Industrial Communication Systems, Assignment 1

Modbus Master TCP/IP

Valtteri Nikkanen 282688

Lauri Nuutinen 283219

Väinö Kurvi 274490

Table of contents

Introduction

Assignment is to build a python code using socket library that works as a Modbus master. Master sends requests to slave and slave responses accordingly to protocol. We are not allowed to use any higher-level libraries for Modbus TCP. The script configured so that it can be run locally.

Our master script supports the following requirements:

- Read coils
- Read Discrete inputs
- Read holding registers
- Read input registers
- Write single coil
- Write single register

When executed the master script request’s function code, address, quantity of registers and coils and value to set from the user. The script's output mimics the Wireshark display.

Theoretical Background

Modbus is a layer 5 protocol for moving simple data. In this instance the data was moved over TCP/IP protocol.

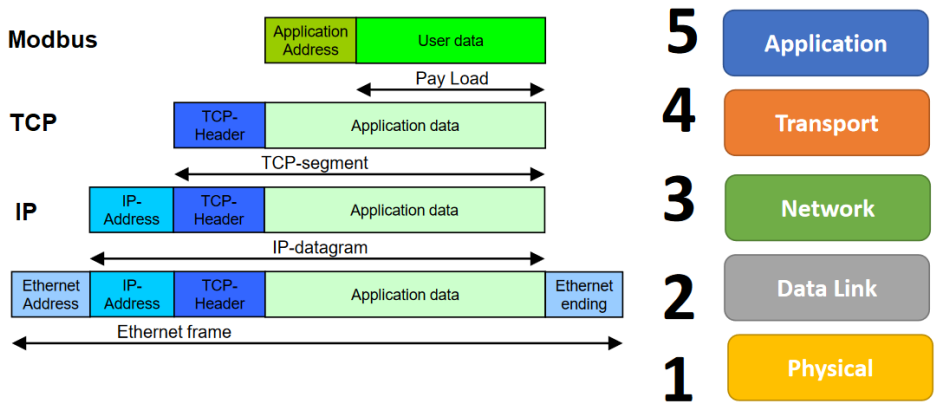

Figure 1: Protocol stack

Data is encapsulated in every layer and the layers protocol adds a header to it.

Modbus message consists of transaction identifier, protocol identifier, length, unit identifier, function code, register address and data.

Transaction identifier is used to synchronize messages between server and client. Protocol identifier implicates which version of Modbus is used. Length tells how many bytes are left in the message. Unit identifier is set to 0 in master and ignored in the slave client. Function code is sent by master and tells whether to read or write to the table.

CODE	FUNCTION	REFERENCE
01 (01H)	Read Coil (Output) Status	0xxxx
03 (03H)	Read Holding Registers	4xxxx
04 (04H)	Read Input Registers	3xxxx
05 (05H)	Force Single Coil (Output)	0xxxx
06 (06H)	Preset Single Register	4xxxx
15 (0FH)	Force Multiple Coils (Outputs)	0xxxx
16 (10H)	Preset Multiple Registers	4xxxx
17 (11H)	Report Slave ID	*Hidden*

Figure 2:

There are some examples what different function codes do in figure 2. Register address tells which table to access and where it is located in user data memory.

Reference	Description
0xxxx	Read/Write Discrete Outputs or Coils. A 0x reference address is used to drive output data to a digital output channel.
1xxxx	Read Discrete Inputs. The ON/OFF status of a 1x reference address is controlled by the corresponding digital input channel.
3xxxx	Read Input Registers. A 3x reference register contains a 16-bit number received from an external source—e.g. an analog signal.
4xxxx	Read/Write Output or Holding Registers. A 4x register is used to store 16-bits of numerical data (binary or decimal), or to send the data from the CPU to an output channel.

Figure 3: Reference number

Reference number depends on what function you want to use according to figure 3. Data section tells how many cells you want to read or what you want to write to the cells.

