Automated Boba Station

ECE 445 Design Document - Spring 2019

Team 49 - Hunter Huynh, Timothy Ko, Jordan Wu

TA: John Kan

---

1 Introduction **1**

1.1 Objective 1

1.2 Background 1

1.3 High-level Requirements 2

2 Design **2**

2.1 Block Diagram 3

2.2 Physical Design 4

2.3 Power Supply 4

2.4 Liquid Dispensing Mechanism 5

2.4.1 Solenoid Valve 6

2.4.2 Solenoid Control 6

2.5 Solid Dispensing Mechanism 7

2.5.2 Servo and Door 8

2.5.2 Servo Control 8

2.6 Weight Sensing Mechanism 9

2.6.1 Load Sensor 9

2.6.2 Load Sensor ADC 10

2.7 Control Unit 11

2.7.1 Microcontroller 11

2.8 Software 13

2.9 Tolerance Analysis 17

3 Costs and Schedule **18**

3.1 Cost Analysis 18

3.1.1 Labor 18

3.1.2 Parts 18

3.2 Schedule 20

5 Ethics and Safety **21**

5.1 Ethics 21

5.2 Safety 21

6 References **22**

1 Introduction

1.1 Objective

Boba, a popular drink among millenials, has prices that are still largely dictated by the manual labor involved in making it, so shops still require many employees. Unlike coffee, making boba tea requires handling both solids (boba, etc), and liquids (tea, syrup, milk) [1]. With the large variety of recipes, human workers are prone to make mistakes. Finally, taste consistency is hard to achieve without an automated solution, leaving drinks sometimes oversweet.

Our goal is to develop an automated boba station would have multiple dispensers connected to a U-shaped gutter that would all drip into a cup. We would have multiple dispensers for cold liquids and solids for toppings, simplifying the entire process of creating a boba drink, which will improve the efficiency of boba stores. In addition, because boba is currently made manually, you most likely either make too much or make too little for the cup size. This product will dispense predetermined amounts for each liquid/solid, customizable by a web user interface, reducing food waste and money.

1.2 Background

The creation of a boba drink comes in a multitude of configurations. However, it generally consists of just solids and liquids. In general, the process of creating a boba drink consists of putting the toppings (solids) specified by the order in the cup, pouring the tea and milk (milk tea may already be mixed together with a specific concentration of milk and tea), then adding the user specified amount of sugar and ice (such as 0%, 25%, 50%, 75%, or 100% of the normal amount) [1] [5]. For this project, we will just focus on the basic drink, milk tea with boba, which consists of cold brewed tea (black tea or green tea), milk (almond milk, regular milk, etc), sugar syrup, and tapioca pearls (boba).

Because the creation of boba drinks is currently all manual and based off of percentages, there is no unified concentration of each component of the drink. For example, 25% sugar level varies employee to employee and drink to drink because it is done manually. The concentration of milk and tea varies as well as the amount of toppings is in a drink. If done automatically, the concentrations will be unified per configuration. Eventually, we'd like this solution to allow for more permutations of drinks by adding in more dispensers with different ingredients and connecting it to this system.

1.3 High-level Requirements

- Have at least two dispensers, one for tapioca pearls and one for the milk tea.
- The station must be able to dispense a user-specified amount of tea, sugar syrup, milk, and boba with no more than ±10% error in mass.
- Have a web interface to control the amount of liquids/solids dispensed in each dispenser.

2 Design

This automated boba station will have two dispensers: one for tapioca pearls and one for milk tea. We will be using a “revolving door” to dispense the tapioca pearls and a solenoid valve to dispense the milk tea. An employee can manually refill the containers holding the milk tea and tapioca pearls. There will then be a sloped gutter the dispensers will dispense to, which direct the ingredients into a cup placed at the bottom. The cup will be on a small raised platform with a load sensor underneath, which is connected to its own microcontroller. Each dispenser will be powered by 12 V and controlled by a microcontroller. All microcontrollers will have a Wi-Fi chip and a simple web server listening for instructions. One of our computers will be then controlling the microcontrollers for each dispenser through sending HTTP requests as well as open up a web server to serve the web UI to control the amount of liquids/solids dispensed in each and when to begin the boba making process.

