PRO PAWS FEEDER FOR CATS AND DOGS USING ARDUINO AND TELEGRAM BOT

MUHAMMAD AMIRUL IMAN BIN HASSAN

Bachelor of Computer Science
(Computer System and Network)

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	: MUHAMMAD AMIRUL IMAN BIN
Date of Birth	
Title	: PRO PAWS FEEDER FOR CATS AND DOGS USING ARDUINO AND TELEGRAM BOT
Academic Session	:

I declare that this thesis is classified as:

☐	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐	RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑	OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

(Student's Signature)	(Supervisor's Signature)
	Ts. Dr. Zuriani Bt Mustaffa
New IC/Passport Number	Name of Supervisor
Date: 11 July 2023	Date: 27 July 2023

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

SUPERVISOR' S DECLARATION

I hereby declare that I have checked this project, and, in my opinion, this project is adequate in terms of scope and quality for the award of the degree of *Doctor of Philosophy/ Master of Engineering/ Master of Science in ………………………….

____

(Supervisor' s Signature)

Full Name	: Ts. Dr. Zuriani binti Mustaffa
Position	: Senior lecturer
Date	: 27 July 2023

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

STUDENT' S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

____

(Student' s Signature)

Full Name	: MUHAMMAD AMIRUL IMAN BIN HASSAN
ID Number	: CA20160
Date	: 27 OCTOBER 2022

PRO PAWS FEEDER FOR CATS AND DOGS USING ARDUINO AND TELEGRAM BOT

MUHAMMAD AMIRUL IMAN BIN HASSAN

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Computer Science (Computer System and Network) with Honors

Faculty of Computing

UNIVERSITI MALAYSIA PAHANG

OCTOBER 2022

ACKNOWLEDGEMENTS

I want to specifically thank my supervisor, Ts. Dr. Zuriani Binti Mustaffa, for giving me the chance to work on this wonderful project with her on the topic of Pro Paws Feeder. Her support also enabled me to conduct a lot of research and learn a lot of new information, for which I am extremely grateful. I also want to express my gratitude to my parents and friends, who greatly assisted me in completing this project in the allotted time.

ABSTRACT

This thesis aims to address the problem of pet owners facing difficulties in providing adequate food for their pets during extended periods of absence. The study proposes the development of a cost-effective solution, the Pro Paws Feeder (PPF) application, which functions similarly to expensive pet feeders available in the market. The PPF can be controlled remotely using a Telegram bot, with an Arduino Uno serving as the main controller connected to an online module for accessibility. The PPF is constructed using recyclable materials and incorporates a low-power motor to ensure a sustainable environment. The objectives of the research include studying existing pet feeders, designing, and developing the PPF to be affordable and efficient for pet owners, and evaluating its functionality. The scope encompasses users aged between 18-80 years with Telegram-enabled devices, primarily focusing on canine and feline owners. The development involves utilizing IoT technology and employing Arduino and C programming techniques. Upon completing this project, it shows that the PPF can fully functioned as intended. It feeds, read the status of the food, release food by its portion and many more without any problem at all. Telegram integration with PPF has been fully utilised and will be improved from time to time to properly function as we want it should be.

ABSTRAK

Tesis ini bertujuan untuk mengatasi masalah pemilik haiwan peliharaan yang menghadapi kesulitan dalam menyediakan makanan yang mencukupi bagi haiwan peliharaan mereka ketika mereka pergi bercuti untuk jangka masa yang panjang. Kajian ini mencadangkan pembangunan satu penyelesaian yang berpatutan, aplikasi Pro Paws Feeder (PPF), yang berfungsi serupa dengan pemakan haiwan peliharaan yang mahal yang terdapat di pasaran. PPF boleh dikawal jauh menggunakan bot Telegram, dengan Arduino Uno berfungsi sebagai pengawal utama yang disambungkan ke modul atas talian untuk kemudahan akses. PPF ini dibina menggunakan bahan kitar semula dan mempunyai motor berkuasa rendah untuk memastikan persekitaran yang mampan. Objektif penyelidikan ini termasuk mengkaji pemakan haiwan peliharaan sedia ada, merancang dan membangunkan PPF agar berpatutan dan cekap untuk pemilik haiwan peliharaan, serta menilai fungsinya. Lingkupnya melibatkan pengguna berumur antara 18-80 tahun dengan peranti yang menyokong Telegram, terutamanya pemilik anjing dan kucing. Pembangunan ini melibatkan penggunaan teknologi IoT dan menggunakan teknik pemrograman Arduino dan C. Setelah menyelesaikan projek ini, ia menunjukkan bahawa PPF berfungsi sepenuhnya seperti yang dijangka. Ia memberi makan, membaca status makanan, melepaskan makanan mengikut saiz bahagian, dan banyak lagi tanpa sebarang masalah. Integrasi Telegram dengan PPF telah digunakan sepenuhnya dan akan diperbaiki dari semasa ke semasa untuk berfungsi sebaiknya seperti yang diingini.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Every weekend, people usually travel to their hometown or for a short vacation. Sometimes people also travel for a very long time especially during long weekends and public holidays such Eid Fitr, Chinese New Year and Deepavali which takes longer duration of leaving their home. This could be very worrisome and concerned by pet’ s owner. Some pets such as reptile like snake and geckos are not the one to worry about because of their eating habits that eat once a week. But some pet such cats and dogs need to eat almost three times a day or maybe more. Currently, there are many pet’ s hotels available, however pet’ s hotels are mostly expensive especially for a long period of time. Not all pet’ s owner could afford to send their cats and dogs there specifically for those who own many pets. In addition, most of the existing pet feeder are too expensive to afford by Malaysian pet owners. Besides, during holiday season, the pet’ s hotel is usually fully booked by the customers.

Concerning that matter, this project purpose a Pro Paws Feeder (PPF) application that is a lot cheaper an able to function as nearly the same as the expensive existing pet feeder existing in the market. The PPF can be controlled online using Telegram bot. In this project, Arduino Uno will be use as the main controller attach to online module to make sure it available most of the time to be access by the pet’ s owner. The PPF will be built from recyclable material such container and requires a low-power motor, which will make the PPF will sustain a good environment. The owner does not need to worry about starving their pets when they are not available at home.

1.2 Problem Statement

Many people love to go back to their hometown or having short vacation during weekend. On occasion, people will travel for a week or more straight during a public holiday. The owner of the pet can be very anxious and concerned about this. Some species, such reptiles like snakes and geckos, only eat once a week, which is easy to take care of and does not require higher maintenance. However, certain pets, like cats and dogs, require eating up to three times every day. Hotels for pets are typically rather expensive, especially for longer stays. Not all pet owners had the means to transport their canines and felines there. Additionally, the majority of the available pet feeders are too expensive for Malaysian pet owners to afford.

Due to that matter, this project aims to create Pro Paws Feeder (PPF) application that is significantly less expensive and can perform nearly identically to the pricy existing pet feeders now available on the market. A Telegram bot can be used to control the PPF online. In order to ensure that the online module is frequently accessible by the pet owner, the main controller for this project will be an Arduino Uno attached to an online module. Because it uses a low-power motor and is constructed from recyclable materials like plastic bottles, the PPF will contribute preserve the environment. When the pet is not at home, the owner does not have to worry about starving them.

1.3 Objective

The aims to carry out this project work are:

i. To study and review the existing pet feeder on the market.

ii. To design and develop Pro Paws Feeder that are affordable yet efficient to use by pet’ s owner.

iii. To evaluate the functionality of PPF.

1.4 Scope

PPF are easy to use at all levels of knowledge. Since it is integrated with Telegram Bot, it complies with Telegram usage policy that stated it is not suitable for children to use. PPF can be use on mobile phone either on Android or iOS devices on supporting Telegram applications.

Scope of the project consists of:

User scope:

i. Age between 18-80 years old that have communication devices that support Telegram
ii. Canines and Felines owners

System scope:

i. Covers animals’ daily needs (food)

Development scope:

i. Involving IoT
ii. Using Arduino and C/C++ programming techniques

1.5 Thesis Organization

This thesis consists of five chapters. Chapter 1 discuss about the introduction of this project. Chapter 2 discuss about literature review of this project. Chapter 3 discuss about methodology. Chapter 4 discuss about implementation and testing. Lastly, Chapter 5 discussing about conclusion of this project.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

There are a lot of existing Pet Feeder available on the market. Existing pet feeder provide various design, size, and program. Despite, we know that pet feeder serves one purpose, that is to feed the pet efficiently. To develop the best and efficient pet feeder with minimum expenses, first we need to analyse a few of existing pet feeder that might have their advantages to be maintained and disadvantages to be reduces. In these chapter, comparison, and research about a few of existing pet feeder available and commercialise on the market today are discussed.

2.2 Existing Systems

These are the current existing pet feeder advantages and disadvantages.

2.2.1 EPetz Smart Pet Food Feeder

Figure 2.1 shows the EPetz Smart Food Feeder and followed by its details.

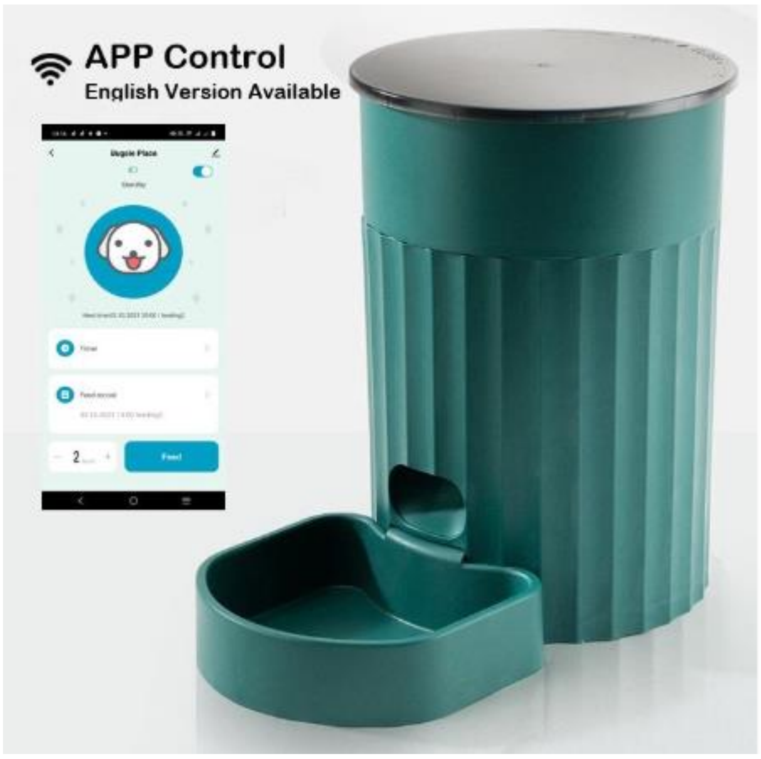


Figure 2.1 EPetz Smart Pet Feeder

The EPetz Smart Pet Food Feeder is a smart that can feed cats dogs and rabbit. It can also be connect used via a data hotspot. Therefore, it is not required for Wi-Fi to always be on at home. The feeding schedule will be saved in the memory setting once it has been configured via the app. The EPetz Smart Pet Food Feeder will not become inoperable if the Wi-Fi goes down. In truth, planned feeding will still occur without Wi-Fi, but it won't be recorded on the app [1]. The features of this product are the storage capacity up to 3L, a DC 5V USB converter power supply and four AA batteries serve as a dual power source, has smart app control and it can customise feeding plan.

2.2.2 Xiaomi PETKIT Smart Pet Feeder

Figure 2.2 shows the Xiaomi PETKIT Smart Pet Feeder and followed by its details

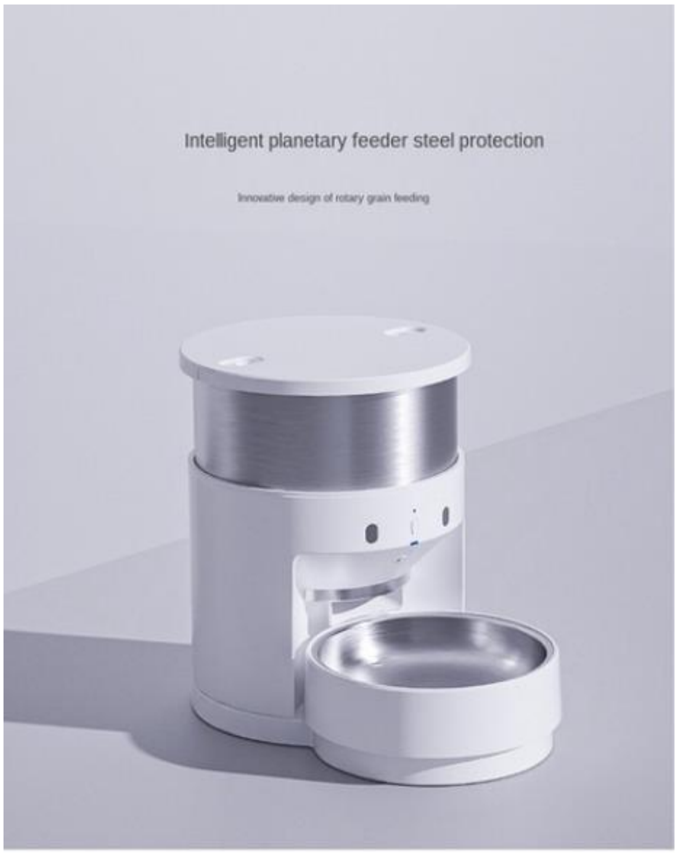

Intelligent planetary feeder steel protection
Innovative design of rotary grain feeding

Figure 2.2 Xiaomi PETKIT Smart Feeder

The revolutionary rota-table bowl on the Xiaomi PETKIT Smart Pet Feeder makes sure that food is distributed evenly inside the bowl without spilling out. In addition, the Xiaomi PETKIT Smart Pet Feeder incorporates a concealed food outlet that is quickly covered when food is delivered to stop pets from biting and damaging the food dispenser. Additionally, there is a low food sensor that alerts owners, keeping pets from going hungry. The features of this product are the base has a built-in weighing sensor which can accurately control the number of grains each time, able to customize feeding plan, rechargeable built-in lithium battery and have three lock system to keep the food fresh.

2.2.3 Ramboo Nanny Pet Smart Food Feeder

Figure 2.3 shows the Ramboo Nanny Pet Feeder and followed by its details

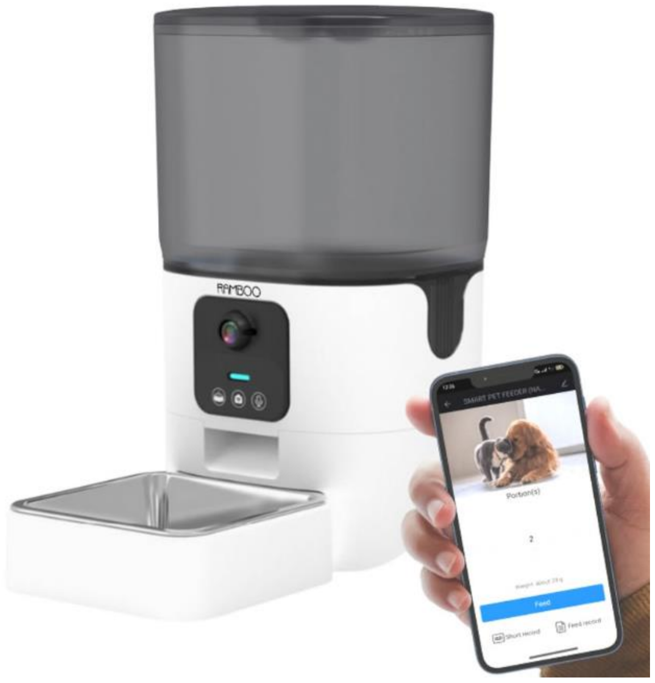


Figure 2.3 Ramboo Nanny Pet Feeder

Pet owner are increasingly using pet food dispensers with cameras. It enables them to communicate, bond, and keep an eye on their pets from a distance. Long trips away from pets will be much more tolerable thanks to the 1080P 90-degree wide angle camera and two-way speech connection offered by the Ramboo Nanny Pet Feeder. The nicest aspect is how reasonably priced this 6L feeder is. The features of this product are the feeder can be control via Smart app, able to customize feeding plan, available record audio, take pictures and videos feature and it able to share control using the same link to operate the dispenser at the same time with our family.

2.3 Comparison of the existing system

2.3.1 Analysis of comparison on existing system

A few research has been made and these are the three existing system available on the market and PPF itself in comparison of their features. showed in Table 2.1.

Table 2.1 Features of existing system

Features	**EPetz Smart Pet Food Feeder**	**Xiaomi PETKIT Smart Pet Feeder**	**Rambo Nanny Pet Automatic Food Feeder**
Power options	Power adapter and battery	Power adapter and battery	Power adapter and battery
Connectivity	Wi-Fi connection	Wi-Fi connection & Bluetooth	Wi-Fi connection
Price	RM 250++	RM 900++	RM 600++
Tank	3.5 Litre opaque	3 Litre opaque	6 Litre transparent
Food type	Dry food	Dry food	Dry food
Feeding type	Timer and program	Timer and program	Timer, program & direct feeding with button on food feeder
Camera	Absent	Absent	Exist for live view
Sensor	Food level sensor	Pet tracking sensor for feeding, pressure sensor to avoid pet biting & food level detection	Pet tracking sensor for feeding & food level detection
Maintenance	Removable AA battery need replacement	Not required	Not required

2.3.2 Relevance of comparison with project title

From the analysis comparison in the analysis in Table 2.1, a few things to be implemented onto the PPF to ensure it can be sustaining the goods and avoid the bad. For the good things, it is promising that I should retain the functioning element from the existing pet feeder on the market. PPF main elements that need to have been the online connectivity and two power options. This is to ensure that PPF can be access anywhere and anytime without any problem. In addition, for the things to avoid that are excessive power input and expensive material usage. As the reason to this as PPF should not consume a lot of power because of it is going to be use for a very long time.

2.4 Summary

From this chapter, I can conclude a few findings that are useful for this project. One of it is every product have its own advantages and disadvantages regardless of their brand and price. Not all cheaper brands have bad product and not all pricey brands have all the best item. Therefore, in my system, I will take what is good to be implemented and try to avoid the things that does not give any benefit or may cause problem to pet owner and the system itself.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter will discuss about the methodology that will be implemented in developing the PPF. It includes steps and how to make it work from time to time in a planned way. This chapter also covered the Software Development Lifecycle (SDLC), project requirement, proposed design, data design, proof of initial concept, testing plan, potential use of purposed solution and Gantt chart.

3.2 Project Management Methodology

A technique for producing software that is of the highest quality and least expensive in the shortest period is the Software Development Life Cycle, or SDLC. An organisation may quickly create high-quality software that has been thoroughly tested and is ready for use in production thanks to the SDLC's well-organized phases [2]. Common SDLC models include the waterfall model, spiral model, V-model, and agile model. Figure 3.1 illustrates the SDLC cycle.

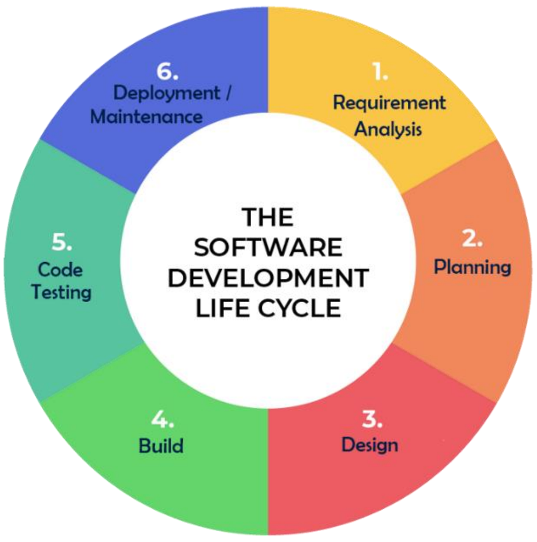


Figure 3.1 SDLC

SDLC works by reducing software development costs while both raising quality and speeding up production. By adhering to a strategy that eliminates the typical hazards of software development projects, SDLC accomplishes these seemingly incompatible goals. This plan begins by looking for flaws in the current systems. The requirements for the new system are then defined. After that, it goes through the processes of analysis, planning, design, development, testing, and deployment to construct the software.

Figure 3.2 shows the Agile methodology and its stages in development process on each stage.

