LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title
CS5068NI– Cloud Computing & IoT

Pet Feeder

Assessment Type
10% Proposal Report

Semester
2024 Spring

Group Members

London Met ID	Student Name
23049051	Digdarshan Bhattarai
23049041	Nikhil Bhandari
23049181	Aseem Gautam
23048479	Ritika Khatri
23049062	Sirson Sharma Chapagain
23050193	Aakash Mandal

Assignment Due Date:
Assignment Submission Date:
Submitted to: Mr. Sugat Man Shakya
Word Count: 3628

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded

Table of Contents

Table of Figures

Tables of Tables

1. Introduction

Pets are a big part of many families around the world, with animals like dogs and cats being popular companions. Studies show that having pets helps people feel happier by reducing stress, anxiety, and depression. Pets also encourage their owners to exercise more and talk to others, making them great for improving health and relationships. However, one major problem arises when pet owners need to leave their pets alone due to work trips or other commitments. During such times, it becomes hard to maintain proper feeding schedules, leading to issues like overfeeding, underfeeding, or bad behavior. On top of that, many regular pet feeders are easy for pets to break, which can cause frustration and extra costs for the owners.

Pet feeder device can contribute huge to the owner of the pet. They can be stress free from when to fed, quantity amount to feed the pet, and can enjoy their vacation without worrying about their pet.

1.1 Current Scenario

According to the American Pet Products Association (APPA), 70% of U.S. households, or approximately 90.5 million families, own a pet as of 2023. Feeding schedules remain a top concern for pet owners, with surveys indicating that:

- 35% of pet owners struggle to feed their pets at regular times due to work commitments.
- 20% worry about overfeeding or underfeeding their pets in their absence.
- Traditional feeders are prone to tampering, with reports of pets breaking into feeders to access extra food.

The global smart pet feeder market experienced year-on-year (YoY) growth of 5.2% in 2021 to reach a market valuation of US$ 187.3 million. Worldwide sales of smart pet feeders are expected to surge at 7.8% CAGR to reach US$ 419.2 million by the end of 2032. We can see that pet feeder is growing in market too. (Mr, n.d.)

1.2 Problem Statement and Project as a Solution

Pet owners face challenges in maintaining consistent feeding schedules for their pets due to busy lifestyles, especially when they are away for work trips or other commitments. Traditional pet feeders also pose issues, as they are often not secure enough and can be easily tampered with or broken by pets, leading to overfeeding or spilled food.

Project as a Solution: The proposed IoT pet feeder addresses these challenges by:

- Automating feeding schedules that can be set and adjusted using a mobile app.
- Allowing manual feeding through app controls.
- Ensuring feeder security by mounting the device on walls, making it tamper-proof, and delivering food through pipes to prevent pets from tampering with the mechanism.
- Allowing pet owners to check on their pets eating food or not.

2. Aim and Objectives

2.1 Aim

The aim of this project is to design and develop a pet feeder device that automates feeding schedules, allows remote manual control, also calls the dogs by making sound when time to feed and ensures tamper-proof for pets like cats and dogs.

2.1 Objectives:

- Automated Feeding: Create a system that dispenses food at set times, with adjustable schedules.
- Manual Feeding Option: Include a feature that allows owners to manually feed their pets when needed.
- Durability and Safety: Ensure the design is robust and safe for pets to interact with.
- Sensor Integration: Integrate sensors, such as weight sensors, to monitor food levels and ensure the correct amount of food is dispensed.
- Signal the Pet: Making the sound when it’s time to feed the pets and call out them.
- Mobile App: We can control the dispenses times.
- Notify the owner: Notify the owner when food dispensed if dog eat or not using ultrasonic sensor.

3. Background

Pets are important members of many households, providing companionship and emotional support. However, maintaining consistent feeding schedules is a common challenge for pet owners, especially when they are away due to work, travel, or other commitments. Improper feeding routines can lead to health issues like overfeeding or underfeeding.

Traditional feeders lack advanced functionality and are often prone to tampering, making them unsuitable for long-term use. This project introduces a smart IoT-based pet feeder that automates feeding, enhances security, and provides real-time monitoring through a mobile app.

3.1 System Overview

This project focuses on developing a smart pet feeder that simplifies and improves the process of feeding pets. The system is powered by an ESP32 microcontroller, which acts as the main controller to manage all operations. A servo motor is used to dispense food at specific times set by the user through a mobile app. When it is time for feeding, buzzer notifies the pet that the food is dispensed. To ensure the pet actually eats the food, an ultrasonic sensor detects whether the food is consumed, and a load cell measure the amount of food in food container , preventing overfeeding or underfeeding.

The feeder is designed to be tamper-proof by incorporating a wall-mounted design. By securely mounting the feeder on a wall and delivering food through a pipe, the system prevents pets from damaging the feeder or accessing extra food. The feeder is controlled using a mobile app developed on the Blynk platform, which allows users to set feeding schedules, monitor their pet’s eating habits, and manually dispense food if necessary.

3.2 Design Diagram

3.2.1 Block Diagram

Diagram showing connections between the ESP32, servo motor, load cell, ultrasonic sensor, buzzer and power supply.

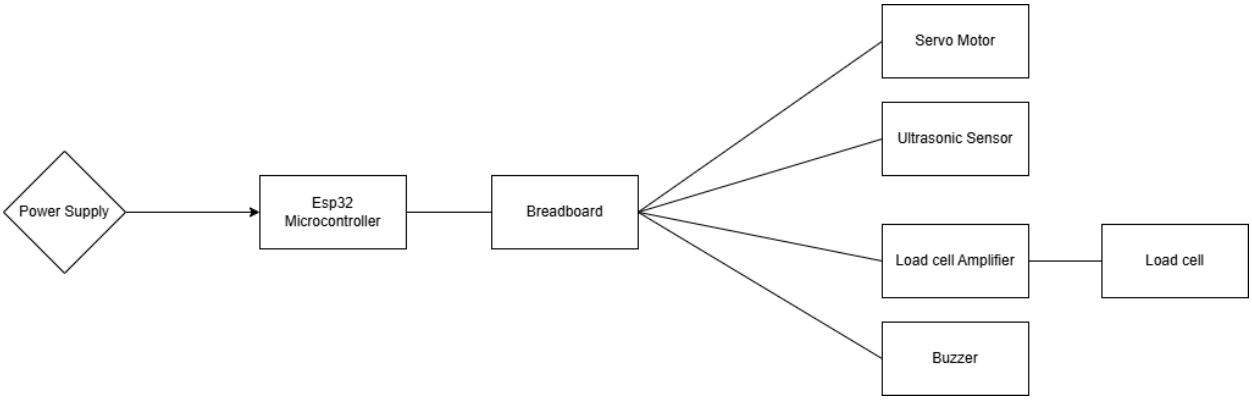

Figure 1: Block Diagram

3.2.2 Hardware Architecture

This shows the system architecture for pet feeder.

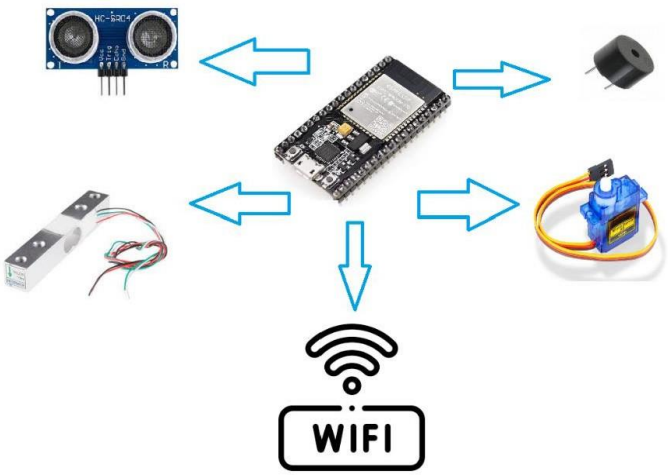

Figure 2: Hardware Architecture

3.2.3 Flowchart

This flowchart shows the working mechanism of pet feeder.

