ICS TECHNOLOGIES

BASIC PLC PROGRAMMING AND NETWORKING

SOFT PLC LAB SOLUTIONS

Lab Description: In this lab, you will learn to program a software-based programmable logic controller, i.e., *soft PLC*, to control a lamp with multiple control contact switches. You will also learn to analyze industrial network traffic between the PLC and a Human Machine Interface (HMI) system, i.e., the "operator console".

This lab consists of the following exercises:

- Introduction to ladder logic and MODBUS I/O
- MODBUS TCP/IP messaging

Lab Environment: The lab environment requires a Linux host system running the Ubuntu 14.04.4 LTS distribution and the Labtainer framework. The Linux host can be a virtual machine and must have Internet access. The Labtainer framework can be installed and run as described in the Labtainer Student Guide, available at: https://my.nps.edu/web/cisr/labtainers.

This lab uses the Linux soft PLC that is supported by the open-source OpenPLC project [1]. This soft PLC runs in a Docker container in the Labtainer framework.

In addition to the Labtainer framework, you will need to install the following applications on the Linux host system:

1. PLCopen Editor v1.04 for MacOS and Linux. Available at: http://www.openplcproject.com/plcopen-editor. You will use this tool to create ladder logic programs to run on the PLC.
2. Radzio! Modbus Master Simulator. Available at: http://en.radzio.dxp.pl/modbus-master-simulator/. This lab is tested with Radzio! version 0.2.2 (build date: February 10, 2017). This tool communicates with the PLC via the MODBUS TCP/IP protocol [2,3]. You will use this tool as the HMI to control the lamp and contact

switches. This tool is a Windows program and is to be run with Wine (see below).

3. Wine Windows emulator. Binary package for Ubuntu 14.04.4 LTS can be installed using the *apt-get install* command. This lab is tested with Wine version 1.6.2. You will use this tool to run RadZio.
4. Wireshark network protocol analyzer. Binary package for Ubuntu 14.04.4 LTS can be installed using the *apt-get install* command. This lab is tested with Wireshark version 1.12.1. You will use this tool to capture and analyze MODBUS traffic between the PLC and Radzio.

Lab Configuration

Figure 1 shows the software components required by this lab.

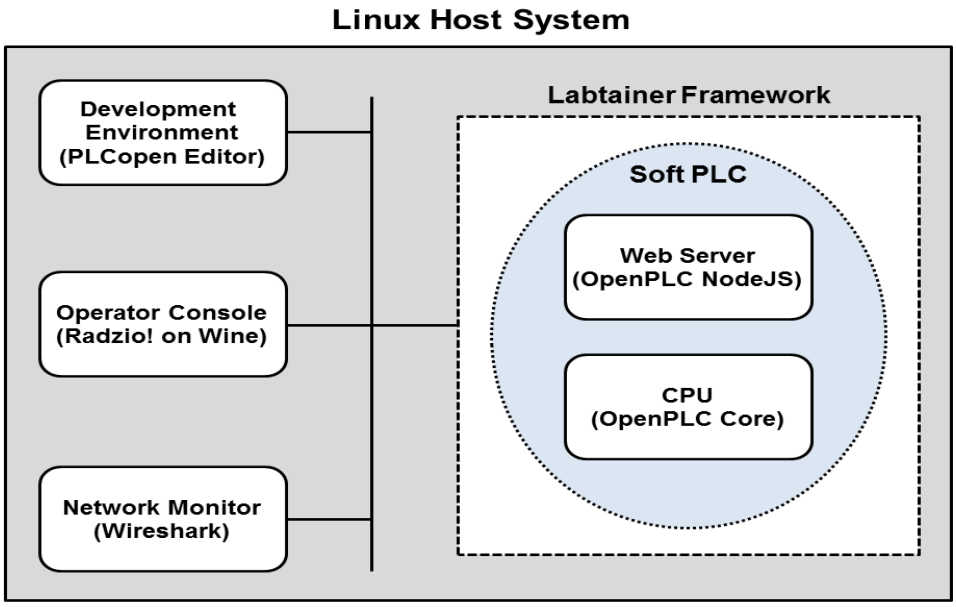


Figure 1. Lab configuration

The IP address used to communicate with the soft PLC varies and must be obtained after the lab is started (see section "Starting the Lab" below).