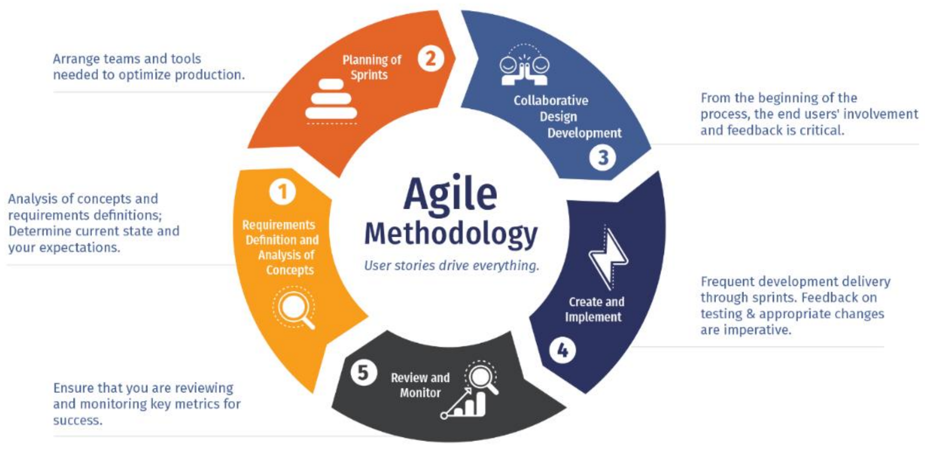

Agile Methodology
User stories drive everything.
1 Requirements Definition and Analysis of Concepts
Analysis of concepts and requirements definitions; Determine current state and your expectations.
2 Planning of Sprints
Arrange teams and tools needed to optimize production.
3 Collaborative Design Development
From the beginning of the process, the end users' involvement and feedback is critical.
4 Create and Implement
Frequent development delivery through sprints. Feedback on testing & appropriate changes are imperative.
5 Review and Monitor
Ensure that you are reviewing and monitoring key metrics for success.

Figure 3.2 Agile Methodology

For this project system development, Agile method will be selected as the SDLC to be implemented. The justification of selecting Agile is it combines incremental and iterative strategy, releasing the product on a continuous cycle and testing and improving it at each iteration [3]. Firstly, we need to define the needs in this phase. Began by I list out and describing the business potential and how to schedule the time and effort required to complete the PPF project. This data let me evaluate the technical and financial viability.

In phase two, I work with stakeholders to define needs once the project has been determined. Given that my project is not the first on the market, I may build and construct my system using the feedback from the existing pet feeder. This is to demonstrate the functionality of new features and how they will integrate with my current system.

Once I determine what is required, my work can begin [8]. As the designers and developers, I start working on this project to deliver a viable product within the allocated time is the aim of the designers and developers. The project has minimal functionality and will go through many rounds of improvement. As for this, user point of view is very critical as I put my friends to be the user as they also own pets at their home.

In the agile technique, quick failure is encouraged. The idea is that if we fail quickly and early, we can address little problems before they become larger ones. As this project is new to me to explore, I am expecting that I will explore more and there are possibilities of failing a few times and ensure I get back to solve it from time to time as it can develop my problem-solving skills and therefore making the system a lot better than what has been planned for.

Last but not least, the review and monitoring phase in Agile method, I implement it during the development of PPF. I need monitor every specification and functionality of this system to ensure that this system will run smoothly and function as intended.

3.2.1 Project Requirement

The project requirement for this system is consists of a few things such its Functional requirement [4], Non-functional requirement, constraint, and limitation [5] of this system. Table 3.1 shows the requirement type of this projects.

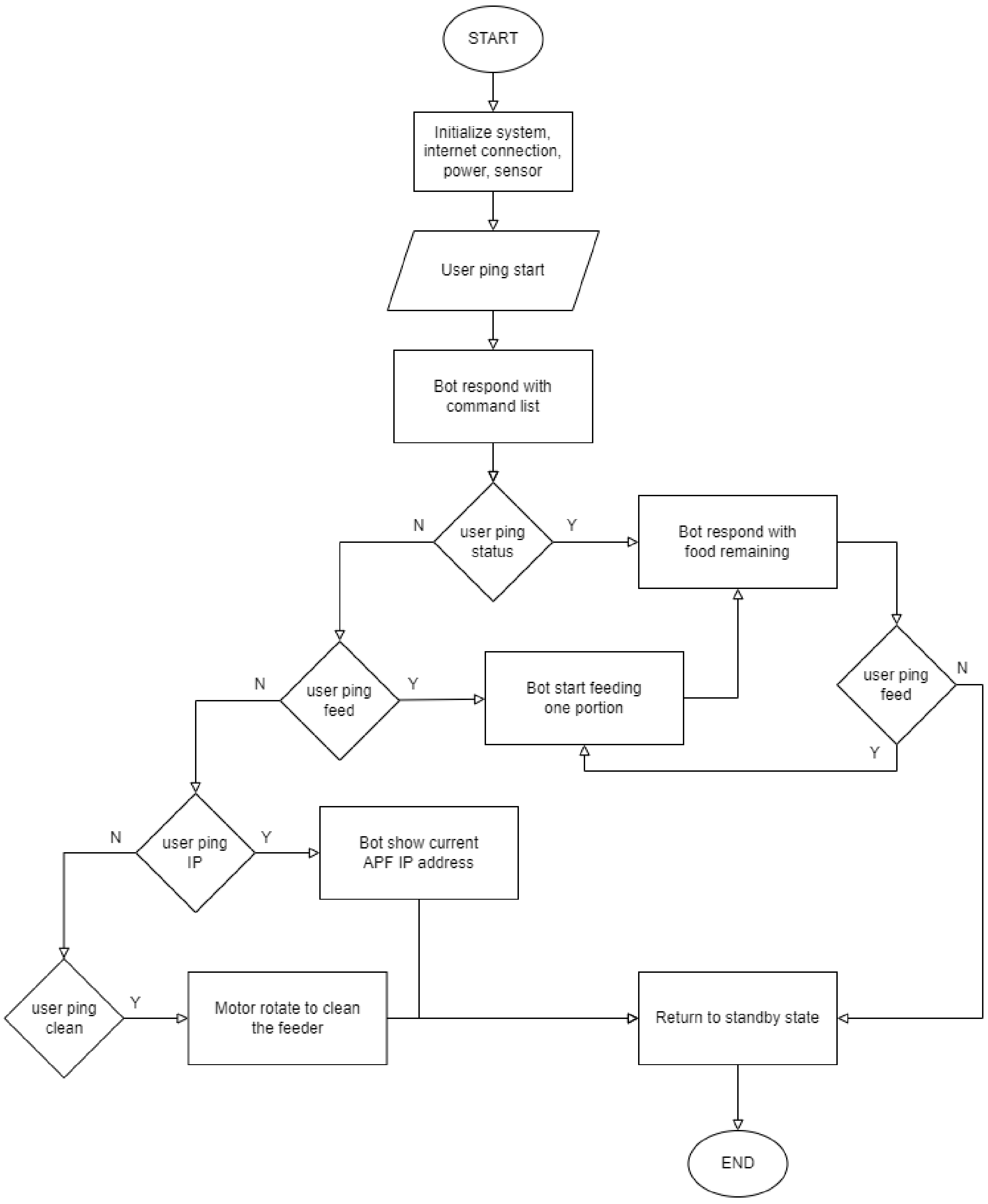
Table 3.1 Requirement type

Type of requirement	**Content**
Functional Requirement	- Authentication of user when the user using the system - Immediate save program in flash memory when no direct power is available - Does not open dispenser door if the sensor does not detect any food inside the container
Non-Functional Requirement	- Downtime of the system should be less than 0.1% of the total uptime of the system unless there is no power available - Status of the food level should be send no greater than 1 minutes to user after being instruct - The software used should be available on mobile phones (iOS and android), tablets and personal computers.
Constrains and limitations	- Quality of the product As this product will try to save cost by using recycled material and cheap resources, the quality may or may not reach the target buyers expectation. - Risk of the product As this is an experimental product without real-world daily testing, some of the product’ s part may or may not be safe such battery overheating or detachment of any part of the products

3.3 Propose Design

3.3.1 Flowchart

Figure 3.3 show the system process flowchart.



START
Initialize system, internet connection, power, sensor
User ping start
Bot respond with command list
user ping status
N
Y
Bot respond with food remaining
user ping feed
N
Y
Bot start feeding one portion
user ping feed
N
Y
user ping IP
N
Y
Bot show current APF IP address
user ping clean
Y
Motor rotate to clean the feeder
Return to standby state
END

Figure 3.3 Flowchart diagram of PPF

The flowchart consists of a few activities that need to be follow in order to make the functionality of the PPF system work at it best. First, the user starts with turn on the power on the switch at the power socket. Then, user need to wait for the system to initialize a few components before it ready to use. After that user can ping /start at the Telegram chat bot and chat bot will respond with a few commands line that user can use as shown in Figure 3.3. If the user chooses to use command /status, the bot will send the remaining food level inside the PPF container. If the user chooses to use command /feed, the PPF will drop one portion of food set inside the food bowl under the container. If user choose to command /ip, the PPF will give the current IP address that PPF currently using from the network. If user use command /clean, PPF will empty the container regardless of any food level remaining in it.

3.3.2 Context Diagram

Figure 3.4 show context diagram of PPF.

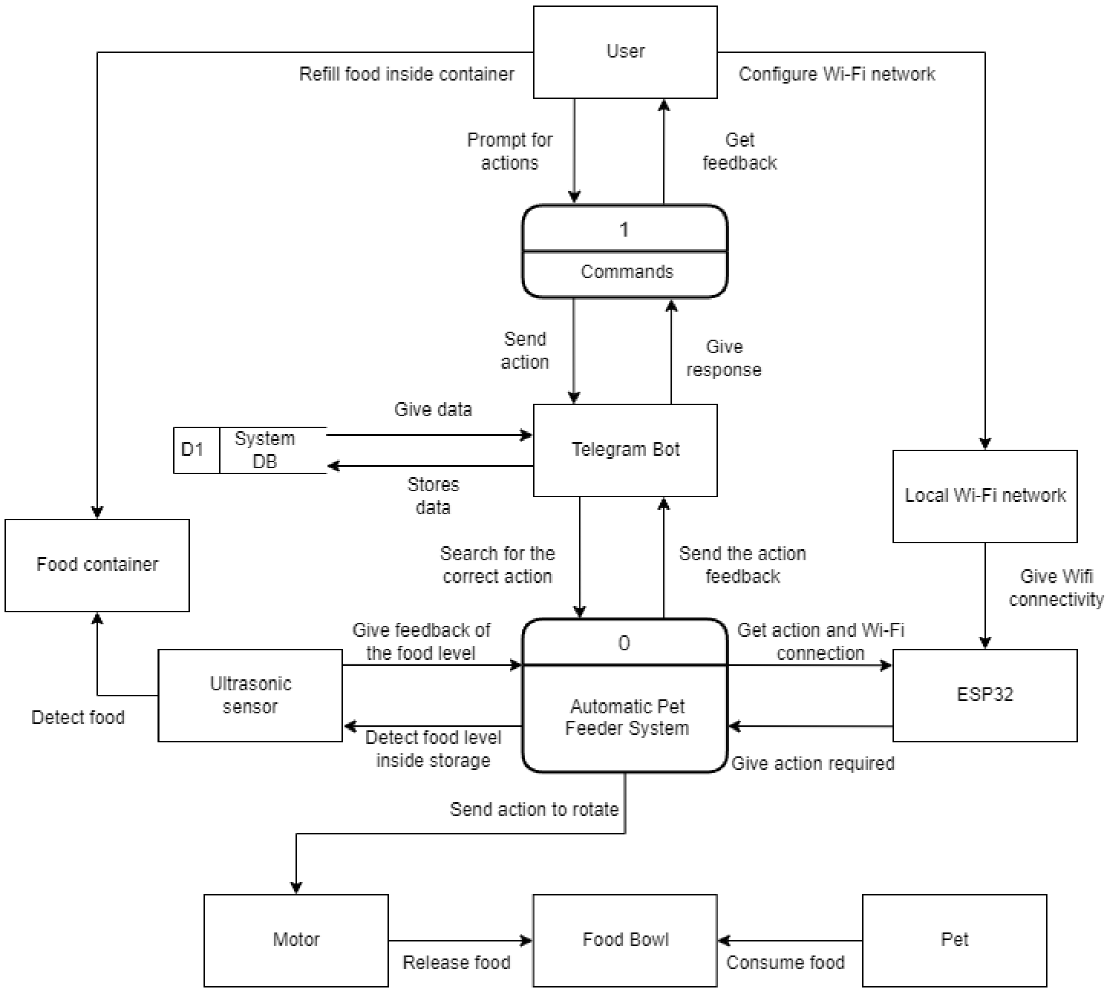


Figure 3.4 Context diagram of PPF

There are entity and process that interact to each other. As for this, the main component is the entity to entity itself. For the Telegram Bot entity, we can see that User can prompt action to it and user will receive action from the Telegram bot. For the Ultrasonic sensor, it will detect food level in the container and send the feedback to the main entity that is the Arduino as the system itself. For ESP32 module that work as wireless connector to the Wi-Fi and will receive the connection from Local Wi-Fi network to the module.

3.3.3 Use Case Diagram

Figure 3.5 shows the use case diagram for PPF system

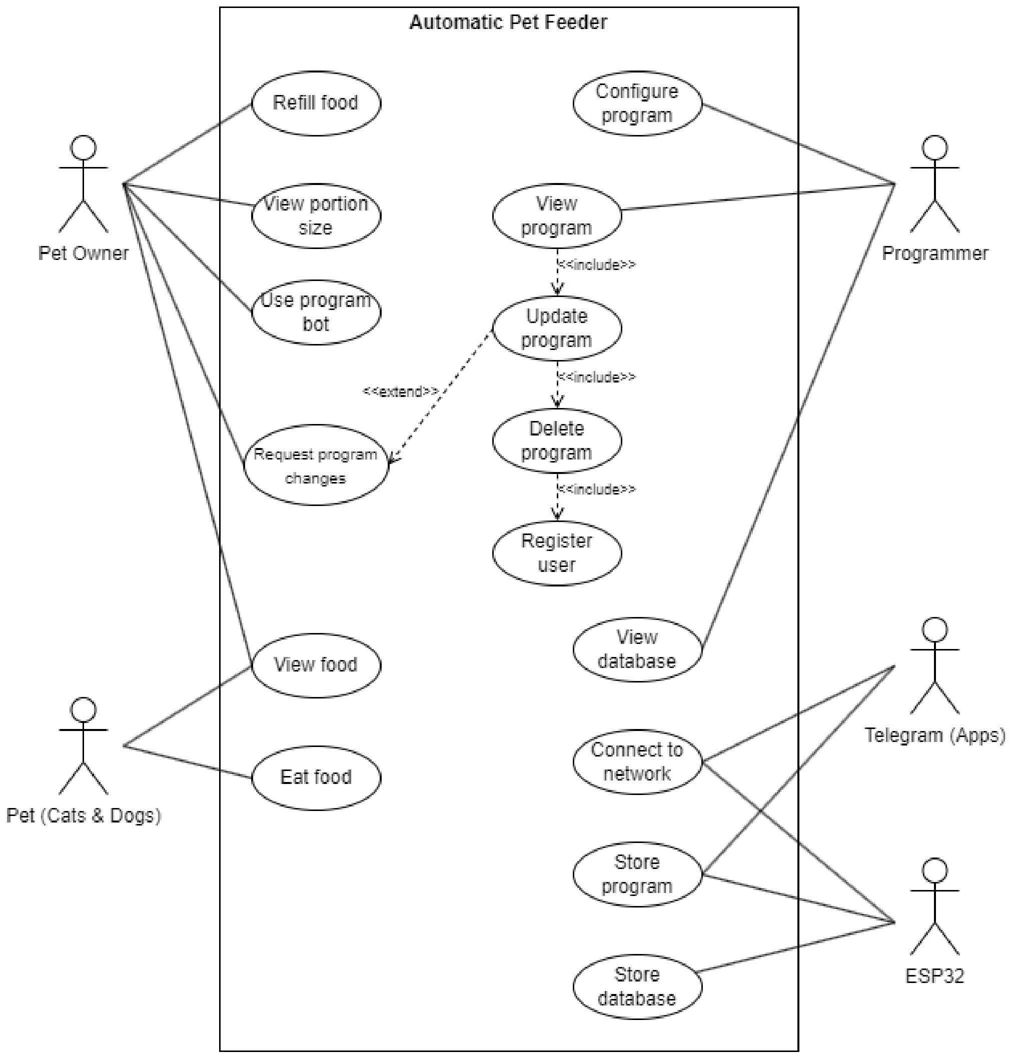
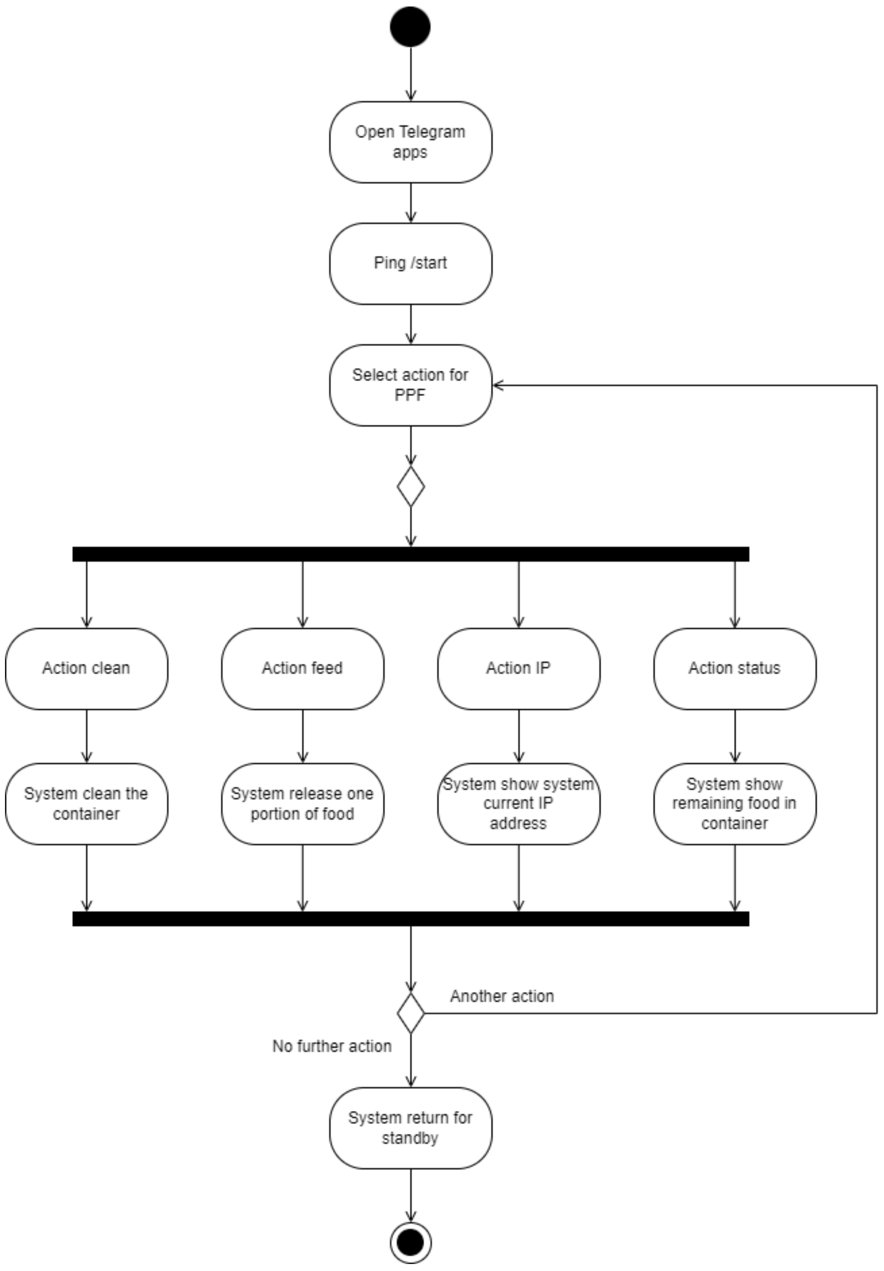


Figure 3.5 Use case diagram for PPF

For this use case diagram, we can see the interaction between actor and the system. There are five actors for this use case that is the owner, pet, programmer, Telegram and Arduino UNO module. As for Pet Owner, they can refill food, view portion, use the bot and request program changes from the programmer. Also, for the programmer, they can configure, view, update, delete and register user for the programme including to view the stored database about the owner info and IP address history.

3.3.4 Activity diagram

Figure 3.6 shows the activity diagram for PPF system



Open Telegram apps
Ping /start
Select action for PPF
Action clean
Action feed
Action IP
Action status
System clean the container
System release one portion of food
System show system current IP address
System show remaining food in container
Another action
No further action
System return for standby

Figure 3.6 Activity diagram for PPF

The activity diagram includes a few steps that must be taken in order for the PPF system to run as efficiently as possible. The user begins by turning on the power at the power socket's switch. The user must then wait until a few system components are initialised before the system is ready for usage. The Telegram chatbot will then answer to the user's ping with a few command lines, as illustrated in Figure 3.3, and the user can use those. The bot will send the amount of food still within the PPF container if the user chooses to use the command /status. The PPF will drop one serving of food placed in the food bowl underneath the container if the user chooses to use the command /feed. The PPF will provide the current IP address that PPF is currently using from the network if the user chooses to execute /ip. Regardless of the amount of food still inside, if the user issues the command /clean, PPF will empty the container. User can choose to user the same command again or just ignore the bot and it will return to initial state.

3.4 Data Design

3.4.1 Entity Relation Diagram (ERD)

Figure 3.8 shows the ERD of the PPF system that are consist of five main elements of the system.

