University College of Southeast Norway

www.usn.no

FMH606 Master's Thesis 2017

Systems and Control Engineering

IoT and Modbus protocol development

Alexander Zhang Gjerseth

Faculty of Technology, Natural Sciences and Maritime Sciences

Campus Porsgrunn

University College of Southeast Norway

www.usn.no

Course: FMH606 Master's Thesis, 2017

Title: IoT and Modbus protocol development

Number of pages: 56

Keywords: Internet of Things, Modbus, Software development, Laboratory Assignment

Student: Alexander Zhang Gjerseth

Supervisor: Nils-Olav Skeie, Ole Magnus Brastein (co-supervisor)

External partner: None

Availability: Open

Approved for archiving: _____

(supervisor signature)

Summary:

An Internet of Things (IoT) device has been developed by the "SMART" research group at University College of Southeast Norway. It is intended for laboratory experiments. It uses Modbus TCP on top of the TCP/IP stack for communication. The main tasks were to both develop a set of objects for Modbus communication using Object-Oriented Analysis and Design (OOAD) methods and also an application based on the objects to communicate with the IoT Device. In addition, a laboratory assignment on Modbus, IoT and network using the application developed should be proposed.

By using Unified process as development process, Unified Modeling Language and design patterns, the application and software objects was developed in one solution and by using the OOAD approach. A proposal to a laboratory assignment was also created using the application and software objects as part of the assignment.

Modbus has been implemented in the software as a IoT protocol for communication with the device. The developed software has been tested and is working according to the requirements collected. The laboratory assignment proposed will give the students knowledge in technology relevant for the future.

Preface

This is a master thesis conducted in the fourth semester in master program System and Control Engineering at University College of Southeast Norway (HSN).

I would like to thank my supervisor Nils-Olav Skeie, co-supervisor Ole Magnus Brastein and Per Kristian Fylkesnes for their time and valuable help throughout this project.

The source code for the software developed in this thesis has been sent to the supervisor.

Porsgrunn, 15.05.2017

Alexander Zhang Gjerseth

Contents

1 Introduction

Modbus is an old, well known and widely used communication protocol in the industry. One of the goals for this thesis is to find out if it is suitable to use as an Internet of Things(IoT) Protocol.

1.1 Background

IoT has become an important field in the technological advances the world has seen the last years. Both for consumer use and as Industrial IoT (IIoT) which is IoT for the industry. IoT often are devices intended for the consumer like smart thermostats and lightbulbs, and are made for convenience. While IIoT for example can be used in high risk environments to improve safety.

The background for this thesis is an IoT box created by the “SMART” research group at HSN. The intention behind the box is to have a lab exercises where the students gets hands on experience with communication protocols, IoT, Modbus and network.

1.2 Objectives

The following objectives are gathered from the thesis description found in Appendix A.

a) Give an overview of the status for IoT protocols.

b) Make a description of the intranet with a router, switch, DHCP server and computers. Include documentation for setup of the DHCP server,

c) Discuss the usage of the Modbus TCP protocol as an IoT protocol,

d) Develop a set of software objects in C# for Modbus TCP client/server communication using the Modbus TCP protocol specification and OOAD methods,

e) Develop an application, based on the Modbus TCP communication objects, for communicating with the IoT device,

f) Evaluate how a Network Analyzer software like the Wireshark application can be used to debug and trace the Modbus messages between the application and the IoT device,

g) Include a module in the application for testing of the error conditions in the Modbus TCP protocol,

h) Propose a laboratory assignment, on both BSc. and MSc. Programmes (IA and IIA) that can use the application, Wireshark, and the objects, with documentation. Setup of the router with the DHCP server for the LAN, IoT system, and Modbus TCP should be the learning objectives.

i) Include a report module for documenting the time period, traffic of messages and range of Modbus registers used. The preferred format of the report should be PDF. Evaluate C# components for the PDF format.

1.3 Related Work

When it comes to using Modbus as an IoT protocol there exists some cases, but it is not commonly used. Intel has an example where Modbus is used, but only as a local protocol connecting to a gateway that uses another application layer protocol for communication over the internet.[1] This is similar to the work for this thesis, but for this thesis Modbus is the only application layer protocol that will be used for communication between the application and the IoT device.

Another aspect of this thesis is creating a laboratory assignment focusing on the IoT Device, Modbus and IoT. When it comes to laboratory products that focuses on IoT, ARM, one of the world's leading manufacturers of semiconductors, has developed a IoT Education kit as part of their ARM University program.[2] This education kit contains both hardware boards, software licenses and teaching material, both for lectures and lab exercises. But most of the focus when using this kit is on learning the ARM mbed IoT device platform and Android SDK.

On HSN, there exists a laboratory assignment on IIoT in systems and control laboratory course on Industrial IT and Automation masters.[3] The assignment uses either a raspberry pi or a myRio from National Instruments. But this assignment focuses on using a web services for IoT commutation, while the assignment proposed in this thesis will use Modbus TCP as a IoT protocol for direct communication with the IoT device.

1.4 Report Structure

This report starts with an introduction and background for this thesis. Followed by the system description chapter describing the different parts that make up the system. The next chapter is about the methods used. Furthermore there is a chapter describing the analysis phase of the development. Next there is a chapter where the results are presented. Followed by a discussion chapter and then a conclusion chapter.

2 System Description

This chapter will present the IoT device, describe the intranet used to connect to the IoT device. Before discussing the laboratory assignment that will be proposed.

IoT is a term describing the interconnection between devices via the internet. There currently exists no universal standard for IoT, but a lot of different groups are working on creating an open standard for the IoT.[4] But for this report, IoT is defined as a network of devices using the internet to communicate.

There are two main ways to connect to a IoT device, either use an Application Programming Interface (API) or a protocol on top of the Transmission Control Protocol/Internet Protocol (TCP/IP) stack. The TCP/IP stack is the set of protocols used on the internet. The API is often used to connect to middleware, and then the middleware connects to the device.[5] While a protocol on top of the TCP/IP stack would ensure direct connection to the IoT device. The task description found in Appendix A states that for this project a protocol on top of the TCP/IP should be used.

To better describe IoT protocols and the TCP/IP stack, the concept of the OSI model is introduced. The OSI model is a conceptual framework that can be used to divide network into seven layers.[6] This separation makes it easier to tie protocols up to their purpose in a network. The seven layers of the OSI model is Physical, Data Link, Network, Transport, Session, Presentation and Application. For the physical and data link layer the TCP/IP suite has a range of different protocols, but for this thesis the Ethernet protocol is required. For layer 3, the network layer the TCP/IP stack uses the IP protocol. On layer 4, transport layer the stack defines either the use of TCP or User Datagram Protocol. The IoT protocol to use needs to be implemented on layer 7 in the OSI model, which is the application layer. Figure 2.1 shows the TCP/IP stack represented using the OSI model and the structure of the Ethernet frame.

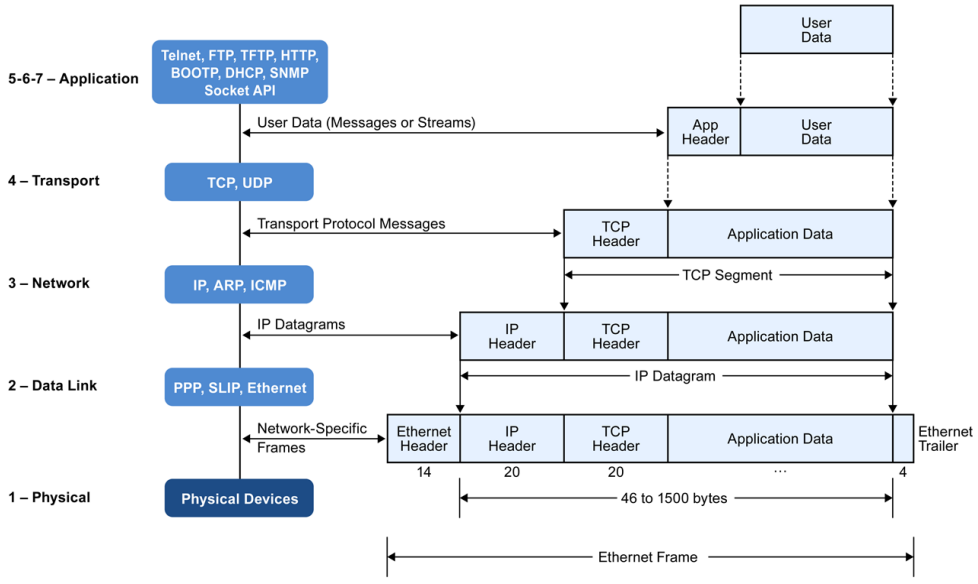


Figure 2.1 TCP/IP stack represented using the OSI Model. To simplify the model layer 5,6 and 7 in the OSI model is grouped together.[7]

It is described in Appendix A that Modbus TCP should be used as IoT protocol. Modbus TCP is an application layer protocol belonging to layer 7 in the OSI model.

2.1 The IoT Device

A IoT device is defined as an embedded connected device that has an IP address.[6] And it has some sensor(s) for collecting data and/or actuator(s) for performing a tasks. Because it is a connected device the data collected by the sensor can be sent to an application. The application can then analyze the data, and then for example send a command back to the IoT device to make it perform an action based on the sensor data.

The IoT device used in this project has been developed for laboratory experiment purposes by the "SMART" research group at HSN. The device uses Modbus TCP protocol for commutation. The device is designed to give the students practical experience both with the Modbus protocol and communication with a IoT device. A picture of the IoT device is shown in Figure 2.2.

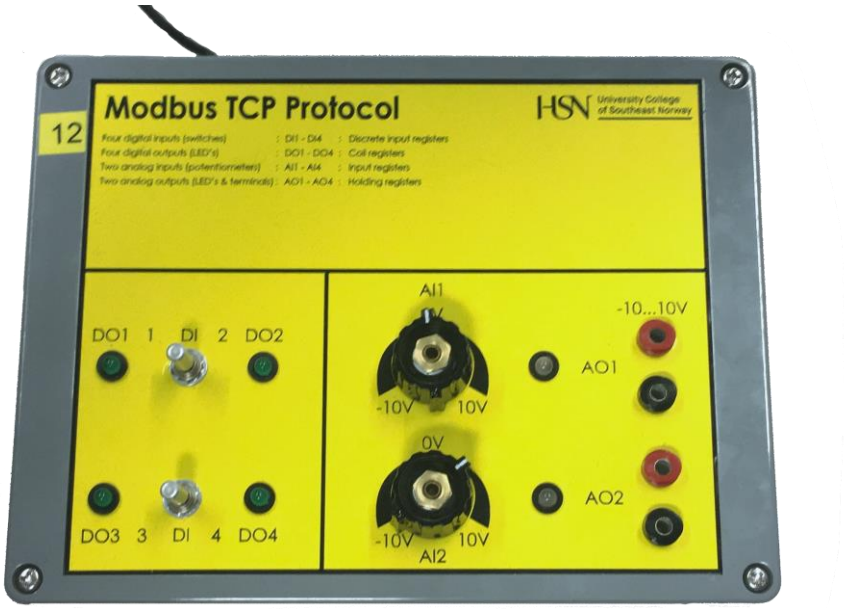


Figure 2.2 The IoT Device.

The IoT device has both analog and digital inputs and outputs. On the left side of the front is where the digital Input / Output (I/O) is located. Consisting of two switches and four LEDs, these I/Os are not hard connected, but instead mapped to Modbus registers. The two switches, acting as digital input, are mapped to the Modbus Discrete input register. Then the four LEDs, acting as digital outputs, are mapped to the Modbus Coil Register. The registers are only accepting a single bit value, 0 or 1. For the digital input switch, the register value is 1 for off and 0 for on. The switch has three states, but only mapped to two registers, since there is only two LED's per switch. The upper switch is meant to be paired with LEDs DO1 and DO2, and the lower switch is meant to be paired with DO3 and DO4.

For analog I/O there is two knobs for input, and two LEDs and 2 voltage output connectors. Each knob is a potentiometer with 16-bit resolution. The value of these potentiometers can be

read using the input register in Modbus. The LEDs and voltage output can be set using the holding register in Modbus. But because of the design of the IoT device, Coil register 5 and 6 needs to be set high to be able to write to holding registrars for analog output. This is because there is a calibration happening when coil register 5 and 6 is set to low, as they default are. As long as there is power there is no need to set coil 5 and 6 after it is initially done. Holding register 9 and above is used for configuration of the IoT Device.

Since the analog input is 16 bit, it gives a value in type of short, which is a value between -32768 to 32767. The output register is also a 16 bit, so there is no need for conversion between input and output. The resolution of voltage output is calculated in equation (2.1).

$$\frac{voltage\ range}{number\ of\ bits} = \frac{20\ volt}{65536} = 305\ \mu V/\ step \tag{2.1}$$

To keep a connection to the IoT device open, a Modbus package must be sent every two seconds. Else the IoT device will set the RST flag in the TCP header, forcing the connection to be terminated.

2.2 Intranet

An intranet is a local private network where only a given set of people has access to the network.[8] The main parts an intranet consists of is a router, one or multiple switches, one or multiple computers and/or other connected devices. An example of an intranet is shown in Figure 2.3.

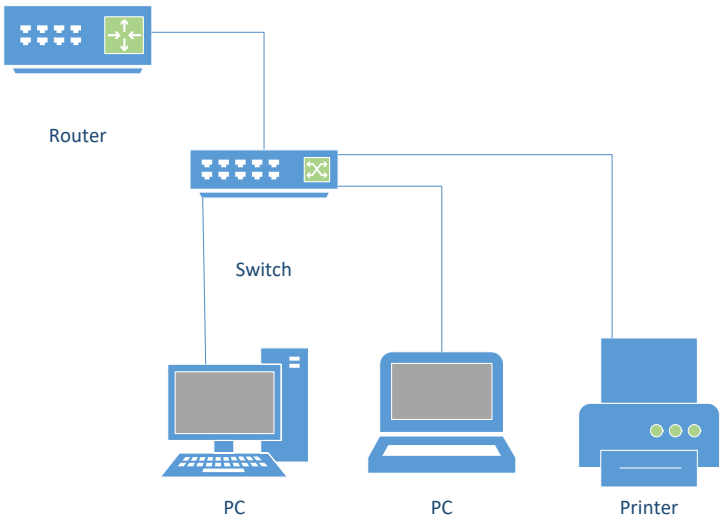


Figure 2.3 An intranet set up with a router, switch, two computers and a printer.

2.2.1 Router

The router's main purpose is to know all the Internet Protocol (IP addresses and direct packages to its destination using its routing table)[6]. A router operates on layer 3 in the OSI model which is the network layer.

2.2.2 DHCP

Every device connected to the network needs an IP address. And by using DHCP (Dynamic Host Configuration Protocol) devices connected to the network gets assigned a IP address from a given range of IP addresses. It also sets settings as lease time for the IP address handed out.

Typically, in a network the DHCP server is located on a router or a dedicated server. Having multiple DHCP servers on the same network will cause problems, because multiple sources are giving out IP addresses.

