LAB University of Applied Sciences

Ennakoiva kunnonvalvonta ja käyttöliittymän luominen CODESYS-ohjelmointiympäristössä

LAB-ammattikorkeakoulu
Insinööri (AMK), Tieto- ja Viestintätekniikka
2022
Tommi Töyrylä

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Töyrylä, Tommi	Opinnäytetyö, AMK	2022
	Sivumäärä	
	49	

Työn nimi

Ennakoiva kunnonvalvonta ja käyttöliittymän luominen CODESYS-ohjelmointiympäristössä

Tutkinto

Insinööri (AMK)

Toimeksiantajan nimi, titteli ja organisaatio

Erno Mustonen, System Sales Manager, SEW-EURODRIVE

Tiivistelmä

Työn tarkoitus oli tutkia ennakoivaa kunnonvalvontaa ja online kunnonvalvontaa Sew Eurodrive:n DUV40A värähtelyn monitorointijärjestelmän avulla, sekä luoda järjestelmälle erillinen graafinen käyttöliittymä.

Työssä esitellään kunnossapitoa, ja kunnonvalvontaa. Näiden avulla käydään läpi erityisesti teollisuudessa olevia PLC-koneita, niiden toimintaa ja logiikkaohjelmointia.

Projektin toimintaa tutkitaan erilaisten käytännön testien avulla. Projektissa luodaan toimistotiloissa toimiva prototyyppi, jonka toimintaa tutkitaan. Prototyypissä on pieni sähkövaihde, sekä kolme anturia mittaamassa sen värähtelyä.

Työstä saatava mahdollinen hyöty on yksinkertainen ja helposti luettava käyttöliittymä usealle monitorointijärjestelmälle, joiden avulla monitoroitavien laitteiden kunnonvalvonta helpottuu. Monitoroitujen laitteiden datan analyysin avulla voidaan mahdollisesti jopa ennakoida laitteiden tulevia rikkoja.

Asiasanat

Ennakoiva kunnonvalvonta, PLC, CODESYS

Abstract

Author(s)	Type of Publication	Published
Töyrylä, Tommi	Thesis, UAS	2022
	Number of Pages	
	49	

Title of Publication

Predictive maintenance and creation of user interface in CODESYS environment

Name of Degree

Engineer (UAS)

Name, title and organization of the client

Erno Mustonen, System Sales Manager, SEW-EURODRIVE

Abstract

The purpose of this work is to study predictive maintenance and online condition monitoring utilizing SEW EURODRIVE's DUV40A vibration monitoring systems, and to create a separate graphical user interface for the system.

The work studies maintenance and condition monitoring. With the help of these subjects, the work also explores what PLCs are and how their logics can be programmed.

The subjects of the project are studied with the help of various practical tests. A prototype is built at the office space, and its operation will be studied. The prototype contains a small electrical gear and three DUV40A sensors to measure the gears vibration.

The potential benefit of this project is a simple and easy-to-read graphical interface for multiple monitoring systems, which makes it easier to monitor the condition of the multiple devices at once. Data analysis of the devices can potentially be utilized in predictive maintenance.

Keywords

Predictive maintenance, PLC, CODESYS

Sisällys

Liitteet

1 Johdanto

Ennakoivan kunnonvalvonnan tavoitteena on pyrkiä huoltamaan laitteistot ennen kuin ne hajoavat. Yritykselle on katastrofi, mikäli laitteet hajoavat yllättäen ja uuden laitteen saamiseksi tilalle meneekin kuukausi, jonka aikana tuotanto seisoo paikallaan, mutta kulut juoksevat. On kuitenkin kallista ja työlästä vahtia, ja tarkastella koneiden kuntoa jatkuvasti. On myös epärealistista olettaa, että määräaikaishuolloissa vaihdettaisiin koneen kaikki osat kokonaan uusiksi. Toimivien osien vaihtaminen on resurssien tuhlaamista. Tämän lisäksi mahdolliset asennusvirheet huollon yhteydessä voivat vaikuttaa negatiivisesti laitteen toimintaan. Joissain tapauksissa määräaikaishuoltoja voidaan tehdä enemmän kuin ennakoimattomia huoltoja, täten nostaen kuluja. Tämän takia projektissa tutkittu automatisoitu jatkuva kunnonvalvonta on erittäin hyvä ratkaisu mille tahansa yritykselle.

Projektin toimeksiantajana toimii Sew Eurodrive. Projektissa käytetään pääsääntöisesti Sew Eurodrive:n tuotteita. Sew Eurodrive valmistaa vaihdemoottoreita, teollisuusvaihteita, taajuusmuuttajia, servokäyttöjä sekä hajautettua käyttöautomaatiota. Sew Eurodrive on Saksassa, Bruchsalissa vuonna 1931 perustettu yritys. Nykyään se on maailmanlaajuinen yritys, jolla on 19 000 työntekijää yli 430 toimipaikassa 52 eri maassa. Yrityksen liikevaihto oli 3,31 miljardia euroa vuonna 2021. Yritys laajentui Suomeen vuonna 1975. Yrityksellä on Suomessa pääkonttori, kokoonpanotehdas ja huoltokeskus, jotka sijaitsevat Hollolassa. Näiden lisäksi yrityksellä on aluekonttoreita ympäri Suomea. (Sew Eurodrive.)

Työn tavoitteena on luoda järjestelmä, joka hyödyntää SEW DUV40-kunnonvalvonta-antureita ennakoivassa kunnonvalvonnassa sekä Online kunnonvalvonnassa. Järjestelmään kuuluu myös erillinen visuaalinen käyttöliittymä, sekä SEW:in ohjelmoitava logiikka. Ohjelmoitavan logiikan rooli projektissa on vastaanotta antureiden keräämä data ja näyttää se käyttöliittymässä, käyttäen CODESYS- ohjelmointiympäristöä sekä CODESYS Visualization –ohjelmaa. Järjestelmässä on myös projektia varten keinotekoisesti heikennetty vaihdemoottori, joka tuottaa huomattavan määrän värinää. Tätä värinää käytetään järjestelmän testaamisessa.

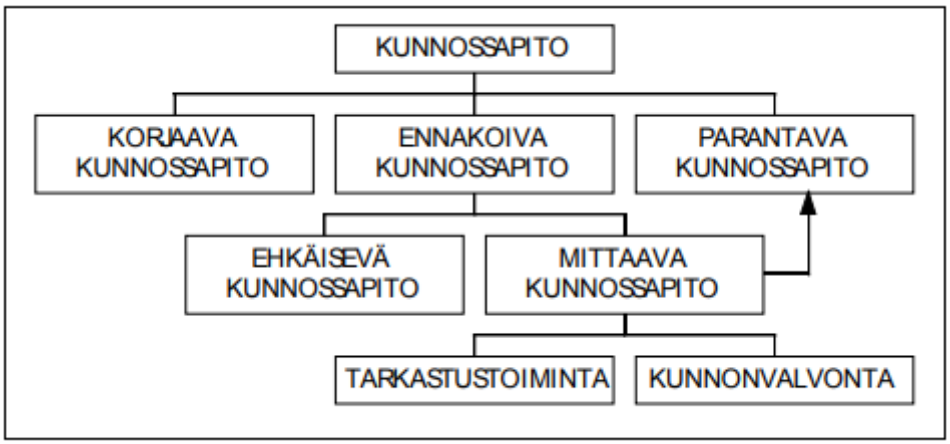
Järjestelmän luomisen lisäksi työssä on tutkittu, miten kyseinen järjestelmä toimisi käytännössä testausympäristön avulla. Työtä varten aloitettiin testausympäristön luominen, tavoitteena aloittaa testit lähitulevaisuudessa. Testausympäristössä on sähkömoottori sekä vaihde pyörimässä liian pienellä öljymäärällä. Tämän on tarkoitus rikkoa vaihde hitaasti, samalla, kun antureiden mittaustuloksia tutkitaan. Testauksen tavoitteena on tutkia sitä, miten data muuttuu koneiden hajotessa ja kuinka hyvin hajoamisen voisi havainnoida tai jopa ennustaa etukäteen.

Opinnäytetyön toisessa luvussa käydään läpi projektin teoriaa, miten projektin eri osat toimivat, ja mitä ne tekevät. Kolmannessa luvussa siirrytään käytäntöön, missä luodaan ja kasataan projekti kokonaisuudessaan. Neljännessä luvussa suunnitellaan ja aloitetaan tulevaa kenttätestiä. Kenttätestissä suunnitelmana on siirtää toimistossa kasattu projekti testiin, missä projektin toimivuutta testataan ajamalla oikeaa vaihdemoottoria niin pitkään, että se alkaa hajoaa.

2 Kunnossapito ja ohjelmoitava logiikka

2.1 Kunnonvalvonta

Kunnossapidon voi lajitella kahteen päälajiin, suunniteltuihin ja suunnittelemattomiin toimenpiteisiin. Suunniteltua kunnossapitoa on ehkäisevä kunnossapito, kunnostaminen sekä parantava kunnossapito. Suunnittelematonta kunnossapitoa on erilaisten häiriöiden korjaamiset, sitten kun ne ilmenevät. Suunnittelematon kunnossapito on käytetyin kunnossapidon muoto. Ehkäisevää kunnossapitoa ovat erilaiset suunnitellut toimenpiteet, kuten säännöllinen öljynvaihto. Ehkäisevä kunnossapito voi olla jaksotettua tai laitteiston kuntoon liittyvää, esimerkiksi fyysisestä kulumasta riippuvaa. (Helle, 2004.) Kuva 1 havainnollistaa eri lajeja.



Kuva 1. Kunnossapidon luokat (ABB, 2000)

Kunnonvalvonta on kunnossapidon alaluokka. Se sisältää muun muassa datan keräämistä, monitorointia ja etävalvontaa. Nämä kaikki ovat suhteellisen uusia käsitteitä kunnonvalvonnassa ja kunnossapidossa. Kyseiset alaluokat ovat vasta viimeisten vuosikymmenien aikana kehittyneet tarpeeksi pitkälle, että niiden toteuttamine on realistista. (Helle, 2004.)

Ennakoiva kunnonvalvonta on kunnonvalvonnan alalajike missä tavoitteena on pyrkiä ennakoivilla toimenpiteillä estämään laitteiston hajoaminen. Tämän avulla voidaan säästää resursseja sekä välttyä yllättäviltä tuotannon seisokeilta. Ideaalitapauksessa mikään ei hajoa, vaan kaikki laitteistot huollettaisiin ennen kuin hajoaminen tapahtuisi. (Sandberg-Diment, 1984.)

Ennakoiva kunnossapito on samankaltaista kuin suunniteltu kunnossapito, mutta sen sijaan, että huoltoja tehtäisiin esimerkiksi kerran kuukaudessa, huollot toteutettaisiin, kun

huolloille huomattaisiin tarvetta. Kalenterin ja kellon sijaan käytettäisiin tämän hetken kuntoa mittarina huollon tarpeelle. Reaaliaikaisen kunnon selvittäminen on kuitenkin haastavaa. Tämän takia laitteet tarvitsevat jatkuvaa kunnonvalvontaa. (O&M Best Practices Guide, 2022.)

Moottorien ja vaihteiden kulumista on vaikea ennakoida. Mikäli viat kyetään erottamaan silmämääräisesti tai korvalla ilman minkäänlaisia työkaluja, on vahinkoa jo tapahtunut. Erilaisilla ajoitetuilla mittauksilla tai tarkastuksilla voidaan pystyä ennakoimaan mahdollinen hajoaminen, mutta nämä vaativat resursseja. Mittausten ottaminen ja tulosten analyysi vaativat työkaluja ja osaavaa henkilöstöä. Tämän takia mittausprosessin automatisointi ohjelmiston ja antureiden avulla on tehokasta. Se myös helpottaisi vaikeasti havaittavien ongelmien havaitsemista, jotka voisivat jäädä huollosta vastaavalta henkilöltä inhimillisen virheen takia huomioimatta. (ABB:n TTT-käsikirja, 2000.)

Ennakoiva kunnossapito on kuitenkin kalliimpaa kuin suunniteltu kunnossapito, johtuen korkeammasta hankintahinnasta sekä laitteiston käyttämisestä vastaavan henkilöstön koulutuksesta tai uuden henkilöstön palkkauksesta johtuvista kuluista. Ennakoivan kunnossapidon lisääminen nykyisiin järjestelmiin on myös joissain tilanteissa haastavaa. Pitkällä aikavälillä ennakoiva kunnossapito alkaa kuitenkin maksamaan itseään takaisin, koska yllättäviä seisokkeja tulee vähemmän, sekä turhia huoltoja toteutetaan vähemmän. (O&M Best Practices Guide, 2022.)

Toimintavarmuuskeskeinen kunnossapito on kunnossapitomallien yhdistelmä. Kunnossapitomenetelmässä arvioidaan, mikä on paras tapa huoltaa mikäkin kohde. Halvemmat ja helposti vaihdettavat laitteet voidaan ajaa loppuun, samalla korkean prioriteetin laitteisiin käytetään ennakoivaa kunnossapitoa. Kyseinen kunnossapitomalli sisältää muiden kunnossapitomallien hyödyt ja haitat, mutta lisäksi vaatii vielä enemmän mallin käyttäjältä. Käyttäjän pitää osata arvioida ja suunnitella kunnonvalvonta tarkasti ja oikein saadakseen sen hyödyt. (O&M Best Practices Guide, 2022.)

Projektin yhteydessä haastateltiin Sew Eurodrive:ltä kunnonvalvonnan ammattilaista (Liite 1). Häneltä kysyttiin kunnonvalvontaan liittyviä kysymyksiä, jotka ovat osittain yleispäteviä kunnonvalvontaan, ja osittain liittyvät työhön itsessään. Tämän avulla haluttiin kokemuksia ja näkökulmaa projektiin henkilöltä, joka työskentelee alan parissa.

2.2 PLC

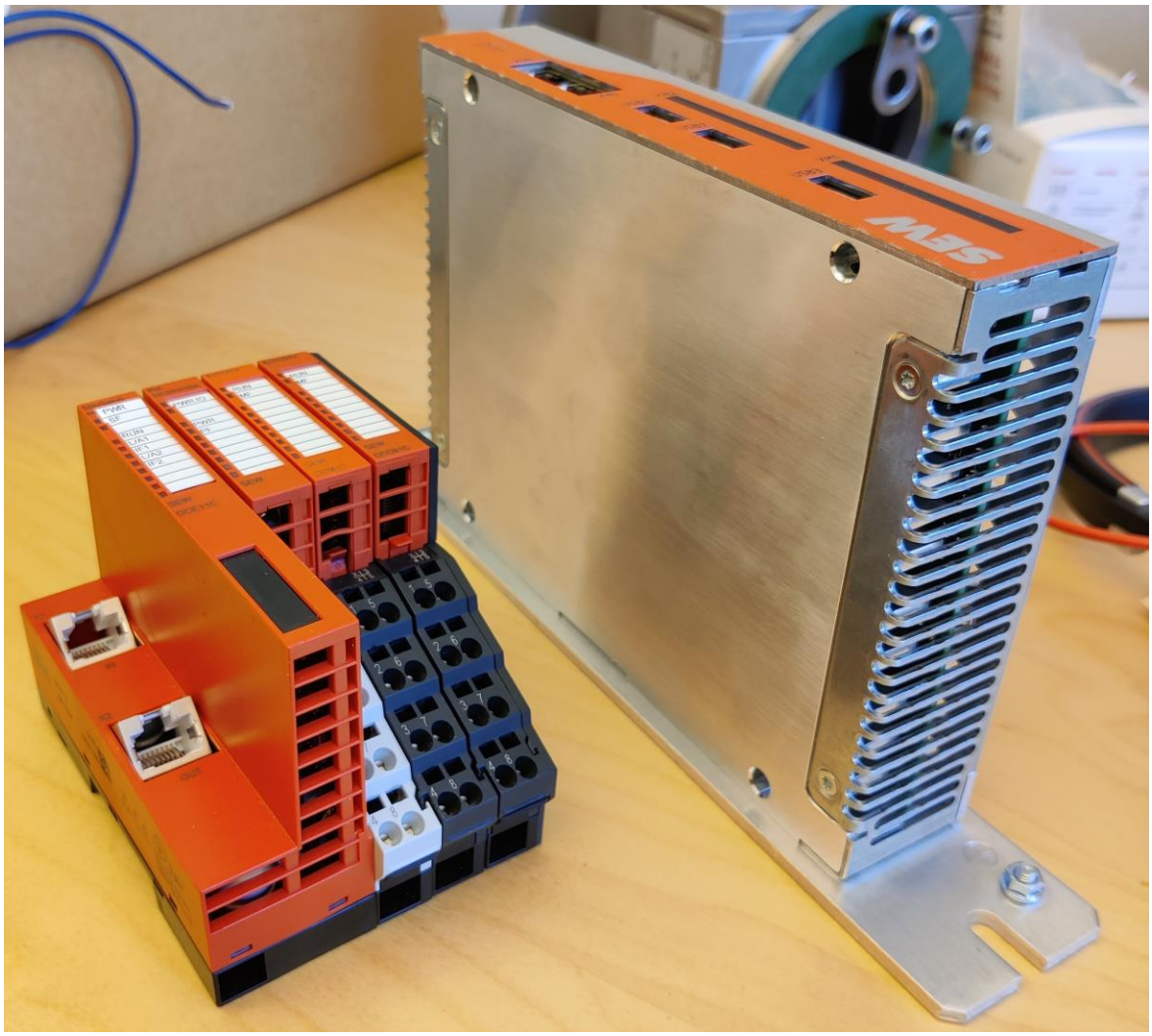
PLC lyhenne tulee sanoista ’Programmable logic controller’, eli ohjelmoitava logiikka. PLC on teollinen mikroprosessoripohjainen tietokone, jota voi ohjelmoida. Tietokoneessa on erilaisia tulo- ja lähtöportteja, joihin voi liittää ulkoisia laitteita, kuten antureita tai

toimilaitteita. Näiden liitäntöjen avulla voidaan teollisuudessa hallinnoida eri laitteita ja antureita. Ensimmäiset PLC:t otettiin käyttöön vuonna 1968. Ne kehitettiin korvaamaan monimutkaiset releillä toimivat ohjausjärjestelmät. Näissä ensimmäisissä PLC laitteistoissa oli sisään- ja ulostulot liitynnät ulkoisia kytkentöjä varten. Jokainen sisääntulomoduuli pystyi lukemaan 16 eri tuloa, ja ulostulomoduulit kykenivät lukemaan 16 eri tuloa. (Parr, 1998.)

