# Enhancing a Virtual SCADA Laboratory Using Simulink

Zach Thornton, Thomas Morris

## To cite this version:

Zach Thornton, Thomas Morris. Enhancing a Virtual SCADA Laboratory Using Simulink. 9th International Conference on Critical Infrastructure Protection (ICCIP), Mar 2015, Arlington, VA, United States. pp.119-133, 10.1007/978-3-319-26567-4_8 . hal-01431017

**HAL Id: hal-01431017**
**https://inria.hal.science/hal-01431017**

Submitted on 10 Jan 2017

Chapter 8

# ENHANCING A VIRTUAL SCADA LABORATORY USING SIMULINK

Zach Thornton and Thomas Morris

**Abstract** This chapter describes a virtual supervisory control and data acquisition (SCADA) security laboratory and the improvements made using Simulink. The laboratory was initially constructed using virtual devices written in Python that simulate industrial processes, emulate control system ladder logic functionality and utilize control system communications protocols. However, given the limitations of Python programs with regard to modeling industrial processes, an improved model was constructed using the Simulink modeling environment. Custom and commercially-available human-machine interfaces used in real-world SCADA environments were deployed in the new laboratory. In addition, various attacks were developed and implemented against the virtual SCADA system. The behavior of the improved laboratory and its earlier version are compared against the physical system after which both were modeled.

**Keywords:** SCADA laboratory, virtualization, Simulink

## 1. Introduction

Industrial control systems, including supervisory control and data acquisition (SCADA) systems, are essential components of critical infrastructure assets such as electric power grids, oil and gas pipelines, chemical processing facilities, water treatment and supply systems, and transportation systems. Cyber threats – and actual attacks – targeting SCADA systems are well documented, making it imperative to design and implement sophisticated defensive techniques and countermeasures [10].

Fundamental risks in SCADA systems can be identified by researching attack patterns, attack vectors and attack impact. Traditionally, these research efforts have engaged testbeds that incorporate scaled physical models and the accompanying hardware, software and information and communications technologies to realize complete cyber-physical systems. However, these testbeds present two limitations to researchers. First, only researchers with hands-on access to a testbed can engage in SCADA intrusion studies. Second, a testbed is typically expensive, difficult to expand and difficult to maintain.

A virtual SCADA laboratory was developed to help address these limitations. The laboratory was designed to be portable, distributable and expandable. It closely models and communicates with commercial SCADA products, and executes in a virtual computing environment. However, the physical process model, initially a curve-fit of a laboratory process, did not adequately capture the real-world process. This chapter describes the original laboratory as well as recent improvements that employ Simulink to model physical processes.

This research has two main contributions. The first is the detailed workings of a virtual laboratory that is designed for SCADA security research. The second contribution is the integration of Simulink in the testbed to model physical processes with high fidelity.

## 2. Related Work

Most modern industrial processes in sectors such as electricity, oil and gas, water, transportation and chemicals are controlled by SCADA systems. These systems incorporate numerous components that can be broken down into four major categories. At the lowest level are sensors and actuators. Sensors include meters, gauges, calipers and transmitters; they are transducers that convert physical phenomena into electrical signals. Actuators, such as pumps/compressors, valves and motors, receive control signals and manipulate physical processes; thus, they are controllers themselves. The second level comprises distributed controllers, which include programmable logic controllers (PLCs), programmable automation controllers (PACs) and intelligent electronic devices (IEDs). Special-purpose computer systems interface with sensors, implement custom control logic and control actuators based on the control logic and system state. Distributed controllers also include network communications interfaces that connect to upstream systems, including the higher supervisory control layer and human-machine interfaces. The third level is the supervisory control layer, which stores process data, implements system-level control schemes and manages the distributed controllers. At the highest level reside the human-machine interfaces that enable human operators to monitor and control physical processes.

Morris et al. [9] describe a typical small-scale research environment that uses commercial-available equipment to model industrial processes and control systems. The laboratory testbed comprises seven systems: a gas pipeline, storage tank, water tower, industrial blower, assembly line conveyor, steel rolling process and chemical mixing system. Two of the physical systems, the gas pipeline and storage tank, were used as the basis for the virtual systems described in this chapter. Figure 1 shows images of the physical systems and human-machine interface screens in the laboratory.

