Mr. Mix: The Automated Home Bartender

Brian Moore
Robert Hulbert

Computer Engineering

California Polytechnic State University, San Luis Obispo

Table of Contents

Table of Figures

1.0 Introduction

1.1 System Overview

Mr. Mix is an automated home bartender system that allows the end user to easily request a mixed beverage from the selection over a wireless network and have it prepared from the available set of liquors and mixers within the system. To accomplish this functionality, Mr. Mix operates from a microcontroller that accesses an Amazon Web Services (AWS) server to collect the variety of beverages available as well as retrieve requests from the end user who also interfaces to the server via a wireless capable device. After retrieving the request from the server, the microcontroller actuates solenoids for flow control and uses measurement calculations from the load cells to effectively pour the drink for the end user.

1.2 Goals and Objectives

Goals:

- Allow users to observe all known beverage combinations from their given selection
- Allow users to modify a known beverage or combination to their liking
- Allow users to request a selected beverage or combination of their choosing
- Pour the requested contents into user requested glass size
- Allow any authorized user to accomplish the above goals

Objectives:

- Implement user interface on AWS server
- Link mixed beverage database to the AWS server
- Establish connection and operation between AWS server and microcontroller
- Use solenoids to control the flow of all component beverages
- Use load cells to correctly weigh container and control overall beverage quantity
- Use LED Screen to display relevant information to user
- Program microcontroller to interconnect all components

1.3 Outcomes and Deliverables

Outcomes:

- Basis for further development of an automated home bartending system
- Concrete system design experience
- System implementation experience

Deliverables:

- Prototypes of functional aspects
- Program files for the software
- Complete design specification for a fully functional system

2.0 Product Definition

2.1 Overview

Mr. Mix is designed so anyone can create new and interesting cocktails without formal training. The product is designed to work with any internet connected device and create custom cocktails with the push of a button.

The device is controlled by a microcontroller with embedded wireless capabilities. By connecting to the cloud the microcontroller can receive instructions which allow the system to function properly. This allows for the controller to be a web interface accessed on any internet connected device.

Mr. Mix can be used at any size gathering and all guests can order from their own devices. The intuitive interface allows drinks to be quickly ordered. The process does not distract users from their surroundings. The device allows the host to spend time with guests instead of time making drinks.

2.2 Consumer Requirements

1. The device shall be able to fit on a countertop
2. The device shall be food safe
3. The device shall be easy to clean
4. The device shall connect to a home Wi-Fi network
5. The device shall be controllable from a smartphone or laptop
6. The user interface shall be intuitive.
7. The device shall prevent unauthorized users from ordering drinks
8. The device shall be able to mix at least two liquids
9. The device shall be able to pour into multiple sized glasses

2.3 Engineering Requirements

1. The device shall dispense set amounts of liquid
   - 1.1. There shall be at least two different liquids
   - 1.2. Each liquid shall be individually controlled
   - 1.3. The exact amount of liquid shall be configurable per request
     - 1.3.1. The device shall be able to dispense a minimum of 8 ounces per request
     - 1.3.2. The device shall be able to dispense a maximum of 20 ounces per request
     - 1.3.3. The device shall be accurate within .1 ounces
   - 1.4. The device shall prevent liquid from dispensing if there is no cup present
   - 1.5. The device shall allow users to determine when liquid is poured

2. The device shall communicate with the cloud over Wi-Fi
   - 2.1. The wireless network shall be configurable
   - 2.2. Users shall be able to register the device to their account
   - 2.3. The device shall receive messages from the cloud
     - 2.3.1. The device shall receive messages as they are generated by the server
     - 2.3.2. The device shall receive messages from the cloud at any time
   - 2.4. The device shall retain configuration in the event of a power loss

3. The device shall provide an authorization code for authorized users
   - 3.1. Authorization codes shall be unique
   - 3.2. The device shall only have one authorization code at any given time
   - 3.3. The device owner shall be able to revoke authorization codes at any time
   - 3.4. The device owner shall be able to generate a new authorization code at any time
   - 3.5. The device shall display the authorization code to the users
   - 3.6. The device shall allow anyone with the proper authorization code to pour drinks

4. The device shall be usable in a household environment
   - 4.1. The device shall plug into a standard outlet
   - 4.2. The device shall use minimal power

4.3. The device shall have a minimal footprint
4.4. The device shall be movable within the house

2.4 Criteria

The development of Mr. Mix is directed by the engineering requirements along with the following criteria, listed from highest to lowest priority.

Ease of Use:

The experience of using Mr. Mix shall be positive. This is essential for the device to be a benefit to household entertainment instead of a hassle.

Accuracy:

The drinks created by Mr. Mix shall meet the expectations of users. The accuracy of the product ensures that the customized cocktails selected by users meet expectations of quality and taste.

Security:

Mr. Mix shall be usable for entertaining guests in a "bring your own device" model. The owner should not have to worry about unauthorized ordering of drinks.

Consumer Viability:

While it is not a requirement of this project Mr. Mix shall be designed with the consumer in mind. The final design is plausible for consumer sale and could be extended to a complete business in the future.

3.0 System Design

3.1 Design Process

The design of Mr. Mix is split into modular components of hardware, software, and mechanical. These three aspects interact, but serve independent purposes. For this reason, the design process for each component is independent. Each component is designed as a black box which does not need to rely on the implementation details from other components. A top-down approach allows the connections between each system to be designed before any specific subsystem is designed.

Mechanical:

The mechanical design aspect of Mr. Mix revolves around previously prototyped pressure models. The goal for the mechanical design is to minimize large pressure equipment normally associated with fluid dispensing. This minimization also reduces the need for the corresponding large electrical equipment providing a more portable system. Gravity provides a large negative acceleration on the fluid within the container which means that it can be used to induce flow out of the bottle. However, this flow is halted when there is a pressure differential between the gas inside the bottle and outside. This pressure differential can be induced by restricting air flow preventing any liquid from flowing out of the bottle. This simple method was first tested by applying a membrane that restricted air flow effectively cutting off fluid flow. In the final design, the airway is actuated by a solenoid to control fluid flow. Aside from the minimization of electromechanical pressure systems, the design also focuses on implementing a versatile beverage top that regulates flow.

