Kitchen Dry Ingredient Tracker: Final Report

ECE 445: Senior Design
Team 43
1st May 2024

Anju Jain (anjuyj2)
Nynika Badam (nbadam2)
Sanjana Kumar (spkumar4)

TA: Vishal Dayalan
Professor: Arne Fliflet

Abstract

This report outlines the motivation and construction of the Kitchen Dry Ingredient Tracker system, built by Team 43 for ECE 445 Senior Design. The system tracks the weights of up to three spices through an iOS mobile application with the use of load cells, RFIDs, and LEDs. The RFIDs track multiple spice spots allowing for accurate weight measurements to be assigned to each spice. The LEDs provide a visual representation for spices that are running low. The system utilizes a PCB to connect RFIDs, load cells, and LEDs to an ESP32. The PCB is powered through a wall outlet. The final product fulfills all high level requirements.

Contents

1. Introduction

1.1 Purpose

In today's world, there is no end-to-end maintenance system for kitchen ingredients. It is hard to keep track of ingredients in our kitchen, their availability, and gauge when a necessary grocery run is required. Even though online grocery shopping has gotten very popular, consumers still frequently go to physical grocery stores. The primary grocery shopper in U.S. households made an average of 1.6 shopping trips per week in 2022 [12]. This implies that many consumers usually forget to buy ingredients they need and are forced to go back to the store. The Kitchen Dry Ingredient Tracker is designed to optimize these grocery trips by cutting down on the number of grocery runs a consumer makes.

The system is designed to track and communicate with users about their spice necessities. The system's app allows users to tailor each individual spice to different lower weight threshold measurements. The system uses RFIDs to allow users to place spice containers in any slot while recognizing the spice and its lower weight threshold measurement. The app maintains a dynamic digital grocery list by adding or removing spices based on their individual weights. If any items are in the grocery list, the user will receive a notification. The Grocery List interface also has a Google Maps view of grocery stores near the user's current location to encourage them to stop at a grocery store. To allow for a visual representation that a spice has run below the user specified weight limit, the LED light at the container's spot turns red.

1.2 Functionality

1.2.1 High Level Requirement 1

The MCU pulls weight data captured from the load cells every 30 seconds for the 3 spices tracked.

This is useful for users who have multiple people in their household who cook. When a member is at home cooking and using the spices, the weights will be reflected within the app. If another user happens to be outside grocery shopping, they will always have the most up to date spice information.

1.2.2 High Level Requirement 2

The app will have an "Ingredient Dashboard" and "Grocery List" interface. The Ingredient Dashboard allows users to specify lower weight thresholds for each spice within the range of 0 - 500 grams and recorded in whole grams. The Grocery List interface maintains a list that adds/removes spices based on whether their recorded weight is below its lower weight threshold.

Since the average consumer makes 1.6 shopping trips per week, this app will eradicate unnecessary shopping trips by maintaining a dynamic grocery list.

1.2.3 High Level Requirement 3

Users can place spice containers in any of the 3 spots. The container's RFID keeps track of the container's position, allowing for the right comparison between the spice's weight and its lower threshold weight.

This design is more user friendly as a spice can be put in any spot. The spices are tied to a certain RFID and not spot, allowing users to not worry about where they place the spice in the design.

