CO_2ffee: Coffee Bean Freshness Tracker

By

Abrar Murtaza
Joshua Meier
Nathan Colunga

Final Report for ECE 445, Senior Design, Spring 2025

TA: Surya Vasanth

07 May 2025

Project No. 4

Abstract

For our Senior Design Project, we developed a coffee appliance that tracks bean freshness by monitoring carbon dioxide (CO_2) released over time. The system integrates multiple sensors and a data logging module to measure and record CO_2 levels, providing users with quantifiable insight into the beans' freshness state. This report outlines the design requirements, subsystem implementations, and verification results, demonstrating that the system met all functional specifications and performed as intended.

Table Of Contents

1. Introduction

1.1 Problem

Many coffee enthusiasts care deeply about achieving the ideal level of bean freshness, as it significantly affects both flavor and overall quality. Beans roasted just one day earlier may still be too fresh, while after a short period, they can become stale. When purchasing freshly roasted beans from a roastery, customers are typically given only an estimated usage window (often within a month), without precise guidance on when the beans reach peak freshness. As a result, those who are particular about coffee quality lack a reliable way to determine when their beans are optimally fresh. Beans that are too new can result in overly acidic coffee [2], while older beans may lose the distinct flavor notes that define the roast, leading to a flat or stale brew [2]. Without a way to gauge freshness, users risk suboptimal extraction and inconsistent coffee quality [1][3]. To address this issue, we developed a custom coffee container capable of detecting the freshness of beans based on measurable factors. This allows users to track freshness over time and brew their coffee when the beans are at their ideal state, ensuring a more consistent and high-quality cup.

1.2 Solution

For our project, we created a container designed to track the amount of CO_2 remaining in coffee beans, as this correlates directly with their freshness [1]. This measurement is based on the weight of beans added, as well as the detected concentration of CO_2 that accumulates in the container over time. Our system consists of an inner and outer container. The inner container is a standard roastery coffee bag that holds the beans and helps preserve them for as long as possible, utilizing an airtight seal combined with a degassing valve. The combination of the seal and valve

ensures that no oxygen enters the container with the beans, while allowing CO_2 to escape from the inner container to maintain consistent pressure. The outer container houses all electronic components, including the weight sensor (which measures the weight of the beans and accounts for the inner container), the CO_2 sensor, the motor for opening the outer container, and other required components, all of which are enclosed in a dedicated section.

For operation, when the user wants to add a newly roasted bag of beans, they press a button to open the outer container and place the coffee bag at the center of the weight sensor located at the bottom. They then use a mobile interface to indicate that new beans have been added, along with selecting the roast type (light, medium, or dark). The user presses the button again to close the lid with the beans inside. The container then measures the weight of the beans and the initial CO_2 concentration in the outer container, which typically ranges from 400 to 1500 ppm at atmospheric levels depending on location. While the container remains closed, CO_2 is gradually released from the inner to the outer container. Every minute, the system updates the CO_2 concentration reading and uses this data to calculate the amount of CO_2 released per gram of coffee beans. This rate of release is compared to the expected initial release rate for the selected roast type, and the system calculates the percentage of CO_2 remaining in the beans. It is important to note that the system assumes 100% CO_2 retention upon placement into the container. This assumption is based on the expected use case, where beans are added shortly after purchase, making the time since roasting minimal and CO_2 loss negligible. The final design is shown in Figure 1.1.

1.3 Visual Aid

Figure 1.1 - CO_2ffee on the Right Among Other Coffee Appliances

1.4 High Level Requirements

- The freshness rating must be reported as a percentage, representing the CO_2 remaining in the beans relative to its original state (100%). This gives the user a clear idea of the freshness of the beans.
- The weight sensor readings must accurately reflect bean withdrawal by ±2 %, and combined with CO_2 sensor data, they must determine the CO_2 loss per gram of beans. The weight being precise is important as it allows us to accurately measure CO_2 released per gram of beans.
- The user can select from three bean types, and press a button to open/close the outer lid for bean withdrawal. This is needed because different beans contain different levels of CO_2 per gram and need to be reflected in the system. The button is used in order for the user to retrieve beans from the system and for the system to know that there are beans being withdrawn.

2 Design and Verification

2.1 Block Diagram

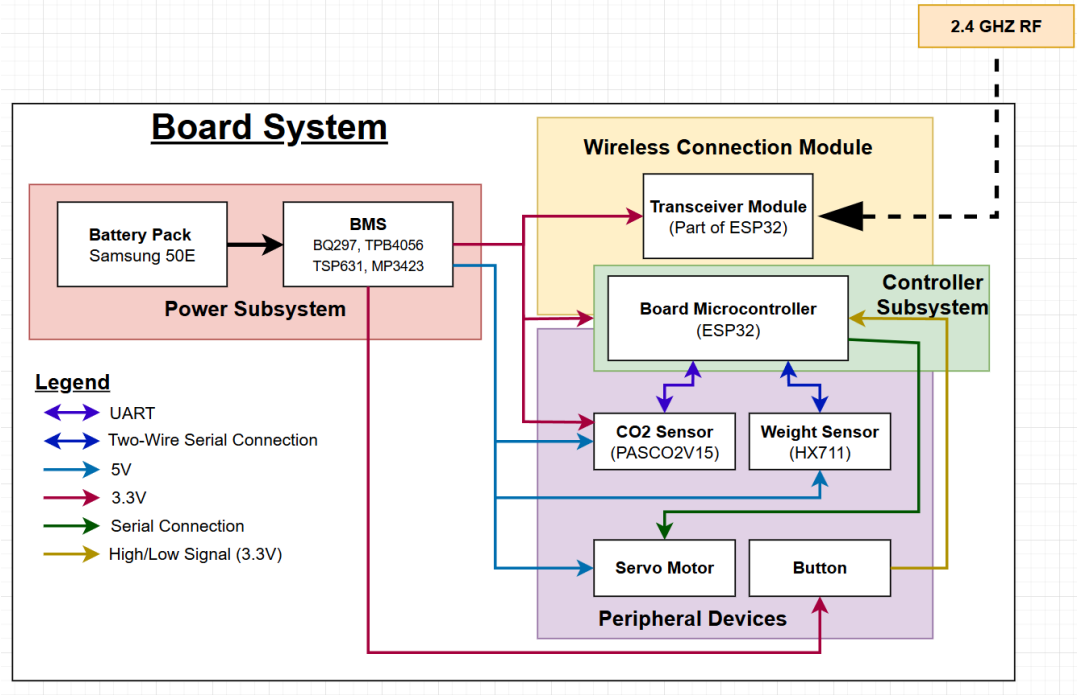


Figure 2.1 - Block Diagram of Components

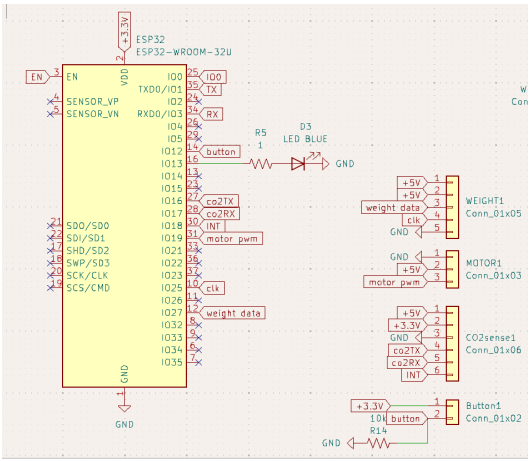
2.2 Subsystem Overview

All subsystems and their connections and communications are shown in Figure 2.1.

