2018

Virtual ICS test bed

Roxanne Brooks
Iowa State University

Virtual ICS test bed

by

Roxanne B Brooks

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Co-majors: Computer Engineering, Information Assurance

Program of Study Committee:
Douglas W. Jacobson, Major Professor
Thomas E. Daniels
James A. Davis

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2018

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

NOMENCLATURE

BAS	Building Automation System
CDC	Cyber Defense Competition
CIA	Confidentiality, Integrity, Availability
FBD	Function Block Diagram
FPM	Facilities Planning and Management
HVAC	Heating, Ventilation and Air Conditioning
ICS	Industrial Control System
IL	Instruction List
ISEAGE	Internet-Scale Event and Attack Generation Environment
ISU	Iowa State University
LD	Ladder Logic
MMS	Modbus Master Simulator
NSTB	Nation SCADA Test Bed
PLC	Programable Logic Controller
SCADA	Supervisory Control And Data Acquisition
SFC	Sequential Function Chart

ABSTRACT

Many organizations utilize Building Automation Systems (BAS) to ensure that all the systems in the building are functioning properly, for example their Heating, Ventilation and Air conditioning (HVAC) systems. More advanced versions of these systems, known as Supervisory Control and Data Acquisition (SCADA) systems have been architected to monitor intricate industrial processes and critical energy systems. While some security standards, such as NIST, have been developed, both BAS and SCADA systems are vulnerable to being compromised, especially if they have been networked through the Internet. Compounding the challenge, most of these Industrial Control Systems (ICS) are proprietary, which translates into organizations being unable to test their vulnerability for themselves.

The objective of this project was to construct a high-fidelity model of a complex Industrial Control System based on the system currently in place at Iowa State University. This would allow Iowa State an opportunity to test the fortitude of their industrial systems. This project, therefore, was designed to create a functioning virtual model of the systems within two buildings at Iowa State University. Additional constraints on this effort included working within the existing ISERink environment and utilization of OpenPLC, the only known open source PLC tool available at this time.

This project successfully created a fully functional virtual model, including:

- Identification and selection of tools (Modbus Master Simulator and ScadaBR)
- Configuration of tools
- Integration of tools

- Creation of a test data set
- Connections among the data
- Generation of alerts for centralized monitoring
- System testing

CHAPTER 1. INTRODUCTION

1.1 Objective

Industrial Control Systems (ICS) have existed for a very long time; unfortunately, security has not been seriously considered until very recently. After events such as Stuxnet in 2010 and the Target HVAC breach in 2013 drew media attention, more people began to realize the significance of ICS security and how badly it was lacking. An ICS test bed would be extremely useful to help develop and test ICS security, however most ICS tools today are proprietary, so researchers cannot obtain the access needed to perform experiments. With the help and guidance of Iowa State, the objective of this project was to construct a high-fidelity model of a complex Industrial Control System, such as Iowa State's ICS, within the ISERink environment, which could be used in future instances for security vulnerability testing as well as for testing future integration capabilities.

1.2 ICS at ISU

The campus of Iowa State University is composed of many buildings, each of which contains numerous systems that provide both safety and comfort. If there were to be an issue with, for example, the HVAC system in a building, it could potentially disrupt classes and other scheduled activities. Protective measures are in place so that the systems within every building are monitored and, should something malfunction, the Facilities Planning and Management (FPM) team would immediately be notified. The protective measures in question are currently produced and monitored by a third-party vendor who works closely with Iowa State in order to identify and swiftly rectify any issues as they arise. Iowa State's third-party tool consists of three main parts: the actual systems being monitored, the sensors that record the systems' states, and the central monitoring device that the sensors report to.

This type of system is commonly referred to as an Industrial Control System (ICS). This can be seen in Figure 1 below.

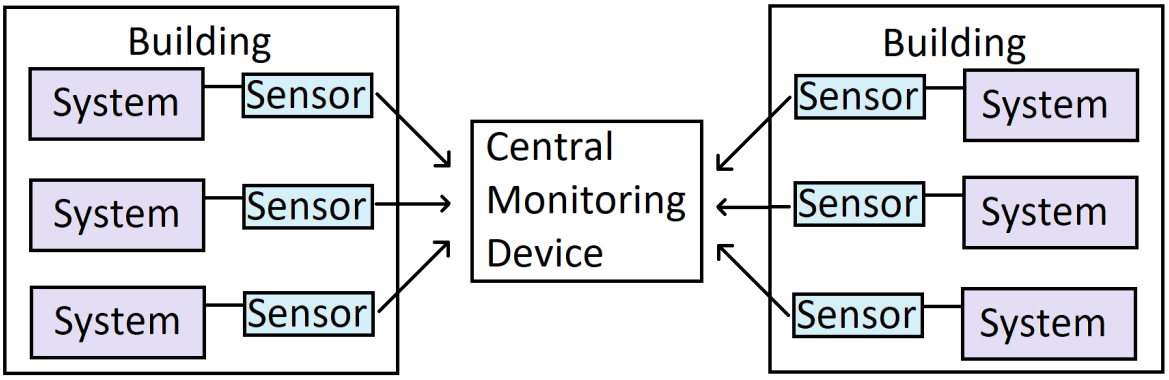


Figure 1 *ICS illustration*

1.2.1 Systems

The systems being monitored include thermostats, HVAC systems, fire alarm systems, steam valves, as well as other devices that exist within buildings in order to achieve operability. Essentially any industrial machine that is capable of being monitored is monitored by an ICS. While the current third-party contractor also has the ability to monitor keycard access to campus buildings, Iowa State has instead decided to use a different system and kept that functionality segmented.

1.2.2 Sensors

The sensors are the endpoints that connect to everything from steam valves to fire alarms and intruder warnings in places such as the infamous steam tunnels. The sensors are connected to these devices and monitor their output, which could be as simple as the ON/OFF state on a light switch, or more complex such as the current temperature on a thermostat. The sensors regularly update and report this information back to the central monitoring device based on the frequency parameters that have been configured.

1.2.3 Central Monitoring Device

This device compares the values that it receives from the sensors to what it expects to receive. Typically, the values are within the expected range and nothing will happen, but when they are not a flag is thrown which alerts whoever is managing the device that there is an unexpected result. The device also shows which sensor in which building threw the flag and what its value is. This allows the manager to quickly alert the group in charge of Iowa State's systems, in this case Facilities Planning and Management (FPM), who will then physically go and inspect the system in question. The people in charge of the central monitoring device are also in charge of determining when a flag should be thrown and creating an event such that a flag will be thrown when those parameters are met.

CHAPTER 2. RELATED WORKS

ICS: SCADA vs BAS

Within the domain of ICS, there is a distinction between Supervisory Control And Data Acquisition (SCADA) and a Business Automation System (BAS). For this project specifically, the system developed is modeled after a BAS, not a SCADA system. When researching these systems, it became apparent that people often incorrectly use SCADA interchangeably with BAS instead of ICS.

SCADA systems are extremely precise and therefore very expensive. SCADA systems are most frequently found in the manufacturing industry and throughout the energy sector. BAS, on the other hand, are much less expensive and less precise, because precision is not as crucial. A BAS is put within a building to monitor systems such as HVAC and fire alarms and ensure regular functionality.

The most important difference to understand between SCADA and BAS is that SCADA is used for critical systems and BAS is not. Therefore, while this test bed was modeled after a BAS, its configurability allows the potential to be used as a SCADA system as well. For this project, therefore the larger, more encompassing term ICS is used.

3.2 Alternate Test Beds

Before considering the construction of a new test bed, it is worth examining what test beds already exist and whether they would meet the needs that this project hoped to satisfy.

3.2.1 DETERLab

DETERLab is based out of University of Southern California's Information Sciences Institute at the University of California, Berkley. They are funded by the Department of Homeland Security, the National Science Foundation, and the Department of Defense. More

than 20 organizations have partnered with the DETERLab in order to do cyber security research. The actual code behind DETERLab is based around Emulab's Network Emulation Test Bed. It is significant that anyone can apply for access to this test bed with a faculty sponsor since few test environments are open to the public.

DETERLab, like ISERink, has built-in features such as network traffic generation. Unlike ISERink, however, the internet access is only through the SSH connection used to manage the experiment. In addition, while DETERLab can only be accessed via a remote connection, an instance of ISERink can be spun up and accessed anywhere. For ease of access and troubleshooting therefore, the ISEAGE test bed was utilized for this project.

3.2.2 National SCADA Test Bed (NSTB)

The NSTB is being utilized by National laboratory groups such as Argonne, Idaho, Lawrence Berkeley, Los Alamos, Oak Ridge, Pacific Northwest, and Sandia National Laboratories. These groups are each utilizing the NSTB for different research projects, but essentially the test bed aims to provide an environment that allows work toward critical security vulnerabilities and threats that the energy sector faces.

