

Senior Design II

Water Analogy



UNIVERSITY OF
CENTRAL FLORIDA

UCF Senior Design Team Fall 2017-Spring 2018
ECE Department

Inalvis Alvarez-Fernandez
Justin Berte
Richard Bielski
Zachary Freer



Table of Contents

Executive Summary	1
Product Description	2
Motivation	2
Goals and Objectives	2
Requirements and Specifications	2
Overall Design	2
Hardware Functions	3
Software Functions	4
House of Quality	5
Design Constraints and Standards	6
Standards	6
Lead Soldering Safety Standards	6
Soldering Iron Safety	6
Solder, Flux and Cleaners	6
Personal Exposure	6
Waste Management	7
IPC PCB Standards	7
Power Supply Standards	7
RoHS	9
Wireless Network Standards	9
Wired Equivalent Privacy	9
Wireless Protected Access	10
Wireless Protected Access 2	10
802.11n	10
2.4 GHz	10
Python Language Standards	11
Code Layout	11
Indentation	11
Tabs Versus Spaces	12
Maximum Line Length	12
Line Breaks	12
Blank Lines	12
Source File Encoding	12
Imports	12
	137

Module Level Names	13
String Quotation Marks	13
Whitespace	13
Trailing Commas	13
Comments	13
Block	14
Inline	14
Naming Conventions	14
Modules	14
Classes	14
Type Variables	14
Exceptions	14
Functions	14
Global and Instance Variables	14
Constants	14
C Language Standards	15
Indentation	15
Line Continuation	15
Braces	15
Limits	15
Statements Per Line	16
Maximum Line Length	16
Line Breaks	16
Includes	16
Jump Statements	16
Variable Declarations	16
One variable per declaration	17
Declare at Beginning of Function	17
Comments	17
Block	17
Single Line	17
Inline	17
Naming Conventions	17
Header Files	18
Class Names	18
Function Names	18
Constant Names	18

Variable Names	18
Parameter Names	18
Errors	18
Shells Scripting Standards	18
File Extensions	18
File Header	19
Error Handling	19
Syntax	19
Comments	19
Indentation	19
Line Length	19
Constraints	19
Time Constraints	19
Economic Constraints	20
Macroeconomic Constraints	20
Microeconomic Constraints	20
Social Constraints	20
Health and Safety Constraints	21
Hardware Constraints	21
Water Proof	21
Mechanical Constraints	22
Research and Background Information	24
Similar Projects	24
Computer and Controller Hardware Considerations	24
Raspberry Pi	24
ATmega328P	25
ATSAMD21G18	25
MSP430G2	25
ATmega2560	26
Conclusion	27
Wired Communication Technologies	27
Parallel vs. Serial	27
UART	28
SPI	28
I2C	29
Wireless Communication Technologies	29
Ad Hoc Network on the Raspberry Pi 3	29
	139

Remote Display of the System	30
Chrome Remote Desktop	30
Teamviewer	31
Virtual Network Computing (VNC)	31
Remote Desktop Protocol	31
Conclusion	32
Wireless Module Considerations	32
Built-In vs. USB Dongle	32
Conclusion	33
Printed Circuit Board	33
PCB Composition	34
Substrate Layer	34
Copper Layer	34
Solder Mask Layer	35
Silkscreen Layer	35
PCB Terminology	36
Design Recommendations For Better Reliability	36
Valve Control Considerations	38
Motorized Ball Valve	38
Brushed DC Motor	38
Servo Motor	39
Stepper Motor	40
Unipolar Stepper Motor	40
Bipolar Stepper Motor	41
Conclusion	41
Motor Considerations	42
Bipolar Stepper Motor 1	42
Bipolar Stepper Motor 2	42
Bipolar Stepper Motor 3	43
Bipolar Stepper Motor 4	43
Conclusion	43
Driver Considerations	44
Stepper Motor Driver 1	45
Stepper Motor Driver 2	45
Stepper Motor Driver 3	45
Stepper Motor Driver 4	46
Stepper Motor Driver 5	46

Conclusion	46
Light Emitting Diodes Strip	48
Analog and Digital Strips	48
Analog LED Strips	48
Digital LED Strips	49
Chip Size	49
Flexibility	49
Water Proof	49
Rayonn FLB6	50
Adafruit LPD8806	50
ALITOVE WS2812B	50
Conclusion	50
Mechanical Components Considerations	51
Valves	51
Brass Ball Valve	51
True Union Ball Valve	52
Pump Consideration	52
Flow Meters	54
P3 P0550 Water Meter	54
Fill-Rite In-Line Digital Meter	54
Pressure Sensing	55
Pressure Gauges	56
Analog vs. Digital Pressure Gauge	56
Ashcroft DG25 Digital Pressure Gauge	57
Dwyer Instruments DPGA-04	57
Noshok 40-911-800 PSI NPT	57
Winters PEM Steel Dual Scale Economy Pressure Gauge	57
Conclusion	58
Pressure Transducers	58
Honeywell 19mm Series	59
Honeywell MLH Series	60
Eyourlife Universal 30PSI	60
Touchscreen	60
Qualities	61
Resistive vs. Capacitive	61
Touch Screen Comparison	62
Software Tools	64

Communication	64
Slack	64
Asana	65
Development	65
Vim	65
Qt Designer	66
Arduino IDE	66
SolidWorks	66
Eagle	67
Hostapd	67
RealVNC Connect and Viewer	67
Documentation and Storage	68
Google Drive	68
GitHub	68
Draw.io	69
Design	70
Hardware Design	70
Hardware Block Diagram	70
Hardware Design Overview	71
Mechanical Overview	75
Valve	75
Pipes	76
Motor	77
Pressure Sensor	77
Power PCB Design	77
Signal PCB Design	81
PCBs Bill of Materials	84
Microcontroller	88
Raspberry Pi 3	88
Light-Emitting Diode	89
Software Design	90
Software Block Diagram	90
Graphical User Interface	91
Raspberry Pi and Microcontroller Interactions	92
Microcontroller Control	93
TkInter	94
Qt	94
	142

PyQt vs. PySide	94
Qt Designer	95
Conclusion	95
Machine Learning	95
Design Summary	96
Prototyping	98
Printed Circuit Board	99
Power PCB Design	99
Signal PCB Design	100
Power and Signal PCB Fabrication	100
Power and Signal PCB Population	100
Software	100
GUI	100
Atmega2560 Stepper Motor Functionality	101
Atmega2560 Transducer Functionality	102
Prototype Expectations	103
Potential Hardware Issues	103
Potential Software Issues	103
Testing	104
Hardware Testing	104
Microcontroller Testing	104
Code Implementation	104
Communication: RaspberryPi 3-ATmega2560	104
ICSP Programming	105
Outputting from the ATmega2560	106
Pressure Transducers	106
Motor Testing	107
Motor Driver Testing with Function Generator	107
Motor Driver with Microcontroller	108
Wireless Module Testing	109
Raspberry Pi and Microcontroller Interaction	110
LED Matrix Testing	110
Software Testing	111
Software Testing Overview	111
GUI	112
Machine Learning Testing	112
	143

I2C	113
Analog Input Testing	113
Simulated Testing	114
Integration Testing	114
Final Design	116
Final Project Functions	116
Usage Guide	118
Administrative Content	120
Project Milestones	120
Critical Path	121
Budget and Finance	122
Conclusion	125
References	127
Appendix A- Alitove LED Strip Datasheet	
Appendix B- Sotera System FR1118-P10 User's Manual	

Figures Index

1	Figure 2.4-1: House of Quality	5
2	Figure 4.2.5-1: ATMEGA2560 16AU	26
3	Figure 4.6.1-1: PCB Layers Layout	34
4	Figure 4.8.5-1: Nema 23 CNC Stepper Motor	44
5	Figure 4.9.6-1: SMAKN® TB6600 Motor Driver	47
6	Figure 4.10.5-1: ALITOVE WS2812B Individually Addressable LED Strip	51
7	Figure 4.11.1.2-1: True Union Ball Valve	52
8	Figure 4.11.2-1: Aqua Pulse Koi Pond Pump	54
9	Figure 4.11.3.2-1: Fill-Rite Digital Meter	55
10	Figure 4.11.4.2.3-1: Eyourlife Universal 30PSI Transducer	60
11	Figure 4.13.2-1: Raspberry Pi Touch Display	64
12	Figure 5.1.1-1: Hardware Block Diagram	71
13	Figure 5.1.2-1: Overall System Design	72
14	Figure 5.1.2-2: Diode Electrical Representation	73
15	Figure 5.1.2-3: BJT Electrical Representation	74
16	Figure 5.1.2-4: Mosfet Electrical Representation	75
17	Figure 5.1.3.1: Motor and Valve Gear Design	76
18	Figure 5.1.5-1. 5 Volt Power PCB Rail Schematic	78
19	Figure 5.1.5-2. TPS40305 Efficiency vs Output Current	79
20	Figure 5.1.5-5. Power PCB Schematic	81
21	Figure 5.1.6-1: I ² C Signal	82
22	Figure 5.1.6-1. Signal PCB Schematic	83
23	Figure 5.1.10-1: LED Cascade Method	89

24	Figure 5.1.10-2: Pin Configuration	89
25	Figure 5.2-1: Software Block Diagram	91
26	Figure 6-1: Signal/Power Distribution Prototype	99
27	Figure 6.2.1-1: GUI Prototype	101
28	Figure 7.1.1.3-1: ISCP Testing (Slave Left / Master Right)	105
29	Figure 7.1.3-1. Bipolar Stepper Motor Clockwise Driving Sequence	107
30	Figure 7.1.4-1: Motor Driver Testing	108
31	Figure 7.1.4.1-1: Motor Driver with Microcontroller Testing	109
32	Figure 8-1: Final System	116
33	Figure 8.1-1: Software Flowchart	117
34	Figure 8.2-1: Motor and Sensor Wiring	118
35	Figure 8.2-2: Signal PCB Wiring	119
36	Figure 8.2-1: Critical Path Diagram	122

Tables Index

1	Table 3.1.3-1: Classes of Technology Equipment	7
2	Table 3.1.3-2: Circuit Definitions	8
3	Table 4.7.5-1. Valve Control Options Characteristics	42
4	Table 4.8.5-1. Motor Consideration Characteristics	44
5	Table 4.9.6-1. Stepper Driver Consideration Characteristics	47
6	Table 4.12.2-1. Touch Screen Considerations	62
7	Table 5.1.5-1. 5 Volt Power PCB Rail Components	79
8	Table 5.1.7-1. Power PCB Bill of Materials	84
9	Table 5.1.7-2. Signal PCB Bill of Materials	85
10	Table 5.1.10-1: LED Pin Configuration	90
11	Table 7.1.2-1: Pressure to Voltage Relation	106
12	Table 7.2.2-1: Analog Input Measurements	114
13	Table 8.1-1: Milestones	120
14	Table 8.2-1: Overall System BOM	122

1. Executive Summary

The University of Central Florida requires graduating engineering students to create a capstone project. Our project consists of creating a portable board that demonstrates a mechanical representation of circuit components for students to have an easier visualization of circuit theory. The system is constructed with clear PVC pipes, motorized valves, gauges, flow meters, and an interactive display that allows students to set the circuit parameters. A pump supplies water from the lower retention tank to the main line that feeds the components. Valves are set up in each one of the component lines to control the state of the line by adjusting the rate at which the water flows, representing the current. These valves are controlled by using stepper motors, which open at specific angles to allow a controlled current set by the user through the interactive display. Measurements are made by the use of pressure transducers, which gives students/users a visual representation of the voltage (pressure) as well as the rate of current flow at the various lines. These measurements are correlated to component equations such as Ohm's Law to facilitate the student's understanding of circuit theory. Each component has a plaque explaining the component, its function, and relevant equations. LEDs are allocated through each one of the lines; they are programmed to display the intensity of water flow, showing an increase or decrease of current. The overall system is mounted to a frame on wheels to enable portability and set at a reasonable height to increase usability for the user/student. This allows the user/student to operate and watch the whole system function without any difficulty.

Due to our team's collaboration with Power & Energy Society (PES) IEEE the system is composed of two parts. PES is constructing the representation of components such as: KVL, KCL, resistor, inductor, and capacitor representations, while our team works on the MOSFET, BJT and diode configurations. The overall system gives the user free control over the components and their parameters.

2. Product Description

This section discusses the motivation behind the project's goals and objectives. It covers the various specifications and requirements of the overall project design. Hardware and software functions are discussed with specific values, components used, dimensions and capabilities. The House of Quality compares the power consumption, user friendliness, installation ease, cost and performance with relation to the latency, setup time size and pressure and their effects with one another.

2.1. Motivation

Circuit theory is a very challenging topic for those who are being introduced to it for the first time. Current, unlike water is not something that can be seen and students often struggle to understand how it flows, and affects circuit components. Since water flow is something that can be seen and is often used as analogy by many instructors when teaching circuit theory, our team decided to utilize this concept and apply it for students to truly visualize and understand electronic concepts. This is done by equating water flow to current flow, pressure to voltage, etc.

2.2. Goals and Objectives

This project's objective is to create a mechanical system for easier visualization of current flow by using water, to demonstrate circuit theory and components (resistors, inductors, capacitors, diodes, transistors, etc) responses. The project is created with the intention to be used for educational purposes and help new students understand and physically visualize how circuit components behave given certain conditions. Instructors using the product will have the option of controlling the system via wireless communication, thus improving the student/teacher classroom interaction.

2.3. Requirements and Specifications

Requirements and specifications are very important to maintain throughout the course of the project. There are hardware and software functions specific to the project; the following sections discusses those points as well the overall design dimensions.

2.3.1. Overall Design

When it came time to plan out the entire design, since there was a limited budget, optimizing the system for a little of a cost as possible was put at the forefront. However, great thought was put into ensuring that quality and

aesthetics were not sacrificed because ideally this project will be displayed for public use.

- Overall cost of the system should be kept to the minimum while also providing an aesthetically pleasing and functional product.
- Overall size should be approximately 3.5 inches deep, 5 feet wide, and 6 feet 4 inches tall.

2.3.2. Hardware Functions

The electrical and electronic components serve a variety of purpose. One major aspect of this was the power PCB which converted the AC power from the wall into DC power for the electronic components of the project. A buck converter was also utilized to adjust the voltage level so that our power PCB supplied 12V and 5V. Each electrical component receives its power from this board. The other major part of the electrical hardware was the signal PCB which served as a hub for the data processing and system control. An ATmega2560 microcontroller receive user input from the single board computer and processed that data so that it sent out the correct control signals to the appropriate stepper-motor drivers. The microcontroller on this PCB also controlled the corresponding LEDs for the selected component simulation. Finally, feedback from the pressure transducers was sent to this signal board for the microcontroller to process and transmit back to the single board computer to display to the user.

- Single board computer (Raspberry Pi 3).
- External 7 inch touch screen display to interface with the device allowing user control over components parameters such as beta values for BJT, voltages across different junctions, as well as voltages across the device as a whole.
- Trigger the following events using general purpose input and output pins:
 - Adjust the valves of the systems to reflect the user's input in the interface (Latency < 750 ms).
 - Trigger stepper motors using relays and adjusting the amount the motor turns to allow different amounts of water to flow representing set current.
 - Trigger start and stop of the system's pump using a relay
- Control light emitting diodes to accentuate the current flow throughout the system (Latency < 750 ms).
 - LEDs flash in the direction of current flow through the devices.
 - As the "voltage" (pressure) increases across the device, the LEDs will flash faster representing an increase in current flow through the various components. This speed represents a value set by the user predetermined by the programmer.
- Power PCB.

- Manage power distribution to the various components such as LEDs, stepper motors, Raspberry Pi, touchscreen, and pump.
- A seamless operation under any conditions. (Power Requirement < 75 W)

2.3.3. Software Functions

The ultimate goal to that was to be obtained in all the software development aspects of this project was to ensure intuitive, smooth, and obvious user interactions. This meant that the GUI was designed so that it was self-explanatory and aesthetically pleasing. This meant that the Raspberry Pi optimally transmitted the user input to the microcontroller which efficiently and effectively adjusted the valves and ran the simulation. And this meant that the LEDs were programmed as a visualizer to emphasize the electrical current flow analogy through the PVC and water. In order to set a standard to achieve these goals, brainstorming occurred to lay out a few requirements we aimed to meet.

- Clean user friendly graphical user interface. (Size Requirement < 250 MB)
- Ability to navigate quickly and seamlessly through menu options.
- Set component values with visual feedback and store them in memory.
- Provide a visual representation of the overall circuit for the user to set parameters and components to be used before starting the system.
- Allow the user to start and stop the system when desired. (Latency < 750 ms)
- Create safety standards so the user is not able to trigger all the components at once, limit each component parameter options. This will reduce the probability of errors in the system since all the parameters will be predetermined and calculated.
- Parameters for each component will be provided to the user by using a dropdown bar displaying all the options the user has.
- Provide users with information about the components in a clear and concise fashion (summarize important component information as well as display what the current flow through and pressure “voltage” across the system should be). This includes component related equations, stage of operation of the transistors, etc.
- Ability to control the system via a remote desktop.
- Setup an adhoc network connection to access the Raspberry Pi 3.
- Ability to mirror the graphical user interface and project it.

- Interact with the interface with minimum delay.

2.4. House of Quality

The requirements of the House of Quality were chosen based on the software and hardware requirements. Input power refers to the power necessary to feed the overall system given that is composed of many motors, LED strips, etc. Cost refers to the total cost of building the system, based on the required parts. Graphical User Interface (GUI) is the visual application which the user will interact with on the 7 inch touch screen. This interface must be as smooth as possible containing high resolution graphics to reflect when screen sharing, while being able to fit under the size constraint of 250 MB. Latency refers to the maximum time permitted for the physical system to reflect the user's action on the graphical user interface. This time constraint is 750 ms to allow adequate time to begin the process, but still optimum performance from the user's perspective. Setup time refers to the overall time the user would spend setting up the system in a new environment, it includes the screen sharing setup time as well as system turn on; it directly correlates to installation effort. Performance takes into to account the overall behavior of the system included but not limited to response time, user friendliness, and accuracy. See figure 2.4-1 for the various requirements and their effect with one another.

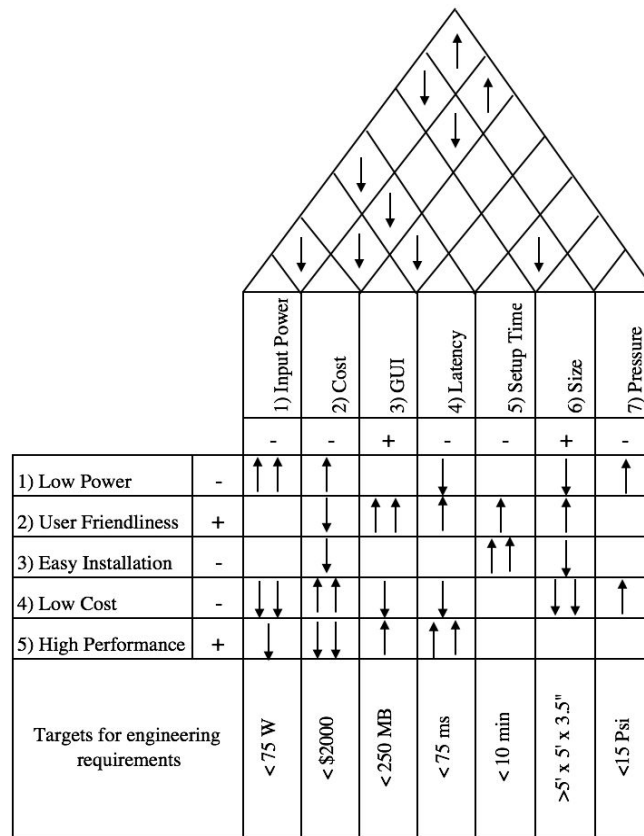


Figure 2.4-1: House of Quality

3.Design Constraints and Standards

This chapter discusses the various standards and constraints that have an impact on the design requirements. It identifies the various standards to follow regarding hardware and software development.

3.1. Standards

The standards covered in this section discusses lead soldering, PCB, power supply standards, and RoHS standards. Software standards include wireless network, python and C language standards.

3.1.1. Lead Soldering Safety Standards

This section discusses the safety standards to follow when soldering devices to the printed circuit board. Solder contains lead which is considered to be a toxic substance that has significant health risks. Therefore, it is important to safely handle solder to minimize health hazards. [1]

3.1.1.1. Soldering Iron Safety

Soldering irons are usually set to about 400 degrees celsius, therefore they must be handled with care to avoid burns. Any wires or components being in direct contact with the soldering iron must be handled with tweezers, pliers or clamps. Soldering must be conducted on a solid, leveled and grounded surface. It is also important to keep the tip of the soldering iron as clean as possible, residues of lead can build up increasing the fumes released into the air. To do so maintain the cleaning sponge wet and clean the soldering iron after every solder or when necessary.

3.1.1.2. Solder, Flux and Cleaners

Make sure to follow manufacturer's instructions and standards that are provided by IPC. To reduce the direct exposure to lead, use lead free or low lead solder when possible. It is important to wear eye protection and wash hands after soldering sessions to eliminate the possibility of ingestion or contact with eyes.

3.1.1.3. Personal Exposure

It is important to wear protective clothing to avoid burns, and keeping a clean and properly managed soldering area. Do not eat or drink in the soldering area to avoid ingestion that can cause serious health consequences. Wear masks and work on a well ventilated area to avoid inhalation of soldering and flux fumes. These fumes can cause damage and irritation of the respiratory system.

3.1.1.4. Waste Management

Make sure to follow RoHS standards when disposing of lead soldering waste since it is very hazardous. Lead solder must be discarded in a properly labeled container with lid. There is only one drop container allowed in each soldering location and the labels must be supplied by EH&S (Environmental Health and Safety).

3.1.2. IPC PCB Standards

IPC, Institute for Printed Circuits is a global trade association connecting electronic industries that provides standard to assure quality, reliability, and consistency in electronic assembly and manufacturing process. IPC is accredited by the American National Standards Institute (ANSI), however the standards apply to every manufacturer, designer, assembler and testers around the world. Several IPC standards apply to the design and component mounting. IPC-2221B covers the Generic Standard on Printed Circuit Board Design, specifying board assembly testability, layout design, base and conductive materials, electrical properties, thermal management, interconnections and more. IPC-A-600G covers the acceptability of printed boards, providing information on various solder coatings, printed contacts, solder mask, etc.[2]

Following these standards will guarantee quality, reliability and accountability of the final design of the power and signal PCBs. Thus, increasing the life of the product and improving communication between designer and supplier. Compliance with IPC ensure the quality tests of the PCB are met in the production line which minimizes additional costs due to rework and delays.

3.1.3. Power Supply Standards

Product conformance marks worldwide provide easy identification of compliance with specific industry standards. “Conformité Européenn (CE Mark), was introduced a little over 30 years ago for products sold within the European Economic Area to show conformity with legal requirements in respect of relevant safety, health and environmental directives”[3]. IEC60950-1 characterizes technology equipment in different classes based on their power supply. The table below gives an overview of the various classes and their characteristics.

Table 3.1.3-1: Classes of Technology Equipment

Class I	Equipment achieves electric shock protection through basic insulation and protective earth grounding. This requires all conductive parts that could assume a hazardous voltage in the event of basic insulation failure to be connected to a protective earth conductor.
Class II	Equipment provides protection using double or reinforced insulation and hence no ground is required.

Class III	Equipment operates from a SELV (Safety Extra Low Voltage) supply circuit, which means it inherently protects against electric shock, as it is impossible for hazardous voltages to be generated within the equipment.
-----------	---

To understand what type of class our project falls in it is necessary to look at the various Circuit Definitions shown in the following table.

Table 3.1.3-2: Circuit Definitions

Circuit Type	Definition
Hazardous Voltage	Any voltage exceeding 42.2 Vac peak or 60 Vdc without a limited current circuit.
Extra-Low Voltage (ELV)	A voltage in a secondary circuit not exceeding 42.4 Vac peak or 60 Vdc, the circuit being separated from hazardous voltage by at least basic insulation.
Safety Extra-Low Voltage (SELV) Circuit	A secondary circuit that cannot reach a hazardous voltage between any two accessible parts or an accessible part and protective earth under normal operation or while experiencing a single fault. In the event of a single fault condition (insulation or component failure) the voltage in accessible parts of SELV circuits shall not exceed 42.4 Vac peak or 60 Vdc for longer than 200 ms. An absolute limit of 71 Vac peak or 120 Vdc must not be exceeded.
Limited Current Circuits	These circuits may be accessible even though voltages are in excess of SELV requirements. A limited current circuit is designed to ensure that under a fault condition, the current that can be drawn is not hazardous. Limits are detailed as follows: Δ For frequencies < 1 kHz the steady state current drawn shall not exceed 0.7 mA peak ac or 2 mA dc. For frequencies above 1 kHz the limit of 0.7 mA is multiplied by the frequency in kHz but shall not exceed 70 mA. Δ For accessible parts not exceeding 450 Vac peak or 450 Vdc, the maximum circuit capacitance allowed is 0.1 μ F. Δ For accessible parts not exceeding 1500 Vac peak or 1500 Vdc the maximum stored charge allowed is 45 μ C and the available energy shall not be above 350 mJ.

The power and signal PCB design fall under the ELV circuit type falling under Class I equipment classification with a maximum voltage of 12V. All other components that require a power supply which will be managed through the power PCB such as motors, LEDs, drivers, and pressure sensors are CE

compliant based on the specification provided in their datasheets. It is important to meet all these industry requirements and standards due to the final product being used by the public addressing any liability and safety requirement that will allow for a fully functional non defective product.

3.1.4. RoHS

RoHS compliance is required by the electronics industry. RoHS stands for Restriction of Hazardous Substances, it restricts the use of lead, mercury, cadmium, hexavalent chromium, polybrominated biphenyls, polybrominated diphenyl ethers, bis phthalate, benzyl butyl phthalate, dibutyl phthalate, and diisobutyl phthalate. [4] These chemicals are dangerous to health and the environment during manufacturing and disposal. Compliance with Waste from Electrical and Electronic Equipment (WEEE) is a must to properly dispose/ recycle electric and electronic equipment containing these chemicals.

Several of the parts used in this project are listed as RoHS compliant such as transducers and LED strips. Furthermore, the use of lead to solder components into the printed circuit board must be limited and the soldering procedure must be compliant with RoHS. RoHS dictates the maximum level of lead to be less than 1000 ppm (part per million) or mg/L [5]. While the usage of lead solder will not come close to this number, there are lead free soldering alternatives and safety precautions that can be taken to reduce the risk of personal exposure.

3.1.5. Wireless Network Standards

Wireless communications will be implemented for the screen mirroring feature of the system. Configuring the wireless network for an ad hoc connection to optimize latency and allow for smooth viewing of the application in process. Encryption of the network connection is a secondary requirement as sensitive data will not be flowing through the system. Wit that in mind, it is still important to keep the network secure so that only the intended user will be able to interact with the system variables. Overall the protocols should be efficient, secure, and simple to allow for the best performance.

3.1.5.1. Wired Equivalent Privacy

WEP is a security protocol specified by IEEE Wireless Fidelity Standard. The protocol was designed to provide the same standard of security which one would experience over a wired network connection. Because wireless networks are not bound by walls and physical security restrictions, security is a top priority for the network. WEP being the first major wireless network security was effective, but not without its flaws. The University of California at Berkeley through research were able to report major security flaws in WEP, making the protocol vulnerable to attacks. These attacks are able to intercept and modify transmissions gaining access to WEP networks. The protocol is still very effective as the Wireless

Ethernet Compatibility Alliance never intended WEP to be the sole security mechanism for wireless networks. [6]

3.1.5.2. Wireless Protected Access

WPA is the successor to the security protocol WEP. WPA improved upon WEP data encryption, and provided efficient user authentication, where WEP was deemed insufficient. WPA uses an encryption method called Temporal Key Integrity Protocol (TKIP). TKIP uses a per-packet mixing function, integrity check, and an extended initialization vector and re-keying mechanism. [7]

3.1.5.3. Wireless Protected Access 2

WPA2 is the successor to the security protocol WPA and is the most current security protocol for wireless networks. WPA2 implements Counter Mode Cipher Block Chaining Message Authentication Code Protocol (CCMP). CCMP provides a much stronger and more reliable secure network than TKIP implemented in originally in WPA. Although WPA2 goes through great lengths to ensure the security of a wireless network it is not perfect. A recent exploit has been discovered using key reinstatement attacks, which allows the attacker to gain access to data which was previously assumed to be encrypted and secure.

Although WPA2 is not a perfect solution to providing security to wireless networks, it is the most secure protocol currently. Due to this the Raspberry Pi 3 ad-hoc connection was configured to use WPA2 protocol. Because our implementation of the Raspberry Pi 3 will not be transferring sensitive data, the security flaws in WPA2 become even more minute than they previously were considered. This led to the decision to implement WPA2 protocol as the best option for the systems implementation. [7]

3.1.5.4. 802.11n

802.11n is the most recent specification for wireless LAN communications drafted by IEEE. 802.11n is a set of standards designed to increase the network speed and improve reliability at extended ranges. 802.11n uses multiple input / multiple output technology, as well as frame aggregation to decrease the time between transmissions. Using this set of standards the network is able to reach up to 600 Mbps, but typically between 100 and 200 Mbps on moderate hardware. These data rates work perfectly for the design of the system. A remote desktop application recommends a minimum network throughput of 128 Kbps, the 802.11n providing throughput on the scale of Mbps allows for plenty of overhead and smooth client viewing even at a very high resolution. [8]

3.1.5.5. 2.4 GHz

The band frequency of 2.4 GHz has been marked by FCC regulations as Industrial, Scientific, and Medical bands. 2.4 GHz is an unlicensed band to which

can be freely operated on and is commonly used for Wireless Networks. This allows this band to be used for free in the implemented system without interfering with other communication devices. 2.4 GHz allows for a good balance between range and penetration, as well as no need for a large antenna. This is important because as classrooms become larger the frequency band will need to travel a farther distance as well as through obstructions such as chairs desks and the outer lining of the system. [9]

3.1.6. Python Language Standards

Tim Peters, a long term Python programmer, in PEP (Python Enhancement Proposal) 20 emphasis how critical “readability counts”[10]. Because consistency is important with developing software, we decided to follow a majority of the style guidelines laid out in “PEP 8 -- Style Guide for Python Code” [11].

The following subsections under 3.1.6. deal with the coding conventions we used when developing our Python source code. The software development team agreed to follow the guidelines listed below unless abiding by them made the code less readable, less consistent with the remaining code (for example, if using older, third-party modules) and/or incompatible with the Python interpreter used on the Raspberry Pi. Another reason why we may have temporarily opted out of following our guidelines was if we were reusing prior code that predated this project’s style guidelines. Python programmers Guido van Rossum, Barry Warsaw, and Nick Coghlan claim that there is no reason to modify old code; it just consumes time unnecessarily and unproductively[11].

Code Layout

The following sections. deal with file formatting specifics. Indentation, tab-related details, line lengths, line breaks, blank lines, file encoding, imports, and dunder names technicalities are all addressed. Arguably, this section is the most important and strictly-followed one for our Python standards. Since Python uses whitespace to identify blocks of code, knowing when to use tabs or spaces and how many is critical. If this is inconsistent, the Python script will not be able to be interpreted. Also with imports and dunder names, if there is not a consistent method for implementing these, the interpreter may run into confusion.

Indentation

Unlike what the PEP 8 style guideline suggests, one tab was used per indentation level. If any lines continued into subsequent text line(s), continuation line(s) were aligned one indentation level (of one tab) greater than the initial line indentation level. Arguments were not used on the first line whenever a line of code continued into the next line of the file. Any closing parenthesis, bracket, or brace in a multiple-line construct was lined up under the first non-whitespace character of the last line of the construct [11].

Tabs Versus Spaces

A tab was used to perform indentations. A space is 0x20; a tab is 0x09. Because of the difference in their values Python does not interpret spaces and tabs the same despite four spaces appearing identical to one tab to the eyes of the programmer.

Maximum Line Length

The number of characters per line of code was limited to a maximum of 79. For comments and docstrings, the limit was 72. We limited the characters to these values to allow for viewing two files side-by-side without obstructing code from the other file. PEP 8 explains how even if the text editor used has an auto-wrapping feature, the default tool often disrupts the visual structure of the code since it does not wrap based on semantics [11]. Whenever code needed to be wrapped, the continuation guidelines under **Indentation** section were followed.

Line Breaks

When it comes to long, mathematical expressions, line breaks were used to keep the code within the 79 character limit mentioned in **Maximum Line Length** section. In order to prevent operators quickly visible to keep the code as clear and readable as possible, line breaks were inserted before operators in long expressions so that the operator was the first thing typed in the next line. Compound statements were avoided; an individual line was used for each statement.

Blank Lines

Each class and high-level function definition was separated by two blank lines above and below it. Method definitions within a class were encased with one blank line (one above and one below the definition). We also decided to use a single blank line to surround loops, conditional expressions, blocks of code, and closely related lines to identify logical sections.

Source File Encoding

As per PEP 8 guidelines, we chose to use UTF-8 encoding since we were using Python 3 [11]. The Python Enhancement Proposal 3131 encourages programmers to use ASCII-only identifiers and English words whenever possible (no abbreviations, acronyms, or technical terminology) to enhance readability [12].

Imports

Individual lines were used when importing modules. This was always performed at the top of each file after file comments and before declaring global variables

and constants. Whenever applicable, standard library imports were grouped first, followed by third party imports, followed by local imports. Each type of imports were separated by a single, blank line. Absolute imports were used whenever possible. e.g.: `from package import absolute_import`

In addition, wildcard imports were avoided to prevent confusing the interpreter with conflicting names in the namespace. Wildcard imports use an asterisk, e.g.: `from package import *`

Module Level Names

Dunders, names surrounded by two underscores (e.g. `__main__`, `__name__`, `__version__`) were placed after the module docstring but before the import statement section mentioned in previous section.

String Quotation Marks

PEP 8 states that both single- and double-quoted strings are the same functionally [11]. We chose to use single-quoted strings because doing so minimized keystrokes. However, when it came to docstrings, PEP 257 states that double-quotes should be used [13].

Whitespace

In order to avoid unnecessarily filling up the empty space in the source code, extraneous whitespace was not used. Whitespace inside parentheses, brackets, and braces; before commas, semicolons, and colons; and between a trailing comma and the closing parenthesis were not used [11]. No spaces were left between a function call and its argument list or between an object and an index or slice value. As per suggestions in PEP 8, single spaces were used around operators with lowest priority in expressions with multiple operations happening [11]. A single space also was used to surround binary operators of assignment, augmented assignment, comparisons, and booleans. The only exception was when the “=” sign was used in the argument of a function for a keyword argument or a default parameter [11].

Trailing Commas

Trailing commas were not used whenever optional. In the cases where they were mandatory (i.e. when creating a one-element tuple), parentheses were used to enclose the one-element tuple for readability purposes.

Comments

In effort to avoid miscommunication to ourselves or anyone else reading our code, we agreed that all comments had to be complete sentences with proper punctuation and grammar. Only one space was used after a period. There are three main styles of commenting: block comments, inline comments, and documentation strings; we agreed not to use documentation strings because we felt they were unnecessary for our purposes.

Block

Whenever we used a block comment, we ensured that it was indented at the same level as the block of code that it described. Each line of the comment was initiated with a “#” and a space. If we had a multi-paragraph block comment, we separated each paragraph by a line with only a “#”.

Inline

As per suggestions in PEP 8, inline comments were completely avoided [11]. These comments are made on the same line as the statement of code.

Naming Conventions

Naming conventions are another key style guideline to establish. If each type of variable has their own style, anyone reading the code can quickly and easily glance at any variable and tell what type it is.

Modules

For any packages and modules we created for our system, we used short, lowercase names [11].

Classes

For all classes defined in our code, we used the CamelCase convention which means that the first letter of every word used in the class name is capitalized. If an abbreviation was used in the class's name, the whole abbreviation was capitalized.

Type Variables

Type variables also followed the CamelCase convention.

Exceptions

User defined exception names also used CamelCase for their names with the addition of “Error” adjoined to the end of the name [11].

Functions

Function and method names followed an all lowercase snake_case convention which means that each individual word was separated by an underscore.

Global and Instance Variables

Global and instance variables were also all lowercase and in snake_case.

Constants

Finally, module level constants were named in an all capital convention utilizing underscores to separate individual words.

3.1.7. C Language Standards

Due to the unavailability to the C standard without licensing the document. A custom standard was laid out and implemented for programming the microcontroller. In order to be as consistent as possible between all the source code, the PEP 8 style guide was referred to for formatting guidelines. The following sections deal with file formatting and internal syntax specifics. Indentation, multi-line continuations, curly-brace placement, character limits, operations per line restrictions, tab-related details, line lengths, line breaks, blank lines, includes, and jump command technicalities are all addressed. Unlike with Python code, whitespace, tab/space consistency, and indentation are not as critical with C code. So most of these guidelines are purely for readability and producing clean code's sake.

Indentation

Tabs will be used to indent blocks of code, blocks of code will be indented immediately following an opening brace and a new line character, or an optional brace statement and a new line character.

Line Continuation

If any lines of code continued into another line of code, the continuation line(s) were aligned one indentation level (of one tab) greater than the initial indentation level of that line of code. If a function with its arguments required more than one line of text, no arguments were placed on the initial line. Any closing parenthesis, brackets, or braces for a line of code that spans multiple lines was lined up underneath the first non-whitespace character of the initial line of the construct. The closing symbol was immediately followed by the semicolon.

Braces

Parenthesis will not be used when the body of the code within the construct is only one line. This applies to all if (and else if and if), while, for, and do statements.

Braces for classes, functions, and statements such as if, for, ect. will implement K & R style [9]. This style places the opening brace directly after the statement without the need for a line break. The closing brace will be placed vertically, inline with the first non-whitespace character of the initial line of the construct. See the **Maximum Line Length** segment under Section 3.1.6 for a more detailed explanation.

Empty parentheses, brackets, and braces shall be condensed with no space line break between them: `void foobar() {};` .

Limits

These two following sections deal with reducing the amount of code that can be on a single line to improve readability by reducing the complexity of a single statement.

Statements Per Line

Each statement will be followed immediately by a newline character. Multiple statements will not be written on the same line.

When an if, for, while, or do statement was used, a new line separated the condition from the body. See the **Maximum Line Length** segment under Section 3.1.6 for a more detailed explanation.

Maximum Line Length

In order to be consistent with the Python standards used a column will not exceed 79 characters. This number was chosen to allow for viewing two source code files side-by-side without cutting out any text by the window. If it is unavoidable that a single column will have more than 79 characters line-wrapping will be used. Lines where this behavior is not possible include long URLs, package, and include statements.

Line Breaks

A single blank line must appear between methods, and between import statements.

A single blank line may appear between statements as needed to organize code. Two blank lines must appear between classes in the same file.

When an arithmetic or logical expression violated the maximum line length standard (see the above paragraph) a line break was used to keep the code within the 79 character limit. In order to keep the involved operators visible, line breaks were inserted before operators in long expressions so that the operator was the first symbol typed in the next line.

Includes

Individual lines were used for including header files. This was always done at the head of each file after the file comments but before defining any constants or initializing any global variables. These includes were grouped first by standard library includes, then by third party includes, and finally by local or custom includes. Each group of header files were separated by a single, blank line.

Jump Statements

Statements that cause the program to jump out of sequence such as “goto”, “break”, and “continue” were avoided. Instead, we broke the code into smaller functions and constructs to improve traceability [14].

Variable Declarations

The purpose of consistent variable declarations is to compile all of the variables used in the code so that anyone reading can see all the variables used in that class or function immediately. This can help eliminate redundant or unnecessary variables when refactoring the source code. Restrictions on how variables can be

declared can also reduce complexity and help clean up the code to improve readability.

One variable per declaration

Each variable will have its own line as well as its own data type. Variables will not be stacked in same declaration regardless of sharing types.

Declare at Beginning of Function

Local variables will be declared at the beginning of the function in which they are used.

Comments

In effort to avoid miscommunication to ourselves or anyone else reading our code, we agreed that all comments had to be complete sentences with proper punctuation and grammar. Only one space was used after a period. There are three main styles of commenting: block comments, single line comments, and inline comments.

Block

Block comments will be used at the beginning of each file with author's name revision date, and a brief description of the code to come. Other uses were to at the beginning of substantial functions or blocks of code. Block comments are opened at the same indentation level with a forward slash and an adjacent asterisk (`/*`) and are closed at the initial indentation level with an asterisk and an adjacent forward slash (`*/`).

Single Line

Single line comments were used and indented to match the line being commented. Each of these comments began with a double forward slash (`/**`) and a space. If we had a multi-paragraph single-line comment, we used the block comment method as described in section 3.1.5.2.3.1. However, if there were just a few sentences of comments, we just used multiple single line comments.

Inline

In effort to be consistent with our other source code, inline comments were completely avoided. these types of comments are made like single line comments (see section 3.1.7.2.3.2.) except immediately after on the same line as the statement of code they describe (see section 3.1.4.5.2).

Naming Conventions

Naming conventions are another key style guideline to establish. If each type of variable has their own style, anyone reading the code can quickly and easily glance at any variable and tell what type it is.

Header Files

For any header files we created for our system, we used short, lowercase names.

Class Names

Class names will use CamelCase convention where the first letter in the class name will be uppercase followed by an uppercase letter for each definitive word. If an abbreviation was used in the class's name, the whole abbreviation portion of the name was capitalized. Structs also followed the same naming convention as classes.

Function Names

Function names will use an all lowercase snake_case convention where each definitive word is separated by an underscore and all letters will be lowercase.

Constant Names

Constant names will use an ALL_UPPER_SNAKE_CASE where each word is separated by an underscore and all letters will be uppercase.

Variable Names

Variable names will use lower_snake_case where each word is separated by an underscore and all letters will be lowercase.

Parameter Names

Parameter names will use lower_snake_case where each word is separated by an underscore and all letters will be lowercase.

Errors

Any variable used as a flag will use CamelCase for their names with the addition of "Error" adjoined to the end of the variable name.

3.1.8. Shells Scripting Standards

Since the Raspberry Pi needed to initialize itself without any user support, we created scripts that ran when the device booted up. According to Paul Armstrong, author of "Shell Style Guide", Bash (Bourne-Again SHell) is the sole language able to utilized to create executable scripts [15]. Because of that, Bash was chosen to be the language in which all our scripts were written.

File Extensions

In order to signify that the file is a Bash script, the file extension ".sh" was added even though this is unnecessary in Linux operating systems like Raspbian on the Raspberry Pi.

File Header

Each Bash script must start with a header comment starting with “#!/bin/bash”. This shebang tells what interpreter to use with read the file with. In a new line, a description of the file was given.

Error Handling

In order to to catch exceptions, “|| exit 1” was appended to the end of lines that were critical to run. This was done because scripts exit with a return of 0 if it was successful and a non-zero value if it it was unsuccessful [16].

