Mini-Mixer: The Miniature Wireless Automated Drink Mixer

Thomas Bergens, William Tuggle

Dept. of Electrical and Computer Engineering, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — This paper describes the design considerations and implementation of a wirelessly controlled, fully automatic miniature drink mixer. This automated mixing solution is targeted for many household use cases, including its versatile application as an indoor kitchen appliance as well as small outdoor events such as patio and garden parties. The Mini-Mixer solution is focused on ease-of-use, relative portability, copious drink combinations, and speed as seen by the consumer.**

Index-Terms -- microcontroller, microcomputer, power MOSFET, H bridge, peristaltic pump, RGB LED

I. Introduction

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 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. 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 Mini-Mixer is divided into three primary subsystems: the "Embedded Server", the "Embedded Controller", and the "Client System". Each of these have their own appropriate software and hardware representations. The Embedded Controller is responsible for doing most of the heavy-lifting in the context of actually controlling the pumps, thereby producing the main goal of the Mini-Mixer. The Embedded Controller also maintains communications with the Embedded Server to allow for requests to be served quickly. 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 Client System is the main method by which the user interfaces and controls the Mini-Mixer. There, the user will issue requests that the Mini-Mixer will process and complete. Overall, the integration of these sub-systems provides a seamless and responsive transition from drink request to drink completion that has not been accomplished in previous projects with the budget of the Mini-Mixer.

II. System Components

The Mini-Mixer has been designed with modularity in mind. This configuration is chosen to allow other engineers or the user to easily modify individual components of the Mini-Mixer without compromising on the functionality of another subsystem. The major components of the Mini-Mixer comprise of the following:

- Embedded Controller System
- Embedded Server System
- Client System

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 uses a popular wireless communications standard to both setup and use the Mini-Mixer. The Client System includes the Client Interface that includes used to control and command the Mini-Mixer. Using the Wireless transceiver on their device, the user is able to connect to the Mini-Mixer through the provided Client Application on their device. The Client Application has a Login Interface used to authenticate with a user's given credentials to be greeted with their personal Drink Menu. At the Drink Menu, the user is able to create, edit, remove drinks. The Drink Menu includes a

status indicator of the Mini-Mixer for various things such as mixing state and connectivity. 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 designated drinks for each slot will be presented.

The Embedded Controller System is responsible for the actual dispensing of the mixed drink. There are a total of 6 pumps – one for each ingredient in the Mini-Mixer. Each pump is connected to their individual liquid containers through food-safe tubing. Each pump is connected in one direction to the microcontroller. Each pump is also connected to a half h bridge; this half h bridge is connected to the pump, the microcontroller, and the 12 Volt power supply. The microcontroller has a serial transceiver which will be interfaced with the serial transceiver of the Embedded Server. The Embedded Controller receives instructions for each liquid pump which includes how much liquid is required from each fluid container for a specific drink mix. The Embedded Controller then determines the optimal sequence of pumping and begin the mixing process. The embedded controller reports back to the Embedded Server when each pump has begun and completed the pumping process. An overview of these major components as well as their subsystems and relations can be found in Figure 1.

