Bubble Tea Machine

ECE 445 Final Report

Team 30: Emily Hall, Saisita Maddirala, Tracy Tang

Professor: Pengfei Song

TA: Hojoon Ryu

Spring 2022

Abstract

This report introduces an automated Bubble Tea Machine. This device allows users to order a drink from a button pad, and then it automatically dispenses the drink according to the user's choices. The device uses food-safe components that make the final product (a bubble tea drink) safe to consume. This report contains the development of this machine and discusses the produced working model. Future work that could be done to better the product is discussed. Additionally, challenges met in creating the prototype are listed alongside modifications that could remedy these problems.

Table of Contents

Introduction

1.1 Statement of Purpose

This project aims to produce a durable, food-safe device for the storage and dispensing of bubble tea. Bubble tea is typically a mixture of black tea, milk, flavoring, and tapioca pearls (called boba or bubbles). Bubble tea shops are immensely popular on the UIUC campus. While incredibly tasty to drink, they are not cheap. An average person can expect to spend about $6 for their drink. This project aimed to provide a more affordable option for students around campus.

1.2 Solution Overview

The created prototype allows the user to select the type of drink desired. A blinking LED guides the user through the selection process and there are options for the size and the presence of boba in the drink. In further iterations of the design, these options could be extended to flavoring for tea, flavoring for boba pearls, and types of milk. Once selections are made, boba is dispensed by a valve controlled by a servo. The liquid is dispensed by a peristaltic pump.

1.3 Visual Aid

Figures 1 and 2 display the initial concept and the completed machine, respectively. Figure 1 shows the milk and tea in separate reservoirs as well as the boba being in a reservoir with a valve on the side; whereas in the finished product, the boba is held in a reservoir with a valve at the base of a funnel that acts as the reservoir.

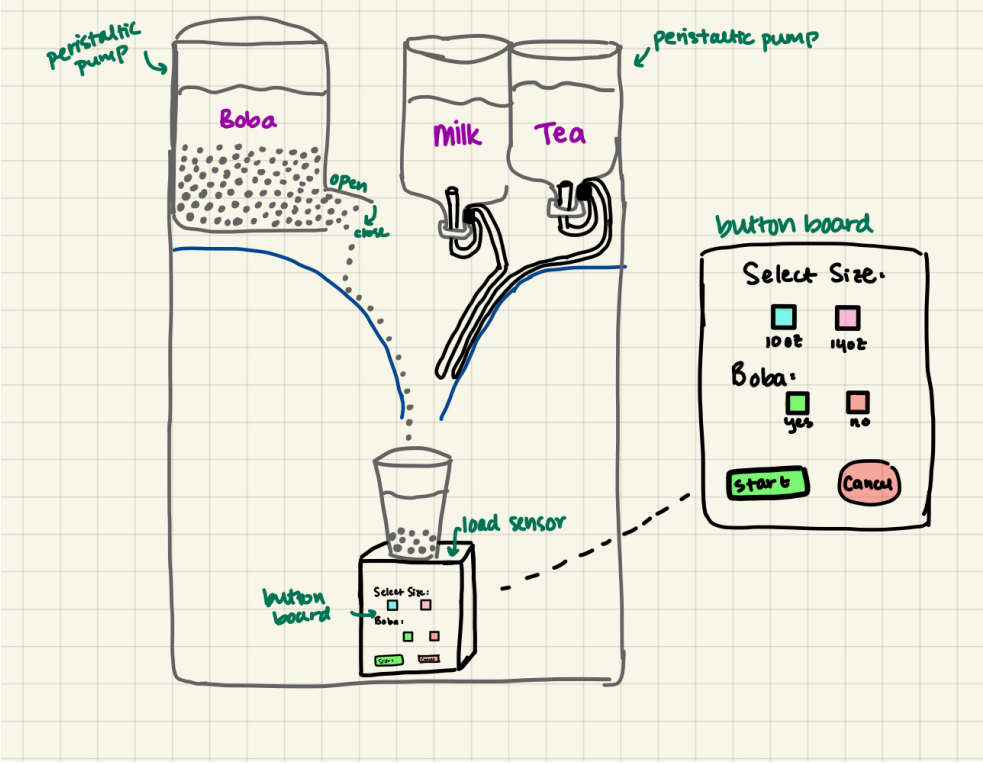

Figure 1: Preliminarily Sketch of Device.

Figure 2: Completed Prototype.

1.4 Objectives & High-Level Requirements

- The device must provide the user with many combinations of size and ingredients. We will have two size options (10 oz, and 14 oz), and an option for no boba. Each combination will be allowed, giving the user 4 possible drink options.
- The device must be able to dispense pre-calculated amounts of liquid and boba into the cup.

	Boba	**Milk & Tea**
10 oz	35-42g	160-175g
14 oz	35-42g	270-300g

Table 1. Drink Size Measurements.

- The device must start/cancel under the appropriate conditions. It should start only if the “start” button is pressed. The machine should stop the order if the “cancel” button is pressed.

Design

2.1 Block Diagram

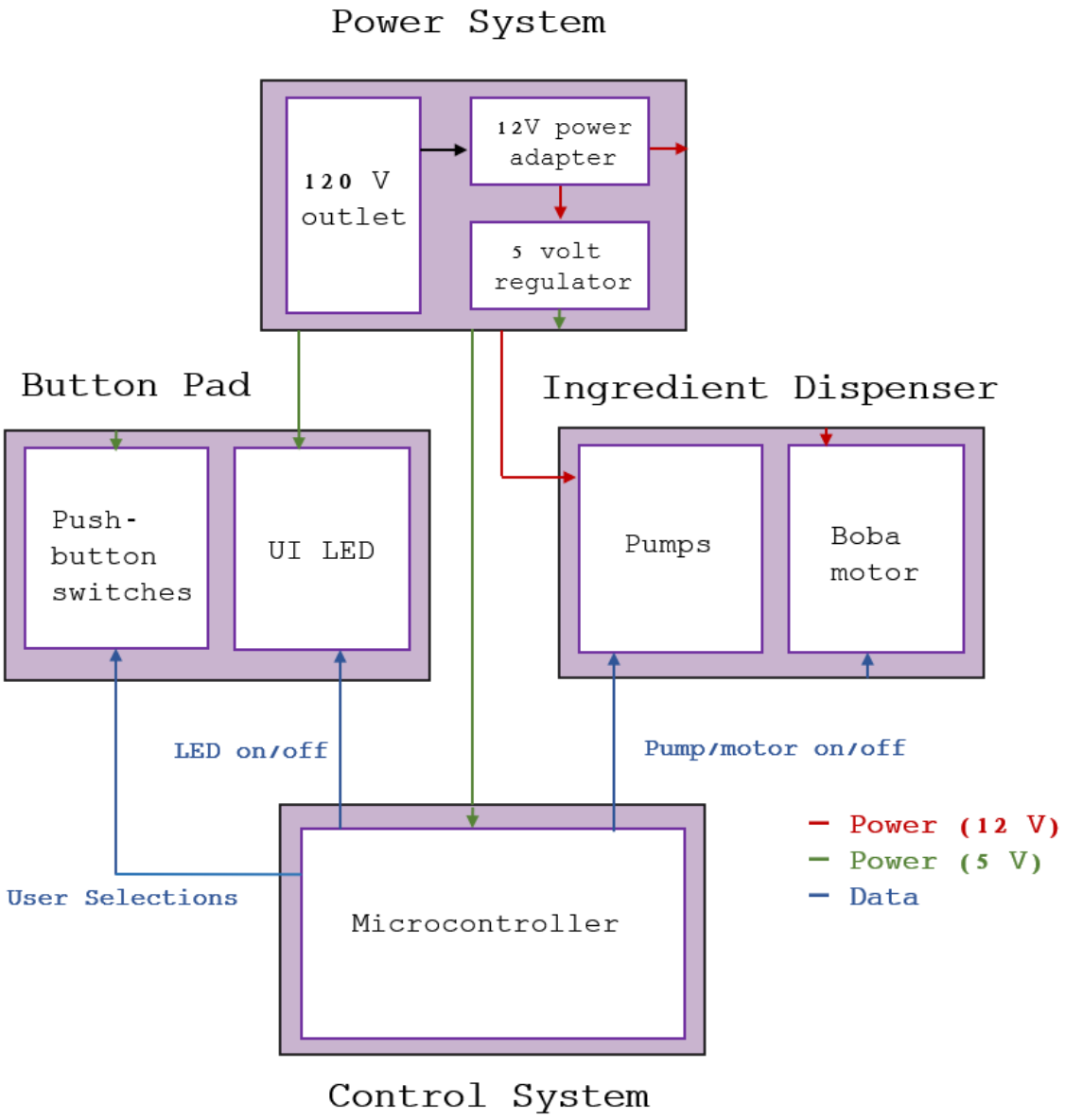

Figure 3: Block Diagram

2.2 Subsystem Descriptions

2.2.1 Power System

The prototype requires a 5V power source and a 12V power source. The device uses a power adapter to step down from 120V (from the

grid) to 12V. The system utilizes a 5V linear voltage regulator to step down from 12V to 5V.

