İSTANBUL TECHNICAL UNIVERSITY

Faculty of Computer Science and Informatics

Tia Portal Web Application Using Openness

INTERNSHIP PROGRAM REPORT

Elif KÖSELER

150170701

1 July-29 July 2019

1. INFORMATION ABOUT THE INSTITUTION

Siemens is an international technology company founded by Werner von Siemens and Johann Georg Halske in Berlin in 1847, Siemens is operating in 200 countries worldwide in three main sector: industry, energy and health. (Figure-1) 9 years after the establishment came to Turkey. The company established the first telegraph system in the Ottoman Empire in 1856 and the first telephone line in 1881. Republic of Turkey also met with several new products thanks to Siemens such as steam machine, dynamo, X-ray device, hydropower plant and high speed train. [3] (Özdemir, 2017)

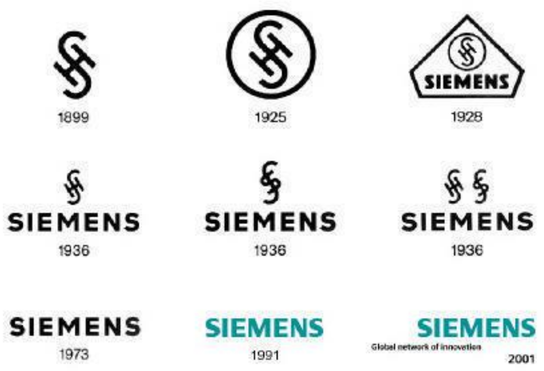


Figure 1 - Evolution of Siemens logo [4] (Siemens Logo - Design and History of Siemens Logo., 2012)

The head office of Siemens in Turkey is located in Kartal, Istanbul, and they have more than 2000 employees.

2. INTRODUCTION

Siemens is a huge global company so it has a lot of departments in Kartal such as R&D, Human Resources, IT, Internet of Things (Corporate Technologies), etc. In IoT department, they develop TIA Portal (Totally Integrated Automation Portal) Program with divided small PLC groups. For instance, some team names: Simocode, As-I, Safety.

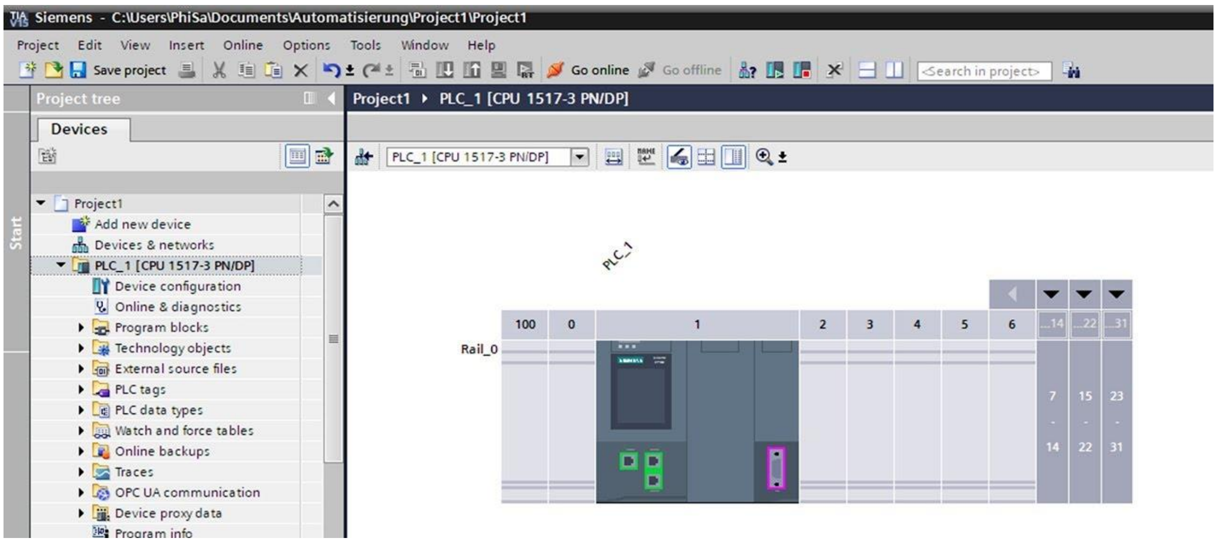
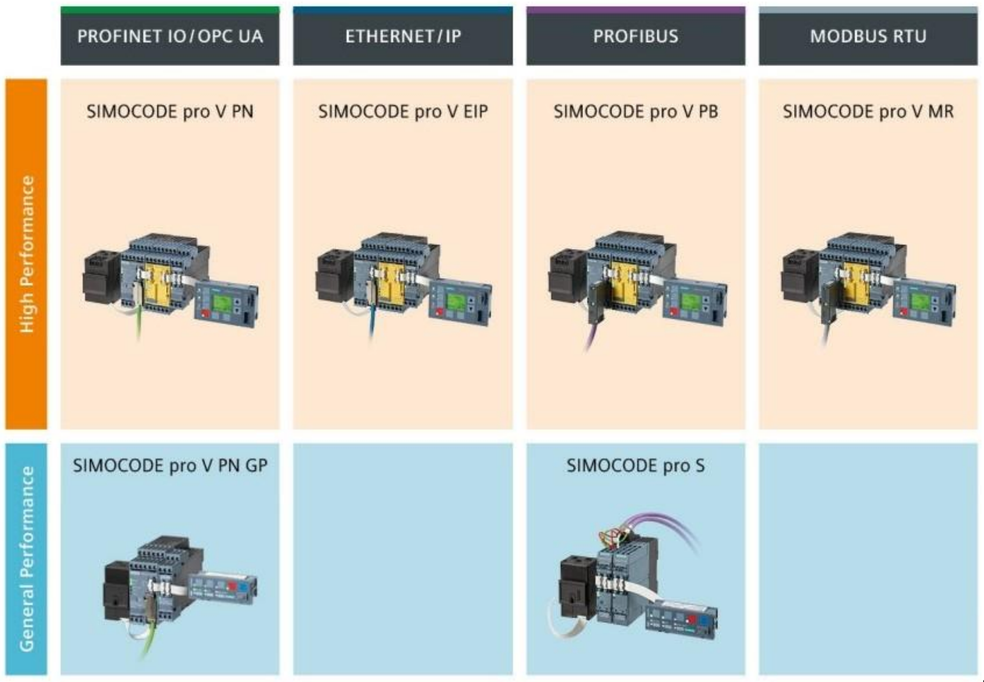