Instruction to run the script

To run the script you need to run the python script after which you will be greeted with instruction text asking you what you want to do as shown in figure 4.

```
C:\Users\vanik\PycharmProjects\ICS_Ass1>py modbus_master.py
What would you like to do?
  1- Read Coil (Output) Status
  2- Read Discrete Inputs
  3- Read Holding Registers
  4- Read Input Registers
  5- Write Single Coil
  6- Write Single Register
  q/Q to quit
 ---->
```

Figure 4: Instruction text in command line

You can then choose what feature of the Modbus master you want to use by writing the corresponding number to the command line. Here you can't write anything else without getting prompted again by the program.

If you choose the option 1 to read coil (output) status you will be next prompted by further questions about the register address and how many cells you would like to read as shown in figure 5. Here there is no real failsafe or checking to make sure you wrote the correct information so the program will crash if the wrong information is entered.

```
C:\Users\vanik\PycharmProjects\ICS_Ass1>py modbus_master.py
What would you like to do?
  1- Read Coil (Output) Status
  2- Read Discrete Inputs
  3- Read Holding Registers
  4- Read Input Registers
  5- Write Single Coil
  6- Write Single Register
  q/Q to quit
 ----> 1
Please enter the register address: 8000
Please insert how many you would like to read: 10
```

Figure 5: Read Coil (Output) Status prompts in command line

After this the program will print you out the corresponding registers or coils. This is shown in Figure 6.

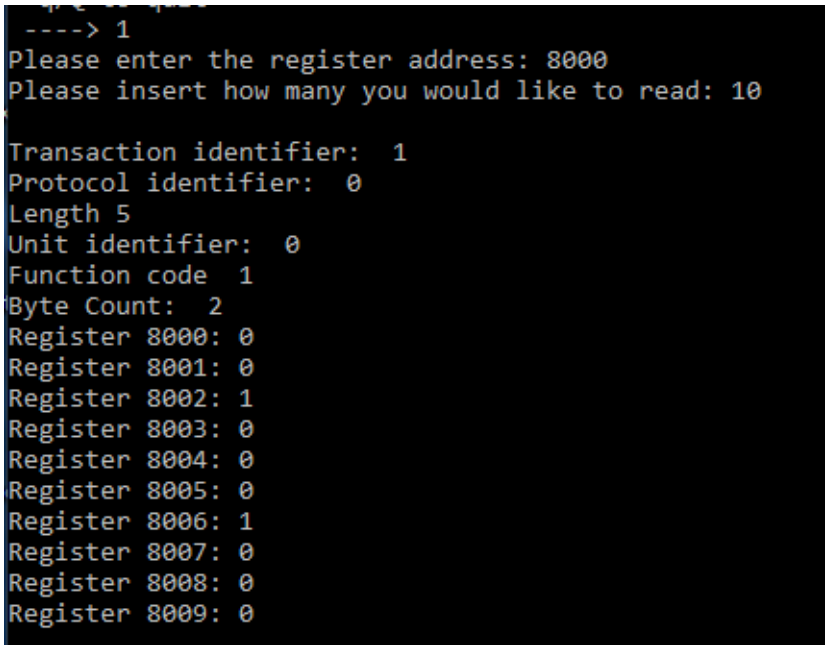

Figure 6: Showing the register values as they are in the Modbus slave

After all this the program goes back to the beginning and asks what you would like to do with it as was shown in figure 4.

If you choose options 6 or 7 or the writing options, you will be asked to specify what to write instead of how many registers to read. This is shown in Figure 7 and 8.

