LAB 1: INTRODUCTION TO SCADA CONTROL SYSTEMS

Estimated Time: 1 hour and 40 minutes

Purpose: The purpose of this Lab exercise is to familiarize the student with the virtual SCADA Control Systems, and begin configuring the virtual environment that will be used throughout the remaining labs.

Objective: The student will research software, protocols and tools used by SCADA Control Systems and the components within the UAH environment. The student should be able to operate the control system environments at the conclusion of this lab.

Lab Setup and Requirements: The student will need a browser with internet access to conduct research and download any necessary lab files. This lab assumes the student has previous experience using computer simulations.

EXERCISE #1 - INTRODUCTION TO THE UAH SCADA AND DOCKER ENVIRONMENT

The UAH environment will utilize Virtual Box to simulate two industrial control systems:

1. One Water Storage Tank
2. One Station Gas Pipeline

The Water Tank and Gas Pipeline can only be run individually.
Login for the virtual machine = username:ccre, password:ccre

The following scripts are provided on the virtual machine: /home/ccre/Scada lab/scripts. These scripts will be utilized throughout the labs, so familiarize yourself with the content and usage of each script.

1. gaspipeline.sh - Stops & removes all containers, and then starts the gas pipeline simulation
2. watertank.sh - Stops & removes all containers, and then starts the Water Tank simulation
3. netstart.sh - Initialize the virtual network for the simulation (Netstart should only be run once during initial setup of the environment. In addition, it will fail if there are any existing containers attached to either network)
4. netstop.sh - Stops the virtual networks
5. cleanup.sh - Brute Force, removes all containers

The following references are provided to supplement or provide additional information on the tools, protocols, and software used within this environment.

- OpenPLC
- HMI
- SCADABR
- MATLAB
- ModBus
- Virtual Box
- OpenPLC Editor

EXERCISE #2 - SETUP SCADA LAB ENVIRONMENT

Section 1: Import the virtual machine into virtual box.

1. Download and install Virtual Box on your host machine.
2. Copy scadalab.ova file from the UAH Cybersecurity SCADA Labs disc provided.
3. In the Virtual Box Manager, select File>Import Appliance to import the virtual machine.
4. Click "Choose a virtual appliance file to import" icon and browse to the scadalab.ova. Click "Import".

Section 2: Login to virtual machine and run Water Tank Docker containers.

1. To start the virtual machine in the VirtualBox Manager, select the scadalab VM, right-click and select Start>Normal Start. Login using the credentials provided in Exercise 2 (username:ccre, password:ccre).
2. Open Terminal by clicking on Applications -> Terminal Emulator
3. Navigate to the scripts folder with the command:

```
cd /home/ccre/scadalab/scripts
```

4. Run the script to configure the network with the command:

```
./netstart.sh
```

   If it asks for a password, type: ccre

5. Start the Water Tank simulation with the command:

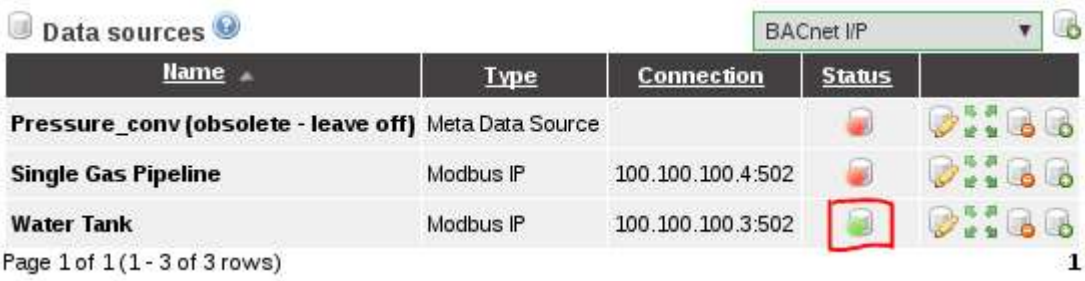
```
./watertank.sh
```

6. Launch the water tank HMI by opening the internet browser (Applications > Web Browser) and navigate to:

   100.100.100.2:8080/ScadaBR

   Login to ScadaBR using username:admin, password:admin

7. Click on Data Sources on the top menu and then enable water tank data sources to allow ScadaBR to pull data from OpenPLC.