Lab Installation

To set up this lab, do the following on your host system:

1. Install the Labtainer framework. See the Labtainer Student Guide.

2. Install the PLCopen Editor tool. Download the ZIP archive from http://www.openplcproject.com/plcopen-editor. Uncompress the ZIP

archive (`PLCopen Editor v1.04 – Linux.zip`) into a permanent directory on your Linux system. The default unzipped directory name is "PLCopen Editor".

3. Download the sample Hello World ladder logic program (`Hello_World.xml`) from http://www.openplcproject.com/getting-started-linux. The program is at the bottom of the page. Copy the file to the directory where the PLCopen Editor tool resides (see step 2 above).

4. Install the following Python packages required by PLCopen Editor: *python-wxgtk2.8, pyro, python-numpy, python-nevow, python-matplotlib,* and *python-lxml.* You can use the following command:

   ```
   sudo apt-get install python-wxgtk2.8, pyro, python-numpy, python-nevow, python-matplotlib and python-lxml
   ```

5. Install the Wine Windows emulator.

   ```
   sudo apt-get install wine
   ```

6. Install the Radzio! Modbus simulator. Download the ZIP archive from http://en.radzio.dxp.pl/modbus-master-simulator/ to a permanent directory on your Linux system, e.g., ~/home/Radzio. Uncompress the ZIP archive (RMMS.zip).

7. Install the Wireshark protocol analyzer. Make sure to enable privileged users to run Wireshark.

   ```
   sudo apt-get install wireshark
   ```

What you need to know to do the lab

This lab assumes the following:

1. You have taken TCP/IP networking course(s);
2. You are familiar with basic terminology and concepts on ICS, MODBUS, and MODBUS TCP/IP;

3. You have hands-on experience with Wireshark. Minimally, the student must know how to filter a particular packet type, customize display columns, etc.

Note for Instructors

We suggest that this lab be conducted in a supervised lab environment, and that the following materials be covered at the beginning of the lab session:

1. Labtainer installation, including installation of Virtual Box and a Linux VM (if a Linux system is not already available). Refer to the Labtainer Student User Guide.
2. Review of ICS fundamentals.
3. Review of the MODBUS and MODBUS TCP/IP protocols.
4. Review of tool usage, i.e., PLCopen Editor and Radzio!

Starting the Lab

Read through the entire lab before beginning.

The lab is started from the Labtainer working directory on your host system. From there, issue the command:

```
./start.py softplc
```

The soft PLC automatically runs if the lab is successfully started.

In the lab's virtual terminal, use the `ifconfig` command to get the IP address of the `eth0` interface.

In the host system, do the following:

1. Launch the browser → navigate to `http://<eth0 address>:8080/` The Nodejs webserver should report that the PLC is running.
2. In the browser, select "View PLC logs" to see the PLC's log.
3. Open a terminal and navigate to the Radzio directory.
4. Launch Radzio! (`wine RMMS.exe`).
5. In Radzio, select Connection → Settings.
6. Select Modbus TCP and enter the IP address of the `eth0` interface and TCP port number `502` as shown in Figure 2. → Click OK.