Hardware:

The main goal of the hardware design is to minimize overall power consumption and cost of components. With this focus in mind, an embedded architecture is the most viable option as it concentrates in these areas. A microcontroller with low power wireless connectivity in addition to relatively low power components fits with the above criteria. However, the most power consuming and expensive aspect is the flow control of the beverages. Most commercial flow control products are motor actuated valves which require significant power and time to operate. These motors are also extremely expensive and generically used for laboratory precision measurements. Low-power solenoids with electronic switches are a much more viable option in terms of power consumption, cost, and reaction times.

Software:

The software for Mr. Mix is designed with scalability and maintainability in mind. Cloud architecture provides massive opportunities for scaling software on demand allowing the device to meet scaling requirements for the future if needed. By comparing popular cloud platforms such as Google Cloud Platform, Microsoft Azure, and Amazon Web Services an informed design decision can be made. Looking at the engineering requirement and the different options Amazon Web Services sticks out as the best option for its interconnected platform and simple support for IoT devices. If future development on the system could happen making sure the software can easily be updated is also a requirement during the design process. The documentation and developer support for Amazon Web Services is vast allowing future development to be much easier than other platforms. In addition, many developers already know Amazon Web Services and could be hired to support Mr. Mix in the future if needed.

3.2 System Architecture

Mr. Mix separates the computationally intensive aspects from the physical device to lower power consumption and hardware cost. As depicted in the figure below, the microcontroller of Mr. Mix, ESP8266, communicates via Wi-Fi to the AWS server. In doing so, the application-level computation is extracted from the physical device. The AWS server provides the user with a simple front end to request a variety of mixed beverages obtained via database or combination choice of said user. Then the AWS server processes the request filtering any unauthorized users and sends the data down to the ESP8266.

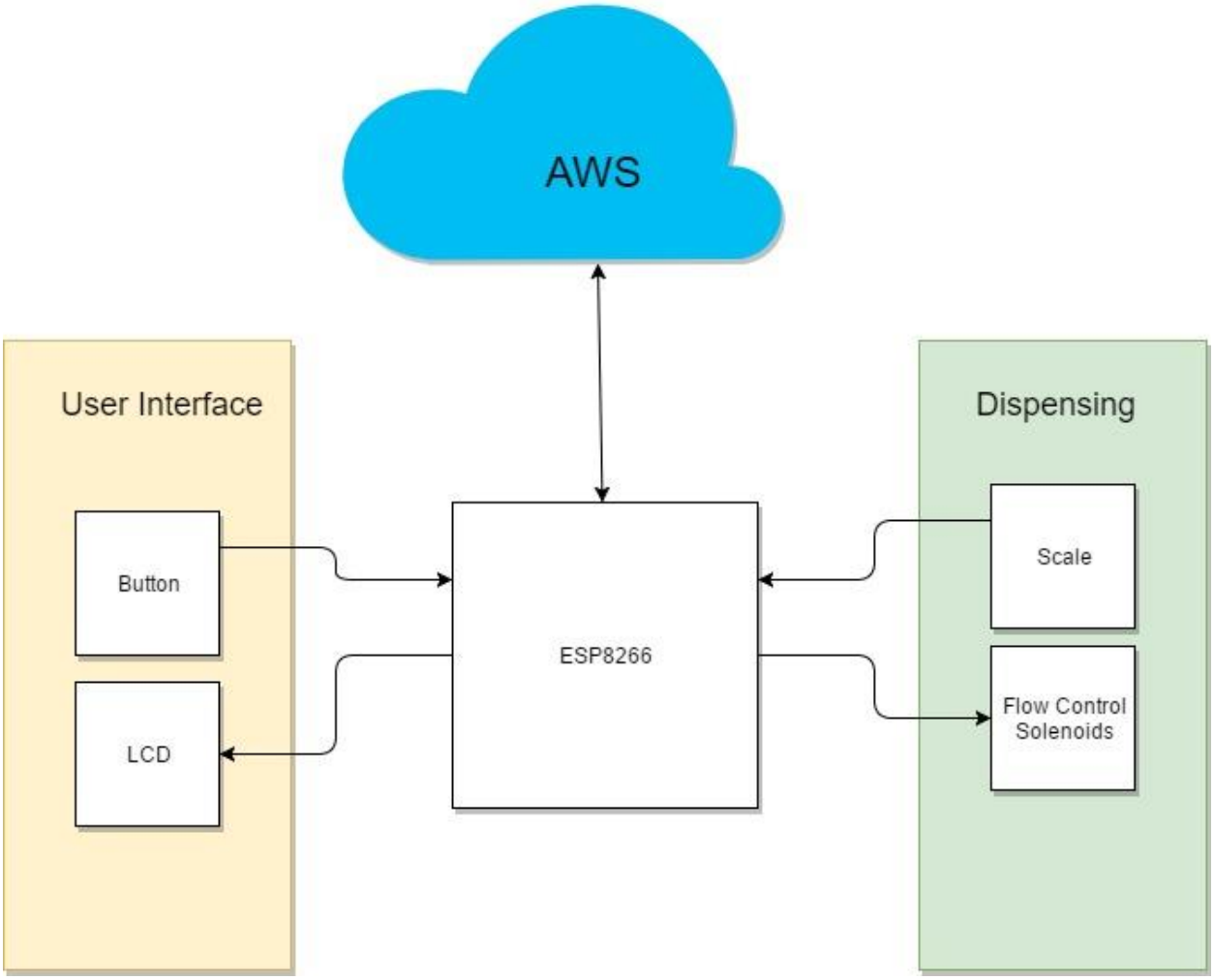

Figure 1: Mr. Mix System Architecture

The ESP8266 interacts simply with the user to indicate when the beverage is ready to be poured via an LCD screen to display information and a button for user confirmation as shown in the user interface section above. Once the user has confirmed their request, the ESP8266 measures the current weight via the digital scale and proceeds to dispense fluid until the volume specified in the request has been reached which constitutes the dispensing section.

