DESIGN DOCUMENT: MINI-MIXER

DEPARTMENT OF ELECTRICAL ENGINEERING & COMPUTER SCIENCE
UNIVERSITY OF CENTRAL FLORIDA
DR. SAMUEL RICHIE
SENIOR DESIGN I

GROUP 14
THOMAS "TJ" BERGENS
TJBERGENS@KNIGHTS.UCF.EDU
WILLIAM "DAVIDSON" TUGGLE
DTUGGLE@KNIGHTS.UCF.EDU

TABLE OF CONTENTS

1. EXECUTIVE SUMMARY

The concept of an autonomous drink mixer has been around for quite some time, and has been realized in different projects and products. Like many other concepts, such realizations tend to have their set of advantages and disadvantages. The Mini-Mixer is a project that tries to maximize the advantages shared in previous projects, while also trying to minimize the disadvantages. One of the main factors that is to separate the Mini-Mixer from other similar projects is the performance to price ratio; a ratio in which the Mini-Mixer will strive to make as high as possible. Quality is another major factor in the Mini-Mixer, where the system as a whole will be simple and intuitive to use, all the while being flexible enough to allow customization that will satisfy a user's needs. Safety is very important to the design of the Mini-Mixer; important enough that certain subsystem designs were heavily influenced by it. Supporting technology that is prevalent was also a high priority in the design of the Mini-Mixer, with a mobile app specifically designed to be the primary interface for it. All of these factors were decided to be integrated into a compact package no larger than the average sized kitchen appliance.

2. PROJECT DESCRIPTION

The idea of the Mini-Mixer was brought about after several brainstorming sessions while trying to determine a project that was fit for our group's collective skillsets. Our basic requirements for such a project was to have something that included a multitude of modern technologies that are relevant in today's industries. We wanted to take these technologies and find a practical way to apply them to a problem. Once a problem and general solution had been determined, we wanted to take proposed features and find interesting ways to implement them given the context of this project.

2.1 DESCRIPTION

The Mini-Mixer is a small appliance designed to be compact, fast, and user-friendly for everyday use in a kitchen or dinner/garden party setting. The appliance is designed to automatically mix drinks as quickly and accurately as possible. The appliance takes advantage of the user's smart phone to provide a simple, intuitive interface to create and order drink mixtures on the machine via wireless communication. The mixture management interface is used to store custom mixtures, as well as suggest new or popular mixtures for the user to try, based on what is available in the machine.

2.2 MOTIVATION

As the world inches closer and closer to automation in all facets of the consumer's life, there is significant room for improvement in the area of beverage dispensers. Specifically, there are very few options for a high quality drink mixer that is both affordable and small enough to carry to different locations such as parties, beach houses or other "getaway" destinations. A compact drink mixer can serve as both a household appliance, as well as a novelty at a dinner or garden party. With the introduction of the mobile smart phone that has come to dominate the tech gadget market, we can use this to our advantage to provide a "remote" of sorts to the drink mixer as a means of further adding convenience and features at little to no additional cost on the part of the design of the machine. This realization opens us many possibilities to enhance a drink mixing experience, such as the ability to have user accounts with favorite drinks and the ability to share recipes with friends with ease.

2.3 GOALS AND OBJECTIVES

The main objective of his project is to design a drink mixer that is small and simple enough to have as many use cases as possible for the consumer. A one-off permanent fixture such as Coca-Cola's Freestyle machine [1] and similar products is not viable for the everyday consumer. However, these types of machines have become extremely popular at restaurants in recent years due to the combination of ease-of-use and high level of customization in mixing the drinks. The average person will only be willing to purchase an appliance such as this if the number of use cases increases to a point that makes it a reasonable investment for the size and price point of the appliance. We plan to achieve this by making the unit as small as possible to become a semi-portable machine that has a robust feature set suitable for many applications in the consumer's life.

3. REQUIREMENTS AND SPECIFICATIONS

The following set of requirements specifications are what we will use to define all functional features in the design of the Mini-Mixer. These requirements specifications have been carefully selected to reflect the general motivation and goals outlined in the project description. The requirements specifications have been divided into two contexts – the hardware and the software specifications. With each specification, we provide a brief reasoning and insight into the selection of the requirement specification and how it directly applies to the goals and objectives of the project.

3.1 HARDWARE REQUIREMENTS SPECIFICATIONS

- The unit will be designed to be very compact – small enough to place on a tailgate, countertop, or take out to your patio.

- The unit shall have a dry weight of no more than 40 pounds.

- The "mixing" process is to be reasonably fast and should not keep the user waiting too long.
  - The unit should produce a mixed drink from start to finish in no longer than 1 minute.
- The "mixing" process should have good accuracy, with a margin of error that you could expect from a human bartender.
  - The amount of fluid in the components of each mix should have an error of no more than +/-10%.
- The variety of drinks should be sufficiently large so that the user has enough choice of mixes that they are not constantly switching out drinks or frustrated at the lack of options.
  - The unit shall provide enough resources to hold six different fluids.
- The unit should have a power rating that is safe for both the branch circuit of the source as well as for the user. A power rating around the average small kitchen appliance shall be chosen.
  - The unit shall consume no more than 600 Watts of power under load.
- The mixer is meant to be low-cost to be viable in the consumer market. The prototype should be comparatively low-cost as well.
  - The mixer prototype should have a combined total cost of no more than $800.
- The mixer should be extremely easy to use through a mobile device.
  - The mixer will be controlled using a mobile device with an application.
- The mixer should be both portable and have a semi-universal bottle acceptance. These containers should be able to accommodate both standard 750mL spirit's bottles as well as 1-2 Liter soda bottles.
  - The size of the accepted fluid containers shall be no higher than 250mm.
- The mixer should be able to produce a mixture at a chilled temperature that is appropriate for consumption of cocktails and general fluid mixtures.
  - The unit should produce a mixture with an initial temperature of no higher than 55 degrees Fahrenheit.
- The unit is meant to be semi-portable as well as small enough to fit on common tabletop areas. With this in mind, we are aiming for a size similar to the common microwave appliance.
  - The unit shall have dimensions no larger than 2-foot Height X 3-foot Width x 3-foot Depth.
- The unit should be able to accommodate most popular cocktail glass types including highball, Collins, and martini glasses.
  - The unit shall accept a glass size of 6 inches in height and 4 inches in diameter.
- The unit's server should be able to accept cocktails submissions from the mobile device at an acceptable range from the unit. With this in mind, the range should be acceptable for a kitchen or outdoor gathering setting.
  - The unit should be able to accept cocktails orders from a range of up to 20ft.

3.2 SOFTWARE REQUIREMENTS SPECIFICATIONS

- The mobile application is meant to be very user-friendly and simple to use, while still providing a variety of mixture choices.
  - We're placing a limit of no more than a 4 step process from the application's start screen to a drink in the cup.
- The mobile application should be able to create all possible combinations of the given fluids in the mixing unit.
  - The application should be able to create 128 different combinations of the fluids in the machine.
- The user should be able to adjust the fluid ratios in their mixtures. This must also be in line with the expected accuracy of our pumps used for the fluids.
  - The mobile application should be able to create mixtures in units/steps of 0.5oz.
- The mobile application should limit the total mixture size to a value that satisfies the cup size, average cocktail size, and expected size of general mixtures that are otherwise not cocktails.
  - We are placing a limit of no more than 8 ounces (~237 milliliters) on the total mixture size that application can create.
- The time it takes from submission of the mixture until the mixing unit is ready to begin mixing should be sufficiently fast so that the user is not frustrated with the wait time.
  - The total time from submission of the mixture until the machine begins mixing should be no longer than 1 second.
- The user should be able to store a large number of custom drinks in their profile.
  - The maximum allowed custom mixtures for a single user will be limited to 100.
- The user should be able to view a top list of most favorited/popular drinks while still allowing the application to remain performant.
  - The maximum allowed size of the top list will be limited to 100.
- The mobile application should only provide the amount of drinks available to the machine.
  - The application will provide the limit of drinks on the machine, which is 6. Specifically, the application may only provide options for the current types of the 6 drinks in the machine at any given time.
- The application should have the ability to suggest drinks to the user, given the ingredients on hand. This will add a component of exploration and convenience to the user when they are ready to try something new.
  - The client application will need to have the ability to suggest at least 1 mixture to the user, based on available drinks.

4. REALISTIC DESIGN CONSTRAINTS

Realistic Design Constraints are to be carefully considered, as this can greatly impact the outcome of the design as well as the implementation. Failing to consider major design constraints can result in complications or failure to implement components or even entire requirements specifications during the prototype construction phase.

4.1 ECONOMIC AND TIME-BASED

There is a considerable constraint placed on the Mini-Mixer economically. In order to produce a system that will be accessible to many people, it must be affordable enough to manufacture. Even in the home setting, there are still economic constraints; high power consumption is not attractive, as it will lead to a higher electric bill. These factors must be taken into consideration when designing the Mini-Mixer for such a large range of users. The time frame to research, produce, implement, and test a working prototype of the Mini-Mixer is around a total 20-25 weeks. This is not considering other course loads, work or personal obligations of the team. This leaves us with a very limited time frame to go from an idea to a working prototype of the Mini-Mixer. We are considering this to be a fairly significant time-constraint that must be considered in the scope of the project. Our team has also been restricted in the human resources that are contributing to this project. Originally, the Mini-Mixer team consisted of 3 persons: Two Computer Engineering students and one Electrical Engineering students. Due to unforeseen circumstances, we are now a team of two Computer Engineers working on the design of the Mini-Mixer. This poses and issue of limited knowledge surrounding the Electrical Engineering aspects of our design. We also have the obvious human resource limitations that come with working in a team of two persons. This is considered a major constraint on our team that we must carefully monitor throughout the design and implementation of the Mini-Mixer.

4.2 ENVIRONMENTAL, SOCIAL, AND POLITICAL

From an environmental and political standpoint, there are no constraints imposed on the Mini-Mixer in any capacity. Although there aren't any social constraints imposed on the Mini-Mixer per se, there is some converging point between social and safety that must be considered, and that is on alcoholic beverages. This is further discussed in Section 4.3.

4.3 ETHICAL, HEALTH, AND SAFETY

One major consideration that arises out of designing and implementing a drink mixer is that of the risk surrounding serving alcoholic drinks. This actually touches on ethical, health, and safety issues altogether. We must be aware of

the amount and types of alcoholic beverages we are mixing and dispensing from the machine. For example, we cannot allow the machine to dispense “too much” alcohol to a single user in a given time frame. We also must consider the ethical issue of the age required to drink alcoholic beverages. Ideally, we will need to have mechanisms in place to prevent minors from being able to dispense alcoholic beverages at their leisure.

4.4 MANUFACTURABILITY AND SUSTAINABILITY

Manufacturability and sustainability influences the design of the Mini-Mixer considerably, because in order to satisfy these constraints, the Mini-Mixer must be designed with modular subsystems. These subsystems must all function properly at the lowest level, and must be compatible with other subsystems at the high level. At the highest level, each subsystem must be replaceable. With subsystems that are both replaceable and compatible with each other, it is possible for them to be manufactured independently at the same time. For sustainability, the option to be able to replace a subsystem is critical; without it, it is unlikely to sustain such a system for a prolonged period of time.

5. RESEARCH

The research behind the Mini-Mixer is a key component of the design process, particularly in early development. Our research efforts have been used to drive the design of the Mini-Mixer early on in the process. Specifically, we have been able to iterate over proposed implementations of our requirements specifications very quickly by researching all available solutions, as well as previous efforts to tackle our problem set. We have been able to utilize the findings of similar projects to throw out bad ideas as well as use previous projects as a foundation of inspiration for the design and implementation of the Mini-Mixer.

5.1 SIMILAR PROJECTS

During the early phase of research, we encountered many different projects that have attempted to tackle the same problem space. Some of these projects were the basis for inspiration later in our own design. Some had obvious issues in their choice of design which allowed us to avoid these approaches very early on. The projects with complete documentation were particularly helpful in gaining some insight into the thought process behind their respective implementations in the problem space. Proper documentation allowed us to study designs as well as part choices for each feature and even how they fared during the testing phase. We used these findings in our own research and considerations for the part choices in our own designs. For this reason, we chose to pursue further research only on projects that had sufficient documentation for their design and implementation process.

5.1.1 PROJECT-BASED SOLUTIONS

Under the Sun Drink Mixer was a project designed by students here at the University of Central Florida that shares a similar set of goals to that of the Mini-Mixer. Its primary purpose is to provide a system that can mix drinks using solar energy as its power source. Like the Mini-Mixer will be, it is an autonomous device that can be operated via a mobile phone application. It features a barcode scanner that allows a user to scan a barcode that will reference a local server to determine what drink to make. The machine used a gas based solution along with 12 Volt solenoids to serve as the pumping mechanism. The case was constructed using plywood, and the fluid bottle compartment used was a large Styrofoam box.

The *Automated Beverage Dispenser* was a project designed by students at Georgia Tech. It features a touch screen interface that is drove by a programmable logic controller running a modified version of VxWorks's real time operating system. Relays are used to control the fluid pumps, and liquid level sensors are used to detect the levels of various fluids used. The hardware for this project was divided into two units, 'dispensing unit' and 'control unit'. Similar to the Mini-Mixer, the system takes in a user generated recipe and makes the drink within a reasonable amount of time. An interesting note is that the cost of this particular project was over $3000, primarily due to the high priced hardware that was used. One of Mini-Mixer's goals is to be reasonably cheap to produce, so practically none of the hardware used in this project will be used in the Mini-Mixer, although some of the ideas that incorporated said expensive hardware may be similar to that found in the Mini-Mixer.

The *Automated Drink Mixer* project was designed by students at Oregon State University. Like the other projects discussed, the primary goal was to provide an autonomous system that will make drinks for users. *Automated Drink Mixer* 's other goals are interesting in that they align pretty close with that of the Mini-Mixer's. One immediate goal that stands out drink mixing requirement of under 120 seconds, which is close to the Mini-Mixer's requirement of a minute or less. Another goal similar is that of the size and cost constraints; the size constraint of the *Automated Drink Mixer* is only two feet taller than Mini-Mixer's constraint, and $200 cheaper than Mini-Mixer's cost. The weight of the Mini-Mixer is to be 10lb less than the *Automated Drink Mixer*, which is also similar quantitatively. An interesting goal specified in this project that was not explicitly handled in Mini-Mixer's initial requirements was the maximum voltage level; from a safety perspective, this is a requirement that addresses both hardware design and safety. For the *Automated Drink Mixer*, that maximum was set to 50V. Interestingly, this is roughly under the minimum voltage that can negate the effects of the human body's resistance, which can make the current passing through potentially lethal. For this reason, a maximum voltage will certainly be taken into consideration during the design of the Mini-Mixer.

The *SFSU Drink Mixer* was designed by students at San Francisco State University. Once again, its primary purpose was to be an automated drink mixer that a user could easily control. The hardware design of this project in theory was good, but the actual execution and outcome wasn't as good. The construction used was an open style wood design, with exposed wiring and tubing. A breadboard was used, so no permanent design was in place. Only two containers for fluids were used, and said containers were polycarbonate mason jars. The concept that was used for fluid level detection was interesting; a differential pressure sensor was used in order to do so. However, there seemed to be an accuracy issue with this approach, where the team suggested "better signal conditioning" to mitigate this. Nevertheless, this approach is interesting and may be further researched for the Mini-Mixer's purposes.

5.1.2 COMMERCIAL SOLUTIONS

Bartendro - The Bartendro is a Kickstarter-baked project to develop and commercialize an automated drink mixing product. [2] Their product uses custom peristaltic pumps with Arduino-powered PCBs embedded in each pump to control the flow of liquids to the dispensing area. These are all controlled using a Raspberry Pi which is used to serve the UI and dispatch commands to each pump. The hardware and software of the Bartendro is completely open source which allows for replication and modification. The commercial version of the Bartendro comes with up to 15 peristaltic pumps Which allows a wide range of drink combinations. The Bartendro 7 is the closest with respect to features as we envision with our own design. The Bartendro 7 has a price tag of $2,499.99 which is far above our cost constraints. The open-source nature of this project allows us to study and discuss similar functionality for our own design and functional specifications and requirements. The most interesting takeaway from the design of this solution was their design choice to power each peristaltic pump with an attached Arduino-powered PCB. This makes for a very expensive pump to sell but at the same time this allowed for some very interesting features and overall design choices. For example, having an entire microcontroller for each pump allowed for embedded illuminating LEDs complete with animations sequences during the pumping process. This also allowed them to use Ethernet ports to network all of their pumps together for communications.

Coca-Cola Freestyle Machine - The Coca-Cola freestyle machine is a large standing machine that vends hundreds of combinations of soda and juice mixtures. The machine is meant to replace the traditional fountain drink station that dominates the restaurant industry as well as quick-service stations such as convenience stores. While the machine's portability and power requirements are well out of the range for our project, it is worth noting the easy-to-use interface that comes with the machine. The freestyle machine sports a massive touch-screen on the unit, as well as a mobile application as a means of interfacing with the device. Both of these user experiences are very straightforward and simple to

use, while maintaining an impressive level of drink customization. Both UIs have a similar interface, with the mobile app offering an accounting feature where a user can store their own mixes and even share them with others. The UI itself defaults to a tree-based layout of “bubbles” that represent a main drink type. You can select the main drink type and proceed down a level in the tree to encounter several variants of that base drink, including additional flavors as well as diet and caffeine-free choices. The entire process for selecting any pre-made drink on the machine is no more than a few steps, with each step simply being a touch event on the screen. The mobile application offers a way to save favorites of the pre-made mixes for a user. The user can also create their own drink combinations under a “My Mixes” menu where they can store and edit their own mixtures. The mixture creation process is particularly interesting as the user is able to select up to three different ingredients and adjust their proportions by altering a pie chart using the touch screen. This makes for a very simple and intuitive interface for creating and editing drink mixtures.

5.2 HARDWARE COMPONENT CONSIDERATIONS

Our first set of considerations is that of the hardware components. This is arguably the most important of all component considerations as this drives our core design which determines things such as power consumption, size, and software component choices. The chosen hardware components will also determine the technologies that are available to consider as well. As we are interested in diversifying our usage of different technologies in the design of the Mini-Mixer, this will be the primary concern when considering the viable considerations of the hardware components.

5.2.1 EMBEDDED CONTROLLER

The embedded controller is essentially the “meat” of the project and will be responsible for doing most of the heavy-lifting in the context of actually controlling other pieces of hardware and producing the main goal of the Mini-Mixer. The embedded controller is going to be primarily responsible for the control of the hardware liquid pumps as well as the communication to and from the Embedded Server. This will include things such as accepting mixing instructions from the Embedded Server, accepting update requests, and reporting this information back to the Embedded Server. We will have to consider features such as support for Pulse Width Modulation (PWM) to control the fluid pumps. The selection of the controller will need to be carefully selected to be able to support the requirements of our pumps and the communication between the Embedded Controller and Embedded Server. The Texas Instruments MSP430 family is one that is rich and diverse, offering many different generations and series of microcontrollers within it. The one specific controller that was researched was the MSP430G2553, which is one of two controllers that comes with the TI LaunchPad. Due to the familiarity of which the team already has with this

particular controller, it has considerable potential to become the MCU of choice for the MiniMixer. The primary advantage of this µC is the ease of which it is to program it; there are two main IDE's that can be used, each of which providing a certain level of simplicity and control. GCC supports the MSP430 architecture, which helps in the development process. It contains 24 GPIO pins, which is a reasonable amount for the price point of this microcontroller ($2-$4/unit). This particular controller is clocked at 16MHz. The controller also comes with other nice features, such as two built in 16 bit timers, support for UART, I2C, and SPI, as well has having 8 built in comparators and a built in temperature sensor. Because of its affordability, the LaunchPad series has had a reasonably sized community formed on the internet, with many tutorials and documentation available. The power consumption of the MSP430G2553 is very low, which is negligible compared to the rest of the hardware's power requirements. Similar to the MSP430 family, the ATMega family also has a large variety of microcontrollers. Of the family, the primary interest of the team's research was the ATMega328 µC. This particular controller was chosen because it is the same one used on the very popular Arduino Uno. Inherently, it has a plethora of community support. Many "maker" movements use the Arduino Uno, some of which have internet video channels specifically for projects utilizing the Uno. The controller can be programmed with the Arduino IDE, which also has much support online as well. If necessary, it is possible to program in assembly with this controller with no additional hardware, and some extra software that can be acquired for no cost. The 328 has 23 GPIO pins, which like the MSP430 is a reasonable amount for the price point. Like the MSP430, it contains support for UART, I2C, and SPI, while also having a built in temperature sensor. The 328 has a clock rate of 20MHz, which is a small but noticeable difference over the MSP430's 16MHz clock rate (assuming each operate at around 1 cycle per instruction). The MSP430, however, contains more voltage comparators than the 328 does. Perhaps the biggest advantage of this controller is its support for additional hardware due to it being the controller used on the Uno; any hardware 'shield' that is compatible with the Arduino Uno is also compatible with the 328. This brings a lot of opportunities to the project with this much support. The PIC24FJ64GA002 is a microcontroller offered by Microchip that provides a decent amount of features and functions. Like both the MSP430 and the ATMega328 controllers previously mentioned, this controller also supports UART, SPI, and I2C serial communications. It has 21 general purpose input/output pins, which is a bit lower than either of the previously mentioned controllers, but it makes up for this by having five 16 bit and two 32 bit timers built in. It is capable of providing 16 MIPS, which is comparable to the other two controllers assuming they operate at around one cycle per instruction. This controller also contains a 10-bit ADC with 16 channels, supporting 500k samples per second. This controller contains two voltage comparators, which although less than what the MSP430 provides, is one more than what the ATMega328 provides. One disadvantage that is not necessarily the fault of the controller itself is the lack of a programmer bundles with it. Perhaps most important drawback of the controller is the lack of community support; although some tutorials for this

controller do exist, it is not nearly as abundant as those tutorials found for the MSP430 and ATMega controllers.