2.2.1 Subsystem 1 - Peripheral Devices (i.e. Sensors, Inputs, and Motors)

The peripheral devices subsystem needed to consider four main components: the CO_2 sensor, weight sensor, servo motor, and button, all of which needed to be managed by the controller. When picking out components, it was important to consider the available voltage levels on the board, the precision of sensor data, and the overall small profile to fit in the relatively small project.

Below, Figure 2.2 showcases the schematic of how the peripheral devices were connected to the controller (ESP32). Figure 2.3 highlights the physical connections on the board that were used to connect to each device.



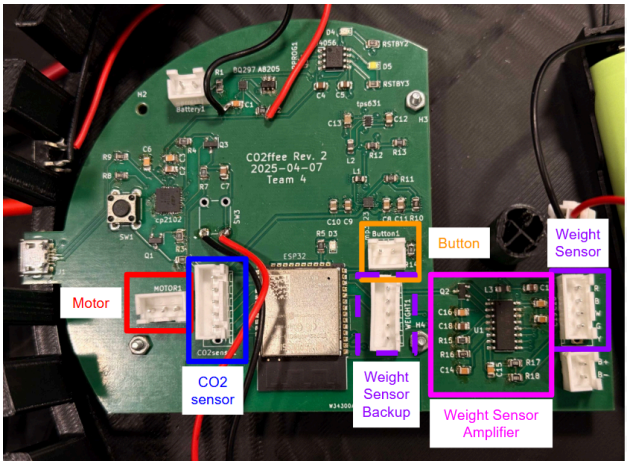


Figure 2.2 - Peripheral Devices Schematic

Figure 2.3 - Peripheral PCB Layout

To begin with, the servo motor was relatively simple to integrate into the system. It uses a simple PWM control scheme and could therefore be connected to nearly every pin on the ESP32 as most of them could support this data type. One of the main considerations when picking out a component was the limits of the ESP32, as the controller could only output 3.3V as a PWM, the selected motor must operate in this range. Under these considerations, the HS-318 [4] servo motor was selected due to its compatibility with system requirements and availability through the department's component inventory. The testing of this component was simple and once the ESP32 was verified to be programmable it was simply a test of plugging in the motor and running a variety of scripts to get an idea of the range of the motor in turning radius. After a general idea of the angles available, the motor was mounted to the container and calibrated to open and close the lid within the desired degree amounts.

While the servo motor was simple to operate, the button was even simpler. It is the simplest of I/O devices and could be used with any of the ESP32 pins. It was decided that the button would be an SPST button (a button that holds its state until pressed again) as this would make programming the interaction simpler as the system could simply periodically check the button's state. In the end, the button that was used in the system was an arbitrary button off of

Amazon that was used in previous personal projects and cost around 33¢ per button. Verifying this component was simple as the script only needed to probe the button every hundred milliseconds to determine its state and to be sure it was switching states when the button was pressed.

When integrating the weight sensor there were not many options available. The most common and affordable sensors were those used most commonly with Arduinos created by Sparkfun and were, therefore, our first choice (specifically the SEN-13329 [5]). However, these sensors required the usage of an external amplifier unit to turn the data from the load cell into controller-readable data (SEN-13879 [6]). This forced us to integrate this amplifier onto the board itself which was rather simple as the layout of the external amplifier was rather simple and could be replicated on our board with relative ease. Below in Figure 2.4, the schematic of this amplifier is shown. The PCB location of this amplifier can be seen in Figure 2.2 as a “Weight Sensor Amplifier”. You may have noticed that there were two connections for weight sensors in Figure 2.2, the “Weight Sensor Backup” connection is the connection that bypasses the amplification circuit and was used before integrating the amplifier onto the PCB to debug and test the overall system. The load cell when used initially outputs a range of numbers without a unit and needs to be scaled to have a unit attached to the readings. This was done using an additional scale where the weight of objects was compared to the readings from both scales and the weight sensor was easily scalable to output the desired grams and was found to be extremely accurate within ±2%.

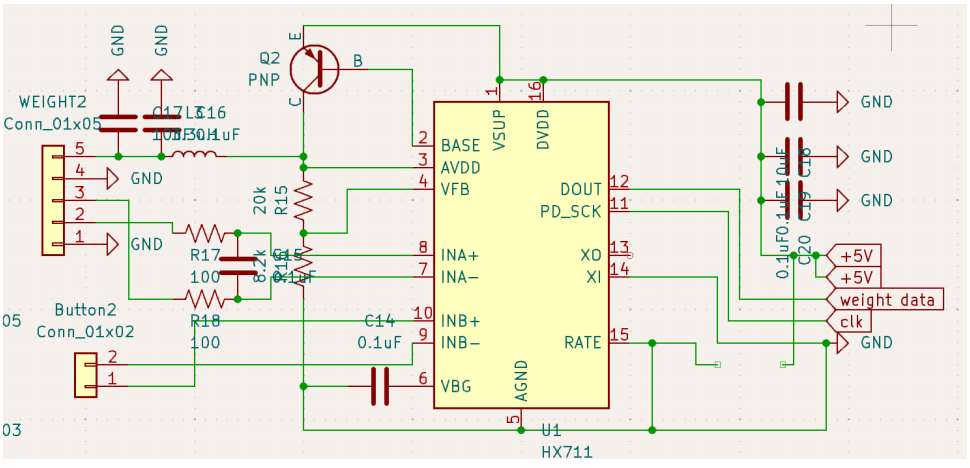


Figure 2.4 - Weight Sensor Amplifier Schematic

The final component of the peripheral devices was the CO_2 sensor. This component came with a few challenges. Once again there were very few options that were affordable and nearly immediately the PASCO2V15 [7] was selected for its price and seemingly ease of use. The sensor needed to be placed into the main container and because the sensor required a supporting circuit and was a SMD component it would need a dedicated PCB to function properly. The schematic of this PCB can be seen below in Figure 2.5 and the PCB and mounted component can be seen in Figure 2.6. The sensor allowed for multiple communication protocols and the UART communication method was decided to be used for simplicity which restricted the pins that could be used on the ESP32 making its pin designation forced to be IO16/17 as those were the only available pins that support this method. To verify this system worked as intended multiple locations were tested for their CO_2 readings and compared against expected values online for certain locations and it was therefore verifiable that the CO_2 sensor was accurate to the CO_2 readings (listed as ±5% on the datasheet below 3000 ppm).