8. Click on Graphical Views, select the Water Tank HMI from the drop down menu.

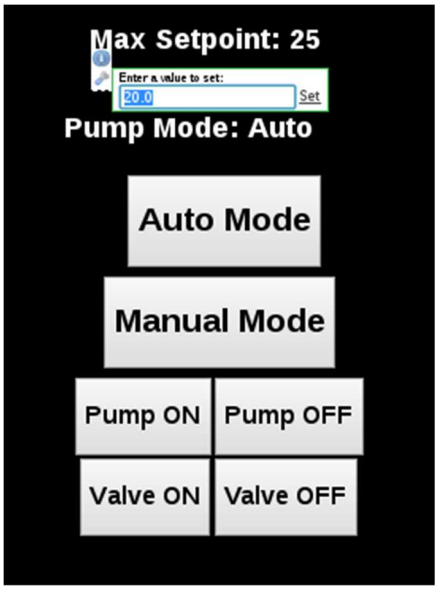
9. Verify the levels are changing on the Water Tank HMI.

CHECKPOINT: Refer to Troubleshooting Guide for FAQs regarding configuration and setup.

EXERCISE #3 - INTERACT WITH HMI AND CONFIGURE FOR UNSAFE OPERATIONS

Section 1: Interact with the Water Tank HMI

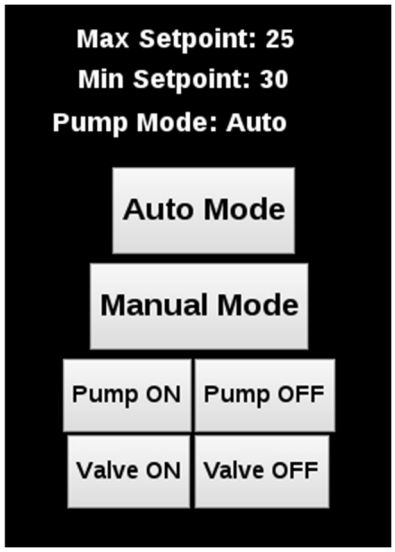
1. Pull up the Water Tank HMI in ScadaBR. Turn the Pump On by turning on Manual Mode and clicking the "Pump On" button. Wait ~30 seconds and observe the water tank activity. Now turn the Pump Off by clicking the "Pump Off" and observe the activity.
2. Turn the HMI on Auto and observe the fluctuations in water levels.
3. Change setpoints to Minimum=20 and Maximum=25 and observe difference when range is narrowed.



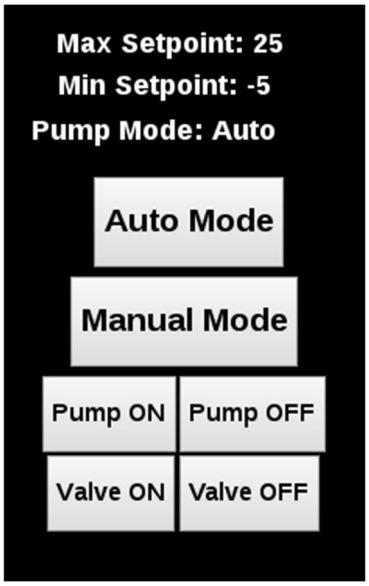
Section 2: Configure Water Pump for Unsafe Operation

The following modifications could be made by an attacker which would cause undesirable results.

