Automatic Pet Sitter

Final Report - Fall 2022

Abhijit Chebiyam (abhijit3@illinois.edu)

Joseph Choi (jschoi5@illinois.edu)

Tyler Huang (tylerh4@illinois.edu)

December 8, 2022

Team 21

Professor: Arne Fliflet

Teacher Assistant: Hojoon Ryu

Contents

1 Introduction

1.1 Problem

Tyler's friend Andrew is a recent college grad who is now working full time. Andrew has a pet cat that he has trouble feeding while he's away at work because he does not want to leave food and water out while he is away from home. Andrew would like a way to automatically feed his cat with the peace of mind of not overfeeding, while also knowing how much food and water his cat is consuming throughout the day. Lastly, Andrew would like to be provided updates on when the food/water bowl needs to be refilled and how much needs to be dispensed.

1.2 Solution

The solution to this problem is to design an automatic dispenser setup that delivers food and water to the user's pet. Our solution will involve a couple of subsystems in order to provide users' pets with the most comfort and care. Each dispenser is purposed for a single pet. The first and main subsystem will revolve around a pressure pad/sensor that will act as the main trigger for the dispensing system. The pressure pad can be customized to a certain weight range for the specific pet and while configure the amount of food and water needed based on those specifications and the breed of the pet itself. The next subsystem involves the dispensing mechanism for food and water. Finally, a notification subsystem will send updates to the user when their pet is eating or drinking. Furthermore, our solution also includes an automatic mode that will dispense a set amount of food or water at set intervals throughout the day (all determined by the user).

1.3 Visual Aid

1.4 High Level Requirements

These three requirements outline what's most important towards building a successful and optimal food/water dispensing system:

1. **Accuracy:** We want there to be easy communication between the user and when a pet interacts with the dispensing system. This will be judged by the accuracy of the dispensing system to provide food to the pet, as well as how accurate the measurements of the pet's weight are.

2. **Timeliness:** The user's ability to react when a pet interacts with the dispensing system will depend on how quickly these updates can be sent. The pet's weight, the amount of food/water to be dispensed, as well as the updates corresponding to these actions need to be fast enough for any situation that may occur.

3. **Adaptability:** The dispensing system must be able to account for a variety of scenarios in order to ensure the safety of the pet and to maintain the user's trust. This includes having the right contingency plan when the pet does not step onto the weighing scale long enough or when a refill is needed or even when the dispensing system malfunctions.

1.5 Block Diagram

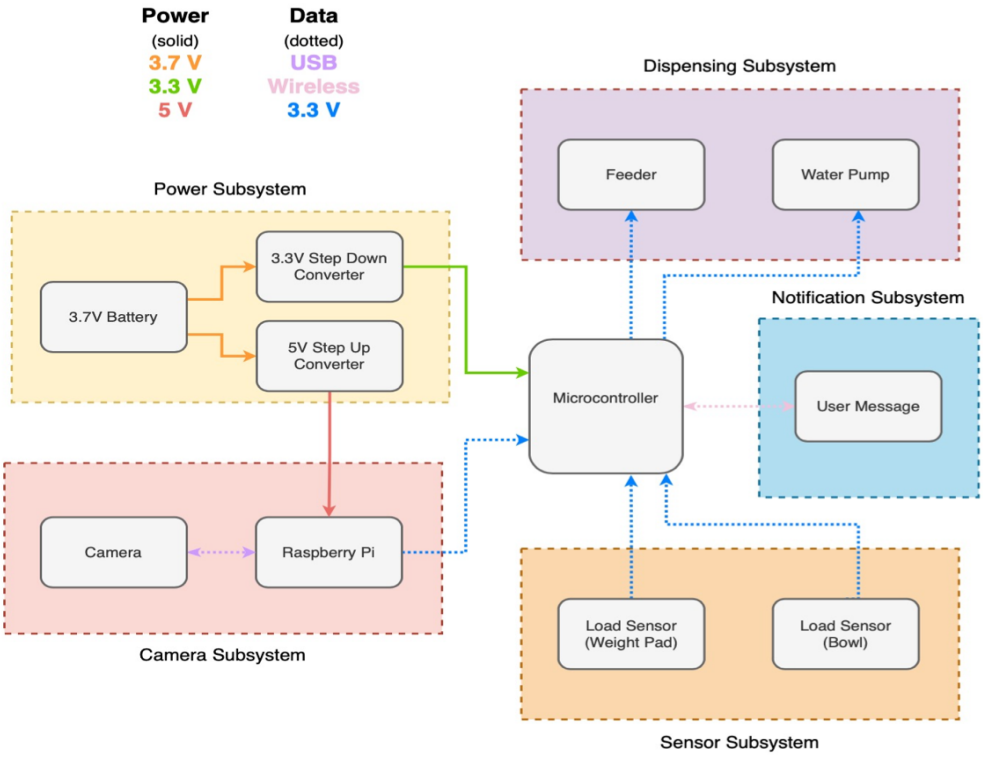


Figure 1: Original Block Diagram

2 Design

2.1 Design Procedure/Details

2.1.1 Microcontroller

For the PCB design, our first approach was using an ATMEGA32U2 design for our implementation. During our initial design process for the PCB, our alternative was a PCB design with an ATMEGA328P, but there has been a continuous shortage on these chips and we had to resort to using a less familiar microcontroller. The ATMEGA32U2 ran on 5V and was compatible with our servo motor, which needed 5V to run. Additionally, the ATMEGA32U2 is programmable by USB by computer. All the necessary programs for programming were available online, which included the DFU programmer and avr-gcc. The design was purposefully kept simple. As shown in Appendix A, the design consisted of a USB-B connector, which connected to the 5V input of the microcontroller and 22Ω pull up resistors for the D+ and D- pins of the USB. External timing was dealt with using a 15pF capacitor and 8MHz crystal configuration. However, we got stuck on an issue where the ATMEGA32U2 wasn't showing as a device on a computer. Initially, there was the idea that it was a soldering issue, so both traditional soldering and soldering with solder paste and a heat gun were used. The Unfound device issue still lingered, so there was extensive multimeter checking on voltages and proper grounding on each component. There was an issue with the second capacitor on the External Timing circuit not being grounded properly, but even after grounding it properly the issue remained. The D+ and D- voltage reading were both around 0.33V. The D+ and D- have an average voltage difference of 3V, but there was no voltage difference and the voltages were too low. A third issue of the PCB was made where completely new parts were soldered with the same component values. After investing a considerable amount of time into debugging, it was decided that we had to move on in the design process in order to fulfill other functionalities in the project.