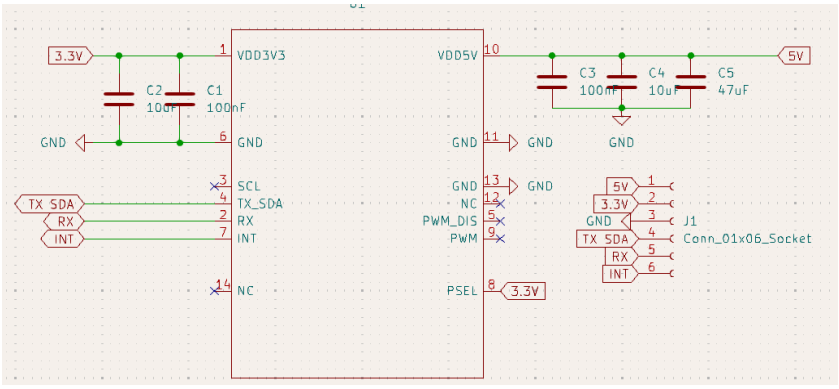


Figure 2.5 - CO_2 Sensor Board Schematic

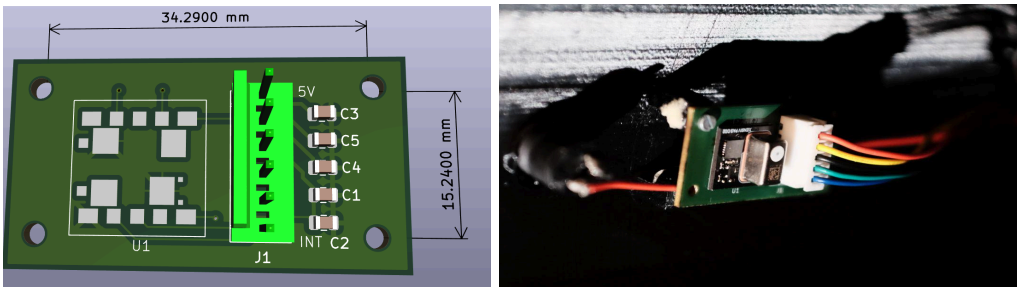


Figure 2.6 - CO_2 Sensor PCB Layout (left) and PCB Mounted in Container (right)

2.2.2 Subsystem 2 - Controller

The controller subsystem consists of two main components: the ESP32 microcontroller and a USB-to-UART interface. We selected the ESP32 specifically for its built-in Wi-Fi capabilities, which allow it to host its own wireless network and communicate data with a user interface. The USB-to-UART interface includes a CP2102 bridge chip that enables micro-USB connectivity and facilitates programming of the microcontroller by converting USB signals into UART data. The primary role of this subsystem is to manage all peripheral devices (as shown in Figure 2.7) and execute control tasks that drive the system's core functionality. Figure 2.7 shows the schematic diagram detailing the connections between the ESP32 and its peripherals, while Figure 2.8 presents a high-level block diagram illustrating their interactions.

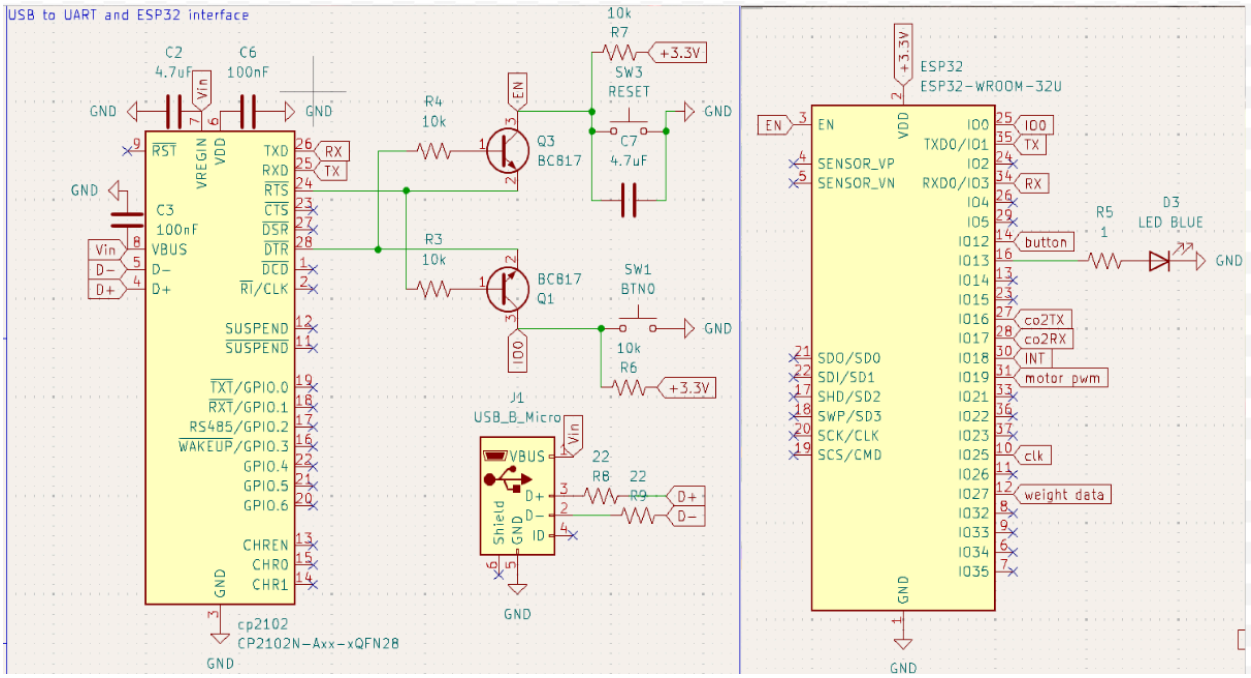


Figure 2.7 - ESP32 and Support Circuit for Programming

There are two main control tasks that govern the functionality of the system. The first is the button and lid task, which continuously polls the button state every 500 milliseconds. When the button is pressed, the lid is opened via a servo motor. Upon release, the lid closes, and new baseline values—such as CO_2 concentration and bean weight—are recorded. This ensures the system captures a stable reference point for subsequent freshness calculations. The second core task is the CO_2 sampling task, which measures the CO_2 concentration inside the container once per minute. This sampling rate is based on the sensor datasheet, which

indicates that accuracy significantly decreases with more frequent measurements. After each valid sample, the system updates the freshness level of the coffee beans based on the estimated CO_2 loss. The conceptual steps are as follows: the system first calculates the difference between the current and baseline CO_2 concentrations in parts per million (ppm). This value is then multiplied by the container volume to convert the CO_2 loss into milligrams. The resulting mass is divided by the weight of the beans in grams to obtain the CO_2 lost per gram of beans. Finally, the freshness score is computed as the ratio of remaining CO_2 to the estimated initial CO_2 content per gram.

It is important to note that CO_2 sampling and freshness updates only occur while the lid is closed. This ensures that the measurements reflect accumulated CO_2 released from the beans rather than contamination from ambient air. If the lid is open, sampling is temporarily skipped. Furthermore, if the CO_2 concentration exceeds 3000 ppm, the system initiates an aeration routine. This routine opens the lid to release excess CO_2 and resets the baseline, since sensor accuracy degrades beyond this threshold, as documented in the datasheet.