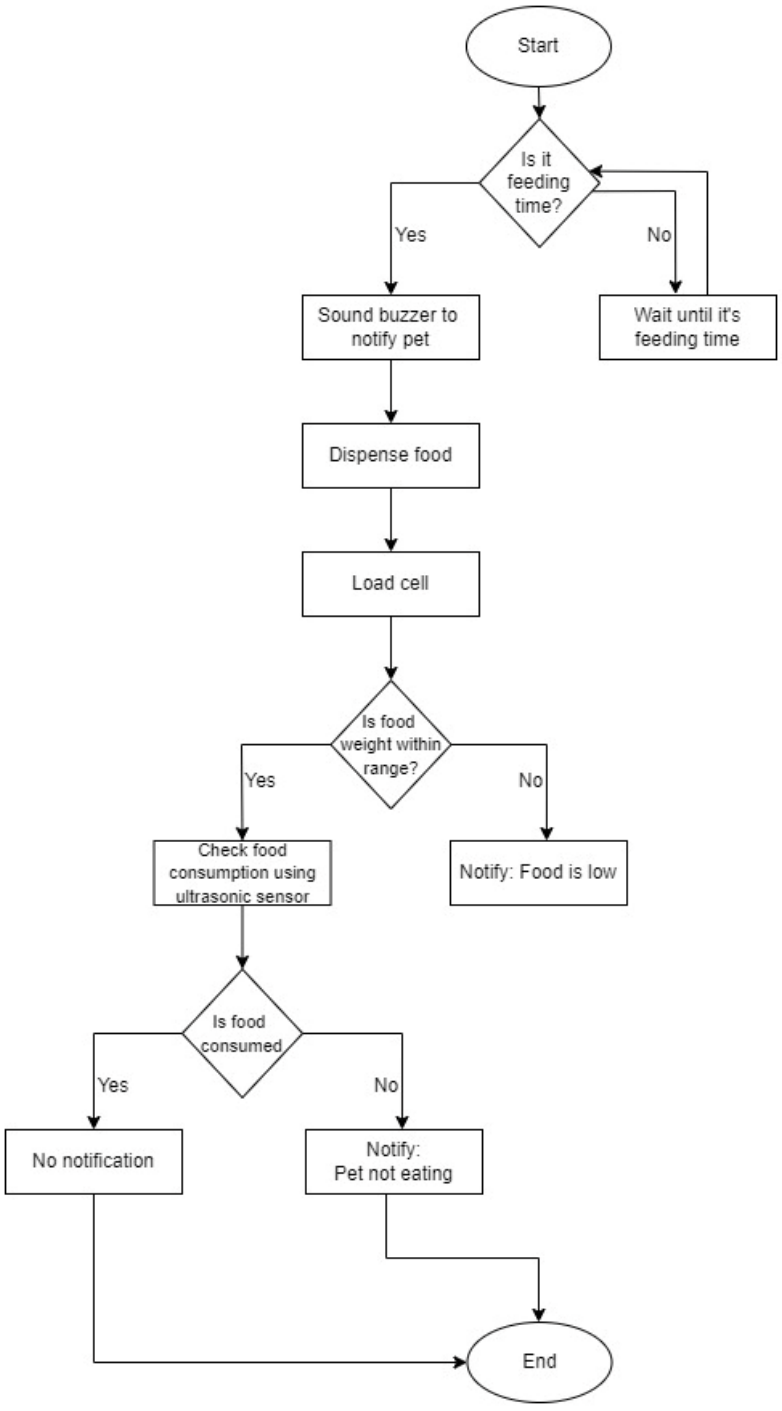

Figure 3: Flowchart

3.2.4 Circuit Diagram

Circuit diagram for Pet feeder.

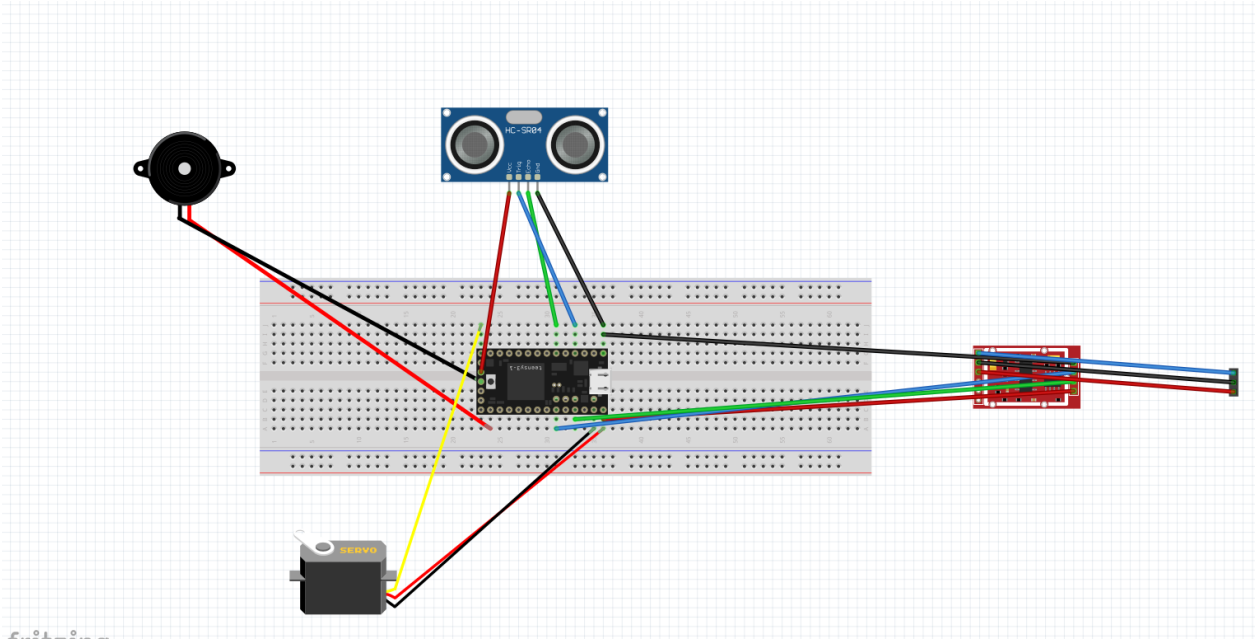

Figure 4: Circuit Diagram

3.2.5 Schematic

Schematic diagram of Pet feeder:

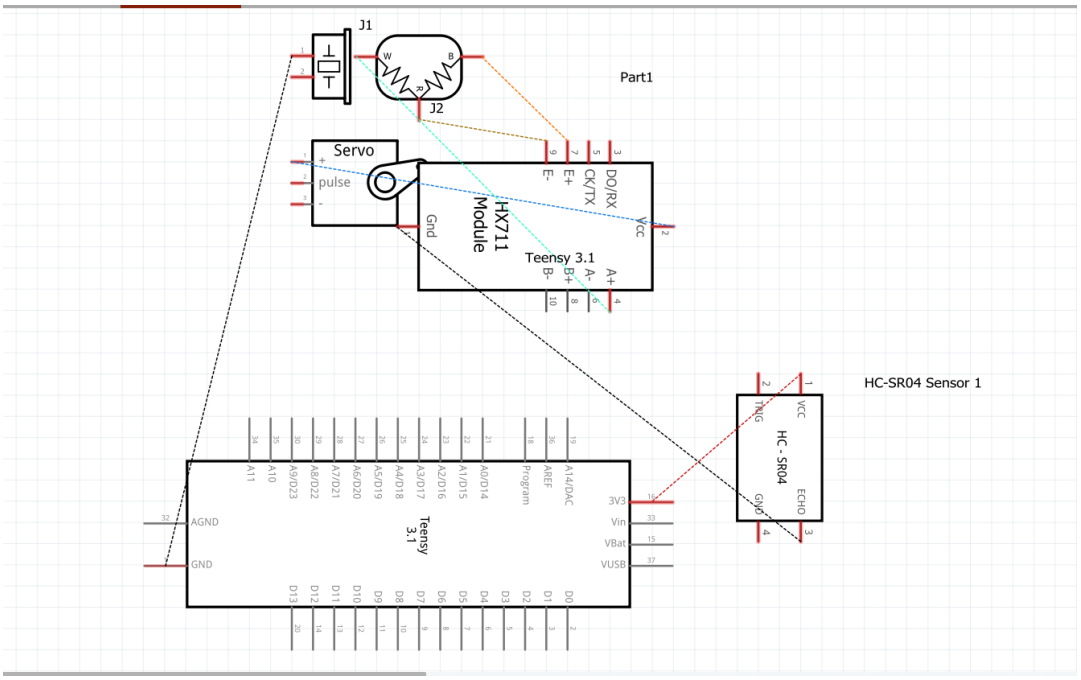

Figure 5: Schematic Pet Feeder

3.3 Requirement Analysis

3.3.1 Hardware Requirements

1. ESP32 Microcontroller

The ESP32 is the central processing unit of the system, equipped with built-in Wi-Fi and Bluetooth capabilities. It acts as the brain of the feeder, controlling all operations such as dispensing food, reading sensor data, and communicating with the Blynk app. The Wi-Fi capability ensures seamless remote control via the mobile app.

