Tampereen ammattikorkeakoulu

Automaatio-ohjelmiston testausympäristön kehitys

Jaakko Haataja

OPINNÄYTETYÖ
Huhtikuu 2022

Konetekniikan tutkinto-ohjelma, Koneautomaatio

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Konetekniikan tutkinto-ohjelma
Koneautomaatio

HAATAJA, JAAKKO:
Automaatio-ohjelmiston testausympäristön kehitys

Opinnäytetyö 28 sivua, joista liitteitä 4 sivua
Huhtikuu 2022

Opinnäytetyön tarkoituksena oli kehittää toimeksiantajana toimivan yrityksen valmiutta soveltaa ohjelmistotestausta automaatiosovelluksiin. Automaatiosovellusten monimutkaisuus ja vaativuus ovat kasvaneet teknologian kehittyessä, mikä asettaa entistä enemmän painoarvoa sovellusten testaamiselle jo sovellusta kehitettäessä. Testauksen helpottamiseksi suunniteltiin ja toteutettiin testausympäristö, jolla testaus voidaan tuoda luonnolliseksi osaksi automaatio-ohjelmistosuunnittelua.

Työssä esitellään automaatiojärjestelmän rakennetta, ohjelmistotestauksen perusteita sekä automaatiosovelluksen testaamisen ongelmia. Opinnäytetyön empiirisessä osassa esitellään testausympäristön vaatimukset, rajoitukset, sen sovellusrakenne sekä testausympäristöllä toteutettu ohjelmistotesti.

Kehitystyön tuloksena saatiin testausympäristön toimiva versio, jossa on riittävä määrä toimintoja tehdä rajattuja testejä sovellusta kehittäessä. Opinnäytetyössä esitellään myös testausympäristön jatkokehitysehdotuksia, joilla testausominaisuuksia voidaan lisätä. Kehitysehdotuksiin kuuluu muun muassa erilaisten toimilaitteiden lisääminen prosessisimulaatioon sekä jo simuloitujen toimilaitteiden tarkkuuden parantaminen esimerkiksi matemaattisella mallilla.

Asiasanat: automaatio-ohjelmisto, ohjelmistotestaus, prosessisimulaatio

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Mechanical Engineering
Machine Automation

HAATAJA, JAAKKO:
Development of a Simulation Platform for Automation Software

Bachelor's thesis 28 pages, appendices 4 pages
April 2022

The purpose of this thesis was to improve the testing capabilities of automation applications for the employing company. Automation applications have grown in size and complexity as technology has advanced, which in turn causes further demand for comprehensive testing even during the design process. To meet this demand, a test environment was designed and developed.

This thesis introduces the structure of an automation system, the basic principles of software testing, and the problems of conventional testing of automation software. The empirical part of the thesis introduces the design and development process of the simulation environment, proof that tests can be done in a developed environment. It also includes suggestions on how to further develop and improve the testing environment.

The result of this study was a working test platform on which rudimentary, simple tests could be done to support automation application development during the design process.

Key words: automation application, software testing, process simulation

SISÄLLYS

.

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena oli parantaa toimeksiantajana toimivan yrityksen valmiuksia testata automaatiojärjestelmän ohjelmistoa kehittämällä testausympäristön konsepti ja ensimmäinen toteutus. Toimeksiantaja, Insta Grouppiin kuuluva Insta Automation Oy, tarjoaa erilaisia automaatiopalveluita ja -ratkaisuja, joiden suunnitteluun ja ylläpitotöihin sisältyy automaatio-ohjelmiston kehitystyö.

Automaatio-ohjelmistojen testausmahdollisuudet ovat usein rajattuja valmistajien kehitysympäristöihin lisäämiin simulointitoimintoihin, jotka eivät kuitenkaan ole tarpeeksi kattavia tarjoamaan ohjelmiston testausta sen käyttökohteen vaatimalla tavalla. Automaatio-ohjelmistojen testaaminen on myös haastavaa, sillä usein käyttökohteet ovat prosesseja, joiden alasajaminen ei ole mahdollista pelkän ohjelmistotestauksen vuoksi. Käyttökohteet voivat myös olla osa kriittistä infrastruktuuria kuten vesihuoltoa, tai ne sisältävät raskaita nostimia, joiden vikatilanteet voivat aiheuttaa vahinkoa henkilöille, tai kiinteistölle.

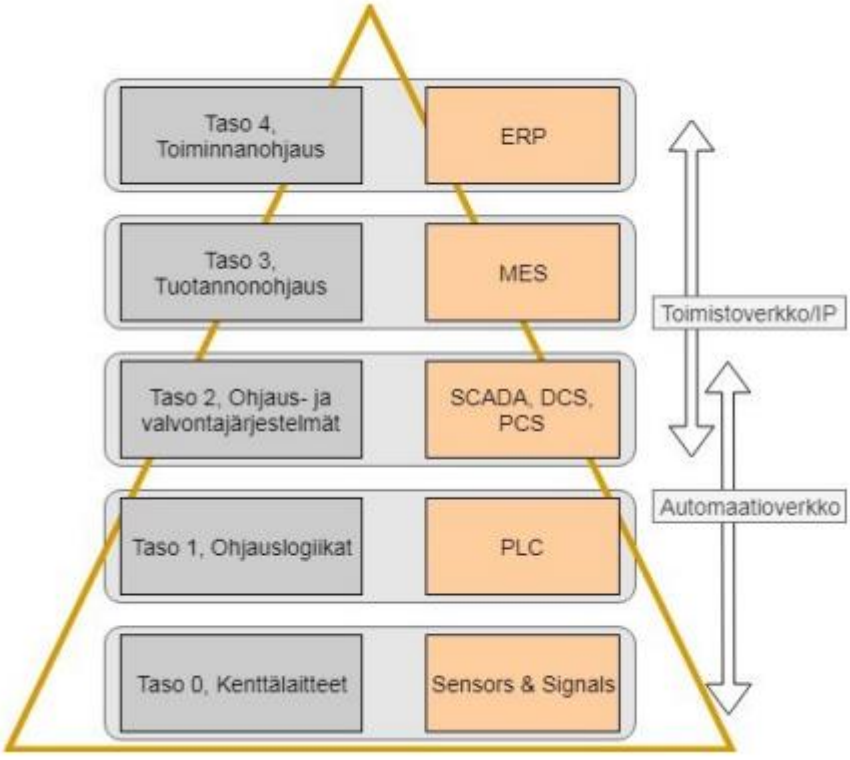
Tarkoituksena oli todistaa, että on mahdollista kehittää helposti lähestyttävä ja ymmärrettävä kokonaisuus valmiista testiympäristöstä, jolla automaatiosuunnittelija voi testata kehittämänsä ohjelman uusia toimintoja tai muutoksia, sekä ottaa käyttöönsä työkaluksi jokapäiväistä suunnittelua ja ohjelmointia. Tämä testiympäristö mallintaa erään asiakkaan näyttämömekaniikan automaatiojärjestelmää.

2 TEORIA

Tässä kappaleessa esitellään opinnäytetyölle keskeisiä käsitteitä ja viitekehys, minkä perusteella testausympäristön kehitystyö aloitettiin. Tähän sisältyy tyypillisen automaatiojärjestelmän rakenne, tietoa ohjelmistotestauksesta yleisesti, automaatio-ohjelmiston testauksen ongelmien yksityiskohtia, sekä ohjelmointiparadigma, jota käytettiin opinnäytetyön empiirisessä osassa.

