



SETTING UP AN INDUSTRIAL CONTROL SYSTEMS LABORATORY

Graduation Project Internship
SecurityMatters BV

Haryanto Natalius Liuwan
hn.liuwan@student.fontys.nl

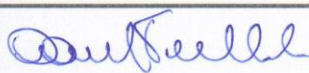
GRADUATION / INTERNSHIP REPORT
FONTYS UNIVERSITY OF APPLIED SCIENCES
HBO-ICT: English Stream

Data student:	
Family name , initials:	Liuwan, H.N.
Student number:	2331039
project period: (from – till)	February – May 2014
Data company:	
Name company/institution:	SecurityMatters BV
Department:	-
Address:	De Lismortel 31, 5612 AR Eindhoven
Company tutor:	
Family name, initials:	Trivellato, D.
Position:	Project Manager
University tutor:	
Family name , initials:	Schellekens, C.
Final report:	
Title:	Setting Up ICS Simulation Laboratory
Date:	1 st February 2014

Approved and signed by the company tutor: DANIEL TRIVELLATO

Date: 06.06.2014

Signature:



Preface

This document is the final report of “Industrial Control Systems Simulation Laboratory Development” graduation project internship. The internship was carried out at SecurityMatters from 1st February 2014 until 31st May 2014 by Haryanto Natalius Liuwan, a double degree Information and Communication Technology student at Fontys Hogescholen.

The reasons why I chose this graduation project are because I like to have interaction with hardwares, and I already learnt some of networking module at my home university in Indonesia.

In this moment I would like to say thank you to Daniel Trivellato as my company mentor, Casper Schellekens as my university mentor, and all colleagues of SecurityMatters, the company where I accomplished this graduation project internship.

I hope you will enjoy reading the report, Thank you.

Eindhoven, May 2014

Haryanto Natalius Liuwan

Table of Contents

Preface	2
Table of Contents.....	3
Summary	1
Glossary	2
Chapter 1: Introduction	4
1.1. Introduction	4
Chapter 2: The Company	6
2.1. History	6
2.2. Mission, Vision, and Values	6
Chapter 3: The Assignment	8
3.1 Background	8
3.2. Assignment Overview	8
3.3. Project Deliverables	8
3.4. Goals.....	9
3.5. Changes During the Project	9
3.5.1. Change of Phase Order.....	9
3.5.2. Programming Language	9
3.5.3. Project Focus	10
Chapter 4: Phasing and Methodology	11
4.1. Initiation and Research Phase	11
4.2. Learning Phase	12
4.3. Development Phase	12
4.3.1. Defining Requirements	13
4.3.2. Analysis and Design	13
4.3.3. Implementation.....	13
4.3.4. Testing	13
4.3.5. Evaluation.....	13
4.4. Rebuilding Laboratory Phase	13
Chapter 5: Learning Phase.....	14
5.1. Objective	14
5.2. Topics	14

5.2.1. ICS/SCADA System Components and Protocols.....	14
5.2.2. Industrial Process	14
5.2.3. Electricity Distribution	14
5.2.4. Protocols for each Application.....	15
5.3. Result	15
5.3.1. Project Plan.....	15
5.3.2. Answers for Some Research Questions	15
Chapter 6: Development Phase	16
6.1. Objective	16
6.2. Progress.....	16
6.2.1. Learn Programming Language and Platforms	16
6.2.2. Protocol Behaviour Learning	17
6.3.3. Selecting Modbus Library	18
6.3.4. Simple GUI Implementation.....	19
6.3.5. Complex GUI and API Implementation	19
6.3.6. Setting up PLC Simulator	20
6.3. Result.....	21
6.3.1. Modbus HMI Application.....	21
Chapter 7 Laboratory Rebuilding Phase	23
7.1. Objective	23
7.2. Process.....	23
7.3. Results	23
7.3.1. Network Topology Design.....	23
7.3.2. Parts installation.....	24
Chapter 8 Conclusion and Recommendations.....	25
8.1. Final Conclusion.....	25
8.2. Recommendations	25
8.2.1. Use Software With Real PLC	25
8.2.2. Application Expansion Using APIs.....	25
8.2.3. Laboratory Expansion to Support More Protocols	26
Evaluation.....	27
References.....	28
Appendix A: Project Plan	I
Appendix B: Graduation Project Survey.....	XV

Appendix C: User Requirement Specification Document Modbus HMI Application ...	XIX
Appendix D: Software Design Document Modbus HMI Application	XXXIV

Summary

Industrial Control Systems are at the core of industrial production processes. They enable automation of production processes with a simple and effective system rather than manual and 24/7 analogue monitored systems used in old times. Modernization of these systems leads to connected system that can be vulnerable to cyber-attacks.

SecurityMatters is a start-up company founded in 2009 which focuses on network monitoring and security systems. The company has some products aimed to secure network environment from cyber-threats.

The assignment is about developing a new HMI application that is using an industrial protocol of interest for company and integrating it into an existing test laboratory. The software made in this graduation project internship expands the company's demonstration and security testing instruments to support Modbus/TCP protocol as well.

This project is separated into four phases: initiation, learning, development, and laboratory rebuilding. The initiation phase did before project started was intended to keep the project on the track by formulating some questions to lead the project.

The learning phase's objective is to learn basics of ICS/ SCADA systems and paves a way for next phases.

The development phase intended to develop an application for simulation laboratory. A HMI application made on this phase can be used as demonstration instrument and security testing tool for SecurityMatters employees.

The laboratory rebuilding phase intended to move simulation laboratory facilities to new headquarter and integrate it with peripherals used in new headquarter.

ICS/SCADA system attacks are real threats that needs to be treated seriously in order to prevent large attacks that might affect company and national securities since it might control sensitive facilities.

Finally, there are some ways to improve this project, connect the software with real PLCs, develop a new application based on the API made in this project, and laboratory facilities expansion to support more protocols.

Glossary

Term	Description
API	Abbreviation for “Application Programming Interface”. This is the interface that allows another application to interact with particular application.
Cloud	Distributed computing via network, where a program runs on network rather than locally.
DLL	Abbreviation for “Dynamic-Link Library”. This is Microsoft’s shared library implementation for Windows systems.
ICS	Abbreviation for “Industrial Control System”. This is the general term for devices used for industrial production.
GUI	Abbreviation for “Graphical User Interface”. This is the type of user interface that allows user to interact with icons and other visual indicators instead of command line.
HMI	Abbreviation for “Human-Machine Interface”. This is the application that built to monitor and control ICS instruments.
Jamod	Open-source Java library for Modbus protocol communication.
Java	Object Oriented Programming language developed by Sun Microsystems.
JavaFX	Platform for creating GUI applications based on Java programming language.
Linux	Free alternative operating system for computers developed by Linus Torvalds.
Modbus	Master-Slave communication protocol developed by Modicon Electronics used for ICS communication.
MVC	Abbreviation for “Model-View-Controller”. This is the programming pattern for OOP that separates the application logic from the GUI.

OOP	Abbreviation for “Object Oriented Programming”. This is the programming paradigm that sees every elements of a program as interacting objects.
OS X	Unix-based graphical interface operating systems developed and marketed by Apple Inc.
PLC	Abbreviation for “Programmable Logic Controller”. This is the component of ICS which has a logic function to determine outputs to actuators as the programmed consequence of inputs from the sensors.
SCADA	Abbreviation for “Supervisory Control and Data Aquisition”. This is a way to monitor, control, and acquire data from remote controllers and field devices. SCADA consists of industrial hardware and software components.
Scene Builder	A component of JavaFX that allows to design Graphical User Interfaces without extensive knowledge of the underlying code.
Windows	Operating system developed by Microsoft.

Chapter 1: Introduction

1.1. Introduction

ICS along with its components are the core of industrial production processes today. They enable automation of production processes with a simple and effective system rather than manual and 24/7 analog-monitored systems used in old times. Figure 1.1 shows an example SCADA network where multiple ICS exchange information to carry out the industrial process.

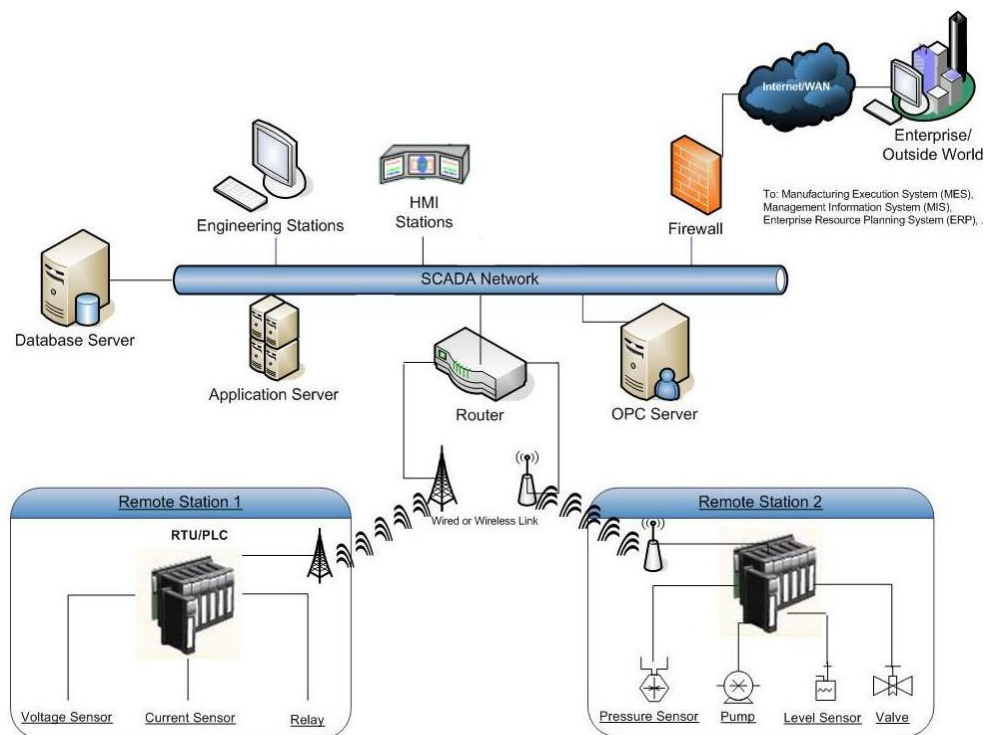


Figure 1.1: SCADA Architecture Using Internet

The most common ICS/SCADA system (Figure 1.1) used to monitor and manage operations of automation instruments is the HMI. It represents industrial devices in an interface, and is used to control and monitor industrial devices anywhere and everywhere.

Modernization opens an opportunity of monitoring and managing ICS remotely, simplifying large scale plant management.

With the usage of TCP/IP networks in ICS/ SCADA systems, cyber-attacks in the past years have been increasingly targeting such systems. For example, the StuxNet malware attack in 2010 successfully disrupted the process of Iran's nuclear power plants. These circumstances led to a need to find better way to protect industrial networks and its equipment.

The purpose of this graduation project is to contribute to the development of a laboratory for ICS attack simulation aimed to discover possible vulnerabilities on ICS instruments that might be exploited by hackers.

The structure of this final report is the following:

- Chapter 2 introduces the host company.
- Chapter 3 describes the assignment.
- Chapter 4 defines the phasing and methodology.
- Chapter 5 describes the first phase of the project.
- Chapter 6 describes the second phase of the project.
- Chapter 7 describes the third phase of the project.
- Chapter 8 contains conclusions and recommendations.

Chapter 2: The Company

2.1. History

SecurityMatters is a start-up company which focuses on network monitoring and security systems. The company was founded in 2009 by Damiano Bolzoni, Sandro Etalle and Emmanuele Zambon to bring the new security technology to the market. Despite aged just 4 years, SecurityMatters succeeded to prove capabilities of their product by having a top-notch companies as their client such as The Boeing Company.

2.2. Mission, Vision, and Values

The mission of SecurityMatters is to deliver game-changing technology that makes its customers more secure and in control. The company vision is to be recognized as leading provider of high-quality innovative solutions for cybersecurity.

As a start-up company, SecurityMatters has to keep improving the performance of its products to compete with the other companies that are working on the same field around the world. To achieve that, the company has a research and development team which employs a simulation laboratory and tools to simulate real world situation on industrial instruments and systems as well as attack scenarios that might occur in a production process.

2.3. Products

In line and industrial from cyber-threats, SecurityMatters products aim at securing enterprise networks.

Currently, the company has two main products. SilentDefense ICS, for securing an industrial environment, and SilentDefense Web, to secure web applications. Both of the products have an advantage to detect attacks without a signature, thus protecting systems from zero-day attacks.

SilentDefense secures industrial processes and web-applications by analyzing and reporting malicious traffic that might be generated from hackers and unauthorized users of the system. SilentDefense defines a whitelist of allowed communication patterns from each device, and makes

it as a base of monitoring. Whenever an abnormal communication is detected, SilentDefense will report and capture the traffic along with the details that helps the IT security team to resolve it. Figure 2.1 shows an example deployment of SilentDefense ICS to an industrial network.

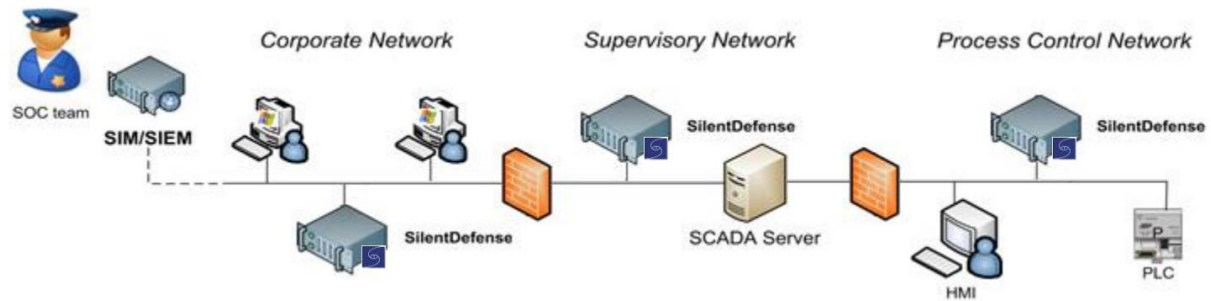


Figure 2.1: SilentDefense™ Topology Example

Chapter 3: The Assignment

3.1 Background

Recently, the company moved their headquarter from Enschede to Eindhoven. While most of the operations are now carried out in the new location, the simulation laboratory is still located in Enschede. The relocation of the simulation laboratory was scheduled for the beginning of May 2014. Meanwhile, the company also wants to expand their demo instruments to support the testing of more industrial protocols.

The company has a HMI communicating with real and simulated PLCs using MMS and IEC 61850 protocols in their simulation laboratory. By expanding the protocol support, the company will have broader choice for their potential customers and internal team to test misuse cases regarding ICS/SCADA system security, and to have better understanding about vulnerabilities that a PLC might have.

3.2. Assignment Overview

The assignment is about developing a new HMI application that is using other protocols than the company currently has and integrating it into the laboratory that was going to be moved in May 2014. The software made in this graduation project internship expands the company's demonstration and security testing instruments to support Modbus/TCP protocol as well.

The assignment is expanded into 4 phases: initiation, research, development, and lab rebuilding phase. The assignment will be detailed in chapter 4, and details of each phases will be described in chapter 5 through 7.

3.3. Project Deliverables

The final deliverables of this assignment are:

1. An application to enable communication between a server and a PLC that displays process values in a GUI, and allows to simulate unusual and incorrect traffic for security testing.

2. Simulation laboratory in Eindhoven headquarter and integrate it with the developed application.

3.4. Goals

This project was started mainly because SecurityMatters needs to constantly research on vulnerabilities of industrial networks and systems. Furthermore, it needs to move the simulation laboratory to the new headquarter in Eindhoven. The main goals of this project is to provide SecurityMatters with an extra instrument to simulate and analyze communications in an industrial network, discover new ICS vulnerabilities, and improve the performance of its products.

3.5. Changes During the Project

There have been some changes during the project progress, which are described below:

3.5.1. Change of Phase Order

At the end of the first phase, the company realized that it was not feasible to move the simulation laboratory devices from Enschede to Eindhoven at that moment. The reason is that there are servers for which the company cannot afford to have a down time, since the development team was working on a new software release.

As a solution, the company mentor decided that the laboratory rebuilding phase would be shifted into 3rd phase instead of 2nd phase.

3.5.2. Programming Language

At the beginning of the development phase, the company requested programs to be made on Java programming language. The reason is that a Java application will have more compatibilities across platforms used in SecurityMatters computing environment and will be easier to use and maintain.

This decision directly affects the development phase since the Java programming language has to be learned before starting the real application development. While this was delaying the progress of the development phase, it is good both for the developer education and the company to have the application written in Java language, as this opens the

opportunity for the application to be used in other environments such as Linux or OS X rather than just Windows.

3.5.3. Project Focus

This project was preliminary designed with more technical networking tasks in mind. However, to the requirements of university focusing the assignment on software development, these parts of the project were minimized to accommodate.

Chapter 4: Phasing and Methodology

This project is organized into four phases: initiation, learning, development, and laboratory rebuilding. The summary and timeline of phasing are described on following picture:








		Task Mode ▾	Task Name ▾	Duration ▾	Start ▾	Finish ▾
1			Initiation	5 days	Tue 12/17/13	Sat 12/21/13
2			Initiation	1 day	Fri 1/31/14	Fri 1/31/14
3			Learning	30 days	Tue 2/4/14	Mon 3/17/14
4			Development	45 days	Mon 3/10/14	Fri 5/9/14
5			Learning	5 days	Mon 5/12/14	Fri 5/16/14
6			Laboratory Rebuilding	15 days	Mon 5/12/14	Fri 5/30/14

Figure 4.1: Project Phasing

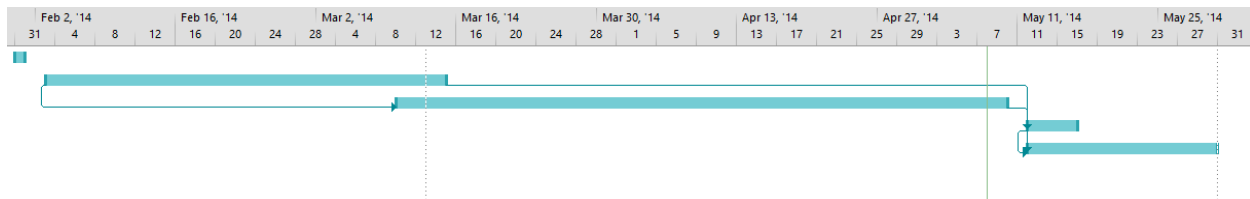


Figure 4.2: Project Timeline

4.1. Initiation and Research Phase

This phase was done before the actual start of the internship. The objective of this phase was to make progress be assured during the graduation project.

The project was carried out with these questions in mind to keep the project on track. The research questions are:

1. How to make a network topology and define configurations to prevent hackers having access to equipment directly?
2. How to find a method to find vulnerabilities of the devices (PLC, etc.)?
 - 2.1. What programming language and protocols should the HMI application use?
 - 2.2. How do Programmable Logic Controllers communicate?
 - 2.3. How can Programmable Logic Controllers be attacked?
3. How to find a method to find vulnerabilities of the network?

3.1. What devices are needed as an addition to the simulation laboratory in Eindhoven headquarter?

4.2. Learning Phase

This phase was done as a first phase of the internship, and the purpose is to have more understanding about ICS and SCADA systems before starting the development phase. The learning phase will be detailed more on chapter 5.

4.3. Development Phase

The development phase consisted of developing an HMI application for the company after knowledge about PLC protocols was acquired. The development phase will be detailed more in chapter 6.

