Automatic Dog Feeder

by

Tatiya Seehatrakul

Martin Viray Ocampo

A report submitted in partial fulfillment of the requirements for
the degree of Bachelor of Engineering in
Mechatronics Engineering

Project Advisor

Asst. Prof. Dr. Narong Aphiratsakun

Examination Committee:

Dr. Jerapong Rojanarowan, Dr. Wisuwat Plodpradista,
Assoc. Prof. Dr. Jiradech Kongthon, Assoc. Prof. Dr. Vorapoj Patanavijit,
Dr. Amulya Bhattarai, Dr. Ehsan Ali

Assumption University
Vincent Mary School of Engineering
Thailand
March, 2023

Approved by Project Advisor:

Name: Asst. Prof. Dr. Narong Aphiratsakun

Signature:

Date:_____

Plagiarism verified by:

Name: Dr. Ehsan Ali

Signature:

Date: 02-Mar-2023 (Turnitin: 6%)

Abstract

Automated technology is becoming more and more trendy these days. One of the reasons why it is becoming more and more popular is its ease of use and the comfort it brings to the user. Keeping pets brings joy and comfort to one's life, but it comes with some inconveniences and additional tasks. This is where a pet feeder comes into play. This device can help users feed their pets. The system uses an Arduino Uno R3 as a controller. The device has three feeding portions that can be set in the display page when it initially launches. The user can set 1-3 portion size depending on the pet's weight and also set the feeding time in which the feeder dispenses food. Multiple sensors are placed in the feeder, one IR sensor in the container to notify the user when food levels are low and one more IR sensor to notify the user when to check the food bowl. The user is notified via the line application that sends a message to the user.

Contents

1 Introduction

1.1 Introduction of Topic

In today's age, automated technology is being integrated into the lives of many people to make their lives easier and more convenient. Owning pets can enrich people's lives, but it also come with responsibilities that can cause stress. Pet owners know all too well that feeding their pets provides joy and well-being, but sometimes they are too busy to give their pets the food they need, or they are not at home to check if they are eating or not. This is where a pet feeder comes in. This device replaces manual feeding of pets with a modern device [1]. Pet owners now have a convenient way to feed their pet. In addition, portion sizes are easier to control with this system, as users can set the right portion size for their pet. This could help pets maintain their weight and eliminate the problem of pet owners giving too much or too little food [2]. This pet feeder dispenses food for animals less than or equal to 5 kg.

1.2 Project overview

Nowadays, pet owners are not always around to feed their pets or may have difficulty doing so. Giving out optimal amounts of food at the right times and keeping track of the dog consistently can be challenging. Thus, this project aims to address this problem by automating the feeding process with various features using Arduino [3].

1.3 Project Objective

This project is about developing a pet feeder machine with various modes and functions. The food release mechanism is actuated with a servo motor. The important information is shown on an LCD screen. The following list contains the objectives that this project aims to achieve.

1. Food pellets are released according to the timers set by the user from the keypad at the machine. In our project, user can set up to 3 timers per day. The user has to press 'A' key to set the timer 1, 'B' key to set the timer 2, 'C' key to set the timer 3, and 'D' key as a manual release button.

2. Portion size from 1-3 can be chosen from the keypad before setting the timers. User can set the portion size according to the pet's size. The portion size must be entered before setting the timer or pressing the manual release button, otherwise the system will show "Please SET Portion, press '#' to set".

3. The user is not allowed to set an invalid portion size or time, otherwise the system will ask the user to input the number again.

4. The stepper motor with an auger spins each time the food is released and the spining time depends on the portion size set by the user.

5. There are two IR sensors in the system. The first one is placed at the exit of the food tube in such a way to prevent overfilling of the food pellet at the tray. If this sensor detects any object, the system will not be dispensing the food and the LCD screen shows "Food Tray FULL, CANNOT Dispense". Another one is placed at the bottom

of the food container to check the food level. If this sensor cannot detect any food, the system will remind the user to fill more food by showing "Food Supply LOW, Refill the Food" on the LCD screen.

6. A Line notification function notifies the user via the mobile application along with a message whenever the food tray is left out on the tray.

7. All the information is shown on 20x4 LCD screen including date, time, portion size, and all the timers set by the user. Additionally, the time is also shown on the TM1637 seven segment display module to be easily seen.

1.4 Scope of work

This automatic pet feeder is designed to make pet owner's life easier. With a food dispensing mechanism that is state of the art, with zero errors such as food pellets overflowing, food pellet getting jammed and others. This pet feeder has more sensors that prevent such errors from happening.

1.5 Motivation

Making pet owner's life easier is the motivation for our work. No longer do pet owners have to stress whether or not their pet is already fed. This device eliminates that stressful factor that busy pet owners have.

2 Literature Review

2.1 Background Study

The Arduino Uno Mega 2560 is used as a controller in this project to control numerous sensors, receive camera data, display LCD, and operate programs via electronic devices. The C and C++ programming language is used by this controller to interface with each component.

2.2 Theory

2.2.1 Dog Feeding Chart

Using the dog feeding chart as a standard, this dog feeder can determine how much food the dog needs. Puppies should be fed between 5 and 6 percent of their developing body weight, whereas older dogs only require about 2 percent of their optimum body weight. However, utilizing a dog feeding chart is a more precise approach to feed and help dogs have a healthy weight. Some dogs may require more, or less, food to maintain optimum weight.

The age of the dog does not determine the amount of food it needs to be given because from 1 - 4 month old puppies need wet food, because they haven't developed their teeth enough to eat dry food [4].

Table 2.2.1: Dog Feeding Chart.

Dog Size (kg)	Dry Food Feeding Amount (grams)
1	25
2	50
5	105

3 Components

3.1 Controller

3.1.1 Arduino Mega 2560

An Arduino mega is used in this project due to its larger memory space and the increased number of I/O pins compared to other boards like the Arduino Uno or Arduino Nano [5]. This board is particularly well suited for our project because it requires multiple inputs and outputs or require large amounts of data to be stored and processed. The C and C++ programming languages are are used to communicate with the various components in the system, allowing the controller to control and monitor the performance of these components. The program provides necessary instructions to the controller, which then interacts with the components.

3.1.2 ESP32

The ESP32 is a system-on-a-chip microcontroller designed for low-power IoT applications. It provides Wi-Fi and Bluetooth connectivity, integrated security features, low-power consumption, and a range of peripherals [6].

3.2 Logic Controller

3.2.1 Real Time Clock DS3231

The DS3231 is a dependable real-time clock that is simple to use and can be used for various purposes. Its high precision, low energy consumption, and temperature adjustment capabilities make it a top pick for projects that demand precise timekeeping and date recording. No matter if you're constructing a data logging setup, a time stamping machine, or a meteorological station, the DS3231 is a flexible and trustworthy real-time clock that will fulfill your requirements [7].

3.3 Drivers

3.3.1 A4988 Stepper Motor Driver

An A4988 module is used to control stepper motor to be operated easily. This module contains complete micro stepping driver with translator. It is a driver of micro-stepping for the bipolar type stepper motors. It is designed to drive a stepper motor with high current and high voltage, providing an effective and efficient solution for controlling the movement of the motor [8].

3.4 Sensors

3.4.1 IR sensor

An IR sensor is used to detect an object in front of it. It shoots out a ray of light that is not visible for the human eye that if interrupted, will send a signal to the controller [9]. This sensor is placed in such a way to prevent overfilling of the food pellet.

3.5 Actuator

3.5.1 42HB38 Nema 17 Stepper Motor

A stepper motor is used as an actuator to release the food pellet. It is a device that converts electrical power, in terms of pulses, into the axis of the movement [10]. There are two types in which a stepper motor is performed, that is the closed-loop whereas another one is the open-loop. Open-loop control works by the motor winding being supplied by a sequence of voltage surges that activates the stepper motor, on the other hand, In a closed-loop control, the stepper motor is activated via a sequential voltage surges or vector control [11]. In this project, the main component that will be dispensing the dog food is the Stepper Motor. The decision was made to use this instead of the Servo due to the possibility of the food pellets being caught in between the platform closing off the Servo Motor. The Stepper Motor allows for the dog food pellets to flow in a systematic formation that is less likely to cause a block in the system. Thus, the closed-loop stepper motor was implemented to the automatic dog feeder.

3.6 Display Module

3.6.1 Liquid Crystal Display 2004

The 2004 Liquid Crystal Display (LCD) is a type of display screen. It has 20 characters and 4 lines, allowing it to display more information in the compact space [12]. This module in our project used to display all the important information of the machine to the user, such as current date, current time, portion size and time entered by the user, and food level status.

3.6.2 I2C Serial Interface Module

An I2C LCD module is used with LCD screen to reduce the complexity of the wiring. Because the I2C protocol uses just two wires, it eliminates the need for multiple wires to be connected to the LCD [13]. This reduces the amount of wiring required in this project. This module can be communicated with an Arduino by adding the its library.

3.6.3 TM1637 Module

A TM1637 module is used as a digital clock to display the time for the user, by receiving the signal from RTC module. It is a driver chip provides an interface between the microcontroller and the display module, allowing the microcontroller to control the display. This module contains 4 digit seven-segment display. It supports various display modes, such as time display, date display, and temperature display, and can also be used to control the brightness of the display [14].