2.1 Automaatiojärjestelmän rakenne

Automaatiojärjestelmä sanana kuvaa sitä kokonaisuutta, joka yhdistää itse prosessin, prosessin käyttäjän sekä prosessia hyödyntävän tahon tietojärjestelmät yhteen (Automaatiosuunnittelun prosessimalli 2007, 10). Tähän kokonaisuuteen viitataan ANSI/ISA-95-standardissa automaatiopyramidina. Kuvassa 1 on esiteltynä automaatiopyramidi.



KUVA 1. Automaatiopyramidi (Korkeamäki 2019, 8)

Tyypillisesti automaatiopyramidin kuvaama järjestelmäkokonaisuus jakautuu tiedonsiirtoprotokolliltaan kahteen: tasot 4 ja 3 toimivat pelkästään konsernin toimistoverkossa tyypillisellä internet-protokolalla. Tasot 1 ja 0 toimivat automaatioverkossa, jolla on useita vakiintuneita toteutustapoja, mm. ASI-, Profibus-, Profinet- sekä EtherCAT-kenttäväylät. (Korpela, 2020.) Näiden tasojen yhdyskäytävänä toimii taso 2, joka kommunikoi sekä toimisto- että automaatioverkon kanssa. (Korkeamäki 2019, 8.) ISA-95-standardi määrittää automaatiopyramidin tasot seuraavasti:

Taso 4

Tasolla 4 toimii yrityksen toiminnanohjaus (viitataan usein myös ERP, Enterprise Resource Planning), tietojärjestelmä, jonka avulla yrityksen johto organisoi ja suunnittelee yrityksen toimintoja. Tällä tasolla järjestelmän toimintaa käsitellään pitkillä aikaväleillä; päivinä, viikkoina, kuukausina tai jopa vuosina. (Korpela 2020.)

Taso 3

Tasolla 3 sijaitsee MES, Manufacturing Executing System, eli tuotannonohjaus. Tuotannonohjausjärjestelmä on automaatiojärjestelmän ylin taso, joka kerää informaatiota tason 2 ohjausjärjestelmältä, kuten koneiden käyntiaikoja, valmistuneita tilauksia sekä mahdollisia vikatilanteita. Se myös vastaanottaa ja käsittelee toiminnanohjaukselta saatuja tietoja mm. saapuneista tilauksista. (IEC62264-1 2013, 20.) Tuotannonohjauksen tasolla käytettävät aikamääreet ovat minuuteista tunteihin (Korpela 2020).

Taso 2

Tällä tasolla toimii ohjausjärjestelmä, joka on automaatiojärjestelmän korkein taso. Tämä taso vastaa prosessin valvonnasta sekä alempien tasojen ohjaamisesta. Ohjausjärjestelmän tarkoitus on kerätä prosessin osat samaan automaatiojärjestelmään, jotta koko prosessin ohjaaminen tapahtuu keskitetysti. Prosessiteollisuudessa yleisesti käytössä olevat ohjausjärjestelmät ovat yleisesti hajautettuja ohjaus- ja valvontajärjestelmiä, kuten prosessinohjausjärjestelmät (PCS, Process Control System) ja hajautetut ohjausjärjestelmät (DCS, Distributed Control System). Maantieteellisesti laajalle hajaantuvat järjestelmät, kuten vesi- tai energiaverkkoa valvovat ja ohjaavat prosessit, yhdistetään tyypillisesti

yhdistettyyn valvomo- ja datankeruujärjestelmään (SCADA, Supervisory Control and Data Acquisition). Ohjausjärjestelmä saa pääsääntöisesti tietonsa tasolla yksi toimivista PLC:iltä, joita se myös ohjaa saadun informaation perusteella. (Korkeamäki 2019, 8 – 9.)

Ohjausjärjestelmän käyttöpäätteenä toimii tyypillisesti yksi tai useampi järjestelmävalvomo, jossa prosessin käyttäjä voi manipuloida prosessia, tai seurata prosessin etenemistä automaattisesti. Valvomo voidaan toteuttaa yksinkertaisimmillaan näyttöpaneelilla tai paneelitietokoneella (HMI, Human Machine Interface), jolta käyttäjä saa tietoa prosessin tilasta, sekä mahdollisista hälytyksistä tai vikatilanteista. Nykypäivänä on kuitenkin yleistä, että valvomo-ohjelmistoon liitetään useita eri ohjelmistoja tallentamaan ohjausjärjestelmän saamia tietoja esimerkiksi historiatietokantaan, sekä kunnossapitosovelluksiin. (Korkeamäki 2019, 9.) Ohjausjärjestelmän tasolla käytettävät aikamääreet ovat sekunneista minuutteihin (IEC62264-1 2013, 18).

Taso 1

Taso 1 vastaa prosessin toimilaitteiden manipulaatiosta, sekä niiden tuottaman tiedon kuuntelemisesta, sekä ylöspäin välittämisestä (Korkeamäki 2019, 10). Tason keskiössä on prosessin ohjelmoitavat logiikat (Programmable Logic Controller, PLC). Ohjelmoitavat logiikat ovat eräänlaisia tietokoneita, jotka ovat erityisen toimintavarmoja sekä kestäviä (Hassapis 2000, 345).

Ohjelmoitavat logiikat toteuttavat prosessin toimilaitteiden ohjausta logiikkaan tallennetun ohjelman mukaisesti, ja se vastaa mm. prosessissa tapahtuvien sekvenssien ohjaamisesta, kiertojen laskemisesta, toimilaitteiden liikkeen ohjauksesta sekä ajoituksesta. (IEC 61131-1 2003, 7.)

Taso 0

Taso 0 on automaatiopyramidin alin taso, käsittää prosessin toimilaitteet. Näihin lukeutuu mm. prosessin anturit ja koneet, sekä koneiden ohjaimet, kuten taajuusmuuttajat. Kenttälaitteiden tehtävänä on muuttaa prosessin fysikaaliset suureet prosessia manipuloiville logiikoille ymmärrettävään muotoon, sekä muuttaa logiikan ohjaussignaalit vastaavasti fysikaalisiksi suureiksi. Toimilaitteet kytketään osaksi prosessia digitaalisesti tai analogisesti. (Korpela 2020.)

2.2 Sovellustestaus

Sovellus- tai ohjelmistotestaaminen on kokeellista toimintaa, joka tuottaa tietoa ohjelmiston laadusta päätöksentekoa varten. Tämä toteutuu testaamalla ohjelmistoa sen virheiden löytämiseksi. (Katara, Vuori & Jääskeläinen 2016, 19.)

Testaus voidaan jakaa kahteen erityyppiseen testaukseen, jotka ovat staattinen- sekä dynaaminen testaaminen. Staattinen testaaminen on ohjelman dokumentaation ja lähdekoodin tutkimista, jotta mahdollisia virheitä ohjelman toiminnasta löydettäisiin. Dynaaminen testaaminen on ohjelman testaamista syötteillä, ja ohjelman ulostulon perusteella voidaan etsiä ohjelmistosta virhettä. (Katara ym. 2016, 55.)