Yksinkertaisimmillaan PLC:t ovat tietokoneita, jotka suorittavat jatkuvasti samaa tehtävää. Ne voivat esimerkiksi lukea yhteydessä olevien laitteiden tilatietoja, suorittaa ohjelmistonsa laitteilta saatujen arvojen avulla ja lopulta kirjoittavat ulostuloja, jota toiset laitteet voivat hyödyntää. Tämä prosessi toistuu jatkuvasti ja tiheään tahtiin. Yhden kierroksen kesto mitataan jopa millisekunneissa.

Tietotekniikan kehittyessä myös PLC:t ovat kehittyneet. Niistä on tullut tehokkaampia, kestävämpiä sekä monipuolisempaan käyttöön sopivia parempien yhteyksien avulla. PLC voi olla esimerkiksi yhteydessä linjaston antureihin, ja samalla kerätä antureiden datan ja lähettää ne tietopankkiin, sekä kontrolloida linjaston hälytyksiä. Tämän lisäksi data voidaan esittää graafiseen käyttöliittymään luomaan interaktiivista visualisointia, jonka avulla käyttäjä voi kontrolloida linjastoa etänä. Iso tekijä on myös erilaiset I/O-moduulit, ja niiden kehitys. Nykyään PLC:hen voi liittää I/O-moduulin, joka kykenee lukemaan erilaisia laitteita. Anturilta lähtee liitäntä I/O-moduuliin, joka taas liitetään moduulista PLC:hen. I/O-moduuleita voi myös muokata PLC:hen sopivaksi lisäämällä erilaisia laajennusyksiköitä moduuliin.

Alapuolella olevassa kuvassa (Kuva 2) on esitetty SEW MOVI-C CONTROLLER UHX65A progressive PLC, kuvassa oikealla, sekä MOVI-C I/O-System C, kuvassa vasemmalla. UHX65A PLC on tehokkaampi Sew-Eurodrive:n PLC. Se on tarkoitettu korkeaa suorituskykyä vaativaan teollisuuskäyttöön. Laite vaatii toimiakseen muistikortin, mutta siinä on myös ylimääräinen muistikortin paikka Windows 10 IoT -käyttöjärjestelmää varten.



Kuva 2. SEW UHX65 PLC ja SEW MOVI-PLC I/O-System C

2.3 Ulkoiset liitynnät

Tulot ja lähdöt, eli I/O, muodostavat ohjausjärjestelmien perustan. Teollisuudessa I/O on iso osa laitteistoa ja niiden toimivuutta. Esimerkki tulosta on anturi, joka mittaa PLC:lle dataa, kuten lämpötilaa. Lähtö voi olla esimerkiksi hälytysvalo, joka syttyy, kun tulon anturilta datan lukenut PLC huomaa, että lämpötila on liian korkealla.

PLC kykenee keskustelemaan eri laitteiden kanssa erilaisten kommunikointimoduulien avulla. Näiden avulla PLC voi luoda Peer to Peer (P2P) yhteyden laitteiden välille, jolloin laitteet voivat jakaa tietojansa keskenään sekä kommunikoida. Tuloportista tulee laitteelle dataa anturilta ja lähtöportista menee seuraavalle laitteelle komentoja. I/O-moduuleita on sekä analogisia että digitaalisia. Esimerkki tulomoduuliin kytkettävästä laitteesta on mittari, joka mittaa moottorin lämpötilaa, ja esimerkki lähtömoduuliin kytkettävästä laitteesta on

hälytysvalo, joka menee päälle, kun anturin antama data moottorin lämpötilasta ylittää PLC:n ohjelmassa asetetun rajan.

Kaikki laitteet eivät aina toimi mutkattomasti keskenään. Eri valmistajien valmistamissa laitteissa voi olla pieniä eroja keskenään. Tämän takia laitteille on kehitetty erilaisia standardeja. Harvinaisemmille antureille tarvitaan joskus erikoisempia I/O-moduuleita toimiakseen. I/O-moduuleissa on aina eroja myös ominaisuuksiltaan, kuten liityntäjännite. Eri moduulit voivat myös soveltua eri käyttötarkoitukseen. On olemassa moduuleita, jotka toimivat digitaalisesti, jotkut analogisesti ja osa sisältää molempia. Erilaiset kommunikointimenetelmät voivat vaatia erilaisia topologioita, mitkä voivat vaikuttaa moduulin kytkentätapaan.

Aiemmassa luvussa kuvassa (Kuva 2) näkyy PLC, mutta sen vieressä näkyy myös I/O:n mahdollistava I/O-moduuli nimeltä SEW MOVI-PLC I/O-System C. Kyseiseen I/O-moduuliin on lisätty kaksi ylimääräistä I/O-kanavaa. Kyseisessä moduulissa on myös OCE11C Ethercat-väylä, jonka avulla I/O-moduulin saa Ethernet-kaapelilla liitettyä PLC:hen.

2.4 Kommunikointi ja kenttäväylät

Erilaiset kenttäväylät ja kommunikaatioprotokollat ovat tärkeitä PLC:n käytössä. Yhteyden luomisessa voidaan käyttää kenttäväylää. Fyysinen, kaapelin avulla toimiva yhteys on stabiili ja nopea. Teollisuudessa tästä on hyötyä, koska yhteyksien pitää olla toiminnallisia aina, kun laitteet ovat päällä. Kenttäväylät ovat yleinen ratkaisu tähän.

Ensimmäiset kenttäväylät yleistyivät 1980-luvulla. Aluksi markkinoilla oli monia erilaisia ratkaisuja useilta eri laitevalmistajilta ja laitteiden välinen toimivuus eri valmistajien välillä oli heikkoa. Ajan myötä standardit yleistyivät perustuen yleisemmin käytössä olleisiin laitteisiin. Useat nykyajan kommunikointistandardit ovat peräisin 1990-luvulta.

Kommunikointiprotokolla on standardi, joka määrittelee laitteiden välisen yhteyden ja tiedonsiirron. Kommunikointiprotokollia on käytössä myös teollisuusmaailman ulkopuolella lähes kaikessa modernissa tietoliikenteessä. Erilaiset kommunikointiprotokollat mahdollistavat useiden eri laitteiden keskenään toimimisen.

Kaksi yleistä kommunikointiprotokollaa ovat Profinet IRT sekä EtherCAT. Molemmat protokollat ovat teknisiltä spesifikaatioiltaan verrattavissa olevia kaapeleiden avulla toimivia kommunikointiprotokollia. Profinet IRT on osa Siemensin kehittämää Profinet protokolla kokonaisuutta. Protokolla kykenee asynkroniseen kommunikointiin, sekä useampaan erilaiseen topologiaan. EtherCAT on Beckhoffin kehittämä kommunikointiprotokolla.

Protokolla vaatii EtherCAT isäntäportin, sekä yhden tai useamman EtherCAT orjaportin. Data liikkuu isännältä orjille. Tämä rajoittaa mahdollisia topologioita. EtherCAT on tutkimuksen mukaan Profinet IRT:tä nopeampi, mikäli samassa laitteessa on merkittävä määrä laitteita kiinni. (Prytz, 2008.)

OPC tulee sanoista ”Open Platform Communications”. Sen ensimmäinen versio julkaistiin vuonna 1996 ja sen on kehittänyt OPC Foundation. OPC on useamman tiedonsiirtostandardin yhdistelmä, joita käytetään teollisuudessa. OPC standardin tarkoitus on mahdollistaa eri valmistajien laitteiden keskinäinen toimivuus, sekä Windows – laitteiden että teollisuuden laitteiston välinen toimivuus. OPC perustuu Microsoftin teknologioihin. Laite tarvitsee OPC – serverin, jonka jälkeen mikä tahansa OPC:ta tukeva laite voi kommunikoida kyseisen laitteen kanssa. Toisin kuin aiemmassa kappaleessa mainitut protokollat, OPC on suunniteltu toimimaan langattomasti lähiverkon välityksellä. Standardia kehitetään edelleen ja se on saanut useita päivityksiä. Uudemmat versiot toimivat vanhempien versioiden kanssa. OPC:ssa on myös eri spesifikaatioita, kuten OPC DA (OPC Data Access) ja OPC HDA (OPC Historical Data Access). Näistä OPC DA on yleisemmin käytetty spesifikaatio. OPC tunnetaan nykyään nimellä OPC Classic ja sen on korvannut OPC-UA. (OPC Foundation a.)

OPC-UA on OPC:n spesifikaatio, joka on riippumaton alustasta ja on kehitetty avoimen lähdekoodin periaattein. UA lyhenne tulee sanoista ’Unified Architecture’. Sen ensimmäinen versio julkaistiin vuonna 2008. Sen on kehittänyt OPC Foundation. OPC-UA sisältää kaikki edellisen OPC:n spesifikaatioiden ominaisuudet, mutta niiden lisäksi kehitystyötä on jatkettu pidemmälle. Alustariippumattomuus ja avoin lähdekoodi mahdollistavat sen, että millä tahansa laitteella on mahdollista käyttää OPC-UA:ta, mikä mahdollistaa useita uusia käyttötapoja. Myös standardin turvallisuuteen on panostettu enemmän ja tietoturvamahdollisuuksia on paljon enemmän kuin ennen, kuten datan salaus sekä eritasoisten käyttäjien luonti. (OPC Foundation a.)

OPC-UA palvelin mahdollistaa sen, että OPC-UA-protokollaa hyödyntävä asiakaskone voi lukea ja muokata sallittuja osuuksia laitteessa samalla täyttäen vaadittavat tietoturvastandardit. Kommunikointi ei ole lukittu yhteen tiettyyn väylään, vaan data voi liikkua niin Ethernet-kaapelin välillä kuin langattomasti tai satelliitin kautta. Protokollan käyttöön ei vaadita laitestandardia, vaan mikä tahansa laite, mille on tukea, voi hyödyntää protokollaa. (OPC Foundation b.)

2.5 PLC-logiikkaohjelmointi

PLC-tietokoneen logiikkaohjelmointi suunniteltiin alun perin helposti lähestyttäväksi. Tavoitteena oli, että ohjelmointi ei vaatisi erityistä tietoteknistä osaamista, vaan henkilö, joka ennen vastasi releillä toimivan ohjausjärjestelmän toiminnasta osaisi myös ohjelmoida PLC:t toimimaan halutulla tavalla.

PLC:n prosessori suorittaa kahta tehtävää. Ensimmäinen tehtävä on suorittaa laitteen käyttöjärjestelmää. Toinen tehtävä on suorittaa käyttäjän tekemää omaa sovellusta. Sovelluksen luomiseen käyttäjä tarvitsee erillisen tietokoneen sekä ohjelmiston. Ohjelmistoja on useita erilaisia eri yrityksiltä ja ne monesti riippuvat laitevalmistajasta. Esimerkiksi Siemensin PLC:den yhteydessä joutuu käyttämään Siemens TIA Portal-ohjelmointiympäristöä. Ohjelmistoissa voi olla pieniä eroja, miten ne toimivat, mutta jos ne tukevat samaa IEC 61131-3 standardia, erot eivät vaikuta PLC:n ohjelmointiin.

Ohjelmistot käyttävät nykyään IEC 61131-3 -standardia. Standardi on kolmas versio vuonna 1993 julkaistusta IEC 61131 -standardista. Standardi määrittelee PLC:ssä käytettävää ohjelmistoarkkitehtuuria sekä ohjelmointikieliä. Ohjelmointikieliä standardiin kuuluu viisi, joista kolme on graafisia ja kaksi tekstipohjaista. Standardiin määritellyt ohjelmointikielet ovat seuraavat:

- Structured Text (ST), eli rakenteinen teksti, on tekstipohjainen kieli.
- Instruction List (IL), eli käskylista, on tekstipohjainen kieli.
- Ladder Diagram (LD) eli relekaavio-ohjelmointi, on graafinen kieli.
- Function Block Diagram (FBD) eli toimilohkokaavio on graafinen kieli.
- Sequential Function Chart (SFC) eli vuokaavio-ohjelmointi on graafinen kieli.

Käyttäjä voi luoda omia ohjelmia erilaisilla ohjelmistoilla hyödyntäen edellä mainittuja ohjelmointikieliä. PLC:n ohjelmointiin on muitakin ohjelmointikieliä riippuen ohjelmointiympäristöstä sekä laitteista, mutta nämä viisi kieltä muodostavat IEC 61131-3 –standardin.

Kielistä rakenteinen teksti (ST) muistuttaa eniten perinteistä ohjelmointikieltä. Se on korkean tason tekstipohjainen kieli. Kielessä luodaan ensin ohjelmassa käytettävät muuttujat erillisessä ohjelmalohkossa.Tämän jälkeen luodaan ohjelma, jota PLC käy läpi. Kielen on sanottu muistuttavan erityisesti Pascal-ohjelmointikieltä (Nieke, 2022). Alla olevassa kuvassa (Kuva 3) on esimerkki koodista. Ohjelman yläosassa on ohjelman muuttujalista, jossa alustetaan ohjelman muuttujat, sekä määritellään ohjelman nimi. Esimerkkiohjelmassa on luotu kaksi Integer-muuttujaa. Ohjelman alaosassa muuttujille

annetaan ensin arvot, ja sitten vertaillaan niitä. Mikäli ensimmäinen muuttuja on pienempi kuin toinen, muutetaan ensimmäisen muuttujan arvo samaksi kuin toisen muuttujan arvo.

```
PROGRAM demoST
VAR
    iValue : INT;
    iCompare : INT;
END_VAR
```

```
iValue := 0;
iCompare := 1;

IF iValue < iCompare := TRUE
THEN
    iValue := iCompare;
END_IF
```

Kuva 3. Rakenteinen teksti esimerkkiohjelma

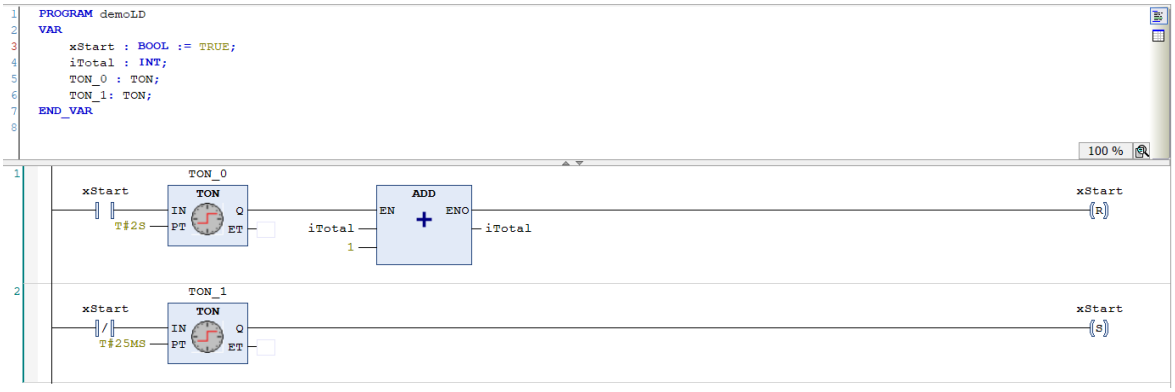
Käskylista (IL) on matalan taso ohjelmointikieli. Uusimmassa IEC 61131-3 standardissa kieli on vanhentunut. Kielen syntaksi koostuu yksittäisistä komennoista, jotka laite käy järjestyksessä läpi. Ensin ohjelmaa ohjeistetaan lataamaan haluttu muuttuja akkuun, jonka operaatiot kohdistuvat akun sisältöön. Kaikki annetut käskyt koskevat kyseistä muuttujaa, kunnes sen tilalle ladataan uusi muuttuja. Alla olevassa kuvassa (Kuva 4) on esimerkkiohjelma, jossa ensin määritellään kolme muuttujaa, ja kahdelle näistä annetaan alkuarvo. Alla olevassa ohjelman suorituksessa ensin ladataan akkuun muuttujan iValue arvo, sitten akkuun lisätään muuttujan iAdd arvo, ja tämän jälkeen akussa oleva yhteenaskettu tulos talletetaan muuttujaan iVar.

```
PROGRAM demoIL
VAR
    iValue : INT := 6;
    iAdd : INT := 12;
    iVar: INT;
END_VAR
```

```
LD      iValue
ADD     iAdd
ST      iVar
```

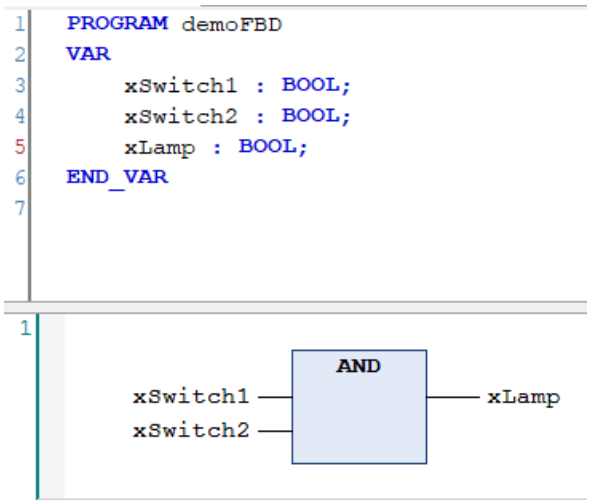
Kuva 4. Käskylista esimerkkiohjelma

Relekaavio-ohjelmointi (LD) on IEC 61131-3 standardin suosituin kieli, sijoittuen IEEE Spectrumilla sijalle 50 vuonna 2021 (Cass, Kulkarni & Guizzo, 2021). Se on graafinen kieli, joka muistuttaa ulkonäöltään sähkökaavioita. Kielessä vasemmalla olevasta jännitekiskosta jännite johtuu vasemmalta oikealle päätepisteenään käämi. Matkalla voi olla esimerkiksi ajastimia tai muuttujia. Alla olevassa kuvassa (Kuva 5) on esimerkki relekaavio-ohjelmasta. Ensin alustetaan muuttujat, ja samalla määritellään tietylle muuttujalle arvo. Muuttujien alapuolella luodaan Network, eli Virtapiiri, joka lähtee päälle, kun muuttuja xStart on tosi. Koska xStart alustetaan jo todeksi, lähtee ohjelma heti käyntiin. Kun xStart on tosi, TON_0-ajastin lähtee käyntiin. Kahden sekunnin kuluttua ohjelma lisää yhden yksikön lisää iTotal-muuttujaan. Tämän jälkeen ohjelma kääntää muuttujan xStart epätodeksi, joka taas käynnistää toisen ajastimen, jonka jälkeen muuttuja palaa takaisin todeksi ja ohjelma alkaa alusta. Tämä kaikki johtaa siihen, että ohjelma lisää laskuriin yhden arvon lisää aina kahden sekunnin välein, niin kauan kuin ohjelmaa suoritetaan.