Figure 2 - Sample screenshot from TIA Portal Framework [1] (How to get SIEMENS TIA Portal V15!, 2018)

TIA Portal is a Siemens engineering framework that is using for all automation task in manufacturing, customers can use controllers, I/O, HMI, drivers, motion controllers and motor management systems in one single environment. (Figure-2). [8] (SiemensAG, n.d.)



,

Figure 3 - Types of Simocode

Beginning of my internship, I researched and gathered information about TIA Portal, because I had no idea at the first. I was doing internship in Simocode Team under IoT departmen. So, after the research, my mentor asked me to learn what simocode is, what it does. Simocode is a industrial device from Sirius family whose full name is Sirius Motor Management and Control Device. It is flexible, modular motor management system for motors with constant speeds in the low-voltage performance range. Simocode supports connection between integrated circuit and motor feeder and also open connection with Profibus DP, Profinet / DPC UA, Modbus RTU, Ethernet / IP. There are several types of Simocode: Simocode Pro S, Simocode Pro V, Simocode Pro V PN and also safety versions. (Figure-3) [5] (SIMOCODE pro: Motor Management and Control Devices.)

After completing the research of Simocode, I learnt informations about connections between devices that Profibus (Process Field Bus) is a standard for fieldbus communication and Profinet (Process Field Network) is an internet communication over industrial Ethernet. Profinet has larger bandwidth than Profibus, thus it is faster than Profibus and can send bigger size messages. In addition, I also gathered information about other connection types and devices to include to the project such as AS-Interface, AS-I devices, MPI (Message Passing Interface) and Simatic devices. My mentor gave a project for me and my internship mate to improve ourselves. However, the project improved me at team study. Through project, I became better at C# and .NET Framework with Visual Studio.

3. DESCRIPTION AND ANALYSIS OF THE INTERNSHIP PROJECT

The aim of the project is creating an application that can control TIA Portal with help of the Openness on web. A web program, a socket program which contains two sockets and an Openness program are used in project. Web application sends commands to medium socket program after that it sends received commands to Openness program. Openness program parses commands and applies on to TIA Portal and then it replies to medium socket program in connection with web application. As I mentioned before, I did not do project on my own. We did the project with my friend Emre from internship. And also the project given by my mentor and his mentor. So, I experienced to working with several people.

At the first week, I researched TIA Portal, Simocode and also Openness. I learnt information about Openness that using the Openness API, we can access almost all tools of TIA Portal such as controlling project data, PLC data and HMI data. In sum, Openness an API Interface for WinCC and STEP7 in TIA Portal to automate engineering tasks without entering TIA Portal Framework, thus everyone can create its own application without waiting to TIA Portal opening time. [6] (TIA Portal Openness: Introduction and Demo Application., 2019). At beginning of the second week, we made a meeting with his mentor to do task sharing and also he drew a basic draft for project. As a result of meeting, I was going to do Openness part, connection with TIA Portal Program and connection with socket program. And Emre was going to do Web Application of the project, socket program and connection between socket and Web Application. (Figure-4)