Figure 6: ESP32

2. Servo Motor

A servo motor provides precise rotational movement controlled by the microcontroller. It is used to dispense food in controlled portions at scheduled times. Its precision ensures accurate food delivery, helping to avoid overfeeding or underfe

Figure 7: Servo motor

3. Load Cell and Amplifier

A load cell measures the weight of food dispensed, while the amplifier boosts the signal for accurate readings. This measure the food in the container if its low it notify the owner that its low on food.

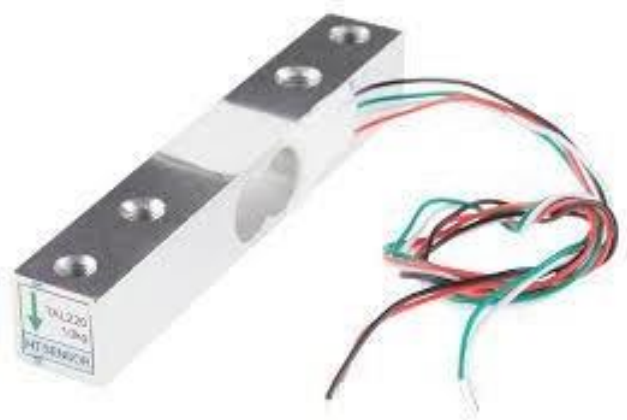

Figure 8: Load cell

4. Ultrasonic Sensor

An ultrasonic sensor measures distance by emitting sound waves and detecting their reflection. In this system, it is used to detect whether the food in the tray has been consumed by the pet. This information is relayed to the app, giving owners insight into their pet's eating habits.

Figure 9: Ultrasonic Sensor

5. Breadboard and Wiring

A breadboard is used for connecting components without soldering, and jumper wires facilitate these connections. These are used during the prototyping phase to simplify assembly and testing of the circuit.

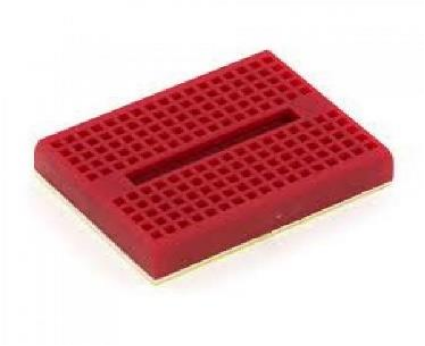

Figure 10: Breadboard

6. Buzzer

A buzzer is a small, simple output device that produces sound when an electric current is applied. In your IoT pet feeder project, it can be used to alert pets or owners at specific times, such as during feeding.

Figure 11: Buzzer

3.3.2 Software Requirements

1. Arduino IDE

A programming environment used to write and upload code to the ESP32 microcontroller. Develops and uploads the firmware that controls the hardware components, including the servo motor, sensors, and Wi-Fi communication. Manages the integration of sensor data, motor control, and Blynk connectivity.

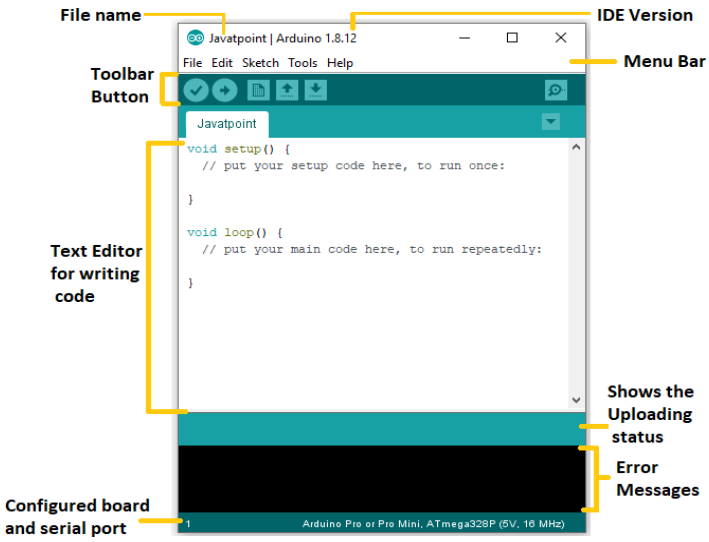

Figure 12: Arduino Ide

2. Blynk IOT

Blynk is a low-code Internet of Things (IoT) platform that allows users to build, manage, and remotely control connected devices. In this project we used it for control our pet feeder device . We can Dispense food anytime by click on the button in blynk app and we can set food dispense time in it too, we get notification through this app when food is eaten or not by pet. (Blynk, n.d.)

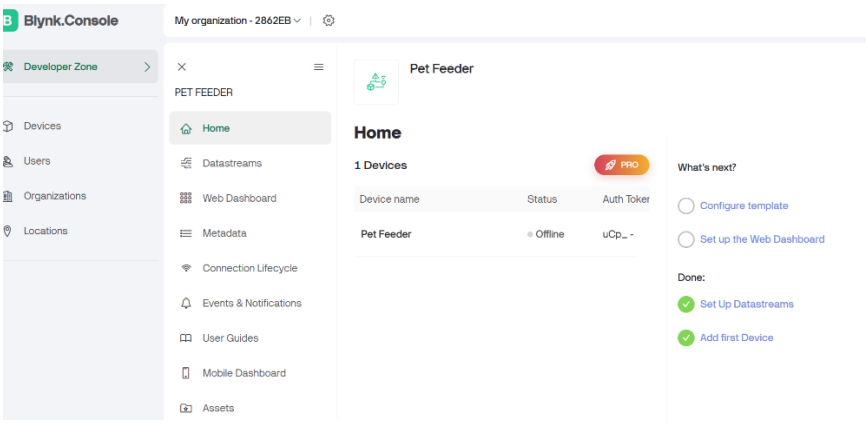

Figure 13: Blynk

3. Fritzing

Used for creating circuit diagrams and visualizing hardware connections. Design a detailed circuit diagram that includes the ESP32, servo motor, load cell, amplifier, ultrasonic sensor, and power adapter. Connect components virtually to show how they will be assembled on a breadboard or soldered onto a PCB. Use Fritzing to ensure all connections are correct before physically assembling the hardware. (Anon., n.d.)

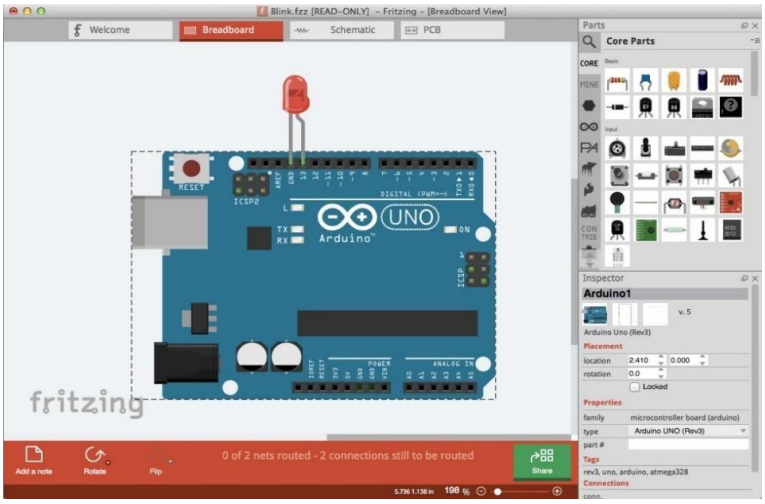

Figure 14: Frit

2. Draw.io

Ideal for creating flowcharts, block diagrams, and system architecture diagrams. Create a flowchart showing the logic flow of the system, such as the steps for dispensing food, monitoring sensors, and sending app notifications. Develop a block diagram to represent the overall system architecture, including the interaction between hardware (ESP32, sensors, motor) and software (Blynk app, firmware). Use sequence diagrams to show how commands flow between the app, ESP32, and hardware components.

3. MS Word

A tool for documenting your project, writing detailed descriptions, and compiling diagrams. Write your project report, including sections like the introduction, system overview, requirement analysis, and future work. Insert diagrams created in Fritzing and Draw.io to visually support your descriptions. Format the document with headings, bullet points, and tables for clarity and professionalism.

4. Development

4.1 Planning and Development

At first, we had a group meeting to discuss ideas for our project. We wanted to work on something unique that could solve a real problem. After a lot of brainstorming, we decided to make an automatic pet feeder. The idea was to create a device that could make life easier for pet owners by feeding their pets on time, even when they weren't home.

