4-2017

Raspberry Pi Human Machine Interface and Control System for an Electromagnet Water Filter

Thad Haines
Montana Tech of the University of Montana

Frank Joyce
Montana Tech of the University of Montana

Raspberry Pi Human Machine Interface and Control System for an Electromagnet Water Filter

Thad Haines and Frank Joyce

Montana Tech

April 21, 2017

Table of Contents

List of Figures

List of Tables

Abstract

The purpose of this senior design project was to take an ongoing project's manually controlled water filtration system with no data logging and transform it into an easily configurable automated open-loop process with data logging.

Instead of using an off the shelf programmable logic controller, a Raspberry Pi was chosen to act as the master controller and data logger. This choice enabled easier end-user human-machine-interface, convenient data log accessibility, and a wide variety of future expansion opportunities.

The physical system being controlled consists of three pumps, four valves, and two magnets that are energized or de-energized by relays which operate according to voltage levels sent from the Raspberry Pi.

A python coded GUI displayed on a touchscreen allows user input of mode run times and the number of cycles the process should repeat while also displaying continuously updated system information.

In addition to automated control, the user has the option of manually controlling the system by way of physical switches on an external panel with LEDs that serve as a visual indicator of system status.

The final system designed was considered a success as a data log was produced containing the systems operation along with correct temperature and pH data that match the expected and observed system operation through an automated cycle and a manually controlled cycle.

Introduction – Why the Project Happened

The opportunity to create some kind of control and data logging system was originally proposed to Thad Haines by Dave Hutchins at the Silver Dollar Saloon in late March of 2016. Dave, an environmental engineering graduate student, happened to be working with Dr. Jerry Downey on the development of a continuous-flow ion-exchange reactor to recover metals from very dilute wastewater streams. This reactor system was initially described by Dave as 'a couple pumps, some valves and two magnets that filter water'. A rough system sketch was then produced on the back corner of a torn up roller derby hand bill (see Figure 1).

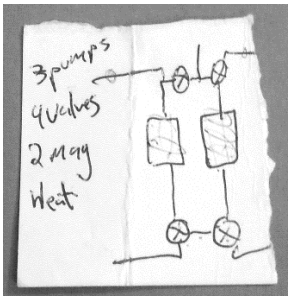

Figure 1. Initial system description.

3 months later, Thad agreed to work on the project for his senior design and a meeting was held with Dave so that more detailed information could be obtained. From this meeting, initial goals of being able to log temperature data of each magnet within ±1 °C and the input pH every second, as well as being able to manually control the system flow in addition to creating and running various timed automated cycles were set. Dave suggested the using a Raspberry Pi (RPI) and LabVIEW on the project and provided a more official system diagram (see Figure 2).

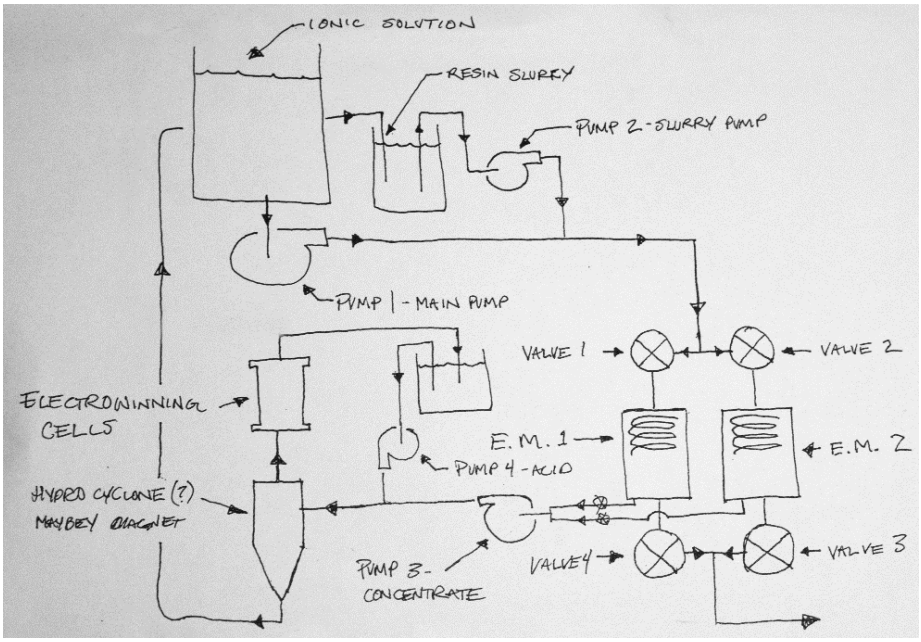

Figure 2. Overall system diagram.

After researching PLCs, pH sensors, and LabVIEW; the following observations were made:

While a PLC would be an obvious choice for system automation, since the reactor itself was still in a prototype phase, there were no set 'flow regime' times to program in. This meant that someone would have to reprogram the PLC whenever a different time setting was desired. Placing this burden on users of the system seemed like something to avoid or, if completely unavoidable, make as easy as possible.

Since pH is logarithmic, the accuracy of this data point was very important to the project. An Orion Star A214 pH (OSpH) meter was already being used by the group to collect data that met their desired requirements. As luck would have it, this particular meter had the ability to communicate readings via the serial communication standard RS-232.

LabVIEW, while apparently being fairly common for measurement collection, presented two large issues. The graphical approach to programming required a less than familiar workflow and code organization scheme compared to a more familiar script style language. Also, while Montana Tech did at one point have a LabVIEW license, that time has passed and a new license was beyond the self-imposed $1000 total cost limit of the entire project.

Due to these few key points, it was decided that the RPI was to act as the master controller and data logger for the project. Since the RPI has a very large user base and is very Python friendly, it was predicted that anything the project required could be accomplished using open source code. This meant licensing fees could be avoided, serial communication could be used to read pH data from the pre-existing meter, and a custom graphical user interface (GUI) could be designed for users to easily set mode times and view real-time system data.

While somewhat simplifying certain parts of the project, a RPI solution greatly increased the amount of software development and would require the use of appropriately sized relays, an update to the current manual control section, and a more customized hardware solution in general. As such, in late August, Frank Joyce was brought in on the project to develop the hardware solution of the system.

With this team, general direction, and list of goals in mind, the project was expected to be complete and producing data logs of a functional system by early April 2017.

How it Works

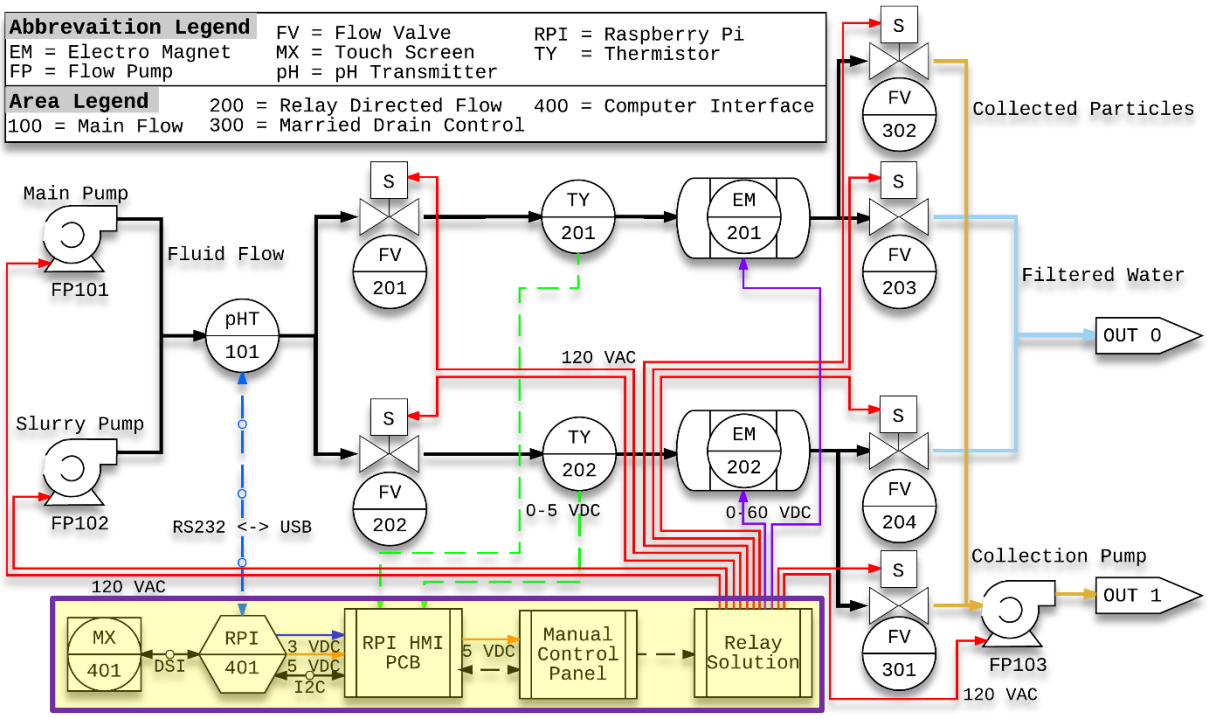

Figure 3. P&ID of system.

The piping and instrumentation diagram above shows a more standardized version of the system to be controlled. Essentially, fluid will be pumped through a system of valves that direct the flow around electromagnets that will be used to collect magnetic particles from suspension. The shaded yellow region indicates what was improved upon or added to the original system which includes the RPI and touchscreen, a customized PCB, an updated manual control panel, a relay solution, and data acquisition of temperature and pH data. The following sections will describe each item in further detail. A detailed list of hardware used and associated costs is provided in Appendix A.

Raspberry Pi and Touchscreen

The driving force behind the automation and data collection of the system is a Raspberry Pi 3 (see Figure 4). Table 1 lists a variety of specs for the Pi – however, the most useful features included the number of available universal serial bus (USB) ports, onboard Wi-Fi, inter-integrated circuit (I2C) capability, and a DSI connector that enabled use of a 7" touchscreen.

Figure 4. Raspberry Pi 3.

Table 1. Raspberry Pi 3 Specs.

System on Chip:	Broadcom BCM2837
CPU:	4X ARM Cortex-A53, 64-bit, 1.2 GHz
GPU:	Broadcom VideoCore IV
RAM:	1 GB LPDDR2 (900 MHz)
Networking:	10/100 Ethernet, 2.4 GHZ 802.11n
Bluetooth:	4.1 Classic, Low Energy
microSD:	Sandisk Ultra 8GB Class 10 microSDHC
GPIO:	40-pin header, populated
Ports:	HDMI. 3.55mm analogue AV jack, 4xUSB 2.0, Ethernet, CSI (Camera serial interface), DSI (Digital serial Interface)

One of the USB ports is dedicated to communicating to the OSpH meter through a USB-RS232 adapter while another is reserved for a thumb drive that contains the custom code and stores the created data logs.

The I2C bus is used for communication to 3 general purpose input output (GPIO) expanders and a 12 bit analog to digital converter (ADC). Two 8 channel expanders serve as digital output banks used to trigger to each device's associated relay while the 3rd expander is 16 channels and functions as a digital in to monitor system status. The monitored point of this 3rd expander

is after the manual control and thus reflects the actual voltage level seen by the relay. A complete expander to relay description may be found in in Table 7 in Appendix B. The rational for two 8 channel output expanders instead of one 16 channel expander had to do with a max total current limit of 125 mA from each expander. Each relay was allotted 20 mA for proper operation and as such, required more than one expander could provide.

The ADC on the I2C bus collects voltages from two thermistors placed inside the fluid flow near each magnet. This voltage level is converted to a useful temperature through a coded lookup table and a linear approximation process. Since the thermistors are to be used in an enclosed space, testing of the thermistors was carried out prior to installation. The acceptable results from these tests, are shown in Table 8 in Appendix C.

As shown in Figure 5, the touchscreen is oriented vertically and the initial interface is very similar to one that may be found on any modern PC. The chosen touchscreen is an official RPI product and provided little to no difficulty in installation, configuration, or use.

A 5 VDC, 2.4A power supply is used for powering the RPI and touchscreen as well as all manual control panel LEDs and GPIO expanders. This amperage was found to be more than sufficient for all required components to operate at once.

Figure 5. Vertically Configured Touchscreen.

Python Coded GUI

The code involved in the project utilized built-in Python 3.4 libraries for most functionality, an open source adafruit library for easier analog to digital conversion, and PySide Qt bindings for GUI elements. This combination of software was chosen so that the finished project would adhere to the GNU Lesser General Public License Guidelines (LGPL). This essentially means no licensing fees were required and, if desired, the code could be sold or incorporated into other proprietary projects as long as the LGPL code was made available to any end users.

The completed GUI is shown below in Figure 6. As shown on the left side of the figure, mode times are selected via dropdown menus. The right side of the figure shows a typical Run Mode screen where system time, system status, device status, pH, and temperatures update every second while mode times updates every 0.05 second.