Dynaamista testaamista voidaan suorittaa erilaisilla testityypeillä, joista yleisiä ovat matalan tason yksikkötestaaminen, tai korkeammalla tasolla järjestelmätestaaminen. Yksikkötestauksessa ohjelman pienempää palasta, kuten funktiota tai luokkaa, testataan syötteellä, jolla testattavan osion lopputila tai ulostulo tiedetään. Järjestelmätestaaminen on sovelluksen testaamista laajemmalla mitta kaavalla esim. käyttöliittymän kautta. Järjestelmätestaamisessa etsitään ohjelmasta virhettä empiirisin menetelmin. (Katara ym. 2016, 29, 85, 149.)

Vaikka testauksen tarkoitus on löytää ohjelmiston virheet, mikään määrä testaamista ei takaa ohjelmiston virheettömyyttä. Testaaminen ei myöskään paranna ohjelmiston laatua, vaan systemaattinen testaaminen antaa kuvan ohjelmiston laadun tasosta, ja tarjoaa mahdollisuuden löytää parannettavia osa-alueita. (Katara ym. 2016, 20.)

2.3 Automaatiosovelluksen testausongelmat

PLC-pohjaisten automaatiosovellusten käyttö suuriin ja monimutkaisiin sovellusratkaisuihin on tuonut esille suuria ongelmia käytännöissä, kuinka sovelluksia kehitetään ja testataan. Testaamisen arvioidaan aiheuttavan yli puolet sovelluskehityksen kuluista. (Hassapis 2000, 345.)

Automaatiosovelluksen testaaminen irrallaan käyttökohteena toimivaa prosessia on haastavaa ja monimutkaista, sillä sovelluksen toiminta on sidoksissa prosessin logiikalle syöttämään tietoon. Jotta sovellusta voidaan testata, pitää logiikan muuttujia manipuloida käsin sovelluksen kehitysympäristöstä, tai prosessia tulee simuloida ulkoisella sovelluksella. Manuaalisesti muuttujia manipuloimalla ei kuitenkaan tuota täysin oikeaa prosessia vastaavaa kuvaa, sillä automaatioprosessiin kuuluu tyypillisesti toisistaan riippuvia, aikakriittisiä tapahtumia. Näin ollen, prosessin simulaatio ulkoisella sovelluksella on vastaus testausongelmiin. (Korkeamäki 2019, 3.)

2.4 Oliopohjainen ohjelmointi

Oliopohjainen ohjelmointi (OOP, Object Oriented Programming) on ohjelmointiparadigma, jossa ohjelman tietoa ja toimintoja mallinnetaan loogisesti ihmiselle ymmärrettäviin kokonaisuuksiin. Oliopohjaisessa ohjelmoinnissa tieto- ja toimintokokonaisuudet kerätään luokiksi. Luokka sisältää tiedot ja toiminnallisuudet, eli metodit, joita luokasta muodostettavat oliot käyttävät tiedon vastaanottamiseen, lähettämiseen sekä käsittelyyn. (Olio-ohjelmointi C++, n.d.) Olioon viitataan myös sen englannin kielisellä termillä, objektilla (Ohjelmointi 1, n.d.).

OOP-rakenne helpottaa sovelluksen jatkokehittämistä, sillä se mahdollistaa laajojen kokonaisuuksien pilkkomisen helposti lähestyttäviksi palasiksi. Yksittäisiä toimintoja on helppo lisätä tai muokata, rikkomatta koko ohjelmakoodin rakennetta. (Laaksonen 2011.)

3 TESTAUSYMPÄRISTÖN KEHITYS

Tässä kappaleessa esitellään tehty suunnittelu ja kehitystyö testausympäristön PoC (Proof of Concept, soveltuvuusselvitys) -toteutuksesta. Seuraavissa kappaleissa määritellään testausympäristön vaatimukset ja rajaukset, miten testausympäristön sovellusrakenne muodostaa, testausympäristön eri sovellukset, käytetyt työkalut ja sovelluskomponentit. Myös opinnäytetyön tekijän kehittämä prosessisimulaatiosovellus esitellään tarkemmin.

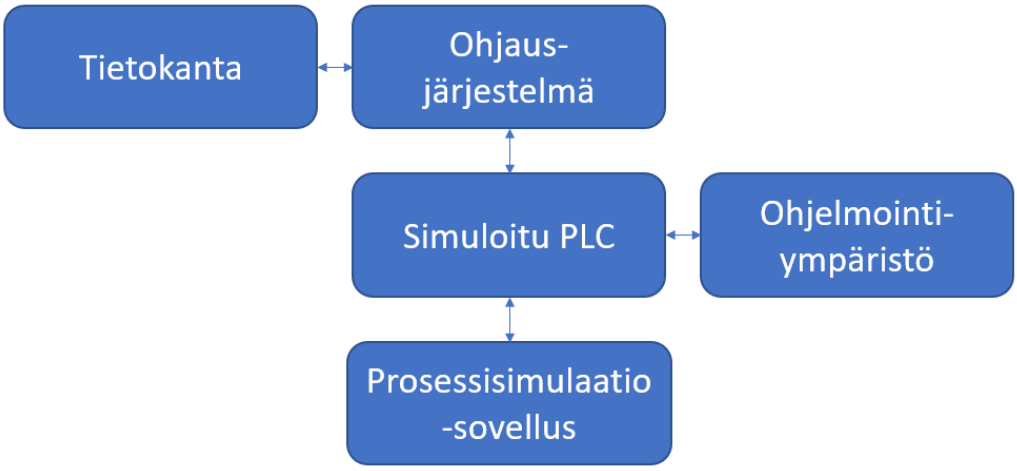
3.1 Testausympäristön vaatimusmäärittely ja rajaukset

Luvussa 2.3 esiteltyjen automaatiosovelluksen testauksen ongelmien ratkaisemiseksi, etsittiin testausympäristöllä selkeitä ja konkreettisia ongelmia, jotka sen tulisi ratkaista. Testausympäristön vaatimuksena oli, että sillä voitaisiin toteuttaa ohjelmistotestausta kattavammin ja helpommin, kuin perinteisen automaatiosovelluksen kehitysympäristöön integroidulla PLC-simulaattorilla voi. Kehitysympäristön integroidun simulaattorin suurimmiksi haasteiksi tunnistettiin seuraavat ongelmat: logiikan sisääntulojen tilojen manipulointi suuressa projektissa on työlästä ja hidasta, mikä saattaa tehdä tietynlaisen, aikakriittisen testaamisen mahdottomaksi, sekä takaisinkytkentätieto kenttälaitteelta on lähtökohtaisesti mahdotonta. Tätä vikaa voidaan kiertää ohjelmoimalla logiikan ohjelmaan simulaatioominaisuuksia, mutta tämä tuottaa ylimääräistä työtä ja varaa logiikan muistista tilaa.

Edellä mainittujen rajoitusten ratkaisemiseksi testausympäristön osaksi suunniteltiin prosessisimulaatiosovellus, joka matkii kenttälaitteita ja tarjoaa takaisinkytkentäinformaatioita simuloiduilta toimilaitteilta. Sovellus kykenee myös kirjoittamaan logiikan sisääntuloihin todellisuutta vastaavia tietoja. Testausympäristön kehityksen ensisijainen kohde määriteltiin olevan Insta Automationin ylläpitämät teatterit, tarkemmin näiden teattereiden näyttämömekaniikka. Testausympäristön rajattiin myös tehtäväksi Siemens-pohjaiseen järjestelmään Siemensin tarjoamien, kehitysympäristöön liitettävien, apusovellusten vuoksi. Näistä keskeisin on Siemens PLCSIM Advanced, joka esitellään tarkemmin kappaleessa 3.3.

