FRAMEWORK FOR EVALUATING THE READINESS OF CYBER FIRST RESPONDERS RESPONSIBLE FOR CRITICAL INFRASTRUCTURE PROTECTION

THESIS

Jungsang Yoon, CPT, USA

AFIT-ENG-MS-16-M-054

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-16-M-054

FRAMEWORK FOR EVALUATING THE READINESS OF CYBER FIRST RESPONDERS RESPONSIBLE FOR CRITICAL INFRASTRUCTURE PROTECTION

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Electrical Engineering

Jungsang Yoon, B.S.E.E.
CPT, USA

24 March 2016

AFIT-ENG-MS-16-M-054

FRAMEWORK FOR EVALUATING THE READINESS OF CYBER FIRST RESPONDERS RESPONSIBLE FOR CRITICAL INFRASTRUCTURE PROTECTION

THESIS

Jungsang Yoon, B.S.E.E.
CPT, USA

Committee Membership:

LTC Mason J. Rice, PhD
Chair

Maj Benjamin W. Ramsey, PhD
Member

Jonathan W. Butts, PhD
Member

AFIT-ENG-MS-16-M-054

Abstract

First responders go through rigorous training and evaluation to ensure they are adequately prepared for an emergency. As an example, firefighters continually evaluate the readiness of their personnel using a defined set of criteria to measure performance for fire suppression and rescue procedures. From a cyber security standpoint, however, this same set of criteria and rigor is severely lacking for the professionals that must detect, respond to and recover from a cyber-based attack against the nation's critical infrastructure.

This research provides a framework for evaluating the readiness of cyber first responders responsible for critical infrastructure protection. The framework demonstrates the development of evaluation environment, criteria and scenarios that are modeled from NFPA 1410 standards concept that is used for assessing the readiness of firefighters. The utility of framework is exhibited during a military cyber training exercise and demonstrates the ability to evaluate the readiness of cyber first responders for industrial control systems when responding to the cyber-based attacks in the scenarios. Although successful, the results and analysis provide a context to develop a physical processes simulation tool, called Y-Box. The Y-Box creates more accessible, representational, realistic and evaluation-friendly environment to enhance the framework. The Y-Box demonstrates its application through the simulation of the first two stages in a wastewater treatment plant. Its performance test demonstrates its ability to interface with different types of signals from multiple programmable logic controllers with an acceptable range of error. The utility of simulation is extended with the development of potential attacks that can be used in a cyber exercise involving industrial control systems.

AFIT-ENG-MS-16-M-054

I dedicate this thesis to my wife and daughter for their endless support and love.

Acknowledgements

I would like to sincerely thank my committee and Stephen Dunlap for the countless hours of support and encouragement throughout the research.

Jungsang Yoon

Table of Contents

List of Figures

Figure Page

List of Tables

List of Abbreviations

Abbreviation		Page
ICS	Industrial Control Systems	1
ISA	International Society of Automation	8
GICSP	Global Industrial Cyber Security Professional	8
CSSA	Certified SCADA Security Architect	8
SCADA	Supervisory Control and Data Acquisition	8
ENISA	European Union Agency for Network and Information Security	8
NSTB	National SCADA Test Beds	10
HMI	Human Machine Interface	11
ES	Engineering Station	13
PLC	Programmable Logic Controller	13
IO	Input and Output	13
VDC	Volts Direct Current	13
VAC	Volts Alternating Current	13
AI	Analog Input	13
DI	Digital Input	14
AO	Analog Output	14
DO	Digital Output	14
HVAC	Heating, Ventilation and Air Conditioning	15
WWTP	Wastewater Treatment Plant	28
ADC	Analog to Digital Converter	36
GPIO	General Purpose IO	36
DAC	Digital to Analog Converter	36

FRAMEWORK FOR EVALUATING
THE READINESS OF
CYBER FIRST RESPONDERS RESPONSIBLE FOR CRITICAL
INFRASTRUCTURE PROTECTION

I. Introduction

1.1 Motivation

In a scene repeated by several motion pictures, burglars conduct a heist to steal a precious piece of art from a museum by manipulating the security camera with a recording that shows normal activities. While the guards watch the manipulated view of the museum, the thieves effortlessly steal the art without being detected.

If the guards detected the manipulated camera view, it would have prevented the precious art piece from being stolen. Just like the guards have a central role in protection of their properties as a first line of defense, the cyber first responders for Industrial Control Systems (ICS) have their own to protect against cyber-based attacks. If this type of attack is unstopped and occurs to the national critical infrastructures, the damage done can have detrimental impacts on the public's safety. Evaluation on the readiness of cyber first responders for ICS is in critical need to minimize or prevent any damage from the cyber-based attacks. Currently, the evaluation of ICS cyber professionals is not standardized and primarily conducted through exam-based certifications, lacking real-time interaction with ICS.

1.2 Research Goals and Hypotheses

This thesis presents a framework for evaluating ICS cyber professionals through the development of scenarios including evaluation criteria and environments that are enabled by a physical processes simulation tool.

The research goals are:

1. The evaluation from the framework provides valuable feedback to improve the readiness of cyber first responders for ICS.

2. The simulation tool provides an accessible, representational, realistic and evaluation-friendly ICS environment, designed to train and assess the cyber first responders for ICS.

The research hypotheses are:

1. The evaluation concept for first responders can be extended to the evaluation of the cyber first responders for ICS.

2. The evaluation of cyber first responders for ICS can be conducted in an evaluation environment that simulates physical processes.

3. Physically observable characteristics from simulated physical processes can be effectively demonstrated through visualization.

4. The simulation tool can interact with types of physical signals typically used in industrial applications and connect to multiple programmable logic controllers at once.

This research proceeds with the assumption that high cost and geographical constraint to replicate the real physical processes prevents its implementation for evaluation environment.

1.3 Thesis Layout

Chapter 1 introduces the motivation and goal of this thesis. Chapter 2 describes background information that leads to framework development for evaluating the readiness of cyber first responders for ICS and a creation of physical processes simulation tool for evaluation environments. Chapter 3 explains the methods to evaluate the framework and the simulation tool. Chapter 4 discusses the results collected in Chapter 3. Chapter 5 summarizes with the conclusions and discusses a significance of this research. This chapter offers recommendations for future work.

1.4 Special Consideration

The simulation of physical processes for the evaluation environment in Section 3.1 is developed prior to the creation of physical processes simulation tool, to partially fulfill the requirements for this research. It is used to evaluate the framework and as a pilot study to see the effectiveness of the custom application model for the physical processes simulation before the full development of the tool. Described in Section 3.2, the tool is ultimately created to complement the limitations discovered from the pilot study. The limitations are described in Section 4.1.2. Chapter 4 provides analysis of results for the framework and the tool in Section 4.1 and 4.2, respectively.

II. Background and Literature Review

Section 2.1 compares the current evaluation for first responders (e.g., firefighters) with the one for ICS cyber professionals. Section 2.2 discusses different types of industrial control systems testbeds and provide a context for the development of a physical processes simulation tool. Section 2.3 provides an overview of elements that are minimally required to replicate an evaluation environment.

2.1 Current Evaluation

Evaluation of first responders using realistic scenarios plays a vital role in determining mission readiness in the areas of public safety. It is hard to imagine a newly recruited firefighter responding to an emergency situation without the proper assessment of their ability to perform. Moreover, it is inconceivable for a fire station to respond to a burning building without evaluating their personnel on the standard tactics required to fight a fire. Indeed, it is critical that firefighters have the ability to respond appropriately for the given situations they may face, such as the ability to adequately lay the initial attack line and back-up line, and obtain the appropriate water pressure within a time limit.

To evaluate the mission readiness of firefighters, fire departments often use the NFPA 1410 national standards as a common set of criteria [6]. The NFPA 1410 provides a scenario-based standard that has been adopted by the community for evaluating the readiness of firefighter first responders. The standards use real-world scenarios and specify objectives, evaluation criteria and metrics for assessing the readiness of firefighters. The evaluation scenarios identify weaknesses in training and provide assurance that personnel are ready to respond appropriately.

Although first responders have used common criteria guidelines for decades to assess the readiness of their personnel, the notion is in its infancy for cyber professionals. Current training evaluation relies primarily on exam-based certifications. This method of evaluation, however, is not sufficient given the responsibilities associated with national critical infrastructure protection.

A cyber-based attack against the nation's critical infrastructure could have devastating consequences that directly impact public safety. There is a growing awareness of the threats posed by cyber-based attacks and the implications; however, little is being done to ensure the competency and preparedness of the cyber professionals that will be called upon to detect, respond to and recover from an attack.

2.1.1 Assessing Readiness.

It is imperative that first responders are continually evaluated against realistic scenarios that may be encountered. Firefighters undergo extensive training and evaluation that mirrors real-world situations to ensure an individual will respond adequately when called upon. A common set of evaluation criteria helps prepare firefighters for such responses and helps identify training deficiencies that need attention. Unfortunately, this same set of criteria and rigor is severely lacking for the cyber security professionals associated with responding to a cyber-based attack against the nations critical infrastructure.

2.1.1.1 Standard on Training for Emergency Scene Operations.

Fire department personnel engaged in emergency scene operations use the NFPA 1410 evolutions standard for training evaluation [3]. This standard specifies criteria and metrics that can be adapted to local conditions and serves as a mechanism for evaluating minimum acceptable performance during training activities.

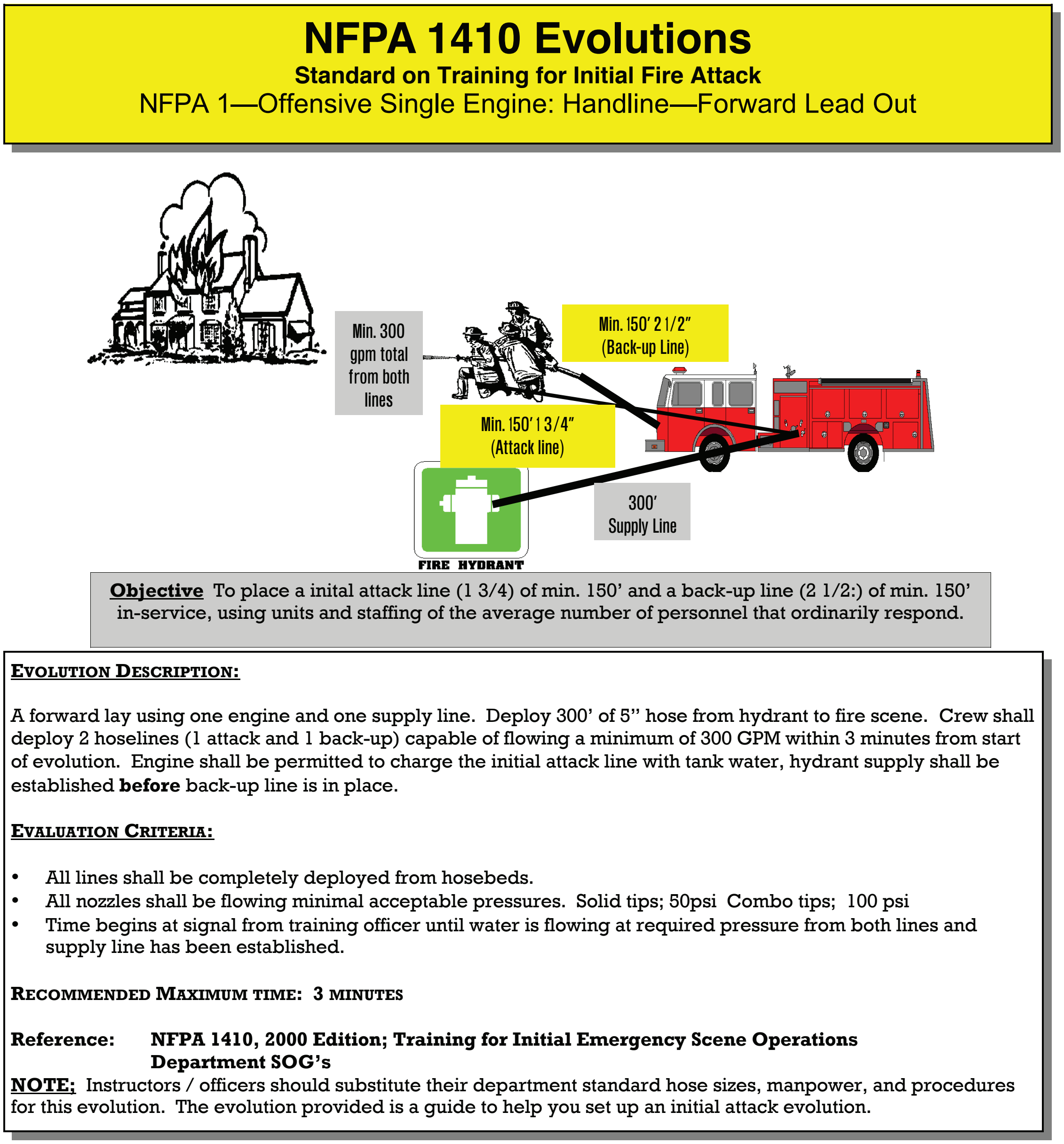
Figure 1 shows a representative evolution training standard for a handline-forward lead out operation. This example simulates a response to a typical structure fire where the company must secure a hydrant and lay supply lines towards the building on fire. The firefighters are evaluated on the ability to correctly apply the forward lay water supply tactic to obtain the appropriate water pressure to suppress a fire.

The example highlights the various criteria the team is evaluated on and specifies the maximum time to complete the objective. The NFPA 1410 provides numerous scenarios and criteria for evaluation that are based on tactics that relate to real-world scenarios. It is important to note that the guidelines and criteria can be adapted to meet local and scenario-specific requirements.

2.1.1.2 Cyber First Responders.

Historical events have demonstrated the susceptibility to disruptive cyber-based attacks against critical infrastructure systems [31]. Attacks against ICS are on the rise as they target operational capabilities within power plants, factories and refineries [11]. As an example, ICS-CERT has issued alerts for multiple campaigns (e.g., Havex RAT [16] and BlackEnergy [18]) aimed at targeting critical systems by exploiting vulnerabilities in products from GE, Advantech/Broadwin and Siemens [17]. Similarly, a recent SANS report claims that a cyber attack was responsible for a power outage in Ukraine [23]. According to the report, hackers likely compromised control systems and installed malware to trip breakers to cut power and prevent technicians from detecting the attack.

Attacks targeting national critical infrastructure can result in devastating consequences. As a first line of defense, organizations spend increasing amounts of money to train and hire cyber security personnel to prevent, identify and mitigate attacks [29]. From a maturity standpoint, however, the ability to evaluate the readiness of


Min. 300 gpm total from both lines
Min. 150' 2 1/2" (Back-up Line)
Min. 150' 1 3/4" (Attack line)
300' Supply Line
FIRE HYDRANT

NFPA 1410 Evolutions

Standard on Training for Initial Fire Attack

NFPA 1—Offensive Single Engine: Handline—Forward Lead Out

Objective To place a inital attack line (1 3/4) of min. 150' and a back-up line (2 1/2:) of min. 150' in-service, using units and staffing of the average number of personnel that ordinarily respond.

EVOLUTION DESCRIPTION:

A forward lay using one engine and one supply line. Deploy 300' of 5" hose from hydrant to fire scene. Crew shall deploy 2 hoselines (1 attack and 1 back-up) capable of flowing a minimum of 300 GPM within 3 minutes from start of evolution. Engine shall be permitted to charge the initial attack line with tank water, hydrant supply shall be established **before** back-up line is in place.

EVALUATION CRITERIA:

- All lines shall be completely deployed from hosebeds.
- All nozzles shall be flowing minimal acceptable pressures. Solid tips; 50psi Combo tips; 100 psi
- Time begins at signal from training officer until water is flowing at required pressure from both lines and supply line has been established.

RECOMMENDED MAXIMUM TIME: 3 MINUTES

Reference: **NFPA 1410, 2000 Edition; Training for Initial Emergency Scene Operations**
Department SOG's

NOTE: Instructors / officers should substitute their department standard hose sizes, manpower, and procedures for this evolution. The evolution provided is a guide to help you set up an initial attack evolution.

Figure 1. Example NFPA 1410 evolution training standard [6].

cyber first responders is in its infancy. Training is disparate and the requisite skill sets have not been standardized [21]. Much attention has been given to frameworks for system security and organizational risk (e.g., the NIST Framework for Improving Critical Infrastructure Cybersecurity [20]); however, organizations do not have a standardized means to evaluate if personnel in a cyber first responder role are adequately prepared to respond to an incident.

Current evaluation for ICS cyber security skill-sets relies primarily on professional certifications. The International Society of Automation (ISA), a professional association, developed a knowledge-based certificate program designed to test the security standards described in ISA99 through a multiple choice exam[13]. The ISA99 standard provides guidelines in areas such as requirements for ICS security management, security risk assessment and system design, and technical security requirements for ICS components. Similarly, the Global Information Assurance Certification organization offers the Global Industrial Cyber Security Professional (GICSP) certification that tests ICS security professionals on essential ICS security related knowledge areas [7]. The topics for the test questions include access management, cybersecurity essentials for ICS, ICS architecture, ICS modules and elements hardening and ICS security monitoring. The Information Assurance Certification Review Board offers a Certified SCADA Security Architect (CSSA) certification for individuals that pass a 100 question exam on knowledge relating to securing a Supervisory Control and Data Acquisition (SCADA) system [4].

The primary concerns with the certification programs are a lack of evaluation criteria against a common set of standards and assessing the ability to apply knowledge, concepts, or experiences to real-time situations associated with an actual exploitation of ICS [10]. In a study performed by the European Union Agency for Network and Information Security (ENISA) that examined existing ICS certification programs, a key recommendation was the development of a framework for standardizing and evaluating certified ICS security personnel [21].

In addition to certification programs, United States Government organizations have implemented various critical infrastructure response efforts to include the Cyber Defense Initiative (CDI) and Cyber Storm. The CDI is sponsored by the Federal Emergency Management Agency (FEMA) and offers training courses to prepare technical personnel and managers associated with critical infrastructure protection [15]. The training uses lectures, lab exercises and online material to help students prepare for and respond to a cyber-based terror attack.

Similarly, Cyber Storm is a DHS-sponsored exercise initiated in 2006 that tests and evaluates the plans, policies and procedures for cyber security response professionals [19]. Primarily intended to evaluate coordination and information sharing, Cyber Storm focuses on policies and procedures associated with responding to a cyber-based attack against the nations critical infrastructure.

Both government-sponsored efforts highlight the need for a standardized evaluation framework for cyber first responders. Indeed, a common evaluation criteria is needed that can be tailored to an organizations respective environment.

2.2 Evaluation Environment

An evaluation environment including real ICS is necessary to provide real-time situations that evaluation criteria can be applied within. In most cases, the direct use of live ICS is not always feasible due to high loss caused by down time and potential damage for evaluation. Responding to this shortfall, many types of ICS testbeds were developed as solutions for the cyber security research including functionality tests between control systems and education for the responders.

2.2.1 Existing Testbeds.

In order to provide a realistic ICS environment, large-scale testbeds in places like Idaho National Labs and Sandia National Labs recreate the real-world control systems, networks and physical processes [14]. Mississippi State's ICS testbed presents a real-world control system and real-world physical processes to support its cyber security research and education [12]. Others fully or partially simulate their desired ICS environments with or without the real-world equipment. Reaves *et al.*'s [5] testbed fully simulate its ICS environment with virtual devices and simulator to replace the control systems and physical processes, respectively. Wertzberger [32] *et al.*'s testbed simulates physical processes and network while employing real-world control systems. This research focuses on creating the ICS environment through the simulation of physical processes. The simulation of physical processes in this research is designed to satisfy the following attributes:

- Accessible: The physical processes are not geographically limited and cost much less than the full suite of equipment.

- Expandable: The physical processes may be expanded to reflect a complex ICS environment.

- Compatible: The physical processes may be connected to the different types of real-world control systems.

- Separable: The physical processes may be monitored separately from the control system interface.

The current solutions to the physical processes in their testbeds are summarized according to attributes in Table 1.

While the physical processes in National SCADA Test Beds (NSTB) and Mississippi State could be ideal, they are constrained by geographic location and are quite

Table 1. Attributes of physical processes in the testbeds.

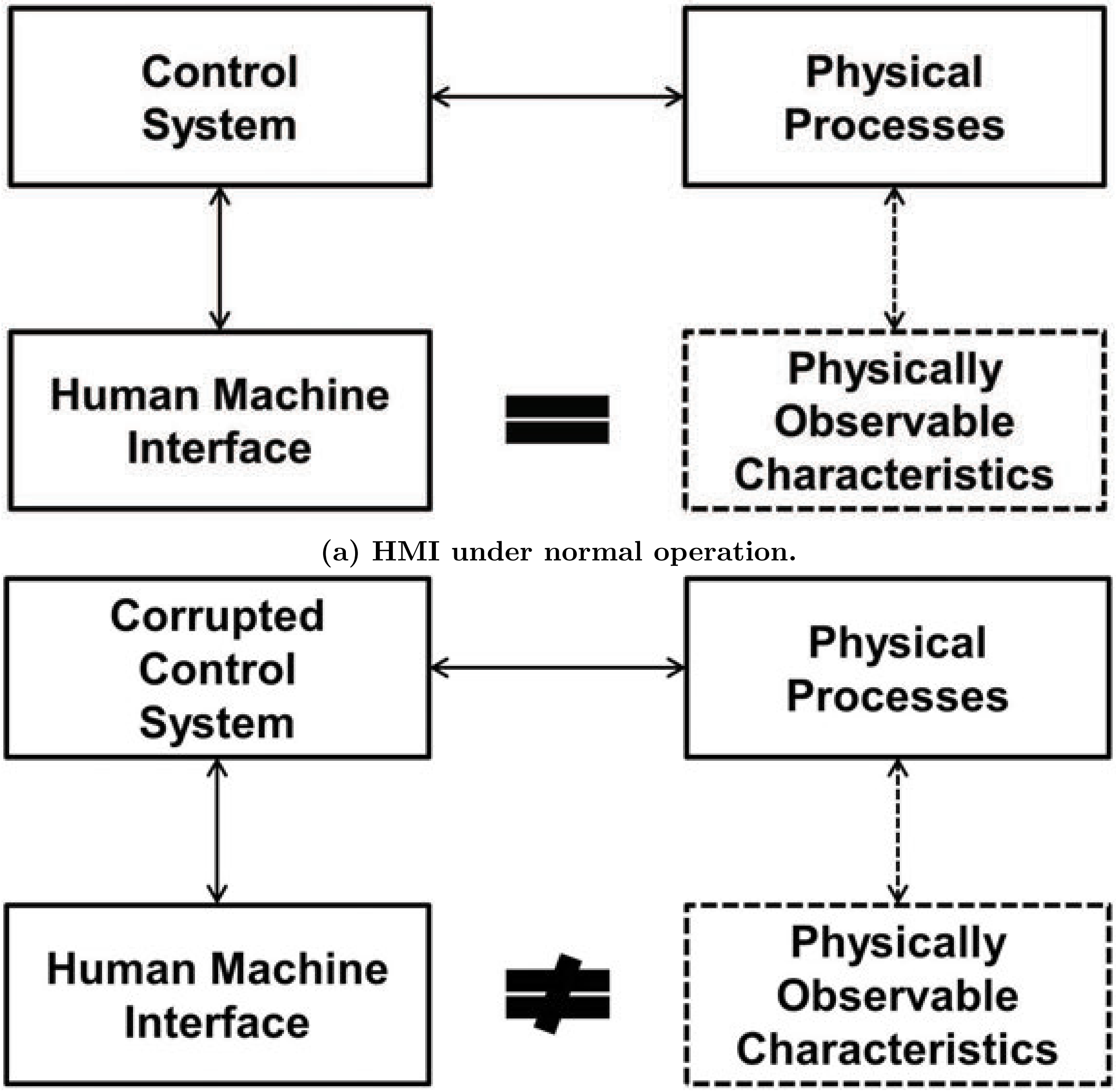
Physical processes	Accessible	Expandable	Compatible	Separable
NSTB		X	X	X
Mississippi State University		X	X	X
Reaves *et al.*	X	X	X	
Wertzberger *et al.*	X	X	X	

expensive to build. Although the testbeds from Reaves *et al.* and Wertzberger *et al.* can be accessible, expandable and connected to the real-world control systems, they do not necessarily separate the views from the physically observable characteristics of physical processes and what are processed by the control systems. The attribute of separable allows the view of physically observable characteristics, which can be used to discern the Human Machine Interface (HMI) under normal operation from maliciously modified or malfunctioning HMI as seen in Figure 2.

If a museum wanted to replicate the movie scene discussed in Section 1.1 as an exercise to test their defensive capabilities, the exercise coordinators might be tempted to use the security cameras to monitor the progress of the thieves and to evaluate the response of the guards. Were the thieves to execute the same camera attack, the coordinators would be as blind as the guards themselves. Sadly, this is exactly how cyber exercises are conducted today; the coordinators rely on the same view of the exercise as the defenders. If attackers manipulate the view of the defenders, the coordinators are unable to identify what the attackers have done and why the defenders were unable to detect the changes. A view that can't be altered by attackers is necessary for effective control and evaluation by the coordinators.

2.3 Industrial Control Systems and Physical Processes

The evaluation environment is primarily consisted of ICS and physical processes.