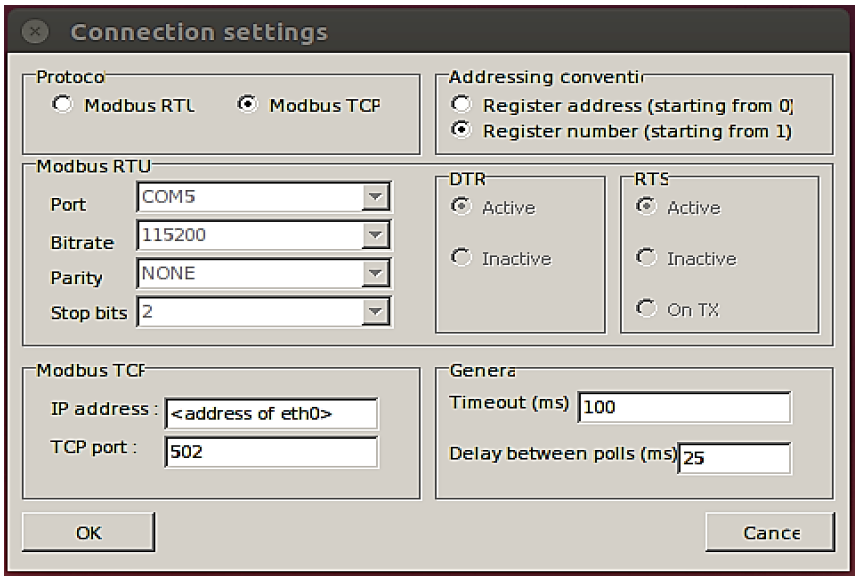


Figure 2. Radzio! configuration

7. Open another terminal and navigate to the PLCopen Editor directory.
8. Launch PLCopen Editor (`python PLCOpenEditor.py`).
9. Launch Wireshark and configure it to capture on the Docker interface (`docker0`).

OpenPLC I/O mapping

The MODBUS data model [2] consists of four data blocks with different characteristics. Figure 3 shows the data blocks and their corresponding data types. Each block can have up to 65536 data elements that are addressed from 0 to 65535.

Primary tables	Object type	Type of	Comments
Discretes Input	Single bit	Read-Only	This type of data can be provided by an I/O system.
Coils	Single bit	Read-Write	This type of data can be alterable by an application program.
Input Registers	16-bit word	Read-Only	This type of data can be provided by an I/O system
Holding Registers	16-bit word	Read-Write	This type of data can be alterable by an application program.

Figure 3. MODBUS Data Model [2]

The mapping between the MODBUS data blocks and the device I/O space is vendor specific [2]. Figure 4 shows the OpenPLC I/O address space [4].

Register Type	Usage	PLC Address	Modbus Address	Register Size	Value Range	Access
Coil Registers	Digital Outputs	%QX0.0 - %QX99.7	0 - 799	1 bit	0 or 1 OFF or ON	read and write
Discrete Input Registers	Digital Inputs	%IX0.0 - %IX99.7	0 - 799	1 bit	0 or 1 OFF or ON	read only
Input Registers	Analog Inputs	%IW0 - %IW99	0 - 1023	16 bits	0 to 65,535	read only
Holding Registers	Analog Outputs	%QW0 - %QW99	0 - 1023	16 bits	0 to 65,535	read and write

Figure 4. OpenPLC I/O address space[4]

This lab only uses the Coil Registers and two MODBUS functions: Read Coils (function code=0x04) and Write Single Coil (function code=0x05). Refer to the MODBUS specification [2] for more details on these functions. Note that the data field of a Read Coils response contains the status of multiple coils, one bit per coil [2].

Radzio!, by default, only queries the statuses of ten coils, numbered from 1 to 10. The MODBUS addresses of these coils are 0x00 through 0x09.

Status bit	Coil Number	MODBUS address
0	1	0
1	2	1
2	3	2
3	4	3
4	5	4
5	6	5
6	7	6
7	8	7
8	9	8
9	10	9

Lab Files that are Needed: The sample Hello World program must be downloaded from the OpenPLC project and stored in the same directory where the PLCopen Editor source resides.

EXERCISE 1: INTRO TO LADDER LOGIC AND MODBUS I/O

The Hello World program is a simple ladder logic program that uses a button (contact switch) to control a lamp (coil) (Figure 5). The button and lamp are attached to PLC addresses %IX0.0 and %QX0.0, respectively. When the button is pressed and released, the lamp turns on for 2 seconds and then turns off [5].