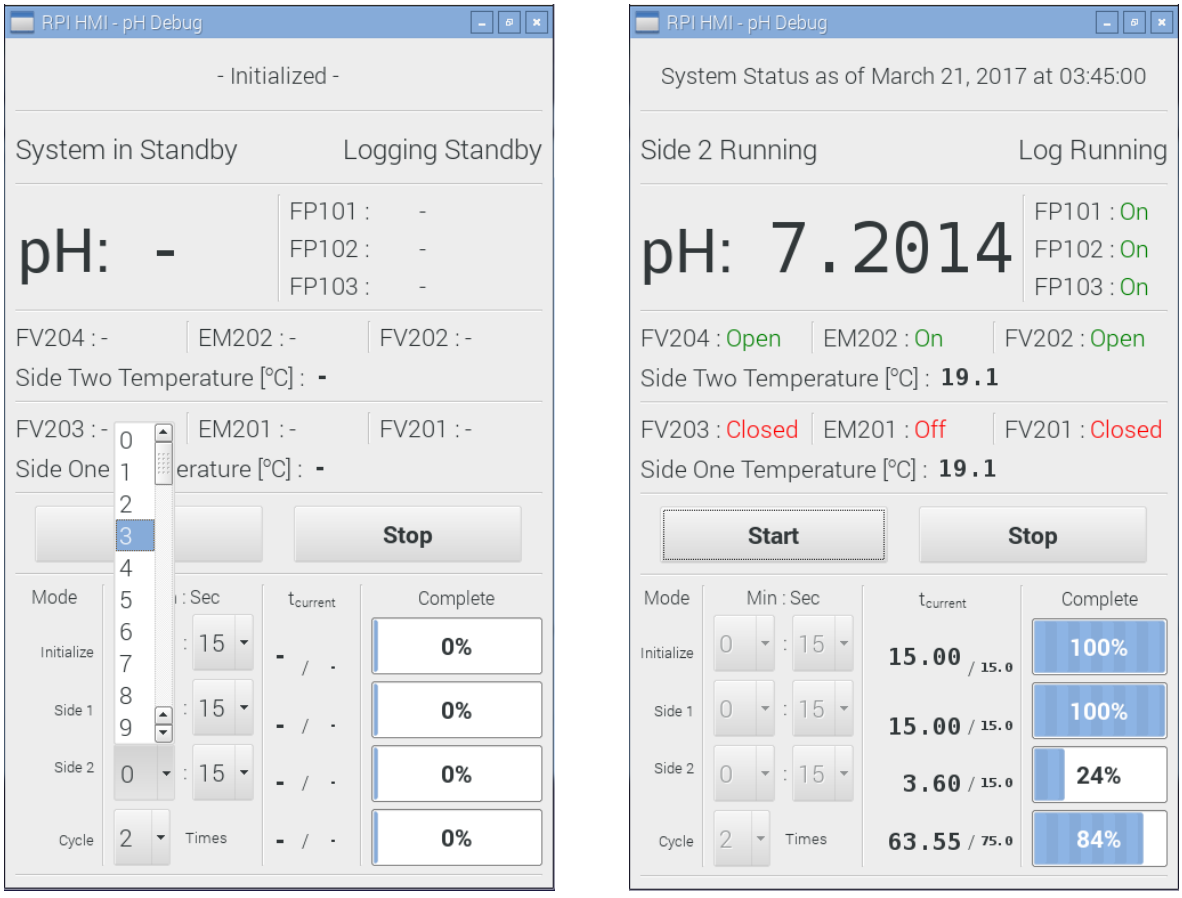

Figure 6. Finished GUI. Left: Configuration Mode, Right: Run Mode.

While a detailed code explanation is beyond the scope of this paper, simplified code flow diagrams are presented in Appendix D. Essential qualities of the code involve multithreading, csv reading and writing, logging, I2C communication, RS-232 communication, ADC, linear approximation, and global queues.

Raspberry Pi HMI PCB

A printed circuit board (PCB) was designed for easier connectivity between the Raspberry Pi, manual control panel and thermistors. The general PCB idea is depicted below in Figure 7.

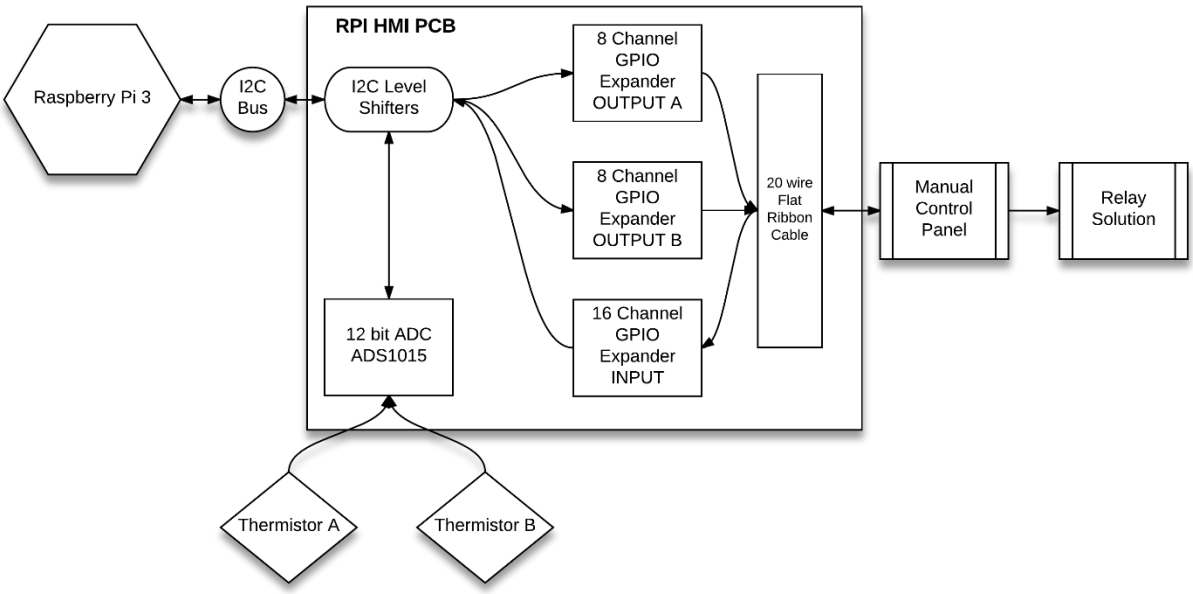

Figure 7. Basic overview of PCB Connectivity**.**

Another feature added to the PCB was a set of I2C Level Shifters. These enabled the GPIO expanders and ADC to operate at 5 V instead of the default 3.3 V. This meant the relays selected could have a higher threshold voltage and the ADC would have a larger usable range. One Multisim simulation of the level shifter circuit is shown in Figure 8 while full results of these level shifter simulations may be found in Appendix E.

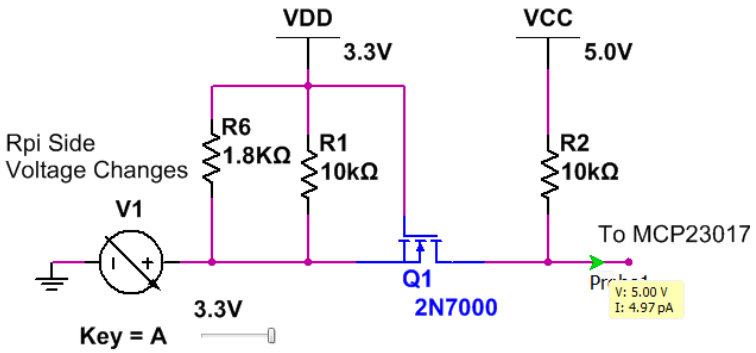

Figure 8. MOSFET level shifter circuit.

As shown above, a low voltage on the left side results in higher voltage on the right side and vice versa. A zero voltage on either side will result on a zero voltage on the opposite side. This is required because I2C does the sending and receiving of data on the same channel.

Due to ease of use, Express PCB was chosen to manufacture the designed PCB. While not required, a schematic was made in Express SCH to generate a netlist that was then used in Express PCB to aid in the layout of board traces and components. The Express SCH schematic may be found in Appendix E while the final PCB design is shown below in Figure 9.

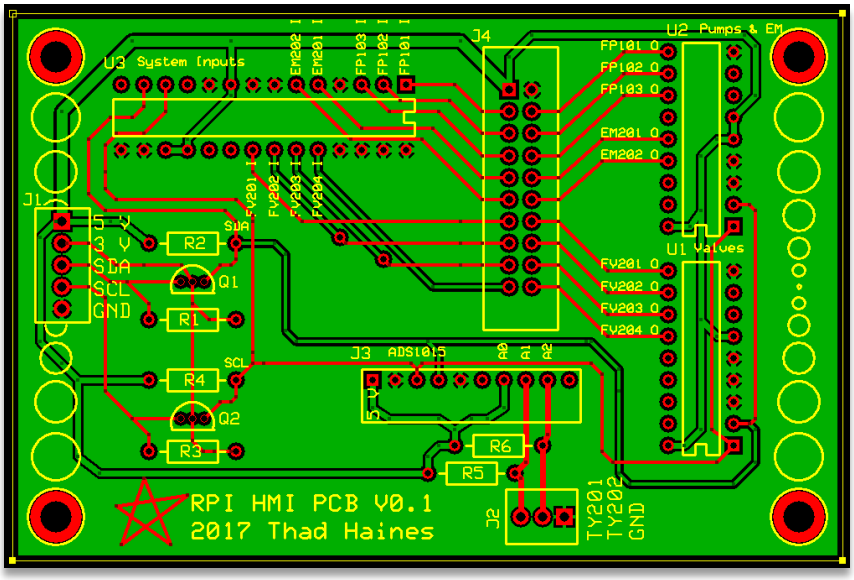

Figure 9. Final Express PCB schematic.

After the PCB was manufactured, 120 solder points were required in order to populate the PCB with specified components. A completely populated PCB is shown in Figure 10.

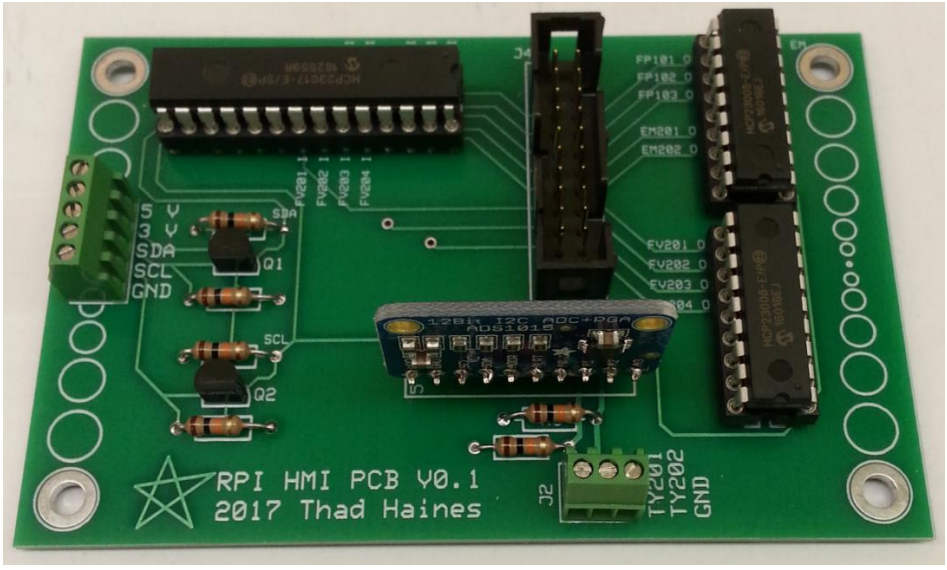

Figure 10. Populated PCB.

After the PCB was verified as working correctly, it was fixed to the back of the manual control panel for easier accessibility to the required switches and thermistors.

Manual Control Panel

Since the system had to have each device be controllable by automatic and/or manual control, the existing manual control panel, Figure 11, had to be redesigned.

Figure 11. Original Manual Control Panel.

A new faceplate layout, Figure 12, was created in SolidWorks and is shown below. Each control space was designed to be composed of 3 switch plates that each control an individual device.

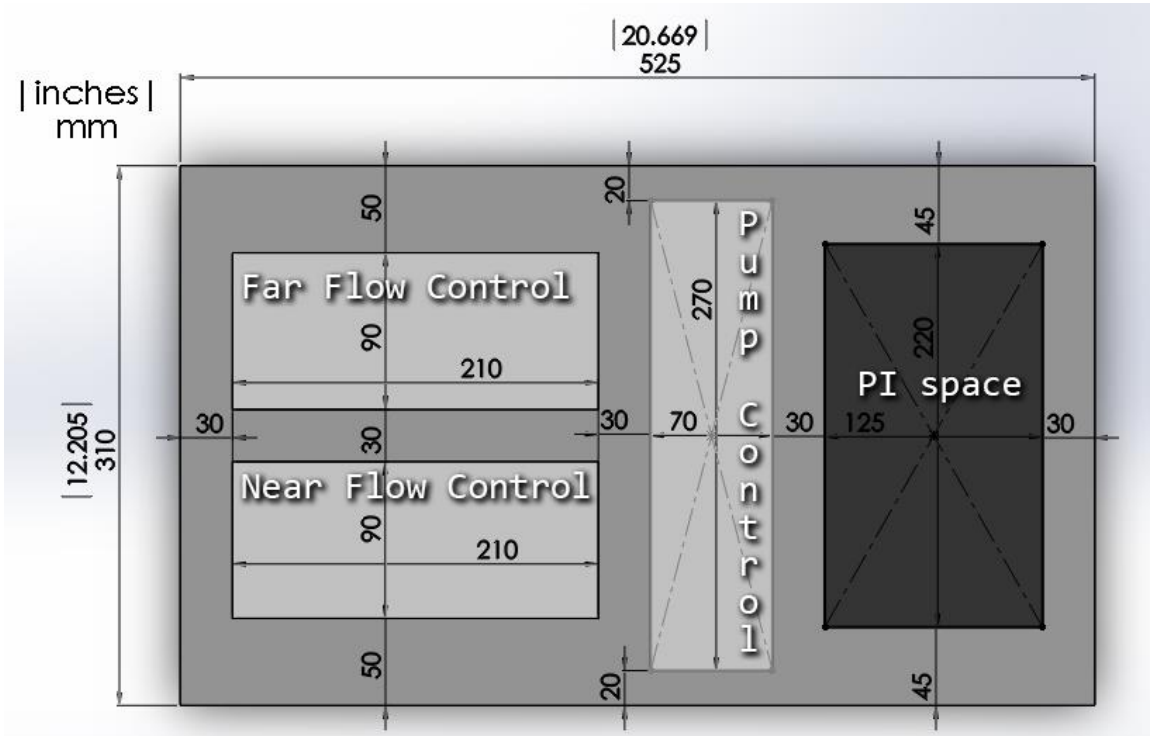

Figure 12. Proposed New Manual Control Panel Layout.

The design for the switch plates, Figure 13, was done using SolidWorks and then 3D printed.