3.7 Other Components

3.7.1 LM2596 Module

LM2596 module or a buck converter is regulator that is used to step down DC voltage from 3,2-40V to 1.25-35V. It consists of an inductor, a switch, and a control circuit, which work together to regulate the output voltage. When the input voltage is applied, the switch turns on and off at a high frequency, causing the inductor to store and release energy. The control circuit adjusts the duty cycle to regulate the output voltage [15]. This module is used in the system in order to step down the DC voltage from 12V DC to 7.5V DC.

3.7.2 4*4 Matrix Keypad

A matrix keypad is used for the user to easily type numeric values and symbols to communicate with the system. It consists of 16 buttons arranged in a 4x4 matrix, with each button connecting two of the eight rows and columns in the matrix. To detect button presses, the microcontroller scans the rows and columns of the keypad to see which button is pressed. This is done by first setting one of the rows to high and reading the state of the columns. If a button is pressed, it will connect the row and column, which the microcontroller will detect as a high level on the corresponding column [16].

3.7.3 100-240V AC to 2A 12V DC Power Adapter

A 100-240V AC to 2A 12V DC Power Adapter is a power supply that transforms standard AC electricity from an electrical outlet into DC power for electronic devices. This adapter can accept AC voltage ranging from 100 to 240 volts, making it compatible for use in different parts of the world with varying voltage levels ,and for our case an Arduino Mega 2560 [17]. Initially, the Arduino is powered by the USB cable wire with the MG90S servo motor as a main actuator. After that, it is replaced with a stepper motor 42HB38 Nema 17 with a A4988 stepper motor driver. Generally, the minimum voltage used to drive motors is 12V [18]. This is the reason why a 100-240V AC to 2A 12V DC power adapter is used.

3.7.4 Heat Sink

A heat sink is a device used to dissipate heat away from a hot component in order to prevent it from overheating and failing. It works by providing a large surface area for the component to transfer its heat to. This large surface area increases the rate of heat transfer from the component to the surrounding air, allowing the heat to dissipate more efficiently [19].

3.7.5 SPST switch

An SPST switch is a type of an electrical switch that consists of a single pole or input terminal and a single output or throw terminal. An SPST switch typically consists of a lever or button that can be moved between two positions, one for ON and one for OFF. Pressing the line symbol of the switch means the switch is ON, the system becomes a closed circuit which allows the electricity to flow through. Pressing the circle symbol of the switch means the switch is OFF, the system becomes an open circuit which blocks the electricity to flow through. [20].

4 Systems Designs

4.1 Block Diagram

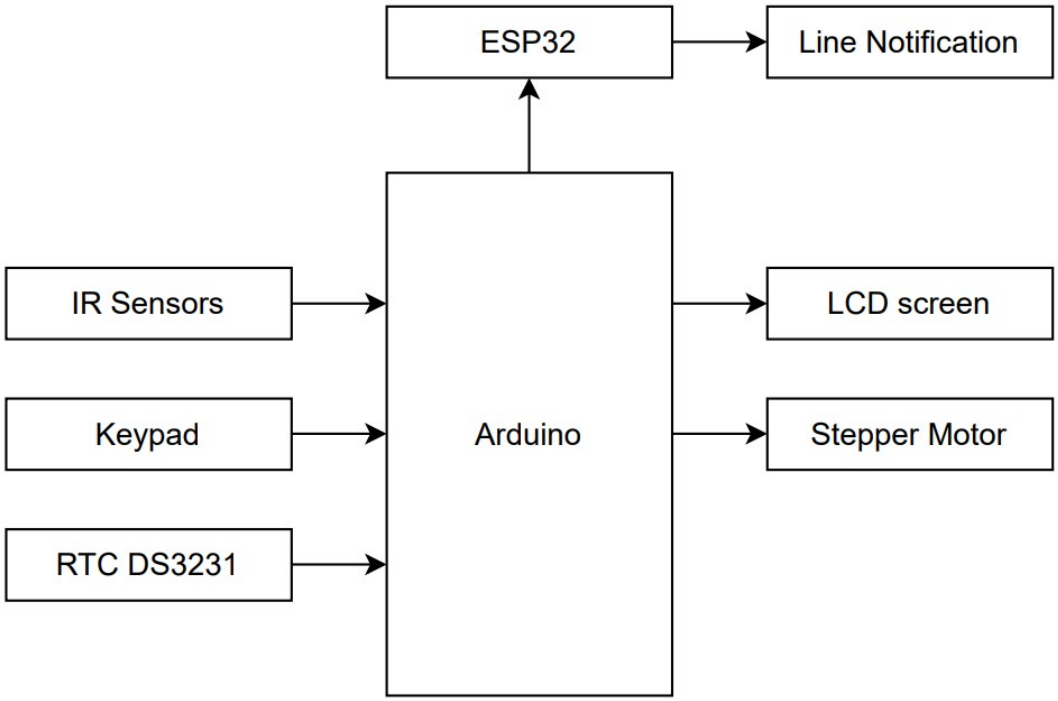

Fig. 4.1: Block Diagram of the System.

4.2 Flowchart Diagram

4.2.1 loop() Function

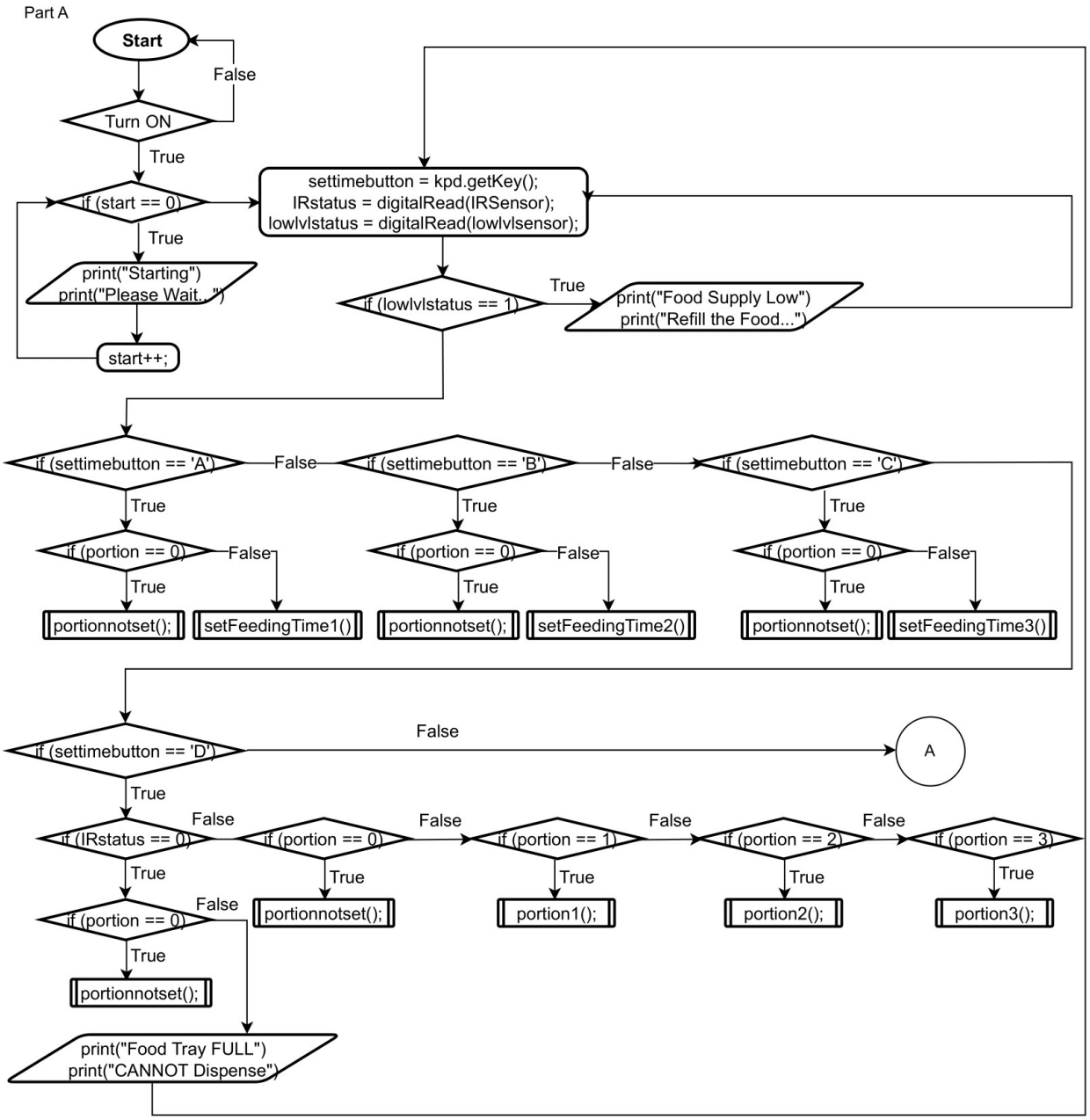

Fig. 4.2.1: Flowchart of void loop() Function Part A.

4.2.2 loop() Function cont.