(a) HMI under normal operation.

(b) Maliciously modified or malfunctioning HMI.

Figure 2. Comparison of HMIs to the views of physically observable characteristics of physical processes.

2.3.1 Industiral Control Systems.

ICS is a general term used to describe an automation system that manages and monitors the functionality of an industrial physical processes [9]. The HMI or Engineering Station (ES) and Programmable Logic Controller (PLC)s are minimally required components of ICS for the evaluation environment.

2.3.1.1 HMI and ES.

An HMI or ES displays communications to and from PLCs and other physical processes in a human-readable interface. This provides ICS operators the ability to manage and monitor the physical processes [9].

2.3.1.2 Programmable Logic Controllers.

PLCs can be specialized to automate the functions in the physical processes [9]. PLCs for ICS typically include CPU modules, communication modules, power supply and various types of Input and Output (IO) modules [22]. CPU modules act as the primary control unit of the PLC and are programmed for each specific application. Communication modules allow PLCs to communicate with other devices, such as HMIs, data historians and other PLCs. IO modules allow PLCs to interface with physical sensors and actuators to control an industrial processes. A sensor is any device that measures an attribute of a system (e.g., temperature, pressure and flow rate). An actuator is any device that controls physical aspects of a system (e.g., motor, valve and heating element). Types of IO modules include analog, digital and specialty modules. Digital signals are measured as either high or low and can be either Volts Direct Current (VDC) or Volts Alternating Current (VAC). Typical analog signals can be measured as either a voltage (0 VDC to 10 VDC) or as a current (4 mA to 20 mA or 0 mA to 20 mA) in an industrial application. Analog Input (AI) modules and

Digital Input (DI) modules allow a PLC to collect data from various types of sensors. Analog Output (AO) modules and Digital Output (DO) modules allow a PLC to send control data to actuators. Specialty modules, such as thermocouple input modules, are used in ICSs to accomplish application-specific operations. This research focuses on the interaction with the analog and digital modules of PLCs.

2.3.2 Physical Processes.

Physical processes (e.g., bar screening and grit removal from wastewater) can be automated with the use of actuators and sensors in industrial facilities. PLCs command actuators to control the physical processes which is generally based on process data from sensors.

III. Methodology

Section 3.1 describes the experimental design of evaluation criteria, environment and scenarios for the framework, to be evaluated in a cyber training exercise. Section 3.2 describes the functionality of physical processes simulation tool referred as Y-Box hereafter. It further describes the Y-Box's experiment set-up for a performance test and applications.

3.1 Development and Evaluation of Framework

To help develop the framework for evaluating the readiness of cyber first responders, a study is conducted using the NFPA 1410 concept applied to ICS cyber security personnel. The study evaluates the readiness of cyber security personnel to respond to targeted cyber attacks during a military cyber training exercise that occurred over a one week period. Although various scenarios are incorporated throughout the training exercise, the evaluation for cyber first responders focuses on developing scenarios, defining criteria and specifying metrics for evaluation using the NFPA 1410 concept. At the conclusion of the military cyber training exercise, the study is evaluated to validate the utility of framework.

The training exercise consisted of: (i) a team of military cyber professionals responsible for ICS security; (ii) a red team that applies adversary tactics to exploit ICS and create effects; and (iii) an evaluation team that measures the effectiveness of the cyber professionals at identifying, responding and mitigating the attacks initiated by the red team. The training environment consisted of a Siemens Apogee Heating, Ventilation and Air Conditioning (HVAC) system that is configured for standard operations. An evaluator monitoring capability is implemented for the Apogee system to enable the evaluation team to discern physical effects created by the red team and observe response actions of the cyber professionals.

3.1.1 Evaluation Criteria.

The training objectives, tactics and criteria are specified using the NFPA 1410 concept. The following example details an evaluation scenario for ICS cyber security tasks.

Denial-of-service attack targeting the human machine interface (HMI)

- Objective: Identify a denial-of-service attack and minimize effects on physical system operations.
- Description: Denial-of-service attack is launched from external IP that exhausts resources on the HMI computer.
- Type: Loss of control and loss of awareness.
- Evaluation Criteria:
  - Detect within three minutes.
  - Report within six minutes.
  - Prevent physical process down time.
- Reference: Operational response plans and security response plans.

In this example, the cyber security professionals should detect a denial-of-service attack targeting the HMI within three minutes and report the attack through appropriate channels within six minutes. Additionally, the physical process should not be disrupted. The attack creates a loss of control and awareness for the operator. Operational response plans and security response plans are used to derive the expected actions and requirements for the cyber first responders.

3.1.2 Evaluation Environment.

There are certain limitations in creating an operational environment for assessment of ICS cyber security tasks. The use of real-world systems is not always feasible due to potential damage and down-time for evaluation. Additionally, developing a real-world system for evaluation purposes (e.g., water treatment facility, HVAC of a building and gas pipeline flow control system), can be prohibitively expensive and time consuming. The ability to incorporate ICS components coupled with simulated attributes, however, has proven useful for training environments [8].

The cyber training exercise and evaluation incorporates the Siemens Apogee HVAC system that includes an air conditioner, heater, and fan. To simulate the physical processes and enable the evaluation team to observe physical effects, the input/output settings for temperature and flow are simulated using a process control model implemented in software. Figure 3 shows the Apogee system and components used in the cyber training exercise.

PXCM1 is a programmable controller-modular (PXCM) that provides digital and analog control of building sensors and actuators. PXCM1 receives sensor measurements through the analog module and initiates control actions according to programmed logic through the digital module. For example, if a sensor is reporting a temperature of 73° F and the air conditioner setpoint is configured for 70° F, PXCM1 will initiate a control signal to turn on the air conditioner. The status of the system is relayed through an operator control panel that provides indicator lights and readings for current conditions and provides the ability to override or modify settings

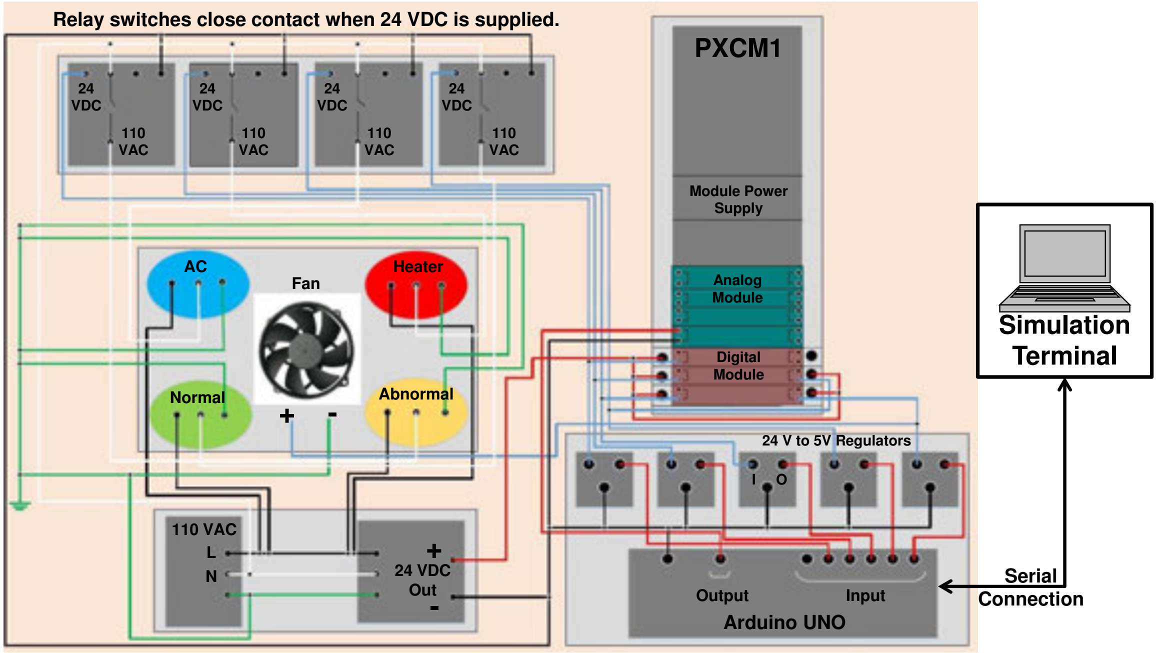


Figure 3. Apogee HVAC system schematics.

[30]. The Arduino UNO is a customizable microcontroller that provides 6 analog inputs, 14 digital inputs/outputs and 6 PWM outputs. The Arduino microcontroller is incorporated into the training exercise configuration to emulate the actuators and sensors that control the physical environment conditions.

For the physical environment, a custom application in the simulation terminal is used to emulate operating conditions. The application characterizes heating and cooling parameters that alter the environmental settings. As shown in Figure 4, the simulation terminal provides inputs to the Arduino microcontroller based on the application output that represents environmental conditions (e.g., air temperature). The microcontroller translates the settings into analog and digital voltage and sends the corresponding signals to PXCM1. PXCM1 receives the signals and initiates the appropriate control and notification actions. Similarly, when an operator initiates a control action, the PXCM1 sends the control message to the Arduino UNO which, in turn, updates the environmental variables. For example, if an operator initiates an action to turn on the cooling fan, the PXCM1 sends a digital control signal that is received by the Arduino. The Arduino sends an update message to the simulation terminal to turn on the cooling fan. The simulation incorporates the change, and the environment temperature will start to decrease accordingly.

The initial environment settings specified in the custom application include setpoint, minimum temperature and maximum temperature. The real-time status of the heating and air conditioning is represented via a graphical interface shown in Figure 5. Note that this representation indicates the true physical state of the system and is considered "ground truth" for environmental conditions.

The Apogee system incorporates FTP (Port 21) and Telnet (Port 23) services as well as the Siemens P2 application (Port 5033) for management services. Using these protocols, the HMI provides the graphical interface for the operators. As represented in Figure 6, the display shows the current status of the system and enables the ability to control the environment. Under normal operating conditions, the HMI should reflect the actual physical environment conditions (i.e., the HMI should be consistent with the custom application interface). An attacker that targets the integrity of the system could exploit the Apogee system and change the operating parameters, while masking the changes in the environment from the operator on the HMI.

3.1.3 Scenarios.

The scenarios are developed to evaluate specific objectives for cyber security professionals. Note that for the purposes of this paper, the focus is on defining a framework for evaluating cyber first responders, and not the performance of the cyber security professionals during this specific training exercise. As such, the focus of analysis is on the ability to use the NFPA 1410 concept for evaluating cyber first responders in an actual training exercise scenario.

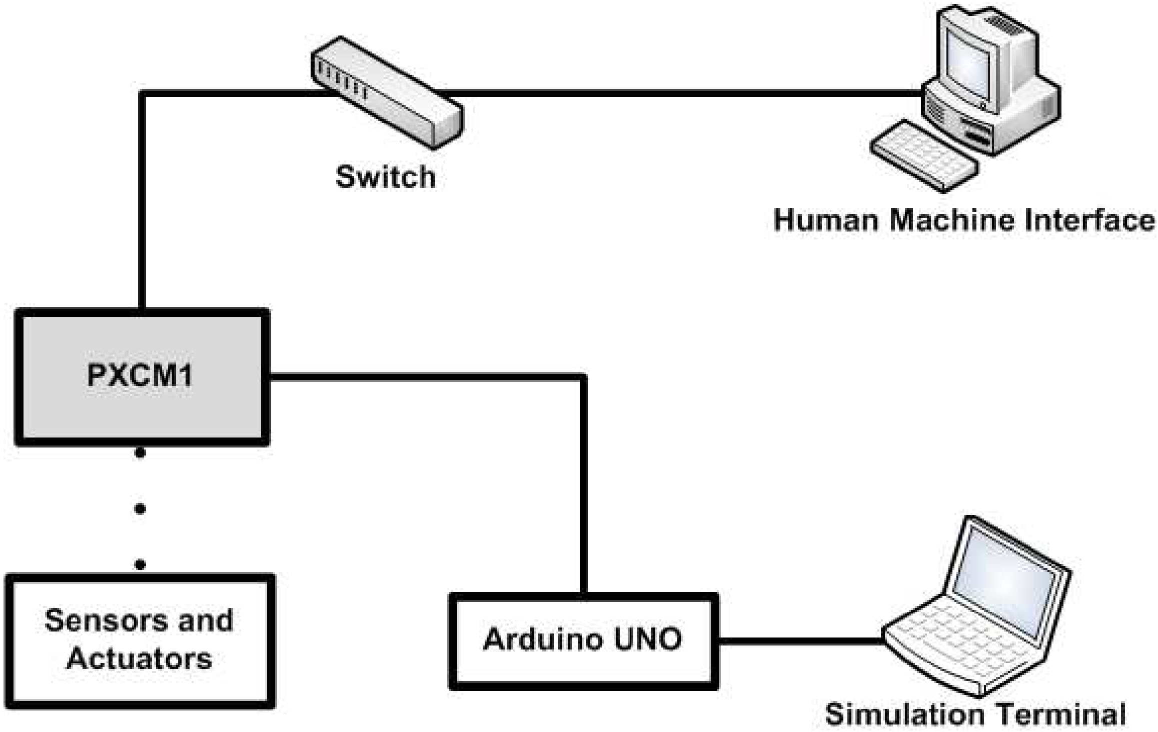


Figure 4. Functional diagram of the exercise system environment.

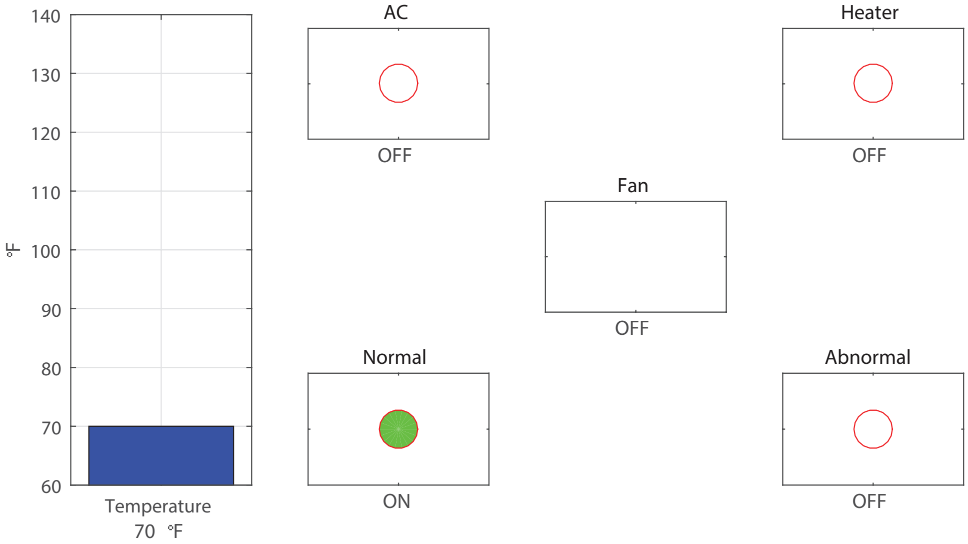


Figure 5. Representation of the real-time status of environmental conditions.

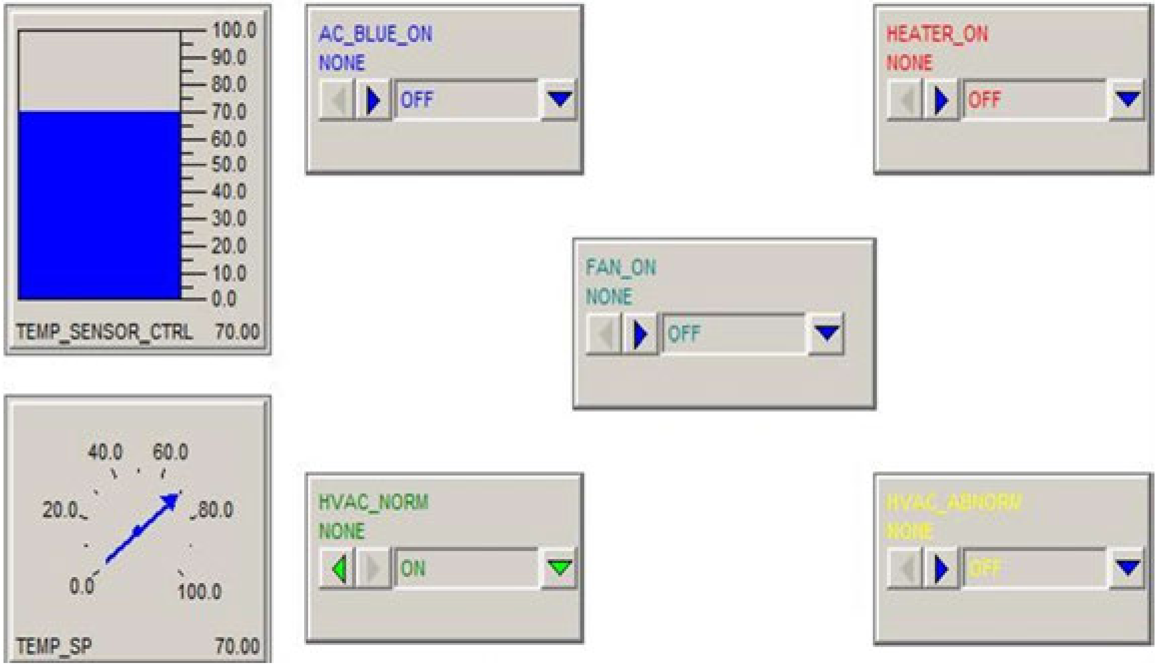


Figure 6. The operator HMI for monitoring and controlling environmental conditions.

The scenarios are derived using operational requirements and expected actions of the cyber security professionals. Five scenarios are presented that are evaluated during the training exercise using the NFPA 1410 concept.

Gain Remote Access and Exfiltrate Data

- Objective: Identify remote system compromise and detect exfiltration of critical system data.
- Description: The attacker performs a dictionary attack on the PXCM1 FTP username/password and uses the credentials to login via Telnet. The Telnet service for Apogee allows the attacker to change configurations or values to control the HVAC and provides the attacker remote access. The attacker also uses the FTP server to exfiltrate Apogee configuration data.
- Type: Loss of confidentiality.

- Evaluation Criteria:
  - Detect within three minutes.
  - Report within six minutes.
  - Determine how system was compromised and identify all compromised components.
  - Remove compromised access.
  - Reconfigure to prevent further compromise.
  - Prevent physical process down time.
- Reference: Standard operating procedures, response action plans and cyber defensive tactics, techniques and procedures manual.

System Denial-of-Service Attack

- Objective: Identify denial-of-service attack and restore system control.
- Description: The attacker uses a SYN flood to exhaust the resources of PXCM1. The attack prevents the operator from controlling system processes.
- Type: Loss of control.
- Evaluation Criteria:
  - Detect within three minutes.
  - Report within six minutes.
  - Recover physical process control within five minutes.
  - Determine how system was compromised and identify all compromised components.

  - Remove compromised access.
  - Reconfigure to prevent further compromise.
- Reference: Standard operating procedures, response action plans and cyber defensive tactics, techniques and procedures manual.

System Crash

- Objective: Identify system compromise and restore system control and monitoring.
- Description: The attacker exploits a known vulnerability for the Apogee HMI OS running Windows Vista SP1. A metasploit module allows the attacker to exploit and crash the Apogee server.
- Type: Loss of control and loss of awareness.
- Evaluation Criteria:
  - Detect within three minutes.
  - Report within six minutes.
  - Recover physical process control within five minutes.
  - Determine how system was compromised and identify all compromised components.
  - Remove compromised access.
  - Reconfigure to prevent further compromise.
- Reference: Standard operating procedures, response action plans and cyber defensive tactics, techniques and procedures manual.

Repeated Reboot

- Objective: Identify system compromise, eradicate malware and restore system control and monitoring.
- Description: The attacker installs a script on the network that sends a repeated reboot command to PXCM1 through the Siemens P2 protocol. PXCM1 receives the command and authorizes a reboot of the system. After the system reboots, another command initiates a reboot to repeat the process.
- Type: Loss of control and loss of awareness.
- Evaluation Criteria:
  - Detect within three minutes.
  - Report within six minutes.
  - Identify the malicious script installed that is forcing the reboot.
  - Remove the script without effecting configuration of the physical process.
  - Determine how system was compromised and identify all compromised components.
  - Remove compromised access.
  - Reconfigure to prevent further compromise.
- Reference: Standard operating procedures, response action plans and cyber defensive tactics, techniques and procedures manual.

Covert Manipulation of Control

- Objective: Identify system compromise, eradicate malware and restore system operation parameters.

- Description: The attacker gains access to the Apogee system and modifies the control code on PXCM1. The attacker increases the actual physical temperature, while providing a normal display on the HMI.
- Type: Loss of integrity.
- Evaluation Criteria:
    - Detect within six minutes.
    - Report within nine minutes.
    - Identify the malicious script that is manipulating the temperature and masking the impacts.
    - Remove the script without effecting configuration of the physical process.
    - Determine how system was compromised and all compromised components.
    - Remove compromised access.
    - Reconfigure to prevent further compromise.
- Reference: Standard operating procedures, response action plans and cyber defensive tactics, techniques and procedures manual.

3.1.3.1 Attack Details.

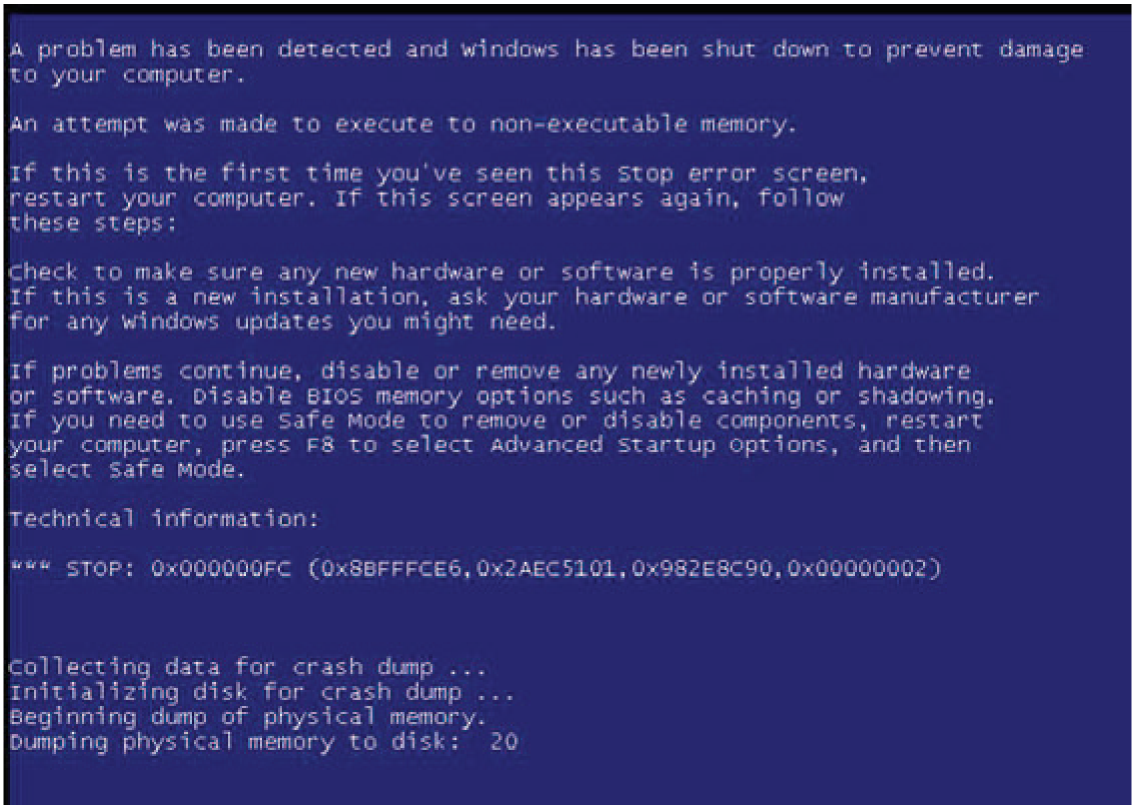
This section details the various attack scenarios and key indicators for the cyber defense team.

- Gain Remote Access and Exfiltrate Data: The red team initiates an exploit to obtain the admin username and password to extract sensitive data stored in

PXCM1 using the FTP service. The default usernames for Apogee accounts are classified as high, medium and low for different privileges based on roles. The high classification includes admin privileges, and the username and passwords are not case sensitive. Additionally, the login credentials are the same for the FTP and Telnet services. The red team executes a dictionary attack to obtain the credentials and gain remote system access through the Telnet service. Once system access is obtained, the red team accesses the PXCM1 configuration files and extracts the data using the FTP service. Key indicators for the cyber professional team include failed login attempts, unauthorized access and increased network activities.

- System Denial-of-Service Attack: Using adjacent network access, the red team initiates a denial-of-service attack using a SYN flood to exhaust the Apogee system resources. The red team gains access to a vulnerable system on the same network segment, and then sends a large number of SYN packets to PXCM1. The attack results in a loss of control for the operator by preventing the ability to send control commands for the HVAC system. Key indicators for the cyber professional team include increased network traffic, unauthorized access and loss of system control.

