Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Jaakko Koski

Jäänestojärjestelmän osien simulointi PLCnext-tuoteperheen Axioline PLC:llä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikan tutkinto-ohjelma

Insinöörityö

26.05.2021

Tekijä Otsikko Sivumäärä Aika	Jaakko Koski Jäänestojärjestelmän osien simulointi PLCnext-tuoteperheen Axioline PLC:llä 37 sivua 26.05.2021
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	sähkö- ja automaatiotekniikka
Ammatillinen pääaine	automaatiotekniikka
Ohjaajat	tekninen johtaja Tomas Wallenius lehtori Markku Inkinen

Tässä työssä perehdyttiin Phoenix Contactin PLCnext-alustaan ja toteutettiin sillä Wicetec Oy:n jäänestojärjestelmän komponenttien ja tietoliikenteen simulointeja. Tarkoituksena oli simuloida komponentteja ja Modbus TCP/IP -tietoliikennettä mahdollisimman tarkasti ja näin helpottaa jäänestojärjestelmän ohjelmiston testausta todellisilla tilatiedoilla ja datanvaihdolla häiriötilanteet huomioiden.

Simuloinnit toteutettiin käyttämällä AXC F 2152 PLC:tä ja siihen liitettäviä tarvittavia I/O-moduuleja. Ohjelmointiin käytettiin PLCnext Engineer -kehitysympäristöä ja structured text -ohjelmointikieltä. Simulointijärjestelmän käyttämistä varten rakennettiin WEB-käyttöliittymä.

Ohjelman vaatimuksena oli mallintaa komponentteja, jotka mahdollistavat normaaliajon lisäksi erilaisten vikatilanteiden simuloinnin käyttöliittymän kautta. Toisena vaatimuksena oli voimalan ja jäänestojärjestelmän välisen datanvaihdon toimiva simulointi.

Tuloksena saatiin järjestelmä, jolla kyettiin simuloimaan komponentteja sekä niihin aiheutuvia vikoja. Modbus-rajapinta mahdollisti voimalan kontrollerin simuloinnin voimalan ja jäänestojärjestelmän välisessä datanvaihdossa. Ympäristöllä on tarkoitus pidemmällä aikavälillä saavuttaa myös rahallista säästöä, kun kaikkia fyysisiä laitteita ei ole tarpeellista hankkia testaustiloihin.

Avainsanat	PLC, Simulointi, Tuulivoimalan jäänesto, Modbus TCP/IP, WIPS, PLCnext, Phoenix Contact

Author Title Number of Pages Date	Jaakko Koski Simulating Anti-icing System Components with Phoenix Contact PLCnext Product Family PLC 37 pages 26 May 2021
Degree	Bachelor of Engineering
Degree Program	Electrical and Automation Engineering
Professional Major	Automation Engineering
Instructors	Tomas Wallenius, Chief Technology Officer Markku Inkinen, Senior Lecturer

In this thesis work Phoenix Contact's PLCnext platform was studied and used to create a system to simulate components and data exchange of the Icing prevention system produced by Wicetec Oy. Purpose was to simulate components and Modbus TCP/IP data exchange as accurately as possible to help with the Icing prevention Systems software testing with true I/O state information and corresponding data traffic.

Simulations were implemented by using AXC F 2152 PLC attached with required I/O modules. Programming was done with PLCnext Engineer development environment and by using structured text programming language. For using the simulation system, WEB user interface was created.

Requirement of the simulation program was to model components in their normal operation as well as to be able to simulate any faults in them through the user interface. Second requirement was to simulate data exchange between the turbine and WIPS controller.

As a result, working simulation system was created and simulation of components and their faults was possible. Modbus interface enabled functional data exchange to simulate turbine's controller and its data exchange to Icing protection systems controller. In long term, this is expected to result in financial savings because acquiring all physical devices to testing environment is not necessary.

Keywords	PLC, Simulation, Wind turbine Icing prevention, Modbus TCP/IP, WIPS, PLCnext, Phoenix Contact

Sisällys

Lyhenteet

BUS	*Power-line communication protocol.* Komponenttien välisen tiedonsiirron sisäinen väyläprotokolla.
ESM	*Execution and Synchronization Manager.* Tehtävien hallinta- ja synkronointijärjelmä.
GDS	*Global Data Space.* Globaalien muuttujien tallennustila.
NC	*Normally closed*. Avautuva kosketin.
OSI	*Open Systems Interconnections Reference Model*. Tiedonsiirtoprotokollien yhdistelmän kerrosten malli.
PDU	*Protocol Data Unit.* Tiedonsiirron yksikön protokolla.
RX	*Receive signal.* Signaalin vastaanotto.
TOF	*Off Delay Timer.* Viiveellä pois kytkevä ajastin.
TON	*On Delay Timer.* Viiveellä päälle kytkevä ajastin.
TX	*Transmit signal.* Signaalin lähetys.
VPN	*Virtual private network.* Virtuaalinen erillisverkko.
WIPS	*Wicetec Ice Prevention System.* Wicetec Oy:n suunnittelema ja valmistama tuulivoimaloiden jäänestojärjestelmä.

1 Johdanto

Tämä insinöörityö on tehty Wicetec Oy:lle, joka suunnittelee ja valmistaa jäänestojärjestelmiä tuulivoimaloihin. Työn tarkoitus on rakentaa simulointiympäristö, jolla voidaan simuloida jäänestojärjestelmän komponentteja sekä järjestelmään integroitavia voimalan osia ja niiden tietoliikennettä.

Simulointiympäristöä on tarkoitus käyttää apuna jäänestojärjestelmän ohjelmistotestauksessa. Komponenttien simulointi on nähty tarpeelliseksi, sillä kaikkia komponentteja ei ole aina mahdollista tai mielekästä hankkia yrityksen omiin toimitiloihin. Järjestelmään liittyy usein myös läheisesti voimalan omia komponentteja rajapintoineen, jotka täytyy huomioida järjestelmän ohjelmiston testauksessa ja joita ei ole suoraan mahdollista hankkia testaustiloihin. Simuloimalla voidaan myös paremmin testata eri vikatilanteiden vaikutusta ohjelmistoon. I/O-rajapinnan ja laitteiden väliseen tiedonsiirron simulointiratkaisuun on päädytty, koska on haluttu käyttää oikeita kenttälaitteita kytkentöineen, niiden lukemia todellisia tilatietoja sekä fyysisten laitteiden välistä dataliikennettä. Kehitysvaiheessa ohjelmistoja testataan aina myös ohjelmallisesti simuloimalla virheenpaikannusympäristössä, mutta näin ei päästä todentamaan fyysisten rajapintojen ja kytkentöjen toimintaa viiveineen ja niiden vaikutusta ohjelmistoon.

Ympäristö suunnitellaan, rakennetaan ja ohjelmoidaan käyttäen Phoenix Contactin Axioline PLC:tä, sille hankittavia tarvittavia I/O-moduuleja, apureleitä sekä johdotuksia. Simulointiympäristö tulee toimimaan suoraan rajapintana testilaboratoriossa oleville jäänestojärjestelmän etä-I/O-laitteille ja tiedonsiirrosta vastaavalle kontrollerille. Kaikki laitteet kytkentöineen sijoitetaan Wicetec Oy:n toimiston testaustiloihin. Simulointialusta suunnitellaan siten, että sitä voidaan tarpeen mukaan laajentaa simuloimaan joko uusi jäänestojärjestelmän komponentteja tai voimalan omia tavalla tai toisella jäänestojärjestelmään liittyviä komponentteja tai osakokonaisuuksia.

2 Jäänestojärjestelmä

Wicetec Oy on vuonna 2014 perustettu suomalainen yritys, joka kehittää ja valmistaa jäänestojärjestelmiä tuulivoimaloihin. Tuulivoimaloiden siipien jäätyminen aiheuttaa pohjoisissa olosuhteissa suuria tuotantotappioita, näiden tappioiden minimointi on Wicetec Oy:n jäänestojärjestelmän eli WIPS:n (engl. Wicetec Ice Prevention System) päätarkoitus. Jäisissä olosuhteissa operointi aiheuttaa itse tuotantotappioiden lisäksi ylimääräistä rasitusta voimalan eri osille. Pahimmillaan voimaloita voidaan joutua pysäyttämään pitkäksikin aikaa ja niiden elinkaari voi lyhentyä merkittävästä siiville, laakereille ja vaihteistolle aiheutuvasta rasituksesta. [1, s. 3.]

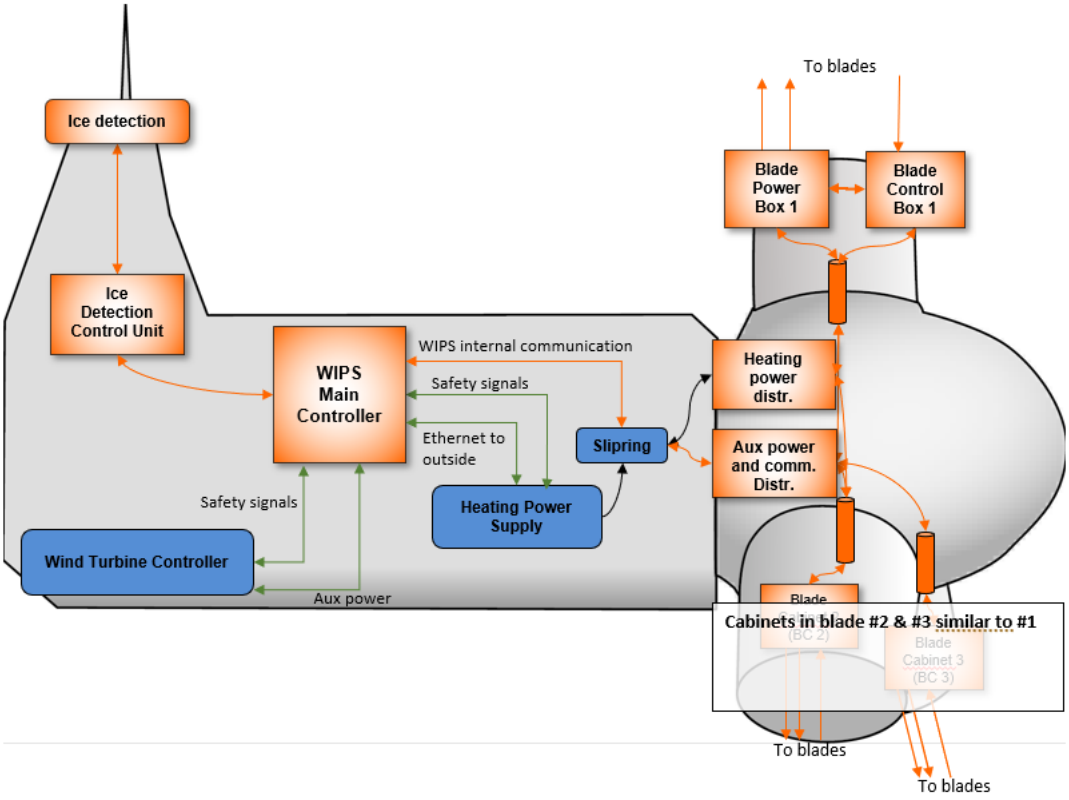
Wicetecin jäänestojärjestelmä perustuu hiilikuidusta valmistettuihin lavan pinnalle kiinnitettäviin lämmityselementteihin. Yksinkertaistettuna siiven elementtejä lämmitetään sähköllä jään syntymisen estämiseksi tai sen poistamiseksi. Jäänestojärjestelmä pystyy operoimaan voimalan ollessa tuotannossa, jolloin jäähdytys on siipiin kohdistuvan ilmavirran takia voimakasta. Oikein aseteltu ja aivan siiven pinnassa sijaitsevalla ohuella elementillä on mahdollistettu lämmityksen kohdistaminen nopeasti suoraan siiven alueille, joissa jään kertymisestä on eniten haittaa siipien aerodynamiikalle [kuva 1]. Ratkaisulla on pyritty kustannustehokkaan lämmittämisen minimoimalla lämmityksessä tarvittavaa energiaa. Tyypillinen järjestelmän energiankulutus on alle 0,3 % voimalan vuotuisesta tuotannosta. [2; 3; 4.]

