SMART SNACK DISPENSER

By

Eric Nieto Gonzalez

Adam Kramer

Elinor Simmons

Final Report for ECE 445, Senior Design, Spring 2025

TA: Surya Vasanth

7 May 2025

Project No. 23

Abstract

This report is a comprehensive overview of our project, the Smart Snack Dispenser. The Smart Snack Dispenser helps prevent unintentional overeating by automating portion control. Users can select portion sizes anywhere from 15 g to 70 g of Skittles, M&Ms, and peanuts. The machine also features nutrition tracking to promote more mindful and healthier eating habits. Our machine is made up of six different subsystems that include various components such as a microcontroller, touchscreen display, stepper and dc motors, and four unique sensors. This report will walk through our high-level requirements, subsystem overview, design process, results and verifications, and cost breakdown. The final section reflects on our project's successes, challenges, and potential future work.

Contents

1. Introduction

To help prevent mindless overeating, we propose the Smart Snack Dispenser. This is a machine that automates portion control and nutrition tracking. This report is an overview of our project development and is organized as follows:

- Chapter 1 – Introduction: This chapter describes our problem and solution, as well as our high-level requirements and subsystems.
- Chapter 2 – Design: This chapter walks through our design process and explains our design decisions.
- Chapter 3 – Verification: This chapter discusses our results and the verifications made to ensure our project was successful
- Chapter 4 – Costs and Schedule: This chapter provides a breakdown of our cost of parts and labor. It also includes a detailed schedule of the work done throughout the semester.
- Chapter 5 – Conclusion: This chapter summarizes are success, challenges, and possible future work. It also discusses the ethical considerations of our project.

1.1 Problem

The problem that we seek to address is that oftentimes while snacking, people lose track of how much they eat. This can lead to unwanted weight gain, unhealthy eating habits, and a poor relationship with food. While mindful eating can help address the consequences of uncontrolled snacking, practical devices engineered to help users adopt healthier and more mindful snacking habits are in need. Currently, the market lacks such devices, leaving people to rely on willpower alone to manually measure and track their calorie intake when it comes to snacking. Research shows that people make over 200 unconscious food-related decisions daily. Without a solution to automate the measuring and tracking process, it only becomes more likely that people will mindlessly snack, especially later in the day, due to decision fatigue. People snacking might be too hungry and impatient or not have the energy to manually track what they eat.

1.2 Solution

The solution that we propose is a Smart Snack Dispenser (SSD). This is designed to help people who snack make more mindful decisions about the amount that they eat. The secondary function of the SSD is to track the calories and macronutrients in the snacks that they have dispensed for later viewing. Through a combination of six main sensors, four motors, and one touchscreen display, which are all connected to the main microcontroller, the SSD is meant to function as a standalone kitchen appliance with its only wired connection being to a standard wall outlet. The SSD can dispense M&M's, peanuts, and Skittles, or any combination of the three.

1.3 High Level Requirements

For our project to be considered successful, we defined three high-level requirements: accuracy, speed, and usability.

1.3.1 Accuracy

Our first high-level requirement was that the machine must dispense the correct portion within a 15% tolerance or less. Since our main project focus is to help users control their portion sizes, it is important for us to make sure that the portions are weighed properly. Another requirement was that the machine must correctly keep track of the nutrients that are in each portion. These nutrients include calories, sugar, protein, sodium, and fat. For us to help users maintain a balanced diet, we must be accurate in tracking the nutrients that the user is consuming.

1.3.2 Speed

Our second high-level requirement was that the machine must dispense the snack within 30 seconds or less after the snack has been selected. We also wanted to ensure that user input was immediately registered as it is selected on our touchscreen interface. For this product to be beneficial to the user, we needed to make our machine operate faster than a person could do by hand. If a person could portion out and track the nutrients of their snack by themselves faster than our machine, then our machine would not be of much use.

1.3.3 Usability

Our final high-level requirement was that the user interface must be smooth and organized. It should be easy for a user to dispense a snack or find their personalized data. It is important for us to make the user interface intuitive to use for the consumer. If the machine is difficult or confusing to operate, the user is unlikely to use the product. We also wanted the user interface to be visually appealing to enhance the user experience and ensure that the interface is well-designed.

1.4 Subsystem Overview

In Figure 1, we can see our final block diagram. Our project is made up of six different subsystems. A brief description of each subsystem is provided below.

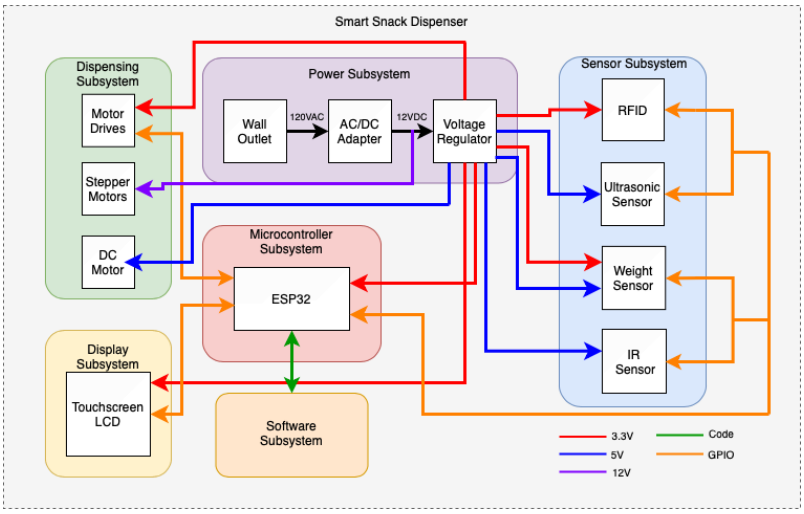

Figure 1: Block Diagram

1.4.1 Dispensing Subsystem

The dispensing subsystem includes our motor drivers, stepper motors, and dc motor. The stepper motors rotate our wheels to dispense our snacks, while the dc motor provides vibration to help with any jamming. Both motors are controlled through the motor drivers.

1.4.2 Microcontroller Subsystem

This subsystem acts as our central hub and interconnects our software to our sensor, dispensing, and display subsystems. We chose the ESP32 for its Wi-Fi and Bluetooth capabilities.

1.4.3 Sensor Subsystem

This subsystem includes RFID, an ultrasonic sensor, a weight sensor, and an IR sensor. The RFID allows for each user to have their own profile. The ultrasonic sensor checks to see if a bowl is in place before dispensing. The weight sensor weighs the dispensed amount. Finally, the IR sensor checks if the snacks need to be refilled.

1.4.4 Touchscreen LCD Subsystem

This is our display subsystem. It acts as our main interface for users to select and dispense their snacks. It also displays any information from our sensors through the ESP32.

1.4.5 Software Subsystem

This subsystem contains all of our internal features. This includes a calorie limit and nutrition tracking. The software subsystem also interconnects our microcontroller to the rest of the subsystems.

1.4.6 Power Subsystem

Our power subsystem includes a 12 V wall adapter, a 12 V to 5 V voltage regulator, and a 5 V to 3.3 V voltage regulator. All three of these voltages are used to power our various components such as our sensors, motors, and microcontroller.

2 Design

Our physical machine is shown in Figure 2 as a visual aid. Figure 3 showcases our final PCB design which includes the ESP32, a 12 V to 5 V voltage regulator outlined in red, and a 5 V to 3.3 V voltage regulator outlined in blue.