- System Crash: The red team utilizes a known vulnerability for the Apogee server operating system (Windows Vista SP 1) to force a system crash. Using adjacent network access, the red team initiates an attack that causes the Apogee server to crash, as shown in Figure 7. The attack results in a loss of control for the operator until the system is manually rebooted. Key indicators for the cyber professional team include unauthorized commands, unauthorized access and system failure.



```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

An attempt was made to execute to non-executable memory.

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x000000FC (0x8BFFFCE6,0x2AEC5101,0x982E8C90,0x00000002)



Collecting data for crash dump ...
Initializing disk for crash dump ...
Beginning dump of physical memory.
Dumping physical memory to disk:  20
```

Figure 7. Bluescreen effect created by system attack.

- Repeated Reboot Attack: The red team initiates an attack to force PXCM1 to continually reboot. Using services offers via the Siemens P2 protocol, the red team uses a malicious script to send repeated reboot commands to PXCM1. The attack results in loss of control for the operator and the inability to monitor environmental conditions. Key indicators for the cyber professional team include unauthorized commands and system failure.

- Covert Manipulation of Control: The red team gains remote access to the Apogee server and manipulates the control code of PXCM1. In this attack, the red team overrides the temperature reading with a manipulated value. As seen in Figure 8, the operator sees normal operating conditions, whereas the attack has caused the actual environment to increase in temperature. Key indicators for the cyber professional team include unauthorized commands, unauthorized access and compromised system integrity.

3.2 Functionality and Evaluation of Y-Box

Section 3.2.1 and 3.2.2 explains the Y-Box system architecture and hardware design to describe its functionality. Performance test and applications of the Y-Box are described in Section 3.2.3 and 3.2.4.

3.2.1 System Architecture.

For the development of the first version of Y-Box, a preliminary site survey is conducted in a Wastewater Treatment Plant (WWTP). The types of Y-Box input and output modules in the first version are similar to the types seen in the PLCs in the surveyed wastewater treatment plant.

A key design consideration is that the Y-Box exchanges physical signals only, such as voltage or current, with PLCs to not interfere with any cyber activities in the

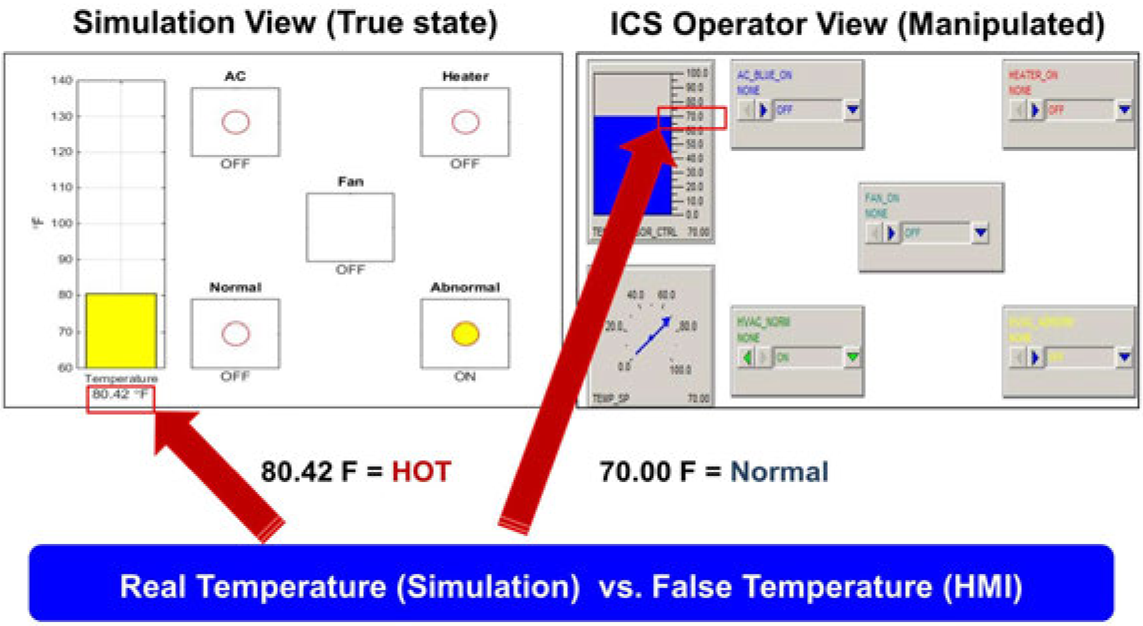


Figure 8. Attack results that manipulate environmental settings and mask implications from the operator.

connected control systems. For the physical processes simulation, the Y-Box interacts directly with a PLC's IO modules to facilitate the real-time and accurate exchanges of physical signals, replacing PLCs interaction with the real actuators and sensors. A simulation terminal collects actuator data from PLCs, simulates the response of a physical system, and then generates sensor values to send back to the PLCs. To interact with a PLC, the Y-Box inputs are connected to the PLC outputs, and the Y-Box outputs are connected to the PLC inputs. This process is shown in Figure 9. Similar to a PLC, the Y-Box consists of a CPU module, an optional communication module, and various IO modules. The proposed Y-Box architecture is shown in Figure 10. The Y-Box is intended to be flexible. The Y-Box modules are designed to allow connection to a variety of PLCs, sensors and actuators.

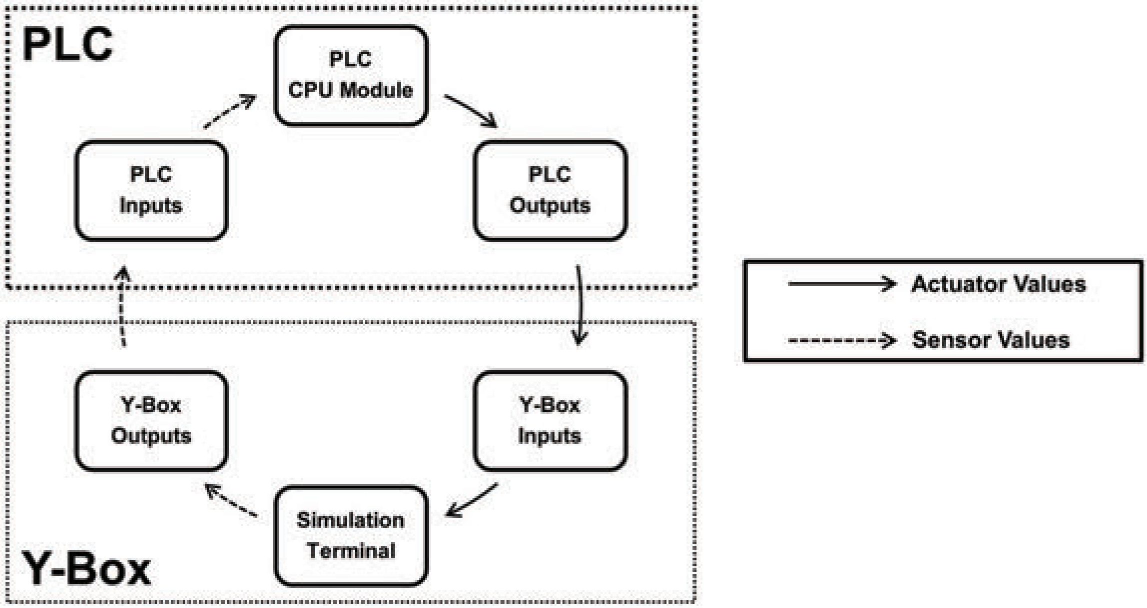


Figure 9. Simulation process.

3.2.1.1 Analog Input Modules.

AI modules read analog signals generated by a PLC intended to control actuators (e.g., pump and control valve). AI modules on the Y-Box support up to eight inputs simultaneously and can measure voltage and current. The analog data is then retrieved by the CPU module for use in simulation. The AI module supports physical actuators to connect to the voltage or current signals without negatively impacting the accuracy of the signals. This allows the PLC to control a real actuator, while still allowing the Y-Box to monitor the actuator control data. Through this capability, the Y-Box can monitor an actuator and override the control data to simulate events such as an actuator failure. AI modules can also be connected to real sensors that can monitor a physical process. This flexibility allows the Y-Box to be used in a variety of applications.

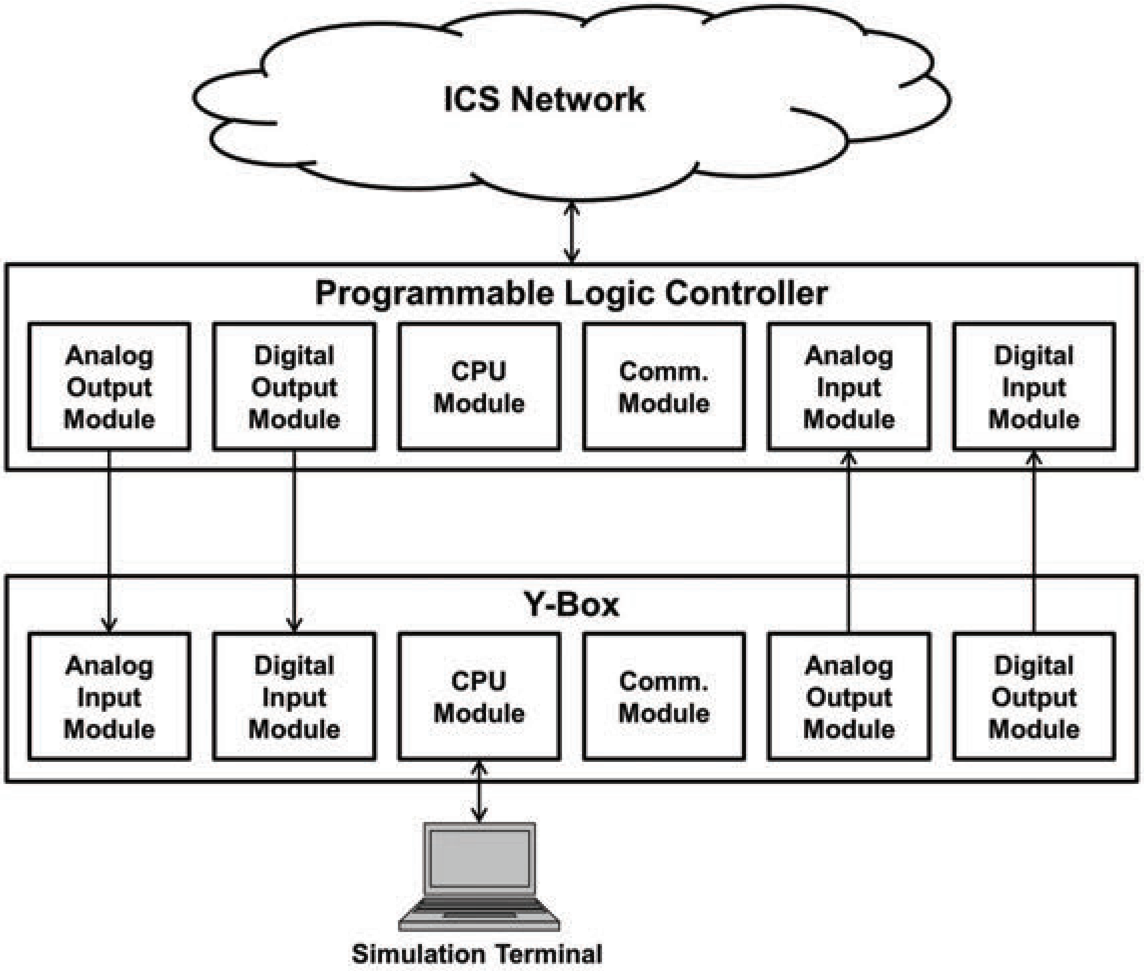


Figure 10. System architecture overview.

3.2.1.2 Digital Input Modules.

DI modules read digital signals generated by a PLC intended to control discrete components in a system (e.g., light bulbs). DI modules support up to seven inputs simultaneously and can be used with 24 VDC signals, which are typically used as high signals for digital modules. The measured data are then retrieved by the CPU module for simulation. Similar to the AI modules, DI modules can be connected in a variety of configurations to ensure flexibility.

3.2.1.3 Analog Output Modules.

AO modules generate analog signals that simulate sensor outputs that are measured by a PLC. AO modules support up to eight outputs and can be configured to generate either voltage or current. The PLC uses this computed sensor data from the simulation to determine the desired actuator control data for the system.

3.2.1.4 Digital Output Modules.

DO modules generate voltage levels that simulate discrete sensors (e.g., level switches) measured by a PLC. Similar to the DI modules, DO modules support up to seven channels and can be used with 24 VDC.

3.2.1.5 CPU Module.

The CPU module is responsible for collecting data from Y-Box input modules and sending data to Y-Box output modules. The CPU module can be connected via USB to a simulation terminal or a communication module. Depending on the connectivity requirements, the CPU module can act as a simple data pass-through, or it can run a simulation program directly. This allows the CPU module to run a simulation while isolated from any simulation terminals. Note that, if the CPU module acts as a pass-through, a simulation terminal must be connected to the CPU module to execute the simulation.

3.2.1.6 Communication Module.

The communication module provides a remote connection between the simulation terminal and the CPU module. A Raspberry Pi is capable of serving as a communication module for the Y-Box as it has built-in USB and Ethernet ports. This type of configuration is shown in Figure 11. The communication module allows the connection of multiple CPU modules in Y-Boxes with a single simulation terminal via Ethernet connection.

3.2.1.7 Simulation Terminal.

The simulation terminal receives actuator control data from the CPU module and uses this data in a simulation program to compute the process values (e.g., water level and flow rate) of the simulated physical system. The process values are used to calculate analog and digital sensor values, which are sent to the Y-Box CPU module for output to a PLC. The simulation program can be written in any language capable of serial or network communication. The simulation terminal connects to the Y-Box CPU module via USB, using an ASCII protocol. The received analog values range

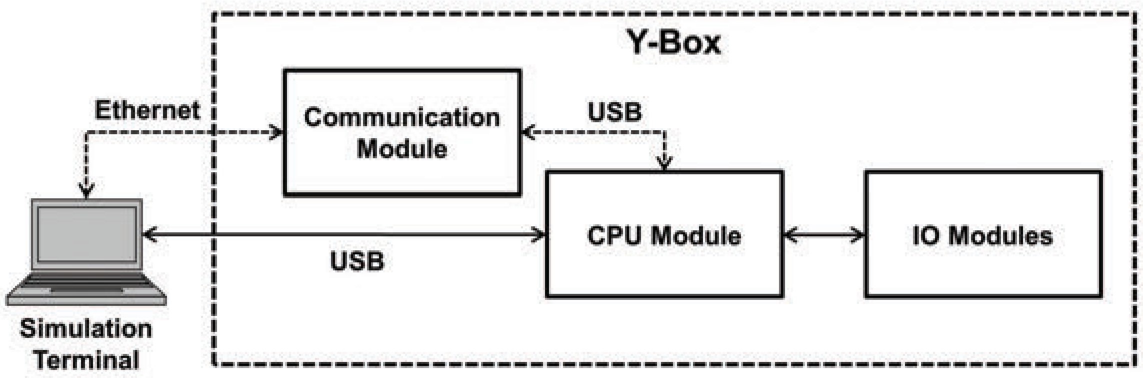


Figure 11. Communication module connection.

from 0 and 4095 (12-bits), and digital values may be 0 or 1 to indicate low or high status, respectively.

3.2.2 Hardware Design.

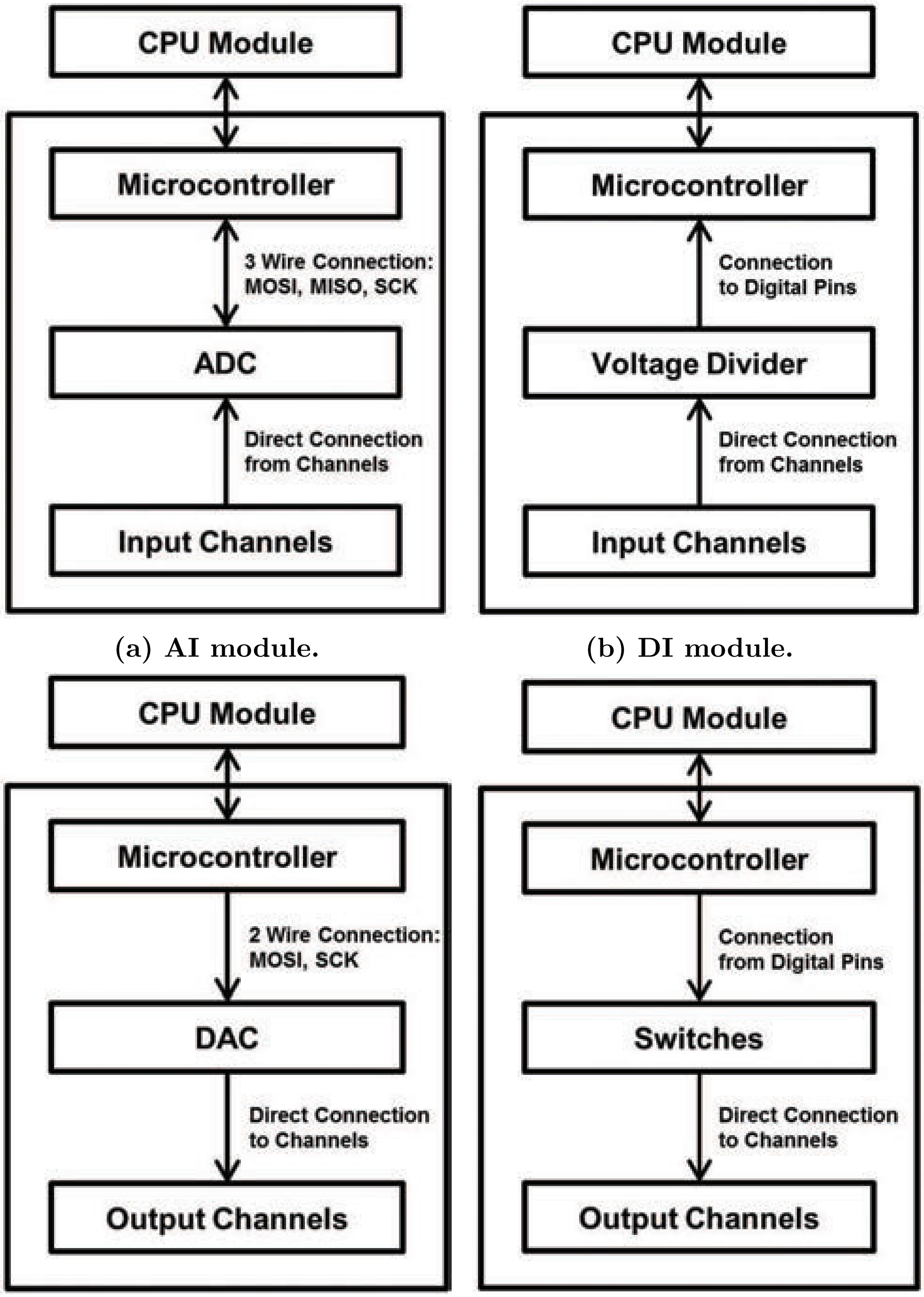
This section discusses the hardware design of the individual Y-Box modules. Each IO module has one ATTiny44A microcontroller to handle communication with the CPU module as shown in Figure 12. Main parts for the modules are listed in Table 2.

3.2.2.1 Analog Input Module.

The AI module supports up to eight inputs. A series of DIP switches are used to individually configure each input for voltage or current. In voltage mode, the input signal is measured directly. In current mode, a precision 250 Ω shunt resistor is used to convert a 0 mA to 20 mA signal to a 0 VDC to 5 VDC signal. For each input, an operation amplifier (opamp) is used to buffer the signal and limit any current leakage. This buffer allows the current from the input to be unaffected should another device need to measure the same signal. In voltage mode, another opamp is used as a voltage divider to reduce the 0 VDC to 10 VDC signal down to 0 VDC to 5 VDC. In current

Table 2. Main Parts

Part Name	Module	Description	Manufacturer
MCP3208	AI	8 ch. 12-bit ADC	Microchip
MCP4902	AO	2 ch. 12-bit DAC	Microchip
ATTiny44A	All IO	Microcontroller	Atmel
Arduino Micro	CPU	Microcontroller (ATmega32U4) Board	Atmel
LM324	AI, AO, DI	Operational Amplifier	TI
74HC138	CPU	Demultiplexer	NXP
XTR111	AO	Voltage to Current Converter	TI
KA7805	All IO	5V Voltage Regulator	Fairchild


CPU Module
Microcontroller
3 Wire Connection: MOSI, MISO, SCK
ADC
Direct Connection from Channels
Input Channels
CPU Module
Microcontroller
Connection to Digital Pins
Voltage Divider
Direct Connection from Channels
Input Channels
CPU Module
Microcontroller
2 Wire Connection: MOSI, SCK
DAC
Direct Connection to Channels
Output Channels
CPU Module
Microcontroller
Connection from Digital Pins
Switches
Direct Connection to Channels
Output Channels

(a) AI module.

(b) DI module.

(c) AO module.

(d) DO module.

Figure 12. IO modules communication flow.

mode, this opamp simply acts as another buffer for the 0 VDC to 5 VDC signal. The divided signal is then sent to an Analog to Digital Converter (ADC) with 12-bit resolution. The ADC converts the 0 VDC to 5 VDC signal into an integer value which is then read by the microcontroller via a serial connection. The microcontroller continuously reads the channel values from the ADC via serial peripheral interface and prepares them for transmission to the CPU module.

3.2.2.2 Digital Input Module.

The DI module supports up to seven inputs. For each input, one opamp is used as a voltage divider to reduce the 0 VDC to 24 VDC signal to 0 VDC to 5 VDC. The reduced signal is then read using a General Purpose IO (GPIO) pin of the microcontroller.

3.2.2.3 Analog Output Module.

The AO module supports up to eight outputs simultaneously. A 12-bit Digital to Analog Converter (DAC) is used to generate a 0 VDC to 5 VDC signal. An opamp is used as a voltage amplifier to raise the voltage to 0 VDC to 10 VDC. This voltage may then be read directly by a PLC analog input. Alternatively, the 0 VDC to 10 VDC signal may be fed into an optional voltage to current converter which converts the signal to current that ranges from 0 mA to 20 mA.

3.2.2.4 Digital Output Module.

The DO module supports up to seven outputs simultaneously. The microcontroller's GPIO pins are used to output a 0 VDC or 5 VDC signal. The 5 VDC signals control NPN and PMOS transistors to output either 0 VDC or 24 VDC.

3.2.2.5 CPU Module.

The CPU module communicates with the IO modules using a Serial Peripheral Interface. The Arduino Micro microcontroller in the CPU module acts as a bus master and connects to the microcontroller in each IO module. The CPU module contains a demultiplexer which is used to select a single IO module for communication. The Arduino Micro's digital pins are used to control the demultiplexer and select the IO module to send information. Each IO module has an input that is used to select that module. The current CPU module supports up to eight connected IO modules simultaneously. The number of modules can be expanded in future versions by simply adding another demultiplexer.

The protocol used to communicate with the IO modules is shown in Table 3. To identify the slot number and type of each attached IO module, the CPU module sends the heartbeat command to each slot. In return, it receives a unique ID for the IO module connected to the selected slot. The heartbeat command is also periodically used to verify the connection with each IO module during operation.

3.2.3 Experiment Design.

The goal of Y-Box is to simulate physical processes by interfacing with PLCs via various types of signals. To accomplish this goal, the Y-Box needs to accurately measure signals generated by a PLC, and it needs to be capable of generating signals that a PLC can measure. To simulate realistic systems, the Y-Box needs to be capable of scaling to interface with multiple PLCs at one time. This evaluation focuses primarily on the accuracy of the Y-Box IO modules.

Table 3. **Communication protocol.**

Command	Byte 1								Byte 2							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Heartbeat	1	X	X	X	X	X	X	X	N/A							
Read AI	0	X	X	X	Channel				N/A							
Read DI	0	0	X	X	Channel				N/A							
Read DI All	0	1	X	X	X	X	X	X	N/A							
Write AO	0	Channel			D											
Write DO	0	0	X	Channel				D	N/A							
Write DO All	0	1	D								X	X	X	X	X	X

X - Don't Care
D - Data
1 - High
0 - Low

3.2.3.1 Approach.

To evaluate the Y-Box AI module, an analog signal is generated by the PLC and measured by the Y-Box. Similarly, to evaluate the Y-Box AO module, an analog signal is generated by the Y-Box and measured by the PLC. The measured values are then compared to the expected values to calculate the percent error. It is important to note that the measured value is actually the error caused by the PLC in addition to the error caused by the Y-Box. These two error sources cannot be readily separated in this experiment.

3.2.3.2 Performance Metric.

The selected metric for the performance test is based on percent error over range of the Y-Box IO modules as shown in Equation 1. The analog values for the Y-Box modules range from 0 to 4095 (12-bit resolution). The digital values for the Y-Box range from 0 to 1 to indicate low or high.

$$Percent\ Error\ Over\ Range\ (\%) = \frac{Abs.\ (Sent\ Value - Received\ Value)}{Max.\ Value - Min.\ Value} \times 100 \quad (1)$$

A similar metric is used by industrial PLC manufacturers such as Rockewell Automation to measure the precision of their equipment [27]. Two types of PLCs, CompactLogix and ControlLogix from Rockwell Automation, are selected to determine the acceptable range of error. CompactLogix has the accuracy of ±0.7% and ±0.6% over the range for AI voltage and current modes, respectively, where its AO has ±0.5% for both modes at room temperature [25]. ControLogix presents module error from ±0.1% to ±0.6% depending on the various modules [26]. Considering the measurements reflecting the errors from both the Y-Box and PLCs, it is considered to be acceptable for the Y-Box to have the percent error over range similar to that of CompactLogix.