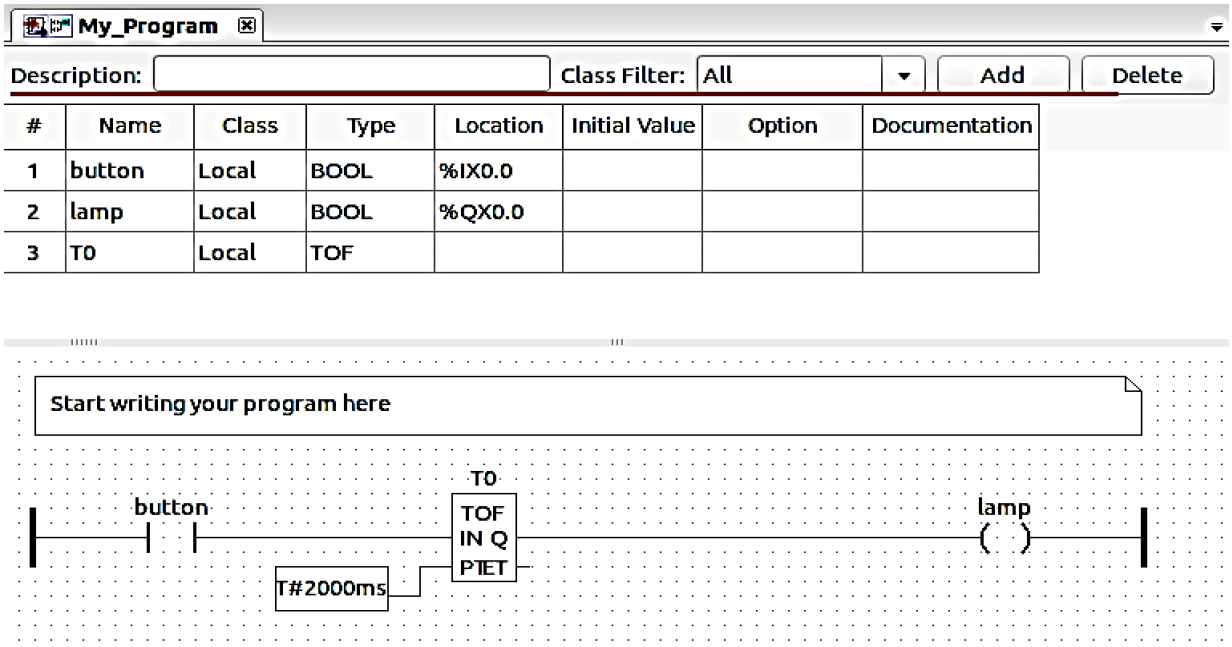


Figure 5. PCLopen Editor display of Hello World program

The objective of this exercise is to familiarize you with ladder logic programming and MODBUS I/O operations. The Hello World program has a configuration bug that renders the button ineffective. Specifically, the state of the button cannot be changed because the button is located in a read-only area, i.e., the Discrete Input Register area in OpenPLC address space (Figure 4).

Your task is to fix the bug so that the button can control the lamp and work through the questions below. For the lab report, you need to provide details using screen shots and description of the observed behavior.

Read through the entire exercise before beginning.

Activity 1: Fix the bug

In the host system, do the following:

1. In the PLCopen Editor, open the Hello World project (`Hello_World.xml`) stored in the PLCopen Editor directory.
2. Expand "Programs" and double-click "My_Program" to display the program in ladder logic.
3. Change the location of the button to `%QX0.1` (Figure 6).