3.3 Mechanical Architecture

Due to the variety and expense of industry standard dispensation systems, the mechanical architecture of Mr. Mix is subdivided into three main components: container, sealing apparatus, and dispensing mechanism.

For simplicity, Mr. Mix allows any common beverage container with a spout approximately 11/16 in in outer diameter. This setup allows for simple switching between empty beverage containers as well as minimizing cleaning of Mr. Mix. Furthermore, commercial beverage containers are made to be food safe integrating well with the food safe tubing.

As the only standard to the container is outer diameter size, the sealing apparatus is a silicone, food-safe rubber stopper that is lenient to about 1/16 in in outer diameter. The rubber stopper is modeled after common apparatuses used for pouring liquors at restaurants. These stoppers can control flow rate through the ratio of the diameter sizes of the air and fluid openings. The rubber stopper mimics the ratio found in restaurants which is 1/8 in outer diameter to ¼ in outer diameter of air and fluid, respectively.

With a constant flow rate, the last task of the dispensing mechanism is to meter out desired quantities based on the actuator. Mr. Mix utilizes gravity instead of a pressurizing motor to withdraw the fluid from the container. This system requires the container to be held inversely to its natural state. The force generated on the liquid by gravity creates a vacuum in the bottom (now top) of the container. The pressure is normally equalized via the air opening at the top (now bottom) of the container. This opening is then sealed by an actuator (solenoid) which when desired equalizes the pressure in the top of the container dispensing the beverage.

3.3.1 Mechanical Evaluation

To reduce cost of the overall product, the mechanical architecture lacks stability in its operating environment specifically in container structure and dispensing reliability. Although the current system is extremely cost effective and functional, the finished product does not have industrial quality.

The current container choice allows for high versatility between bottles and does not require cleaning of used containers. However, industrial containers differ within a single brand which does not account for entire companies. This variety creates problems when designing a product structure to house the container. A designed container for each beverage allows for structural planning of the final product and is required for a portable system.

Although the gravity induced dispensing system is inexpensive and effective, it only prevents the liquid from flowing if it is properly sealed. Due to user error, leaks can occur and create large, potentially expensive consequences. Furthermore, this system does not seal the liquid from the atmosphere which can result in contamination as well as fluid loss. Fluid loss can be fixed using a siphon, but contamination remains a problem if all the beverages are not used within their appropriate lifetime. The current system provides a proof of concept, but the mechanical architecture requires more planning and equipment for a finished product.

3.4 Hardware Architecture

The final design of the hardware architecture of Mr. Mix is depicted below and can be separated into three main modules, two of which are from the system architecture. The first module is the ESP8266 which effectively acts as command center of the remaining two modules. It is a low power device capable of wireless communication to send and receive information from the AWS server as well as outputting the proper data on the pins below to the corresponding module. The next two modules are the functional aspects of Mr. Mix.

The LCD HD44870 and pull-down button on the left provide a low-level user interface to provide basic information and operation of the device. Through this basic UI, a user will be able to read basic information and set certain device level settings. Furthermore, the button allows for immediate changes on the device or to indicate the readiness to pour the beverage.

Once the user indicates that the container is ready, the dispensing mechanism begins. The load-balancing cells measure the weight of the cup via the change in voltage across the bridge network. Then, the microcontroller triggers the component beverages via the PMOS switches which open the solenoids for flow until the weight has reached the value indicated by the microcontroller.