While another member was proceeding with programming the dispensing functionalities with an Arduino, another member created a new PCB design. The design is shown in Appendix B. It was decided an alternative approach would be to use a different microcontroller due to the previous USB programming method for the microcontroller was not working. The ESP32-Wrover-E was decided

on. First, it utilized traditional programming methods with an external programmer connecting directly to its pins. Second, this microcontroller had WiFi [5] capabilities, which would simplify our design for the notification system as a Raspberry Pi wouldn't be needed. Although all the parts were due to arrive shortly after the mock demo many parts weren't successfully delivered. However, we proceeded to solder the parts that arrived and found that the ESP32-Wrover-E system was successful and indeed programmable with an external programmer.

The Arduino design was the final implementation we ended up with. The Arduino provided 5V to the servo motor and also had the capabilities of connecting to an ESP8266 kit that was successfully used for the notification subsystem.

2.1.2 Dispensing Subsystem

DC Servo Motor

The motor will be connected to the micro controller and will control the rotating door for that contains and dispenses the food in the food container. Its power will be supplied by the power subsystem and will require five volts and two amps.

Food/Water Containers

The containers will be used to store dry food and water for dispensing. The food container will have the Servo Motor and rotating door at its spout to control the dispensing of the food. The containers should be closed to preserve the stored food/water.

Water Pump

The water pump will get 5V of power from the power subsystem and dispense water out of the water container into the bowl. The water pump will sit in the water and a tube will lead to the bowl to dispense water.

The initial approach was to use two servo motors for the water and food dispensing, but we quickly decided a water pump and tube would is the smarter method. For the food dispensing, we maintained the use of a 5V motor. The following calculations were used to find the theoretically needed

torque for our motor.

$$N = \tau / r$$

$$r = 2in * (2.54cm/1in) = 5.08cm$$

$$N = \tau / r \ = (20kg * cm)/5.08cm = 3.93kg$$

After moving on to the actual implementation and physical design, we found out that the motor didn't have enough power to rotate the door without it getting stuck most of the time. The current wall adapter we were using was a 5V and 1A output, which gave us:

$$P = V * I = 5V * 1A = 5W$$

This was clearly not enough. With the motor running on 5V, we decided to increase the current supplied to increase power. A 5V and 2A adapter proved to supply enough power for the servo motor to dispense food through physical test runs.

$$P = V * I = 5V * 2A = 10W$$

For the water dispensing, we found out pumps and tubes were easier to work with and most importantly much cheaper. The pumps worked with the 5V supply being controlled by the analog input connector. The only calculation necessary was timing the output of the water because overflow would be protected by the max capacity weight communicated by the bowl sensor.

2.1.3 Power Subsystem

3.7V Lithium Battery

The 3.7 V Lithium Battery will supply power to our components. No components are directly connected to the battery, instead converters adjust the voltage to 3.3V or 5V in order to supply the correct voltage to our components.

5v Step Up Converter

A 5V step up converter will take the 3.7 Lithium battery as input, and supply the Raspberry Pi

with 5V of power

3.3V Step Down Converter

A 3.3V step down converter will take the 3.7 Lithium battery as input, and supply the micro controller with 3.3V of power

The initial approach to power our project was using a 3.7V battery that would be converted to 3.3V and 5V by a step-down converter and step-up converter. We originally chose this design so that we could have a battery powered project and that would be able to feed power to our Raspberry Pi and our micro controller. An alternative approach could have also been to use a 5V battery with a step down to 3.3V; However, we chose our original design so that the change in voltage from battery to converter would not be as large.

In our final design, we decided to use a 5V wall adapter as the only component in our power subsystem. With a wall adapter, we would not have to worry about recharging the battery, which could cause huge problems in an autonomous project. Furthermore, with the change to only using Arduino, we no longer needed to supply power at 3.3V.

2.1.4 Sensor Subsystem

The Load Sensors in the weight pad will be used to determine whether the owner's pet is on the weight pad. The data from the sensor will be sent to the micro controller to determine whether the sensor reading should trigger the dispensing subsystem to dispense food and water to the bowls.

Load Sensor (Weight Pad)

The Load Sensors in the bowl will be used to determine whether the user's pet is on the weight pad. If so, it will trigger the dispensing subsystem to dispense food to the pet while it is still on the weight pad.

Load Sensor (Bowl)

The Load Sensors in the bowl will be used to determine the amount of the food or water has been consumed by the pet. The data from the sensor will be sent to the micro controller to calculate how much food or water was consumed based on data from the dispensing system versus the amount of food or water in the bowl.

Our original design was to use SEN-10245 sensors under the weight pads and the bowls to weigh them. We would use four of these load cells under the weight pad and one under each of the bowls. Four cells would help to accurately determine the weight of the pet regardless of their location on the weight pad, and one cell under each bowl would be adequate as it would be placed under the center of the bowl. Alternative approaches could have been to use load resistivity sensors (which we switched to in our final design); However we chose to use load cells as they could more easily be implemented with our physical design.

In our final design, we decided to switch to using load resistivity sensors under the weight pad and bowls instead of the cells. This was due to us having trouble setting up the SEN-10245 with Arduino as we could not get accurate readings with the load cells.

2.1.5 Notification Subsystem

For our notification subsystem, we will create a program that lets the user know when their pet is interacting with the load and dispensing subsystems, and subsequently sends them a text message. This process will be developed through several steps. First, we want to update the Raspberry Pi through ths Raspberry PI Imager which will burn the OS onto the SD card. Then, we will install Tensorflow on the Raspberry Pi terminal. This will contain the Tensorflow object detection API. After that, we will install OpenCV and its related dependencies. Then, we want to compile and install Protobuf, which implements Google's Protocol Buffer data format. We should also setup the Tensorflow directory structure and modify the PYTHONPATH environment variable to point at some directories inside the TensorFlow repository. After downloading the SSD-Lite model from the TensorFlow detection model zoo, we can enable Picamera in the Raspberry Pi configuration menu. From here, we should create a Twilio account and extract SID, AUTH-TOKEN, the Twilio-generated phone number, and the phone number you want to send the message to. We should set these environment variables before running the program. Lastly, we need to write the script which will contain the object detection algorithm. Each detection frame has three objects: class name, confidence, and location. One important thing to note is that if the midpoint of the pet's location

is in the refill box or the dispensing box for more than 10 frames, it will send a message to the user stating one or the other. Lastly, we should test with various pets and other forms of human error to determine timeliness and accuracy.