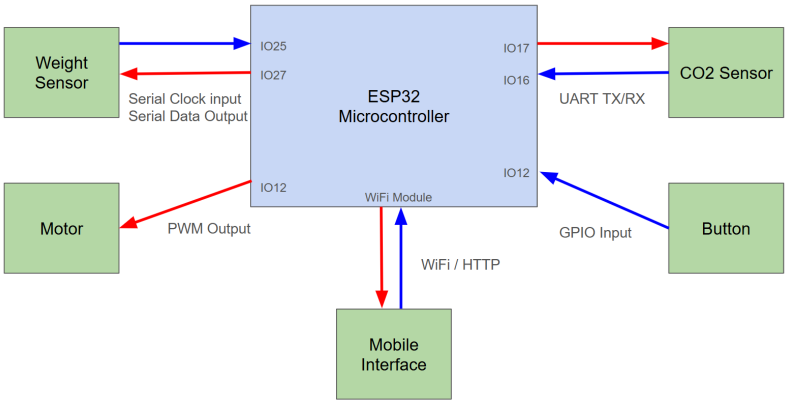


Figure 2.8 - ESP32 Interactions with Peripherals

To verify proper operation, a series of tests were conducted for each peripheral connected to the ESP32 microcontroller. For the weight and CO_2 sensors, test programs periodically recorded and displayed sensor readings to the terminal. These tests confirmed consistent and accurate CO_2 readings across different environments. The weight sensor demonstrated reliable performance, with detection accuracy within ±2% for all tested items.

The servo motor was also tested extensively, sweeping through its expected operating range from 60° to 120° and responding correctly to button input to open and close the lid. Interaction between the ESP32 and the mobile interface is discussed in the following section.

With the control software complete, the main freshness tracking algorithm was ready for testing and verification. In the simulation shown in Figure 2.9, 340 grams of dark roast beans were placed in the container and monitored over a 2-hour period to evaluate long-term freshness tracking. As shown in the graph, CO_2 concentration increased as the beans released gas, leading to a gradual decrease in freshness. A total of 2114 ppm of CO_2 was released, resulting in a final freshness score of 99.59% after 2 hours. Although the CO_2 release curve appears linear, this is due to the limited observation window; over longer periods, dark roast beans are known to release CO_2 rapidly at first, followed by a slower, plateauing release rate.

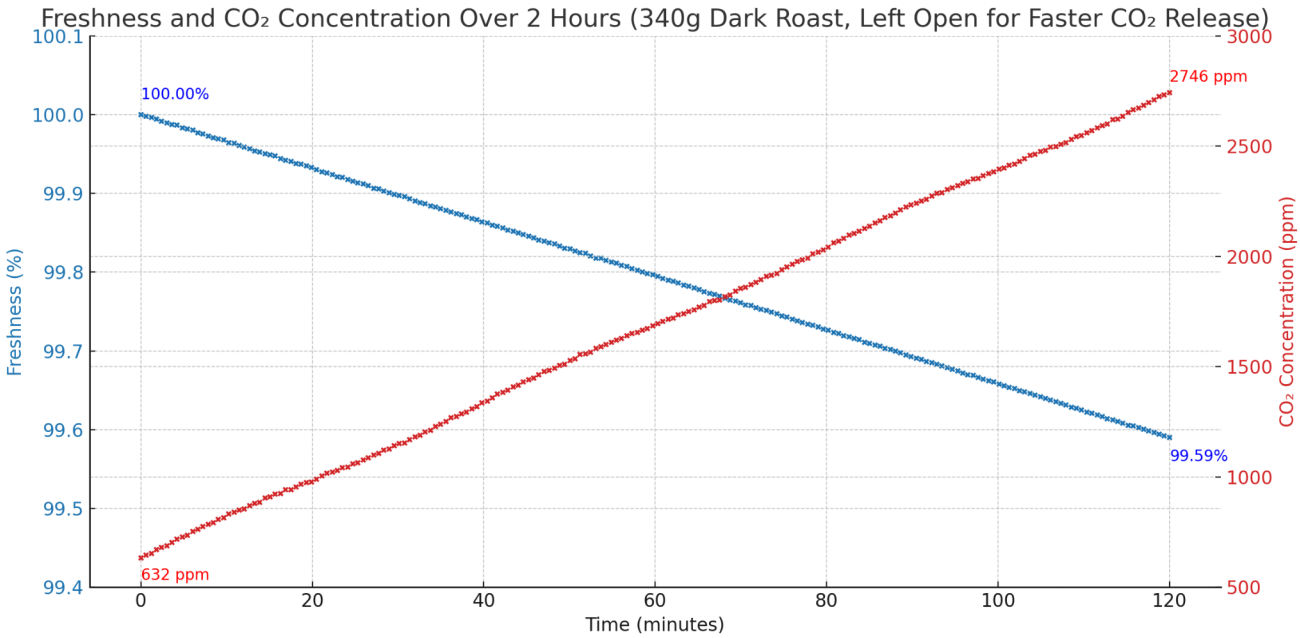


Figure 2.9 - Freshness Tracking Simulation

2.2.3 Subsystem 3 - Wireless Connection Module and User Interface

The wireless connection module and mobile interface consist of the ESP32 microcontroller, which includes a built-in Wi-Fi module, and any web-enabled device such as a phone or laptop that serves as the user interface (UI). The ESP32 hosts its own Wi-Fi network and web page, while the UI connects to this network. These components operate in a server-client relationship, where the ESP32 acts as the server and the UI as the client, communicating via the HTTP protocol. The client sends HTTP requests, and the ESP32—running an HTTP server—handles and responds to them appropriately.

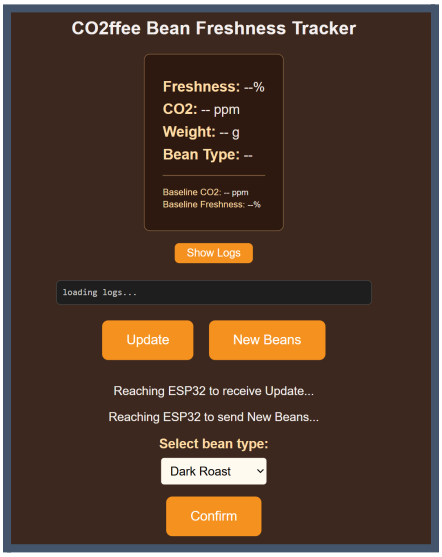


Figure 2.10 - User Interface

The client can send three types of requests through the UI as depicted in Figure 2.10. The first is the update request, which is a GET request that asks the server to prepare key data such as freshness, CO_2 concentration, weight, and bean type. The server responds in JSON format, and the UI formats this information in the main data box. The second is the new beans request, which is a POST request that sends the roast type of newly added beans to the server. This information is stored and used for future freshness calculations, and the server returns an acknowledgment to the client. Lastly, the show logs request is another GET request that returns a large raw text block containing system logs, which the UI displays for debugging and transparency purposes.

Verifying this subsystem involves ensuring that the ESP32 successfully hosts its Wi-Fi network and runs the HTTP server. A web-enabled device then connects to the network to test whether HTTP requests receive the correct responses. The verification was successful, with all requests returning valid responses and latency consistently remaining below 5 seconds, as desired.