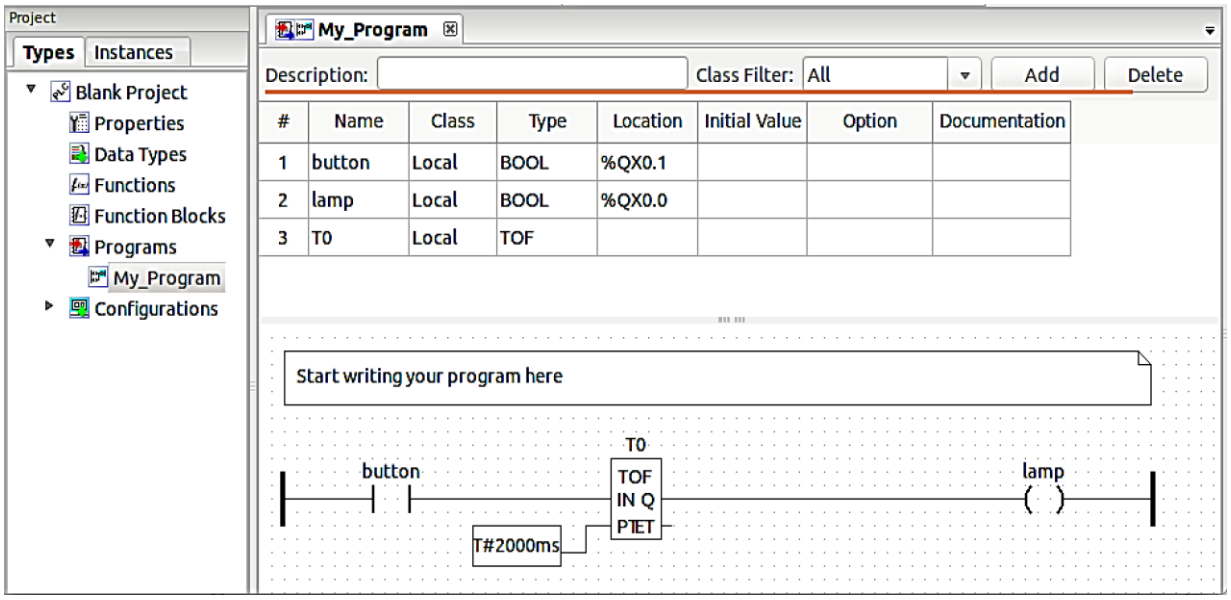


Figure 6. Modified Hello World program

4. Double-click the comment block → enter the following:
   "`<last name>: <brief statement about this activity>`"
5. Select File → Save As. Specify `<last name>-Hello-World-mod.xml` as the file name. Click Save.
6. Select File → Generate Program. Specify `<last name>-Hello-World-mod.st` as the file name, and select "ST files (*.st)" as the file type. Click Save.

The last step converts a graphical program into a Structured Text (ST) program, i.e., a program written in IEC 61131-3 Structured Text programming language [6]. OpenPLC only accepts input programs written in ST.

Activity 2: Upload the ST program to the PLC

In the host system, do the following:

1. In the browser, navigate to the home page.

2. Select "Browse" → select the `<last name>-Hello-World-mod.st` file. Click Open.
3. Select "Upload Program". The status string "Program compiled without errors!" should be returned.
4. Navigate back to the home page.
5. Select "View PLC logs".

Question 1.1: What are the last two messages recorded in the "OpenPLC Server" log?

In the Labtainer virtual terminal, view the Node.js server's log (`nodejs.log`) stored in the `/var/log` directory.

```
sudo tail -f /var/log/nodejs.log
```

Question 1.2: What is the last message recorded in `nodejs.log`?

Activity 3: Test the modified program

In the host system, do the following:

1. In Radzio, select File → New.
2. In the "ASCII display" box, click Enable.
3. In the "+1" row (MODBUS address 0 which is the output address %QX0.0), enter "lamp" in the Alias cell.
4. In the "+2" row (MODBUS address 0 which is the output address %QX0.1), enter "button" in the Alias cell (Figure 7).

