USING THE IEEE C37.118 PROTOCOL TO ADD HARDWARE IN THE LOOP TO THE CYPRES TESTBED

A Master Thesis

by

KOLTEN CHARLES KNESEK

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Dr. Ana Goulart
Committee Members,	Dr. Katherine Davis
	Dr. Wei Zhan
Head of Department,	Jorge Leon

December 2021

Major Subject: Engineering Technology

ABSTRACT

The Cyber Physical Resilient Energy Systems (CYPRES) project at Texas A&M University aims to create a next generation of energy management systems (EMS) using a secure cyber-physical systems (CPS) modeling foundation. The CYPRES research team has created the Resilient Energy Systems Lab (RESLab) cyber-physical testbed. Our primary contribution to this project was to implement Hardware-in-the-Loop (HIL) to create a full end-to-end testing ground for the RESLab testbed. This HIL would be capable of taking in voltage and current phasor measurement data inputs from a power grid simulator using the IEEE C37.118 industry standard protocol and output actual voltage signals to a relay protection system. In this way, the relay can be tested and perform actions as if it were connected to a full-scale power grid network. This was accomplished by utilizing PowerWorld Dynamic Studio (PWDS) for the power grid simulator, National Instruments' (NI) CompactRIO platform for the voltage generator, and a Schweitzer Engineering Laboratories (SEL) Protection Relay for detecting the voltages.

Another major contribution of this work was designing and testing false data and command injection cyber attacks on the IEEE C37.118 protocol and testing different machine learning (ML) classifiers to determine their level of accuracy in effectively detecting the cyber attack on this industrial protocol. This was completed by using PowerWorld Dynamic Studio (PWDS) power simulator to generate the C37.118 traffic and Common Open Research Emulator (CORE) to emulate the network where the C37.118 packets are transmitted. The attack scripts were written using Python programming language network packet dissector called Scapy. The three different machine learning classifiers evaluated were k-Nearest Neighbor, Decision Tree, and Naïve Bayes.

Finally, our HIL implementation can also help answer the research question of what constitutes a relay misconfiguration and how this misconfiguration can be detected. The misconfiguration can be caused by human error or a malicious user trying to perform a cyber attack on the grid.

DEDICATION

To my mother, father, brother, sister, and girlfriend. For all the love and support they have given.

ACKNOWLEDGMENTS

First, I would like to thank Dr. Ana Goulart for her guidance as my advisor and all the work she has done to get me to this point. I would not be where I am without her.

I would like to thank my friend and colleague Patrick Wlazlo for always being someone to talk to as we have gone through this graduate process together. He also helped me to create and conduct the C37.118 attacks and machine learning.

I would like to thank Dr. Katherine Davis for providing me with my project and giving me the opportunity to contribute to this exciting testbed.

I would like to thank Hao Huang for working with me to understand the goal for Hardware-In-the-Loop and for always being there to help.

Lastly, I would like to thank all the amazing professors at Texas A&M for all the knowledge and skills they have taught me over the years that have allowed me to complete this project.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professors Dr. Ana Goulart and Dr. Wei Zhan of the Department of Engineering Technology and Industrial Distribution and Professor Dr. Katherine Davis of the Department of Electrical and Computer Engineering.

All other work conducted for the thesis was completed by the student, under the advisement of Dr. Ana Goulart of the Department of Engineering Technology and Industrial Distribution.

Funding Sources

Graduate study was supported by the US Department of Energy Cybersecurity for Energy Delivery Systems program under award DE-OE0000895.

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iii

ACKNOWLEDGMENTS iv

CONTRIBUTORS AND FUNDING SOURCES v

TABLE OF CONTENTS vi

1. INTRODUCTION AND LITERATURE REVIEW 1

    1.1 Introduction 1
        1.1.1 Problem Identification 1
    1.2 Literature Review 2

2. Description of Project 5

    2.1 Project Overview 5
    2.2 Hardware in the Loop Components 5
        2.2.1 IEEE C37.118 6
        2.2.2 Simulated Power Grid 8
        2.2.3 NI CompactRIO 11
        2.2.4 NI LabVIEW 14
            2.2.4.1 IEEE C37.118 Control Mode Function 14
            2.2.4.2 Manual Control Mode Function 15
        2.2.5 NI FPGA 16
        2.2.6 SEL-351 Protection Relay 18
        2.2.7 HIL Testbed 20

3. Implementing Hardware In the Loop in the RESLab Testbed 23

    3.1 HIL Operation Program 23
        3.1.1 C37.118 Operation Mode 23
        3.1.2 Manual Mode 25
    3.2 Verifying the HIL Outputs 27
        3.2.1 AcSELerator QuickSet 27

4. Testing and Detecting Relay Misconfiguration 30

4.1 Setup for New PowerWorld Testcase 30
4.2 Relay Fuzzing 32

5. MiTM Attack on Synchrophasor Protocol and ML Detection 40

5.1 IEEE C37.118 Packet Dissection 40
5.2 Testbed Architecture 41
5.2.1 PowerWorld VM 44
5.2.2 Client VM 44
5.2.3 CORE VM 46
5.3 False Data and Command Injection 47
5.4 Data Set Collection and Generation 48
5.5 Machine Learning Classifiers 50
5.5.1 K-Nearest Neighbor 52
5.5.2 Decision Tree 53
5.5.3 Naive Bayes 53
5.6 Results 54

6. Summary and Conclusions 56

6.1 Contributions 56
6.1.1 Lessons Learned 56
6.2 Future Work 58

REFERENCES 59

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

1.1.1 Problem Identification

The fourth industrial revolution has seen the implementation of Internet of Things (IoT) devices into everything from smart appliances to smart cars, and the power grid is no exception. As more and more communication devices are added to create the next generation smart grid, the need for cyber-attack mitigation becomes more significant. This requires a power grid testbed that can be used for end-to-end testing of different types of cyber-attacks as well as allow for the development of different preventative and mitigation techniques.

The current Resilient Energy Systems Laboratory (RESLab) testbed [1, 2] contains all the current industry standard power-grid network equipment. It can simulate different power grid networks but has no way of taking the values from the simulated grid and using it to drive the physical equipment so that the equipment operates as if it were directly connected to a power grid. This dissertation focuses on how can we implement hardware-in-the-loop (HIL) to bridge this gap between simulation and physical hardware as well as identify any new vulnerabilities that would be associated with this. An example of the RESLab testbed updated to include the HIL is shown in Figure 1.1. The RESlab testbed currently has a utility control center, a communications network, and a substation. The HIL will add a protection relay device that will be connected to the real-time automation controller (RTAC) in the substation.

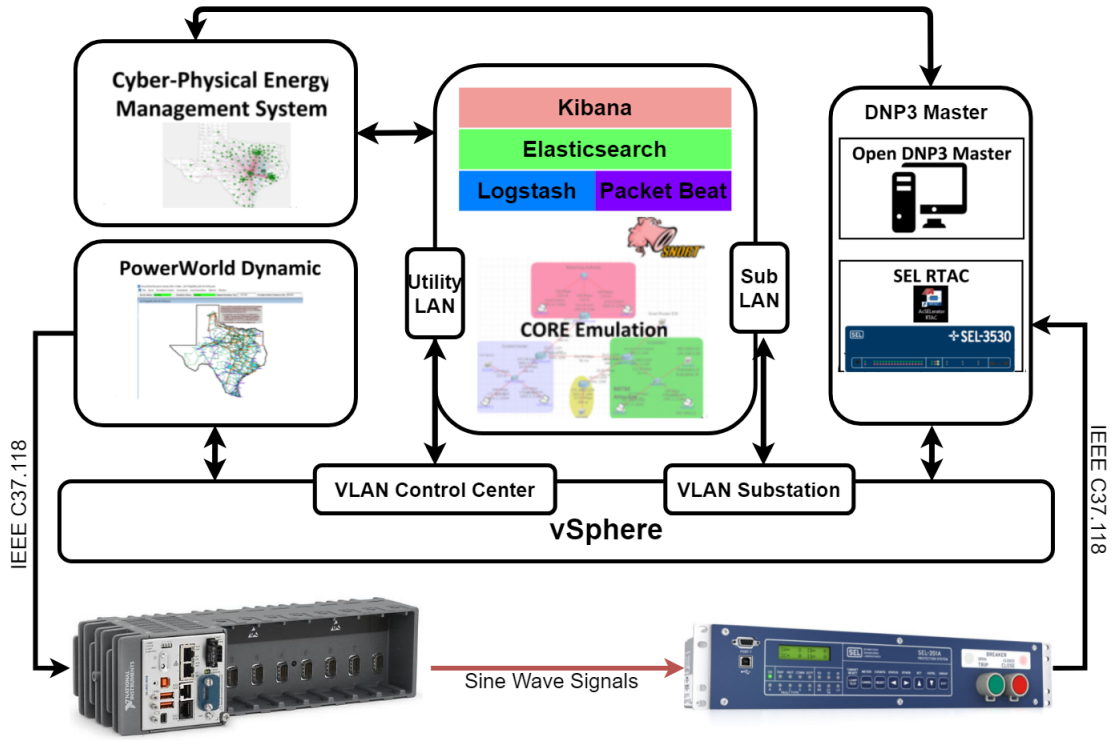

Figure 1.1: RESLab Testbed with Hardware-In-the-Loop

1.2 Literature Review

We have been working on the Cyber Physical Resilient Energy Systems (CYPRES) [1] project to develop a way for driving the power grid protection relays currently installed in the ResLAB Testbed without using high voltage power supplies. There are other groups that have implemented HIL for power grid testbeds but only one of them have done it using low voltage inputs. This is what my work has been focused on improving upon.

The article in [3] discusses how the PRIME testbed at Pacific Northwest National Laboratory (PNNL) is an improvement over current power system testbeds because they introduce Remote Hardware in the Loop (RHIL) into what is normally a software-simulation-only testbed. Most power system testbeds use industry standard simulation programs to conduct their test and training. The first problem with these software simulators is that the communication protocols are typically abstracted/over-simplified to the point that cyber-defense testing is not feasible. The sec-

ond problem is that these simulators do not replicate the latency between detecting a line fault and the time to recover because most simulators are event-driven simulators, which do not follow a real-time clock but execute to the next scheduled event [4]. These problems are solved by adding RHIL to physically represent a substation environment. The RHIL is constructed using three main components: a power grid simulator to generate the synchrophasor values, a National Instruments (NI) Compact RIO to convert the digital synchrophasor measurement values from digital to analog, and power protection relays to read the analog inputs from the Compact RIO as real signal inputs from a power grid. By physically representing a substation, a variety of new test scenarios can be created to test network latency and integrity.

The purpose of our work is to implement a HIL solution similar to the RHIL by PNNL, but improve upon it to also include a manual control option to allow for greater control of the relay input variables.

The article in [5] discusses a new type of cyber-physical testbed that utilizes real time models of power grid network models, power hardware-in-the-loop, and physical hardware such as inverters and system controllers. It allowed for accurate power grid network testing over all components that are typically used in the modern power grid. By using the actual hardware instead of only simulating it, these tests also account for any added latency caused by the hardware that a simulation would not show.

Unlike the testbed in [5], our approach to add hardware-in-the-loop does not require the high power and expensive power grid generation equipment that is necessary to drive the different power grid relay devices.

Also, in our experiments the main communication protocol used was the IEEE C37.118, an industrial application layer protocol for the transmission of Synchrophasor data packets for power systems [6].

For the other major contribution of this work – conducting cyber attacks the IEEE C37.118 protocol – several other works were examined to see how they compared with our work. In the first related work [7], the authors focused on designing a Denial-of-Service (DoS) attack on the IEEE

C37.118 protocol that would prevent packets from being received by a phasor data concentrator (PDC). The authors then created a way for the IEEE C37.118 traffic to do Multipath-Transmission Control Protocol port hopping in order to mitigate the DoS attack.

In another related work [8], the authors focused on modifying the global positioning system (GPS) timestamps that are received by phasor measurement units (PMU) so that the units report incorrect GPS times. Unlike these two other works, our attack on IEEE C37.118 focuses on intercepting and manipulating the packets while they are in transit as well as modifying the phasor values.

2. Description of Project

2.1 Project Overview

The primary goal of this project was to develop a way to take simulated synchrophasor (phasors with amplitude, phase, and frequency values) AC voltage and current measures from the PowerWorld Dynamic Studio (PWDS) simulator and feed them to a protection relay. This connection to the relay needed to be done in a way that utilized low voltage inputs to the relay but still had the relay operate as if these inputs were of medium or high voltage. For example, if the simulated voltage from PWDS is 75 kVolts, the inputted value to the relay needs to be 0.615 Volts in order for the relay to show that it is reading 75 kVolts. This makes the relay operate as if it is actually connected to a power grid. This then allows for cyber-physical testing of the relay and any other connected components in an environment that represents normal power grid operation without the need for full scale power grid generation equipment.

Another goal of this project was to test the resilience of these systems against cyber threats in order to demonstrate the need for cyber-physical testing on power grid networks. This was done by compromising one of the industry standard communication protocols, i.e., the IEEE C37.118 protocol, that carries the phasor measurement units (PMU) to add for false data and command injection.

2.2 Hardware in the Loop Components

The Hardware-In-the-Loop for the RESLab testbed can be broken into several major components. The first component is the IEEE C37.118 Synchrophasor protocol which is the communication protocol used by power grid systems to transmit phasor measurements. The second component is PowerWorld Dynamic Studio (PWDS), a power system simulator that generates the simulated phasor measurements. The third component is the NI CompactRIO which is used to generate analog voltages from the information in the C37.118 data packets. The last component is the Schweitzer Engineering Laboratories 351 (SEL-351) Protection Relay which is a common

power grid device that monitors transmission lines. These are the main components that make up the HIL.

2.2.1 IEEE C37.118

The IEEE C37.118 standard is used for the transmission of synchrophasor data measurements from a Phasor Measurement Unit (PMU) to a Phasor Data Concentrator (PDC). The article in [6] goes into detail on the structure and contents of the IEEE C37.118 packet. A C37.118 packet exists in the application layer of a data packet. These packets can run over any type of transmission medium and can be transmitted using either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) as well as serial connections. The TCP or UDP communication follows a client-server architecture. In our case, the TCP client was the NI CompactRIO and the TCP server was PWDS.

There are three main formats of a C37.118 packet: command, configuration, and data. These are shown in Figure 2.1. All three packet formats contain the same information blocks with the only difference being the command, configuration, and data blocks themselves.

Command frames are sent from a client to the server and will include one of three different commands for negotiation of what information the server should send. The first command is Data Transmission Off. This informs the server to end the transmission of data frames to the client. The second command is Data Transmission On, which informs the server to begin sending the Data Frames to the client. The third is Send Configuration Frame, which requests the configuration frame from the PMU.

Configuration frames contain the necessary information about the structure of the data frames so that the client will know how to parse the data. These are sent from the server to the client. Typical parameters contained within this type of frame include time resolution, number of stations, station IDs, number of PMUs in each station, phasor and station names, conversion factors and nominal line frequencies for each PMU, and the rate of transmission of data frames.

Command Frame

Synchronization word	Framesize	PMU/DC iD number	SOC time stamp	Time Quality Flag	Fraction of second (raw)	Command data	CRC
2 Bytes	2 Bytes	2 Bytes	4 Bytes	1 Byte	2 Bytes	2 Bytes	2 Bytes

Configuration Frame

Synchronization word	Framesize	PMU/DC iD number	SOC time stamp	Time Quality Flag	Fraction of second (raw)	Configuration data	CRC
2 Bytes	2 Bytes	2 Bytes	4 Bytes	1 Byte	2 Bytes	Varies	2 Bytes

Data Frame

Synchronization word	Framesize	PMU/DC iD number	SOC time stamp	Time Quality Flag	Fraction of second (raw)	Measurement data	CRC
2 Bytes	2 Bytes	2 Bytes	4 Bytes	1 Byte	2 Bytes	Varies	2 Bytes

Figure 2.1: IEEE C37.118 Packet Frames

Data frames contain the measured values taken by PMUs and other relevant data such as flags per station. These frames are sent from the server to the client. The phasor measurements received per PMU are raw unscaled integers in rectangular form. The voltage value is an unsigned short integer and the phase angle is a signed short integer. In order to get these integers to represent their actual values, they must be multiplied by their corresponding conversion factor that was sent in the configuration frame and then converted from rectangular to polar. Data frames also include frequency deviation from the nominal as a signed short integer that needs to be added to its nominal line frequency sent in the configuration frame. The last value is the rate of change of frequency as a signed short integer. An example of an actual C37.118 data packet is shown in Figure 2.2, which carries information of three stations. Station 1 measurements are highlighted in the figure.