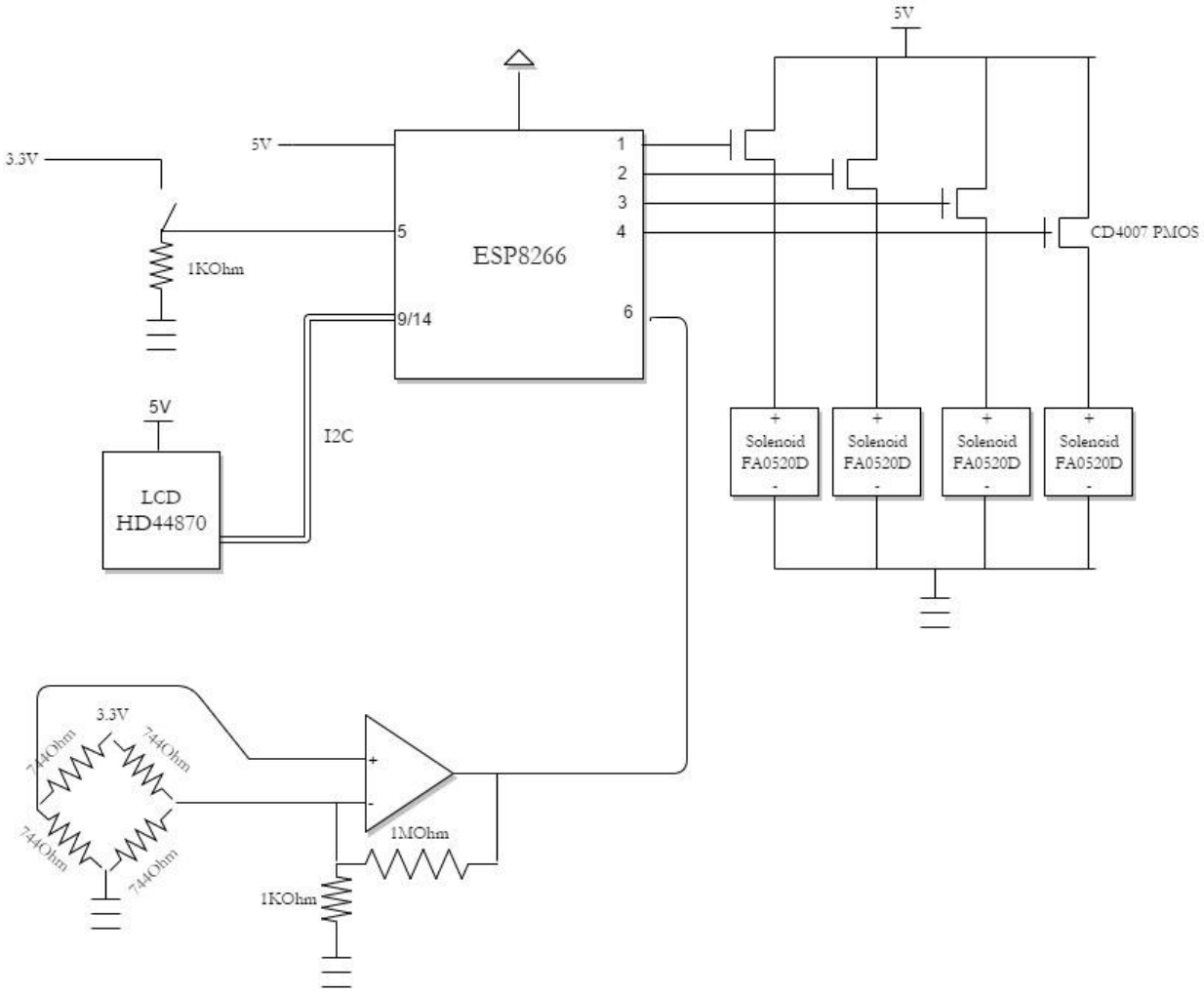

Figure 2: Mr. Mix Hardware Schematic

3.4.1 Hardware Components

ESP8266:

The ESP8266 is powered by a 5V input indicated in the diagram above. The wireless chip onboard is run off 3.3-3.6V which is also the HIGH voltage applied to the individual pins. Pins 1-4 are GPIO pins configured as outputs to trigger on and off via high and low respectively. Pin 5 is also GPIO, but configured as an input for basic input. Pin 6 is an ADC used to convert the analog values from the load cells to a digital value for computing the weight of the cup and beverage. Finally, pins 9 and 14 are configured with the protocol I2C to communicate with the LCD screen.

USER INTERFACE:

LCD:

The LCD HD44870 displays a maximum of 16 characters per line and has 2 lines of display. It is powered by a 5V power source and communicates with the ESP8266 using the I2C protocol.

BUTTON:

Due to the pull-down resistor, pin 5 is normally off to minimize power consumption. When the generic button is pushed, pin 5 receives the 3.3V input from the source.

DISPENSING:

CD4007 and Solenoid PA520D:

The PMOS transistors from the CD4007 function as electronic switches and their nominal switch value is 2.5V which is under the 3.3V supplied by the ESP8266. This turns on the switch with an approximate .2V drop from the 5V power source. The Solenoids operate at 4.5V and 60mA which is supplied by the transistor in the on state.

Bridge Network and Amplifier:

The bridge network is powered by a 3.3V source and measures the difference in voltage across the load bearing cells. This voltage is then amplified by an operational amplifier with a gain of 1000 to satisfy a large change in voltage for the onboard ADC which has 12 bits of resolution.

3.4.2 Hardware Evaluation

The design goals of the hardware architecture focus on low power and low cost which these devices satisfy. Unless utilized, most of the components remain in the off state not using any power. However, the ESP8266 will consistently use power in the form of wireless communication. Wi-Fi as a protocol consumes much more power than a low power Bluetooth device would. Despite, the power benefit of using Bluetooth, it does not have the functionality Wi-Fi possesses and thus the benefits of the capability outweigh the power cost.

To maintain cost efficiency, the user interface on the device is minimized to a low-level button and text interface. Similar products in industry use large, touch capable screens to interact and respond with user requests. This situation requires the personal device web service to handle complicated functionality and may hinder user experience.

The dispensing mechanism like the user interface requires more complex machinations to achieve functionality because of prioritizing power and cost. This electrical system only actuates air flow not pressurization which indeed requires less power, but a significantly more complex mechanical system to compensate for flow control. Overall, the hardware design yields a low cost and power product, but has multiple areas of improvement in the device user interface and dispensing mechanism.

3.5 Software Architecture

Mr. Mix is powered by combining a Wi-Fi enabled microcontroller with a powerful cloud backend on Amazon Web Services (AWS). The software is designed from the top down allowing each component to be modeled to work together efficiently and meet engineering requirements. However, the software is built from the bottom up to test smaller units individually before integrating with the larger architecture.

The backbone for the software is the AWS Internet of Things platform (AWS IoT). The platform provides a lightweight communication channel between the microcontroller systems and the backend systems hosted on the cloud. This design allows for all computationally intense tasks to be completed in a robust environment while allowing the microcontroller to interface with the hardware components. The AWS platform provides security, user management, on demand computation, cloud storage, and on demand databases. Using the AWS cloud Mr. Mix can scale the backend system with any number of devices on the network. This capability ensures that the software architecture can support the current requirements and any future development of the system.

Software for the microcontroller runs on Mongoose OS providing native integration with the AWS IoT Platform. Additionally, Mongoose OS provides easy APIs for connecting to wireless networks and creating ad-hoc wireless networks. The microcontroller software is built to efficiently receive messages from AWS and perform the requested action. Using Mongoose OS, the microcontroller software can meet all engineering requirements and provide seamless integration with the cloud backbone.

3.5.1 Backend Software

The backend software stack for Mr. Mix is built on the Amazon Web Services platform. The software stack is split into four main components which are built on top of the AWS Cloud. Leveraging the AWS cloud allows for simple integration between components as well as robust security features supporting fine grained access control.

