7-2011

A water distribution and treatment simulation for testing cyber security enhancements for water sector SCADA systems.

Justin Robert Adams
University of Louisville

A WATER DISTRIBUTION AND TREATMENT SIMULATION FOR TESTING CYBER SECURITY ENHANCEMENTS FOR WATER SECTOR SCADA SYSTEMS

By

Justin Robert Adams
B. S., University of Louisville, 2010

A Thesis
Submitted to the Faculty of the
University of Louisville
J. B. Speed School of Engineering
as Partial Fulfillment of the Requirements
for the Professional Degree

MASTER OF ENGINEERING

Department of Computer Engineering and Computer Science
University of Louisville

July 2011

A WATER DISTRIBUTION AND TREATMENT SIMULATION FOR TESTING SECURITY ENHANCEMENTS IN WATER SECTOR SCADA SYSTEMS

Submitted by:__________________________________
Justin Robert Adams

A Thesis Approved On

____
(Date)

by the Following Reading and Examination Committee:

____
Jeffrey L. Hieb, Co-director

____
Rammohan K. Ragade, Co-director

____
James H. Graham

ACKNOWLEDGEMENTS

The author would like to thank Dr. Hieb and Dr. Graham for the initial ideas and the opportunity to work on this project and Dr. Ragade and Dr. Hieb for constant advice and help though out the creation process. The final thanks goes to Brad Luyster for collaboration because without the prototype implemented by him this wouldn’t have been possible.

ABSTRACT

Supervisory control and data acquisition (SCADA) systems are used by many critical infrastructures including electric power production and distribution, water and waste water treatment, rail transportation, and gas and oil distribution. Originally isolated proprietary systems, SCADA systems are increasingly connected to enterprise networks and the Internet and today use commercial hardware and software. As a result SCADA systems now face serious cyber-security threats. The need for testing and evaluation of developed cyber-security solutions presents a challenge since evaluation on actual systems is usually not possible and building complete physical testbeds is costly. This thesis presents the design and development of a water systems simulation for testing and evaluation of cyber-security enhanced field devices.

The simulation consists of two main parts: a human machine interface/master terminal unit (HMI/MTU) component and a water treatment and distribution component. The HMI/MTU part supports new security protocols used to communicate with the hardened remote terminal unit (RTU). The water system simulates a water treatment and distribution center. A data acquisition (DAQ) module was used in conjunction with LabVIEW^{TM} to create a water distribution and treatment simulation that could be interfaced with an actual field device. Field device I/Os are wired to the DAQ which then interface with the LabVIEW$^{\text{TM}}$ simulation. The simulation supports: selectable polling of I/O, graphical representation of I/O, random water usage, constant water usage, and simulation data collection. The simulation uses a modular design pattern so that it can be easily extended in the future. Initial testing with a hardened RTU prototype confirmed the ability of the simulation to interact with real hardware and identified some minor errors in the prototype's security protocol implementation. With additional DAQ devices the simulation could be extended to simulate larger water systems.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

SCADA systems have found their way into most parts of the US infrastructure. Whether it is for HVAC or total plant operation, people have come to rely on these systems to get many jobs done. They are used in electric generation plants, chemical plants, water plants, and many other industrial settings. Many of the tasks controlled by these systems require constant monitoring and immediate response time for efficiency and safety. As this technology has grown, security for these systems has fallen behind. A significant security issue is in cyber-security due to a lack of isolation between these SCADA systems and enterprise networks and the Internet. This has opened up SCADA systems to a world which is full of hackers, viruses, and other malware. A recent malware that targeted SCADA systems was Stuxnet, which included a PLC rootkit that was designed specifically for SCADA systems and is suspected of targeting Iranian nuclear installations. While it didn’t cause any damage to the general population, it did manage to damage some centrifuges which hindered plant production.

Over the past few years, there has been a significant amount of research and development towards developing more secure SCADA systems. The majority of current solutions are considered IT fixes, consisting of firewalls and protection done in software. While this is a good start, it fails to address all of the current problems. Cyber security efforts include the development of protocol enhancements and security appliances for SCADA systems. For example, the University of Louisville is currently working on a prototype security hardened remote terminal unit (RTU) for the water sector that incorporates several security features and a hardened architecture. There are several other organizations that are also creating custom solutions to these problems, but they are running into testing problems. Testing hardware to be used in SCADA applications in an actual SCADA environment can be dangerous. If any errors are found, it could be hazardous to workers and customers. For example, it could be possible that tests show new hardware and/or software inadvertently added too much chlorine to the water supply. Similar errors could also lead to tanks going empty which would severely hinder water distribution.

For the reasons stated above, using testbeds and simulations is vital for the advancement of SCADA security. The advantage of testbeds and simulations is that they can test prototypes and new designs without jeopardizing actual systems. Testbeds and simulations can have an expensive startup cost, especially if they are to mimic a sophisticated SCADA system; however, they can be used for multiple tests, don't take up as much space, and are nowhere near as expensive as building a separate SCADA system just for testing. They can also test situations that wouldn't be easy to create on an actual system. As development of new SCADA cyber security technologies continue, simulation for testing and evaluation will become increasingly important. This thesis focuses on creating a simulated testing environment for security hardened field devices in the water sector, specifically the prototype designed by Hieb and Graham [11].

The organization of this thesis is as follows. Chapter Two contains background information on SCADA systems. It then discusses the current cyber-security problems and what is being done to prevent these in the future. It also presents proper simulation and modular design and how current simulations have advanced cyber-security. Chapter Three covers what parts make up the simulation and how each part should function after completion. Chapter Four talks about the actual parts created and what information was used to finish each part. It also gives an overview of the GUI on each part so that someone could use the programs for testing. Chapter Five discusses what was done to test the simulation and how it has already helped test the field device used for this thesis. Lastly, Chapter Six summarizes all the previous chapters and tries to outline possible future work for the simulation.

II. LITERATURE REVIEW

A. SCADA Systems

Supervisory controls and data acquisition (SCADA) systems provide telemetry to industrial processes [5]. They are commonly used in plants like electrical, water, and chemical manufactures. A SCADA system is composed of three main components, the master terminal unit (MTU), one or more remote terminal units (RTU), and a communication network. There is also a human user interface (HMI) which is sometimes part of the MTU. The very basic layout of one of these systems is referred to as a one-to-one system. This means there is only one MTU and one RTU. As the system grows the number of RTUs used grows significantly and if the system gets large enough, there could be more than one MTU. For larger systems, the HMI and MTU tend to become separate components.

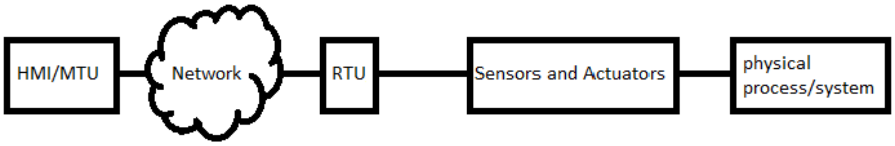


FIGURE 1 – A one-to-one SCADA system

The MTU is what all the RTUs connect to using the communication network. This could be considered the manager of the system. It asks the RTUs for data from day to day operations. It is also what collects data and stores it to be viewed later. RTUs would be like the normal employees of the system. They are each given individual tasks and are expected to report to the MTU on the status of these tasks when asked. The communication network would be like a language that the workers and employees use to relay messages. The HMI would be like the big brother of the system. It gets its information from the MTU but is usually a more graphical representation of the most recent data. This is where operators can actually observe what is happening over the whole system and ask the MTU to push changes to the RTUs. Its purpose is to "provide the user with the capability to exercise control over a specific device and to confirm its performance in accordance with the directed action" [1]. SCADA systems are used to maintain normal functions, collect data from the system, and relay important information to the operators. For the Water Sector SCADA systems are used to provide remote status and operation of pumps, control the addition of treatment chemicals, and monitor reservoir levels and flow rates.

1. History

The very first SCADA systems were analog based. This meant all of the controls were analog dials and knobs and each signal had its own data line. This created a very large footprint in a plant and made these systems very costly.

The original HMI’s consisted of a large variety of analog controls as dials and knobs and lights on a control board. Since this was the case, the system wasn’t connected to the outside world which meant security usually ended at locking the door to the room where it was held. It also meant that adding or removing parts to control the system was quite an expensive and long ordeal. The old setups were eventually turned into Personal Computers, also known as PCs. This made updating the interfaces much simpler and more cost effective [4].

The MTU controls what the RTUs do as well as inform the HMI about the most current states of sensors and actuators. This is usually to regulate something. In the Water Sector this would include turning on and off pumps to maintain an acceptable water level in tanks, the addition of chlorine or a similar chemical to help maintain safe drinking water, or other functions that require constant observation.

Remote Terminal Units are the next part of this system. They are what collects and directly connects to the sensors. They relay the collected data to the MTU or HMI/MTU and take orders on what to do from the MTU. Their big advancement was when Programmable Logic Controllers, also known as PLCs, were introduced in conjunction with microprocessors. This allowed for a cheap solution to add RTUs as employers saw fit without having to get them custom made [4].

The final part is the communication network. These were originally all analog cables that were bundled together. This made a whole system contain as many cables as signals unlike the newer digital cables that can carry multiple signals on one cable. When PLCs and Microprocessors took off serial connections became the way to transmit data between them. This allowed for digital connections, which resulted in significantly less cables going from one point in the system to another. The most recent enhancement for cables is the use of Ethernet cables. They are extremely cheap, can be used reliably for longer distances, and can introduce remote work to any part in the whole system. More recently, this has also led to online data storage. This is very helpful for data analysis but tends to be overlooked in the budget. Over the lifetime of a system, databases tend to cost a lot of money to maintain. Because the startup costs are also usually high, they are usually not custom, which means it will be less secure than a custom job [4].

2. Cyber – Security Vulnerabilities in SCADA

Many current cyber security problems in SCADA systems are due to advances in technology. These advances have made current systems consist largely of PCs and commercial hardware for MTU and RTU platforms and the use of TCP/IP for parts of the control network. The problem with Ethernet cables is that most enterprise networks already use them as their main network and the data being collected on SCADA systems is wanted for many business aspects. This resulted in a lot of these enterprise networks simply being connected to SCADA systems. This creates a lack of separation for these systems. The isolation that existed before this helped a lot with security but this simple connection now allows potential internal hackers the opportunity to shut down or interface with SCADA systems by a variety of attacks. These include but aren’t limited to signal injection, signal interception, denial-of-service attacks, and logical reprogramming of the system. Other possible side effects could be just having a busy office network that results in delays of time critical information or has enough mass broadcasts to shut down

PLCs that aren’t equipped to handle them. Most office networks are also connected to the internet. This opens SCADA systems to the rest of world. Hackers, Viruses, Worms, and other common problems associated with Internet connections have become threats. It also means that a standard has to be used to talk between the HMI and RTUs. Two common protocols used in SCADA systems are MODBUS and DNP3. This means that someone that wants to get into the system, a hacker, should be able to do some damage by simply understanding these two protocols [10]. The last side effect to using Ethernet cables is that the total number of cables significantly decreases due to the conversion to digital instead of analog signals. This means that it becomes easier to identify which cables are going where and it could make it easier to hack a line going to a specific RTU.

The major benefit of using PCs is that they can create a cost effective and adaptable system while allowing its users to monitor the RTUs at a rate that makes the system more like a real time control system. The down side is with PCs, the system now has to worry about the operating systems vulnerabilities, patches, passwords, and basic upkeep along with system vulnerabilities. It also gives potential hackers greater processing power to try and figure out the system. They could try to do this with Denial of Service attacks on sensors, Integrity attacks, and Phishing attacks. Consequences of attacks on these systems can result in financial and environmental damages as well as endangering public health and safety.

While the abundance of PLCs in the SCADA world keeps cost down, it means that there is another common standard used in almost every system. This means that with PLC programming knowledge, an intruder could program the PLC to give false data or any number of things that would be harmful to the system. This would hinder the integrity and availability of the SCADA system which is why research is being done to secure RTUs [4].

