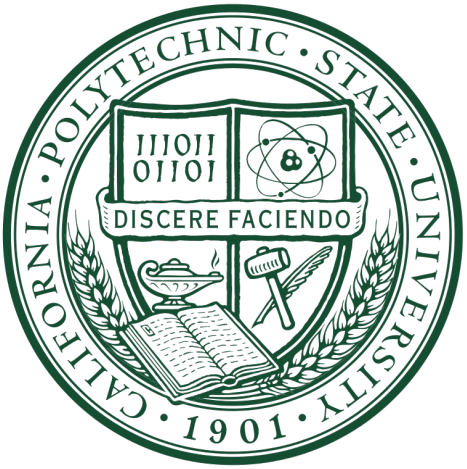
Blend-It

Wine Blending Distribution System

Final Design Report

June 4th, 2017



Connor Clarry, ME

Matt Moren, CPE

Russell Temple, ME

Statement of Disclaimer

Since this project is a result of a class assignment, it has been graded and accepted as fulfillment of the course requirements. Acceptance does not imply technical accuracy or reliability. Any use of information in this report is done at the risk of the user. These risks may include catastrophic failure of the device or infringement of patent or copyright laws. California Polytechnic State University at San Luis Obispo and its staff cannot be held liable for any use or misuse of the project.

Table of Contents

Table of Figures

Executive Summary

Greg Fields, the owner of Blend-It Winery and Bistro, wants to open an urban winery where the customers can automatically design their own custom wine blends. The project was worked on in conjunction by two different teams. One team which consisted of two mechanical engineers and one computer engineer were responsible for designing and building the physical system itself. The other team consisted of nine computer science majors who were responsible for developing the mobile applications as well as the server. The teams designed a system that receives a wine blend order from a customer application and automatically dispenses the blend in the machine. The machine can be hooked up to five different kegs simultaneously, allowing for countless different wine blend combinations. The system uses a Raspberry Pi running Node-Red, as well as two Arduinos, to perform all of the logic operations in the system. The pouring for the device is controlled by electronic three-way valves which use infrared flow meters to accurately measure the amount of wine that should be poured. The following report breaks down the design, build and test process the physical device went through.

Chapter 1: Introduction

Currently, there is no method to quickly and precisely create new, unique wine blends. Blend-it Winery and Bistro is an urban winery that wants to revolutionize the wine-tasting experience. This project's goal was to develop an automated system that allows customers to design their own custom blends, and have it dispensed in front of them.

The client, Greg Fields, is an individual with extensive background in Computer Science and Information Resource Management, with a passion for viticulture and winemaking. Due to his background, he is an advisement resource that our team will often check in with to share ideas and verify feasibility. In researching the wine industry, he has found a way to create leverage on the market: a micro-winery that offers on-demand, custom blends by the glass. Additional stakeholders for this project include: the bartender who will have to operate the machine itself, the restaurant customers who will order the drinks, and the parts suppliers. While Greg's insight and desires are most important, we must also keep other stakeholders in mind.

This project was worked on by two different senior project teams. Our team consists of two mechanical and one computer engineer from Cal Poly. We were responsible for developing the physical system to simultaneously pour specifically designed blends of up to five different wines.

The other team is composed of nine computer science engineers who are split into three different groups, each focused on a different aspect of the interaction between the customer and the device itself.

Groups:

- Customer Application: Tablet application for inhouse selection of wine varietal percentages.
- Bartender Application: Gives the bartender control over which blends the machine will pour at appropriate times.
- Server: Collection of data from the applications to send pertinent information to the machine.
- Machine: Apparatus to draw wine from kegs based off customer input.

Our device is responsible for communicating to the server directly. It will not have direct access and communications with either the consumer or bartender applications. Our device will be able to take the data it receives to electronically control the wine dispensing.

Management/Teamwork

During these last seven months our team worked both effectively and enthusiastically on our project. The team is composed of one computer engineer and two mechanical engineers, so our roles were initially tailored to best fit our majors. The computer engineer (Matt) is more focused on the coding and electric circuit portion of the project, while the mechanical engineers (Connor and Russell) were tasked with designing the physical system itself. However, as time passed each member of the team began to take on more individual roles. Matt was in charge of developing the code for the Arduino and Raspberry Pi. He has also spent a great deal of time researching and doing calculations for the power requirements of our device. Russell spent time researching and ordering some of the mechanical parts such as adapters. He also developed tests which we used for design verification. Connor spent most of his time assembling and manufacturing the actual device. He also spent time investigating the communication protocols which we use to interact with the server. Another aspect of our team interactions involves collaborating with the other senior project team. We participated in bi-weekly meetings with them to ensure that we were on same page and to answer any questions that arose. We also have Slack and GitHub accounts which allows us to collaborate on our code.

Chapter 2: Background

Presently, there is no system with the ability to mix various kegged wines via an automated process. The current practice for wineries that allow customers to create their own blends involves sitting down at a table with a variety of different wine bottles or barrel samples and then hand measuring and pouring each sample into a glass. This existing method used by the wine industry is outdated, inefficient, and imprecise. The percentage of wine drinkers who are between the age of 21 and 29 has increased substantially from around 23% in 2010 to 36% in 2015 [1]. However, this age group is visiting tasting rooms far less when compared to older generations. This may be because the younger generation believes that the current wine tasting experience is intimidating, not engaging and slow paced. Greg envisions a computer-controlled

wine blending system integrated with smart phone applications, and has tasked us with its creation. They can access the application on their phone while they are in the restaurant or they can use one of the tablets available throughout the establishment.

When developing potential solutions for this project, we considered a variety of needs. The device must be able to...

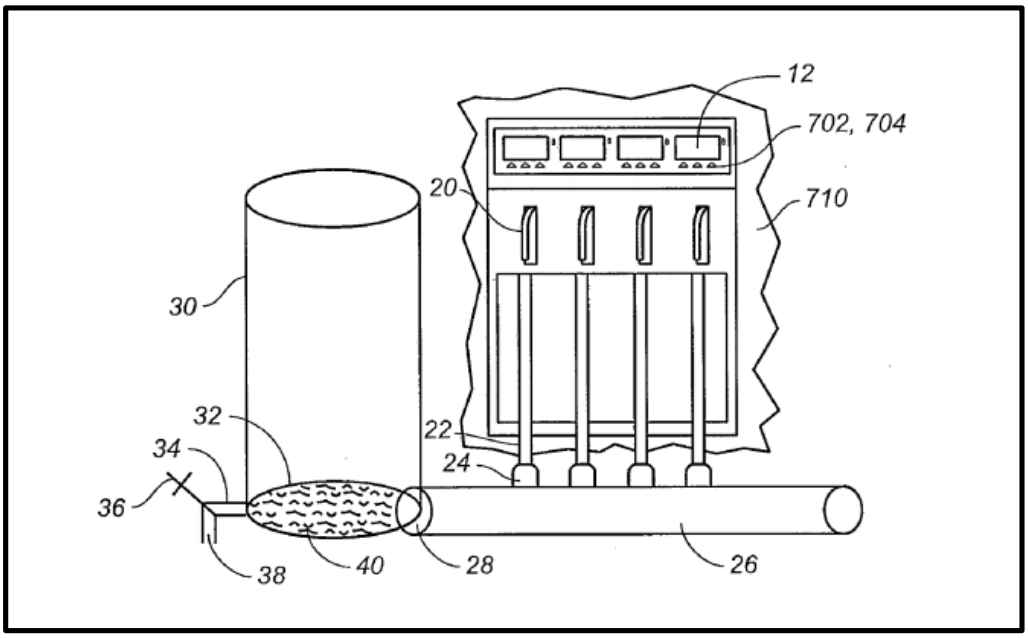
- Dispense wine in very small quantities for wine tastings. The typical pour for a tasting is about 1.5oz or 45mL. This means that in a hypothetical blend in which a customer selects 3% of a certain wine, the device must be capable of dispensing only about 1.35mL of wine.
- Fill a bottle of wine in a reasonably short amount of time. This is because a bartender does not want to be held up if a number of bottles have to be filled.
- Receive the commands from an external server and perform the required pouring actions immediately.
- Prevent the wine from being exposed to oxygen before it is poured. Exposing the wine will cause oxidation and have a detrimental effect on the quality of the wine and flavor.

Benchmarking

The benchmarking sections discusses five similar devices that the team researched. These devices gave insight into the challenges we'll face, and ideas for methodology based on field-tested devices.

Wine Blending System and Method

During the initial background research for this project we were unable to find any patents for designs that were similar what we envisioned. However, a few weeks ago, a patent lawyer,



working for Greg Fields, found a patent(#US20150336784) that was almost identical to our potential design (**Figure 1)**. The patent was filed by Napa Technology and is described as Wine Blending System and Method **[2]**. The patent is vaguely worded and fairly all-encompassing so it covers most similar ideas. This discovery does not change the design or development of our device. Greg believes that we can instead file an improvement patent by adjusting the way in which the user selects their wine blend within the customer application.

Figure 1: Patent #US20150336784 Schematic

Coca-Cola Freestyle Machine®

One device that shares similarities with our design is the Coca-Cola Freestyle® machine **(Figure 2**). The Coca-Cola Freestyle® is a touch screen soda fountain which is capable of dispensing 165 different flavors of carbonated and non-carbonated beverages.

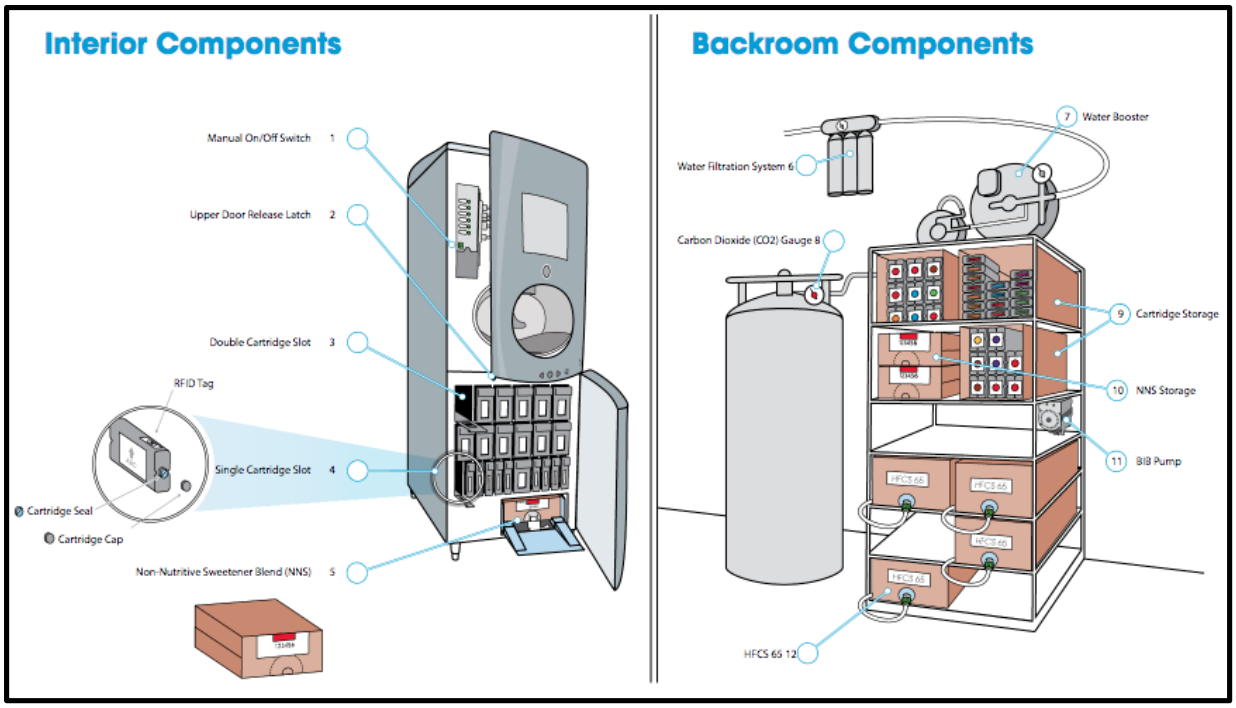


Figure 2: Coca-Cola Freestyle® Machine Diagram

The Coca-Cola Freestyle® machine allows the customer to use a touchscreen or phone application to select the drink flavor of their choice and then dispense the exact amount they want. The machine uses a cartridge of highly concentrated flavoring which is blended with carbonated water and sweetener. This product is similar to our desired device because it allows the user to quickly dispense their drink of choice and it allows the user to directly interact with it electronically. However, the Freestyle is not capable of measuring out exact ratios of different drink types because the pour time is entirely dependent upon how long the user holds down the dispense button. The Freestyle is also different because the drink flavors are simply injected into a stream of carbonated water rather than being pumped or drawn from an external container. [3]

Automated Color Dispensing System and Method

Another product that has some similarities to our project is an Automated Color Dispensing System and Method, or Patent #US8666540 B2 (**Figure 3**), known colloquially as an automated paint mixing machine [4]. The user inputs a desired color, then the machine selects a base paint color and then injects it with a variety of different tints that are determined by predefined color formulas. These tints are drawn from color cartridges using pumps and

dispensed from injector nozzles. This system is similar to the device that we intend to create because they are both capable of dispensing extremely small amounts of liquid with high accuracy, but the paint machine uses a pump to draw the liquid out of a cartridge, while our device will only need to control the flow of the pressurized wine. Also, oxygenation is not a concern for the paint machine, but is critical to avoid in our device.

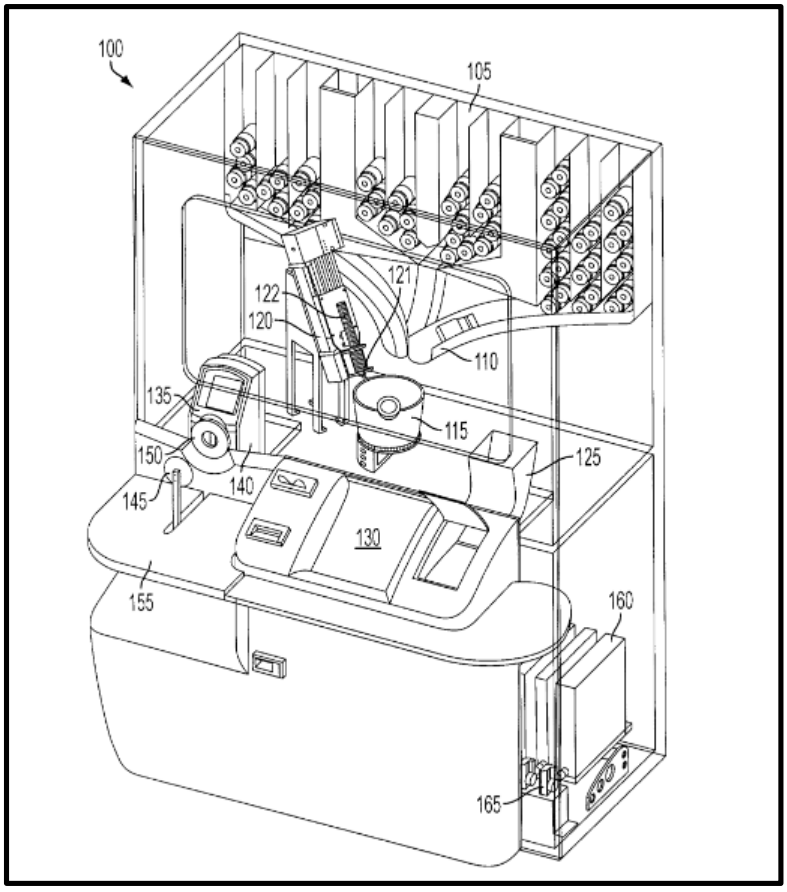


Figure 3: Overview Image from Patent #US8666540 B2

Brita Hydration Station 2000S®

An additional existing product that shares functionalities with our device is the Brita Hydration Station 2000S® (**Figure 4**). The 2000S is a wall mounted water filtration system that uses an electronic motion sensor to turn on and off depending on whether there is a bottle placed underneath the spout [5]. The 2000S is similar to our wine blending device because it uses an electronic valve to control the flow of a pressurized liquid. However, the 2000S is not capable of

dispensing predetermined and exact quantities of liquid, which is a characteristic that is essential to our design.

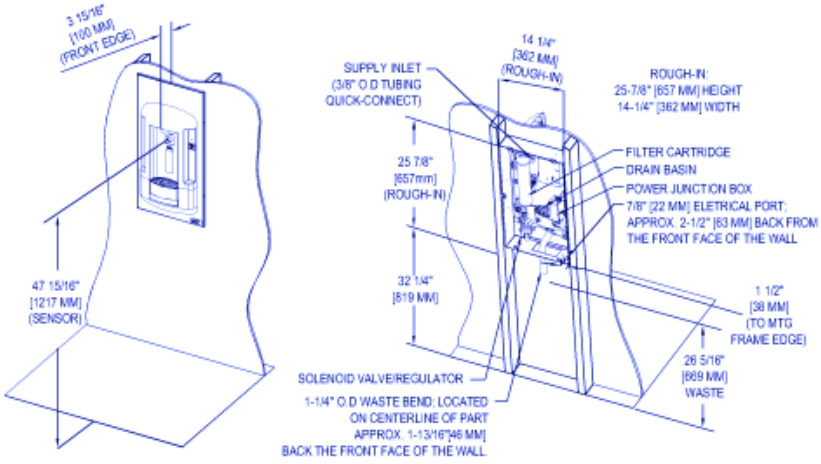


Figure 4: Brita Hydration Station 2000S® Technical Drawing

Monsieur Automated Bartender

One final product that shares similarities with our wine blending device is the Monsieur automated bartender (**Figure 5**). For this device the user simply selects the drink of their choice from the built in touchscreen and the machine will automatically mix the corresponding ingredients in the exact amounts required. The Monsieur and our wine blending device are both capable of dispensing liquids in very accurate amounts; the Monsieur differs from our device, because it uses peristaltic pumps to draw the liquid from one of eight different ingredient containers while our device will use a pressurized valve [6]. Additionally, because the Monsieur doesn't dispense wine, it does not need to account for oxidation contamination.

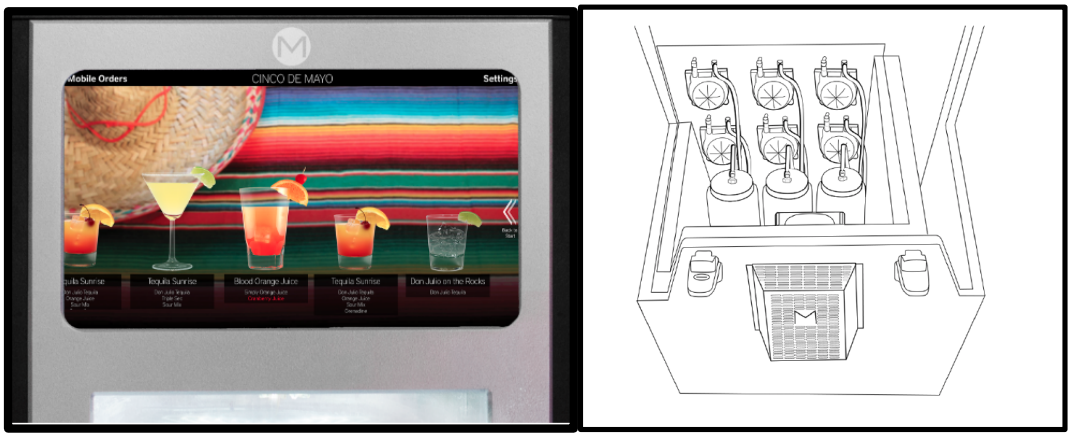


Figure 5: Monsieur Interface and Internal Design Drawing

Chapter 3: Design Development

The following subsections: Objectives, Requirements, Specifications Breakdown; will explain how the customer objectives yielded the engineering specifications for the device.

Objectives

As summarized in the background, the objective for this project it to develop a prototype device that can take up to five different kegs of wine and dispense an accurate blend of the wines which have been selected by the customer using a phone application. Additionally, the pour volume will vary for a tasting, glass, or bottle. The pours need to dispense at a reasonable rate and the device must be easy to maintain and operate. The input data will originate from an application currently under simultaneous development. The machine must utilize microcontrollers to parse the input received and send the desired information to the device accordingly. These microcontrollers will need to be reliable and able to run constantly, and the microcontroller code must be well-documented and easy to decipher, to allow for future modification and calibration. Because the electronics of the system will be in close proximity to the wine, it is important to fully enclose the electronics in a sealed enclosure. This will prevent them from coming into contact with wine or other liquids while the device is operational. This system will be designed to work with the other team who is responsible for the development and implementation of mobile applications that talk directly to the device. The tubes and valves need to be free of oxygen permeation to prevent tainting the wine as it travels from the keg to the

customer. We are also aiming to minimize the amount of custom-made hardware in the design. By using pre-manufactured parts in our design, the device will be easier to repair, maintain, and operate.

Requirements

In order to effectively turn the requirements described above into measurable engineering specifications we used Quality Function Deployment (QFD). This QFD shown in Appendix A contains everything from customer requirements to engineering specifications along with a review of how customers are satisfied by competing products. The descriptions listed under the voices section represent the customer requirement, weighted on a 1-5 scale, with five being the most important. The terms listed under the measures tab represent the specifications for the design. The QFD is then filled in depending on the strength of the relationship between each customer requirement and specification. The targets at the bottom of Table 1 represent the required value for each specification. There is also a customer ratings heading under which the competition is benchmarked for each of our requirements. After we determine the requirements and select the relationship strength with each choice, the QFD process weighs the importance of each requirement and lists it as a percentage of a whole at the bottom of the chart.

The QFD analysis shows the competing product with the highest customer rating is the Brita Hydration Station with a score of 45 while the other products had a maximum score of 38. This suggests that investigating the design of the Brita is most beneficial to the development of our device.