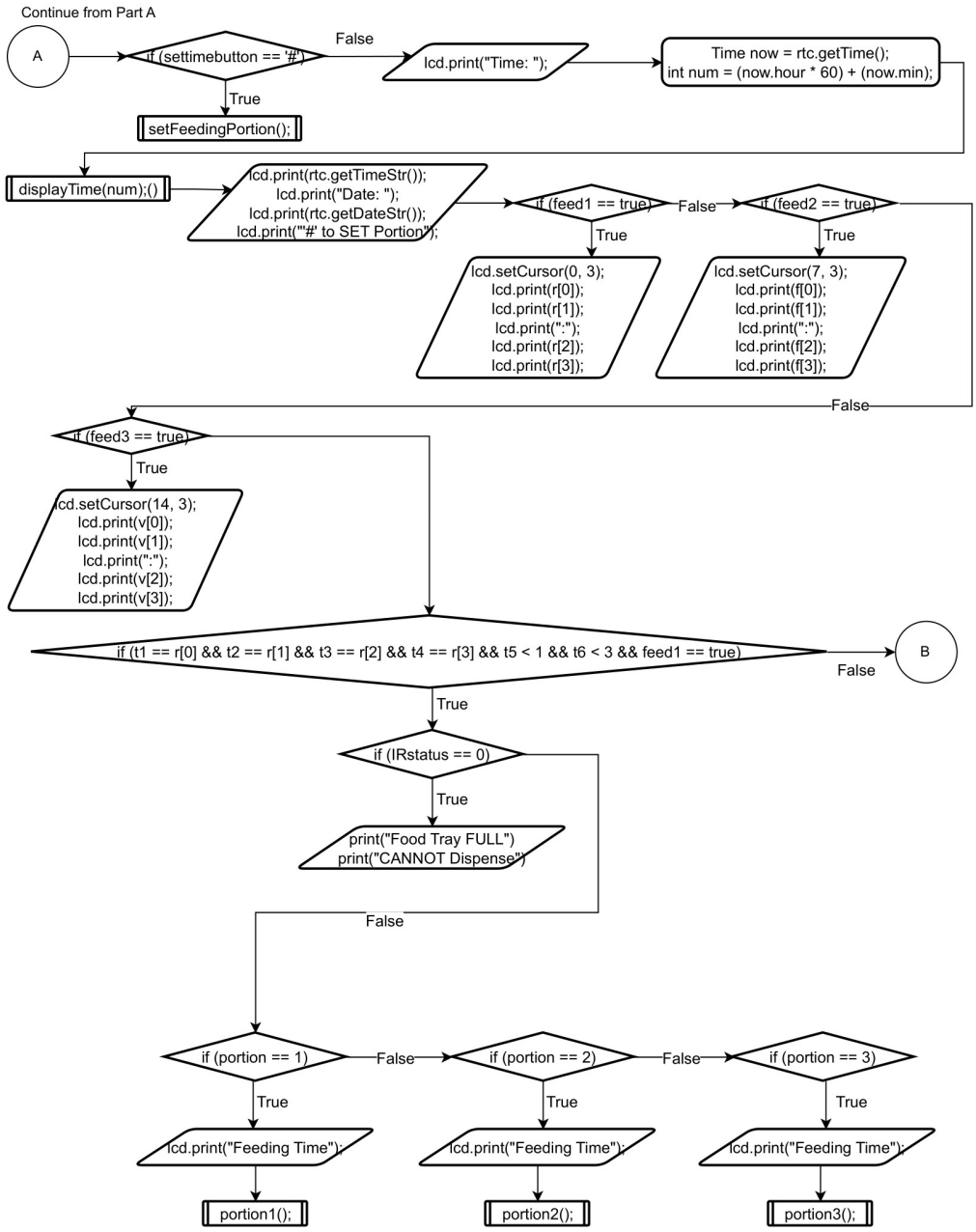

Fig. 4.2.2: Flowchart of void loop() Function continued from Part A.

4.2.3 loop() Function cont.

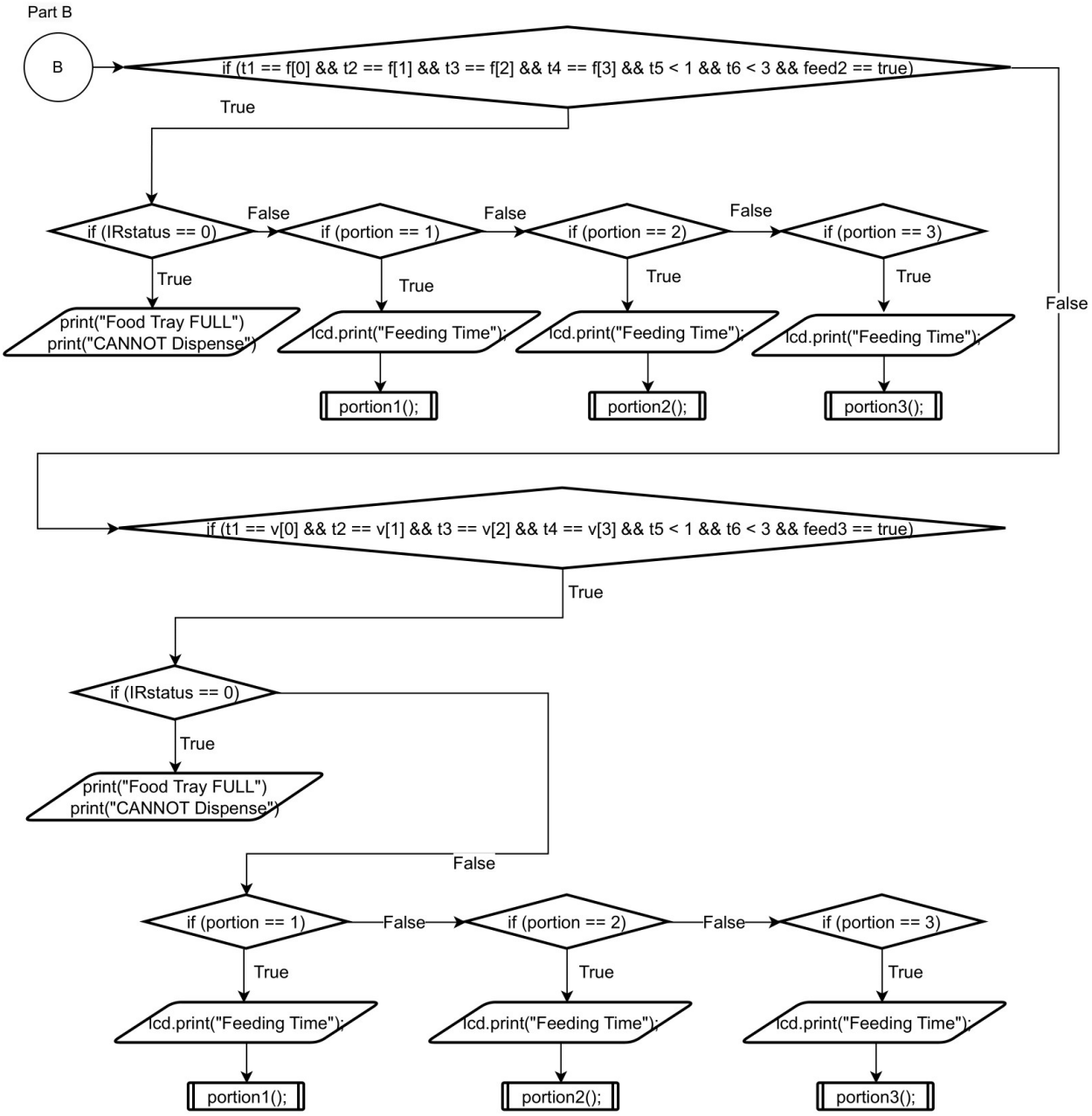

Fig. 4.2.3: Flowchart of void loop() function Part B.

