Ari Tenhunen

Automaationsuunnittelun tehostaminen Openness-rajapinnan avulla

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

12.3.2024

Tiivistelmä

Tekijä:	Ari Tenhunen
Otsikko:	Automaationsuunnittelun tehostaminen Openness-rajapinnan avulla
Sivumäärä:	55 sivua
Aika:	12.3.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikan tutkinto-ohjelma
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Jorma Räty, Metropolia Ammattikorkeakoulu Jukka Uotila, Siemens Oy, vastaava työnjohtaja

---

Tämän insinöörityön keskeisenä tavoitteena oli tutkia TIA Openness -rajapinnan mahdollisuuksia PLC-ohjelmistokehityksen automatisoinnissa TIA Portal -ympäristössä. Työssä tarkasteltiin perusteellisesti TIA Opennessia, TIA Portalin toimintoja ja ohjelmistokehitystyökaluja.

Tarve tutkimukselle syntyi yrityksen johdon halusta korvata manuaalinen kopiointityö automatisoinnilla. Tällä pyrittiin vähentämään automaatiosuunnittelijan työaikaa ja vastaamaan kasvavien projektien asettamiin aikataulu- ja resurssivaatimuksiin.

Insinöörityön tuloksena kehitettiin sovellus, joka käytti TIA Openness -rajapintaa ohjelmalohkojen generointiin TIA Portal -projektiin. Tämän avulla pyrittiin poistamaan toistuvaa työtä. Sovellus toteutettiin käyttäen Python-ohjelmointikieltä ja Bootstrap-komponentteja käyttöliittymässä.

Tutkimuksen myötä saavutettiin konkreettinen ratkaisu automaation tehostamiseksi PLC-ohjelmistokehityksessä, mikä vastasi yrityksen tarpeisiin. Sovelluksen avulla saavutettiin merkittävä työaikasäästö ja kyky vastata joustavasti projektien vaatimuksiin.

Avainsanat: Openness, TIA Portal, PLC, Ohjelmistokehitys

Abstract

Author:	Ari Tenhunen
Title:	Enhance Automation Engineering Using Openness Interface
Number of Pages:	55 pages
Date:	12 March 2024
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Software engineering
Supervisors:	Jorma Räty, Principal Lecturer Jukka Uotila, Project Manager

---

The main objective of the study was to investigate the possibilities of the TIA Openness interface in automating PLC software development within the TIA Portal environment. The research thoroughly examined TIA Openness, TIA Portal functions and software development tools.

The need for the research arose from the commissioner's desire to replace manual copy-paste tasks with automation aiming to reduce the workload of automation designers and meet the time and resource requirements imposed by growing projects.

As a result of the study, an application was developed utilizing the TIA Openness interface for generating program blocks in the TIA Portal project. The aim was to eliminate repetitive work. The application was implemented using the Python programming language and Bootstrap components in the user interface.

The study produced a concrete solution to enhance automation in PLC software development to suit the company's needs. The application led to significant time savings and the ability to respond flexibly to the project's requirements.

Keywords: Openness, TIA Portal, PLC, Software development

Sisällys

Lyhenteet

API: Application Programming Interface. Ohjelmointirajapinta mahdollistaa sovellusten tiedon välisen kommunikoinnin.

XML: Extensible Markup Language. Merkintäkieli, jota käytetään tietojen tallentamiseen ja siirtämiseen.

PLC: Programmable Logic Controller. Ohjelmoitava logiikkakontrolleri, jota käytetään teollisuusautomaation ohjausjärjestelmien ohjelmointiin, hallintaan ja valvontaan.

TIA: Totally Integrated Automation. TIA-järjestelmä tarjoaa yhtenäisen alustan, joka kattaa ohjelmoitavat logiikkakontrollerit (PLC), käyttöliittymät, käyttöjärjestelmät, ajurit ja muut teollisuusautomaation osa-alueet.

OB: Organization Block. Organisaatiolohko (pääluokka) sisältää käyttäjän määrittelemää ohjelmakoodia, joka ajetaan automaattisesti tietyissä tilanteissa kuten PLC:n käynnistyessä tai pysähtyessä.

DB: Data Block. Datalohko (tietueet) sisältää erilaisia muuttujia, kuten numerotietoja, tekstitietoja tai muita tietotyyppejä. Sitä käytetään tiedon tallentamiseen ja jakamiseen eri ohjelman osien välillä.

FB: Function Block eli toimilohko, joka kapseloi useita ohjelmalohkoja ja muuttujia. Samanlaista toiminnallisuutta voidaan kutsua eri osissa ohjelmaa tai eri projekteissa.

FC: Function Chart joka käyttää graafisia elementtejä, kuten palkkeja ja liitäntöjä, kuvaamaan ohjelman toimintaa.

DLL: Dynamic Link Library on Windowsissa kirjasto, joka sisältää ohjelmakoodia, ja sitä voivat käyttää useat ohjelmat yhteisenä resurssina.

1 Johdanto

Insinöörityön tavoitteena on selvittää ja hyödyntää Siemens Osakeyhtiön TIA Portal -ympäristön tarjoaman TIA Openness -rajapinnan etuja asiakasyritys Cimcorp Oy:n automaatiosuunnittelussa sekä standardikirjastojen luonnissa.

Openness-rajapinnan avulla saadaan toistuvia tehtäviä automatisoitua. Näitä olisivat muun muassa taajuusmuuttujien konfigurointi, automaatiojärjestelmän ohjelmalohkojen toteutus sekä kokonaisuuden verkotukset, jotka tehostavat sovelluskäyttäjän työaikaa manuaalisen työn jäädessä vähemmälle.

Asiakasyritys Cimcorp Oy käyttää valmiita itse luomiaan omia standardikirjastoja TIA Portal -kehitysympäristössä, jotta samaa ohjelmakoodia sekä konfiguraation tekoa ei tarvitse kirjoittaa ja määritellä toistuvasti. Haasteeksi on kuitenkin koettu kirjastojen manuaalinen käyttö TIA-ympäristössä. Ratkaisuna on automatisoida kirjastojen avulla luotuja ohjelmalohko-objekteja TIA Openness -rajapinnan kautta.

Insinöörityön teoreettinen viitekehys käsittelee TIA Portal -ympäristön tarjoaman Openness-rajapinnan ominaisuuksia ja sen tuomia etuja suunnittelutyön tehostamiseksi automaatioprojekteissa.

Työssä perehdytään käsitteisiin TIA Portal, TIA Openness, Excel, MS Visual Studio -työkalut ja niihin kuuluvat ohjelmointikielet sekä esitellään lopputuloksena asiakasyrityksen kanssa yhdessä kehitetty TIA Openness -sovellus.

Insinöörityön ulkopuolelle on rajattu yksityiskohdat sekä niiden toiminnallisuudet TIA Portal -ympäristön sisältämien eri sovelluskehittimien toiminnoista. Automaatiosuunnittelu, TIA Portal, TIA Addon ja XML-merkintäkieli käsitellään yleisellä tasolla. Aineistoa on kerätty verkosta, kirjoista ja haastatteluilla.

2 Lähtökohdat

Cimcorp Oy on rengasteollisuuden automaatiojärjestelmien johtava toimittaja koko maailmassa. Yritys on erikoistunut robotiikkaan perustuvien automaatiojärjestelmien toimittamiseen rengas-, elintarvike- ja jakeluteollisuudelle. Cimcorp Oy suunnittelee, ohjelmoi sekä kokoaa robotteja, joista valtaosa menee vientiin. Yhtiö on perustettu vuonna 2003. [1.]

Cimcorp-konsernin omistaa japanilainen Murata Machinery, Ltd., jonka pääkonttori sijaitsee Ulvilassa ja tytäryhtiöt Kanadassa, Yhdysvalloissa, Intiassa sekä Espanjassa. Cimcorpin Suomen huoltopisteet toimivat Helsingissä, Lahdessa ja Jyväskylässä. Keväällä 2019 Ulvilassa työskenteli 330 henkilöä ja Pohjois-Amerikassa 100 henkilöä. Vuonna 2020 konserni työllisti yli 500 henkilöä. [1.]

Yrityksen toiminta on laajentunut merkittävästi viime vuosina, ja sen automaatiojärjestelmiä on toimitettu yli 40 eri maahan ja kuuteen maanosaan. Cimcorp Oy:n asiakaskuntaan kuuluvat useat suuret rengas- ja elintarvikeyritykset, kuten Continental, Goodyear, Cordiant, Tigar Tyres, ICA, Mercadona, Olvi, Sinebrychoff, Posti, Fazerin Leipomot ja Valio. [2.]

Yrityksen liikevaihto on kasvanut huomattavasti, ja vuonna 2020 se oli 94,5 miljoonaa euroa. Cimcorp erottautuu kilpailijoistaan laadukkaiden ratkaisujensa avulla. [2.]

2.1 Aihe ja tavoite

Insinöörityön aihe lähti siitä, kun sain työharjoittelupaikan Siemens Osakeyhtiöstä ja minulle esitettiin Cimcorp Oy:tä asiakkaaksi. Tarvetta oli toteuttaa asiakkaalle sovellus, joka poistaa toistuvaa työtä. Ajatus herätti mielenkiintoni, ja niin aloitin insinöörityön tämän aiheen parissa.

Kehitystyön suunnitteluvaiheessa laadittiin Cimcorp Oy:n kanssa tavoitteet sovelluksen toteuttamiselle. Tavoitteena oli kehittää sovellus, jolla kyetään tuottamaan Tia Portal -ympäristöön yksinkertaisia automatisoituja ohjelmalohkoja yksinkertaisella käyttöliittymällä. Ratkaisun tulisi olla niin helppokäyttöinen, että sen omaksuminen ei vaadi ymmärrystä ohjelmoinnista.

TIA Openness -sovelluksen yksi keskeisistä ajatuksista on tehdä automatisoinnista niin helppoa, että ohjelmalohkojen luontia pystyy kohtuullisen pienellä vaivalla automatisoimaan kuka tahansa automaatiosuunnittelija, joka hallitsee perustasoiset IT-puolen käyttötaidot.

Automatisoinnin tuominen käyttäjien itsensä ulottuville mahdollistaa sen, että usein rajallisia ja kalliita alan asiantuntijaresursseja voidaan vapauttaa aikaa syövien perusrutiinien suorittamisen sijaan haastavampien ongelmien ratkaisuun.

Työn aiheena olevan sovelluksen nähtiin täydentävän hyvin yrityksen tarpeita ja tukevan osaltaan näkemystä helposta ja tehokkaasta automatisointiprosessista. Sovellus nopeuttaa ja tehostaa myös alan ammattilaisten työskentelyä.

Markkinoilla on valmiiksi olemassa sekä kaupallisia että avoimen lähdekoodin ratkaisuja perustuen TIA Openness -rajapinnan tekniikoihin muun muassa ”TIA Portal Openness Scripter” ja ”TIA Portal Openness Explorer”, mutta näiden lähestymistapa ei sellaisenaan soveltunut Cimcorp Oy:lle. Tutkimalla valmiita ratkaisuja todettiin myös, että jonkin olemassa olevan tuotteen muokkaaminen yrityksen käyttämälle tekniikalle sopivaksi olisi joka tapauksessa niin mittava työ, että kokonaan oman ratkaisun kehittäminen koettiin järkevimmäksi ratkaisuksi.

Henkilökohtaisena tavoitteena on oppia enemmän TIA Openness -ohjelmointirajapinnan mahdollisuuksista automatisoida työvaiheita, Python-ohjelmointia ja TIA Portal -ohjelmointityökalun käyttöä.

2.2 Rajaus

Ohjelmointityökaluksi rajattiin Siemens Osakeyhtiön kehittämä TiaPortal -ympäristö, joka keskittyy Step7 Professional -ohjelmistoon. WinCC- ja HMI-laitteet jätettiin tarkastelun ulkopuolelle. Räätälöidyn sovelluksen kehitysympäristöksi valittiin Visual Studio ja ohjelmointikieleksi Python oman kokemuksen ja Siemensin TIA Openness -ohjelmointirajapinnan soveltuvuuden takia. Excel-tiedoston formaatti rajattiin Excel-työkirjamuotoon (.xlsx).

2.3 Tutkimuskysymykset

Tutkimusmateriaalin kerääminen alkoi pyytämällä Siemens Osakeyhtiöltä TIA -ohjelmistot, jotka asensin virtualisoituun ympäristöön. Siemens Osakeyhtiön kautta saatiin tunnukset SiePortal-palveluun, josta löytyi huomattava määrä kirjallisuutta aiheesta. Kehitystyön aikana ohjelmistojen käsikirjoista selvisi tarvittavat ominaisuudet (properties), komennot (method) ja tapahtumat (events), joita opinnäytetyön teon aikana tuli laajasti tutkittua ja testattua rajapintojen keskinäistä toimintaa ja yhteensopivuutta. Unit -testaukseen käytettiin Visual Studion -kehitystyökaluja.

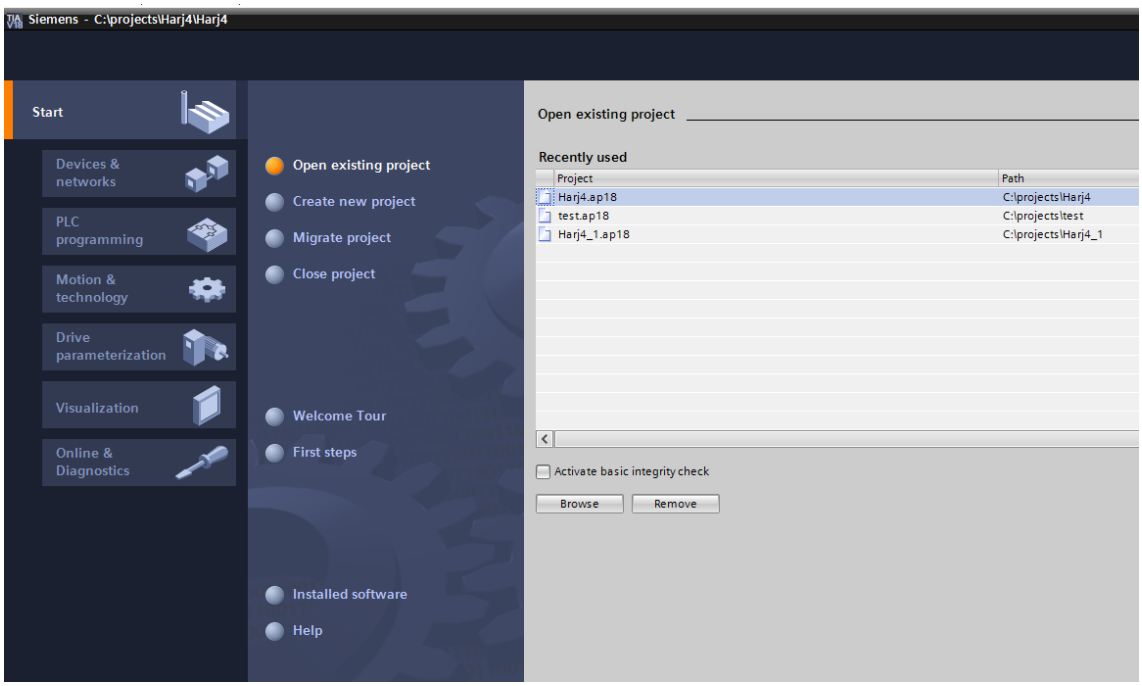
Tutkimusvaiheen aikana haastattelin loppuasiakasta ja selvitin heidän näkemyksensä käytön tavoitteesta ja sain kuulla havaintoja aikaisemmasta toteutuksesta.

Lopputyön lähtökohtana oli selvittää vaatimusmäärittelyt, joiden pohjalta yhteistyössä Siemens Osakeyhtiön ja loppuasiakkaan edustajien kanssa ilmaantui seuraavia näkökulmia.

- Voiko uudistuneen TIA Portal -version sisältämää TIA Openness -rajapintaa käyttää tehokkaasti generoimaan sovelluslohkoja ja laitteiden konfiguraatiota ja miten toiminnollisuudet ovat kehittyneet?
- Voidaanko automaattisella generoinnilla välttää virheitä projektin laadun parantamiseksi?
- Voidaanko rajapinnan avulla saada standardointi ja kirjastolohkojen käyttö seuraavalle tasolle?
- Luoko Openness -tekniikka tulevaisuudessa automaation ohjelmoinnin mahdolliseksi käyttäen IT-maailmasta tuttuja työkaluja ja menetelmiä alkaen versionhallinnasta Git:n avulla päättyen Python skriptauksen avulla generoitujen ohjausjärjestelmän toimilohkojen luontiin?
- Täyttääkö TIA Portal Openness -teknologia nykypäivän koneautomaation ja kyberturvan näkökulmasta projektihallinnan vaatimukset?

3 Automaatiosuunnittelu TIA Portal -ympäristössä

TIA Portal on Siemens Osakeyhtiön kehittämä automaatiosuunnittelun ohjelmistokokonaisuus, joka tarjoaa työkalut ohjausjärjestelmien ohjelmointiin, konfigurointiin ja diagnostiikkaan. Tällä hetkellä sille ei ole verrattavissa olevaa kilpailijaa markkinoilla. Yksi syy tähän on, että TIA Portal on ainoa ohjelmisto, joka sisältää yhteisen alustan ohjaussovellusten kehittämiseen ja visualisointien luomiseen. TIA Portal yhdistää logiikkajärjestelmien ohjelmoinnin, käyttöliittymien visualisointien luomisen operointipaneleille (HMI) sekä SCADA-luokan valvomoiden kokonaiskehityksen sisältäen muun muassa keskitetyn käyttäjähallinnan. Luonnollisesti kun kaikki on yhdessä kehitysympäristössä, työskentely on tehokkaampaa kuin osien ollessa erikseen. Lisäksi kehitysympäristö antaa erittäin miellyttävän käyttökokemuksen ja vaikutelman sitä käytettäessä. Kaikki on järjestetty selkeästi, kaikilla asetuksilla aina ohjelmoitavan logiikan konfiguroinnista visualisointeihin on sama periaate ja niiden lisääminen ja muokkaaminen on erittäin helppoa. [3.] Kuvassa 1 on esimerkkinäyttö projektista, jossa yksittäisten välilehtien avulla päästään heti helposti kaikkiin projektin osiin.



Kuva 1. TIA Portal -näkymä.

TIA Portal -kehitysympäristö kattaa useita sovellusohjelmia mukaan lukien Step7 Professional, Step7 Safety Advanced, WinCC Comfort/Advanced/Unified/Professional sekä StartDrive. Näihin ohjelmistoihin sisältyy valmiita kirjastolohkoja, jotka on suunniteltu nopeuttamaan sovelluskehitystä ja kyseisiä ohjelmalohkoja käytetään usein monenlaisten toimintojen suorittamiseen esimerkiksi datan siirtoon, säätöpiireihin ja liikkeen ohjaukseen. [4.]