3.2 Testausympäristön sovellusrakenne

Testiympäristön keskiössä on simuloitu PLC. Tällä virtuaalilogiikka kommunikoi ohjausjärjestelmän kanssa, jolta se vastaanottaa ohjauskäskyjä ja jolle logiikka välittää tietoa siitä, mitä se saa prosessisimulaatiosovellukselta. Prosessisimulaatiosovellus manipuloi sen sisäisiä virtuaalisia toimilaitteita simuloidun PLC:n ulostulojen tilojen mukaisesti. Kappaleessa 2.1 esiteltyyn automaatiopyramidin tasoa 0 vastaa testausympäristössä seuraavassa kappaleessa esiteltävä prosessisimulaatiosovellus. Prosessisimulaation toiminta ja sen kehitys esitetään tarkemmin kappaleessa 3.4. Kuvassa 2 havainnollistetaan testausympäristön rakennetta.



KUVA 2. Testausympäristön sovellusrakenne

3.3 Testausympäristössä käytetyt sovellukset

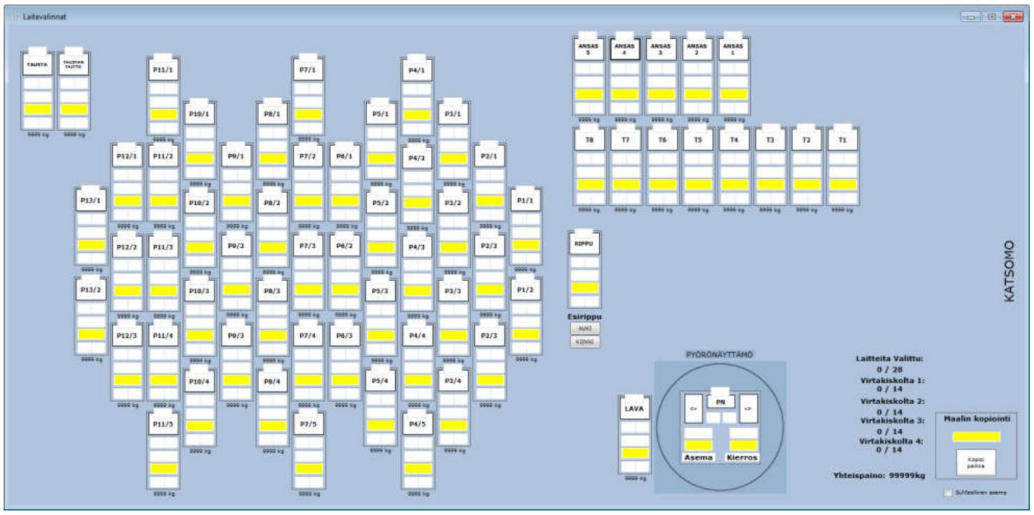
Seuraavat opinnäytetyön kappaleet esittelevät kuvassa 2 havainnollistetut testiympäristön sovellukset, sekä kehitysympäristöt, joita käytettiin kehitystyön aikana.

Testiympäristön sovellukset asennettiin Windows-pohjaiselle virtuaalikoneelle, joka toimi kehitysympäristön alustana. Virtuaalikoneen toimintaa tai konfiguraatiota ei käsitellä tässä opinnäytetyössä.

Whisper Show Control & tietokanta

Whisper Show Control on Instan kehittämä yhdistetty ohjaus- ja valvomosovellus, jolla teattereiden operaattorit manipuloivat näyttämömekaniikkaa näytösten aikana, ja sen ulkopuolella esimerkiksi lavasteita ripustaessa. Testausympäristöön valittu versio Whisper-sovelluksesta on Windows PC:lle suunniteltu ohjelmisto, johon on liitetty SQL-tietokanta. Sovelluksen kautta käyttäjä voi tallentaa esiasetettuja konfiguraatioita toimilaitteiden tiloista tietokantaan, mikä nopeuttaa prosessin muuttamista tilasta toiseen esitysten aikana. Tietokantaan on myös tallennettu toimilaitteiden oletusparametrejä, jotka prosessin käyttäjä voi palauttaa laitteille tarpeen vaatiessa.

Kuvassa 3 esitellään näkymä, josta käyttäjä voi manipuloida näyttämömekaniikan toimilaitteita, sekä nähdä voimassa olevat hälytykset ja varoitukset. Tätä näkymää käytettiin luvussa 3.5 esitellyn testin tekemiseksi.



KUVA 3. Whisper-käyttöliittymä (Insta Automation 2017)

SIMATIC NET OPC-yhdyskäytävä

Open Platform Communications (OPC) on yhteensopivuusstandardi, jota noudattavat laitevalmistajat mahdollistavat kommunikaation erilaisten järjestelmäkomponenttien välillä. OPC-protokolla kehitettiin toimimaan tiedon välittäjänä ja standardisoijana PLC:n ja HMI- tai SCADA-sovelluksen välillä. (OPC Foundation n.d.)

Testausympäristön ohjausjärjestelmän kommunikaatio virtuaalilogiikalle toteutettiin Siemensin Simatic NET OPC-rajapinnalla. Simatic NET-sovelluksella luodaan

yhdyskäytävä ohjausjärjestelmän ja virtuaalisen logiikan välille, jolloin ohjausjärjestelmä voi välittää käskyjä ja vastaanottaa informaatiota logiikalta kommunikaatiorajapinnan avulla. Tämä tehdään määrittämällä Siemens Communication Settings nimisellä OPC-määrittelykäyttöliittymäsovelluksella. OPC-yhdyskäytävään määritellään kommunikaatioväylän osapuolet sekä verkko-osoitteet.

Opinnäytetyön liite 1 esittää esimerkin kommunikaatio-osapuolten määrittelystä, joka tehtiin suunnittelutyötä aloitettaessa. Yhdyskäytävään määriteltiin testausympäristöä isännöivän virtuaalisen tietokoneen verkkosovitin ja IP-osoite, sekä kommunikaation toisen osapuolen logiikan tyyppi, sekä logiikan IP-osoite.

Siemens TIA Portal

Siemens Totally Integrated Automation Portal on automaatio-ohjelmointiympäristö, jolla Siemens-pohjaisissa automaatiojärjestelmissä toteutetaan automaatio-ohjelmointi. TIA Portalilla suunnittelija voi kehittää PLC-ohjelman ja -konfiguraation lisäksi HMI:n käyttöliittymän, sekä konfiguroida mahdolliset väylälaitteet vastaamaan prosessin tarpeita.

Testausympäristöä varten TIA Portalia käytettiin lataamaan virtuaalilogiikkaan ylläpidettävän teatterin koko logiikkakoodi. Tämän lisäksi ohjelmointiympäristöä käytettiin prosessisimulaatiosovelluksen kehityksen aikana monitoroimaan muutoksia virtuaalilogiikan I/O:ssa.