At first glance this seemed to be the test bed that I was looking to create, but unfortunately the NSTB appears to be inaccessible to individuals not directly associated with the National Labs. This follows the theme of ICS tools and the lack of open source equivalents. For all of these reasons – most notably the dearth of accessible data – the ISERink test bed was selected for this project.

CHAPTER 3. DESIGN AND IMPLEMENTATION

In order to more fully understand the system created from this project, it would be beneficial to understand how the individual tools used in this project function. The system begins with a tool used to establish a secure yet realistic environment known as the ISERink test bed. The following tools all run simultaneously within the ISERink environment:

- OpenPLC – simulates the current state of the systems being monitored using files written with Ladder Logic. The creator also developed PLCOpen Editor for creating and editing Ladder Logic programs.
- Modbus Master Simulator – allows the user to view and manipulate registers in OpenPLC, instead of requiring direct interface with physical devices.
- ScadaBR – a tool with a sophisticated user interface that connects OpenPLC programs and monitors the states of its registers for unusual events based on identified thresholds. If ScadaBR detects any unusual events, it will trigger alerts and notify the person or group in charge of monitoring that system, including email notifications if desired.

3.1 ISERink Test Bed Environment

ISEAGE stands for Internet-Scale Event and Attack Generation Environment. The ISEAGE Lab is an Information Systems Security Laboratory within Iowa State which has created the ISERink test bed, along with other related tools. The ISERink test bed is a virtual model of the Internet which allows researchers to design and test cyber security scenarios within a controlled environment. This even includes creating background noise on the network similar to that which exists within the real Internet. Perhaps the most unique aspect of the ISERink is that it is portable and self-sufficient so that an instance can be deployed on

any hypervisor. Because of this, ISERinks have been distributed to many educational institutions throughout the state of Iowa and beyond. The ISEAGE Lab is most well-known for the many Cyber Defense Competitions that are put on annually for college and high school students, hosted within the ISERink environment. The ISEAGE Lab provides an excellent opportunity for students and researchers alike to learn more about cyber security in a one-of-a-kind controlled environment.

3.2 OpenPLC

PLC stands for Programmable Logic Controller. OpenPLC is a tool created by a doctoral student at the University of Alabama, Huntsville, named Thiago Alves, who observed that PLCs were growing in popularity and wanted to test their security capacity. After concluding that he could not access the source code to any PLCs in order to test them, he decided to simply develop his own open source alternative. In order to make this tool as close to traditional PLCs as possible, Alves adhered to the IEC 61131-3 standard, which defines the programming languages and architecture for PLCs. The OpenPLC tool itself emulates the functions of electromechanical relays based on the configurations of the programs uploaded. The tool is very versatile and can connect to RaspberryPis, Arduinos, ESP8266s and many other devices since it is open source which allows users to create their own drivers to interface with. For those without a hardware budget, it even allows users to create a completely virtual system by connecting to a Modbus I/O device. Before connecting an OpenPLC instance to a controller, it must be programmed using one of the five programming languages defined by the IEC 61131-3 Standard.

3.3 PLCOpen Editor

PLCOpen Editor supports all five of the languages defined in the IEC 61131-3 standard. These languages include: Ladder Logic (LD), Function Block Diagram (FBD),

Instruction List (IL), Structured Text (ST), and Sequential Function Chart (SFC). Ladder Logic was the language of choice for this project. The PLCOpen Editor also contains a "Generate Program" command that will translate any of the PLC languages into a Structured Text (ST) file format required by the OpenPLC server.

3.3.1 Ladder Logic

Ladder Logic (LD) is a programming language that is frequently used for PLCs. It consists of a graphical diagram which looks like circuit diagrams of the devices being controlled, as seen in Figure 2. It also resembles a ladder with rungs, which is where the name originates.

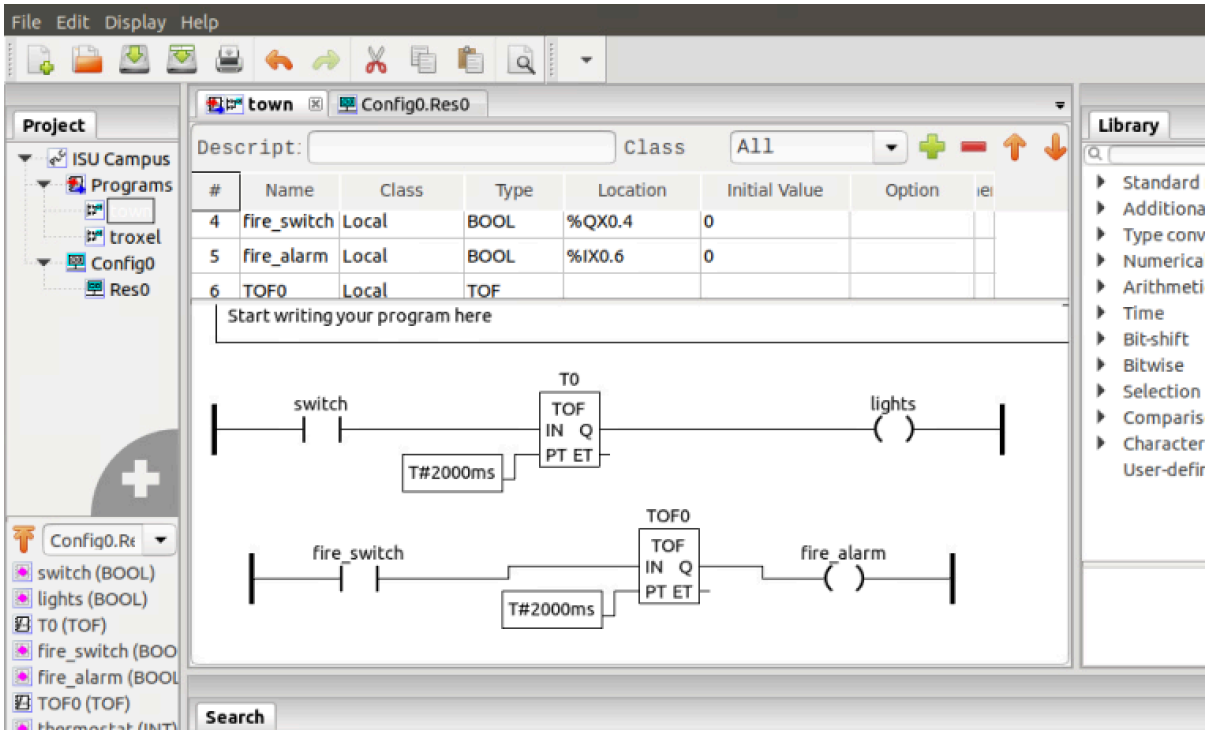


Figure 2 *PLCOpen Editor with LD file types*

Using pin mapping and digital logic, it is easy to create a function that controls the relationship between multiple points, for example, a switch that controls turning a light on and off. The biggest obstacle in pin mapping is understanding the correlation between

location and register type. The selection of register types will depend upon 1) the type of information the register must contain, and 2) whether the state will need to be manipulated. With a Modbus you can choose between four types of registers: coil, discrete input, input, and holding registers (Figure 3).

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 3 *Register types*

Coil and discrete input registers can only hold one bit of data, so they can only be in the ON or OFF state. This would include the example of a light switch which only has two states. Input and holding registers are 16 bits and can hold more complex data such as integers. These registers would be more appropriate to hold data like the temperature on a thermostat.

States can be manipulated when using coil and holding registers, as they have read and write access. In contrast, input and discrete input registers only have read access; these registers can only change state if a connected device causes it to change. For example, in my program there is a light connected to a light switch. The light cannot be changed unless the state of the light switch changes. The light data is stored in a discrete input register and the switch in a coil register.

Once the data points have been added and configured, the converted ST file can be uploaded to the OpenPLC server. Each device configuration can be tested with a Modbus

simulator, or with a physical device depending on how the OpenPLC server has been configured.

3.4 Modbus Master Simulator (MMS)

The MMS is an example of a Modbus I/O device. It is a free virtual alternative to an Arduino, RaspberryPi, or other physical device that allows the user to interact with the data points created in the LD program. The MMS is the simplest, but arguably most essential aspect of this system. Once downloaded, the IP and Port address of the OpenPLC server are entered in the Connect dialog box (Figure 4) to link the MMS to OpenPLC.