2.2.2 I/O System

The input and output system consists of an LED and 6 buttons. These 6 buttons consist of two size options: 10oz and 14oz, Yes/No options for Boba, Start Drink, and Restart Order. The buttons and the LED require 5V. The signals from the buttons are passed to the microcontroller. The LED flashes at different rates as the user moves through the button options. To signal that the machine is ready to take a new order, the button flashes slowly. Once the user has made a size selection it flashes more quickly. After the user makes a boba selection, it flashes very quickly to signify that the user should press “start”. If at any point the user selects “restart”, the LED flashes slowly indicating that a size selection should be made.

2.2.3 Ingredient Dispensing System

The liquid ingredients are dispensed with a peristaltic pump. The liquid is held in a reservoir made from a plastic bottle that is turned upside down. The cap of the bottle has been affixed with a watertight seal around a silicon tube and the liquid is pumped through the tube. The tubes are food-safe.
The tapioca pearls were more difficult to dispense. To maintain a pleasant texture they need to be covered in syrup. While insuring their texture from becoming mushy, the syrup ensures that they stick to one another. The solution to dispensing the boba was to hold them in enough syrup that we were essentially dispensing a liquid. A valve at the bottom of the funnel opened for 1 second. This allowed about 1-2oz of boba and syrup to escape into the cup.

2.2.4 Control System

An ATmega328p microcontroller is the heart of the control system. Figure 4 depicts the state machine that the control system passes through as it receives input from the user. Table 2 describes the states, their inputs, and their outputs. Once powered on, the machine begins in state 1. This is the idle state. It will be in this state anytime it is not taking input from a user or making a drink. Once a user begins an order by selecting a drink size, the machine moves to state 2, waiting on a boba choice. Once a boba option is selected it moves into state 3. In both states 2 and 3, if “restart” is pressed, the system will return to the idle state. If the machine is in state 3 and “start” is pressed, the system will begin to prepare a drink. Once completed the machine will return to the idle state.

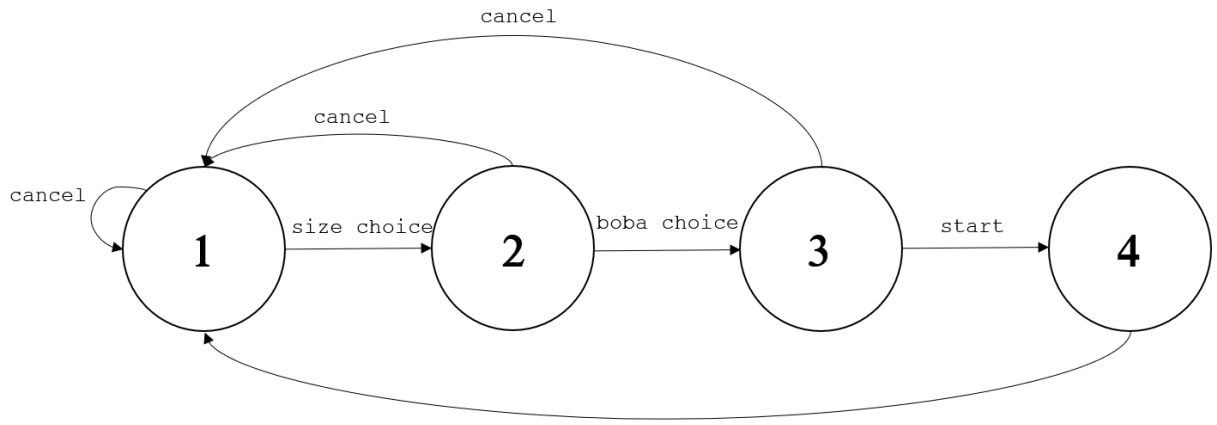

Figure 4: State Machine

State	Description	Inputs accepted	Outputs
1	- Machine is powered on - Ready for size	- Size choice - Cancel	- Size LEDs - Cancel LED

	choice		
2	- Ready for boba choice - Ready for cancel button press	- Boba choice - Cancel	- Boba LEDs - Cancel LED
3	- Ready for start button press - Ready for cancel button press	- Start button press - Cancel	- Start LED - Cancel LED
4	- Dispense boba, milk, and tea	_____	- Boba motor - Milk pump - Tea pump

Table 2: Control System State Descriptions

2.3 Design Adjustments

2.3.1 Load Cell

The initial design of the Bubble Tea Machine included a load cell that was used to measure the amount of boba and liquid dispensed into the cup. This element of the design was not implemented into the current prototype as there was a communication issue between the load cell amplifier and the ATmega328p. The amplifier (Hx711) outputs 24 bits of 2's complement numbers. Data about the weight placed on the load cell is outputted from the amplifier serially. The microcontroller must be able to read this data from the Hx711 under the right conditions. The data is outputted when the DOUT (Figure 5) goes low (meaning "data is ready for retrieval" [2]). There should also be 25-27 clock pulses sent to the PD_SCK pin (Figure 5). This is so that the 24 bits can shift out. We manually tried to get these conditions met. We placed condition statements to make sure DOUT goes low and manually sent 27 clock pulses (Figure 28 of appendix C)[9]. These condition statements were connected to an LED that should turn on once all conditions

are met, indicating that the bits of data were all sent to the microcontroller. However, the LED never went on. We think there needs to be more initialization that may have needed to be done to connect the amplifier to the microcontroller. Perhaps the Hx711 needed to be powered on and off first, maybe a different library needed to be installed, etc.

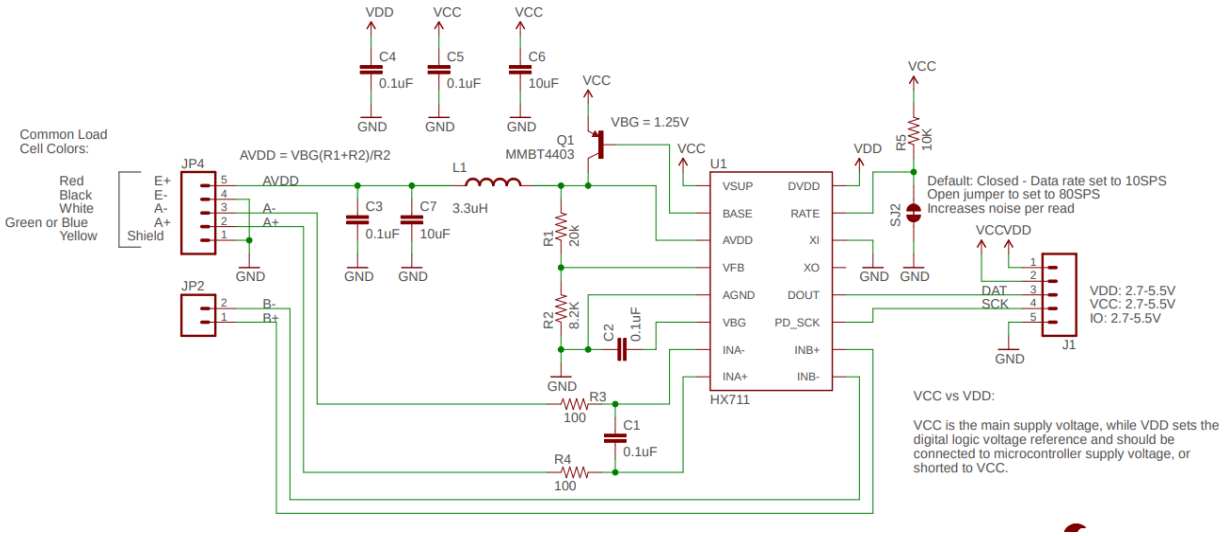

Figure 5. Hx711 Schematic

2.3.2 Single Pump

Furthermore, the preliminary designs had separate liquid dispensing mechanisms for the milk, tea, and flavors of syrup. In the prototype, only a single liquid dispenser was used. This is because a single pump requires a whole h-bridge chip for current to be dissipated evenly across the chip. This would significantly crowd the printed circuit board, and also be redundant. A single liquid dispenser demonstrates the feasibility of dispensing additional liquids.