5.2.2 EMBEDDED SERVER

The Embedded Server is the “heart” of the entire system as it is responsible for handling messages and commands between the Client System and the Embedded Controller. The Embedded Server will also require all hardware necessary to facilitate communications for both the Client System and the Embedded Controller. The Embedded Server will be hosting the server component of our client-server software stack so we will have to consider hardware choices that will support a modern web server stack. As communication between the Embedded Server and Embedded Controller will almost certainly be a form of wired serial communications, we will have to consider hardware components that support this directly. We will also require an Embedded Server that is lightweight and can be portable enough to meet our Requirements Specifications. The Raspberry Pi was the first viable option that was researched for the embedded server. The second generation Model B was the model that was chosen for research. This particular model boasted a quad core 900MHz ARM Cortex A-7, with 1GB of SDRAM and HDMI support. The second gen model B had 46 GPIO pins, which is more than enough for Mini-Mixer’s purposes. Power consumption of the Raspberry Pi is roughly 4W, which is low relative to the power requirements of other components of the Mini-Mixer. With these hardware specifications, the Raspberry Pi is a strong possibility to handle the server side requirements of the Mini-Mixer. One other strong suit of the Raspberry Pi is its large community, with many tutorials on the internet for all sorts of projects. The BeagleBone was the second option that was researched for the embedded server. The newest version, BeagleBone Black (BBB) was researched for the project. It sports a 1GHz ARM Cortex-A8, 512MB of DDR3 RAM, and 69 GPIO pins. The BeagleBone consumes less power than the Raspberry Pi, at around 2-3W. The BeagleBone also has 65 GPIO pins, providing even more expandability than the Raspberry Pi. The BeagleBone’s design has been fully open sourced, with schematics and code available straight from a wiki. Because the BeagleBone has not existed for as long as the Raspberry Pi, it does not have nearly as large of a community. In some instances, it can be difficult to find tutorials for certain applications. Nevertheless, the community is still of respectable size with a lot of documentation on the community wiki. Overall, the BeagleBone essentially provides better hardware than what the Raspberry Pi does with less power consumption.

5.2.3 CLIENT SYSTEM

The client system is the main method by which the user interfaces and controls the mixing machine. Our considerations are based on ease-of-use as well as the typical devices that the user has at their disposal. We are most interested on

selecting a client device that meets or exceeds our ease-of-use requirements of the project. This means selecting a hardware device that the client is familiar with as well as a device that can host the client software that will meet those needs as well. Our first consideration is that of the mobile platform. In the case of mobile devices, the hardware device is almost always tied to one Operating System. This is an unfortunate outcome of the industry but this will allow us to choose one type of device and develop a polish and feature-complete application. As the Operating System is generally not a choice with the hardware vendor, we will be considering a wide range of hardware devices that will accompany the Operating System of our choice. However, we must keep in mind the devices that are owned by the team or have the resources to require, as we must ensure that the Client Application is fully working through extensive testing. For this reason, we will be considering only hardware devices that are already available to the team. In effect, the choice of hardware becomes directly tied to the choice of the Operating System. The mobile platform is ideal as many mobile hardware devices come equipped with several different means of communication, mostly being wireless. However, the devices typically have a wired connection in the form of a Universal Serial Bus (USB) that can be used for wired serial communications. This is important to consider as there are some initial setup steps that may be required to have a working wireless connection. Most of these devices are very light and portable, which allows the user to walk around while operating the application. There is a concern about the user sending commands to the drink mixer remotely, which may result in liquid spillage, though this can be mitigated through software to some extent. Another hardware consideration for the Client System is that of the Personal Computer. The advantage of this is we are not necessarily tied to a single Operating System and can either choose an Operating System of our liking, or we can choose a Programming Language that will allow us to support all Operating Systems at once. However, the only personal computer that is likely to be seen in a setting for the Mini-Mixer is a laptop. This is due to the social settings that accompany uses of the Mini-Mixer. Despite these setbacks, this is a popular choice for client communications in several DIY products similar to the Mini-Mixer.

5.2.4 HARDWARE INTERFACE

The hardware interface components are comprised of a status indicator of some fashion as well as some simple input. The hardware interface is not meant to have significant input and should only cover administrative purposes to keep the machine as operationally simple as possible. The status indicator is to be used to show certain states of the device such as connectivity and mixing state with minor details. The status indicator’s main function is to show a quick-look of the state of the machine and generally let the user know that the mixing device is on and working.

Status Indicator - The status indicator is to work as a way for the user to determine what “state” the machine is in. The status indicators should update in near real-time and be easy to read or interpret. The status indicator should only display relevant information regarding the state of our machine. In our case, we are mostly interested in power, wireless connectivity, and mixing state or progress.

Character Display - Character displays have been in use for a long while and has matured as a technology over time. Due to the large variety of displays of this type, the type of microcontroller used to drive the display was chosen to include a reasonably sized set of this variety; that controller being the Hitachi HD44780 microcontroller. This microcontroller drives a dot-matrix liquid crystal display, and has become a standard in the industry. Many enthusiasts and hobbyists have written multiple libraries to simplify usage of this controller through code. One advantage of this controller is it includes an ASCII character set, leaving less work for the developer. Another advantage of this controller is that it requires as little as 6 lines to control with no additional hardware. With a shift register, this can be reduced to 3 lines with no change in functionality. The HD44780 supports 4-bit and 8-bit modes of operation. Although the 4-bit operation requires less active lines, it also makes programming more difficult. With many GPIO pins available on the embedded server, the 4-bit mode of operation is likely of no use for the Mini-Mixer. The controller supports different LCD sizes, from as small as 8x1 (one row of 8 characters) up to 80x8 (8 rows of 80 characters). Due to the low power supply requirements of this controller (2.7-5V), power consumption is of little concern relative to the rest of Mini-Mixer’s power requirements.

TFT LCD - TFT technology has existed for some time now, and often is used in many consumer electronics. Such a display is primarily used as an electronics functional output, and as of recent, sometimes both input and output. This family of display technology certainly provides the best quality output of all the types of displays researched. However, it comes at the price of increased complexity, as well as a considerable increase of GPIO pins necessary to drive it. Although the power consumption isn’t as high as other parts of the Mini-Mixer, relative to the other display types, it is considerably higher. When considering the primary function of the Mini-Mixer, it can be determined that the output display is not what provides the main functionality; something in which a TFT display is generally responsible for.

Light Emitting Diodes - The LED display family is the most primitive of today’s digital display interfaces. Inherently, a display of this nature is very easy to implement into a project in regard to both the hardware and software aspects. The LED display family ranges from using single LED’s to different types of segmented LED displays (7 segment, 9 segment, etc.). Research was constrained to single and segmented LED displays, because interfacing a more complex LED display (such as a dot matrix display) would be pointless when a

character display could be used instead. The obvious advantage of using single or segmented LED displays is, as mentioned before, the ease of which it can be integrated into the project. Driving single LEDs is trivial, and driving a multi-segment LED display can be done easily through either hardwiring to a microcontroller and controlled via software, or using an IC that can decode data from a source and display it appropriately on the display. Power consumption of these displays is of little concern relative to the rest of the project. As easy and convenient as these displays are, they lack in flexibility. Unfortunately, only small amounts of information can be displayed with single LEDs, and slightly more with multi-segment displays. For this display to be effective, a considerable amount of thought and experimentation as to how output would be formatted would be required. It may even be impossible, depending on the requirements of the rest of the system.

Interface Controls - The interface controls are used by the user to alter the major states of the machine. This would include things such as powering on and off the machine, as well as cancelling or confirming the mixing process. Depending on the approach taken, the interface controls may play a role in activating a “setup” mode, as would be required in certain cases of communicating between the mixer and the client’s wireless device.

Touched-Based Controls - Touch-based control schemes have been immensely popularized by the recent advancements in mobile technology. Due to its user control simplicity, it is a great way to provide input to an electronic device. With its popularity, the time it takes the average user to learn how to use the device would be decreased considerably. Unfortunately, as is often the case, simplifying the user experience results in a more complicated hardware and software design. Interfacing the display to an MCU requires more control lines or additional hardware if said control lines are to be conserved. Device selection would become limited to those that provide libraries that allow for simplified programming, as writing an entire library from scratch would be unreasonable and is outside the scope of this project. Using a touch based interface would almost certainly require it to be embedded into the display, which would require the display itself to be more complex. Likely the screen would be a TFT or similar display, which brings an additional amount of complexity to the project that was described in the *TFT Display* portion.

Switches and Buttons - Switches and buttons are the most basic digital input devices used on electronic devices. Due to this, they are very easy to implement and program in a project. There are a wide variety of switches and buttons, with differences as little as the color scheme and actuation type, to more important features such as functionality. Depending on the power supply used it might not be necessary, but one type of switch that will almost certainly be used for the Mini-Mixer will be a rocker switch, that will control the main power for the Mini-Mixer. Pushbuttons can be used in addition to the display to allow the user to

view any pertinent information the Mini-Mixer can provide. A “keypad” could be created with a matrix of buttons if the design required it.

5.2.5 POWER SUPPLY SYSTEM

Personal Computer Power Supply - There is a general trend in the “make” movement where electronic enthusiasts and hobbyist use an ATX power supply for their projects. This is, generally speaking, a fairly cost effective way to supply a project with the power demands of Mini-Mixer. However, this particular power supply will not work straight out of box and requires some slight modifications. The reason behind this is primarily for safety implementations that most PC power supplies have. One of these is a minimum load requirement, where the power supply will only supply proper voltage levels if a minimum load is supplied (this load generally is the motherboard). Most power supplies will not turn on unless a certain pin from the 24 pin ATX motherboard power connecter is set to ground, which is another factor that must be taken into consideration. Besides these requirements, an ATX power supply would server the Mini-Mixer well, as it provides +/-12V, +/-5V, and +/-3.3V rails; each of these can serve a purpose in the Mini-Mixer. For instance, the 12V rail could be used with the pumps, while the 5V and 3V can be used for the embedded server and embedded microcontroller, respectively.

Custom Design - Although the team consists of two computer engineers, it may be possible to still design the power supply system. However, this would require extra research into the total power consumption, as well as other attributes of each part. The advantage of creating a custom design is it can be built to specifically the Mini-Mixer’s requirements. This can be done in a number of ways, but will likely involve a step down transformer, full wave rectifier, a linear regulator, a few resistors, and a few capacitors. Likely, that circuit would supply the highest rated voltage the project would require, and for other parts that require less voltage, a buck converter would be used. What would be difficult to accomplish in designing a custom power supply is not necessarily satisfying the power requirements for the Mini-mixer, but the different standards and regulations that are in place for such a system. Safety would be one major concern, and would be of high importance during actual research. Some research into other aspects, such as fusing and heat dissipation, would have to be done as well.

5.2.6 COOLING METHOD

The cooling method is to be considered as we would like the mixed drinks to be cold by the time it reaches the user’s mouth. Several solutions have been considered based on the requirements of our project as well as the resources that the average user will have available to them when mixing drinks.

Thermoelectric Cooling - An initial consideration that was researched included a thermoelectric cooling solution, using a Peltier plate with heat sinks and fans. Upon investigating further, some solutions that are typically used for cooling CPUs and other small electronics were considered. [3] [4] [5] These plates run on a 12V source at 5-10 amps which is in the area of our power requirements for our pumps. To meet our power requirement, this would be a very careful choice to make as this type of solution would contribute around ten percent to our total power requirements. The units are typically around 500 grams in weight so the added weight to our unit is minimal which is excellent for our portability requirements. This would also require a custom fitting where air ports will need to be machined and the entire assembly will need to be mounted in a particular way to cool the air inside of the unit.

Refrigerant-based Cooling - The more traditional solution to cooling our ingredients is the use of a refrigerant-based system. These systems are extremely common in home appliances as well as automotive air conditioning. The air conditioning system for this method includes a refrigerant compressor, a condensing element and an evaporating element at the very least. Such miniature systems exist for our size constraints such as the R134a Compressor from Purswave Technology. [6] However, these systems consume anywhere from 250-350 Watts of power on a 12 Volt power supply. This type of cooling system would easily become the heaviest user of power in the entire product. In addition, the compressor, evaporator, and condenser can collectively weigh several pounds when the system is fully charged with refrigerant. As we have strict power and weight limits, this is a major concern. In addition, these parts will need to be assembled separately which would require careful orientation of all components, with the connections requiring a blowtorch to connect the copper endpoints. We would then need to have the system tested and our team has no experience in this area so this is a major issue when considering this solution.

Ice-based Cooling - The most straight-forward solution considered for the project is using ice-based cooling for the drinks. This could take the form of either applying the ice around the drink containers within the machine, or requiring the user to place ice cubes in their cup before or after the mixing process. This would require the user to have a source of ice that is reasonably close to the mixer. In the case of cooling the drinks within the Mini-mixer, the user would be required to have a fairly large amount of ice to fill this entire area. We would also need to consider a drain pan or port for the melted ice. The enclosure design would also have to be considered for this as we would require very easy access for placing and removing ice from the machine. The simplest case using ice-based solutions is to have the user fill the cup with their desired amount of ice before mixing. A couple issues arise out of this. First, the user would have to ensure the drink does not overflow or that they can confirm that the dispensed amount will pour into their glass without issue. We would also need to guarantee an ice source. As our target uses cases are for house parties and general kitchen use, we have a

reasonable guarantee that the user has an ice-maker from their freezer, or is storing ice within their freezer. This method uses much less ice than filling the Mini-Mixer directly as well.

5.2.7 FLUID PUMP SYSTEM

Arguably the most important hardware component of the entire mixing system is the choice of the fluid pumps. There are a wide array of pump types and sizes. Due to monetary and size constraints, we are restricting our considerations to the “micro pump” variety. That is – small pumps that are around the size that can be held with a single human hand and are light, cheap, low-power, and pump on the order of at least a few ounces of liquid per minute.

Peristaltic pump - Peristaltic pumps are a type of fluid displacement pumps that is suited for all kinds of fluids. This is mainly due to the fact that no mechanical parts ever touch the fluid, as the fluid is only ever in the supplied tubing at all times. This has several advantages that allow the fluid to not be contamination by internal parts of a pumping system. The pump works by usually having a spinning roller press against a sub-section of the fluid tubing in a circular pattern, causing the fluid to be pushed out of one end, and collected in the other. This makes the pump self-priming as a low pressure vacuum is created on the pumping side. Small pumps with flow rates around 100 Milliliters per minute are inexpensive and low-powered, typically operating at 12 Volts at around a few hundred milliamps. Another great advantage to these types of pumps are they tend to be very accurate, with error of flow rates in the single percentage range. [7] [8] This makes it very ideal for our project as we can have a large number of these inexpensive pumps in a small and low-powered environment. One major disadvantage is the cost; most pumps tend to increase dramatically if we need to go over a flow rate of 100 Milliliters per minute.

Gear Pump - One popular pump type consideration is that of the gear pump. A gear pump uses circulating gear to create displacement of the fluid. These types of pumps are very popular with vicious fluids, such as oil, and other fluids that you might find in a combustion engine. These pumps are not very accurate – to such an extent that it was difficult to find a geared micro pump that actually specified a rate of flow. Of the pumps that do specify a rate of flow, we typically see the specification in units of gallons per hour, which also speaks to the accuracy and precision of the pumps not being ideal at all. These pumps operate around the 12V range which is ideal. However, the current draw tends to be much higher in the range of 1-2 amps as compared to peristaltic pumps that operate under 1 amp of current. The pumps are also very cheap with a price range around 25 percent lower than what can be found with peristaltic pump offerings. These pumps are not self-priming, which is a major concern as this would require either a gravity-fed fluid input, or priming by the user – both of which will ultimately require more work on the part of the user and makes for a less-than-friendly drink mixer. Additionally, there is a health and safety concern

surrounding cleaning the gearing and cross-contamination of fluids, both of which are major issues when dealing with consumable beverages. [9] [10] [11]

Gas Pump - Gas Pumps are a very popular option in the beverage service industry. Gas-driven pumps are known for robust delivery within fountain drink stations that are seen in nearly every facet of the food service industry. Gas pumps are called as such due to them being driven by a form of gas, usually carbon dioxide. The pumps are self-priming and have flow rates on the order of several ounces per second. The major concern of the pumps with respect to our project is the added necessity of a gas source. This is a very heavy component that adds to both the size and weight of the mixer. This would also require the user to routinely replace or refill the carbon dioxide canister for continued operation. [12]

Solenoid Valve - Another consideration for fluid delivery was the implementation of a solenoid valve to manage the flow of the liquid. These pumps work by opening and closing an electronic valve to start or stop the flow of fluids. The pumps operate in the 12 Volt range with a current draw around 1-2 Amps. [13] The pumps do not tend to be gravity-fed and would require pressurization of the fluids in some form. This of course would require a gas compressor and we are met with some of the same issues that plague the gas-powered pump solution, specifically regarding the increased weight and size of the device as well as the increased maintenance required by the user.

5.2.8 FLUID STORAGE

The fluid storage selection is an important component of how the user interacts and maintains the device. The selection should reflect our goals of being very simple to use while still covering the required functionality of the device. The liquids are to be sourced by the user which means in most cases the user will have bought the beverages and flavoring in bottled-form from a store. Therefore, we must consider storage solutions that interface well with transferring fluids from one container to another. Safety is of most concern with storage; the fluids shall never come into contact with anything other than the pump tubing, to prevent possible contamination. The storages used should be either replaceable or easy to wash and reuse.

5.2.9 FRAME AND ENCLOSURE

The framing is what will house and mount all hardware components of the mixing device. The options available to us are vast in the scope of our size and weight requirements. Similar projects have found creative, and sometimes simple, solutions to house their mixing design. Not only does the framing determine the housing for the components, but the mount points and physical configuration of

all hardware components as well. We would like to consider frames and enclosures that are very simple to assemble and can be modified on-the-fly. That is – we would like the ability to iterate over the arrangement and configuration of our hardware components without compromising the constructions of the frame and enclosure itself. This will allow us to test different approaches to the physical configuration as well as allow room for modifications or adjustments should any unforeseen issues arise. We will consider a few approaches based primarily on the aesthetics, price point, and ease of assembly.

Wooden-enclosed Frame - One consideration for the framing is an enclosed, wooden frame. The main advantage to this is wood being relative cheap and available in many different types. This would be an enclosure matching our dimensions of the requirements specifications using a wood type appropriate for the Mini-Mixer, ensuring that it is food-safe and fire-resistant. We would also want to consider wood types that are not prone to water damage as the enclosure will consist mostly of containers of liquid. One great advantage to using a wooden enclosure is the ability to adjust or refine the design of the enclosure very quickly, as it would only require re-mounting components by driving them into the wood at different locations. Wood is also strong while being fairly easy to manipulate. There are many different types of tools to form wood into virtually any way imaginable. However, any assembly beyond basic geometric shapes is entering carpenter territory, for which our team has no experience. In addition, wood tends to be very heavy and can allow us to reach our weight limit very quickly. We would also have to ensure the outside of the wood is sanded and treated properly to prevent user from hurting themselves with splinters or cuts due to unfinished wood. Although not crucial to the functionality of the system, aesthetics are of concern with a wood based enclosure; much time would be spent to make the enclosure look as professionally presentable as possible. This could potentially take a lot of trial and error testing, which would consume even more time.

Steel Rod Open Layout - Another consideration for the framing is by using an assembly of steel rods or tubing to create a scaffold of sorts to mount all components of the Mini-Mixer. This would bring up an immediate concern of the chilling the mixtures. This would require an external cooler to store the ingredients or the user would have to rely on ice-based solutions or a pre-chilled ingredient. Another concern is keeping the hardware and electrical components protected from the liquid components. When using a scaffold frame, there isn't much room to separate the fluid pumping system away from the electrical components. We would also be concerned about the client's immediate access to the electrical components of the device. Durability is another concern as many fragile components of the Mini-Mixer will be exposed to both the user and the environment around it. The joints to connect the steel rods would have to either be specially machined or 3D-printed. However, the major advantage of this framing is the simplicity and reduced weight of the Mini-Mixer compared to other considerations.

Copper-pipe Open Layout – Similar to the Steel Rod Open Layout, we also have the choice of using small diameter (one half inch) copper tubing to construct and open layout configuration. One major advantage to this configuration is the fact that copper tubing is already a standard use case for commercial and home water distribution systems. This means that we can find standard sizes at nearly every major hardware store. Specifically, we have the advantage of using an array of copper fittings to form our framework with ease. [14] This allows us to iterate over our design at any point of the implementation process by simply running to a nearby hardware store and picking up new parts. This would also be beneficial should we encounter sudden issues or have parts of the frame damaged during the implementation. Copper piping and fittings are very popular so we have the added advantage of these construction materials being relatively inexpensive compared to other considerations. With the respect to aesthetics, the shiny copper finish is a beautiful touch to the look of the project, adding a sharp, elegant look to the design with very little effort. Similar to the framing choices, we have several different solutions at our disposal regarding the containment of the liquids. Most of our pump considerations are self-priming, so we will focus on solutions that do not necessarily provide a gravity-fed input to the pumps. The main concern behind liquid containment is ensuring that we are using food-safe material for the containers. This gives an obvious bias towards pre-constructed containers as we can verify their compliance by using only FDA-approved containers.

Self-contained Ingredients - The use of self-contained ingredients – that is using the container the ingredients are originally purchased in, is an obvious choice that can be considered. This comes with the advantages that the user can just open the container and place it inside the machine. When the user is done mixing, they can just remove the container and place it somewhere for storage with very little work. However, the user would be limited in what ingredients to use by the space limits within the mixing machine. We would also have to ensure our ingredient extraction method is nearly universal for all types of bottles and sizes.

Uniform Containers - Another consideration is using a uniformly-sized container for the ingredients. This would allow for careful selection of the size and materials used to be easy to handle and clean. This would also allow the choice of a container size that uses the maximum amount of space within the machine. A major disadvantage is that the user would have to transfer their ingredients from the purchased bottle to our containers. We would have to ensure our containers allow the transfer of liquid with as little issues as possible. The user may want to save any leftover ingredients so we would require the container to be practical for storage as well. A major design requirement of the containers would be selection of a form-factor and material that is approved by health & safety standards and can handle many different types of liquids with little risk of cross-contamination.

