jamk

Datalohkojen autonominen generointi TIA Portal-ohjelmointiympäristössä

Nikke Heiskanen

Opinnäytetyö
Toukokuu 2021
Tekniikan ala
Insinööri (AMK), sähkö- ja automaatiotekniikka

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Heiskanen, Nikke

Datalohkojen autonominen generointi TIA Portal-ohjelmointiympäristössä

Jyväskylä: Jyväskylän ammattikorkeakoulu. Toukokuu 2021, 29 sivua.

Tekniikan ala. Sähkö- ja automaatiotekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Verkkojulkaisulupa myönnetty: kyllä

Tiivistelmä

Toimeksiantajana opinnäytetyölle toimi konsultointi-, suunnittelu- ja projektinhoitoyhtiö CTS Engtec Oy. Työ tehtiin yhteistyössä Jyväskylän toimipisteen kanssa, mikä on suuntautunut sähkö- ja automaatiosuunnitteluun.

Toimeksiantaja halusi selvittää opinnäytetyön avulla TIA Portal Openness-rajapinnan mahdollisuudet ja rajoitukset sekä kuinka rajapintaa voidaan hyödyntää TIA Portal lohkojen luomiseen. Työn konkreettisena tavoitteena oli kehittää toimeksiantajalle sovellus, joilla pystytään nopeuttamaan TIA Portal automaatioprojekteissa käytettyjen datalohkojen luomista.

Datalohkojen luominen vaatii paljon yksittäisten parametrien ja asetusten syöttämistä manuaalisesti. Useiden manuaalisten toimenpiteiden toistaminen vie todella paljon aikaa ja lisää riskiä mahdollisiin kirjoitusvirheisiin. Edellä mainittuun työvaiheeseen haluttiin kehittää tehokkaampi toteutustapa ja tästä syystä opinnäytetyötä lähdettiin toteuttamaan yrityksen kanssa.

Sovellus kehitettiin Visual Studio-ohjelmankehitysympäristössä ja ohjelmointikielenä käytettiin C#. Kommunikointi sovelluksen ja TIA Portalin välille muodostettiin linkittämällä sovellukseen kaksi TIA Portal .dll tiedostoa. Nämä tiedostot mahdollistavat TIA Portal objektien lukemisen ja funktioiden suorittamisen sovelluksesta käsin. Datalohkojen luominen tapahtuu lukemalla Excel mallipohjasta lohkon parametrit, joista muodostetaan oikeanlainen lohko TIA Portaliin. Sovellus luo ensin annetuista parametreistä oikeanmallisen XML-tiedoston, joka importoidaan TIA Openness-rajapinnan avulla TIA Portal lohkoksi. Sovelluksen avulla on mahdollista importoida useita lohkoja samanaikaisesti.

Lopputuloksena saatiin kehitettyä tehtävänannon mukainen sovellus, jolla pystytään autonomisesti generoimaan TIA Portal datalohkoja. Tutkimustyön tuloksena pystyttiin kartoittamaan rajapinnan mahdollisuuksia ja rajoituksia ja näin ollen saatiin toimeksiantajalle parempi käsitys siitä mitä kaikkea oman sovelluksen ja rajapinnan avulla pystytään tekemään ja mitä ei.

Avainsanat (asiasanat)

TIA Portal, TIA Openness, generointi, Excel, C#, Visual Studio, datalohko

Muut tiedot (salassa pidettävät liitteet)

jamk

Description

Heiskanen, Nikke

Autonomous generation of data blocks in TIA Portal programming environment

Jyväskylä: JAMK University of Applied Sciences, May 2021, 29 pages.

Engineering and technology. Degree Programme in Electric and Automation Technology. Bachelor's thesis.

Permission for web publication: Yes

Language of publication: Finnish

Abstract

The thesis was commissioned by the consulting, engineering, and project management company CTS Engtec Oy. The thesis was done in collaboration with the Jyväskylä office, which is oriented towards electrical and automation engineering.

Purpose of this study was to find out the TIA Portal Openness application programming interface possibilities and limitations, as well as how the TIA Openness can be used to create TIA Portal blocks. The concrete goal of the thesis was to develop an application that can speed up the creation of data blocks used in TIA Portal automation projects.

Creating data blocks requires a lot of manual work and each block parameter must be written individually. Repeating several manual operations is really time consuming and increases the risk of possible mistakes. The aim was to develop a more efficient implementation method to create TIA Portal blocks and for this reason the thesis was started to be implemented with a company.

The application was created in the Visual Studio development environment and the programming language used was C#. Communication between the application and TIA Portal was established by linking two TIA Portal .dll files to the application. These files allow you to read TIA Portal objects and perform functions from within the application. Data blocks are created by reading the block parameters from the Excel file, which form the correct type of block for the TIA Portal. The application first creates a valid XML file from the given parameters, which is imported using the TIA Openness into a TIA Portal block. The application makes it possible to import several blocks at the same time.

As a result, an application according to the assignment was developed, which is able to autonomously generate TIA Portal data blocks. As a result of the research work, it was possible to identify the possibilities and limitations of the TIA Openness and thus the client was given a better understanding of what can be done with the application and TIA Portal.

Keywords/tags (subjects)

TIA Portal, TIA Openness, generate, Excel, C#, Visual Studio, Data block

Miscellaneous (Confidential information)

Sisältö

Kuviot

1 Johdanto

1.1 Toimeksiantajan esittely

CTS Engtec Oy on vuonna 1973 Kouvolaan perustettu insinööritoimisto. Pääkonttori sijaitsee Kouvolassa sekä toimipisteitä löytyy Jyväskylästä, Helsingistä, Turusta, Kotkasta ja Pietarista. Vuonna 2019 yrityksen liikevaihto oli yli 11 milj. € ja yrityksen henkilöstö muodostui tuolloin 167 henkilöstä. CTS Engtec tarjoaa hankkeiden eri vaiheisiin projektinhoito-, suunnittelu-, projektikehitys-, ja ylläpitopalveluitaan. Osaamista löytyy useilta eri toimialoilta, mutta yritys tunnetaan parhaiten metsäteollisuusosaamisesta. Opinnäytetyö tehtiin yhteistyössä Jyväskylän toimipisteen kanssa, mikä on puolestaan suuntautunut sähkö- ja automaatiosuunnitteluun. Jyväskylän toimiston osaamisalueita ovat erityisesti prosessiautomaatiojärjestelmien käyttöönotto ja suunnittelu. (CTS Engtec 2021; CTS Engtec_jyväskylä_automaatio 2019)

1.2 Toimeksiantajan tavoitteet

TIA Portal on monipuolinen ohjelmointityökalu, johon on yhdistetty perinteiset Siemensin ohjelmistot, joilla mahdollistetaan esimerkiksi logiikoiden ja näyttöpaneelien ohjelmointi sekä käyttöliittymän suunnittelu. Siemens on myös kehittänyt TIA Openness -ohjelmointirajapinnan, jonka avulla mahdollistetaan omien sovellusten ja TIA Portalin välinen kommunikointi ja tämän avulla voidaan helpottaa ja automatisoida tiettyjä työvaiheita.

Toimeksiantaja oli tietoinen, että TIA Openness -ohjelmointirajapinnan avulla on mahdollista luoda oma sovellus, joka voisi nopeuttaa TIA Portal datalohkojen luomista projekteja tehdessä. Tiedettiin, että kyseinen sovellus olisi mahdollista tehdä, mutta sen tarkempi toteutustapa ja mahdolliset hyödyt ja rajoitukset täytyi selvittää itse.

Uutta projektia luodessa pitää luoda omat datalohkot muun muassa jokaiselle anturimittaukselle, venttiilille, moottorille ja säätimelle. Projektista riippuen datalohkoja voi olla useista kymmenistä moniin satoihin. Näiden datalohkojen pitää lisäksi olla toimeksiantajan lohkokirjaston mukaisia. Vaikka näille jokaiselle lohkoille löytyy valmiit pohjatiedostot toimeksiantajan kirjastosta, niin osa lohkojen parametreistä täytyy syöttää manuaalisesti yksi kerrallaan. Useiden manuaalisten toisto-

jen tekeminen lisää myös riskiä virheisiin. Toimeksiantajan toive siis oli etsiä ratkaisu ja toteutustapa uudelle sovellukselle, jolla pystytään automatisoimaan edellä mainittu datalohkojen luominen. Sovellus auttaisi työntekijöitä vähentämään manuaalisessa työssä syntyviä kirjoitusvirheitä ja vapauttamaan enemmän aikaa muihin olennaisiin työtehtäviin. Siemens ei tarjoa valmiita työkaluja lohkojen autonomiseen generointiin ja tästä syystä opinnäytetyötä päätettiin yhdessä yrityksen kanssa lähteä toteuttamaan.