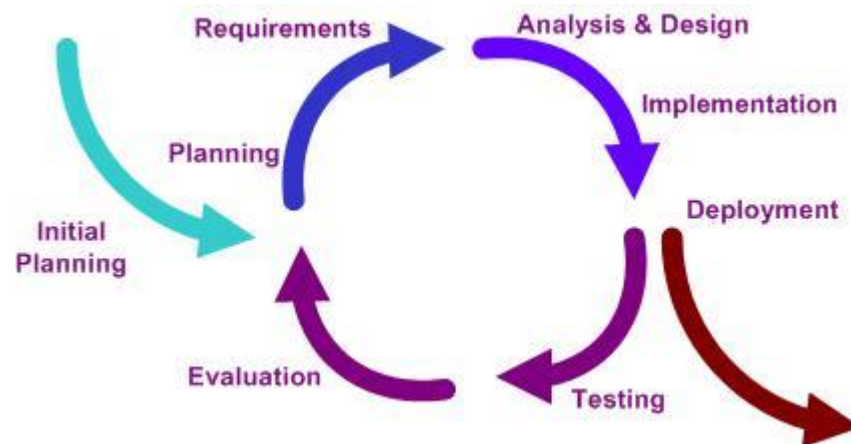


Figure 4.3: Iterative Methodology

In this phase, iterative methodology was used both because the developer was not familiar with Java programming language and in order to continuously get and integrate feedback from company mentor about the result. More precisely, the development is achieved by the iteration of the following steps:

4.3.1. Defining Requirements

In this step, requirements are defined for each iteration. The requirements itself are based on the goals that are being achieved. Example: Requirement to implement simple client-server communication using the Jamod library.

4.3.2. Analysis and Design

As the next step, requirements are translated into analysis and design. The upcoming implementation are based on the design made in this step. Example: Use case of Jamod library simple implementation.

4.3.3. Implementation

This step implements design made on previous step. Example: Create an application that implements Jamod library.

4.3.4. Testing

This step tests result made on implementation phase to know whether features are working as intended.

4.3.5. Evaluation

After testing step, evaluation step will be taken and feedback collected to prepare the next iteration.

4.4. Rebuilding Laboratory Phase

The rebuilding laboratory phase intended to rebuild the laboratory that had to be moved from Enschede to new headquarter in Eindhoven. The rebuilding laboratory phase will be detailed more on chapter 7.

Chapter 5: Learning Phase

5.1. Objective

This phase was done to learn the basic concepts of ICS/SCADA systems since this field is new for the writer. The objectives of this phase are :

1. To gain knowledge how ICS / SCADA systems work.
2. To gain knowledge how PLCs communicate and what protocols can they use.
3. To answer research questions.
4. To have more understanding about what is going to be developed in the next phase.

5.2. Topics

The main topics which have been studied in this phase are presented below:

5.2.1. ICS/SCADA System Components and Protocols

This topic was chosen as the topic learnt to acquire basic knowledge about what ICS/ SCADA systems are and how they work, along with typical components and common protocols . This knowledge paves the next step of learning since the project itself will correlate with ICS/ SCADA system components such as PLC and industrial networks.

5.2.2. Industrial Process

This topic includes ICS/ SCADA system component role on common industrial process such as water treatment, oil and gas mining, and power plants including wind, hydro, nuclear, and heat power generator.

5.2.3. Electricity Distribution

This topic was chosen after obtain knowledge on the electricity transmission and distribution process, including transmission substation for high voltage electricity, distribution substation for low voltage electricity, smart meter, and smart grid.

The importance of learning about the electricity distribution process is in preparation of the study of applications that simulates this process.

5.2.4. Protocols for each Application

The knowledge about components used in industrial process and electricity distribution was then coupled with protocols that might be used on each processe to have a complete overview of real situations.

5.3. Result

The learning phase led to the following results:

5.3.1. Project Plan

The first result of learning phase is project plan, a document that describes activities planned for the next phases of the project. This document defines practical steps for next activities and also as an indicator of the project progress.

5.3.2. Answers for Some Research Questions

5.3.2.1. *What Programming Language and Protocols Should the HMI Application use?*

The HMI Application should be designed according to the requirements of the company's. In this case, the HMI application was developed using Java programming language and the Modbus protocol.

5.3.2.2. *How do Programmable Logic Controllers Communicate?*

In the past, PLCs were communicating using serial ports and proprietary protocols, since there was no standard regulating how they should communicate. However, PLC nowadays are usually communicating over TCP/IP and standard protocols to enhance connectivity, ensure compatibility, and reduce development cost.

5.3.2.3. *How can Programmable Logic Controllers be Attacked?*

PLCs can be attacked by sending wrong or too many commands to the device itself. Depending on PLC characteristics, the effects on the PLC may vary. It may simply reject the command or completely crashes.

Chapter 6: Development Phase

6.1. Objective

The development phase was started with some objectives in mind:

1. Learn Java and JavaFX programming language and platform.
2. Know how PLC device communicate in detail.
3. Find libraries that suits application requirements.
4. Develop the HMI application for SecurityMatters.

6.2. Progress

The development was carried out in several iterations:

6.2.1. Learn Programming Language and Platforms

The first iteration consists of learning the programming language and platforms that will be used for application development. Java programming language and JavaFX platform were chosen as the base of the software that will be developed.

Java was chosen because it provides multi-platform support. This is a benefit for SecurityMatters since it ensures interoperability. For example: Developer uses Ubuntu Linux platform, while project manager uses Windows and others are using OS X platform. Furthermore, Java is a well-known programming language within SecurityMatters, thus facilitates the reuse and maintenance of application code.

JavaFX was chosen as the GUI platform because it provides multi-platform support and it is intended to replace Swing, an older GUI platform for Java. This choice ensures that the application is using the latest technology that will not be deprecated fastly and also can be developed and maintained easily.

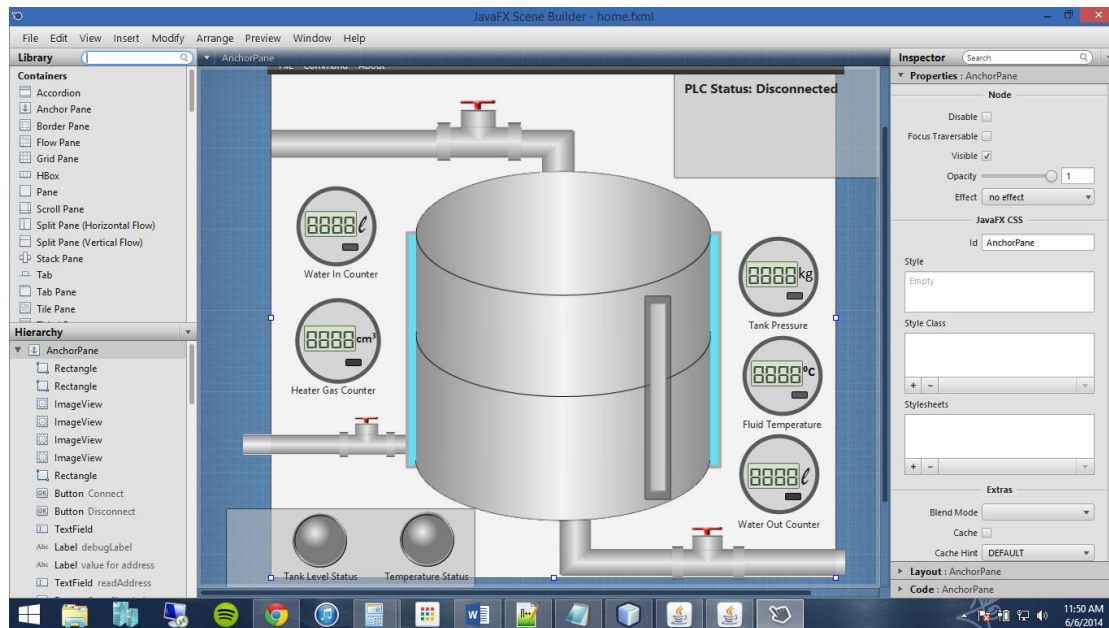


Figure 6.1: JavaFX Scene Builder Example

6.2.2. Protocol Behaviour Learning

Protocol behaviour learning was needed as the next step, to know how an Modbus HMI application communicates and interacts with PLC. This iteration was approached using two methods: theoretical and practical.

Protocol specification sheets were studied to know how the protocol is designed and how it works.

The approach used in practical consisted of wiretapping communications over protocol simulator and analyzing it to observe how the protocol was used in practice was done using Wireshark and RawCap application to capture traffic generated between simulators.

A PLC Simulator was used to simulate traffics that might occur in a real world scenario, using tool already present in the simulation laboratory of SecurityMatters. While it might not be perfectly accurate, it still give a sufficient overview about how particular protocol works.

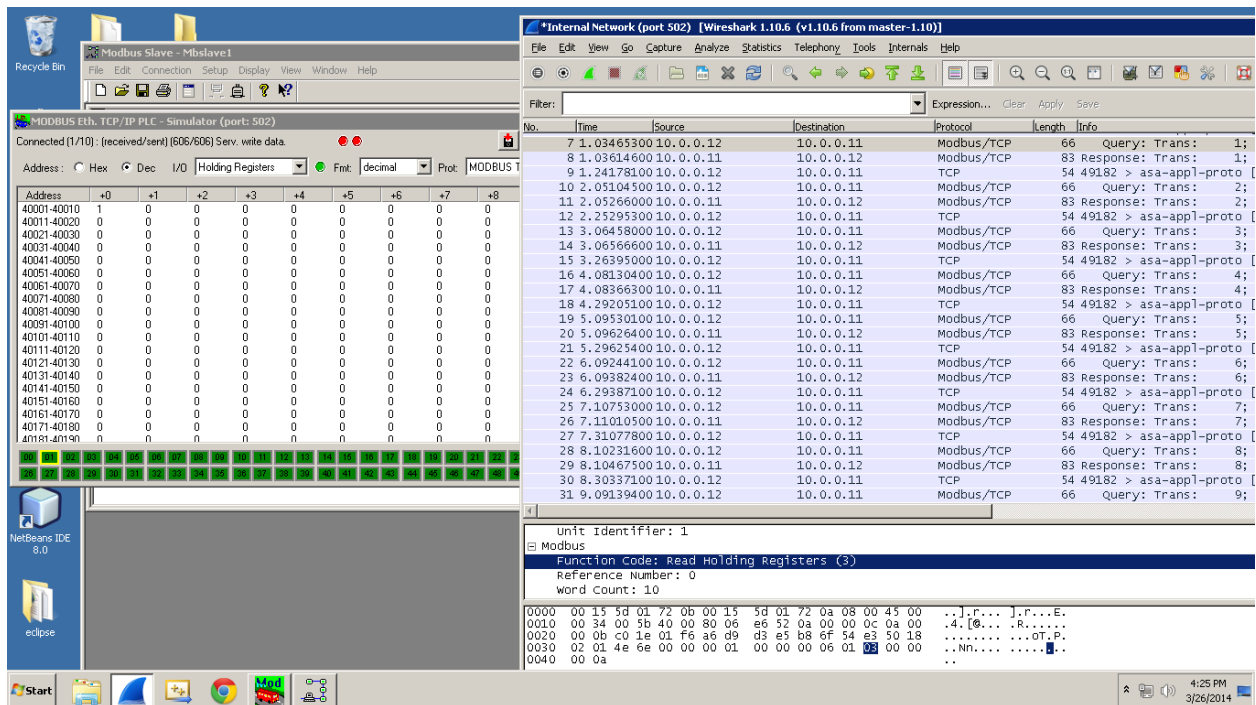


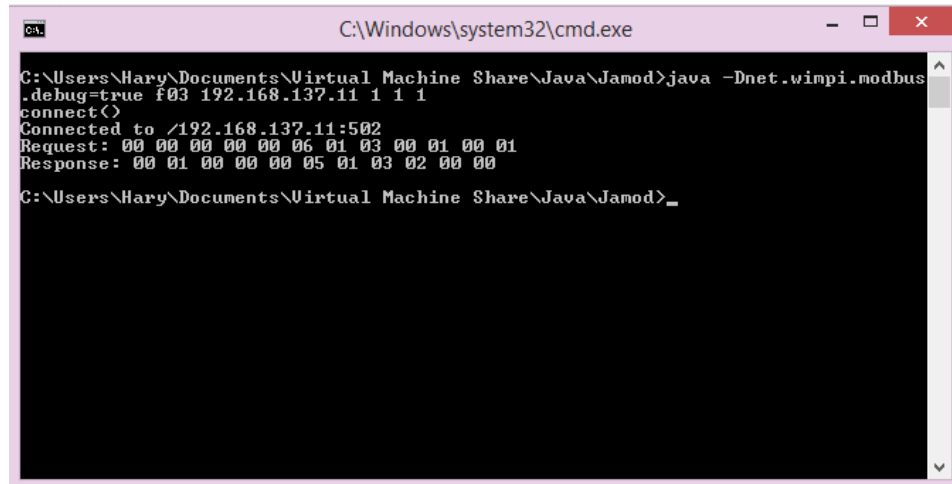
Figure 6.2: Wiretapping Progress

6.3.3. Selecting Modbus Library

Given the availability of several Modbus libraries, the HMI application development will reuse one of these libraries to simplify the development process. Finding the best library is quite tricky since every library has its own characteristics.

For the Modbus HMI Application, the Jamod library was eventually selected because it is based on Java programming language and implements all the functions that were needed for the application itself. Several libraries such as libmodbus, Modbus4j and Jamod2 were considered, but since Jamod was the simplest to implement, has a good documentation, and was written in Java, it was chosen eventually.

Below is the simple Jamod implementation in Command Line Interface result from this iteration to prove that the HMI was able to communicate with PLC simulator:



```
C:\Windows\system32\cmd.exe

C:\Users\Hary\Documents\Virtual Machine Share\Java\Jamod>java -Dnet.wimpi.modbus
.debug=true f03 192.168.137.11 1 1 1
connect()
Connected to 192.168.137.11:502
Request: 00 00 00 00 00 06 01 03 00 01 00 01
Response: 00 01 00 00 00 05 01 03 02 00 00

C:\Users\Hary\Documents\Virtual Machine Share\Java\Jamod>
```

Figure 6.3: Jamod Simple Implementation

6.3.4. Simple GUI Implementation

After the proof-of-concept built in the previous iteration, a preliminary GUI was implemented. This would be the proof-of-concept that JavaFX and existing libraries can be combined.

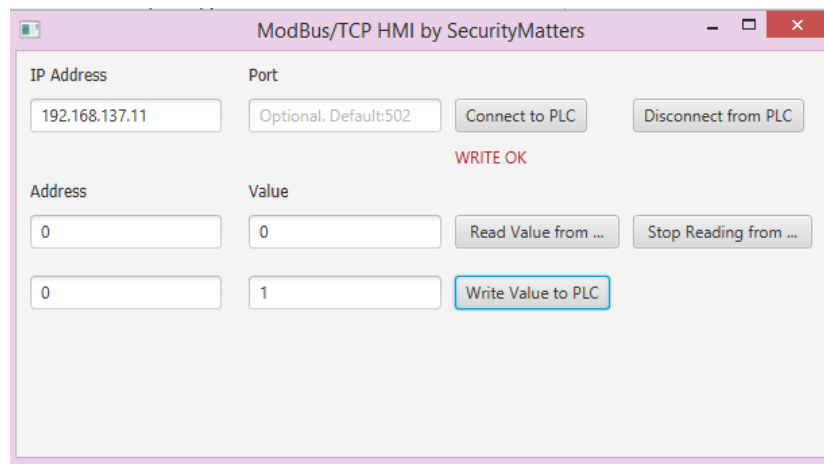


Figure 6.4: Jamod Preliminary GUI Implementation

6.3.5. Complex GUI and API Implementation

After being succesful in the implementation of a simple GUI, a more elaborated GUI was designed to resemble a real industrial HMI. To achieve that, APIs were created to simplify development of complex application and also for further development.

Upon development of complex software, APIs are definitely needed to ensure that the programmer will not write the same code twice. This approach will reduce the development effort since a method has only to be written once and can then be reused. So

whenever the programmer has an error in the particular method, the programmer will only have to change the method, and nothing else.

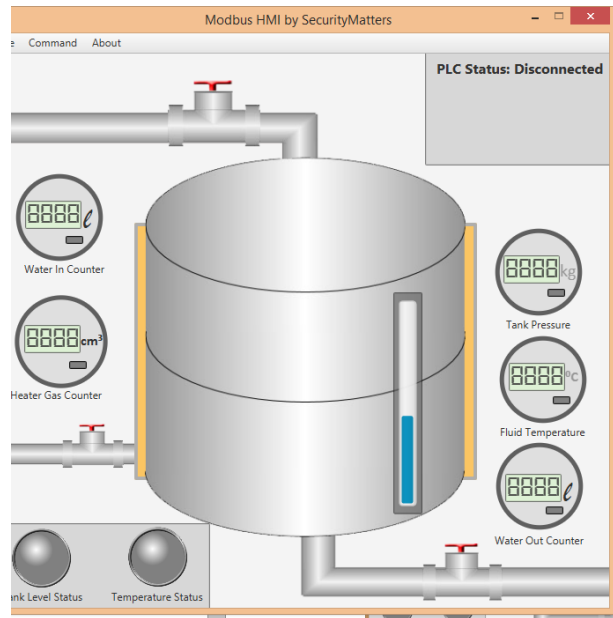


Figure 6.5: Final Working Application

6.3.6. Setting up PLC Simulator

During the development, Modbus protocol traffic was generated using a PLC Simulator. This simulator generates PLC traffics for essential core functions based on a logic implemented in a script.

The script made for this project is the improvement of an existing script prepared by a company's employee. Changes were made to make the PLC's logic more realistic and add more features to match with HMI application requirements defined in earlier stages.

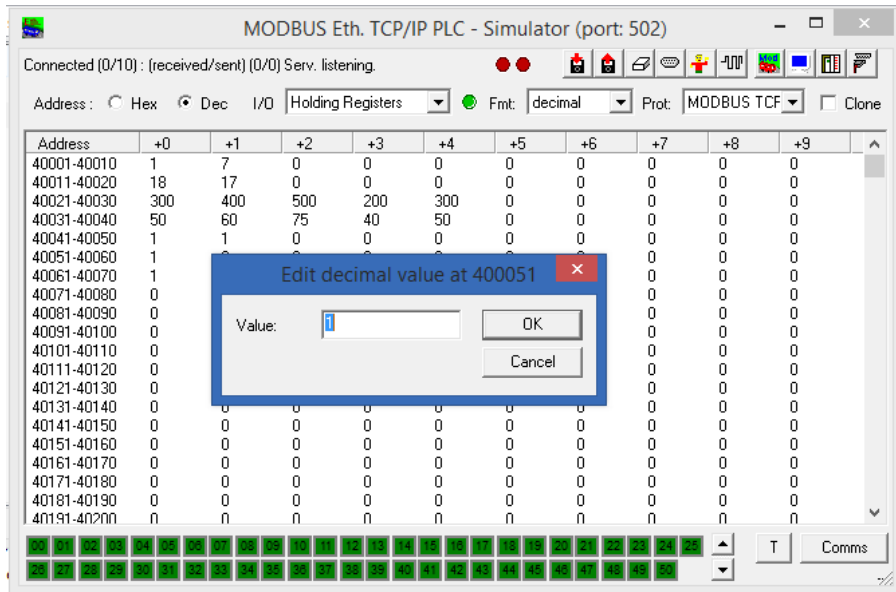


Figure 6.6: PLC Simulator Software

6.3. Result

6.3.1. Modbus HMI Application

The resulting Modbus HMI is an application that resembles an industrial water boiler, with some more functions to test PLC from several threats. This application is using Jamod as the Modbus message translator between the PLC simulator or device with the HMI application itself.

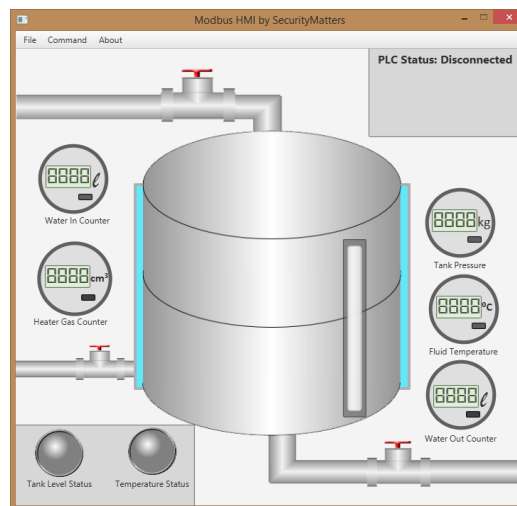


Figure 6.6: Modbus HMI Application

As mentioned earlier, this application is designed to generate Modbus traffic both for normal and test functionalities. The normal functions are connect, read, write, and disconnect from PLC, while test function consist of sending custom message directly to the PLC.