3.2.3.3 Parameters.

The parameters listed here could potentially impact the performance of the Y-Box, but are not specifically evaluated in this research.

- Number of IO modules: Although it is not expected to impact the accuracy of Y-Box, the number of Y-Box IO modules connected at one time may potentially impact performance. For this evaluation, one of each type of IO module (AI, AO, DI and DO) is connected to a single Y-Box CPU module.
- Selected IO channels: The selected channel number for the Y-Box and PLC impacts the accuracy if the various channels perform differently. This could be caused by minor hardware differences between the channels. Although not

specifically addressed in this research, the channel numbers for the experiments are randomized to account for these potential errors.

- Setup and hold time refer to the amount of time from when a signal is generated to when it is measured. Pilot studies show that 400 ms is a sufficient delay to allow all signals to reach steady state. This delay is used for all measurements of the system.

3.2.3.4 Calibration.

All analog signals have some level of error created by flaws in the various hardware components. PLCs often use a calibration process to account for these errors and correct the component inaccuracies [27]. Calibration of the Y-Box is conducted during the experiments to account for the margin of error in the hardware components. The calibration values for each channel in the analog IO modules are measured and recorded. A precision voltage source and precision voltage meter are used to measure the calibration values. The raw measurements from each experiment are recorded, and the calibration is applied as a post-processing step using Equations 3 and 2. The calibration impact is discussed in Section 4.2.1.

$$\begin{array}{c} Calibrated\ Data\ from \\ AI\ Module \end{array} = \frac{Received\ Value\ by\ Y{-}Box\ (0\ to\ 4095)}{Calibration\ Value\ (V\ or\ I)} \times 4095 \tag{2}$$

$$\begin{array}{c} Calibrated\ Data\ from \\ AO\ Module \end{array} = \frac{Received\ Value\ by\ PLC\ (0\ to\ 4095)}{Calibration\ Value\ (V\ or\ I)} \times 10\ V\ or\ 20\ mA \tag{3}$$

3.2.3.5 PLCs.

This evaluation utilizes two Allen Bradley PLCs to measure the performance of the Y-Box. The first PLC is a ControlLogix PLC with the following modules: 1756-L61, 1756-ENBT, 1756-OF4, 1756-OB8, 1756-IF16 and 1756-IB16. These modules provide the ControlLogix with one of each type of IO (AI, AO, DI, DO). The second PLC is a CompactLogix 1769-L23E-QBFC1. This CompactLogix PLC also has each type of IO. The ControlLogix used for evaluation offers 16-bit for AI current and voltage modes, 16-bit for AO voltage mode and 15-bit for AO current mode [27]. CompactLogix offers 8-bit for all data types [24]. These two PLCs are used to evaluate the accuracy of Y-Box IO modules; future work will evaluate the Y-Box with more types of PLCs.

Two configurations are used to evaluate the performance of the Y-Box with the PLCs. The Y-Box is first evaluated with just the ControlLogix PLC connected and then again with both PLCs connected. The accuracy achieved with one PLC connected is compared to the performance with two PLCs connected. This determines if having two PLCs connected at one time impacts the performance. Also, the performance of the Y-Box with the ControlLogix is compared to the performance with the CompactLogix. This helps show the difference in performance between PLCs with varying resolution.

For each configuration, the input and output channels are randomly selected for both the PLCs and Y-Box. It is important to note that the selected ControlLogix channels are randomly changed for the second configuration.

3.2.3.6 Analog Values.

The amplitude of the analog signals is expected to impact the accuracy of the measurements. To evaluate this impact, five analog values are used: 0, 1024, 2048, 3072 and 4095. In current mode, these values represent 0 mA, 5 mA, 10 mA, 15 mA and 20 mA, respectively. In voltage mode, these values represent 0 V, 2.5 V, 5 V, 7.5 V and 10 V, respectively. This provides a range of values to determine the accuracy of the Y-Box.

3.2.3.7 Digital Values.

All digital values are measured as high or low.

3.2.3.8 Experiment Set-up.

The experiment set-up for evaluation is shown in Figure 13. The data collector residing in the simulation terminal interacts with CPU modules of PLCs and the Y-Box to send and receive the necessary signals for the experiments. IO modules in PLCs and Y-Box are interconnected to send and receive the signals processed by the CPU modules. Below is the process used to collect data from PLC or Y-Box in sequential steps. For analog modules, 50 measurements per channel per value for 5 different values (e.g., 0, 1024, 2048, 3072 and 4095) are collected to provide sufficient data for analysis. For digital modules, 50 measurements per channel per value for 2 different values (e.g., high and low) are collected to provide sufficient data for analysis.

1. Determine parameters for the next measurement.

2. Write an analog or digital values to either the PLC or Y-Box.

3. Wait 400 milliseconds.

4. Read measured value from either the PLC or Y-Box.

5. Record measurement and expected value.

6. Wait 100 milliseconds.

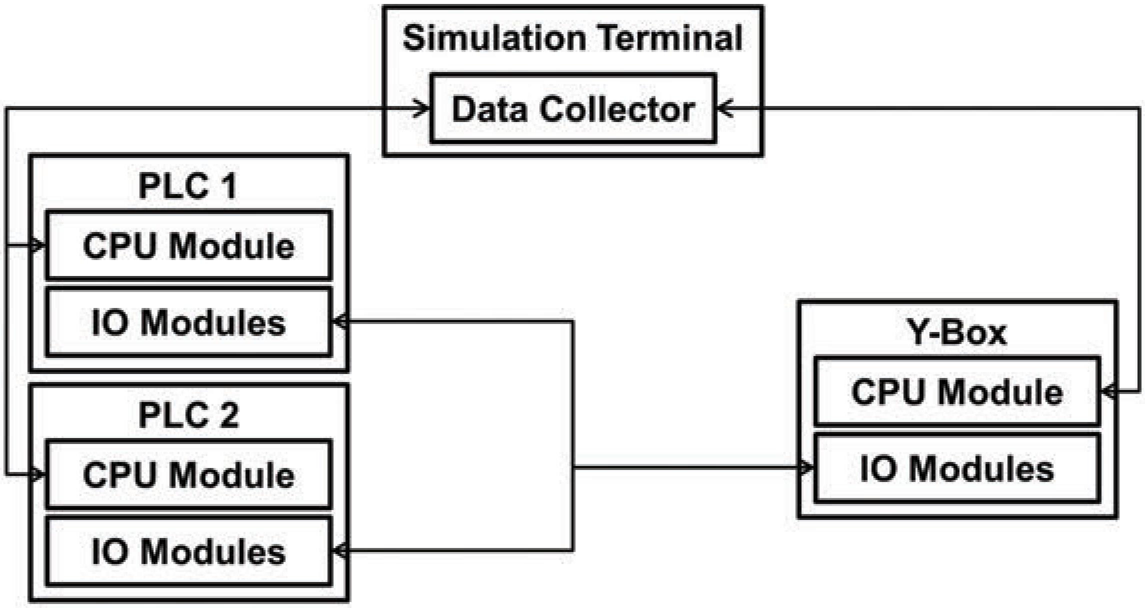


Figure 13. Experiment set-up.

3.2.4 Applications.

This section describes how the Y-Box simulates the physical processes controlled by PLCs in two scenarios. The Y-Box physically connects to the PLCs (ControlLogix and CompactLogix), an exploded view is offered in Figure 14. The implementation for the scenarios follows the simulation process shown in Figure 9.

3.2.4.1 Implementation of the First Scenario.

The first scenario simulates the bar screen stage in a WWTP. Bar screens are typically the first step of processing wastewater in a WWTP to mechanically filter out large objects (e.g., bulky solids, rags and plastics) [1]. Bar screens help reduce clogging and damage of valves and pumps. The first seven steps in Figure 15 illustrate the sequential steps of the bar screen stage. The boxes with sharp corners are part of the PLC process, while the boxes with soft corners are for the Y-Box. It also shows the dependencies represented in the directional arrows. The arrows illustrate the flow

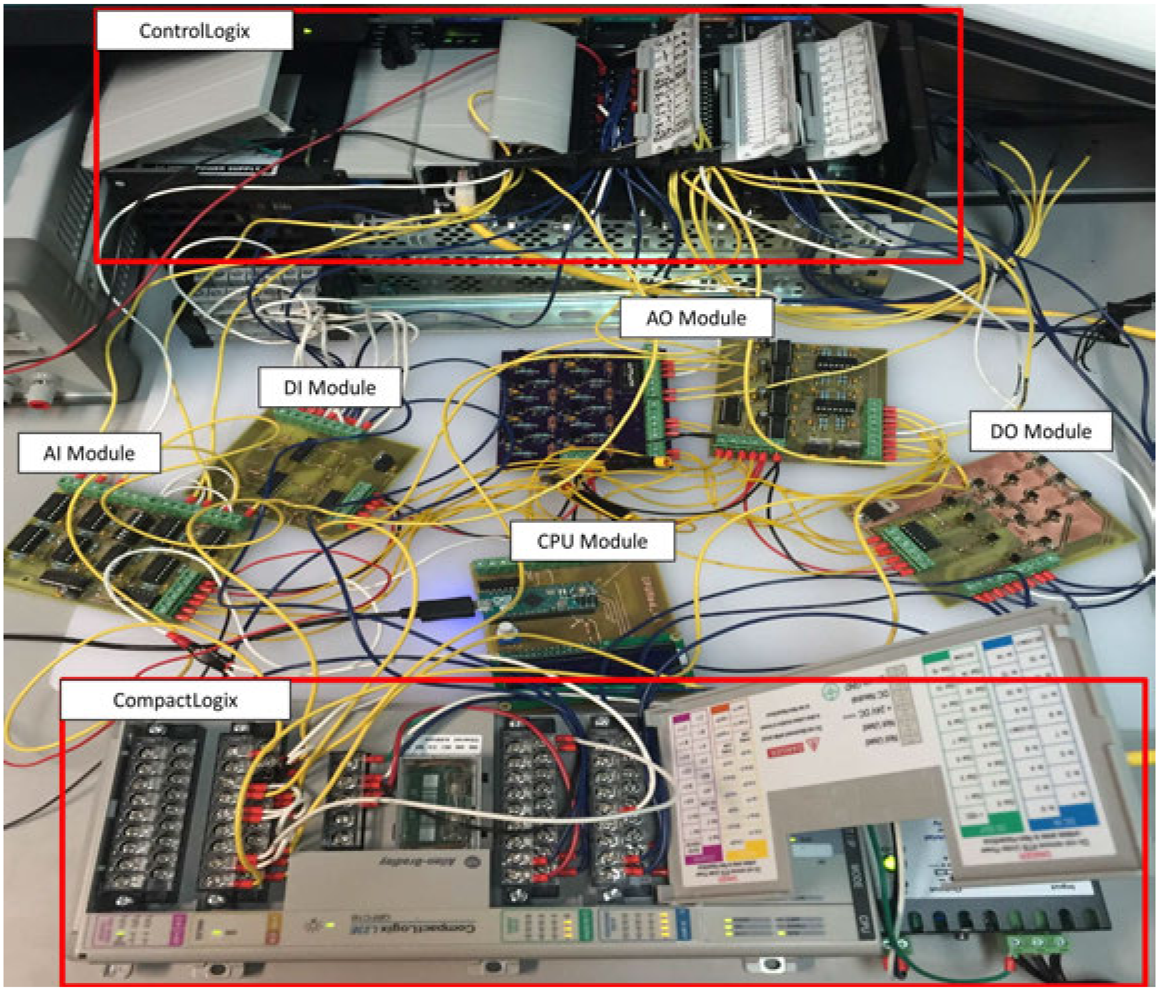


Figure 14. Physical setup for scenarios.

of required data to compute process data for the sensor values and to determine the actuator control data.

- Initial States: Initial states of wet well, stage 1, stage 2 and clog level are required to give the starting points of the intended simulation. Wet well acts as a reservoir of wastewater waiting to enter the bar screens stage. Stage 1 is the water level in a bar screen stage before the filtering process (stage 2). The initial states seen in Figure 16a are defined once at the start of simulation.

- Influent Rate: Influent rate is the user defined value that can be adjusted throughout the simulation as seen in Figure 16a. Influent rate determines the amount of wastewater entering from wet well to stage 1 in the bar screen stage. An analog output of the Y-Box is used to send it to the PLC (Figure 15).

- Wet Well Level: Wet well level depends on the influent and pump rate. Wet well level goes up by the amount of influent and down by the amount being pumped out. It has two digital sensors to indicate low and high levels of wastewater. Two digital outputs of the Y-Box are used to send the levels of wet well to PLC 1 (Figure 15).

- Pump: Pump rate is determined within PLC 1 based on the influent rate and the water levels in the wet well and stage 1. For example, pump rate matches influent rate when wet well level is high and stage 1 level is low. On the other hand, pump stops regardless of influent rate when wet well level is low and stage 1 level high. An analog output of PLC 1 is used to send the pump rate data to the Y-Box (Figure 15).

- Stage 1 Level: Stage 1 level is determined by the Y-Box based on the pump rate and clog level of the bar screen. The clog level indicates the amount of material built up in the bar screen. The clog level goes up when the bar screen collects

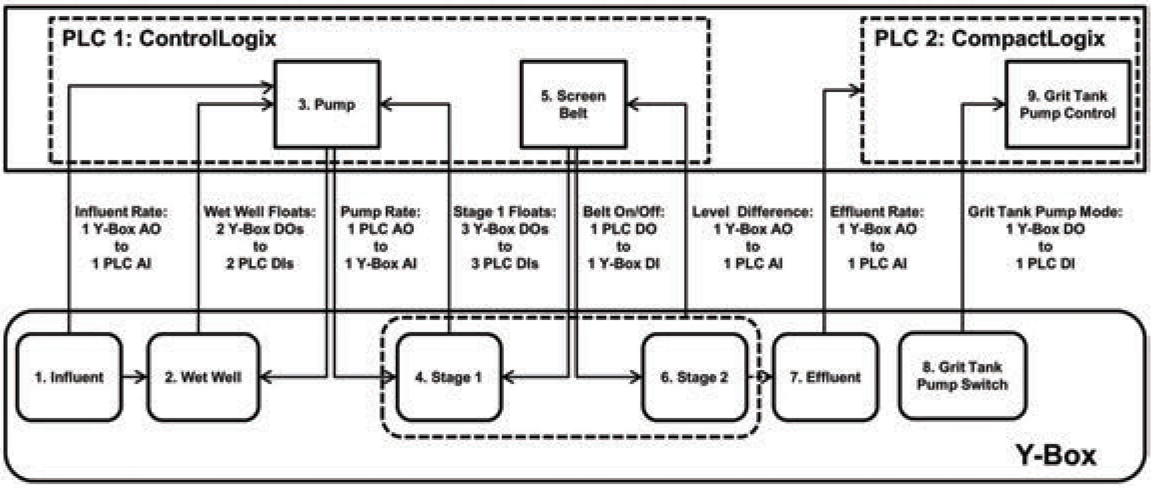

PLC 1: ControlLogix
3. Pump
5. Screen Belt
PLC 2: CompactLogix
9. Grit Tank Pump Control
Influent Rate: 1 Y-Box AO to 1 PLC AI
Wet Well Floats: 2 Y-Box DOs to 2 PLC DIs
Pump Rate: 1 PLC AO to 1 Y-Box AI
Stage 1 Floats: 3 Y-Box DOs to 3 PLC DIs
Belt On/Off: 1 PLC DO to 1 Y-Box DI
Level Difference: 1 Y-Box AO to 1 PLC AI
Effluent Rate: 1 Y-Box AO to 1 PLC AI
Grit Tank Pump Mode: 1 Y-Box DO to 1 PLC DI
1. Influent
2. Wet Well
4. Stage 1
6. Stage 2
7. Effluent
8. Grit Tank Pump Switch
Y-Box

Figure 15. Scenario 1 and 2 sequential steps with dependencies.

more objects depending on the filth rate of wastewater or down when the bar screen is cleaned. The bar screen is cleaned according to the clog clean rate when the bar screen belt is on. The clog clean rate and filth rate are defined as seen in Figure 16a and can be adjusted throughout the simulation. The clog in the bar screen blocks, with varying degree, the water transfer from stage 1 resulting in the rise of water level. High pump rate and clog level would make stage 1 level rise faster. Stage 1 has three digital sensors to indicate the water levels (Low, High, High High). Three digital outputs of the Y-Box are used to send the levels of stage 1 to PLC 1 (Figure 15).

- Bar Screen Belt: Bar screen belt operation includes two modes (i.e., automatic and running). The automatic mode turns the bar screen belt on and off periodically. The running mode keeps the bar screen belt running when the level difference between stage 1 and 2 exceeds a set point defined in PLC 1. Y-Box sends the stage 1 and 2 level difference to PLC 1 via an analog output. Once the mode is determined, a digital output of PLC 1 is used to send it to the Y-Box (Figure 15).

- Stage 2 and Effluent rate: Stage 2 releases the transferred water from stage 1 when it exceeds the amount stage 2 can hold. The amount of water that stage 2 can hold is defined before the simulation. Effluent rate is determined by the amount of water stage 2 releases. An analog output of the Y-Box is used to send it to PLC 2 (Figure 15). PLC 2 measures the amount of flow into the grit tank from the effluent rate.

3.2.4.2 Implementation of the Second Scenario.

The second scenario is the extension of first scenario. It adds one more stage, grit tank, to the first scenario with the addition of a PLC (Allen Bradley, CompactLogix L23E) (Figure 15). The grit tank purpose is to remove smaller objects (e.g., sand, broken glass, silt and pebbles) that may damage pumps and other mechanical devices [2]. The last two steps in Figure 15 illustrate the sequential steps of grit tank stage.

- Grit Tank Pump Switch: The grit tank pump switch in the Y-Box acts as a manual override for its operation. It is defined as seen in Figure 16a and can be toggled between automatic and running mode. The modes are similar to those in the bar screen belt. The automatic mode turns the grit tank pump on and off periodically according to the control data sent by PLC 2. The running mode simply keeps the pump running. A digital output of the Y-Box is used to send the manual override data to PLC 2 (Figure 15).

- Grit Tank Pump Control: While PLC 2 controls the pump operation when the switch is set to the automatic mode, its command is superseded when the switch is set to running mode.

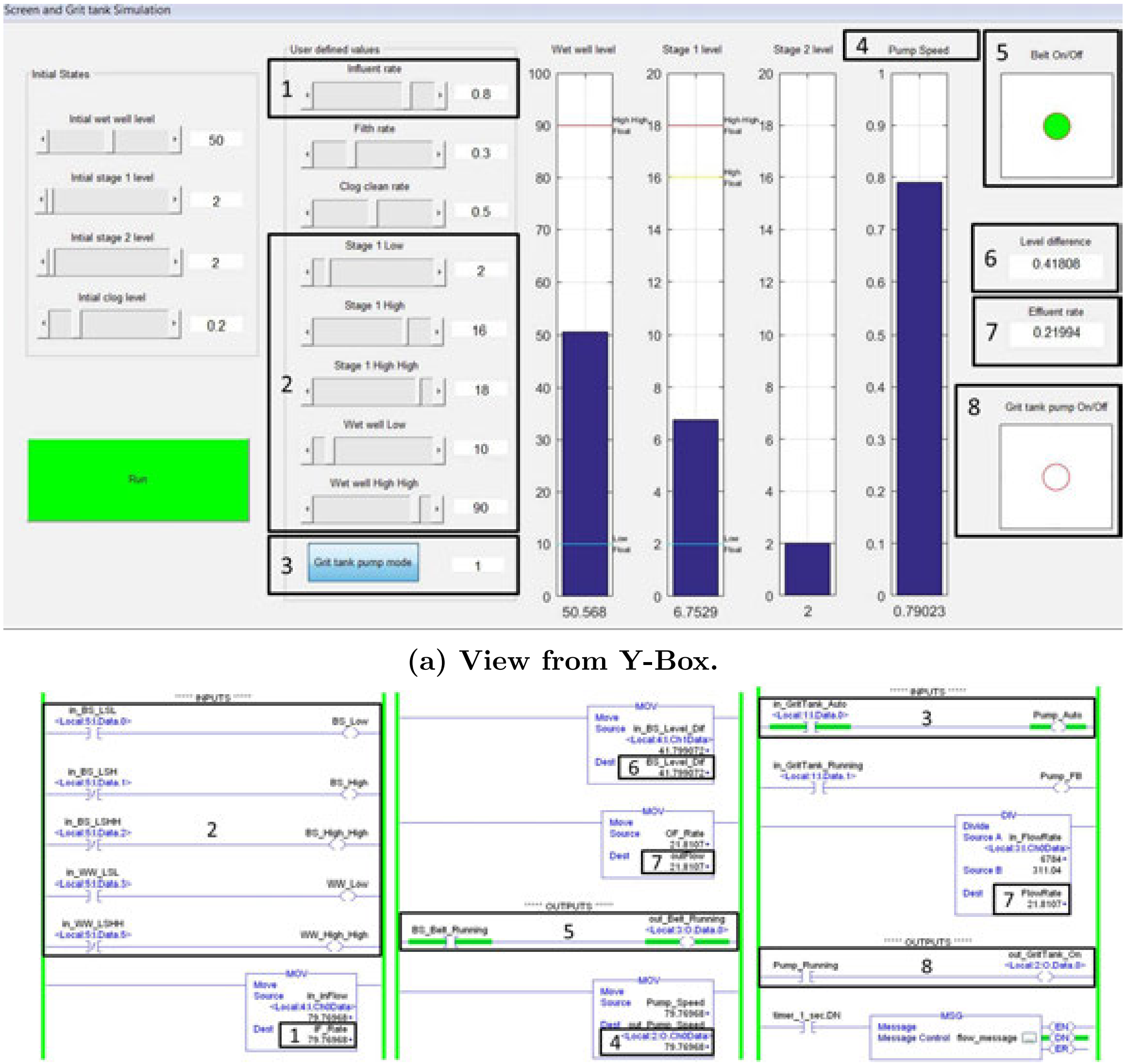
3.2.4.3 PLC and Y-Box Views of the Scenarios.

Under normal operation, it is essential that the views from PLCs and the Y-Box are nearly equivalent. When the run button is pressed, the simulation follows the loop process as shown in the Figure 9. Figures 16a, 16b and 16c are captured simultaneously during the simulation and show the views from PLCs and the Y-Box. Initial states in Figure 16a are defined once before the simulation starts. The user defined values in Figure 16a can be adjusted throughout the simulation. Box 2 (in the user defined values panel) indicate the levels of floats in the wet well and stage 1 of bar screen stage. The floats turn on when its water level is higher (for High or High High float) or lower (for Low float) than the defined set points. Figure 16a displays ∼50 and ∼7 for wet well and stage 1 levels, respectively, not tripping any floats. Box 2 from Figure 16b shows that the digital inputs of PLC 1 match the Y-Box view as seen in Figure 16a. The number 1 in Box 3 from Figure 16a indicates auto mode, matching what is shown in PLC 2 (Figure 16c). Box 5 and 8 in Figure 16a also match with the status shown in Box 5 in Figure 16b and Box 8 in Figure 16c. For the analog values in Box 1, 4, 6 and 7, the values between PLCs and Y-Box closely match within the Y-Box performance error when the PLC percent scale is converted to Y-Box 0 to 1 scale. The exact match of values in Box 7 in Figure 16b and Figure 16c show that there is a successful communication between PLC 1 and PLC 2 for data exchanges.

Table 4 shows the comparison of analog and digital values in the corresponding boxes from the simulation and PLCs views.

Table 4. Comparison of the readings from the views of simulation and PLCs

Box Number	1	2	3	4	5	6	7	8
Simulation	80%	All off	On	79.023%	On	41.808%	21.994%	Off
PlCs	79.769%	All off	On	79.769%	On	47.799%	21.811%	Off



Screen and Grit tank Simulation

Initial States

Intial wet well level 50

Intial stage 1 level 2

Intial stage 2 level 2

Intial clog level 0.2

User defined values

Influent rate 0.8

Filth rate 0.3

Clog clean rate 0.5

Stage 1 Low 2

Stage 1 High 16

Stage 1 High High 18

Wet well Low 10

Wet well High High 90

Grit tank pump mode 1

Level difference 0.41808

Effluent rate 0.21994

(a) View from Y-Box.

(b) View from PLC 1.

(c) View from PLC 2.