Spec #	Specification Description	Target (units)	Tolerance	Risk
1	Pressure	10 psi	Max	L
2	Flow Rate	12.5 mL/s	±0.5mL/s	L
3	Tubing Inner Diameter	5/16”	Go/ No Go	L
4	Material	Wine/Food Safe	Go/ No Go	L
5	Disassembly	Quick and easy	Go/No Go	L
6	Power Rating	12V	Max	L
7	Electrical Safety	No shock hazards	Go/ No Go	L
8	Organized Code	Adheres to Code Guidelines	Go/ No Go	L
9	Cost	500	Max	M

Table 1: Device Specification Requirements

Listed above is a copy of our engineering specifications (Table 1). The rationale is explained in the breakdown below.

Specifications Breakdown

- Organized Code (~17% weight)
    - This is weighted as most important because it affects every other part of the project. We will adhere to the NASA C code specifications. [7]Code must be well commented especially where system changes may occur for tuning. There is little to no risk for this specification.
- Max Pressure (~15% weight)
    - The line must not be pressurized to more than 10 psi. Anything higher than 10 psi will carbonate the line. Kegs available on the open market exceed our pressure requirements by about 15 times. So the system should easily be able to withstand the pressure.
- Flow Rate (~15% weight)
    - The flow rate is of high importance because if the flowrate is not exactly what we need, then the amount of wine poured in a glass will be incorrect. This is a low risk specification because the flowmeter is heavily tested before it is shipped to us.
- Materials(~12% weight)
    - The materials used in our design need to be made out of wine/food safe material. This will ensure that we do not negatively affect the flavor of the wine. This is a low risk specification because we will verify that all of our products are clearly identified as being food/wine safe.
- Disassembly (~12% weight)
    - The device should be simple to disassemble in order. This will make cleaning and replacing parts a relatively simple process.
- Power Consumption (9% weight)
    - The power box that we are using in our design has a maximum output of 12V. This means that all peripherals must not require more than 12V. This is a low risk requirement because we will not purchase parts that require more than 12V.

Additionally, microcontrollers offer pre-defined low-power operation modes.

- Electrical Safety (~9% weight)
    - Our method of power distribution used a single wall outlet. This minimized the electricity used and makes our final design more adaptable and electrically safe.
- Tubing (~8% weight)
    - This specification is a low priority because the tubing need only be non-toxic and allow sufficient flow rate, both of which are covered in other specifications.
- Max Cost (~8% weight)
    - This specification is not heavily weighted because our sponsor stated that our budget was flexible, within reason.

Concept Generation

Once we figured out which specifications and requirements needed to be met, be began the process of concept generation. This concept generation was accomplished during idea generation sessions that we performed as a team. These idea generation sessions utilized creative brainstorming techniques to develop new ideas and concepts. The first idea generation method that we used was called brainsketching. Brainsketching is a technique in which each member of the team is given a piece of paper and then allotted five minutes to sketch a design idea. After those five minutes are up they have to pass the paper to the next teammate who is then given 5 minutes to add to the initial sketch. This process continues until each member of the group has contributed to the sketch. Our brainsketch resulted in a simple outline of our entire system as seen in Appendix A. This helped us to move forward with our design because it gave us a better idea of which parts and pieces would be needed to bring the whole thing together.

Another type of concept generation that we performed was an initial code brainstorming session in conjunction with the other senior project group. Some of the initial notes can be found in Appendix A. While working with the other teams, we envisioned various potential scenarios our device could encounter. For example, we defined the sequence, from start to finish, of how a drink order would go from being placed by the customer to being poured by the machine. This allowed us to visualize the communication between the device, server, and applications, and fostered discussion of possible impediments. See **Figure 6** for a graphic overview of this example use case.

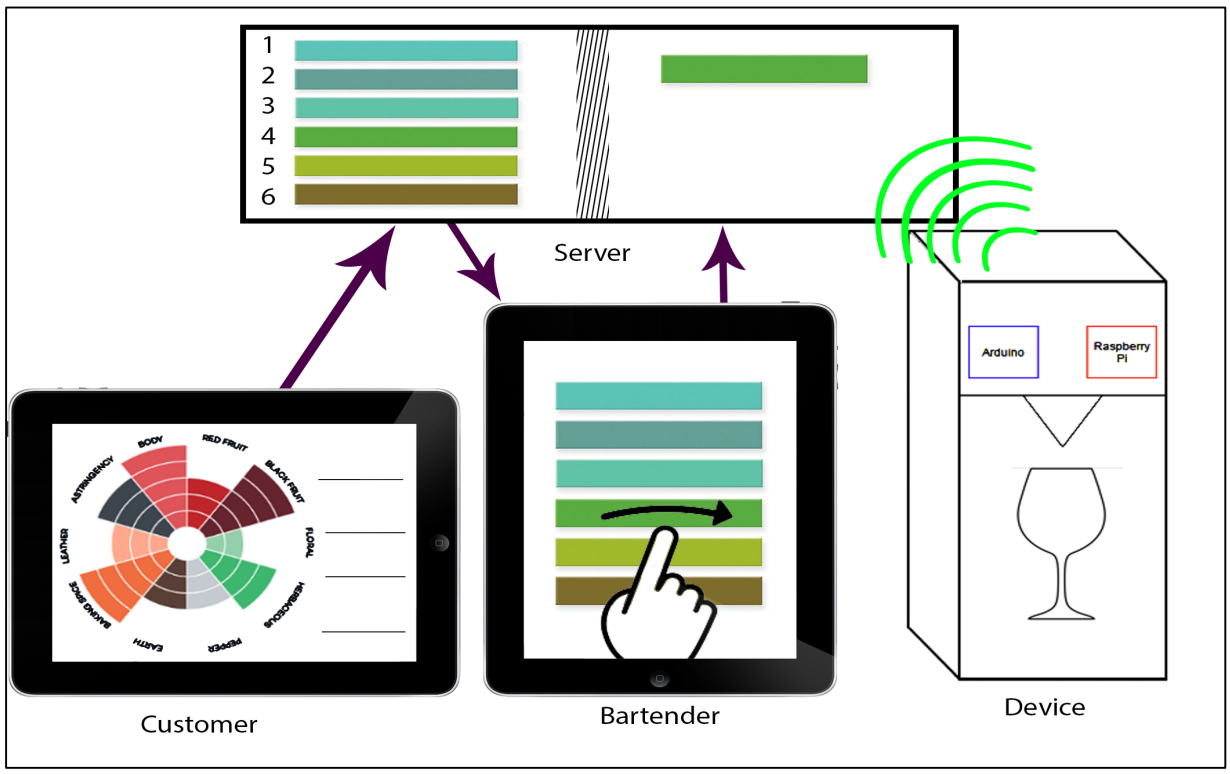


Figure 6: Use Case Graphic

Example Use Case:

1. The customer chooses a flavor profile, which is then converted to blend percentages that match the flavor.
2. The volumes are stored in the server in a queue for the bartender application to access.
3. From the queue on their screen, the bartender can select one pour at a time.
4. The bartender application sends the confirmation to the server.
5. The server sends the pour amounts to the Raspberry Pi.
6. The Raspberry Pi handles the server communication and passes the open time for each valve to the Arduino.
7. The Raspberry Pi monitors the flow rate of each line and relays further commands to the Arduino as necessary.
8. The Raspberry Pi informs the server that the pour is complete.
9. The server requests that the bartender verify the completion of the pour in the app.
10. Repeat as necessary.

Because of the limited amount of ways to pull wine from a pressurized keg, our idea generation

came from specifications of individual components and combinations of components. We evaluated parts and importance of specifications based on their impact to the system and their compatibility with other components. We also looked at a variety of machines that had similar attributes that our final device needs to involve.

Idea Selection

The idea selection process for the device was broken down into the individual components of our design. Each system component was individually discussed.

Kegs/Couplers

Our sponsor provided five ball-lock Cornelius kegs for us. Based on his prior experience, he believes they will serve his purposes, so the team didn't spend any time ideating kegs or couplers. The keg coupler, which connects the tubing to the kegs in a secure manner, has many varieties, but only one type works with the specific ball-lock interface of the Cornelius keg.

Tubing

The device tubing has to be easy to clean and also flexible so that it can be easily manipulated to fit into our final device housing. We wanted the largest diameter tubing possible because this would have the smallest friction losses, which dictates the allowable line length between the keg and nozzle. The maximum internal tubing diameter is dictated by the outside diameter of the ball-lock barb which is 5/16". The most commonly used line materials are PVC, Polyethylene, and 304 stainless steel. However, only stainless steel and PVC are available with an internal diameter of 5/16". The line losses due to friction are .4 psi/foot for PVC and .3 psi/foot for stainless steel [8]. However, stainless steel is $82 for 100 feet of tubing while PVC is only $20 for 100 feet of tubing. Additionally, stainless steel tubing is very rigid and hard to shape while PVC is very flexible. A decision matrix for the tubing typed can be seen in **Figure 7.** Based on this information, we decided that 5/16" PVC tubing was best suited for our application.

				
Concept		PVC	304 Stainless Steel	Polyethylene (Datum)
Criteria	Weight			
Cost	4	S	-	
Aesthtic	3	+	+	
Frictional	5	+	+	
Flexibility	3	S	-	
Inner Diameter	5	+	+	
Food Safe	5	S	S	
	Total	13	9	

Figure 7: Tubing Pugh Matrix

Valve Systems

We developed three possible valve combinations to monitor and control the wine flow. The process used to generate these combinations is described in the flowchart shown in **Figure 8**. This flowchart allowed us to apply our design requirements to the selection process.

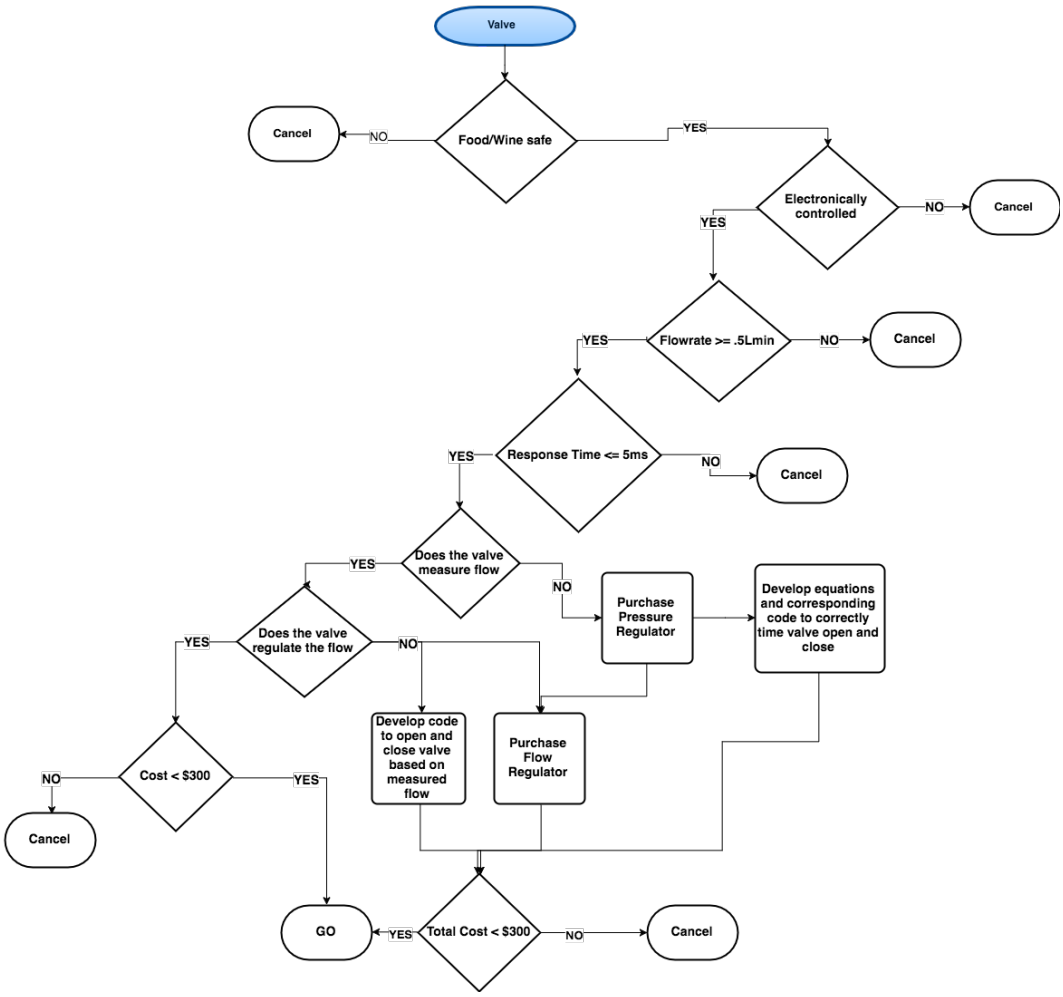
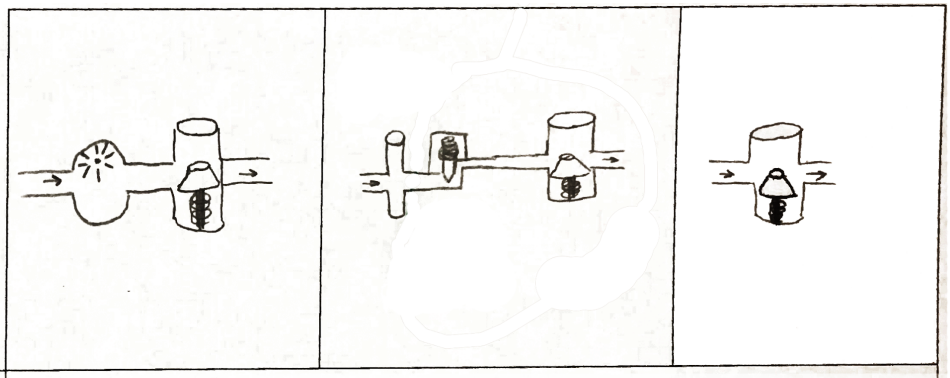


Figure 8: Valve System Flowchart

The first choice was a combination of an on/off valve with a flow-meter. The flow-meter would be used as a feedback loop to our microcontrollers so that we know exactly how much wine is being poured every time the device is being used. This allows our microcontroller to automatically adjust valve open times, increasing pour accuracy. The second choice was a combination of an on/off valve, a pressure regulator, and a flow meter, allowing precise flow rate. The third choice was simply an on/off valve that would require the development of equations to relate valve opening time to amount of liquid dispensed. A decision matrix for these three choices can be seen in **Figure 9.**



	Baseline	Alternatives		
Criteria	Coke Freestyle	Sensor	3 Valve	On/Off
Feasibility	0	0	0	1
Accuracy	0	0	0	-1
Cost	0	1	0	1
Maintainability	0	1	1	1
mL Tracking	0	1	1	0
Code Difficulty	0	-1	0	0
Availability	0	1	-1	1
Food Safe	0	1	-1	1
Flow Control	0	-1	1	-1
SUM +	0	5	3	5
SUM -	0	2	2	2
SUM 0	0	2	4	2
Total	0	3	1	3

Figure 9: Valve System Pugh Matrix

We decided to go with the combination of an on/off valve and a flow-meter. We believe this option is the most cost effective way to obtain the accuracy we desire. With a single on/off valve we have no option to change the flow or amount of wine poured without doing manual calculations. The second option required components that were expensive and food safe versions of each component were very difficult to find. We also realized that having more components in series with each other allows more opportunity for individual parts to fail. Without a system to check on every component in the line regularly, failures would be extremely hard to detect and troubleshoot during device operation.

Microcontrollers

All of the valves in the device will be controlled electronically. In order to develop code modularly and separate tasks, we decided upon two separate microcontrollers. One microcontroller will be solely responsible for communicating with the physical device valves and the other will be responsible for server communication and electronic sensor feedback. For

example, the ON/OFF valves can be tested completely independently of the server communications. However, the second microcontroller must also be housed in the device, which will add to the total power consumption. Our team decided that the benefits of easier testing and maintainability outweigh this downside.

The codeset for each microcontroller will also be modular. The Arduino code will simply open and close each of the five on/off valves on the device, as per instructions from the Raspberry Pi.. The Raspberry Pi code will be responsible for parsing input from the server, which will be developed by another team. Preliminary discussions with the server team suggest that polling a server HTTP socket will be the best form of communication, but code design is not yet complete. In addition to receiving server instructions, the Raspberry Pi code will handle the feedback from flow-meters, report necessary information back to the server, and calculate valve open times to be sent to the Arduino. A chart showing the code outline and affected parts can be seen in **Figure 10**. A complete diagram of the device and the major components can be seen in **Figure 11**.

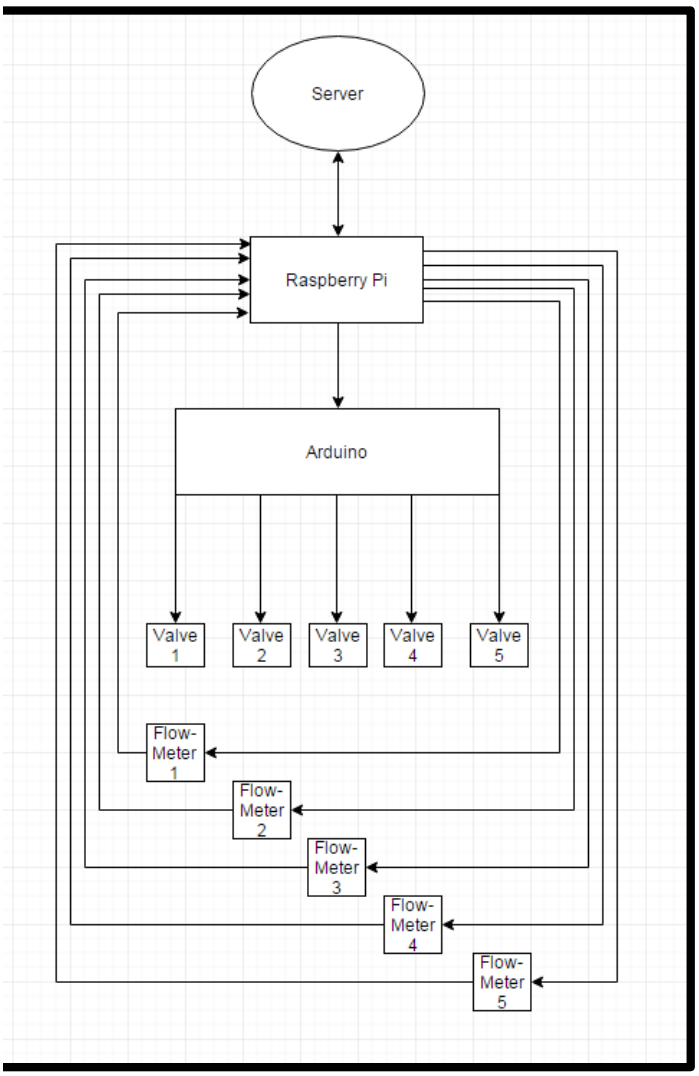


Figure 10: Microcontroller Responsibility Flowchart

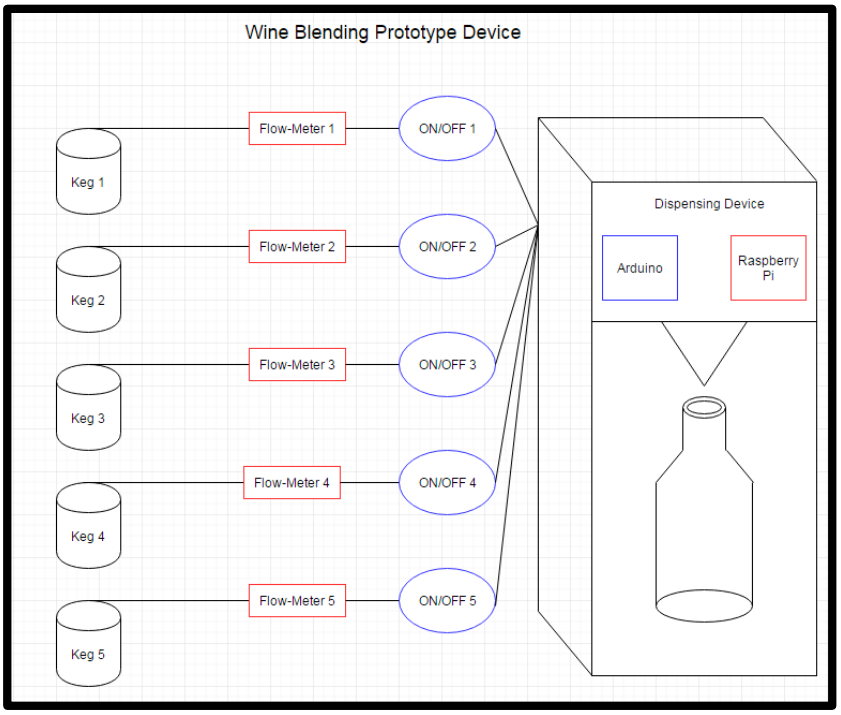


Figure 11: General Device Functional Diagram

Technical Content

PSI Calculations

We had to ensure that the pressure within the keg would be enough to force the wine through the tubing to the valve. We determined this through the use of line balancing, which refers to calculating the line lengths needed to ensure one psi of wine flowing out of the system, when we know that the keg itself will be pressurized to a max of 10 psi. At any pressure higher than 10 psi, carbonation occurs, which ruins the wine[8]. In order to do this we had to account for the frictional losses in the tubing, loss due to gravity, and losses across the valves themselves. The aggregate loss across the valves is about three psi. The average height of a bar is 42 inches and we estimate the output nozzle for our design will be about 18 inches above that. The keg itself is two feet tall but we use the average liquid height of 12 inches. This is subtracted from the nozzle height to get a total elevation gain of four feet. The pressure loss per foot of elevation gain is about .5 psi resulting in a total elevation loss of two psi[8]. When you subtract the losses due to elevation and the valves you are left with four psi if you want to have one psi output. The friction losses in 5/16" PVC tubing are .40 psi per foot of length which means that the maximum tube length is 10 feet[8]. Having 10 feet of tubing would provide a reasonable amount of flexibility in regards to keg placement. However, this length could easily be improved by elevating the keg.