The initial idea was to create a wall-mounted device with a tamper-proof design to keep pets safe while securely delivering food through a pipe. To add functionality, we planned to include features like a load cell to monitor food levels, an ultrasonic sensor to detect if the food bowl was empty, and a camera to observe the pet's activity. Additionally, we thought of integrating a mobile app to allow pet owners to control feeding schedules, dispense food manually, and receive notifications.

4.2 Resource Collection

The main resources required to develop this IoT project are: ESP32 microcontroller, servo motor, ultrasonic sensor, load cell, load cell amplifier, breadboard, jumper wires and buzzer.

Some of the components, like the ESP32, servo motor, buzzer, and jumper wires, were provided by the college's resource department after submitting a request letter to our faculty coordinator, Mr. Shishir Subedi. Other materials, including the ultrasonic sensor, load cell, amplifier, and breadboard were purchased by the team members. Each member contributed to ensuring we had all the necessary components. To assemble and present the project, basic materials like chart papers, glue, tapes, and markers were also purchased by the team.

4.3 System Development

Phase 1: We set up the Blynk Platform

To configure the Blynk app for the IoT pet feeder, I first created an account on the Blynk platform and logged in to access the dashboard. Next, I created a new template for the project, where I defined the device type as ESP32 and specified key features of the feeder. I added two data streams: one for manual food dispensing and another for setting scheduled food dispensing times. Additionally, I navigated to the "Events and Notifications" section to create a custom event. This event triggers a notification to the owner whenever the ultrasonic sensor detects the pet nearby after food dispensed to monitor pet is eating food or not.This setup ensures smooth interaction between the app, sensors, and the ESP32, making the system user-friendly and effective. Get the important things to connect our ardiuno ide and blynk app (#define BLYNK_TEMPLATE_ID "TMPL6copVhtpu"

#define BLYNK_TEMPLATE_NAME "Pet Feeder"

#define BLYNK_AUTH_TOKEN "uCp_GteGFFIJXSvlZZqYx1BCjx-Hf264"). By the help of this we can connect with esp32.

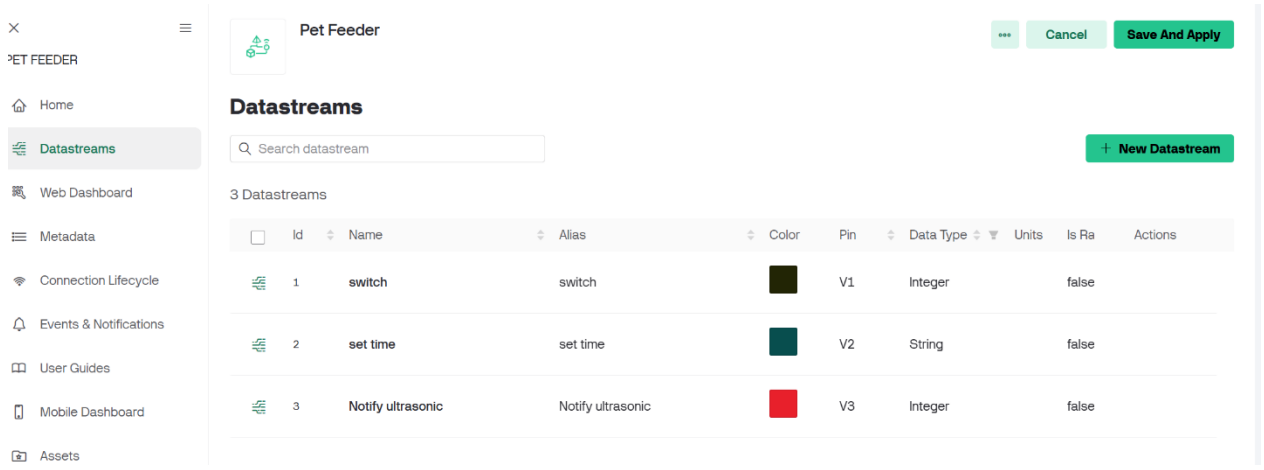

Figure 15: Blynk Development

Phase 2: Connecting ESP32 to Servo Motor for Scheduled Feeding

In Phase 2, I focused on connecting the ESP32 to the servo motor and implementing both manual and scheduled food dispensing features using the Blynk app. The servo motor was connected to the ESP32, with the signal pin wired to GPIO D12, and the power and ground pins connected to the ESP32's 3.3V and GND, respectively. I configured the Blynk app by assigning the V1 data stream for manual dispensing and the V2 data stream for scheduled feeding. In the code, the servo motor was programmed to rotate continuously for 10 seconds when triggered by the scheduled dispense (V2) feature. For the manual dispense feature (V1), the servo motor operated continuously while the Blynk app switch remained on and stopped as soon as it was turned off. And We test it and it worked fine like we imagine.

Phase 3: Buzzer for Notification During Feeding

In Phase 3, we added a buzzer to the system to notify the pet during feeding times. We connected the buzzer to the ESP32 using a breadboard, with the signal pin attached to GPIO D18 and the power and ground pins connected to the 3.3V and GND of the ESP32. In the Blynk app, we set it up so that when the scheduled feeding time (V2) was activated, the buzzer would beep for 2 seconds before the servo motor started dispensing food. This beep was meant to alert the pet that food was ready and also remind the owner about the feeding schedule. We tested the buzzer and servo motor together to make sure they worked in sync, and everything ran smoothly.

Phase 4: Ultrasonic Sensor for Pet Detection

In Phase 4, we added an ultrasonic sensor to the system to check if the pet is near the feeder. The trigger pin of the sensor was connected to GPIO 4, and the echo pin was connected to GPIO 5 on the ESP32, with the power and ground pins connected to the 3.3V and GND. The sensor works by measuring the distance and can detect if the pet is within 30 cm of the feeder. If the pet is nearby, it confirms the pet's presence, but if no pet is detected within 30 seconds of the food being dispensed, a notification is sent to the owner through the Blynk app. This notification uses the "dog_not_nearby" event to let the owner know that the pet hasn't approached the feeder. We tested this feature to make sure it detects accurately and sends notifications reliably. Adding the ultrasonic sensor made the system more helpful by keeping track of whether the pet is using the feeder, giving the owner extra peace of mind.

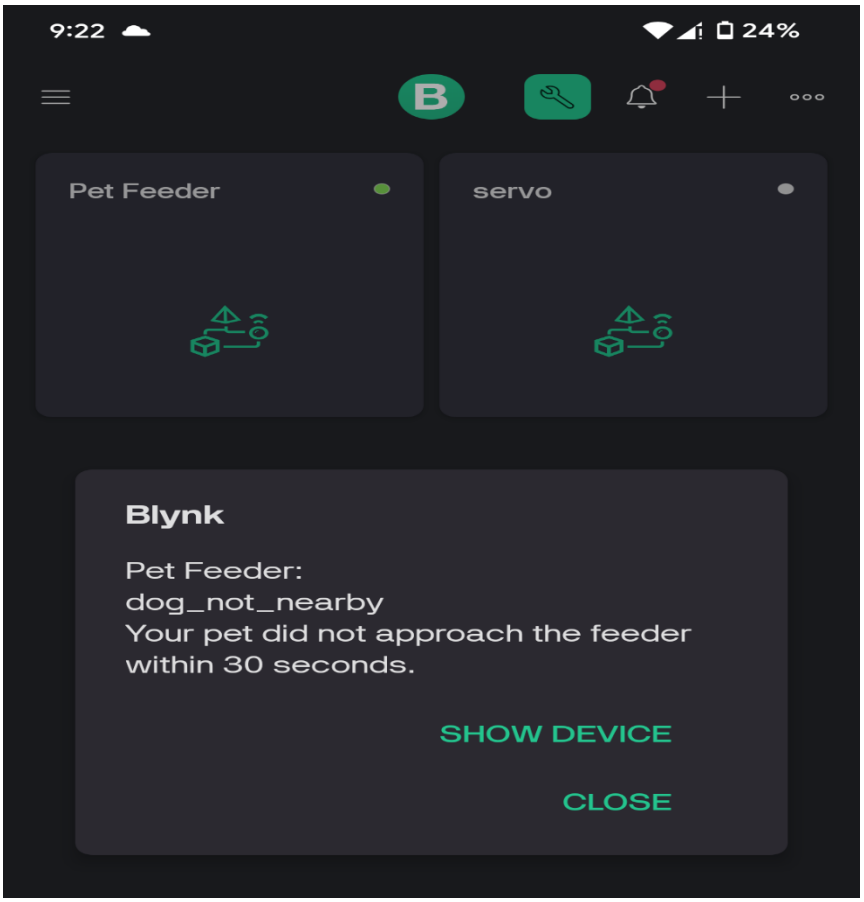

Figure 16: Notification Pet Detection

Phase 5: Integrating Load Cell for Monitoring Food Levels

In this phase, the load cell and its amplifier were integrated with the ESP32 to monitor the food levels in the dispenser. The load cell measured the weight of the food remaining in the container, and the ESP32 processed this data. The system was programmed to alert the owner when the food levels were low. The load cell was tested to verify the accuracy of its readings and to ensure the system provided real-time updates on food availability.