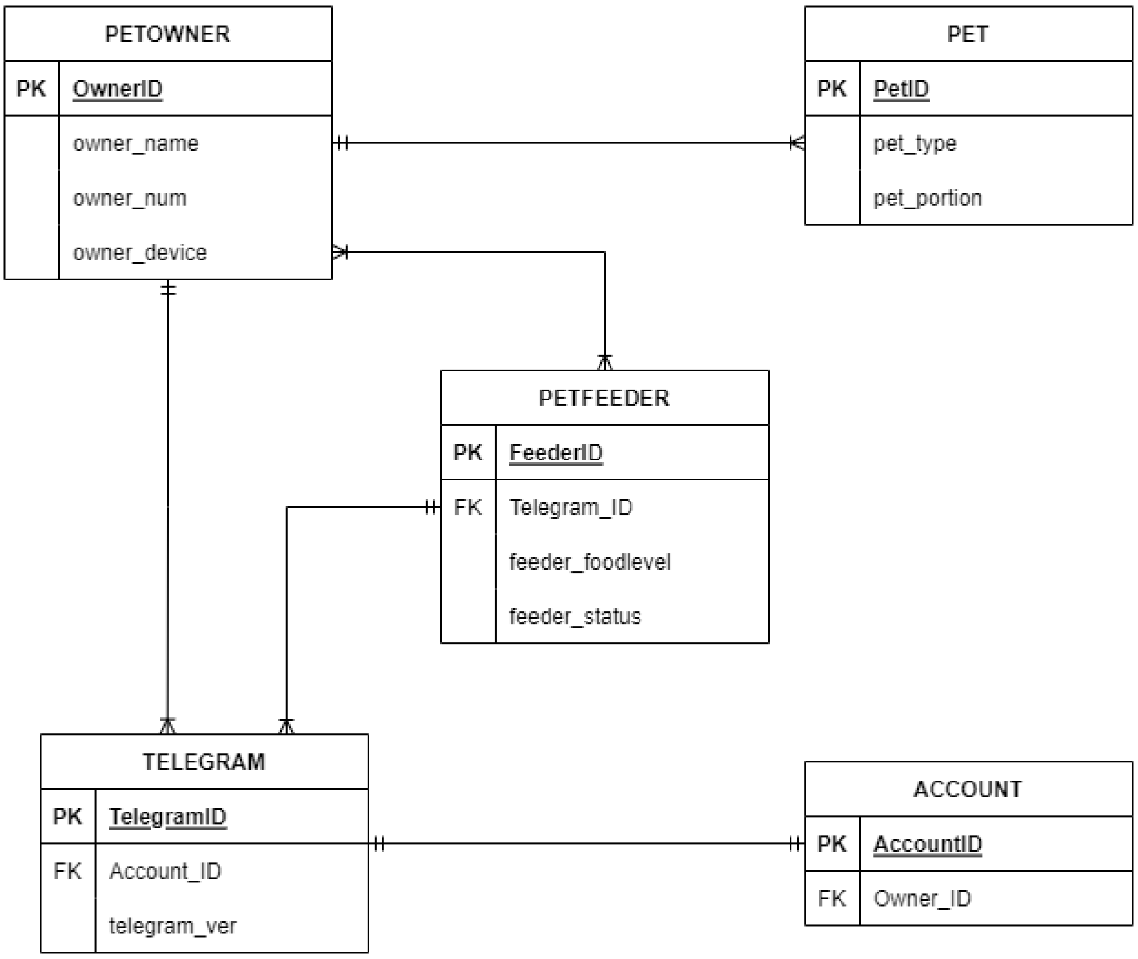


Figure 3.7 ERD of PPF

As for ERD, there are few things that need to be understand throughout the system. There are few entities and their relation to one another. For example, the PETOWNER can have one or more PET, but pet can only have one and only one owner at a time. Also, the TELEGRAM on every user device can have only one ACCOUNT and only one account are allowed per user number. Next, PETOWNER can have one or more PETFEEDER as they like and needed and the PETFEEDER itself can be owned by one or more PETOWNER at a time as this is to ensure whole family member can access the PETFEEDER as a register user and feed their pet accordingly

3.4.2 Data Dictionary

Data dictionary consist of data elements utilised or recorded in a database, information system, or as a component of a research study are given names, definitions, and properties. In addition to offering instructions on interpretation, acceptable interpretations, and representation, it specifies the meanings and goals of data pieces within the framework of a project. Information about data elements is also provided by a data dictionary. Table 3.2 until 3.6 below show the data dictionary related to the ERD from figure 3.8.

Table 3.2 Pet owner data dictionary

Attributes	**Description**	**Type**	**Length**	**PK/FK**
OwnerID	User ID	int	10	PK
Owner_name	User name	varchar	255	-
Owner_num	User phone number	int	12	-
Owner_device	User device type (android, iOS, windows, Mac)	varchar	100	-

Table 3.3 Pet data dictionary

Attribrutes	**Description**	**Type**	**Length**	**PK/FK**
PetID	Pet ID	int	10	PK
Pet_type	Type of pet	varchar	20	-
Pet_portion	Amount of food per feed	int	100	-

Table 3.4 Pet feeder data dictionary

Attribrutes	Description	Type	Length	PK/FK
FeederID	Feeder ID	int	10	PK
Telegram_ID	Telegram apps	varchar	100	FK
Feeder_foodlevel	Food level remaining in the feeder	varchar	6	-
Feeder_status	Feeder availability status	varchar	100	-

Table 3.5 Telegram data dictionary

Attribrutes	Description	Type	Length	PK/FK
TelegramID	Telegram ID	int	10	PK
Account_ID	Account of the Telegram apps	varchar	100	FK
Telegram_ver	Version of the telegram	varchar	20	-

Table 3.6 Account data dictionary

Attribrutes	**Description**	**Type**	**Length**	**PK/FK**
AccountID	Account ID	int	10	PK
Owner_ID	Owner ID required for account	varchar	100	FK

All the data dictionaries are acquired from the Entity Relation Diagram (ERD) from Figure 3.8.

3.5 Proof of Initial Concept

A proof of concept shows how a specific business idea function. A proof of concept can be used to demonstrate the viability of a product, service, business plan, or work procedure. A proof of concept is a hypothetical demonstration of an idea's possible production and use as opposed to a prototype, which is a functioning model of the proposed product. Figure 3.7 show the early proposed design of the project.

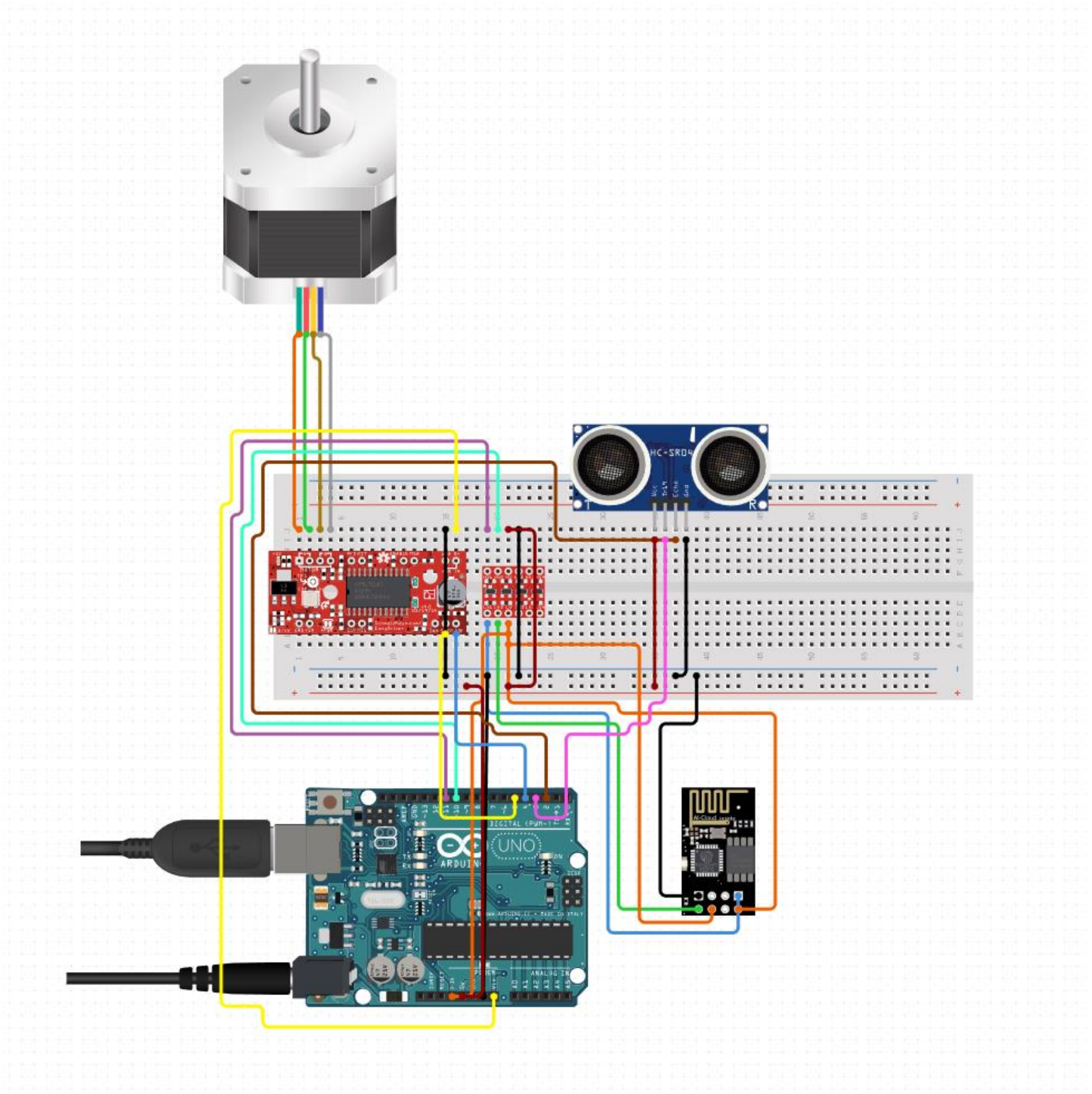


Figure 3.8 Initial design of the PPF system

The initial design consists of many parts to ensure that the system is well developed to meet its requirement. In this section, type of item, software and component use will be explained. Figure 3.8 until Figure 3.17 show parts that are used in this project.

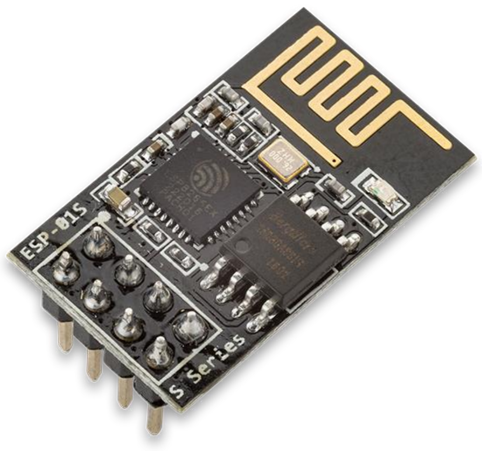
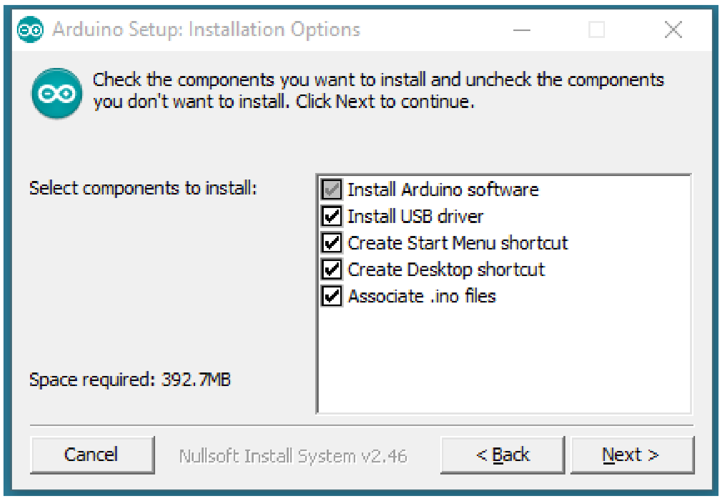


Figure 3.9 ESP 32 module


Arduino Setup: Installation Options

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- [✓] Install Arduino software
- [✓] Install USB driver
- [✓] Create Start Menu shortcut
- [✓] Create Desktop shortcut
- [✓] Associate .ino files

Space required: 392.7MB

Cancel | Nullsoft Install System v2.46 | < Back | Next >

Figure 3.10 Installation of Arduino IDE

A microcontroller board called ESP32 is based on the ATmega328P. It contains 6 analogue inputs, a 16 MHz ceramic resonator, 14 digital input/output pins (six of which can be used as PWM outputs), a USB port, a power jack, an ICSP header, and a reset button. It comes with everything needed to support the microcontroller; to get started, just plug in a USB cable, an AC-to-DC adapter, or a battery.

All my component will be attached and integrated with this microcontroller. As for the language itself, C or C++ will be used to program the system. The Arduino board's USB connection is used to link the board to a computer through a USB cable. The board's serial port and power supply are both provided by the cable. Arduino are programmable using its software Arduino IDE available for free on Arduino official website.

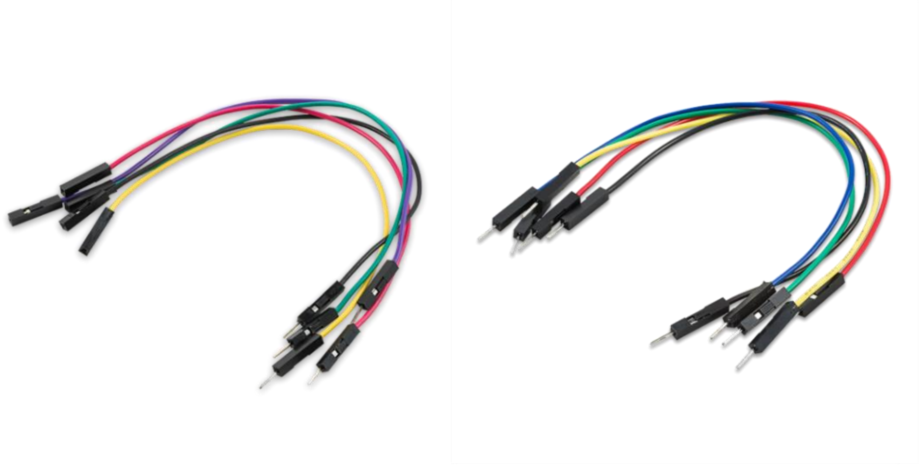


Figure 3.11 Jumper wire connecter (male to male – male to female)

A jumper wire is an electric cable used to link distant printed circuit board electric circuits. It is possible to short-circuit and jump to the electrical circuit by connecting a jumper wire to the circuit.

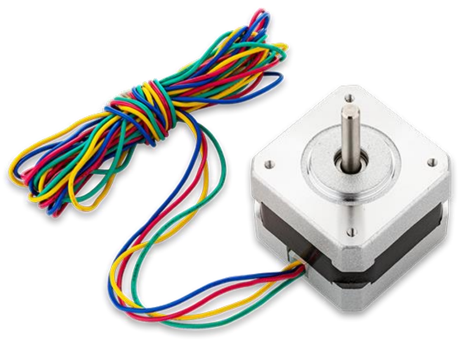


Figure 3.12 NEMA stepper motor

This gear-like component has teeth that are arranged around a central shaft, and the stepper motor uses a pulsed electrical current that is controlled by a stepper motor driver to precisely move it in one step. The rotor is rotated through one exact and fixed increment of a full turn with each of these stepper motor pulses. The rotary component can make full or partial turns as needed as the current is switched between the wire coils arranged in series around the outside of the motor, or it can be made to stop abruptly at any point along its revolution. This device is used to rotate the food feeder to ensure that food dropped to the bowl in the correct amount. The angle can be set in the Arduino software and adjustable from time to time to adjust the portion according to the pet owner.

Figure 3.13 Power adapter 12V – 2A

An adapter made to provide precisely 12 Volts of direct current to a device is known as a 12V DC power supply. The provided voltage must exactly match the specifications of the machinery. This device will power up the Arduino and supple the power to all the component such motor and sensor on the PPF system.

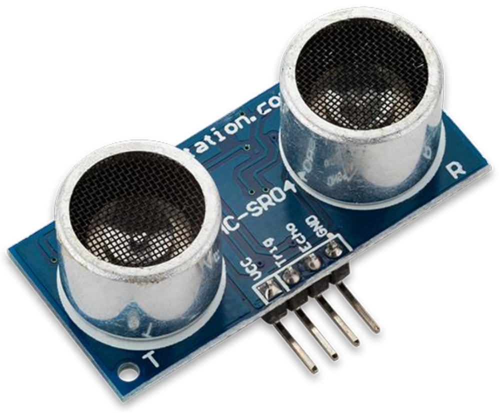


Figure 3.14 SR04 – Ultrasonic sensor

The HC-SR04 ultrasonic sensor uses SONAR to determine the distance of an object just like the bats do. It provides exceptional non-contact range detection from 2 cm to 400 cm or 1 inch to 13 feet in an easy-to-use compact with high accuracy and consistent readings. Although soft materials like cloth can be challenging to detect acoustically, the operation is unaffected by sunshine or dark materials. It includes an ultrasonic transmitter and receiver module in its entirety. This module is used to determine the remaining food level inside the food container precisely and let the user know how much quantity available for their pets.

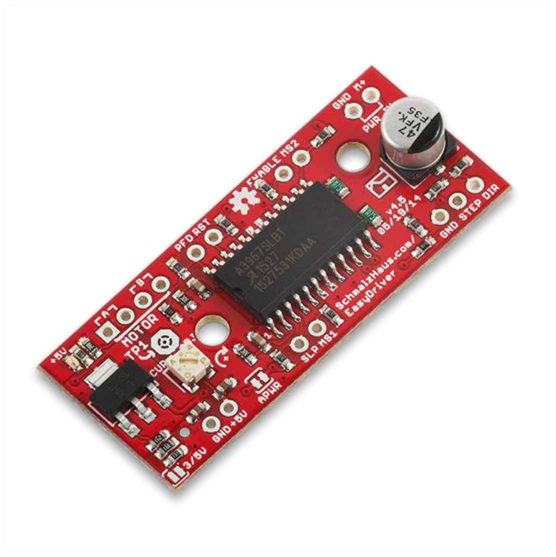
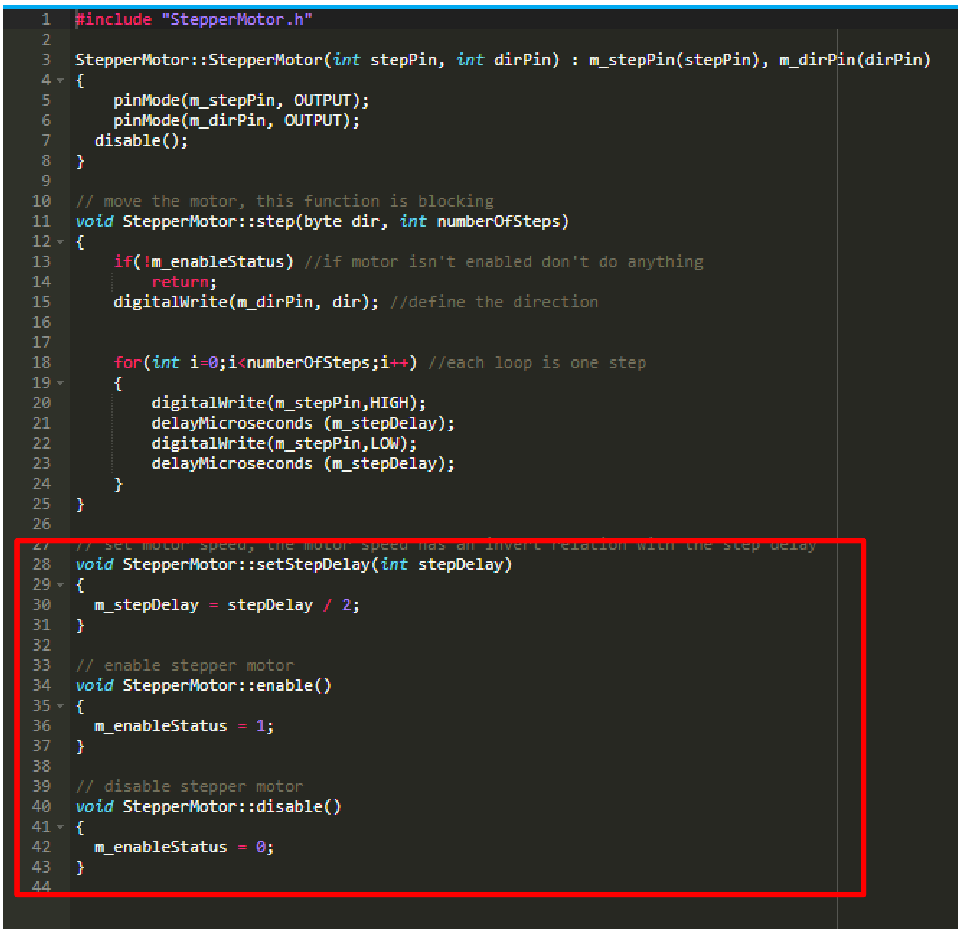


Figure 3.15 Stepper motor driver

Stepper motors make it simple to place objects accurately. They are utilised in a variety of equipment types for precise pulse signal-based rotation angle and speed control. Stepper motors are perfect for quick acceleration and responsiveness since they produce strong torque with a small size [7]. Due to their mechanical construction, stepper motors also maintain their position when stopped. A driver which receives pulse signals and translates them to motor motion. This driver is use specifically to control the angle, speed and direction of NEMA motor in the PPF system.



```
1   #include "StepperMotor.h"
2
3   StepperMotor::StepperMotor(int stepPin, int dirPin) : m_stepPin(stepPin), m_dirPin(dirPin)
4   {
5       pinMode(m_stepPin, OUTPUT);
6       pinMode(m_dirPin, OUTPUT);
7     disable();
8   }
9
10  // move the motor, this function is blocking
11  void StepperMotor::step(byte dir, int numberOfSteps)
12  {
13      if(!m_enableStatus) //if motor isn't enabled don't do anything
14          return;
15      digitalWrite(m_dirPin, dir); //define the direction
16
17
18      for(int i=0;i<numberOfSteps;i++) //each loop is one step
19      {
20          digitalWrite(m_stepPin,HIGH);
21          delayMicroseconds (m_stepDelay);
22          digitalWrite(m_stepPin,LOW);
23          delayMicroseconds (m_stepDelay);
24      }
25  }
26
27  // set motor speed, the motor speed has an invert relation with the step delay
28  void StepperMotor::setStepDelay(int stepDelay)
29  {
30    m_stepDelay = stepDelay / 2;
31  }
32
33  // enable stepper motor
34  void StepperMotor::enable()
35  {
36    m_enableStatus = 1;
37  }
38
39  // disable stepper motor
40  void StepperMotor::disable()
41  {
42    m_enableStatus = 0;
43  }
44
```

Figure 3.16 Stepper motor code

As this is an early project development there are few codes that has been developed to ensure the system work as intended. In figure 3.19, the code for stepper motor is created to ensure the motor rotate in the correct angle and timing. This code is not yet been tested as the NEMA stepper motor itself has not integrated with the Arduino board yet. The code itself is in C/C++ language where the code is written on online code builder. There should be more code integrated to the motor but for now it is just a basic step to achieve better function for it to be use. As we can see between line 28 until line 44 it is the code for the delay of the motor to rotate in its count cycle [10].