The Idaho National Laboratory (INL) National SCADA Testbed is a research facility that is designed to evaluate control systems representative of

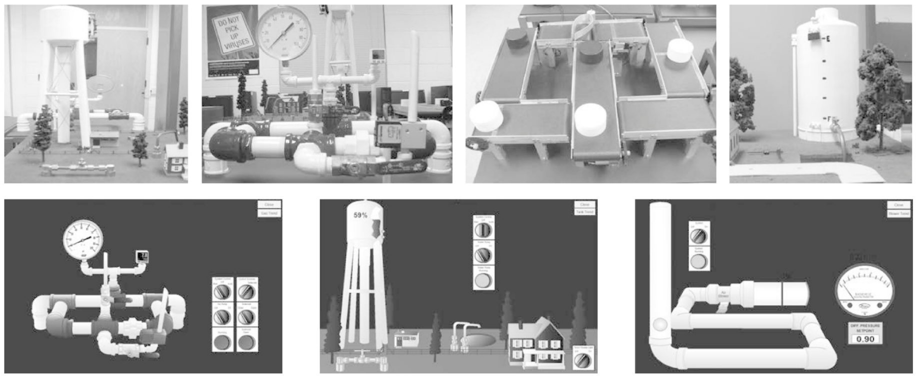

*Figure 1.* Laboratory systems and human-machine interfaces.

those used in the critical energy infrastructure [6]. Idaho National Laboratory's 890-square-mile Critical Infrastructure Test Range is designed to accurately model physical systems. The test range incorporates industrial-scale sensors, actuators, automation systems, human-machine interfaces and more.

Sandia National Laboratory's National Supervisory Control and Data Acquisition Testbed is sponsored by the Department of Energy's Office of Electricity Delivery and Energy Reliability. Like the Idaho National Laboratory facilities, the Sandia testbed is designed to research the effects of cyber attacks. The testbed incorporates a real power generation system and power loads to analyze the impacts of cyber attacks on power grid components. Current activities include applying autonomous agents in SCADA systems, creating novel cryptographic security mechanisms and conducting system assessments and red-teaming [12].

Hahn et al. [4] describe an experimental testbed that models two electric substations connected to a control center. The testbed incorporates commercial human-machine interfaces, a soft remote terminal unit and physical overprotection relays and autotransformers [4].

While the four laboratory testbeds described above have numerous research applications, they are not easily portable, distributable or expandable. This significantly limits the number of individuals or groups that could use these testbeds for education, training and research activities.

Genge et al. [3] have proposed a framework based on Emulab and Simulink to recreate cyber components and physical processes when conducting security analyses of industrial control networks. The underlying architecture strikes a balance between experiments that consist entirely of physical components as in [2] and experiments that are based entirely on simulated components. Genge and colleagues also list several functionalities required to conduct cyber-physical experiments. The list includes supporting a wide range of physical processes, real malware and SCADA software, high fidelity cyber/physical layers and typical industrial control network components. However, the SCADA experimentation framework and architecture are notional because they have not yet been implemented.

Mahoney and Gandhi [7] describe SCADASim, a framework for industrial control system simulation. SCADASiM simulates legacy SCADA systems in order to facilitate regulatory compliance monitoring. It allows for the rapid recreation of messages between cyber and physical systems so that the messages can be analyzed for regulatory compliance. While Mahoney and Gandhi refer to a control system simulation of a water supply system, their paper focuses on monitoring communications for compliance purposes, not on industrial control system simulation.

Reeves and Morris [11] describe a virtual testbed for industrial control system security. The testbed, which is written in Python and incorporates a process simulator and a programmable logic controller emulator, is designed to be interoperable with commercial industrial control equipment. The testbed implements realistic communications protocols and closely approximates programmable logic controller programming functionality and the physical behavior of the systems described in [9]. The testbed of Reeves and Morris served as the foundation of the work described in this chapter.

Simulators for programmable logic controllers and physical processes are available from Rockwell Automation, Mathworks Simulink, MHJ Software, Modellica and other vendors. These simulators have good modeling capabilities in their respective fields. However, they are not designed for SCADA system modeling and would require custom tools, such as those described in this chapter, to create a comprehensive industrial control system testbed.

This chapter describes a laboratory testbed that combines the benefits of a physical system and simulators. It closely models the behavior of real systems and is easily expandable; although it is designed for SCADA system modeling, it can incorporate other simulators. The laboratory testbed also facilitates the integration of open-source tools and commercial software, simplifying future enhancements.

## 3. SCADA Laboratory Overview

The virtual laboratory described in this chapter incorporates three of the four components described in the previous section: (i) a gas pipeline simulation with sensors and actuators; (ii) a programmable logic controller simulation; and (iii) a human-machine interface. Also included are attack generation and attack detection systems. Each laboratory component is run on a separate virtual machine (VM). This makes maintenance of the systems much easier – when errors occur, the system can easily be restored to its previous working state. It also means that a virtual network can be utilized across the virtual machines. Thus, all traffic between the virtual machines is real network traffic that can be logged, disrupted and modified as in a physical system.