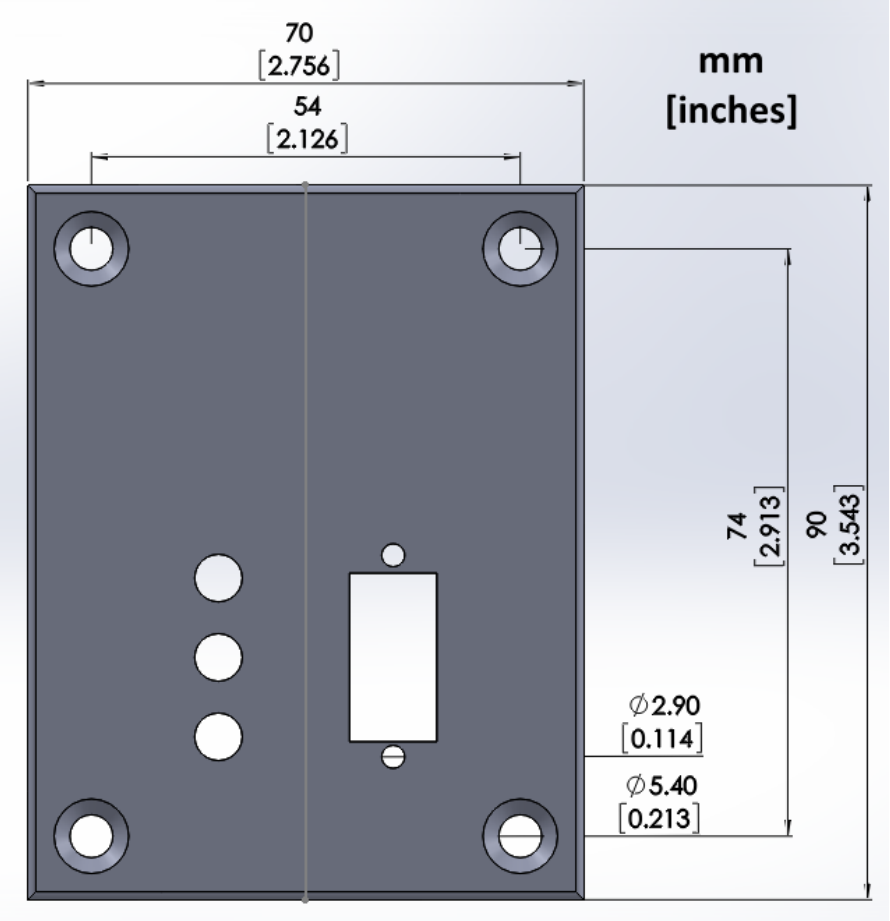

Figure 13. Design for Switches.

Each switch has three LEDs to show the state of the device, a toggle switch to change the operation of each device, and a blank top area for a label. The physical wiring of each toggle switch shown in Appendix E and the operation of each switch is described below in Table 2.

Table 2. Switch Plate Operation.

Toggle Position	Device Operation	Green LED	Red LED	Blue LED
Top	Always On	On	Off	Off
Middle	Always Off	Off	On	Off
Lowest	Automation Run	Varies according to programming	Off	On

The completely updated manual control panel is shown in Figure 14 on the next page.

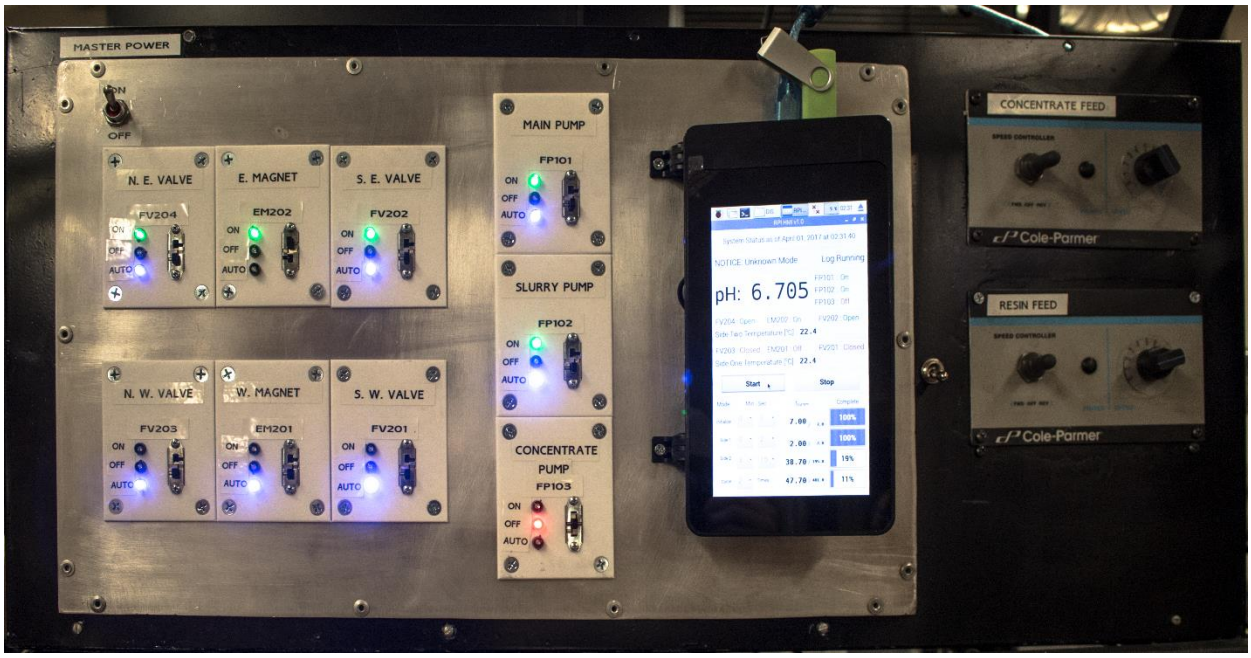

Figure 14. Final Manual Control Panel.

As shown above, each switch plate is labeled according to what device it controls and provides visual feedback as to the current operating status of the device. The physical size of the control panel allowed pump controls to be mounted to the right of the newly designed panel.

Relay Solution

The last piece of the project that needed to be designed for were the relays. The relays needed to be able to handle the various outputs of the system both in terms of rated load and control voltage (see Figure 15 & Table 3).

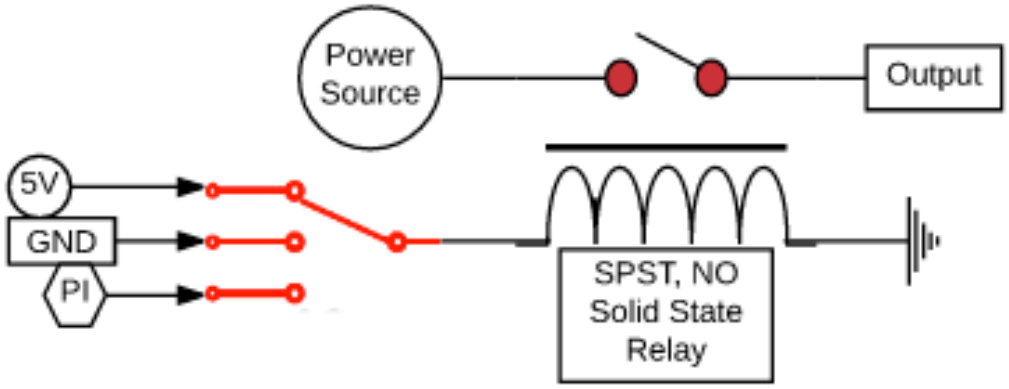

Figure 15. Basic relay model for outputs.

Table 3. Output/Relay Parameters.

Outputs	Power Source	Relay Brand	Current Rating	Voltage Rating	Maximum Current Draw	Switch, Contact
FP101	120VAC	Omega	25A	24 to 280VAC	7.5A	SPST, NO
FP102	120VAC	Omega	10A	24 to 280VAC	3A	SPST, NO
FP103	120VAC	Omega	10A	24 to 280VAC	3A	SPST, NO
FV201	120VAC	Sainsmart	2A	74 to 240VAC	0.3A	SPST, NO
FV202	120VAC	Sainsmart	2A	74 to 240VAC	0.3A	SPST, NO
FV203	120VAC	Sainsmart	2A	74 to 240VAC	0.3A	SPST, NO
FV204	120VAC	Sainsmart	2A	74 to 240VAC	0.3A	SPST, NO
EM201	60VDC	Omega	20A	0 to 100VDC	11A	SPST, NO
EM202	60VDC	Omega	20A	0 to 100VDC	11A	SPST, NO

The figure and table above show the basics of what needed to happen electrically for the relays and their associated outputs. The relays needed to be able to handle the various load currents and voltages while also being able to be controlled by a 0V-5V signal.

The wiring of the relays was pretty simple. 7 of the 9 devices were controlled with 120VAC and therefore there just needed to have a break in the ‘hot’ (black colored) wire and run through the relays’ switch contacts (see Figure 16).

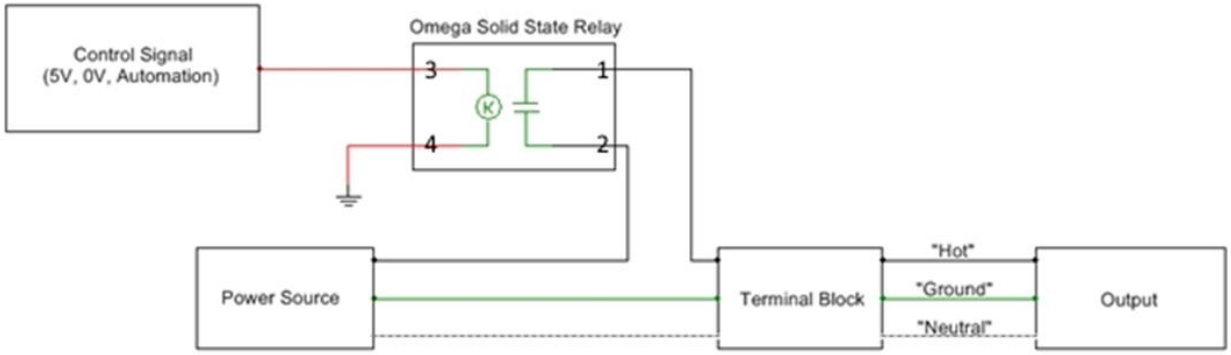

Figure 16. Diagram of wiring for a three-phase output.

The two electromagnets are DC loads powered by a BK Precision 1902 DC Switching Power Supply. The loads are controlled with a constant current from the supply and since the two loads are highly inductive, extra protection in the form of a flyback diode needed to be added in order to protect the relay contacts (Equation 1).

$$V_L = L * \frac{di}{dt} \quad (1)$$

As seen in Equation 1, when the current of an inductive load changes very rapidly (is stopped abruptly), the voltage will increase to a very large magnitude. By placing the flyback diode in parallel with the load, a path is provided for the inductor current to flow when the circuit is interrupted. The reverse breakdown voltage of the diode needed to be higher than the voltage driving the circuit (60 VDC) and the forward current needed to be higher than the current seen by the load (11 A). These diodes allow for extra protection for the relay and will also give it a longer life span.

Another consideration due to the high voltages and currents being used for the loads were the wire sizes connecting both the AC and DC circuits. For the AC outputs, an online NEC wire size calculator was used (see Figure 17).

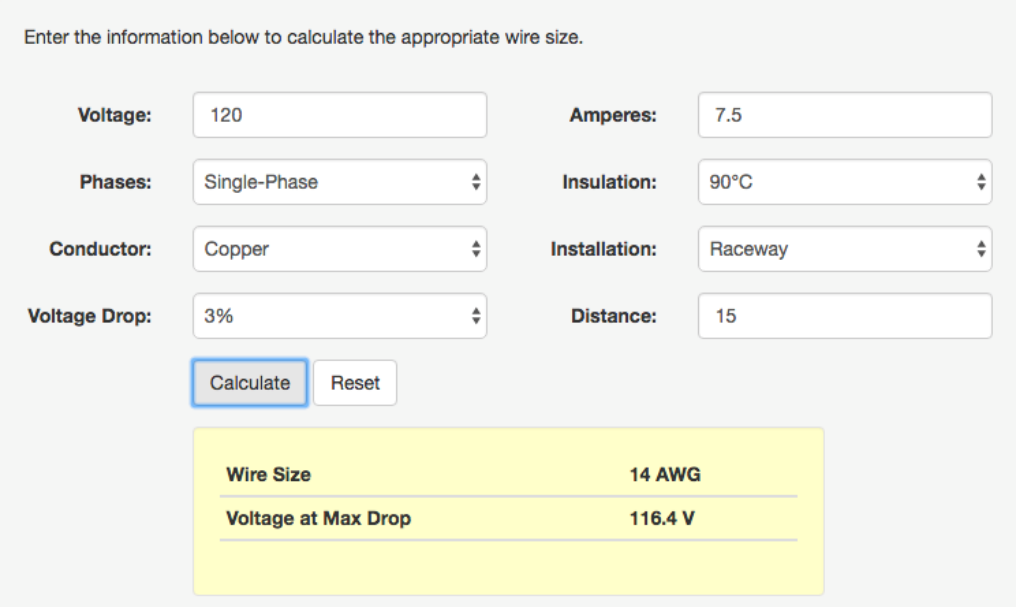

Figure 17. Wire size calculator for AC outputs.

For the DC outputs, a table was used (see Figure 18).

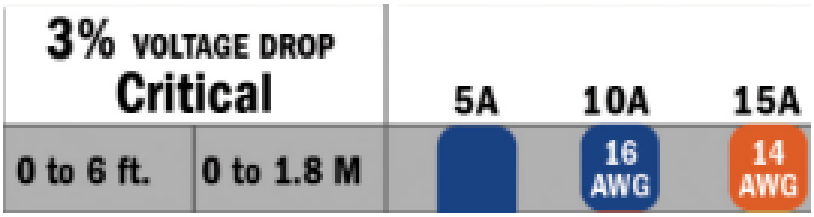

Figure 18. Wire size table for DC outputs.

Since a supply of 14 AWG wire was already on hand, it was desired to keep everything at or under this gauge. By using the length of the wire as the variable, it was quite easy to achieve this.

All of the relays, heat sinks, wiring, and everything associated with powering the system was required to fit into a pre-existing enclosure near the system (see Figure 19).