1.3 Henkilökohtaiset tavoitteet

Henkilökohtaisena tavoitteenani oli syventyä TIA Portal -ohjelmointityökaluun ja perehtyä tarkemmin siinä käytettävien datalohkojen rakenteisiin ja luomisprosessiin. TIA Portal on yksi käytetyimmistä automaatioalan ohjelmointityökaluista ja sen osaamisesta on todennäköisesti hyötyä tulevaisuuden työtehtävissä (PLC programming n.d.). TIA Openness -rajapinta ei ollut entuudestaan tuttu ja tavoitteenani olikin tutustua sen mahdollisuuksiin sekä oppia kuinka rajapinnan ja TIA Portalin välinen kommunikointi tapahtuu. Oman sovelluksen tekeminen on aina kiehtonut ja tämä olikin oiva tilaisuus ottaa selvää, kuinka sellainen onnistuu käytännössä. Tavoitteena oli parantaa omia ohjelmointitaitoja siten, että tulevien sovellusideoiden aloittaminen ei vaatisi niin paljoa opettelua, vaan kehitys voitaisiin aloittaa ilman perusteiden läpikäymistä. Oikeaan ongelmaan ratkaisun etsiminen ja uuden tuotteen kehittäminen motivoi itseäni aivan eri tavalla. Yhtenä itselle asetettuna tavoitteena oli myös kehittää tyypillisen projektinhallinnan taitoja kuten aikatauluttamista, tiedon keräämistä ja sen organisointia.

2 Tietoperusta

2.1 Automaatio yleisesti

Automaatiolla tarkoitetaan ohjelmoitua järjestelmää tai laitetta, mikä toimii itsenäisesti ilman ihmisen jatkuvaa vaikutusta ja ohjausta. Tarkoituksena on yleensä korvata ihmisten tekemä lihas- tai päättelytyö ohjelmoidulla logiikkalaitteella, joka suorittaa sille annettuja komentoja autonomisesti. Logiikan toiminta perustuu sähköisiin tulo- ja lähtöliitäntöihin, joita ohjaamalla voidaan suorittaa haluttuja toimenpiteitä. Logiikkalaitteen lähtöjen avulla voidaan esimerkiksi sytyttää valoja päälle ja pois. Tuloja voidaan puolestaan käyttää takaisinkytkentään ja seurata haluttua mittausta.

Logiikkaan tulevaa mittaustietoa voidaan hyödyntää lähtöjen ohjaamiseen ja mahdollistetaan halutun prosessiarvon säätäminen tai pitäminen tasaisena. (Automaatio ja automaatiojärjestelmät n.d; Gates 2018; Prosessitekniikan perusta Automaatiotekniikka n.d.)

Yksinkertaisia esimerkkejä automaatiosta ovat liikennevalojen ohjaus ja huoneen lämpötilan säätäminen. Logiikkalaitteilla on myös mahdollista toteuttaa kokonaisen tehtaan automaatiojärjestelmä, jolla voidaan hallita tehtaan toimintaa valvomosta käsin. Valvomo onkin hyvin olennainen osa automaatiojärjestelmää, sillä sen avulla yleensä ohjataan ja seurataan prosesseja. (Automaatio ja automaatiojärjestelmät n.d.)

Automaatiojärjestelmiä hyödyntäminen teollisuudessa on hyvin yleistä. Jotain tiettyjä manuaalisia työvaiheita on tuotannon puolesta kannattavampaa automatisoida robotilla tai koneella. Automaatiolla voidaan parantaa työn tehokkuutta, luotettavuutta, tuotantokustannuksia, ja turvallisuutta sekä pienentää jätteiden määrää ja työvoiman tarvetta. Haittapuolina ovat kuitenkin järjestelmien ja laitteiden kalliit hankintakustannukset ja suurempi tarve kunnossapitoon. (Jones 2019.)

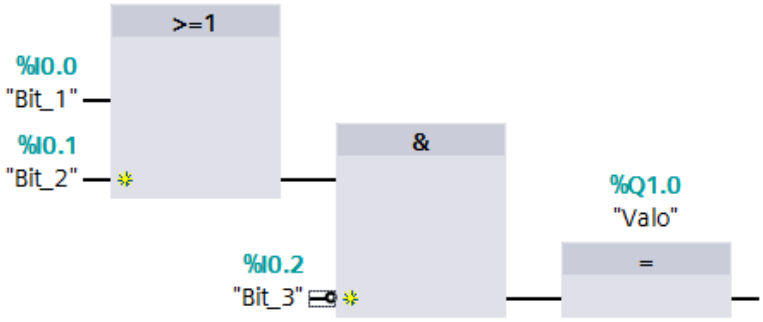
2.2 TIA Portal -ohjelmointiympäristö

TIA Portal (Totally Integrated Automation) on Siemensin kehittämä ohjelmointiympäristö automaatioprojektien ohjelmointiin ja suunnitteluun. TIA Portaliin on yhdistetty Siemensin perinteiset ohjelmistot kuten WinCC, SINAMICS startdrive, STEP 7, SIMOTION SCOUT TIA ja SIMOCODE ES. Useiden ohjelmistojen yhdistäminen mahdollistaa lähes kaiken projektiin liittyvien asioiden hoitamisen yhden sovelluksen avulla. TIA Portalissa voidaan esimerkiksi tehdä käyttöliittymien, logiikkojen, turvaratkaisujen ja taajuusmuuttajien ohjelmointia sekä automaation ylläpitoa, diagnosointia ja konfigurointia. TIA Portal on suunniteltu helppokäyttöiseksi ja tehokkaaksi työkaluksi, jolla mahdollistetaan useiden ominaisuuksien sulava toimivuus keskenään. (Software in TIA Portal n.d; TIA-PORTAL n.d.)

2.3 Logiikkaohjelmointi

Logiikkaohjelmoinnilla tarkoitetaan erinäisten toimintojen ja komentojen syöttämistä logiikkalaitteeseen, jotta se osaa toimia autonomisesti juuri halutulla tavalla. Komentojen suunnittelut tapahtuvat TIA Portalissa ja kun koodi on valmis, niin se ladataan logiikkaan.

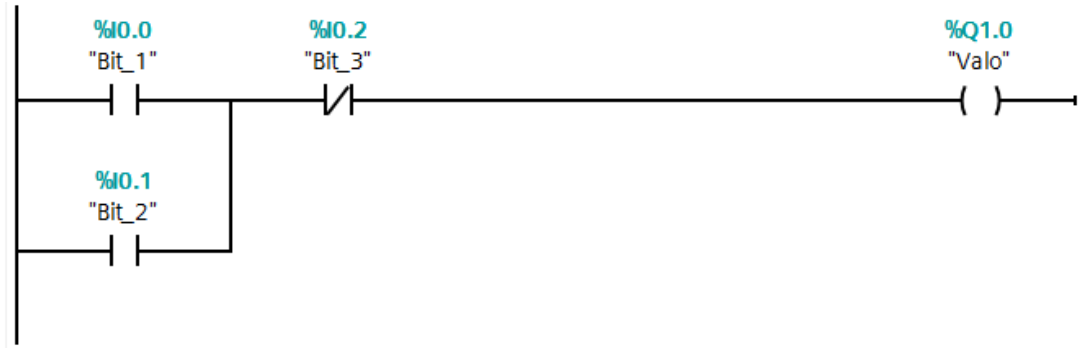
TIA Portalissa logiikkaohjelmointi tapahtuu muodostamalla erilaisia lohkoja yhteen, kuten organisointilohkoja (OB), datalohkoja (DB), funktioita (FC) ja toimilohkoja (FB). Funktioita ja toimilohkoja on mahdollista ohjelmoida viidellä eri ohjelmointikielellä: Function Block Diagram (FBD), S7-GRAPGH, Statement List (STL), Ladder Diagram (LAD) ja Structured Control Language (SCL). S7-GRAPH on askelmaisen sekvenssien ohjelmointiin suunnattu ohjelmointikieli, missä askelten komennot tehdään joko LAD tai FBD kielellä. Alla on yksinkertainen esimerkkiohjelma toteutettu edellä mainituilla eri ohjelmointikielillä (ks. kuviot 1–4). Ohjelma sytyttää valon vain silloin kun Bit_3 on pois päältä ja joko Bit_1 tai Bit_2 on päällä. (Dixon 2018; S7-GRAPH V5.3 for S7-300/400 Programming Sequential Control Systems 2004; SIMATIC STEP 7 Basic/Professional V15.1 and SIMATIC WinCC V15.1 2018, 36; 4515-4519.)