Safety Hazards Evaluation

Another piece of our design evaluation was to perform a preliminary hazards check for our design. We wanted to ensure that our current design did not have any features that would be an issue in the future. We used the Design Hazards Checklist provided to help us identify any problems. The only area of concern was related to our pressurized keg. However, the will not be an issue because we are only pressurizing the system of a maximum of 10 psi. The full checklist can found in Appendix A.

HIGH LEVEL CODE FUNCTIONALITY	
Device: ARDUINO	
Functions	**Details**
receive_input()	This function will communicate with the Raspberry Pi microcontroller and receive the quantities or time corresponding to each ON/OFF valve
open_valve_1()	This will control the first ON/OFF valve
open_valve_2()	This will control the second ON/OFF valve
open_valve_3()	This will control the third ON/OFF valve
open_valve_4()	This will control the fourth ON/OFF valve
open_valve_5()	This will control the fifth ON/OFF valve
blend_current()	This function will take the current input and call all open_valve functions appropriately.
close_all()	This function will be added for safety and will individually close all five valves.

Table 2: Arduino Function Overview

Functional Code Overview - Arduino

A detailed overview of the Arduino code base and functionality can be seen in Table 2. The Arduino code will have functions to control all five of the ON/OFF valves and have a function to read in the data received from the Raspberry Pi. Additional functions will be added as needed for readability purposes. Large functions are undesirable while testing and code maintenance naturally becomes more difficult. Decomposing the tasks of the microcontroller is an important factor in creating, testing, and maintaining code. Some additional functions are likely to be useful for parsing received data into usable data types by the Arduino. Ideation notes

from a syntax free code overview can be found in Appendix A.

HIGH LEVEL CODE FUNCTIONALITY	
Device: RASPBERRY PI	
Functions	**Details**
receive_server_ input()	This function will communicate with the server and receive the quantities or time corresponding to each ON/OFF valve.
flow_meter_1()	This will obtain the data from the flow-meter on valve 1
flow_meter_2()	This will obtain the data from the flow-meter on valve 2
flow_meter_3()	This will obtain the data from the flow-meter on valve 3
flow_meter_4()	This will obtain the data from the flow-meter on valve 4
flow_meter_5()	This will obtain the data from the flow-meter on valve 5
send_blend()	This function will send the necessary data to the Arduino Microcontroller.
server_update()	This function will send all of the flow-meter data to the server so changes can be made to calculations and algorithms if necessary.

Table 3: Raspberry Pi Function Overview

Functional Code Overview - Raspberry Pi

The overview of the Raspberry Pi code can be seen in Table 3. The Raspberry Pi will have functions that receive server input and individual flow-meter outputs. Each flow-meter will send a signal representing the current flow-rate to the microcontroller. This data will be sent to the server so initial calculations can be adjusted based on real-time feedback. This will allow computer calibration for accurate pouring. The functions related to this functionality will be tested extensively to insure all components related and affected by the code are working as expected. It will also have functions to transmit necessary data to the server and to the Arduino.

The code base for the Raspberry Pi is expected to be much larger than the code base of the Arduino, due to the nature of multi-device communications, parsing, and flow-meter feedback adjustments.

Code Clarity

The code will be written and documented in a way that makes it easy to modify in the future. This can be done easily by providing descriptive function names and commenting the code so future users and device administrators can change. Also, the code will adhere to NASA's code safety specifications. [7] Because of the nature of code and the possible need for changes, a small user's manual with the complete original code and all functions described will be included with the final device. This manual will include a detailed description of the code functions and variables, as well as troubleshooting tips. It will also include the exact parts list we use for our final design so that replacement parts can be easily found and purchased when necessary.

System Power

All of our electronic components that require a power source will come from a single power strip from a single wall outlet. This will allow us to have one high power source that we can then distribute to each of the components necessary. This will also make wiring the components of the device simpler. It will allow maneuverability of the device as well as the ability to completely move the system in the event that the customer needs to relocate the system.

Test Fixture Goals

When the test fixture has been completed, we will require a strict testing protocol in order to validate our initial prototype. This initial boundary diagram for the prototype is pictured in Appendix A. This will require graduated cylinders with volumes from 5 to 100 mL. Our initial pours are expected to be error prone, but tuning of our control system will be based off this error. We will test various mL pours and see how the system responds. If it is pouring correctly at some range but not at others, we will have to create separate code systems for the different volumetric ranges. If the system is under-pouring or over-pouring, then we will adjust our controller and sensor gain values accordingly and retest at the same volume to see how the system responds. The more testing we can run, the more accurate the device will become. It is impossible to pour each line at every possible volume, but if we can develop general system

control, and have a sensor that can react to any fluctuations in the system parameters, then it will be possible to have accurate pour volumes under small system variations. See Appendix A for an picture of our test fixture concept modeling.

Test Fixture Limitations

There are some limitations to our test fixture. Due to cost and availability, we will be testing our system with water and air instead of wine and inert gasses. Because wine has different fluid properties, this could affect output flow rate. Ideally, this can be accounted for in the controller by changing the controller and sensor gain values to correspond to the different properties when we eventually test with a keg of wine. Another factor affecting the test results is temperature. In the final concept, the wine kegs will be refrigerated, but our tests will be at room temperature. This will narrowly affect flow rate, and all our components can be operated at lower temperatures, so this is not a concerning factor. From our previous line calculations, the maximum line distance we can have is 10 feet due to the maximum pressure to avoid saturating the wine with inert gas. We cannot recreate this due to space limitations for our testing area. To compensate, we will adjust the keg air pressure to maintain the same output pressure at the On/Off valve that we would see at 10 psi with 10 feet of tubing. Also, we will manually code in the different volumetric inputs and will have no communication with the servers as they are not operational yet. This will not affect testing, but will however be a challenge to integrate once the servers have been completed. Lastly, we will not have the time to run any lifecycle/failure calculations on the components of our system. We cannot guarantee the timespan the machine will be operational or estimate which parts will need replacements after heavy usage.

Chapter 4: Final Design Description

Our final design has three major subsystems: electrical, mechanical, and software. We developed all three of these subsystems concurrently to ensure they cooperate together and all components work as desired. There are unique physical, electrical, and software tests that relate to all of these subsystems.

Final Design

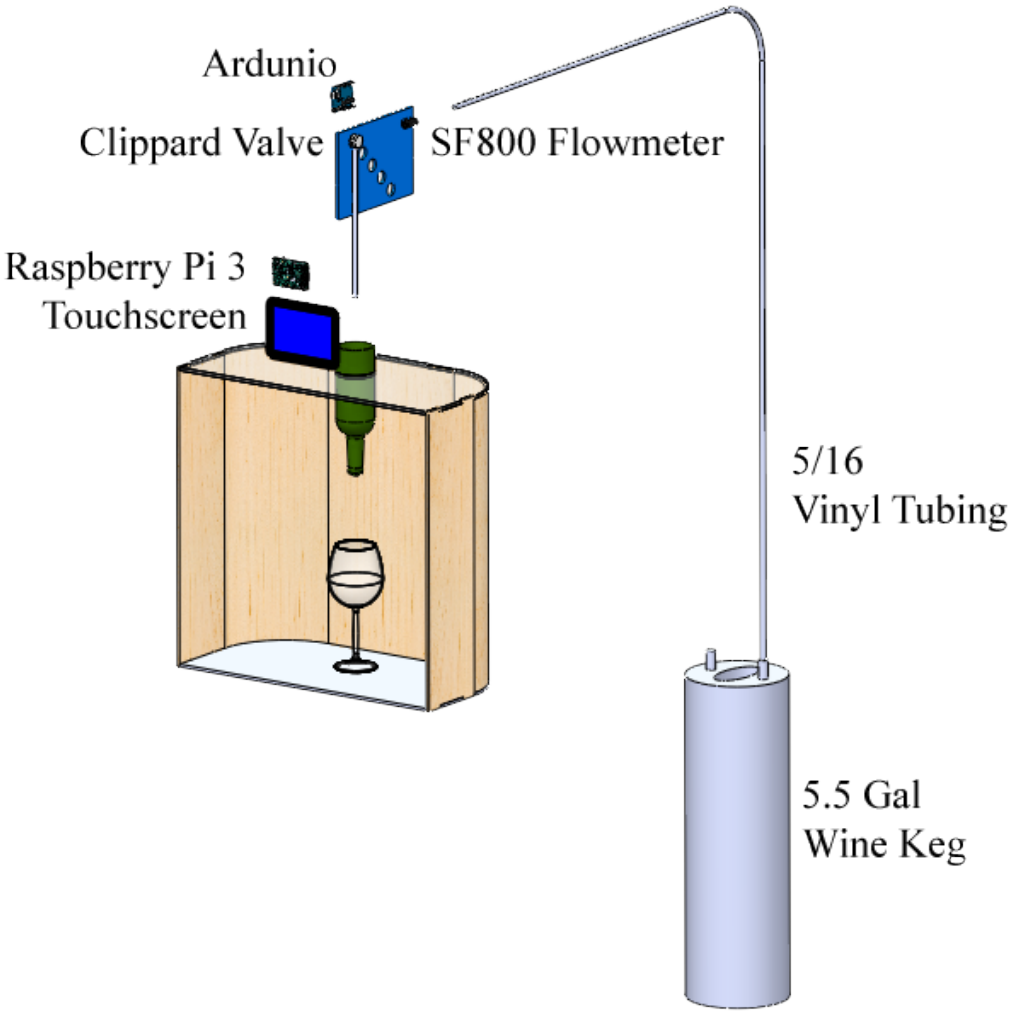


Figure 12: Final concept after CDR review

The final design utilizes all of our previous research. We implemented the standard 5.5 gallon keg selected by our sponsor. From here, we attached the compatible in and out post connections to adapt the inlet and outlet for tubing. Our 5/16" calculated tubing size was implemented and proven to be adequate in moving fluid 10ft at 10 psi. From here, we used our two micro controllers to communicate with the valve through low voltage input to a relay switch that completes the 12 volt valve circuit. Once the circuit is completed, the voltage causes the valve to change from its normally closed state to an open state for a set amount of time. Now that we have proven this concept, we plan to implement logic into the machine using the Swissflow SF800 flow meter. This will allow us to have redundancy in our fluid lines to ensure we are

pouring the precise amount of milliliters specified by the customers interest.

Mechanical System

The mechanical subsystem of this design consists of relatively few components. The components for this subsystem include the valve, flowmeter, keg, tubing, and the adapters which connect each of the respective components to the tubing. The two major components in this subsystem are the valve and the flowmeter.

Valve

One of the most important components that we selected was the Clippard NR4-3(Appendix B). This valve was selected because of its low cost($86) and unique design. This valve is an electronically controlled 3-way normally closed isolation valve. This means that in the off position, liquid inlet is closed while the air inlet is open. When the valve is turned on it, opens the liquid inlet allowing the wine to flow. When the valve is turned off again, the liquid inlet is closed while the air inlet is opened. This flushes all of the wine from the line into the glass, ensuring that our device pours the exact amount it is supposed to. We also selected the NR4-3 valve because we could easily control it with our Raspberry Pi.

Flowmeter

The next major component in our mechanical subsystem is the Swissflow SF800 Flowmeter. This was selected because it was the lowest cost infrared flow meter that was compatible with a micro controller. Infrared flow meters have a much longer lifespan than the common paddle wheel flow meters.

Tubing

The tubing we selected for our design is the Kuriyama clear PVC tubing with an interior diameter of 5/16” ID. This tubing was selected because it met the tubing requirements outlined previously in the report. These requirements were that it be made from PVC and have an inner diameter of 5/16”. A specifications sheet for this tubing can be found in Appendix B.

Keg

This kegs used for this project are 5.5 gallon Corney Kegs. These were provided to us by our sponsor. Appendix B

Adapters

There are two different adapters used in the mechanical subsystem. One adapter is the Ball Lock Disconnects which connect the Corney Keg to the PVC tubing. The other adapter that we used was the ⅛" NPT to 5/16" Hose ID. This adapter connects the tubing to both the flow meter and the valve. Appendix B

Electrical System

A total of five valves and five flow sensors need to be powered for the system to work. Each valve requires a 12VDC source and each flow sensor requires a source in the range of 5VDC to 24VDC. To simply distribute power, we purchased an 18 channel 12VDC power supply unit. This power supply was selected because it neatly housed the electrical circuitry required to split power from a wall (120 VAC) to multiple 12VDC sources. Each of these channels was at an appropriate current and voltage rating to power all valves and sensors. A full system wiring diagram can be seen in **Appendix B**.

Power Distribution Panel

To distribute power to each of our five valves we purchased an 18-Channel Distributed Power Supply Box made by JoyNano that is commonly used for security cameras. The panel takes an input voltage of 120VAC, which is the standard wall outlet power in the United States. The input voltage is split into 18 separate channels that are all at 12VDC with a maximum current output of 1.1A at each channel. Every channel is fused for overcurrent protection. All of the devices we are powering can run at this voltage, making the system circuitry much simpler. The panel has a locking door that will be locked at all times when the device is plugged into for electrical safety. Labels with instructions and warnings will be placed on the outside of the panel detailing how to power down the system and open the case when maintenance or troubleshooting needs to be performed on the unit. This unit greatly simplified our power distribution and comes in a clean and lockable box. This saves us time developing an AC/DC converter to run to our devices and reduced the amount of electrical wiring we will have to perform to create a working system. We weighed time of development and cost of building a distribution circuit ourselves and concluded

purchasing a system from a reliable manufacturer for $50 that met all of our criteria was the best solution. Specifications for this unit can be found in **Appendix D**.

Microcontrollers

We used two separate microcontrollers for our final design. Both the Raspberry Pi 3 and Arduino Uno both provide specific and unique functions to our device. Utilizing these devices together allows us the freedom and modularity our final design requires.

Raspberry Pi Microcontroller

We decided to use a Raspberry Pi 3 microcontroller to handle interfacing with the server team. The cost and accessibility of this microcontroller played a large factor in our decision. This microcontroller will be used to power and receive feedback from our chosen flow-meters. It will also communicate with the Arduino microcontroller with appropriate information about when and what kegs to mix wine from. This controller will also be used to interface with another project team in the future. A common communication protocol will be used for our device to communicate with the server.

Arduino Uno Microcontroller

We decided to use an Arduino Uno microcontroller for valve control. It's compact size allows it to be mounted anywhere we need it and it is simple to program. The Arduino will connect 5 channels of our 8-channel relay. These connections will act as switch controllers for the valves. This microcontroller has a response time of about 5ms and has a low cost. The Arduino is also compatible with our relays and has the ability to communicate with other microcontrollers. All of these factors played into our final decision.

Relays

We chose an 8-channel DC 5V Relay Module for controlling valve switching with our microcontroller. This unit cost about $10. These relays are designed for high-voltage switching up to 250V and can handle current up to 10A. Combined, the five valves the relay module will be in charge of switching will be well below the current and voltage limits. Detailed calculations

showing the power draw of the system through the relays can be seen in **Appendix E**. Initially, we were going to use transistors for our switching mechanism. Transistors are commonly used electrical components used for switching in low voltage devices. The inability of transistors to adequately switch 12V, the necessary input voltage to run our valve circuit led us to use a relay module. The only cons of using relays is that they are louder and are prone to mechanical and/or electrical failure after extended periods of use. The relay module we are using in our design is comprised of 8 separate Songle relays. The datasheet for these relays can be found in **Appendix D**. We feel that we can eliminate the loudness concerns be mounting the relay module inside of the power distribution panel that will normally be closed and locked. Instructions for when and how to replace the relay module will be included in the user's manual we will develop for our final design. Using storage on the raspberry pi microcontroller, we can keep track of the number of times each relay has been switched. This number can be used to alert the user when any of the relays are approaching their posted life cycle expectancy. A button on the microcontroller can be used to output the life-cycle percentage of each relay to the LCD screen. After weighing cost, efficiency, reliability, and ease of maintenance; this relay module was the best candidate to handle valve switching.

Valve Circuitry

Each of the five valves runs at 12VDC with a power draw in the range of 1.0A to 7.2A. From calculations and circuit testing, we concluded that a circuit resistance of 20 Ohms would be sufficient. A detailed chart of the electrical components used in a single line of the circuit used to power the valve can be seen in **Appendix B**.

Flow Meter Circuitry

Each flow meter is rated to run between 5VDC and 24VDC. We will run the meters at 5VDC off of the flow meter Arduino microcontroller. A full chart detailing the resistor values for powering the meters at 5VDC can be seen in **Appendix E**. The flow-meters will run on separate power circuitry, isolating it from the valve and relay circuit.

Software System

There are two main sets of code that will run on our device. The Arduino Uno code will be responsible for operating hardware in the system. The Arduinos will both be flashed with a protocol called Firmata, allowing them to be controlled directly from the Raspberry Pi code without the need to write and change Arduino IDE sketches. The Raspberry Pi will run Node-Red, and will be responsible for communication with the flow meter and valve Arduinos as well as communicating with the server being developed by another team.

Arduino Code

The Arduinos will be flashed with a system protocol called Firmata. This protocol allows easy communication with the Raspberry Pi to receive flow meter input and valve control, without the need to code on the Arduinos and only edit code on the Raspberry Pi.

Raspberry Pi Code

The Raspberry Pi will use a serial communication with the Firmata flashed Arduino Unos. This communication allows the Arduinos to be programmed via the LCD screen which is also running off of the Raspberry Pi. We are planning to use HTTP long-polling to communicate with the server. HTTP long-polling works by constant pinging a server and requesting information. If a new blend is sent to the server, the Raspberry Pi will be notified by polling and properly identified by the server. The Raspberry Pi will then transmit the information it receives to the Arduino to control the hardware necessary to complete the pour. We are also planning on implementing a relay lifecycle monitor on the Raspberry Pi to alert the user via the user interface when a specific relay on the 8-channel relay module is approaching the manufacturer's specified lifetime that can be seen in **Appendix D**. This monitor will be available to view on the LCD screen when a user requests to see it. An automatic message will be displayed when a relay is getting close to the end of its lifecycle.

Cost Analysis

We made an effort to select parts that balance functionality with cost effectiveness. The bill of materials was separated into 5 sections: power, the two micro controllers, and pressurized materials, and the enclosure. All shipping and tax were included into each part. Our final product

was under the allotted budget of $2,500 with a total end cost around $1,500. There were research and development costs associated with testing of improper parts and fittings that pushed sponsor's investment into the machine higher than the cost of making the final design. These costs translated into a total $2,150 spent, while the machine cost was lower at $1,500. We were able to save money on small components like resistors, adaptors, and tubing that were functional yet reasonably priced. With the larger ticket items like the touchscreen, valve and flow sensor, we were willing to pay a premium for the quality and functionality they would provide. The final bill of materials with the costs of each part and where they were purchased can be found in **Appendix C**.

Safety Considerations

For our project there were a couple different safety considerations that we have to take into account. One of these considerations is that the keg containing the wine will be pressurized with up to 10 psi. However, the kegs that we are using are rated to withstand pressures of up to 150 psi. We believe that with a safety factor of 15 the pressurized keg should not be a concern. Another safety concern that we have to account for is that we are running 120V wall power into the 18 Channel Distributed Power Box. Due to this high amount of voltage we will be having the campus electrician verify the wiring before we plug it into the wall. However once this book is correctly wired the connections will be wrapped in heat shrink tubing to prevent anyone from making contact with it. This power box steps the voltage down to 12V for all of the other connections, so electrocution is not a concern. A formal version of the design hazard checklist can be found in Appendix A. There are no other major safety concerns that we had to consider.

Maintenance and Repair Considerations

When we were designing and developing our wine blending apparatus we had to take into consideration both the maintenance and repair of the device. Our design is intended to be used for long periods of time in a commercial environment. Therefore, the device must be simple to repair and maintain. One way in which we address the maintenance of the device will be by providing the sponsor with recommended cleaning regimen for the tubing, valve, and flow meter. This cleaning regimen will ensure that residue from the wine does not build up on the interior of any of the components. If the wine residue was allowed to build up, it could alter the accuracy of

the flow meter or prevent the valve from properly opening and closing. Another consideration that we took in regards to maintenance was through the use of quick connect/disconnect fittings. By making the fittings quick connect/disconnect, the tubing can be easily removed without the need for special tools. Another way in which we made our design easy to maintain is by using simple header interface so that a user can go in and adapt system settings without having to adjust the source code itself. One way in which we designed our device to be easier to repair was by using entirely off the shelf parts. We do not use a single custom component in our entire design. This means that if something breaks or malfunctions, a replacement part can easily be ordered.

Chapter 5: Product Realization

System Programming - Running the System

We did all of the programming for the system through Node-Red, a software suite that is editable online and makes microcontroller programming easy to read and debug. We run the Node-Red software off of the Raspberry Pi running the Raspbian operating system. Below is the general flow of our program beginning from receiving server input.

Turning on: The system turns on by flipping the switch on the power strip that the system is connected to.

Starting Node-Red: Node-Red starts by typing the command “node-red-start” in the operating system terminal.

Viewing Node-Red: Opening the web browser on the Raspberry Pi will take the user to the user interface of the system. Two separate tabs will open. The first tab will be the flow overview of the entire system The second tab shows the amounts poured and pour status of the current blend.
Troubleshooting: If Node-Red fails to start or the web interface fails to load, there are a few quick troubleshooting things that can be checked. The first is ensuring the Raspberry Pi is connected to the internet. This is done by looking at the top right of the screen at the WiFi symbol. If Node-Red still does not start, in the terminal type in “node-red-stop” and then “node-

red-start".

These steps as well as expanded upon troubleshooting tips can be found in **Appendix G**.

System Programming - Node Red Code Flow

Receiving Server Input: Once the server is ready for the machine to pour a blend, they will tap a button on the Node-Red user interface stating they are ready for a pour. Once our ultrasonic sensor registers that a glass or bottle has been placed properly, the device will pull a packet of data from the server. This packet contains information about the next customer in the queue and the desired pour amounts from each keg of wine. The programing flow for this can be seen below in **Figure 13**.