2.1 Block Diagram

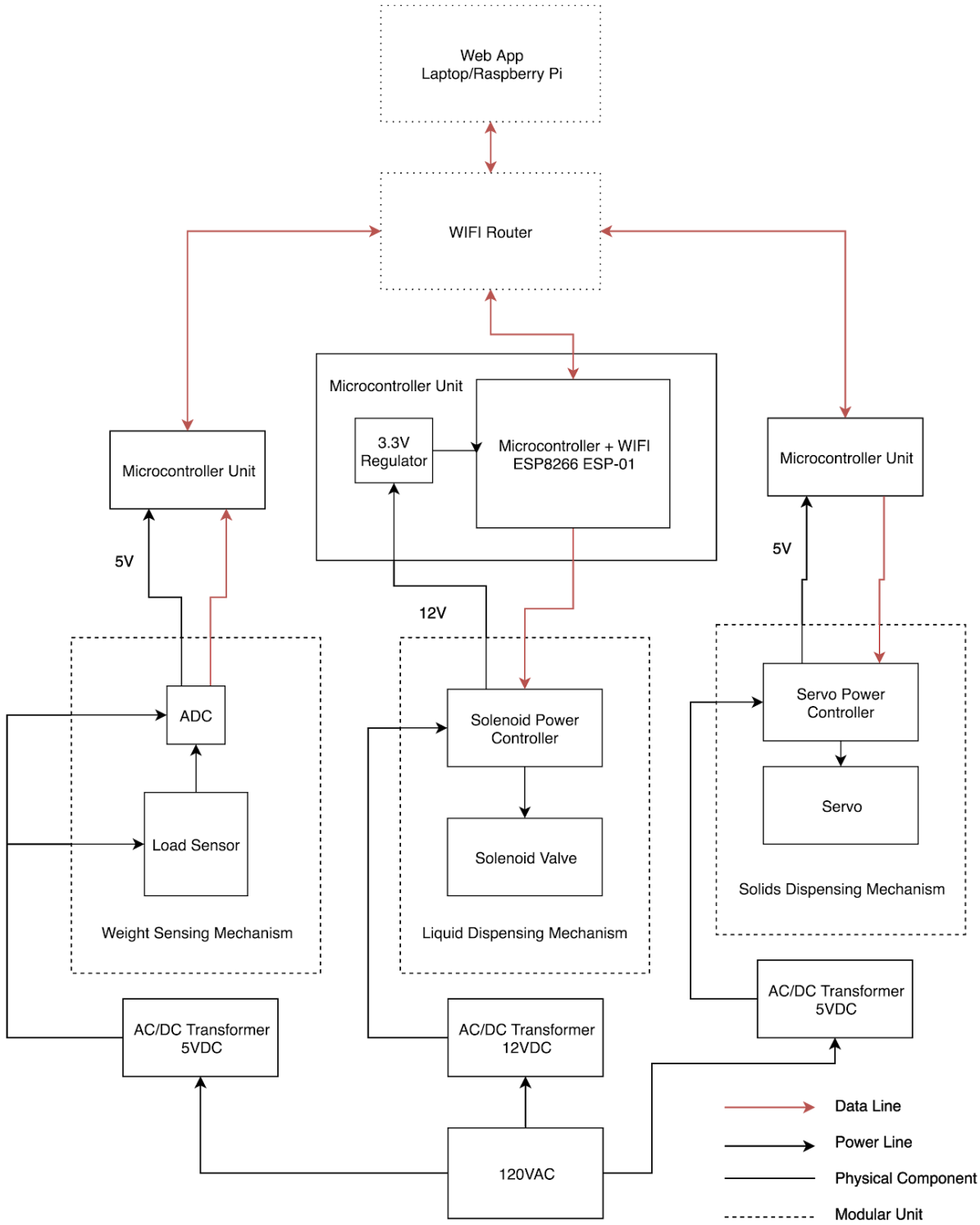

Figure 1. Block Diagram

2.2 Physical Design

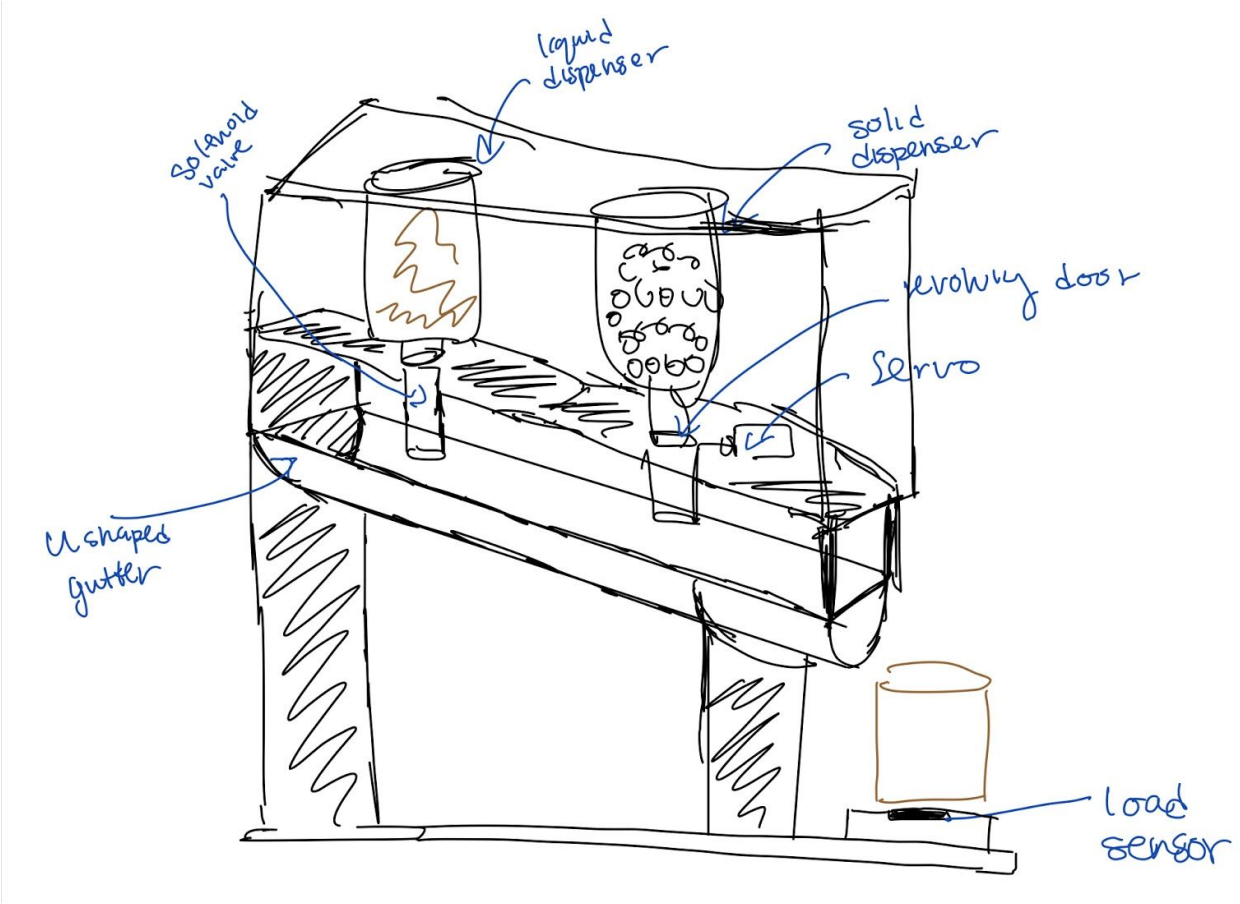

Figure 2. Physical Design Sketch

2.3 Power Supply

2.3.1 120 VAC to 12 VDC Transformer

The power supply provides the circuit with 12 V at all times. The power supply would be a wall converter, converting the 120 AC to 6 A 12 V DC. This would sustain our liquid dispensensing mechanisms and its microcontrollers with consistent power.

Requirement	**Verification**
1. Converts 120 VAC to 12 VDC.	1A. Connect the input of the power supply to a wall outlet outputting 120 V AC. 1B. Measure the output voltage using an oscilloscope. Check that the output voltage is in the range 12 V +/- 5%.

2. Maintains a safe temperature below 140°C for at least 99% of the time.	2A. Check that the power supply's temperature is below 130°C using an IR thermometer during verifications 1A and 1B.

2.3.1 120 VAC to 5 VDC Transformer

This power supply will provide 5 V DC to the solid dispensing mechanism and weight sensing mechanism.

Requirement	**Verification**
1. Converts 120 VAC to 5 V DC.	1A. Connect the input of the power supply to a wall outlet outputting 120 V AC. 1B. Measure the output voltage using an oscilloscope. Check that the output voltage is in the range 5 V +/- 5%.
2. Maintains a safe temperature below 140°C for at least 99% of the time.	2A. Check that the power supply's temperature is below 130°C using an IR thermometer during verifications 1A and 1B.

2.4 Liquid Dispensing Mechanism

The liquid dispensing mechanism is a core part of our automated boba station since it dispenses the necessary liquids in order to make a milk tea drink. The amount of liquids dispensed and when to dispense will be determined by the main server.