3. SCADA Security Solutions

It is very important to keep up with research in this field because "SCADA systems are the backbone of the critical infrastructure, and any compromise in their security can have grave consequences" [10]. There are many security challenges that research is trying to find solutions for. Some of these challenges are: Access Control, Firewalls and Intrusion Detection Systems, Protocol vulnerability, Cryptography and key management, Device and OS security, and security management [15]. An effective way to analyze a large portion of these challenges is through the use of Attack Trees. Attack Trees allow people to see the easiest ways to attack their system and the most likely locations the attacks are going to happen at. This allows people to add security measures where security would be most beneficial [5]. To help with the problem of Integrity and Authentication in RTUs, Bhatt, Graham, and Patel tested no security, SSL/TLS, Authentication Octets using software, Authentication Octets using hardware, and Challenge Response [3]. The time it took for communication with Authentication Octets in both cases was significantly greater than with no security, so it was not a viable solution for many applications. SSL/TLS and Challenge Response showed some promise, but SSL/TLS has known vulnerabilities. Their recommendation was to use Challenge Response Authentication to improve SCADA security. Challenge Response is a fairly simple concept where two devices talk to each other. If a restricted operation is asked of by one device, the device that asked will get a message that is known as a challenge. This means that this device will need to prove it has authentication to execute the restricted operation. The device will then respond with some authentication decided upon in the protocol. If the authentication is valid the operation will be executed. The Security Hardened Field Device Prototype described in *Designing Security-Hardened Microkernels for Field Devices* uses this approach [11].

B. Simulation

A simulation is simply an imitation of a real problem. The purpose is to gain the most knowledge from a limited amount of trials. It is usually a good idea to use a simulation when the resources to test new ideas are limited, the consequences could be hazardous, the group doing the tests wants to check a reaction to a stimulus that happens very rarely, or these tests also want to be run on a later date [8]. For instance, the incorrect handling of pump valves to a water tank by the RTU or PLC in a SCADA network might result in a lack of water for a city or might create an overflow of water that goes into the tank. Both are very serious issues that want to be avoided at all costs. Simulations tend to be software modules that represent parts of a system. An example of these software modules that are made specifically for simulations is SIMSCRIPT III described by Bailey, Marjanski, Markowitz, and Rice [2].

1. Simulation Design

There are approaches to building a simulation. One is classical Design of Experiments (DOE) and the other is modern Design of Experiments. Classical DOE's tend to use random variability to try and account for outside forces on the system. Most software testing tends to use modern DOE's. Modern DOE's stay away from random variability because software is usually written with a finite set of paths and each path has an expected outcome. For simulating water treatment and distribution a classical DOE approach would be most appropriate [8]. There are also two typical types of simulations, continuous and discrete. Most situations in this world can be modeled with both types, but in general one is significantly less complex for a given situation. A continuous simulation tends to involve differential equations because at any point in time, it needs to be modeled correctly. Discrete simulations tend use step functions and multiple states. There are a finite amount of events that can happen and the simulation knows every possible order of events that is possible [6].

There are many programs that are designed for water simulation. Most focused on the chemical process part of treatment or waste treatment or the actual design layout of a water treatment plant. The few that weren't didn't seem to use any digital or analog inputs/outputs. A completely software based water treatment simulation does not allow for testing of hardware without significant modifications. The specific ones that were found are called HydrauliCAD [13], WatPro [14], and TECHNEAU Water Treatment Simulator [7].

2. SCADA – Security Simulation

Some work in simulation for SCADA cyber-security has been done, most notably The National SCADA Testbed (NSTB). This specific SCADA testbed focuses on electric power but could be used in the water sector. A testbed is a combination of hardware and a software simulation that represents all vital parts of a system including the network and is created for specific testing. Most testbeds tend to be internal ones for specific companies and the majority of them don't actually allow for testing with current hardware. In *Building a SCADA Security Testbed* the authors "propose an open-source, modular SCADA modeling testbed that allows real time communication with external devices using SCADA protocols" [12]. This was proposed to run scenarios related to Denial of Service attacks along with many other security vulnerabilities with SCADA systems. Scenarios are specific tests run on testbeds. The Challenge Response data along with SSL/TLS and Authentication Octets mentioned above in SCADA Security Solutions was collected from a scenario on a testbed at the University of Louisville [3]. This testbed consisted of 5 RTUs and one MTU. They used DNP3 as the protocol and had an HMI to mimic real-life monitoring. Four of the five RTUs were located on the University of Louisville campus and the fifth was about 100 miles away and connected through the internet [3].

C. Modular Design

Modular Design uses a series of modules that can stand alone to accomplish a larger task. Each module has a purpose and tends to be structured more like a physical part than with normal coding standards. This approach is particularly useful in building scalable simulations that are easily modifiable. For a Water Plant simulation, this means there is a module for Pumps, Tanks, and other parts of the system. This structure tends to aid itself to working on smaller parts which is helpful when trying to distribute workloads and allows updating of modules without updating the whole structure. The last part that a Modular Design does is the ability to reuse modules in similar structures without having to change code [9]. This type of design is very helpful in simulation design and implementation. Since simulations tend to be specialized for a given test, the option to swap out modules depending on the test allows one testbed to be used for multiple tests which saves money and time.

III. DESIGN OF SIMULATION

Cyber-security for SCADA systems is a constantly changing environment. To stay ahead in this environment constant research and development has to be done. This is done through the use of simulations, testbeds, and evaluations of current systems. To keep ahead, simulations and testbeds can be used to help evaluate many prototypes and bolt on security solutions. This allows full testing in a safer environment than an actual system and keeps down cost. A security hardened field device prototype developed by Hieb and Graham was created for use in the Water Sector [11]. While the initial testing and evaluation shows promising potential, additional testing is needed before the device could be tested in a commercial application. This thesis focuses around the development of a small residential water system simulation created for testing this and other prototype Water Sector SCADA field devices.

For this simulation a one-to-one SCADA system is used and is made up of two main design components. The first component is the HMI/MTU, what operators would normally be viewing, and the second is the water system, the simulation that would mimic the normal behaviors of a water treatment and distribution system. The security hardened RTU would be placed between these two parts and interface with the simulation in two different ways. The HMI/MTU would interface to the RTU through UDP on a network while the water system simulation would interface with the hardened RTU’s analog and digital I/O.

A. Properties of the HMI/MTU

The HMI/MTU that is to be created needs to have a graphical display and it will need to be able to talk to the security hardened RTU prototype. Graphically, there needs to be a way to represent both analog and digital I/O. For this simulation digital I/O will primarily be used to turn on pumps or open valves. Analog values tend to be tank levels and pipe flows. In SCADA systems RTUs are the server, and the HMI/MTU polls the RTU, therefore the HMI/MTU simulation will also need to have a way to specify a polling frequency. The HMI/MTU needs to continually poll the RTU to keep displayed values as close to real time values as possible. To be able to talk to the prototype, a basic understanding of the device is needed. The prototype is an implementation of Hieb and Graham’s design [11]. From the perspective of simulation design the important parts include the network protocol, the challenge/response design for restricted I/O, SHA-256 hashing for using the challenge/response, and the need for a User ID. SHA stands for secure hashing algorithm. Using SHA-256 creates a 32 byte hash of the input and gives the input 128 bits of security against attacks [17]. For testing, a static IP address is used, but to maintain flexibility the HMI/MTU should be able to connect to any IP address and port number specified by the user.

1. Simple SCADA protocol used by prototype

This section describes the SCADA protocol used to communicate with the RTU over the UDP protocol. The protocol is a simplified SCADA protocol discussed in the prototype implementation [11]. The structure of a protocol UDP packet is as follows. The first byte will always be the User ID followed by a one byte Operation field and then followed by a multi byte Data field. Data is interpreted differently depending on the Operation. Operations are as follows.

TABLE I

ACCEPTED OPERATIONS BETWEEN THE RTU AND HMI/MTU

Operation #	Operation	Direction
0	Read	HMI/MTU → RTU
1	Select/Write	HMI/MTU → RTU
2	Operate	HMI/MTU → RTU
3	Challenge	RTU → HMI/MTU
4	Challenge Response	HMI/MTU → RTU
5	Read Response	RTU → HMI/MTU

For a Read operation no Data is present. For Select/Write and Operate operations that specify the I/O that is to be written to, the Data is one byte. For a Challenge operation Data is 4 bytes that were randomly generated by the RTU known as the snonce and is used in the hashing to get a valid Challenge Response. For a Challenge Response operation Data is 4 bytes randomly generated by the HMI/MTU known as the cnonce and is followed by 4-32 bytes, depending on security, of data that was the result of the hashing. A Read Response operation’s Data is a byte that shows the current status of the I/O that was asked about. To get a valid challenge response, the HMI/MTU must take the snonce, cnonce, and a pre-shared secret, S_{ID}, that is associated with a specific User ID, U_{ID}, on the RTU and concatenate the byte arrays and then use SHA-256 hashing on the byte array. This will produce a 32 byte array. This is what is returned in the Challenge Response as the 4-32 bytes. The current prototype uses just the first 4 bytes in the array. If the system needs to be more secure, all 32 bytes could be sent and verified. The HMI/MTU will need to get from the user: the U_{ID} and S_{ID}. The preshared secret will be referred to as a key from here on. This concludes the RTU specific requirements of the HMI/MTU.

B. Water System Simulation

Most water systems consist of both water treatment and water distribution. A water distribution system consists of one or more pump stations. Each pump station has a water tower (tank) which is filled by turning on pumps located next to the water tower. Water flows from the water tower to customers, and pressure is maintained by gravity. The pumps keep the water tower filled with water. Before being distributed, water must first be purified, which is performed by the water treatment system. A water treatment system essentially consists of some number of interconnected reservoirs. Chemicals are added to the water as it flows from one reservoir to another, with chlorine being the most common purification chemical. Foreign material is also allowed to settle out of the water while it is in the water treatment system. The hardened RTU could be used in either water treatment or distribution. Therefore both parts of the water system are to be simulated. Water Treatment will consist of a reservoir, a chlorine or other chemical additive monitor, and a final holding tank before it goes to the Distribution system. The Distribution system for this specific simulation will consist of a pumping station that will consist of one tank and two pumps. This is currently a limit due to the amount of I/O on the DAQ.

1. Water Treatment

The specific layout of the water treatment system that is being modeled consists of a reservoir, a tank, a flow valve, and the pipe. It is shown in Figure 2.

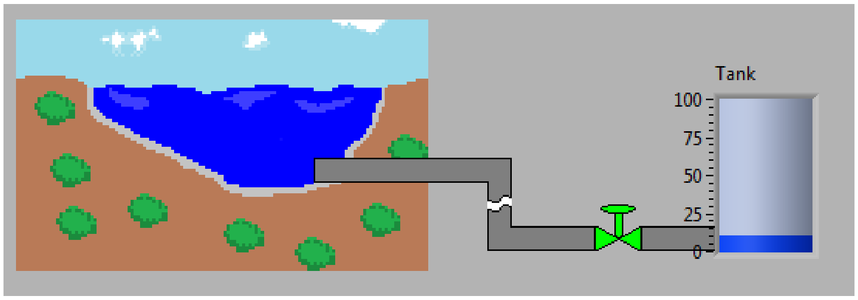


FIGURE 2 – Water Treatment Layout

The reservoir will have the assumption that it won’t ever run out and that it is gravity fed. The elevation of the reservoir in relation to the tank will help to determine the GPM that will be flowing into the tank. It will also help in showing a very rough estimate of PSI out of the reservoir. The GPM calculated will help add chlorine to the tank. This same GPM will be assumed as the out flow. Since the in and out flow will be known, this will make it easier to find the actual ratio of chlorine in the tank. The variables that will need to be specified at start up should be the tanks `Max Volume`, the `Solution’s ppm`, the `Elevation Height` of the reservoir, the `Pipe Diameter` from the reservoir, the `Flow Percentage Open`, and the `Equalized Volume` of the tank. All of these can be changed during run time. The one input that will be needed from the DAQ is the `Set ppm` that should be able to change and affect the amount of chlorine added to the tank. The equations that are used to track the ppm in the tank and PSI and GPM in the pipe are shown in Appendix II.

2. <u>Water Distribution</u>

The specific layout of the water distribution system that is being modeled consists of two pumps, a tank, and pipe. It is shown in Figure 3.