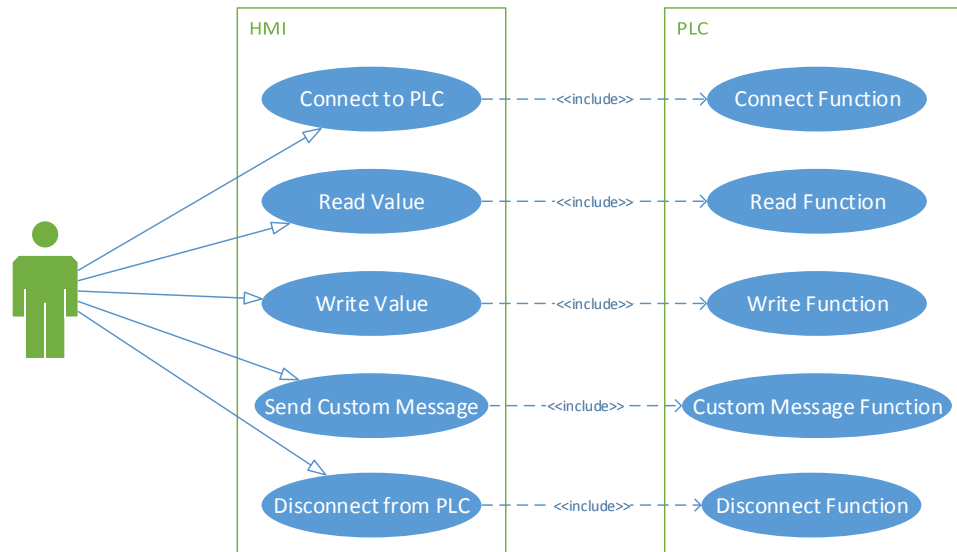


Figure 6.7: Modbus HMI Application Use Cases

The application is based on MVC architecture, which separates application interface from the logic part to make further development easier. The model and controller are based on Java and the view is based on JavaFX.

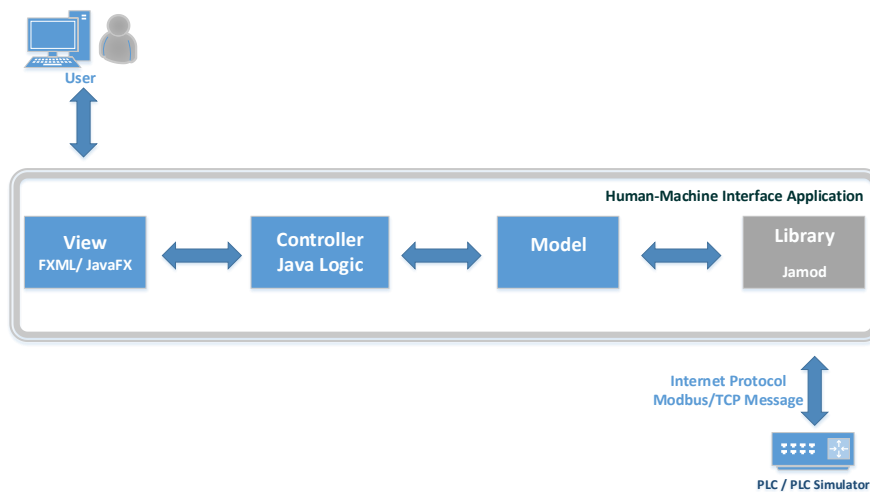


Figure 6.8: Modbus HMI Application Architecture

Chapter 7 Laboratory Rebuilding Phase

7.1. Objective

The objectives of this phase are:

1. Move simulation laboratory from Enschede to the new Eindhoven headquarter.
2. Integrate simulation laboratory with peripherals in new headquarter.

7.2. Process

7.2.1. Network Consideration

Lab movement started with the network design draft that will be realized once equipments will be moved to the new headquarters.

In the past three months, employees had to deal with slow and unreliable wireless connection which affects productivity. The new network design draft included the infrastructure changes required to provide ethernet cable connection to SecurityMatters employees.

7.2.2. Device preparation

The network design phase continued with device procurement as a preparation before peripheral arrived to the Eindhoven headquarters. In this phase cables were also prepared according to the design requirements.

7.3. Results

The results of this phase are the following:

7.3.1. Network Topology Design

Following some discussion regarding network design for new headquarter, the result is shown in the network map below:

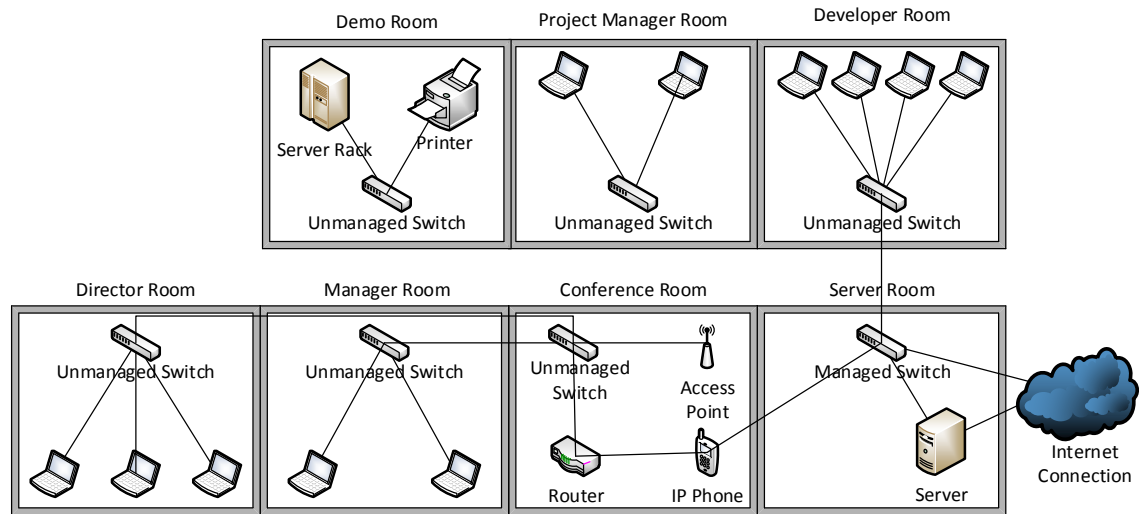


Figure 7.1: Network Map in Eindhoven Headquarter

7.3.2. Parts installation

Devices that arrived were installed according to the design diagram on room A,B, and C to bring connectivity to those rooms. The other rooms were still waiting for servers that have not yet arrived due to some internal delays.

Chapter 8 Conclusion and Recommendations

8.1. Final Conclusion

ICS/SCADA system attacks are real threats that need to be treated seriously in order to prevent large attacks that might affect company and national safety and security.

One of the ways to prevent attacks is to actively test devices for vulnerabilities to some exploits. In this way, both users and vendors could be aware to the threats to their systems. Along with the tests, production environments must have some mechanism to prevent or reduce effects from ICS/ SCADA system attacks.

The HMI software made in this project will be useful for internal testing and demonstration purposes, and the API developed will make future Modbus application development easier.

PLC protocol simulation was possible using PLC simulator application, it reflects essential PLC functionalities but not vendor specific characteristics and vulnerabilities that might be found on real PLCs.

The moving of the simulation laboratory to Eindhoven headquarter has major effects for both employees and potential customers. Effects for employees are better control and faster access to servers. Effect for potential customers is the possibility to see the company's products capabilities in a simulated working environment rather than just through videos or other marketing methods.

8.2. Recommendations

8.2.1. Use Software With Real PLC

My first recommendation for this project is to connect the HMI application to real PLCs instead of a simulated one. The reason is because only real PLCs can be tested for vulnerabilities, so HMI application test feature could be more effective.

8.2.2. Application Expansion Using APIs

My second recommendation is to make another Modbus HMI application using the APIs developed in this project, to have another use or study case for the Modbus protocol.

8.2.3. Laboratory Expansion to Support More Protocols

Last but not least, I would suggest to develop more protocol support for simulation laboratory facilities. In this way, SecurityMatters could convince more potential customers to buy and use its products.

Evaluation

The most interesting part of the internship is to learn about components that are really used in different industries, including PLC devices and its protocols because I never had a chance to learn such thing. This situation made me feel more challenged and eager to know more about it.

All project tasks were challenging for me. Even though I had some difficulties on learning some concepts, there was always support from company mentor and colleagues from SecurityMatters.

Aside from knowledge, I also gained soft skills like time management and interpersonal skills. This valuable experience teaches me how to be a better person on my upcoming career.

Overall, it was a good experience to work together with SecurityMatters company and its amazing colleagues. Aside from mistakes that I made, I am proud of those achievements.

References

Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems (IDC Technology)
by Gordon Clarke CP Eng BEng MBA, Deon Reynders Pr Eng BSc (ElecEng) (Hons) MBA.

Reference number	Source	URL Address	Description
1	Wikipedia	http://en.wikipedia.org/wiki/Modbus	Modbus protocol description
2	Modbus Organization site	http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf	Modbus protocol specification
3	ABB AC800M product site	http://www05.abb.com/global/scot/sco t349.nsf/veritydisplay/b48192f9da2e1 947c12579c7005ea32a/\$file/3BSE063 717_A_en_Compact_800_5.1_Flexibl e_process_control_products.pdf	ABB AC800M brochure
4	Remmon site	http://www.remmon.com/en/products/web_based_hmi/mtom.php	HMI application example
5	Viet Phat group site	http://vietphatgroup.com/index.php?mod=news&cpid=207	SCADA system architecture example
6	Lockheed Martin site	http://id.lockheedmartin.com/blog/stuxnet-whitepaper-updated	Stuxnet description
7	SecurityMatters site	http://www.secmatters.com/resources	SilentDefense product information
8	Wikipedia	http://en.wikipedia.org/wiki/Iterative_and_incremental_development	Iterative development method
9	Wikipedia	http://en.wikipedia.org/wiki/Application_programming_interface	API
10	Wikipedia	http://en.wikipedia.org/wiki/Cloud_computing	Cloud
11	Wikipedia	http://en.wikipedia.org/wiki/Dynamic-link_library	DLL
12	Wikipedia	http://en.wikipedia.org/wiki/Industrial_Control_System	ICS
13	Wikipedia	http://en.wikipedia.org/wiki/Graphical_user_interface	GUI
14	Wikipedia	http://en.wikipedia.org/wiki/Human-machine_interface	HMI
15	SourceForge	http://jamod.sourceforge.net/	Jamod
16	Wikipedia	http://en.wikipedia.org/wiki/Java_(programming_language)	Java
17	Wikipedia	http://en.wikipedia.org/wiki/JavaFX	JavaFX

18	Wikipedia	http://en.wikipedia.org/wiki/Linux	Linux
19	Wikipedia	http://en.wikipedia.org/wiki/Model-view-controller	MVC
20	Wikipedia	http://en.wikipedia.org/wiki/Object-oriented_programming	OOP
21	Wikipedia	http://en.wikipedia.org/wiki/OS_X	OS X
22	Wikipedia	http://en.wikipedia.org/wiki/PLC	PLC
23	Wikipedia	http://en.wikipedia.org/wiki/SCADA	SCADA
24	Wikipedia	http://en.wikipedia.org/wiki/JavaFX	Scene Builder
25	Wikipedia	http://en.wikipedia.org/wiki/Microsoft_Windows	Windows

Appendix A: Project Plan

PROJECT PLAN**Fontys University of Applied Sciences****HBO-ICT: ENGLISH STREAM****Industrial Control Systems Simulation Laboratory Development**

Student Data	
Name	Liuwan, Haryanto Natalius
Student Number	2331039
Project Period	1 st February 2014 until 31 st May 2014
Company Data	
Company Name	SecurityMatters BV
Department	
Address	Twinning Centre K.5.10 Eindhoven
Tutor Name	Trivellato, Daniel
Position	Project Manager
University Tutor	
Name	Schellekens, Casper

Approved and Signed by:

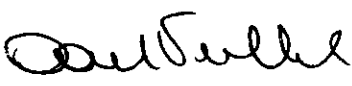
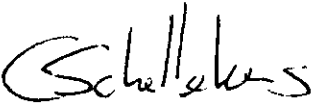

		
Date: 14-03-2014	Date: 14-3-2014	Date: 14-3-2014
Daniel Trivellato Company Tutor	Casper Schellekens University Tutor	Haryanto Natalius Liuwan Project Manager

TABLE OF CONTENTS

INTRODUCTION.....	IV
PROJECT STATEMENT	V
2.1. Formal Client	V
2.2. Project Leader	V
2.3. Project Context	V
2.4. Project Motivation	V
2.5. Project Results.....	VI
2.6. Project Deliverables and Non-Deliverables	VI
Ang project non-deliverables are:	VI
2.7. Project Constraints.....	VI
Possible project constraints are:	VI
2.8. Project Risks	VII
2.9. Research.....	VII
PROJECT PHASING	VIII
MANAGEMENT PLAN	X
4.1. Money.....	X
4.2. Skills.....	X
4.3. Quality	X
4.4. Information	XII
4.5. Time	XII
4.6. Organisation	XIII

CHAPTER 1

INTRODUCTION

This document describes the detailed plan of the writer's graduation project carried out at SecurityMatters BV. The project assignment itself consist of Setting up an Industrial Control Systems Cybersecurity simulation laboratory, to simulate cybersecurity threats that may target real-word industrial operations. The writer's role in this project is as the laboratory engineer and software designer.

The main topic of the graduation project is about cybersecurity in Industrial Control Systems, and as the first step of this project, the writer has to study Industrial Control Systems equipment and protocols. Second, the writer has to design and develop an application to enable communication between industrial equipment. Finally, the writer has to rebuild and integrate the simulation lab that SecurityMatters currently has in Enschede.

CHAPTER 2

PROJECT STATEMENT

2.1. *Formal Client*

Daniel Trivellato will be the representative of SecurityMatters B.V in this project.

2.2. *Project Leader*

The writer will act as project leader and is responsible for all communication between the project participants and the external parties.

2.3. *Project Context*

Cyber-attacks in the past years increasingly targeted Industrial Control Systems such as Supervisory Control and Data Acquisition, servers, and Programmable Logic Controllers. For example, StuxNet malware attack in 2010 successfully disrupted the process of Iran's nuclear facilities at a huge scale. These circumstances led to a need to find better method to protect industrial networks and their equipment.

SecurityMatters is a start-up company which focuses on network monitoring and security systems. Their products are more focused on securing industrial processes from cyber threats. As a company which focuses on security, SecurityMatters needs some research on how Industrial Control Systems, for example Programmable Logic Controller, can be misused by the other parties. Recently, the company moved their headquarter from Enschede to Eindhoven. While most of the operations are now carried out in the new location, the simulation laboratory is still located in Enschede, The relocation of the simulation laboratory is scheduled for the beginning of May 2014.

2.4. *Project Motivation*

This project was started mainly because SecurityMatters needs to constantly research on vulnerabilities of industrial networks and systems. Furthermore, they need to move their simulation laboratory to the new. Therefore, the goals of this project are:

1. Develop some applications to simulate normal and abused communication between

Industrial Control Systems.

2. Rebuild and integrate the simulation laboratory on the new headquarter.

2.5. *Project Results*

The end result of this graduation project would be:

1. Applications to enable communication between a server and a Programmable Logic Controller that also features a functionality to show values in a Graphical User Interface whenever it is needed, and to simulate wrong traffic for security testing.
2. Rebuilding the simulation laboratory in Eindhoven headquarter and integrate it with the developed applications.

2.6. *Project Deliverables and Non-Deliverables*

The project deliverables are:

1. Project Plan
2. Application(s), along with:
 - a. Requirements Document
 - b. Design Document
3. Laboratory rebuilding design

The project non-deliverables are:

1. Activity notes
2. Meeting minutes

2.7. *Project Constraints*

Possible project constraints are:

1. The preferred programming language by the company is Java, which is not mastered by writer yet.
2. Applications must be able to communicate with MMS and ModBus protocols, which are new to the writer.

3. Existing simulation laboratory complexity level is not been known to the writer yet.

2.8. *Project Risks*

Project has the following risks:

1. Learning fails to meet expectations; if this happened, then project leader will discuss with formal client, to obtain more information and knowledge regarding protocols that will be used.
2. Applications fail to meet in terms of functionality expectations, expectations functionalities; if this happened then project leader will discuss with formal client, to get more resources and help from colleagues during the development phase. To make sure that work is done on schedule, formal client and project leader will meet weekly to discuss about the progress of the project.
3. Rebuilding laboratory phase may not be successful because of missing equipment, unknown peripherals, and time constraint. If this happened then project leader will discuss this circumstance with formal client, since the equipment itself are mission-critical, thus a good preparation and coordination would be needed.

2.9. *Research*

Research is one of the most important parts of this project because the topic is completely new for the writer. Some of the research questions are as follows:

1. What programming language and protocols should the application use?
2. How do Programmable Logic Controllers communicate?
3. How can Programmable Logic Controllers be attacked?
4. What devices are needed as an addition to the simulation laboratory in Eindhoven headquarter?

Hopefully these questions will be answered as the project progresses in order to help defining the final product.

CHAPTER 3

PROJECT PHASING

This chapter describes an approximate schedule for the graduation project. The project started on February 1st 2014 and is expected to finish on May 31st 2014.

Phase 1: Learning Phase

Started on February 1st 2014 until March 14th 2014, and May 12th 2014 until May 16th 2014.

Time for corresponding phase: 6 weeks

Deliverable(s) : Project plan

Tasks :

1. Project definition

Gather general informations needed and determine the requirements of the project.

2. Learn from particular sources

Search internet and books for information regarding the project topic to build the required understanding for the next phases.

3. Consult with formal client

Discuss with formal client about the progress of the learning phase..

4. Finalize project plan

Plan the project for tracking project progress.

Phase 2: Development Phase

Started on March 10th 2014 and expected to be finished on May 9th 2014.

Time for corresponding phase: 9 weeks

Deliverable(s) : Applications, design, and requirement documents

Tasks :

1. Create requirement documents

Define client requirements and report them into a requirement document.

User Requirements Specification Document

2. Create design documents

Create a documentation of the application that will be developed.

3. Develop the application

Realization from design documents.

4. Test application

Test whether the application functions as required.

5. Discuss with formal client

Discuss with formal client about the progress of the development phase.

Phase 3: Laboratory Rebuilding Phase

Expected to start on May 12th 2014 and expected to be finished on May 31st 2014.

Time for corresponding phase: 3 weeks

Deliverable(s) : Design and report of new simulation laboratory

Tasks :

1. Determine requirements of the laboratory

Define client requirements and report to formal client what devices are needed.

2. Make a design of the new laboratory

Make a documentation of the laboratory design that would be implemented.

3. Implementation

Rebuild the laboratory as by design document.

4. Discuss with formal client

Discuss with formal client about the progress of the laboratory rebuilding phase.

CHAPTER 4

MANAGEMENT PLAN

4.1. *Money*

No money is involved since all room and equipment are provided by formal client.

4.2. *Skills*

The skills needed for each phase are the following:

1. Learning Phase

During this phase, analytic skills are needed as most of time spent for searching and studying information, and planning skills are needed to write the project plan.

2. Development Phase

On this phase, Java and C# language programming skill would be needed as the application are going to be written on corresponding language. Also, knowledge about Supervisory Control and Data Acquisition or Industrial Control System protocols such as ModBus and MMS would be needed.

3. Laboratory Rebuilding Phase

During this phase, networking and planning skills are needed as the laboratory needs to be reconfigured, and there is a chance that a new devices will be added to the laboratory.

4.3. *Quality*

The overall quality of the end product can be measured in several aspects:

1. Features

A good application requires all features listed in the requirement document.

2. Modularity

A good product, including application and laboratory design need to be modular so that they can be expanded in case it is needed.

3. Interface

A good application needs a straight-forward interface to make it easy to use and understand.

4.4. Information

The following table shows the actions required by each project participant in relation to the project deliverables.