Requirement	Verification
1. The dispensing mechanism must dispense 60 mL +/- 5% milk tea in 1 second of opening.	1. Connect it to a microcontroller and open it for one second. 2. Measure whether the amount of dispensed material is accurate.

We approximated that 60mL amount of milk tea would dispense for one second of an ½ inch NPS (nominal pipe size) open valve. This was done by making an estimated 1/2 diameter hole with our hand over a cup and pouring a water bottle directly (upside down, like how it would be in our machine since the solenoid valve is gravity fed) into the hole for two seconds. We then measured the water level and divided it by two, since we did it for two seconds. There was around ¼ cup of water, which is around 60 mL. This obviously needs some fine tuning as we

don't actually have the valve. So once we have the valve, we would do the same verification steps and measure the amount of liquids dispensed as well as get its weight.

2.4.1 Solenoid Valve

This will be used to control whether or not liquid is being dispensed. It should be normally closed when no voltage is applied, and open when sufficient voltage is applied across its terminal. The solenoid valves we will be using only opens when 12 V DC is applied. Therefore, the GPIO (General-purpose input/output) signals are unable to directly open the valves.

Requirement	**Verification**
1. Valve must open when 12 V +/- 5% is applied.	1A. Apply 12V DC to the valve's input. 1B. Ensure that the valve opens fully within 1 second of verification 1A.
2. Valve must be gravity fed (allow flow of liquid by gravity alone when open), as no other liquid pressurizing mechanism is in place.	2. During verifications 1A and 1B, place a liquid container above the valve and attach it. Ensure that liquid both enters and exits the valve.

2.4.2 Solenoid Control

The solenoid control, Fig. 3, will amplify the GPIO signal from a microcontroller to 12V via a transistor so that the solenoid valve can be opened and closed.

Requirement	**Verification**
1. Must be able to output 12 V +/- 5% and 3 A +/- 5% to the solenoid valve.	1A. Apply 3.3V to transistor base. 1B. Measure the output voltage and current using an oscilloscope. Check that the output voltage is in the range 12 V +/- 5% and check that the output current is in the range 3 A +/- 5%.

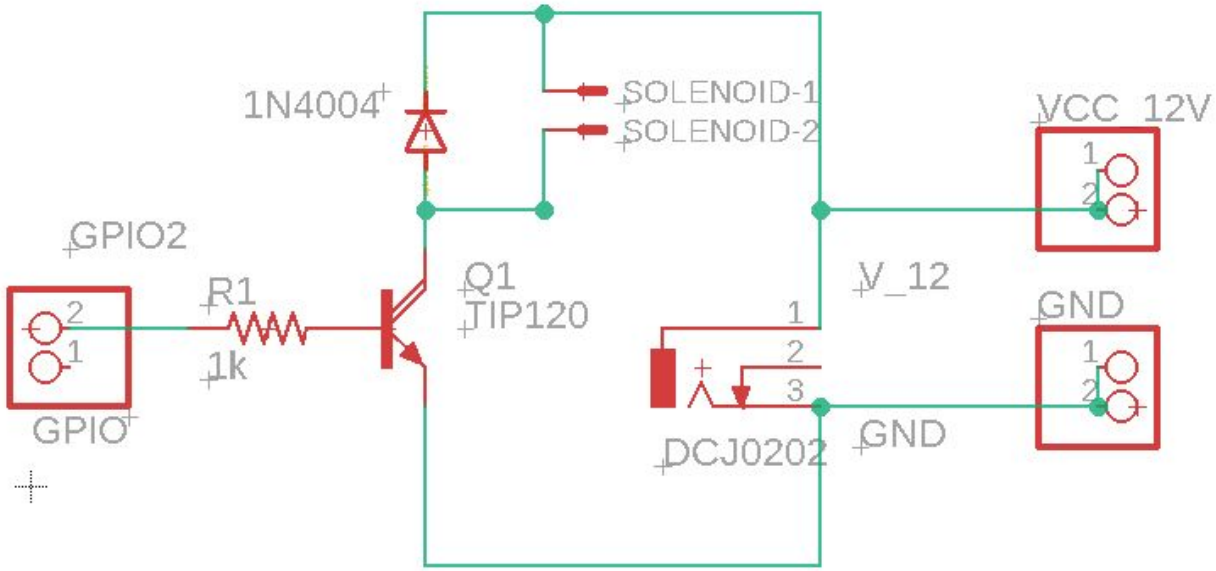

Figure 3. Solenoid Control Circuit Diagram

2.5 Solid Dispensing Mechanism

The solid dispensing mechanism is used to dispense the topping, tapioca pearls. After user research (we interviewed friends who worked at boba places such as Latea), we learned that the boba pearls needed to be covered with honey right after they boiled and didn't need to be held in a liquid solution (water or sugar water). Thus, we are planning on having a "revolving door" driven by a servo to it to open and close it as part of the solid dispensing mechanism.

Requirement	Verification
1. The solid dispensing mechanism must dispense 2 +/- 1 tapioca pearls during 1 second of opening.	1A. Connect it to a microcontroller and open it for one second. 1B. Measure whether the amount of dispensed material is accurate.

We approximate two tapioca pearls dispensed per second. According to the American Key Food Products, tapioca pearls are around 4 mm [9]. We are using a 1 inch PVC tube for the door (described below), we know the maximum amount of pearls that can be pushed through is six since an inch is 25 mm. However, tapioca pearls may be bigger and six means that all pearls must be aligned properly. Thus, we can safely say that there would be an average of 2-3 pearls passing through at every turn, which would occur every one or so seconds. We will be testing and flushing out the correct amount of tapioca pearls that are dispensed per second. To do

accomplish this, we would set it up and open the door for one second. Then we will measure the amount of tapioca pearls that were dispensed.

2.5.2 Servo and Door

The "revolving door" will be a 1" Diameter PVC tube with a hole drilled on on one side facing upwards. The solids will be dropped into the tube from a chamber above via gravity. The servo will then rotate the tube 180 degrees so the hole faces downwards, allowing the solids to fall out via gravity while sealing the chamber above. The prescribed Towerpro SG92 Servo provides 2.5 kg of torque, which should be sufficient.

Requirement	Verification
1. Servo must provide enough torque to rotate the tube.	1A. Connect the servo to the PVC tube and apply a 1ms pulse of 3.3 V DC to the servo. Ensure that the PVC tube rotates easily.
2. Servo must be able to rotate 180 degrees.	2A. Repeat verification 1A, but with a pulse of 2.4ms. Ensure that the PVC tube rotates any amount greater than or equal to 180 degrees.
3. Door must be 1" wide.	3A. Measure the door with ruler and verify whether it's 1" wide.