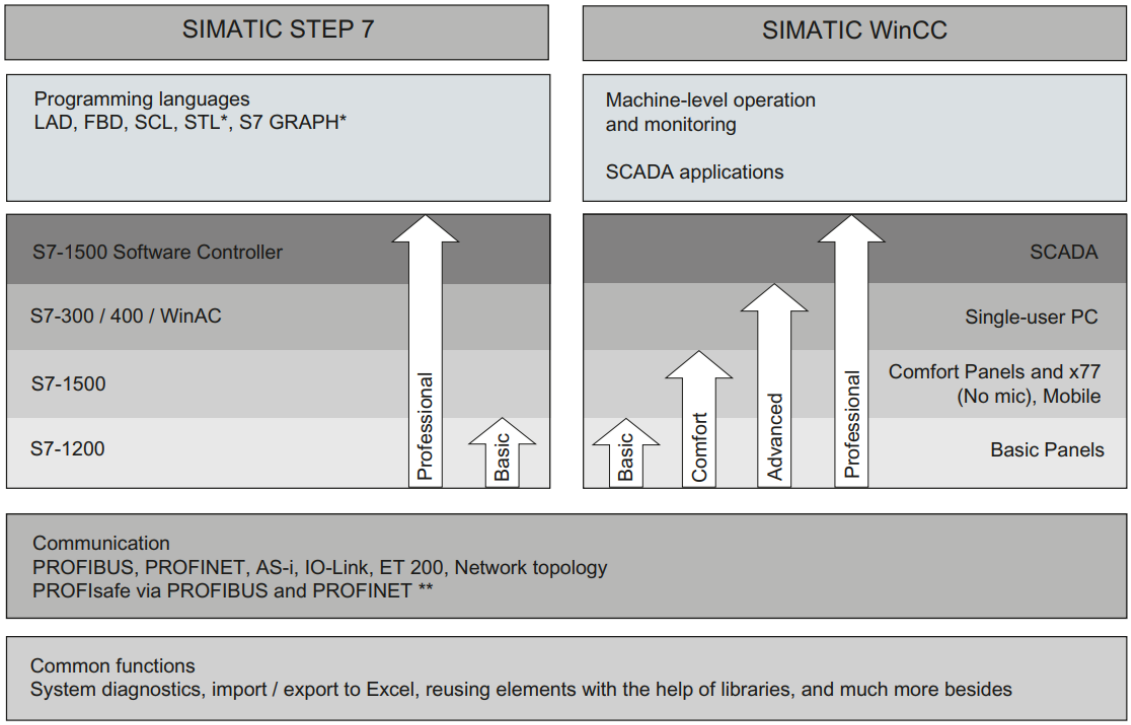
Step7 Professional -ohjelmistoa käytetään logiikkaohjelmointiin malleille S7-1500/1200/ET200SP sekä vanhemmille malleille S7-300/400. Käytettävissä olevat ohjelmointikielet ovat LAD (tikapuu), FBD (toimilohko), SCL (lausekieli), STL (käskylista) ja S7 GRAPH (sekvenssi). Kielet noudattavat IEC 61131-3 -standardia.

Step7 Safety Advanced -pakettia käytetään turvaohjelmien ohjelmointiin ja yhdessä turvalogiikan kanssa näillä saadaan toteutettua applikaatio kattaen nykyisten EU-konedirektiivien vaatimukset. [5.]

WinCC-ohjelmistoa käytetään HMI-näyttöjen ja valvomojärjestelmien kehittämiseen. Sillä luotua sovellusta voidaan ajaa Windows/Linux-tietokoneissa, kiinteästi asennetussa kovennetuista operointipaneleissa sekä kannettavassa mobiilipanelissa. [5.]

StartDrive -ohjelmistoa käytetään Sinamics-taajuusmuuttajien konfigurointiin ja parametrien määritykseen. [5.]

Kuva 2 kokoaa Simatic Step7 / WinCC:n tärkeimmät ominaisuudet kuten ohjelmointikielet, verkotukset, kirjastoja ja monia muita toimintoja.



Kuva 2. Simatic Step7 / WinCC:n tärkeimmät ominaisuudet [6.]

3.1 SIMATIC S7-1500

Saksalaisen Siemensin kehittämä S7-1500 -logiikkaohjain (PLC) on tällä hetkellä uusin ja tehokkain ohjausjärjestelmä, jota käytetään teollisuuden tuotantoprosessien valvontaan ja ohjaukseen. S7-1500-tuoteperheen prosessorit ovat

nopeita, tarkkoja, omaavat korkean suorituskyvyn sekä helpon käytettävyyden ollen näin ollen optimaalinen ratkaisu monimutkaisiin automaatiotehtäviin. Käyttökohteita ovat muun muassa elintarviketeollisuus, kemian teollisuus, logistiikka sekä koneenrakennus. [7, s. 697-746.]

S7-1500-tuoteperhe sisältää integroidut turvaominaisuudet ja tuki löytyy useille eri kommunikaatioprotokollille kuten PROFINET, PROFIBUS ja OPC UA, jotka mahdollistavat helpon integroinnin muihin järjestelmiin horisontaalisesti sekä vertikaalisesti. IT/OT-integraatiota varten S7-1500 sisältää edellä mainittujen lisäksi tuen RestApille, MQTT:lle, SQL:lle ja monelle muulle protokollalle ”open user communication” -kirjastolohkojen avulla. OUC:lla voi tarvittaessa luoda myös oman räätälöidyn TCP/IP- tai UDP/IP-pohjaisen protokollan. [7, s. 697-746.]

S7-1500 OPC UA Server/Client mahdollistaa kommunikoinnin eri valmistajien laitteiden välillä ja mahdollistaa tiedonsiirron eri järjestelmien välillä. OPC UA (Unified Architecture) on standardi teollisuuden tiedonsiirrolle ja mahdollistaa yhteensopivuuden eri protokollien välillä. [7, s. 73-84.]

S7-1500 WebServer -toiminto mahdollistaa web-pohjaisen käyttöliittymän luomisen HMI-laitteisiin mahdollistaen käyttäjän suorittaman prosessin ohjaamisen ja valvonnan missä tahansa, kunhan heillä on pääsy lähiverkkoon taikka internet vpn-tunnelin kautta. Toiminto on erityisen hyödyllinen hajautetuissa järjestelmissä, joissa käyttäjät voivat olla fyysisesti kaukana erillään laitteista. [7, s. 73-84.]

S7-1500 SQL -kirjastolohkopaketti mahdollistavaa tiedon tallentamisen ja noutamisen SQL-tietokannasta tarjoten tehokkaan tavan tallentaa ja käsitellä suuria määriä tietoa. [8.]

S7-1500 FTP-kirjastolohkopaketti mahdollistaa tiedostojen siirtämisen laitteiden välillä käyttäen File Transfer Protocol -protokollaa ja on hyödyllinen muun muassa prosessidatan raportoinnissa ylemmän tason järjestelmälle. [9.]

3.2 Sinamics-taajuusmuuttajat

Siemens Osakeyhtiö on kehittänyt sarjan nykyaikaisia ja huipputekniikkaa sisältäviä taajuusmuuttajia – SINAMICS, G120C, G120 ja S120. [10.]

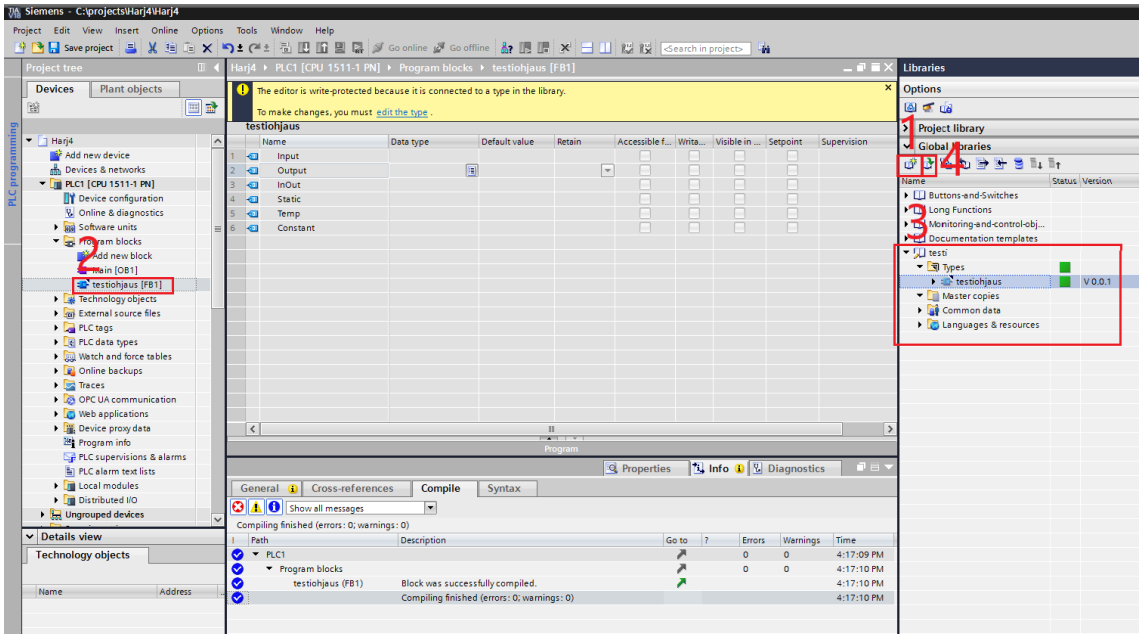
SINAMICS G120C on suunniteltu taloudelliseksi, tilaa säästäväksi ja helposti käytettäväksi taajuusmuuttajaksi, joka soveltuu pumppaamiseen, tuuletukseen ja/tai paineilman puristamiseen ja käyttöön kuljetinjärjestelmistä kattaen 0,75–150 hv (0,55–132 kW) tehoalueen. Sinamics-taajuusmuuttujat sisältävät integroidut diagnostiikkatoiminnot sekä nopean käyttöönoton, käyttäjäystävälliset liitännät ja helppokäyttöiset parametrointityökalut. [11.]

Kuva 3. Mobiililaitteilla etäyhteys taajuusmuuttujaan [10.]

3.3 Kirjastointi

TIA Portal -ympäristössä kirjastojen avulla voidaan luoda, hallita ja jakaa koodia ja muita ohjelmaelementtejä eri projekteissa. Kirjastot jakautuvat kahteen pääluokkaan: projektikirjastoihin ja globaaleihin kirjastoihin. Projektikirjastot sijaitsevat yksittäisten projektien sisällä ja niissä on usein versiointitarkoituksiin käytettyjä lohkoja. Toisaalta globaalit kirjastot toimivat keskitettyinä arkistoina erilaisille ohjelmalohkoille, tietorakenteille, tunnistetaulukoille tai vaikkapa

kokonaisille objekteille ollen logiikka, operaatiopaneli tai hajautettu IO-asema Profinet/Profibus-kenttäväylässä. [12.] Työn toteutuksessa hyödynnetään globaalikirjastoja.



Kuva 4. TIA Portal -projekti ja avattuna globaalikirjasto näkymä

Luodaan uusi globaali kirjasto. Ensiksi napsautetaan "Uusi kirjasto" -painiketta. Sen jälkeen avautuu pieni ikkuna, jossa voi määrittää kirjaston nimen, polun, tekijän ja kommentin. Tässä annamme kirjastolle nimeksi ”testi”, jonka jälkeen luodaan napsauttamalla "Luo".

Seuraavaksi kopioidaan ”testiohjaus”-ohjelmalohko TIA Portal -projektin oikealle puolelle ”Types” -nimiseen kansioon. Tämän jälkeen avautuu uusi ikkuna, jossa kysytään nimeä ja versionumeroa. Kuvassa 4 laitoin versioksi ”v0.0.1” ja nimeksi ”testiohjaus”. Jos haluaa ”testiohjaus” ohjelmalohkoon tehdä muutoksia, toistetaan samat vaiheet kuin edellä ja kirjoitetaan ”v0.0.1” version sijaan esim. ”v0.0.5” versioksi. Samaan globaalikirjastoon pääsee käsiksi toisessa projektissa napsauttamalla ”Avaa”, josta valitaan ”testi”-kirjastotiedosto.

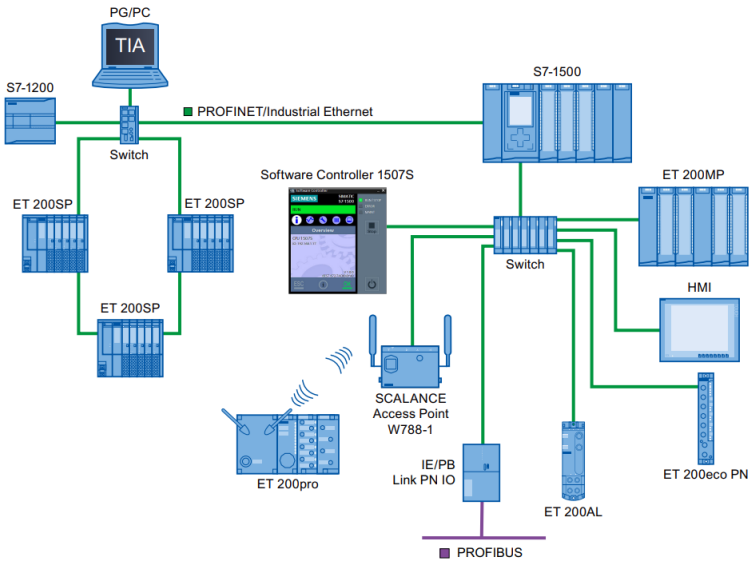
3.4 HMI-panelit Comfort / WinCC Unified

TiaPortal WinCC on tarkoitettu operaatiopaneelien ja valvomosovellusten luontiin. Tia Openness avulla näihin voidaan luoda/muokata/poistaa hmi tageja, hälytyksiä, kuvasivuja ja monia muita objekteja saman rajapinnan kautta. Tässä insinöörityössä ei käsitellä laajemmin HMI-tuotteita, mutta mahdollisuudet ovat laajat sovelluksen jatkokehitystä ajatellen.

3.5 Profinet

Profinet on Siemens Osakeyhtiön kehittämä teollisuusverkkoprotokolla, joka on integroitu Tia Portal -ympäristöön, joka pohjautuu Ethernet-standardiin ja mahdollistaa tehokkaan kommunikoinnin eri laitteiden ja järjestelmien välillä teollisuusympäristössä. Profinet tunnetaan avoimuudestaan ja yhteensopivuudestaan, mikä tekee siitä ihanteellisen ratkaisun monenlaisiin teollisuusautomaatiotarpeisiin. Profinet tarjoaa reaaliaikaisen, nopean ja luotettavan tavan integroida erilaisia laitteita ja järjestelmiä automaatioympäristössä. [13, s. 15–16.]

Profinet IO on teollisuuden ethernetin (Industrial Ethernet) looginen jatke, joka on integroitu kokonaisvaltaisesti Totally Integrated Automation (TIA) -konseptin alle. Profinet IO pohjautuu Profibus DP:n menestykselliseen käyttöhistoriaan lisäten siihen innovatiivisia Ethernet-teknologian konsepteja, mikä mahdollistaa näiden kahden tekniikan yhdistämisen saumattomasti. [13, s. 19–41.]



Kuva 5. Profinet integroitu laitteisiin [13, s. 16.]

Insinöörityössä luodussa asiakassovelluksessa hyödynnetään Profinetin verkkotopologiaa verkottamaan dynaamisesti muun muassa logiikkaohjaimia ja taajuusmuuttajia keskenään TIA Openness -rajapinnan kautta, jolloin vältytään toistuvalta manuaaliselta konfiguraatiotyöltä.

3.6 IEC 61131-3 -ohjelmointikielet

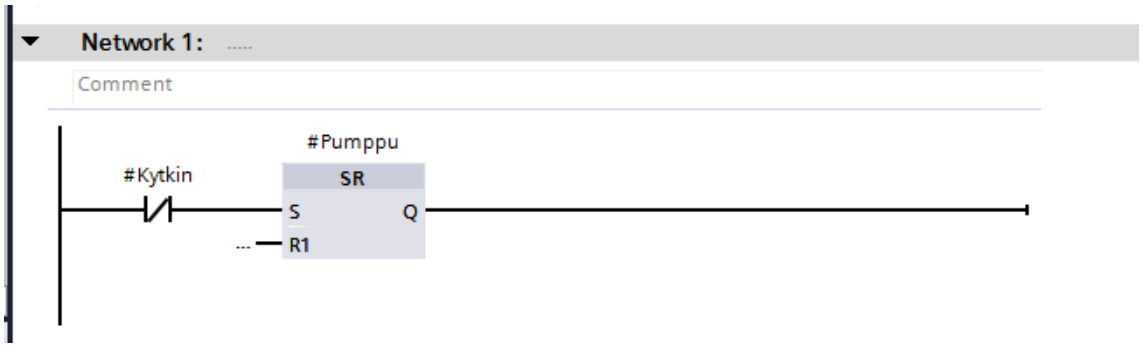
Jokaisella automaatiovalmistajalla on käytössään ohjelmointikieliä, jotka on tarkoitettu heidän omien kehitysympäristöjen PLC-ohjelmointiin. Nämä kielet ovat periaatteeltaan samankaltaisia, mutta kuitenkin toiminnoiltaan spesifisiä eivätkä siten ole siirrettävissä eri valmistajien eri kehitysympäristöjen välillä.

Yhteensopivuuden harmonisoinnin tukena on olemassa kansainvälinen standardi IEC 61131-3, joka yhdistää kielet viiteen ohjelmointikielityyppiin. Nämä ovat LD - Ladder Diagram, FBD - Function Block Diagram, IL - Instruction List ja ST - Structured Text sekä SFC - Sequential Function Chart. Kielet jaetaan niiden muodon perusteella graafisiin ja tekstipohjaisiin. Graafisiin kieliin kuuluvat LD, FBD ja SFC. Tekstipohjaisiin kieliin kuuluvat ST ja IL. [14.]

3.6.1 Ladder Diagram

Tikapuukaavio (Ladder Diagram) kuuluu graafisiin ohjelmointikieliin. LD perustuu relelogiikan periaatteeseen. Ohjelma kirjoitetaan yksittäisiin virtapiireihin, joita kutsutaan pikemminkin verkoiksi (network) lisäämällä elementtejä kuten kytkentä- ja irrotuskoskettimia, aliohjelmatoimintoja, toiminnallisia kompleksisia lohkoja ja sekä kutsuja muihin ohjelmalohkoihin. Ladder Diagram -kieli on aloittelijoille sopiva yksinkertaisuutensa vuoksi. Monimutkaisemmissa sovelluksissa se kuitenkin muuttuu epäkäytännölliseksi ja epäselväksi. [14.]