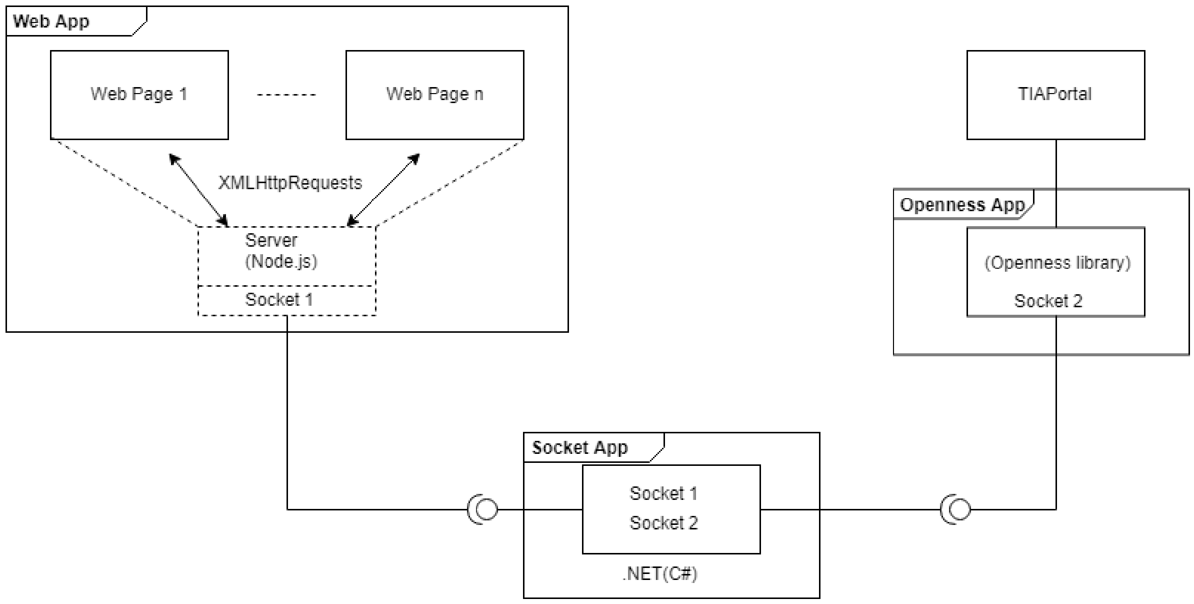


Figure 4 - Project Structure

3.1 Programming Languages

Siemens uses C# programming language along with .NET that are used for TIA Portal Program development. C# is developed by Microsoft. It is quite similar to C ++ and Java, but C# has similarities and differences. For example, unlike C++, has a 100% object orientation technique. Unlike Java, the pointer can be used in C#. Thus, it can work in harmony with older software components.

There are some advantages of C# programming languages, the most important of these is the program can work on different operating systems. Unlike C ++ or Visual Basic, C# codes are not compiled directly into machine code. Firstly, IL is compiled to a code. The file of this first compiled code is called assembly and its extension is exe. When this file is attempted to run, the .Net Framework is activated and converts the IL code into machine code (Figure-5) so that the computer can now understand the code. This is why we must have .Net Framework installed on that computer, because .Net Framework IL translates the code that the computer can understand. Therefore, the first time we run the program we created, our program run slow, but it will accelerate in later.

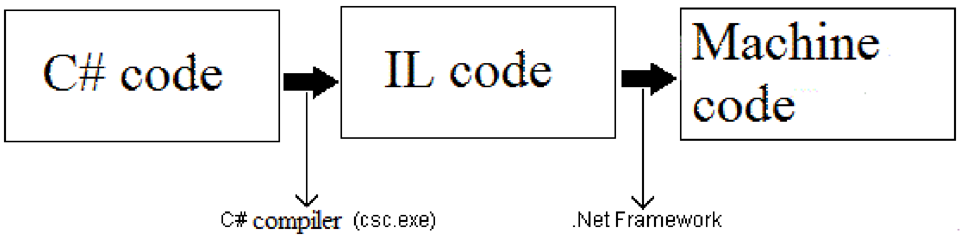


Figure 5 - C# and .Net Framework

In addition to C# and .Net Framework, as I mentioned before I used Openness API.

We have made a project with Openness that they offer to customers during the Siemens internship process. Openness is the API interface for WinCC and STEP 7 in the TIA Portal. It automates engineering tasks and integrates into your own development environment with TIA Portal. [2] (Jon Stenerson, 2015). We can use Openness to create applications such as a code generator for HMI images and PLC software blocks. This is useful as manual adjustments in projects involve a high susceptibility to errors. Moreover, this automation allows you to save execution time, enabling you to work more efficiently.

In my first week, after learning what TIA portal is and what it does and its features, I tried to understand the Openness library. Then I opened a project with using Openness library. It was not difficult to understand the Openness library with a little bit of C# knowledge and object-oriented programming logic, which I quickly understood even though I had never used C# before. (Figure-6) [6] (TIA Portal Openness: Introduction and Demo Application., 2019).

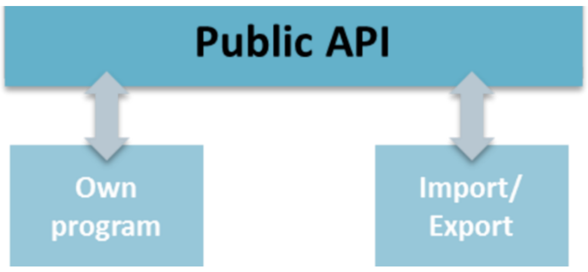
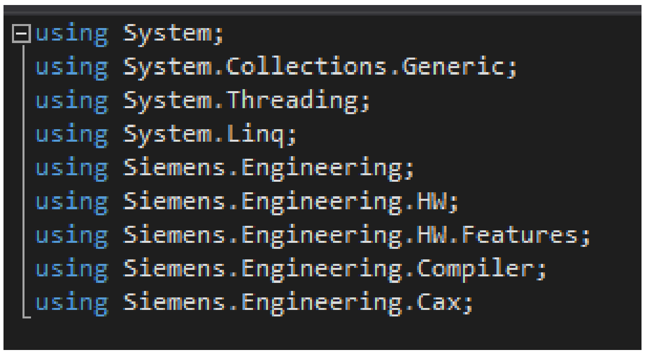


Figure 6 - Openness Schema

3.2 Development Environments

As I mentioned before when doing the project, I needed the TIA Portal to check the accuracy of the code using the user interface and to add new features. I discovered its features easier when using TIA Portal with user interface. I used TIA Portal in version 16, however you can change version with changing just a row.



