ECE445
SENIOR DESIGN PROJECT LABORATORY
FINAL REPORT

PawFeast: Food on Demand

Team No. 30
Arash Amiri (arasha3@illinois.edu)
Omkar Kulkarni (onk2@illinois.edu)
Kathryn Thompson (kyt3@illinois.edu)

TA: Aishee Mondal
Professor: Yang Zhao
Date: May 6, 2025

Abstract

The increasing reliance on automation in home systems and access to electronic tools offers a promising platform for more intelligent pet care. This project aims to combine everything we have learned throughout UIUC and make a purposeful project out of it. We introduce a smart pet feeder system that addresses common feeding issues such as irregular feeding times, overfeeding, underfeeding, and food contamination. The unit integrates a microcontroller control board (ESP32), lots of sensors, a load cell, RFID, and a web UI to support responsive automation and real-time monitoring. The product includes a stepper motor-controlled food dispenser with timed controls, multi-pet family RFID access, and a covering lid. Power comes from a custom battery pack with buck converters and regulator circuits that deliver safe and stable voltage levels to all subsystems. Tolerance analysis is also carried out to ensure power reliability in loaded states, and verification procedures ensure that the system meets high-level functional specifications. This paper demonstrates a modular and scalable solution to automated pet care systems, with potential applications in home and veterinary settings.

Table of Contents

1. Introduction

1.1. Problem

All household pet owners must remember to feed their pets—whether they are birds, guinea pigs, rabbits, dogs, or cats—at set periods during the day. But there can be times when people forget to feed their pets, double feed them, or have trouble feeding them on schedule. This can be due to poor communication between family members or even occur when pet owners come home late from work or prior commitments. During these times, pets either overeat, go hungry, or eat irregularly. As a result, timer-based pet feeders have been developed to release food at set intervals. When food sits out for extended periods of time, the risk of the food becoming stale or getting infected with bugs increases drastically [6.1].

If an insect is near your food, it is a common reaction to remove or swat it away, but that mechanism is not present for the pet's food bowl, especially when the family is not at home. Another issue is that smarter pets can nose into the food storage and overeat or eat something that their bodies will reject (like chocolate for dogs). With a set schedule to limit food dispensation throughout the day, the pet will not take advantage of either a careless overpour of food or of an unmanned food station. Furthermore, data on how much food was dispensed and eaten will help track and optimize the pet's eating habits.

1.2. Solution

We are seeking to solve this problem by having timers coupled with a button and an RFID component to release food, thus preventing overeating. Given all of these conditions, the dispenser will release the pet food. This ensures that the pet is not only fed on time, but that the food is not polluted when the pet goes to eat; additionally, a chime will go off signaling to the pet that it is time for them to eat [6.2]. We will incorporate a notification system to alert the owner(s) whether the pet has eaten or when the food store has low levels of food.

If the pet were to not eat all the food at once and leave the food dispenser, the pressure sensor will tell the dispenser to cover the food until the pet returns, ensuring freshness and preventing bugs.

1.3. Functionality

Over this semester, we have created an automated on-demand pet food dispenser. This pet feeder dispenses food at user-preset times for multiple pets using RFID tags on the pets' collars for identification. The dispenser requires both the presence of the RFID tag and the press of a button to dispense food. Given both of these events, a lid will slide off the bow, allowing food to be dispensed. A load cell located underneath the food bowl will send weight readings to the brain specifying the capacity of the bowl. This alters how much food is dispensed, accounting for food already present in the bowl. After the pet finishes eating, the load cell reads the weight of the food bowl and sends it to the User Interface. This is for logging purposes and allows the user to track their pet's feeding habits. Users are able to create multiple pets and feeding times for the pets. The pet feeder is able to reflect these changes and dispense food within 30 seconds of any update via the User Interface. Presence sensors will be used to notify the user of the current amount of food in the food store. This pet feeder features dual power modes, battery power for portability, and a wall outlet for sustained, long-term use.

1.4. Block Diagram

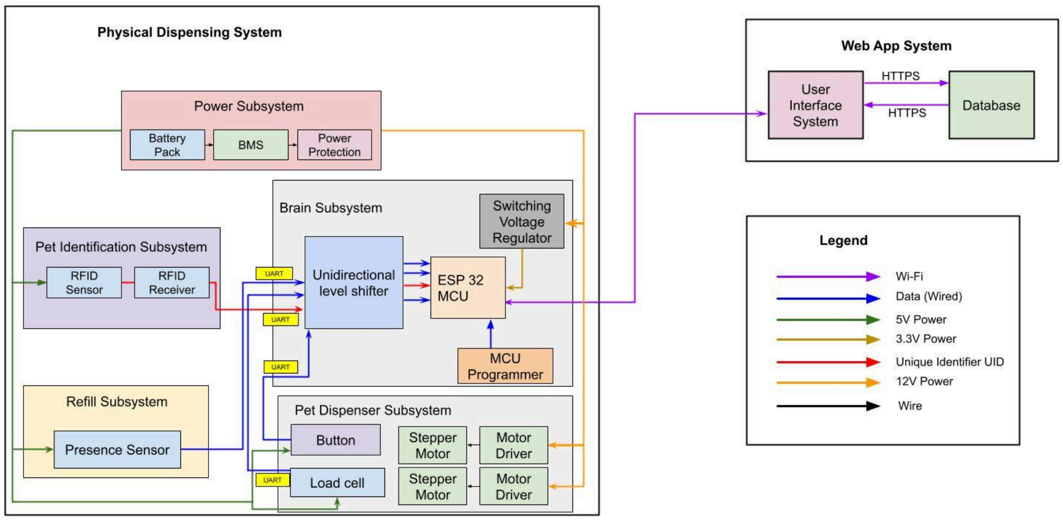


Fig. 1: Block diagram of system architecture

1.5. High-Level Requirements List

For this project to be considered successful, our project must meet the following requirements:

1.5.1. The Pet Food Dispenser must not dispense food before user preset times, such as 10 AM and 6 PM, daily, with a 10-minute tolerance.

1.5.2. The Pet Food Dispenser must dispense food into the designated food bowl upon detecting the presence of the pet via the button and the 2+ RFID tags on the pet's collar within 2 minutes.

1.5.3. The user interface must update within 5 minutes of an event on the application we create detailing the pet's feeding status, low food store (20% & 10%), and unfinished meals.

2. Design

2.1. Design Procedure

We separated this project into 6 subsystems: Brain, Power, Food Dispensal, Food Store, User Interface, and RFID. Of these subsystems, Omkar was the primary for the Brain and Power, Arash was the primary for the Food Dispensal and Food Store, and Kathryn was the primary for User Interface and RFID. While each team member was the primary for 2 subsystems, each team member helped debug, implement, and test other subsystems. A brief description of each subsystem is provided below, however, we will go more in-depth in the following sections.