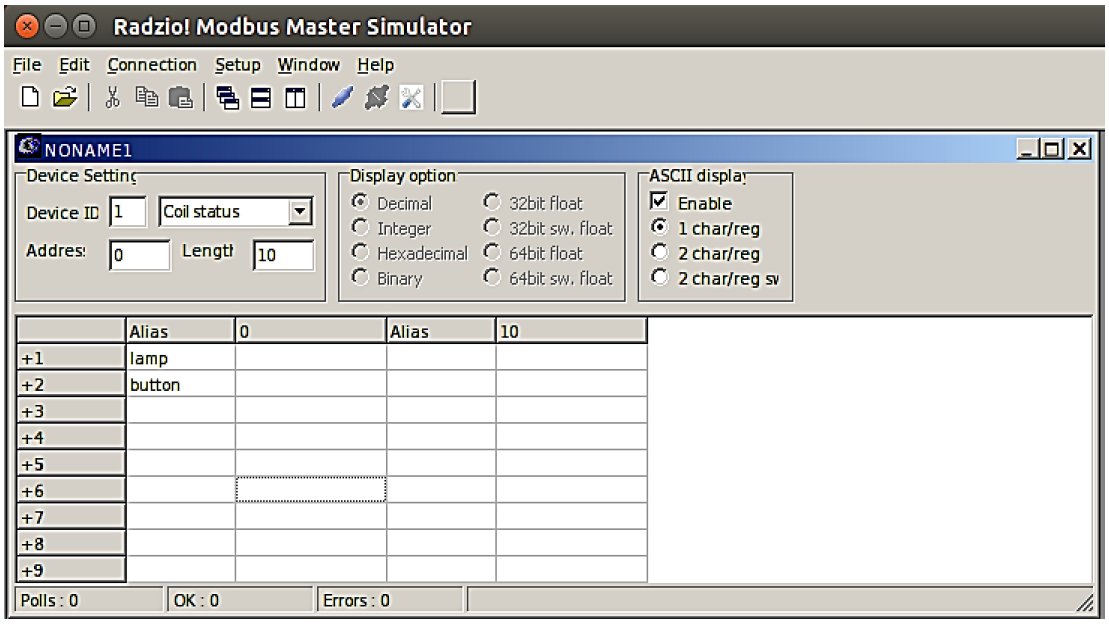


Figure 7. Console interface

5. Select File → "Save window as...". Specify Hello-World-mod as the file name. Click Save. This action will create a file named `Hello-World-mod.rmf` in the Radzio directory.
6. Select Connection → Connect. Once connected, Radzio! Starts polling the MODBUS devices. The state of each MODBUS address is reported in the cell next to the Alias cell.
7. In the browser, navigate back to the home page.
8. Select "View PLC logs".
9. Set the button variable to ON. In the "+2" row, double-click the cell after "button" (in the "0" column). A dialog window will appear. Click On → click Update.

   Question 1.5: What are the states of the lamp and button?

10. Set "button" to OFF.

    Question 1.6: What are the states of the lamp and button?

    Question 1.7: Was the state of the lamp changed immediately after the button is set to OFF? Explain the observed behavior.

11. Set "lamp" to ON.

Question 1.8: What are the states of the lamp and button? Explain the observed behavior.

12. Set "button" to ON.

    Question 1.9: What are the states of the lamp and button?

13. Set "lamp" to OFF.

    Question 1.10: What are the states of the lamp and button? Explain the observed behavior.

14. Select Connection → Disconnect.
15. In the browser, navigate back to the home page.
16. Select "View PLC logs".

    Question 1.11: What are the new messages recorded in the "OpenPLC Server" log?

In the Labtainer virtual terminal, view the `syslog` file.

```
sudo tail –f /var/log/syslog
```

Question 1.12: How many "WriteCoil…" messages are in the syslog? Describe the correlation between the WriteCoil messages and the actions taken in steps 9-13 above.

Deliverables

You need to submit the following files for this exercise:

1. A report (in PDF format) with answers to the listed questions. The report should include screen shots and explanation of the observed behavior.
2. The programming files:
   a. `<last name>-Hello-World-mod.xml`
   b. `<last name>-Hello-World-mod.st`
   c. `<last name>-Hello-World-mod.rmf`

EXERCISE 2: MODBUS TCP/IP MESSAGING

The MODBUS TCP/IP is used to transport MODBUS protocol data across a TCP/IP network. The MODBUS TCP/IP payload inside a TCP/IP packet consists of a MODBUS Application Protocol (MBAP) header and a MODBUS protocol data unit (PDU) [5] as shown in Figure 5 [3].