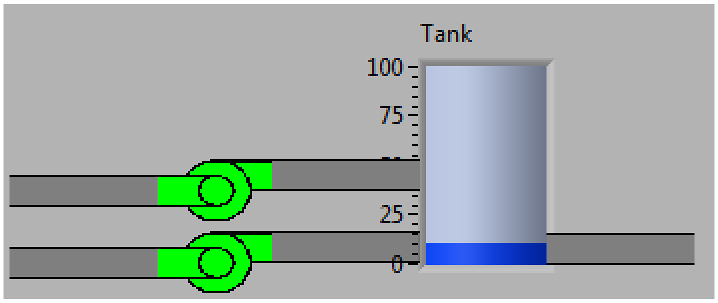


FIGURE 3 – Water Distribution Layout

The interface with the DAQ will emulate pump and tank readings. The pump I/O should just be reading what the current state is, on/off, from the RTU and react accordingly. The tank level should be set inside the Water Distribution part and should only be readable to the RTU. The starting tank level and the pump flow rates should be specified before this part should start running. To accurately design a system, there has to be an in and out flow related to the tank level. The pumps should be able to give the program a decent number for flow in, but to represent a decent out flow many parts will need to be combined. The out flow should start with some formula, either a step or continuous function that approximates normal water usage for an individual person with a small variance to keep the system non deterministic. The function will be using approximates so the amount of people needed to make this accurate will have to be high, but since water plants tend to service many people this shouldn’t be a problem. There should also be an out flow that is added to the current one that focuses on abnormalities like extra units running for prolonged periods of time or people not running units at all, for vacation or vacancies. This will also include extra units running or being turned off randomly over a specific interval. Since the main reason for having a simulation is to allow for variability, there should be a way to represent some normal disasters, like a failed pump and any other problems that might arise that could hinder a normal Water Distribution Center. Lastly, there should be a way to collect data for long periods of time. This can be achieved by being seen on a graph or saved as a file so that different runs of the same simulation can be compared to each other.

IV. IMPLEMENTATION OF SIMULATION

This chapter describes the implementation of the HMI/MTU and water system simulation. It includes the mathematical models used and the justification for specific implementation choices. Section A describes the creation of the HMI/MTU, the second section describes the water treatment part of the simulation, and section three discusses the water distribution part of the simulation. To make things easier to understand, Figure 4 has a drawing that shows what will be simulated and how it will connect to the RTU.

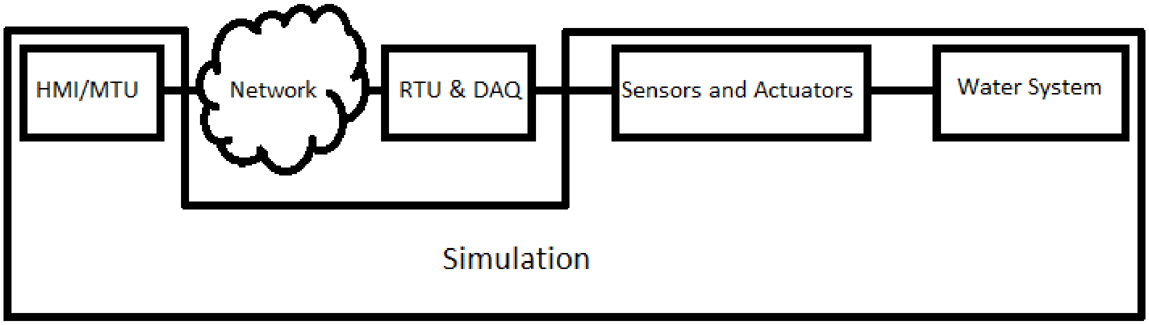
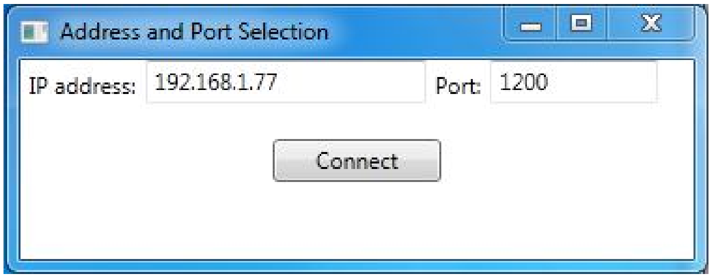


FIGURE 4 – Simulation Diagram

A. Control HMI/MTU Implementation

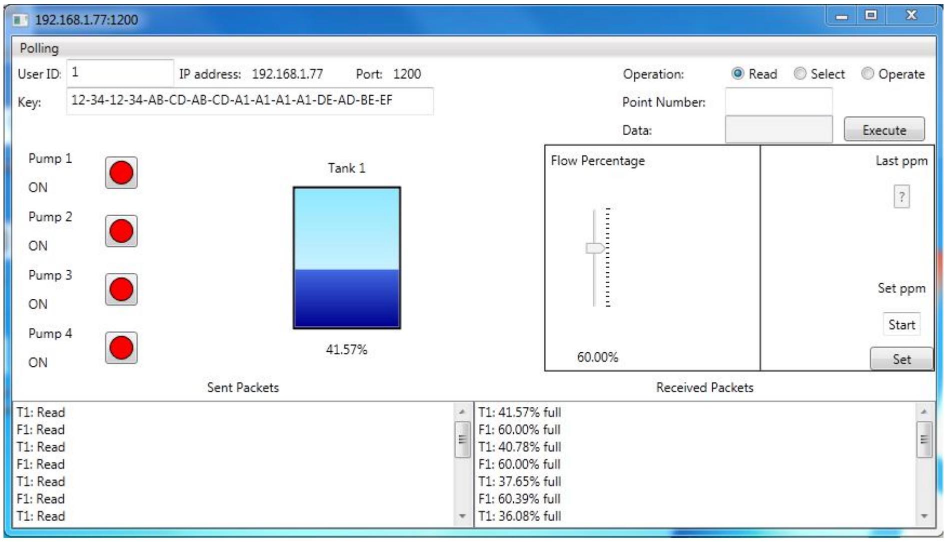
The HMI/MTU was created in C#’s WPF format. This was decided upon because of past experience with this format and for aesthetic reasons. Creating an HMI/MTU consists of three main components. They are the presentation layer, the operations, and the security. The presentation layer is shown in Figures, the security consists of the `User ID` and `Key`, and the operations are connecting to the RTU, polling, and handing user requests. Since the first thing that needs to be done for operations is connecting to the RTU, the first window that pops up on execution is a window that asks for an IP address and port number of the RTU to which the HMI/MTU will connect. This is shown in Figure 5.



IP address: 192.168.1.77 Port: 1200

FIGURE 5 – Address and Port Selection

This is the only input the user can give that isn’t likely to be changed during monitoring. This is used to know where to send packets and where to expect replies from. After this is obtained, the main window is opened which contains graphical representations of tanks and pumps. The interface also includes the original testing elements which allow specifying a point number, operation, and if applicable the data to be set. This was done for testing purposes, but left in the HMI/MTU so that during SCADA testing the HMI/MTU can be used to build specific protocol requests.



User ID: 1 IP address: 192.168.1.77 Port: 1200

Key: 12-34-12-34-AB-CD-AB-CD-A1-A1-A1-A1-DE-AD-BE-EF

Operation: ● Read ○ Select ○ Operate

Point Number:

Data:

```
T1: Read
F1: Read
T1: Read
F1: Read
T1: Read
F1: Read
T1: Read
```

```
T1: 41.57% full
F1: 60.00% full
T1: 40.78% full
F1: 60.00% full
T1: 37.65% full
F1: 60.39% full
T1: 36.08% full
```

FIGURE 6 – HMI/MTU Main Window

Because accuracy on the tank levels and pumps is important to know as close to real time as possible, there is a background polling of each data point that polls the tanks most frequently but also polls the pumps for change or for a current state. The poll times can be changed in the `Polling` Menu which allows the user to set any of the current poll frequencies which includes turning off polling to certain points. Because the protocol doesn’t always specify which point the response is for, the UDP sockets are different for each pump and tank in the system. This allows for us to know which point the response was meant for. For polling to work correctly, the `User ID` has to be valid and the `key` must match on the RTU for that `User ID`.

The security part starts with the hashing that is created once a challenge has been issued. This is created by simply concatenating the key with the snonce and cnonce. A snonce is a server nonce and cnonce is a client nonce. This byte array is then placed in the SHA-256 hashing and the result is a 32 byte array. In this specific prototype only 4 of these bytes are forwarded to the RTU. To help with security measures, each operator would need to have a `User ID` and `key` to view anything current on the HMI/MTU.

The `User ID` and `key` can both be changed at any point in time on the main window to allow other users to log in easily or to deny access by simply removing the key from the window. This will make the HMI/MTU fail at all attempts to read, select, or operate on the RTU.

If there is a valid `User ID` and `Key` the `UDP` class handles all the commands that are sent and received from the RTU. The `UDP` class is inherited by `AnalogUDP` and `DigitalUDP` so that they can handle the commands for GUI elements. `AnalogUDP` updates `Tanks, DigitalUDP` updates `Pumps`, and `FlowUDP` updates pipe flow percentage. The `Pumps` are buttons with ellipses that are colored in by 2 different colors depending on which state they are in. These were created to make a more user friendly design. There is also text next to them for those who would rather read the state. The `Tanks` are made up of 2 rectangles with the bottom one changing its height depending on the water level. There is also a percentage full to show more exact numbers for water level. The flow percentage and Set ppm part goes to the Water Treatment Simulation. This shows the flow percentage in the pipe currently and shows the current set point for ppm as well as showing what it was set at last. This allows for people to change the ppm while viewing the current ppm. Since this sensor is analog, there is no actual way for the HMI to tell if the value is actually correct. The only way for it to know the set point isn't accurate is if the packet wasn't delivered. The very last part of the main window that can be helpful is the Sent Packets/Received Packets lists. If everything is going correctly the commands will show up with the data needed to understand what is being sent or received, but if there is a connection problem, the `key` doesn't match the `User ID`, or the `User ID` doesn't have permission to access all points you will see "Failed to receive

Response". This is included for troubleshooting. When the window is shrunk to its minimal form it covers up all parts of the window that aren't needed for operation.

B. Creation of Water Treatment

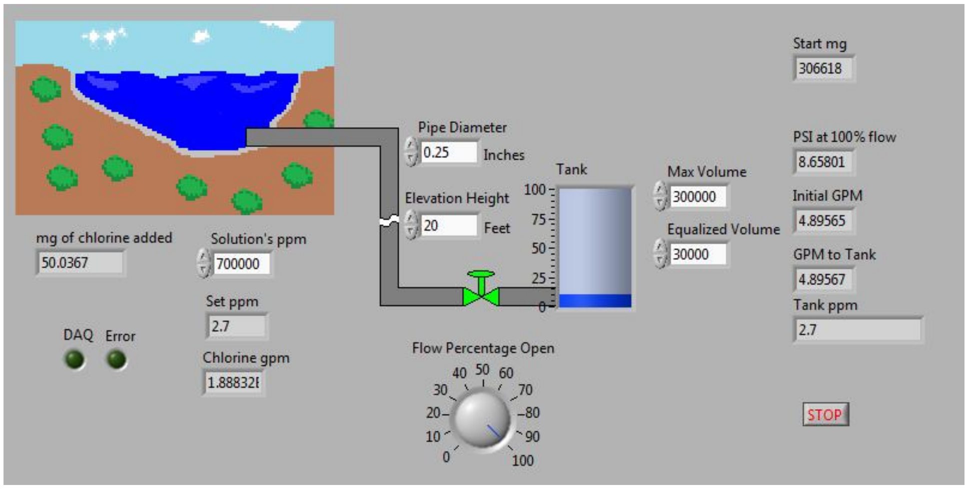
The Water Treatment part was created in LabVIEW 2010^{TM} because it was the best option for communicating with the DAQ. Since the simulation needs to communicate with hardware that has multiple inputs and outputs the DAQ that was chosen is the National Instruments NI USB-6009 which contains 12 digital I/O, 8 Analog inputs, 2 Analog outputs, and can have the digital I/O at 5V or 3.3V power. From the DAQ one analog input and one analog output is used for this part. The simulation uses AO.1 as the `Flow Percentage Open` and AI.7 as the `Set ppm` for the Chlorine ppm. The Water Treatment is only one module. It also has a few variables that need to be set but the simulation does not require them so it will adjust to these when they are entered. The list of variables is:

- `Pipe Diameter` – The diameter of the Pipe into the `Tank` in inches.
- `Elevation Height` – The vertical pipe height from the Reservoir to the `Tank`.
- `Solution's ppm` – The chlorine solution's ppm. The mix known as Hth is the starting ppm.
- `Max Volume` – The maximum volume of the `Tank`.