1.3 Subsystem Overview

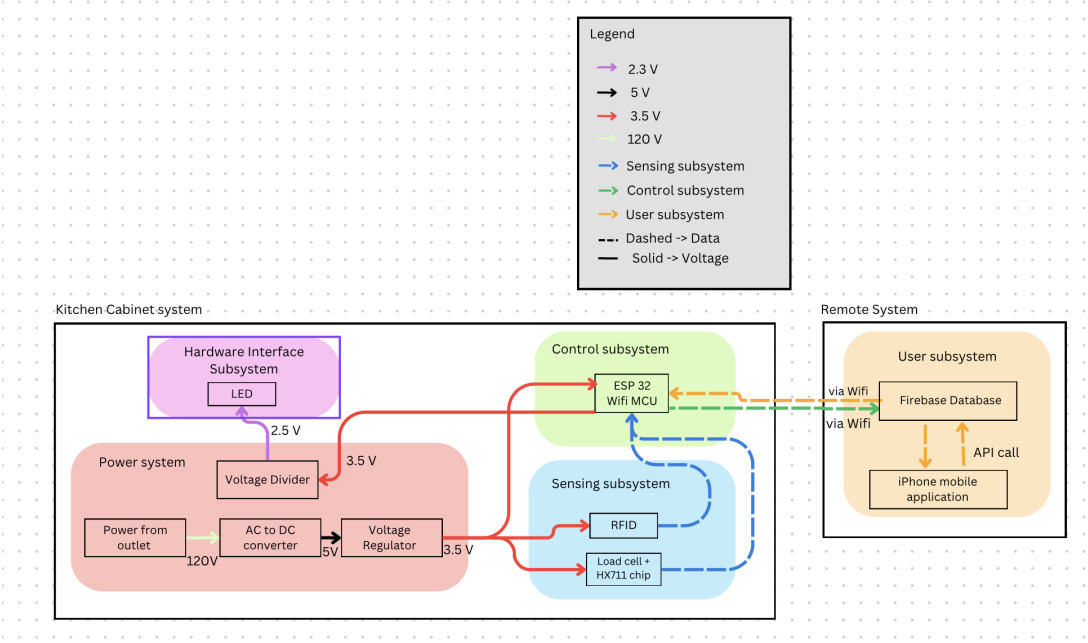

Figure 1: Block Diagram

The system is made up of a power subsystem, control subsystem, sensing subsystem, hardware interface subsystem, and user interface subsystem. The power subsystem powers the device. A wall power adapter converts 120V AC to 5V DC and a voltage regulator is then used to step down the 5V to the 3.3V needed by ESP32, RFIDs, and load cells. The system uses resistor voltage dividers to step down the voltage from 3.3V to 2.3V for the LEDs. The sensing subsystem includes the RFIDs and load cells. The container's RFID keeps track of the container's position, allowing for the right comparison between the spice's weight and its lower threshold weight. The load cells in conjunction with the HX711 chip measure the weights of the spices. The control subsystem is made up of the ESP32. The Wifi protocol of the ESP32 is used to communicate with the Firebase database to store data. This data is transferred and shown

within the app. The user interface subsystem allows the user to input the spice names and their lower thresholds, see spice weights, maintain a grocery list, receive grocery list notifications, and see a Google Map view of grocery stores 0.5 miles away from them. The hardware interface subsystem consists of LEDs that turn red when the spice weight runs below its lower weight threshold.

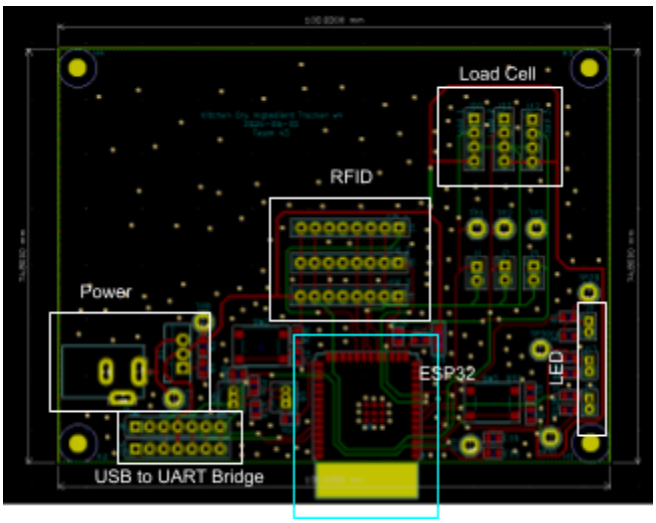

Figure 2: Final PCB Design

Figure 2 highlights the main components of the PCB. The Power box contains the barrel jack connector and the LM317. The RFID box contains the connections required for the three RFIDs. The Load Cell box contains the load cell connections required for the power, clock, ground, and data transfer for each load cell. The LED box highlights the connections required for the LEDs. The ESP32 is highlighted with the blue box. The USB to UART Bridge box has all the connections required for programming our ESP32 with Arduino. The PCB uses the Bare Minimum schematic for the ESP32 as found on the course website. The wrong footprint was chosen for the BJTs required for programming the ESP32. However, since the PCB does contain the two switches, GPIO0 pull-down switch and RESET/CHIP_PU pull-down switch, we were still able to program the ESP32. The PCB successfully connected the load cells, RFIDs, and LEDS to the ESP32 and provided the correct voltages to each component.

2. Design

2.1 Subsystem Diagrams and Schematics

Figure 3 through Figure 9 are all the app screens and KiCad schematics used in the project.