Siemens S7-PLCSIM Advanced

Siemens PLCSIM Advanced on Siemensin kehittämä simulointityökalu, jonka avulla testausympäristössä voidaan simuloida virtuaalista Siemens S7-1500-sarjan ohjelmoitavaa logiikkaa. TIA Portaalin simulointitoimintoon verrattuna PLCSIM Advanced tarjoaa enemmän vapautta simuloida prosessia, sekä vaikuttaa siihen ulkoisella sovelluksella Siemensin toimittaman Advanced Sim API:n (Application Programming Interface, ohjelmointirajapinta) ansiosta. Tätä rajapintaa hyödynnettiin kappaleessa 3.4 esiteltävän prosessisimulaatiosovelluksen kehityksessä.

Opinnäytetyön liitteessä 2 esitellään PLC-simulaattorin käyttöliittymä. Liitteestä poiketen, testausympäristössä käytettiin ohjelmaversiota 2.0, jonka käyttöliittymä on identtinen esitettävän käyttöliittymän kanssa.

Microsoft Visual Studio

Prosessisimulaatiosovelluksen kehittämiseen käytettiin Microsoft Visual Studio integroitua ohjelmistokehitysympäristöä (IDE, Integrated Development Environment). Microsoft tarjoaa Visual Studiosta kolmea eritasoista ohjelmistoversiota, Community, Professional ja Enterprise.

Visual Studio tarjoaa pelkän koodin kirjoitusalustan lisäksi laajennuksia ja ominaisuuksia, jotka helpottavat ja nopeuttavat ohjelmistokehitystä, kuten mm. IntelliCode-täydennystyökalun, CodeLens-seurantatyökalun, sekä Git-versionhallintaintegraation. (Microsoft 2022a.)

3.4 Prosessisimulaatiosovellus

Prosessisimulaatiosovelluksen tehtävä on simuloida oikeassa kohdeympäristössä toimivia kenttälaitteita, kuten antureita ja taajuusmuuttajia. Sovellus vastaa virtuaalisen logiikan käskyihin liikuttaa simuloituja toimilaitteita, sekä antaa sille simulaatioprosessista tarvittavia tietoja logiikan sisääntuloihin.

Prosessisimulaatiosovelluksen vaatimuksena, kappaleessa 3.1 esitettyjen rajoituksien kiertämisen lisäksi, oli sovelluksen skaalattavuus. Sovelluksen tulee mahdollistaa useiden toimilaitteiden rinnakkainen simulaatio, jotta testausympäristö vastaisi oikeaa prosessia, sekä toimisi simuloidulle logiikalle ladatun ohjelman kanssa saumattomasti.

3.4.1 Sovelluksen kehitystyön aloittaminen

Prosessisimulaatiosovelluksen kehittäminen aloitettiin käytännössä tyhjältä pöydältä. Siemensin tarjoaman PLCSIM Advanced API:n ominaisuudet rajasivat käy-

tettävät ohjelmointikielet C++:aan tai C#:iin. Näistä opinnäytetyön tekijä valitsi aikaisemman kokemuksensa perusteella C#:in, sillä graafisten elementtien lisääminen ja integroiminen sovellukseen on kyseisellä ohjelmointikielellä helpompaa.

Graafinen C#-projekti päätettiin toteuttaa C#.NET 6 ohjelmistokehyksellä, ja käyttöliittymä kehitettiin Windows Forms-kehyksellä. Forms-kehys tarjoaa ohjelmistotyökaluja kehittää tehokkaasti käyttöliittymä Windows-sovellukselle. (Microsoft 2022b.)

3.4.2 Kehitystyön rajaaminen

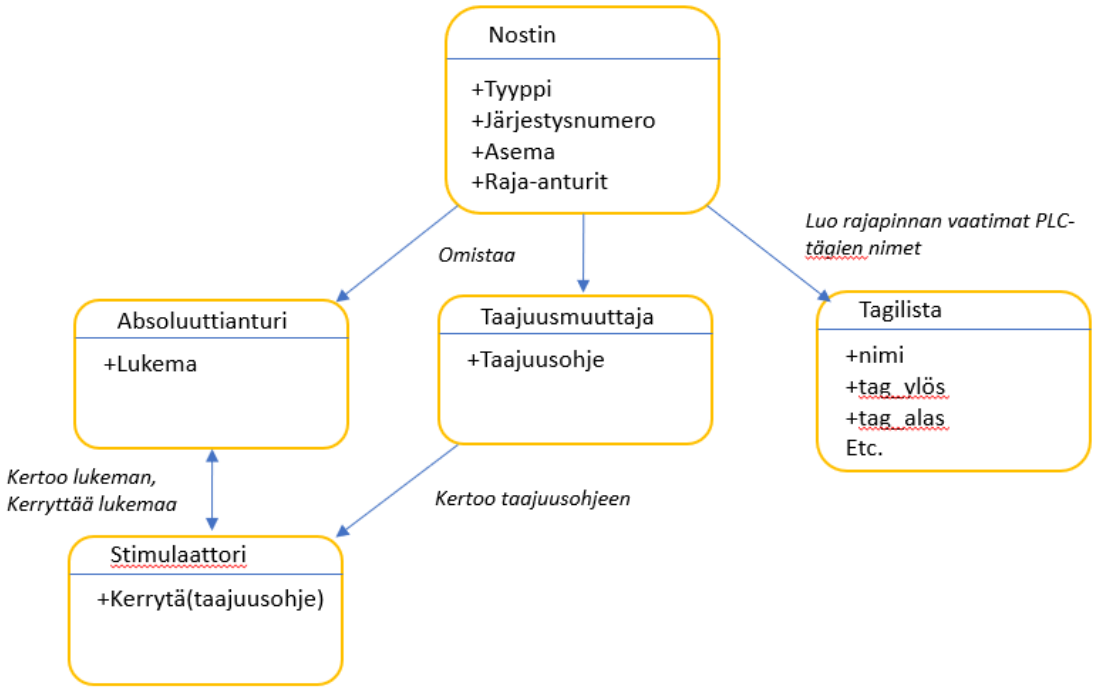
Simuloitujen kenttälaitteiden toiminnallisuuden mallintaminen reaalimaailmaa täydellisesti vastaavaksi ei määritelty tarpeelliseksi tämän opinnäytetyön raameissa, mutta toiminnallisuuden tulee vastata läheisesti käytössä olevia kenttälaitteita.

Prosessisimulaatiosovelluksen PoC-toteutuksen ominaisuuksia rajattiin tämän opinnäytetyön laajuudelle seuraavasti: sovelluksen tulee luoda ja käynnistää simuloitava virtuaalilogiikka, simuloida taajuusmuuttajalla ohjattavaa sähkömoottoria sekä nostinta, jota moottori liikuttaa, sekä tähän prosessin osaan kuuluvia antureita, simuloitavia nostimia tulee pystyä lisäämään tai poistamaan vaivattomasti, sekä niitä sekä nostimeen liittyvien koneiden parametrit voidaan määrittää helposti, sekä sovelluksen tulee näyttää visuaalisesti nostimen asema.

Prosessisimulaatiosovelluksen tehokkuuden tarve tiedostettiin, jotta simulaatiosovellus ei hidastu ja täten muutu epäluotettavaksi isoa prosessia simuloitaessa, mutta sille ei asetettu tavoitteita tai rajoja.