2.5.2 Servo Control

The servo can be controlled directly via PWM from the microcontroller. The logic level will need to be stepped up from 3.3 V to 5 V via a BSS138 logic level shifter.

Requirement	Verification
1. Servo control signal must provide 5 V PWM.	1A. Apply 3.3 V PWM to GPIO_2 on servo control board. B. Measure Servo pin 1 with oscilloscope. Ensure 5 V PWM peaks. C. Ensure pulse width measured matches input pulse width.

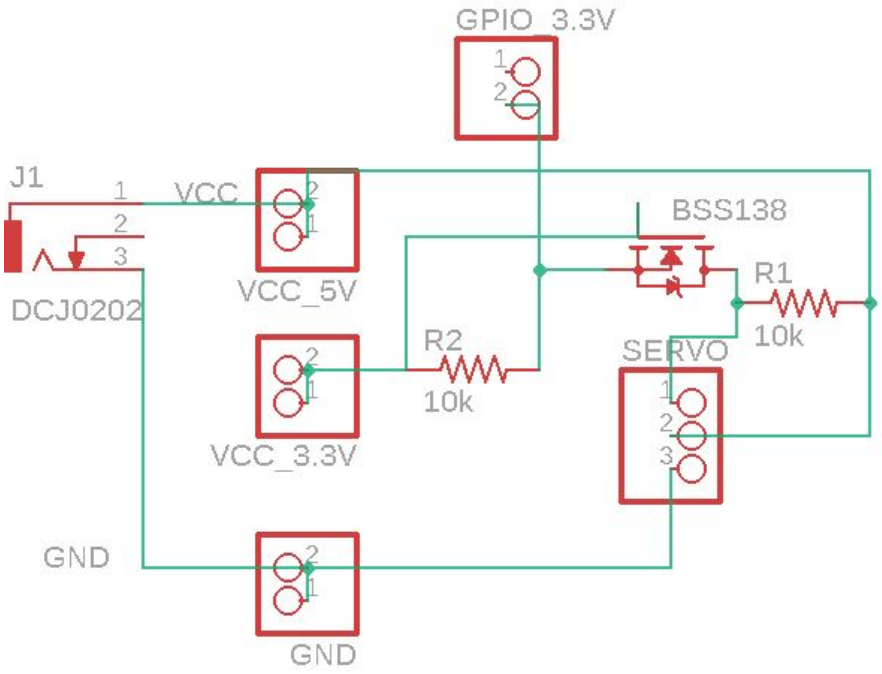

Figure 4. Servo Control Circuit Diagram

2.6 Weight Sensing Mechanism

The weight sensing mechanism is used to measure the weight of the cup and liquids/solids dispensed in it. This data will be used in coordination with the dispensing mechanism to control how much of that certain ingredient we have already dispensed and need to dispense based on the configuration. We will be measuring the weight of pre-measured amount tapioca pearls to find the relation between N amount of tapioca pearls and its weight in grams. Milk tea is still very close to the weight of water, where 1 mL of water is equal to 1 gram (unit conversion). Milk is 1.03 mL/g due to the extra nutrients and we can estimate that Milk tea would be around 1.02 mL/g since it isn't fully milk.

2.6.1 Load Sensor

The load sensor will be mounted under the platform the cup sits on and outputs an analog signal. We can use the TAL221 load sensor for this purpose, which outputs voltages 0.7 V +/- 0.15 V [4].

Requirement	Verification
1. Sensor must be accurate to > 0 g and < 5 g to reliably measure liquids. Sensor must	1A. Plug in load sensor to microcontroller 1B. See if digital output corresponds with increasing pre-measured weight

support up to 500 g, the typical weight for a drink.	

2.6.2 Load Sensor ADC

Because load cells have a very small resistance change, most devices can't actually detect the change so we are going to need another device that can take the very small change in resistance and turn it into something we can measure accurately. This is where an HX711 will come into play, as it functions both as an ADC and an amplifier. It must be able to read an analog signal outputted from the load cell and amplifies it and converts it to a digital signal via an ADC to be read by the microcontroller.

Requirement	Verification
1. Must be able to read voltages 0.7 V +/- 0.15 V from a load cell and convert it to a digital value.	1A. Connect the ADC circuit to the load cell and to a microcontroller. 1B. Apply 24 pulses to pin CLK. The microcontroller should obtain a 24-bit number from the DATA pin proportional to the weight on the load cell.

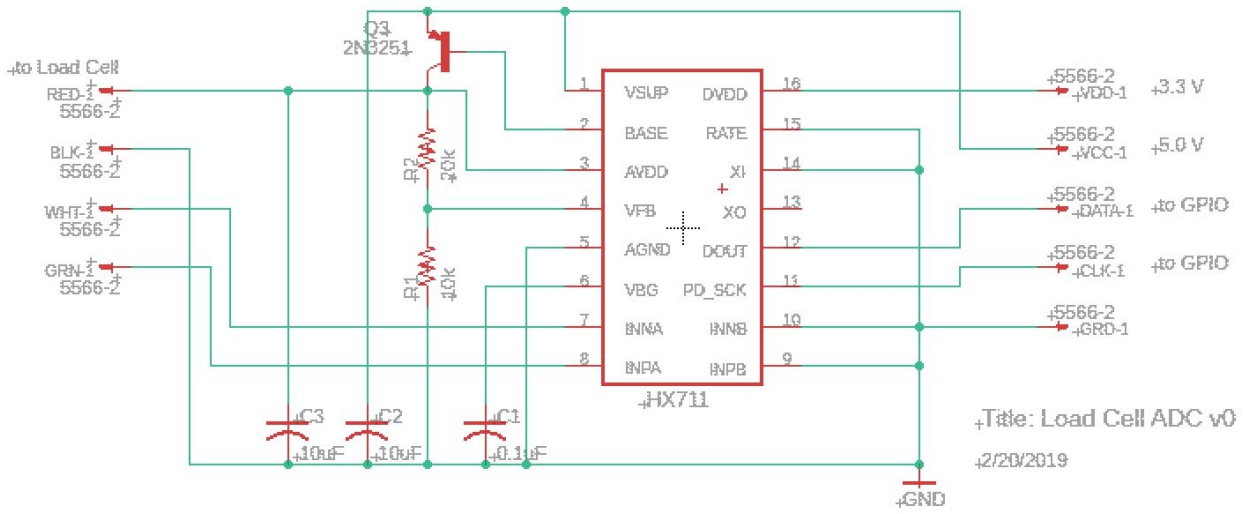

Figure 5. Load Sensor ADC Circuit Diagram

2.7 Control Unit

The control unit controls the different dispensing mechanisms, when to open and close as well as takes input from the weight sensing mechanism to make decisions. It will store the configurations set by the user through web UI it serves and manage the system accordingly. For more information on the high level architecture and communication, look below to the Software Section 2.8.1.