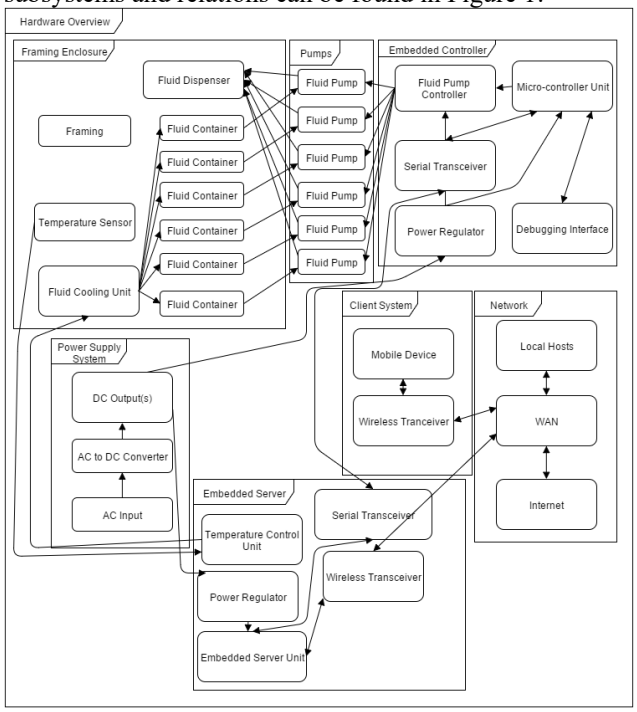

Fig. 1 Hardware Overview

III. Hardware Components

At its core, the Mini-Mixer is comprised of six liquid pumps that drive the mixing process. These pumps are driven by an Embedded Controller that receives instructions and sends commands to the pumps during the mixing process. The Embedded Controller receives commands from the Embedded Server. The Embedded Server stores 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.

A. Embedded Controller

The ubiquitous Atmel ATmega328P was chosen as the Embedded Controller for the Mini-Mixer, providing many GPIO pins and known hardware support due to it being the microcontroller of the Arduino Uno [1]. UART hardware is utilized to establish serial communication between it and the Embedded Server. 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 analog pins, capable of providing a PWM signal to the peristaltic pumps; accordingly, each pin corresponds to a pump in the Mini-Mixer. Providing power to the 328P is done via a Molex connector modified to where only the +5 V and ground lines are connected. In order to utilize the 328P, a 16MHz crystal, two 22pF ceramic capacitors, one 10 kΩ through-hole resistor, one 100kΩ through-hole resistor, and one green LED are necessary. 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. The two capacitors are equally as important, as they are required to provide a load for the crystal. The 10kΩ resistor is tied from ground 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 schematic designed in EAGLE can be seen in Figure 2.

All of these components are assembled on a two-layer PCB, reducing space and ensuring connections are secured.

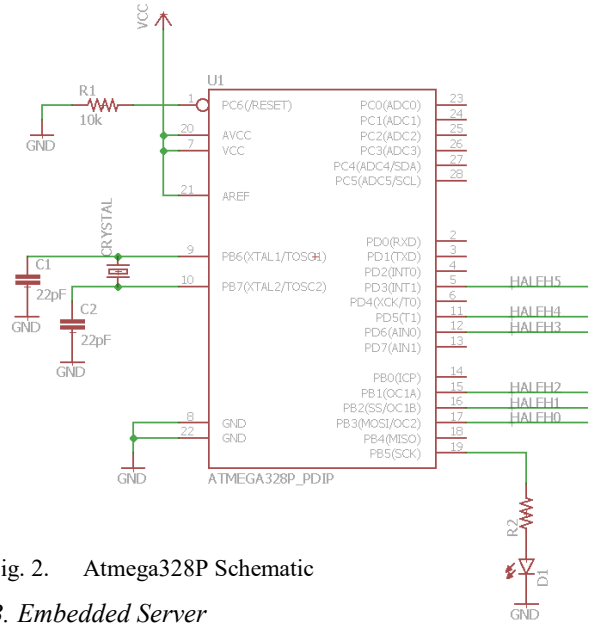

Fig. 2. Atmega328P Schematic

B. Embedded Server

The BeagleBone: Black was chosen to be used as the embedded server for the Mini-Mixer, with its open source advantages [2]. It is running the pre-installed Debian distribution; this distribution was chosen due to it being the default operating system on the BeagleBone, and due to its support, is likely one of the more stable operating systems available for it. The status indicator hardware is controlled with several of the BeagleBone's GPIO pins, while communication to the microcontroller (as previously stated) uses two GPIO pins.

C. Client System

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 is not an issue for almost all modern devices. The app is designed to not be resource intensive, which will help support older phones and laptops. An approach is also 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.

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

D. Hardware Interface

A status indicator is composed of a LCD backlit display mounted on the Mini-Mixer chassis. The LCD screen is used to provide immediate information in regards states of the system that are relatively easy to describe. Such states include "On", "Mixing", "Ready". These are controlled by the embedded server via GPIO pins. The character display utilizes the HD44780 controller, making it relatively easy to program. The display is controlled by the embedded server using 6 GPIO pins. The character display provides various prompts and feedback from the Mini-Mixer that the user will be able to clearly read.