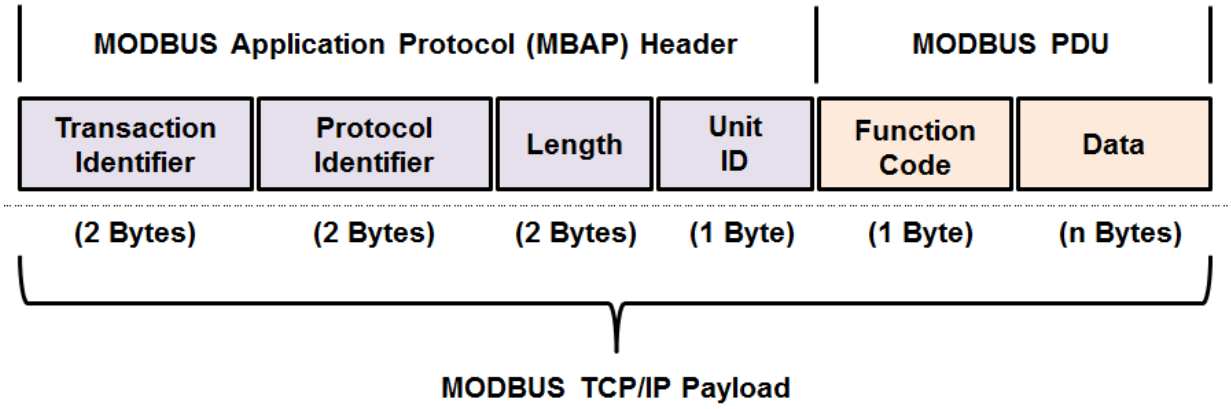


Figure 8. MODBUS TCP/IP payload format

The objective of this exercise to is to gain basic understanding of the MODBUS TCP/IP protocol. Your task is to create a simple ladder logic program to control a lamp using three contact switches, and use Wireshark to capture and inspect the MODBUS TCP/IP traffic between the Radzio and the PLC.

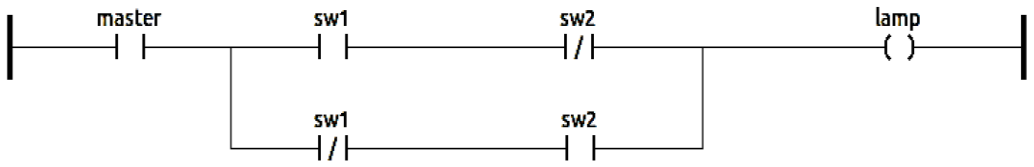
For the lab report, you need to provide details using screen shots (including Wireshark displays) and description of the observed behavior.

Read through the entire exercise before beginning.

Activity 1: Create a new program

In the host system, do the following:

1. Create a program that implements the following logic:



Coil/Switch	Address
lamp	%QX0.0
master	%QX0.1
sw1	%QX0.3
sw2	%QX0.5

Set all "Initial Value" to zero.

2. Save it as `<last name>-three-switches.xml`.
3. Generate the ST file and name it as `<last name>-three-switches.st`.
4. Upload the ST to the PLC.
5. Verify that the program is running on the PLC.

Activity 2: Test the new program

In the host system, do the following:

1. In Wireshark, start capturing packets on the Docker interface (`docker0`).
2. In Radzio, create a new file and program the MODBUS addresses used in your "three switches" program.
3. Save the file as `<last name>-three-switches.rmf`.
4. Connect to the PLC.

   Question 2.1: What are the states of the lamp and switches?

5. Set "master" to ON.

   Question 2.2: What are the states of the lamp and switches? Explain the observed behavior.

6. Set "sw1" to ON.

   Question 2.3: What are the states of the lamp and button? Explain the observed behavior.

7. Set "sw2" to ON.

   Question 2.4: What are the states of the lamp and button? Explain the observed behavior.

8. Set "sw1" to OFF.

   Question 2.5: What are the states of the lamp and button? Explain the observed behavior.

9. Set "sw2" to OFF.

   Question 2.6: What are the states of the lamp and button? Explain the observed behavior.

10. Disconnect Radzio.
11. Stop the Wireshark capturing. Save the packet capture to a file named `<last name>-three-switches.pcap`.

In the Labtainer virtual terminal, view the `syslog` file.

```
sudo tail –f /var/log/syslog
```

Question 2.7: How many "WriteCoil…" messages are in the syslog?

Activity 3: Analyze MODBUS TCP/IP traffic

In this activity, you need to analyze the PCAP file and correlate the captured Write Single Coil functions and the actions taken in Activity 2.

Show your work with screen shots of the PCAP data used to answer the questions.

In the host system, do the following:

1. Set Wireshark to use the MODBUS TCP/IP filter, i.e., `mbtcp`.
2. Locate the first Write Single Coil request.

Question 2.8: What is the TCP port number used by Radzio to communicate with the PLC?