3.6 Testing/Validation Plan

To ensure that the program follows the process it is supposed to perform, a test case table is extracted to check for any errors that may occur during this application use. Table 3.7 shows the user input command to bot and table 3.8 shows the action command of the PPF.

Table 3.7 User input command to Telegram test case

No	1
Test case	User input command to Telegram bot
Description	Test the action command to Telegram bot such status, feed and clean. Prerequisite: 1. User cannot use any of the action prompt. 2. User does not get any respond from the bot.
Test steps	1. The user need to make sure that the user device and PPF is connected to Wi-Fi. 2. Restart Telegram apps (Force stop) and open it again. 3. If only step 2 does not work – Restart user device and PPF system.
Expected result	1. User can get action feedback from PPF. 2. User can use Telegram bot as usual.
Result: (Pass √ Fail x)	Pass

Table 3.8 PPF action to user test case

No	2
Test case	PPF respond and action
Description	Test PPF system to ensure it release the correct set of portions at real-time. Prerequisite: 1. User must make sure user' s device and PPF are connected to the Wi-Fi. 2. User must make sure that there is enough food reserve in the storage container.
Test steps	1. User prompt /feed action at Telegram bot.
Expected result	1. PPF should release correct food portion in real-time.
Result: **(Pass √ Fail x)**	Pass

3.7 Potential Use of Proposed Solution

Pro Paws Feeder (PPF) are expected to serve a better environment for pets when the owner is not around them. This PPF should provide the essentials necessities to pet especially cats and dogs. A lot of benefits that we can get if this PPF works. One of it is no more guilt to owner. All of us lead busy lives, and our schedules occasionally prohibit us from maintaining a regular feeding plan for our animals. It's not always possible to ensure that our pets will eat at the same time every day. The assurance that our pet will be fed while we are gone or at work comes from feeders. When we are confident that our pet will eat the same meals every day, we can easily do our daily activity without any distraction.

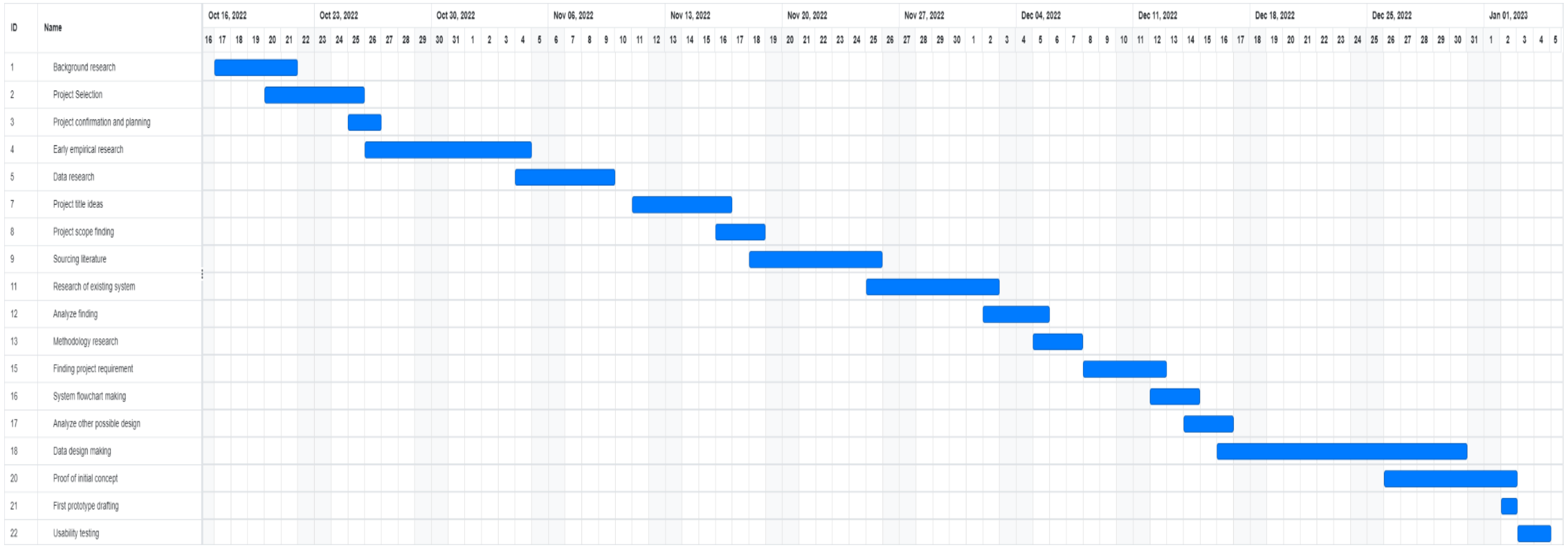
Also, PPF also can make the food sustain a better quality. This automatic feeder' s main benefit is that they always give the pets fresh food. Every time the dog and cat eat, it will receive fresh food because the owner has filled the device with it. Only the owner will be able to set the time. The bowls don't need to be refilled each time our pets need food. These are also advantageous; we may purchase exterior door bowls if we take our pet outside.

Lastly, we can lessen wastage of food. The most important thing to remember is that we should give our dogs enough food to enjoy while also promoting their growth and development. Most veterinarians advise owners to feed their dogs and cats a specific amount of food, and these feeders make it easier to feed pets so that they develop in a particular way. We can provide our pets with the correct amount of food using these automatic feeders, allowing them to grow properly and maintain excellent physical health.

3.8 Gantt Chart

One of the most common and effective methods of displaying activities tasks or events displayed against time is a Gantt chart, which is frequently used in project management. A list of the activities is located on the chart's left side, and a suitable time scale is located along the top. A bar is used to symbolise each activity, and the position and length of the bar correspond to the activity's beginning, middle, and finish dates. Figure 3.17 below show the PPF Gantt chart.

Figure 3.17 Gantt chart of PPF



CHAPTER 4

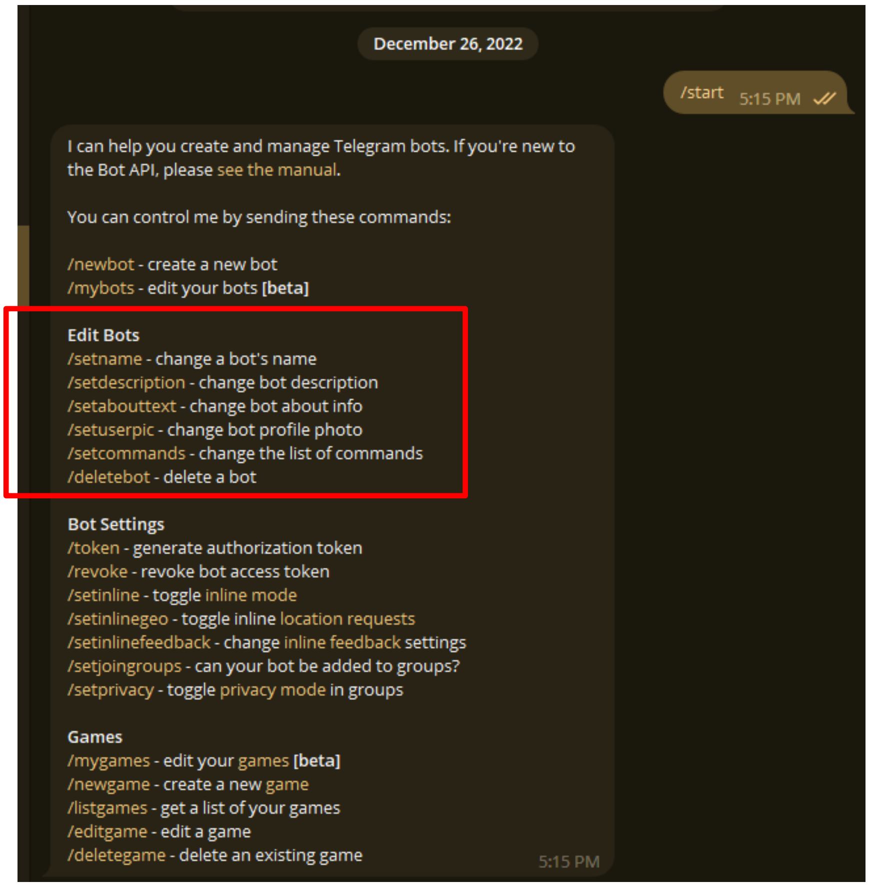
RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, the entire process of implementing the project development will be explained in detail. All hardware or software required are used to begin developing the system. In addition to this, this phase requires to develop connectivity, functionality, and the results. This chapter also includes the prototyping and testing of the Pro Paws Feeder (PPF) from the code, components, and integration with Telegram.

4.2 Implementation Process

This section discusses the implementation process that involve in completing this project, starting from the beginning of the development process including codes and interfaces. Figure 4.1 shows the first step to complete PPF system, which is obtaining Bot Token from Bot Father.



December 26, 2022

/start 5:15 PM

I can help you create and manage Telegram bots. If you're new to the Bot API, please see the manual.

You can control me by sending these commands:

/newbot - create a new bot
/mybots - edit your bots **[beta]**

Edit Bots
/setname - change a bot's name
/setdescription - change bot description
/setabouttext - change bot about info
/setuserpic - change bot profile photo
/setcommands - change the list of commands
/deletebot - delete a bot

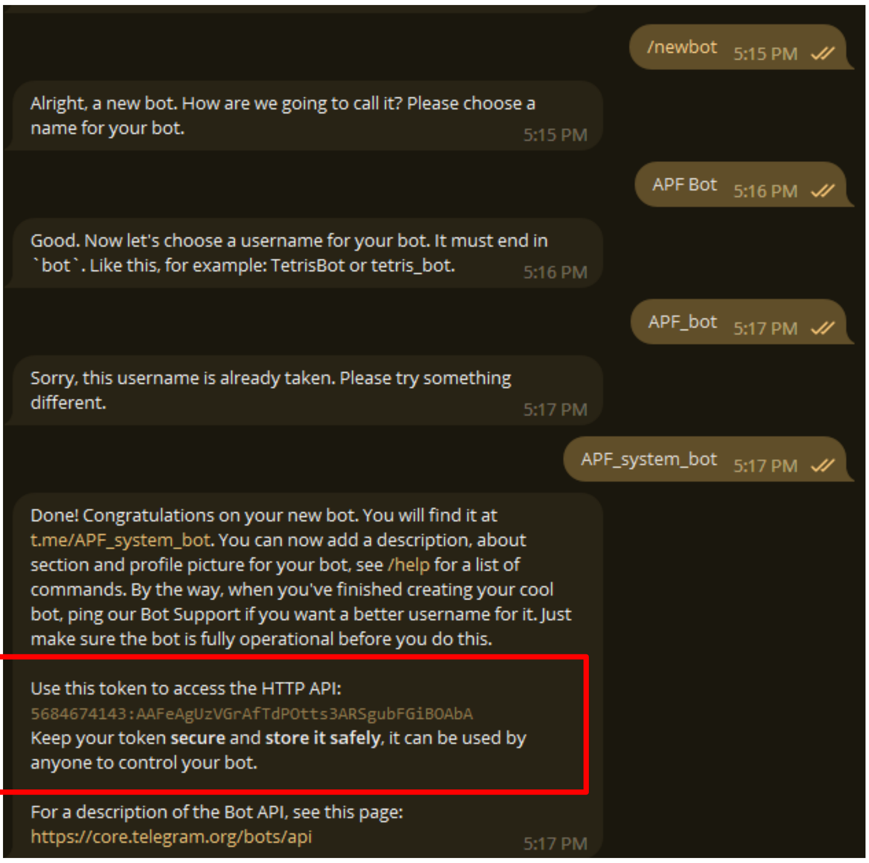
Bot Settings
/token - generate authorization token
/revoke - revoke bot access token
/setinline - toggle inline mode
/setinlinegeo - toggle inline location requests
/setinlinefeedback - change inline feedback settings
/setjoingroups - can your bot be added to groups?
/setprivacy - toggle privacy mode in groups

Games
/mygames - edit your games **[beta]**
/newgame - create a new game
/listgames - get a list of your games
/editgame - edit a game
/deletegame - delete an existing game

5:15 PM

Figure 4.1 BotFather options from telegram

This first step in creating a fully functioning system is by obtaining the most important agent in this system that is the Bot itself. This can be obtained by using Bot Father, created by Telegram to allow user or developers to create and experimenting their own bot. The Bot Father can easily find when we search in Telegram search bar Bot Father. Add the bot to the account and we can start by sending /start command to the bot. The Bot Father will respond a few sets of options that we can use. As for now we just want to create a new bot by obtaining its token from Bot Father.


/newbot 5:15 PM

Alright, a new bot. How are we going to call it? Please choose a name for your bot. 5:15 PM

APF Bot 5:16 PM

Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot. 5:16 PM

APF_bot 5:17 PM

Sorry, this username is already taken. Please try something different. 5:17 PM

APF_system_bot 5:17 PM

Done! Congratulations on your new bot. You will find it at t.me/APF_system_bot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:
`5684674143:AAFeAgUzVGrAfTdPOtts3ARSgubFGiBOAbA`
Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

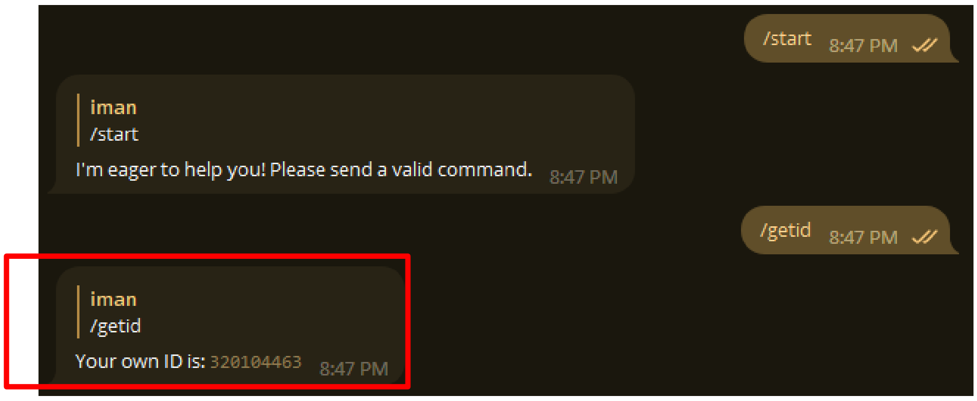
For a description of the Bot API, see this page:
https://core.telegram.org/bots/api 5:17 PM

Figure 4.2 Getting bot Token from BotFather

Figure 4.2 show when we send /newbot is sent from the options provided by Bot Father, a blank or unprogrammed bot can be created by obtaining its token, which can be used later in in this project.

Figure 4.3 ID bot on Telegram

Figure 4.3 shows that we are going to obtain User ID from IDBot create by Telegram themselves to help user find their user ID from Telegram when needed. Telegram User ID is unique as one ID is assigned to one user only in the world. This ID is used to create authorization in the PPF where only the User ID that are added or assign to PPF can only use PPF.


/start 8:47 PM

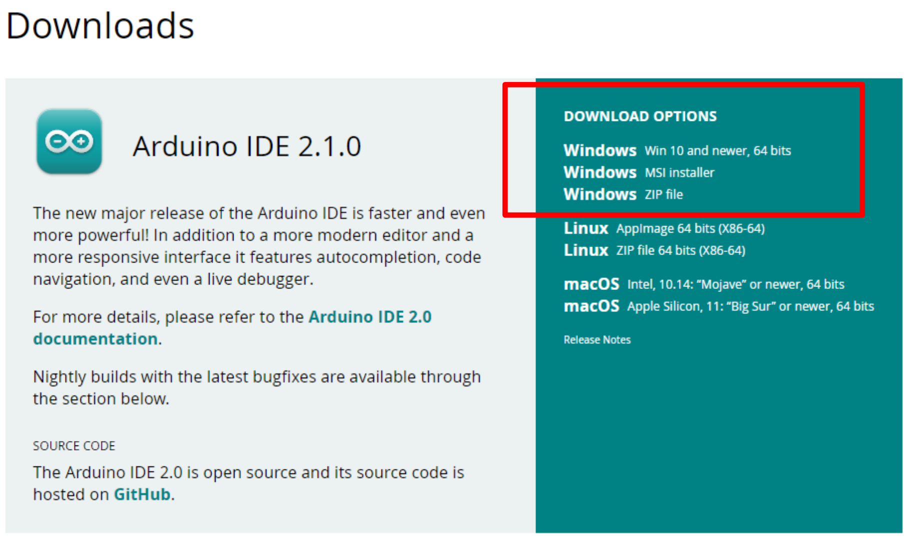
iman
/start
I'm eager to help you! Please send a valid command. 8:47 PM

/getid 8:47 PM

iman
/getid
Your own ID is: 320104463 8:47 PM

Figure 4.4 Obtaining user ID

As shown in Figure 4.4, we start by sending /start command to IDBot and the bot will respond with sender ID that we need to save and use for later on in this project.


Downloads

Arduino IDE 2.1.0

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the **Arduino IDE 2.0 documentation**.

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on **GitHub**.

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file
Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)
macOS Intel, 10.14: "Mojave" or newer, 64 bits
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits
Release Notes

Figure 4.5 Download and install arduino IDE

Figure 4.5 shows Arduino page that we need visit to download Arduino IDE. This is where we programme the PPF and integrate it into the ESP32.