The *Brain* handles all product firmware and programming.

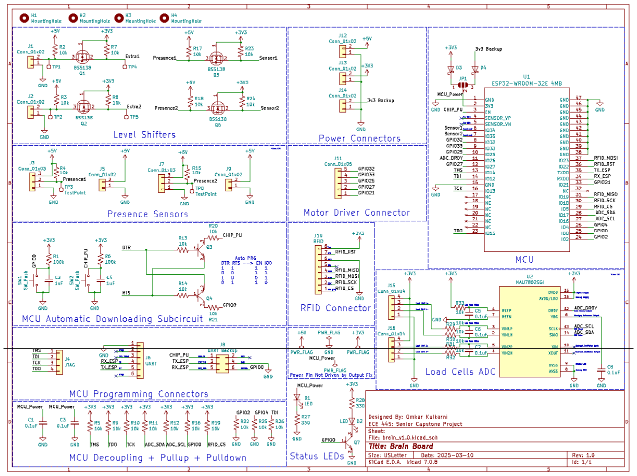


Figure 2. Brain Schematic Capture

The *Refill Subsystem* was responsible for having pairs of Break Beam Sensors to detect food storage levels and send findings to the MCU.

The *Pet Dispensing Subsystem* was split into a load cell to handle how much food is needed and left, and also a motor to handle the dispensing of the food itself. There was also a driver and stepper motor to open and close the food lid, and a button to detect pet interaction.

The *Power Subsystem* made sure that we had easy access to various voltages: 12V, 5V, and 3.3V throughout the system.

The *User Interface* handles all communications between the ESP32 and the web server. This is where the user is able to create pets and add feeding times to the pets that determine if the pet feeder should dispense food. The user is also able to see the history of their pet feeder events here.

The *RFID Subsystem* is where we handle the identification of pets. We used the RC522 breakout board for this project to act as our tag reader.

2.2. Design Details

There have been no block-level changes made to our design throughout the course of the project. When planning the design, we thought through all possible design choices carefully to reduce the number of changes we would have to make down the line. Luckily, our project worked in the first iteration as initially designed. Throughout the project, we had to make zero design changes and sacrifices. We stuck to every component of the block diagram and the overall project proposal. We never ran into issues where parts went out of stock or issues with the machine shop, so we never had to fundamentally alter the desired functionality. The machine shop helped make sure every technical sensor and component physically worked with each other and that the project was attainable. Given that everything used was within the power ratings we initially went about, we never worried about not having enough power to supply the project or use external power supplies.

2.3. Subsystem Overview

2.3.1. Subsystem 1: Refilling Food Store

The subsystem for checking if the food store is empty will include a through-beam optical sensor that is able to determine whether an object is present—this will serve to detect whether the food inside of the container is low and needs to be refilled. When the food store reaches a certain level, a signal will be sent to subsystem 6 (the brain), which will send an automated message to notify the owner to refill the food store.

2.3.2. Subsystem 2: Power System

The power system will supply energy to all of the various subsystems, including the microcontroller of subsystem 6 (the brain), the stepper motor driver in subsystem 3, and the various sensors in subsystems 1 (refilling food) and 3 (food dispenser). The low voltage buses will be 3.3V, 12V, and 5V, all of which will be powered by a combination of rechargeable lithium-ion batteries with voltage regulators and buck converters. This compact, custom 3s2p battery pack will allow the pet dispenser to be portable, even for family vacations or road trips—more importantly, it will provide a steady current, even when the pack voltage drops (unlike standard AA/AAA cells). The 12V to 3.3V regulator chip will be Monolithic Power Systems's MP2315SGJ-Z, which can take a 12V input and output up to 2.5A [6.7]. The 12V to 5V regulator chip will be Monolithic Power Systems's MP2338GTL-Z [6.6]. The 12V input into the motor driver IC circuit will come straight from the battery pack. There will also be a backup supply voltage of 5V from a typical barrel jack wall outlet, integrated into the brain board. This will implement the AMS1117 LDO, which is perfect for 5V to 3.3V power conversion. Just to be clear, this 5V barrel-jack addition will just be included on the brain board as a backup power subcircuit in case of power shutdown.