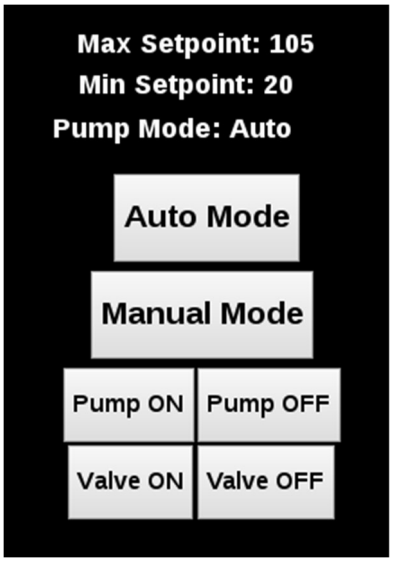
1. Attack 1: Set the Minimum setpoint more than the Maximum setpoint Expected result: This turns the pump on/off frequently which might damage the pump. The level stabilizes at maximum setpoint



2. Attack 2: Set the Minimum setpoint to -5. Expected Result: Tank completely empties. The pump never starts once the level reaches zero



3. Attack 3: Maximum setpoint set to more than 100. Expected Result: HMI becomes unreliable and the tank overflows



EXERCISE #4 - INTRO TO LADDER LOGIC

This exercise will introduce basic Ladder Logic and familiarize students with how Ladder Logic can be created using the PLCopen Editor.The PLCopen Editor is a software that enables you to write PLC programs, which contains the logic that the PLC must execute.

This lab will be conducted using the virtual machine and Water Tank Docker Containers
SECTION 1: Modify Water Tank Ladder Logic to Add Alarm

1. Open a terminal (Applications -> Terminal Emulator) and navigate to PLCOpen Editor folder:

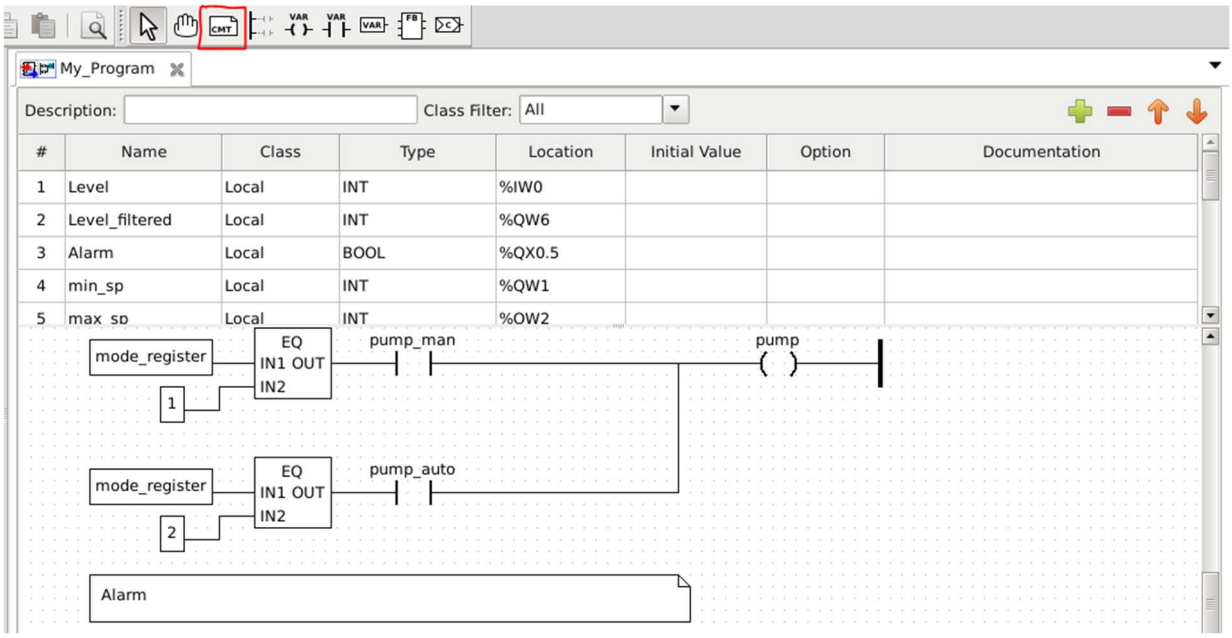
```
cd /home/ccre/scadalab/lab1/editor/
```