4.2.4 setFeedingTime1() Function

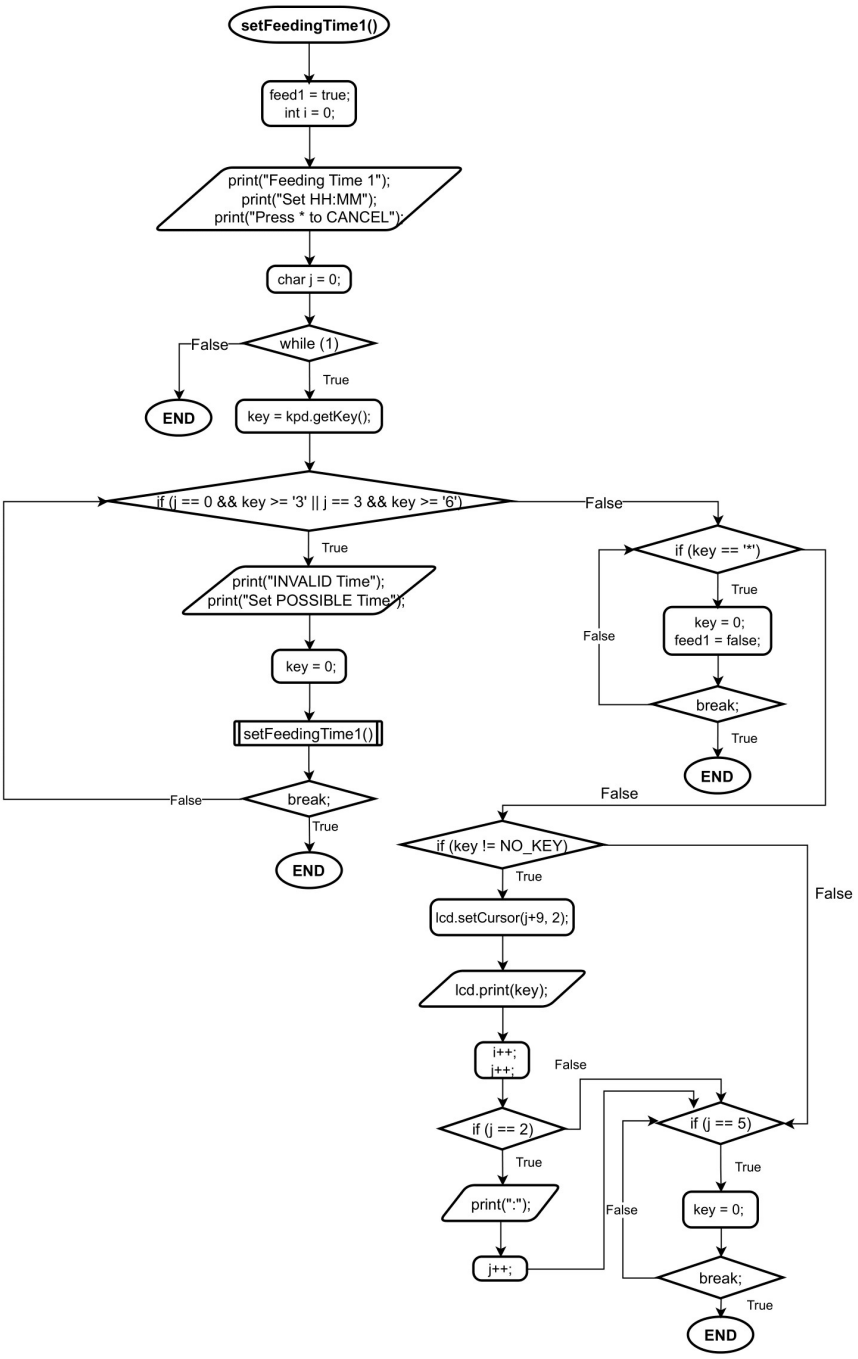

Fig. 4.2.4: Flowchart of void setFeedingTime1() Function.

4.2.5 setFeedingTime2() Function

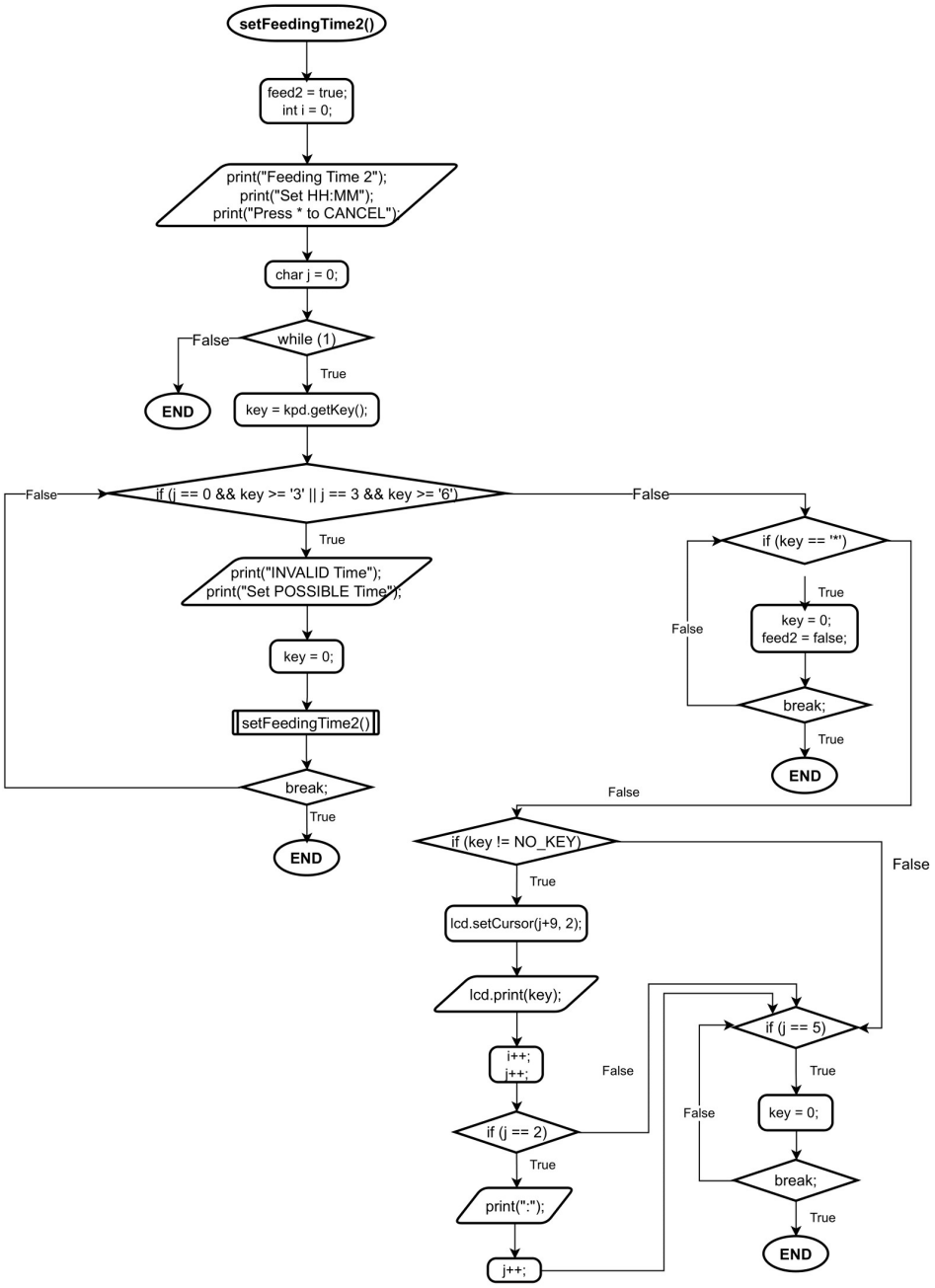

Fig. 4.2.5: Flowchart of void setFeedingTime2() Function.