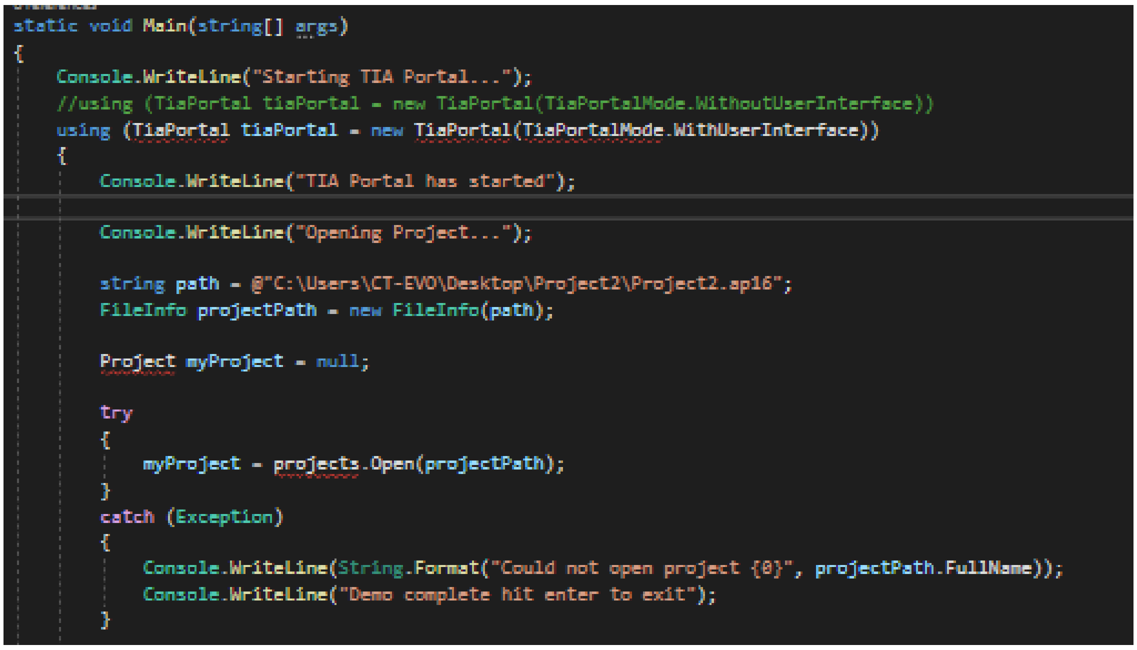
```
using System;
using System.Collections.Generic;
using System.Threading;
using System.Linq;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Cax;
```

Figure 7 - Sample libraries that I used in Visual Studio

I also used Visual Studio 2019 to develop my project. We all know Visual Studio is a popular IDE developed by Microsoft. With Visual Studio, we can develop software, edit and compile the code we have written, and easily convert it into an application. Thanks to its many features (code completion, debugging, compilers, graphic designers, database connections, etc.), it is a program that simplifies the software development process. Specially to developing software with .NET framework, it is a must. Adding *Siemens.Engineering* libraries is also very easy when using Visual Studio and other libraries such as *System.Linq*. (Figure-7)

3.3 Details

The project consists of 3 main parts; Openness app, Web app and Socket app. I created the part of Openness in the project. At the first, tasks of Openness app are opening a TIA Portal project which is created previously, adding hardware devices, editing properties of these devices, connecting devices each other using correct connection type and compiling the whole project. [7] (TIA Portal Openness:Introduction and Demo Application, 2019). Unfortunately, the first tasks were harder than I expected especially the connection and compile parts. Thus, actually I did add hardware devices part at the first of course after the opening project part. In addition, we created communication between my internship colleague and I by creating a syntax, but I will explain the syntax below as we create and integrate it later in the code.



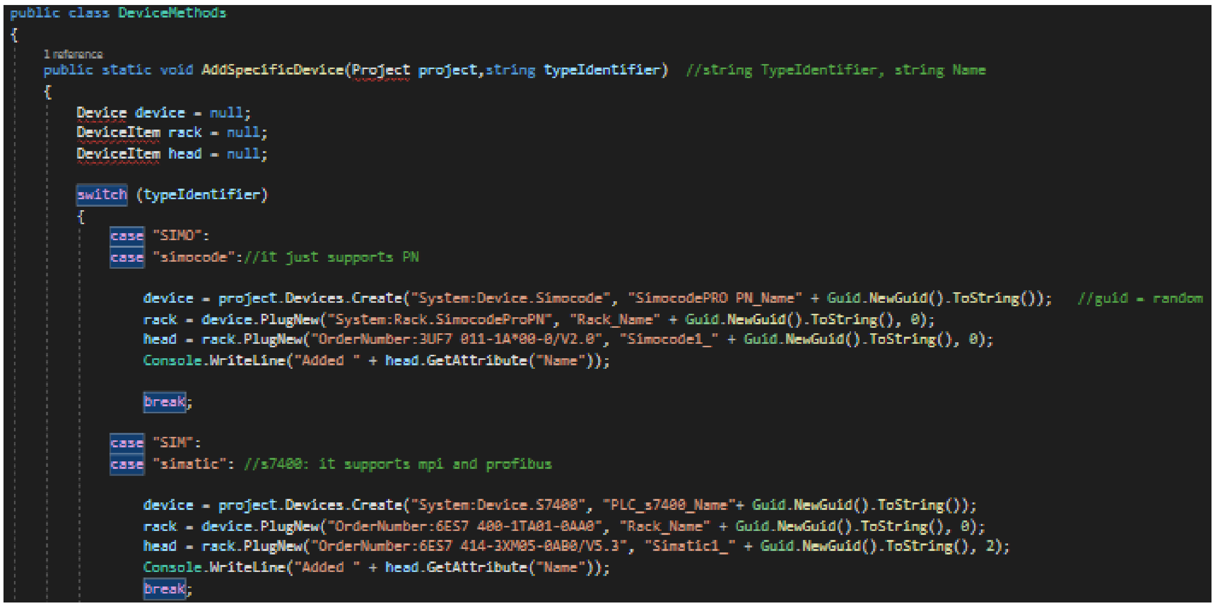
```
static void Main(string[] args)
{
    Console.WriteLine("Starting TIA Portal...");
    //using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface))
    using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
    {
        Console.WriteLine("TIA Portal has started");

        Console.WriteLine("Opening Project...");

        string path = @"C:\Users\CT-EVO\Desktop\Project2\Project2.ap16";
        FileInfo projectPath = new FileInfo(path);

        Project myProject = null;

        try
        {
            myProject = projects.Open(projectPath);
        }
        catch (Exception)
        {
            Console.WriteLine(String.Format("Could not open project {0}", projectPath.FullName));
            Console.WriteLine("Demo complete hit enter to exit");
        }
```