2.1.1 User Subsystem

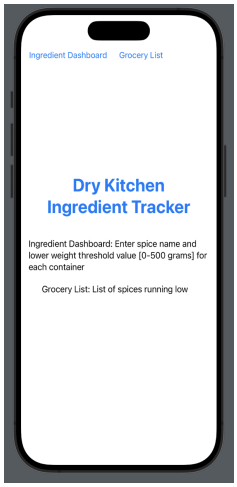

Figure 3: Home Screen

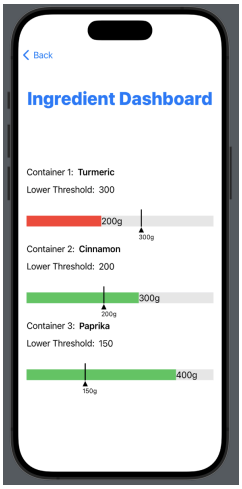

Figure 4: Ingredient Dashboard

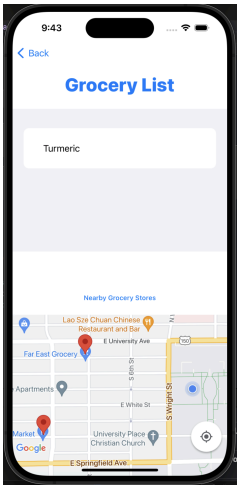

Figure 5: Grocery List

2.1.2 Sensing Subsystem

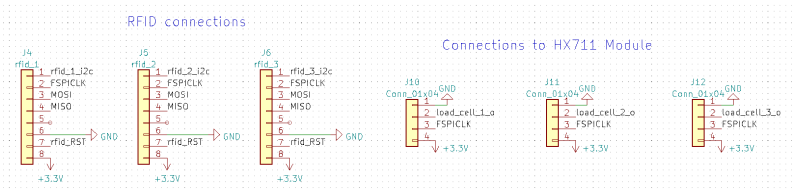

Figure 6: Sensing Subsystem Schematic

2.1.3 Control Subsystem

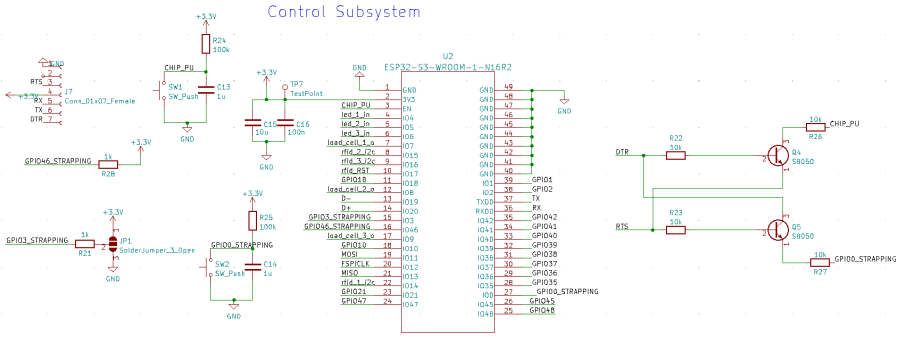

Figure 7: Control Subsystem Schematic

2.1.4 Power Subsystem

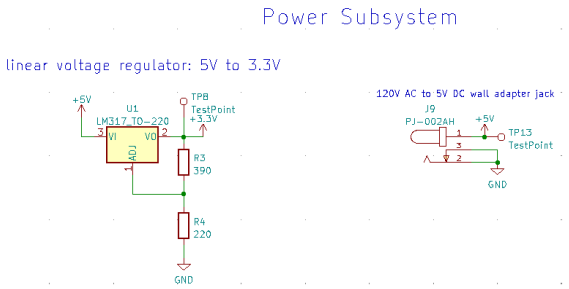

Figure 8: Power Subsystem Schematic

2.1.5 Hardware Interface Subsystem

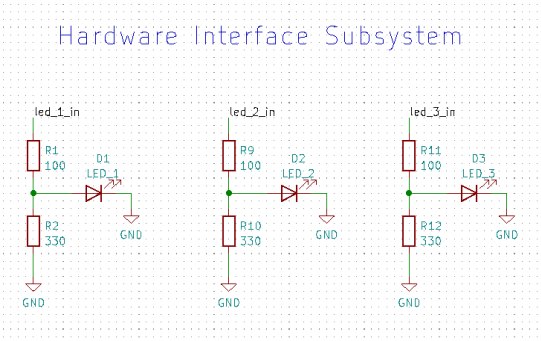

Figure 9: Hardware Interface Subsystem Schematic

2.2 Design Description and Justification

2.2.1 Linear regulators (LM317) Tolerance Justification

Table 1: Current Analysis

Part	**Worst Case Current Draw @ 3.3V**	**Comment**
ESP32-S3	500mA	Page 56 of the datasheet [6]
LED	120mA	Datasheet [8]
Total	620 mA	

Table 2: Variable Values for Formula Estimating Junction Temperature

Variable	**Value**	**Comments**
$\max(T_j)$	125 C	Maximum operating junction temperature of LM317TO-200 package [11]
i_{out}	620 mA	Maximum current draw of components on 3.3V power
vin	5 V	Output of wall power output adapter
vout	3.3 V	Operating voltage of components
θ_{jc}	5 C/W	Junction-to-case thermal resistance of LM317TO-200 package found in datasheet [11]
θ_{ca}	32.9 C/W	The regulator datasheet conveniently specifies θ_{ja}= 37.9 C/W [11]
T_a	30 C	Let's assume for a warm board