Figure 2: Physical Machine

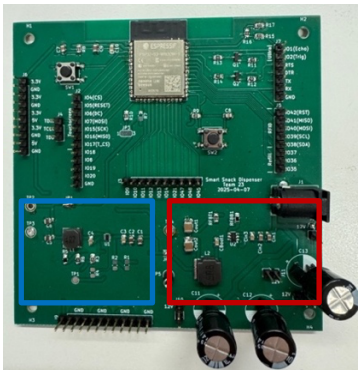

Figure 3: Main Custom PCB

2.1 Dispensing Subsystem

We considered many different designs for the physical dispensing of the snacks. We wanted to implement the simplest design while minimizing jamming and maximizing consistency and accuracy in the amount we dispensed across trials. We first considered a rack-and-pinion linear actuator design, where a gear would rotate and push a geared track attached to a plunger system forward, and push the snack contained in the plunger out into a chute emptying into a bowl. We also considered a classic electric linear actuator using the same mechanism of pushing the snack out of a container and into a chute emptying into a bowl. However, we decided that both mechanisms would be more difficult to design due to the complexity of the moving mechanical parts. We also considered an auger screw (also known as a screw conveyor), such as those used in agriculture to move dry bulk materials like grain, feed, and fertilizer. We considered this design due to its continuous screw motion, which we thought would be easy to implement with a stepper motor or a simple dc motor and encoder. However, after consulting with the machine shop, they advised against this design because in their experience this design has a high likelihood of jamming and creates lots of friction which could degrade the mechanism over time. Finally, we landed on a rotating wheel dispensing system, where underneath each snack a 3D printed wheel is placed. Each wheel has four cutouts separated by 90° that a small amount of a given snack falls into each time the stepper motor is triggered to rotate by 90°. When the wheel rotates by 180°, the snack falls into a cup beneath the machine. We chose the XY42STH34-0354A stepper motor, which operates at 12 V [1], along with the DRV8825 stepper motor driver [2]. We also applied a dc motor to induce physical shaking in case any of the snacks got jammed. Specifically, we utilized the ROB-11696, which operates at 5 V [3], and paired it with the TB6612 dc motor driver [4].

2.2 Microcontroller Subsystem

For our desired microcontroller, which acts as the heart of this system, we chose the ESP32 as our programmable chip. This was chosen because it can be coded using a common software called Arduino IDE. This chip also offers a wide range of GPIO pins and can use Bluetooth and Wi-Fi, which are necessary in our application. The ESP32 has thorough documentation online which allows us to clearly familiarize ourselves with all the options it has. Even though the ATMega328 was another reliable option, it does not have access to wireless features like the ones mentioned for the ESP32. Similarly, the STM32 also does not include built-in Wi-Fi. The ECE 445 website included an “ESP32 Example Board” (see [5]). This page contained a bare minimum circuit that was necessary for the ESP32 to function. This schematic included a programming circuit, which was needed to download our code onto the microcontroller. Since this schematic included all of our needs, we used this design for our PCB. The schematic that was used for our PCB is shown in Figure 4.

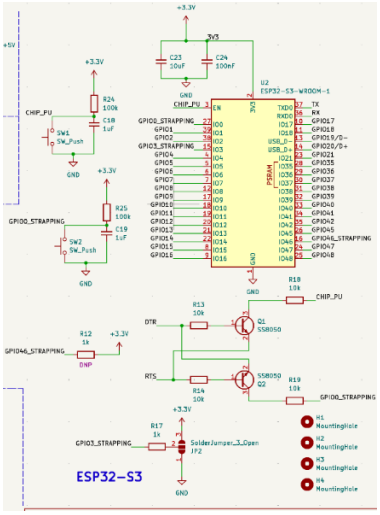

Figure 4: Schematic of Microcontroller Subsystem

2.3 Sensor Subsystem

2.3.1 RFID Sensor

To correctly identify each user that would be using the SSD, we chose to use an RFID sensor. This allows us to give tags to each participating user so that they can tap each time they wished to use the machine. This also works with those who have tags that are not registered to the dispenser, since they still fall under the same software. We chose the RC552 RFID which operates at 3.3 V, making it compatible with our ESP32 [6]. Alternate options were software-based accounts, where passwords and account information would have stored directly in the microcontroller. This way the user would tap their respective account on the touchscreen and use our coded keyboard to enter their unique password. This was not chosen as we wanted to save as much memory in our ESP32 as possible for other necessary functions.

2.3.2 Ultrasonic Sensor

We chose an ultrasonic sensor to identify the presence of a bowl. This prevents any prevented snack spillage and unnecessary waste of food. We specifically chose the HC-SR04, which operates at 5 V [7]. Since this sensor operates at 5 V, we had to add a small voltage divider circuit to make the component compatible with the ESP32. The schematic for this circuit is shown in Figure 5, where “Echo” is one of the signals from the ultrasonic sensor. We confirmed that the voltage was properly stepped down using Equation (1).

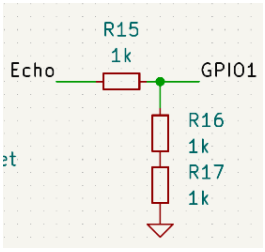

Figure 5: Voltage Divider for Ultrasonic Sensor

$$V_{out} = V_{in} * \frac{R16 + R17}{R15 + R16 + R17} = 5\ V * \frac{2000\Omega}{1000\Omega + 2000\Omega} = 3.33\ V \quad (1)$$

We had originally considered a PIR sensor, but this sensor only provided momentary signals. After motion was detected, the signal would no longer be active. The ultrasonic sensor provided us with a continuous signal, allowing us to constantly check if a bowl was in place.

2.3.3 Weight Sensor

Our weight sensor was made up of two components, the load cell and the load cell amplifier. We chose to use the TAL221 500 g load cell because we wanted a load cell big enough to allow reasonable portions and various bowl sizes, but small enough to read weight in our desired range [8]. Additionally, we used a HX711 load cell amplifier so that we could read data from our load cell. This amplifier operated at two voltages. One was 5 V, which was the excitation voltage, and the other was 3.3 V, which was the logic level. Both voltages were compatible with our system. This load cell amplifier was made into a custom PCB by using the schematic provided by SparkFun [9]. Our schematic and PCB are shown in Figure 6 and Figure 7. There were not many alternative options for this sensor. Since we needed to measure weight, it did not make sense to use anything other than a load cell. As for the HX711, this was the only product that we found that was compatible with a microcontroller. The HX711 was also particularly useful because it allowed us to easily tare the bowl by using the "HX711.h" library and the *scale.tare()* function.

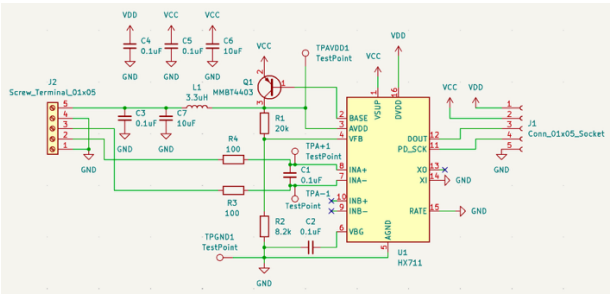

Figure 6: Load Cell Amplifier Schematic

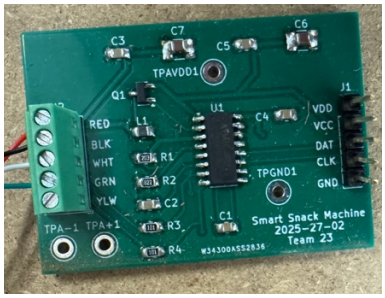

Figure 7: Load Cell Amplifier PCB

2.3.4 IR Sensor

The IR sensor allowed us to see if the snack containers needed to be refilled by checking to see if the beam was broken by snacks or not. This ensured that the snack never ran out while dispensing and would let the user know to restock before continuing. We specifically chose the 3 mm IR Break Beam from Adafruit [10]. We connected the transmitter to 5 V for better range, as recommended by Adafruit, and the receiver was connected to 3.3 V as this was our logic level. An alternative to this sensor could have been a photoresistor, but we believed that this sensor would have had more challenges with ambient light. In contrast, the IR sensor can handle a variety of light conditions, which makes it ideal since homes can vary in lighting.