Syntax

Curly braces were used around variables to clearly show what the variables were (i.e. \${variable}).

Comments

Comments were initiated with the “#” symbol followed by a space. Only single- and multi-line block comments were used.

Indentation

In order to stay consistent the spacing guidelines we used for our Python and C code, we used four spaces for our indentation. Blank lines between logical blocks of code were used to improve readability.

Line Length

The maximum line length was limited to 79 characters. Since lines that are longer than that cannot be separated easily, nothing was done to split long lines.

3.2. Constraints

This section discusses the various constraints that affect this project. These constraints include but are not limited to time, mechanical and various hardware constraints.

3.2.1. Time Constraints

The most significant constraints of this project is time. Due to academic requirements the final product has to be delivered by the end of Senior Design II in the spring of 2018. Thus, time management is very important throughout the course of the two semesters. While Senior Design I consists mostly of deciding on a project idea, identifying requirements and necessary parts, developing a concept and the proper documentation, it is necessary to expedite and set milestones that are usually set for Senior Design II in order to complete the project. There are several aspects of the project that would cause delays in the planned deliverables and milestone completion, these include: mechanical parts

definition, identification of electrical components that are compatible and meet the requirements (torque, pressure sensing, metering), lack of experience and knowledge of mechanical flow equations, and parts ordering lead time. Due to the unfamiliarity with mechanical fluid flow, copious amounts of research delayed some of the milestones completion. Also, the lack of understanding of the behavior of liquids through pipes and pressure losses, led to a significant amount of testing the original mechanical systems without the electrical implementation. The necessary testing delayed some of the electrical design and software development. Once the mechanical elements of the projects were identified and fully understood, it became easier to develop the necessary equations to be utilized in the software development and power management.

3.2.2. Economic Constraints

Economic constraints were those that externally restrained our project, whether in the brainstorming, prototyping, developing, testing, or documenting phases. Economic constraints specifically relate to economic factors. These can be broken up into two categories: macroeconomic constraints and microeconomic constraints.

3.2.2.1. Macroeconomic Constraints

Macroeconomic constraints relate to large-scale factors which affect the whole economy and any entity that participates in it. One of the biggest effects dealt with supply production. For example, we needed a water pump that was compact so that it did not occupy much space on our display but also powerful enough to create enough pressure build-up in the PVC pipes. However, since there was no demand for small pumps except for aquarium like applications and since there was no demand for powerful pumps except for swimming-pool applications, smaller, powerful pumps were rare and expensive.

3.2.2.2. Microeconomic Constraints

Microeconomic constraints relate more precisely to those which affected our team and project and not a broad scope of other entities. One of the scarce resources we had to manage was our budget. Siemens sponsored our project by donating \$5,000 to help support our financial burden. Being effectively limited to \$5,000, we were force to be cautious with our spending, especially when it involved adding an item to our budget or replacing an item on it.

3.2.3. Social Constraints

Although cheap and/or free labor would have been a simple way to increase our efficiency, in order to comply with moral and ethical obligations, this project was not tested with animals and no small children were utilized to assist in any part of the development processes. Each individual involved with in this project has schedules outside of this project. Due to the fact that this project is not feasible to

be each individual's full time job scheduling to work on the project will be a tasked and scheduled. Meetings will need to be efficient and placed on weekends or other times where each member is able to spare time from their other activities such as school and work.

3.2.4. Health and Safety Constraints

Waters quality was one issue we realized may affect the health of the surrounding environment of the project. Although it would not be evident until after the danger spreads, standing water can be an incubator for several types of bacteria and parasites. Not to mention insects can use it as a breeding ground for their young [17]. Due to the portability of our project, we were not able to add an automatic water-replacement system to it; however, after a couple weeks, an error message would display on the touchscreen reminding the operators to check and replace the water used.

3.2.5. Hardware Constraints

There are three hardware constraints that affect the project tremendously. Waterproofing all the electrical components including the PCBs is very important since this project main concept deals with handling water. Mechanical constraints have a significant effect on the project's design and components definition. The computational portion of this system involves a full CPU with an operating system as well as a MCU. The two devices will communicate using I2C protocol and will act as the brain portion of the system.

3.2.5.1. Water Proof

Ingress Protection (IP) Rating followed by a two digit number is used to classify degrees of protection provided by an electrical enclosure. This a standard approved by the American National Standard (ANSI) and published by the National Electrical Manufacturers Association (NEMA). This classification provides "protection of persons against access to hazardous parts and protection of equipment against the ingress of solid foreign objects and the ingress of water" [18] The first digit indicates the level of protection against contact or ingress of solids including but not limited to screws, wires, large surfaces of the body and dust; the numbers range from 0 to 6. The second digit is strictly dedicated to ingress of liquids (water), including but not limited to dripping, splashing, water jets and immersion; the classification ranges from 0-8. For both of these digits the highest being the maximum level of protection.

Since the functionality and visibility of this project revolves around the application of water to establish relation to current in circuit theory, it is important to provide ingress protection for all the electrical equipment. Due to the high risk of water contact, a minimum of IP66 is required. This classification provides protection against ingress of dust and projection of of powerful jets of water against the

enclosure from any direction. While none of the electrical equipment will be immersed in water with the exception of the pump, it was strongly considered to acquire components such as LEDs with IP67 providing protection against some sort of immersion in case of accidents.

3.2.5.2. Mechanical Constraints

Electrical and Computer Engineering students have very limited understanding and/or knowledge of fluid mechanics. However, circuit analysis and programming come as second nature. Familiarization with fluid flow equations and pressure losses are a must to be able to develop the concept of the project. Initially, a design of the outlook of the system was created based on little understanding of the mechanical effects on the circuit representation. Fluid flow restrictions altered the design of the components such as MOSFET, BJT and diode. Due to having to identify and demonstrate pressure changes to relate them to voltage, several mechanical concepts had to be considered such as head losses, pressure losses due to the fittings.

Mechanical parts such as the valves had to meet requirements such as having low operating torque. The reason for this restriction is because small DC stepper motors are limited in the amount of turning torque they provide. The sizes of the motor were also restricted due to the requirement of having a visually appealing and a lighter weight product. Higher torque stepper motors are of bigger dimensions and much heavier. For example, since the operating torque of the valves is 2Nm, the original motor was chosen to have 2.4Nm. This 2.4Nm motor met the requirements necessary to turn the valves, however, it had a body length of 10.4 cm, frame size of 5.7 by 5.7 cm, and 1.5kg. Having several of these motors (11) in our project would have added 15kg to the overall weight of the project. A higher torque motor also requires a higher rated current (e.g. 1.8A) which would increase the power consumption of the overall system.

It is essential to keep the dimensions of the system small enough to be portable and big enough to be visually appealing and understandable. Reducing the size to the bare minimum would make the board look cluttered and hard to understand; it would also cause changes to the calculated pressure losses which would affect the overall system. The longer the pipes, the easier it is to demonstrate pressure differences that are directly correlated to the voltage differences. If the pipes are short, the pressure differences would not be significant enough for the pressure sensors to measure and use for calculation purposes.

Clear PVC pipes are not easily or readily available. Clear fittings such as male and female adapters, elbows, and tees are much more expensive than regular white PVC. Therefore, the pipes associated with current flow and connection of parameters are all clear PVC to be able to make visible the flow of water. However, fittings that connect to flow meters, gauges and valves are of white

color. Thus, optimizing the budget while also not affecting the integrity of the visual design.

Another big constraint is the sensor and flow meter resolution. Due to having very low precision in these measuring tools, a lot of testing had to be done to obtain accurate results. Higher resolution meters and gauges would have increased the overall budget of the project tremendously. The objective of this project is to demonstrate circuit theory, therefore precision is not of utmost importance. Performing tests and acquiring data for the necessary points was sufficient for our purposes as long as a certain percentage error was assumed.

4. Research and Background Information

Chapter 4 covers the research performed to be able to choose the various components and technologies necessary to construct the Water Analogy Board. It gives an overview of similar projects and their functionalities compared to this project. Computer and controller hardware considerations such as the raspberry pi and the ATmega 2560 were discussed along with their capabilities to fit this project. Wired and wireless communication technologies were researched and compared to find the most suitable for the project. Research was conducted to learn how to design and construct a PCB including the terminologies, requirements and necessary components for our project specific applications. Mechanical components such as ways to control the valves, motors types, motor drivers, water pumps, flow meters and pressure sensors were investigated. Additionally, software tools to make the project management, software development and hardware design easier were considered and utilized through the course of the project.

4.1. Similar Projects

This idea originated based on a visit to Eaton Experience Center in Pittsburgh, PA. The Experience Center has a system containing only resistor, KVL, KCL, inductor, and capacitor representations without any user control on the dashboard aside from turning the system on and off. The valves and components in the system are handled manually by the user, thus providing room for overexertion of the pipes, no control over the current flow, and no interactive dashboard. Our project builds on Eaton's concept, optimizes the design, incorporates transistors and diode, adds friendlier fully interactive interface and hands free capability.

4.2. Computer and Controller Hardware Considerations

Computer hardware was a main area of research. Since the whole system had to be operated through the selected architecture, it was critical to make sure the device had a high enough clock speed, memory, storage, and enough General Purpose Input/Output (GPIO) pins to control several LEDs, motors, and a water pump. The device chosen must be compatible with a touch screen and have network capabilities to work with Virtual Network Computing (VNC) for external, educational use.

4.2.1. Raspberry Pi

At first glance, the Raspberry Pi seemed like an optimal first choice. The Raspberry Pi is a Single Board Computer (SBC) that has a quad core 1.2GHz

Broadcom BCM2837 64bit CPU according to its website [19]. With the latest generation, Raspberry Pi 3 Model B, it includes hardware to support wireless Local Area Network (LAN), which would permit VNC connection. The Raspberry Pi also contains a Digital Serial Interface (DSI) port for operation with a touch screen. In terms of operating the system, the Raspberry Pi provides simple means of user input and output whether locally, through a touch screen graphical user interface (GUI) or remotely through a virtual user interface accessed through a VNC connection. Unfortunately, although the Raspberry Pi has 40 pins, only 26 are GPIO (6 are general) pins. However, since the Raspberry Pi runs Raspbian, a Linux operating system distribution, programming the computer and operating it would be simpler.

4.2.2. ATmega328P

The ATmega328P is a 8-bit microcontroller. It has 32KB flash memory, 2KB of RAM, and 23 general purpose I/O pins, 6 of which are capable of pulse width modulation [20]. The microcontroller was considered because of its small physical size. The microcontroller is also implemented on the arduino uno making for easy development and testing of the code. Upon further investigation the microcontroller was also capable of serial communication such as UART, SPI and I2C [21]. Communication in parallel is not an option on this device due to the limited number of GPIO pins, and ultimately the reason this microcontroller was not selected was due to it's lacking of GPIO pins. It wouldn't be unfeasible for all of the components to be run off of only 23 pins with communication implemented.

4.2.3. ATSAMD21G18

The ATSAMD21G18 is a 32-bit microcontroller. It has 256KB flash memory, 32KB of RAM, and 20 general purpose I/O pins 6 of which are capable of pulse width modulation [22]. The microcontroller was considered because of it's easy development on the arduino zero. Having access to the arduino IDE is an ideal situation for developing and testing before implementation. The microcontroller is also able to communicate using serial methodologies "Six serial communication modules (SERCOM) configurable as UART/USART, SPI or I2C, three 16-bit timer/counters" [22]. Communication using parallel methodologies would be difficult because of the constraint on the number of GPIO pins. The microcontroller only having 20 general purpose input and output pins would put a constraint on how many pins can be used for communication. Given the restraint on the number of GPIO pins the microcontroller proved to not be an optimal choice.

4.2.4. MSP430

One of the families of microcontrollers considered was the MSP430. This was the first name in microcontrollers considered because it is one that each group member was familiar with using [22]. One advantage to using these microcontrollers was their 16 bit word width. This would allow for faster

arithmetical and logical computations. According to chapter 8, “Digital I/O Introduction” their family guide, MSP430 microcontrollers can have up to eight digital GPIO ports with up to eight pins each [22]. Although driver control would not be an issue with the amount of pins a MSP430 would offer, the maximum amount of RAM that come with these processors is 66 KB of RAM which would not be enough memory to run the program code (which will involve connecting to a computer via VNC) and the GUI XML [23].

4.2.5. ATmega2560

The ATmega2560 is an 8-bit RISC based microcontroller. It has 256KB of flash memory, 8KB of RAM, and 54 general purpose I/O pins 14 of which are capable of pulse width modulation [24]. This Microcontroller was considered because of its sheer number of General Purpose I/O pins. This is important to the system because of the number of components which need to be controlled through the microcontroller, 6 motor drivers, several LED strips, communication with the Raspberry Pi 3, and the pump. The ATmega2560 providing 54 GPIO pins allows for flexibility on how to communicate with these components without the hinderance of number of available pins. The ATmega also is implemented on the arduino mega 2560 development board, which could allow for easy development of the MCU and testing before being implemented off the board. The MCU has an input voltage of 7-12 volts and it operates at 5 volts. It is capable of providing 40 mA of DC current through Input Output pins and a clock speed of 16MHz [25]. The variety of GPIO pins on this microcontroller also offer a wide variety of ways to communicate with the Raspberry Pi 3 both in parallel and serial methodologies. “I2C: 20 (SDA) and 21 (SCL). Support I2C (TWI) communication using the Wire library.”[25], the ability to communicate with the device through a serial communication such as I2C will allow for better synchronization, and more free GPIO pins to communicate with components.



Figure 4.2.5-1: ATMEGA2560 16AU [26]

4.2.6. Conclusion

After much deliberation, the Raspberry Pi was selected to handle the user interactions with the system through VNC and the GUI. Because of its quad core processor, handling more graphic and memory intensive applications suited it better. Although it was possible to optimize the wiring of peripheral devices to the Raspberry Pi, in order to meet the requirements established by the senior design coordinators, the ATmega2560 microcontroller was selected to work in conjunction to the Raspberry Pi as its slave. This allowed for more flexibility for add-ons and adjustments to the hardware because of the ATmega2560's 54 GPIO pins (which is significantly more than the Raspberry Pi). The microcontroller, on receiving commands and data from its Raspberry Pi master, handles arithmetic and logical computations and controls the motors (through the drivers), the water pump (through a driver as well), and LEDs.

4.3. Wired Communication Technologies

Wired communication will be used between the Raspberry Pi 3 and the microcontroller, as well as between the microcontroller and the LEDs. The communication protocol implemented will be responsible for transferring data between devices. The protocol must be able to communicate with multiple devices with minimal latency. The conservation of GPIO pins is also of high importance as there are a limited number of available pins for the systems

4.3.1. Parallel vs. Serial

Both Parallel and Serial methodologies have their advantages and disadvantages when it comes to data transmissions. Parallel transmissions are faster when compared with serial transmissions, because each bit can be sent at once. Serial on the other hand has to wait for the previous bit data to clear on the line before sending the next bit. This gives the ability to process data at a faster rate when communicating between the Raspberry Pi 3 and the ATmega2560. Due to the simplicity of parallel transmissions between the Raspberry Pi 3 and the ATmega2560 it was the first tested implementation. Originally six transmission lines ran between the Raspberry Pi and microcontroller. One line acted as a ready/not ready line to inform the microcontroller when it was safe to process the incoming data, one line for the direction of the motor to be implemented, and four lines to act as the actual data transfer lines, representing motors 1-6 in binary. This implementation allowed us to select the desired motor and the direction which it should be moved, then implementing a standard step sequence to assure proper implementation. This implementation was a success, but was deemed inefficient for our application. "Although parallel transmission can transfer data faster, it requires more transmission channels than serial transmission. This means that data bits can be out of sync, depending on transfer distance and how fast each bit loads." [27]. Due to the necessity of many GPIO pins to control motors, pumps, and LEDs it is in the best interest of the

system to use as few GPIO pins for communication. Serial communication does a much better job at this, as it only requires two lines for transmission, as well as the ability to ensure synchronization between the two communicators. For this reason serial communication was investigated as a viable alternative.

4.3.2. UART

Utilizing the Universal Asynchronous Receiver-Transmitter (UART) hardware for serial communication was first considered. Between the members of the group, everyone was familiar with configuring a UART interface, and since UART is convenient because it only requires one transmit line (TX) and one receive line (RX). (This limits UART's communication abilities to two devices per tx-rx connection pair.) The limitation of having only two devices per UART connection did not influence the decision to take advantage of this technology or not because only one Raspberry Pi 3 Model B and one ATmega2560 needed to communicate with each other. Note that it does not require a clock line (hence "asynchronous") between the devices. A device transmits with UART by setting the TX line low, followed by up to a byte, potentially a parity bit, then one or two stop bits. The only chance the devices get to resynchronize is on the transmitter's start-bit's falling edge. In order to maintain synchronicity between the devices, the UART on both devices should be initialized to the same settings: transmission speed, bit rate, data length, parity bit, and stop bits. Since the two participants in the serial communication operate at different frequencies (the Raspberry Pi's Broadcom BCM2837 processor runs at 1.2Ghz [19] and the ATmega2560 microcontroller runs at 16Mhz [28]) using an asynchronous method was worrisome. Although the devices can get in sync once at the start of every transmission, if the timings were not in sync within a margin of 30% with a deviation of 5% according to Keim's analysis, the transmitted data may be received differently [29]. Ultimately, the lack of a better synchronization mechanism dissuaded the use of this interface.

4.3.3. SPI

The second serial protocol researched was Serial Peripheral Interface (SPI). Unlike UART, SPI is a synchronous communication method that utilized a master-slave architecture. The sole master device generates the data frames for the slave. Dissimilar to UART, SPI requires the use of four lines. The Master Output Slave Input (MOSI) and Master Input Slave Output (MISO) function as their name describes. Since the ability for one master to have multiple slaves is supported, a slave select line (SS) is needed for the slave(s) to know which slave the master wants to address. And since this is a synchronous protocol, the master outputs a serial clock (SCLK) line as well. For unidirectional communication, the master device uses the SS line to select which slave device it wants to initiate communication. Then the master will output the SCLK signal, and one bit to the selected slave on the MOSI wire to which the slave will respond with one bit from that slave along the MISO wire [30]. This process

repeats until the master has completed its message, but despite this, Yuan Gao states in his response to “What are the pros and cons of I2C versus SPI interface?” that SPI is simple and faster than most other serial protocols. [31]. Unfortunately, SPI occupies more pins than other serial protocols explored. Since the ATmega will be controlling six drivers and one pump, conserving pins was important in the decision against using SPI.

4.3.4. I²C

The third serial protocol option examined was I²C. Inter-Integrated Circuit allows for multiple masters and multiple slaves. Similar to SPI, I²C is a synchronous protocol and similar to UART, I²C only requires two pins: a clock wire (SCL) and a data wire (SDA). Since there is only two wires, the master(s) and slave(s) utilize the same data line to communicate; this can cause issues if two signals are transmitted at the same time. Another drawback with using I²C is the fact that since there is only one data wire, the master must transmit a byte with the desired slave’s address before sending data [30]. Although this reduces the transmission speed between the devices this factor was overlooked.

With the system design in mind, a protocol that was synchronous and would require minimal pins was preferred. Since I²C only needed two wires, this left the rest of the digital GPIO pins for the motor and water pump drivers. Also since the ATmega2560 microcontroller was the only slave to the Raspberry Pi master, transmission and addressing conflicts would not be an issue. Despite the factor that the transmission speed is slightly slower with I²C, to have a synchronous (in order to reduce variables) serial protocol held more weight when deciding which communication method to utilize.

4.4. Wireless Communication Technologies

As technology advances so does education. To be able to display the system on a remote screen is vital as class sizes become bigger. Implementing an ad hoc network using the Raspberry Pi 3 eliminates the dependency on a third party network to be up and configured. With the ad hoc network able to accept connections from a client a Remote Display Mirroring application. This will allow the user to display the user interface on a 3rd party monitor, projector, or other displaying device. Allowing the information to be shared with the entire audience and allow interaction from multiple parties.

4.4.1. Ad Hoc Network on the Raspberry Pi 3

An Ad Hoc network powered by the Raspberry Pi was chosen to allow for communication to the system through a remote connection without the dependency on other variables such as other wifi network connections or access to the internet. The Raspberry Pi 3 having an integrated network card it can receive connection from other devices. Tools used for the Raspberry Pi to act as

an access point include hostapd, isc-dhcp-server, and udhcpd. These tools can be used to configure the access point's name, passwords, and selection of the wireless card acquiring values from a helpful tutorial [32]. Further the network can be started and stopped with a single terminal command. This terminal command is configured in the rc.local file and making the file executable, allows the command to run at the startup every time the raspberry pi is booted. The implementation allows for an access point to be immediately available upon startup without any interaction with the device.

4.4.2. Remote Display of the System

A remote display of the applications user interface was deemed a top priority. Education of students has changed drastically with the advancements in technology and larger class rooms. A seven-inch display is inadequate to present to an entire classroom of students, the ability to mirror and interact with the user interface on a projector or larger display solves this problem. The ad hoc network is an essential part of the remote displaying portion of this system. Once a user has connected to the ad hoc connection like any other WIFI connection the user is then free to remote desktop into the device. The remote desktop application chosen for this system is Realvnc. The server has been implemented on the Raspberry Pi 3 to allow connections to it and the client will be need to installed on the user's external device. Once the user has the client installed on their device they will enter the Raspberry Pi's static IP address and insert the password. This will instantly allow the user access to Raspberry Pi's interface. The implemented system allows for this ability to mirror the user interface without the use of unpredictable external variables.

4.4.3. Chrome Remote Desktop

Chrome Remote Desktop is a relatively recent development by Google. This software is installed as an extension to the Google Chrome browser and allows the user to remotely control the host computer which sends its screens interface to the client. Google's protocol allows for key and mouse events to be sent from the client.

The most convenient aspects is that in order to utilize Chrome Remote Desktop's features, the only thing required to install is the Google Chrome web browser and Chrome Remote Desktop's browser extension, all of which are free and simple to install. Since, according to W3Schools, 76.9% of people run Google Chrome [33], just requiring our remote users to download an extension to an application they probabilistically already have, reduces possibilities for external complications. Unfortunately, though, we decided to use the Raspberry Pi to host an ad hoc network to local area network connectivity issues (see section 4.4.1). This meant that the Raspberry Pi was not able to access the Google servers required for supporting their remote desktop application. Although there may be ways to spoof the application, Google Chrome is a resource hungry application,

meaning that it consumes a lot of the device's RAM. With the Raspberry Pi 3 Model B only having 1 GB and already experiencing a reduced amount of available RAM with Chrome running, relying on Chrome Remote Desktop for user interactions may stress the Raspberry Pi and create lag or other issues during operation.

4.4.4. Teamviewer

Teamviewer is a remote desktop application built to run on Windows, OSX, Linux, Android, and iOS. It is a free application possible of allowing a remote connection into the operating system running the server for teamviewer. It supports features such as wake-on-LAN. It also supports multiple users connecting to share a session if need be. Teamviewer runs through teamviewer servers which allows connection to the remote desktop without need for port forwarding or alterations to a router firewall. For the application which this would be used this feature is not needed. In fact the exact opposite is needed. While the remote sign in would be nice, these devices running the client and server portion of teamviewer will not be connected to the internet. They will be expected to communicate over an adhoc connection with no intermediate value.

Given the fact that teamviewer requires connection to the teamviewer servers and the devices will not have an internet connection this method can not be implemented. A method that is able to connect over ad hoc connection using the host and the clients IP addresses are a better option.

4.4.5. Virtual Network Computing (VNC)

VNC is a desktop sharing program with which users can transmit key and mouse events over a network connection. At its core, VNC utilizes Remote Framebuffer (RFB), a communication protocol which allows remote access of the GUI of a computer. One critical perk this provided is platform independence because it operates with the framebuffer in RAM. We wanted our users to be able to remotely interact with our system without any device restriction, so its platform independence and ability to work cross platform (as long as the software used support it) met that requirement. One large advantage VNC had over Chrome Remote Desktop is that it does not require an outside server to connect to the client to the host. Although, software companies, like RealVNC provide this feature, at its core, VNC can function on a local area network using TCP/IP protocols.

4.4.6. Remote Desktop Protocol

The Remote Desktop Protocol (RDP) was developed by Microsoft to remotely connect to a remote computer system across a network. Much like VNC, RDP runs over TCP/IP [34] and does not require an external server like Chrome Remote Desktop as long as the host system is on the same network. RDP was

designed for Windows which means if a Windows computer is utilized to remotely connect to the Raspberry Pi using RDP, no additional software will be needed to install on the client.

4.4.7. Conclusion

Although RDP provides optimized bandwidth utilization and only transmits the minimum amount of graphics necessary to keep the client's screen congruent with the host, this works best on Windows devices [35]. VNC just sends complete frames of the host to the client. Although, this may not be optimal, it allows VNC to function consistently on any computer and mobile device. In addition, "VNC uses TCP [which] means small packet loss... RDP chases its packages by UDP" [35]. Ultimately the ability to screen share to multiple viewers and also for each viewer to see connected user's actions prompted us to utilize VNC technology for our system.

4.5. Wireless Module Considerations

Since in order to implement VNC communications between the Raspberry Pi and a client, a network connection must be established. We had previously decided to create a wireless ad-hoc network using the Raspberry Pi. So instead of utilizing a third-party wireless access point, the Raspberry Pi created a network for devices to connect to communicate. Although this reduced any potential errors based on reliance of an external source for connectivity (which may fail or raise connection errors), this posed other potential problems relating to the strength of the network signal broadcasted from the Raspberry Pi. We now had to determine what wireless antenna is strong enough to utilize.

4.5.1. Built-In vs. USB Dongle

The primary area of concern came from forum posts discussing the performance of the include, built-in W-Lan Module, a Broadcom BCM43438. Many Raspberry Pi users have complained about the performance of the the built-in module. It became apparent that Broadcom initially released the BCM43438 along with faulty driver software and firmware [36]. This lead to an initial adverse reaction towards using the preinstalled wifi chip. On top of those issues, one user, clivem, discussed that although it is IEEE 802.11n, it is only a single channel at 20 MHz [36]. Version "n" can reach speeds 6 times faster than version b and provide spread of 3 times the coverage [37]. And because this this was a hardware issue, not one that could be fixed by upgraded firmware or patched driver software. The only way to work around this would be to use an external, USB dongle.

After a quick search on Amazon, many external USB WiFi dongles result. The most compact, "Best Seller" listed started at \$7.99 [37]. A device like this that supported a 150 Mbps data transfer rate due to its 802.11n compliance, would have no problem supporting VNC seamlessly with a far reach. However, one

consideration was that a lower-end dongle like this was \$7.99. Although that is relatively cheap, being on a budget, we rather not spend more if not necessary.

Some users on the official Raspberry Pi forums were reporting a patchy connection from 10 meters and even down to 6 meters [38]. This was initially a concern because of our own issues with limited reception range. However, many users suggested to change region location in through the rasp-config terminal command and disable the WiFi power savings settings by typing “sudo iwconfig wlan0 power off” in the command line to fix this issue [38].

4.5.2. Conclusion

After modifying some of the initialization files for the Broadcom BCM43438 wireless chip, we experimented to find the maximum unimpeded distance we could connect to the Raspberry Pi host. From a Galaxy S7 smartphone, a range of about 40 meters was reached; from a Dell Inspiron 13 notebook computer a range of about 30 meters was reached. For our purposes of hosts an ad-hoc wireless network, we felt this distance was sufficient and decided to reduce costs and not purchase an additional wireless dongle.

4.6. Printed Circuit Board

Printed circuit boards (or PCBs as they are commonly referred to as), are quite possibly the most commonly utilized device in the modern day used to form electrical circuits. PCBs electrically connect the various components of a circuit through the use of pads, traces, and other devices which are components etched onto the copper boards that are the foundations of PCBs. PCBs are generally comprised of a base substrate layer topped with one or more copper layers that contain the traces as well as the grounds, these layers are then topped with a layer of solder mask which in turn isolates the different traces from one another, the designing of these boards can be a very demanding task due to the fact that not only does one have to design the circuit, they also have to make sure that the layout will be able to fit on the board. Printed circuit boards in modern days typically are comprised of multiple layers, typically 8 but some of the more extreme boards are even larger. Traditionally PCBs possessed holes that extended through the entire circuit board that the components were set through and then soldered in place. However due to the recent influx in the development of through hole components now most circuit boards have the components attached directly to the surface. This advancement in components now allows PCBs to have components mounted on both the top and on the bottom thus doubling the effective size of modern PCB circuits. Printed circuit boards are very popular in electronics now days due to the reduction in the amount of possible short circuits and wire failures that were common in circuits implemented in prior years by just jumping wires between components.

4.6.1. PCB Composition

Printed circuit boards are comprised of multiple layers of materials that are combined together utilizing heat as well as adhesives to result in one single complete object. These layers are the substrate, copper layer, solder mask, and the silkscreen. The general layout of these layers are shown in Figure 4.6.1-1.

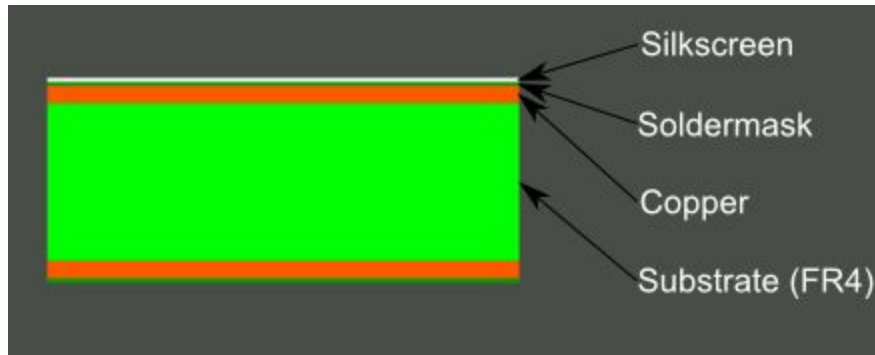


Figure 4.6.1-1. PCB Layers Layout [39]

4.6.1.1. Substrate Layer

The foundation for printed circuit boards are referred to as the substrate. The most common type of material typically utilized in high end printed circuit boards is FR-4 (or FR4). This FR4 designation refers to glass reinforced epoxy laminate material that is used in the fabrication of rods, sheets, tubes, and PCBs. This FR4 material is made of epoxy, bonded with resin, and is highly flame resistant. The letters FR stand for flame retardant and is used to show that the material is compliant with the standard UL94V-0. FR4 has a very high strength to weight ratio and has nearly zero water absorption, this makes this material perfect for the central layer of a printed circuit board since in both humid and dry conditions this material retains both its high electrical insulation and high strength properties. Some of the more cheaply made PCBs utilize epoxies or phenolics as their substrates, these materials are very cheap and one can tell when they are working with this type of substrate due to that fact that during the soldering process these substrates typically leave a fairly bad smell in the air. Another downside to these cheaper substrates is that they are much more flimsy than FR4 and typically have lower coefficients of electrical insulation than FR4.

4.6.1.2. Copper Layer

The first layer that goes on top of the substrate is a copper film. This film is applied generally to both sides of the substrate and boards made in this fashion are referred to as two layer boards, however in cheaper electronics and boards where not as much space is needed for the electrical components the copper layer is only applied to one side of the substrate, these boards are in turn referred to as single layer or one layer boards. When a board is referred to as a two or single layer board this refers to the number of copper layers present on

the printed circuit board, this number of layers can go from one to sixteen or more layers. The thickness of these copper layers vary per circuit board. The copper thickness is generally referred to in weight by ounces per square foot and each ounce per square foot corresponds to approximately 35 micrometers. Most common printed circuit boards have a copper thickness of one ounce per square foot however for more high power printed circuit boards, the thickness of this copper layer can extend to two or even three ounces per square foot. This additional thickness allows higher current to flow through the copper without destroying the printed circuit board.

4.6.1.3. Solder Mask Layer

The layer that goes on top of the copper layer is called the solder mask layer. This is the layer that gives the printed circuit board its trademark typical green coloring, although the green color is not necessary some PCBs have different colors such as red or even yellow. This layer is film of polymer that protects the copper from oxidation as well as protects against unintentional solder bridges from being formed. A solder bridge occurs when two conductors are linked due to excess solder spilling over and combining them together. This layer is typically applied to all printed circuit boards whether they are hand soldered or mass soldered in a factory, however for the later case this solder mask is completely necessary. This is due to the fact that most automatically soldered PCBs use a solder bath technique in which the whole board as well as the component to be attached are dipped in molten solder, thus with no isolation this method can not work. Once this layer is applied, holes must be made to allow the components attached to the board to come into contact with the copper layer. These holes are made utilizing a process called photolithography. While most PCBs are made professionally and it is recommended to have them made this way, one can pretty easily be made at home with a simple copper plate and a solder mask.

4.6.1.4. Silkscreen Layer

The final layer that is applied to the printed circuit board is the silkscreen layer. This layer is typically applied to the component side of the PCB and is used to identify and label multiple things such as logos, components, part numbers, warnings, and manufacturing marks. This allows humans to better understand and construct the full PCB circuit one it is necessary. One of the most common uses of the silkscreen layer is to put manufacturing marks on the PCB, there are no set standards on where these marks should go but they are typically placed in non critical areas of the board to prevent confusion with the consumer. While the most typical color of silkscreens is white, these can be obtained in most colors such as white, black, yellow, red, or even blue.

4.6.2. PCB Terminology

The purpose of this section is to introduce and define some commonly utilized printed circuit board (PCB) terms that will be used in the design section.

- **Annular Ring:** A ring of copper that surrounds a plated through hole in a PCB.
- **DRC:** This stands for design rule check, this is a check performed to make sure there are no errors in the design. Such errors could be incorrectly touching traces, too small of drill holes, or too thin of traces.
- **Drill Hit:** This is a place on the printed circuit board in which a hole should be drilled or in which a hole is drilled.
- **Pad:** This refers to a place on a printed circuit board in which a piece of metal is exposed so that the components can be soldered onto it.
- **Surface Mount:** A method of attaching components to a PCB that does only requires setting the components on the PCB and attaching them with solder.
- **Trace:** A continuous path of copper on a PCB that is used to electrically connect two components together.

4.6.3. Design Recommendations For Better Reliability

Printed circuit boards are extremely crucial components in electrical systems, it is for this reason that they should be designed in such a manner to ensure long lasting reliability in these components. Although every PCB is designed in a different manner there are some general rules that should be followed regardless of PCB application, these will be listed in this section. Most of the rules are good practice and will extend the lifetime of the printed circuit boards other will help to reduce the size of the boards thus making the overall design even more compact which is extremely desirable in most cases.

Rule #1: Keep the traces as short and direct as physically possible

If the traces are longer than absolutely necessary, the additional length will add additional intrinsic resistance and inductance to the board which can reduce the overall performance of the board. These added flaws can have drastic negative effects especially in high speed applications.

Rule #2: Keep related components in same area on the PCB

This is very important due to the multiple benefits achieved by following this rule. The first benefit is that with the components grouped together the traces between the components are in turn shorter which can reduce the negative effects associated with long traces that are outlined in rule #1. Another benefit that this rule provides is that it makes troubleshooting problems that occur on the printed circuit board easier due to the fact that the components are close together which makes it easier to trace through the circuit and find where something went wrong.

The final benefit this rule offers is that with the components grouped so close together when capacitor banks are used to filter out noise they are much more effective than if they are spaced out.

Rule #3: Utilize Power Planes

Utilizing this rule helps to ensure that there are minimal voltage drops from the power supply to the components. It is recommended that multiple supply lines are run throughout the board to ensure proper supply voltage is provided to each the necessary components. Finally for multilayer boards it is recommended to utilize an entire plane as the ground plane to ensure adequate ground return paths.

Rule #4: Utilize decoupling capacitors

Capacitors are extremely inexpensive and should be implemented whenever necessary. Decoupling capacitors shield the on board components from high frequency noise that could introduce errors into the system. Capacitors should be kept within a set range of values in order to lower the complexity of the list of materials necessary to replicate the printed circuit board design.

Rule #5: Utilize heat sinks to lower the thermal resistance of the circuit

Heat sinks are low thermal resistance devices that are attached externally to the circuit to dissipate great amounts of heat. If this heat is not dissipated by external heat sinks the thermal resistance of the circuit will greatly increase. Heat sinks with high areas and external fins are extremely effective in heat dissipation.

Rule #6: Be cautious when using automated signal routing software

Automatic routing functions are available in most CAD PCB design softwares, however these functions can malfunction and route the traces incorrectly thus it is safer to route major traces by hand instead to ensure reliable design implementation. Utilizing the Design Rule Check (DRC) function included in most PCB design softwares will check designs for multiple criteria and should be used as often as possible throughout the design process.

Rule #7: Carefully layout the silkscreen layer

While this is the final layer applied to the printed circuit board, properly implementing this can reduce headaches and rework in the future. Since this layer holds valuable information for the end user such as part numbers as well as warnings, the font and orientation used to display this information should be easily readable.

The rules mentioned above are very good general rules to follow when designing a printed circuit board regardless of purpose. There are many more rules that can be applicable depending on the actual function of the PCB. Following the above rules as well as others more specific to the function of the PCB can greatly improve the reliability as well as lifetime of the printed circuit board.

4.7. Valve Control Considerations

Research into finding a way to control the valves was a very fundamental task for this project. Due to the desired ability of the device users to manipulate the parameters of the different components via the touchscreen display, a functional way to precisely adjust the position of the valves has a need to be implemented. Important considerations taken into account during the research of this phase were reliability, precision, ease of control, as well as output torque.

4.7.1. Motorized Ball Valve

The first method of valve control was to simply utilize motorized ball valves to control the amount of water flow through the pipe. A typical motorized ball valve has three input wires, ground, forward, and reverse. As can be expected the ground wire is always tied directly to ground while if the ball valve is desired to be opened, the forward wire is supplied a set DC voltage and it will proceed to open until the supplied voltage is terminated. The process to close the valve is the same as the opening process except that now the reverse wire is the one that is being supplied the DC voltage.

This method offers a couple of perks. The first benefit offered is that the control of this device is extremely simple, the only necessity is to have a switch connected to each of the input wires (both forward and reverse) and depending on which direction the valve needs to turn, that is the switch that is operated. Another perk to this device is that it offers variable position which is a necessity for this project, all that is needed is to supply an input voltage for an amount of time equal to the degree of opening that is desired, however this timing would have to be determined through intensive experimental testing.

4.7.2. Brushed DC Motor

The second method of valve control considered was the utilization of a simple brushed DC motor to turn the ball valve. A brushed DC motor is a motor specifically designed to operate off a direct current power supply. The theory behind the operation of a DC motor is fairly simple, as current is passed through a coil of wire surrounding a ferromagnetic core the core will become magnetized. In a DC motor the core is surrounded by magnets of opposite polarities, when the core is also magnetized it will be repelled from the magnet with the same polarity causing it to spin. Every half cycle however the current is reversed through the coil thus magnetizing the core to the opposite polarity, this allows the motor to continuously turn between the surrounding magnets, converting the electrical energy input into torque [40].

This method of valve control offers a couple of benefits. One of the advantages of DC motors is that the speed of the motor can be adjusted fairly simply by just

adjusting the input voltage used on the motor, for example if a motor has a base speed of 2000 revolutions per minute provided the supply voltage is 400 VDC, it will run at 1000 revolutions per minute provided a supply voltage of 200 VDC. Another advantage of DC motors is that they hold a fairly constant torque regardless of the speed they are being run at, which is highly beneficial in applications where variable speeds are necessary [41].

In addition to these advantages, there are also a few disadvantages associated with brushed DC motors. One of the major disadvantages for this type of motor is that they lack precision control, meaning that it would be nearly impossible to replicate precise opening for valves given specific input values by the users. Another disadvantage to brushed DC motors is that they tend to have short lifespans as in the majority of these motors components tend to touch and thus wear down as time goes on, facilitating the need for frequent servicing.

4.7.3. Servo Motor

The third method considered to vary the opening of the ball valve was to utilize a servo motor attached directly to the valve. The basic theory behind how a servo motor functions is the same as the way a DC motor operates (refer to section 4.9.2) however a servo motor has circuitry added to it that allows it to function more precisely and with a higher torque than a basic DC motor. The added circuitry that allows more precision to the servo as opposed to the DC motor is a comparison feedback circuit that only sends an output voltage to the motor when the input signal differs from the output signal. When the signals differ the motor will turn and a potentiometer connected to the motor will also turn thus changing the output voltage signal, when this potentiometer output voltage equals the known input voltage signal the difference will be zero and no voltage will be provided to the motor thus it will stop turning. The added torque gained from the servo motor as opposed to a basic DC motor is due to added gears in a servo motor which convert some of the high speed output of a basic DC motor into additional torque instead, thus servo motors have less speed but higher torque than basic DC motors [42].

Servo motors offer various advantages when used in a valve control application. The first advantage of servo motors is their ability to be semi precisely controlled, servo motors are generally used with a PWM (Pulse Width Modulation) input. This enables the device to have a repeatable position for a given input due to the fact that if there is a set pulse width for a given constant, every time that constant is selected that same pulse just needs to be supplied. An error arises here however due to the fact that it is impossible to achieve perfect PWM since a square wave cannot realistically be achieved in practice, this will yield a very slight variance every time the same pulse is attempted to be supplied. Another advantage of servo motors is that they are fairly simple to control, all that is really

needed to control the motor is a microcontroller with PWM output pins to utilize as inputs to the motor. [42]

While there are multiple advantages to the use of servo motors in valve control applications, there are also a few disadvantages associated with them. The first disadvantage is the maintenance required with them, as the brushes utilized in the motors wear down as they are used, they need to be replaced. This is not ideal in our application due to the inaccessible nature of the motors after they are installed. Another disadvantage to using servo motors for this application is they have poor thermal performance. This poor thermal performance is due to the fact that the heat from the motor is generated internally within the motor and there is very little heat dissipation from the inside of the motor to the outside, also due to the restricted space in our device next to the valve control system there is no room for additional devices to dissipate the heat given off by the motor. [43]

4.7.4. Stepper Motor

The final method of valve control considered was to use a stepper motor connected to the ball valve to vary the opening. Stepper motors vary from servo motors in the fact that they are an open loop system while servo motors are a closed loop system. This simply means that a stepper motor typically does not possess a feedback sensor to confirm that it is in the desired position, the reason why this is not required in a stepper motor is because the motor moves in specified “steps” every time it moves, these steps can generally range from 0.75 to 90 degrees per step depending on which stepper motor is being used. These steps are achieved due to the stator having teeth equally spaced around the perimeter that the inner rotor moves between. There are two distinct types of stepper motors, bipolar and unipolar. [44]

4.7.4.1. Unipolar Stepper Motor

Unipolar stepper motors are generally used in applications where the user does not want to implement a complicated driving circuit or the user does not need a high amount of torque. Unipolar stepper motors are simple to drive due to the fact that there is no need to implement an H-bridge to switch the direction of the current flow through the motor to change the magnetic field. The H-bridge does not need to be used because there are generally three wires per winding on a unipolar stepper motor, two of these wires are simply different ends to the same wire while the third is connected to the middle of the other wire. This third wire is always kept at a high voltage and when current is desired to flow a certain direction, one of the ends of the other wire is then connected directly to ground, if the current is desired to flow the other direction then the other end of the wire is connected to ground instead. This configuration results in low torque however due to the fact that only half of the phase has current flowing through it at any given time resulting in a weaker created magnetic field than if the whole phase had current flowing through it [45].