Kuva 1. Lämmityselementin asemointi tuulivoimalan lavassa [5.]

Jäänestojärjestelmä sisältää huomattavan määrän komponentteja, jotka voivat vaihdella voimalamallin ja järjestelmän revision mukaan. Jäänpoistojärjestelmä sisältää normaalisti vähintään:

- siipien lämmityselementit
- siipien lämpötilamittaukset
- lämmityksen ohjauskeskukset etä-I/O laitteineen
- pääkeskuksen, jossa järjestelmän kontrolleri sijaitsee
- jääntunnistusjärjestelmän.

Järjestelmän osat sijoittuvat nasellin ja navan puolelle sekä itse siipiin. WIPS-järjestelmän osat integroituvat kuvan 2 mukaisesti osaksi voimalaa. WIPS-järjestelmän omat osat on merkattu oranssilla ja voimalan järjestelmään integroitavat osat sinisellä. [5.]



Kuva 2. WIPS-järjestelmän osat [5.]

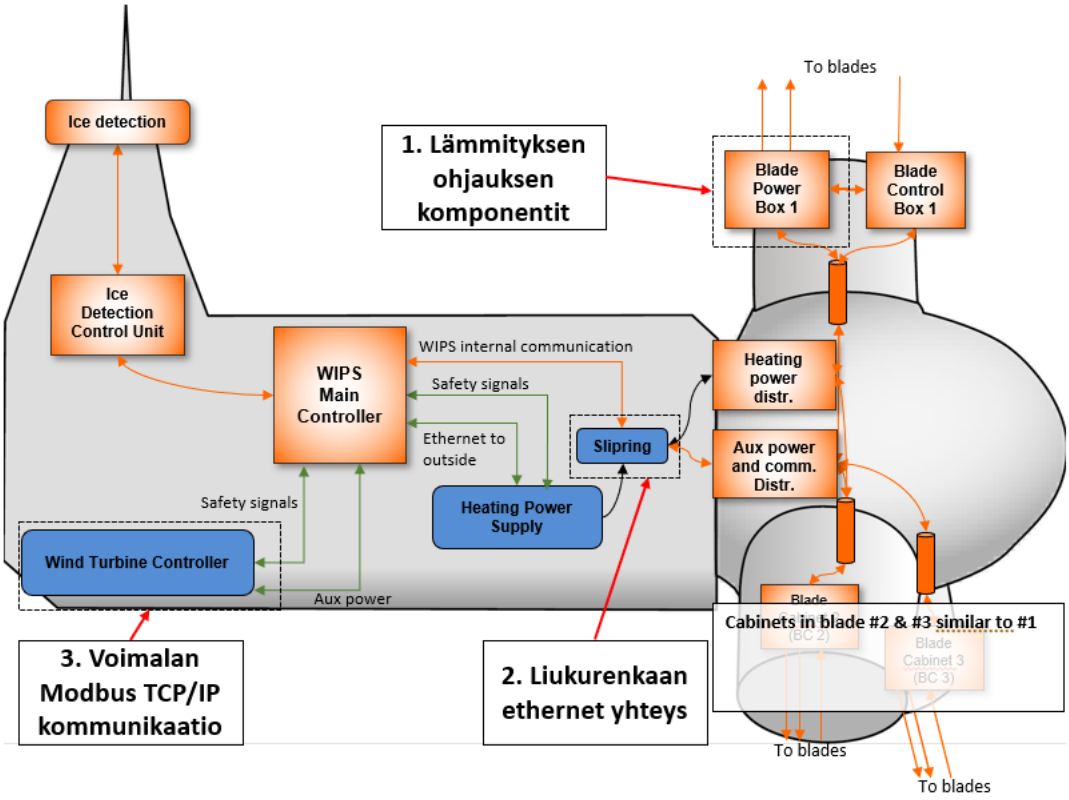
3 Simulointijärjestely ja simuloitavat osat

Simulointiympäristön sijoituspaikaksi oli valittu Wicetec Oy:n toimiston testitila. Tiloista löytyy WIPS-järjestelmän kontrolleri kenttälaitteineen lähelle todellisuutta vastaavalla verkkorakenteella. Tällä kokonaisuudella on mahdollista ajaa WIPS-järjestelmän ohjelmistoa ja testata sen toimintaa pelkällä I/O-rajapinnalla ilman kaikkia järjestelmän komponentteja tai lämmityselementtejä. Työssä toteutettava simulointialusta liittyy suoraan tähän järjestelyyn eli tarkoitus on mahdollistaa komponenttien simulointi I/O-rajapinnalle ja tiedonsiirron osalta mallintaa voimalan ja jäänestojärjestelmän välistä datanvaihtoa. Simulointialustalle oli varattu paikka WIPS-järjestelmän etä-I/O-laitteiden vierestä, jotta kytkentämatkat jäisivät kohtuullisen lyhyiksi, käytännössä aina alle kahteen metriin. Toteutettavaksi valitut simuloinnit keskittyvät järjestelmän kolmeen eri osa-alueeseen [kuva 3]:

1. Lämmityksen ohjauksen komponentit, tässä tapauksessa kontaktorit sekä moottoroidut automaattikatkaisimet, jotka ovat suoraan osa WIPS-järjestelmää.

2. Liukurengas ja sen Ethernet-yhteys, joka on voimalan komponentti ja jonka läpi WIPS:n tietoliikenne nasellista navan puolelle kulkee.

3. Voimalan kontrolleri ja sen Modbus TCP/IP -kommunikaatio WIPS:n kontrollerin kanssa.



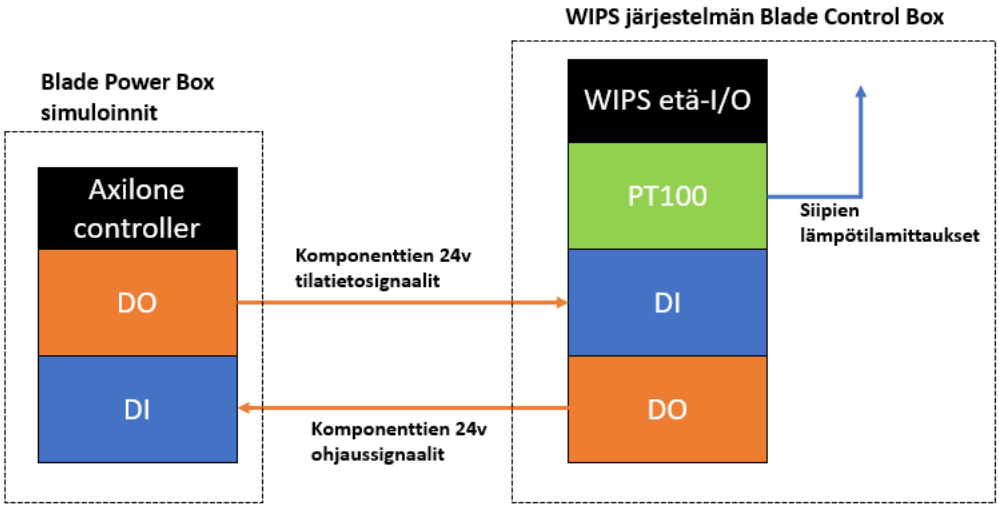
Kuva 3. Simulointitoteutukset osat voimalaan integroidussa jäänestojärjestelmässä [5.]

3.1 Lämmityksen ohjauksen komponentit

Lämmityssähkön ohjaus tapahtuu navan puolelle lähelle siipien tyviä sijoitetuilla Blade Power Boxien komponenteilla, jotka ovat lämmityksen kytkevät kontaktorit sekä turvakomponentteina toimivat moottoroidut automaattikatkaisimet. Nämä saavat digitaaliset ohjaussignaalinsa viereen sijoitetuilta Blade Control Boxeilta, jossa sijaitsevat järjestelmän etä-I/O-laitteet. WIPS-kontrolleri eli jäänestojärjestelmää ohjaava PLC puolestaan kommunikoi etä-I/O-laitteiden kanssa Ethernet-yhteydellä toteutetulla Modbus TCP/I -protokollalla. Yhteys kulkee nasellista liukurenkaan ja navan puolella sijaitsevan kytkimen läpi Blade Control Boxeille.

Blade Power Boxien osalta simuloinnit on rajattu ohjauksen pääkomponentteihin eli kontaktoreihin ja moottoroituihin automaattikatkaisimiin. Lisäksi kokonaisuuteen haluttiin yhdistää referenssilämpötilan mittaus PT100-anturilla. Rajapintana simuloinnissa on etä-

I/O-laitteilta tulevat digitaaliset ohjaussignaalit, joita luetaan simulointijärjestelmän PLC:n DI-moduuleilla sekä vastaavasti DO-moduuleilla etä-I/O-laitteille palautettava tilatiedot [kuva 4].



Kuva 4. Blade Power Box – Blade Control Box -simulointirajapinta.

3.1.1 Kontaktorit

Kontaktorit ovat lämmityksen ohjauksen päälle- ja poiskytkennän pääkomponentti. Jokaista siipeä kohti on yksi tai useampi kontaktori järjestelmän toteutustavasta ja kytkennöistä riippuen, työssä käytetyssä järjestelmässä ainoastaan yksi. Itse kontaktorit ovat toiminnaltaan ja WIPS-logiikan osalta suoraviivaiset: Kunkin siiven etä-I/O kirjoittaa kontaktorin yhdellä outputilla kiinni ja lukee kontaktorin tilan yhdellä inputilla. Vikatilanne syntyy, jos lähdön ja tulon välille syntyy yli määritellyn ajan kestävä ristiriita eli kontaktori ei aukea tai sulkeudu komentojen mukaan määräajassa. Käytetyille kontaktoreille on ilmoitettu sulkuviiveeksi noin 20 ms ja avausviiveeksi alle 15 ms.

3.1.2 Moottoroidut automaattikatkaisimet

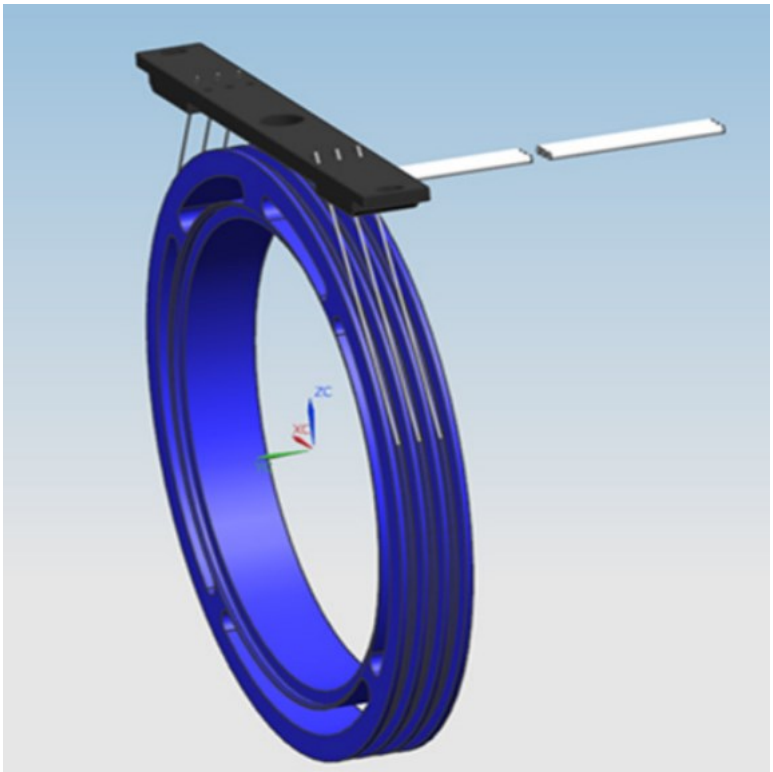
Automaattikatkaisimet (engl. Circuit Breaker) ovat lämmityssähkön ohjauksen ensisijainen turvakomponentti. Katkaisimien roolina on toimia ylivirrasta laukeavana automaattisulakkeena sekä mahdollistaa huoltojen tai muiden toimenpiteiden ajaksi virtapiirin luotettava katkaisu, jota pelkästään kontaktoreilla ei saavutettaisi. Käytetyt katkaisimet on moottoroitu erillisellä lisäkomponentilla eli ne voidaan avata tai sulkea etänä. WIPS:n etä-I/O:t ohjaavat kunkin siiven automaattikatkaisimen moottoria kahdella erillisellä lähdöllä: Sulkukomennon pulssi sekä avauskomennon pulssi, jotka ovat kestoiltaan muutamia sekunteja toiminnan varmistamiseksi. Avauskomennon pulssi on releistyksellä rakennettu määrääväksi ristiriitojen välttämiseksi ja turvallisuuden varmistamiseksi.

