Process and Machine Simulation for Testing PLC Programs

by

Colby Tarr and Mike DeLeo
Team ECE-810

A Senior Project Report Submitted to the Faculty of
Electrical and Computer Engineering
Penn State Erie, The Behrend College

Faculty Advisor: Dr. Rasouli
Co-Advisor: Dr. Walters
Co-Advisor: Prof. Panda

Sponsor Supervisor: Dr. George Walters
Sponsor: Penn State Behrend ECE, Erie, PA

November 2018

1 Problem Statement

Programmable Logic Controllers (PLCs) are widely used in industry to control processes and machines. PLCs are easily programmable and have a broad range of applications. Currently, at Penn State Behrend (PSB), PLC programming is taught in courses offered by both the ECE and ECET departments. In Burke 143, there is a PLC lab that is equipped with Allen-Bradley CompactLogix PLCs, PanelView 5300 touch-screen Operator Interface Terminals (OITs), software for PLC programming and OIT development. However, there are no actual processes or machines to control because such systems are prohibitively expensive and potentially hazardous.

The current way this issue is being dealt is with a simulation written in PLC logic that runs on the PLC. While this works, there are limitations to placing the simulation internally in the PLC. There is no interface with the simulation for the user. The sim program internally configures bits alongside the main control program in order to simulate the process. To make changes to the simulation, users must enter the PLC program and make runtime edits to the simulation program. This is not only tedious, but can be difficult as it requires specific knowledge of how the simulation runs and which internal values correspond to the desired physical state of the process being simulated.

1.1 Need Statement

There is a need for a way to simulate a process or machine that can be used to test and evaluate PLC programs written to control manufacturing processes.

1.2 Objective Statement

The objective of this project is to design an infrastructure so that a simulator that can be used with a PLC. While the current solution works, there is room for improvements. The focus is to allow the students to develop and test programs for process and machine control. This can be done by giving the user an interface with the simulation. A visualizer will be designed to provide this interface so that the user no longer needs knowledge of the sim code in order to use it.

1.3 Background and Related Work

PLCs are widely used for automation of industrial processes for manufacturing. PLCs are microprocessor-based controllers that uses programmable memory to store instructions that will produce functions like logic, sequencing, timing, counting, and arithmetic to control machines and processes. Inputs such as switches, meters, and output devices like valves and motors are being controlled by the PLC. A program is loaded into the memory of the PLC and then with the use of the inputs and outputs, carries out the defined control rules. PLCs are programmed with ladder logic. PLCs are similar to computers by both having specific functionalities that make them extremely useful. Computers are optimized for calculation and display task, PLCs are optimized for control task and the industrial environment [1].

Operator Interface Terminal (OIT) are industrially hardened user interfaces that allow communication with the PLC. They often display informative graphics and send operator inputs from a touch screen. Technically, an OIT is a Human Machine Interface (HMI), but OIT is used to distinguish a self-contained terminal from a more powerful computer based HMI [2].

HMI refers to a computer running a software package that is integrated with the Supervisor Control and Data Acquisition system (SCADA). An HMI such as this displays complex graphic for user interface and runs supporting programs. Another term for an HMI computer is a view node that is used to distinguish them from a SCADA [2].

The SCADA system is a system of software and hardware elements that allows industrial organizations to Control industrial processes locally or at remote locations. A SCADA server is connected to all the relevant PLC's and serves as a way of passing data between the PLC's and HMI view nodes [2].

There are several options for use of Single Board computers for this project. One of the most important requirements is timing. If a simulation is put on a microcontroller it must be fast enough to interact with the I/O of the PLC in less than 1 millisecond at all times.

The first option to use is the BeagleBone Blue. The BeagleBone Blue (BBB) has a Linux OS, and can easily connect to a PC via Ethernet for TCP connections. The BBB can transmit data via a Wi-Fi connection. On board it has 512MB of DDR3 RAM, and 4GB of main memory. DDR3 RAM is Double Rate 3 SDRAM. The processor is a 1GHz ARM Cortex-A8. With an operating system on board, some of the processor's time will be spent on OS maintenance [3].

The second option is the Raspberry pi B+. The Pi B+ requires a Linux operating system, but can be substituted for a lower maintenance Linux OS. The throughput of the Ethernet port is 1Gbps, and the throughput of the USB 2.0 ports is 300Mbps. The Pi has an HDMI that allows a monitor to be connected. Also, on board it has a 1.4GHz ARM Cortex-A53 Processor. The Pi has 1GB of SDRAM (Synchronous Dynamic Random-Access Memory), and SDRAM's limitation is that it can only read or write in one clock cycle, not both. In addition, SDRAM uses twice the voltage required and is considerably slower than DDR3 RAM [4].

2 Requirements Specification

In the following sections, we dive into the details of what the product of this project should be. Specifically, the marketing and engineering requirements.

2.1 Marketing Requirements

The marketing requirements for this project are as follows. The device should be:

1. Beneficial in an educational setting.
2. Easy to use for students and instructors.
3. Safe for the user/PLC.
4. Durable
5. Reasonably precise and have reasonable fidelity.
6. Inexpensive.
7. Easy to reproduce.
8. Easy to expand, and modify.