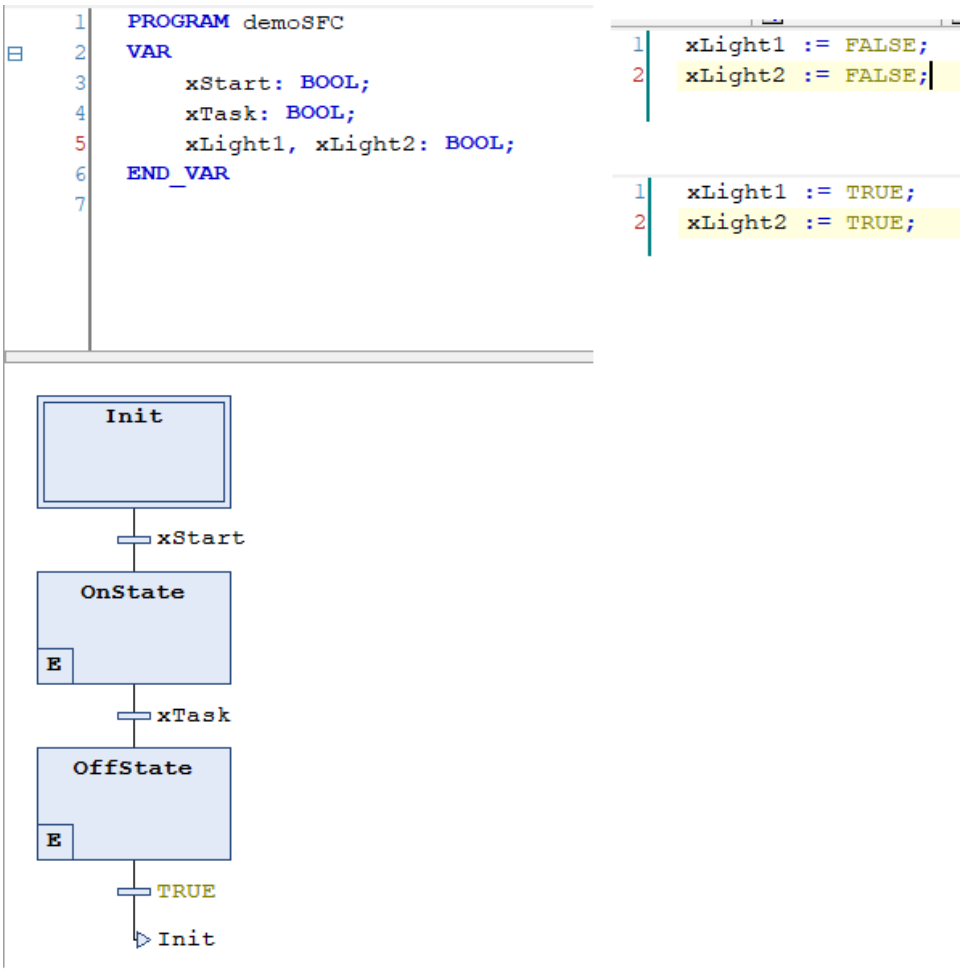
Kuva 5. Relekaavio-ohjelmointi esimerkkiohjelma

Toimilohkokaavio (FBD) on graafinen kieli, jossa luodaan lohko, jossa on sisääntulo sekä ulostulo. Ulostulo voi olla myös toisen lohkon sisääntulo. Data liikkuu lohkon sisällä vasemmalta oikealle. Lohkolla voi olla useampi eri päätepiste. Näitä lohkoja voi käyttää erikseen, yhdistää toisiinsa, tai jopa käyttää eri ohjelmissa. Ohjelmointikieli noudattaa perinteisiä ohjelmointikielten sääntöjä ja komentoja. Alla olevassa lyhyessä esimerkkiohjelmassa (Kuva 6) alustetaan kolme muuttujaa. Jos kaksi näistä on tosi, kolmas muuttuu myös todeksi. Tätä ohjelmaa voisi esimerkiksi käyttää jonkinlaisena hälytysohjelmana. Mikäli nämä kaksi asiaa toteutuvat samaan aikaan, hälytyslamppu menee päälle.



Kuva 6. Toimilohkokaavio esimerkkiohjelma

Vuokaavio-ohjelmointi (SFC) on viimeinen IEC 61131-3 -standardin kielistä. Se on graafinen ja muistuttaa ulkonäöltään perinteisiä vuokaavioita. Se etenee visuaalisesti alaspäin askeleittain, mutta samassa askeleessa voi olla horisontaalisesti useampi muuttuja. Askeleen komennot toteutetaan samanaikaisesti. SFC on kieli, joka kykenee rinnakkaislaskentaan. Alla olevassa kuvassa (Kuva 6) on esimerkkiohjelma tehty SFC-kielellä, hyödyntäen myös rakenteista tekstiä. Aluksi ohjelmassa on alustettu muuttujat. Tämän jälkeen ohjelma siirtyy ensimmäiseen lohkoon, jossa suoritetaan ensimmäinen Rakenteinen teksti -ohjelma. Tämä siirtymä riippuu ohjelman alussa alustetusta muuttujasta. Ohjelma muuttaa kaksi muuttujaa todeksi. Tämän jälkeen ohjelma odottaa, tuleeko muutosta ohjelman alussa alustetulle muuttujalle. Mikäli muuttuja muuttuu, siirtyy ohjelma toiseen osuuteen. Tässä ohjelma käy läpi toisen Rakenteinen teksti -ohjelman, jossa se vaihtaa samat muuttujat epätodeksi kuvan 6 oikean yläreunan mukaisesti.



Kuva 7. Vuokaavio-ohjelmointi esimerkkiohjelma

Kieliä voi käyttää keskenään sekaisin. Esimerkiksi Rakenteinen teksti -kieltä voi sisällyttää jokaisessa graafisessa kielessä suoritettavaksi mihin tahansa vaiheeseen, kuten aikaisemmassa SFC-ohjelmassa tehtiin (Kuva 7). Myös Rakenteinen teksti -kielessä voi esimerkiksi kutsua ja käyttää Toimilohkoja kesken ohjelman.

Ohjelmat luodaan ensin erillisessä ohjelmointiympäristössä. Tämän jälkeen ohjelmat ladataan PLC:n muistikortille, josta laite osaa käyttöjärjestelmän avulla suorittaa ohjelmoidut komennot. Lataaminen tapahtuu ohjelmointilaitteelta PLC:lle Ethernet-yhteydellä. Kun ohjelma on ladattu ja muistikortti asetettu PLC:hen, alkaa PLC suorittamaan ohjelmaa, kuten sitä on ohjeistettu.

3 Projektin kehitys ja kokoonpano

3.1 Projektin kasaus

Projektia varten täytyi luoda helposti testattava kohde. Kohteen piti kyetä luomaan värinää, mitä pystyttiin mittaamaan. Värinää hyödynnettiin antureiden toiminnan tutkimisessa ja testauksessa. Projekti vaatii myös ohjelmalogiikan luontia, millä saadaan luettua antureiden mittaamaa dataa. Tätä logiikkaa hyödyntämään projektiin kehitetään graafinen käyttöliittymä. Käyttöliittymä luo yhteyden logiikkaan, ja näyttää käyttäjälle antureiden mittaamia arvoja.

3.1.1 Toimiston testiprojekti

Projektissa käytettiin Sew Eurodrive:n MOVI-C CONTROLLER progressive UHX65 -PLC:tä. Se on yhdistettynä kahdeksanpaikkaiseen Ethernet-kytkimeen. Projektissa mittauskohteena toimii hajoamispisteessä oleva W10/DT56L4 vaihdemoottori. Moottori saa virtansa MOVITRAC LTE-B taajuusmuuttajasta. Vaihdemoottoria on mittaamassa kolme kappaletta SEW DUV40A – kunnossapitojärjestelmiä.

Projekti kasataan kahteen eri osaan. On erillinen ”räkki”, mihin on kiinnitetty taajuusmuuttaja, vaihdemoottori sekä vaihdemoottoria mittaavat SEW DUV40A–anturit. Toinen osa on asennuslevy, mihin on asennettu PLC, Ethernet-kytkin sekä sulakkeet. Myös aiemmin projektissa testikäytössä olleet I/O-moduulit siirtyivät uuteen testiin.

Projektiin on luotu sähkökaaviot (Liite 2), joista näkee, miten toimiston testiprojektin sähköt on suunniteltu. Kaavioista on peitetty asiakkaan tiedot, sekä Sew Eurodrive:lle tärkeät tiedot.

Ennen kuin projektin PLC:tä voidaan hyödyntää, tarvitsee se alustaa. Tämä vaatii muistikortinlukijan. PLC:n muistikortti avataan tietokoneella ja sinne asennetaan Sew Eurodrive:n luomat konfigurointitiedostot. Tiedostoista voidaan mahdollisesti muokata joitain asetuksia, kuten laitteen käyttämää IP-osoitetta. Kun tarvittavat muutokset on tehty, tallennetaan tiedosto ja siirretään muistikortti takaisin PLC:hen. Kun PLC:n käynnistyy, se tarkistaa konfigurointitiedostosta oikeat asetukset ja ottaa ne käyttöön.

3.1.2 SEW DUV40A

SEW DUV40A on värähtelyn monitorointijärjestelmä. Järjestelmässä on antureita, jotka kykenevät mittaamaan ja monitoroimaan värähtelyä sekä lämpötilaa. Järjestelmässä on myös erilaisia sisäänrakennettuja analysointijärjestelmiä, joita voi hyödyntää monitoroidun

datan jälkikäteen tehdyssä erillisessä analyysissa. Käyttäjä voi määrittää laitteelle erilaisia hälytysarvoja, jotka voivat käyttäjän halutessa vaikuttaa datan analyysiin hälytysmerkinnöillä. Käyttäjä voi myös asettaa erilaisia raja-arvoja, jotka määrittävät, milloin järjestelmä alkaa monitoroimaan dataa. Käyttäjä pystyy myös lisäämään monitoroitavia arvoja ylimääräisillä tuloilla. Nämä tulot voidaan integroida monitorointiin sekä analyysiin. Ne toimivat, kuten järjestelmän vakioarvot. Laitteen asetusten muokkaaminen onnistuu SmartWeb-sovelluksen avulla, tai SmartUtility-sovelluksella. Laitteen mukana tulee ilmainen SmartUtility Light -sovellus, mutta ylimääräinen ja maksullinen SmartUtility on myös saatavilla. Kyseisessä sovelluksessa on enemmän ominaisuuksia kuin Light-versiossa.

Laite on pienikokoinen sekä kestävä. Siinä on kaksi ulkoista näppäintä. RESET-näppäin kuittaa nykyiset hälytykset, mikäli sitä painaa kahden sekunnin ajan. Mikäli näppäintä painaa kymmenen sekunnin ajan, se käynnistää koko laitteen uudestaan. TEACH- näppäin asettaa laitteen Teach-tilaan, mikäli näppäintä pitää viiden sekunnin ajan pohjassa. Teach-tilassa DUV40A tarkkailee tulevia värähtelyitä ja määrittää niiden avulla ajan myötä hälytysten raja-arvot järjestelmälle itsenäisesti. Raja-arvoja voi kuitenkin käyttäjä muokata halutessaan jälkikäteen manuaalisesti.

Monitorointijärjestelmän lisääminen osaksi jo olemassa olevaa järjestelmää on mahdollista. Laitteessa on paikka ylimääräiselle tulolle, jota voi monitoroida. Monitoroitavia arvoja voi muokata, esimerkiksi mittatarkuuksien säätämisellä. Laite on oma kokonaisuus, joka toimii erillään, joten sen toiminta ei vaikuta muiden järjestelmien toimintaan.

Kunnossapitojärjestelmä asennetaan tarkkailtavan vaihteen tai vaihdemoottorin läheisyyteen kiinni analysoimaan värinää sekä lämpötiloja. Mikäli järjestelmä huomaa, että ennalta määrätyt rajat ylitetään, se lähettää hälytyksen. Kyseistä hälytystä voidaan hyödyntää käyttäjän haluamalla tavalla.

SEW DUV40A:ssa on 128MB:n verran vapaata muistia. Järjestelmä tallentaa kyseiseen muistiin automaattisesti mittaustuloksiaan. Tallennuksen tiheyttä voidaan muokata. Tiheämmät tallennusajat antavat enemmän tutkittavaa dataa, mutta täyttävät muistin nopeammin. Kun muisti tulee täyteen, alkaa laite tallentamaan vanhimman datan päälle uutta dataa. Datan voi ladata laitteelta SmartWebin avulla ja tarkemmin analysoida SmartUtility-sovelluksella. (Sew Eurodrive, 2020.)

3.2 Projektin logiikan ohjelmointi

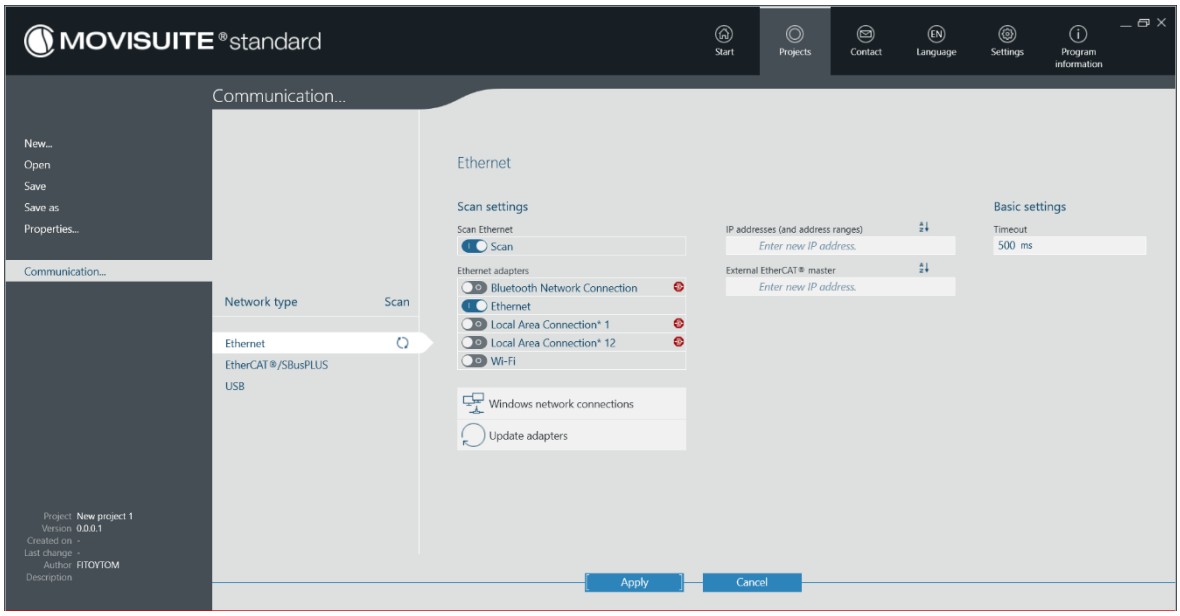
Projektiin luotiin logiikka, joka kerää DUV-antureiden datan yhteen paikkaan. Tämä vaatii ohjelmiston luomista PLC:hen. Ohjelmisto luo yhteydet DUV-antureihin, ja lukee niiden dataa. Data näytetään lopulta käyttäjälle käyttöliittymän kautta, joten dataa pitää muokata käyttöliittymään sopivaksi. Ohjelmistossa pitää ottaa huomioon myös mahdolliset ongelmat, kuten yhteyden katkeamiset antureihin.

3.2.1 Ohjelman alustus ja yhteydet

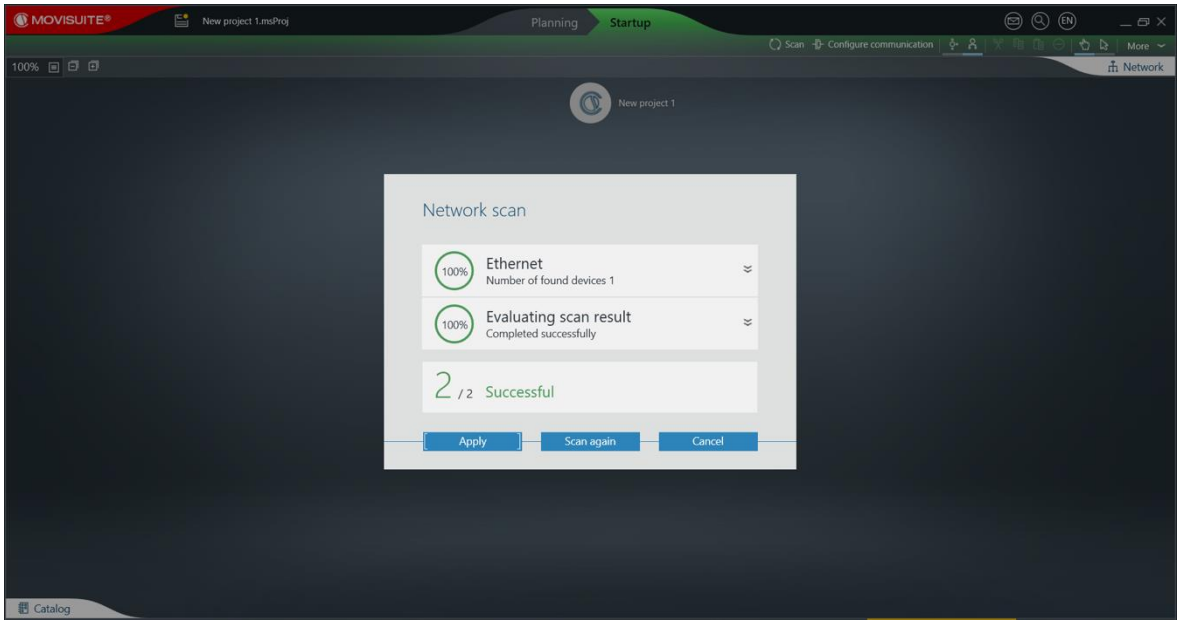
Projektin logiikan ohjelmointiin käytettiin Sew Eurodrive:n omaa MOVISUITE-ohjelmistoa. Kyseinen ohjelmisto helpottaa Sew Eurodrive:n laitteiden käyttöä mahdollistamalla erilaisia engineering-toimintoja, kuten laitteen asetusten ja osoitteiden muuttamista. Engineering-toimintojen lisäksi ohjelmistossa tulee mukana CODESYS ohjelmointiympäristö, sekä useita Sew Eurodrive:n tekemiä kirjastoja kyseiselle ohjelmointiympäristölle.