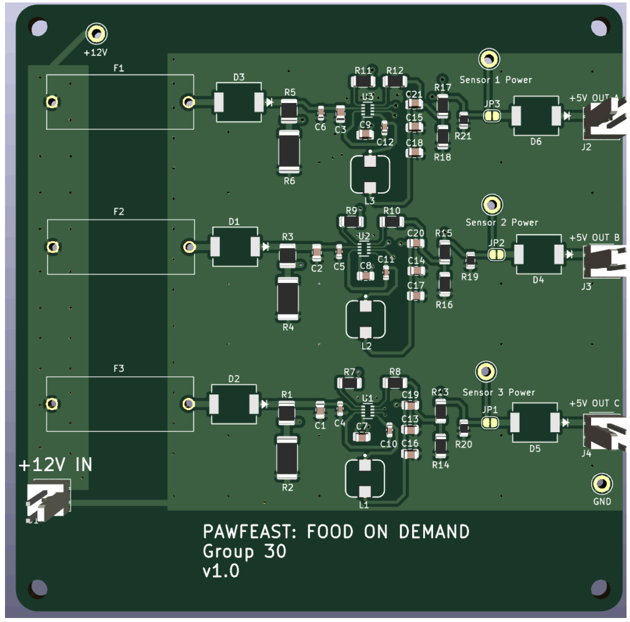


Figure 3: 3D View of Power Board

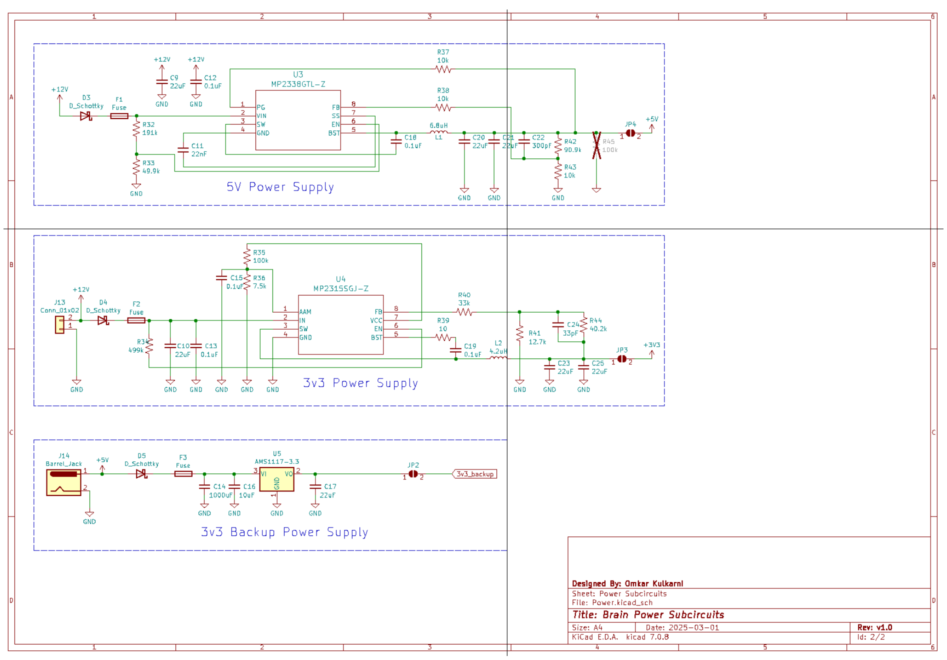


Fig. 5: Schematic Capture of Power Board

2.3.3. Subsystem 3: Food Dispenser

This subsystem will utilize a E Series Nema 17 stepper motor that will rotate a door to the food container when needed and allow a set amount of food through before rotating again to close the door [6.8]. The motor will be controlled by a stepper motor controller IC (DRV8825) with a custom control circuit that takes in 12V and outputs 3.3V to the motor. We will define hard cut-offs that will be coded for the maximum amount of food to be dispensed and specify times at which food may be dispensed during the day.

The main objective of this subsystem is to make the system an on-demand food system. We will utilize a large button for the dog to step on when hungry. When pressed, the system will start the motors to release food, on the condition that enough time has elapsed since it ate last. The pet would be trained to ask for food using this button. Only when the button is pressed and enough time has passed will food be dispensed. We had to go about designing a motor circuit board for the stepper motor driver.

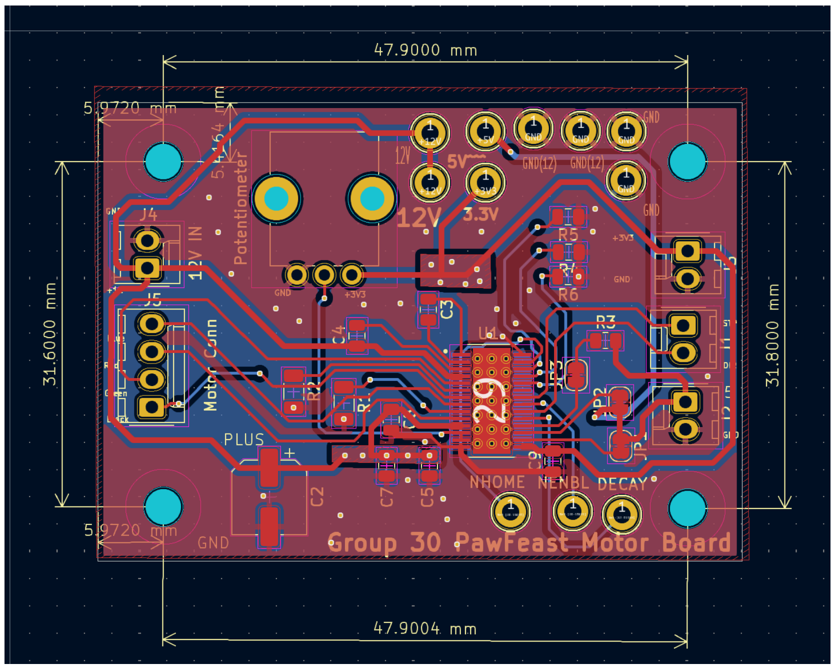


Figure 4: Motor Board PCB Layout

2.3.4. Subsystem 4: RFID for pet identification

We will have a receiver for a minimum of 2 HF RFID tags 9662 Long Distance Passive Alien H3. We will use the RC522 RFID chip for reading and writing to the RFID chips. The RFIDs would be used to signal which pet is using the dispenser, so the owner can utilize the same dispenser for one or more pets.

2.3.5. Subsystem 5: User Interface

The subsystem for notifications will be a web application that we will build that is able to connect with the pet feed dispenser system. For the application side, we will utilize Bottle for our backend, SQLite for our database, and HTML pages for our frontend to create a user-friendly method to check the pet's eating habits. This will notify users when the pet has been fed, if the food tank is low and requires refilling, and if the food was covered due to an empty bowl or a partially filled one.

2.3.6. Subsystem 6: Brain

The brain subsystem will take in inputs from all of the other sensor subsystems and output the appropriate signals to the user interface (for notifications) and the food dispenser motor driver. This system will also track the amount of time that has passed and sound a soft chime for when enough time has passed for the pet to be able to eat. The ESP32 WROOM-32E-4N microcontroller, known for its wifi connectivity and security measures, would have a set of I/O pins for taking in these signals. This MCU programmer circuit and other control-level circuitry—firmware or equivalent—would be incorporated onto this board along with a UART circuit to detect pet RFIDs (done with the UHF RFID reader JRD-4035).

2.4. Tolerance Analysis:

1 Power Dissipation vs Supply Power

The power subsystem needs to supply power to multiple sensors, drivers, and the microcontroller. The ESP32 microcontroller dissipates a maximum RF power of:

$$P_{\text{ESP32}} = 19.5 \text{ dBm} = 0.0981 \text{ W}.$$

It is supplied with 3.3 V and can support input currents of 600 mA to 750 mA, leading to a maximum supplied power of:

$$P_{\text{ESP32 supply}} = 3.3V \times 0.75A = 2.475W.$$

The reflective object sensor has a maximum power dissipation of:

$$P_{\text{reflective}} = P_{\text{phototransistor}} + P_{\text{diode}} = 150 \text{ mW} + 75 \text{ mW} = 225 \text{ mW}.$$

The Vishay Presence sensor dissipates:

$$P_{\text{Vishay}} = 5V \times 5 \text{ mA} = 10 \text{ mW}.$$

The RFID receiver operates at 3.3 V with a peak output current of 200 mA, giving:

$$P_{\text{RFID}} = 3.3V \times 0.2A = 0.66W.$$

The two motor drivers each take a supply voltage of 12 V and drive up to 1 A, resulting in:

$$P_{\text{motor}} = 12V \times 1A = 12W \text{ per motor}.$$

For two motors:

$$P_{\text{motors total}} = 2 \times 12W = 24W.$$

Two load cells each take 5 V with 10 kΩ pull-down resistors, giving the following from the combination of Ohm's Law and P=IV:

$$P_{\text{load cell}} = \frac{5V^2}{10k\Omega} = 2.5 \text{ mW per sensor}.$$

For two load cells:

$$P_{\text{load cells total}} = 2 \times 2.5mW = 5mW.$$

Summing up the total power consumption:

$$P_{\text{total}} = 2.475W + 0.225W + 0.01W + 0.66W + 24W + 0.005W = 27.375W.$$

The battery pack is a 3s2p configuration with 3.7 V nominal 3200 mAh cells, giving:

$$V_{\text{pack}} = 3 \times 3.7V = 11.1V,$$
$$Q_{\text{pack}} = 2 \times 3200mAh = 6400mAh,$$

which results in an energy capacity of:

$$E_{\text{pack}} = V_{\text{pack}} \times Q_{\text{pack}} = 11.1V \times 6.4Ah = 71.04Wh. \tag{1}$$