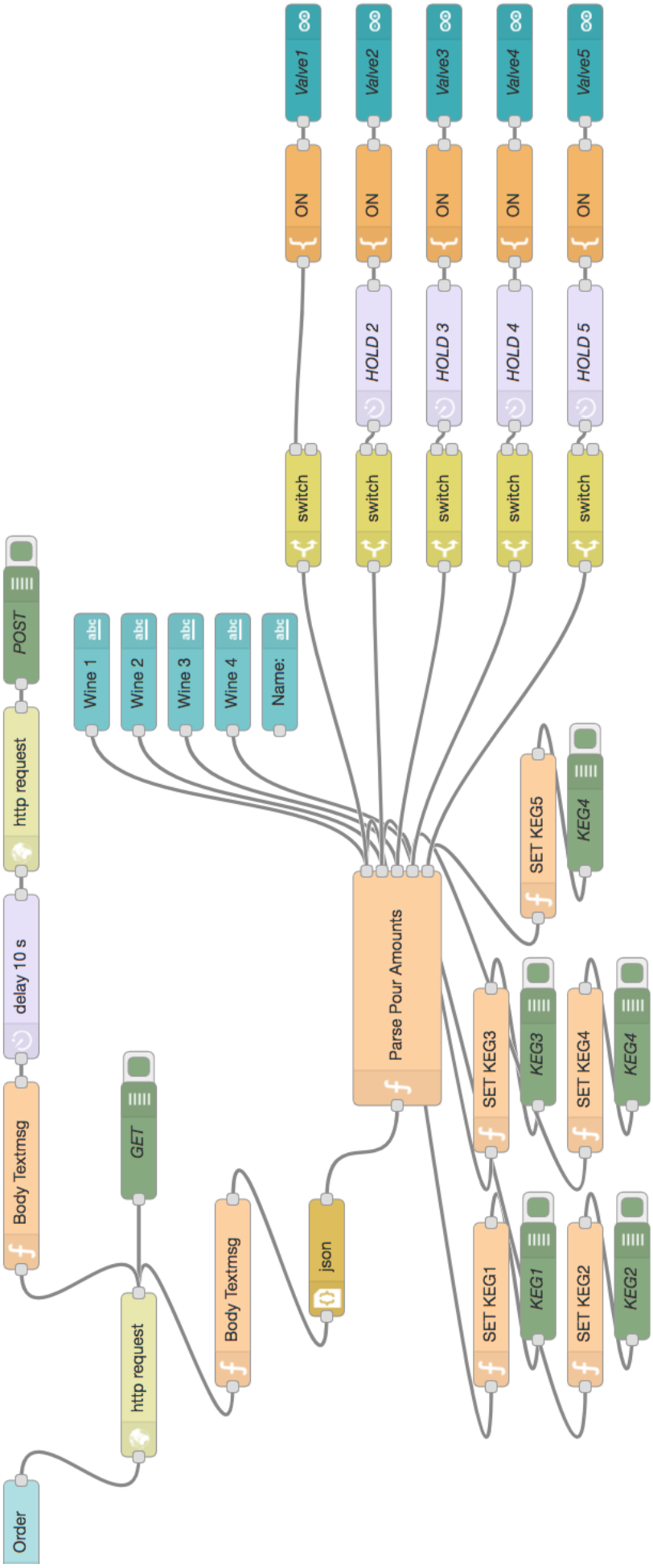


Figure 13: Server input code

Parsing Server Input: After receiving the packet of data from the server, the desired amounts are separated from the array and each keg is stored into variables that are later used in the flow meter code to tell our valves when to stop pouring. The code for this process is displayed in **Figure 14.**

```
pour1 = {};
pour2 = {};
pour3 = {};
pour4 = {};
pour5 = {};

pour1.payload=msg.payload.PourArrays[0][0];
pour2.payload=msg.payload.PourArrays[0][1];
pour3.payload=msg.payload.PourArrays[0][2];
pour4.payload=msg.payload.PourArrays[0][3];
pour5.payload=msg.payload.PourArrays[0][4];

return [pour1, pour2, pour3, pour4, pour5]
```

Figure 14: Server packet parse code

Pour Begins: After receiving all of the variables, the valves open and begin pouring wine into the glass. Pulses are taken from our flowmeters and then an integration is calculated to continually sum the volume of wine crossing the flow meter. The code for this calculation can been seen in **Figure 15 & 16**. We used a midpoint approximation integration for summing the amount poured from each keg, which proved to be very accurate at a small timestep. See **Figure 17** for an example of midpoint approximation integration.

```
var prev = context.get('prev')||0;
var current;
var output;
var timestep = 0.5;

current = msg.payload;
output = ((current + prev) / 2) * timestep;
prev=current;
context.set('prev', prev);
msg.payload = output / 60;

return msg;
```

Figure 15: Midpoint integration code

```
var pour1 = context.get('pour1')||0;
if (msg.payload == 10000) {
    pour1 = 0;
}
else {
    pour1+=msg.payload;
}
context.set('pour1', pour1);
msg.payload = pour1;

return msg;
```

Figure 16: Midpoint integration summation code

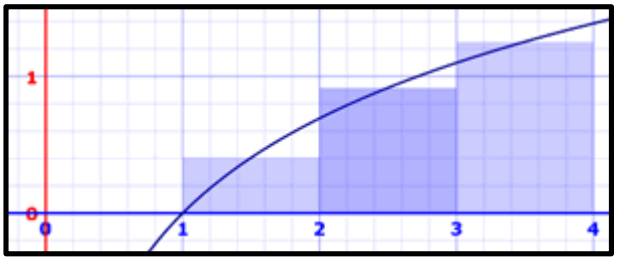


Figure 17: Midpoint Integration example

Pour Finishes: Once the valves are opened by the order code, it is the response of the sensor code that close the valves. The pulses from the sensors are collected, limited to the 1's, converted to pulse frequency, multiplied by the supplied K factor of 5.6 mL per pulse, integrate to find the volume poured. Once the desired amount of wine has been poured by each keg, the valve associated with that keg turns off. Once all of the valves are turned off, the variables keeping track of the total amounts poured from each keg are reset to 0 until the next server packet is received. See **Figure 18** below for a complete diagram of the pouring logic.

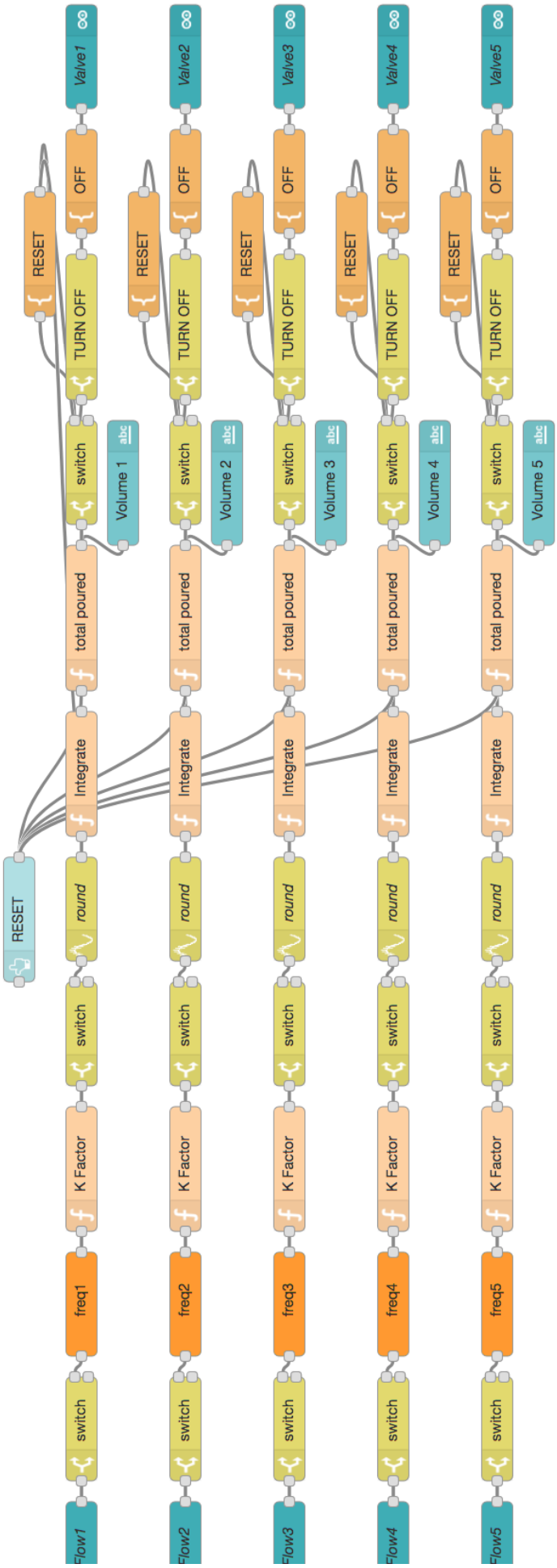


Figure 18: Diagram of pouring logic

Notifying the Server: After the pour is complete, the device sends a message back to the server notifying it of a complete pour and stating it is ready for the next pour in the queue. S**ee Figure 13** for a complete diagram of receiving and sending to the server.

Future Recommendations: During the setup of the system, we ran into a lot of issues regarding running two Arduino boards off of the Raspberry Pi board. Through research and troubleshooting, we found a way to map usb ports to specific hardware devices through Raspbian's device manager infrastructure called "udev". Creating a set of udev rules allowed us to connect to each board in Raspberry Pi regardless of which port they were assigned to by the operating system. However, Node-Red struggled to stay connected to both boards with any consistency during redeploys of the software. We ended up putting a 10 microfarad capacitor into the reset pin of one of the Arduino boards to prevent an auto-reset of the boards hardware during redeployment in Node-Red. This solution seemed to make connecting to the boards more consistent during testing.

Electrical Subsystem

Power Supply: The power supply that we purchased was modified to best fit the needs for our design. One modification that we made was to enlarge an opening on the side of the box to run the power cable from the supply to the power strip. We installed a cable clamp ring in this opening to ensure that the live wire could not be pulled loose from the box. Additionally, we added a small polycarbonate platform where we could mount the resistors for the valves. A picture of the interior of the power supply box is shown in **Figure 19.**

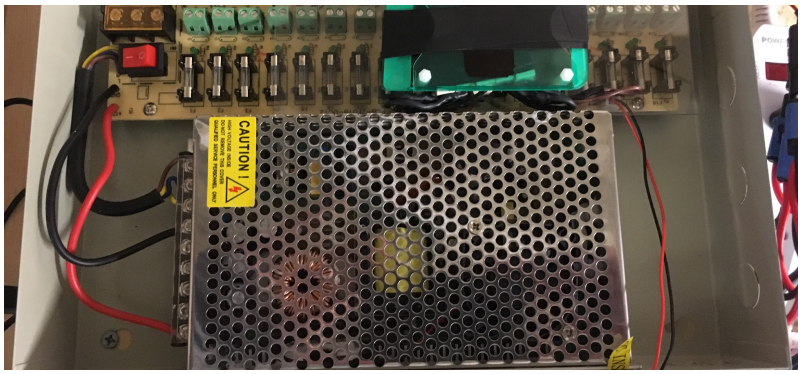


Figure 19: Power supply box

Valves: First we soldered and applied heat-shrink to the quick-disconnect adapters so the wires

for each valve were removable. We also soldered two 10Ω resistors between the power supply and the valves to lower the valves to their rated current and prevent overheating and improve power consumption. The valves were then wired to the relays which control the operation of the valves.

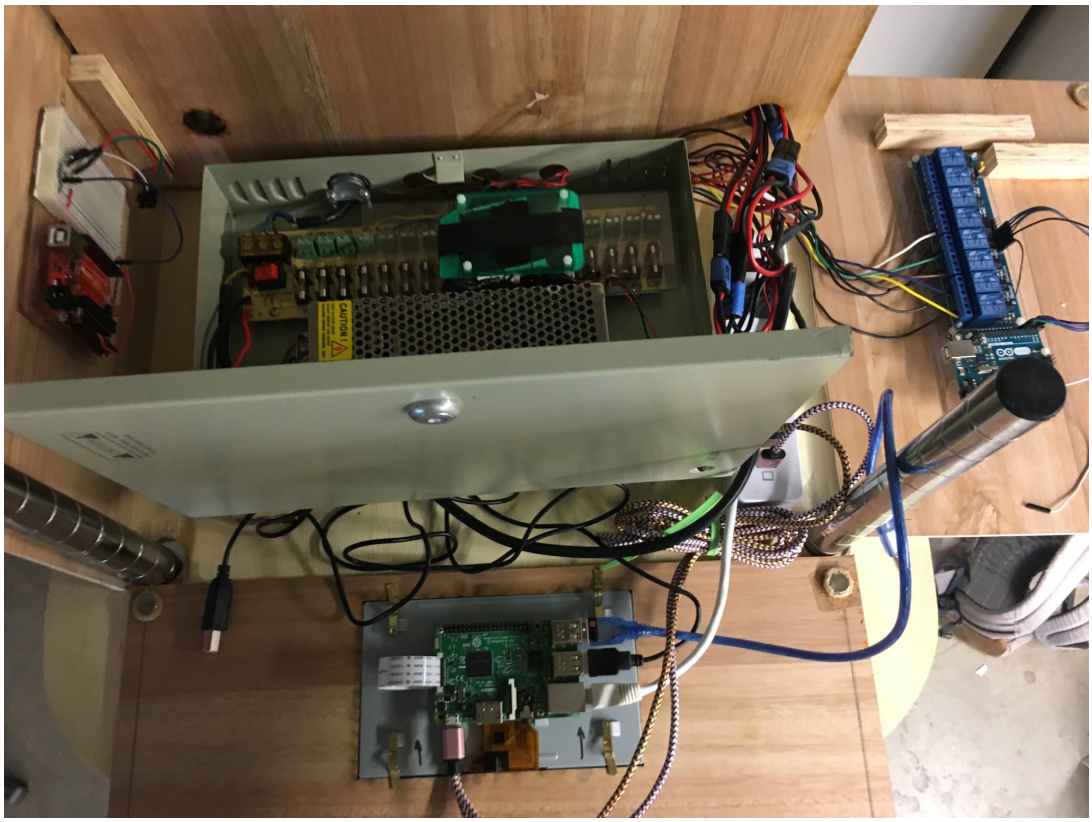


Figure 20: Electrical subsystem top view

*Relays:*The relays were wired to the valve control Arduino, which uses its 5 volt pins to open and close the electro-mechanical switches in each relay. Each relay is also wired to its own channel on the power supply box. The 8-channel relay board is powered from the Arduino the controls them. This relay board has red LED's on each channel to indicate the status of the relay (open/closed). The relays can be seen in blue on right hand panel in **Figure 20** above.

Flow meters: The Swissflow sf800 meters are three pin flow rate sensors. The voltage pin was wired directly to a breadboard with 5 volts from the Arduino receiving their input. The ground of the flow meter is also connected to the ground of the Arduino. Finally, the third pulse pin is wired in parallel with the voltage pin in a circuit on the breadboard. A diagram of this can be

seen below in **Figure 21**.

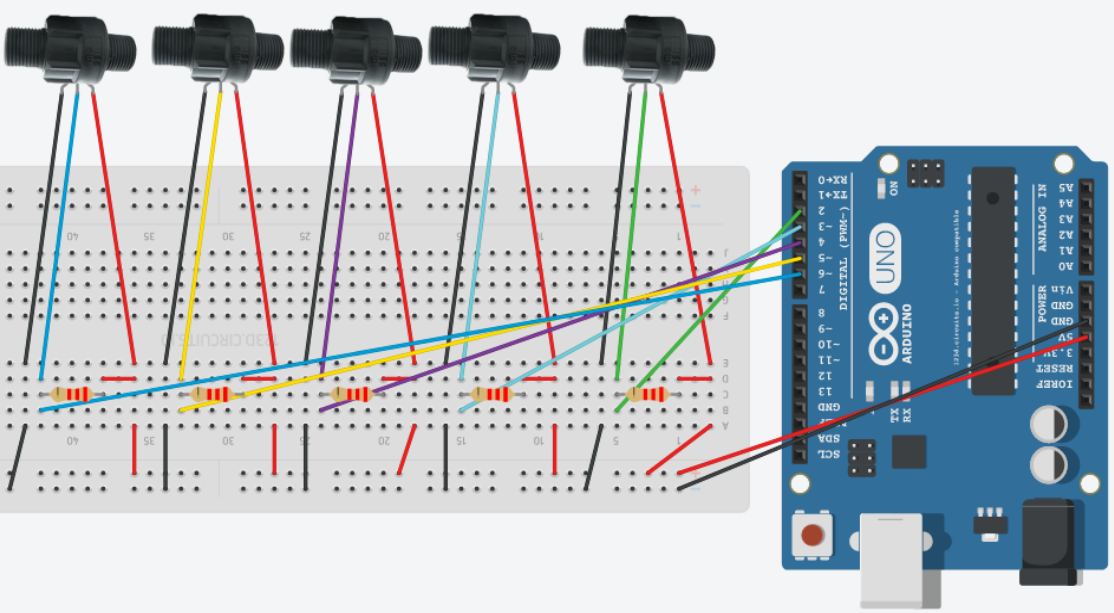


Figure 21: Wiring diagram for flowmeters.

Touchscreen: The touchscreen display, seen in **Figure 20**, was attached to the Raspberry Pi using the included ribbon cable. It is powered by a USB cable that is connected to the power strip.

Arduino Unos: The Arduino Unos are connected to the Raspberry Pi via a USB cable which supplies power and serial port communication. One of these Arduinos is used to power and receive flowmeter sensor data. The other board is used to control the valve switching via the relay.

Raspberry Pi: The Raspberry Pi is powered using a USB cable which plugs into the power strip.

Ultrasonic Range Sensor: The ultrasonic range sensor was glued to a 3" diameter wooden disk which was cut on the laser cutter. The ultrasonic sensor probes protrude from two holes in the wooden disk as seen in **Figure 22**. The ultrasonic sensor is wired to the Raspberry Pi for power and receiving the output. Two pins are for power and ground, while the other two are echo and

trigger. The trigger acts as a speaker that emits ultrasonic sound waves. Once these waves hit an object, they reflect and are detected by the echo probe, which acts as a microphone for ultrasonic waves.

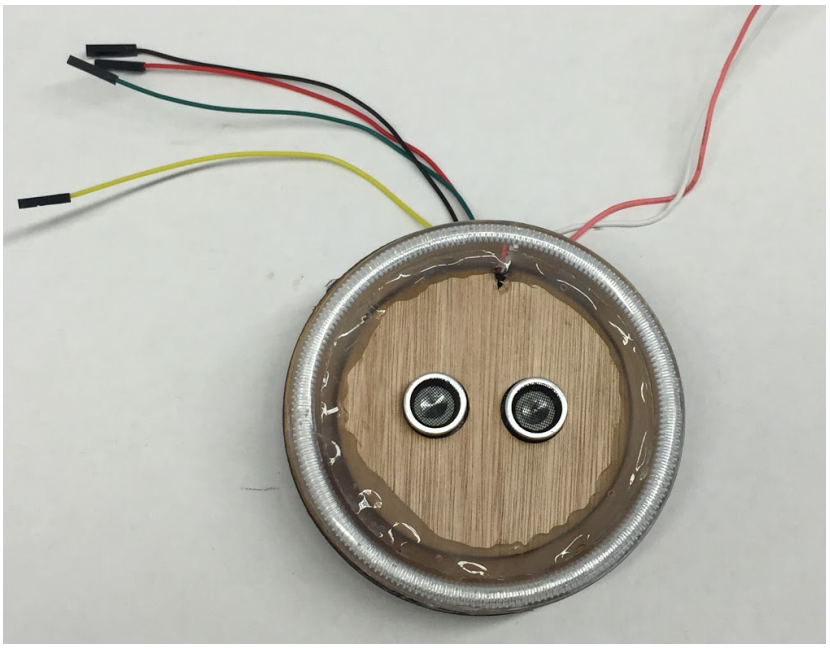


Figure 22: Wooden disk with ultrasonic sensor a LED ring.

LED Rings: There are two 3” diameter red LED rings in our system. The first LED ring is glued to the same wooden disk as the ultrasonic sensor. The second LED ring is glued to the wine bottle pouring spout. Both of these LEDs are wired to the relays and power supply box so that they can be programmatically controlled to turn on based on the output from our ultrasonic sensor**.**

Recommendations:

We recommend not using the Raspberry Pi GPIO pins for outputting to devices not already included the system. The Pi is running two other microcontrollers and running too many devices off of it can mess with incoming signals, leading to the device malfunctioning. If any wiring needs to be done to the electronics, ensure the device is turned off and unplugged completely. Due to the complex nature of the wiring, we recommend having an electrician or experienced operator perform any wiring task.

We are using an ultrasonic sensor to detect the presence of a vessel under the spout. The system

may not work properly if this malfunctions. This device was included to prevent the device from pouring wine when there was no glass or bottle underneath it. The red halo LED should light up when it detects a vessel. If it does not, try slightly moving the glass or bottle around in the enclosure until the sensor detects it.

The power supply box we are using seems to switch between voltages because of a malfunctioning voltage regulating screw. The device is very sensitive and seems to change when we move the entire system from being shaken around. While the device can still run if the correct voltage (VDC) is output to all channels, we recommend finding an alternative 10+ channel 12VDC power supply box that has a better voltage regulation system.

This device has pressurized liquids flowing very close to electronics. We installed a wet wall to prevent wine from getting into the electronics in the case of a leak. **Do not remove the wet wall that separates the valves from the electronics during operation. If the wet wall fails and wine gets into the electronics, immediately switch the device off by either hitting the switch on the power strip or unplugging the device from the wall.**

Tubing

The first part of assembling the tubing system was cutting our tubing to the calculated maximum length of 10 feet. The 5/16" ID tubing was then attached to the (black) liquid out hose barb on the keg as seen in **Figure 23**. We also attached a hose clamp to this fitting to ensure that it was airtight.

Figure 23: Keg with air and liquid lines mounted using hose barbs.

The other end of this tubing was connected to a brass 3/8” barbed to threaded 3/8” adapter. We also attached a hose clamp to this connection. The threads on this adapter were coated in 10 layers of white teflon tape to ensure a airtight seal with the flowmeter as seen in **Figure 24.** Before we attached the flowmeter to the adapter, we placed a 5/16” o-ring at the base of the opening. We attached another adapter to the other side of the flow meter and ran a piece of tubing between it and the valve.