When the Raspberry Pi failed to function, we had to quickly find an alternative approach for the notification functionality. We decided to proceed with an ESP8266 dev kit that could easily connect with the microcontroller system. Additionally, the ESP8266 was capable of connecting to home WiFi and we programmed it to connect to the PushingBox server and utilized the Prowl API to send quick and simple notifications that alerted the pet owner of pet activity. The code and interface of the notification system can be seen in the appendix.

2.1.6 Camera Subsystem

Raspberry Pi 4 Model B

The Raspberry Pi connects to the camera and captures photos when the load sensor of the weight pad is activated. The Pi should be able to power the camera and control it to take pictures of the pet when it is on the pad to eat or drink. The Pi then sends the photos to the user through the notification subsystem.

Camera

The camera will be powered by the Raspberry Pi and positioned to have the pressure pad and pet in frame when capturing photos.

3 Verification

3.1 Sensor Subsystem Testing

Our final version of our project used Load Resistivity Sensors in our Sensor Subsystem under the bowls and weight pad. These sensors work by decreasing their resistance as more force is applied to them. As we pivoted away from using the SEN-10245 load sensors that we had in our original design, we ran tests to determine the amount of resistance the sensor would have with different values of force applied.

Force (g)	Force (lb)	Resistance (Ohm)	Voltage (V)
100	0.22	5968	3.13
1000	2.2	982	4.55
10000	22	249	4.88

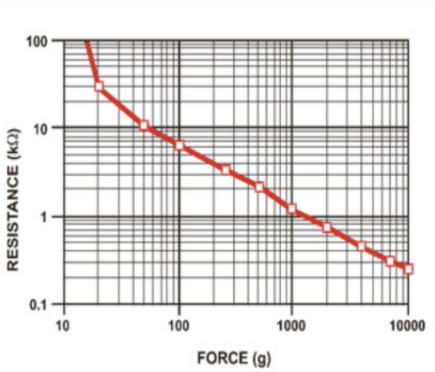


Figure 2: Force vs Resistance Graph and Table [6]

This testing allowed us to determine two key things:

1. With these Load Resistivity Sensors, we could get good analog readings of a pet's weight for up to 22 lbs. This is because the resistance of the sensors decreases by a large enough magnitude with each lb of weight onto it before 22 lb, such that when we read the analog voltage output we can determine a precise weight.

2. The resistance of the sensor had a greater rate of change at lower weights, meaning we can more precisely determine the weight of a pet when it is lower. This is ideal for our project as heavier pets would not need precise weighing to determine if the pet was on the pad, whereas for very light pets, the sensor would need more precise readings to determine if the pet triggered the weight pad.

3.2 Dispenser Subsystem Testing

Testing our Dispenser Subsystem is important for two main reasons:

1. In order for our project to dispense the correct amount of food and water when set to automatic mode. The user will determine how much food and water to dispense to their pet, therefore our dispenser must be controlled and programmed to dispense that amount accurately.

2. When the pet triggers the weight pad, we want to dispense a continuous, but not too large amount of food and water, therefore requiring precisely controlling the amount of food and water dispensed.

Trial #	Food Pieces
1	24
2	30
3	17
4	17
5	27
Avg.	23

Trial #	Output (sec.)
1	1.1
2	1.3
3	1.8
4	N/A
5	1.3
6	2.0
7	5.3

Figure 3: Food and Water Dispenser Testing Tables

The table on the left contains the results of our food dispenser testing. In each trial, we allowed the servo motor to make one full rotation, and then counted how many pieces of food were dispensed. We performed 5 trials with varying results due to the random nature of food falling out of the container. We found that on average 23 pieces of food would be dispensed on one full rotation of the servo motor. With this data, we would be able to know how many rotations must be performed by the servo motor in order to dispense a set amount of food pieces.

The table on the right contains the results of our water dispenser testing. In each trial, we turned on the water pump and then timed how long it would take for the water to start filling into the bowl. Water would take sometime to travel from the pump in the water container, through the tubing, and finally into the bowl, causing a delay between turning the motor on and water being dispensed

in the pet's bowl. We found after 7 trials that the average delay between turning on the motor and water being dispensed into the bowl to be about 2.13 seconds. In one trial, despite the motor being on and the container having water, we found that no water was being dispensed due to a kink in the tubing.

From these results we were able to determine that approximately a 2.13 second delay should be added between turning on the water pump and counting the timed dispense, as no water would actually be dispensed in those first 2.13 seconds. We also now knew to set up our physical design to have the the tubing from water pump to bowl be as straight as possible without any kinks.

3.3 Notification Subsystem Testing

We tested our Notification Subsystem to determine the delay between triggering of the weight pad by a pet, and the user receiving a notification.

Trial #	Msg Latency (sec.)
1	1.32
2	1.53
3	1.45

Figure 4: Notification Subsystem Testing Table

The testing of this subsystem was very simple as we just needed to time from when we placed weight on the weight pad, to when we saw a notification on our phone. We found that the notification subsystem reacted very quickly, and was able to send a notification in an average 1.43 seconds over 3 trials. We found this timing to be suitable for our project.

3.4 Requirements and Verification Table

Our Requirements and Verification Table (See Appendix C), included requirements and verification for our power subsystem, sensor subsystem, dispensing subsystem, notification subsystem, and camera subsystem. Throughout our project however, two of our subsystems had substantial changes such that the RV table was no longer applicable.

1. The first is the power subsystem, where we decided to switch from a battery to a wall adapter to better suit the needs of our project. Furthermore, we switched to using an Arduino which only required a 5V input, meaning that we no longer needed converters as we could just use a 5V wall adapter. We were able to test the wall adapter's voltage output and found it to be within the 0.1V tolerance we set for the battery.

2. We found that our camera subsystem would not be able to be implemented due to our Raspberry Pi having issues. This meant that we would not be able to verify any of the requirements that we had in our table originally.

Despite these setbacks, we were still able to fulfil our requirements for the other three subsystems, as well as verify through the testing that we had in our table. The previous sections go over the testing in greater detail, with graphs and table.

4 Cost

The total cost for all parts required for this project before shipping is $168.2. With shipping costs adding another 5% and sales tax being 6.25%, these extra costs tack on $18.92. With this project being relatively expensive, we can expect each person to cost $50.48 per hour in terms of labor costs. In total, a salary of $50.48/hr x 3 hours x 45 days = $6,814.80 would cover the cost of each team member. Since our team has 3 people, this means we have $6,814.80 x 3 = $20,444. With the cost for the parts being $187.12, this comes out to a total project cost of $20,631.12.