The marketing requirements are grouped into a hierarchy, and the relative importance of each requirement is given in the following List 1.

Main Objective: To build a PLC Simulator

- Make A PLC simulator that is Educational [0.33]
  - [0.4] Beneficial – Should be beneficial in an educational setting
  - [0.5] Easy – Should be easy for students and instructors to use
  - [0.1] Enjoyable - It should be interesting for students to work with
- Utilities [0.33]
  - [0.5] Safe - If given egregious user input, should not break or break the PLC, If given egregious user input, should not be a hazard to the user
  - [0.25] Durable – Should be able to survive a live lab environment
  - [0.25] Accurate - Reasonably accurate
- Future Use [0.33]
  - [0.25] Reproducible: The simulator should be able to reproduced easily
  - [0.5] Modifiable: The simulator should be able to easily modified
  - [0.25] Inexpensive: The simulator should be inexpensive to reproduce

List 1. Objective Tree.

2.2 Engineering Requirements

Table 1 summarizes the engineering requirements for this project.

Marketing Requirements	Engineering Requirement	Justification
#1	should reduce the time the user spends on the simulator	The device will prevent the student from needed to make manual changes in the simulator to test a scenario
#2	should be no more than an hour to learn to connect to the PLC	Making a device that is easy to use makes it enjoyable to work with in turn making it a more valuable tool
#3	Should not become a hazard if 24v dc is applied to any terminal.	If a student plugs in 24v, the simulator should not pose as a hazard to the user/PLC
#4	The device should survive a drop of 5 inches onto concrete	The device is to be used in the classroom and should not need maintenance for each use
#5	should display accurate information to a percentage of the expected	The data and simulation serve no purpose if it is not accurate and realistic
#6	cost less than $100	it should be no issue to replace if needed
#7	should be easily made by resources at Behrend	It should not be complicated to reproduce the simulator, this would help if there is a need for more of them.
#8	should have the option to add an I/O expander	The device is more valuable if it can expand to any use rather than be limited to a single simulation and a limited number of IO.
Marketing Requirements The Device Should Be: 1. Beneficial in an educational setting. 2. Easy to use for students and instructors. 3. Safe for the user/PLC. 4. Durable 5. Reasonably precise and have reasonable fidelity. 6. Inexpensive. 7. Easy to reproduce. 8. Easy to expand, and modify.		

Table 1. Engineering requirements.

2.3 Constraints

None.

2.4 Standards

None.

3 Design

In the following, it is examined which options would be a good fit to solve the problem. Following this, a selection is made.

3.1 Design Alternatives

This section describes the 3 design alternatives that were developed and considered.

3.1.1 BeagleBone Blue

The first option for a microprocessor to use is the BeagleBone Blue (BBB). The BBB will be connected to the PLCs I/O modules using General Purpose Input Output (GPIO) and analog signals. A simulation will be preloaded onto the BBB in order to evaluate the PLC process. As stated previously, the simulator must be able to interact with the I/O of the PLC in less than 1 millisecond at all times. With an operating system on board, some of the processor's time will be spent on OS maintenance. An interrupt would be defined in the kernel of the OS in order to satisfy this constraint. This interrupt would need to take precedence over all other processes. The BBB will also be connect to a computer via Ethernet. The PC will have a program that will interpret the results of the simulation visual, so that the user has a detailed understanding of the successes and quality of their PLC program.

3.1.2 Raspberry Pi B+

The second option for a microprocessor is the Raspberry Pi B+. The Pi will be connected to the PLC using GPIO and analog signals. The simulation will be preloaded onto the Pi and will take advantage of the lightweight OS and interrupts in order to meet the timing constraint. Unlike the BBB, the Pi has an HDMI port that can be connected to a monitor and USB for a keyboard and mouse. This will allow the visualization of the output from the simulation to be done on the device, not requiring a connection to a PC.

3.1.3 MCU and Raspberry Pi B+

The third option is to use both a MCU and Raspberry Pi B+. The MCU will act as the simulator and the Pi will be a visualizer. The MCU will be connected to the PLC with GPIO and analog. The simulation code will be preloaded. This design takes advantage the speed of an MCU and the fact that there is no OS, in order to meet the time constraint and guarantees near real time I/O. From the MCU, the Pi will receive the results of the simulation through I2C or SPI. The Pi will use HDMI and USB to connect to a monitor, mouse, and keyboard to provide an interface for the user. The Pi will have a program that easily allows the user to interpret their PLC program.

3.2 Design Selection

The three design alternatives were evaluated using a strengths and weaknesses analysis. Table 2 lists the most significant strengths and weaknesses for each alternative.

Design Alternative	**Strengths**	**Weaknesses**
BeagleBone Blue	- Fast memory storage ++ - 512MB Flash, 4GB (Main Memory) + - More efficient ARM architecture (better pipeline) +++	- Bulky OS, More OS maintenance - - - - Uses Ethernet to Visualize on a PC - - More expensive (~$90) - -
Raspberry Pi B+	- Lightweight OS, takes less time for OS maintenance +++ - Faster Processor (1.4GHz) ++ - Cheaper (~$35) +	- Only has SDRAM (1GB) Main Memory is stored on an SD card - - Installation of OS through SD Card (overhead to get started) - - -
MCU with Raspberry Pi B+	- More Interactive Simulation +++ - Higher Reliability + - MCU guarantees real time response +++	- requires a program in a higher level language to visualize - - - - Could require more debugging - - - Requires more overhead -

Table 2. Strengths and weaknesses analysis for design alternatives [3]-[5].