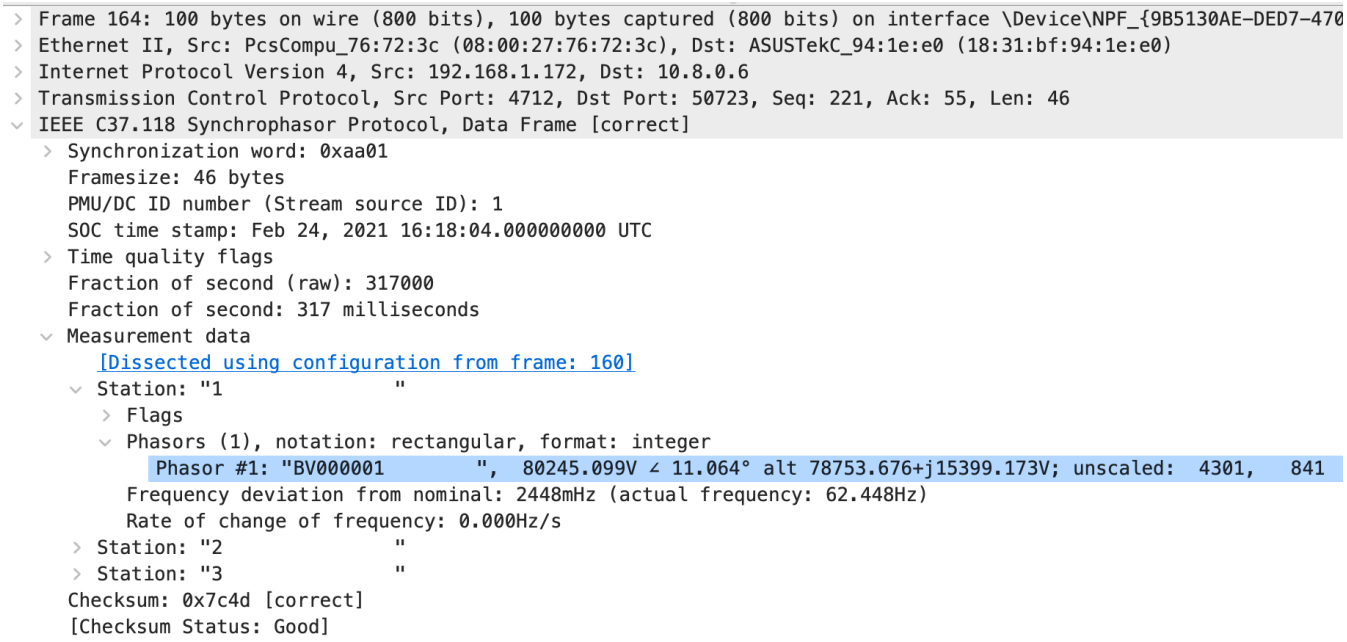

Figure 2.2: An example of an IEEE C37.118 data packet for a three-bus substation.

There is a standard header frame which is the same for all C37.118 frames. The first two bytes of every C37.118 frame is a synchronization word which indicates which of the three C37.118 frames it is: $0xAA41$ for Command Frame, $0xAA31$ for Configuration Frame, and $0xAA01$ for Data Frame. The header includes other useful information such as the framesize, PMU/DC ID number, SOC timestamp, time quality flags, and the fraction of a second the packet was sent. The header is followed by the payload specific to its frame type. At the end of the frame there is a two-byte checksum that is calculated using CRC-16/CCITT-FALSE.

2.2.2 Simulated Power Grid

PowerWorld [9] is a software package used for simulating high voltage power grid networks. This package allows for testing different power grid failures to predict the outcome and thus implement real-world efforts to prevent any serious failures from occurring. There are two main components to this software package that are relevant to this paper, PowerWorld Simulator and PowerWorld Dynamic Studio.

PowerWorld Simulator is used for the design and development of different power grid net-

works. It can be used to model a power grid network such as buses, transmission lines, generators, and loads. Once the network parameters are created, events, such as a generator going down, can be set to occur at specific times during the real-time simulation and the simulator will adjust based on that event.

PowerWorld Dynamic Studio is used to run the simulation created with PowerWorld Simulator in real-time. The real-time values can be viewed within the program but can also be sent to other programs or physical hardware. The most relevant of these is IEEE C37.118 protocol which is utilized to gather data from PowerWorld as if it were a phasor data concentrator (PDC). This ability of PowerWorld to output PMU data over other power system mediums makes it useful in testbeds that want to physically replicate just one section of a much larger system but still want to validate their network against the entire system.

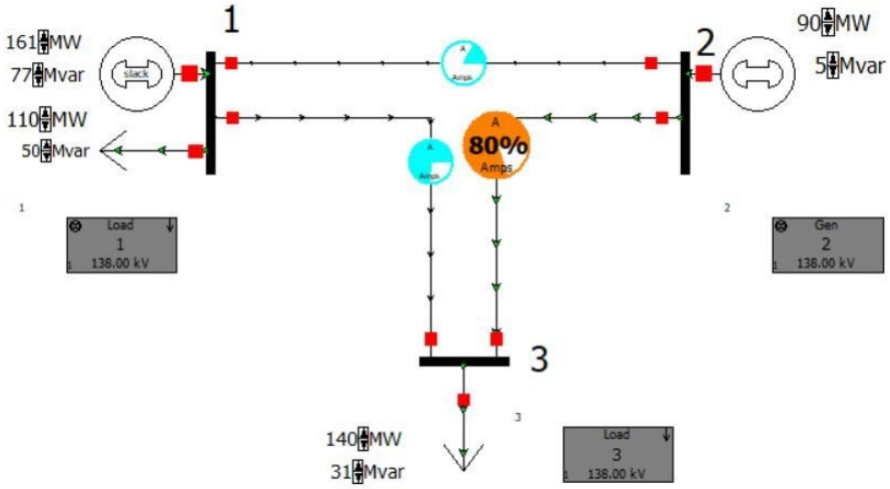

Figure 2.3: Small Three-Bus PowerWorld DS Model

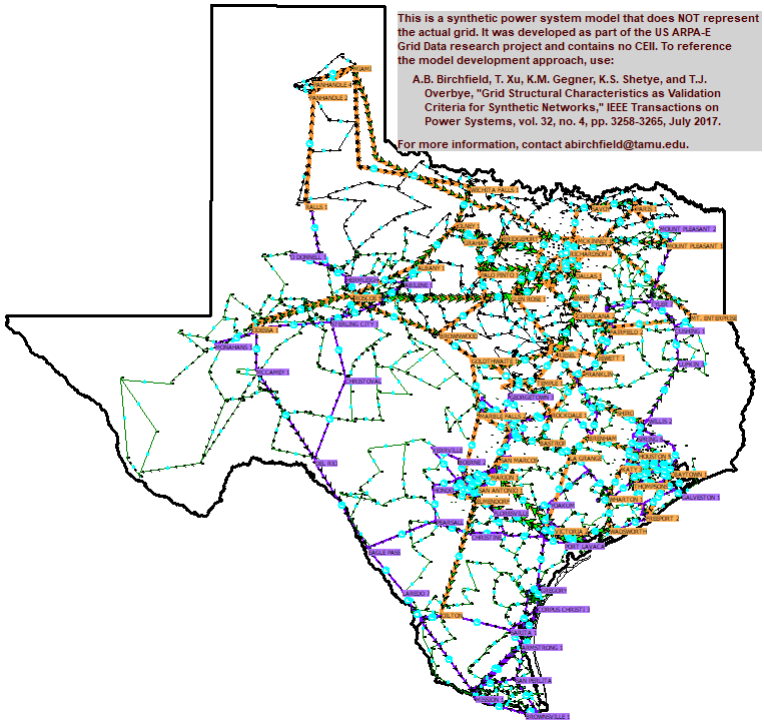

Figure 2.4: Texas 2000-Bus PowerWorld DS Model

During the development of this project, several PWDS simulation models were used. For the initial development of this project, a small three-bus test case, shown in Figure 2.3, was used to develop and test a major portion of the code written for the NI CompactRIO. The benefit of this this three-bus case was that it had all the necessary information for IEEE C37.118 but with just three buses so that the packet sizes were small and easy to learn to dissect. Once the HIL was able to be conducted using the three-bus case, work was done to allow the LabVIEW program to scale up to a much larger case. Specifically, the Texas 2000-bus Synthetic model [10, 11], which is the primary model used for testing the RESLab testbed, was the desired power grid simulation for running the HIL. This simulation represents a synthetic power system model that includes 2000 different buses and hundreds of substations that resemble the Texas power grid. Figure 2.4 shows the Texas 2000-bus model in PowerWorld simulator.

2.2.3 NI CompactRIO

National Instruments (NI) CompactRIO system [12] is a platform of modular devices that can be customized for different applications. They are used for high-speed, high-performance data processing that utilizes multiple inputs and outputs and is easily programmable using National Instruments LabVIEW software tool. The benefits of this system for this project are its integrated field programmable gate arrays (FPGA) for generating the necessary AC sine waves as well as its network connected real-time processor which is used for LabVIEW network programming. With these tools, we created a program that establishes a client TCP connection to a power grid simulator (PWDS) to receive synchrophasor measurement data units in the format of C37.118 data packets. Our program reads those PMU packets to output relative voltages to a protection relay. The specific module used for this research contains an analog output module which is used to output the AC sine waves.

The specific National Instruments equipment selected for this are a cRIO-9035 and a NI-9264. The cRIO-9035, shown in figure 2.5, is an eight-slot embedded controller that contains a field-programmable gate array (FPGA) which is used for generating the analog AC Sine Waves. The second component is the NI-9264 C Series Voltage Output Module which is installed in one of the eight module slots of the cRIO-9035. The NI-9264 module, shown in figure 2.6 is a 16-channel voltage output device capable of outputting 16 different voltages between +10 and -10 Volts. This device is used for outputting the AC sine waves. Output connections for the NI-9264 are attached using terminal blocks where terminals 1-16 are the analog outputs and 20-35 are the corresponding grounds for each analog output channel, shown in table 2.1.

Table 2.1: Analog Outputs and Corresponding Grounds for NI-9264

Analog Output Terminal	Corresponding Ground Terminal
1	20
2	21
3	22
4	23
5	24
6	25
7	26
8	27
9	28
10	29
11	30
12	31
13	32
14	33
15	34
16	35

Figure 2.5: Nation Instruments cRIO-9035

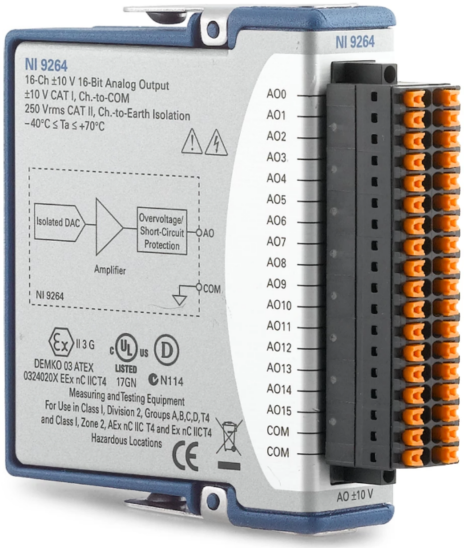

Figure 2.6: National Instrument NI-9264

To conduct digital to analog voltage conversions using a NI CompactRio, a feature known as NI CompactRIO FPGA Mode is utilized. This feature was used to generate the sine waves generated by the Compact RIO. PMU measures sent from PowerWorld DS are parsed using a custom

LabVIEW digital to analog program. This program then scales down the data measurements and uses custom FPGA program to output the analog voltage to the NI 9264 Analog Output Module attached to the CompactRIO. These outputs of the NI 9264 are connected to the Low-Level Test Interface of the SEL-351 protection relay system.

2.2.4 NI LabVIEW

National Instruments (NI) LabVIEW [13] is graphical programming language used for rapid development of applications requiring visualization of the inputs and outputs. Specifically for this project, LabVIEW is utilized to configure and control the NI CompactRIO. The program developed is capable of operating in two different modes: IEEE C37.118 control, and Manual Control. As such, the HIL Code is split into two functions based on the two modes. Besides the standard NI LabVIEW packages, another package called Frame APIs for PDC by NI [14] was also installed to simplify the addition of C37.118 protocols into the program.

2.2.4.1 IEEE C37.118 Control Mode Function

The IEEE C37.118 control mode establishes a TCP network connection to the PowerWorld DS simulation model to receive IEEE C37.118 synchrophasor data packets, process those packets to extract the desired values, generate corresponding sine waves, and output those sine waves over the analog output module on the NI CompactRIO connected to the low-level test interface on the SEL-351 protection relay. The C37.118 control mode is broken up into three sequences: initialization, main loop, and shutdown.

The initialization sequence of the C37.118 control mode function begins by first establishing a TCP connection to the PowerWorld simulation. This is then followed by sending a C37.118 command packet requesting the configuration frame. The program then waits until it receives the entire configuration frame and processes it. Lastly, another command packet is sent requesting that the data stream from PowerWorld be started. The initialization sequence also opens the FPGA program.

The main loop sequence of the C37.118 control mode function consists of a *while* loop that

runs all of the main loop code until the operator selects to end the program. The first item inside the *while* loop is a TCP receive function which waits until it receives a new data packet from PowerWorld. When a new data packet is received, it is parsed and dissected using the information from the configuration packet collected in the initialization sequence. From this information, the user then is able to select which specific PMUs' values should be outputted to the SEL-351. These values are then passed into a function that generates the three phases for each PMUs. This is required since PowerWorld only outputs one phase value for each PMU but three are needed to create the three-phase sine waves. This function generates a waveform graph of what the expected output from the CRIO should be. This is all done twice, once for the voltage and again for the current. All this information is then exported into the FPGA program which is run in parallel with the HIL program.

The shutdown sequence is entered after the operator has ended the *while* loop in the main loop sequence. This sequence does three things. It first closes the FPGA program so that it does not continue to run once the HIL program ends. Second, it sends a final command packet to PowerWorld telling it to stop sending data packets. Lastly, it terminates the TCP connection.

2.2.4.2 Manual Control Mode Function

The Manual Control Mode Function operates in a similar way to the IEEE C37.118 control mode except that instead of receiving the synchrophasor control values from PowerWorld DS, the values are entered manually to LabVIEW and are then used to generate the sine wave outputs. This mode allows for greater manipulation of the sine waves being generated and can create more complex test cases for testing the SEL-351 relays, such as testing relay misconfiguration.

Other than the open and close functions for the FPGA, the rest of the code for the Manual Control mode function exists inside a *while* loop that runs until the operator ends the HIL program. Inside the *while* loop, the string values from the table on the Front Panel are extracted. These string values are then converted into double precision. This allows mathematical operation to run on these values and they can be used to drive the FPGA program. Other than the magnitude values, which have to be scaled down to a voltage for the low-level test interface, these values are exported

directly to the FPGA program that is running in parallel. These values are also shown using a wavefrom graph so that the operator can visualize the waveforms being sent out. Once all the values have been sent to the FPGA program, the HIL program waits until the duration set for these values have passed before being updated. The *while* loop allows the program to loop through all the rows in the Front Panel table until it reaches an empty row or the operate force stops it, at which point the HIL and FPGA programs terminate.

2.2.5 NI FPGA

One of the key features of the NI CompactRIO are the built-in FPGA modules. These modules allow for rapid development of programs that require high refresh rates which cannot be achieved by software only. It is these FPGA modules that were utilized to generate the different AC sine waves that are outputted from the CompactRIO.