Costs			
Description	Manufacturer	Quantity	Extended Price
20KG Digital Servo Full Metal Gear High Torque	Lewansoul	1	$15.98
Pet Food Storage Container	Pission	1	$11.99
Pet Water Fountain Pump Replacement	Chewy	1	$5.95
Suteck Plastic Access Panel	Suteck	2	$9.94
Raspberry Pi 4 Model B/2 GB	PiShop.us	1	$45
OB5647 Camera Sensor Raspberry Pi Platform Evaluation Expansion Board	Digi-Key Electronics	2	$45.98
702528 3.7 V Lithium-Ion Battery Rechargeable (Secondary) 500mAh	Digi-Key Electronics	1	$6.49
Pololu 5V Step-Up Voltage Regulator U3V12F5	Polulu	1	$45
ETA9740 Battery Charger Power Management Evaluation Board	Digi-Key	1	$4.90
Half-bridge strain gauge Load Cell Body Scale Weighing Sensor Amplifier	Onilab	2	$15.98

5 Conclusion

The course of this semester provided an exciting opportunity to work on a project would integrate all forms of learning across our college classes. In general, we learned a lot from working on a physical system which was constantly revised and inspected. The first aspect of this project that we undermined was how long the physical design actually took compared to what we expected. Had we been more experienced in soldering, we may have been able to speed up this process, however we spent several hours at the lab trying to figure out what was wrong with our design. This was actually rooted from our inability to design and solder a working PCB. We thought after two iterations, our PCB would be functional, however we soon found that we were running out of time and that our PCB designs weren't recognizable by the computer after several soldering iterations. The last thing we realized was that the individual parts we purchased contributed to a high overall cost. We thought that we had been cost-efficient, however changing our design cost us a lot of time and money. These key takeways have led us to understand the improvements we need to make from next time onwards. First, we should expect the orders to be delayed. Waiting for the order put us on standby while instead we should've been brainstorming other design features we could've developed or what we would need to do if that order didn't lead us in the right direction. We also spend less time on developing the same version of the PCB if it's not working. It's much better to instead develop another PCB and check its functionality. Lastly, even if the subsystems don't integrate well with each other, we should build and test them seperately much earlier in the project. For example, breadboarding can be one of the most effective ways to check if all voltages and currents are flowing as intended. Overall, this project was a great learning opportunity and allowed us to build a product from start to finish through our own strategy and planning.

The IEEE Code of Ethics states the need to uphold safety, health, and welfare of the public. The goal of our project is to automatically dispense sustenance for the pet and give the user a better understanding of the well-being of their pet when they are away. While the project should greatly help users with taking care of their pet, it is not a complete substitute for the care of the pet. The owner is responsible for the overall understanding and care of the well-being and health of their pet. To uphold the Code Ethics, we will explicitly warn users of this.

IEEE Code of Ethics states in Section 1.5 that members must be committed to seek and accept honest criticism of their work and correct their errors. To uphold this code, we will communicate with TA's and the professor during the whole process of creating our project. We will carefully listen to criticism and be thoughtful and quick to fix errors that arise during development.

As team members, we will strive to uphold Section II and III of the IEEE Code of Ethics by making sure members of the group and the TA is being treated fairly and with respect and that there is no harassment or discrimination.

With the system taking pictures and sending them to a mobile device, as a group, we will make sure that the pictures are being securely sent to the user and privacy of the user is kept.

References

[1]IEEE Code of Ethics, author = IEEE, title = IEEE Code of Ethics, year = 2022, note = Last accessed 28 September 2022, url = https://www.ieee.org/about/corporate/governance/p7-8.html

[2]Arduino Bathroom Scale With 50 Kg Load Cells and HX711 Amplifier, author = Instructables, title = Arduino Bathroom Scale With 50 Kg Load Cells and HX711 Amplifier, year = 2022, note = Last accessed 12 September 2022, url = https://www.instructables.com/Arduino-Bathroom-Scale-With-50-Kg-Load-Cells-and-H/

[3]Arduino Bathroom Scale With 50 Kg Load Cells and HX711 Amplifier, author = Instructables, title = Arduino Bathroom Scale With 50 Kg Load Cells and HX711 Amplifier, year = 2022, note = Last accessed 15 October 2022, url = https://www.instructables.com/Arduino-Bathroom-Scale-With-50-Kg-Load-Cells-and-H/

[4]Load Cell Amplifier HX711 Breakout Hookup Guide - Digi-Key, author = Media Digikey, title = Load Cell Amplifier HX711 Breakout Hookup Guide - Digi-Key, year = 2022, note = Last accessed 2 December 2022, url = https://media.digikey.com/pdf/Data20Sheets/Sparkfun20PDFs

[5]ESP32-WROVER-E ESP32-WROVER-IE - Espressif, author = Espressif, title = ESP32-WROVER-E ESP32-WROVER-IE - Espressif, year = 2022, note = Last accessed 28 November 2022, url = https://www.espressif.com/sites/default/files/documentation

[6]Force Sensitive Resistor (FSR), author = Adafruit, title = Force Sensitive Resistor (FSR), year = 2022, note = Last accessed 30 October 2022, url = https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr

[7]Microcontroller Tutorial (1/5): What is a Microcontroller?, author = Build Electronic Circuits,

title = Microcontroller Tutorial (1/5): What is a Microcontroller?, year = 2022, note = Last accessed November 7 2022, url = https://www.build-electronic-circuits.com/microcontroller-tutorial-part1/