- Equalized Volume – The volume the tank equalizes at and is assumed throughout the simulation.
- Flow Percentage Open – The percentage of flow allowed through the valve.
- mg of Chlorine added – This is created by the Solution’s ppm and used to calculate the Tank ppm.
- Set ppm – The wanted ppm in the Tank. This is limited to 0 – 12.75 with .05 increments.
- Chlorine gpm – The added gpm due to chlorine.
- Start mg – The initial chlorine in mg in the Tank.
- PSI at 100% flow – The PSI in the pipe at 100% flow
- Initial GPM – The GPM in the tank before the chlorine is added and flow is restricted.
- GPM to Tank – The actual GPM to the Tank after chlorine is added and flow is restricted.
- Tank ppm – The ppm in the Tank. This should be the same as the Set ppm unless the Set ppm was recently changed.
- STOP – This stops the simulation
- Tank – This shows the fullness of the Tank.

The Elevation Height is needed for PSI calculations and Pipe Diameter is needed for flow calculations. Both are assumed to be greater than zero in the simulation. The Max Volume needs to be larger than the Equalize Volume. The Equalize

`Volume` is expected to be larger than the GPM that is coming in or the chlorine in the `Tank` tends to be inaccurate. Figure 7 shows the Treatment piece. The calculations used to display GPM and PSI do not account for resistance in the pipe. They are an estimate for actual flow and pressure rather than expected numbers. The larger the horizontal and vertical length of the pipe the worse the numbers will be.



Start mg 306618

Pipe Diameter 0.25 Inches

PSI at 100% flow 8.65801

Max Volume 300000

Initial GPM 4.89565

Elevation Height 20 Feet

Equalized Volume 30000

GPM to Tank 4.89567

Tank ppm 2.7

mg of chlorine added 50.0367

Solution's ppm 700000

Set ppm 2.7

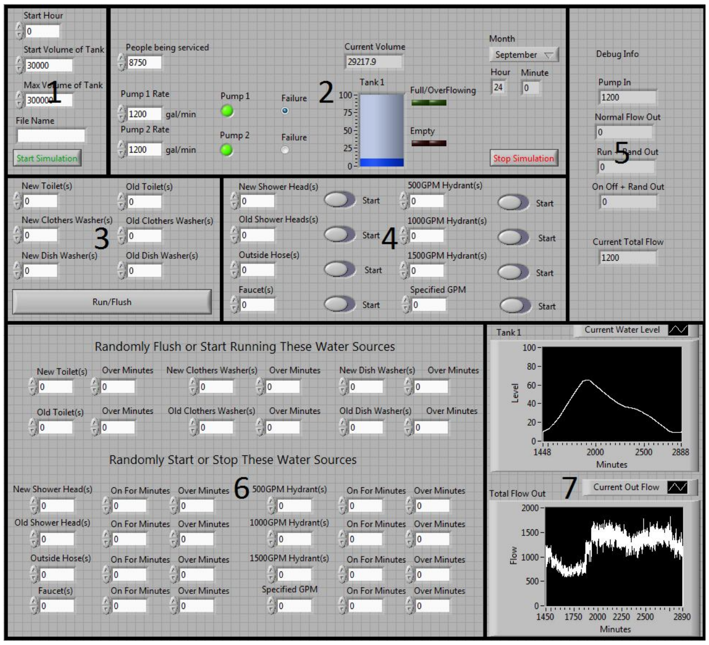
Chlorine gpm 1.88832E

FIGURE 7 – Water Treatment Simulation

C. Creation of Water Distribution Simulation

This part was also created in LabVIEW 2010^{TM} for ease of use with the DAQ. Since the same DAQ is used and there are only two analog outputs total, we are limited to 1 tank level and 12 pumps. The actual simulation only consists of 1 tank and 2 pumps which uses the other analog output and two digital I/O. It uses AO.0 as the tank output to Point 5 on the RTU and P1.0 and P1.1 as inputs to pumps that are to be connected to Point 0 and 1 from the RTU. The simulation is broken into modules. Each module has a specific task and is connected through the main module called `Water`

`Distribution.vi` shown in Figure 8. This contains all the modules and connects them to make the working simulation. This also holds all of the controls that feed into the system and updates all the Indicators that are shown. The Front Panel of this module is what is seen by the user. It consists of 7 different sections.



Start Hour 0
Start Volume of Tank 30000
Max Volume of Tank 30000
File Name

People being serviced 8750
Pump 1 Rate 1200 gal/min
Pump 2 Rate 1200 gal/min
Current Volume 29217.9
Month September
Hour 24
Minute 0

Debug Info
Pump In 1200
Normal Flow Out 0
Run + Rand Out 0
On Off + Rand Out 0
Current Total Flow 1200

New Toilet(s) 0
Old Toilet(s) 0
New Clothers Washer(s) 0
Old Clothers Washer(s) 0
New Dish Washer(s) 0
Old Dish Washer(s) 0

New Shower Head(s) 0
Old Shower Heads(s) 0
Outside Hose(s) 0
Faucet(s) 0
500GPM Hydrant(s) 0
1000GPM Hydrant(s) 0
1500GPM Hydrant(s) 0
Specified GPM 0

Randomly Flush or Start Running These Water Sources

New Toilet(s) 0 Over Minutes 0
New Clothers Washer(s) 0 Over Minutes 0
New Dish Washer(s) 0 Over Minutes 0
Old Toilet(s) 0 Over Minutes 0
Old Clothers Washer(s) 0 Over Minutes 0
Old Dish Washer(s) 0 Over Minutes 0

Randomly Start or Stop These Water Sources

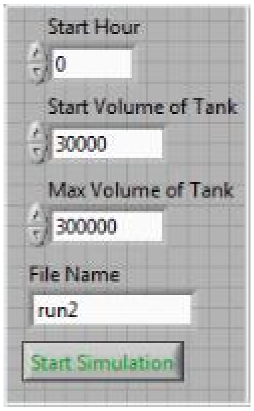
New Shower Head(s) 0 On For Minutes 0 Over Minutes 0
Old Shower Head(s) 0 On For Minutes 0 Over Minutes 0
Outside Hose(s) 0 On For Minutes 0 Over Minutes 0
Faucet(s) 0 On For Minutes 0 Over Minutes 0
500GPM Hydrant(s) 0 On For Minutes 0 Over Minutes 0
1000GPM Hydrant(s) 0 On For Minutes 0 Over Minutes 0
1500GPM Hydrant(s) 0 On For Minutes 0 Over Minutes 0
Specified GPM 0 On For Minutes 0 Over Minutes 0

FIGURE 8 – Simulation Main Module

The first section needs be set before the rest of simulation will run. It consists of:

- `start hour` –The hour of day the simulation should start running at.
- `start volume of tank` – The initial volume of the Tank.
- `max volume of tank` – The total volume of the Tank.
- `Start Simulation` – Pressed to start simulating.
- `File Name` – The name wanted for a saved file of data created.

The `File Name` is only entered if a .csv is desired for the current run. It will save a file in the *LabVIEW Data* folder that is normally created in a user's *Documents* folder on Windows.



Start Hour
0

Start Volume of Tank
30000

Max Volume of Tank
300000

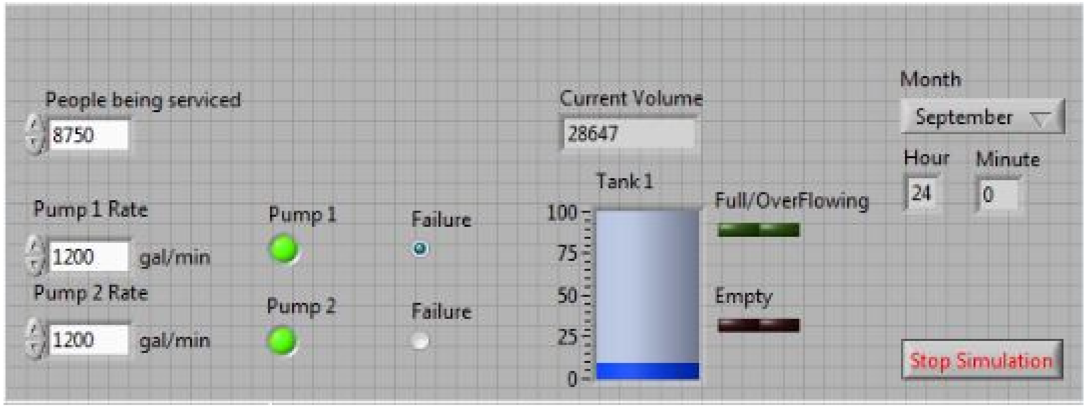
File Name
run2

FIGURE 9 – First Section

The second section is where most of the observations will be done. It holds four variables that effect the simulation. Default values are given but to show accurate data should be set before starting the simulation. These are the first four in the list.

- `People being serviced` – The amount of people being serviced by this distribution plant. This will set up the water being used per minute.
- `Month` – The water usage per month changes so this helps simulate different months more accurately.
- `Pump 1 Rate` – The rate in GPM that Pump 1 is set to.
- `Pump 2 Rate` – The rate in GPM that Pump 2 is set to.
- `Pump 1` – This is an LED indicator that shows if Pump 1 is on or off.
- `Pump 2` – This is an LED indicator that shows if Pump 2 is on or off.
- `Pump 1 Failure` – This will make Pump 1 have a GPM of 0 even if it is on.

- Pump 2 Failure – This will make Pump 2 have a GPM of 0 even if it is on.
- Current Volume – This shows the current Volume in the Tank.
- Tank 1 – This shows a Graphical representation of Tank 1 with its current volume.
- Full/OverFlowing – This indicator starts flashing if Tank 1 is full.
- Empty – This indicator starts flashing if Tank 1 is empty.
- Hour – The water usage per hour changes so this shows the current hour being simulated.
- Minute – The current minute being simulated.
- Stop Simulation – This button will stop the simulation.



People being serviced 8750

Pump 1 Rate 1200 gal/min

Pump 2 Rate 1200 gal/min

Current Volume 28647

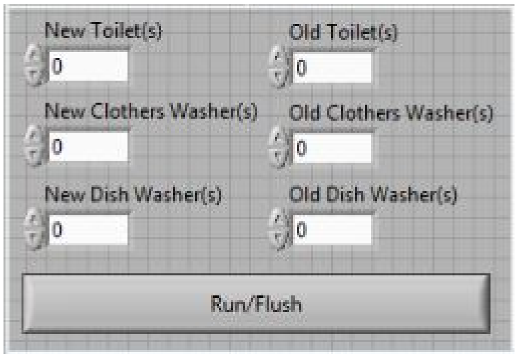
Month September

Hour 24

Minute 0

FIGURE 10 – Second Section

Section three contains all water based appliances that are turned on and automatically turn off or flush. The number of each appliance can be selected. These numbers should be chosen as numbers above normal usage for positive numbers or below normal usage for negative numbers.


New Toilet(s) 0

Old Toilet(s) 0

New Clothers Washer(s) 0

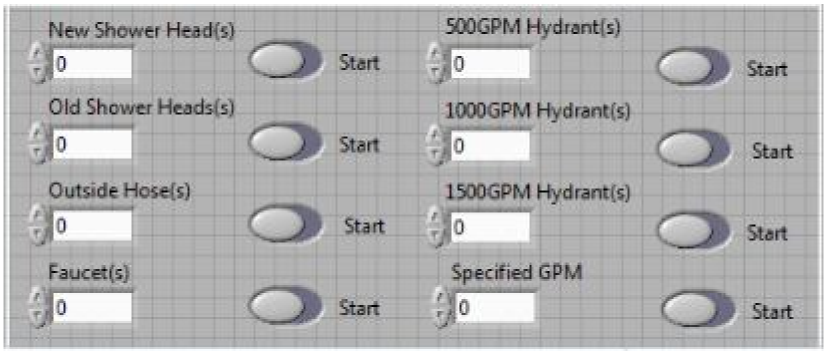
Old Clothers Washer(s) 0

New Dish Washer(s) 0

Old Dish Washer(s) 0

FIGURE 11 – Third Section

Section four is similar to section three in the sense that these are appliances that should be selected as the number above or below normal usage, but different because these are appliances that are usually turned on and turned off manually.


New Shower Head(s) 0

500GPM Hydrant(s) 0

Old Shower Heads(s) 0

1000GPM Hydrant(s) 0

Outside Hose(s) 0

1500GPM Hydrant(s) 0

Faucet(s) 0

Specified GPM 0

FIGURE 12 – Fourth Section

Section five consists of Indicators to make sure each section is operating as expected but is more for debugging when making changes to the code than for the simulation.

- `Pump In` – The total of the two pumps in GPM
- `Normal Flow Out` – The water usage for just the amount of people selected.
- `Run + Rand Out` – The water usage for section three and part of section six.
- `On Off + Rand Out` – The water usage for section four and the other part of six.