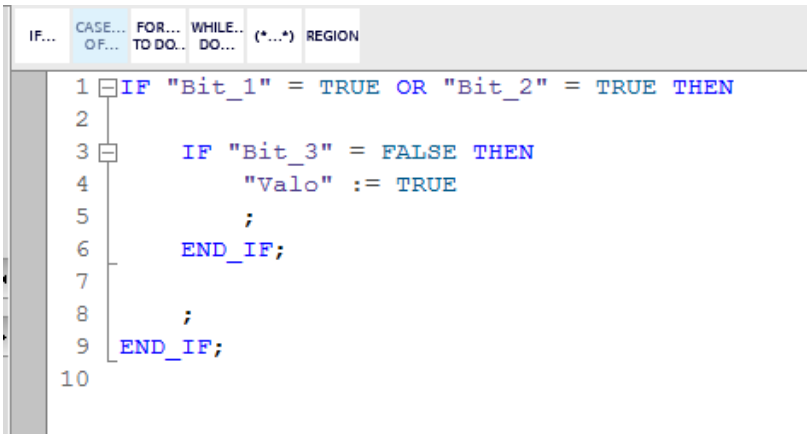
Kuvio 1. Function Block Diagram (FBD)

```
1       A(
2       A       "Bit_1"                 %I0.0
3       O       "Bit_2"                 %I0.1
4       )
5       AN      "Bit_3"                 %I0.2
6       =       "Valo"                  %Q1.0
```

Kuvio 2. Statement List (STL)



Kuvio 3. Ladder Diagram (LAD)



Kuvio 4. Structured Control Language (SCL)

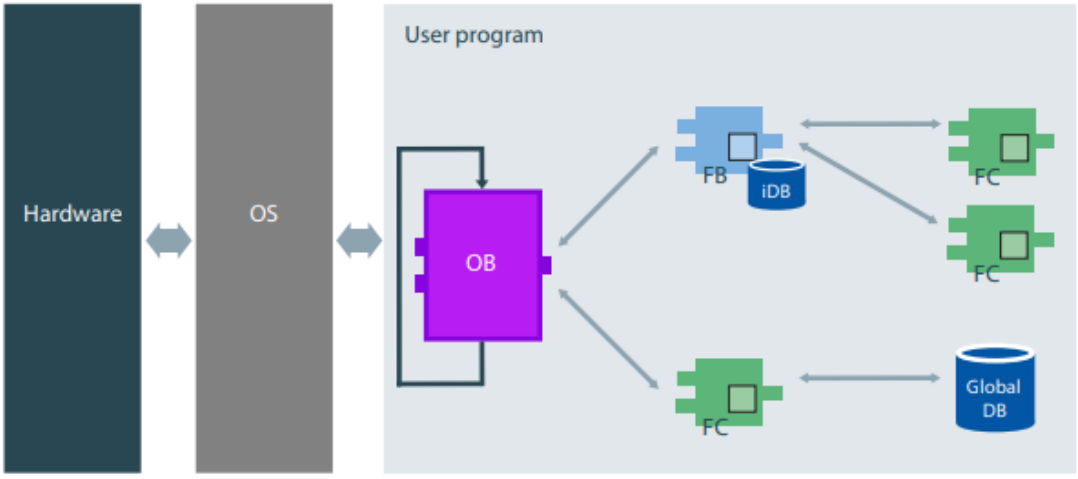
Funktiot ovat nimensä mukaisesti lohkoja, jotka suorittavat sen sisälle syötetyt komennot. Esimerkiksi yhden funktiolohkon sisälle voitaisiin tehdä laskutoimitus, jolla lasketaan säiliön tilavuus litroina säiliön pinnankorkeutta mittaavan ultraäänianturin sähkövirrasta. Koko ohjelman komennot olisi mahdollista syöttää yhden funktiolohkon sisälle, mutta koska komentoja on lähes aina todella paljon, niin ohjelman selkeyttämiseksi komennot jaetaan erillisiin lohkoihin. Esimerkiksi säiliön tilavuuden laskemiselle ja säiliön poistoventtiilin ohjaukselle tehdään yleensä omat lohkot. (SIMATIC STEP 7 Basic/Professional V15.1 and SIMATIC WinCC V15.1 2018, 4516.)

Isoimmissa projekteissa saattaa ilmetä useasti toistuvia toimenpiteitä, joiden logiikkaohjelmointi vaatii useiden samankaltaisten funktiolohkojen luomista. Tällaisissa tilanteissa voidaan hyödyntää toimilohkoja, jotka käyttäytyvät kuten funktiolohkot, mutta niiden parametrejä ja sisääntuloarvoja

on mahdollista muokata. Esimerkiksi jos tehtaan automaatiojärjestelmässä on käytössä satoja samanlaisia säiliön pinnanmittauksia, niin useiden funktiolohkojen sijaan voidaan käyttää vain yhtä toimilohkoa. Funktiolohkoista poiketen toimilohkoilla on puolestaan jokaista mittausta varten oma linkitetty datalohko, johon arvot jäävät talteen, vaikka ohjelma olisi jo suoritettu. Funktiolohkoilla ei sen sijaan ole omaa muistia eikä niiden arvot tallennu sykliltä toiselle. (SIMATIC STEP 7 Basic/Professional V15.1 and SIMATIC WinCC V15.1 2018, 4517.)

Toimilohkoihin linkitettyjä datalohkoja kutsutaan instanssidatalohkoiksi, joidenka rakenne määräytyy myös toimilohkon mukaan. Toisena datalohkotyyppinä on globaalidatalohko, jonka rakenne ja muuttujat pitää tehdä käsin. Globaalidatalohkoin pääsee kaikki ohjelmien lohkot käsiksi, kun taas instanssidatalohkot on osoitettu tietylle toimilohkolle. TIA Portalissa on myös mahdollista tehdä tageista koostuvia tagilistoja. Tagit ovat ohjelmaan syötettyjä absoluuttisia osoitteita, jotka on mahdollista nimetä vapaasti. (SIMATIC STEP 7 Basic/Professional V15.1 and SIMATIC WinCC V15.1 2018, 4517–4519; What is the difference between an instance data block and a global data block and how does a CALL call influence the DB register? 2011)

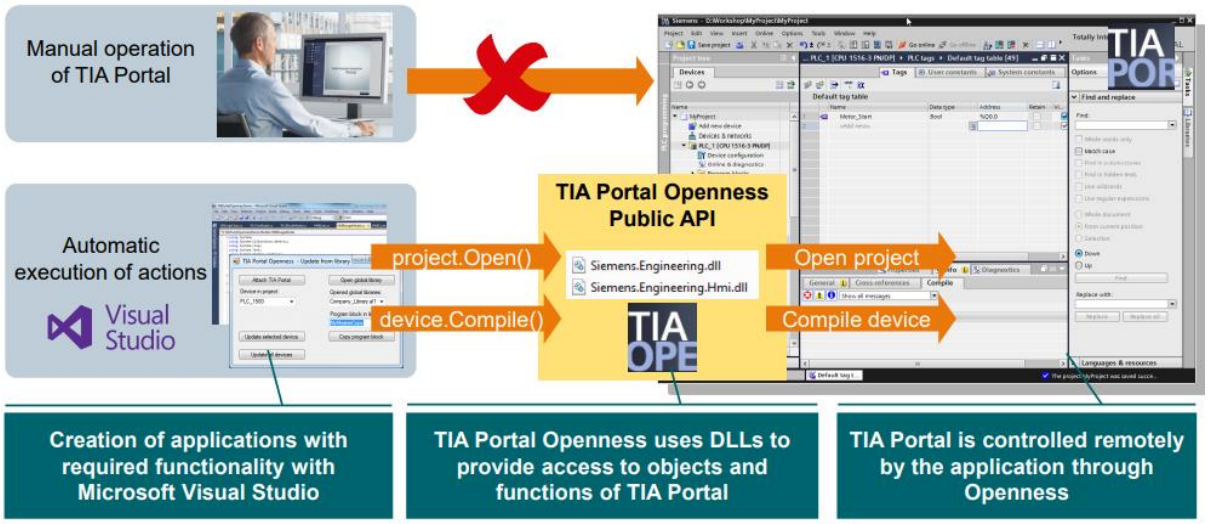
Organisointilohko on ohjelman korkein lohko, joka suoritetaan aina ensimmäisenä. Organisointilohko ohjaa myös virheiden käsittelyä, logiikan käynnistyskäyttäytymistä ja ohjelman suoritussykliä. Organisointilohkoon on mahdollista kirjoittaa suoritettavia toimintoja, mutta yleisesti lohkoa käytetään ajamaan muita alempia lohkoja kuten FB, FC tai muita OB (ks. kuvio 5). Halutut lohkot sijoitetaan OB:hen ja ne ajetaan edellä mainitussa järjestyksessä. (SIMATIC STEP 7 Basic/Professional V15.1 and SIMATIC WinCC V15.1 2018, 4515–4516.)