## 3.1 Process Simulation

Physical process simulation is the fundamental component of the laboratory. In a real-world environment, the process is typically a physical/chemical/mechanical system that has to be measured and controlled. The process is usually described in terms of high-order differential equations.

In the case of the gas pipeline simulation, the components modeled included a gas compressor and a solenoid release valve. Four scenarios were considered for pressure changes to the system:

**1.** If the gas compressor is operating and the valve is closed, then the pressure will rise indefinitely.

**2.** If the gas compressor is not operating and the valve is open, then the pressure will fall until it reaches zero.

**3.** If the gas compressor is operating and the valve is open, then the pressure will rise to an equilibrium pressure.

**4.** If the gas compressor is not operating and the valve is closed, then the pressure will remain constant.

The simulations employ a curve-fitted model described by Morris et al. [9]. The first two scenarios are modeled using the quadratic equations:

$$p = 2.0052t^2 \tag{1}$$

$$p = 0.098t^2 - 4.439t + 49.83 \tag{2}$$

where $t$ is the time and $p$ is the pressure.

The third scenario is modeled using a piecewise model such that the pressure converges to an equilibrium value of about 7.8 psi. This value was discovered empirically for the system described in [2]. The following equation describes the response as the pressure rises to 7.8 psi:

$$p = -0.0210857t^2 + 0.77319t + 0.151637 \tag{3}$$

The following equation describes the response after the pressure exceeds 7.8 psi:

$$p = 7.3 + \texttt{rand}(-0.02, 0.01) \tag{4}$$

where `rand` is a Python function that generates a uniform random number between the values –0.02 and 0.01.

In the fourth scenario, the pressure is constant.

These equations are implemented in a Python script and are executed as a separate process that runs continuously. The process receives updated commands for the actuators from the virtual controller and sends updates about the current state to the controller.

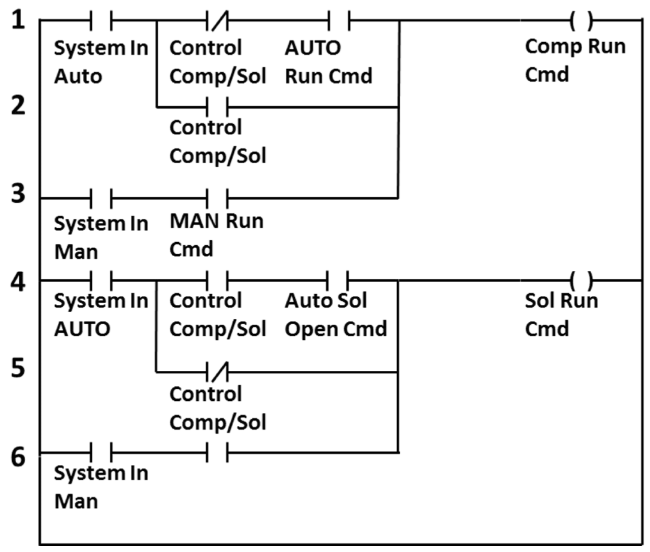

*Figure 2.* Ladder logic program.

## 3.2 Programmable Logic Controller Simulation

A central part of the laboratory is the simulation of programmable logic controller hardware and software. In a real-world system, a typical programmable logic controller is programmed to perform four steps in an infinite loop: read inputs, analyze current state, calculate responses and write outputs. This process is captured by the controller simulation. The responses are written to a set of output registers, which are converted to analog signals and sent to actuators.

Almost all programmable logic controllers are programmed using ladder logic. The programming paradigm is so named because each individual program resembles a ladder with one or more rungs. Figure 2 shows a ladder logic program used in the SCADA laboratory [9].

The virtual programmable logic controller devices (VDEVs) in the laboratory simulate the ladder logic programs in [2]. The virtual programmable logic controller devices emulate the gas pipeline process as closely as possible within the confines of Python programming. Each data read, calculation and output setting takes place sequentially, corresponding to each successive ladder logic rung (top to bottom). Figure 3 shows the emulation of the ladder logic program in Figure 2.