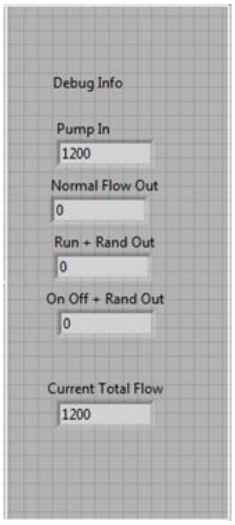
- `Current Total Flow` – This is the sum of IN - OUT using above numbers unless the sum of OUT is negative. If OUT is negative the total of OUT is set to zero since water shouldn't ever flow back into the tank from the water lines.

$$Current\ Total\ Flow = Pump\ In - Flow\ Out - Run\ Out - On\ Off\ Out$$

Using the `Current Total Flow` along with the `Old Tank Level` will get the `New Tank Level`.

$$New\ Tank\ Level = Old\ Tank\ Level\ + Current\ Total\ Flow$$

This is the basic equation used in the Water Distribution Simulation.



Debug Info

Pump In: 1200

Normal Flow Out: 0

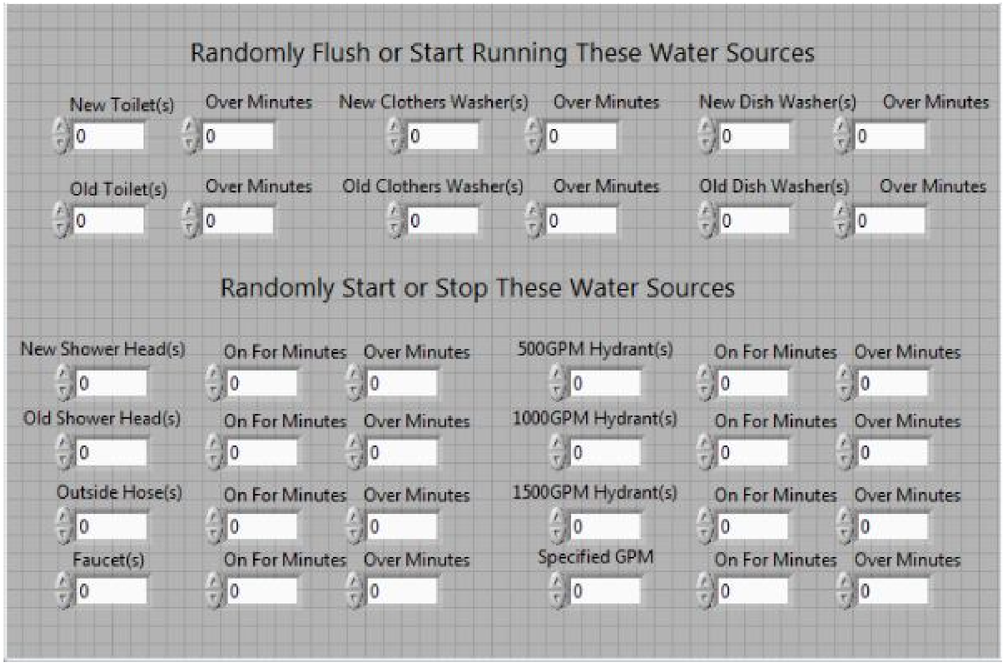
Run + Rand Out: 0

On Off + Rand Out: 0

Current Total Flow: 1200

FIGURE 13 – Fifth Section

Section six is a combination of what is in section three and four but has a random distribution to it. This means that it is possible to say 50 dishwashers should be started across the next 60 minutes or 40 people took 15 minute showers in the past 60 minutes. This is also counted above and beyond the normal usage for a person or with a negative can mean people weren't running those water sources over the given time.



Randomly Flush or Start Running These Water Sources

New Toilet(s)	Over Minutes	New Clothers Washer(s)	Over Minutes	New Dish Washer(s)	Over Minutes
0	0	0	0	0	0
Old Toilet(s)	Over Minutes	Old Clothers Washer(s)	Over Minutes	Old Dish Washer(s)	Over Minutes
0	0	0	0	0	0

Randomly Start or Stop These Water Sources

New Shower Head(s)	On For Minutes	Over Minutes	500GPM Hydrant(s)	On For Minutes	Over Minutes
0	0	0	0	0	0
Old Shower Head(s)	On For Minutes	Over Minutes	1000GPM Hydrant(s)	On For Minutes	Over Minutes
0	0	0	0	0	0
Outside Hose(s)	On For Minutes	Over Minutes	1500GPM Hydrant(s)	On For Minutes	Over Minutes
0	0	0	0	0	0
Faucet(s)	On For Minutes	Over Minutes	Specified GPM	On For Minutes	Over Minutes
0	0	0	0	0	0

FIGURE 14 – Sixth Section

The last section, section seven is made up of two graphs. These show tank level over time and flow out over time. They show up to a whole day's worth of data and scroll once that buffer has been reached to allow for the most up to date data to be displayed. Other features of this main module include the actual file saving and DAQ connection checking. If the file exists, it will add "Another Run" in the row it starts from and append to the current file. If it doesn't exist yet, it will create it and label the two columns Tank Level and Flow Out. If the DAQ is not connected, or connected incorrectly a message will be displayed that asks for the user to make sure the DAQ is connected correctly.

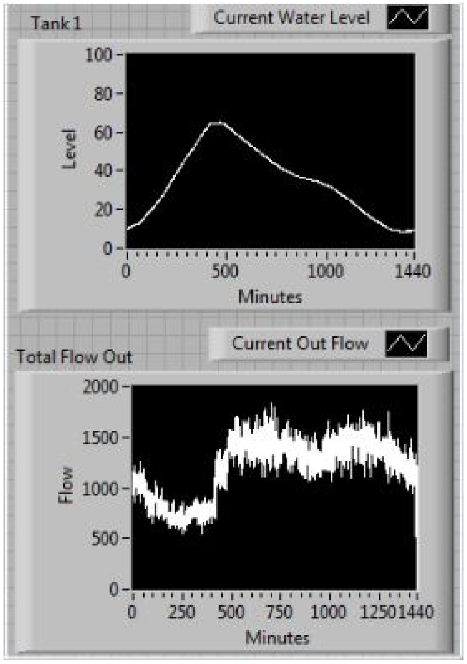


FIGURE 15 – Seventh Section

Since the whole simulation is contained in a timed loop, there needs to be a way to keep track of how long this simulation has actually been running. The Time.vi module uses the count of the timed loop wherever it is placed to keep track of time. It has two indicators, *Minute* and *Hour*. The *Hour* can be set at start up. Each loop count increases *Minute* unless it puts *Minute* to 60. At this point, *Minute* goes to zero and *Hour* is incremented. *Hour* rolls over once it gets to 24 so that it won't ever read larger than 23 and will continue this cycle until the simulation stops running. These numbers are used in OutFlow.vi, StartWater.vi, and help plot the graph.

To effectively model the average Flow Out for many people, many assumptions were made. The module that handled these is called OutFlow.vi. The first step was to obtain about how many gallons per day an average person used. The Louisville Water Company was kind enough to help with this. In an E-mail to the author on February 23, 2011, Beverly Soice gave charts that showed the water usage for both of their distribution plants per day for every month in 2009 and 2010. The data used in this simulation uses the average water usage per month. This was then divided by 780,000 because the Water Companies website states that “Today, Louisville Water Company provides water to about 780,000 people in Louisville Metro …” This means that the number is obviously smaller than that, but the numbers obtained were similar to a variety of other sites that stated about how much water a person uses per day. The average fluctuation for gallons per day was 17 so this variability is represented by Gaussian White Noise with the standard deviation set to 17 which is a built in VI in LabVIEW^{TM}.

TABLE II

DAILY WATER USAGE PER MONTH

Month	Gallons per day
January	144.4615
February	142.0705
March	140.4846
April	146.7513
May	154.9603
June	178.4244
July	187.7744
August	190.9026
September	197.3295
October	178.4154
November	145.0372
December	140.7615

After the amount per day was determined, the usage over the time of day was determined. The only data that was found came from EPCOR and is shown in FIGURE 16 [16]. The distribution for a day in Canada shouldn't differ very much from somewhere in the US. The graph was then broken into height at each hour for both summer and winter. These heights were kept and divided by the sum to get a percentage for each hour across a day. These percentages are shown in Table III. The actual water demand in Figure 16 wasn't used for this simulation because we had more accurate numbers that are shown in Table II.

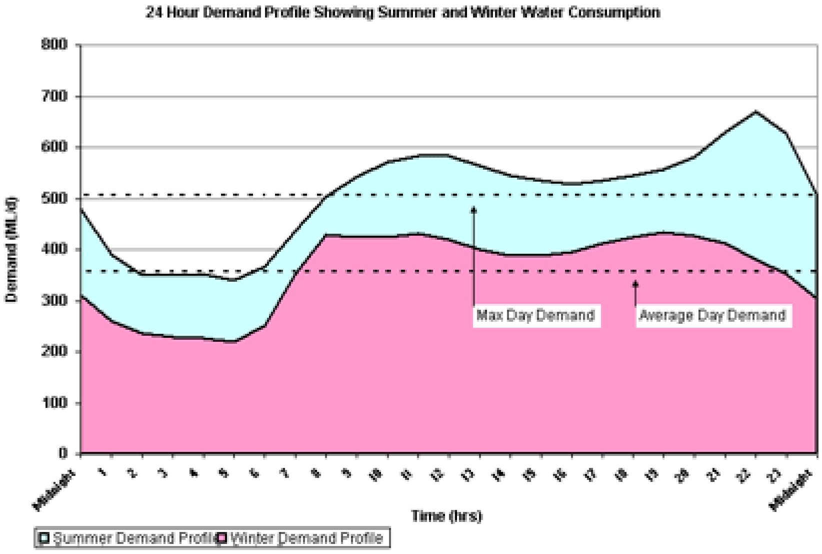


FIGURE 16 – 24 Hour Demand from www.epcor.ca

TABLE III

PERCENTAGE OF WATER USED PER HOUR

Hour	Winter Percentage	Summer Percentage
0	3.63	3.95
1	3.04	3.13
2	2.81	2.88
3	2.46	2.88
4	2.46	2.88
5	2.57	2.72
6	2.69	3.05
7	4.09	3.62
8	5.03	4.12
9	5.03	4.45
10	5.04	4.69
11	5.03	4.77
12	4.91	4.77
13	4.68	4.69
14	4.44	4.53
15	4.44	4.36
16	4.56	4.36
17	4.8	4.44
18	4.91	4.53
19	5.03	4.61
20	4.91	4.77
21	4.91	5.19
22	4.44	5.51
23	4.09	5.1

These numbers were put into the simulation. Using the numbers in Table 2, the summer months became May, June, July, August, September, and October and the winter months became November, December, January, March, and April. By figuring out which month the user wants to simulate, which hour it currently is, and the amount of People being services, the calculations turn out to be

$$People\ Serviced\ *\ (Hour\ Percentage / 100)\ *$$

$$Gallons\ per\ Day\ /\ 60\ minutes\ per\ hour\ =\ Out\ Flow;\ Out\ Flow\ >=\ 0.$$

To allow for variability, four VIs were created, one for constant strains on the system, one for things that create spikes on the system, one for random strains on the system, and one for random spikes on the system. The one that allows for constant strains on the system is called runappliances.vi. This one modules shower heads, outdoor hoses, faucets, and a few different Fire Hydrants. The constant gallons per minute used for this were obtained through a variety of online sources and are summarized in Table IV.

TABLE IV

WATER USAGE IN GPM

Shower head made before 1992	5.5
Shower head made in or after 1992	2.5
Faucet	2.5
Watering Hose	10
Fire Hydrant	1500, 1000, 500

The total for this is called Total Usage and is the sum of the amount of each that is turned on times its respective GPM. If the unit isn't turned on the GPM is considered zero. The module that creates spikes on the system is called onoffappliances.vi. This models toilets, dish washer, and clothes washers. The constants for these were obtained the same as in Table IV and are summarized in Table V.

TABLE V

NUMBERS FOR STARTWATER.VI

Toilet made before 1992	3.5 gpm
Toilet made in or after 1992	1.6 gpm
Clothes Washer Old	40 gallons per load(a load is 1 hour)
Clothes Washer New	20 gallons per load(a load is 1 hour)
Dishwasher Old	10 gallons per load(a load is 1 hour)
Dishwasher New	7 gallons per load(a load is 1 hour)

To obtain the gallons over an hour, there is an array of length 60 and when the Run/Flush button is pressed with any of the washers selected. It adds 1/60th of the number to each slot in the array. When the Total Usage for the minute is calculated, the toilets times the number of them summed with the number in the array is passed on then the minute for the array is set back to zero. This allows for the gallons to be dispersed across an hour and can be added to so that those systems can be run every minute if needed. The one that adds random strains is called RunRand.vi. This is like runappliances.vi but takes the number of appliances chosen and runs only a portion of them each minute over the duration set. The last one that adds random spikes is called OnOffRand.vi. This is like onoffappliances.vi and does the same thing as RunRand.vi does but for onoffappliances.vi.