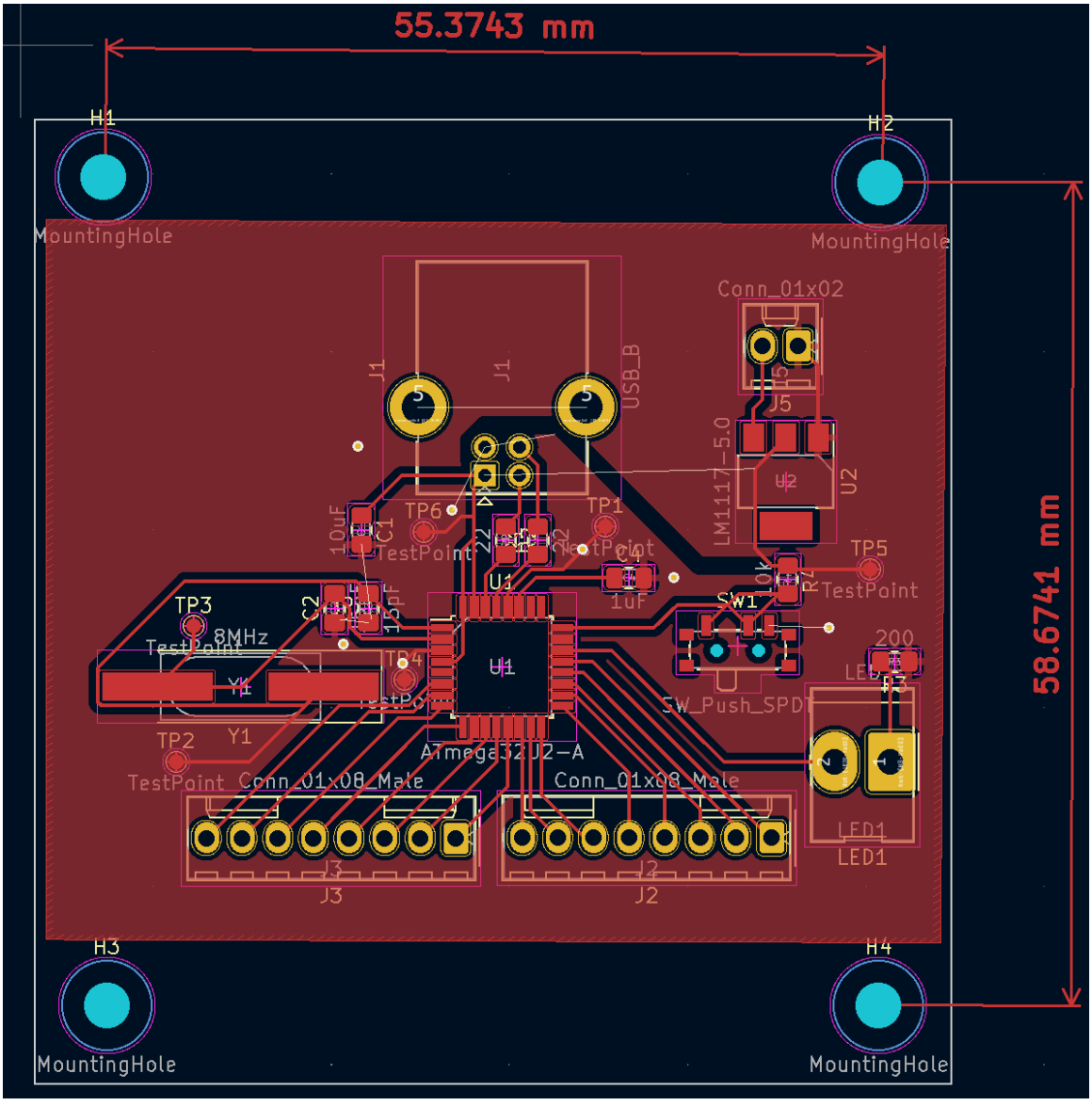
[8]How to Connect ESP8266 to WiFi: A Beginner's Guide, author = Electronics Hub, title = How to Connect ESP8266 to WiFi: A Beginner's Guide, year = 2022, note = Last accessed October 15 2022, url = https://www.electronicshub.org/connect-esp8266-to-wifi/

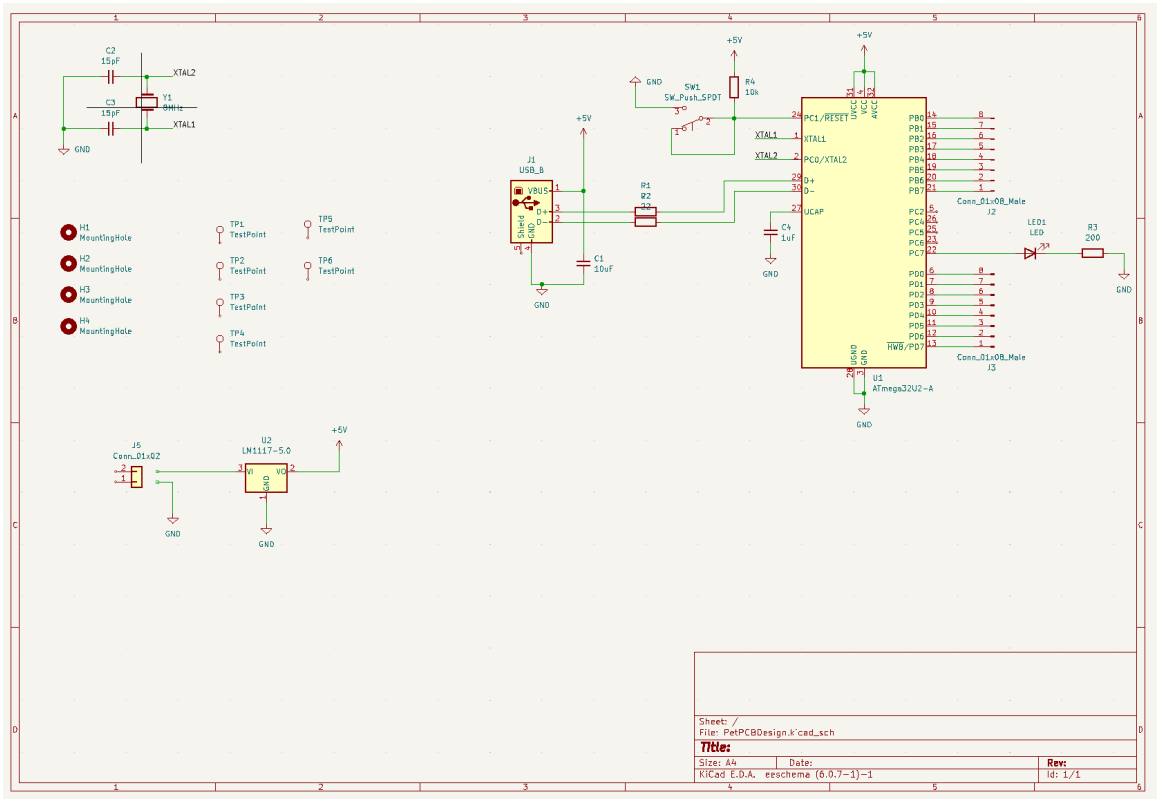
[9]Connectors, author = USB in a NutShell - Chapter 2 - Hardware, title = Connectors, year = 2022, note = Last accessed 18 September 2022, url = https://www.beyondlogic.org/usbnutshell/usb2.shtml

[10]How To Solder: A Complete Beginners Guide, author = Makerspaces, title = How To Solder: A Complete Beginners Guide, year = 2022, note = Last accessed 2 November 2022, url = https://www.makerspaces.com/how-to-solder/