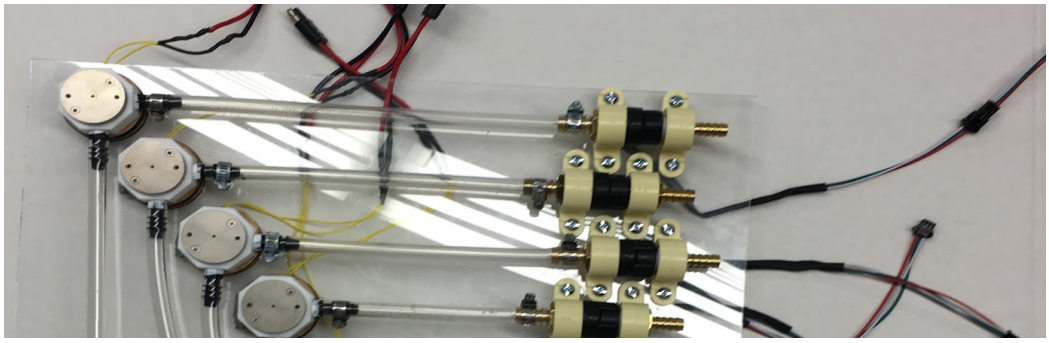


Figure 24: Valve and flow meter connections

The tubing was attached to the inlet of the flow meter with a threaded hose barb adapter. The final length of tubing runs from the outlet of the valve into our wine bottle pouring spout seen in **Figure 25**.

Figure 25: Wine bottle pouring spout

The wine bottle pouring spout consists of the top half of a wine bottle glued to a metal ring covered in neodymium magnets. These magnets allow the bottle to be attached to the sheet of metal mounted on the back of the enclosure. There is also a ring of LED lights glued to the metal ring which turn on when the wine is being poured.

Recommendations: There is one change that we would make to our tubing system if we were to do it again. One change we would make is to find a correctly threaded barbed adapters to attach to the flow meters. The flow meter itself is british and uses BSP threads which are very difficult to find fittings for here in the United States. We were forced to use an adapter with NPT threads. This made it difficult to make a perfectly watertight seal between each component. We would recommended that proper BSP threaded barbed adapters be located and used instead.

Frame

The first step in constructing the frame for our system was to assemble the metal shelving subframe seen in **Figure 26**.

Figure 26: Metal shelving subframe.

The next step was cutting out the two wooden platforms which would serve as the bases for each each level of the system. As seen in **Figure 27** the bottom level supports the wine glass, while the top level supports the electromechanical components and the upper control housing.

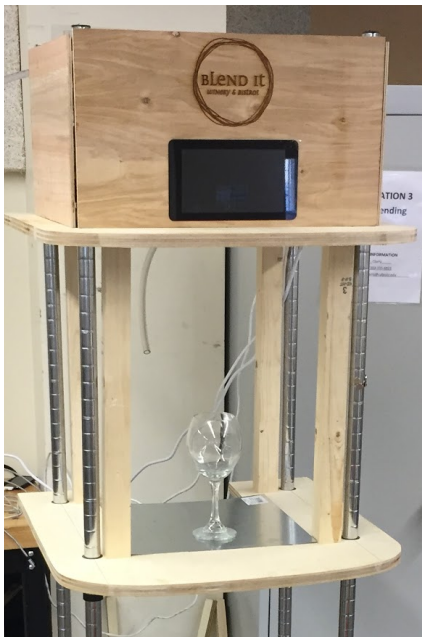


Figure 27: Frame of our system

The platforms were cut into 22x22” squares using the table saw seen in **Figure 28**. After the squares had been cut, we used a bandsaw to cut out the curved front edge. The edges were cleaned up using the belt sander shown in **Figure 29,** to ensure that they were properly shaped.

Figure 28: Belt sanding platform edge

Figure 29: Table saw platforms to size

The four wooden support posts were manufactured next. These were cut down to 18” long using the chop saw. The final part of the frame is the upper control housing for the electromechanical components. This upper control housing consists of four individually cut sides. Each side is a piece of ¼” plywood which was cut out using the laser cutter shown in **Figure 30**. This laser cutter was also used to etch the Blend-It logo into the front panel of the housing.

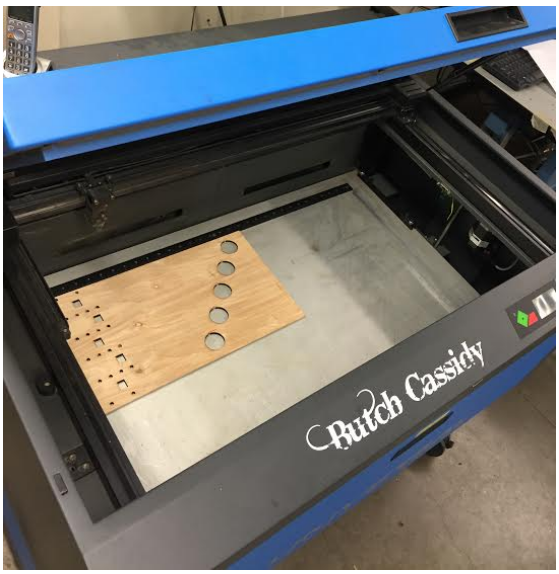


Figure 30: Laser cutter used for fabrication

After each side of the control housing had been laser cut to size, we used construction adhesive to attach a neodymium magnet to the inside corners of each panel. The magnet allow the panels to attach to the metal posts on the subframe.We also used construction adhesive to attach the relays and the arduinos to the inside of the side panels. As seen in **Figure 31** the relays and valve arduino are mounted on the right side panel, while the flow meter breadboard and arduino are mounted on the left side panel.

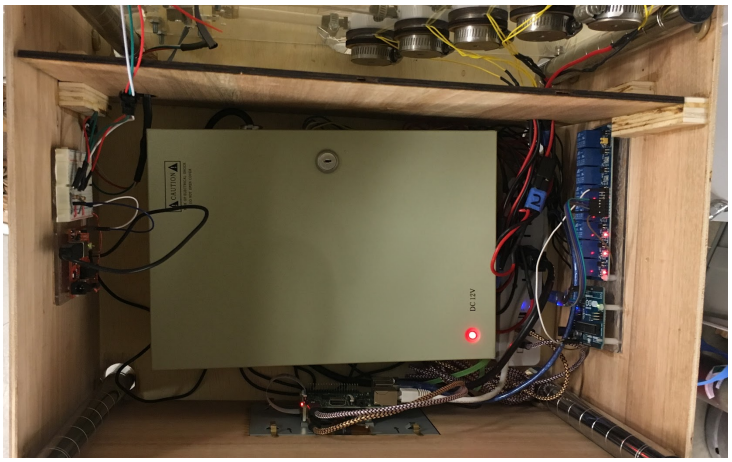


Figure 31: Top view of upper control housing

The touchscreen display was mounted in the laser cut hole on the front panel and held in place by four brass tabs as seen in **Figure 32**.

Figure 32: Close up view of touchscreen mounted on front panel

The final component of the frame that we manufactured was the mounting panel for the valves and the flow meters. This panel was made out of ¼" laser cut acrylic. In order to properly mount the valves in the holes, we had to fasten them on the backside using hose clamps. The flow meters were mounted to the panel using plastic brackets and steel bolts as seen in **Figure 33**.

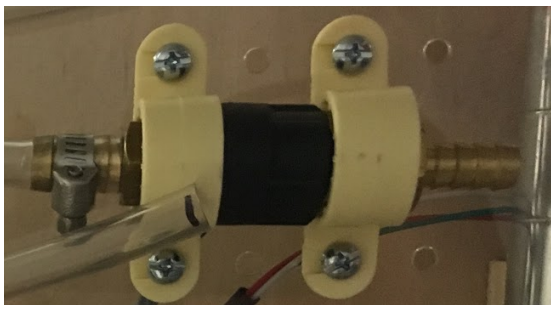


Figure 33: Flow meter mounted to acrylic panel

Recommendations: The only recommendation that we have for the future manufacturing of the frame, would be to cut the top and bottom platforms using a laser cutter. The laser cutter we had access to was not large enough to accommodate the pieces of wood. If a laser cutter was used it would substantially decrease the amount of manufacturing time required for each piece.

Enclosure

The enclosure for our system was the final step in our manufacturing process. The first part of the enclosure that we manufactured was the insert which nestles between the four wooden support posts of the frame. The insert consisted of various wooden and acrylic panels which were cut out using the laser cutter. As seen in **Figure 34** the panels of the insert were all notched, which allowed them to fit together similar to puzzle pieces.

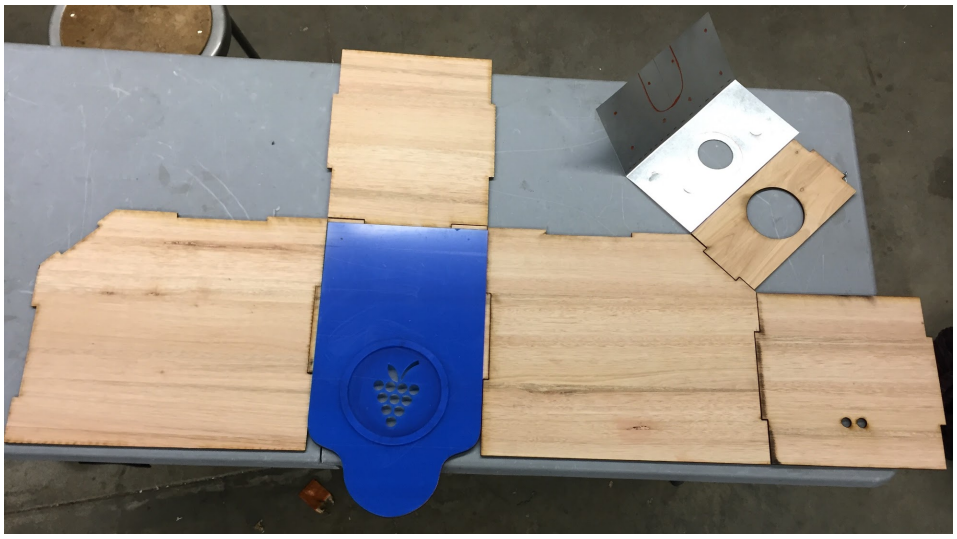


Figure 34: All of the panels for the enclosure insert laid out prior to assembly

After we finished cutting out all of the panels for the insert, we assembled them and glued them together using construction adhesive. The completely assembled insert can be seen below in **Figure 35**.

Figure 35: Front and rear views of fully assembled enclosure insert

The next step in the enclosure manufacturing process was to laser cut and then mount the front display panels. The front display panels were laser cut and then mounted to each the front two support columns. Before we mounted the panels to the columns we checked them for fitment and then spray painted them black. In **Figure 36** the display panels can be seen in their mounting location during the fitment test.

Figure 36: Front panel mounting location prior to painting

After we installed the front panels, we began manufacturing the shell of our enclosure. The shell of our enclosure is made of Polyboard plastic sheeting. We first spray painted the Polyboard Cranberry Red and allowed it to dry. After we finished painting, we mounted the Polyboard to the frame of our device. The Polyboard was mounted to the exterior edges of the top and bottom platforms using 30lb 3M Foam Mounting Tape. **Figure 37** shows the strips of mounting tape along the edge of the platforms.

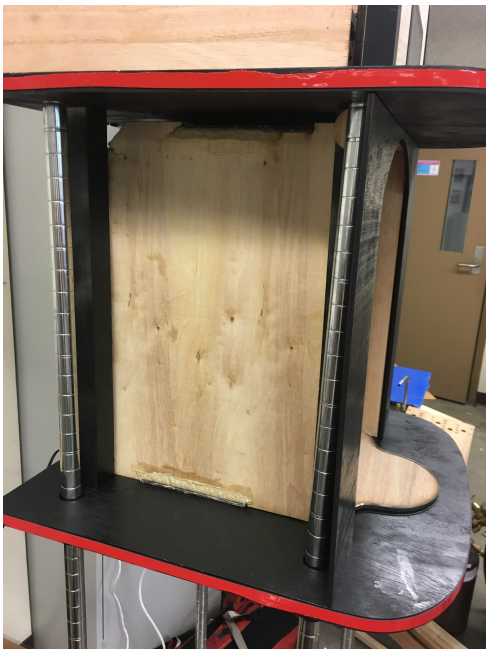


Figure 37: Mounting tape along the edges of the top and bottom platform

Once the tape was applied to the platforms, we attached the Polyboard to the frame. After the Polyboard was secured to the frame we installed wooden cross-members on the metal subframe. These wooden cross-members were attached to the inside edges of the shell to ensure that it maintained the proper shape. **Figure 38** shows how the wooden cross-members are installed on the subframe, and how they are attached to the inside of the shell.

Figure 38: Wooden cross-members mounted to subframe

After the shell was fully secured we cut out the opening on the front using an Xacto knife. Once the opening was cut out we mounted aluminum corner brackets to the edges of the front panel, and to the edges of the upper control box. These pieces were cut to length using the table and mounted to the enclosure using mounting tape. **Figure 39** shows our system once the enclosure had manufactured and assembled.

Figure 39: Device with fully assembled enclosure

Recommendations: For future manufacturing we would recommend that the shell of the enclosure be fabricated out of powder coated sheet metal. The sheet metal would have a far more professional appearance, as well as being more rigid and durable. We would also recommend that the inside of the enclosure insert be painted with a coat of polyurethane. The polyurethane would prevent spilled wine from staining any of the wood panels.

Chapter 6: Design Verification (Testing)

Our project testing can be broken down into 3 subsections and full system testing. Full system testing will not begin until each subsystem has adequately passed tests to our project specifications. The three subsystems include the mechanical, electrical, and software aspects of the system. Each of these subsystems has their own testing criteria and specifications they need to meet for our project to have full functionality.

Mechanical Component Testing

There are a variety of different mechanical components that play a role in our design. The intent for our device is that it will be used by our sponsor in the Blend-it winery and Bistro. Therefore we had to verify that all of the components were performing exactly as desired.

Valve Control Test

The Clippard valve needs to be controllable through the use of the Raspberry Pi. We will verify this by sending the valve a command to open and then close using a Raspberry Pi. We will have the valve inlet line pressurized with water at 10 psi. If the test is successful we will see the water begin to flow out of the valve when the open command is sent, and we will see the water stop when the close command is set.

Flow Meter Calibration

The accuracy of our critical to the ability of our device to pour precisely the right amount of wine. We need to verify that the flow rate of the liquid coming out of the valve is being correctly measure by our flow meter. We will test this by placing graduated cylinder underneath the valve and then turning it on for five seconds. We will then divide the volume of liquid in the cylinder by five to find the flow rate. We will compare this flowrate to the value recorded by flow meter to ensure that they are the same.

Electrical Component Testing

Many of the components in our system directly or indirectly receive power either from a standard wall outlet, usb power, or from another component in the system. Along with individual component testing to ensure accuracy, all electrical components will be tested to ensure compatibility. By sequentially adding components to the system, we can validate every part of

the system and troubleshoot problems that arise in an intelligent manner. Sequentially adding and testing components will make our testing process faster, allowing us to fine tune our system to make it as accurate and precise as possible. This will also allow us to create a broader group of tests to create a robust prototype design.

Arduino Uno

The Arduino Uno will be used to power the 8-channel relay and control the relay switching. We have already tested the Arduino's ability to control the relay switches effectively. The Arduino will also interface with the Raspberry Pi 3 microcontroller. To test this, we will set up a serial communication port between the microcontrollers and send data from the Raspberry Pi to the serial port. This device intercommunication will be proven if the Arduino can receive all of the information sent from the Raspberry Pi 3. The Arduino may also be used to power our flow-meters. We will test the flow-meters being powered off of the Arduino and compare the accuracy between the Arduino, Raspberry Pi 3, and 18-channel power supplies.

Raspberry Pi 3

The Raspberry Pi 3 microcontroller will be used primarily to power the flow-meters and interface with the server being developed by a separate senior project team. The flow-meters will be powered from the Raspberry Pi output pins and the accuracy of the data received back will be compared against alternate power sources. The interface between the Raspberry Pi and the Arduino will primarily be coded and integrated into this board. This board will also control the LCD screen that the user will be able to interface.

8-Channel Relay

The relay is used as a switch for all five of our electronically controlled valve. We have already tested the relays ability to switch while interfaced with the Arduino Uno microcontroller. The relay was very responsive and was not as loud as we initially expected. The relay was continually tested passively as we ran the device. Each relay in was always responsive without signs of slowing down or changing switching speeds.

18-Channel Power Distribution Panel

To test our distribution panel, we will power the panel from a standard wall outlet and hook up a single line of our system to one of the channels. We have proven that the distribution panel provides ~12V to all devices connected to it. There is a voltage regulation screw that we noticed at times gets bumped, lowering the voltage to all 18 channels. By manually changing the position of the voltage regulation screw inside the box and measuring DCV out on a channel, we can fine tune the box to provide the required 12V. This can be measured by using a multimeter. During testing, our measured DVC output from each channel ranged from ~12V to ~14V as the regulation screw was very reactive to small changes. Voltages in this range did not seem to negatively affect any of our connect devices.

Specification Verification Checklist

We will perform the following tests to confirm that our design successfully meets each of our specifications:

1. Pressure: The entire system will be pressurized to 10 psi.
   To ensure that the system can withstand 10 psi, we will fill the system with a dark colored liquid and then pressurize the system to 10 psi. We will then check for any leaks or air bubbles within the line. This ensures that there is no oxygen within the line. If any leaks are identified, then steps will be taken to prevent them.
2. Flow Rate: The system must be have a flow rate of at least 12.5 mL/s.
   To verify that the system has a flow rate of at least 12.5 mL/s, we will place a graduated cylinder beneath the valve outlet and open it for 4 seconds. If there is not at least 50 mL of water in the cylinder, we will have to shorten the length of the length of the tubing.
3. Tubing Inner Diameter: The tubing must have an inner diameter of 5/16".
   We will measure the inside diameter of the tubing with calipers to ensure that it is 5/16". If the tubing is not 5/16" we will have to acquire new tubing.
4. Material: All material that comes in contact with wine must be wine-safe.
   All of the components that we selected for our design are made out of material that will not affect the wine.

5. Disassembly: Must be quick and easy to disassemble.
   None of the mechanical components will require special tools to remove. We will have three individuals who are not a member of our team try to disassemble the device using only the instruction manual. If they struggle with the process we will look to simplify the design.
6. Power Consumption: The max power consumption will be 12 volts.
   We will verify that none of the components or peripherals require more than 12 volts. If the require more than 12 volts we will have to select another product.
7. Electrical Safety: No shock hazards.
   We will verify that there is no exposed wiring that could be touched by an individual. We will also make that there is no possibility of wine coming into contact with electrical components.
8. Organized Code: Easy to read and understand.
   The code make sure to clearly and thoroughly comment on all of our code. We will then have a person who is unfamiliar with coding try to read and understand it. If they have difficulty with it, we will ask them we can improve it.
9. Cost: Total cost of single line system is less than $500.
   After we finish the project, we will calculate an overall project cost and present it to our sponsor.

Chapter 7: Conclusions and Recommendations

Conclusion

Overall this project was a tremendous accomplishment. We feel as if we tackled two or three senior projects tied into one, and finished with a product that is both functional and aesthetically pleasing. There were, however, some things we would improve given more time. We had some trouble getting both Arduinos to simultaneously connect to the Raspberry Pi. There was not any simple way to view which port the operating system assigned to the board. Since Node-Red uses this port, code has to be changed if they switched for any reason. We recommend creating "udev" rules to symbolically link each specific Arduino board to the port assigned by the operating system. By typing the command "ls -l /dev/valves" and "ls -l /dev/flowmeters" into the terminal, it is easy to see which port each board is connected to. Also, during the Raspberry Pi startup and upon redeploy in Node-Red, the Pi supplies voltage to the relay and therefore the power supply box. This means that the LED lights and valves will turn on during the computer startup and sometimes during the redeploy of code in Node-Red. Our solution to this was to unplug the power supply box from the power strip during these times to mitigate the risk of accidental excitation voltage causing the valves to turn on without our control. If the lines are pressurized and the box is not unplugged, the valves will open without user control to close them, causing fluid to spill at a rapid rate. We think that further improvements to the code could allow for the Raspberry Pi to not supply voltage to the relay at these times and would solve this problem. If this code is not feasible, a power switch mounted to the front panel could be added to effectively shut off valve power as we are currently doing by unplugging the box. Also, the code was limiting us to pour each wine separately with a delay between each. We think that with more testing we could achieve simultaneous pours of all five wines, making the process much faster. If this was not feasible, then it would be more realistic to add code that starts the next pour after the last was finished, eliminating the delay between wines and again making the process much faster.

References

[1] Nuss, Holly. "Wine Market Council Stands by Their 2016 Consumer Research on Millennial Wine Consumption Habits." (n.d.): n. pag. Wine Market Council, 24 Mar. 2016. Web. 20 Oct. 2016.

[2] Moezidis, Nick, and Edward A. Vetter. Wine Blending System and Method. Napa Technology, assignee. Patent US20150336784. 26 Nov. 2015. Print.

[3] "COCA COLA FREESTYLE DISPENSER." FCC Applications FCC ID. The Coca-Cola Company, 25 Oct. 2015. Web. 26 Oct. 2016. <https://fccid.io/document.php?id=1255893>.

[4] Milhorn, Kirsten Elizabeth. Color Dispensing System and Method. Kirsten Elizabeth Milhorn, assignee. Patent US8666540 B2. 4 Mar. 2014. Print.

[5] Brita Hydration Station with Lifecycle Control (n.d.): n. pag. MODEL 2000S. Brita, 1 Feb. 2012. Web. 20 Oct. 2016. <https://www.hawsco.com/downloads/dl/file/id/14346/2000s_specsheet.pdf>.

[6] "PRODUCT PAGE." The Automated Bartender Professional Quality Cocktails in Seconds. MONSIEUR, n.d. Web. 26 Oct. 2016. <http://monsieur.co/product>.

