# Smoothie Recipe Maker

Team 49 – Avyay Koorapaty, Anay Koorapaty, Max Gendeh

ECE 445 Design Document – Spring 2025

TA: Jason Zhang

Mar. 6th, 2025

# Table of Contents

# 1. Introduction

## 1.1 Problem

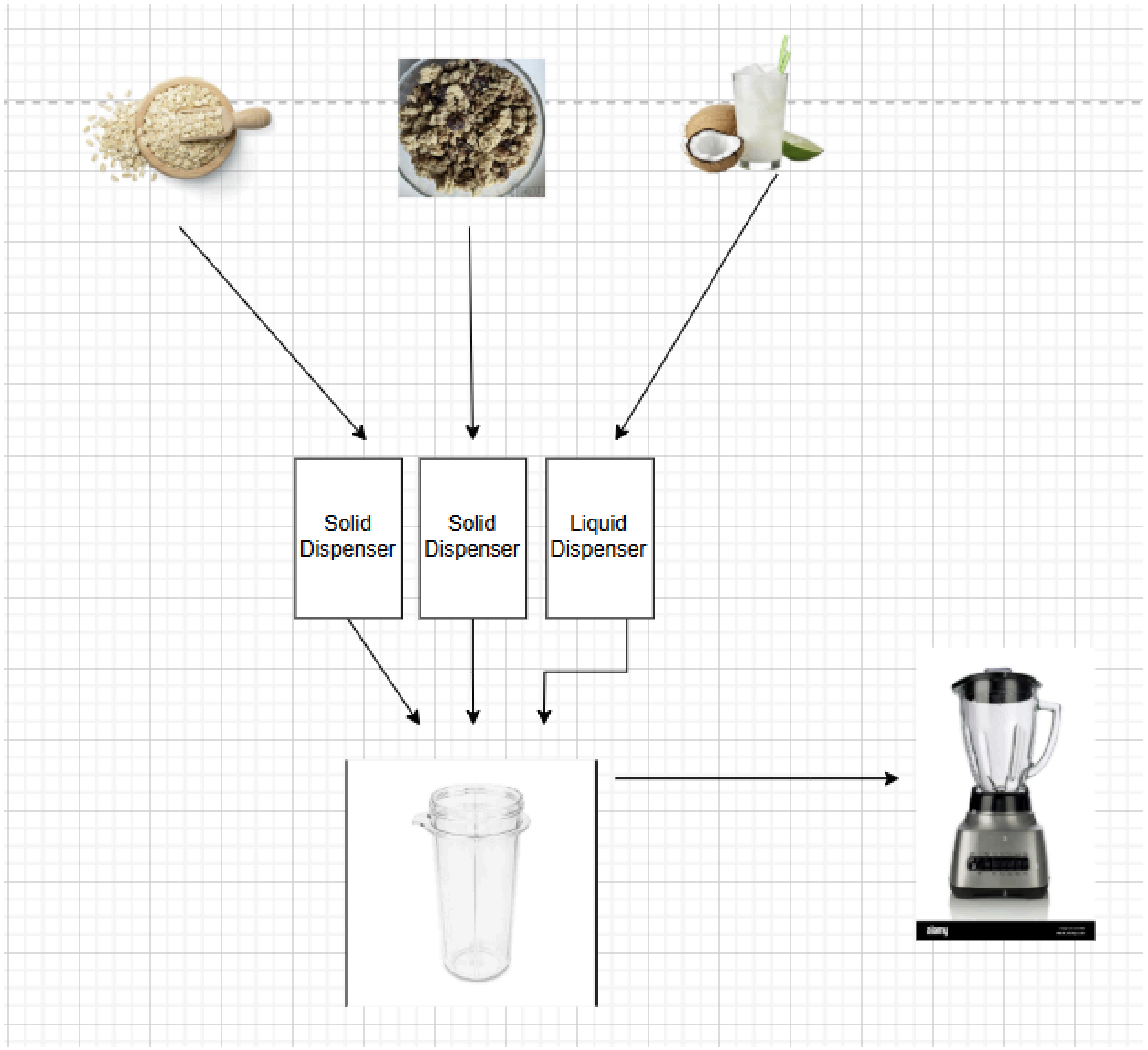
Making smoothies often requires measuring different ingredients with different measurement utilities. Liquid ingredients must be measured in mL, while solid ingredients are usually in units of cups or tablespoons. This can be a long process especially when the ingredients are numerous and are of various sizes, textures, and shapes.  Additionally, the measurement steps are repeated for every subsequent smoothie made. To make the process more efficient, we will automate measuring and dispensing of ingredients. This will simplify creating smoothies with different recipes and reduce repetitive steps when creating more than one smoothie. The products currently on the market that automate the process of making smoothies are either too simplistic or are not cost effective enough. The Ninja drink makers, although affordable,  turn liquid ingredients into frozen drinks but do not provide the capability to deal with solid ingredients. Other smoothie machines such as Albert's smoothie station and the Milk station company smoothie station are essentially vending machines that are not commercially available. These machines likely cost upwards of $3,000 to buy.  To address these issues, our system will be simple, versatile, and cost effective.