Kuvio 5. Logiikkaohjelman rakenne (SIMATIC STEP 7 Basic/Professional V15.1 and SIMATIC WinCC V15.1 2018, 4511.)

2.4 TIA Openness -rajapinta

TIA Portal Openness on Siemensin valmistama rajapinta TIA Portal ohjelmointiympäristöön. Rajapinta mahdollistaa kommunikoinnin omien sovellusten ja TIA Portalin käyttöliittymä- ja logiikkaohjelmoinnin työkalujen välillä. Tämä tarkoittaa, että oman sovelluksen avulla voidaan esimerkiksi suorittaa TIA Portal komentoja tai hakea tietoa projektitiedostoista. Rajapinnan ja oman sovelluksen toimiminen edellyttää kahden TIA Portal .dll-tiedoston linkittämistä sovellukseen. Nämä .dll-tiedostot ovat jaettuja kirjastotiedostoja, jotka sisältävät tarvittavat muuttujat ja menetelmät, jotta TIA Portal objekteihin ja toimintoihin päästään ulkoisen sovelluksen avulla käsiksi (ks. kuvio 6).

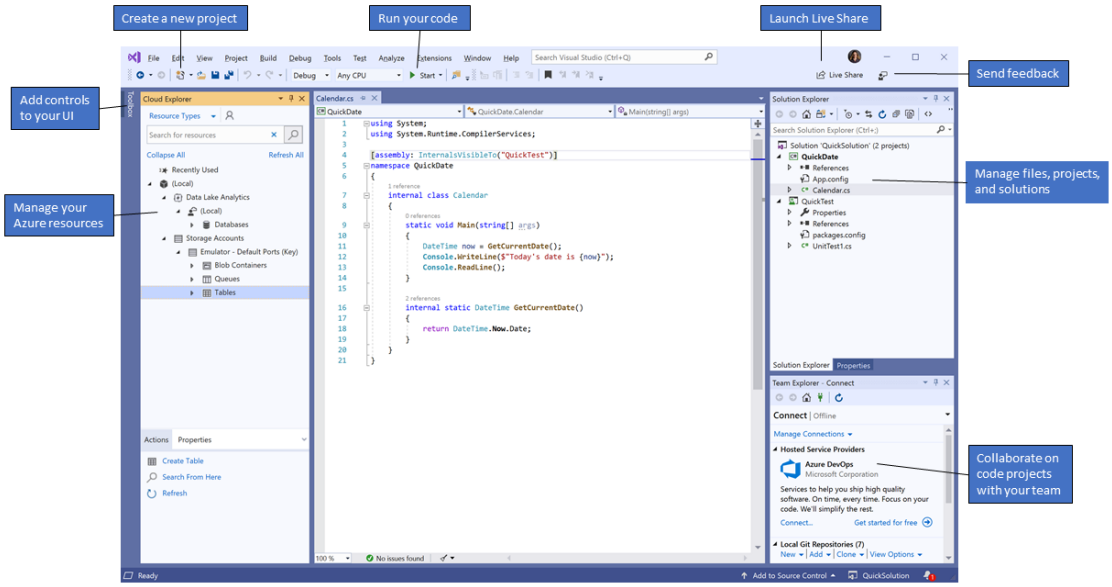


Kuvio 6. TIA Openness toiminta (TIA Portal Openness 2018.)

Omien ohjelmien yhdistäminen TIA Portaliin ei varsinaisesti tuo mitään uusia ominaisuuksia, kun rajapinnalla on käytössä samat komennot kuin TIA Portalissa. Omien sovellusten hyötyinä ovat kuitenkin useiden komentojen suorittaminen autonomisesti, minkä avulla voidaan helpottaa joidenkin työvaiheiden tekemistä sekä vähentää virheitä. Koska rajapinnan avulla on mahdollista lukea useita ohjelmointityökalun objekteja samanaikaisesti, voidaan projektitiedoston objekteista etsiä tietoa nopeammin kuin perinteisesti selaamalla yksittäisiä objekteja kerrallaan. Teoriassa olisi myös mahdollista suunnitella oma sovellus uudenlaisella käyttöliittymällä mikä toimisi TIA Portin tavoin. (Excel code generator for TIA Portal Openness 2019; Smith, 2019; TIA Portal Openness: Introduction and Demo Application 2020.)

2.5 Visual Studio

Visual Studio on Microsoftin valmistama ohjelmankehitysympäristö, joka tukee monipuolisesti eri ohjelmointikieliä ja laajennuksia (ks. kuvio 7). Visual Studio mahdollistaa erilaisten nettisivujen, mobiili- ja tietokonesovellusten kehittämisen. Ohjelman työkalujen avulla voidaan kirjoittaa, korjata ja testata koodia kätevästi. Tavallisista ohjelmointiympäristöistä poiketen Visual Studio mahdollistaa myös sovelluksen visuaalisen suunnittelun, koodin täydentämisen ja kokoamisen. Kattava työkalujen määrä helpottaa sovelluksen kehitysprosessia. Visual Studio onkin eniten käytetty ohjelmointiympäristö koko maailmassa. (Top IDE index 2021; Visual Studio 2019 n.d; Welcome to the Visual Studio IDE 2019.)



Kuvio 7. The Visual Studio 2019 ohjelmointiympäristö (Welcome to the Visual Studio IDE 2019.)

2.6 Ohjelmointikielet C#/VB

C# (lausutaan englanniksi C sharp) on Microsoftin luoma olio-orientoitunut ohjelmointikieli, joka toimii .NET Frameworkissa. Ensimmäinen C# versio julkaistiin 2002. C#:n juuret ovat C ohjelmointikielestä ja on hyvin samankaltainen kuin C++ tai Java. C# ohjelmointikielen osaamista on siis helppo soveltaa edellä mainittujen kielien käytössä ja päinvastoin. C# on hyvin yksinkertainen ohjelmointikieli ja rakenteeltaan selkeälukuinen (ks. kuvio 8). Se on myös yksi käytetyimmistä ohjelmointikielistä. (A tour of the C# language 2021; C# Introduction n.d.)

```
using System;

namespace Hello
{
    0 references
    public class HelloWorld
    {
        0 references
        public static void Main(string[] args)
        {
            string name = "C#";

            // See if an argument was passed from the command line
            if (args.Length == 1)
                name = args[0];

            Console.WriteLine("Hello, " + name + "!");
        }
    }
}
```

Kuvio 8. ”Hello world” sovellus C#-kielellä (VB.NET and C# Comparison 2016.)

Visual Basic on Microsoftin luoma yksinkertainen olio-orientoitunut ohjelmointikieli. VB on suunniteltu ymmärrettäväksi aloitteleville ja kokeneille ohjelmoijille. Ensimmäinen Visual Basic versio julkaistiin 1991, mutta kieli on kuitenkin kehittynyt ajan saatossa. Vuonna 2002 julkaistiin mullistava versio Visual Basic.Net, jossa ohjelmointikieleen yhdistyi .NET Framework. Visual Basic.Net version avulla sovelluksia pystytään tekemään ohjelmointikielien C++, C# ja Javan kaltaisesti (ks. kuvio 9). (Lesson 1 : Introduction to Visual Basic 2020; What is VB.Net? Introduction, History, Features, Advantages, Disadvantages. n.d.)

```
Imports System

Namespace Hello
    0 references
    Class HelloWorld
        0 references
        Overloads Shared Sub Main(ByVal args() As String)
            Dim name As String = "VB.NET"

            'See if an argument was passed from the command line
            If args.Length = 1 Then name = args(0)

            Console.WriteLine("Hello, " & name & "!")
        End Sub
    End Class
End Namespace
```

Kuvio 9. ”Hello world” sovellus VB-kielellä (VB.NET and C# Comparison 2016.)

2.7 XML-tiedostot

TIA Openness rajapinta mahdollistaa oikean mallisten XML-tiedostojen importoinnin TIA Portal toimilohkoiksi. Jotta importoiminen onnistuisi, niin haluttu data täytyy muuntaa ensin XML-tiedoston muotoon.

XML eli “Extensible Markup Language” on merkintäkieli, jota käytetään datan varastointiin ja siirtämiseen järjestelmien välillä. XML-tiedosto sisältää tageja ja tekstejä. Tagit muodostavat datan rakenteen ja sijainnin. Teksti on puolestaan tageihin yhdistetty data, joka halutaan varastoida. XML-tiedostot ovat siis tietyn tyyppisiä tekstitiedostoja, jotka käyttävät räätälöityjä tageja kuvaamaan tiedoston rakennetta ja kuinka dataa tulisi tallentaa ja kuljettaa (ks. kuvio 10). (What is an XML file and How Do I Open One? 2021.)

