Hands Free Drinks Mixer

ECE 445 Design Review

Spring 2019

Team 40: Dave Ha, Eric Mysliwiec, Matthew Gross

TA: Channing Philbrick
# Hands Free Drinks Mixer

## 1 Introduction

1.1 Objective  
1.2 Background  
1.3 High-Level Requirements

## 2 Design

2.1 Block Diagram  
2.2 Physical Design  
2.3 Power Circuitry
   - 2.3.1 120VAC to 12VDC Converter  
   - 2.3.2 12VDC to 5VDC transformer
2.4 Motors
   - 2.4.1 Pumps  
   - 2.4.2 Disk Stepper Motor
2.5 Sensors
   - 2.5.1 Flow Meters  
   - 2.5.2 Photocell and Laser  
   - 2.5.3 Weight Sensors
2.6 Logic
   - 2.6.1 ESP8266 Microcontroller  
   - 2.6.2 Motor Drivers  
   - 2.6.3 Refill Indication Control  
   - 2.6.4 MUX/DeMUX Logic
2.7 User Interface
   - 2.7.1 RFID  
   - 2.7.2 Push Buttons  
   - 2.7.3 LCD  
   - 2.7.4 Refill LED
2.8 Server
2.9 Supporting Material
   - 2.9.1 Refill Indication Control Circuit and Simulation  
   - 2.9.2 Motor Drivers  
   - 2.9.3 Software Flow Chart  
   - 2.9.4 Photocell/Laser System Test  
   - 2.9.5 Stepper Motor Calibration Calculations  
   - 2.9.6 Throughput Analysis of ESP8266
2.10 Risk Analysis
2.11 Tolerance Analysis

## 3 Cost and Schedule

3.1 Cost Analysis  
3.2 Schedule

## 4 Safety and Ethics

## 5 References
1 Introduction

1.1 Objective

Waiting in lines is something that most people cannot avoid on a day to day basis. In fact, people wait in
lines so often that Richard Larson, an MIT professor, estimates “some people spend a year or two of
their lives waiting in line.” [1] A great way to minimize this waste of time is to decrease the wait time to
get a drink at the bar. For bartenders, trying to serve every customer and finding the right tab behind
the bars can be challenging. Adding more staff to speed up the process is neither space nor cost
effective, especially since some bars have very limited space behind the counter. Smaller lines and faster
processing at bars is needed in order to increase the efficiency of sales, as well as reduce the amount of
stress for both the bartenders and the customers.

Building an automated drink mixer will ensure that there are fewer tasks for bartenders, so the waiting
time could be cut significantly. This machine will assist bartenders by serving customers who want
preselected drink specials for the day. Having this system in place will cause a significant increase in the
amount of drinks served in a given amount of time, by reducing the amount of tasks for bartenders.

1.2 Background

Places such as Pour Bros have incorporated RFID cards to track customers’ drinks and then allows the
customers to pour their drinks themselves. [2] However, not all bars can transition to this style and
atmosphere. Furthermore, Pour bros incorporates a large, clunky station as opposed to a personalized
pouring experience. This results in almost no lines and requires much less workers. The proposed
solution will differ in that it is smaller in size and is personalized, whilst adding the ability to transmit
total sales data via Wi-Fi upon shutdown. This will assist bartenders when lines get long instead of
functioning as a self serve device. The increased rate in drinks being made will keep customers happy
and streamline sales during extreme rush hours.

Additionally, through having an RFID scanner for cards, the system will remove the need for the
bartenders to look for the right tab among many cards. To make the drinks, all that the bartender would
need to do is scoop some ice with the cup, and put it on a slot in the machine. The only task that the
customer needs to do is to place his RFID tag on the sensor and select the drinks through using the
buttons; Ultimately, the responsibility falls upon the customer himself.
1.3 High-Level Requirements

The success of the project will be determined by the following quantified measurements:

- Has to be able to fill a standard 16 oz solo cup with 6 oz (236ml) of liquids in 2 minutes using at least 2 different liquids.
- Has to be able to scan at least 3 unique RFID cards and keep a log of tabs, each tab consisting of 20 or less drinks, until shutdown of the system.
- Has to be able to send the log of tabs in the form of a small text file that is less than 10 kB in size to an email through the Wi-Fi in less than a minute, with full connection to an internet network.

