReady_TON(IN := DuplexTimer_B, PT:= T#10m);*)

            if Objects.Option.TestingMast.Runtime.HF_AWT_B = true then
                if Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 600 then
                Objects.Option.TestingMast.Runtime.HF_AnalyzingCount_INT := 0;
                Objects.Option.TestingMast.Runtime.HF_AnalyzingDuplexCount_INT := 0;
                DuplexTimer_B := false;
                State_STR := 'Hyde_Mast_Low';
                end_if;
            elsif Objects.Option.TestingMast.Runtime.HF_ATX_B = true then
                if Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 110 then
                Objects.Option.TestingMast.Runtime.HF_AnalyzingCount_INT := 0;
                Objects.Option.TestingMast.Runtime.HF_AnalyzingDuplexCount_INT := 0;
                DuplexTimer_B := false;
                State_STR := 'Hyde_Mast_Low';
                end_if;
            elsif Objects.Option.TestingMast.Runtime.HF_ART_B = true then
                if Objects.Option.TestingMast.Runtime.HF_PressureInput_INT < 600 then
                Objects.Option.TestingMast.Runtime.HF_AnalyzingCount_INT := 0;
                Objects.Option.TestingMast.Runtime.HF_AnalyzingDuplexCount_INT := 0;
                DuplexTimer_B := false;
                State_STR := 'Hyde_Mast_Low';
                end_if;
            end_if;
            if AnalyzingDuplexReady_TON.Q then
                Objects.Option.TestingMast.Runtime.HF_AnalyzingDuplexCount_INT := Objects.Op-
tion.TestingMast.Runtime.HF_AnalyzingDuplexCount_INT + 1;
                if  Objects.Option.TestingMast.Runtime.HF_ATX_B = true AND Objects.Option.Test-
ingMast.Runtime.HF_PressureInput_INT >= 150 then State_STR := 'LoopCheck';
```

```
                elsif Objects.Option.TestingMast.Runtime.HF_ART_B = true AND Objects.Op-
tion.TestingMast.Runtime.HF_PressureInput_INT >= 800 then State_STR := 'LoopCheck';
                elsif Objects.Option.TestingMast.Runtime.HF_AWT_B = true AND Objects.Op-
tion.TestingMast.Runtime.HF_PressureInput_INT >= 900 then State_STR := 'LoopCheck';
                else State_STR := 'LoopCheck';
                end_if;
            end_if;
        end_if;
    end_if;
```

(* STATE = LoopCheck, Checks the amount of clean analyzes.*)

```
elsif State_STR = 'LoopCheck' then

    if Objects.Option.TestingMast.Runtime.HF_AnalyzingCount_INT >= 3 AND Objects.Option.Testing-
Mast.Runtime.HF_AnalyzingDuplexCount_INT >= 3 then
        State_STR := 'Hyde_Report';

    else State_STR := 'Hyde_Mast_Low';
    end_if;
```

(* STATE = Hyde_Report, sends event to the screen that the program is ready.*)

```
elsif State_STR = 'Hyde_Report' then

    if Report_B then
        if Objects.Option.TestingMast.Runtime.HF_PressureInput_INT > 0 then
            Temp_DINT := INT_TO_DINT(Objects.Option.TestingMast.Runtime.HF_PressureInput_INT);
            SendEvent_FB(EventCode_DINT := Events.Options.TestingMast.HydraulicFlushDone,
                        Parameter1_DINT := Temp_DINT,
                        Parameter2_DINT := 0);
        end_if;
        Objects.Option.TestingMast.Runtime.HF_LampSwitch_B := true;
        HydeTimer_B := false;
        DuplexTimer_B := false;
        Report_B := false;
    end_if;

    if Options.TestingMast.Command.StopHydraulicFlush_B or MI.UserControl.State.Tr.Manual_B then
        State_STR := 'RESET';
    end_if;
```

(* STATE = Hyde_Error, sends event to the screen that the program is in error state.*)

```
elsif State_STR = 'Hyde_Error' then
```

```
    if HydeErrorTimer_TON.Q then
        Temp_DINT := INT_TO_DINT(Objects.Option.TestingMast.Runtime.HF_AnalyzingCount_INT);
            SendEvent_FB(EventCode_DINT := Events.Options.TestingMast.HydraulicFlushTimeError,
                        Parameter1_DINT := Temp_DINT,
                        Parameter2_DINT := 0);
            Objects.Option.TestingMast.Runtime.HF_LampSwitch_B := true;
            HydeTimer_B := false;
            Report_B := false;
    elsif Report_B then
            Temp_DINT := INT_TO_DINT(Objects.Option.TestingMast.Runtime.HF_PressureInput_INT);
            SendEvent_FB(EventCode_DINT := Events.Options.TestingMast.HydraulicFlushError,
                        Parameter1_DINT := Temp_DINT,
                        Parameter2_DINT := 0);
            Objects.Option.TestingMast.Runtime.HF_LampSwitch_B := true;
            HydeTimer_B := false;
            Report_B := false;

    elsif Tractor.EnergySystem.State.LowBattery_B = true then
        Temp_DINT := INT_TO_DINT(Objects.Option.TestingMast.Runtime.HF_PressureInput_INT);
            SendEvent_FB(EventCode_DINT := Events.Options.TestingMast.HydraulicFlushBatteryLow,
                        Parameter1_DINT := Temp_DINT,
                        Parameter2_DINT := 0);
            Objects.Option.TestingMast.Runtime.HF_LampSwitch_B := true;
            HydeTimer_B := false;
            Report_B := false;

    end_if;

    if Options.TestingMast.Command.StopHydraulicFlush_B or MI.UserControl.State.Tr.Manual_B then
        State_STR := 'RESET';
    end_if;
end_if;

Features.Drive.Command.TestingMast.Stop_B := Options.TestingMast.State.MastAdjustActive_B or Op-
tions.TestingMast.State.LeakageTestActive_B or Options.TestingMast.State.HydraulicFlushActive_B;

Objects.Option.TestingMast.Runtime.State_STR := State_STR;
Objects.Option.TestingMast.Runtime.DriveStop_B := Features.Drive.Command.TestingMast.Stop_B;
END_FUNCTION_BLOCK
```