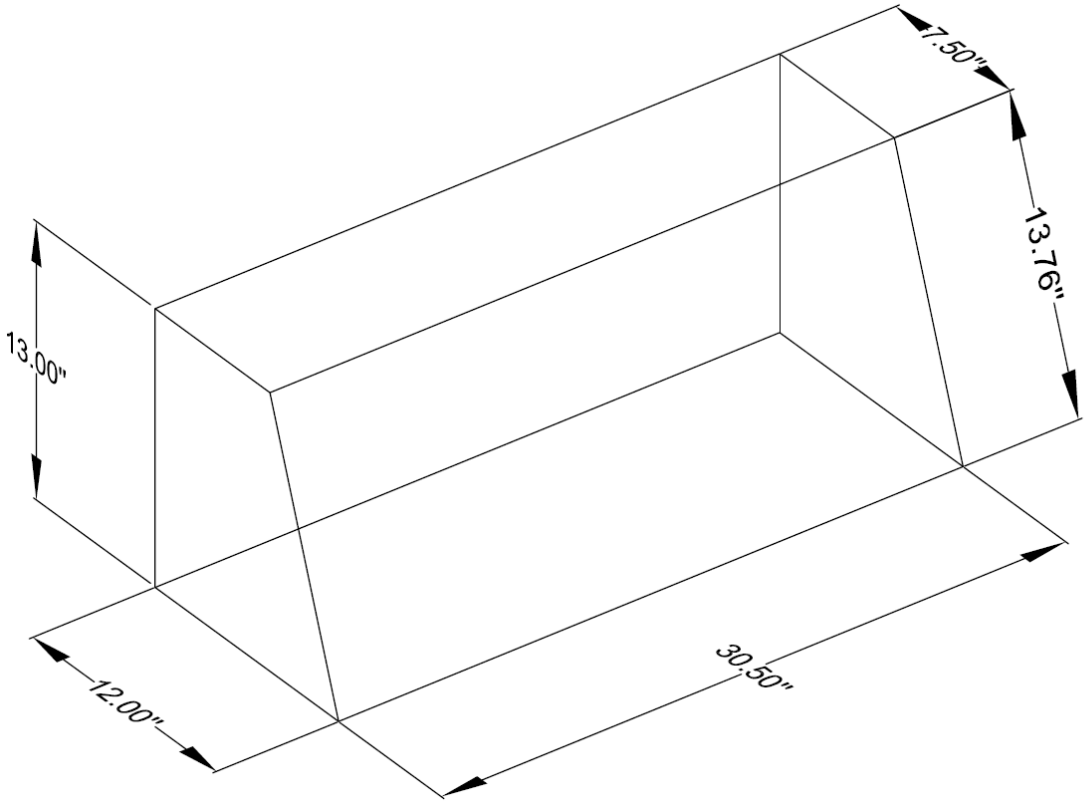

Figure 19. 3D drawing of enclosure.

The inside of the enclosure was updated to improve the connectivity of the outputs along with making it easier to identify where the various outputs were being powered. Figures 20 and 21 on the following page show the before and after pictures of the inside of the enclosure.

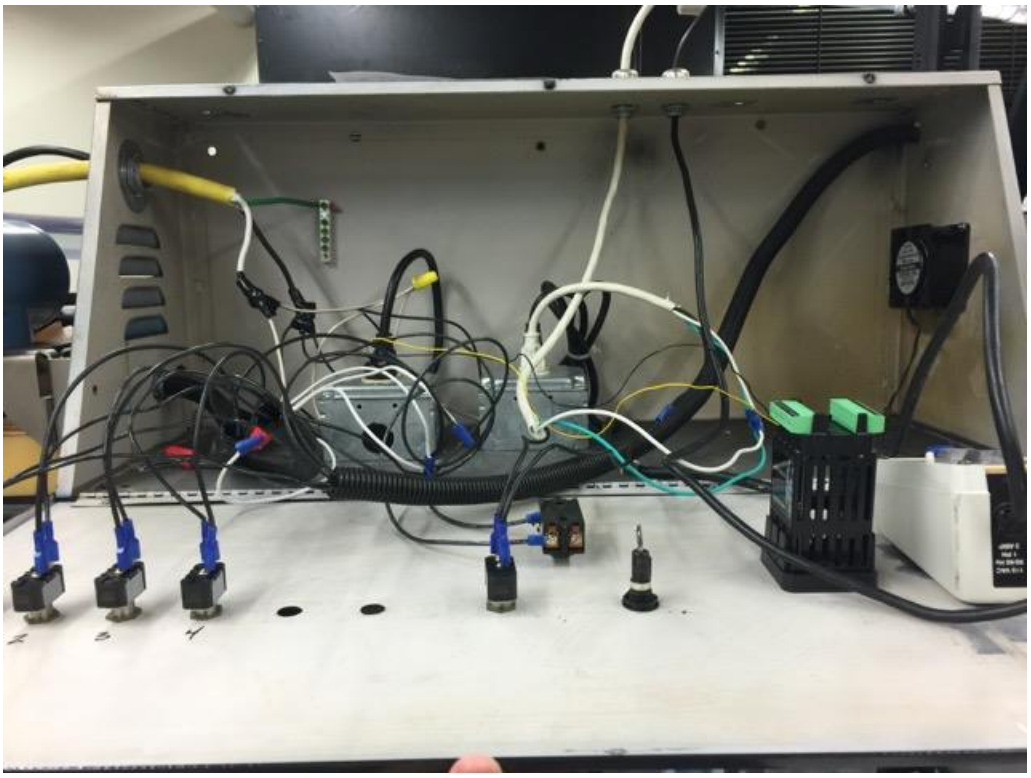

Figure 20. Before photo of the inside of the enclosure.

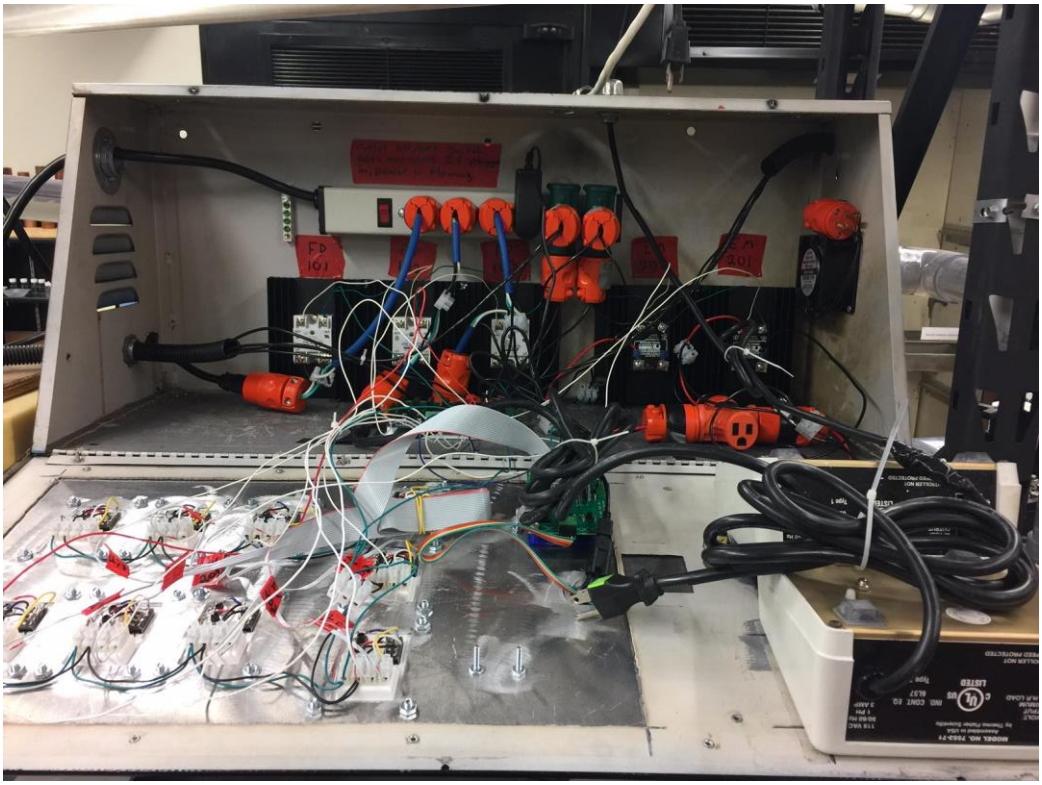

Figure 21. After photo of the inside of the enclosure.

How to Use It

This section is meant to be used as a tutorial reference on how to use the designed system. Topics covered include: Powering the system on and off, configuring the Orion Star pH meter, operation of the BK Precision power supply, connecting required USB components, viewing content of RPI HMI USB drive, GUI features, running the system in Automatic and or Manual Mode, and a brief explanation of the produced log files.

Powering the System On or Off

After plugging the main 20A power cord in, power to the RPI and relays is controlled via the 'MASTER POWER' switch located in the upper left corner of the front panel and circled below in Figure 22.

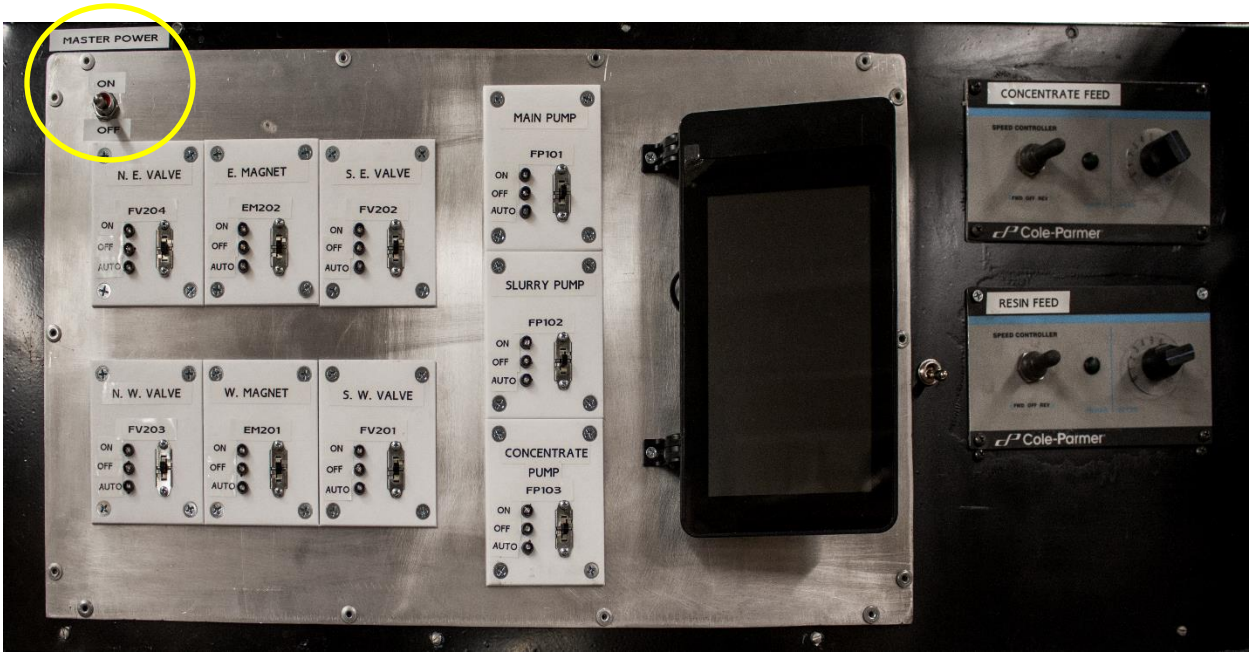

Figure 22. Unpowered Control Panel.

Once turned to the 'ON' position, the LEDs should light up and the RPI will begin its boot up process and scroll through assorted boot text. Once the finished booting, the RPI system will appear as shown in Figure 23 on the next page.

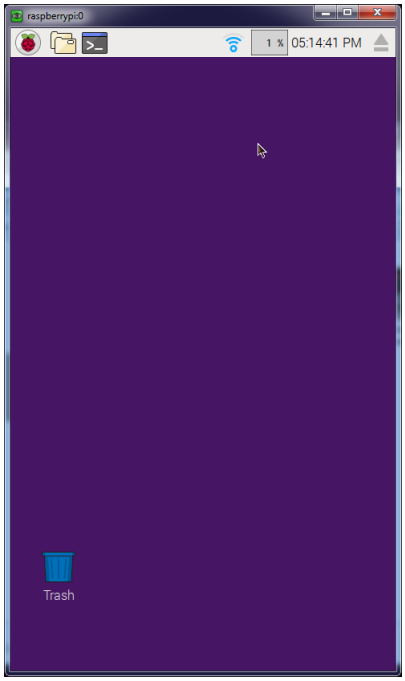

Figure 23. RPI default screen upon boot.

The RPI is now ready for further use.

To turn the RPI off, navigate to ''Shutdown…' via the main raspberry menu in the top left corner and then select 'Shutdown' as illustrated below in Figure 24. Reboot may also be chosen.

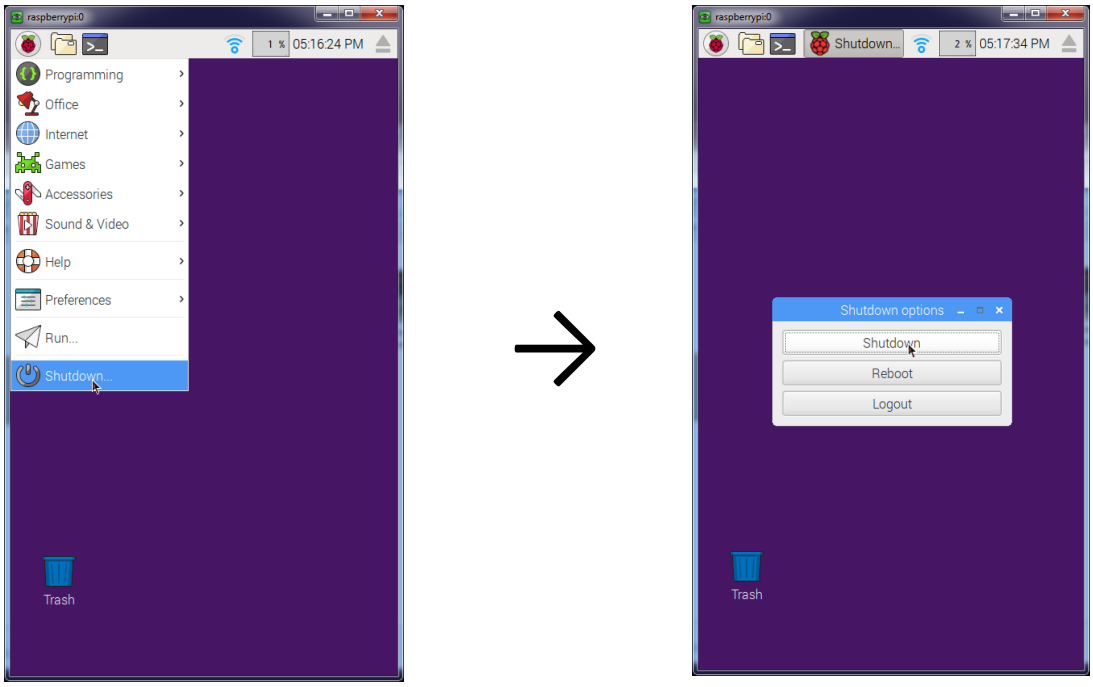

Figure 24. RPI Shutdown process.

Configuring the Orion Star A214 pH Meter

First, the associated Orion Star pH (OSpH) meter power supply must be plugged in. The meter should default into its standard 'live measure' mode as shown in Figure 25.