The CompactRIO has two different operation modes, Scan Mode and FPGA mode. The benefits of Scan Mode are that it is simpler to develop new programs in and easier for users that are new to CompactRIO to learn since it includes a library of prebuilt modules for FPGA and does not require the user to directly design any FPGA code. The limitation of this, however, is that Scan mode has a limited refresh rate of about only 1 kHz. Most power grid sine waves operate about 60 Hz when combined with only a 1 kHz refresh rate. This results in each individual sine wave only having a refresh rate of about 17 times per cycle, which created waves with steps instead of a normal sine wave. This made the use of Scan Mode not a viable option for this project. This required the use of FPGA Mode since it is capable of producing designs with MHz of refresh rates which produces more accurate sine waves.

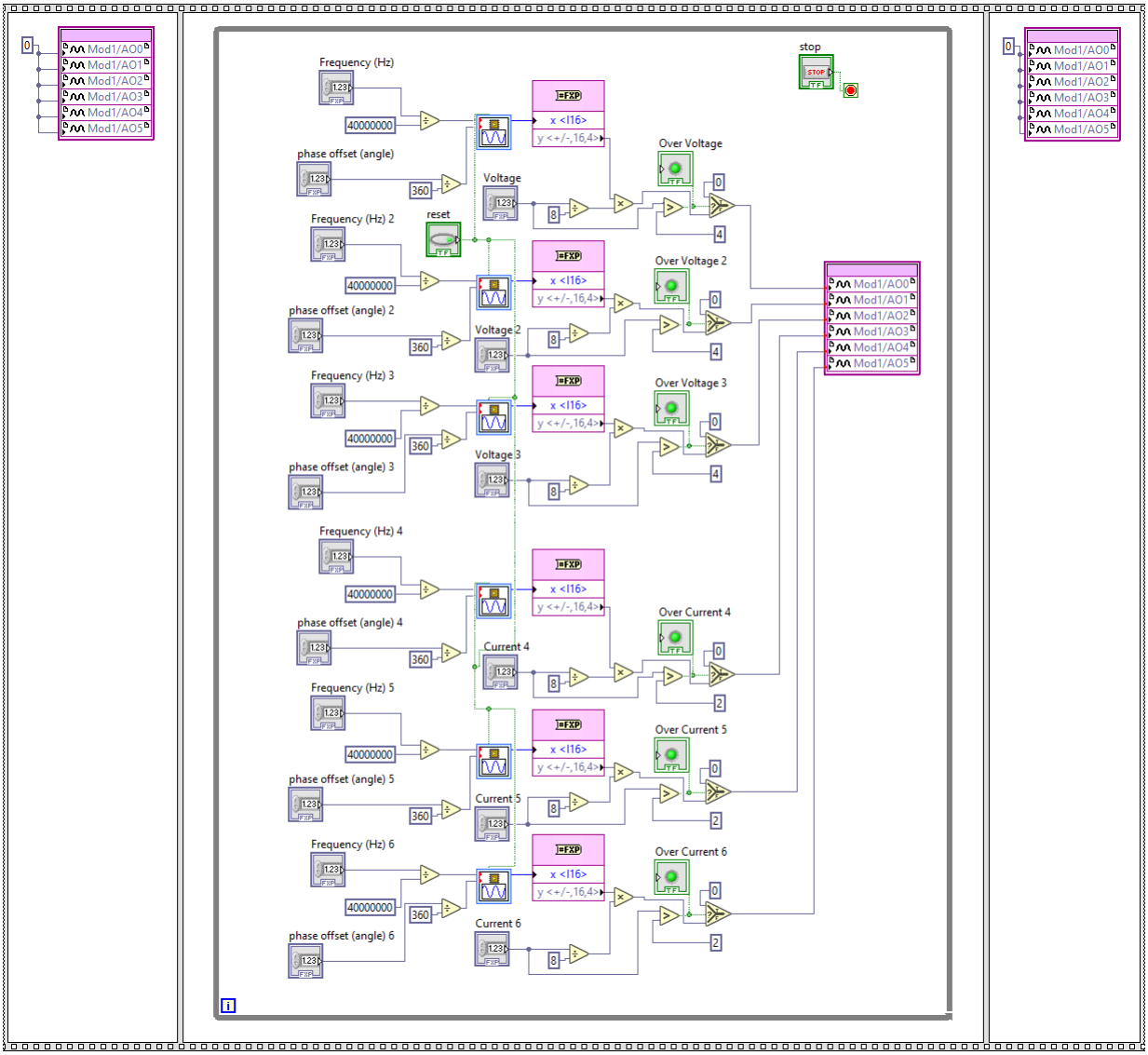

Figure 2.7: HIL Program FPGA Code

The LabVIEW FPGA program in this project, shown in figure 2.7 consist of four parts: Inputs, sine wave generation, voltage protection, and outputs. There are 20 inputs into the FPGA program. Since the SEL-351 takes in three-phase voltage and current, the FPGA has to create six sine waves, three for voltage and three for current. Each sine wave has its own magnitude, frequency, and phase control. This makes up eighteen of the inputs. The remaining two inputs are a boolean stop for ending the FPGA program and a boolean reset for resetting the waveforms to their set values. The

waveform generation is done by a built-in LabVIEW FPGA function for sine wave generation with inputs from frequency and phase. The output from the sine wave generator is then converted to a corresponding value between -4 and 4. This value is then adjusted using the magnitude. Once the final output is created, it is then verified to make sure the desired value does not exceed the maximum input value for the SEL-351 in order to prevent damage to the hardware. If the value exceeds the maximum, then a boolean indicator is activated. Otherwise, the final value is outputted using the NI-9264 Analog Output module. As a final layer of protection and to encourage proper startup and shutdown of the program, all analog outputs are set to zero at the start and end of the FPGA program.

2.2.6 SEL-351 Protection Relay

The Schweitzer Engineering Laboratories 351 (SEL-351) Protection Relay Systems [15] are a line of power grid protection relays that are used to detect a wide range of power grid faults and trip circuit breakers when those detected faults are outside of the specified parameters. These relays are common protection field devices used in utilities and industrial electrical systems. They are connected to the communication network and have an internet protocol (IP) address. Because of this, they are cyber-physical devices which makes them a focus of the CYPRES project.

The connection between the NI cRIO and the SEL-351 is done by utilizing the low-level test interface which can be accessed from inside the SEL-351 protection relay by removing the connector between J2 and J12 and attaching jumper wires, shown in figure 2.8. However, the main board still needs some of the connection from the lower board in order for the main board to be powered. These include two +15V, three +5V, BATT_RET, BATT_COM, BATT_DIF, and the six GND left of the VS pins. The low-level test pins, which are VA, VB, VC, IC, IB, and IA are connected to the NI-9264 module according to the wiring diagram in Figure 2.9. Utilizing this setup allows the SEL-351 to operate normally but instead of requiring the hundreds of Volts input to operate, it can operate on less than two voltages to drive the inputs.

With the SEL-351, we can provide a way to drive the voltage and current inputs on the relay using low voltages so that the relay operates as if it was directly connected to a power grid network.

This will expand the capabilities of the RESLab testbed and allow for future research testbeds utilizing HIL and the SEL-351 protection relay.

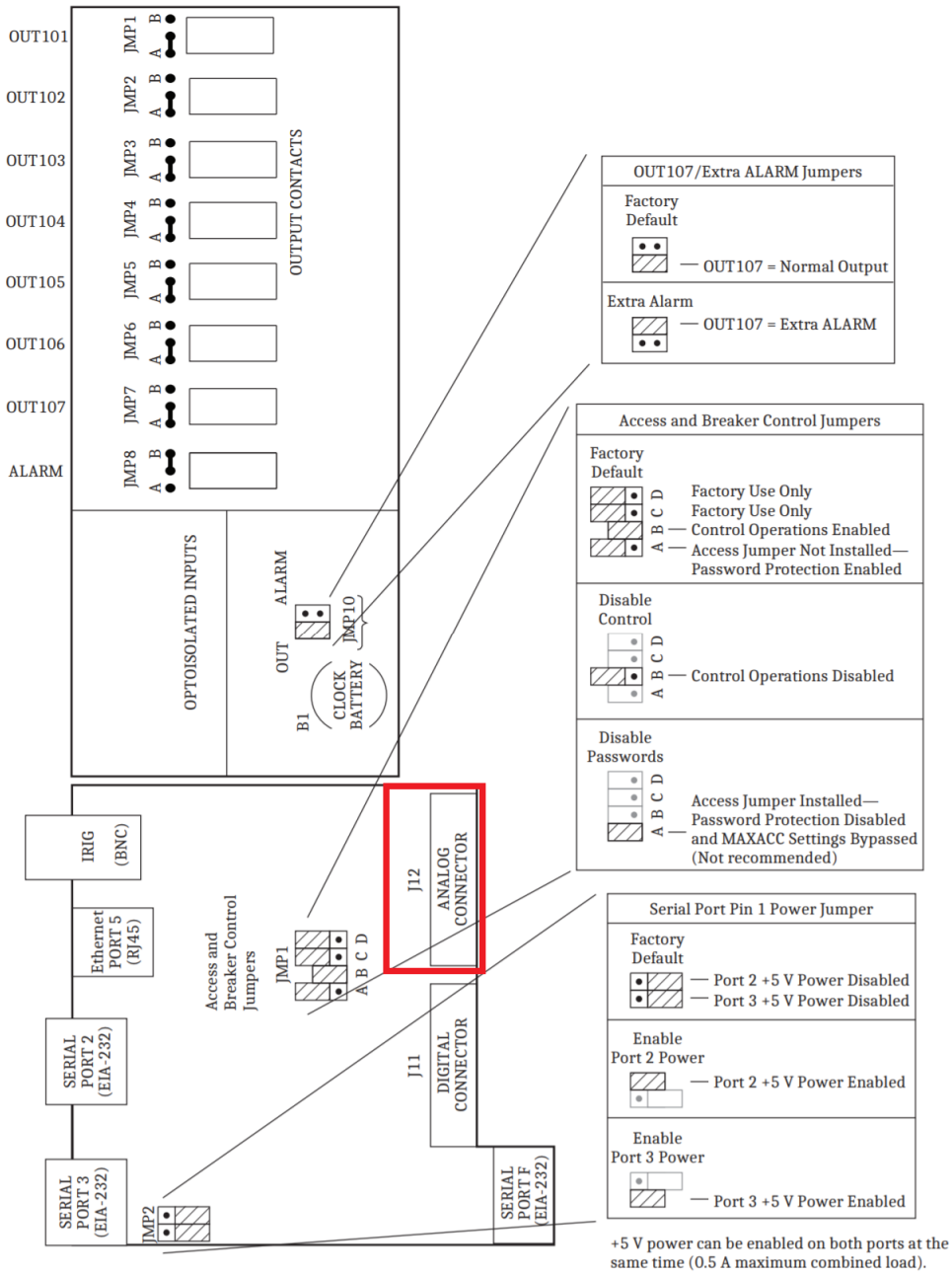

Figure 2.8: Jumper, Connector, and Major Component Locations on the SEL-351 Main Board

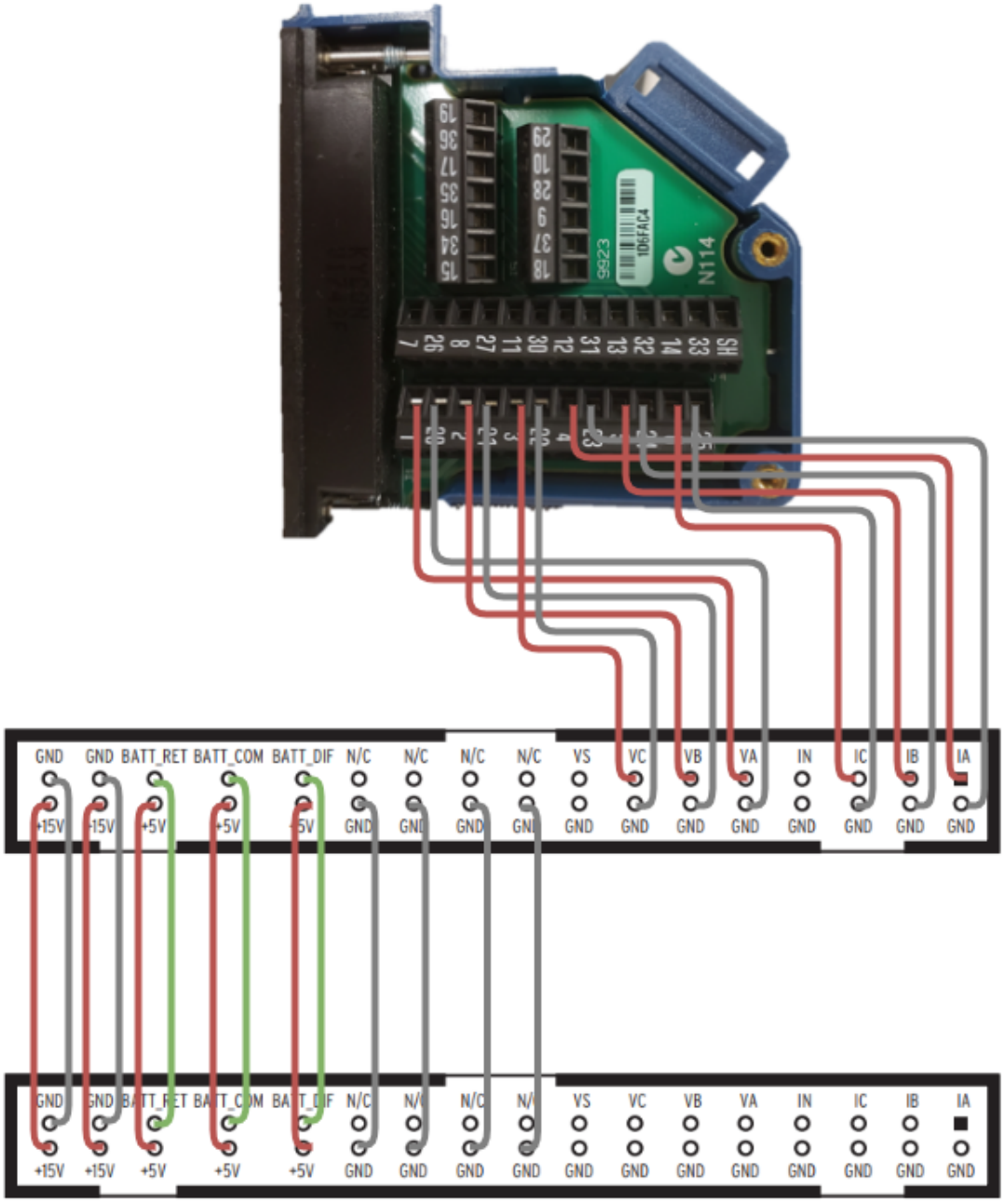

Figure 2.9: Low-Level Test Interface (J2 or J12) Connector

2.2.7 HIL Testbed

The hardware-in-the-loop (HIL) testbed created for this project consists of several components: PowerWorld DS, LabVIEW, NI CompactRIO with analog output module, and the SEL-351. Figure 2.10 shows the software and hardware components of our project. PowerWorld DS and LabVIEW

are Windows applications that are installed on network connected PCs that are attached to the same network as the NI CompactRIO. The LabVIEW interface is used to start the program on the NI CompactRIO, set any parameters, and view outputs. When the LabVIEW program is started, the NI CompactRIO initiates the IEEE C37.118 connection to PowerWorld DS. The SEL-351 relay is connected to the Analog Output Module on the NI CompactRIO using the relay's Low-Level Test Interface, which is used to bypass the high voltage inputs located on the exterior of the machine. There are six power connections made between the NI CompactRIO and the relay, three for three-phase voltage and three for three-phase current.

In order to show the benefits of the Hardware-In-the-Loop, we are utilizing the Manual control mode of the CompactRio to conduct relay fuzzing to look for any possible configuration within the SEL-351 protection relay. This is done by adjusting, for different periods of time, each of the different voltage and current parameters for magnitude, phase, and frequency to values that should trigger the relay to trip and noting whether the relay responds appropriately. By doing this, we are able to verify the functionality of the protection relay as well as check that the relay has been configured correctly for proper operation. Once initial testing with the HIL Testbed is complete, the HIL equipment will be integrated into the RESLab testbed so that it can be used to design new test case scenarios and and evaluate control applications utilizing the other components of the RESLab testbed.