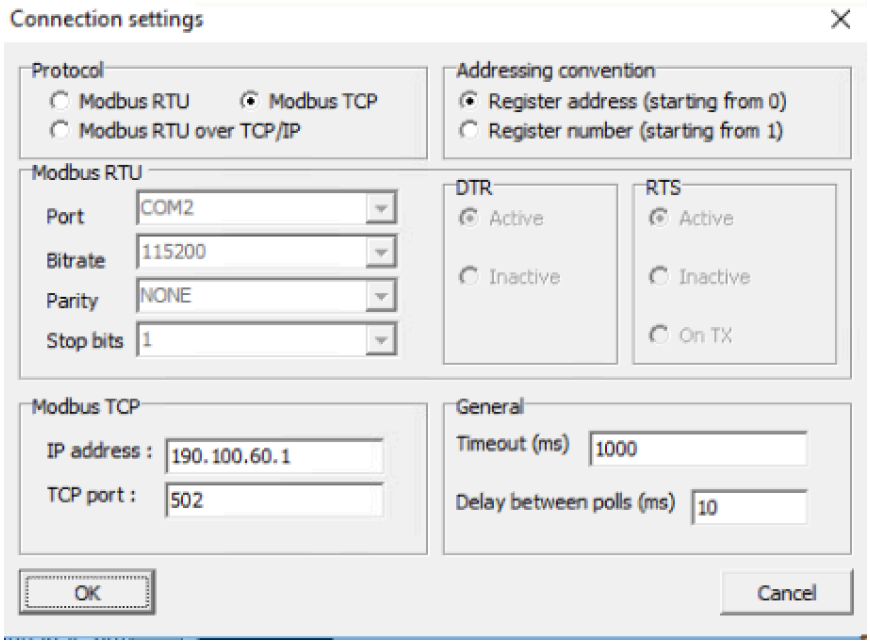


Figure 4 *MMS Connect dialog box*

The Polls, OK, and Errors metrics at the bottom of the application reflect whether a true connection has been made (Figure 5). As the Polls number increases, so should the OKs.

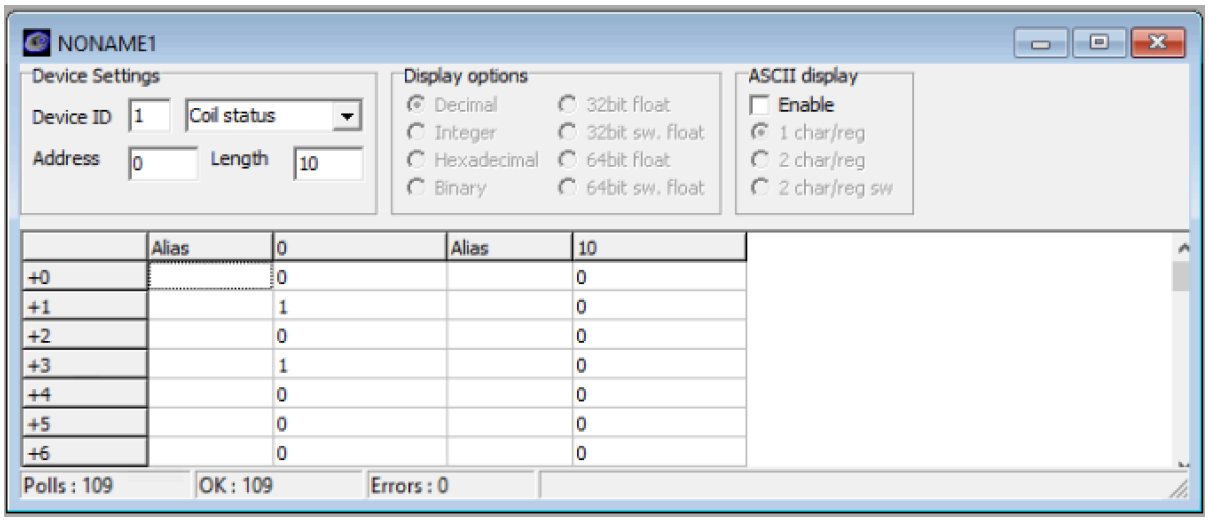


Figure 5 *MMS display*

This simulator displays the current state of all of the registers in the OpenPLC instance. The simulator can also manipulate the data in the coil and holding registers, which should in turn affect the states of the input and discrete input registers based on the LD program (Figure 6). Without this tool, it is not possible to see what is going on in OpenPLC.

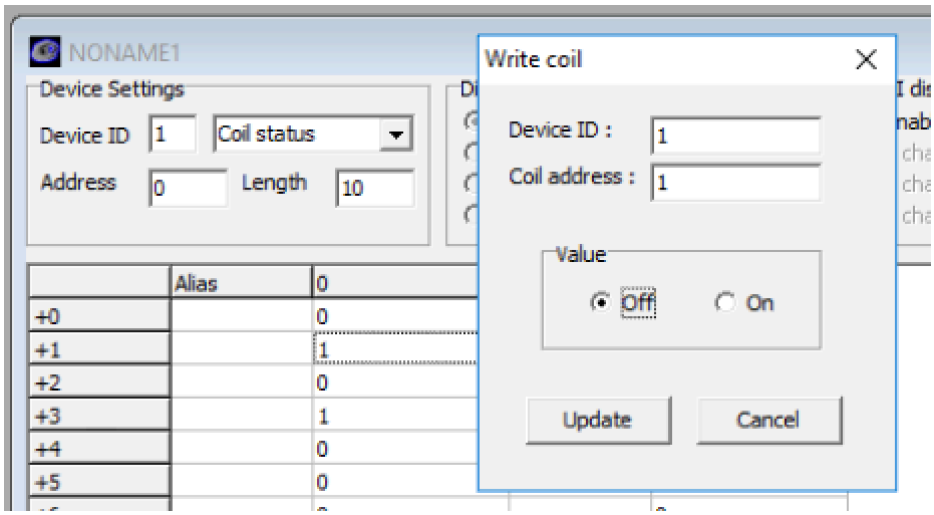


Figure 6 *MMS toggle register states*

3.5 ScadaBR

After testing the OpenPLC data via data manipulation to ensure it works as expected, ScadaBR can be configured to monitor the data. ScadaBR is an open source Supervisory

Control and Data Acquisition (SCADA) system which was created in Brazil. This type of system is designed to monitor and potentially control all of the devices that are connected to it. This is essential to an automation system as it is a centralized point that can be configured to warn of any issues with connected devices in lieu of manually checking all of them individually. This particular tool also has a very well-written, intuitive user interface.

3.5.1 Data Sources and Points

Configuring ScadaBR to identify a project's data points is slightly more complex, but significantly more useful, than the MMS. The first step is to add a new data source to ScadaBR, as shown below in Figure 7.

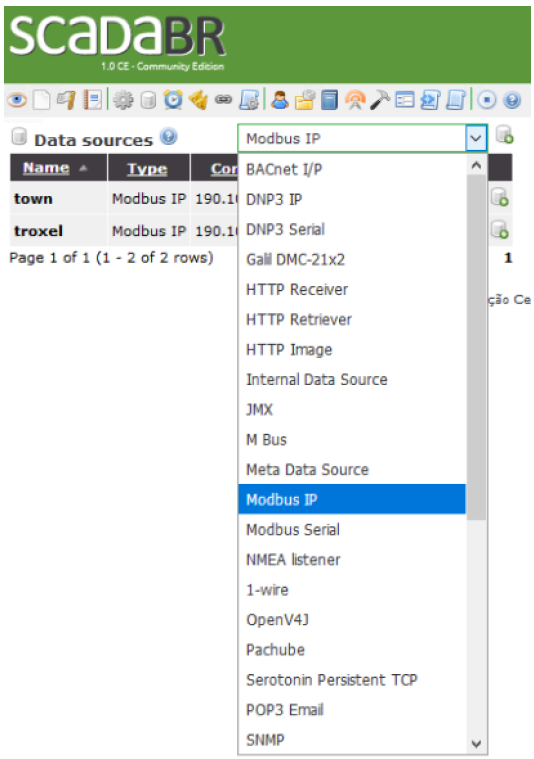


Figure 7 *ScadaBR data source types*

ScadaBR's list of data source types is very dynamic. Perhaps the most attractive aspect of ScadaBR is its ability to connect multiple data sources as well as multiple types of data sources. Because the tool used to interface with OpenPLC in this project is the MMS,

the data source type in this instance is “Modbus IP”. In this project, two Iowa State buildings (Troxel and Town Hall) were added.

Once the source type is selected, a new window will appear for configuring the details of the data points (Figure 8). This window offers an extensive list of parameters that help ensure the correct data points are configured.