2 Design

The block diagram in Figure 1 contains five main modules: the power circuitry module, motors module, sensors module, logic module and user interface module. The power circuitry ensures that the standard 120 volt wall outlet gets properly converted to usable voltages – 5V DC for the microcontroller and sensors and 12 VDC for the custom PCB and the motors module. The motors module controls the core of the system. This includes rotating the disk with the stepper motor and pumping/dispensing the liquids. The sensors module is used for the fine tuning of the system. Things like measuring the flow of liquids, calibrating the position of the cup and warning when an ingredient is low can all be done within the sensors module. The logic module is the brains of the system. This module uses data in two different ways. Firstly, the microcontroller reads various inputs and displays the menu on the LCD for proper drink selection. Secondly, the refill indication control uses driving logic from the weight sensors beneath each liquid tank to identify empty bottles. Finally, the user interface module allows the user to interact with the system. Any request for a drink or information being delivered to the user is done here. Independent from the system, a server or a network-connected computer is necessary to receive tab log file from the machine upon system shutdown sequence.
2.1 Block Diagram

Figure 1. Block Diagram
2.2 Physical Design

The blue parts in Figure 2 will be cut from wood and fastened with screws and glue. The yellow disks will be 3D printed with the black cutout (see Figure 2) used to hold the cup while the red cutout (see Figure-2) is used for calibration with a photocell sensor. The top disk will be attached to the wooden structure with screws. The bottom disk will be attached to the stepper motor which will be sitting in a cutout in the wood (not shown). In the case that the disk struggles to support the weight of the cup, two supports can be easily added.

2.3 Power Circuitry

Distributing power to components is essential to the success of this project. Not enough power will cause the components to stop running, while too much power could potentially destroy components.
2.3.1 120VAC to 12VDC Converter

Most components in the project will be unable to handle the 120V supplied by the wall outlet. This converter will lower the voltage to a much more usable 12V.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Must supply 12VDC +/- 20% and 2A +/- .25A from a 120VAC source.</td>
<td>1) To verify:</td>
</tr>
<tr>
<td></td>
<td>a) Use a multimeter to measure the voltage supplied by the converter.</td>
</tr>
<tr>
<td></td>
<td>b) Then use an oscilloscope to verify stability.</td>
</tr>
<tr>
<td>2) Must maintain thermal stability below 50 degrees Celsius while supplying</td>
<td>2) While performing the above test, use a thermal camera to view the</td>
</tr>
<tr>
<td>2A at a minimum for a duration of 5 minutes.</td>
<td>temperature.</td>
</tr>
</tbody>
</table>

2.3.2 12VDC to 5VDC transformer

There are some components in the project that can only withstand 5V. A switching voltage regulator will be used to step down the voltage instead of a linear regulator in order to avoid needing a heatsink and fan. [3]

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Must supply 5V +/- 10% with at least 0.5 +/- 0.1 A. [4]</td>
<td>1) To verify:</td>
</tr>
<tr>
<td></td>
<td>a) Use a multimeter to measure voltage</td>
</tr>
<tr>
<td></td>
<td>b) Use an oscilloscope to verify stability.</td>
</tr>
<tr>
<td></td>
<td>c) Then use a 10 Ohm resistor and multimeter to verify amperage output.</td>
</tr>
</tbody>
</table>

2.4 Motors

The motors carry out the mechanical tasks of the machine such as moving liquid from the tanks to the cup. Furthermore, it moves the cup such that it accurately rests below dispensers for the duration of the pour.
2.4.1 Pumps

The pumps will have different dispensing rates depending on the type of liquid being dispensed. Per expectations, there will need to be a larger rate of dispensing for non-alcoholic drinks than alcoholic.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| 1) Alcoholic liquid pumps must provide at least 100 ml of fluid per minute. | 1) To verify:  
  a) Turn on the pump until water is being dispensed.  
  b) Then turn on the pump for one minute and measure how much fluid was dispensed. |
| 2) Non-alcoholic liquid pumps must provide at least 250 ml of fluid per minute. | 2) To verify:  
  a) Turn on the pump until water is being dispensed.  
  b) Then turn on the pump for one minute and measure how much fluid was dispensed. |

2.4.2 Disk Stepper Motor

The stepper motor will allow us to position the cup under whatever nozzle is about to be dispensing. It will have a one way communication to the microcontroller and can be re-centered when necessary. [5] The stepper motor will give 28 N of force at 20 mm which translates to 5.6 N of force at the center of the cup (100 mm). [6]