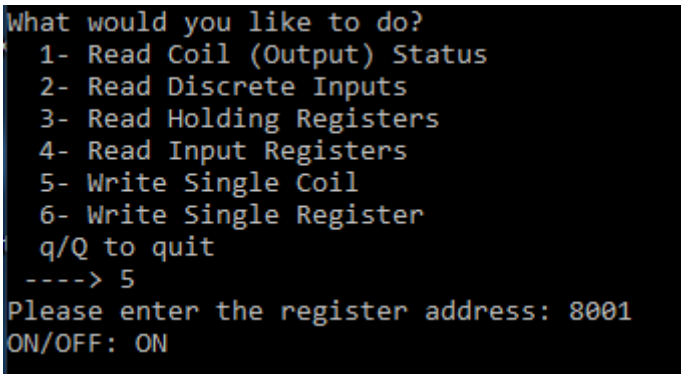

Figure 7: Write Single Coil in Command line

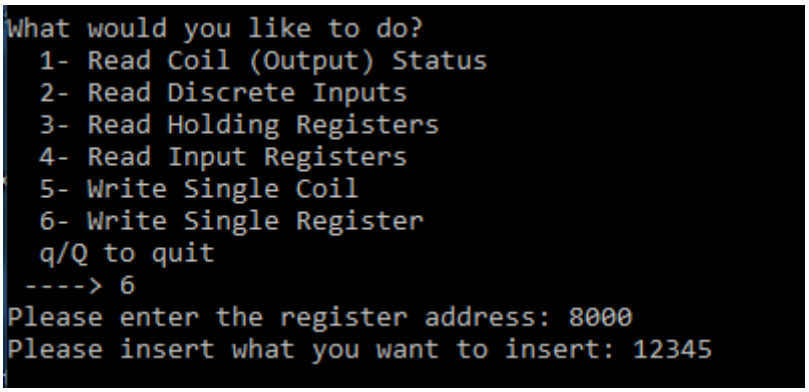

Figure 8: Write Single Register in Command line

After which the program will show you the status of the Modbus slave. In option 5 the coil is set to 1 if the given input is “ON” and off if the given input is anything else. In option 6 there is no checking if the given value is in the correct range. We thought this would be unnecessary as the tool is not aimed toward Modbus beginners.

Wireshark captures from supported communication modes

Here are Wireshark captures from each of the communication modes pictured next to the corresponding python commands, figures 9-14.

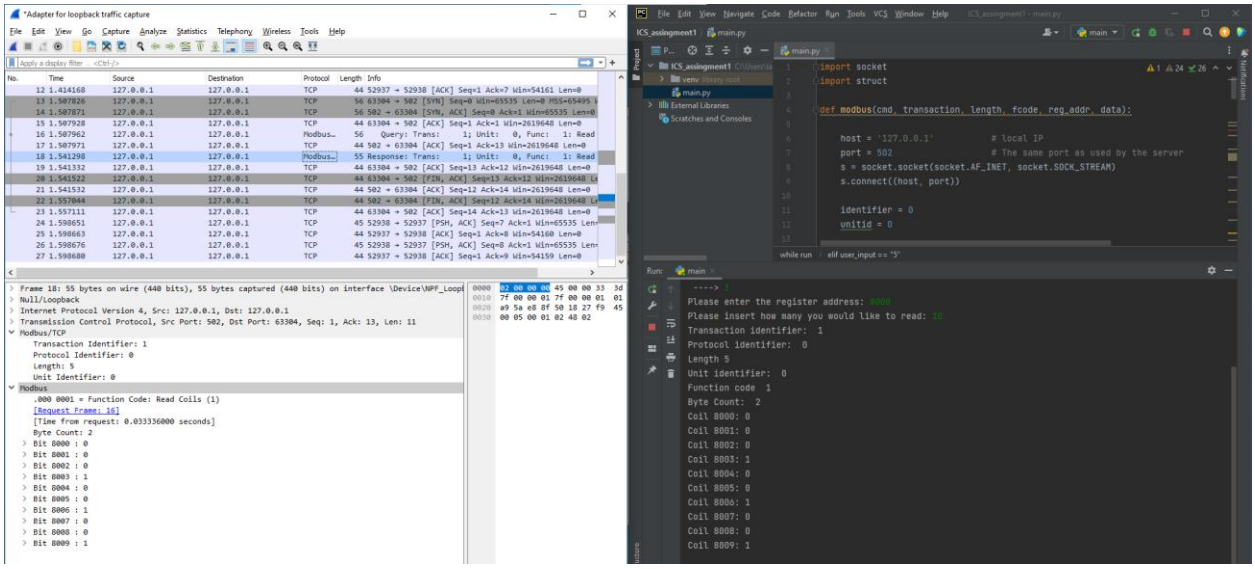

Figure 9: Wireshark capture of Function code 1 (Read Coil (Output) Status).