## 1.2 Solution

Our system will be able to make smoothies by following preset smoothie recipes or recipes the user creates. Our solution will be a three compartment structure with two solid compartments and one liquid compartment. The two solid compartments will have stepper motors to control the dispensing system while the liquid compartment will have a solenoid valve controlling its flow rate. The ingredients will be dispensed one at a time into a cup sitting on a load cell, which measures weight. The motors and solenoid valve are part of the Actuation Subsystem, and the load cell is part of the Sensors Subsystem. Complementing this, we will have software to control the motors, converting the load cell weight to tangible measurement quantities like cups, mL, and tablespoons. This is the Control Subsystem. To be able to create different kinds of smoothies, there will be a UI for inputting recipes and selecting preset recipes. This is the UI Subsystem. These subsystems will work together to automatically dispense the appropriate amounts of ingredients into the cup for different recipes, making the ingredient measuring and placing process more efficient. The user can then empty the cup's contents into a blender. Our dispensing subsystem can only deal with very small ingredients such as grains and oats and wet ingredients like fruits cannot be utilized. To get around this fact, we will allow users to deposit ingredients into the blender cup directly and the LCD display will, in real-time,  show the amount of each ingredient deposited.

We will use a microcontroller and the software we upload to it to control the motors and solenoid valve for proper dispensing. The microcontroller takes information from the load cell and the UI recipe to control the motors. The UI will have an LCD display and buttons to allow the user to select a preset recipe or input their own recipe.