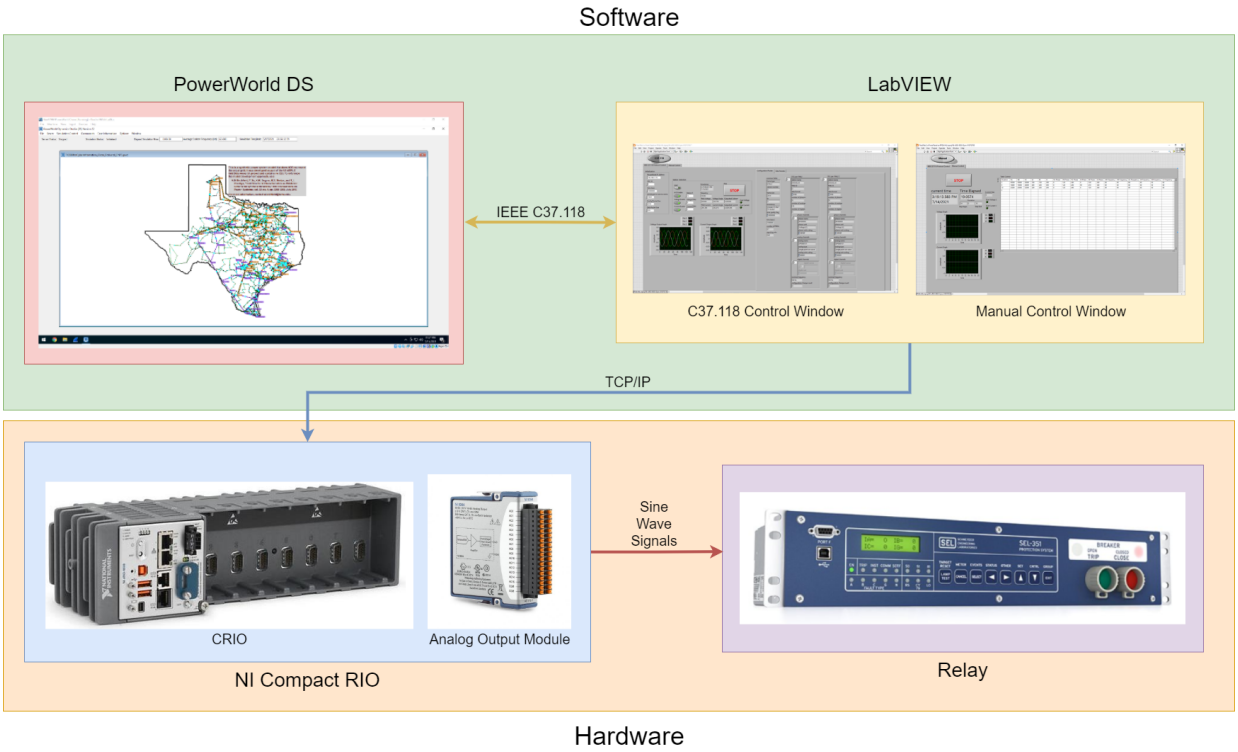

Figure 2.10: HIL Testbed Flow Diagram

3. Implementing Hardware In the Loop in the RESLab Testbed

3.1 HIL Operation Program

The HIL Program is a custom LabVIEW program built specifically to drive the Low-Level Test interface on the SEL-351 Protection Relay. This program allows for two different modes of operation, C37.118 Operation Mode, and Manual Operation Mode. The C37.118 Operation Mode allows the SEL-351 to be driven by a C37.118 synchrophasor packet stream. By driving the SEL-351 with a synchrophasor packet stream, the relay can operate off of the data from a power grid simulator, making it function similar to how it would if connected to an actual power grid transmission line. This proves a way to test the functionality of the relay without having to invest in expensive and complex high voltage power grid equipment to operate the relay. The Manual Operation Mode proved another way to control the SEL-351 but instead of requiring another source to drive it, it can instead be driven directly from the LabVIEW program. Using the Manual Operation Mode, variables of the sine wave signals inputted to the low-level test interface can be adjusted. This is beneficial as it provides a way to test relay misconfiguration.

3.1.1 C37.118 Operation Mode

The C37.118 operation mode, shown in Figure 3.1 is used by first setting the variables under the Initialization section before the starting the program. These values are used to properly establish the C37.1178 communication. If these values are not set correctly before the start of the program, the program will not be able to operate correctly and must be force ended. The required fields are the IP address of the server running PowerWorld DS, the PDC ID of the stream, the time base for the simulation, the port number for the C37.118 transmission, the size of the configuration packet in bytes, and the size of the data packet in bytes.

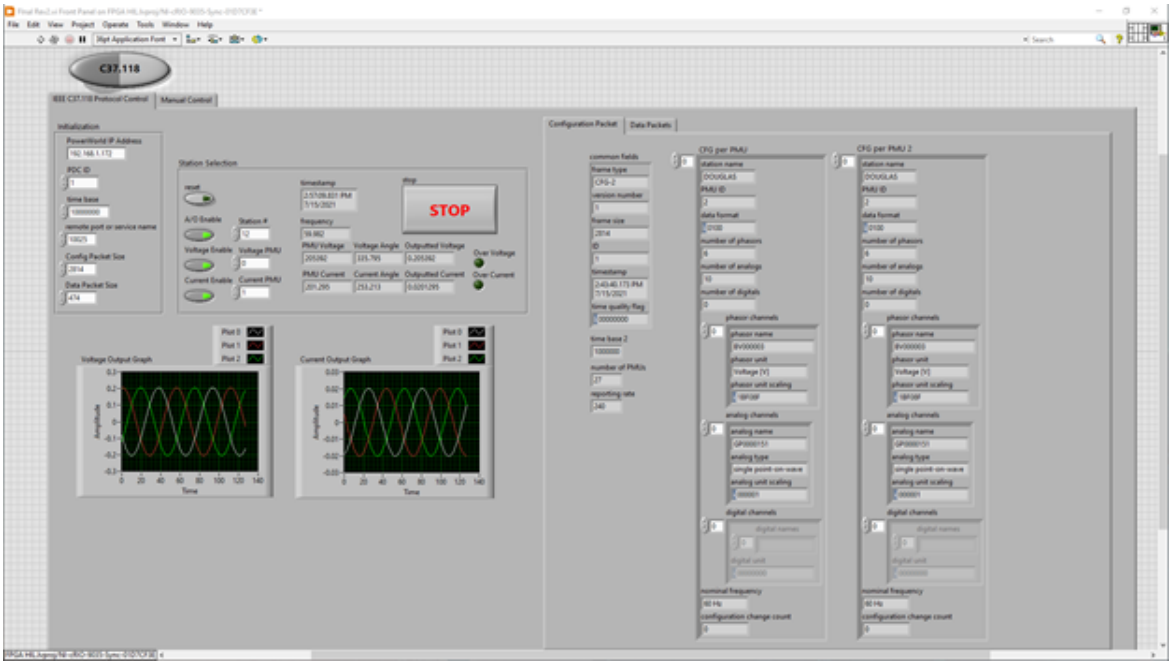

Figure 3.1: LabVIEW Front Panel IEEE C37.118 Control

The next section labeled Station Selection is used for selecting and controlling the desired PMU stream when there is multiple PMUs included in the C37.118 data stream. The Station # is the input for selecting which station the desired PMU is in. Voltage PMU and Current PMU are used to select the correct PMU values for each from the list of PMUs within the selected station. The A/O Enable is used to turn on and off all analog output, and Voltage Enable and Current Enable are used to either specifically turn on or off either the voltage or current analog outputs. The reset toggle switch is used to manually reset the output waveforms if one of them gets out of sync. Timestamp, Frequency, PMU Voltage, Voltage Angle, PMU Current, and Current Angle show the current timestamp, frequency, voltage magnitude, voltage angle, current magnitude, and current angle received from the most recent data packet respectively. Outputted Voltage and Outputted Current are the scaled down values of PMU Voltage and PMU Current that is outputted from the CompactRIO. Over Voltage and Over Current light up when either the desired voltage or current to be outputted is greater than the rated voltage for the relay. Whenever the voltage or current is over the rated limit, the values for that output are set to zero in order to prevent damage to the relay.

The large Stop button is used to properly stop and end the program. When this button is pressed, the C37.118 data stream is turned off, the analog output is set to zero, and the LabVIEW program is terminated. The graphs show the three-phase sine waves for both voltage and current as they are being outputted by the CompactRIO. Since PowerWorld only outputs one voltage phase value and one current phase value for each station and does not output magnitude and phase values for each sine wave within a three-phase electric power system, only VA phase and IA phase are set to the exact PowerWorld value. VB and IB are shifted 120 degrees from the values of VA and IA and VC and IC are shifted 240 degrees from the values of VA and IA. All three voltage sine waves are set to the same received voltage magnitude and all three currents are set to the same received current magnitude.

3.1.2 Manual Mode

The Manual operation mode, shown in figure 3.2 is used to create different test scenarios that allows the user to manipulate every aspect of the analog outputs. This operation mode is used by setting each of the different parameters within the Table Control to a desired testing value. Once all the desired test parameters have been set, the program can then be started and will run through each row of parameters for their set duration before moving on to the next row. When the program reaches a row that has no set duration value, the program will automatically terminate.

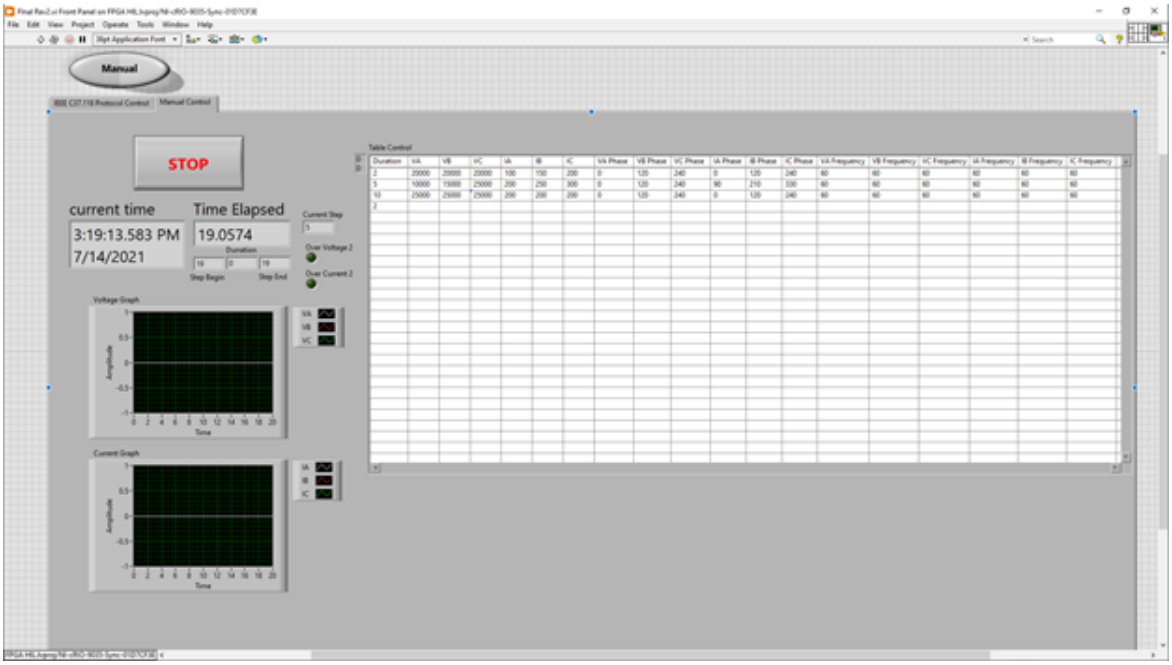

Figure 3.2: LabVIEW Front Panel Manual Control

The first column labeled Duration is used to specify how long, in seconds, the values for the rest of the parameters in that row should be set for. For example, if the duration for row 0 is set to 2, then the other parameter values will be set on the analog output for 2 seconds. The columns labeled VA, VB, and VC correspond to voltage magnitude for each independent sine wave of the three-phase voltage sine wave outputted to the relay. The columns labeled IA, IB, and IC correspond to current magnitude for each independent sine wave of the three-phase current sine wave outputted to the relay. The columns labeled VA Phase, VB Phase, and VC Phase correspond to voltage phase for each independent sine wave of the three-phase voltage sine wave outputted to the relay. The columns labeled IA Phase, IB Phase, and IC Phase correspond to current phase for each independent sine wave of the three-phase current sine wave outputted to the relay. The columns labeled VA Frequency, VB Frequency, and VC Frequency correspond to voltage frequency for each independent sine wave of the three-phase voltage sine wave outputted to the relay. The columns labeled IA Frequency, IB Frequency, and IC Frequency correspond to current frequency for each independent sine wave of the three-phase current sine wave outputted to the relay.

The Stop button on the left is used to immediately stop the program before the test has finished. The Current Time displays the current system time as a reference. Time Elapsed displays the time elapsed, in seconds, since the program was started. Current Step corresponds to which row is currently being processed, starting at 0. Step Begin is the time in Time Elapsed that the current step began, and Step End is the time in Time Elapsed the current step will end. Step Duration corresponds to how many seconds the current step will last. Over Voltage and Over Current light up when either the desired voltage or current to be outputted is greater than the rated voltage for the relay. Whenever the voltage or current is over the rated limit, the values for that output are set to zero to prevent damage to the relay. The graphs show the three-phase sine waves for both voltage and current as they are being outputted by the CompactRIO.

3.2 Verifying the HIL Outputs

Since the outputs from the CompactRIO are sine wave voltages with peak-to-peak values between zero and 4 Volts, these values can be seen on an oscilloscope; however, in order to determine what the actual voltages and currents that are represented by these values, the peak-to-peak voltages would need to be multiplied by their conversion values set in LabVIEW. This would be 11390 for the voltage values and 9300 for the current values. This would give the actual values that are being set as inputs into the LabVIEW HIL program.

3.2.1 AcSELerator QuickSet

AcSELerator QuickSet [16] is a program used to view the data received by the SEL-351 Relay. Once connected to the relay, the HMI is selected and opened. The information displayed in the HMI for the phasors is shown in Figure 3.3. This figure shows the different angles between each sine wave with relation to VA. It also shows the relative magnitude of the three-phase voltage's and current's individual sine waves in relation to one another. This window also shows the frequency of the received signal. The primary use of this display is to visualize how the relay is perceiving the received inputs from the CompactRIO, since the CompactRIO is not actually generating Kilovolts of voltage and hundreds of Amperes worth of current, but is instead using two conversion values

to adjust the voltage and current.