3.4.3 Sovelluksen rakenne

Ohjelmointityötä päätettiin lähestyä oliopohjaisen ohjelmointiparadigman mukaisesti kappaleessa 2.4 esitettyjen hyötyjen vuoksi. Sovellukselle määriteltiin Unified Modeling Language-standardia mukaileva luokkakaavio (kuva 7), joka osoittaa kuinka sovelluksen simuloima toimilaite saa toimintonsa ja jäsenmuuttujansa.



KUVA 7. Sovelluksen luokkakaavio

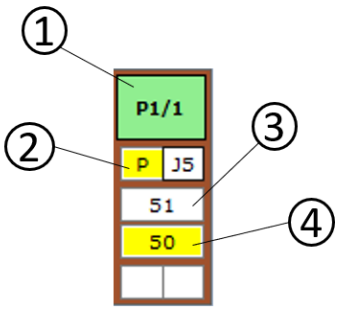
Jotta nostinten lisääminen, poistaminen sekä parametrien asettaminen ei vaatisi lähdekoodin muuttamista, määritettiin nostimen tyyppiä, järjestysnumeroa sekä asemaa vastaavat tiedot luettavaksi Microsoft Excelillä tuotettavasta tiedostosta, joka asetetaan samaan tiedostokansioon sovelluksen kanssa. Oliopohjaisen ohjelmointilähestymistavan vuoksi nostimia voidaan lisätä vaivattomasti enemmän, eikä nostimen lisäys aiheuta muutoksia ohjelman koodiin. Sovelluksen käynnistyessä, ohjelma lukee tarvittavat tiedot Excel-taulukosta, ja luo tarvittavan määrän Nostin-olioita ohjelmaan.

Nostinten toiminnan visuaalisen seuraamisen toteuttamiseksi, jokaiselle Nostin-oliolle luodaan ohjelmallisesti oma ohjauskokoelma, johon kuuluu yleisiä Windows-sovelluksen ohjauksia, kuten tekstikenttiä, painikkeita sekä liukusäädin, joka kuvaa nostimen asemaa. Ohjauskokoelman toiminta sekä ulkonäkö esitellään tarkemmin luvussa 3.5.

Siemensin tarjoaman API tarjoaa sovelluksen käyttöön PLCInstance-luokan. Tämä luokka mahdollistaa kommunikaation virtuaalisen logiikan kanssa, ja se tarjoaa laajan funktiokirjaston mm. logiikan sisääntulojen kirjoittamiseksi sekä sen ulostulojen tilojen lukemiseksi. Tämän kirjaston avulla prosessisimulaatiosovellus välittää virtuaalilogiikalle tiedon Nostin-oliolta luettavien jäsenmuuttujien tilojen perusteella. Osa funktiokirjastosta on esiteltynä opinnäytetyön liitteessä 3.

3.5 Testin teko

Edellisissä kappaleissa esiteltyjen sovellusten onnistuneen integraation jälkeen testausympäristöllä tehtiin yksinkertaisia testejä, jotka ovat tyypiltään kappaleessa 2.2 määriteltyjä dynaamisia järjestelmätestejä. Toimilaitteelle annettiin ohjaussovelluksen kautta käsky liikkua asetusarvon määrittelemään korkeuteen, ja sen paikka luettiin liikkeen loputtua prosessisimulaatiosovelluksessa, sekä ohjaussovelluksessa. Toimilaitteen liikkeelle oli määritelty alarajaksi 50 mm ja ylärajaksi 4500 mm. Kuvassa 9 esitetään ohjaussovelluksen toimilaitteen käyttöliittymä, sekä selitetään sen keskeiset toiminnot.

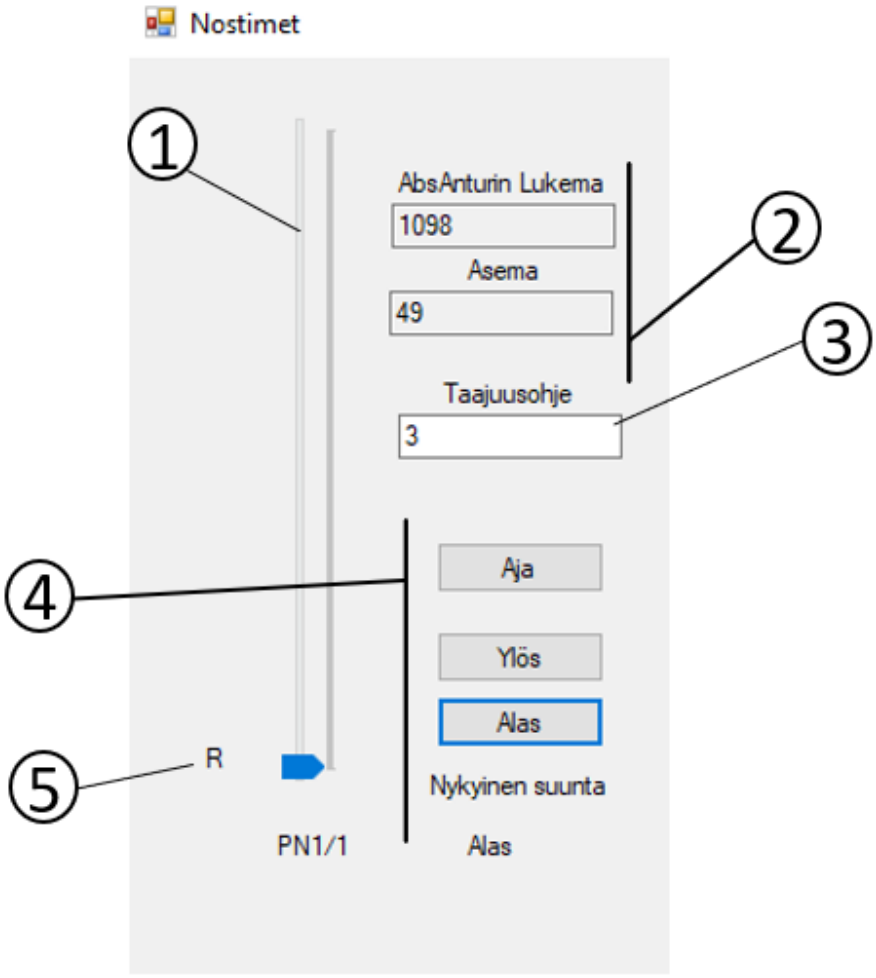


1. Toimilaitteen tunnus
2. Paikoitus- tai käsiajo valinta
3. Toimilaitteen asema
4. Toimilaitteen korkeuden asetusarvo

KUVA 9. Ohjausjärjestelmän käyttöliittymä kenttälaitteelle

Nostinkenttälaite oli asetettu liikkumaan 50 millimetrin korkeuteen. Liikkeen päätyttyä ohjaussovelluksen käyttöliittymästä luettiin nostimen asettuneen asemaan 51 mm.

Kuvassa 10 esitetään prosessisimulaatiosovellus saman testin aikana. Kuvassa selitetään myös sovelluksen keskeiset toiminnallisuudet.