XML-tiedostojen hyötyinä ovat niiden helppolukuisuus ja muokattavuus. Tiedostot käyttävät tietokonekielen sijaan ihmiskielen todellisia sanoja ja näin ollen ovat selkeästi ymmärrettävissä. Tiedostojen yhdenmukainen rakenne ja hyvin organisoidut tagit mahdollistavat tiedon käsittelyn riippumatta käyttöjärjestelmästä tai laitteesta. (XML:n perusteet. n.d.)

```
<?xml version="1.0"?>
- <ROOT>
  - <Customers>
    - <Customer CustomerName="Arshad Ali" CustomerID="C001">
      - <Orders>
        - <Order OrderDate="2012-07-04T00:00:00" OrderID="10248">
            <OrderDetail Quantity="5" ProductID="10"/>
            <OrderDetail Quantity="12" ProductID="11"/>
            <OrderDetail Quantity="10" ProductID="42"/>
          </Order>
        </Orders>
        <Address> Address line 1, 2, 3</Address>
      </Customer>
    - <Customer CustomerName="Paul Henriot" CustomerID="C002">
      - <Orders>
        - <Order OrderDate="2011-07-04T00:00:00" OrderID="10245">
            <OrderDetail Quantity="12" ProductID="11"/>
            <OrderDetail Quantity="10" ProductID="42"/>
          </Order>
        </Orders>
        <Address> Address line 5, 6, 7</Address>
      </Customer>
    - <Customer CustomerName="Carlos Gonzlez" CustomerID="C003">
      - <Orders>
        - <Order OrderDate="2012-08-16T00:00:00" OrderID="10283">
            <OrderDetail Quantity="3" ProductID="72"/>
          </Order>
        </Orders>
        <Address> Address line 1, 4, 5</Address>
      </Customer>
    </Customers>
</ROOT>
```

Kuvio 10. Esimerkki XML-tiedoston rakenteesta (Arshad 2019.)

3 Toteutus

Työ aloitettiin kartoittamalla toimeksiantajan tarpeet sekä käymällä läpi annetut lähtötiedot. Heiltä saatu tehtävänanto oli hyvin suppea, mutta työn tarkoituksena olikin ottaa selvää uuden sovelluksen toimintaperiaatteesta sekä selvittää, mitä kaikkea sovelluksen ja rajapinnan avulla olisi mahdollista tehdä. Lähtötietoja kirjattiin ylös haastattelemalla suunnittelijoita, joille sovellus tulisi käyttöön. Haastatteluissa selvisi, että en ole ensimmäinen, joka on lähtenyt selvittämään rajapinnan mahdollisuuksia ja generaattorin toimintaperiaatetta. Kyseisistä aiheista löytyikin muutamia esimerkkidemoja ja artikkeleita, millä suunnittelutyö saatiin tehokkaasti aluille.

3.1 XML-tiedostojen generointi

Tutkimuksesta selvisi, että TIA Opennessin julkaiseman demosovelluksen avulla on mahdollista muuntaa XML-tiedostoja TIA Portal lohkoiksi, jos ne ovat juuri oikean mallisia. Työn konkreettisena tavoitteena olikin luoda sovellus, joka generoi Exceliin syötetystä datasta oikean muotoisia XML-tiedostoja, jota voidaan hyödyntää TIA Opennessissa. Jotta XML-tiedostojen luominen onnistuisi, jouduttiin niiden rakenteeseen ja toiminnallisuuteen perehtymään tarkemmin. Tämän tiedon selvittämisessä hyödytti edellä mainittu demosovellus, jonka avulla oli mahdollista myös suorittaa tämä haluttu toimenpide päinvastaisessa järjestyksessä. Demosovelluksen avulla oli siis mahdollista luoda valmiista TIA Portal lohkoista XML-tiedostoja. Näistä XML-tiedostoista pystyttiin ottamaan mallia, että miltä generaattorin lopulliset tiedostot tulisi näyttää. XML-tiedostoista nähtiin myös selkeästi, että missä TIA Portal ohjelman parametrit sijaitsevat. Parametrien paikantaminen helpotti ymmärtämään paremmin XML-tiedostojen rakennetta. Näitä esimerkkitiedostoja luodessa käytettiin toimeksiantajan omaa lohkokirjastoa ja jokaisesta erilaisesta mittaus-, säädin-, venttiili-, pumppu-, venttiili-, globaalidatalohkosta ja tagilistasta tehtiin sovellukselle omat XML-mallitiedostot.

Kun XML-tiedostojen rakenne oli saatu selville, niin seuraavaksi lähdettiin ottamaan selvää, kuinka sellaisia pystytään luomaan sekä kuinka niiden parametrejä voidaan korvata ja lukea valmiista Excel-taulukosta. Toimeksiantaja suositteli käyttämään kyseisen generointisovelluksen valmistamiseen VB-ohjelmointikieltä ja Visual Studio-ohjelmankehitysympäristöä. Pitkän tiedonetsinnän ja kokeilujen jälkeen saatiin aikaiseksi sovelluksen ensimmäinen versio**.** Sovelluksella pystyttiin valitsemaan Excel-tiedosto, josta parametrit luetaan ja toimilohko, jonka XML-tiedostoon parametrit

kirjoitetaan. Käytännössä ohjelma siis ottaa valitun lohkon XML-mallitiedoston ja korvaa siihen Excelistä luetut parametrit. Ohjelmaan tehtiin myös toimeksiantajan toiveiden mukaisesti mahdollisuus luoda globaalidatalohkoja tai tagilistoja, niin että halutut muuttujat luetaan valmiista Excel tiedostosta (ks. kuvio 11). Tämä ominaisuus oli helppo lisätä, koska molempien lohkojen generointi onnistuu lähes samalla tavalla, eikä niiden rakenne ole riippuvainen mitenkään toimeksiantajan lohkokirjastosta.

	1	2	3	4	5	6
1	Name	Data Type	Startvalue	Comment	NameDB	NumberDB
2	Temp1	Int	100	temperature1	ekaDB	900
3	Temp2	Int	200	temperature2	ekaDB	900
4	Temp3	Int	300	temperature3	ekaDB	900
5	Valve10	Bool	False	shutvalve1	TokaDB	901
6	Valve20	Bool	True	shutvalve2	TokaDB	901
7	Valve30	Bool	False	shutvalve3	TokaDB	901
8	Valve40	Bool	True	shutvalve4	TokaDB	901
9						
10						

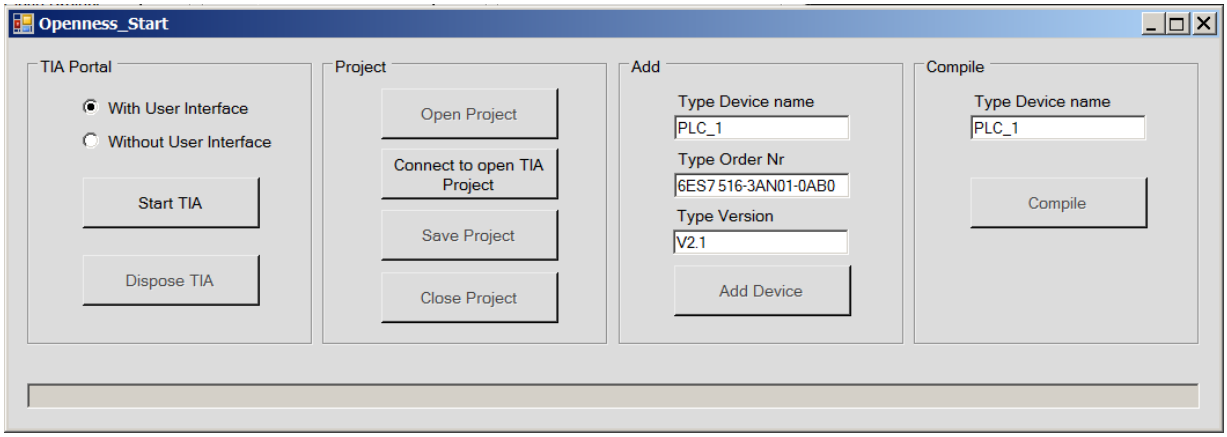
Kuvio 11. Globaalidatalohkon parametrit Excelissä

Vaikka sovelluksen ensimmäisellä versiolla pystyttiin generoimaan oikeanlaisia XML-tiedostoja ja täyttämään toimeksiantajan kriteerit, niin ei se silti ollut tarpeeksi käytännöllinen ja aikaa säästävä, että sitä olisi kannattanut käyttää. Tämä prosessi Excel tiedostosta lopulliseksi lohkoksi vaatii juuri oikean mallisen Excelin luomisen, joka muunnetaan uuden sovelluksen avulla XML-tiedostoksi. Tämä tiedosto syötetään TIA Openness demosovellukseen, joka sitten vasta siirtää tiedoston lohkoksi TIA Portaliin. Demosovelluksen avulla on mahdollista luoda lohkoja vain yksi kerrallaan, joten edellä mainittu prosessi jouduttaisiin toistamaan useita kertoja, mistä johtuen se olisi todella hidas ja epäkäytännöllinen.