The virtual programmable logic controller devices communicate with the process simulator by emulating the analog and digital communications received from the sensors and actuators. The responses are calculated and sent back to the process simulator. The JavaScript Object Notation (JSON) is used to interface with the process simulator. This was chosen for reasons of simplicity and debugging purposes [11].

```
## Ladder Logic: Set compressor and
## solenoid open or closed
Points['CompRunCmd'].set(False)
if points['SystemInAUTO'].get():
  if (points['AUTOCompRunCmd']get() and
      not points['ControlComp/Sol'].get()):
    points['CompRunCmd'].set(True)
  elif points['ControlComp/Sol']get():
    points['CompRunCmd'].set(True)
if (points['SystemInMAN'].get() and
    points['ManCompRunCmd'].get()):
  points['CompRunCmd'].set(True)

points['SolOpenCmd'].set(False)
if points['SystemInAUTO'].get():
  if (points['AUTOSolOpenCmd'].get() and
  points['ControlComp/Sol'].get():
    points['SolOpenCmd'].set(True)
  elif not points['ControlComp/Sol'].get():
    points['SolOpenCmd'].set(True)
if (points['SystemInMAN'].get() and
points['MANSolOpenCmd'].get()):
  points['SolOpenCmd'].set(True)
```

*Figure 3.* Ladder logic emulation.

The virtual programmable logic controller devices communicate with other virtual devices via Modbus/TCP by using the `modbus-tk` Python library. This enables the virtual programmable logic controller devices to communicate with external devices such as physical programmable logic controllers and human-machine interfaces using a standard SCADA communications protocol. It also enables researchers and students to route, view, capture and analyze traffic as in a real SCADA system. The Modbus/TCP traffic in the simulation is indistinguishable from Modbus traffic in real SCADA devices. This was demonstrated by using the Snort intrusion detection system to capture all the Modbus traffic between a virtual programmable logic controller device and an external SCADA device such as a human-machine interface. Because Snort has pre-configured rules for detecting various communications protocols, including Modbus/TCP, an instance of Snort was executed and configured to capture all Modbus/TCP traffic.

Figure 4 shows an entry in the Snort system log. The first line shows that an event was logged according to the Modbus/TCP response detected rule. The time, date, IP addresses and TCP information pertaining to the Modbus/TCP transaction are also provided. The figure shows that Snort sees the traffic from the virtual programmable logic controller device to the human-machine interface as typical Modbus/TCP traffic.

```
Snort IDS: Modbus TCP - Response Detected [**]
[Classification: Not Suspicious Traffic] [Priority: 1]
06/09-11:08:30.265212 00:0C:29:1E:3B:A6 -> 00:0C:29:90:D1:3F
type:0x800 len 0x4C
10.128.0.1:502 -> 10.128.0.4:50983 TCP TTL:64 TOS:0x0 ID:20393
IpLen:20 DgmLen:62 DF
***AP*** Seq: 0x5C15A47A Win: 0xE3 TcpLen: 32
TCP Options (3) => NOP NOP TS: 36095 435179
```

*Figure 4.* Snort log entry.

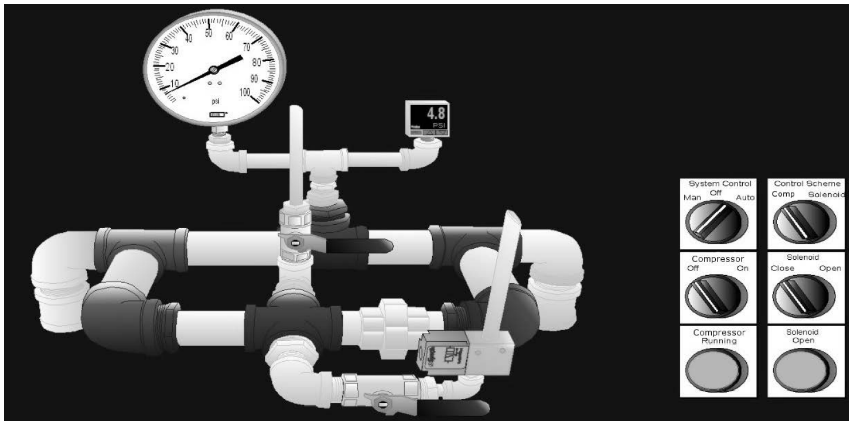

*Figure 5.* GE iFix human-machine interface.

## 3.3 Human-Machine Interface

Human-machine interfaces constitute the third main type of component in the laboratory. A human-machine interface is needed by a human operator of an industrial control system to visualize the changing process and interface with the control system. Two human-machine interfaces are incorporated in the laboratory. The first is a GE iFix human-machine interface, which is identical to the gas pipeline interface described in [9]. Figure 5 shows an image of the gas pipeline human-machine interface screen.

The GE iFix human-machine interface is a widely-used SCADA product. However, because the software is proprietary and not easily distributable, a second human-machine interface was developed using the Python `TkInter` library. Figure 6 shows an image of the Python `TkInter` human-machine interface screen.

The laboratory configuration enables commercial and custom human-machine interfaces to be ported to the laboratory and tested for vulnerabilities. This enables research on human-machine interface vulnerabilities as in [8] to be conducted without the need to access a physical SCADA laboratory.