This means the battery can theoretically supply:

$$P_{\text{supply}} = 71.04W \text{ for one hour.}$$

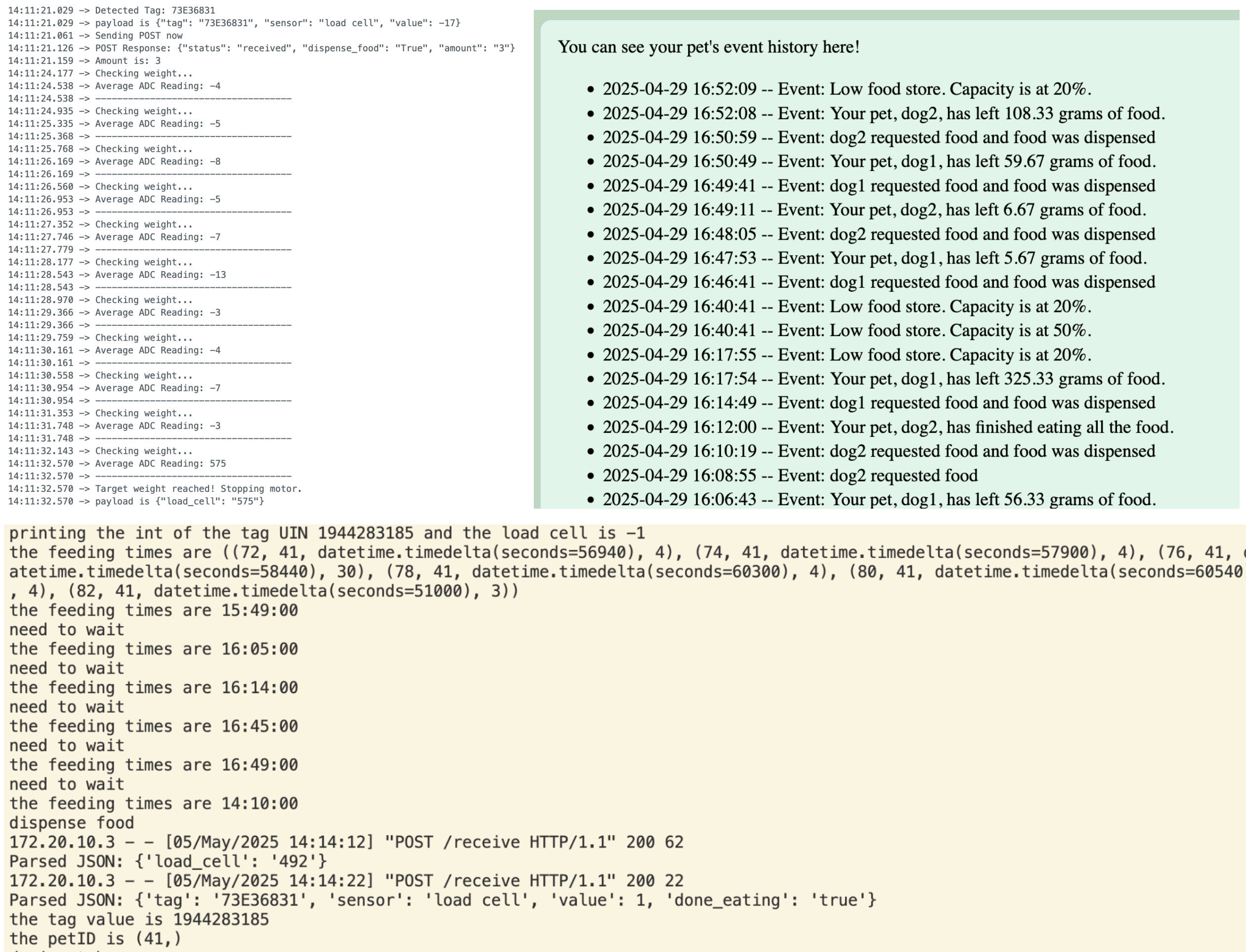
Since $P_{\text{supply}} > P_{\text{total}}$, the battery should be sufficient for at least an hour of continuous operation.

3. Verification

Each subsystem has a different verification process depending on our Requirements and Verification Table displayed in Appendix A. For refilling the food store, we delineated how two pairs of presence sensors would detect whether the food store was low. Our initial percentages of 10% and 20% were changed to 50% and 20% based on how the machine shop integrated them into the polycarbonate enclosure, and notifications are sent to the backend when the food has gone below these thresholds, based on Fig. 6 below. The fact that we can see all of these values means that the ESP32 microcontroller can detect the presence sensors and load cell readings, and send them to the backend after calculations. This satisfies the requirements for three other subsystems as well: the Food Dispensing Subsystem (which must read the weight of the food dispensed), the Brain Subsystem, which computes all logic, and the User Interface Subsystem (must be able to update the current status of the pet within five minutes).

We are able to see the RFID tag being read and that the tags are unique and identifiable through the screenshot of the user interface in Figure 6. Our initial requirement was that tags would be readable within a 6 inch range, however, we discovered that the reader was unable to read the value of the tags from more than a centimeter away. This was a result of poor antenna design.

The next requirements for the Food Dispensing Subsystem are that the lid must only open when the RFID and button are pressed together as mentioned earlier; the motor will then dispense the set amount of food depending on how much food is already in the bowl, if any. These requirements are not numeric and were verified through our successful demonstration.



```
14:11:21.029 -> Detected Tag: 73E36831
14:11:21.029 -> payload is {"tag": "73E36831", "sensor": "load cell", "value": -17}
14:11:21.061 -> Sending POST now
14:11:21.126 -> POST Response: {"status": "received", "dispense_food": "True", "amount": "3"}
14:11:21.159 -> Amount is: 3
14:11:24.177 -> Checking weight...
14:11:24.538 -> Average ADC Reading: -4
14:11:24.538 -> ------------------------------------
14:11:24.935 -> Checking weight...
14:11:25.335 -> Average ADC Reading: -5
14:11:25.368 -> ------------------------------------
14:11:25.768 -> Checking weight...
14:11:26.169 -> Average ADC Reading: -8
14:11:26.169 -> ------------------------------------
14:11:26.560 -> Checking weight...
14:11:26.953 -> Average ADC Reading: -5
14:11:26.953 -> ------------------------------------
14:11:27.352 -> Checking weight...
14:11:27.746 -> Average ADC Reading: -7
14:11:27.779 -> ------------------------------------
14:11:28.177 -> Checking weight...
14:11:28.543 -> Average ADC Reading: -13
14:11:28.543 -> ------------------------------------
14:11:28.970 -> Checking weight...
14:11:29.366 -> Average ADC Reading: -3
14:11:29.366 -> ------------------------------------
14:11:29.759 -> Checking weight...
14:11:30.161 -> Average ADC Reading: -4
14:11:30.161 -> ------------------------------------
14:11:30.558 -> Checking weight...
14:11:30.954 -> Average ADC Reading: -7
14:11:30.954 -> ------------------------------------
14:11:31.353 -> Checking weight...
14:11:31.748 -> Average ADC Reading: -3
14:11:31.748 -> ------------------------------------
14:11:32.143 -> Checking weight...
14:11:32.570 -> Average ADC Reading: 575
14:11:32.570 -> ------------------------------------
14:11:32.570 -> Target weight reached! Stopping motor.
14:11:32.570 -> payload is {"load_cell": "575"}
```