The formula below is used to estimate the junction temperature:

$$P_D = iout \times (vin - vout)$$
$$T_{ja} = P_D \times \theta_{ja}$$
$$\Rightarrow T_j - T_a = P_D(\theta_{jc} + \theta_{ca})$$
$$\Rightarrow T_j = P_D(\theta_{jc} + \theta_{ca}) + T_a$$
$$\Rightarrow T_j = i_{out}(vin - vout)(\theta_{jc} + \theta_{ca}) + T_a \quad (2.2.1)$$

Formula in regards with system values:

$$T_j = i_{out}(vin - vout)(\theta_{jc} + \theta_{ca}) + T_a = 620mA(5 - 3.3)(5 + 32.9) + 30 = 69.95\,C$$
$$T_j \approx 69.95C < max(T_j)$$

This process of estimating T_j(Eq. 2.2.1) is not meant to be very exact. Since the estimated temperature is 30 degrees below the maximum, the LM317TO-200 package is an appropriate linear regulator to use.

2.2.2 LM317 Resistor Analysis

The wall power adapter brings the 120V AC down to 5V DC. Other than the LEDs, everything uses 3.3V. To bring down the 5V to 3.3V, the LM317 linear voltage regulator is used. Equation 2.2.2 shows the equation used to find R1 and R2 for the schematic shown in Figure 10. Through trial and error, we found R1 and R2 to be 100Ω and 220Ω respectively.

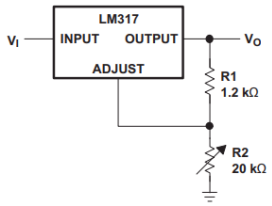

Figure 10: LM317 Regulator Circuit With Minimum Program Current [11]

$$V_{OUT} = V_{REF}\left(1 + \frac{R2}{R1}\right) \tag{2.2.2}$$
$$1.25\left(1 + \frac{220}{100}\right) = 4V$$

The calculated Vout (Eq. 2.2.2) of the linear voltage regulator was 4V, however when we measured the actual output of the LM317, we found it to be 3.5V which is within the tolerance of 3.3 ± 0.3V for the ESP32, HX711, and RFIDs.

2.2.3 LED Voltage

The LEDs operating voltage range is 2.3 ± 0.3V according to the datasheet [8]. A resistor voltage divider is used to step down the 3.5V received from the ESP32 to 2.5V needed by the LEDs (Eq. 2.2.3). R1 and R2 were found through trial and error.

$$2.5 = 3.5 \times \frac{R_2}{R_2 + R_1} \tag{2.2.3}$$
$$R_1 = 100\Omega \quad R_2 = 330\Omega$$

2.2.4 Load Cells Calibration Factors

To set up each load cell to read accurate readings, three individual calibration factors are needed. To obtain an accurate calibration factor (Eq. 2.2.4), three readings of the same weight object were used.

$$Calibration\ Factor = \frac{\frac{(reading\ 1 + reading\ 2 + reading\ 3)}{3}}{known\ weight\ of\ object} \tag{2.2.4}$$

Load cell 1 calibration factor = 763.326586
Load cell 2 calibration factor = 828.224932
Load cell 3 calibration factor = 808.082656

2.3 Design Alternatives

The original resistor values for the LM317 were 390Ω for R3 and 220Ω for R4 as shown in Figure 8. When testing the output voltage of the LM317, we found that Vout was 1.7V. The ESP32, RFIDs, and load cells all need 3.3V ± 0.3V. Using Equation (2.2.2) and trial and error, we found that changing R3 to 100Ω and leaving R4 unchanged gives an output voltage of 3.5V, which is within the tolerance of all components on our PCB.

Based on the MFRC522 datasheet [9], the antennas have a reading range up to 50mm. The original design implemented a metal casing around each spot to block overlapping RFID signals. We assumed that the metal casing would allow the antenna and readers within the casing to operate normally based on our design. Through testing, we found irregularities in the RFID readers recognizing tags. To fix this issue, we removed the metal plating behind the RFID readers.

The initial implementation read RFIDs at each spot in a loop. If a particular spot had a valid RFID, it would take a measurement from that spot's load cell. The RFIDs tag numbers at each spot were read correctly, but their load cell readings were incorrect. They constantly output -234 and 668 regardless of the items placed on the load cells. To rectify the issue, we separated the load cell readings and the RFID readings. First, the load cells are set up, then read at three spots and then powered down. Second, the RFIDs are set up, read at the three spots, and then shutdown. The loop continues by going back to the load cells. The load cell clock was initially set to the same clock as the RFIDs. This caused the constant -234 and 668 readings. The RFIDs use the SPI protocol while load cells do not. We addressed this by changing the load cells clock pins from GPIO12 to GPIO36. Once the clock pins were altered, the RFIDs were not accurately read. Through trial and error, we found changing the RFIDs reset pins from GPIO17 to GPIO38 fixed the issue.