4.2.6 setFeedingTime3() Function

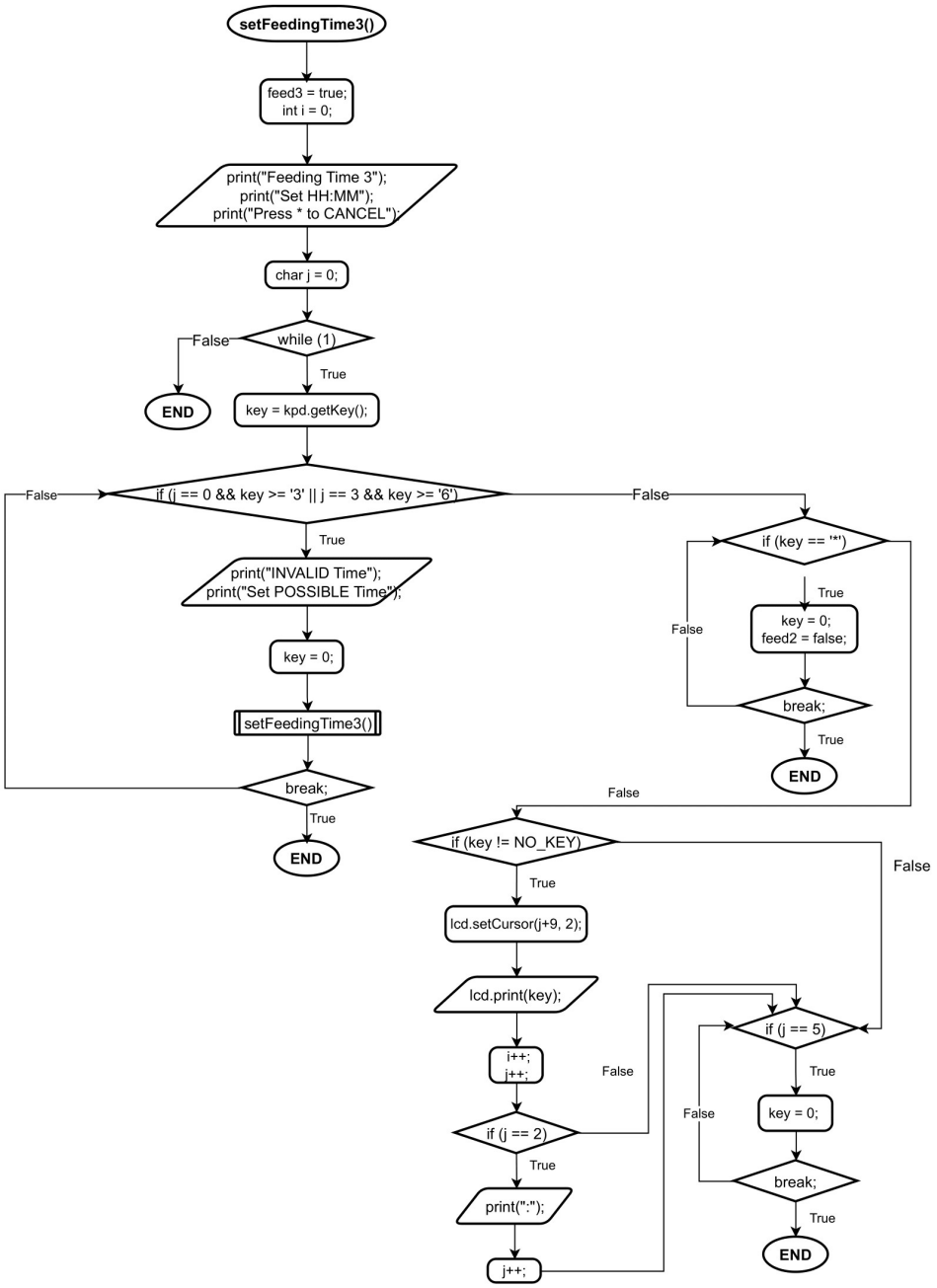

Fig. 4.2.6: Flowchart of void setFeedingTime3() Function.

4.2.7 setFeedingPortion() Function

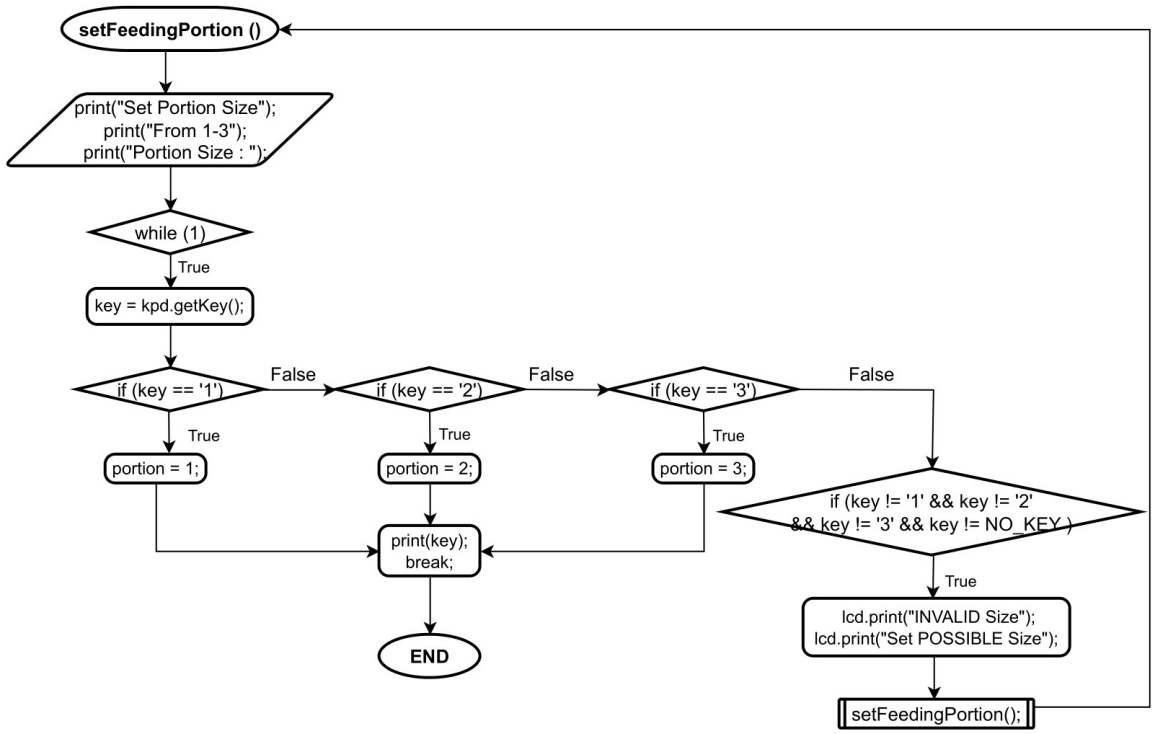

Fig. 4.2.7: Flowchart of void setFeedingPortion() Function.

4.2.8 portionnotset() Function

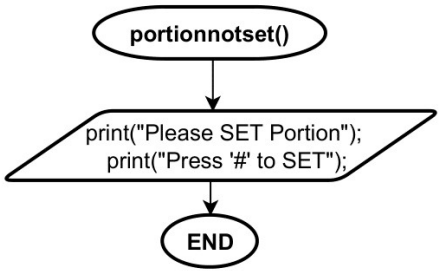

Fig. 4.2.8: Flowchart of void portionnotset() Function.

4.2.9 portion1() Function

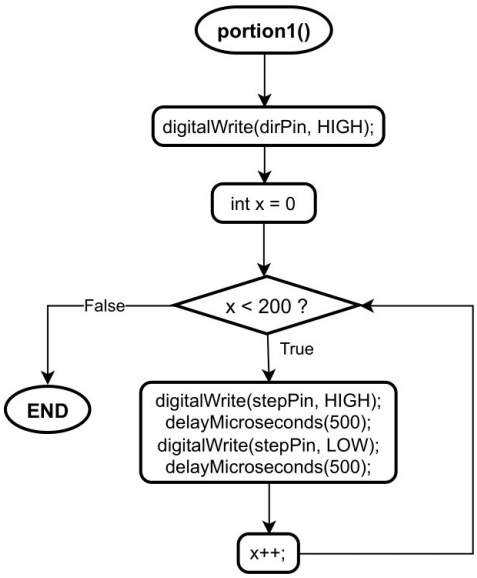

Fig. 4.2.9: Flowchart of void portion1() Function.

4.2.10 portion2() Function

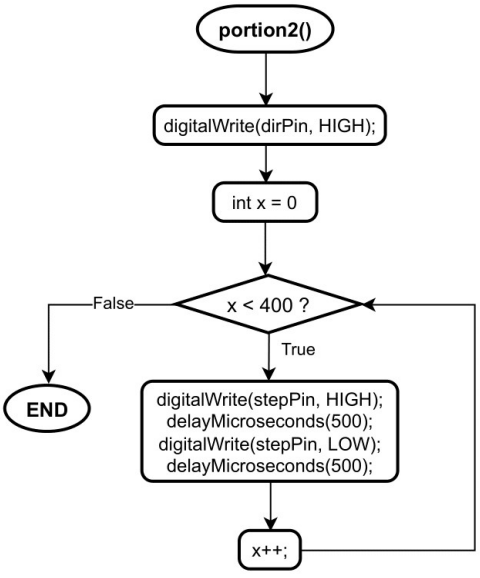

Fig. 4.2.10: Flowchart of void portion2() Function.

4.2.11 portion3() Function

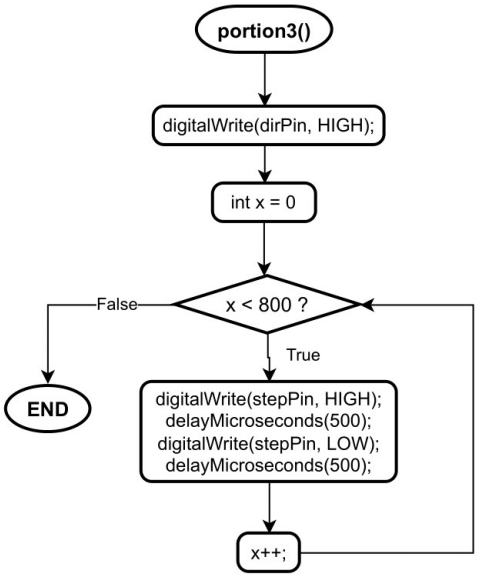

Fig. 4.2.11: Flowchart of void portion3() Function.