Kun projektin komponentit on kasattu ja testattu toimivaksi, sen ohjelmoitavaa logiikkaa voi alkaa ohjelmoimaan käyttäen MOVISUITE-ohjelmistoa. MOVISUITE:ssa luodaan uusi projekti. Heti uutta projektia luodessa MOVISUITE kysyy, haluatko luoda Planning-projektin vai Network scan -toiminnolla etsiä PLC:n. Planning-projektissa ohjelmisto luo virtuaalisen PLC:n, jota voi käyttää demonstraatioon ja projektin suunnitteluun. Etsimällä PLC:n verkosta MOVISUITE vaatii, että käyttäjä siirtää luomansa ohjelmiston PLC:lle ja suorittaa ohjelmaa oikealla PLC:llä, eikä vain virtuaalisoidulla PLC:llä.

MOVISUITE-ohjelmisto kykenee etsimään verkossa olevia PLC:tä Network scan -toiminnon avulla. Toiminto tarkastaa valitun verkon, ja mikäli kyseisestä verkosta löytyy sopivia PLC:itä, käyttäjä voi lisätä haluamansa mukaan projektiin. Prosessi näkyy kuvissa 8 ja 9.

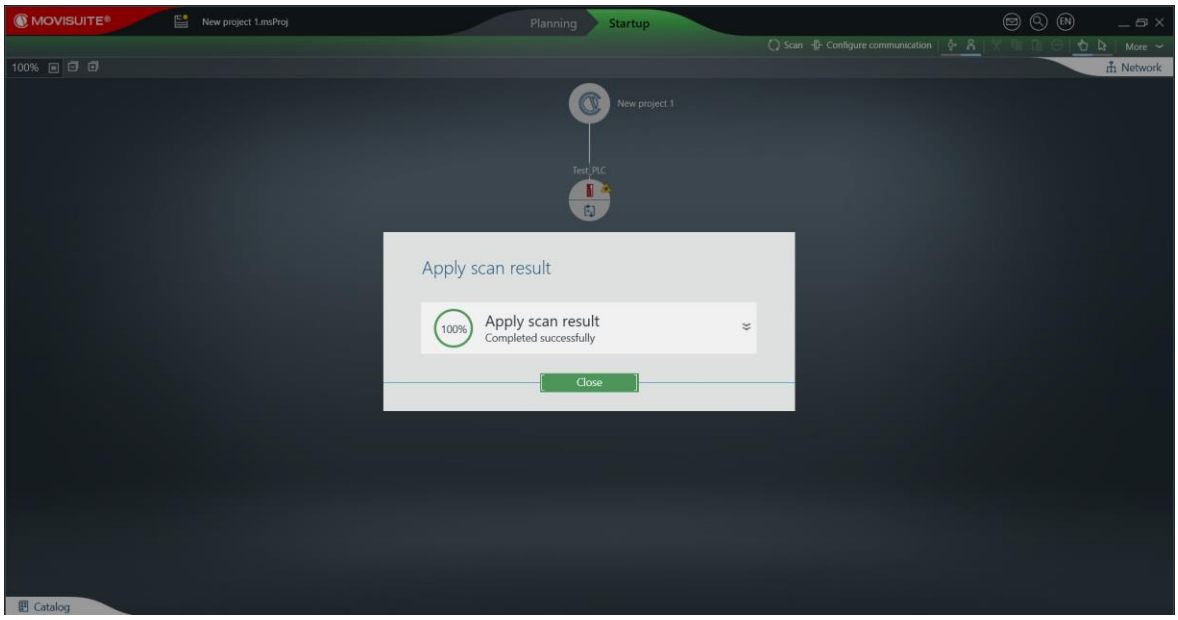


Kuva 8. Network Scan määrittely



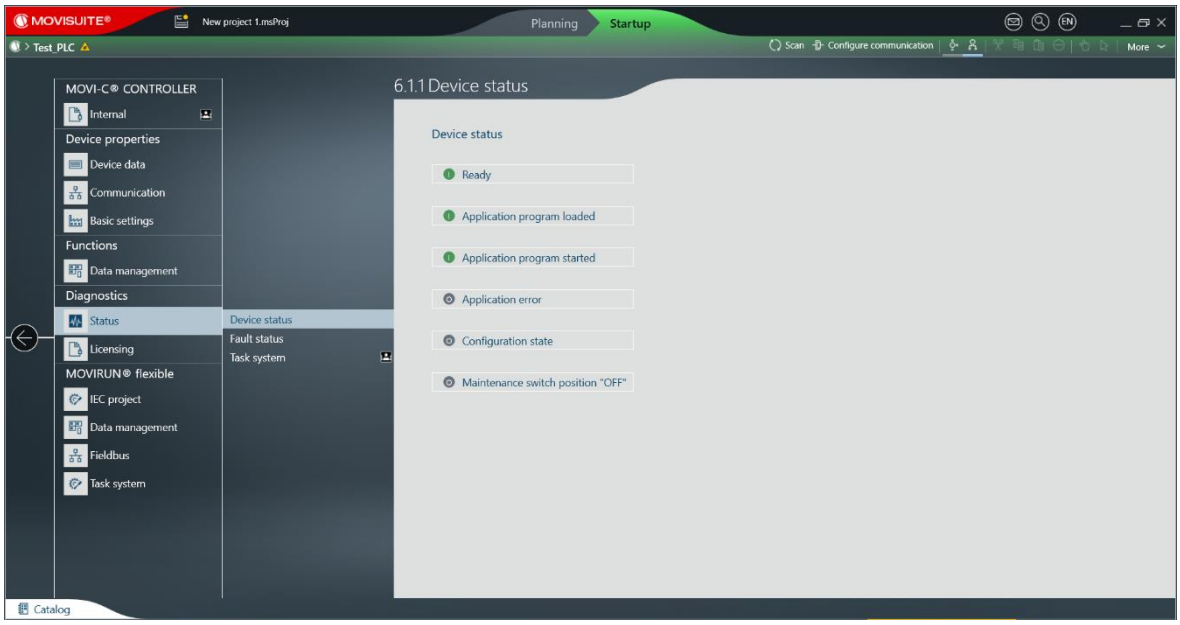
Kuva 9. Onnistunut skannaus

Lisäämällä löydetyt PLC:t projektiin, ilmestyvät ne projektin verkkonäkymään, kuten kuvassa 10 näkyy. Verkkonäkymästä näkee helposti kaikki projektin laitteet. Näkymästä ilmenee myös laitteiden nimet ja miten ne ovat yhteyksissä muihin projektin laitteisiin. Kyseinen ominaisuus helpottaa isompien projektien hallintaa.



Kuva 10. Laitteen lisäys onnistui

Valitsemalla käyttöliittymästä haluamansa PLC:n, pääsee käyttäjä tutkimaan kyseisen PLC:n tietoja, kuten esimerkiksi sarjanumerot ja firmwaren versiot. Valikoista voi konfiguroida PLC:tä esimerkiksi lisäämällä lisenssejä, sekä muuttaa PLC:n asetuksia, kuten kellonaikaa, tehtävien suoritussykliä ja PLC:n nimeä. Valikossa voi myös muuttaa kommunikointiasetuksia ja lisätä erilaisia kommunikointiväyliä. Esimerkkinä kuvassa 11 on näkyvillä projektin PLC:n Main Component -välilehti, mistä näkee tämänhetkisen PLC:n tilanteen. Laite on kuvassa toimintakykyinen, sinne on ladattu ohjelma ja se suorittaa ohjelmaa ilman virheitä.

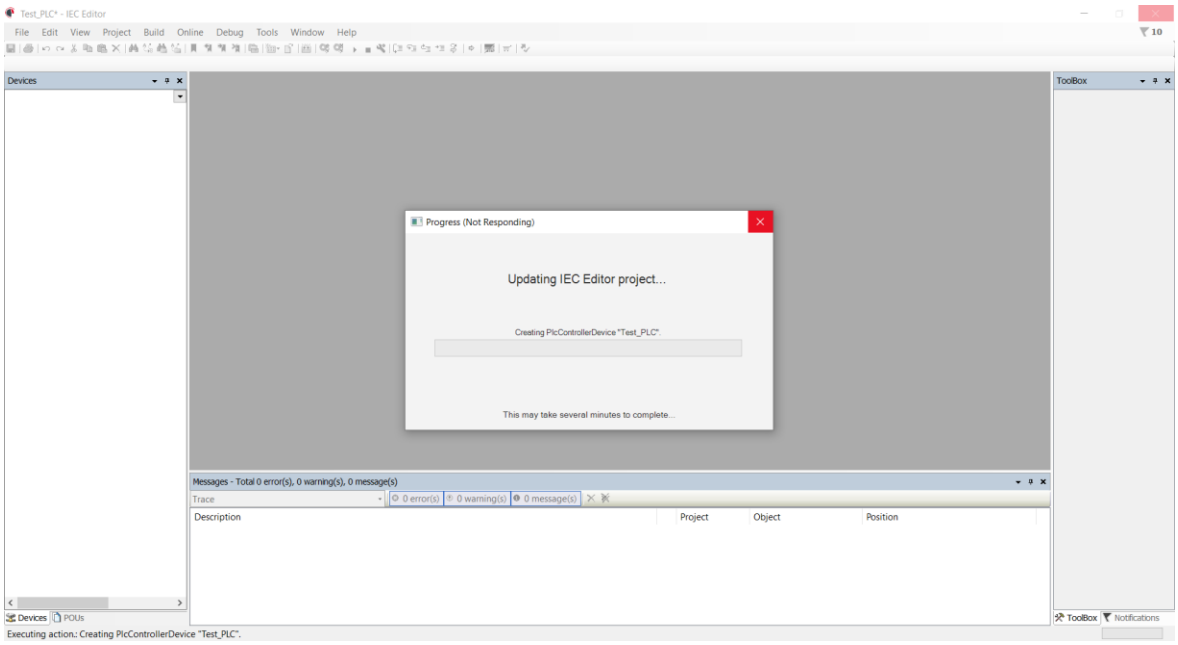


Kuva 11. MOVISUITE:n Device status -välilehti

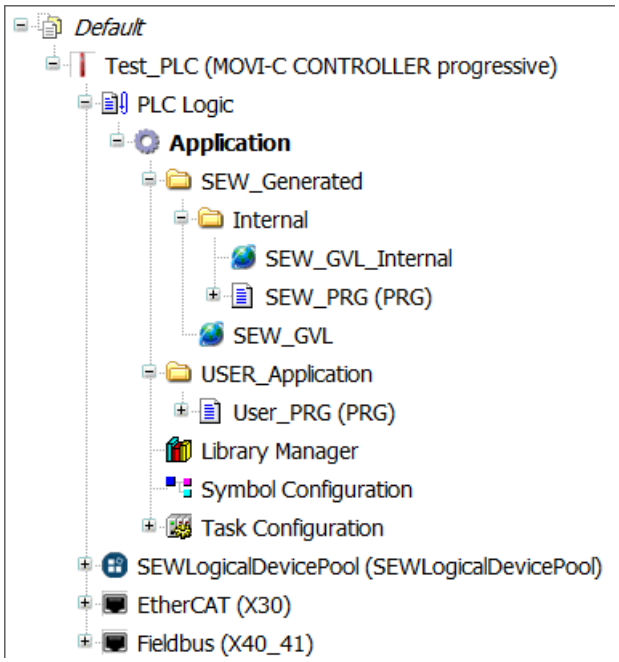
Sivupalkin alaosassa on IEC project -välilehti. Kyseisestä välilehdestä pääsee ohjelmoimaan PLC:hen IEC 61131-3 -standardin avulla. Valikossa on kolme vaihtoehtoa:

- Update IEC project
- Create new IEC project
- Open IEC editor.

Kyseessä on uusi projekti, joten sille luodaan uusi IEC projekti. Tämä tuo CODESYS:in ja SEW-EURODRIVE:n oletuskirjastot projektiin automaattisesti, luo tarvittavat alustukset sekä tyhjän ohjelman, mihin käyttäjä voi alkaa tekemään haluamansa ohjelmaa. Kuvassa 12 on kyseinen tapahtuma käynnissä ja kuvassa 13 on kyseisen prosessin lopputuloksena luotu kansiorakenne. Projektiin on alustettu kaikki vaadittavat ohjelmat ja kirjastot. Projektissa on myös SEW-EURODRIVE:n kansio, SEW_Generated. Tätä kyseistä kansiota ja sen sisältöä ei tule tavallisen CODESYS-projektin yhteydessä, vaan ainoastaan MOVISUITE:n avulla luoduissa projekteissa. Tässä projektissa SEW:in automaattisesti luoduista tiedostoista muokataan vain kuvassa 14 näkyvää SEW_GVL-tiedostoa. Tässä tiedostossa alustetaan globaalit muuttujat, eli muuttujat, joita voi mikä tahansa projektin ohjelmista käyttää vapaasti. Muut kansion ohjelmista alustavat ja käynnistävät erilaisia Logger-ohjelmia, jotka projektin mahdollisen kaatumisen tai muun ongelman yhteydessä kirjoittavat lyhyen raportin PLC:n muistikortille. Näihin ohjelmiin ei käyttäjän tarvitse tehdä muutoksia.



Kuva 12. Uuden projektin luonti

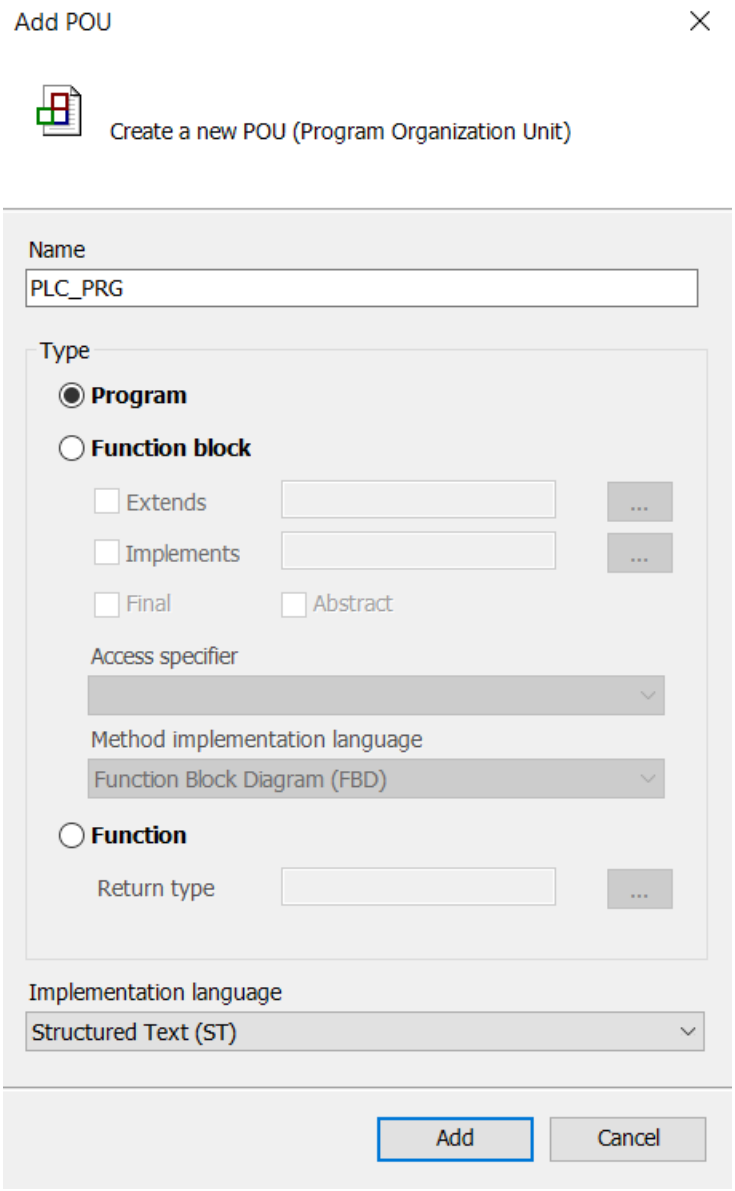


Kuva 13. Uuden projektin kansiorakenne

3.2.2 Antureiden käyttöönotto

Luodun projektin kansiorakenteesta puuttuu projektin pääohjelma. Pääohjelman tehtävänä on luoda ja ylläpitää yhteys DUV:eihin. Muut ohjelmat tulevat hyödyntämään pääohjelmassa luotua yhteyttä antureihin. Koska projekti tulee käyttämään useampaa ohjelmaa hyödyksi, luodaan pääohjelma juurikansioon ja apuohjelmat luodaan USER_Application-kansioon. Pääohjelman kieleksi tulee Rakenteinen teksti ja ohjelman

nimeksi PLC_PRG. Alla olevassa kuvassa (Kuva 14) näkyy pääohjelman luontiprosessi. Kaikki projektin ohjelmat luodaan samalla tavalla, ja samasta näkymästä, mutta eri asetuksilla.



Kuva 14. Pääohjelman luonti

Ohjelma tarvitsee toimiakseen projektiin kirjastoja. Kirjastot ovat kokoelmia, missä on valmista koodia, joiden avulla voidaan tehdä ohjelmia nopeammin ja tehokkaammin. Projektissa hyödynnetään SEW-EURODRIVE:n omia kirjastoja, joista osa on tätä projektia varten tehtyjä. Projekti myös vaatii ylimääräisiä kirjastoja näidenkin lisäksi. Kirjastoja voi ladata suoraan CODESYS Store-palvelusta ja manuaalisesti asentaa niitä CODESYS:in asennuskansioon. Kaikkia kirjastoja ei ole siirretty tai hyväksytty CODESYS Store-alustalle. Uudemmissa CODESYS-versioissa pystyy joitakin kirjastoja lataamaan sovelluksen kautta

suoraan Library Manager -työkalun avulla. Käyttäjä voi myös itse tehdä omia kirjastoja. Tässä projektissa on yksi räätälöity kirjasto, jonka avulla pystyy luomaan yhteyden OPC-UA:n avulla DUV-kunnossapitojärjestelmään ilman CODESYS:in omaa OPC-UA ominaisuutta.

Kun kirjasto on lisätty projektiin onnistuneesti, pääohjelmassa voi käyttää sen Toimilohkoja. Toimilohkot löytyvät SensorInterfaceBaseV2-nimisestä kansiosta, jonka kirjasto tuo mukanaan, kun se tuodaan projektiin. Tämä kansio löytyy SEW_SensorInterface-nimisen nimiavaruuden alta. Muuttujien alustamisessa voidaan helpottaa ohjelmointia jatkossa luomalla muuttuja nimeltä fbSensor, ja antamalla tälle muuttujalle arvoksi SEW_SensorInterface.SensorInterfaceBaseV2.