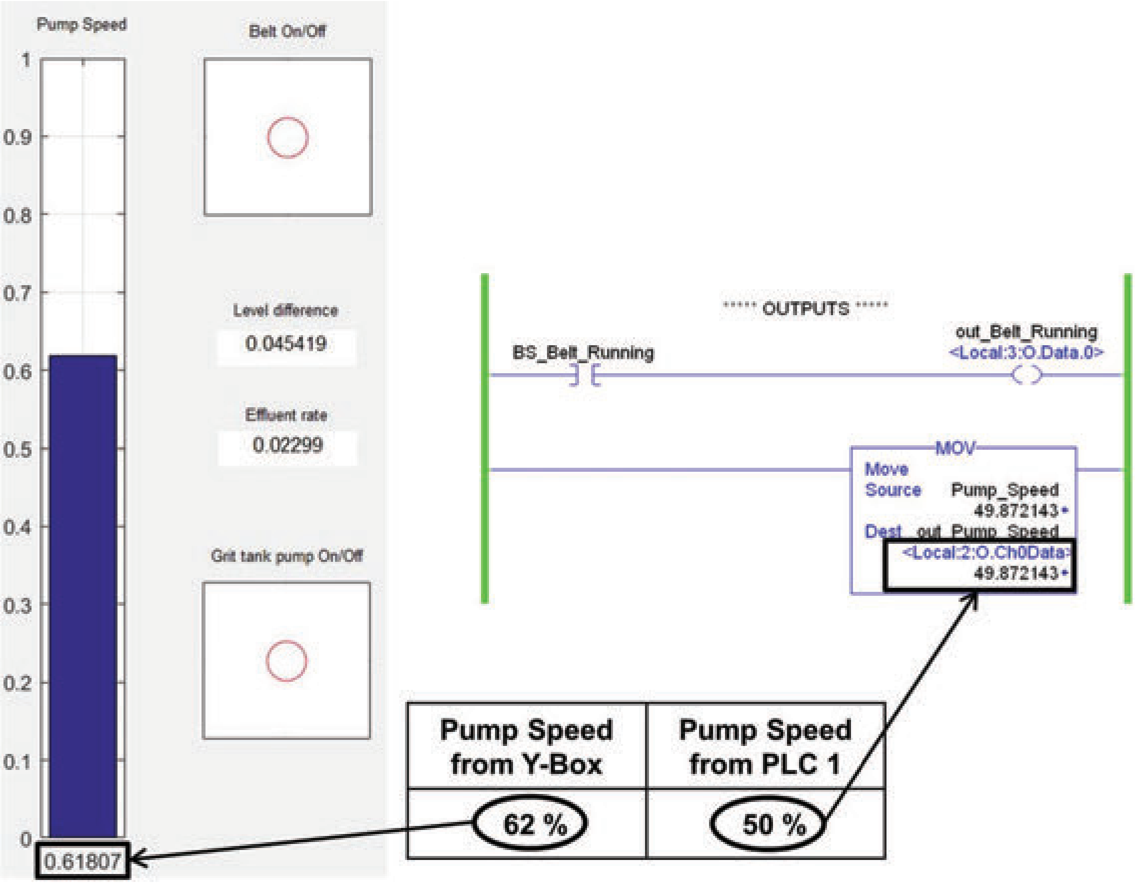
Figure 16. Y-Box View vs. PLCs View.

3.2.4.4 Attacks for Scenarios.

Moreover, a couple of potential attacks, utilizing views from PLCs and the Y-Box are developed within the simulation scenarios to demonstrate the utility of the Y-Box. When the attacks are used in an exercise, the ICS operators have the PLC views where the exercise controllers have both views from PLCs and Y-Box. This way, the exercise controller can tell immediately when the attackers maliciously modify the HMI while observing the responses from the ICS operators.

- **Attack 1 for Scenario 1**: Attack 1 is a configuration modification attack against PLC 1. PLC 1 includes an automatic scaling feature for configuration of analog inputs and outputs. Once the scaling values are set by an engineer, the true analog voltage or current level is hidden from the engineers and the operators. The only way to detect this modification is to manually examine the configuration of the PLC or to physically examine the process characteristics. For this attack, the original scaling range of the analog output that controls the pump is changed from 0% - 100% to 0% - 80%. This causes an intended control value of 50% to actually set the pump to 62%, causing the pump to run harder than intended. Figure 17 shows theimpact of the configuration modification. The left side shows the Y-Box view of the real actuator value, while the right side shows the PLC view. This type of attack could be used to interfere with the physical processes of an ICS or to damage actuators.

- **Attack 2 for Scenario 2**: Attack 2 utilizes a ladder logic modification to achieve a similar result on PLC 2. The ladder logic modification causes the input read by PLC 2 for the flow rate into the grit tank to scale incorrectly. This causes the PLC to think that less water is flowing into the grit tank than there really is. The modification is transparent to the operators, and the only way to detect this change is to manually examine the ladder logic. Figure 18



Level difference: 0.045419

Effluent rate: 0.02299

Source: Pump_Speed 49.872143

Pump Speed from Y-Box	Pump Speed from PLC 1
62 %	50 %

Figure 17. Attack 1 example.

shows the difference between the PLC value and the true value of the flow rate. This type of attack can be used to interfere a physical process or to influence billing and accounting data.

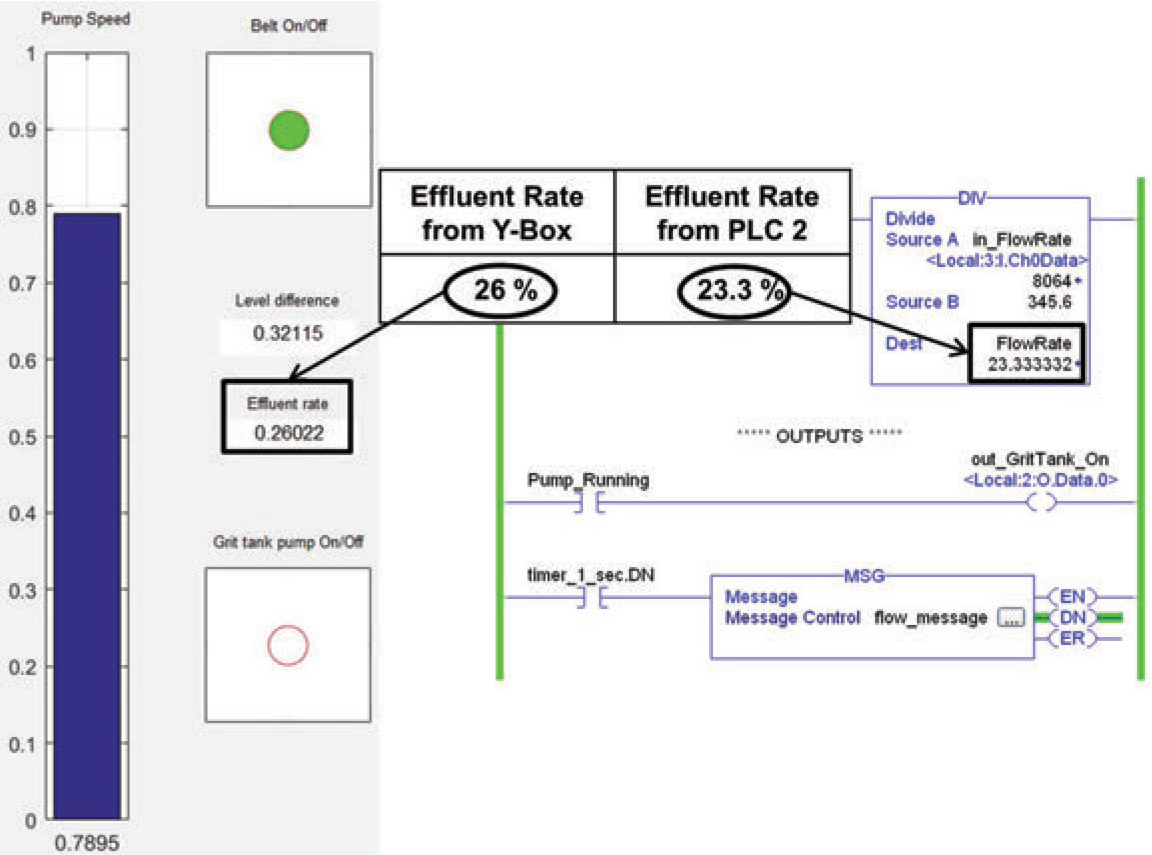


Figure 18. Attack 2 example.

IV. Results and Analysis

The evaluation criteria, environment and scenarios in Section 3.1 are successfully employed during a military cyber exercise, proving the practicality of the framework for evaluating the readiness of cyber first responders. The results from the Y-Box performance test and applications are positive as well. Section 4.1 discusses the results of framework evaluation. Section 4.2 discusses the results of performance test and applications of the Y-Box.

4.1 Framework Evaluation Results

In each scenario, the cyber professional team was evaluated against the set criteria that was derived from the NFPA 1410 concept. The evaluation team was able to determine the effectiveness of the cyber professional team against a measurable set of criteria. Based on the results of the exercise using the NFPA 1410 concept, the evaluation team was allowed to:

- Evaluate the readiness of the cyber professional team to respond effectively to cyber attacks.
- Determine deficiencies in the cyber professional team's ability to identify and mitigate effects from the cyber attacks.
- Determine the ability of the cyber professional team to minimize the cyber attack effects on the physical processes.
- Identify and map shortfalls in cyber professional team responses to both training deficiencies and capability deficiencies.
- Provide focused feedback to the cyber professional team on effective tactics and actions that were not effective.

- Identify the key indicators used by the cyber professional team to respond to the attacks.
- Evaluate the utility and effectiveness of the standard operating procedures, response action plans and cyber defensive tactics, techniques and procedures manual.

Use of the NFPA 1410 concept was critical for identifying the readiness of the cyber professional team. Evaluations in previous exercises were not able to tie team actions directly to requirements and specified evaluation criteria. One of the key advantages was that the formal process helped drive scenario creation and focus evaluation on skill sets and actions that cyber first responders would face during an actual event. Moreover, the results support further exploitation of the custom application for future simulation of environments.

4.1.1 Recommendations.

Although the initial findings for evaluating cyber first responders using the NFPA 1410 concept is positive, some key challenges were identified during the training exercise. The first major challenge is the incorporation of an environment that is adequate for training evaluation. A training environment is needed that can be readily configured to match operational parameters and provide the capability of evaluating response actions by the cyber first responders. Additionally, organizations must dedicate resources to identify evaluators and assessment teams that are capable of implementing the scenarios and exploiting the environment. Finally, the development of common scenarios similar to the scenarios in NFPA 1410 is needed.

4.1.2 Limitations in Hardware.

Arduino UNO was able to meet the requirements to simulate a simple physical process with two different views for the defenders and the coordinators, respectively, however, some limitations are discovered. First, its inputs and outputs are not modular, creating difficulty to expand to accommodate specific number of inputs and outputs from PLCs. Only way would be to increase the number of Arduino UNO. Second, it is not entirely compatible with typical signal types used in ICS, operating only in voltage mode. These limitations are heavily considered for the development of the Y-Box so that it would enhance the framework.

4.2 Y-Box Results

The Y-Box performance test focuses on its accuracy with PLCs. Its applications focus on the ability to demonstrate representational, realistic and evaluation-friendly physical processes simulation.

4.2.1 Performance Test.

The experiment measurements are processed to discuss Y-Box performance impacted by calibration, number and types of PLCs connected and analog values. Table 5 describes the overall performance of the Y-Box. In Figures 19, 20, 21 and 22, the line in the box indicates median while the lower edge and upper edge of the box indicate 1st and 3rd quartiles respectively. The whiskers are considered as boundaries for the most extreme points. The + symbols are outliers.

4.2.1.1 Calibration Impact.

This section evaluates the calibration impact on the performance of the Y-Box. Uncalibrated and calibrated measurements are drawn in boxplots to show distribution over the performance metric (percent error over range). In both AI and AO, (in Figures 19a and 19b), the calibrated data show noticeable improvement with a tighter group closer to 0% error over range than the uncalibrated data. The ideal case is 100% accuracy with the percent error over range of 0% for all measurements. For all further results, the calibrated data are used to partially compensate for the inaccuracies caused by the flaws in the hardware components.

4.2.1.2 Multiple PLCs.

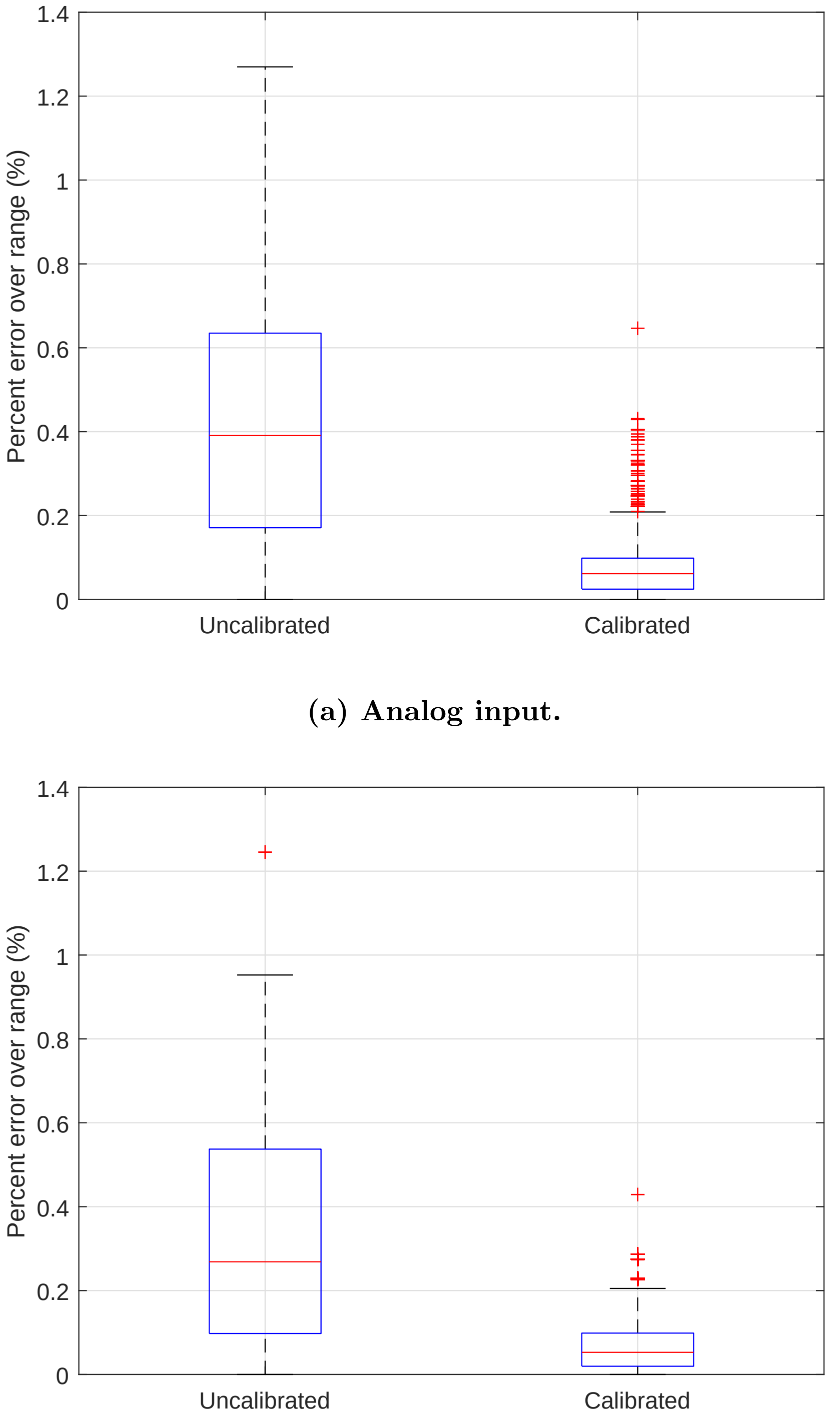
Figure 20 compares Y-Box performance with ControlLogix from the two configurations in Section 3.2.3.5. It is speculated that another power source from CompactLogix in the second configuration adds noise to the circuits, potentially affecting the data measurements for accuracy. Separation of digital and analog grounds in Y-Box analog modules may be required to account for the difference in performance. For all further results, measurement data from the second configuration are used to depict the Y-Box performance.

4.2.1.3 Different PLCs.

Figure 21 compares the Y-Box performance by the types of connected PLCs. Data for the comparison are collected from the second configuration described in Section 3.2.3.5. One of the major differences between ControlLogix and CompactLogix is the resolution to detect data changes for their analog modules, possibly explaining the difference of Y-Box performance with each PLC.

4.2.1.4 Analog Amplitude.

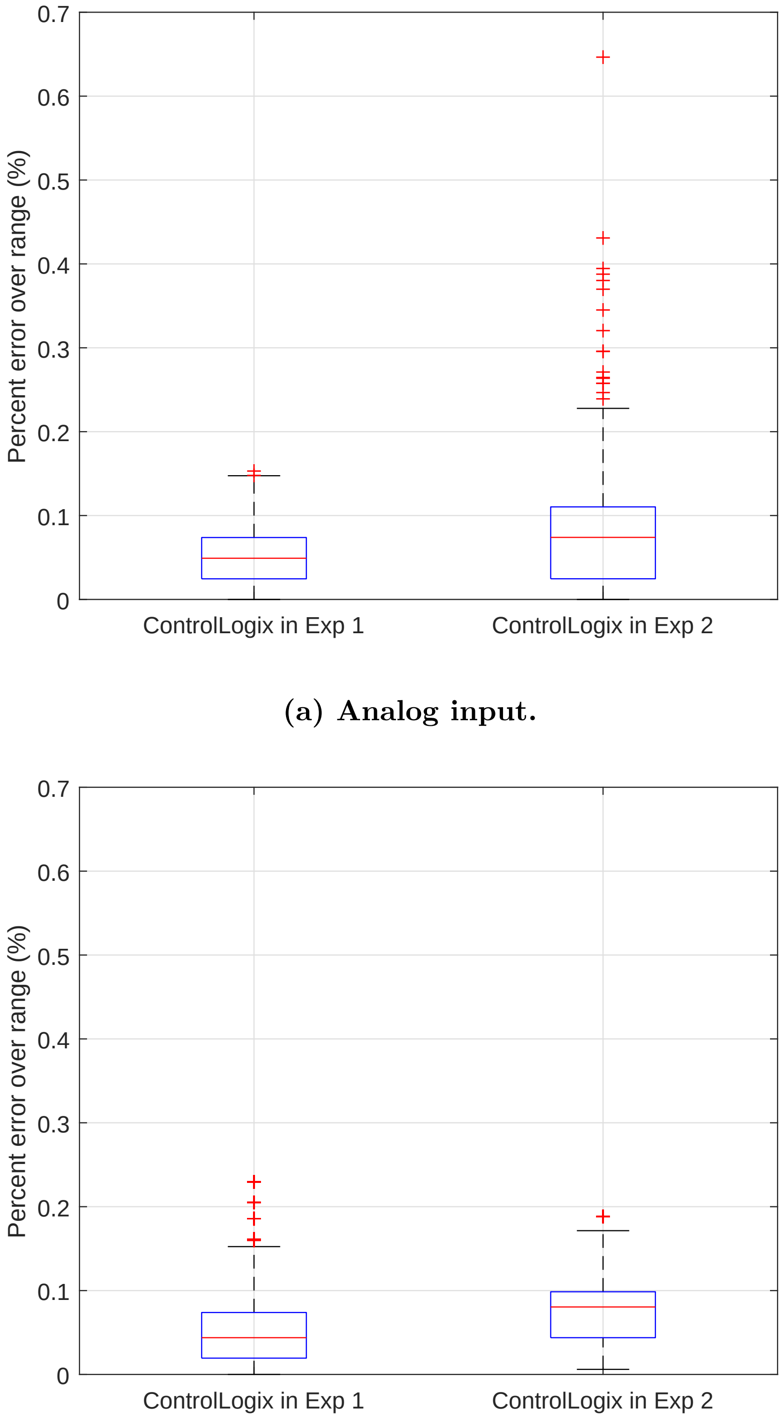
Figure 22 shows the Y-Box performance across its range with varying analog amplitude. Data for this analysis are also collected from the second configuration



(a) Analog input.

(b) Analog output.

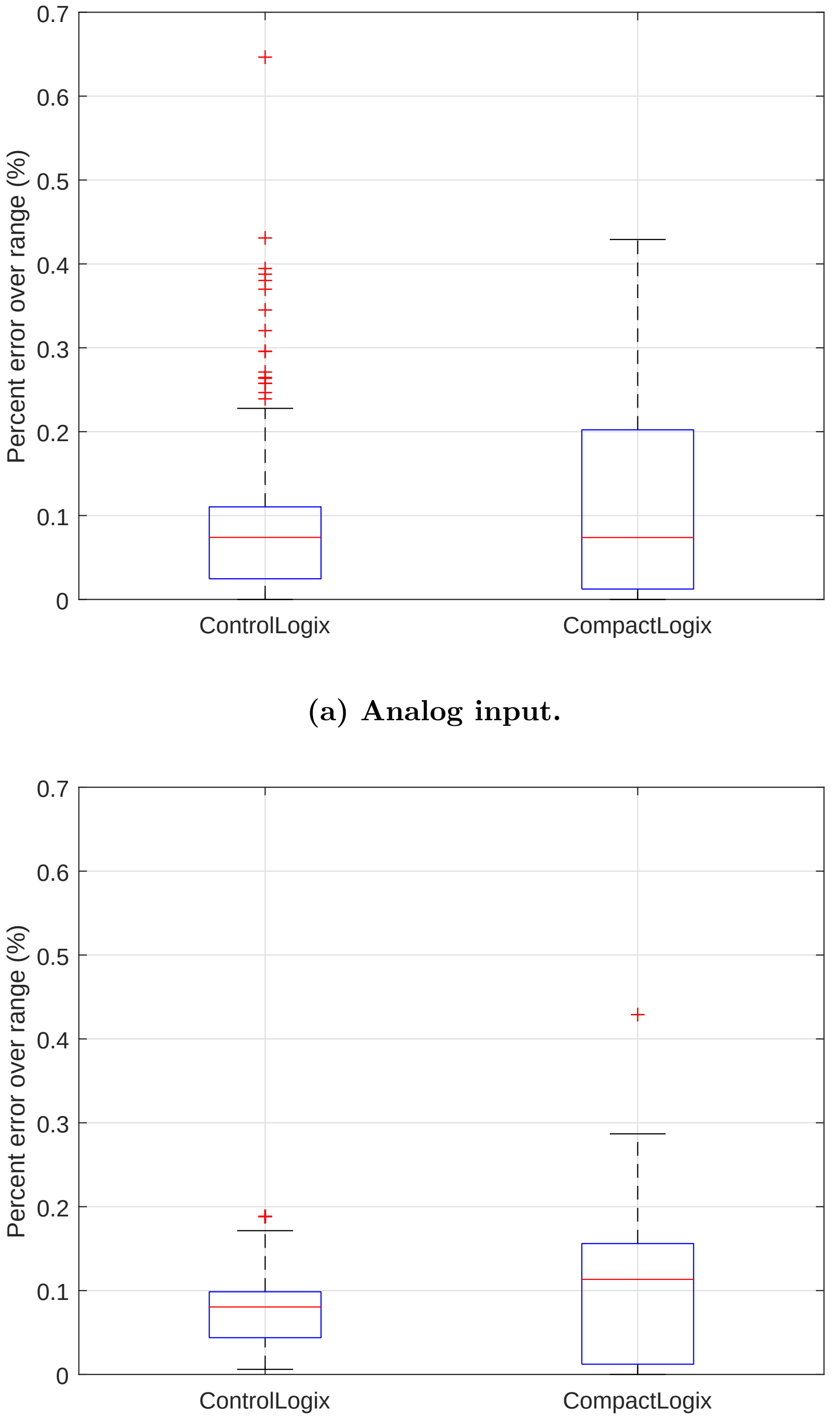
Figure 19. Comparison between calibrated and uncalibrated data.



(a) Analog input.

(b) Analog output.

Figure 20. Comparison of Y-Box performance with ControlLogix in experiment 1 and 2.



(a) Analog input.

(b) Analog output.

Figure 21. Comparison of Y-Box performance with ControlLogix and CompactLogix.

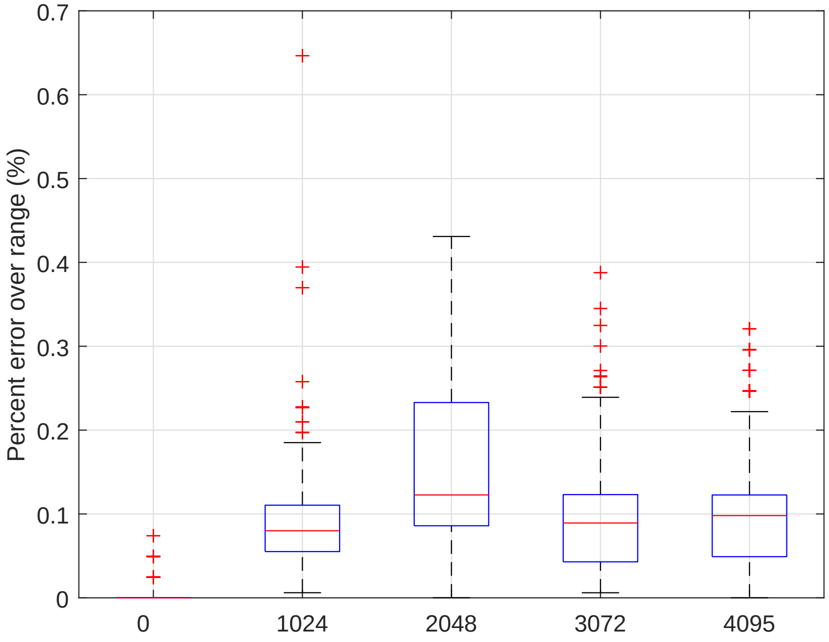
described in Section 3.2.3.5. The measurements are sorted according to the five representational values introduced in Section 3.2.3.6. All measurements except one in Figures 22a and 22b are within ±0.5% error without an exceptional deviation from each other. The vast majority of measurements in the experiment are within ±0.25% of percent error. This points to consistent Y-Box performance regardless of analog amplitude for input and output.