Tilatietoina katkaisimilta luetaan yhdellä sisääntulolla sen auki/kiinni-tilatieto ja toisella ylivirtasuojan tila. Ylivirtasuojan lauetessa automaattikatkaisinta ohjaava moottori jää eri asentoon kuin itse katkaisin, eli vikatilanteissa moottorin tila täytyy ensin resetoida ajamalla se auki-asentoon, joka samalla myös resetoi katkaisimen ylivirtasuojan tilatiedon. Komponenttien avausviiveeksi on ilmoitettu 15 ms, joka on turvakomponentin kannalta oleellisin arvo.

3.2 Liukurengas

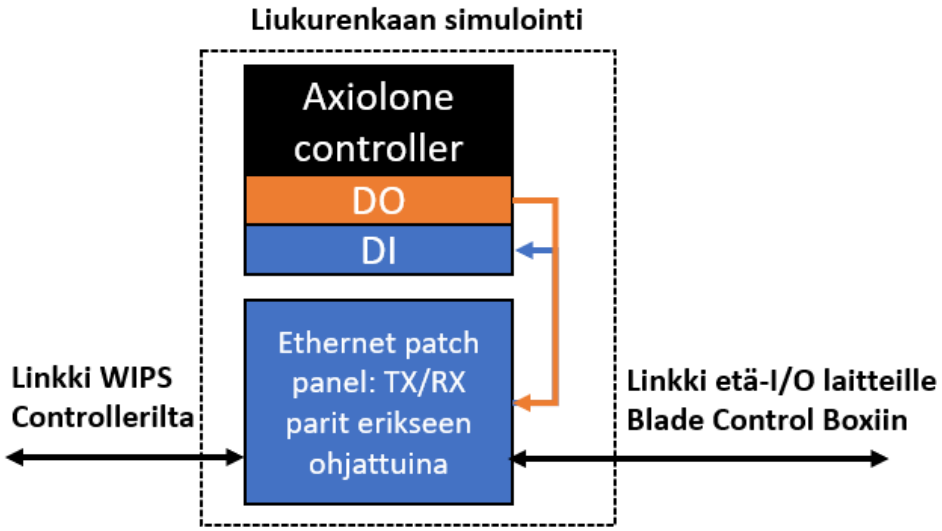
Liukurenkaasta tuulivoimaloiden yhteydessä puhuttaessa tarkoitetaan voimalan paikallaan olevan nasellin ja pyörivän navan yhdyspistettä, jonka kautta kulkee tietoliikenne sekä sähkövirta. Voimalan siipikulmien säätö tapahtuu usein sähköllä, ja sähköä tarvitsevat myös lukuisat mittalaitteet, säätimet, etä-I/O:t yms. Myös kaikki pyörivältä puolelta saatava data pitää lähettää tavalla tai toisella nasellin puolelle ja edelleen valvomoihin. Tämä tapahtuu usein liukurenkaan läpi langallisesti sarjaväylää tai Ethernet-linkkiä käyttämällä.

Liukurenkaissa käytetyt materiaalit ja pinnoitteet vaihtelevat valmistajan mukaan, mutta kaikille renkaille yhteistä on itse pyöreä rengas ja siihen liittyvät harjamaiset, liukuvat kontaktit [kuva 5]. Metallista rengasosaa vasten olevat harjat valmistetaan tavallisesti hiilikuidusta, komposiiteista tai jalometalleista.



Kuva 5. Liukurenkaan signaalien kontaktipinnat [6, s. 2.]

Liukurenkaan osalta päädyttiin simuloimaan fyysistä Ethernet-linkkiä, jonka läpi järjestelmän kaikki tietoliikenne Nasellin puolelta napaan kulkee. Simulointi suunniteltiin järjestettäväksi mahdollistamalla Ethernet-liikenteen TX- ja RX-parien katkaisu Ethernet-kytkentäpaneelin, apureleiden ja simulointialustan kontrollerin avulla [kuva 6]. Näin pyritään mahdollistamaan liukurenkaalla mahdollisesti tapahtuvien Ethernet-linkin katkosten simulointi ja niiden vaikutusten tutkiminen etä-I/O-laitteisiin. Käytännössä järjestelyllä voidaan myös simuloida myös minkä tahansa muun linkin osan, esimerkiksi kytkimen, liittimien tai johtimien aiheuttamia katkoksia. On kuitenkin huomioitava, että häiriöiden osalta suunnitellulla järjestelyllä voidaan simuloida ainoastaan eri pituisia katkoksia, ei suoranaisia jännitehäiriöitä. WIPS-järjestelmässä tällä yhteydellä on suora vaikutus Blade Power Boxin komponenttien ohjaukseen. Näiden komponenttien simulointi liittyy tältä osin tiiviisti yhteen liukurenkaan simulointiin ja näiden voidaankin katsoa toimivan toisiinsa liittyvänä kokonaisuutena. [6.]



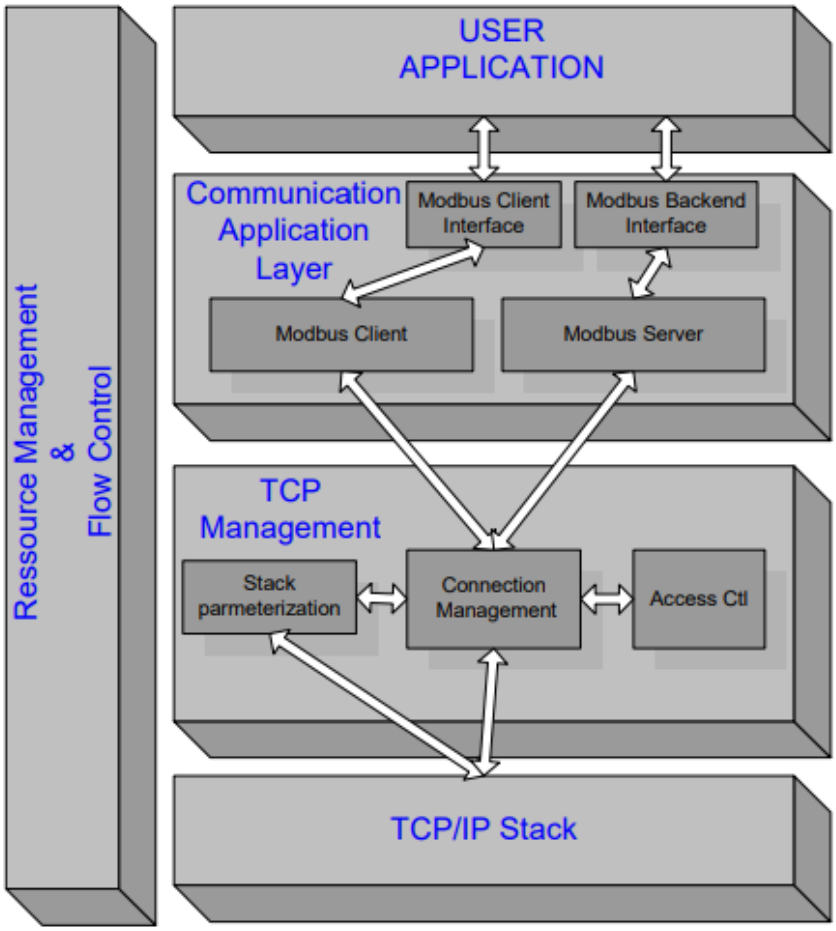
Kuva 6. Liukurenkaan simulointijärjestely.

3.3 Voimalan kontrollerin Modbus TCP/IP-rajapinta.

WIPS-järjestelmä kommunikoi voimalan kontrollerin kanssa ja WIPS:n voidaankin katsoa olevan voimalan alijärjestelmä. Täten voimalan tulee kyetä kontrolloimaan sitä sekä tarkkailemaan WIPS:n tilaa. WIPS-järjestelmän kommunikointitapa ja integraatiotaso riippuvat voimalatyypistä, toteutettavan simuloinnin osalta voimalan ja WIPS-kontrollerin välisessä kommunikaatiossa käytetään Modus TCP/IP -liikennöintiä.

Modbus TCP/IP perustuu jo 1979 julkaistuun Modbus-protokollaan, jota alun perin käytettiin lähinnä sarjaväyläliikennöinnissä. Kun puhutaan pelkästä Modbus-protokollasta, tarkoitetaan sillä sen määrittämää Simple Protocol Data Unit (PDU) -muotoa eli OSI-mallin kuljetuskerroksen informaation sisältävän datapaketin tarkkaa rakennetta. Sen tukemat datatyypit ja komennot ovat siten rajoitetut, vaikkakin OSI-mallin mukainen siirtokerros on Modbus TCP/IP -liikennöinnissä päivitetty sarjaväylästä Ethernetiin ja kuljetuskerros TCP-kehykseen. TCP-kehyksen sisäänrakennetun datan eheyden tarkastuksen ansiosta on Modbusin omasta, sarjaväyläprotokollossa käytetystä redundanssin tarkastuksesta voitu luopua. Modbus TCP/IP -formaatilla ei myöskään enää ole sarjaväylän asettamia nopeuteen ja tuettuihin laitemääriin liittyviä rajoituksia, vaikka kommunikointi

onkin edelleen Client - Server (Master – Slave) hierarkkista ja luku- ja kirjoituskomennot ovat säilyneet samoina kuin sarjaväyläprotokollassa [kuva 7]. [7; 8, s 4–5.]

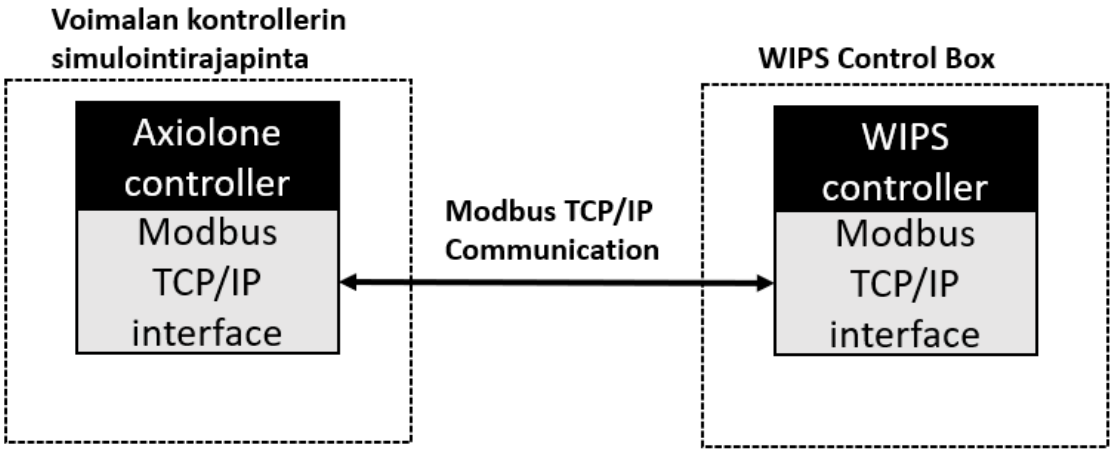


Kuva 7. Modbus TCP/IP- komponenttien arkkitehtuurimalli [9, s. 7.]