## 1.3 Visual Aid

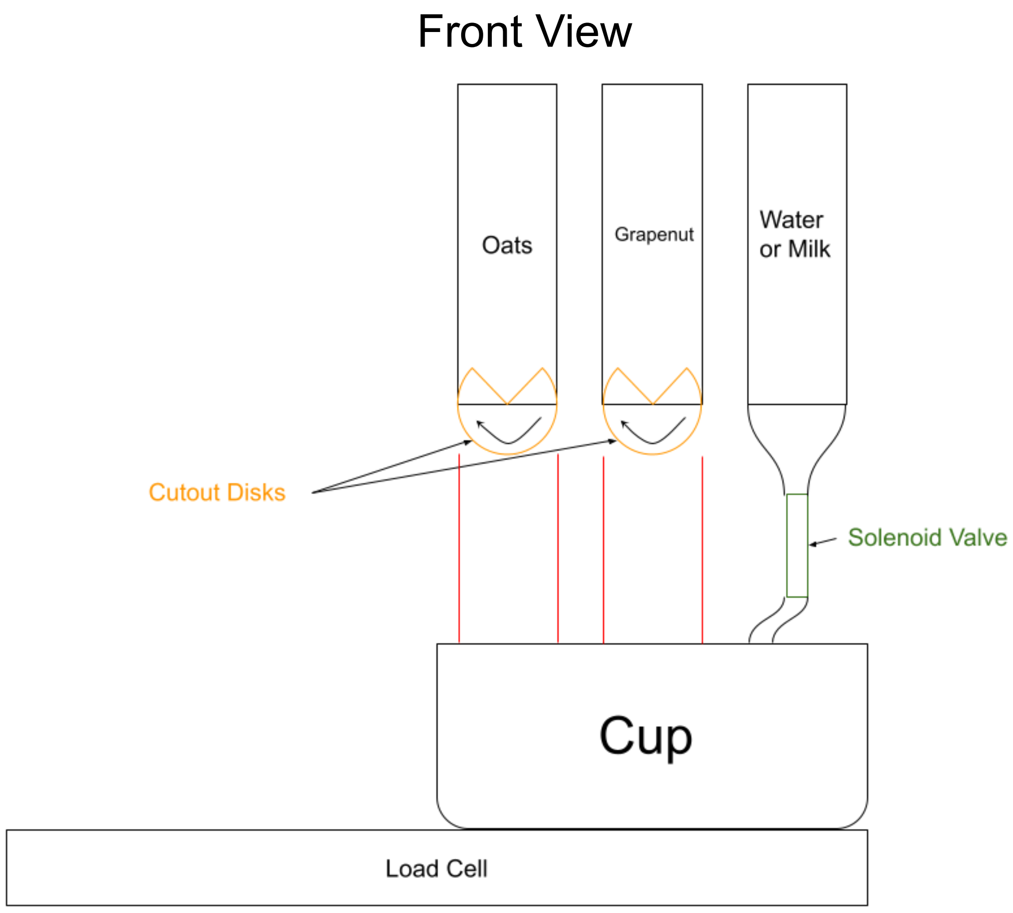


## 1.4 High-level Requirements

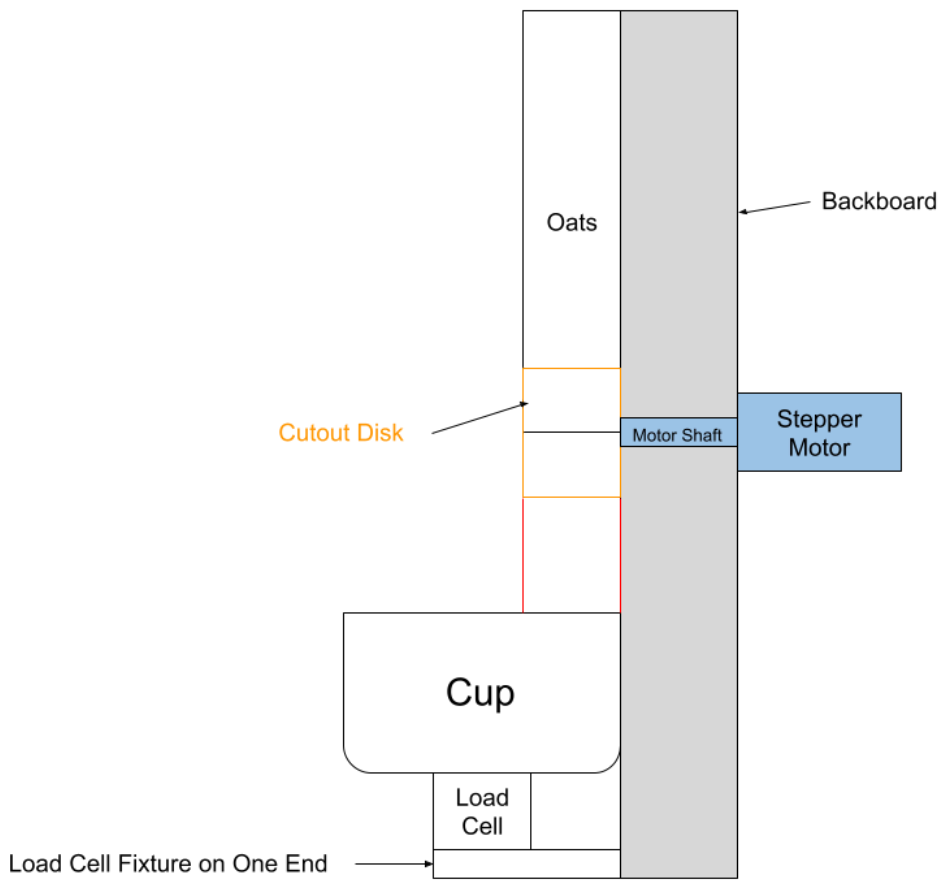
- The system takes a maximum of 1.5 minutes from the start of dispensing to all of the ingredients being inside the blender.
- Measurement accuracy will be to ±20% of intended ingredient amount.
- The system will dispense the correct ingredients given a preselected or user inputted recipe.