Phase 6: Final Design and System Testing

After all components were individually tested, the system was fully integrated. The ESP32, servo motor, buzzer, ultrasonic sensor, load cell, and Blynk app were connected and tested together as one unit. This final test ensured that the entire system worked as intended—automating feeding schedules, notifying the owner, and monitoring the pet's activity. The system was also tested for its durability, including tamper-proof features and secure wall mounting.

Design part of this Project:

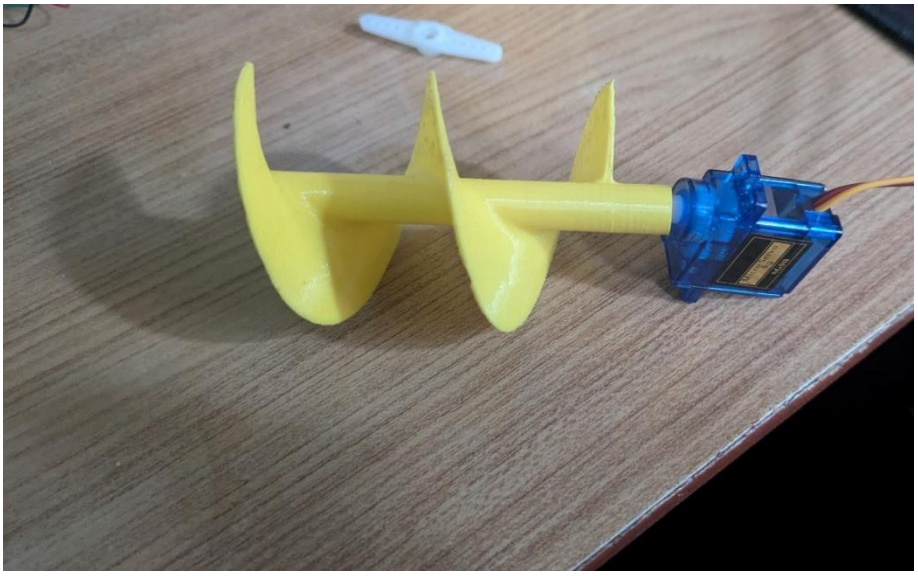

Figure 17: Worm Gear

Figure 18: Design

We attach servo motor with worm Gear. The objective of using a servo motor with a worm gear in this design is to ensure precise and controlled food dispensing, preventing unequal or accidental food distribution. The worm gear provides a self-locking mechanism that stops unwanted motion, ensuring the system only operates when triggered, making the feeding process reliable and accurate.

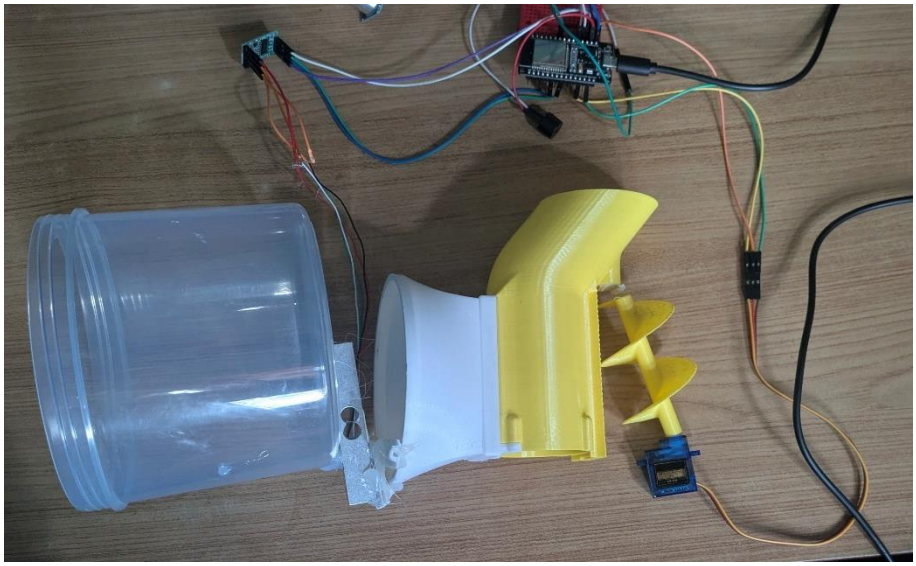

Figure 19: Design for Pet feeder

The design is there is food container to store Pet food which is monitor by the weight sensor if the food weight low it notifies the owner “the food is low”. The food is dropped in the white part, and it stuck in the yellow part with worm gear which holds the food. It only drops it when the servo motor rotates.

5. Result and Finding

Finding

At the end of the project, a fully functional **IoT-enabled Smart Pet Feeder** was developed. This system automates pet feeding schedules, provides real-time monitoring, and enables remote control through a mobile application created using the Blynk platform. The key functionalities include:

- **Automated Feeding**: The system dispenses food at scheduled intervals, adjustable via the app.
- **Manual Control**: Owners can manually dispense food through the app, providing flexibility.
- **Pet Monitoring**: An ultrasonic sensor used to monitor if food is consumed after the food is dispensed by detect the pet nearby at that time.
- **Tamper-Proof Design**: The feeder is wall-mounted, with food delivered via a pipe to prevent tampering by pets.
- **Sound Alerts**: A buzzer signals feeding time, helping pets recognize their feeding schedule.

The system provides pet owners with convenience and peace of mind, ensuring proper feeding even when they are away.

Results

Test 1:

Test Case	Load Cell Integration
Objective	To verify the load cell measures and monitors food weight accurately.
Activity	- Place food in the feeder and record the weight using the load cell. - Compare the measured weight with the actual weight of the food.
Expected Result	The load cell provides accurate readings of the food weight and transmits data to the app.
Actual Result	The load cell accurately measured the food weight.
Conclusion	The test 1 is successful.