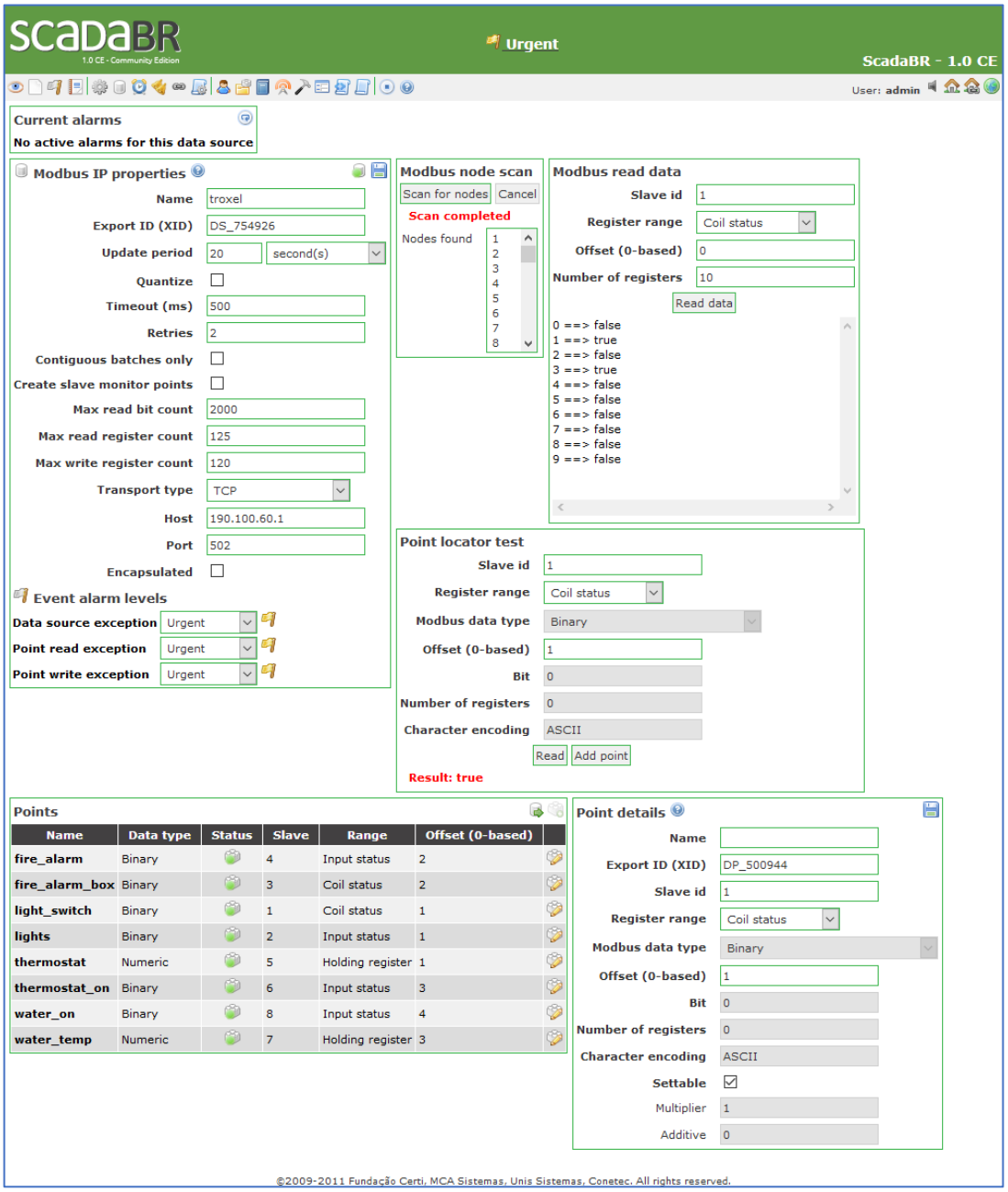


Figure 8 *ScadaBR data point configuration*

The first box in the top left corner (Figure 9) requests information such as the IP address and Port number, similar to the MMS. However, since ScadaBR is a monitoring device, it offers more control over the connection properties such as customization of the update period and timeout.

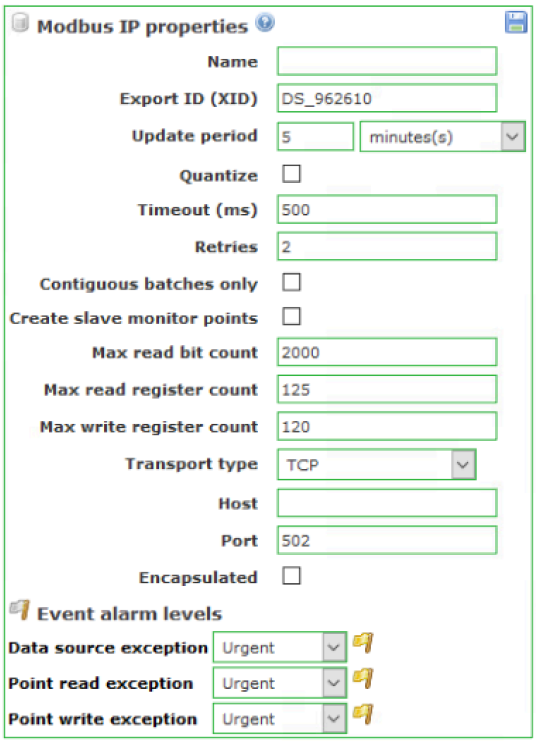


Figure 9 *ScadaBR data source properties*

After this data source has been saved, the next task is to add data points. ScadaBR can perform a Modbus node scan and then populates a list of data values based on the register range selected, as seen below in the Modbus node scan and Modbus read data boxes (Figure 10). To take that one step further and ensure that the right data point is added to the data source, the "Point locator test" box directly below the other two on the same page verifies the current value of the selected data point before the "Add point" button generates another box for a quick data point addition. The image below shows the scans and tests that display the current values of each register, to help ensure that the correct data point is added.

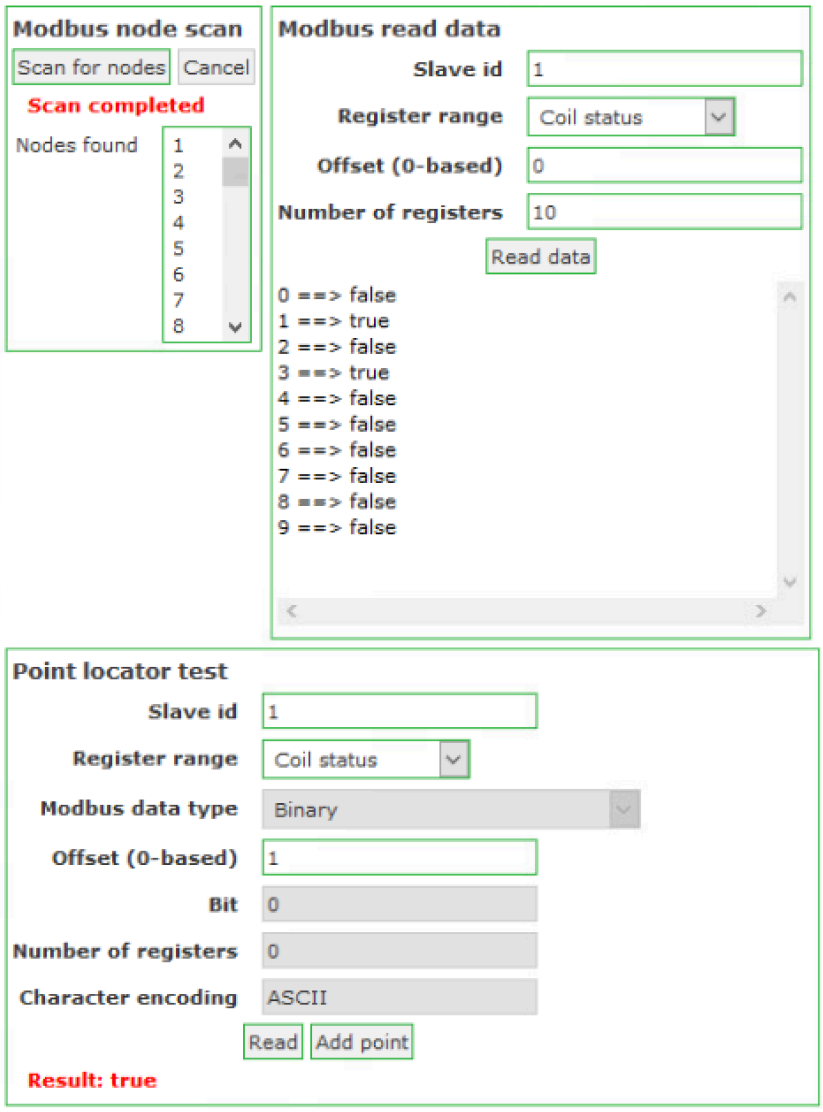


Figure 10 *ScadaBR data point view current state*

If that process is followed, the Point details box, which is used to actually add points to the data source, will have the slave id, register range, and offset preconfigured correctly. Once this procedure has been followed multiple times it is possible to bypass the preceding steps and simply configure those values manually. The final step is to click the floppy disk

button in the top right corner of the Point details box and then the data point will appear in the Points box beside it (Figure 11).