## 2. Design

### 2.1 Physical Design

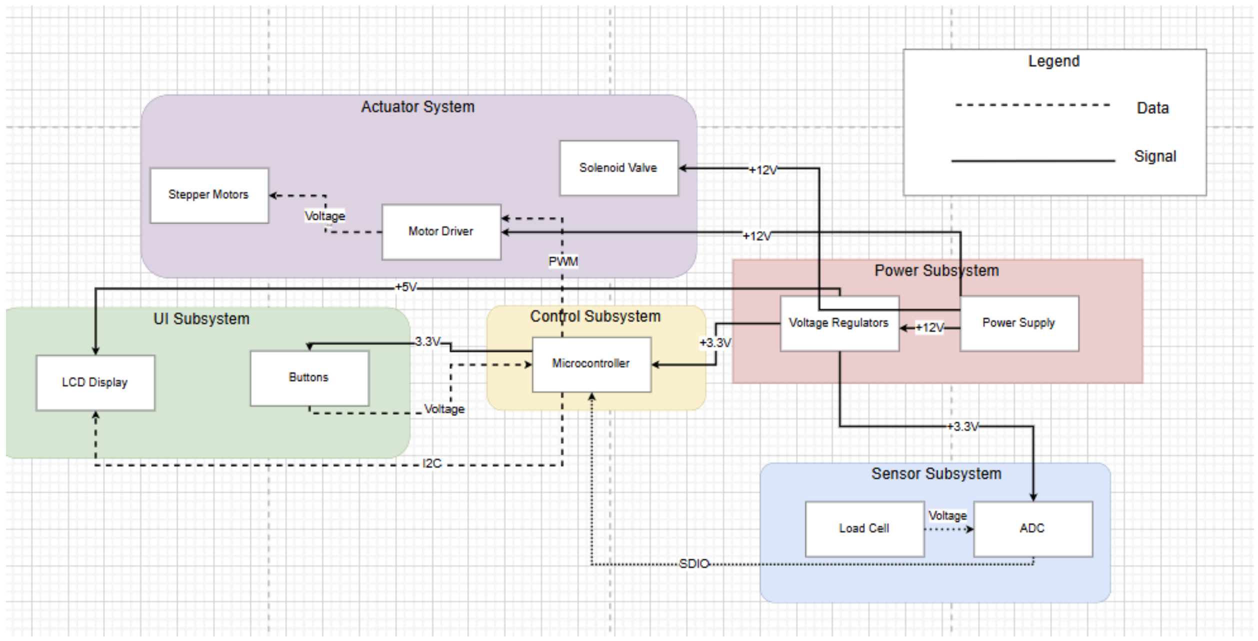


## Side View



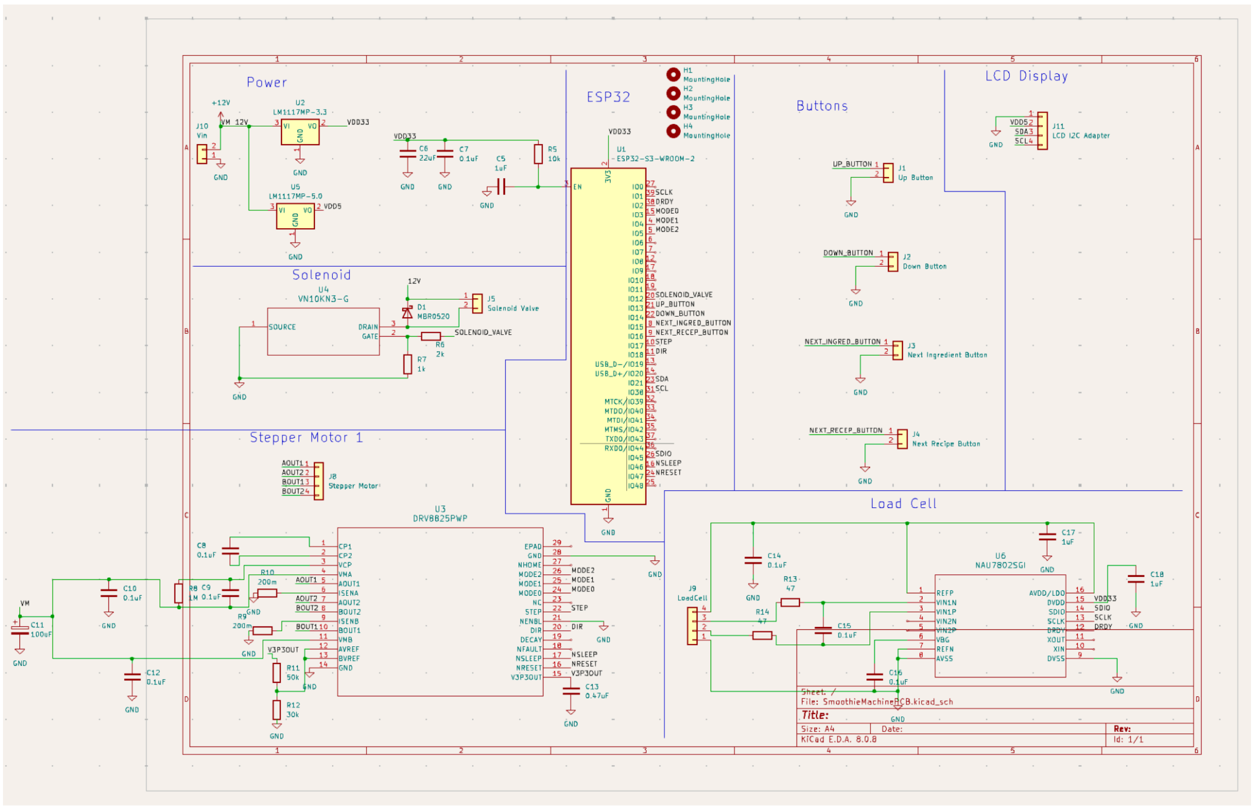
The physical design consists of the compartments mounted vertically in three columns, two of them for solid ingredients and one for a liquid ingredient. The stepper motors are for the solid ingredient compartments. They will be mounted horizontal, i.e. parallel with the ground, and rotate disks as big as the bottom of a compartment. Each disk will have a hole, where the solid ingredient can fall into. A hose will route the liquid ingredient into the solenoid valve and out of it to the cup. The cup sits on one end of the load cell, which is fixed to the structure on the other side.

## 2.2 Block Diagram



The critical subsystems are the Actuator, Sensor, Control, UI, and Power subsystems. The Actuator subsystem dispenses the solid and liquid ingredients from their compartments into the cup sitting on the load cell. The Sensor subsystem measures the mass of each ingredient added using the load cell. The UI subsystem uses buttons and an LCD Display to allow the user to make their own recipes. The Control subsystem uses the information from the UI and Sensor subsystems to execute the dispensing of ingredients for the correct recipe and in the correct amounts, by controlling the Actuator subsystem. The Power subsystem provides power to all components.

## 2.3 PCB Schematic



## 2.4 Subsystem Overview

### 2.4.1 Actuation

The Actuation Subsystem contains the stepper motors and solenoid valve that are used to dispense each ingredient into the cup. We will be using ROB-09238 or 17HS19-2004S1 stepper motors. The stepper motors will rotate a disk with a pocket. This pocket sits underneath a compartment filled with an ingredient. As the pocket faces upwards, the ingredient will fall into the pocket. When the motor rotates the pocket to face down, the ingredient in the pocket will fall into the cup. Multiple iterations of rotating the disk with the pocket will dispense the ingredient from the compartment into the cup in intervals of small amounts. We plan to use a DRV8825 motor driver that will allow us to precisely control the positioning of the motor, and thus, the where the pocket is. We will need to connect the ESP32's GPIO digital pins  to the DRV8825 driver, such as GPIO 4 to DIR pin of driver, and GPIO 5 to EN pin of driver. We will need to wire the PWM signal output of the microcontroller to the STEP input of the driver . These digital pins can also be used to change the mode of the stepper motor. There are three input pins on the driver corresponding to 8 binary modes. The modes are full step, half step, quarter step, 8 microsteps, 16 microsteps, and 3 32 microstep modes. This controls how much the motor will rotate in degrees based on the step angle. The step angle for our motor is 1.8 degrees, denoting how much the motor will turn for one full step. At the rising edge of the PWM step signal, the motor changes state to the next degree position. When the DIR pin of the driver is high, the motor turns clockwise and counterclockwise when the DIR pin is low.  The stepper motor we will be using is a 4 wire bi-polar motor having two coils. The two wires of one coil will be wired to AOUT1(+) and AOUT2(-)  on the driver and the wires of the second coil will be wired BOUT1(+) and BOUT2(-). The motor driver will be supplied with 12V, which will be used to power the stepper motor.  The driver also has a pin for specifying the decay mode. The decay is how fast the current is reduced in the coil of the motor when the driver switches the current off in the motor. We will be utilizing the mixed decay mode, which combines the smooth deceleration of slow decay with the fast deceleration using fast decay. Details about controlling the speed of the stepper motor will be provided in the Control subsystem section.