2.3.3 One LED No analog inputs

Finally, the design included an LED for each button. These were supposed to help guide the user through using the button pad. In

order to have enough pins on the microcontroller to have these 6 outputs meant we needed to use the analog inputs to take input from the buttons. This greatly complicated the I/O system. Troubleshooting the analog inputs and the buttons became a bottleneck in designing the system so we chose to put it aside. The single LED on our current prototype is sufficient to guide a user through the button pad.

Design Verification

3.1 Power System

The power system needs to provide power within a threshold so that all the parts run (but also do not burn them out). We initially built our power system on a breadboard and only once we were sure that the voltage was within our threshold (refer to Table 1 in Appendix A) did we test the power system on our PCB. We double-checked that the incoming voltage (from the power adapter) is 12v (Figure 7) and that the regulator steps down the voltage to 5v (Figure 8).

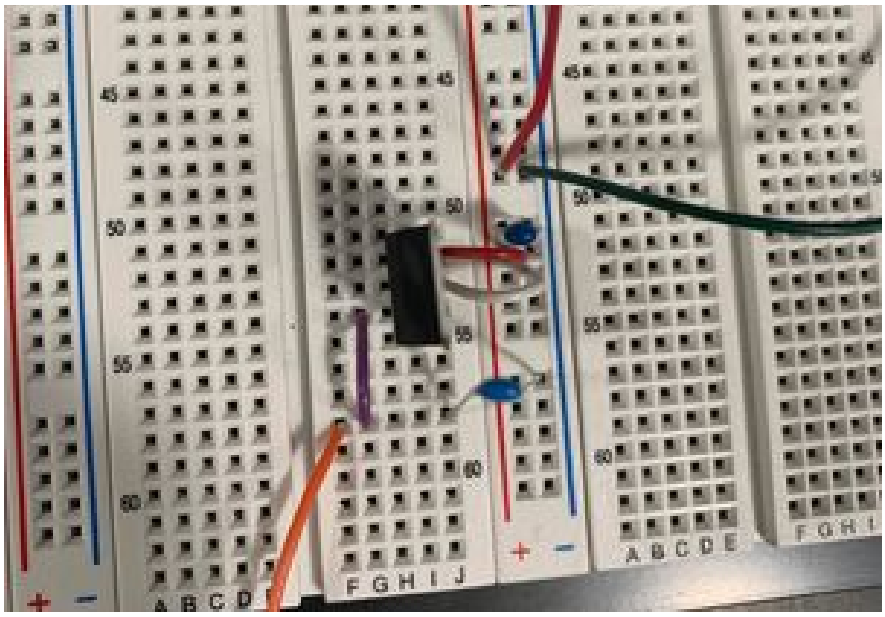

Figure 6. Power System on Breadboard

Figure 7. Incoming Voltage

Figure 8. Stepped Down Voltage

3.2 I/O System

We arguably had the most immediate trouble with the I/O system. Getting the buttons to react when they were pressed was really difficult because a lot of the buttons were set as analog pins on our microcontroller. However, after switching our buttons to digital pins, they reacted better. Again, we tested our buttons on the breadboard first (Figure 9). Once the LEDs reacted to the button presses consistently (with negligible delay), we connected the button pad to the PCB. Figures 10 and 11 show the button signal when they were connected to the analog and digital pins on the microcontroller respectively. Once all the buttons were moved to the digital pins, the button pad worked effectively and all of our requirements were met (refer to table 2 in appendix A).

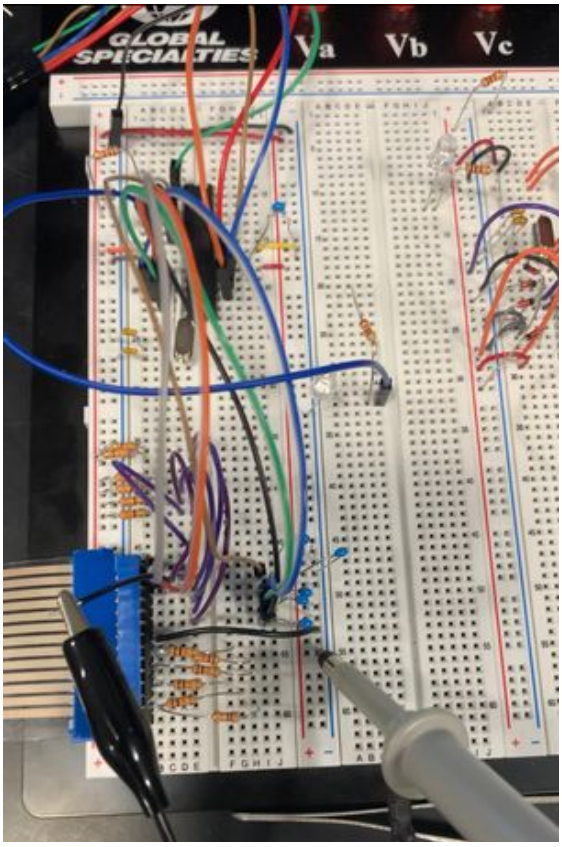

Figure 9. Button Pad and LED on Breadboard

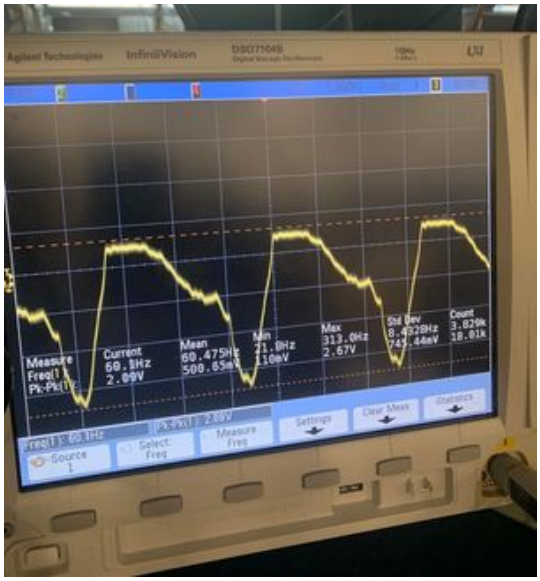

Figure 10. Button Analog Input

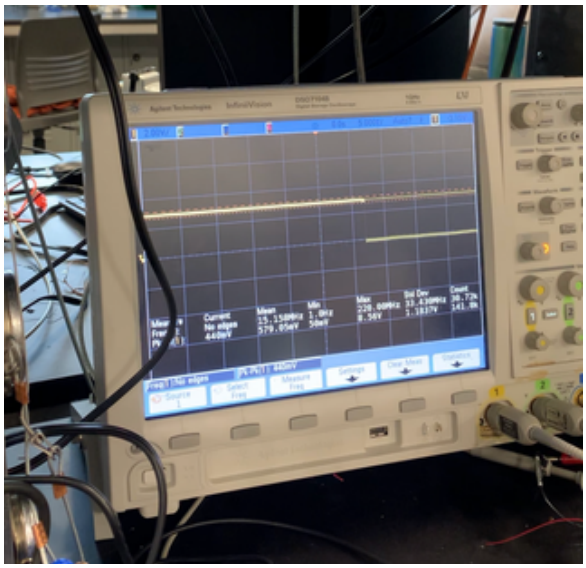

Figure 11. Button Digital Input

3.3 Ingredient Dispensing System

Our group also faced some challenges with the ingredient dispensers. Starting with the liquid pumps, the H-bridge produced a little conflict. As usual, we tested the H-bridge on a breadboard. As mentioned before, the H-bridge could not evenly distribute the current across the chip, [8] so we decided to eliminate the extra, redundant pump. However, our requirements regarding the H-bridge were still met (refer to table 3 in appendix A). The pump for the combined milk and tea dispenser still ran and there was an accurate PWM signal sent to the H-bridge (Figure 12).