2.7.1 Microcontroller

Our microcontrollers will receive requests through Wi-Fi and perform the corresponding action on the mechanisms. Each of the mechanisms will have its own microcontroller, receiving or sending data through GPIO 2 on the microcontroller. We currently plan to use the ESP8266 (ver. ESP-01). Control of the solid dispenser servo will be through PWM, while the liquid dispenser gets a "ON/OFF" based on logic level 1/0. Programming will be done via an external Arduino to act as a USB to serial converter. [13]

Requirement	Verification
1. Microcontroller connected to network and can communicate with laptop.	1A. Ping microcontroller IP from laptop. B. Verify response
2. Set a HIGH signal (3.3 V) on GPIO 2 when a Request "ON" is sent to the microcontroller. Respond "ACK".	2A. Send "ON" to microcontroller from laptop B. Probe GPIO 2. Check for 3.3 V
3. Set a LOW signal (0 V) on GPIO 2 when a Request"OFF" is sent to the microcontroller. Respond "ACK".	3A. Send "OFF" to microcontroller from laptop B. Probe GPIO 2. Check for 0 V
4. Set a PWM signal MIN (pulse width 1 ms) for 2 seconds, then off on GPIO 2 when Request "PWM_MIN" is received. Respond "ACK".	4A. Send "MIN" to microcontroller from laptop B. Probe GPIO 2 with oscilloscope. Check for 3.3V PWM with pulse width 1 ms.
5. Set a PWM signal MAX (pulse width 2 ms) for 2 seconds, then off on GPIO 2 when Request "PWM_MIN" is received. Respond "ACK".	5A. Send "MAX" to microcontroller from laptop B. Probe GPIO 2 with oscilloscope. Check for 3.3 V PWM with pulse width 2 ms.

6. Read value from GPIO 2 when Request “GET_VAL” is received. Respond with value.	6A. Send “MAX” to microcontroller from laptop B. Verify response value is proportional to weight on sensor by varying weight.

2.7.2 Control Unit Power Delivery

Since the ESP8266 has a 3.3 V VCC, 5 V or 12 V from input from the transformer must be dropped down. To do so we will use a TSR1-2433 converter.

Requirement	Verification
1. Must supply 3.3 V DC at 1 A +/- 5%	1A. Attach V_In and GND to test bench power supply, vary input voltage between 4.5 V to 12.5 V. B. Probe V_Out with oscilloscope and insure stable 3.3 V output. C. Attach microcontroller, ensure stable 3.3 V under load.

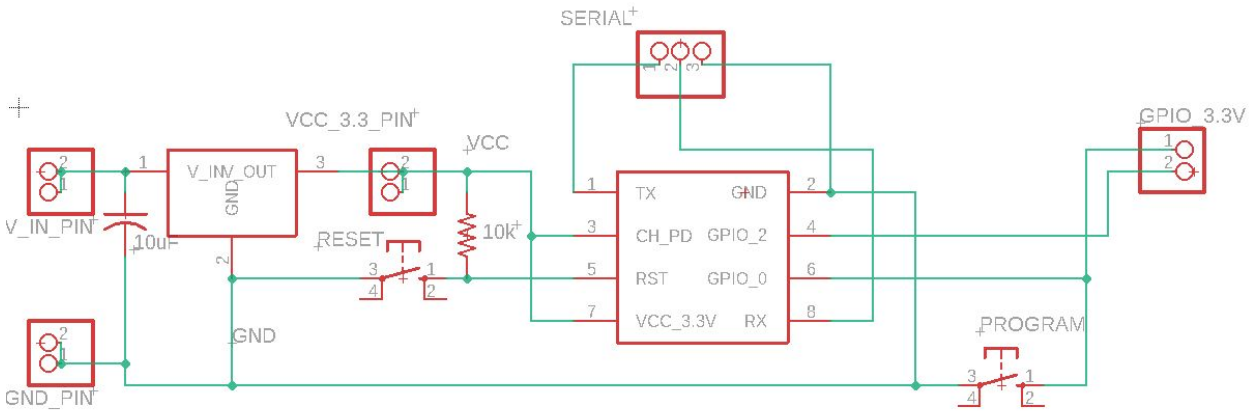

Figure 6. Control Unit Circuit Diagram

2.7.3 Web Server

This web server will be run on a personal computer, hosting a web UI that starts/stops a boba making process and allows a user to customize the liquid and solid dispensers. It will also send HTTP requests to the microcontroller, which host a micro web server to provide instructions on when to start and stop dispensing the corresponding ingredients. HTTP requests generally take

100 ms round trip, which is negligible compared to the multiple seconds we are taking to dispense liquids and solids.

Requirement	Verification
1. Must be able to send HTTP requests through port 80.	1A. Send a curl to https://google.com and does it get a response back. Curl uses port 80.

2.8 Software

The software routes information between the main server and the microcontrollers controlling the dispensers. All routing and information of transfer will be done via HTTP and thus TCP to ensure accurate retrieval of information and instructions.

2.8.1 High level architecture

A main web server, running on a computer (personal computer) will serve the user-friendly web User interface, which will receive user input in the configuring the amount of liquids and solids to dispense. It will store, mutate, and retrieve the configurations on a MongoDB database. Based on the load sensor input and the user configurations, the main server will send instructions to the microcontrollers on whether to begin or stop dispensing their respective ingredients. All of this must be done on the same Wi-Fi network, in order for the main server to contact the microcontrollers via their IP address.