4.7.4.2. Bipolar Stepper Motor

Bipolar stepper motors are generally implemented in applications where the user desires a higher torque than what can be provided with a unipolar stepper motor, however this higher torque comes at the cost of a need to utilize a more complicated driving circuit than those used for unipolar stepper motors. While a unipolar stepper generally has three leads per winding phase due to the center tap wire on the winding, a bipolar stepper motor generally only has two leads per winding phase. These two leads are just two ends to the same wire, this makes the use of a H-bridge necessary to operate these types of motors due to the necessity of the current direction to be able to be reversed so that the magnetic field can change. This configuration results in a higher torque than that of the unipolar stepper motors since in bipolar stepper motors the whole phase has current flowing through it as it is operated as opposed to the unipolar configuration which only has current flowing through half of the phase at a time [45].

4.7.5. Conclusion

After the creation of Table 4.7.5-1 it was clear to the group what method of valve control would be the best for our application. Right away we eliminated the possibility of using either brushed DC motors or motorized ball valves for our application, this is due to the fact that even though the ease of control of both of these options is very easy the precision of these choices is very low which makes them impossible to implement into our project. Next the debate arose between utilizing servo or stepper motors for the valve control. Servo motors were eventually eliminated because even though they are extremely easy to control and they have high precision as well as high output torque, they have a tendency to need to be serviced more frequently than stepper motors due to the wearing out of the brushes in the motor when they are used. Finally stepper motors were settled on as to be our method of controlling the valves in our project, but now we had to choose between utilizing unipolar stepper motors or bipolar stepper motors. Initially unipolar stepper motors were the most appealing option since the only difference between them and bipolar stepper motors was that the output torque is lower and they are easier to operate. However it was soon discovered that since the valves being used in our project required two Newton meters of torque to operate a unipolar stepper motor would not be able to provide the necessary output. Therefore even though the bipolar stepper motors are more complex to operate, due to their high output torque, high precision, as well as high reliability they were eventually chosen as our means to control our valves.

Table 4.7.5-1. Valve Control Options Characteristics

	Reliability	Precision	Ease of Control	Torque
Motorized Ball Valve	High	Low	Easy	High
Brushed DC Motor	Low	Low	Easy	Low
Servo Motor	Low	High	Easy	High
Unipolar Stepper Motor	High	High	Easy	Low
Bipolar Stepper Motor	High	High	Hard	High

4.8. Motor Considerations

As stated in section 4.7.5 due to the necessary requirements of the valve control motor to have high torque and high precision, a bipolar stepper motor was the one chosen for this project. However after the determination of the type of motor needed to be used, an additional important decision had to be made and that is which of the hundreds of bipolar stepper motors available on the market should be used. The important quantities considered when comparing the different bipolar stepper motors were cost, holding torque, size, and precision.

4.8.1. Bipolar Stepper Motor 1

The first bipolar stepper motor considered is a Nema 23 bipolar stepper motor. This particular version of this stepper motor has a holding torque of 1.9 Nm which is undesirable due to the valve that needs to be turned having a turning torque of 2 Nm. This stepper motor possesses a step angle of 1.8 degrees or 200 steps for a full revolution, this is desirable due to the wide variety of positions this would allow the valve to be opened to. However the overall size of this bipolar stepper motor is quite large due to its dimensions being 57mm X 57mm X 76mm, this is undesirable due to the low amount of available space where the motor would be placed. The overall cost of this motor is approximately thirty six dollars [46].

4.8.2. Bipolar Stepper Motor 2

The second bipolar stepper motor considered is also a Nema 23 bipolar stepper motor. This particular version of this stepper motor has a holding torque of 1.26 Nm which is undesirable due to the valve that needs to be turned having a turning torque of 2 Nm. This stepper motor also possesses a step angle of 1.8 degrees or 200 steps for a full revolution, this is desirable due to the wide variety

of positions this would allow the valve to be opened to. The overall size of this bipolar stepper motor is very small due to its dimensions being 56mm X 56mm X 56mm, this is desirable due to the low amount of available space where the motor would be placed. The overall cost of this motor is approximately twenty six dollars [47].

4.8.3. Bipolar Stepper Motor 3

The third bipolar stepper motor considered is a simple geared bipolar stepper motor which was found on robotshop.com with product code RB-Phi-210. This particular version of stepper motor has a massive holding torque of 2.94 Nm which is highly desirable due to the valve that needs to be turned having a turning torque of 2 Nm. This stepper motor also possesses a step angle of 0.067 degrees or 5373 steps for a full revolution, this is very highly desirable due to the wide variety of positions this would allow the valve to be opened to. However the overall size of this bipolar stepper motor is quite large due to its dimensions being 42mm X 42mm X 94mm, this is highly undesirable due to the low amount of available space where the motor would be placed. The overall cost of this motor is approximately forty four dollars [48].

4.8.4. Bipolar Stepper Motor 4

The final bipolar stepper motor considered is a simple geared bipolar stepper motor which was found on Jameco.com with part number 2158531 . This particular version of stepper motor has a very small holding torque of 0.31 Nm which is highly undesirable due to the valve that needs to be turned having a turning torque of 2 Nm. This stepper motor also possesses a step angle of 1.8 degrees or 200 steps for a full revolution, this is very highly desirable due to the wide variety of positions this would allow the valve to be opened to. The overall size of this bipolar stepper motor is quite small with its dimensions being 42mm X 42mm X 38mm, this is highly desirable due to the low amount of available space where the motor would be placed. The overall cost of this motor is approximately six dollars [49].

4.8.5. Conclusion

After the creation of Table 4.8.5.1 it was easier to decide which bipolar stepper motor should be implemented in our project to open/close the valves. By observing the third column where the holding torques of the different motors are listed, bipolar stepper motor 4 could easily be eliminated due to its holding torque being nowhere near the 2 Nm torque required to operate the valve. While bipolar stepper motors 1 and 2 both also do not have the 2 Nm torque required to turn the valve, with the integration of gears attached to both the valve and stepper motor the required torque to turn the valve drops to under 1.2 Nm. Next by observing the first column, bipolar stepper motor 3 can be eliminated, this is due to the height of the motor at 94 mm being too large to fit in the limited space

available for our motor that controls the valves. Finally, since the precision of bipolar stepper motors 1 and 2 are both the same, the final eliminating factor was cost of the motor. Since bipolar stepper motor 2 is 10 dollars cheaper per motor and has both the required torque to turn the valve as well as small enough dimensions to fit in the space available, bipolar stepper motor 2 was the one decided on for our valve controlling operation, this motor is shown below in Figure 4.8.5-1.

Table 4.8.5-1. Motor Consideration Characteristics

	Size (L,W,H) (mm)	Precision (Degrees per Step)	Torque (Nm)	Cost (US Dollars)
Bipolar Stepper Motor 1	57,57,76	1.8	1.9	36
Bipolar Stepper Motor 2	56,56,56	1.8	1.26	26
Bipolar Stepper Motor 3	42,42,94	0.067	2.94	44
Bipolar Stepper Motor 4	42,42,38	1.82	0.31	6

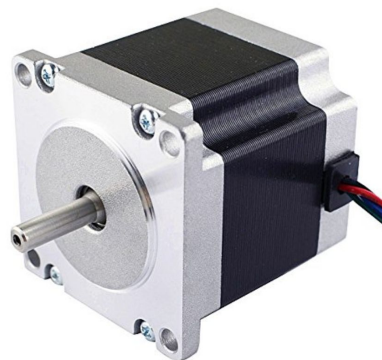


Figure 4.8.5-1: Nema 23 CNC Stepper Motor

4.9. Driver Considerations

As stated in section 4.8.5 due to the high precision, high output torque, small size, and low cost the bipolar stepper motor determined to be used for our project is a Nema 23 bipolar stepper motor. However after the determination of the model of bipolar stepper motor needed to be used, an additional important decision had to be made and that is which of the many bipolar stepper motor

drivers available today on the open market should be used to operate ours. The important quantities considered when comparing the different bipolar stepper motor drivers were cost, input voltage, size, and output current.

4.9.1. Stepper Motor Driver 1

The first bipolar stepper motor driver considered was a simple two phase microstepping stepper motor driver available from circuitspecialists.com with item number CW230. This particular bipolar stepper motor driver has a fairly narrow input voltage range of twenty four to thirty six volts DC, this is undesirable due to the fact that this would require another power supply rail to be added to the power PCB just to accommodate this high input voltage. The output current of this bipolar stepper motor driver can be varied from nine hundred milliamps all the way to three amps, this is desirable due to the fact that the bipolar stepper motor chosen to turn our valves has a necessary input current of 2.8 amps. This bipolar stepper motor driver has a fairly large size shown by its dimensions being 115mm X 72mm X 32mm, this is undesirable due to the low space available for the housing of the bipolar stepper motor driver. The overall cost of this bipolar stepper motor driver is approximately forty dollars [50].

4.9.2. Stepper Motor Driver 2

The second bipolar stepper motor driver considered was a simple single bipolar stepper motor driver available from robotshop.com with item number RB-Phi-345. This particular bipolar stepper motor driver has a fairly wide input voltage range of ten to thirty volts DC, this is desirable due to the fact that the power PCB already has a voltage supply rail of twelve volts so this driver can be integrated with no change to the PCB design. The maximum output current of this bipolar stepper motor driver is four amps, this is desirable due to the fact that the bipolar stepper motor chosen to turn our valves has a necessary input current of 2.8 amps. This bipolar stepper motor driver has a fairly small size shown by its dimensions being 76.2mm X 63.5mm X 25mm, this is desirable due to the low space available for the housing of the bipolar stepper motor driver. The overall cost of this bipolar stepper motor driver is approximately seventy five dollars [51].

4.9.3. Stepper Motor Driver 3

The third bipolar stepper motor driver considered was a SMAKN® TB6600 stepper motor driver. This particular bipolar stepper motor driver has a fairly wide input voltage range of nine to forty two volts DC, this is desirable due to the fact that the power PCB already has a voltage supply rail of twelve volts so this driver can be integrated with no change to the PCB design. The maximum output current of this bipolar stepper motor driver is four amps, this is desirable due to the fact that the bipolar stepper motor chosen to turn our valves has a necessary input current of 2.8 amps. This bipolar stepper motor driver has a fairly small size shown by its dimensions being 96mm X 71mm X 37mm, this is desirable due to

the low space available for the housing of the bipolar stepper motor driver. The overall cost of this bipolar stepper motor driver is approximately fifteen dollars [52].

4.9.4. Stepper Motor Driver 4

The fourth bipolar stepper motor driver considered was a simple single bipolar stepper motor driver available from robotshop.com with item number RB-Pol-176. This particular bipolar stepper motor driver has a fairly wide input voltage range of eight to thirty five volts DC, this is desirable due to the fact that the power PCB already has a voltage supply rail of twelve volts so this driver can be integrated with no change to the PCB design. The maximum output current of this bipolar stepper motor driver is two amps, this is undesirable due to the fact that the bipolar stepper motor chosen to turn our valves has a necessary input current of 2.8 amps. This bipolar stepper motor driver has a fairly small size shown by its dimensions being 20.32mm X 15.24mm X 31mm, this is desirable due to the low space available for the housing of the bipolar stepper motor driver. The overall cost of this bipolar stepper motor driver is approximately six dollars [53].

4.9.5. Stepper Motor Driver 5

The final bipolar stepper motor driver considered was a dual bipolar stepper motor driver available with the ability to control two bipolar stepper motors simultaneously from robotshop.com with item number RB-Dfr-539. This particular bipolar stepper motor driver has a fairly wide input voltage range of eight point two to forty five volts DC, this is desirable due to the fact that the power PCB already has a voltage supply rail of twelve volts so this driver can be integrated with no change to the PCB design. The maximum output current of this bipolar stepper motor driver is 1.6 amps, this is undesirable due to the fact that the bipolar stepper motor chosen to turn our valves has a necessary input current of 2.8 amps. This bipolar stepper motor driver has a fairly small size shown by its dimensions being 83mm X 55mm X 25mm, this is desirable due to the low space available for the housing of the bipolar stepper motor driver. The overall cost of this bipolar stepper motor driver is approximately nineteen dollars [54].

4.9.6. Conclusion

After the creation of Table 4.9.6-1 it was easier to decide which bipolar stepper motor driver should be implemented in our project to control the stepper motors. By observing columns number one and three in the table where the required range of input voltages to the stepper motor drivers as well as the size of the stepper motor drivers are listed, stepper motor driver one could easily be eliminated due to the necessity of adapting the PCB design to provide an input voltage rail high enough to act as input to the stepper motor driver and its size being very large compared to the other drivers considered. By referring to column two of the table where the output currents of the stepper motor drivers are listed

it can be seen that stepper motor drivers four and five can both be eliminated, this is due to the fact that the bipolar stepper motor we chose to turn our valves has a necessary 2.8 amps input current while stepper motor drivers four and five can only supply a max output current of 2 and 1.6 amps respectively, not nearly enough to power our motor. Stepper motors two and three both have high output currents, input voltage ranges that include the readily available voltages from the currently designed power PCB, as well as reasonably similar dimensions. However by referring to column four where the price of the different stepper motor drivers are listed it can be seen that since stepper motor driver two costs approximately five times as much as stepper motor driver three, this stepper motor driver was eliminated as an option. Due to the high output current, small dimensions, acceptable input voltage, as well as low price bipolar stepper motor driver three was the one decided on to drive the stepper motors that control the valves in our project, this stepper motor driver is pictured below in Figure 4.9.6-1.

Table 4.9.6-1. Stepper Driver Consideration Characteristics

	Input Voltage (V)	Output Current (A)	Size (L,W,H) (mm)	Cost (Dollars)
Stepper Motor Driver 1	24-36	0.9-3	115 X 72 X 32	40
Stepper Motor Driver 2	10-30	4	76.2 X 63.5 X 25	75
Stepper Motor Driver 3	9-42	4	96 X 71 X 37	15
Stepper Motor Driver 4	8-35	2	20.32 X 15.24 X 31	6
Stepper Motor Driver 5	8.2-45	1.6	83 X 55 X 25	19



Figure 4.9.6-1: SMAKN® TB6600 Motor Driver

4.10. Light Emitting Diodes Strip

Light Emitting Diodes (LEDs) strips were considered to be able to incorporate certain functions and capabilities to the project. LEDs are used to indicate more obviously the increase/decrease in the current passing through various components by increasing and decreasing the speed in which the LEDs flash. RGB (Red, Green, Blue) strips were necessary to facilitate visualization of the different branches that contribute to the main current and to demonstrate Kirchoff's Current Law. Various requirements were taken into account when choosing the LEDs strip that are able to fully implement the project's necessary functionalities. These include size, flexibility, microcontroller compatibility, power consumption, and water resistance.

4.10.1. Analog and Digital Strips

This section discusses the different types for LED strips that were considered for this project. Analog and digital LED strips have different uses and qualities which are analyzed below to arrive to the most suitable decision based on the project's requirements.

4.10.1.1. Analog LED Strips

Analog LED strips have every LED connected in a parallel manner. Due to the way they are connected, they are unable to be addressed individually, which means that if a specific color is set or they are set to blink at a specific pattern, the settings will affect the entirety of the LED strip as long as the power supply allows it. Due to the serial connection between the red, green and blue LED, these have to be powered by a 12V DC supply. Each segment composed of three LEDs draws a total of 60mA, assuming all three LEDs are on [55]. The more segments there are on the strip, the more current will need to be drawn from the power supply. The overall current can be determined by multiplying the length of the strip, the number of LEDs per length of the strip, and the amount of current drawn per LED (1 segment = 3 LEDs, 60mA/segment, 20mA per LED). It is recommended to use Pulse Width Modulation (PWM) dimming techniques to achieve higher efficiency and control the strip (low power consumption), additional circuitry is needed to connect the microcontroller to the LED strip. NPN transistors are used to be able to sink to ground the required amperage per LED and connect to the microcontroller PWM output pins. [55]

Analog LED strips are most commonly used for lighting purposes in building or houses, and integrating higher voltage components [56]. Although, they are much more cost effective than digital LED strips, analog LED strips do not allow for individually addressing segments on the LEDs and therefore do not meet the project's requirements.

4.10.1.2. Digital LED Strips

Digital LED strips are individually addressable, thus each segment has an individual chip. This quality allows for each LED to be any color and flash at any rate at any time, they are independent of each other. Additionally, due to the IC built in every LED the strip is controlled differently than the analog strip. Instead of having four connections, three to control each individual RGB and one to connect to the power supply; the digital LED strips usually have three: GND, DI (data in), Power [57]. A power supply of 5V DC is necessary to power the LEDs and each segment is composed of the same RGB LEDs thus, each segment draws 60mA (0.3W), for a one meter strip containing 30 segments per meter, the total current drawn is 1.8A/meter, 9W/meter. For the two simple facts that digital strips are individually addressable and require no additional circuitry to be connected to the microcontroller, this was the obvious choice.

4.10.2. Chip Size

There are various sizes of digital LED strips: 3528, 5050, 5630. These numbers refer to the dimensions of the diodes, for example 3520 refers to 3.5mm x 2.8 mm chip size [58]. The greater the chip size the higher the lumen output, the higher the power consumption. Lumen output of 3528 is 6 to 8 lumens, for the 5050 is 16-22 lumens and for 5630 is 45-50 lumens. For this specific project the lumen output of the the LEDs is not of great impact since, the objective is not to light up a whole area but simply demonstrate a pattern based on the programming of the lights, therefore 5630 was discarded. Chip size 5050 was selected on top of the 3528 due to its easy availability in the market and meeting all the previous listed requirements.

4.10.3. Flexibility

The flexibility of the LED strips is very important when it comes to this project. Due to the various pipe turns and lengths, it was necessary to identify a strip that could be easily bent and altered. There are many individually addressable strips in the market that are able to be cut, elongated and bent to the project's needs.

4.10.4. Water Proof

Due to the nature of the project, it is very important that all the electrical components are protected against any water damage. Therefore, this must be a built in characteristic of the chosen LED. Silicone weatherproof sheathing with minimum Ingress Protection (IP) 66 preferable IP67, protecting from total dust ingress and high pressure water jets/immersion between 15cm to 1m in depth. [18]

4.10.5. Rayonn FLB6

The Rayonn FLB6 is a waterproof LED strip operating at a 12V DC voltage. The led strip features a 5060 Top SMD LEDs at a beam angle of 120 degrees. The LEDs are not individually addressable instead they addressable every 3 LEDs. 3 LEDs draw 60mA of current per section. The thickness of the led strip is 3.5mm and come in a length of 5 meters. The led features a long life and is cuttable to a desired length.

4.10.6. Adafruit LPD8806

The adafruit LPD8806 drives off a 12V voltages source with a driving current of 18mA. It features a two-wire control mode with a shift clock up to 20 MHz. The circuit design supports PWM on each channel with a 1024 grayscale effect. The led strip uses industrial grade CMOS process, this provides multi-channel constant output 3 channel and 4 channel. The signal allows for multi-gray full-color lighting. [59]

4.10.7. ALITOVE WS2812B

The Alitove WS2812B is an individually addressable LED strip. The led strip features 5050 components which allows for a clear display for individual LEDs. The led strip is capable of transmission of a signal with a distance of 5 meters without any increase circuit. This is crucial to allow for transmission of the “blink” function through a branch in the system. The transfer protocol used for this led strip is a single NZR communication mode. The led strip is capable of full-color mode. It is waterproof and is capable of supporting all of the purposes of the system. [57]

4.10.8. Conclusion

ALITOVE WS2812B Individually Addressable LED Strip Light 5050 RGB SMD 150 pixels Dream Color Waterproof IP66 Black PCB 5V DC was chosen (See Fig. 4.10.5-1). This specific strip meets all the project requirements and is fully compatible with Arduino and Raspberry Pi. It is readily available which made it easy to communicate with the supplier to obtain product specifications. This strip has 30 segments per meter, 24 bit full color, operating voltage of 5V DC, built-in IC short circuit protection, and signal reshaping circuit (ensure distortion of the wave does not accumulate), and consumes 0.3W per LED. [57]



Figure 4.10.5-1: ALITOVE WS2812B Individually Addressable LED Strip

4.11. Mechanical Components Considerations

This section discusses the various mechanical components that were considered for the project. These components include: valves, pumps, flow meters, and pressure sensors. Several options from each one of the components were analyzed and a decision was made based on the qualities that fit the project requirements and constraints best.

4.11.1. Valves

Valves are used in our project to be able to regulate pressure and represent various circuit components such as resistors and diodes. Since the valves are operated using a bipolar stepper motor it was essential to choose one with the lowest operating torque. Due to the established dimensions of the PVC pipes the valves must be of 1.27 cm ($\frac{1}{2}$ ") diameter and of female connection to fit perfectly into the pipe without the need of a reducer. The following two types of valves were considered when studying the ways of operation.

4.11.1.1. Brass Ball Valve

Brass ball valves are usually threaded and have a one sided handle (lever) fixed to the valve by means of a screw. Everbilt Lead Free Threaded FPT x FPT Ball Valve has the following dimensions: width of 11.7cm, 5.7 cm of length and height of 3.8cm. The valve weights about 0.2 Kg and has a maximum pressure of about 4137 kPa which exceeds our system maximum pressure by XX kPa. Torque information was unavailable in the product documentation, the only indication that it possessed a low operating torque was in the product description without any specific torque value.

Since this valve is easily available in hardware stores it was easy to purchase and test. It was tested using the Bipolar Stepper Motor 2 which has a 1.26 Nm operating torque. Gages were designed to achieve a higher torque through the motor; one was attached to the lever of the valve and the other to the rotor of the

motor. No matter the gear ratio the motor was unable to move the valve. Furthermore, since the valve is threaded additional adapters are needed to connect the valve to the PVC pipes. Due to the much higher operating torque needed to turn the valve and additional adapters which would increase the cost of the project, the brass valve was ruled out.

4.11.1.2. True Union Ball Valve

True union PVC ball valves have a maximum working pressure 1034k Pa, about four times less than the brass ball valve, but still meets the requirements of the project. The true union ball valve has the following dimensions: width of 5.4 cm, 10.5 cm of length and height of 8 cm. This valve has detailed specifications containing information of the operating torque which is 2 Nm. [60] Due to the motor's operating torque plus the amplification provided by the use of gears, the motor was able to easily move the valve in either direction. True union ball valve is easy to take apart and put back together which made the gear design very simple for the slot where the center of the gear sits. This made it easy to test the operation of the motor and the valve. (See figure 4.11.1.2-1)

The price from the brass valve and the true union valve are very close together. Also, since the valve is not threaded there is no need to purchase adapters which reduces the cost slightly. This type of valves are not as easily available as the brass valves. They are very limited and only distributed by two suppliers. Therefore, when testing was done and verified that the valve met the project requirements, all the necessary valves were ordered.



Figure 4.11.1.2-1: True Union Ball Valve

4.11.2. Pump Consideration

The “neutral ground” is represented by the use of a tank where all the water supplied to the system is stored. A main pipe is connected to the tank and used to distribute the water through the components in the system and then is then drained back to the tank. Due to this a submersible pump is necessary to transport all the water from the tank up to the main line. The length of the main pipe from the tank to the top is about 2.4m (8ft), therefore it is required that the pump is able to have enough force to pump the water to the top. It is also necessary for the pump to be able to build enough pressure to acquire

measurements to represent the voltage drops through the system. Research was conducted in how to choose the necessary pump and perform the right calculations.

To be able to select a pump that met with the above characteristics the volumetric flow rate of the pump and losses through the pipe play a significant role. The volumetric flow rate (F) is depending of the average velocity (v_{avg}) the water is pumped and the cross sectional area of the pipe ($A_c = \frac{\pi D^2}{4}$).

$$F = v_{avg} A_c$$

The power input needed to overcome frictional losses in the flow due to viscosity of the water is determined by the following equation:

$$W_{pump} = F * \Delta P$$

where W_{pump} is the power input of the pump (W) and ΔP (Pa) is the pressure drop through the pipe due to either frictional losses or head losses. [61] To be able to calculate pressure losses, one must consider Bernoulli's Equation:

$$P_1 + \frac{1}{2}\rho v_1^2 + \rho g h_1 = P_2 + \frac{1}{2}\rho v_2^2 + \rho g h_2$$

where ρ is the density of water ($\frac{1g}{cm^3}$), g is the acceleration of gravity ($9.8 \frac{m}{s^2}$), h is the height (m) of where the pressure point lays. Pressure drop due to head losses (Pa) can be calculated using the following equation:

$$h_L = f \frac{L}{2D} \rho v_{avg}^2$$

where $f = \frac{64}{Re}$ (friction factor), Re is Reynold's number, obtained from the following equation:

$$Re = \frac{v_{avg} D \rho}{\mu}$$

μ is the dynamic viscosity of water which at 20 degrees celsius is 1.002×10^{-3} Pas. Based on these equations it was determined that a flow rate of $525.75 \frac{cm^3}{s}$ was enough to overcome the losses and maintain a pressure of about 103.4kPa (15 psi) at the top of the pipe.

A pump meeting these requirements was purchased (500 GPH Fountain Pump). Testing was performed to verify the calculated values. The pump was placed inside the tank, a 2 m horizontal pipe was used to transport the water from the tank to the top horizontal pipe, and measurements were taken at various points using pressure gauges. The pressure at the top pipe was measured to be 0Pa (0 PSI), which is completely different to what was calculated. The flow rate was measured with a flow meter to be $221 \frac{cm^3}{s}$ which is 42% less than the maximum flow listed in the specifications. Further research determined that the height of which the pump is placed plays a significant role in its maximum flow output. Therefore, this percentage drop needed to be considered when determining the pump. A standard 1200 GPH ($1261.80 \frac{cm^3}{s}$) pump was chosen to be able to obtain a desired maximum flow rate of $529.96 \frac{cm^3}{s}$. See Figure 4.11.2-1.



Figure 4.11.2-1: Aqua Pulse Koi Pond Pump

Aqua Pulse Koi Pond Pump AP1200 was the pump of choice, with a maximum head height of about 5.2 m (17 ft), which is 2.4 times bigger than the maximum head height of the 500 GPH Fountain Pump (2.1m, 7ft) [62]. Although they both possess the same voltage rating (120V), their wattage differ by 70W, due to the 500 GPH having a current of 1A and 1200 GPH 1.58A [55]. This will affect the overall power consumption of the system.

4.11.3. Flow Meters

Flow meters are necessary components in our project to be able to demonstrate the value of the flow of current through each one of the branches of the circuit representation. The value of the flow rate is only for demonstration purposes, therefore there was no need to find a flow meter that could provide an output. Due to this and market availability, flow meters with batteries were considered.

4.11.3.1. P3 P0550 Water Meter

P0550 Water Meter is a compacted low cost water meter with a minimum flow rate of $32.81 \frac{cm^3}{s}$ (0.52 gal/min). It has a maximum operating pressure of about 800 kPa (116 PSI), an accuracy of $\pm 1.3\%$, and is battery powered [63]. The minimum flow rate as well as operating pressure are more than sufficient for the project requirements. However, even though this flow meter has an LCD screen displaying the total water flow and volume accumulation at specific times, it does not display the flow rate. For this reason this flow meter was not suitable for our application.

4.11.3.2. Fill-Rite In-Line Digital Meter

Since we were only interested in the flow rate, it was important to choose a flow meter that was able to display this value. The Fill-Rite In-line Digital Flow Meter meets all the requirements. It has a maximum operating pressure of 999.7 kPa (145 PSI), flow rate in the range of $189.3-1640.4 \frac{cm^3}{s}$ (3-26 GPM), and an

accuracy of $\pm 1\%$. The LCD display has many features such as displaying battery condition, calibration mode, rate of flow and total flow along with the unit of measurement. It is important to perform calibration on this product to eliminate as much error as possible in the readings. This is done by determining the correction factor using the following equation:

$$CF_p = CF_C * \frac{\text{Actual Value}}{\text{Display Value}}$$

where CF_p is the proper correction factor and CF_C is the current correction factor.

This flow meter has an inlet/outlet of 2.54 cm (1 in), since the diameter of the PVC pipes is of 1.27 cm (0.5 in) two 1 in to 0.5 in NPT adaptors per flow meter are needed. This flow meter is much more expensive than the P0550, with a price difference of about \$68 however, this was the most economic flow meter available that met the project requirements. These characteristics indicate that the Fill-Rite Flow Meter was the most suitable option for this project. (See the figure below 4.11.3.2-1) [64]



Figure 4.11.3.2-1: Fill-Rite Digital Meter

4.11.4. Pressure Sensing

Pressure sensing is one of the most important aspects in this project. The correlation of pressure to voltage make it the driver component in the system. Voltage differences (pressure drops) are what drives components such as transistors and diodes, therefore it was important to research the various ways water pressure can be affected to fully understand how to represent each circuit components in the system. To have some control over the pressure differences it was important to obtain accurate pressure readings to be able to manipulate the valve openings through the use of the motor. To achieve this, several options were considered such as pressure gauges and pressure transducers. This section gives an overview of these options.

4.11.4.1. Pressure Gauges

A pressure gauge is an instrument used to measure the internal pressure of a system. Trerice Pressure gauges are constructed with a tube sensing element. When pressure is applied to the tubing it flexes and results in motion of the dial face pointer. The pressure medium is the composition of the mass being measured, air, water, steam, etc. Because the system will be measuring water pressure which is a non corrosive media, a bronze or brass bourdon tube and brass socket will be applicable. The socket connection is the port which the medium will enter. Pressure gauges usual contain a male socket connection, but it is possible for them to be female. When considering a pressure gauge one of the most important characteristics to consider is accuracy. Accuracy is defined as a percentage + or - of the scale range where the gauge will return an accurate reading with within that percentage of accuracy. The second characteristic to consider is range of the dial. The range of the dial will show the maximum operating pressure, as well as the specificity of the gauge. It is important to not overwork the pressure gauge, the max anticipated pressure of the system should not exceed 75% of the measurement range as to not damage the gauge itself. Environmental conditions can impact performance of the pressure gauge, especially waterproofing concepts. [65]

4.11.4.1.1. Analog vs. Digital Pressure Gauge

Digital pressure gauges offer many advantages over analog gauges. Digital pressure gauges are capable of providing a steady reading in high vibration situations, where as analog gauges can have difficulty in this area due to fluctuation in pressure. Digital pressure gauges are easy to read, there is no estimation on where the pressure lies, instead the exact pressure is displayed on the LCD. The readability of digital pressure gauges removes parallax error caused by angle of view which can cause issues in calculations. Many digital gauges also come with calibration tools to adjust errors in readings, where analog gauges do not support this. The most enticing feature of digital pressure gauges is their level of accuracy when compared to analog gauges. Analog gauges are limited by the human eye where digital pressure gauges are not.

While digital pressure gauges may seem like the more advanced option, there are still many drawbacks to their implementation. Due to the way digital pressure gauges are implemented they are hindered by a refresh rate. This causes for an average pressure to be displayed over a time period, or just the value which happens to be provided when the refresh interrupt is triggered. This prevents the reading of oscillating values on digital pressure gauges. Analog gauges also allow the operator to view trending patterns such as a rising in the pressure value. By far the biggest advantage to analog pressure gauges versus digital pressure is the lack of power needed to be provided to an analog gauge. Digital pressure gauges need to be powered at all time to display a pressure, whereas when analog pressure sensors are installed they will continue to function until

complete failure of the gauge itself, no need for batteries or an external power source. [66]

4.11.4.1.2.Ashcroft DG25 Digital Pressure Gauge

The Ashcroft features a large LCD display with nine engineering units. The unit also features a bar graph for a specified range to help eliminate the stereotypical conclusion that digital gauges struggle with viewability for trends. The gauge is capable of an accuracy of + or - 0.5 percent. The gauge is capable of handling a range from 0 through 25,000 PSI. The case is IP67 weatherproof, and the wetted materials consist of laser welded stainless steel. Upon consideration of this pressure gauge it was deemed extremely over engineered for our application. The system would not use even one percent of the measurable PSI. The price point also being considered \$250 was way more than other options which were considered. [67]

4.11.4.1.3.Dwyer Instruments DPGA-04

The Dwyer Instruments DPGA-04 is a digital pressure gauge which supports liquid pressure. It is capable of recording pressures up to 5 PSI. It requires a 9 volt alkaline battery to power the device and is accurate to + or - one percent. It has DPGA silicon sensor as its wetted material and comes in at a price point of \$75.00. Due to the limited range of the considered pressure gauge it was deemed as not powerful enough for the system. On top of the low pressure requirements of the gauge it came in at a high price point even though considered to be cheap for digital pressure gauges. The 9 volt battery was also a negative toward this gauge as it would require maintenance.

4.11.4.1.4.Noshok 40-911-800 PSI NPT

The Noshok pressure gauge is an analog pressure gauge which is capable of measuring liquid pressure. It is rated for up to 800 PSI and is a top mounted 1/4th inch port. The pressure gauge is accurate to + or - 3 percent over the range of the gauge. The wetted parts consist of brass and copper weldings. The Noshok pressure gauge has a price point of \$75 which when considered compared to other options came in high for an analog gauge. Due to 800 PSI being supported for this gauge, it was deemed as over engineered for the required application. [68]

4.11.4.1.5.Winters PEM Steel Dual Scale Economy Pressure Gauge

The Winters PEM steel dual scale economy pressure gauge is an economic pressure gauge with a 1/4th inch bottom mount. The pressure gauge has a dial display output in PSI and kPa. The gauge is rated for up to 15 PSI or 100 kPa. It is capable of measuring liquid pressure as well as gas pressure due to it being composed of brass wetted parts. The gauge is accurate to + or - three percent

over the entire range of the gauge. The cost of this gauge is \$12.15, which makes this the most cost effective choice considered. [69]

4.11.4.1.6. Conclusion

In conclusion, the Winters PEM Steel Dual Scale Economy Pressure Gauge seemed to be the best option. Upon further evaluation the pressure gauges as a whole were deemed unnecessary due to the desire to incorporate feedback using pressure transducers into the system. Feedback was considered necessary due to the work in plotting each and every combination of pressure across different branches of the system. With feedback implemented the system would be able to receive information on pressure from all locations and make adjustments to the valves to reflect the desired pressure drop. To still provide the user with the visual representation of the pressure, the values recorded from feedback will be displayed on the User Interface in the proper location. This allows for the system to incorporate the best of both analog and digital pressure gauges. Using feedback it is possible to display a clear concise value for the user to read while maintaining the specificity of exact measurements of the analog pressure gauges.

4.11.4.2. Pressure Transducers

Pressure transducers are devices that convert pressure to an analog electrical signal. These are very useful for electrical and electronics application. There are different types of transducers available in the market capacitive and piezoresistive. These transducers have various ways of measuring pressure such as absolute, differential and gage. This section discusses the various types of pressure transducers and which met the criteria for our project.

Capacitive transducers are composed of two parallel plates that form a capacitor. One of the plates is attached to a diaphragm and the other to a substrate. When pressure is applied to the plate attached to the diaphragm, it bends, increasing or decreasing the distances between the two plates which changes the capacitance created by the two plates. This capacitance is then converted to a voltage that is linearly proportional to the pressure applied to the diaphragm plate.[70]

Piezoresistive transducers are composed of a semiconductor like silicon that changes resistance when a pressure is applied to the material. A device is connected to it to detect the resistance differences by supplying a small current, the change in resistance changes the amount of current that passes through the material [71] Similarly, to the capacitive transducer, an output voltage is generated linearly proportional to the pressure. This type of transducers are very good for highly sensitive applications.

There are different type of pressure sensing transducers. Those who measure absolute pressure provide an output proportional to the difference between the

applied pressure and the built in vacuum reference. Differential pressure sensing refers to the the difference in pressure sensed at each port of the transducer. Finally, gage type, measures the difference between the applied pressure and the atmospheric pressure [72]. To eliminate changes in the pressure measurements under the same conditions, it was important to select a pressure sensor with an absolute type of measurement.

To be able to choose a correct pressure transducer, it was first important to understand the pressure specifications, application accuracy, and supply and output voltage range. Due to the low operating pressure of this project, pressure sensors with an operating range of 0-15 PSI (0-103.4 kPa) were sufficient. Additionally, the accuracy of the device needed to be checked based on the application. Unavailability in the market of sensors with small pressure ranges, led to choosing the closest available range. This affected the accuracy of the sensors. The percentage of full span (%FS) specifies device accuracy based on the static and total error bands; this is given in the device's datasheet. The accuracy in an specific application is given by the following equation:

$$A_{pp} = A_{dev} * \frac{FS_{dev}}{FS_{app}}$$

where A_{pp} is the application accuracy and A_{dev} is the device accuracy [72]. Therefore, for a 15 PSI application, using a 30 PSI device with a device full span percentage of $\pm 3\%$, gives an overall application accuracy of 6% FS.

Many transducers operate ratiometrically, with the output voltage varying with dependance to the supply output. For example, with a power supply of 5 V and a 50% pressure span, the output voltage produced is 2.5 V. This type of operation is ideal for this project application. Due to the characteristics of the microcontroller, the sensors can be powered via output pin.

4.11.4.2.1. Honeywell 19mm Series

Honeywell 19mm Series have very precise small pressure range applications that meet this project's criteria of a 0-15 PSI range. They have the ability to measure a great variety of media such as fuel, propane, water, emissions, etc. A typical full-scale span of $\pm 0.1\%$ with a maximum of $\pm 0.25\%$ could provide very accurate readings due to an overall application accuracy of 0.25% under worst conditions [73]. This error is very small and the most ideal, however, this project does not need such an accurate device. This pressure transducer has a voltage output range from 0-100 mV, which means that the readings would have to be amplified for the microcontroller to be able to see and process the measurements. Furthermore, these transducers are very costly, \$148.24. Due to the high price and low output voltage, this transducer was discarded.

4.11.4.2.2. Honeywell MLH Series

Honeywell MLH Series, just like the 19mm Series is able to measure various media including water. This transducer starts at the 0-50 PSI range with a ratiometric operating providing a full scale span of $\pm 0.50\%$ gives an output in the range of 0.5-4.5V [74]. Using equation (7), the overall application accuracy would be 1.67% FS. This accuracy is much higher than the 19mm Series but still very low for this project application. This transducer is ideal for our application but also very costly, \$110.28. It also requires an additional connector which adds extra \$53.58 to the overall price.

4.11.4.2.3. Eyourlife Universal 30PSI

The Eyourlife Universal 30PSI pressure transducer G7 is designed to work with oil, fuel, diesel, gas, water, and air pressure. For the application of the system the transducer will be implemented to measure water pressure and relate the value to the microcontroller. The transducer converts PSI into an analog voltage which will be passed to an analog input pin of the microcontroller. The transducer is regulated for a PSI range of 0-30 PSI. The component requires a 5V DC input voltage and will output via the signal wire a corresponding analog voltage. The transducer is to be operated under the temperature range of 40-125 degrees Centigrade. Although, this device has a full scale span of $\pm 3\%$, providing an overall application accuracy of 6% under worst conditions, through some calibration relatively accurate measurements can be obtained. The price of this transducer is very economic, \$17.98 including the connector. Therefore, this was the transducer of choice. See figure 4.11.4.2.3-1 for the transducer. [75]



Figure 4.11.4.2.3-1: Eyourlife Universal 30PSI Transducer

4.12. Touchscreen

For any product or system that requires user interaction, a method to retrieve the user input is necessary. Since we wanted to minimize the amount of pieces required to operate our system, we opted out of using a tradition pointer device like a mouse. This lead of to the conclusion to use a touch screen device. A sharp touch screen display would not only increase the usability and reduce the complexity of the project, but also increase its aesthetic appeal.

4.12.1. Qualities

When considering a touch screen, multiple aspects need to be considered. One of the biggest aspects that differentiate this technology is whether its a resistive based or capacitive based touch screen. Although, to an unknowing third party, they may appear the same from a distance, their performances and fortes are different when their qualities are analyzed more closely.

4.12.1.1. Resistive vs. Capacitive

Resistive touch screens contain two layers with space between them and electrodes running all horizontally on one layer and all vertically along the other. The processor then computes the location of the touch by using the vertical and horizontal strips of electrodes as a coordinate system [76]. Because it relies on an intentional pressure, this method is highly accurate, especially in comparison with capacitive touch screens since those rely on any distortion of their screen's electrostatic field to activate. This means any slight touch would register as an input. Since our system will be displayed in a public area where people could accidentally bump buttons and where moisture could accumulate naturally or by users, a resistive touch screen would ideally be optimal based on this reasoning.

One popular advantage to capacitive touch screens are their multi touch capabilities [76]. Resistive touch screens end up jumping between the contact points when more that one press is registered instead. In our prototype user interface, we locked it so that can only be one selection at a time, so multi touch is no longer an advantage.

Resistive and capacitive touch screens are made out of different materials and are delicate in different ways. Resistive touch screens have a pliable outer screen layer in order to be sensitive to the user's input pressure. Because of its softness, it can be more easily damaged compared to a capacitive screen glass layer [76]. However, the glass of a capacitive screen is not without its own drawbacks. Glass can easily be scratched or cracked. However, even with severe cracks in the screen, as long as the capacitive layer underneath the screen is not damaged, the screen can still respond to user inputs since it does not rely on touch but capacitance [77]. Also, damage from scratching and impacts can be reduced by using tempered glass screen protectors which are not available for resistive touch screens [78].

One final factor is visibility. Our team's intention is to display our project in a bright atrium. Because of this, the screen needs to have good visibility in bright light. Due to the layer and space technology behind the standard resistive touch screen, more glare is visible on these devices [76].

Although resistive touch screens provide a more accurate and flexible input method where users can use any device to apply pressure to the screen, we realized durability and visibility were more important factors, especially in lieu of being designed to be publically displayed. We figured having the best display would not matter if it was broken or not readable, so we decided to choose a capacitive touch screen.

4.12.2. Touch Screen Comparison

The following table (Table 4.12.2-1) lists a few touch screens we considered for our project and some important data about each device.