Rekisterit ovat aina joko 1 tai 16 bitin luettavia tai luettavia ja kirjoitettavia rekistereitä. Käytännössä tuplarekisterien kokoiset eli 32-bittiset datatyypit ovat yleisesti ohjelmisto-tasolla tuettuja. Yhdellä luku- tai kirjoituskomennolla voidaan kerralla lukea tai kirjoittaa rajattu määrä rekistereitä, esimerkiksi 16 bitin rekistereitä voidaan yhdellä komennolla kirjoittaa enintään 63. Rekisteriavaruus on rajattu enintään 65 536 rekisteriin johtuen standardin 16-bittisestä osoitteistosta. Täten suurten datamäärien tai suurikokoisempien

tiedostojen siirtäminen ei Modbus TCP/IP -protokollaa käyttäen ole käytännöllistä. Protokolla on kuitenkin saavuttanut vakaan aseman teollisessa tiedonsiirrossa ja sen etuna on avoin lähdekoodi verrattuna esimerkiksi PROFINET-protokollaan. [10.]

Suunnitellussa toteutuksessa on tarkoitus simuloida voimalan kontrollerin käyttämiä Modbusin kirjoitus- ja lukukomentoja sekä rajatusti voimalan kontrollerin Modbus-serverin rekisteriä siltä osin kuin WIPS:n kontrolleri lukee siltä tietoa. Tähän on tarkoitus hyödyntäen simulointialustan kontrollerilta vaadittavaa Modbus TCP/IP -tukea ilman erillisiä verkkomoduuleita [kuva 8]. Toteutuksessa käsitellään rajapintaa Modbus-liikenteen, rekistereiden ja kirjoitettavan raakadatan osalta; itse komentoihin tai toiminnallisuuksiin ei perehdytä.



Kuva 8. Simulointialustan kontrollerin ja WIPS-kontrollerin välinen Modbus TCP/IP -yhteys.

4 Simulointijärjestelmän laitteisto

Työtä suunniteltaessa valikoitui Phoenix Contact melko pian kontrollerin toimittajaksi. Tähän vaikuttivat Wicetecin hyvät kokemukset useista kyseisen valmistajan tuotteista ja halu kokeilla uuden PLCnext-tuoteperheen kontrolleria.

4.1 Vaatimusmäärittelyt laitteistolle

Simuloitavien osien ja niiltä vaadittujen toiminnallisuuksien perusteella laadittiin vaatimusmäärittely laitteistolle:

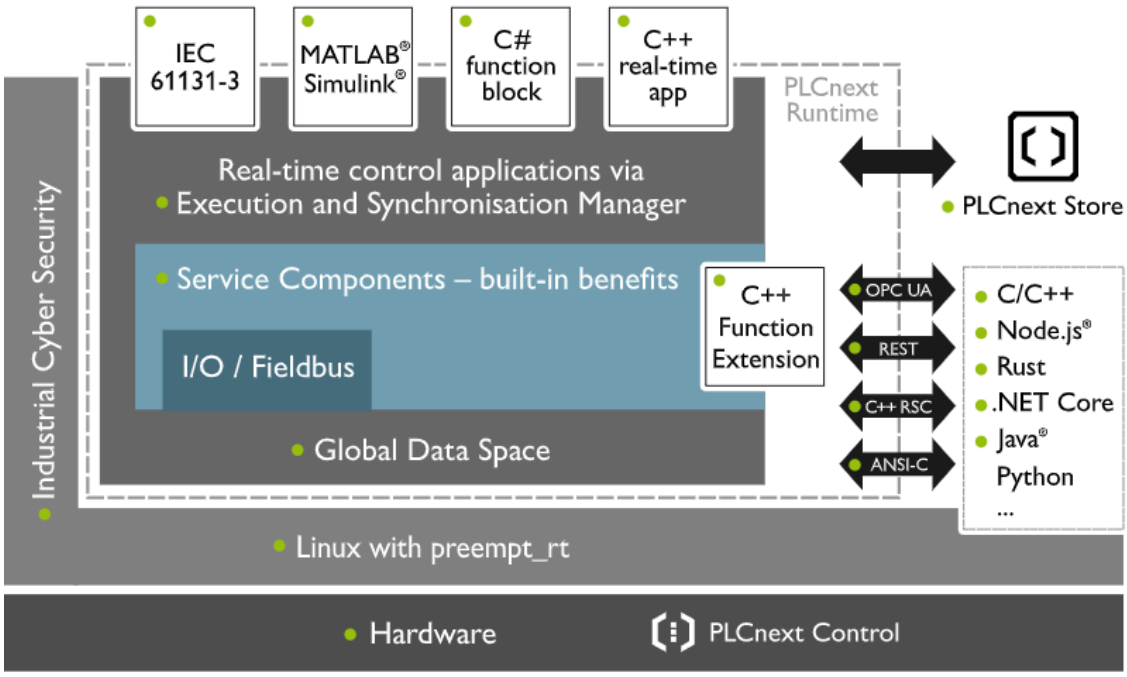
1. minimissään 15 kappaletta digitaalisia lähtöjä ja tuloja (24V, 1-johdin kytkentä)
2. PT100-antureiden lukemisen mahdollisuus
3. tuki Modbus TCP/IP -liikennöinnille
4. etäkäytön mahdollistavat verkko-ominaisuudet
5. valmius WEB-pohjaiseen HMI toteutukseen
6. Ethernet-liikenteen fyysisen katkomisen PLC:n avulla mahdollistavat osat
7. alustan laajennettavuus ja riittävä suorituskyky.

4.2 AXC F 2152

Työn suunnitteluvaiheessa PLCnext-tuoteperheestä ei ollut saatavilla kuin yksi malli: AXC F 2152. Ominaisuuksiltaan kyseinen kontrolleri yhdistettynä mukana tulevaan ohjelmointiympäristöön ja tuettuihin I/O-moduuleihin osoittautui tutkinnan jälkeen käyttötarpeeseen sopivaksi ja täyttävän vaatimusmäärittelyt.

Simulointijärjestelmän toteuttamiseen valittu kontrolleri edustaa Phoenix Contactin uutta PLCnext-sukupolvea. Valittu kontrolleri pohjaa perusrakenteeltaan edellisen sukupolven Axiocontrol-sarjaan. Kontrollerista on suunnittelun ja työn tekemisen aikana julkaistu eri versioita, joissa vaihtelevat tuetut ominaisuudet, suorittimen ydinten määrä sekä säänkesto. AXC F 2152 edustaa suorituskyvyn osalta keskilinjaa. Kontrolleri käyttää samaa Axioline F local bus -väylää kuin edeltäjänsä ja tukee täten kaikkia Axioline-sarjan komponentteja, joiden valikoima on jo kasvanut laajaksi. Yhteensä väylä mahdollistaa 64:n I/O-moduulin kytkemisen ja ohjaamisen yhdellä kontrollerilla. Ulkonäöltään kontrolli ei käytännössä eroa edellisen sukupolven Axiocontrol AXC 1050 PLC:stä. [11, s. 13.]

Päivitetyn hardwaden lisäksi suurin käyttäjälle näkyvä ero edelliseen sukupolveen on käyttöjärjestelmän vaihtuminen Linux RT -järjestelmään sekä ohjelmointiympäristön vaihtuminen PC WORX -ympäristöstä PLCnext Engineering -ympäristöön. PLCnext Engineering on lisenssimaksuton kehitysohjelmisto, jota käytetään PLC:n I/O-moduuleiden konfigurointiin, ohjelmointiin sekä HMI-toteutusten tekemiseen. Alusta tukee edeltäjästään poiketen lukuisten ohjelmointikielien käyttöä [kuva 9]. Ohjelmisto on perustoiminnallisuuksiltaan verrattavissa Siemensin TIA-portaaliin ollen kuitenkin paljon kevyempi ja monelta osin suppeampi. [11, s. 13.]



Kuva 9. PLCnext-rajapinnat [12.]

PLCnext-kontrollerit perustuvat Real-Time Linuxiin ja tarkemmin sen PREEMPT_RT-versioon. Tavallisista käyttöjärjestelmistä poiketen reaaliaikaisella käyttöjärjestelmällä kaikkien prosessien täytyy olla aikarajoitteisia. Aikarajoitteisuus on käytännössä vaatimus kaikille teollisuudessa käytettäville ohjelmoitaville logiikoille. Prosesseilla on myös prioriteetteihin perustuva hierarkia, jolloin tärkeät prosessit suoritetaan juuri niin nopeasti kuin laskentateho sallii muista prosesseista välittämättä. Tämä mahdollistaa lyhyet latenssit ohjelmiston kriittisissä osissa. Trendinä PLC-maailmassa onkin ollut havaittavissa siirtyminen suljetuista järjestelmistä kohti Linux-pohjaisia ratkaisuja. Phoenix Contactin lisäksi ainakin Wago on julkaissut oman PREEMPT_RT-versioon pohjaavan tuoteperheen. Myös Siemens on julkaissut Debianiin perustuvan SIMATIC Industrial OS -version, joka on yhteensopiva useiden SIMATIC IPC -laitteiden kanssa. Etuna avoimemmissa Linux-pohjaisissa PLC-ratkaisuissa on laajempi tuki eri ohjelmointikielille ja eri rajapinnoille. [13.]

AXC F 2152 tekniset tiedot:

- suoritin: Arm Cortex-A9 2x 800MHz
- keskusmuisti: 512Mb DDR3 SDRAM
- local bus: Axioline F – tuki 64 moduulille
- verkkoliitännät: 2x Ethernet RJ45
- tuki Modbus TCP/IP-liikennöinnille erikseen ladattavilla kirjastoilla
- kenttälaitteiden päivitysnopeus: 4 laitetta:1ms; 64(maksimi) laitetta, 16ms
- käyttöjärjestelmä: Linux PREEMPT_RT.

4.3 I/O-moduulit

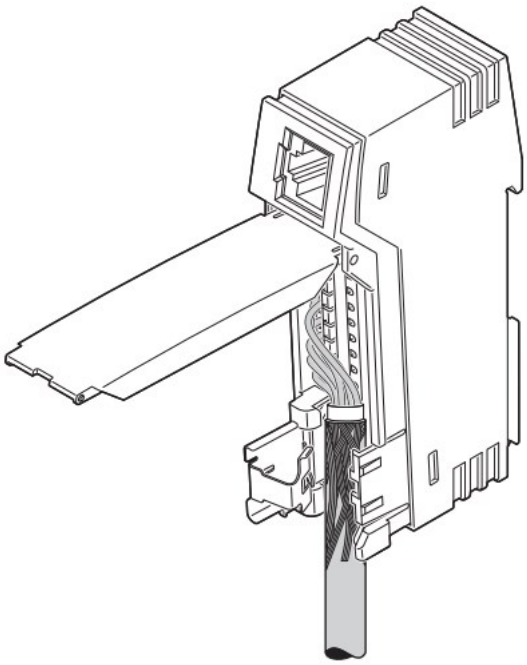
I/O-moduulien osalta valittiin vaatimusmäärittelyn perusteella riittävät osat. Simulointijärjestelmän käyttöpaikka on sisällä toimistotiloissa, joten tarvetta äärilämpötiloja tai värinää kestäville moduuleille ei ollut. Valitut osat ovat luokkansa edullisimpia mutta käyttötarkoitukseen riittäviä. Valitut input- sekä output-moduulit ovat 16-paikkaisia. Täten ylimääräisiä input- tai output-paikkoja ei juuri ole. Myöhemmin järjestelmää mahdollisesti laajennettaessa ei moduulien lisääminen kontrollerin perään kuitenkaan tuottaisi ongelmia. Molemmat moduulit käyttävät 1-johdin-kytkentätapaa. PT100-moduulin osalta valittiin edelleen edullisin neljää anturia tukeva moduuli, joka tukee sekä 2-, 3- tai 4-johdinkytkentöjä. Simulointijärjestelmän osalta varauduttiin käyttämään 3-johdinkytkentöjä.

Valitut I/O-moduulit:

- Digital output -moduuli: AXL F DO16/1 1H
- Digital input -moduuli: AXL F DI16/1 1H
- PT100-moduuli: AXL F RTD4 1H.