Our initial schematic design used the operating voltage of 5V that was found in the HX711 datasheet [2]. However, when we ran the calibration Arduino code, the load cells were not recognized by the program. After consulting online tutorials [5], the recommended operating voltage of the HX711 was 3.3V.

The HX711 depends on the ESP32's CLK signal to output the weight from the load cells bit-by-bit. In the original design, these two components were operating at different voltages, ESP32 at 3.3V and HX711 at 5V. Any communication between them required logic level shifters. The logic level shifters would allow us to convert those voltages while still keeping the original data. Since the revised design has both components operating at the 3.3V, the logic level shifters were removed from our design.

3. Cost and Schedule

3.1 Cost

Computer Engineer: $56/hr [3] Electrical Engineer: $49/hr [4] Average: $52.5/hr

Number of work hours per person: 8wks * 30hrs/wk = 240 hrs/person (3.1.1)

Salary of one member: 52.5 * 2.5 * 240 = $31,500 (3.1.2)

Total salary across 3 members: $31,500 * 3 = $94,500 (3.1.3)

Cost of physical design material: $40 Machine Shop Labor: $20.3/hr [13]

Total Cost from Machine Shop: $40 + ($20.3/hr * 3hrs) = $100.90 (3.1.4)

This product is not currently commercially viable yet.

Table 3: Parts Cost

#	Item	Count	Unit Price	Total Cost	Source
1	Load Cells (x4)	1	-	19.99	Amazon
3	RFID readers	4	5.14	20.56	Digikey
5	HX711 (x5)	2	10.66	21.32	Walmart
6	Logic Level Shifter	4	0.62	2.48	Digikey
7	Logic Level Shifter	4	0.63	2.52	Digikey
8	Power Barrel Connector	2	0.62	1.24	Digikey
9	Resistors 100 ohm	28	0.02	0.64	Digikey
10	Resistors 220 Ohm	6	0.10	0.60	Digikey
11	Resistors 390 Ohm	6	0.10	0.60	Digikey
12	Power Supply Adapter	1	-	5.49	Amazon
13	Freenove ESP32 S3 Wroom Cam Board	1	-	19.90	Amazon
14	SMD Switch	5	0.17	0.85	Digikey
15	Logic Level Shifters for Breadboard	-	-	7.99	Amazon

	ECE Electronic Supply Shop				
#	Item	Count	Unit Price	Total Cost	Source
1	Resistors 330 Ohm	32	-	-	ESS
2	Resistor 1k Ohm	28	-	-	ESS
3	Resistor 10k Ohm	36	-	-	ESS
4	Resistor 100k Ohm	28	-	-	ESS
5	Capacitors 0.1 uF	48	-	-	ESS
6	Capacitors 1 uF	28	-	-	ESS
7	Capacitors 10 uF	48	-	-	ESS
8	LM320T220	4	-	-	ESS
9	ESP32-S3 Wroom	4	-	-	ESS
10	SS8050-G	4	-	-	ESS
11	Switch Tactile	7	-	-	ESS
	TOTAL			**103.18**	

3.2 Schedule

Table 4: Schedule

Week	Task	Person
2/12 - 2/18	Finish design document	All
	Finalize parts	Anju
	Meet with Gregg to start building	All
	Start board design (development board)	Anju
	Start designing PCB	Anju
	Wireframing app	Sanjana
2/19 - 2/25	Continue designing PCB	Anju
	Start initial app design	Sanjana

	Order parts	Nynika	
	Prepare for design review	All	
2/26 - 3/3	Continue app development	Sanjana	
	Finish first round of PCB design	Anju	
	Design review w/ professors and TA	All	
3/4 - 3/10	1st order PCB	Anju	
	Continue app development	Sanjana	
3/11 - 3/17	Spring Break		
3/18 - 3/24	Functional app DONE	Sanjana	
	Start soldering	Anju	
	Connect app to ESP32 via Wifi	Sanjana	
	Start load cell + RFID functionality	Sanjana	Anju
	Start app notifications	Nynika	
3/25 - 3/31	3rd order PCB	Anju	
	Complete LED functionality	Sanjana	
	Firebase Connection Via Arduino	Sanjana	
	Complete load cell functionality	Sanjana	
4/1 - 4/7	4th order PCB	Anju	
	Arduino Code Debugging	Sanjana	Anju
4/8 - 4/14	5th order PCB	Anju	
	Run thorough test to ensure app, LED, RFID, and load sensors all work properly	Sanjana	Anju
4/15 - 4/21	Mock demo	All	
	Begin final paper	Sanjana	Anju
4/22 - 4/28	Finished app notifications	Nynika	