Lisäämällä kuvan 15 mukaisen koodin, saadaan ohjelman sisälle ohjelmalohko, joka alustaa vaadittavat muuttujat oikeiksi, sekä seuraa kaikkia DUV:in muuttujia. Koodissa oleva CASE-lohko vahtii diState-muuttujan arvoa. Kun arvo muuttuu ykköseksi, ohjelma suorittaa rivit 8–13 järjestyksessä. Rivillä 13 diStaten arvo muutetaan kahdeksi, jolloin se siirtyy seuraavaan osaan ohjelmaa. Arvoa kontrolloidaan erillisestä aliohjelmasta.

```
5   CASE diState OF          //connection to the DUV #1
6
7       1:
8       fbsensor.sUrl :='opc.tcp://192.168.10.101:4840';        //DUV #1
9       fbsensor(xEnable :=FALSE);
10      fbSensor.stSensorDataId := SEW_SensorInterface.SensorDataBaseId.gc_stEnglish;
11
12      fbsensor.xenable :=TRUE;
13      diState := 2;
14
15      2:
16      fbsensor();                          //keep the sensor running, reading data
17  END_CASE;
```

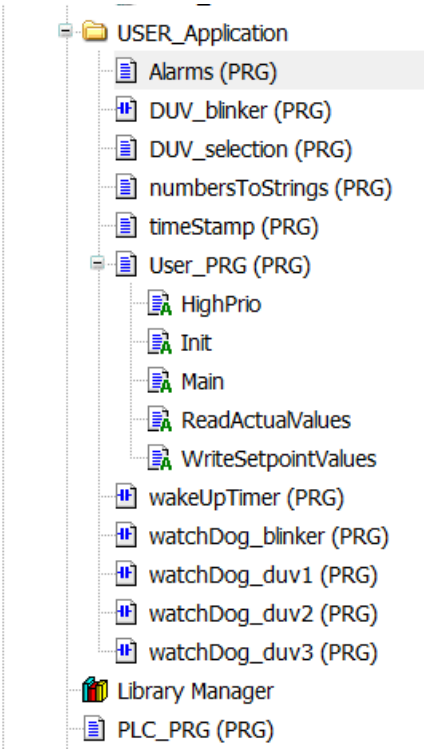
Kuva 15. Yhteyden alustus

Kuvan 15 ohjelmalohkoa voi käyttää uudelleen helposti muihin DUV-laitteisiin. Ohjelmalohkon voi kopioida ja muuttaa vaadittavat parametrit kuten DUV:in osoitteen, jolloin laitteeseen saa luotua yhteyden ohjelmassa. Projektissa on käytössä kolme eri anturia. Kaikki anturit on liitetty projektiin samanlaisella ohjelmalohkolla. Lyhyessä testissä ohjelmisto saatiin tukemaan useampaakin anturia, mutta projektissa käytetään kolmea.

3.2.3 Aliohjelmat

Projektissa on useita aliohjelmia pääohjelman lisäksi. Ohjelmat on kirjoitettu joko rakenteellisella tekstillä tai relekaavio-ohjelmoinnilla. Nämä ohjelmat tekevät ohjelmiston

käynnistymisestä sulavampaa sekä helpottavat ja mahdollistavat visuaalisen käyttöliittymän käyttöä. Esimerkiksi käyttöliittymän hälytykset ovat tässä projektissa luotu aliohjelmien avulla. Kuvassa 16 näkee aliohjelmien kansiorakenteen. Ohjelmat ovat yhden tason pääohjelman alapuolella.

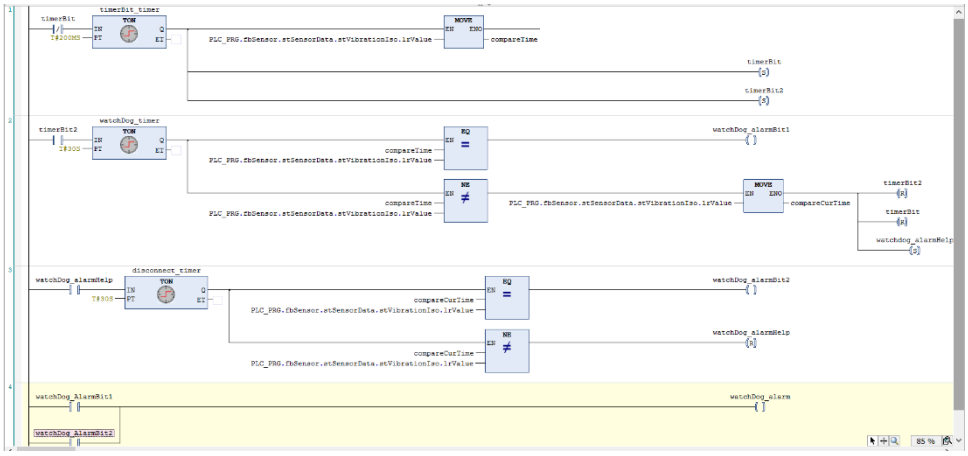


Kuva 16. Aliohjelmien kansiorakenne

Alarms-ohjelma (Kuva 17) vahtii, onko millään DUV:illa varoitus- tai hälytystilaa päällä. Hälytystilat löytyvät pääohjelman yhteyden avulla. Hälytystilojen arvoja voi muokata SmartWeb:in kautta. Jos hälytystila menee päälle, visuaaliseen käyttöliittymään tulee siitä hälytys. Ohjelma toimii myös watchDog_duv-ohjelmien kanssa (Kuva 18). Jokaisella DUV:illa on oma watchDog_duv-ohjelma, joka vahtii vain yhtä laitetta. Ohjelmat vahtivat, että DUV:ien raportoima data muuttuu tietyin väliajoin. Antureiden mittaamassa värinän datassa on aina hieman varianssia. Vanhan ja uuden datan vertailulla varmistetaan, että DUV:it ovat mittaustilassa, eikä yhteys niihin ole katkennut. Mikäli data pysyy samana määrittellyn ajan, voidaan todeta, että yhteys on katkennut tai DUV on pysähtynyt. Tämä muuttaa tietyn muuttujan todeksi, mikä taas asettaa Alarms-ohjelmassa hälytyksen päälle, joka näkyy käyttöliittymässä hälytyksenä.

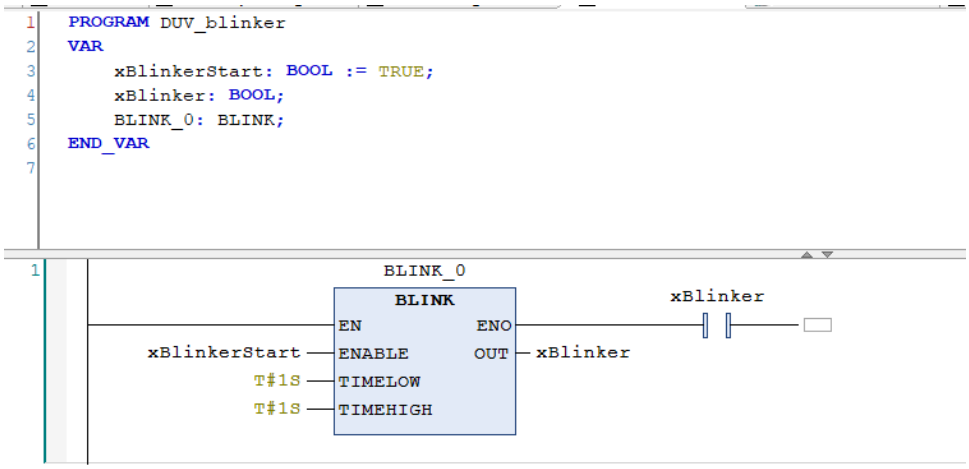
```
IF PersistentVars.xAllowDuvTemp = FALSE THEN
    IF lrTempValue1 > PLC_PRG.fbSensor.stSensorData.stTemperature.lrUpperPreAlarm THEN
        xDuvTempWarn1 := TRUE;
    ELSE
        xDuvTempWarn1 := FALSE;
    END_IF
    IF lrTempValue1 > PLC_PRG.fbSensor.stSensorData.stTemperature.lrUpperMainAlarm THEN
        xDuvTempAlarm1 := TRUE;
    ELSE
        xDuvTempAlarm1 := FALSE;
    END_IF

    IF lrTempValue2 > PLC_PRG.fbSensor2.stSensorData.stTemperature.lrUpperPreAlarm THEN
        xDuvTempWarn2 := TRUE;
    ELSE
        xDuvTempWarn2 := FALSE;
    END_IF
    IF lrTempValue2 > PLC_PRG.fbSensor2.stSensorData.stTemperature.lrUpperMainAlarm THEN
        xDuvTempAlarm2 := TRUE;
    ELSE
        xDuvTempAlarm2 := FALSE;
    END_IF
```

Kuva 17. Alarms ohjelma



Kuva 18. Duv_watchDog ohjelma

Ohjelma nimeltä DUV_blinker (Kuva 19) varmistaa, että yhteys käyttöliittymän ja logiikan välillä pysyy aktiivisena. Ohjelma on kaksiosainen ja toinen osa suoritetaan käyttöliittymän taustalla. Logiikan puoli muuttaa muuttujan tilaa sekunnin välein. Käyttöliittymän osuus vahtii kyseistä muuttujaa ja mikäli se pysyy muuttumattomana useamman sekunnin ajan, on näiden kahden ohjelman välinen yhteys katkennut. Tämän jälkeen käyttöliittymään ilmestyy virheilmoitus yhteyden katkeamisesta, jotta käyttäjä saa mahdollisimman nopeasti tietoonsa, ettei käyttöliittymän data enää päivity.



Kuva 19. Duvblinker ohjelma

Ohjelma nimeltä numbersToStrings (Kuva 20) muuttaa lukuarvoja niitä vastaaviksi merkkijonoiksi. Esimerkiksi kun anturin vikatilan arvo on nolla, ei laite ole kunnolla käynnissä. Tällöin vikatilan arvo on Inactive. Kun arvo on yksi, laite on käynnissä mutta ei hälytystilassa. Tilaksi muuttuu tässä tilanteessa No alarm. Ohjelma myös tekee saman pääohjelman yhteyden kanssa. Se vahtii DUV:ien diState-muuttujan tilaa, ja muuttaa tilan käyttäjälle helpommin luettavaksi tekstiksi.

```
CASE PLC_PRG.fbSensor.stSensorData.iAlarmState OF          //ALARM STATE
    0:
    sAlarmState1 := "INACTIVE";
    1:
    sAlarmState1 := "NO ALARM";
    2:
    sAlarmState1 := "PREALARM";
    3:
    sAlarmState1 := "ALARM";
END_CASE
```

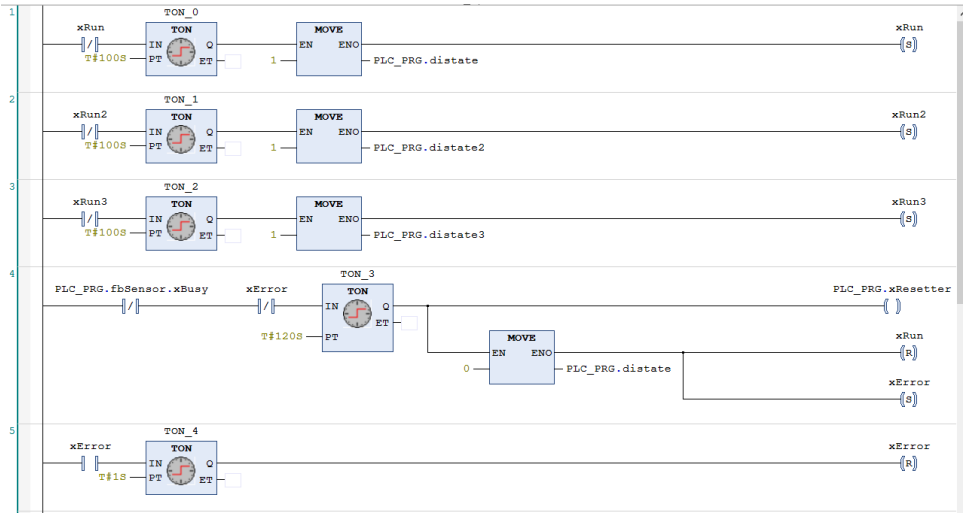
Kuva 20. NumbersToStrings ohjelma

timeStamp-niminen ohjelma (Kuva 21) hyödyntää pääohjelmaa ja sen syklejä laskeakseen, kuinka kauan ohjelmisto on ollut toiminnassa. Pääohjelma käy kaikki rivinsä läpi kerran millisekunnissa. Jokaisella ohjelmiston kierrolla se lisää laskurin arvoa yhdellä. Tästä laskurista saa selville, kuinka kauan ohjelma on ollut päällä, muuntamalla arvot sekunneiksi, minuuteiksi, tunneiksi ja päiviksi. Nämä arvot muutetaan yhdeksi helposti luettavaksi arvoksi ohjelman sisällä ja kyseinen arvo näytetään käyttöliittymässä.

```
iRunTimeSeconds := PLC_PRG.dwcounter / 1000;
iRunTimeMinutes := iRunTimeSeconds / 60;
iRunTimeHours := iRunTimeMinutes / 60;
iRunTimeDays := iRunTimeHours / 24;

iVisuTimeSeconds := iRunTimeSeconds - (iRunTimeMinutes*60);
iVisuTimeMinutes := iRunTimeMinutes - (iRunTimeHours*60);
iVisuTimeHours := iRunTimeHours - (iRunTimeDays*24);
iVisuTimeDays := iRunTimeDays;

```

Kuva 21. Timestamp ohjelma

Viimeinen ohjelma, wakeUpTimer (Kuva 22), vastaa pääohjelman käynnistyksestä, sekä DUV:ien uudelleenkäynnistymisestä vikatilassa. Kun PLC ja pääohjelma käynnistyy, DUV:it eivät aloita alustusta automaattisesti. Tämä ohjelma käynnistää ajastimen, jonka laukeaminen kestää kaksi minuuttia. Kahden minuutin jälkeen pääohjelmassa tapahtuu DUV:ien alustus. Tämä auttaa DUV:eja käynnistymään varmemmin. Ohjelmassa on myös toinen osio, missä ensimmäisen käynnistyksen jälkeen ohjelma käynnistää alustuksen alusta, mikäli DUV ei käynnistynyt kunnolla ensimmäisen alustuksen jälkeen. Tämä voi auttaa mahdollisissa vikatilanteissa, tai jos ohjelma kadottaa yhteyden DUV:iin väliaikaisesti.



Kuva 22. Wakeuptimer ohjelma

3.3 Visuaalinen käyttöliittymä

Projektiin on kehitetty erillään toimiva visuaalinen käyttöliittymä. Käyttöliittymän tulee olla helposti käytettävä, sekä näyttää kaikki käytössä olevat DUV-anturit kerralla.

Käyttöliittymässä on useita eri sivuja, joista näkee antureiden antamaa dataa, hälytyksiä tai trendejä anturien datasta.

3.3.1 CODESYS Visualization

Käyttöliittymä on toteutettu hyödyntäen CODESYS Visualization-ohjelmaa. CODESYS-ohjelmointiympäristössä on mahdollisuus lisätä visuaalisia elementtejä erilaisille sivuille. Nämä visuaaliset elementit voivat hyödyntää ohjelmistojen muuttujia ja jopa muokata niitä. Projektissa on luotu erillinen käyttöliittymä-sovellus, joka on yhteydessä projektin logiikkaan. Käyttöliittymä lukee projektin dataa luomalla yhteyden PLC:hen, joka suorittaa logiikkaa. Käyttöliittymä näyttää PLC:n dataa helposti luettavassa muodossa joko suoraan arvoina tai erillisinä graafeina. CODESYS Group on luonut erillisen CODESYS Visualization -ohjelman, jonka avulla voi millä tahansa ohjelmaa tukevalla laitteella käyttää kyseistä käyttöliittymää ilman muita asennettavia sovelluksia, mikäli päätelaite on samassa verkossa kyseisen projektin kanssa.

CODESYS Visualizationin lataamisen voi ohittaa, jos visualisointia hallinnoivassa laitteessa on CODESYS WebVisu käytössä. Tämä ominaisuus muuntaa CODESYS Visualization -käyttöliittymän verkkoselaimelle yhteensopivaksi, sekä jakaa tätä verkkosivustoa lähiverkossa. Tämän jälkeen käyttöliittymää voi käyttää millä tahansa laitteella, joka tukee moderneja verkkoselaimia ja on samassa verkossa kuin visualisoinnista vastaava PLC. (CODESYS Group.)

Projektissa oli lähtökohtana helposti lähestyttävä käyttöliittymä, mistä näkisi kerralla kaikki mahdolliset projektin laitteet ja niiden tämänhetkisen tilat Käyttöliittymän lähtökohtana oli yksinkertaisuus ja tärkeän datan tuominen näkyviin. Projektin käyttöliittymä on suunniteltu käytettäväksi tietokoneella ja isommalla näytöllä, joten mobiililaitteilla katseltuna WebVisu on heikosti toimiva.

Käyttöliittymän tärkeimpänä ominaisuutena on nähdä projektiin liitettyjen laitteiden tärkeimmät arvot ja tilat mahdollisimman selkeästi. Tämä toteutuu hyödyntämällä logiikan ohjelmaa. Käyttöliittymällä on yhteys logiikkaan, ja sen muuttujiin. Muuttujat näytetään käyttöliittymässä ja niiden arvoja päivitetään jatkuvasti. Logiikassa on myös ylimääräisiä ohjelmia, jotka on luotu käyttöliittymää varten. Esimerkiksi joitakin arvoja muutetaan logiikassa helpommin luottavaksi.