The chosen design is the MCU and Raspberry Pi B+ option because it offers a combination of Real Time Computing with a cheap operating system for visualization. The MCU provides real time access to the I/O of the PLC to the simulation. The Pi is a cheap and powerful single board computer that can handle a higher level program to interpret the simulation data and create an informative visualization for the user. Our team also has familiarity with the Pi and a PIC MCU, which will be in an advantage over using a new device. In addition to this, our team is more familiar with the Pi, and this prior knowledge should be of great assistance moving forward.

3.3 System Level Design

3.3.1 System Level Design: Level 0

The Level 0 design for the PLC Simulator is given in Figure 1. Details of the inputs, outputs and functionality are given in Table 1.

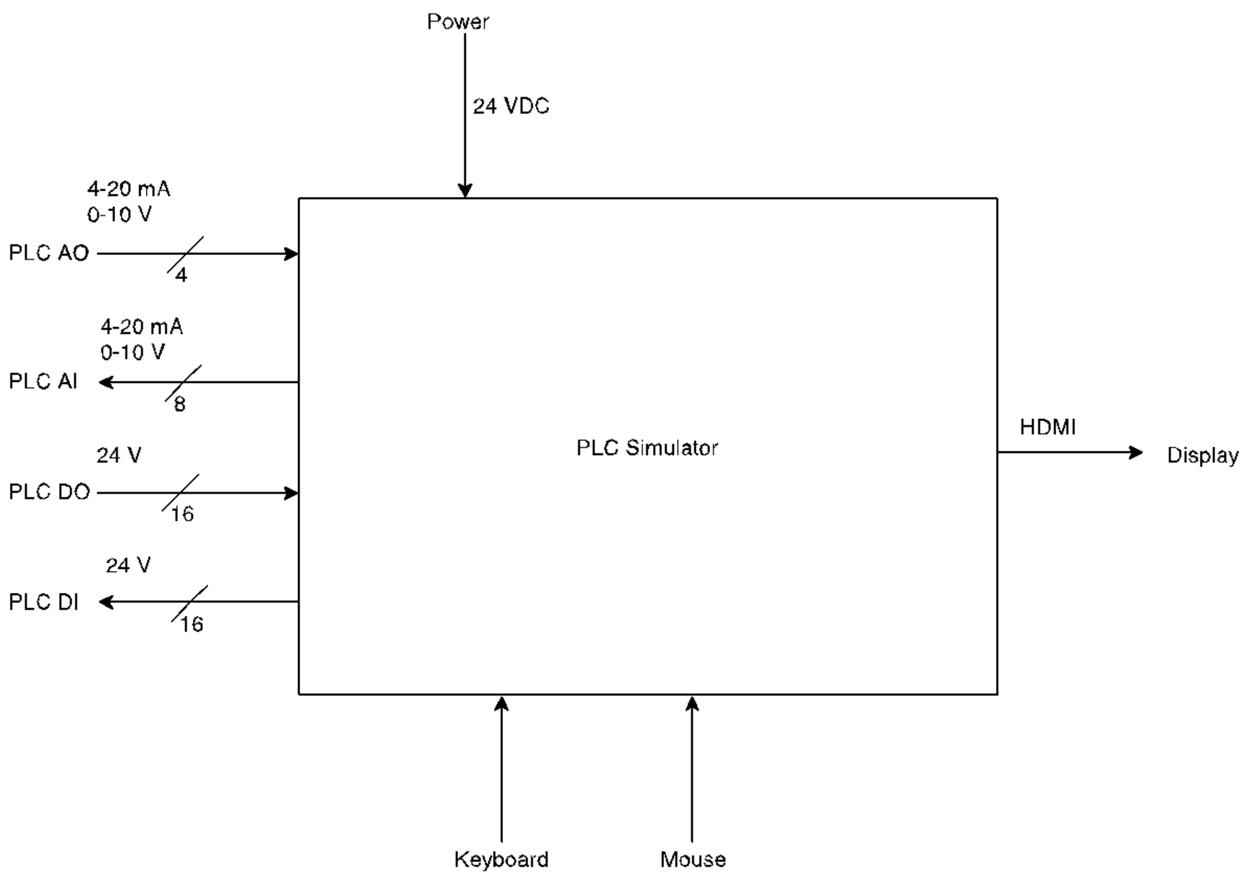

Figure 1. Level 0 design for the PLC Simulator.

Module	PLC Simulator
Inputs	PLC AO (Analog Out) – 4 Wires @ 4-20mA and 0-10V PLC DO (Digital Outputs) – 16 Wires @ 24V Keyboard Mouse Power – 24VDC
Outputs	PLC AI (Analog In) – 8 Wires @ 4-20mA and 0-10V PLC DI (Digital Inputs) – 16 Wires @ 24V HDMI Display
Functionality	The PLC Simulator acts as a simulator and visualizer of a process. It receives input from the PLC about the operator's actions to the process and gives feedback to the PLC. It also provides a visualization to the simulation.