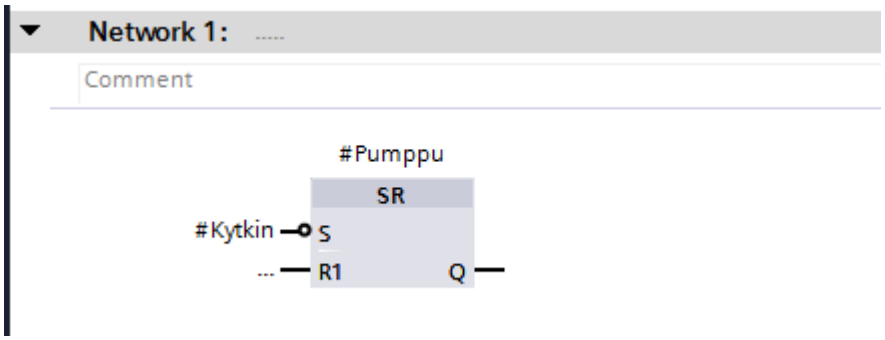
Kuvassa 6 on esimerkki LAD-kielestä, joka kytkee pumpun päälle tai pois. Tämä edustaa standardisoitua LD-kieltä, joka on luotu Siemens Osakeyhtiön TIA Portal -ohjelmointiympäristössä. Kuvasta käy ilmi, että kyseessä on selkeä ja havainnollinen ohjelmointikieli.



Kuva 6. Tikapuukaavio, jossa on kytkin ja eteenpäin lukemiseen tarkoitettu toiminnallinen lohko ”Pumppu”.

3.6.2 Function Block Diagram (FBD)

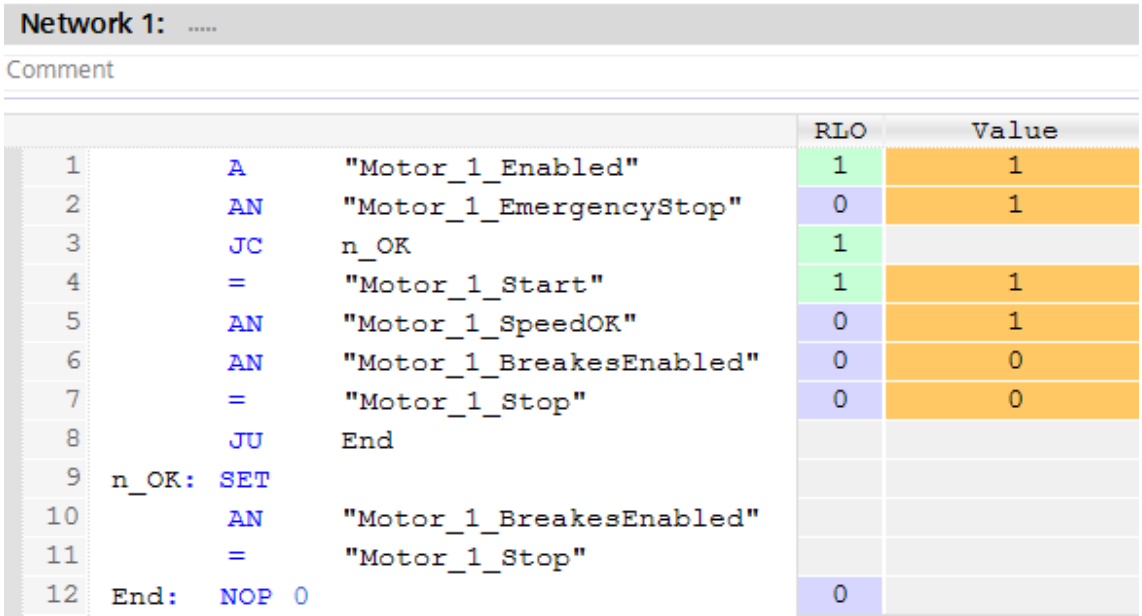
Function Block Diagram (FBD) on graafinen ohjelmointikieli, joka käyttää lohkokaavioesitystä kuvaamaan ohjelman logiikkaa. [14.]



Kuva 7. Edellisen kuvan toteutus lohkokaavion muodossa

3.6.3 Instruction List (IL)

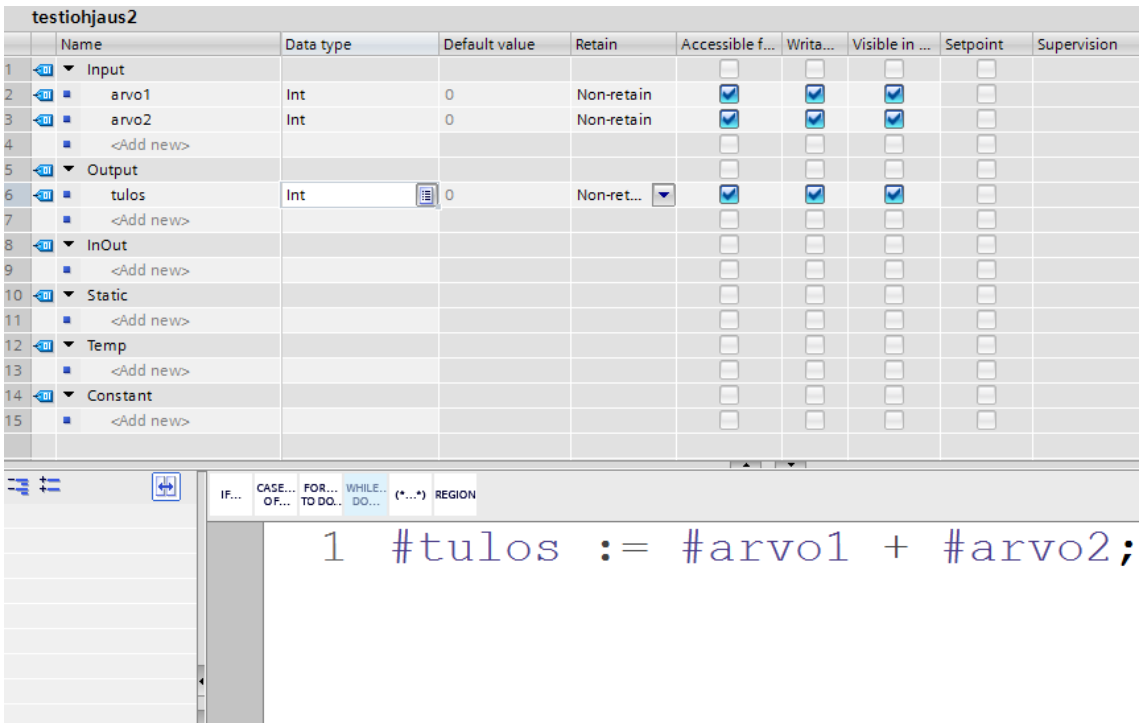
Instruction List (IL) on alemman tason tekstipohjainen ohjelmointikieli, joka on samanlainen kuin assembler. Sitä käytetään harvemmin, mutta se on hyödyllinen erittäin aikakriittisissä sovelluksissa. Nykyään uusien ohjelmien kirjoittamisessa pyritään välttämään IL-ohjelmointia ohjelman hankalan luettavuuden vuoksi. [14.]



Kuva 8. STL-kielellä toteutettu moottorin ohjaus sisältäen vertailuja ja hyppykäskyjä. Virtapiiristä voidaan todeta rekisteriakun oloarvot.

3.6.4 Structured Text (ST)

Structured Text, kuten nimi viittaa, kuuluu tekstipohjaisiin ohjelmointikieliin. Kielen rakenne on samankaltainen kuin esimerkiksi C-kielessä määritettyjen ilmausten ja käskyjen avulla. Toisin kuin IL-kielessä, ohjelmoijan ei tarvitse tuntea suurta määrää käskyjä ja ohjelma on myös selkeämpi. Structured Text -kieltä pidetään tehokkaana ohjelmointikielenä. Selkeyden vuoksi sitä käytetään monimutkaisempiin ohjelmiin. [14.] Kuvassa on esimerkki SCL-kielestä TIA Portal -kehitysympäristössä, joka perustuu ST-kielen standardiin. Vaikka tämä ohjelma ei ole visuaalisesti niin selkeä, se on edullisempi monimutkaisempiin ohjelmiin, kuten suurien laskentatehtävien kirjoittamiseen, jolloin ohjelman kirjoittaminen käyttäen LD- tai FBD-kieltä olisi hyvin monimutkaista.

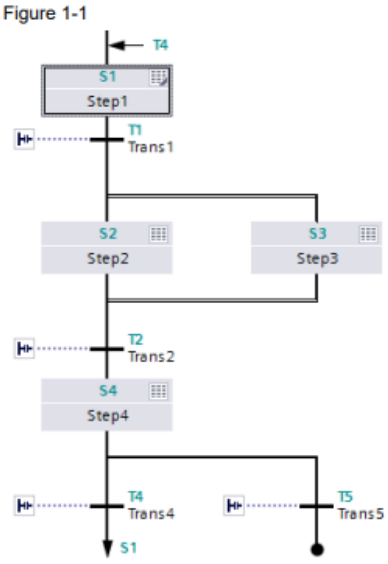


Kuva 9. Esimerkki SCL-kielestä TIA Portal -kehitysympäristössä.

3.6.5 Sequential Function Chart (SFC)

Sequential Function Chart (SFC) on graafinen ohjelmointikieli, joka kuvaa ohjelman toiminnan sekvenssikaaviona. Kieli sopii erityisen hyvin prosessien

kuvaamiseen, jotka etenevät määrätyssä järjestyksessä ja vaiheissa. Esimerkiksi voimme käyttää SFC:tä kuvaamaan tuotantolinjan toimintaa, jossa raaka-aineet käyvät läpi useita tuotantovaiheita. [14.]



Kuva 10. Esimerkki SFC -kielestä TIA Portal -kehitysympäristössä.

TIA Portal -ympäristössä GRAPH on sekvenssifunktionaalinen graafinen ohjelmointikieli, joka noudattaa SFC-kielen standardia. GRAPH on suunniteltu ja soveltuu parhaiten tilakoneiden ohjelmointiin.

GRAPH koostuu käynnistyslohkosta, välilohkoista sekä lopetuslohkosta. Välilohkot voivat jakautua useisiin haaroittuviin osiin ja näitä voi olla useita. Lohkot edustavat yksittäisiä tiloja ja kullekin lohkolle on liitetty toiminto, joka suoritetaan. Lohkojen välillä on askel, johon toteutetaan siirtoehto siirtymistä lohkojen välillä. Siirtyminen seuraavaan lohkoon on ehdollinen määritellyn ehdon perusteella. Jokainen elementti, eli siirtymä ja toimintolohko, voidaan ohjelmoida millä tahansa standardissa määritellyllä kielellä, mukaan lukien oma SFC. Usein käytetään esimerkiksi yhteyttä strukturoituun tekstiin. Yleensä käytetään useampia lohkoja, joista kukin suorittaa pienen osan koko tilakoneesta. [14.]

3.7 S7-1500-ohjelmarakenne

Automaatiosovelluksen ohjelma konfiguroidaan TIA Portal -kehitysympäristössä käyttäen dedikoituja ohjelmalohkoja. Ohjelmalohkot jaetaan neljään perustyyppiin (OB/FC/FB/DB). Ohjelman peruslohkot tunnetaan nimellä organisaatiolohkot (OB). Tietojen tallentamiseen, jotka liittyvät johonkin loogiseen kokonaisuuteen, käytetään nimeä datalohkot (DB). Toistuvien ja monikäyttöisten osien toteutusta varten on olemassa funktioita ja toiminnallisia lohkoja (FC/FB). [40, s. 323–338.]

3.7.1 OB (Organization block)

OB-lohkot ovat ohjelman peruslohkoja. Projektin luomisen yhteydessä luodaan automaattisesti myös päälohko (OB1). OB-lohkoja kutsutaan syklisesti ja niistä kutsujaan aliohjelmalohkoja. OB-lohkoja on useita eri tyyppejä. Näitä ovat muun muassa jatkuvan syklin OB, aikakeskeytysohjelma, diagnostiikkakeskeytys sekä laitetason hw-keskeytys. [15, s. 10–11; 40, s. 323–338.]

3.7.2 DB (Datablock)

Datalohkot toimivat tietojen tallentamiseen ja säilyttämiseen, jotka kuuluvat loogiseen kokonaisuuteen. Datalohkot sisältävät muuttujien arvoja, joita käytetään ohjelmassa. Datalohkoja voi olla ohjaimessa hyvin laaja määrä, ja ne luodaan siten, että ne sisältävät tietoja, jotka liittyvät toisiinsa. Jokainen ohjelman funktiolohko voi kirjoittaa tietoja datalohkoon tai lukea niitä. [15, s. 12; 40, s. 323–338.]

3.7.3 FB (FunctionBlock)

FB:t muodostuvat aliohjelmista, jotka suoritetaan aina, kun funktiota kutsutaan. Tällaista kutsua kutsutaan FB:n instanssiksi. Jokaiselle FB kutsulle on varattu datalohko, johon käsitellyt tiedot ja FB:n parametrit tallennetaan. Kyseisen datalohkon staattiset tiedot säilyvät tallessa myös FB:n suorittamisen jälkeen ollen

remanenttinen. FB-tyyppiä tyypillisesti käytetään silloin, kun tehtävän suorittamiseen tarvitaan enemmän kuin yksi funktio ja kun tietoja on säilytettävä FB-kutsun jälkeen jatkuvaa ohjelman suoritusta varten. [15, s. 11; 40, s. 323–338.]

3.7.4 FC (Function)

FC sisältää aliohjelman, joka suoritetaan pääohjelman kutsun jälkeen. Kutsuminen voi tapahtua myös toisesta FC/FB-lohkosta. FC-kutsuun ei liity minkäänlaista erillistä staattista muistia, jolloin funktion suorituksen jälkeen sen aikana käytetyt väliaikaistiedot menetetään. FC:ta voidaan kutsua useita kertoja eri osista ohjelmaa. Tämän vuoksi FC-tekniikka on sopiva, kun kirjoitetaan yksinkertaisia ohjelman osia, jotka toistuvat usein. [15, s. 11; 40, s. 323–338.]

4 TIA Portal Openness

Ohjelmointirajapinta eli API (Application Programming Interface) on lyhennys viestintäprotokollille ja aliohjelmille, joita erilaiset ohjelmatoteutukset käyttävät kommunikointiin toistensa kanssa.

> Ohjelmointirajapintoja voi ajatella eri ohjelmistojen välille tehtyinä siltoina, joita pitkin tieto kulkee edestakaisin ohjelmistojen ja organisaatioiden välillä. [16.]

Ohjelmointirajapinta käyttö pääsääntöisesti auttaa helpottamaan ohjelmoijan työtä tekemällä siitä suoraviivaisempaa. Ohjelmointirajapintasovelluksen välisessä kommunikaatiossa periaate on ottaa käyttäjän pyyntö, joka lähetetään sille palveluntarjoajalle, joka toteuttaa tarvittavan prosessoinnin pyynnölle ja lopulta lähettää palveluntarjoajan tuottamat tulokset takaisin käyttäjälle. [16.]

Avoin rajapinta on ohjelmointirajapinta, jonka kaikki ominaisuudet ovat julkisia ja jota voi käyttää ilman rajoittavia ehtoja. Edellytyksenä on myös rajapintakuvauksen ja dokumentaation oleminen avoimesti saatavilla. Avoimen rajapinnan käytön etuna on mahdollisuus laatia rajapintaa hyödyntävän ohjelma ilman rajapinnan toimittajan erillistä hyväksyntää tai pakollisia lisenssimaksuja. [17.]

TIA Portal Openness on avoin ohjelmointirajapinta, joka mahdollistaa TIA Portal -projektien muokkaamisen ulkoisen erillisohjelman avulla. TIA Openness -rajapinta koostuu kahdesta osasta:

- C#.NET-sovellus, joka käyttää rajapintaa.
- XML-tiedostot, joita käytetään ohjelmakoodin tuomiseen ja vientiin TIA Portalista.

Ulkoinen ohjelma hyödyntää TIA Portal Openness-rajapinnan XML-tiedostojen tuontia ja vientiä. Rajapinta on avoin osa TIA Portal -ympäristön logiikasta.

Ensimmäinen vaihe TIA Portal Openness -sovelluksen luomisessa on määritettävä Windowsin käyttäjätileistä "Siemens Openness user" -ryhmään tietokoneen asetuksissa. Tämä antaa käyttäjän oikeuden käyttää rajapintaa. Sovellus voidaan ohjelmoida Visual Studiolla, jonka käyttö edellyttää Siemensin omaa DLL-tiedostoa, joka on ensin lisättävä Visual Studio -projektin riippuvuuksiin. [18.]

TIA Openness käyttää toimintoja TIA Portalin kautta Siemens.Engineering.dll- ja Siemens.Engineering.Hmi.dll-kirjastoissa. Nämä kirjastot sisältävät yksittäiset luokat, jotka on jaettu niiden tarkoituksen mukaisesti. Näiden luokkien avulla voidaan esimerkiksi käynnistää TIA Portal taustalla (WithoutUserInterface) tai käyttöliittymän kanssa (WithUserInterface). Taustalla suorittaminen on tehokkaampaa, koska yksittäiset toimet suoritetaan sen jälkeen nopeammin. Käynnistyksen jälkeen voidaan suorittaa tavallisia projektitöitä, kuten niiden luomista, avaamista, tallentamista ja sulkemista. Lisäksi voidaan hallita laitteita, visualisointeja, yksittäisiä lohkoja, tunnisteita ja käyttää monia muita TIA Portalin ominaisuuksia. TIA Portal Openness mahdollistaa lisäksi yksittäisten kohteiden, kuten tunnustaulukoiden, datalohkojen ja toiminnallisten lohkojen, vienti- ja tuontimahdollisuuden XML-tiedostoihin. [18; 19.]

TIA Portal Openness avulla voidaan merkittävästi automatisoida prosesseja, toteuttaa TIA Portaalin käskyjä etäsessiona, mikä tekee ympäristön käytöstä tehokkaampaa ja nopeampaa. Tia Openness avulla virheiden määrä

minimoituu, tehtävien suorittaminen nopeutuu ja näin ollen yrityksen kilpailukyky paranee sovellusten ollessa laadukkaampia, mikä liittyy myös tuotantokoneiden tehokkaampaan käyttöön. Merkittävä suurin etu on tuotantokoneiden käyttöönottoajan lyhentäminen. Yleisimpiä käyttötapoja ovat esimerkiksi projektien automaattinen generointi useille eri tuotantokoneille.

TIA Portal Openness -rajapinnan yksi tärkeimmistä ominaisuuksista on sen avoimuus, mikä tarkoittaa kykyä integroida TIA Portal -ympäristö helposti muihin järjestelmiin mahdollistaen luoda omia räätälöityjä sovelluksia, joiden avulla kuka tahansa voi laajentaa TIA Portalin toiminnallisuutta.

4.1 TIA Openness -rajapinnan kehitys

Ennen TIA Portal Openness -rajapinnan olemassaoloa järjestelmät eivät olleet kunnolla yhteensopivia ja integroituivat vaikeasti toisiinsa. Tämä tarve johti Siemens Osakeyhtiön päätökseen kehittää rajapinta, joka mahdollistaisi eri valmistajien laitteiden yhteensopivuuden Siemensin automaatiojärjestelmien kanssa.