2.2.4 Subsystem 4 - Power System

We want our power system to ensure that each component receives enough power to work correctly under their expected load. We compared the voltage and current ratings of each component at maximum load and decided to use a 3.7V, 2.6Ah Li-ion battery. We can see the respective voltages and maximum current draws of each component in Table 2.1 below.

Table 2.1 - Component Power Table

Component	Operating Voltage	Maximum Current Draw
ESP32	3.3V	500mA
Push Button	3.3V	Negligible
PASCO2V15 CO_2 Sensor (3.3V Rail)	3.3V	10mA
PASCO2V15 CO_2 Sensor (5V Rail)	5V	300mA
Servo Motor	5V	1A
SEN-14729 Load Cell	5V	5mA
HX711 Load Cell Amplifier	5V	1.5mA

Our BMS consists of 2 components, the protection and charging circuits. The BQ297 is a single-cell Li-ion battery protection IC that we incorporated to ensure that it has protection from overcurrent. The TP4056 is a single cell linear battery charger that allows us to charge the battery via the USB port that we also use to program the ESP-32. The schematics and layouts of each component are shown in Figure 2.11 and 2.12 below.

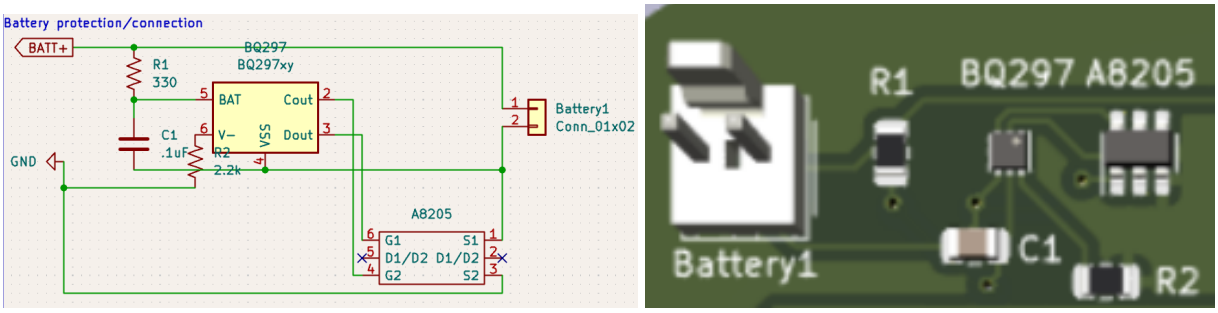


Figure 2.11 - BQ297 Battery Protection Circuit Schematic and Layout

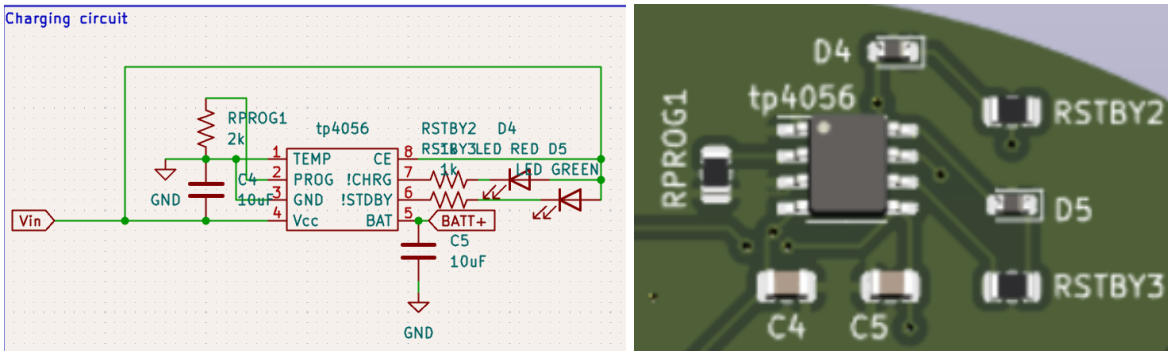


Figure 2.12 - TP4056 Linear Battery Charger Circuit Schematic and Layout

Now that we have the battery and BMS set up, we now need to Figure out how to produce 3.3V and 5V from the 3.7V battery. We decided to use DC-DC converters as they have high efficiency and minimal heat dissipation.

Starting with the 3.3V DC-DC converter, here in Eq 2.1 are the calculations for the maximum current draw for the rail:

500mA (ESP-32) + 10mA (PASCO2V15) + 0mA (Button) = **510mA** (Eq. 2.1)

Considering our max current draw of 510mA, we decided on the TPS631. The TPS631 outputs 3.3V and 1.5V max at 93+% efficiency with a 1.2mm x 2.1mm package and minimal circuitry around it. The schematic and layout for the 3.3V conversion circuit using the TPS631 can be found in Figure 2.13 below.

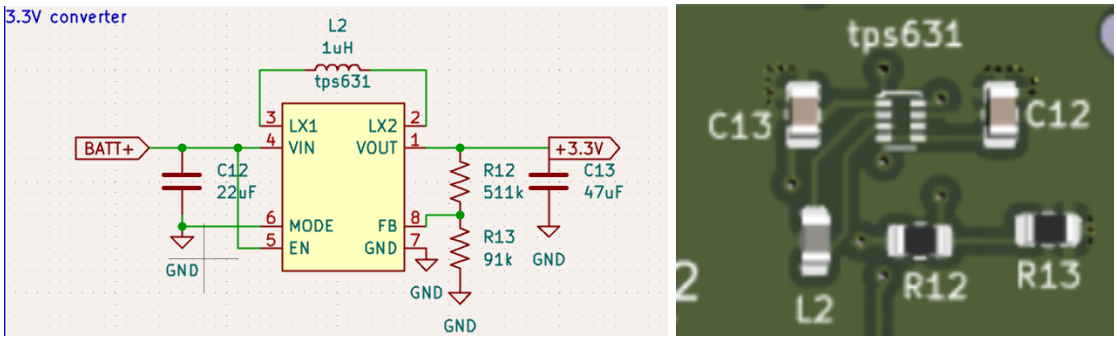


Figure 2.13 - TPS631 3.3V Buck Converter Circuit Schematic and Layout

When we looked for a 5V boost converter for our system, we followed an extremely similar process to our selection process for our 3.3V buck converter component. Starting with the maximum current draw for the 5V system, calculations can be found in Eq. 2.2 below:

300mA (PASCO2V15) + 1A (Motor) + 5mA (Sen-14729) + 1.5mA (HX711) = **1.3065A** (Eq 2.2)

With our considerations, we used the MP3423, allowing us to convert our 3.7V battery to 5V and up to 3.1A at 94%+ efficiency (96%+ for our typical use case) in a 2mm x 2mm package. The schematic and layout for this 5V boost converter system can be found in Figure 2.14 below.