Table 4.12.2-1. Touch Screen Considerations

Name:	Screen Size (mm)	Display Length (in)	Resolution (pixels)	Technology Type	Connect	Price
Raspberry Pi Touch Display (Official)	194 x 110	7	800 x 480	Capacitive	DSI & 1 GPIO	\$60.00
Waveshare 10.1" LCD Touch Screen	265 x 170	10.1	1024 x 600	Capacitive	HDMI & USB	\$116.99
kuman 5" Resistive Touch Screen	122 x 79	5	800 x 480	Resistive	HDMI & USB	\$34.95
LANDZO 7in Touch Screen	150 x 51	7	800 x 480	Capacitive	HDMI & USB	\$40.88

Residing in the first item column of Table 4.12.2-1, the first contender was the official "Raspberry Pi Touch Display as endorsed on their web site [79]. The home website for this screen boasts easy initialization by merely plugging in a ribbon cable between the screen and the Raspberry Pi's DSI port along with a jumper cable to 5V DC pin and a common ground. All the drivers necessary are native to the Raspberry Pi's operating system, Raspbian [79]. This completely eliminated any potential compatibility issues a third-party screen may bring. This capacitive touch screen was 7 inches with a pixel density of 800 x 480. This was a fairly small screen with a large border surround the actual screen.

Albeit, although the official Raspberry Pi 7" Touchscreen Display" was the simplest option, a seven inch screen initially seemed smaller. Thus, we searched for a larger screen. We came across the "Waveshare Raspberry Pi 10.1inch HDMI LCD Capacitive Touch Screen", (see the second item column of Table 4.12.2-1) which sported a 10.1 inch diagonal and had a high pixel density of 1024 x 600 pixels so that the larger screen had good displaying detail [80]. Although, the size seemed really nice, the large diagonal came with a hefty price of \$116.99. The 10.1 diagonal would allow for the system's GUI to be more detailed and contain larger buttons, but its cost was out of our budget.

Since cost is a limiting factor concerning our resources, we also looked at one resistive touch screen despite favoring the alternate technology (which was discussed in section 4.13.1.1). The "kuman 5 inch Resistive Touch Screen" was the sole contender of resistive touch screens, see item column three of Table 4.12.2-1). This consideration was justified because of the screen's price of \$34.95, more than ten dollars cheaper than the official touch screen [48]. Another major downside to this model its display has only a 5 inch diagonal. The small dimensions could potentially make accessing the GUI tedious.

Lastly, the 'LANDZO 7 Inch Touch Screen" was considered. This was a much cheaper alternative to the official touch screen because it was only priced at \$40.88 [81]. Hardware design-wise, the black border around the visible part of the screen is a lot more minimal. Unfortunately though, the HDMI and USB cables attach themselves to the side of the screen. Since we planned to mount the touch screen onto the rest of our stand, we wanted to reduce the number of exposed wires.

The main issue that dissuaded us from using one of the alternative touch screens is their need for installing additional drivers. Reviews posted on online retail sites [80][48][81] have expressed many compatibility issues. Although the size of the Waveshare model was convenient, it was too expensive for our budget. The cheapest touch screen was the smaller resistive kuman one. Although being a resistive touch screen would not be an issue for us, the smaller display size could make operating the GUI challenging. Finally, the LANDZO screen was a cheaper alternative to the official model, but the placement of the HDMI and USB cable reduce the aesthetic appeal we are trying to obtain. So in the end, after we considered our options, we decided to choose the "Raspberry Pi Touch Display" for its ease of use, compatibility, and visual appeal.(See Fig. 4.13.2-1)



Figure 4.13.2-1: Raspberry Pi Touch Display

4.13. Software Tools

Throughout the lifecycle of the project at hand, various types of software was used. We used third-party software in order to help our efforts be more effective and efficient. In every aspect of of project, whether the program was used for team communication, in the design and development process, for documenting and storing our progress. Most of the software we chose to use was from personal previous experiences, however some applications (i.e. Asana) were discovered and chosen through research and experimentation to find and determine what served our purposes best.

4.13.1. Communication

The software used for communication purposes needed to fulfil any and every communication need. We used Slack as a hub for team communications between ourselves, between our team and cooperative teams, and between our team and our faculty advisor and Asana as a bulletin board for our tasks both individually and as a team.

4.13.1.1. Slack

With the team spread out through central Florida, a central communication medium was needed. Slack provides a free service that unified our conversations. Through Slack, multiple channels can be created which helped organize our conversations into categories such as “general” (project related subjects for our team with our faculty advisor, Dr. Chan), “contributors” (talk with and concerning our collaborators in PES), and “off topic” (general conversation between the four members of our group). Slack allows the admin to control which members have access to which channels, but any member of the Slack group has the ability to initiate a direct message thread with any other of the members. Another advantage this platform provides is third-party software integration with services like GitHub, Google Drive, and Asana (which our team had previously decided to utilize). Notifications can be configured for the web app, desktop app,

and/or mobile app to stay current with any updates in the channels and direct messages [82][83].

4.13.1.2. Asana

Although Slack provided a central hub for all our team's communications, we still lacked a task manager to consolidate our tasking and project deadlines. We discovered PCmag's Editor's Choice application for task managing. Jill Duffy praised it for being flexible, lightweight, and free [84]. Unfortunately, Asana lacks a desktop app and is only supported on mobile and web platforms. The advantage Asana provided for us was a user interface that clearly displayed workflow. Each member could create tasks and assign members to it, due dates, and a list of what needed to be accomplished specifically. The application also provides analytics to evaluate each member's contributions. Besides its beautifully designed layout, its native integration with Slack really made this app more handy since all our communication was in a sole location [82].

4.13.2. Development

Applications that assisted our team in the design and development helped up create and connect the various aspects of our projects. Text editors, Integrated Development Environments (IDEs) and GUI designers helped us construct the underlying source code that controlled our system. CAD programs assisted with the hardware aspects of our design. We also used software packages on the Raspberry Pi to assist us in establishing network connections.

4.13.2.1. Vim

Mr. Kartik Ayyar, a software engineer, made the claim that a good programmer should understand the code better than a Integrated Development Environment (IDE) [85]. The software development team for this project felt the same way. Since both developers felt confident in their ability to read and write code, there was no need to rely on an IDE's limited semantic knowledge.

Another factor we considered is that most all the development will be performed on the Raspberry Pi over a network connection. If we decided to use an IDE, seeing its GUI would be necessary. Since the Raspberry Pi has a limited RAM and WiFi card, there would be slight lag. By using a terminal text editor like Vim, only a SSH (Secure SHell) was necessary since there was no GUI. This allowed the editor, through a network connection, to reach practically-real-time speed since it was just text being sent over the network.

We decided to use Vim, a lightweight, console based text editor, because of the plethora of keyboard commands which the development team was already familiar with.

4.13.2.2. Qt Designer

Upon the decision to use Python 3 for the user interface backend, one of the deciding factors for utilizing the Qt framework was because of Qt Designer. Qt designer allows developers to drag-and-drop various layouts, spacers, buttons, items, widgets, containers, and other elements onto the workspace. When the developer selects an element, the property editor enables properties like object name, widget dimensions and color, frame effects, and label formatting to be changed. The creator can then preview the design to verify that it looks as it was intended. On saving Qt Designer generates the XML code for the front end as a *.ui file. From here, the GUI developer just needs to program how the buttons specifically function in terms of the application. Instead of spending hours, coding the XML script for the GUI, our team was able to focus on the backend, testing to ensure the GUI front end works with the backend so that users of our system can interact with it as intended.

4.13.2.3. Arduino IDE

Because the computer engineer team lacked familiarity with how to program the ATmega 2560 microcontroller, we started with utilizing the Arduino IDE. The Arduino IDE provides a useful set of training wheels to learn how to work with the ATmega2560 development board. As the Tinkerer expresses in his blog, using the IDE wastes the microcontroller's resources and does not allow programmer access to peripherals [86]. So in effort to understand the microcontroller better and to have more direct control over the hardware, we migrated away from using the Arduino IDE and utilized Vim to program the chip.

4.13.2.4. SolidWorks

SolidWorks is a 3D modeling computer aided design and engineering computer program. This software was used to design the overall system including the PES contributions. Since the software is designed to be able to model in true dimensions, all the components were created in a library based on the dimensions provided by each specification. The components were then put together to model the overall system mounted into the board. This was very important to keep both PES design and our design uniform [87]. Also, due to the design being true to size, it reduces the possibility of errors and waste of materials when placing the components, cutting the pipes and aligning the gears from the motors and the valves. Furthermore, the gears necessary to increase the operating torque of the motors and turn the valves were designed using this software. This made the design very simple when it came to the teeth of one gear fitting perfectly into the holes of the other. This alignment is necessary to be able to achieve the necessary torque without running into issues like the gears detaching from their set point. (See hardware section for gear design). The modeled gears were then laser cut to eliminate as much human error as possible and use a strong enough material that will not easily break or wear down easily after contact with the other gear.

4.13.2.5. Eagle

Eagle is an Autodesk product used to design printed circuit board layouts. It has features such as schematic mapping, built-in libraries, electronic rule checking, etc. This software was used to be able to design the signal and power PCBs. Thanks to many open sources, Autodesk tutorials and documentation the software was fairly easy to learn and use once the PCB components, the total voltage inputs/outputs and current values were identified. Built-in libraries provided the necessary information about the various components and the pinouts. Electronic rule checking was very useful to validate the schematic designs and make sure there were no errors. Finally, real time design synchronization saved time by syncing all the changes in the layout without having to address each individual components the change could have affected. (See PCB designs in sections 5.1.5 and 5.1.6)[88]

4.13.2.6. Hostapd

Hostapd is a space daemon used for an access point's authentication. This software is used on the Raspberry Pi 3 to set up an ad-hoc network, giving the Raspberry Pi 3 the ability to act as an access point. It implements 802.11 by IEEE and supports WPA and WPA2 authentication. Hostapd is a service with no front end, rather it is intended to be run in the background in a server like manor. This proved to be the ideal situation for the system to be implemented as the user should not even be aware of the service being running except when initially connecting to the the wireless access point. The wireless network is a requirement to this system to allow for the user to mirror the user interface without the need for a 3rd party network connection, eliminating uncontrollable variables. [89]

4.13.2.7. RealVNC Connect and Viewer

RealVNC was chosen as the remote desktop application due to the ease of implementation, as well as the ability to directly address a private IP. When the Raspberry Pi 3 acts as a wireless access point using Hostapd it is assigned the private IP address 169.254.166.141, This is a static IP address which will never change upon boot up of the system, so it can be assured the connection point will always be there. Having the Raspberry Pi 3 without access to the internet to remote desktop into the user interface a direct connection will be used. This method in the traditional sense will allow the user to connect to the host using the host's private IP address assigned by the router. The traditional implementation would depend on another piece of hardware to act as the middle-man between the client and host. Instead of the traditional implementation the Raspberry Pi 3 will act as the middle-man and the host. This allows the system to eliminate another dependency. The client will need to install RealVNC server's counterpart RealVNC viewer. This makes for easy setup and implementation for the user as

they will only be required to download an application and input the Raspberry Pi 3's private IP address. [90]

One of the advantages for us to adopt RealVNC into our system is that RealVNC Connect comes preinstalled on the Raspberry Pi and requires minimal setup to initiate [91]. In addition, the RealVNC Viewer software is always free to use as long as the user accepts the EULA during the installation process [92]. This enables anyone who wants to remotely use our system to freely and easily do so.

4.13.3. Documentation and Storage

Almost entirely all of our documentation, presentations, and reports were stored on Google Drive and its Google Documents software suite (Documents, Sheets, Presentation). The only exception was for our source code which we utilized for its storage and any associated documents relating to the code.

4.13.3.1. Google Drive

Since the members of the team work remotely, it is important to be able to access any files related at any time from any location. Our solution was to turn to Google's cloud storage service, Google Drive. Michael Muchmore wrote a review on the service and praised it for its collaboration functionality like device-to-device file-syncing and 15 GB of free storage space. [93]. Muchmore claims that Google Drive offers better tools for real-time collaboration and free storage space compared to any competitors [93]. This cloud service also offers offline file access when using the Google Chrome web browser with the Google Drive extension installed. Because of its ability to "store any file", "see [my] stuff anywhere", and "share files and folders" in real time, we decided to use Google Drive as our file hosting medium [94].

4.13.3.2. GitHub

Git is an open-source version control system. Git allowed for the software development team to store each version of the software in a central repository (repo) where each member could download, make changes to, and upload their latest revision of the code [95]. Git helps manage file changes and ensure file integrity. Git is purely a command-line tool though. Since the members of the team needed to collaborate remotely to maximize their time, they choose to utilize GitHub.com, an online service that allowed the team to use a single online repo that everyone can access, modify, and update. Members of the team could create their own branch off the repo. This allowed for their updates to be tested without merging it with the master (main) branch to verify that the code still functions as desired. If by chance, an error went unnoticed, GitHub lets its users track the changes made in the code across versions which could be reverted to if desired.

4.13.3.3. Draw.io

In order to provide a visual representation of the system being designed, software was needed to create diagrams. We chose to utilize Draw.io, a free, simple diagram web application that we have had prior experience using. Draw.io provides a plethora of basic images, symbols, icons, text, and lines that can be dragged-and-dropped onto the workspace. Just in case the application does not natively support a desired icon, it also supports user-created icons which can be made out of any common image file type (e.g. .jpg or .png). Draw.io is a non-subscription, non-membership service that integrates services like Google Drive and GitHub for file saving purposes. The project file is saved as a .xml file, but the diagram can be exported as .jpg, .png, .pdf, or .svg. One of the downsides to this application is that it is purely a web based application with no desktop or mobile native integration, but we could work around this. “Although it lacks graphical enhancements and sharing capabilities, by using Google Drive, our team could easily collaborate and review each other’s work [96].

5. Design

Design of the system is a very important step in the development process. Both hardware design and software design is used to ensure the most effective and efficient building of the system occurs. Designing the system before implementation allows for the saving of time and money. It also allows the group to visualise the tasks at hand which need to be priorities. Designing the system allows for us to measure progress of the system and gauge correspondingly which processes should be implemented next. Hardware design will provide a way to perceive which components interact with one another and which values need to be provided at each angle. Software design will show how the system will interact on a software level and provide a way to develop software strategically and efficiently. It will also allow for code for each controller to be developed separately with the idea of what each component is responsible for.

5.1. Hardware Design

The hardware design is an overview of all the hardware components in the system and how they interact with each other. It will list the functions of each component and how it operates in addition to the communication between each device. The devices components will be listed as well as characteristics. The diagrams will visualize the interaction between the devices as well as the data transfer between them. The overall purpose of this section is to show the integration of each hardware in the system and where it will perform.

5.1.1. Hardware Block Diagram

Referring to the block diagram representing the electrical, electronic, and miscellaneous components of the project below (Figure 5.1.1-1), the hardware and power systems side of this project can be understood. Each block represents a key element of the system. Red blocks represent the electrical aspect of the labeled component (Section 5.2.1. explains the software and data and control flow through the system.) Red arrows represent power being drawn between components. Grey blocks represent mechanical aspects of the system. And lastly, black arrows represent a mechanical relation between components. Any labels next to an arrow describe the amount of voltage being supplied to the respective components. The power PCB served as the main hub of the power aspect of our system. It converts the wall's AC power into DC power and converted the voltage level specific for each component of the project that needed to be powered.

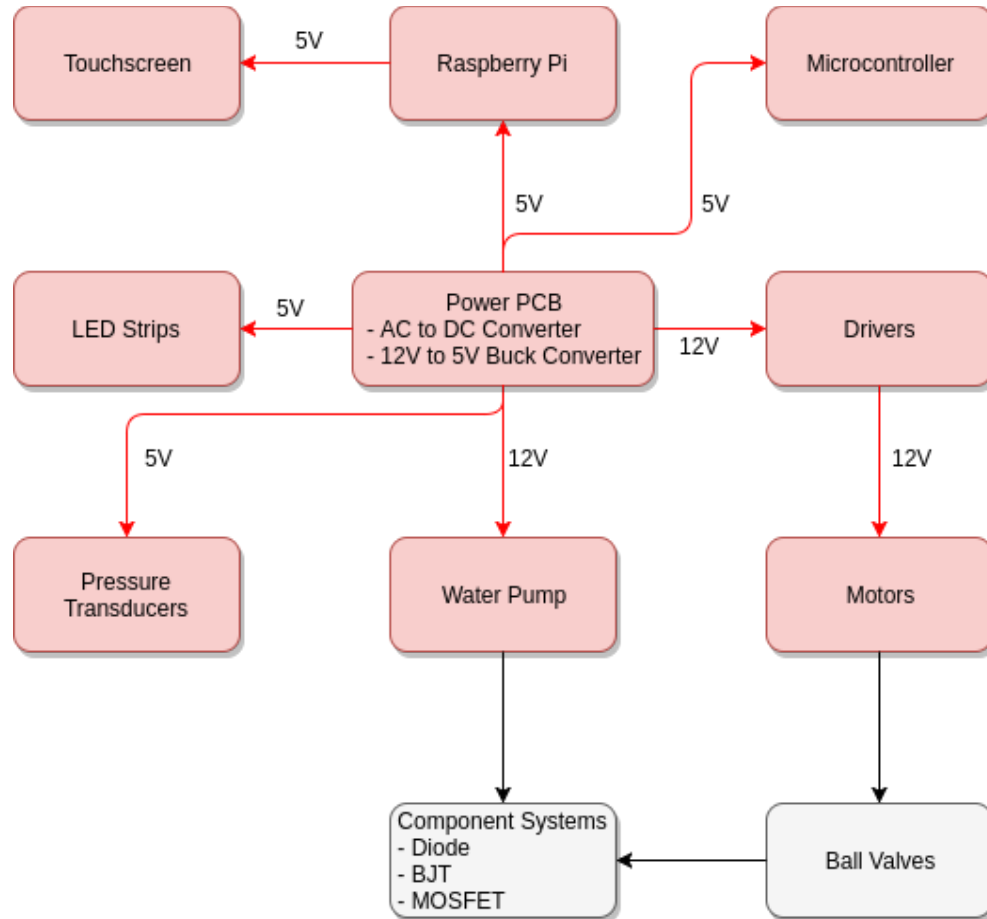


Figure 5.1.1-1: Hardware Block Diagram

5.1.2. Hardware Design Overview

The below design shown in Figure 5.1.2-1 illustrates the desired final configuration of our project. The three stages from left to right represent the mechanical representations of the BJT, Mosfet, and Diode respectively. The parts labeled with a V are valves, FM represents flow meters, and S represents pressure sensors. Valves will be utilized to control flow as well as change pressures at various points, the sensors will measure pressures and these will be related to various “voltage” representations, and the flow meters will be used to examine the rate of flow through various branches.

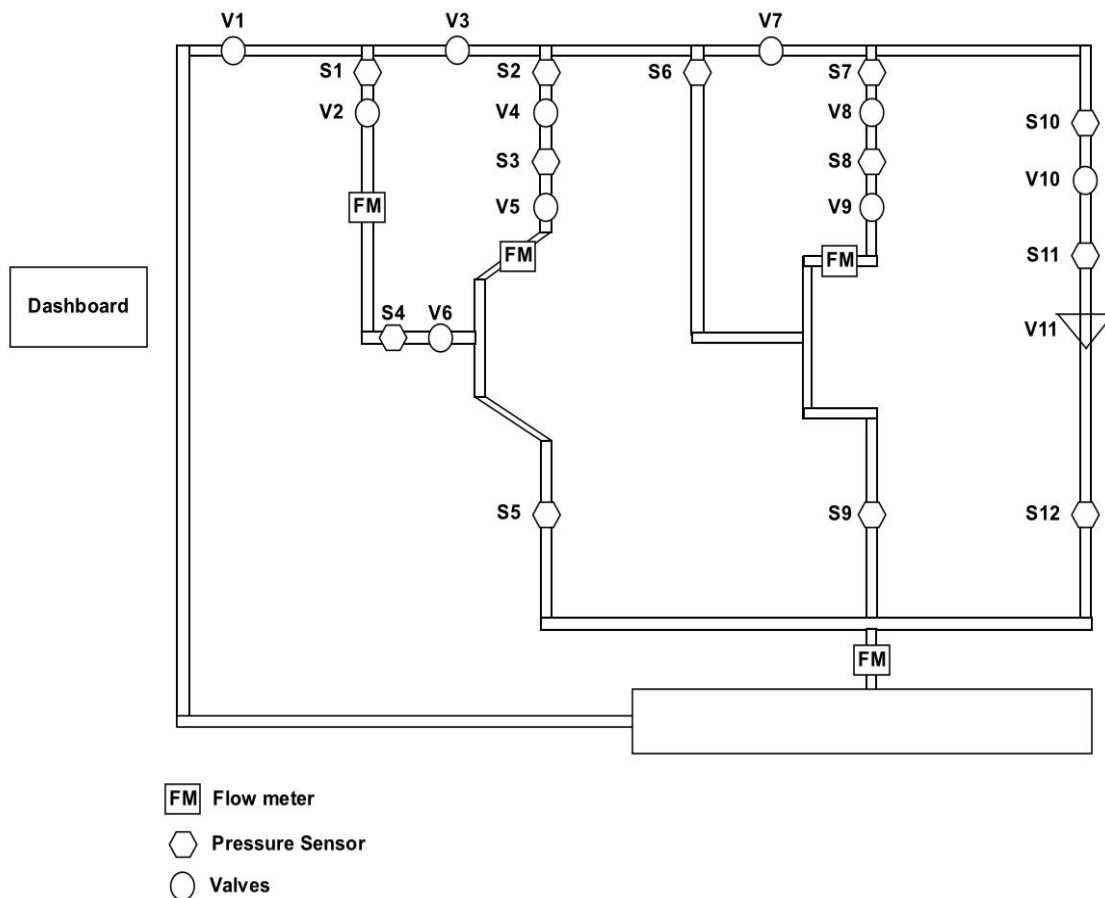


Figure 5.1.2-1: Overall System Design

Figure 5.1.2-2 shows the electrical schematic for a basic diode circuit that is to be represented in our project by the rightmost branch. The input voltage V_i is represented by the pressure difference between S10 and S12, this pressure can be controlled by varying the opening of V10. V10 represents the resistor R , the voltage drop across this resistor V_r is represented by the pressure drop between sensors S10 and S11, this drop can be manipulated by varying the openings of both valves V10 and V11. V11 represents the diode in the below circuit, in cutoff mode water will flow towards the closed valve that represents the diode but not through it until the desired pressure difference between S11 and S12 is achieved which corresponds to less than the turn on pressure “voltage” of the diode, at this point V1 will close cutting off the flow of water and the pump will be turned off keeping the voltage across the diode constant and approximately equal to the input voltage. When the diode is on, water will now be allowed to flow through the valve and the opening of V11 and V10 will be changed to maintain a constant pressure drop across the diode from S11 to S12 that will equal the turn on pressure “voltage” of the diode. The values that will be controlled by the users will be the resistor value R and the input voltage V_{in} from these values the voltage drops as well as if the diode is on will be determined.

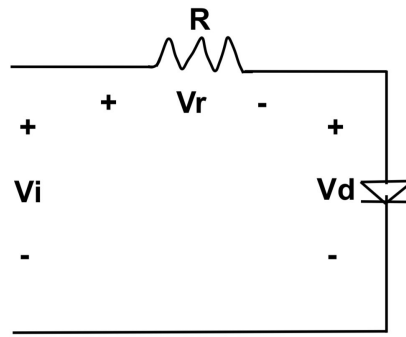


Figure 5.1.2-2: Diode Electrical Representation

Figure 5.1.2-3 shows the electrical schematic for a basic BJT circuit that is to be represented in our project by the leftmost branch. The input voltage V_{in} is represented by the difference in pressures shown by sensors S1 and S5, this pressure is changed by utilizing valve V2 which corresponds to the resistor R_b . The biasing voltage V_{cc} is represented by the difference in pressure shown by sensors S2 and S5, this pressure is set by varying the opening of the valve V4 which represents the resistor R_c . The valve V3 in between the collector and base branches of the BJT representation is utilized to give a difference in pressure thus ensuring that V_{in} does not always equal V_{cc} , however a limitation to this approach is that the pressure “voltage” at V_{in} will always be higher than the pressure “voltage” at V_{cc} due to the pressure drop across the valve. The base to emitter (V_{be}) voltage shown below is represented by the pressure difference between sensors S4 and S5, once the BJT is operating, this pressure difference will be set to a specific value equal to the turn on pressure chosen, this will be set by adjusting the opening of the valve V6. Finally the collector to emitter (V_{ce}) voltage shown below is represented by the difference in pressure shown by sensors S3 and S5, depending on the mode of operation the BJT is set in this pressure drop “voltage” will be either large (representing forward active mode) or small (representing saturation mode). This pressure will be regulated by varying the opening of the valve V5 which will build up pressure at the sensor S3. The values that will be able to be set by the user for this component will be the resistor values R_b and R_c as well as the two input voltages V_{in} and V_{cc} , from these values the mode of operation will be calculated as well as the drops across the resistors.

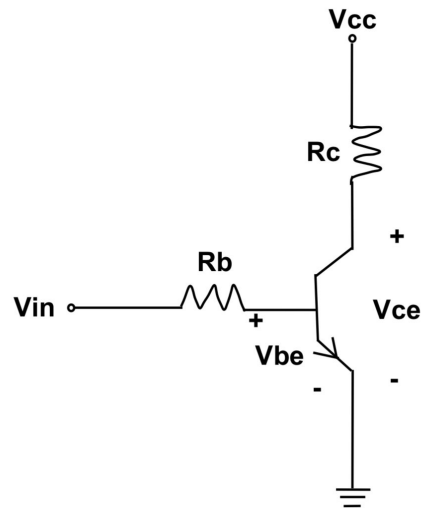


Figure 5.1.2-3: BJT Electrical Representation

Figure 5.1.2-4 shows the electrical schematic for a basic MOSFET circuit that is to be represented in our project by the middle branch. The input voltage V_{in} is represented with the pressure difference between the sensors S6 and S9, this voltage is set by varying the opening of the valve V7. The input voltage V_{cc} is represented by the pressure difference between the pressure sensors S7 and S9, this pressure difference is set by changing the opening of the valve V8 which represents the resistor R_d . Due to the inclusion of the valve V8, the voltage at V_{cc} can never exceed the voltage at V_{in} due to the inherent pressure drop across V8. The gate to source (V_{gs}) voltage is set by taking the pressure difference between sensors S6 and S9, it can be seen from the picture below that this pressure “voltage” is simply equal to the input voltage V_{in} . Finally the drain to source (V_{ds}) voltage is represented by taking the pressure difference between the sensor S8 and S9, this pressure “voltage” will be manipulated depending on the mode of operation the MOSFET is being operated in (little voltage drop in triode region and large voltage drop in saturation mode), this pressure difference will be set by changing the degree of opening of valve V9. The values that will be controlled by the users will be the resistor value R_d as well as the input voltages to the system V_{in} and V_{cc} , the mode of operation as well as voltage drops throughout the system will be set from these values.

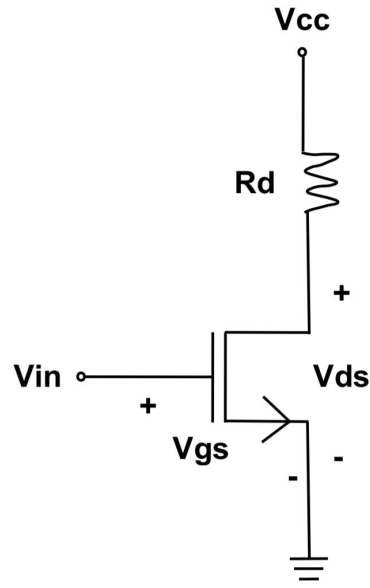


Figure 5.1.2-4: Mosfet Electrical Representation

5.1.3. Mechanical Overview

This section describes the design and placement of the various mechanical components in the board design. These were placed strategically to be able to create the pressure drops (voltage drops), obtain accurate measurement and represent specific circuits.

5.1.3.1. Valve

As was previously mentioned, 2 Nm of operating torque is needed to be able to turn the clear PVC Ball Valve. The chosen motor stepper motor has only 1.26 Nm. Therefore, gears were designed to compensate for the necessary operating torque. To accomplish this, the number of teeth ratio necessary to turn the valve was determined using the following equation:

$$T_D = T_A \frac{Teeth_D}{Teeth_A}$$

where T_D represents the torque required to turn the valve, T_A represents the torque of the motor. This equation gives a teeth ratio of approximately 1.25. As long as the ratio is equal to or greater than this value the motor will be able to turn the valve.

The design of the gears also took into consideration the motor rotor and valve leveler connector dimensions and its shape. Through some testing, it became obvious that due to the water flow the torque to turn the valves needed to be increased, so the teeth ratio was increased to approximately 1.83 to accommodate for this. See the figure below for the gear design for both the motor and the valve respectively.

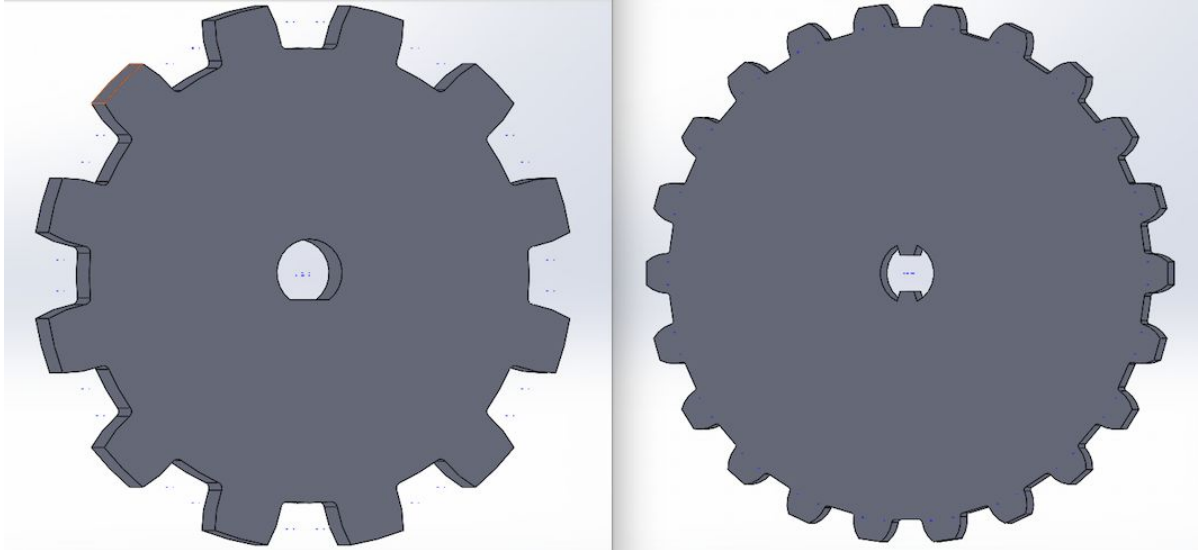


Figure 5.1.3.1: Motor and Valve Gear Design

The gears images are not to scale, due to the higher quantity of teeth existing in the Valve Gear this one has a radius of 52mm, while the Motor Gear has a radius of 27mm. The teeth fit perfectly in one another, giving an increase of 0.31Nm of torque. The alignment of the valve and motor gears were tested during the motor and driver testing.

5.1.3.2. Pipes

The pipes' layout was designed based on the desired circuit arrangement. The pipes have a diameter of $\frac{1}{2}$ " to easily accommodate for all the components that need to be connected to it. All the fittings that allowed the connection and turn of pipes are of the same diameter.

Pressure transducers have a female type joint with a diameter of $\frac{1}{8}$ ". To be able to acquire measurements the transducers were included in the system by adding a "T" at the point where the measurement is to be taken. Therefore, a $\frac{1}{2}$ " male x $\frac{1}{8}$ " female NPT coupler was necessary to properly install the transducer without having any water leakage or pressure escape.

In the other hand, the flow meters have a 1" opening, which is double the diameter of the pipe. This resulted in having to use a 1" x $\frac{1}{2}$ " PVC Sch. 40 reducer bushing and a $\frac{1}{2}$ " PVC male adapter to be able to connect the pipe to the bushing. Once sealed, the connection between the pipe and flow meter had no water leakage. These were also placed strategically throughout the board to provide water flow values to equate to current flow in the designed circuit.

5.1.3.3. Motor

Motors were placed in closed proximity and with alignment of all the valves. It was important to maintain the alignment to be able to achieve the necessary torque to rotate valves. To do so, the motors were supported within the frame of the board, exposing only the rotor and gear at a certain distance to match the valve gear placement. The wires and motor drivers were routed in the back of the board to maintain the aesthetics of the design.

5.1.4. Pressure Sensor

Transducers provided us with the ability to adjust the pressure differences by manipulating the valves without having to hardcode any of the values. Through the use of the microcontroller, the pressures will be sampled once the user inputs the values and starts the system, the program will calculate the pressure differences correlate it to a voltage drop and adjust the valves to obtain the necessary values.

The pressure transducer will be used to adjust the mechanical valves so that the corresponding pressure differential represents voltage in the system. This will be performed by relaying the pressure differential to the microcontroller. Once the microcontroller receives the pressure differential it will process the data to determine if the correct relationship has been applied. If the relationship is as it should be, the microcontroller will issue no commands. If the relationship is not correct, the microcontroller will adjust the valves correspondingly in an attempt to create the proper pressure differential. This will be done until the proper pressure is received by the microcontroller to reflect voltage in the system.

5.1.5. Power PCB Design

The output of the AC to DC adaptor that will serve as the input to the power printed circuit board is a 12 volt DC voltage with an output max current of 16.67 amps. Due to the requirements of multiple voltage rails to be included on the power pcb as well as the extremely high levels of power being dealt with on each rail, two different switching regulators were originally going to be utilized to output our correct voltages. However since one of the necessary voltage rails was already 12 volts which is the output of the AC to DC converter, only one voltage rail had to be designed. This voltage rail that was constructed is the 5 volt rail, the devices powered from this rail are the raspberry PI 3, all of the pressure transducers, as well as the LED strips that are being utilized. The recommended input current to the raspberry PI to ensure fast speeds as well as reliability is 2.5 amps, the pressure transducers utilized in our design were determined in the lab to draw approximately 1.5 milliamps of current as they are being powered, and finally as stated in section 4.10.1.2 the current draw per meter of LEDs being lit is approximately 1.8 amps. The 2.5 amps to the raspberry PI will be a constant current draw, however while the device is running ten pressure transducers total

will be running simultaneously so the current draw from them will be approximately 1.5 milliamps * 10 transducers or 15 milliamps, in addition to this the worst case scenario for the amount of LEDs being lit up is approximately 3 meters therefore the max amount of current being drawn by the LEDs is going to be approximately 1.8 amps * 3 meters or 5.4 amps. The total current draw therefore for the 5 volt power rail is approximately 5.4 + 2.5 + 0.015 or 7.915 amps, this value is approximated to 8 amps for simplicity and to account for any excess current draw. Utilizing these values of a 12 volt DC input, 8 amps output current, and 5 volt DC output voltage the WEBENCH software for power schematic designing provided for free by Texas Instruments (<http://www.ti.com/design-tools/overview.html>) was utilized to provide the schematic pictured below in Figure 5.1.5-1 while the components utilized are listed in Table 5.1.5-1. It can be seen that the switching regulator utilized to obtain the constant 5 volt output on this rail is the TPS40305 switching regulator. This regulator was chosen above the others offered by TI's WEBENCH software due to its extremely high efficiency which is displayed in Figure 5.1.5-2, where the blue, green, and orange lines represent 13, 12, and 11 volts input respectively from this it can be seen that at our current draw of 8 amps and input voltage of 12 volts, the efficiency is approximately 95 percent. This high efficiency is desirable due to the already extremely high power demands of our project.

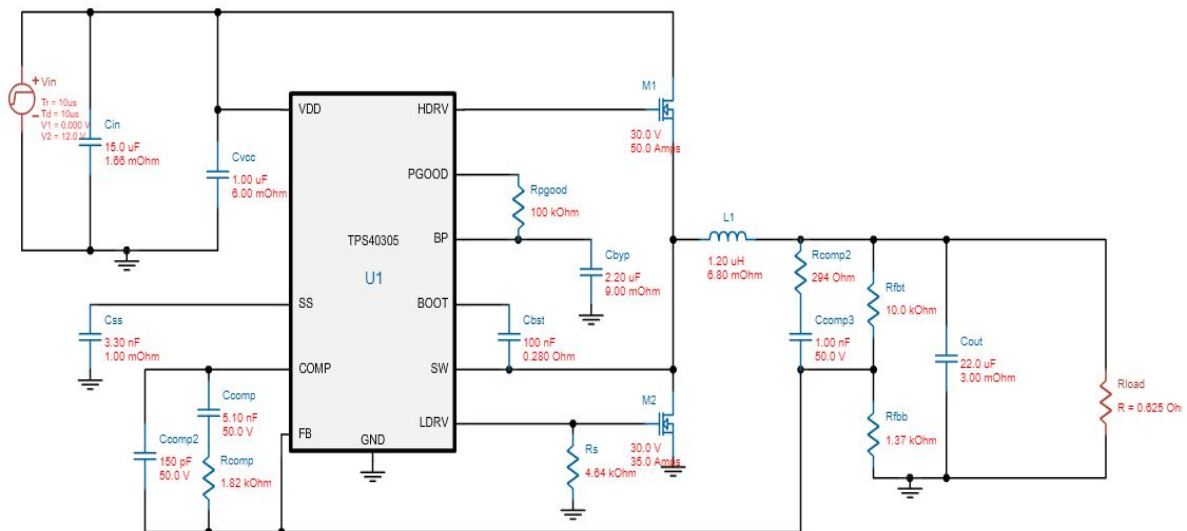


Figure 5.1.5-1. 5 Volt Power PCB Rail Schematic

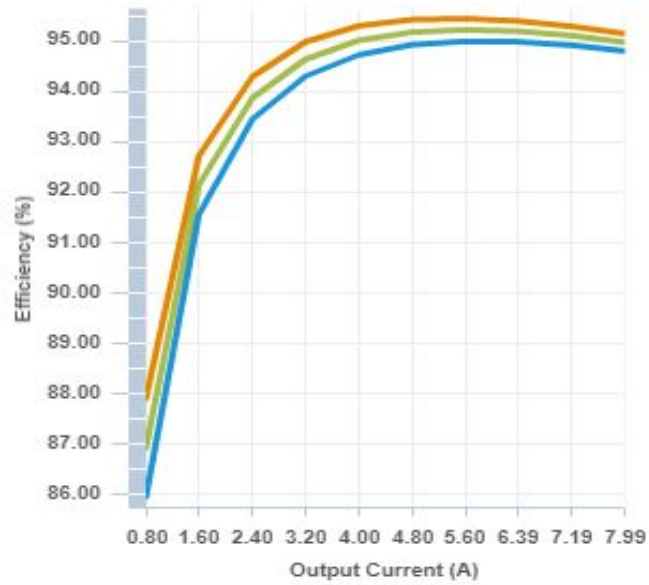


Figure 5.1.5-2. TPS40305 Efficiency vs Output Current

Table 5.1.5-1. 5 Volt Power PCB Rail Components

Part	Attribute	Value
Cbst	Capacitance	100 nF
Cbyp	Capacitance	2.2 uF
Ccomp	Capacitance	5.1 nF
Ccomp2	Capacitance	150 pF
Ccomp3	Capacitance	1 nF
Cin	Capacitance	15uF
Cout	Capacitance	22uF
Css	Capacitance	3.3 nF
Cvcc	Capacitance	1 uF
L1	Inductance	1.2 uH
M1	Vds Max	30 V
M2	Vds Max	30V
Rcomp	Resistance	1.82kOhm

Rcomp2	Resistance	294 Ohm
Rfbb	Resistance	1.37 kOhm
Rfbt	Resistance	10 kOhm
Rpgood	Resistance	100 kOhm
Rs	Resistance	4.64 kOhm

The other rail which will be implemented in addition to the 5 volt power rail will be a 12 volt power rail. This rail will have the drivers which control the bipolar stepper motors to turn the valves as well as the atmega2560 which will be the “brains” behind the signal PCB. The current draw of the bipolar stepper motor drivers were observed during the testing session on the lab to be approximately 1.4 amps while the motor is functioning while the recommended input current to the atmega2560 is approximately 1 amp max. The input current to the atmega2560 is a constant 1 amp however the max amount of bipolar stepper motors that will be running at any point in time simultaneously is 4 therefore the max current draw from the drivers will be 6 drivers * 1.4 amps/driver or 8.4 amps. The total current draw therefore for the 12 volt power rail is approximately 8.4 + 1 or 9.4 amps, this value is approximated to 10 amps for simplicity and to account for any excess current draw. Due to the output of the AC to DC converter being 12 volts with a max output current of 16.67 amps, there is no additional circuitry required to supply the power required from the 12 volt rail as it can be pulled directly from the converter.

Utilizing the Eagle schematic and PCB design software the above two designs were combined to yield the circuit shown below in Figure 5.1.5-3, this circuit utilizes a 12 volt input from an AC to DC converter and outputs both a constant 12 and 5 volts on two separate rails.

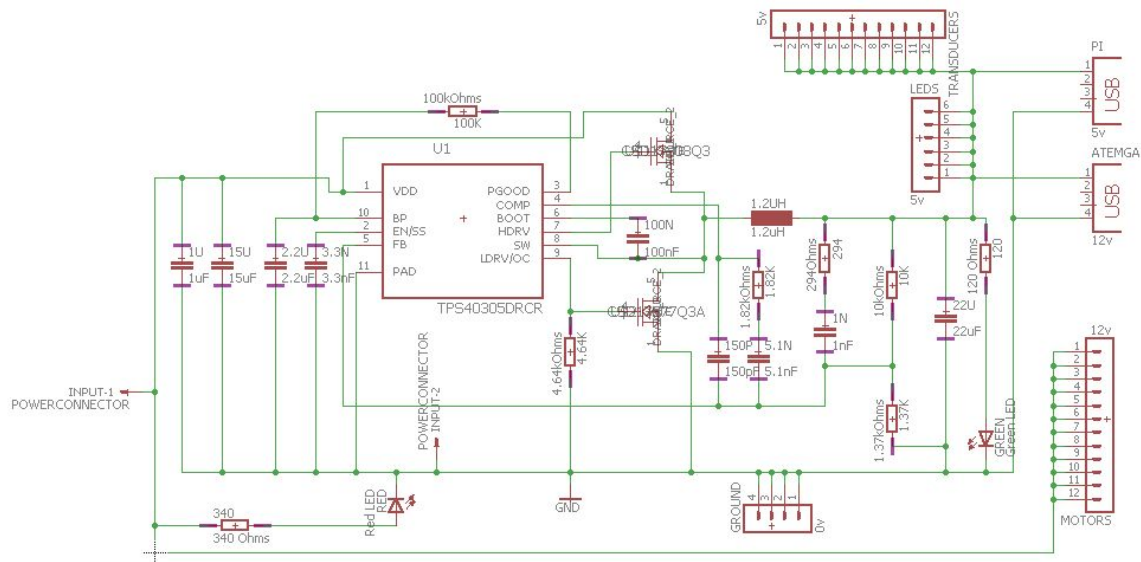


Figure 5.1.5-5. Power PCB Schematic

5.1.6. Signal PCB Design

Figure 5.1.6-1 displays the schematic of the signal PCB. The design of the signal PCB was obtained by extracting the necessary items from the Arduino Mega 2560 Development Board. The schematic is composed by the ATmega 2560 and ATmega 16 U2-MU chips. The ATmega 2560 is used to read, control and program LEDs, motor drivers, pressure transducers, etc. A total of 28 Digital I/O pins were allocated to connectors to be used as direction (DI), and enable (EN) ports for the motor drivers, and LED Data connectors. There are 16 Analog I/O pins allocated to reading the the analog voltage that are output by the pressure transducers. Additional 12 pulse width modulated pins were allocated to control the pulse (PUL) for the motor drivers. Only two pins were needed for communication purposes. Regardless of how many devices need to communicate with each other only two lines and a common Ground are needed because I²C works on a data and a clock bus. For this reason, connectors for the serial data line (SDA) and serial clock line (SCL), physical pin 44 and pin 43 on the ATmega2560 respectively. The SCL bus is used to synchronize every data transfer over the SDA bus. The devices connected to these busses are either masters (there is usually only one) or slaves. The master initiates all data transfers and controls the SCL line. Slaves can communicate through the SDA bus but only in response to the master, not on their own initiative. Since there is a sole data bus, each device must have a unique address so that the master can direct the data to a specific slave. For each transmission, the master first sends out a start condition: SDA is reset before the clock's edge. The master then broadcasts the address of the slave it wants to communicate with over SDA followed by the action it wants the slave to perform (logic level 1: data request; logic level 0: data transmission). After the addressed slave replies with an acknowledgment, the eight bits of data is sent over the SDA bus from either the

master or slave depending on the command bit. At the end of the transmission, an acknowledgement is sent before the master sets the SDA bus high to indicate the stop condition. Refer to Fig. 5.1.6-1 for a visual representation of this process.