Table 3. Level 0 details for the PLC Simulator

3.3.2 System Level Design: Level 1

The Level 1 design is given in Figure 2. Details of the inputs, outputs and functionality for each module in the design are given in Tables 2 through 6.

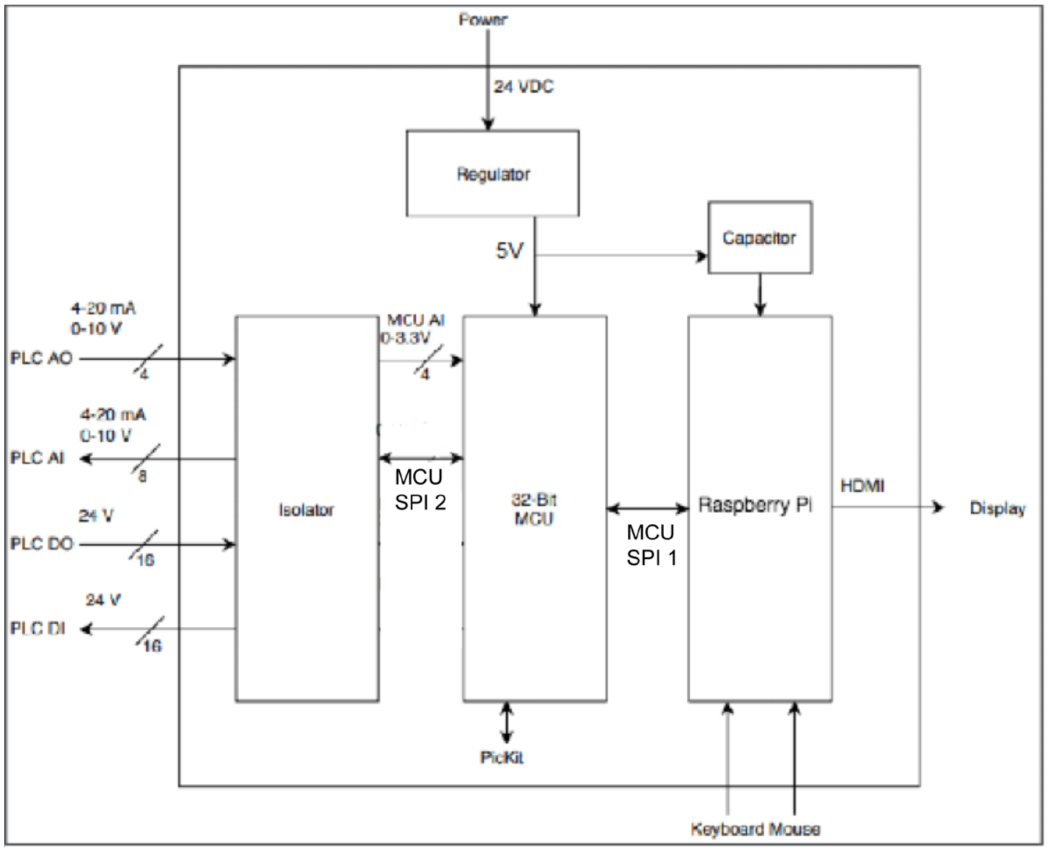

Figure 2. Level 1 design for the PLC Simulator.

Module	Isolator
Inputs	PLC AO (Analog Out) – 4 Wires @ 4-20mA and 0-10V PLC DO (Digital Outputs) – 16 Wires @ 24V MCU SPI 1
Outputs	MCU AI – 4 Wires @ 0-3.3V MCU SPI 1 PLC DI (Digital Inputs) – 16 Wires @ 24V PLC AI (Analog In) – 8 Wires @ 4-20mA and 0-10V
Functionality	The Isolator uses transistors to change the range of input from the PLC to 0-3.3V, this also serves to resize hazardous input. It also changes the range of input from the MCU from 0-3.3V to 0-10V. The SPI is used to communicate the analog values (from the MCU) to the DACs inside the Isolator.

Table 4. Level 1 Isolator details.

Module	32 Bit MCU Simulator
Inputs	Power MCU – 5V MCU AI – 4 Wires @ 0-3.3V MCU SPI 1 MCU SPI 2 PicKit Wires
Outputs	MCU SPI 1 MCU SPI 2
Functionality	Computes the Simulation of the process. Receives input from the Isolator, which receives input from the PLC. The simulator maps its inputs from the isolator to its outputs for the Raspberry Pi. In the future, this module is used for running the actual simulation of the process in question. Values of the MCU's Digital inputs and outputs are retrieved via MCU SPI 1 from the isolator. Values of the MCU's Analog Outputs are also pushed via MCU SPI 1 to the isolator. Any of the values intended for the raspberry pi are sent/retrieved with MCU SPI 2.

Table 5. Level 1 32 Bit MCU Simulator details.

Module	Raspberry Pi
Inputs	Power Pi – 5V Keyboard Mouse MCU SPI 2
Outputs	HDMI Display MCU SPI 2
Functionality	Produces the visualization of the simulation, and uses the inputs from the MCU as the data for the visualization. This is then sent to the display. The Raspberry Pi simultaneously collects data from the pins, and updates the visual display. Then, changes made are applied by sending them via MCU SPI 2 when requested.