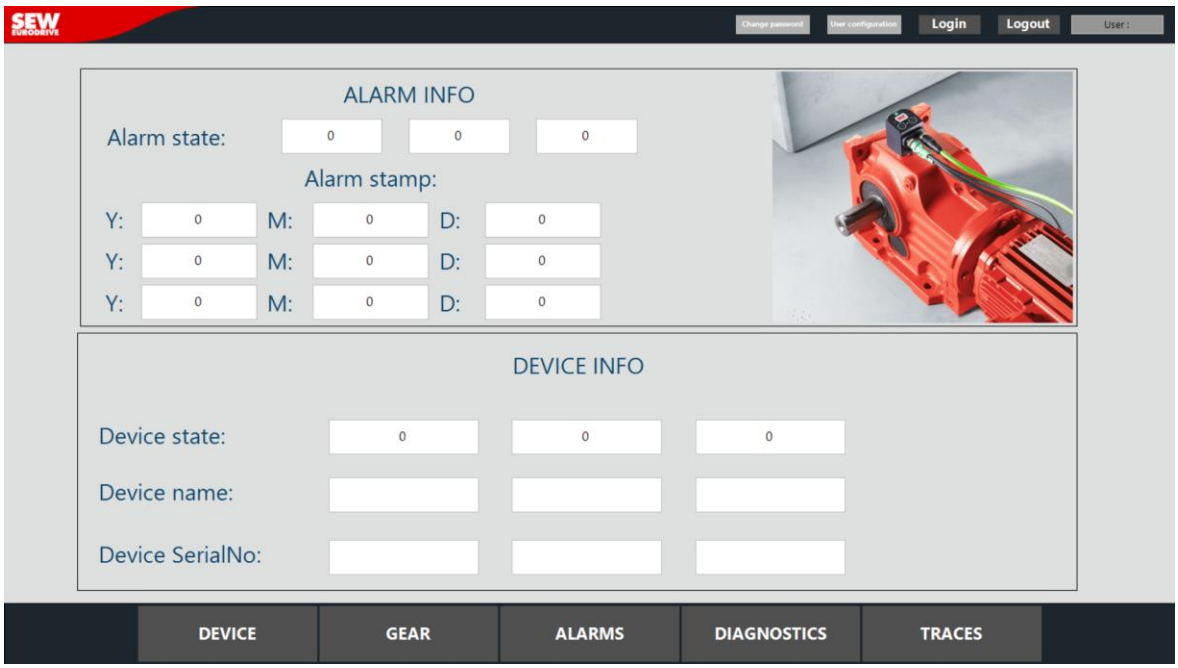
3.3.2 Projektin sovellus

Käyttöliittymän sivut koostuvat CODESYS:in Visualization Toolbox -valikosta löytyvistä valmiista erilaisista visuaalisista työkaluista, kuten tiettyjä arvoja näyttävistä

tekstilaatikoista, Traces-trendeistä, sekä sivun lukemista helpottavista viivoista. Työkalut vedetään sivulle työkalulaatikosta, jossa niitä pääsee muokkaamaan haluamikseen, esimerkiksi muuttamalla työkalun kokoa, väriä tai näytettäviä arvoja. Työkaluilla on tietynlainen visuaalinen tyyli, joka riippuu projektissa valitusta Visualization Manager:in tyylistä. Tässä projektissa on käytössä SEW-EURODRIVE:n oma tyyli.

Käyttöliittymään kuuluu aina näkyvissä ja aktiivisena olevat ylä- ja alapalkit. Yläpalkissa on SEW-EURODRIVE:n logo, sekä käyttäjien hallinnointi. Yläpalkki toimii myös erillisenä yhteystilan varoituksena, mikäli käyttöliittymän ja logiikan yhteys katkeaa. Mikäli käyttöliittymän ja logiikan välinen yhteys katkeaa, tulee yläpalkkiin tästä ilmoitus. Alapalkista voi vaihtaa aktiivisena olevaa näkymää. Palkit on tehty omana erillisenä sivuna, joka tulee käyttöliittymässä aina päällimmäiseksi. Palkit ovat näkyvissä alla olevissa kuvissa.

Ensimmäinen sivu mikä tulee näkyviin käyttöliittymässä, on DEVICE-valikon alla. Kyseisen sivun alaosa näyttää projektiin kiinnitetyt DUV-kunnonvalvontajärjestelmien nimet, toimintatilan ja sarjanumeron. Ylempi osuus näyttää laitteiden hälytystilat sekä hälytystilan aikaleiman. Sivulla on myös tyhjän tilan täytteenä kuva kiinnitetystä DUV-kunnonvalvontajärjestelmästä. Kyseinen sivu on nähtävissä alapuolella kuvassa 23.

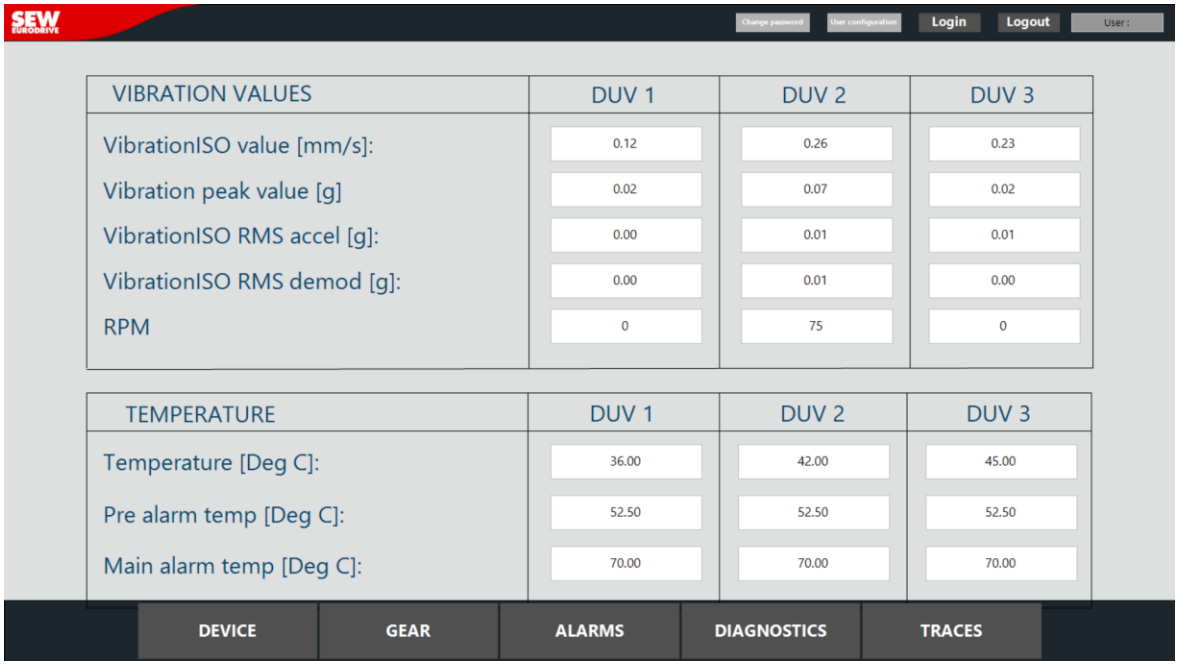


Kuva 23. Device-valikko

Gear-valikko (Kuva 24) antaa käyttäjälle laajemman kuvan järjestelmän tilasta Sieltä näkee jokaisen laitteen tärkeimmät arvot suoraan. Sivulla on myös näkyvissä mahdollinen ylimääräinen tulo DUV-anturilta. Tässä projektissa tulokanavaan on liitetty analoginen tulo moottorin virtamuuntajalta, joka muutetaan DUV:in SmartWebin avulla moottorin

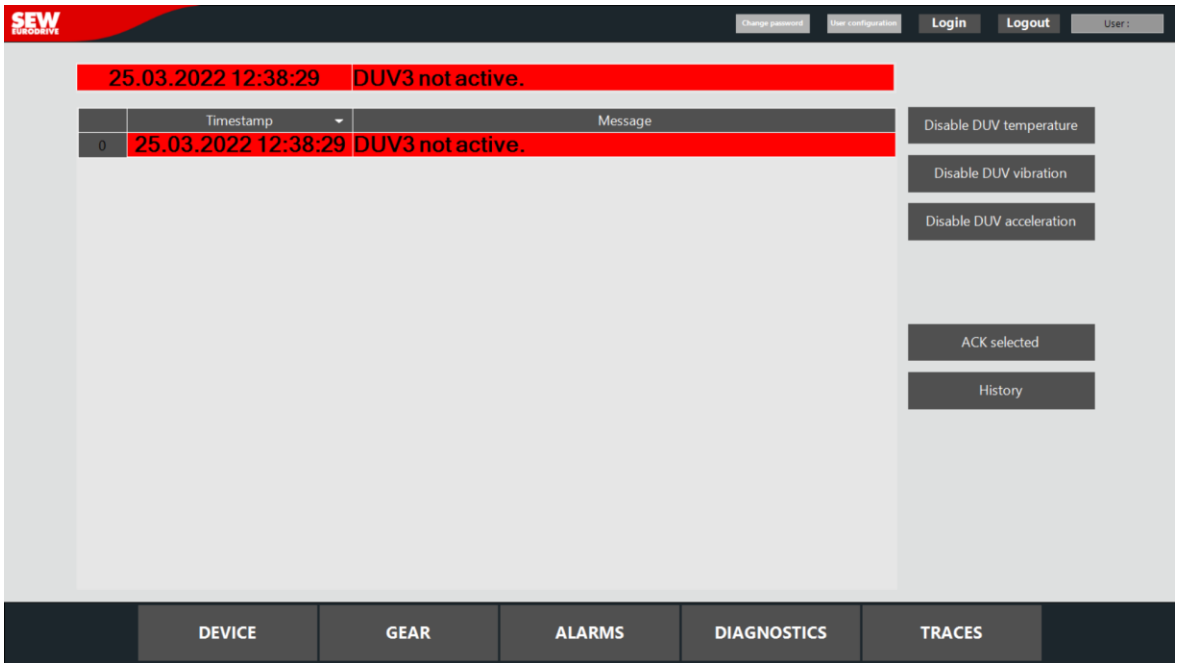
kierroslukumittariksi. Moottorin kierrosmäärä mahdollisen hajoamisen yhteydessä on hyödyllistä dataa. SmartWebin avulla voidaan myös määrittää moottorin kierroksiin liittyviä hälytyksiä, mikäli arvo poikkeaa normista.

Anturin ylimääräinen tulo kommunikoi OPC-UA:n avulla. Projektin logiikka luo ylimääräisen yhteyden DUV järjestelmään käynnistyksen yhteydessä. Tällä yhteydellä siirretään anturista dataa, joka taas muutetaan logiikassa esitettävään muotoon ja lopulta näytetään käyttöliittymässä. Muutkin käyttöliittymässä käytetyt arvot on tuotu näkyviin kyseisellä tavalla, mutta yhteydessä ei ole käytetty OPC-UA kommunikaatioprotokollaa.



Kuva 24. GEAR-valikko

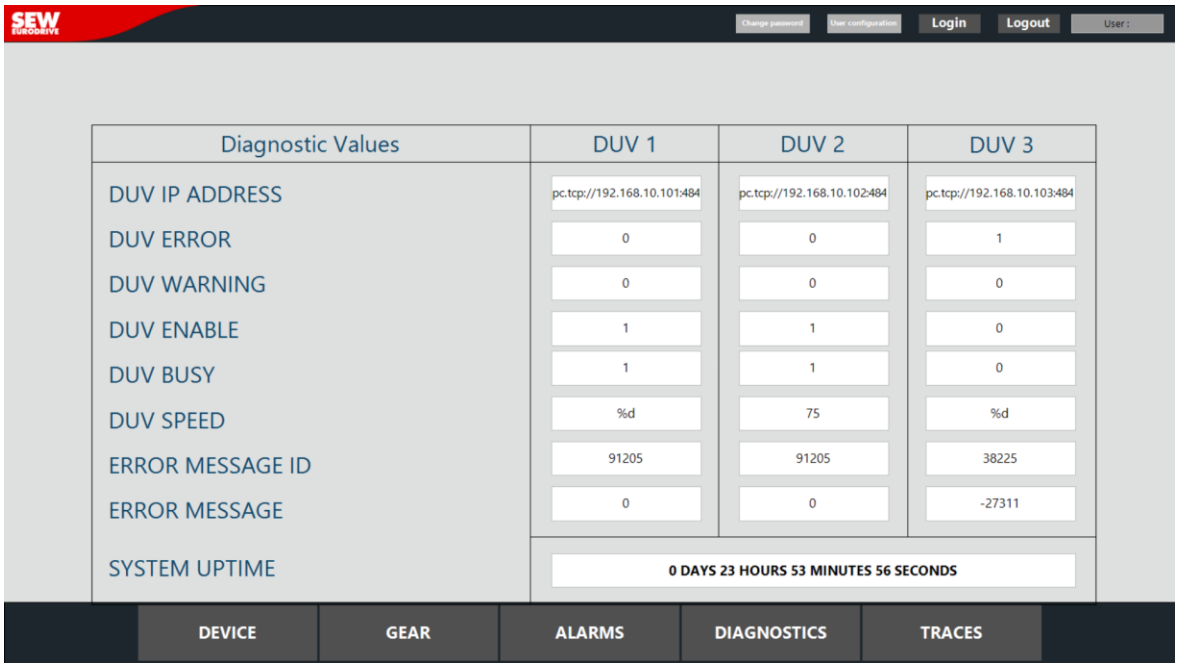
Käyttöliittymän kolmas sivu on hälytykset (Kuva 25). Tälle sivulle tulee ilmoitus, jos jollakin DUV:eista on hälytys aktiivisena. Hälytysten raja-arvot voidaan asettaa SmartWebin kautta. Ohjelma näyttää hälytyksen näkymässä punaisella taustalla, ja varoitukset keltaisella taustalla. Käyttäjä voi myös halutessaan ottaa osan hälytyksistä pois päältä oikean reunan valikoista. Hälytyksistä myös kerätään PLC:n muistiin historialistaa, josta voi tutkia hälytyksiä myöhemmin. Näihin tallentuu aikaleimat, jotka kertovat, milloin hälytys on tapahtunut. Projektiin on myös lisätty muitakin hälytysarvoja antureiden arvojen lisäksi. Esimerkiksi, jos anturit lopettavat päivittämästä arvojansa, menee hälytys päälle Duv_watchDog-ohjelman ansiosta. Tämä tarkoittaa sitä, että anturi ei ole jostain syystä yhteydessä tai päällä ja se vaatii käyttäjältä toimenpiteitä.



Kuva 25. ALARMS-valikko

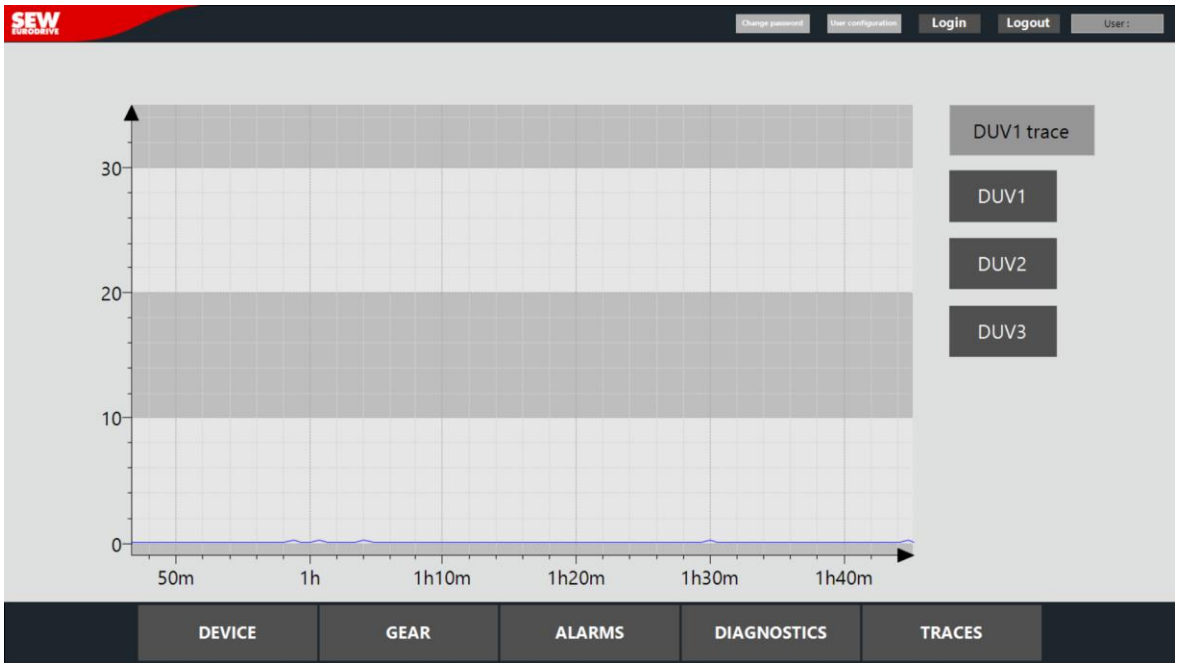
DIAGNOSTICS-sivulla (Kuva 26) on edistyneemmälle käyttäjälle tarkoitettua tietoa. Sivulle on kerätty erilaisten tilojen tietoja DUV:eilta, sekä kuinka kauan logiikka on ollut toiminnassa. Käyttöliittymän ylärivillä on laitteen osoite, jotta käyttäjä tietää, mistä laitteesta on kyse. Toisella rivillä on vikatilan informaatio. Jos vikatilan arvo on 1, laitteella on vikatila aktiivisena. Alimmat rivit näyttävät vikatilasta lisätietoa. ERROR MESSAGE ID ja ERROR MESSAGE -rivit antavat vikakoodeja, jotka merkitsevät erityyppisiä mahdollisia vikoja laitteissa.

Keskimmäiset rivit kertovat laitteen käynnissä ollessa muuttuvista arvoista. DUV WARNING -rivi kertoo, onko laitteella aktiivisena varoitusta, esimerkiksi lämpötilasta. Tämä on ylimääräisenä tässä näkymässä, koska käyttäjä voi asettaa omia varoituksia digitaalisen tai analogisen tulon kautta, jotka eivät mahdollisesti näkyisi ALARMS-välilehdessä. DUV ENABLE -rivin arvo kertoo siitä, onko ohjelma tällä hetkellä käynnistänyt anturia. Tämä arvo menee väliaikaisesti pois päältä, mikäli ohjelmassa on kommunikaatio-ongelma ja se yrittää korjata sitä. DUV BUSY -rivin muuttuja kertoo siitä, onko DUV aktiivisesti ajamassa ohjelmaansa. Anturilla voi olla aktiivisena vikakoodi, mutta anturi voi olla ilman vikatilaa. Tässä tapauksessa tämä muuttuja muuttuisi todeksi. Kuvassa oleva DUV SPEED -rivin muuttuja näyttää analogisen tulon muuttujan arvon. Projektissa analogisessa tulossa on moottorin pyörimisnopeus, joten sivulla se on nimetty kyseiseen muotoon.