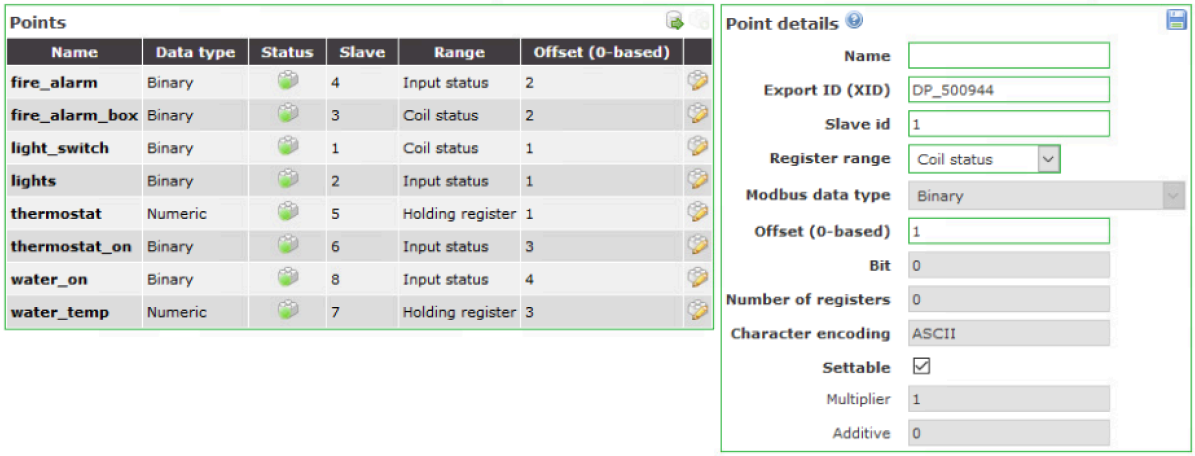


Figure 11 *ScadaBR configured data point details*

Within each data source, an extensive number of points can also be added for monitoring. Each point within the data source is labelled and assigned an ID number. If there is even an issue with a specific data point or it becomes unresponsive, ScadaBR will issue an alert with the name of the data source and the slave id of the data point. Within each building in this project, data points were configured which represented lights, fire detection, and temperatures.

3.5.2 Alarms

When all data points have been configured, the alarms can be set on the devices requiring alerts. There are simple alarms such as when a device changes state, or when it changes to a specific state, which is extremely useful for devices like fire alarms. For more complex data points, more technical alarms can be set, such as if the temperature on a thermostat reaches or exceeds a certain value. Additionally, multiple alarms can be set for one data point. The example below in Figure 12 illustrates different alarms for temperature thresholds on a thermostat.

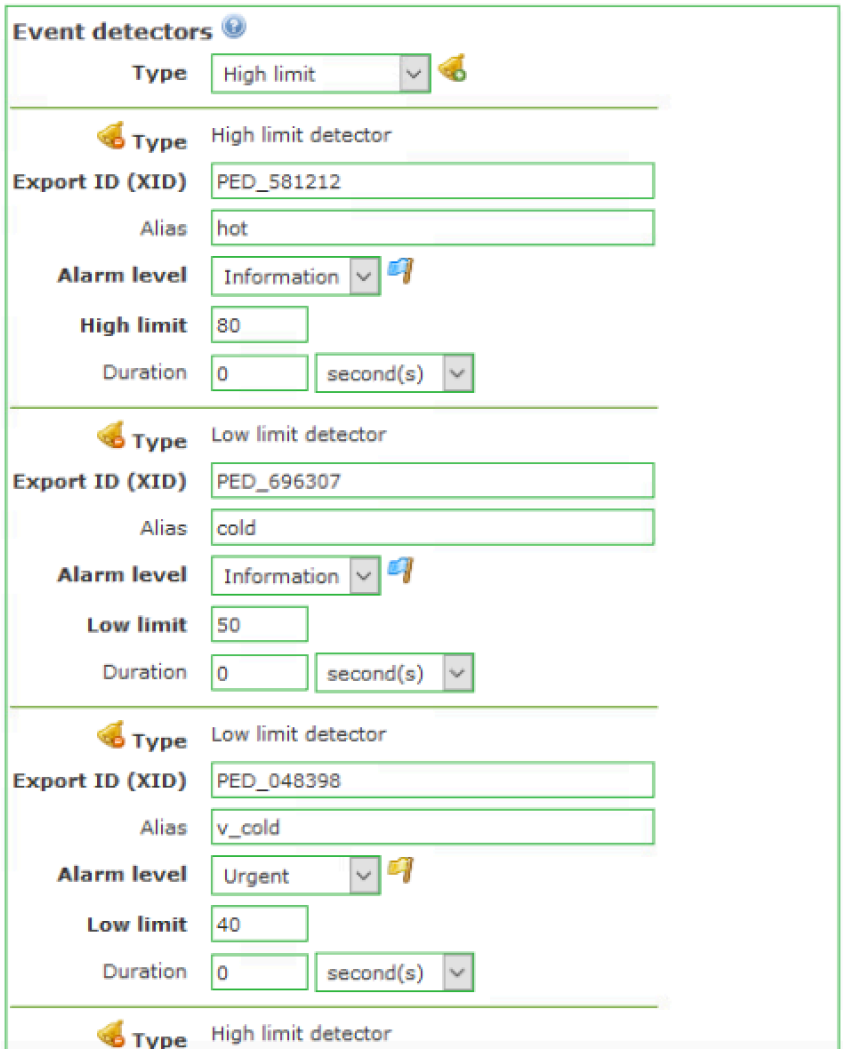


Figure 12 *ScadaBR temperature alarms*

Scada BR can also handle more complex, compound events, where multiple flags have to be thrown in order to set off the alert. For example, schedules can be created in ScadaBR to reflect business hours when a building should be open, and the lights are expected to be on. The state of the lights is then compared to these parameters. In this project, ScadaBR has a compound event that says IF the lights are turned ON AND the schedule says it is EITHER (NOT work hours OR IS the weekend), THEN an alarm should go off. In this situation if it is the weekend (or after business hours), but the lights are not on,

there will not be an alarm, or if the lights are on but it is during business hours and not the weekend there will not be an alarm.

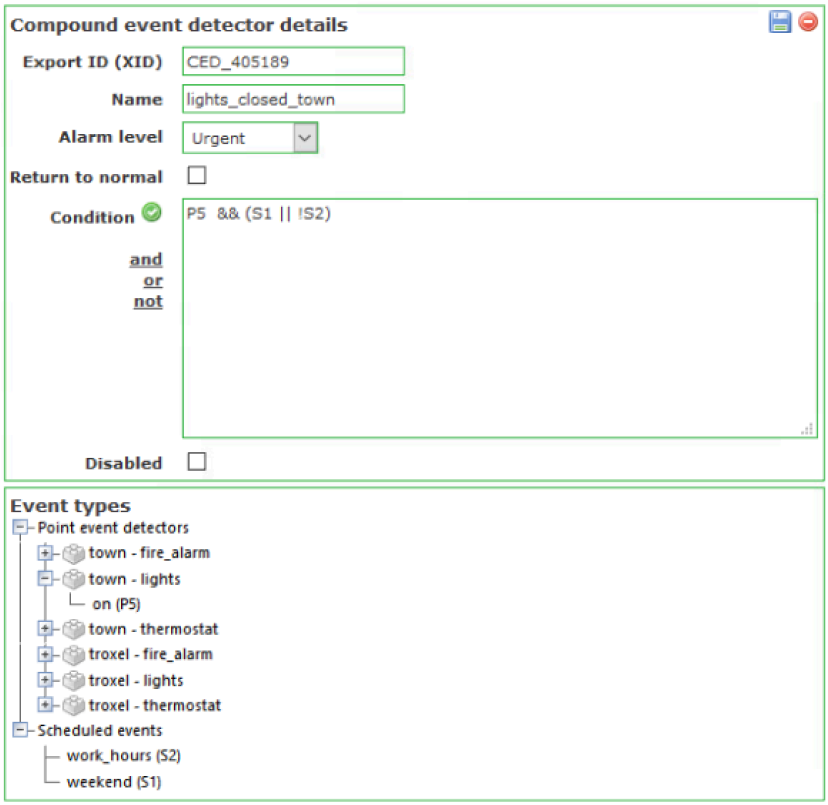


Figure 13 *ScadaBR compound event details*

So as seen in Figure 13 above, the condition is P5 && (S1 || !S2), where P5 represents the lights data point (in the town data source) being in the ON state, and one of the two following states being true: the scheduled event S1, weekend is TRUE, or the scheduled event work_hours (S2) is NOT TRUE (also known as FALSE).

While a compound event requires multiple individual events to be created, each individual event does not need to generate an alert. So, in this example, it would be preferable to simply choose None when setting the Alarm Level rather than getting an alert every time the lights were turned on, or every time it is the weekend.

When setting different alarms, an important element to consider is the notification that will be displayed when the alarm is triggered. There are five types of alarms from which to choose: None, Information, Urgent, Critical, and Life Safety. Each of these signify different levels of importance and are represented using different colored flags, some of which can be seen in Figure 14.