2.4 Touchscreen LCD Subsystem

We chose the Hosyond MSP4022 4.0" touchscreen for our user interface [11]. This display operated at 3.3 V, making it easy to use with the ESP32. We wanted a touchscreen that would not take up too much

of our budget and would also be large enough for easy interaction. This touchscreen manufacturer used a variety of drivers. For our case, we used the SPI ST7796S Driver. This display also operated at 3.3 V, making it easy to connect with the ESP32. Alternate user interfaces screens could have been buttons alongside a serial LCD display. However, we did not pursue this option as it would have either been too bulky due to the number of buttons needed for all our functions, or too limiting, as fewer buttons would have restricted our selection options.

2.4 Software Subsystem

The finalized code consists of 1,500 lines which is composed of initialized libraries, variables, Arduino setup function, Arduino loop function, and 15 custom functions. The custom functions handle a variety of steps like setting up each sensor, a math function for all the needed calculations, multiple drawing functions to ensure the displays shows clean text, and much more. This was all done through the software Arduino IDE, which allows us to connect our computer straight into the ESP32 through a programmer. Calorie tracking was also implemented by allowing the user to set an upper limit for themselves. Once applied, the counter would subtract the number of calories in the weighed portion from the limit after each dispense. This would keep going until the counter hits a negative value which informs us that the user has finally reached their goal. Furthermore, Bluetooth and Wi-Fi functions were also created to call needed information outside of the ESP32 and to connect to the computer needed for external data storage. Therefore, the computer itself also had python code to ensure a connection was established between the ESP32 and the computer. This allowed for crucial information like data, time, user, snacks and the order they were chosen, the weighed amount, calories, fats, sodium, protein, and sugars to be transferred over.

2.6 Power Subsystem

Our power subsystem contained three main parts. This includes a 12 V wall adapter, a 12 V to 5 V voltage regulator, and a 5 V to 3.3 V voltage regulator. These were all voltages that we needed to power our different subsystems. Specifically, our stepper motors needed 12 V. The RFID, touchscreen display, motor drivers, and ESP32 needed 3.3 V. Our ultrasonic sensor and the dc motor needed 5 V. Lastly, the load cell amplifier needed both 5 V and 3.3 V.

To select our components, we needed an estimate for the current draw. After looking through datasheets we came up with an estimated output current of around 2.96 A. A breakdown of the current draw for each component can be seen in Table 1.

Table 1: Estimated Maximum Total Current Draw

Device	Maximum Current Draw
RFID (RC522)	30 mA
Ultrasonic Sensor (HC-SR04)	20 mA
Emitter (x3)	60 mA
Receiver (x3)	30 mA
Motor Driver (TB6612 or DRV8825)	~10 mA
Stepper Motor (XY42STH32-0354A)	0.7 A
DC Motor (ROB-11696)	0.8 A
Touchscreen Display (MSP4022)	~300 mA

HX711 Load Cell Amplifier with 500 g load cell	~10 mA
ESP32-S3-WROOM-1 (Wi-Fi + all GPIOs)	~ 1 A
	Total: ~2.96 A

Our final value of 2.96 A is over-estimated. This is because not all of these components will be operating at the same time. The reason we still used this number as our estimate was because we wanted to ensure that enough current would be provided.

2.6.1 12 V to 5 V Voltage Regulator

We chose the LMR51430 from Texas Instruments as our voltage regulator. This is a synchronous buck converter that is capable of outputting three amps of current. This is specifically a switching regulator. We chose a switching regulator over a linear voltage regulator due to the large voltage drop. For a linear voltage regulator when the voltage is dropped, that difference is converted as heat. If we had used a linear voltage regulator for a voltage drop of seven volts and a maximum current draw of three amps, this would have resulted in 21 W of power dissipated as heat (see Equation (2)). This is clearly too much power, which is why we used a switching regulator. In Figure 8, we can see a simplified version of our voltage regulator circuit. This was taken from the datasheet [12].

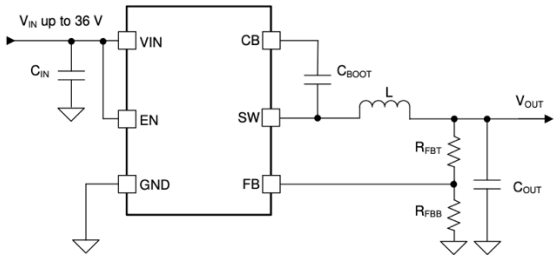

Figure 8: General LMR51430 Schematic

In this datasheet, there is a "Detailed Design Procedure" section. This section was a walkthrough of how to select the components for a 12 V input and 5 V ± 3 % output. This is the exact input and output voltage that we needed, so we used the same recommended components from the datasheet. The 3 % tolerance from this example is also how we chose this tolerance in our requirements for this voltage regulator. The schematic that we used for our PCB with all of the component values can be seen in Figure 9.

$$P_D = I_{out}(V_{in} - V_{out}) = 3\ A * (12\ V - 5\ V) = 21\ W \quad (2)$$

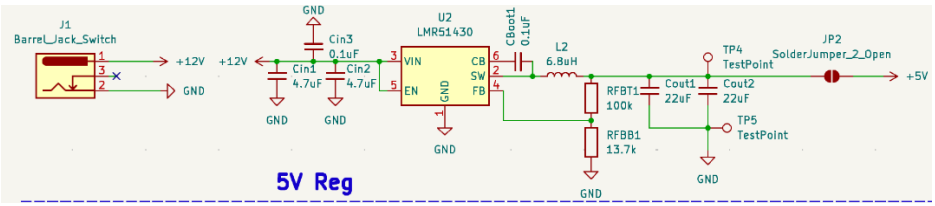

Figure 9: 12 V to 5 V Voltage Regulator Schematic

2.6.2 12 V to 5 V Voltage Regulator

We chose the TPS62933O from Texas Instruments for this voltage regulator. This is a synchronous buck converter that is capable of outputting three amps of current. This also a switching regulator. Our voltage drop is a smaller this time (1.7 V), which would give us a heat dissipation of 5.1 W if we had used a linear voltage regulator (see Equation (3)). Although this may have been manageable with a heat sink, we did not want to have to worry about overheating or using extra components like heat sinks, so we stuck with a switching regulator.

$$P_D = 3\ A * (5\ V - 3.3\ V) = 5.1\ W \tag{3}$$

In Figure 10, there is a general schematic for this voltage regulator. This was taken from the datasheet [13]. This datasheet also had a section called “Detailed Design Procedure.” This section contained a table that had recommended resistor, capacitor, and inductor values for a 3.3 V output. The ECE 445 website also had an “ESP32 Example Board.” This board used the same voltage regulator for a 5 V to 3.3 V conversion. We compared the values between the ECE 445 website and the datasheet and found that they were almost the same. The only difference was that the datasheet had a smaller C_{out} value and did not provide a C_{in} value. Since we had the same needs as the ECE 445 example board, we ended up using the same component values. We decided to use the larger output capacitance to help with the voltage ripple and we used the same C_{in} value since the datasheet did not provide a value, as previously mentioned. The detailed schematic that we used for the PCB is shown in Figure 11.