4.2.1.5 Overall Performance (Analog Modules).

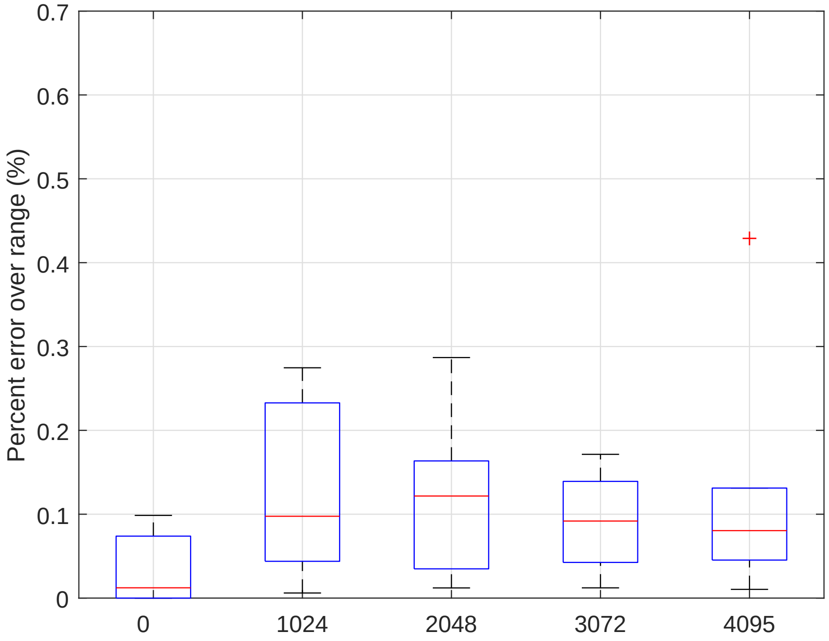
Table 5 shows mean, standard deviation and minimum and maximum percent error. Table 5 shows the calibrated data from the second configuration as discussed. Y-Box AI and AO with ControlLogix has the lower mean and standard deviation than with CompactLogix, suggesting Y-Box performance on accuracy and consistency varies with respect to the types of PLCs. All maximum values that illustrate the Y-Box worst performance are measured well within ±0.7%. The minimum values that illustrate the Y-Box best performance is as low as 0%. While all measurements range from 0% to ±0.7%, the means for Y-Box AI and AO are less than ±0.1%, significantly below the acceptable percent error over range proposed in Section 3.2.3.2.

Table 5. Overall system performance.

		Mean (%)	**SD (%)**	**Min (%)**	**Max (%)**
Y-Box AIs	**ControlLogix**	0.076	0.061	0.000	0.647
	CompactLogix	0.113	0.113	0.000	0.429
	Combined	0.088	0.084	0.000	0.647
Y-Box AOs	**ControlLogix**	0.073	0.044	0.006	0.188
	CompactLogix	0.101	0.095	0.000	0.429
	Combined	0.087	0.075	0.000	0.429



(a) Analog input.



(b) Analog output.

Figure 22. Comparison of Y-Box performance with different analog amplitudes.

4.2.1.6 Digital Performance.

As expected, all digital values between the digital modules of Y-Box and PLC match with 100% accuracy, eliminating the need for further analysis.

4.2.2 Applications.

The Y-Box successfully demonstrats the multiple stages of WWTP by accurately representing the dependencies between the simulated sensors and actuators. The dependencies are validated by comparing the simulation with the expected behaviors during the repeated simulation runs. The dependencies are easily adjustable if required to make the simulation more representational and realistic. As seen in Section 3.2.4.3, the nearly identical views under the normal operation confirm the exchange of signals with negligible margin of error. In addition, a couple of attack examples in Section 3.2.4.4 demonstrate that the simulation with the attack scenarios can be incorporated into a cyber exercise immediately. The similar type of attack was performed as described in Section 3.1.3 and proven effective during the military cyber exercise.

V. Conclusion

This chapter summarizes overall conclusions of the previous chapters.

5.1 Conclusions of Research

This research demonstrates the utility of applying the NFPA 1410 concept for evaluating cyber first responders. Findings from the cyber training exercise demonstrate that the NFPA 1410 concept enhances the ability to evaluate the readiness of cyber first responders against real-world scenarios. The results show that the framework provides a means to adequately identify deficiencies in cyber first responders' ability to identify and mitigate attacks, identify key indicators used to respond to attacks, and provide feedback to enhance response action and training.

Through the applications, the Y-Box demonstrates its ability to simulate the complex physical processes by interacting with various types of signals from multiple PLCs. The performance test of Y-Box demonstrates its precise exchanges of signals with PLCs, which are required to correctly simulate the physical processes and give intended signals back to PLCs. Utilizing the Y-box simulation terminal, the physically observable characteristics are effectively visualized, separate from HMI, to enhance the evaluation mechanism. Affordable cost to build the Y-Box with its small footprint demonstrates it is relatively free of resources and geographic constraints.

5.2 Research Hypothesis.

The successful evaluation of cyber first responders in an exercise confirms the hypotheses that evaluation method can be developed from the one used for first responders (e.g., firefighters) and evaluation can be conducted in a simulated environment. Through the performance test and applications, the Y-Box confirms the hypothesis that the simulation tool can interface physical signals, typically used for industrial applications, from multiple PLCs of different kinds. The Y-Box applications confirm the hypothesis that physically observable characteristics from the simulated physical processes can be effectively visualized through a customized graphical interface.

5.3 Significance of Research

The current process for evaluating the readiness of cyber professionals for critical infrastructure protection is overly reliant upon on exam-based certifications. Unfortunately, this process does not provide an adequate ability to examine the true effectiveness of cyber first responders in real-world scenarios. With attacks targeting critical infrastructure on the rise and the potential devastating implications to public safety, it is imperative to find a means to evaluate the readiness of the personnel that will be the first-line responders. Through this research, the framework for evaluating the readiness of cyber first responders is carefully developed and its utility is tested. Based on the positive results from the test, the framework can be expanded to provide comprehensive and standardized evaluation method for cyber first responders. The Y-Box can be used to support the effort by facilitating the extension of the scenario types that require more complex, realistic evaluation environments.

5.4 Recommendations for Future Research

The recommendations are collected based on the experience from this research.

5.4.1 Common Scenarios for Evaluation.

This research presents five potential scenarios that are used in a cyber exercise. Common scenarios that are mapped to reference documents such as the NIST Special

Publication 800-82 Guide to Industrial Control System Security can be developed to meet the local and specific needs of evaluation.

5.4.2 Simulation within CPU Module.

This research has a dedicated simulation terminal that connects to the CPU module of the Y-Box for the simulation of physical processes. The simulation terminal requires substantial resources for software license and establishment of its hardware platform. The cost can be reduced with the development of open source graphic user interface that can be run within the CPU module.

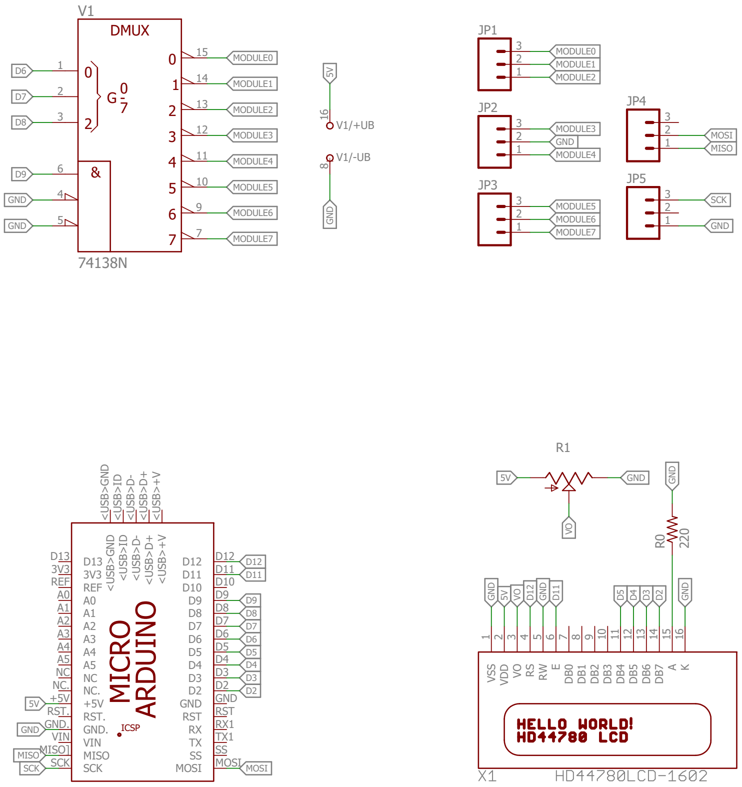
5.4.3 Physical Processes Simulation Library.

The Y-Box demonstrates the first two stages within WWTP. When more types of physical processes simulation are added, user-friendly library feature that can select a desired simulation can support users with limited or no background knowledge on the Y-Box. The library feature should be accompanied by the PLC ladder logic codes and specifics on the physical connection between the Y-Box and PLCs. User-friendly hardware design with more precise components is in progress.

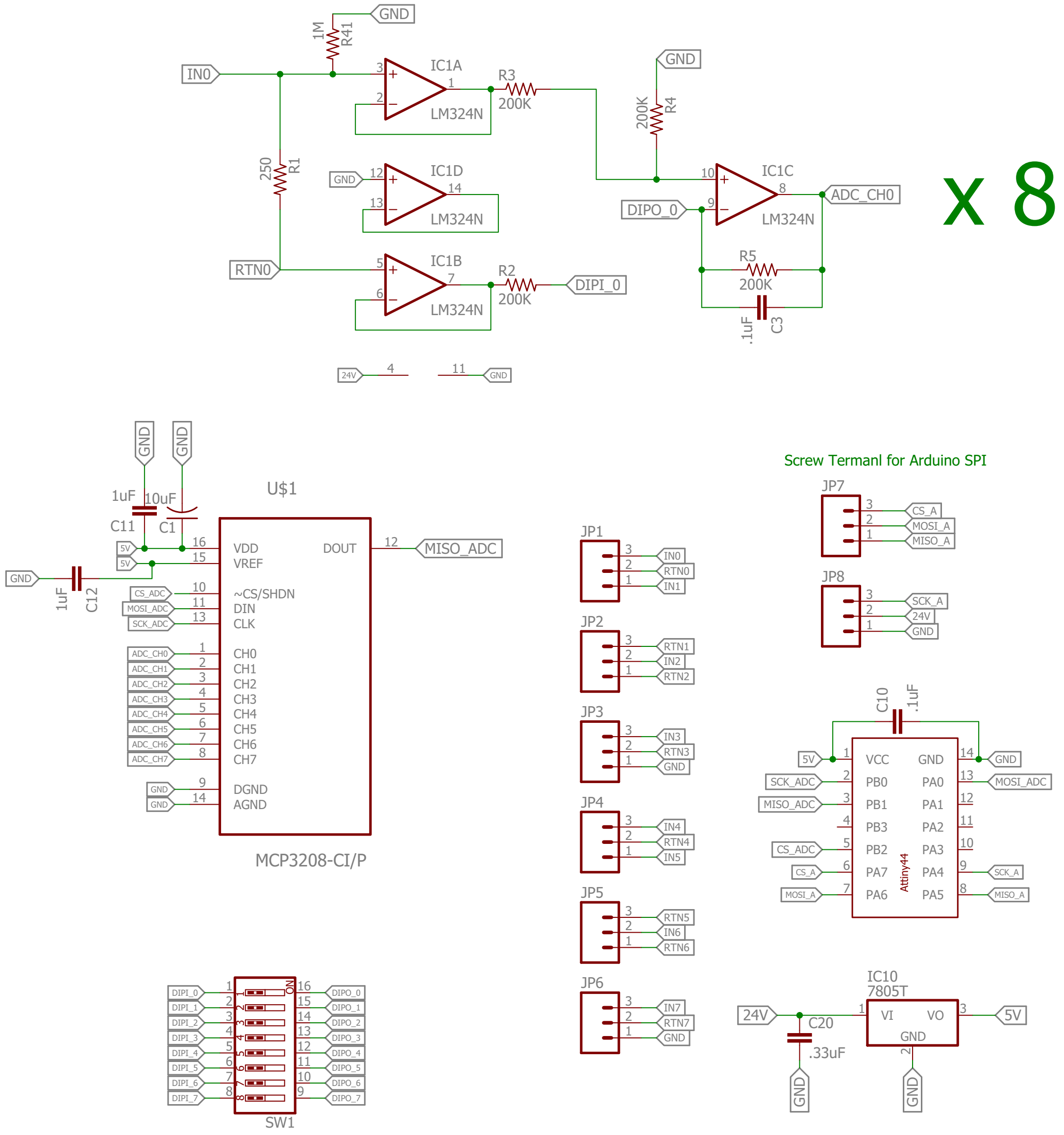
Appendix A. Y-Box Schematic for Modules

Schematics are drawn by Eagle PCB Design Software using open source library.

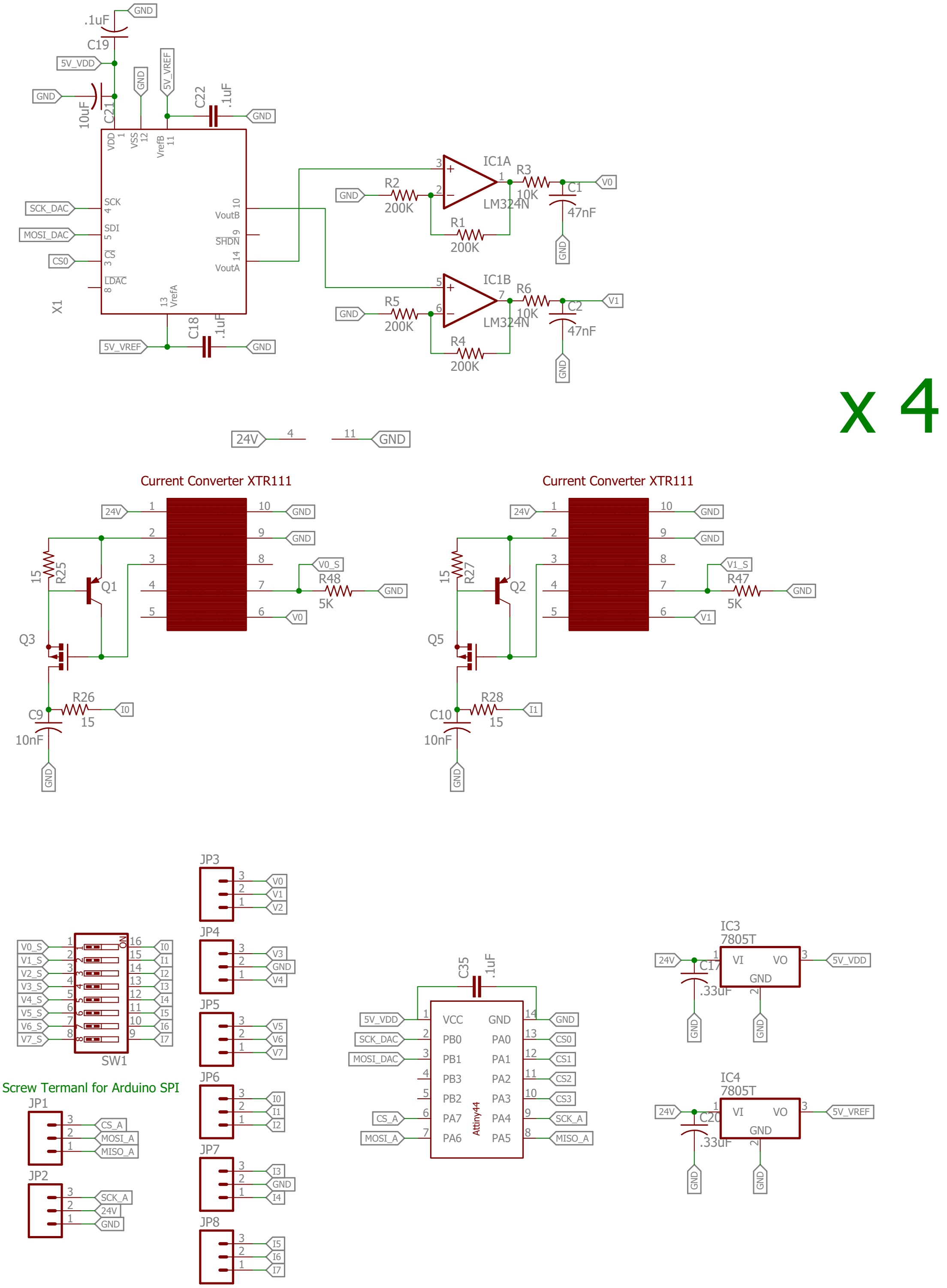
A.1 CPU Module



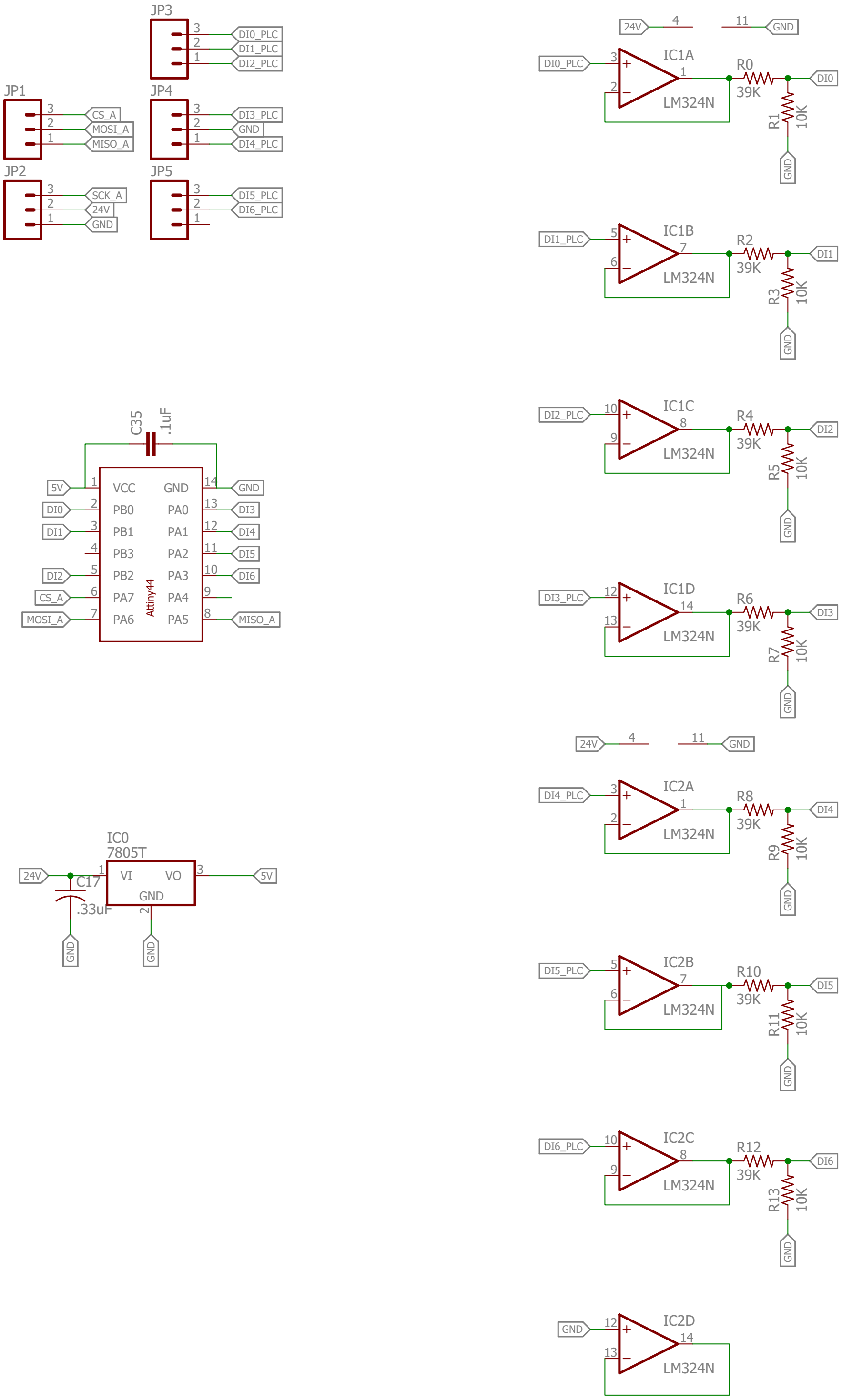
A.2 Analog Input Module



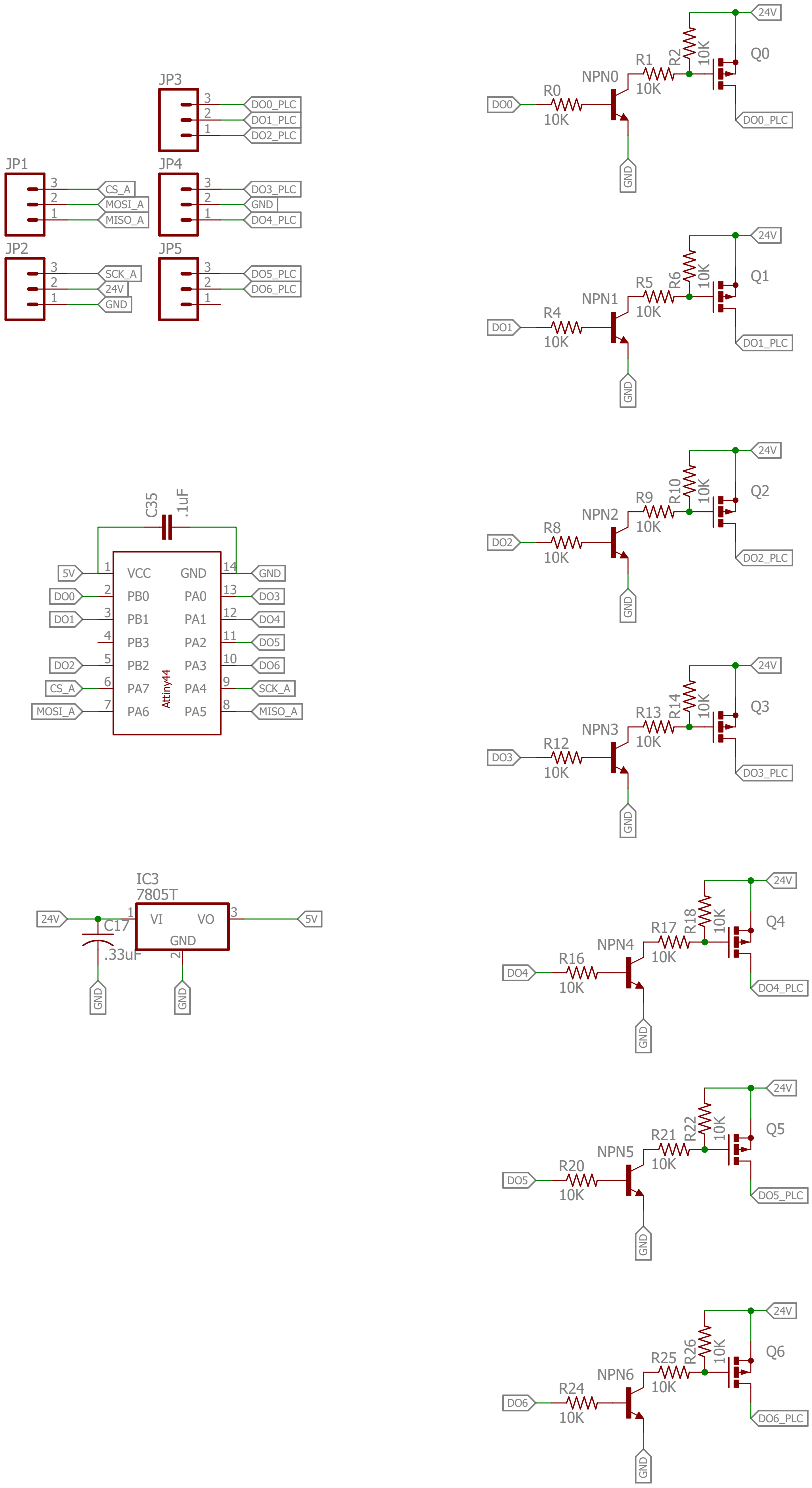
A.3 Analog Output Module



A.4 Digital Input Module



A.5 Digital Output Module [28]


JP3
3 DO0_PLC
2 DO1_PLC
1 DO2_PLC
JP1
3 CS_A
2 MOSI_A
1 MISO_A
JP4
3 DO3_PLC
2 GND
1 DO4_PLC
JP2
3 SCK_A
2 24V
1 GND
JP5
3 DO5_PLC
2 DO6_PLC
1
C35 .1uF
Attiny44
5V 1 VCC
DO0 2 PB0
DO1 3 PB1
4 PB3
DO2 5 PB2
CS_A 6 PA7
MOSI_A 7 PA6
GND 14 GND
DO3 13 PA0
DO4 12 PA1
DO5 11 PA2
DO6 10 PA3
SCK_A 9 PA4
MISO_A 8 PA5
IC3 7805T
24V 1 VI
VO 3 5V
GND 2
C17 .33uF
GND
GND
DO0 R0 10K NPN0 R1 10K R2 10K Q0 24V DO0_PLC GND
DO1 R4 10K NPN1 R5 10K R6 10K Q1 24V DO1_PLC GND
DO2 R8 10K NPN2 R9 10K R10 10K Q2 24V DO2_PLC GND
DO3 R12 10K NPN3 R13 10K R14 10K Q3 24V DO3_PLC GND
DO4 R16 10K NPN4 R17 10K R18 10K Q4 24V DO4_PLC GND
DO5 R20 10K NPN5 R21 10K R22 10K Q5 24V DO5_PLC GND
DO6 R24 10K NPN6 R25 10K R26 10K Q6 24V DO6_PLC GND

Appendix B. Microcontroller Code

Codes for microcontrollers are written using the open source Arduino Software.