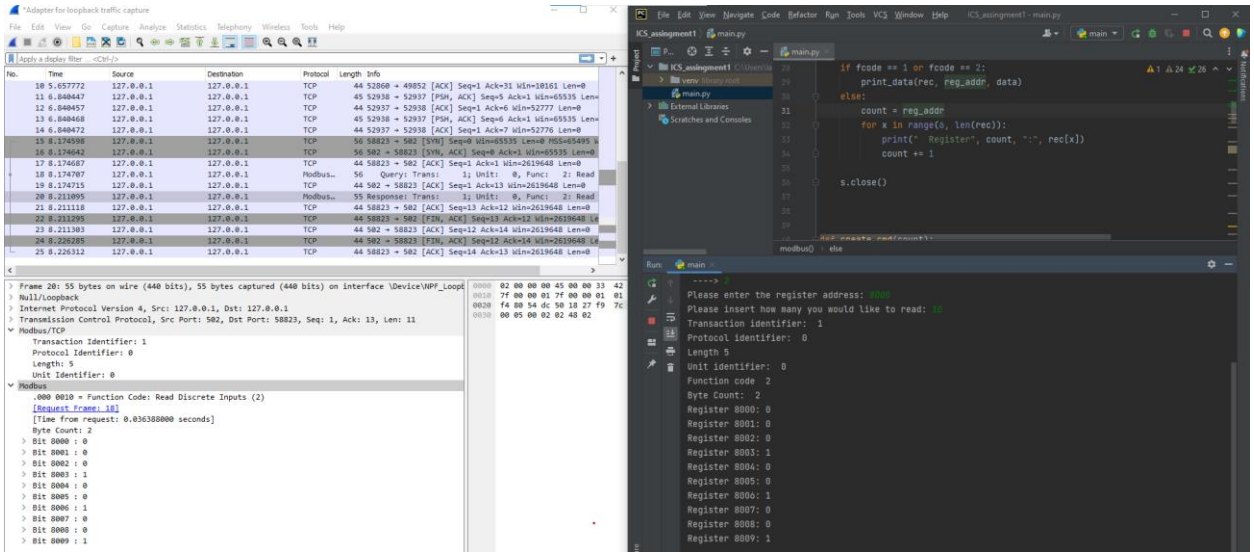

Figure 10: Wireshark capture of Function code 2 (Read Discrete Inputs).