KUVA 10. Prosessisimulaation käyttöliittymä kenttälaitteelle

Kuvan 10 osoittimien selitykset:

1. Liukusäädin, joka kuvastaa toimilaitteen asentoa
2. Toimilaitteen paikoitusanturin lukema, sekä siitä päätelty asema
3. Toimilaitetta ohjaavan taajuusmuuttajan taajuusohje
4. Toimilaitteen paikaillisohjainta simuloivat painikkeet
5. Toimilaitteen virtuaalisen rajakytkimen indikaattori

Virtuaalitoimilaitteelle toteutettiin yhteensä yhdeksän, kappaleen alussa kuvattua, paikoitustestiä toiminnan varmistamiseksi. Testien tulokset esitetään taulukossa 1.

TAULUKKO 1. Paikoitustestien tulokset

Alkuasema (ohjausjärjestelmä)	Maali	Lopullinen asema, Ohjausjärjestelmä	Lopullinen asema, simulaatio	Rajakytkin aktivoitunut
500	50	51	49	Kyllä
55	900	966	930	Ei
900	1200	1286	1239	Ei
1286	1600	1722	1659	Ei
1722	2000	2100	2024	Ei
2702	3000	3116	3002	Ei
3116	3500	3759	3623	Ei
3759	4500	4669	4500	Kyllä

Paikoitustestien tuloksia ei käsitellä tarkemmin tässä opinnäytetyössä, sillä tämän opinnäytetyön tarkoituksena oli tuottaa ympäristö, josta tuloksia voidaan kerätä.

3.6 Jatkokehitysehdotukset

Testausympäristöä kehittäessä ilmeni tapoja, joilla testausympäristön ja prosessisimulaatiosovelluksen kehitystä voisi jatkaa. Luvussa 3.1 mainittujen rajoituksien purkaminen on oleellinen osa testausympäristön jatkokehitystä.

Opinnäytetyön esittämä testausympäristö soveltuu vain yhden asiakasteatterin automaatiojärjestelmään. Testausympäristöä voitaisiin laajentaa kattamaan myös muiden logiikkavalmistajien järjestelmiä. Tämä työ on laaja, ja tätä opinnäytetyötä aloitettaessa oli selkeää, että vain Siemens tarjoaa tarpeeksi kattavan valikoiman sovelluksia ja protokollia, joilla testausympäristö voitaisiin toteuttaa.

Muiden logiikkavalmistajien saattamiseksi osaksi testausympäristöä tulisi kehittää lisää sovelluksia, jotka mahdollistavat virtuaalisen automaatiojärjestelmän rakentamisen.

Prosessisimulaatiosovelluksen kehitystyötä voidaan pitää lähes päättymättömänä. Toimilaitteiden simulaation tarkkuutta voidaan parantaa matemaattisilla malleilla, simulaatiosovelluksen käyttöliittymää voidaan kehittää käyttäjäpalautteen pohjalta sekä sovellukseen voidaan lisätä mahdollisuus simuloida toimilaitteiden vikatilanteita. Simulaatiosovellus voidaan myös jatkojalostaa konfiguroitavammaksi, jotta sen käyttöönotto asiakkaan automaatiojärjestelmän rakenteesta riippumatta olisi vaivattomampaa. Tämä voitaisiin toteuttaa lisäämällä esimerkiksi Excel-taulukoista luettavaa tietoa sovellukseen.

4 POHDINTA

Tämän opinnäytetyön tehtävänä oli tuottaa automaatiosovelluksen testaamiseen soveltuvan testausympäristön PoC-toteutus, joka tulisi toimimaan pohjana jatkokehitykselle testaustyökalun kehityksessä, jonka on tarkoitus paikata puutteita nykyisissä automaatiotestauksen käytännöissä.

Opinnäytetyön lopputuloksena saatiin ympäristö, jolla voi suorittaa automaatiojärjestelmän järjestelmätestausta. Ensimmäisen testin tuloksesta voidaan nähdä, että prosessisimulaatio ei vastaa täysin käytetyn asiakkaan automaatiojärjestelmän toimilaitteita, mutta mahdollisuus tehdä tämä testi osoittaa, että opinnäytetyön konsepti testausympäristöstä on toimiva. Simulaation toimilaitteiden mallintaminen, sekä toimilaitetyyppien lisääminen tulee olemaan jatkokehityksen keskiössä.

Opinnäytetyö oli tekijälleen erityisen opettavainen. Testiympäristön rakentamisen edellytyksenä oli perehtyminen jokaiseen järjestelmän sovellukseen, sekä erilaisiin protokolliin ja tapoihin, millä sovellukset kommunikoivat keskenään. Prosessisimulaatiosovelluksen kehittäminen haastoi opinnäytetyön tekijän osaamista ohjelmoinnissa sekä käsitystä sovelluksen rakenteesta, mutta käytetyn ohjelmointikielen dokumentointiin riittävä perehtyminen mahdollisti ensimmäisen prosessisimulaatiosovelluksen version kehittämisen.

LÄHTEET

Hassapis G. 2000. Soft-testing of industrial control systems programmed in IEC 1331-3 languages. ISA Transactions 39. Kreikka.

IEC 61131-1. 2003. Programmable controllers – part 1: General information. Helsinki: Suomen Standardoimisliitto SFS. Luettu 4.4.2022. Vaatii käyttöoikeuden. https://online.sfs.fi/fi/index.html.stx

Insta Group. 2022a. Automaatio ja sähköistys. Verkkosivu. Viitattu 6.4.2022. https://www.insta.fi/asiakastarinat/tag/automaatio-ja-sähköistys

Insta Group. 2022b. Tietoa konsernista. Verkkosivu. Viitattu 6.4.2022. https://www.insta.fi/tietoa-meista

Insta Automation Oy. 2017. Whisper käyttöohjekirja. Julkaisematon.

Katara M, Vuori M & Jääskeläinen A. 2016. Ohjelmistojen testaus koulutusmateriaali. Tampereen teknillinen yliopisto 2016.

Korkeamäki H. 2019. Simulointi automaatiosovelluksen testauksessa. Tekniikan ja luonnontieteiden tiedekunta. Automaatiotekniikan diplomi-insinöörin tutkinto-ohjelma. Tampereen Yliopisto. Diplomityö.

Korpela M. 2020. Ohjausjärjestelmät koulutusmateriaali. Tampereen Ammattikoulu 2020.

Laaksonen A. 2011. Ohjelmoinnin jatkokurssi koulutusmateriaali. Helsingin Yliopisto. https://www.cs.helsinki.fi/u/ahslaaks/kesa11/ohja/oliot2.html

Microsoft. 2022a. Visual Studio 2022. Luettu 5.4.2022. https://visualstudio.microsoft.com/vs

Microsoft. 2022b. Windows Forms .NET 6 documentation. Luettu 5.4.2022. https://docs.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-6.0