	Participant	Project Plan	Requirement Document	Design Document	Final application and report	Graduation Final Report
Project Leader	Haryanto Natalius Liuwan	Cr, Ar, S, Di	Cr, Ar, S, Di	Cr, Ar, S, Di	Cr, Ar, S, Di	Cr, Ar, S, Di
Company tutor / Formal Client	Daniel Trivellato	Di, R, Ar, A	Di, R, Ar, A	Di, R, Ar, A	Di, R, Ar, A	Di, R, A
University Tutor	Casper Schellekens	Di, R, Ar, A	Di, R	Di, R	Di, R	Di, R, A

Legend:

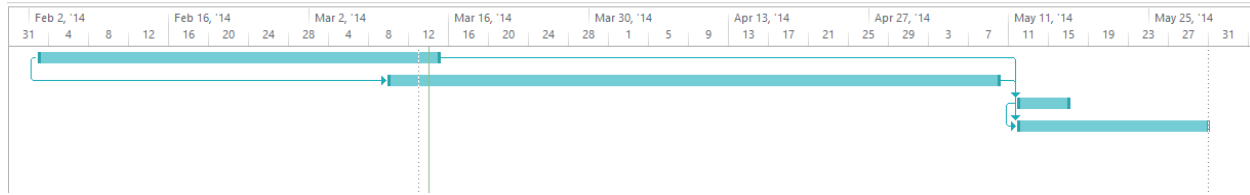
Cr	Create
Di	Discuss
A	Approve
S	Send
R	Receive
Ar	Archive

4.5. Time

The following figures summarize the duration and execution dates of the project activities.

Task Name	Duration	Start	Finish	Predecessors
Learning	30 days	Mon 2/3/14	Fri 3/14/14	
Development	45 days	Mon 3/10/14	Fri 5/9/14	1SS
Learning	5 days	Mon 5/12/14	Fri 5/16/14	1,2
Laboratory Rebuilding	15 days	Mon 5/12/14	Fri 5/30/14	3SS,2

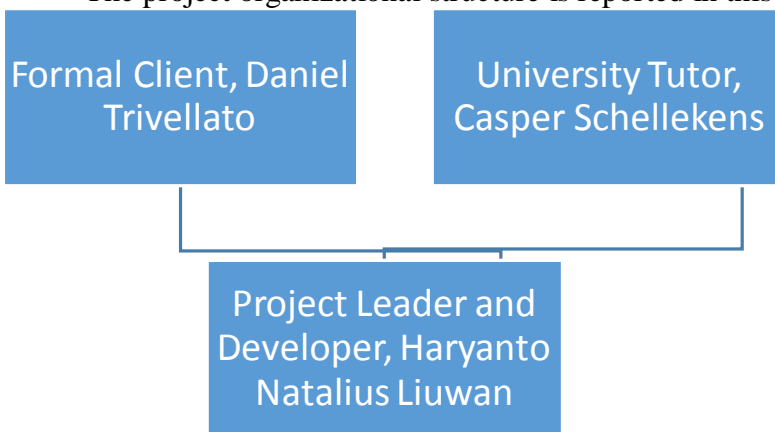
User Requirements Specification Document



A learning phase will precede both the development phase and laboratory rebuilding phase since the topics covered are quite different.

4.6. Organisation

The project organizational structure is reported in this person:



Contacts:

Project Leader

Name: Haryanto Natalius Liuwan

Phone: +31616855000

e-mail: hn.liuwan@student.fontys.nl

Address: Amalia van Anhaltstraat 26, 5616BH Eindhoven

Company Tutor

Name: Daniel Trivellato

Phone: +31642483416

e-mail: d.trivellato@secmatters.com

Address: Twinning Centre K.5.10

University Tutor

User Requirements Specification Document

Name: Casper Schellekens

Phone: 08850 73223

e-mail: c.schellekens@fontys.nl

Address: Fontys R1 4.47

Appendix B: Graduation Project Survey



University of Applied Sciences

Appendix A : Graduation Project Survey

HBO-ICT: English Stream

Data student:

Name student : Initials: H.N. Name: Liuwan
First name: Haryanto Natalius Studentnumber.: 2331039
Telephone:0683268545 E-mail.: hn.liuwan@student.fontys.nl

Data company:

Name company/organisation : SecurityMatters BV
Visiting adress : Mirastraat 93 7521 ZG Enschede
Company mentor : Initials: D. Name: Trivellato
Telephone:0642483416 E-mail.: daniel.trivellato@secmatters.com
Department/ position: Project Manager

Startdate Graduation project : February 2014

Duo Graduation project : No

Accepted by student: date: signature:

Accepted by company: date: signature:

Hand in date Graduation Project Survey:

User Requirements Specification Document

Approved by graduation project coordinator: yes/no

date:

signature:

Remarks _____ :

—

PLEASE SEND THIS FORM BY EMAIL TO THE INTERNSHIP COORDINATOR
IMMEDIATELY AFTER THE INTERNSHIP INTERVIEW HAS TAKEN PLACE.

Description of the graduation project:

1. Describe the problem analysis:

The company needs to do some research about the threats that may occur on Industrial Control System (ICS) environment and what is the solution for each threats. Because nowadays hackers think that attacking such system are more challenging, yet these systems are critical. It means that there must be some security methods applied to protect continuity of these systems.

2. Describe the graduation assignment.

Graduation assignment consist of:

- Studying literature of Industrial Control Systems, Cyberthreats, and Cybersecurity solutions. What are the threats that may occur on working Industrial Control System environment.
- Determining peripheral needed to build Industrial Control System environment.
- Setting up peripherals used in Industrial Control System environment.
- Simulating attack scenarios of the peripherals involved in Industrial Control System.
- Delivering report of vulnerabilities of devices used and possible attack scenarios that may happen on working Industrial Control Systems environment.

3. What is the research component of this assignment?

- Making a network topology and defining configurations to prevent hackers to have access to equipment directly.
- Finding a method to find vulnerabilities of the devices (PLC, etc.).
- Finding a method to find vulnerabilities of the network.

4. What are the methods and tools?

Methods are going to be the part of the research.

5. How and by whom will you be guided by the company?

Daniel Trivellato will accompany intern(s) as a project manager.

6. What fields of Study play an important factor in realizing the graduation assignment?

The fields of study that takes part in realizing the graduation assignment are networking, electronic hardware, security, monitoring, design, and realization.

OTHER DETAILS:

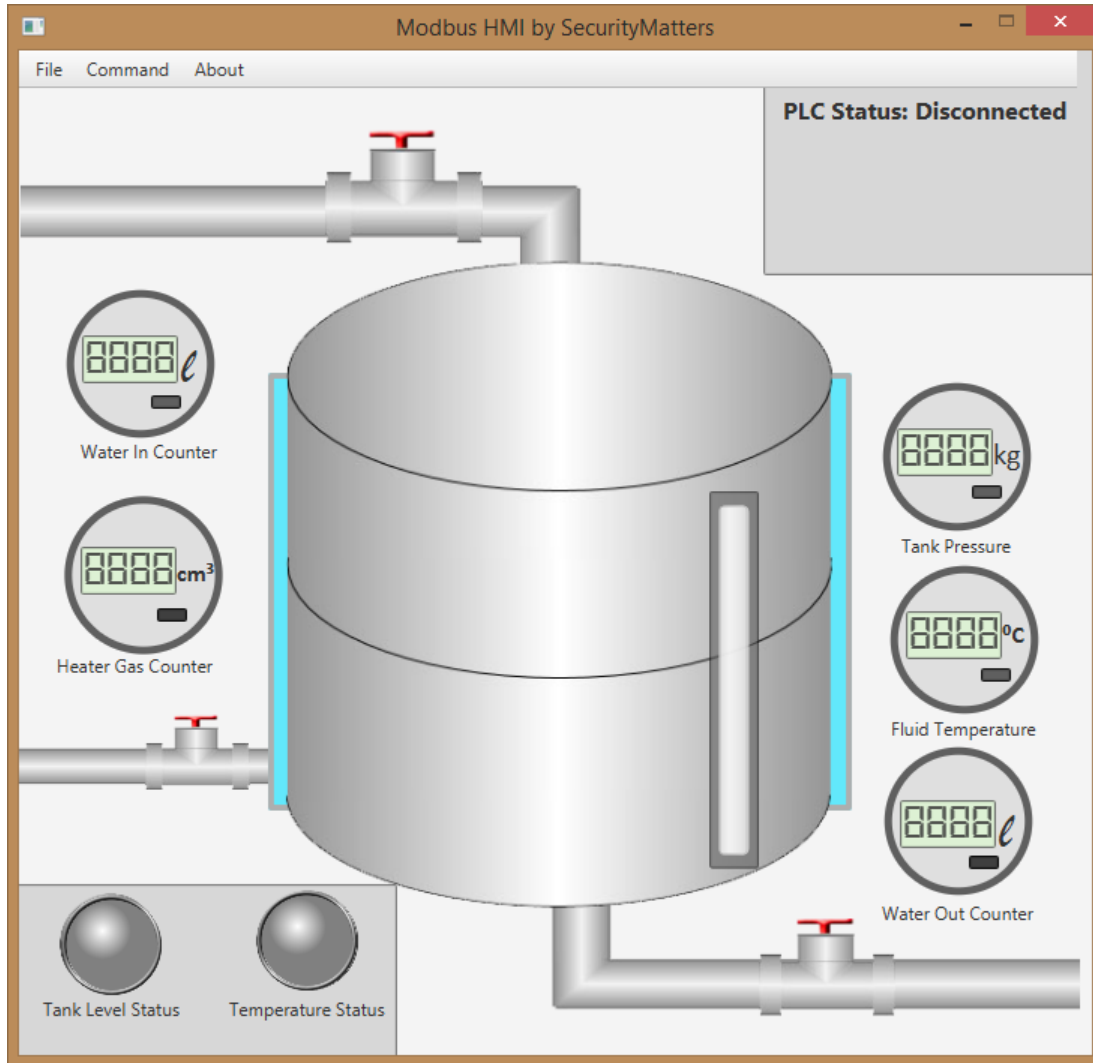
Preference university tutor:

1. Casper Schellekens
- 2.

Appendix C: User Requirement Specification Document Modbus HMI Application

USER REQUIREMENTS SPECIFICATION

Human-Machine Interface for Modbus Application Protocol



**SecurityMatters
2014**

Table of Contents

Chapter 1: Introduction	XXIII
1.1. Purpose	XXIII
1.2. Structure of the Document	XXIII
1.3. Definitions and Acronyms	XXIII
Chapter 2: Software Description	XXVI
2.1. Context	XXVI
2.2. Objectives	XXVI
2.3. Architecture	XXVI
2.4. Concept of Operations	XXVII
2.5. HMI Users and User Interface	XXVII
2.6. Methodology	XXVIII
2.7. Assumptions and Dependencies	XXVIII
Chapter 3: Software Requirements	XXIX
3.1. System Interface	XXIX
3.1.1. Communication Interface(s)	XXIX
3.1.2. Software Interface(s)	XXIX
3.2. Functional	XXIX
3.3. Graphical User Interface	XXIX
3.4. Performance Requirements	XXIX
3.5. Methodology	XXIX
3.6. Software System Attribute	XXIX
3.6.1. Reliability	XXIX
3.6.2. Maintainability	XXIX
3.6.3. Portability	XXX
Chapter 4: Use Cases	XXXI
Reference	XXXIII

Chapter 1: Introduction

1.1. Purpose

This User Requirements Specification describes the architecture and system design of the Modbus Human-Machine Interface software for SecurityMatters BV, as a part of the graduation project “Setting up an Industrial Control Systems Cybersecurity Simulation Laboratory”.

This document is intended to be read by:

- All responsible for further development of this application.
- Users of the product.

1.2. Structure of the Document

This document is structured as follows:

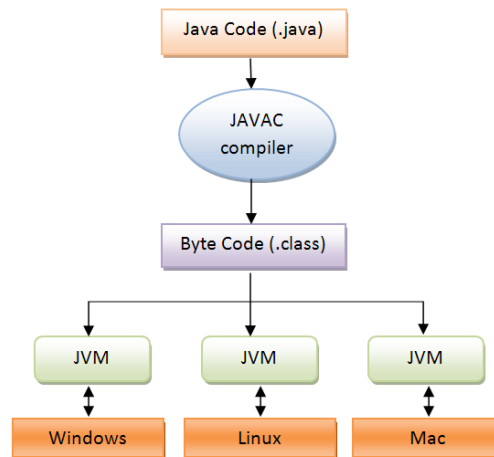
- Section 1.3 contains a glossary of pertinent terms and abbreviations.
- Chapter 2 provides a general description of the software and its intended use.
- Chapter 3 describes the requirements upon which the software was built.
- Chapter 4 contains some example use cases of the software.

1.3. Definitions and Acronyms

The concepts and acronyms used in this document are described as follows:

1.3.1. Java

Java is an object-oriented programming language developed by Sun Microsystems. Java is available for almost every device in market. The most fascinating feature from Java is its ability to run on multiple platforms as long as there is a Java Virtual Machine installed.

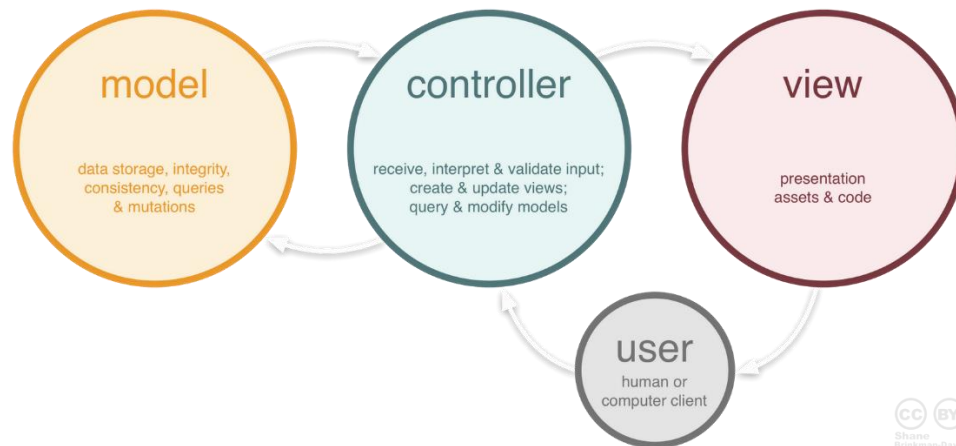


1.3.2. Object-Oriented Programming

Object-Oriented Programming is a programming paradigm that considers the main elements of a program as an “object” with specific functionalities. This paradigm is widespread as it promotes and facilitates software reusability. By using Object Oriented Programming, programmers do not have to rewrite the same code for the same function in their programs.

1.3.3. Model-View-Controller

Model-View-Controller is a programming pattern for Object-Oriented Programming that separates the application logic from the user interface. By splitting application logic and user interface, developers have more flexibility on expanding an application in the future. As the name suggests, the pattern involves the interaction of three (classes of) components: a Model, a View, and a Controller. The Model is in charge of representing and managing the information required by the system; the View renders the interface which enables the user to interact with the program; finally, the Controller contains the logic of the program and translates user/system input into the output to be rendered by the View.



1.3.4. Graphical User Interface

A Graphical User Interface is an interface that allows users to interact with a program using graphics.

1.3.5. JavaFX

JavaFX is a platform for creating Graphical User Interface applications through a Java library. JavaFX is platform independent like Java. JavaFX intends to replace the older “Swing” platform in the future because of its larger set of capabilities compared with the latter.

1.3.6. Scene Builder

Scene Builder is a component of JavaFX that allows to design Graphical User Interfaces without extensive knowledge of the underlying code. Scene Builder offers a drag and drop designer interface and will produce FXML files used by JavaFX.

1.3.7. Industrial Control System

Industrial Control System is a general term used to encompass control systems used in industrial environments, including Supervisory Control and Data Acquisition (SCADA) systems and as well as Programmable Logic Controllers (PLCs).

1.3.8. SCADA

SCADA is a way to monitor, control, and acquire data from remote controllers and field devices. SCADA consists of industrial hardware and software components.

1.3.9. PLC

A component which contains the logic to determine outputs as the programmed consequence of inputs from the sensors.

1.3.10. Human-Machine Interface

A software built to monitor and interact with PLCs. HMIs are usually custom built according to user needs and the needs of the application domain. Example of commercial HMIs include: WinCC by Siemens, RSView by Rockwell Automation, and DigiVis500/Network Managers by ABB.

1.3.11. Modbus

Modbus is an application protocol used in Industrial Control System or SCADA systems, originally developed by Modicon. Modbus uses Master-Slave paradigm as a way to communicate between devices. Modbus can be used both in serial and TCP/IP based communications (Modbus/TCP). In this documents we will focus on the latter.

1.3.12. Java Modbus Library

Java Modbus Library, known as Jamod, is an open source Java library which implements Modbus functionalities.

Chapter 2: Software Description

2.1. Context

SecurityMatters is a start-up company which focuses on network monitoring and intrusion detection systems. Their products aim at securing industrial processes from cyberthreats. As a company which focuses on security, SecurityMatters needs instruments for demonstrating the capabilities of its technology to its prospect. Currently, the company has a Human-Machine Interface communicating with real and simulated Programmable Logic Controllers using MMS and IEC 61850 protocols. Recently, the company decided to develop a Modbus-based HMI.

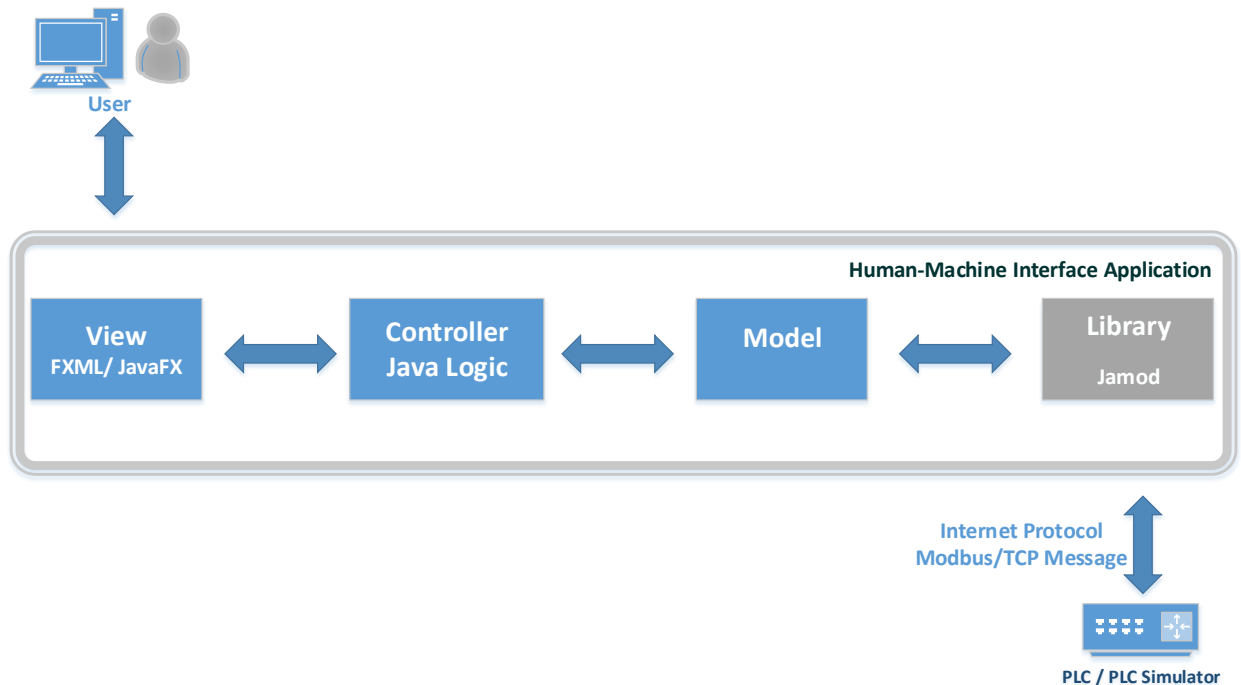
2.2. Objectives

The HMI for Modbus application protocol is built with the following goals in mind:

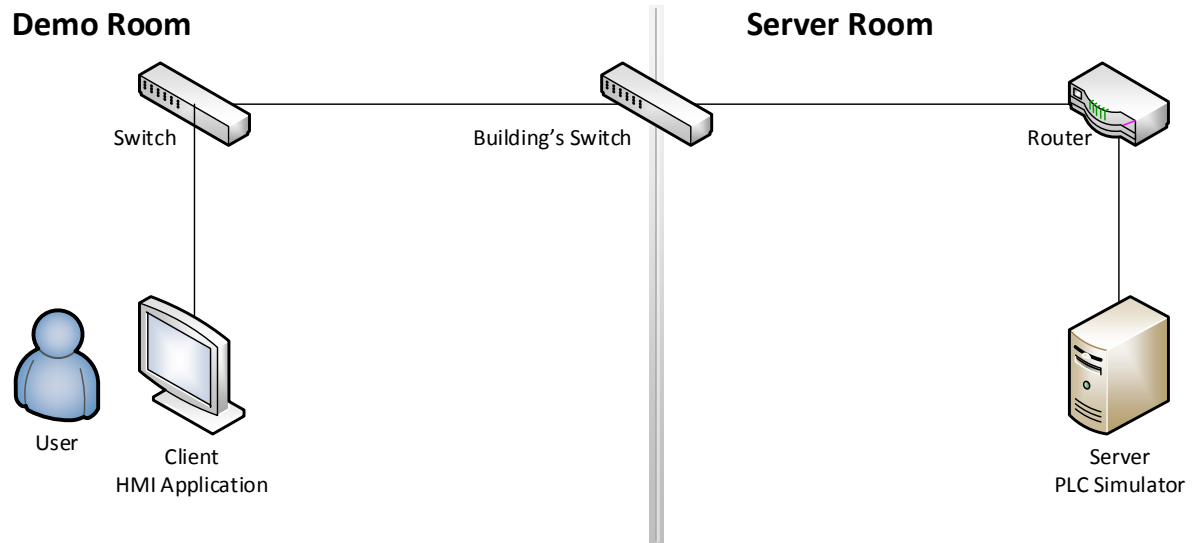
1. To simulate HMI-PLC communication using the Modbus protocol.
2. To show how the products of SecurityMatters react to those communications.
3. To test PLCs vulnerability on some attack scenarios.