2.2.3 Switch

A switch operates on level 2 in the OSI model and this is the Data link layer. A switch connects devices together based on the Media Access control (MAC) address of the device. A MAC address is a unique identifier assigned to network interface.

2.2.4 Description of the intranet for this system

A CISCO RV320 router is configured with a DHCP server. The router is then connected to a switch. The IoT Device is connected to the switch. And a computer is also connected to another gate at the switch. Figure 2.4 shows how the intranet are connected.

Appendix B documents how to configure the DHCP server on CISCO RV320.

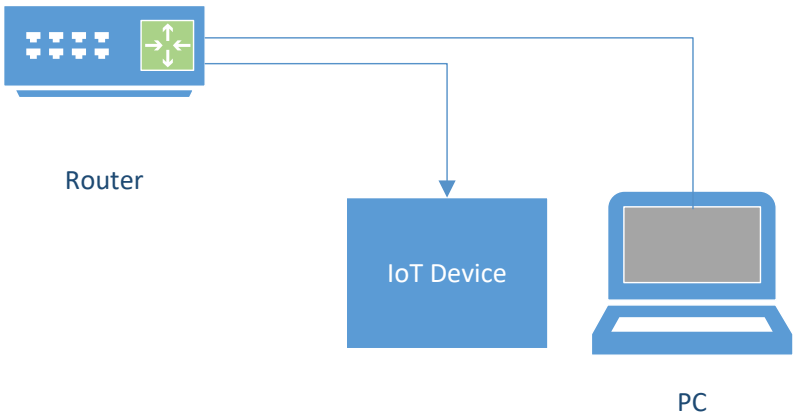


Figure 2.4 Intranet for IoT System.

2.3 Laboratory Assignment

Laboratory assignments are tasks designed to give students both theoretical knowledge but also practical experience in certain subjects. To learn the students about IoT, protocols and interconnection, it is important to first make sure they have the theoretical knowledge to be able to tackle a practical problem. This can be achieved by study about the basic principles of IoT, protocols and networking, then write in a report a brief overview of what they read. And after they learn the basic theory, it is important that the tasks in the practical part is using this theory.

It is also important that the assignment is a set of tasks to be solved and not a step by step guide to follow. By using a step by step guide, the students will be handed the solution, instead of using their minds to try to find the solution, making the learning outcome limited.

A diagram showing how to connect the intranet for the assignment are shown in Figure 2.4. The diagram shows a computer connected to the router and the IoT device connected to the router.

3 IoT Protocols

As described in chapter 2, there is no universal standard for IoT. Based on the definition of IoT made earlier, this chapter will give an overview of different protocols working on the application layer and is suitable IoT and IIoT.

3.1 OPC UA

OPC stands for Open Platform Communications and is a set of specifications for industrial automation.[9] UA stands for Unified Architecture and this protocol is used for collecting data and control.[10] It is the new OPC protocol with better features than its predecessor OPC Classic. OPC Classic consisted of three separate specifications: OPC Data Access, OPC Alarms & Events and OPC Historical Data Access. The specifications were based on windows technology and therefore bound to the windows system. While the new UA is platform independent, making it possible to be used on a range of hardware platforms and operating systems.

OPC UA operates on layer 5-7 in the OSI model.

3.2 CoAP

CoAP stands for Constrained Application Protocol.[11] It is designed for use on constrained devices. That is devices with constrained resources, for example limited battery life. CoAP uses UDP as transport layer protocol. UDP is a connectionless protocol, that is a method for transferring data without ensuring that the recipient is available or ready to receive data. Since CoAP uses UDP it supports UDP broadcast and multicast.

CoAP is based on client-server relationship, where the client sends a request to the server and the server sends a response back to the client.

A sensor node in CoAP is often a server and note a client. So, the sensor has resources for indicating status or alter state that the client can access.

CoAP is operating on layer 7 in the OSI model.

3.3 MQTT

MQTT stands for Message Queue Telemetry Transport and is a messaging protocol designed for lightweight machine to machine (M2M) communication.[11] It uses TCP and consists of one server known as a broker and one or many sensors known as clients that is connected to the broker. In MQTT the clients can subscribe to what is known as a topic. A topic is an address where all the messages are published to. One client can subscribe to multiple topics.

Each client can also register what is known as a Last Will and Testament message on the broker. If the client disconnects the broker will then send out the message to the subscribers.

MQTT is working on layer 5-7 in the OSI model.[12]

3.4 XMPP

XMPP stands for Extensible Messaging and Presence Protocol.[13] It is originally a protocol for instant messaging and routing of XML (Extensible Markup Language) data. XMPP is an open and free protocol.

XMPP operates on level 5-7 in the OSI model.[14]

3.5 Modbus

Modbus is an open protocol developed by Modicon.[15] When released in 1979 it was a serial communications protocol for use with PLCs. Now there are three main versions of Modbus; Modbus RTU, Modbus ASCII and Modbus TCP. The foundation is the same for all three types and only interface specifics are different. Modbus RTU is for communicating over serial using binary representation of data. Modbus ASCII is for communication over serial using ASCII characters for protocol communication. And Modbus TCP is using TCP/IP for communication.

Modbus RTU and ASCII needs the slave ID in the beginning of the message and a checksum at the end. While Modbus TCP needs only a Modbus Application Header (MBAP header) at the start of each message. The data part of a Modbus message is called a Protocol Data Unit (PDU) and is the same for all three versions of Modbus. The PDU consists of Modbus function code and then data. The whole Modbus message, where the header and PDU is combined is called an application data unit(ADU). A comparison of the ADU structure for Modbus RTU and Modbus TCP can be seen in Figure 3.1. Modbus TCP operates on layer 7 in the OSI model.

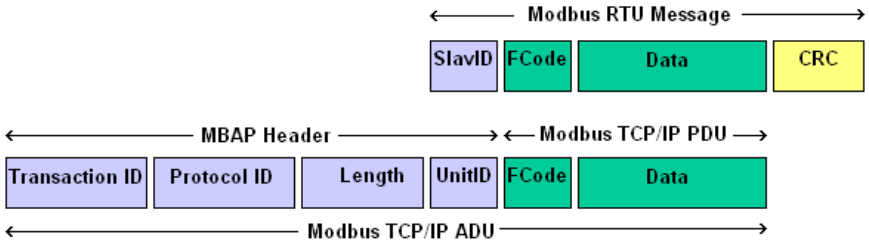


Figure 3.1 ADU: Modbus RTU ADU and Modbus TCP ADU

The function code is a set of codes specified by the Modbus specification.[16] Storing the data in Modbus are done on the slave in four different registers. An overview and specification for each register is shown in Table 3.1.

Table 3.1 Modbus Registers overview.

Register	Access	Size
Coil	Read/Write	1 bit
Discrete Input	Read Only	1 bit
Holding	Read/Write	16 bit
Input	Read Only	16 bit

When it comes to security the Modbus protocol has nothing to offer. And since it has a known data structure for the sending data, the data can easily be intercepted and read by unwanted persons. In addition, when using Modbus TCP, anyone who gets access to the network can connect to the device and send commands. The same applies for Modbus RTU and Modbus ASCII.

3.6 Comparison of the Protocols

Table 3.2 shows a comparison of the different protocols presented in this chapter.

Table 3.2 Comparison of IoT Protocols.

Protocol	Transport	Messaging	Advantages
OPC UA	TCP	Request/Response Publish/Subscribe	Platform Independent
CoAP	UDP	Request/Response	Multicast Support
MQTT	TCP	Publish/Subscribe	Last Will and Testament message
XMPP	TCP	Request/Response Publish/Subscribe	Open protocol
Modbus TCP	TCP	Request/Response	Open protocol and easy to implement

3.7 IoT Protocol for this thesis

In this thesis, the task description stated that Modbus TCP should be used. As described in chapter 3.5, Modbus TCP works on layer 7 and is an open protocol. Modbus TCP is an easy protocol to implement, and a big part of this is because the Modbus organization has created a implementation guide for implementing Modbus TCP.[17]

An important part of Modbus is the ADU build for each Modbus TCP message. Table 3.3 shows the structure and parts making up the ADU for Modbus TCP.

Table 3.3 Table showing the Structure of a Modbus ADU.

Field	Part of	Length	Description
Transaction Identifier	MBAP	2 Bytes	This is a unique id for each transaction set by the client and echoed by the server.
Protocol Identifier	MBAP	2 Bytes	Set by client and is always set to zero.
Length	MBAP	2 Bytes	Set for each message and is indicating the total number of bytes in the message.
Unit Identifier	MBAP	1 Bytes	Set by client and echoed by the server. It is used as an identifier for the connected slave(server)
Function Code	PDU	2 Bytes	2 bytes set by the client indicating the function to be performed by the slave. This is then echoed by the server.
Data	PDU	Variable	Data to be sent is inserted here

As seen in the table, the ADU consist of a MBAP header and a PDU containing a function code and data. But there exist three types of PDUs in Modbus: MODBUS Request PDU, MODBUS Response PDU and MODBUS Exception Response PDU. The Request PDU is sent by the client. Then the server sends back a ADU with either a Response PDU or an Exception response PDU, depending if there was an error. Figure 3.2 shows the processing of the Modbus request on the Modbus server.

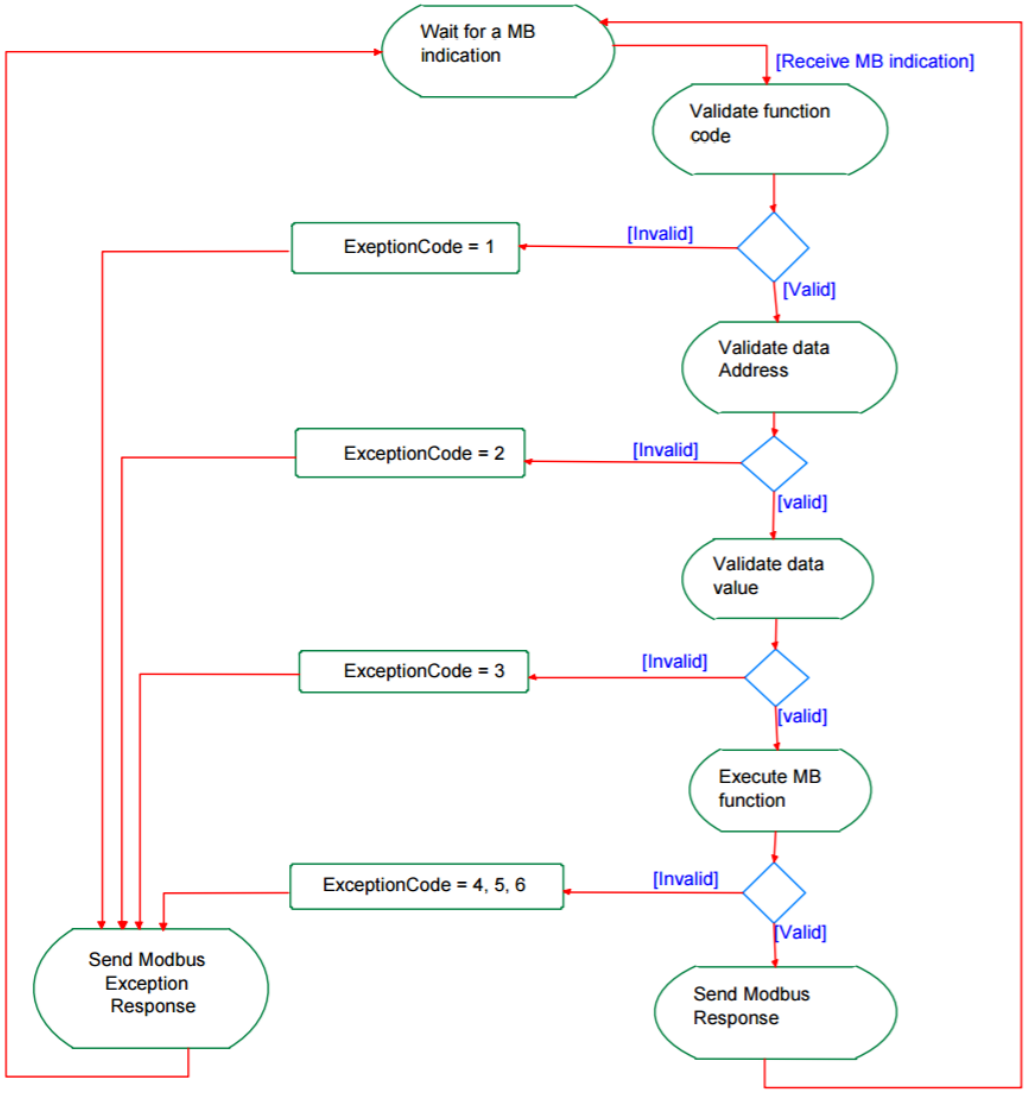


Figure 3.2 Modbus Transaction state diagram. [16]

Another important aspect to Modbus is that is limited to selected data types as shown in Table 3.1. Coil and Discrete input register are limited to on/off values. And Holding and Input register are of type short, which are 16-bit a integer value from -32768 to 32767 can be stored.

Because of this Modbus don't support float numbers. But by defining a standard, for example if a three-digit number is sent, then the decimal is the last number. So, to illustrate. If the value 422 is received, the float number would be 42.2.

4 Methods

In this chapter the different tools and methods used to solve the tasks in this assignment are presented.

One of the main tasks in this thesis is to develop software based on the Object-Oriented Analysis and Design (OOAD) approach. Both develop a set of objects for client server communication but also an application for communicating with the IoT device. To do this properly a development process chosen to be used as well as tools and design patterns relating to objects.

4.1 Unified Process

Unified process is a software development process.[18] It is use case driven and is an incremental and iterative development process. A software development project can be divided into four main parts; Inception, Elaboration, Construction and Transition. These parts are then broken into smaller parts known as iterations. This is shown in Figure 4.1. After each iteration, the project has incremented with additional or improved features. And, because testing is a part of each iteration in UP, after each iteration the software should be free of bugs.

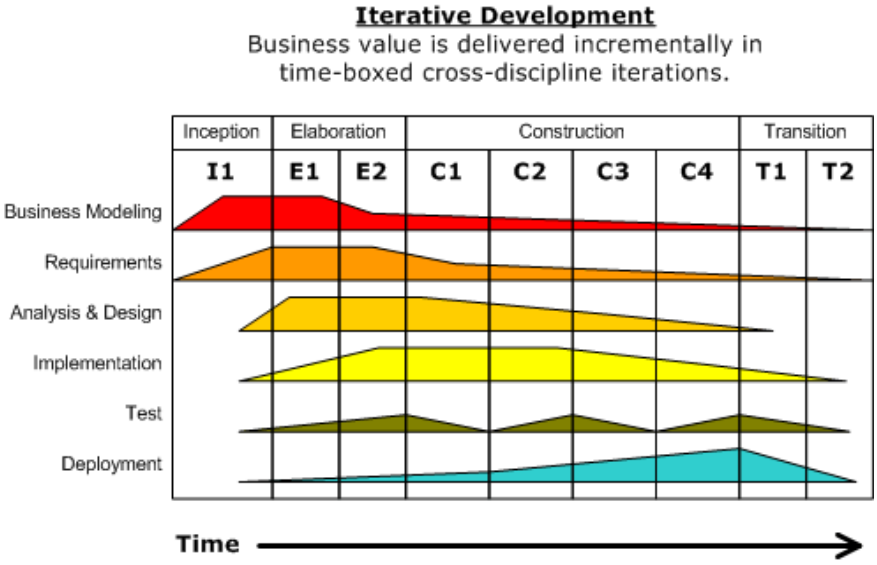


Figure 4.1 Diagram showing how workload on different areas changes throughout a Unified Process project.[18]

A use case is a functionality in the system and to say it is use case driven means each iteration is focusing on a use case. It is common that the use case selected for the first iteration in UP is the hardest or the most work heavy.