You can see your pet's event history here!

- 2025-04-29 16:52:09 -- Event: Low food store. Capacity is at 20%.
- 2025-04-29 16:52:08 -- Event: Your pet, dog2, has left 108.33 grams of food.
- 2025-04-29 16:50:59 -- Event: dog2 requested food and food was dispensed
- 2025-04-29 16:50:49 -- Event: Your pet, dog1, has left 59.67 grams of food.
- 2025-04-29 16:49:41 -- Event: dog1 requested food and food was dispensed
- 2025-04-29 16:49:11 -- Event: Your pet, dog2, has left 6.67 grams of food.
- 2025-04-29 16:48:05 -- Event: dog2 requested food and food was dispensed
- 2025-04-29 16:47:53 -- Event: Your pet, dog1, has left 5.67 grams of food.
- 2025-04-29 16:46:41 -- Event: dog1 requested food and food was dispensed
- 2025-04-29 16:40:41 -- Event: Low food store. Capacity is at 20%.
- 2025-04-29 16:40:41 -- Event: Low food store. Capacity is at 50%.
- 2025-04-29 16:17:55 -- Event: Low food store. Capacity is at 20%.
- 2025-04-29 16:17:54 -- Event: Your pet, dog1, has left 325.33 grams of food.
- 2025-04-29 16:14:49 -- Event: dog1 requested food and food was dispensed
- 2025-04-29 16:12:00 -- Event: Your pet, dog2, has finished eating all the food.
- 2025-04-29 16:10:19 -- Event: dog2 requested food and food was dispensed
- 2025-04-29 16:08:55 -- Event: dog2 requested food
- 2025-04-29 16:06:43 -- Event: Your pet, dog1, has left 56.33 grams of food.

```
printing the int of the tag UIN 1944283185 and the load cell is -1
the feeding times are ((72, 41, datetime.timedelta(seconds=56940), 4), (74, 41, datetime.timedelta(seconds=57900), 4), (76, 41, d
atetime.timedelta(seconds=58440), 30), (78, 41, datetime.timedelta(seconds=60300), 4), (80, 41, datetime.timedelta(seconds=60540)
, 4), (82, 41, datetime.timedelta(seconds=51000), 3))
the feeding times are 15:49:00
need to wait
the feeding times are 16:05:00
need to wait
the feeding times are 16:14:00
need to wait
the feeding times are 16:45:00
need to wait
the feeding times are 16:49:00
need to wait
the feeding times are 14:10:00
dispense food
172.20.10.3 - - [05/May/2025 14:14:12] "POST /receive HTTP/1.1" 200 62
Parsed JSON: {'load_cell': '492'}
172.20.10.3 - - [05/May/2025 14:14:22] "POST /receive HTTP/1.1" 200 22
Parsed JSON: {'tag': '73E36831', 'sensor': 'load cell', 'value': 1, 'done_eating': 'true'}
the tag value is 1944283185
the petID is (41,)
```

Fig. 6: User Interface Pet History and ADC Readings

The power subsystem has two main requirements. The first is that the 3V3 bus voltage must be within ±0.2V, and the 5V must be within ±0.5V—both of these aspects were satisfied based on the pictures of the multimeter below in Fig. 7. The second requirement about safety was satisfied with power protection components like Schottky diodes for reverse protection and 2.5A rated fuses for overcurrent protection.

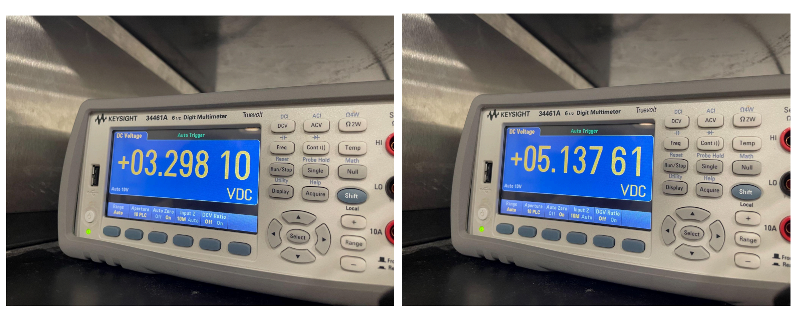


Fig. 7: Multimeter Voltages For the Two Low-Voltage Buses

4. **Cost Analysis**

4.1. Labor Costs

Assuming a salary of \$42/hour based on ECE post-graduate salary averages, the labor costs for our members over 10 weeks at 20 hours per week per member and 4 weeks at 10 hours per week per member is calculated as:

$$\text{Total Cost} = 3\times((20\text{ hours/week} \times 10\text{ weeks}) + (10\text{ hours/week} \times 4\text{ weeks}))\times\$42$$

$$= 3 \times (200 + 40) \times 42$$

$$= 3 \times 240 \times 42$$

$$= 30,240$$

The supply shop has not given us a specific quote for price, so we will assume it takes 15 hours to build our food store at an hourly rate of \$25/hour or \$375 total. This rate is based on Illinois state averages.

Thus, the total estimated labor cost is \$30,615.