The servo motor that releases the boba was initially supposed to be powered by a servo motor trigger. This trigger has 3 potentiometers that set the positions of where the servo stops. However, we accidentally burnt out the trigger so we ended up using a servo library on the Arduino IDE. The library allowed us to set the time and positions the servo should move to (Figure 23, 27 of appendix C). Although we did not use a servo trigger, we were still able to power and control the servo boba motor, checking off another requirement.

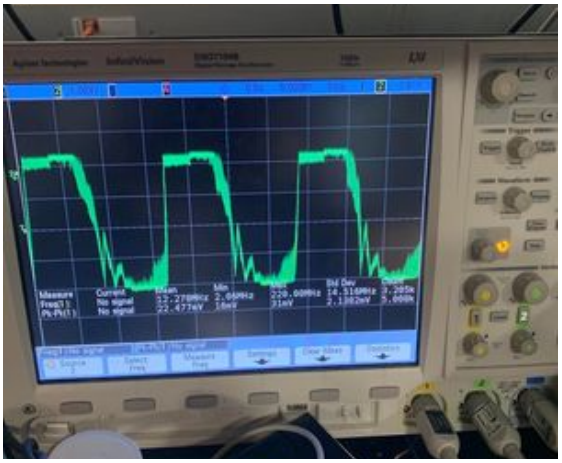

Figure 12. H-Bridge PWM Signal

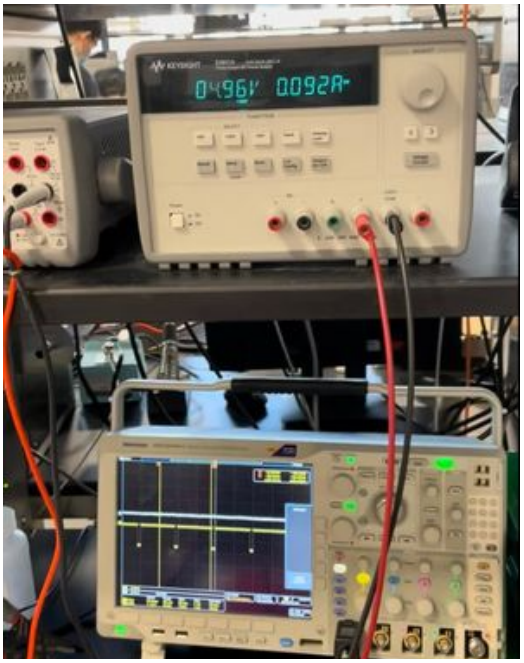

Figure 13. Servo PWM Signal

Figure 14. Servo controlled with button pad

3.4 Control System

The microcontroller that we used (ATMega328P) proved to be extremely helpful in integrating each of the parts together. The code that is programmed onto the microcontroller (Figures 25, 26 of appendix C) assures that requirements 1 and 2 are met (Table 4 of appendix A). Looking at the state diagram that the code goes through (Figure 4), it is set so that the machine will not start making the drink until the “start” button is pressed. In addition, the “cancel” button can be pressed at any other state except when the “start” button is pressed. This is clear with the LED output as well.

With the load cell, however, we faced a lot of problems. The 1kg load cell that we used recognizes changes in weight based on the differences in resistance (output in volts). However, these voltages have very negligible differences, requiring us to use an amplifier. The amplifier had a tough time communicating with the microcontroller (as mentioned before) so we, unfortunately, could not configure the

load cell within the time frame to complete this project. This is the only requirement/feature that we could not implement. Although we could not utilize the load cell to control when each dispenser would release their respective ingredients, we were still able to implement this functionality through time delays in our code. Figure 15 shows the number of ounces produced with the time that the pumps are on (in seconds). Using this data, we ended up leaving the pumps on for 143 seconds for a 10 oz drink and 258 seconds for a 14 oz drink. The boba motor is a bit less consistent, but we decided to leave the servo open for 1 second. This releases about 1-2 ounces of boba with the right amount of syrup and agitation. With these delays in time, the weight of the final drink is pretty close to the desired ounces (Figure 16).

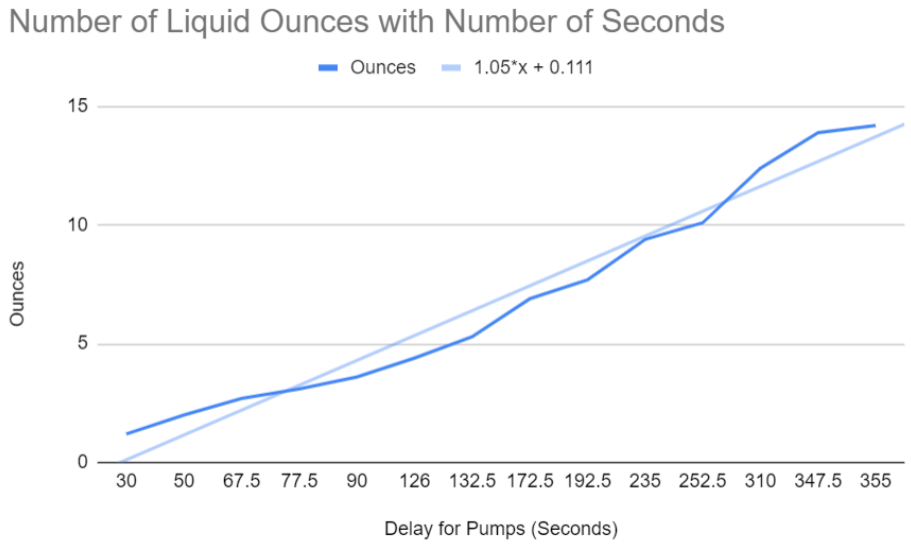

Figure 15. Plot Showing Number of Liquid Ounces with Number of Seconds

Figure 16. Weight of 10 oz Drink

Cost & Schedule

4.1 Cost Analysis

4.1.1 Labor

According to the Illini Success Annual Report 2019-2020, a Computer Engineer would make an average of $99,145. This leaves us with an hourly wage of $47.67. Assuming we each work around 14 hours a week, with 8 weeks remaining (112 total hours) and using the formula ($/hour) * 2.5 * hours to complete, the total labor cost of a computer engineer would be $13,347.60. Each of us would average around $5,339.04 for labor costs.

4.1.2 Parts

Module	Product ID	Price per Unit	Quantity	Price
12V Power Adapter	1470-3113-ND	$11.66	1	$11.66
5v Voltage Regulator	MC7805CTG-ND	$0.65	3	$1.95

ATMega328 Microcontroller Bootloader Uno	X000048	$5.87	2	$11.74
Peristaltic Pumps	1150	$24.95	2	$49.90
Silicone Tubing	3659	$3.50	1	$3.50
1kg Load Cell Sensor	1528-4540-ND	$3.95	1	$3.95
500g Load Cell Sensor	1568-1899-ND	$11.25	1	$11.25
HX711 Amplifier	1568-1436-ND	$9.95	1	$9.95
Load Sensor Combinator	474-BOB-13878 (Mouser)	$1.95	1	$1.95
H-Bridge Motor	Bridgold-31	$8.99	1	$8.99
Servo Motor	900-00005-ND	$16.72	1	$16.72
WIG-13118 Servo Motor Trigger	1568-1363-ND	$17.95	1	$17.95
				Total: $149.51

4.2 Schedule

Week	**Tasks**	**Emily**	**Saisita**	**Tracy**
2/21	Design Doc Check, Finalize + Order Parts	Complete Design Document,	Complete Design Document,	Complete Design Document,