4.2 UML

UML stands for Unified Modelling Language and is a standard modelling language for documenting software systems.[19] UML provides a set of diagrams to develop and each diagram serves its own purpose. By using a development process that uses UML, it makes sure

the software is well documented and it makes it easier for others who knows UML to both understand the software and to make changes to it. UP uses UML as a tool in the development process. The UML diagrams chosen for this project are: Use Case diagram, Interaction diagram and class diagram.

A use case diagram is used to show the use cases and actors in a system. As stated in 4.1 a use case is a functionality while an actor is entity that interacts with the use case. The purpose of a use case diagram is to get a high-level view of the system. Only one use case document will be created for the whole software. The use case diagram is developed in the analysis phase when using the OOAD approach. For each iteration when using UP, in the analysis phase, a fully dressed use case document based on the use case for the current iteration are created.

Interaction diagram is used to describe the interaction between the different parts of the system. There are two types of interaction diagram: Sequence diagram and Collaboration diagram. For this project sequence diagram was chosen as interaction diagram. This because it shows the interaction with focus on time, while a collaboration diagram only focuses on the link between objects. A sequence diagram is developed based on the main success scenario and extensions in the fully dressed use case document. In UP, there will be one interaction diagram produced per iteration. An interaction diagram created in the design phase of each iteration when using OOAD approach.

Class diagram is a diagram describing the classes and connection between the classes. It also gives an overview of attributes and methods for each class. Only one class diagram will be created in design phase of the first iteration, and this diagram will be modified in the later iterations.

4.3 Version Control System

A version control system is a system for keeping track of changes to files over time.[20] There exist three main types of version control systems. The first one is local version control systems. This is a system where all the changes to files is kept in a simple database on the local machine. Figure 4.2 shows the structure of a local version control system. Such a database when it comes to version control is known as a repository.

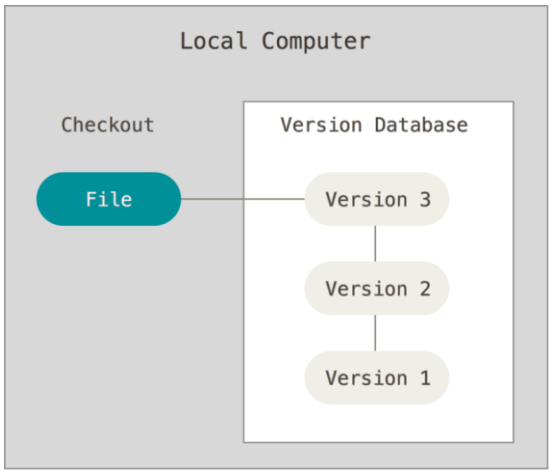


Figure 4.2 Local Version Control System.[20]

The problem with a local version control system is that it is designed to work only locally on one machine, so there is not possible to keep track of changes done by other computers. A solution for this problem is Centralized Version Control System (CVCS). The idea behind CVCS is that it uses a repository located on server and multiple clients can check files in and out of that server. Figure 4.3 shows the structure of a CVCS.

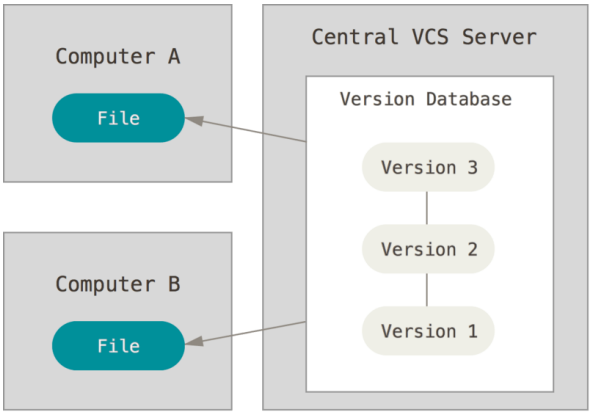


Figure 4.3 Centralized Version Control System.[20]

An advantage that CVCS has over a local VCS is that it gives the users of the CVCS possibility to see changes made by other users. A downside to CVCS is that because it consists of a single server, in an event where the server goes down, the repository can't be accessed and therefore not save new changes. Or worse, if the hard drive of the server fails the whole repository is lost. To solve this issue, Distributed Version Control Systems were introduced. As the name implies the version control system is distributed instead of only consist on a single server. It still has a single server, but each computer working on the files mirror the whole repository. So, in an event where the server fails, it can be restored by copying the repository from any of the clients. Figure 4.4 shows the structure of a Distributed Version Control System.

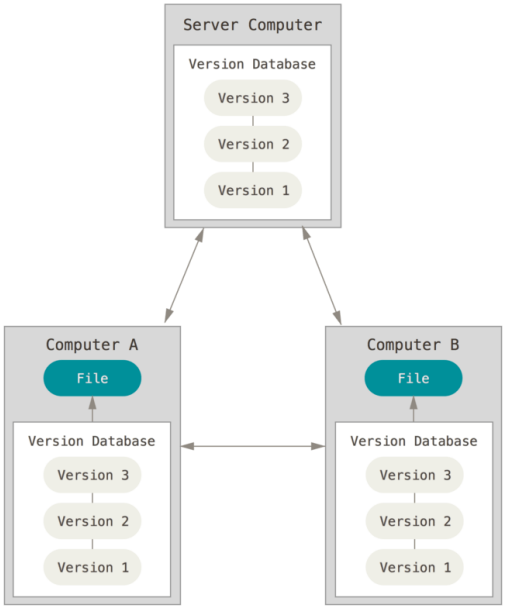


Figure 4.4 Distributed Version Control System.[20]

Since the development of the software took place both at school on a portable computer and at home on a desktop computer, GitHub was chosen as version control system. GitHub is distributed version control system and has a web based interface for keeping track of the repository. Since GitHub is a distributed it keeps a copy of the repository on each location which gives an added security and backup in case something happens to either.

4.4 Coding Conventions

Coding conventions is guidelines for the programming style.[21] The main reason to use coding conventions is that it gives the code a structure and the code a specific style.

For the software developed C# naming conventions is used. The specific naming conventions used are lower camel case for variable names and Hungarian notation for naming controls.

Lower camel case is using initial lowercase letter for first letter in the name.[22] If the name consists of multiple words there is no space between the word, but the first letter of the next word is in uppercase. For example, to name a variable Log Event in lower camel casing would become logEvent.

Hungarian notation is first putting in a short name of the type of object and then a name for the object.[23] For example, to name a button Connect, the name using Hungarian notation would be btnConnect.

Another coding convention used is keeping each class in separate files. This ensures that it is easy to find the code for a specific class since each class are in separate files.

4.5 Design Patterns

A design pattern is a general repeatable solution to common problems that occur when designing software.[24] In addition to being a template for solving a problem it also can make the code more readable.

For design of the software the Facade pattern is used. This pattern is a part of the 23 patterns defined as Gang of four patterns. Facade pattern is about hiding a complex system behind a new interface class. The facade pattern is used in the design phase.

Another group of patterns are called general responsibility assignment software patterns (GRASP). [25]This group consists of patterns with guidelines for assigning responsibility to classes and objects. Grasp patterns is used when creating interaction diagram. From GRASP the following patterns has been used: controller, information expert and creator pattern.

The controller pattern is used to give another class than a UI class the responsibility of dealing with system events. Creator pattern is used to assign responsibility to a class that will create an instance of another class if certain criteria are fulfilled. Information expert pattern is used to assign responsibility to the class that has the information to fulfill it.

4.6 Testing

For testing of the software, a test plan is used. A test plan is a document describing how to test the software and other factors concerning testing, for example where to test and

hardware/software requirements.[26] The test plan contains test cases which are cases created based on the software requirements and fully dressed use cases. The purpose of a test plan is to ensure that the test can be repeated and still obtain the same results. And in addition, be used to perform tests according to the plan after changes to the software has been made. This is to verify that the testing of the cases still has the gives the same results. The test plan for the software can be found in Appendix C.

4.7 Wireshark

Wireshark is a network protocol analyzer. It can be used to capture packets sent over a network in real time.[27] By selecting a network interface, packages sent over the network are shown in the graphical user interface. It is possible to add filter to view only certain packages with the selected attribute. Wireshark has a filter for Modbus, enabling for decoding Modbus packages sent over the network.[28] This makes it possible not only to see that a Modbus message is sent over the network, but also know what it contains.

By applying the Modbus filter only packages built up as valid Modbus messages are recognized by Wireshark and displayed. So TCP messages for SYN, ACK and RST is not displayed. This can be confusing because these messages can affect the connection between Modbus client and server.

Figure 4.5 shows a screenshot of the Wireshark application. This screenshot shows packages sent on the network and because the Modbus filter is enabled, it only shows packages using Modbus/TCP protocol. The selected package is a package sent from the computer to the IoT Device. As shown in the Modbus section in the lower part of the screenshot, Wireshark can decode a Modbus ADU and display the its content, including the values sent.

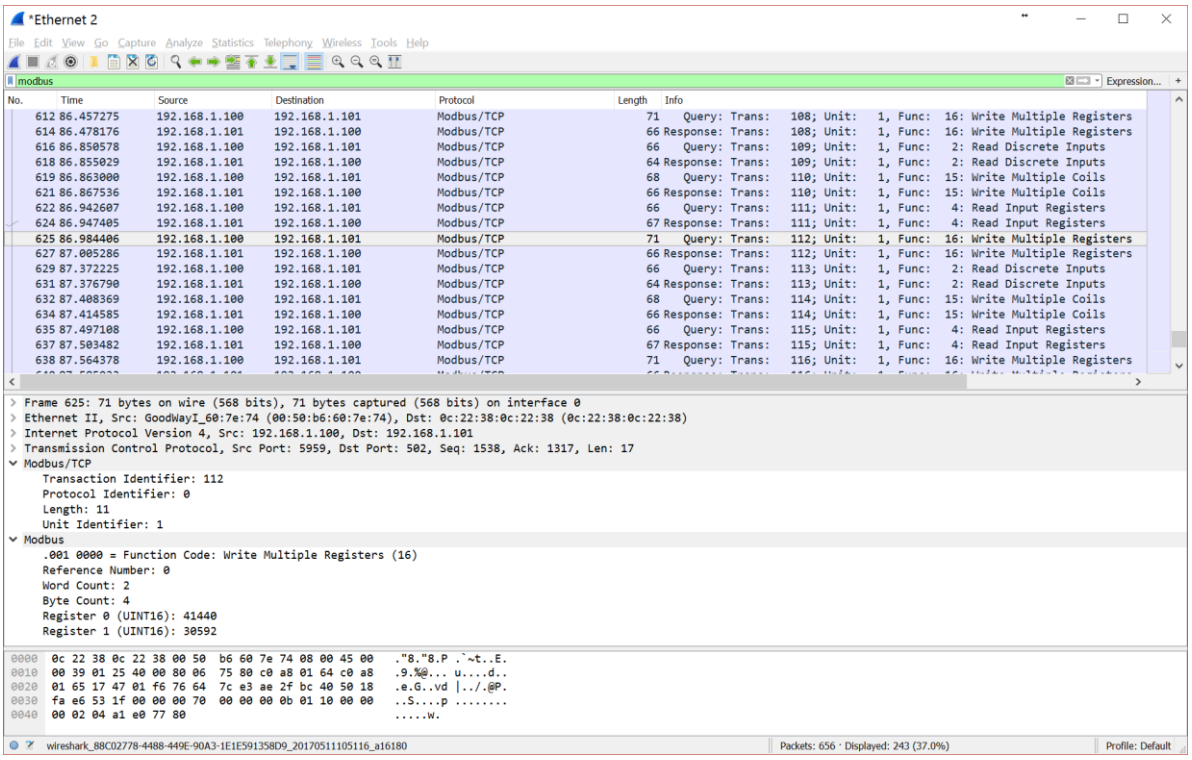


Figure 4.5 Screenshot of Wireshark with Modbus filter enabled and a message selected.

5 Analysis

This chapter covers the requirements for the software and the analysis of the first iteration. The main result from the analysis for each iteration is the fully dressed use case created.

5.1 Requirements Specification

The IoT device needs to be connected to a network setup with DHCP to obtain an IP address. The IP address of the IoT device is required to be able to establish a connection to it.

The device has two 3-State toggle switches for digital input. The state of the switches is stored in the Discrete input register. For digital output the device has four LEDs that can be set by writing to the coil registers. The LEDs are supposed to light according to the position of the digital input switches. There is no direct connection between the input switches and the output LEDs, but both are accessible by using Modbus. So, the logic needs to be performed in the software.
For analog input the device has two knobs that are potentiometers that gives a number from -32768 to 32767. The value of the potentiometers can be read from the input registers. For analog output, there is two LEDs and two sets of connectors that can give out a voltage between -10 and 10 volts. To control the outputs, a value needs to be written to the holding register. The register is 16 bit so a value from -32768 to 32767 is accepted. When writing a value of 32767 to holding register with address 00 the device should output 10 volts on AO1 and the AO1 LED should be green.

There should be a ability to generate a PDF report containing time period, traffic of messages and range of Modbus registers used.

5.2 FURPS+

FURPS+ is a classification system for collection of requirements.[19] The name FURPS+ is an acronym standing for: Functionality, Usability, Reliability, Performance, Supportability. And the + is used for design limitations. Below are the requirements gathered based on the requirement specification using FURPS+.

Functionality:

- Handle Digital I/O: Read digital inputs, write digital output and handle output based on input.
- Handle Analog I/O: Read analog inputs, write analog output and handle output based on input.
- Handle Modbus Communication: Connection and communication with Modbus Slave
- Generate Report: Generate a PDF report based on the communication with IoT Device.

Usability:

- Interact using digital switches and analog knobs on IoT Device.
- Keyboard and mouse for input.

- GUI for displaying information using a tab view. One tab page for writing and reading to registers using textboxes. One tab page for controlling the inputs on the device and indicating the outputs. And one tab page showing the messages sent between the application and the IoT device.

Reliability:

- Running when connected to IoT Device

Performance:

- Continuous polling of Device with option to do it manually.
- GUI update after new poll

Supportability:

- None

+ Implementation:

- Connection between software and IoT device using TCP/IP protocol.
- Needs a Router that has a DHCP server giving out IP between 100 and 150.
- Programming language is C#.