[7] Marvin, Rob. "NASA's 10 Rules for Developing Safety-critical Code." Software Development Times. 8 Jan. 2015. Web. 15 Nov. 2016.

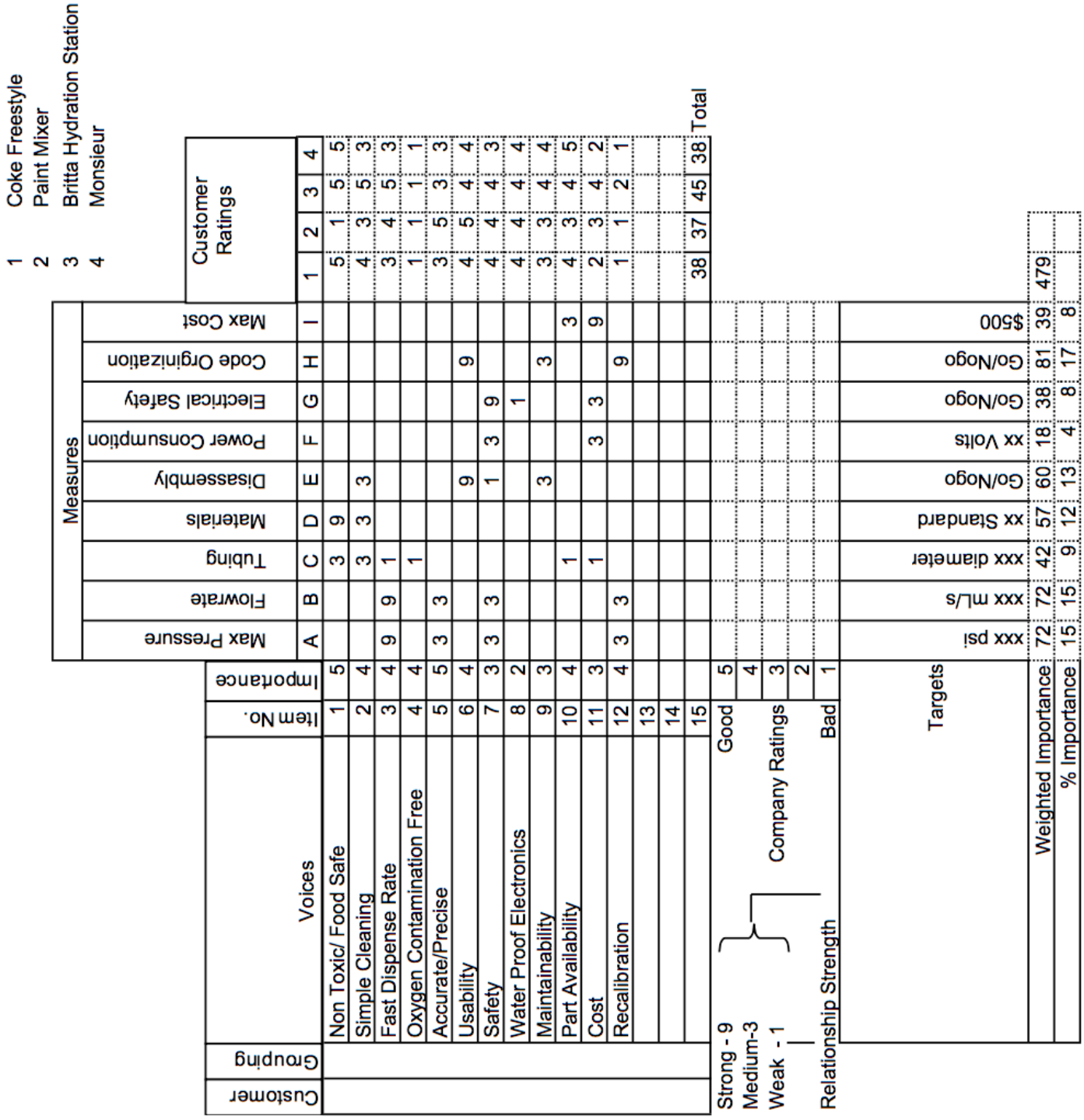
[8] Chris. "Determining the Right Pressure for Your Draft Beer System - KegWorks Blog." KegWorks Blog. N.p., 23 May 2016. Web. 4 Nov. 2016.

[9] Smith, Brad. "Keg Line Length Balancing – The Science of Draft Beer." *BeerSmith*

Home Brewing Blog. N.p., 14 July 2011. Web. 5 Nov. 2016. <http://beersmith.com/blog/2011/07/14/keg-line-length-balancing-the-science-of-draft-beer/>

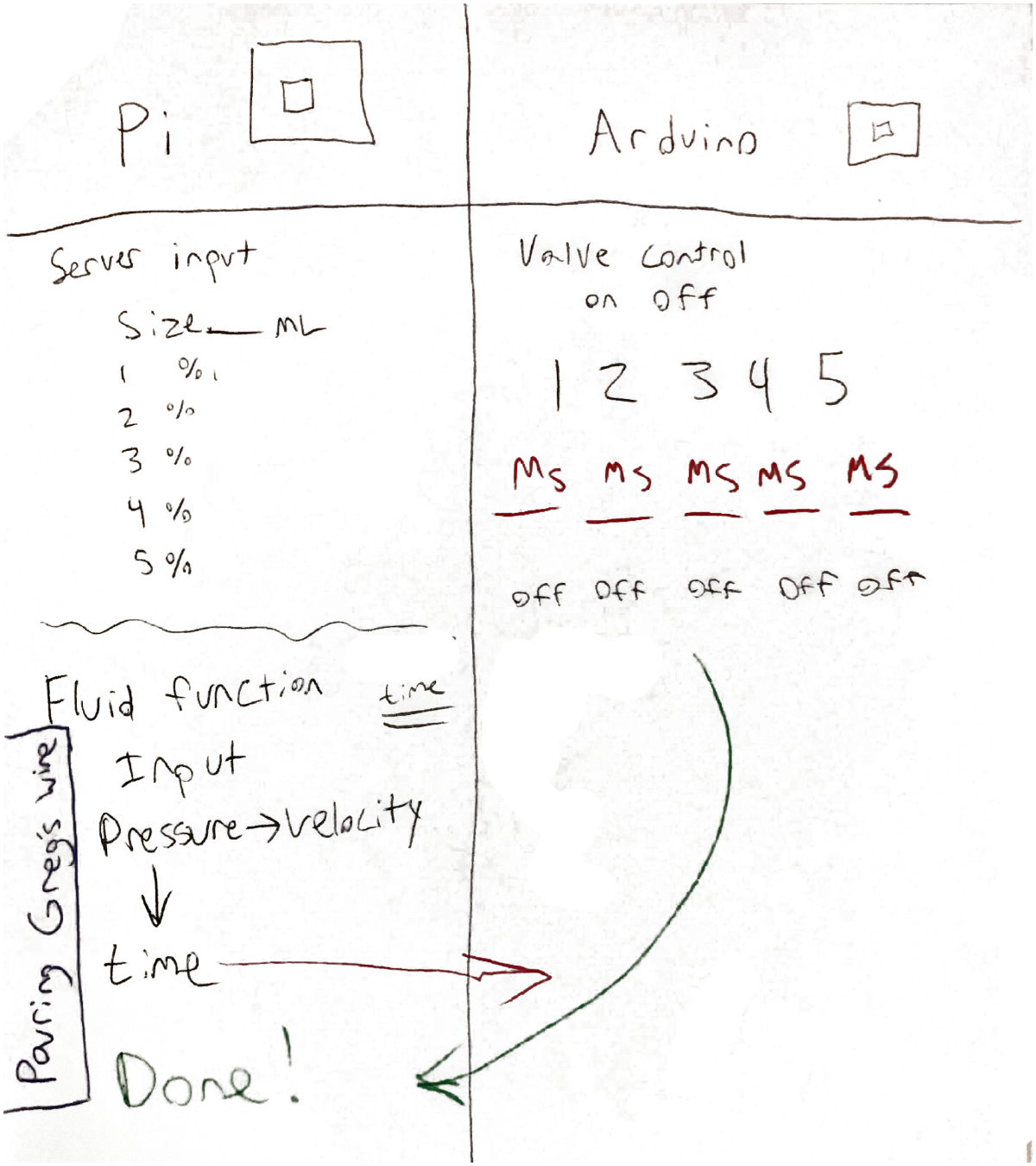
Appendices

Appendix A: Quality Function Deployment and Ideation

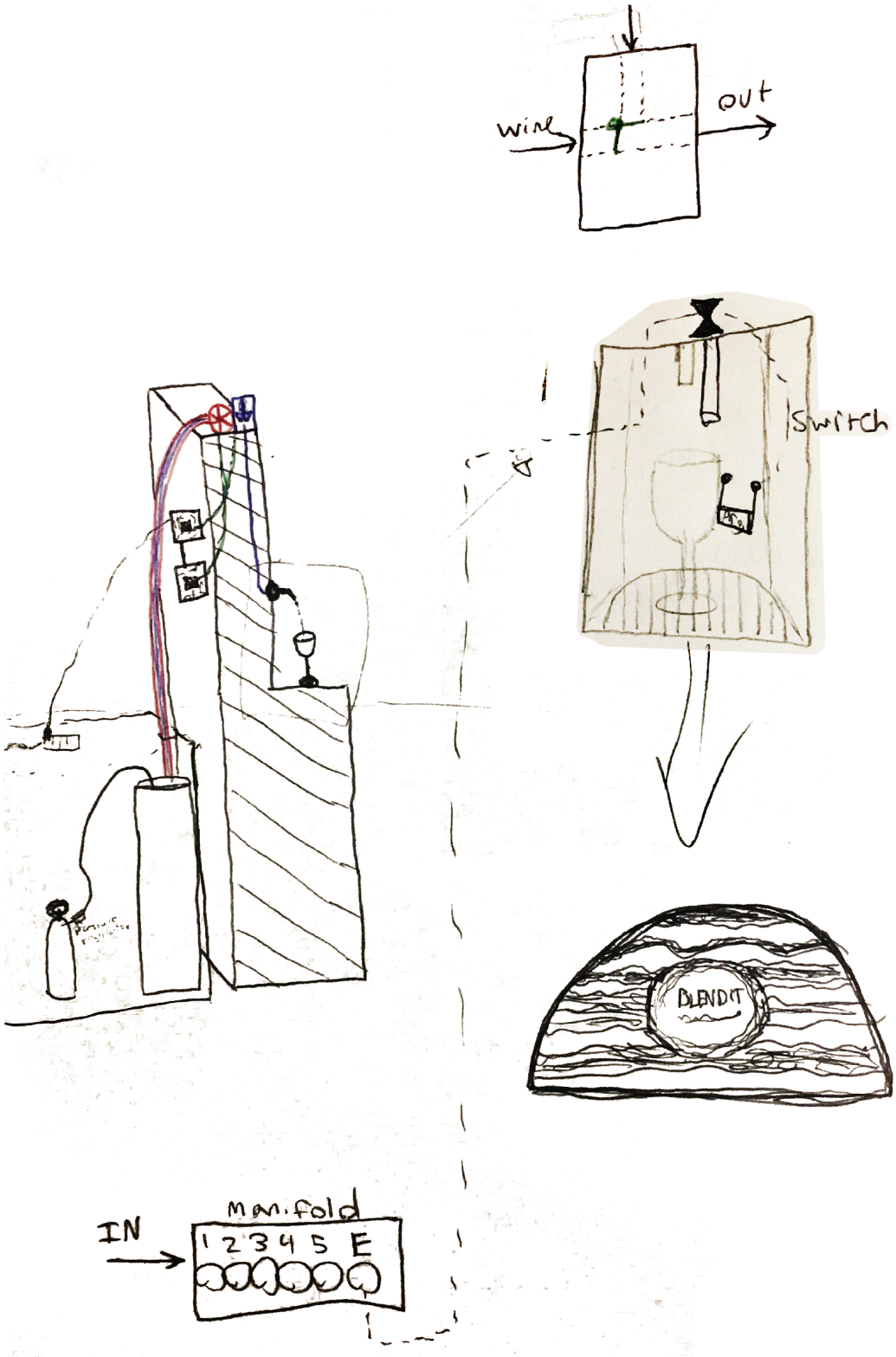


Ideation Sessions

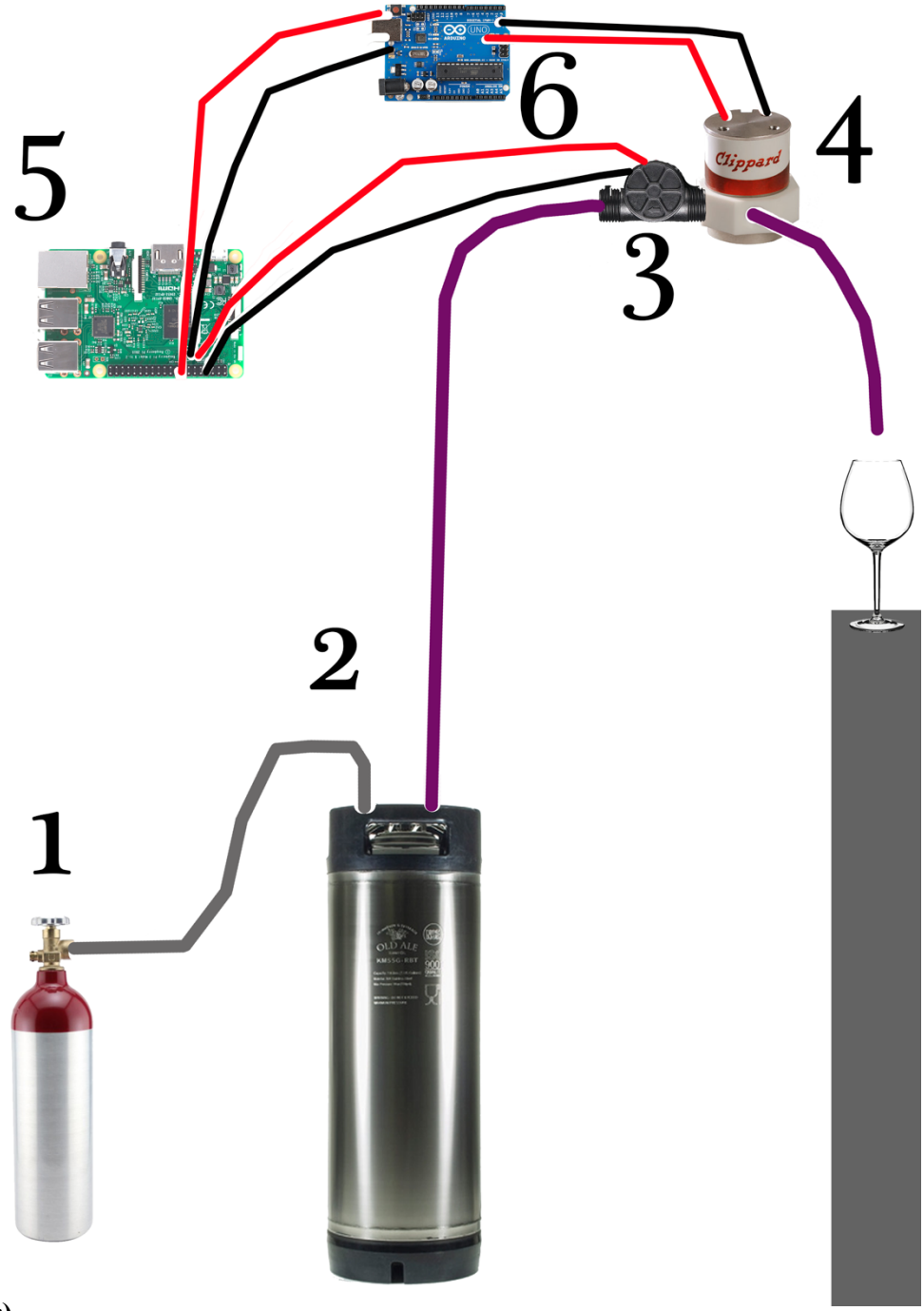
Microcontroller Code Ideation



Device Brainsketching



Test Fixture Boundary Diagram



KEY

Test Fixture

1. Inert Gas Canister
2. Cornelius Keg
3. Flow Rate Sensor
4. Clippard Valve
5. Raspberry Pi
6. Arduino

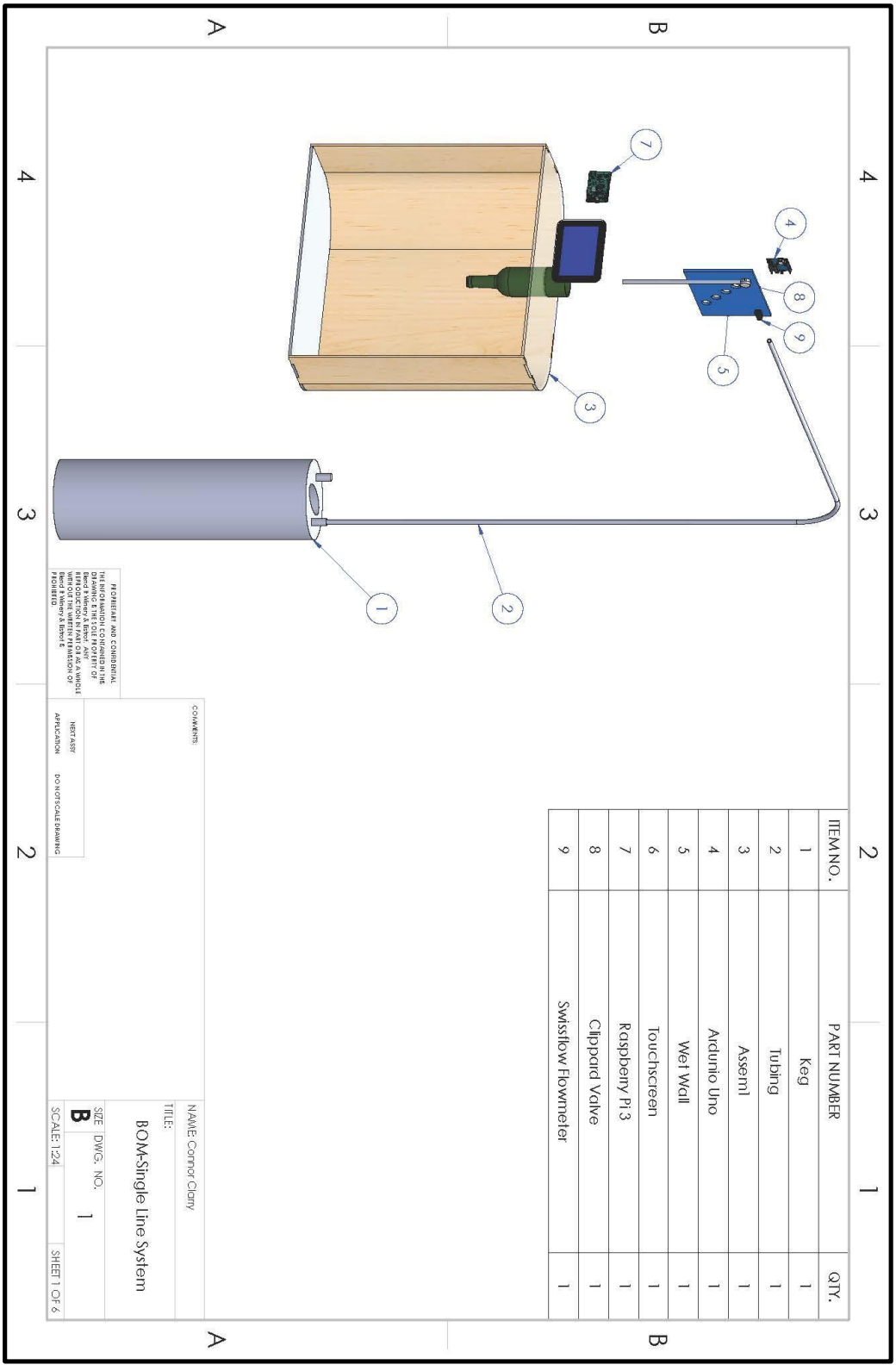
Line type

- **Purple: Wine (Water)**
- **Grey Inert Gas (Air)**
- **Red/Black: Electrical**