4.4 Ethernet-kytkentäpaneeli ja apureleet

Liukurenkaan simulointia varten saatiin käyttöön Phoenix Contactin PP-RJ-SCC-mallinen Ethernet-kytkentäpaneeli (engl. Ethernet Patch Panel) [kuva 10] apureleineen, joita voidaan kontrolloida simulointijärjestelmän kontrollerin digitaalilähdöillä. Paneeli on tarkoitettu Ethernet-kaapeloinnin yhdyspisteeksi eli sillä voidaan yhdistää teollisen Ethernet-kenttäväylän signaalit yksitellen tavalliselle Ethernet-kaapeloinnille sopivaksi. Ethernet-liikenne suunniteltiin ohjattavaksi releiden ja paneelin läpi johdin kerrallaan. Jäänestojärjestelmässä käytetyn Modbus TCP/IP:n osalta tämä tarkoitti RX- sekä TX-parien käyttöä. Liikenteen ohjaus suunniteltiin siten, että pareja on mahdollista kontrolloida releillä erikseen eli simuloida joko toisen tai molempien parien katkoksia.



Kuva 10. PP-RJ-SCC Ethernet-kytkentäpaneeli [16.]

4.5 Asennukset ja käyttöönotto

Hankittu Axioline PLC oli Wicetecille täysin uusi tuote. Tästä johtuen apua yrityksen sisältä ei ollut saataville eli kaiken tiedon etsintä tapahtui pitkälti manuaaleja kahlaamalla. PLCnext-alustan foorumeilta löytyi jonkin verran käyttöönottoa helpottavaa materiaalia, vaikkakin alustan ympärille muodostunut yhteisö oli vielä hyvin pieni verrattuna jo laajalti käytössä olevien tuotteiden vastaaviin. Phoenix Contactin manuaalit osoittautuivat kuitenkin erittäin kattaviksi ja hyviksi. Tämä oli havaittu yrityksen sisällä jo muidenkin jo pidempään käytössä olleiden kyseisen yrityksen tuotteiden kohdalla.

Axioline PLC:n asennus

AXC F 2152 asennettiin sille testiympäristöstä varatulle paikalle DIN-kiskoon. Ennen kontrolleria kiinnitettiin kiskoon BUS-väylämoduulit PLC:n ja sen I/O-moduulien välistä

sisäistä tiedonsiirtoa varten. BUS-väylä jakaa moduuleille myös niiden tarvitseman sisäisen 5V:n käyttöjännitteen. PLC ja DI-, DO- sekä PT100-moduulien kiinnitys väylään tapahtui vaivattomasti ilman työkaluja. I/O-moduuleille täytyi vielä BUS-väylän tarjoaman sisäisen käyttöjännitteen lisäksi johdottaa kontrollerilta 24V:n ulkoinen käyttöjännite. Itse PLC:lle kytkettiin myös 24V:n käyttöjännite testiympäristön virtalähteeltä. Verkkoyhteyden osalta kontrolleri liitettiin Ethernet-kaapelilla testiympäristön verkkoon kytkimen kautta. Tämä mahdollistaisi niin Modbus TCP/IP -kommunikoinnin WIPS-kontrollerin kanssa kuin myös suunnitellun simulointijärjestelmän WEB GUI:n etäkäytön.

PLC:n firmware päivitettiin heti asennuksen yhteydessä uusimpaan julkaistuun. Päivitysprosessi suoritettiin suoraan Linux RT:n komentoriviltä. Päivitys oli pakollinen, sillä uusin PLCnext Engineering -ohjelmointiympäristö vaati toimiakseen uusimman firmwaren. Lisäksi asetettiin järjestelmälle oikea aikavyöhyketiedosto. Asetuksissa päädyttiin käyttämään Itä-Euroopan aikavyöhykettä (engl. EET, Eastern European Time) ilman kesäaikaa. Tätä perusteltiin sillä, että kesäajan käyttö voi olla ongelmallista etenkin tiedonkeruun kannalta sekä eri maiden asiakkaiden kanssa työskennellessä.

Johdotukset ja Ethernet-kytkentäpaneelin asennus

Jäänestojärjestelmän etä-I/O-rajapinnan lähdöt ja tulot löytyivät suoraan järjestelmän sähkökuvista. Ne johdotettiin johtokourua pitkin ja kytkettiin valmiiksi I/O-moduuleille [kuva 11] siten, että WIPS:n etä-I/O:n lähdöt kytkettiin Axioline-kontrollerin input-moduuleihin ja tulot output-moduuleihin.

Kuva 11. Axioline PLC:n digitaaliset lähdöt ja tulot.

Ethernet-kytkentäpaneeli apureleineen asennettiin lähelle WIPS:n etä-I/O-moduuleja turhien pitkien Ethernet-johdotusten välttämiseksi. Apureleet johdotettiin suoraan Axioline PLC:n DO-moduulille ja Ethernet-kaapelin RX- ja TX-parit kierrätettiin releiden NC-koskettimien kautta paneelille [kuva 12]. Koska käytössä olevat DO-kortit ovat 1-wire-mallisia eli lähdöissä ei ole V-johdinta, johdotettiin apureleiden tarvitsema V-DO-moduulilta löytyvältä korttien apuvirtojen ketjuttamisen tarkoitetulta liittimeltä. Kaikki kytkennät kirjattiin ylös tulevaa ohjelmointivaihetta varten.

Kuva 12. Ethernet-kytkentäpaneelin apureleet.

5 Simulointiohjelma

Asennusten jälkeen alkoi tutustuminen itse Axioline-kontrollerin PLCnext Engineering -ohjelmointiympäristöön. Dokumentaatioiden lukemiseen käytettiin huomattava määrä tunteja. Samalla toteutettiin testiohjelma, jonka avulla tutustuttiin ympäristöön, testattiin I/O-moduulien toiminta sekä HMI-toimintojen rakentaminen. Alusta erosikin huomattavasti opintojen aikana käytetyistä Siemensin, Omronin tai Beckhoffin vastaavista.

WIPS-järjestelmä ja sen etä-I/O laitteet olivat ohjelmointityön aikana käytettävissä, mikä mahdollisti jatkuvan testauksen. Myös keskuksia oikeilla komponenteilla oli rajatusti käytettävissä. Tätä hyödynnettiin vertaamalla simulointirajapinnan käyttäytymistä todellisiin komponentteihin.

Simulointeihin käytetty ohjelmisto toteutettiin IEC 61131-3 -standardin Structured Text -ohjelmointikielellä. Kirjastoina käytettiin PLC:n omia vakiokirjastoja sekä PLCnext Storesta ladattuja, Phoenix Contactin tekemiä Modbus TCP/IP -kirjastoja. Ohjelma pyrittiin jakamaan mahdollisimman pieniin ja loogisiin aliohjelmiin (enlg. POU, program organization unit) kokonaisuuden hallinnan helpottamiseksi.

5.1 Ohjelman vaatimusmäärittely

Vaatimusmäärittely laadittiin simulointiin valittujen komponenttien ja rapintojen perusteella. Määrittelyssä otettiin huomioon myös mahdolliset ympäristön laajennukset sekä etäkäytettävyys. Vaatimusmäärittelyä laadittiin tarkoituksella suuntaa antavaksi ja sitä varauduttiin täydentämään projektin edetessä:

- Simuloitujen komponenttien tulee kyetä vastaamaan digitaalisiin sisääntuloihin oikeilla digitaalisilla ulostuloilla ja oikea-aikaisesti simuloitaessa komponentin normaalia toimintaa.

- Komponenttien tiloja täytyy voida tarkkailla helposti graafiselta käyttöliittymältä.

- Mahdollisuus simuloida mitä tahansa vikatilaa eli pakottaa simuloidulle komponentille halutun vian määräämä ulostulo käyttöliittymän kautta.

- Käyttöliittymän kautta tulee kyetä lähettämään Modbus TCP/IP -kirjoitus- ja lukukomentoja WIPS-kontrollerille.

- Kontrollerilla tulee olla Modbus TCP/IP -serveri, jolta WIPS-kontrolleri voi lukea halutun muotoista dataa.

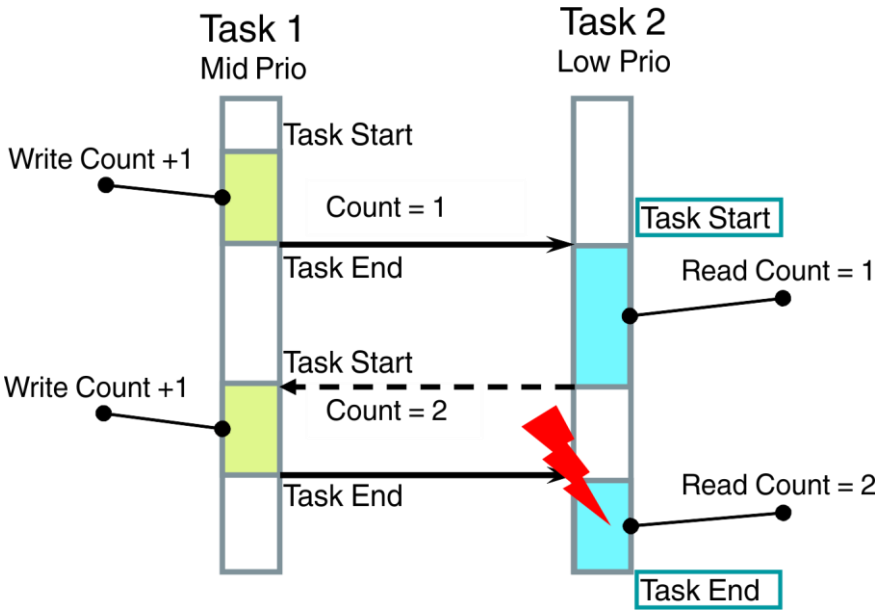
- Ohjelmisto tulee rakentaa siten, että se on modulaarinen ja helposti laajennettavissa.

- Käyttöliittymän on oltava WEB-pohjainen ja sen täytyy tukea etäkäyttöä.

5.2 I/O-moduulien konfigurointi ja Global Data Space

Ohjelmointityö aloitettiin asentamalla PLCnext Engineering -ympäristön viimeisin vakaa versio. Projektipohjan luomisen jälkeen määriteltiin Axioline kontrolleriin lisättyjen I/O-moduulien tarkat mallit ja ohjelmistoversiot, jotta ne tunnistuisivat oikein projektille. Tämän jälkeen päästiin tarkastelemaan korttien lähtöjä sekä testaamaan niiden toiminta. PT100-kortille määritettiin käytettävien antureiden tyyppi, tässä tapauksessa 3-johtoiset anturit, R0 resistanssi 100, resoluutio 0.01°C/0.01Ω sekä tyyppi Pt DIN.

PLCnext-kontrollereiden PREEMPT_RT-alustan päällä toimivat Execution and Synchronizaion Manager (ESM) sekä Global Data Space (GDS). GDS pitää sisällään ohjelman reaaliaikaisuuteen sidotut, globaalit muuttujat. Globaalit muuttujat mahdollistavat matalamman prioriteetin prosessien eheän ja johdonmukaisen suorittamisen keskeytyksistä huolimatta [kuva 13]. Käytännössä GDS tallentaa välimuistiin muuttujien arvoja, joita prosessit keskeytysten jälkeen käyttävät. Ilman tätä ominaisuutta matalamman prioriteetin prosessien inputit voisivat keskeytysten takia muuttua kesken syklin suorittamisen ja lopputulokset olisivat epäjohdonmukaisia. Muuttujia hallitaan erillisellä IN/OUT-porttien listauksella, jolla määritetään muuttujan lähde ja kohde. IN/OUT-portteja voidaan määrittää suoraan I/O-moduuleille mutta myös eri aliohjelmien välille.