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| 1) Stepper motor must be able to consistently align a given nozzle within 1 inch from the center of the solo cup. This translates to being within around +/- 11.46 degrees of the target. | 1) With the disk connected to the stepper motor and the stepper motor connected to a cutoff switch:  
  a) Engage cutoff switch.  
  b) Mark how far the disk continues to rotate after cutoff switch has been pressed. |
| 2) Must be able to occasionally and accurately calibrate itself to within +/- 5.73 degrees of the center of the cup when triggered by the photocell/laser system. | 2) Use the method for requirement one.  
  a) If this method does not produce accurate enough results, photocell/laser system may be needed to cut off the motor at an exact location. |
2.5 Sensors

This project’s sensors will be purposed to re-calibrate the position of the cup, measure the amount of fluid dispensed, and detect when the weight of an ingredient is below an arbitrary threshold.

2.5.1 Flow Meters

The flow meters will increase accuracy in dispensing of the liquids. Attached to the tubing, it will collect the flow data real-time and send it to the microcontroller to logically calculate when to stop the dispensing. The standard liquid volume of a full red solo cup is 12 fl.oz (355ml) [7] so the dispensed volume must be half of that to account for ice volume.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| 1) Must be able to calculate a total dispensed volume of 178ml +/- 10 ml. | 1) To verify:  
  a) Run a premeasured amount of fluid through the meter.  
  b) Compare the meter value to the actual value. |

2.5.2 Photocell and Laser

The photocell and laser will allow us to calibrate the stepper motor positioning of the cup using the microcontroller. This is necessary to do at startup as well as during use in order to avoid error in the stepper motor positioning.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| 1) Photocell must be able to detect the laser going through a 1/8” radius hole in under a second. | 1) Write a simple program that does the following:  
  a) Read the photocell data.  
  b) Print this data along with time stamps.  
  c) Shine a laser at the photocell and see how quickly it detects this change. |
2.5.3 Weight Sensors

The weight sensors will be used to detect when an ingredient is getting low. It will then notify the user by lighting up an LED. In order to know the proper weight at which the system notifies the user, custom bottles of uniform weight (plastic 2L bottle) will be used, which weigh 45 g empty and 2042 g full. 

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| 1) Must be able to withstand up to 2.042 kg. | 1) To verify:  
   a) Test sensor reading at 1 kg.  
   b) Then place 2.042 kg or more on the sensor.  
   c) Retest the sensor reading with 1 kg on the sensor and compare to first test read. |
| 2) Must be able to differentiate 270 g from 270 g +/- 75 g. | 2) To verify:  
   a) Measure 270 g on a precise scale and compare value to the sensor’s value.  
   b) Then add measure another 75 g and add it to the sensor. |

2.6 Logic

The logic unit serves as a central processing unit for all of the data collected through the sensor, giving orders to specific modules or devices as needed. It will get the RFID input and store it in a log of tabs, as well as sending the data to the server (or an email). It will control the selection of drinks through push buttons, as well as displaying different drinks to the LED. Furthermore, it will make sure that the cups rotate the right amount of steps to correctly position beneath the required nozzle for the drink recipe.

2.6.1 ESP8266 Microcontroller

The microcontroller, powered by 3.3VDC, will run the written program to interface all data and make decisions accordingly. The program can be divided into seven parts, as seen software logic diagram (Figure 8) in section 2.9.3:

1. Establishing connections with all sensors and modules, and initializing all signals and data structures to zeros
2. Wait for customer interaction through an RFID sensor, and show selection menu on the LCD screen when the user scans his/her id/sensor
3. Interact with user choices through button signals: navigating through the menu, and selecting/cancelling order
4. Updating the data structure with proper key-pair value of user ID and number of drinks bought
5. Sending rotation signals to the step motor, and liquid dispensing signal to the peristaltic pump to make the mix according to the recipe stored in the program.
6. Reset and align the disk to its zero position, indicating that the drink is done. Return to step 2
7. Upon holding the cancel button for 3 seconds, initiate the shutdown sequence; send the log file to an email (server), and power off the motor. Indicate on LCD that the machine is good to be unplugged.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Must be able to send data to LCD through Parallel Digital I/O connection.</td>
<td>1) LCD functionality will be verified with running code; specifically, by running <code>lcd.print(&quot;Hello World&quot;)</code> and visually verifying the displayed message on board. Exact steps in section 2.7.3.</td>
</tr>
<tr>
<td>2) Must be able to send and receive data to/from RFID sensor through SPI connection.</td>
<td>2) After initializing the RFID module, read the value through <code>mfrc522.PICC_ReadCardSerial()</code> and print values inside <code>mfrc522.uid.uidByte</code> buffer onto terminal. Exact steps in section 2.7.1.</td>
</tr>
<tr>
<td>3) Must be able to read push button press into a variable, through digital I/O pins.</td>
<td>3) Set pinmode of of any GPIO’s to Input, and print make a buttonState variable and continuously loop and read value. Exact steps in section 2.7.2.</td>
</tr>
</tbody>
</table>
| 4) Must be able to send pulses of 3.3V onto the stepper motor through digital I/O pin. | 4) Connect the motor driver circuit to pin 3, and run the following sketch code:
   a) inside setup(), call pinmode(3, OUTPUT);
   b) inside loop(), call digitalWrite(3, HIGH);
   c) visually confirm that motor rotated half a step.
   d) call digitalWrite(3, LOW);
   e) visually confirm that motor rotated another half a step. |
| 5) Must be able to send 3.3V signal to the right flow meter & peristaltic pump, and read the output pulse from the flow meter through the digital I/O pin. | 5) Connect the flow meter input to pin 3, flowmeter output to pin 2, and peristaltic pump to pin 4 of ESP8266. Pump pipe must be fully immersed in any liquid. Then run the following sketch code: |
| 6) Must be able to send 3.3V signal to the pinhole laser through digital I/O. | 6) Connect the laser to pin 3, and run the following:  
|---|---|
| a) inside setup(), call pinmode(3, OUTPUT);  
b) call pinmode(2, INPUT); to control the flowmeter.  
c) call pinmode(4, OUTPUT); to control the pump.  
d) inside loop(), call digitalWrite(3,HIGH); and digitalWrite(4,HIGH); to power the pump and flowmeter  
e) serial.print(digitalRead(2)); to see the flowmeter value.  
f) visually confirm that motor, and flowmeter are operational.  |
| 7) Must be able to receive the value of photo cell through Analog-to-digital pin. | 7) Connect the sensor to pin A1, and run the following:  
|---|---|
| a) inside setup(), call pinmode(A1,INPUT);  
b) inside loop(), call serial.print(analogRead(A1));  
c) shine a laser on the sensor, and visually verify that the value read changes as laser is turned on/off.  |
| 8) Must be able to set the LED through digital I/O pin to indicate successful WiFi connection. | 8) Connect LED to pin 3, and run the following sketch:  
|---|---|
| a) #include “ESP8266WiFi.h” and set const char* ssid and const char * password to right values  
b) in setup(), call Wifi.begin(ssid,password); pinmode(3,OUTPUT);  
c) if(WiFi.status() == WL_CONNECTED) |
9) Must be able to send a file smaller than 1MB in size in 91 ms, on a 802.11 a/b/g/n network (802.11b network has speeds up to 11 Mbps, whereas others go up to 600 Mbps. Across different networks of a/b/g/n, the speed should average higher). Due to speed of additional programming constructs, the upper bound for acceptable speed will be 1s.

9) Call the following sketch code:
   a) get time before sending email through time = millis()
   b) Send email through gsender->Send(to, message);
   c) get time afterwards, and subtract time to see if it meets requirements by running Serial.print(millis() - time).
   Thoroughly discussed in section 2.8

2.6.2 Motor Drivers

In order to prevent damaging the microcontroller, a separate system of transistors is needed to control the pumps and stepper motors. To use the least amount of current through the microcontroller, the control pin will drive a MOSFET which allows a 5VDC voltage to turn on a BJT. The BJT will then take the 12VDC from the power supply to drive the motors. The motor circuitry will also be protected using diodes to handle any sudden spikes in voltage from a blockage or mechanical failure.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Must provide 12VDC +/- 1V to all motors.</td>
<td>1) Measure test point at end of the Motor Driver circuit while motor is connected and ON, measure for 12VDC in parallel</td>
</tr>
<tr>
<td>2) Must provide at least 0.35 +/- 0.05 A to drive the stepper motor.</td>
<td>2) Link Digital Multimeter in series from the end of the Motor Driver to the pump motor and measure the current</td>
</tr>
<tr>
<td>3) Must provide at least 0.8 +/- 0.05 A to drive the high flow pump motors. [10]</td>
<td>3) Link Digital Multimeter in series from the end of the Motor Driver to the stepper motor and measure the current</td>
</tr>
<tr>
<td>4) Must provide ground-fault for the pump motors.</td>
<td>4) Connect end of motor driver directly to ground, send a 500 ms pulse to the driver from the control side, see if the breaker tripped</td>
</tr>
</tbody>
</table>
2.6.3 Refill Indication Control