Figure 25. Standard starting screen of the Orion Star A214.

In order for proper communication to the RPI, navigate to the 'Mode and Settings' menu via pressing the Setup Button, then pressing the down arrow (log/print button) to select the correct menu and press f3. The only required setting here is "**Read Type: Continuous**". This menu process is shown below in Figure 26.

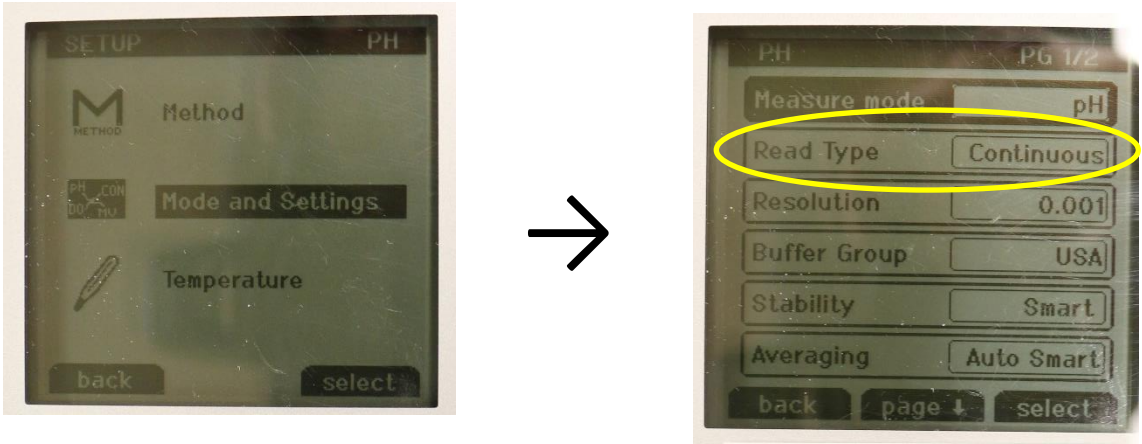

Figure 26. Configuring the Orion Star for Communication – Part 1.

After confirming the Read Type setting, press back (f1) until the default screen is displayed again. Then press Setup. Navigate to the 'Settings' menu and make sure the Export Data option is On. Selecting this by pressing f3 will lead to another menu that must be set to mirror the below figure. This means the following options must be selected "Printing: ON; Comm Setup: RS232; Data Format: PC (CSV); Comm Config: 9600,8,1,N".

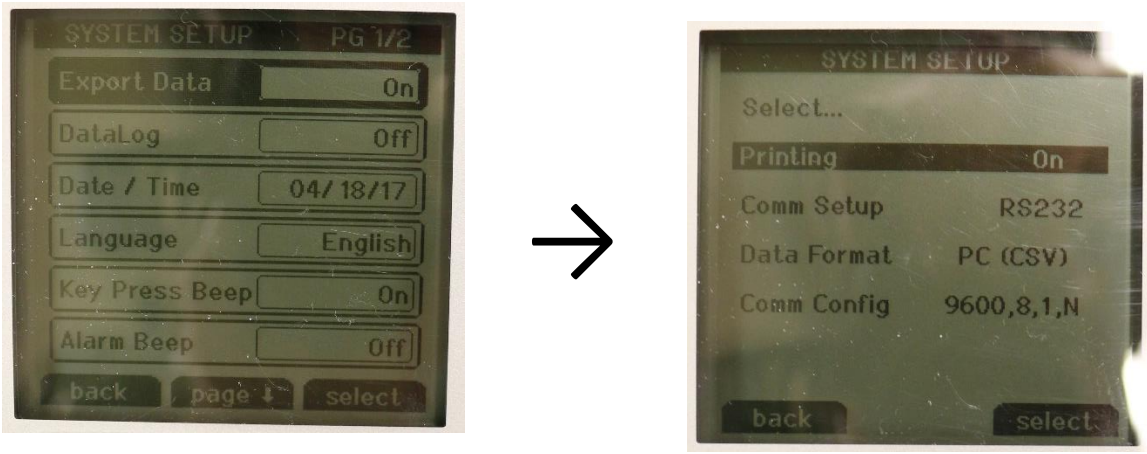

Figure 27. Configuring the Orion Star for Communication – Part 2.

The pH meter is now configured to correctly communicate with the RPI. Press the back button until the default 'live measure' screen is shown again for the unit to respond to RS232 commands.

Operating the BK Precision Power Supply

Operation of the BK Precision 1902 is very straight forward. The jacks on the back (red – positive terminal, black – negative terminal) are used to connect power to the DC relays via the colored spades. Once the unit is plugged into power and turned on it will perform an initialize sequence. After this phase is complete, the current knob is turned clockwise until the desired current is displayed in amps (A). The relays create an open circuit, so the desired current will disappear and display '0.0' as shown in Figure 28.

Figure 28. BK Precision Power Supply.

After a relay is energized, voltage will vary to supply the configured current, which will also now appear on the supply. The supply will switch from the C.V. light to the C.C. light on the power supply indicating Constant Current Operation. The current will be flowing through the electromagnets at this time.

Required USB Connections

The two devices that communicate to the RPI are the OSpH meter and a USB stick that contains the GUI code and also serves as storage for generated log files.

The OSpH must be connected via an RS232 to USB adapter cable shown in Figure 29. This cable plugs into the back of the meter and then is plugged into an open USB port on the RPI. This is shown in Figures 30 and 31.

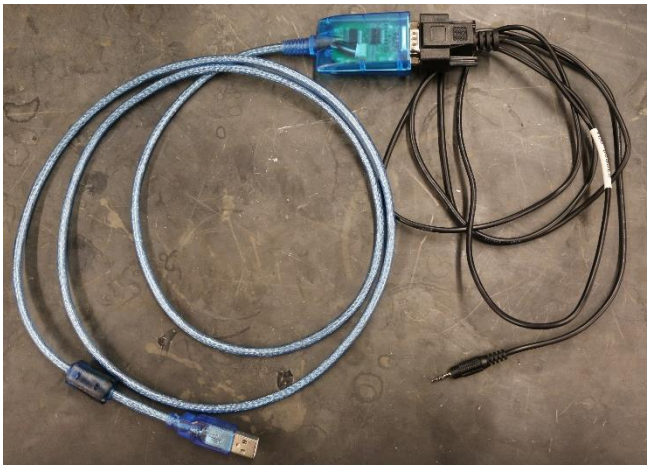

Figure 29. RS232 to USB cable.

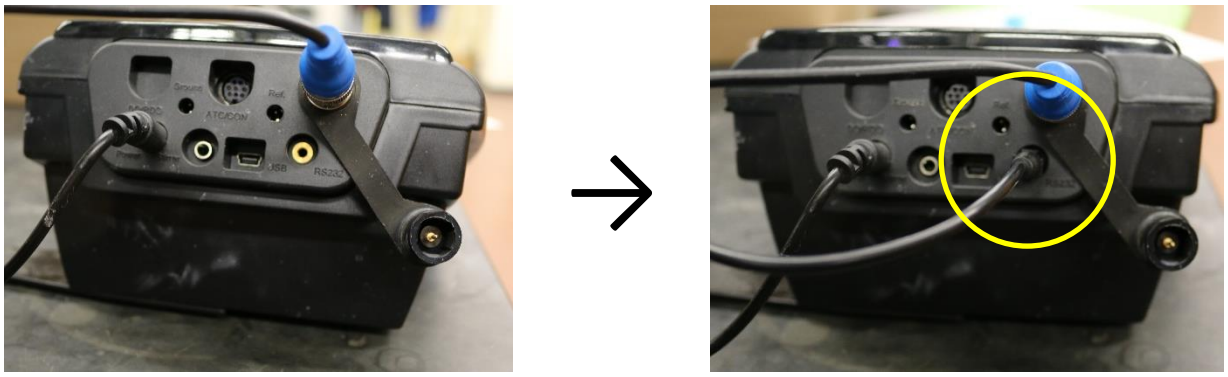

Figure 30. Connecting the USB to RS232 to the Orion Star pH meter.

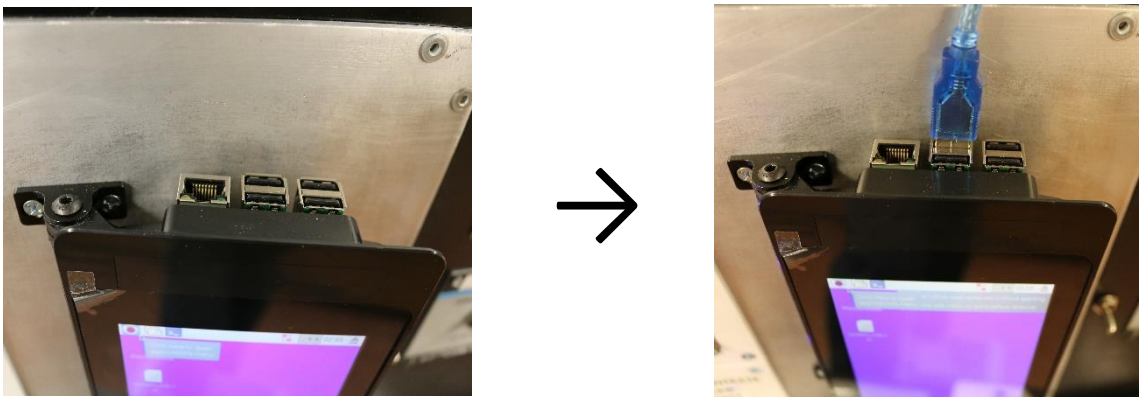

Figure 31. Connecting the USB to RS232 to the RPI.

The USB drive containing the code may now be connected to the RPI as shown in Figure 32.

Figure 32. Complete USB RPI connections.

If the pH meter readings are consistently ‘0.00’, safely eject the USB drive by pressing the eject icon in the upper right corner of the RPI and remove the pH meter USB connection, wait 10 seconds, then plug the meter in and then re-insert the USB drive.

The pH meter must be connected and recognized by the system before the USB drive in order for the RPI to correctly mount each device. There is unfortunately no way to know if this process has been successful without running the GUI and seeing if the meter values are read and displayed correctly.

RPI HMI USB Drive Contents

Upon inserting a USB drive into the RPI a popup such as the one shown below will present itself. If 'OK' is selected the contents of the drive will be displayed.

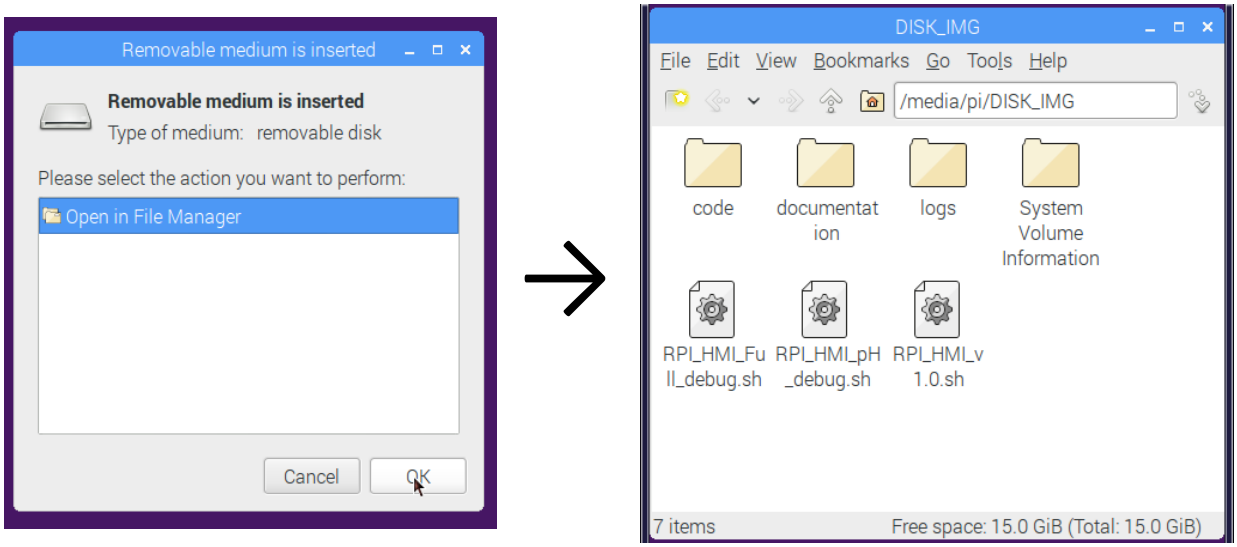

Figure 33. Opening the RPI USB Drive and Viewing Contents.

The USB drive will contain numerous folders and 3 shortcuts (.sh) files to run the RPI GUI. The details of each item are listed in Table 4. The two debug modes are included for trouble shooting the operation of the system if required.

Table 4. Explanation of RPI USB disk contents.

Item Name	Purpose
Code Folder	Contains all code to make program run. Any alteration to these files may result in non-functional / unexpected operation.
Documentation Folder	Contains tutorials about setting up the RPI and this tutorial along with a text file explaining the terms of licensing.
Logs Folder	This is were created log files are stored. Also contains a MATLAB file that will plot the data logs in a visually pleasing manner.
System Volume Folder	Default Drive folder – may not always be present.
RPI_HMI_Full_debug.sh	Shortcut to GUI where temperature and pH values are randomly generated. Log records all displayed values
RPI_HMI_pH_debug.sh	Shortcut to GUI where pH values are randomly generated but temperature is correctly collected from the thermistors and displayed. Log records all displayed values.
RPI_HMI_v1.0.sh	Shortcut to GUI where pH meter values and thermistor values are read and displayed. Meant to be the final working version of the code.

GUI Features

The features of the GUI are listed and described in Table 5 below and numbered in reference to the screenshots of the GUI as shown in Figure 34.