Table 6. Level 1 Raspberry Pi details.

Module	Capacitor
Inputs	Power CAP – 5V
Outputs	Power Pi – 5V
Functionality	Provides a safe shut down for the BeagleBone by giving it a bleeding voltage when power is cut off.

Table 7. Level 1 Capacitor details.

Module	Regulator
Inputs	Power – 24VDC
Outputs	Power MCU – 5V Power CAP – 5V
Functionality	Takes a 24V input and scales it down to 5V for the MCU and Pi

Table 8. Level 1 Regulator details

3.3.3 System Level Design: Level 2

Details of the inputs, outputs and functionality for each module in the design are given in Tables 7 through 10.

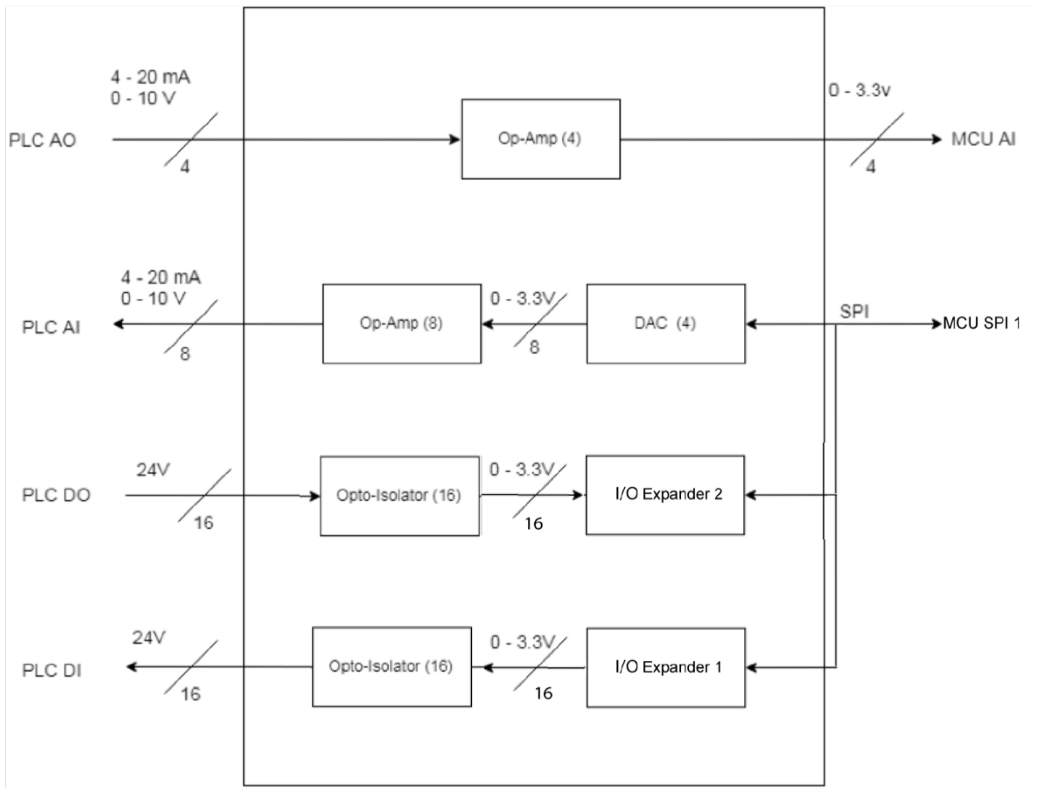

Figure 3. Level 2 design for Isolator.

Module	Opto-Isolator (16) (3rd row from top)
Inputs	PLC DO (Digital Output) – 0 or 24V
Outputs	I/O 2 Input - 16 Wires 0-3.3V
Functionality	Takes a 0-24V input and converts it to a 0-3.3V signal. Opto-Isolator will consider values 24V and greater to be 3.3V and values 0V and less will be 0V. An intermediate value might be chosen as a midpoint cutoff for the values.

Table 9. Level 2 Opto-Isolator (3rd from top) details.

Module	Opto-Isolator (16) (4th row from top)
Inputs	I/O 1 Input – 16 Wires 0-3.3V
Outputs	PLC DI (Digital Input) – 0 or 24V
Functionality	Takes a 0-24V input and converts it to a 0-3.3V signal. Opto-Isolator will consider values 24V and greater to be 3.3V and values 0V and less will be 0V. An intermediate value might be chosen as a midpoint cutoff for the values.

Table 10. Level 2 Opto-Isolator (4th from top) details.

Module	Op-Amp (8)
Inputs	DAC AO – 8 Wires @ 0-3.3V
Outputs	PLC AI (Analog Input) – 0-10V
Functionality	Takes a 0-3.3V input and converts it to a 0-10V signal. The purpose of this module is to ensure that the AO to the PLC is only between 0v and 10v. Input values outside the range can be considered statistically improbable, as the values are coming from a controlled environment (MCU).

Table 11. Level 2 Op-Amp (8) details.