Table 1: Test for load cell integration

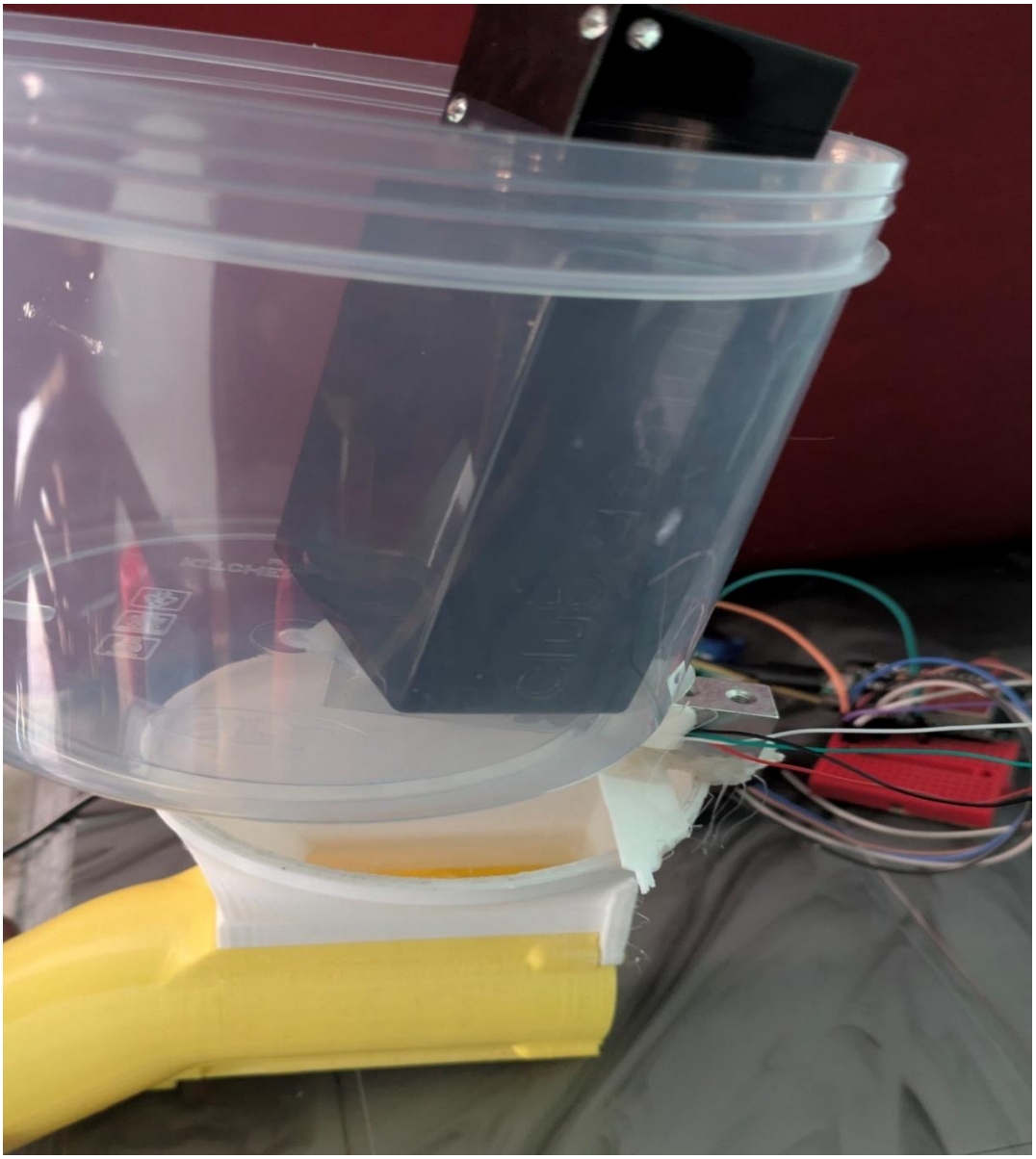

Figure 20: Testing Load cell

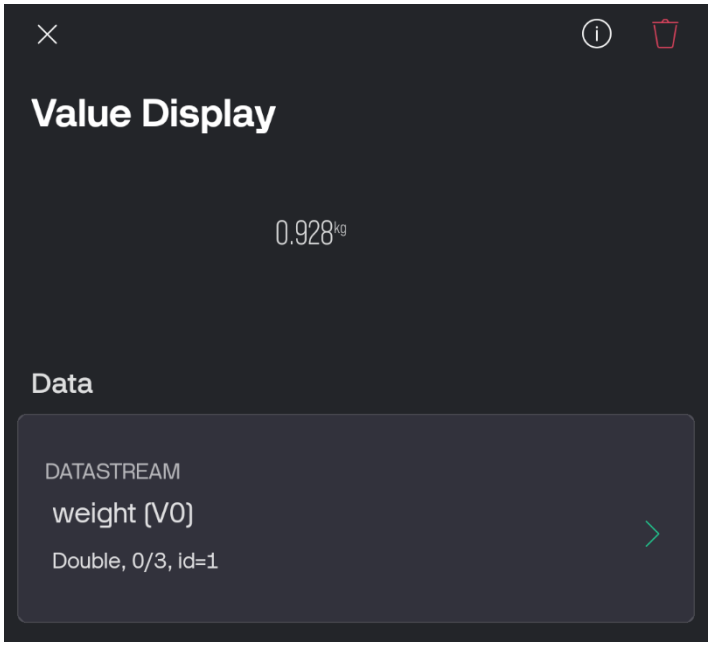

Figure 21: Weight Display in Blynk app

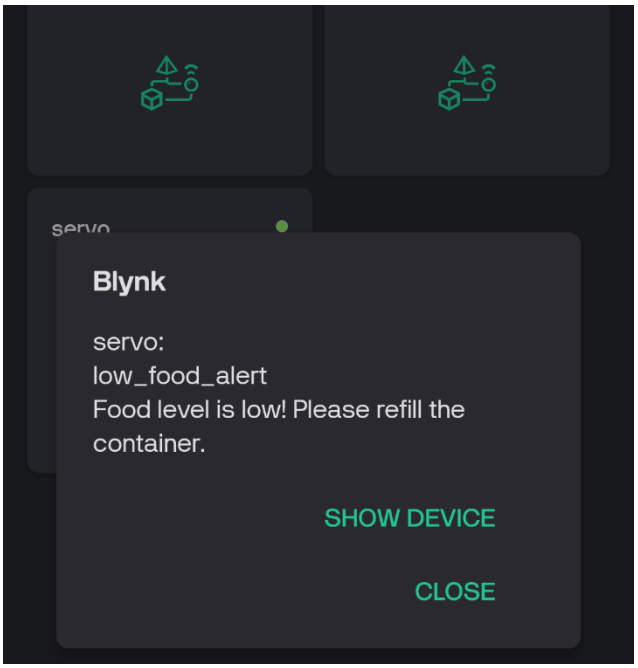

Figure 22: Getting notification food is low

Test 2:

Test Case	Food Dispensing Mechanism
Objective	To verify the servo motor dispenses food at scheduled times.
Activity	-Set a feeding schedule. - Observe the servo motor's roate 10 sesc at the scheduled time.
Expected Result	The servo motor rotates to release a food for 10 sec at scheduled time.
Actual Result	The servo motor dispensed the food for 10 sec.
Conclusion	The test is successful.

Table 2: Test for food dispensing mechanism

Figure 23: Food Dispense

Test 3:

Test Case	Buzzer Functionality
Objective	To test if the buzzer sounds at feeding times to alert the pet.
Activity	- Set a feeding schedule in the Blynk app. - Observe if the buzzer sounds at the scheduled feeding time.
Expected Result	The buzzer produces an audible sound at the set feeding time to alert the pet.
Actual Result	The buzzer functioned as expected, producing a sound at the set time.
Conclusion	Test 3 is successful.

Table 3: Test for Buzzer Functionality

Test 4:

Test Case	Ultrasonic Sensor Functionality
Objective	To verify that the system detects the dog's presence after dispensing food and sends a notification if the dog does not approach the feeder within 30 seconds.
Activity	- - Set a feeding schedule and dispense food. - Observe if the ultrasonic sensor detects the dog within 30 seconds after food dispensing.
Expected Result	If the dog is not detected within 30 seconds, a notification is sent to the owner stating, "Your dog did not approach the feeder within 30 seconds."
Actual Result	The system sent the notification when the dog was not detected within the time frame.
Conclusion	The test is successful.

Table 4: Test for Ultrasonic Sensor Functionality

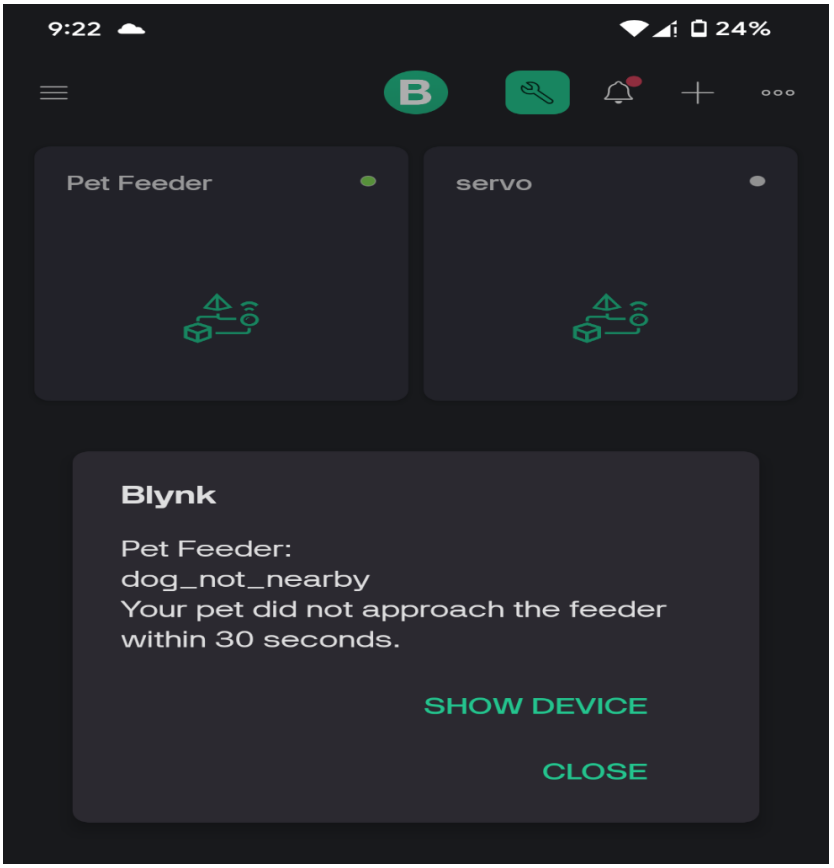

Figure 24: Ultrasonic Test

6. Future Work

In the future, adding a camera to the Smart Pet Feeder could be a great enhancement. This would allow pet owners to see what their pets are doing in real-time, helping them make sure the pet is eating the food and behaving well. The camera could be a small, low-power module that works with the ESP32 or similar microcontroller. The camera feed could be shown live on the mobile app, so the owner could easily check in on their pet. Additionally, motion detection could be added to alert the owner if the pet is near but not eating or if anything unusual is happening. This feature would bring peace of mind to pet owners, especially when they're away from home, and would make the pet feeder even more useful.