3.2 Toimilohkojen importoiminen

Opinnäytetyössä tehtävänä oli myös ottaa selvää, mitä kaikkea rajapinnan ja oman sovelluksen avulla on mahdollista toteuttaa. Tästä syystä tutkittiin, onko mahdollista luoda vain yksi sovellus, mikä pystyisi hoitamaan datan lukemisen Excelistä, generoinnin sekä useiden lohkojen valmistamisen vain napin painalluksella.

Osa TIA Openness demosovelluksista (ks. kuvio 12) oli tehty C# ohjelmointikieltä käyttäen, joten niiden toimintaperiaatteita lähdettiin tutkimaan tarkemmin. Tutkimustyön tarkoituksena oli etsiä demotiedostoista toteutusmenetelmiä, mitä voitaisi hyödyntää oman sovelluksen kehittämisessä. Samalla havaittiin, että Siemens on dokumentoinut todella kattavasti, kuinka C# komentoja pystytään hyödyntämään TIA Openness-sissa. Kattava dokumentointi ja C# esimerkkisovellukset johtivat oman sovelluksen ohjelmointikielen vaihtamiseen. Molemmat ohjelmointikielet ovat hyvin samankaltaisia, joten kielen vaihtaminen oli hyvin luontevaa. Yhdessä Openness demosovelluksissa oli luotu yhteyden mahdollistaminen TIA Portalin ja sovelluksen välille, joten tätä demosovellusta päätettiin käyttää uuden sovelluksen pohjana ja rakentaa kaikki muut ominaisuudet tähän päälle.



Kuvio 12. TIA Openness demosovellus (TIA Portal Openness: Introduction and Demo Application 2020.)

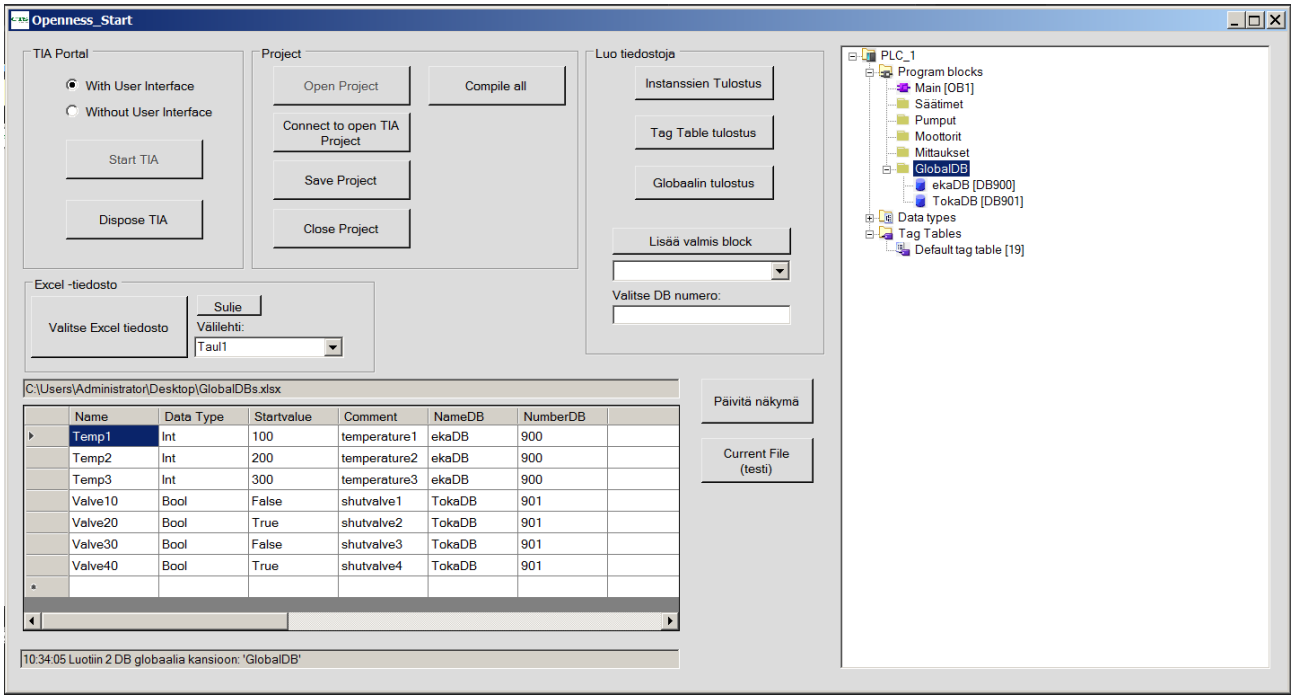
Aikaisemmin luotu generaattori yhdistettiin tähän uuteen C# sovellukseen. Nyt sovelluksen avulla on mahdollista luoda yhteys Excelin ja TIA Portalin välille sekä pystytään luomaan lohkoja vain tämän yhden sovelluksen avulla. Tässä vaiheessa Exceliin on myös mahdollista kirjoittaa useampien lohkojen parametrit ja viedä kaikki samanaikaisesti TIA Portaliin. Esimerkiksi useiden kymmenien lohkojen luominen samanaikaisesti on huomattavasti nopeampaa kuin sovelluksen ensimmäisellä versiolla tai manuaalisesti käsin kirjoittamalla.

3.3 Käyttöliittymän ja sovelluksen parantelu

Koska kyseessä on kuitenkin uuden sovelluksen kehittäminen, jonka tarkoituksena on nopeuttaa jotain tiettyä työvaihetta, niin sovelluskehitystä on mahdollista jatkaa lähes ikuisesti. Sovellusta

luodessa syntyi paljon kehitysideoita, joilla pystyttäisiin helpottamaan sovelluksen käyttämistä ja nopeuttamaan sen käyttöönottoa. Sovellukseen lisättiin paljon visuaalisia ominaisuuksia, kuten mahdollisuus nähdä valitun Excel-tiedoston sisällä olevat sarakkeet ja TIA Portal projektinäkymä (ks. kuvio 13). Näiden avulla käyttäjä pystyy helposti varmistamaan, että mitä dataa ollaan siirtämässä ja mihin projektikansioon. TIA Portal projektinäkymästä on mahdollista valita toimilohkojen tulostuskansio, voidaan tehdä uusia kansioita ja poistaa tiedostoja. Uusia projekteja luodessa nämä voivat olla mukava lisä organisoinnin parantamiseksi.

Sovellukseen tehtiin useita lukituksia ja varoitusilmoituksia, jos käyttäjä on tekemässä jotain mikä saattaisi sotkea TIA Portal projektitiedostoa. Esimerkiksi jos vaikka käyttäjä yrittää luoda lohkoa, jonka nimi tai numero on jo olemassa, niin ohjelma estää päällekirjoituksen ja antaa käyttäjälle virheilmoituksen sekä kertoo, miksi toiminto estettiin. Sama lukitus saatiin toimimaan myös datalohkojen ja taglistojen muuttujien kanssa. Jokainen projektin muuttuja käydään yksitellen läpi ja jos tulostettava muuttuja on jo olemassa, niin lohkon luominen estetään. Koska instanssidatalohkot vaativat linkitetyn toimilohkon toimiakseen, niin virheilmoitus tulee myös, jos linkitettyä toimilohkoa ei löydy. Koska toimeksiantajan projekteissa toimilohkot ovat lähtökohtaisesti aina samat, niin ohjelmaan tehtiin ominaisuus, jonka avulla pystytään lisäämään tällainen puuttuva toimilohko, jotta instanssidatalohkojen tulostaminen onnistuu.



Kuvio 13. Valmis sovellus, jossa vasemmalla on lähtötietoina oleva Excel ja oikealla TIA Portal projektin rakenne.