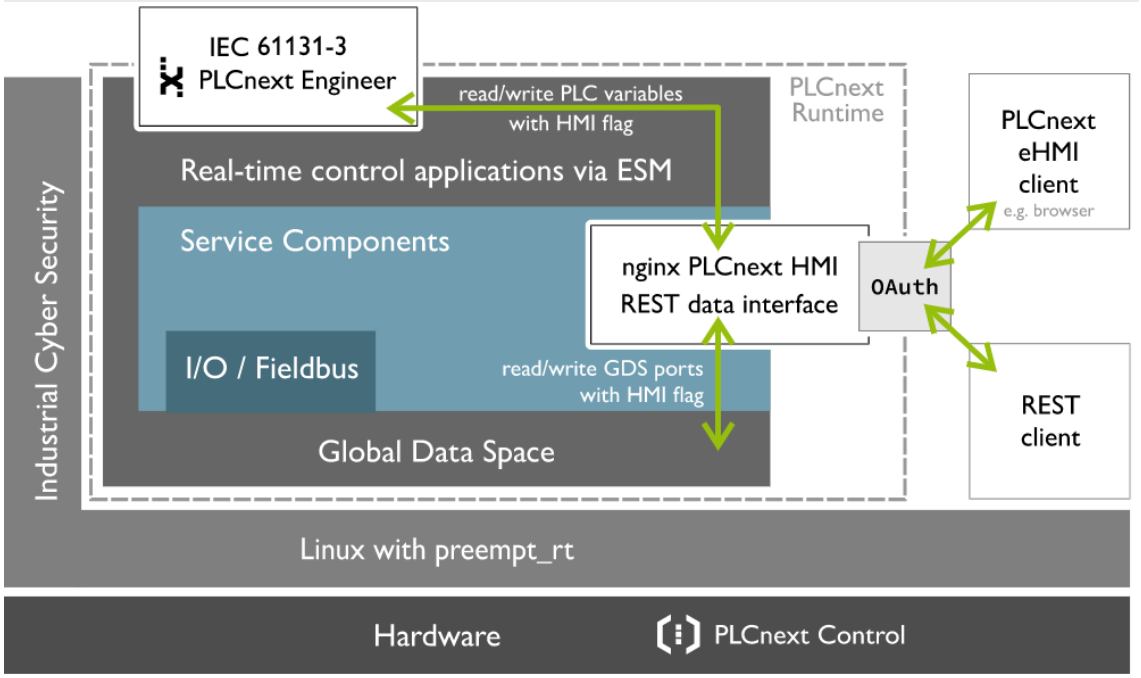
Kuva 13. GDS-toimintamalli alhaisemman prioriteetin tehtävän keskeytyessä [17.]

Portit määritettiin jo tehtyjen ja merkattujen kytkentöjen mukaan. Käytännössä ensin luotiin komponenttikohtaiset aliohjelmat, jonka sisälle luotiin muuttujat, joiden tyypiksi määritettiin IN-portti digitaalisille sisääntuloille sekä PT100-antureille ja vastaavasti OUT-portti digitaalisille lähdöille. Aliohjelmien muuttujien tyyppien määrityksen jälkeen muuttujat asetettiin GDS-järjestelmälle erillisellä Port List -välilehdellä halutuille I/O-moduulien tuloille ja lähdöille.

5.3 Käyttöliittymäratkaisu ja käyttäjähallinta

Ennen varsinaisen ohjelman kirjoitusta perehdyttiin mahdollisiin käyttöliittymäratkaisuihin. Käyttöliittymälle oli vaatimuksena WEB-pohjainen toteutus sen monipuolisuuden ja helpon etäkäytettävyyden takia. Axioline PLC:n sisäänrakennetut palvelukomponentit sisältävät kaiken tarvittavan web-serveristä alkaen. Graafisen käyttöliittymän rakentaminen päätettiin toteuttaa PLCnext Engineering -ohjelmalla, aivan kuten ohjelmointikin. Sisäänrakennetut kirjastot osoittautuivat tutustumien perusteella riittäviksi käyttöliittymän rakentamiseen, vaikka saatavilla olevien graafisten elementtien määrä olikin rajallinen.

Vaihtoehtoisiakin toteutustapoja olisi ollut PLCnext-ympäristön tukiessa lukuisia eri ohjelmointikieliä ja rapintoja, mukaan lukien REST-rajapintaa, jolle myös PLCnextin oma HMI-ratkaisu perustuu [kuva 14]. Tämä olisi mahdollistanut esimerkiksi avoimen lähdekoodin React JavaScript -kirjastojen käytön. Vaikka projektia aloitettaessa valittiinkin kontrollerin integroitu HMI-ratkaisu, on tulevaisuudessa siirtyminen laajempiin avoimen lähdekoodin kirjastoihin mahdollista itse Backendiä muuttamatta. Tämän mahdollistaa GDS-arkkitehtuuri, jolla voidaan jakaa dataa eri rajapinnoille.



Kuva 14. PLCnext REST data interface [18.]

WEB-käyttöliittymän käyttäjähallinnan toteutuksessa käytettiin PLC:n sisäänrakennettuja käyttäjäluokkia. Luokilla on mahdollista eritellä eri tasoisia oikeuksia omille tunnuksille ja näin rajata käyttöliittymän toimintoja sallimalla tietylle luokalle esimerkiksi vain luku oikeus. Alustan toteutuksessa luotiin aluksi yksi yleinen, Admin-tasoinen tunnus, jolla HMI:n käyttöä ei ole rajattu, mutta tunnus itsessään on rajattu ainoastaan käyttöliittymän käyttöön eli kontrollerin käyttöjärjestelmään pääsy on rajattu ulos.

Käyttöliittymän ulkoasun ja käytettävyyden suhteen pyrittiin yksinkertaiseen ja toimivaan ratkaisuun. Siipien komponentit päädyttiin rajaamaan omalle sivulleen, samoin liukurenkaan simuloinnin hallinta sekä Modbus-rajapinnan käyttö. Simulointialustaa ei käytetä yrityksen ulkopuolella ja kaikki sen mahdolliset käyttäjät ovat teknisesti osaavia. Täten liittymän sisäistä ohjeistusta ei nähty tarpeelliseksi toteuttaa.

5.4 Kontaktorit, moottoroidut automaattikatkaisimet ja PT100-anturi

Lämmityksen ohjauksen peruskomponenttien osalta toteutus oli suoraviivaista ja perustui ns. interlocks-määrittelyyn huomioiden WIPS:n etä-I/O-laitteiden ohjaussignaalit sekä simulointiohjelman käyttöliittymän kautta annetut komennot. Periaatteena on, että simuloitu komponentti reagoi WIPS:n etä-I/O-ohjaussignaaleihin komponentin normaalia toimintaa vastaavalla tavalla, jos simulointiympäristön käyttäjä ei ole muuta komentanut [esimerkkikoodi 1]. Käyttöliittymän kautta annetut pakotukset määräävät aina suoraan lähtöjen tilan. Komponenteille luotiin omat aliohjelmansa, jonka sisälle luotiin I/O-moduulien tulo- ja lähtöporttien lisäksi komponenttien ohjelman sisäiset tilatiedot sekä HMI-ohjauksen muuttujat. Käytännössä simulointirajapinnan digitaalisten lähtöjen tilatiedot asetettiin vastaamaan ohjelman sisäisiä tilatietoja, jotka logiikka muodostaa WIPS:n ohjaussignaalien ja HMI-muuttujien perusteella. Aliohjelmien alkuun asetettiin alustuslogiikka, joka suoritetaan kerran ohjelmaa käynnistettäessä ja jolla asetetaan lähdöt komponentin normaaliin auki-tilaan. Tämä oli tarpeellista, sillä osa komponenttien signaaleista tulee avautuvilta koskettimilta.

```
IF X_CB1_Close = TRUE AND Q_CB1_state_tripped = TRUE THEN
    Q_CB1_state_closed := TRUE;
    QH_CB1_state_closed := TRUE;
    END_IF

IF Q_CB1_state_closed = TRUE AND X_CB1_Open = TRUE OR XH_CB1_Force_Open = TRUE
AND XH_CB1_Force_Close = FALSE THEN
    Q_CB1_state_closed := FALSE;
    QH_CB1_state_closed := FALSE;
    END_IF

IF Q_CB1_state_closed AND XH_CB1_Tripp_HMI = TRUE  AND XH_CB1_Force_Close =
FALSE THEN
    Q_CB1_state_closed := FALSE;
    QH_CB1_state_closed := FALSE;
    Q_CB1_state_tripped := FALSE;
    QH_CB1_State_Tripped_HMI := TRUE;
    END_IF
```

```
IF X_CB1_Open OR XH_CB1_Force_Open = TRUE AND Q_CB1_state_tripped = FALSE THEN
    Q_CB1_state_tripped := TRUE;
    QH_CB1_State_Tripped_HMI := FALSE;
    END_IF

IF XH_CB1_Force_Close = TRUE AND Q_CB1_state_tripped = TRUE THEN
    Q_CB1_state_closed := TRUE;
    QH_CB1_state_closed := TRUE;
    END_IF
```

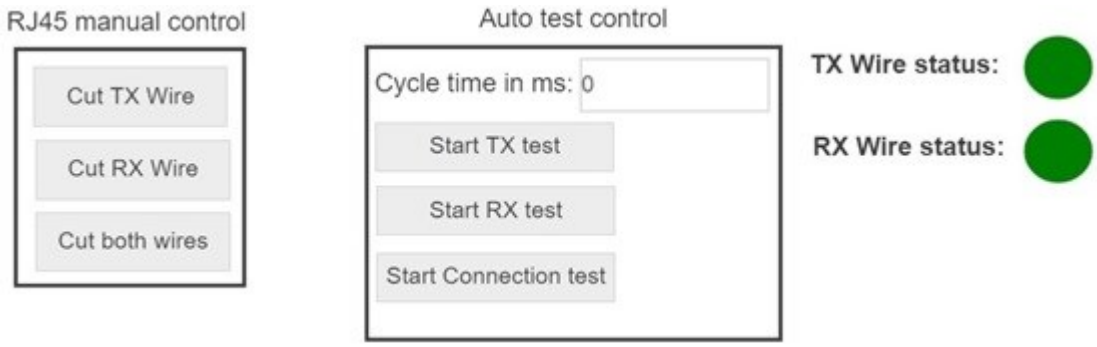
Esimerkkikoodi 1. Yhden automaattikatkaisimen ohjelmointitoteutus.

Kontaktorien osalta toteutus oli vastaava, mutta yksinkertaisempi. Lämmityksen ohjauksen komponenttien tilatiedot ja pakotuskomennot kasattiin yhdelle HMI-sovelluksen sivulle. Lisäksi näkyviin laitettiin referenssiksi yhden PT100-anturin lämpötila [kuva 15]. Anturin PT100-moduulilta luettu lämpötiladata käsiteltiin ennen HMI-näytölle viemistä heksadesimaalimuodosta kokonaisluvuksi ja edelleen liukuluvuksi. Liukuluku jaettiin sadalla, jolloin saadaan anturin mittaama todellinen lämpötila kahden desimaalin tarkkuudella. Muutokset onnistuivat vakiokirjastojen datatyypin muunnosfunktioilla.

Kuva 15. Lämmityksen ohjauksen komponenttien HMI-toteutus.

5.5 Liukurengas

Liukurenkaan Ethernet yhteyden RX- ja TX-parien kontrolloinnin ohjaukseen toteutettiin aliohjelma, jolla käyttöliittymän kautta voidaan asettaa joko toiselle tai molemmille pareille kerralla yhteys päällä/poikki -syklin aika-arvo millisekunteina [kuva 16]. Tämän jälkeen, kun testiohjelma käynnistetään painonapista, toistuvat syklit, kunnes ohjelma pysäytetään. Syklien aika-arvoja voi muuttaa kesken ohjelman ajon. Käyttöliittymän kautta on myös mahdollista ohjata käsin RX/TX-parin toimintaan eli katkoa yhteyksiä manuaalisesti painonappeja painamalla.



Kuva 16. Liukurenkaan Ethernet-yhteyden HMI-toteutus.

Syklinen testiohjelma toteutettiin TON- ja TOF-ajastimilla, jotka hakevat syklien arvot HMI:n muuttujilta kokonaislukuina ja konvertoivat ne ajastimien vaatimaan aikaformaattiin [esimerkkikoodi 2].

```
IF XH_Timer_TX_Start = TRUE AND XH_Timer_RX_Start = FALSE AND
XH_Timer_RJ45Connection_Start = FALSE THEN

      TX_TIMERVALUE := INT_TO_TIME(XH_TXTIMERVALUE);
      TX_TIMERVALUE1 := TX_TIMERVALUE;
      TX_TIMERVALUE2 := TX_TIMERVALUE;

TOF1(IN := TX_TON_OUTPUT, PT := TX_TIMERVALUE1, Q => TX_TOF_OUTPUT, ET =>
TX_TOF_ELAPSED);
Timer_TX_Close := TX_TOF_OUTPUT;

TON1(IN := NOT TX_TOF_OUTPUT, PT := TX_TIMERVALUE2, Q => TX_TON_OUTPUT, ET =>
TX_TON_ELAPSED)
END_IF
```

Esimerkkikoodi 2. TX johdinparin yhteyden ajastimien ohjelmointitoteutus.