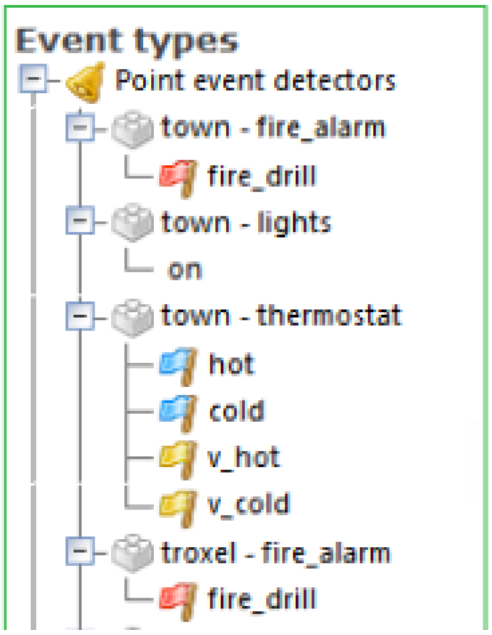


Figure 14 *ScadaBR multiple event types*

When multiple alarms are going off, the notifications will prioritize the highest ranked alarm that has not yet been acknowleged. In Figure 15, seen below, the top of the page is flashing a Life Safety flag, even though the newest alarm in this case is an Information alarm. In addition, this image illustrates that if an alarm has not been acknowleged but has stopped, a line will strike through the flag in the Alarm level column, and the Inactive time column will report at what time the alarm was cancelled.

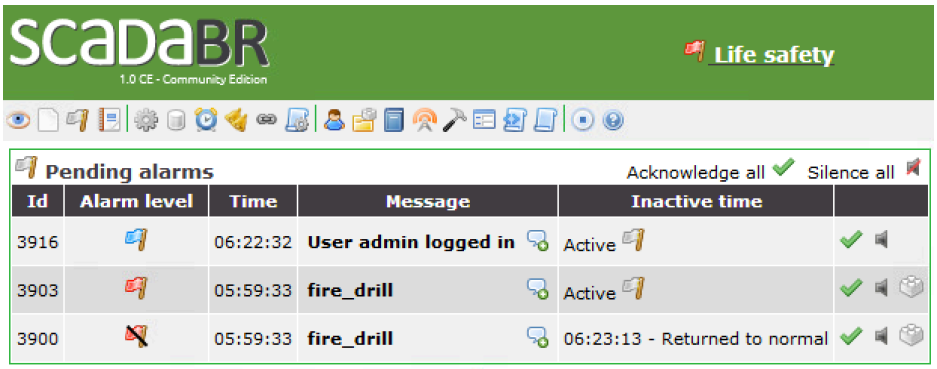


Figure 15 *ScadaBR pending alarms*

When dealing with pending alarms, the far-right column contains clickable icons. In order to dismiss the alarm, the green checkmark serves as an acknowledge button, which will clear that alarm from the Pending Alarms table. If there are many alarms in the table, clicking the acknowledge all button at the top of the tables is an efficient way to clear all of them at once. The sound icon, when clicked turns into a sound icon with a red line stricken through it. This serves to temporarily mute, but not acknowledge the desired alarm. Another click will un-mute the alarm. If both of the fire drill alarms shown above were either muted or acknowledged, the next highest flag would replace the alarm flashing at the top of the page – in this case Information would flash at the top.

3.5.3 Users and Privileges

As with a real ICS, multiple users with different jobs might need access to ScadaBR. Any administrator can add or modify a user's settings so that they are only allowed access to the data points that they need to perform their duties. This is an application of the security principle of least privilege. In this instance the emt account (shown in Figure 16) will get notifications when a fire alarm is triggered in any building, but will not be able to see any of the other data points.

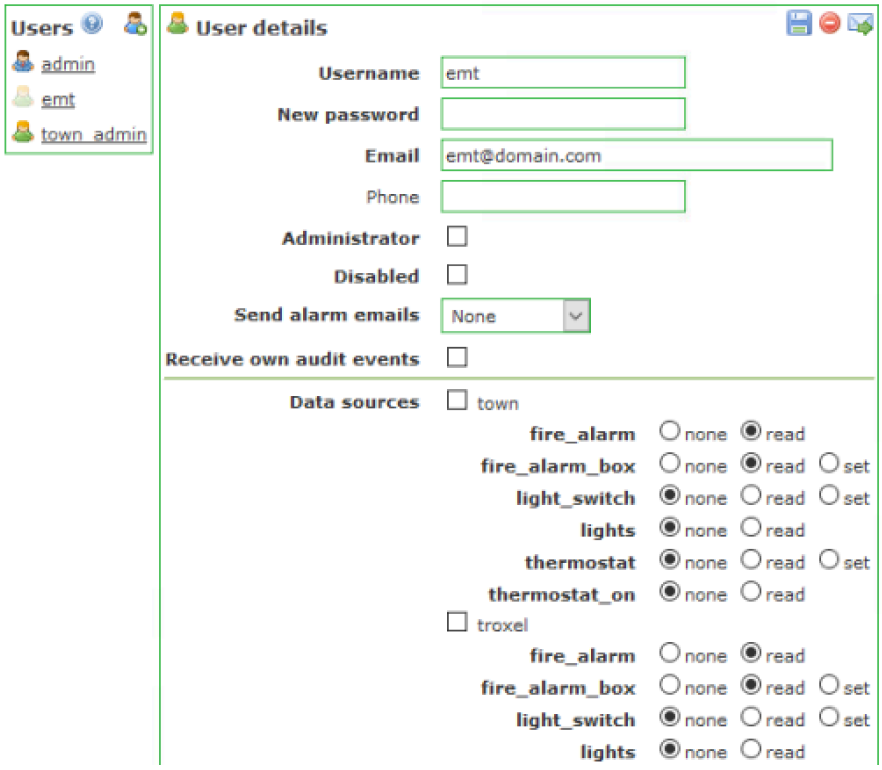


Figure 16 *ScadaBR user details*

As shown below in Figure 17, the emt account has significantly less options within ScadaBR than the administrator account that has been shown in the rest of this document. Comparing the toolbar below to the one seen in Figure 15 indicates that the privileges of this user are much more restricted.

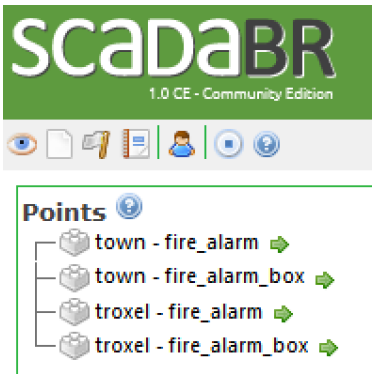


Figure 17 *ScadaBR user view*

In conclusion, ScadaBR can be a very powerful tool for automating device monitoring. Once ScadaBR has been configured, it is very easy to add or modify data points as well as alarms, or even add a new type of data source, and replace the MMS with a physical device. There are also security settings built in to restrict access for regular users.

CHAPTER 4. RESULTS

As aforementioned, the Iowa State campus consists of three main parts that work together to create the building management system:

1. The actual systems (e.g. HVAC, fire detection, etc.)
2. The sensors attached to those systems, and
3. The central monitoring device

The virtual model consists of three mirroring components.

1. The systems are simulated by the Master Modbus Simulator
2. The monitoring sensors are represented by the OpenPLC instance, as it tracks the states of the systems as well as their effect on other parts of related systems, and
3. ScadaBR represents the central monitoring device, which has surprisingly similar capabilities to Iowa State's central monitoring system, but is open source instead of proprietary

4.1 Completed System

The figure below (Figure 18) is a diagram of the test bed configuration and illustrates where each tool is hosted, and which tools interact. There are two servers running within the ISERink environment, as depicted with different colored boxes encompassing the tools running on each server. The two OpenPLC images represent the two files being run on the same server. While it would be comparable to have two different servers running OpenPLC, each with one of the LD files, because of space confinements it made more sense to run both files on one server and connect to them separately.