This will be a custom built PCB, utilizing a voltage divider, an op-amp and an n-channel mosfet. This drives an LED when the weight drops to near 10% of the maximum fluid in the tanks. [7][11][9]

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Must drive an LED at 5V +/- 2V with 50 mA +/- 20 mA when weight drops to 270 grams +/- 75 grams.</td>
<td>1) Configure the circuit but use two potentiometers instead of a weight sensor to confirm the circuit’s function.</td>
</tr>
</tbody>
</table>

2.6.4 MUX/DeMUX Logic

The Multiplexor and Demultiplexor logic serve as both a way to expand upon the microcontroller’s lack of pins and as a voltage step-up to communicate with the rest of the 5V logic signals. The Microcontroller can receive 5V logic but cannot transmit at that voltage. The flow sensor logic is also paired in because of the symmetry of the flow of information.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| 1) Must be able to output 5V +/- 0.5V with at least 20mA when given a 3.3V signal | 1) To verify:  
  a) Set Digital MultiMeter to DC Volts  
  b) Set power supply to 3.3V and connect to rails on breadboard  
  c) Connect rails to all select pins to make binary 7  
  d) Connect enable pin to 5V high  
  e) Measure DC voltage out of each pin to see that 5V is output at only the selected pin  
  f) Measure the current going from the current selected pin to a 1k Ohm resistor  
  g) Repeat from Step 3 for all binary numbers from 6 down to 0 |

2.7 User Interface

User interface will provide a means for the user to interact with the system. Through the RFID sensor, the user will be able to let the system know who is buying the drink, and increment the value on the tab. Through the buttons, the user will be able to select the drinks, and cancel the order during the selection stage. The LED will display relevant information as the user progresses through the order. Furthermore, the refill LED will light up when the liquid tank runs low.
2.7.1 RFID

The RFID sensor will need to correctly read the unique value stored in a card, namely i-cards. Furthermore, it will need to communicate with the ESP8266 microcontroller and send the data over to be processed.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Must be able to read at least 3 different RFID tags through SPI, each within 0~3 seconds (clock speed on ESP8266 is 80 MHz, and on SPI, half. With slowest rates of 8bit/writes, a 32-byte string should take $32 \times 40MHz = 0.8ms$. Adding time for other programming constructs such as loops, should be less than 1s)[12].</td>
<td>1) Run the following:   a) time = millis()   b) read in the RFID card through mfrc522.PICC_ReadCardSerial()   c) check time difference by serial.print(millis() - time);   d) Do it three times for different cards, and see if the content in the buffer differs. Repeat, and see if the content matches that of earlier iteration.</td>
</tr>
</tbody>
</table>

2.7.2 Push Buttons

Push buttons, powered by the 5V DC transformer, will need to correctly send the push signals to the ESP8266 microcontroller upon being pressed and released. There will be four buttons: left, right, select and exit/shutdown. Button logic is explained in the Microcontroller submodule section 2.7.1.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) The push buttons must respond to pressure within $\frac{5}{8}$ of its center.</td>
<td>1) connect the button on the board to pin 2, and run the following:   a) inside setup(), call pinmode(2, INPUT);   b) inside loop(), call buttonState = digitalRead(2);   c) serial.print(buttonState);   d) press within $\frac{5}{8}$ of its center and see response on console</td>
</tr>
</tbody>
</table>
2.7.3 LCD

The LCD, powered by 3.3 VDC, will display the value formed by the Logic module through getting the input from the microcontroller. Will display drink names during selection, and the ID upon scanning the RFID.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1) Must display the value set by the logic module (ESP8266) through I2C; since 2 rows of 16 characters are transferred at most, 32 bytes of data are transferred; these need to be transferred within 50 ms (since clock speed of ESP8266 is 80 MHz, and 32 * 80 MHz = 0.4 ms, but accounting for other programming constructs, upper bounded to 50 ms).</strong></td>
<td><strong>1) run the following:</strong></td>
</tr>
<tr>
<td></td>
<td>a) <code>#include &lt;LiquidCrystal.h&gt;</code></td>
</tr>
<tr>
<td></td>
<td>b) set button constants for different digital i/o inputs, as following: <code>RS=D2, EN=D3, d4=D5, d5=D6, d6=D7, d7=D8;</code></td>
</tr>
<tr>
<td></td>
<td>c) in setup(), set row, col by <code>lcd.begin(16,2)</code></td>
</tr>
<tr>
<td></td>
<td>d) <code>time = millis();</code></td>
</tr>
<tr>
<td></td>
<td>e) print message by <code>lcd.print(time-millis());</code></td>
</tr>
<tr>
<td></td>
<td>f) visually confirm message on lcd, and see time spent</td>
</tr>
</tbody>
</table>