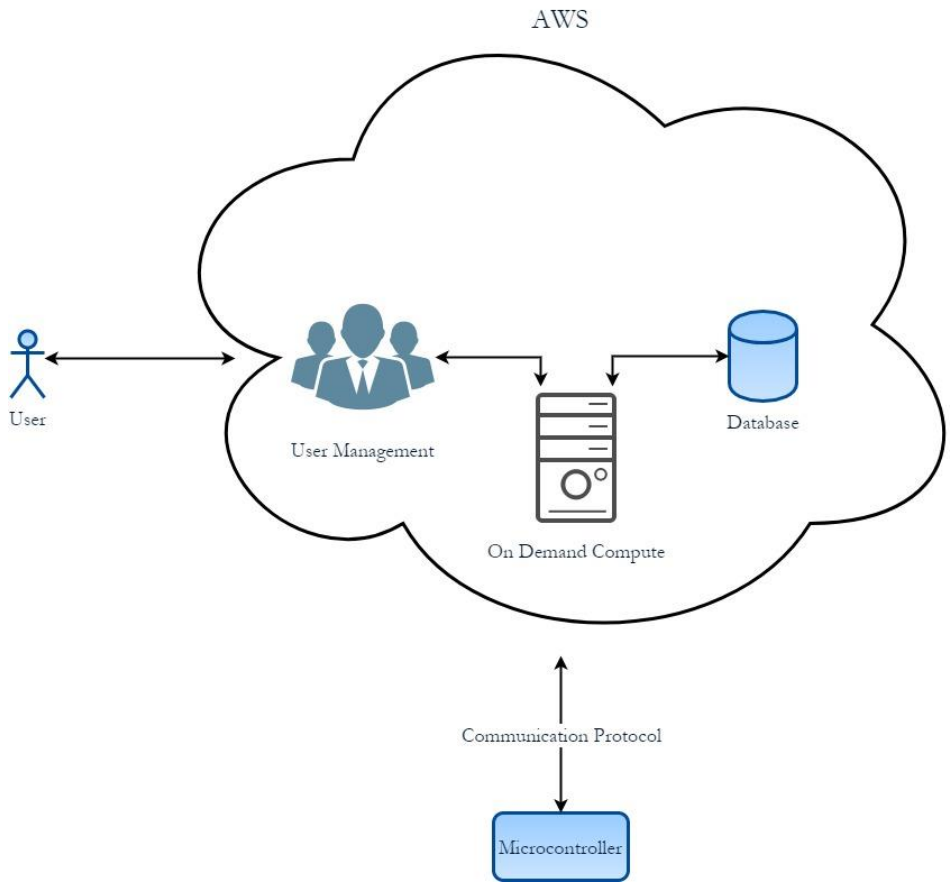

Figure 3: Mr. Mix Backend Software Components

Communication Protocol:

Communication between AWS and the microcontroller is built on the AWS IoT platform using MQTT. The protocol provides lightweight messaging support in a publish-subscribe model built on top of TCP/IP. The protocol also has low network bandwidth and requires only a small code footprint. In addition to the support within AWS and Mongoose OS MQTT is ideal for Internet of Things devices such as Mr. Mix.

The messages sent to the microcontroller over MQTT utilize QoS 1 meaning that all messages will be delivered to the device at least once. This does mean that on the microcontroller software some level of deduplication must be performed. However, because most of the communications occur on at least the order of seconds the overhead for the microcontroller is minimal.

Messages sent to Mr. Mix contain a single JSON object which conforms to a strict specification.

```
{
   eventId:"7ae3859d-acc1-49dd-be82-12d204502f88", //Will be a unique UUID for the message
   messageType: 0, //Valid values are 0, 1, or 2
   payload: {} //Contents will differ based on messageType
}
```

Figure 4: JSON specification for Mr. Mix

This specification allows for easy deduplication of messages and the ability to send multiple message types to the same processor. Currently Mr. Mix supports three message types; however, the specification allows to more in the future if needed.

On Demand Compute:

The backend computations needed for Mr. Mix is relatively low most of the time, but can spike during peak load. The servers required to support peak traffic can be costly even though the CPU is idle most of the time. To solve this problem the backend software leverages AWS Lambda, an on-demand compute service. This allows for different capabilities of the backend services to be written as individual functions and spun up on demand. The charges for functions hosted on AWS Lambda are only incurred while the function is running. During peak load, many parallel threads are spun up to handle higher demand.

Database:

Mr. Mix stores all data in Dynamo DB, a No SQL database within the AWS ecosystem. Using a No SQL database gives the backed systems capability for changing the data model as the system evolves without needing to redesign the entire database. Dynamo DB allows each table to define a hash key and an optional range key within the JSON object. The hash key and range key are then used for efficiently storing and looking up the requested objects in the database. There is also capability for creating secondary indexes within the data as well, however the current data model does not take advantage of the feature.

The data model stored by Mr. Mix is stored across five tables within Dynamo DB.

```
{
   userId:"44e91c25-c711-4f39-a759-5a990f50ed2e", //Hash Key
   registeredDevices: ["f54144ae-dbcc-4b46-b3f0-e7b960e512c2"]
}
```

Figure 5: User Table Data Model

The User table stores information about all registered users who own a Mr. Mix device. The userId will be unique and generated by AWS Cognito. When users sign up through Cognito an accompanying database entry is created in Dynamo DB so the backend software can store additional data about the user. Currently, the only additional data stored is a list of devices registered to the user. This allows the user to manage devices they own and change device configuration, update available mixers, or manage the active session.

```
{
   deviceId: "f54144ae-dbcc-4b46-b3f0-e7b960e512c2", //Hash Key, stored in device firmware.
   activeSession: "SN0V7C", //Optional, will be an empty string if there is no active session
   ownerId: "44e91c25-c711-4f39-a759-5a990f50ed2e",
   mixers: [
      {mixerId: "2ec6f390-16cb-4791-836a-142b1ecb672c", tapNumber: 0},
      {mixerId: "dc756f79-446d-4873-b2c8-e874324b6e31", tapNumber: 1}
   ],
   recipes: ["5d247e9b-54be-4712-9ebf-bd8c10a59453"], //Precomputed when the mixers are updated
   configuration: {} //Stores any configuration information for the device
}
```