	Final demo	All	
	Complete final presentation slides	All	
	Mock final presentation	All	
4/29 - End	Final presentation	All	
	Final paper (due 5/1)	Sanjana	Anju
	Lab notebook (due 5/2)	All	

4. Requirements and Verifications

4.1 User Subsystem

Table 5: User Subsystem R&V Table

Requirements	Verification
The Ingredient Dashboard can track up to 3 spices.	The Ingredient Dashboard on the mobile application displays three spices with individual progress bars indicating lower weight threshold and current weight progress.
Ingredient Dashboard screen displays an error message for invalid numbers entered into the Lower Threshold subfield. Lower Threshold must be a whole number in the range of 0 - 500 grams.	For spice 1, set the lower threshold as 100 grams. The Ingredient Dashboard screen should not show an error message and the progress bar should have a marker at 100 grams. Now, set the lower threshold for spice 1 as 2.5 grams. The Ingredient Dashboard shows an error message and its progress bar turns gray.
Each spice on the dashboard visualizes the recorded weight of the spice through a progress bar. The progress bar ranges from 0 - 500 grams with a marker indicating the user specified lower weight threshold. When the recorded spice weight falls below the lower weight threshold, the filled portion of the progress bar turns red. Otherwise it remains green.	A progress bar only shows up if there are spices added to the Ingredient Dashboard. The spice's weight will be shown as a progress bar with a marker indicating its inputted lower weight threshold. Place a spice container that is above the lower weight threshold in a spot on the device. Withdraw the spice from the container until it is below its threshold. The filled portion of the progress bar should turn red after 30 seconds.
Ingredient Dashboard receives updates on spice weight every 30 seconds. The Ingredient Dashboard data is stored in an online database.	Have a container already in a spot on the device and make sure it has already been added to the Ingredient Dashboard. Remove or add spice into the container. After 30 seconds the progress bar in the Ingredient Dashboard should reflect the change in weight.
Dynamic grocery list synchronizes with the Ingredient Dashboard and only spices that are	Have a container already in a spot on the device and make sure it has already been added to the Ingredient Dashboard. Remove

below the lower weight threshold are displayed.	spice from the container. After 30 seconds the spice should show up in the Grocery List screen. Have a container already in a spot on the device and make sure it has already been added to the Ingredient Dashboard. This spice needs to already be below its lower weight threshold and the Grocery List screen should include that spice. Add the spice to the container so that its weight goes above its threshold. After 30 seconds the spice should disappear from the Grocery List.
A notification is generated and the spice name and Google Maps view will appear in the Grocery List screen when a measured weight falls below the user set threshold.	For spice 2, set the lower threshold to a value lower than the current tracked weight in the app view. A notification should appear in the notification center of the phone immediately. The name of the spice and a Google Maps view of nearby grocery stores should appear in the Grocery List.

4.2 Sensing System

Table 6: Sensing Subsystem R&V Table

Requirements	Verifications	Results
One Load Cell + Firebase Connection	Load Cell: Measure out 50 grams of spice using a scale (baseline weight). Place the 50 grams of spice on the load cell to see if the value outputted is the same as the baseline weight with a resolve of 10 grams. Load Cell + Firebase Connection: Send data from load cell to ESP32. Send the collected data from ESP32 to the Firebase database. Check to see if collected data is present in the database.	Scale reads 50 grams. Load cell reads 49.2 grams for the weight of the item.
Reads multiple load cells consecutively.	Put three different weights in each spot. Move them around to ensure that each load cell measures the appropriate	

	weight.	
Only the 3 valid RFID tags will be read and recognized by the device.	To ensure only valid tags are read, a fourth tag will be used as an edge case. Place an invalid card at a spot, the serial monitor prints out the tag number, the spot the card is in and that it is an invalid card.	Valid tags: C683455AE D58FFE29 D6EBEFAD Invalid tag: 968EEAAD
Reads tags despite one or multiple spots not having tags.	Place 1 tag at spot 1 and keep spot 2 and 3 without any tags. The serial monitor prints out the tag number, the spot the card is in and whether the tag is valid or invalid.	
Vcc = 3.3V± 0.3V [2] [9]	Take a multimeter and attach the positive probe to Vcc of RFID and the negative probe to GND. The multimeter needs to read 3.3V± 0.3V. Take a multimeter and attach the positive probe to Vcc of the HX711 and the negative probe to GND. The multimeter needs to read 3.3V± 0.3V.	RFID and HX711 each read 3.5V.
Load cell reading is displayed on serial monitor for valid RFID tag at particular spot.	Place a valid RFID tag and weight at any spot. The serial monitor prints out the tag number, the spot the card is in, confirms the tag's validity, and weight of the item in grams. Place an invalid RFID tag and weight at any spot. The serial monitor prints out the tag number, the spot the card is in, and that the tag is invalid. No weight is shown.	
Load cell readings should be reflected in Firebase database	Place a valid RFID tag and container of spice at any spot. Each container is tied to a specific RFID tag. The weight of the container should be reflected in both the app's Ingredient Dashboard page and Firebase database.	