4.3 Schematic diagram

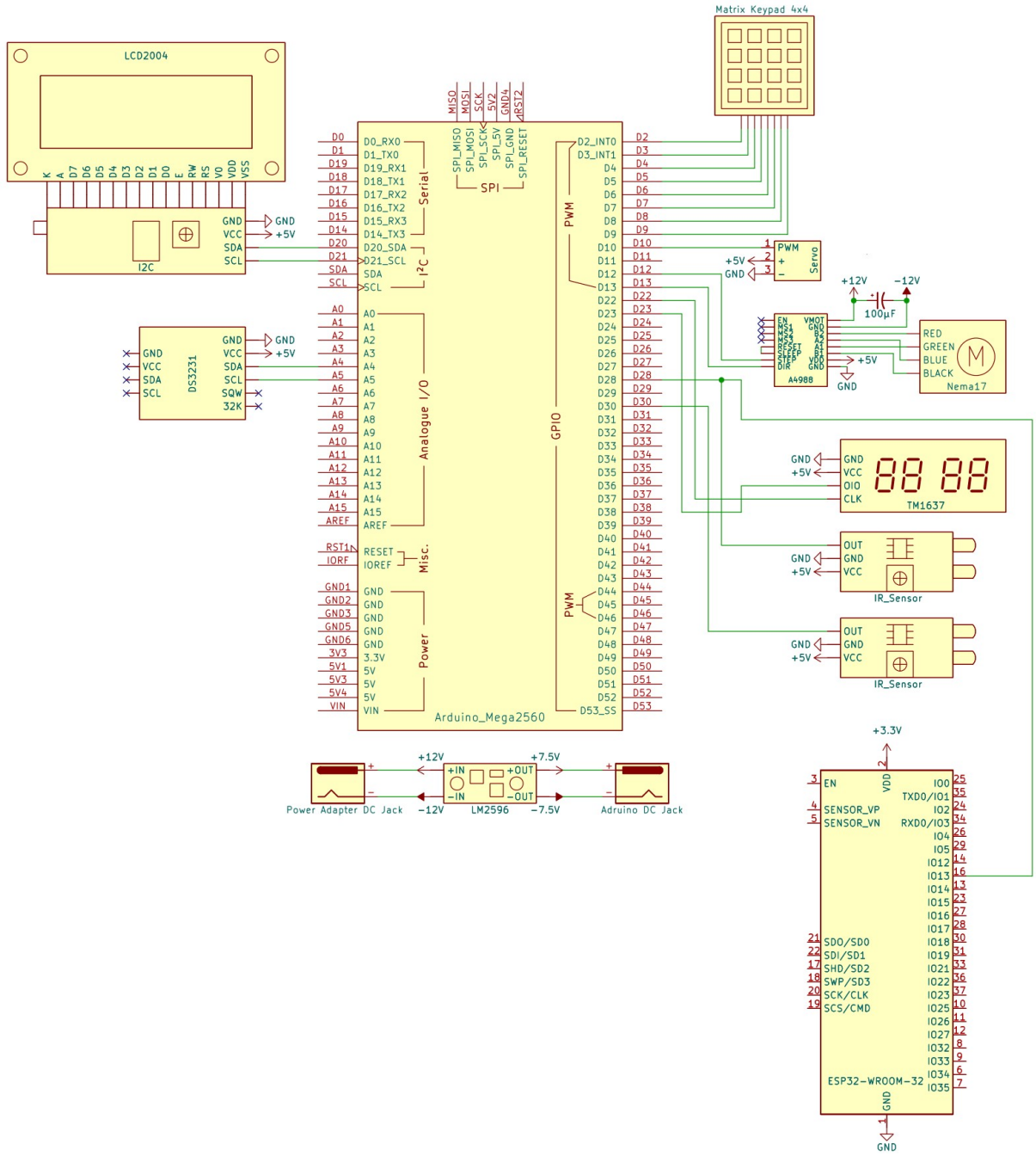

Fig. 4.3: Schematic Diagram.

5 Software Designs

5.1 Source Code

The source code of both Arduino and ESP32 can be accessed via the Github repository "@munpooxholic/PetFeeder_SP2_2-2022" under folder "<> Code"

Fig. 5.1: "@munpooxholic/PetFeeder_SP2_2-2022".
https://github.com/munpooxholic/PetFeeder_SP2_2-2022

5.2 Arduino Source code

Code is uploaded to the Arduino IDE in C++ language.

5.2.1 Libraries

1. The DS3231.h library implements the features to read and write the current time. This library is used along with the RTC DS3231 module.

2. The Keypad.h library implements the features to read and apply different operations from the data from the keypad inputs. This library is used along with the 4*4 matrix keypad membrane.

3. The TM1637.h library implements the features to control the TM1637 seven segment display module and. The system uses the data from the real time clock to display the time. This library is used along with the TM1637 seven segment display module.

4. The LiquidCrystal_I2C.h library implements the features to control LCD screen which is connect with I2C. This library is used along with the LCD2004 with the I2C serial interface module.

5. The Wire.h library implements the features to communicate with the I2C module.

```
1 #include <DS3231.h>
2 #include <Keypad.h>
3 #include <TM1637.h>
4 #include "LiquidCrystal_I2C.h"
5 #include "Wire.h"
```

Listing 5.2.1: Libraries Used in the system.

5.2.2 Main Function : void loop()

The system starts by displaying "Starting, Please Wait..." text on the LCD screen. An int variable "start" is declared as a counter. An "if (start == 0)" statement ends with "start++;". Thus the text will be shown on the screen only one time.

```
1 void loop()
2 {
3   if (start == 0)    //start == 0
4   {
5     lcd.print("Starting");
6     lcd.print("Please Wait...");
7     start++;
8   }
```

Listing 5.2.2.a: Source Code of the void loop() of the Arduino IDE Part a.

The next "else" statement will be operated only if "start != 0". This contains another "if (lowlvlstatus == 1)" sub-statement. This sub-statement verifies the status from the IR sensor that checks if the food in the container is empty or not. If the food container is empty, the system displays "Food Supply LOW, Refill the Food..." text until the food container becomes full again.

```
9   else  //start != 0
10  {
11    settimebutton = kpd.getKey();
12    IRstatus = digitalRead(IRSensor);
13    lowlvlstatus = digitalRead(lowlvlsensor);
14
15    if (lowlvlstatus == 1)  //Food container is EMPTY
16    {
17      lcd.print("Food Supply LOW");
18      lcd.print("Refill the Food...");
19    }
```

Listing 5.2.2.b: Source Code of the void loop() of the Arduino IDE Part b.

The next "else" statement will be operated under a condition "lowlvlstatus != 1", which means the food container is full. This statement consists of the following conditions :

1. If the 'A', 'B', or 'C' key is pressed while the "portion size == 0", the system goes to the "portionnotset()" function.
2. If the 'A', 'B', or 'C' key while the "portion size !=0", the system goes to the "setFeedingTime1()", "setFeedingTime2()", or "setFeedingTime3()" function according to the portion size selected.
3. If the 'D' key is pressed while the "IRstatus == 0" while the "portion size == 0", the system goes to the "portionnotset()" function.
4. If the 'D' key is pressed while the "IRstatus == 0" and the "portion size != 0", the system shows the "Food Tray FULL, CANNOT Dispense" text.
5. If the 'D' key is pressed while the "IRstatus != 0" and the "portion size == 0", the system goes to the "portionnotset()" function.
6. If the 'D' key is pressed with "IRstatus != 0" and the "portion size != 0", the system goes to the "portion1()", "portion2()", or "portion3()" function according to the portion size selected.
7. If the '#' key is pressed, the system goes to "setFeedingPortion()" function.

```
20      else  //Food container is FULL
21        if (settimebutton == 'A')
22        {
23          if (portion == 0)
24          {portionnotset();}
25          else
26          {setFeedingTime1();}
27        }
28
29        if (settimebutton == 'B')
30        {
31          if (portion == 0)
32          {portionnotset();}
33          else
34          {setFeedingTime2();}
35        }
36
37        if (settimebutton == 'C')
38        {
39          if (portion == 0)
40          {portionnotset();}
41          else
42          {setFeedingTime3();}
43        }
44
45        if (settimebutton == 'D')
46        {
47          if (IRstatus == 0)  //Food tray is FULL
48          {
```

```
49          if (portion == 0)
50          {portionnotset();}
51          else
52          {
53          lcd.print("Food Tray FULL");
54          lcd.print("CANNOT Dispense");
55          }
56        }
57
58        else  //Food tray is empty
59        {
60          if (portion == 0)
61          {portionnotset();}
62
63          if (portion == 1)
64          {portion1();}
65
66          if (portion == 2)
67          {portion2();}
68
69          if (portion == 3)
70          {portion3();}
71        }
72      }
73
74      if (settimebutton == '#')
75      setFeedingPortion();
```

Listing 5.2.2.c: Source Code of the void loop() of the Arduino IDE Part c.

The next statement is to store the numbers whenever the user inputs the numbers to set each timer. These stored numbers will be compared with the time from real time clock afterwards.

```
76      if (feed1 == true)
77      {
78        lcd.print(r[0]);
79        lcd.print(r[1]);
80        lcd.print(":");
81        lcd.print(r[2]);
82        lcd.print(r[3]);
83      }
84
85      if (feed2 == true)
86      {
87        lcd.print(f[0]);
88        lcd.print(f[1]);
89        lcd.print(":");
90        lcd.print(f[2]);
91        lcd.print(f[3]);
92      }
93
94      if (feed3 == true)
95      {
96        lcd.print(v[0]);
97        lcd.print(v[1]);
98        lcd.print(":");
99        lcd.print(v[2]);
100       lcd.print(v[3]);
101     }
```

Listing 5.2.2.d: Source Code of the void loop() of the Arduino IDE Part d.