TIA Portal Openness -rajapinnan ensimmäinen versio julkaistiin vuonna 2011 ja siitä lähtien Siemens on jatkuvasti kehittänyt ja päivittänyt rajapintaa vastaamaan käyttäjien tarpeita ja parantamaan automaatioinsinööritöiden tehokkuutta. [21, s. 1–2.]

Ennen TIA Portal Openness -rajapintaa Siemensin TIA Portalissa käytettiin muita ohjelmointirajapintoja, kuten API-interfacea WinCC:lle ja STEP7 5.x:lle. [21, s. 1–2.]

Vanhat versiot rajapinnoista eivät tarjonnut samanlaista laajuutta ja monipuolisuutta kuin nykyinen TIA Portal Openness. Sen avulla voidaan käyttää API-metodeja monipuolisesti TIA Portalin eri toiminnallisuuksiin, kuten projektien luomiseen, tarkistamiseen ja ylläpitämiseen. Lisäksi TIA Openness tarjoaa mahdollisuuden vaihtaa ohjelmisto- ja laitetietoja SimaticML- ja AutomationML-formaateissa, mikä helpottaa eri järjestelmien ja laitteiden yhteensopivuutta ja integrointia. [21, s. 1–2.]

4.2 Rajapinnan toiminnallisuudet

Siemens on jatkuvasti kehittänyt TIA Openness -rajapinnan toiminnallisuuksia jokaisen uuden TIA Portal -version julkaisussa. Tämä on johtanut edistysaskeliin useissa ohjelmointikieliä sisältävien ohjelmalohkojen käsittelyssä ja logiikan hallinnassa. [22.]

PLC-laitteiden käyttö TIA Openness -rajapinnan kanssa mahdollistaa laitteiden välisen tiedonvälityksen ja datan tallentamisen. PLC-laitteet ovat tärkeä osa teollisuuden automaatiojärjestelmiä ja niiden avulla voidaan ohjata muun muassa tuotantolinjoja. TIA Openness -rajapinnan avulla PLC-laitteiden toimintoja voidaan ohjata ja seurata HMI-laitteiden avulla. [23, s. 7–8.]

HMI-laitteiden käyttö TIA Openness -rajapinnan kanssa mahdollistaa laitteiden välisen tiedonvälityksen ja datan tallentamisen. HMI-laitteet tarjoavat käyttäjälle käyttöliittymän, jonka avulla laitteiden toimintoja voidaan ohjata ja seurata. HMI-laitteet mahdollistavat myös laitteiden välisen tiedonvälityksen, joka on tärkeää esimerkiksi teollisuuden automaatiojärjestelmissä. [23, s. 7–8.]

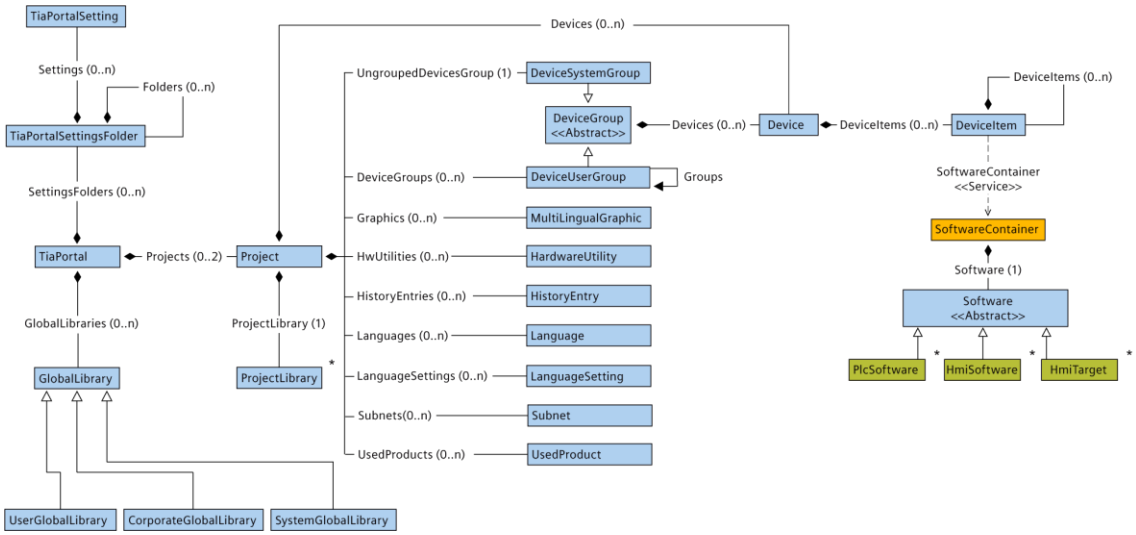
TIA Portal Openness-rajapinnan avulla käyttäjät voivat luoda, muokata, lukea ja poistaa projektissa olevia tietoja. Lisäksi luettua dataa voidaan kätevästi hyödyntää myös muissa projekteissa. Siemens tarjoaa laajan valikoiman toimintoja rajapinnan dokumentaation kautta. Dokumentaatio kattaa satoja metodeja, joista ohjeistuksen voi käydä lataamassa ilmaiseksi yrityksen verkkosivuilta. [23, s. 7–8.]

TIA Portal Openness -integrointi tapahtuu DLL-tiedostojen avulla. Nämä ovat dynaamisia linkkikirjastoja, joita käytetään ohjelmistojen sisäisten toimintojen toteuttamiseen. Siemensin Openness-rajapinnassa tarvittavat DLL-tiedostot ovat "Siemens.Engineering.dll" ja "Siemens.Engineering.Hmi.dll. Nämä tiedostot sisältävät koodia, joka mahdollistaa Openness-rajapinnan käytön yhdessä TIA Portalin kanssa. "Siemens.Engineering.dll" sisältää useita toimintoja, joita tarvitaan TIA Portalin tiedonhallintaan, kuten projektien avaaminen ja tallentaminen.

"Siemens.Engineering.Hmi.dll" sisältää toimintoja, jotka tarvitaan HMI-järjestelmien ohjelmointiin, kuten näyttöjen luominen ja asettaminen. [23, s. 28–29.]

TIA Portal Openness-rajapinnan "import/export"-käyttö perustuu objektien vientiin ja tuontiin XML-formaatin välityksellä, jossa objektit muunnetaan XML-muotoon ja takaisin objekteiksi. Tätä menetelmää käytetään erilaisiin osiin, kuten toimilohkoihin (FB), funktioihin (FC), tietueisiin (DB), verkkotopologiaan ja lukuisiin muihin komponentteihin. Lisäksi rajapintaa voidaan käyttää kirjastokomponenttien tuomiseen ja viemiseen projektiin, joko projektikirjastoissa tai globaalikirjastoissa. [23, s. 28–29.]

TIA Portal Openness -rajapinnan funktioilla on mahdollista muokata TIA Portal -ympäristöä kuten luoda uusi projekti tai avata jo olemassa oleva projekti sekä konfiguroida laitteita ja verkotuksia. Ympäristöä on mahdollista käynnistää joko ilman graafista käyttöliittymää tai graafisen käyttöliittymän kanssa. [23, s. 24–26.]

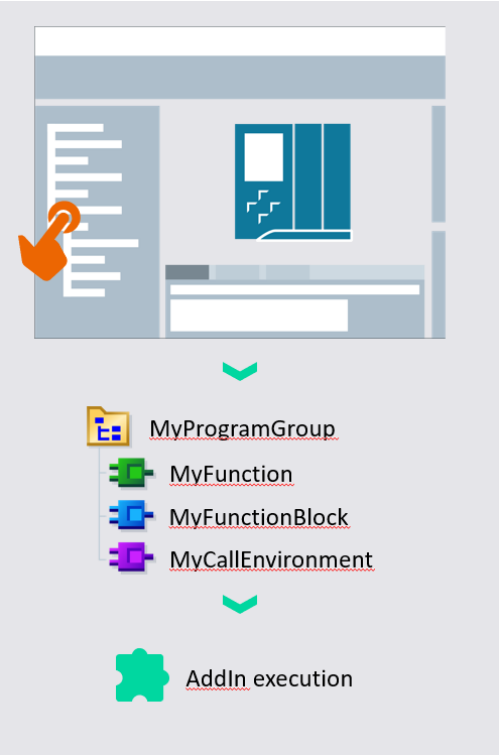


Kuva 11. TIA Openness -rajapinnan objektiluokkien relaatiot. [24.]

4.3 Lisäosat

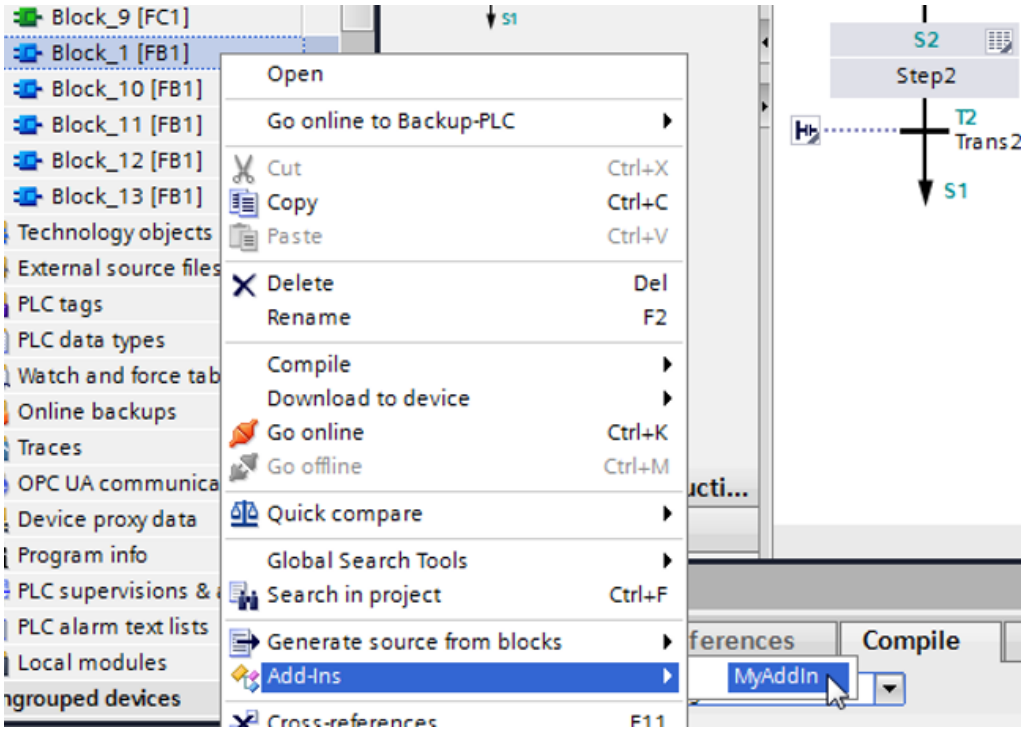
TIA Portal Addon -lisämodulit tarjoavat käyttäjälle kätevän tavan suorittaa ja laajentaa TIA-portaalin toimintoja Openness API:n avulla. Addonit voivat olla vuorovaikutuksessa TIA-objektien kanssa. Ne voivat suorittaa tiedosto-I/O-, verkkotoimintoja ja kutsua ulkoisia sovelluksia. [25.]

Esimerkiksi ”Excel Exporter” -addon mahdollistaa datan tallentamisen Excel-taulukkoon ollen hyödyllinen esimerkiksi datan analysoinnissa ja raportoinnissa. Toisaalta esimerkiksi ”Signal Exporter” -addon mahdollistaa signaalien tallentamisen CSV-tiedostoon.



Kuva 12. Kuvaus TIA Portal -lisäosan suorituksesta [24.]

Esimerkiksi TIA Portal Addon -modulin avulla voidaan TIA Portal -ympäristössä tehdä FC suoraan FB-lohkon muunnos ilman manuaalia toimenpiteitä kuten suorittamalla vanhasta lohkosta leikkaa-liimaa-tekniikalla muuttujien ja virtapiirien kopiointia uuteen lohkoon.



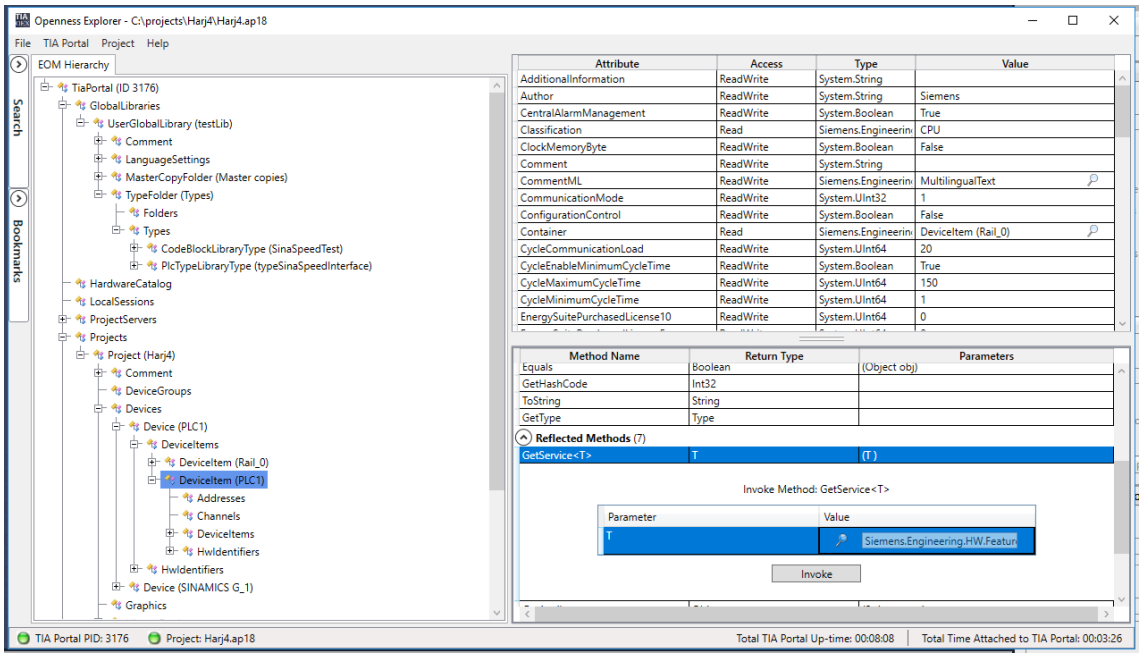
Kuva 13. TIA Portal -ympäristön projektipuu näkymä, josta valitaan ”MyAddin”

Verrattuna TIA Portal Openness -sovelluksiin TIA Portal Addon -modulit näkyvät valikoissa TIA ympäristössä ja ovat kontekstiherkkiä, mikä tarkoittaa, että ne voidaan ottaa hallitusti käyttöön tai poistaa käytöstä. TIA-portaalin käyttäjä voi käyttää addoneita ilman tietämystä Openness-rajapinnasta.

TIA Portal Addon -moduleita voidaan jakaa avoimesti joko käyttämällä ”Add-in Runner” tai ”Add-in API” -vaihtoehtoja. ”Add-in Runner” on kaikkien saatavilla ja tarjoaa mahdollisuuden suorittaa lisäosia TIA-portaalin kontekstissa, mutta vaatii käyttäjältä erillisen asennuksen. Lisäksi TIA Portal Openness täytyy olla esiasennettu. ”Add-in API” tarjoaa ympäristön TIA-apuohjelmien kirjoittamiseen ja julkaisemiseen mutta on saatavilla vain TIA-portaalin projektinavigoinnin ja kirjastonavigoinnin kontekstivalikossa. [25.]

4.4 Openness Explorer

”TIA Openness Explorer” -työkalun tarkoituksena antaa selkeä yleisnäkymä TIA Openness-rajapinnan objektimallista puumaisessa graafisessa muodossa. Työkalua voi käyttää ilman, että luo oman sovelluksen, ja tarkistaa, onko TIA portal -ympäristössä tämä tieto käytettävissä TIA Openness-rajapinnan kautta. Työkalua ei ole tarkoitettu käytettäväksi ohjelmien tekoon vaan suunnattu kehittäjille, jotka haluavat yleisnäkymän rajapinnasta ja nopeuttaakseen työtään. [26, s. 4.]



Kuva 14. ”TIA Openness Explorer” -näkymä

Hierarkiapuusta voidaan hakea laite, jolloin oikealla puolelle näkymää ilmestyy attribuutteja. Esimerkiksi voidaan “GetService”-attribuuttilla selvittää, mitä ohjelmalohkoja tietynlainen laite sisältää. Ohjelman avulla voidaan myös suorittaa metodeja (methods), jotka välittävät parametreja. [26, s. 4–6.]

Tässä opinnäytetyössä hyödynnettiin ”TIA Openness Explorer” -työkalua asiakkaan sovelluksessa, koska sen avulla löytyi kätevästi Openness-funktioita ja ominaisuuksia, joita kaikkia ei ollut mainittu nykyisessä dokumentaatiossa.

4.5 Openness Scripter

”Openness Scripter” on Siemens AG:n kehittämä työkalu, joka hyödyntää TIA Openness -rajapintaa. Sen avulla voidaan makroilla eli komentosarjakomennoilla automatisoida tehtäviä TIA Portal -projekteissa. Käyttäjältä ei vaadita ohjelmointiosaamista, mikä tekee ”Openness Scripterin” käyttämisestä helpompaa. [28, s. 4.]

Käyttäjä voi saada haluamansa alustuksen makroille ”Template Openness Scripter” -valmispohjista. Lisäksi työkalussa voidaan käyttää Windows DOS-komentojonoa Visual Basic (VB) tai linkittää työkalu TIA Portalin ”external applications”-kirjastoon. [28, s. 5.]

”Openness Scripter” -ohjelmiston voi ladata Siemensin omilta verkkosivuilta ja on täysin ilmainen. Työkalu tarvitsee toimiakseen samat edellytykset kuin TIA Openness-rajapinta kuten Microsoft .NET Framework täytyy olla asennettuna samassa tietokoneessa tai virtuaalikoneessa. Lisäksi Microsoft Windowsin käyttäjätili tulisi lisätä ”Siemens Openness User”-ryhmään. Scriptien suorittamiseen tarvitaan TIA Portal STEP 7 Professional tai WinCC Comfort/Advanced/Unified/Professional maksullinen lisenssi. Myös Trial -lisenssi sopii kokeilukäyttöön. [27, s. 1–7; 28, s. 5–6].

5 Ohjelmistokehyksen työkalut

5.1 Automaattinen koodin generointi

Automaattinen koodin generointi on ohjelmistosuunnittelun prosessi, joka tuottaa konekielistä koodia tietystä abstraktista kuvauksesta. Sen tavoitteena on minimoida koodin kirjoittamiseen tarvittava aika ja vähentää manuaalisen koodauksen virheiden riskiä.