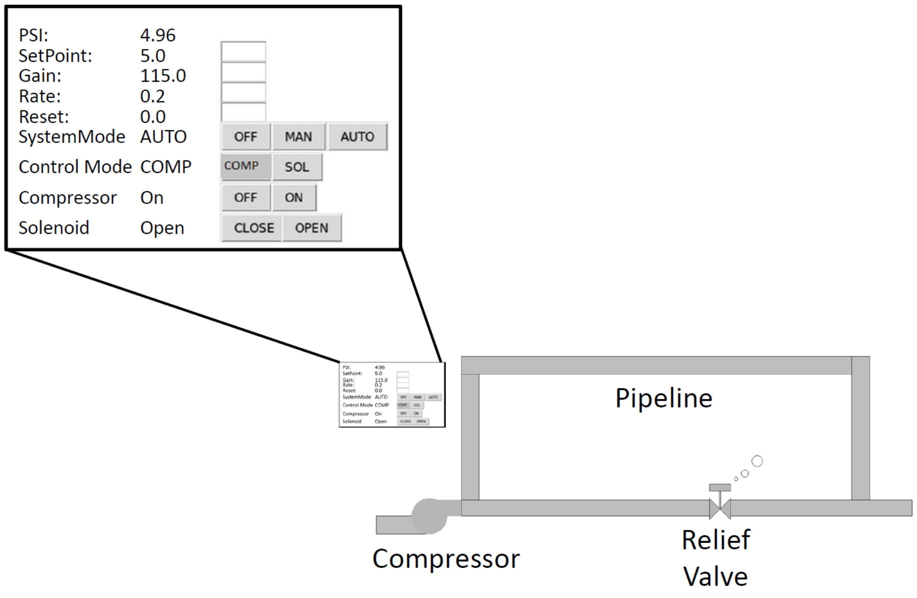

*Figure 6.* Python `TkInter` human-machine interface.

## 4. SCADA Laboratory Enhancements

A real-world process system such as a gas pipeline is typically a complex physical/chemical/mechanical system that has to be measured and controlled. The interrelationships of pressure, temperature, velocity and density of the fluid in the pipeline are approximated by a set of high-order differential equations While these equations are often approximated for computation and simulation, it is clear that the simple quadratic equations presented above are inadequate to study the complex behavior of a pipeline during a cyber attack. Although the differential equations could be programmed using Python and the complex interactions of the fluid with mechanical devices such as compressors and valves could be simulated, native Python is not designed to support such complex mathematics. Also, having to hand code the equations for each new simulation is an unnecessary step, especially if the equations are already available.

Simulink was chosen as the process modeling tool to enhance the existing laboratory due to its popularity as an engineering design and research tool. The SimHydraulics package provided by Simulink offers libraries for modeling and simulating hydraulic systems, including components such as compressors, pumps, valves, actuators and pipelines. Not only does SimHydraulics include pre-configured models of these components, but it also supports the use of custom components via the Simscape language [2].

Initially, a system very similar to the one described in [9] was constructed in Simulink. A simple system comprising a compressor, valve, pipeline and fluid was incorporated. (In the physical system, the fluid is air and its source

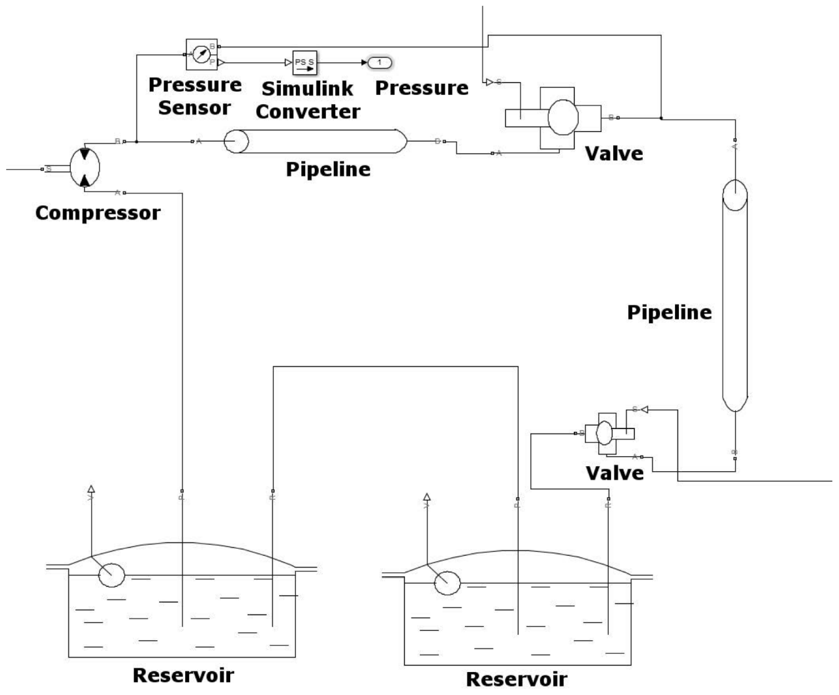

*Figure 7.* Initial Simulink design.