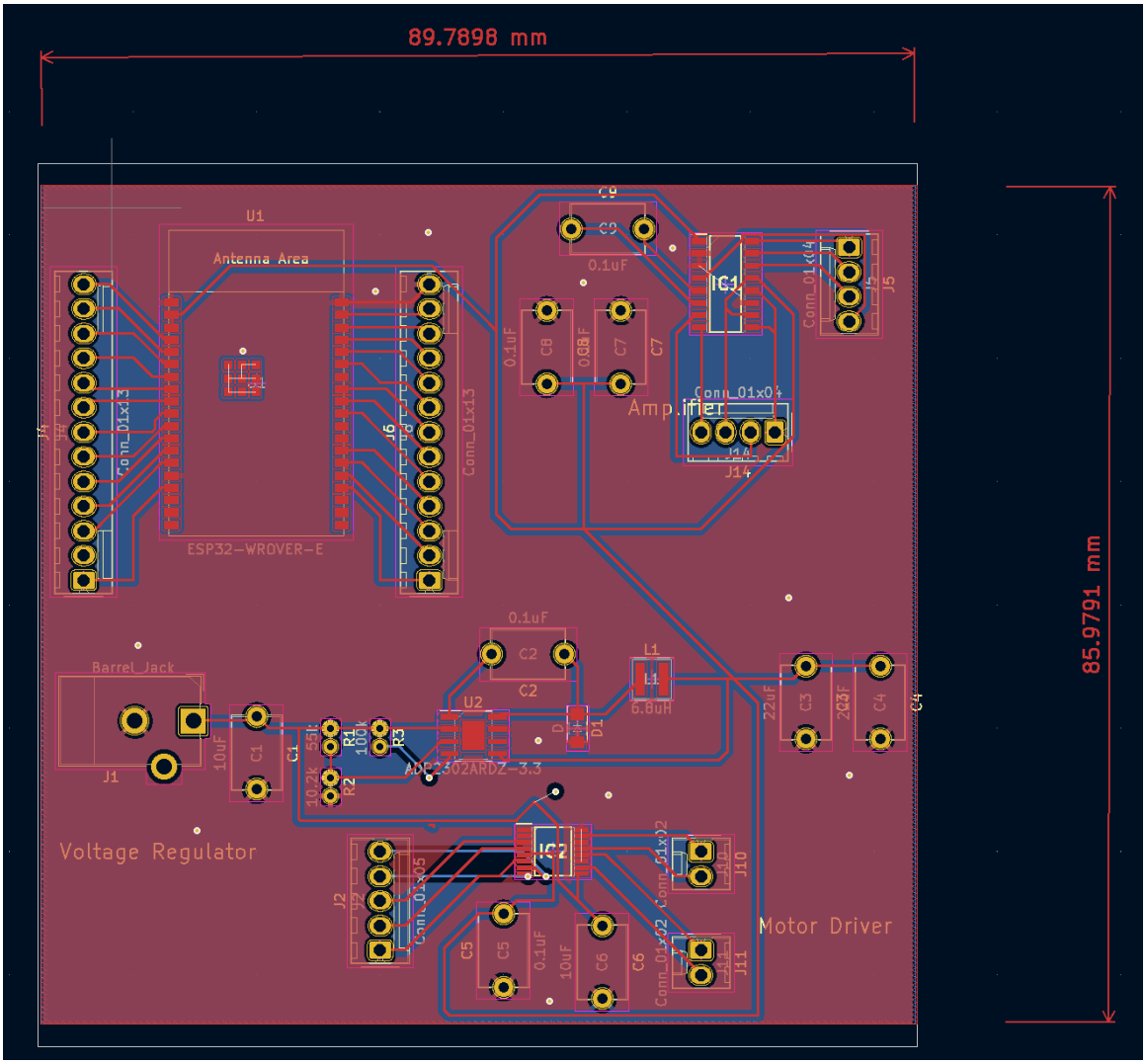
Appendices

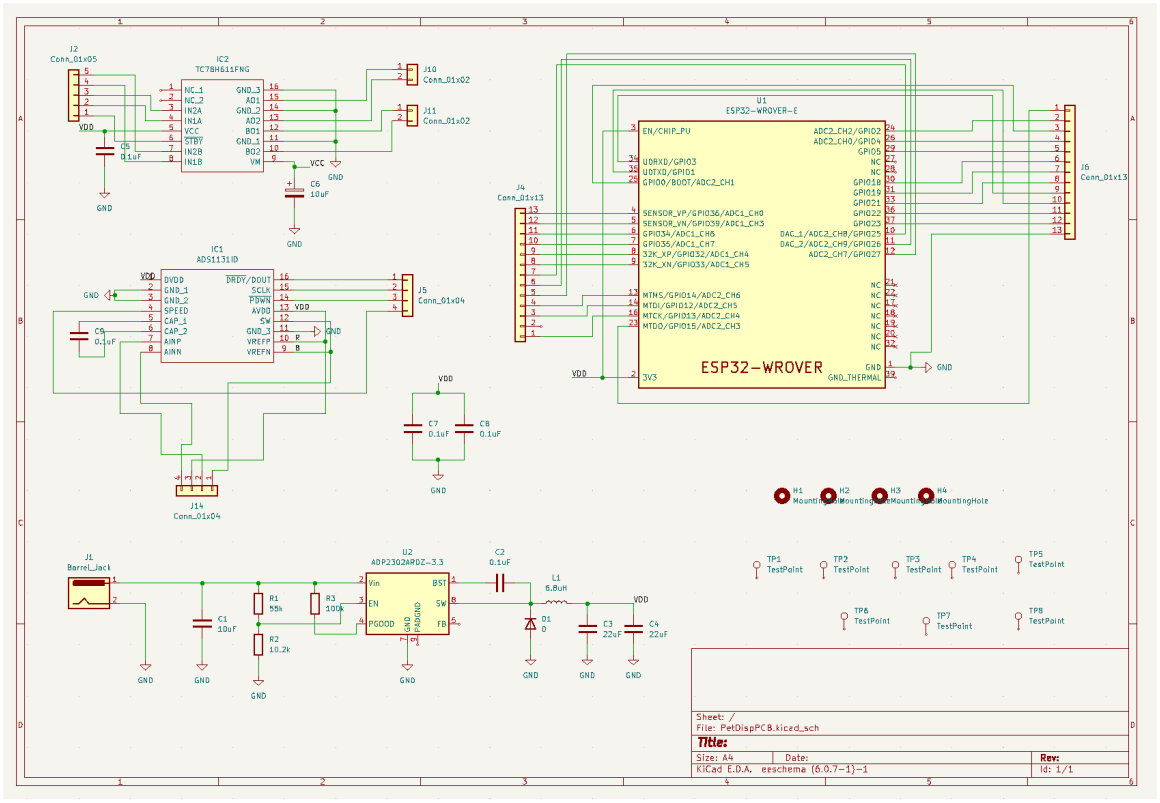
A First PCB Design




C2
15pF
C3
15pF
Y1
8MHz
XTAL2
XTAL1
GND
H1
MountingHole
H2
MountingHole
H3
MountingHole
H4
MountingHole
TP1
TestPoint
TP2
TestPoint
TP3
TestPoint
TP4
TestPoint
TP5
TestPoint
TP6
TestPoint
J5
Conn_01x02
U2
LM1117-5.0
VI
VO
GND
+5V
J1
USB_B
VBUS
D+
D-
Shield
GND
R1
22
R2
22
C1
10uF
SW1
SW_Push_SPDT
R4
10k
C4
1uf
U1
ATmega32U2-A
PC1/RESET
XTAL1
PC0/XTAL2
D+
D-
UCAP
UVCC
VCC
AVCC
UGND
GND
HWB/PD7
J2
Conn_01x08_Male
J3
Conn_01x08_Male
LED1
LED
R3
200
Sheet: /
File: PetPCBDesign.kicad_sch
Title:
Size: A4
Date:
KiCad E.D.A. eeschema (6.0.7-1)-1
Rev:
Id: 1/1

B Second PCB Design




ESP32-WROVER
ESP32-WROVER-E
TC78H611FNG
ADS1131ID
ADP2302ARDZ-3.3
Sheet: /
File: PetDispPCB.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. eeschema (6.0.7-1)-1
Id: 1/1

C Requirements and Verification Table

Power Subsystem

	Requirements	Verification
3.7 V Lithium Battery	The 3.7V Battery should be able to supply at least 3.7V +/ 0.1V.	Measure voltage produced by battery to see if it is within required tolerance.
5V Step Up Converter	The 5V step up converter should be able to supply at least 5V +/ 0.1V.	Measure voltage produced by converter to see if it is within required tolerance.
3.3V Step Down Converter	The 3.3V step down converter should be able to supply at least 3.3V +/ 0.1V.	Measure voltage produced by converter to see if it is within required tolerance.

Sensor Subsystem

	Requirements	Verification
Load Sensor (Weight Pad)	This load sensor should be able to accurately determine the weight of the pet +/- 0.1kg up to 50 kg.	Place a weight on the weight pad to see if weight can be accurately determined by our load sensor.
Load Sensor (Bowls)	This load sensor should be able to accurately determine the weight of food or water in the bowl +/- 0.1kg up to 50 kg.	Place a weight in the bowl to see if weight can be accurately determined by our load sensor.

Dispensing Subsystem

	Requirements	Verification
DC Servo Motor	The motor should be able to push or rotate the door at a precise amount proportional to the opening of the container to control rate of dispensing.	Attach a door to the Servo Motor, and then to the container to see if the motor can rotate to allow food to fall through at a consistent rate.