5.2.10 ILLUMINATION

Illumination in the project refers to lights on the Mini-mixer that are primarily focused on aesthetics. This leaves us with a wide range of approaches to implementing the feature. The placement and type of Illumination is going to heavily rely on the choice of the machine's enclosure.

5.3 SOFTWARE COMPONENT CONSIDERATIONS

The software components of the Min-Mixer embody all of the logic required to operate the machine and fulfill the requirements specifications. The major components have been modularized into the Embedded Controller, Embedded Server, and Client System. These are three logical modules that encompass all functionality of the Mini-Mixer. The Embedded Controller is responsible for taking mixing commands and controlling the liquid pumps. The Embedded Server is responsible for handling the drinks database and facilitating a user accounting system. The Client System is mostly responsible for the User Interface for the client to control the Mini-Mixer.

5.3.1 EMBEDDED CONTROLLER

Assembly - Second to machine code, assembly coding is as low level as a developer can get with an embedded controller. Due to this, the assembly code for each embedded controller may differ depending on architecture differences. Nevertheless, assembly provides more control than any other higher level language. Because of this, however, programming becomes more arduous and can potentially take a multiple amount of time longer to implement a process versus a higher level language. Programming in the controller's assembly could introduce a learning curve depending on what is used. Certain factors, such as instruction orthogonality, also can have an effect on how certain systems are programmed. Although it is true that in some cases assembly code can provide much quicker code than a compiler can, with the advancement and improvement of compilers over the last several decades, this is a rare occurrence. For the most part, assembly will only be used if and where absolutely necessary for the Mini-Mixer.

The C Programming Language - Although originally not intended, C is the most commonly used programming language for embedded controllers. Its immediate advantages over assembly automatically make it attractive to the vast majority of embedded system developers. From ease of readability, to pointer manipulation and structure support, C has many strengths. Tasks that would be rather difficult in assembly can be done in a fraction of time with C. Because of how long C has existed, as well as how much it has been used, many compilers exist for it. In other words, the chances of finding a microcontroller that does not have an associated C compiler for it is very, very slim. Due to the team's familiarity with C, it is probable that it will be the choice of language to program the MCU for the

Mini-Mixer. It is relatively easy to also do inline assembly within C code, which brings in any advantages assembly might have.

The C++ Programming Language - While C is the most used language for embedded systems, C++ in the past decade has been gaining ground as an alternative. Because of its object oriented attributes, it allows for easier code structuring and flow than C. Although this definitely improves the workflow for a developer, it comes at the cost of resource management; generally speaking, resources are constrained on an embedded controller. C++, if the developer is not careful, is not as resource conservative as C is. This can cause major issues depending on the application; however, for the purposes of the Mini-Mixer, the issues that would arise would likely not be of much concern. While an object oriented language would be well suited to the task of the embedded server, it is hard to determine whether it would be as well suited for the microcontroller. There might not be much advantage to abstracting the hardware out into its own class, depending on what and how much hardware is being used. For this reason, C++ might be considered as a possible language for the µC, but with preference towards C.

The Ada Programming Language - Unlike C or C++, Ada is an object oriented programming language that was originally designed with the intent to be used for embedded systems. Ada's primary focus is to produce safe and stable code through detecting possible errors during compile time vs run time. For this reason, Ada is often used in critical systems where failure can result with devastating consequences. Ada provides interoperability with other languages, which can be advantageous in multiple scenarios. However, when considering the intended usage for the Mini-Mixer, Ada does not seem as well suited for its purposes as does C. Compared to C++, however, Ada may be a viable alternative. In the event that it is deemed necessary to use an object oriented language for the Mini-Mixer, further evaluation will be necessary to decide between the two.

5.3.2 EMBEDDED SERVER

The Embedded Server can be thought of as the "heart" of the software components in the project. The Embedded Server is responsible for interfacing with the Clients and the Embedded Controller. Since the Client and Embedded Controller do not directly communicate with each other, the Embedded Server is a key component of the entire software system. As the user's only interaction with the system is from the Client's point of view, the Embedded Server will be the endpoint for all commands sent and received from the Mini-Mixer. Our goal is to create a set of API endpoints that can be used by any type of client and does not rely on our choice of client, should we decide to support multiple types of clients or change the client in the future. As the choice of Operating System usually depends entirely on the hardware choice of the Embedded Server, we

have to be careful when consideration platform-dependent languages. For this reason and in an attempt to keep each component of the project as modular as possible, we will be considering languages and frameworks that are as cross-platform as possible. Due to our limited time and development resources, we will only be considering programming languages for which the team has non-trivial experience.

Python with Django Framework - A major consideration of software choice is Python using the Django framework. [15] [16] Specifically, an app built on top of Django called Django REST Framework will be considered. [17] Django REST framework incorporates a RESTful (Representational State Transfer) design approach to API programming. The REST architecture operates on the client-server model which lines up exactly with our project's implementation. The architecture is to be stateless which means that all data required to respond to a request is sent in a single request. This also means that the state of the session is handled entirely by the client, although this cannot be true in some cases (e.g., Client authentication). There are several other requirements to RESTful design, though these are the most relevant to our project. The Django framework itself provides us with an Authentication and Accounting backend which are the most important features that would be used by the framework for our project. The Django REST Framework gives us those advantages of Django along with a robust framework for building a complete API. Part of the team has intermediate experience with this software stack which will be a factor in consideration.

Java with Dropwizard Framework - A similar approach to the Django stack, the Java programming language with an API framework such as Dropwizard is being considered due to the team's familiarly to Java through our academics. [18] Dropwizard is another RESTful API framework that can be used to develop API endpoints for our application. [19]
However, this would require us to either build our own Accounts and Authentication systems, or make use of another third-party SDK.

5.3.3 CLIENT SYSTEM

The client system component of the Mini-Mixer is the remote control of the entire mixing solution. Careful consideration must be taken on this component to cover as many general uses cases for the user as possible. We must also maintain a balance of chosen platform(s) with respect to our collective background knowledge, hardware resources, and time constraints that are present in this project. Of most concern is the time constraints when considering the size of our team and project will force us to choose to focus on one platform to provide the best user experience with the prototype.

Mobile Platform - The mobile and handheld platform is a major consideration for our choice of the client component. However, our hardware choice of a client will

dictate the programming language and SDKs that will need to be used as each hardware vendor is usually tied to a specific language and SDK and in some cases, the platform required to develop with. The mobile platforms are ideal as many devices come with several forms of wireless communication, including Wi-Fi, Bluetooth, and NFC. These communications are almost always supported very well in the form of the SDK for the particular mobile platform. This allow us to explore many different methods of communicating with the drink mixer that suits our needs.

Google Android - The Google Android operating system is a major consideration for our client choice. This would require uses of the Android SDK, which is based on Java. There is an immense amount of documentation, both first-party and third-party, on developing applications for the Android operating system. There is also the added advantage that our team is very familiar with the Java programming language and already has some novice experience in developing Android applications.

Apple iOS - The choice of Apple's iOS for the operating system would constrain us entirely to the Apple ecosystem. This requires an Apple OS X operating system to develop with the iOS SDK. We would also be required to develop our software with the Objective-C or Swift programming languages. Although iOS is one of the two major mobile operating systems, our team does not currently have the resources to purchase and develop for this device.

Native PC Application - A native PC Application was an initial consideration. This opens us to many different software stacks to implement the system. However, native PC applications are generally platform-dependent and would limit us to larger devices such as Desktop and Laptop Computers. We can mitigate the platform dependence by choosing a language that runs on a virtual machine, such as Java, but we would still be limited in the scope of devices that we can support. In addition, the development time required for a native PC application can be as much, or more, than development for a mobile device.

Native Web Application - A Web Application was another consideration for implementing the client interface. This is an application that can be loaded from the server and run in a computer browser using elements of Hypertext Markup Language (HTML), JavaScript, and Cascading Style Sheets (CSS). This would likely require nothing more than a device with a browser. This means that we could cover both mobile and larger platforms with a single codebase. However, this requires extensive experience in this area as the web page would have to be responsive and automatically adjust to many different screen sizes and input types.

5.4 COMMUNICATIONS CONSIDERATIONS

The communications considerations for the Mini-Mixer are regarding the communications hardware and protocols between the distinct hardware components within the Mini-Mixer system. This includes the communication between the Embedded Server and Client system as well as that of the Embedded Server and Embedded Controller. As the Embedded Server and Embedded Controller are going to be physically close to each other as they will be housed within the Mini-Mixer, we will mainly consider physically wired means of communications as this is generally the cheaper and more reliable solution for close-range communications. The Embedded Controller will likely be a type of microcontroller so we will need to have a distinct focus on serial communications between the Embedded Controller and Embedded Server, as this is one of the most straight-forward and most popular means of communications with this type of hardware configuration.

5.4.1 CONTROLLER-SERVER

The communications selection surrounding the embedded controller and server is fairly important as it dictates what hardware to use for the controller and server. This is especially important as our choice of communications will depend on our I/O pin resources we have on both the controller and server. As the method of communication isn't entirely important, due to the devices being close together and data transfer being relatively small and not time-sensitive, we have considered a wide range of interesting mediums to consider for the project.

RS-232 w/ UART – As one of the oldest and most used standards for establishing serial communication, RS-232 is an immediate consideration for Embedded Server to Embedded Controller communications. It has a distinct advantage of being practically ubiquitous amongst most current day microcontrollers, all the while avoiding being resource demanding by only requiring two lines for communication with another device; namely, the Tx and Rx lines. With the UART hardware, most of RS-232 is taken care of. A simple sequential ASCII based command set could be devised and implemented relatively easily using RS-232 and UART, making it all the more appealing.

Bluetooth and Bluetooth LE - Bluetooth and its low energy derivative(LE), is a wireless standard operating in the 2.4 GHz area that is mainly used for short-distance wireless communications. The range of Bluetooth varies somewhat depending on the environment and devices used to implement the standard. However, the range is typically under ten meters from the host. A Bluetooth host can connect several clients at once, so it is sometimes referred to as an implementation of a Wireless Personal Area Network (WPAN). There are many different transport protocol implementations over the Bluetooth Standard. Two of the most popular that we have looked at for consideration are Radio Frequency

Communications (RFCOMM) and Logical Link Control and Adaptation Protocol (L2CAP). L2CAP is a packet-based protocol that allows for guaranteed delivery. It can be altered for best-effort mode, however. The packet size varies up to 64 kilobytes and the packet itself can be formatted to the programmer's requirements. [20] L2CAP offers two flow control modes that resemble that of Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) over Internet Protocol (IP). Enhanced Retransmission Mode (ERTM) is a retransmission mode that gives the protocol a reliable communication channel similar to that of TCP. Streaming Mode (SM) has no retransmission or flow control and is mean to be a "best-effort" protocol, much like that of UDP. It was worth considering Bluetooth with L2CAP transport solely due to the fact that L2CAP has similarities to TCP/UDP for which the team is already familiar. RFCOMM is a protocol specifically crafted to emulate RS-232 serial communications. [21] RFCOMM itself is actually an implementation over the L2CAP layer. As RFCOMM is meant to be a drop-in replacement for serial communications, the implementation is very similar to what you would expect from wired RS-232 equivalents. Our team chose to consider RFCOMM due to our familiarity and experience with wired RS-232 communications. As the total amount of commands required to pass to and from the server and controller is relatively low, RFCOMM can be considered a solid choice as the serial commands would be simple to define and makes the overall communications between the hardware as simple as possible.

Wi-Fi - Wi-Fi is an implementation of a Wireless Local Area Network based on the Institute of Electrical and Electronics Engineers' (IEEE) 802.11 standards [22]. The most popular transport protocol used with the Wi-Fi implementation is the Transmission Control Protocol/Internet Protocol (TCP/IP). This allows us to use a set of standards and protocols that we are already familiar with from developing and using web applications as well as from academic studies. However, a wireless protocol isn't entirely necessary as both the controller and the server are within close proximity. In addition, the commands being transported are very simple and would not require the large amount of overhead that comes with designing and implementation a communications solution over Wi-Fi. We would also have to ensure that both the server and the controller implement compatible Wi-Fi hardware modules, which is particularly concerning for the embedded controller as this would consume many digital pins on the board.

Serial Peripheral Interface - Serial Peripheral Interface (SPI) is another serial data protocol primarily used for communication between microcontrollers and other embedded devices. [23] The protocol itself is a de facto standard so vendors may choose to implement the protocol differently. The protocol is synchronous and is configured in a master/slave format. The designated master can communicate with several slaves in a synchronous format. Communication between the master and its slaves is done using a Slave Select (SS) line for each

slave. This ensures both the direction of transfer and the selected device that data is being transferred.

5.4.2 CLIENT-SERVER

The client-server communications are the means of communication between the client interface and the embedded server. To meet our connectivity requirements, the communications will need to be wireless and have a range that is suitable for our specifications. This will mostly constrain us to the popular wireless communications implementations as the Client System will be mobile and physically detached from the Mini-Mixer. We will also be limited by the common wireless configurations that are offered in the popular mobile hardware components. We will consider two of the most popular methods of wireless communication, as their hardware availability, maximum range, and data throughput meet or exceed our requirements specifications.

Bluetooth and Bluetooth LE - Bluetooth is again considered as a means of communication between our hardware components. In this case, the range (~10 meters) of communication meets our requirement of being able to control the Mini-Mixer from a distance. This would require the design and implementation of serial commands between the client and server. As the server would be hosting several different resources such as user accounts and custom mixes, it may not be ideal to use this method of communication due to the complexity of the commands required.

Wi-Fi - Wi-Fi is another communication method to consider when dealing with client-server communications. With this, we can simply use existing TCP/IP protocols with HTTP requests to deliver content. In this manner, the Embedded Server can be treated as a normal Web Server and the client as any Internet-capable device. This is the traditional method of client-server configurations as HTTP is designed specifically for this type of communication. This would also open up the possibility of being able to remotely configure the Mini-Mixer from the client device itself. This would require the use of Wi-Fi P2P mode between the server and client. However, this would eliminate the difficulties in setting up the Mini-Mixer in different network environments. This is a major advantage to the user and brings Wi-Fi to the forefront of our considerations.

6. RELEVANT STANDARDS

The relevant standards in the context of our project are something to consider as these underlying standards are what dictate the construction, features, and operation of the hardware and software components that we will be considering. While our project is not directly affected by these standards for the most part, we do need to be aware of them and how they interact with the various components

of the Mini-Mixer. The standards of most concern to us would be the health and safety standards surround the storage and dispensing of consumable liquids. This is an area that we are more or less directly responsible for ensuring we are within compliance as there are legal implications to not conforming to these standards if our prototype were to become a consumer product.

6.1 HARDWARE

When dealing with mechanical pumps and the rubber tubing required to pump and carry liquids through the pump itself, we must consider a standard that has been created specifically for those situations. The ASTM D2000 standard is used to standardize rubber products in automotive applications. [24] In our case, the automotive application is the fluid pump with the rubber product being that of the rubber tubing. This standard is focused on testing methods for the integrity of the material in various environmental conditions, including compression, extreme temperatures, tension, stiffening at low temperatures, among many other scenarios. We will need to ensure that the tubing we use to interface with the mechanical pumps will comply with this standard. The Mini-Mixer will contain several hardware components that will need to be assembled on a Printed Circuit Board (PCB). Because of this, we must consider the standard that accompanies the designs of PCBs. In this case, this is the IP-2221A standard. [25] This is a generic standard that details everything surrounding PCB design. This includes materials, general requirements, physical properties, electrical properties, thermal management, assembly, and quality assurance among many other topics. We will have to keep this standard in mind when designing our PCB(s).

6.2 SOFTWARE AND COMMUNICATIONS

The Mini-Mixer will consist of a number of different hardware components that will need to interact with each other in some way. Some hardware components are a large distance away so even wireless communication standards will need to be considered. One wireless standard we will have to consider when designing the Mini-Mixer is the IEEE 802.11 set of standards. [26] These are a complete set of specifications and standards that define a wireless local area network (WLAN) over a specific set of frequencies. It is crucial to consider this standard if we are going to be communicating using hardware equipment over the bands that IEEE 802.11 specifies. Another standard to consider is that of the Bluetooth Core Specification. [27] This standard defines the protocols and physical layers that are behind the Bluetooth technology. This would typically be an alternative to the Wi-Fi set of standards when considering this for the Mini-Mixer. For physical communications, we must consider the RS-232 (EIA-232) standard for serial communications. [28] This is the basis of most serial communications, defining the protocol and transmission of data in point-to-point connections.

6.3 FOOD AND SAFETY

The Mini-Mixer contains a number of components that will be used to handle various stages of mixing liquids. These liquids are going to be almost immediately consumed by a user. For this reason, we have to consider Food handling standards that define the requirements and specifications for the equipment that will be handling these liquids. The first standard to consider is NSF International Standard/American National Standard 170. [29] This standard is essentially a glossary of terms that defines food equipment terminology that is to be found in subsequent standard specifications. This is important as it serves as a definitive reference for any terminology used in this family of standards so that there is no ambiguity. Another standard to consider is the NSF/ANSI 2 standard. [30] This contains a set of requirements detailing the materials, design, and assembly for equipment that handles food products. This is the main standard to consider when handling our liquids in the Mini-Mixer. Specifically, we must ensure that the materials used from the containers, tubing, connections, pumps, and dispenser are all within compliance of this standard. We will also need to ensure that that the Mini-Mixer complies with some of the cleanliness requirements within the standard.

7. DESIGN DETAILS

In this section, we describe the design details of the implementation of the components of the Mini-Mixer. This includes all hardware components as well as any software components that compliment them. Each component of the hardware and software system was carefully chosen to meet our design goals through the requirements specifications. We also took careful consideration of the relevant standards of each component as well as the design constraints that have been imposed on us in the context of this project.

7.1 HIGH-LEVEL OVERVIEW

The design of the Mini-Mixer system has been made to be a modular as possible with each component having a specific purpose in the system. Ideally, we would like the entire system to follow the design principle of modular systems, where specific modules can be switched out, upgraded, modified, or replaced with little to no effort necessary. This essentially means that the components choices of one module should not impact that of another module, given the inputs and outputs of each module still match. This design approach allows for a cleaner design with the ability to iterate or modify the design in the future, including making it hackable by the user. One of the most important aspects of this choice of design is the ability to parallelize the development of the prototype using the resources of our team. The Mini-Mixer will comprise of four major subsystems:

- Embedded Controller System

- Embedded Server System
- Client System
- Liquid Controls System

The Embedded Server system is essentially planted in between all of the other systems so we will review this first as a basis for the other major components. The Embedded Server is responsible for controlling and communicating between the other major systems, namely the Client System and Embedded Controller System. The Embedded Server is responsible for storing all user accounting information as well as available ingredients and all drink mixes created by the users. The Embedded Server will have serial communications functionality for the wired communications between itself and the Embedded Controller. These communications are responsible for sending the ingredients list along with the proper ratios to the Embedded Controller for further processing. The Embedded Server will also play a key role in the Setup process of the Mini-Mixer. The Server will need to maintain the state of connectivity between both the Client(s) and the Access Point. The Server will also be responsible for updating the Temperature status of the drink container by utilizing a temperature sensor connected to one of its input/output ports. The Client System encompasses the client device(s), software, and connectivity used to communicate with the Mini-Mixer. The Client System will use a popular wireless communications standard to both setup and use the Mini-Mixer. The Client System will include the Client Interface that will be used to control and command the Mini-Mixer. Using the Wireless transceiver on their device, the user will be able to connect to the Mini-Mixer through the provided Client Application on their device. The Client Application will have a Login Interface used to authenticate with a user's given credentials and be greeted with their personal Drink Menu. At the Drink Menu, the user will be able to create, edit, remove drinks. The Drink Menu will include a status indicator of the Mini-Mixer for various things such as mixing state and connectivity. The Embedded Controller System is responsible for the actual dispensing of the mixed drink. The Embedded Controller System will have an embedded microcontroller used to control the pumps for dispending liquid. There will be 6 pumps – one for each ingredient in the Mini-Mixer. Each pump will be connected to their individual liquid containers through food-safe tubing. Each pump will be connected in one direction to the microcontroller. Each pump will also be connected to a half h bridge; this half h bridge is connected to the pump, the microcontroller, the 12V power supply. The microcontroller will have a serial transceiver which will be interfaced with the serial transceiver of the Embedded Server. This will serve as a means of communication between these two components. The Embedded Controller is expected to receive instructions for each liquid pump which includes how much liquid is required from each fluid container for the drink mix. The Embedded Controller will then determine the optimal sequence of pumping and begin the mixing process. The embedded controller is expected to report back to the Embedded Server when each pump has begun and completed the pumping process. From the user's point-of-view, the usage of the Mini-Mixer will be straightforward. The user will handle all

controls of the Mini-Mixer from within our provided mobile application. At first use, the user will simply need to connect directly to the Mini-Mixer and will be guided through a fairly quick setup mode, where the setup is very similar to what users encounter when setting up a home router or connecting to a new secure Access Point. Once done, the user can connect back to their normal Home Access Point and begin using the Mini-Mixer. The user will only need to login with the mobile application and immediately have the ability to view and modify their own personal mixed drinks. When the user needs to modify the drinks within the Mini-Mixer, they can enter a settings mode where the current (if any) designated drinks for each slot will be presented.

7.2 HARDWARE COMPONENTS

The hardware components of the Mini-Mixer are divided up into several modules. The Embedded Controller, Embedded Server, and Client System will more or less cover all hardware required to implement the software of the Mini-Mixer. These systems can be thought of as the hosts for the logic of the Mini-Mixer system. The remaining hardware components are external to these devices and help provide the functionality needed to implement the requirement specifications. To get an idea of how all of the hardware components fit together to form the Mini-Mixer, a hardware block diagram has been constructed in Figure [7.1] to illustrate the associations between the different hardware components being considered.