2.3. Architecture

The HMI will be developed in Java using JavaFX and Jamod libraries. The Graphical User Interface will be designed using Scene Builder to provide an efficient and effective interface design. The overall software architecture is represented in the figure below.



The HMI enables users to communicate with a PLC simulator over the network. The HMI and PLC simulator will be deployed in the cyberlab of SecurityMatters to be available for use in demonstrations.



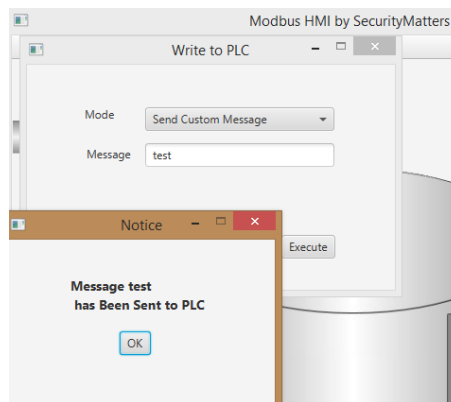
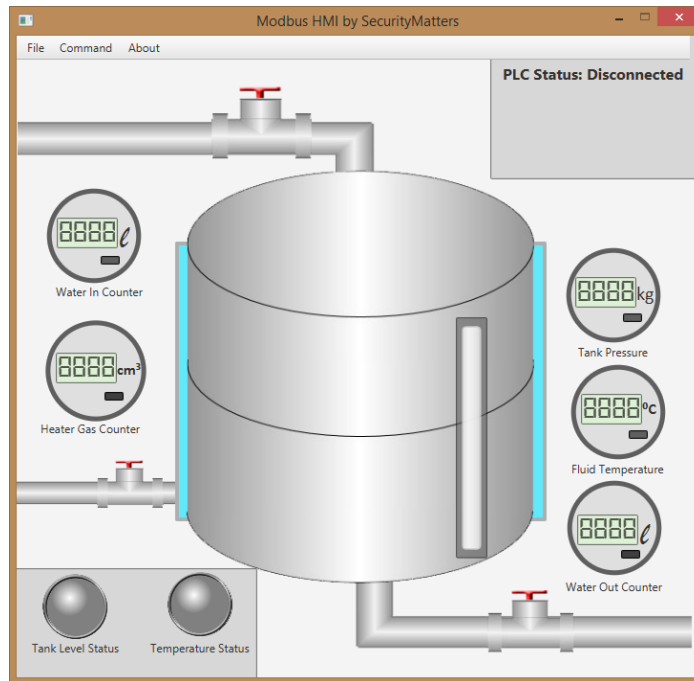
2.4. Concept of Operations

The application will be capable communicating using Modbus protocol, and triggering standard operational functions to a PLC, as well as sending custom commands to identify vulnerabilities of the device.

2.5. HMI Users and User Interface

The intended users of the HMI are SecurityMatters employees simulating an industrial process and demonstrating the capabilities of the company's products, and "testers" interested in finding vulnerabilities of the PLC.

The Graphical User Interface represents an industrial water boiler controller through which the user can modify the boiler operation. Below are some screenshots of the main HMI windows and an example of notification raised when sending custom commands to the PLC.



2.6. Methodology

The application's development is based on Object-Oriented Programming and Model-View-Controller programming pattern. Both patterns will allow easier programming progress for present and future improvement.

2.7. Assumptions and Dependencies

This application depends on a few software components. Such components need to be installed by the user on the system where the application should run. Those software dependencies are:

1. Java Runtime Environment.
2. Programmable Logic Controller:

A real or simulated PLC program must be running and connected to the same network as the HMI software.

Chapter 3: Software Requirements

3.1. System Interface

The system interface requirements are grouped into two groups:

3.1.1. Communication Interface(s)

The software must communicate using Modbus/TCP protocol.

3.1.2. Software Interface(s)

The software would have to provide interface such as methods on classes. With such interface, it will be possible to improve application in the future.

3.2. Functional

The main functionalities that must be supported by the application are:

1. Establish a connection with a PLC.
2. Read values from a PLC.
3. Write values to a PLC.
4. Send custom commands to a PLC.

The details of every function will be described in Chapter 4 along with the use cases.

3.3. Graphical User Interface

The requirements of the Graphical User Interface:

1. Ease of understanding and use, to make sure the audience of demonstration will understand the operations being executed.
2. Resemblance of the Graphical User Interface look and feel with that of existing industrial HMIs.
3. Feedback on every action carried out by the user, to make sure that the user understands that the action requested has been accomplished.

3.4. Performance Requirements

The HMI is built with no particular constraints on the software performance (e.g. speed, bandwidth).

3.5. Methodology

The application should be based on Java programming language as company's request.

3.6. Software System Attribute

3.6.1. Reliability

The application must be reliable and contain appropriate error-handling and input validation to prevent application crash on normal operations. In case connection dropped, the application must be able to reconnect automatically.

3.6.2. Maintainability

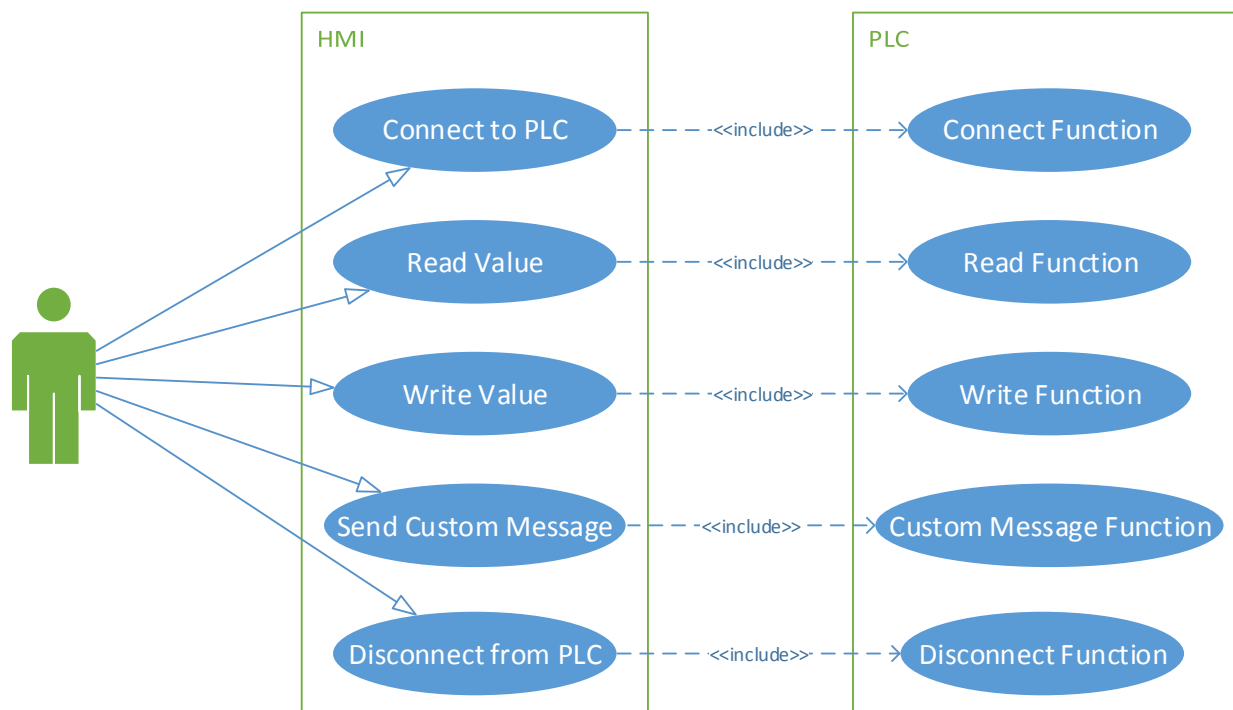
The application will come with design documentation and the code is written according to Object-Oriented Programming and Model-View-Controller to ensure maintainability and reusability.

3.6.3. Portability

The final deliverable is a program executable on any platform equipped with a Java Virtual Machine, to make sure that the application is easily portable to other systems.

Chapter 4: Use Cases

This chapter describes some use cases of the Modbus HMI and the effects of a user command on the software program. The use cases are described by the following diagram and table:



Use case ID	1
Use case name	Connect to PLC
Goal	Create a connection to PLC
Actor	User
Pre-condition	HMI is not connected yet
User actions	<ol style="list-style-type: none"> 1. User clicks Connect in menu bar. 2. User inputs PLC's IP address and port. 3. User clicks Connect. 4. Connect window will close and application will be connected with PLC.
Effect	<p>The HMI will be connected with the PLC.</p> <p>The HMI starts reading values from device.</p>
Note	If the PLC does not exist or input values are wrong, device might not be connected

Use case ID	2
Use case name	Read Value
Goal	Reads value from PLC
Actor	User
Pre-condition	HMI is connected with PLC
User actions	<ol style="list-style-type: none"> 1. User clicks Read in menu bar.

User Requirements Specification Document

	<ol style="list-style-type: none"> 2. User inputs address to read from PLC. 3. User clicks Read. 4. The application will show value on corresponding address.
Effect	HMI shows values on address desired
Note	If the PLC address entered is out of range an exception will be raised

Use case ID	3
Use case name	Write Value
Goal	Writes value to PLC
Actor	User
Pre-condition	HMI is connected with PLC
User actions	<ol style="list-style-type: none"> 1. User clicks Write in menu bar. 2. User inputs address and value to write to PLC. 3. User clicks Write 4. The application will show notification that the operation is done.
Effect	PLC value on specified address is changed
Note	If the PLC address entered is out of range an exception will be raised

Use case ID	4
Use case name	Send Custom Message
Goal	Sends custom message to PLC
Actor	User
Pre-condition	HMI is connected with PLC
User actions	<ol style="list-style-type: none"> 1. User clicks Test in menu bar. 2. User selects "Custom Message". 3. User specifies message to send to PLC. 4. User clicks Execute. 5. The application will show notification that the operation is done.
Effect	The response of the PLC cannot be predicted
Extensions	This option could crash the PLC

Use case ID	5
Use case name	Disconnect
Goal	Release connection from PLC
Actor	User
Pre-condition	HMI is connected with PLC
User actions	<ol style="list-style-type: none"> 1. User clicks Disconnect in menu bar. 2. The HMI will disconnect from PLC.
Effect	The PLC will be disconnected.
Extensions	After disconnecting, the user cannot do any functions on PLC

Reference

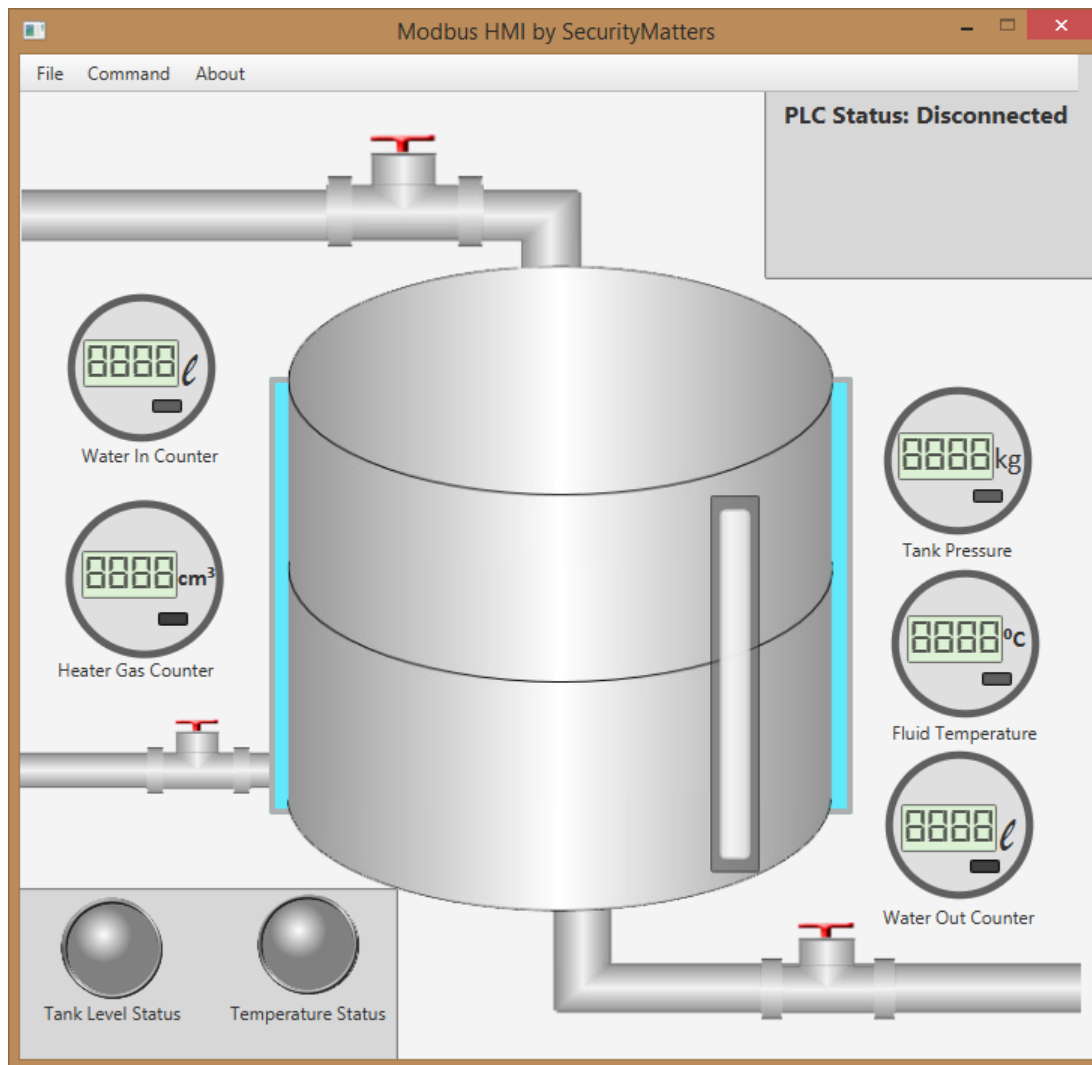
Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems (IDC Technology) by Gordon Clarke CP Eng BEng MBA, Deon Reynders Pr Eng BSc (ElecEng) (Hons) MBA.

Reference number	Source	URL Address	Description
1	Wikipedia	http://en.wikipedia.org/wiki/Modbus	Modbus protocol description
2	Modbus Organization site	http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf	Modbus protocol specification
3	Wikipedia	http://en.wikipedia.org/wiki/Iterative_and_incremental_development	Iterative development method
4	Wikipedia	http://en.wikipedia.org/wiki/Application_programming_interface	API
5	Wikipedia	http://en.wikipedia.org/wiki/Cloud_computing	Cloud
6	Wikipedia	http://en.wikipedia.org/wiki/Dynamic-link_library	DLL
7	Wikipedia	http://en.wikipedia.org/wiki/Industrial_Control_System	ICS
8	Wikipedia	http://en.wikipedia.org/wiki/Graphical_user_interface	GUI
9	Wikipedia	http://en.wikipedia.org/wiki/Human-machine_interface	HMI
10	SourceForge	http://jamod.sourceforge.net/	Jamod
11	Wikipedia	http://en.wikipedia.org/wiki/Java_(programming_language)	Java
12	Wikipedia	http://en.wikipedia.org/wiki/JavaFX	JavaFX
13	Wikipedia	http://en.wikipedia.org/wiki/Linux	Linux
14	Wikipedia	http://en.wikipedia.org/wiki/Model-view-controller	MVC
15	Wikipedia	http://en.wikipedia.org/wiki/Object-oriented_programming	OOP
16	Wikipedia	http://en.wikipedia.org/wiki/OS_X	OS X
17	Wikipedia	http://en.wikipedia.org/wiki/PLC	PLC
18	Wikipedia	http://en.wikipedia.org/wiki/SCADA	SCADA
19	Wikipedia	http://en.wikipedia.org/wiki/JavaFX	Scene Builder
20	Wikipedia	http://en.wikipedia.org/wiki/Microsoft_Windows	Windows

Appendix D: Software Design Document Modbus HMI Application

SOFTWARE DESIGN DOCUMENT

Human-Machine Interface for Modbus Application Protocol



SecurityMatters
2014

Table of Contents

Chapter 1: Introduction	XXXVIII
1.1. Purpose	XXXVIII
1.2. Structure of the Document	XXXVIII
1.3. Definitions and Acronyms	XXXVIII
1.3.1. Java	XXXVIII
1.3.2. Object-Oriented Programming	XXXIX
1.3.3. Model-View-Controller	XXXIX
1.3.4. Graphical User Interface	XXXIX
1.3.5. JavaFX	XXXIX
1.3.6. Scene Builder	XXXIX
1.3.7. Industrial Control System	XL
1.3.8. SCADA	XL
1.3.9. PLC	XL
1.3.10. Human-Machine Interface	XL
1.3.11. Modbus	XL
1.3.12. Java Modbus Library	XL
Chapter 2: Software Overview	XLI
2.1. Context	XLI
2.2. Functionalities	XLI
2.3. Design	XLI
Chapter 3: Software Design	XLII
3.1. Software Design	XLII
3.2. Assumptions and Dependencies	XLIII
3.3. Design Rationale	XLIII
Chapter 4: Graphical User Interface	XLIV
4.1. Overview of User Interface	XLIV
4.2. Screen Images	XLIV
4.3. Screen Objects and Actions	XLVII
Chapter 5: Class Design	XLVIII
Chapter 6: Sequence Diagrams	LIII
6.1. Connect to PLC	LIII
6.2. Disconnect from PLC	LIII
6.3. Auto Mode Activation	LIV

6.4. Manual Mode Activation LIV

6.5. Read Discrete Input or Coil Status from PLC..... LV

6.6. Write Single or Multiple Discrete Input Status to PLC..... LV

6.7. Read Input or Holding Registers Value from PLC LVI

6.8. Write Holding Register Value to PLC LVI

6.9. Send Custom Command to PLC.....LVII

Chapter 7: Development Methodology LVIII

Reference LX

Appendix LXI

Appendix A: Address Table..... LXI

Chapter 1: Introduction

1.1. Purpose

This design document describes the architecture and system design of the Modbus Human-Machine Interface software built for SecurityMatters BV, as a part of the graduation project “Setting up an Industrial Control Systems Cybersecurity Simulation Laboratory”.

This document is intended to be read by:

- All responsible for further development of this application.

1.2. Structure of the Document

This document is structured as follows:

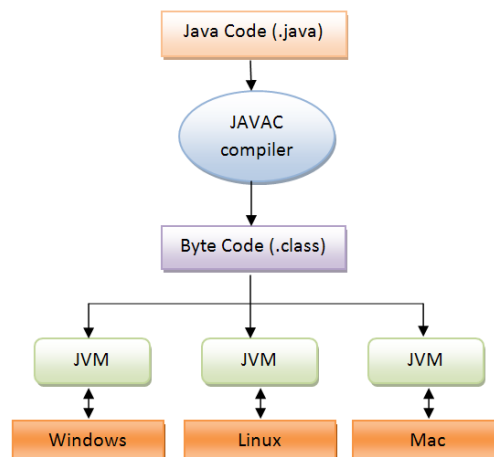
- Section 1.3 contains a glossary of pertinent terms and abbreviations.
- Chapter 2 provides a general overview of the software.
- Chapter 3 describes the software design.
- Chapter 4 describes the user interface design.
- Chapter 5 describes the software classes.
- Chapter 6 provides a sequence diagram of the software.
- Chapter 7 describes the development methodology adopted.