2. Run PLCopen Editor with the command:

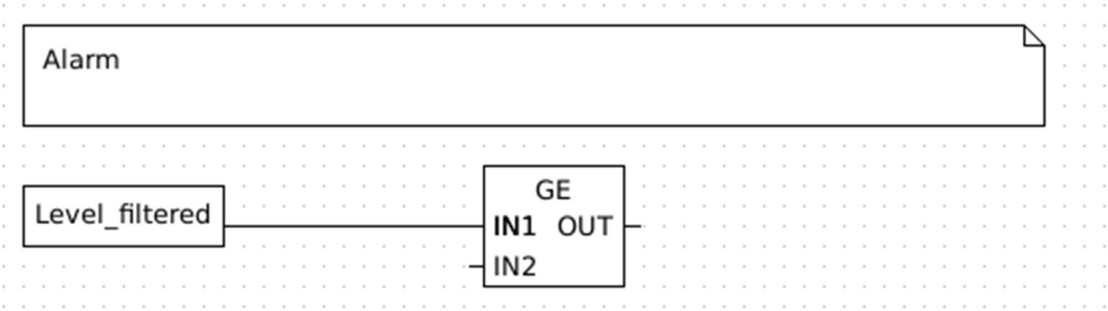
```
python PLCOpenEditor.py
```

3. In the PLCopen Editor, select File>Open. Navigate to ccre/scadalab/lab1 folder (you can find the ccre folder on the left sidebar). Double click the Water_Tower.xml file in the Lab 1 Directory.
4. Double-click on "My Program" and scroll to the bottom of the drawing area to add in the new logic for the alarm.

5. Select the "CMT" icon from the menu bar, and draw a new block at bottom and add "Alarm" in the Comments. Select OK.

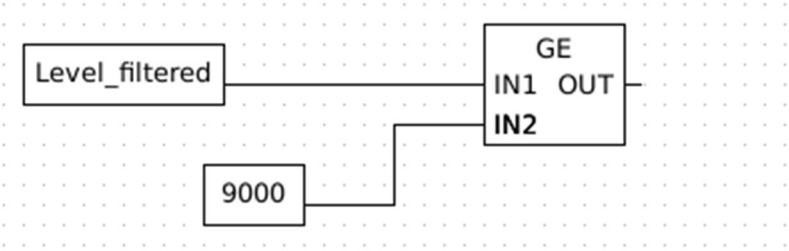


6. Right-click and select "Add>Block". In the "Block Properties" box, expand Comparison and select GE (Greater than or Equal). Select OK.
7. Right-click and select "Add>Variable". Set Expression = Level_filtered. Select OK.
8. Attach line from variable "Level_filtered" to "IN1" on the Comparison block.

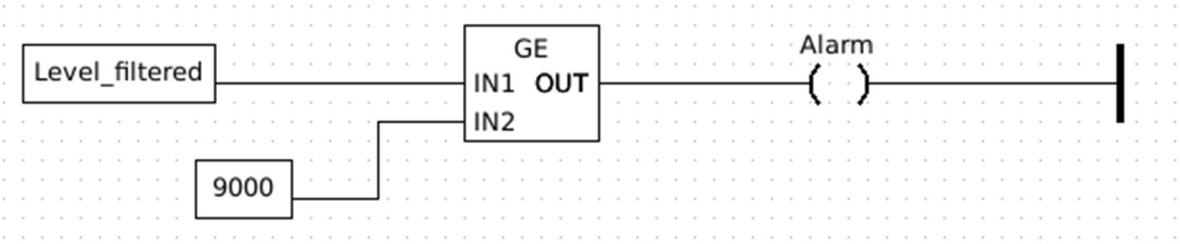


9. Right-click and select "Add>Variable", Set Expression = 9000 (Note: Equal to 90%). Select OK.

10. Attach line from variable "9000" to "IN2" on the Comparison block.



11. In the table, highlight level_filtered row and then the green "+" to add a variable in the row below. For new variable, Name=High_Alarm, Type=BOOL, Location=%QX0.5. (If High_Alarm is already in table just verify variable settings.)
12. To the right of the comparison block, Right-click and select "Add>Coil". In the "Variable=High_Alarm". Select OK.
13. Attach line from "OUT" to "High_Alarm".
14. Right-click and select "Add>Power Rail". Select Right Power Rail. Select OK.
15. Attach line from "High_Alarm" to the Power Rail. Final Ladder logic will look like the image below:



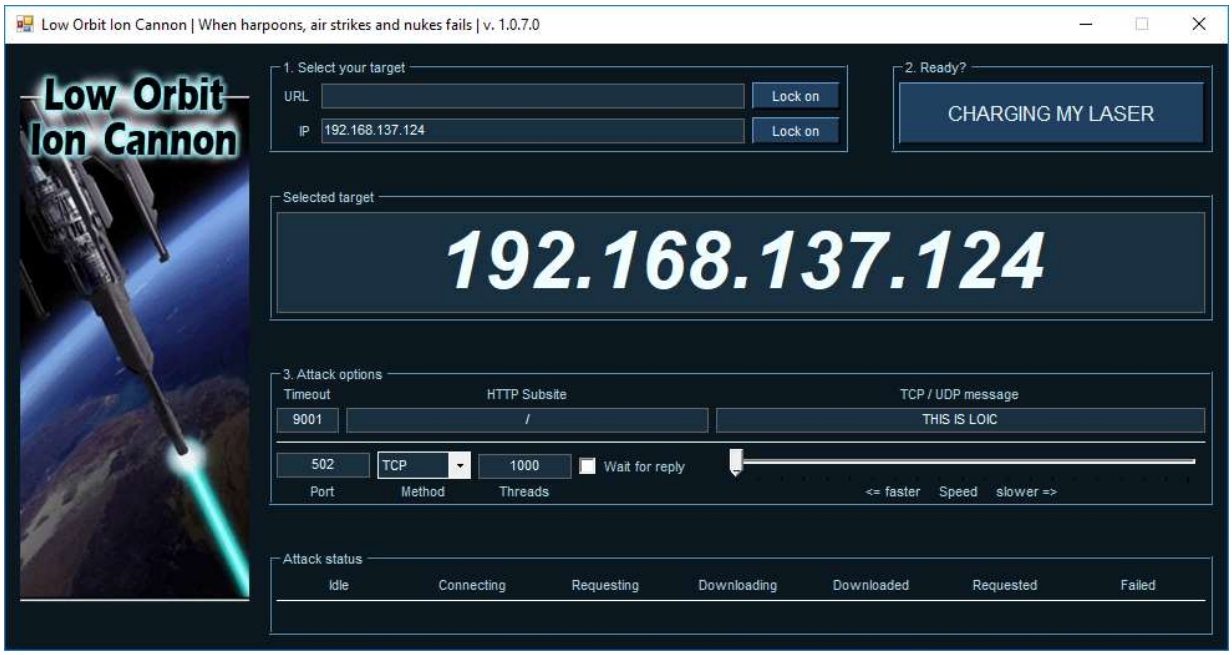
16. Save file.
17. Select File>Generate Program and save file as watertower_alarm.st in the folder ccre/scadalab/lab1
18. Launch the internet browser and go to 100.100.100.3:8080
19. Click "Choose File" and grab the watertower_alarm.st file created. Click Upload Program. You should receive a "Program compiled without errors" message. (Disregard any errors regarding conversion.)
20. Open ScadaBR again. If closed, launch the internet browser on the virtual machine and navigate to 100.100.100.2:8080/ScadaBR and login to ScadaBR (username:admin, password:admin)
21. Test the alarm to observe it coming on when the level is greater than 90. (Setpoints may need to be changed to allow level to reach 90)

EXERCISE #5 – TAKING A TARGET DOWN

This exercise will introduce the concepts of Denial of Service (DoS) and Distributed Denial of Service (DDoS). Both techniques have the purpose of taking a target down by flooding the target with a large amount of data in a short time. The Low Orbit Ion Cannon (LOIC) will be used to perform the attacks for this exercise.