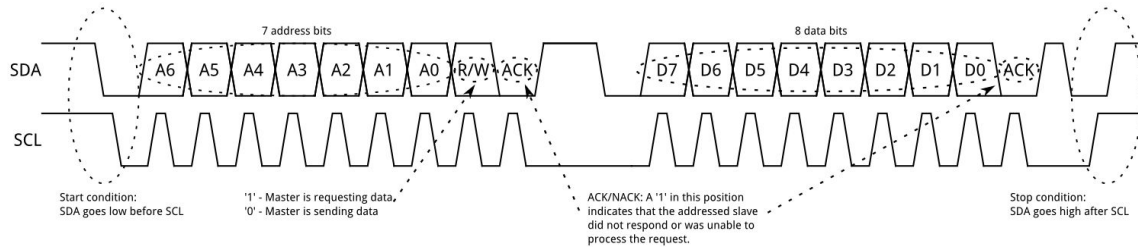


Figure 5.1.6-1: I²C Signal [97]

Finally, connectors were allocated for ICSP. ICSP or in-circuit serial programming allows for the microcontroller to be programmed after installation on the signal PCB. Without ICSP the chip would need to be pre programmed prior to installation and could not be altered or updated. The pins required for ICSP include Vin, Ground, MISO, MOSI, SCK, and Reset. Vin and Ground provide power to the board while it is being programmed allowing for the executable code or bootloader to be flashed to the device. MISO stands for master in slave out and is responsible for the slave microcontroller sending data to the master. MOSI stands for master out slave in and supports the master sending data to the slave microcontroller. SCK stands for serial clock which allows for the master and slave microcontrollers to stay in sync during transmission of the data between the boards. The reset pin is used to reset the board to allow for the bootloader or executable code to be updated and run on the slave microcontroller. Overall this provides a way for the signal PCB to take full advantage of the pins incorporated through the implemented code and circuit design for the system.

Input power can be connected to the POWER pins and through the USB port. From the POWER pins 3.3V and 5V can be used to supply to any external component necessary. The power supplied through the USB port is managed by the ATmega 16 U2-MU and an ultra-low power linear regulator. The ATmega 16 is connected to the ATmega 2560 through serial communication via pins PE0 and PE1.

Both chips have green and yellow LEDs. The green LED turns on when the chip is on the the yellow LEDs turn on when their corresponding PWM pin is enabled. They also have a reset button assigned to each one in case they need to be resetted or turned off (button is kept pressed) without disconnecting the Signal PCB from the input power. There are two 16 MHz crystals, one connected to

each chip, to maintain stable frequencies and steady clock signals.

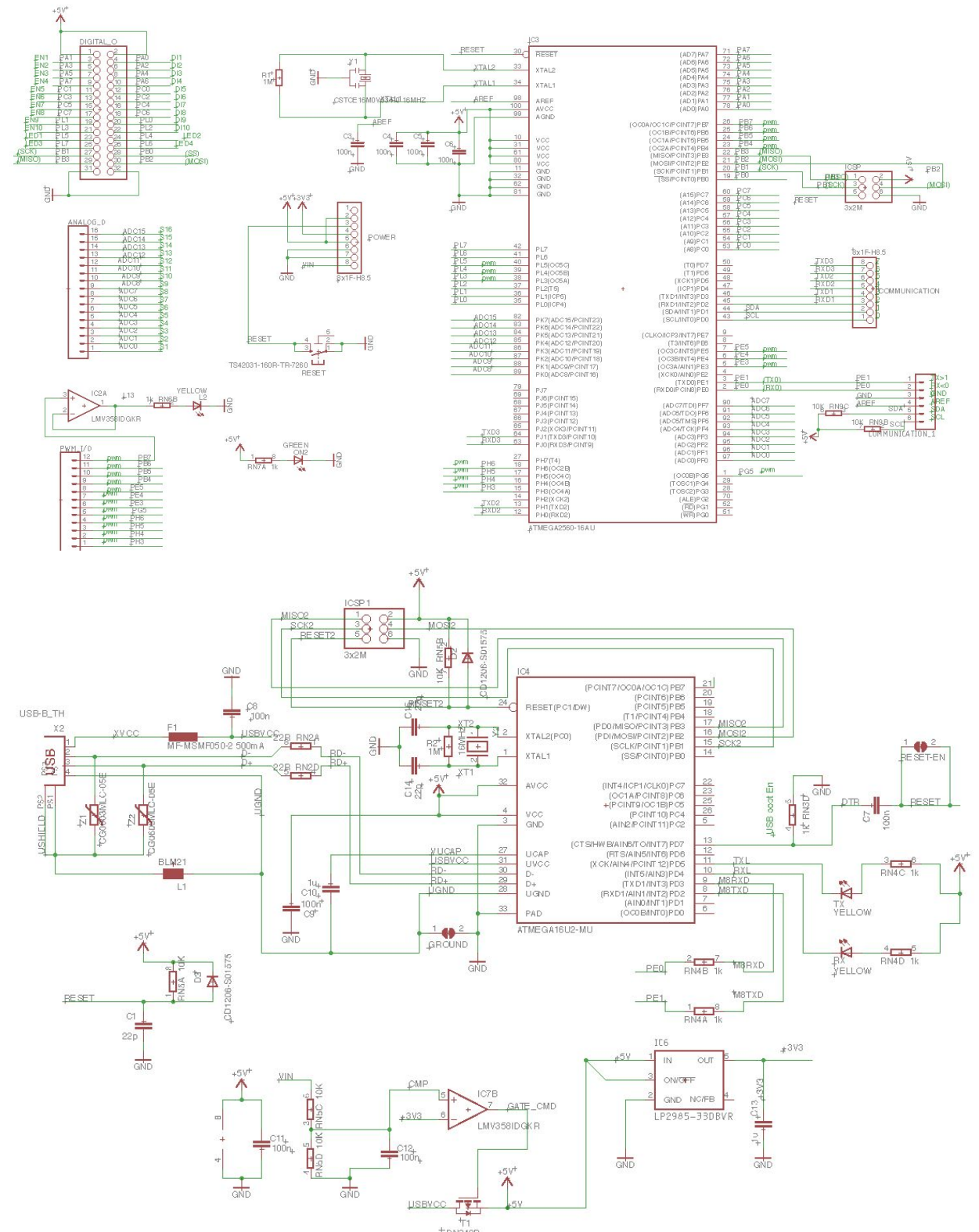


Figure 5.1.6-1. Signal PCB Schematic

5.1.7. PCBs Bill of Materials

This section will show the total bill of materials for the components used on the printed circuit boards utilized in our project, these boards are the power PCB as well as the signal PCB. The first bill of materials shown in Table 5.1.7-1 shows the components used in the power PCB as well as the quantity and cost while Table 5.1.7-2 shows the components used in the signal PCB as well as the quantity and cost.

Table 5.1.7-1. Power PCB Bill of Materials

Part	Manufacturer	Part Number	Qty	Price	Attribute	Value
Cbst	AVX	08053C104KAT 2A	1	0.01	Capacitance	100 nF
Cbyp	AVX	0805ZD225KAT 2A	1	0.03	Capacitance	2.2 uF
Ccomp	Kemet	C0603C512J5G AC7867	1	0.08	Capacitance	5.1 nF
Ccomp2	Samsung Electro Mechanics	CL21C151JBAN NNC	1	0.01	Capacitance	150 pF
Ccomp3	AVX	04025A101JAT2 A	1	0.01	Capacitance	1 nF
Cin	TDK	C2012X5R1V15 6M125AC	1	0.21	Capacitance	15uF
Cout	MuRata	GRM32ER71C2 26KE18L	1	0.31	Capacitance	22uF
Css	MuRata	GRM033R61A33 2KA01D	1	0.01	Capacitance	3.3 nF
Cvcc	MuRata	GRM219R71E10 5KA88D	1	0.02	Capacitance	1 uF
L1	Coilcraft	XAL6030-122ME B	1	0.65	Inductance	1.2 uH
M1	Texas Instruments	CSD17308Q3	1	0.25	Vds Max	30 V
M2	Texas	CSD17577Q3A	1	0.21	Vds Max	30V

	Instruments					
Rcomp	Vishay-Dale	CRCW04021K8 2FKED	1	0.01	Resistance	1.82kOhm
Rcomp2	Vishay-Dale	CRCW0402294 RFKED	1	0.01	Resistance	294 Ohm
Rfbb	Vishay-Dale	CRCW04021K3 7FKED	1	0.01	Resistance	1.37 kOhm
Rfbt	Vishay-Dale	CRCW080510K 0FKEA	1	0.01	Resistance	10 kOhm
Rpgood	Vishay-Dale	CRCW0402100 KFKED	1	0.01	Resistance	100 kOhm
Rs	Vishay-Dale	CRCW04024K6 4FKED	1	0.01	Resistance	4.64 kOhm
U1	Texas Instruments	TPS40305DRCR	1	0.85		

Table 5.1.7-2. Signal PCB Bill of Materials

Name	Manufacturer	Part Number	Qty	Price	Description	Value
Analog	Arduino	PINHD-1X16	1	1.19	Pin Header	
COMM	Arduino	PINHD-1X8	1	1.98	Pin Header	
C1	Samsung Electro-Mech anics	C-EU0603-RND	1	0.1	Capacitance	22pF
C3	AVX	C-EU0603-RND	1	0.01	Capacitance	100nF
C4	AVX	C-EU0603-RND	1	0.01	Capacitance	100nF
C5	AVX	C-EU0603-RND	1	0.01	Capacitance	100nF
C6	AVX	C-EU0603-RND	1	0.01	Capacitance	100nF
C7	AVX	C-EU0603-RND	1	0.01	Capacitance	100nF

C8	AVX	C-EUC0603	1	0.01	Capacitance	100nF
C9	AVX	C-EU0603-RND	1	0.01	Capacitance	100nF
C10	MuRata	C-EU0603-RND	1	0.02	Capacitance	1uF
C11	AVX	C-EUC0603	1	0.01	Capacitance	100nF
C12	AVX	C-EU0603-RND	1	0.01	Capacitance	100nF
C13	AVX	C-EU0603-RND	1	0.01	Capacitance	100nF
COMM_1	Arduino	PINHD-1X6	1	1.98	Pin Header	
F1	AVX	L-EUL1812	1	0.74	Inductance	500mA
IC2	Texas Instruments	LM358D	1	0.8	OP AMP	LM358D
IC3	Microhip Technologies	ATMEGA1280-16AU	1	12.44	Micro-controller	ATMEGA 2560-16AU
IC4	ATMEGA8U2-MU-ND	ATMEGA8U2-MU	1	2.12		ATMEGA 8U2-MU
IC7	Texas Instruments	LM358D	1	0.8	Op Amp	LM358D
IC6	Texas Instruments	TPS77033	1	0.42	Linear Regulator	LP2985-3 3DBVR
ICSP	Arduino	PINHD-2X3	1	1.92	Pin Header	ICSP
ICSP1	Arduino	PINHD-2X3	1	1.92	Pin Header	ICSP
L2	OSRAM	LEDCHIP-LED0805	1	0.25	LED	YELLOW
L1	Würth Electronics	WE-CBF_0805	1	0.2	Ferrite Beads	BLM21

ON	OSRAM	LEDCHIP-LED0 805	1	0.25	LED	GREEN
POWER	Arduino	PINH-1X6	1	1.19	Pin Header	
PWMI/O	Arduino	PINH-1X12	1	1.19	Pin Header	
R1	Vishay Dale	R-EU_R0603	1	0.1	Resistor	1MOhm
R2	Vishay Dale	R-EU_R0603	1	0.1	Resistor	1MOhm
R3	Vishay Dale	R-EU_R0603	1	0.1	Resistor	27Ohm
R4	Vishay Dale	R-EU_R0603	1	0.1	Resistor	27Ohm
RESET	ebay	TS42	1	1.17	TS42	TS42
RN1	Vishay Dale	4R-NCAT16	5	0.1	Array Chip Resistor	10KOhm
RN2	Vishay Dale	4R-NCAY16	5	0.1	Array Chip Resistor	22Ohm
RN3	Vishay Dale	4R-NCAY16	5	0.1	Array Chip Resistor	1kOhm
RN4	Vishay Dale	4R-NCAY16	5	0.1	Array Chip Resistor	1kOhm
RN5	Vishay Dale	4R-NCAT16	5	0.1	Array Chip Resistor	10kOhm
RX	OSRAM	LEDCHIP-LED0 805	1	0.25	LED	YELLOW
T1	STMicroelectr onics	PMOSSOT23	1	0.27	MOSFET	FDN340P
TX	OSRAM	LEDCHIP-LED0 805	1	0.25	LED	YELLOW
X2	Samtec Inc.	PN61729	1	0.93	BERG USB connector	

Digital_O	Arduino	PINHD-2X16	1	1.19	Pin Header	
Y1	Microchip Technology	RESONATORM U	1	0.89		16MHz
Y2	Microchip Technology	RESONATORM U	1	0.89		16MHz
Z1	Bourns Inc	VARISTORCN0 603	1	0.44	VarResistor	PGB1010 604
Z2	Bourns Inc	VARISTORCN0 603	1	0.44	VarResistor	PGB1010 604

5.1.8. Microcontroller

The microcontroller will be the main processor of the system essentially acting as the brain. It will be responsible for initializing the system, this entails providing all of the motors attached to the valves of the pipe with the corresponding inputs responsible for turning the valve a certain distance. This will allow the system to function in an automated fashion without any manual labor. The microcontroller will also be responsible for turning the system on and off essentially acting as a switch. After the system has been initialized and started the microcontroller will receive feedback from the pressure transducers. When the microcontroller receives feedback from the transducers it will calculate the pressure drops between them and compare the pressure drops to the corresponding desired values. If the desired values do not match the measured value the valves will be readjusted accordingly in an effort to adjust the pressure between the transducers. The microcontroller will also be responsible for flashing the LEDs sequentially in the direction of flow through the pipe. The speed of the flashing of the LEDs will be in correspondence to the flow rate through that branch of the pipe. In addition the microcontroller will relay the pressure received from the transducers to the Raspberry Pi 3 to be displayed on the graphical user interface by its corresponding component.

5.1.9. Raspberry Pi 3

The Raspberry Pi 3 is used mainly as the front end of the system. It is responsible for running the graphical user interface where the user can interact with the system. The Raspberry Pi 3 will also be responsible for relaying the corresponding components values from the graphical user interface to the microcontroller for calculations. The Raspberry Pi 3 supporting a full linux operating system will also be responsible for producing an ad hoc network for a client to connect to. This will allow for the client to remote desktop into the

Raspberry Pi 3 essentially mirroring the graphical user interface for an audience. In addition the Raspberry Pi 3 will receive transmissions from the microcontroller with information regarding the pressure of the transducers. This information will be displayed on the graphical user interface so the voltage drops across components are visible.

5.1.10. Light-Emitting Diode

The LEDs in the strips are configured in a cascaded method (shown in Figure 5.1.10-1), which transmits data sent by the microcontroller in a single line. This allows for a smooth flow of data transmission through the LED strip. Each pipeline representing the connection between the various components will have LEDs running behind it. The higher the current set by the user, the faster the LEDs will blink in a flow like manner to indicate direction. Each pipeline will be color coded to indicate where the current comes from and how is separated and/or merged after each branch. For circuit representations such as the BJT, the LEDs will represent the current relation between beta and the base current with respect to the collector current. The LEDs will provide a visual guide when it comes to current flow since in electronics this is something that is impossible to observe and is only possible to measure.

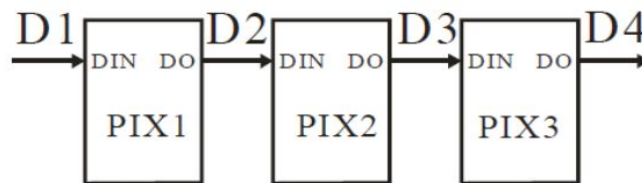


Figure 5.1.10-1: LED Cascade Method

Each LED has the following pin configuration displayed in Figure 5.1.10-2 and described in Table 5.1.10-1. This configuration allows for the LED strip to be cut at any one of the connections between one LED and the other. Since each LED had their own individual IC. They can be cut at any of the pins and be used separate from one another, as long as a connection is established via wire or connector. [58]

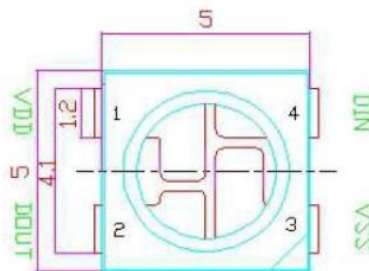


Figure 5.1.10-2: Pin Configuration

Table 5.1.10-1: LED Pin Configuration

Pin Number	Symbol	Function Description
1	V_{DD}	Power supply (4.5V~5.5V)
2	D_{out}	Control data signal output
3	V_{SS}	Ground
4	D_{in}	Control data signal input

5.2. Software Design

The software design is implemented to gather a full understanding of how the system interacts from a software standpoint of view. This section will outline how the software systems are to perform in the given conditions. It will state the variables and their importance. The software design will show the interaction between the software elements. It will show how the system as a whole will operate at a software level providing an overview of how the hardware components operate with the given software to form the entirety of the system.

5.2.1. Software Block Diagram

Referring to the block diagram below (Figure 5.2-1), the software and control systems side of this project can be understood. Each block represents a key element of the system. Blue blocks represent the software aspect of the labeled component (Section 5.1.1. diagrams the electrical hardware, power, and mechanical aspects of the system.) Blue arrows represent a digital signal being transmitted between components. Grey blocks represent mechanical aspects of the system. And lastly, black arrows represent a mechanical relation between components. Any labels next to an arrow describe the type of hardware or protocol used for the given communication. The Raspberry Pi served as the main hub of our system. It displayed the GUI through a touch screen connected through a Digital Serial Interface (DSI) port. From the touch screen, it received the system initialization preferences and settings for the rest of the system. This could also be accomplished via Virtual Network Computing (VNC) from a remote computer. The Raspberry Pi then sent all the data to the microcontroller via Inter-Integrated Circuit (I²C). The microcontroller then sent the appropriate direction and pulse signals to the drivers which controlled the motors for the user selected component. After the system was initialized, the water pump and LEDs were activated until the user reset the system from the GUI.

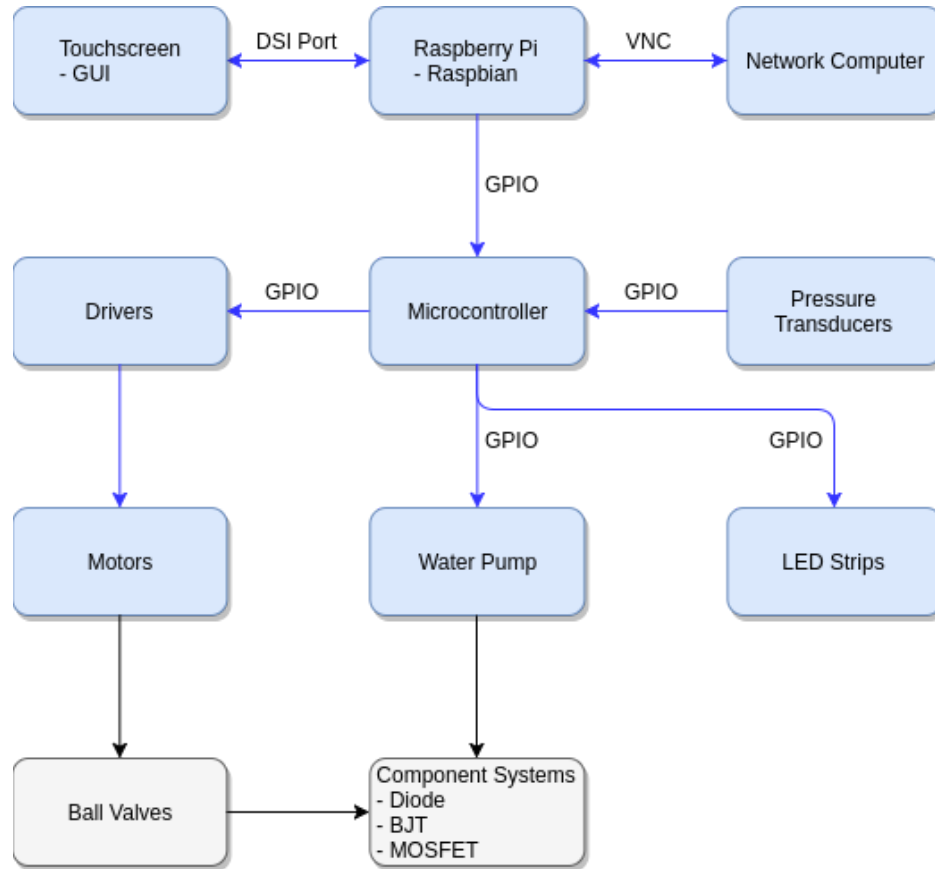


Figure 5.2-1: Software Block Diagram

5.2.2. Graphical User Interface

GUI development is a key aspect of designing our system. Unlike the rest of the software and a lot of the hardware needed to operate the system, the GUI is the key methods for a user to interact with the system (whether physically or through VNC). Because of this, the GUI was designed to be simple, lightweight, and with obvious, well-defined buttons.

One of the greatest constraining factors that we had to be aware of was the screen size and resolution. The touchscreen display was 7 inches with a viewable screen size of 155mm x 86mm and a resolution of 800 x 480 pixels. Because of the smaller screen size, we encountered the challenge of ensuring that our user interface was complete, intuitive, and operable with minimal errors. [zimg2] displays how we accomplished these goals. Firstly, we created a strip of buttons that ran along the left side of the GUI. These were created so that the user can access the various screens of the main menu. We wanted each button to have the same dimensions, so we experimented with various numbers of buttons and realized we could fit three buttons comfortably, where a user could select the button he or she desired without much precision.

The “Setup” menu is where the user initialized the system before running. This menu displayed a image of the system using circuit schematic symbols where their respective components were located. The user was then prompted to select one of the circuit symbols to modify that component’s parameters. For each component, there were a set of drop-down menus with preset values (to prevent user error) for each parameter of the component (e.g. for a diode, its V_T , V_{IN} , etc). Once each parameter was set, the user would be prompted to push “Start”. This would lock the screen until the user pushed “Stop” or “Reset” which would pause the system so the user could adjust the parameters or stop the system and return to the main menu respectively.

The “VNC” menu provided instructions on how initialize VNC for remote control from any WiFi-compatible device. It describes how to connect to the ad-hoc WiFi network the Raspberry Pi broadcasts and sign in using the WPA2 passphrase. It also instructs on how to install RealVNC Viewer and connect to the now-local Raspberry Pi VNC server.

The “Settings” tab allows the user to control other various aspects of the system.

Despite making all the menu screens and buttons as intuitive as possible, we wanted to ensure that the user had a means of finding instructional help on how to use the system if they need it, so we created a “Help” menu accessible from the top left of the screen. This menu offered general information about the system and its designers and a guide on how to configure and run the system, describing all of its menus, buttons, and results. We decided to make the “Help” button smaller than the rest of the other buttons and widgets because our object was to make the GUI as intuitive as possible to eliminate the need for instructions. Thus, since we felt we accomplished our goal, we felt justified in making the “Help” menu, a little harder to notice and access.

Finally, we created a startup shell script for the Raspberry Pi, so that whenever it boots up, the computer automatically launches the GUI for our system. This was intended to reduce the initial steps to turn on the system for the first time. We also eliminated all means for a user to close the GUI by locking it at the touch screen’s specific aspect ratio, 800 x 480 pixels, so that the GUI cannot be resized to access other parts of the Raspberry Pi. The only means to close the application is through an external keyboard which we prevented by physically blocking the Raspberry Pi’s USB ports.

5.2.3. Raspberry Pi and Microcontroller Interactions

The first step to communicating via I²C was to initialize the Raspberry Pi. This was done utilizing the module “smbus”. This import contains methods that handle interpreting the data read and translating the data to be written. On the

microcontroller side, we utilized the “Wire.h” header file which contained the functions needed for serial communications.

Whenever the “Start” button was pressed on the GUI, a method would retrieve the values inputted in the “Settings” tab to realize what data the Raspberry Pi needed to send to the microcontroller. Once it does, another method would retrieve the values of selected circuit component and transmit the “Start” constant then the component code along with the circuit parameters for the component to the microcontroller. Whenever the “Stop” or “Reset” buttons were pressed on the GUI, the Raspberry Pi would transmit the “Stop” or “Reset” constant to the microcontroller.

The microcontroller ran on a loop until it was interrupted by a serial communication from the Raspberry Pi.

5.2.4. Microcontroller Control

When the “Start” constant was received, the microcontroller would wait for the component code. It would use this code to choose with function to call. Each component had their own function which first stored the component parameters into local variables. After the values were stored, the microcontroller then used formulas relating circuit theory to fluid dynamics to determine how wide the valves of the system needed to be open. From there it calculated the amount of steps the stepper motor(s) needed to rotate in order for the system component to accurately represent the circuit component with the user selected parameters.

After performing the necessary calculations, a function that set a high output direction signal and output a pulse width modulation (PWM) signal to the component’s driver(s) to turn the motor(s) the calculated amount of steps in order to open the valve(s). When that function left its call, the next function looped a high signal to the water pump and controlled the LED display for the selected component.

When the “Stop” constant was received, the microcontroller would call a function that exited the water pump and LED loop procedure.

When the “Reset” constant was received, the microcontroller would first call the “Stop” function then it called a function that reset a low output direction signal and outputted a PWM signal to the component’s driver(s) to turn the motor(s) the previously calculated amount of steps in order to close the valve(s).

After deciding to use a Raspberry Pi to handle the user I/O through the touch screen display, and deciding to use Python to Since Python is an ever-changing programming language. With the major backward-incompatible release of Python 3.0 in late 2008, many of the previous platform-independent Python

toolkits for GUI development were outdated. According to the “Graphic User Interface FAQ” on the official Python website, the only known bindings to be Python 3-compatible are for Qt (using PyQt or PySide) and Tkinter [98]. With that insight, further research went into these two toolkits.

5.2.4.1. Tkinter

The advantage that Tkinter (Tk Interface, because it is an interface to the Tcl / Tk widget package) offers over GUI toolkits is that it is easy to install and setup since it is included when Python is installed. Python also provides information on how to use Tkinter in their documentation [99]. The Tkinter package, tkinter, is lightweight, cross-platform object-oriented code that is stable and fast [100]. For our purposes, Tkinter was the first choice because of the faint familiarity with the library. GUI development has been done with tkinter which would reduce the learning curve for creating an optimal GUI for this project. However, one of the major complaints is the complexity of the code beyond the simple model displayed in tutorials and documentation [101][102]. Despite the warnings about the intricacy of creating a multilayer GUI with tkinter, the toolkit was considered a forerunner for our GUI development because Tcl is also open source; it is freely usable without any licensing issues [100].

5.2.4.2. Qt

Qt (“cute”) is framework developed in C++ with language bindings for Python under the PyQt and PySide modules. The Qt Company which manages the software, provides Qt Designer to assist in GUI development when using the framework which is a great advantage since we would be able to drag-and-drop objects onto our GUI and would not have to manually code it up.

5.2.4.2.1. PyQt vs. PySide

Qt is another popular GUI toolkit, and the only other one with known compatibility with Python 3 according to the official Python side [98]. However, unlike Tkinter where there is only one binding, both PySide and PyQt are similar bindings to the same Qt toolkit. DrAl on the askubuntu forums gives a good comparison between the two mappings. As far as maintenance is concerned, both bindings are kept up-to-date but PySide is more frequently updated [103]. As Qt is updated, the documentation for both PyQt and PySide are auto-updated and generated based on the main Qt documentation [103]. However, from our observations, the documentation for PyQt5 (the current version) was clearer and simpler to understand. As for as licensing, PyQt is distributed under GPL version 2 or 3 or a commercial license [104]. On the other hand, PySide is LGPL-licensed software which allow for open source and proprietary development [105]. Despite the licensing differences (which would not affect our project), after turning to a stackoverflow question post, “PyQt or PySide - which one to use [closed],” the decision that using PyQt over PySide may reduce future software bugs because

of PyQt's more stable support for Python 3 according to the common consensus of the responders.

5.2.4.2.2. Qt Designer

One of the advantages to using Qt is utilizing Qt Designer. This software suite is a tool that is optimal for designing and constructing GUIs with Qt Widgets. Instead of either typing XML code manually (e.g. with using Tkinter) Qt Designer lets its users drag-and-drop layouts, spacers, buttons, containers, and other widgets into the project workspace [106]. Before saving the file, the creator has the option to preview the file within the Designer application to ensure that the user interface functions and looks as intended. Qt provides a QUILoader class which allows for quick and simple integration of the XML file into the program. The menu options for each widget, allows the developer to rename the classes and properties of each object in the GUI [107].

5.2.4.3. Conclusion

Both Tkinter and Qt are useful toolkits for GUI creation; however, since Qt Designer provides a drag-and-drop interface for Qt, the decision to utilize Qt (and more specifically PyQt for the reasons mentioned in section 5.2.4.2.1) was made [106]. Instead of exerting time and effort with learning how to write XML code, the time could be more effectively be used on improving GUI functionality and appearance when using software to generate the front end code.

5.2.5. Machine Learning

Machine learning will be used to allow the system to fine tune itself and find the corresponding inputs to reflect the desired output. The system will implement a supervised learning technique where available inputs will be provided to the microcontroller as well as the desired outputs. The inputs of the system will consist of the valves available for each component, while the desired outputs will be the pressure drops between transducers. The microcontroller will be the device physically executing each run and test, but the algorithm and data will be processed and stored on the Raspberry Pi 3. Each state of the system will be recorded and tested by the Raspberry PI 3 in connection to the microcontroller when the feedback from the transducers becomes available it will be plotted with the corresponding state and given a value of acceptance. The value will be better if closer to the desired result and worse as it gains distance from it. The algorithm will then process this data to find a pattern and ultimately reach its goal pressure drop (with a certain percent tolerance). The code will be developed in python as that is the recommended language on the Raspberry Pi 3 as well as favored by data scientists for the art of machine learning. The data set and results will be stored and processed on the Raspberry PI 3 allowing for updating of the algorithm as well as recalibration of the system. Overall this will provide the

system with a way to adapt to changes in the system and provide better performance. [108]

5.3. Design Summary

Hardware and Software design must be implemented together to achieve the overall desired system. The software design implemented on the Raspberry Pi 3 will cover the user interface using PyQt and essentially act as the front end for the microcontrollers software. The machine learning design will also be implemented on the Raspberry Pi 3 and will allow the system to interface with the microcontroller transferring and receiving feedback. The design implemented for the microcontroller is the bridging gap between software and hardware design. The design will cover communication with the hardware components and essentially being the central controlling unit of the system.

Mechanical design although not an extension of the fields of computer or electrical engineering is nonetheless vital to the system. The mechanical design needs to incorporate the proper dimensions and positioning to provide the user with an accurate representation of a circuit from a mechanical representation. The mechanical representation of the circuit will further inject into the electrical design of the system.

The electrical design of the system will further provide visual aid to the user. Step motor controlled valves will allow for current flow represented by water flow to be controlled and further derived by the microcontroller. The electrically controlled valves further allows for pressure drops across components to be set to accurately represent voltages in the system. The pump will act as the voltage source of the system initiating water flow and pressure on components. The pressure transducers allow for pressure to be converted into an analog voltage which can be translated back into pressure by the microcontroller and interpreted as feedback to allow for correction of the system. This is made possible by the software design on the microcontroller and Raspberry Pi 3.

The software design of the system provides the user with a visual representation of the circuit through the graphical user interface. The graphical user interface also doubles as the input interface for the user allowing the setting of values for the components of the system. The software design will also feature the ability to control the components and position them to the desired values using conversion formulas from desired outputs to degree alterations of the valves to steps of the motors themselves. The software design in combination with the ability to alter the system values implements a machine learning algorithm featuring the ability for the system to learn from itself on a prior knowledge of faults and successes. This allows for the system's ability to function in combination with electrical and mechanical designs.

Combining mechanical electrical and software designs for the system the overall system design acts in coerts to provide the user with the overall experience desired by the developers. This is the overall goal of the system to correctly resemble a electrical circuit in a mechanical representation. While the system is intricate and technical from an engineering standpoint it boils down to the experience of the user and performance in the field of providing accurate visual representation to further the user's learning.

6. Prototyping

Prototyping is a very important part of the development process. It allows for developers to visualize the system as a whole without actually having everything finalized. This provides the ability to add, test, remove, and replace components to achieve optimum performance throughout the system before setting the final design in stone. Prototyping also allows for the development of components which depend on systems which may be in progress or not fully completed yet allowing for a more agile development process.

Figure 6-1 represents the current signal and power prototype. The breadboard represents the power distribution, using a DMM supplying 12 V, representing one of the rails in the power PCB that supplies power to the motor driver and the water pump. A voltage divider is used to drop the voltage to 5V representing the other rail in the power PCB that supplies power to the Raspberry Pi, microcontroller, LED strips, and pressure transducers. However, since the power comes directly from the outlet and not a DC battery or DMM, the power PCB manages the power from each one of the components and is explained in section 5.1.5. All the red lines are the power supplied to each one of the components, and all the components are commonly grounded (black/gray wire).

The purple line connected from the driver to the microcontroller serves as the direction signal, the blue is the enable line, and the orange the pulse. The green line from the microcontroller to the LED strip controls the data I/O. The neon yellow line from the microcontroller to the transducers reads the analog voltage output due the various pressures. The yellow and white lines from the microcontroller to the Pi establishes I2C communication between the two which outputs the readings and data to the touch screen, and allows the user control over the system. The development board is used to develop the signal PCB and edited to fit and allocate all the components in the system. The set-up in figure 6-1 would be replicated for all the components in the system. The connectors are strategically located so the signal lines going to each components are easier to install.

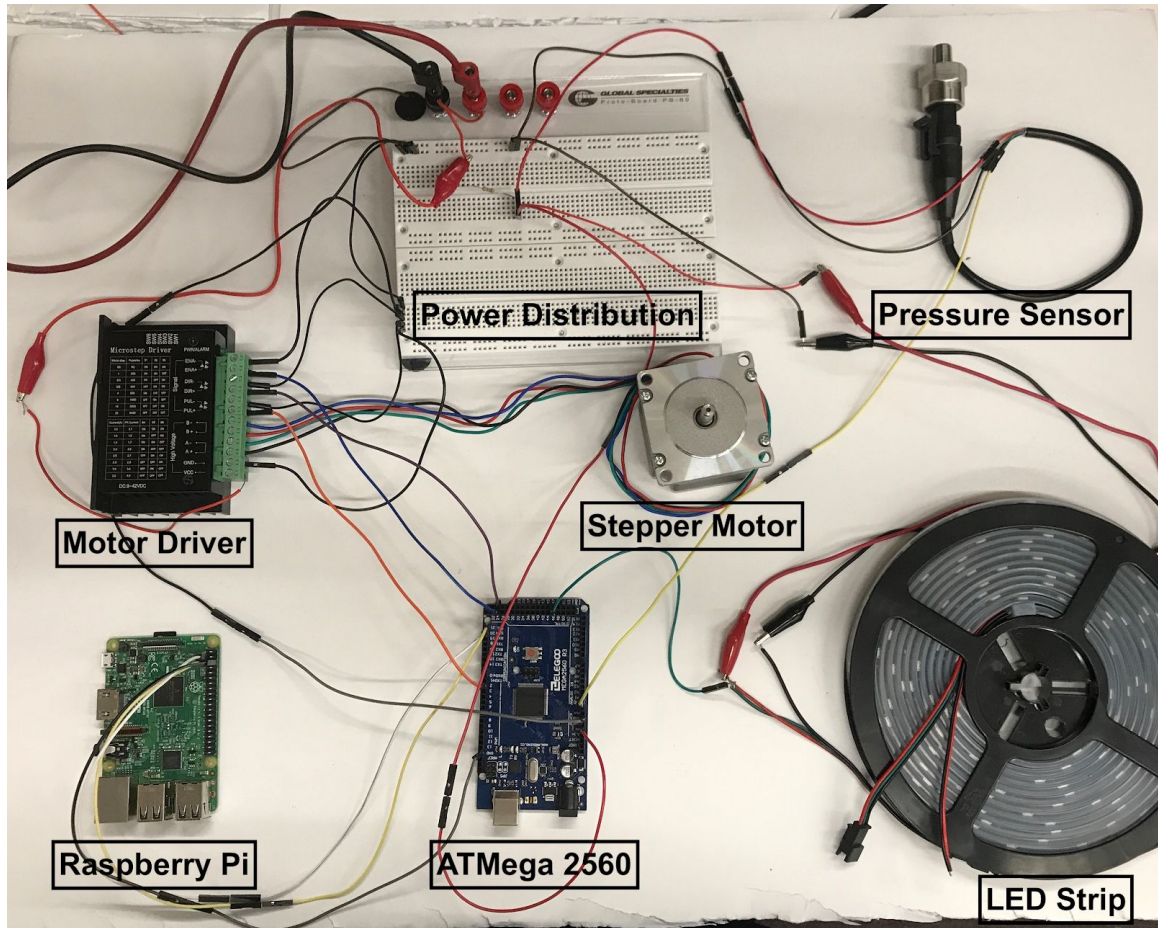


Figure 6-1: Signal/Power Distribution Prototype

6.1. Printed Circuit Board

Sections 5.1.5 as well as 5.1.6 show the schematic layouts of the power and signal PCBs respectively. However actually laying out these designs on a physical PCB, getting the PCB printed from a manufacturer, and populating the respective PCBs with the correct components is a task that is still to be completed.

6.1.1. Power PCB Design

Section 5.1.5 shows the schematic of the circuit to be implemented on the power PCB as well as displays a list of the components that will be used to populate this PCB. Using this schematic as well as the component libraries available in the Eagle PCB design software the signal PCB will be laid out in Eagle and this file will be sent to a PCB creation company for fabrication.

6.1.2. Signal PCB Design

Section 5.1.6 shows the schematic of the circuit to be implemented on the signal PCB as well as displays a list of the components that will be used to populate this PCB. Using this schematic as well as the component libraries available in the Eagle PCB design software the signal PCB will be laid out in Eagle and this file will be sent to a PCB creation company for fabrication.

6.1.3. Power and Signal PCB Fabrication

This section will be utilized to display the completed power as well as signal PCBs after they have been fabricated and delivered back to us at a later date.

6.1.4. Power and Signal PCB Population

Sections 5.1.5 and 5.1.6 list the components necessary to populate the Power PCB and Signal PCB respectively. These components have been ordered from DigiKey as well as Coilcraft.

6.2. Software

Prototyping software is a very important to the software development process. In nearly all current software development methods, they implement some type of prototyping. This section will outline the different prototypes implemented for the system. It will show their purpose, how they function and the insight gained from their application. It will also go into detail on how the prototypes were built. The prototypes will be evaluated and determined which components will be implemented into the final build for each segment of the system.

6.2.1. GUI

The graphical user interface prototype was implemented to determine the required fields for each component, as well as a functional layout for the user to interact with. The prototype would provide an initial starting point to build off of and later implemented into the system.

The original prototype was written using a GUI library in python called Tkinter. It had 4 main pages, initialize, start/stop, VNC, and settings. The initialize tab had input text boxes where the user could choose any value they would like for each of the components. The start/stop section of the application would display the circuit with the values being pulled from the initialize tab, this would allow the user an overview of the system before deciding to start or stop the system. The system would be powered off and on using two buttons (start and stop) located on this page. The VNC section of the application provided detailed instructions on how to setup remote viewing of the graphical user interface. The settings

page provided settings specifically for the graphical user interface, it allowed the user to toggle full screen as well as close the application.

Upon evaluation of the original prototype it was decided to switch away from Tkinter in favor of PyQt. PyQt offered more functionality as well as a nicer overall appearance to the application. In the revision of the graphical user interface drop down menus were also implemented over the ability for the user to enter any value for the components. This feature was implemented to prevent the user from breaking the system with infeasible values. The ability for the user to exit full screen and the application was also removed to prevent the user from altering the Raspberry Pi 3's operating system due to security purposes. The initialization and startup page were also combined to allow the user a more straight forward experience when interacting with the system over separate areas.