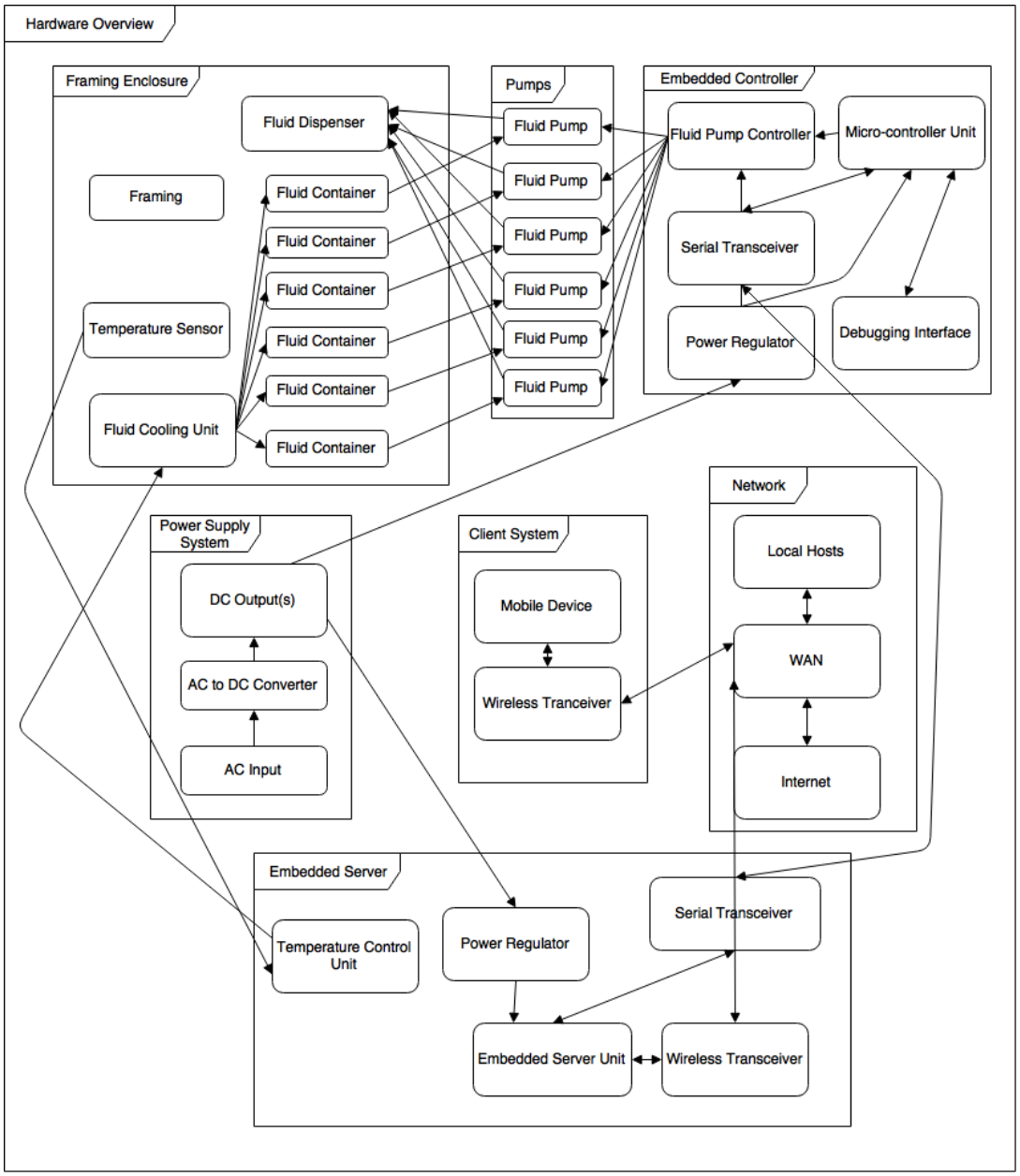

Figure [7.1]

At its core, the Mini-Mixer is comprised of six liquid pumps that will be driving the mixing process. These pumps will be driven by an Embedded Controller that will receives instructions and send commands to the pumps during the mixing process. The Embedded Controller will receive commands from the Embedded Server. The Embedded Server will store the drinks database as well as the user accounts associated with them. The Embedded Server is responsible for facilitating the communication between the Embedded Controller and the Client System. The Client System is a wirelessly connected mobile device that will be used as a User Interface for the client using a mobile application.

7.2.1 EMBEDDED CONTROLLER

The embedded microcontroller of choice for the Mini-Mixer is the ATMega328P, which provides many GPIO pins and known hardware support due to it being used for the Arduino Uno. UART will be used for communication between the BeagleBone. This only requires two pins to be used; namely, pins 2 and 3 on the microcontroller (Rx and Tx). 6 pins of the 328P are capable of providing a PWM signal to the peristaltic pumps; this works out well since 6 peristaltic pumps are used in the Mini-Mixer. Providing power to the 328P will be done by only utilizing the 5V rail from the power supply, with a decoupling capacitor on the power inputs to help mitigate any electrical noise. The schematic for the controller was designed in EAGLE, and the PCB layout of the controller was designed in ExpressPCB. Pictured in Figure [7.2] is the schematic of the controller, and the PCB layout in Figure [7.2.1]:

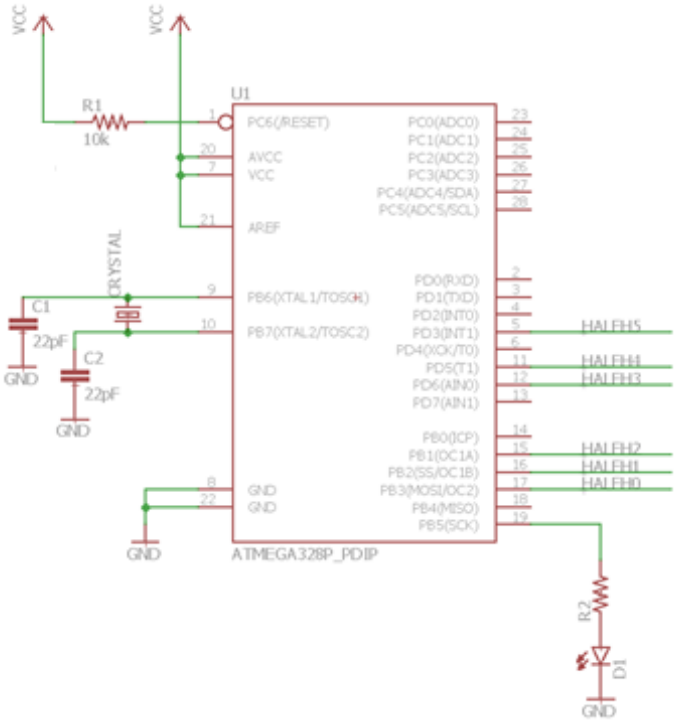

Figure [7.2]

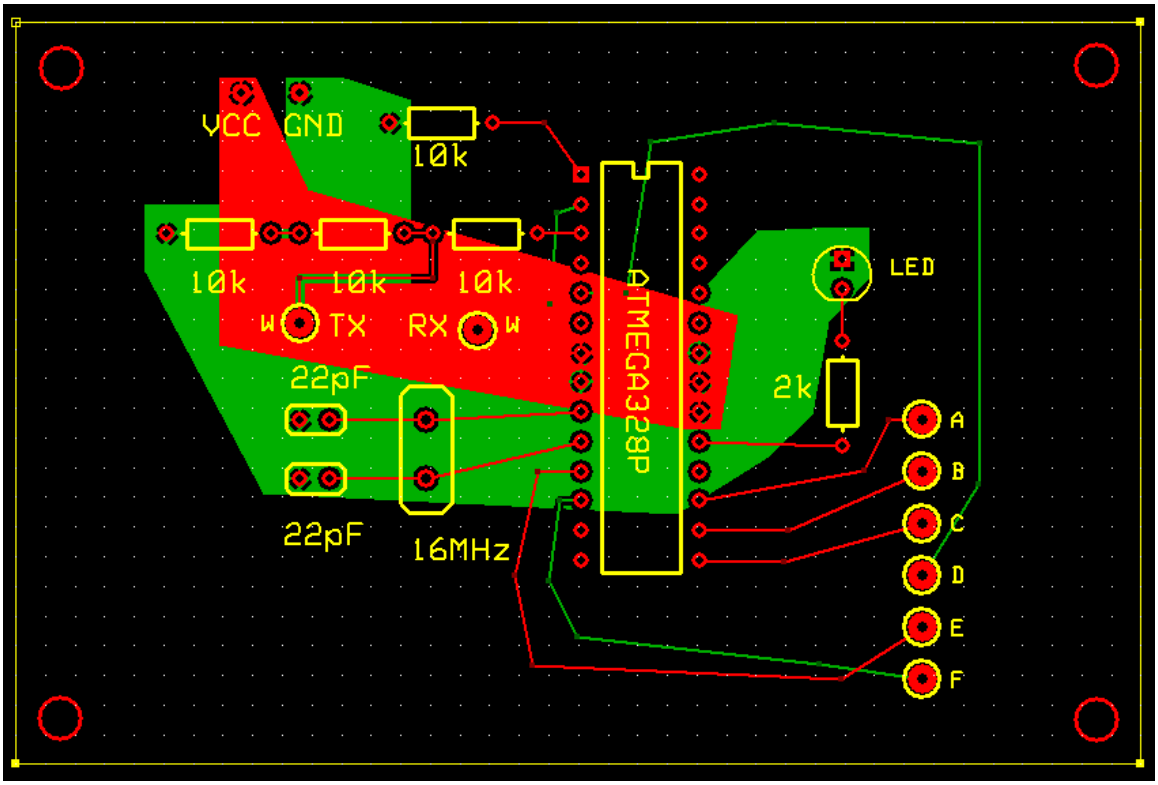

Figure [7.2.1]

The hardware used for the 328P is as follows:

- Atmel ATMega328P
- 16 MHz crystal
- 2x 22pF ceramic capacitors
- 10kΩ through-hole resistor
- 100Ω through-hole resistor
- Green 5mm LED

An external crystal was chosen instead of utilizing one of the internal clocks in the 328P due to the reason that the internal clock is much less accurate than the crystal. All three internal clocks are also necessary to use all 6 analog pins to drive the pumps with. The two capacitors are equally as important, as they are required to provide a load for the crystal. The range of these capacitors is 12-22pF, and the value used **must** fall in that range; otherwise, the 328p will not operate at the expected 16MHz and will potentially affect certain components (e.g. UART). The 10kΩ resistor is tied from VCC to an active low pin on the 328P to prevent the controller from resetting itself continuously. The 100Ω resistor is used in conjunction with the LED to provide a visible status on the state of the controller (state being ‘On’ or ‘Off’). A PCB was designed for all three SN754410’s; the PCB layout is pictured in [7.3]:

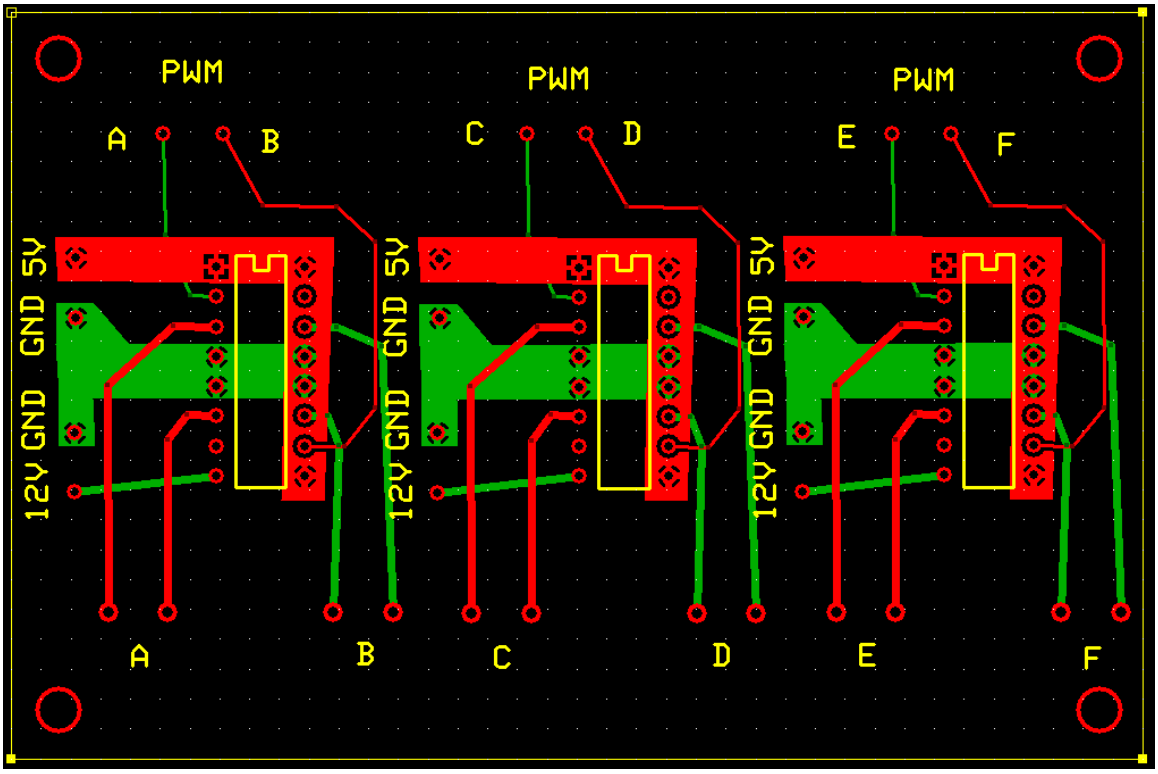

Figure [7.3]

A simple voltage divider is also used on the output of the TX pin for the 328P, because the Beaglebone works with 3.3V high TTL logic as opposed to 5V for the 328P. Therefore, that voltage divider simply “steps down” the voltage to an acceptable level. A small aside: Although 5V would work with UART communications, it is important to acknowledge there isn’t a lot of current involved. If there was more current involved, the Beaglebone could potentially be damaged beyond repair. Please keep that in mind if you try to connect a 5V (or higher) device to a Beaglebone Black!

7.2.2 EMBEDDED SERVER

The BeagleBone: Black (referred to as BeagleBone throughout this section) was decided to be used as the embedded server for the Mini-Mixer, with its open source advantages. It will be running the pre-installed Debian distribution; this distro was chosen due to it being the default operating system on all BeagleBone’s, and due to its support, is likely one of the more stable operating systems available for it. The status indicator hardware will be controlled with several of the BeagleBone’s GPIO pins, while communication to the microcontroller (as previously stated) will use two GPIO pins.

7.2.3 CLIENT SYSTEM

Mobile Device - The mobile hardware device of choice is any device running the Android Operating System. This is due to the team having extensive hardware resources for this OS. This includes current popular mobile phone models, previously older popular phone models, newer mobile tablets, as well as older mobile tablets. Due to almost all devices that run Android having Wi-Fi capability, supporting such devices as inputs to the Mini-Mixer will be fairly straightforward. The app will be designed to not be resource intensive, which will help support older phones and laptops. An approach will also be taken to keep the size of the app to a minimum, once again appealing to a wide variety of users that may have minimal amounts of space on their devices.

Personal Computer - The Personal Computer will likely be supported with the Mini-Mixer by means of using a modern internet browser (i.e. Firefox, Chrome, SeaMonkey, etc). Using standard internet technologies and languages, a wide variety of modern browsers will be supported. Inherently, any computer capable of using said modern internet browser will be supported by the Mini-Mixer.

7.2.4 HARDWARE INTERFACE

Interface Controls - All of the user interaction with the Mini-Mixer will be done via client device running the mobile app. The only major requirement of the client device is that it must run a newer version of Android, and it must have Wi-Fi capabilities.

7.2.5 POWER SUPPLY SYSTEM

It was determined with the power consumption of the Mini-Mixer that a 500w switching power supply would be the best choice. The Mini-Mixer utilizes the Thermaltake TR2 TR-500 500w ATX computer power supply. The power supply contains 12V, 5V, and 3.3V rails that are each capable of supplying sufficient amperage for each component in the Mini-Mixer. From the perspective of the user, the usage of this power supply makes it convenient for repairs in case the power supply goes bad. All electronic components in the Mini-Mixer utilizes standard 4 pin Molex power connectors to interface with the PSU, which allows for easy replacement if necessary.

7.2.6 COOLING SYSTEM

The cooling system will consist of a small air-to-air thermoelectric system powered by a Peltier plate. This system will be acquired and is expected to run on 12 Volts and consume anywhere from 70-90 Watts. A 5000 RPM fan is attached to the cool side of the plate, to move the cool air down towards the bottles. The Embedded Server will control and monitor this cooling system,

ensuring it keeps the contents of the Mini-Mixer within acceptable temperature ranges.

7.2.7 FLUID PUMP SYSTEM

The fluid pump system will be implemented using 12 volt peristaltic pumps capable of pumping 500 milliliters per minute with impressive accuracy. The exact accuracy of the particular pumps sourced is not specified, but similar sized pumps used in DIY solutions such as the Bartendro cite an accuracy of less than 10 milliliters for a single serving drink. This lines up directly with our requirements specifications and we hope to achieve the same or better results with our own peristaltic pumps. The pumps have an estimated current draw of around 500 milliamps which is well within our power requirements for the Mini-Mixer. The pumping system will consist of 6 peristaltic pumps – one for each ingredient. As mentioned in the research section, these pumps have the inherent advantage of avoiding ingredient contamination by never coming into contact with the ingredients themselves. The pumps themselves are Pulse Width Modulated (PWM) to control the flow of fluid. We aren't very concerned with the variable rate of flow when starting the pumping process, though this may come as an advantage when stopping the pumps to avoid splashing of a full drink cup by gradually slowing the pump down with PWM. Each pump is driven using a Half-H driver; these drives will be provided via three TI SN754410 ICs. These devices support a PWM input, which is what several of the GPIO pins on the embedded controller are used for. Safety diodes are built into the integrated controller, which will help prevent damage to the IC. Each half-h bridge supports up to 1000 milliamps of constant current, which satisfies the 500 milliamp requirement for the peristaltic pump. A PWM duty cycle of roughly 70% is used for the pumps in the Mini-Mixer; this provides output flow high enough to meet the requirements of the Mini-Mixer, while at the same time avoiding turbulent output that can possibly cause a mess.

7.2.8 FLUID STORAGE

Fluid storage is important to the Mini-Mixer, as the containers used must fit within the space constraints. The containers themselves must be safe to use, to conform to all food safety standards. Taking these factors into consideration, it was decided that 14oz Rubbermaid polyethylene wash bottles will be used. These bottles are translucent, which make it easy for the user to determine fluid levels in the Mini-Mixer. The material used in the bottle is safe; the same material is used in many consumers bottled beverages and such. The caps on the bottles are drilled out, allowing the pump tubing to reach the bottom. This cap makes it easy to secure the tubing in place, while also providing a convenient way to refill and/or wash the container after usage. The Mini-Mixer has a designated compartment that houses each of these containers; this compartment is located

in such a manner that allows each container to sit securely yet be easily refilled when necessary.

7.2.9 FRAME AND ENCLOSURE

A full ATX computer case is used as the enclosure for the Mini-Mixer, keeping it within the required size constraints and weight, while providing a clean and sleek appearance. The specific case used for the Mini-Mixer is the Thermaltake Spedo Advance, with its dimensions being 21.1 x 9.1x 24.0 inches. Some parts of the case were modified in order to better suit the needs of the Mini-Mixer; for instance, the hard drive cage was removed, and the front panel of the case was drilled to allow for the spigot to pass through. A custom pump rack was built out of aluminum in order to securely mount the pumps in the case. To protect the power supply of an accidental leakage, the backing of the pump rack was covered in vinyl. The various electronics (PCBs, Beaglebone Black, etc) were first mounted to protoboards via adhesive standoffs made to fit in 1/8inch holes. From there, the back of the protoboard was covered in long strips of double faced tape in two stacks. The usage of the protoboard was done to prevent any possible grounding issues with the electronic component and the enclosure. The 5 ¼" drive bay was left in the case, and used to mount the AC unit in the Mini-Mixer with tie wraps. The liquid containers were placed in the front of the case, and securely placed with Velcro backing. Tie wraps and electrical taping were used to route the cables and reduce tension on power cables to avoid any accidental component removal.

7.2.9 ILLUMINATION

The illumination is the addition of lighting on the Mini-Mixer as part of an added aesthetic effect and to make the Mini-Mixer appear more interactive and animated. The illumination will be dynamic and controlled by the Embedded Server. The Embedded Server will be responsible for changing the state of the illumination depending on the mixing state of the Mini-Mixer. A combination of LED's and cold cathode tubes will be used to provide this lighting experience. Strips of amber LED's will be lined around the Mini-Mixer, giving a distinctive, ambient feel to it. Our group has chosen to illuminate the Mini-Mixer using RGB Waterproof Flexible LED Strip Lights. [33] The lights will be placed on the bottom trim of the Mini-Mixer. The lights will be illuminated when the machine is turned on. The Mini-Mixer will flash the lights during a mixing process to have the machine appear more animated. As the exact length of LED strips required for the Mini-Mixer can vary, depending on the exact dimensions of the enclosure as well as the power requirements of the strips themselves, a robust PCB has been designed to facilitate this. The PCB is a general purpose MOSFET driver board with a maximum current of 5 Amperes as allowed by the thickness of the traces. Pull-down resisters were added to the gates of the MOSFETs to prevent a

floating ground condition. Figure [7.4.1] shows the schematic of the MOSFET board.