The module that is responsible for producing what is referred to above as Pump Out is called Pumps.vi. For each pump, if the actual pump is read ON from the RTU and the failure radio button isn't selected, it simply adds the pump rate specified by the user to Pump Out. If either condition isn't met, the pump rate is set to 0 which produces this:

Pump 1 Rate + Pump 2 Rate = Pump Out. This VI also passes on any errors it might get from the DAQ. Usually this only happens when the DAQ isn’t actually connected so it cannot read from it.

The last parts of the above equation are Old and New Tank Level. These are handled in Tank.vi. This deals with keeping track of the tank level, outputting the tank level to the DAQ, and checking for overflow and empty. The controls that are taken from the user are Max Volume and Start Volume. Start Volume gives Current Volume a starting point for the next iteration. Each iteration uses the previous Tank Volume and one more number to calculate the Current Volume. This number is the result from Pumps.vi – RunningWater.vi – StartWater.vi – OutFlow.vi. This is the number that will determine if the tank level increases, decreases, or stays constant. To display the tank level correctly in the tank indicator the current volume is divided by the max volume and the tank displays 0-100. This value is then converted to 0-3.3V, input values accepted by the RTU, and is sent across the DAQ as an Analog signal for the RTU. The last part in Tank.vi is the error code if the DAQ isn’t connected which is passed through to Water Distribution.vi. If anything arises that would require troubleshooting, there is a section in Appendix IV that should address all problems.

V. TESTING AND EVALUATION

The testing and evaluation was done by checking each part of the simulation individually then checking the simulation as a whole. The parts that were checked were the HMI/MTU, the Water Treatment Simulation, and the Water Distribution Simulation. They were checked on basic functionality, communication protocol, and data accuracy. As a whole simulation the system needs to communicate flawlessly and show accurate data across the system.

A. HMI/MTU Testing

There are a few ways the HMI/MTU was tested. It was checked for RTU communication, message integrity, and GUI interaction. For these tests the RTU has to be connected or packets have to be analyzed and responded to. For testing the RTU was connected through the network and the HMI/MTU was run on a PC on the same network. Initially the HMI/MTU was unable to communicate with the RTU because the RTU was not responding to ARP requests. To make the HMI/MTU talk to the RTU for more than the initial minute after power on, there was one line that was entered into command prompt. This line added the MAC address of the field device to the arp table **arp –s xxx.xxx.xxx.xxx “yy-yy-yy-yy-yy-yy”** where x is a part of the IP address for the RTU and y is part of the MAC address for the RTU. The first important test is to make sure that the protocol used to talk to the RTU is correct. This was done by entering a valid User ID, a valid Key, and asking for reads from a variety of points. There was a small problem in the SHA-256 encoding on the RTU. It was sending over the ASCII response instead of the byte response. This created a very large data packet that was limited to each byte being one of 16 different values. After this was fixed, the HMI/MTU was considered operational. Once the HMI/MTU was operational some use cases were assessed. These are as follows:

1. Which operations are permitted with each User ID (1 - 4)?

TABLE VI

TESTED USER PERMISSIONS DATA

User ID	Read Access	Write Access
1	Digital 1,2,3,4 Analog 1,2	Digital 1,2,3,4 Analog 1,2
2	Digital 1,2,4 Analog 1,2	Digital 1,2,4 Analog 1,2
3	Digital 1,4 Analog 1,2	None
4	Digital 1,4 Analog 1,2	Digital 1,4 Analog 1,2

2. What happens without a valid User ID and without any User ID?

   What happens without a valid Key and without any Key?

TABLE VII

USER ID AND KEY DATA

User ID (None, Valid, Invalid)	Key (None, Valid, Invalid)	Data (yes, no)	Protected Data (yes, no)
Valid	Valid	yes	yes
Valid	Invalid	yes	no
Valid	None	yes	no
Invalid	any	no	no
None	any	no	no

3. Does the polling work correctly?

Yes. This was tested with the default setup and a few others. The polling does operate as specified in that window.

4. What happens if all the polls are set to 0?

With all polls set to 0 the HMI/MTU doesn't update. The digital signals can still be written to or read from if selected though.

5. Do all of the digital buttons update after toggling on the GUI?

Yes they do. The only time they do not is when the read cannot be done.

It should be noted that in use case 2 when there is an invalid key or no key, there is a problem between the HMI/MTU and the RTU. If a protected data point is requested, the challenge response will be invalid which causes the RTU to continually ask for a new challenge response. Because the HMI/MTU keeps getting a challenge request, this causes an infinite loop until the HMI/MTU is exited. Another problem identified happens when the User ID doesn't permit any access to a protected data point but asks to read or write to it anyways. This results in a challenge response even though the request should be ignored immediately. Both of these are now known errors in the RTU code.

These results were expected. The only results that seemed odd were the challenge request's problems but this seems to be on the RTU not the HMI/MTU. This shows that the digital points can be written to and read from through the use of a User ID and Key and that analog points can be read the same way. This can now be used to test the network side of the RTU.

B. Water Treatment Testing

The water treatment simulation part was also run on a PC that had drivers that supported the USB-6009 DAQ module, had LabVIEW 2010^{TM}, and was connected to it. The drivers were not supported in Linux so a Windows PC was used. Assuming this is tested after the water distribution part, the DAQ should be functional. The tests for this are:

1. Does the `Pipe Diameter` affect the `Initial GPM`?
2. Does `Elevation Height` affect `PSI at 100% flow`?
3. Does the `Flow Percentage Open` show in the HMI/MTU?
4. Does the `Flow Percentage Open` affect the `GPM to Tank`?
5. Does the `Tank ppm` eventually equalize after changing the `Set ppm` in the HMI/MTU?
6. Does the `Solution’s ppm` affect the `Chlorine gpm` and `GPM to Tank`?

All of these were tested and the results for all were affirmed. This means that everything in the water treatment part is working correctly. This section does not show anything over time so no graphs were produced.

C. Water Distribution Testing

The water distribution simulation part was run on a PC that had drivers that supported the USB-6009 DAQ module, had LabVIEW 2010^{TM}, and was connected to it. The drivers were not supported in Linux so a Windows PC was used. The first step in testing the simulation is to make sure the DAQ is working. The easiest way to do this is to connect the 2 digital inputs to ground and +5v on the DAQ at different times to make sure the pump lights toggle. This can also be a quick way to tell which pump is which input. The digital inputs on the DAQ supports both 5v and 3.3v connections. There are a few ways to test the analog out. The easiest for this set up is to hook the analog out of the DAQ to the analog in on the RTU and see if tank levels change in the HMI/MTU when it changes in the simulation. The output from the RTU is 3.3v which is important to note because a tank at 100% would read around 3.3v instead of 5v. Another option would be to hook the analog out to an oscilloscope and view the voltage change.

After the DAQ has been confirmed to be in working order, the simulation was tested. This started with a lot of verification that buttons and numerical controls and indicators were using the formulas correctly and outputting likely data. Some of those tests were as follows:

1. Check the fail box and make sure *Pump In* goes to just one pumps output. Test with both pumps.
2. Fill the Tank. Make sure the Full lights flash on and off.
3. Empty the tank. Make sure the Empty lights flash on and off.
4. Keep the tank between full and empty. Make sure the empty and full lights are off.
5. Set 1 and 2 of each appliance individually. Make sure the output of *Run* or *On Off* is correct for that appliance or twice that appliance. Also do this with -1 and -2. For the *On Off* appliances make sure they run the correct amount of time only.
6. Specify a new file name and run the simulation. Make sure the headers are correct then run the simulation again with the same file name. Make sure it appends to the previous file.
7. Check the random distributions and make sure they are correct. Make sure they can handle negative numbers.

There were also a few tests run to make sure the graphs across a day full of trial data looked similar to Figure 16. The curve looked very similar in `Test Flow Out` which was expected and showed that the data was being used correctly. This also showed that the numbers being generated weren't exactly the same which created the randomness that was intended. Run 1 and Run 2 were done with the same initial values. There was one pump on with *GPM* at 1200, a *max volume* at 300000, *people* at 8750, *start volume* at 30000, *start hour* at 0, *month* at Sept. and *name* at run1or run2. These are the graphs that show the Tank Level which should be similar and the total flow out which should look like Figure 16 and should have a few noticeable differences between the two of them. The end tank levels of these runs were 3310.7 for Run 1 and 28647 for Run 2. Since September is a summer month in this simulation it should follow the summer trend. It was chosen because the most water used in a month happened in September.

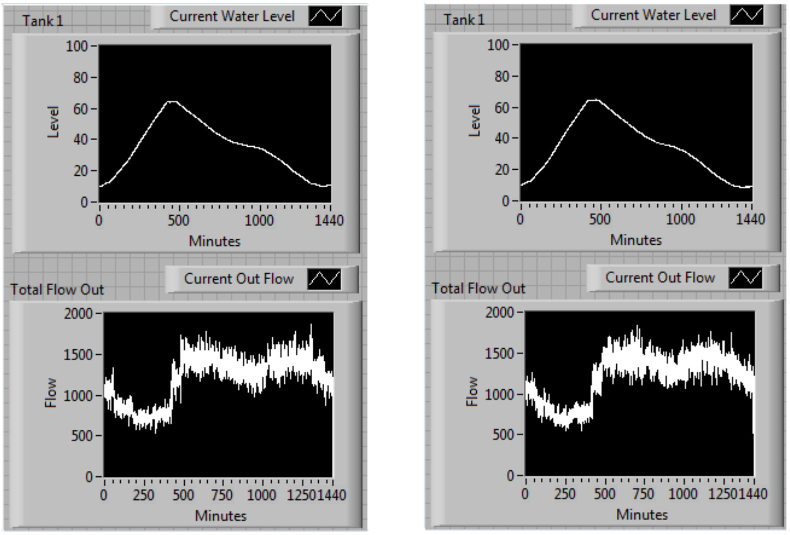


FIGURE 17 – Run 1(left) Run 2(right)

To show the same effect with a winter month March was selected and *name* Run 3 and Run 4 were done with the same initial values as 1 and 2. March had the least water used in a month. The end tank levels of these runs were both full but the graphs show what the water Flow Out should be like and shows some variability. There is also significantly more water used during the summer month with the same starting variables. This also follows the expected output.

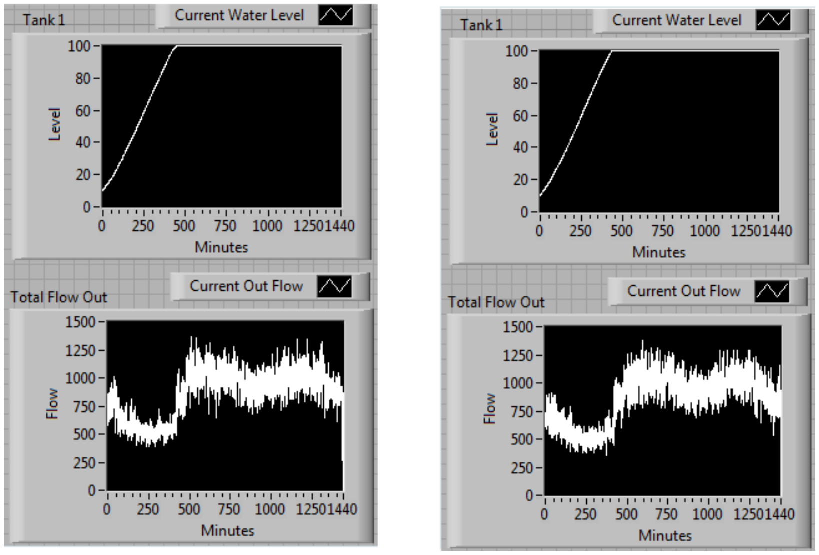


FIGURE 18 – Run 3(left) Run 4(right)

D. Full Simulation Results

After each part has been tested with the RTU, the whole project was ready to be tested. The pumps were turned on/off from the HMI/MTU and affected the pumps in the simulation. The analog value in the simulation was sent through to the HMI/MTU for displaying the tank level. The initial analog values being sent through were actually flipping the high and low bits when the RTU converted the signal to digital which initially caused some weird results. Once this was fixed analog signals being passed through were within a tolerable range to the actual value sent. The only other test that was done was to see how high the *pumps in* had to be with a *max tank level* of 300000 before the simulation at a poll of 1 on just the tank sensor to give the HMI/MTU < 90% full to 100% full as sequential reads. This was done by setting *people* to 0 and turning on just one pump. With the *pumps in* at 24000 about half of the time it did this. Closer to

12000 there was constantly 1 to 2 reads between 90% and 100%. Since the actual fill time is a 1 real second is equal to 1 simulation minute this means that the pump would actually need to be about 60 times the number found in this test to actually have the HMI/MTU not notice this increase. It is also worth noting that in the tests *pumps in* were 1200 so even with both pumps running this would never approach 12000 or 24000. This was also done with no people using water. This is also very unlikely and would make the tank rising be slower due to normal water usage. This shows the boundary conditions of the simulation and that it should be adequate for most applications.