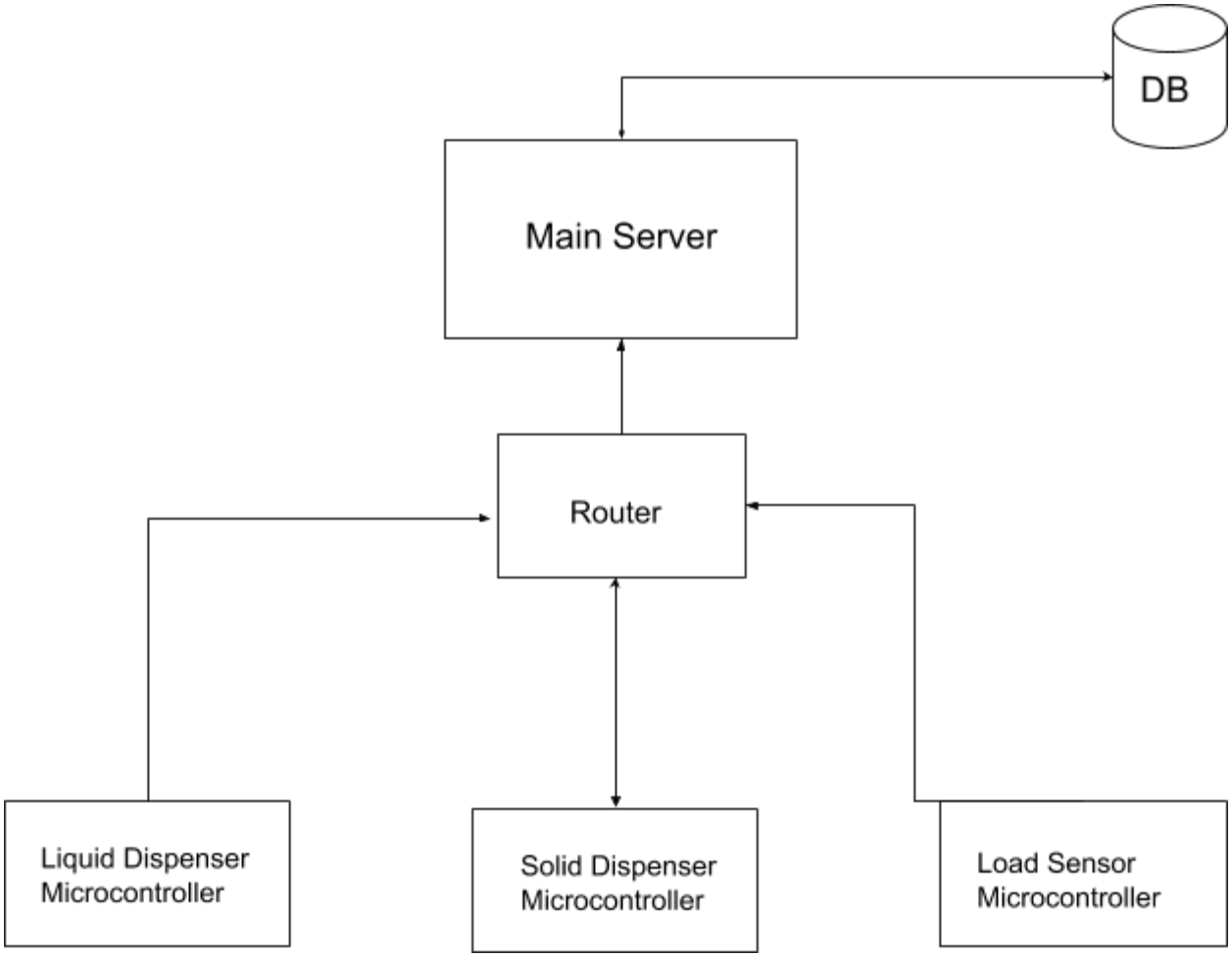

Figure 7. Software High Level Architecture

2.8.2 Routing and Dispensing Logic

Figure 8 explains the network transfer logic when a user starts the boba making process, beginning from the main server receiving that input. Essentially, once the boba making process begins, the main server makes a request to the database for the configuration on how much boba/liquid should be dispensed. Based on the configurations, it begins to give instructions to the respective microcontrollers that control the liquid dispenser and the solid dispenser, sequentially, starting with the the solid dispenser. Once a microcontroller linked to a dispenser receives a request, it unpackets it and checks whether the request is a request to start the dispensing. The main server will also start polling the microcontroller responsible for the load sensor and will receive the load sensing data every half second (this interval will most likely change as we do more experiments, but HTTP requests are around 100 ms round-trip so there's plenty of room to see how often we'd like to poll the load sensor microcontroller). Based on the weight, the main server decides when to send a request to the dispensing microcontroller to stop.

To combat the deficiencies of networking (server crashing, network packet loss, etc), there will also be a upper-bound on how long a dispenser dispenses for and the microcontroller must return an acknowledge response back to the main server whenever it receives an instruction from it, else the main server will make another response after waiting for an acknowledge response for a specified amount of time.

The main server will contact the solid dispenser first, just in case a tapioca pearl gets stuck on the gutter. Once the signal to stop the solid dispenser is sent and the acknowledge response is received, we begin liquid dispensing. If a button to stop the boba making process is pressed, all the dispensers will stop and everything will reset.