Question 2.9: What are the values of the MBAP header fields?

Question 2.10: What are the values of the MODBUS PDU fields?

Question 2.11: To which action in Activity 2 does this packet correspond? Describe how the "Reference Number" and "Data" fields relate to this action.

Question 2.12: What are the values of the data bits returned in the Read Coils response that precedes the Write Single Coil request? Describe how the bit values correspond to this action.

Question 2.13: What are the values of the data bits returned in the Read Coils response that follows the Write Single Coil request? Describe how the bit values correspond to this action.

3. Locate the next Write Single Coil request.

   Question 2.14: What are the values of the MODBUS PDU fields?

   Question 2.15: To which action in Activity 2 does this packet correspond? Describe how the "Reference Number" and "Data" fields relate to this action.

   Question 2.16: What are the values of the data bits returned in the Read Coils response that precedes the Write Single Coil request? Describe how the bit values correspond to this action.

   Question 2.17: What are the values of the data bits returned in the Read Coils response that follows the Write Single Coil request? Describe how the bit values correspond to this action.

4. Locate the next Write Single Coil request.

   Question 2.18: What are the values of the MODBUS PDU fields?

Question 2.19: To which action in Activity 2 does this packet correspond? Describe how the "Reference Number" and "Data" fields relate to this action.

Question 2.20: What are the values of the data bits returned in the Read Coils response that precedes the Write Single Coil request? Describe how the bit values correspond to this action.

Question 2.21: What are the values of the data bits returned in the Read Coils response that follows the Write Single Coil request? Describe how the bit values correspond to this action.

5. Locate the next Write Single Coil request.

   Question 2.22: What are the values of the MODBUS PDU fields?

   Question 2.23: To which action in Activity 2 does this packet correspond? Describe how the "Reference Number" and "Data" fields relate to this action.

   Question 2.24: What are the values of the data bits returned in the Read Coils response that precedes the Write Single Coil request? Describe how the bit values correspond to this action.

   Question 2.25: What are the values of the data bits returned in the Read Coils response that follows the Write Single Coil request? Describe how the bit values correspond to this action.

6. Locate the next Write Single Coil request.

   Question 2.26: What are the values of the MODBUS PDU fields?

   Question 2.27: To which action in Activity 2 does this packet correspond? Describe how the "Reference Number" and "Data" fields relate to this action.

   Question 2.28: What are the values of the data bits returned in the Read Coils response that precedes the Write Single Coil request? Describe how the bit values correspond to this action.

Question 2.29: What are the values of the data bits returned in the Read Coils response that follows the Write Single Coil request? Describe how the bit values correspond to this action.

7. In the browser, navigate back to the home page.
8. Select "Stop" to stop the PLC.

Deliverables

You need to submit the following files for this exercise:

1. A report (in PDF format) with answers to the listed questions. The report should include screen shots and explanation of the observed behavior.
2. The programming files
   a. `<last name>-three-switches.xml`
   b. `<last name>-three-switches.st`
   c. `<last name>-three-switches.rmf`
3. The PCAP file `<last name>-three-switches.pcap`.

WHAT TO SUBMIT

When you have completed the lab, run the following command from the Labtainer workspace directory:

./stop.py softplc

Submit the resulting ZIP file along with a ZIP archive containing all deliverables listed in each exercise above.

REFERENCES

[1] The OpenPLC project. http://www.openplcproject.com/

[2] Modbus Organization, "MODBUS Application Protocol Specification V1.1b3," April 26, 2012. http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[3] Modbus Organization, "MODBUS Messaging on TCP/IP Implementation Guide V1.0b," October 24, 2006. http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf

[4] The OpenPLC Project, "SCADA Supervisory Software". http://www.openplcproject.com/scada

[5] The OpenPLC Project, "Running the first project". http://www.openplcproject.com/getting-started-linux

[6] PLCopen, "Introduction into IEC 61131-3 Programming Languages". http://www.plcopen.org/pages/tc1_standards/iec_61131_3/

This lab was developed by the Center for Cybersecurity and Cyber Operations at the Naval Postgraduate School in Monterey California by Thuy D Nguyen.