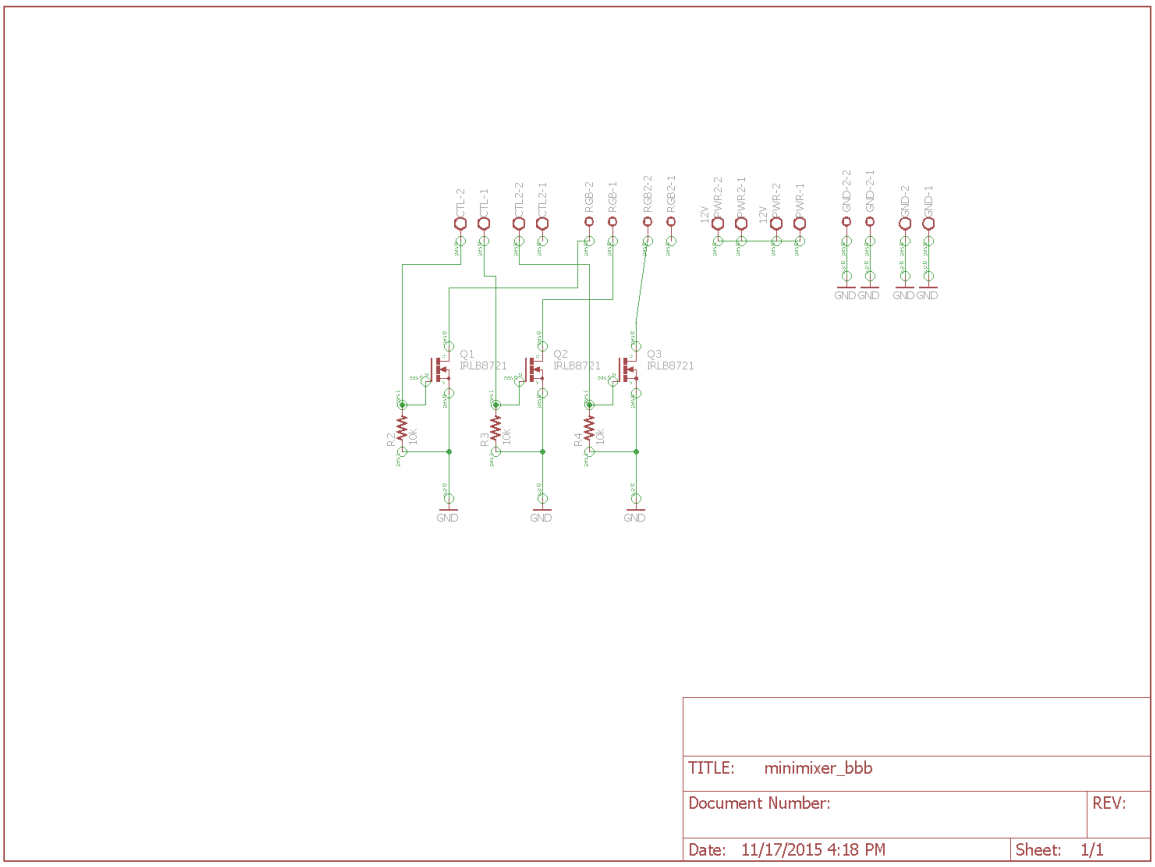

Figure [7.4.1]

The breadboard design is a two-layer PCB with traces thick enough to allow up to 5 Amperes of current along each trace. The breadboard is shown in Figure [7.4.2]

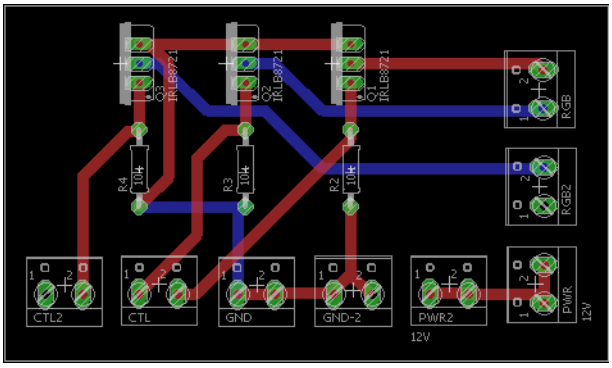

Figure [7.4.2]

7.3 SOFTWARE COMPONENTS

The software components are the logic behind the Mini-Mixer and are what power the features of the Embedded Controller, Embedded Server, and Client System. As such, the actual software driving these components are divided are modularized for each component. The overall design of these components revolves around the client-server software architecture. The client would be the Client System and the server is the Embedded Server. Nearly every user input/output is carried out on the client-side of this setup and the server is used to handle the rest. The notable exceptions to this are the actual dispensed drink, the user's installed ingredients, and the hardware reset and power buttons on the Mini-Mixer.

7.3.1 EMBEDDED CONTROLLER

The embedded controller was programmed using the C programming language. This language was chosen due to the high availability of resources on the MCU as well as being a higher-level language than the alternative. This is important as the software needs to precisely control several different pumps at virtually the same time, while also communicating with and accepting commands from the Embedded Server. The commands sent from the embedded server to the embedded controller will be sent serially via UART. The embedded server will generate a request that the controller will parse and then control the pumps in the intended manner. A top-down approach was taken into the design of the software for the embedded controller. This design approach was selected because it synthesizes a large portion of other high level sub-systems into a functional higher-level system with a systematic, programmable backend. As previously stated, there are 6 pumps; each pumps will be differentiated from each other by use of a single character letter. For ease of usage, these letters will be the first 6 letters of the English alphabet: "A", "B", "C", "D", and "E". There are two modes of dispensing that the Mini-Mixer will use; namely, a "parallel" mode and a "sequential mode". As the names suggest, the parallel mode runs two or more pumps at the same time, while the sequential mode runs at one pump at a time. This is advantageous for two reasons; one reason being that having pumps running at the same time with the parallel can help cut down the time it takes to make the drink (depending on the recipe, the time cut can be significant). The other reason is that with a sequential mode, certain recipes that call for each ingredient to be poured one at a time to get the "layered" color effect can be done. The Mini-Mixer will utilize both modes in a sequential manner, first dispensing all ingredients that will run in parallel, and then dispensing all ingredients that will be done sequentially. This design decision allows for the advantages of both while providing an easy yet effective manner to implement said design. Each command that is sent from the embedded server to the controller is simplified down to a single ASCII character, for ease of implementation and debugging purposes. Every drink request issued from the

embedded server to the embedded controller will always begin with an '***R***' command, representing a new request. From there, two numbers will be sent; each indicating the amount of each ingredient that corresponds to the two pump modes. From there, each ingredient and its amount will follow sequentially, where the ingredient will correspond to the alphabet character labeled pump responsible for dispensing it, followed by a float corresponding to the amount of each ingredient (measured in ounces). After as many pumps and their associated ingredient amount as specified parallel pumps has been supplied by the embedded server, the same process is done for the sequential ingredients and pumps. The general form of an instruction looks like the following:

Request Command	# of Parallel Pumps	# of Sequential pumps	Pump Label	Amount (Oz)	.....	.....	Pump Label	Amount (Oz)	.....	.....
'R'	uint	uint	char	float	.....	.....	char	float	.....	.....

An example drink mix request from the embedded server to the embedded controller would look like the following:

Request Command	# of Parallel Pumps	# of Sequential pumps	Pump Label	Amount (Oz)	.....	.....	Pump Label	Amount (Oz)	.....	.....
R	2	2	B	1.1	C	0.2	A	0.8	F	2.3

This sequence of commands would start a drink request with two ingredients, "" and "C", dispensed in parallel, while the other two ingredients are sequentially dispensed in the order in which the commands were issued from the embedded server. Following the top-down design pattern, we observe what happens during each dispense mode for a drink mix request. The parallel dispense mode has a set containing a various number of pumps, each with their own amount of a corresponding ingredient. The approach taken for dispensing is to let each pump start at the same time, and periodically check to see how long each pump has been running for. The duration of time the pump has been activated will roughly correspond to the amount of liquid dispensed in a linear relationship. During initial setup, using each amount of ingredient to determine an approximate running time for each pump, float variables that correspond to end times will be set that will be used for the previously mentioned periodic checking. Once an end time has been met or exceeded, the associated pump will be shut down. The sequential mode for dispensing is more straightforward; each pump will operate for the duration corresponding to the amount of the ingredient said pump is dispensing. As mentioned previously, it will be done in the order in which the commands were issued from the embedded server. The pumps will be represented in the controller's software as structs named "Pump". Each pump will contain a char corresponding to the pump's label, a float for the amount of

ounces it will dispense, and a time that represents how long the pump is to run for. During the initialization for the parallel mode, an array of Pumps will be created to store each Pump associated with the parallel mode. This will allow for ease of management and control, as well as reducing the length of code required. Time is accurately kept track of by utilizing one of the three internal timers in the code; however, this required that the same timer be multi-purposed in that it would also generate the necessary PWM signal required to drive two of the pumps. Two of the three timers are 8 bit, and the other one is 16 bit; for the Mini-Mixer, the 16bit timer was configured to operate in the same manner as the other 8 bit timers for uniformity.

7.3.2 EMBEDDED SERVER

From a high level perspective, the Embedded Server will be responsible for taking a user created recipe (or a recipe already present on the server), and converting it into a set of simple ASCII character commands that will be sent to the Embedded Controller via UART. Bidirectional communications will happen between the two to ensure proper operation. The Embedded Server will also be responsible for controlling and monitoring the temperature for the thermoelectric cooling system in the Mini-Mixer. This is an important subsystem, as it keeps the ingredients cool enough to not go bad. The Embedded server will also control the various LEDs in the Mini-Mixer, important in relaying status information to the user physically. Much of the power of Debian Linux will be leveraged to handle all of these subsystems together with just the Embedded Server. The Embedded Server will be implemented using the Python programming language with the Django REST Framework on the Linux operating system. Our team is most familiar with Python so this was a natural fit for the Embedded Server. We also have the advantage of the robust standard library that comes packaged with Python. There is also the extremely popular Python Package Index repository which supplies an easy-to-use tool to download and install third-party Python libraries and applications. The Django REST Framework was chosen as we have decided to use a client-server configuration with the traditional request-response lifecycle. The Django REST Framework allows us to implement a clean API using the best programming practices outlined by the REST methodologies. We also have the huge advantage of the Django base framework which gives us a structured MVC architecture to build out our application in the most modular way possible. The Django framework also provides us with built-in administration and accounting features, so that we can easily define our accounting scheme and have an account management interface ready-to-go. Should we choose to later pursue other platforms or multiple platforms for the client system, our Embedded Server will require almost no additional development as is granted by the use of the REST methodologies.

Service Discovery and Wireless Configuration – The Mini-Mixer will implement service discovery using the Zero-configuration networking (zeroconf)

standard. A Python third-party library simply named *zeroconf* will be used to implement service announcing over both Wi-Fi P2P in Setup Mode and the normal Wi-Fi WLAN connection in the Normal Mode. [34]. The actual configuration of the wireless interface on the Beaglebone Black is somewhat complex so a popular Linux tool called *iw* will be used to configure the wireless interface for both Setup and Normal modes. [35] The *iw* tool will be accessed directly from a Django app using Python's built-in *subprocess* module to run *iw* by calling the Linux command. The *subprocess* module allows us to run Linux commands safely while capturing exit status codes and error messages from standard output to handling directly within the Django app. As the Embedded Server will be implemented using RESTful API design, we can describe the entire functionality of the Embedded Server by describing each API endpoint. The calls to the REST API will be made by the client using a uniform resource locator (URL) over HTTP when connected to the same LAN as the Embedded Server. The API can be described using the uniform resource identifier (URI) as RESTful design is dictated around defines "resources" and the actions that can be applied to them. The API will be described by defining each resource as a URI and describing all methods and their function for each resource. An entire resource will be defined in requests and response using the standard JSON format. [36] JSON is a specific Internet Media Type [MIME] that helps to standardize the file format between communications on the Internet. [37] The official MIME type of JSON is known as "application/json" and is what we will be using for all requests and responses that require the transmission of resources. [38]

/connection – The connection resource defines the Mini-Mixer's wireless connection configurations. The connection resource properties are defined in Table [7.1] as shown:

Property Name	Value	Description
id	Integer	A unique id of the connection configuration.
networkName	String	The SSID of the network, if applicable.
securityType	String	The type of wireless security being used. Can be WPA or WPA2.
password	String	The pre-shared password for the connection, if applicable.
networkType	String	The type of network connection. Can be p2p or wlan.
isActive	Boolean	Defines whether the given network is the currently connected network.

Table [7.1]

Methods are provided for the Client to view and change the configuration as needed. The available methods are as follows:

Connection: get – This method will be used to retrieve the current wireless connection settings (if any) of the Mini-Mixer. This will typically be used to display the current connection details to the user while in Setup Mode. The HTTP request may look like the following:

GET /connection/details/{Connection id}

The optional property {Connection id} can be the id of a Connection resource. If no property is specified, then all connections are returned in the response body as an array of Connection resources.

Connection: create – This method is used to create a new connection configuration. The HTTP request looks like the following:

POST /connection/add

The POST request body should contain at least one Connection resource to be added. The HTTP response code will return 200 if the configuration was successfully saved and a HTTP response code 500 otherwise.

Connection: edit – This method provides the ability to edit current connection configurations saved on the Mini-Mixer. The HTTP request looks like the following:

PUT /connection/edit

This method simply requires an existing Connection resource in the request body. The Connection resource id is required and all other parameters are optional, depending on the data to be updated.

Connection: delete – This method gives the ability to remove Connection resources from the wireless configuration of the Mini-Mixer. The HTTP request looks like the following:

DELETE /connection/delete

This request requires a request body with a Connection resource id. The HTTP status code will return 200 if successful. If the specified configuration is active or does not exist, the HTTP status code will be 500.

Connection: set – This method is used to set the new configuration of the wireless module on the Mini-Mixer. The HTTP request looks like the following:

POST /connection/set

The request body should contain a Connection resource with at least the Connection id property. This method will return a HTTP status code 200 if the specified Connection resources exists and will immediately set the new connection. If the connection does not exist, a HTTP status code 500 will be returned and the Mini-Mixer will keep the current wireless configuration.

/user – The user resource describes the user accounting methods of the Mini-Mixer. This includes authenticating to the Mini-Mixer with an existing account, creating a new account, requesting basic profile information as well as updating that profile information. The user resource properties are defined in Table [7.2] as shown:

Property Name	Value	Description
id	Integer	The user's unique id.
username	String	The username of the client.
password	String	The password of the user.

Table [7.2]

The available methods are as follows:

User: register – This method is used to register new users with the currently connected Mini-Mixer. The HTTP request is as follows:

POST /user/register

The HTTP request requires a User resource with at least the username and password. A 200 response will be returned upon successful creation along with the User resource in the response body. If the username already exists, a 500 HTTP status will be returned.

User: login – This method is used to authenticate with the Mini-Mixer and obtain an authorization token to be used in future requests. The HTTP request is as follows:

POST /user/login

The request body must be a User resource with at least the username and password in order to authenticate. The request response will have HTTP code 200 and an authorization token in the response body if login was successful. The request response will return HTTP status 500 with an error message in the response body if the login was unsuccessful.

/ingredients – The ingredients resource is used when the client needs to view the current ingredients inside the Mini-Mixer or manage the configuration of each ingredient. This resource will also be used to add or remove ingredients from the Mini-Mixer. The ingredients resource properties are defined in Table [7.3] as shown:

Property Name	Value	Description
id	Integer	The id of the ingredient.
name	String	The name of the ingredient.
amount	Integer	The initial amount of the ingredient, in units.
amountLeft	Integer	The estimated remaining amount of ingredient left in the container, in units.
type	String	The type of ingredient being used.
location	Char	The location of the ingredient in the Mini-Mixer. Can also denote an ingredient placed in storage.
unit	String	The type of unit in amounts.

Table [7.3]

The available methods are as follows:

Ingredients: list – This method will list the ingredients entered into the Mini-Mixer according to the filtered parameters. The HTTP request is the following:

GET /ingredients/list

The HTTP request accepts a request body containing any one of the parameters of an ingredient resource. The HTTP response will contain one or more Ingredient resources that match the parameters specified.

Ingredients: add – This method will add an ingredient to the internal list of the Mini-Mixer. The HTTP request is as follows:

POST /ingredients/add

The HTTP request accepts an ingredient resource with a required name parameter. The request response will return HTTP status 200 and the full Ingredient resources in the response body. An HTTP status 500 will be returned if there is a duplicate-named ingredient.

Ingredients: edit – This method provides the ability to edit or update a particular ingredient in the Mini-Mixer. The HTTP request looks like the following:

PUT /ingredients/edit

This method requires an existing Ingredients resource with at least the Ingredient id parameter in the request body. The HTTP request response code will return 200 with the full Ingredient resource if successful. Otherwise, a HTTP status code of 500 will be returned.

Ingredients: delete – This method allows Ingredients to be removed from the Mini-Mixer database. The HTTP request looks like the following:

DELETE /ingredients/delete

This request requires a request body with an Ingredients resource id. The HTTP status code will return 200 if successful. Otherwise, the HTTP status code will return 500.

/sensor – The sensor resource is used to provide information on the state of the Mini-Mixer. The sensor resource properties are defined in Table [7.4] as shown:

Property Name	Value	Description
type	String	The type of sensor metric.
status	Boolean	The current status of the sensor metric.
metric	Integer	The metric value for the given sensor.

Table [7.4]

The status methods include metrics such as internal temperature of the Mini-Mixer, state of the internal cooling unit, current uptime of the machine, as well as other metrics that are found to be useful or novel in nature. The available methods are as follows:

Sensor: list – This method will return a list of Sensor resources. A sensor resource can be a fluid pump, temperature sensor, uptime, or wireless connection state and strength. The HTTP request is as follows:

GET /sensor/list

The HTTP request body can contain a Sensor resource with at least the Sensor id to filter sensors. If no request body is sent, the response will return a list of

Sensor resources with a 200 HTTP status response. Invalid requests will return a HTTP response code 500.

/recipes – The recipes resource is the endpoint used for everything surround the drink mixes owned by the authenticated user. This is also where the user can add, edit, remove their own recipes. The resource will also serve as a way to find the top used recipes on the Mini-Mixer and provide suggestions for new recipes for the user. The connection resource properties are defined in Table [7.5] as shown:

Property Name	Value	Description
id	Integer	The unique id of the recipe.
description	String	A brief description of the recipe.
ingredients []	List	A list of ingredient ids and their quantities for the recipe.
creator	String	The username of the recipe's creator.
name	String	The name of the recipe.
ordered	Integer	The number of times the recipe has been ordered since creation.

Table [7.5]

The available methods are as follows:

Recipes: list – This method will list the ingredients entered into the Mini-Mixer according to the filtered parameters. The HTTP request is the following:

GET /recipes/list

The HTTP request accepts a request body containing any one of the parameters of a recipes resource. The HTTP response will contain one or more recipes resources that match the parameters specified.

Recipes: top – The method will provide a list of the Top (up to 100) most popular recipes on the current Mini-Mixer. The HTTP request is as follows:

GET /recipes/top

The HTTP request requires no request body and will return a HTTP response code 200 and list of ordered Recipe resources in the response body.

Recipes: add – This method will add a recipe to the internal list of the Mini-Mixer. The HTTP request is as follows:

POST /recipes/add

The HTTP request accepts an ingredient resource with a required name parameter. The request response will return HTTP status 200 and the full Recipes resources in the response body. An HTTP status 500 will be returned if there is a duplicate-named Recipe.

Recipes: edit – This method provides the ability to edit or update a particular ingredient in the Mini-Mixer. The HTTP request looks like the following:

PUT /recipes/edit

This method requires an existing Recipes resource with at least the Recipes id parameter in the request body. The method also requires the currently logged in user to be the creator of the recipe. The HTTP request response code will return 200 with the full recipes resource if successful. Otherwise, a HTTP status code of 500 will be returned.

Recipes: delete – This method allows Ingredients to be removed from the Mini-Mixer database. The HTTP request looks like the following:

DELETE /recipes/delete

This request requires a request body with a recipes resource id. The recipe is also required to be created by the currently logged in user. The HTTP status code will return 200 if successful. Otherwise, the HTTP status code will return 500.

/mixer – The mixer resource is used to control and send commands that describe the actual physical mixing functions of the Mini-Mixer. The mixer resource properties are defined in Table [7.6] as shown:

Property Name	Value	Description
queuePosition	Integer	The position of the recipe order in the mixing queue.
recipe	Nested object	The object representing the recipe to be mixed.
isMixing	Boolean	Whether the drink is currently being mixed.
timeLeft	Integer	The estimated time left until the order is complete.
orderId	Integer	A unique identifier for a given order.
customerId	Integer	The unique Id of the user who placed the order.

Table [7.6]

This is where the user can initiate a mix given a recipe, manage the mixing queue, or perform actions such as starting and stopping the mixing process. The available methods are as follows:

Mixer: getQueue – This method will return the Mixer's queue as well as filter the queue given a Mixer resource parameter. The HTTP request is as follows:

GET /mixer/queue/list

The HTTP request may contain a Mixer resource with at least one resource parameters with the response being a full list of Mixer resources that were filtered from the queue using the request parameters. Otherwise, the response will contain the entire queue as a list of Mixer resources.

Mixer: order – This method is used to place an order on the Mini-Mixer. The HTTP request is as follows:

POST /mixer/order/create

The request body must contain the recipe parameter for a successful order. If successful, the HTTP response code will be 200 and the response body will contain the full Mixer resource for the order. Otherwise, the HTTP status will return 500.

Mixer: start – This method is used to begin the actual mixing process. The HTTP request is the following:

POST /mixer/order/start

The HTTP request body must contain the order ID being started. The order will only start mixing if it is at the top of the queue and the user who placed the order is starting it. A HTTP status code of 200 will be returned if this is the case. A HTTP status code of 500 will be returned otherwise.

7.3.3 CLIENT SYSTEM

Mobile Platform - The mobile platform of choice will be the Android Operating System. This has mainly been chosen over other considerations due to our team's familiarity with Java. The Android Operating System also boasts and very comprehensive SDK to implement our application. The development environment is also not limited to one single platform, as is the case in other considerations. The entire development stack is completely free of charge, which the exception of publishing to Android's Play Store, should we choose to do so. This is a major advantage for us when we consider our monetary constraints. We are also pleased with the User Interface design guidelines that are provided by Google. This is a great learning resource as our team does not have extensive knowledge

in the area of UI/UX design. The design of the Client Interface will closely follow Google's Material Design guidelines [40]. The Client System will be described using states of the mobile application and mockups as illustration for each state. The major states of the mobile applications are as follows:

- Setup Mode.
- Login and Account Creation.
- Home Screen.
- Top Drinks.
- Suggested Drinks.
- Ingredient Manager.
- Create a Drink.
- Edit and Manage My Drinks.