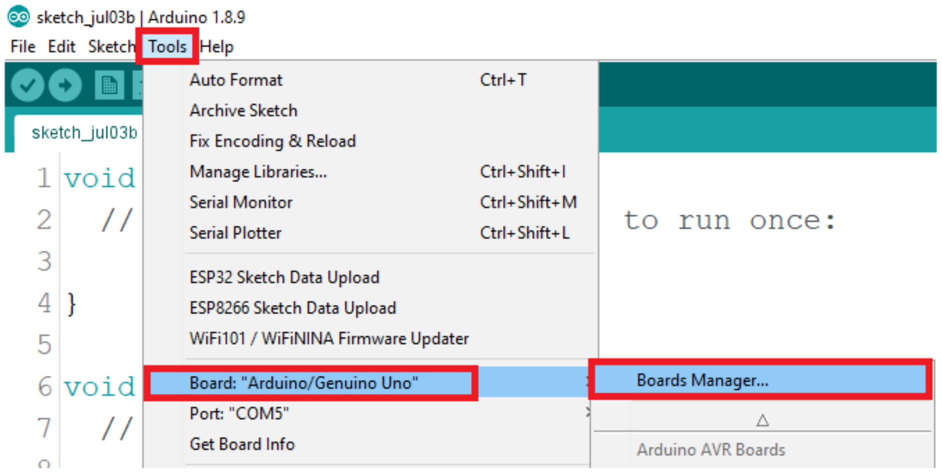
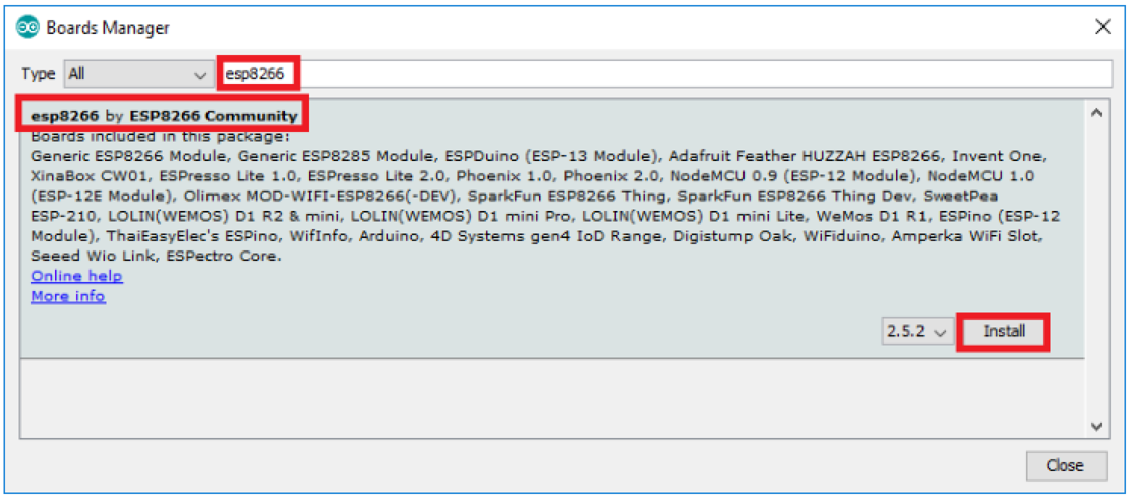

sketch_jul03b | Arduino 1.8.9
File Edit Sketch Tools Help
sketch_jul03b
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
ESP32 Sketch Data Upload
ESP8266 Sketch Data Upload
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM5"
Get Board Info
Boards Manager...
Arduino AVR Boards
void
//
}
void
//
to run once:

Figure 4.6 Steps to download ESP32 board

Figure 4.6 shows steps to download ESP32. After Installing the Arduino IDE, we need to import board library. The Arduino environment can be extended through the use of libraries, just like most programming platforms. Libraries provide extra functionality for use in sketches such working with hardware or manipulating data.


Boards Manager

Type All esp8266

esp8266 by **ESP8266 Community**
Boards included in this package:
Generic ESP8266 Module, Generic ESP8285 Module, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, WeMos D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core.
Online help
More info

2.5.2 Install

Close

Figure 4.7 ESP32 board install

Figure 4.7 shows ESP32 board extension install. After searching board manager tab, search on the search bar ‘ESP32’ and install the first result on the search suggestion.

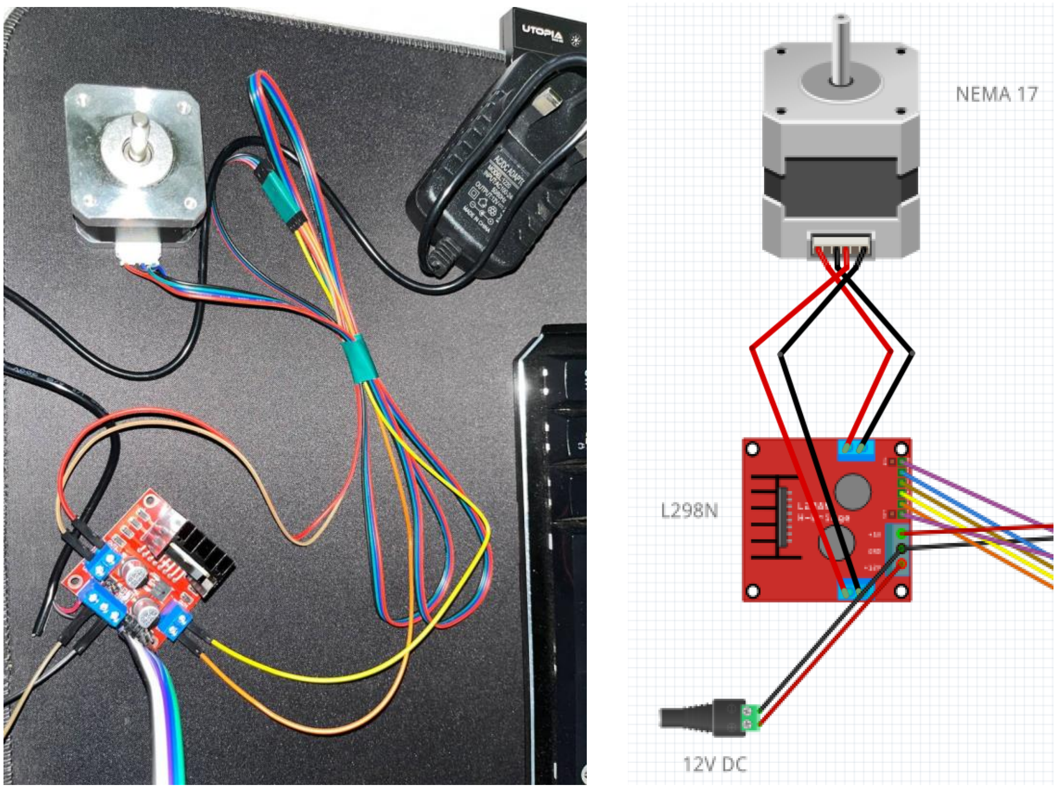


Figure 4.8 Connection between motor, driver and AC

Figure 4.8 shows the connection on the physical view of the components and the schematic view of the connection. As we can see the connection between the motor and the L298N motor driver are connected via four cables. The two red cable is the live wire supplying the 12V current each to the motor and sending the instruction to the motor. Two black wires are both for ground connection of the motor and the driver. The L298N driver also connected directly to the 12V DC socket through two cables. One for live to send the voltage and one for ground.

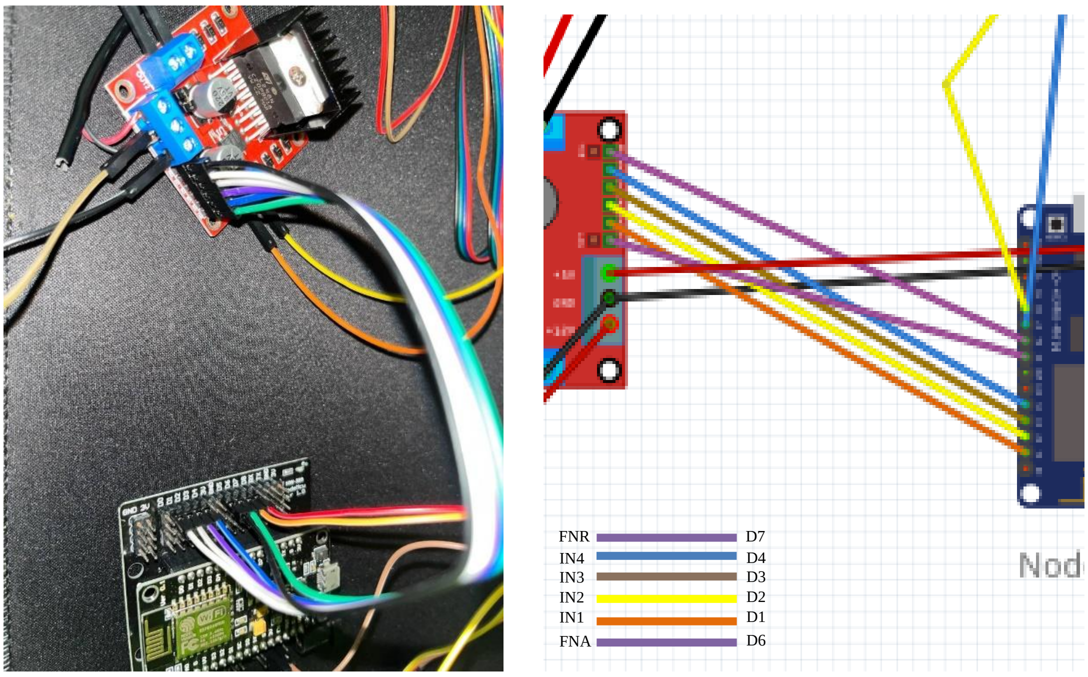


Figure 4.9 Connection between driver to ESP32

Figure 4.9 shows the connection between the L298N motor driver and the ESP32 module. The motor driver is connected via 8 cables in total to the ESP32 module. As we can view on the schematic, the port' s function are as follows:

1. FNA (Bottom purple cable) - This is the enable input for Motor A. It is used to control the speed of the motor. Connect this to a PWM (Pulse Width Modulation) enabled GPIO (General Purpose Input/Output) pin on the NodeMCU module.

2. IN1, IN2, IN3, IN4 (Orange, yellow, brown, blue cable) - This is an input pin for Motor A. It controls the direction of the motor. Connect this to a GPIO pin on the NodeMCU module.

3. FNR (Top purple cable)**:** These pins are used to control the direction of the motors. By sending a logic HIGH signal to one of these pins and a logic LOW signal to the other, the motor will turn in one direction. By reversing the logic levels on these pins, the motor will turn in the opposite direction.

4. GND (Black cable) - This is the ground pin for the L298N module. Connect this to a ground pin on the NodeMCU module.

5. 5V+ (Red wire) - This is the live wire where power from the motor drive being converted and transfer into 5V current to power up the ESP32 module.

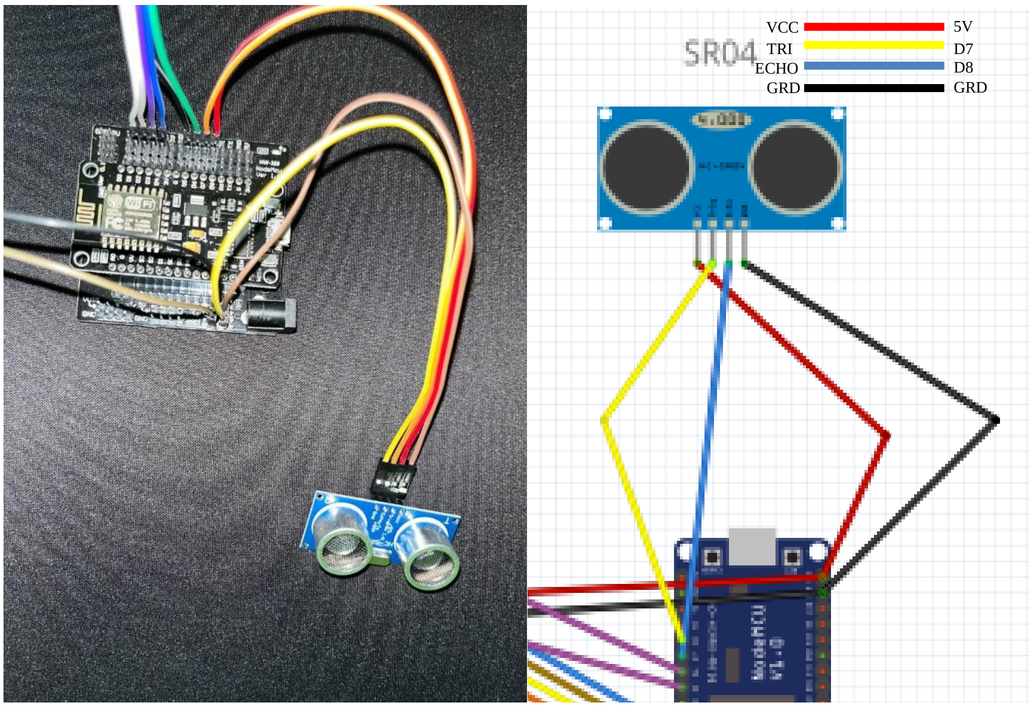


Figure 4.10 Connection between sensor and ESP32

Figure 4.10 shows the connection between the sensor and ESP32. The sensor is connected to the ESP32 module by 4 cables. As we can view on the schematic, the port' s function are as follows:

1. VCC: This pin is connected to the power supply (5V or 3.3V) of the system.

2. GND: This pin is connected to the ground of the system.

3. TRIG: This pin is used to trigger the ultrasonic signal. When the TRIG pin is set to HIGH for at least 10 microseconds, the ultrasonic signal is transmitted.

4. ECHO: This pin is used to receive the ultrasonic signal. The ECHO pin outputs a pulse whose duration is proportional to the time taken for the ultrasonic signal to travel to the object and back. The duration of the pulse can be measured using a microcontroller and can be used to calculate the distance of the object from the sensor.

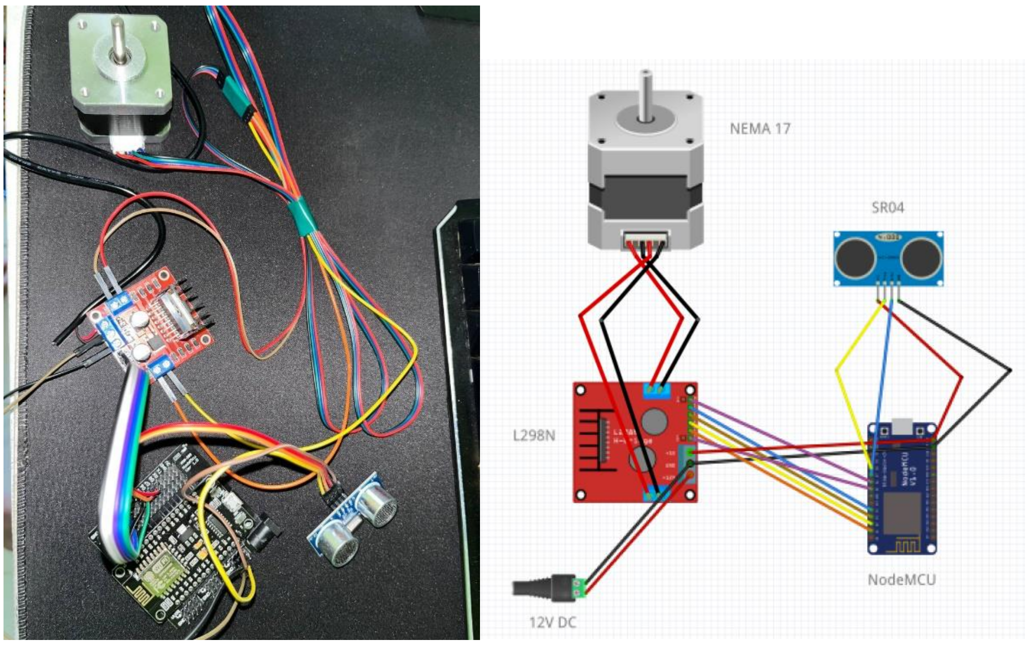


Figure 4.11 Overview connection and schematic

Figure 4.11 is the overview of the overall module of the PPF system that are connected to one another to create a complete and functioning pet feeding system.

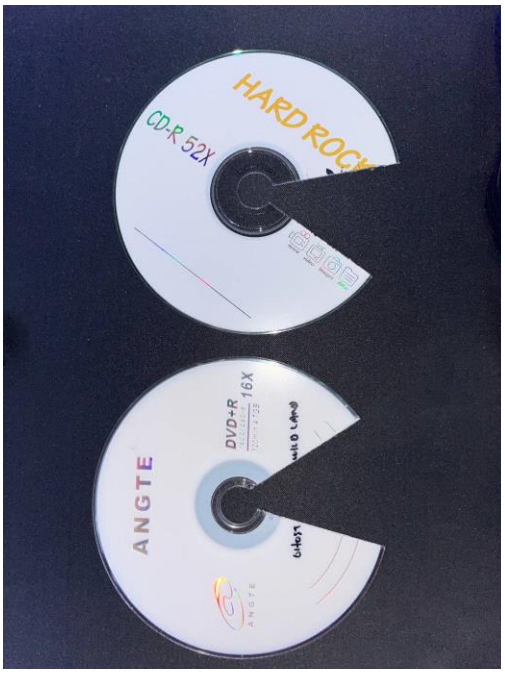


Figure 4.12 Old disk as dispenser door

Figure 4.12 shows old disk that are recycle as the dispenser rotator or door which is used to dispense the food from the container to the bowl. Two disks will be use as the dispenser opening. One will be fixed into its place at the opening or the bottom of the container, the other will be attach to the motor.

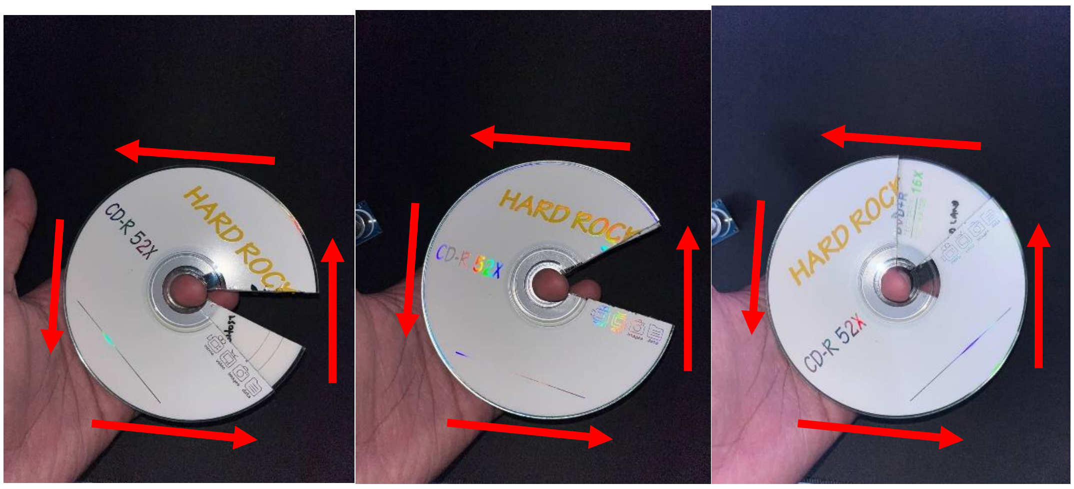
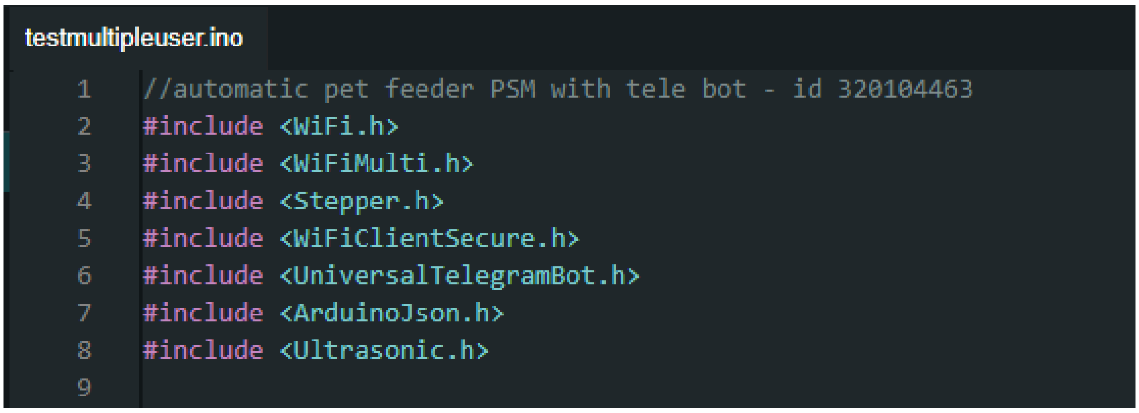


Figure 4.13 Disk movement on motor

Figure 4.13 shows the disk movement of the motor start from the left to right. The disk will rotate one piece only and the other one is fixed on the dispenser opening. The disk will rotate at constant speed in one direction until the set is served. The speed of the disk moving can be pre-set depends on the opening size of the gap. This is because if the disk opening is wide (as in figure), the rotation acquired need to be fast to ensure the food not dispensing too much from the container into the bowl. As there are many recycle disk out there available, we can fix the speed of the rotation and only adjust the opening size of the disk to ensure only one variable is used.



testmultipleuser.ino

```
//automatic pet feeder PSM with tele bot - id 320104463
#include <WiFi.h>
#include <WiFiMulti.h>
#include <Stepper.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>
#include <Ultrasonic.h>

```

Figure 4.14 Libraries required

Figure 4.14 shows the beginning of the program line where we start to import the libraries that are used in this program. The port’s function are as follows:

1. **WiFi.h**: This library allows us to connect our Arduino board to a Wi-Fi network and provides functions for managing Wi-Fi connections, configuring network settings, and communicating over the internet.

2. **WiFiMulti.h**: This library is an extension of the Wi-Fi library and provides additional functionality for managing multiple Wi-Fi connections. It allows our Arduino board to automatically switch between different Wi-Fi networks if one becomes unavailable.

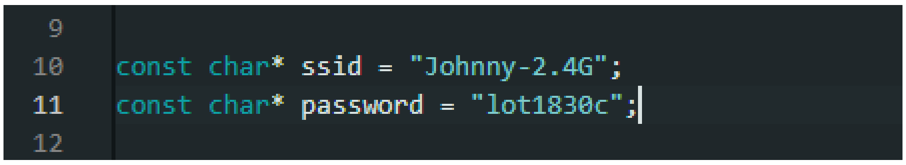
3. **Stepper.h**: The Stepper library is used for controlling stepper motors. It provides functions to control the speed, direction, and steps of a stepper motor, enabling precise movement and positioning.

4. **WiFiClientSecure.h**: This library is an extension of the Wi-Fi library and provides secure communication over HTTPS. It allows us to establish secure connections to web servers using SSL/TLS encryption.