		Draft+Finalize PCB layout	Draft+Finalize PCB layout	Draft+Finalize PCB layout
2/28	Design Review, PCB Board Review	Complete Design Review, Finalize parts order	Complete Design Review, Finalize PCB layout and get board approved (first-round order)	Complete Design Review, Finalize PCB layout and get board approved (first-round order)
3/7	Order PCB and request machine shop work	Buy components for the build of the design (mechanical components), Test load sensor output (reference voltage) to determine resistor values, bring project to the machine shop (if we've received parts)	Buy components for the build of the design (mechanical components), Test load sensor output (reference voltage) to determine resistor values, bring project to the machine shop (if we've received parts)	Buy components for the build of the design (mechanical components), Test load sensor output (reference voltage) to determine resistor values, bring project to the machine shop (if we've received parts)
3/14	Spring Break	- - - - - - -	- - - - - - -	- - - - - - -
3/21	Finish Soldering PCB and write out microcontroller	Write out microcontroller code, work on individual	Solder PCB, work on individual progress	Write out microcontroller code, Help Tracy Solder

	code	progress report	report	PCB, work on individual progress report
3/28	Individual Progress Reports Due	Test input and output from load cell	Test output to the servo motor. Make sure servo motor and motor trigger powers the servo appropriately.	Test output to pumps. Test pumps with load cell sensor and check that the motor driver powers the motor appropriately.
4/4	Test all components	Test all inputs and outputs together	Test all inputs and outputs together	Test all inputs and outputs together
4/11	Complete all tests	Finish testing, create mock demo	Finish testing, create mock demo	Finish testing, create mock demo
4/18	Mock Demos	Mock Demo, begin final demonstration (last-minute touch-ups)	Mock Demo, begin final demonstration (last-minute touch-ups)	Mock Demo, begin final demonstration (last-minute touch-ups)
4/25	Demonstrations	Final demonstration, start working	Final demonstration, start working	Final demonstration, start working

		on final paper	on final paper	on final paper
5/2	Final Paper Due	Finish Final Paper, add last-minute touch-ups	Finish Final Paper, add last-minute touch-ups	Finish Final Paper, add last-minute touch-ups

Conclusion

5.1 Accomplishments

The prototype created throughout this project was ultimately successful: it automated the Bubble Tea making process. The pumps accurately and reliably dispensed liquid according to our objectives. With marginal agitation, the boba could also be reliably dispensed. The control system stably moved through the states. The button presses were correctly ignored or registered by the microcontroller according to the system's current state.

5.2 Uncertainties

The primary uncertainty with the design is dispensing the boba. If the texture of the syrup is too thin or thick, or a single pearl gets jammed, the amount of boba that is dispensed can vary wildly.

5.3 Ethical Considerations

The primary ethical consideration in developing this prototype was food safety. The food remains in sealed and food-safe containers while inside the machine. The silicon tubing is food-safe. Additionally, after every use, we advise that warm, soapy water is run through the machine to clear any food particles that might allow mold to grow within the device.

5.4 Future Work

This device has the potential to be catered to many different environments. For example, adding a payment system would allow this device to act as a Bubble Tea vending machine. However, several elements would require refinement before it would be useful. Dispensing the boba remains the most significant issue. In our testing, we discovered that agitating the boba gently while the valve is open reliably moves the boba through the valve. In the future, a small motor could be placed above the boba reservoir to stir the boba whenever the valve is open. Furthermore, implementing the load cell would greatly aid in the boba problem. The feedback from the scale would allow the system to wait until enough boba has been dispensed. With the current method of timing, the amount of boba dispensed can vary.

Additionally, adding options for other types of milk, and offering a wide variety of syrups would enhance the consumer appeal of the machine. This would require adding a system to maintain a cool temperature for the milk.

References

[1] *Quadruple half-H drivers (rev. C) - adafruit industries*. (n.d.). Retrieved March 17, 2022, from https://cdn-shop.adafruit.com/datasheets/l293d.pdf

[2] *Description features AVDD*. (n.d.). Retrieved March 18, 2022, from https://cdn.sparkfun.com/assets/b/f/5/a/e/hx711F_EN.pdf

[3] *VCC: 2.7-5.5V VBG = 1.25V. (n.d.). Retrieved April 15, 2022, from https://cdn.sparkfun.com/assets/f/5/5/b/c/SparkFun_HX711_Load_Cell.pdf*

[4] IEEE Code of Ethics. IEEE Code of Ethics. (n.d.). Retrieved February 02, 2022, from https://www.ieee.org/about/corporate/governance/p7-8.html

[5] *Direct PIN IO (Arduino) | Marc's Blog. (n.d.). Retrieved April 2, 2022, from http://blog.marcsymonds.me/arduino-code/direct-pin-io-arduino/*

[6] *Servo Motor Basics with Arduino | Arduino Documentation. (n.d.). Docs.arduino.cc. Retrieved April 23, 2022, from https://docs.arduino.cc/learn/electronics/servo-motors*

[7] *Blink. (n.d.). Www.arduino.cc. Retrieved April 13, 2022, from https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink*

[8] *Can L293D really be put in parallel? (2015, July 28). Arduino Forum. Retrieved April 17, 2022, from https://forum.arduino.cc/t/can-l293d-really-be-put-in-parallel/326439*

[9] *What kind of protocol does the HX711 use? (2018, November 24). Arduino Forum. Retrieved April 20, 2022, from*

https://forum.arduino.cc/t/what-kind-of-protocol-does-the-hx711-use/402474/5

[10] *How to read and display 24 bit two's compliment in decimal form. (2013, October 31). Arduino Forum. Retrieved April 22, 2022, from https://forum.arduino.cc/t/how-to-read-and-display-24-bit-twos-compliment-in-decimal-form/191457/3*

[11] *USB - cdn.sparkfun.com. (n.d.). Retrieved April 18, 2022, from https://cdn.sparkfun.com/assets/0/4/5/1/9/SparkFun_openScale_schematic.pdf*

[12] *Fluctuation value at loadcell 10000kg + hx711. (2021, January 4). Arduino Forum. Retrieved April 23, 2022, from https://forum.arduino.cc/t/fluctuation-value-at-loadcell-10000kg-hx711/689238/6*

Appendix A: Requirements and Verification Tables

Requirement	Verification
1. Bubble Tea Machine must plug into the wall and receive 12VDC ± 5% from the power supply.	1A. Cut off the end of the power supply and use an oscilloscope to check that the output voltage of the power supply stays within 5% of 12V.
2. Voltage regulator must send 5v ± (1.5-4)% to the microcontroller, Servo Motor, and load cell sensor/Hx711 amplifier.	2A. Voltage regulator must send 5v ± (1.5-4)% to the microcontroller, Servo Motor, and load cell sensor/Hx711 amplifier.
3. Voltage regulator must supply 150-300 mA to Peristaltic Pumps and microcontroller.	3A. Vin in Figure 17. 3B. Change R1 from Figure 17 so that the output current is the appropriate amount for each component. 3C. Probe Vo from figure 17 and measure current with the adjusted resistors to ensure the current supplied is 150-300 mA.

Table 1. Power System RV Table

Requirement	Verification

1. The push-button switches must send accurate signals to the appropriate ingredient containers.	1A. Connect the start signal from figure 18 (J19) to an oscilloscope and the start signal (pin 28) from figure 19 to a different channel on the oscilloscope. Check that these signals match. 1B. Check that the start_led from figure 19 turns on. 1C. Repeat 1A. For the 10oz button (J14), and the 14oz button (J16) from figure 18 to their respective signals/LEDs from figure 19.
2. The push-button switches receive 40-60 mA of current.	2A. Connect the 5V for the buttons to a 100Ω potentiometer. 2B. Change the potentiometer until it reaches 40-60 mA. 2C. Measure the current with a multimeter.

Table 2. Button Pad RV Table

Requirement	**Verification**
1. Servo motor must run when powered by the servo trigger.	1A. Connect servo motor output from servo trigger (figure 20) to a multimeter. 1B. Check that 5v±0.25 is being supplied.

2. The pumps (DC motor) must run when powered by the H-bridge.	2A. Connect Vcc2 (pin 8) from figure 21 to a multimeter. 2B. Check that 12v±0.25 is being supplied.
3. The Servo motor trigger should leave the door of the servo motor open for 7-10 seconds for the boba to be dropped.	3A. Connect the out pin of the servo motor trigger pin to the oscilloscope. 3B. Adjust the time potentiometer (potentiometer C) until open for 7-10 seconds. 3C. Ensure that the servo motor does not shut before the completion of seconds.
4. PWM signal is supplied to H-bridge accurately (circuit works).	4A. Connect PWM signal from figure 21 to oscilloscope. 4B. Check that the wave we see on an oscilloscope is an accurate PWM wave. 4C. Connect PWM signal (en1 in h-bridge) from figure 21. 4D. Check that the wave we see on an oscilloscope is an accurate PWM wave.
5. Servo motor receives 4-6 mA.	5A. Connect the 5V for the Servo Motor to a 100Ω potentiometer. 5B. Change the potentiometer until it reaches 4-6 mA. 5C. Measure the current with a multimeter.