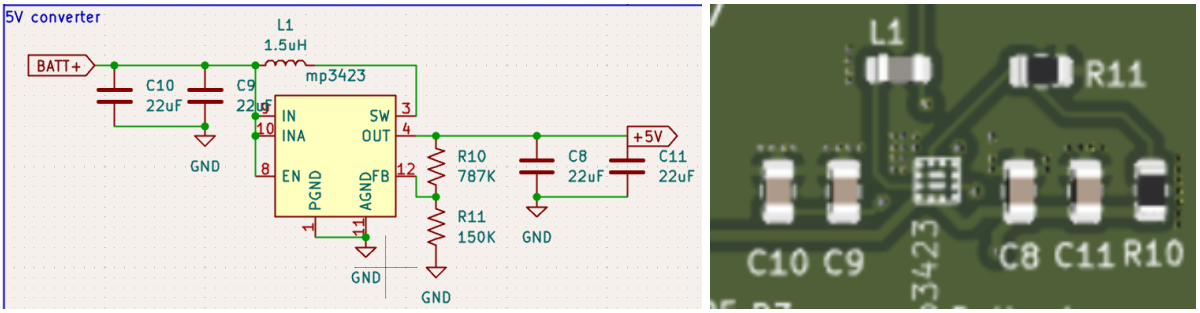


Figure 2.14 - MP3423 5V Boost Converter Circuit Schematic and Layout

2.2.5 Physical Design

The final physical design did not change much compared to the original design found in the Design Document (preliminary design shown in Figure 2.15). The concept of separating the

main container where the beans will be weighed and where CO_2 readings will be taken from the Electronic Housing remained throughout the project and the only other considerations taken into account were the mounting apparatuses needed to mount all PCBs and components both in and out of the container. It is also worth noting that the reason that the PCB is shaped somewhat like a circle is to prevent the PCB from not fitting within the bounds of the electronic housing walls. Because the container was a circle and the PCB was also a small circle there would be no way for the PCB not to fit in the container as a smaller circle will always fit in a larger circle if it is inside and has one point tangent to the larger circle. The added section of the PCB that breaks its circle structure was made after the physical layout was completed and it was known that it would fit fine within the provided space.

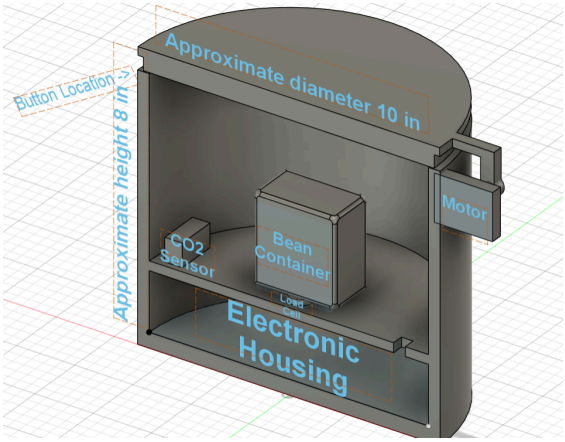


Figure 2.15 - Preliminary Design of Physical Layout

The designing phase of the physical layout took place after the first PCB design phase so the dimensions of the mounting holes and overall PCB was easily accessible. Taking these dimensions to create the necessary standoffs for mounting and external access for the USB port was straightforward for the electronic housing. The mounting holes for the battery were similarly simple as a basic battery housing was used and its dimensions were readily available on its datasheet.

As the PCB for the CO_2 sensor was also created before the design phase of the physical layout it was similarly trivial to create a mounting bracket on the inside of the main container with the dimensions created when designing the CO_2 PCB. The bracket was placed on the inside right below the button as this location is presumably harder to see from the perspective of the user making the final product slightly more appealing.

For mounting the weight sensor it was also only a matter of finding the datasheet and orienting the hole at the bottom of the main container to allow the other side of the weight sensor to be in the middle of the container with the correct screw hole size (M5). For the actual weight plate itself, it was simply a disk that had a bit of space for the hole so the head of the screw remained flush with the surface of the plate. M5 nuts were also used as adjustable standoffs to easily determine how high each element needed to be for optimum placement and minimum interference with one another.

Mounting the button was a simple matter of creating a hold in the container with the correct dimensions found by measuring using wire and a measuring tape.

Mounting the servo motor also was just a matter of looking at the datasheet, finding its dimensions, and creating both a bracket to firmly hold the motor in place on the exterior of the main container as well as placing it so that the expected center of rotation for the lid is lined up with the center of the containers circle. The lid itself was one of the harder components to design as its mounting section had to line up with the servo motor's mounting extra components whose dimensions were not readily available and needed to be hand measured. The lid was also designed to slightly self-correct itself into the container as the container's lip and lid's edge were both angled to slide into one another with more accuracy.

Figure 2.16 shows the final product both inside and outside as annotated and Figure 2.17 shows the CAD models used to 3D print the container.

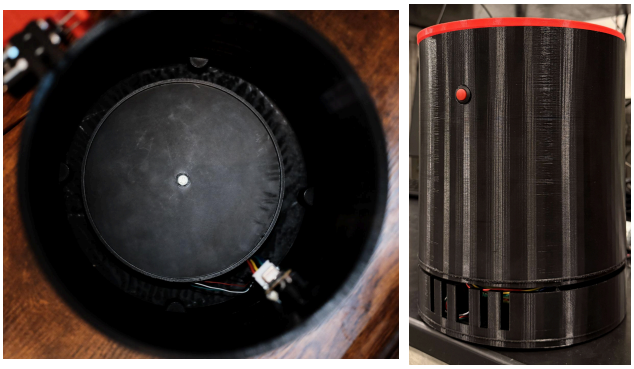


Figure 2.16 - Final Product Exterior (left) and Interior (right)