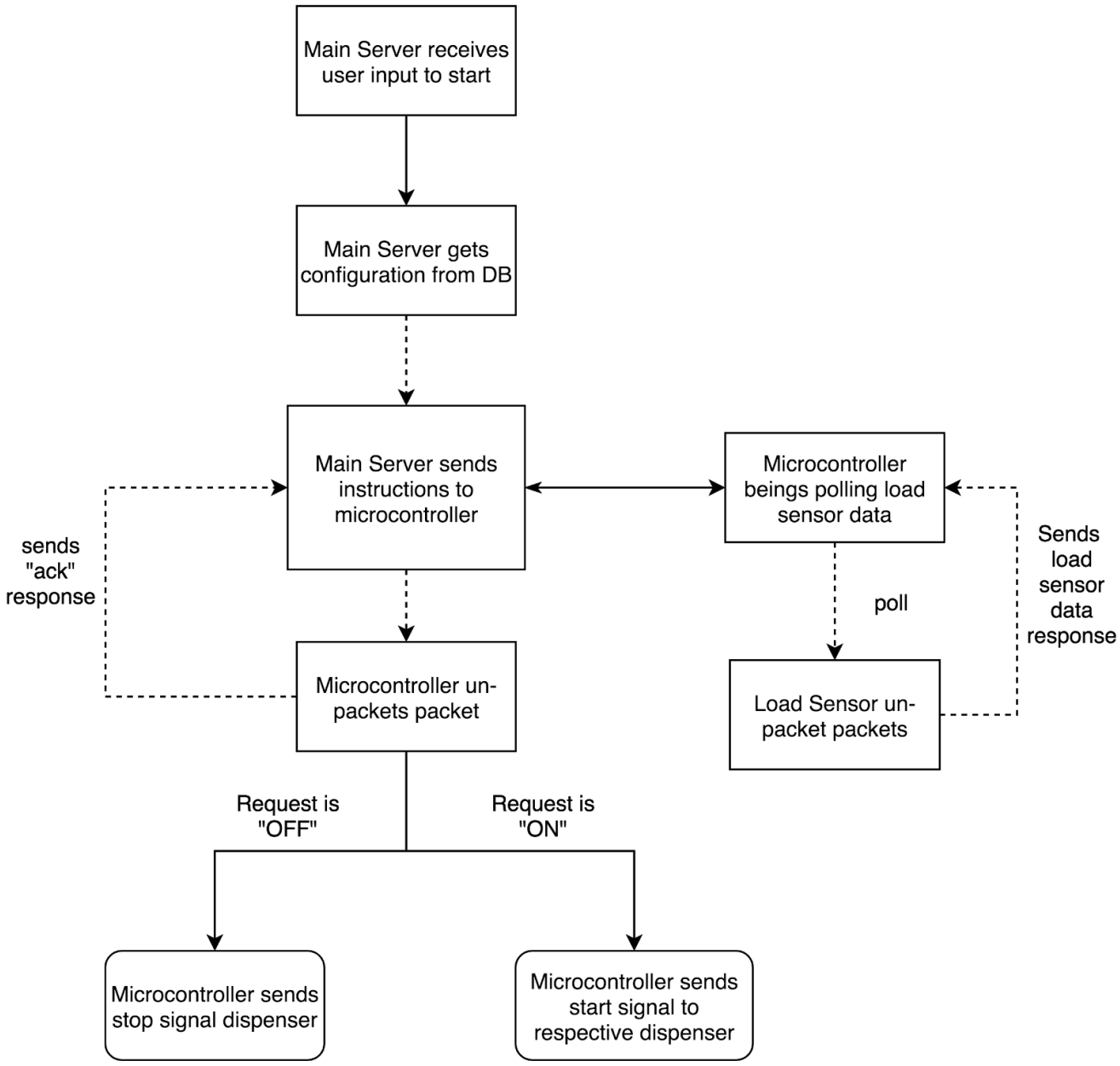

Figure 8. Routing and Dispensing Logic

2.8.2 Web User Interface

The Web UI will be implemented using the standard HTML/CSS/Javascript with a python web server. The same server implementing this UI will also be controlling the microcontrollers through HTTP requests in the boba machine. This will be integral to our project since we are building for a user, which requires an interface.

2.9 Tolerance Analysis

A critical aspect of our system is its ability to figure out how much liquid or solid it has dispensed and act accordingly, so knowing when to continue or stop dispensing through the opening of the solenoid valve for liquids and spinning of servo. This is done through finding the relations between time and the amount of liquid/solids dispensed in coordination with the weight of the ingredients in the cup as ingredients are dispensed. As described above in the requirements, we have a load sensor detecting the weight of the cup, which sends its signal through an amplifier and ADC, which is sent to a microcontroller. The main server will then send HTTP requests to the microcontroller for the current weight as an ingredient is dispensing. This data will allow the main server to make decisions on when to stop dispensing. Thus, we'd like to analyze the tolerance in the accuracy of the load sensor as well as the time intervals between the load sensor sending signals and the main server retrieving the data and whether this system can tolerate this time interval.

First off, let's look into the accuracy of the load sensor. The factors that influence the accuracy are nonlinearity, hysteresis error, repeatability, and temperature effects on zero balance and span. The combined error characteristic combines non-linearity, hysteresis error, and repeatability [11]. Thus, the accuracy of this load cell is

$$\varepsilon > \sqrt{(\varepsilon_c^2 + (\varepsilon_z * L * N * t / W) + (\varepsilon_s * t)^2)}$$

where ε is the Accuracy of the load cell (%), ε_c is the combined error in %, ε_z is the temperature effect on zero balance (%/C), ε_s is the temperature effect on span (%/C), L is the rated capacity of the load cell, N is the number of load cells to be used, W is the maximum load to be measured, t is the temperature variation range of the load cell. Thus, from the TAL221 Load Cell datasheet, we come up with the following equation.

$$\varepsilon > \sqrt{(0.05^2 + (0.1 * 1500kg * 1 * 1/1500kg) + (0.1 * 1)^2)} = 0.033\%$$

Thus, the scale will be sufficiently accurate if the resolution is 1/3000. Now, let's put this in the perspective of our system. We'll be first dispensing boba, in which a ~5 g plastic standard medium 16.9 fl. oz cup is placed on the platform with the load sensor underneath it. Each tapioca pearl is around 4 mm wide in diameter, a pearl is approximately <= 1g since there are 152 g per cup of dry tapioca pearls [10]. A cup generally has 20 to 30 pearls per drink, meaning there should be around 20 to 30 g + 5 g from the cup. The lower bound, with the error percentage would be 0.66 g while the upper bound would be 0.99, meaning that if there weren't any delays in the data transfer and instructions sent and the dispenser would immediately be closed once the cup reaches a certain weight, we would have an error of at most one pearl. For the milk tea, or

any liquids in this case, the error would be 1 mL of the corresponding liquid, which isn't much since a standard plastic medium cup is 500 mL or 16.9 fl oz.

Now, let's look into the time difference between the measurement of weight sent to the microcontroller until it reaches the main server, in which it will decide whether to stop the dispensing. An average HTTP request takes around 500 ms for the round trip with TLS (security protocol) [12]. However, we won't be using HTTPS nor will we need a Domain Name Server lookup since we would be operating in the same network, which cuts down the entire response time down to 200 ms [12] with the majority of time for the microcontroller to respond. Now the data transfer from the load sensor to microcontroller is pretty negligible in this case, since they are all data transfers, which means the majority of time would be because of the HTTP requests. Thus, the lower bound in which the main server retrieves the correct load time and sends the instruction to stop dispensing would be 200 ms while the upper bound would be 399 ms such that the microcontroller receives data from the load sensor right after it already sent back a response to the main server, requiring the main server to send another request to realize it needs to send an instruction to stop.

However, 399 ms is very negligible in this system compared to the accuracy of our load sensor, which would have an effect of plus or minus one tapioca pearl and 1 mL. As a summary, the system can handle the tolerance of 399 ms and the 0.033% error of the load sensor since 1 tapioca pearl or 1 mL is something we can tolerate.