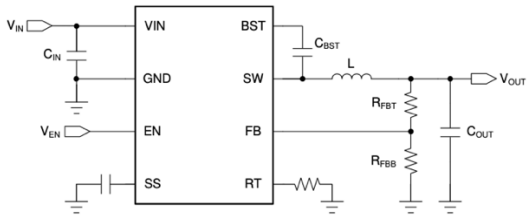

Figure 10: General TPS62933O Schematic

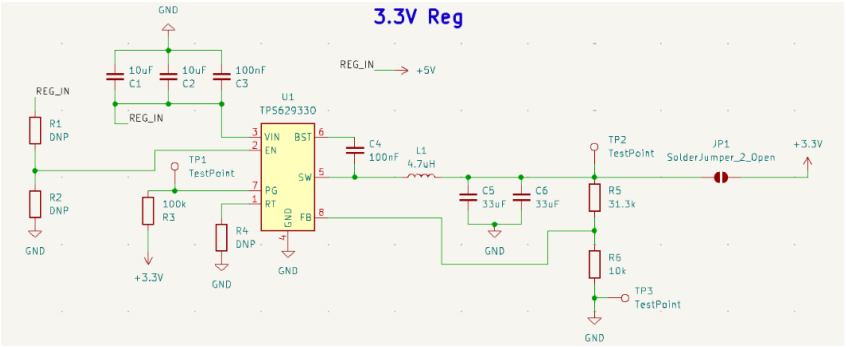

Figure 11: 5 V to 3.3 V Voltage Regulator Schematic

2.6.3 12 V Wall Adapter

We decided to use a wall adapter as our main source of power. We chose this over a battery because we wanted the machine to act as a household appliance. We also did not want the user to have to worry about replacing or recharging the batteries. Since 12 V was our largest needed voltage, we chose a 12 V wall adapter. Specifically, we chose the L6R36-120 from Tri-Mag [14]. This wall adapter is rated for 36 W. According to the 12 V to 5 V regulator datasheet, this regulator has around a 90 % efficiency. For a maximum output power of 15 W, this would require a 16.67 W input (see Equation (4)). The wall adapter datasheet stated that this adapter has around an 85 % efficiency. This means that for a 16.67 W output we would need a 19.61 W input (see Equation (5)). We made our rating a little larger than this (36 W) in case our efficiencies were lower than expected.

$$15\ W \div 0.9 = 16.67\ W \tag{4}$$

$$16.67\ W \div 0.85 = 19.61\ W \tag{5}$$

3. Verification

3.1 Dispensing Subsystem

The detailed requirements and verification table can be found in Appendix A Table 1.

To verify that our stepper motors meet the 30 second time requirement, we recorded several trials of dispensing our largest allowable amount (70 g). In our best-case scenario, which includes minimal jamming and most wheel slots being completely filled, we recorded a time of 23.62 seconds. However, our other trials included moderate to severe jamming. Our recorded times and the average time for these trials are shown in Table 2.

Table 2 Dispenses with jamming issues

37.35 seconds
30.30 seconds
31.40 seconds
31.37 seconds
30.17 seconds
32.37 seconds
31.12 seconds
Average: 32.01 seconds

Our average time with jamming was 32.01 seconds. Overall, we found that in ideal cases we were able to meet our high-level speed requirement. However, for cases with jamming, we were just above our requirement. We still consider this high-level requirement as met since jamming was not entirely in our control and is mainly an issue with physical design. We were able to qualitatively verify that the motor was spinning in the correct direction by 90 degrees each turn through observation.

3.2 Microcontroller Subsystem

The detailed requirements and verification table can be found in Appendix A Table 2.

We coded all of our software using Arduino IDE and programmed it onto the ESP32 using a USB to UART bridge and the programming circuit mentioned in Section 2.3. We used the ESP32 chip that was provided by the Electronics Services shop. This was the ESP32-S3-WROOM-1-N16, which has 16 MB of flash [15]. All of our code programmed onto the ESP32 successfully through the programming circuit on our main PCB, which confirms that our chip had enough storage. Since our project was fully functional, we were also able to verify through observation that our subsystems were communicating through the GPIOs on our microcontroller. This is also shown through our verifications of other subsystems. We can further verify this from Figure 12, where we can see that there is nothing physically connecting our machine and the computer.

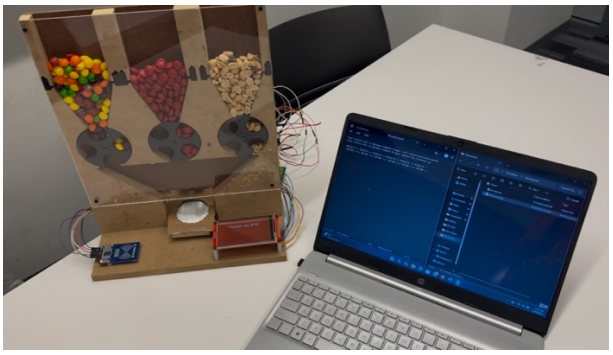

Figure 12: Machine and computer with no physical connection

3.3 Sensor Subsystem

The detailed requirements and verification table can be found in Appendix A Table 3.

3.3.1 RFID

For our tests, we had two distinct RFID tags. One was a white tag that was pre-registered as “Eric’s Tag” and a blue tag that was unregistered. The two tags are shown in Figure 13. In Figure 14, we can see that when the white tag is scanned the screen displays “Eric’s Tag.” In Figure 15, we can see that when the blue tag is scanned the screen displays “Unknown tag.” This confirms that each tag is being registered distinctly. Finally, each RFID tag has a unique ID, which has been verified to be genuine as shown at the top of the display. We know that this ID is correct, since it can only be changed with a machine that is accessible at the manufacturing plant.

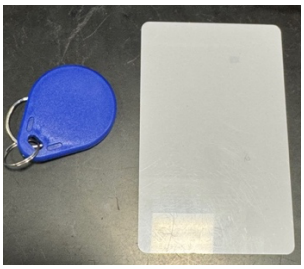

Figure 13: Two RFID Tags

Figure 14: Eric's tag on display

Figure 15: Unknown tag on display

3.3.2 Ultrasonic Sensor

In Figure 16, we can see that the distance reads 2,162 cm and to "Please place bowl." Our ultrasonic sensor faces forward. The reason that the distance is so large is because there is nothing placed, and the sensor is just reading out into the open. When a bowl is placed in front of the sensor, we can see in Figure 17 that the distance now reads 8 cm. Finally, when the bowl is placed the display now reads "Bowl placed" (see Figure 18). The sensor is coded so that it considers the bowl placed at 5 cm. Through these images, we can see that the sensor fully works as intended.

Figure 16: Ultrasonic sensor without bowl

Figure 17: Ultrasonic sensor with bowl closer

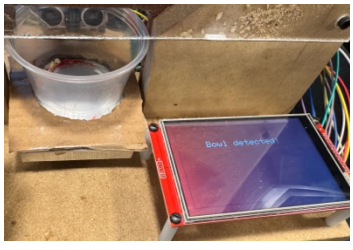

Figure 18: Bowl placed

3.3.3 Weight Sensor

To verify our 15% tolerance, we first compared the user-defined amount to the amount that was actually dispensed. Then, we also compared the weight of the dispensed amount to the weight read from a commercial food scale. We compared these weights using the Equation (6). Our measurement tables for each snack can be seen in Appendix B.

$$\% \text{ error } = \left| \frac{\text{desired } - \text{ measured}}{\text{desired}} \right| * 100 \tag{6}$$

Our tables show that our error percentages are under 15%, which verifies our 15% tolerance requirement. In Figure 19, we can see in the top left corner that a bowl has been placed, and the current weight reads -0.02 g. The reason that there is a negative is because Machine Shop placed our load cell upside down, so this negative can be ignored. Since our weight sensor is very sensitive, it can pick up readings from slight vibrations, so we could not get it to read exactly zero grams. However, the bowl weighed around four grams, so we consider the bowl to be successfully tared.

Figure 19: Bowl tare verification

3.3.4 IR Break Beam Sensor

In Figure 20, we can see that the peanuts fall below the IR sensors, allowing the receiver to sense the IR beam. In Figure 21 the screen says, “Refill Peanuts.” In Figure 22, the peanuts are filled above the IR sensors, preventing the receiver from sensing the IR beam. At the bottom of Figure 22, we can see that the refill notification is now gone. These series of pictures verify that our IR break beam sensor is working as intended.

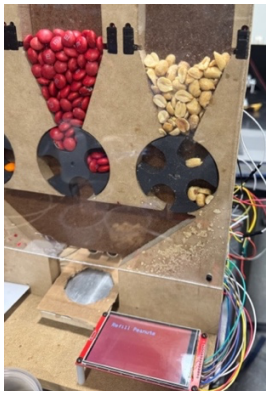

Figure 20: Peanuts need refilled

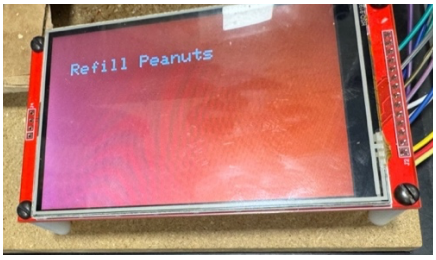

Figure 21: Refill Screen

Figure 22: Peanuts filled

3.4 Touchscreen LCD Subsystem

Figures 23 and Figure 24 show the display and database information. The display's information corresponds with the database's information. This verifies that the touchscreen displays the correct values. We were able to verify that the display was touch capable and only registered user input where the buttons were drawn through observations during our testing. This was further proven in our in-person demonstration of the machine. As proven in other sections such as Section 3.3.2 and Section 3.3.4, we found that the touchscreen would display the correct notifications when necessary.