Module	I/O Expander 1
Inputs	MCU SPI 1
Outputs	MCU SPI 1 I/O 1 Output – 16 Wires @ 0-3.3V *Technically unnamed
Functionality	I/O Expander 1 has the responsibility of transmitting the MCU's Digital Out Values to an Opto-Isolator. The I/O component is controlled by the MCU via SPI protocol (MCU SPI Wire), and has 16 wires connected to its partner Opto-Isolator via I/O 1 Output.

Table 12. Level 2 I/O Expander 1 details.

Module	I/O Expander 2
Inputs	MCU SPI 1 I/O 2 Input – 16 Wires @ 0-3.3V *Technically unnamed
Outputs	MCU SPI 1
Functionality	I/O Expander 2 has the responsibility of collecting the translated PLC DI values from its partner Opto-Isolator. It communicates with the MCU via SPI protocol, and is polled by the MCU for its values via the MCU SPI line.

Table 13. Level 2 I/O Expander 2 details.

Module	DAC (4)
Inputs	MCU SPI 1
Outputs	MCU SPI 1 DAC Output – 8 Wires @ 0-3.3V *Technically unnamed
Functionality	The MCU transmits the analog data it wishes to output in SPI protocol over the SPI wire. This is received by the DACs and is converted to an analog signal which is received by the Op-Amp (8).

Table 14. Level 2 DAC (4) details.

Module	Op-Amp (4) (Top module)
Inputs	PLC AO (Analog Output) – 0-10V
Outputs	MCU AI – 4 Wires 0-3.3V
Functionality	Takes a 0-10V input and converts it to a 0-3.3V signal. The purpose of this module is to ensure that any voltage 0v to 10v is changed to the range 0-3.3v. If a voltage is higher than 10v, it is output as 3.3v.

Table 15. Level 2 741-PLC AO details.

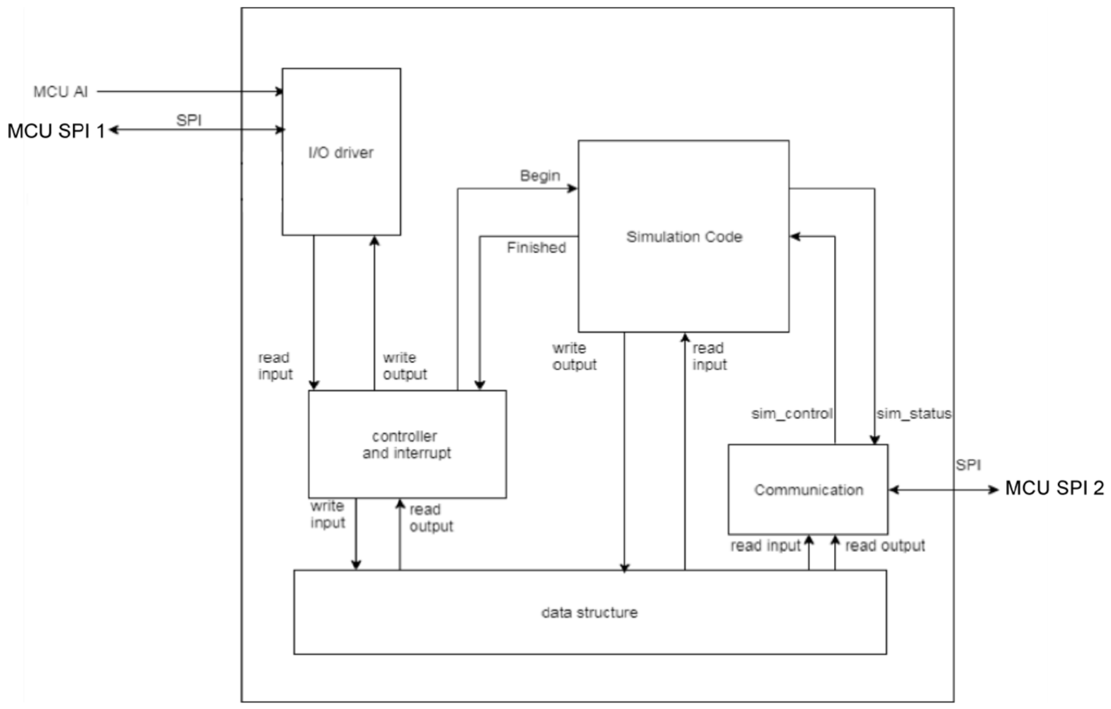

Figure 4. Level 2 Design for MCU.

Module	I/O driver
Inputs	MCU SPI 1 MCU AI - 4 Wires, 0-3.3V *For simplicity, slash is not shown in picture Write output
Outputs	Read input MCU SPI 1
Functionality	The I/O driver be the interface to the wires coming from the isolator. The SPI wire connects here, as well as the Analog Input wire.

Table 16. Level 2 I/0 Driver details.

Module	Controller and interrupt
Inputs	Read input Read Output Finished
Outputs	Write output Write input Finished
Functionality	The controller has an interrupt set to go off every millisecond. In this intercept, it will read the inputs from the I/O driver and then write to them through write output. It will always tell the simulation when to run through the begin signal and when it returns the finished signal, the controller will read the results through read output. These outputs and then written to MCU DO and MCU AO through write output.

Table 17. Level 2 Control and interrupt details.