DC Servo Motor	The motor must be powerful enough to move the door with 5 lbs of food weighting the door down.	Measure and place 5 lb of food on top of the servo motor door to see if it can still rotate.
DC Servo Motor	The motor must be able to be controlled by the microcontroller.	Connect the servo motor to the microcontroller to see if it can be programmatically turned on and off.
Food/Water Containers	The containers need to be able to store 5 lbs of food or water.	Measure 5 lb of food and water and see if each container can hold that amount of food or water.
Food/Water Containers	The containers must have a spout that is able to be closed by the door.	Check to see if the water pump with tubing can fit in the water spout, and if the servo motor with the door can cover the spout of the food container.
Water Pump	The water pump must be able to be turned on or off by the microcontroller.	Connect the water pump to the microcontroller to see if it can be programmatically turned on and off.

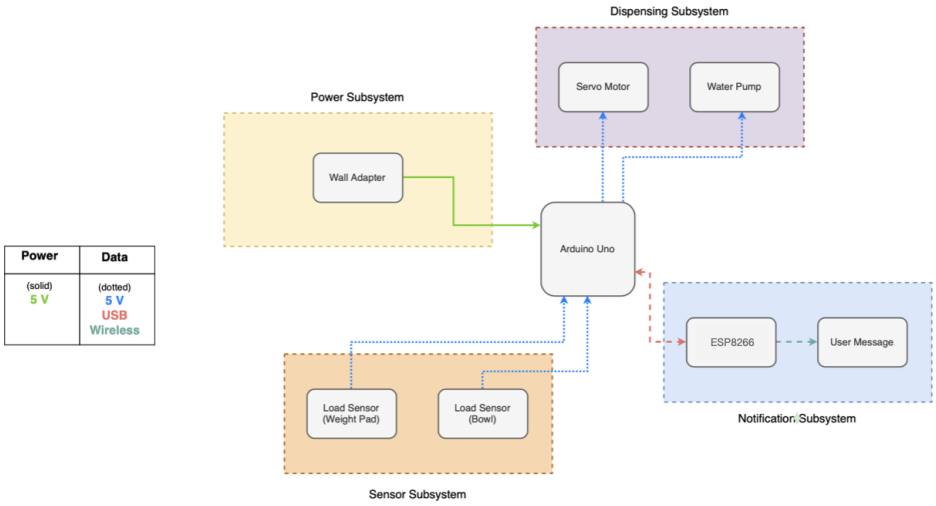
Notification Subsystem

	Requirements	Verification
SMS API	The API should be able to send an alert to the user regarding their pet's interaction with the dispenser.	Place a weight on the pad and use a timer to record the latency.
Message/Notification	A notification must be sent to the User at most 2-5 seconds after the Pet has stepped on the pad.	Place a weight on the pad and use a timer to record the latency.

Camera Subsystem

	Requirements	Verification
Raspberry Pi 4 Model B	The Raspberry Pi must be able to power and connect to the camera.	Connect Raspberry Pi to Camera and see if Raspberry Pi can be programmed.
Raspberry Pi 4 Model B	The Raspberry Pi must capture photos through the camera in response to the pad load sensors	Connect Raspberry Pi to Camera and see if Raspberry Pi can trigger camera to take a photograph.
Camera	The camera should be able to take colored photos with a minimum of 720p resolution.	Take a photo with the camera and determine if it meets the quality requirements.