5.3 Use Case Diagram

Based on the functional requirements found using FURPS+ the following UML use case diagram shown in Figure 5.1 was developed.

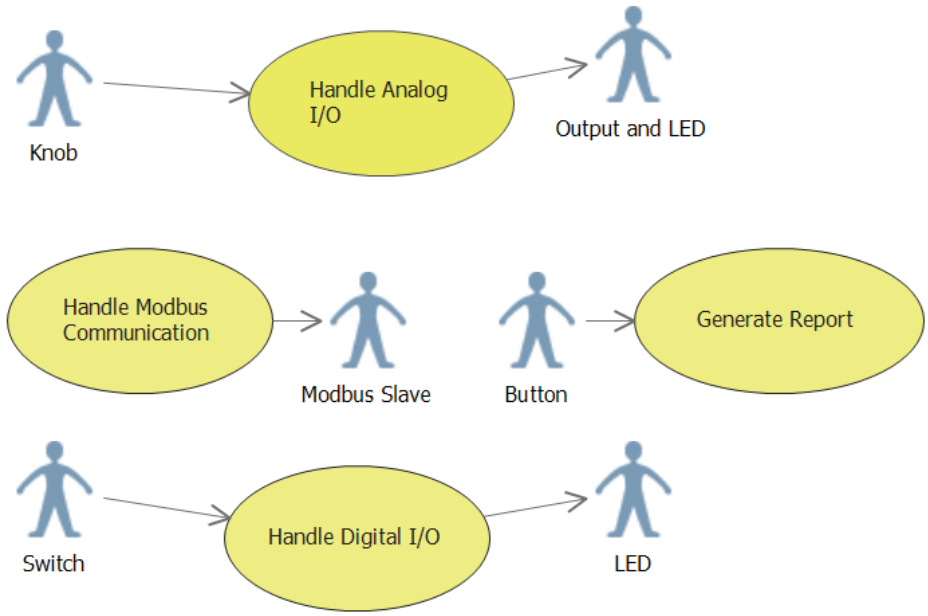


Figure 5.1 Use case diagram for the system.

The use case diagram contains six actors and four use cases. “Handle Analog I/O” reads the position of the analog knobs and sets the output according to value read.

"Handle Digital I/O" reads the position of the switch and sets the LEDs according to the position of the switch.

The use case "Handle Modbus Communication" handles all the communication with the Modbus slave.

The last use case is "Generate Report" which is handling generating of the PDF report containing information about the Modbus communication with the IoT device.

5.4 Prototype Of User Interface

Before the first iteration, a prototype of a user interface for the software was developed. The purpose for creating a prototype is to give an idea how the user interface might look. Screenshots of the prototype is shown in Figure 5.2, Figure 5.3, Figure 5.4 and Figure 5.5.

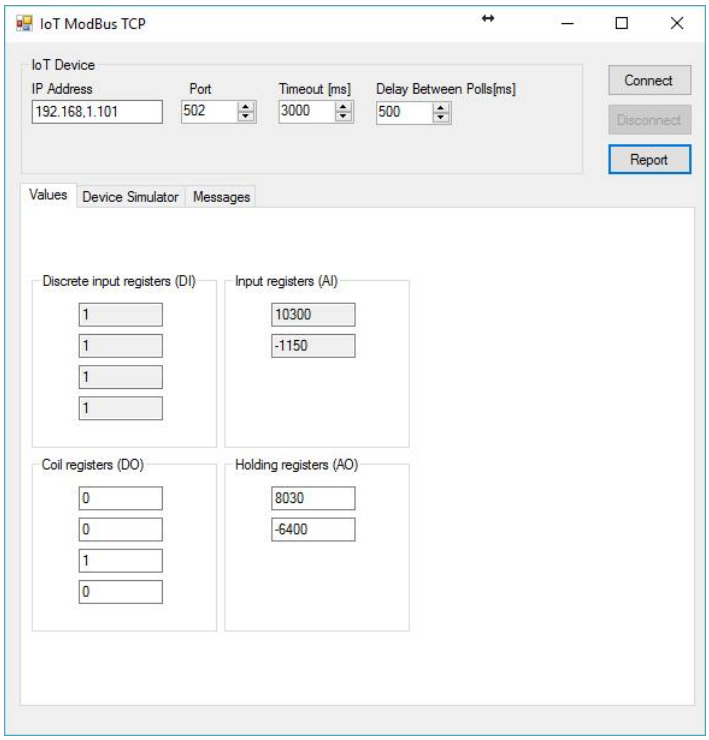


Figure 5.2 Screenshot of prototype with value tab selected.

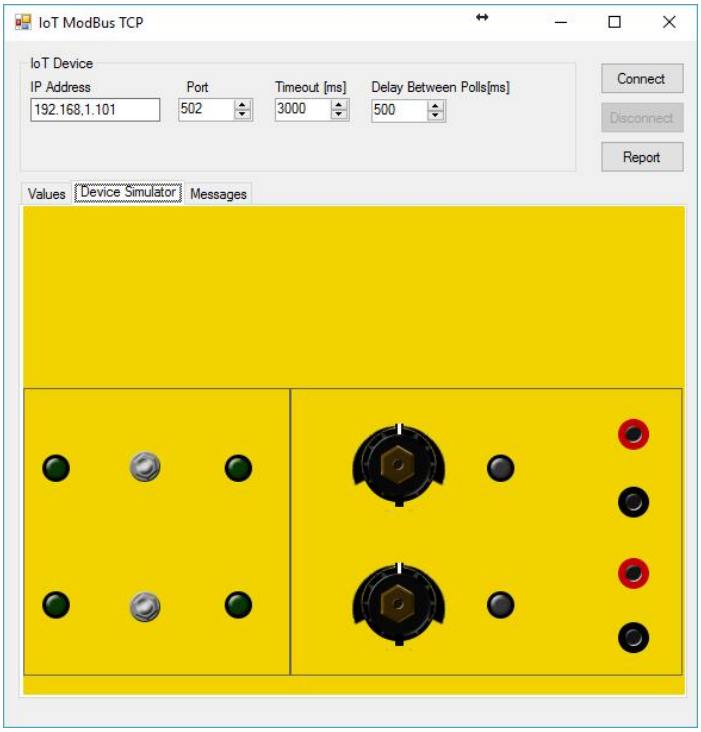


Figure 5.3 Screenshot of prototype with Device Simulator tab selected.

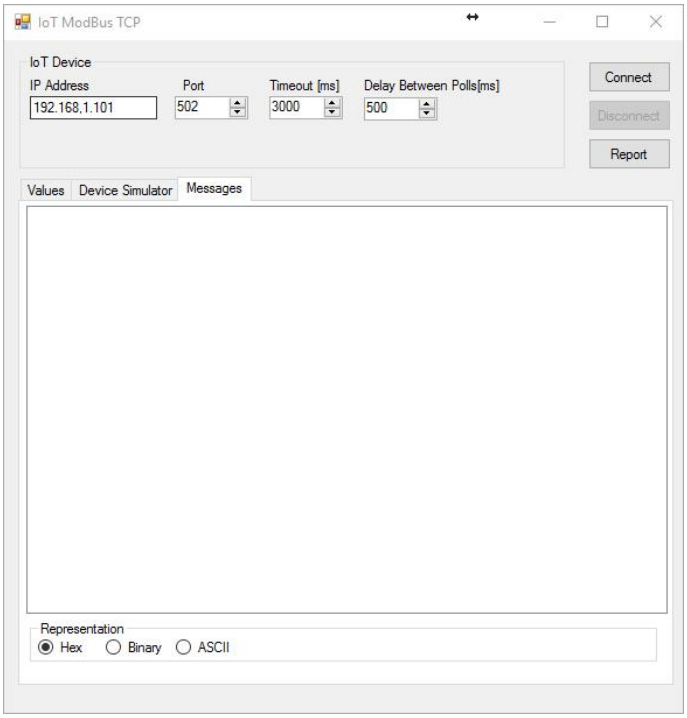


Figure 5.4 Screenshot of prototype with Message tab selected.

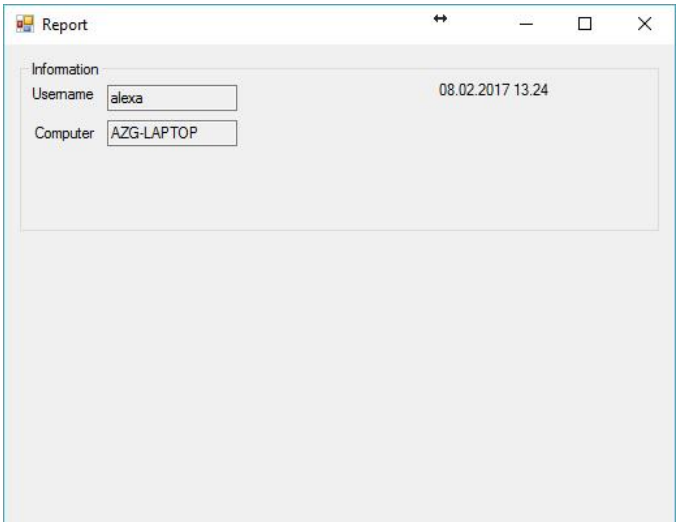


Figure 5.5 Screenshot of prototype with Report form open.

5.5 The First Iteration

In UP, the use case for the first iteration is either the hardest one or the one with highest risk.[18] For this application, the use case selected for the first iteration was Handle Modbus Communication. Out of the four use cases found in the use case diagram in Figure 5.1 it is the most complex use case to implement, and the use case associated with the highest risk.

5.5.1 Fully Dressed Use Case

Table 5.1 shows the fully dressed use case created for the Handle Modbus Communication use case. A fully dressed use case document gives a more detailed overview of how the use case should work.[19] The most important parts of the fully dressed use case is the main success scenario and the extensions. That is because they are used to test the software when done. When using UP there should be produced one fully dressed use case diagram per iteration.

Table 5.1 Fully dressed use case for Handle Modbus Communication use case.

Use case section	Comment
Use Case name	Handle Modbus Communication
Scope	IoT Modbus
Level	User Goal
Primary actor	IoT Device
Stakeholders and interests	User needs system to communicate with Modbus Slave

Preconditions	Computer and IoT device needs to connect to a network with router and DHCP server.
Success guarantee	Communicate with IoT Device using Modbus TCP/IP.
Main Success Scenario	A. Read IP address and port from GUI. B. Connect to IoT Device. C. Save connection time D. Wait for Input E. Create ADU F. Send ADU G. Wait for Response from Slave H. Record transaction info I. Goto D
Extensions	A1. No IP address or port, give error message B1. Not able to connect, Give error message B2. On disconnect record stop time D1. If no changes in input & not first iteration, then read registers to keep alive. G1. If message received has exception code, show exception message on screen and disconnect
Special req.	None
Technology list	None
Frequency of occurrence	Every time communicating with Modbus Slave
Miscellaneous	None

6 Results

In this chapter the results for this thesis will be presented. First the results of implementing the software based on the analysis done in chapter 5 and the Modbus TCP implementation guide.[17] Then a part describing testing of the software. And then a description of the proposed laboratory assignment created.

6.1 Software Implementation

After the analysis phase in the first iteration, the design phase was conducted. The results from the design phase for the first iteration is the interaction diagram. Based on main success scenario from the fully dressed use case in Table 5.1 and by using the design patterns described in 4.5 the interaction diagram in Figure 6.1 was created.

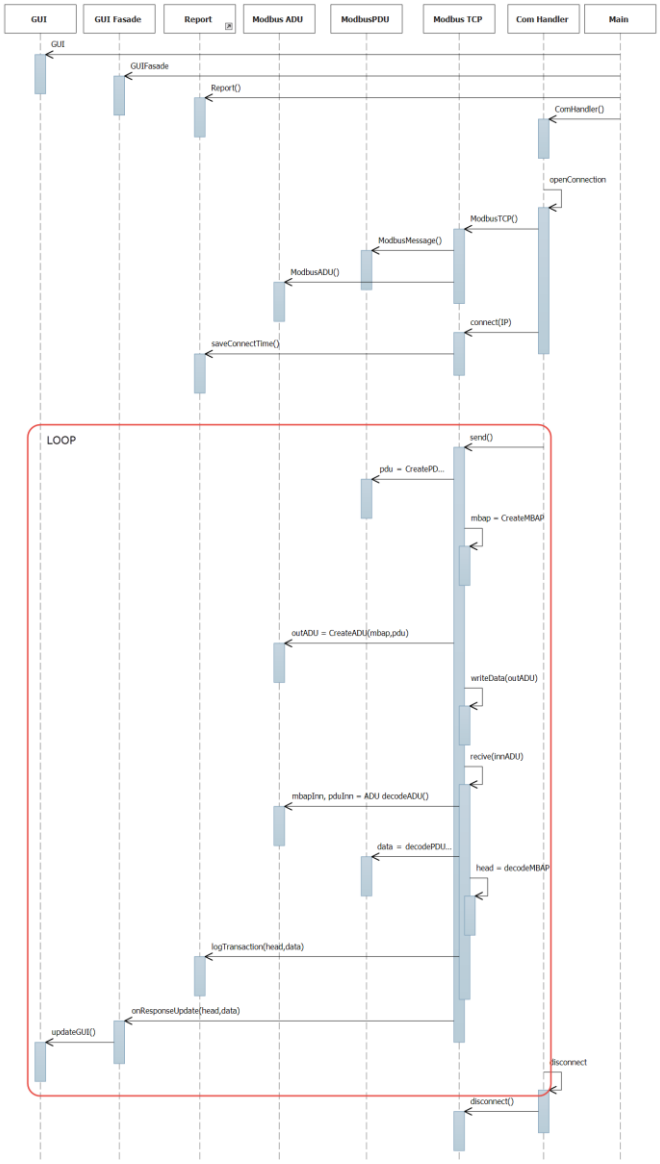


Figure 6.1 Interaction diagram for use case "Handle Modbus Communication".

The fully dressed use cases and interaction diagrams for the rest of the iterations can be found in Appendix D.

The other result of the design phase is the class diagram. A class diagram shows the classes, attributes and methods for the software and these are taken from the interaction diagrams. A simplified class diagram showing only the important attributes and methods are shown in Figure 6.2.