2.7.4 Refill LED

The refill LED will light for the operator when a tank is below an arbitrary threshold and needs to be refilled.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1) Must be clearly visible from 2 meters away with a drive current of 50 +/- 20 mA.</strong></td>
<td><strong>1) Connect the LED to a 5V power supply with a series resistor and step back to determine if it is visible by 2 meters.</strong></td>
</tr>
</tbody>
</table>

2.8 Server

The server will receive the log of bar tabs upon the shutdown sequence from the microcontroller. Through the Wi-Fi module, the microcontroller will send a text file of entries, which consist of key-value pair from user ID to number of drinks purchased. Since having a server is outside the scope of this class and project, data will simply be received through an email transmission (SMTP).
### Requirement

1) **Must be able to receive email from the Wi-Fi module on ESP8266 through SMTP protocol, independent of web client or OS.** Since speed of email transfers are heavily dependent on the backend services of each domain, the target time to receive email is within 1 minute.

### Verification

1) **Run the following:**
   a) `#include “Gsender.h”`
   b) `time = millis();`
   c) `send email by running if(gsender->Subject(subject)->Send(emailstring,”Setup Test”)) serial.print(“Success”)`
   d) `serial.print(millis() -time);`
   e) `visually confirm email on another computer, and check time`

---

### 2.9 Supporting Material

#### 2.9.1 Refill Indication Control Circuit and Simulation

Figure 3 shows a mock design of the refill indication control can be achieved using a voltage divider, operational amplifier, n-channel mosfet and an LED. As seen in the simulation results (Figure 4), the LED would have been theoretically illuminated for values above 14 Ohm.

![Refill Indication Control Schematic](image)

**Figure 3. Refill Indication Control Schematic**
2.9.2 Motor Drivers

Figure 5 contains the simulation of a brushed motor driver. The motor is simulated with a resistor and inductor since LTSPICE does not have a motor component to test. Motors in real life will not be linear and will have built up inertia which will serve as a source when the Bipolar Junction Transistor moves from saturation to cut-off. This excess energy would destroy most BJT’s and thus the diode next to the simulated motor provides a path back so the current winds down with the motor’s natural friction instead of destroying other components. Figure 6 and Figure 7 show the results of this simulation as V2 is swept from 0V to 5VDC. The motor is rated for 12VDC and 0.8A which the system operates within as shown by this simulation.
Figure 6. Motor Driver Voltage Simulation

Figure 7. Motor Driver Current Simulation
2.9.3 Software Flow Chart

![Software Flow Chart](image)

2.9.4 Photocell/Laser System Test

A simple arduino program was written to print the data received at the analog to digital pin. A photocell from the 3.3V power supply was connected to the A to D pin and a resistor was connected from this pin to ground. Finally, a laser was shined onto the photocell, giving the output shown in Figure 9. The zeros represent no laser and the 1023 represents the laser being shined. This test not only proved this system
would work but also showed that the system is not sensitive to nearby lighting and a pinhole is not needed for the photocell.

Figure 9. Photocell/Laser System Output
2.9.5 Stepper Motor Calibration Calculations

Figure 10 demonstrates how values of tolerance was calculated for the cup positioning and calibration.

\[
\theta = \sin^{-1}\left(\frac{r/2}{R}ight) \quad \text{where } \Theta \text{ is the degrees of the circle of radius } R \text{ needed to catch a circle of radius } r.
\]

Example: the cup has a diameter of 3.75 inches, which gives a radius of 1.875 inches. The cup will sit 3 inches out from the center of the turning disk; thus \( \theta = \sin^{-1}\left(\frac{1.875}{3}\right) = 18.21^\circ \). The dispenser should be 1" from the center of the cup so the radius shrinks from 1.875 to 1 = r, which gives 9.594°. The recalibration distance is arbitrarily set at 0.5 inches which gives a +/- 4.78° tolerance. These are both well within the 1.8° step size of the chosen stepper motor so tolerance will not be exceeded.

2.9.6 Throughput Analysis of ESP8266

The average data transfer rate from an ESP8266 to a local machine, with underutilized download speed for the machine is shown in Figure 11 representing time series of data.
There are some drops and peaks, and the speed is not stable throughout; however, the mean speed comes out to 341 Kilobytes per second, which definitely satisfies the project needs to transfer a small text file.