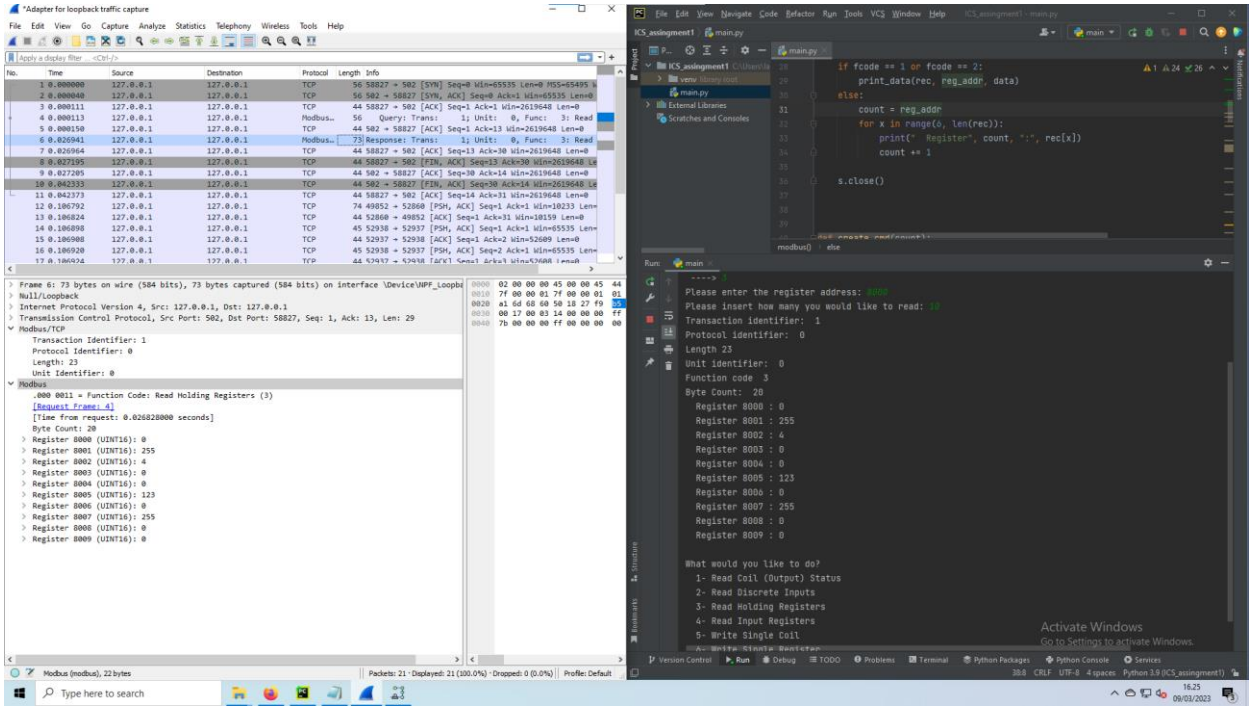

Figure 11: Wireshark capture of Function code 3 (Read Holding Registers).

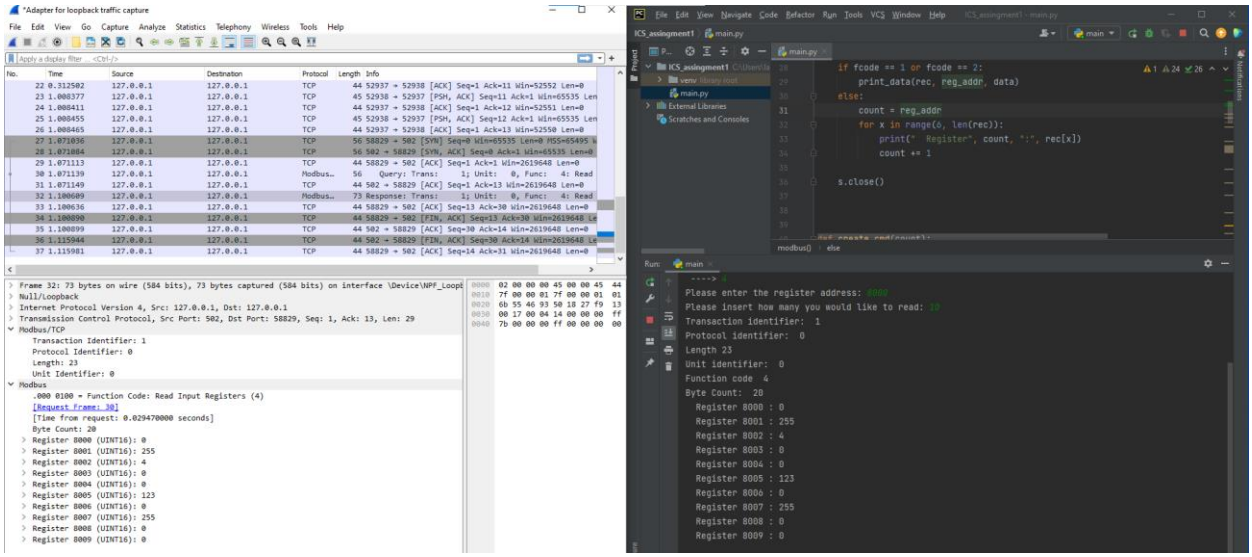

Figure 12: Wireshark capture of Function code 4 (Read Input Registers).