6. Peristaltic Pumps receive 200-300 mA.	6A. Connect the 5V for the Peristaltic Pumps to a 100Ω potentiometer. 6B. Change the potentiometer until it reaches 200-300 mA. 6C. Measure the current with a multimeter.

Table 3. Ingredient Dispenser RV Table

Requirement	**Verification**
1. Bubble tea machine should only start making the drink when the “start” is pressed.	1A. Connect boba_motor (pin3) signal (figure 19) to the oscilloscope. 1B. Ensure the signal is high when the start signal (figure 19) is high.
2. Bubble Tea Machine should ignore the “cancel” button once “start” has been pressed.	2A. Connect cancel_button (figure 18) to an oscilloscope. 2B. Connect the start_button (figure 18) to a different channel on the oscilloscope. 2C. Connect the cancel_led signal (figure 19) to a different channel on the oscilloscope. 2D. Ensure that the cancel_button signal is only high when the start signal is low and the cancel_led signal is high. Alternatively, we can make sure that the

	cancel_led is not on when the start_button is high and the cancel_led is on when the start_button is low.
3. Load Cell Sensor stays within appropriate voltages for each drink size. 10oz: (3mV±.75), 14oz: (3.9mV±.75)	3A. Hook up the green and white wires of the load cell sensor to a multimeter. 3B. Verify that the voltage difference is the appropriate amount for each ingredient and drink size. 10oz: (3mV±.75), 14oz: (3.9mV±.75)
4. The microcontroller receives 100-150mA.	4A. Connect the 5V for the microcontroller to a 100Ω potentiometer. 4B. Change the potentiometer until it reaches 100-150 mA. 4C. Measure the current with a multimeter.