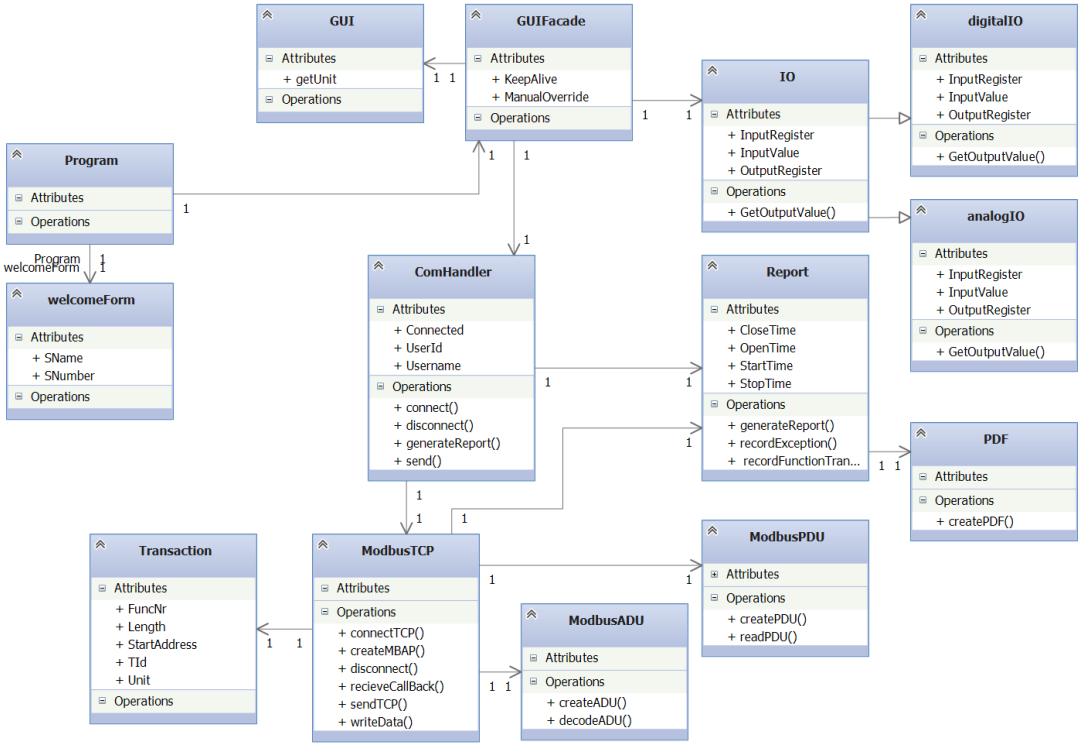


Figure 6.2 Simplified class diagram for the application.

In the objectives for this thesis there were two software development task. One was developing a set of objects in C# for Modbus TCP client/server communication and the other was to develop an application based on the objects created in the first task. Since these two were so closely related, they were developed in one solution.

The Modbus communication is implemented using the facade pattern and therefore offers a facade class with the more complex system hidden behind the facade. The facade class has been named ComHandler and can be seen in the class diagram in Figure 6.2. The ComHandler is works as a interface class and contains a set of simple methods and attributes. Behind the facade class there is the ModbusTCP class handling interaction with multiple classes.

To use the ComHandler, an object of needs to be created. The ComHandler constructor has two parameters that needs to be provided, a name as string and a student number as int. Then there are four delegates in ComHandler that needs to be added to event handlers. The code for this is show below.

```
ModbusCom.OnResponseData += new ComHandler.ResponseData(OnResponse);
ModbusCom.OnError += new ComHandler.ErrorData(OnError);
ModbusCom.OnOutData += new ComHandler.OutData(OnOutData);

ModbusCom.OnException += new ComHandler.ExceptionData(OnException);
```

After this is done the connect method in ComHandler can be called. The method has two parameters, the IP of the Modbus slave as a string and the port number as a int.

Then when connected the send method in ComHandler can be used to communicate with the IoT Device. There are two versions of the send method, accepting different parameters. One is for sending a read function code and the other is for write function code. Table 6.1 shows the Modbus function codes supported and Table 6.2 shows the supported Modbus exception codes that the Modbus slave can respond with.

Table 6.1 Modbus function codes supported.

Function Code	**Function Type**
1	Read Coil
2	Read Discrete Input
3	Read Holding Registers
4	Read Input Registers
5	Write Single Coil
6	Write Single Holding Register
15	Write Multiple Coils
16	Write Multiple Holding Registers

Table 6.2 Modbus exception codes supported.

Exception Code	**Exception Type**
1	Illegal Function
2	Illegal Data Address
3	Illegal Data Value
4	Slave Device Failure
5	Acknowledge
6	Slave Device Busy

8	Memory Parity Error
10	Gateway Path Unavailable
11	Gateway Target No Response
12	Timeout

In ComHandler there is a method named generateReport. When called, this method generates a PDF report that gets put on the desktop of the user's computer. The file contains the username, student number, information about connection time, statistics on function code calls and an exception code log. An example of a report can be found in Appendix E.

The two classes digitalIO and analogIO has been created and are used to calculate output based on input.

The source code for the software objects for can be found on:

https://github.com/Zhangnamstyle/IoTUsingModbus/tree/master/IoTModbus/Modbus

And the source code for the whole Modbus application can be found on:

https://github.com/Zhangnamstyle/IoTUsingModbus/tree/master/IoTModbus/IoTModbus

6.1.1 Graphical User Interface

The graphical user interface in the developed software is based on the prototype. But throughout the development process changes has been made. There are now only two tabs, IoT Device Simulator and function code tester. And there is a separate form opening when starting the application. In this form the user is asked to input username and student number. There is also not a separate form for report, only a button on the main screen that generates a PDF report on the desktop. A screenshot of the welcomForm is shown in Figure 6.3. Figure 6.4 shows the Device simulator tab and Figure 6.5 shows the Function Code Tester tab.

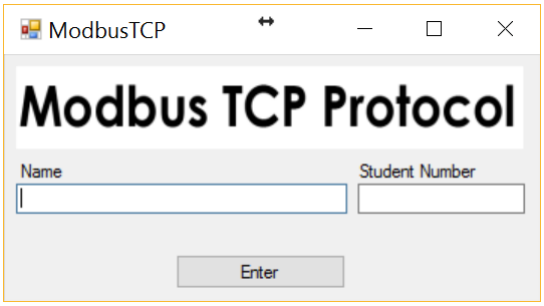


Figure 6.3 Screenshot of the GUI for welcomeForm.

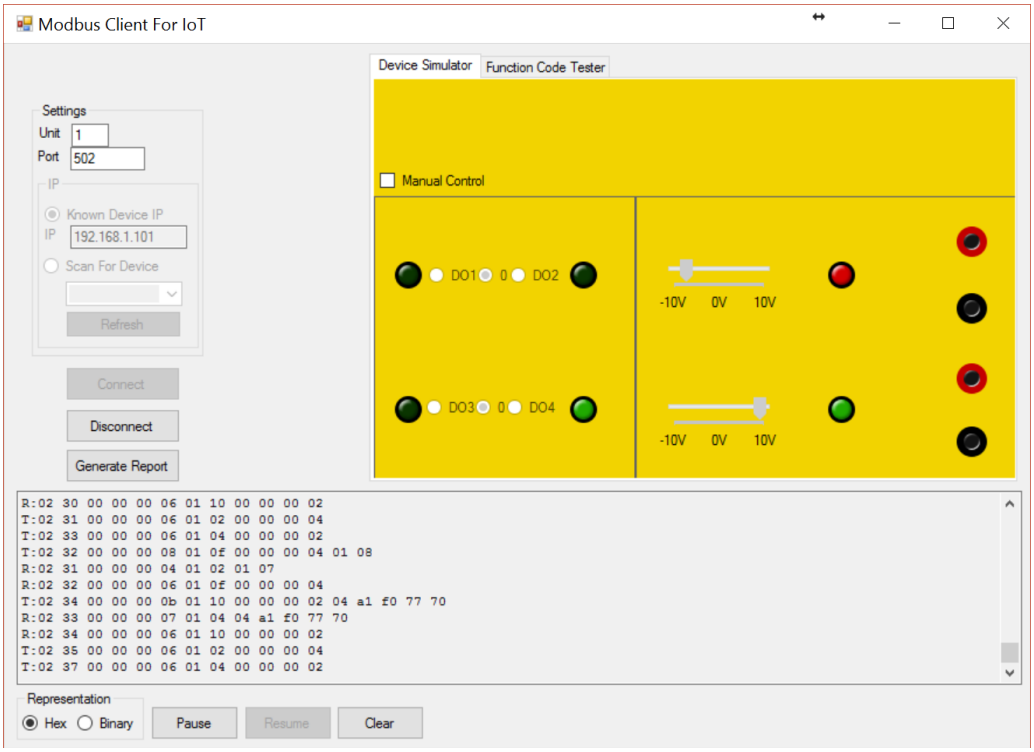


Figure 6.4 Main screen with Device Simulator tab selected.

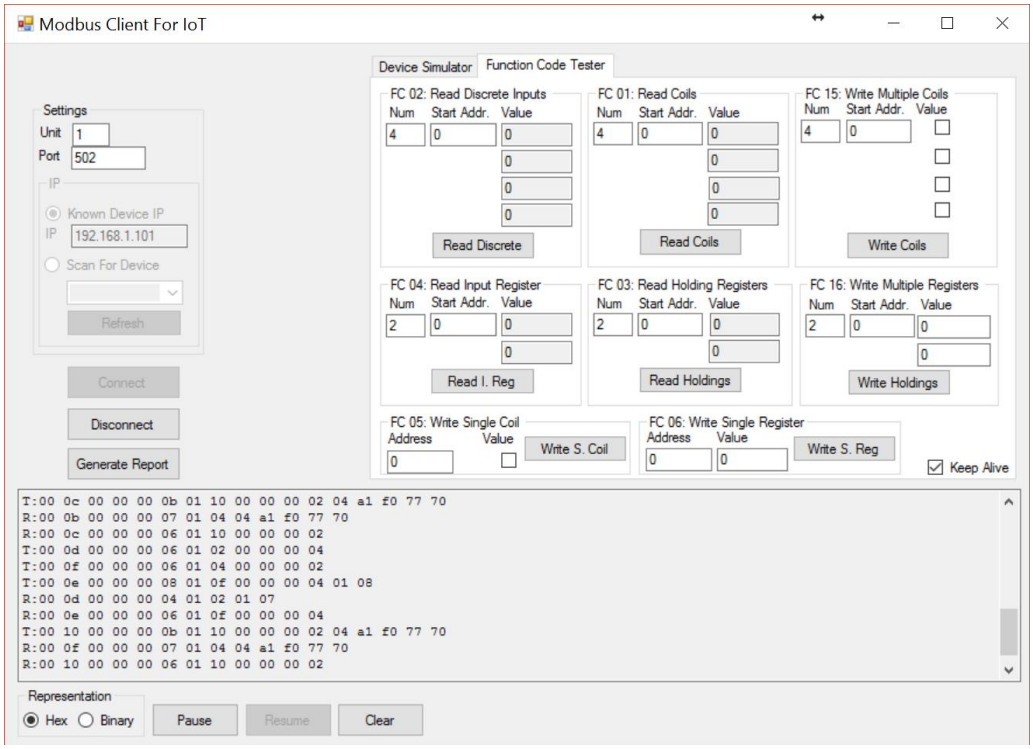


Figure 6.5 Main screen with Function Code Tester tab selected.

6.2 Testing

Since UP has been used as development process, testing is a part of each iteration. So after each iteration most bugs has been identified and fixed during the development process. But to make sure the software fulfilled the requirements and to identify any bugs remaining, a last was conducted. This test was done according to the test plan in Appendix C. The test cases were performed and all but one had the expected outcome. There were one bug in the PDF report where the total time the application has been open showed the wrong value.

6.3 Laboratory Assignment

A proposal to a laboratory has been created. It is one lab assignment for both BSc. And MSc. programs and it consist of four parts. The first part is theory, mainly Modbus but also about LAN, DHCP and IoT. In this part the students need to find information and write the answers in a report for the Assignment.

The next part is network setup. In this part the students need to configure the router and set up a DHCP server on the router with a specific range. They also need to make sure both their computer and the IoT device is handed a IP address by the DHCP server.

Then the next part is practical Modbus where they are going to use the IoT Device and the software developed in this thesis to solve a given task. In this part the students also needs to use Wireshark to debug Modbus messages.

Finally, the last part is development of software for Modbus Communication. In this part the students are going to develop a Modbus communication application using the dynamic-link library (DLL) developed in this thesis. Or if they want a challenge they can try writing the whole Modbus communication their self.

The proposed assignment text can be found in Appendix F

7 Discussion

The main focus during this thesis has been to develop a working application for communication with the IoT Device. The application produced has been developed according to the requirement specification in Chapter 5.1.

For the development of the application, UP was used. And in UP every iteration has an equal fixed length. For this software development project, the length of each iteration was not specified, but a deadline for the development process was set. This was mainly because the first iteration contained a large workload compared to the three remaining iterations. As expected the first iteration was time consuming while the rest was done in a short time. The deadline was set to a formal meeting where a demo would be held. The application was developed within the deadline and a working program was demonstrated. There were a few minor issues, but the main functionality worked according to the requirements. Also some feedback was received from the demo, and these were implemented after the meeting. Also the minor issues that occurred was fixed right after the meeting.

To generate the PDF report, different C# libraries was evaluated, but ITextSharp was chosen.[29] This because it was easy to implement and well documented. The other libraries evaluated was either poorly documented or was hard to implement.

The intranet used for testing and development has consisted of a the IoT device, a CISCO RV320 router with DHCP and a laptop. But according to the Task description, a switch should also have been part of the intranet. A switch has not been used since it operates on Layer 2, the data link layer, and would not affect the software, since this is handled by the TCP/IP stack.

The suggested laboratory assignment covers the topics stated in the objectives. The theory part will give the students the information they need for solving the rest of the assignment. The assignment is given as tasks that they need to solve, and not as a step by step guide to follow. This is the case in many other laboratory assignments at HSN. While a step by step guide make sure everybody gets through the assignment, the learning outcomes are small since you are told what to do.

7.1 Future Works

During testing holding register 8 and above was written to. Those registers are linked to the configuration of the device. Writing to those registers resulted in that the configuration for the device was deleted and the device needed to be configured using USB. A fix has been started implemented in the application by restricting writing to Holding register addresses above 99. But to implement this fix the firmware of the IoT device configuration register address to start from an address above 99. The bug that shows the wrong total time open the application has been open in the PDF report will also need to be fixed.

Another suggestion for future work is to extend the application so that there is a tab for Lab work. In this tab the Modbus action to perform can be given. In the current solution, this action must be given to the students from the person responsible for the exercise. By making the software give each student a random action helps making sure each student does their own lab work and not just copy each other. In the same Lab tab page, controls for testing the ADU that the students make can also be added. And since the action is given by the application, it is

possible to compare the student created ADU with the correct ADU. The analog outputs on the IoT device has not been included in the suggested laboratory assignment. But this can be added in the future by making the students connect LEDs or measure the output voltage using a multimeter.

8 Conclusion

A set of objects in C# has been developed for Modbus communication according to the requirement specification and by using OOAD methods. An application using these objects was also developed. This application and the objects is then used in a proposed laboratory assignment. The IoT device developed by the “SMART” research group is an embedded system and it has an IP address. It also has sensors and actuators making it an IoT device. A laboratory assignment on IoT is very handy because IoT is still a relatively new field with lots of possibilities. And by getting hands on experience the students get a better understanding of IoT and the technologies making it possible. And since IoT is part of Internet 4.0, which are called the fourth industrial revolution, the theory they learn are relevant for the future.