4 Tulokset

Valmis sovellus suunniteltiin toimimaan ohjelmien TIA Portal 15.1 ja TIA Openness 15.0 versioiden sekä Siemens S7-1500 logiikan kanssa. Sovellus muodostuu muutamista ”ohjelmointitiedostoista” joiden yhteisrivimääräksi tuli yli 6000 riviä koodia. Opinnäytetyön tuloksena oli C# ohjelmointikielellä toteutettu tietokonesovellus, joka mahdollistaa useiden TIA Portal lohkojen luomisen samanaikaisesti Excel-taulukkoon syötettyjen parametrien mukaisesti. Sovelluksen avulla voidaan tulostaa tavallisia globaalidatalohkoja ja tagilistoja sekä toimeksiantajan lohkokirjastojen mukaisia instanssidatalohkoja. Sovellukseen tehdyn käyttöliittymän avulla on myös mahdollista poistaa ja lisätä kansioita sekä tiedostoja, joka varmasti helpottaa lohkojen organisointia ja ohjelman käyttämistä. Sovelluksen lisäksi tehtiin valmis Excel-taulukko, jota voidaan käyttää pohjana parametrien syöttämiseen ja joka on juuri oikean mallinen, että sovellus osaa lukea sen sarakkeita oikeaoppisesti. Käyttöönottoa varten tehtiin sovelluksen kirjalliset käyttöohjeet ja toimintaselostukset, jotka helpottavat uusia käyttäjiä perehtymään sovelluksen toimintaan.

5 Pohdinta

5.1 Työn tulosten ja tavoitteiden vertailu

Opinnäytetyön konkreettisena tavoitteena oli luoda sovellus, jolla pystytään luomaan Exceliin syötetystä datasta oikeanmallisia XML-tiedostoja, joita TIA Openness rajapinnan avulla voidaan siirtää TIA Portal projekteihin. Sovellukselta odotettiin säädin-, mittaus-, venttiili-, pumppu- ja moottorilohkojen generointia sekä tagilistojen tuontia jostain vakiomuotoisesta Excelistä. Sovelluksen tarkoituksena oli siis nopeuttaa tätä manuaalista työvaihetta, joka säästäisi työntekijän aikaa. Toimeksiantaja ei tiennyt, kuinka sovellus olisi mahdollista toteuttaa, joten tutkimustehtäväksi annettiin sovelluksen ja rajapinnan mahdollisuuksien kartoittaminen, jolla rajattiin työn laajuus sopivan kokoiseksi.

Valmis sovellus toteuttaa kaikki työlle asetetut kriteerit ja toimii toimeksiantajan haluamalla tavalla. Tutkimuskysymyksiin löydettiin monipuolisia vastauksia, minkä lisäksi halutut rajapinnan mahdollisuudet ja rajoitukset tulivat hyvin selville. Rajapintaa tutkiessa huomattiin, että tämä haluttu generaattorisovellus on mahdollista yhdistää suoraan TIA Portaliin, eikä generoitujen XML-tiedostojen importtausta tarvitsisi tehdä erikseen. Tämän havainnon avulla sovelluksella pystyttiin suorittamaan useita työvaiheita samanaikaisesti ja näin ollen nopeuttamaan datalohkojen luomista huomattavasti. Tämä oli opinnäytetyön kannalta mullistava löytö, sillä työn tarkoituksena oli etsiä datalohkojen luomiseen uusi tehokas toteutustapa ja tutkia mitä kaikkea rajapinnan avulla on mahdollista toteuttaa.

Tutkimustyön aikana löydetyt rajoitukset ja ongelmat olivat myös hyvin olennaisia. Havaintojen avulla pystyttiin rajaamaan työn laajuutta oikeanlaiseksi ja toimeksiantajalle pystyttiin kertomaan paremmin, mitä kaikkea sovelluksella pystytään tekemään ja mitä ei. Tästä johtuen toimeksiantajalla on parempi käsitys tulevaisuuden jatkokehitysmahdollisuuksista sovelluksen suhteen.

5.2 Henkilökohtaisten tulosten ja tavoitteiden vertailu

Henkilökohtaisina tavoitteina oli perehtyä tarkemmin TIA Portal-ohjelmointiympäristöön ja TIA Openness-ohjelmarajapintaan. Kyseiset aiheet ovat automaatiotekniikan alalla hyvinkin yleisiä ja

työn aikana opittua tietoja voidaan soveltaa tulevaisuudessa. Myös ohjelmointi ja sovelluskehitys olivat todella keskeisiä aiheita, joita haluttiin kehittää.

Vaikka TIA Portalia käytettiin opintojen aikana, niin silti opinnäytetyön aikana sovelluksesta löydettiin uusia ominaisuuksia ja kehityttiin sen käyttämisessä. Työn aikana päästiin vihdoin käsittelemään oikeita työelämän projekteja ja näkemään ohjelmarakenteiden sisään paremmin. Koulussa opittuja taitoja pystyttiin soveltamaan uusien oppien kanssa ja ohjelmasta tulikin työn aikana todella tuttu. TIA Openness oli puolestaan aivan vieras ja sen tutkimiseen menikin todella paljon aikaa. Onneksi aihetta pystyi tutkimaan Siemensin omilta sivuilta, josta löytyi dokumentit rajapinnan ja C# ohjelmointikielen yhteistoiminnasta sekä esimerkkisovelluksia, joiden avulla pystyttiin konkreettisesti testaamaan ohjelmien välistä kommunikointia. Nyt rajapinnan mahdollisuudet ja toimintaperiaatteet ovat todella selkeitä.

Ennen opinnäytetyön aloittamista VB ja C# ohjelmointikieliä olin käyttänyt todella vähän ja niiden osaaminen oli muutenkin hyvin alkeellisella tasolla. Työn aikana ohjelmointitaitoja haluttiin kehittää siten, että toimeksiantajan toiveiden mukainen ohjelma olisi mahdollista tehdä. Vaikka lopullisen sovelluksen toimintaperiaate on mahdollista tiivistää yhteen lauseeseen, niin ei sen ohjelmointitavat ole kuitenkaan yhtä helposti selitettävissä. Sovellusta tehdessä piti ottaa selvää esimerkiksi, kuinka pystytään luomaan yhteys Exceliin ja lukemaan valitun tiedoston soluja. Myös XML-mallitiedostojen luku ja parametrien muokkaamismahdollisuudet piti selvittää. Toimilohkojen luominen TIA Portaliin, TIA Portalin objektien luku ja hyödyntäminen sovelluksen käyttöliittymässä tuli tutkia. Sovelluksen käyttöliittymään täytyi miettiä mahdolliset lukitukset ja virheilmoitukset, jotta sovelluksesta saataisiin mahdollisimman käyttäjäystävällinen. Työn valmiiksi saaminen edellytti laajaa osaamista edellä mainituista aiheista. Kaiken tämän tutkimuksen ja kehittämisen myötä ohjelmointitaidoissa huomattiin todella suurta kehitystä.

Henkilökohtaisina tavoitteina oli myös kehittyä projektinhallinnassa. Työ osoittautuikin hyvin itsenäiseksi ja useita kuukausia kestävä tutkimustyö ja kehittäminen paransivat sinnikkyyttä ja tiedonhakutaitoja. Halutut projektityöskentelytaidot eivät suoranaisesti kehittyneet opinnäytetyön aikana, koska kaikki tutkimus ja kehittäminen tehtiin hyvin pitkälti itsenäisesti. Vastuun kantaminen tuli itsenäisen työskentelyn takia hyvin luontevaksi.

Koska työn tehtävänanto oli alkuun hyvin laaja ja aiheen rajaus täytyi tehdä tutkimuksen havaintojen perusteella, ei aikatauluttamista pystytty tekemään perinteisellä tavalla. Työn aloitusvaiheessa ei vielä tiedetty, mitä työ tulee pitämään sisällään, niin yksityiskohtaista aikatauluttamista ja työvaiheiden takarajaa oli vaikea määrittää.

5.3 Jatkokehitys

Opinnäytetyön aikana syntyi erilaisia jatkokehitysideoita, joilla sovellusta voitaisiin parantaa vielä entisestään. Datalohkoja importoidessa havaittiin, että lohkojen parametrien rakenne saattaa vaihdella TIA Portal versioiden ja eri logiikkojen välillä. Jos importoitavan XML-tiedoston ja TIA Portalin parametreissä on liikaa eroavaisuuksia, niin lohkon luominen saattaa epäonnistua ja antaa käyttäjälle virheilmoituksen. Sovellus ei ole siis yleispätevä jokaisen olemassa olevan version ja logiikan kanssa. Yhtenä jatkokehitysideana olisikin käydä jokainen mahdollinen logiikka ja ohjelmaversio läpi sekä luoda näille omat XML mallipohjat tai vaihtoehtoisesti keksiä jokin toinen ratkaistu universaaliin yhteensopivuuteen.

Jotta uudet toimilohkot toimivat TIA Portal ohjelmassa, niin niitä täytyy kutsua ohjelman sisällä. Toimeksiantajan projekteissa kutsuminen tapahtuu yhdellä funktiolohkolla, joka sisältää jokaisen suoritettavan toimilohkon. Kuitenkin jos nykyisen sovelluksen avulla luodaan uusia lohkoja, joita halutaan kutsua niin ne eivät kuitenkaan päivity tähän kutsuvaan funktiolohkoon. Kyseinen funktiolohko täytyy siis päivittää aina manuaalisesti, tai vaihtoehtoisesti jatkokehitysideana voitaisi suorittaa funktiolohkon generointi ja päivitys automaattisesti.