Login and Account Creation – Consider that the Mini-Mixer has been newly unboxed and the user is preparing to operate the machine. Once the Mini-Mixer has been unboxed and plugged into a power supply, the user will download and install the Mini-Mixer mobile application on their Android device. Once the application is started, the user will be met with the initial login screen. In the background, the client application will attempt to discover any Mini-Mixer service over the currently connected network. It was considered to allow the application to discover services on both the current connected network and services in range of Wi-Fi P2P, but this would cause the user to get disconnected from their current network and possibly lose connectivity. Mini-Mixer discovery will be implemented using Zero-configuration networking (zeroconf). [41] Zeroconf uses Domain Name Service (DNS) based service discovery to broadcast registered services to clients using standard DNS queries. This is extremely beneficial as the client application requires no initially-stored or preprogrammed information about the Mini-Mixer that it would like to connect to. This type of setup even allows multiple Mini-Mixer's to be chosen that are within the client's wireless range. The client application will implement the support for this service discovery using the available classes in the Android API provided by *NsdManager* for Normal or Operating Mode. [42] [43] The features provided by zeroconf will allow us to announce a unique name for a given Mini-Mixer as well as the service type and port. If no service is discovered on the currently connected network, the application will notify the user with a toast message and provide an optional action to go to Setup Mode, as shown in Figure [7-5]:

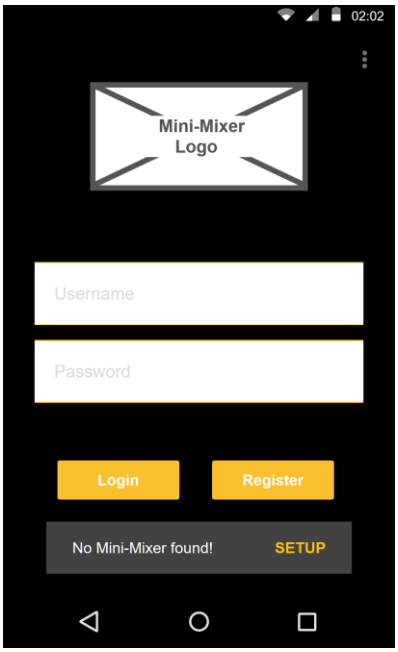

Figure [7-5]

In the case that the user did not enter setup mode from the toast, or simply missed the opportunity before the toast disappeared, there is an alternative way to access setup mode. The user can touch the action overflow icon at the top right of the login screen to be provided with a menu as shown in Figure [7.6], which provides access to setup mode as well as some information about the application.

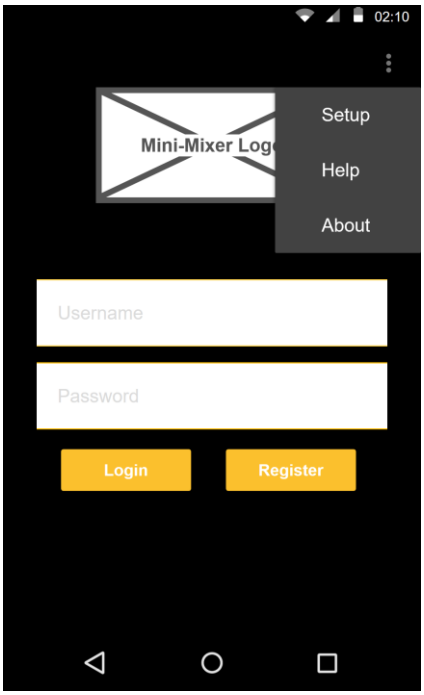

Figure [7.6]

In the case that the user is connected to a Wi-Fi network with at least one Mini-Mixer service, a toast will appear indicating that a Mini-Mixer was discovered, with an optional action to show a list of available Mini-Mixers as shown in Figure [7.7]:

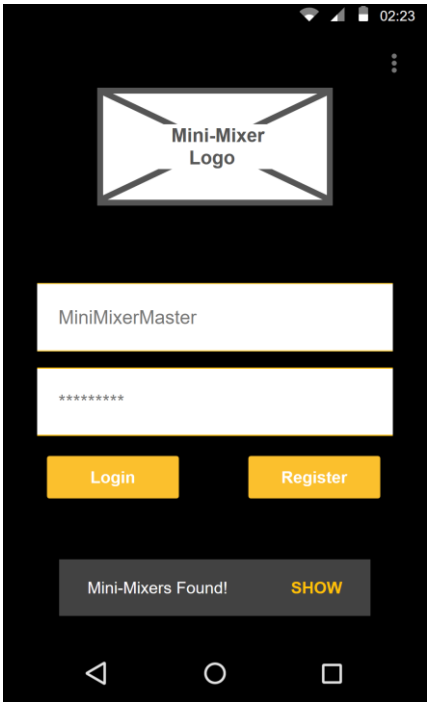

Figure [7.7]

The user may then either choose to login, register or show the Mini-Mixer's that have been found. All of these actions will lead to a bottom sheet being displayed so that the user may select and connect the Mini-Mixer of their choosing. The "show" action button will simply allow the user to select a Mini-Mixer to then either Login or Register. The Login and Register buttons will display the bottom sheet, but will then attempt to connect to the Mini-Mixer and then carry out their respective functions. The mockup in Figure [7.8] illustrates what the bottom sheet may look like for these actions.

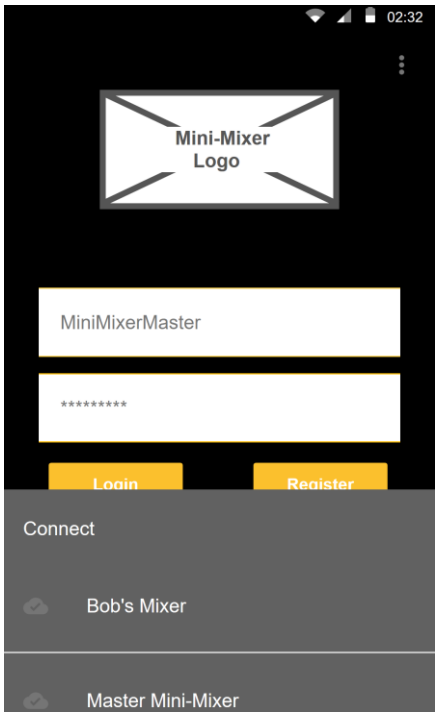

Figure [7.8]

Setup Mode – In the case that this is the first time a user is setting up their Mini-Mixer, the Setup Mode will be selected from the login screen either through a toast message or the dropdown menu. If the user is currently using their Wi-Fi device, the application will prompt the user with a dialog warning of network disconnection as it switches over to Wi-Fi P2P as shown in Figure [7.9]:

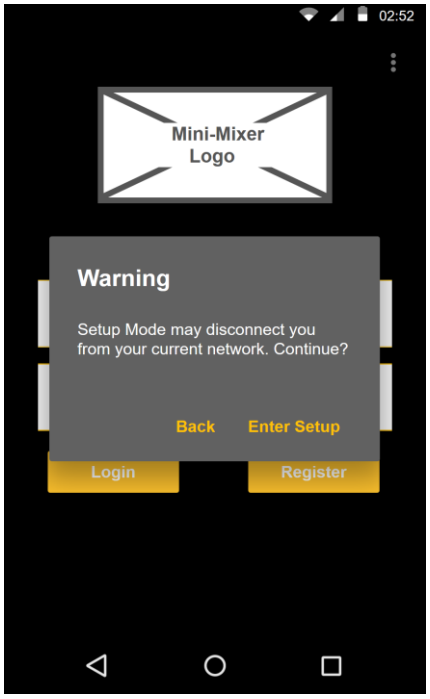

Figure [7.9]

Once confirmed, the application will proceed to initiate service discovery while showing an indefinite progress bar on the screen in the same position as the Setup Mode confirmation dialog. The client application will implement the support for this service discovery using the available classes in the Android API, provided by *WifiP2pManager* for Setup Mode. Once the service discovery is complete, a bottom sheet will appear with all discovered Mini-Mixers. The user will select a Mini-Mixer to setup and the application will proceed to the setup page. The setup page will contain fields for SSID, Security Type, and Password for the user to enter for their given home router.

Home Screen – The home page is the default landing page for a user after they have logged into the Mini-Mixer. The home page contains status indicators for the Mini-Mixer including internal temperature, wireless signal, and current mixing state. The home screen will also show the current mixing queue of order drinks and progress on any currently mixing drinks. From the home screen, the hamburger menu can be accessed to select other components of the application, such as settings, top drinks, suggested drinks, and the ingredient manager. The home page will also contain a floating action button, which will be present on all screens of the application. [44] The floating action button will be used to create, edit, and place drink orders very quickly. A flowchart of the process is shown in Figure [7.10]:

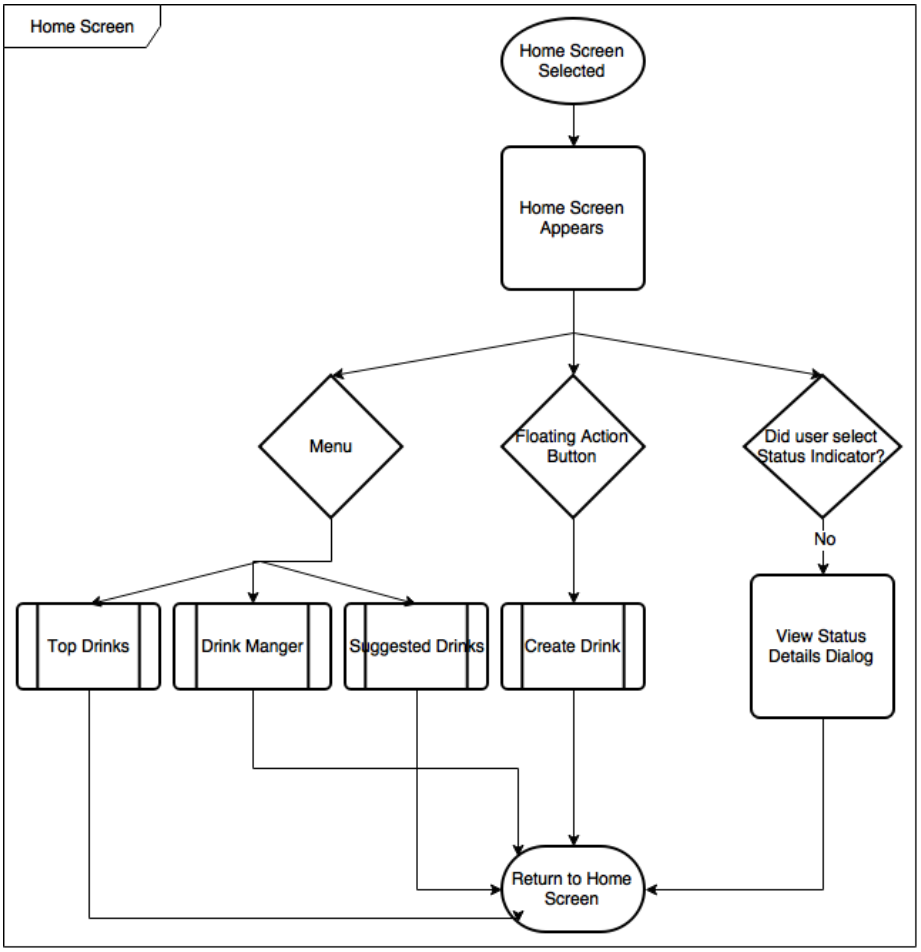

Figure [7.10]

Top Drinks – This screen will contain a list of top 100 drinks on the Mini-Mixer. The hamburger menu and the floating action button will be present as well to aid in navigation away from this screen. The top drinks will be displayed as a list of Card Components. [45] The card component for each recipe will contain two actions – order recipe and add to the user's current recipes. The card can be expanded on touch into Content Blocks which will provide additional details about the selected recipe. The process flowchart is shown in Figure [7.11]:

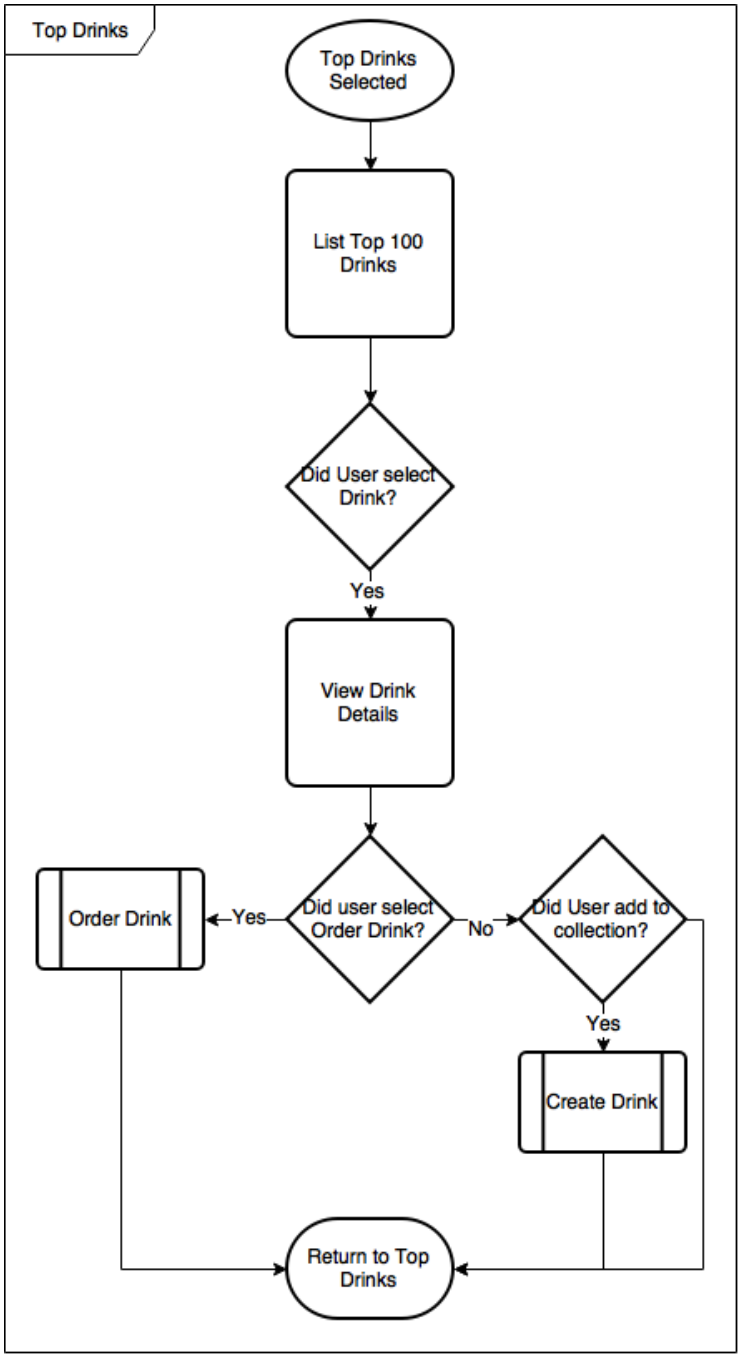

Figure [7.11]

Suggested Drinks – The suggested drinks tool will provide an overlay with a card component containing a suggested drink from available ingredients. The card overlay can be refresh for a new recipe to be suggested until the user is satisfied or wishes to exit the tool. The process flowchart is shown in Figure [7.12]:

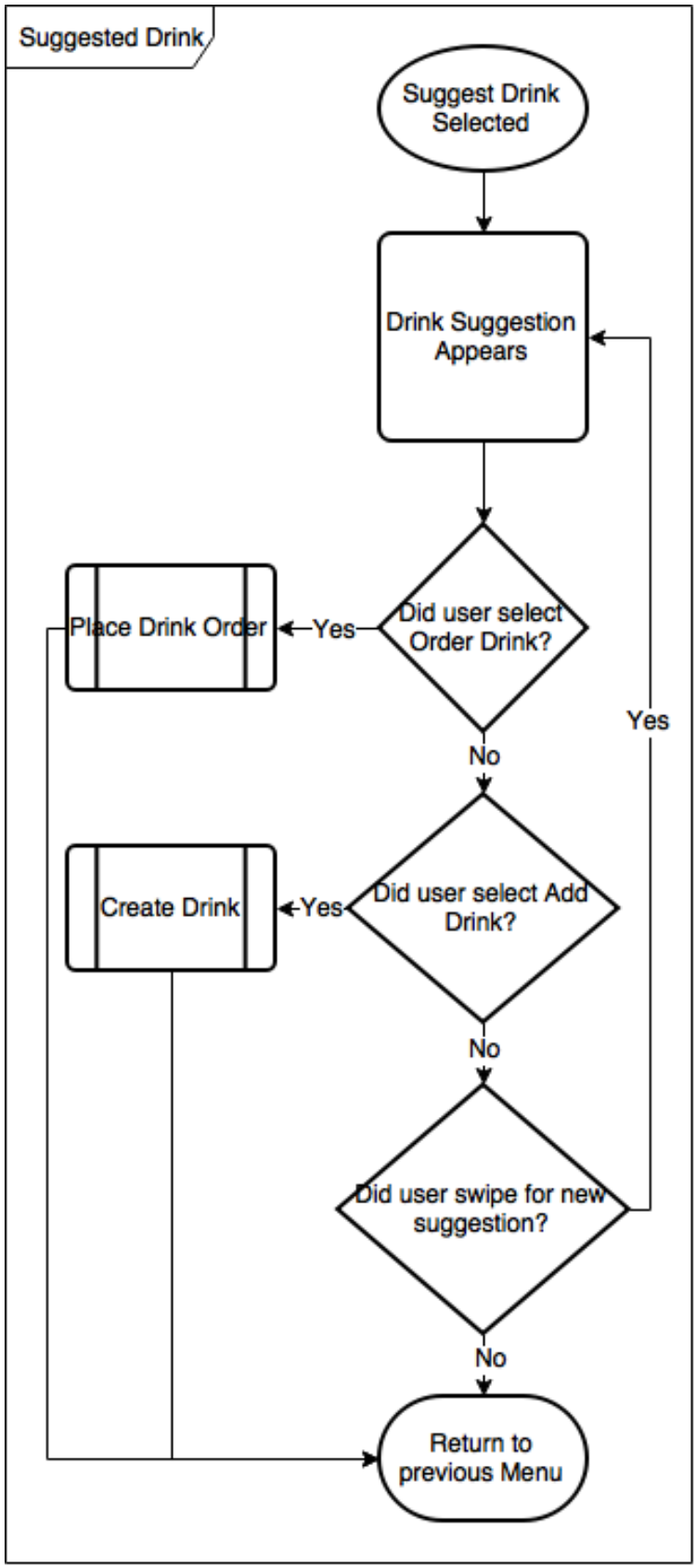
Suggested Drink
Suggest Drink Selected
Drink Suggestion Appears
Did user select Order Drink?
Yes
Place Drink Order
No
Did user select Add Drink?
Yes
Create Drink
No
Did user swipe for new suggestion?
Yes
No
Return to previous Menu

Figure [7.12]

Ingredient Manager – The ingredient manager is the interface that can be accessed from the hamburger menu which will be used to manage entered ingredients as well as moving ingredients from storage to a designated pump. The ingredient manager will consist of a screen using mobile tab components to differentiate between pumping stations and storage. [46] Each tab will be labeled with the pump id with one tab designated ingredients in storage. The tabs designating liquid pump spot will have a main card displaying the current ingredient as well as a list of smaller cards below it with available ingredients. The user only needs to select a smaller card to switch out an ingredient at a given liquid pump. The process is shown in a flowchart of Figure [7.13]:

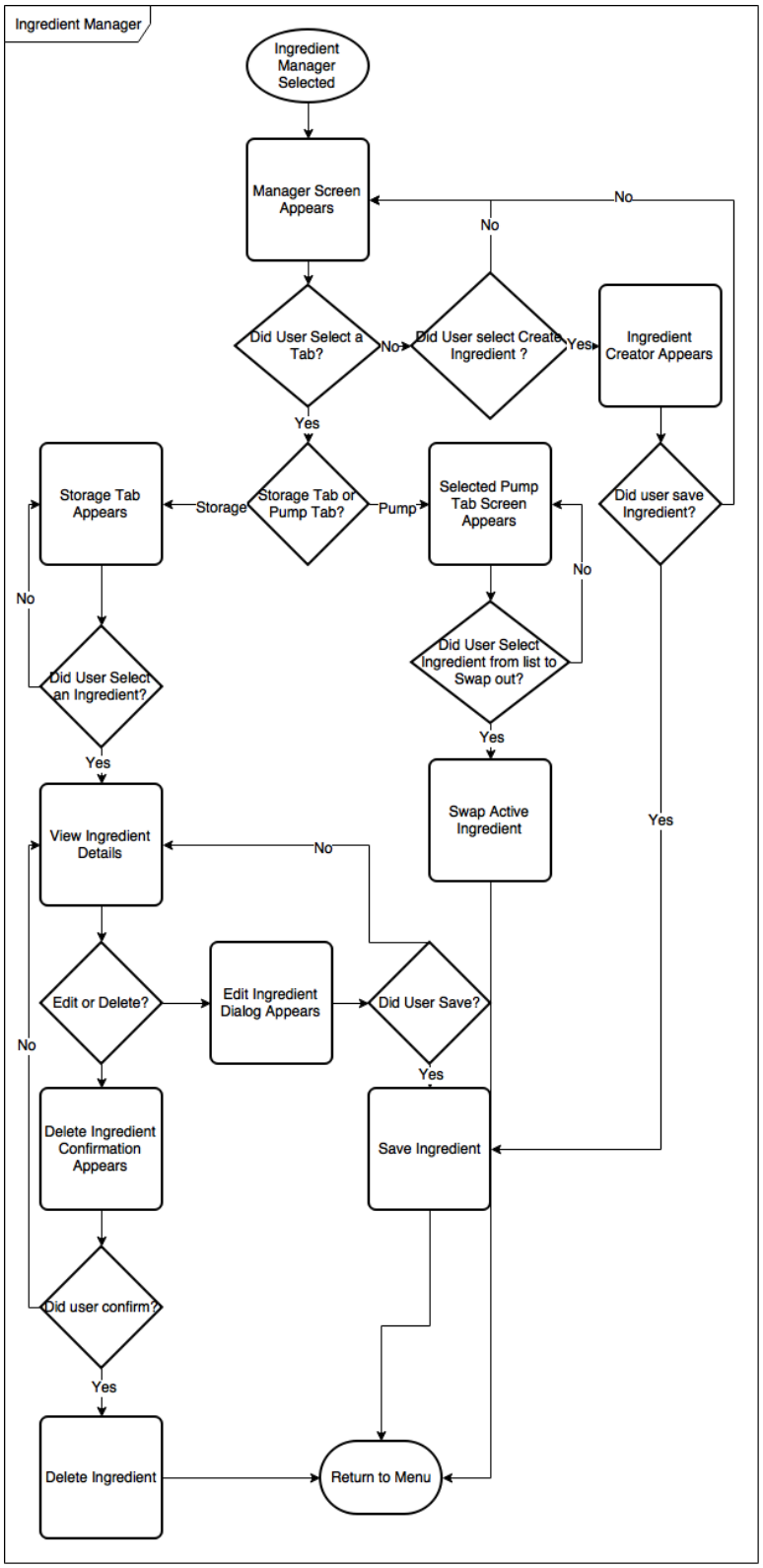
Ingredient Manager
Ingredient Manager Selected
Manager Screen Appears
Did User Select a Tab?
No
Did User select Create Ingredient ?
Yes
Ingredient Creator Appears
No
Yes
Storage Tab Appears
Storage
Storage Tab or Pump Tab?
Pump
Selected Pump Tab Screen Appears
Did user save Ingredient?
No
No
Did User Select an Ingredient?
Did User Select Ingredient from list to Swap out?
No
Yes
Yes
Swap Active Ingredient
Yes
View Ingredient Details
No
Edit or Delete?
Edit Ingredient Dialog Appears
Did User Save?
No
Yes
Delete Ingredient Confirmation Appears
Save Ingredient
Did user confirm?
Yes
Delete Ingredient
Return to Menu

Figure [7.13]

Create a Drink – The drink creation tool can be accessed from either the navigation button or the hamburger menu. The drink creation tool will feature a list of at least default two discrete sliders that are divided into five parts. Beside each slider will be a selection list for each available ingredient that has been entered into the Mini-Mixer. A floating button at the bottom of the sliders list will be used to add more ingredients if needed, up to a maximum of six total discrete sliders. The drink can be named at the top of the screen which will contain a title text field and a “done” button which will create and save the drink. A description of the drink can be added below the discrete sliders within a test field. A flowchart of the process is shown in Figure [7.14]:

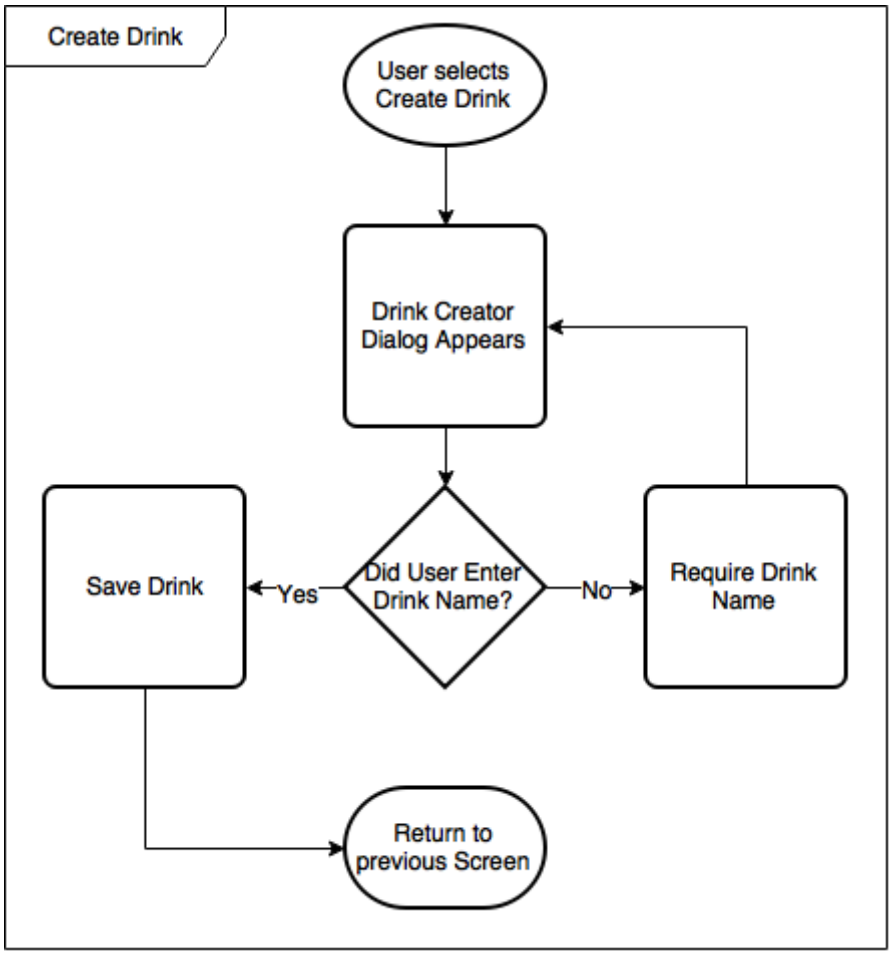

Figure [7.14]

Edit and Manage My Drinks – The drink manager can be accessed by either the floating action button or the hamburger menu. The drink manager will consist of a list of cards containing the drinks owned by the user of the application. The user can touch the card to provide more information in an expanded form. The user can also use the edit and order action buttons to be taken to the drink creation/editing screen or ordering screen, respectively. The flowchart for the Drink Manager is shown in Figure [7.15]:

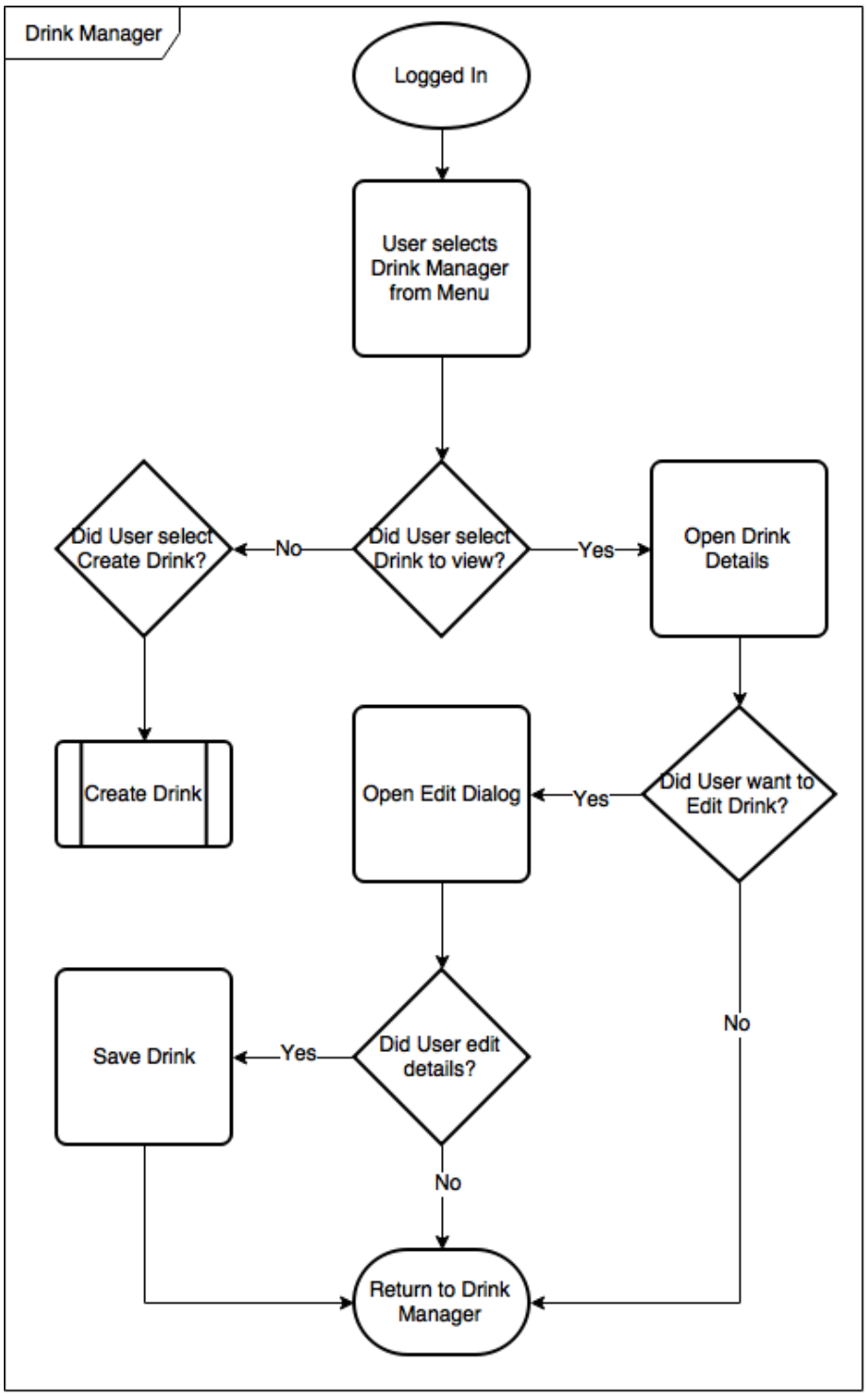
Drink Manager
Logged In
User selects Drink Manager from Menu
Did User select Drink to view?
No
Did User select Create Drink?
Yes
Open Drink Details
Create Drink
Open Edit Dialog
Yes
Did User want to Edit Drink?
No
Save Drink
Yes
Did User edit details?
No
Return to Drink Manager

Figure [7.15]

Personal Computer Platform - The personal computer platform is another client that we will be pursuing to some extent. This is mainly due to the fact that the Django REST Framework has a built-in feature where API endpoints have automatically generated views – in our case web pages. While not entirely practical or operational from a user's standpoint, this makes for a great debugging and testing platform to support, should we allow users to develop their own client applications in the future. The Interface for the PC will be provided by the mostly automatically generated Web Browsable API. [47] The Web Browsable API will mostly serve as a convenient point for experienced user to hack with the Embedded Server and create their own applications and experiences aside from the provided mobile client system. For our purposes, the Web Browsable API will serve as a software testing point for our test cases as well as during our iterative phases of our programming plan for the Mini-Mixer software.

7.4 COMMUNICATIONS

The communications between the hardware components has to facilitate each component appropriately without compromising the performance or capability of each component. We have a unique challenge in our Mini-Mixer implementation as we would like to both initially setup the Mini-Mixer as well as normally operate the machine exclusively using wireless technologies available to our hardware choices. With this in mind, we have chosen the communications methods carefully for each pair of hardware components.

7.4.1 CONTROLLER-SERVER

The controller-server communications will be facilitated by UART hardware over serial RS-232 standard communications. The commands being passed between the Embedded Controller and Embedded Server are small and simple and can be defined elegantly as serial commands. The UART connections only require two pins on each device, one for transmission (Tx) and one for receiving (Rx). This is ideal as we have a number of pumps and other sensors that will be accommodating the available pins on our MCU. ASCII character commands will be sent in a sequential manner from the Embedded Server to the Embedded Controller, with possible parity bit checking to determine if data transmission was successful. Unlike I2C, acknowledgement is not supported natively with UART; however, it is possible to emulate the behavior to ensure that the controller properly receives the commands. Certain functions will be written to facilitate this communication between controller-and server even further.

7.4.2 CLIENT-SERVER

The client-server communications will consist of a mixed-mode operation with a Wi-Fi module on the Embedded Server. The mixed-mode operation is divided into two modes:

- Wi-Fi WLAN Mode, or Normal Operating Mode.
- Wi-Fi P2P Mode, or Setup Mode.

We are taking advantage of Android's built-in Wi-Fi P2P mode support of our Client System in order to implement a seamless setup mode for the Mini-Mixer. In this mode, which will be activated by a "reset" physical input on the machine, the machine will disconnect from any current Wi-Fi access points and enter the Wi-Fi P2P mode with a preset PIN/passphrase. Once the "reset" input has been activated on the Mini-Mixer, the client should be able to enter a "setup" mode and connect to the Mini-Mixer serving as the host. At this point, the user will enter in information about their Access Point for the Mini-Mixer to connect and submit the connection settings to be applied. The Mini-Mixer's server will accept this request with the new/modified connection settings and apply them to the WLAN interface of the Embedded Controller. At this point, the Mini-Mixer will have dropped out of "Setup Mode" and entered "Normal Operation" mode, ceasing any P2P connection with client. If the settings are successful, the Mini-Mixer will indicate a "connected" status on its indicator. If not, the Mini-Mixer will indicate a "failed" status on the indicator and go back into "Setup" mode for the client system to attempt to apply new settings. Setup Mode is only activated when the Mini-Mixer cannot connect to an access point after leaving Setup Mode or if the user manually activates the physical "reset" input on the Mini-Mixer. Normal Operation mode will begin once at least one successful Setup has been completed. At this point, the Mini-Mixer is connected to an Access Point of the user's choosing in the traditional manner as most Wi-Fi devices are accustomed to. Once in this mode, it is up to the client to connect their device to the same Access Point, at which point the user may enter the application and login to their account using their credentials and make HTTP requests to the Embedded Server to control the Mini-Mixer. The Mini-Mixer will remain in this mode as long as there is a working connection to the Access Point.

The hardware implementation of this hybrid method for the Embedded Server will require a wireless module capable of ad-hoc or Wi-Fi P2P mode within a Linux operating system environment. We also require a device that is capable of connecting to a normal WLAN using an Access Point that is generally considered to be a home Wi-Fi router. For this, we need to ensure that the module used supports the major wireless standards in use as well as the data security protocols that are supported and preferred. The wireless standards most commonly in use today are IEEE 802.11n (draft), IEEE 802.11g, and IEEE 802.11b. There are a number of security protocols but we are going to focus on two of the most secure and forgo supporting the protocols that may still be in use

but are considered deprecated by the general security community. For this, we intend to support Wi-Fi Protected Access (WPA) and its successor, Wi-Fi Protected Access II (WPA2). [48] These requirements are going to be achieved using USB module for the Beaglebone Black. The module chosen is based on the RTl8192/8188CUS Chipset which has Linux drivers capable of this ad-hoc mode as well as supports our chosen wireless standards and security protocols. [49] [50] On the client side, this functionality is provided by the hardware and firmware of the mobile device. In our case, this could be any of the major mobile devices and smart phones that have been released in the past several years. We have determined that although not all devices will support the Wi-Fi Direct standard, we are confident that a sufficient and majority of modern mobile devices support this standard, which is appropriate for the Mini-Mixer. [51]

8. PROTOTYPE CONSTRUCTION AND PROGRAMMING

This sections outlines the methods of assembling the prototype as well as programming the software. All parts have been sourced from various vendors with the choice being heavily influenced by the price of the component. A construction plan has been outlined as well as a programming plan and roadmap.

8.1 PARTS ACQUISITION AND BILL OF MATERIALS

Each part that will be acquired for the construction of the Mini-Mixer will be documented for reference. Parts will be purchased from vendors and websites like Adafruit, Digikey, Amazon, eBay, etc. Some electrical components, such as resistors and capacitors, are already in possession and will not be purchased. These components will be logged for reference, however. The Bill of Materials for the Mini-Mixer is as listed below:

Bill of Materials

Item	***Name***	***Quantity***	***Description***
1	Atmel ATMega 328P	1	Embedded Microcontroller
2	BeagleBone: Black	1	Embedded Server
3	Peristaltic Pumps w/ Tubing	6	Mini-Mixer's pumps
4	TI SN75440 H-Bridge IC	2	H-Bridge Integrated Circuit for pumps
5	Colored Mechanical Push Button	2	Buttons for user input
6	TI LM22674 Buck Converter IC	2	Step down buck converter for embedded server and controller
7	Various Enclosure Materials	N/A	Miscellaneous Enclosure Materials
8	eTopxizu AC to DC 12V	1	Mini-Mixer's Power Supply

	20A Power Supply		
9	Adafruit Character Display	1	Character Display for Mini-Mixer's output
10	5mm LED	6	Indicator lights
11	Wi-Fi Module	1	Network connectivity
12	22pF Ceramic Capacitor	2	Crystal load capacitors
13	100Ω Through-hole Resistor	7	Resistors for LEDs
14	16MHz Crystal	1	Used as clock for the microcontroller
15	1KΩ Through-hole Resistor	1	For an active low pin on the microcontroller
16	Amber LED Strip	4	For enclosure lighting
17	Plexiglas Sheets	4	Material the Mini-Mixer's framing will consist of
18	RTV Silicon	1	Used to weld the plexiglas sheets
19	Polyethylene Wash Bottles (1000mL)	6	To hold the user's ingredients
20	Temperature Module	1	Monitoring the temperature of the Mini-Mixer
21	Peltier Cooler	1	Mini-Mixer's cooling device

8.2 PCB VENDOR AND ASSEMBLY

The Mini-Mixer's PCB will be constructed by 4pcb.com. They have several student discounts, which help with the overall cost of the Mini-Mixer. Specifically, the two layer $33/PCB discount will be utilized. The assembly of the PCB will be done by the team, preferably in the Senior Design Laboratory; this is to be done manually to also reduce the cost of production of the Mini-Mixer. It should be noted that lead presents a health and safety concern. To address this concern, it was decided that lead free solder will be used to assemble the board. This will require equipment that can handle the high melting point for lead free solder; lead free solder's melting point is around 100 degrees Fahrenheit higher than leaded solder. Multiple PCB boards will be ordered, in case assembly goes wrong and the board is damaged. However, in the final budget component listing, only one PCB will be listed, as there will only be one PCB in the Mini-Mixer's final iteration.

8.3 PRECONSTRUCTION

There will be a preconstruction phase before beginning actual construction of the Mini-Mixer. During the preconstruction phase, some small tests will be conducted so that data can be collected and be statistically analyzed in a manner that will

provide meaningful results. One test that will be done will be to run the pumps to find a relationship between the flow rate of the pump and the time elapsed. This data will be utilized to program a function that will convert the amount of ounces in an ingredient to the amount of time necessary to let the pumps run. Another test, similar to the previously described test, is to run the pumps to find a relationship between the flow rate of the pump and the frequency of the PWM signal used for the input. This is important to determine, as this will allow for a balance between speed of dispersing and safe usage of the pumps.

8.4 FINAL PROGRAMMING PLAN

The software development process is one of the most important areas of planning in the construction of the prototype. In the case of the Mini-Mixer, the development team involved in the software development also must develop the hardware as we are limited in human resources. For this reason, we are taking careful consideration in how we approach the software implementation process and how it correlates to the needs of our hardware development. We have three major software packages that are connected to each other to implement our functionality. However, we have two software developers to delegate resources to each software package. The design of the software is in such a way that all three software packages could be developed concurrently if we had more human resources. Due to our constraints, this forces us to prioritize two software packages initially if we wish to continue to develop concurrently. As we are under a time constraint, and our software packages are designed to be modular, we have elected to continue to develop at least two software packages concurrently during the implementation of the Mini-Mixer prototype. We initially considered Agile and its derivative development methods as a consideration for implementing our prototype. However, the requirement to deliver a working product very early in the development process would be a strain on our small team. Instead, we have decided to pursue a more traditional model that the Agile methods take some of their philosophy from. The software development process we will be following for the Mini-Mixer prototype will be the Iterative and Incremental Build Model. [52] This model has similarities to the Agile methods but with less of a focus on delivering a high-featured product early in the development process. This method also allows for developers to work on multiple software packages at the same time, with little interference between modules. This process allows for our team to manage risk by refactoring at any point of failure during each build increment. This is particularly important as we have identified some issues regarding the known specifications of our fluid pumps and the expectations of the unknown specifications. We are acquiring our fluid pumps from a third-party, so the datasheets for the fluid pumps are not immediately available aside from some limited information from the vendor. Due to reasons stated in the hardware design section, we have decided to continue to pursue these pumps for their immediate advantages. However, the exact accuracy and timing of the fluid pumps are unknown and will require a certain degree of testing and iteration to meet our requirements specifications. The incremental build

process can address these issues very nicely as we can adjust for these unknowns on-the-fly without affecting other software packages in the prototype. In our implementation, the first two increments have been chosen to be the Embedded Controller and the Embedded Server, due to their close relationship and the ability to debug client logic as needed directly from the Embedded Server. These two increments will be development in parallel, along with their communications model which could be thought of as a third increment. Once the requirements of at least one increment are satisfied, work on the Client System will begin, along with the increment for the communications between the Client System and Embedded Server. This build model allows us to backtrack on any increment in the case that the design requirements change or an unexpected issue arises that forces us to redesign components of the system.

8.5 HARDWARE CONSTRUCTION

As mentioned in an earlier section, the construction of the prototype will be done in an iterative process. For one of the first iterations, the enclosure will be constructed with wood, and will be left semi-open for ease of access to various parts. A standard cooler will be used to house the various components of the prototype in this early iteration. This cooler will be picked so that the Peltier cooler may be easily swapped in and out as necessary when conducting testing. The Embedded Server will simply be mounted in a similar manner as it will be in the final build, with the difference being the prototype's mounting will not be a semi-permanent mount. The Embedded Controller will also have similar mounting to that of the Embedded Server. Due to the cheap cost of the Embedded Controller's microcontroller, the prototype will initially feature this controller built on a breadboard, and then as the iterations progress, transitioned into a protoboard. One of the last iterations will switch to use the PCB as opposed to the protoboard. At least one SN754410 H-Bridge IC will be used to test several of the peristaltic pumps, to ensure parallel and sequential pumping work as intended. The two LM22674 buck converters will be used to power the Embedded Controller and the Embedded Server in early iterations, as will the final ones. These buck converters will not be prototyped on a breadboard, but rather on a protoboard; the reason behind this is because of the current limitations of a breadboard. Several amps of current will be necessary to run several pumps at once, of which a breadboard's internal metal strips cannot handle without overheating and possibly melting the plastic around it. The SN754410 will also be constructed on a protoboard as opposed to a breadboard for these reasons.