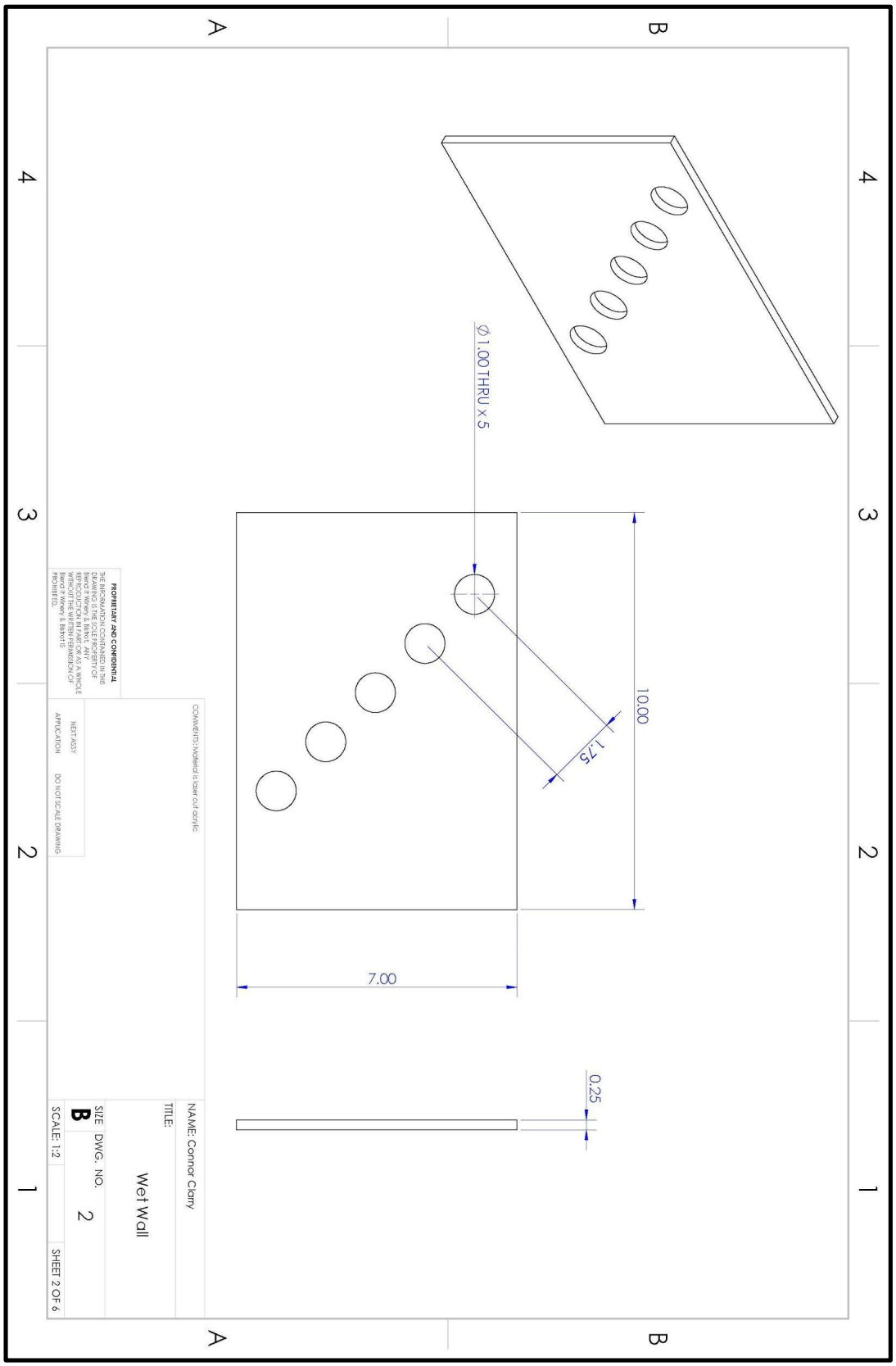
Concept Modeling

Appendix B: Drawing Packet

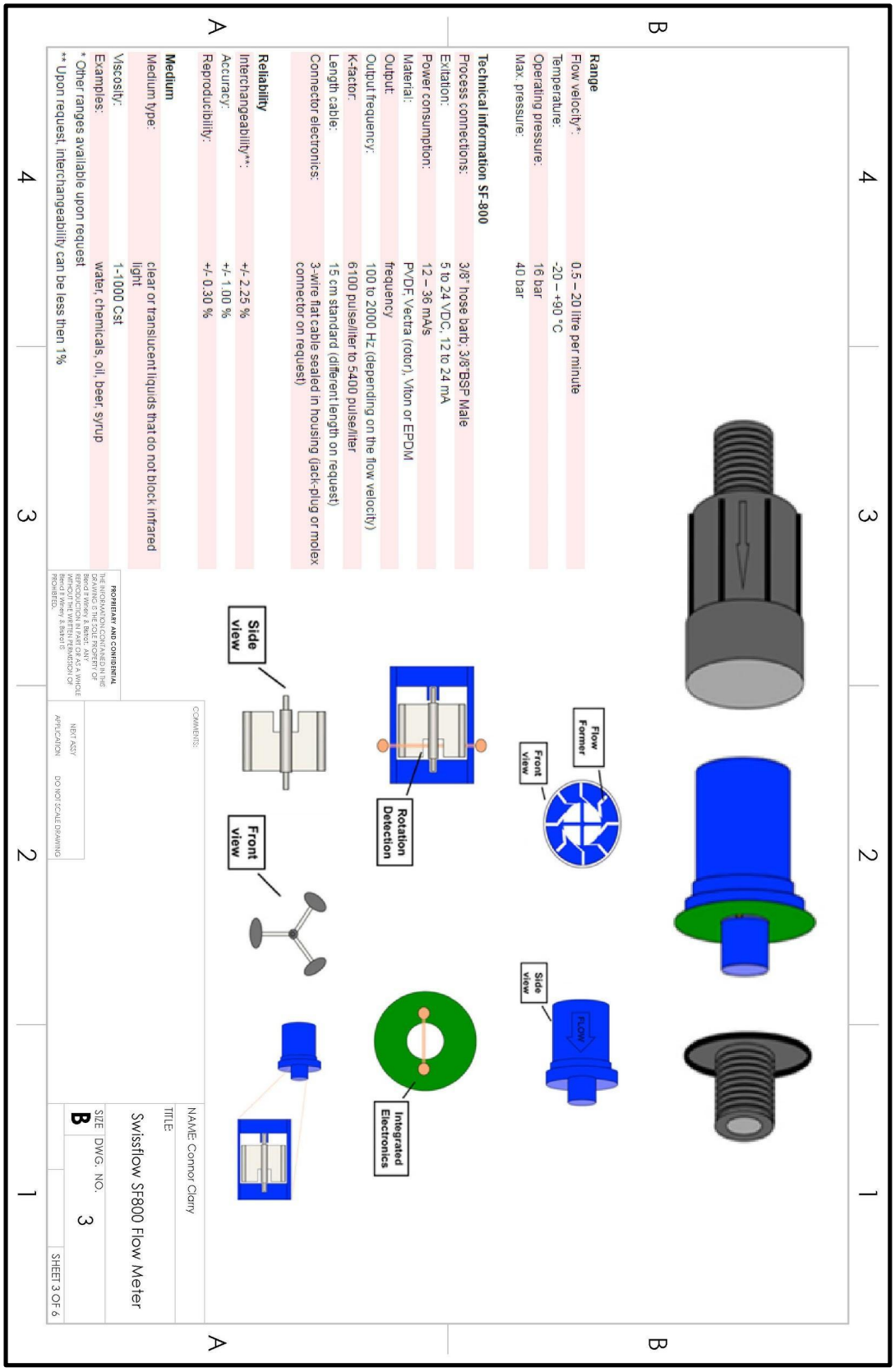
Single Line System Drawing



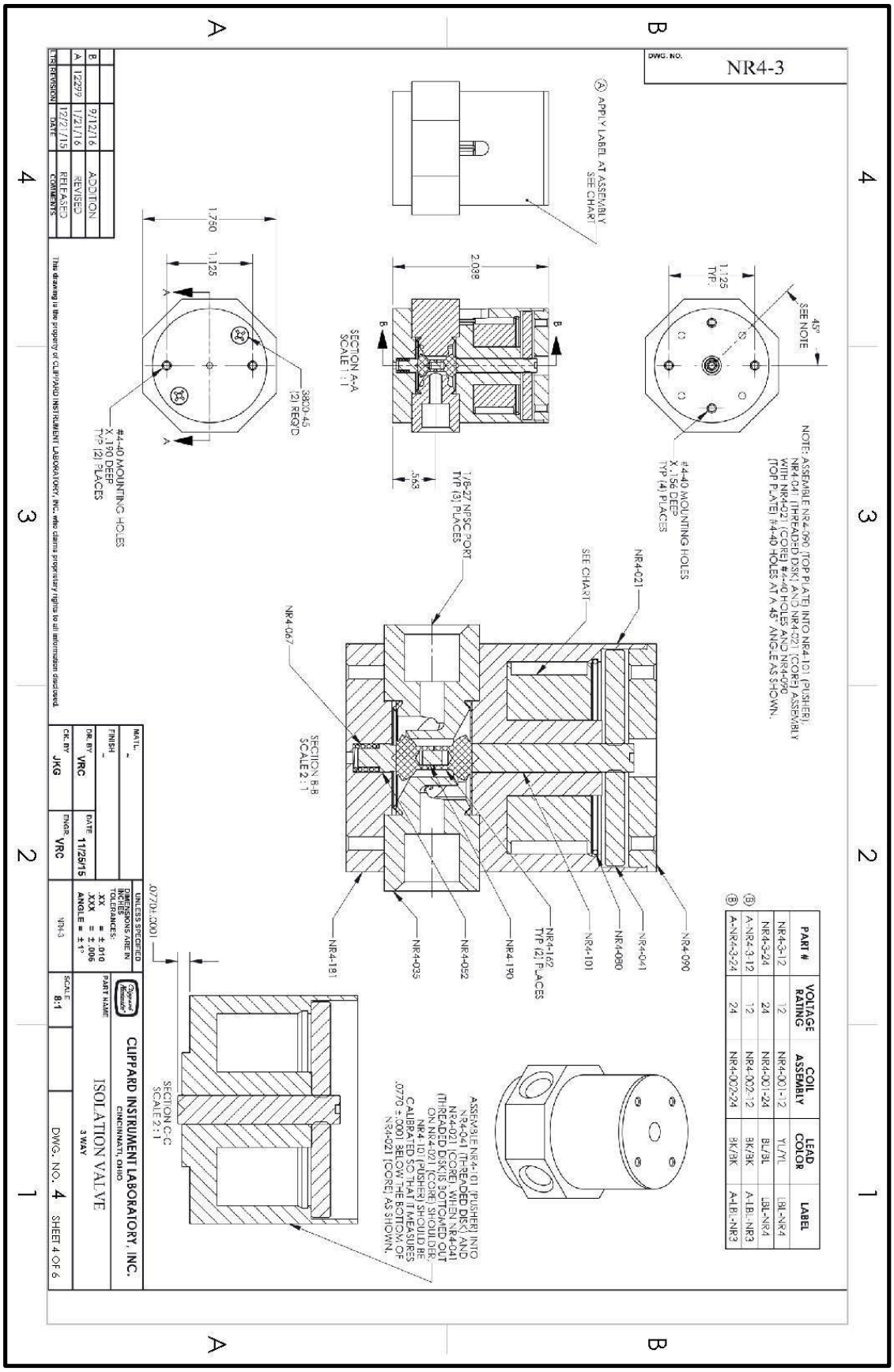
Wet Wall Drawing



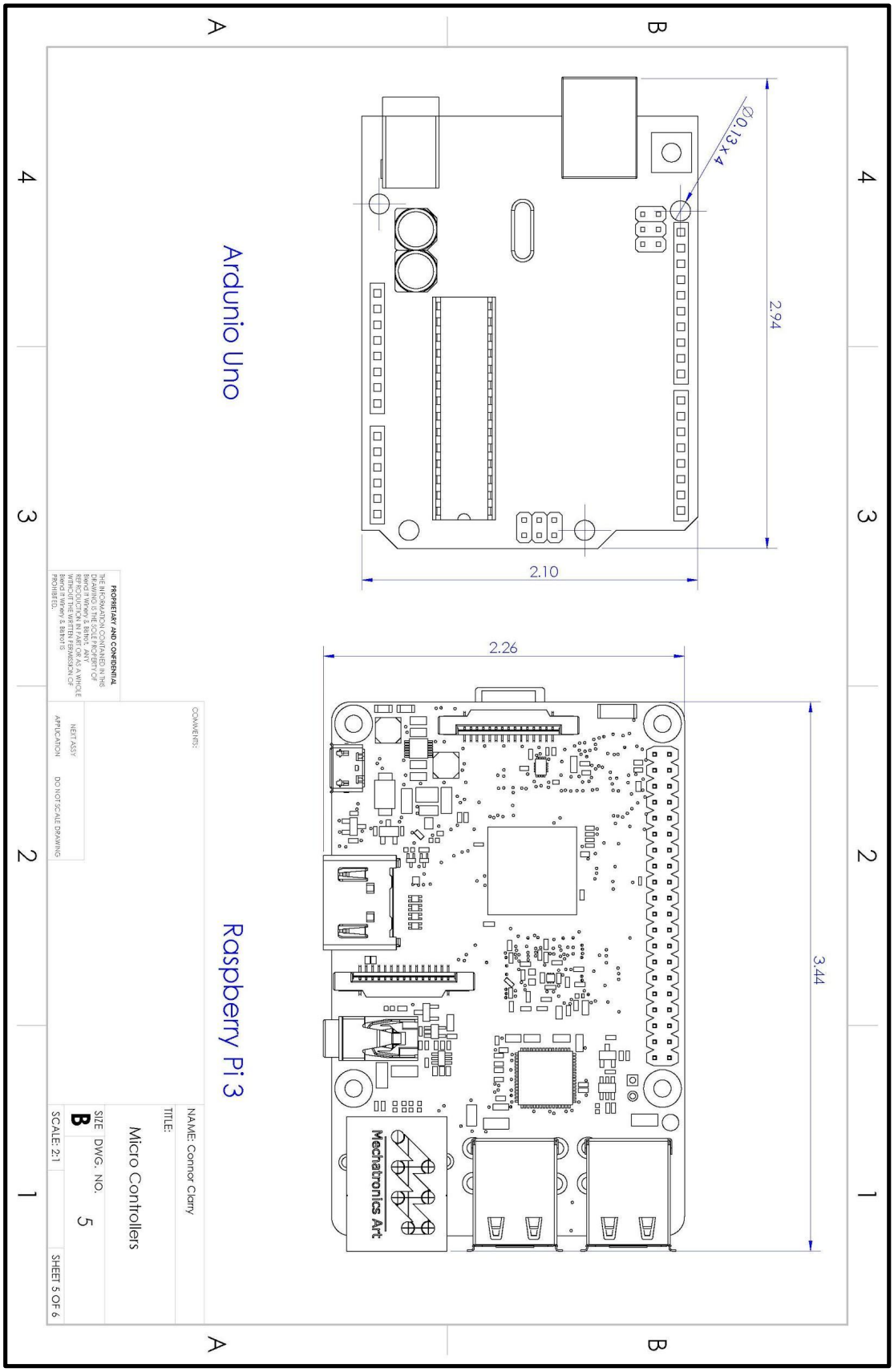
Swissflow SF800 Flow Meter



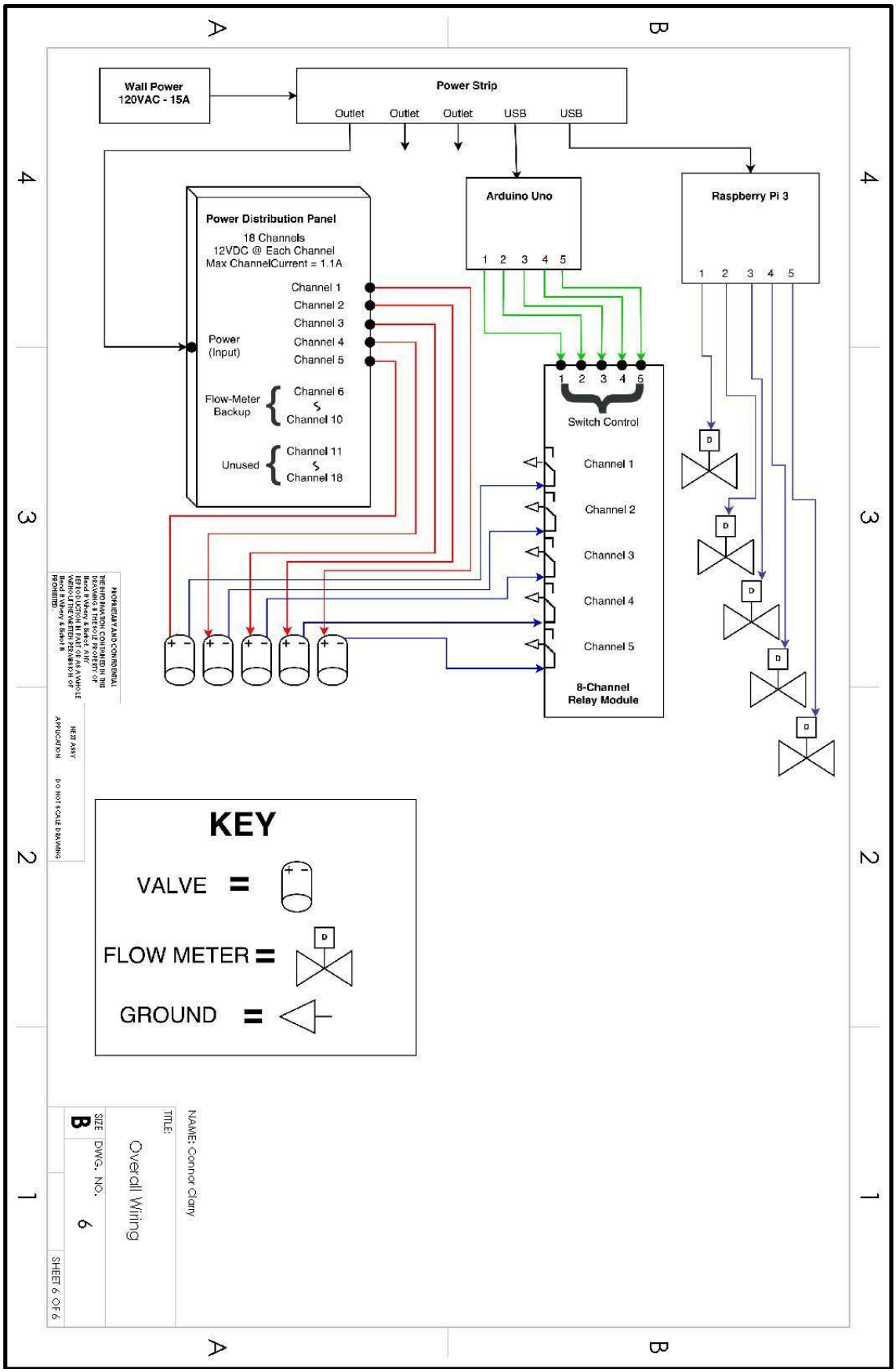
Clippard 3-Way Isolation Valve



Raspberry Pi 3 and Arduino Schematics



Top Level Full System Wiring Diagram



Appendix C: BOM

tem	Item			Description	Qty	Cost/Unit	Total	Part Number	Supplier	Link
r	100			JoyNano 12V DC 18-Channel Distributed Power Supply Box	1	$49.99	$49.99	B01LWW5GGA	Amazon	https://tinyurl.com/gv23hlt
	101			10-Ohm 10-Watt 10% Wirewound Resistor (2-Pack)	5	$2.49	$12.45	2710132	Radioshack	https://tinyurl.com/jzt33kr
	102			100-Piece 1/2-Watt Carbon-Film Resistor Assortment	1	$7.49	$7.49	2710306	Radioshack	https://tinyurl.com/gmspa6c
	103			Poweradd 3-Outlet Power Strip 5-foot with 3 USB	1	$13.99	$13.99	PS-9009WE	Amazon	https://tinyurl.com/zaq4zdu
	104			JBtek 8 Channel DC 5V Relay Module	1	$8.98	$8.98	B00KTELP3I	Amazon	https://tinyurl.com/z9njvmt
	105			Micro USB to USB 3 Pack	1	$10.89	$10.89	B017OXNC28	Amazon	https://tinyurl.com/hxjcotz
berry Pi	200			Raspberry Pi 3 Model B Motherboard	1	$35.70	$35.70	B01CD5VC92	Amazon	https://tinyurl.com/hrlpzzp
	201			Samsung EVO 32GB Class 10 Micro SDHC Card with Adapter	1	$10.00	$10.00	MB-MP32DA/AM	Amazon	https://tinyurl.com/zbvknpn
	202			Raspberry Pi 7" Touchscreen LCD Display	1	$60.00	$60.00	83-16872	MCM Electronics	http://www.mcmelectronics.com/product/83-16872
	203			Swissflow Meter	5	$64.00	$320.00	SF800	Swissflow	http://www.swissflow.com/sf800.html
no	300			Smraza Starter Kit for Arduino with Uno R3	1	$12.00	$12.00	B01LX65IK2	Amazon	https://tinyurl.com/zossdw8
	301			3-Way Normally-Closed Isolation Valve	5	$86.68	$433.40	EDU-MXF1-SXYY	Clippard	http://www.clippard.com/part/NR4-3-12
urized Wine	400			Corney Keg (5.5 Gal)	5	30	150			
	401			Pair of Ball Lock Disconnects 1 Gas 1 Liquid	5	$8.25	$41.25	H5-RVY8-AEQH	Amazon	https://tinyurl.com/gr5qvrj
	402			Kuriyama Kuri Tec PVC, Clear, 5/16 inches ID	80	$0.18	$14.40	K050-0507X100	Amazon	https://tinyurl.com/z3sz2uk
	403			1/8" NPT x 5/16" Hose ID Black HDPE Adapter	10	$0.25	$2.50	62115	US Plastic	https://tinyurl.com/jnrb49q
sure	501			.25 in Tube Clamps	20	$1.38	$27.60	UC953A	Home Depot	Purchased in Store
	502			Acryllic Utility Panel	1	$17.97	$17.97	MC-21	Home Depot	Purchased in Store
	503			Aquarium Silicone	2	$4.57	$9.14	OO688	Home Depot	Purchased in Store
	504			Pegboard	2	$13.86	$27.72	109099	Home Depot	Purchased in Store
	505			.5 in tube clamps	5	$1.14	$5.70	6772595	Home Depot	Purchased in Store
	506			50 ft sprinkler wire	1	$21.77	$21.77	49275142	Home Depot	Purchased in Store
	507			11x14 Lexan Polycarbonate	1	$7.98	$7.98	LWS2048-4	Home Depot	Purchased in Store
	508			Snaptite Connector 3/8"	1	$1.28	$1.28	841ST-10	Home Depot	Purchased in Store
	509			1-1/8 PVC White	1	$5.08	$5.08	531194	Home Depot	Purchased in Store
	510			Construction Adhesive	1	$4.58	$4.58	LN 907	Home Depot	Purchased in Store
	511			.75 in Plywood	1	$16.48	$16.48	454559	Home Depot	Purchased in Store
	512			1'x1' Gal Steel Sheet	2	$5.98	$11.96	GFS12X12	Home Depot	Purchased in Store
	513			Machine Screw #10-32-.75"	6	$1.18	$7.08	800994	Home Depot	Purchased in Store
	514			SS Clamp 10pk	2	$8.57	$17.14	6720595	Home Depot	Purchased in Store
	515			6in Heat-Shrink tubing	1	$8.09	$8.09	2781611	Radioshack	Purchased in Store
	516			Power cord for Arduino uno	1	$4.99	$4.99	2603141	Home Depot	Purchased in Store
	517			LED Ring	1	23.53	$23.53	ZXRKCOBS-100MM-R	Amazon	https://tinyurl.com/y8yndywh
	518			White Polywall Panel 4 x 8	1	23.88	$23.88	63003	Home Depot	Purchased in Store
	519			Angle SLD Alum 96 x 1 x 1/16	1	$10.53	$10.53	800047	Home Depot	Purchased in Store
	520			Painters Touch 2x Gloss Cranberry Spray Paint	4	3.87	$15.48	249863	Home Depot	Purchased in Store
	521			Painters Touch 2x Gloss Black Spray Paint	1	3.87	$3.87	249061	Home Depot	Purchased in Store
	522			Lead-Free Brass Hose Barb Adapter 3/8 in. x 3/8 in. FIP	10	3.46	$34.60	800159	Home Depot	Purchased in Store
	523			1/2 in. x 260 in. Yellow PTFE Tape (Gas Teflon Tape)	1	2.97	$2.97	31403D	Home Depot	Purchased in Store
						Total	$1,492.46			

Appendix D: Specification Sheets from Vendors

JoyNano 12V DC 18-Channel Distributed Power Supply Box Specifications

Description:

1. The JoyNano 18-Channel Power Distribution Units (PDU) can distribute 12VDC power for up to 18 cameras, offer a more professional alternative to individual power adapters.