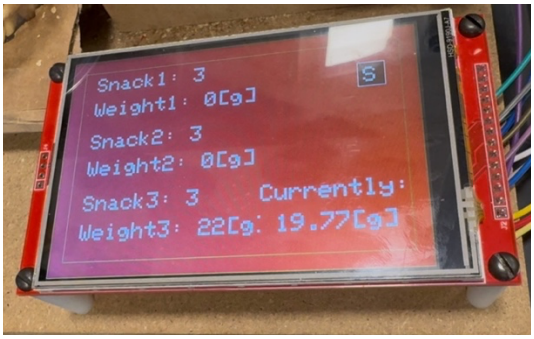

Figure 23: Touchscreen Display Information

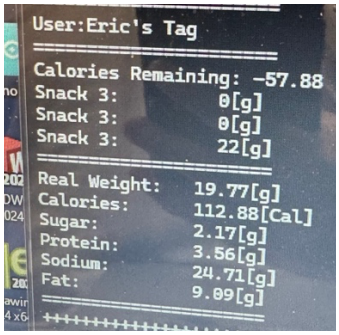

Figure 24: Database Information

3.5 Software Subsystem

The detailed requirements and verification table can be found in Appendix A Table 5.

Our first requirement was to ensure that the display only shows what the user has chosen to be on the screen. This refers to the multiple screens of information they can select from as the process is going rather than just being limited to one screen of information. The user was able to choose between snack information, current nutritional values, or chosen snacks. As shown in our in-person demonstration, this was verified observationally.

Our second requirement was to keep track and display the correct nutritional values. The values shown in Figure 25, Figure 26, and Figure 27 were found by taking the nutritional values from the labels on each snack and converting them to per gram using Equation (7). This was done so that we could track the nutrition per dispensed serving.

$$Nutrition\ per\ Gram = \frac{Nutrition\ Value}{Grams\ per\ Serving} \tag{7}$$

An example calculation for the calories in peanuts is shown in Equation (8).

$$\frac{160\ Calories\ per\ serving}{28\ g\ per\ serving} = 5.71 \tag{8}$$

```
Snack peanuts
calPerG3 = 5.71;
sugarPerG3 = 0.11;
proteinPerG3 = 0.18;
sodiumPerG3 = 1.25;
fatPerG3 = 0.46;
```

Figure 25: Peanuts Nutrition per gram

```
Snack M&Ms
calPerG2 = 5.12;
sugarPerG2 = 0.64;
proteinPerG2 = 0.07;
sodiumPerG2 = 0.71;
fatPerG2 = 0.21;
```

Figure 26: M&Ms Nutrition per gram

```
Snack Skittles
calPerG1 = 3.93;
sugarPerG1 = 0.75;
proteinPerG1 = 0.00;
sodiumPerG1 = 0.15;
fatPerG1 = 0.08;
```

Figure 27: Skittles Nutrition per gram

The nutrition values collected for a 19.77 g serving of peanuts is shown in Figure 24. An example calculation for how the calories were obtained is shown in Equation (9).

$$CalPerG3 * Real\ Weight = Calories \rightarrow (5.71) * (19.77) = 112.88\ [Cal] \quad (9)$$

The calories shown in Figure 24 correspond to the value that we obtained in Equation (9), verifying that the nutrition is being tracked correctly.

3.6 Power Subsystem

The detailed requirements and verification table can be found in Appendix A Table 6.

3.6.1 12 V Wall Adapter

In order to verify that our wall adapter was within our 5% tolerance, we first observed the output using a multimeter. In Table 3, we have five different measurements of the output voltage and the average of those values. Then we compared this average to our tolerance.

Table 3 12 V Wall Adapter Output Voltage

12.0355 V
12.0354 V
12.0353 V
12.0353 V
12.0353 V
Average: 12.03536 V

Since our average is larger than 12 V, we compared this value to the positive tolerance. Our calculation to get this value is shown in Equation (10).

$$Positive\ tolerance = 12\,V + 12\,V * 5\% = 12.6\,V \quad (10)$$

We can clearly see that 12.03536 V is less than 12.6 V, so our 12 V wall adapter meets our 5% tolerance requirement. We were also able to verify that when we connected this output to our stepper motors, they were able to operate properly. In our verifications, we mentioned using an oscilloscope to measure the percent error. We found that this was not necessary because the output voltage was so constant, we were unable to see any ripple in the waveform. We also mentioned checking the current and power in our verifications. We did not have access to a load that would provide us continuously with 3 A, so we were unable to check that the maximum output current was 3 A and the output power was 15 W. However, we found that this was not necessary because we were able to power our entire machine with just the wall outlet, meaning that the necessary current was being provided.

3.6.2 12 V to 5 V Voltage Regulator

We once again measured the output voltage using a multimeter. Our measurements and the average value are shown in Table 4.

Table 4 12 V to 5 V Regulator Output Voltage

4.95875 V
4.95842 V
4.95853 V
4.95856 V
4.95880 V
Average: 4.958612 V

For this voltage regulator, we had a 3% tolerance requirement and since our average is less than 5 V, we compared this value to the negative tolerance. Our calculation is shown in Equation (11).

$$Negative\ tolerance\ =\ 5\ V\ -\ 5\ V\ *\ 3\%\ =\ 4.85\ V \tag{11}$$

We can see that 4.958612 V is greater than 4.85 V, so our 12 V to 5 V regulator meets our 3% tolerance requirement. As we mentioned in our 12 V wall adapter discussion, we did not find using the oscilloscope necessary for the same reasons as above. Once again, we also did not find the 3 A maximum output current to be a necessary requirement. This was put in place because 3 A was the maximum possible output current from our voltage regulator and we were not sure what our exact output current would be. However, since our machine was able to fully operate, we know that our output current was 3 A or less.

3.6.3 5 V to 3.3 V Voltage Regulator

To verify our 5% tolerance for our 5 V to 3.3 V voltage regulator, we observed our output using the multimeter. Our measurements and the average value our shown in Table 5. Since our average value is greater than 3.3 V, we compared this value to the positive tolerance. Our calculation is shown in Equation (12).

Table 5 5 V to 3.3 V Regulator Output Voltage

3.34102 V
3.34086 V
3.34098 V
3.34092 V
3.34096 V
Average: 3.340948 V

$$Positive\ tolerance\ =\ 3.3\ V\ +\ 3.3\ V\ *\ 5\%\ =\ 3.465\ V \tag{12}$$

We can see that 3.340948 V is less than 3.465 V, so our 5 V to 3.3 V regulator meets our 3% tolerance requirement. For all of the same reasons that we mentioned in the 12 V wall adapter section and the 12

V to 5 V voltage regulator section, we did not find it necessary to use the oscilloscope or confirm that the maximum output current was 3 A.

4. Costs and Schedule

4.1 Part Costs

The table detailing our part costs is shown in Appendix C Table 1. The total actual cost of our project came out to be $152.43.

4.2 Labor Costs

The average starting salary of an Electrical Engineering graduate from the University of Illinois Urbana-Champaign is $87,769. There are 52 weeks in a year and 40 hours a week would give an hourly rate of $42.19/hour. We estimated that we each worked an average of 10 hours per week per person. We started working in week 4 and we finished in week 15. This is 12 total weeks, so this gives us a total of 120 hours per person. We used Equation (13) to calculate our total labor cost estimate.

$$ideal\ salary\ (hourly\ rate) * actual\ hours\ spent * 2.5 = \$42.19 * 120 * \ 2.5 = \$12{,}657 \qquad (13)$$

This adds up to $12,657 per person. Our total labor costs come out to $37,971. We estimate that Machine Shop worked for a total of eight hours at a rate of $40/hour. This comes out to $320. Overall, all of the labor costs at up to $38,291.