### 2.10 Risk Analysis

This project is a combination of many smaller subsystems and components, and as such there is the inherent risk of an integration failure. The individual subsystems interfacing with each other poses the largest risk because the calculations depend upon a certain level of accuracy when using control logic to drive and decipher the data coming in and going out. Therefore, the block with the largest amount of risk attached to it is the central logic unit, or more specifically the ESP8266. These microcontrollers were originally developed to be as cheap as possible and as such continue to run the risk of shorts and fried board to this day. As the project begins to be physically realized, there are expected to be errors and problems with the logic unit as it is flushed out to completion. As the project progresses into the integration stage, there runs an even greater risk of destroying all of the development of the system around the ESP8266. Overdriving the ESP8266 with too much current or voltage will destroy all of the work put into it, delay development and testing times, and plummet team morale. To compensate for this, multiple ESP8266 units will be kept on hand in expectation for this inevitable failure. The greatest weakness of this project will be setting successful operation upon a fragile central computing unit with a known history of failure.

### 2.11 Tolerance Analysis

The design most sensitive to tolerances in this project is the calibration and rotation of the bottom disk. If the stepper motor cannot align the cup under each nozzle the fluids will be spilled everywhere. Also if it is unable to calibrate accurately and consistently, the stepper motor will struggle to align the cup with a nozzle over time.

The numbers were chosen carefully by using measurements and trigonometry calculations. Various brands of red solo cups were measured, yielding that all brands have a 3.75 inch diameter or larger at the opening of the cup. Given this length, it was determined that, for safe functioning, the stepper motor must center the cup under each nozzle within one inch from the center of the cup. This will
ensure that no fluids are being dispensed outside of the cup. Since the stepper motor has one inch of error allowed, there is, at the least, only .875 inch left until fluids miss the cup. This is why .5 inch was chosen as the max calibration error. Given the worst case scenario, the stepper motor misaligns the cup by 1 inch and the calibration is off by .5 inch. There still remains .375 inch of space until fluid will be dispensed outside of the cup.

3 Cost and Schedule

3.1 Cost Analysis

When the project is complete, the overall cost of development will include both the time each project member put into working on the prototype as well as the price of all the parts used for the prototype. To calculate the labor cost, it is assumed that each of the three members will be making around 40 dollars an hour, working 10 hours a week and will do this for 16 weeks. The total labor cost is calculated as as shown in Equation 1.

\[ \text{Total Labor Cost} = 3 \text{ (members)} \times \frac{\$40}{\text{hr}} \times \frac{10 \text{ hr}}{\text{wk}} \times 16 \text{ (weeks)} \times 2.5 = \$48,000 \]

\[ (\text{EQ 1}) \]

The parts list and costs for both the prototype and the potential bulk production price are shown in Figure 12.

<table>
<thead>
<tr>
<th>Part</th>
<th>Cost (prototype)</th>
<th>Cost (bulk ~100)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFID reader (Amazon)</td>
<td>$6.99</td>
<td>$5.59</td>
</tr>
<tr>
<td>Push Button x4 (DigiKey)</td>
<td>$.96</td>
<td>$.80</td>
</tr>
<tr>
<td>Mini LCD (DigiKey)</td>
<td>$10.25</td>
<td>$8.10</td>
</tr>
<tr>
<td>LED x5</td>
<td>$0.75</td>
<td>$0.60</td>
</tr>
<tr>
<td>12V DC to 5V DC transformer (DigiKey)</td>
<td>$2.88</td>
<td>$2.56</td>
</tr>
<tr>
<td>120V AC to 12V DC Converter (Amazon)</td>
<td>$15.84</td>
<td>$12.68</td>
</tr>
<tr>
<td>High Flow Rate pumps x5 (Amazon)</td>
<td>$129.00</td>
<td>$60.75</td>
</tr>
<tr>
<td>Stepper motor (Amazon)</td>
<td>$13.99</td>
<td>$11.19</td>
</tr>
<tr>
<td>NodeMCU (Amazon)</td>
<td>$8.39</td>
<td>$2.00</td>
</tr>
<tr>
<td>MCP23017 Expander chip (DigiKey)</td>
<td>$1.24</td>
<td>$0.94</td>
</tr>
<tr>
<td>Shields (PCBWay)</td>
<td>$8.00</td>
<td>$0.60</td>
</tr>
</tbody>
</table>
With a labor cost of $48,000 and a prototype cost of $312.16, the total cost of the project will be about $48,312.