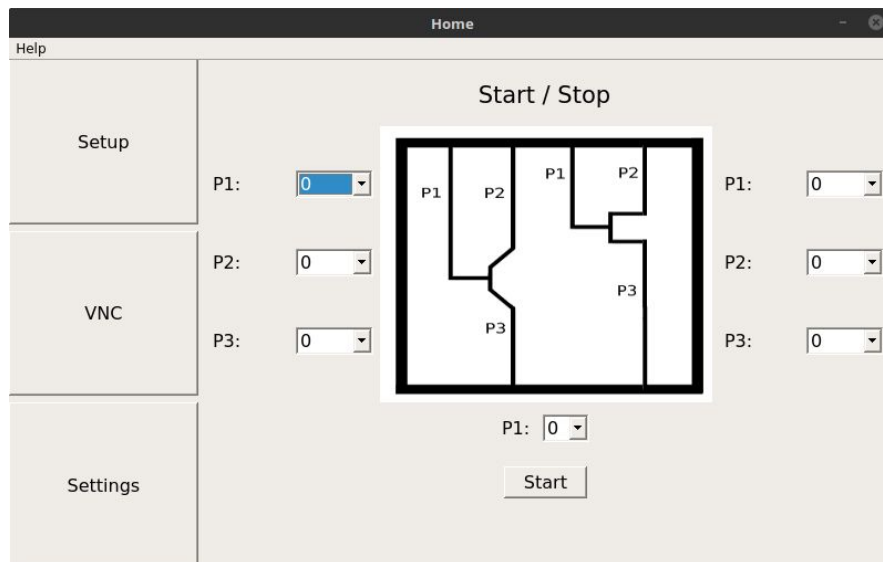


Figure 6.2.1-1: GUI Prototype

6.2.2. Atmega2560 Stepper Motor Functionality

In order to demonstrate the ability for the microcontroller to turn the motor a desired distance. Designing this application a code block was implemented on the microcontroller which would take an integer corresponding to the number of steps which the motor should be turned. The code also implemented a boolean type variable which corresponded to the direction which the motor would turn, and an integer variable to determine the speed at which the motor should be stepped. A final boolean variable was used to communicate with the driver motor when it should be enabled. On the hardware portion of this prototype there are three wires shared between the microcontroller and the motor driver. The enable wire is responsible for communicating that the motor driver should be enabled. The directional wire is responsible for communicating with the motor driver as to

the direction in which it should be turned. Finally, the pulse wire will communicate with the motor driver as to trigger a step to the motor.

On implementation of coded application the microcontroller was able to enable the motor driver and trigger a step in the proper direction. With this application the motor was able to turn smoothly despite it's stepping function. The microcontroller was able to change direction of the motor with minimal delay. Upon implementation the microcontroller was able to turn the motor at a very high speed, this rose the question as to which speed the motor should be turned due to the torque drop off as speeds increase, further testing would be required to find this.

6.2.3. Atmega2560 Transducer Functionality

The Atmega2560 is responsible for receiving voltages from the pressure transducers. The pressure transducers will convert the pressure at a given point in the system to analog voltage which the microcontroller can read from an analog input pin. The Atmega2560 will read from the input pin it will then convert the corresponding voltage to a pressure and pass that pressure using i2c protocol to the Raspberry Pi 3 for display on the graphical user interface.

The analog input for the Atmega2560 is capable of reading voltages up to 5V. The analog input reads voltage for 1024 units. This allows for differentials in .0049 V giving a very high accuracy in analogread executions. The analog pin is capable of reading at a rate of 100 microseconds or up to 10 thousand times per second. This will allow for a high rate of sampling for the application.

The transducer functionality is very important to the system because it will be responsible for gathering data for the system. The data calculations for the application will come from a machine like testing phase. The system will be given starting values from analysis of the system. The desired pressure drop between pressure gauges will be hardcoded into the application. When the microcontroller receives the pressure converted to voltage values it will compare those values to those hard coded values. The microcontroller will then turn the valves in an attempt to correct the pressure drops and get it closer to the desired value. If the value ends up in a worse position than it was in the previous position the values are rolled back to the previous state and the tried state is marked as worse. If the new position produces a better result the state is updated and the state is saved for the next iteration. This will allow the system to find the optimum position of the valves for each case in which the system will run. This will require many hours of runtime for each test case of the system. The wear on the system requires careful consideration as to the integrity of the overall application. With this it will eliminate a lot of the theoretical equations from the system but implements other issues. [109]

6.3. Prototype Expectations

Prototypes will be expected to perform every function of the system in an intermediate state. This means that the prototypes do not need to be the final implementation or even implement everything in one system, but instead need to be able to display all the functions of the system. This will show the ability for the system to operate in the way described in this document without finalizing everything. To show the ability for every function of the system to act correctly and efficiently without final implementation.

6.3.1. Potential Hardware Issues

Potential hardware issues include malfunctions from both the power PCB and the signal PCB. If either PCB malfunctions it can damage other components of the system raising various problems since the system is interconnected. Component failure is always a concern as if a component fails it can cause for inaccurate reading, malfunctions in the system, and overall failure of the system. IF a component fails it will have to be replace before the system can be deemed reliable again.

6.3.2. Potential Software Issues

Software issues can arise when code is not implemented correctly or if not tested properly. Software issues can result in hardware issues if the problems output corresponds to a hardware component. If a communication line goes down between components they will no longer be able to provide feedback or interact with one another. Another fragile part of the software of the system is the operating system of the raspberry pi 3. If the operating system is not shutdown correctly and becomes corrupt it will render the system useless as the front end is run through that operating system. Software issues can also extend to things such as the ad hoc wireless network with if malfunctioning would render the remote desktoping application of this system unusable. The VNC protocol also has to be carefully considered as upon malfunction would also render this portion of the system useful. A wide variety of prototyping and testing will be implemented as to prevent such scenarios from occurring.

7. Testing

Testing is an important part of the system. It will allow the project to become as accurate as possible. Testing will allow for a reliable system to be formed and to run as intended. Both hardware and software will be tested in this section. The hardware will be tested to ensure the components work as intended while the software components will ensure that the code runs properly and is as optimized as possible. Both hardware and software testing is very important and will provide feedback on how the overall design will work as a whole.

7.1. Hardware Testing

Hardware testing is the application of implementing the design and testing its functionality. The section will outline the positive and negative results of each test, and what can be learned. It will go into detail on how the testing is performed and what was hoped to be achieved by each. Testing is one of the most important part of design as it assures positive results upon final implementation. Through these testing scenarios the reward is a system with a very high performance rate.

7.1.1. Microcontroller Testing

The microcontroller testing on a hardware level is very important. It will cover 3 main sections, code implementation, communication, and IO. Code being the driving force behind the microcontroller can not be separated completely from the hardware testing. Drivers and stubs will be used to perform tasks that the software would normally. Communication will cover how the microcontroller is able to communicate to and from the raspberry pi 3. Finally IO will include testing all of the relevant outputs and inputs which the microcontroller will need to perform in the system. This involves outputting PWM, standard high and low, and reading analog input voltages.

7.1.1.1. Code Implementation

The beginning stages of testing the microcontroller involved validating the execution of code on the ATmega2560. This was performed by using by using modified example code [27]. The code verified the execution of code on the processor and that the input and output pins are working correctly.

7.1.1.2. Communication: Raspberry Pi 3 - ATmega2560

Testing the communication from the Raspberry Pi 3 to the ATmega2560 involved modifying example code [110]. This provided us with the ability for the Raspberry Pi 3 to communicate to the ATmega2560 through parallel communication reading high and low values passed through the GPIO pins. The microcontroller was able to read the data passed through the GPIO pins and process them into meaningful variables such as motor number, number of steps and ready state.

The task then formed to test out serial communications from the Raspberry Pi 3 to the ATmega2560, for more information on why this became a requirement please refer to section 4.4 Wired Communication. Upon implementation of I2C communication we were able to pass values from the Raspberry Pi to the microcontroller through a serial communication protocol.

7.1.1.3. ICSP Programming

In-circuit serial programming was tested by wiring a slave and a master Atmega2560 together to successfully program the slave microcontroller. The master Atmega2560 was loaded with the arduino isp sketch provided by arduino and the slave microcontroller was wiped clean. The master was then wired to the slave using the 5V, Ground, MISO, MOSI, SCK, and Reset pins. Once the two microcontrollers were wired together, arduino studio has the option to burn a bootloader or upload a sketch using a programmer. Using the programmer tool under arduino studio the master Atmega2560 acts as a middleman between the computer and the slave Atmega2560. Using this method it was possible to both upload a sketch to the slave device and burn a bootloader to the slave device over ICSP Programming.

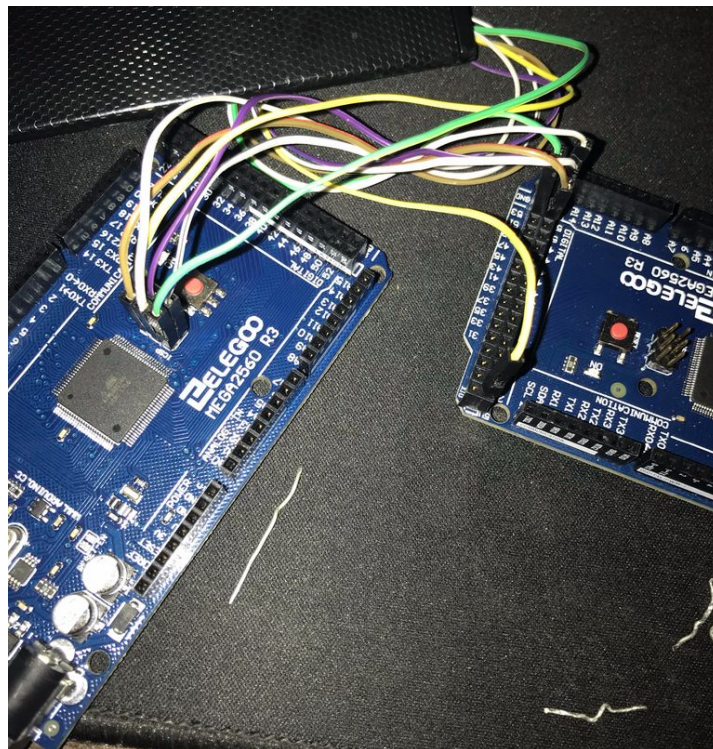


Figure 7.1.1.3-1: ICSP Testing (Slave Left / Master Right)

7.1.1.4. Outputting from the ATmega2560

Once communication between the Raspberry Pi 3 and the Atmega2560 was accomplished, the task then became to perform corresponding outputs to the components. Each motor driver was given three GPIO pins from the Atmega2560, an enable pin, directional pin, and a pulse pin. The enable pin was wired to the enable slot on the driver which would allow for the driver to provide output to the motor. The directional pin is wired to the directional slot on the driver, if a low voltage was supplied to the directional slot the motor would step in a counterclockwise rotation, if a high voltage was supplied the motor would step in a clockwise rotation. The pulse pin is wired to the pulse slot on the driver, a high voltage immediately followed by a low voltage signals one step for the motor. The motor was tested in both clockwise and counterclockwise directions at with ten, five, two, and one millisecond delays between high and low voltages. With the driver set to four-hundred steps to equal one full rotation and a one millisecond delay between high and low voltages we were able to rotate the motor one full rotation in eight-hundred milliseconds or 0.8 seconds. Giving a great response time for rotating the valve in the system.

7.1.2. Pressure Transducers

Pressure transducers were tested by supplying a 5V DC to the power line (red wire). The output line (green wire) voltage was measured. When the sensor is in idle and output voltage of 0.5V was measured. An air pump with a digital setting was used to apply a known pressure to the transducers to verify the accuracy and the voltage levels that correlates to each individual pressure. However, these values may be different when measuring water pressure to air pressure. See the table below containing the pressure to output voltage relation.

Table 7.1.2-1: Pressure to Voltage Relation

Applied Air Pressure	Measured Output Voltage	Datasheet Output Voltage
0 PSI	0.46 V	0.50 V
3 PSI	0.86 V	0.93 V
4 PSI	1.01 V	1.00 V
5 PSI	1.22 V	1.25 V
6 PSI	1.31 V	1.34 V
7 PSI	1.40 V	1.50 V
8 PSI	1.49 V	1.60 V

9 PSI	1.59 V	1.75 V
10 PSI	1.84 V	1.77 V
11 PSI	2.09 V	2.00 V
12 PSI	2.20 V	2.25 V

7.1.3. Motor Testing

The bipolar stepper motor was tested by simply connecting the motor to a power supply available in the senior design lab. As stated in the bipolar stepper motor research section 4.7.4.2 the bipolar stepper motor chosen by us has 4 output wires, the driving sequence of the motor we chose is displayed below in Figure 7.1.3-1. Using the power supply with the output voltage set to 12 volts, the wires were energized in the proper pattern to rotate the motor clockwise to confirm the functionality of the motor. This test was performed to make sure that the motor functioned before being connected to the bipolar stepper motor driver so if the motor no longer functioned correctly when connected to the driver it could be concluded that it is in fact the driver that is malfunctioning.

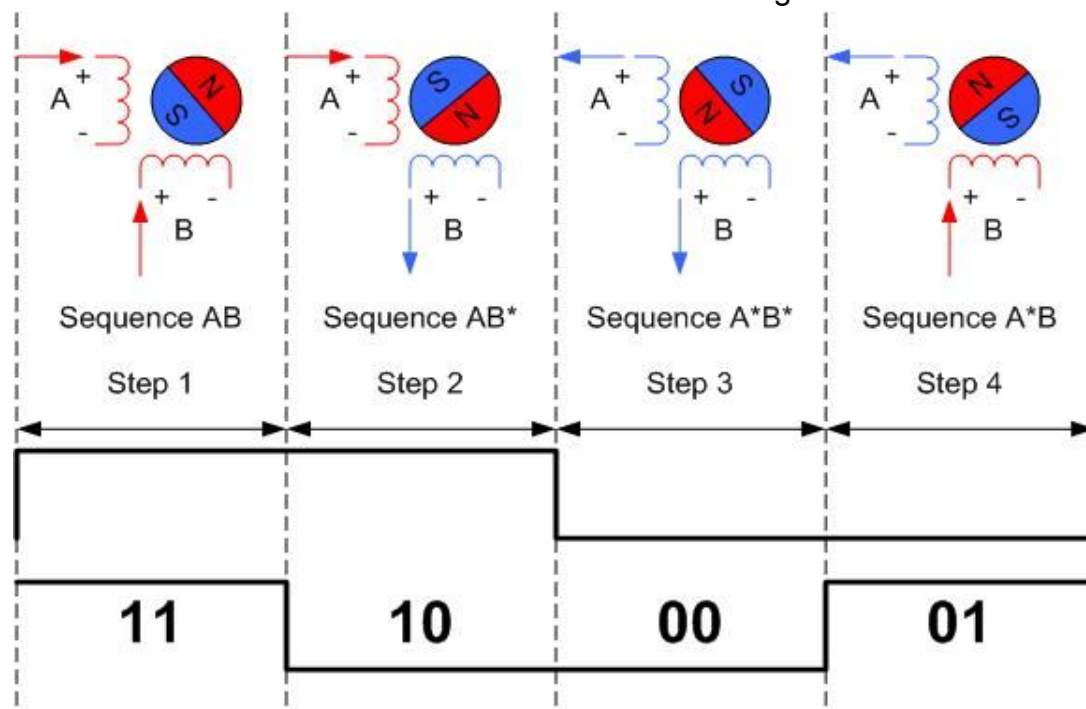


Figure 7.1.3-1. Bipolar Stepper Motor Clockwise Driving Sequence [111]

7.1.4. Motor Driver Testing with Function Generator

The driver was tested by connecting the PUL (pulse) terminal to the function generator, the negative PUL and DIR (direction) terminals to the common ground. The high voltage terminals connect to the motor and the supply voltage.

A 12 V DC was supplied to the Vcc, the blue wire from motor was connected to the B- terminal, the red wire to the B+ terminal, the green wire to the A- terminal, and the black wire to the A+ terminal. The connection of the motor wires establishes the direction of the rotation. A sinusoidal input was supplied to the positive PUL terminal to control the signal. A current of 1.4 A is drawn from the power supply when motor is operating. This caused the motor to rotate clockwise, supplying the sinusoidal input to the negative PUL caused the motor to rotate counterclockwise. See figure 7.1.4-1 for test set-up.

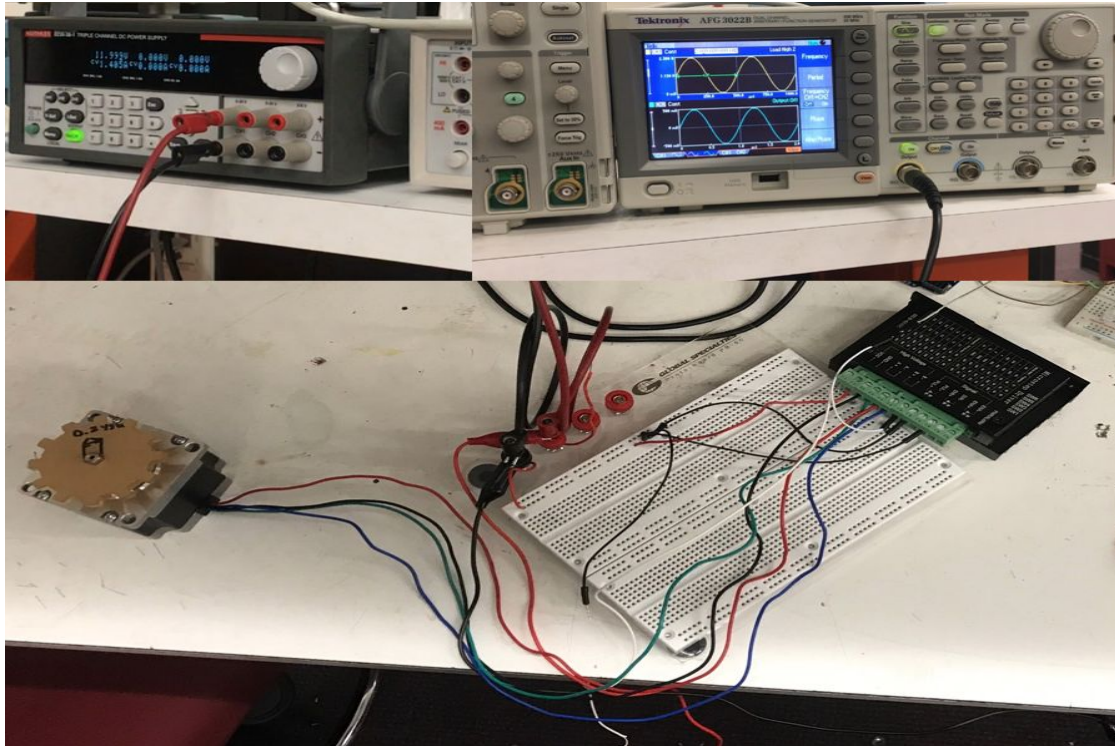


Figure 7.1.4-1: Motor Driver Testing

7.1.4.1. Motor Driver with Microcontroller

After the motor and driver were successfully implemented using the signal generator it was then tasked to implement the motor and driver using the microcontroller. The microcontroller was given three outputs and a ground to control the driver. The enabled wire was attached to a general purpose IO pin which was responsible for enabling and disabling the driver through the drivers enable pin, a high voltage would enable the driver and a low voltage would disable it. The directional wire was attached to a general purpose IO pin which was responsible for controlling the direction in which the step motor would rotate, this was implemented by attaching the directional wire to the directional pin on the motor driver where a high voltage corresponded to a clockwise step and a low voltage corresponded to a counterclockwise rotation. The pulse wire was attached to a general purpose IO pin capable of pulse width modulation and was responsible for sending a pulse of a high voltage followed by a low voltage to

signal one step, the wire was attached to the pulse pin of the motor driver. Once driver was successfully able to turn the driver, it was then strived to configure the microcontroller for optimum timing of each step. From previous testing with the function generator it was confirmed from the datasheet that the shorter interval between each step the less power each step was capable of. This posed the problem of finding the best zone for speed performance and power performance of the motor. See the image below for the test set-up.

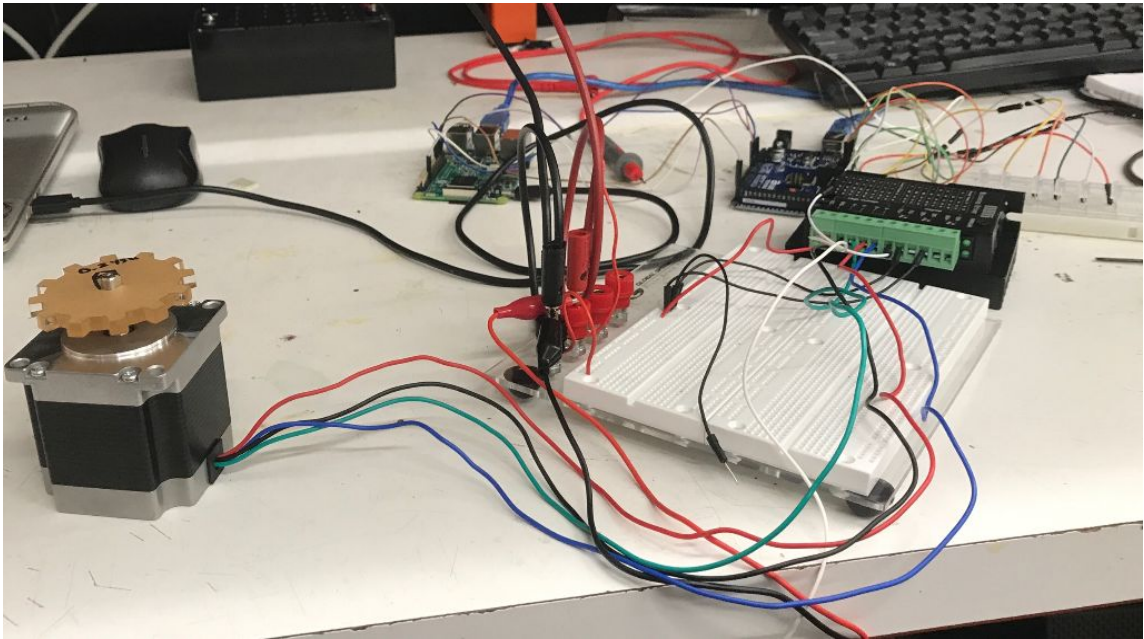


Figure 7.1.4.1-1: Motor Driver with Microcontroller Testing

7.1.5. Wireless Module Testing

The internal wireless module of the Raspberry Pi 3 (Broadcom BCM43438) is barely visible to the naked eye. The original fear was that the range of the chip would not extend far enough to provide good connectivity to the client due to its small size and lack of an antenna. Upon further research it was made known that the antenna of the chip is integrated into the board to allow the Raspberry Pi 3 for better connectivity. Testing the connectivity of the Broadcom BCM43438 first required configuration of the ad hoc network. The chip was setup to operate at 2.4GHz using 802.11n protocol. The first test case for the wireless module included connecting to the chips wireless network in an unobstructed area to find the optimum range of the device. Two main client devices were used, the Galaxy S7 smartphone and the Dell Inspiron 13 notebook. In an unobstructed scenario the Galaxy S7 proved reliable data transfer to and from the Raspberry Pi 3 through 40 feet, while the Dell Inspiron 13 proved reliable data transfer through 30 feet. In these ideal conditions the ranges proved to be more than required for the system. The next test case included testing unideal situations including multiple obstructions between the client and BCM43438 chip. With multiple

obstructions in play the Galaxy S7 and Dell Inspiron 13 both proved to have successful data transfer to and from the Raspberry Pi 3 for up to 20 feet.

With the gathered data from the testing of the BCM43438 chip it was deemed fit to handle the requirements of the system. This is due to its simple integration and its ability to transfer data reliably from a distance. In application the system will be implemented in classrooms. The tests producing an unobstructed range of up to 40 feet allows for plenty of leeway on the positioning of equipment. [112]

7.1.6. Raspberry Pi and Microcontroller Interaction

Integrating the Raspberry Pi 3 and the Atmega2560 comprised of three major test cases. Test case 1 involved creating software on the Raspberry Pi 3 to pass variable values to the Atmega2560. This test case will provide the ability for the graphical user interface running on the Raspberry Pi 3 to transmit component values to the Atmega2560 for calculation. Test case 2 involved communicating feedback from the Atmega2560 to the Raspberry Pi 3. This test case will be used to communicate feedback from the pressure transducers back to the Raspberry Pi 3 for display on the graphical user interface. Test case 3 is to integrate test case 1 and 2 simultaneously to ensure no interference in communication.

Test case 1: Test case 1 was implemented by creating a python application on the Raspberry Pi 3 capable of transmitting data to the Atmega2560 using i2c protocol. The Atmega2560 was coded to receive the transmission and output the corresponding integer through the output LED. The test was successfully implemented and the Atmega2560 was able to receive data from the Raspberry Pi 3.

Test case 2: Test case 2 was implemented by writing a python application capable of receiving a transmission over i2c and outputting it to the screen. The Atmega2560 was implemented using code to pass an integer through i2c. Upon execution of the test case the Atmega2560 was able to reliably transfer data to the Raspberry Pi 3.

Test case 3: Test case 3 was implemented by combining the software from test cases 1 and 2. The software was modified to transmit an integer from the Raspberry Pi 3 to the Atmega2560. The Atmega2560 processes the integer and then retransmits the integer back to the Raspberry Pi 3 for displaying. The test case was successfully implemented and the devices were able to transmit data back and forth between each other without interference.

7.1.7. LED Matrix Testing

Light emitting diode matrix testing involved the WS2812B LED Strip. The WS2812B involves 3 main wires vin, control, and ground. Vin is powered by a 5 volt DC voltage. For the testing of the LED strip's vin wire was connected to the

Atmega 2560's 5 volt pin, the control wire was connected to a general purpose input output pin, and the ground wire attached to a ground pin on the microcontroller. The microcontroller implemented the NeoPixel library which is a C++ library used to interact with the LED matrix. The library supports a combination of C++ code as well as interfacing with the LED driver's assembly code. The LED matrix supports RGB color coding meaning it supports all types of different colors, for this test it is only necessary for the matrix to output a white display from the LED as that is what will be used in the system. The microcontroller was coded to provide 5 volts to the LED strip and controlled the output of each individual LED by using the general purpose input output pin. The leds are triggered to display by first initializing each LED using the setPixelColor function before executing the show method to display the initialized LEDs with the corresponding color.

The code was designed to increment from zero to the last LED in the desired range. The function will also take a directional variable which allow for the lights to blink in the corresponding direction of current of the system. The function executes by setting the the corresponding color and executing the show function for that LED. The function then clears the shown leds and increments forward and repeats the process. This allows for the visualization of the current flowing down the corresponding branch of the circuit. The speed at which the LEDs are incremented can be altered through a variable input the the function. Overall the microcontroller is able to increment and turn on and off the LEDs in an efficient and effective rhythm between.

7.2. Software Testing

Software testing allows for software to be confirmed valid without actually having all the components running in the system. The software will undergo manual testing as well as automated testing. Automated testing will allow for combinations to be process to ensure success of each object. The software will also be tested upon interaction with other software devices. This will be performed first using stubs and drivers in case the other software component is not finished. Upon completion of all of the software components they will be tested to ensure their interactions are both fluid and reliable.

7.2.1. Software Testing Overview

In order to verify accurate and precise performance each aspect of the system, we first created unit tests to evaluate each method and function throughout our code. This would ensure that each low level component of our code produced the desired results. Then we created test cases that checked each class and structure used. These functional cases evaluated each object we created. Then, we assessed the entire code for a single device using mock objects to represent external components of the system relevant to the device being tested. Our final

test involved putting each component together and running the system as a whole.

7.2.1.1. GUI

As we were developing the GUI (the main window and dialog boxes with layouts, spacers, buttons, item viewers, containers, and input and output widgets placed in them), we used Qt Designer's preview mode under its "Form" (Ctrl + R) menu to test the frontend code. When this is selected, Qt Designer opens the .ui project file. It then takes this file and generates the C++ code from it. After compiling the code, it will load the executable and display the GUI without any backend programming. Since there is no functionality implemented in the backend, Qt Designer provides an interface to switch between the various pages in containers if any widgets are hidden. This is how we ensured that the frontend was designed and formatted correctly.

In order to test the backend, we created a testing script in the Python language. This script used called methods from the GUI initialization script to configure the backend of the GUI. In order to verify that every button pushed, a record of every widget interaction was displayed on the console. Whenever the "Start" button was pressed under the "Setup" menu, all the data from input widgets was compiled into lists and outputted to the console so that we could easily compare our input values to what the GUI recorded.

The last test was to verify that the user interface was compatible with the touch screen and the Raspberry Pi. In order to test this, we used VNC to connect to the Raspberry Pi so that we could control the mouse pointer. Even though the touch screen was connected to the Raspberry Pi, the GUI script could not run (during the testing stages) without using the terminal. Once the GUI loaded, we verified that the aspect ratio of the GUI matched the screen seamlessly. We also practiced navigating through the windows and menus to ensure that the Raspberry Pi could render the GUI trans smoothly that a user could not leave the application by using a brute force method. We also checked the aspect ratio of all of the buttons and widgets to ensure that they were small enough to not clutter the screen but big enough for a finger to flawlessly interact with.

7.2.1.2. Machine Learning Testing

Machine learning being a very interecruit part of the overall system it is very important to run tests. Testing of the machine learning algorithm will ensure the algorithm is properly bringing in feedback and reacting correspondingly to further improve the calibration of the system. While this section is titled machine learning testing it should be titled machine learning teaching as the algorithm most probably will not be perfect upon implementation. Alterations to the code will need to be implemented to further allow for the system to learn from itself.

Success of this testing section will be based on how well the system is able to calibrate itself to achieve its desired outputs. Performance will be based on how quickly the algorithm is able to come to these conclusions. At worst case the algorithm will test all combinations randomly until the desired output is achieved, but in reality the goal is to provide the algorithm with a thinking like manner as to which combination should be implemented in the next attempt. This allows for redundant combinations with a high probability to achieve an output not in correlation with the desired value to be overlooked and therefore no need to waste time on attempting these values.

7.2.1.3. I²C

In order to test the I²C communication between the Raspberry Pi and the ATmega2560, we created a test script in Python for the Raspberry Pi which initialized a serial connection with a specific address that referenced the ATmega2560 slave. It then allowed us to input data through the console which the Raspberry Pi would transmit whenever we hit "Enter" to the ATmega2560 using the I²C communication protocol. Whenever the Raspberry Pi received data, it would automatically display it on the console. On the microcontroller side, we flashed it with a program that initialized a serial connection with a specific address that reference the Raspberry Pi master. When the program ran, the ATmega2560 would store any data received from its data line in a variable. When the transmission was complete, it would send the data in the variable back to the master via the data line.

We were able to verify the validity and consistency of the data transmission via I²C between the two devices by making sure that the data we sent from the Raspberry Pi was identical to the data it received and displayed on the console.

7.2.2. Analog Input Testing

Analog inputs will be used to receive information from the pressure transducers. The pressure transducers convert the water pressure inside of the piping of the system to an analog voltage and transmit that voltage to an analog pin of the microcontroller. The information will be processed and adjustments will be made to the valves to achieve the desired pressure values. Testing the analog inputs involved hooking a DC power supply to the analog input pin and a segment of code which could output the corresponding read value for the input pin. The code block read the value from the input pin storing it as an integer. We know from the datasheet that the analogRead function retrieves the voltage and represents it as an integer from 0 to 1023 in correspondence to 0 to 5 volts. The code then converted the the integer value to blinks of the integrated LED to represent the retrieved integer. [113]

Table 7.2.2-1: Analog Input Measurements

Voltage	Measured (Integers Round Up)	Calculated
0V	0	0
1V	206	204.6
2V	415	409.2
3V	626	613.8
4V	833	818.4
5V	1023 (Maxed)	1023

7.2.3. Simulated Testing

Simulated testing revolves around the premise of machine learning. The entire system being fully incorporated an algorithm will be run in an attempt to optimise desired outcome values. While this will be used as a teaching intermediate step for the system it will also double as simulated testing for the system. The Raspberry Pi 3 will work in parallel with the Atmega2560 to discover input variables corresponding to the desired pressure drop between transducers.

During the machine learning intermediate stage all functions of the system will need to be functional and implemented. This allows for each aspect of the system to be deemed functioning in the proper manor. The motors are measured to ensure the desired rotation is being executed by the microcontroller in each possible scenario of the system. Although mechanical components are normally very difficult to apply to simulated testing with this implementation everything from pump performance to the proper function of the valves in the pvc pipe. The simulated testing will allow the system to test every possible input from the user and assures the system will work as intended despite the parameters. Since the machine learning phase and the simulated testing will be done in parallel it is important when faults are detected in the system they are fixed before the system goes on with simulated testing. This is due because alterations to the system after the implementation of machine learning will render previous data useless. Overall the simulated testing of the system will be the final step in assuring performance of the system upon implementation overall between mechanical and electronic devices.

7.2.4. Integration Testing

The integration of both the standalone hardware components, software components, as well as the combination of both of them together is one of the

most important steps in the implementation of this project. Due to prior testing it has already been determined that the stepper motor drivers function as expected with the bipolar stepper motors that are being utilized in our design. However once the signal and power PCBs are completed, it will need to be seen if the power supplied to the driver is enough to keep it powered as well as if the atmega2560 GPIO pins will be able to consistently supply the required power to enable the direction as well as pulse pins of the driver to obtain accurate positioning of the valves, this will be determined through thorough testing in the lab when the signal and power boards are finished.

Through basic testing in the lab, it has been determined that the pressure transducers chosen to be used in this project provide a fairly accurate output voltage corresponding to the pressure at that point in the PVC piping. This voltage however during testing was read on a DMM, during the actual implementation of this project this voltage will have to be interpreted by the atmega2560 by feeding this output voltage of the sensor into an analog input pin of the atmega2560. While the output voltage read by the DMM during testing accurately represented the pressure it still has yet to be determined if once the signal board is completed and the atmega2560 chip integrated into it the signal interpreted by the chip will still be reliable. This will be tested by measuring the output voltage of the sensor with the DMM as well as with the signal board and comparing the values seen once the signal PCB is completed.

8. Final Design

This chapter covers the final system design. It gives instructions of how to use the system as well as wiring. It also provides an overview of the final project functions and instructions on how the user can address issues within the system. The final product met all the requirements including being aesthetically pleasing and easy to use. There were some issues experienced during the building and implementation phases of the project. These issues include: changes in sensors after machine learning which affected the learned data and water drainage due to pressure build up. The final product can be observed in Figure 8-1.

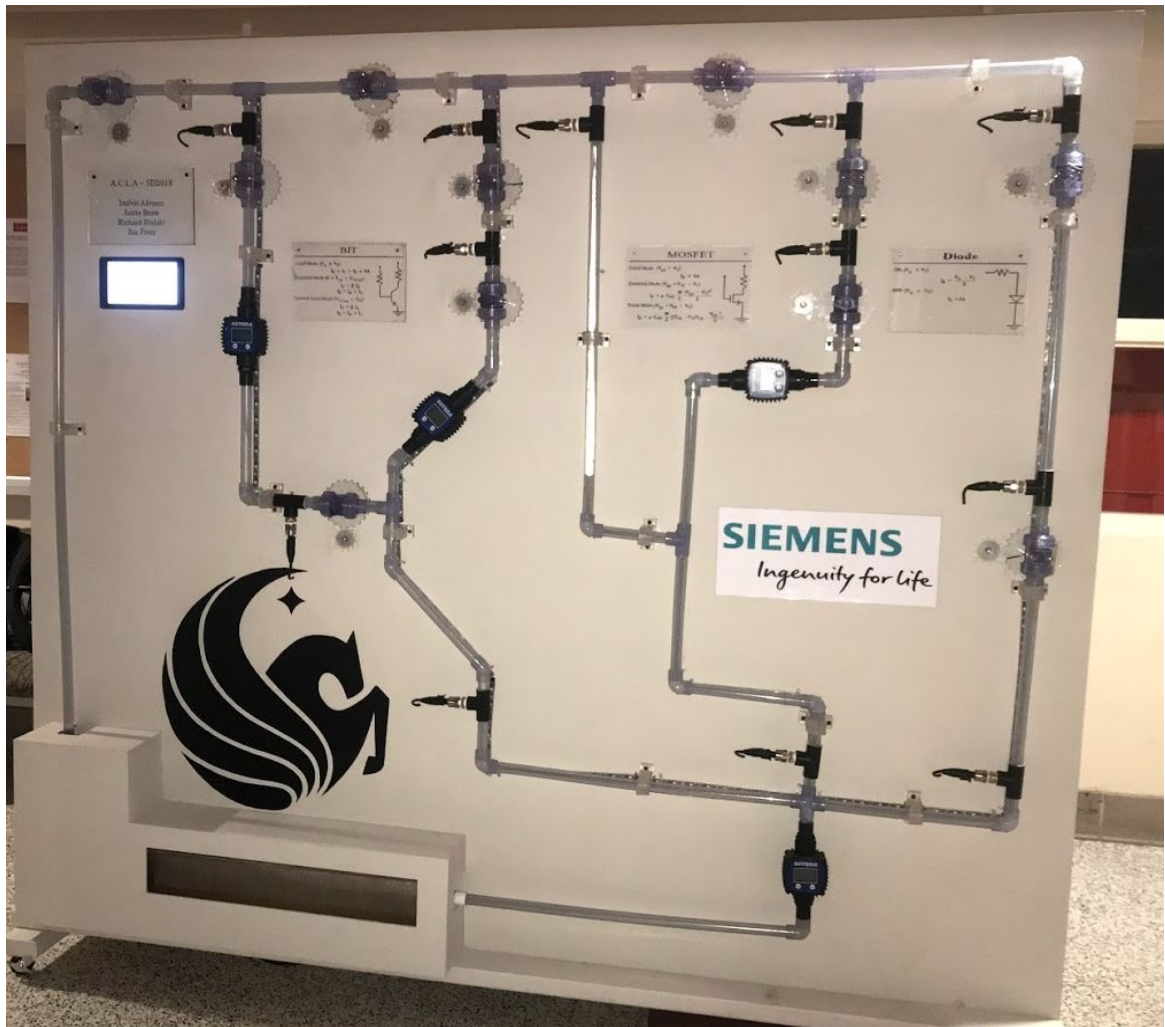


Figure 8-1: Final System

8.1. Final Project Functions

When the Raspberry Pi boots up, a startup script automatically loads the Python 3 GUI script which shows the user three options of components. From there the

user can select either the BJT, MOSFET, or diode to simulate. For each of the respective components, the application transitions to a menu of parameters for the selected electronic device. At this point, the user can select “Back” to return to the main menu. If the user is satisfied with the parameters selected, he/she pushes “Start” for the Pi to transmit the values via I2C protocol to the ATMEGA2560 microcontroller. The microcontroller realizes based on the data received which of the two devices are unused and send signals to the respective motor drivers to turn the valves to those branches, blocking any water from flowing through them. Then the microcontroller directs the valves of the desired branch to rotate so that it will emulate the results of the user selected component with the given parameters. After all the motors are aligned, the microcontroller signals the relay to activate the water pump, then reads the analog pressure transducers. Upon storing values, the microcontroller transmits them to the Pi which displays the pressures (converted to the corresponding voltage levels based on prior calculations) as voltages on the GUI. After the transmission, the microcontroller blinks the LEDs to help illustrate the current flowing. Whenever the user wants to stop the current representation, they may select “Stop” on the GUI. This causes the Pi to signal the microcontroller to stop the pump then reset all the motors to the neutral position (see Figure 8.1-1 below).

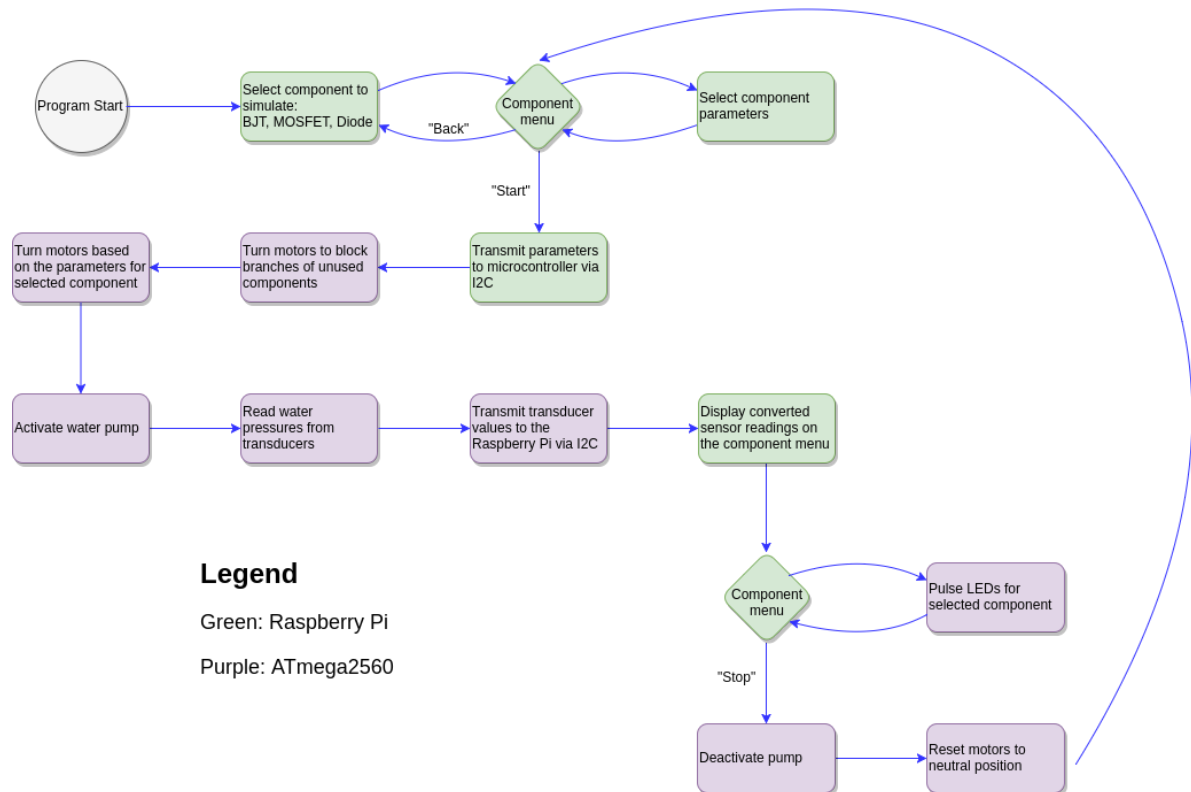


Figure 8.1-1: Software Flowchart

8.2. Usage Guide

Before plugging in the system, ensure that every valve is fully opened. If any of them are not, firmly rotate the gear until it is aligned. This alignment happens when the black line in the big gear (attached to the valve) is aligned in the same direction of the pipe connected to the valve. Verify that all the motor, sensor, pump and LED connections are correct based on Figure 8.2-1. The driver is wired to the motor following the colored lines which correspond to the color of the cables in the motor. The signal wires are connected to the signal PCB using the coding labeled in each one of the wires of the system. The same process is followed to ensure connection of the sensors as well as the LEDs.

The power PCB received power from the AC to DC converter. The LEDs and sensors are power through the 5V rail, while the motors are powered by 12V rail. The power PCB also supplies power to the signal PCB and Raspberry Pi 3 via USB ports. The AC to DC converter and the pump are both connected to an outlet. After supplying power, wait for the Raspberry Pi to boot up and load the GUI. Once the GUI is loaded, the system is ready for operation. Following the instructions in section 8.1, the user is able to operate the system without any problems.