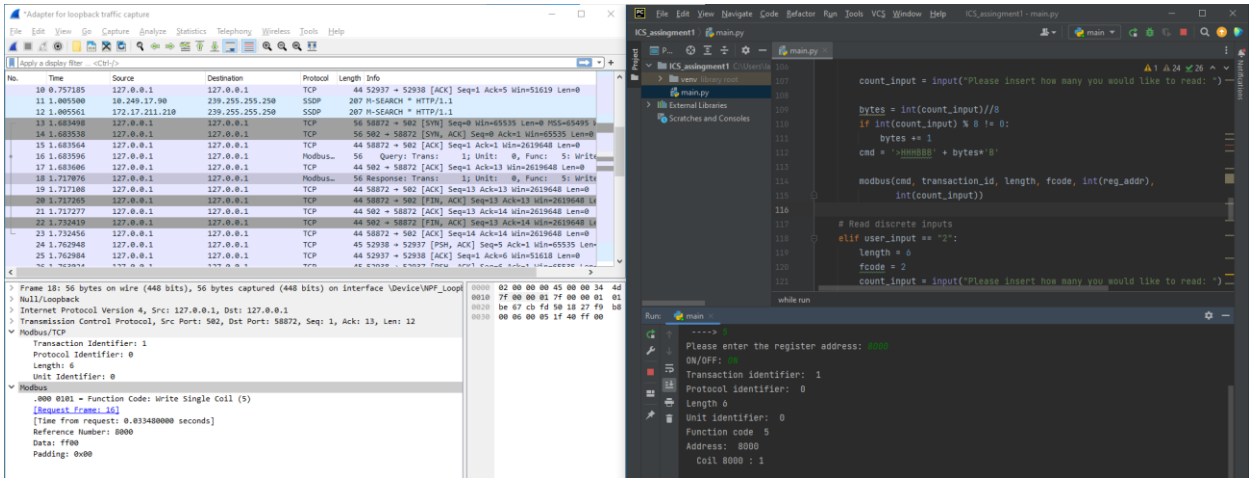

Figure 13: Wireshark capture of Function code 5 (Write Single Coil).

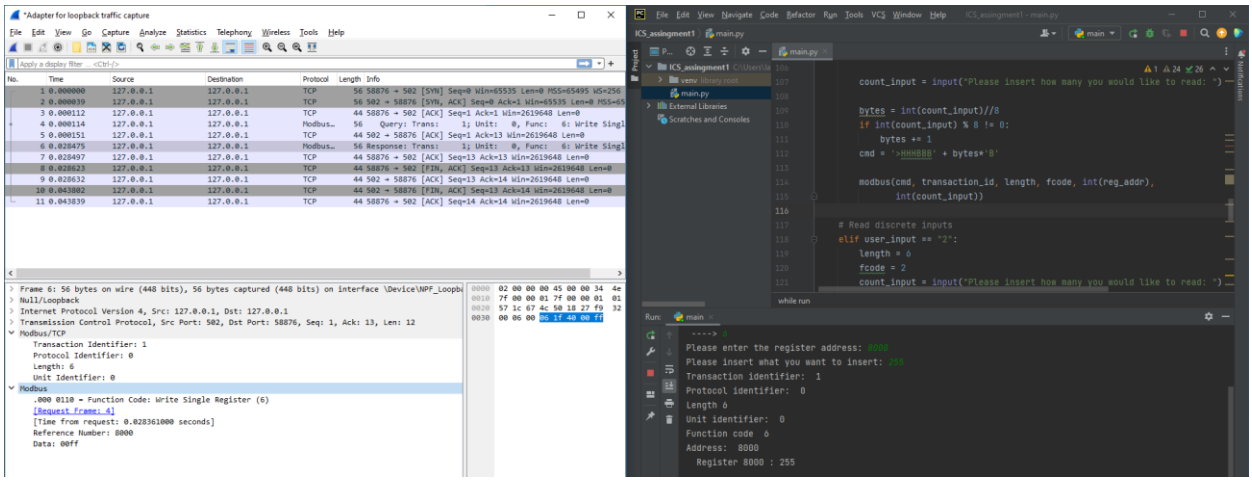

Figure 14: Wireshark capture of Function code 6 (Write Single Register).

Conclusions

All in all, Modbus is a simple protocol for transporting simple data. Our Modbus master program fulfills all the given requirements for the given assignment. The code should be simple to run and understand especially with the given instructions. Modbus Slave and Modbus Master tools are a pain to use in the trial form as the 10-minute time limit is annoying.