Table 4. Control System RV Table

Appendix B: Circuit Schematics

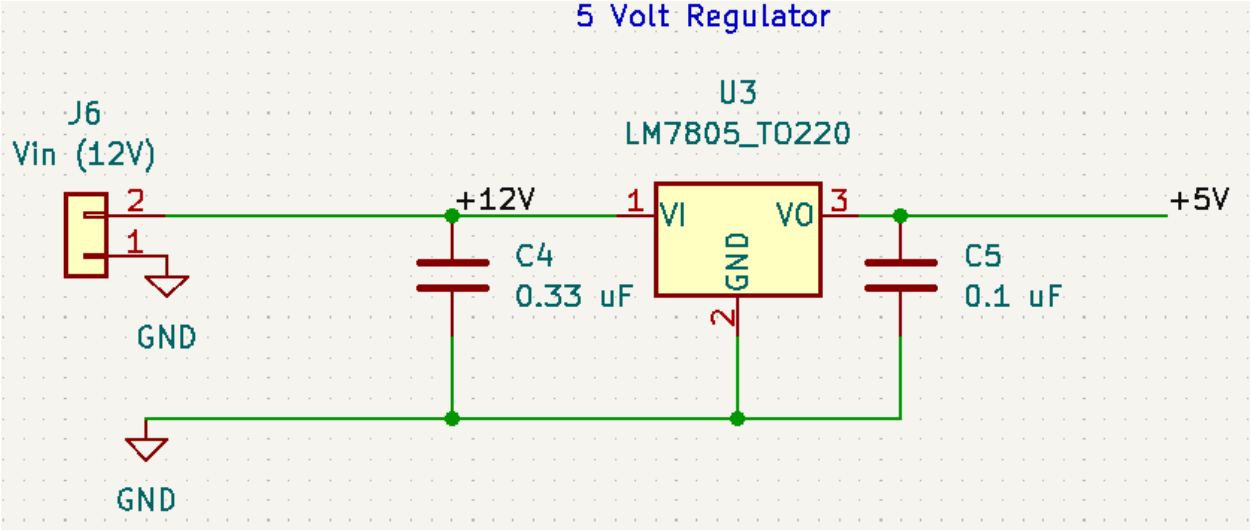

Figure 17. Voltage Regulator Schematic

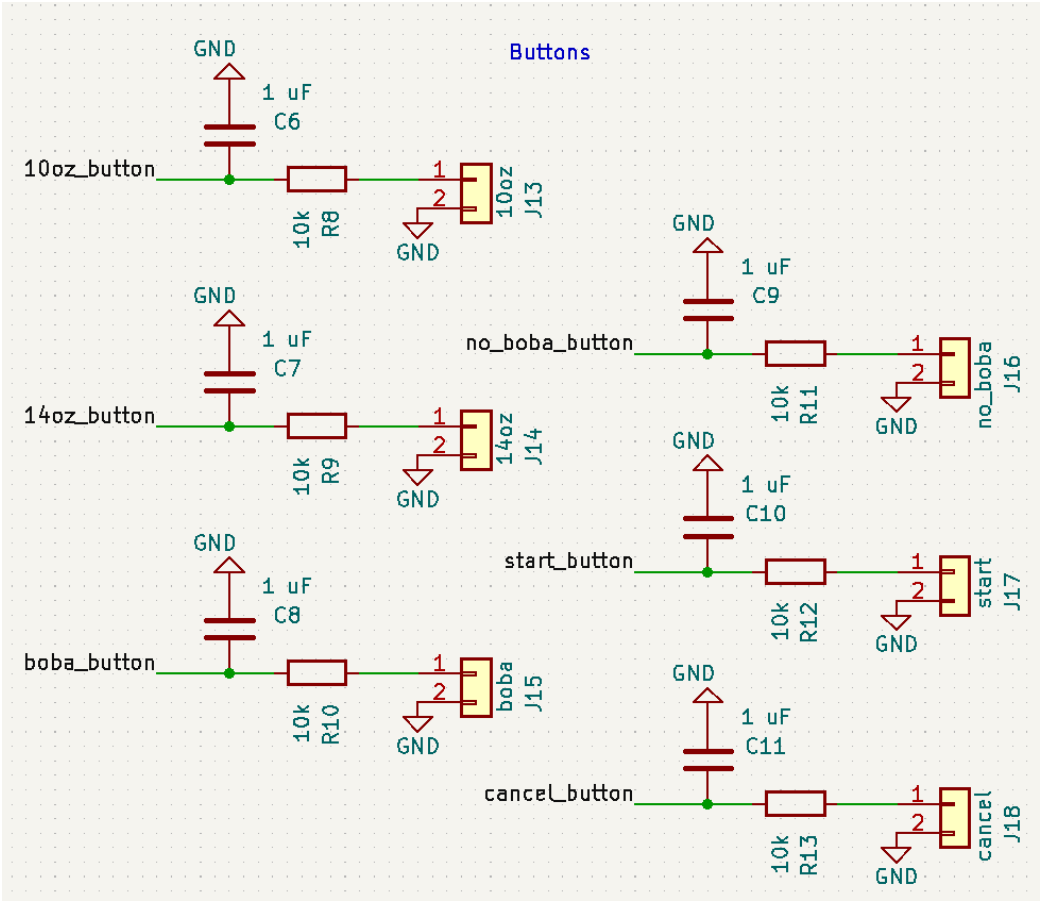

Figure 18. Button Pad Schematic

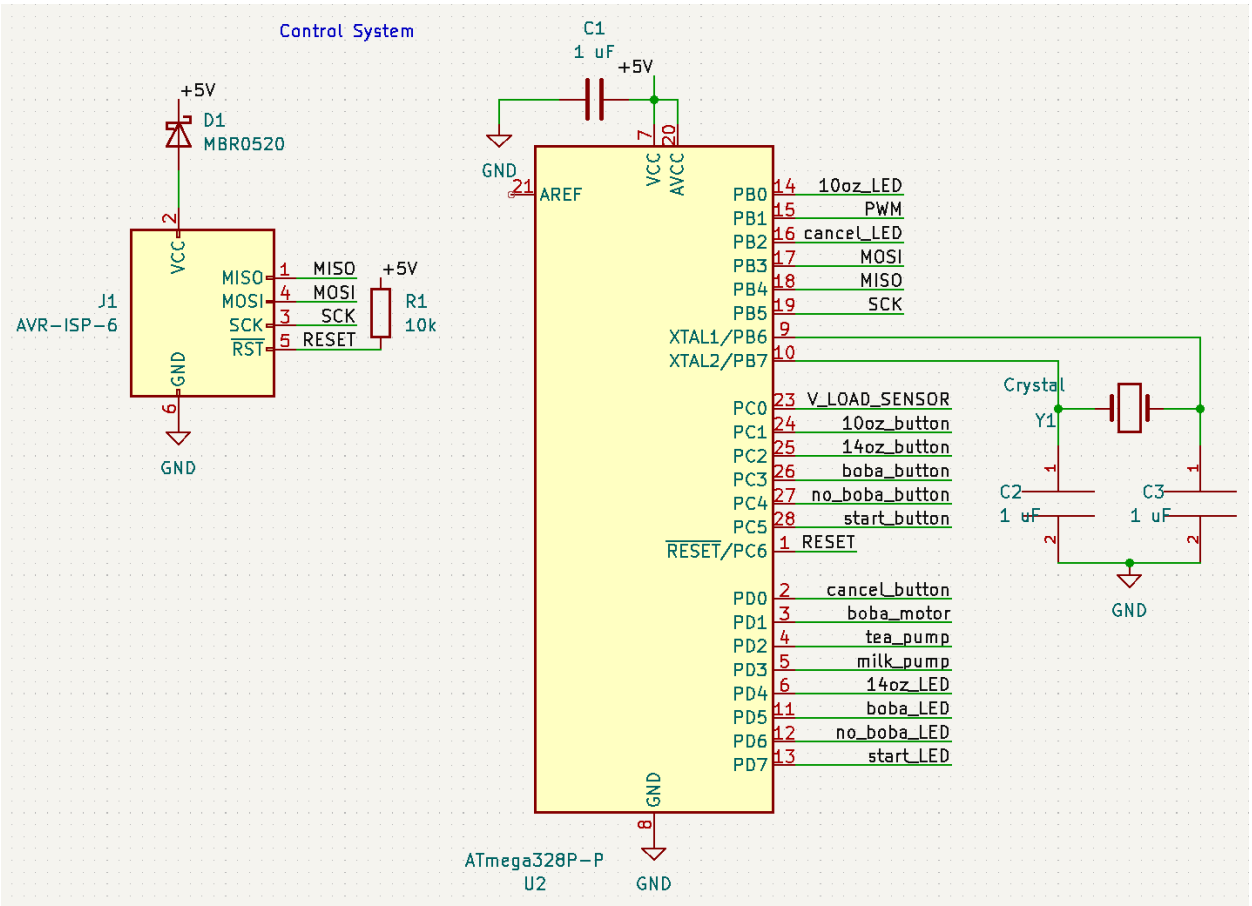

Figure 19. Control System Schematic

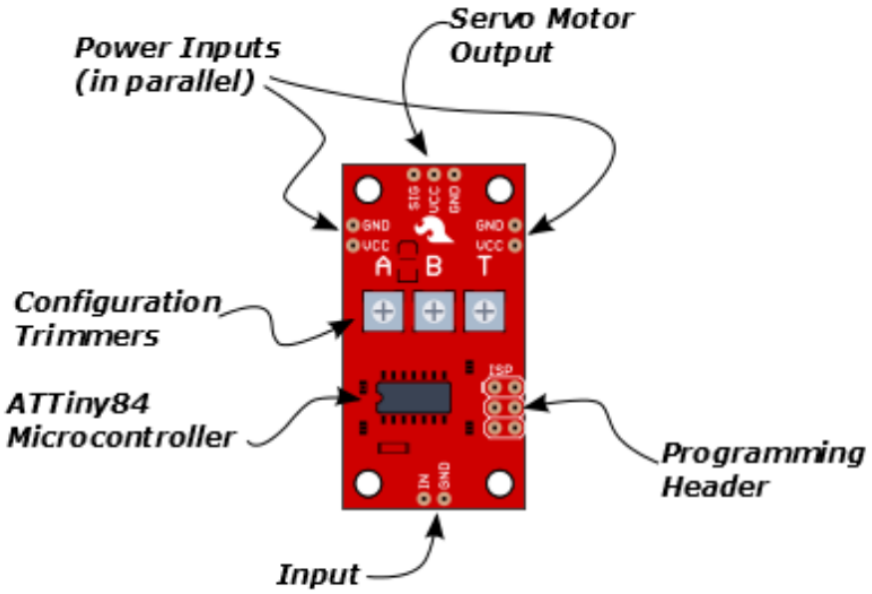

Figure 20. Servo Trigger Pinout

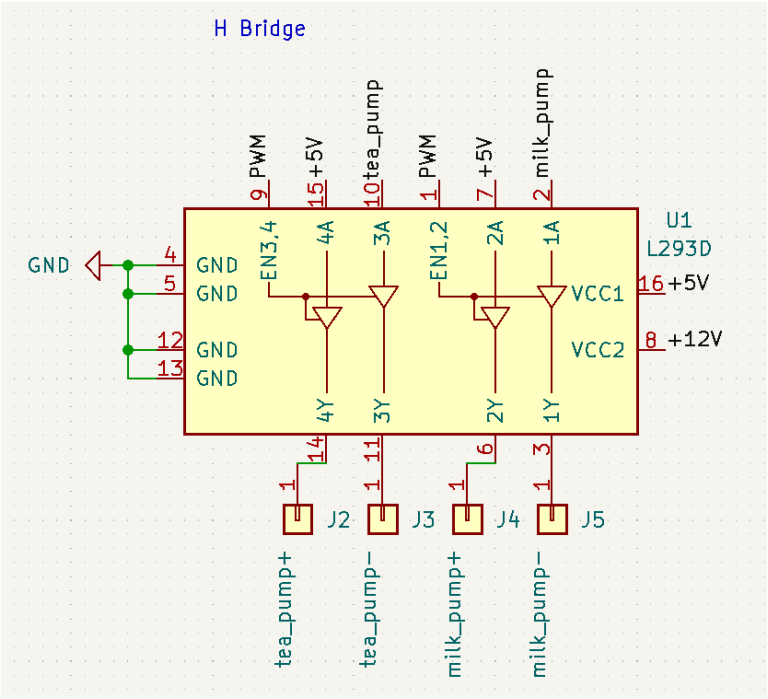

Figure 21. Ingredient Dispenser Schematic

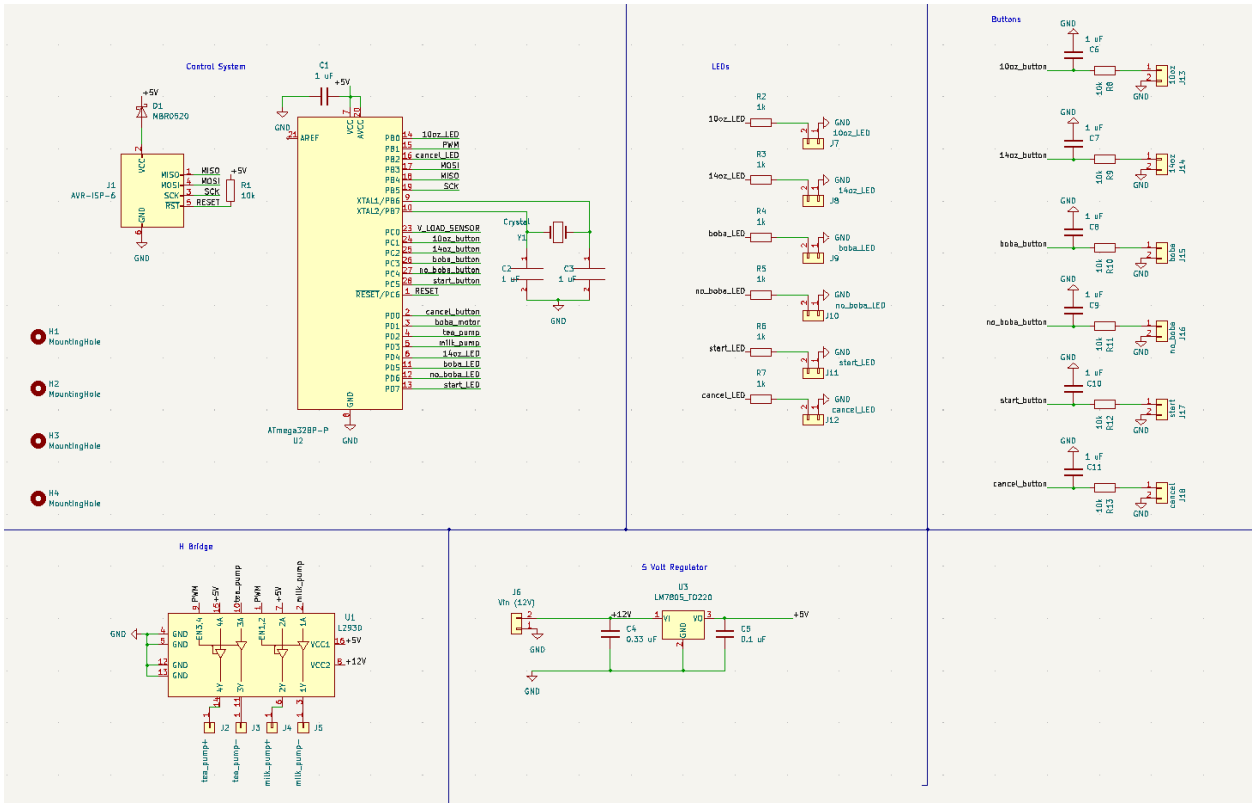

Figure 22. Overall Schematic

Appendix C: Code

```
int boba_closed = 10;    // variable to store the servo position
int boba_open = 70;


void setup() {
  myservo.attach(BOBA_MOTOR);  // attaches the servo on pin 9 to the servo object
  myservo.write(boba_closed);

  // set LED pin modes to output
  pinMode(START_LED, OUTPUT);

  // set motor and pumps pin modes to output
  DDRB = B00000111; // pumps
  PORTB = B00000000;

  // set button pin modes to input
  pinMode(I_10OZ_BUTTON, INPUT);
  pinMode(I_14OZ_BUTTON, INPUT);
  pinMode(BOBA_BUTTON, INPUT);
  pinMode(NO_BOBA_BUTTON, INPUT);
  pinMode(START_BUTTON, INPUT);
  pinMode(CANCEL_BUTTON, INPUT);

  // enable internal pullup
  digitalWrite(I_10OZ_BUTTON, HIGH);
  digitalWrite(I_14OZ_BUTTON, HIGH);
  digitalWrite(BOBA_BUTTON, HIGH);
  digitalWrite(NO_BOBA_BUTTON, HIGH);
  digitalWrite(START_BUTTON, HIGH);
  digitalWrite(CANCEL_BUTTON, HIGH);

  digitalWrite(START_LED, LOW);
  ready_for_size = 1;
  ready_for_boba = 0;
  ready_for_start = 0;
  ready_for_dispense = 0;
}
```

Figure 23. Setup Code

```
void loop() {
//  myservo.write(boba_closed);
  PORTB = B00000000;  // turn off milk&tea
  // put your main code here, to run repeatedly:

      // init user options
      //      value     |   size_choice     |       boba_choice        |     start_choice     |     cancel_choice
      //        0           no selection          no selection              no selection            no_selection
      //        1              10oz                   boba                     start                  cancel
      //        2              14oz                 no_boba                     N/A                    N/A
      size_choice = 0; // we are ready to take a new order
      boba_choice = 0;
      start_choice = 0;
      cancel_choice = 0;

      ready_for_size = 1;
      ready_for_boba = 0;
      ready_for_start = 0;
      ready_for_dispense = 0;

    // take in user input

    // prompt for size buttons

    while(ready_for_size == 1 and cancel_choice == 0){
      digitalWrite(START_LED, LOW);
      delay(200); // wait for 1/5 second
      digitalWrite(START_LED, HIGH);
      delay(200); // wait for 1/5 second

      // store user size choice
      if (digitalRead(I_10OZ_BUTTON) == LOW) {
        size_choice = 1;
        // lock size selection and prepare for boba selection
        ready_for_boba = 1;
        ready_for_size = 0;
      }
      else if(digitalRead(I_14OZ_BUTTON) == LOW){
        size_choice = 2;
        // lock size selection and prepare for boba selection
        ready_for_boba = 1;
        ready_for_size = 0;
      }
```

Figure 24. Code for Size State

```
// prompt boba buttons
while(ready_for_boba == 1 and cancel_choice == 0){

  digitalWrite(START_LED, LOW);