5. **UniversalTelegramBot.h**: This library enables communication with the Telegram Bot API. It provides functions for sending and receiving messages, handling commands, and interacting with Telegram bots from our Arduino board.

6. **ArduinoJson.h**: The ArduinoJson library is used for parsing and generating JSON (JavaScript Object Notation) data. It allows our Arduino board to read and write JSON data, making it easier to work with web APIs, configuration files, and other data formats.

7. **Ultrasonic.h**: The Ultrasonic library is designed for working with ultrasonic distance sensors. It provides functions for reading distance measurements from ultrasonic sensors, allowing our Arduino board to detect objects or calculate distances based on the time it takes for sound waves to bounce back.



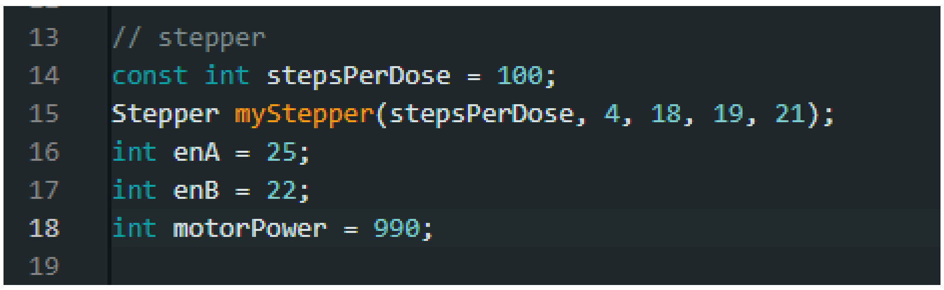
```
const char* ssid = "Johnny-2.4G";
const char* password = "lot1830c";
```

Figure 4.15 Wifi SSID and password details

Figure 4.15 shows the Wi-Fi SSID and password details. The purpose of each line:

1. **const char* ssid = "Johnny-2.4G"**: This line defines a constant character array (string) variable named ssid and assigns it the value "Johnny-2.4G". The ssid represents the name of the Wi-Fi network we want our Arduino board to connect to. We should replace "Johnny-2.4G" with the actual SSID of our Wi-Fi network.

2. **const char* password = "lot1830c"**: This line defines another constant character array (string) variable named password and assigns it the value "lot1830c". The password represents the password or passphrase required to connect to the Wi-Fi network specified in the ssid variable. We should replace "lot1830c" with the actual password for our Wi-Fi network.



```
// stepper
const int stepsPerDose = 100;
Stepper myStepper(stepsPerDose, 4, 18, 19, 21);
int enA = 25;
int enB = 22;
int motorPower = 990;
```

Figure 4.16 Define stepper motor rotation

Figure 4.16 shows declaration of the NEMA 17 stepper motor. Each line function are as follows:

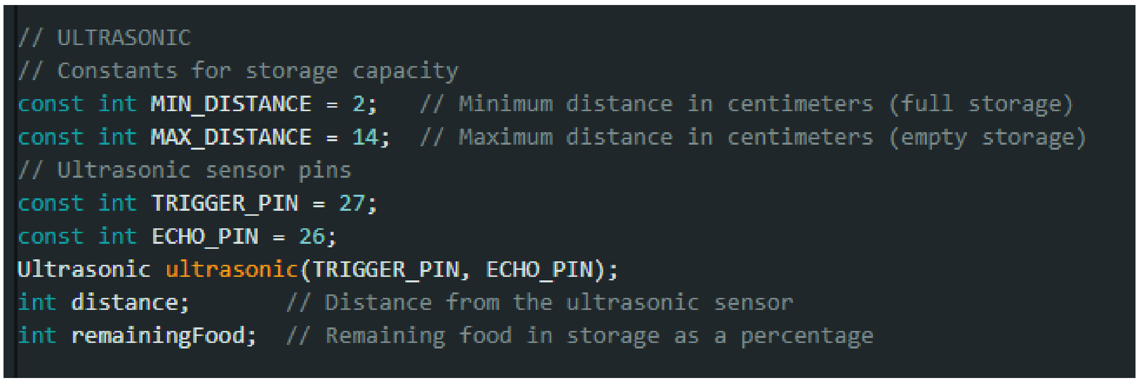
1. **const int stepsPerDose = 100**: A constant integer variable ‘stepsPerDose’ is declared with a value of 100. This is used to define the number of steps needed for each dose to be dispensed by the stepper motor.

2. **Stepper myStepper(stepsPerDose, 4, 18, 19, 21)**: An object ‘myStepper’ of the ‘Stepper’ class is created with ‘stepsPerDose’ as the number of steps per revolution and pins ‘4’ , ‘18’ , ‘19’ , and ‘21’ as the motor control pins.

3. **int enA = 25**: An integer variable ‘enA’ is declared and assigned the value of pin ‘25’ . This pin is used to enable/disable the first channel of the motor driver.

4. **int enB = 22**: An integer variable ‘enB’ is declared and assigned the value of pin ‘22’ . This pin is used to enable/disable the second channel of the motor driver.

5. **int motorPower = 990**: An integer variable ‘motorPower’ is declared and assigned the value of 990. This is used to set the maximum power for the stepper motor.



```
// ULTRASONIC
// Constants for storage capacity
const int MIN_DISTANCE = 2;   // Minimum distance in centimeters (full storage)
const int MAX_DISTANCE = 14;  // Maximum distance in centimeters (empty storage)
// Ultrasonic sensor pins
const int TRIGGER_PIN = 27;
const int ECHO_PIN = 26;
Ultrasonic ultrasonic(TRIGGER_PIN, ECHO_PIN);
int distance;       // Distance from the ultrasonic sensor
int remainingFood;  // Remaining food in storage as a percentage
```

Figure 4.17 Define SR-04 sensor

Figure 4.17 shows declaration of the ultrasonic sensor. Each line function are as follows:

1. **const int MIN_DISTANCE = 2**: This line declares a constant integer variable named MIN_DISTANCE and assigns it the value of 2. It represents the minimum distance between the sensor and the food inside the container. This depends on the size of the container.

2. **const int MAX_DISTANCE = 14**: This line declares another constant integer variable named MAX_DISTANCE and assigns it the value of 14. It represents the maximum

distance between the sensor and the food inside the container. This depends on the size of the container.

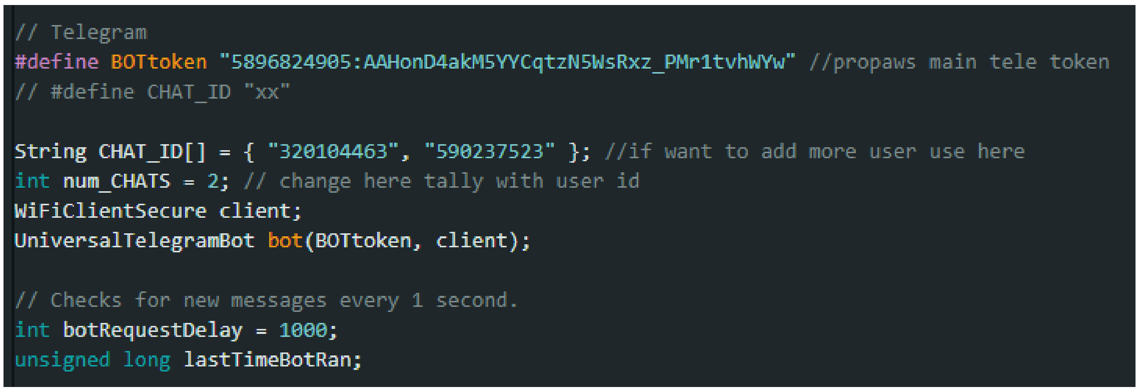
3. **const int TRIGGER_PIN = 27**: This line declares a constant integer variable named TRIGGER_PIN and assigns it the value of 27. It represents the pin number on the Arduino board to which the trigger pin of the ultrasonic sensor is connected.

4. **const int ECHO_PIN = 26**: This line declares another constant integer variable named ECHO_PIN and assigns it the value of 26. It represents the pin number on the Arduino board to which the echo pin of the ultrasonic sensor is connected.

5. **Ultrasonic ultrasonic(TRIGGER_PIN, ECHO_PIN)**: This line creates an instance of the Ultrasonic class from the Ultrasonic library. It initializes the ultrasonic object with the trigger and echo pin numbers provided (TRIGGER_PIN and ECHO_PIN) to enable distance measurement using an ultrasonic sensor.

6. **int distance**: This line declares an integer variable named distance without assigning it an initial value. It will be used to store the measured distance value obtained from the ultrasonic sensor.

7. **int remainingFood**: This line declares another integer variable named remainingFood without assigning it an initial value. It will be used to represent the amount of remaining food in the container as a return value.



```
// Telegram
#define BOTtoken "5896824905:AAHonD4akM5YYCqtzN5WsRxz_PMr1tvhWYw" //propaws main tele token
// #define CHAT_ID "xx"

String CHAT_ID[] = { "320104463", "590237523" }; //if want to add more user use here
int num_CHATS = 2; // change here tally with user id
WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);

// Checks for new messages every 1 second.
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;
```

Figure 4.18 Telegram token

Figure 4.18 shows declaration of Telegram. Each line function are as follows:

1. **BOTtoken "5896824905:AAHonD4akM5YYCqtzN5WsRxz_PMr1tvhWYw"**: This line uses a preprocessor directive (#define) to define a macro named BOTtoken

with the value "5896824905:AAHonD4akM5YYCqtzN5WsRxz_PMr1tvhWYw". It represents the access token for a Telegram bot.

2. **String CHAT_ID[] = { "320104463", "590237523" }**: This line declares a String array named CHAT_ID with two elements. It contains the chat IDs of the users or groups with which we want our Telegram bot to communicate.

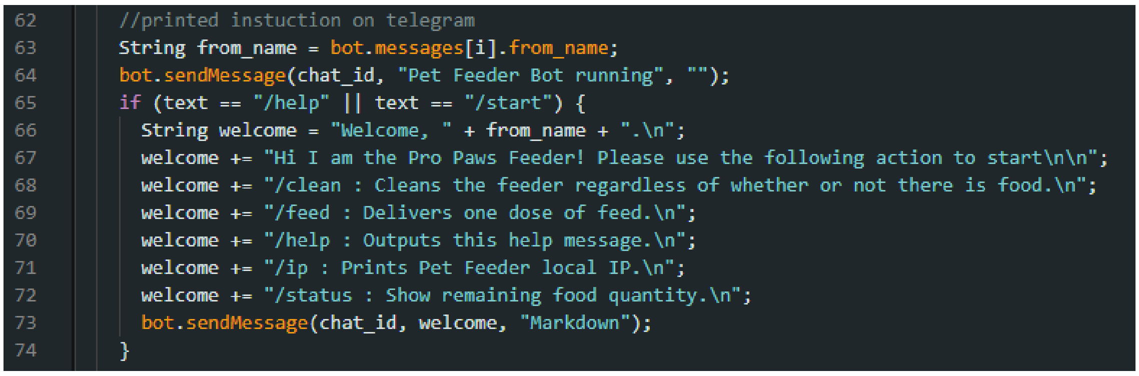
3. **int num_CHATS = 2**: This line declares an integer variable named num_CHATS and assigns it the value 2. It represents the number of chat IDs stored in the CHAT_ID array. Make sure to update this value if we add or remove chat IDs from the array.

4. **WiFiClientSecure client**: This line declares an instance of the WiFiClientSecure class named client. It is used for establishing a secure connection over HTTPS. This is commonly used in combination with the Telegram library to securely communicate with the Telegram Bot API.

5. **UniversalTelegramBot bot(BOTtoken, client)**: This line creates an instance of the UniversalTelegramBot class named bot. It initializes the bot object with the Telegram bot access token (BOTtoken) and the client object for making secure connections. This allows our Arduino board to interact with the Telegram Bot API using the provided access token.

6. **int botRequestDelay = 1000**: This line declares an integer variable named botRequestDelay and assigns it the value 1000. It represents the delay (in milliseconds) between consecutive requests made by the Telegram bot.

7. **unsigned long lastTimeBotRan**: This line declares an unsigned long variable named lastTimeBotRan without assigning an initial value. It is typically used to store the timestamp of the last time the bot performed an action or made a request. This variable can be useful for implementing timing or scheduling functionality in our code.



```
//printed instuction on telegram
String from_name = bot.messages[i].from_name;
bot.sendMessage(chat_id, "Pet Feeder Bot running", "");
if (text == "/help" || text == "/start") {
  String welcome = "Welcome, " + from_name + ".\n";
  welcome += "Hi I am the Pro Paws Feeder! Please use the following action to start\n\n";
  welcome += "/clean : Cleans the feeder regardless of whether or not there is food.\n";
  welcome += "/feed : Delivers one dose of feed.\n";
  welcome += "/help : Outputs this help message.\n";
  welcome += "/ip : Prints Pet Feeder local IP.\n";
  welcome += "/status : Show remaining food quantity.\n";
  bot.sendMessage(chat_id, welcome, "Markdown");
}
```

Figure 4.19 Printed instructions and response from bot

Figure 4.19 shows printed instructions and response from bot. These lines of code are related to giving user a set of instruction that can be used with ESP32 module. The functions of each line are as follows:

1. **String from_name = bot.messages[i].from_name**: This line retrieves the name of the sender from the from_name property of the bot.messages[i] object. It assigns the name to the from_name variable, which can be used later in the code.

2. **bot.sendMessage(chat_id, "Pet Feeder Bot started up", "")**: This line sends a message to the specified chat_id using the bot.sendMessage() function. It sends the message "Pet Feeder Bot started up" to the chat identified by chat_id. The empty string as the third argument represents an empty parse mode for the message.

3. The subsequent lines are conditionally executed if the text variable is equal to "/help" or "/start". This is done using the logical OR operator. It creates a welcome message that includes a brief explanation of the available commands:

- **"/clean"** command will clean the feeder, regardless of whether or not there is food.

- **"/feed"** command will deliver one dose of feed.

- **"/help"** command will output a help message.

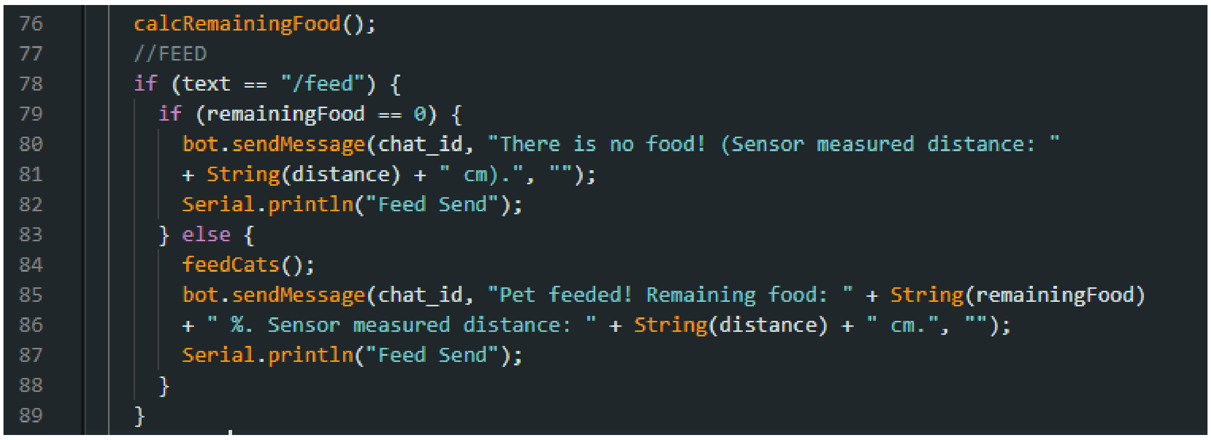
- **"/ip"** command will print the IP address of the pet feeder.

- **"/status"** command will show the remaining food quantity.

The welcome message is then sent back to the user using the ‘bot.sendMessage()’ function, with the message formatted in Markdown.

4. The following lines construct a welcome message stored in the welcome variable using string concatenation (+=). The message includes instructions and available commands for the user interacting with the bot.

5. **bot.sendMessage(chat_id, welcome, "Markdown")**: This line sends the constructed welcome message to the chat identified by chat_id using the bot.sendMessage() function. The third argument, "Markdown", specifies the parse mode for the message.



```
calcRemainingFood();
//FEED
if (text == "/feed") {
  if (remainingFood == 0) {
    bot.sendMessage(chat_id, "There is no food! (Sensor measured distance: "
    + String(distance) + " cm).", "");
    Serial.println("Feed Send");
  } else {
    feedCats();
    bot.sendMessage(chat_id, "Pet feeded! Remaining food: " + String(remainingFood)
    + " %. Sensor measured distance: " + String(distance) + " cm.", "");
    Serial.println("Feed Send");
  }
}
```

Figure 4.20 Function feed

Figure 4.20 shows the feed function. These lines of code are defining the instruction and situation if /feed is send to the bot. The functions of each line are as follows:

1. **if (text == "/feed") {**: This line checks if the text variable is equal to "/feed". If the condition is true, the code block within the curly braces following this line will be executed.

2. **if (remainingFood == 0) {**: This line checks if the remainingFood variable is equal to 0. If the condition is true, it means there is no food remaining.

3. **bot.sendMessage(chat_id, "There is no food! (Sensor measured distance: " + String(distance) + " cm).", "")**: This line sends a message to the chat identified by chat_id using the bot.sendMessage() function. The message informs the user that there is no food available, and it includes the measured distance from the sensor in centimetres.

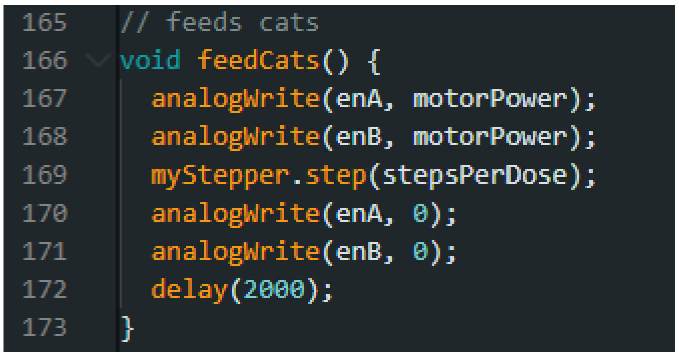
4. **Serial.println("Feed Send")**: This line prints "Feed Send" to the serial monitor. It is used for debugging or logging purposes and provides feedback when the "Feed" command is sent.

5. The else statement that follows indicates that if the condition in the first if statement is false (the text variable is not equal to "/feed"), the code block within the else block will be executed.

6. **feedCats()**: This line calls a function named feedCats(). It presumably performs the necessary actions to feed the cats or trigger the feeding mechanism.

7. **bot.sendMessage(chat_id, "Pet feeded! Remaining food: " + String(remainingFood) + " %. Sensor measured distance: " + String(distance) + " cm.", "")**: This line sends a message to the chat identified by chat_id using the bot.sendMessage() function. The message informs the user that the pet has been fed, and it includes the remaining food percentage (remainingFood) and the measured distance from the sensor in centimetres.

8. **Serial.println("Feed Send")**: This line prints "Feed Send" to the serial monitor, similar to line 4. It is used for debugging or logging purposes and provides feedback when the "Feed" command is sent.



```
// feeds cats
void feedCats() {
  analogWrite(enA, motorPower);
  analogWrite(enB, motorPower);
  myStepper.step(stepsPerDose);
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  delay(2000);
}
```

Figure 4.21 Declare feed function

Figure 4.21 shows the function used to dispense a single dose of pet food using a stepper motor. Here is what each line of code does:

1. **analogWrite(enA, motorPower)**: Sets the PWM signal to the motor driver enable pin A, which controls the power sent to the motor. ‘motorPower’ is a variable that determines the voltage level of the signal and is set to ‘990’.

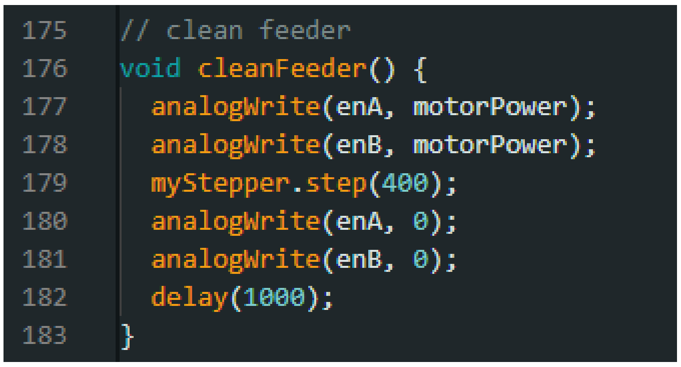
2. **analogWrite(enB, motorPower)**: Sets the PWM signal to the motor driver enable pin B, which controls the power sent to the motor.

3. **myStepper.step(stepsPerDose)**: Rotates the stepper motor by the number of steps required to dispense a single dose of cat food. The number of steps is defined by the ‘stepsPerDose’ variable, which is set to ‘100’.

4. **analogWrite(enA, 0)**: Turns off the PWM signal to the motor driver enable pin A, which stops the motor from rotating.

5. **analogWrite(enB, 0)**: Turns off the PWM signal to the motor driver enable pin B, which stops the motor from rotating.