E. Power Supply System

The Mini-Mixer's power is provided by the Thermaltake TR2 500W switching power supply. Due to its switching operation, it is more efficient than other conventional power supplies. The advantages that the standard 4 pin Molex connectors provide are utilized for the Mini-Mixer, providing both flexibility and the option to replace components if necessary.

F. Cooling System

The cooling system consists of a small air-to-air thermoelectric system powered by a Peltier plate. The system is comprised of a Peltier plate rated at 12 Volts with a current draw of 6 Amps. The Peltier plate is couple on both sides of the plate by two heatsinks and CPU fans to control the air flow through the heat sinks. The cold side of the Peltier plate has a small VGA cooler repurposed to serve as the heat sink and fan. The fan is measured at 45 x 45 x 10mm with a rated speed of 5000RPM. The hot side of the Peltier plate uses a large 90 90mm x 90mm heat sink with matching fan to help dissipate the heat from the plate. The components are attached to the Peltier plate using a layered composite thermal adhesive rated with a temperature tolerance of -40C to 150C which is well outside the range of the Peltier plate temperature with proper heat sinks and ventilation. This system will be acquired and is expected to run on 12 Volts and consume anywhere from 70-90 Watts. The Embedded Server monitors the cooling system using a digital temperature sensor to, ensure it keeps the contents of the Mini-Mixer within acceptable temperature ranges. The thermoelectric heat exchanger is mounted on the chassis of the enclosure with the cold side facing an insulated container chamber and the hot side facing ambient air.

G. Fluid Pump System

The fluid pump system is 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, and can vary depending on the duty cycle chosen for a particular mixer. 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 consists of 6 peristaltic pumps – one for each ingredient. 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. It should be noted that the relationship between the frequency of the PWM signal and the flow rate of the pump is not linear; experimentation was required to determine what ranges of frequencies give near linear relationships. Each pump requires at least one digital I/O pin for each direction of flow. In our case, we only care about one direction of flow so only one digital I/O pin will be required for each pump on the microcontroller. Each pump is driven using a half-H driver; these are provided via three TI SN754410 integrated controllers. These devices support a PWM input, which is what several of the GPIO pins on the embedded controller will be used for. Safety diodes are built into the integrated controller, which will help prevent damage to the integrated controller. Each half-H bridge supports up to 1000 milliamps of current, which satisfies the 500 milliamps requirement for the peristaltic pump.

H. 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 Rubbermaid 14oz bottles would be used. 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 holed out to allow 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 bottles are held in place by Velcro strips, making it easy to take them out of the Mini-Mixer when desired.

I. Frame and Enclosure

The Mini-Mixer's enclosure features a repurposed full ATX Thermaltake Spedo Advance computer chassis. This enclosure was modified to fit the needs and purposes of the Mini-Mixer. The dimensions of the chassis are 21.1 x 9.1x 24.0 inches, which is within the size constraints for the Mini-Mixer. A custom pump bracket and pump rack was built to hold all 6 pumps in an easy to access way. The rack is mounted to the case, and is easily removable if necessary.

J. 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 is dynamic and controlled by the Embedded Server. The Embedded Server is responsible for changing the state of the illumination depending on the mixing state of the Mini-Mixer and preferences of the user. Strips of RGB LEDs are lined around the Mini-Mixer, giving a distinctive, ambient feel to it. The lights are placed on the bottom trim of the Mini-Mixer. The lights are 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. The RGB LEDs are driven using power MOSFETs powered by the power supply as neither microcontroller can drive the Voltage or Current required to power the LEDs [3]. Figure 3 shows the power MOSFET schematic used in our implementation.