Module	Data structure
Inputs	Write input write output
Outputs	Read input read output
Functionality	This is a data array containing a copy of the MCU Analog Input, Analog Output, Digital Input and Digital Output. Write Input copies AI and DI into a data array. Write Output copies AO and DO from the result of the simulation code into the array. Read input copies AI and DI from the array to the communication module. Read output copies AO and DO from the array to the communication module.

Table 18. Level 2 data structure details.

Module	Simulation Code
Inputs	Begin Read input Sim_control
Outputs	Write output Finished Sim_status
Functionality	When the Simulation code receives the begin signal, it will read the inputs from the data structure through read input and then complete one iteration. It will then write the output back to the data structure through write output. Lastly it will send a signal on finished to the controller, to let it know that the outputs are ready. Sim_control are given by the user and act as functions such as, reset the simulation. Sim_status sends details about the simulation to the user such as, the tank temperature or the tank level.

Table 19. Level 2 Simulation Code details.

Module	Communication
Inputs	Read input Read output Sim_status MCU SPI 2
Outputs	Sim_control MCU SPI 2
Functionality	The communication module manages a connection between the MCU and the Raspberry Pi. Over the SPI connection it will send a copy of the inputs and outputs as well as sim_status to the Raspberry Pi for visualization purposes. It may also receive sim_control which it then sends to the Simulation code to be handled.

Table 20. Level 2 Communication details.

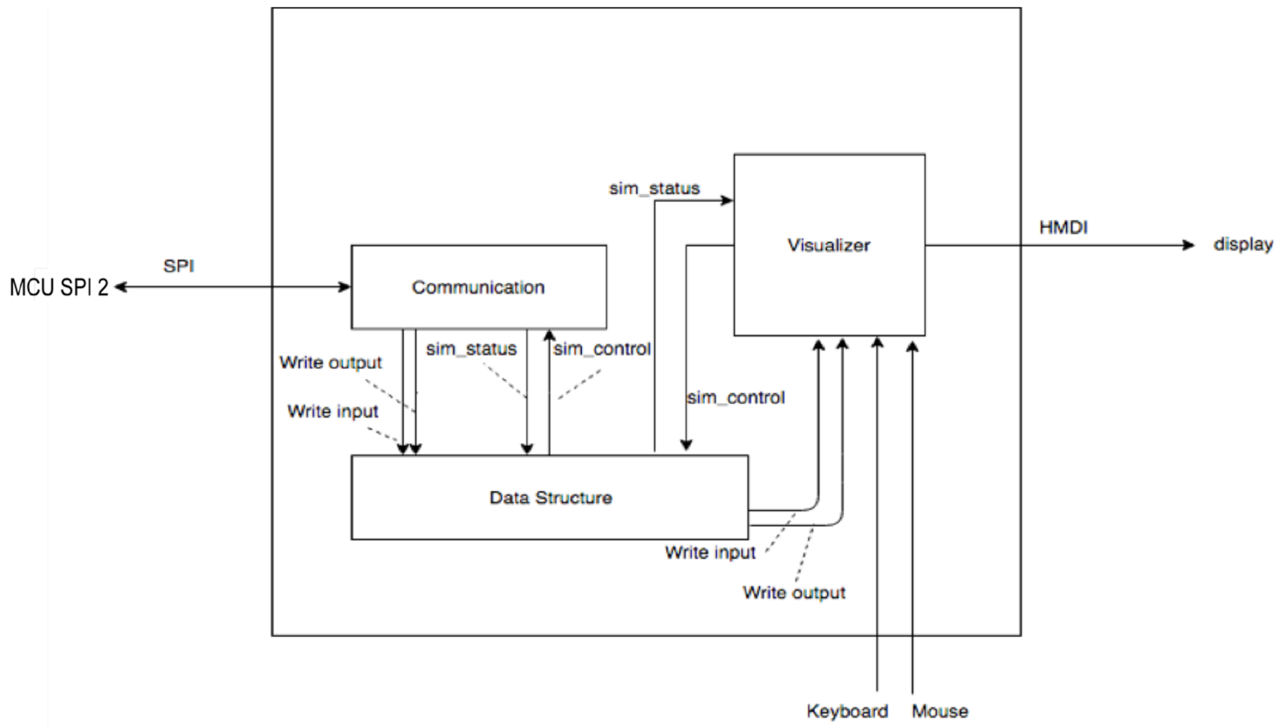

Figure 5. Level 2 Design for Raspberry Pi.

Module	Communication
Inputs	MCU SPI 2 Sim_control
Outputs	Write output Write input Sim_status MCU SPI 2
Functionality	The Communication module manages the link between the MCU and Pi. It will receive a copy of the inputs, outputs, and sim_status and save them to the data structure. It will also transmit and sim_control signals that are generated by the user.

Table 21. Level 2 Communication details.

Module	Data structure
Inputs	Write output Write input Sim_status Sim_control
Outputs	Sim_control Sim_status
Functionality	The data structure is an array of data that contains a copy of the inputs, outputs, sim_status, and sim_control.

Table 22. Level 2 Data Structure details.