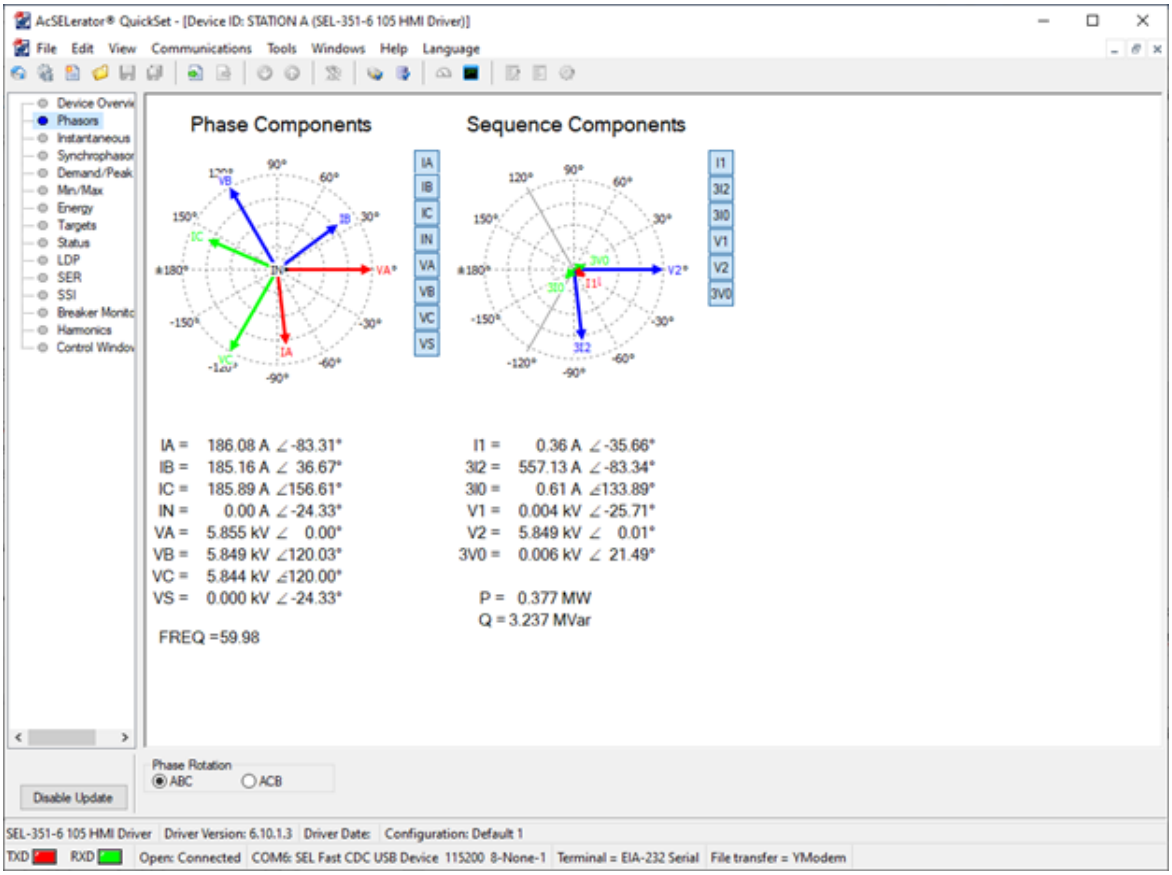

Figure 3.3: SEL-351 AcSELerator QuickSet Phasor Display

The reason for these conversion values is that since the values from the CompactRIO are meant to represent the values that would be expected by the low-level test interface on the SEL-351 Protection Relay, the voltages inputted into the HIL LabVIEW program must be converted as if they were being run through the transformers located on the SEL-351. These conversion values were found by taking the ratio between what the original high input was and what was being read by the relay. These conversion values were found to be static and did not need to be adjusted as the voltage and current values increased.

AcSELerator QuickSet can also be used to change and configure setting on the SEL-351 Pro-

tection Relay. This is done reading the setting from the device and adjusting any of the values accordingly. For example, using the default settings on the SEL-351, the maximum voltage that could be detected was only about 50 KiloVolts, which was an issue since the maximum voltage for many of the PowerWorld simulations would typically approach 200 KiloVolts. This was corrected by adjusting the relay settings for PTR Phase (VA, VB, VC) PT Ratio in Group 1/Set 1 from the default value of 180, shown in Figure 3.4 to 720. This allowed the relay to read maximum values of over 200 KiloVolts.

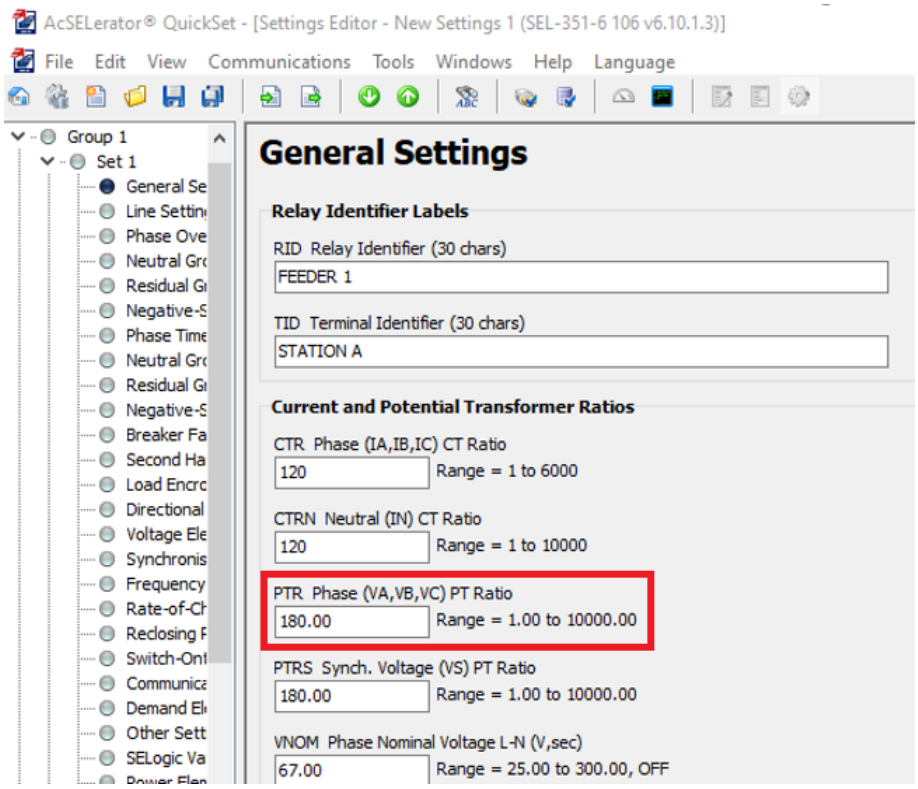

Figure 3.4: PTR Phase (VA, VB, VC) PT Ratio Setting in SEL-351 Protection Relay

4.1 Setup for New PowerWorld Testcase

The LabVIEW HIL Program can be easily re-purposed to work with almost any other C37.118 stream. This is done by changing some of the variables within the C37.118 Operation Mode Front Panel to correspond to the new C37.118 stream. The first variable that needs to be set is the PowerWorld IP address; this needs to be changed to the IP address of the new PowerWorld simulation. The second variable that needs to be set is the PDC ID for the new PowerWorld simulation. The third variable is the simulation Time Base. The standard for this is typically 1,000,000 but can be set in the simulation to different values. The next variable is the PowerWorld port number which is normally 4712 but also can be adjusted in the simulation. The last two variables are the configuration packet size and data packet size. These must be set to match the size of the configuration and data packets exact size in order for the program to properly process and dissect this C37.118 stream.

One way to do this is to use an open-source tool called PMU Connection Tester [17]. This application allows for testing and troubleshooting C37.118 connections. The configuration packet size can be found using this tool, setting the TCP connection IP and port addresses, setting the protocol, device ID, and selecting connect. If these parameters are set correctly the C37.118 stream will be displayed and begin being parsed. Figure 4.1 shows the location within PMU connection tester to find the value for the configuration packets frame length. Figure 4.2 shows the location within PMU connection tester to find the value for the data packets frame length. In these figures, the configuration frame length for this stream is 174 and the data frame length is 44. These are the values that would be entered into configuration and data packet size variables in the LabVIEW HIL program.

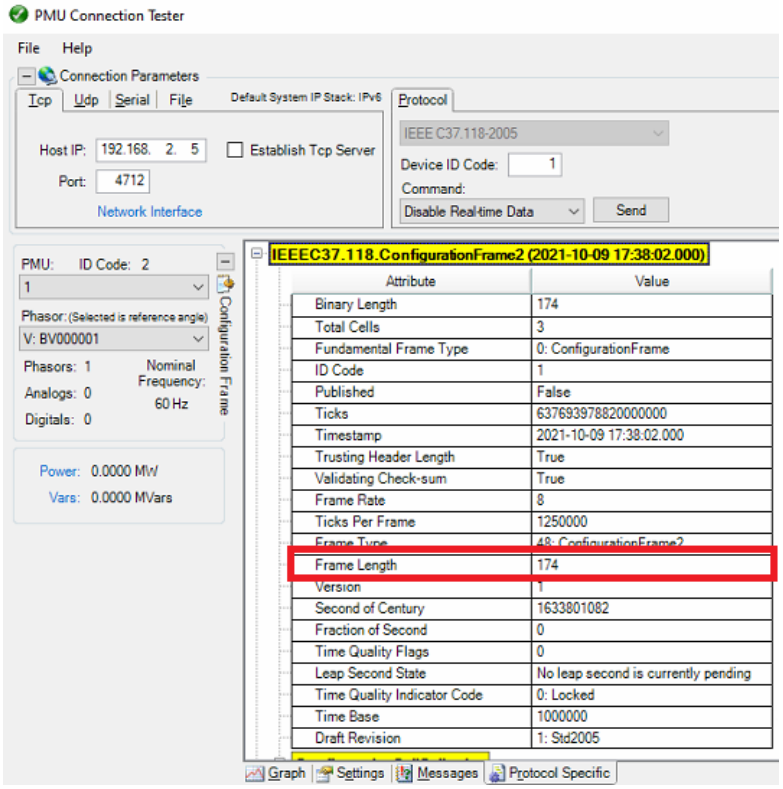

Figure 4.1: PMU Connection Tester Configuration Frame Length

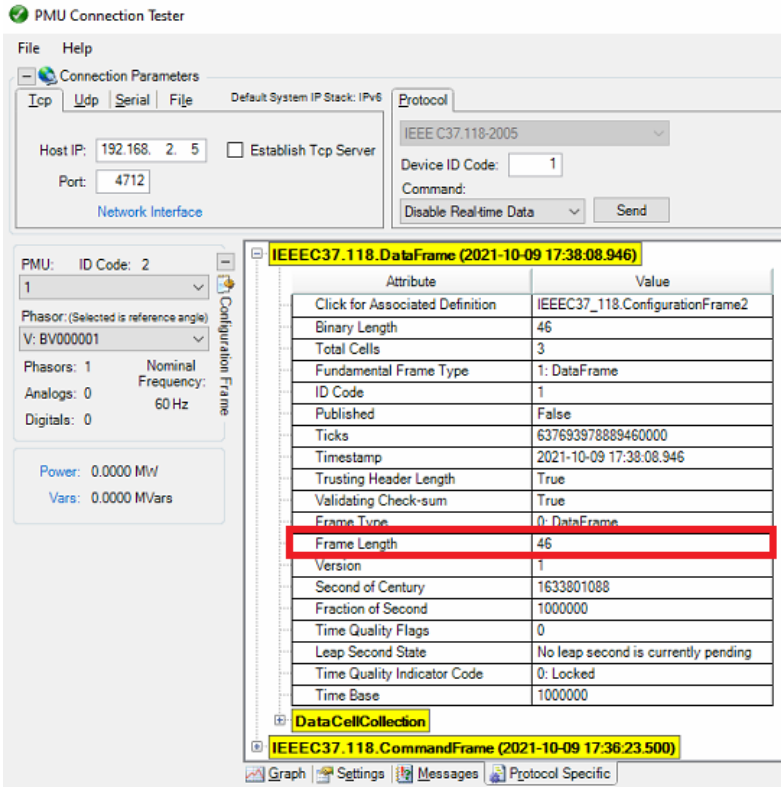

Figure 4.2: PMU Connection Tester Data Frame Length

4.2 Relay Fuzzing

The relay fuzzing was conducted using the the LabVIEW HIL program Manual Operation mode. Using this mode, multiple tests can be conducted on the SEL-351 that test all the different variables associated with the AC sine waves being sent. These include magnitudes, phases, and frequency for each sine wave for both voltage and current. The results from these test are shown in tables 4.2 to 4.8. Any parameters that are unlisted were set to the default values listed in table 4.1. Voltages are stated in Volts. Currents are stated in Amps. Phases are in degrees, and frequencies are in Hertz.

Table 4.1: Default Fuzzing Values

	Standard Values (STD)
Voltage Magnitudes	50000
Current Magnitudes	100
Voltage Phase	0, 120, 240
Current Phase	0, 120, 240
Voltage Frequency	60
Current Frequency	60

Table 4.2 shows the testing parameters set for conducting the relay fuzzing test on voltage for the SEL-351. This test found that the maximum voltage reading on the SEL-351 only reached 240kV which was less that the 250kV that was desired for testing. The reason for this is this is a limitation of the maximum voltage that can be outputted by the CompactRIO. Since the CompactRIO could not drive the voltage on the SEL-351 any higher than 240kV, the test was unable to cause the relay to trip. This showed that the default conditions for the relay to trip within 60 seconds is greater than 240kV and therefore cannot be tripped under these conditions. Durations and Times Tripped are in seconds.

Table 4.2: Voltage Magnitude Fuzzing Test

Test Number	Voltage A	Voltage B	Voltage C	Duration	Trip Occurred?	Time Tripped
1	25000	25000	25000	60	NO	N/A

Table 4.3 shows the testing parameters that were set for conducting the relay fuzzing test on current for the SEL-351. This test found that when the current is set for 1000A that it takes 20 seconds before the relay will trip, shown by test number three. From there the current amperage

was increased in steps of 100A until the time to trip was instantaneous, which was found to be around 1800A. For all but test four, current B and C were left at zero. This was done to prevent potentially overloading the relay. However, by comparing test three and four, where test three has only current a set to 1000A and test four has all three set to 1000A, both tests took the same amount of time before the trip occurred. From there it was concluded that the relay trip time is not affected by the number of currents active but by the highest value. Figure 4.3 shows a plot of time to trip vs. current.

Table 4.3: Current Magnitude Fuzzing Test

Test Number	Current A	Current B	Current C	Duration	Trip Occurred?	Time Tripped
1	1000	0	0	5	NO	N/A
2	1000	0	0	10	NO	N/A
3	1000	0	0	20	YES	12.5
4	1000	1000	1000	20	YES	12.5
5	1100	0	0	20	YES	8.5
6	1200	0	0	10	YES	6.5
7	1300	0	0	10	YES	5.5
8	1400	0	0	10	YES	4.5
9	1500	0	0	5	YES	4
10	1600	0	0	5	YES	3.5
11	1700	0	0	5	YES	3
12	1800	0	0	5	YES	immediate

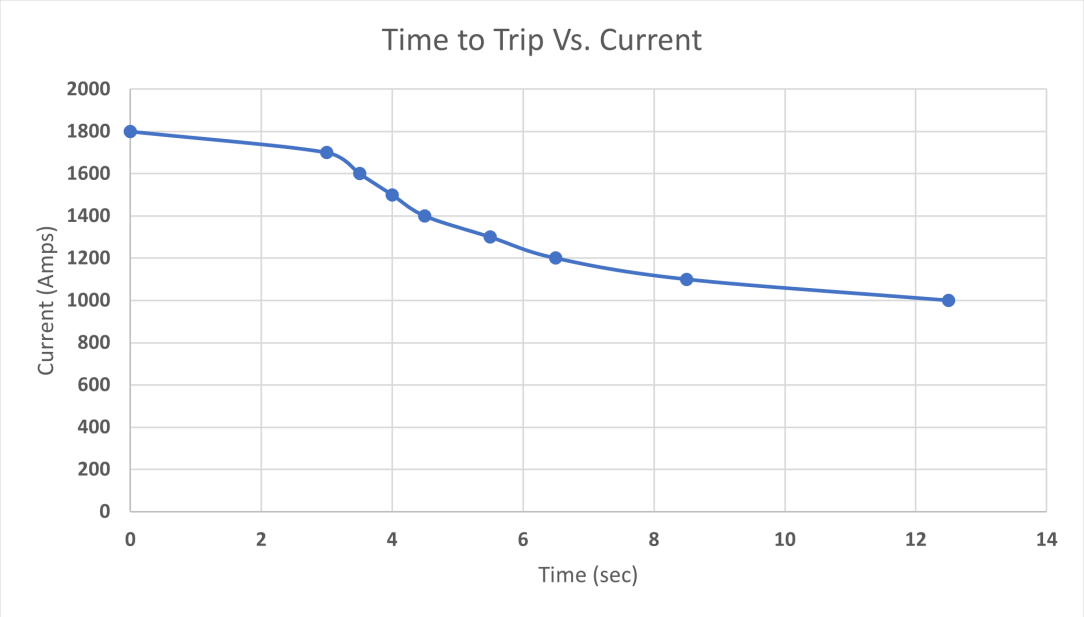

Figure 4.3: Time to Trip Vs. Current Graph

Tables 4.4 and 4.5 shows the results of the relay phase fuzzing test. This test involved adjusting the phase of each sine wave to observe any changes and see if this would cause the relay to trip or throw an alarm using the default configuration. The results showed that no significant changes in the phase of any of the sine waves cause the relay to trip or throw an alarm. What was observed was that the relay uses the phase of voltage A as the baseline for all the voltage and current phases. For example, when voltage A phase was set to 90 degrees and all other sine waves were set to normal values, the relay would read voltage A phase as 0 degrees and all other phases would be their normal value minus 90 degrees, such as voltage B phase would be 30 instead of 120. However, if the phase of any of the other sine waves besides voltage A phase were changed to something other than their normal value, all other phases would still read correctly and the one changes would read the value it was changed to. Examples of the results from these test are shown in table 4.6. The numbers in the Test Results column correspond to a test number in either table 4.4 or table 4.5 based on "Volt" or "Cur". The phase values are the reading that were detected for each phase of either voltage or current based on which test it corresponds to.