4.2. Part Costs

ACQUIRED MISCELLANEOUS SENSORS AND PARTS					
Power					
Part Description	**Manufacturer**	**Part #**	**Quantity**	**Cost/Item**	**Cost**
BMS	N/A	N/A	1	9.99	9.99
Li-ion Batteries	N/A	N/A	3	14.88	44.64
12V-5V Voltage Regulator	N/A	MP2338GTL	5	2.54	12.70

Sensors and Motors					
Part Description	**Manufacturer**	**Part #**	**Quantity**	**Cost/Item**	**Cost**
Presence Sensor			1	5.95	5.95
Load Cell	DigiKey	1528-4541-ND	2	3.56	7.12
Nema 17 Stepper Motor	STEPPERONLINE	17HE15-1504S	1	10.88	10.88

Parts for Breadboard Demo					
Part Description	**Manufacturer**	**Part #**	**Quantity**	**Cost/Item**	**Cost**
Motor Breakout Board			1	15.95	15.95
Load Cell Breakout Board			1	5.95	5.95

Miscellaneous					
Part Description	**Manufacturer**	**Part #**	**Quantity**	**Cost/Item**	**Cost**
Dog Food Bowl	Walmart	N/A	1	1.99	1.99
Dog Food	Walmart	N/A	1	5.00	5.00

PCB Parts

Part Description	Manufacturer and Part #	Quantity	Order Quantity	Cost
0.1uF	KEMET – C0603C103F3GECAUTO	13	ESHOP	0
1uF	Taiyo Yuden – TMK107B7105KA-T	2	ESHOP	0
0.1 uF	Yageo – CC0603KRX7R7BB104	4	ESHOP	0
22uF	Samsung Electro-Mechanics – CL21A226MAYNNNE	16	30	3.12
22nF	KEMET – C0805C223F3GECAUTO	4	10	9.33
1000uF	Nichicon – PCG1A102MCL1GS	1	1	2.33
10uF	KYOCERA AVX – F951E106KAAAQ2	1	ESHOP	0
300pF	KYOCERA AVX – 08053C301JAT2A	4	10	1.46
33pF	Murata Electronics – GJM1555C1H330FB01D	1	4	0.56
D_Schottky	Vishay General Semiconductor - Diodes Division – VS-30BQ015-M3/9AT	11	11	7.76
Fuse holder	Keystone Electronics – Fuse Holder	3	3	2.43
Conn_01x02	Molex – 1718560102	9	10	2.45
Conn_01x03	Molex – 0022272031	2	6	1.74
JTAG	Harwin Inc. – M20-9990446	1	2	0.32
UART	Harwin Inc. – M20-9990645	1	2	0.56
UART Backup	Würth Elektronik – 61200621621	1	1	0.43
RFID	HARTING – 14110813001000	1	1	2.68
Conn_01x04	Molex – 0022292041	2	4	2.88

Part Description	Manufacturer and Part #	Quantity	Order Quantity	Cost
0.1uF	KEMET – C0603C103F3GECAUTO	13	ESHOP	0
1uF	Taiyo Yuden – TMK107B7105KA-T	2	ESHOP	0
0.1 uF	Yageo – CC0603KRX7R7BB104	4	ESHOP	0
22uF	Samsung Electro-Mechanics – CL21A226MAYNNNE	16	30	3.12
22nF	KEMET – C0805C223F3GECAUTO	4	10	9.33
1000uF	Nichicon – PCG1A102MCL1GS	1	1	2.33
10uF	KYOCERA AVX – F951E106KAAAQ2	1	ESHOP	0
300pF	KYOCERA AVX – 08053C301JAT2A	4	10	1.46
Barrel_Jack	Same Sky (Formerly CUI Devices) – PJ-102AH	1	2	1.52
Conn_01x05	Harwin Inc. – M20-9990546	1	2	0.4
6.8uH	Würth Elektronik – 7447786006	4	4	9.44
4.2uH	Würth Elektronik – 744053004	1	1	1.4
BSS138	onsemi – BSS138	4	10	1.58
SMD-TRANSISTORS-NPN-25V-500MW_SOT-23	NextGen Components – S8050	2	ESHOP	0
100k	Yageo – RC0805FR-07100KL	7	ESHOP	0
10k	Susumu – HRG3216P-1002-B-T1	41	ESHOP	0
191k	Panasonic Electronic Components – ERJ-14NF1913U	4	10	1.51
49.9k	Stackpole Electronics Inc. – RMCF2512FT49K9	4	10	1.26
499k	Vishay Dale – RCS0805499KFKEA	1	3	0.6
7.5k	Vishay Dale – CRCW08057K50FKEAHP	1	3	0.6
10 ohms	TE Connectivity Passive Product – CRGP2512F10R	1	2	1
33k	Vishay Dale – RCS080533K0FKEA	1	3	0.42
12.7k	Vishay Dale – CRCW121012K7FKEA	1	3	0.6
90.9k	Stackpole Electronics Inc. – RNCP1206FTD90K9	4	10	0.19
40.2k	Vishay Dale – CRCW080540K2FKEAHP	1	2	0.4
SW_Push	Omron Electronics Inc-EMC Div –	2	2	2

Part Description	Manufacturer and Part #	Quantity	Order Quantity	Cost
0.1uF	KEMET – C0603C103F3GECAUTO	13	ESHOP	0
1uF	Taiyo Yuden – TMK107B7105KA-T	2	ESHOP	0
0.1 uF	Yageo – CC0603KRX7R7BB104	4	ESHOP	0
22uF	Samsung Electro-Mechanics – CL21A226MAYNNNE	16	30	3.12
22nF	KEMET – C0805C223F3GECAUTO	4	10	9.33
1000uF	Nichicon – PCG1A102MCL1GS	1	1	2.33
10uF	KYOCERA AVX – F951E106KAAAQ2	1	ESHOP	0
300pF	KYOCERA AVX – 08053C301JAT2A	4	10	1.46
	B3U-1000P			
TestPoint Red	Keystone Electronics – 5011	13	ESHOP	0
ESP32-WROOM -32E 4MB	Espressif Systems – ESP32-WROOM-32E-N4	1	3	14.52
NAU7802SGI	Nuvoton Technology Corporation – NAU7802SGI	1	2	5.28
MP2338GTL-Z	Monolithic Power Systems Inc. – MP2338GTL-Z	4	5	12.7
MP2315SGJ-Z	Monolithic Power Systems Inc. – MP2315SGJ-Z	1	2	5.84
AMS1117-3.3	UMW – AMS1117-3.3	1	3	2.04
Test Point Black	Keystone Electronics – 5011	1	ESHOP	0
Fuse @ 2.5 amps	Littelfuse Inc. – 023502.5MXP	3	3	1.53
Crystal Oscillator	Murata Electronics – XRCGB27M120F3G00R0	2	0.32	2.82
Rfid Chip	NXP USA Inc. – MFRC52202HN1,151	2	7.91	15.82
Motor Driver	Texas Instruments – DRV8825PWPR	2	4.61	9.22

4.3. Total Costs

Cost Type	Cost
Total Labor Costs	$30,615
Total Part Cost	$250.91
Total	**$30,865.91**