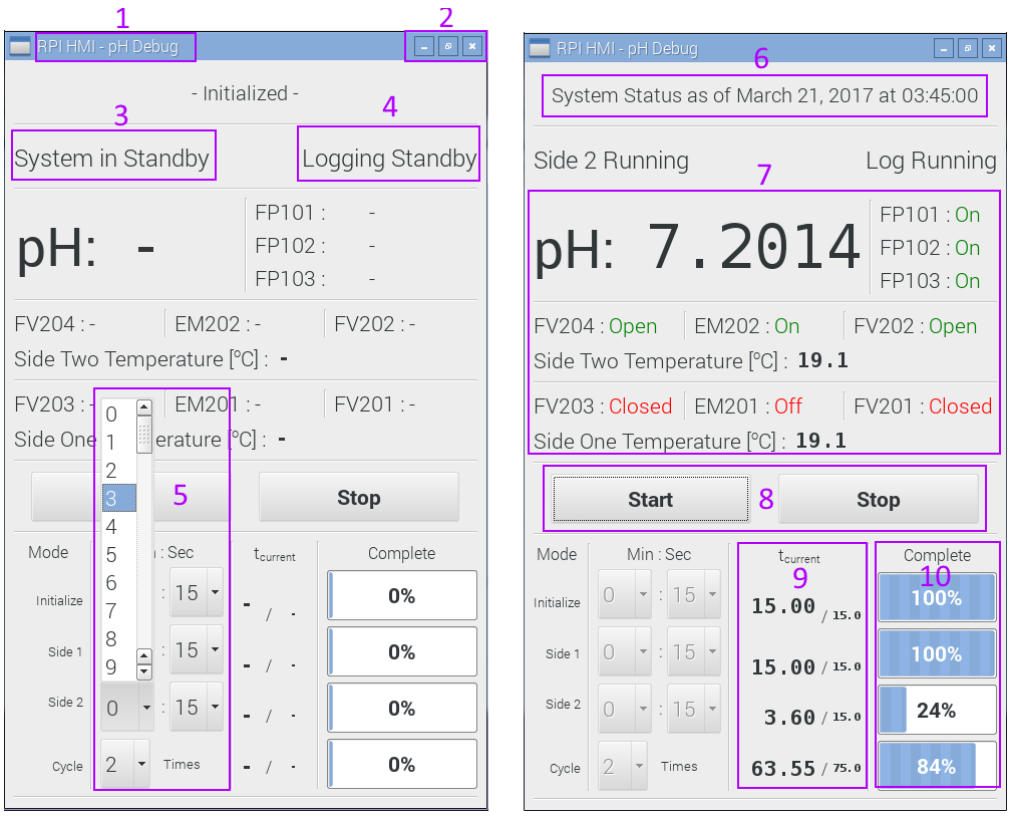

Figure 34. GUI Explained.

Table 5. Description of GUI Elements and Functions.

Number	Name	Function
1	Title	Displays version of GUI being executed.
2	Minimize, Maximize, Close Buttons	Used to resize or close the GUI.
3	Interpreted systems Status	Displays known system modes (i.e. Standby, Side 1 Run, Unknown, etc.)
4	Log Status	Displays current log status.
5	Mode Time Configuration	Used to alter mode run times and number of cycles to repeat.
6	Time Display	Displays current system time. Used to confirm GUI is updating.
7	Collected Data Display	Displays all collected data from devices. Updates every second.
8	Start and Stop Buttons	Used to start and stop the system and log process.
9	Current Mode Time Counters	Displays current cycle completion of each mode.
10	Percent Complete Progress Bars	Shows total percent complete of cycle.

Running the System

The System may be run as an automated cycle or a manually controlled process. Each of which is described in below.

As an Automated Process

In order to run the system as an automated process:

1. Open the desired version of RPI HMI GUI from the USB drive.
2. Select desired mode run times for each mode.
3. Select the number of cycles.
4. Press the start button.

The system will run through the user defined sequence and then enter standby mode. This standby mode will continuously log system information until the stop button is pressed.

As a Manually Controlled Process

If manual control is desired simply use the toggle switches associated with each device. This will work without any signal from the RPI however, the RPI is still required to be powered.

If logging is desired, open the v1.0 GUI and press start. While automated signals will still be sent to the devices, manual control overrides these signals.

Log Files Explained

The log files produced are stored in the /logs folder located on the RPI HMI USB drive and are named according to the date and time of the first entry, followed by the type of GUI ran (i.e. pH debug, full debug, V1.0). These files are .csv (comma separated value) type and may be opened in Excel or easily manipulated in MATLAB or similar data handling program. The first row is used as a column header followed by individual data entries approximately 1 second apart. A log opened in a standard text editor is shown below in Figure 35.

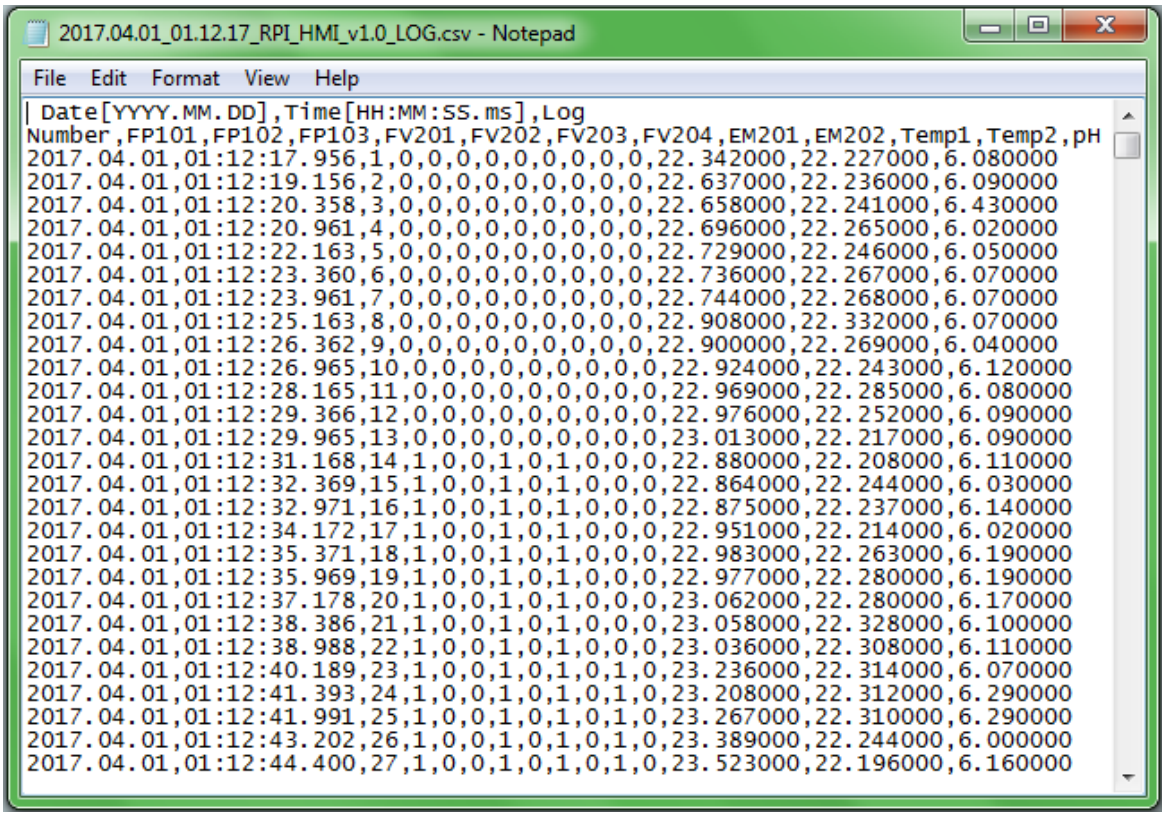

Figure 35. Data Log Example.

A MATLAB script has been included inside the /logs folder that was used to generate the figures presented in the Results section of this report. The code should plot any log file with no problem so long as it is properly identified as the log_file_name variable in line 4 of the code.

Test Results

In order to test how well the new automation and data collection system worked, two tests were run. The first test was an automated system run and the second was a manually controlled run. As per project goals set earlier, each test had to correctly record the system status, magnet temperatures, and pH data from the Orion Star meter. Correctly recorded data was defined as accurately reflecting what the system was actually doing. The collected data from the automated run is plotted below in Figure 36.

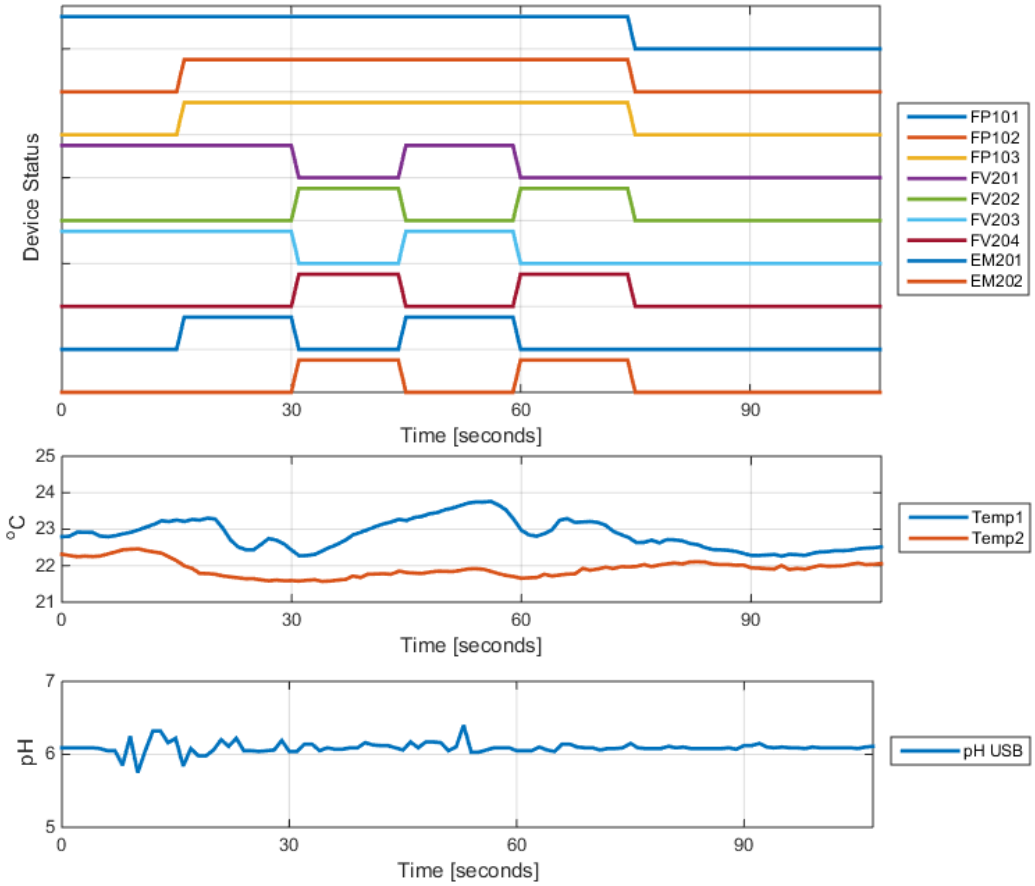

Figure 36. Data from automated system run.

As seen in from Figure 36, the log shows the system ran a predefined initialize mode for 15 seconds (FP101, FV201, FV203 ON), a side 1 run mode for 15 seconds (FP101, FP102, FP103, FV201, FV203, EM201 ON), followed by a side 2 run mode (FP101, FP102, FP103, FV202, FV204, EM204 ON), repeated the two run modes, and then went into standby (ALL off). This was the exact set of events programmed to occur and all physically observed outputs behaved as expected. Temperature data appears realistic and is believed to be within the ±1 °C range previously observed. Recorded pH data matched what was being displayed on the physical meter during the test.

The manual control test was decided to be as follows: Start logging, system standby for 10 seconds, (FP101, FV201, FV203, EM201 ON) until temperature is reaches ~28 °C, then EM201 will be turned off while pump and valves remain ON until temperature returns to approximate initial state, then standby (ALL OFF) for 10 seconds. The data log collected from this test is shown in Figure 37 below.

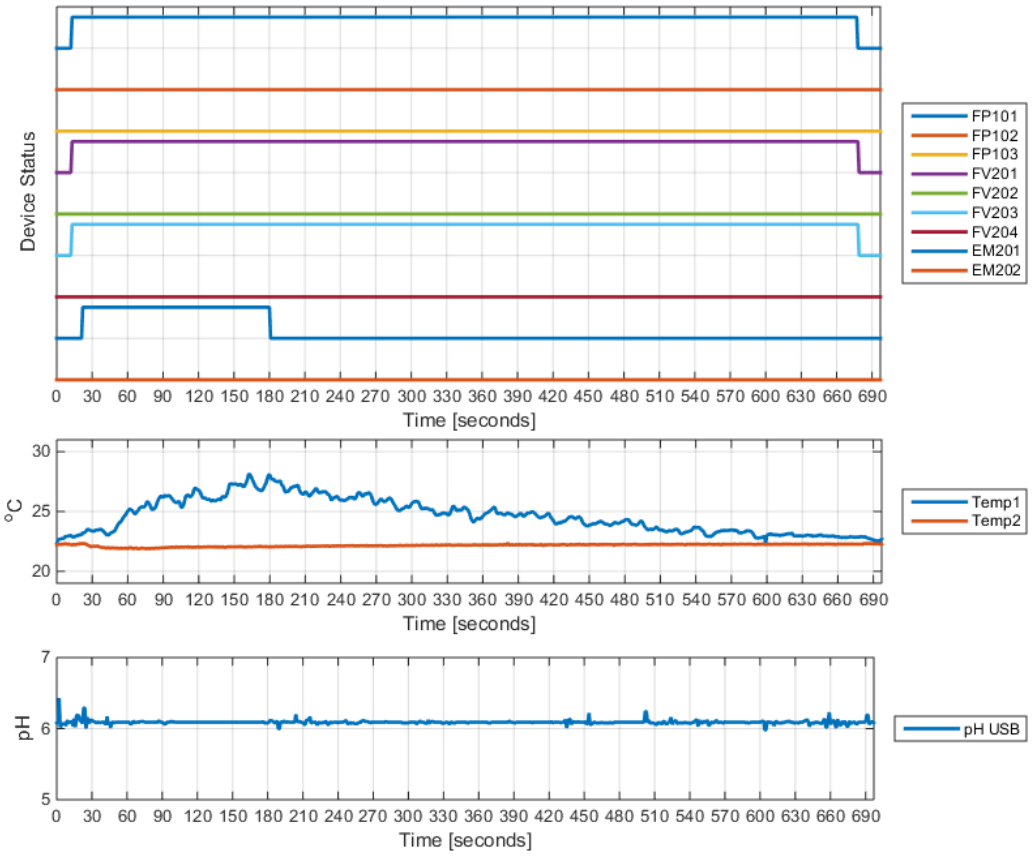

Figure 37. Data from manually controlled test.