1.3. Definitions and Acronyms

The concepts and acronyms used in this document are described as follows:

1.3.1. Java

Java is an object-oriented programming language developed by Sun Microsystems. Java is available for almost every device in market. The most fascinating feature from Java is its ability to run on multiple platforms as long as there is a Java Virtual Machine installed.

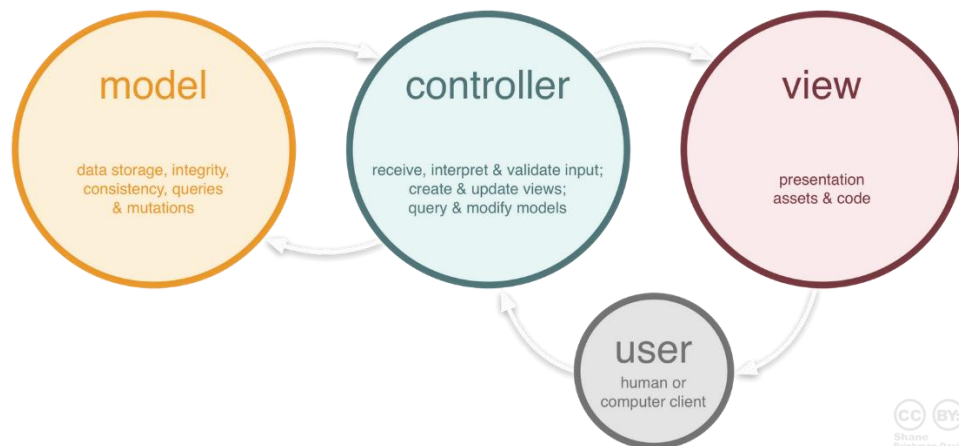


1.3.2. Object-Oriented Programming

Object-Oriented Programming is a programming paradigm that considers the main elements of a program as an “object” with specific functionalities. This paradigm is widespread as it promotes and facilitates software reusability. By using Object Oriented Programming, programmers do not have to rewrite the same code for the same function in their programs.

1.3.3. Model-View-Controller

Model-View-Controller is a programming pattern for Object-Oriented Programming that separates the application logic from the Graphical User Interface. By splitting application logic and user interface, developers have more flexibility on expanding an application in the future. As the name suggests, the pattern involves the interaction of three (classes of) components: a Model, a View, and a Controller. The Model is in charge of representing and managing the information required by the system; the View renders the interface which enables the user to interact with the program; finally, the Controller contains the logic of the program and translates user/system input into the output to be rendered by the View.



1.3.4. Graphical User Interface

A Graphical User Interface is an interface that allows users to interact with a program using graphics.

1.3.5. JavaFX

JavaFX is a platform for creating Graphical User Interface applications through a Java library. JavaFX is platform independent like Java. JavaFX intends to replace the older “Swing” platform in the future because of its larger set of capabilities compared with the latter.

1.3.6. Scene Builder

Scene Builder is a component of JavaFX that allows to design Graphical User Interfaces without extensive knowledge of the underlying code. Scene Builder

offers a drag and drop designer interface and will produce FXML files used by JavaFX.

1.3.7. Industrial Control System

Industrial Control System is a general term used to encompass control systems used in industrial environments, including Supervisory Control and Data Acquisition (SCADA) systems and as well as Programmable Logic Controllers (PLCs).

1.3.8. SCADA

SCADA is a way to monitor, control, and acquire data from remote controllers and field devices. SCADA consists of industrial hardware and software components.

1.3.9. PLC

A component which contains the logic to determine outputs as the programmed consequence of inputs from the sensors.

1.3.10. Human-Machine Interface

A software built to monitor and interact with PLCs. HMIs are usually custom built according to user needs and the needs of the application domain. Example of commercial HMIs include: WinCC by Siemens, RSView by Rockwell Automation, and DigiVis500/Network Managers by ABB.

1.3.11. Modbus

Modbus is an application protocol used in Industrial Control System or SCADA systems, originally developed by Modicon. Modbus uses Master-Slave paradigm as a way to communicate between devices. Modbus can be used both in serial and TCP/IP based communications (Modbus/TCP). In this documents we will focus on the latter.

1.3.12. Java Modbus Library

Java Modbus Library, known as Jamod, is an open source Java library which implements Modbus functionalities.

Chapter 2: Software Overview

2.1. Context

SecurityMatters is a start-up company which focuses on network monitoring and intrusion detection systems. Their products aim at securing industrial processes from cyberthreats. As a company which focuses on security, SecurityMatters needs instruments for demonstrating the capabilities of its technology to its prospect. Currently, the company has a Human-Machine Interface communicating with real and simulated PLCs using MMS and IEC 61850 protocols.

The software described in this document has been built to expand the company's demonstration instruments to support Modbus/TCP protocol as well.

2.2. Functionalities

The HMI for Modbus application protocol is built with the following functionalities in mind:

1. As a Human-Machine Interface application that will be able to communicate using Modbus protocol.
2. As a demo instrument for Modbus protocol.
3. As a PLC vulnerability tester on some attack scenarios.

2.3. Design

The HMI was designed with Object-Oriented Programming and Model-View-Controller pattern in mind to ensure modularity and reusability. More details on the software design will be provided in coming chapters.

Chapter 3: Software Design

3.1. Software Design

The HMI was developed using Java as programming language, and using JavaFX as Graphical User Interface platform. The HMI's development is based on Object-Oriented Programming and Model-View-Controller programming pattern. Both patterns will allow easier programming progress for present and future development.

The implementation using Object-Oriented Programming pattern is mainly on the Modbus application protocol classes, which implement functions from Jamod library to allow easier access to Modbus functions.

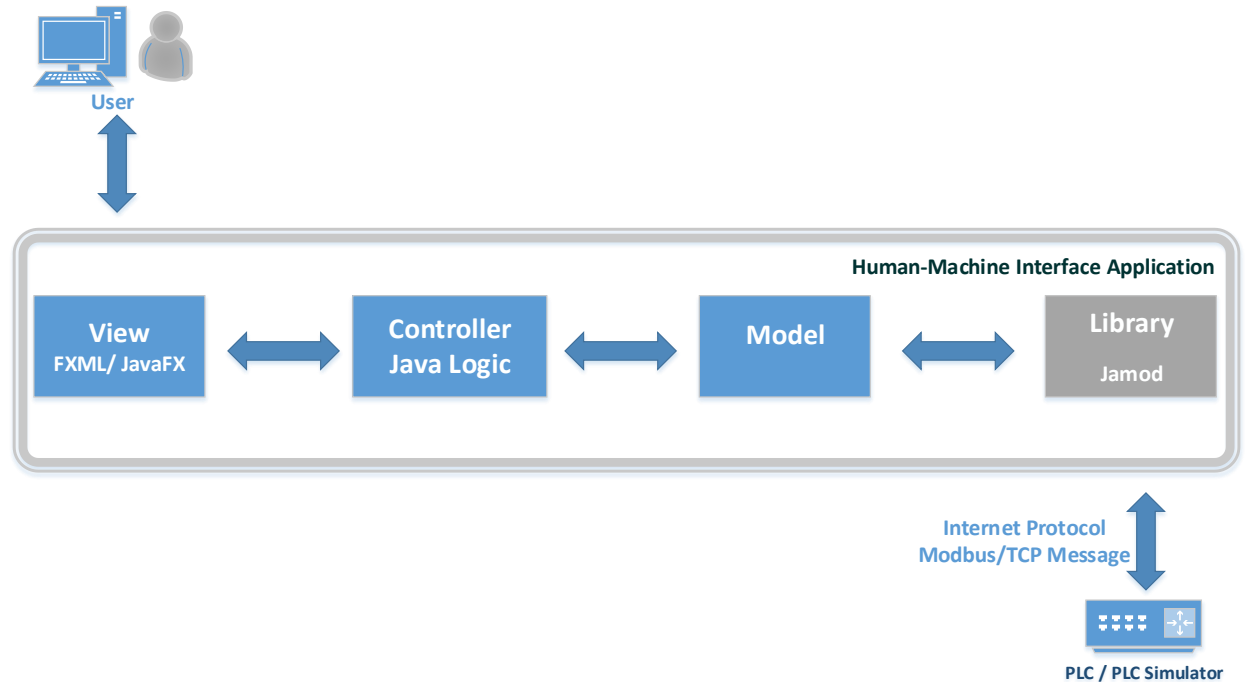
The implementation of Model-View-Controller pattern applied to this application resulted in Controllers and Model written in Java programming language, and the View part written in JavaFX language.

To design the Graphical User Interface, Scene Builder, a component of JavaFX was used to ensure effective and efficient design process. Moreover, this is the newest Java platform for drawing Graphical User Interface resulting in a future-proof and modern interface that can run seamlessly on other machines that support Java.

Below is the diagram of the software components from the Model-View-Controller point of view. The JavaFX view component will be detailed in Chapter 4, while Modbus model and controller classes will be detailed in Chapter 5.



There is no data storage part in this application, because the function of the HMI is to collect and send real-time data from PLCs using Jamod library via Modbus/TCP protocol. Overall, the process consists mainly of querying and translating data from a PLC which is connected to the Human-Machine Interface application.



3.2. Assumptions and Dependencies

This application depends on a few software components. Such components need to be installed by the user on the system where the HMI should run. Those software dependencies are:

1. Java Runtime Environment.
2. Programmable Logic Controller:

A real or simulated PLC program must be running and connected to the same network as the HMI software.

3.3. Design Rationale

The design of the HMI aims at providing users with a comprehensive yet easy to understand tool for simulating an industrial process and demonstrating the capabilities of the company's products. Few more things are taken into considerations as well:

1. Cross platform interoperability.
2. Availability of libraries and APIs.
3. Expandability in the future.

During the development phase of this application, different combinations of components were tested, such as C++ programming language with Libmodbus library, as well as several libraries in java, including modbus4j and jamod2. At the end, the Java-jamod combination was selected because of requirements and feasibility of development itself.

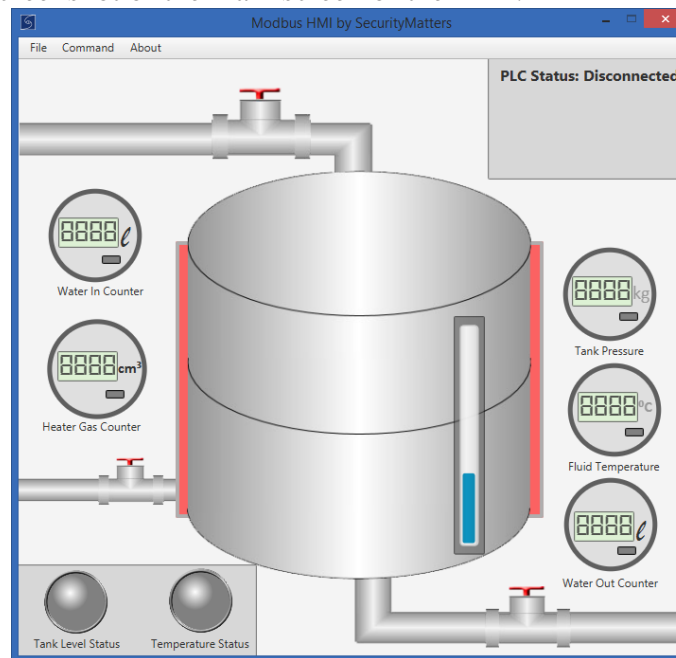
Chapter 4: Graphical User Interface

4.1. Overview of User Interface

The Graphical User Interface represents an industrial water boiler controller through which the user can modify the boiler operation, and is designed with two main concepts in mind:

1. Ease of understanding and use, to make sure the audience of demonstrations will understand the operations being executed.
2. Resemblance of the Graphical User Interface look and feel with that of existing industrial HMIs.

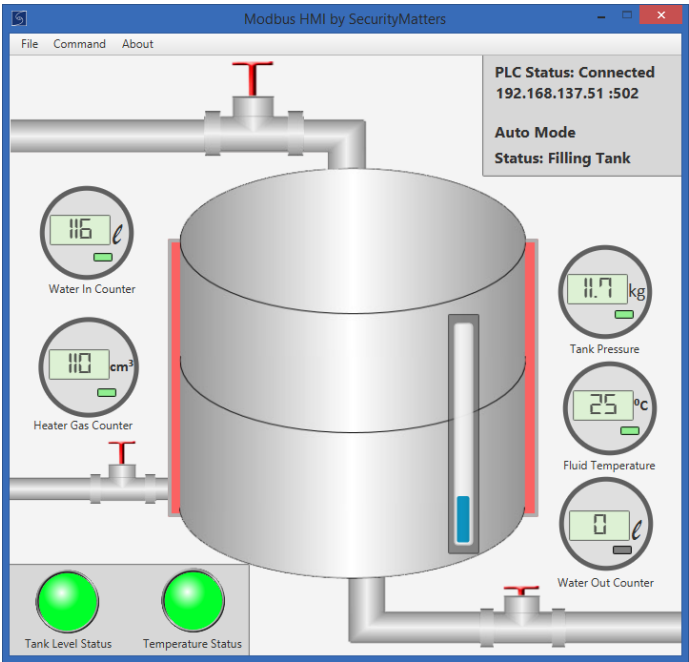
Below is the screenshot of the main screen of the HMI:



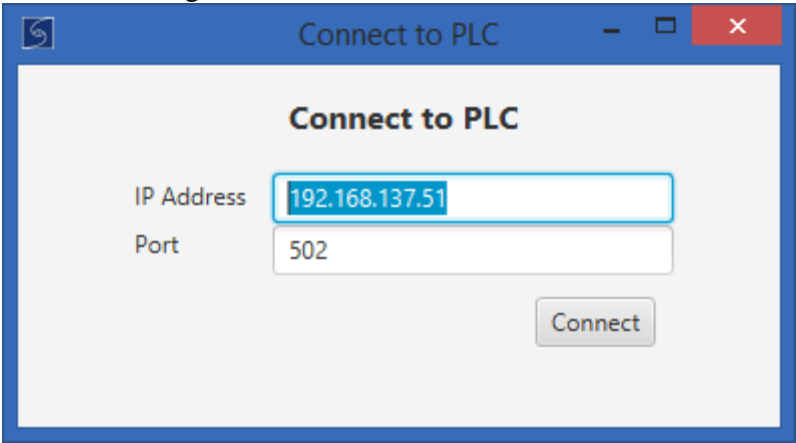
4.2. Screen Images

Below are some screenshots of the main HMI windows and a short description of the functionality of each window:

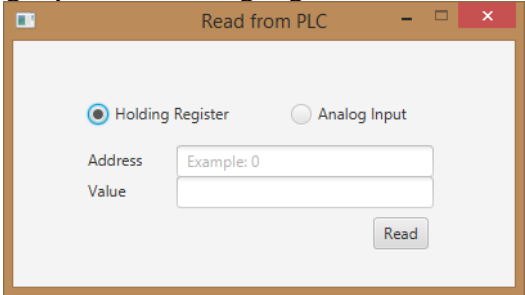
Main working window in case PLC is connected.



The window for connecting to PLC.

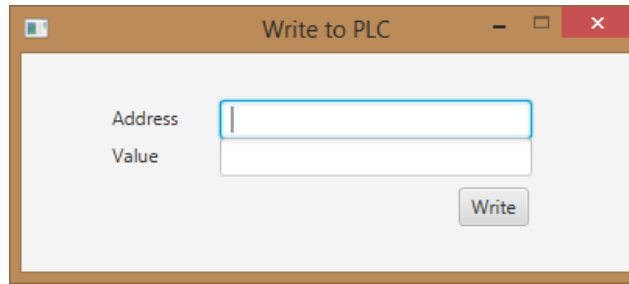


The window for reading input and holding register values from PLC.



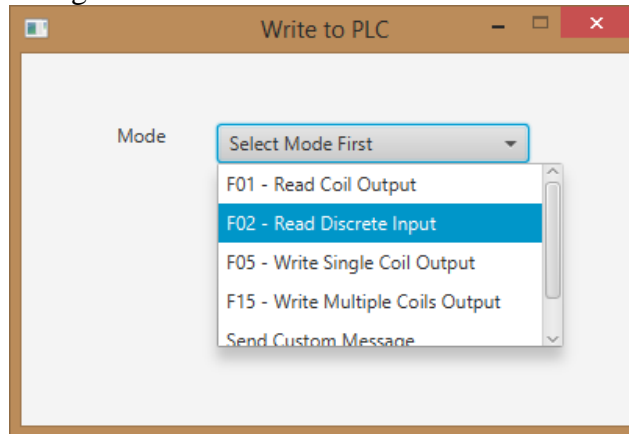
The window for writing holding register values to PLC.

User Requirements Specification Document



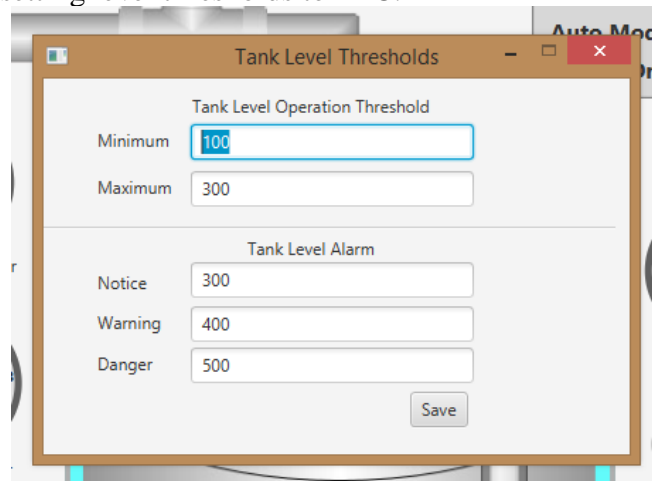
A screenshot of a software window titled "Write to PLC". It features two input fields: "Address" and "Value". The "Address" field is currently selected with a blue border. A "Write" button is located at the bottom right of the window.

The window for sending test commands to PLC.



A screenshot of the "Write to PLC" window with a dropdown menu open for the "Mode" field. The dropdown menu lists the following options: "Select Mode First", "F01 - Read Coil Output", "F02 - Read Discrete Input" (highlighted in blue), "F05 - Write Single Coil Output", "F15 - Write Multiple Coils Output", and "Send Custom Message".

The window for setting level thresholds to PLC.

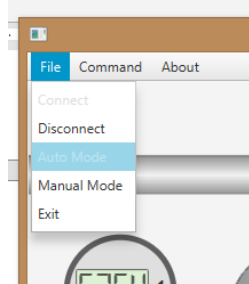


A screenshot of a software window titled "Tank Level Thresholds". It contains two sections: "Tank Level Operation Threshold" and "Tank Level Alarm". The "Tank Level Operation Threshold" section has input fields for "Minimum" (set to 100) and "Maximum" (set to 300). The "Tank Level Alarm" section has input fields for "Notice" (set to 300), "Warning" (set to 400), and "Danger" (set to 500). A "Save" button is located at the bottom right of the window.

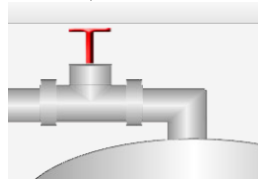
4.3. Screen Objects and Actions

The interface objects with which the user can interact are designed to be intuitive, and to every action carried out by the user the interface will provide feedback, to make sure that the user understands that the action requested has been accomplished.

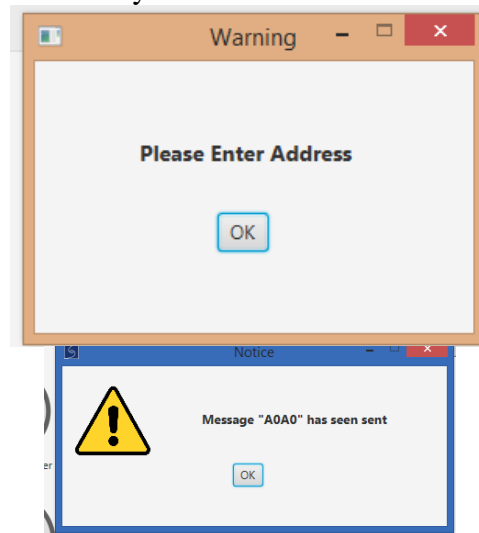
Main menu, used to access main functions of the HMI.



