PLC and Modbus

1 Overview

This exercise explores the use of the Modbus protocol and a PLC to control a simple device. It is assumed the student has had an introduction to PLC's, ladder logic and the Modbus protocol. It is also assumed the student has some experience with Wireshark.

Details of the Modbus protocol can be found at: `https://http://modbus.org/docs/PI_MBUS_300.pdf` A tutorial on the use of Ladder Logic is at: `//https://www.plcacademy.com/ladder-logic-tutorial/`

2 Lab Environment

This lab runs in the Labtainer framework, available at http://my.nps.edu/web/c3o/labtainers. That site includes links to a pre-built virtual machine that has Labtainers installed, however Labtainers can be run on any Linux host that supports Docker containers.

From your labtainer-student directory start the lab using:

```
labtainer softplc2
```

Links to this lab manual will be displayed.

3 Lab topology

The lab includes three components as shown in Figure 1:

- Slave Device A simple slave device containing two buttons and a lamp. The device has an ethernet interface via which it is connected to a PLC using Modbus TCP/IP.
- PLC A programable logic controller implemented using the OpenPLC Soft-PLC implementation on a Linux based computer. The PLC is connected to the slave device via Modbus TCP/IP over ethernet. The PLC has a seperate ethernet connection to an HMI computer. For convenience of the lab, the PLC computer includes Wireshark and will be used to capture, review and replay network traffic between the PLC and the slave device.
- HMI A computer that interacts with the PLC using a web browser to load programs and monitor its operation. This computer also contains an OpenPLC Editor for constructing programs using "Ladder Logic" (LD).

The PLC and the slave device communicate via Modbus TCP/IP, with the PLC acting as the master. The HMI component in this topology does not use Modbus, rather it simply interacts with the PLC via the PLC's web server.

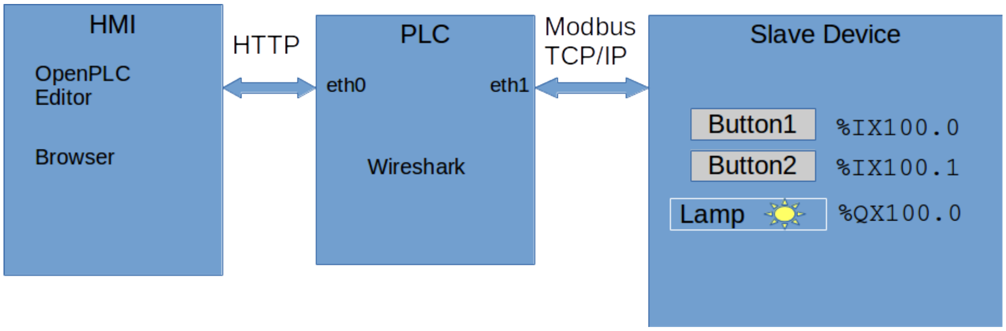


Figure 1: Soft PLC Lab Topology

4 Tasks

In this lab, you will create a simple “hello world” Structured Text program using Ladder Logic. You will upload this program from the HMI to the PLC and use it to control the slave device. You will use Wireshark to observe traffic between the PLC and the slave device.

4.1 Explore

Find the `Physical Board` window. This is your virtual slave device. Click the two buttons and confirm nothing happens. This board is connected to the PLC via Modbus TCP/IP, however the PLC is not yet running and is not yet programmed.

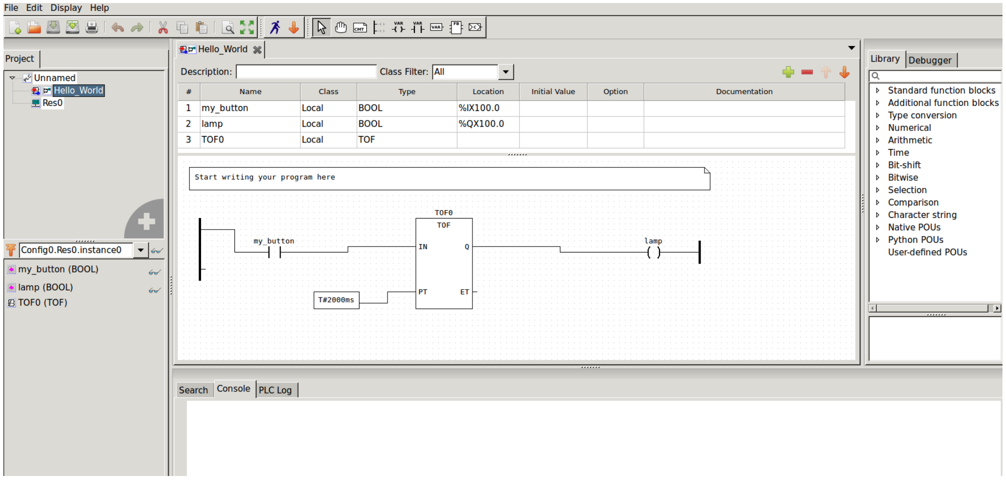
4.2 Build and run PLC program

In this task, you will deploy an existing “hello world” program onto the PLC and demonstrate its use.