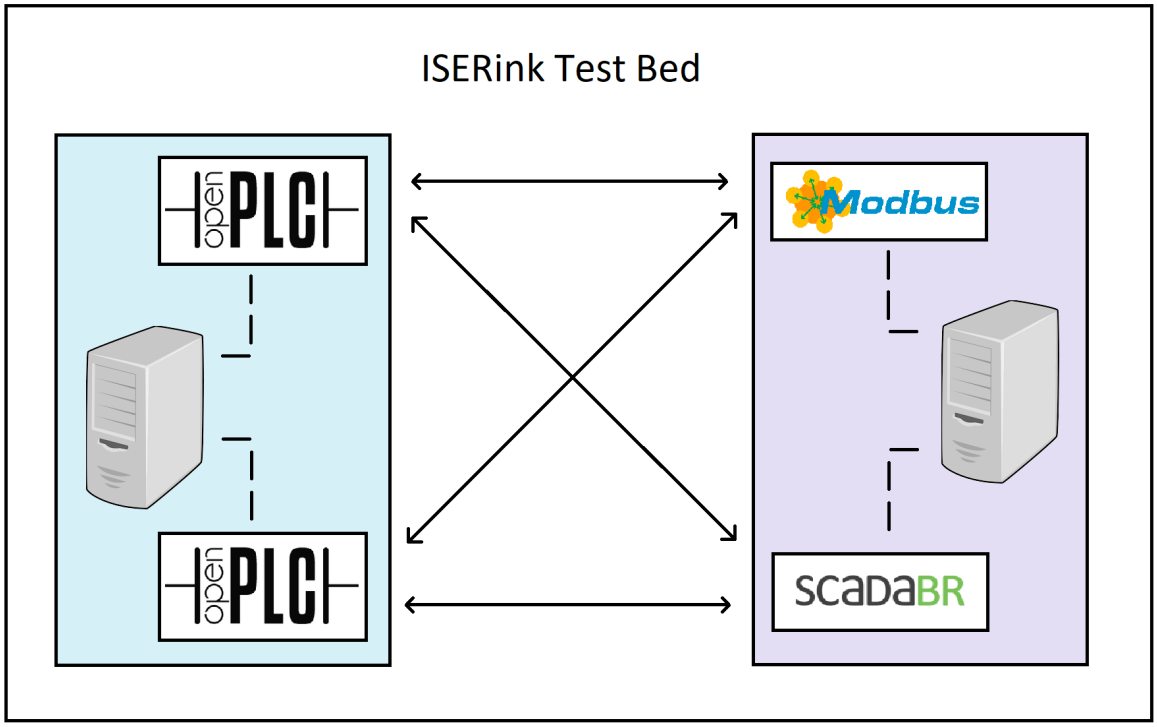
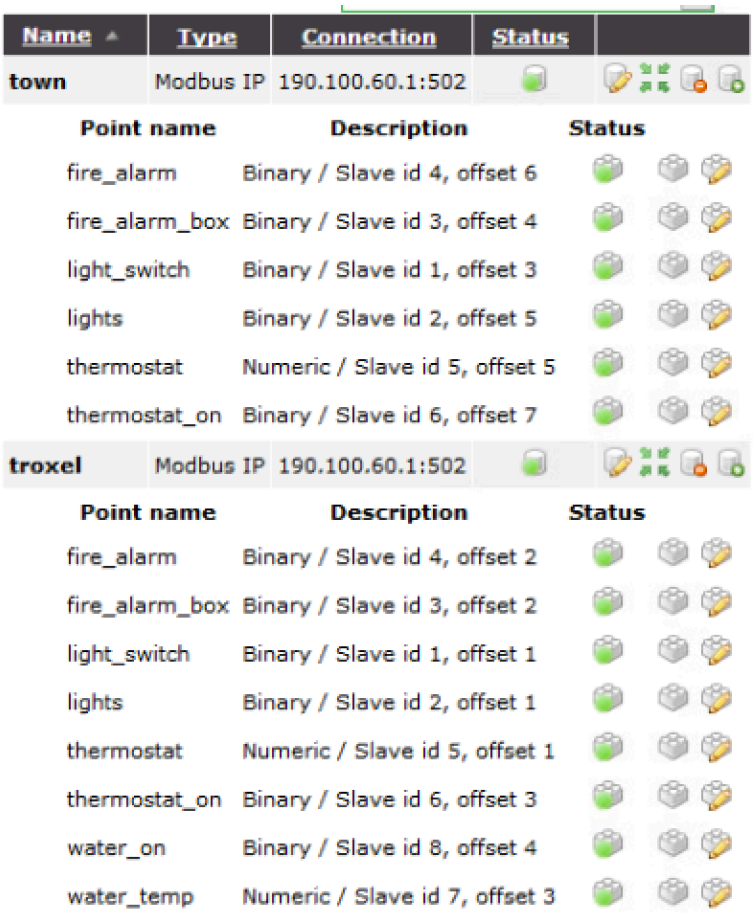


Figure 18 *System diagram*

4.2 Data Points

The data points created for this test bed sampled different types of data that are monitored at Iowa State, though they are not a complete set due to both confidentiality reasons and redundancy. Buildings at Iowa State could have hundreds of points being monitored. The data points as they exist within their respective data sets in OpenPLC can be found in the Appendix, which is the ST file being run on the OpenPLC server. The data points as they exist within ScadaBR can be seen in Figure 19 below.



Name	Type	Connection	Status	
town	Modbus IP	190.100.60.1:502		

Point name	Description	Status
fire_alarm	Binary / Slave id 4, offset 6	
fire_alarm_box	Binary / Slave id 3, offset 4	
light_switch	Binary / Slave id 1, offset 3	
lights	Binary / Slave id 2, offset 5	
thermostat	Numeric / Slave id 5, offset 5	
thermostat_on	Binary / Slave id 6, offset 7	

troxel	Modbus IP	190.100.60.1:502		

Point name	Description	Status
fire_alarm	Binary / Slave id 4, offset 2	
fire_alarm_box	Binary / Slave id 3, offset 2	
light_switch	Binary / Slave id 1, offset 1	
lights	Binary / Slave id 2, offset 1	
thermostat	Numeric / Slave id 5, offset 1	
thermostat_on	Binary / Slave id 6, offset 3	
water_on	Binary / Slave id 8, offset 4	
water_temp	Numeric / Slave id 7, offset 3	

Figure 19 *ScadaBR data points*

4.4 Challenges

While this is a very user-friendly system, numerous challenges were overcome in the process of creating it. The following are some important things to consider if this system is recreated in the future.

4.4.1 ISERink

While ISERink is a very useful and versatile test bed, it does not appear compatible with Macintosh computers. A solid chunk of time was invested in simply creating vanilla virtual machines due to the amount of errors encountered while working on a Macintosh. Once the decision to switch to a Windows computer was made, none of the previous errors were encountered ever again.

4.4.2 OpenPLC

When following the instructions to start the OpenPLC server, users should simply open a terminal, navigate to the OpenPLC folder and type ./start_openplc.sh. There is, however, an error binding to the socket (Figure 20) unless the following command is run instead sudo ./start_openplc.sh. The only difference is that OpenPLC is running with administrative privileges in the second case (Figure 21). This is a small detail, but a necessity for communication between the tools.

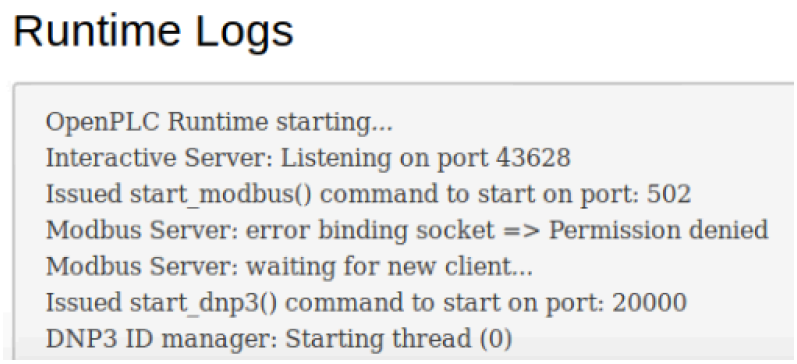


Figure 20 *OpenPLC no sudo*

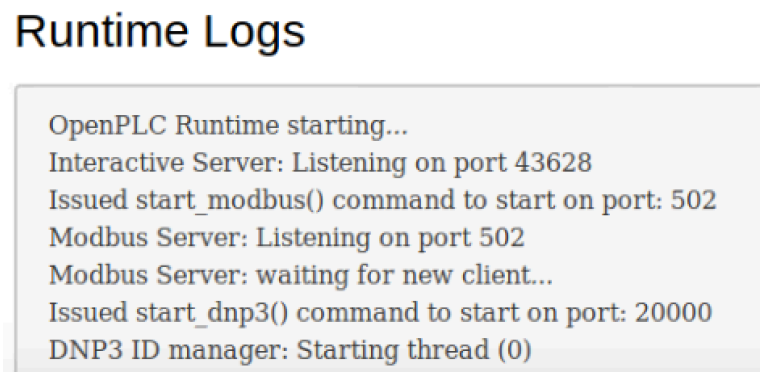


Figure 21 *OpenPLC with sudo*

4.4.3 PLCOpen Editor

When a file has been created, the final step before it can be uploaded to the OpenPLC server is to execute the Generate Program command (Figure 22). If this feature is used and then modifications are made to the file, then Generate Program must be rerun. The Generate

Program function can only be run once; if the user needs to run this function again they must save the LD file and then re-open it before Generate Program can be run again.