Ohjelmistokehittäjät voivat vähentää manuaalisen vaivan tarvetta kirjoittaa lähdekoodia käyttäen erikoistyökaluja, jotka generoivat koodia automaattisesti mallin tai syöttötiedon perusteella. Toisin sanoen ohjelmistosuunnittelijat luovat

mallin, joka kuvaa ohjelmiston toimintaa korkealla tasolla, mikä lopuksi voidaan sitten muuntaa automaattisesti ohjelmakoodiksi ja kääntää tietokoneella. [41.]

Automaattinen koodinluonti on tullut yhä suositummaksi viime vuosina ohjelmistojärjestelmien kasvavan monimutkaisuuden vuoksi. Mitä monimutkaisemmiksi järjestelmät tulevat, sitä vaikeampaa ja aikaa vievää on kehittäjille manuaalisesti kirjoittaa ohjelmakoodia, joka on sekä tehokasta että virheetöntä. Automaattinen koodinluonti tarjoaa ratkaisun tähän ongelmaan automatisoimalla koodin luomisprosessin. [41.]

Automaattinen koodinluonti vähentää merkittävästi ohjelmiston kehitykseen tarvittavaa aikaa. Automaattista koodinluontiprosessia käyttämällä kehittäjät voivat keskittyä muihin ohjelmistokehityksen osa-alueisiin, kuten testaukseen ja vianetsintään. Tämä voi johtaa nopeampiin kehitysaikoihin ja nopeampaan markkinoille tuontiin ohjelmistotuotteille. [41.]

Automatisoidun koodin generoinnin yksi keskeisistä eduista on tuottavuuden tehostaminen, jossa kehittäjät säästävät merkittävän määrän aikaa automatisoimalla toistuvia tehtäviä, kuten lähdekoodin kirjoittamista tai ”gettereiden” ja ”settereiden” generointia. Tämä aika voitaisiin käyttää tehokkaammin korkeamman tason tehtäviin, kuten ohjelmiston suunnitteluun ja testaamiseen. Tämän takia automaattinen koodin generointi voisi auttaa kehittäjiä saattamaan projektit päätökseen nopeammin ja parantamaan kokonaistuottavuutta. [29.]

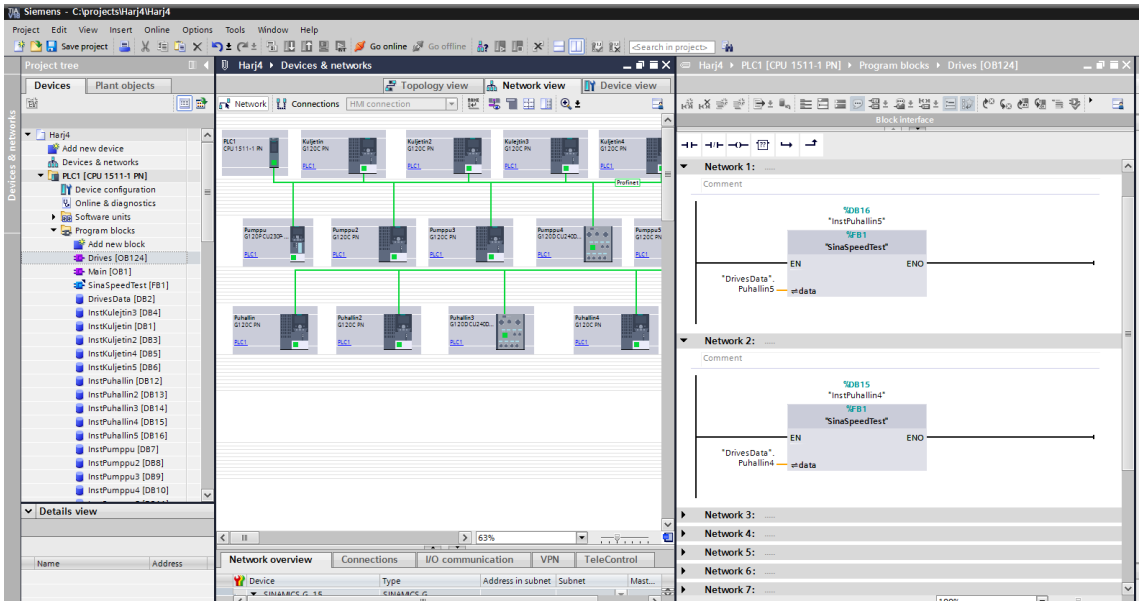
Toiseksi automaattinen koodin generointi voi auttaa vähentämään virheiden riskiä, jotka voivat syntyä manuaalisen koodin kirjoittamisen seurauksena. Tämä johtuu siitä, että generoitu koodi perustuu malliin tai syötedataan, mikä vähentää inhimillisten virheiden mahdollisuutta. [29.]

Kolmas etu automaattisessa koodin generoinnissa on parantunut koodin yhdenmukaisuus. Kun koodi generoidaan automaattisesti, se noudattaa vakiintuneita koodausstandardeja ja käytäntöjä, mikä helpottaa muiden kehittäjien koodin ymmärtämistä ja ylläpitoa. Tämä voi olla erityisen hyödyllistä suurissa

projekteissa, joissa on paljon kehittäjiä, koska se varmistaa, että kaikki ovat samalla sivulla koodauskäytäntöjen suhteen. [29.]

Lisäksi automaattinen koodin generointi voi auttaa vähentämään koodin kopiointiriskiä. Kun koodi generoidaan automaattisesti, on vähemmän todennäköistä, että kehittäjät kirjoittavat samankaltaisia koodilohkoja itsenäisesti, mikä voi johtaa epäjohdonmukaisuuksiin ja bugeihin. [29.]

Koska generoitu koodi noudattaa vakiintuneita koodausstandardeja, sitä on yleensä helpompi ylläpitää ja päivittää ajan mittaan. Kun päivityksiä tarvitaan, ne voidaan usein tehdä suoraan syötedataan tai malliin, ja automaattista koodin generointiprosessia voidaan ajaa uudelleen päivitetyn koodin luomiseksi, mikä sisältää muutokset. Tämä voi merkittävästi vähentää vaivaa, joka vaaditaan monimutkaisten ohjelmistoprojektien ylläpitämiseen ja päivittämiseen sekä auttaa vähentämään uusien virheiden riskiä päivitysten yhteydessä. Koska koodi generoidaan automaattisesti, siinä on vähemmän tilaa inhimillisille virheille varmistaen, että päivitykset toteutetaan oikein. [29.]



Kuva 15. TIA Portal -projekti, jossa on tapahtunut automaattisesti generoitunut sovelluskoodi sekä lisäksi automaattisesti lisättyjä Sinamics-taajuusmuuttujia

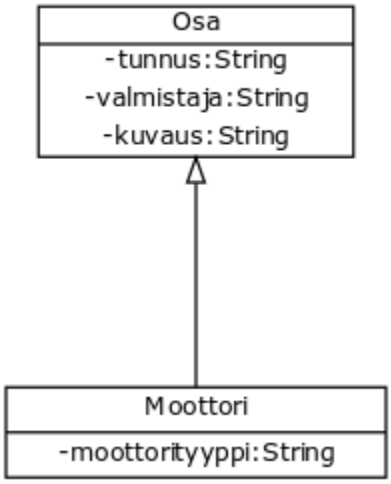
TIA Openness-rajapinnan avulla ja jokaiseen laitteeseen on asetettu yksilölliset IP-osoitteet.

Tarkastellaan tarkemmin esimerkkinä automaattista koodin generointia TIA Portal -ympäristössä. Kuvassa on lisätty projektiin viisitoista taajuusmuuttujaa, mikä olisi manuaalisesti käsin vielä mahdollista, mutta jos laitteita joutuisi lisäämään satoja, olisi konfigurointi hyvin työlästä ja aikaa vievää niin siksi automaattinen koodin generointi on tarpeen. Ratkaisuna haasteeseen olisi luoda ohjelma TIA Openness -rajapinnan avulla, joka generoisi projektiin automaattisesti satoja tai tuhansia laitteita ja konfiguroisi laitteet yhteensopiviksi kuten kuvassa esitettynä. Tämä korvaisi valtaosan manuaalisesta työstä automatisoinnilla.

Kokonaisuudessaan automaattinen koodin generointi on merkittävä työkalu ohjelmistokehityksessä. Sen ymmärtäminen ja soveltaminen parantaa ohjelmistojen laatua, nopeuttaa kehitysprosesseja ja vähentää virheitä.

5.2 Olio-ohjelmointi

Object-Oriented Programming (OOP) on tietojenkäsittelytieteen ohjelmointiparadigma, joka perustuu luokkien ja objektien käsitteeseen, jota käytetään ohjelmiston jäsentämiseen yksinkertaisiksi ja uudelleen käytettäviksi koodipiirroksiksi (kutsutaan yleensä luokiksi) ja joita käytetään yksittäisten objektien esiintymien luomiseen. Nykyään on olemassa monia oliopohjaisia ohjelmointikieliä, mukaan lukien JavaScript, C++, Java ja Python. [31.]



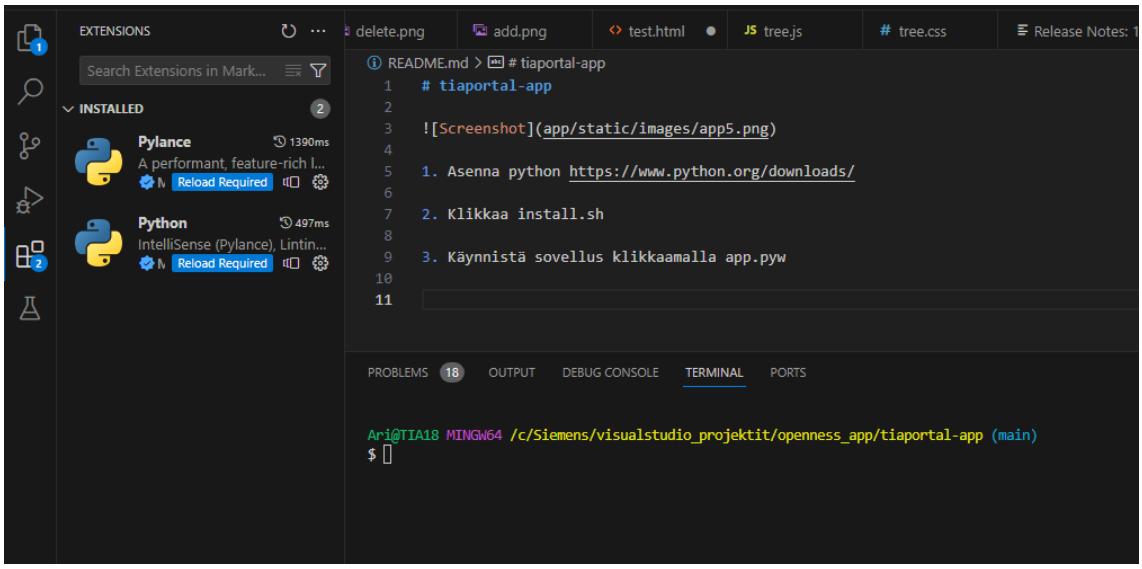
Kuva 16. Luokat Osa ja Moottori kuvaaminen [30.]

OOP-kielet eivät välttämättä rajoitu olio-ohjelmointiparadigmaan. JavaScript, Python ja PHP-ohjelmointikielet sallivat kaikki sekä proseduurit että oliopohjaiset ohjelmointityylit. [31.]

Luokka on abstrakti suunnitelma, joka luo tarkempia, konkreettisia objekteja. Luokat voivat sisältää myös menetelmiksi kutsuttuja toimintoja, jotka ovat käytettävissä vain tämän tyyppisille objekteille. Nämä toiminnot on määritelty luokassa ja suorittavat toimintoja, jotka ovat hyödyllisiä kyseiselle objektityypille. [31.]

5.3 Microsoft Visual Studio

Visual Studio Code on ilmainen ja tehokas lähdekoodieditori, joka toimii paikallisesti asennettuna tietokoneessa taikka verkkoversiona ja on saatavilla Windowsille, macOS:lle, Linuxille ja Raspberry Pi OS:lle. Siinä on sisäänrakennettu tuki JavaScriptille, TypeScriptille ja Node.js:lle. Visual Studio Code sisältää runsaasti ekosysteemilaajennuksia muille ohjelmointikielille kuten C++, C#, Java, Python, PHP ja Go sekä suoritusalustoille kuten .NET ja Unity. Ympäristöt kuten Docker, Kubernetes, Amazon Web Services, Microsoft Azure ja Google Cloud Platform ovat myös tuettuna. [32.]



Kuva 17. Visual Studio Code sisältää monipuoliset kehitys- ja testaustyökalut.

Visual Studio Codessa on IntelliSense-koodin täydennysmuuttujia, menetelmiä ja tuotuja moduuleja varten graafinen virheenkorjaus esim. linting, monen kursorin muokkaus, parametrivinkit ja muut tehokkaat muokkausominaisuudet, toimiva koodinavigointi, refaktorointi, sisäänrakennettu lähdekoodin ohjaus ja Git-tuki. Suurinta osaa näistä voidaan mukauttaa Visual Studio -tekniikalla. [32.]

Visual Studio Code todellisuudessa on rakennettu käyttäen Electron-kuorta, Node.js:ää, TypeScriptiä ja Language Server Protocolia. Alustaa päivitetään kuukausittain. Monet laajennukset päivittyvät automaattisesti niin tarvittaessa. Tuen runsaus vaihtelee eri ohjelmointikielissä ja niiden laajennuksissa aina yksinkertaisesta syntaksin korostamisesta ja hakasulkeiden sovituksesta virheen korjaukseen ja uudelleenkäsittelyyn. [32.]

5.4 C#-ohjelmointikieli ja .NET Framework

Vuonna 2000 Microsoft loi C#-kielen, joka tunnetaan nimellä C Sharp. C# on olio-ohjelmointikieli, jota käytetään .NET Frameworkissa. C# on suunniteltu yksinkertaiseksi, tehokkaaksi, monipuoliseksi ja sitä käytetään usein monenlaisten työpöytä-, verkko- ja mobiilisovellusten rakentamiseen. [33.]

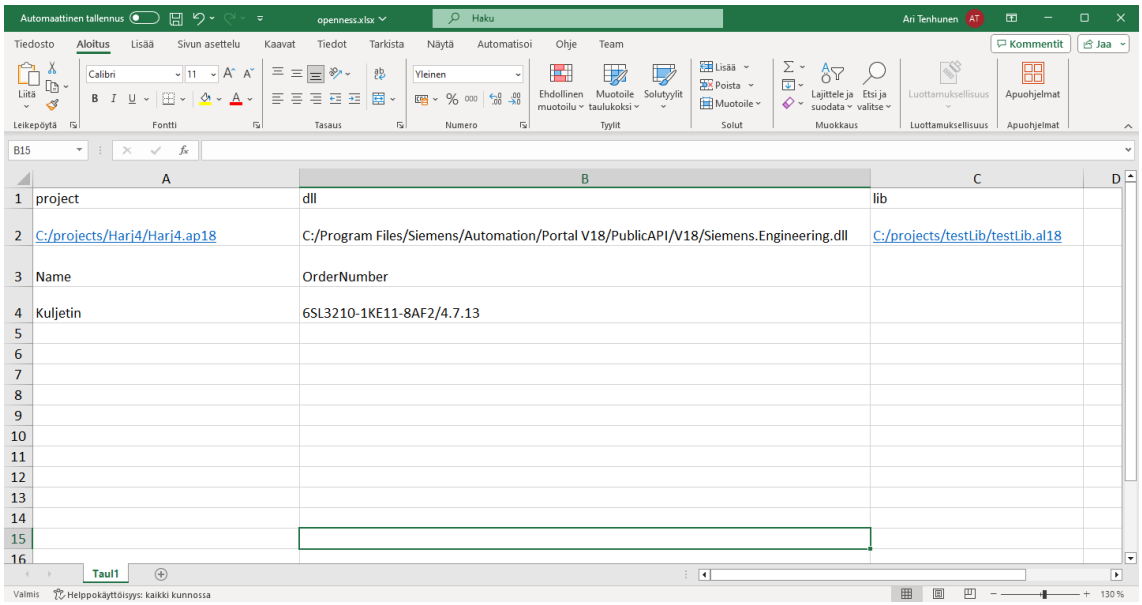
.NET Framework on Microsoftin suunnittelema ja kehittämä ohjelmistokehys, joka ilmestyi vuonna 2002 ja on yksinkertaisesti sanottuna virtuaalikone eri kielillä kirjoitettujen ohjelmien kääntämiseen ja suorittamiseen. Kehystä käyttää lomakepohjaiset (WinForms) sovellukset, Web-pohjaiset sovellukset ja Web-palvelut. [33.]

.Net-alustalle on saatavilla useita ohjelmointikieliä, joista VB.Net ja C# ovat yleisimpiä ja tarjoavat samalla paljon toimintoja ja sisältävät laajan tuen alan standardeille. [33.]

5.5 Microsoft Excel

Microsoft Excel on Microsoftin tuottama kaupallinen taulukkolaskentaohjelmisto, jonka avulla käyttäjät voivat järjestää, muotoilla ja laskea tietoja kaavoilla. [34.]

Ohjelmisto on osa Microsoft Office -pakettia ja on yhteensopiva muiden Office-ohjelmiston sovellusten kanssa. Kuten muutkin Microsoft Office -tuotteet, myös Microsoft Excelin voi nykyään ostaa pilvipalveluna tilauspohjaisesti Office 365:n kautta. Ohjelmisto on saatavilla Windows- ja Mac OS -käyttöjärjestelmille. [34.]