If water is not draining to the user's liking, a valve can be slowly and loosely unscrewed a little when the pump is not activated. This allows air to enter the pipes and the pressure to be relieved. The user has to make sure the valve is screwed in tightly before the next run. If any leaks appear in the system, the pipes can be taken apart and PVC cement can be used to re seal the pipe or cover the leak. Furthermore, if either the flowmeters, pressure transducers, motors, drivers or LEDs are experiencing any difficulties, kindly refer to the user's manual to address the issue or the BOM to purchase a replacement.

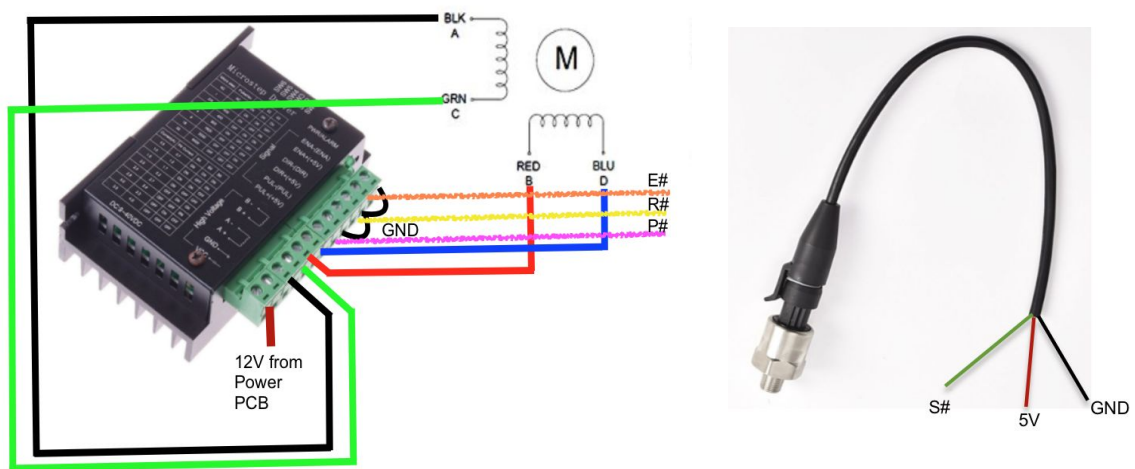


Figure 8.2-1: Motor and Sensor Wiring

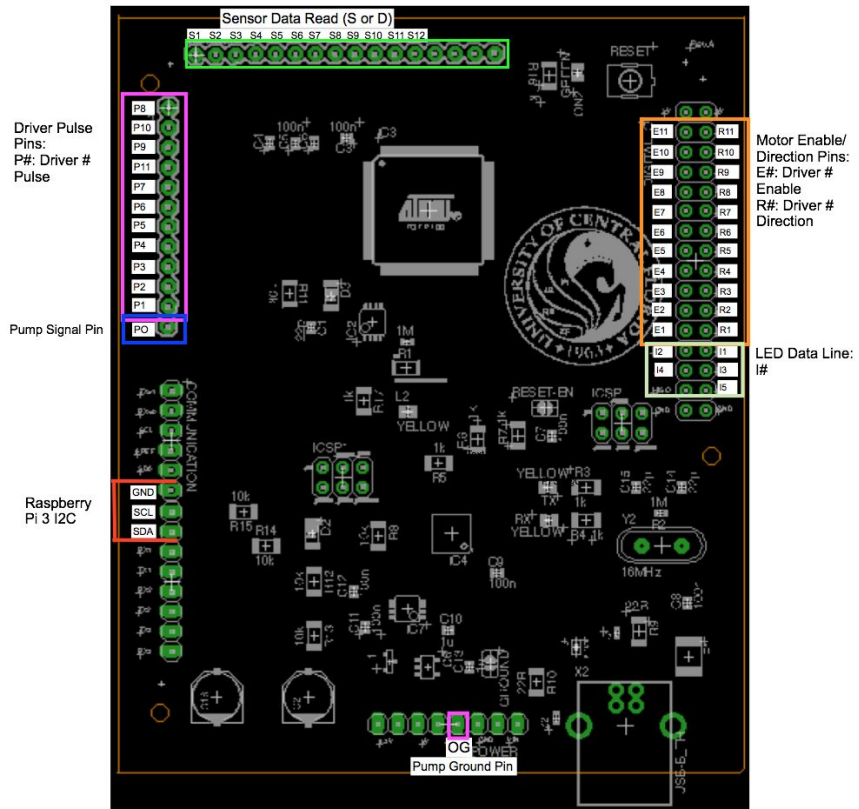


Figure 8.2-2: Signal PCB Wiring

9. Administrative Content

Administrative content involves the planning and management of the project. This includes milestone, bill of materials and critical path. Administrative content such as this, is important to keep the project on track and allow the visualization of tasks to be performed. It is also important to follow the bill of materials as this keeps the project on budget and prevents frivolous testing on components which will not be implemented into the system. Having administrative content acts as what a manager would bring to the table of a project to help keep the project under budget and on track.

9.1. Project Milestones

The following table displays the milestones that have been determined by the team based on the current status of the project and block diagram. All these milestones were made through the course of Senior Design 1 and 2.

Table 9.1-1: Milestones

	Description	Due Date
Spring 2018	Final Presentation	4/13/2018
	Final Product Complete	4/12/2018
	Final Checkoff	4/10/2018
	User Interface Implementation	4/04/2018
	Testing	4/01/2018
	Software/Hardware Integration	3/30/2018
	Machine Learning Implementation	3/25/2018
	Software Deliverable (Raspberry Pi Control/Application)	2/07/2018
	Hardware Deliverable	2/07/2018
	Power/ Signal PCB received	1/01/2018
Fall 2017	Order PCBs/Start Construction	12/10/2017
	Finish Designing Signal/Power PCBs	12/10/2017
	Final Paper Due	12/04/2017
	UI Completed	12/01/2017

	100 Page Submission	11/17/2017
	Finish Ordering Parts	11/10/2017
	60 Page Submission	11/03/2017
	UI Rough Draft	10/30/2017
	Finalize Motor	9/15/2017
	Autocad Design	9/08/2017
	Send out Proposal	9/08/2017
	Finish Proposal	9/05/2017

9.2. Critical Path

In the figure 8.2-1 a layout of the project as a whole with estimated time cost represented in days. The red line denotes the critical path. The critical path is a path through the development of the system which does not contain any slack time. Slack time is considered extra time which can be used for development while waiting on another process to complete. When a path contains slack time there is a grace period which provides more flexibility for the development of that process. When developing on the critical path it is extremely important to meet deadlines and stay on schedule. Any missed deadlines on the critical path will directly impact the schedule of the system extending it further into the future.

Looking at the figure 8.2-1 we see the critical path composed of Finalize Motor (10), Finish Parts (15), Build (45), Hardware Deliverable (14), Software Hardware Integration (14), Testing (7), Final Checkoff (21), Finalize Product (21), Final Presentation (1). This path is the critical path because there is no longer route which costs more to extend to from start. The critical path adding up to a cost of 148 days from start to finish. Any missed deadlines or changes to the critical path will push the cost of the project beyond 148 days.

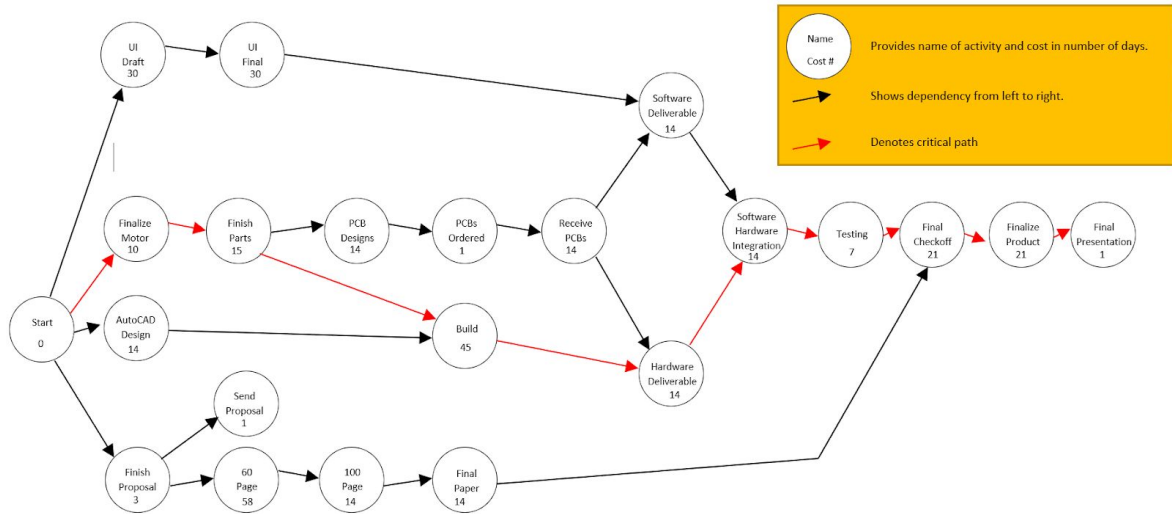


Figure 9.2-1: Critical Path Diagram

9.3. Budget and Finance

A proposal was created and presented to Siemens, who kindly agreed to sponsor the project. A bill of materials (BOM) was generated and updated based on changes in design, testing results and changes in project specifications and applications.

The following table shows an estimated quantity of the materials needed to build the proposed project. It also contains the price per unit of each material, providing an overall project estimated cost. This estimate was created based on the current design of the project containing only the MOSFET, BJT and diode representations.

Table 9.2-1: Overall System BOM

System	Description	Qty	Price/Unit	Subtotal
Miscellaneous	Raspberry Pi 7" Touchscreen Display	1	\$69.99	\$69.99
	8 oz. PVC Clear Cement	1	\$4.94	\$4.94
	Raspberry Pi 3 - Model B - ARMv7 w/ 1G RAM	1	\$35.00	\$35.00
	1/2" x 520" Thread Seal Tape	1	\$1.38	\$1.38
	Alitove 16.4' LED Strip	4	\$26.99	\$107.96
	ATmega 2560	1	\$11.98	\$11.98
	Power PCB	1	\$150.00	\$150.00
	Low Current Nema 23 CNC Stepper Motor 1.8A 340oz.in/2.4Nm CNC Mill Lathe Router	7	\$48.00	\$336.00
	SainSmart CNC Micro-Stepping Stepper Motor Driver 2M542 Bi-polar	7	\$38.00	\$266.00

	2phase 4.2A Switch			
	500 GPH Fountain Pump	1	\$42.98	\$42.98
	Pressure Sensor	2	\$12.15	\$24.30
	1/2" Clear PVC 90 degree Elbow	2	\$6.09	\$12.18
	1/2" x 5' Clear PVC Pipe, Sch. 40	10	\$7.99	\$79.90
	1/2" PVC Male Adapter 436-010L	12	\$0.30	\$3.60
	1/2 in. Schedule 40 PVC Tee Clear (not threaded)	4	\$6.66	\$26.64
	1/2 in. Schedule 40 PVC White Tee (threaded)	2	\$0.71	\$1.42
	1/2 in. Male x 1/4 in. Female NPT Coupler	2	\$3.98	\$7.96
	1/2" Clear PVC True Union Ball Valve	2	\$14.33	\$28.66
	Jumper Cables	2	\$3.95	\$7.90
	Bread board	1	\$19.95	\$19.95
	36" x 72" x 0.22" Acrylic Sheet	1	\$125.00	\$125.00
	Sande Plywood (Common: 1/2 in. x 4 ft. x 8 ft.; Actual: 0.472 in. x 48 in. x 96 in.)	2	\$32.00	\$64.00
	2 in. Steel Zinc-Plated Corner Brace (4-Pack)	5	\$2.85	\$14.25
	1/2 in. Electrical Metallic Tube (EMT) 2-Hole Straps (25-Pack)	4	\$4.33	\$17.32
	#2 x 1/4 in. Flat Head Phillips Zinc Wood Screw (8-Piece/Bag)	10	\$1.18	\$11.80
	Weld-On 4 Acrylic Adhesive - 4 Oz and Weld-On Applicator Bottle with Needle	2	\$8.55	\$17.10
	LEDMO Switching Converter, AC/DC Power Supply Adapter Transformer for LED Strip Lights, AC 100V/240V to DC 12V 10A 120W LED Strip Light Power Supply Switching Mode Converter	1	\$13.98	\$13.98
BJT	1/2" x 10' Clear PVC Pipe, Sch. 40	4	\$20.74	\$82.96
	1/2" Clear PVC 90 degree Elbow	2	\$6.09	\$12.18
	1/2 in. Schedule 40 PVC Tee Clear (not threaded)	1	\$6.66	\$6.66
	1/2 in. Schedule 40 PVC White Tee (threaded)	3	\$0.71	\$2.13

	1/2" Clear PVC 45 degree Elbow	4	\$10.24	\$40.96
	Fill-Rite Digital Display, In-Line Digital Meter	3	\$81.00	\$243.00
	1/2" Clear PVC True Union Ball Valve	4	\$14.02	\$56.08
	1 in. x 1/2 in. PVC Sch. 40 Reducer Bushing	6	\$1.38	\$8.28
	1/2 in. Male x 1/4 in. Female NPT Coupler	3	\$3.98	\$11.94
	Pressure Sensor	5		\$0.00
MOSFET	1/2" Clear PVC 90 degree Elbow	6	\$6.09	\$36.54
	1/2 in. Schedule 40 PVC Tee Clear (not threaded)	1	\$6.66	\$6.66
	1/2 in. Schedule 40 PVC White Tee (threaded)	2	\$0.71	\$1.42
	1/2" NPT Polypropylene Bulkhead Fitting w/ NBR Gasket	1	\$5.25	\$5.25
	1/2" Clear PVC True Union Ball Valve	2	\$14.02	\$28.04
	Pressure Sensor	2		\$0.00
	1 in. x 1/2 in. PVC Sch. 40 Reducer Bushing	4	\$1.38	\$5.52
	1/2 in. Male x 1/4 in. Female NPT Coupler	2	\$3.98	\$7.96
	Fill-Rite Digital Display, In-Line Digital Meter	2	\$81.00	\$162.00
Diode	Fill-Rite Digital Display, In-Line Digital Meter	1	\$81.00	\$81.00
	Pressure Sensor	3	\$17.98	\$53.94
	1/2" Clear PVC True Union Ball Valve	2	\$14.02	\$28.04
	1/2" Clear PVC 90 degree Elbow	1	\$6.09	\$6.09
	1/2 in. Schedule 40 PVC Tee Clear (not threaded)	1	\$6.66	\$6.66
	1/2 in. Schedule 40 PVC White Tee (threaded)	3	\$0.71	\$2.13
	1 in. x 1/2 in. PVC Sch. 40 Reducer Bushing	2	\$1.38	\$2.76
	1/2 in. Male x 1/4 in. Female NPT Coupler	3	\$3.98	\$11.94
	1/2" NPT Polypropylene Bulkhead	1	\$5.25	\$5.25

	Fitting w/ NBR Gasket			
Estimated Total Cost		\$2,417.58		

10. Conclusion

The overall purpose of this system is to provide a teaching aid for students pursuing educations in science and engineering fields. Professors will be able to visually represent a Bipolar Junction Transistor, a Metal–Oxide–Semiconductor Field-Effect Transistor, and a Diode. The components operating such that the water flow through the pipes acts as current flow through a wire. Voltages across components are represented by pressure drops between pressure transducers. This will allow for the professors to represent these components using a new method to those students who may benefit from visual aids. The project also poses an amazing learning experience for both the electrical engineers and computer engineers part of the senior design group.

The electrical engineers have the opportunity to gain experience with printed circuit boards. Both power PCB and signal PCB are incorporated into the system allowing for the electrical engineers to not only to obtain knowledge of power distribution throughout the electrical components, but also to obtain knowledge on signal processing to interface communication about the system. This will also provide the experience of designing and having fabricated printed circuit boards which in turn is a highly desirable skill to have due to its prevalence in the industry today. The electrical engineers will also be provided the opportunity to wire all of the components ensuring the proper voltage and current is supplied, this will allow experience to be gained since the size of the device is so large that even the lengths of the wires will need to be considered due to the voltage drops across them which is something that rarely needs considered in small scale devices/projects.

The computer engineers will gain immense knowledge in both software development and embedded systems. The experience gained by interfacing the Raspberry Pi 3 with the Atmega2560 will allow for the computer engineers to gain experience programming on a full blown linux operating system as well as an embedded programming on the Atmega2560. In addition to obtain programming experience the computer engineers also have the opportunity in interacting with the design of the signal PCB. The computer engineers will be responsible for the selection of the required output pins and corresponding components of the development board before the electrical engineers design the actual printed circuit board. The computer engineers gain knowledge in receiving feedback from components and issuing communicational commands to receive desired results. The computer engineers also have the opportunity to work in the up and coming field of machine learning. Machine learning offers a plethora of

knowledge of advanced algorithms. It also poses the case for computer engineers and electrical engineers to work together to accomplish a mutually desirable goal.

The cooperation of the computer engineers and electrical engineers is an absolute necessity for the success of the system. The computational components must work flawlessly with the electrical components for a machine learning algorithm to be implemented. If the components do not work together to perfection implementation of the algorithm becomes a futile endeavor as the system will learn from ill properly retrieved data.

Siemens sponsorship allows the acquisition of the necessary materials to make this system meet all the criterias and requirements. Updates on the project were provided monthly which allowed the project to stay on track and meet all the milestones. This provides the students with the experience in financial and engineering management.

The overall system allows for not only a useful teaching aid to be incorporated into the circuit classes of inspiring engineers, but also to provide the engineers working on the system with knowledge and understanding on a broad spectrum of the field of electrical and computer engineering. This system will be documented throughout so inspiring engineers of the future can not only learn from the final product in the classroom but also by the development process of the system providing value to not only introductory level students, but also higher level students who are interesting in the inner workings of the system.

11. References

- [1] "Department of Engineering Health and Safety: Soldering Safety Standards" [Online] Available: <https://safety.eng.cam.ac.uk/procedures/Soldering/soldering-safety>
- [2] "IPC Standards," IPC Standards: What Every Manufacturer Should Know. [Online]. Available: http://www.ipc.org/4.0_Knowledge/4.1_Standards/OEM-Stds/IPC-OEM-Stds-A4-English-11-11-ONLINE.pdf. [Accessed: 30-Nov-2017].
- [3] "Power Supply Safety Standards, Agencies and Marks," power-supply-safety-standards-agencies-and-marks.pdf. [Online]. Available: <http://www.cui.com/catalog/resource/power-supply-safety-standards-agencies-and-marks.pdf>. [Accessed: 30-Nov-2017].
- [4] "RoHS Guide," RoHS Compliance Guide: FAQ on RoHS Compliance. [Online]. Available: <http://www.rohsguide.com/rohs-faq.htm>. [Accessed: 30-Nov-2017].
- [5] "RoHS Guide," RoHS Compliance Guide: Regulations, 10 Substances, Exemptions, WEEE. [Online]. Available: <http://www.rohsguide.com/>. [Accessed: 30-Nov-2017].
- [6] "What is Wired Equivalent Privacy (WEP)? - Definition from WhatIs.com," SearchSecurity. [Online]. Available: <http://searchsecurity.techtarget.com/definition/Wired-Equivalent-Privacy>. [Accessed: 30-Nov-2017].
- [7] "What is Wi-Fi Protected Access (WPA)? - Definition from WhatIs.com," SearchMobileComputing. [Online]. Available: <http://searchmobilecomputing.techtarget.com/definition/Wi-Fi-Protected-Access>. [Accessed: 30-Nov-2017].
- [8] "What is 802.11n? - Definition from WhatIs.com," SearchMobileComputing. [Online]. Available: <http://searchmobilecomputing.techtarget.com/definition/80211n>. [Accessed: 30-Nov-2017].
- [9] "CHAPTER 6 : Layout," syque.com. [Online]. Available: <http://syque.com/cstyle/ch6.7.htm>. [Accessed: 30-Nov-2017].
- [10] T. Peters, "PEP 20 -- The Zen of Python," Python.org, 19-Aug-2004. [Online]. Available: <https://www.python.org/dev/peps/pep-0020/>. [Accessed: 26-Oct-2017].
- [11] G. van Rossum, "PEP 8 -- Style Guide for Python Code," Python.org, 05-Jul-2001. [Online]. Available: <https://www.python.org/dev/peps/pep-0008/>. [Accessed: 26-Oct-2017].
- [12] M. von Löwis, "PEP 3131 -- Supporting Non-ASCII Identifiers," Python.org, 01-May-2017. [Online]. Available: <https://www.python.org/dev/peps/pep-3131/>. [Accessed: 26-Oct-2017].
- [13] D. Goodger and G. van Rossum, "PEP 257 -- Docstring Conventions," Python.org, 29-May-2001. [Online]. Available: <https://www.python.org/dev/peps/pep-0257/>. [Accessed: 26-Oct-2017].
- [14] "C Coding Standard," C Coding Standard. [Online]. Available: <https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html#brace>. [Accessed: 27-Oct-2017].

- [15] P. Armstrong, "Shell Style Guide." [Online]. Available: <https://google.github.io/styleguide/shell.xml>. [Accessed: 27-Oct-2017].
- [16] "Shell Script Standards," Vokal Engineering. [Online]. Available: <https://engineering.vokal.io/Systems/sh.md.html>. [Accessed: 07-Nov-2017].
- [17] R. Hunter, "Is Standing Water Really Dangerous?," Mama's Health.com. [Online]. Available: <http://mamashealth.com/environmentalhealth/standingwater.asp>. [Accessed: 12-Nov-2017].
- [18] "ANSI/IEC 60529-2004 ,," Microsoft Word - NEMA Pub of ANSI adopt of IEC stds disclaimer.doc. [Online]. Available: <https://www.nema.org/Standards/ComplimentaryDocuments/ANSI-IEC-60529.pdf>. [Accessed: 30-Nov-2017].
- [19] "Raspberry Pi 3 Model B," Raspberry Pi. [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. [Accessed: 09-Oct-2017].
- [20] "8-bit AVR Microcontrollers," ATmega328/P. [Online]. Available: http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf. [Accessed: 30-Nov-2017].
- [21] "ATmega328P," Microchip Technology Inc. [Online]. Available: <http://www.microchip.com/wwwproducts/en/ATmega328P>. [Accessed: 30-Nov-2017].
- [22] "Atmel SAM D21E / SAM D21G / SAM D21J," Atmel | SMART SAM D21 Datasheet. [Online]. Available: https://cdn.sparkfun.com/datasheets/Dev/Arduino/Boards/Atmel-42181-SAM-D21_Datasheet.pdf. [Accessed: 30-Nov-2017].
- [23] "MSP430 ultra-low-power MCUs – Products," 16-bit 32-bit MCU | Low-power MCUs | Products | Microcontrollers (MCU) | TI.com. [Online]. Available: <http://www.ti.com/microcontrollers/msp430-ultra-low-power-mcus/products.html>. [Accessed: 22-Oct-2017].
- [24] "Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V," Atmel ATmega640/1280/1281/2560/2561 Datasheet. [Online]. Available: http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf. [Accessed: 30-Nov-2017].
- [25] "Arduino MEGA 2560," Arduino MEGA 2560. [Online]. Available: <http://www.mantech.co.za/datasheets/products/A000047.pdf>. [Accessed: 30-Nov-2017].
- [26] atmega2560-16au-lqfp100.jpg (800×800). [Online]. Available: https://sfeproduct.com/727-thickbox_default/atmega2560-16au-lqfp100.jpg. [Accessed: 30-Nov-2017].
- [27] "QUANTIL Inc," Data Transmission - Parallel vs Serial Transmission. [Online]. Available: <https://www.quantil.com/content-delivery-insights/content-acceleration/data-transmission/>. [Accessed: 30-Nov-2017].
- [28] "Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V" *Atmel*, Atmel Corporation, [Online]. Available: http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf. [Accessed: 09-Oct-2017].

- [29] R. Keim, "UART Baud Rate: How Accurate Does It Need to Be?," All About Circuits, 25-Jan-2017. [Online]. Available: <https://www.allaboutcircuits.com/technical-articles/the-uart-baud-rate-clock-how-accurate-does-it-need-to-be/>. [Accessed: 09-Oct-2017].
- [30] stevenvh, "Re: USART, UART, RS232, USB, SPI, I2C, TTL, etc. what are all of these and how do they relate to each other?," Stack Exchange, 13-Aug-2012. [Online]. Available: <https://electronics.stackexchange.com/questions/37814/usart-uart-rs232-usb-spi-i2c-ttl-etc-what-a-re-all-of-these-and-how-do-th>. [Accessed: 09-Oct-2017].
- [31] Y. Gao, "What are the pros and cons of I2C versus SPI interface?," Quora, 16-Feb-2014. [Online]. Available: <https://www.quora.com/What-are-the-pros-and-cons-of-I2C-versus-SPI-interface>. [Accessed: 09-Oct-2017].
- [32] P. Martin, "Using your new Raspberry Pi 3 as a WiFi access point with hostapd," Frillip's Blog, 13-Jun-2016. [Online]. Available: <https://frillip.com/using-your-raspberry-pi-3-as-a-wifi-access-point-with-hostapd/>. [Accessed: 30-Nov-2017].
- [33] "Browser Statistics," w3schools.com. [Online]. Available: <https://www.w3schools.com/browsers/default.asp>. [Accessed: 01-Nov-2017].
- [34] "Understanding the Remote Desktop Protocol (RDP)," Microsoft. [Online]. Available: <https://support.microsoft.com/en-us/help/186607/understanding-the-remote-desktop-protocol-rdp>. [Accessed: 02-Dec-2017].
- [35] "Why use VNC? Comparison of RDP and VNC protocols.," AbtoVNC Software, 09-Apr-2014. [Online]. Available: <http://remote-screen.com/comparison-of-rdp-and-vnc-protocols>. [Accessed: 02-Nov-2017].
- [36] "Pi3B wifi brcmf_sdio_hdparse · Issue #1313 · raspberrypi/linux," GitHub. [Online]. Available: <https://github.com/raspberrypi/linux/issues/1313>. [Accessed: 05-Nov-2017]. "Pi3B wifi brcmf_sdio_hdparse · Issue #1313 · raspberrypi/linux," GitHub. [Online]. Available: <https://github.com/raspberrypi/linux/issues/1313>. [Accessed: 05-Nov-2017]. "Pi3B wifi brcmf_sdio_hdparse · Issue #1313 · raspberrypi/linux," GitHub. [Online]. Available: <https://github.com/raspberrypi/linux/issues/1313>. [Accessed: 03-Nov-2017].
- [37] "Edimax EW-7811Un 150Mbps 11n Wi-Fi USB Adapter, Nano Size Lets You Plug it and Forget it, Ideal for Raspberry Pi / Pi2, Supports Windows, Mac OS, Linux (Black/Gold)," Amazon. [Online]. Available: https://www.amazon.com/Edimax-EW-7811Un-150Mbps-Raspberry-Supports/dp/B003MTTJOY/ref=sr_1_5?ie=UTF8&qid=1510291146&sr=8-5&keywords=wifi+dongle&dpID=31ChKj3d17L&preST=_SX300_QL70_&dpSrc=srch. [Accessed: 03-Nov-2017].
- [38] " pi 3 wifi range," Raspberry Pi Forums. [Online]. Available: <https://www.raspberrypi.org/forums/viewtopic.php?t=139021>. [Accessed: 03-Nov-2017].
- [39] *PCB Basics*. [Online]. Available: <https://learn.sparkfun.com/tutorials/pcb-basics>. [Accessed: 30-Nov-2017].
- [40] "S. Vorkoetter, "How Electric Motors Work," <http://www.stefanv.com>. [Online]. Available: <https://web.archive.org/web/20131002195720/http://www.stefanv.com/rcstuff/qf200212.html>. [Accessed: 30-Nov-2017].

- [41] *DC Motors | Advantages and Hazards of Operating DC Motors*. [Online]. Available: <https://www.galco.com/comp/prod/moto-dc.htm>. [Accessed: 30-Nov-2017].
- [42] *Servo Motor Basics, Working Principle & Theory*. [Online]. Available: <https://circuitdigest.com/article/servo-motor-basics>. [Accessed: 30-Nov-2017].
- [43] "Advantages & Disadvantages of Servo Motor," *PLC, PLC LADDER, PLC EBOOK, PLC PROGRAMMING*. [Online]. Available: <http://plc-scada-dcs.blogspot.com/2011/12/servo-motor-advantages-disadvantages.html#axzz4vRzgj3Qo>. [Accessed: 30-Nov-2017].
- [44] D. Collins, "Stepper Motor Basics," *Linear Motion Tips*, 19-Jan-2017. [Online]. Available: <http://www.linearmotiontips.com/stepper-motor-basics/>. [Accessed: 30-Nov-2017].
- [45] George, Sam, and C. Specialists, "Unipolar Stepper Motor vs Bipolar Stepper Motors," *Simply Smarter Circuitry Blog*, 30-Mar-2016. [Online]. Available: <https://www.circuitspecialists.com/blog/unipolar-stepper-motor-vs-bipolar-stepper-motors/>. [Accessed: 30-Nov-2017].
- [46] "CNC Stepper Motor Nema 23 Bipolar 2.8A 269oz.in/1.9Nm CNC Mill Lathe Router" Amazon. Available: https://www.amazon.com/dp/B00PNEPI0A/ref=asc_df_B00PNEPI0A5226821/?tag=hyprod-20&creative=395033&creativeASIN=B00PNEPI0A&linkCode=df0&hvadid=198075247191&hvpos=1o3&hvnetw=g&hvrnd=3768024401152337696&hvpone=&hvtwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlcint=&hvlcphy=9011793&hvtargid=pla-414268560157. [Accessed: 30-Nov-2017].
- [47] "Nema 23 CNC Stepper Motor 2.8A 178.5oz.in/1.26Nm CNC Stepping Motor DIY CNC Mill" Amazon. Available: https://www.amazon.com/dp/B00PNEPF5I/ref=asc_df_B00PNEPF5I5226870/?tag=hyprod-20&creative=395033&creativeASIN=B00PNEPF5I&linkCode=df0&hvadid=198064502357&hvpos=1o1&hvnetw=g&hvrnd=6338692856528211934&hvpone=&hvtwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlcint=&hvlcphy=9011793&hvtargid=pla-323464553412. [Accessed: 30-Nov-2017].
- [48] "12V, 1.7A, 416 oz-in Geared Bipolar Stepper Motor" RobotShop. Available: http://www.robotshop.com/en/12v-17a-416oz-geared-bipolar-stepper-motor.html?gclid=CjwKCAjw4KvPBRBeEiwAlqCB-RjP4MHZK84W5HPJe-rG-_iGXi-CaM3VAxhhBbhqbtvTgEfnliJVVSxoCYk4QAvD_BwE#reviewBox. [Accessed: 30-Nov-2017].
- [49] "12 Volt 1.8 Step Angle Bipolar Stepper Motor" Jameco Electronics. Available: <https://www.jameco.com/Jameco/Products/ProdDS/2158531.pdf>. [Accessed: 30-Nov-2017].
- [50] "24-36 Volt DC and 0.9-3.0 Amp Stepper Motor Driver" Circuit Specialists. Available: https://www.circuitspecialists.com/cw230.html?otaid=gpl&gclid=CjwKCAjw7MDPBRAFEiwAppdF9HwUzWhD_PwQa35-LhhFSjQpvlquHFmgCoyMNq6KMRBobD5sURVZDhoC3rIQAvD_BwE. [Accessed: 30-Nov-2017].
- [51] "10-30 V 4A Single Bipolar Stepper Motor Driver" robotshop. Available: http://www.robotshop.com/en/10-30v-4a-single-bipolar-stepper-motor-driver.html?gclid=CjwKCAjw7MDPBRAFEiwAppdF9Kcp8IGZEhJxUGF9GgQCT0T-s4cjjb9J26CjLY5ceWbM01D7zEsyx0CjYkQAvD_BwE. [Accessed: 30-Nov-2017].
- [52] "1 x single axis TB6600 Stepper Motor Driver board." nastelroy. Available: <https://nastelroy.files.wordpress.com/2016/02/tb6600-1-axis.pdf>. [Accessed: 30-Nov-2017].
- [53] "Pololu 8-35V 2A Single Bipolar Stepper Motor Driver A4988." robotshop.

Available: <http://www.robotshop.com/media/files/pdf/datasheet-1182.pdf>. [Accessed: 30-Nov-2017].

[54] "DRV8825 Stepper Motor Controller." Texas Instruments, Jul-2014. available:

<http://www.robotshop.com/media/files/pdf/dual-bipolar-stepper-motor-controller-arduino-datasheet.pdf>. [Accessed: 30-Nov-2017].

[55] "RGB LED Strips," Current Draw | RGB LED Strips | Adafruit Learning System, 04-May-2015. [Online]. Available: <https://learn.adafruit.com/rgb-led-strips/current-draw>. [Accessed: 30-Nov-2017].

[56] N. Cravotta, "Driving LEDs - Choosing Between Analog and Digital Topologies," Driving LEDs Choosing Between Analog Topologies | DigiKey, 27-May-2011. [Online]. Available: <https://www.digikey.pt/en/articles/techzone/2011/may/driving-leds--choosing-between-analog-and-digital-topologies>. [Accessed: 30-Nov-2017].

[57] Alitove WS2818 LED Strip Datasheet (Appendix A)

[58] J. Wisdom, "3528 vs 5050 vs 5630 LED SMD Diodes," Heraco Lights. [Online]. Available: <https://heracolights.com/2014/03/10/3528-vs-5050-vs-5630-led-smd-diodes/>. [Accessed: 30-Nov-2017].

[59] "LPD-8803/8806 Datasheet," LPD8806 datasheet Greeled-3. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/lpd8806_english.pdf. [Accessed: 30-Nov-2017].

[60] "1/2" Clear PVC True Union Ball Valve," *PVC Fittings Online*. [Online]. Available: <https://www.pvcfittingsonline.com/1-2-clear-pvc-true-union-ball-valve.html>. [Accessed: 30-Nov-2017].

[61] Çengel, Y. and Cimbala, J. (n.d.). *Fluid Mechanics Fundamentals and Applications*. New York: McGraw Hill, 2006, p. 321-337.

[62] "Total Pond 500 GPH Fountain Pump-MD11500," The Home Depot, 07-Nov-2017. [Online]. Available: <https://www.homedepot.com/p/Total-Pond-500-GPH-Fountain-Pump-MD11500/202017052>. [Accessed: 30-Nov-2017].

[63] "Model P0550," p0550_manual.pdf. [Online]. Available: http://www.p3international.com/manuals/p0550_manual.pdf. [Accessed: 30-Nov-2017].

[64] Sotera Systems. *FR1118-P10 User's Manual*. Tuthill. (Appendix B)

[65] "What is a Pressure Gauge?," 1 - 2 pressure.pdf. [Online]. Available: <http://www.cpinc.com/Terice/Pressure/1%20-%202%20pressure.pdf>. [Accessed: 30-Nov-2017].

[66] "Advantages and Disadvantages of Digital Pressure Gauges," WIKA Instrument. [Online]. Available: http://www.wika.us/solutions_digital_pressure_gauges_advantages_en_us.WIKA. [Accessed: 30-Nov-2017].

[67] "DG25 Digital Pressure Gauge," DG25 Digital Pressure Gauge. [Online]. Available: http://www.ashcroft.com/products/pressure_gauges/digital_gauges/dg25-digital-pressure-gauge.cfm. [Accessed: 30-Nov-2017].

[68] "Noshok 40-911- 0 psi to 800 psi - kPa npt size 1/4 4" Liquid Pressure Gauge," Noshok 40-911- 0 psi to 800 psi - kPa npt size 1/4 4" Liquid Pressure Gauge. [Online]. Available: <http://noshok.massmeasure.com/40-911--0-psi-to-800-psi---kPa>. [Accessed: 30-Nov-2017].

[69] "Winters PEM Series Steel Dual Scale Economy Pressure Gauge, 0-15 psi/kpa, 2" Dial Display, /-3-2-3% Accuracy, 1/4" NPT Bottom Mount," Industrial Pressure Gauges: Amazon.com: Industrial & Scientific. [Online]. Available: https://www.amazon.com/Winters-Economy-Pressure-Display-Accuracy/dp/B0087UBDAQ/ref=sr_1_2?ie=UTF8&qid=1501165138&sr=8-2&keywords=pressure%2Bgauge%2B15%2Bpsi%2Bwinters. [Accessed: 30-Nov-2017].

[70] "PRESSURE TRANSDUCERS," ipt_tech-note_1.pdf. [Online]. Available: http://www.sensata.com/download/ipt_tech-note_1.pdf. [Accessed: 30-Nov-2017].

[71] "Multiphysics Cyclopedia," COMSOL. [Online]. Available: <https://www.comsol.com/multiphysics/piezoresistive-effect>. [Accessed: 30-Nov-2017].

[72] "PRESSURE TRANSDUCERS," ipt_tech-note_2.pdf. [Online]. Available: http://www.sensata.com/download/ipt_tech-note_2.pdf. [Accessed: 30-Nov-2017].

[73] "19 mm Series," 19 mm Series Low-cost, Stainless Steel, Isolated Pressure Sensors. [Online]. Available: <https://sensing.honeywell.com/honeywell-sensing-19mm-series-isolated-pressure-sensors-product-sheet-009132-3-en.pdf>. [Accessed: 30-Nov-2017].

[74] "Heavy Duty Pressure Transducers," MLH Series, 6 bar to 550 bar | 50 psi to 8000 psi Heavy Duty Pressure Transducers. [Online]. Available: <https://sensing.honeywell.com/honeywell-sensing-heavy-duty-pressure-transducers-mlh-series-datasheet-008118-8-en.pdf>. [Accessed: 30-Nov-2017].

[75] "Eyourlife Universal 30PSI Pressure Transducer Sender Solenoid for Oil Fuel Gas Air Water G7," Amazon.com: Eyourlife Universal 30PSI Pressure Transducer Sender Solenoid for Oil Fuel Gas Air Water G7: Automotive. [Online]. Available: <https://www.amazon.com/Eyourlife-Universal-Pressure-Transducer-Solenoid/dp/B00RCPDE40>. [Accessed: 30-Nov-2017].

[76] "Capacitive vs. Resistive Touchscreens," Capacitive vs. Resistive Touchscreens. [Online]. Available: <http://www.capacitive-resistive-touchscreens.articles.r-tt.com/>. [Accessed: 28-Oct-2017].

[77] R. Ram, "Re: How can cell phone touch screens continue to function even after the glass gets ridiculously cracked?," Quora, 05-Mar-2015. [Online]. Available: <https://www.quora.com/How-can-cell-phone-touch-screens-continue-to-function-even-after-the-glass-gets-ridiculously-cracked>. [Accessed: 28-Oct-2017].

[78] A. Ravi, "Re: How beneficial is a tempered glass screen protector over a normal screen guard for smartphones?," Quora, 08-Dec-2015. [Online]. Available: <https://www.quora.com/How-beneficial-is-a-tempered-glass-screen-protector-over-a-normal-screen-guard-for-smartphones>. [Accessed: 28-Oct-2017].

[79] "Raspberry Pi Touch Display," Raspberry Pi. [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-touch-display/>. [Accessed: 28-Oct-2017].

- [80] "Waveshare Raspberry Pi 10.1inch HDMI LCD Capacitive Touch Screen with Case for Raspberry Pi 2 / 3 - Black," Amazon. [Online]. Available: https://www.amazon.com/Waveshare-10-1inch-HDMI-LCD-Capacitive/dp/B01CU7VX5Q/ref=sr_1_10?s=electronics&ie=UTF8&qid=1508912572&sr=1-10&keywords=raspberry+pi+3+capacitive+touch+screen. [Accessed: 29-Oct-2017].
- [81] "LANDZO 11.6 Inch 1920X1080 HDMI PS3 PS4 WiiU Xbox360 1080P LED Display Monitor for Raspberry Pi 3, 2 1 Model B B Plus Windows 7 8 10 System (LANDZO 11.6 Inch Display)," Amazon. [Online]. Available: https://www.amazon.com/LANDZO-Touch-Screen-Raspberry-Display/dp/B01ID5BQTC/ref=sr_1_1?ie=UTF8&qid=1508910756&sr=8-1&keywords=raspberry+pi+capacitive+touch+screen&refinements=p_72%3A2661618011. [Accessed: 29-Oct-2017].
- [82] A. W. Warner, "7 Reasons How Slack Strengthens Our Business," FooPlugins, 24-Nov-2015. [Online]. Available: <https://fooplugins.com/slack-team-communication-tool/>. [Accessed: 25-Oct-2017].
- [83] M. Mansfield, "What is Slack and How Do I Use It for My Team?," Small Business Trends, 07-Jun-2016. [Online]. Available: <https://smallbiztrends.com/2015/12/slack-use-team.html>. [Accessed: 24-Oct-2017].
- [84] J. Duffy, "Asana," PCMAG, 13-Oct-2017. [Online]. Available: <https://www.pcmag.com/article2/0,2817,2408011,00.asp>. [Accessed: 24-Oct-2017].
- [85] K. Ayyar, "Re: Why do a few people prefer command-line text editors over GUI editors, even in cases where a window manager is available?," Quora, 15-Aug-2011. [Online]. Available: <https://www.quora.com/Why-do-a-few-people-prefer-command-line-text-editors-over-GUI-editors-even-in-cases-where-a-window-manager-is-available>. [Accessed: 23-Oct-2017].
- [86] Tikerer, "Why Arduino Is Not The Right Educational Tool," Hack Van de Dam, 08-May-2013. [Online]. Available: <http://www.hackvandedam.nl/blog/?p=762>. [Accessed: 23-Oct-2017].
- [87] "Home," SolidWorks. [Online]. Available: <http://www.solidworks.com/>. [Accessed: 30-Nov-2017].
- [88] "PCB Layout Software Features | EAGLE | Autodesk," Autodesk 2D and 3D Design and Engineering Software. [Online]. Available: <https://www.autodesk.com/products/eagle/features>. [Accessed: 30-Nov-2017].
- [89] "hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator," hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator. [Online]. Available: <https://w1.fi/hostapd/>. [Accessed: 30-Nov-2017].
- [90] "RealVNC," Docs | VNC Connect and Raspberry Pi | RealVNC. [Online]. Available: <https://www.realvnc.com/en/connect/docs/raspberry-pi.html>. [Accessed: 30-Nov-2017].
- [91] "VNC (Virtual Network Computing)," VNC (Virtual Network Computing) - Raspberry Pi Documentation. [Online]. Available: <https://www.raspberrypi.org/documentation/remote-access/vnc/>. [Accessed: 12-Nov-2017].
- [92] "RealVNC," Download VNC Viewer | RealVNC. [Online]. Available: <https://www.realvnc.com/en/connect/download/viewer/>. [Accessed: 11-Nov-2017].