3 Costs and Schedule

3.1 Cost Analysis

3.1.1 Labor

We assume a reasonable salary of $50 / hour. Further, we estimate that each group member will work an average of 12 hours a week for 15 weeks. Therefore, the estimated labor cost is

$$\frac{\$50}{hr} * 2.5 * (\frac{12\ hrs}{1\ week} * 15\ weeks) * (3\ partners) = \$67,500$$

3.1.2 Parts

Part	**Cost**
Liquid Dispensing Mechanism	**$42.64**
Brass Liquid Solenoid Valve - 12V (Digikey; 1528-1280-ND)	$24.95

TIP120 Transistor (Digikey; TIP120GOS-ND)	$0.69
DCJ0202 Barrel Jack (Mouser; 806-KLDX-0202-A)	$0.90
1N4001 Diode (Digikey; 641-1310-1-ND)	$0.11
Schumacher PC-6 120AC to 6A 12V DC Power Converter	$15.99
Solid Dispensing Mechanism	**$13.16**
Towerpro SG92 Servo (Digikey; 1528-1076-ND)	$5.95
BSS138 - SMD (Digikey; BSS138CT-ND)	$0.31
DCJ0202 Barrel Jack (Mouser; 806-KLDX-0202-A)	$0.90
120VAC to 5V 2A DC Power Converter	$6.00
Weight Sensing Mechanism	**$28.10**
Load Cell - 500g (Digikey; 1568-1899-ND)	$11.25
Load Cell Amplifier HX711 (Digikey; 1568-1436-ND)	$9.95
DCJ0202 Barrel Jack (Mouser; 806-KLDX-0202-A)	$0.90
120VAC to 5V 2A DC Power Converter	$6.00
Microcontroller x 3	**$59.70**
ESP8266 ver. ESP-01 (Digikey; 1568-1235-ND)	$6.95
Traco Power TSR 1-2433 (Digikey; 1951-2742-ND)	$6.14
10uF Electrolytic Capacitors (Mouser; 661-EKXF451ELL100MJ2)	$0.81
Non-Electrical Components	**$16.00**
Liquid Container (Soda bottles)	$1.00
Solids Container	$5.00
General Structure	$10.00
Total	**$159.60**

3.2.3 Grand Total

$$Grand\ total\ cost\ = labor + parts = \mathbf{\$67{,}659.60}$$

3.2 Schedule

Week	Hunter	Timothy	Jordan
2/25/2019	Design finalized	Design finalized	Design finalized
3/4/2019	Breadboard testing of load sensing mechanism. Physical design of load sensing platform sketched.	Breadboard testing of liquid dispensing mechanism. Physical design of liquids dispenser sketched.	Breadboard testing of servo mechanism. Breadboard testing of microcontroller system. Physical design of liquid dispenser + gutter sketched.
3/11/2019	PCB of load sensing mechanism done. Physical designs verified by machine shop.	PCB of liquids mechanism done. Physical designs verified by machine shop.	PCB for microcontroller + servo done. Physical designs verified by machine shop.
3/18/2019	Software configuration of load sensing mechanism done.	Software configuration of liquids mechanism done. Start work on WebUI	Software configuration of liquids mechanism done. Software configuration of multiple microcontrollers system.
3/25/2019	PCB soldering completion	PCB soldering completion	PCB soldering completion
4/1/2019	Assembly of physical module	Assembly of physical module	Assembly of physical module
4/8/2019	Assembly of physical module	Assembly of physical module	Assembly of physical module
4/15/2019	Prepare for demo	Prepare for demo	Prepare for demo
4/22/2019	Prepare for presentation	Prepare for presentation	Prepare for presentation

4/29/2019	Prepare for presentation Complete final report	Prepare for presentation Complete final report	Prepare for presentation Complete final report

5 Ethics and Safety

5.1 Ethics

While working on this project, we will abide by the IEEE Code of Ethics and the ACM Code of Ethics in their entirety. For this project, it is important to commit ourselves to #1 of the IEEE Code of Ethics: "to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or the environment" [2]. We will do so by following safety precautions described in the next section. For instance, we will use certified transformers to convert 120 V AC to 5 V DC and 12 V DC. Further, we will also be sure to commit ourselves to #7 of the IEEE Code of Ethics: "to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others" [2].

5.2 Safety

The main safety hazard is the use of a 120 V AC power from a wall outlet. To mitigate this hazard, we will use certified transformers (such as the Schumacher PC-6 DC Power Converter) to immediately transform this to 5 V DC and 12 V DC and will never use 120 V directly in any way. We will also abide by United States regulations regarding electrical devices such as those described by NISTIR 8118r1 [3]. From the end user's perspective, there are few safety concerns since the user just has to interact with the web interface and then grab a cup of boba.

Another concern is potential health and disease issues. Boba, or tapioca pearls, are made of tapioca starch and can be safety kept at room temperature for 10 hours [6]. Boba may be coated with honey, which is itself antibacterial because of its high osmolarity (concentration), high acidity, and presence of hydrogen peroxide [7]. On the other hand, room temperature milk tea can only be safety kept at room temperature for 2 hours [8]. This could be a concern when demand is low enough such that the milk tea is replenished infrequently. However, the station is designed for use only with cold liquids, so milk tea can be safely kept for longer than 2 hours.

6 References

[1] “Everything You Need to Know About Bubble Tea”, 2016. [Online]. Available: https://www.souschef.co.uk/blogs/articles/everything-you-need-to-know-about-bubble-tea [Accessed February 6, 2019]

[2] “7.8 IEEE Code of Ethics”. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed February 6, 2019]

[3] “A Guide to United States Electrical and Electronic Equipment Compliance Requirements”, 2017. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2017/NIST.IR.8118r1.pdf [Accessed February 6, 2019]

[4] “TAL221: MINIATURE LOAD CELL” [Online]. Available: https://cdn.sparkfun.com/assets/9/9/a/f/3/TAL221.pdf [Accessed February 7, 2019]

[5] “Milk Tea & Iced Tea” [Online]. Available: http://www.sunnysboba.com/milk-tea--iced-tea.html [Accessed February 7, 2019]

[6] “Bubble Tea Supply” [Online]. Available: https://www.bubbleteasupply.com/faq/ [Accessed February 7, 2019]

[7] “Honey: its medicinal property and antibacterial activity” [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3609166/ [Accessed February 7, 2019]

[8] “Danger Zone” [Online]. Available: https://www.fsis.usda.gov/wps/portal/fsis/topics/food-safety-education/get-answers/food-safety-fact-sheets/safe-food-handling/danger-zone-40-f-140-f/CT_Index [Accessed February 7, 2019]

[9] “Tapioca Pearls Products from AKFP” [Online]. Available: http://akfponline.com/specialty-products/tapioca-products/tapioca-pearls/ [Accessed February 19, 2019]

[10] “Tapioca, pearl, dry - 1 cup” [Online]. Available: https://www.nutritionix.com/i/usda/tapioca-pearl-dry-1-cup/513fceb775b8dbbc21002ddc [Accessed February 19, 2019]

[11] “How to Use Load Cells” [Online]. Available: https://www.aandd.jp/products/weighing/loadcell/introduction/pdf/6-1.pdf [Accessed February 19, 2019]

[12] “Measuring HTTP response times with cURL” [Online]. Available: https://ops.tips/gists/measuring-http-response-times-curl/ [Accessed February 20, 2019]

[13] “A Beginner's Guide to the ESP8266” [Online]. Available: https://tttapa.github.io/ESP8266/Chap01%20-%20ESP8266.html [Accessed February 19, 2019]