All students will target the water tank PLC running on the lecturer's computer. At first, only one student at a time will be allowed to attack the simulation (DoS). Then, all students in the class will perform the attack at the same time (DDoS).

1. Download LOIC from here and extract the contents of the zip file on a folder.
2. Open LOIC.exe. On the main window, insert the target IP and click on Lock on.
3. Under "Attack options" type 502 on port, select Method "TCP", type 1000 in Threads, and uncheck the "Wait for reply" option.
4. Click on "CHARGING MY LASER" and observe the results on the lecturer computer.

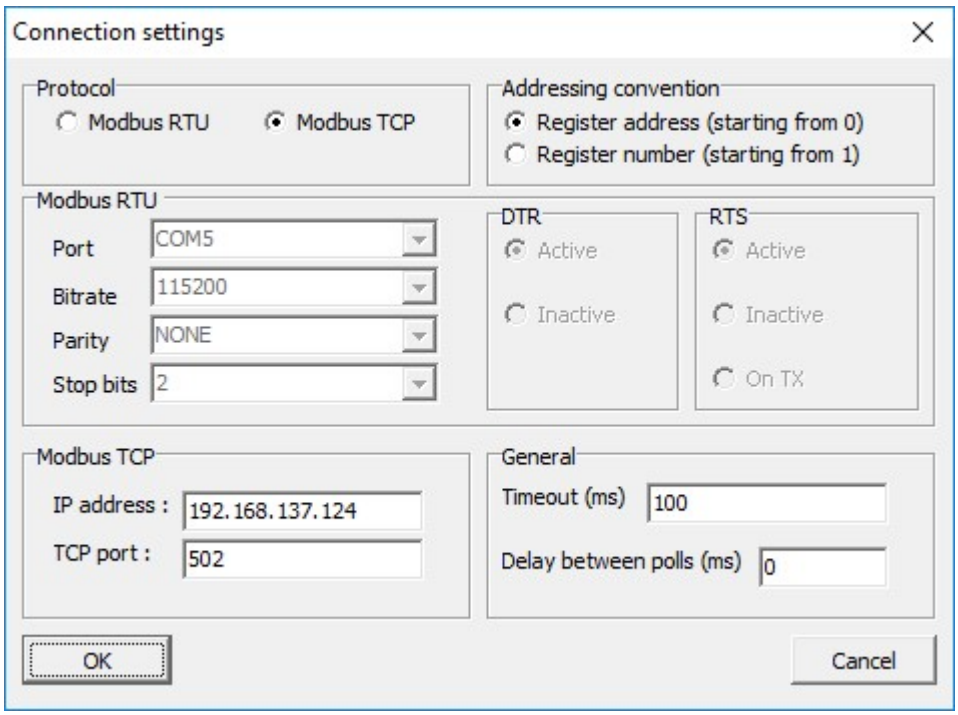


5. Verify the behavior of the system when only one student is performing the attack and when all students are attacking at once.