4.3 Control Subsystem

Table 7: Control Subsystem R&V Table

Requirements	Verification
The data pulled from the HX711 chip to the MCU is transferred to the Firebase database every 30 seconds.	Have a container already in a spot on the device and make sure it has already been added to the Ingredient Dashboard. Remove or add spice into the container. After 30 seconds the progress bar in the Ingredient Dashboard should reflect the change in weight.

4.4 Power Subsystem

Table 8: Power Subsystem R&V Table

Requirements	Verification	Results
Wall adapter outputs 5V ± 0.25V	Connect the positive probe of the multimeter to pin 1 of the barrel jack connector and the negative probe to pin 2. The multimeter should read 5V ± 0.25V.	Vout measured 5.03V.
LM317 outputs 3.3V ± 0.3V [11]	Connect the positive probe of the multimeter to Vout of the LM317 and the negative probe to GND. The multimeter should read 3.3V ± 0.3V.	Vout measured 3.5V.

4.5 Hardware Interface Subsystem

Table 9: Hardware Interface Subsystem R&V Table

Requirements	Verification	Results
The LED switches on at a spot when the spice's weight is below its lower threshold.	Place a valid RFID tag and container of spice at any spot. The container of spice should weigh below its lower threshold. The Arduino code pulls the spice's lower threshold from the Firebase database. If the lower threshold is greater than the load cell reading at that spot, the LED at the spot is switched on. The code also updates the runningLow flag of the spice to True in Firebase. Add spice to the container such that its weight is above its lower threshold. The Arduino code pulls	

	the spice's lower threshold from the Firebase database. Since the lower threshold is lower than the load cell reading at that spot, the LED at the spot is switched off. The code also updates the runningLow flag of the spice to False in Firebase. Replace the valid tag with an invalid tag. If the spot's LED was previously on, the LED switches off.	
Vcc = 2.3V± 0.3V [8]	Take a multimeter and attach the positive probe to the positive lead of the LED and the negative probe to GND. The multimeter needs to read 2.3V± 0.3V.	Vcc of LEDs measured are 2.5V

5. Conclusion

5.1 Accomplishments

Our project successfully pulls weight data captured from the load cells every 30 seconds for the 3 spices tracked. The weight data is sent by load cells to the ESP32 and then stored within Firebase. The data in Firebase is presented to the user through the app. To maintain accuracy, the weight of the spice container and its RFID are subtracted from the load sensor's total measurement to calculate the actual weight of the spice. The app is organized using three screens: Home, Ingredient Dashboard, and Grocery List. The Home screen provides instructions to the user on how to use the app. The Ingredient Dashboard allows users to input spice names and their lower thresholds, and presents weight data as progress bars. The Grocery List displays spices that have run below its lower weight threshold. It also houses a Google Map view to showcase grocery stores that are 0.5 miles away from the user's current location. The app provides notifications for users when spices appear in the Grocery List screen. The spice containers are tied to one of the three RFIDs, allowing users to place the spices in any of the three spots. The RFID keeps track of the container's location, and allows for the right comparison between the spice's weight and its lower threshold. The LEDs are successful in providing a visual representation for spices that have run below the user specified lower weight threshold. They remain turned on until the spice weight goes above the lower threshold, the spot is empty, or an invalid RFID tag is put in. The final PCB design was successful in connecting the load cells, RFIDs, and LEDs to the ESP32. The project achieved all the high level requirements.

5.2 Uncertainties

The initial idea for the PCB was to replicate the HX711 module. While researching the components of the HX711 in its datasheet [2], two of the resistor values were unspecified. We attempted to try and find the R1 and R2 values, however we were unsuccessful. We began by trying to find the voltage across each of the resistors when Vdd was 5V. The voltage across R1 was 22 mV and the voltage across R2 was 18 mV. The voltages across each of these resistors did not make any sense in terms of the circuit shown in the datasheet. We then tried to figure out how to measure the current through each resistor, which would allow us to use Ohm's Law to find the resistor values. The problem with this was that it would require us to desolder each surface mount resistor from the modules we had received, and then solder them to a different PCB to then use a multimeter to find the current. Since we had only four HX711 modules, we wanted to keep the fourth as a spare for emergencies as the load cell wires were very temperamental, easily frayed, and short, making soldering of the wires to the module very difficult. Our second attempt was to use the mulitmeter's two wire resistance function to find the resistor values. This method showed R1 as Overload and R2 around 2.7 MΩ.. While we were able to get a value for R2, we still did not have one for R1. As the final deadline approached, we decided to use the premade modules instead of recreating our own.