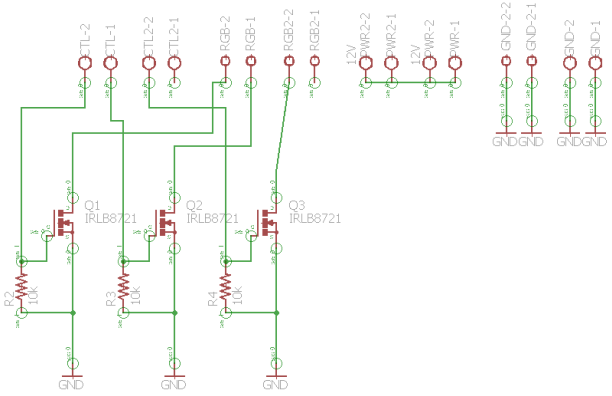

Fig. 3 Power MOSFET Schematic

IV. 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 and the user's installed ingredients.

A. 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 microcontroller as well as being a higher-level language than the alternative. This is important as the software was required 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 are sent serially via UART. The embedded server generates a request that the controller parses and then controls 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 is differentiated from each other by use of a single character letter. For ease of usage, these letters are the first 6 letters of the English alphabet: "A", "B", "C", "D", and "E". Only uppercase letters are used for this purpose, for advantages later seen in parsing. There are two modes of dispensing that the Mini-Mixer uses; 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. Aside from these two main reasons, another advantage can be found by bringing the two modes together; it is possible to create a drink quickly while providing a good mix by using both modes to create the desired drink. 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 are sent; each indicating the amount of each ingredient that corresponds to the two pump modes. From there, each ingredient and its amount follows 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. When the final sequential pump and ingredient amount is, a status echo is sent to the embedded server, where the embedded server will issue the final command, "S", to start the drink mixing. An example drink mix request from the embedded server to the embedded controller would look like the following:

R 2 3 A 1.1 B 0.4 F 3.0 E 2.7 D 0.8 S

This sequence of commands would start a drink request with two ingredients, "A" and "B", dispensed in parallel, while the other three ingredients are sequentially dispensed in the order in which the commands were issued from the embedded server. The duration of time the pump has been activated roughly corresponds 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 are set to be used for the previously mentioned periodic checking. Once an end time has been met or exceeded, the associated pump is shut down. The sequential mode for dispensing is more straightforward; each pump operates for the duration corresponding to the amount of the ingredient said pump is dispensing. The pumps are represented in the controller's software as data structures named "Pump". Each pump contains 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 is created to store each Pump associated with the parallel mode. The two dispense functions also facilitates communication between the controller and Embedded System, providing feedback as to the status as the pumps, which the Embedded System then uses to provide a status to the user through both the on board status indicators and the user mobile phone app/website.

B. Embedded Server

From a high level perspective, the Embedded Server is 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 happen between the two to ensure proper speedy operation. The Embedded Server is also responsible for monitoring the temperature of 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 also controls the character LCD in the Mini-Mixer, which is 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 is implemented using the Python programming language with the Django REST Framework on the Linux operating system. 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.

The Mini-Mixer implements service discovery using the Zero-configuration networking (zeroconf) standard through "iw" Linux tool. The iw tool is accessed directly from a Django app using Python's built-in subprocess module to run iw by calling the Linux command. As the Embedded Server is 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 is 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. All requests that do not change data or state of the Embedded Server are implemented using GET requests. Otherwise, POST requests are used to indicate a state change or data manipulation on the Embedded Server. Figure 4 illustrates the entire set of resources required to power our API.

Resource	Description
/connection	Server-side wireless connection configuration.
/user	Account Management, Authentication
/ingredients	Describes single(or many) ingredient and properties.
/sensor	Provides sensor state info(temperature, uptime)
/recipes	Describes one(or many) drink recipes and actions.
/mixer	Manages submission and status of drink orders.

Fig. 4 Embedded Server API

C. Client System

The mobile platform of choice is the Android Operating System. The Android Operating System boasts and very comprehensive SDK to power the application. The Android User Interface Guidelines were used to dictate the look-and-feel and well as the user experience of the application. User Interface design guidelines that are provided by Google. The Client System is described using states of the mobile application. The major states of the mobile applications are as follows:

- Login and Account Creation
- Home Screen
- Top Drinks
- Suggested Drinks
- Ingredient Manager
- Recipe Manager

The initial components that the user may encounter are the Login and Home screens. The login screen is used for account creation and authentication. The Home Screen the hub from where the user can access the remaining components of the client system. The main components of the application consist of the Ingredient Manager and the Recipe Manager. The Ingredient Manager is used to view, edit, and modify the drinks that are active inside the Mini-Mixer, as well as drinks that had been previously entered but may not reside in the machine. The Recipe Manager utilizes the ingredients from the Ingredient Manager to allow the user to create and edit their own, personal drink mixtures. These are then used by the Top and Suggested Drinks components to display popular drinks or new drinks that the user may want to try. A flowchart exhibiting these main components can be found in Figure 5.