Figure 8 - Related codes of opening a project

We need to start the TIA Portal before opening the project. We can start the TIA Portal with or without the user interface, we specify this feature as a mode inside the code. There are some advantages and disadvantages of using the user interface or not. When using the user interface, should be reminded that the project is slow while you can follow it very easily.

We also can use it quickly and efficiently without using the user interface. After starting TIA Portal, I created a project and assigned to null, then opened the project in specified path in main method. The project assigned to open one. I also defined an error handling mechanism to avoid errors when project could not open. (Figure-8)

Afterward, *RunServer* method is called to run server to take input data and send to the proper function.



```
public class DeviceMethods
{
    1 reference
    public static void AddSpecificDevice(Project project,string typeIdentifier)  //string TypeIdentifier, string Name
    {
        Device device = null;
        DeviceItem rack = null;
        DeviceItem head = null;

        switch (typeIdentifier)
        {
            case "SIMO":
            case "simocode"://it just supports PN

                device = project.Devices.Create("System:Device.Simocode", "SimocodePRO PN_Name" + Guid.NewGuid().ToString());   //guid = random
                rack = device.PlugNew("System:Rack.SimocodeProPN", "Rack_Name" + Guid.NewGuid().ToString(), 0);
                head = rack.PlugNew("OrderNumber:3UF7 011-1A*00-0/V2.0", "Simocode1_" + Guid.NewGuid().ToString(), 0);
                Console.WriteLine("Added " + head.GetAttribute("Name"));

                break;

            case "SIM":
            case "simatic": //s7400: it supports mpi and profibus

                device = project.Devices.Create("System:Device.S7400", "PLC_s7400_Name"+ Guid.NewGuid().ToString());
                rack = device.PlugNew("OrderNumber:6ES7 400-1TA01-0AA0", "Rack_Name" + Guid.NewGuid().ToString(), 0);
                head = rack.PlugNew("OrderNumber:6ES7 414-3XM05-0AB0/V5.3", "Simatic1_" + Guid.NewGuid().ToString(), 2);
                Console.WriteLine("Added " + head.GetAttribute("Name"));
                break;
```

Figure 9 - A part of AddSpecificDevice method

In the first step of adding devices, I examined the parts of the device. Considering the device as an object, a device consists of 3 main parts which are device, rack and head. In fact that data type of rack and head is the same which is DeviceItem. To put it briefly, there is a rack in the device, and there are device items in the rack, including the head. I created two methods to adding devices first one is *AddSpecificDevice(Project, string)* that adds specific device according to coming type identifier and the second one is *AddMultipleDevices(Project, int)* that adds multiple devices to test program according to coming number. As a matter of fact, the project is not using *AddMultipleDevices* method, it is used for just testing. In *AddSpecificDevice* method, firstly I defined device, rack and head then I inserted device to project, rack to device and head to rack. And also, I printed out insertion on console to check. In insertion part, I added devices according to type identifier, in detail according to system names and order numbers of device. I used *Guid* when naming in order to avoid receiving errors from the name conflict. Eventually, I added 2 Simocode devices, 2 Simatic devices and 1 As-I device with this function. We used “add” command to add devices in our syntax. (Figure-9)

After completing the add devices section, I created a method to edit the properties of the devices. For this method, we used the "change" command in our syntax. In *ChangeProperties(Project, int, string, string, string)* method, you can edit device name, author and comment. Before the coming parameters of function, input which is command breaks into tokens then it comes to the method in *FunctionSelector* method. First, I took index to find exact device. After I took name, comment and author to change. If name, comment or author comes “x”, there is no change on proper property according to our syntax. I got device before checking strings with *GetDevice* method. Then according to device name, I got device’s head, because there was a difference between PLCs’ head and other devices’ head. For the avoiding error, I defined two methods *GetPlcHead* and *GetHead.* I checked names with if statement and send to related methods. Selecting names of the devices in adding device part here was very useful for me. I changed name, author and comment with *GetAttribute* and *SetAttribute* methods easily.