Figure 2.17 - CAD Models of All Container Components

3. Costs and Schedule

Table 3.1 - Total Cost Table

Part Costs and Description						
Part Number	**Manufacturer Part Number**	**Description**	**Quantity**	**Unit Price**	**Extended Price USD**	**links**
1568-1436-ND	SEN-13879	LOAD CELL AMP HX711	1	$10.95	$10.95	link
336-5886-1-ND	CP2102N-A02-GQFN20R	IC USB TO UART BRIDGE QFN20	1	$5.59	$5.59	link
296-43985-1-ND	BQ29700DSER	IC BATT PROT LI-ION 1CELL 6WSON	1	$0.52	$0.52	link
4518-8205ACT-ND	8205A	MOSFET 2N-CH 20V 5A SOT23-6	1	$0.37	$0.37	link
5503-TPB4056B2X-ES1RCT-ND	TPB4056B2X-ES1R	LINEAR BATTERY CHARGER 1 CELL 8-	1	$1.05	$1.05	link
1589-1642-1-ND	MP3423GG-Z	IC REG BOOST ADJ 9A 14QFN	1	$3.46	$3.46	link
296-TPS631000DRLRCT-ND	TPS631000DRLR	DC DC CONVERTER	1	$1.42	$1.42	link
1568-1900-ND	SEN-14729	LOAD CELL 5KG STRAIGHT BAR TAL22	1	$13.12	$13.12	link
445-8657-1-ND	MLZ2012M1R0HT000	FIXED IND 1UH 800MA 100 MOHM SMD	1	$0.10	$0.10	link
445-6757-1-ND	MLZ2012N1R5LT000	FIXED IND 1.5UH 900MA 100MOHM SM	1	$0.10	$0.10	link
BH-18650-PC-ND	BH-18650-PC	BATTERY HOLDER 18650 PC PIN	1	$3.12	$3.12	link
1568-1488-ND	PRT-12895	BATTERY LITH-ION 3.7V 2.6AH 1865	1	$6.62	$6.62	link
PASCO2V15AUMA1	Infineon Technologies	CO_2 SENSOR IN PPM	1	$20.92	$20.92	link
Values include: 1, 22, 330, 1k, 2k, 2.2k, 10k, 91k, 150k, 511k, 787k	--	Various resistors of different values	17	$0.10	$1.70	--
Values include .1uF, 4.7uF, 100nF, 10uF, 22uF, 47uF	--	Various capacitors of different values	19	$0.10	$1.90	--
Colors include: Green, Red, Blue	--	different colored LEDs	3	$0.14	$0.42	--
BC817-25LT1G	onsemi	TRANS NPN 45V 0.5A SOT23-3	2	$0.15	$0.30	link
10118194-0001LF	Amphenol ICC (FCI)	CONN RCPT USB2.0 MICRO B SMD R/A	1	$0.41	$0.41	link
HS-318	Hi-Tec	25 TOOTH SERVO MOTOR	1	$18.78	$18.78	link
TS02-66-50-BK-260-LCR-D	Same Sky (Formerly CUI Devices)	PUSHBUTTON THT	2	$0.10	$0.20	link
--	OVERTURE	1.75MM PLA 3D PRINTER FILAMENT	1	$24.00	$24.00	link
Total part cost:					$115.05	

Labor Cost			
Number of People	**Average hourly salary**	**Estimated individual hours worked**	_
3	$45	200	
GRAND TOTAL COST:			**$27,115.05**

Table 3.2 - Weekly Schedule

Week of	Task	Assignments
February 17	Abrar: Brainstormed possible freshness tracking algorithms Nathan: Began schematic planning Josh: Start selecting components for power subsystem	Proposal Review
February 24	Abrar: Polished the freshness tracking algorithm and procedures Nathan: Designed first PCB schematic Josh: Work on layout for first round of PCB	PCB Review
March 3	Abrar: Set up ESP-IDF environment to write code on ESP32 Nathan: Designed CO_2 sensor schematic and ordered PCB Josh: Noting down specs/calculations of power components	First Round PCB and Design Document
March 10	Abrar: Programmed arduino for breadboard demo Nathan: Preparing for Breadboard Demo Josh: Helped wire components for Breadboard Demo	Breadboard Demo
March 17	Enjoy Spring Break	Spring Break
March 24	Abrar: Wrote motor and button drivers and tests Nathan: Assisted in bench testing PCB/began 3D design Josh: Soldered PCB and helped test power	
March 31	Abrar: Wrote weight and CO_2 sensor drivers and tests Nathan: Further board testing and 3D modeling/updated CO_2 Josh: Helped with new CO_2 sensor PCB layout	Third Round PCBs / Individual Reports
April 7	Abrar: Designed UI, wrote code and tests for HTTP server/Wi-Fi Nathan: Modular testing and 3D printing container Josh: Added HX711 module onto PCB	Fourth Round PCBs
April 14	Abrar: Wrote final system logic, integrating all drivers Nathan: Assembling of final container and final debugging Josh: Helped test all individual subsystems	Team Contract Assessment
April 21	Abrar: Ran simulations to produce results with logs and graphs Nathan: Assisted in final PCB testing and assembly Josh: Helped assemble whole product and test everything together	Mock Demo
April 28	Abrar: Prepared results, including logs and graphs for final demo Nathan: Final demo final touches and presentation prep Josh: Soldered round 4 PCB and helped test it	Final Demo/Mock Presentation
May 5	Abrar: Finished up final presentation and paper Nathan: Presentation work and final paper revision Josh: Work on presentation and final paper	Final Presentation / Papers / Checkout

4. Conclusion

4.1 Accomplishments

Considering we finished everything that we expected to for our project, we do have a lot of accomplishments. We were able to satisfy all of our high level requirements and develop a product that a person could use practically. The user would be able to open the container, put their beans in, select a bean type on the mobile interface, and close the container. Over time, they could use the beans and track the freshness levels of the beans. This wouldn't have been possible without meeting all of our subsystem requirements, which we also completed.

4.2 Uncertainties

Although we would largely consider our project a success, there were still a couple of issues and challenges we ran into throughout our design. In terms of hardware, soldering small components proved to be a major challenge as it required steady hands and careful placement. Poor soldering was also hard to detect and debug given the components were very small. Additionally the CO_2 sensor datasheet was not very clear with the layout so we had to go through a couple of iterations on the custom PCB for the CO_2 sensor before having it align with the actual sensor. In terms of software, a big issue was ensuring thread safety for concurrency. Since we have multiple threads running concurrently, all shared data between the threads initially corrupted one another leading to bad data so we had to use mutual exclusion to protect the data.

We also have some things we would update about the project if we had more time. The first change is making the battery bigger to allow the system to be more portable and survive longer without charging it. The second thing we would improve is our physical design tolerances. Because we wanted to open the electronics housing easily when working on the project, the bottom and top housing are not secured together. We would want to change this

so nothing falls apart when transporting it. The final thing is moving our interface to a Bluetooth connection instead of Wi-Fi, as it draws less current overall and allows the user to use the internet while still being connected to our device.

4.3 Ethical Considerations

Our coffee bean freshness detector assumes that beans start with 100% CO_2 retention, estimating initial content based on the selected bean type; accordingly, freshness is reported as an approximate percentage of remaining CO_2. In line with IEEE Code of Ethics I.5, we will clearly inform users of these assumptions and limitations. To ensure safety per ACM Code 1.2, our power system includes precautions such as housing inactive batteries in an isolated, non-conductive box secured with screws and covers to prevent hazards. Additionally, following IEEE Code I.1, all bean-contact surfaces will use food-safe materials, while the outer container will be made from stable, non-reactive PLA to ensure sanitary storage.

4.4 Future Work

While no one in our group has any current plans of further development, this product has a large capacity for improvements and refinements that could push it to be a marketable product. Currently the structure itself is too large, bulky, unrefined, and contains exposed wires. The effectiveness of the product also needs further testing and needs to be tested with an actual user to determine and rectify any problems that may arise stemming from user error. The market for this device might also be slightly niche as it only is attractive for the most avid coffee lovers. However, these problems could be rectified and production of a marketable product that satisfies its target demographic.

References

[1] "Measuring Carbon Dioxide (CO_2) Levels in Roasted Coffee," MTPak Coffee, May 2021. [Online]. Available: https://mtpak.coffee/2021/05/measuring-carbon-dioxide-CO_2-levels-roasted-coffee/. [Accessed: May 7, 2025].

[2] "Is Your Coffee Too Fresh?" Clive Coffee. [Online]. Available: https://clivecoffee.com/blogs/learn/is-your-coffee-too-fresh. [Accessed: May 7, 2025].