Module	Visualizer
Inputs	Write output Write input Sim_status Keyboard mouse
Outputs	Sim_control HDMI
Functionality	The visualizer will receive inputs, outputs, and sim_status from the data structure. A higher-level program will interpret this data and create a visualization to a monitor through the HMDI. Through the use of the keyboard and mouse the user may give a sim_control (ex. A reset button) which is saved in the data structure and eventually send to the simulation.

Table 23. Level 2 Visualizer details.

4 Testing

The follow details several tests that will verify the functionally and fidelity of modules as well as the system as a whole.

4.1 Unit Tests

One example of a unit test is a test of the isolator, specifically Opto-Isolator (16) (3rd from top). The important fact to note is what the isolator should do. The isolator does not allow values outside the range to propagate outside the range. They are clipped to the limits, as is shown. The first step would be to test 0V and make sure that we get 0V on the output. Then test 10V and make sure we get 3.3V. Then test 24V and make sure we only get 3.3V on the output

Test Writer:						
Test Case Name:		Isolator Test 1: Opto-Isolator (16) (3rd from top)			Test ID #:	
Description		Verify the input and output voltages of the Isolator Module, Opto-Isolator (16) (3rd from top)			Type:	
Tester Information						
Name of Tester:					Date:	
Hardware Ver:					Time:	
Setup		Separate the Isolator module from the rest of the system. Apply a voltage to the PLC DO terminal and check the result of MCU DI with an oscilloscope.				
Step	Input	Expected Output	Pass	Fail	N/A	Comments
1	PLC DO - 0v	MCU DI - 0v				
2	PLC DO voltage is greater than 3.3V	MCU DI – 3.3v				
3	PLC DO - voltage is greater than 0V	MCU DI – 3.3v				
4	PLC DO -voltage is less than 0V	MCU DI - 0v				
Overall Test Results						

Table 24. Matrix test for Unit Test 1.

4.2 Integration Tests

One example of an integrations test would be to test that the PLC AO and DO are properly and safely read by the MCU.

Test Writer:						
Test Case Name:		Isolator Test 2: Full Test			Test ID #:	
Description		Verifies that the Isolator module is fully working with converting values for all of its modules.			Type:	
Tester Information						
Name of Tester:					Date:	
Hardware Ver:					Time:	
Setup		Connect the PLC and MCU terminals to the isolator. Run a program on the PLC. The simulation code in the MCU simply reads inputs and copies them to outputs.				
Step	Input	Expected Results	Pass	Fail	N/A	Comments
1	PLC DO - 10v	MCU DI - 3.3V				
2	MCU AO – (SPI) Value of 3.3V	PLC AI - 10v				The SPI protocol contains a payload value, which is translated in the DAC to an analog output.
3	PLC AO - 10v	MCU AI – 3.3v				
4	MCU DO – 3.3v	PLC DI - 10v				
Overall Test Results						

Table 25. Matrix test for Integration Test 1.

4.3 Acceptance Tests

Acceptance test would be to check the latency from the I/O of the PLC and to the monitor.

Test Writer:			
Test Case Name:	Device Acceptance test 1	Test ID #:	
Description	Test the full functionally and fidelity of the device	Type:	
Tester Information			
Name of Tester:		Date:	
Hardware Ver:		Time:	
Setup	A PLC is connected to the isolator terminals and the BeagleBone is connected to a monitor, keyboard, and mouse.		

Step	Input	Expected Output	Pass	Fail	N/A	Comments
1	PLC program turns on a valve	The PLC should see an output that the valve is on in less than one millisecond.				
2	The user enters an option to simulate power failure to a module	Outputs to the specified module are off				
3	The PLC program has had a valve that has been open for sometime	The display should show the level of water in the tank as well as any corresponding temperatures				
4	The device is dropped onto concrete at height of 5 inches	The device should operate normally afterwards				
Overall Test Results						

Table 25. Matrix test for Acceptance Test 1.

5 Summary

PLC's are widely used in industrial applications for manufacturing processes. There is a need for a device that can simulate and evaluate PLC programs. An MCU is to be used to run this simulation, and interface with a PLC. A single board computer, Raspberry Pi, is to be used to interface with the MCU and produce a visualization of the simulation. There is a component, called the isolator, which has characteristics such that it can translate the range of input from the PLC to an appropriate range for the MCU and vice versa. SPI is used to communicate with the DAC's, in the isolator, to the MCU and SPI to the single board from the MCU. Our testing is designed to support said claims about the components, such that the design follows the engineering requirements. Moving forward with development, work will be focused on the intersection of the testing and component development to produce a robust prototype of the design.

References

[1]W. Bolton., *Programmable Logic Controllers ; Sixth Edition*. 2015.

[2]W. George, NSF Project Proposal (need more info)

[3]"BeagleBoard.org", *Beagleboard.org*, 2018. [Online]. Available: https://beagleboard.org/blue. [Accessed: 10- Oct- 2018].

[4]"Raspberry Pi 3 Model B+ - Raspberry Pi", *Raspberry Pi*, 2018. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. [Accessed: 10- Oct- 2018].

[5]"Industry's Broadest and Most Innovative 32-bit Microcontroller (MCU) Portfolio," AVR Assembler -. [Online]. Available: https://www.microchip.com/design-centers/32-bit. [Accessed: 28-Nov-2018].