To continue with connection part, after compilation part, it was the part that forced me the most. There are 4 connection types in the project which are PN, PB, MPI and As-I. For this feature we used “con” command with indexes of source and target device. Because connection supports between two devices. At the first, I created a *selectConnection* method to select connection type that consists indexes of source and target device and connection type identifier parameters. According to connection type identifier, it sends to related functions which are *connectToPN*, *connectToPB*, *connectToMPI* and *connectToASI.*

For the connection between two devices, there are many pieces to use. As part of the head the interface and as the part of interface the node join to the hierarchy I mentioned earlier. In *connectToPN* method, foremost I created a subnet composition that contains subnets, then I created new subnet to add existing subnet composition with specifying special system name and name by using Guid. After that, I created temporary target and source devices for connection, I got heads of the devices depending on whether PLC or not. Then I created an interface whose data type is DeviceItem and I found the interface in all device items by checking to see if it contains the word "PROFINET" or “interface”. Later, I pulled node composition from the interface by using *GetService<NetworkInterface>* method. I connected first node of node composition to the subnet. I made the same process separately for both devices, nodes connected to the same subnet, therefore connection was provided.

In the other methods, *connecToPN* method is often the same only when creating a subnet system names and words used when searching for interfaces are different. For instance, in *connectToASI* method, look for the “AS-Interface” word when searching interface device item.

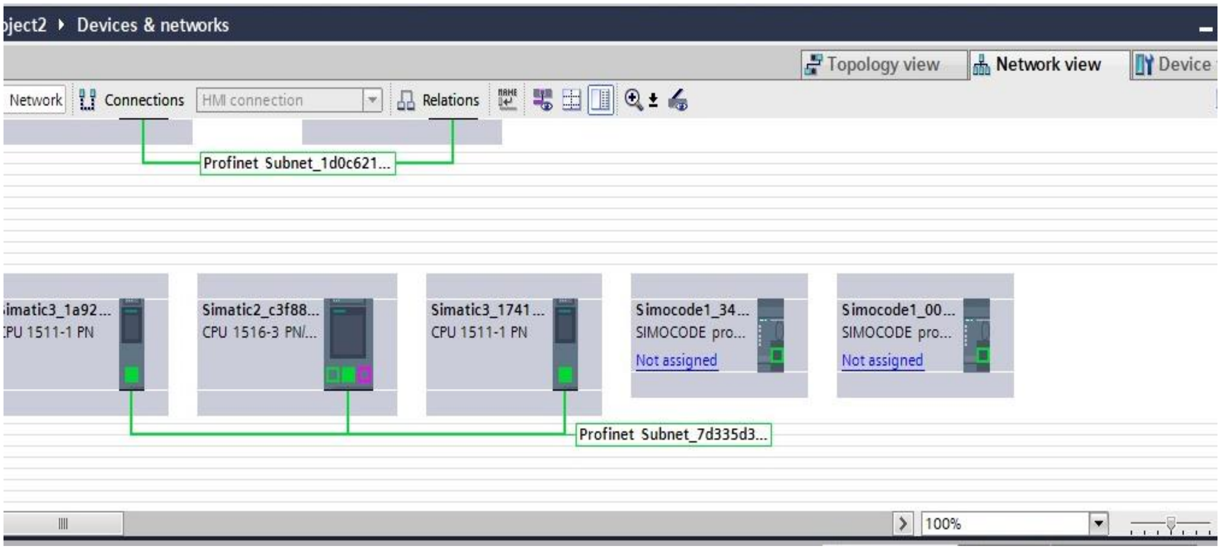
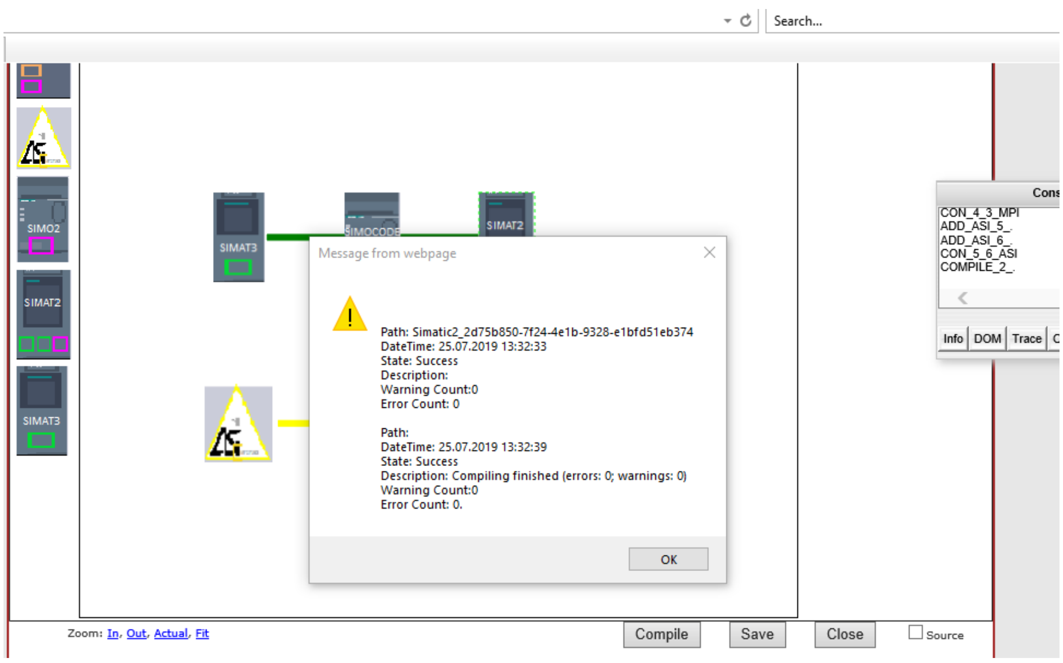