5.3 Future Work

Based on the MFRC522 datasheet [9], the antennas have a reading range up to 50mm. We decided to use metal casings to block RFID signals between the three readers. We assumed that the metal casing would allow the antenna and readers within the casing to operate normally. Through testing, we found that the metal interfered with the antennas and readers. To fix this issue, we removed the metal plating behind the RFID readers. In the future, the physical design would replace the metal casing with a plastic casing.

This project poses a fire hazard with its exposed wires and PCB as well as the lack of thermal sensors. These problems can be eradicated by placing the PCB in an enclosure and placing the wire connections within the physical design itself. Thermal sensors should be added at the various sensors to constantly monitor the temperature. If any of the thermal sensors were to read a temperature higher than the optimal range, the ESP32 would shut off power to the entire design and the app would notify the users of the situation.

In the future, we would recreate the HX711 module for the load cells on the PCB.

Currently, new data from the database is displayed on the app only after the screen gets refreshed. To make the app more responsive, the app should trigger displaying new data whenever the Firebase database gets updated.

5.4 Ethical Considerations

The project complies with the IEEE [7] and ACM Code of Ethics [1]. The current design of this product contains safety issues. Due to time and Machine Shop constraints, the product has exposed wires and an uncovered PCB. In the final iteration of the product, the PCB, all wires, and LEDs would be enclosed within the physical design. The addition of thermal sensors will safeguard against potential fire hazards caused by overheating by disconnecting the circuit from power.

This product does not pose a risk for cross contamination. The spices are placed within plastic airtight containers. Therefore if any containers fall down or move, the spices are safely contained within the container. If any spices fall onto the device, a user can use a wipe to remove the mess.

Since the app uses a user’s location, there is an ethical and privacy issue. The app will require access to the user’s location, however, the app will not be using the user’s information for malicious purposes. Following Google’s Privacy and Terms [10], our application will utilize Google’s robust security measures to protect all user data. This aligns with the ACM Code of Ethics stating, "Computing professionals should only use personal information for legitimate ends and without violating the rights of individuals and groups [...] taking precautions to prevent

re-identification of anonymized data or unauthorized data collection [...] and protecting it from unauthorized access and accidental disclosure" [1]. As this app will only be available on Apple devices, the user will have the option to choose their privacy setting in their Apple device settings. Additionally, users can opt in and out of location sharing, ultimately keeping the power in the user's hands [10].

References

[1] "ACM Code of Ethics." acm.org. Accessed Feb. 2, 2024. [Online.] Available: https://www.acm.org/code-of-ethics

[2] AVIA Semiconductor. "24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales." HX711 datasheet.

[3] "Computer Engineering Salary in Illinois." ZipRecruiter. Accessed: Feb.18th, 2024. [Online]. Available: https://www.ziprecruiter.com/Salaries/Computer-Engineering-Salary--in-Illinois

[4] "Electric Engineering Salary in Illinois." ZipRecruiter. Accessed: Feb.18th, 2024. [Online]. Available: https://www.ziprecruiter.com/Salaries/Graduate-Electrical-Engineer-Salary--in-Illinois

[5] “ESP32 with Load Cell and HX711 Amplifier (Digital Scale).” Random Nerd Tutorials. Accessed March 13th, 2024. [Online]. Available: https://randomnerdtutorials.com/esp32-load-cell-hx711/

[6] Espressif. "ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U", ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U datasheet, Sept. 2021 [Revised Nov. 2023]

[7] "IEEE Code of Ethics." ieee.org. Accessed: Feb. 2, 2024. [Online.] Available: www.ieee.org/about/corporate/governance/p7-8.html

[8] Liteon. "Lite-On Electronics, Inc." LTL-4223 datasheet.

[9] NXP Semiconductors. “MFRC522 Contactless Reader IC.” MFRC522 datasheet. [Revised May 2007].

[10] “Privacy and Terms.” Google. Accessed: April 24th, 2024. [Online]. Available: https://policies.google.com/privacy?hl=en-US

[11] Texas Instruments. "LM317 3-Terminal Adjustable Regulator." LM317 datasheet. Sept. 1997 [Revised Apr. 2020].

[12] T. Ozbun. "Grocery shopping: U.S. consumers' weekly trips per household 2006-2022." *statista.com*. Accessed: Feb. 14, 2024. [Online.] Available: https://www.statista.com/statistics/251728/weekly-number-of-us-grocery-shopping-tripsper-household/

[13] “Woodworking Salary in Illinois.” ZipRecruiter. Accessed: April.30th, 2024. [Online]. Available: https://www.ziprecruiter.com/Salaries/Woodworking-Salary--in-Illinois