Stored numbers from the previous step are compared with the number from the RTC DS3231. If the time entered from the user is the same as real time clock, the food will be dispensed. These functions also check if the food tray is full or not. If the food tray is full, the system displays "Food Tray FULL, CANNOT Dispense". If the food tray is empty, the system dispenses the food according to the portion size set by the user.

```
102     if (t1 == r[0] && t2 == r[1] && t3 == r[2] && t4 == r[3] && t5 < 1
103        && t6 < 3 && feed1 == true)  //Feeding time1
104     {
105       if (IRstatus == 0)  //Food tray FULL
106       {
107         lcd.print("Food Tray FULL");
108         lcd.print("CANNOT Dispense");
109       }
110       else  //Food tray EMPTY
111       {
112         if (portion == 1) //Portion size = 1
113         {
114           lcd.print("Feeding Time 1");
115           lcd.print("Food is coming!");
116           portion1();
117         }
118         if (portion == 2) //Portion size = 2
119         {
120           lcd.print("Feeding Time 1");
121           lcd.print("Food is coming!");
122           portion2();
123         }
124         if (portion == 3) //Portion size = 3
125         {
126           lcd.print("Feeding Time 1");
127           lcd.print("Food is coming!");
128           portion3();
129         }
130       }
131     }
132
133     if (t1 == f[0] && t2 == f[1] && t3 == f[2] && t4 == f[3] && t5 < 1
134        && t6 < 3 && feed2 == true)  //Feeding time2
135     {
136       if (IRstatus == 0)  //Food tray FULL
137       {
138         lcd.print("Food Tray FULL");
139         lcd.print("CANNOT Dispense");
140       }
141       else  //Food tray EMPTY
142       {
143         if (portion == 1) //Portion size = 1
144         {
145           lcd.print("Feeding Time 2");
146           lcd.print("Food is coming!");
147           portion1();
148         }
149         if (portion == 2) //Portion size = 2
150         {
151           lcd.print("Feeding Time 2");
152           lcd.print("Food is coming!");
153           portion2();
```

```
154           }
155           if (portion == 3) //Portion size = 3
156           {
157             lcd.print("Feeding Time 2");
158             lcd.print("Food is coming!");
159             portion3();
160           }
161         }
162       }
163
164       if (t1 == v[0] && t2 == v[1] && t3 == v[2] && t4 == v[3] && t5 < 1
165          && t6 < 3 && feed3 == true)  //Feeding time3
166       {
167         if (IRstatus == 0)  //Food tray FULL
168         {
169           lcd.print("Food Tray FULL");
170           lcd.print("CANNOT Dispense");
171         }
172         else  //Food tray EMPTY
173         {
174           if (portion == 1) //Portion size = 1
175           {
176             lcd.print("Feeding Time 3");
177             lcd.print("Food is coming!");
178             portion1();
179           }
180           if (portion == 2) //Portion size = 2
181           {
182             lcd.print("Feeding Time 3");
183             lcd.print("Food is coming!");
184             portion2();
185           }
186           if (portion == 3) //Portion size = 3
187           {
188             lcd.print("Feeding Time 3");
189             lcd.print("Food is coming!");
190             portion3();
191           }
192         }
193       }
194     }
195 }
```

Listing 5.2.2.e: Source Code of the void loop() of the Arduino IDE Part e.

5.2.3 Sub Function : void setFeedingPortion()

This function lets the user set the portion size from 1-3 by not allowing invalid input. If the time entered by the user is a valid number, the system prints "key" on the LCD screen. If the time entered by the user is an invalid number, the system goes back to the "setFeedingPortion()" function.

```
void setFeedingPortion()
{
  lcd.print("Set Portion Size");
  lcd.print("From 1-3");
  lcd.print("Portion Size : ");
  while (1)
  {
    key = kpd.getKey();

    if (key == '1')    //Portion size = 1
    {
      portion = 1;
      lcd.print(key);
      break;
    }
    if (key == '2')    //Portion size = 2
    {
      portion = 2;
      lcd.print(key);
      break;
    }
    if (key == '3')    //Portion size = 3
    {
      portion = 3;
      lcd.print(key);
      break;
    }
    if (key != '1' && key != '2' && key != '3' && key != NO_KEY )
    {
      lcd.print("INVALID Size");
      lcd.print("Set POSSIBLE Size");
      setFeedingPortion();
    }
  }
}
```

Listing 5.2.3: Source Code of the void setFeedingPortion() of the Arduino IDE.

5.2.4 Sub Function : void portionnotset()

If an invalid number of the portion size is entered from the main function or the "void loop()", the system goes to the "portionnotset()" function to let the user inputs the valid portion size.

```
  void portionnotset()
{
  lcd.print("Please SET Portion");
  lcd.print("Press '#' to SET");
}
```

Listing 5.2.4: Source Code of the void portionnotset() of the Arduino IDE.

5.2.5 Sub Function : void portion()

Every time the food is dispensed, the system goes to the "void portion()" function. There are 3 sets of the "void portion()" functions in the system, which are including:

1. void portion1() : The direction of rotation of the motor is set to "HIGH", the motor rotates up to "200" pulses and the speed of the rotation is defined according to the delayed time.

2. void portion2() : The direction of rotation of the motor is set to "HIGH", the motor rotates up to "400" pulses and the speed of the rotation is defined according to the delayed time.

3. void portion3() : The function to control the stepper motor is optimized differently compared to the portion1 and portion2 due to the problem of the motor turns into the wrong direction under more load. The direction of rotation of the motor is set to both "HIGH" and "LOW", and the motor rotates up to "100", "400", "100", "400", and "100" pulses accordingly.

```
void portion1() //Portion size = 1
{
  digitalWrite(dirPin, HIGH);
  for (int x = 0; x < 200; x++)
  {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(500);
  }
```

Listing 5.2.5.a: Source Code of the void portion1() of the Arduino IDE.

```
void portion2() //Portion size = 2
  {
    digitalWrite(dirPin, HIGH);
    for (int x = 0; x < 400; x++)
    {
      digitalWrite(stepPin, HIGH);
      delayMicroseconds(500);
      digitalWrite(stepPin, LOW);
      delayMicroseconds(500);
    }
  }
```

Listing 5.2.5.b: Source Code of the void portion2() of the Arduino IDE.

```
void portion3() //Portion size = 3
{
  for (int x = 0; x < 100; x++)
  {
    digitalWrite(stepPin, LOW);
    delayMicroseconds(500);
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(500);
  }

  digitalWrite(dirPin, HIGH);
  for (int x = 0; x < 400; x++)
  {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(500);
  }

  digitalWrite(dirPin, LOW);
  for (int x = 0; x < 100; x++)
  {
    digitalWrite(stepPin, LOW);
    delayMicroseconds(500);
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(500);
  }

  digitalWrite(dirPin, HIGH);
  for (int x = 0; x < 400; x++)
  {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(500);
  }

  digitalWrite(dirPin, LOW);
  for (int x = 0; x < 100; x++)
  {
    digitalWrite(stepPin, LOW);
    delayMicroseconds(500);
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(500);
  }
}
```

Listing 5.2.5.c: Source Code of the void portion3() of the Arduino IDE.

5.2.6 Sub Function : void setFeedingTime()

There are three sets of this function in the system, which are the "void setFeedingTime1()", void "setFeedingTime2()", and "void setFeedingTime3()".

These functions let the user type the time of each timer in the form of HH:MM. The function begins by setting the cursor at j+9 position. after the number is input then "j++". The numbers entered by the user are be saved to be compared with the time from the real time clock afterwards. The system stops asking more input when "j == 5" which means the user has already entered the time in the form of HH:MM. To delete the current timer, the '*' key must be pressed. Additionally, the user will not be allowed to input an invalid number.

```
void setFeedingTime()
{
  feed1 = true;
  int i = 0; char j = 0;
  lcd.print("Feeding Time");
  lcd.print("Set HH:MM");
  lcd.print("Press * to CANCEL");
    while (1)
    {
      key = kpd.getKey();
      if (j == 0 && key >= '3' || j == 3 && key >= '6') //Invalid time
      {
        lcd.print("INVALID Time");
        lcd.print("Set POSSIBLE Time");
        key = 0;  setFeedingTime();
        break;
      }

      if (key == '*')       //Delete time
      {
        key = 0;  feed1 = false;
        break;
      }

      if (key != NO_KEY)  //Valid time
      {
        lcd.setCursor(j+9, 2);
        lcd.print(key);
        r[i] = key - 48;
        i++; j++;
          if (j == 2)
          {
            lcd.print(":");
            j++;
          }
      }

      if (j == 5)
      {
        key = 0;
        break;
      }
    }
}
```

Listing 5.2.6: Source Code of the void setFeedingTime() of the Arduino IDE.

5.3 ESP32

Code is uploaded to the ESP32 in C++ language.

5.3.1 Setup

To use this WIFI function, the user has to generate Line token and put it in the source code. The WIFI name password must also be entered. Mobile hotspot can also be used, but if the user uses Apple Iphone 12 or later, the "Maximize compatibility" in the personal hotspot tab has to be turned on to reduce the WIFI to be 2.4GHz instead of 5GHz which cannot be used with ESP32.

```
#define WIFI_SSID "iphone13 promax"
#define WIFI_PASSWORD "123456789"
#define LINE_TOKEN_IR "X08U4IOm5YZKKPvcF0CXm66oizEoGEdSbC5OJB6weZ5"
String message1 = "Food Tray FULL. Please check if your pet has
                   finished the food...";
```

Listing 5.3.1: Source Code of the Setup of the ESP32.