Figure 10 - Example of multiple connection on TIA

On the other hand, to provide connection between three or more devices, I needed to another method called by *multiConnection.* In this method, the code follows the path: project > device > device item > interface > nodes > node, then checks node is connected to a subnet or not, then if is connected source node connects subnet of target device. The connection provided between three or more devices in same type in the event. We used “multicon” command in selection method. (Figure-10)

Furthermore, the most challenging part for me is that compilation of the project. Compilation includes two type: hardware and software. We used “compile” command to compile method in our syntax. Unfortunately, I did not have time to compile the hardware part because I spent a lot of time on the software part. Anyway, in *Compile* method, firstly I got device from all devices by using index of device. Then, I got PLC Software of the device with *GetPlcSoftware* method. In this method, I used *SofwareContainer* type and *GetService* method to obtain *PlcSoftware*, it returns *PlcSoftware* or *null*. After, getting PLC Software, it has also two parts: PlcSoftware and CodeBlock, so I created two different methods to compile these parts which are *CompileCodeBlock* and *CompilePlcSoftware*. These methods return string because we wanted to see this message on our web application site. Openness has a compiler result data type that is *CompilerResult*.

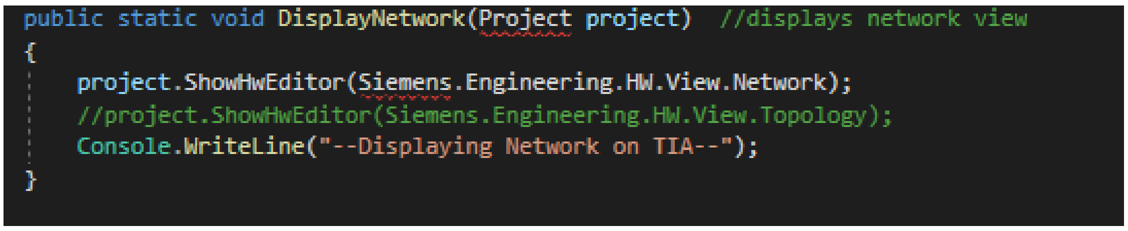


```
Path: Simatic2_2d75b850-7f24-4e1b-9328-e1bfd51eb374
DateTime: 25.07.2019 13:32:33
State: Success
Description:
Warning Count:0
Error Count: 0

Path:
DateTime: 25.07.2019 13:32:39
State: Success
Description: Compiling finished (errors: 0; warnings: 0)
Warning Count:0
Error Count: 0.
```

```
CON_4_3_MPI
ADD_ASI_5_.
ADD_ASI_6_.
CON_5_6_ASI
COMPILE_2_.
```

Figure 11 - Result of the compilation on web

I collected all results on this type, then I created another function to write this result called by *WriteCompilerResults*. This function collects State, WarningCount, ErrorCount, Path, DateTime and Description of *CompilerResult* and then it returns in string type to compiler functions. (Figure-11)



```
public static void DisplayNetwork(Project project)  //displays network view
{
    project.ShowHwEditor(Siemens.Engineering.HW.View.Network);
    //project.ShowHwEditor(Siemens.Engineering.HW.View.Topology);
    Console.WriteLine("--Displaying Network on TIA--");
}
```

Figure 12 - DisplayNetwork method

At the last, I created three methods which are *Close*, *Save* and *DisplayNetwork*. They work with "close", "save" and "show" commands respectively. For these methods Openness has prepared methods such as *Save*, *Close* and *ShowHwEditor*. It was simple and clear, so I did not spend a lot of time creating these functions. Just to add, we can display the project as topology or network with helping of *ShowHwEditor* method. And I used this method often to check the code. (Figure-12)

Last but not least, I created *FunctionSelector* method to select proper function for commands coming from server with using switch case statement. And I needed to remind that input data (commands) is split to tokens in *RunServer* method, then is sent to *FunctionSelector*.

4. CONCLUSION

As a consequence of my internship, I learn a lot of things that actually I did not expect in this short term. Firstly, I did my internship project with my internship colleague, therefore I learnt team work process and also had a friend. During the internship project, I worked with C# and .Net framework, it helped me to develop in this programming languages. Also I worked with Openness API, actually before the internship I had not even heard Openness and TIA Portal. However, now I believe in myself on Openness and controlling API. It was a good experience that I had not worked with API before. I tried to observe corporate life in Siemens, I participated to meetings with many employees of Siemens. I had a good time with the Simocode team and took advices about lessons and career life. As a result, Siemens has given me much more than I expected, thanks to my object-oriented programming base, learning and practicing C# in a short time fulfilled my self-confidence. Presenting our project was also a good experience to the IoT department. I think that I have developed my English with international meetings in Siemens.

5. REFERENCES

[1] *How to get SIEMENS TIA Portal V15!* (2018, October 10). Retrieved from Youtube: https://www.youtube.com/watch?v=xE25v2Y2_t4.

[2] Jon Stenerson, D. D. (2015). *Siemens Step 7 (TIA Portal) programming : a practical approach.* Lexington, KY.

[3] Özdemir, B. T. (2017). About. In *History of Siemens from Empire to Republic : in the light of documents from state archives.* İstanbul.