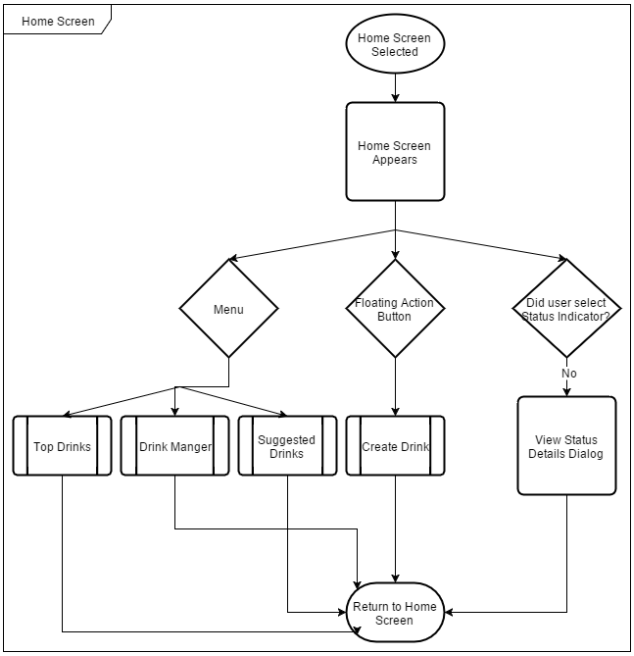

Fig. 5 Home Screen Flow Chart

V. COMMUNICATIONS

The communications between the hardware components facilitates each component appropriately without compromising the performance or capability of each subsystem. We have a unique solution to these challenges in our Mini-Mixer implementation as we aimed 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.

A. Controller-Server

The controller-server communications are 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.

B. Client-Server

The client-server communications 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
- Wi-Fi P2P Mode

We have taken 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, as well as offer multiple connection options should the user not have a supported access point or otherwise. Once the Mini-Mixer application is launched on the user's device, and option is present to enter a P2P mode and connect to the Mini-Mixer serving as the host. The user can then choose to enter information about their Access Point for the Mini-Mixer to connect and submit the connection settings to be applied. The Mini-Mixer's server accepts this request with the modified connection settings and applies them to the WLAN interface of the Embedded Server. At this point, the Mini-Mixer can now be accessed through Wi-Fi P2P or through the configured access point, depending on the user's preference. If the settings are successful, the Mini-Mixer will indicate a "connected" status on its hardware indicator interface. Otherwise, the Mini-Mixer will indicate a "failed" status on the indicator and the user will need to fall back to the P2P connection to the Mini-Mixer. Once both connection methods have been configured, it is up to the client to connect their device to the same Access Point or through P2P, 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 hardware implementation of this hybrid method for the Embedded Server requires a wireless module capable of 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, have chosen a wireless module that supports the major wireless standards in use as well as the data security protocols that are supported and preferred. For this, we intend to support Wi-Fi Protected Access (WPA) and its successor, Wi-Fi Protected Access II (WPA2). These requirements are 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. 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.

VI. Conclusion

The Mini-Mixer 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 is 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 has exceeded our expectations, and effectively raised the bar for project based autonomous drink mixing solutions.

Acknowledgement

The authors would like to acknowledge the considerable help and guidance given to our team by Dr. Samuel Richie. The authors would also like to acknowledge the various peers and mentors that were willing to provide feedback and critique on the countless issues encountered while designing and implementing this project.

Thomas Bergens, is a 24-year-old Senior Computer Engineering student at the University of Central Florida. Thomas is actively pursuing a working career in software engineering. He currently holds a Network Security position at a web hosting company.

William Tuggle is a 20-year-old Senior Computer Engineering student at the University of Central Florida. He has accepted a position at Northrop Grumman after a summer and fall extended internship. He plans to continue his education through graduate school, with a focus on engineering.

References

[1] SN754410 Quadruple Half-H Driver Datasheet, Website: http://www.ti.com/lit/ds/symlink/sn754410.pdf

[2] BeagleBone Rev A3 System Reference Manual, Website: http://beagleboard.org/static/beaglebone/a3/Docs/Hardware

[3] V. Barkhordarian Power MOSFET Basics, Website: http://www.irfc.com