Table 4.4: Voltage Phase Fuzzing Test

Test Number	Voltage A Phase	Voltage B Phase	Voltage C Phase	Duration	Trip Occurred?	Time Tripped
1	90	120	240	10	NO	N/A
2	120	120	240	10	NO	N/A
3	240	120	240	10	NO	N/A
4	0	0	240	10	NO	N/A
5	0	90	240	10	NO	N/A
6	0	240	240	10	NO	N/A
7	0	120	0	10	NO	N/A
8	0	120	90	10	NO	N/A
9	0	120	120	10	NO	N/A

Table 4.5: Current Phase Fuzzing Test

Test Number	Current A Phase	Current B Phase	Current C Phase	Duration	Trip Occurred?	Time Tripped
1	90	120	240	10	NO	N/A
2	120	120	240	10	NO	N/A
3	240	120	240	10	NO	N/A
4	0	0	240	10	NO	N/A
5	0	90	240	10	NO	N/A
6	0	240	240	10	NO	N/A
7	0	120	0	10	NO	N/A
8	0	120	90	10	NO	N/A
9	0	120	120	10	NO	N/A

Table 4.6: Example Relay Readings from Relay Phase Fuzzing

Test Results	Phase A	Phase B	Phase C
Volt 1	0	30	150
Volt 2	0	0	120
Volt 5	0	90	240
Volt 8	0	120	90
Cur 1	90	120	240
Cur 5	0	90	240
Cur 8	0	120	90

Tables 4.7 and 4.8 show the results from the relay frequency fuzzing tests. Frequencies were tested between a maximum of 70Hz and a minimum of 50Hz because these were the maximum and minimum frequencies that the relay was capable of detecting and so no test were conducted outside this range as a result. These tests did not cause the relay to trip or throw an alarm. However, similar to the Phase Fuzzing tests, it was noticed how the frequency of voltage A was used as the baseline frequency that all the other phases were based off of. For example, when the frequency of just voltage A was changed, the relay would assume that the frequency for all the other phases should be equal to the frequency of voltage A, but since they are not, the reading for all other phases would become sporadic and the magnitudes would read inaccurately. When the frequency of VA was changed from the nominal of 60Hz to 59Hz, it would cause all other phase readings to become sporadic but the magnitudes would remain relatively accurate. When the frequency of VA was changed from the nominal of 60Hz to 50Hz, it would cause all other phase readings to become sporadic and the magnitudes of all but voltage A to drop to a fifth of what they were supposed to be. For example, voltage B would drop from 50,000V to 10,000V. When the frequency of VA was changed from the nominal of 60Hz to 70Hz, it would cause all other phase readings to become sporadic and the magnitudes to become half of what they were supposed to be; so voltage B would

read 25,000V instead of 50,000V. When any other phase besides voltage A was changed, only the phase and magnitude reading for that phase would be affected. Whenever one of the other phases were changed, their phase reading would become sporadic and their magnitude would become a fifth of what they were supposed to be. If the magnitude was changed to a value below 50Hz or above 70Hz, the relay would no longer detect that phase and show it as down.

Table 4.7: Voltage Frequency Fuzzing Test

Test Number	Current A Phase	Current B Phase	Current C Phase	Duration	Trip Occurred?	Time Tripped
1	59	60	60	10	NO	N/A
2	50	60	60	10	NO	N/A
3	70	60	60	10	NO	N/A
4	60	59	60	10	NO	N/A
5	60	50	60	10	NO	N/A
6	60	70	60	10	NO	N/A
7	60	60	59	10	NO	N/A
8	60	60	50	10	NO	N/A
9	60	60	70	10	NO	N/A

Table 4.8: Current Frequency Fuzzing Test

Test Number	Current A Phase	Current B Phase	Current C Phase	Duration	Trip Occurred?	Time Tripped
1	59	60	60	10	NO	N/A
2	50	60	60	10	NO	N/A
3	70	60	60	10	NO	N/A
4	60	59	60	10	NO	N/A
5	60	50	60	10	NO	N/A
6	60	70	60	10	NO	N/A
7	60	60	59	10	NO	N/A
8	60	60	50	10	NO	N/A
9	60	60	70	10	NO	N/A

5. MiTM Attack on Synchrophasor Protocol and ML Detection

Phasor measurement units (PMUs) use the IEEE C37.118 protocol to send telemetry to phasor data collectors (PDC) and human machine interface (HMI) workstations in a utility control center. However, the C37.118 protocol utilizes the internet protocol (IP) stack without any authentication mechanism. This means that the protocol is vulnerable to false data injection (FDI) and false command injection (FCI). In this chapter, we describe different scenarios in which C37.118 protocol's integrity and confidentiality can be compromised by a malicious agent. We also show how machine learning (ML) algorithms can be used to detect such attacks.

5.1 IEEE C37.118 Packet Dissection

In order to conduct a man-in-the-middle (MiTM) attack on the IEEE C37.118 protocol, there had to be a way to properly dissect, interpret and modify the packets in real time. The problem was that no open-source code existed that allowed for simple dissection and creation of C37.118 packets using a common program language, such as Python. This would require the development of a custom C37.118 packet dissector. To simplify the development of this C37.118 packet dissector, a packet manipulation program, known as Scapy [18], was utilized.

The Scapy packet manipulator is a Python package that is used to build or dissect different types of network packet protocols. Some of the native protocols that it supports are Bluetooth, HTTP, Netflow, SCTP, and TCP. Scapy also allows custom packet dissectors to be written for any protocol that is based on the Open System Interconnection (OSI) model [19]. For instance, the authors in [20] developed new Scapy libraries for another industrial control protocol called Distribute Network Protocol 3 (DNP3). In the case of this project, the IEEE C37.118 network protocol was not a native protocol to Scapy which meant that this protocol would have to be manually built and added to the Scapy protocols list.

There are three main complications with the IEEE C37.118 protocol that require extra programming compared to simpler protocols that have static packet sizes, fields, and types. The first

complication is that there are three types of IEEE C37.118 packets: Command, Configuration, and Data. Since the format for each type of packet is different, Scapy must be able to interpret which type of C37.118 packet it has received in order to properly dissect it. The second complication is that the number of items in the configuration and data packets can vary greatly based on the number of phasors. This requires Scapy to be able to determine both the number of stations and phasors per station associated with a specific C37.118 stream. The third complication is that C37.118 sends all the phasor values in the data packets appended one after the other. The only way to interpret what values correspond to which phasors is to use the configuration packet associated with that C37.118 stream. In order for Scapy to be able to properly parse the data packets, it must first parse the configuration packet and store the information from the configuration packet so that it can be referenced later for dissecting the data packets.

5.2 Testbed Architecture

The testbed architecture built to create the C37.118 cyber attack consisted of three virtual machines: PowerWorld VM, Client VM, and the Common Open Research Emulator (CORE) VM. Figure 5.1 shows the layout of how a synchrophasor data is transmitted between a PMU and a control center. It also shows the ideal place that an attacker would want to target in order to have the largest impact, which is right before the synchrophasor packets are sent to the control center because this is the point where the C37.118 packets contain the most PMU measurements.

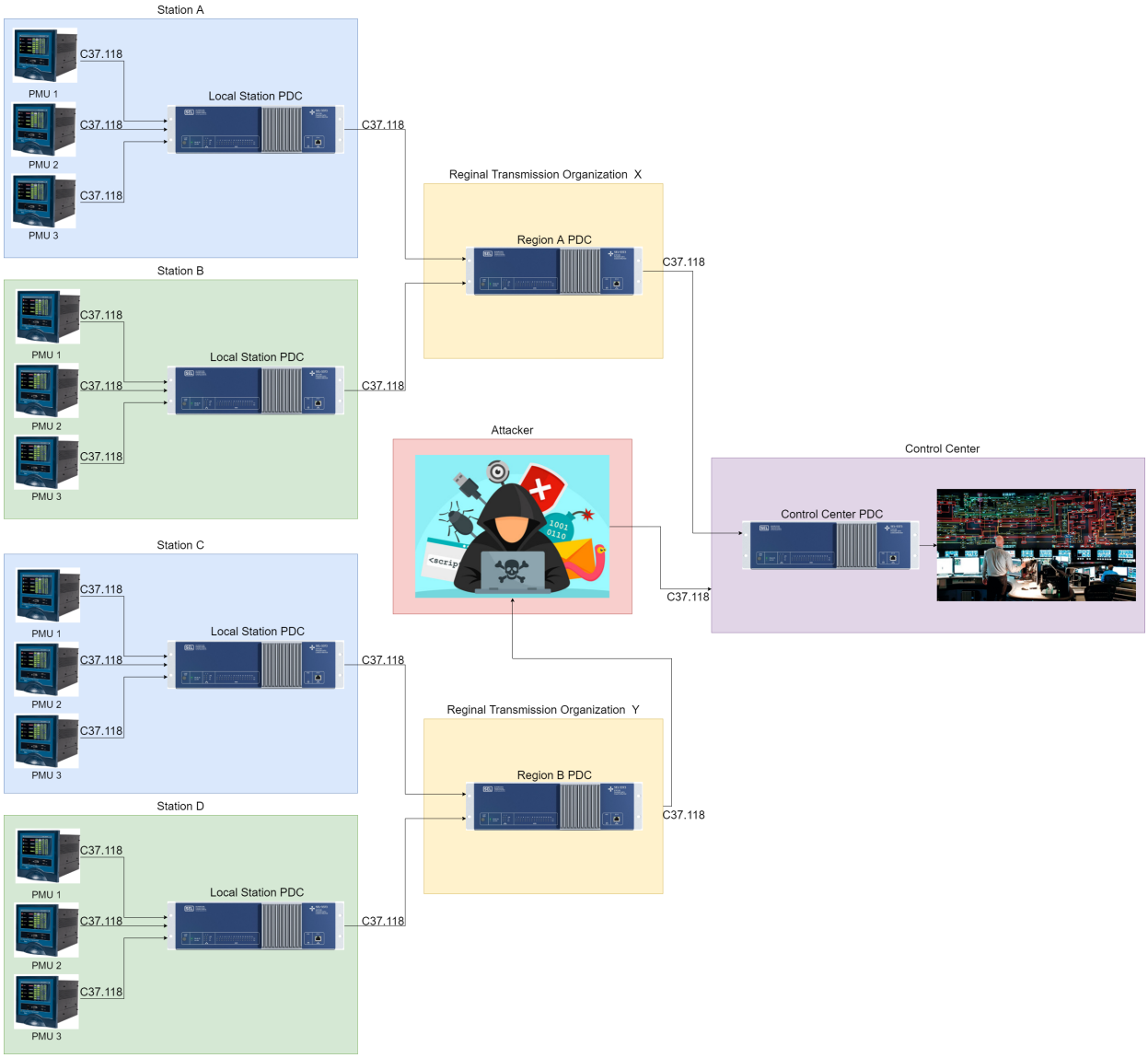

Figure 5.1: Example Layout of Synchrophasor Data Transmission from a PMU to a Control Center

The PowerWorld VM is the server that simulates a C37.118 power grid network. The Client VM acts as a regional transmission organization that is receiving the C37.118 data from the PowerWorld VM. The CORE VM creates an emulated network that is used to connect the PowerWorld VM and Client VM while also providing a realistic attack vector on the C37.118 packet stream. Table 5.1 shows the different resources that were assigned to each of the VMs that allowed them to operate ideally. The virtual environment was hosted using an open-source program called Oracle

VirtualBox [21]. This program allows for the creation and operation of multiple virtual machines, including all their virtual hardware, simultaneous as well as the creation of virtual networks that can be used by the virtual machines to communicate with each other independently. Figure 5.2 shows the layout of the virtual machines used in the C37.118 attack testbed.

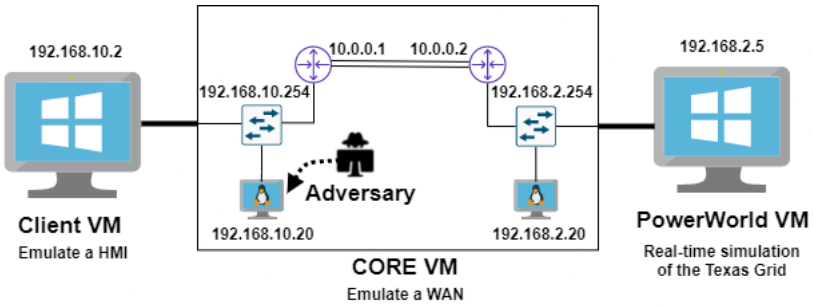

Figure 5.2: C37.118 Attack Testbed Diagram

In this testbed, two virtual networks were created. The first was between the PowerWorld VM and the CORE VM. The second was between the CORE VM and the Client VM. The PowerWorld VM and CORE VM each only had one network interface but the CORE VM had two network interfaces so that it could connect to both the PowerWorld VM and the CORE VM simultaneously.

Table 5.1: Resources Available to Each VM in the Testbed

VM Name	Operating System	Software	Internal Network	# of Cores / RAM
PowerWorld VM	Windows 10	PowerWorld	PowerWorld	4 Cores / 8GB RAM
CORE VM	Ubuntu 18.04	CORE	PowerWorld / Control	2 Cores / 6GB RAM
Client VM	Windows 10	PMU Connection Tester	Control	2 Cores / 6GB RAM

5.2.1 PowerWorld VM

The PowerWorld VM is a Windows 10 machine that is used primarily to run PowerWorld Simulator and PowerWorld Dynamic Studio (PWDS). PowerWorld Simulator is used to design and create new power grid simulations with different numbers of PMUs and types of events. These simulations are then simulated using PowerWorld Dynamic Studio. PowerWorld Dynamic Studio also allows for different types of power grid communication protocol and for other programs to connect to it as if it were a real power grid. Specifically, in the C37.118 Attack, PWDS functions as a small part of a power grid network that transmits its PMU measurements to a larger regional organization that consolidates the PMU measurements for multiple other parts of a larger power grid network. PWDS provides a realistic network packet stream that is representative of what would be seen in an actual power grid network and it is for that reason that it was selected as the server for the C37.118 Attack. Figure 2.3 shows the small three-bus power grid network that was used for generating the C37.118 packet stream utilized in this attack.

5.2.2 Client VM

The Client VM is a Windows 10 machine that used to connect to PWDS, initialize the C37.118 packet stream, and parse the packets. This was done by utilizing a program called PMU Connection Tester [17]. PMU Connection Tester is an open-source program that is capable of receiving and parsing synchrophasor packet streams. This allows the Client to operate as a power grid regional organization that consolidates the PMU measurements from multiple smaller power grid phasor data concentrators. In this test, only one C37.118 packet stream was necessary; however, multiple instances of PMU Connection Tester could be used to connect to multiple different phasor data concentrators. PMU Connection Tester running in the client VM is shown in Figure 5.3.