4.3 Schedule

The table detailing our schedule throughout the semester can be found in Appendix C Table 2.

6. Conclusion

6.1 Accomplishments

Throughout the verification chapter, we proved that the SSD was successfully completed and fulfilled all the requirements. We were able to implement all our desired sensors for their intended purposes. This was achieved by carefully selecting our components and developing the code needed to integrate them into our system. Another accomplishment was completing our high-level requirements. As we proved in Section 3.1, we were able to meet our time requirement in ideal scenarios. We considered this to be successful, since jamming was not entirely in our control. In Section 3.3.3 we also showed all of our weight measurements within our 15 % tolerance. Finally, the various figures in our verification chapter show the user interface as organized and visually appealing. In the end, our machine was able to operate as an independent appliance, with the only connection being through a wall outlet. Every subsystem—from sensor inputs to snack delivery—met our requirements, making it a cohesive, dependable smart snack appliance that is ready for real world use.

6.2 Uncertainties

A major uncertainty we had in our machine was the subject of anti-jamming. Unfortunately, there were scenarios where the snacks would get stuck within the stock area. This was typically caused by the snacks sticking together from the sugar and salt, or from the snacks getting stuck in the wheel after falling an odd way. This slowed down our dispensing process and made our timing requirement more difficult. This also caused our dispensing mechanism to be inconsistent. Depending on the level of jamming, the machine would dispense anywhere from one piece to the maximum pieces that could fit in the wheel. This affected not only our timing, but also our weight requirement. Anti-jamming can be improved by placing a dc motor in the center of each stock to attack the problem directly. This will significantly improve our solution considering that before we only had one dc motor in an arbitrary place. This was done due to Machine Shop suggesting this option. Furthermore, the code can be improved for the dc motors such that they activate when their weight has not changed if two rotations have passed. This is beneficial as it can provide input to the dc motors and allow them to activate when needed rather than every three turns.

Another topic that we are uncertain of is the geometric shape of our dispensing wheels, which were designed by Machine Shop. All three wheels were made identical. This was unideal because each snack weighed a different amount, so each wheel dispensed a different amount on each turn. For example, for a full wheel, Skittles would dispense around five grams per turn, while M&Ms would dispense around seven grams per turn. Since our wheels did not produce consistent amounts across the three snacks, the tolerance for each snack was different, making our weight requirements more difficult to achieve. A potential solution is to create multiple iterations of each wheel to find the most optimal wheel, allowing us to provide a more consistent amount.

6.3 Ethical considerations

6.3.1 Ethics

The SSD raises ethical concerns related to user privacy, data security, and autonomy. Since the system tracks individual snacking habits and nutritional data through RFID, it is essential to comply with ethical standards outlined by organizations such as the *IEEE Code of* Ethics (see [16]) and the *ACM Code of Ethics and Professional Conduct* (see [17]). *IEEE's Principle 1* emphasizes the need to prioritize the well-being and privacy of individuals, ensuring that users' personal information remains secure and is not shared without explicit consent. Strong encryption, secure authentication, and strict access control measures must be implemented to prevent unauthorized access to user data.

Furthermore, *ACM's Principle 2.9 (Ensure Fairness and Non-Discrimination)* emphasizes the responsibility of technology designers to create systems that are fair and accessible to all users. The SSD should be designed to prevent unintentional bias, ensuring that all individuals, regardless of their dietary restrictions, cultural preferences, or physical abilities, can use the system effectively. This includes offering a variety of snack options that cater to different nutritional needs, designing an intuitive interface that accommodates diverse users, and ensuring that the system does not unintentionally disadvantage any group. By integrating fairness into the design, the dispenser can serve as an inclusive and equitable tool for promoting healthier snacking habits.

6.3.2 Safety

Safety is a critical consideration in the design of the SSD, particularly regarding electrical and food hygiene risks. The *IEEE Code of Ethics Principle 1* states that engineers must "hold paramount the safety, health, and welfare of the public," which directly applies to preventing hazards associated with electrical components and food safety.

- Electrical Safety: Since the dispenser plugs into a wall outlet, it must comply with industry standards for insulation, grounding, and circuit protection to prevent electrical shocks, short circuits, and overheating. Overcurrent protection and temperature monitoring should also be included to mitigate fire hazards.
- Food Safety: The design must ensure that snacks remain uncontaminated. In accordance with *ACM's Principle 1.2 (Avoid Harm),* the dispenser should minimize food exposure to external contaminants. The use of food-grade materials, airtight containers, and easily cleanable surfaces will help maintain hygiene. Users should also receive clear maintenance and cleaning guidelines to prevent bacterial buildup or cross-contamination.

By adhering to IEEE and ACM ethical and safety standards, the SSD can be designed as a secure, user-friendly, and responsible solution that prioritizes privacy, accessibility, and well-being.

6.4 Future Work

There are multiple sections that we want to improve upon in the future. First, we want to expand our SSD into a 3D geometric shape to take up more volume. We want to pursue this as our current design limits us in storage. Concepts for this idea can be seen below in Figure 28. Pursuing this would allow us to increase snack storage, so that the user can refill less frequently and enjoy their snack for a longer period. Increasing overall space would also allow us to increase snack options to cover a wider range of nutrition. This would allow each user to mix and match more to their liking, while also reaching all of their goals.

Another aspect that needs more work is our anti-jamming system. The current solution to this problem is our dc motor with the uneven weight. This allows us to create vibrations in the machine to shake all the snacks down to a lower level. Unfortunately, this only proved to be effective for minimal jamming. More severe jams, caused by the snacks sticking together from the sugar and salt or from getting stuck in the wheel, had to be managed by hand. This also bring up the topic of cleaning procedures. In order to turn our project into a commercial product, we need to ensure that it can be cleaned properly. A proposed idea would be to create removable stock containers for easy-access cleaning. A small, thin brush would also be provided to reach harder places like the custom-made channels.

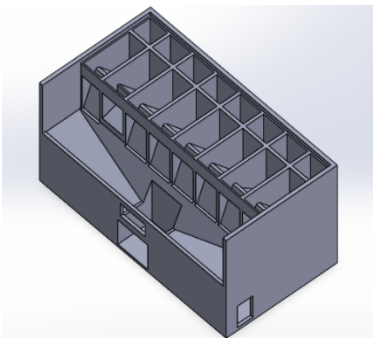

Figure 28: Future Design Concept

Finally, even though we successfully implemented Bluetooth data transfer to a desired computer, this is still limited by range as the computer needs to be within a certain distance for this to work. Our new solution for this is create a mobile app that can be accessed through Wi-Fi. This allows each user to have access to their collected data from their phone at any location in the world.

References

[1] “1.8°42MM High Torque Hybrid Stepping Motor,” (n.d.), adafruit.com. [Online]. Available: https://cdn-shop.adafruit.com/product-files/324/C140-A+datasheet.jpg, Accessed on May 6, 2025.

[2] “DRV8825 Stepper Motor Driver Carrier, High Current,” (n.d.), pololu.com. [Online]. Available: https://www.pololu.com/product/2133, Accessed on May 6, 2025.

[3] Hobby Motor - Gear,” 2016, sparkfun.com. [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/951/ROB-11696 Web.pdf, Accessed on May 6, 2025.

[4] “Adafruit TB6612 1.2A DC/Stepper Motor Driver Breakout Board,” 2022, adafruit.com. [Online]. Available: https://www.adafruit.com/product/2448, Accessed on May 6, 2025.

[5] “ESP32-S3-WROOM Example Board: Motor Controller,” 2025, courses.grainger.illinois.edu. [Online]. Available: https://courses.grainger.illinois.edu/ece445/wiki//esp32 example/index, Accessed on May 6, 2025.