7. Conclusion

The IoT-based pet feeder system addresses some of the biggest challenges faced by pet owners, such as irregular feeding schedules, the risk of overfeeding or underfeeding, and pets tampering with traditional feeders. This project provides a smart solution by automating feeding times and giving pet owners the ability to control feeding remotely through an app. The wall-mounted design ensures that pets can't interfere with the system, and the sensors help track whether the pet has eaten, giving owners peace of mind.

The growing demand for smart pet feeders in the market shows that this solution is relevant and needed. By combining automation with manual control and including features like pet motioning and food level monitoring, this project helps make pet care more convenient and reliable. Overall, this smart pet feeder aims to improve the feeding routine of pets, making it easier for owners to manage their pets' needs, even when they are away.

References

Anon., n.d. *Fritizing.* [Online]
Available at: https://fritzing.org/
Blynk, n.d. *Blynk.* [Online]
Available at: https://blynk.io/
Mr, F., n.d. *Fact Mr.* [Online]
Available at: https://www.factmr.com/report/smart-pet-feeder-market

Appendix

Source code:

```
#define BLYNK_TEMPLATE_ID "TMPL6copVhtpu"
#define BLYNK_TEMPLATE_NAME "Pet Feeder"
#define BLYNK_AUTH_TOKEN "uCp_GteGFFIJXSvlZZqYx1BCjx-Hf264"

#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <ESP32Servo.h>
#include <HX711.h>

char auth[] = BLYNK_AUTH_TOKEN;  // Blynk authentication token
char ssid[] = "Nikhil 2.4";
char pass[] = "98448714002";

Servo myServo;
int servoPin = 12;  // GPIO pin connected to the servo
int buzzerPin = 18; // GPIO pin connected to the buzzer
int echoPin = 5;    // GPIO pin connected to the ultrasonic sensor echo
int trigPin = 4;    // GPIO pin connected to the ultrasonic sensor trigger

HX711 scale;        // HX711 instance
#define DOUT 4      // Data pin for HX711
#define SCK 3       // Clock pin for HX711

float calibrationFactor = -7050; // Calibration factor for the load cell
float weightThreshold = 50.0;    // Weight threshold (grams) to send a low-food alert
```

```
BlynkTimer timer;       // Timer to handle scheduled tasks

int scheduledHour = -1;  // Placeholder for scheduled hour
int scheduledMinute = -1;// Placeholder for scheduled minute
bool isRotating = false; // Variable to track if the servo should rotate

// Handle switch press on Virtual Pin V1
BLYNK_WRITE(V1) {
  int switchState = param.asInt();  // Get the switch state (0 or 1)
  if (switchState == 1) {  // If switch is ON
    isRotating = true;     // Start rotating
  } else {
    isRotating = false;    // Stop rotating
    myServo.write(90);     // Reset to neutral position (stop)
  }
}

// Handle Timer Input on Virtual Pin V2
BLYNK_WRITE(V2) {
  TimeInputParam t(param);
  if (t.hasStartTime()) {
    scheduledHour = t.getStartHour();
    scheduledMinute = t.getStartMinute();
  }
}

void rotateServoForDuration(int seconds) {
  digitalWrite(buzzerPin, HIGH); // Turn on the buzzer
  delay(2000);                   // Wait for 2 seconds
  digitalWrite(buzzerPin, LOW);  // Turn off the buzzer
```

```
  unsigned long startTime = millis();
  unsigned long duration = seconds * 1000;

  while (millis() - startTime < duration) {
    for (int angle = 0; angle <= 180; angle++) {
      myServo.write(angle);
      delay(20);
    }
    for (int angle = 180; angle >= 0; angle--) {
      myServo.write(angle);
      delay(20);
    }
  }
  myServo.write(90); // Stop the servo at neutral position
}

void checkScheduledTime() {
  int currentHour = hour();
  int currentMinute = minute();
  int currentSecond = second();

  if (currentHour == scheduledHour && currentMinute == scheduledMinute &&
currentSecond == 0) {
    rotateServoForDuration(10); // Rotate servo for 10 seconds
    scheduledHour = -1;
    scheduledMinute = -1;
  }
}

void checkWeight() {
```

```
  if (scale.is_ready()) {
    float weight = scale.get_units(); // Get weight in grams
    Serial.print("Weight: ");
    Serial.print(weight);
    Serial.println(" g");

    Blynk.virtualWrite(V3, weight); // Update weight on Blynk app (Virtual Pin V3)

    if (weight < weightThreshold) {
      Serial.println("Low food detected! Sending notification...");
      Blynk.logEvent("low_food", "Food level is low. Please refill the feeder.");
    }
  } else {
    Serial.println("HX711 not found.");
  }
}

void setup() {
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);

  myServo.attach(servoPin);
  myServo.write(90);

  pinMode(buzzerPin, OUTPUT);
  digitalWrite(buzzerPin, LOW);

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  scale.begin(DOUT, SCK);
```

```
  scale.set_scale(calibrationFactor); // Set calibration factor
  scale.tare();                       // Reset the scale

  timer.setInterval(1000L, checkScheduledTime);
  timer.setInterval(5000L, checkWeight); // Check weight every 5 seconds
}

void loop() {
  Blynk.run();
  timer.run();

  if (isRotating) {
    for (int angle = 0; angle <= 180; angle++) {
      myServo.write(angle);
      delay(20);
    }
    for (int angle = 180; angle >= 0; angle--) {
      myServo.write(angle);
      delay(20);
    }
  } else {
    myServo.write(90);
  }
}
```

```
  // Initialize load cell
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  scale.set_scale(2280.f);     // Set scale factor for your load cell (calibrate accordingly)
  scale.tare();               // Reset the scale to zero

  Serial.println("System Initialized!");
}

void loop() {
  // Read the weight from the load cell
  long weight = scale.get_units(10); // Average 10 readings

  // Print weight for debugging
  Serial.print("Weight: ");
  Serial.println(weight);

  // Check if the food level is below the threshold (adjust this value)
  if (weight < foodThreshold) {
    Serial.println("Food level low, dispensing food...");

    // Dispense food by moving servo
    feederServo.write(90);  // Rotate servo to 90 degrees (adjust for your feeder
mechanism)
    delay(2000);            // Wait for the servo to finish the action (adjust time as needed)

    feederServo.write(0);   // Reset servo back to initial position
    delay(1000);            // Wait before next action

    // Activate buzzer to notify feeding
    tone(BUZZER_PIN, 1000, 500);  // Buzzer sound (1000 Hz for 500 ms)
```

```
    // Wait to avoid continuous operation
    delay(5000);
  } else {
    Serial.println("Food level sufficient.");
  }

  // Add any additional functionality if needed, such as remote control or notifications

  delay(1000); // Delay to prevent excessive readings and delay the loop
}
```

Task Division

Member Name	Task
Nikhil Bhandari	- **Oversee project timelines and ensure milestones are met.** - **Coordinate communication between team members.** - **Background and System development part** - **Build Device mechanism.** - **Write and upload code to the ESP32 using Arduino IDE.** - **Helping making the hardware design.**
Digdarshan Bhattarai	- **Help in Arduino IDE.** - **Integrate the servo motor control with feeding schedules.** - **Test and debug the microcontroller code to ensure proper functionality.**
Sirson Sharma	- **Set up the Blynk app for remote control and scheduling.** - **Establish communication between the ESP32 and the Blynk app.** - **Test real-time commands and data exchange**
Aseem Gautam	- **Configure the 12V DC adapter to provide consistent power to all components.**
	- **Ensure proper wiring and connections for safe and efficient operation.** - **Test for power stability and manage potential issues like overheating or voltage drops.**
Aakash Mandal	- **Helping to build the device.** - **Integrate the ESP32 module with necessary components like motors and sensors.**
Ritika Khatri	-**Document the development process, including schematics, code, and app configurations. Prepare a project report and proposal.** -**Helping in building the device.**