[3] M. Yeretzian, S. Aeschbacher, K. Jordan, and L. M. Blank, "How Fresh Is Fresh? Understanding the Influence of Roasting, Grinding, and Storage on Coffee Aroma," J. Agric. Food Chem., vol. 66, no. 47, pp. 11835–11844, 2018. [Online]. Available: https://pubs.acs.org/doi/10.1021/acs.jafc.7b03310. [Accessed: May 7, 2025].

[4] "HS-318 Servo," ServoCity. [Online]. Available: https://www.servocity.com/hs-318-servo/. [Accessed: May 7, 2025].

[5] "TAL220M4 Load Cell Datasheet," SparkFun Electronics. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/TAL220M4M5Update.pdf. [Accessed: May 7, 2025].

[6] "HX711 Data Sheet," SparkFun Electronics. [Online]. Available: https://cdn.sparkfun.com/assets/b/f/5/a/e/hx711F_EN.pdf. [Accessed: May 7, 2025].

[7] XENSIV PAS CO_2 5V Sensor – Product Brief, Infineon Technologies, v01.00. [Online]. Available: https://www.infineon.com/dgdl/Infineon-XENSIV_PAS_CO_2_5V_Sensor-ProductBrief-v01_00-EN.pdf. [Accessed: May 7, 2025].

[8] Monolithic Power Systems, "MP3423 9 A, 600 kHz High-Efficiency Synchronous Step-Up Converter with Output Disconnect," MP3423 Datasheet, [Online]. Available: https://www.monolithicpower.com/en/mp3423.html. [Accessed: May. 6, 2025].

[9] Texas Instruments, "TPS631000DRLR 1.5-A Output Current High-Power-Density Buck-Boost Converter," Digi-Key Electronics. [Online]. Available: https://www.digikey.com/en/products/detail/texas-instruments/TPS631000DRLR/15965499. [Accessed: May. 6, 2025].

[10] Texas Instruments, "BQ2970 Cost-Effective Voltage and Current Protection IC for Single-Cell Li-Ion Batteries," BQ2970 Datasheet, [Online]. Available: https://www.ti.com/product/BQ2970. [Accessed: May. 6, 2025].

[11] Top Power ASIC Corp., "TP4056 1 A Stand-Alone Linear Li-Ion Battery Charger," LCSC Electronics. [Online]. Available: https://www.lcsc.com/product-detail/Battery-Management-ICs_TPOWER-TP4056_C382139.html. [Accessed: May. 6, 2025].

[12] SparkFun Electronics, "PRT-12895 18650 3.7 V 2.6 Ah Lithium-Ion Battery," Digi-Key Electronics. [Online]. Available: https://www.digikey.com/en/products/detail/sparkfun-electronics/PRT-12895/5271298. [Accessed: May. 6, 2025].

Appendix A Requirement and Verification Table

Requirements	Verifications	Verification status (Y or N)
Peripheral subsystems Subsystem		Y
This subsystem must send and receive reliable data to the controller subsystem.	The CO_2 sensor outputs approximately 420 ppm at ambient and increases in a higher concentration of CO_2 which can be done with calibrated scales and coffee beans determining the loss of CO_2 and increase in ppm respectively. The decrease in weight should coincide with the increase in ppm over time and during every ppm reset the weight will be taken again. The resulting calculation from this process should show that the CO_2 decreases the same amount within a 10% error margin. This process will also be done to calibrate the container in order to get an accurate freshness measurement. All values of the ppm and weight measurements will be taken over time (no beans will be removed during this process as a control) and the resulting CO_2 losses will be compared.	Y
	This sensor must output data consistent with previously tested objects on a calibrated scale and correctly outputs data within 5%. Multiple trials comparing results will be recorded and the average error will be calculated and should be below 5% error.	Y
	The servo motor must rotate a given angle provided by the controller within 10 degrees. This motor will be run multiple times between two angle values to be sure the angle does not diverge greatly from the expected servo angles.	Y
The motor must respond according to the button input and CO_2 sensor readings.	Container automatically opens before reaching 3000 ppm in the container and opens upon the user pressing the button which opens the container at the next available time in the operation cycle and closes when the button is pressed again.	Y

The CO_2 sensor must periodically record a measurement of the ppm of the CO_2 in the container and immediately update the server notifying the user.	The CO_2 sensor will be configured to update the ppm reading of the CO_2 every hour or whenever the user utilizes a forced update utilizing the web server. This means that the user should be updated within approximately 1 minute from pressing the force update.	Y
Controller Subsystem		Y
The esp32 Wi-Fi module must have a reliable Wi-Fi signal in order to update the user of the coffee status and receive the needed information for calibration.	ESP32 has a single and is able to host servers that the user may interact with. Which should be reachable as long as both the container and phone are within an internet range. Which should have the ability to communicate with the ESP32 with minimal delay at most 5 seconds communication delay.	Y
The ESP32 must be easily programmable to quickly debug	There should be no other steps beyond simply plugging in the usb micro b connection to the board and uploading a program to reprogram the ESP32.	Y
The controller should be interrupted by the user pressing the outer button or interacting on the app to update or putting in new beans.	Regardless of what current process the ESP32 is doing it should put the user first and accommodate the users commands with minimal delay. This should never result in a mixup in commands or a crash in the software onboard.	Y
Wireless Connection Module and Mobile Interface		Y
Reliable Wi-Fi connection to have constant connection to the controller subsystem in order to send and receive data.	Verify that the controller subsystem maintains a continuous and stable Wi-Fi connection by monitoring connection status over a prolonged period and under varying conditions (e.g., distance, interference). This can include automated tests that repeatedly ping the controller and log any disconnects or latency spikes.	Y
A server should be running on the microcontroller to receive requests from the mobile interface.	Confirm that a server is running on the microcontroller by sending test requests from the mobile interface and verifying that valid responses are received within an acceptable response time. Log and analyze the responses to ensure all endpoints behave as expected.	Y

Sensor readings and a freshness report should be displayed and user inputs should be available on the mobile interface.	Validate that sensor readings and the freshness report are correctly displayed on the mobile interface by comparing live data against known calibration values and verifying that the display updates. Also, test user inputs (such as selecting bean types or triggering the lid) to confirm that changes are immediately reflected on the interface and that commands are properly executed by the controller.	Y
Power Subsystem		Y
Supplies continuous power to all subsystems at rated voltages without dropping current required during operations when the motor is activated. Is able to recharge and continue functioning without a jump in voltage or current from the recharge port.	The power supply for the 3.3 V should never diverge more than 0.2 V even if the motor is running at peak power or if the user plugs in the device. The 5V power supply should never diverge more than 0.4 V under the same conditions. This allows for enough leeway for all devices to operate at their rated voltages. This will be tested by monitoring the voltage of the power lines when turning on the motor or plugging in the devices as these are the times with the greatest disturbance in the system.	Y
Protects the battery from overcharge and protects from surge in power demand.	To test that the battery is safe from overcharge, the current to the battery will be monitored while charging to be sure that the on board linear charging IC is correctly operating. To test battery protection a similar process will be done when a large demand is suddenly pulled from the battery and the battery protection module correctly eases the battery into the new power demand.	Y