The solenoid valve is a DIGITEN DC 12V normally closed solenoid valve, a valve that only opens when supplied with power. It is mounted with the liquid ingredient feeding into it, and the output of the solenoid valve routed into the cup. The solenoid valve only requires one GPIO pin to switch it on and off. We plan to use GPIO 9 for this. By using software to trigger the GPIO pin of the microcontroller, we can turn the solenoid valve on and off, restricting or allowing the flow of liquid.  Additionally, there is a flyback diode in parallel with the solenoid valve to slowly dissipate energy built up in the magnetic field of the solenoid when it is powered. This is to prevent spurious voltage spikes that can damage other sensitive components.

To supply power to the solenoid valve, we are using the 12V supply from our power subsystem. The solenoid valve is in series with an N-MOS transistor with a turn-on voltage between 1.0 to 2.5V. When Vgs, the voltage between the gate and source of the transistor, is between 1.0 to 2.5V, there is a conducting path from the 12V supply through the solenoid to ground, and the valve opens. When the voltage Vgs is less than 1.0V, the transistor is off, and there is no conducting path from the 12V supply through the solenoid to ground, and the valve remains closed.  There is a voltage divider circuit at the gate of the transistor with an ESP32 GPIO input. In a voltage divider circuit with two resistances R1 and R2, the output voltage is $V_{out} = V_{in} R2/(R1 + R2)$. In our circuit,  the resistances are 2k and 1k Ohms respectively. The GPIO pin of the microcontroller can output 3.3V or 0V.  When $V_{in}$ = 3.3V, $V_{out}$ = (3.3)(1000)/(3000) = 1.1V, the transistor is on. When $V_{in}$ = 0V, $V_{out}$ = (0)(1000)/(3000) = 0V, the transistor is turned off.

| Requirements | Verifications |
|---|---|
| • Motors must be able to rotate the disk between 0 and 180 degrees with tolerance of 5 degrees. | • Stepper motor’s step angle of 1.8 degrees means it cannot be off by more than 3.6 degrees, so tolerance is satisfied. |
| • Motors can run at speeds from 60 rpm to 180 rpm, in order to control dispensing rate. | • Run motor for ten revolutions, and measure the time it takes, then convert to rpm |
| • The solenoid valve must be able to control flow rate of water and milk to a minimum of 1 cup per 20 seconds. | • Test the following with water and milk.<br>• Fill the container that feeds into the solenoid valve with liquid, with the solenoid valve closed.<br>• Collecting the solenoid valve output in a standard unit sized cup, open the solenoid valve and start a timer at the same time. |

| | |
|---|---|
| | ● End timer when cup is filled, checking that the timer value is under 20 seconds. |

**2.4.2 Sensors**

The Sensors Subsystem contains the load cell, which is used to dispense the correct amount of each ingredient. A typical smoothie size is 12-16 ounces [6], or 340 to 454 grams, and load cells should have a maximum capacity 25 to 30 percent greater than the load [7]. Therefore, to keep some buffer, including the mass of the cup holding the dispensed ingredients, our load cell has a maximum capacity of 1kg. The load cell's voltage output must be fed through an analog to digital converter to make it intelligible to the microcontroller, which is a digital system. The NAU7802SGI analog to digital converter gives 24 bits of precision, converting the load cell's voltage into a number between 0 and $2^{24} - 1$, which corresponds to the mass range of 0-1 kg. The NAU7802SGI takes 2.7 to 5.5V as supply power, so the output from the 3.3V voltage regulator will power it. The microcontroller will read the NAU7802SGI output by first checking the DRDY (data ready) output, and then reading registers 0x12, 0x13, and 0x14, which contain the 24 bit conversion result [8]. The load cell can be driven by 3.3V [9].