2. Max output 20 Amp total, maximum for each port is 1.1A, allows the power source to support several cameras simultaneously.

3. Each outlet is individually fused and each panel is surge protected. LED light for monitoring the power status.

4. Safety Protection: Overvoltage protection, short circuit protection and overload protection. Hiccup mode, recovers automatically after faulty condition is removed.

5. Pro-size box provides plenty space for wires, key lock prevent unauthorized access. Easy to install, can be mounted to any surface, plugged and up in seconds.

Specifications:

Input Voltage: AC110V/ 220V

Output Voltage: 12V DC

Output Current: 20A (1.1A Each)

Power: 240W Max

Channel Number: 18CH

Material: Metal

Dimension: 315 x 220 x 60mm (L x W x H) 12.4 x 8.7 x 2.4 inch

Net Weight: 2060g 4.5lb

Environmental Requirement:

Working temperature: -10 to 50 degree Celsius

Storage temperature: -20 to 60 degree Celsius

Environmental humidity: 10-95%

Usage:

Protects your electronic devices from over voltage damages, increase life of electronic circuits and products you are using with.

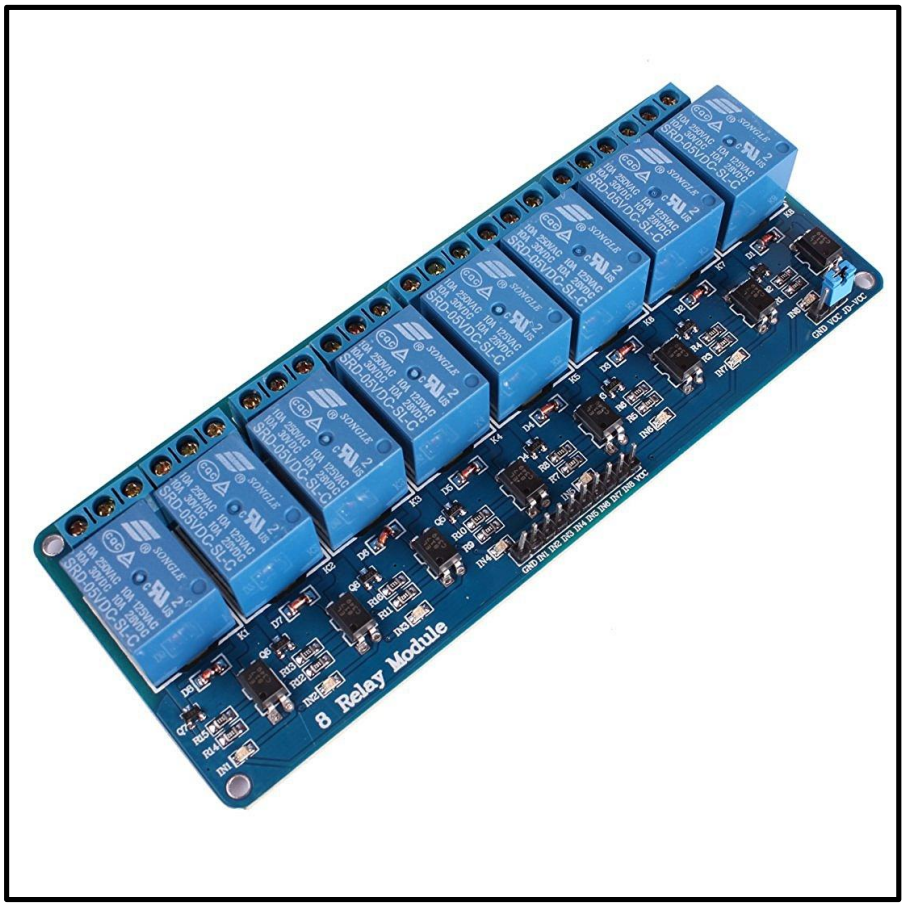
Ideal for CCTV surveillance, communications, LED strip lights, security camera, network, etc.

Notice:

1. Input voltage 110V/230V is selected by switch. Before power on please check input voltage avoiding damage.

2. Indoor use only. Professional Installation of qualified electrician is highly recommended.

JBtek 8 Channel DC 5V Relay Module



- 5V 8-Channel Relay interface board, and each one needs 15-20mA Driver Current
- Equiped with high-current relay, AC250V 10A ; DC30V 10A
- Standard interface that can be controlled directly by microcontroller (Arduino , 8051, AVR, PIC, DSP, ARM, ARM, MSP430, TTL logic)
- Indication LED's for Relay output status

Songle Relay Specification Sheet (Page 1)

SONGLE RELAY

松乐继电器 ® SONGLE RELAY	RELAY ISO9002	SRD

1. MAIN FEATURES

- Switching capacity available by 10A in spite of small size design for highdensity P.C. board mounting technique.
- UL,CUL,TUV recognized.
- Selection of plastic material for high temperature and better chemical solution performance.
- Sealed types available.
- Simple relay magnetic circuit to meet low cost of mass production.

2. APPLICATIONS

- Domestic appliance, office machine, audio, equipment, automobile, etc.
  (Remote control TV receiver, monitor display, audio equipment high rushing current use application.)

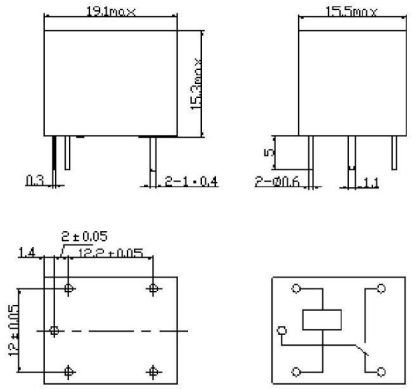
3. ORDERING INFORMATION

SRD	XX VDC	S	L	C
Model of relay	Nominal coil voltage	Structure	Coil sensitivity	Contact form
SRD	03、05、06、09、12、24、48VDC	S:Sealed type	L:0.36W	A:1 form A
				B:1 form B
		F:Flux free type	D:0.45W	C:1 form C

4. RATING

CCC	FILE NUMBER:CH0052885-2000	7A/240VDC
CCC	FILE NUMBER:CH0036746-99	10A/250VDC
UL /CUL	FILE NUMBER: E167996	10A/125VAC 28VDC
TUV	FILE NUMBER: R9933789	10A/240VAC 28VDC

5. DIMENSION(unit:mm) DRILLING(unit:mm) WIRING DIAGRAM



Songle Relay Specification Sheet (Page 2)

6. COIL DATA CHART (AT20°C)

Coil Sensitivity	Coil Voltage Code	Nominal Voltage (VDC)	Nominal Current (mA)	Coil Resistance (Ω) ±10%	Power Consumption (W)	Pull-In Voltage (VDC)	Drop-Out Voltage (VDC)	Max-Allowable Voltage (VDC)
SRD (High Sensitivity)	03	03	120	25	abt. 0.36W	75%Max.	10% Min.	120%
	05	05	71.4	70				
	06	06	60	100				
	09	09	40	225				
	12	12	30	400				
	24	24	15	1600				
	48	48	7.5	6400				
SRD (Standard)	03	03	150	20	abt. 0.45W	75% Max.	10% Min.	110%
	05	05	89.3	55				
	06	06	75	80				
	09	09	50	180				
	12	12	37.5	320				
	24	24	18.7	1280				
	48	48	10	4500	abt. 0.51W			

7. CONTACT RATING

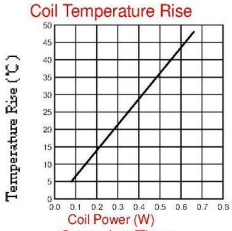
Item \ Type	SRD	
	FORM C	FORM A
Contact Capacity Resistive Load (cosΦ=1)	7A 28VDC 10A 125VAC 7A 240VAC	10A 28VDC 10A 240VAC
Inductive Load (cosΦ=0.4 L/R=7msec)	3A 120VAC 3A 28VDC	5A 120VAC 5A 28VDC
Max. Allowable Voltage	250VAC/110VDC	250VAC/110VDC
Max. Allowable Power Force	800VAC/240W	1200VA/300W
Contact Material	AgCdO	AgCdO

8. PERFORMANCE (at initial value)

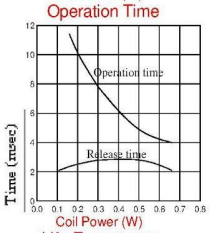
Item \ Type	SRD
Contact Resistance	100mΩ Max.
Operation Time	10msec Max.
Release Time	5msec Max.
Dielectric Strength Between coil & contact Between contacts	 1500VAC 50/60HZ (1 minute) 1000VAC 50/60HZ (1 minute)
Insulation Resistance	100 MΩ Min. (500VDC)
Max. ON/OFF Switching Mechanically Electrically	 300 operation/min 30 operation/min
Ambient Temperature	-25°C to +70°C
Operating Humidity	45 to 85% RH
Vibration Endurance Error Operation	 10 to 55Hz Double Amplitude 1.5mm 10 to 55Hz Double Amplitude 1.5mm
Shock Endurance Error Operation	 100G Min. 10G Min.
Life Expectancy Mechanically Electrically	 10^7 operations. Min. (no load) 10^5 operations. Min. (at rated coil voltage)
Weight	abt. 10grs.

9.REFERENCE DATA

Coil Temperature Rise

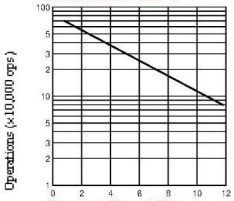


Operation Time

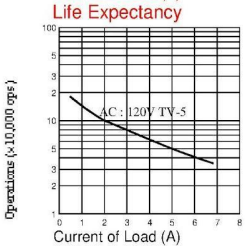


Life Expectancy

AC120V/DC24V cosΦ=1

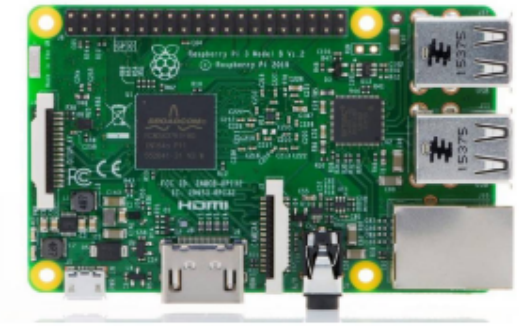


Life Expectancy



Raspberry Pi Specifications

Raspberry Pi 3 Model B



RASPBERRYPI-MODB-1GB

RPI-MODB-16GB-NOOBS

Technical Specification:

- Broadcom BCM2837 64bit Quad Core Processor powered Single Board Computer running at 1.2GHz
- 1GB RAM
- BCM43143 WiFi on board
- Bluetooth Low Energy (BLE) on board
- 40pin extended GPIO
- 4 x USB 2 ports
- 4 pole Stereo output and Composite video port
- Full size HDMI
- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source (now supports up to 2.4 Amps)
- Expected to have the same form factor has the Pi 2 Model B, however the LEDs will change position

Arduino Uno Specifications

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Appendix E: Detailed Supporting Analysis

Valve Power Calculations

Terms: Resistance = R; Power = P; A = Amperage; V = Voltage; I = Current

Equations:

Voltage = Current * Resistance (V=IR)

Power = Current * Voltage (P = IV)

Resistance = R; Power = P; A = Amperage; V = Voltage; I = Current

R_{VALVE} = 20Ω

P_{MIN} = 1.0W

P_{MAX} = 7.2W

$P_{DESIRED}$ = ~5.0W

V_{INPUT} = ~12V

I_{INPUT} = 1.1A

To achieve a power draw of 5.0W at 12V, we need a current of ~.42A.

With an internal resistance (R_{VALVE} = 20Ω), the current will drop from 1.1A to .6A.

To step the current down from .6A to .42A, an additional ~8.6Ω or resistance will need to be added to the circuit.

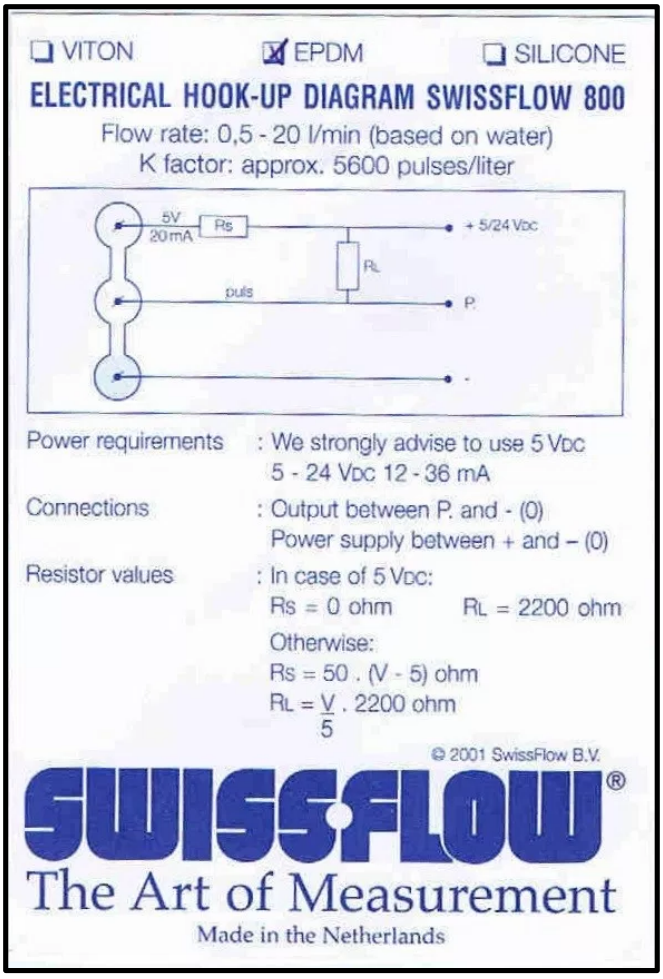
For the additional resistance, we will use a 10Ω resistor rated to withstand up to 10A of current.

This will bring the total circuit resistance (R_{TOTAL}) to 30Ω.

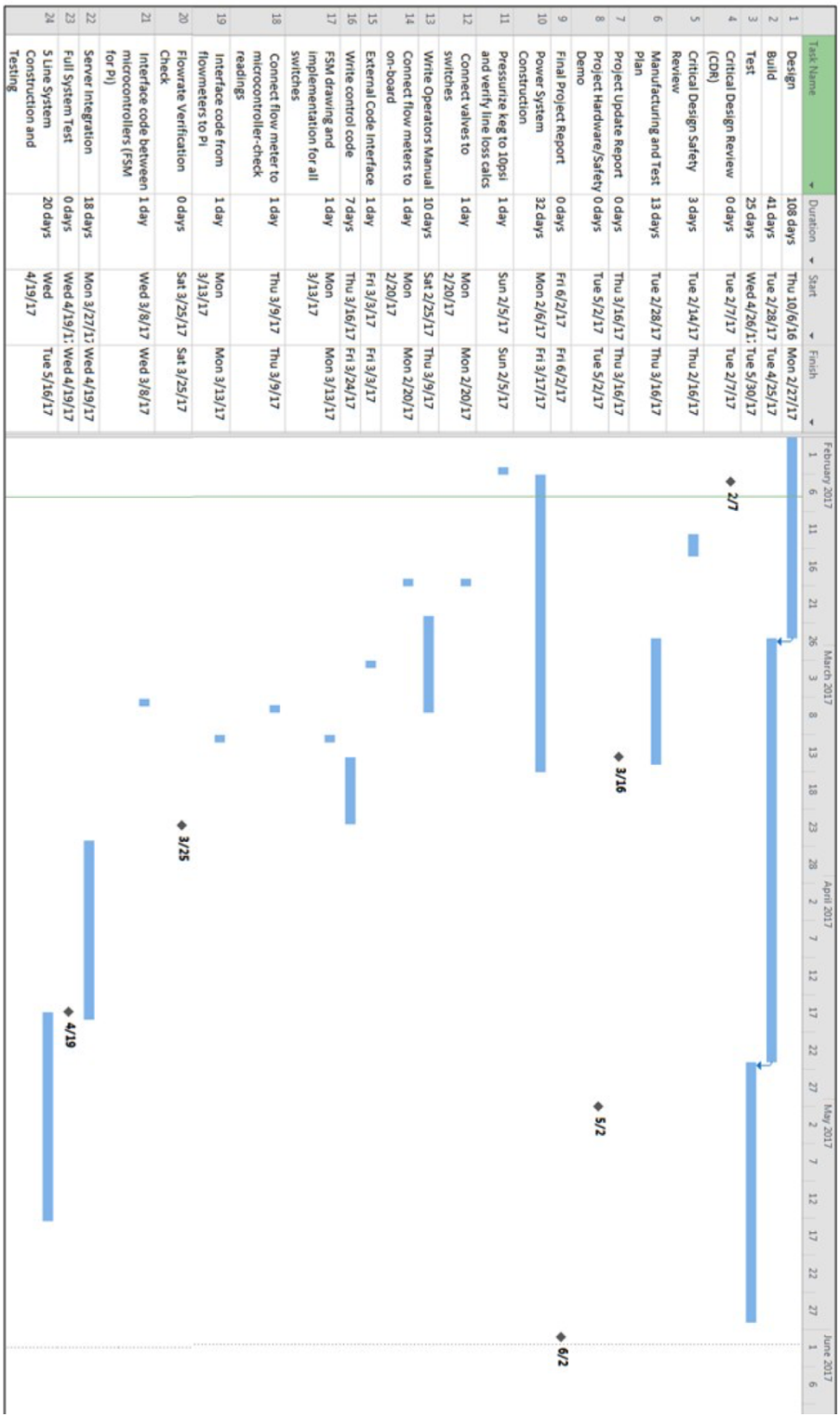
Using I = V/R, we can show that at a constant 12V with 30Ω of resistance, the current in the circuit will be .4A. Using P = IV, we can show that with 12V and a current of .4A, the total power draw of the valve will be 4.8W. This is safely in the range of 1.0W and 7.2W and close to our desired power (5.0W) that the valve specification sheets lists as an acceptable power draw.

Flow Meter Power Calculations

The spec sheet provided below for the Flow-Meter provides equations for the resistances to be used at various voltages. The manufacturer recommends powering the meters at 5VDC. We will do this with our Raspberry Pi microcontroller. At 5VDC, additional resistance of 2.2kΩ needs to be added in parallel with the input voltage and pulse output of the meter. We purchases a 2.2kΩ resistor that will provide this extra resistance. In the event the Raspberry Pi is unable to power the flow-meter, we will use our power supply box at 12VDC, which is in the acceptable range of 5VDC and 24VDC the flow-meter manufacturer provided. Running the meter at 12VDC will change the extra resistance needed in the circuit. A resistor of 350Ω will be added in series with the power supply input and a resistor in parallel of 5280Ω will be added between the input voltage and pulse output of the flow-meter.



Appendix F: Project Gantt charts



Appendix G: Operators Manual with safety guidelines.

Electrical Safety Warning:

This device contains a 12VDC power supply box at the top of the unit. This box has a lock and key and is to remained locked as long as power is being supplied. There is an external red LED on the outside of the box indicating if it is receiving power. **DO NOT ATTEMPT TO MAKE CHANGES TO THE ELECTRIC CIRCUITRY WITHOUT CONSULTING AN OPERATOR AND/OR ELECTRICIAN.**

Running the System:

Turning on: The system turns on by flipping the switch on the power strip that the system is connected to.

Starting Node-Red: Node-Red starts by typing the command "node-red-start" in the operating system terminal.

Viewing Node-Red: Opening the web browser on the Raspberry Pi will take the user to the user interface of the system. Two separate tabs will open. The first tab will be the flow overview of the entire system The second tab shows the amounts poured and pour status of the current blend.

Pouring: To pour a blend that is on the server queue, simply go to the UI homepage for Node-Red. In the top left make sure that you are displaying the "Home" tab. To pour the blend, hit the "Pour Order" button. To pour consecutive blends, hit the "Reset" button and then "Pour Order" again. **You must hit "Reset" between pours or the device will not function properly.**

Troubleshooting:

Node-Red Failure:

If Node-Red fails to start or the web interface fails to load, there are a few quick troubleshooting things that can be checked. The first is ensuring the Raspberry Pi is connected to the internet.

This is done by looking at the top right of the screen at the WiFi or Ethernet symbols. If Node-Red still does not start, in the terminal type in “node-red-stop” and then “node-red-start”.

No Valves Opening Failure:

If valves appear that they are not opening, first ensure that there is power supplied to the box. A red LED on the outside of the power supply box should be lit. If they still won’t open, use a multimeter on VDC settings to check one of the channel voltages. If it is lower than 12VDC, the valves will not function properly. A screw inside the power supply box can be twisted to change the voltage on the device. Twist this very slowly until the multimeter gets a reading between 12VDC and 14VDC.

Server or port switching:

If the whole system seems to spontaneously disconnect or stop functioning, it is most likely due to one of the microcontrollers being assigned wrong. This is usually fixed by completely turning the system of and back on again.