is a compressor; however, because Simulink requires a source and an open air supply is not an option, a reservoir of non-descript fluid was chosen as the fluid source.) A valve was also positioned between the valve controlled by the virtual programmable logic controller device and the return reservoir; this valve simulates the load and changes position at least once a second.

Figure 7 shows the initial design of the pipeline system in Simulink. Not pictured in the figure are the conversions from physical signals to Simulink signals, the angular velocity source for the motor and other Simulink components. Each component was modeled in terms of its properties.

*Table 1.* Hydraulic motor properties.

| | |
|---|---|
| Motor Displacement | 5 in$^3$/rev |
| Volumetric Efficiency | 0.92 |
| Total Efficiency | 0.8 |
| Nominal Pressure | 100 psi |
| Nominal Angular Velocity | 188 rpm |
| Nominal Kinematic Viscosity | 18 cSt |

As an example, Table 1 lists the properties of the hydraulic motor. The motor displacement in the table indicates the volume of fluid displaced per revolution. The volumetric efficiency is the percentage of fluid that flows out of the compressor. The total efficiency is the volumetric efficiency taking into account the mechanical efficiency of the compressor. The nominal pressure, nominal angular velocity and nominal kinematic viscosity are the expected operating conditions of the compressor.

Because the states of the compressor and valve are controlled by virtual programmable logic controller devices, the Simulink model must communicate with the virtual programmable logic controller devices. Simulink also contains libraries for communicating via UDP packets.

Rate limiters and zero-order holds ensure that the UDP transmission rate and the data transmission rate in the actual pipeline are compatible. The ASCII value conversions transform the commands received from the virtual programmable logic controller devices to binary values for opening/closing the valve and turning the compressor on/off. Each of the three values (compressor, valve and pressure) from the simulation is sent to a different UDP port and each command from a virtual programmable logic controller device is received on a different UDP port. As discussed earlier, the virtual programmable logic controller devices communicate with the process simulator by sending JSON attribute-value pairs. To address the difficulty involved in processing text in Simulink, an interface was written in Python to operate between the virtual programmable logic controller devices and the Simulink model. The interface receives JSON attribute-value pairs, strips the plaintext attribute portion and sends each command from a virtual programmable logic controller device to the appropriate UDP port. It also receives each individual value sent from Simulink, formats the values to be sent as an attribute-value paired UDP packet and sends the packet to a virtual programmable logic controller device.

## 5. Experimental Results

This section demonstrates the fidelity of the simulation by comparing the Python and Simulink simulations against the physical model. Normal behavior as well as startup and attack behaviors are compared and contrasted.

### 5.1 Normal Operation

Figures 8, 9 and 10 show the physical, Python and Simulink systems operating under normal conditions. The setpoints for all three systems are 15 psi, the time range is close to four minutes and the pressure scale is 0–25 psi. The dashed line represents the 15 psi setpoint.

As can be seen, the pressure changes in the Simulink simulation resemble those in the physical system much more closely than the Python simulation. While the Python simulation can be modified to resemble the physical model more closely, this would require the modification of the system equations described above. In the Simulink model, a change to the compressor speed, compressor efficiency, valve size or other physical property can alter the pressure response of the system.

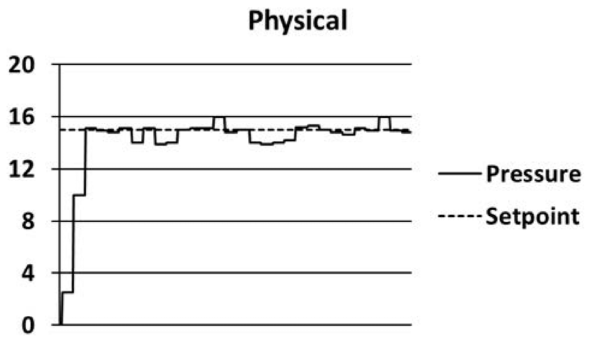

*Figure 8.* Physical system (normal operation).