Kuva 26. DIAGNOSTICS-valikko

Käyttöliittymän viimeisellä sivulla on näkymässä trendi DUV:in tärinästä (Kuva 27). Trendi visualisoi DUV:in lähihistorian dataa. Se kerää lyhyin väliajoin datasta pisteen ja näiden pisteiden avulla saadaan piirrettyä graafi. Trendin arvoja voidaan muuttaa, jotta saadaan näytettyä pidemmältä aikaväliltä dataa harvemmin, tai lyhyemmältä aikaväliltä tarkemmin. Oikealla puolella trendiä on harmaa laatikko, mistä näkee, minkä laitteen trendi on aktiivisena näkymässä. Aktiivinäkymää voi vaihtaa laatikon alapuolelta. Ohjelma nauhoittaa trendejä riippumatta siitä, onko trendi tällä hetkellä aktiivinen käyttöliittymässä. Ohjelma myös tallentaa trendien arvot lokaalisti SQLite-tietokantaan.



Kuva 27. TRACES-valikko

4 Projektin testaus

Projektin käyttöä kyettiin testaamaan vain toimisto-olosuhteissa. Pieneen W10/DT56L4-vaihdemoottoriin on laitettu laakeriin ylimääräinen paino, mikä tuottaa epätavallisen suuren määrän värinää, erityisesti kun vaihdemoottori käy korkeammilla kierroksilla. Tämän lisäksi mittaavat DUV40A-järjestelmät on sijoiteltu hieman eri paikkoihin, jotta ne antavat erilaista dataa. Näiden avulla voidaan simuloida sitä, miten projekti suoriutuisi oikeassa tilanteessa.

Kaikki projektin toiminnot täyttävät vaatimukset ja minimitavoitteet. Anturit mittaavat dataa ja käyttöliittymä päivittyy muutosten mukaan. Hälytykset toimivat niin antureilla kuin myös käyttöliittymässä. PLC kykenee luomaan yhteyden uudelleen antureihin, jos jokin antureista menettää yhteytensä väliaikaisesti. Projekti oli kokonaisuudessaan stabiili, ja se onnistui pitämään PLC:tä ja antureita käynnissä yhtäjaksoisesti yli kahdeksan vuorokauden ajan, samalla pyörittäen mitattavaa moottoria minimaalisesti. Myös käyttöliittymän käyttäminen useammalla laitteella WebVisun avulla saman verkon sisällä onnistui.

Vaihdemoottorin värinä on epätasaista, eikä se vastaa normaalissa käytössä olevan moottorin värinää. Tämän takia toimiston tutkimuksen dataan ei voi luottaa. Värinän määrä heittelehtii erittäin paljon jo pienen kierroslukujen vaihtelun jälkeen. Järjestelmää ei saada myöskään pidettyä tarpeeksi pitkään käytössä yhtäjaksoisesti, ja datan laatu on liian heikkoa, että siitä voisi saada käytäntöön soveltuvaa tietoa. Myös antureiden hälytysrajat on kalibroitu eri tavalla kuin ne olisivat oikeassa käytössä johtuen värinän epätasaisuudesta sekä testaamisolosuhteiden helpottamisesta. Optimaalisessa tilanteessa mittaus kestäisi huomattavasti pidemmän ajan, käytössä olisi suurempi vaihdemoottori, sekä värinät olisivat tasalaatuisempia.

Toimiston testin lisäksi olisi ollut hyvä saada suunniteltu isompi testi tehtyä, mutta useiden viivästysten jälkeen kyseinen testi ei ehdi käynnistymään vielä lähiaikoina. Testi on kuitenkin aloituspisteessä, ja sen vienti kyseiseen pisteeseen vaati hieman projektin muokkausta.

5 Kenttätestin pohjustus

Projektin kasaus ja testaus toimistoympäristössä onnistui jollain tasolla, mutta toimiston pienikokoinen vaihdemoottori ei ole realistinen kohde mitä mitata, joten data voi olla vääristynyttä. Tämän takia on tärkeää saada siirrettyä projekti mittaamaan jotakin lähempänä teollisuuden normeja. SEW-EURODRIVE:llä on huollon puolella käytöstä poistettuja sähkömoottoreita ja -vaihteita, mitä päätettiin käyttää hyödyksi projektin isommassa testissä. Testin ideana on ajaa aluksi vaihdetta normaalisti ja sitten hiljalleen vähentää vaihteen öljymäärää, simuloiden pientä öljyvuotoa. Vaihde olisi ajossa mahdollisimman paljon, jotta dataa saataisiin mahdollisimman paljon.

Testiä varten projekti kuitenkin tarvitsisi pieniä muutoksia. Toimiston versiossa on käytössä kolme DUV:ia, mutta testissä olisi vain kaksi. Testissä olisi myös käytössä WLAN-yhteys, jotta projektin tilanteen tarkastus olisi helpompaa, koska sen voisi tehdä langattomasti, kunhan vain olisi samassa tilassa. Tämän jälkeen käyttäjän tarvitsee vain liittyä kyseiseen verkkoon langattomasti ja vaihtaa oman päätelaitteen osoite oikeaksi. Käyttäjä voi sitten avata päätelaitteellansa verkkoselaimen ja mennä WebVisun osoitteeseen, ja käyttöliittymä olisi täten saatavissa.

Ennen testin aloittamista todettiin, että projektin käyttöliittymän toimivuus on heikko johtuen käyttöliittymän käytön rajallisuudesta. Käyttö vaatii, että CODESYS Visualization -ohjelma ladataan koneelle. Huomattiin, että lataamisen voi ohittaa WebVisun avulla, mutta WebVisun käyttö vaatii, että jokin laite silti pitää projektia päällä CODESYS Visualizationin avulla. Tämän ohittamiseen käyttöön otettiin PLC:n toinen muistikorttipaikka ja sinne laitettiin muistikortti, mihin oli ladattu Windows 10 IoT Core -käyttöjärjestelmä. Kyseinen käyttöjärjestelmän tehtävä on käynnistymisen jälkeen käynnistää CODESYS Visualization-ohjelma. Tämän jälkeen muut laitteet voivat käyttää käyttöliittymää WebVisun avulla, ilman että päätelaitteelle tarvitsee ladata ylimääräisiä ohjelmistoja. Käyttöjärjestelmää varten tarvitsi vain lisätä käyttöjärjestelmä samaan verkkoon, ja ladata CODESYS Visualization –ohjelma muistikortille.

Projektiin lisättiin myös ylimääräinen anturi. Tätä varten projektiin lisätään analogiatulomoduuli OAI45C, jotta saatiin liitettyä ja luettua Pt1000-anturin dataa. Pt1000 on lämpötila-anturi, joka sijoitettiin projektissa eri osaan vaihdetta kuin DUV:it. Tällä testataan, kuinka paljon DUV:in lämpötilamittari heittää luotettavaksi todetusta anturista. DUV:in lämpötilamittaukseen vaikuttaa järjestelmän oma lämpötila, joten viileämmässä ilmastossa järjestelmä ilmoittaa lämpötilan olevan hieman korkeampi kuin se oikeasti on, mikä voi vaikuttaa testiajon aikana tulevaan dataan. Projektiin kiinnitettiin myös tulevaisuuden testauksia varten ylimääräisiä I/O-moduuleita. Näillä ei ole käyttöä vielä,

mutta ne mahdollistavat sen, että jos testin aikana tulee uusia ideoita, niiden toteuttaminen nopeasti olisi helpompaa.

Anturin lisääminen projektiin oli nopeaa ja helppoa. Kun tarvittavat I/O-moduulit oli liitetty fyysisesti paikoilleen ja johdot yhdistetty PLC:hen sekä anturiin, piti projekti avata MOVISUITE:lla. Projektissa lisättiin skannaustoiminnon avulla kaikki uudet I/O-moduulit osaksi projektia. Tämän jälkeen alustettiin projekti lukemaan oikeaa muistilohkoa oikeassa muodossa käynnistys parametrien avulla. Kun projektille on kerrottu, että kyseessä on Pt1000-anturi ja kerrottu, missä muistilohkossa se sijaitsee, osaa CODESYS muuttaa anturin antamat arvot suoraan valittuun muotoon, tässä tapauksessa celsiukseksi. Tämä data otetaan sitten pääohjelmaan muuttujaksi, jonka jälkeen käyttöliittymässä näytetään kyseinen muuttuja.

Näiden muutosten jälkeen testi oli aloittamista vaille valmis. Kuitenkin matkassa oli useampi epäonninen muutos ja testin aloitus viivästyi niin paljon, että tähän tutkimukseen ei saatu yhtään dataa. Laitteiden saatavuus oli heikkoa, joten esimerkiksi taajuusmuuttajien saaminen oli hidasta, sekä testipaikkaa jouduttiin vaihtamaan useaan otteeseen.

6 Yhteenveto

Ilman testin tuloksiakin projekti on vielä osittain keskeneräinen ja siitä löytyy paljon parannettavaa. Päällimmäinen kehitysidea on projektin kenttätestin toteutus. Projektiin tullaan jatkossa lisäämään antureita enemmän, suunnitelmissa on, että ainakin kolme kappaletta lisää. Projektissa on myös yksi kriittinen puutos, mitä ei saatu ratkaistua tutkimuksen aikana. Projektin pitäisi kyetä lähettämään keräämänsä datat eteenpäin jonkinlaiseen pilvipalveluun. Ominaisuutta on tutkittu, mutta se on toistaiseksi karsittu projektista pois, kunnes ensimmäiset testit saadaan suoritettua. Tulevaisuudessa pitää myös kehittää projektin datan keräämistä. Mahdollisia väliaikaisia piikkejä pitäisi kyetä suodattamaan paremmin. Tällä hetkellä voi käydä niin, että väliaikainen ympäristössä tapahtuva liike anturin välittömässä läheisyydessä voi vaikuttaa anturin dataan, tosin mahdollisuus siihen on pieni, ja tapauksen merkitys kokonaisuudelle on vähäinen.

Projekti kuitenkin onnistuu ainakin osittain tavoitteessaan. Projektin avulla voidaan monitoroida laitteita, jonka pitäisi auttaa mahdollisessa kunnonvalvonnassa. Projektissa on myös jonkinlaista tiedon tallennusta DUV40A-järjestelmien oman muistin ansiosta, sekä käyttöliittymän datan keräämisestä. Käyttöliittymän kehitys sujui hyvin, ja lopputulos on toimiva. Projektin kehityksen aikana ilmenneistä teknisistä ongelmista päästiin yli, kuten lämpötilamittauksen riittämättömyys testiprojektissa, sekä käyttöliittymän rajallisuus yhdellä tietokoneella.

Lähteet

ABB. TTT-käsikirja. Kunnonvalvonta ja huolto, 2000. Viitattu 8.4.2022. Saatavissa http://www.oamk.fi/~kurki/automaatiolabrat/TTT/23_Kunnonvalvonta%20ja%20huolto.pdf

Cass, S., Kulkarni, P. & Guizzo, E. 2021. Top Programming Languages. IEEE Spectrum. Viitattu 16.3.2022. Saatavissa https://spectrum.ieee.org/top-programming-languages/

CODESYS Group. CODESYS WEBVISU. Viitattu 8.4.2022. Saatavissa https://www.codesys.com/products/codesys-visualization/webvisu.html

Helle A. 2004. Teollisuuden käynnissäpidon prognostiikka. Viitattu 12.4.2022. Saatavissa https://www.vttresearch.com/sites/default/files/pdf/symposiums/2005/S236.pdf

Nieke R. 2022. Programming PLCs using Structured Text. University of Nijmegen. Saatavissa http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.49.2016&rep=rep1&type=pdf

OPC Foundation. OPC Classic a. Viitattu 8.4.2022. Saatavissa https://opcfoundation.org/about/opc-technologies/opc-classic/

OPC Foundation. Unified Architecture b. Viitattu 8.4.2022. Saatavissa https://opcfoundation.org/about/opc-technologies/opc-ua/

O&M Best Practices Guide, 2022. Types of Maintenance Programs. Viitattu 12.4.2022. Saatavissa https://www1.eere.energy.gov/femp/pdfs/OM_5.pdf

Parr, E. A. 1998. Computers and Industrial control. Industrial Press Inc. Saatavissa https://books.google.fi/books?redir_esc=y&hl=fi&id=zLwtngK3T1UC&q=438#v=snippet&q=438&f=false

Prytz G. 2008. A performance analysis of EtherCAT and PROFINET IRT. ABB AS Corporate Research Center. Viitattu 13.3.2022. Saatavissa https://www.ethercat.org/pdf/english/ETFA_2008_EtherCAT_vs_PROFINET_IRT.pdf

Sandberg-Diment, E. 1984. Personal computers; Preventive maintenance for an aging computer. The New York Times. Saatavissa https://www.nytimes.com/1984/08/14/science/personal-computers-preventive-maintenance-for-an-aging-computer.html

SEW EURODRIVE. A tradition of movement. Viitattu 8.4.2022. Saatavissa https://www.sew-eurodrive.fi/company/our_drive/history/history.html

SEW EURODRIVE. Manual Diagnostic Unit Vibration DUV40A. 2020. Saatavissa https://download.sew-eurodrive.com/download/pdf/29190266.pdf

Liite 1. Haastattelu kunnonvalvonnasta

Tutkimusta varten haastateltiin Sew Eurodrive:llä huollon puolella työskentelevää kunnossapidon ammattilaista. Kysyin häneltä muutaman aiheeseen liittyvän kysymyksen sähköpostilla, joihin hän vastasi seuraavanlaisesti:

Mitä dataa pitää saada datan analyysia varten? Esimerkiksi hampaiden hammasluvut, laakerien tyypit, nopeusalueet, yms.

> *Luotettavaa analyysiä varten kaikki mitattavan kohteen pakkotaajuudet pitää tuntea. Pakkotaajuuksia on vaihteiden tapauksissa hammastusten ryntötaajuudet, laakerien eri osien pyörimistaajuudet sekä akseleiden pyörimistaajuudet. Näiden taajuuksien laskemiseen tarvitaan nuo mainitsemasi hammasluvut, laakerien tyypit ja käytettävät nopeudet.*

Mihin dataan analyysi perustuu?

> *Analyysi perustuu mitattuun dataan, tätä voi olla esimerkiksi; värähtely, lämpötila, moottorin käyttämä virta, kuormitustieto applikaatiosta.*
>
> *Värähtelyn tapauksessa: Värähtelysignaalia analysoitaessa käytetään pääsääntöisesti kahta eri kuvaajaa apuna. Aikatasosignaali, eli värähtelyn amplitudi (voimakkuus) ajan funktiona. Lisäksi käytetään nopeus- ja kiihtyvyysspektrejä, spektrit saadaan aikatasosignaalista FFT-muunnoksella. Spektreissä värähtely esitetään värähtelyn amplitudi (joko nopeus tai kiihtyvyys) taajuuden funktiona.*
>
> *Aikatasosta ja spektreistä tunnistetaan mitatun kohteen pakkotaajuudet ja näihin taajuuksiin liittyvät ilmiöt (harmoniset monikerrat ja sivunauhat).*

Mitä ovat teollisuuden tyypillisimmät viat ja mistä ne yleensä johtuvat?

> *Vaihteiden tapauksessa yleisiä vaurioita ovat öljyvuodot, sekä laakeri- ja hammastusvauriot.*
>
> *Öljyvuodoissa vauriot johtuvat yleisimmin;*
>
> - *Ulkopuolinen lika*
> - *Liiallisen lämpötilan (liiallinen kuormitus tai ympäristön lämpötila)*
> - *Voitelupuute*

- *Yleinen kuluminen*
- *Virheellinen asennus*

Laakeri- ja hammastusvauriot johtuvat yleisimmin;

- *Voiteluongelmat (liian vähäinen voitelu, kontaminoitunut tai vanhentunut voiteluaine, väärä voiteluaine)*
- *Ylikuormitus (jatkuva ylikuorma, jolloin vaihde on alimitoitettu tai äkillinen ylikuormitus eli jumitilanne applikaatiossa)*
- *Virheellinen asennus*

Kuinka hyvin vikaantuminen voidaan ennakoida ajallisesti? Miten siihen voidaan vaikuttaa (Kuinka tiheästi tarvitaan dataa, laakeri vikaantuu hitaasti)

Vikaantumisen ennakointi ajallisesti on hankalaa, toisin sanoen, vaihteen viimeistä käyttöpäivää on vaikea antaa.

Trenditieto auttaa vikaantumisen ennakoinnissa. Esimerkiksi laakerien vikaantuessa vaurio esiintyy värähtelymittauksissa eri tavalla vaurion edetessä.

Mitä ammattimies havaitsee mittarien ulkopuolelta (fyysisiä ominaisuuksia, tukidata, täydentävä data)

Trenditieto auttaa vikaantumisen ennakoinnissa. Esimerkiksi laakerien vikaantuessa vaurio esiintyy värähtelymittauksissa eri tavalla vaurion edetessä.

Mitattavan kohteen ulkopuolelta tulevat värähtelyt, ymmärtää applikaatiolle tyypilliset värähtelytasot. Paikan päällä käydessä näkee ulkoiset tekijät (asennusympäristö, lämpötila, petirakenne).

Digitalisaatio vs ”fyysinen, perinteinen kunnonvalvonta”

Digitalisaatio varmastikkin tulevaisuutta. Vertaan digitaalisia kunnonvalvontaratkaisuja autojen OBD-järjestelmään, jonka yleistymiseen ja standardointiin meni hetki mutta nykyään löytyy joka autosta.

Perinteinen kunnonvalvonta on alkuinvestoinneiltaan kustannustehokas ja ketterä ratkaisu, tämä kuitenkin vaatii aina asiantuntijan mittaukseen ja analysointiin.

Kunnossapidon kustannustehokkuus

Jos otetaan esimerkki; 10k€ kunnonvalvontajärjestelmä on helpompi myydä 100k€ vaihteeseen kuin 10k€ vaihteeseen asiakkaan miettiessä hankintakustannuksia.