Valves, which the user can open/close (i.e. send on/off value to PLC) upon clicking.

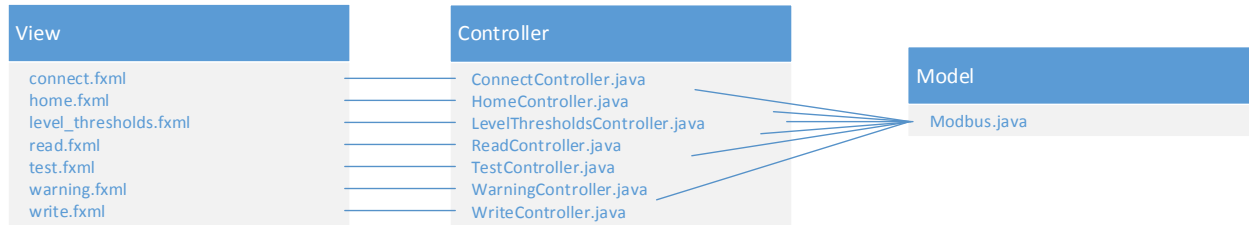


Warnings and notifications to users in case the values sent to the PLC are incomplete or incorrect, or have been correctly received.



Chapter 5: Class Design

The operation of the HMI is achieved by the interaction of a number of Java classes. The Model component consists of the Modbus class for translating user requests into commands for PLC and translating values from PLC to Graphical User Interface values. The controller controls each of view part of the HMI. Below are the classes involved and descriptions of the class:



Class name: Modbus	
Description: The Modbus class is the interface with the Jamod library. This class is also responsible for preparing values needed for every Jamod transaction	
Methods (operations)	Method Description
connect(String ipIn) connect(String ipIn, String portIn)	A method to initiate connection with a PLC device.
customWrite(byte[] Value)	A method to write data in bytes to PLC.
disconnect()	A method to disconnect a PLC device.
f01ReadCoil(String readAddress) f01ReadCoils(String readAddress, String countValue)	A method to perform read coil(s) operation on PLC.
f02ReadDiscreteInput(String readAddress) f02ReadDiscreteInputs(String readAddress, String countValue)	A method to perform read discrete input(s) operation on PLC.
f03ReadHoldingRegister(String readAddress) f03ReadHoldingRegisters(String readAddress, String countValue)	A method to perform read holding register(s) operation on PLC.
f04ReadInputRegister(String readAddress) f04ReadInputRegisters(String readAddress, String countValue)	A method to perform read input register(s) operation on PLC.
f05WriteCoil(String writeAddress, Boolean value)	A method to perform write coil operation on PLC.
f06WriteHoldingRegister(String writeAddress, Boolean value)	A method to perform write register operation on PLC.
f15WriteCoils(String writeAddress, String value)	A method to perform write coils operation on PLC.

f16WriteHoldingRegister(String ReadAddress, String[] Value)	A method to perform write registers operation on PLC.
getIp()	A method that returns IP address written on Modbus class.
getPort()	A method that returns port number written on Modbus class.
warning(String val1)	A method to show warning window along with specified message.

Class name: ConnectController	
Description: The ConnectController class is the controller of connect.fxml and reconnect.fxml view, which is used get parameters for connecting and reconnecting a PLC.	
Methods (operations)	Method Description
connect(String ipIn) connect(String ipIn, String portIn)	A method to send needed parameters and command for connect() method in Modbus class. Whenever connection succeeded, parameters used will be saved.
setStage(HomeController t21, Stage temp1)	A method to initialize connect window and receiving needed parameters from previous controller. This method also reads a last succeeded connection parameter file if that exists.
setStageDirect(HomeController t21, Stage temp1)	A method to initialize reconnect window and timer countdown.
timerIteration()	A method to show countdown timer in UI and defines action needed to reconnect.

Class name: HomeController	
Description: The HomeController class is the controller of home.fxml view, which controls all control and notifications for user.	
Methods (operations)	Method Description
automode()	A method to turn on auto mode for PLC.
connected()	A method to enable Graphical User Interface elements for PLC functions
disconnect()	A method to enable Graphical User Interface elements for PLC functions.
initialize(URL url, ResourceBundle rb)	A method to initialize controller, including timers on controller.
manualmode()	A method to turn off auto mode for PLC.
pauseread()	A method to turn off automatic value reading.
read()	A method to read corresponding values from PLC.
readOnce()	A method to read corresponding values from PLC once.
reconnect()	A method to initialize reconnect function in case connection dropped.

User Requirements Specification Document

resumeRead()	A method to turn on automatic value reading.
setStage(Stage temp)	A method to initialize the main window, including disable Graphical User Interface elements until PLC connected.
showConnect()	A method to show connect window.
showRead()	A method to show read window.
showTank()	A method to show tank level thresholds window.
showTemp()	A method to show temperature thresholds window.
showTest()	A method to show test window.
showWrite()	A method to show write window.
shutdown()	A method to destroy running threads for the application in the system.
tankLampOff()	A method to set tank level alarm image black.
tankLampOn(Image im)	A method to set tank level alarm image with specified image.
tempLampOff()	A method to set temperature level alarm image black.
tempLampOn(Image im)	A method to set temperature level alarm image with specified image.
toggleHeater()	A method to turn heater on or off.
toggleInValve()	A method to turn input valve on or off.
toggleOutValve()	A method to turn input valve on or on.

Class name: LevelThresholdsController	
Description: The LevelThresholdsController class is the controller of level_thresholds.fxml view, which controls both temperature and tank level threshold values.	
Methods (operations)	Method Description
comparevalue(String[] values, String offsetAddr)	A method to verify values written with requested value.
condition(String[] values)	A method to verify whether values on text field are correct.
save(String[] values, String offsetAddr)	A method to send needed parameters and command for saving the values on PLC.
setStage(HomeController t2, Stage temp, String alarmLimitIn, String offsetAddrIn)	A method to initialize window and receiving needed parameters from previous controller.

Class name: ReadController	
Description: The ReadController class is the controller of read.fxml view, which controls read input and holding register functions.	
Methods (operations)	Method Description
f03ReadHoldingRegister(String addr)	A method to initiate read a value from PLC.

User Requirements Specification Document

f04ReadInputRegister(String addr)	
setStage(HomeController t21, Stage temp)	A method to initialize window and receiving needed parameters from previous controller.
setValue(String val)	A method to set Graphical User Interface elements according to value received.

Class name: TestController	
Description: The TestController class is the controller of test.fxml view, which controls test interface.	
Methods (operations)	Method Description
custom(String valText)	A method to send custom message to PLC.
f01ReadCoil(String addr)	A method to send needed parameters and command to read coil action on PLC.
f02ReadDiscreteInput(String addr)	A method to send needed parameters and command to read discrete input action on PLC.
f03ReadHoldingRegister(String addr0)	A method to send needed parameters and command to read holding register action on PLC.
f04ReadInputRegister(String addr)	A method to send needed parameters and command to read input register action on PLC.
f05WriteCoil(String addr, Boolean val)	A method to send needed parameters and command to write coil action on PLC.
f06WriteHoldingRegister(String addr, String val)	A method to send needed parameters and command to write holding register action on PLC.
f15WriteCoils(String addr, String val)	A method to send needed parameters and command to write coils action on PLC.
f16WriteHoldingRegisters(String offsetAddr, String[] values)	A method to send needed parameters and command to write holding registers action on PLC.
hexStringToByteArray(String s)	A method to change string received to byte of hexadecimal data type.
setRadio(Boolean tempBool)	A method to set radio button according value received from PLC.
setStage(HomeController t2, Stage temp)	A method to initialize window and receiving needed parameters from previous controller.

Class name: WarningController	
Description: The WarningController class is the controller of warning.fxml view, which controls all notifications for user.	
Methods (operations)	Method Description
setStage(String text1)	A method to set warning message.

Class name: WriteController	
Description: The WriteController class is the controller of write.fxml view, which controls all notifications for user.	
Methods (operations)	Method Description

User Requirements Specification Document

comparevalue(Stromg add, String val, String com)	A method to verify values written with requested value.
setStage(HomeController t2, Stage temp)	A method to initialize window and receiving needed parameters from previous controller.
f06WriteHoldingRegister(String add, String val)	A method to send needed parameters and command for saving the values on PLC.

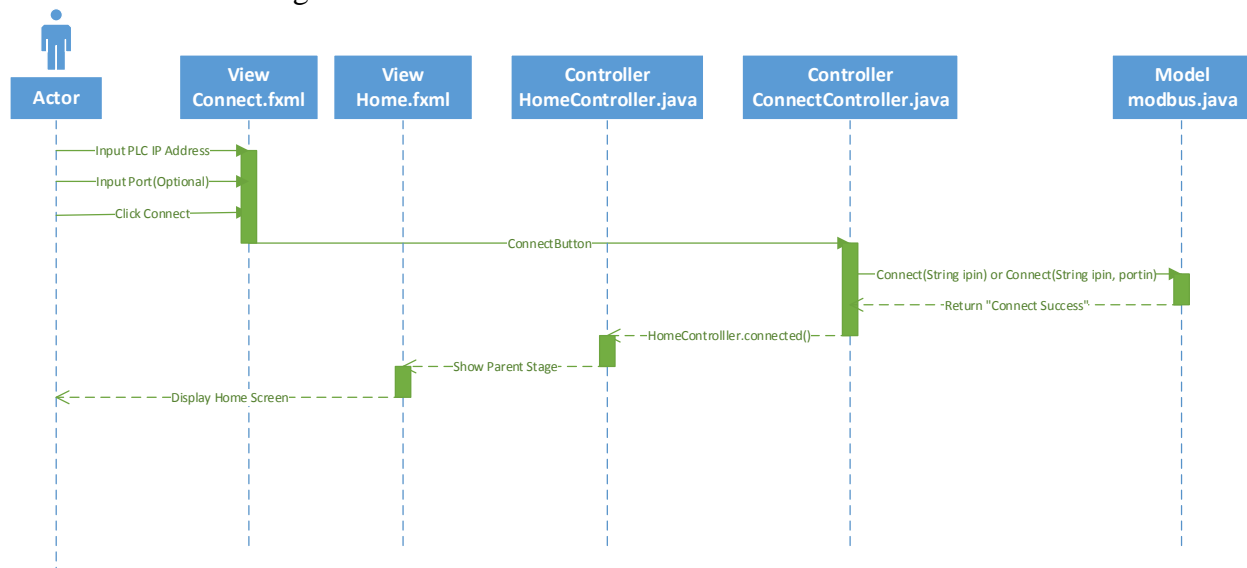
Chapter 6: Sequence Diagrams

Below are the sequence diagrams that illustrate the operation of the application in response to user actions.

6.1. Connect to PLC

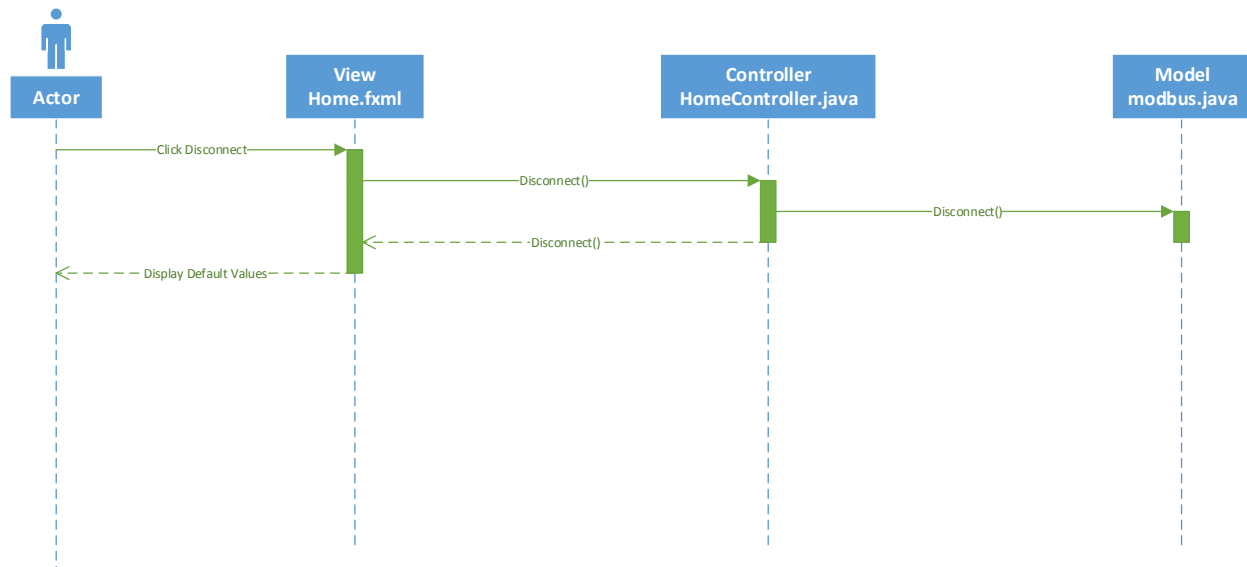
To connect to a PLC, the user will have to specify the IP address and port (optional) of the PLC. Clicking connect will send those parameters to the ConnectController class, and ConnectController will pass those values to the Modbus class. Then, modbus.java model will send needed commands to the PLC and reads the PLC's response. When the connection is established, the Modbus class will return "Connect Success" message as a reply to ConnectController.java.

Upon receiving success confirmation from the Modbus class, ConnectController will close Connect.fxml view and calls connected() method from call HomeController to start reading values from PLC.



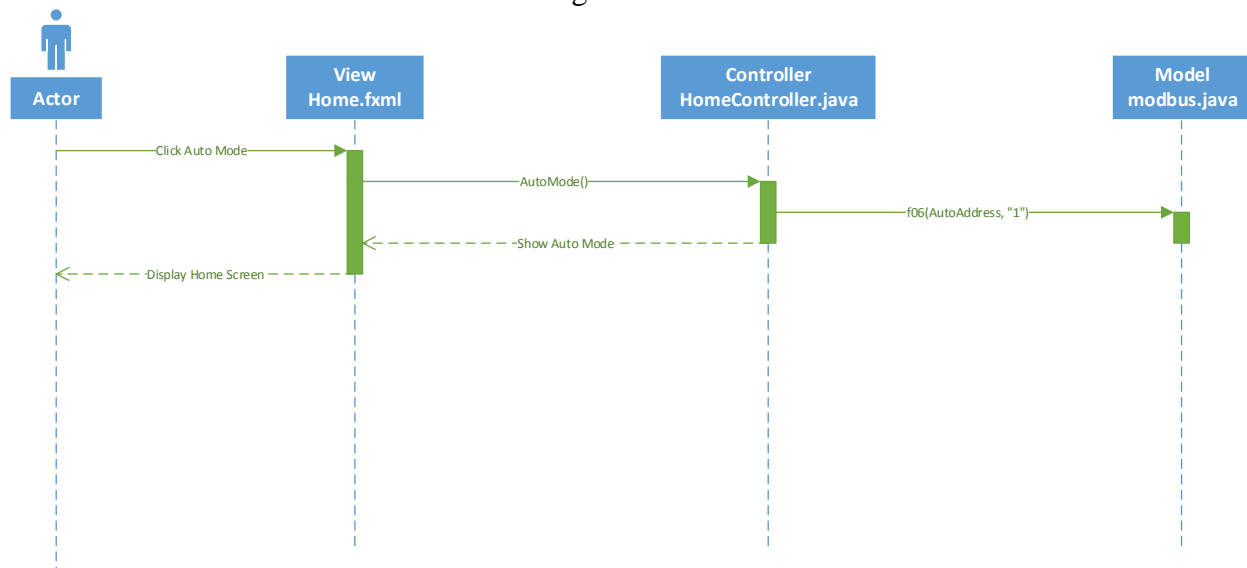
6.2. Disconnect from PLC

To disconnect the PLC, the user can click the disconnect button on the menu. The disconnect() method will be initiated in class HomeController and class modbus. Since disconnection does not have return value, HomeController will directly disable the interaction of the Graphical User Interface with the PLC.



6.3. Auto Mode Activation

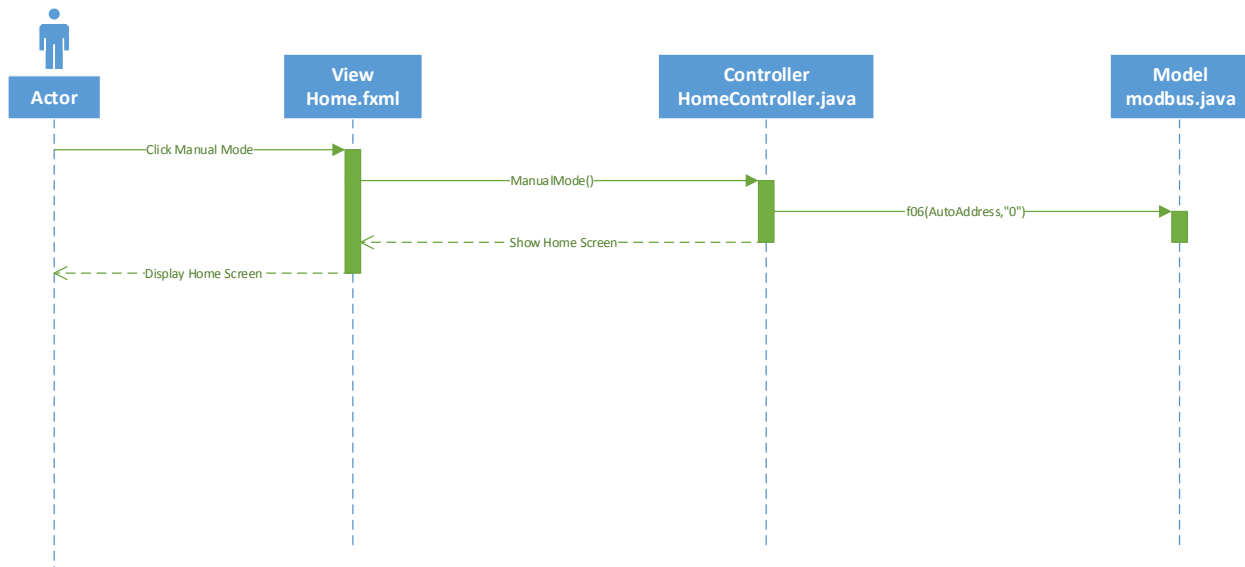
The application features an “auto mode” that simulates the industrial process by automatically issuing commands to the PLC. To activate auto mode, the user can click the auto mode menu item. The autoMode() method will be initiated in class HomeController, then method f06(AutoAddress,”1”) will be invoked from class Modbus to turn on auto mode flag on the PLC.



6.4. Manual Mode Activation

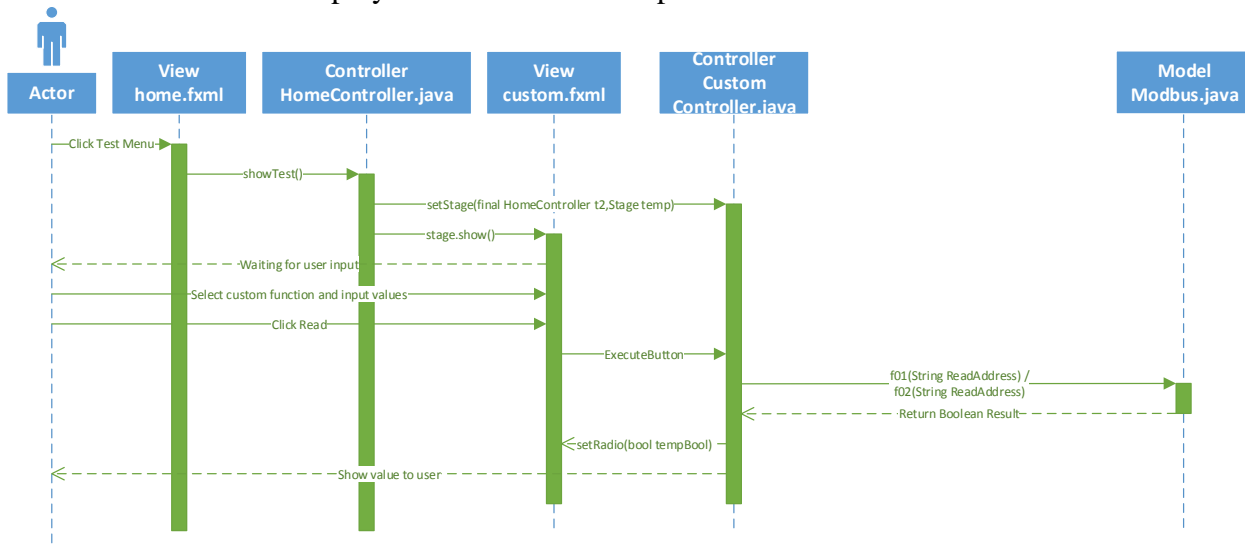
To activate manual mode, the user can click manual mode on menu. The manualMode() method will be initiated in class HomeController, then f06(AutoAddress,”0”) method and parameters will be sent to Modbus class to turn off auto mode flag on the PLC.