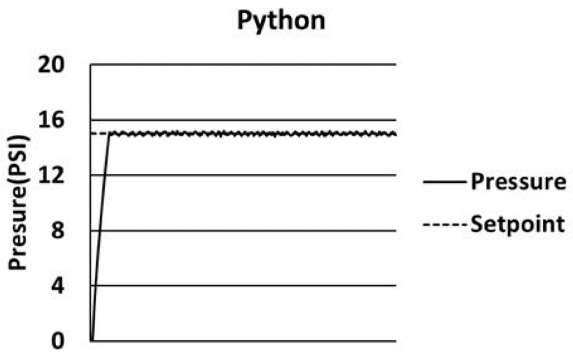

*Figure 9.* Python system (normal operation).

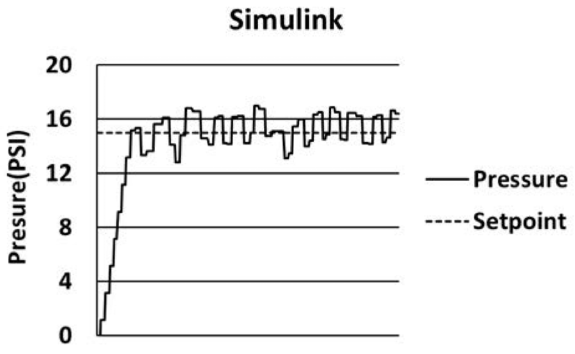

*Figure 10.* Simulink system (normal operation).

## 5.2 Startup Operation

Figure 11 shows the system behavior at startup. As seen in the figure, the stair-step increase and the initial overshoot of the physical model are more closely replicated by the Simulink model than by the Python model.

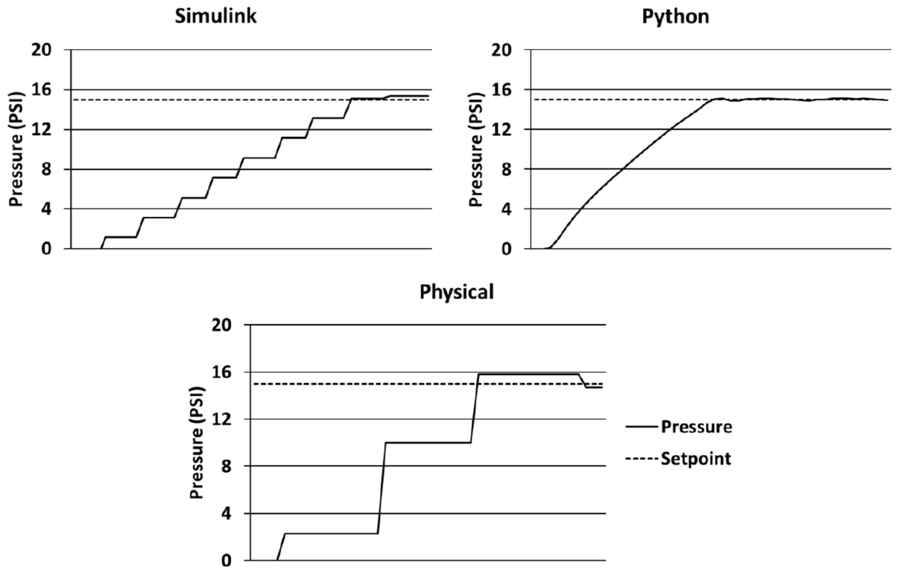

*Figure 11.* System startup.

## 5.3 Attack Operation

Command injection attacks can be used by malicious entities to adjust the settings in a SCADA system. One way to implement such an attack is to impersonate a SCADA client, send a command to the server and modify settings such as the setpoint, control (PID) parameters and valve state. An example altered control setpoint attack was implemented. In this attack, the attacker purported to be a Modbus device with a unique Modbus device number, acted as a client and sent a command to the server to alter the system setpoint. Upon receiving the command, the server proceeded to alter the setpoint and adjust the process actuators to achieve the new setpoint.

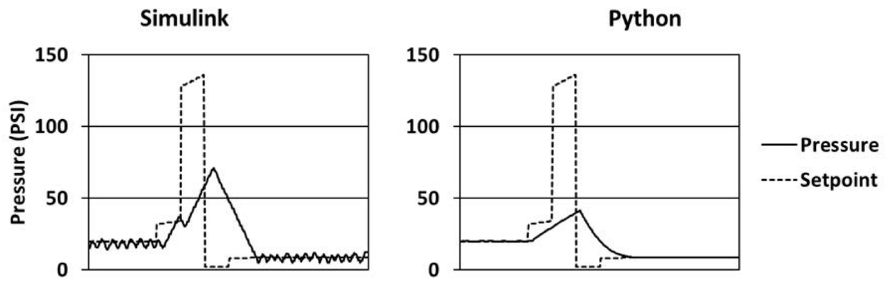

*Figure 12.* Python and Simulink simulations under attack.