To summarize the results found, for the testing being done with this simulation the limits given are way above the expected values. There shouldn’t be a problem with the HMI/MTU not being updated in time to turn off pumps to prevent overflow due only to delay in the relay of information unless the polling is set too high. The combination of the HMI/MTU and Water System should allow the RTU to be tested enough to be reliable in a real world situation.

VI. CONCLUSIONS AND FUTURE WORK

Many critical infrastructures, including the water sector, use SCADA systems, and it has been established that these systems have serious cyber-security vulnerabilities. The development of SCADA simulations for the purpose of research, evaluation, and testing of new SCADA cyber-security approaches and technology is essential. This thesis has proposed the design and implementation of a simulation for SCADA testing with hardened field devices.

As shown in Chapter Five, the simulation created for testing has already revealed some issues with the current prototype. As the prototype improves, there will be more bugs and hopefully the simulation will help catch these also. Since this simulation was created to help test RTU's, it has already demonstrated its usefulness. It found a protocol mistake where ASCII was being sent instead of binary bytes and a byte mix up where the high and low bits in the analog byte where getting flipped. In the water system parts, there is now a basic design for flow percentage, tank level, set points for chlorine, and pump values. These designs can be reused to create more complex systems in the future because of the modules created for water usage and water operations. If testing an RTU doesn't require a complex system or is in the initial stages of testing, there is a reliable water distribution and treatment system simulation readily available. The water distribution system should be flexible enough to test a variety of situations that range from normal usage to edge cases. The HMI/MTU should be able to work with any RTU that has implemented the challenge response protocol outlined in Table I. This can also be used to test a variety of network attacks and determine how well the RTU handles these attacks.

In the future the simulation could be expanded. The HMI/MTU created was made for a small residential water system. If this is to be scaled to be commercial water system, more components would need to be added. The system was limited by the analog outputs on the DAQ and having only one RTU. To make the water simulation parts larger, an additional DAQ would need to be purchased that contained more outputs to allow for more data to be sent. The previous suggestion would also result in a larger HMI/MTU. Another way to create a larger HMI/MTU would be to get another DAQ and RTU and making them both report to the same HMI/MTU. This would require a small rework because the current HMI/MTU expects just one IP Address. It shouldn't be difficult to allow multiples and link them to parts of the HMI/MTU. There is also a lot of room to improve the HMI/MTU. It currently can monitor sensors and toggle digital controls, but does not do many of an MTU's other tasks. This would result in better testing of the network and response time of the RTU. There was also the suggestion that there be a lower level piece of code that handled the challenge response part so that it could be used under current HMI's.

For both parts that are modeled after water systems there is maintenance needed to keep the simulation close to real life water usage. The water usage per month is currently for 2010 and the Louisville area. As the years pass this number will be further from the actual water usage and, if it isn't changed, will make this simulation outdated.

There could also be water usage for other cities that would need to be selectable in the simulation. If the emerging smart grid technology starts to take off, water usage curves throughout the day are also likely to change to less water usage during the day and more usage during the night. This would require the curve for both summer and winter to be updated. The water treatment part is not very accurate in PSI and GPM in the pipe as the pipe gets longer. There is also an assumption that when chlorine levels change, the tank is evenly mixed each minute. This is not realistic and could be improved with more accurate modeling. The tank level in this simulation could also be added as a value passed to the HMI/MTU. The addition of weather disasters would also be a nice touch to the water system. Lastly water hammer is not accounted for in this simulation. Since the implementation of such a potential problem is likely to happen in a water system it would be beneficial. Simulation of water hammer would require a significant amount of additional simulation components, and its specific relevance to cyber security of SCADA systems should be established before expending the effort to implement it.

BIBLIOGRAPHY

[1] W. J. Ackerman and W. R. Block, “Understanding Supervisory Systems,” *Computer Applications in Power, IEEE*, vol. 5, no. 4, pp. 37-40, October 1992.

[2] S. M. Bailey, A. Marjanski, H. M. Markowitz, and S. V. Rice. The SIMSCRIPT III programming language for modular object-oriented simulation. *Proceedings of the 37th conference on Winter Simulation*, Dec. 4–7, 2005, pp. 621–630.

[3] G. D. Bhatt, J. H. Graham, and S. C. Patel, “Improving the cyber security of SCADA communication networks,” *Communications of the ACM – Barbara Liskov: ACM’s A.M. Turing Award Winner*, vol. 52, no. 7, pp.139-142, July 2009.

[4] T. Brown, “Security in SCADA Systems: how to handle the growing menace to process automation,” *Computing and Control Engineering*, vol. 16, no. 3, pp.42-47, June 2005.

[5] E. Byres, M. Franz, and D. Miller, “The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems,” *International Infrastructure Survivability Workshop (IISW’04)*, Institute of Electrical and Electronics Engineers, Lisbon, December 4, 2004.

[6] F. E. Cellier. Combined continuous/discrete simulation: applications, techniques and tools. *Proceedings of the 18th conference of Winter simulation*, 1986, pp.24-33.

[7] J. Dudley, L. Rietveld, and P. Ross, “TECHNEAU Water Treatment Simulator: Modelling Framework (Version 1.0),” Internet: http://www.techneau.org/fileadmin/files/Publications/Publications/Deliverables/D5.4.4.pdf, Sept. 2008, [February 18, 2011].

[8] M. S. Eldred, A. A. Giunta, and S. F. Wojtkiewicz, “Overview of modern design of experiments methods for computational simulations,” *41st AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, AIAA-2003-0649, 6-9 January 2003.

[9] R. B. Findler and M. Flatt, “Modular Object-Oriented Programming with units and Mixins,” *ACM SIGPLAN Notices*, vol. 34, no. 1, pp. 94-104, January 1999.

[10] A. Giani, G. Karsai, T. Roosta, A. Shah, B. Sinopoli, and J. Wiley, "A Testbed for Secure and Robust SCADA Systems," *ACM SIGBED Review – Special issue on the 14th IEEE real-time and embedded technology and applications symposium (RTAS'08) WIP session*, vol. 5, no. 2, Article no. 4, July 2008.

[11] J. Graham and J. Hieb, "Designing Security-Hardened Microkernels for Field Devices," in IFIP International Federation for Information Processing, vol. 290; *Critical Infrastructure Protection II*, eds. Papa, M., Shenio, S., (Boster: Springer), pp. 129-140, 2008.

[12] J. Hu, A. Mahmood, C. Queiroz, Z. Tari, and X. Yu, "Building a SCADA Security Testbed," nss, pp. 357-364, 2009 Third International Conference on Network and System Security, 2009.

[13] "HydrauliCAD: AutoCAD hydraulic analysis software program using Epanet calcs for modeling of water pipe network distribution systems," Internet: http://www.hydraulicad.com/index.html, 1998, [February 18, 2011].

[14] "Hydromantis – Water & Wastewater Treatment Software & Engineering – WatPro," Internet: http://www.hydromantis.com/WatPro.html, 2011, [February 18, 2011].

[15] V. M. Igure, S. A. Laughter, and R. D. Williams, "Security issues in SCADA networks," *Computers and Security*, vol. 25, no. 7, pp. 498-506, May 2006.

[16] "Peak water demand period." Internet: http://www.epcor.ca/en-ca/Customers/water-customers/pressure-and-supply/Pages/PeakWaterDemand.aspx, [May 15, 2011].

[17] "SHA-256 cryptography software." Internet: http://www.cryptosys.net/sha256.html, March 20 2009, [July 13 2011].

APPENDIX I – Glossary

DAQ – data acquisition

GPM – gallons per minute

GUI – graphical user interface

HMI – human machine interface

I/O – input/output

LED – light emitting diode

MTU – master terminal unit

PC – personal computer

PLC – programmable logic controller

PPM – parts per million

PSI – pressure per square inch

RTU – remote terminal unit

SCADA – security control and data acquisition

SHA – secure hash algorithm

TCP/IP – transmission control protocol/ internet protocol

UDP – user datagram protocol

VI – virtual instrument

WPF – windows presentation format

APPENDIX II – Mathematical Equations for Water Treatment Simulation

These are the equations that are used in the Water Treatment Simulation. The variables used are described below.

$$\mathrm{I_{GPM}} = \frac{7.48\ \mathrm{Gal}}{1728\ \mathrm{in}^3} * \frac{32\ \mathrm{ft}}{\mathrm{s}} * \frac{60\ \mathrm{s}}{\mathrm{min}} * \frac{12\ \mathrm{in}}{\mathrm{ft}} * \left(\left(\frac{\mathrm{P_D\ in}}{2}\right)^2 \right) * \pi * \mathrm{F_P}$$

The Gal/in^3 is just a conversion to similar units. The 32ft/s is the velocity in the pipe due to gravity. The next 2 are unit conversions. The rest is getting the area across the pipe by radius squared times pi.

$$\mathrm{C_{mg}} = \mathrm{T_{GPM}} * \frac{3.785411784\ \mathrm{L}}{\mathrm{Gal}} * \mathrm{St_{ppm}}$$

ppm is 1milligram per liter and the unit conversion 3.785 gets gallons into liters. This is actually mg/min but since the math is always done each minute this can be ignored.

$$\mathrm{S_{mg}} = \mathrm{V_E} * \frac{3.785411784\ \mathrm{L}}{\mathrm{Gal}} * \mathrm{St_{ppm}}$$

$$\mathrm{P_{PSI}} = \frac{\mathrm{H_E\ ft}}{2.31}$$

The 2.31 is a constant that seems to be universally accepted for pressure related to gravity. It could be expanded to be more precise.

$$\mathrm{T_{GPM}} = \mathrm{I_{GPM}} + \mathrm{C_{GPM}}$$

$$\mathrm{N_{mg}} = \mathrm{C_{mg}} - \frac{\mathrm{T_{GPM}} * \mathrm{O_{mg}}}{\mathrm{V_E}} + \mathrm{O_{mg}}$$

This adds the mg of chlorine and the old mg in the tank and subtracts the mg leaving the tank.

$$T_{ppm} = \frac{N_{mg}}{V_E * \frac{3.785411784\ L}{Gal}}$$

This uses the new mg content and the current volume of the tank to get ppm.