[4] *Siemens Logo - Design and History of Siemens Logo.* (2012). Retrieved from https://www.famouslogos.us/siemens-logo/.

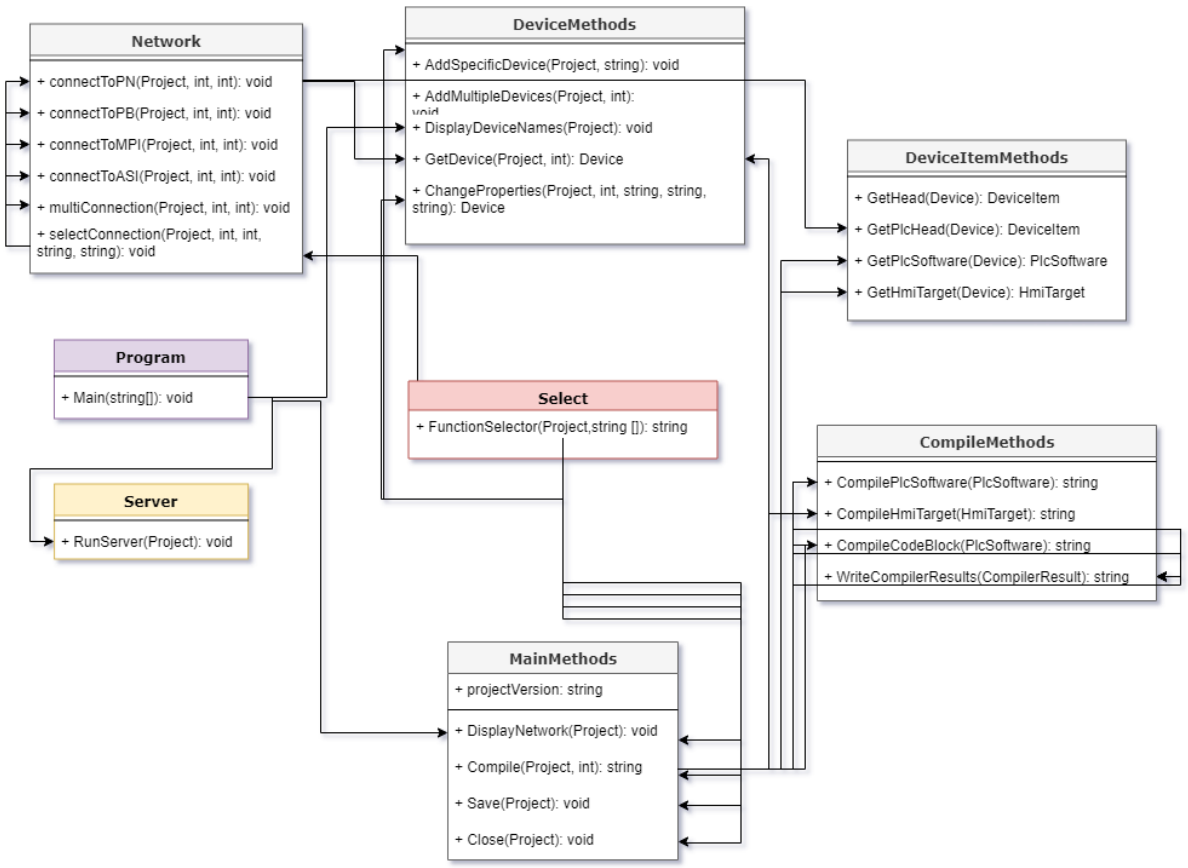
[5] *SIMOCODE pro: Motor Management and Control Devices.* (n.d.). Retrieved 2019, from https://new.siemens.com/global/en/products/automation/industrial-controls/sirius/sirius-monitor/simocode.html.

[6] *TIA Portal Openness: Introduction and Demo Application.* (2019, March 5). Retrieved from https://support.industry.siemens.com/cs/document/108716692/tia-portal-openness:-introduction-and-demo-application?dti=0&lc=en-WW.

[7] TIA Portal Openness:Introduction and Demo Application. (2019, May).

[8] *Totally Integrated Automation Portal.* (n.d.). Retrieved from https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html.

6. APPENDIX



Network
+ connectToPN(Project, int, int): void
+ connectToPB(Project, int, int): void
+ connectToMPI(Project, int, int): void
+ connectToASI(Project, int, int): void
+ multiConnection(Project, int, int): void
+ selectConnection(Project, int, int, string, string): void
DeviceMethods
+ AddSpecificDevice(Project, string): void
+ AddMultipleDevices(Project, int): void
+ DisplayDeviceNames(Project): void
+ GetDevice(Project, int): Device
+ ChangeProperties(Project, int, string, string, string): Device
DeviceItemMethods
+ GetHead(Device): DeviceItem
+ GetPlcHead(Device): DeviceItem
+ GetPlcSoftware(Device): PlcSoftware
+ GetHmiTarget(Device): HmiTarget
Program
+ Main(string[]): void
Select
+ FunctionSelector(Project,string []): string
Server
+ RunServer(Project): void
CompileMethods
+ CompilePlcSoftware(PlcSoftware): string
+ CompileHmiTarget(HmiTarget): string
+ CompileCodeBlock(PlcSoftware): string
+ WriteCompilerResults(CompilerResult): string
MainMethods
+ projectVersion: string
+ DisplayNetwork(Project): void
+ Compile(Project, int): string
+ Save(Project): void
+ Close(Project): void

Figure 13 - UML diagram of Openness App