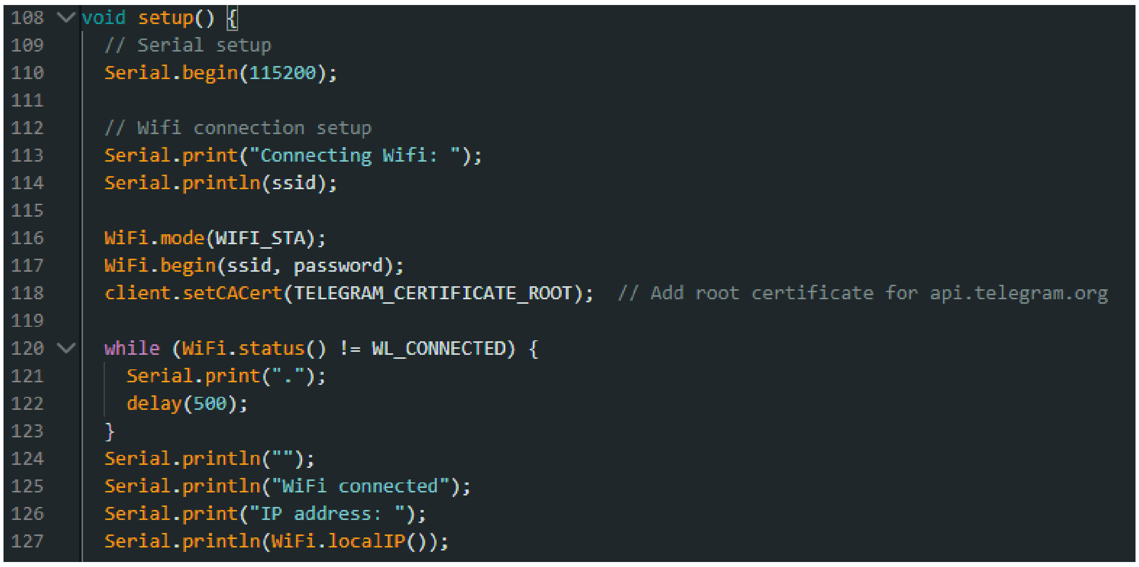
6. **delay(2000)**: Pauses the program execution for 2 seconds to allow the motor to complete the dose dispensing process.



```
// clean feeder
void cleanFeeder() {
  analogWrite(enA, motorPower);
  analogWrite(enB, motorPower);
  myStepper.step(400);
  analogWrite(enA, 0);
  analogWrite(enB, 0);
  delay(1000);
}
```

Figure 4.22 Declare clean function

Figure 4.22 shows the declaration of clean function. This function ‘cleanFeeder()’ controls the stepper motor to clean the feeder by rotating it 400 steps. It sets the ‘enA’ and ‘enB’ pins to ‘motorPower’ value to enable the motor driver to drive the stepper motor, then calls ‘myStepper.step(400)’ to rotate the stepper motor 400 steps, and finally, it sets ‘enA’ and ‘enB’ pins to 0 to stop the motor. It adds a delay of 1 second at the end of the function to make sure the motor stops completely.



```
void setup() {
  // Serial setup
  Serial.begin(115200);

  // Wifi connection setup
  Serial.print("Connecting Wifi: ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  client.setCACert(TELEGRAM_CERTIFICATE_ROOT);  // Add root certificate for api.telegram.org

  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
```

Figure 4.23 Connecting to Wi-Fi from Telegram

Figure 4.23 shows the status command code. This code shows the call function and statements related to setting up and establishing a Wi-Fi connection on Telegram. The following code describes as below:

1. **Serial.print("Connecting Wifi: ")**: This line prints the message "Connecting Wifi: " to the serial monitor.

2. **Serial.println(ssid)**: This line prints the value of the ssid variable (the Wi-Fi network name) to the serial monitor.

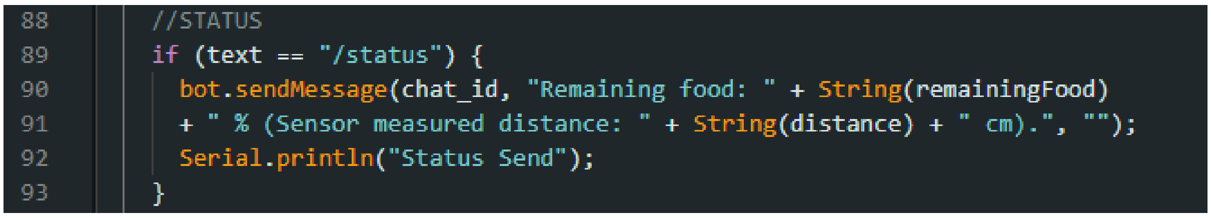
3. **WiFi.mode(WIFI_STA)**: This line sets the Wi-Fi mode to station mode, which allows the Arduino board to connect to an existing Wi-Fi network as a client.

4. **WiFi.begin(ssid, password)**: This line initiates the connection to the Wi-Fi network specified by the ssid and password variables.

5. **client.setCACert(TELEGRAM_CERTIFICATE_ROOT)**: This line sets the root certificate for the client object, which is used for establishing a secure connection with the Telegram API. The TELEGRAM_CERTIFICATE_ROOT represents the root certificate for api.telegram.org.

6. The following lines create a loop that continuously checks the connection status until it is successfully connected. It prints dots to the serial monitor (Serial.print(".")) and introduces a delay of 500 milliseconds (delay(500)) between each check. Once the

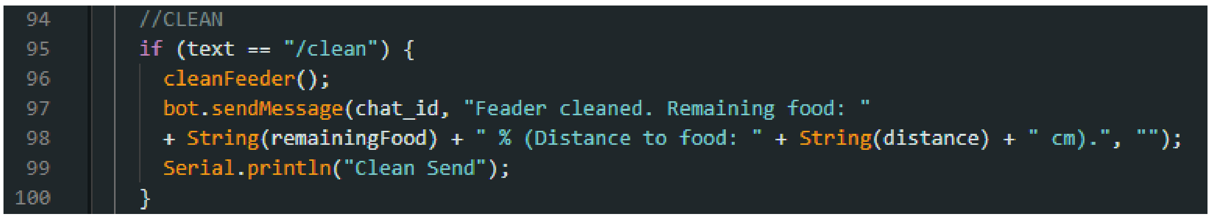
connection is established, it prints messages to the serial monitor, including the local IP address.



```
//STATUS
if (text == "/status") {
  bot.sendMessage(chat_id, "Remaining food: " + String(remainingFood)
  + " % (Sensor measured distance: " + String(distance) + " cm).", "");
  Serial.println("Status Send");
}
```

Figure 4.24 Status command

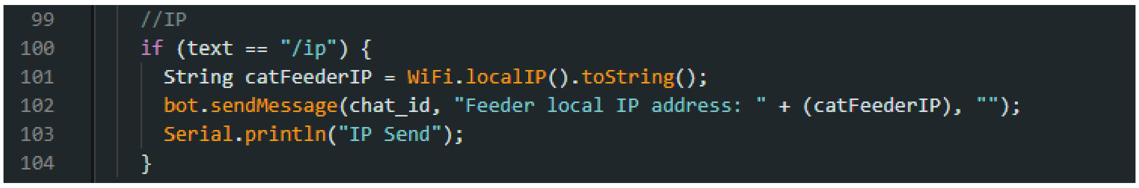
Figure 4.24 shows the status command code. This code block is a conditional statement that checks if the received text from the Telegram bot is "/status". If the received text is "/status", the ‘calcRemainingFood()’ function is called to calculate the remaining food percentage and ultrasonic measured distance. Then, the ‘bot.sendMessage()’ function is used to send a message back to the user with the remaining food percentage and ultrasonic measured distance information. The percentageFood value is converted to a string using the ‘String()’ function, and the distance is also converted to a string using the same method. The message is sent using the ‘bot.sendMessage()’ function.



```
//CLEAN
if (text == "/clean") {
  cleanFeeder();
  bot.sendMessage(chat_id, "Feader cleaned. Remaining food: "
  + String(remainingFood) + " % (Distance to food: " + String(distance) + " cm).", "");
  Serial.println("Clean Send");
}
```

Figure 4.25 Clean command

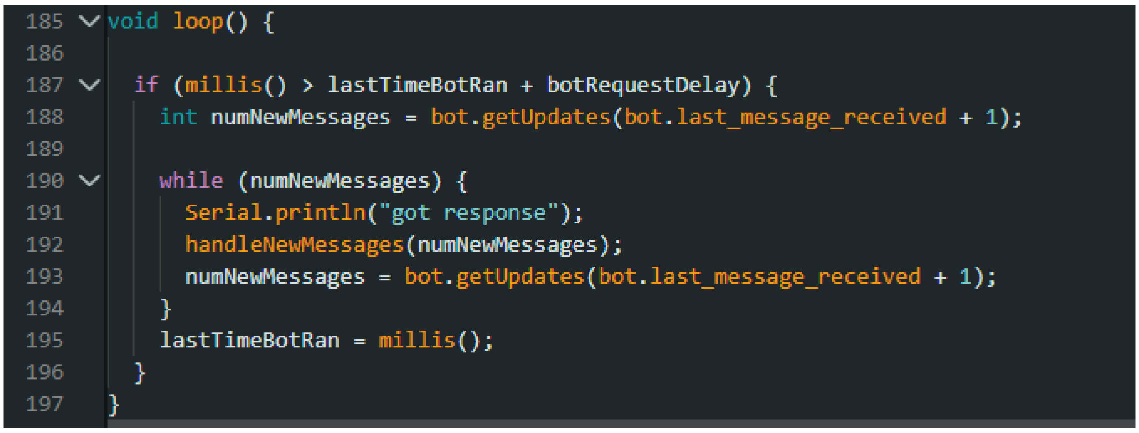
In Figure 4.25 shows clean command. This section of code, the function ‘feedCats()’ is called to dispense a portion of food. Then, a message is sent to the Telegram bot indicating that the feeder has been cleaned and the remaining food percentage and distance to the food are displayed.



```
//IP
if (text == "/ip") {
  String catFeederIP = WiFi.localIP().toString();
  bot.sendMessage(chat_id, "Feeder local IP address: " + (catFeederIP), "");
  Serial.println("IP Send");
}
```

Figure 4.26 IP command

Figure 4.26 shows the IP command. This code block manages the "/ip" command received by the Telegram bot. When this command is received, the local IP address of the device running the code is retrieved using the ‘WiFi.localIP()’ function and converted to a String. The Telegram bot then sends a message back to the user containing the local IP address.



```
void loop() {

  if (millis() > lastTimeBotRan + botRequestDelay) {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

    while (numNewMessages) {
      Serial.println("got response");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastTimeBotRan = millis();
  }
}
```

Figure 4.27 Message checkup function

Figure 4.27 shows the message check-up function in Telegram bot. The following line is described as below:

1. **if (millis() > lastTimeBotRan + botRequestDelay)** : This line checks if the current time (obtained using the millis() function) is greater than the sum of the previous time the bot ran (lastTimeBotRan) and the bot request delay (botRequestDelay). This condition is used to control the frequency at which the bot performs certain actions.

2. **int numNewMessages = bot.getUpdates(bot.last_message_received + 1):** This line retrieves the number of new messages from the Telegram bot using the getUpdates() function of the bot object. It passes the ID of the last received message plus one as a parameter to retrieve only the new messages since the last check. The number of new messages is stored in the numNewMessages variable.

3. The following lines create a loop that handles the new messages:

- **while (numNewMessages)** : This line initiates a loop that executes as long as there are new messages to handle.

- **Serial.println("got response")**: This line prints the message "got response" to the serial monitor. It serves as a debugging or logging message indicating that a response has been received from the Telegram bot.

- **handleNewMessages(numNewMessages)**: This line calls a function named handleNewMessages() and passes the numNewMessages variable as an argument. This function is responsible for processing and handling the new messages received from the Telegram bot.

- **numNewMessages = bot.getUpdates(bot.last_message_received + 1)**: This line updates the numNewMessages variable by retrieving the number of new messages again. This ensures that any new messages received during the execution of handleNewMessages() are accounted for in the next iteration of the loop.

- **lastTimeBotRan = millis()**: This line updates the value of lastTimeBotRan with the current time using the millis() function. It is used to keep track of the last time the bot performed its actions.

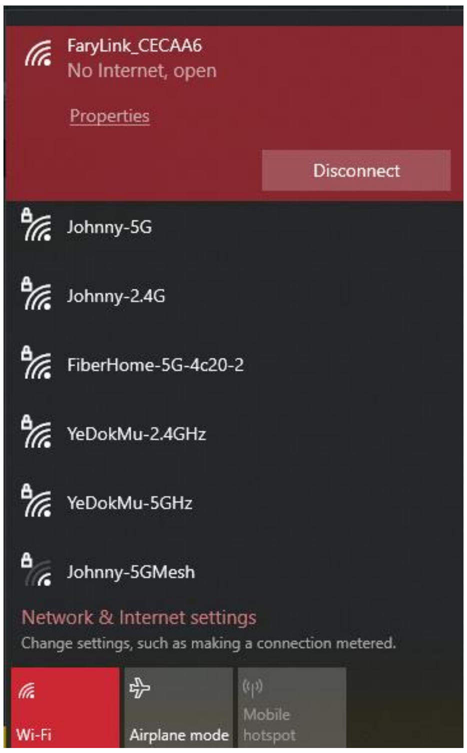
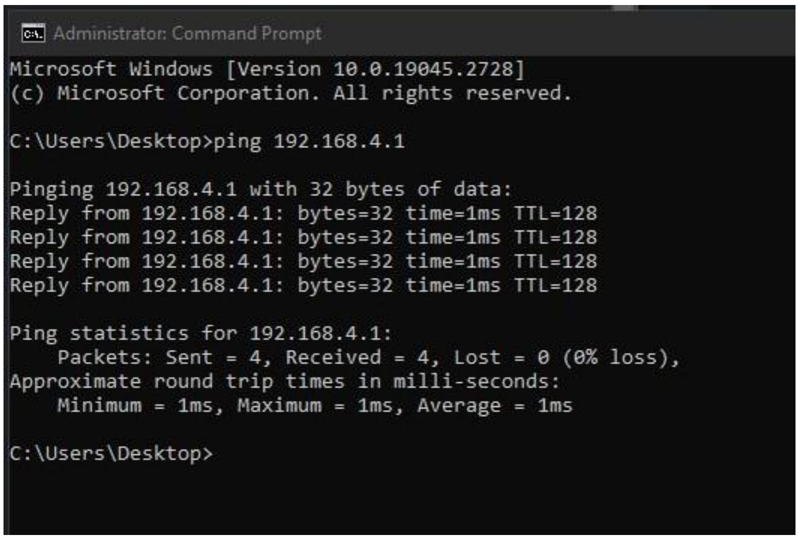

FaryLink_CECAA6
No Internet, open
Properties
Disconnect
Johnny-5G
Johnny-2.4G
FiberHome-5G-4c20-2
YeDokMu-2.4GHz
YeDokMu-5GHz
Johnny-5GMesh
Network & Internet settings
Change settings, such as making a connection metered.
Wi-Fi
Airplane mode
Mobile hotspot

Figure 4.28 ESP32 online

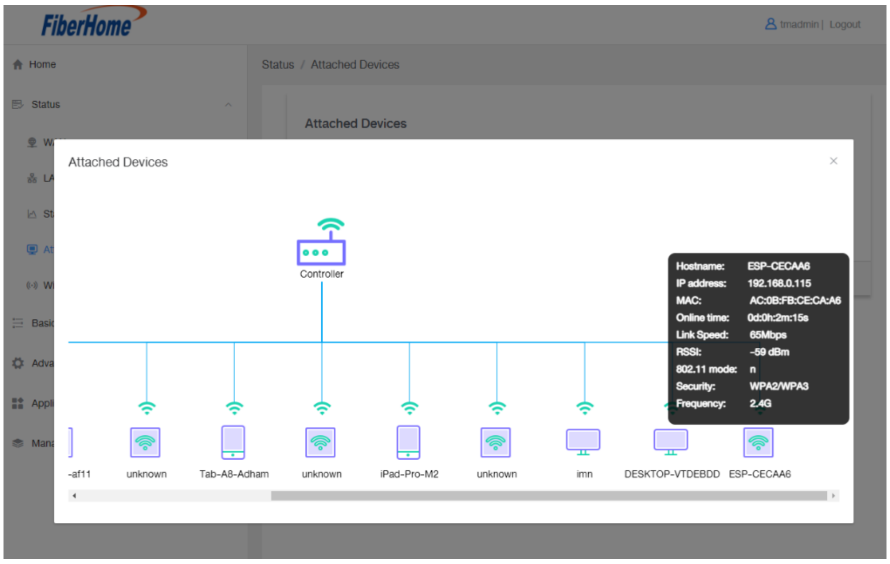
Figure 4.28 shows that ESP32 is online and ready to pair with end device.



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Desktop>ping 192.168.4.1

Pinging 192.168.4.1 with 32 bytes of data:
Reply from 192.168.4.1: bytes=32 time=1ms TTL=128
Reply from 192.168.4.1: bytes=32 time=1ms TTL=128
Reply from 192.168.4.1: bytes=32 time=1ms TTL=128
Reply from 192.168.4.1: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.4.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms

C:\Users\Desktop>
```

Figure 4.29 Verify ESP32 connection

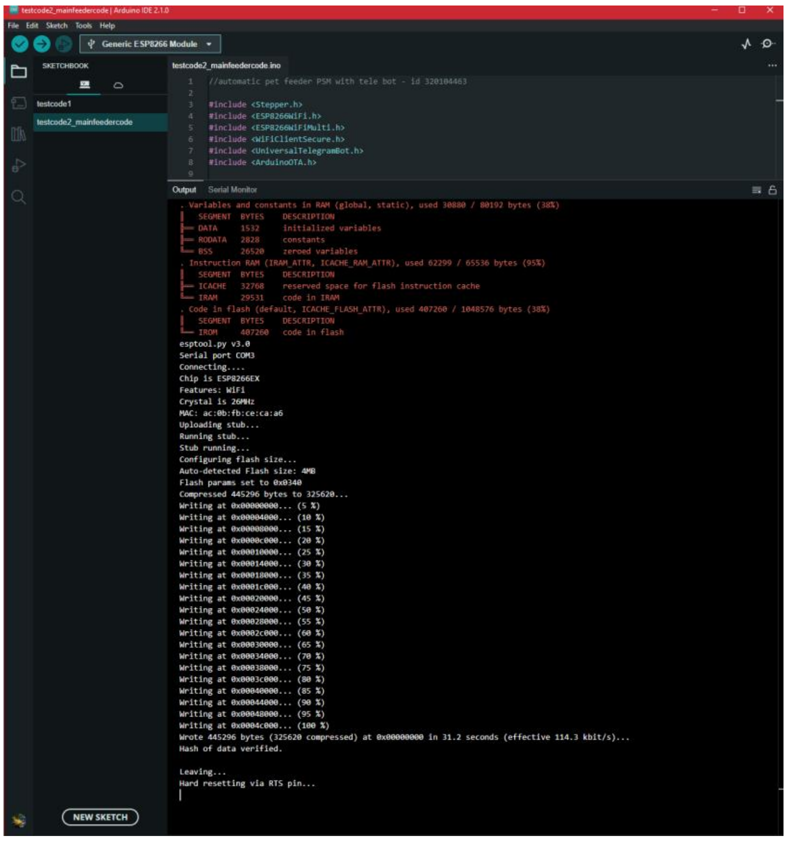
Figure 4.29 show that my computer is connected to ESP32 access point, and we ping it default IP address. As result, the connection is successfully established between end device and ESP32.