Table 5: Task division

Individual Contribution Structure

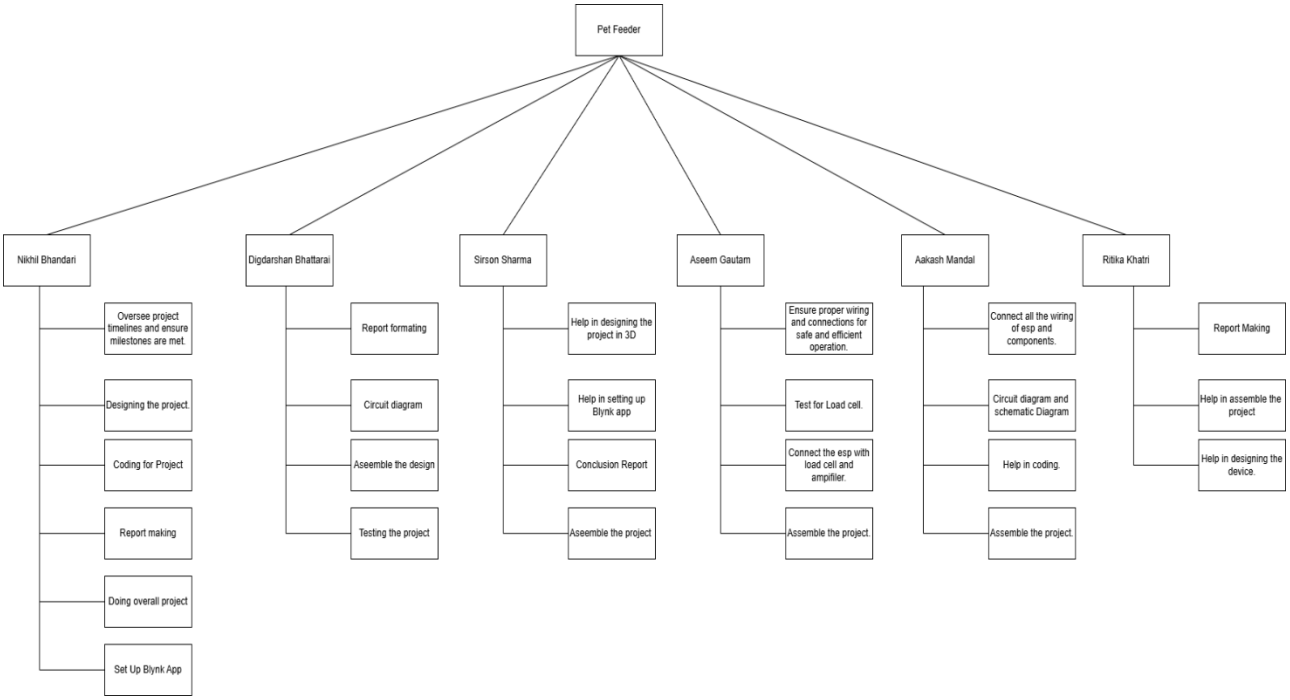

Figure 25:Individual Task

Project Photos

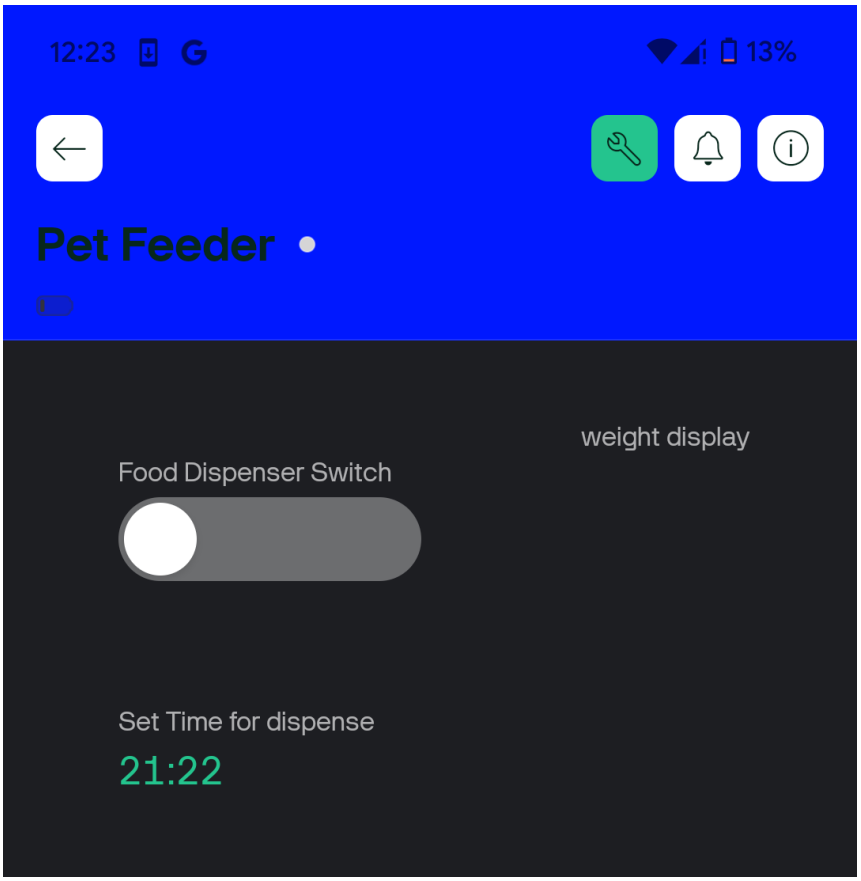

Figure 26: Blynk app

Figure 27:project Design

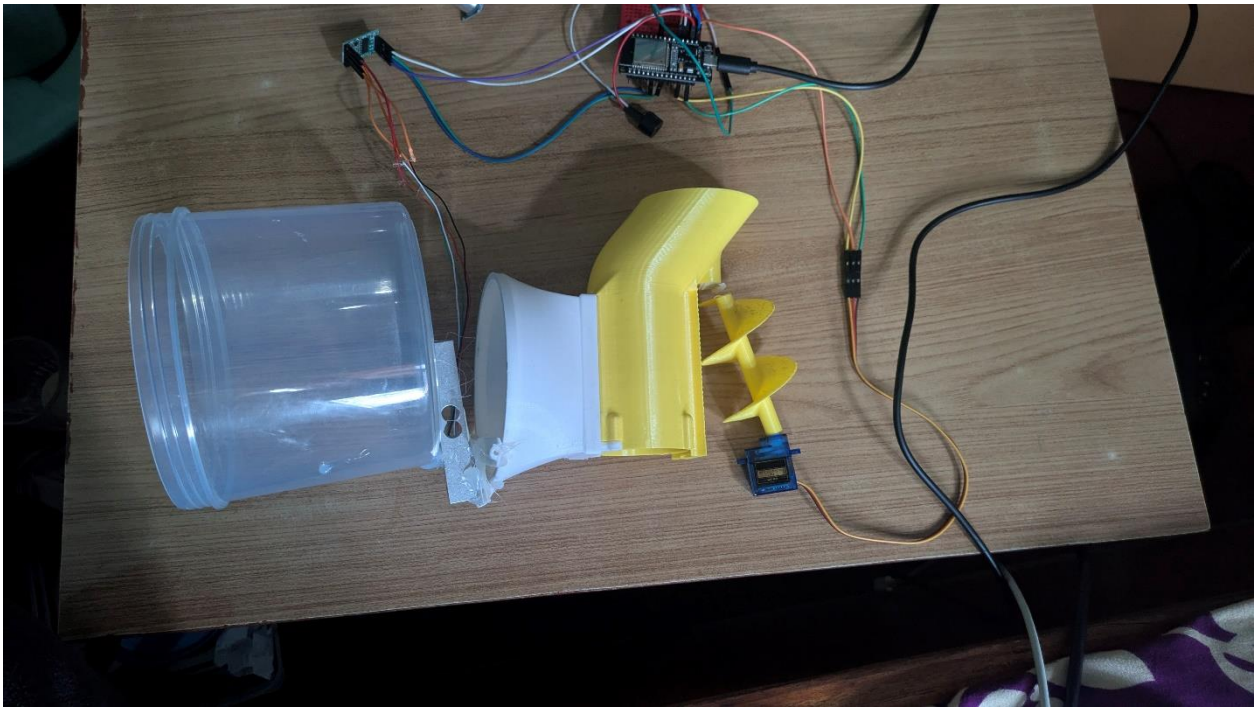

Figure 28:project Photos