Figure 6: Device Table Data Model

In the Device table information is stored about a specific Mr. Mix device. The deviceId is stored on each device during production and will be unique across all devices. Additionally, each device can only have one activeSession or owner at a time which will be enforced by the backend systems. When the mixers for the device are updated the recipes will be precomputed based on all available mixers. This occurs by looking up all potential recipes for each mixer present and filtering out any options where all ingredients are not present. This can be a computationally intense task and will save time overall versus calculating options every time a request is made by the user. The configuration field can be used to store any configuration information that is also stored on the device.

```
{
   sessionId: "SN0V7C", //Hash Key, must be 7 alphanumeric characters
   sessionStart: "2017-06-10T22:23:46Z", //ISO 8601 Timestamp in UTC
   isActive: true,
   deviceId: "f54144ae-dbcc-4b46-b3f0-e7b960e512c2"
}
```

Figure 7: Session Table Data Model

Metadata about every session is stored in the Session table. The sessionId is a unique seven-character alphanumeric code. This will be displayed to the user on the LCD screen informing them to go online and request a drink. The software can have over 78 billion unique sessions before the design is no longer viable. If 1 million devices created a new session once a day it would take over 214 years before the system ran out of unique Ids. The table also stores information about when the session starts and if is currently the active session on the device.

```
{
   mixerId: "2ec6f390-16cb-4791-836a-142b1ecb672c", //Hash Key
   name: "Vodka",
   type: 0, //0 for alcoholic 1 for non-alcoholic
   usedIn: ["5d247e9b-54be-4712-9ebf-bd8c10a59453", "b0ffb70f-b838-4662-ab30-73371c94a718",
"b1606360-3294-44d5-8518-ea713f7ff536", "094a1b28-82fe-4fd2-a49e-af6f8aeda455", "02ff587c-8d2e-42a2-
8fec-6867757ed036"] //All recipes containing the mixer
   description: "A good social lubricant"
}
```

Figure 8: Mixers Table Data Model

The Mixer table is used to store information about different mixers supported by Mr. Mix. The usedIn field stores all recipes in the database where the ingredients contain the mixer. This allows for easy lookup into the Recipe table given an included mixer.

```
{
   recipeId: "5d247e9b-54be-4712-9ebf-bd8c10a59453", //Hash Key
   name: "Vodka Cranberry",
   description: "A cocktail classic",
   ingredients: [
      {mixerId: "2ec6f390-16cb-4791-836a-142b1ecb672c", proportion: .25},
      {mixerId: "dc756f79-446d-4873-b2c8-e874324b6e31", proportion: .75}
   ], //All proportions must add up to 1
   finishing: "Garnish with lime"
}
```

Figure 9: Recipe Table Data Model

Storing the Recipe information this table is used to keep track of default proportions for various cocktails. The proportions are used so the recipe can be scaled to any drink size within the engineering requirements. Additional information for finishing the cocktail is also included allowing the user to garnish their drinks appropriately after Mr. Mix has properly poured.

User Management:

The user management for Mr. Mix is powered by AWS Cognito. This service provides pre-built secure storage and validation for user credentials. AWS Cognito allows the backend software to abstract away the complications of safely storing user credentials and validating them later. The AWS APIs allow for the software to prompt a user to log in and then use tokens provided by AWS Cognito for access to the authenticated sections of the website.

3.5.2 Microcontroller Software

The software running on the ESP8266 is powered by Mongoose OS. Mongoose OS provides an operating system built to run on the ESP8266 with native support for AWS IoT integration. The operating system supports all native capabilities of the board, but builds on them and provides powerful functions for configuring wireless networks, connecting to AWS, GPIO, SPI, UART, and I2C.