[6] “CN0090,” (n.d.), digikey.com. [Online]. Available: https://www.digikey.com/en/products/detail/sunfounder/CN0090/18668629, Accessed on May 6, 2025.

[7] “HC-SR04 Ultrasonic Sonar Distance Sensor + 2 x 10K resistors,” 2018, adafruit.com. [Online]. Available: https://www.adafruit.com/product/3942, Accessed on May 6, 2025.

[8] “14728,” (n.d.), digikey.com. [Online]. Available: https://www.digikey.com/en/products/detail/sparkfun-electronics/SEN-14728/9555602, Accessed on May 6, 2025.

[9] “SparkFun HX711 Load Cell ,” 2019, sparkfun.com. [Online]. Available: https://cdn.sparkfun.com/assets/f/5/5/b/c/SparkFun HX711 Load Cell.pdf, Accessed on May 6, 2025.

[10] “IR Break Beam Sensors with Premium Wire Header Ends - 3mm LEDs,” 2021, adafruit.com. [Online]. Available: https://www.adafruit.com/product/2167, Accessed on May 6, 2025.

[11] 4.0 Inches 480x320 TFT Touch Screen LCD Display Module SPI ST7796S Driver for Arduino R3/Mega2560,” 2023, amazon.com. [Online]. Available: https://www.amazon.com/Hosyond-480x320-Display-ST7796S-Mega2560/dp/B0CKRJ81B5?th=1, Accessed on May 6, 2025.

[12] “LMR51430 SIMPLE SWITCHER® Power Converter 4.5-V to 36-V, 3-A, Synchronous Buck Converter in a SOT-23 Package,” 2022, ti.com. [Online]. Available: https://www.ti.com/lit/ds/symlink/lmr51430.pdf, Accessed on May 6, 2025.

[13] “TPS6293x 3.8-V to 30-V, 2-A, 3-A Synchronous Buck Converters in a SOT583 Package,” 2022, ti.com. [Online]. Available: https://www.ti.com/lit/ds/symlink/tps62933o.pdf, Accessed on May 6, 2025.

[14] “L6R36 Series 36W Wall Mount Power Supply,” (n.d.), tri-mag.com. [Online]. Available: https://www.tri-mag.com/wp-content/uploads/2021/07/L6R36 Series 1908a.pdf, Accessed on May 6, 2025.

[15]“ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U,” (n.d.), espressif.com. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1 wroom-1u datasheet en.pdf, Accessed on May 6, 2025.
[16]“IEEE Code of Ethics,” 2025, ieee.org. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html, Accessed on May 6, 2025.
[17]“ACM Code of Ethics and Professional Conduct,” 2025, acm.org. [Online]. Available: https://www.acm.org/code-of-ethics, Accessed on May 6, 2025.

Appendix A Requirement and Verification Table

Table 1 Dispensing Subsystem Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. Stepper motors must: a. Dispense snacks within 30 seconds or less b. Spin in the correct direction	1. Record the time of a dispense with a stopwatch. 2. See if the motor is spinning in the correct direction. 3. Accuracy wll be checked through our code and comparing real results with the datasheet provided.	Y
2. Wheels: a. Can produce the ideal number of pieces for each snack on average b. Minimize jamming	1. Create and test wheel designs to ensure the right amount is being grabbed per spin by running multiple dispenses. 2. Test the dispensing to see how often the dispenser will jam.	Y
3. DC motor must: a. Spin a small weight fast enough to provide the necessary vibration to avoid snack jamming	1. Run a dispense and check if the containers are vibrating.	Y

Table 2 Microcontroller Subsystem Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. All code will be stored within the ESP32 to ensure full independence.	1. Buy a chip that has enough local storage for the necessary programming.	Y
2. ESP32 must relay communication between subsystems when signals are sent through the GPIOs.	2. Run dispenses and verify that the subsystems are communicating properly.	Y

Table 3 Sensor Subsystem Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. RFID must: a. Read the tag b. Display the correct information associated with that tag	1. Check that the correct user is displayed when the tag is scanned. 2. The data collected will be compared with the datasheet of the RFID alongside the tags where the correct information can be read.	Y
2. Ultrasonic sensor must recognize when the distance is shortened due to the placement of a bowl.	1. Check the code output terminal to see if the sensor is reading the correct distance when a bowl is not present. 2. Place and bowl and check the output terminal to see if it reads a shorter distance.	Y
3. Weight sensor must: a. Measure the correct amount of weight within 15% b. Be able to tare the weight of the bowl	1. Compare the weight sensor readings with a commercial food scale. 2. Measure the weight of the bowl and then check to see if that weight is subtracted from the weight sensor at the end of dispensing.	Y
4. The receiver must: a. Be able to sense the IR beam from across the length of the container b. Not sense the IR beam when snacks are present, so a false notification is not sent	1. Place the IR sensors the correct distance away from each other and check the code terminal to verify that there is a reading. 2. Place an object between the two sensors and check that there is no reading from the receiver.	Y

Table 4 Touchscreen LCD Subsystem Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. The LCD correctly shows values collected from various sensors.	1. Verify that the values placed on the screen are correct and match the measured values. 2. Ensure that the value placement is within reason and is readable.	Y
2. The display should be capable of user touch to be able to select a variety of options.	1. Write the necessary code that is needed to make the touchscreen function. 2. Verify that touch location is within reason and does not register at a different part of the screen.	Y
3. The LCD should display the correct notifications when necessary.	1. Perform scenarios where notifications are necessary, such as not placing a bowl or the containers need refilled and ensure that these notifications are displayed on the screen.	Y

Table 5 Software Subsystem Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. The machine should only display the features that the user has chosen.	1. Run through the dispensing process and ensure that the correct features are enabled and implemented.	Y
2. The machine must keep track of the correct nutrition values.	1. Compare the displayed nutrition per portion with our own values calculated by hand. 2. Compare the daily total with our own values calculated by hand.	Y

Table 6 Power Subsystem Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. The wall adapter must be able to provide 12 V $\pm$ 5% and a maximum output current of 3 A.	1. Check the output voltage with a multimeter and observe the output with the oscilloscope to measure the percent error. 2. Connect the stepper motors to ensure that they are provided the necessary voltage to operate. 3. Connect to the 12 V to 5 V regulator and measure the output power to ensure it is receiving the necessary amount of power to output 15 W.	Y
2. The 12 V to 5 V voltage regulator must be able to provide 5 V $\pm$ 3% and a maximum output current of 3 A.	1. Check the output voltage and current with a multimeter before and after connecting the voltage regulator to the rest of the circuit. 2. Check the output voltage with the oscilloscope to measure the percent error.	Y
3. The 5 V to 3.3 V voltage regulator must be able to provide 3.3 V $\pm$ 5% and a maximum output current of 3 A.	1. Connect to the 5 V to 3.3 V regulator once the 12 V to 5 V regulator once it has been verified and check the output voltage and current with a multimeter. 2. Check the output voltage with the oscilloscope to measure the percent error. 3. Connect to the rest of the circuit and operate the machine to ensure all the other components are receiving their necessary power.	Y

Appendix B Weight Sensor Measurement Tables

Table 1 M&M Weight Measurements

Entered Portion	Weighed Portion	% Error between Entered and Weighed Portion	Commercial Scale Weight	% Error between Weighed Portion and Commercial Scale Weight
15 g	13.67 g	8.87 %	12 g	13.92 %
22 g	20.61 g	6.32 %	21 g	1.86 %
70 g	63.26 g	9.63 %	60 g	5.43 %

Table 2 Skittles Weight Measurements

Entered Portion	Weighed Portion	% Error between Entered and Weighed Portion	Commercial Scale Weight	% Error between Weighed Portion and Commercial Scale Weight
15 g	16.92 g	12.8 %	17 g	0.471 %
22 g	21.81 g	0.864 %	19 g	14.79 %
70 g	61.34 g	12.37 %	57 g	7.61 %

Table 3 Peanuts Weight Measurements

Entered Portion	Weighed Portion	% Error between Entered and Weighed Portion
15 g	15.04 g	0.267 %
22 g	21.5 g	2.27 %