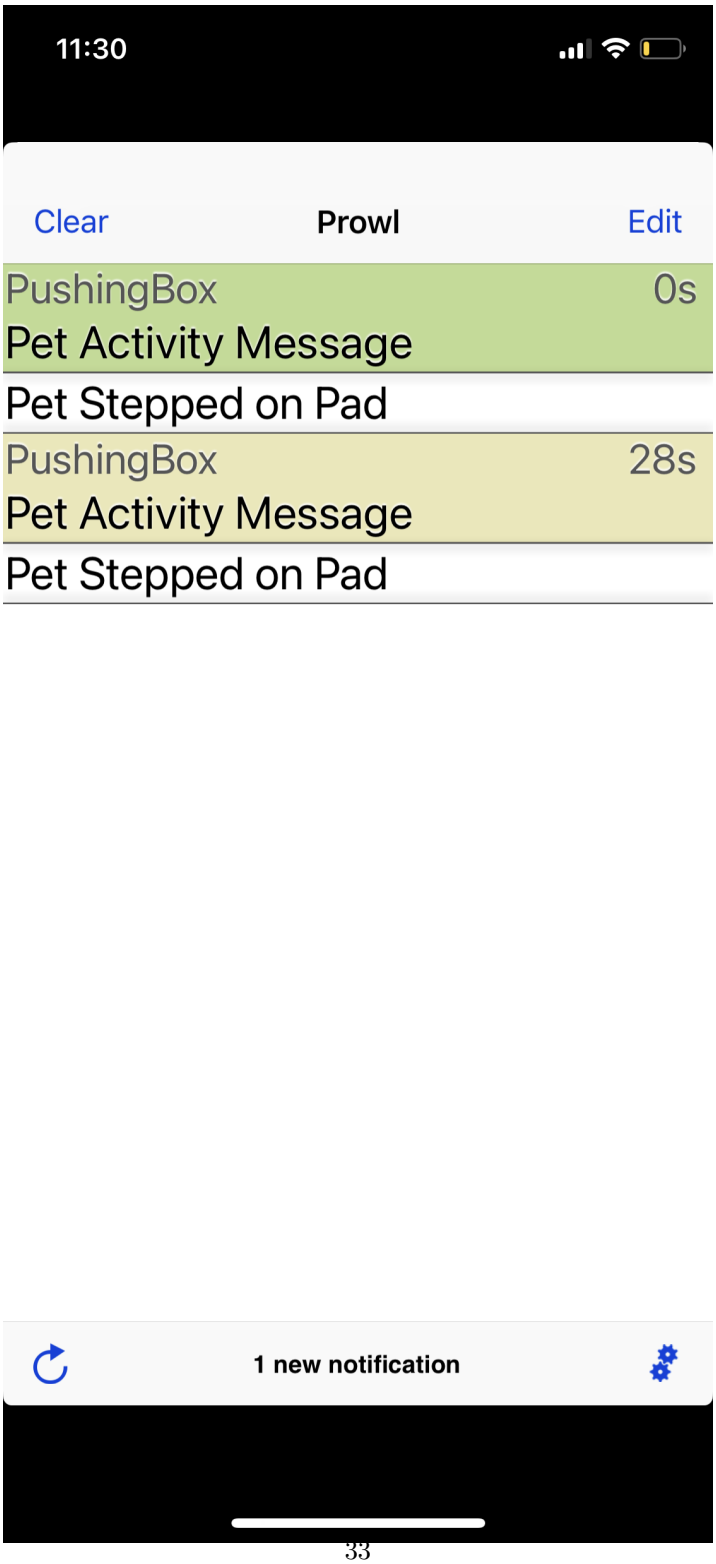
D Final Block Diagram



E Schedule

Project Schedule		
Week	Task	Person
September 26th - October 2nd	Finish Design Document	Everyone
October 3rd - October 9th	Order Parts for Prototype	Joseph
October 3rd - October 9th	Start PCB design Start Power System Design	Tyler
October 3rd - October 9th	Start 3D print designs	Abhi
October 10th - October 16th	Check Parts and Compatibility	Joseph
October 10th - October 16th	Finalize Power System Start Load Sensor	Tyler
October 10th - October 16th	Print First 3D print versions Start camera attachment	Abhi
October 17th - October 23rd	Finish PCB designs	Joseph
October 17th - October 23rd	Finish Load Sensor Pad/Bowl Implementation	Tyler
October 17th - October 23rd	Camera/Sensor Communication Print First 3D print versions	Abhi
October 24th - October 30th	Order PCB	Joseph
October 24th - October 30th	Revisions	Tyler
October 24th - October 30th	Mobile Device Communication	Abhi
October 31st - November 6th	PCB Revisions	Joseph and Tyler
October 31st - November 6th	Device notification system	Abhi
November 7th - November 13th	Finalize Notification system	Everyone
November 14th - November 20th	Mock Demo and Final Testing	Everyone
November 28th - December 4th	Final Demo	Everyone
December 5th - December 8th	Final Presentation	Everyone

F Notification System




AT&T
Sunday, December 4
11:33
27°
Mostly Cloudy
H:40° L:17°
9:00–9:30AM
Nehemiah Call
Prowl
now
PushingBox—Pet Activity Message
Pet Stepped on Pad
Swipe up to unlock

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(FSR_PIN, INPUT);
  pinMode(LED, OUTPUT);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

   while (WiFi.status() != WL_CONNECTED) {
       delay(500);
       Serial.print(".");
   }

   Serial.println("");
   Serial.println("WiFi connected");
   Serial.println("IP address: ");
   Serial.println(WiFi.localIP());
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  int fsrADC = analogRead(FSR_PIN);
  float force;
  if (fsrADC != 0) // If the analog reading is non-zero
  {
    Serial.println("Reading: " + String(fsrADC) );
    delay(500);
  }
  else
  {
    // No pressure detected
  }

  if (fsrADC > 900) {
    digitalWrite(LED, LOW);
    Serial.print("connecting to ");
    Serial.println(host);

    // Use WiFiClient class to create TCP connections
    WiFiClient client;
    const int httpPort = 80;
    if (!client.connect(host, httpPort)) {
        Serial.println("connection failed");
        return;
    }

    // We now create a URI for the request
    String url = "/pushingbox";
    url += "?devid=";
    url += devid;

    Serial.print("Requesting URL: ");
    Serial.println(url);

    // This will send the request to the server
    client.print(String("GET ") + url + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Connection: close\r\n\r\n");
    unsigned long timeout = millis();
    while (client.available() == 0) {
        if (millis() - timeout > 5000) {
            Serial.println(">>> Client Timeout !");
            client.stop();
            return;
        }
    }
```