Modbus TCP has been implemented in the software as a IoT protocol. It is possible to send function codes both for reading and writing to the registers on the IoT device. And if the device sends a exception code, the software shows the user the exception. Generating a PDF with transaction information has been implemented.

References

[1] M. T. J. (2016). *A Comparison of IoT Gateway Protocols: MQTT and Modbus*. Available: https://software.intel.com/en-us/articles/a-comparison-of-iot-gateway-protocols-mqtt-and-modbus

[2] ARM. (2017). *Course/Lab Material for Internet of Things*. Available: https://www.arm.com/support/university/educators/iot-material/index.php

[3] H.-P. Halvorsen. (2016). *Industrial Internet of Things (IIoT) using Raspberry Pi*. Available: http://home.hit.no/~hansha/documents/subjects/SCE4206/iiot_raspberry_pi.htm

[4] R. Sutaria and R. Govindachari. (2013). *Understanding The Internet Of Things*. Available: http://www.electronicdesign.com/iot/understanding-internet-things

[5] Intel Corporation. (2014, Developing Solutions for the Internet of Things. Available: http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/developing-solutions-for-iot.pdf

[6] N.-O. Skeie, "Industrial Information Technology," 2015.

[7] Micrium Software. (2017). *Part 3: IoT Protocol Stack Options*. Available: https://www.micrium.com/iot/internet-protocols/

[8] Wikipedia contributors. (2017). *Intranet*. Available: https://en.wikipedia.org/wiki/Intranet

[9] OPC Foundation. (2017). *What is OPC?* Available: https://opcfoundation.org/about/what-is-opc/

[10] OPC Foundation. (2017). *Unified Architecture*. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/

[11] T. Jaffey. (2014). *MQTT and CoAP, IoT Protocols*. Available: https://eclipse.org/community/eclipse_newsletter/2014/february/article2.php

[12] HiveMQ. (2015). *MQTT Essentials Part 3: Client, Broker and Connection Establishment*. Available: http://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment

[13] XSF. (2017). *An Overview of XMPP*. Available: https://xmpp.org/about/technology-overview.html

[14] A. N. Solutions. (2017). *Solution Technology*. Available: http://aunigma.com/solution_technology/

[15] Wikipedia contributors. (2017). *Modbus*. Available: https://en.wikipedia.org/wiki/Modbus

[16] Modbus.org, "MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3," 2012.

[17] Modbus.org, "MODBUS MESSAGING ON TCP/IP IMPLEMENTATION GUIDE V1.0b," 2006.

[18] Wikipedia contributors. (2017). *Unified Process*. Available: https://en.wikipedia.org/wiki/Unified_Process

[19] N.-O. Skeie, "Object-oriented Analysis, Design, and Programming using UML and C#," 2016.

[20] S. Chacon and B. Straub. (2014, 1). *Getting Started - About Version Control*. Available: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

[21] Wikipedia contributors. (2017). *Coding conventions*. Available: https://en.wikipedia.org/wiki/Coding_conventions

[22] Wikipedia contributors. (2017). *Camel case*. Available: https://en.wikipedia.org/wiki/Camel_case

[23] Wikipedia contributors. (2017). *Hungarian notation*. Available: https://en.wikipedia.org/wiki/Hungarian_notation

[24] Wikipedia Contributors. (2017). *Software design pattern*. Available: https://en.wikipedia.org/wiki/Software_design_pattern

[25] Wikipedia contributors. (2017). *GRASP (object-oriented design)*. Available: https://en.wikipedia.org/wiki/GRASP_(object-oriented_design)

[26] H.-P. Halvorsen. (2017). *Software Testing*. Available: http://home.hit.no/~hansha/documents/industrial_it/resources/resources/Software%20Engineering/Software%20Testing%20Video.pdf

[27] U. Lamping, R. Sharpe, and E. Warnicke. (2014). *Wireshark User's Guide For Wireshark 2.1*. Available: https://www.wireshark.org/docs/wsug_html_chunked/

[28] Wireshark. (2017). *Display Filter Reference: Modbus*. Available: https://www.wireshark.org/docs/dfref/m/modbus.html

[29] B. Lowagie and P. Soares. (2017). *iTextSharp 5.5.11*. Available: https://www.nuget.org/packages/iTextSharp/

Appendices

Appendix A Task Description

Appendix B Setup of DHCP server on CISCO RV320

Appendix C Test Plan

Appendix D Documentation for the second, third and fourth iteration.

Appendix E Example of Modbus PDF report

Appendix F Proposed laboratory assignment

Appendix A: Task Description

University College of Southeast Norway

Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: IoT and Modbus protocol development

HSN supervisor: Nils-Olav Skeie

HSN co-supervisor: Ole Magnus Brastein (PhD student)

Task background:
Internet of Things (IoT) and Industrial Internet of Things (IIoT) are systems for collecting information based on the TCP/IP protocol, using internet or an intranet for communication. However, to get access to the information either a protocol on top of TCP/IP or an Application Programming Interface (API) must be used. The API is normally linked to a cloud service handling the high-level communication with the IoT device, while a protocol on top of TCP/IP can communicate directly with such a device.
The "Smart" research group at HSN has developed an IoT device for laboratory experiment purposes to let students get more hands-on experience with the Modbus protocol and communication with IoT devices. This device is using Modbus TCP as the protocol on top of TCP/IP instead of a cloud service.
Figure 1 shows the IoT device with several I/O devices. These I/O devices are switches for digital inputs, LEDs as both digital and analog outputs, potentiometers for analog inputs, and voltage outputs with measurements points for analog outputs. The I/O devices are connected to the I/O ports on a CPU card and the Modbus TCP protocol will be used to interconnect these I/O devices. The Modbus protocol is commonly used in the industry.

Figure 1: The IoT device with a set of digital and analog inputs and outputs.

The IoT device will be used in a wired local area network (LAN), as an intranet, consisting of a router with a Dynamic Host Configuration Protocol (DHCP) server and a switch, and any computers.

Address: Kjølnes ring 56, NO-3918 Porsgrunn, Norway. **Phone:** 35 57 50 00. **Fax:** 35 55 75 47.

Task description:
The tasks will be to make an overview of IoT protocols, implement the Modbus TCP as an IoT protocol, develop a Windows application handling the interconnection of the I/O devices in the IoT device, and propose a student laboratory assignment based on these tasks. The task specification will be:

- Give an overview of the status for IoT protocols,
- Describe the intranet with a router, switch, DHCP server and computers. Include documentation for setup of the DHCP server,
- Discuss the usage of the Modbus TCP protocol as an IoT protocol,
- Develop a set of software objects in C# for Modbus TCP client/server communication using the Modbus TCP protocol specification and OOAD methods,
- Develop an application, based on the Modbus TCP communication objects, for communicating with the IoT device, and handling the interconnection of the I/O devices,
- Evaluate how a Network Analyser software like the WireShark application can be used to debug and trace the Modbus messages between the application and the IoT device,
- Evaluate C# components for the PDF format and use one of these components for making a PDF report. The report should document the time, traffic of messages and range of Modbus registers used.
- Include a module in the application for testing of the error conditions in the Modbus TCP protocol,
- Propose a laboratory assignment, on both BSc. and MSc. Programmes (IA and IIA) that can use the application, Wireshark, and the objects, with documentation. Setup of the router with the DHCP server for the LAN, IoT system, and Modbus TCP should be the learning objectives.

Student category: IIA student

Practical arrangements:
A router and an IoT device will be available for testing. The workplace is Campus Kjølnes of University College of Southeast Norway. There will be weekly meetings with the supervisor.

The thesis work will start first week of January, and the deadline for thesis hand-in will be in the middle of May. An oral presentation with examination will take place on the Campus no later than end of June.

Signatures: 2. FEBRUARY 2017

Student (date and signature): Aleksan Zhang Gjerseth

Supervisor (date and signature): Nils-Kristian

Appendix B: Setup of DHCP server on CISCO RV320

Guide for setting up DHCP server on CISCO RV320

Open your browser and input 192.168.1.1 in the Address field. You will then be redirected to the page shown in Figure 1.

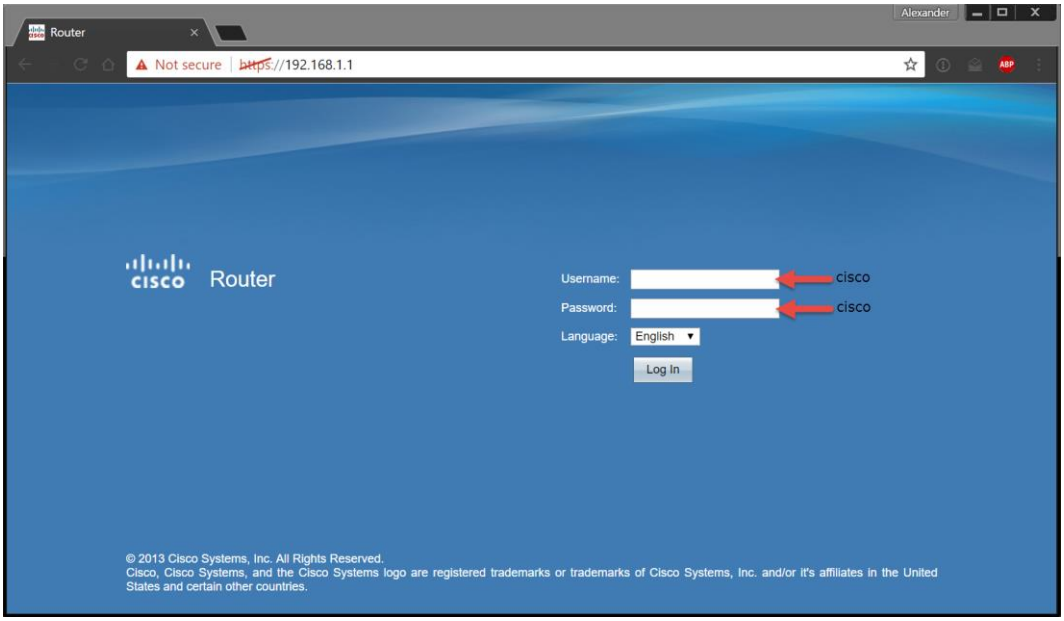


Figure 1 Login Page for CISCO RV320

If you are not displayed the page shown in Figure 1 but instead the page shown in Figure 2. Click where the arrow shows.

Else if you are shown Figure 1. Enter cisco as both username and password. You will then be presented with the page presented in Figure 4. If you are not granted access, use a pen or a similar object to push and hold the reset button located on the front of the RV320 router.

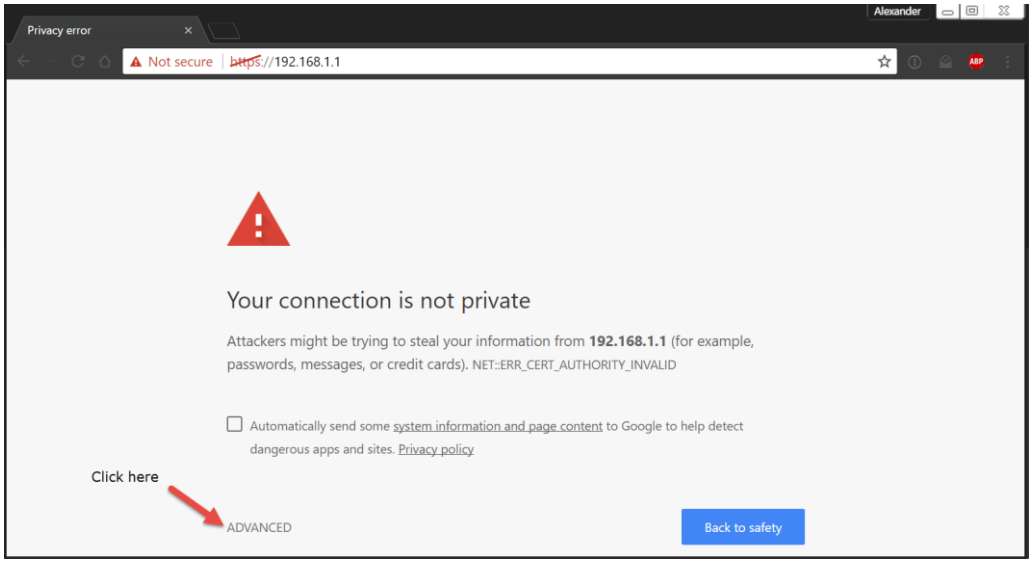


Figure 2 Not secure warning page

Then click on “Proceed to 192.168.1.1(unsafe)” as shown in Figure 3.

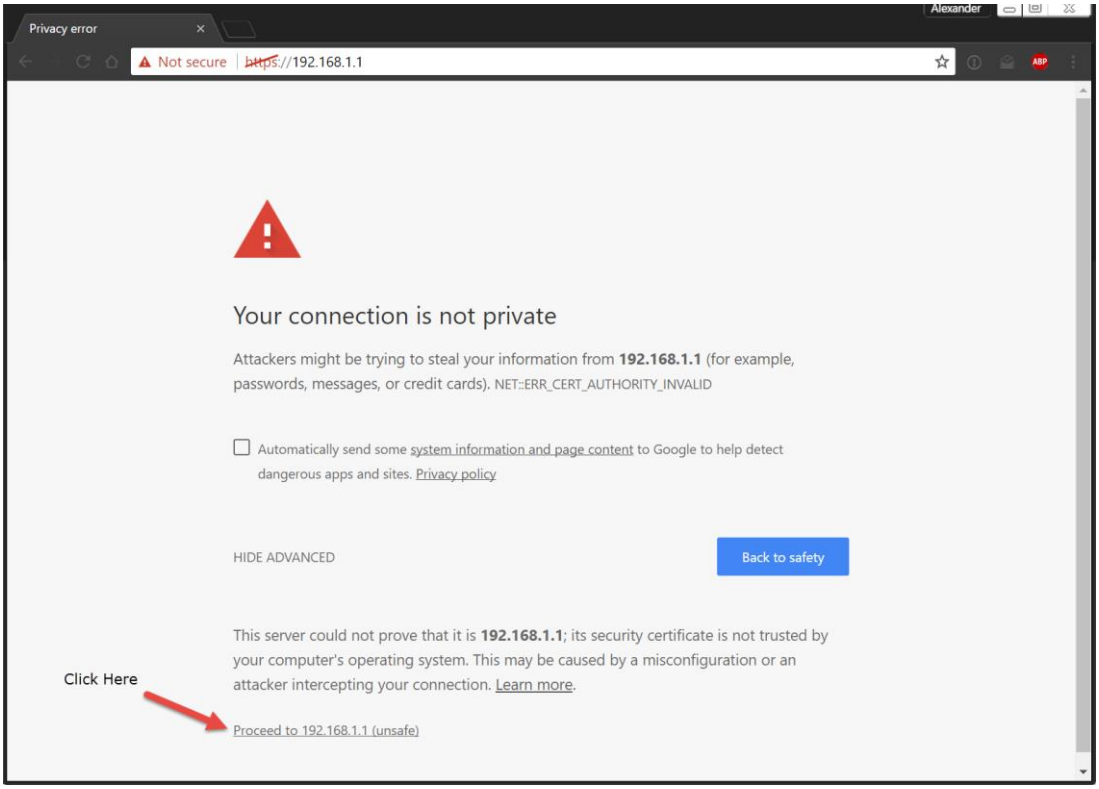


Figure 3 Warning page showing advanced part where you will click proceed.