4.4. Schedule

Date	Goals - Team Member
Week 3	1. Finish Team Contract (due 02/14) - All 2. Create Weekly Goals for the semester - All 3. Start Project Proposal (due 02/13) - All 4. Start a Parts List - Arash
Week 4	1. Submit Team Contract (due 02/14) - All 2. Finalize a Parts List for the project - All 3. SUBMIT Project Proposal (Due 02/13) - All 4. Start working on Breadboarding - All 5. Start designing PCB - Omkar 6. Start looking at Amazon for parts/make the initial order - Kathryn
Week 5	1. Review Proposal with Prof and TAs - All 2. Start finalizing the design of PCB - Omkar 3. Get Amazon parts - Arash, Kathryn
Week 6	1. Finalize and send out Power PCB - Omkar 2. Finalize Design Document - All 3. Start working on the user interface app - Kathryn
Week 7	1. Submit order for Power PCB - Omkar 2. Design Document (Due 03/06) - All 3. Teamwork evaluation (Due 03/5) - All 4. Keep working on the user interface app - Kathryn 5. Start programming ESP32 - Kathryn 6. Work on testing the sensors and motors - Arash, Omkar 7. Connect ESP32 to the user interface - All 8. Finalize layout for the Brain PCB - Omkar
Week 8	1. Breadboard Demo - All 2. Program ESP32 - Kathryn 3. Order second round of PCB (Due 03/13) - Arash, Omkar
Week 9	1. Spring Break 2. Start working on 3rd round PCB - Arash, Omkar
Week 10	1. Finalize 3rd round of PCB if necessary- Arash, Omkar 2. Start individual progress reports - All 3. Work on the user interface - Kathryn 4. Work on ESP32 Code - Kathryn 5. Start integrating the product - All
Week 11	1. Submit the 3rd round of PCB 2. Submit individual progress reports - All, individually 3. Start testing functionality with PCB - All 4. Test integrated product - All
Week 12	1. Team contract fulfillment - All 2. Finalize project - All

	3. Make sure the project acts as built - All 4. Finish and refine the integrated product - All 5. Prepare for Mockup Demo - All
Week 13	**1. Mock Demo** - All
Week 14	**1. Final Demo** - All
Week 15	**1. Final Presentation (05/6)** - All **2. Final Paper (Due 05/07)** - All 3. Lab Notebook (Due 05/08) - All

5. Conclusion

During the course of the semester, we had many successes. Firstly, we were able to successfully implement the programming of the ESP32 microcontroller on our own custom PCB. We were also able to spot-weld a functional battery pack and create a clean and organized wiring structure that effectively powered all subsystems. Throughout the period between the design phase and the final build, we didn't need to change the design and were able to execute everything we initially set out to do. We developed a local web application backend that could consistently communicate with the ESP32, and we tested the functionality of our custom motor driver PCB. These accomplishments played a key role in a larger system that could dispense food in timed intervals. There were some challenges we faced: the dispensing motor would occasionally jam, particularly with certain food types. Another issue was that one of the motor driver boards was soldered better than the other. This allowed us to prove that our schematic and layout design worked, but we needed to use a development board for the second motor. In the future, we want to increase the RFID range by buying a better receiver or increasing the transceiver power. We would also integrate a water dispensing system in parallel and strengthen the dispensing mechanism. Other enhancements that could be made are connectivity with smartphones, solar power charging, multi-pet support (meaning multiple feeders), and a camera to record or detect the pet. With all the data we log in the pet history page, it would be nice to have data insights for the owner. In designing, ethical and safety aspects were key for us per the IEEE Code of Ethics and ACM Code of Ethics. We did not harm any pets during the test and development phases, and included features to safeguard against power. We also manage data privacy and storage per the ACM Code of Ethics. Through the IEEE and ACM Code of Ethics, we valued each person's input and had a healthy working relationship within the team and with our TA. We didn't order excess parts and did not go about waste when testing, building, and designing. Overall, the project taught us about soldering, crimping, and all of the other aspects related to PCB design, assembly, and testing, as well as soft skills, which we hope to carry forward in our careers.

6. References

6.1. Petmate, “How long can you keep dry dog food once opened?,” *Petmate*, Jun. 26, 2024. [Online]. Available: https://www.petmate.com/blogs/petmate-academy/how-long-can-you-keep-dry-dog-food-once-opened. Accessed: Feb. 18, 2025.

6.2. All Things Wolf, Meow Too, “Pet food automatic containers – Pros and cons,” [Online]. Available: https://allthingswoof.com/blog/44916/pet-food-automatic-containers-pros-and-cons. Accessed: Feb. 18, 2025.

6.3. ACM Code 2018 Task Force: Executive Committee, Association for Computing Machinery, “ACM Code of Ethics and Professional Conduct,” Jun. 22, 2018. [Online]. Available: https://www.acm.org/code-of-ethics. Accessed: Feb. 18, 2025.

6.4. Institute of Electrical and Electronics Engineers, *IEEE Recommended Practice for Electrical Safety in Industrial and Commercial Power Systems*, IEEE Std 3007.3-2012, 2012. doi: 10.1109/IEEESTD.2012.6201457.

6.5. Sparkfun, “FSR 402 Data Sheet,” Oct. 26, 2010. [Online]. Available: https://cdn.sparkfun.com/assets/8/a/1/2/0/2010-10-26-DataSheet-FSR402-Layout2.pdf. Accessed: Feb. 18, 2025.

6.6. MPS, Mouser Electronics, “MP2338,” Dec. 11, 2020. [Online]. Available: https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MP2338GTL. Accessed: Feb. 18, 2025.

6.7. MPS, Mouser Electronics, “MP2315S,” Dec. 21, 2015. [Online]. Available: https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MP2315S. Accessed: Feb. 18, 2025.

6.8. StepperOnline, “E Series Nema 17 Bipolar 42Ncm(59.49oz.in) 1.5A 42x42x38mm 4 Wires w/ 1m Cable & Connector - 17HE15-1504S,” May 25, 2023. [Online]. Available: https://www.omc-stepperonline.com/e-series-nema-17-bipolar-42ncm-59-49oz-in-1-5a-42x42x38mm-4-wires-w-1m-cable-connector-17he15-1504s. Accessed: Feb. 18, 2025.

6.9. Espressif Systems, “ESP32-WROOM-32E / ESP32-WROOM-32UE Datasheet,” [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf.[Accessed: Feb. 18, 2025].

6.10. M. Toll, DIY Lithium Batteries, 1st ed. CreateSpace Independent Publishing Platform, 2017.

6.11. GeckoDrive, “Understanding Stepper Motors with DC Motor Theory,” [Online]. Available: https://www.geckodrive.com/understanding-stepper-motors-with-dc-motor-theory/. [Accessed: Feb. 18, 2025].

6.12. Texas Instruments, “DRV8825 Datasheet,” [Online]. Available: https://www.ti.com/lit/ds/symlink/drv8825.pdf. [Accessed: Feb. 18, 2025].

6.13. Texas Instruments, “UC3717A Datasheet,” [Online]. Available: https://www.ti.com/lit/ds/symlink/uc3717a.pdf. [Accessed: Feb. 18, 2025].