| Requirements | Verifications |
|---|---|
| ● Differential mass measurements, i.e. the measurement of the change in weight caused by adding some of an ingredient, must be accurate to +- 10 grams. | ● Start with the cup on the load cell, then add a 100 gram object.<br>● Check that the mass measurement, i.e. the result from the ADC converted to a mass value, is 100 grams more than just the cup, +- 10 grams.<br>● Repeat for 50, 150, 200, 250, and 300 grams to confirm the requirement for a range of ingredient mass additions. |
| ● The load cell must be able to make mass measurements up to 600 grams. | ● Place objects with masses known to be 100, 200, 300, 400, 500, and 600 grams on the load cell.<br>● Check that the mass measurement, i.e. the result from the ADC converted to a mass |

| | |
|---|---|
| | value, increments by 100 each time, confirming that the load cell can adequately make mass measurements through the range of masses. |
| ● Calibrate load cell so that absolute weight measurements are accurate to +- 10 grams | ● Use the voltage values from the ADC when zero grams are on the load cell and when close to the maximum number of grams are on the load cell, and figure out offset and gain needed for conversion from the voltage value to the absolute gram measurement, with this formula: $Mass = (voltage - offset) * gain$ <br>● Ensure that gram measurements of 100, 200, 300, 400, 500, and 600 gram objects are correct to within +- 10 grams, so our measurement range is correctly measured by the load cell. |

**2.4.3 Control**

The Control Subsystem contains the ESP32 microcontroller and interfaces with the UI and Actuation subsystems. The microcontroller receives inputs from the buttons and load-cell sensor and outputs signals to the DRV8825PWP motor driver  and LCD display.  The ESP32 contains two 12 bit ADC's that can be used in place of the NAU7802SGI for simplifying the system but receiving less precise measurement of the load. The software we use to program the microcontroller will be responsible for controlling the speed of the stepper motor through its PWM output pin.

$$f_{step}(\mu steps/second) = v(rotations/minute) \times 360(degrees/rotation) \times n_m(\mu steps/step)/60(seconds/min) \times (1/\theta_{step}(degrees/step))$$

To achieve our desired rotations/minute v,  we will adjust the frequency of the step signal and the microstepping level, which can be done by setting three mode pins with a combination of high and low signals. The $\theta_{step}$ or step angle  is provided by the datasheet and is 1.8 degrees/step. The LCD display is programmed through the I2C communication protocol. In the code we will have to find the I2C address. Then, we will use the LiquidCrystal_I2C Library to print characters to the lcd at certain positions on the lcd. The first two buttons are used to adjust the ingredient quantity amounts or select between preset recipes. The third button is used to move into preset recipe mode. The fourth button is used to select ingredient quantities for your own recipe.

Because the software of the ESP 32 manages the control subsystem, such as processing load cell data and button inputs, it uses the GPIO pins of the other subsystems and doesn't require any additional pins.

| Requirements | Verifications |
|---|---|
| ● Software must start and stop motors and solenoid to dispense ingredient amounts accurate to +- 10g | ● Take a preset recipe as an example and check the gram amounts of each ingredient<br>● Take note of the current weight of the load cell, and after an ingredient finishes dispensing, stop the software and look at the difference in weight. Check if this is +- 10g of expected weight in preset recipe |
| ● Software is able to change speed of stepper motor from 60 rpm to 180 rpm | ● In the software, set the speed of the motor to 60 rpm and count the time it takes to make 10 revolutions. Make sure this is close to 10 seconds<br>● In the software, set the speed of the motor to 180 rpm and count the time it takes to make 30 revolutions. Make sure this is close to 30 seconds |
| ● Software should debounce a button press so that aren't multiple button presses registered within half a second | ● Press button 3 to switch to preset recipes<br>● Press button 1 or 2 and make sure LCD display only changes once |
| ● Software should display recipe name on first row of LCD, the current ingredient compartment on the second row, the amount in grams on the third row along with its conversion to ounces/cups/tablespoons | ● Press button 3 and check whether recipe name is positioned on first row of LCD<br>● Press button 4 and check if ingredient compartment is positioned on second row of LCD<br>● After pressing button 4, press buttons 1 and 2 and check if ingredient amount shows up in grams, ounces, cups, and |

| | |
|---|---|
| | tablespoons on the third row of the LCD |

### 2.4.4 UI

The UI Subsystem contains the buttons and LCD display. The buttons are used to select between recipes and input a new recipe. To scroll between pre-set recipes and user recipes, the user can press button 3, and then press buttons 1 and 2 to scroll between recipes. Button 3 is pressed again to confirm the recipe that machine should make. An LCD will display the recipe name the user is currently looking at. To create a custom recipe, press button 4. The user can adjust ingredient quantities (in grams) for each ingredient using buttons 1 and 2 and press button 4 again to save and finalize the recipe. Finally, they can press and hold buttons 1 and 2 for faster quantity changes. Each press of buttons 1 and 2 decrements/increments the ingredient amount by 0.5g. The LCD display provides visual aid to the user during this process, displaying the current recipe and ingredient amounts in units common in smoothie recipes such as ounces/cups/tablespoons. The buttons are inputs to the Control Subsystem that each feed into the ESP 32. The buttons we are using are SPST Momentary N.O. Red Pushbutton, which we plan to solder onto a connector that we'll solder onto the PCB. The circuits for each button will connect to GPIO pins 13-16 as it supports pullup resistors that we can enable through software. We will be utilizing a 20x4 LCD display with I2C support. Because the ESP32 has I2C support, we can simplify the wiring to 4 pins. The connections will be SDA to GPIO 21, SCL to GPIO 22. The other 2 pins go to Vcc 5V and GND. The default state of the button will be high due to the pull-up resistor. Pressing the button connects the pin to GND. The ESP reads this change with digitalRead(pin) and displays the appropriate results.