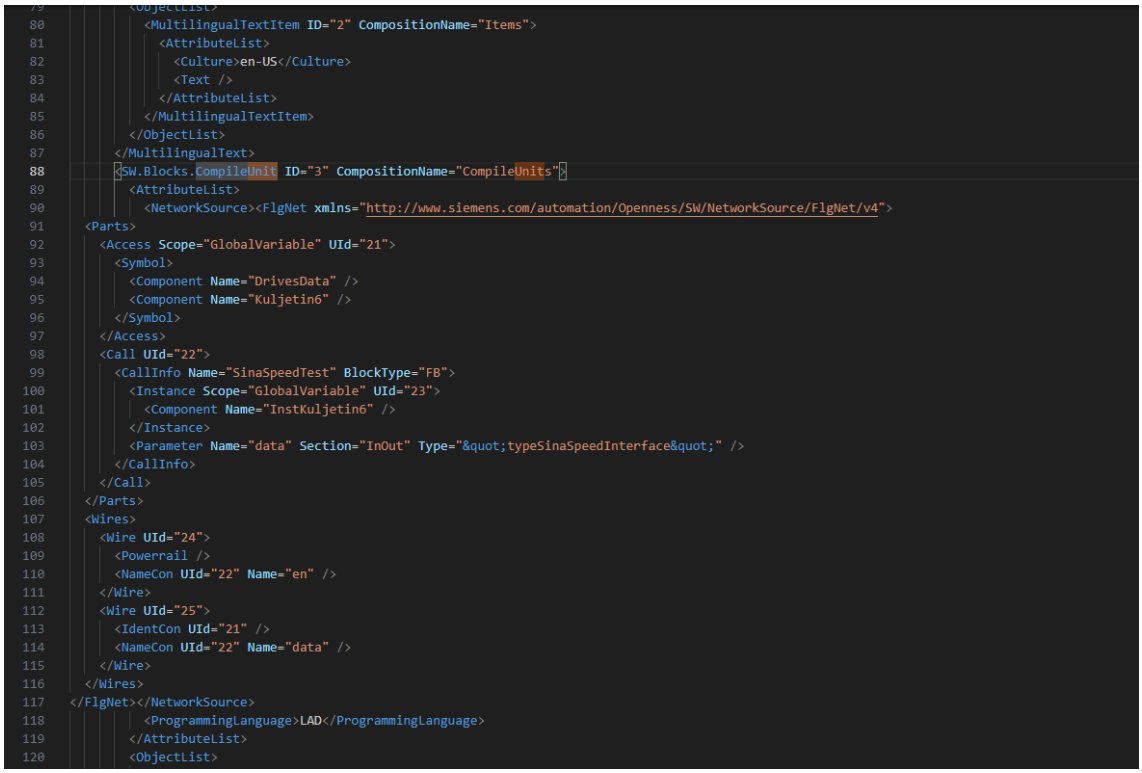
Kuva 18. Kuvassa Excel-työkalu ja sovelluksen esimerkkipohja

MS Excel käyttää kokoelmaa riveihin ja sarakkeisiin järjestettyjä soluja tietojen järjestämiseen ja käsittelemiseen. Informaatiota voidaan myös näyttää kaavioina, histogrammeina ja viivakaavioina. [34.]

MS Excelin avulla käyttäjät voivat järjestää tietoja tarkastellakseen eri tekijöitä eri näkökulmista. Microsoft Visual Basic for Applications on ohjelmointikieli, jota käytetään Excelin sovelluksissa ja jonka avulla käyttäjät voivat luoda erilaisia monimutkaisia numeerisia menetelmiä. Ohjelmoijat voivat kirjoittaa syntaksia suoraan Excelin Visual Basic Editorilla. Visual Basic (VB) on Microsoftin vuonna 1991 luoma oliokieli- ja kehitysympäristö. [35.]

5.6 XML-merkintäkieli

XML on merkintäkieli, jonka lyhenne tulee sanoista Extensible Markup Language. XML on joukko koodeja tai tunnisteita, jotka kuvaavat digitaalisen asiakirjan tekstiä. Tunnetuin yleisesti käytetty merkintäkieli on hypertekstin merkintäkieli (HTML), jota käytetään web-sivujen muotoiluun. XML, joka on HTML:n joustavampi serkku, mahdollistaa monimutkaisten liiketoimintojen toteuttamisen internetin kautta. [36.]



Kuva 19. Openness-rajapinnan generoima XML-tiedosto.

XML tuo verkkosivustoille kehittyneen datan koodauksen ja se auttaa yrityksiä integroimaan tietovirtojaan. Luomalla yksi XML-tunnistesetti kaikille yritystiedoille tiedot voidaan jakaa saumattomasti verkkosivustojen, tietokantojen ja muiden taustajärjestelmien välillä. XML:n voima piilee kuitenkin erityisesti yritysten välisissä transaktioissa. Kun yritys myy tavaraa tai palvelua toiselle yritykselle, on vaihdettava paljon tietoa muun muassa hinnoista, ehdoista, teknisistä tiedoista sekä toimitusaikatauluista. [36.]

HTML:n luonne tekee tällaisesta vaihdosta vaikeaa, ellei mahdotonta, Internetin kautta. XML:n avulla kaikki tarvittavat tiedot voidaan jakaa sähköisesti, mikä mahdollistaa monimutkaisten sopimusten sulkemisen ilman ihmisen väliintuloa. Siksi yritysten väliset verkkomarkkinat, kuten Ariban ja Commerce Onen ylläpitämät, luottavat XML:ään löytääkseen automaattisesti ostajat ja myyjät. Lähitulevaisuudessa yritykset luultavasti arvioidaan niiden XML-tunnisteiden sisällön perusteella. [36.]

5.7 Python-ohjelmointikieli

Python on vuonna 1991 Guido van Rossumin kehittämä ohjelmointikieli, jota käytetään usein verkkosivustojen ja ohjelmistojen rakentamiseen, tehtävien automatisointiin ja tietojen käsittelyyn. [43.] Python on yleiskäyttöinen kieli ja sitä käytetään useiden ohjelmien luomiseen sen selkeän syntaksisuuden takia. Tämä aloittelijoille ystävällisyys on tehnyt siitä yhden nykypäivän eniten käytetyistä ohjelmointikielistä. [38.]

Kuva 20. Python-koodi sovelluksesta, joka käynnistää Flask-palvelimen

Flask on Armin Ronacherin kehittämä ohjelmistokehys, jonka avulla kehittäjät voivat toteuttaa kevyitä selainsovelluksia Python-toteutusympäristössä. [44.]

Pythonnet on paketti, joka mahdollistaa Python-ohjelmoijille yhteisen integraation .NET-kielirungon (CLR) kanssa. [45.]

ElementTree on Python-paketti, joka mahdollistaa XML-dokumenttien luomisen, muokkaamisen ja jäsentämisen. [46.]

Pythonin käyttöä suositaan verkkosivustojen ja ohjelmistojen kehittämiseen, tehtävien automatisointiin, tietojen analysointiin ja tietojen visualisointiin. Koska Python on suhteellisen helppo oppia, monet ei-ohjelmoijat, kuten kirjanpitäjät ja tiedemiehet, ovat omaksuneet Pythonin erilaisiin päivittäisiin tehtäviin, kuten talouden järjestämiseen. [38.]

"Ohjelmien kirjoittaminen on erittäin luovaa ja palkitsevaa toimintaa", sanoo Michiganin yliopisto ja Courseran ohjaaja Charles R Severance kirjassaan Python for Everybody. "Voit kirjoittaa ohjelmia monista syistä, aina elantonsa hankkimisesta vaikean data-analyysiongelman ratkaisemiseen, hauskanpitoon ja jonkun muun auttamiseen ongelman ratkaisemiseen." [38.]

Stack Overflown 2022 Developer Survey paljasti, että Python on neljänneksi suosituin ohjelmointikieli, ja vastaajat sanoivat käyttävänsä Pytonia lähes 50 prosenttia ajasta kehitystyössään. Tutkimustulokset osoittivat myös, että Python on sidoksissa Rustiin halutuimpana teknologiana, ja 18 % sitä käyttämättömistä kehittäjistä ilmoitti jo olevansa kiinnostunut Pythonin oppimisesta. [38.]

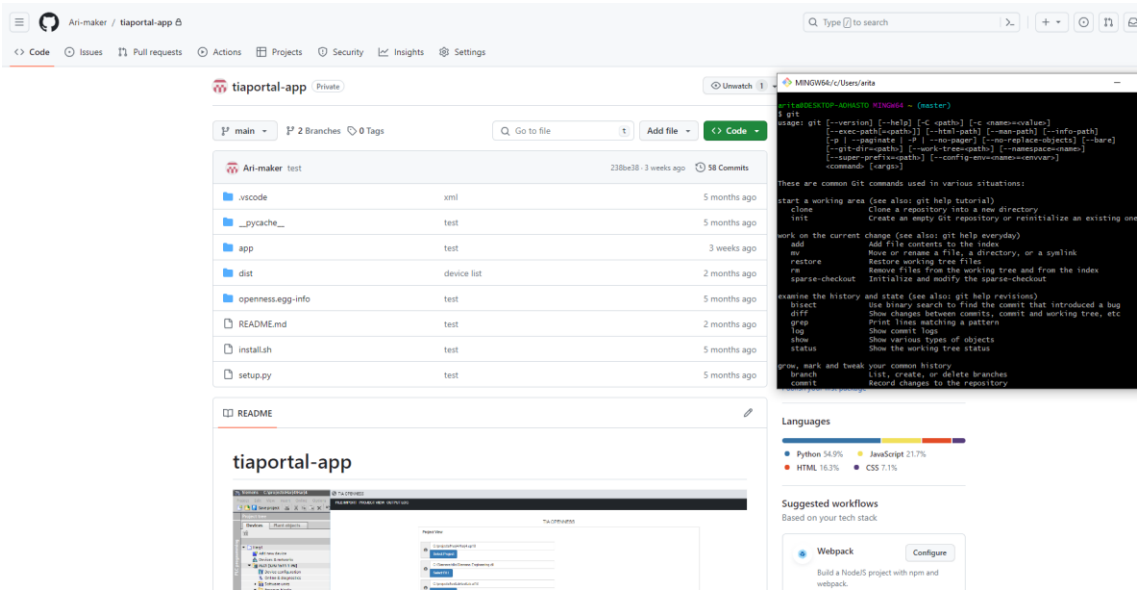
Pythonista on tullut tietotieteen perusosa, jonka avulla data-analyytikot ja muut ammattilaiset voivat käyttää kieltä monimutkaisten tilastolaskelmien tekemiseen, tietojen visualisointien luomiseen, koneoppimisalgoritmien rakentamiseen, tietojen käsittelyyn ja analysointiin sekä muiden dataan liittyvien tehtävien suorittamiseen. [42.]

Python voi rakentaa laajan valikoiman erilaisia datavisualisointeja, kuten viiva- ja pylväskaavioita, ympyräkaavioita, histogrammeja ja 3D-kaavioita. Pythonilla on myös useita kirjastoja, joiden avulla koodaajat voivat kirjoittaa ohjelmia tiedon analysointiin ja koneoppimiseen nopeammin ja tehokkaammin. [42.]

5.8 Git

Git on avoimen lähdekoodin hajautettu versionhallintajärjestelmä. Se on suunniteltu käsittelemään pieniä ja suuria projekteja nopeasti ja tehokkaasti. Se on kehitetty koordinoimaan työtä kehittäjien kesken. Versionhallinnan avulla voidaan seurata ja työskennellä tiimien jäsenten kanssa samassa työtilassa. [39.]

Git on perusta monille palveluille, kuten GitHub ja GitLab, mutta voi käyttää Gitiä ilman muita Git-palveluita. Gitiä voidaan käyttää yksityisesti ja julkisesti.



Kuva 21. Github-sivusto ja Git Bash -komentokehote

Linus Torvalds loi Gitin vuonna 2005 kehittääkseen Linux-ytimen. Sitä käytetään myös tärkeänä hajautetun versionhallintatyökaluna DevOpsille. [39.]

Git on helppo oppia, ja sen suorituskyky on nopea. Se on parempi kuin muut SCM-työkalut, kuten Subversion, CVS, Perforce ja ClearCase. [39.]

6 Työn toteutus

Tässä osiossa käydään läpi tarkemmin opinnäytetyön sovellusta, jonka yhteydessä tarkastellaan ohjelmiston kehitysprosessia tarkemmin. Sovelluksen tarkoituksena on tuottaa ohjelmakoodia ja käyttää korkeamman tason ohjelmointikieltä generoiden ohjelmalohkoja Siemens PLC -projektiin perustuen XML-merkintäkieleen. Sovelluksen käyttö ei ole rajoitettu ainoastaan tämän yrityksen tarpeisiin, vaan se on tarkoitus julkaista avoimeksi GitHubissa, jotta jokainen automaatiosuunnittelija voisi sitä hyödyntää.

Tämän opinnäytetyön päätavoitteena ei ole valmiin tuotteen tuottaminen, joka pystyy automaattisesti luomaan kaiken ohjelmalohkoista ja kirjastoista. Sen sijaan pyrkimys oli tutkia menetelmiä, joilla tämä olisi mahdollista toteuttaa organisaatiossa.

Edellisissä luvuissa oli kattavasti käsitelty erilaisia ohjelmointikieliä ja työkaluja. Niissä ei kuitenkaan käsitellä kaikkia mahdollisia projektin toteutustapoja, joita tutkitaan seuraavasti.

6.1 Toteutusympäristön valinta

6.1.1 Excel ja makrot

Ensimmäinen lähestymistapa on sisällyttää sovellus osaksi lähdedokumenttia ja toimia ikään kuin taustaprosessina. Yrityksellä on käytössään Excel-tiedosto, joka toimii perustana kaikissa projekteissa. Tähän lähdedokumenttiin kerätään kaikki tarvittavat tiedot PLC-ohjelmistoprojektin toteuttamiseksi, mukaan lukien kuljetinlista ja kuljettimien tyypit. Tämän ratkaisun ajatuksena on kehittää sovellus osaksi tätä lähdedokumenttia makrona, joka kulkeutuu käyttäjän mukana aina, kun hän lataa dokumentin verkkolevyltä omalle tietokoneelleen. Tämä mahdollistaa sovelluksen käytön ja ylläpidon kaikille dokumentin käyttäjille. Tapa nähdään järkevänä myös integroinnin kannalta, sillä se mahdollistaa vanhojen makrojen siirtämisen uuteen käyttöliittymään ja selkeyttää dokumenttia

käyttäjän näkökulmasta, mikä tuo kaiken toiminnallisuuden käyttöliittymään ja jättää datan Excel-taulukkoon. Käytännössä Excel-taulukot toimisivat tietokantana ja sovellus tarjoaisi lisätoimintoja, mutta kaikki olisi yhdessä dokumentissa saatavilla koko projektiryhmälle.

Tämän toteutusmallin heikkoutena on sovelluksen vaatimat välikappaleet, jotka ovat tarpeen TIA Portal -ympäristön kanssa kommunikoimiseksi. Excelin Visual Basic -ohjelmakoodia ei voida suoraan käyttää Siemens-rajapintafunktioihin, vaan siihen tarvitaan OpennessScripter.exe-sovellus ja komentojonotiedoston käyttö avaamiseksi ja suorittamiseksi. Tämä yhteistoiminta on käsitelty aikaisemmin, ja yhtenä heikkoutena on, että tämä malli ei mahdollista dynaamista yhteyttä sovelluksen ja TIA Portal -ympäristön välillä, toisin kuin seuraava toteutusmalli.

6.1.2 Microsoft Visual Studio ja C#

Toinen lähestymistapa on kehittää TIA Openness -sovellus hyödyntämällä rajapinnan tarjoamia funktioita C#- tai Visual Basic -ohjelmointikielillä. Ratkaisu vaatii käyttöön Microsoft Visual Studion kehitysympäristön, jossa pääsee suoraan käyttämään Siemensin rajapintafunktioita. TIA Openness -rajapinnan integroiminen Visual Studio -projektiin edellyttää rajapintatiedoston lisäämistä projektin riippuvuuksiin, mutta ohjelmoijan on perehdyttävä tarkemmin rajapinnan luokka- ja oliorakenteisiin. Tämä ratkaisumalli ei tarvitse ulkopuolisia sovelluksia toimiakseen, vaan se mahdollistaa yhteyden säilyttämisen dynaamisesti TIA Portal -ympäristöön koko ohjelman suorituksen ajan. Näin tarjotaan monipuolinen perusta erilaisille toiminnoille. Käyttäjän kannalta sovellus toimii yhtenä sovelluksena, asennettuna tietokoneelle, ja sitä voi hyödyntää kaikissa projekteissa. Ohjelma on myös erillinen projektitiedostoista, mikä antaa käyttäjälle mahdollisuuden valita, mitä dokumentteja käytetään kunkin projektin osalta.

Tämän sovellusmallin toteutus edellyttää ohjelmoijalta syvempää osaamista Microsoftin ohjelmointikielissä ja olio-ohjelmoinnissa. Lisäksi tekijän on oltava

tarkasti perillä rajapintafunktioiden käytöstä, mikä tuo omat haasteensa, vaikka dokumentaatiota olisikin saatavilla.

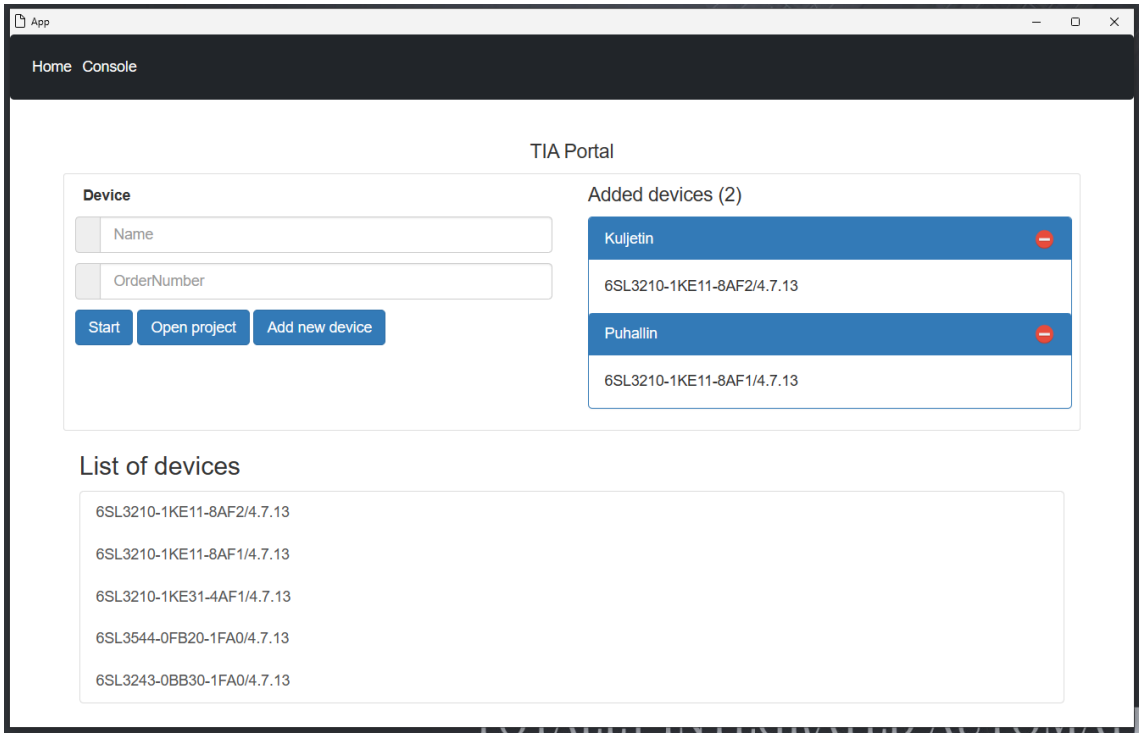
6.1.3 Microsoftin .NET Framework

Kolmas lähestymistapa on samantapainen kuin aikaisempi, mutta C#:n sijaan käytettäisiin toista ohjelmointikieltä, jotka tukevat .NET Frameworkia, kuten C#, C++, Visual Basic, JavaScript, Java, Python ja monia muita. Tähän vaaditaan edelleen ohjelmoinnin osaamista, mutta hyvänä puolena on se, että kehittäjät voivat valita itselleen sopivimman ohjelmointikielen, mikä antaa joustavuutta myös asiakkaille, koska tällöin asiakkaan omia kirjastoja tai lisäosia ei tarvitse muokata toteutuksesta riippumatta yhteensopivaksi toisen yrityksen tarpeisiin. Lisäksi tämä lähestymistapa tuo mahdollisuuden käyttää TIA Portal Openness -rajapintaa esimerkiksi web-sovelluksissa, jolloin sovelluskehitys voidaan toteuttaa Pythonilla. Jos on tarvetta integroida sovellus verkkoselaimeen, niin sen voi tehdä ilman käyttöliittymän muuttamista nopeuttaen kehittäjien työtä.