O_{mg} = initially S_{mg} then last minutes N_{mg}

I_{GPM} = Initial GPM

C_{GPM} = Chlorine GPM

T_{GPM} = Total GPM

C_{mg}= milligrams of Chlorine added per minute

S_{mg}= milligrams of Chlorine at Start

O_{mg}= milligrams of Chlorine Old

N_{mg}= milligrams of Chlorine New

P_D = Pipe Diameter

P_{PSI}=Pipe PSI

F_P = Flow Percentage

H_E = Elevation Height

V_E = Equalized Volume

V_M = Max Volume

St_{ppm}=Set ppm

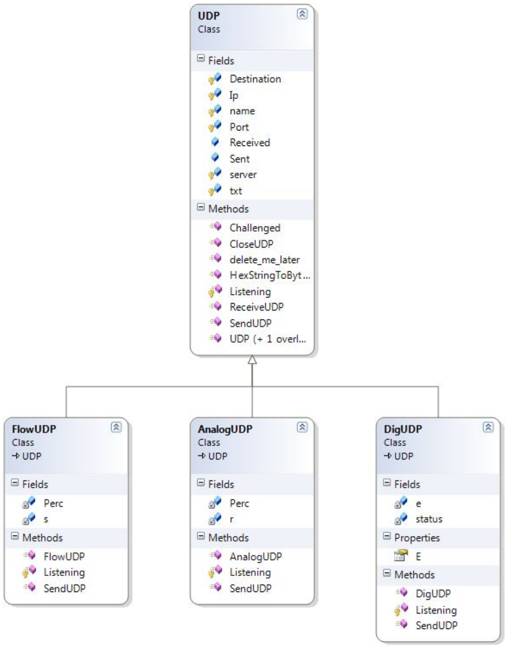
Sn_{ppm}=Solution’s ppm

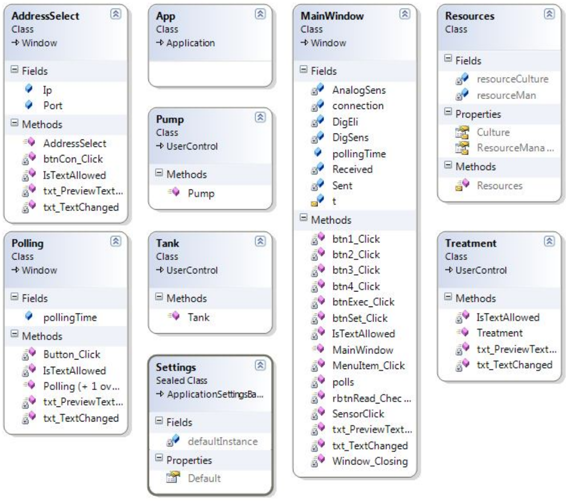
T_{ppm}=Tank’s ppm

ppm=$\frac{1\ mg}{L}$ or parts per million

APPENDIX III – HMI/MTU Class Diagram

This shows the UDP class and what inherits it.





APPENDIX IV – Simulation User Manual

User Manual for all parts of the Simulation

Front Side of HMI

Initial Setup

The computer this is on needs a local network connection to the RTU.

In most cases an arp entry also needs to be put in the computers arp table.

On XP machines this is done by going into the command prompt and entering **arp –s 192.168.1.77 "00-15-c9-13-80-50"** where the IP address is the first set of numbers and the mac address is in "" of the RTU.

On Vista and 7 computers arp –s causes an error. **netsh –c "interface ipv4"** followed by **set neighbors "Local Area Connection" "192.168.1.77" "00-15-c9-13-80-50"** should add the entry to the arp table. This assumes a wired connection. If it is not the wired connection **Local Area Connection** will need to change to the name of that connection. The IP address and MAC address should be changed if the original prototype isn't being used.

Either way to check if it is entered, enter **arp –a** and if the entry took, it should be in this table. In XP I have seen the arp entry go invalid after some time. If the program stops communicating with the RTU and the RTU has been reset recently it might be a good idea to check the arp table again.

Running the program

Enter the IP Address and port number of the RTU is appropriate boxes. The default is the address of the RTU at creation of this document.

Press connect and a second window should open.

- To change background polling of sensors displayed in GUI form.
    - Click on the Polling Menu.
    - Choose the sensor that is desired
    - Simply change the number to a larger one for less frequent polling and a smaller one for polling to increase. Setting a frequency to 0 will cancel polling of that sensor.
- To set the ppm value
    - Enter a number from 0 – 12.75 with a .05 increment into the text box.
    - Press set. Assuming the HMI is talking to the RTU it should set the ppm in the Water Treatment simulation.
- To toggle the pumps from on to off or off to on. Press the pump circle button. If it doesn’t switch, make sure the user ID is able to access the point you are trying to toggle. Make sure the HMI is talking to the RTU. Make sure the Key is correct. Try again.

Testing other points

Select the operation that is to be done. If write, select the value that is to be written. Specify the point operated on. Select the Execute button.

For each action, an entry in Sent Packets should appear. In almost all cases a Received Packet should also appear shortly after the action. If **Failed to Receive Response** appears go through the troubleshooting to get the RTU to talk to the HMI.

Troubleshooting

- Check the arp table in the command prompt. **arp –a** make sure the entry is still there and valid.
- Make sure the IP address and MAC address are correct in the arp table.
- Make sure both the computer with the HMI and the RTU are connected to the network.
- Make sure the IP address and Port number are entered correctly in the HMI.
- Restart the RTU. –make sure the RTU can talk to the Linux box over the network to get the flash information at restart.

Code Structure of HMI

Class descriptions for help with changing later:

- AddressSelect.xaml – the GUI front for entering the IP Address and Port Number
- AddressSelect.xaml.cs – contains the code that limits the actual inputs allowed for IP Address and Port Number
- AnalogUDP.cs – This class inherited UDP.cs. It handles GUI Tank level information
- DigUDP.cs – This class inherited UDP.cs. It handles GUI Pump information.
- FlowUDP.cs – This class inherited UDP.cs. It handles GUI Flow information.

- MainWindow.xaml – This is the main GUI. It contains all the UserControls and authentication. If there is something that should be added that needs to be monitored then this is where it should be added.
- MainWindow.xaml.cs – This is the main class. This is where almost everything is handled and if there is any question as to where something is, it is probably handled or passes through here. **MainWindow** sets up all the GUI elements and UDP information. It also makes all the button clicks functional through RoutedEventHandlers. The sensor numbers for executing operations are currently dependent on where they are in the arrays *DigSens* and *AnalogSens.* **btnSet_Click** handles the setting of the ppm assuming there is a valid number. If the point on the RTU changes for this it should be changed in here. **btnExec_Click** handles the testing values. This is in the top right of the GUI. If anything about this needs to be changed in code it should be done here. **SensorClick** handles the changing of Pump colors and the sending of read and write responses for the GUI Pumps.
- Polling.xaml – This is the GUI for inputting polling frequencies. If a sensor is added or needs to change names, it should be done here. **polls** handles polling and has two instances of *Thread.Sleep(1000);* to create some down time between polls but can be changed if needed for timing constraints.
- Polling.xaml.cs – This isn't where the polling code is. The polling code is in MainWindow.xaml.cs under **polls**. This handles how many sensors are in the polling list and what polling values are valid.
- Pump.xaml – This is the GUI form of a pump.
- Pump.xaml.cs – This only initializes the pump.
- Tank.xaml – This is the GUI form of a tank.
- Tank.xaml.cs – This only initializes the tank.
- Treatment.xaml – This is the GUI form of Flow Percentage and Set ppm.
- Treatment.xaml.cs – This hands Last ppm and makes sure a number is entered in the Set ppm box.

UDP.cs – This handles the UDP setup. It has a basic **SendUDP** setup that is used for testing and is the basis for the other three *UDP.cs classes. Listening is the same way. It has basic functionality but for more specific handling it is required to be created if it is inherited. It also handles timeout problems and Challenge Responses.

Front Side of Water Treatment

Running the Program

- The Elevation Height needs to be specified for PSI to show up.
- The Pipe Diameter needs to be specified for the rest to work.
- The Equalized Volume should be more than the GPM to Tank or odd numbers will appear in Tank ppm.
- The Max Volume should be larger than the Equalized Volume.
- The Solution's ppm should be greater than 0 but less than a million. The default value is that of Hth a common pool chemical with 70% chlorine.
- To change the Set ppm the DAQ needs to be connected to the RTU and the HMI has to be operational. It is changed in the HMI and passed in the water treatment.
- To change the Flow Percentage Open simply turn the knob to the percentage wanted. If a GPM is preferred, turn the knob and check with the GPM to Tank to get the correct percentage.

Troubleshooting

If there is an error there is a good chance the DAQ isn't connected. Make sure it is plugged in. The drivers for the DAQ might also be needed. The DAQ drivers aren't supported on Ubuntu. The drivers are in a folder on the Windows 7 Machine under the user Research's desktop.

Water Treatment VI

This one is significantly simpler than Water Distribution since it is all in one VI. Most of the formulas can be viewed as actual Formula VI components. The 3.785411784 seen all over is the number Gal/L that is used in a few conversions. This one doesn't contain a time function but it is assumed that each loop iteration is a minute.

Front Side of Water Distribution

Running the Program

- Set the Start Hour
- Set the Start Volume of Tank
- Set the Max Volume of Tank
- Set the File Name (optional)
- Set the People being serviced (can be changed during run time)
- Set the Month (can be changed during run time)
- Set the Pump 1 Rate (can be changed during run time)
- Set the Pump 2 Rate (can be changed during run time)
- Click the Start Simulation button

To customize the simulation (this can be done before clicking Start Simulation or after) The pumps can be made to fail at any point during the simulation by simply checking the box that says *Failure* next to the pump that is to fail. Anything below the top two sections should be considered custom. The very bottom section is Random which means the actual amount of the appliance is specified and the time over which the appliance is to be run is specified but the actual times that the appliances turn on or run is up to the system. This can add or subtract usage to help add more variability. The middle sections are instantaneous. These means that once the number is specified and the button is pressed, the appliances run, flush, or start running. If the appliances start running, the user will need to also stop the running when appropriate.

Troubleshooting

If there is an error there is a good chance the DAQ isn’t connected. Make sure it is plugged in. The drivers for the DAQ might also be needed. The DAQ drivers aren’t supported on Ubuntu. The drivers are in a folder on the Windows 7 Machine under the user Research’s desktop.

Water Distribution VI

The main project is called WaterSim.lvproj. This should allow you to build if needed and to see all the VIs used. The main one inside here is *Water Distribution.vi*. If the block diagram of it is looked at the connections to each sub VI should be visible. The most confusing part of LabVIEW^{TM} is that all I/O needs to be on the top layer and is passed through all sub VI. If a change is made that effects the I/O of a single VI the main VI will need to update the interface. This requires a right click. Not very tricky but it can stop a program from working. As a side note, *runappliances.vi* and *onoffappliances.vi* are both used in their random equivalents so anything changed in them will have to be updated in the random Vis.

VI descriptions

- onoffappliances.vi – This handles the values given to Shower Head(s), Outside Hose(s), Faucet(s), Hydrant(s), and Specified GPM. It sums them up and outputs the water usage.
- OnOffRand.vi – This handles the same appliances as onoffappliances.vi but adds *On For Minutes* and *Over Minutes* to handle random usage.
- OutFlow.vi – This handles the amount of water all people would be using. This relies on Month, Hour, and People being serviced
- Pumps.vi – This handles the pump LEDs, failures, and uses Pump Rates.
- runappliances.vi – This handles values given to Toilet(s) and Dish and Clothes Washer(s). It sums them up and outputs the water usage.
- RunRand.vi – This handles the same appliances as runappliances.vi but adds *Over Minutes* to handle random usage.
- Tank.vi – This handles the tank level and sets off the warning lights if the tank is empty or overflowing
- Time.vi – This displays the minute and hour of the current simulation time. It is passed to OutFlow.vi to help with the water usage.
- Water Distribution.vi – This is the main VI that holds the previous ones. It is what the user sees.

APPENDIX V – Sample Output

This is a file created by the Water Distribution Simulation. This specific one is run1.csv that was created for Testing and Evaluation.

Tank Level	Flow Out
10.024	1128.36
10.062	1086.421
10.129	998.866
10.152	1130.475
10.231	964.312
10.267	1091.299
10.286	1142.461
10.347	1015.912
10.393	1062.133
10.39	1209.578
10.395	1186.544
10.397	1193.327
10.439	1073.228
10.526	940.473
10.577	1045.967
10.597	1140.281
10.67	981.701
10.747	969.036
10.804	1028.693
10.857	1038.538
10.893	1094.313
11.018	822.733
11.078	1020.431
11.134	1034.057
11.191	1028.714
11.202	1164.868
11.234	1106.709
11.289	1033.339
11.361	982.828
11.447	943.813
11.451	1187.029
11.472	1137.774
11.484	1162.749
11.53	1064.261
11.602	983.309
11.647	1065.648
11.774	816.806
11.794	1141.669
11.854	1019.073
11.958	887.193
12.033	977.362
12.055	1132.841
12.041	1243.327
12.09	1051.242
12.146	1031.581
12.191	1066.335
12.249	1025.634
12.293	1068.469
12.326	1101.183
12.389	1009.886
12.463	978.512
12.492	1112.606
12.555	1011.214
12.592	1089.383
12.603	1166.038
12.558	1336.007
12.543	1244.086
12.567	1129.615
12.674	878.528
12.766	922.948
12.905	782.253
12.998	920.699
13.077	965.694
13.214	788.153
13.306	923.773
13.395	933.611
13.48	945.166
13.604	827.365
13.741	788.193
13.829	935.657
13.931	895.335
14.088	728.827
14.206	844.34
14.311	885.542
14.418	880.722
14.532	858.343
14.619	936.472
14.729	869.886
14.854	825.014
14.955	898.026
15.085	808.816
15.182	910.312