There will be no external power for buttons since these are passive switches and don't need their own power supply. They simply connect or disconnect the GPIO pin to GND. The ESP32 provides 3.3V logic via its GPIO pins. The internal pull-up resistor draws a tiny current (microamps) from this 3.3V supply through the GPIO pin. Our pushbuttons are likely rated for higher voltages/currents (12V/50mA). At 3.3V and lower microamp currents, they're well within specs.

| Requirements | Verifications |
|---|---|
| ● Pressing button 3 without first pressing button 4 will move to the mode for selection between preset and user-made recipes, which will be done using buttons 1 and 2. The recipe names will be displayed on the LCD. | ● Press button 3 and check that first preset recipe shows up<br>● Press buttons 1 and 2 and see if different recipe names show up |
| ● When in ingredient mass adjustment mode, the LCD should display ingredient amounts in real-time when buttons 1 and 2 are pressed in increments of 0.5 units | ● Check that one press of button 2 increments ingredient amount by 0.5 units<br>● Check that one press of button 1 decrements ingredient amount by 0.5 units<br>● Holding down button 1 or 2 should continuously increment/decrement the ingredient amount and this should be clearly reflected on the LCD display |
| ● Pressing button 4 allows the user to create their own recipe. Buttons 1 and 2 allow the user to change ingredient amounts and button 3 is to to lock the amount into the recipe. The amount of each ingredient will be specified with the LCD showing which compartment's ingredient amount is currently being adjusted. | ● Check that pressing button 4 displays first ingredient compartment on LCD display<br>● Check that pressing buttons 1 and 2 change ingredient amounts and pressing button 3 switches to next ingredient compartment |
| ● Ingredient amounts should be shown on LCD display in grams along with its conversion to standard units in smoothie recipes: ounces, cups, tablespoons within some error tolerance(e.g 5%) | ● Check that the numbers shown on the LCD display for ounces, cups, and tablespoons are equivalent using dimensional analysis with known conversion factors |

### 2.4.5 Power

The Power Subsystem contains a 12V power supply and two voltage regulators, to convert the power supply voltage to 5 volts and 3.3 volts. The 12V supply is used to drive the DRV8825PWP motor driver, and the solenoid valve. The 3.3V regulator output is used to drive the ESP32, and through the

ESP32 drive the buttons. It also drives the load cell and the NAU7802SGI. The 5V regulator output is used to drive the LCD display.

| Requirements | Verifications |
|---|---|
| • Power supply should output a constant voltage of 12V +- 0.1 volts. | Use a multimeter and place positive lead at output of power supply and negative lead at ground. Look at voltage reading. |
| • The first regulator should output 5V +- 0.1 volts.<br>• The second regulator should output 3.3V +- 0.1 volts | Use a multimeter and place positive lead at output of the first regulator and negative lead at ground. Look at voltage reading.<br><br>Use a multimeter and place positive lead at output of second regulator and negative lead at ground. Look at voltage reading. |

## 2.5 Tolerance Analysis

An aspect of the design that poses a threat to the functionality of our smoothie maker is the selection of our motors, particularly the torque the motor has relating to the rpm of the motor. This affects how fast we can turn the disk to stop the dispensing of ingredients. If it is too slow we will overshoot our ingredient measurements and if it is too fast we will undershoot our ingredient measurements. We will be using a stepper motor so we must calculate its rpm.

A stepper motor has a command pulse rate which moves the motor in steps and not continuously.

$$\text{rev/second} = (\text{step/second})/(\text{step/revolution}).$$
$$\text{step/second} = \text{command pulse rate}$$
$$\text{step/revolution} = 360/\text{step angle}$$
$$60 * (\text{step/second})/(\text{step/revolution}) = \text{rpm}$$

We want to achieve a delay from  d = 1 to 3 seconds for the disk to turn and stop the flow of ingredients.

$$\frac{rpm}{60} = \frac{1}{d}$$

$$rpm = \frac{60}{d}$$

We want to choose a stepper motor and program it with the right command pulse rate and to achieve an rpm, depending on the delay we seek. The 17HS19-2004S1 is a bipolar stepper motor that has a step angle of 1.8 degrees. For a delay of 1 to 3 seconds we need our stepper motor to run in the range of 20 to 60 rpm. In order to run our stepper motor in the range from 20 rpm to 60 rpm, assuming a microstep level of 8 microsteps, we need to have the frequency of our step signal range from (20 * 360* 8)/(60 * 1.8) = 533.33 usteps/second to (60 * 360 * 8)/(60 * 1.8) = 1600 usteps/second. We used the equation detailed in the Control Subsystem section for this calculation. We don't want to make our rpm too high, otherwise the disk could fly off or jam.