6.1.4 Toteutusympäristön valinta ja perustelut

Pythonin valitseminen Microsoftin .NET Frameworkin käyttämiseen osoittautui ratkaisuksi, vaikka se on hieman hitaampi kuin C#. Python on maailmanlaajuisesti suosittu ohjelmointikieli, ja sen käyttö työelämässä on kasvanut nopeasti sen helppolukuisen syntaksin vuoksi. Python tukee useita kirjastoja, joita opinnäytetyössä käytetään ovat esimerkiksi Flask, Pythonnet ja ElementTree. Pythonnetin avulla voidaan käyttää TIA Portal Openness -rajapintaa Pythonissa ja ElementTree-kirjastolla on mahdollista kirjoittaa XML-muodossa olevaa tietotyyppiä TIA Portal Openness -rajapinnan käyttämiseen sovelluksessa. Flask tarjoaa selainkäyttöliittymän integroinnin Pythoniin, mikä mahdollistaa saman sovelluksen käytön sekä työpöytä- että websovelluksina ilman tarvetta muokata ohjelmakoodia erikseen. Tämä nopeuttaa kehittämistä ja lisää joustavuutta ja siksi tämä koettiin hyväksi vaihtoehdoksi. Siemensin aikomuksena on kehittää tulevaisuudessa virallinen Python-tuki Openness-rajapintaan, joka poistaa tarpeen Pythonnet-kirjastolle.

Demosovellus syntyi ensimmäisen asiakaspalaveri käynnin aikana syksyllä, jossa lähinnä pystyi käyttöliittymässä avaamaan projektitiedoston ja valitsemaan listasta taajuusmuuttujia, jotka generoitiin lopuksi Tia Portal -projektiin. Demo oli tutkimuksen kannalta erittäin hyödyllinen.



Kuva 22. Ensimmäinen demo sovelluksesta

Tutkimuksen valinnan jälkeen jäi päätettäväksi toteuttaa demosovellus joko selaimessa tai työpöytäsovelluksena. Valinnassa lopulta päädyttiin työpöytäversioon, koska ohjelmaan täytyy päästä käsiksi ilman internetyhteyttä. Käyttäjä joutuisi asentamaan Pythonin koneelle ja siihen tarvittavat kirjastot. Edellä kuvatun vaiheen helpottamiseksi käyttäjille määritellään asennusohjeissa sovelluksen yhtenäinen asennustiedosto, jossa asennetaan automaattisesti kaikki sovelluksen vaaditut Python-kirjastot.

6.2 Aikataulu ja projektin kulku

Sovelluksen suunnitelmaa toteutettiin heinäkuussa 2023, jolloin asiakkaan kanssa sovittiin sovelluksen minimivaatimukset. Opinnäytetyötekijä työskenteli yksin sovelluksen parissa, ja parin viikon välein pidettiin palavereja asiakkaan kanssa. Ensimmäisen kuukauden aikana projekti eteni niin, että pystyttiin tuottamaan yksinkertaisia ohjelmalohkoja TIA Portal -projektiin ja lisäämään sekä verkottamaan taajuusmuuttujia. Tuolloin kiinnitettiin enemmän huomiota sovelluksen toiminnallisuuksiin kuin itse sovelluksen käyttöliittymään. Sovelluksen perustoiminnot olisi tarkoitus saattaa päätökseen vuoden loppuun mennessä ja tuoda samalla esiin jatkokehitysideoita.

6.3 Käyttöliittymän suunnittelu

Asiakkaan kanssa päädyimme siihen, että sovellukseen olisi järkevintä toteuttaa oma käyttöliittymä, joka olisi graafinen, selkeä, helppokäyttöinen ja käyttötarkoitukseensa sopiva. Sovellus tultaisiin toteuttamaan JavaScript- ja Bootstrap-kirjastolla, koska JavaScript on käytetyin selainpuoleinen kieli ja Bootstrap-kirjaston avulla saadaan valmiita käyttöliittymä komponentteja, jotka ovat selkeitä ja kuvaavia. Tämä nopeuttaisi käyttöliittymän toteuttamista ja jäisi enemmän aikaa toimeksiantajan tavoitteiden toteuttamiseen sovelluksessa.

Käyttöliittymässä tulisi olemaan Excel-näkymä, projektin näkymä ja konsolin näkymä. Excel-pohjaisen näkymän oli tarkoitus avautua ensimmäisenä, kun sovelluksen käynnistää, koska se selkeyttäisi käyttökokemusta. Käyttäjälle esitetään yksikertaisuudessaan vain nappi, josta valittuaan avautuisi tiedostokansion näkymä, josta valittaisiin toivottu Excel-tiedosto (kuva 18). Kyseinen Excel-tiedosto sisältää polut projektiin ja kirjastoon sekä taajuusmuuttujista nimen ja tunnuksen. Datan tuomisen jälkeen ohjelma näyttäisi lisätietoja tulleesta tiedostosta.

Projektinäkymä sopii käyttäjälle, jolla ei ole käytössä Excel-tiedostoa ja jotka haluavat vaikuttaa enemmän asioihin liittyen projektin ohjelmalohkojen

generointiin. Näkymässä käyttäjä asettaa polun projektiin sekä valitsee DLL-kirjaston ja asiakkaan omistamaan kirjastoon.

Laiteosiossa pystyisi kenttään kirjoittamaan laitetunnuksen, jonka jälkeen ohjelmisto tallentaa laitteen listaan. Käyttäjän kokemuksen parantamiseksi on mahdollista valita taajuusmuuttujan yksilöllisen tunnuksen taulukosta, jonka se lisää automaattisesti aikaisempaan kenttään. Lopuksi sovellus sisältää napin ohjelmalohkojen generoinnin aloittamiseksi. Konsolinäkymässä olisi teksti luettelo, jossa käyttäjä saisi tietoa prosessin kulusta ja virhetilanteista. Konsolin viestit olisi mahdollista tyhjentää napilla.

6.4 Sovelluksen rakenne

Sovelluksen rakenne olisi tarkoitus toteuttaa Python-ohjelmointikielellä, joka soveltuu hyvin TIA Openness -rajapinnan käyttämiseen, koska Pythonin syntaksi on helppolukuista ja siihen on tarjolla monia hyödyllisiä kirjastoja, jotka nopeuttavat kehittäjän työtä. Sovellukseen tullaan käyttämään XML-merkintää, koska TIA Openness -rajapinta perustuu XML-tiedoston lukemiseen ja kirjoittamiseen. XML-tiedosto sisältää yhden tai useamman ohjelmalohkon (FC) tai tietueen (DB). Näitä voidaan suorittaa haluttu määrä ohjelman suorittamisen aikana, jotka muutetaan TIA Portal -ympäristössä ohjelmalohkoiksi. Tiedoston käyttämä XML-merkintäkielen rakenne on Siemens AG:n toteuttama ja on hyvin standardoitu (kuva 19).

Taulukossa 1 kerrotaan tarkemmin Siemensin käyttämästä XML-rakenteesta.

Taulukko 1. Opennes-rajapinnan standardoitu XML-rakenne

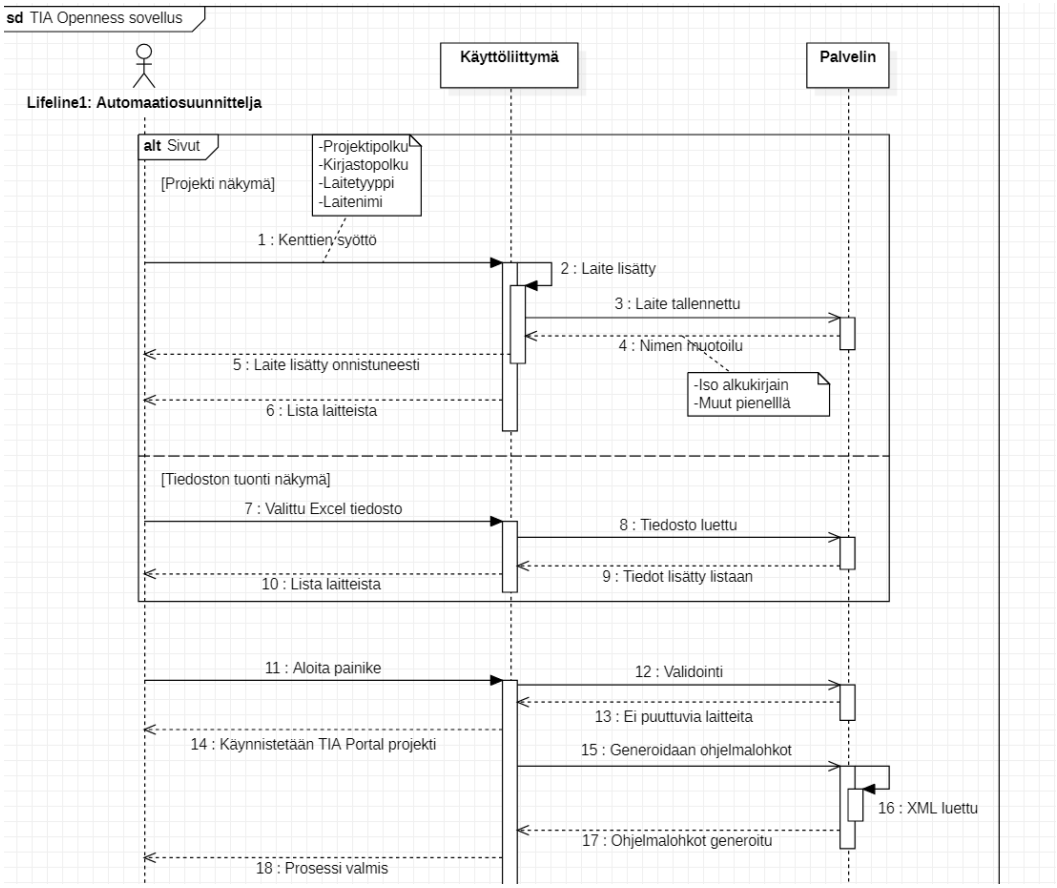
”<SW.Blocks.FC>”	Määritellään toimilohko (FC) tai vastaavasti muita kuten datalohko DB.
”<ObjectList>”	Määritellään kommentti, otsikko ja yksi tai useampi SW.CompileUnit.
”<AttributeList>”	Sisältää NetworkSource ja yhden ohjelmointikielen (ProgrammingLanguage) kuten LAD, FBD, SCL ja STL.
”<SW.Blocks.CompileUnit>”	Sisältää attribuuttilistan ja objektilistan.
”<Parts>”	Sisältää verkotukseen käytettävät muuttujat ja attribuutit.
”<Wires>”	Miten verkotuksen muuttujat ja komennot ovat yhteydessä toisiinsa. Sisältää erilaisia tietoja verkotuksesta ja ne voidaan linkittää toisiinsa.
”<Wire>”	Yksilöllinen verkko attribuutti ”<Wires>” osassa, joita voi olla useita.

Huomattavaa oli se, että jos sovelluksen generoitu tiedoston osien indeksointi olivat samat tai väärässä järjestyksessä niin TIA Openness -rajapinta ei suostunut ajamaan tiedostoa ollenkaan. Ratkaisuna oli tuottaa ohjelman koodissa liukuva luku toistorakenteessa, joka oli samalla oikeassa järjestyksessä.

Sovelluksen ohjelmointi jaettiin kahteen Python-tiedostoon, jossa app.py-tiedostossa käsitellään sovelluksen käyttöliittymää ja yleisiä toimintoja. Openness.py-tiedostossa käsitellään TIA Portal -ympäristöön ohjelmalohkojen generointi, xml toiminnallisuus, kansioiden luonti sekä globaalikirjaston käyttö. Olisi hyvä erottaa käyttöliittymä ja TIA Openness -rajapinta omiin tiedostoihin, koska ohjelmakoodi olisi näin ollen paremmin organisoituna sekä omissa säikeissä, jolloin

rajapinnan ohjelmalohkojen generointi ei vaikuttaisi käyttöliittymän toimintaan huonontaen käyttäjäkokemusta.

Kuvassa 23 on esitetty sekvenssikaavio ohjelman toiminnasta.



Kuva 23. Sekvenssikaavio ohjelman toiminnasta

7 Tulokset, TIA Openness -sovellus

7.1 Valmiin sovelluksen toiminta

Ohjelma toteutettiin Python-alustalle, joka kirjoittaa halutun kaltaista XML-koodia tiedostoon, jonka lopuksi se generoidaan automaattisesti TIA Portal -projektiin ohjelmalohkoiksi TIA Openness -rajapinnan avulla. Ohjelman alkuvaiheessa annetaan käyttäjälle mahdollisuus valita kirjasto, josta haetaan asiakkaan ohjelmalohkot ja projekti, johon ohjelmalohkot generoidaan.

Ohjelma on koosta ja tuottamastaan tietomäärästä riittävän nopea. Ongelmaksi kuitenkin on koettu generointi TIA Portal -ympäristössä sen hieman hitaan suorituskyvyn takia ja kaikkiin rajapinnan toiminnallisuuksiin ei päästy käsiksi ennen kuin projekti oli avautunut. Lisäksi sovellus hakee taajuusmuuttujien tyypit tekstitiedostosta. Asiakas haluaa hakea taajuusmuuttujien tyypit dynaamisesti projektin kautta sovelluksen käyttöliittymään.

Ratkaisuna oli, että käyttäjälle annettiin kaksi erillistä painiketta käyttöliittymään, josta ”Start TIA Portal” -painikkeesta avataan TIA Portal -projekti ja toisesta ”Compile” -painikkeesta aloitetaan ohjelmalohkojen generointi.

Sovelluksen toteutusvaiheessa asiakkaan kanssa päädyimme siihen, että sovelluksen generoima XML-tiedosto olisi mahdollisimman dynaamista ja ohjelman kirjoittamaa. Kopiointia käsin tuoduista XML-pohjista pyrittiin välttämään, jotta jatkossa ei tarvitsisi ylläpitää TIA Portal -projektin ohjelmalohkokirjastoja. Moottoriohjaustoimilohko ”DrivesData” on ainoa ohjelmalohko, jonka kutsu tuotiin käsin sovelluksesta.

Sovelluksen Openness-rajapinnan funktioita haettiin Siemensin dokumentaatiosta mutta sen lisäksi ”Openness Exproler” -työkalun avulla. Lopputuloksena voidaan todeta, että ohjelman osalta on päästy asetettuihin tavoitteisiin.

7.2 Käyttöliittymä

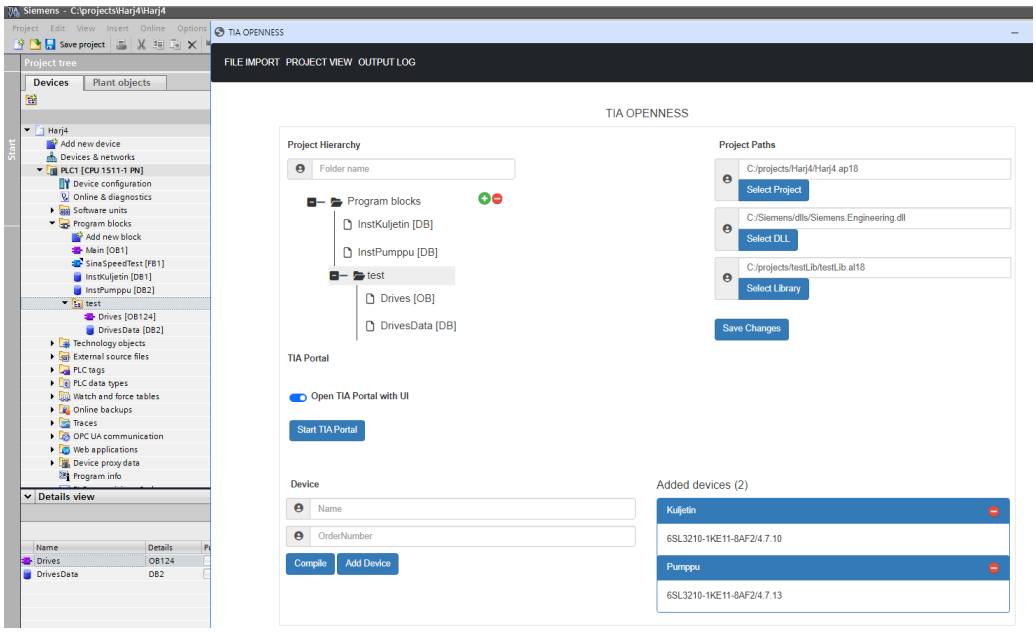
Excel-tiedoston (kuva 18) ja käyttöliittymän osalta pysyttiin pitkälti suunnitelmissa, mutta tiettyjä toiminnallisuuksia lisättiin prosessin aikana niiden osoittautuessa hyödylliseksi. Näitä ovat esimerkiksi ”Start TIA Portal with UI” -vaihtonappi, joka määrittää, näytetäänkö projektin käyttöliittymä taustalla vai ei. Nämä vaihtoehdot eivät vaikuta ohjelmalohkojen generointiin projektissa. Esimerkkikoodissa 1 on toteutustapa vaihtonapista.