B.1 CPU Module

```
//#include <SPI.h>
#include  <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
#define ID_AI 0x52
#define ID_DI 0x53
#define ID_AO 0x54
#define ID_DO 0x55
#define NUM_MODULES 4
#define MOSI_PIN MOSI
#define MISO_PIN MISO
#define SCK_PIN SCK
#define DEMUX_A0 6
#define DEMUX_A1 7
#define DEMUX_A2 8
#define DEMUX_E3 9
#define HB_CMD 0b10101010
#define RDA_CMD 0b01110000
#define WDA_CMD 0b01000000
#define DEFAULT_DELAY 1000
byte  module_types[NUM_MODULES];
byte count = 0;
//assign slots
//analog input module in slot 0
byte slotADC = 0;
//digital input module in slot 1
byte slotDI = 1;
//analog output module in slot 2
byte slotDAC = 2;
//digital ouput module in slot 3
byte slotDO = 3;
byte ID;
unsigned short ai_values[8];
unsigned short ao_values[8];
bool di_values[8];
bool do_values[8];
unsigned long last_time;
unsigned long timestamp;
char input;
char type;
char comma;
char comma2;
char newline;
int channel;
int slot;
int value;
float a_value;

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
  // put your setup code here, to run once:
  pinMode(MOSI_PIN, OUTPUT);
  pinMode(MISO_PIN, INPUT);
  pinMode(SCK_PIN, OUTPUT);
  pinMode(DEMUX_A0, OUTPUT);
  pinMode(DEMUX_A1, OUTPUT);
  pinMode(DEMUX_A2, OUTPUT);
  pinMode(DEMUX_E3, OUTPUT);
  digitalWrite(DEMUX_E3, LOW);
  // This makes the clock rising edge
  digitalWrite(SCK_PIN, LOW);
  Serial.begin(57600);
  enumerate_modules();
}

void loop(){
  unsigned long newtime;
  //HB_CMD check
  if (count == 0) {
    enumerate_modules();
  }
  count++;
  if (count >= 100) {
    count = 0;
  }

   update_values();
  // This code measures the amount of time between updating
the current values
  // This should give us some indication of how fast our
response time will be
  newtime = millis();
  last_time = newtime - timestamp;
  timestamp = newtime;

   if(Serial.available()){
     input = Serial.read();
     switch(input){
       case 'M':
         newline = Serial.read();
         if(newline != '\n'){
            Serial.println("Invalid Syntax");
           break;
        }
         print_modules();
        Serial.println('.');
         break;
       case 'T':
         newline = Serial.read();
         if(newline != '\n'){
            Serial.println("Invalid Syntax");
           break;
        }
          Serial.println(last_time);
         break;
       case 'R':
```

```
  type = Serial.read();
  //slot  =  Serial.parseInt();
  //comma  =  Serial.read();
  channel = Serial.parseInt();
  newline = Serial.read();
  if(newline != '\n'){
    Serial.println("Invalid Syntax");
    break;
  }
  if(type == 'D'){
    if(channel > 6){
      Serial.println("Channel out of range");
      break;
    }
    Serial.print("rd");
    Serial.print(channel);
    Serial.print(',');
    Serial.println(di_values[channel]);
  }
  else if(type == 'A'){
    if(channel > 8){
      Serial.println("Channel out of range");
      break;
    }
    Serial.print("ra");
    Serial.print(channel);
    Serial.print(',');
    Serial.println(ai_values[channel]);
  }
  else{
    Serial.println("UNKOWN TYPE");
    serialFlush();
  }
  break;
case 'W':
  type = Serial.read();
```

```
     channel = Serial.parseInt();
     comma = Serial.read();
     value = Serial.parseInt();
     newline = Serial.read();
     if(comma != ',' || newline != '\n'){
        Serial.println("Invalid Syntax");
       break;
    }
     if(type == 'D'){
       if(value > 1){
          Serial.println("Value must be either 0 or 1");
         break;
       }
        WD(get_slot(ID_DO), channel, value);
       Serial.print("wd");
       Serial.print(channel);
       Serial.print(",");
       Serial.println(value);
    }
     else if(type == 'A'){
       if(value > 4095){
          Serial.println("Value must be less than 4096");
         break;
       }
        WA(get_slot(ID_AO), channel, value);
       Serial.print("wa");
       Serial.print(channel);
       Serial.print(",");
       Serial.println(value);
    }
    else{
       Serial.println("UNKOWN TYPE");
       serialFlush();
    }
    break;
}
```

```
  }
}

//   https://forum.arduino.cc/index.php?topic=234151.0
void serialFlush(){
  while(Serial.available()) {
    Serial.read();
  }
}

void update_values(){
  for(byte i=0; i < NUM_MODULES; i++){
    switch(module_types[i]){
      case ID_AI:
        update_AI(i);
      break;
      case ID_DI:
        update_DI(i);
      break;
      /*
      case ID_AO:
        update_AO(i);
      break;
      case ID_DO:
        update_DO(i);
      break;
      */
    }
  }
}

void update_AI(byte slot){
  for(int i=0; i < 8; i++){
    ai_values[i] = RA(slot, i);
  }
}
```

```
void update_DI(byte slot){
  byte vals = RDA(slot);
  for(byte i=0; i < 8; i++){
    di_values[i] = (vals >> (7-i)) & 1;
  }
}

// This helper function returns the slot number of the first
// module that matches the given ID
byte get_slot(byte id){
  for(byte slot=0; slot < NUM_MODULES; slot++){
    if(module_types[slot] == id){
      return slot;
    }
  }
  return 0xFF;
}

void enumerate_modules(){
  for(byte i=0; i < NUM_MODULES; i++){
    module_types[i] = sendHB_CMD(i);
    if( ! ((module_types[i] >= ID_AI && module_types[i] <=
ID_DO) || module_types[i] == 0)){
      Serial.print("Unexpected module type (");
      Serial.print(module_types[i]);
      Serial.print(") found on slot: ");
      Serial.println(i);
    }
  }
}

void print_modules(){
  for(byte i=0; i < NUM_MODULES; i++){
    Serial.print("Slot # ");
    Serial.print(i);
    Serial.print(": ");
    switch(module_types[i]){
      case ID_AI:
         Serial.println("Analog Input");
      break;
      case ID_DI:
         Serial.println("Digital Input");
      break;
      case ID_AO:
         Serial.println("Analog Output");
      break;
      case ID_DO:
         Serial.println("Digital Output");
      break;
      default:
         Serial.println("Empty");
      break;
    }
  }
}

void  print_test(){
  //ADC
  byte channelADC0 = 7;
  unsigned short ADC_value0 = RA(slotADC, channelADC0);
  Serial.print("ADC Value 0: ");
   Serial.println(ADC_value0);
  //delay(1000);
  //Digital Input
  byte channelDI0 = 1;
  byte DI_value0 = RD(slotDI, channelDI0);
  byte DI_values =  RDA(slotDI);
  Serial.print("DI: ");
   Serial.println(DI_value0);
  Serial.print("DIs: ");
   Serial.println(DI_values);
   //delay(1000);
   //DAC
   //channel from 0 to 7, total 8 channels
   byte channelDAC1 = 1;
   //with channel and first 4 bits, first bit is 0
   byte byte0_DAC1 = 4;
   //8 lower data bits
   byte byte1_DAC1 = 0;
   //write the anlaog values to DAC
   byte DAC_channel = WA (slotDAC, channelDAC1, byte0_DAC1,
byte1_DAC1);
   Serial.print("DAC channel: ");
    Serial.println(DAC_channel);
   //delay(1000);
   //  digital ouput
   //get a single digital value to write
   byte DO5 = 1;
   //get all dital values to write, MSB first, starting with
channel 0
   byte DOs = 0b11110000;
   byte DO_channel = WD (slotDO, 5, DO5);
   byte DO_all = WDA (slotDO, DOs);
   delay(1000);
   Serial.print("DO channel: ");
    Serial.println(DO_channel);
   Serial.print("DO all: ");
    Serial.println(DO_all);
}

void selectSlot (byte slot){
  digitalWrite(DEMUX_A2, (slot >> 2) & 1);
  digitalWrite(DEMUX_A1, (slot >> 1) & 1);
  digitalWrite(DEMUX_A0, slot & 1);
  digitalWrite(DEMUX_E3, HIGH);
}
```

```
void disableSlot(){
  digitalWrite(DEMUX_E3, LOW);
}

void send_cmd(byte slot, byte data){
  selectSlot(slot);
  delayMicroseconds(DEFAULT_DELAY);
  shiftOut(MOSI_PIN, SCK_PIN, MSBFIRST, data);
}

void send_cmd(byte slot, byte data1, byte data2){
  selectSlot(slot);
  delayMicroseconds(DEFAULT_DELAY);
  shiftOut(MOSI_PIN, SCK_PIN, MSBFIRST, data1);
  delayMicroseconds(DEFAULT_DELAY);
  shiftOut(MOSI_PIN, SCK_PIN, MSBFIRST, data2);
}

byte recv(bool disable_slot=true){
  byte ID = shiftIn(MISO_PIN, SCK_PIN, MSBFIRST);
  if(disable_slot){
    disableSlot();
  }
  return ID;
}

byte sendHB_CMD(byte slot){
  byte resp;
  send_cmd(slot, HB_CMD);
   delayMicroseconds(DEFAULT_DELAY);
  resp = recv();
  return resp;
}

unsigned short RA(byte slot, byte channel){
  unsigned short dataADC;
```

```
   send_cmd(slot, channel);
  //time needed for ATTINY to bring data from the selected AD
channel
  //~~9 ms required for the readADC function to finish
   delayMicroseconds(DEFAULT_DELAY);//give extra 11 ms
  //read first byte
  dataADC = recv(false);
  //time needed for next byte to be in USIDR
   delayMicroseconds(DEFAULT_DELAY);
  dataADC = (dataADC << 8) | recv();
  return dataADC;
}

byte RD (byte slot, byte channel){
  byte dataDI;
  send_cmd(slot, channel);
  //time needed for ATTINY to bring data from the selected DI
channel
   delayMicroseconds(DEFAULT_DELAY);
  dataDI = recv();
  return dataDI;
}

byte RDA (byte slot){
  byte dataDI_A;
  send_cmd(slot, RDA_CMD);
  delayMicroseconds(DEFAULT_DELAY);
  dataDI_A = recv();
  return dataDI_A;
}

// is_current is 0 for Volts, 1 for Amps
byte WA(byte slot, byte channel, float value, bool
is_current){
  unsigned short val;
  // This means Current, 0.0 - 20.0
  if(is_current){
      val = (unsigned short)(4095.0 * value / 20.0);
  }
  else{
    // This is volts!
    val = (unsigned short)(4095.0 * value / 10.0);
  }
  return WA(slot, channel, val);
}

byte WA(byte slot, byte channel, unsigned short value){
  return WA(slot, channel, highByte(value), lowByte(value));
}

byte WA(byte slot, byte channel, byte value0, byte value1){
  byte cmd;
  //combine channel information and first 4 bits in the first
byte
  //firt bit is 0
  cmd = (channel << 4) | value0;
  send_cmd(slot, cmd, value1);
   delayMicroseconds(5000);
  // ATTINY must return a byte acknowledging that the write
command was successful
  // ATTINY should send the channel that was written
  return recv();
}

byte WD (byte slot, byte channel, byte value) {
  //combine channel information and digital value
  byte cmd = (channel << 1) | value;
  send_cmd(slot, cmd);
  //time needed for ATTINY to do digitalRead
   delayMicroseconds(DEFAULT_DELAY);
  // ATTINY must return a byte acknowledging that the write
command was successful
```

```
  // ATTINY should send the channel that was written
  return recv();
}

byte WDA (byte slot, byte values){
  byte cmd1 = WDA_CMD | (values >> 2);
  byte cmd2 = values << 6;
  send_cmd(slot, cmd1, cmd2);
  delayMicroseconds(DEFAULT_DELAY);
  // ATTINY must return a byte acknowledging that the write
command was successful
  // ATTINY should send 10 when write command all was
successful
  return recv();
}
```

B.2 Analog Input Module

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/power.h>
#include <avr/sleep.h>
//set bit in I/O register
#ifndef cbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit)) //OR
#endif
//clear bit in I/O register
#ifndef sbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit)) //AND
#endif
#define MOSI_A 6
#define MISO_A 5
#define SCK_A 4
#define CS_A PCINT7
#define SCK_ADC 10
#define MISO_ADC 9
#define CS_ADC 8
#define MOSI_ADC 0
#define NUM_CHANNELS 8
unsigned short current_values[NUM_CHANNELS];

// the setup function runs once when you press reset or power
the board
void setup() {
  cli();//Global Interrupt Disable, disable interrupts during
setup
  pinMode(MOSI_A, INPUT);
  //change to input for MISO_A to make it high impedance
  pinMode(MISO_A, INPUT);
  pinMode(SCK_A, INPUT);
  pinMode(CS_A, INPUT);
  pinMode(MOSI_ADC, OUTPUT);
  pinMode(MISO_ADC, INPUT);
  pinMode(SCK_ADC, OUTPUT);
  pinMode(CS_ADC, OUTPUT);
  //Clock idles at low, 0 mode
  digitalWrite(SCK_ADC, LOW);
  digitalWrite(CS_ADC, HIGH);
  sbi(GIMSK,PCIE0); // Turn on Pin Change interrupt
  sbi(PCMSK0,CS_A); // Which pins are affected by the interru
  spiSlaveInit();
  clear_values();
  sei(); //last line of setup - enable interrupts after setup
}

// the loop function runs over and over again forever
void loop() {
  update_values();
}

ISR(PCINT0_vect) {
  if ((USISR >> USIOIF) & 1){
    resetCounter();
  }
  byte pinState = digitalRead(CS_A);//gives PORTA7 value,
PCINT7
  if (!pinState) {
    //wait until 16 edges to finish 8 bit transfer for shiftO
    //make sure USIDR is full with ADC channel information
    //ensure USIOIF is 0 before while loop
    //make MISO output mode to transmit data to Arduino Micro
    pinMode(MISO_A, OUTPUT);
    while(!((USISR >> USIOIF) & 1));
    //write one bit to USIOIF
    resetCounter();
    byte channel = USIDR;
    if (((channel >> 7) & 1) == 1){
      USIDR = 0x52;
    }
    else {
      //read digital number from the ADC, take some time
      unsigned short dataBytes = current_values[channel];
      byte dataBytes0 = highByte(dataBytes);
      byte dataBytes1 = lowByte(dataBytes);
      USIDR = dataBytes0;
      //wait until first shiftIn is done
      while(!((USISR >> USIOIF) & 1));
      resetCounter();
      //write for the second shiftIn
      USIDR = dataBytes1;
    }
  }
  else{
    //make MISO input
    pinMode(MISO_A, INPUT);
  }
}

void clear_values(){
  for(int i=0; i < NUM_CHANNELS; i++){
    current_values[i] = 0;
  }
}

void update_values(){
  for(int i=0; i < NUM_CHANNELS; i++){
    current_values[i] = readADC(i);
    //delay(1);
  }
}

unsigned short readADC(byte channel){
    unsigned short RequestBytes = (0b00011000 + channel) << 6
    //noInterrupts();
    digitalWrite(CS_ADC, LOW);
    ytransfer(MOSI_ADC, MISO_ADC, SCK_ADC, MSBFIRST,
highByte(RequestBytes));
     byte data0 = ytransfer(MOSI_ADC, MISO_ADC, SCK_ADC,
MSBFIRST,  lowByte(RequestBytes));
     byte data1 = ytransfer(MOSI_ADC, MISO_ADC, SCK_ADC,
MSBFIRST, 0x00);
     digitalWrite(CS_ADC, HIGH);
     //interrupts();
     data0 &= 0b00001111;
     unsigned short bitsADC = (((unsigned short)data0) << 8) |
data1;
     return bitsADC;
}

void  resetCounter(){
  USISR =  1 << USIOIF;
}

// Initialise as SPI slave
void  spiSlaveInit()
{
  USICR = (1 << USIWM0) // SPI mode
           |(0 << USIWM1) // Three Wire Mode
           |(1 << USICS1); // Clock is external
}

byte ytransfer(uint8_t MOSI, uint8_t MISO, uint8_t clockPin,
uint8_t bitOrder, byte val)
{
      int i;
      byte data = 0;
      for (i = 0; i < 8; i++)  {
         if (bitOrder == LSBFIRST){
           digitalWrite(MOSI, !!(val & (1 << i)));
           data |= digitalRead(MISO) << i;
        }
        else{
           digitalWrite(MOSI, !!(val & (1 << (7 - i))));
           data |= digitalRead(MISO) << (7 - i);
        }
         digitalWrite(clockPin, HIGH);
         digitalWrite(clockPin, LOW);
      }
      return data;
}
```

B.3 Analog Output Module

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/power.h>
#include <avr/sleep.h>
//set bit in I/O register
#ifndef cbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit)) //OR
#endif
//clear bit in I/O register
#ifndef sbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit)) //AND
#endif
#define CS_A PCINT7
#define MOSI_A 6
#define MISO_A 5
#define SCK_A 4
#define SCK_DAC 10
#define MOSI_DAC 9
#define CS_DAC0 0
#define CS_DAC1 1
#define CS_DAC2 2
#define CS_DAC3 3

// the setup function runs once when you press reset or power
the board
void setup() {
  cli();//Global Interrupt Disable, disable interrupts during
setup
  pinMode(MOSI_A, INPUT);
 //change to input for MISO_A to make it high impedance
  pinMode(MISO_A, INPUT);
  pinMode(SCK_A, INPUT);
  pinMode(CS_A, INPUT);
  pinMode(MOSI_DAC, OUTPUT);
  pinMode(SCK_DAC, OUTPUT);
  pinMode(CS_DAC0, OUTPUT);
  pinMode(CS_DAC1, OUTPUT);
  pinMode(CS_DAC2, OUTPUT);
  pinMode(CS_DAC3, OUTPUT);
  //Clock idles at low, 0 mode
  digitalWrite(SCK_DAC, LOW);
  digitalWrite(CS_DAC0, HIGH);
  digitalWrite(CS_DAC1, HIGH);
  digitalWrite(CS_DAC2, HIGH);
  digitalWrite(CS_DAC3, HIGH);
  sbi(GIMSK,PCIE0); // Turn on Pin Change interrupt
  sbi(PCMSK0,CS_A); // Which pins are affected by the interru
  spiSlaveInit();
  sei(); //last line of setup - enable interrupts after setup
}

// the loop function runs over and over again forever
void loop() {}
ISR(PCINT0_vect) {
  if ((USISR >> USIOIF) & 1){
    resetCounter();
  }
  byte pinState = digitalRead(CS_A);//gives PORTA7 value,
PCINT7
  if (!pinState) {
    //make MISO output mode to transmit data to Arduino Micro
    pinMode(MISO_A, OUTPUT);
    while(!((USISR >> USIOIF) & 1)){
    }
    resetCounter();//write one bit to USIOIF
    unsigned short command = USIDR;
    if (((command >> 7) & 1) == 1){
      USIDR = 0x54;
    }
    else{
    while(!((USISR >> USIOIF) & 1));
    resetCounter();
    byte dataByte = USIDR;
    //combine data to send to DAC
    byte channel = command >> 4;
    unsigned short data = ((command & 0b00001111) << 8) |
dataByte;
    //takes time for this function
    sendtoDAC(data, channel);
    USIDR = channel;
    }
  }
  else{
    //make MISO input
    pinMode(MISO_A, INPUT);
  }
}

void  resetCounter(){
  USISR = 1 << USIOIF;
}

// Initialise as SPI slave
void  spiSlaveInit()
{
  USICR = (1 << USIWM0) // SPI mode
          |(0 << USIWM1) // Three Wire Mode
          |(1 << USICS1); // Clock is external
}

void sendtoDAC (unsigned short value, byte addr){
  byte CS;
  if ((addr == 0)||(addr == 1)){
    CS = CS_DAC0;
    }
  if ((addr == 2)||(addr == 3)){
    CS = CS_DAC1;
    }
  if ((addr == 4)||(addr == 5)){
    CS = CS_DAC2;
    }
  if ((addr == 6)||(addr == 7)){
    CS = CS_DAC3;
    }
  byte defaultfirstByte = 0b00110000;
  byte firstByteData = highByte(value)| defaultfirstByte;
  byte secondByteData = lowByte(value);
  byte channelBit = (addr & 1) << 7;//0 is channel 0, 1 is
channel 1
  byte firstByteDAC = channelBit | firstByteData;
  byte secondByteDAC = secondByteData;
  noInterrupts();
  digitalWrite (CS, LOW);
  shiftOut(MOSI_DAC, SCK_DAC, MSBFIRST, firstByteDAC);
  shiftOut(MOSI_DAC, SCK_DAC, MSBFIRST, secondByteDAC);
  digitalWrite (CS, HIGH);
  interrupts();
 }
```

B.4 Digital Input Module

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/power.h>
#include <avr/sleep.h
//set bit in I/O register
#ifndef cbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit)) //OR
#endif
//clear bit in I/O register
#ifndef sbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit)) //AND
#endif
#define CS_A PCINT7
#define MOSI_A 6
#define MISO_A 5
#define SCK_A 4
#define DI0 10
#define DI1 9
#define DI2 8
#define DI3 0
#define DI4 1
#define DI5 2
#define DI6 3
#define NUM_CHANNELS 7
byte  current_values[NUM_CHANNELS];

// the setup function runs once when you press reset or power
the board
void setup() {
  cli();//Global Interrupt Disable, disable interrupts during
setup
  pinMode(MOSI_A, INPUT);
  //change to input for MISO_A to make it high impedance
  pinMode(MISO_A, INPUT);
  pinMode(SCK_A, INPUT);
  pinMode(CS_A, INPUT);
  pinMode(DI0, INPUT);
  pinMode(DI1, INPUT);
  pinMode(DI2, INPUT);
  pinMode(DI3, INPUT);
  pinMode(DI4, INPUT);
  pinMode(DI5, INPUT);
  pinMode(DI6, INPUT);
  sbi(GIMSK,PCIE0); // Turn on Pin Change interrupt
  sbi(PCMSK0,CS_A); // Which pins are affected by the interru
  spiSlaveInit();
  sei(); //last line of setup - enable interrupts after setup
}

// the loop function runs over and over again forever
void loop() {
  update_values();
}

ISR(PCINT0_vect) {
  if ((USISR >> USIOIF) & 1){
    resetCounter();
  }
  byte pinState = digitalRead(CS_A);//gives PORTA7 value,
PCINT7
  if (!pinState) {
    //make MISO output mode to transmit data to Arduino Micro
    pinMode(MISO_A, OUTPUT);
    while(!((USISR >> USIOIF) & 1));
    resetCounter();
    byte cmd = USIDR;
    //if Hearbeat ID
    if (((cmd >> 7) & 1) == 1){
       USIDR = 0x53;
    }
    else if ((cmd >> 6) == 1){
      byte DIs;
```