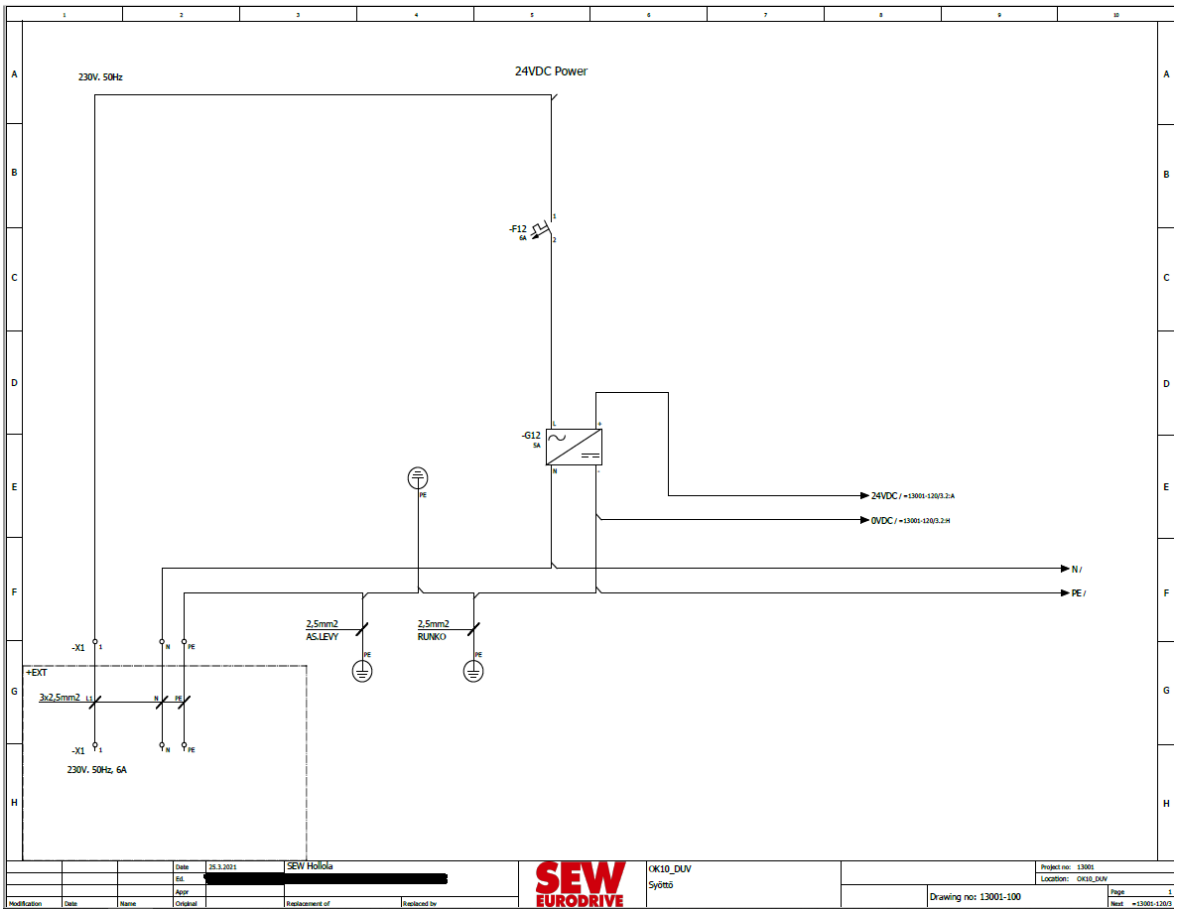
Yleisesti kuitenkin pitäisi miettiä kyseisen applikaation kriittisyys prosessin toiminnalle. 10k€ vaihteen yllättävä vikaantuminen kuitenkin saattaa aiheuttaa kustannuksia asiakkaalle 10 000 € tai jopa 100 000 € tuotantotappiona/päivä. Eli jos 10k€ investoinnilla kunnonvalvontajärjestelmään voidaan ehkäistä tuotantotappioiden syntymistä niin sehän on selvää säästöä.

Haastattelusta selviää, että värinän avulla voidaan selvittää laitteen kunnon tila. Tämän takia SEW DUV40A anturi on valittu kyseiseen projektiin. Värinäanturilla ei kuitenkaan voida estää kaikkia ongelmia, mutta sillä voidaan vaikuttaa yleisimpien ongelmien huomaamiseen. Kunnonvalvonnalla voidaan saavuttaa säästöjä. Säästöjen suuruus riippuvat monitoroitavasta kohteesta.

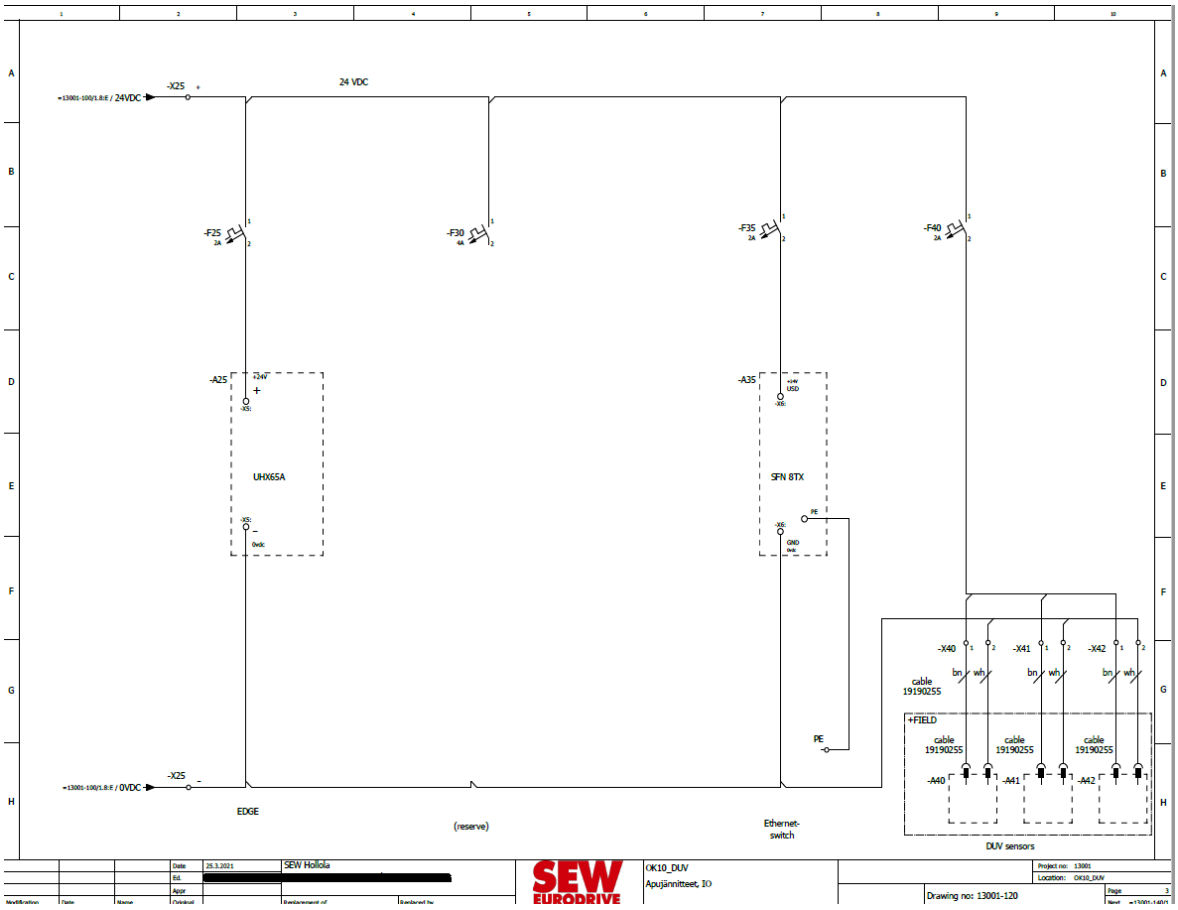
Kunnonvalvonnan digitalisoituminen ja online kunnonvalvonta tulee kasvattamaan suosiotaan. Nämä eivät tule kuitenkaan syrjäyttämään alan ammattilaisia, vaan toimimaan parempina työkaluina. Dataa tulee jatkossa enemmän, mutta edelleen tarvitaan koulutettua henkilöstöä joka osaa tutkia dataa, ja tehdä sen perusteella johtopäätöksiä.

Liite 2. Sähkökaaviot

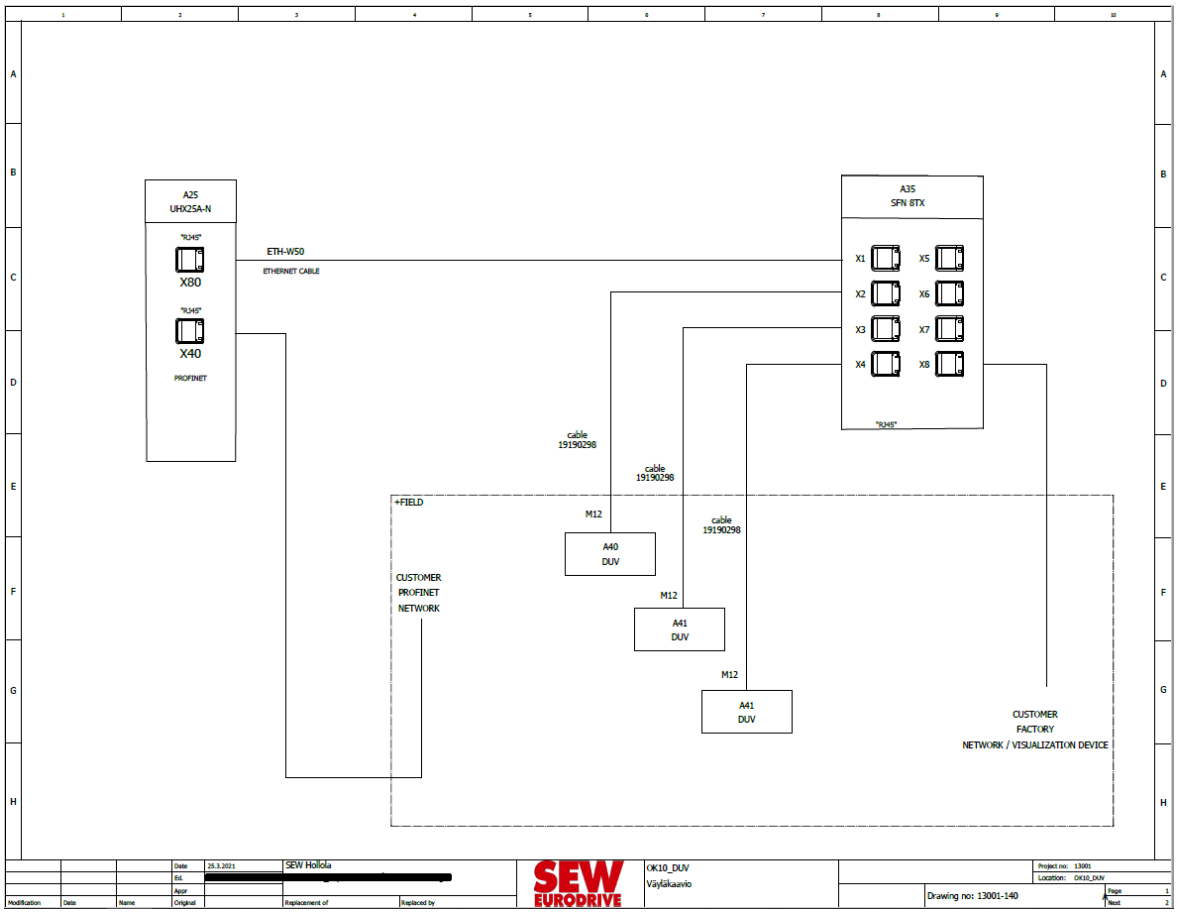
Alapuolella on projektin sähkökaaviot kuvina. Ensimmäisessä kuvassa (Kuva 1) on kaavioitu projektin sähkönsyöttö. Toisessa kuvassa (Kuva 2) on apujännitteet ja I/O. Kolmannessa kuvassa (Kuva 3) on projektin väyläkaavio. Neljännessä kuvassa (Kuva 4) on maadoituskaavio. Viimeisessä kuvassa (Kuva 5) on projektin layout.



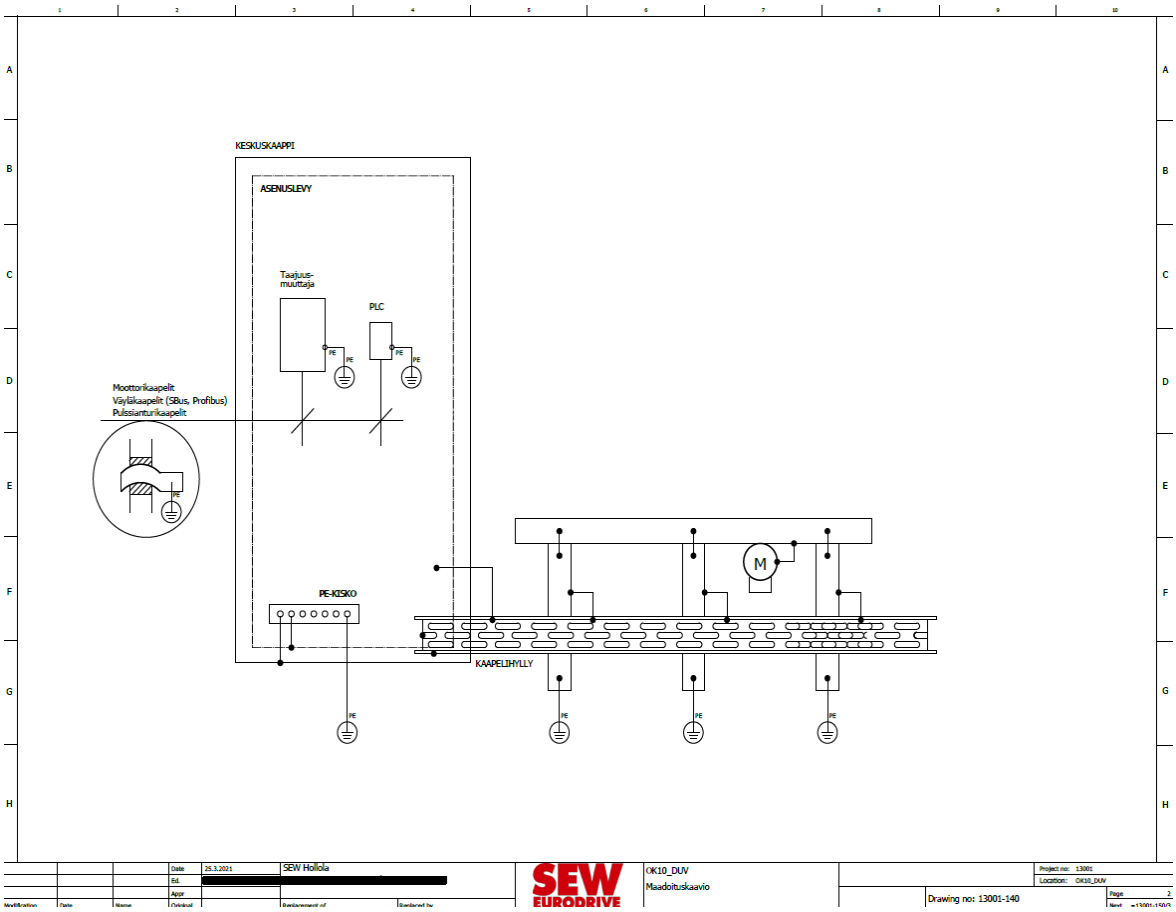
Kuva 1. Syöttö



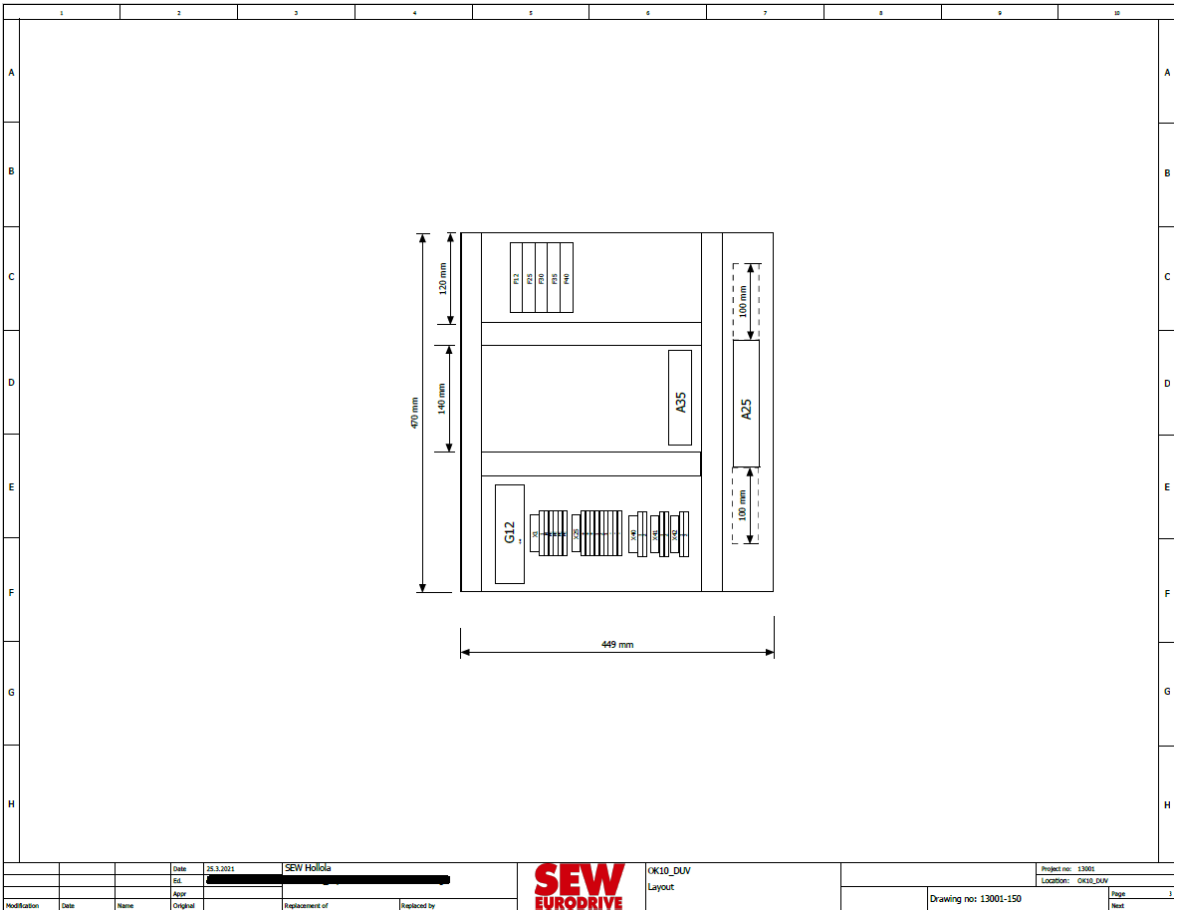
Kuva 2. Apujännitteet, I/O



Kuva 3. Väyläkaavio



Kuva 4. Maadoituskaavio



Kuva 5. Layout