Appendix C Part Costs and Schedule

Table 1 Part Costs

Part	Manufacturer	Retailer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
ESP32-S3-WROOM-1-N16	Espressif Systems	Digi-Key	$5.92	$3.79590	$0
Ultrasonic Sensor (HC-SR04)	OSEPP Electronics	Adafruit	$3.95	$3.16	$0
RFID Tags & Chip (RC522)	SunFounder	Digi-Key	$5.60	$5.60	$0
Touchscreen Display (MSP4022)	Hosyond	Amazon	$18.99	$18.99	$18.99
Load Cell (TAL221)	SparkFun	Digi-Key	$17.89	$12.50	$12.50
Motor Driver x 3 (DRV8825)	Polulu	Polulu	3 x $14.95	3 x $12.65	3 x $14.95
Motor Driver (TB6612)	Adafruit	ECE Supply Center	$10.68	$6.26	$6.26
Stepper Motor x 3 (XY42STH32-0354A)	SparkFun	ECE Supply Center	3 x $25.50	3 x $14.95	3 x $14.95
DC Motor (ROB-11696)	SparkFun	ECE Supply Center	$4.27	$2.50	$2.50
Wall Adapter (L6R36-120)	Tri-Mag	Digi-Key	$10.36	$8.78	$8.78
Voltage Regulator (LM51430)	Texas Instruments	Digi-Key	$1.29	$0.59912	$1.17
Voltage Regulator (TPS62993O)	Texas Instruments	Digi-Key	$1.04	$0.46482	$0.94
Terminal Block Connector (1725685)	Phoenix Contact	Digi-Key	$4.33	$2.82	$2.82
6.8 uH Inductor (SRN8040TA-6R8M)	Bourns Inc.	Digi-Key	$0.58	$0.25750	$0.53
4.7 uH Inductor (HPC 6028NF-4R7M)	TAI-TECH Advanced Electronics	Digi-Key	$0.16	$0.06250	$0.15
NPN Transistor x 2 (SS8050-G)	Compchip Technology	Digi-Key	2 x $0.24	2 x $0.02750	$0
IR Sensor (3 mm IR Break Beam)	Adafruit	Adafruit	$2.95	$2.36	$2.95
HX711 IC Chip	Avia Semicon	LCSC Electronics	$0.45	$0.2493	$0.45
PNP Transistor (MMBT4403LT1G)	onsemi	Digi-Key	$0.13	$0.01252	$0.12
Barrel Jack (PJ-102AH)	Same Sky	Digi-Key	$0.76	$0.43611	$0.69

0.1 uF Capacitor x 10 (CL21B104KBCNNNC)	Samsung Electro-Mechanics	Digi-Key	10 x $0.08	10 x $0.00496	10 x $0.009
22 uF Capacitor x 2 (CL32B226KAJNNNE)	Samsung Electro-Mechanics	Digi-Key	10 x $0.56	10 x $0.13	10 x $0.312
32 kΩ Resistor (RT0805BRD0732KL)	Yageo	Digi-Key	$0.10	$0.04221	$0.09
13.7 kΩ Resistor (ERA-6AEB1372V)	Panasonic Electronic Components	Digi-Key	$0.10	$0.03276	$0.06
100 Ω Resistor x 2 (ERA-6AEB101V)	Panasonic Electronic Components	Digi-Key	2 x $0.10	2 x $0.03276	2 x $0.09
20 kΩ Resistor (ERA-6AEB203V)	Panasonic Electronic Components	Digi-Key	$0.10	$0.03276	$0.09
8.2 kΩ Resistor (ERA-6AEB822V)	Panasonic Electronic Components	Digi-Key	$0.10	$0.03276	$0.09
3.3 uH Inductor (LQM21PN3R3MGRD)	Murata Electronics	Digi-Key	$0.17	$0.06375	$0.16
1 kΩ Resistor x 4	N/A	Electronic Services Shop	N/A	N/A	$0
0.1 uF Capacitors x 10	N/A	Electronic Services Shop	N/A	N/A	$0
1uF Capacitor x 2	N/A	Electronic Services Shop	N/A	N/A	$0
10uF Capacitor x 5	N/A	Electronic Services Shop	N/A	N/A	$0
33 uF Capacitor x 2	N/A	Electronic Services Shop	N/A	N/A	$0
Test Points x 7	N/A	Electronic Services Shop	N/A	N/A	$0
10 kΩ Resistor x 5	N/A	Electronic Services Shop	N/A	N/A	$0
100 kΩ Resistor x 4	N/A	Electronic Services Shop	N/A	N/A	$0
4.7 uF Capacitor x 2	N/A	Electronic Services Shop	N/A	N/A	$0
Total			**$218.35**	**$157.92**	**$152.43**

Table 2 Schedule

Timeline	Task	Description	Team Member
Week 4	Prepare for proposal	Refine subsystems select main components. Come up with physical mechanism and speak with Machine Shop. Write proposal document.	Everyone
Week 5	Order parts	Order main parts needed for breadboard demo.	Elinor
Week 5	Prepare for breadboard demo	Begin building the prototype for the breadboard demo. Continue selecting components.	Everyone
Week 6	Prepare for 1st order PCB	Start designing the first order PCBs for PCB review.	Elinor
Week 6	Continue breadboard demo preparation	Continue wiring and coding for breadboard demo.	Eric
Week 6	Select motors	Research and buy motors for dispensing mechanism. Buy motor driver for breadboard demo.	Adam
Week 7	Submit 1st order PCBs	Submitted two PCB prototypes.	Elinor
Week 7	Continue breadboard demo preparation	Continue wiring and coding for breadboard demo.	Eric
Week 7	Prepare for Design Document	Assign parts and write design document.	Everyone
Week 7	Flowchart	Organize flow of operation for machine.	Adam
Week 8	Breadboard Demo	Display progress during breadboard demo	Everyone
Week 8	Submit 2st order PCBs	Adjust 1st order PCBs and submit for second order. Prepare new PCBs.	Elinor and Adam
Week 8	Order parts	Order parts needed for 1st order PCBs.	Elinor
Week 10	Develop software	Continue working on graphical user interface code.	Eric
Week 10	Solder and test	Solder and test 1st order PCBs.	Elinor
Week 10	Prepare for 3rd order PCBs	Prepare and adjust PCBs for 3rd order. Order more parts for testing.	Elinor and Adam
Week 11	Submit 3rd order PCBs	Submitted adjusted PCBs.	Elinor
Week 11	Download code to microcontroller	Attempt to program code onto custom PCB that has the ESP32. Debug.	Everyone
Week 11	Continue software development	Continue developing code for user interface and interconnecting components.	Eric
Week 11	Get code working with custom PCB	Wire components to custom PCB and attempt to get working. Debug.	Eric and Elinor
Week 11	Prepare for 4th order PCBs	Finalize and prepare PCBs for 4th order.	Elinor and Adam
Week 12	Submit 4th order PCBs	Submitted finalized PCBs.	Elinor
Week 12	Get all components working with custom PCBs	Debugged and got everything working with the custom PCBs.	Eric and Elinor

Week 12	Continue software development	Continue developing code for user interface and interconnecting components.	Eric
Week 12	Test 3rd order PCB	Soldered and tested 3rd order PCB.	Elinor
Week 13	Continue software development and physical testing	Continue developing code and testing dispensing mechanism.	Eric and Adam
Week 14	Test 4th order PCB	Solder and test 4th order PCB.	Elinor
Week 14	Prepare for Mock Demo	Connect everything to final PCB and prepare as much as possible for mock demo.	Everyone
Week 15	Prepare for Final Demo	Finalize project and fix any issues for final demo. Take final measurements and data.	Everyone
Week 15	Prepare for Mock/Final Presentation	Prepare slides for mock/final presentation.	Everyone
Week 16	Final Presentation	Finalize slides and practice for final presentation.	Everyone
Week 16	Final Paper	Write final paper.	Everyone