## 3. Schedule and Cost

### 3.1 Schedule

| Week | Task | Members |
|---|---|---|
| 7 | Order Components | Max |
| 8 | Finalize PCB and breadboard design | Avyay, Anay |
| 9 | Spring Break - meet to discuss/prepare | All |

|  |  |  |
|---|---|---|
| 10 | Prototype the ingredient dispensing and UI | Max, Avyay |
| 11 | Finalize User UI and Measurement Conversion<br>Order PCB 2nd Wave | Max<br>Avyay |
| 12 | Finalize Machine Shop Design<br>Integrate Power with Core Sensors | All<br>Anay |
| 13 | Complete Weight, Motor, Dispensing Mechanisms | Anay, Avyay |
| 14 | Debug, Review, Order Final PCB if Needed | All |
| 15 | Final Demo, debug, review, work on paper | All |

## 3.2 Cost

### 3.2.1 Labor

Assuming $35/hr, an average of 15 hours spent per week, and 15 weeks of work, our total is:

$$\$35 \times 2.5 \times 15 \times 15 \times 3 = \boxed{\$59,062.5}$$

### 3.2.2 Parts

**Amazon:**

----------------------------------------------------------------------------------------------------------------------------------------------------

```
No    Catalog/Part #              Description                                   Units       Qty    Unit Price       Ext Price
```

--------------------------------------------------------------------------------------------------------------------------------------------------------

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | B016MP1HX0 | DC 12V 1/4" Quick connect Solenoid valve | each | 1 | $7.49 | $7.49 |
| 2 | B098KMB3DJ | adafruit 1kg strain gauge load cell | each | 1 | $3.95 | $3.95 |
| 3 | B01GPUMP9C | SunFounder IIC I2C TWI Serial 2004 20x4 LCD Module Shield Compatible with Arduino R3 MEGA (IIC 2004) | each | 1 | $12.59 | $12.59 |

-----------------------------------------------------------------------------------------------------------------------------------------------------

Total: $24.03

**PCBway:**

We plan on ordering 3 rounds of PCBs - totalling around $25 including assorted resistors, capacitors, etc.

### 3.2.3 Total

Factoring in labor and design, we estimate the total development cost to be $59,100.

# 4. Ethics and Safety

We will address the ethical and safety issues with reference to the IEEE and ACM Codes of Ethics and relevant regulatory standards [1].

## 4.1 Ethical Guidelines

Ethical guidelines require that we prioritize public safety and the well-being of users [1]. Because our project interacts directly with users, we want to provide full transparency of the capabilities of the machine. We aim to provide clear documentation, labeling intended use, limitations, and potential risks. An example is during an unlikely event when one of the ingredient chambers gets clogged, we'll let the user know on the LCD to try giving the machine a shake.

### 4.2 Safety Standards

We take user safety extremely seriously. Blenders are known to be dangerous and our design ensures that our product will not interfere with the blending process. Food safety is another concern. We aim to provide that through temperature sensors that let users know for example when the temperature of milk has exceeded a safe temperature for storage. Materials in contact with ingredients will be food-grade and easy to clean to prevent contamination. Finally, with the elimination of any batteries, we minimize the risk of fires and electrical hazards.

### 4.3 Regulations

We will adhere to campus safety and lab policies throughout the development of this project. Our team has set aside time for weekly reviews and feedback on our design throughout the project's lifecycle. With this, we will be able to collect valuable insight and feedback to ensure our project stays within relevant regulations and standards.

### 4.4 Respect and Compliance

Our team is committed to developing an environment of respect and ethical awareness. All members will treat all users fairly and hold each other accountable for any outcomes our project may have. Through this, we hope there will be efficient teamwork and good progress throughout the project lifecycle.

## 5. References

[1 ] IEEE-CS. “Code of ethics,” IEEE Computer Society, https://www.computer.org/education/code-of-ethics (accessed Feb. 12, 2025).

[2] “Oatmeal PNG transparent images,” PNG All -, https://www.pngall.com/oatmeal-png/ (accessed Mar. 6, 2025).

[3] A. satya, “Download strawberry slices on a transparent background, PNG for free,” Vecteezy, https://www.vecteezy.com/png/11198730-strawberry-slices-on-a-transparent-background-png (accessed Mar. 6, 2025).

[4] B. Foundation, “Home of the blender project - free and open 3D creation software,” blender.org, https://www.blender.org/ (accessed Mar. 6, 2025).

[5] K. Beck, “How to determine the RPM on Stepper Motors,” Sciencing, https://www.sciencing.com/determine-rpm-stepper-motors-10033323/ (accessed Mar. 6, 2025).

[6] Vitamix. “Making Large-Batch Smoothies at Home.” vitamix.com. Accessed: Mar. 5, 2025. [Online]. Available: https://www.vitamix.com/us/en_us/articles/making-large-batch-smoothies-at-home#:~:text=A%20typical%20smoothie%20size%20is,you're%20good%20to%20go!

[7] Laumas. “How to choose a load cell? The factors to assess.” laumas.com. Accessed: Mar. 5, 2025. [Online]. Available: https://www.laumas.com/en/blog/guides/how-to-choose-a-load-cell-the-factors-to-assess/#:~:text=As%20a%20rule%20it%20is,much%20as%20100%25%20or%20more.

[8] Nuvoton. *NAU7802 24-Bit Dual-Channel ADC For Bridge Sensors*, *Revision 1.7*. (2012). Accessed: Mar. 5, 2025. [Online]. Available: https://www.nuvoton.com/export/resource-files/NAU7802%20Data%20Sheet%20V1.7.pdf

[9] cs137, adafruit_support_mike, dlleigh. "Load Cell Sensitivity." forums.adafruit.com. Accessed: Mar. 5, 2025. [Online]. Available: https://forums.adafruit.com/viewtopic.php?t=191695