Sovelluksen Python-koodi, joka käynnistää TIA Portal -projektin ja avaa käyttöliittymän tai ilman. Metodia “init” kutsutaan, kun käyttäjä painaa ”Start Tia Portal” -painiketta.

```
def init():
     mytia = tia.TiaPortal(tia.TiaPortalMode.WithUserInterface)
     processes = tia.TiaPortal.GetProcesses()
     project_path = FileInfo (_path)
     if ui
          myproject = mytia.Projects.OpenWithUpgrade(project_path)
     else:
          mytia = tia.TiaPortal(tia.TiaPortalMode.WithoutUserInterface)
     return
```

Esimerkkikoodi 1. Python-koodi ohjelmasta

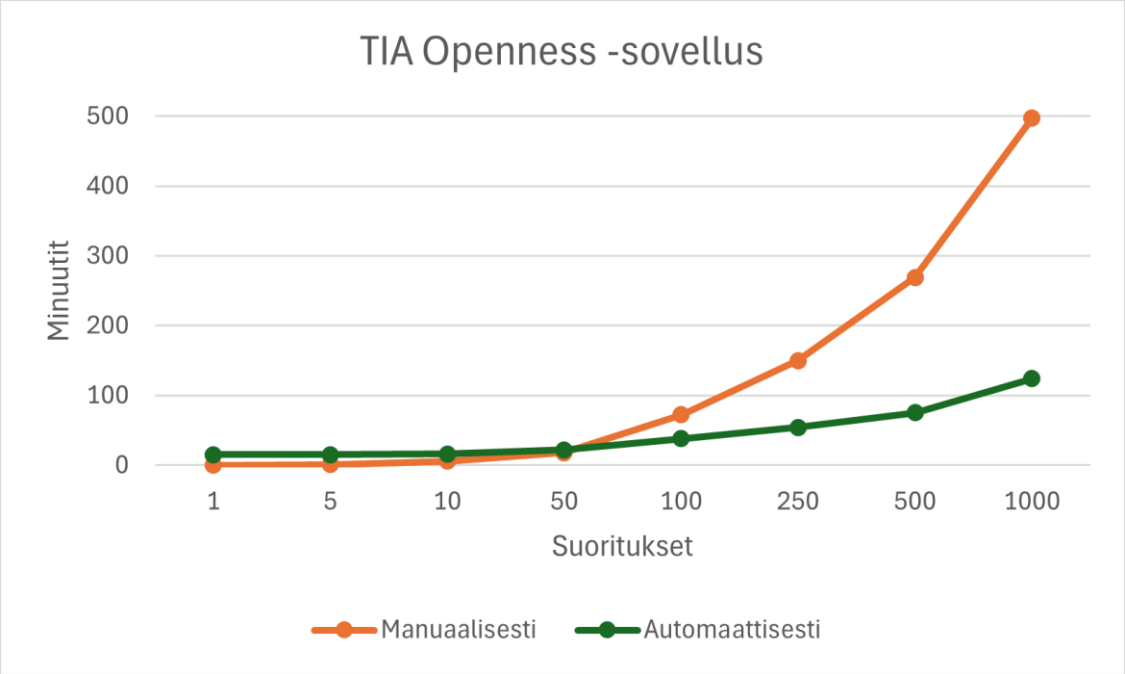
Muita hyödyllisiä käyttöliittymän toiminnallisuuksia olivat kansiopuun rakenne, jolla voidaan organisoida projektipuuta TIA Portal -projektissa. Sovelluksen projektipuuta pystyy raahaa- ja pudota (drag and drop) -toiminnolla siirtelemään kansioita muihin kansioihin sekä tiedostoja, kuten kuvassa näkyy Drives ja tietueet (DB). Painonapit kansion luomiseen ja poistamiseen. Loppukäyttäjille lisättiin painonappi lokikentän tyhjentämiseen. Kuvassa 24 on esimerkki käyttöliittymän toteutuksesta.



Kuva 24. Valmiin sovelluksen käyttöliittymä

7.3 Sovelluksen hyödyt yrityksessä

Havainnoimme sovelluksen generoinnin kestoa verrattuna manuaaliseen työhön TIA Portal -ympäristössä, kuten laitteiden konfiguraatiota, laitteiden verkottamista ja asiakaskirjaston ohjelmalohkoja. Kuvan 25 kaaviosta käy ilmi, että manuaalinen työ ei ole pitkällä aikavälillä kannattavaa.



Kuva 25. Kuva aika eroavaisuuksista manuaalisen ja automaattisen työn välillä.

Asiakkaan sovellus on vielä kehitysvaiheessa, mutta sen avulla voidaan jo vähentää merkittävästi työtunteja TIA Portal -projekteissa. Sovelluksen generoimat ohjelmalohkot ovat aiemmin perustuneet pääosin manuaaliseen työhön ja yrityksen tavoitteena on luopua tästä käytännöstä. Siksi Openness-rajapinta on keskeinen tekijä nykyisillä markkinoilla.

Cimcorp Oy:ssä ei aiemmin ole yritetty luoda vastaavaa ohjelmaa ja TIA Portal Openness -rajapinta vaikuttaa edelleen olevan vähän hyödynnetty ominaisuus, josta on niukasti esimerkkejä verkossa sekä Siemensin rajapinnan dokumentaatiossa. Rajapinta julkaistiin vuonna 2011 ja Siemens on siitä lähtien päivittänyt sitä jatkuvasti, mikä on tuonut parannuksia tähän päivään. Nyt rajapinnan

käyttö alkaa yleistyä, sillä nykypäivän teollisuusjärjestelmät ovat muuttuneet vaativimmiksi, kompleksisemmiksi, ja standardit ovat kasvaneet. Tämän takia uusien automaatioratkaisujen tarve tulee kasvamaan.

Kun sovellusta jatkokehitetään ja otetaan käyttöön laajemmin Cimcorp Oy on todennäköistä, että löydetään uusia innovaatioita ja ratkaisuja vastaaville toteutuksille. Rajapintaa voitaisiin jatkossa hyödyntää entistä laajemmin PLC-ohjelmistokehityksessä.

Tulevaisuudessa työvoimaa voidaan tehostaa niin, että sen resurssit suunnataan kopioinnin sijaan enemmän luovuutta vaativiin ja inhimillisiin tehtäviin. Yksitoikkoiset ja tarpeelliset työt jätetään automaation tehtäväksi.

7.4 Käyttöohjeet

Kun sovellus julkaistaan yleiseen käyttöön, on tarpeen laatia käyttöohjeet. Ensimmäinen vaihe on asentaa Pythonin uusin versio tietokoneelle. Ohjelma ladataan GitHub-sivulta, josta voi ladata zip-tiedoston tai käyttää gitiä komentokehotteesta. Projektin purkamisen jälkeen on käyttäjän asennettava Python-riippuvuudet, jotka on koottu valmiiksi yhteen tiedostoon. Käyttäjän tarvitsee vain suorittaa kyseinen tiedosto, jolloin tarvittavat riippuvuudet on asennettu. Sovellus käynnistetään app.pyw-tiedostosta. Projektin kansiorakenteessa on valmiiksi määritelty asennusohjeet. Käyttäjille tulisi järjestää perehdytys ohjelmiston käyttöön ja tarvittaessa lähdekoodin muokkaamiseen. Sovelluksessa on kattava englanninkielinen kommentointi, jonka tarkoituksena on helpottaa sovelluksen jatkokehittämistä myös muille.

8 Johtopäätökset ja jatkokehitys

Insinöörityön tavoitteena oli tutkia Siemens Osakeyhtiön TIA Openness -rajapinnan mahdollisuuksia PLC-ohjelmistokehityksen automatisoinnissa. Insinöörityössä kehitettiin asiakkaalle Cimcorp Oy -sovellus, joka generoi ohjelmalohkoja Tia Portal -projektiin openness-rajapinnan avulla. Sovellus noudatti XML-merkintäkieltä TIA Openness -rajapinnan ohjelmalohkojen generoinnissa. Tarkoituksena oli, että TIA Portal projektissa kopiointityö pystyttiin korvaamaan yrityksessä automatisoinnilla, joka vähensi automaatiosuunnittelijan työaikaa.

Toimeksiantajan kanssa koettiin järkeväksi ratkaisuksi toteuttaa sovellus Pythonohjelmointi-kielellä. Python oli järkevä valinta sen helpposyntaksisuuden ja yrityksissä kasvaneen suosion vuoksi. Sovellus käytti Flask-, Pythonnet- ja xmltree kirjastoja.

Jatkokehittämisen näkökulmasta olisi hyödyllistä laajentaa sovelluksen HMI-ominaisuuksia TIA Portal Openness -rajapinnan avulla. Tähän sisältyy muun muassa attribuuttien muokkaaminen TIA Portal -ympäristössä ja HMI-käyttöliittymään integroitujen elementtien lisääminen. Vaikka sovelluksen tavoitteena oli alun perin saattaa toiminnallisuudet päätökseen, on huomattu, että käyttöliittymässä olisi mahdollisuus parannuksiin. Tämä voisi sisältää Openness-rajapinnan generoiman XML-tiedoston muokkausmahdollisuuden, fontit, värit, kuvat, animaatiot ja muuta viimeistelyä. Tekoälyn hyödyntämistä XML-tiedostojen muokkaamiseen Openness-rajapinnan generoimiseksi. Sovelluksen käyttöliittymä tukee verkkoselainta ja tästä syystä on mahdollista myöhemmin julkaista sovellus internetiin eri selaimiin. Tämä vaatisi käyttäjältä Internet-yhteyden, mutta käyttäjän tarvitsee asentaa omalle tietokoneelleen ainoastaan TIA Portal -kehitysympäristö.

Lisäksi, kun sovellus on tarkoitus julkaista avoimeksi GitHubiin, on tärkeää, että ohjelmakoodi on asianmukaisesti dokumentoitu. Tämä auttaa muita kehittäjiä ymmärtämään sovelluksen rakennetta ja toimintaa. Laaditut käyttöohjeet ovat

olennainen osa julkaisua, jotta seuraavan olisi helpompi jatkaa sovelluksen kehittämistä.

Työn tavoitteita ja toteuttamista auttoivat Cimcorp Oy:ssä tehdyt haastattelut. Tämä oli oikea tapa kerätä aineistoa havainnoinnin lisäksi, sillä haastateltavat olivat kokeneita automaatiosuunnittelijoita ja sovelluksen loppukäyttäjiä. Vaikka haastateltavien määrä oli pieni, niin sovellukselle asetetut tavoitteet täyttyivät ja sovelluksella saatiin ratkaisu ongelmaan ja tutkimuskysymyksiin.

Tutkimusvaiheen tiedonhankinta suoritettiin keräämällä kirjallista aineistoa erilaisista teknisistä viitekehyksistä, sekä jo olemassa olevista ratkaisuista ja teorioista. Lisäksi tietoa kerättiin keskustelemalla yrityksen viestintäkanavista, joissa oli paljon aiheeseen liittyvää materiaalia.

Henkilökohtaisena mielipiteenä opin paljon uutta automaatiosta ja prosessin aikana Siemens opetti minulle opetustiloissa TIA Portal -ympäristön käyttöä kuten PLC-logiikkojen, ohjelmalohkojen, kirjastojen ja luokkien tekemistä. Siemensin pitämiä teoriakursseista opin ohjelmointikieliä LAD, SCL, C# ja Python.

Työn valmistuttua Cimcorp Oy:n on mahdollista säästää työtunteja jo olemassa olevan sovelluksen toteutuksella. Jatkossa yrityksen PLC-ohjelmointia voitaisiin automatisoida entistä tehokkaammin, mikä mahdollistaisi yrityksen ohjelmistopuolen osaston kyvyn vastata kasvavien projektien asettamiin aikatauluihin ja resursseihin.

Lähteet

1 Cimcorp. 2023. Verkkoaineisto. Wikipedia. <https://fi.wikipedia.org/wiki/Cimcorp>. Luettu 8.9.2023.

2 Case CIMCORP. Verkkoaineisto. Viestintapalvelut. <https://www.viestintapalvelut.fi/case-cimcorp>. Luettu 10.9.2023.

3 Siemens TIA Portal. 2011. Verkkoaineisto. AUTOMA. <https://automa.cz/cz/casopis-clanky/siemens-tia-portal-jednotne-vyvojove-prostredi-pro-automatizaci-/v-prumyslu-2011_03_43212_6058/>. Luettu 10.9.2023.

4 Totally Integrated Automation Portal – Always ready for tomorrow. 2023. Verkkoaineisto. SIEMENS. <https://www.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html>. Luettu 22.9.2023.

5 The Complete Practical Guide to Siemens Tia Portal Programming. Verkkoaineisto. SolisPLC. <https://www.solisplc.com/tutorials/a-practical-guide-to-siemens-tia-portal-programming>. Luettu 16.10.2023.

6 SIMATIC WinCC WinCC Advanced V14 SP1. 2017. Verkkoaineisto. SIEMENS. <https://support.industry.siemens.com/cs/mdm/109747174?c=90171930635&dl=nl&lc=en-LY>. Luettu 4.1.2024.

7 Berger, Hans. 2014. Automating with SIMATIC S7-1500. Erlangen: Publicis Publishing.

8 Connecting a S7-1200 PLC / S7-1500 PLC to a SQL Database. 2022. Verkkoaineisto. SIEMENS. <https://support.industry.siemens.com/cs/us/en/view/109779336>. Luettu 8.1.2024.

9 FTP Client Communication with LFTP Library with S7-1500. 2022. Verkkoaineisto. SIEMENS. <https://support.industry.siemens.com/cs/us/en/view/81367009>. Luettu 12.1.2024.

10 SINAMICS G120C compact vector. 2023. Verkkoaineisto. SIEMENS. < https://www.siemens.com/us/en/products/drives/sinamics-electric-drives/low-voltage-drives/standard-performance-drives/sinamics-g120c.html>. Luettu 3.2.2024.

11 SINAMICS G120 & S120 Drives. 2020. Verkkoaineisto. ACD. <https://blog.acdist.com/sinamics-g120-s120-drives>. Luettu 1.3.2024.

12 Working with Libraries in Siemens TIA Portal - PLC Programming. Verkkoaineisto. SolisPLC. <https://www.solisplc.com/tutorials/working-with-libraries-in-siemens-tia-portal-plc-programming>. Luettu 29.2.2024.

13 Siemens AG. 2016. PROFINET with STEP 7 V14. Nürnberg: Division Digital Factory.

14 Programovací jazyky pro PLC. Verkkoaineisto. Coptel. <https://coptel.cz/mod/page/view.php?id=6737>. Luettu 24.11.2023.

15 Siemens AG. 2012. TIA Portal Module 010-020. Nürnberg: SCE Training Curriculum.

16 Ohjelmointirajapinta toimii siltana ohjelmistojen välillä. 2022. Verkkoaineisto. DNA. <https://www.dna.fi/yrityksille/blogi/-/blogs/ohjelmointirajapinta-toimii-siltana-ohjelmistojen-valilla>. Luettu 11.10.2023.

17 Avoin rajapinta. 2023. Verkkoaineisto. COSS. <https://coss.fi/palvelut/avoimuus/avoin-rajapinta/>. Luettu 11.10.2023.

18 Nový TIA Portal vám urychlí a zjednoduší prác. 2017. Verkkoaineisto. Technickytydenik. <https://www.technickytydenik.cz/rubriky/archiv-technik/novy-tia-portal-vam-urychli-a-zjednodusi-praci_40558.html>. Luettu 17.12.2023.

19 Siemens představuje vizualizační software WinCC v rámci nové verze TIA Portal V14. 2016. Verkkoaineisto. AUTOMA. <https://automa.cz/cz/casopis-clanky/siemens-predstavuje-vizualizacni-software-wincc-v-ramci-nove-ver/ze-tia-portal-v14-2016_11_0_9197/>. Luettu 18.12.2023.

20 Tenhunen, Ari. 2023. Siemens työntekijä. TCC2023_AddIns & Openness esitysmateriaali.

21 Siemens AG. 2022. TIA Portal Openness: API for automation of engineering workflows. Nürnberg: Digital Industries.

22 SIMATIC Openness: Automating creation of projects. 2018. Verkkoaineisto. Siemens. <https://support.industry.siemens.com/cs/document/109477163/simatic-openness-automating-creation-of-projects?dti=0&lc=en-AE>. Luettu 16.12.2023.

23 Siemens. 2023. TIA Portal Openness: Introduction and demo application. Germany: Siemens Industry Online Support

24 Tenhunen, Ari. 2023. Siemens työntekijä. TIA_Portal_V13_SP1_Openness_Rel1a_en_Webinar esitysmateriaali.

25 SIMATIC TIA Portal Openness. 2022. Verkkoaineisto. Siemens. <https://support.industry.siemens.com/cs/document/109815199/simatic-tia-portal-openness-api-for-automation-of-engineering-workflows?dti=0&lc=en-FI>. Luettu: 5.1.2024.

26 Siemens. 2022. TIA Portal Openness Explorer. Germany: Siemens Industry Online Support

27 Siemens. 2023. OpennessScripter: Introduction. Germany: Siemens Industry Online Support

28 Siemens. 2023. OpennessScripter: Detailed Documentation. Germany: Siemens Industry Online Support

29 What is automatic code generation. 2021. Blogi. Collimator. <https://www.collimator.ai/reference-guides/what-is-automatic-code-generation>. Luettu 20.9.2023.

30 Perintä ja abstraktit luokat luokkakaaviossa. 2017. Helsingin yliopisto mooc. <https://materiaalit.github.io/ohjelmointi-s17/part10/>. Luettu 2.1.2024.

31 What is object-oriented programming? OOP explained in depth. 2024. Blogi. Educative. <https://www.educative.io/blog/object-oriented-programming>. Luettu 25.2.2024.

32 What is Visual Studio Code? Microsoft’s extensible code editor. 2022. Verkkoaineisto. Martin Heller. <https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html>. Luettu 8.12.2023.

33 C# | .NET Framework (Basic Architecture and Component Stack). 2023. Verkkoaineisto. Geeksforgeeks. <https://www.geeksforgeeks.org/c-sharp-net-framework-basic-architecture-component-stack/>. Luettu 20.12.2023.

34 Microsoft Excel. 2020. Verkkoaineisto. Techopedia. <https://www.techopedia.com/definition/5430/microsoft-excel>. Luettu 22.12.2023.

35 What Is Visual Basic? Working. 2023. Verkkoaineisto. Spiceworks. <https://www.spiceworks.com/tech/devops/articles/what-is-visual-basic/>. Luettu 28.12.2023.

36 Explaining XML. 2000. Blogi. Eileen Roche. <https://hbr.org/2000/07/explaining-xml>. Luettu 4.1.2024.

37 Most popular technologies language. 2023. Verkkoaineisto. Stackoverflow. <https://survey.stackoverflow.co/2022/#most-popular-technologies-language>. Luettu 12.1.2024.

38 What Is Python Used For? A Beginner’s Guide. 2023. Verkkoaineisto. Coursera. <https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python>. Luettu 17.1.2024.

39 What is Git? 2021. Verkkoaineisto. Javatpoint. <https://www.javatpoint.com/git>. Luettu 20.1.2024.

40 Berger, Hans. 2014. Configuring, Programming and Testing with STEP 7 Professional. Erlangen: Publicis Publishing.

41 Overview of the problems automatic code generation can solve. 2021. Verkkoaineisto. Techtarget. <https://www.techtarget.com/searchsoftwarequality/tip/Overview-of-the-problems-automatic-code-generation-can-solve>. Luettu 14.2.2024.

42 Python Tutorial. Verkkoaineisto. Tutorialspoint. <https://www.tutorialspoint.com/python/index.htm>. Luettu 15.2.2024.

43 Python Introduction. Verkkoaineisto. w3schools. <https://www.w3schools.com/python/python_intro.asp>. Luettu 24.2.2024.

44 Flask Tutorial. 2023. Verkkoaineisto. Geeksforgeeks. <https://www.geeksforgeeks.org/flask-tutorial/>. Luettu 27.2.2024.

45 pythonnet - Python.NET. 2023. Verkkoaineisto. <https://github.com/pythonnet/pythonnet>. Luettu 1.3.2024.

46 Python XML Tutorial with ElementTree: Beginner's Guide. 2018. Verkkoaineisto. <https://www.datacamp.com/tutorial/python-xml-elementtree>. Luettu 2.3.2024.