5.6 Voimalan kontrollerin Modbus TCP/IP -rajapinta

Jo ennakolta oli tiedossa, että voimalan kontrollerin simulointia varten Modbus TCP/IP -kirjastot pitäisi ladata erikseen PLCnext Storesta, sillä ne eivät kuulu mukana tuleviin vakiokirjastoihin. PLCnext Storen tarkoitus on tarjota rajapinta, jonka kautta niin Phoenix Contact kuin kolmannenkin osapuolen edustajat voivat tarjota sekä maksuttomia että maksullisia valmiita ohjelmia tai ohjelmointikirjastoja muiden saataville. Tällä lähestymistavalla PLC:n mukana ei vakiona tarvitse toimittaa kaikkea mahdollista, vaan käyttäjä voi tarpeensa mukaan valikoida haluamansa lisäosat. Phoenix Contact onkin tuonut saataville runsaasti ilmaisia laajennuksia niin tiedonsiirtoon, datankeräykseen, eri tietotyyppien käsittelyyn kuin logiikoiden ohjelmoinnissa käytettäviin valmiisiin funktioihin liittyen. Simulointiohjelman voimalakontrollerin toteutusta varten ladattiin ja käytettiin Modbus_TCP_6-kirjastoa, joka on kattava Phoenix Contactin julkaisema maksuton Modbus TCP/IP -kirjasto. Ilman tarjolla olevia valmiita kirjastoja olisi ollut mahdollista tehdä itse kirjastot vakiona mukana tulevien TCP/IP-kirjastojen varaan, mutta tämä olisi tarkoittanut huomattavaa määrää lisätyötä. [19; 20.]

5.6.1 Modbus Serveri

Modbus Serveri toteutettiin ladattujen kirjastojen MB_TCP_Server-funktiolla [esimerkkikoodi 3].

```
MB_TCP_Server_3(xActivate := BOOL, xAcknowledge := BOOL, xAutoAck := BOOL,
xUDP := BOOL, strBindIp := STRING, uiBindPort := UINT, strDestIp := STRING,
uiDestPort := UINT, tReconnectDelay := TIME, tTimeout := TIME,
uiOffsetInputRegister := UINT, uiOffsetHoldingRegister := UINT, uiOffsetInputs
:= UINT, uiOffsetCoils := UINT, xActive => BOOL, xConnected => BOOL, xError =>
BOOL, wDiagCode => WORD, wAddDiagCode => WORD, udtDiag => MB_TCP_UDT_SER_DIAG,
arrModbusData := MB_TCP_ARR_W_0_7167);
```

Esimerkkikoodi 3. Toteutuksessa käytetty Modbus Server -funktio, esimerkin funktion muuttujat nimettynä niiden datatyypeillä.

Serveri asetettiin käynnistymään automaattisesti ohjelman käynnistyessä ja käyttöliittymään lisättiin serverin antamat diagnostiikkakoodit sekä mahdollisuus sulkea ja käynnistää serveri.

Serverin virheidenkäsittely asetettiin automaattiseksi alustamalla *xAutoAck*-muuttuja staattisesti todeksi. Muiden yhteysasetusten osalta asetettiin WIPS-kontrollerin Modbus-lukunopeutta arviolta parhaiten vastaavat *tTimeout*- ja *tReconnectDelay*-arvot.

Kaikki serverille haluttu data kirjoitettiin rekisteriavaruudesta vastaavaan funktion *MB_TCP_ARR_W_0_7167*-muuttujaan, joka on muotoa *ARRAY [0..7167] OF WORD,* eli se koostuu 16 bitin muistiyksiköistä. Rekisterin virtuaaliset osoitteet määritettiin funktion offset-parametreilla, joilla eroteltiin holding-rekisterit, input-rekisterit, inputit sekä coilit omiin rekisteriavaruuksiinsa. Toteutuksen osalta ei ollut tiedossa, että tarvittaisiin muita kuin holding-rekistereitä. Täten holding-rekistereiden offsetiksi asetettiin 0 rekisterien osoitteiden käsittelyn helpottamiseksi. Input-tyypin rekistereille sekä coileille asetettiin offset-arvoiksi 1000, 2000 ja 3000.

5.6.2 Modbus Client: luku- ja kirjoitusfunktiot

Modbus-luku- ja kirjoitustoiminnot toteutettiin serverin tapaan ladattujen kirjastojen funktioilla. Modbus-kirjoitus- ja lukufunktiot sidottiin molemmat omiin Client-funktioihinsa *iMT-ID*-muuttujalla. Erillisten Clienttien käyttö oli mahdollista, sillä WIPS-kontrolleri tukee useita yhtäaikaisia Modbus TCP/IP -yhteyksiä. Näin ei kaikkien laitteiden kohdalla ole, jolloin eri funktiot joutuvat vuorottelemaan käyttäen samaa Clienttiä. Modbus Client -funktio muodostaa tarvittavan TCP/IP-yhteyden ja kirjoitus- tai lukufunktio välittää sille Modbus-komennon oikeassa muodossa *udtTCP_ComData* INOUT-muuttujalla [esimerkkikoodi 4]. Komennon lähettämisen jälkeen Client välittää samalla muuttujalla diagnostiikan tarvitseman datan eli tiedon, onnistuiko luku tai kirjoitus sekä mahdolliset virhekoodit.

```
MB_TCP_Client_1(udtTCP_ComData := MB_TCP_UDT_COMMUNICATION, xActivate := BOOL,
xAcknowledge := BOOL, xAutoAck := BOOL, strServer_IP := STRING, iPort := INT,
xUDP_Mode := BOOL, tReconnectDelay := TIME, tTimeout := TIME, xActive => BOOL,
xError => BOOL, wDiagCode => WORD, wAddDiagCode => WORD, udtDiag =>
MB_TCP_UDT_CLI_DIAG);
```

Esimerkkikoodi 4. Modbus Client -funktio. Esimerkin funktion muuttujat nimettynä niiden datatyypeillä.

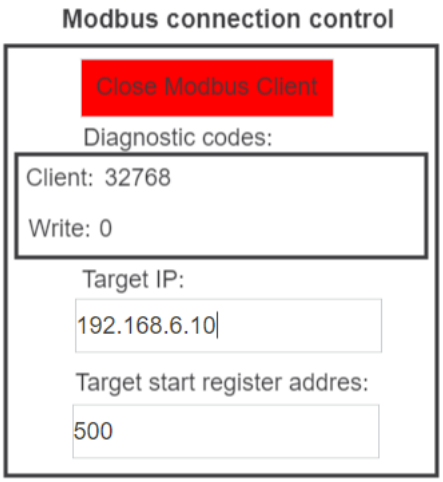
Luku- ja kirjoitusfunktiot toteutettiin kirjastojen FC3 ja F16 [esimerkkikoodi 5] funktioilla. Molemmissa funktioissa rekistereistä vastaa *MB_TCP_ARR_W_1_125*-tietotyypin muuttuja, joka on vastaavaa muotoa kuin Modbus Serverin käyttämä word array mutta lyhyempi.

```
MB_TCP_FC16_1(xActivate := BOOL, xAcknowledge := BOOL, iMT_ID := INT,
tUpdateTime := TIME, uiUnitIdentifier := UINT, wStartRegister := WORD,
uiQuantityOfRegisters := UINT, arrRegisterValue := MB_TCP_ARR_W_1_125, xActive
=> BOOL, xDone => BOOL, xError => BOOL, wDiagCode => WORD, wAddDiagCode =>
WORD, udtTCP_ComData := MB_TCP_UDT_COMMUNICATION);
```

Esimerkkikoodi 5. Modbus write multiple registers (FC16) -funktio.

Rekistereiden lukutoiminto on toiminnaltaan vastaava kuin kirjoitusfunktio. Käytännössä saman muotoiselta rekisterin word array -tietotyypin muuttujalta luetaan tieto siinä missä kirjoitettaessa sinne syötetään haluttu data.

Käyttöliittymälle luotiin serveritoteutuksen tapaan mahdollisuus avata ja sulkea yhteys haluttuun osoitteeseen sekä valita kirjoitus- tai lukukomennon aloitusrekisteri [kuva 17].



Kuva 17. Modbus Clientin ja lukukomennon yhteysasetusten käyttöliittymätoteutus.

5.6.3 Rekisterimuuttujien kirjoittaminen ja lukeminen

Modbus-luku- ja kirjoituskomennot sekä Modbus-serveri ylläpitävät rekistereitään *ARRAY [0..x] OF WORD* -muodossa. Käytännössä tämä tarkoitti sitä, että kaikki rekistereihin kirjoitettava data oli ensin muutettava Word-muotoon ja vastaavasti luettava ja käyttöliittymällä esitettävä data selväkieliseen muotoon. Käyttöliittymälle luotiin tarvittava määrä input- ja output-kenttiä, joilta kirjoitettava data ohjattiin muutosten kautta rekisterimuuttujiin ja joihin luettava data välitettiin selväkielisenä.

Käyttöliittymältä syötettävä ja WIPS-kontrollerin lukema data on usein esimerkiksi lämpötiloja tai tuulennopeuksia käsiteltäessä 32 bittiä pitkiä liukulukuja. Kaikki 32-bittiset muuttujat täytyi siis ensin pilkkoa kahteen 16-bittiseen word-muuttujaan ennen rekisteriin syöttämistä. Vakiokirjastoissa ei tarvittavaan datatyypin muutokseen ollut sopivaa funktiota, joten päädyttiin käyttämään TO_DWORD-, ROR- sekä TO_WORD-funktioita [esimerkkikoodi 6]. Toteutuksessa 32-bittinen liukuluku muutetaan ensin raakadatan sisältäväksi DWORD-muuttujaksi, jonka 16:ta vähiten merkitsevästä bitistä TO_WORD-funktio muodostaa rekisterin WORD_LOW-muuttujan. Tämän jälkeen DWORD-muuttujasta muodostetaan uusi muuttuja, jossa 16 vähiten merkitsevää bittiä vaihtaa paikkaa 16 merkitsevimmän bitin kanssa. Saadusta muuttujasta otetaan edelleen 16 vähiten merkitsevää bittiä talteen WORD_HIGH-muuttujaan.

```
Windspeed_Dword := TO_DWORD(Float32_WindSpeed (* IN *));
Windspeed_Dword_rotated := ROR(Windspeed_Dword (* IN *), 16 (* N *));
Turbine_Windspeed_word_low := TO_WORD(Windspeed_Dword (* IN *));
Turbine_Windspeed_word_high := TO_WORD(Windspeed_Dword_rotated (* IN *));
```

Esimerkkikoodi 6. Tuulennopeuden muunnos 32-bittisestä liukuluvusta kahdeksi word-muuttujaksi.

Toinen tyypillinen kirjoitettavan tai luettavan datan muoto toteutuksessa olivat boolean-muuttujista koostuvat 16 bitin jonot. Niiden lukemiseen tai kirjoittamiseen käytettiin vakiokirjastoista löytyviä SET_BIT- tai GET_BIT-funktioita [esimerkkikoodi 7]. Kirjoituskomennoissa käyttöliittymän komentopainikkeiden tiloihin sidotuista boolean-muuttujista muodostettiin ARRAY [0..x] OF BIT -muotoista dataa käsittelyn helpottamiseksi. Tämän jälkeen arrayn bitit kirjoitettiin rekisteriin syötettäviin word-muuttujiin.

```
FOR Boolean_commands1_counter := 0 TO 7 DO
    IF Boolean_commands1_array [Boolean_commands1_counter] = TRUE THEN
    Boolean_commands1_word := SET_BIT(Boolean_commands1_word (* IN *), Bool-
ean_commands1_counter (* BIT *));
    ELSE
    Boolean_commands1_word := RESET_BIT(Boolean_commands1_word (* IN *), Bool-
ean_commands1_counter (* BIT *));
    END_IF
    END_FOR
```

Esimerkkikoodi 7. Rekisterin boolean-komennoista koostuvan word-muuttujan käsittely, jossa käyttöliittymän painonappien muuttujista on ensin muodostettu bit-array.