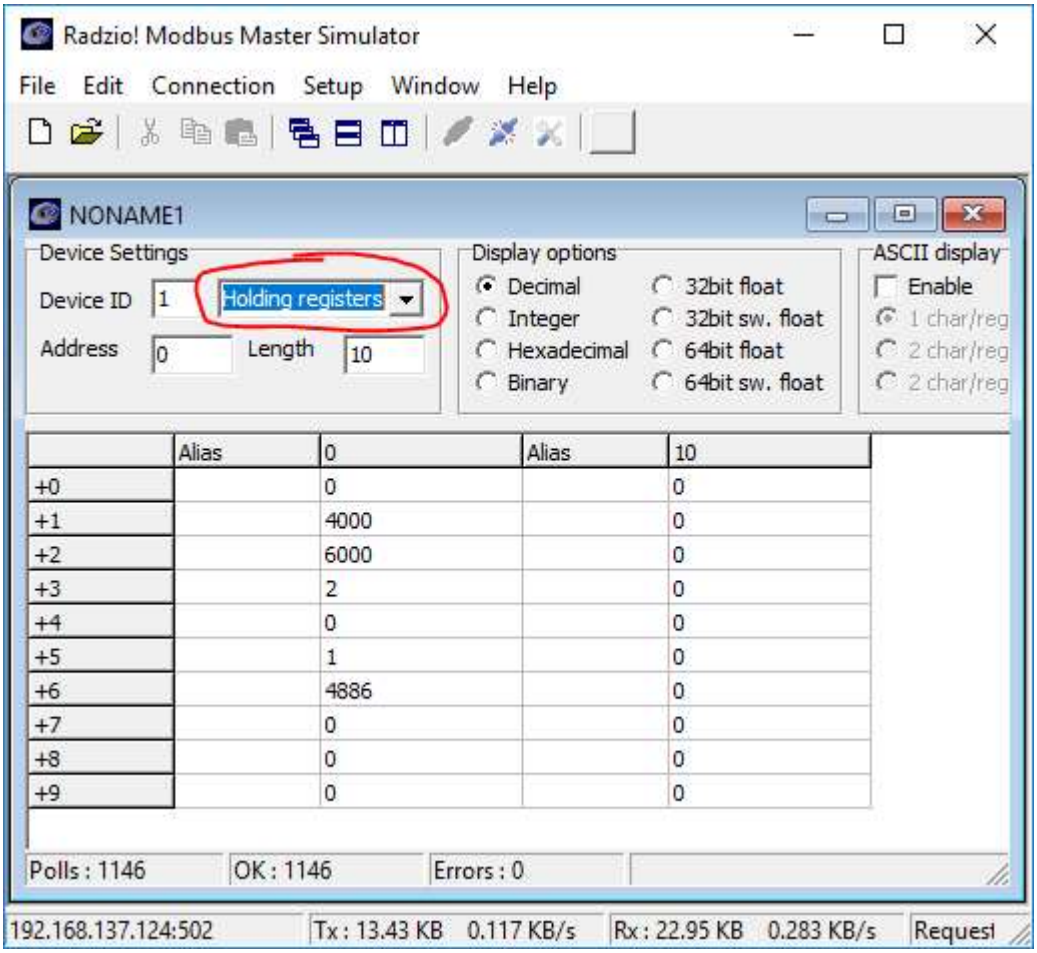
EXERCISE #6 – INJECTION ATTACK

On this exercise students will perform an injection attack by fabricating messages with different settings and sending them to the target PLC. Students will use the Radzio! software to fabricate the messages. The target will be the water tank PLC running on the lecturer's computer. At first, only one student at a time will be allowed to attack the simulation. Then, all students in the class will perform the attack at the same time.

1. Download Radzio! from here and extract the contents of the zip file on a folder.
2. Open RMMS.exe. On the main window, go to Connection->Settings. Select Modbus TCP under "Protocol", Register address starting from 0 under "Addressing convention", and type the target PLC address on "IP address: " field. Also, make sure that the TCP port is 502.



3. Click on File->New and Connection->Connect. On the new spreadsheet that appears, select Holding registers to view the PLC memory data



4. The number on the second line of the spreadsheet (+1) has the min setpoint multiplied by 100: 4000 = 40%. Similarly, the third line (+2) has the max setpoint multiplied by 100: 6000 = 60%. Change both settings by double-clicking on each line and inserting a new value.
5. Evaluate the change of behavior on the simulation after modifying the settings with the fabricated message. Try to use some of the values suggested on Exercise #3.

ACKWOWLEDGEMENTS

This lab was developed at the University of Alabama in Huntsville by Stefanie Smith, Ben McGee, Thiago Alves, Joseph Lee, and Tommy Morris.

OpenPLC is a completely open programmable logic controller with development environment, human machine interface, programmable logic controller source code, and reference hardware available at http://www.openplcproject.com/. The OpenPLC project was founded by Thiago Alves of the University of Alabama in Huntsville.

The Simulink models, human machine interface implementation, and ladder logic program for the gas pipeline and water storage tank test beds used on this laboratory exercise are copyrighted property of the University of Alabama in Huntsville.