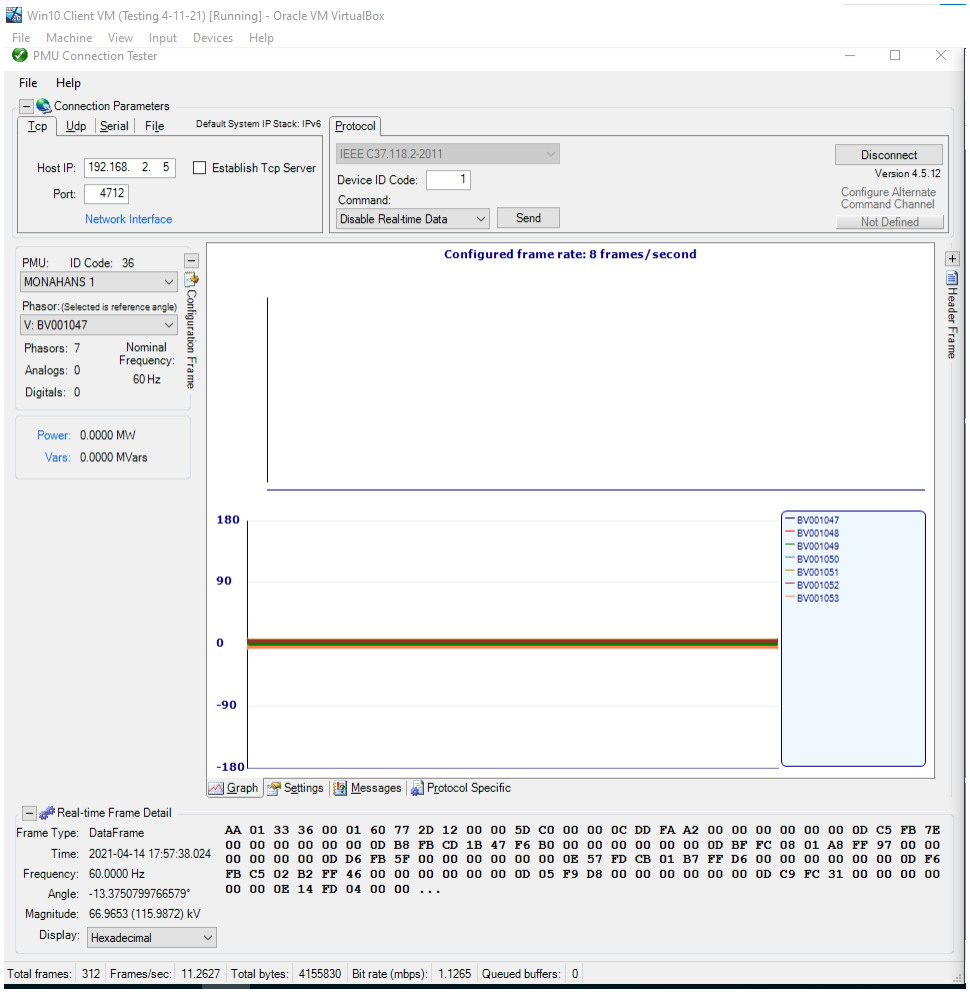

Figure 5.3: PMU Connection Tester on the Client VM

The Client VM was also the location where the C37.118 data was collected for processing by the machine learning (ML) classifiers. This was done by using another open-source program called Wireshark [22]. Wireshark is a program used for monitoring and capturing network packets received and sent over a local network interface on the host machine. In this case, Wireshark was capturing all the network traffic on the interface connected between the Client VM and the CORE VM.

5.2.3 CORE VM

The CORE VM is an Ubuntu machine that was used to emulate the network between the PowerWorld VM and the Client VM. This was done using a program Common Open Research Emulator (CORE) [23]. CORE is a tool developed by the U.S. Navy Research Laboratory which is used to create virtual emulated networks. The benefits of using virtual emulated networks instead of simulated networks is that emulated networks better represent how network traffic flows on a physical network. For example, an emulated network is capable of accounting for the link delay that occurs over transmission lines and between other devices, whereas a simulated network would not include these real-time delays. These delays are important to this attack because without them there is no time for the attack to occur between C37.118 packets.

Figure 5.4 shows the CORE network built for this attack. At the top there are two routers. These routers are used to create separate subnets for the the Client network and the PowerWorld network, since the C37.118 client and server would typically not be on the same subnet. Below the two routers are two switches. These two switches are used to attach two devices to the PowerWorld subnet and the Client subnet. At the bottom left and bottom right corners are two interface connects. These items are used to connect to the two virtual network interfaces on the CORE VM that are connected to the PowerWorld VM and the Client VM. Whenever CORE is started, it takes ownership of the two CORE VM network interfaces and forwards all the network traffic over those interfaces to each of the CORE subnets. Enp0s3 is the interface connected to the Client VM and enp0s8 is the interface connected to the PowerWorld VM. The two machines (N5 and N6) located at the bottom center are two emulated host machines that allow for direct interaction with the two different subnets from with the CORE emulation. N5 is the host that runs the C37.118 attack, i.e., the Adversary host.

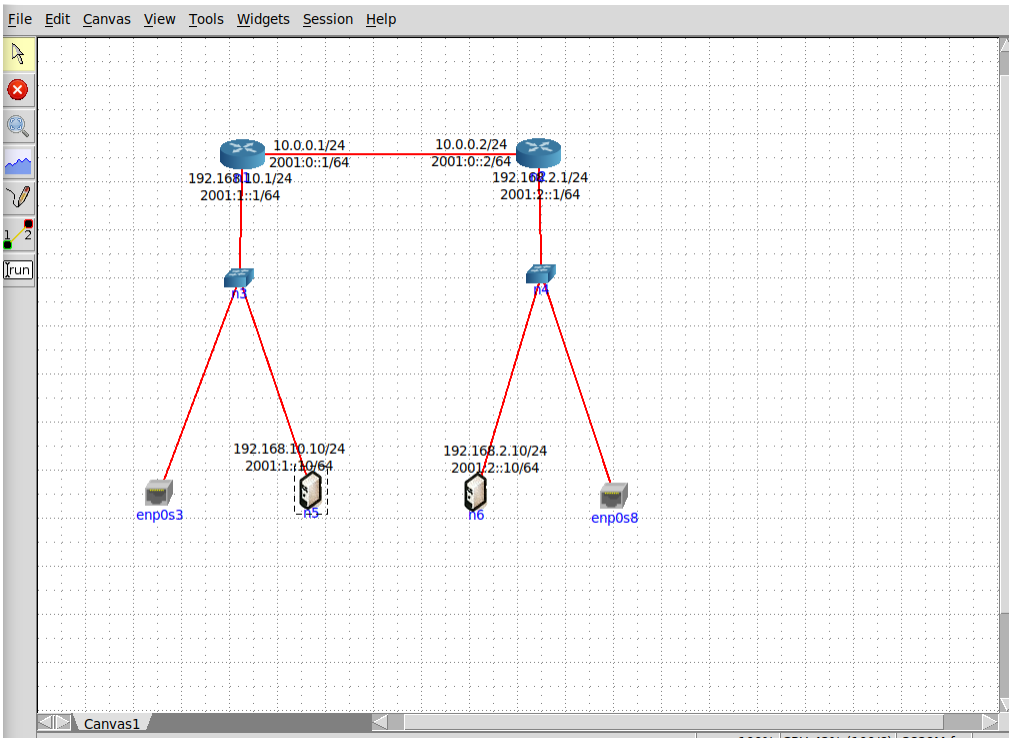

Figure 5.4: C37.118 Attack CORE Network

5.3 False Data and Command Injection

Using the custom Scapy C37.118 protocol, the route between the Client VM and the PowerWorld VM was compromised by the Adversary host shown in Figure 5.2. From this the Adversary host is able to eavesdrop on all the traffic between the Client VM and PowerWorld VM. This means that the attacker is able to learn all information about the different PMUs in the compromised C37.118 packet stream, such as all the station names, number of PMUs, and the measurements read from all the PMUs. This also sets the stage for the next part of the attack, false data and command injection.

There were four different types of false data and command injection attacks created for this project:

- Phase and Voltage Modification,
- Timestamp Modification,

- CRC Modification,
- Transmission Off.

Phase and Voltage Modification, Timestamp Modification, and CRC Modification are all false data injection attacks and Transmission Off is a false command injection attack. These attacks were chosen because each of them affects a different key part of the C37.118 protocol. Phase and Voltage Modification is used to target the phase and/or voltage of a specific PMU and adjust the measurement to a different value than is originally transmitted. In this case the values are set to zero to make the PMU appear as if the transmission line it is connected to has gone offline. Timestamp Modification involved removing the timestamp of each packet and replacing it with a timestamp that is plus or minus 30 seconds from what the actual timestamp was originally. This was done to make the C37.118 packets appear as if they were being received out of order. CRC Modification involved taking the CRC of each packet received, removing it, and replacing it with an incorrect value. This was done to make all the C37.118 packets received appear as if they were corrupted. Transmission Off would intercept all C37.118 command packets, remove the desired command, and replace it with the command for transmission off. This would make PowerWorld receive only commands to turn off the transmission to the client and never receive a Transmission On command, preventing the client from ever being able to start the C37.118 data stream.

5.4 Data Set Collection and Generation

To gather the C37.118 data packets, Wireshark [22] runs on the Client VM to capture packet capture (PCAP) data. The Client VM was used because it offers the point of view of the operator. After each test, the VMs were reset to ensure that no data was carried over from one test to the next. The rate of transmission for all test was set to eight data frames per second and each test was ran for 20 minutes per experiment.

In order to get the PCAP data into a format that could be used by the ML classifiers, a two step conversion process was done. The first step was exporting the PCAP Wireshark packets to a PDML file. From there a custom Python script is run that extracts only the desired features from

the PDML file. This data is saved as a CSV. The features that are saved are: Frame Rate per Second (FRS), Timestamp (SOC), Checksum, Voltage, and Angle.

During a Wireshark packet capture, Wireshark defaults to capturing all network traffic detected on the specified interface. This requires all Wireshark to then be set to filter out all traffic except C37.118 packets. This is done so that there is less data being saved to the PDML file since these files are already very large with just C37.118 packets and slow to parse.

In order for the ML classifiers to process data, it has to be in an acceptable format that can be easily imported. One of those formats is as a CSV file. Wireshark does contain an export to CSV function, but this does not export all the necessary C37.118 data that is required. The only export function built into Wireshark that exported all the required fields was PDML. For this reason, a custom conversion program was built to extract the required files from a PDML format and save it as a CSV.

The reason for converting to CSV instead of directly parsing it using the ML classifiers is because, even though it has all the data for the required features, these files are also very large and mostly filed with unused data. Table 5.2 shows the different files sizes for each of the data types of the same packet capture of the small three-bus case.

Table 5.2: Different File Sizes for a Single 20 Minute Small Test Case

Unfiltered PCAP	2,112 KB
Filtered PCAP	1,238 KB
PDML	251,699 KB
CSV	719 KB

5.5 Machine Learning Classifiers

Machine Learning (ML) Classifiers are useful tools that are capable of detecting different types of attacks. The goal of this project was to determine and compare the precision, recall, and F-1 score of three different ML classifier: k-Nearest Neighbor, Decision Tree, and Naive Bayes. This was done by utilizing an online resource called Scikit-learn [24]. Each of these ML classifiers was also configured to generate multi-labeled outputs so that they could identify packets that have more than one modified field. For example, if both voltage and angle were modified in one packet, multi-labeling allows the ML classifier to classify the attack as both a voltage attack as well as an angle attack instead of only being able to chose one.

Each classifier was trained on the same training data set and the same test split. The normal and each of the attack datasets were first combined into one large compiled dataset. This data in this dataset is then randomized so that when the dataset is split into 80% training and 20% testing there will be an even distribution of all the different types of data throughout. The dataset is then normalized to prevent any over-fitting.

Once the final dataset is created, each of the ML classifiers is trained using the 80% training data set of 50,482 packets. After the classifiers are trained, they then are tested using the 20% testing data set of 12,621 in order to determine each classifier's ability to predict each of the different types of attacks. Lastly, the predicted labels are then compared against the real labels for the test data to determine the categories each data point falls under as well as the efficiency of each algorithm. The different evaluation metrics used for this application are true positives, false positives, true negatives, false negatives, precision, recall, and F-1 score for each ML classifier.

There are four main categories that a classification will fall under: true positives, false positives, true negatives, and false negatives. A true positive is when an attack data point is correctly labelled as an attack data point. A false positive is when a normal data point is incorrectly labelled as an attack data point. A true negative is when a normal data point is correctly labelled as a normal data point. A false negative is when an attack data point is incorrectly labelled as being a normal data point.

The results from these four categories are then used to determine the precision and recall of each classifier in order to understand the efficiency of each ML algorithm. Precision is the measurement of the algorithms ability to correctly classify true and false positives. The equation for precision is shown in figure 5.5. Recall is the measurement of each of the algorithms ability to correctly classify true positives versus false negatives. The equation for recall is shown in figure 5.6. For this application, it was determined that a higher recall rate is more desirable that a higher precision rate because it is more important to detect true positives even though this may generate more false positives as well. The F-1 score is used to find the best balance between precision and recall, where a value as close to one is the most desirable. The equation for finding the F1-score is shown in figure 5.7.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Figure 5.5: Equation for Calculating Precision

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Figure 5.6: Equation for Calculating Recall

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Figure 5.7: Equation for Calculating F1-Score

While there are many other classifier algorithms that exist that could have been used to label the C37.118 attack dataset, these three were chosen because each algorithm is capable of producing

multi-labelled outputs. The other reason these three algorithms were chosen is because each of them operates in a different and unique way so that a wider range of ML classifiers abilities are tested on detecting attacks on C37.118.

5.5.1 K-Nearest Neighbor

The algorithm used by k-Nearest Neighbor (kNN) uses the distance between labeled training data and unlabeled new data to determine how to label the new data [25]. There are several different formulas that can be used for kNN, such as Euclidean, Manhattan, and Minikowski; however, it was determined that the Euclidean distance formula, which is the most commonly used, was the preferred formula for this experiment.

The $\boldsymbol{k}$ in k-Nearest Neighbor is a variable that must be tested to find the optimal value for each application of kNN, since the value changes for each new application. To find the optimal value for $\boldsymbol{k}$, multiple values for $\boldsymbol{k}$ had to be tested through trial and error to determine the optimal value that minimized the mean absolute error function. Using the Elbow method [26], $\boldsymbol{k}$ from 1 to 100 was tested, shown in Figure 5.8, and it was determined that seven was the optimal value for $\boldsymbol{k}$ for this application. These means that the seven data points closest to the new unclassified data point will be used to determine the classification for the new data point.

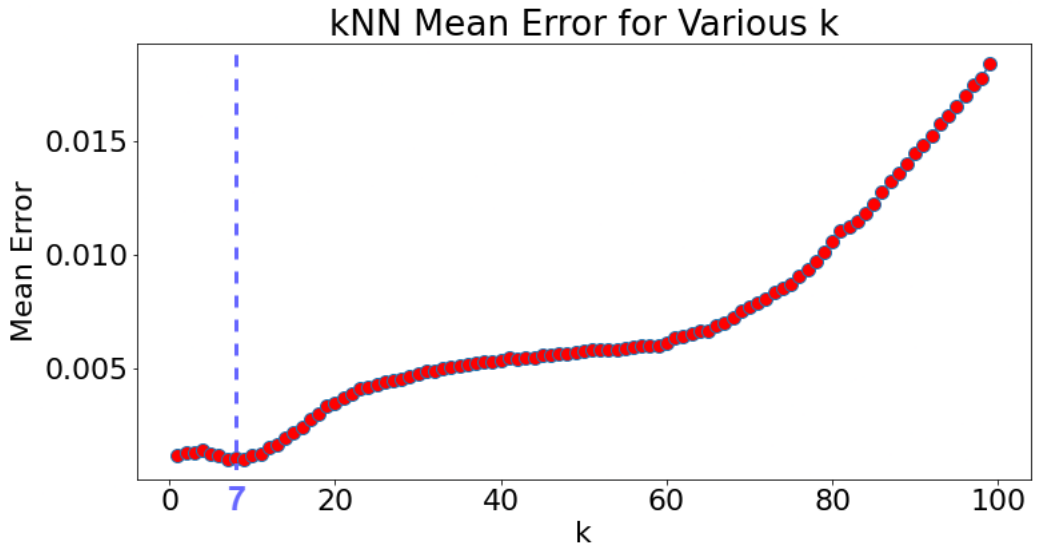

Figure 5.8: Searching for the Optimal Value for ***k*** Ranging from 1 to 100

5.5.2 Decision Tree

The Decision Tree (DT) classifier functions uses a process called binary recursive partitioning in order to build a decision tree capable of labeling data based on the different value features [27]. These DT have a flowchart-like structure where each block represents a test and each branch from that block represents a possible outcome. These decisions continue to branch off until every possible outcome has been connected to a single branch. The DT algorithm uses a function known as the impurity measurement, which determines when a branch should fracture into two different leaf nodes. There are two different versions of the impurity measurement, Gini impurity [28] and entropy. Both versions were tested and it was discovered that for this application that the Gini impurity measurement produced a higher F-1 score and therefore was chosen for this project.