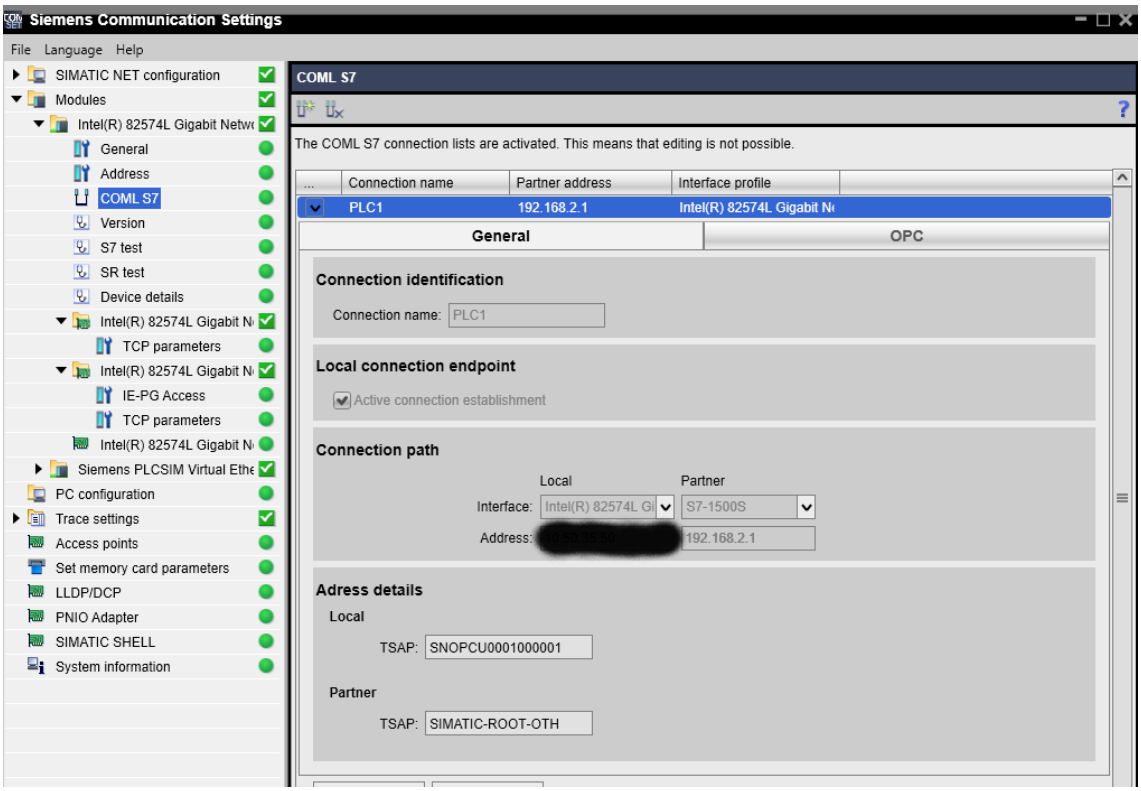
Ohjelmointi 1 koulutusmateriaali. Aalto Yliopisto. Luettu 8.4.2022 https://plus.cs.aalto.fi/o1/2018/w02/ch01/?hl=fi

Olio-ohjelmointi C++ koulutusmateriaali. Metropolia Ammattikorkeakoulu. Luettu 4.4.2022.

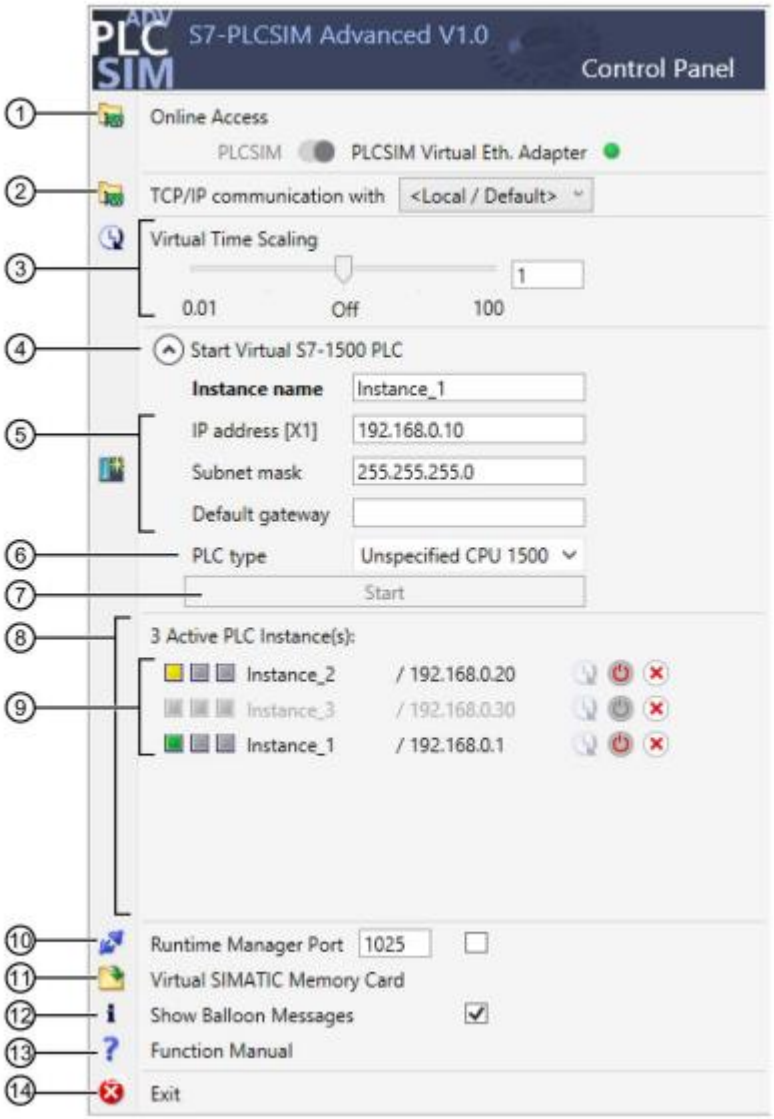
OPC Foundation. What is OPC? Luettu 6.4.2022. https://opcfoundation.org/about/what-is-opc/

LIITTEET

Liite 1. SIMATIC NET konfiguraatioesimerkki



Liite 2. Siemens PLCSIM Advanced käyttöliittymä



Käyttöliittymän toiminnallisuudet ovat:

1. Kommunikaatiorajapinnan valinta
2. TCP/IP-verkkoadapterin valinta
3. Virtuaalisen ajan skaalaus. Asetuksen avulla käyttäjä voi nopeuttaa tai hidastaa virtuaalilogiikan sisäisen kellon kulkua.
4. Virtuaalisen PLC-instanssin konfiguroinnin valikko, sekä nimikenttä.
5. Virtuaalilogiikan verkkoasetukset
6. Virtuaalilogiikan tyyppi
7. Virtuaalisen PLC-instanssin luova painike
8. Lista rekisteröidyistä virtuaalisista logiikoista

9. Virtuaalisten logiikoiden LED-indikaattorit, jotka kuvastavat logiikan tilaa; vihreä väri ilmaisee logiikan olevan RUN-tilassa ja toimivan virheettömästi, keltainen valo ilmaisee logiikan olevan STOP-tilassa, ja punainen vilkkuva valu ilmaisee virheestä logiikalla. Häivytetty rivi listassa tarkoittaa kyseisen logiikan olevan virrattomassa tilassa.
10. Tietoliikennekommunikointiin käytetyn portin valinta
11. Virtuaalisen muistikortin osoittamiseen käytetty painike
12. Kytkee simulaattorin lähettämät Windows-huomautukset päälle tai pois
13. Avaa S7-PLCSIM Advanced-dokumentaation.
14. Sulkee käyttöliittymän. Käyttöliittymän ei tarvitse olla käynnissä virtuaalilogiikoiden toimintaa varten.

Liite 3. Näyte Siemens Advanced Sim API funktiokirjastosta

Table of contents