```
       DIs = (current_values[0] << 7)
             |(current_values[1] << 6)
             |(current_values[2] << 5)
             |(current_values[3] << 4)
             |(current_values[4] << 3)
             |(current_values[5] << 2)
             |(current_values[6] << 1);
      USIDR = DIs;
    }
    else{
       USIDR = current_values[cmd];
    }
  }
  else{
    //make MISO input
     pinMode(MISO_A, INPUT);
  }
}

void  resetCounter(){
  USISR = 1 << USIOIF;
}

void  spiSlaveInit()
{
  USICR = (1 << USIWM0) // SPI mode
           |(0 << USIWM1) // Three Wire Mode
           |(1 << USICS1); // Clock is external
}

byte  readDI (byte  addr){
  byte DI;
  if (addr == 0){
     DI = digitalRead(DI0);
    }
  if (addr == 1){
    DI = digitalRead(DI1);
    }
  if (addr == 2){
    DI = digitalRead(DI2);
    }
  if (addr == 3){
    DI = digitalRead(DI3);
    }
  if (addr == 4){
    DI = digitalRead(DI4);
    }
  if (addr == 5){
    DI = digitalRead(DI5);
    }
  if (addr == 6){
    DI = digitalRead(DI6);
    }
    return DI;
}

void  update_values(){
  for(int i=0; i < NUM_CHANNELS; i++){
    current_values[i] = readDI(i);
    //delay(1);
  }
}
```

B.5 Digital Output Module

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/power.h>
#include <avr/sleep.h>
//set bit in I/O register
#ifndef cbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit)) //OR
#endif
//clear bit in I/O register
#ifndef sbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit)) //AND
#endif
#define CS_A PCINT7
#define MOSI_A 6
#define MISO_A 5
#define SCK_A 4
#define DO0 10
#define DO1 9
#define DO2 8
#define DO3 0
#define DO4 1
#define DO5 2
#define DO6 3

// the setup function runs once when you press reset or power
the board
void setup() {
  cli();//Global Interrupt Disable, disable interrupts during
setup
  pinMode(MOSI_A, INPUT);
  //change to input for MISO_A to make it high impedance
  pinMode(MISO_A, INPUT);
  pinMode(SCK_A, INPUT);
  pinMode(CS_A, INPUT);
  pinMode(DO0, OUTPUT);
  pinMode(DO1, OUTPUT);
  pinMode(DO2, OUTPUT);
  pinMode(DO3, OUTPUT);
  pinMode(DO4, OUTPUT);
  pinMode(DO5, OUTPUT);
  pinMode(DO6, OUTPUT);
  sbi(GIMSK,PCIE0); // Turn on Pin Change interrupt
  sbi(PCMSK0,CS_A); // Which pins are affected by the interru
  spiSlaveInit();
  sei(); //last line of setup - enable interrupts after setup
}

// the loop function runs over and over again forever
void loop() {}
ISR(PCINT0_vect) {
  //if ((USISR >> USIOIF) & 1){
    resetCounter();
  //}
  byte pinState = digitalRead(CS_A);//gives PORTA7 value,
PCINT7
  if (!pinState) {
    //make MISO output mode to transmit data to Arduino Micro
    pinMode(MISO_A, OUTPUT);
    while(!((USISR >> USIOIF) & 1));
    byte cmd1 = USIDR;
     //if Hearbeat ID
     if (((cmd1 >> 7) & 1) == 1){
        USIDR = 0x55;
     }
     else {
       if ((cmd1 >> 6) == 1){
         resetCounter();
         while(!((USISR >> USIOIF) & 1));
         byte cmd2 = USIDR;
         byte DOs = (cmd1 << 2) | (cmd2 >> 6);
         //to indicate the completion of write all
         digitalWrite(DO0, (DOs >> 7) & 1);
```

```
          digitalWrite(DO1, (DOs >> 6) & 1);
          digitalWrite(DO2, (DOs >> 5) & 1);
          digitalWrite(DO3, (DOs >> 4) & 1);
          digitalWrite(DO4, (DOs >> 3) & 1);
          digitalWrite(DO5, (DOs >> 2) & 1);
          digitalWrite(DO6, (DOs >> 1) & 1);
         USIDR = 10;
        }
        else{
          writeDO(cmd1);
        }
      }
  }
   else{
    //make MISO input
    pinMode(MISO_A, INPUT);
  }
}

void resetCounter(){
  USISR = 1 << USIOIF;
}

void spiSlaveInit()
{
  USICR = (1 << USIWM0) // SPI mode
          |(0 << USIWM1) // Three Wire Mode
          |(1 << USICS1); // Clock is external
}

void writeDO (byte channel_value){
  byte addr = channel_value >> 1;
  byte value = channel_value & 1;
  if (addr == 0){
    digitalWrite(DO0, value);
    USIDR = 0;
    }
  else if (addr == 1){
     digitalWrite(DO1, value);
    USIDR = 1;
    }
  else if (addr == 2){
     digitalWrite(DO2, value);
    USIDR = 2;
    }
  else if (addr == 3){
     digitalWrite(DO3, value);
    USIDR = 3;
    }
  else if (addr == 4){
     digitalWrite(DO4, value);
    USIDR = 4;
    }
  else if (addr == 5){
     digitalWrite(DO5, value);
    USIDR = 5;
    }
  else {
     digitalWrite(DO6, value);
    USIDR = 6;
    }
}
```

Appendix C. Simulation code for WWTP stages

```
function varargout = screen_GUI_resizedS2(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @screen_GUI_resizedS2_OpeningFcn,↙
...
                   'gui_OutputFcn',  @screen_GUI_resizedS2_OutputFcn,↙
...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function screen_GUI_resizedS2_OpeningFcn(hObject, eventdata, handles,↙
varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = screen_GUI_resizedS2_OutputFcn(hObject, eventdata,↙
handles)
varargout{1} = handles.output;

function init_S1_Callback(hObject, eventdata, handles)
init_S1 = get(hObject,'Value');
assignin('base','init_S1',init_S1)
set(handles.init_S1Num,'String',num2str(init_S1))

function init_S1_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function init_Clog_Callback(hObject, eventdata, handles)
init_Clog = get(hObject,'Value');
assignin('base','init_Clog',init_Clog)
set(handles.init_ClogNum,'String',num2str(init_Clog))

function init_Clog_CreateFcn(hObject, eventdata, handles)
```

```
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function init_S2_Callback(hObject, eventdata, handles)
init_S2 = get(hObject,'Value');
assignin('base','init_S2',init_S2)
set(handles.init_S2Num,'String',num2str(init_S2))

function init_S2_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function init_Tank_Callback(hObject, eventdata, handles)
init_Tank = get(hObject,'Value');
assignin('base','init_Tank',init_Tank)
set(handles.init_TankNum,'String',num2str(init_Tank))

function init_Tank_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function in_Flow_Callback(hObject, eventdata, handles)
in_Flow = get(hObject,'Value');
assignin('base','in_Flow',in_Flow)
set(handles.in_FlowNum,'String',num2str(in_Flow))

function in_Flow_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function f_Level_Callback(hObject, eventdata, handles)
f_Level = get(hObject,'Value');
assignin('base','f_Level',f_Level)
set(handles.f_LevelNum,'String',num2str(f_Level))

function f_Level_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --------------------------------------------------------------------
function screen_simulation_Callback(hObject, eventdata, handles)

function out_Rate_Callback(hObject, eventdata, handles)
out_Rate = get(hObject,'Value');
assignin('base','out_Rate',out_Rate)
set(handles.out_RateNum,'String',num2str(out_Rate))

function out_Rate_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function t_Rate_Callback(hObject, eventdata, handles)
t_Rate = get(hObject,'Value');
assignin('base','t_Rate',t_Rate)
set(handles.t_RateNum,'String',num2str(t_Rate))

function t_Rate_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function s1_Low_Callback(hObject, eventdata, handles)
s1_Low = get(hObject,'Value');
assignin('base','s1_Low',s1_Low)
set(handles.s1_LowNum,'String',num2str(s1_Low))

function s1_Low_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function s1_High_Callback(hObject, eventdata, handles)
s1_High = get(hObject,'Value');
assignin('base','s1_High',s1_High)
set(handles.s1_HighNum,'String',num2str(s1_High))

function s1_High_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function s1_HighHigh_Callback(hObject, eventdata, handles)
s1_HighHigh = get(hObject,'Value');
assignin('base','s1_HighHigh',s1_HighHigh)
set(handles.s1_HighHighNum,'String',num2str(s1_HighHigh))

function s1_HighHigh_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function t_Low_Callback(hObject, eventdata, handles)
t_Low = get(hObject,'Value');
assignin('base','t_Low',t_Low)
set(handles.t_LowNum,'String',num2str(t_Low))

function t_Low_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function t_High_Callback(hObject, eventdata, handles)
t_High = get(hObject,'Value');
assignin('base','t_High',t_High)
set(handles.t_HighNum,'String',num2str(t_High))

function t_High_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function t_HighHigh_Callback(hObject, eventdata, handles)
t_HighHigh = get(hObject,'Value');
assignin('base','t_HighHigh',t_HighHigh)
set(handles.t_HighHighNum,'String',num2str(t_HighHigh))

function t_HighHigh_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
```

```
end

function run_Screen_Callback(hObject, eventdata, handles)

delete(instrfindall)
%establish connection with micro
micro = serial('COM21', 'BaudRate', 57600);

%open serial port
fopen(micro);

%connection test
fprintf(micro,'M');
connection_test = [{fscanf(micro)}; {fscanf(micro)}; {fscanf(micro)};...
    {fscanf(micro)}; {fscanf(micro)}]
assignin('base','connection_test',connection_test)

%Scenario 1
%for initial tank level
t_Level = get(handles.init_Tank,'Value');
%for initial s1 and s2 level
s1_Level = get(handles.init_S1,'Value');
s2_Level = get(handles.init_S2,'Value');
%for initial clog level
b_Clog = get(handles.init_Clog,'Value');

%Scenario 2

for a = 1:200
%for Scenario 1
%stage 1 floats
%Low for the tank level
s1_LSL = get(handles.s1_Low,'Value');
%High for the tank level
s1_LSH = get(handles.s1_High,'Value');
%High High for the tank level
s1_LSHH = get(handles.s1_HighHigh,'Value');

%wet well tank floats
%Low for the wet well tank level
t_LSL = get(handles.t_Low,'Value');
%High High for the wet well tank level
t_LSHH = get(handles.t_HighHigh,'Value');

%for Scenario 1
%user defined values
```

```
%filth level from the tank
f_Level = get(handles.f_Level,'Value');

%Clog clean rate when belt is moving
c_Rate = get(handles.c_Rate,'Value');

maxS1 = 20;
maxTank = 100;
maxG = 100

DO = [{'WD0'},{'WD1'},{'WD2'},{'WD3'},{'WD4'},{'WD5'},{'WD6'}];
g_Mode = get(handles.g_Mode,'Value');
fprintf(micro,[DO{5},',',num2str(g_Mode),'\n']);
fscanf(micro)

%Analog Outputs
AO = [{'WA0'},{'WA1'},{'WA2'},{'WA3'},{'WA4'},{'WA5'},{'WA6'},{'WA7'}];

%Analog output: in_Flow rate to the tank
in_Flow = get(handles.in_Flow,'Value');
AO_value = floor(in_Flow * 4095); AO_ch = 1;
fprintf(micro,[AO{AO_ch},',',num2str(AO_value),'\n']);
fscanf(micro)

%Analog output: Level difference between stage 1 and stage 2
d_Level = s1_Level - s2_Level;
AO_value = floor(d_Level * 4095 / maxS1); AO_ch = 2;
fprintf(micro,[AO{AO_ch},',',num2str(AO_value),'\n']);
fscanf(micro)

%Digital outputs: three digital ouputs for stage 1 floats
if s1_Level > s1_LSHH
    DO_s1L = 0; DO_s1H = 1; DO_s1HH = 1;
else if s1_Level > s1_LSH
        DO_s1L = 0; DO_s1H = 1; DO_s1HH = 0;
    else if s1_Level < s1_LSL
        DO_s1L = 1; DO_s1H = 0; DO_s1HH = 0;
        else
            DO_s1L = 0; DO_s1H = 0; DO_s1HH = 0;
        end
    end
end

%Low float sensor for stage 1
fprintf(micro,[DO{1},',',num2str(DO_s1L),'\n']);
fscanf(micro)
```

```
%High float sensor for stage 1
fprintf(micro,[DO{2},',',num2str(mod(DO_s1H + 1, 2)),'\n']);
fscanf(micro)
%High High float sensor for stage 1
fprintf(micro,[DO{3},',',num2str(mod(DO_s1HH + 1, 2)),'\n']);
fscanf(micro)

%%Digital outputs: three digital outputs for tank floats
if t_Level > t_LSHH
    DO_tL = 0; DO_tH = 1; DO_tHH = 1;
    else if t_Level < t_LSL
        DO_tL = 1; DO_tH = 0; DO_tHH = 0;
        else
            DO_tL = 0; DO_tH = 0; DO_tHH = 0;
        end
end

%Low float sensor for wet well
fprintf(micro,[DO{4},',',num2str(DO_tL),'\n']);
fscanf(micro)
%High High float sensor for wet well
fprintf(micro,[DO{6},',',num2str(mod(DO_tHH + 1, 2)),'\n']);
fscanf(micro)

%Inputs
%Analog Input
%pump speed is the amound of water going into the screen
AI_ch = 1;
AI = [{'RA0'},{'RA1'},{'RA2'},{'RA3'},{'RA4'},{'RA5'},{'RA6'},{'RA7'}];
fprintf(micro,[AI{AI_ch},'\n']);
AI_value{AI_ch} = fscanf(micro);
AI_value{AI_ch} = str2num(AI_value{AI_ch}(5:end));
p_Speed = AI_value{AI_ch} / 4095;

%Pump speed feedback analog output
p_FB = p_Speed;
AO_value = floor(p_FB * 4095); AO_ch = 6;
fprintf(micro,[AO{AO_ch},',',num2str(AO_value),'\n']);
fscanf(micro)

%Digital Input
DI_ch = 1;
DI = [{'RD0'},{'RD1'},{'RD2'},{'RD3'},{'RD4'},{'RD5'},{'RD6'}];
fprintf(micro, [DI{DI_ch},'\n']);
DI_value{DI_ch} = fscanf(micro);
DI_value{DI_ch} = str2num(DI_value{DI_ch}(5:end));
```

```
b_Mode = DI_value{DI_ch};

%Clog gets removed when belt is running otherwise keeps getting clogged
%according to the filthy level
b_Clog = b_Clog - b_Mode * c_Rate + f_Level * p_Speed;

%clog percent should be between 0 and 100%
if b_Clog > 1
    b_Clog = 1;
end
if b_Clog < 0
    b_Clog = 0;
end

%tank level is added with in_Flow and substracted from the water going↙
into
%the stage 1
t_Level = (t_Level + in_Flow) - p_Speed;

%when tank level is 100, the excess water overflows into the reserve↙
tank
%tank level doesn't decrease in this set up
if t_Level > maxTank
    t_Level = maxTank;
end
%can't be less than 0
if t_Level < 0
        t_Level = 0;
end

stage_transfer = ((s1_Level - s2_Level) * (1 - b_Clog));

%stage 1 level dependent on clog level, transfer rate, level difference
%and in_Flow rate given by pump speed
s1_Level = s1_Level + p_Speed - stage_transfer;

%in_Flow more than the stage 1 capacity then it overflows
if s1_Level > maxS1
    s1_Level = maxS1;
end
%can'b be less than 0
if s1_Level < 0
    s1_Level = 0;
end

%This goes to CompactLogix
```

```
%Analog output: out_Flow rate from stage 2
%out_Flow indicates the flow going out from stage 2 in each iteration
out_Flow = stage_transfer;
AO_value = floor(out_Flow * 4095 / 20); AO_ch = 5;
fprintf(micro,[AO{AO_ch},',',num2str(AO_value),'\n']);
fscanf(micro)

%Scenario 2
%Digital Input for grit tank pump
DI_ch = 2;
fprintf(micro, [DI{DI_ch},'\n']);
DI_value{DI_ch} = fscanf(micro);
DI_value{DI_ch} = str2num(DI_value{DI_ch}(5:end));
g_Pump = DI_value{DI_ch};

%Digital Output for grit tank pump feedback
g_PumpFB = g_Pump;
fprintf(micro,[DO{7},',',num2str(g_PumpFB),'\n']);
fscanf(micro)

a = a + 1;

axes(handles.axes1);
    bar(t_Level); grid on;
    set(gca,'xticklabel',[])
    ylim([0 100]);
    xlabel(num2str(t_Level));
    r1 = refline(0,t_LSL);
    r3 = refline(0,t_LSHH);
    set(r1, 'color','c')
    text(1.5,t_LSL,{'Low'; 'Float'},'FontSize',6)
    set(r3, 'color','r')
    text(1.5,t_LSHH,{'High High'; 'Float'},'FontSize',6)
axes(handles.axes2);
    bar(s1_Level); grid on;
    set(gca,'xticklabel',[])
    ylim([0 20]);
    xlabel(num2str(s1_Level));
    r1 = refline(0,s1_LSL);
    r2 = refline(0,s1_LSH);
    r3 = refline(0,s1_LSHH);
    set(r1, 'color','c')
    text(1.5,s1_LSL,{'Low'; 'Float'},'FontSize',6)
    set(r2, 'color','y')
    text(1.5,s1_LSH,{'High'; 'Float'},'FontSize',6)
    set(r3, 'color','r')
    text(1.5,s1_LSHH,{'High High'; 'Float'},'FontSize',6)
axes(handles.axes3);
    bar(s2_Level); grid on;
    set(gca,'xticklabel',[])
    ylim([0 20]);
    xlabel(num2str(s2_Level));
axes(handles.axes4);
    bar(p_Speed); grid on;
    set(gca,'xticklabel',[])
    ylim([0 1]);
    xlabel(num2str(p_Speed));
axes(handles.axes5);
if b_Mode == 1
    plot(.5,'o','MarkerEdgeColor','r','MarkerSize',↙
20,'MarkerFaceColor','g');
    set(gca,'xticklabel',[],'yticklabel',[])
    ylim([0 1]);
else
    plot(.5,'o','MarkerEdgeColor','r','MarkerSize',↙
20,'MarkerFaceColor','w');
    set(gca,'xticklabel',[],'yticklabel',[])
    ylim([0 1]);
end
axes(handles.axes6);
if g_Pump == 1
    plot(.5,'o','MarkerEdgeColor','r','MarkerSize',↙
20,'MarkerFaceColor','g');
    set(gca,'xticklabel',[],'yticklabel',[])
    ylim([0 1]);
else
    plot(.5,'o','MarkerEdgeColor','r','MarkerSize',↙
20,'MarkerFaceColor','w');
    set(gca,'xticklabel',[],'yticklabel',[])
    ylim([0 1]);
end

set(handles.l_Difference,'String',num2str(d_Level/20))
set(handles.out_Flow,'String',num2str(out_Flow/20))

pause(.5);
guidata(hObject, handles);

end

function c_Rate_Callback(hObject, eventdata, handles)
c_Rate = get(hObject,'Value');
```

```
assignin('base','c_Rate',c_Rate)
set(handles.c_RateNum,'String',num2str(c_Rate))

function c_Rate_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function g_Mode_Callback(hObject, eventdata, handles)
g_Mode = get(hObject,'Value');
assignin('base','g_Mode',g_Mode)
set(handles.g_ModeNum,'String',num2str(g_Mode))
```

Bibliography

[1] Wastewater treatment options. Technical report, London School of Hygiene and Tropical Medicine and Loughborough University, 1999.

[2] U.S. Environmental Protection Agency. Primer for municipal wastewater treatment systems. World Wide Web Page, September 2004. Available at `http://www3.epa.gov/npdes/pubs/primer.pdf`.

[3] National Fire Protection Association. NFPA 1410: Standard on training for initial emergency scene operations, 2015.

[4] Information Assurance Certification Review Board. Certified SCADA security architect, 2015.

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

[6] Firefighter Close Calls. NFPA engine evolution #1. World Wide Web Page, 2000. Available at `www.firefighterclosecalls.com/weeklydrills.php`.

[7] Global Information Assurance Certification. Global industrial cyber security professional, 2015.

[8] R. Jaromin, B. Mullins, J. Butts, and J Lopez. Design and implementation of industrial control system emulators. In *Critical Infrastructure Protection VII, J. Butts and S. Shenoi (Eds.)*, pages 35–46. Springer, Heidelberg, Germany, 2013.

[9] E. Knapp and J. Langill. *Industrial Network Security.* Syngress, Waltham, Massachusetts, 2015.

[10] E. Kovacs. ENISA calls for new ICS/SCADA cybersecurity certification programs. Security Week, February 2015.

[11] M. Lennon. Attacks against SCADA systems doubled in 2014: Dell. Security Week, April 2015.

[12] T. Morris, A. Srivastava, B. Reaves, W. Gao, K. Pavurapu, and R. Reddi. A control system testbed to validate critical infrastructure protection concepts. *International Journal of Critical Infrastructure Protection*, 4(2):88–103, 2011.

[13] International Society of Automation. SA/IEC 62443 cybersecurity certificate programs, 2015.

[14] U.S. Department of Energy. National SCADA test bed - fact sheet. World Wide Web Page, September 2009. Available at http://www.energy.gov/sites/prod/files/oeprod/DocumentsandMedia/NSTB_Fact_Sheet_FINAL_09-16-09.pdf.

[15] U.S. Department of Homeland Security, Federal Emergency Management Agency. Cyberterrorism defense initiative, 2016. Available at www.cyberterrorismcenter.org.

[16] U.S. Department of Homeland Security, Industrial Control Systems Cyber Emergency Response Team. ICS focused malware, ICS-ICSA-14-178-01, 2014.

[17] U.S. Department of Homeland Security, Industrial Control Systems Cyber Emergency Response Team. Ongoing sophisticated malware campaign compromising ICS, ICS-ALERT-14-281-01C, 2014.

[18] U.S. Department of Homeland Security, Industrial Control Systems Cyber Emergency Response Team. Ongoing sophisticated malware campaign compromising ICS, ICS-ALERT-14-281-01D, 2016.

[19] U.S. Department of Homeland Security, National Cybersecurity and Communications Integration Center. Lessons learned from Cyber Storm IV, 2015. Available at www.dhs.gov/sites/default/files/publications/Lessons%20Learned%20from%20Cyber%20Storm%20IV.pdf.

[20] National Institute of Standards and Technology. Framework for improving critical infrastructure cybersecurity, version 1.0, 2014.

[21] A. Pauna. Certification of cyber security skills of ICS/SCADA professionals, 2014.

[22] F. Petruzella. *Programmable Logic Controllers*. McGraw-Hill, New York, New York, 2011.

[23] T. Reuters. Cyberattack that crippled ukrainian power grid was highly coordinated. CBCnews, January 2016.

[24] Rockwell Automation, Milwaukee, Wisconsin. *Compact 8-Bit Low Resolution Analog I/O Combination Module Uer Manual*, November 2001. Available at http://literature.rockwellautomation.com/idc/groups/literature/documents/um/1769-um008_-en-p.pdf.

[25] Rockwell Automation, Milwaukee, Wisconsin. *CompactLogix Controllers Specifications Technical Data*, August 2014. Available at http://literature.rockwellautomation.com/idc/groups/literature/documents/td/1769-td005_-en-p.pdf.

[26] Rockwell Automation, Milwaukee, Wisconsin. *1756 ControlLogix I/O Specifications Technical Data*, June 2015. Available at `http://literature.rockwellautomation.com/idc/groups/literature/documents/td/1756-td002_-en-e.pdf`.

[27] Rockwell Automation, Milwaukee, Wisconsin. *ControlLogix Analog I/O Modules User Manual*, March 2015. Available at `http://literature.rockwellautomation.com/idc/groups/literature/documents/um/1756-um009_-en-p.pdf`.

[28] Rockwell Automation, Milwaukee, Wisconsin. *ControlLogix Digital I/O Modules User Manual*, May 2015. Available at `http://literature.rockwellautomation.com/idc/groups/literature/documents/um/1756-um058_-en-p.pdf`.

[29] NYSE Governance Services and Veracode. Cybersecurity and corporate liability: The board's view, 2015.

[30] Siemens, Buffalo Grove, Illinois. *PXC Modular Series Owner's Manual*, 2013.

[31] D. Storm. Hackers exploit SCADA holes to take full control of critical infrastructure. Computerworld, January 2014.

[32] N. Wertzberger, C. Glatter, W. Mahoney, R. Gandhi, and K. Dick. Towards a low-cost SCADA test bed: An open-source platform for hardware-in-the-loop simulation. In *Proceedings of the 2011 International Conference on Security and Management, Special Track on Mission Assurance and Critical Infrastructure Protection*, 2011.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*
24–03–2016

2. REPORT TYPE
Master's Thesis

3. DATES COVERED *(From — To)*
Aug 2014 — Mar 2016

4. TITLE AND SUBTITLE

Framework for Evaluating the Readiness of
Cyber First Responders Responsible for
Critical Infrastructure Protection

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Yoon, Jungsang, Captain, USA

5d. PROJECT NUMBER
15G264

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-16-M-054

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of Homeland Security ICS-CERT
POC: Neil Hershfield, DHS ICS-CERT Technical Lead
ATTN: NPPD/CSC/NCSD/US-CERT Mailstop: 0635
245 Murray Lane, SW, Bldg 410, Washington, DC 20528
Email: ics-cert@dhs.gov phone: 1-877-776-7585

10. SPONSOR/MONITOR'S ACRONYM(S)

DHS ICS-CERT

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

First responders go through rigorous training and evaluation to ensure they are adequately prepared for an emergency. From a cyber security standpoint, however, this same set of criteria and rigor is severely lacking. This research provides a framework for evaluating the readiness of cyber first responders responsible for critical infrastructure protection. The framework demonstrates the development of evaluation environment, criteria and scenarios that are modeled from NFPA 1410 standards concept that is used for assessing the readiness of firefighter first responders. The utility of framework is exhibited during a military cyber training exercise and demonstrates the ability to evaluate the readiness of cyber first responders when responding to the cyber-based attacks in the scenarios. In addition, the results and analysis from the exercise provide a context to develop a physical processes simulation tool, called Y-Box. The Y-Box creates more accessible, representational, realistic and evaluation-friendly environment to enhance the framework. The Y-Box demonstrates its successful application through the simulation of the first two stages in a wastewater treatment plant.

15. SUBJECT TERMS

Cyber Exercise, Critical Infrastructure Protection, First Responders, Physical Processes Simulation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Mason Rice, AFIT/ENG
					19b. TELEPHONE NUMBER *(include area code)*
U	U	U	UU	127	(937) 255-3636, 4620; mason.rice@afit.edu