Then you will be presented with the a page as the one shown in Figure 1.

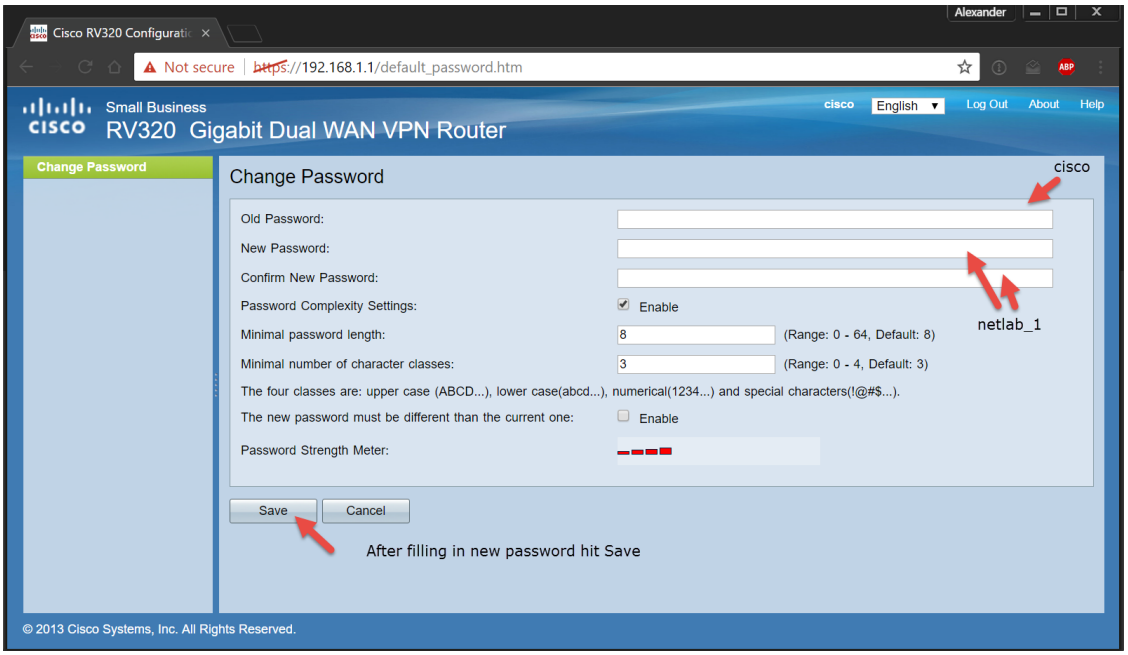


Figure 4 Change password screen

After saving the new password, use it to login when presented with the page shown in Figure 5.

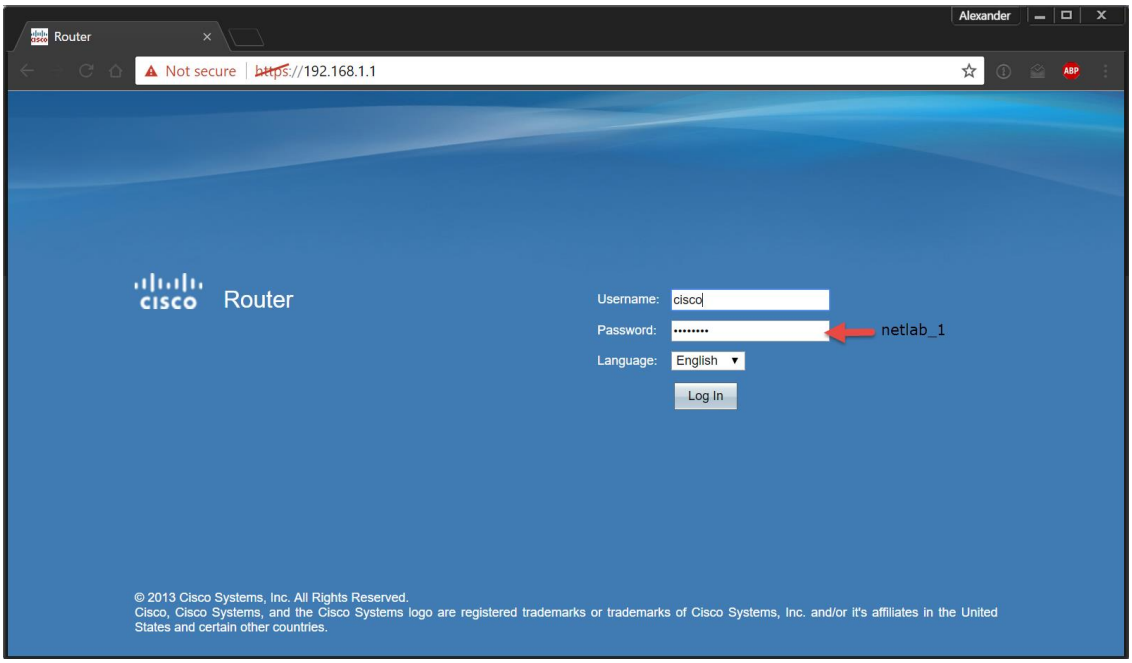


Figure 5 Login page

After logging in, the getting started page in Figure 6 are shown. Click on DHCP in the menu on the left side.

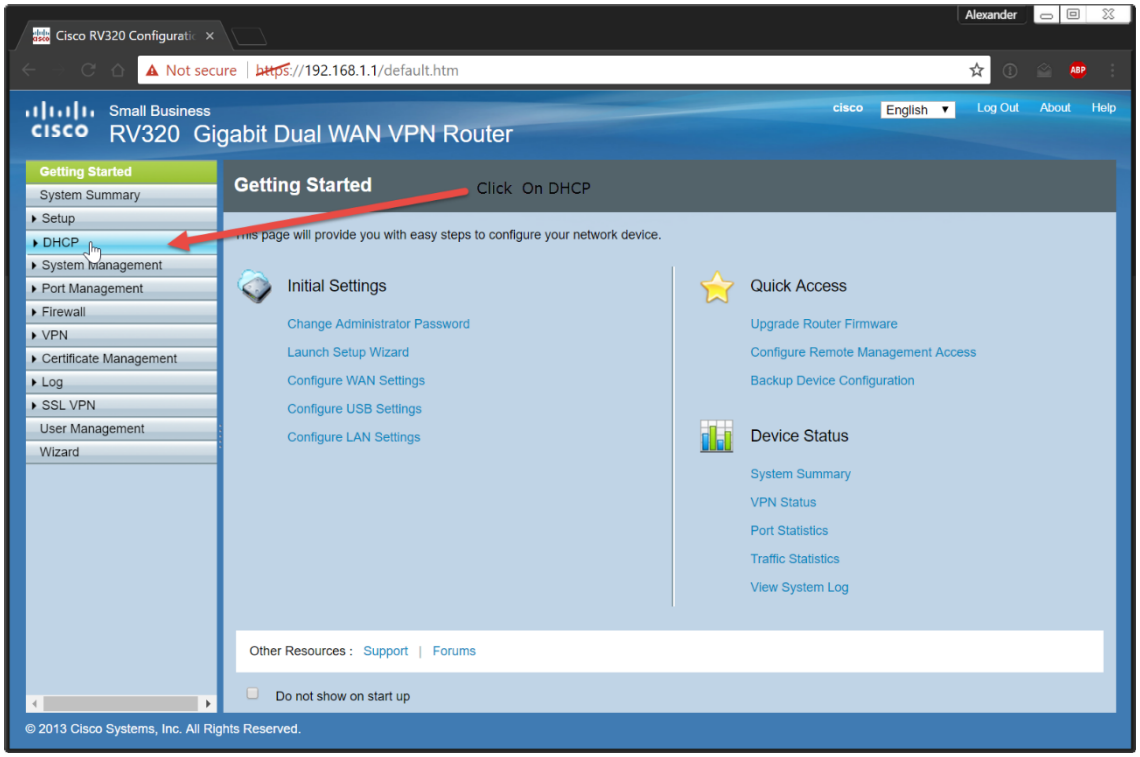


Figure 6 Getting started page, Click on DHCP in the menu

Then the DHCP setup page is displayed as shown in Figure 7. Configure the settings and remember to click save.

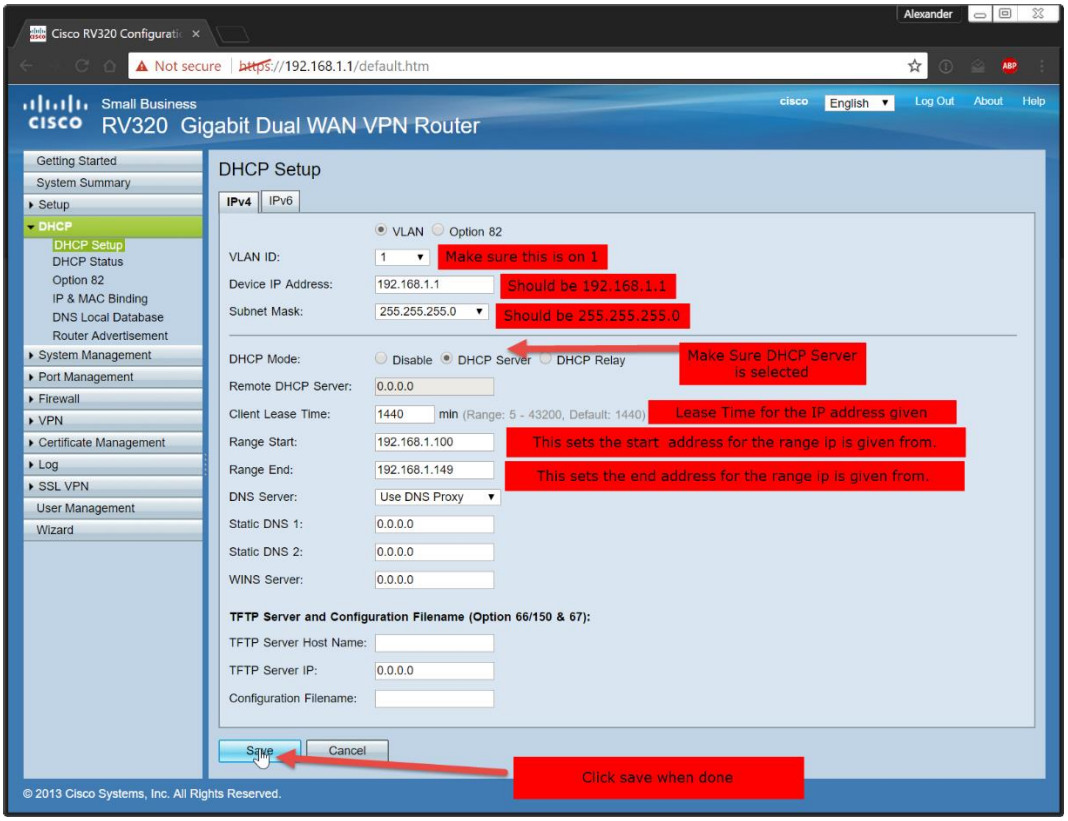


Figure 7 DHCP setup page

After clicking save, you are presented with the same page again. Then click on DHCP status in the menu on the left side as shown in Figure 8.

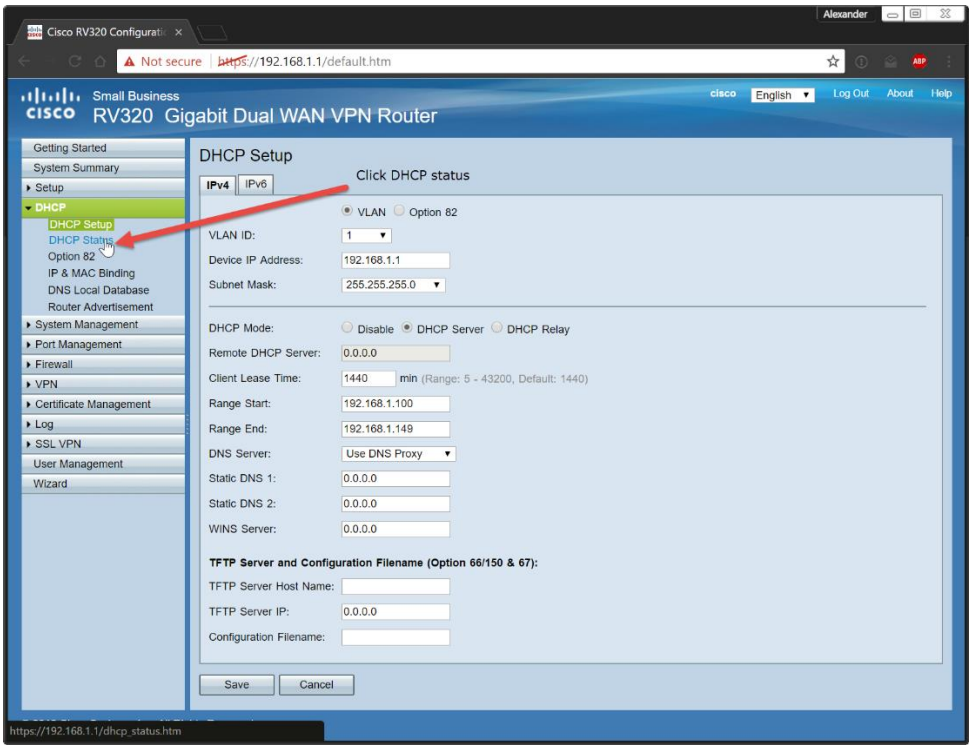


Figure 8 Setup page after clicking save

Then the DHCP status page as shown in Figure 9 will be displayed. This page will show the clients connected to the router and information like the IP address given and remaining lease time

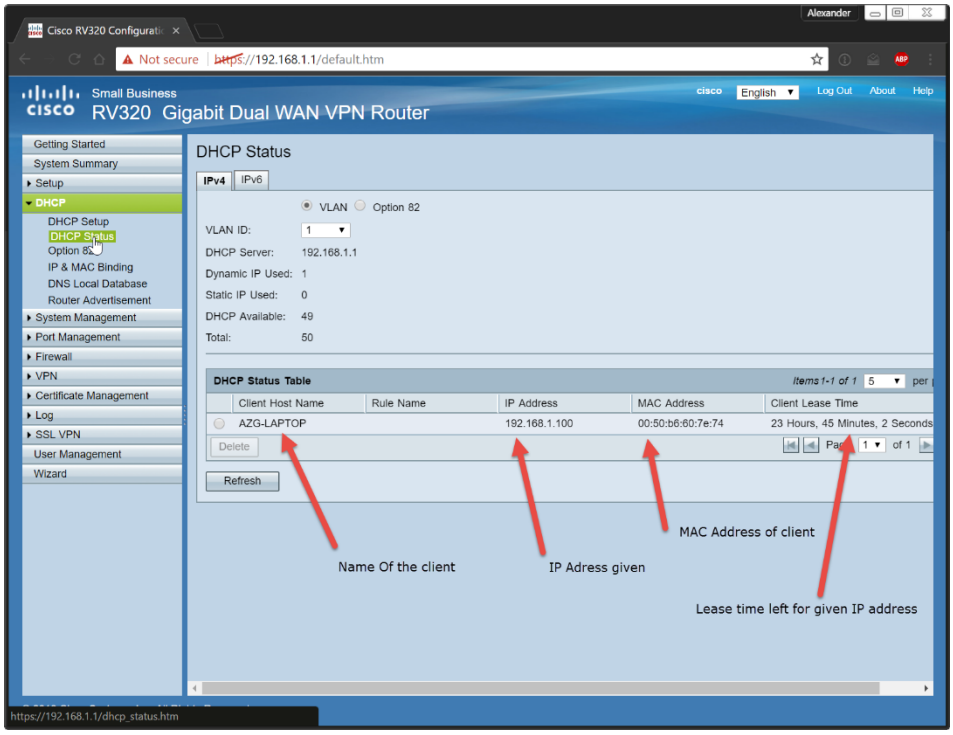


Figure 9 DHCP status page

Appendix C: Test Plan

Test Plan for IoT Modbus Application

The goal of the testing is to make sure the software is working according to the requirement specification and fully dressed use case documents. And uncover if there are any bugs in the software.