```
Hostname:      ESP-CECAA6
IP address:    192.168.0.115
MAC:           AC:0B:FB:CE:CA:A6
Online time:   0d:0h:2m:15s
Link Speed:    65Mbps
RSSI:          -59 dBm
802.11 mode:   n
Security:      WPA2/WPA3
Frequency:     2.4G
```

Figure 4.30 Connection to local network

As we can see in Figure 4.30, the device named ESP-CECAA is connected to local network as shown in the local network terminal.

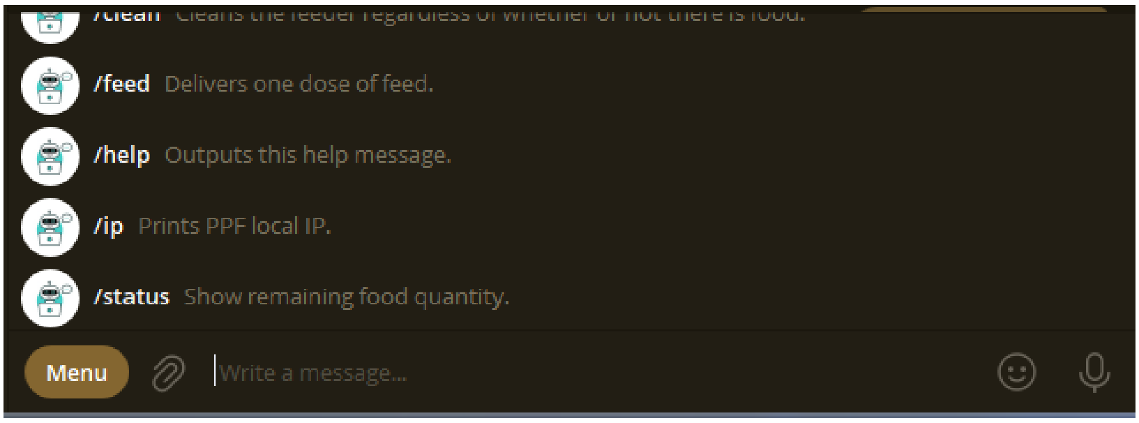

testcode2_mainfeedercode | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Generic ESP8266 Module
SKETCHBOOK
testcode1
testcode2_mainfeedercode
NEW SKETCH
testcode2_mainfeedercode.ino
Output
Serial Monitor

```
//automatic pet feeder PSM with tele bot - id 320104463

#include <Stepper.h>
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoOTA.h>
```

```
. Variables and constants in RAM (global, static), used 30880 / 80192 bytes (38%)
║   SEGMENT  BYTES    DESCRIPTION
╠══ DATA     1532     initialized variables
╠══ RODATA   2828     constants
╚══ BSS      26520    zeroed variables
. Instruction RAM (IRAM_ATTR, ICACHE_RAM_ATTR), used 62299 / 65536 bytes (95%)
║   SEGMENT  BYTES    DESCRIPTION
╠══ ICACHE   32768    reserved space for flash instruction cache
╚══ IRAM     29531    code in IRAM
. Code in flash (default, ICACHE_FLASH_ATTR), used 407260 / 1048576 bytes (38%)
║   SEGMENT  BYTES    DESCRIPTION
╚══ IROM     407260   code in flash
esptool.py v3.0
Serial port COM3
Connecting....
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: ac:0b:fb:ce:ca:a6
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Flash params set to 0x0340
Compressed 445296 bytes to 325620...
Writing at 0x00000000... (5 %)
Writing at 0x00004000... (10 %)
Writing at 0x00008000... (15 %)
Writing at 0x0000c000... (20 %)
Writing at 0x00010000... (25 %)
Writing at 0x00014000... (30 %)
Writing at 0x00018000... (35 %)
Writing at 0x0001c000... (40 %)
Writing at 0x00020000... (45 %)
Writing at 0x00024000... (50 %)
Writing at 0x00028000... (55 %)
Writing at 0x0002c000... (60 %)
Writing at 0x00030000... (65 %)
Writing at 0x00034000... (70 %)
Writing at 0x00038000... (75 %)
Writing at 0x0003c000... (80 %)
Writing at 0x00040000... (85 %)
Writing at 0x00044000... (90 %)
Writing at 0x00048000... (95 %)
Writing at 0x0004c000... (100 %)
Wrote 445296 bytes (325620 compressed) at 0x00000000 in 31.2 seconds (effective 114.3 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Figure 4.31 Arduino code compile complete

In Figure 4.31 shows the completion of compiling and uploading the code into the ESP32.



/clean Cleans the feeder regardless of whether or not there is food.

/feed Delivers one dose of feed.

/help Outputs this help message.

/ip Prints PPF local IP.

/status Show remaining food quantity.

Figure 4.32 Commands in Telegram

Figure 4.32 shows commands in Telegram. Sets of command initially to help use prompt the actions easier.

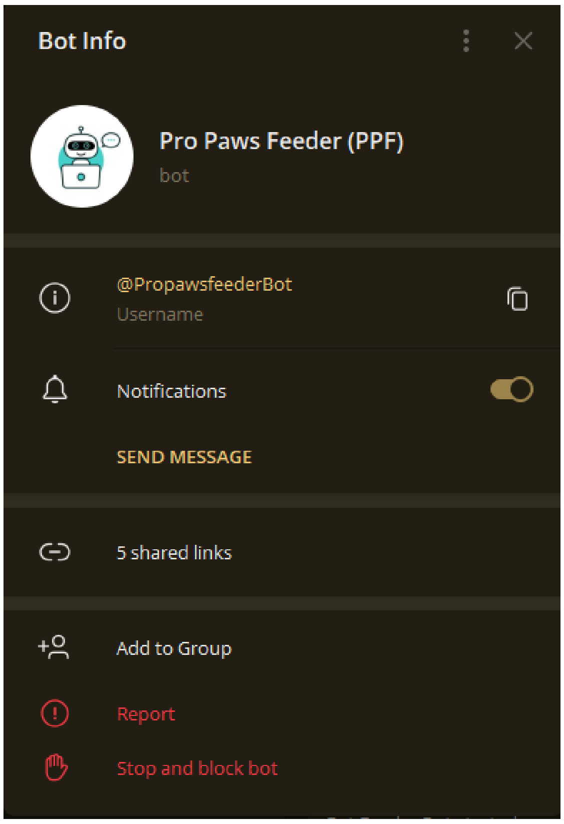
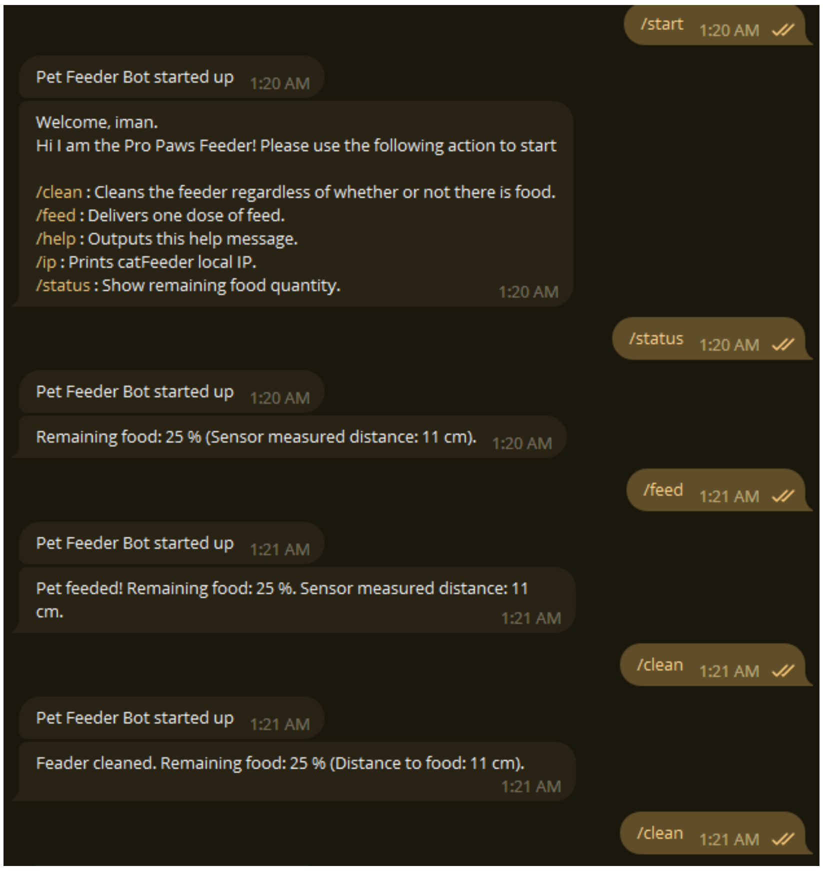


Figure 4.33 PPF bot Telegram

Figure 4.33 shows the PPF bot on Telegram. The bot that is used in this system. The PPF token (**5896824905:AAHonD4akM5YYCqtzN5WsRxz_PMr1tvhWYw**) used to programme with the ESP32 obtained from the Bot Father.



```
/start 1:20 AM
Pet Feeder Bot started up 1:20 AM
Welcome, iman.
Hi I am the Pro Paws Feeder! Please use the following action to start

/clean : Cleans the feeder regardless of whether or not there is food.
/feed : Delivers one dose of feed.
/help : Outputs this help message.
/ip : Prints catFeeder local IP.
/status : Show remaining food quantity. 1:20 AM
/status 1:20 AM
Pet Feeder Bot started up 1:20 AM
Remaining food: 25 % (Sensor measured distance: 11 cm). 1:20 AM
/feed 1:21 AM
Pet Feeder Bot started up 1:21 AM
Pet feeded! Remaining food: 25 %. Sensor measured distance: 11
cm. 1:21 AM
/clean 1:21 AM
Pet Feeder Bot started up 1:21 AM
Feader cleaned. Remaining food: 25 % (Distance to food: 11 cm).
1:21 AM
/clean 1:21 AM
```

Figure 4.34 Set of command and response

As we can see in Figure 4.38, when we send '/start', the bot will respond by sending the list of commands. The '/status' command is to check the food level and the bot will respond and send the remaining food in the storage. The '/feed' will give one portion off food that has been set based on Figure 4.23. The '/ip' will show the current pet feeder ip address. Lastly, '/clean' will empty the storage regardless of food left in the container by rotating the motor twice in revolution.

4.3 Testing and Result Discussion

User Acceptance Test (UAT) was conducted to evaluate the functionality of the Arduino-based pet feeder, Pro Paws Feeder. The testing format involved gathering data on the bot's performance, testing procedure, respondents, measurement method, and project-related findings. The UAT (User Acceptance Testing) revealed that the bot successfully connected to the specified Wi-Fi network, established a secure connection with the Telegram API, and responded to user commands appropriately. The bot effectively handled commands such as feeding the pet, providing status updates, and delivering help messages.

The testing procedure involved simulating user interactions and monitoring the bot's responses. Respondents found the bot's functionality intuitive and were satisfied with its performance. Measurement methods included analysing the accuracy of food remaining measurements and monitoring the successful delivery of messages. Overall, the Pro Paws Feeder achieved its objectives by providing reliable communication with users, feeding the pet accurately, and delivering prompt responses. Appendices containing the UAT details, including test data, procedure, and results, are attached for further reference.

User Acceptance Test

Below is the UAT in Pro Paws Feeder by possible user of the system:

No	Module	Activities	Status	Comment
1.	Telegram	Login account	Yes (✓) No( )	
		Add bot to Telegram	Yes (✓) No( )	
		Start bot	Yes (✓) No( )	
2.	/Start & /Help command bot	Send & view user command in chat	Yes (✓) No( )	
		Bot response	Yes (✓) No( )	
3.	/Clean command bot	Send & view user command	Yes (✓) No( )	
		Bot response	Yes (✓) No( )	
		Empty container	Yes (✓) No( )	
4.	/IP command bot	Bot response	Yes (✓) No( )	
5.	/Feed command bot	Bot response	Yes (✓) No( )	
		Bowl filled	Yes (✓) No( )	
6.	/Status command bot	Bot response with sensor reading food level	Yes (✓) No( )	

This test for user has been performed by:

Name: AKMAL NAZRIEN AZIM BIN AHMAD

Matric ID: TF19037

Signature: azim

Date : 18/5/2023

No	Module	Activities	Status	Comment
1.	Telegram	Login account	Yes (✓) No( )	
		Add bot to Telegram	Yes (✓) No( )	
		Start bot	Yes (✓) No( )	
2.	/Start & /Help command bot	Send & view user command in chat	Yes (✓) No( )	
		Bot response	Yes (✓) No( )	
3.	/Clean command bot	Send & view user command	Yes (✓) No( )	
		Bot response	Yes (✓) No( )	
		Empty container	Yes (✓) No( )	
4.	/IP command bot	Bot response	Yes (✓) No( )	
5.	/Feed command bot	Bot response	Yes (✓) No( )	
		Bowl filled	Yes (✓) No( )	
6.	/Status command bot	Bot response with sensor reading food level	Yes (✓) No( )	

This test for user has been performed by:

Name: AMIRUL HAKIM BIN ZAKARIA

Matric ID: TF19021

Signature: amirul

Date : 18/5/2023

No	Module	Activities	Status	Comment
1.	Telegram	Login account	Yes (✓) No( )	
		Add bot to Telegram	Yes (✓) No( )	
		Start bot	Yes (✓) No( )	
2.	/Start & /Help command bot	Send & view user command in chat	Yes (✓) No( )	
		Bot response	Yes (✓) No( )	
3.	/Clean command bot	Send & view user command	Yes (✓) No( )	
		Bot response	Yes (✓) No( )	
		Empty container	Yes (✓) No( )	
4.	/IP command bot	Bot response	Yes (✓) No( )	
5.	/Feed command bot	Bot response	Yes (✓) No( )	
		Bowl filled	Yes (✓) No( )	
6.	/Status command bot	Bot response with sensor reading food level	Yes (✓) No( )	

This test for user has been performed by:

Name: AHMAD HILMAN BIN AHMAD BADRUDDIN

Matric ID: CB20093

Signature: amad

Date : 25/5/2023

No	Module	Activities	Status	Comment
1.	Telegram	Login account	Yes (✓) No( )	
		Add bot to Telegram	Yes (✓) No( )	
		Start bot	Yes (✓) No( )	
2.	/Start & /Help command bot	Send & view user command in chat	Yes (✓) No( )	
		Bot response	Yes (✓) No( )	
3.	/Clean command bot	Send & view user command	Yes (✓) No( )	
		Bot response	Yes (✓) No( )	
		Empty container	Yes (✓) No( )	
4.	/IP command bot	Bot response	Yes (✓) No( )	
5.	/Feed command bot	Bot response	Yes (✓) No( )	
		Bowl filled	Yes (✓) No( )	
6.	/Status command bot	Bot response with sensor reading food level	Yes (✓) No( )	

This test for user has been performed by:

Name: MUHAMMAD ADHAM BIN MOHAMED AZMI

Matric ID: TF19018

Signature: adham

Date : 25/5/2023

No	Module	Activities	Status	Comment
1.	Telegram	Login account	Yes (✓) No( )	
		Add bot to Telegram	Yes (✓) No( )	
		Start bot	Yes (✓) No( )	
2.	/Start & /Help command bot	Send & view user command in chat	Yes (✓) No( )	
		Bot response	Yes (✓) No( )	
3.	/Clean command bot	Send & view user command	Yes (✓) No( )	
		Bot response	Yes (✓) No( )	
		Empty container	Yes (✓) No( )	
4.	/IP command bot	Bot response	Yes (✓) No( )	
5.	/Feed command bot	Bot response	Yes (✓) No( )	
		Bowl filled	Yes (✓) No( )	
6.	/Status command bot	Bot response with sensor reading food level	Yes (✓) No( )	

This test for user has been performed by:

Name: WAN MUHAMMAD DZULKHAIRIE BIN WAN ZAHARI

Matric ID: CB20096

Signature: kherie

Date : 25/5/2023

CHAPTER 5

CONCLUSION

5.1 Introduction

In this Chapter, we will discuss and conclude all of our work towards Pro Paws Feeder from the early development until done.

i. Conclusion of the project:

In conclusion, the Pro Paws Feeder project has achieved its intended objectives. As the aims is to study the existing pet feeder on the market, to design a Pet Feeder that are affordable and efficient to use by pet owner and fulfilled the functionality of a good pet feeder. The bot performs more than it intended such successfully connects to a Wi-Fi network, establishes a secure connection with the Telegram API, and responds to user commands. The project's goals were met, and the bot demonstrated reliable performance in measuring remaining food, delivering messages, and providing an intuitive user experience. This project lays the groundwork for further advancements in automated pet feeding systems.

ii. Data retrieval and it’s fit into the project objectives:

The data collected throughout the project aligns well with the project's objectives. The testing phase involved gathering data on the bot's performance, including its ability to accurately measure remaining food and respond promptly to user commands. The data obtained confirms that the bot operates as intended and fulfils its purpose of providing reliable communication and automated feeding for pets.

iii. Methodology and project implementation conclusion:

The chosen methodology and implementation approach have proven to be effective in developing the Pro Paws Feeder. The use of Arduino IDE and relevant libraries facilitated the integration of Wi-Fi connectivity, Telegram API communication, and sensor data processing. The step-by-step implementation process ensured a systematic and organized development approach. The successful implementation of the bot demonstrates the soundness of the methodology employed.

iv. Future suggestions and enhancements of the project or research:

For future development and research, several suggestions and enhancements can be considered. Firstly, expanding the bot's capabilities by integrating additional sensors for pet monitoring, such as cameras or health monitoring sensors, could provide more comprehensive pet care functionality. Secondly, implementing scheduling features to automate feeding at specific times could enhance convenience for pet owners. Additionally, improving the user interface and interaction design to be more user-friendly and intuitive would further enhance the user experience. These suggestions aim to improve and expand upon the existing project, providing avenues for future research and development in the field of IoT-based pet care systems.

5.2 Research Constraint

i. Cost constraint:

The cost constraint played a significant role in the project, as it was essential to develop an Pro Paws Feeder within a limited budget. By carefully selecting cost-effective components and utilizing open-source libraries and software, the project was able to minimize expenses while still achieving the desired functionality. The cost constraint influenced decisions such as choosing affordable sensors, leveraging readily available Wi-Fi modules, and utilizing free development tools like the Arduino IDE. Throughout the project, cost considerations were continuously evaluated to ensure that the project remained within the allocated budget. If more budget were allocated, then the Pro Paws Feeder would be better in the of cosmetic and functionality.

ii. Time constraint:

The time constraint posed a challenge in the project, requiring efficient planning and execution to meet the project's objectives within the designated timeframe. A well-defined project timeline was established, outlining the development stages, testing phases, and completion milestones. To optimize time utilization, a structured Agile development process was adopted, involving iterative prototyping, testing, and refinement. Additionally, effective task management, regular progress monitoring, and timely adjustments were crucial in adhering to the time constraint. Despite the time limitations, the project was successfully completed within the prescribed timeline, meeting all the planned deliverables and objectives.

5.3 Future Work

Throughout the completion of this project, several suggestions and enhancements have emerged, showcasing the knowledge and contribution it brings to various stakeholders. Firstly, in terms of academic and university contribution, this project demonstrates the practical application of IoT technology in the field of pet care. By integrating Arduino, Wi-Fi connectivity, and Telegram API, the project provides a hands-on example of how IoT can be utilized to automate pet feeding and enhance communication between pet owners and their pets. This contributes to the university's reputation for fostering innovation and practical solutions in emerging technology domains.

Secondly, from a faculty perspective, this project adds value by enriching the curriculum with a real-world application of IoT. By incorporating this project into coursework or lab activities, students can gain hands-on experience in developing IoT-based systems, strengthening their understanding of hardware integration, wireless communication, and software development. The project also opens avenues for research opportunities, such as investigating the impact of automated feeding on pet behaviour or exploring energy-efficient algorithms for pet monitoring. Thus, it contributes to the faculty's mission of promoting experiential learning and encouraging research initiatives.

Lastly, in terms of societal impact, the Pro Paws Feeder addresses the needs of pet owners who seek an automated and convenient solution for pet feeding. By providing accurate food measurement, timely responses, and remote access through the Telegram platform, the project offers convenience, peace of mind, and improved pet care. It also encourages responsible pet ownership by ensuring that pets are adequately fed even when their owners are away. Additionally, the open-source nature of the project allows for community involvement, fostering collaboration and knowledge sharing among enthusiasts, which further contributes to the broader society.

Overall, this project's suggestions and enhancements extend beyond its technical aspects. It encompasses knowledge dissemination, academic contributions, and societal impacts, making it a valuable endeavour that enriches the university, faculty, and society as a whole.

REFERENCES

[1] A. S (Ed.). (2022, October 28). *8 Best Automatic Pet Feeders (Food Dispenser) in Malaysia 2022*. 8 Best Automatic Pet Feeders (Food Dispenser) in Malaysia 2022. Retrieved December 2, 2022, from http://productnation.co/my/28641/best-automatic-pet-feeder-malaysia/

[2] Half. (2021, October 28). *6 Basic SDLC Methodologies: Which One is Best?* 6 Basic SDLC Methodologies | Robert Half. Retrieved December 2, 2022, from https://www.roberthalf.com/blog/salaries-and-skills/6-basic-sdlc-methodologies-which-one-is-best

[3] Hamilton. (2020, January 31). *Agile Methodology: What is Agile Model in Software Testing?* Guru99. Retrieved December 2, 2022, from https://www.guru99.com/agile-scrum-extreme-testing.html

[4] Geeksforgeeks. (2020, April 28). *Functional vs Non-Functional Requirements - GeeksforGeeks*. Functional Vs Non-Functional Requirements. Retrieved December 2, 2022, from https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/

[5] Asana. (2022, June 13). *6 Project Constraints: Manage Them for Project Success [2022] Asana*. Asana. Retrieved December 2, 2022, from https://asana.com/resources/project-constraints

[6] H. (2022). *Bi-Directional Logic Level Converter Hookup Guide - SparkFun Learn*. Bi-Directional Logic Level Converter Hookup Guide - SparkFun Learn. Retrieved January 12, 2023, from https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide/all

[7] Hübschmann, I. (2020, August 10). *ESP8266 for IoT: A Complete Guide*. Nabto. Retrieved January 12, 2023, from https://www.nabto.com/esp8266-for-iot-complete-guide/

[8] T. (2021). Agile SDLC | Software Development Life Cycle - Javatpoint. www.javatpoint.com. Retrieved January 20, 2023, from https://www.javatpoint.com/agile-sdlc

[9] Hübschmann, I. (2020, August 10). ESP8266 for IoT: A Complete Guide. Nabto. Retrieved January 20, 2023, from https://www.nabto.com/esp8266-for-iot-complete-guide/

[10] J, M. (2013). GitHub - mgrenonville/arduino-cat-feeder: Automatic cat feeder build with Arduino. GitHub. Retrieved January 20, 2023, from https://github.com/mgrenonville/arduino-cat-feeder

APPENDIX A

PROPAWS FEEDER