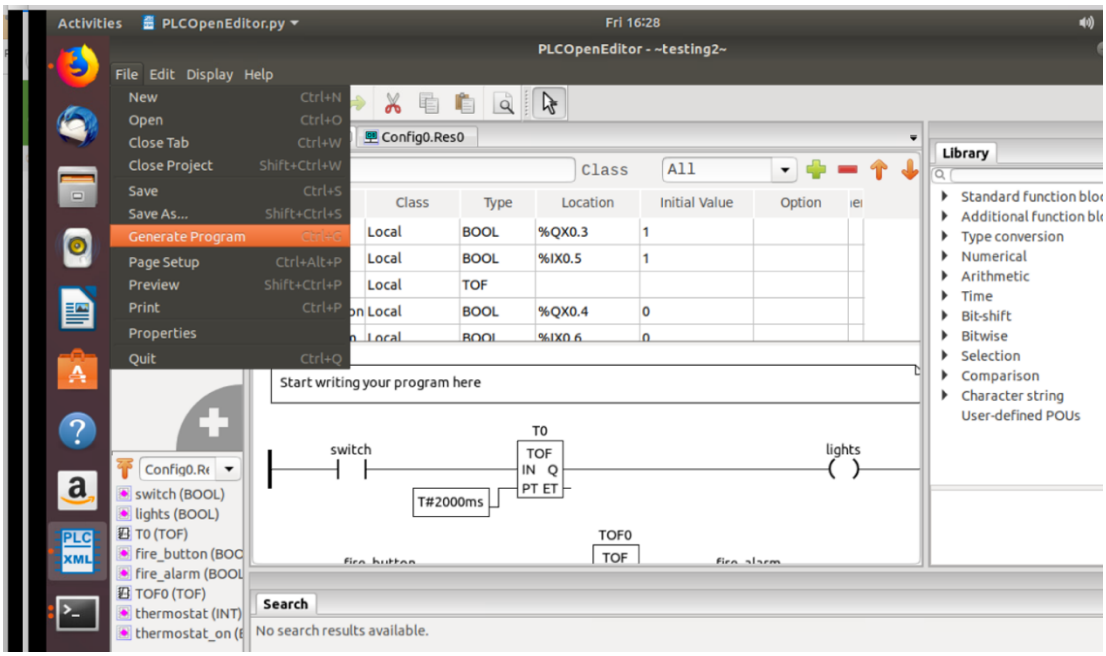


Figure 22 *PLCOpen Editor Generate Program*

4.4.2ScadaBR

The challenge with ScadaBR is that the online documentation and forums are in Portuguese. There is a walkthrough document that has been created in English, but for any troubleshooting or forum discussions, it is essential that Google Translate or a similar tool is utilized – unless you are fortunate enough to speak Portuguese.

CHAPTER 5. CONCLUSION

A virtual model of Iowa State University's ICS was built using open source tools within the ISERink environment that contains:

- connections to data sources
- a sampling of diverse, targeted variables to be monitored, and
- alerts for monitoring an industrial system using open source tools

While deployment of the system in Iowa State's actual environment was beyond the scope of this project, it would simply entail connecting the actual sensors in the buildings in lieu of the virtual Modbus Master Simulator. ScadaBR would become the central monitoring device that would run in parallel to the existing, propriety system so that their outcomes could be compared. Once the virtual system has been validated, testing for security vulnerabilities could commence.

While many people are attuned to the issues of computer security related to protecting sensitive data such as health care information and financial data, the security of physical structures can and should be added to those efforts. This test bed allows for the pursuance of of that security.

Presumably the public sector responsible for critical systems, such as energy and nuclear waste, is at the forefront of ICS monitoring. Private organizations must also become aware of the potential to compromise a building itself, perhaps to impact the contents in the building, to disrupt the systems needed to maintain the integrity of the building, or as Target experienced in 2010 when its HVAC system was hacked, to access other connected systems like financial information.

This project provided an informative look into the world of ICSs, the standards that define them, and the open source tools that drive them. Understanding these systems is a first step in exploring the potential vulnerabilities that exist within them. ICSs are a prime example of Security Through Obscurity. Demystifying these systems and establishing actual security protocols is critical and should become a standard consideration in all organizations' security audits.

CHAPTER 6. FUTURE WORKS

6.1 Future Integration Testing

A possible future avenue for this project would include looking into additions to Iowa State's ICS and testing that integration with the simulated environment before deploying it into the actual environment. In this way, the ICS test bed would function as a testing environment and limit the amount of testing in a production environment. An example of such future integrations could include adding locks and or door security to Iowa State's ICS.

To further determine the accuracy of this test bed and how realistic it truly is, another future project would include integrating real equipment used in Iowa State's ICS with the test environment to be able to more accurately simulate the real environment and potential security flaws. Every component of this test bed could be replaced with a different tool in order to allow for maximum flexibility.

6.2 Security Testing

This test bed can be used to begin assessing numerous aspects of ICS security. Some basic security considerations can be identified by examining the CIA Triad.

- Confidentiality – Is this system confidential? Can anyone see what devices are being monitored or see the state of such devices being transmitted across the network?
- Integrity – Can the information being sent in regard to the status of devices be tampered with? Is the information received correct?
- Availability – Is the state of the devices constantly available? Can someone interfere with the connection and prevent the sensors from sending updates?

Another component that should be considered is the security of the protocols used, the protocol currently in place is Modbus TCP/IP, running on port 502. The Modbus protocol has been around since the late 1970, much before security was a consideration.

Originally, ICS security has relied entirely on an air gap. The majority of groups have shifted away from air gaps for convenience and are compensating with firewalls. Would a simple firewall be enough security to protect these systems?

6.3 ICS Cyber Defense Competition

As this model was built within the ISERink environment for future security testing, integrating it into a future ICS based Cyber Defense Competition (CDC) would give competitors a chance to diversify their exposure and security knowledge while remaining confined within a safe environment. Recent CDCs have attempted to include a cyber-physical element, but until now that was mostly reliant on graduate students' personal exposure and creation of power cyber devices.

6.4 IoT Security

ICS Security is in its infancy, as is home automation. The tools used in this project are typically utilized by hobbyists for IoT home automation projects. Advances in this field may directly or indirectly benefit home automation and therefore IoT, which desperately needs security development.

REFERENCES

Almas, Muhammad Shoaib, et al. "Open source SCADA implementation and PMU integration for power system monitoring and control applications." *PES General Meeting| Conference & Exposition, 2014 IEEE*. IEEE, 2014.

Alves, T. "The Open PLC Project." (2016).

Barnes, Kenneth, and Briam Johnson. *National SCADA test bed substation automation evaluation report*. No. INL/EXT-09-15321. Idaho National Laboratory (INL), 2009.

"ISEAGE Internet-Scale Event and Attack Generation Environment" <http://www.iseage.org/>.

Mirkovic, Jelena, et al. "The DETER project: Advancing the science of cyber security experimentation and test." Technologies for Homeland Security (HST), 2010 IEEE International Conference on. IEEE, 2010.

"Modbus Master Simulator - free software utility for testing Modbus slave devices." <http://en.radzio.dxp.pl/modbus-master-simulator/>.

"ScadaBR" <http://www.scadabr.com.br/>.

APPENDIX OPENPLC ST FILE

```
PROGRAM town
  VAR
    switch AT %QX0.3 : BOOL := 1;
    light AT %IX0.5 : BOOL := 1;
  END_VAR
  VAR
    T0 : TOF;
  END_VAR
  VAR
    fire_switch AT %QX0.4 : BOOL := 0;
    fire_alarm AT %IX0.6 : BOOL := 0;
  END_VAR
  VAR
    TOF0 : TOF;
  END_VAR
  VAR
    thermostat AT %QW5 : INT := 63;
    thermostat_on AT %IX0.7 : BOOL := 1;
  END_VAR
  VAR
    GT29_OUT : BOOL;
  END_VAR
```

```
  T0(IN := switch, PT := T#2000ms);
  light := T0.Q;
  TOF0(IN := fire_switch, PT := T#2000ms);
  fire_alarm := TOF0.Q;
  GT29_OUT := GT(thermostat, 0);
  thermostat_on := GT29_OUT;
END_PROGRAM


PROGRAM troxel
  VAR
    switch AT %QX0.1 : BOOL := 1;
    light AT %IX0.1 : BOOL := 1;
  END_VAR
  VAR
    T0 : TOF;
  END_VAR
  VAR
    fire_switch AT %QX0.2 : BOOL := 0;
    fire_alarm AT %IX0.2 : BOOL := 0;
  END_VAR
  VAR
    TOF0 : TOF;
```

```
END_VAR
VAR
  thermostat AT %QW1 : INT := 65;
  thermostat_on AT %IX0.3 : BOOL := 1;
  water_temp AT %QW3 : INT := 75;
  water_temp_on AT %IX0.4 : BOOL := 1;
END_VAR
VAR
  GT31_OUT : BOOL;
  GT29_OUT : BOOL;
END_VAR

T0(IN := switch, PT := T#2000ms);
light := T0.Q;
TOF0(IN := fire_switch, PT := T#2000ms);
fire_alarm := TOF0.Q;
GT31_OUT := GT(thermostat, 0);
thermostat_on := GT31_OUT;
GT29_OUT := GT(water_temp, 0);
water_temp_on := GT29_OUT;
END_PROGRAM
```

```
CONFIGURATION Config0


  RESOURCE Res0 ON PLC
    TASK TaskTroxel(INTERVAL := T#50ms,PRIORITY := 0);
    TASK TaskTown(INTERVAL := T#50ms,PRIORITY := 0);
    PROGRAM Inst0 WITH TaskTroxel : troxel;
    PROGRAM Inst1 WITH TaskTroxel : town;
  END_RESOURCE
END_CONFIGURATION
```