5.7 Aliohjelmien hallinta

Prosessienhallinta tapahtuu kontrollereihin sisäänrakennetulla Execution and synchronization manager (ESM) -järjestelmällä, jolla voidaan määrittää ohjelman eri prosessien suoritusnopeudet, prioriteetit sekä käytettävä prosessoriydin [kuva 18]. Pääryhmät ESM1 ja ESM2 kuvaavat käytettävissä olevia kahta prosessoriydintä. Toteutuksessa luotiin kolmella eri nopeudella suoritettavia tehtäviä, joille aliohjelmat asetettiin. Kontaktorien, moottoroitujen automaattikatkaisimien ja liukurenkaan simulointien osalta päädyttiin 10 ms suoritussykleihin, joilla päästään aina vähintään riittävän nopeaan reagointiin. Liukurenkaan osalta ei releohjaukselle alle 10 ms syklien käyttämistä katsottu käytännölliseksi. [11, s. 22–24.]

HMI-elementtien ja Modbus-datan parsinnan osalta päädyttiin 100 ms suoritussykleihin. Modus serveri sekä luku- ja kirjoituskomentojen suoritusnopeudeksi asetettiin 1000 ms.

Name	Compone...	Task type		Program type	Int...	...	T...	Watc...
ESM1								
Cyclic100		Cyclic task			100	2	0	100
MainInstance	Arp.Plc.Eclr			Main				
Device_state_UI	Arp.Plc.Eclr			Device_state_UI				
PT100_reading	Arp.Plc.Eclr			PT100_reading				
WTG_Commands_UI	Arp.Plc.Eclr			WTG_Commands_UI				
WIPS_Output_UI	Arp.Plc.Eclr			WIPS_Output				
Enter program instance na...				*Select program type here*				
Cyclic10		Cyclic task			10	1	0	20
Circuit_breakers	Arp.Plc.Eclr			Circuit_breakers				
Contactors	Arp.Plc.Eclr			Contactors				
Enter program instance na...				*Select program type here*				
Cyclic1000_2		Cyclic task			1000	3	0	1000
Modbus_Write	Arp.Plc.Eclr			Modbus_Write				
Modbus_Read	Arp.Plc.Eclr			Modbus_Read				
Enter program instance na...				*Select program type here*				
Enter task name here								
ESM2								
Cyclic1		Cyclic task			10	1	0	20
Slipring_Auto_Timers	Arp.Plc.Eclr			Slipring_Auto_Timers				
Alarms1	Arp.Plc.Eclr			Alarms				
Slipring_Connection_Test	Arp.Plc.Eclr			Slipring_Connection_Test				
Enter program instance na...				*Select program type here*				
Cyclic10_2		Cyclic task			100	2	0	100
MBWrite_parsedata	Arp.Plc.Eclr			MBWrite_parsedata				
MBRead_parsedata	Arp.Plc.Eclr			MBRead_parsedata				
MBServer_parsedata	Arp.Plc.Eclr			MBServer_parsedata				
MBClient1	Arp.Plc.Eclr			MBClient1				
MBClient2	Arp.Plc.Eclr			MBClient2				
Enter program instance na...				*Select program type here*				
Cyclic1000		Cyclic task			1000	3	0	100
WTG_Test_Control	Arp.Plc.Eclr			WTG_Test_Control				
Watchdogs	Arp.Plc.Eclr			Watchdogs				

Kuva 18. Aliohjelmien hallinta Execution and synchronization -managerissa.

5.8 Simulointiohjelman käyttö ja dokumentointi

Simulointialusta on tehty Wicetec Oy:n sisäiseen käyttöön. Ohjelmasta julkaistut, valmiit ja käyttökelpoiset versiot säilötään pilveen. Ohjelmien osalta tallennetaan sekä projekti-tiedosto, jota tarvitaan, jos ohjelmaa halutaan muokata sekä itse kontrollerille ladattava ohjelmatiedosto, I/O-konfiguraatiot sekä HMI-tiedostot. Projektitiedoston tallentamisessa käytetään arkistomuotoa (engl. archive), joka tallentaa lisäksi kaikki projektissa käytetyt

kirjastot. Ohjelma voidaan ladata kontrollerille joko suoraan PLCnext Engineering -kehitysalustalla pelkän arkistotiedoston avulla tai vaihtoehtoisesti siirtämällä valmiit ohjelmat, konfiguraatio ja HMI-tiedostot Axioline-kontrollerin projektikansioon FTP-yhteydellä. Toteutuksessa kontrollerilla käytettiin muistikorttia, joka mahdollistaa myös ohjelman siirtämisen toiselle kontrollerille edellyttäen, että käytetty I/O-moduulien konfiguraatio on sama.

Erilliseen dokumentointiin ei ole kehitysvaiheessa keskitytty, mutta koodin sisäiseen kommentointiin pyrittiin panostamaan. Valmiista ympäristöstä tehtiin pikaohje yrityksen sisäiseen käyttöön ja ohjeessa listattiin käytetty laitteisto, ohjelmatiedostot sekä simuloinnin kattamat jäänestojärjestelmän osat. Tarvetta kattavammalle dokumentoinnille voi tulevaisuudessa tulla, etenkin jos simulointiympäristö kasvaa.

6 Yhteenveto

Työn tavoitteena oli suunnitella ja rakentaa simulointialusta Wicetec Oy:n jäänestojärjestelmän ja sen ohjelmiston testaamista varten. Tarkoituksena oli simuloida järjestelmän komponentteja, niiden vikatiloja sekä voimalan ja jäänestojärjestelmän välistä tietoliikennettä laboratorio-olosuhteissa. Kokonaisuuden tuli olla modulaarinen ja helposti laajennettavissa.

Simulointialusta on käyttökokemusten perusteella osoittautunut toimivaksi ja sille asetettujen vaatimusten voidaan katsoa täyttyneen. Suunnitelmat ovat eläneet ja muuttuneet matkan varrella, mutta päätavoite eli valmiit ja todellisuutta riittävän lähelle vastaavat simuloidut osakokokonaisuudet ovat toteutuneet. Alustalla ajettujen testien perusteella on voitu todeta, että simuloidut osat toimivat jäänestojärjestelmän ja sen ohjelmiston kanssa kuten oikeatkin komponentit, ja tietoliikenne on toiminut kuten voimalaolosuhteissa.

Insinöörityö on simulointialustan osalta rajattu sen tiettyihin toiminnallisuuksiin. Haasteita on työn aikana tuonut alustan jatkuva päivittäminen ja varsinkin se, että alusta on melko varhaisesta vaiheesta alkaen ollut käytössä, ja insinöörityöhön rajattu osuus on osin sekoittunut ja muuttunut alustalle päivitettyihin osiin. Tämä johtuu pitkälti siitä, että alustaa on rakennettu muiden töiden ohella ja sen toteuttaminen on tapahtunut osissa pitkällä ajanjaksolla. Toisaalta simulointialustan jatkuva käyttö ja uudet vaatimukset ovat tuoneet konkretiaa sen tarpeellisuudelle. Toinen haasteita tuottanut alue on ollut PLCnext-tuoteperhe ja sen verrattain nuori ikä. Ongelmatilanteissa ei tietoa tai esimerkkejä löytynyt yhtä helposti kuin jo pidempään käytössä olleiden tuotteiden kohdalla. Myöskään tuoteperheen ympärille kehittynyt yhteisö ei vielä ollut kovinkaan suuri tai aktiivinen. Yllätyksenä vastajulkaistun tuotteen mukanaan tuomat haasteet eivät tulleet, vaan ns. teknologiariski hyväksyttiin kontrolleria valittaessa.

Lähteet

1 Turkia, Ville; Huttunen, Saara & Wallenius, Tomas. 2013. Method for estimating wind turbine production losses due to icing. Verkkoaineisto. VTT. <https://www.vttresearch.com/sites/default/files/pdf/technology/2013/T114.pdf>. Luettu 17.2.2021.

2 Patented heating element technology. Verkkoaineisto. Wicetec Oy. <https://wicetec.com>. Luettu 15.11.2020.

3 Maximum energy production through the winter. Verkkoaineisto. Wicetec Oy. <https://wicetec.com/> Luettu 15.11.2020.

4 Intelligent control. Verkkoaineisto. Wicetec Oy. <https://wicetec.com/> Luettu 15.11.2020.

5 How to prevent Wind Turbine Icing? – The WIPS Technology. Verkkoaineisto. Wicetec Oy. <https://wicetec.com/technology/>. Luettu 15.11.2021.

6 Glenn Dorsey. 2019. High Reliability Slip Ring Design for Wind Turbines White paper. Verkkoaineisto. Moog Inc. <https://www.moog.com/content/dam/moog/literature/products/slip-rings/wind-turbine-slip-rings/Moog-High-Reliability-Slip-Ring-Design-for-Wind-Turbines-White-Paper.pdf>. Luettu 20.1.2021

7 Modbus FAQ. Verkkoaineisto. Modbus Organization Inc. <https://modbus.org/faq.php>. Luettu 15.3.2021.

8 Introduction to Modbus Serial and Modbus TCP. 2008. Verkkoaineisto. Contemporary Control Systems Inc. <https://www.ccontrols.com/pdf/Extv9n5.pdf>. Luettu 20.2.2021.

9 MODBUS Messaging on TCP/IP Implementation Guide. Verkkoaineisto. Modbus Organization Inc. <https://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf>. Luettu 20.2.2021.

10 Object Messaging Specification for the MODBUS/TCP Protocol. 2004. Verkkoaineisto. Modbus-IDA. <https://modbus.org/docs/Object_Messaging_Protocol_ExtensionsVers1.1.doc>. Luettu 20.2.2021.

11 PLCnext Technology user manual. Verkkoaineisto. Phoenix Contact. <https://docs.rs-online.com/1667/0900766b816b7009.pdf>. Luettu 20.10.2020.

12 PLCnext control system. Verkkoaineisto. Cross Company. <https://twitter.com/crosscompany1/status/1202981586083500033>. Luettu 20.3.2021.

13 Martin Boers. 2020. The PLCnext runtime. Verkkoaineisto. Phoenix Contact. <http://plcnext-runtime.com/ch04-00-real-time-programming.html>. Luettu 12.10.2020.

14 SIMATIC Industrial OS – the operating system for applications in the industrial environment. Verkkoaineisto. Siemens. <https://support.industry.siemens.com/cs/document/109782273/simatic-industrial-os-%E2%80%93-the-operating-system-for-applications-in-the-industrial-environment?dti=0&pnid=26135&lc=en-US>. Luettu 10.11.2020.

15 Kontrolli mukana: Sulautettu Linux. Verkkoaineisto. Wago. <https://www.wago.com/fi/sulautettu-linux>. Luettu 10.11.2020.

16 Patch Panel PP-RJ-SCC Package Slip. Verkkoaineisto. Phoenix Contact. <https://www.phoenixcontact.com/online/portal/fi?uri=pxc-oc-itemdetail:pid=2703018&library=fifi&tab=1>. Luettu 24.1.2021.

17 Global Data Space (GDS) Verkkoaineisto. Phoenix Contact. <https://www.plcnext.help/te/PLCnext_Runtime/GDS_Global_Data_Space.htm>. Luettu 15.2.2021.

18 REST data interface. Verkkoaineisto. Phoenix Contact. <https://www.plcnext.help/te/Service_Components/REST_data_interface/REST_data_interface_Introduction.htm>. Luettu 15.2.2021.

19 PLCnext Store Info Center. Verkkoaineisto. Phoenix Contact. <https://store.plcnext.help/st/Home.htm>. Luettu 20.2.2021.

20 Function block library: Modbys_TCP_8. Verkkoaineisto. Phoenix Contact. <https://s3.eu-central-1.amazonaws.com/plcnext-store-prod/doc/Modbus_TCP_8-20201026073915.pdfhttps://store.plcnext.help/st/Home.htm>. Luettu 20.2.2021.