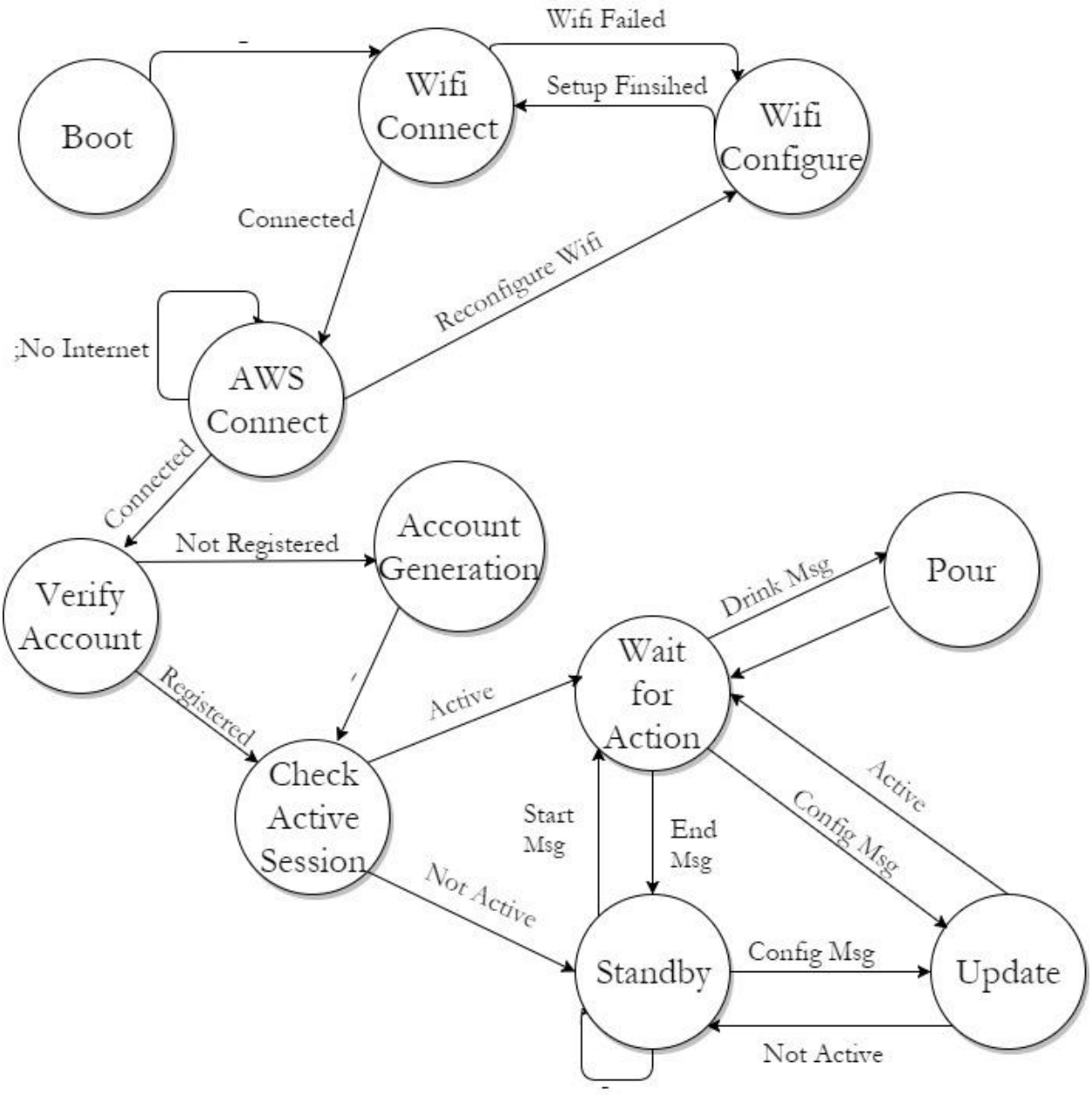

Figure 10: Microcontroller State Diagram

The state diagram can be broken into two distinct components the setup phase and the message processing phase.

During the setup phase the microcontroller ensures that wireless configuration is available and that AWS can be reached successfully. If the wireless network needs to be configured an ad-hoc wireless network is created and the user can connect to the board to complete setup. After the board successfully connects to AWS, the device verifies that a user has registered the devices. If the device has not been registered, it prompts the user to create an account and register the device.

In the processing phase the software waits for incoming messages from AWS and processes them accordingly. When a message comes into the processing unit the messageType is checked and the appropriate processing code is executed. After successfully processing a message the devices goes back into the standby state and waits for a new message to be available for processing.

3.5.3 Software Evaluation

The software for Mr. Mix is developed to be maintainable and scalable. This design successfully demonstrates these capabilities and could easily be extended in the future. The components are encapsulated properly and do not have confusing dependencies to make an unmaintainable platform.

Some of the design decisions around the database could be improved so there is less computational overhead. Although because the backend software leverages the cloud this overhead is achievable at a reasonable cost. This could be improved in the future to create a better product overall.

4.0 System Integration Testing

4.1 Testing Plan

In order to ensure the final design complies with the original goals and objectives, the below Testing Plan analyzes the functionality of the final product to make sure it fully operates as specified in the engineering requirements. The main focus of this test plan covers the four main aspects of the engineering requirements: effective pouring, proper configuration, simple user interface, and low cost.

Test Number	Engineering Requirement	Test	Success Criteria
1	1.1	Check number of working dispensing valves	At least two valves work
2	1.2	Check that valves can be individually controlled	All valves function separately
3	1.3.1	Check smallest dispensable amount	Amount is 8 ounces
4	1.3.2	Check the largest dispensable amount	Amount is 20 ounces
5	1.3.3	Dispense any amount of liquid within the given range	Dispensed amount is accurate within .1 ounces
6	1.4	Attempt to pour liquid without cup present	Liquid should not pour
7	1.4	Attempt to pour liquid with cup present	Liquid should pour
8	1.5	Attempt to pour liquid without pressing button	No liquid should pour
9	1.5	Attempt to pour liquid by pressing button	Liquid should pour
10	2.1	Attempt to alter the device's wireless network	Network should update
11	2.2	Attempt to register a new device to an account	Device should register
12	2.2	Attempt to register an already registered device to an account	Registration should fail
13	2.3.1	Send a message to the device	Devices processes message quickly
14	2.3.2	Send a message to the device after a long period of inactivity	Device processes message quickly
15	2.4	Power cycle device	Ensure device state remains constant
16	3.1	Start a new session	Ensure code is unique
17	3.2	Start a new session	Ensure old session is inactive
18	3.3	End a session	Ensure session no longer works

19	3.4	Start a session	Ensure session is created
20	3.5	Have an ongoing session	Ensure session code is displayed on LCD
21	3.6	Log into an existing session	Ensure an order can be placed
22	4.1	Plug in device	Ensure it is connected to a standard outlet
23	4.2	Check device power consumption	Average power consumption is under 3A
24	4.3	Check device size	Ensure device can fit on countertop
25	4.4	Relocate the device	Ensure the device can still function properly

4.2 Results and Verification

As Mr. Mix is only in prototype stages the testing has not been performed on the prototype. When Mr. Mix is implemented outside of a prototyping environment the testing can be completed and compared against the required results. As part of the future directions for Mr. Mix the results against the testing plan will show how close the final implementation is to meeting the engineering requirements.

5.0 Challenges

The final implementation of Mr. Mix is so far unsuccessful and no working prototype has been produced. Along the way, many challenges with the implementation details have been faced. Given more time and resources, they may have been overcome. Some of the individual components were successful, however, the entire system did not interface as one. If future time was allotted to development of Mr. Mix, a working prototype based on this design could be successfully created.

5.1 Implementation Problems

1. Solenoid
   The solenoid currently in use for the implementation is the incorrect model necessary to function from the switching circuit. The FA0520D model requires 6V and .1A in order to be properly triggered. However, the PMOS switching circuit is capable of producing a max of roughly 4.7V. This situation prevents the solenoid from being triggered by the transistor switch.