6.14. Texas Instruments, *SN74AUP1T34 – Low-Power Single Buffer Gate*, Datasheet, 2025. [Online]. Available: https://www.ti.com/lit/ds/symlink/sn74aup1t34.pdf. [Accessed: Feb. 18, 2025].

6.15. Digi-Key Electronics, “How to drive a stepper motor,” *Digi-Key TechForum*, Mar. 8, 2021. [Online]. Available: https://forum.digikey.com/t/how-to-drive-a-stepper-motor/13412. [Accessed: Feb. 18, 2025].
6.16. Microchip Technology Inc., “Calculating crystal load capacitor,” *Microchip*, 11 Sep. 2020. [Online]. Available: https://microchip.my.site.com/s/article/Calculating-crystal-load-capacitor. Accessed: Mar. 4, 2025.
6.17. Espressif Systems, *ESP32 - Technical Reference Manual Version 5.3*. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf. Accessed: Mar. 4, 2025.
6.18. V. Mohanan, “Espressif ESP-Prog ESP32 JTAG Debug Probe - Pinout Diagram,” *CIRCUITSTATE Electronics*, 19 Sep. 2023. [Online]. Available: https://www.circuitstate.com/pinouts/espressif-esp-prog-esp32-jtag-debug-probe-pinout-diagram/. Accessed: Mar. 4, 2025.
6.19. Espressif, “Introduction to the ESP-Prog Board,” *DigiKey*, [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/308/Intro_to_the_ESP-Prog_Brd_Web.pdf. Accessed: Mar. 4, 2025.
6.20. Future Technology Devices International Ltd**.**, “TTL234X Series Range of Cables Datasheet,” *FTDI Chip*, [Online]. Available: https://ftdichip.com/wp-content/uploads/2023/07/DS_TTL234X-SERIES-RANGE-OF-CABLES.pdf. Accessed: Mar. 2, 2025.
6.21. DRV8825 Stepper Motor Driver Carrier, High Current – Pololu. Pololu, https://www.pololu.com/product/2133. Accessed 2 Apr. 2025.
6.22. IR Breakbeam Sensors. Adafruit, https://learn.adafruit.com/ir-breakbeam-sensors/arduino. Accessed 2 Apr. 2025.
6.23. Adafruit NAU7802 24-Bit ADC Arduino Guide. Adafruit, https://learn.adafruit.com/adafruit-nau7802-24-bit-adc-stemma-qt-qwiic/arduino. Accessed 2 Apr. 2025.
6.24. DRV8825 Stepper Motor Driver with Arduino Tutorial. Last Minute Engineers, https://lastminuteengineers.com/drv8825-stepper-motor-driver-arduino-tutorial/. Accessed 2 Apr. 2025.
6.25. NAU7802 Pet Food Scale Project Overview. Adafruit, https://learn.adafruit.com/nau7802-pet-food-scalehttps://learn.adafruit.com/adafruit-nau7802-24-bit-adc-stemma-qt-qwiic/overview. Accessed 2 Apr. 2025.

7. Appendix

Requirements and Verification Table by Subsystem

Subsystem 1: Refilling Food Store	
Requirement	**Verification**
Must utilize a presence sensor to determine how full the food store is at said volumes, 50% and 20%.	Fill the food store to 20% capacity. Check if the Presence Sensor reflects the capacity.
Must send readings to the UART located in the Brain Subsystems.	Check if the ESP32 can read the value of the sensor.

Subsystem 2: Power System	
Requirement	**Verification**
Must be able to provide enough power to all components within a tolerance of ±0.2V for the 3.3V supply and ±0.5V for the 5V supply during the periods of operation.	We will probe test points with J-hooks to test voltage ripple and average voltage to meet specifications.
If there are any unsafe conditions, such as the supply reaching undervolt range or short-circuiting, the current will be cut-off from the rest of the subsystems to prevent damage and fire hazards.	We added power protection components like fuses and Schottky diodes for reverse polarity and overcurrent protection. We have a commercial BMS that is designed for over- and under-voltage and over-current protection. Successful testing of this product would mean that we discharge cells to their undervoltage limit and verify if the BMS shuts off the load current.

Subsystem 3: Food Dispenser	
Requirement	**Verification**
Implements a timer system to determine "feeding times" assigned by the user. Will not dispense food prior to these times within a small tolerance of ±10 minutes.	We will set the times in the user interface for food dispensing. We will see if the food is dispensed at these times. We will set a timer on our phones to see to what tolerance it takes before the food is dispensed.
Utilizes a button for the pet to step on to indicate the pet's presence. Upon pressing, during the feeding time, the stepper motor will start dispensing food to the food bowl [4.5].	We will see if the food is dispensed when we press the button. We will see how much force is required for the food to be dispensed. We will measure our success through the dispensing of food following the pressing of the button.

Implementing a stepper motor design, a lid will cover and uncover the pet bowl upon request of the signal.	We will verify if this works if the lid closes after the pet leaves. We will test using the RFID tag. We will test the ranges that the RFID tags can be within the RFID reader while the lid stays open. If the lid closes and opens upon detecting the presence of the RFID tag, we will consider the operation successful.
We will also have a load cell to detect the bowl weight, and if there's a stoppage in dispensing, when it should have done a cycle of dispensing.	We will see if the load cell can read the weight of the food bowl. The food bowl will be mounted on top of the load cell. When we connect the load cell to the ESP32 chip, we will be able to read the strain applied to the load cell. We will print these values to the terminal and see if the strain changes as we apply varying pressure to the food bowl.

Subsystem 4: RFID for pet identification	
Requirement	**Verification**
The RFID UIN will be detected by the brain microcontroller within a range of 6 inches.	We will test the ranges that the RFID is able to pick up the tag using a ruler. If the tag is able to be detected, we will see the tag UIN in the Brain Subsystem.
The 2+ RFID tags will be preprogrammed and recognized by the reader.	When we detect the RFID tags with the RFID reader, the tag UIN will be seen by the ESP32. We will be able to visually determine that the tags have unique UINs.

Subsystem 5: User Interface	
Requirement	**Verification**
The user interface will be able to connect with the Brain subsystem to receive different notifications.	The user interface will be able to see and print readings from different sensors/motors on the screen.
The user interface must be able to update with the current status within 5 minutes of an event. This includes pet eating habits, food store levels, and whether food has been dispensed.	We will manually verify if the user interface updates as food is dispensed, the food store level changes, or if the food has not been finished.

Subsystem 6: Brain	
Requirement	**Verification**
Sends and receives UART signals to and from the other subsystems successfully.	The Brain will be able to connect to the different sensors and motors. The values read by the sensors will be printed to the terminal when connecting to the ESP32 chip. In addition, given a command from the ESP32, the server will be able to turn on or off.