Tämänhetkisen sovelluksen avulla on mahdollista generoida TIA Portaliin lohkoja, mutta jos niihin halutaan jälkikäteen tehdä muutoksia, niin ne täytyy joko poistaa ja generoida uudelleen tai muokata manuaalisesti TIA Portalissa. Sovellus ei mahdollista uusien muuttujien tai tagien lisäämistä olemassa oleviin globaalidatalohkoihin ja tagilistoihin. Näitä pystytään luomaan vain, jos lohkoa ollaan luomassa samanaikaisesti. Sovellukseen olisikin varmasti mahdollista kehittää menetelmä, jolla pystyttäisi muokkaamaan globaalidatalohkoja ja tagilistoja ohjelmasta käsin.

Työn aikana luotu sovellus suunniteltiin vain globaali- ja instanssidatalohkojen ja tagilistojen luomiseen. Rajapinnan avulla olisi kuitenkin mahdollista importoida muunlaisia lohkoja, kuten esimerkiksi funktio- ja toimilohkoja sekä WINCC valvomokuvia. Sovelluksessa käytettyjä menetelmiä ja

toimintaperiaatteita voisi varmasti hyödyntää näiden edellä mainittujen lohkojen importoimisessa. Käytännössä sovelluksesta olisi mahdollista tehdä niinkin pitkälle kehittynyt, että kaikki lohkot ja muuttujat voitaisi tehdä sovelluksen avulla kätevämmin kuin TIA Portalissa.

Lähteet

A tour of the C# language. 2021. C# ohjelmointikielen esittely Microsoftin omalla sivulla. Viitattu 19.3.2021. https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.

Arshad, A. 2019. Importing and Processing data from XML files into SQL Server tables. Ohje XML-tiedostojen importoimiseen SQL-taulukoiksi. Viitattu 21.3.2021 https://www.mssqltips.com/sql-servertip/2899/importing-and-processing-data-from-xml-files-into-sql-server-tables/.

Automaatio ja automaatiojärjestelmät. N.d. Automaatiojärjestelmien esittely. Viitattu 26.1.2021. https://valmistajat.fi/menetelmat/elektroniikka/automaatio-ja-automaatiojarjestelmat/.

C# Introduction. N.d. C# ohjelmointikielen esittely. Viitattu 17.3.2021. https://www.w3schools.com/cs/cs_intro.asp.

CTS Engtec. 2021. CTS Engtec yritysesittely niiden omilla sivuilla. Viitattu 26.3.2021. https://www.ctse.fi/yritys.

CTS Engtec_jyväskylä_automaatio. 2019. CTS Engtec PowerPoint-esittely Jyväskylän toimipisteestä. Sisäinen materiaali. Viitattu 11.2.2021.

Dixon, M. 2018. WHAT ARE THE MOST POPULAR PLC PROGRAMMING LANGUAGES? Yleiskatsaus PLC ohjelmointikielistä. Viitattu 21.2.2021. https://realpars.com/plc-programming-languages/.

Excel code generator for TIA Portal Openness. 2019. Esittely Siemensin valmistamasta koodigeneraattorista TIA Openessiin. Viitattu 2.3.2021. https://support.industry.siemens.com/cs/document/109770550/excel-code-generator-for-tia-portal-openness?dti=0&lc=en-WW.

Gates, S. 2018. A Beginner's PLC Overview, Part 3 of 4: PLC Inputs and Outputs (I/O). Aloittelijan PLC I/O yleiskatsaus. Automation. Viitattu 18.2.2021. https://www.automation.com/en-us/articles/2018/a-beginners-plc-overview-part-3-of-4-plc-inputs-an#authorInfo.

Jones, C. 2019. The Pros And Cons Of Automation For Business. Automaation hyödyt ja haitat. Viitattu 26.1.2021. https://www.theearthawards.org/the-pros-and-cons-of-automation-for-business/.

Lesson 1 : Introduction to Visual Basic. 2020. Visual Basic-ohjelmointikielen esittely. Viitattu 24.3.2022. https://www.vbtutor.net/lesson1.html.

Organization Blocks and Program Structure. N.d. Katsaus organisointilohkoihin. Viitattu 28.2.2021. http://www.plccenter.cn/Siemens_Step7/Organisationsbausteine_und_Programmstruktur.htm.

PLC programming. N.d. TIA Portal PLC-ohjelmoinnin esittely Siemensin omilla sivuilla. Viitattu 17.1.2021. https://mall.industry.siemens.com/mall/en/WW/Catalog/Products/10090015.

Prosessitekniikan perusta Automaatiotekniikka. N.d. Artikkeli Oulun yliopiston sivuilta. Viitattu 27.1.2021. https://www.oulu.fi/sites/default/files/content/PTperusta_automaatio.pdf.

S7-GRAPH V5.3 for S7-300/400 Programming Sequential Control Systems. 2004. S7-GRAPH ohjelmointikielen ja sekvenssiohjelmoinnin esittely Siemensin sivuilla. Viitattu 27.3.2021. https://cache.industry.siemens.com/dl/files/630/1137630/att_28560/v1/Graph7_e.pdf.

SIMATIC STEP 7 Basic/Professional V15.1 and SIMATIC WinCC V15.1. 2018. SIMATIC STEP 7 ja WinCC ohjelmointi- ja käyttöohje Siemensin omilla sivuilla. Viitattu 23.2.2021. https://support.industry.siemens.com/cs/document/109755202/simatic-step-7-basic-professional-v15-1-and-simatic-wincc-v15-1?dti=0&lc=en-DE.

Smith, P. 2019. Advantages & Disadvantages of Siemens' TIA Openness. TIA Openness rajapinnan hyödyt ja haitat. Viitattu 8.3.2021. https://www.dmcinfo.com/latest-thinking/blog/id/9877/advantages-disadvantages-of-siemens-tia-openness.

Software in TIA Portal. N.d. Kuvaus TIA Portaalista Siemensin sivuilta. Viitattu 15.2.2021. https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal/software.html.

TIA Portal Openness. 2018. Esitys TIA Portal Opennessin toiminnasta. Viitattu 5.3.2021. https://assets.new.siemens.com/siemens/assets/api/uuid:0fdd52a4-c384-4e55-a89d-ba9181d17fc7/tia-openness.pdf.

TIA Portal Openness: Introduction and Demo Application. 2020. TIA Portal Openness esittely Siemensin sivuilla. Viitattu 7.3.2021. https://support.industry.siemens.com/cs/document/108716692/tia-portal-openness%3A-introduction-and-demo-application?dti=0&lc=en-SE.

TIA-PORTAL. N.d. Kuvaus TIA Portaalista. Viitattu 13.2.2021. https://www.sitrain-learning.siemens.com/FI/en/rw94408/TIA-PORTAL.

Top IDE index. 2021. Luettelo eniten käytetyistä ohjelmankehitysympäristöistä. Viitattu 26.3.2021. https://pypl.github.io/IDE.html.

Visual Studio 2019. N.d. Visual Studio-ohjelmankehitysympäristön esittely Microsoftin sivuilla. Viitattu 16.3.2021. https://visualstudio.microsoft.com/vs/.

VB.NET and C# Comparison. 2016. VBNET ja C# ohjelmointikielien komentojen vertailu. Viitattu 24.3.2021. https://sites.harding.edu/fmccown/vbnet_csharp_comparison.html.

Welcome to the Visual Studio IDE. 2019. Visual Studion esittely Microsoftin sivuilla. Viitattu 15.3.2021. https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019.

What is the difference between an instance data block and a global data block and how does a CALL call influence the DB register? 2011. Instanssidatalohkojen ja globaalidatalohkojen erot Siemensin sivuilla. Viitattu 11.5.2021. https://support.industry.siemens.com/cs/document/15360455/what-is-the-difference-between-an-instance-data-block-and-a-global-data-block-and-how-does-a-call-call-influence-the-db-register-?dti=0&lc=en-WW.

What is an XML file and How Do I Open One? 2021. XML tiedostotyypin esittely. Viitattu 20.3.2021. https://www.indeed.com/career-advice/career-development/xml-file.

What is VB.Net? Introduction, History, Features, Advantages, Disadvantages. N.d. VB.net ohjelmointikielen esittely. Viitattu 20.3.2021. https://www.guru99.com/vb-net-introduction-features.html.

XML:n perusteet. N.d. XML tiedostotyypin perusteet Microsoftin omalla sivulla. Viitattu 25.3.2021. https://support.microsoft.com/fi-fi/topic/xml-n-perusteet-a87d234d-4c2e-4409-9cbc-45e4eb857d44?ui=fi-fi&rs=fi-fi&ad=fi.