  delay(100); // wait for 1/5 second
  digitalWrite(START_LED, HIGH);
  delay(100); // wait for 1/5 second

  // store user boba choice
  if (digitalRead(BOBA_BUTTON) == LOW) {
    boba_choice = 1;
    // lock boba selection and prepare for start selection
    ready_for_start = 1;
    ready_for_boba = 0;
  }

  else if(digitalRead(NO_BOBA_BUTTON) == LOW){
    boba_choice = 2;
    // lock boba selection and prepare for start selection
    ready_for_start = 1;
    ready_for_boba = 0;
  }

  else if(digitalRead(CANCEL_BUTTON) == LOW){
      ready_for_size = 1;
      ready_for_boba = 0;
      ready_for_start = 0;
      size_choice = 0; // we are ready to take a new order
      boba_choice = 0;
      start_choice = 0;
      cancel_choice = 1;
  }
}
```

Figure 25. Code for Boba State

```
// prompt for start
while(ready_for_start == 1 and cancel_choice == 0){
  digitalWrite(START_LED, LOW);
  delay(50); // wait for 1/5 second
  digitalWrite(START_LED, HIGH);
  delay(50); // wait for 1/5 second
  // store user size choice
  if (digitalRead(START_BUTTON) == LOW) {
    start_choice = 1;
    // lock start selection and prepare for drink to be dispense
    ready_for_dispense = 1;
    ready_for_size = 0;
    ready_for_start = 0;
      digitalWrite(CANCEL_LED, LOW);
  }

  else if(digitalRead(CANCEL_BUTTON) == LOW){
      ready_for_size = 1;
      ready_for_boba = 0;
      ready_for_start = 0;
      size_choice = 0; // we are ready to take a new order
      boba_choice = 0;
      start_choice = 0;
      cancel_choice = 1;
  }
}
```

Figure 26. Code for Start State

```
            // dispense drink
            digitalWrite(START_LED, LOW);

            // dispense boba
            if(boba_choice == 1){
              myservo.write(boba_open);
              delay(1000);
              myservo.write(boba_closed);
              delay(5000);
            }

            // dispense milk and tea
            if(size_choice == 1){
              PORTB = B00000010;  // turn on milk&tea
              delay(143000);
              PORTB = B00000000;
            }

            if(size_choice == 2){
              PORTB = B00000010;  // turn on milk&tea
              delay(258000);
              PORTB = B00000000;
            }

            // reset all signals (prepare for new drink order)
            ready_for_size = 1;
            ready_for_boba = 0;
            ready_for_start = 0;
```

Figure 27. Code to Dispense Ingredients

```
      // check out data line and get clock pulses going
      while(digitalRead(V_LOAD_SENSOR) != LOW) // wait until data line is low
        ;
       {
        PORTB = B00000001;
        for (int i=0; i<24; i++){
          clk();
//          PORTB = B00000000;
        }
        clk(); // 25th pulse
      }

      weight = scale.get_units();
      if(weight == 0.0){
        PORTB = B00000001;
      }
      while(weight < 0.7 and weight > 0.1){ //in lbs: (cup is 0.5 lbs)
        PORTB = B00000001;
        PORTB = B00000100;

      }
      while(scale.read() - weight < 511){ // update number to appropriate value
        PORTB = B00000100;
      }
      PORTB = B00000000;

}

 void clk(){
  digitalWrite(PWM, HIGH);
  digitalWrite(PWM, LOW);
 }
```

Figure 28. Code to Manually Connect Hx711 to MCU