Compared to the automatic test, the manual test is roughly 6 times as long. However, it can be seen that the system status followed exactly what the test process was defined to be. Logged pH remained essentially constant – this mirrored what the physical meter was displaying. The Temp1 data line corresponds to the temperature of EM201 and shows that powering the magnet for about a minute and a half caused the temperature to increase by roughly 5 °C. It then took a little over 8 minutes to return to the initial temperature. This much longer cool down time is due to the internal heating the magnet taking much longer to dissipate.

Conclusion – How Well it Works

The two tests conducted showed that the Raspberry Pi HMI and control system can operate as an automated system or be manually controlled and produce a data log that accurately reflects what is happening with the overall system operation, pH level, and magnet temperatures. This log is easily accessible to the user and contains clearly labeled data groupings.

The produced GUI is straight forward to use and provides an easy way to repeatedly change desired mode run times and view real-time system data.

The relays correctly energize and de-energize the specific outputs when desired and are adequately rated and heatsinked to ensure safe operation.

All of the components used for the system were able to fit in a pre-existing enclosure near the system for easy accessibility by whoever is controlling the process.

Total cost of one system was found to be $788.63, which is below the self-imposed limit of $1000 per system. All licensing fees were avoided.

Essentially the Raspberry Pi Human Machine Interface and Control System for an Electromagnet Water Filter performs exactly as designed and expected.

While our contribution to the project was as designed, it should be noted that the actual process being controlled could not operate as initially intended. The valves of the system being controlled were able to be energized through the operation of the relays, but then remained open regardless of the signal being sent. The normally closed valves required a specific pressure differential to operate, which was not present in the physically built piping system. This design flaw was outside of our project scope as the valves were already installed in the system pre-March 2016 and were reported to be ‘fully functional’.

Appendix A: Cost of One System

Table 6. Cost of One System.

Qty	Unit Price [$]	Total Price [$]	Ordered Parts
		Raspberry pi	
1	9.95	9.95	8GB Class 10 SD/MicroSD Memory Card – SD Adapter Included
1	39.95	39.95	Raspberry Pi 3 – Model B – ARMv8 with 1G RAM
1	7.95	7.95	5V 2.4A Switching Power Supply w/ 20AWG 6' MicroUSB Cable
1	14.95	14.95	USB/Serial Converter – FT232RL
1	24.95	24.95	SmartiPi Touch – Stand for Raspberry Pi 7" Touchscreen Display
1	79.95	79.95	Pi Foundation Display – 7" Touchscreen Display for Raspberry Pi
1	7.95	7.95	Premium Female/Male 'Extension' Jumper Wires – 40 x 12" (300mm)
	Sub Total:	185.65	
		Manual Control And installation costs	
10	0.022	0.22	Resistor; Carbon Film; Res 100 Ohms; Pwr-Rtg 0.25 W; Tol 5%; Axial
10	0.022	0.22	Resistor; Carbon Film; Res 680 Ohms; Pwr-Rtg 0.25 W; Tol 5%; Axial
10	1.1	11	Conn; Term Strip; Euro; Low Profile; 4; 8mm; DblRow; 20-12AWG (UL); 20A; 300V
1	5.28	5.28	Chanzon 100pcs (10 colors x 10pcs) 3mm Light Emitting Diode LED Lamp Assorted Kit
1	5.32	5.32	uxcell Metal Nut LED Mounting Holder Panel w 3mm 50 Pieces Black
1	15.95	15.95	Hook-up Wire Spool Set – 22AWG Solid Core – 6 x 25 ft
10	0.98	9.8	Slide Switches DP 3POS 125VDC .5A
1	34.99	34.99	Extension Cord 12/3 25FT
1	22.99	22.99	Cord Extn 16/3 SJOW 50'
3	8.99	26.97	Connecter 3-Wire Orange
4	5.99	23.96	Plug 3-Wire 15A Orange
1	4.99	4.99	Tapcube HD GRND Orange Bulk
	Sub Total:	161.69	
		PCB	
1	75.05	75.05	PCB Express (results in 3 boards)
1	3.06	3.06	5 Position Wire to Board Terminal Block
1	1.62	1.62	3 Position Wire to Board Terminal Block
2	1.27	2.54	IC DIP SOCKET 18POS
1	1.68	1.68	IC DIP SOCKET 28POS
1	0.81	0.81	10 Position Header Connector
1	3.18	3.18	20 Positions Header Connector
1	3.1	3.1	20 Position Cable Assembly
10	0.022	0.22	Resistor; Carbon Film; Res 10 Kilohms; Pwr-Rtg 0.25 W; Tol 5%; Axial
1	9.95	9.95	ADS1015 12-Bit ADC – 4 Channel with Programmable Gain Amplifier
1	2.95	2.95	MCP23017 – i2c 16 input/output port expander
2	1.95	3.9	MCP23008 – i2c 8 input/output port expander

2	0.37	0.74	Fairchild Semiconductor MOSFET 2N7000
2	4	8	10K Precision Epoxy Thermistor – 3950 NTC
	Sub Total:	116.8	
Relays			
1	27.99	27.99	Sainsmart 8 Channel 5V Solid State Relay Module Board.OMRON SSR 4
3	24	72	SSRL240DC10 – 240VAC/10A Solid State Relay
5	22.5	112.5	Finned Heat Sink
2	56	112	SSRDC100VDC20 - 100VDC/20A Solid State Relay
	Sub Total:	324.49	
	Grand Total:	788.63	

Appendix B: Expander to Relay Table

Table 7. Expander to Relay Table.

Expander Designation	Model	Port	Specified For	P&ID #
Output A	MCP23008	0	Valve 1	FV201
Output A	MCP23008	1	Valve 2	FV202
Output A	MCP23008	2	Valve 3	FV203
Output A	MCP23008	3	Valve 4	FV204
Output A	MCP23008	4		
Output A	MCP23008	5		
Output A	MCP23008	6		
Output A	MCP23008	7		
Output B	MCP23008	0	Pump 1	FP101
Output B	MCP23008	1	Pump 2	FP102
Output B	MCP23008	2	Pump 3	FP103
Output B	MCP23008	3		
Output B	MCP23008	4	Magnet 1	EM201
Output B	MCP23008	5	Magnet 2	EM202
Output B	MCP23008	6		
Output B	MCP23008	7		
Input	MCP23017	A0	Valve 0	FV201
Input	MCP23017	A1	Valve 1	FV202
Input	MCP23017	A2	Valve 2	FV203
Input	MCP23017	A3	Valve 3	FV204
Input	MCP23017	A4		
Input	MCP23017	A5		
Input	MCP23017	A6		
Input	MCP23017	A7		
Input	MCP23017	B0	Pump 1	FP101
Input	MCP23017	B1	Pump 2	FP102
Input	MCP23017	B2	Pump 3	FP103
Input	MCP23017	B3		
Input	MCP23017	B4	Magnet 1	EM201
Input	MCP23017	B5	Magnet 2	EM202
Input	MCP23017	B6		
Input	MCP23017	B7		

Appendix C: Thermistor Accuracy Test Results

As shown below in Table 8, both thermistors were tested against a TM902C thermometer at four temperature levels. All levels except 'Hot' were found to be within the ±1 °C range. However, since the thermistor temperature seemed to be skewing slightly higher than the reference temperature, and the high temperature zone represents potentially harmful operating conditions, this seemingly non-acceptable result was deemed acceptable.

Table 8. Thermistor Testing Results.

Device	Cold [°C]	Room [°C]	Hand [°C]	Hot [°C]
TM902C	1	23	33	42
TH201	0.5	23.3	33.2	43.5
TH202	0.6	23.5	33.3	43.6

Appendix D: Simplified Code Flow Diagrams

Below is a flow chart describing the overall actions taken by the GUI. This essentially involves loading and entering a GUI loop that waits to respond to user inputs. The logging thread and cycle thread are simplified on the next page.

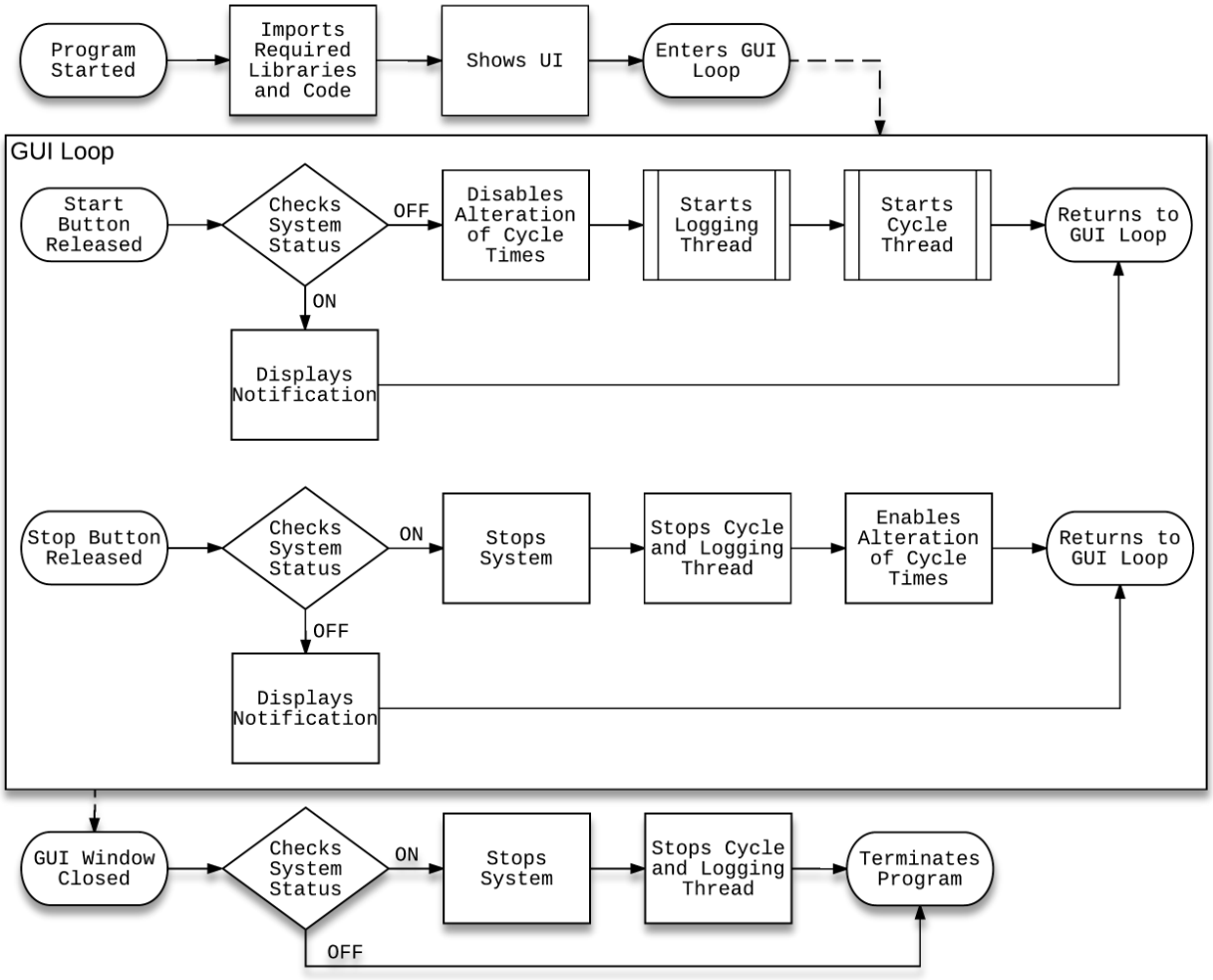

Figure 38. Simplified Code Flow

Figure 39 describes what takes place in the logging thread created by the code in order to create the data logs. The 'Log Loop' starts at the 'Sets Timer' position.

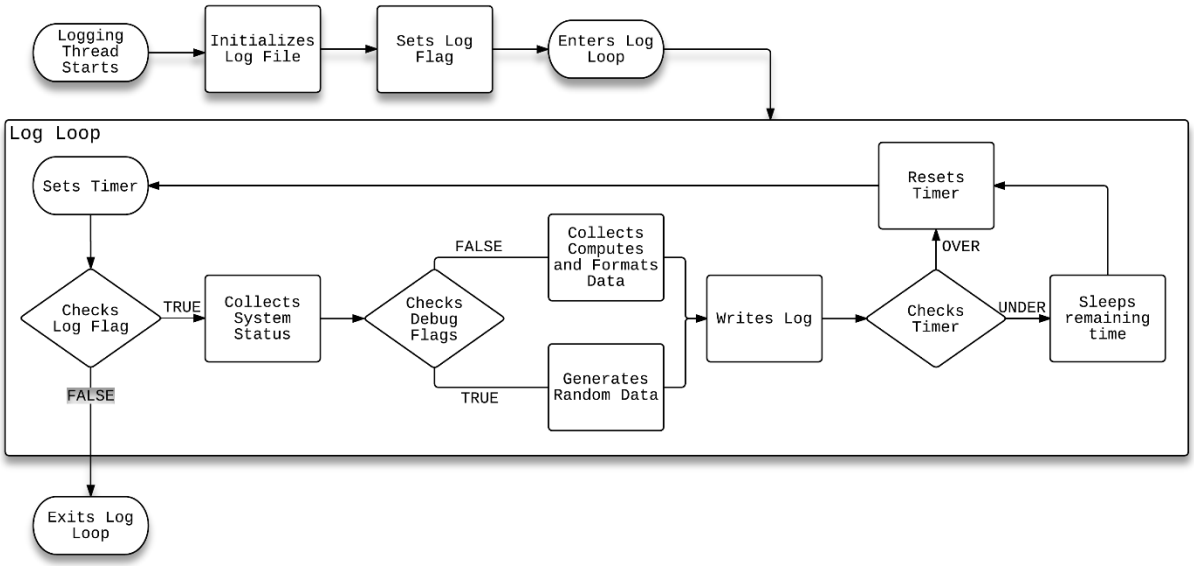

Figure 39. Simplified Logging Thread

Figure 40 shows how the cycle thread counts through each mode in a cycle. The 'Cycle Loop' begins with 'Checks Cycle Counter'. A 'runMode' flow diagram is shown on the next page.