- [93] M. Muchmore and J. Duffy, "Google Drive," PCMAG, 19-Jul-2017. [Online]. Available: <https://www.pcmag.com/article2/0,2817,2403546,00.asp>. [Accessed: 24-Oct-2017].
- [94] "Google Drive - Cloud Storage & File Backup for Photos, Docs & More," Google. [Online]. Available: <https://www.google.com/drive/>. [Accessed: 24-Oct-2017].
- [95] S. Shubham, "Re: Why is the Arduino IDE considered so bad?," Quora, 30-Jun-2015. [Online]. Available: <https://www.quora.com/Why-is-the-Arduino-IDE-considered-so-bad>. [Accessed: 24-Oct-2017].
- [96] "Draw.io Review: Taking Simplicity to New Projects," Blinklist, 23-May-2013. [Online]. Available: <http://blinklist.com/reviews/draw-io>. [Accessed: 25-Sep-2017].
- [97] SFUptownMaker, "I2C," SparkFun. [Online]. Available: <https://learn.sparkfun.com/tutorials/i2c>. [Accessed: 28-Nov-2017].
- [98] "Graphic User Interface FAQ," Graphic User Interface FAQ — Python 3.6.3 documentation, 01-Dec-2017. [Online]. Available: <https://docs.python.org/3/faq/gui.html>. [Accessed: 12-Oct-2017].
- [99] "25. Graphical User Interfaces with Tk," 25. Graphical User Interfaces with Tk — Python 3.6.3 documentation, 01-Dec-2017. [Online]. Available: <https://docs.python.org/3/library/tk.html>. [Accessed: 15-Oct-2017].
- [100] "About Tcl/Tk," Tcl Developer Xchange. [Online]. Available: <https://www.tcl.tk/about/>. [Accessed: 20-Oct-2017].
- [101] Sam DeFabbia-Kane, "Re: Developing GUIs in Python: Tkinter vs PyQt [closed]," Stack Exchange, 07-Jul-2009. [Online]. Available: <https://stackoverflow.com/questions/1094155/developing-guis-in-python-tkinter-vs-pyqt>. [Accessed: 21-Oct-2017].
- [102] B. Oakley, "Re: Choosing wxPython over Tkinter." wxPyWiki, 09-Nov-2010. [Online]. Available: <https://wiki.wxpython.org/Choosing%20wxPython%20over%20Tkinter>. [Accessed: 21-Oct-2017].
- [103] DrAl. "Re: Should I use PyQt or PySide for a new Qt project?" Ask Ubuntu, 24-May-2012. [Online]. Available: <https://askubuntu.com/questions/140740/should-i-use-pyqt-or-pyside-for-a-new-qt-project>. [Accessed: 21-Oct-2017].
- [104] "PyQt," Python Wiki, 06-Aug-2015. [Online]. Available: <https://wiki.python.org/moin/PyQt>. [Accessed: 22-Oct-2017].
- [105] "PySide," Python Wiki, 31-May-2014. [Online]. Available: <https://wiki.python.org/moin/PySide>. [Accessed: 22-Oct-2017].
- [106] "Qt Designer Manual," Qt. [Online]. Available: <http://doc.qt.io/qt-4.8/designer-manual.html>. [Accessed: 02-Oct-2017].
- [107] "PyQt5 Reference Guide," PyQt5 Reference Guide. [Online]. Available: <http://pyqt.sourceforge.net/Docs/PyQt5/>. [Accessed: 22-Oct-2017].
- [108] V. Maini, "Machine Learning for Humans, Part 2.1: Supervised Learning," Medium, 19-Aug-2017. [Online]. Available:

<https://medium.com/machine-learning-for-humans/supervised-learning-740383a2feab>.
[Accessed: 30-Nov-2017].

[109] "analogRead()," Arduino Reference. [Online]. Available:
<https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/>. [Accessed:
30-Nov-2017].

[110] "Blink," Arduino - Blink. [Online]. Available: <https://www.arduino.cc/en/Tutorial/Blink>.
[Accessed: 30-Nov-2017].

[111] mister_rf, "CD-ROM Drive Stepper motor," *edaboard*, 30-Jun-2011. .
Available: <http://www.edaboard.com/thread217270.html>. [Accessed: 30-Nov-2017].

[112] "Raspberry Pi 3 is out now! Specs, benchmarks & more," The MagPi Magazine,
02-Jun-2017. [Online]. Available:
<https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/>. [Accessed: 30-Nov-2017].

[113] "Pololu - Addressable RGB 60-LED Strip, 5V, 1m (WS2812B)," Pololu Robotics &
Electronics. [Online]. Available: <https://www.pololu.com/product/2549>. [Accessed: 30-Nov-2017].

[114] "AnalogInput," Arduino - AnalogInput. [Online]. Available:
<https://www.arduino.cc/en/Tutorial/AnalogInput>. [Accessed: 30-Nov-2017].

[115] "Google Java Style Guide," Google Java Style Guide. [Online]. Available:
<https://google.github.io/styleguide/javaguide.html>. [Accessed: 30-Nov-2017].

[116] "Series DPGA & DPGW Digital Pressure Gages," A_34C_rev2.pdf. [Online]. Available:
http://www.dwyer-inst.com/PDF_files/A_34C_rev2.pdf . [Accessed: 30-Nov-2017].

[117] "PySide - Official Site," Qt Wiki [Online]. Available: <https://wiki.qt.io/PySide>. [Accessed:
22-Oct-2017]

[118] K. Brown, "What Is GitHub, and What Is It Used For?," How-To Geek, 06-Sep-2017.
[Online]. Available:
<https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/>.
[Accessed: 24-Oct-2017].

[119] Broadcom, "BCM43438 WLAN + ST Micro STM32F412 MCU
for 802.11b/g/n," SP-ZZ1MD-C datasheet.

[120] "MSP430x2xx Family User's Guide," Texas Instruments. [Online]. Available:
<http://www.ti.com/lit/ug/slau144j/slau144j.pdf>. [Accessed: 10-Oct-2017].

[121] N. Vatamaniuc, "Choosing wxPython over Tkinter," wxPyWiki 09-Nov-2010. [Online].
Available: <https://wiki.wxpython.org/Choosing%20wxPython%20over%20Tkinter>. [Accessed:
10-Oct-2017].

[122] lunaryorn. "Re: PyQt or PySide - which one to use [closed]" Stack Overflow, 31-Jul-2011.
[Online]. Available:
<https://stackoverflow.com/questions/6888750/pyqt-or-pyside-which-one-to-use>. [Accessed:
22-Oct-2017]

[123] "kuman 5 inch Resistive Touch Screen 800x480 HDMI TFT LCD Display Module with Touch
Panel USB Port and Touch Pen for Raspberry Pi 3 2 Model B RPi 1 B B+ A A+ SC5A," Amazon.

[Online]. Available:

https://www.amazon.com/kuman-Resistive-800x480-Display-Raspberry/dp/B01F3EKJIA/ref=sr_1_3?s=electronics&ie=UTF8&qid=1508912650&sr=1-3&keywords=raspberry+pi+3+resistive+touch+screen. [Accessed: 29-Oct-2017].

[124] D. Kukarsky, "The Pros & Cons Of Using Chrome Remote Desktop," FixMe.IT Official Blog, 26-Oct-2017. [Online]. Available:

<https://blog.techinline.com/2017/06/08/the-pros-cons-of-using-chrome-remote-desktop/>. [Accessed: 01-Nov-2017].

[125] A. Henry, "Five Best Remote Desktop Tools," Lifehacker, 26-Jan-2014. [Online]. Available: <https://lifehacker.com/five-best-remote-desktop-tools-1508597379>. [Accessed: 30-Nov-2017].

[126] "Patriot Koi Pond Pump KP1200," www.halfoffponds.com. [Online]. Available: <https://www.halfoffponds.com/Patriot-Pond-Pond-Pump-p/3918.htm>. [Accessed: 30-Nov-2017].

[127] P. Burgess, "Digital RGB LED Strip," Troubleshooting | Digital RGB LED Strip | Adafruit Learning System. [Online]. Available:

<https://learn.adafruit.com/digital-led-strip/troubleshooting?view=all#overview>. [Accessed: 30-Nov-2017].

Product Specification

Product Name	<u>WS2812B LED STRIP</u>
Product Number	<u>LC-2812BXX-5V</u>
Version Number	<u>V1.0</u>

CE

RoHS

Product description

WS2812B LED Strip use the FPCB (flexible printed circuit board) to be assembly circuit boards, use SMD LED lights and components to be assembled, so that the thickness of the product only just like, Occupy a small space, and can be arbitrarily cut, FPC is soft material, can random winding, and wound freely movable in three-dimensional space without breaking, suitable for irregular place and narrow space of places to install, but also as any possible bending, suitable for advertising, decorative lighting, any combination of the kinds of patterns, LED Chip is built-in IC, then the components is less , pixels can be high, can make attractive appearance, and so on.

IC is designed with one data line, two power lines, 256 gray scale, high brightness, fast data transfer, fast screen refresh, etc. , with the controller, you can achieve monochrome, gradient, water , horse racing, text, numbers, chase, English, pictures, animations, video, any effecton, and so on.

Products using DC5V voltage to power supply, high safety factor, using the import led chip to be the light source , high brightness, light fade is small, maintenance-free, long lifespan , energy-saving and environmental protection, based on the above characteristics, More and more peopel choose led strip as lighting ,decoration product .

Product Image



Product Features

Adopt international leading manufacturer's chips, high brightness ,width angle, color consistency, and stable performance. Each lamp beads form a loop, can be arbitrarily cut according to the needs of the length of the loop, without damaging other parts Convenient to Fix and install, flexible cutting operation make the construction simple, ultra-thin design, low temperature light source, make the selection for the auxiliary material more flexible on installation Energy saving, high brightness, low heat, low energy consumption, Clean, and beautiful;Use DC5V to power supply, more safe on using Protection Rating: IP62 IP65 IP67 IP68, use range is wide Small, multi-colored, no radiation, high shock resistance, is a new generation of green high-tech products

Product Warranty

Warranty is 2 Years

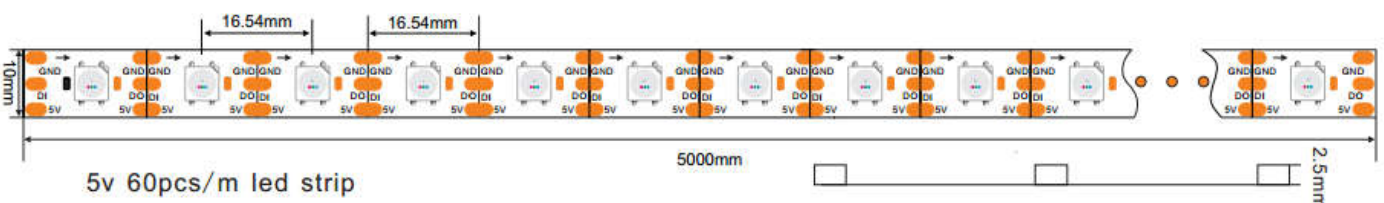
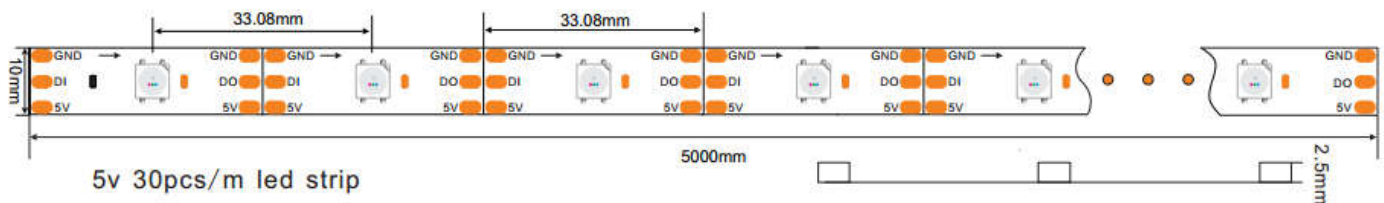
Applications

- City skyline lighting (villas, advertising wall signs, Christmas decorative landscape)
- Irregular design of body decoration (shopping malls, hotels, nightclubs, polygonal wall KTV establishments ceiling cavity design)
- Public lighting lighting (schools, libraries, hospitals, airports, subway stations, railway stations)
- Furniture dark groove trim (door, bar, wine cabinet, wardrobe, TV cabinets, etc.)
- Automotive beauty (body, underbody, wheelbarrow balanced car, etc.)

Project Case








LED Strip dimension



Specifications

1 meter led strip Specifications

Specifications	Voltage	Power	Angle	IP Rating	Color	Color Temperature		Ra		Lumen			Life
						Min	Max	Min	Max	Min	Average	Max	
LC-2812BX30X-5V	5V	9W	120	IP62、IP67 IP65、IP68		2800K	10000K	Ra>70	Ra>80	R:36LM G:90LM B:30LM W:510LM WW:510LM	R:45LM G:120LM B:45LM W:540LM WW:540LM	R:54LM G:150LM B:48LM W:600LM WW:600LM	50000H
LC-2812BX60X-5V	5V	18W	120	IP62、IP67 IP65、IP68		2800K	10000K	Ra>70	Ra>80	R:72LM G:180LM B:60LM W:1020LM WW:1020LM	R:90LM G:240LM B:90LM W:1080LM WW:1080LM	R:108LM G:300LM B:96LM W:1200LM WW:1200LM	50000H
LC-2812BX72X-5V	5V	21.6W	120	IP62、IP67 IP65、IP68		2800K	10000K	Ra>70	Ra>80	R:86LM G:216LM B:72LM W:1224LM WW:1224LM	R:108LM G:288LM B:108LM W:1296LM WW:1296LM	R:129LM G:360LM B:115LM W:1440LM WW:1440LM	50000H
LC-2812BX96X-5V	5V	28.8W	120	IP62、IP67 IP65、IP68		2800K	10000K	Ra>70	Ra>80	R:115LM G:288LM B:96LM W:1632LM WW:1632LM	R:144LM G:384LM B:144LM W:1782LM WW:1782LM	R:172LM G:480LM B:147LM W:1920LM WW:1920LM	50000H
LC-2812BX144X-5V	5V	43.2W	120	IP62、IP67 IP65、IP68		2800K	10000K	Ra>70	Ra>80	R:172LM G:432LM B:144LM W:2448LM WW:2448LM	R:216LM G:576LM B:216LM W:2592LM WW:2592LM	R:259LM G:720LM B:230LM W:2880LM WW:2880LM	50000H

Note, the above optical parameters, models X represents optionally, color, water level, the index display. Etc. , please indicate clearly when ordering.

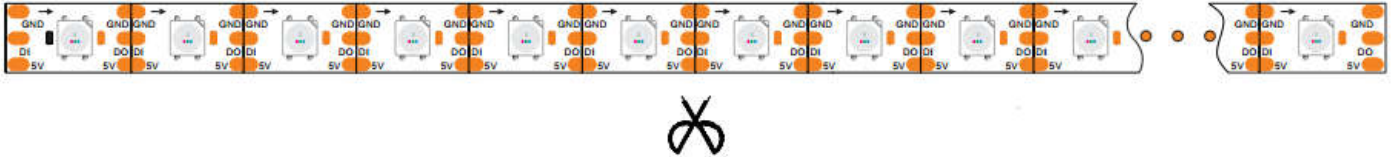
Light decay parameters, test the ambient temperature of 25 degrees, plus or minus 3 degrees

testing time	light fades rate
500H	0%
1000H	0%
2000H	0.5%
4000H	1.2%
6000H	2%

Note, due to the particularity LED product manufacturing process, technical parameters above reflect only the statistics , do not necessarily correspond to the actual parameters of each product, the product may differ from the actual parameters of the above parameters

Strip cut schematic

Scissors to cut at the mouth



LED STRIP wiring diagram

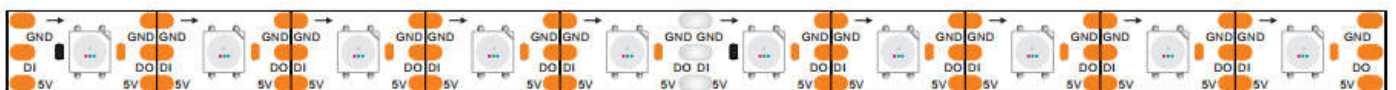
When using lights outdoors, be sure to do waterproof connectors, waterproof power to do well



The heat shrink tubing set into the light band, and then the lights welding firm



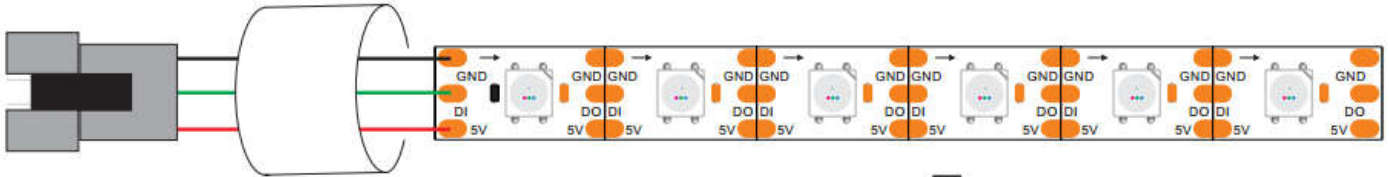
Two weld will shrink tubing to move the welding and weld completely wrap



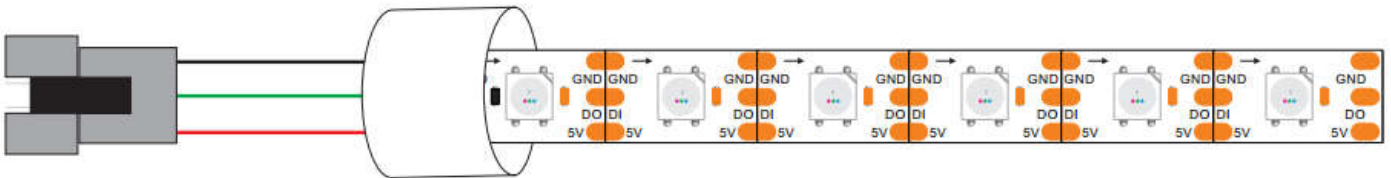
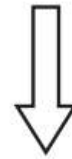
After leaving a good heat shrink tubing, heat shrinkable tube aligned hairdryer to blow completely encased like a light bar



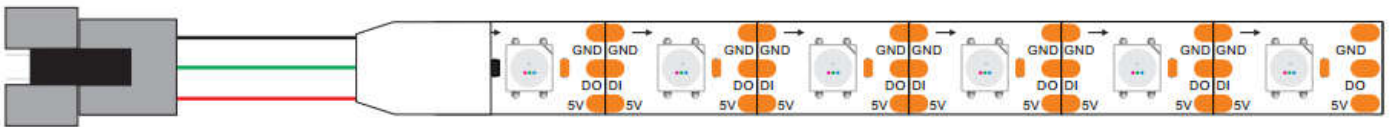
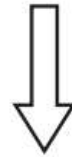
LED Strip wiring diagram



The red, green, yellow and black wires soldered to the lamp terminals tape, then put shrink heat shrink tube

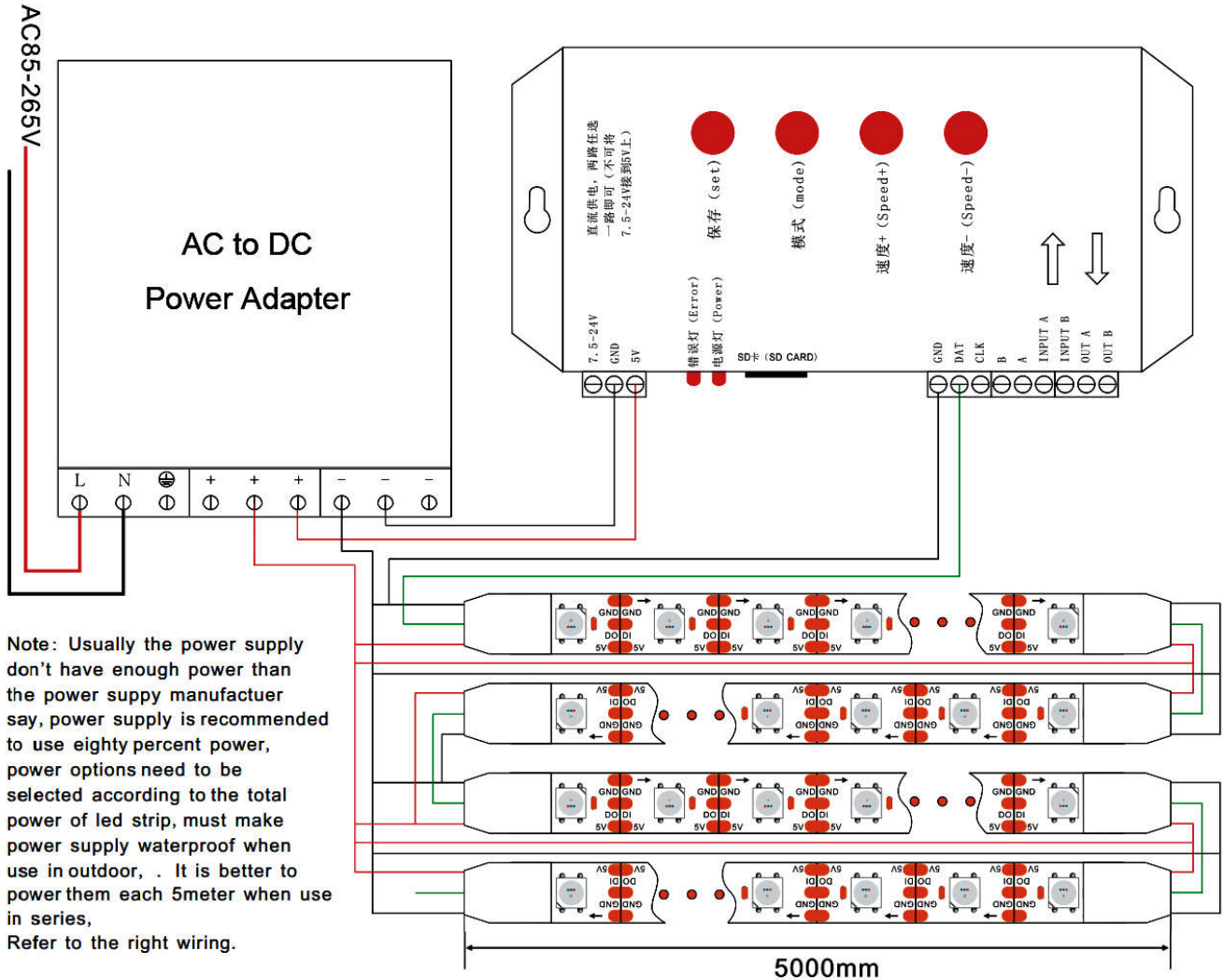


Adjustment shrink tubing, solder joints completely shrink tube



After adjustment shrink tube with hot hair dryer blowing shrink sleeve, shrink wrap the joints until completely

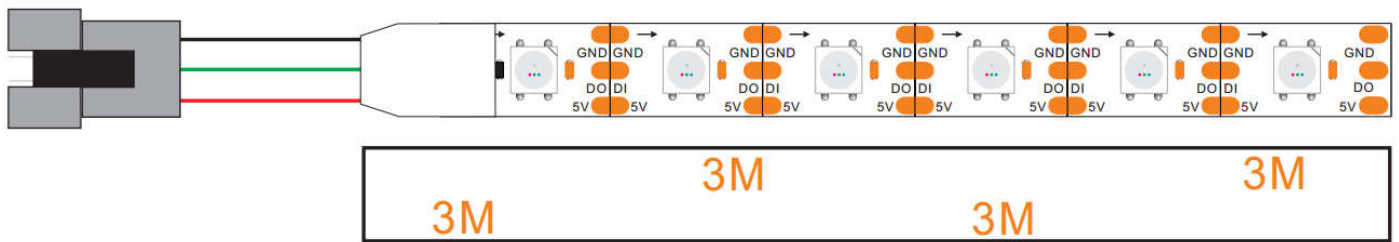
Installation wiring diagram



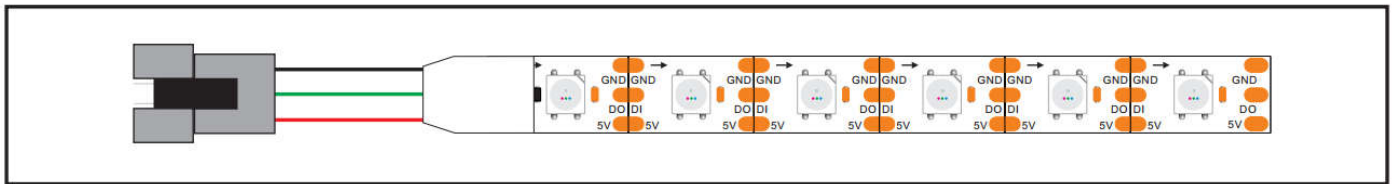
IP30, IP65 LED strip installation diagram



Clean surface debris , ensure the strip surface clean



Pull the 3m paper out of the back of led strip

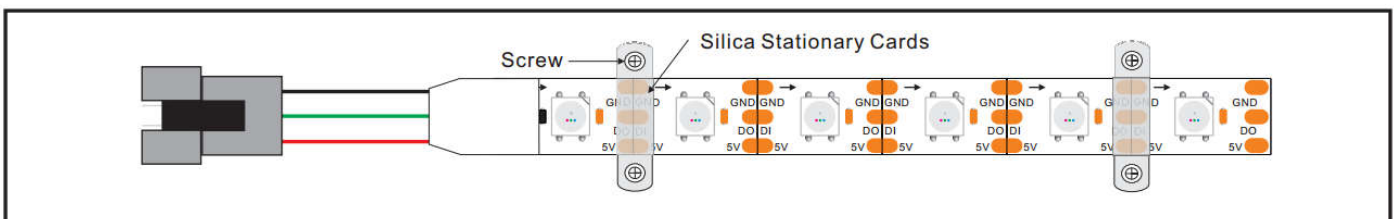


The lights straighten directly fixed to the mounting position

IP67 IP68 LED Strip installation diagram



Clean surface debris , ensure the strip surface clean



Put the led strip on install place , use screw to fixed , each meter need to use 2 silicone snap

Notes

- Please read the specifications and precautions carefully before installation
- Please do not use nails or sharp instruments to install led strip on the object, use double-sided tape or glue or wall mounted on plywood
- Fixed led strip, you can not use acidic glue, it is recommended to use a neutral glue.
- This product is a constant voltage DC5V, make sure Power is right before installation, after installation, check the wiring is correct or not , so that avoid the product been burned
- This product is recommended use in parallel. when the length is more than five meters, the tail so be power to ensure the brightness is enough or uneven

Appendix B

SOTERATM

SYSTEMS

FR1118-P10

User's Manual



WARNING

Read carefully and understand all **INSTRUCTIONS** before operating. Failure to follow the safety rules and other basic safety precautions may result in serious personal injury. Save these instructions in a safe place and on hand so that they can be read when required.

1. INTRODUCTION

1.1 Technical Data

1.1.1 The turbine digital meter is designed for use with the following low viscosity fluids:

- Diesel Exhaust Fluid
- Water (**NOT** for human consumption)

WARNING! Use of other fluids may cause inaccurate readings and can damage the meter!

1.1.2 Flow Rate: 3-26 GPM, flow rates outside of this range may have reduced accuracy.

1.1.3 Operating pressure: 10BAR / 145PSI

1.1.4 Inlet/Outlet: 1" NPT

IMPORTANT! Not suitable for retail sale of dispensed fluids!

1.2 LCD DISPLAY

The meter display features two numerical registers and several function or status indicators.

#	Description
1	Register (5 digit, from 0.1-99999).
2	Battery condition.
3	"Calibration" mode.
4	Resetting current total to 0.
5	Totalizer Register (total cumulative fluid dispensed).
6	Rate of flow being displayed.
7	Unit of measure (liters, gallons, quarts, pints).

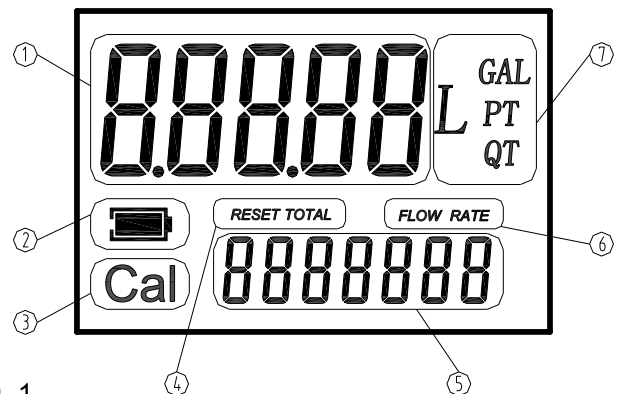


Fig. 1

1.3 USER BUTTONS

The face of the turbine digital meter has two buttons (MENU and RESET) which individually perform two main functions and together, other secondary functions. The main functions performed are:

- "RESET" key: resetting the Register and resettable total (reset total)
- "MENU" key: entering calibration mode.
- Used together, the two keys permit entering configuration mode.

1.4 BATTERY REPLACEMENT

The battery in your in-line digital meter is not replaceable. To minimize the possibility of fire or explosion, it is a sealed unit that does not allow for user replacement. Do not attempt to open or replace the battery in your meter.

2. INSTALLATION

The inlet and outlet for this meter is 1" NPT. It can be easily connected to a pipe or nozzle.

3. DAILY USE

3.1 BUTTON USAGE, CALIBRATION AND MEASUREMENT UNIT CHANGE

- **Reset the present total (See Fig. 2)**
 - 1) When the meter is in standby mode, press the RESET key.
 - 2) The display shows all the segments at once.
 - 3) The meter resets to display "0" on the resettable register.

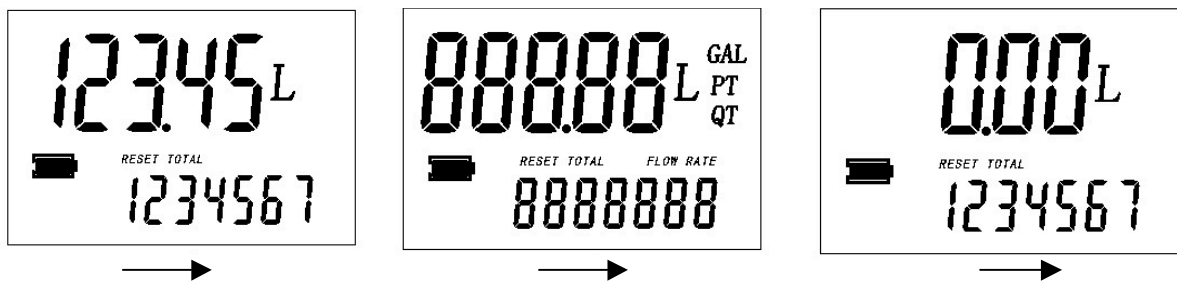


Fig. 2

- **Show current correction factor and general total (See Fig. 3)**

Press MENU and RESET together and hold for two seconds.

Value "1.4000" is the correction factor which can be reset;

"1234567" is the non-resettable total.

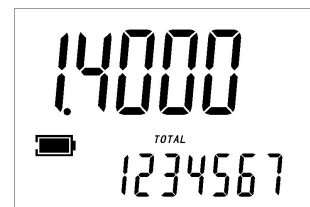


Fig. 3

- **Measurement unit change (See Fig. 4)**

Press MENU and RESET together and hold for five seconds.

The current unit of measure will begin to flash. Press RESET to choose a different unit of measure, then press MENU to confirm.

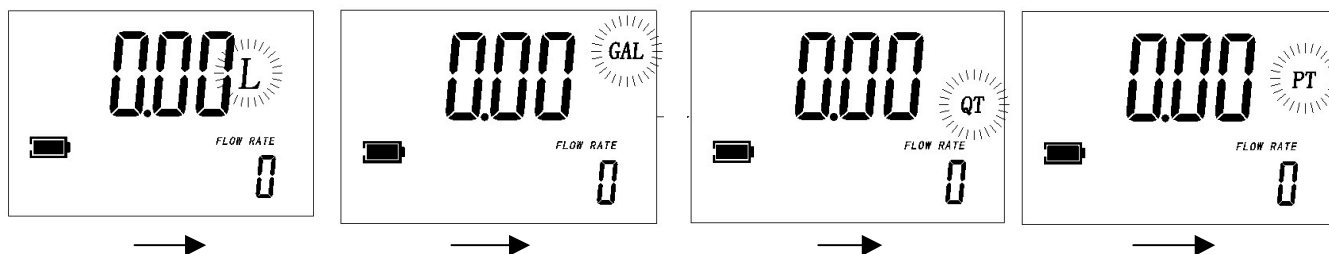


Fig. 4

3.2 RESET THE RESETTABLE TOTALIZER (SEE FIG. 5)

When the meter is in standby, press the RESET key for 2 seconds to reset the totalizer.

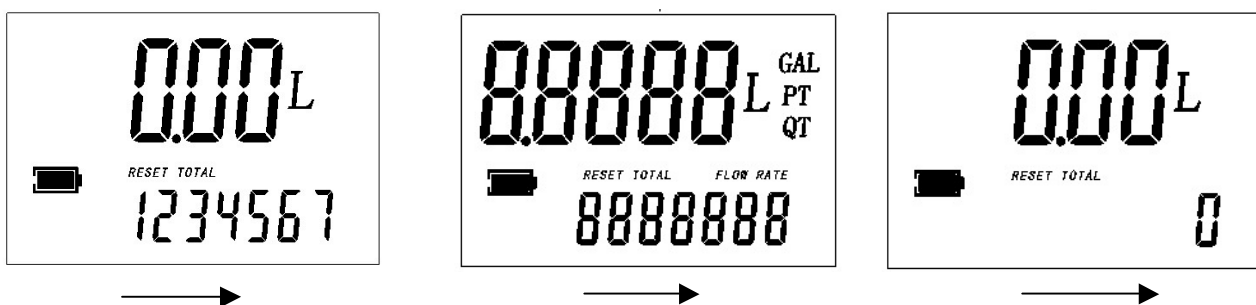


Fig. 5

3.3 Calibration Procedure (Using the Correction Factor)

Carefully follow the procedure indicated below.

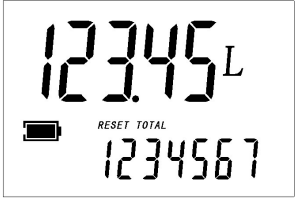
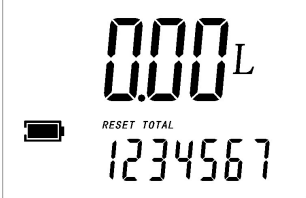
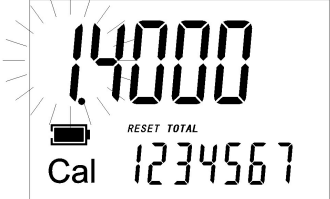

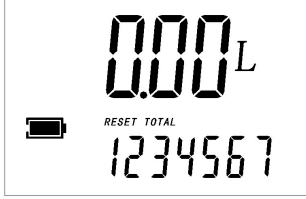
FORMULA

$$\text{Proper correction factor} = \text{current correction factor} \times (\text{actual value} / \text{display value})$$

Example:




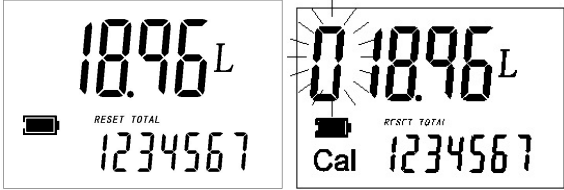
Actual value: 20.75 Display value: 18.96 Current correction factor: 1.000


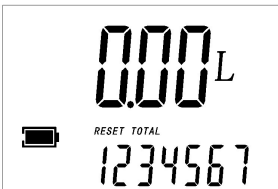
Proper correction factor : $1.000 \times (20.75/18.96) = 1.000 \times 1.094 = 1.094$

1	Wait for the meter to go into standby (blank screen).	
2	With the meter in stand-by mode, reset the resettable total by pressing the "RESET" button.	
3	Press and hold the MENU button until the first digit in the display begins to flash (approximately 3 seconds). The meter is in calibration mode.	
4	Press the RESET button to choose the right digit from 0 to 9. Press the MENU button to start the next digit. So the digit of correction factor can be changed one by one.	
5	Make sure the correction factor is right, press the MENU button. Keep it pressed until the meter exits calibration mode (approximately 3 seconds). The factor is now saved.	

3.4 MODIFY THE CORRECTION FACTOR IN FIELD

PLEASE CAREFULLY FOLLOW THE PROCEDURE INDICATED BELOW.

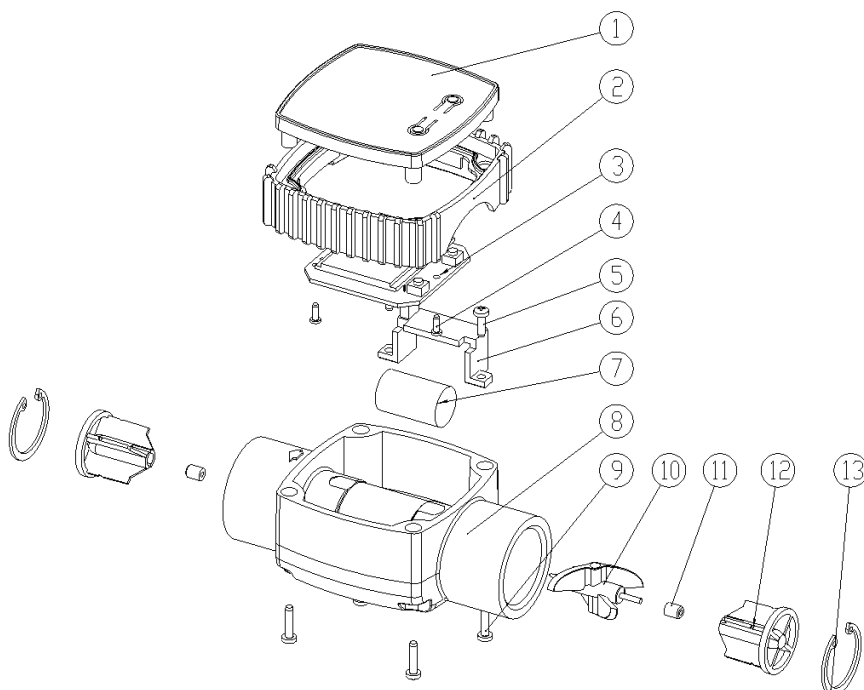
1	Wait for the meter to go to standby.	
2	Reset the register.	
3	<p>Start dispensing into a calibration can. Stop dispensing when 5 gallons of volume is reached. Read the amount displayed vs. the actual amount dispensed into the calibration can. The volume that is displayed on the LCD is the Display Value, not the Actual Value.</p> 	

4	Press and hold the MENU button until the far right digit flashes. Press the RESET button to choose the right digit from 0 to 9. Press the MENU button to go to the next digit so that the Actual Value can be input.	
5	Make sure the correction factor is right and then press and hold the MENU button until the meter exits calibration mode (approx. 3 seconds). The meter is now calibrated and will return to standby.	

4. Troubleshooting

Problem	Possible Cause	Corrective Action
LCD: no indication	Bad battery contacts	Check battery contacts
Imprecise measurement	Wrong Correction Factor	With reference to paragraph 3.3 & 3.4, check the Correction Factor
	The meter works below minimum acceptable flow rate	Increase the flow rate until acceptable flow rate has been achieved (3 – 26 GPM)
Reduced or zero flow rate	Turbine blocked	Clean the turbine.
The meter does not count, but the flow rate is correct	Incorrect installation of gears after cleaning	Repeat the reassembly process.
	Possible electronic card problems	Contact your meter distributor.

5. DIAGRAM AND PARTS LIST



No	Description	Qty.
1	Meter Cover	1
2	Rubber Protection	1
3	Electric Board	1
4	Pin	1
5	Self-Tap Screw	2
6	Battery Holder	1
7	Self-tap	3
8	Meter body	1
9	Self-Tap Screw	4
10	Turbine	1
11	Bearing	2
12	Bearing Holder	2
13	Retaining Clip	2

LIMITED WARRANTY POLICY

Revision Date: August 1, 2014

Fill-Rite and Sotera Products

Tuthill Transfer Systems ("Manufacturer") warrants each consumer buyer of its products ("Buyer") from date of sale that goods of its manufacture ("Goods") shall be free from defects of materials and workmanship.

The duration of the warranty is as follows:

From Date of Sale	Not to Exceed the Following Period from Date of Manufacture	Product Series	
Five Years	60 Months	SP100 Series Pumps	400 Series Pumps
Two Years	27 Months	Heavy Duty Pumps and Meters, 820, 825, and 850 Meters	Cabinet Pumps, Cabinet Meters, TN Meters, TM Meters, TS Meters
One Year	15 Months	Standard Duty Pumps and Meters, 1600 Pumps	Accessories Parts

* proof of purchase should be presented to place of purchase

** see Appendix for definition of "Heavy Duty" and "Standard Duty" products

End users must contact the place where they purchased the product to process a warranty. "Place of purchase" is defined as any authorized TTS Distributor, including any and all retail stores, mail order houses, catalogue houses, on-line stores, commercial distributors.

Manufacturer's sole obligation under the foregoing warranties will be limited to either – at Manufacturer's option – replacing defective goods (subject to limitations hereinafter provided) or refunding the purchase price for such Goods theretofore paid by the buyer, and Buyers exclusive remedy for breach of any such warranties will be enforcement of such obligations of the Manufacturer. If the Manufacturer so requests the return of such Goods, the Goods will be redelivered to the manufacturer in accordance with Manufacturer's instructions FOB Factory.

The remedies contained herein shall constitute the sole recourse of the Buyer against the Manufacturer for breach of warranty. **IN NO EVENT SHALL THE MANUFACTURER'S LIABILITY FOR ANY CLAIM FOR DAMAGES ARISING OUT OF THE MANUFACTURE, SALE, DELIVERY, OR USE OF THE GOODS EXCEED THE PURCHASE PRICE.**

The foregoing warranties will not extend to goods subject to misuse, neglect, accident, improper installation or maintenance, or have been repaired by anyone other than the Manufacturer or its authorized representative. **THE FOREGOING WARRANTIES ARE EXCLUSIVE AND IN LIEU OF ALL OTHER WARRANTIES OF MERCHANTABILITY, FITNESS FOR PURPOSE OF ANY OTHER TYPE, WHETHER EXPRESSED OR IMPLIED.** No person may vary the foregoing warranties or remedies, except in writing signed by a duly authorized officer of the Manufacturer. The Buyer's acceptance of delivery of the Goods constitutes acceptance of the foregoing warranties and remedies, and all conditions and limitations thereof.