### 3.2 Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Eric: Flow Meters, Photocell/Laser Sensor, &amp; Refill Indication System</th>
<th>Matthew Motor Driver &amp; Motors, Power Circuit, Breakout Logic</th>
<th>Dave ESP, LCD, RFID, pushbuttons</th>
</tr>
</thead>
</table>
| 2/25 | 1. Schematic  
1. Flow Meters  
2. Photocell/Laser Sensor  
2. Simulations  
3. Datasheets  
4. Order Parts  
5. Build Prototypes of Breakout Logic, Power Circuit | 1. Plan out entire overview of system logic, including pin assignments  
2. Plan out recipes and positions of ingredients, create data structures  
3. Find libraries and tutorials for all sensors and devices  
4. Order Parts |
| 3/4  | 1. Finalize physical design with Machine Shop  
2. Build Prototypes of Flow Meters  
3. Design, get approved and order PCBs | 1. Test Prototype Circuits  
2. Adjust schematics as necessary | 1. Prototype all logic on the microcontroller:  
a. connect to Wi-Fi network with proper credentials  
b. connect MCP23107 expansion chip  
c. connect and verify RFID, LCD, Buttons  
d. connect MUX and verify signals sent to stepper motor, flowmeters, peristaltic pumps, pinhole laser |

Figure 12. Cost Analysis Table
<table>
<thead>
<tr>
<th>Date</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/11</td>
<td>1. Test and prototype all circuits</td>
<td>1. Test Motor Driver</td>
<td>1. Test Wi-Fi file transmission through email</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Test Power Circuit</td>
<td></td>
</tr>
<tr>
<td>3/18</td>
<td>SPRING BREAK</td>
<td>SPRING BREAK</td>
<td>SPRING BREAK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pump Motor Characterization</td>
<td></td>
</tr>
<tr>
<td>3/25</td>
<td>1. Integrate Sensors with breakout logic</td>
<td>1. Integrate Power Circuit with motor driver, breakout logic</td>
<td>1. Integrate physical container and disc system with microcontroller</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Integrate the PCB together with all the parts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Test make a drink</td>
</tr>
<tr>
<td>4/1</td>
<td>BUFFER TIME</td>
<td>BUFFER TIME</td>
<td>BUFFER TIME</td>
</tr>
<tr>
<td>4/8</td>
<td>BUFFER TIME</td>
<td>BUFFER TIME</td>
<td>BUFFER TIME</td>
</tr>
<tr>
<td>4/15</td>
<td>Mock Demo’s</td>
<td>Mock Demo’s</td>
<td>Mock Demo’s</td>
</tr>
<tr>
<td>4/22</td>
<td>Real Demo and Mock Presentation</td>
<td>Real Demo and Mock Presentation</td>
<td>Real Demo and Mock Presentation</td>
</tr>
<tr>
<td></td>
<td>Prepare final report</td>
<td>Prepare final report</td>
<td>Prepare final report</td>
</tr>
</tbody>
</table>

**Figure 13. Proposed Schedule**

## 4 Safety and Ethics

There are two main issues that the project could potentially involve. The first is the potential to over-serve customers by having an automated drink system put into place. This is a conflict with IEEE Code of Ethics 7.8.1 which states that engineers must hold in high value the “health, and welfare of the public.”[14] To solve this, a drink limiter logic will be implemented, where each individual’s account will hold the number of drinks bought. After a predetermined limit is reached, the customer will need approval in order to place his/her order. The second issue is the potential to serve customers who are underage. Allowing this with the project, especially on a college campus, would be a clear conflict with the pledge to “to strive to comply with ethical design”[14] In order to prevent this, bars would have a system to register an RFID only to customers who provide proper identification. This project does not deal with batteries or any other potential safety issues, and uses no voltages above a normal household outlet. Given that the outlet is following IEEE standards, it should have its own circuit protection. In addition to this, and to protect the people and technology in use within and around this product, power source was chosen carefully -- that contains short-circuit protection, over-voltage shutdown, and an
internal fuse to ensure the protection of the community’s health. Another possible safety issue is a faulty stepper motor. If by some Act of God or through some mechanical problem the stepper motor malfunctioned, it could spin out of control and send parts of this project and drinks flying. To solve this problem, an exit sequence is implemented to manually force a shutdown on motors sub-system upon hitting exit button over 3 seconds in the main menu.
5 References