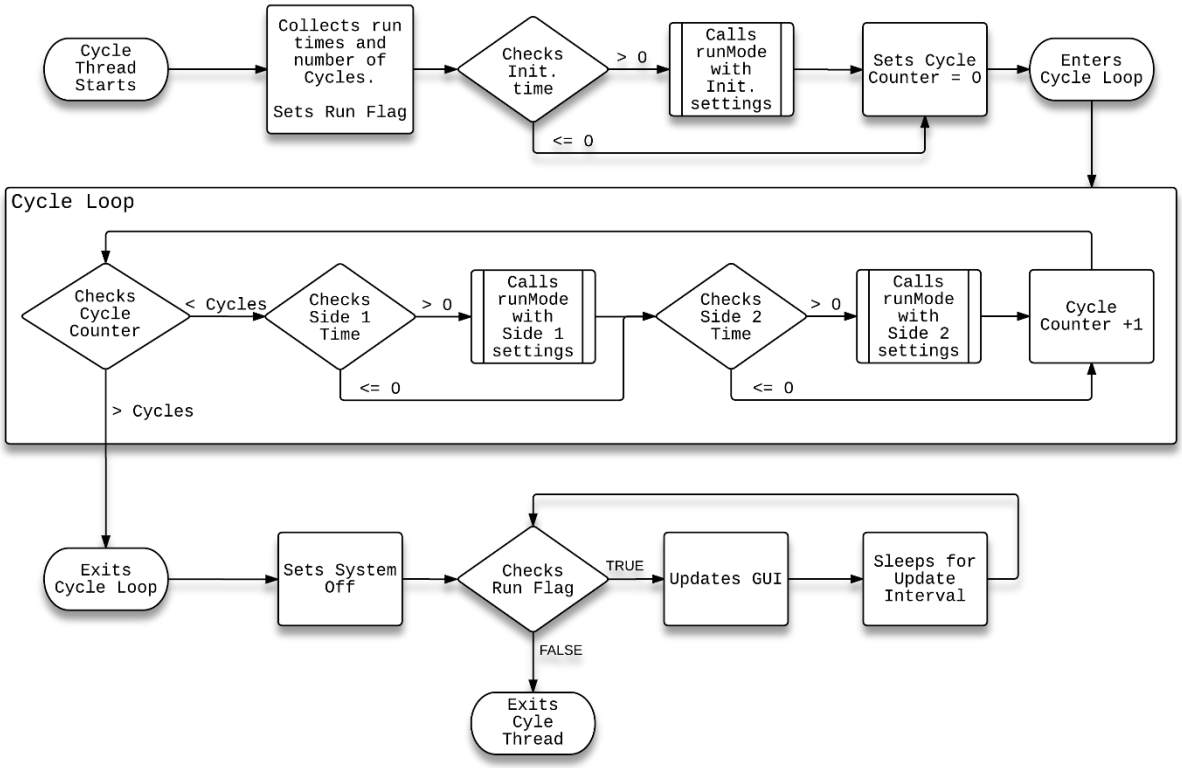

Figure 40. Simplified Cycle Thread.

The ‘runMode’ function, which does the actual timing of each mode, is simplified below. The ‘Timing Loop’ begins at ‘Checks Waited’.

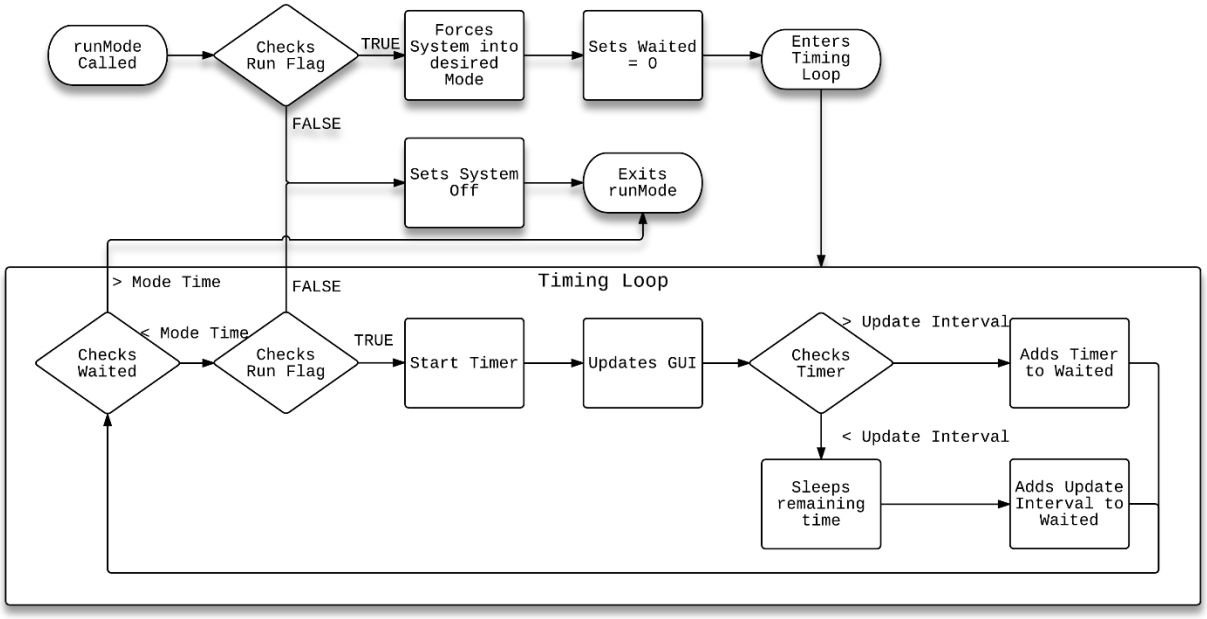

Figure 41. Simplified ‘runMode’ Function.

Appendix E: Project Related Schematics

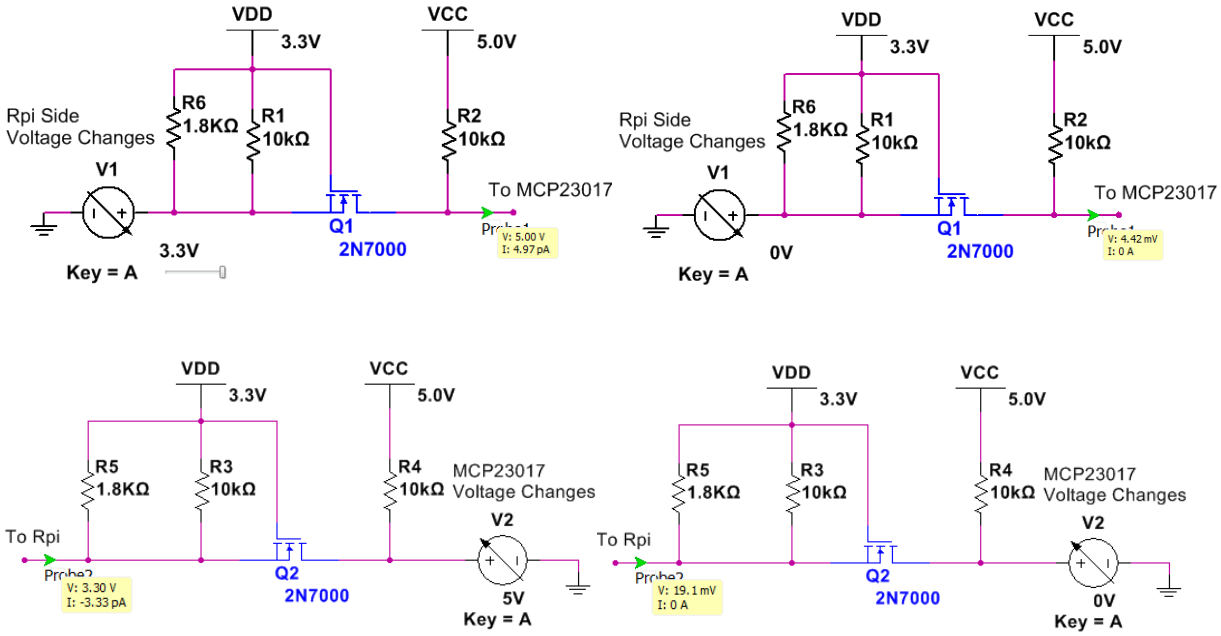

Figure 42. Simulation Results of MOSFET Level Shifters.

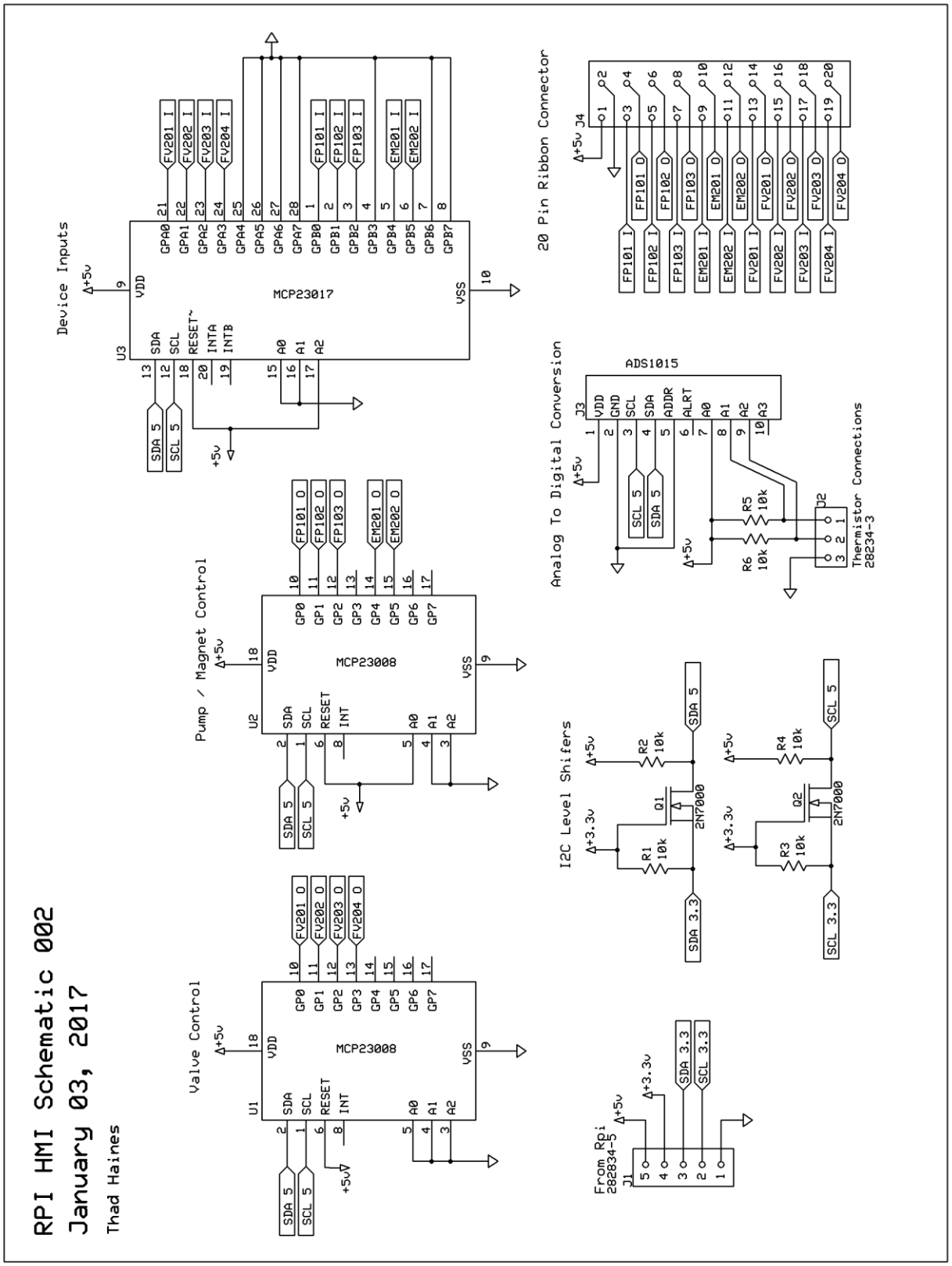

Figure 43. Express SCH Schematic.

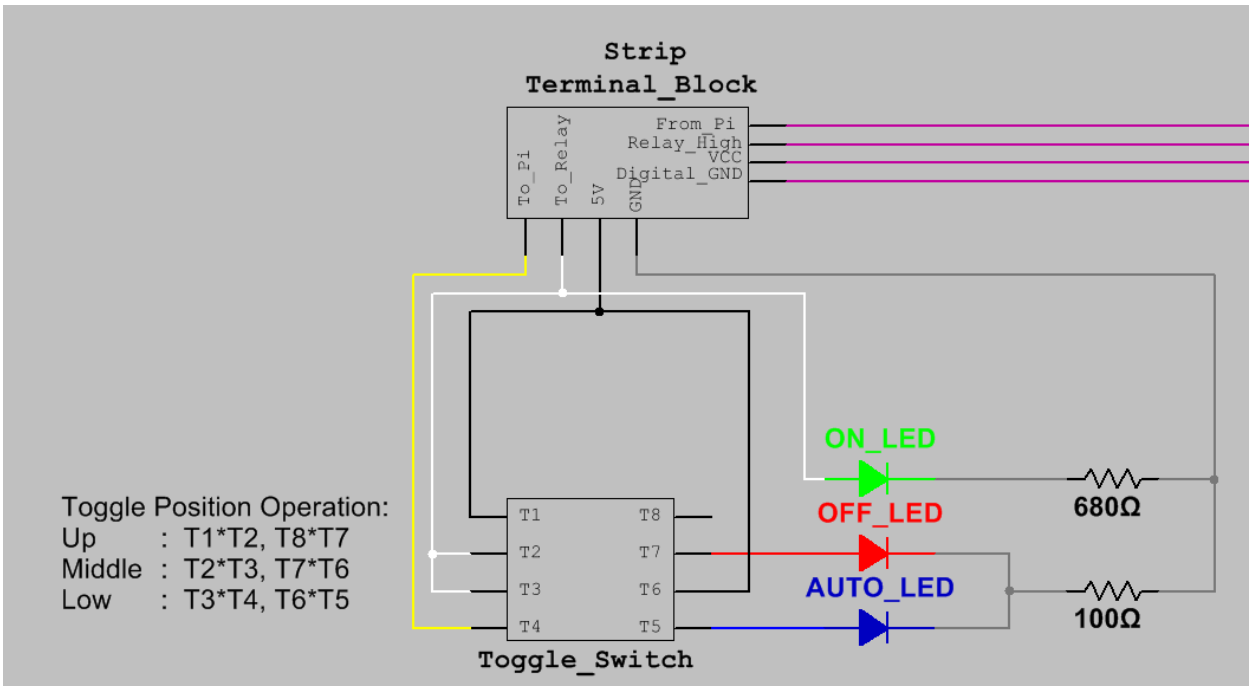

Figure 44. Wiring Diagram for Switch Plates.