2. ESP8266
   The datasheet appears to have documented all the appropriate functionality required of it as per the original design. The pinout diagram, however, is ambiguous. It has the physical pins matched to their functionality in the diagram, but no corresponding references to their representation in JavaScript. This issue means that each of the pins has to be tested to confirm its representation in JavaScript.

3. Mongoose OS
   Mongoose OS proves to be a very lightweight system which is extremely useful for a state operated callback design. As a lightweight system, it is prone to freezing and communication errors with the ESP8266. Depending on the state the microcontroller was left in prior to disconnecting, it requires several reboots to regain effective communication.

5.2 Attempted Solutions

1. Solenoid
   a. Testing Solenoid for Functionality
      As a proof of functionality, the solenoid properly operates when powered by 6V and up to .1A. However, these values surpass the capabilities of the transistors currently in use in the design.

   b. Powering Solenoid Directly From ESP8266
      The ESP8266 is able to power 3.3V from any of its GPIO pins. As a test of last resort, an attempt was made to power the solenoid directly from the microcontroller, but the current draw was far too large for operation.

2. ESP8266
   a. Matching pins to pin representations in JavaScript
      Trial by error mapping of pin port numbers to their values in JavaScript. This test is extremely time consuming without the additional complication of the class interfaces

utilized by Mongoose OS. This platform groups the pins by functionality and has multiple mappings to the same value depending on the chosen functional grouping.

3. Mongoose OS
    a. Multiple Reboot
    The only short term choice for developing on this system is to reboot the application and reattempt communication. From there, proper maintenance of the connection is vital to any form of progress.

5.3 Areas of Success

1. ESP8266 and Wi-Fi
Despite all the complications utilizing Mongoose OS on the ESP8266, a primitive UI to configure the Wi-Fi settings on the ESP8266 operates reliably. Once the settings have been configured, the ESP8266 appropriately reboots and connects to the location Wi-Fi.

2. ESP8266 and AWS
After connecting to the local Wi-Fi, the ESP8266 waits for commands from the AWS server. Another wireless device can publish information to the AWS server. The server then forwards this information and will be printed out in the console from the ESP8266.

3. GPIO and ESP8266
Although the pinout diagram is ambiguous, there are set of commands that retrieve the port number of the on board button and LED. This functionality can be used test the logical operation of the LCD screen once it is working.

4. Mechanical Dispensation Actuation
As a proof of concept of the mechanical design, the solenoid prevents leakage while in its normal off state. Once triggered, the solenoid allows air flow equalizing the pressure and dispensing the fluid. This operation is to be expected based on the mechanical design set forth in this document.

5.4 Next Steps

1. Solenoid
The appropriate solution is to purchase the FA0520B model of the solenoid which only requires 4.5V for operation. This fits well within the boundaries of the transistor switching network and would be easily powered by a 5V rail.

2. ESP8266 Pinout
The pinout must be either manually tested or find the right mapping via another datasheet. Once the pinout has been found, the LCD screen as well as GPIO pins can be implemented for full device functionality.

3. Interface with the load cells

After the dispensing mechanism is fixed, the next aspect of implementation will be to properly calibrate the load cells. This implementation requires that the operational amplifier amplify the smallest voltage difference in the load cells larger than the voltage resolution on the onboard ADC. Once the ADC is receiving voltage measurements, calibration needs to occur to properly measure out the ounces requested by the user.

4. Build the user interface
   The user interface provided by the AWS servers need to be tested for full functionality against all the operating units.

6.0 Conclusion

Mr. Mix as a senior project teaches that the most important phase of any large, time-consuming venture is design. Many of the preliminary prototypes although imperative for brainstorming failed due to the lack of forethought and design. After experimenting with failure, a set of requirements that Mr. Mix must accomplish narrowed down the possible implementations and provided a foundation to begin designing. Mr. Mix was first organized using a top down scheme which produced the main encompassing modules. These modules provide a highly modular design allowing for replacement or improvement to occur with little effect to the surrounding components. The final design of Mr. Mix accomplished the original main requirements set forth at the earliest stages of planning.

Despite the success of the design, the implementation comparatively is not. The design phase took longer than expected especially planning the details of each module. This delay propagated to the purchasing of components and assembly. Even with a descriptive design, implementation and debugging take a comparative amount of time to have a fully functioning product. While the implementation failed, the knowledge gained from designing and debugging was invaluable. If more time were allotted some of the implementation challenges could have been overcome and a functional prototype could have been produced.

The main lessons taken from this project are to design as much as possible in the beginning with a solid set of requirements as foundation as well as to begin implementation of the designed module concurrently with designing. In this way, progress occurs much more rapidly and testing can occur at an early stage allowing effective changes in future design aspects. This project proves that Mr. Mix is a viable product per this design. With an appropriate business model, Mr. Mix would be ready for production.

7.0 References

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
https://aws.amazon.com/documentation/iot/

8.0 Appendices

8.1 Bill of Materials

Item No.	Part Description	Part No.	Supplier Name	Quantity	Price	Extended Price
1	Rubber Stoppers		WidgetCo.com	4	$ 0.56	$ 2.24
2	Digital Scale	WH-B05	Amazon	1	$ 11.50	$ 11.50
3	Pressure Relief Solenoid	FA0520D	Amazon	4	$ 5.08	$ 20.32
4	10 Ft food grade tubing		Amazon	1	$ 5.93	$ 5.93
5	Microcontroller	ESP8266	Amazon	1	$ 8.79	$ 8.79
6	LCD Screen W/I2C Interface	HD44780/UC146	Amazon	1	$ 8.79	$ 8.79
7	Momentary Push Button	DS316	Amazon	1	$ 0.73	$ 0.73
8	Transistor Kit		Cal Poly	1	$ 6.25	$ 6.25
					Total:	$ 64.55