User Requirements Specification Document



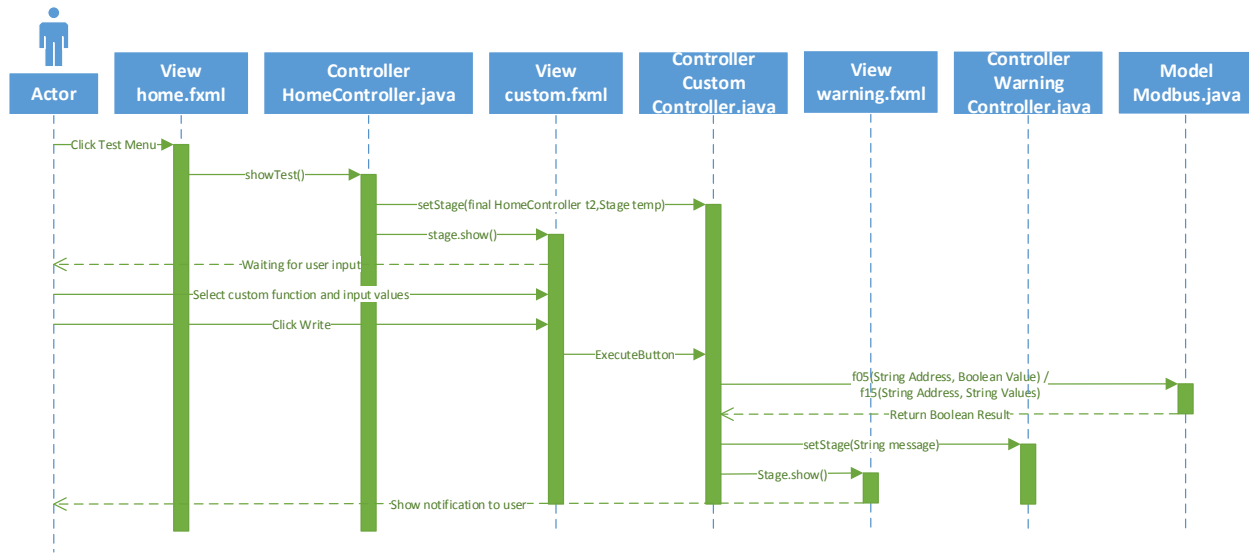
6.5. Read Discrete Input or Coil Status from PLC

To read discrete input or coil status from the PLC, the user must select test menu from home screen and then the user can input the address that she wants to read and click the read button. The parameter will be sent to CustomController class and forwarded to class modbus. The result value is returned to the controller which will initiate the function to display the result on the Graphical User Interface.



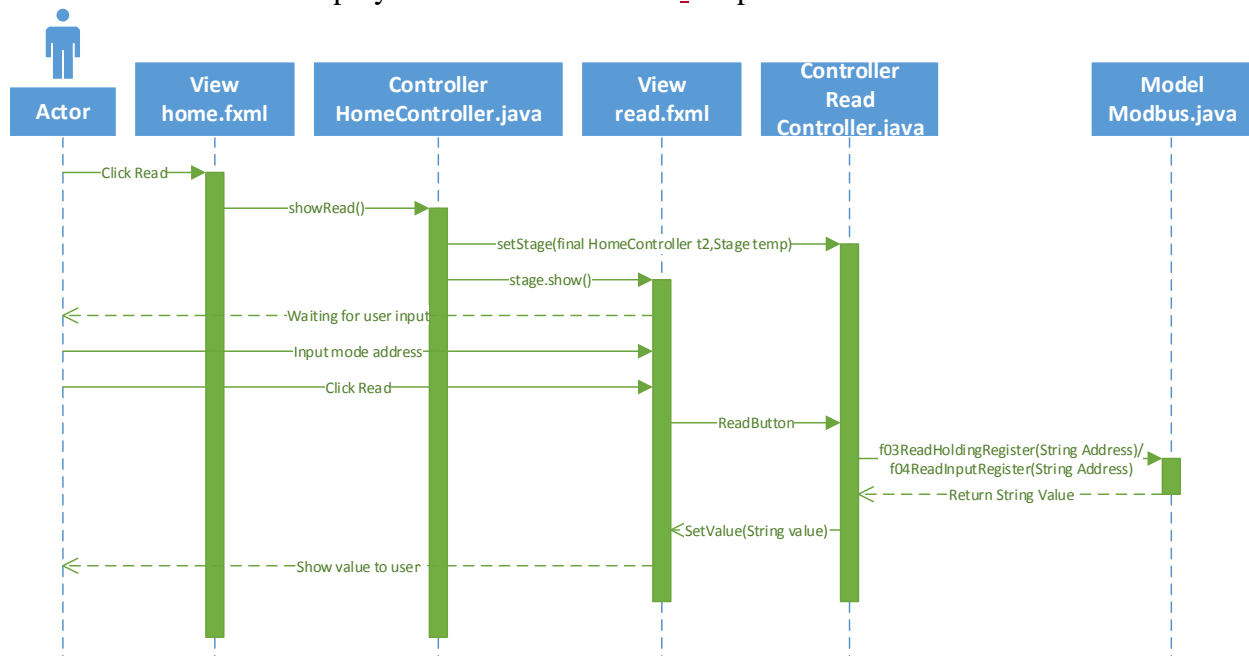
6.6. Write Single or Multiple Discrete Input Status to PLC

To write single or multiple discrete input status to the PLC, the user must select test menu from home screen first and then the user can input the destination address along with status desired and click the write button. The parameter will be sent to class CustomController and forwarded to Modbus class. Then a notification message will be shown to user.



6.7. Read Input or Holding Registers Value from PLC

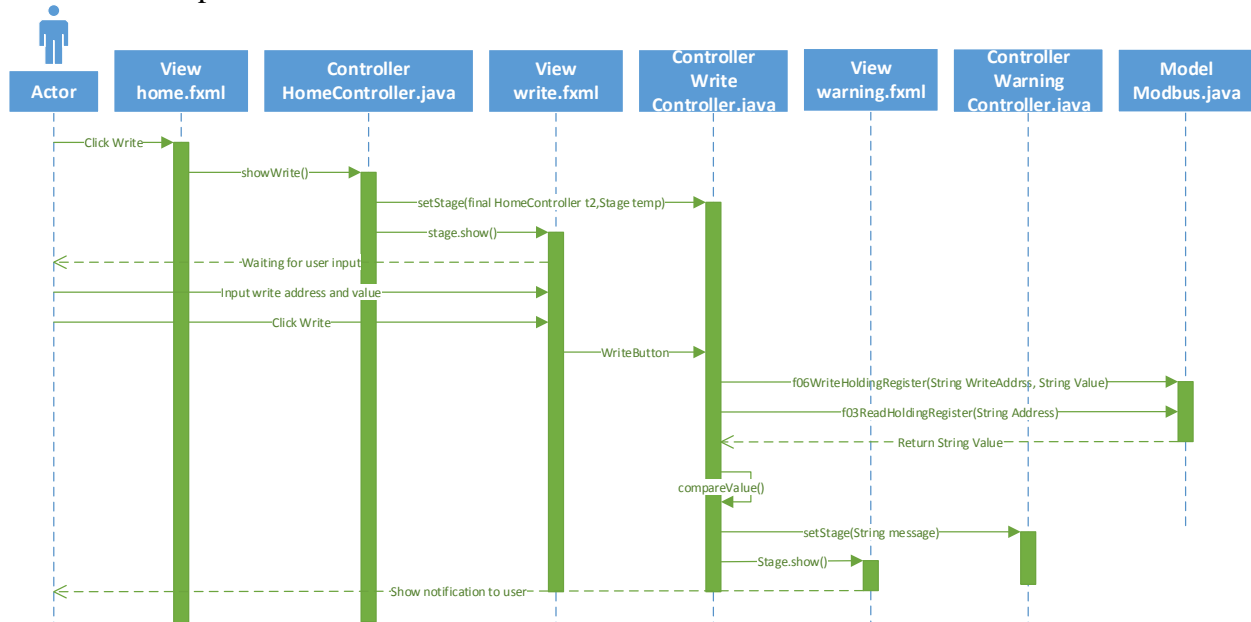
To read input or holding registers value from the PLC, the user must select read menu and then can input the mode and address that the user wants to read and click the read button. The parameters will be sent to class `ReadController` and forwarded to `Modbus` class. The result value is returned to the controller, then the controller will initiate the function to display the result value on the `Graphical User Interface`.



6.8. Write Holding Register Value to PLC

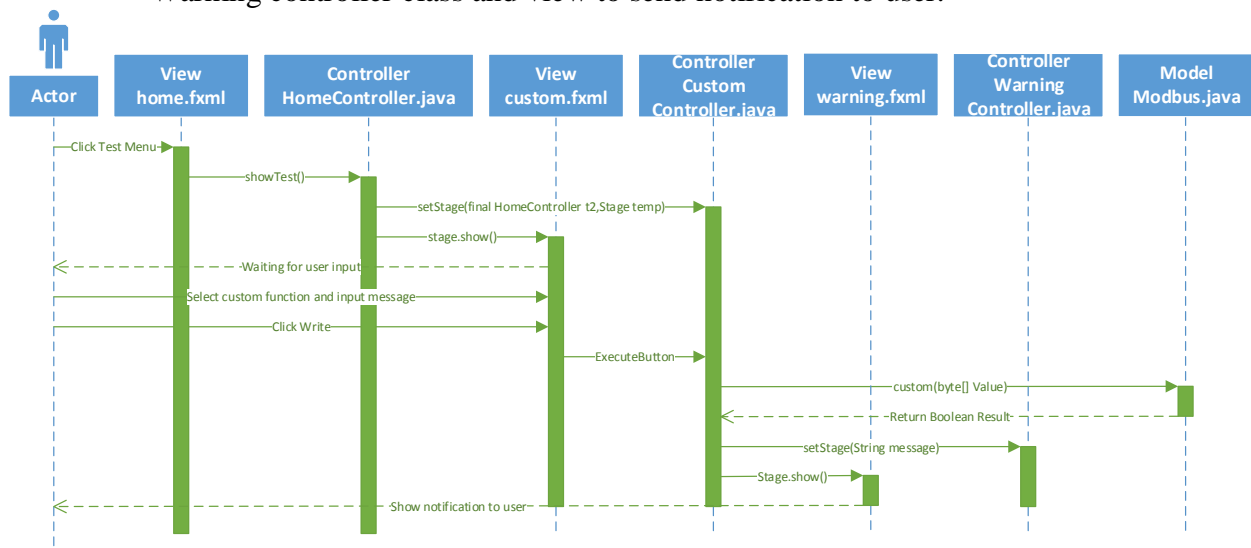
To write holding register values to the PLC, the user must select write menu from home screen and then can input the destination address along with value desired and click the write button. The parameter will be sent to class `WriteController` and forwarded to

Modbus class. Right after the command is sent, a read holding register function is also sent to verify the value written. If the written value and specified value match, the Graphical User Interface will show a success notification.



6.9. Send Custom Command to PLC

To send custom commands to the PLC, the user must select test menu from home screen first, and then user can input the hexadecimal message and click the write button. The parameter will be sent to class CustomController and forwarded to Modbus class and Modbus class will return the result to CustomController. CustomController will invoke Warning controller class and view to send notification to user.

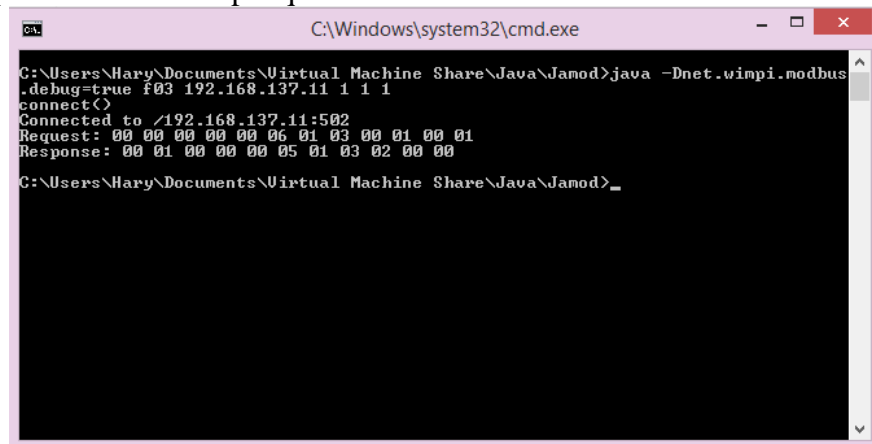


Chapter 7: Development Methodology

The application has been developed using an iterative development method. More precisely, the implementation is achieved by the iteration of the following steps:

1. Defining Requirements
2. Analysis and Design
3. Implementation
4. Testing
5. Evaluation

The first iteration concerned the implementation of jamod2 functionalities in a Java command line-based application to do simple queries to a PLC.

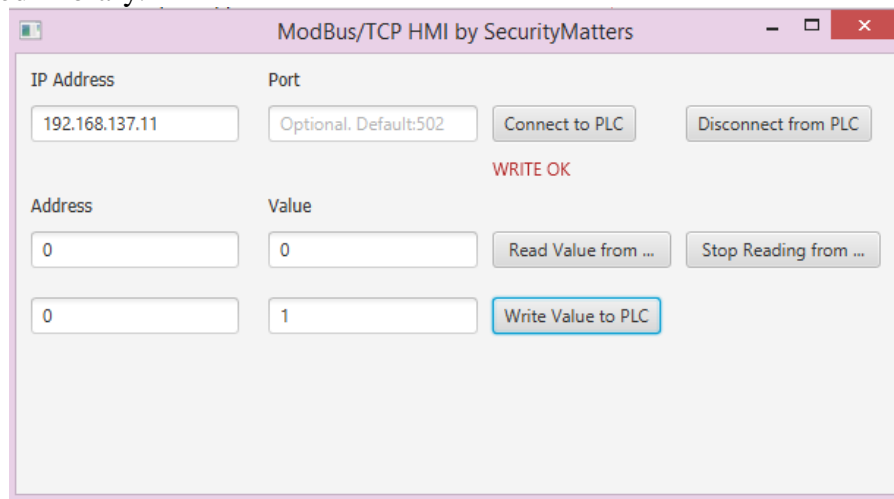


```
C:\Windows\system32\cmd.exe

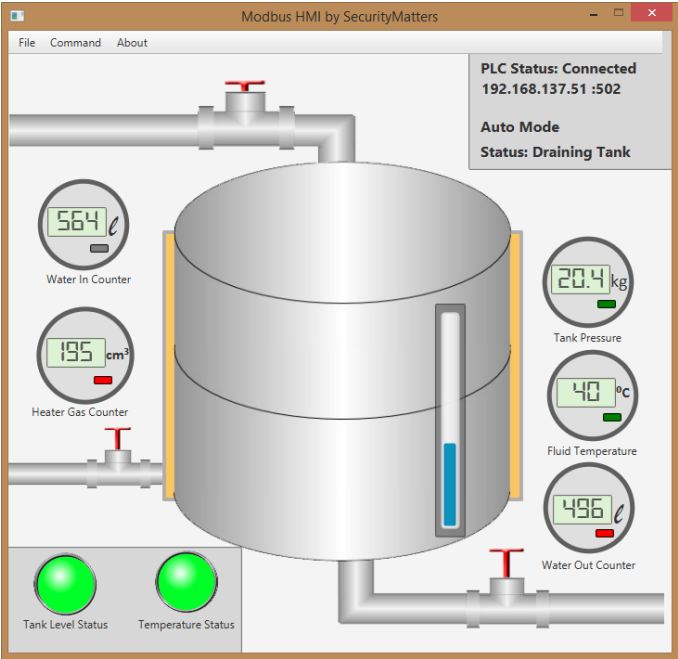
C:\Users\Hary\Documents\Virtual Machine Share\Java\Jamod>java -Dnet.wimpi.modbus
.debug=true f03 192.168.137.11 1 1 1
connect()
Connected to 192.168.137.11:502
Request: 00 00 00 00 00 06 01 03 00 01 00 01
Response: 00 01 00 00 00 05 01 03 02 00 00

C:\Users\Hary\Documents\Virtual Machine Share\Java\Jamod>_
```

The second iteration saw the implementation of a simple Graphical User Interface interfacing with the jamod2 library.



Finally, in the third iteration the final water boiler environment along with full jamod2 library functionalities was implemented.



Reference

Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems (IDC Technology) by Gordon Clarke CP Eng BEng MBA, Deon Reynders Pr Eng BSc (ElecEng) (Hons) MBA.

Reference number	Source	URL Address	Description
1	Wikipedia	http://en.wikipedia.org/wiki/Modbus	Modbus protocol description
2	Modbus Organization site	http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf	Modbus protocol specification
3	Wikipedia	http://en.wikipedia.org/wiki/Iterative_and_incremental_development	Iterative development method
4	Wikipedia	http://en.wikipedia.org/wiki/Application_programming_interface	API
5	Wikipedia	http://en.wikipedia.org/wiki/Cloud_computing	Cloud
6	Wikipedia	http://en.wikipedia.org/wiki/Dynamic-link_library	DLL
7	Wikipedia	http://en.wikipedia.org/wiki/Industrial_Control_System	ICS
8	Wikipedia	http://en.wikipedia.org/wiki/Graphical_user_interface	GUI
9	Wikipedia	http://en.wikipedia.org/wiki/Human-machine_interface	HMI
10	SourceForge	http://jamod.sourceforge.net/	Jamod
11	Wikipedia	http://en.wikipedia.org/wiki/Java_(programming_language)	Java
12	Wikipedia	http://en.wikipedia.org/wiki/JavaFX	JavaFX
13	Wikipedia	http://en.wikipedia.org/wiki/Linux	Linux
14	Wikipedia	http://en.wikipedia.org/wiki/Model-view-controller	MVC
15	Wikipedia	http://en.wikipedia.org/wiki/Object-oriented_programming	OOP
16	Wikipedia	http://en.wikipedia.org/wiki/OS_X	OS X
17	Wikipedia	http://en.wikipedia.org/wiki/PLC	PLC
18	Wikipedia	http://en.wikipedia.org/wiki/SCADA	SCADA
19	Wikipedia	http://en.wikipedia.org/wiki/JavaFX	Scene Builder
20	Wikipedia	http://en.wikipedia.org/wiki/Microsoft_Windows	Windows

Appendix

Appendix A: Address Table

During the development of Modbus Human-Machine Interface Software, some changes were done on the content of the PLC Simulator addresses. The new content of the PLC addresses is as follows:

Analog Inputs

tankLevelAddr = 10

fluidTemperatureAddr = 11

Holding Registers

heatingCounterAddr = 0

coolingCounterAddr = 1

fillingMeterAddr = 10

heatingMeterAddr = 11

drainMeterAddr = 12

tankLevelHighAddr = 20

tankLevelHighHighAddr = 21

tankLevelHighHighHighAddr = 22

tankLevelMinAddr = 23

tankLevelMaxAddr = 24

fluidTemperatureHighAddr = 30

fluidTemperatureHighHighAddr = 31

fluidTemperatureHighHighHighAddr = 32

fluidTemperatureMinAddr = 33

fluidTemperatureMaxAddr = 34

inputValveOpenAddr = 40

heaterOnAddr = 41

outputValveOpenAddr = 42

autoModeAddr = 50

initAddr = 60