5.5.3 Naive Bayes

The Naive Bayes (NB) classifier uses the probability from previous events occurring along with information from new events to determine the future probability of the next event [29]. NB operates using the Bayes theorem. The Bayes algorithm predicts the label of new data points

assuming that the features are independent of one another. However, in the real world features are almost never independent. For this reason, it is called Naive Bayes (NB). For our application, the Naive Bayes theorem was modified to allow for multi-label outputs.

5.6 Results

Using the results of the seven evaluation metrics, each of the three ML classifiers were compared to determine which proved to be the most accurate at detecting the different types of false data and command injection attacks in C37.118 data traffic. Figure 5.10 shows the evaluation metrics results for each type of attack within each ML classifier. As can be seen in Figure 5.9, DT had the highest F1-score for angle modification, voltage modification, and normal traffic. The F1-score for DT tied with NB for CRC modification and timestamp modification. DT had a 100% recall for every dataset whereas kNN and NB only had 100% recall for a couple of datasets. DT also had the highest F-1 score for Transmission Off, which was the hardest attack to detect since it involved preventing the stream of data packets from starting and only generated a three C37.118 packets for that dataset. While each ML classifier proved to be capable of detecting most attacks on C37.118 attacks, it is clear that Decision Tree is the most capable of detecting these types of attacks with the greatest level of accuracy.

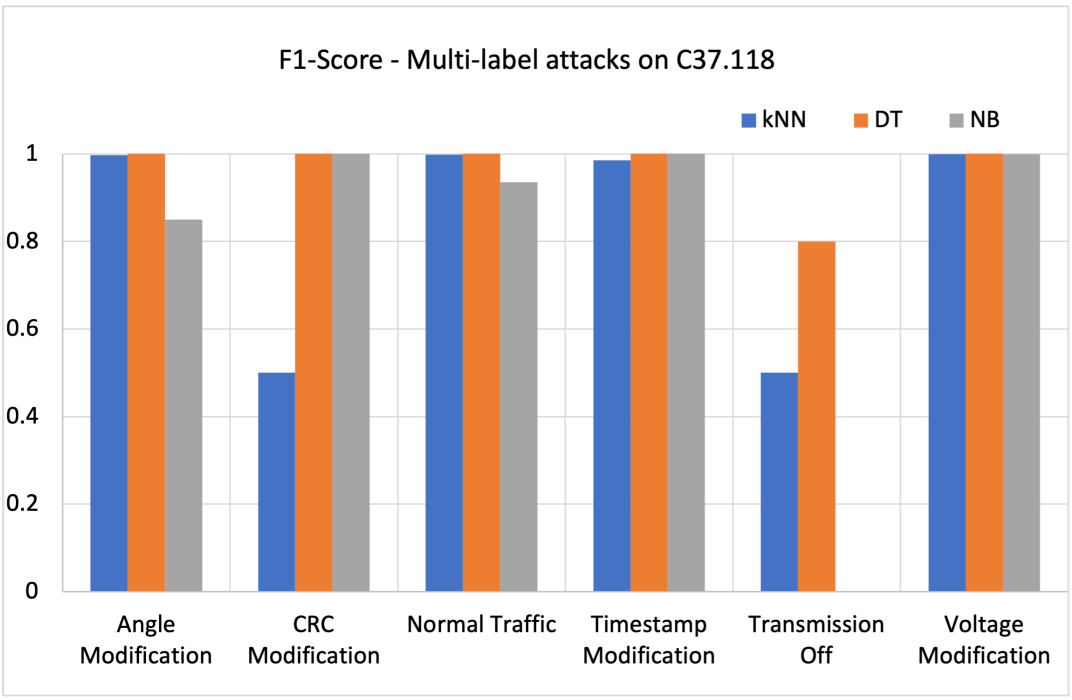

Figure 5.9: F1-score Results for kNN, DT and NB Algorithms.

	FCI or FDI type	True Positive	False Positive	True Negative	False Negative	Precision	Recall	F1-Score
kNN	Angle Modification	2,131	9	10,477	4	0.9957	0.9981	0.9969
	CRC Modification	1	0	12,618	2	1	0.3333	0.5
	Normal Traffic	8,416	24	4,176	5	0.9972	0.9994	0.9983
	Timestamp Modification	963	3	11,630	25	0.9969	0.9747	0.9856
	Transmission Off	1	2	12,618	0	0.3333	1	0.5
	Voltage Modification	2,166	1	10,454	0	0.9995	1	0.9998
DT	Angle Modification	2,135	0	10,486	0	1	1	1
	CRC Modification	3	0	12,618	0	1	1	1
	Normal Traffic	8,421	0	4,200	0	1	1	1
	Timestamp Modification	988	0	11,633	0	1	1	1
	Transmission Off	2	1	12,618	0	0.6667	1	0.8
	Voltage Modification	2,166	0	10,455	0	1	1	1
NB	Angle Modification	2,135	752	9,734	0	0.7395	1	0.8502
	CRC Modification	3	0	12,618	0	1	1	1
	Normal Traffic	8,079	780	3,420	342	0.9199	0.9594	0.9351
	Timestamp Modification	988	0	11,633	0	1	1	1
	Transmission Off	0	3	12,618	0	0	0	0
	Voltage Modification	2,166	3	10.452	0	0.9986	1	0.9993

Figure 5.10: Evaluation Metric for Each FCI and FDI Attacks Organized by Type of Classifier

6. Summary and Conclusions

6.1 Contributions

This work provides a way to conduct end-to-end testing on physical power grid network equipment as well as show vulnerabilities in one of the common network protocols used for this communication, the IEEE C37.118 protocol. As more power grid equipment becomes connected by communication networks, it becomes necessary to test the cyber resilience of all these components independently as well as together. Most cyber-physical power grid testbeds rely primarily on either simulated or emulated networks with little to no physical hardware included in their environment, and the ones that do support hardware-in-the-loop typically require expensive power grid generation equipment which limits the number of groups that can afford to fund projects to support cyber-physical power grid testbeds.

An example of this vulnerability has been shown in this work, where the IEEE C37.118 standard used for transmitting PMU data is manipulated using false data and command injection. This demonstrates that there are already vulnerabilities in current cyber-physical power grid systems that need to be addressed in order to secure these systems and protect against attacks.

6.1.1 Lessons Learned

The first lesson learned in this project was the uses and operation of IEEE C37.118 Synchrophasor protocol. This took a couple of months to fully learn and understand. What helped me the best to learn it was taking Wireshark captures of the protocol and then making my own programs that could send and receive C37.118 packets. This was building up to the C37.118 packet dissection for the LabVIEW HIL program and the development of the Scapy library for the C37.118 attack.

The second lesson learned was how to generate AC sine wave outputs using a NI CompactRIO. This required learning the different operation modes of the CRIO, Scan Mode and FPGA Mode, and how to program each. The first attempt was done using the Scan Mode of the CRIO since it was the simpler of the two options and it worked well for generating DC outputs; however,

when changed to AC outputs the refresh rate was too low to generate acceptable sine waves. This resulted in having to learn the FPGA mode as well and required rewriting the LabVIEW program from scratch since code between the two modes in not interchangeable. This again took several months of reading documentation and trial and error before finally getting the correct output.

The third lesson learned was how to implement ML algorithms. I had never worked with ML algorithms and so I had to first understand what they were, how different types operated, and then how to implement them. Luckily there is a lot of good documentation and resources that explain how to implement ML algorithms that simplified this process. It was also really helpful that I had a partner on this project with experience in ML that was able to teach me everything I needed to know.

The fourth and final major lesson learned was how to interface with the low-level interface on the SEL-351. This proved the most difficult to learn since there is very little documentation about this interface, only a single page in the SEL-351 manual. Learning the interface required reaching out to the manufacture to verify the maximum voltage levels that could be imputed to prevent damaging the hardware. There was also no conversion factors listed for this interface and those had to be tested and calculated manually.

One more lesson learned was how to work as part of a team to create the goals and define the scope of a project when the amount of work required to achieve the desired end goal is unknown. For example, on two separate occasions during this project I had to make predictions on how long it would take to complete a specific task which would require me learning a new skill from scratch. The first time was when I had to learn how to use and operate the C37.118 protocol and then create a custom program for it without any prior experience of synchrophasor protocols. The second was when I had to make a prediction on how long it would take for me to learn to program in LabVIEW FPGA to develop the required sine wave outputs in LabVIEW Scan Mode, which was still being used at the time, but could not accurately generate the sine waves. Both of these tasks I was able to complete within the time frame I originally gave and therefore was able to complete my project on time.

6.2 Future Work

Future Work that can extend this project would be to create new types of experiments in the RESLAB Testbed using the HIL. For example, now that the HIL has been added to the RESLAB testbed, the C37.118 attack also developed in this project could be applied on the synchrophasor communication between the SEL-351 and the SEL RTAC already installed in the RESLAB testbed. From there, the ML classifiers used to detect this attack could be implemented in real-time to observe their performance at detecting synchrophasor attacks. This would prove if adding ML classifier is a practical and necessary addition to the next generation intrusion detection systems in a cyber-physical power grid. This is just one example of how the HIL can be beneficial.

Other areas where this work could be built on would be to automate and consolidate the C37.118 attack, data collection, and ML classification programs. This could be done by building a custom Python script that replaces the PMU Connection Tester program and is able to actively collect the C37.118 packets instead of having to use Wireshark. Once these packets are collected, they could then be stored in a format required by the ML classifier without having to manually parse them. In doing this, the ML classifiers could potentially be implemented in real time. This new program could also be developed to communicate with the Advisory node in the CORE network so that the attacks could be controlled and automated by the new Python program. This would allow for longer test times without requiring an operator present to start the next test and manually parse the data.

The biggest benefit of the HIL is that now test ran within the RESLAB testbed is that now any future test conducted will be able to test the resilience of all the common network components commonly found in power grid systems while also controlling all inputs and outputs from the system. This means that all future test will be conducted end-to-end and are now as close as possible to how this equipment would be operating in a production environment.

REFERENCES

[1] A. Sahu, P. Wlazlo, Z. Mao, H. Huang, A. Goulart, K. Davis, and S. Zonouz, "Design and evaluation of a cyber-physical resilient power system testbed," 11 2020, http://arxiv.org/abs/2011.13552.

[2] P. Wlazlo, A. Sahu, , Z. Mao, H. Huang, A. Goulart, K. Davis, and S. Zonouz, "Man-in-the-middle attacks and defense in a power system cyber-physical testbed (accepted for publication in iet cyber-physical systems: Theory and applications)," 5 2021, https://arxiv.org/abs/2102.11455.

[3] T. Becejac, C. Eppinger, A. Ashok, U. Agrawal, and J. O'Brien, "Prime: a real-time cyber-physical systems testbed: from wide-area monitoring, protection, and control prototyping to operator training and beyond.," 1 2020, http://digital-library.theiet.org/content/journals/10.1049/iet-cps.2019.0049.

[4] J. P. J Belanger, P Venne, "The what, where and why of real-time simulation," *Planet Rt*, 2010, https://blob.opal-rt.com/medias/L00161_0436.pdf.

[5] A. Nelson, S. Chakraborty, D. Wang, P. Singh, Q. Cui, L. Yang, and S. Suryanarayanan, "Cyber-physical test platform for microgrids: Combining hardware, hardware-in-the-loop, and network-simulator-in-the-loop," in *2016 IEEE Power and Energy Society General Meeting (PESGM)*, pp. 1–5, 2016, 10.1109/PESGM.2016.7741176.

[6] "Ieee standard for synchrophasors for power systems," *IEEE Std C37.118-2005 (Revision of IEEE Std 1344-1995)*, pp. 1–65, 2006, 10.1109/IEEESTD.2006.99376.

[7] S. M. Farooq, S. Nabirasool, S. Kiran, S. Suhail Hussain, and T. S. Ustun, "Mptcp based mitigation of denial of service (dos) attack in pmu communication networks," in *2018 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES)*, pp. 1–5, 2018.

[8] S. Bhamidipati, K. J. Kim, H. Sun, and P. V. Orlik, "Wide-area gps time monitoring against spoofing using belief propagation," in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–8, 2019, 10.1109/SAHCN.2019.8824812.

[9] T. J. Overbye, "Powerworld," 2021, www.powerworld.com.

[10] T. A. E. E. Station, "Activsg2000: 2000-bus synthetic grid on footprint of texas," ACTIVSg2000: 2000-bus synthetic grid on footprint of Texas. [Online]. Available: https://electricgrids.engr.tamu.edu/electric-grid-test-cases/activsg2000/.

[11] A. B. Birchfield, T. Xu, K. M. Gegner, K. S. Shetye, and T. J. Overbye, "Grid structural characteristics as validation criteria for synthetic networks," *IEEE Transactions on Power Systems*, vol. 32, no. 4, pp. 3258–3265, 2017, 10.1109/TPWRS.2016.2616385.

[12] *USER MANUAL NI cRIO-9035*, National Instruments, https://www.ni.com/pdf/manuals/376935e_03.pdf.

[13] *"LabVIEW Getting Started with LabVIEW"*, National Instruments, https://www.ni.com/pdf/manuals/373427j.pdf.

[14] N. Instruments, "Frame apis for pdc by ni - toolkit for labview download," VIPM, 2016, 1.0.0.2, https://www.vipm.io/package/national_instruments_lib_frame_apis_for_pdc ?utm_sourcevipm_desktop.

[15] *SEL-351-5, -6, -7 Protection System Instruction Manual*, Schweitzer Engineering Laboratories, https://selinc.com/products/351/docs/.

[16] "Sel-5030 acselerator quickset software," 0.100.38, 2021, https://selinc.com/products/5030/.

[17] J. R. Carroll, "Pmu connection tester," v4.6, 2021, https://github.com/GridProtectionAlliance/PMUConnectionTester.

[18] M. M. R. R. S, R. R and S. G, "Scapy- a powerful interactive packet manipulation program," International Conference on Netowrking, Embedded and Wireless Systems (ICNEWS), Bangalore, India, CIGRE, 10.1109/ICNEWS.2018.8903954, 1-5, 2018.

[19] L. L. Perterson and B. S. Davie, *Computer Networks: A Systems Aproach*, vol. 5th Edition. Elsevier Science, 2011.

[20] N. Rodofile, K. Radke, and E. Foo, "Real-time and interactive attacks on dnp3 critical infrastructure using scapy," in *13th Australasian Information Security Conference (AISC 2015)* (I. Welch and X. Yi, eds.), vol. 161 of *CRPIT*, (Sydney, Australia), pp. 67–70, ACS, 2015, http://crpit.com/confpapers/CRPITV161Rodofile.pdf.

[21] "Virtualbox," Oracle, 6.1, 2021, www.virtualbox.org.

[22] P. Hunt, "Wireshark," 3.4.4, 2021, www.wireshark.org.

[23] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "Core: A real-time network emulator," in *MILCOM 2008 - 2008 IEEE Military Communications Conference*, pp. 1–7, 2008.

[24] "Scikit-learn machine learning in python," Scikit-learn, 2007, https://scikit-learn.org/stable/index.html.

[25] "Sklearn.neighbors.kneighborsclassifier," Scikit-learn, 2007, scikit-learn.org/stable/modules/generated/sklearn.neighbors. KNeighborsClassifier.html.

[26] C. Maklin, "Elbow method in supervised learning(optimal k value)," Medium, 8 2019, medium.com/@moussadoumbia_90919/elbow-method-in-supervised-learning-optimal-k-value-99d425f229e7.

[27] "1.10. decision trees," Scikit-learn, 2007, scikit-learn.org/stable/modules/tree.html.

[28] "Gini impurity and entropy in decision tree - ml," GeeksforGeeks, 7 2020, www.geeksforgeeks.org/gini-impurity-and-entropy-in-decision-tree-ml.

[29] "1.9. naive bayes," Scikit-learn, 2007, scikit-learn.org/stable/modules/naive_bayes.html.