4.2.1 Build the Hello World Program

On the HMI window, start the OpenPLC Editor in the background:

```
OpenPLC_Editor/openplc_editor.sh &
```



Description: Class Filter: All

#	Name	Class	Type	Location	Initial Value	Option	Documentation
1	my_button	Local	BOOL	%IX100.0			
2	lamp	Local	BOOL	%QX100.0			
3	TOF0	Local	TOF				

Figure 2: OpenPLC Editor

When the editor window displays, use `File / Open` and select `hello`. Then double-click on the `Hello World` entry in the project pane on the upper left of the window. This is an example of a LD program that states that if the button is pushed, then the lamp should be lit and remain on for 2 seconds after the button is released.

Use the orange down arrow on the menu bar to generate the program for the OpenPLC runtime. When prompted, save the file in a file with a `.st` file extension. Make a note of the directory where you saved the file. This operation has converted the LD program into a Structured Text (ST) program, which is the form that OpenPLC works with.

You may now close or minimize the OpenPLC Editor. Back on the HMI terminal, find the ST file that you saved and display its contents. That is the program the PLC will run in order to control the slave device. As you can see, ST programs have recognizable programming constructs. You will be asked to modify this program later.

4.3 Start the PLC user interface & Explore

On the HMI terminal, use the firefox browser to access the PLC using port 8080:

```
firefox plc:8080 &
```

Login to the PLC through the browser with `openplc` as the username and password. You should now see the Open PLC web page that controls the PLC. Click the `Slave Devices` button on the left. And double click on the `myserver entry`. This is the slave device that your PLC program will control. Note we define only 3 bits to control the slave device. Two bits as *Descrete Inputs* for two buttons (only one of which is initially used), an one bit for the lamp as *Coils*. Make a note of the slave device IP address and the port number used to communicate using Modbus.

4.3.1 Upload the ST program to the PLC

Select `Programs` on the left pane and use the `Browse` button to locate your ST file. Then upload it to the PLC. Provide a `Name` in the resulting form and press the `Upload program` button. Note the program is compiled on the PLC and results are displayed in the browser. Then click the `Go to Dashboard` button.

The PLC now has your hello world program loaded, but the PLC is not yet running. Click the Start PLC button. Once the PLC is started, click the Monitoring button.

4.3.2 Test the PLC program

On the Physical Board, click button1. The light bulb should come on for a few seconds. Observe the indicators on the browser Monitoring page.

4.4 Observe Modbus traffic

The PLC component includes Wireshark for monitoring the PLC network traffic. Find the plc window, and start wireshark:

```
wireshark &
```

Review the Modbus traffic. Note that Wireshark automatically detects the Modbus traffic, and displays the protocol. Note how all of the Modbus exchanges are either queries from the PLC or responses from the slave. Click button1. Then stop the Wirshark capture (red square button on the menu).

Locate a packet that directed the lamp to light. Note using a display filter of modbus.data==1 will filter the packets to include only queries that set the coil (lamp) to on, and responses that reflect the button is pressed. Once you've identified and selected the packet, click on the Modbus/TCP field and save the raw bytes to a file named on.raw using File / Export Packet Bytes.

Then locate a packet that directed the lamp to turn off. Save its Modbus/TCP raw bytes to a file named off.raw.

4.5 Replay coil commands

Stop the PLC if it is running (click the Stop PLC button in the browser). In the plc window, locate the files you just saved. Then use netcat to replay them to the slave device.

```
cat on.raw | nc <ip> <port>
cat off.raw | nc <ip> <port>
```

Where ip and port are the IP address and port of the slave device. Note that the capturing and replaying of TCP packets is performed on the PLC computer for convenience of the lab. It could occur on any component with access to the network used by the PLC to communicate with the slave device.

4.6 Modify the ST program

Stop the PLC if it is running (click the Stop PLC button in the browser). Go to the HMI terminal. Edit your ST file and change the delay from 2000 ms to 10,000 ms. Back on the browser, go to programs and upload the ST file again, this time giving the program a different name. Start the PLC and click button1. Confirm that your program change has modified the PLC operation.

4.7 Modify the LD program

Stop the PLC and go back to the HMI terminal and return to the OpenPLC editor. Use the editor to define a second button such that pressing either button will cause the lamp to light. You may wish to first use File / Save As / Create Folder to create a copy of the hello world program. To edit the LD program:

- Add button to table Click my_button field in the table and then click the green + icon to add a row to the table of items. Change the name of the new item to my_button2. Note the editor has assigned the very next Location value, which is what we want.
- Add button to schematic Click the contact icon in the menu and then click on the diagram where the new button is to go. Assign my_button2 as the variable name.
- Connect button Click on the unused lead of the power rail and drag a line to the left lead of the button. Then click the right lead of the button and drag a line to the In pin of the TOF0 (Timer Off-Delay) component.

Then save your changes and use the orange down arrow to generate a new ST file and save it. Load your new program into the PLC and run it. You should now be able to use either button to light the lamp.

5 Submission

After finishing the lab, go to the terminal on your Linux system that was used to start the lab and type:

```
stoplab
```

When you stop the lab, the system will display a path to the zipped lab results on your Linux system. Provide that file to your instructor, e.g., via the Sakai site.