The important parts to test is to test if it is possible to establish and hold a connection open with the IoT device. Test if both digital and analog I/O are connected and are logically connected as stated in the requirement specification. Ability to obtain an IP address using DHCP and create a PDF report containing time period, traffic of messages and range of Modbus registers used. Also show Modbus exception code when sending a message with error.

The test method that will be used is functional and nonfunctional testing.

The test will be carried out by the developer of the software.

The testing environment should be on a computer running Windows 10. A intranet should be set up with a CISCO RV320 router with IP address 192.168.1.1. This router must have a DHCP server configured with a IP range between 100 and 150 and default lease time. Also, a IoT device must be part of the intranet. And all the devices in the intranet must be connected using Ethernet.

Since Unified Process is used as development method, testing is a part of each iteration. So the test cases below will be run after all the iterations are done as a final test.

As stated above, the following test cases are based on the functional and nonfunctional requirements for the software. The first column in the table contains the case. The second column shows the expected result from the case. The third column is used to mark if the test case gave the expected result. The fifth is to indicate if the case failed, and the sixth column is for comments.

Test Case	Expected result	OK	Failed	Comment
Connect IoT Device to router	Obtains an IP address from the DHCP server within the range specified.	X		
DI Upper switch is centered	No light on DO1 and DO2	X		
DI Upper switch is at 1	DO1 green light and DO2 no light	X		
DI Upper switch is at 2	DO1 no light and DO2 green light	X		

DI Lower switch is centered,	No light on DO3 and DO4	X		
DI Lower switch is at 3,	DO3 green light and DO4 no light	X		
DI Lower switch is at 4	DO3 no light and DO4 green light	X		
AI1 knob at middle at 0V	AO1 No light and 0 volt output	X		
AI1 knob turned left to -10V	AO1 red light and -10 volt output	X		
AI1 knob turned right to 10V	AO1 green and 10 volt output	X		
AI2 knob at middle at 0V	AO2 no light and 0 volt output	X		
AI2 knob turned left to -10V	AO2 red light and -10 volt output	X		
AI2 knob turned right to 10V	AO2 green and 10 volt output	X		
Push generate PDF button	A PDF file is generated containing time period, traffic of messages and range of Modbus registers used.		X	The field showing total time opened is showing the wrong value.
Write a unsupported function code	Show the exception code on screen and disconnect	X		

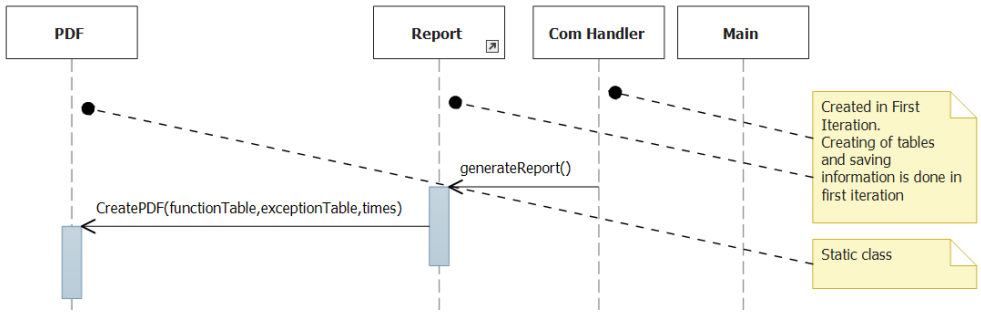
Appendix D: Documentation for the second, third and fourth iteration.

2. Iteration: Generate Report

Fully dressed Use Case:

Use case section	Comment
Use Case name	Generate Report
Scope	IoT Modbus
Level	User Goal
Primary actor	Button
Stakeholders and interests	User needs system to generate a report on Modbus Communication with IoT Device
Preconditions	Needs to be connected with IoT Device Using Modbus
Success guarantee	A generated report contacting information on Modbus Communication with IoT Device.
Main Success Scenario	A. User Input name and student number B. Save connection time C. Save transaction information D. Save disconnect time E. Save timespan connected F. Make table over transaction information G. Wait for button click H. Generate PDF report
Extensions	A1. Give an error if no username or id is inputted C1. If Modbus exception, save in own table with info on transaction H1. If file exists, overwrite existing file
Special req.	None
Technology list	
Frequency of occurrence	Once
Miscellaneous	None

Interaction diagram:

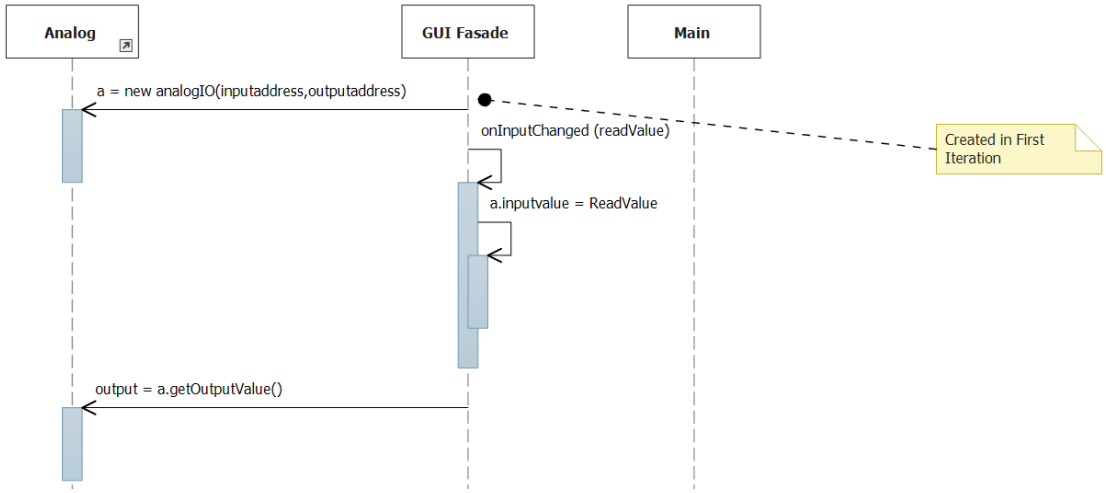


3. Iteration: Handle Analog I/O

Fully dressed Use Case:

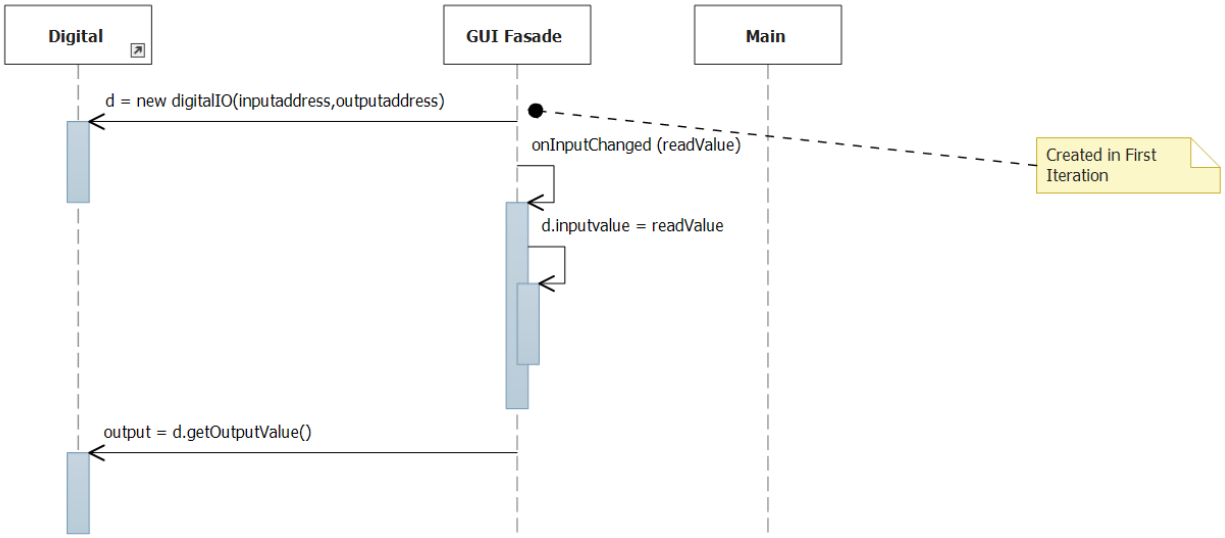
Use case section	Comment
Use Case name	Handle Analog I/O
Scope	IoT Modbus
Level	User Goal
Primary actor	IoT Device
Stakeholders and interests	User needs system to connect the analog input and analog output on IoT Device
Preconditions	Computer and IoT device are connected. And address of input and output register.
Success guarantee	Analog Input and Output on IoT Device are connected.
Main Success Scenario	A. Read input register and save to variable B. Set output register according to value read from input register
Extensions	A. If error, show error message B. If error, show error message
Special req.	None
Technology list	None
Frequency of occurrence	On every poll sampling time
Miscellaneous	None

Interaction diagram:



Use case section	Comment
Use Case name	Handle Digital I/O
Scope	IoT Modbus
Level	User Goal
Primary actor	IoT Device
Stakeholders and interests	User needs system to connect digital input and digital output on IoT Device
Preconditions	Computer and IoT device are connected. And address of input and output register
Success guarantee	Digital Input and Output on IoT Device are connected.
Main Success Scenario	C. Read input register and save to variable D. Set output register according to value read from input register
Extensions	C. If error, show error message D. If error, show error message
Special req.	None
Technology list	None
Frequency of occurrence	On every poll sampling time
Miscellaneous	None

Interaction diagram:



Appendix E: Example of Modbus PDF report

Modbus TCP Protocol

User: Alexander Zhang Gjerseth
ID: 121174
Date: 11.05.2017 10.36.05

Application was open on: 10.32.16 Application was closed on: 10.36.05
Application was open for: 00.09.59 Total Time Connected: 00.02.26

FUNCTION CODE	FUNCTION NAME	START ADDRESS	END ADDRESS	NUMBER OF FUNCTION CALLS
1	Read Coil			0
2	Read Discrete Input	0	4	229
3	Read Holding Register			0
4	Read Input Register	0	2	228
5	Write Singel Coil			0
6	Write Singel Register			0
15	Write Multiple Coils	54	6	230
16	Write Multiple Registers	0	2	228

EXCEPTION CODE	EXCEPTION NAME	TRANSACTION ID	FUNCTION CODE	TIMESTAMP
2	Illegal Data Address	944	143	11.05.2017 10.35.48

Appendix F: Proposed laboratory assignment

Proposed Laboratory Assignment with focus on Modbus, Network and IoT

Software and other resources needed to solve the assignment:

Visual Studio: https://www.visualstudio.com/downloads/
Wireshark: https://www.wireshark.org/#download
IoTModbus.exe
Modbus.DLL

Useful documents for solving this assignment:

MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3:
http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

SCE1306 OOADP Lecture Notes

Part 1: Theory

1. Connect the IoT Device to power and try changing position of switches and knobs. What is happening and why?

2. What is a communication protocol and why do we need to use one?

3. Read part 1-5 in the Modbus Specification 1.1b3 . Given a brief overview of the different types of Modbus and the differences between these types.

4. Modbus is a so-called Client/Server protocol. Give a brief overview of what this means and how it works.

5. Give a brief overview of addressing in Modbus.

6. Give a brief overview of the three messages types in Modbus. Also, discuss functions and exceptions.

7. Give a brief overview of what LAN is.

8. What is DHCP and what is the benefit of using DHCP?

9. What is IoT ? And how is IoT, Modbus TCP and networks related ? Give a brief answer.

Part 2: Network Setup

1. Connect your computer to the router and power on the router. Then log into the router's configuration interface using a web browser. Set username to "Student" and password to "netlab1".

2. Go into Make sure LAN address for router is 192.168.1.1 and configure DHCP to give out IP between 100 and 150.

3. Make sure your computer is set to obtain a IP automatically. Get the local ip address of your computer and show that it is given a IP inside the range set up in DHCP. Include a screenshot.

4. Connect the IoT Device to the router. By using the router's web interface find the IP of both the IoT Device and your computer. Also find the lese time for the IP addresses given

Part 3: Practical Modbus

1. Get action to be performed.

2. Create the PDU, MBAP and ADU for the action given and describe each part.

3. Open the Modbus IoT application and go to the function code tester. Perform the action and take a screenshot of output.

4. Go to Modbus Simulator screen. Try switching the position of switches and knobs. Is there any difference to what happened in step one in theory part?

5. Play around and get familiar with the IoT Device and Modbus Application.
6. Open Wireshark, and start listening on the network. Apply the Modbus filter and send a message using the Modus application. Select the message in Wireshark and take a screenshot of it. Is the captured package readable ?

7. In the Modbus application push the print button. This will generate a PDF report on the desktop of your computer. Include this document as an appendix in your report.

Part 4: Development of software for Modbus Communication

Develop a Modbus communication application using the DLL Modbus and DLL for performing the given action. And generate a report using the generateReport() method In ComHandler class.

Hint: First an object of Modbus.ComHandler. Then Connect and send. Also, the four events in ComHandler needs to be added. This is shown below.

```
ModbusCom.OnResponseData += new ComHandler.ResponseData(OnResponse);
ModbusCom.OnError += new ComHandler.ErrorData(OnError);
ModbusCom.OnOutData += new ComHandler.OutData(OnOutData);
ModbusCom.OnException += new ComHandler.ExceptionData(OnException);
```

Add your code and screenshot of application + report generated in your report.