Figure 12 shows the Python and Simulink simulations under the altered control setpoint attack. The graph has been rescaled to 0–150 psi to accommodate the increased system pressure as a result of the attack. The physical system response is not presented because it is incapable of withstanding the high pressure resulting from the injection attack. Nevertheless, the comparison demonstrates the advantages of the Simulink simulation in that it can achieve much higher pressures than the Python simulation with a simple change and without having to create a new empirical model of the system.

## 6. Conclusions

The virtual laboratory described in this chapter is designed specifically to support SCADA security research. It models small-scale industrial processes, supports programmable logic controller programming, employs the widely-used Modbus/TCP protocol and incorporates commercially-available human-machine interfaces. The process simulation was enhanced by adding a Simulink model of a gas pipeline. The Simulink addition increases the fidelity of the process model and facilitates modifications to the model as well as the incorporation of complex models. Moreover, the laboratory is portable due to its small size and extensive use of virtualization.

The laboratory has facilitated the analysis of weaknesses in the Modbus protocol as well as the identification and exploration of a number of attacks. It is capable of producing normal and malicious Modbus traffic for intrusion detection, anomaly detection and machine learning research.

Future research will focus on implementing a network of programmable logic controllers and a larger gas pipeline system with multiple gas compressor stations. This new configuration will support the investigation of the effects of cyber attacks on a large distributed infrastructure. Other activities will focus on creating datasets of normal and malicious SCADA network traffic to support the development of sophisticated intrusion and anomaly detection techniques. Another enhancement will involve the communications between the Simulink simulation and virtual programmable logic controller devices that currently employ JSON attribute-value pairs in UDP packets. This method is not used by industrial sensors. Therefore, efforts will be undertaken to replace the current method with an industrial sensor protocol such as WirelessHART or ZigbeePro.

## Acknowledgement

This research was supported by the NSF Secure and Trustworthy Cyberspace Program under Grant No. 1315726 and by the Pacific Northwest National Laboratory under U.S. Department of Energy Contract DE-AC05-76RL01830.

## References

[1] J. Beaver, R. Borges-Hink and M. Buckner, An evaluation of machine learning methods to detect malicious SCADA communications, *Proceedings of the Twelfth International Conference on Machine Learning and Applications*, vol. 2, pp. 54–59, 2013.

[2] J. Dabney and T. Harman, *Mastering Simulink*, Prentice Hall, Upper Saddle River, New Jersey 2003.

[3] B. Genge, C. Siaterlis, I. Nai Fovino and M. Masera, A cyber-physical experimentation environment for the security analysis of networked industrial control systems, *Computers and Electrical Engineering*, vol. 38(5), pp. 1146–1161, 2012.

[4] A. Hahn, B. Kregal, M. Govindarasu, J. Fitzpatrick, R. Adnan, S. Sridhar and M. Higdon, Development of the PowerCyber SCADA security testbed, *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, article no. 21, 2010.

[5] J. Hutchinson, Vertebrate flight, University of California Museum of Paleontology, Berkeley, California (`www.ucmp.berkeley.edu/vertebrates/flight/physics.html`), 1996.

[6] Idaho National Laboratory, Common Cyber Security Vulnerabilities Observed in Control System Assessments by the INL NSTB Program, INL/EXT-08-13979, Idaho Falls, Idaho, 2008.

[7] W. Mahoney and R. Gandhi, An integrated framework for control system simulation and regulatory compliance monitoring, *International Journal of Critical Infrastructure Protection*, vol. 4(1), pp. 41–53, 2011.

[8] R. McGrew, Vulnerability Analysis Case Studies of Control System Human-Machine Interfaces, Ph.D. Dissertation, Department of Computer Science and Engineering, Mississippi State University, Mississippi State, Mississippi, 2012.

[9] T. Morris, R. Vaughn and Y. Dandass, A testbed for SCADA control system cybersecurity research and pedagogy, *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*, article no. 27, 2011.

[10] B. Obama, Executive Order 13636: Improving Critical Infrastructure Cybersecurity, The White House, Washington, DC, 2013.

[11] B. Reaves and T. Morris, An open virtual testbed for industrial control system security research, *International Journal of Information Security*, vol. 11(4), pp. 215–229, 2012.

[12] M. Schwartz, J. Mulder, J. Trent and W. Atkins, Control System Devices: Architectures and Supply Channels Overview, Sandia Report SAND2010-5183, Sandia National Laboratory, Albuquerque, New Mexico, 2010.