9. PROTOTYPE TESTING AND VERIFICATION

The goal of the prototype testing and verification phase of development is to ensure that the Mini-Mixer will meet all requirements specifications. Each set of tests are divided up into hardware and software components, much like the initial

requirements specifications. For each set of tests, we will define the testing environment(s) and the tools or equipment required for the set of tests. We will define a “test” as follows:

- Each test will map directly to a requirements specification.
- A one-to-one relationship will be enforced between the tests and requirements specifications.
- Each test will be performed on the individual component or the smallest subset of components possible first.
- A final run of all tests will be performed on the prototype system as a whole.
- The tests will follow a pass or fail procedure following the results of the individual and system-wide tests.

9.1 HARDWARE TEST ENVIRONMENT

The hardware test environment will almost exclusively be within the Senior Design Lab where hardware development will also occur. The consumer environment does not differ much from the Senior Design Lab which allows us to cover all test cases within the Lab. The tools required for these test cases will be the following:

- Scale.
- Timer.
- Measuring Glass.
- Wattmeter.
- Calculator.
- Personal Computer.
- Measuring Tape.
- Thermometer.

9.2 HARDWARE TEST CASES

The following are test cases mapped to each specification requirement. The test procedure will follow the given requirement with an implicit pass or fail result.

- The unit shall have a dry weight of no more than 40 pounds.
  - The unit shall be placed on a scale and weighed to ensure its total weight, neglecting any installed fluids, to not exceed 40 pounds.
- The unit should produce a mixed drink from start to finish in no longer than 1 minute.
  - A timer will be used to determine the time required to mix a given drink. As mixtures can vary significantly due to the customization involved, we will require an average to be calculated from a series of tests. The series of tests will involve mixtures from the range of one ingredient all the way up to the maximum of six ingredients. Each mixture will be timed three times to get an accurate average for the time it takes for a particular number of

ingredients to be mixed. These will then be average together to arrive at the final value for average time to mix a drink. This value must be under one minute.

- The amount of fluid in the components of each mix should have an error of no more than +/-10%.
  - A glass of a known size will be used to measure the mixture produced by the Mini-Mixer. Due to the high level of customization, we will require an average error to be calculated. This will be calculated from a series of tests performed over the range of mixes from minimum ingredients (one ingredient) to maximum ingredients (six ingredients). A series of three tests will be performed for each mixture of ingredients. A calculated percentage of error will be made from each test using the glass of known size. These will then be average together to arrive at a final average value, which will either pass or fail our requirement specification.
- The unit shall provide enough resources to hold 6 different fluids.
  - This requirement specification can be tested by simply counting the containers for fluids in the Mini-Mixer, with a pass or fail result.
- The unit shall consume no more than 600 Watts of power under load.
  - A wattmeter will be used to measure the power of the Mini-Mixer while under a mixing load. The test passes if the power consumption remains under 600 Watts and fails otherwise.
- The mixer prototype should have a combined total cost of no more than 800 dollars USD.
  - This test will pass or fail by simply calculating the total cost of the bill of materials. The test will fail if the total bill of materials exceeds 800 dollars USD.
- The mixer will be controlled using a mobile device with an application.
  - This test will encompass the connectivity and the ability to send commands to the Mini-Mixer from the mobile application. This means that the connectivity must be valid in both Setup mode and Normal operating mode. As we are communicating between the Mini-Mixer and the Client device using HTTP implementations, our test will be to send a HTTP request to the Mini-Mixer in both setup mode and normal operating mode. The test will pass if both cases provide a HTTP response from the Mini-Mixer.
- The size of the accepted fluid containers shall be no higher than 250 millimeters.
  - A metric ruler will be used to test this specification requirement. Each of the six fluid containers will be measured to ensure they are no taller than 250 millimeters, with a pass or fail result.
- The unit should produce a mixture with an initial temperature of no higher than 55 degrees Fahrenheit.
  - A liquid thermometer that is industry calibrated will be used to test this case. The final mixture's temperature will be measured immediately after production, with a pass or fail result.

- The unit shall have dimensions no larger than 2-foot Height X 3-foot Width x 3-foot Depth.
  - A ruler will be used to test this case. The Mini-Mixer will be measured by height, width, and depth upon complete assembly of the prototype. This will have a pass or fail result.
- The unit shall accept a glass size of 6 inches in height and 4 inches in diameter.
  - A ruler will be used to test this case. Glass sizes that do not exceed these dimensions will be measured and then placed under the dispenser of the Mini-Mixer to ensure the dispenser is centered directly over the glass. A pass or fail result will follow.
- The unit should be able to accept cocktails orders from a range of up to 20 feet.
  - A ruler will be used to measure a radius around the Mini-Mixer within the test environment. The Mini-Mixer will be setup to be ready to mix and pour a drink. A test user will stand at 20 feet from the Mini-Mixer and place a mixing order. This will be performed at 4 points around the radius of the Mini-Mixer. A pass or fail will result from each test with the entire test failing if one in the series of tests fails.

9.3 SOFTWARE TEST ENVIRONMENT

The software test environment will be located at the team member's homes, school, or any location with their personal computer. Since the software packages are mostly modular, this allows development to occur concurrently and independently of each team member. The hardware test environment will also be utilized when physical testing of the software logic with the hardware will be required. The following tools are required for these test cases:

- Personal Computer.
- Timer.
- Counter.
- Measuring Glass.

9.4 SOFTWARE TEST CASES

- We're placing a limit of no more than a 4 step process from the application's start screen to a drink in the cup.
  - This will be tested by usage of the mobile application once a user is logged in. The user will start from the home screen to place and start an order. The number of steps required should be no more than 4 or the test will fail.
- The application should be able to create 128 different combinations of the fluids in the machine.
  - This test can be verified by calculating the total number of possible ingredient combinations given the number of ingredients in the machine.

- The mobile application should be able to create mixtures in units/steps of 0.5oz.
  - This will be tested by using a measuring glass to ensure a single pump is able to produce mixtures in the appropriate units. Each pump will be tested individually as well as together with steps equaling the total size limit for a single glass.
- We are placing a limit of no more than 8 ounces (~237 milliliters) on the total mixture size that application can create.
  - The application will have unit test cases to ensure that the recipe creation cannot create a drink size larger than 8 ounces.
- The total time from submission of the mixture until the machine begins mixing should be no longer than 1 second.
  - A timer will be used to time from the point of an order start until the pumps start physically pumping. The test passes if the time is within 1 second and fails if it is over 1 second.
- The maximum allowed custom mixtures for a single user will be limited to 100.
  - A unit test case will be used to attempt to create over 100 mixtures for a single user. The unit test case will then retrieve a list of the owner's recipes. If the recipe count is over 100, the test fails. Otherwise, the test passes.
- The maximum allowed size of the top list will be limited to 100.
  - A unit test case will be created to create greater than 100 custom drinks and then retrieve a list of the top drinks. The test fails if the list is greater than 100 recipes and passes otherwise.
- The application will provide the limit of drinks on the machine, which is 6. Specifically, the application may only provide options for the current types of the 6 ingredients in the machine at any given time.
  - A unit test case will be created to ensure that an order cannot be placed unless all ingredients are fulfilled either by the user or the Mini-Mixer.
- The client application will need to have the ability to suggest at least 1 mixture to the user, based on available ingredients.
  - A unit test case will be created to request a suggested recipe. The test passes if at least one recipe is returned and fails otherwise.

9.5 VERIFICATION

Verification is required to ensure each subsystem functions as intended. Initial verification of the Embedded Server to Embedded Controller communication will be done through connecting to the Embedded Server with a computer via SSH, and issuing commands to the Embedded Controller with a serial terminal emulator. As the prototype build progresses, this test will further validate more subsystems; for instance, when testing a set of commands to run two pumps in parallel, not only is the communication between the Embedded Server and the Embedded Controller verified, but the functionality of the pumps, as well as the hardware driving and powering the pumps is verified. This verification process

will evolve to include the mobile app and software on the embedded system that will autonomously issue the commands to the controller.

10. PROTOTYPE OPERATION

The operation of the prototype has been designed to be as simple as possible for the user in a home environment. We have also designed a set of procedures for the initial setup of the Mini-Mixer to ensure that the device can be setup with as little hassle as possible. There are also a few maintenance procedures required for safety and intended operation of the machine. The usage of the Mini-Mixer itself is almost entirely handled through the mobile application using a menu and step-based approach for viewing, creating, editing, and mixing the user's drinks.

10.1 SETUP AND MAINTENANCE

From the perspective of the user, the only real setup that will be required is to plug the Mini-Mixer into an AC outlet, and set up the Wi-Fi connection on the Mini-Mixer. Maintenance will be conducted by enforcing a periodic flow rate calibration when the user requires dispensing to be very accurate. The user will be responsible for refilling and cleaning the ingredient containers, as well as ensuring that the containers are securely placed inside the Mini-Mixer's designated compartment. The user will also have to ensure that the tubing is properly fixed inside each the containers.

10.2 USAGE

Usage of the Mini-Mixer will be a very simple process, where a drink request can be made with customization in four steps or less. The Mini-Mixer is controlled almost entirely using a mobile device.
Requirements:

- Mobile device with the Android Operating System capable of Wi-Fi connections.
- Home router.
- The Mini-Mixer Client Application.

Simply open the Mini-Mixer app from your device. If this is the first time, a toast will appear saying that a Mini-Mixer is not connected. Touch the "Setup" button and you will be directed through the setup process. Ensure that the Mini-Mixer and you home router is both turned on and have your wireless access details ready. Once the router has been setup, you should be met with a connected toast once you return to the login screen of the application. Ensure that your mobile device is connected to your home router. If you haven't already, you will need to register an account with the Mini-Mixer by touching "Register". Fill in the details and you will be prompted to login. Once inside the application, you will see a main status menu with general information about your Mini-Mixer. The

menu on the top left will direct you to your Drinks, management interface, and suggested drinks. You can use the floating button at the bottom right to create a new drink or get started with placing an order. To exit the application, you can hit the “back” key and the application will exit and return to your home screen.

11. PROJECT MANAGEMENT

The project management section encompasses all facets dealing the administration surrounding the design and implementation of the Mini-Mixer. This includes the most important components such as the division of labor, the intended milestones, and the available budget for the Mini-Mixer prototype.

11.1 DIVISION OF LABOR

As the team responsible for the Mini-Mixer only consists of two Computer Engineering undergraduates, the division of labor will be divided roughly in half. The following table illustrates each team member, and what his division of labor corresponds to:

Thomas	***Davidson***
Embedded Server Programming • Wi-Fi (hardware, setup) • Client-Server communications	Embedded Controller Programming • Pump control • Server-Controller communications
Mini-Mixer Client • Mobile phone application (Android) • Web browser support	Hardware Assembly • Prototyping on breadboards/protoboards • PCB Assembly • Subsystem integration
Mini-Mixer Frame Construction • Various roles	Mini-Mixer Frame Construction • Various roles

Both members have “Mini-Mixer Frame Construction” listed, as the specific roles have yet to be determined. Keep in mind that this is a high level overview that could be broken down into a much larger table, but to conserve space, has been abstracted to this table.

11.2 MILESTONES AND SCHEDULING

The milestones and schedule provided will serve as a means to pace the work of the group and ensure that the project is complete given the limited time constraints. We are also intending on following a software development plan of

incremental and iterative implementation of each major module. This philosophy has been carried over to the implementation of the hardware throughout the scope of the roadmap as well. The roadmap has been carefully planned to consider the goals and objectives of each semester, as well as the time off between each semester including holidays and weekends.

11.2.1 ROADMAP

The roadmap will be our guide for delegating resources and tasks throughout the life of the prototype development. The roadmap is designed to accommodate the iterative and incremental design process, particularly at the beginning of the prototype implementation.

Summer 2015 May 31, 2015 – August 3, 2015 (roughly 10 weeks)

NOTE: * denotes a project milestone

Week 1 (May 31, 2015 – June 6, 2015)

- Completion of the Initial Project and Group Identification Project Document. *
- Initial project research.

Weeks 2 - 5 (June 7, 2015 – July 4, 2015)

- Conduct research for the project to form a "*body of knowledge*". This research can be divided in such a manner that will include the following: *
    - The documentation of senior design projects similar to the Mini-Mixer.
    - Real products available on the market that are relevant to the design of the Mini-Mixer.
- Pertinent hardware (microcontrollers, pumps, valves, etc.) discovered from said similar projects and products.
- Software methodologies, implementations, and testing procedures that relate to the Mini-Mixer.
- Towards the end of this four-week period, a transition from general research (i.e. possible microcontrollers) to more specific research (i.e. which microcontroller) will occur, which will lead into the next time block.

Week 6 (July 5, 2015 – July 11, 2015)

- Select parts and do extensive research on them, documenting all findings. *
- Each member begins to write their own portion of the *Final Documentation.*

Weeks 7 - 9 (July 12, 2015 – July 25, 2015)

- Each member continues to write up their part of the *Final Documentation*, researching and collaborating when necessary.

- By the end of week 9, the *Final Documentation* will be written in its entirety, only subject to slight revisions thereafter. *

Week 10 (July 26, 2015 – August 1, 2015)

- All final revisions are made during the last week.
- The *Final Documentation* is finished no later than August 1st. *

Fall 2015 August 24, 2015 – December 5th, 2015 (roughly 15 weeks)

Weeks 1-4 (August 24, 2015 – September 17, 2015)

- Completion of working prototype of main unit (hardware, software). *
- Initial testing of working prototype.
- Initial documentation.

Weeks 5-8 (September 18, 2015 – October 18, 2015)

- Begin designing final version of the main unit.
- Continued testing and documentation.
- User mobile application development begins, with testing integrated into the testing of the main unit to streamline the process.

Weeks 9-12 (October 18, 2015 – November 14, 2015)

- Final development and testing conducted for the main unit and user application.
- By the end of week 12, the project is declared finished as a working unit. *
- Any pertinent documentation written during weeks 1-12 is integrated into the *Final Documentation* from Summer 2015. *

Weeks 13-15 (November 15, 2015 – December 5, 2015)

- All required documentation is prepared and finish before December 5th, 2015.
- The presentation is created, revised, prepared, and delivered. *

11.3 BUDGET AND FINANCE

The budgeting section outlines the total estimated cost of all parts mentioned in the Bill of Materials. We have a requirement set for the total cost to not exceed a threshold, which is considered our budget for the Mini-Mixer prototype.

11.3.1 ESTIMATED BUDGET

The estimated budget is used to give a good estimation of each part given our acquisition sources, estimated shipping and handling, as well as room for additional parts and materials should the prices fluctuate over time.

Part	Quantity	Price	Total
Atmel ATMega 328P	1	$2.50	$2.50
BeagleBone: Black	1	$50	$50
Peristaltic Pumps w/ Tubing	6	$30	$180
TI SN75440 H-Bridge IC	3	$2.75	$8.25
Full ATX Computer Case	1	$120	$120
Thermaltake 500w PSU	1	$24	$24
Various LEDs	N/A	$15	$15
Wi-Fi Module	1	$12	$12
Various Electronic Components	N/A	$0.00	$0.00
Peltier Cooler	1	$25	$25
14Oz Bottles	1	$1	$6
PCBs	3	$34.66	$104
Total			**$540.75**

12. SUMMARY

The Mini-Mixer, as detailed in this document, is an autonomous drink mixer that strives to provide the best performance for the price point, while conforming to all safety and design standards. Combined with an aesthetically pleasing look and a relatively small form-factor, the Mini-Mixer will be the ideal appliance for any drink mixing necessities. The Mini-Mixer was designed with keeping a simple, intuitive interface in mind, while at the same time making such an interface contain

features that provided enough customization to suit any user's taste. With impressive speed and accuracy, users will be very satisfied with what the Mini-Mixer can provide. The Mini-Mixer uses safe, removable containers that are washable, for the sake of convenience. Compatibility is a large factor in the success of Mini-Mixer, to which the Mini-Mixer will support mobile devices, as well as personal computers. Cost of operation was also taken into consideration for the Mini-Mixer, with design choices geared towards power efficient devices to lower the overall power consumption of the system. The design of the Mini-Mixer utilized both electrical and computer engineering disciplines; this convergence of disciplines within the project was necessary from both a functional and practical standpoint. With only a team of two computer engineering undergraduates, this project has proven to be a challenging yet rewarding one. The design approach of the Mini-Mixer, overall, utilized a top-down approach that allowed for subsystem modularity; this resulted in benefits for several different constraints. This approach also allowed for flexibility of part selection, which is likely what contributed to producing a low budget. Overall, with all of the design decisions taken to fulfill the most goals and constraints, the Mini-Mixer will exceed all expectations, and effectively raise the bar for project based autonomous drink mixing solutions.

13. APPENDICES

The following are additional resources that were required to conduct research as well as develop this document.

13.1 APPENDIX A: COPYRIGHT PERMISSIONS

All diagrams and figures present in this document are original creations by the Mini-Mixer team unless noted in the figure description.

13.2 APPENDIX B: WORKS CITED

[1] http://www.coca-colafreestyle.com/home/
[2] https://github.com/partyrobotics/bartendro
[3] http://www.adafruit.com/products/1335
[4] http://www.ebay.com/itm/Thermoelectric-Peltier-Refrigeration-Semiconductor-Cooling-System-DIY-Kit-Cooler-/261792872834
[5] http://www.ebay.com/itm/New-Refrigeration-Thermoelectric-Peltier-Double-Fan-Cooling-System-Kit-Cooler-/261793842155
[6] http://www.purswave.com/101339-5573/201793_253554.html
[7] http://www.adafruit.com/products/1150
[8] http://www.ebay.com/itm/281697619068
[9] https://www.sparkfun.com/products/10455
[10] http://www.amazon.com/3-12V-Water-Pumping-Electric-RS-360SH/dp/B00D82W60O

[11] http://www.spelchek.com/12v-24v-dc-gear-pump-for-carbonated-soft-drinks/
[12] http://www.xylemflowcontrol.com/beverage-dispensing/air-operated-diaphragm-pumps/n5000-series-n5000-series-pump.htm
[13] https://www.sparkfun.com/products/10456
[14] http://www.homedepot.com/b/Plumbing-Pipes-Fittings-Copper-Pipe-Fittings/N-5yc1vZbuu2
[15] https://www.python.org/
[16] https://www.djangoproject.com/
[17] http://www.django-rest-framework.org/
[18] https://www.java.com/en/
[19] http://www.dropwizard.io/
[20] https://developer.bluetooth.org/TechnologyOverview/Pages/L2CAP.aspx
[21] https://developer.bluetooth.org/TechnologyOverview/Pages/RFCOMM.aspx
[22] http://standards.ieee.org/about/get/802/802.11.html
[23] http://www.st.com/st-web-ui/static/active/en/resource/technical/document/technical_note/DM00054618.pdf
[24] http://www.astm.org/Standards/D2000.htm
[25] http://www.ipc.org/toc/ipc-2221a.pdf
[26] http://standards.ieee.org/about/get/802/802.11.html
[27] https://www.bluetooth.org/en-us/specification/adopted-specifications
[28] http://www.ti.com/lit/an/slla037a/slla037a.pdf
[29] http://standards.nsf.org/apps/group_public/download.php/174/NSF_170-07-Watermarked.pdf
[30] http://www.nsf.org/newsroom_pdf/NSF_2-2012_-_watermarked.pdf
[31] https://www.adafruit.com/products/399
[32] http://www.amazon.com/eTopLED-Single-Output-Switching-Low-cost/dp/B004OWUP5U
[33] http://www.amazon.com/Super-Bright-Waterproof-Flexible-Lights/dp/B005EQROYK
[34] https://pypi.python.org/pypi/zeroconf/0.15.1
[35] http://drvbp1.linux-foundation.org/~mcgrof/rel-html/iw/
[36] https://tools.ietf.org/html/rfc7159
[37] https://tools.ietf.org/html/rfc2045
[38] https://www.iana.org/assignments/media-types/media-types.xhtml
[39] http://addb.absolutdrinks.com/docs/
[40] https://www.google.com/design/spec/material-design/introduction.html
[41] https://tools.ietf.org/html/rfc6763
[42] http://developer.android.com/training/connect-devices-wirelessly/nsd-wifi-direct.html#discover
[43] http://developer.android.com/training/connect-devices-wirelessly/nsd.html#discover
[44] http://www.google.com/design/spec/components/buttons-floating-action-button.html#
[45] http://www.google.com/design/spec/components/cards.html
[46] http://www.google.com/design/spec/components/tabs.html
[47] http://www.django-rest-framework.org/topics/browsable-api/

[48] http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1375945&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F9453%2F30009%2F01375945.pdf%3Farnumber%3D1375945
[49] http://www.adafruit.com/products/814
[50] http://www.realtek.com/products/productsView.aspx?Langid=1&PFid=48&Level=5&Conn=4&ProdID=274
[51] https://www.wi-fi.org/product-finder-results?categories=4
[52] http://www.softdevteam.com/Incremental-lifecycle.asp

13.3 APPENDIX C: DATASHEETS

Atmel ATmega 328P (Microcontroller)
http://www.atmel.com/Images/doc8161.pdf

T.I. SN754410 (H-Bridge)
http://www.ti.com/lit/ds/symlink/sn754410.pdf

T.I. LM22674 (Buck Switch)
http://www.ti.com/lit/ds/symlink/lm22674.pdf

HITACHI HD44780 (Character Display Controller)
https://www.adafruit.com/datasheets/HD44780.pdf

BeagleBone Black (System Reference Manual)
http://www.adafruit.com/datasheets/BBB_SRM.pdf