5.3.2 Main Function : void loop()

This function allows the system to send the Line notification message to the user's mobile phone whenever the IR sensor detects a signal. This IR sensor is the one placed on the food tray to check if the food tray is full.

```
void loop()
 {
  time1 = millis();
  if ((digitalRead(irPin) == HIGH) && (WiFi.status() == WL_CONNECTED))
  {
    while (digitalRead(irPin) == HIGH)
    delay(100);
    Line_Notify1(message1);
  }
  delay(50);
}
```

Listing 5.3.2: Source Code of the void loop() of the ESP32.

5.3.3 Line Notification

Every time the IR sensor detects the presence of food in the food tray, the signal from the IR sensor is sent to ESP32 which allows the Line Notify to send the message to the user as long as both the ESP32 and the user's mobile phone are connected to the same WIFI or mobile hotspot. The message sent to the user can be changed by optimizing the code in the ESP32.

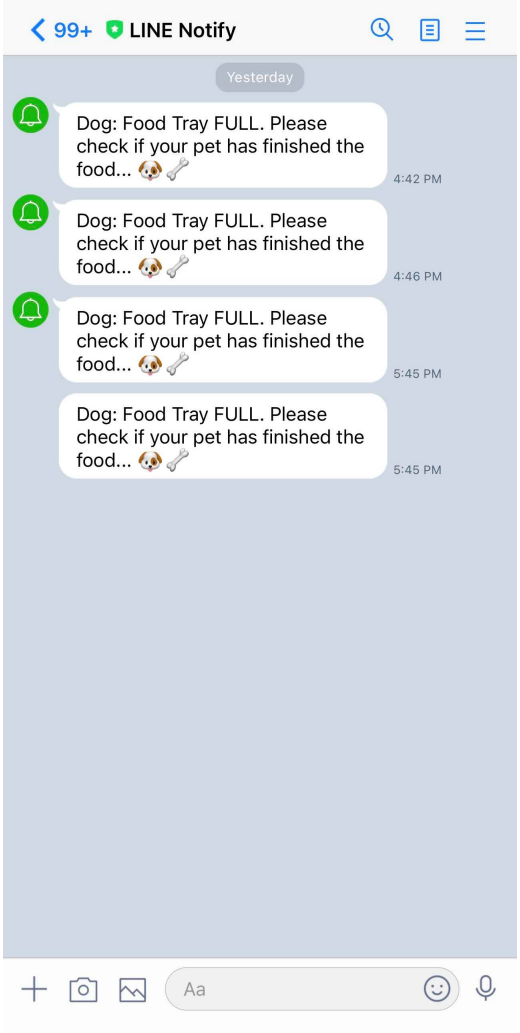

Fig. 5.3.3: Line Notification from ESP32.

6 Hardware Designs

6.1 3D Design

Both the enclosure and auger is designed in Fusion 360, developed by Autodesk, Fusion 360 provided us with a comprehensive set of tools and features that enabled the creation of complex and detailed 3D models.

6.1.1 Auger Design

An auger is used to dispense dry food pellets in a controlled manner. The auger rotates and moves food from the container to the dispensing area. This system ensures that a consistent amount of food is dispensed at each feeding, reducing the risk of overeating or underfeeding. Using an auger helps maintain the freshness of the food as well, since it dispenses the food directly from the storage container and minimizes exposure to air and moisture. This is especially important for dry pet food, which can spoil or become rancid if exposed to the elements for an extended period of time. In addition, an auger can help prevent food jams and clogs in the dispensing mechanism, ensuring that food is dispensed smoothly and continuously [19].

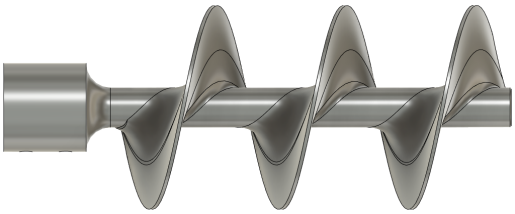

Fig. 6.1.1.a: Side View.

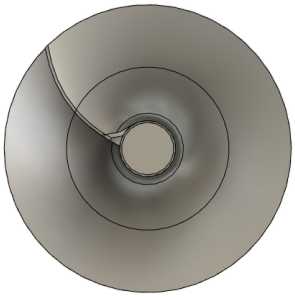

Fig. 6.1.1.b: Top View.

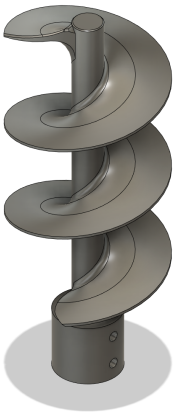

Fig. 6.1.1.c: Isometric View.

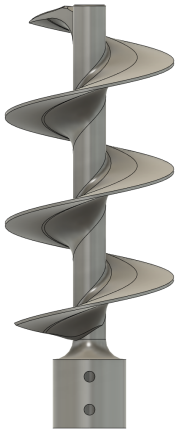

Fig. 6.1.1.d: Front View.

6.1.2 Enclosure Design

The food container is set at the top of the dispenser and the food bowl below the tube over the tray holder. All PCBs and controllers are placed inside the enclosure. All displayed modules, input module and sensors are fit in each hole designed for the enclosure. The back of the machine contains 2 holes for the ON/OFF switch and a DC power connector for the power adapter. Additionally, there are holes for ventilation at the side and the back of the machine.

Fig. 6.1.2.a: Top View.

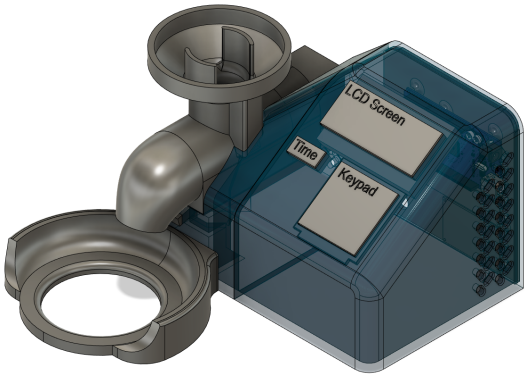

Fig. 6.1.2.b: Isometric View.

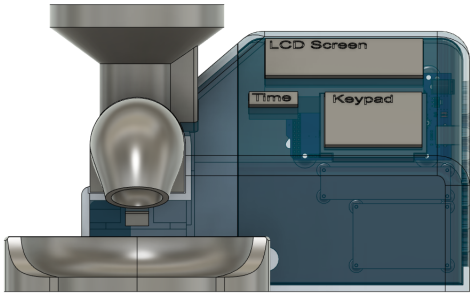

Fig. 6.1.2.c: Front View.

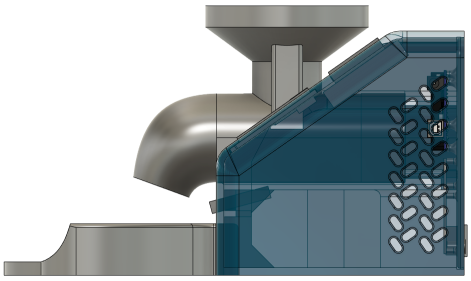

Fig. 6.1.2.d: Side View.

6.2 Assembly

The design uploaded to the Ender V3 3D printer in a .stl format is then printed and assembled. This method allows for a custom-fit in the size and shape of the components, ensuring a secure and stable base for the feeder. It allows for the creation of intricate and detailed designs that can be difficult to achieve using traditional methods.

6.2.1 Soldering

All components are soldered together according to the circuit diagram.

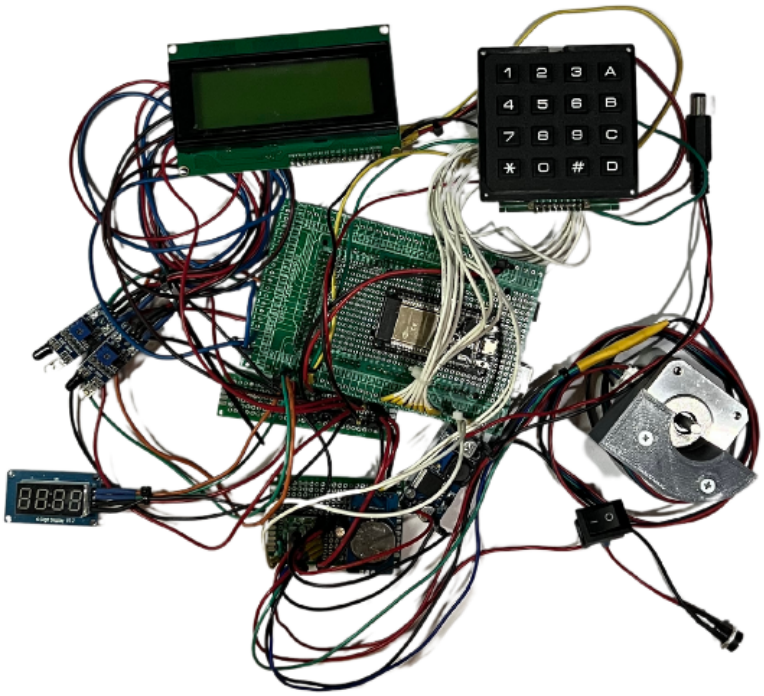

Fig. 6.2.1: Real Circuit Connection.

6.2.2 3D Printed Auger

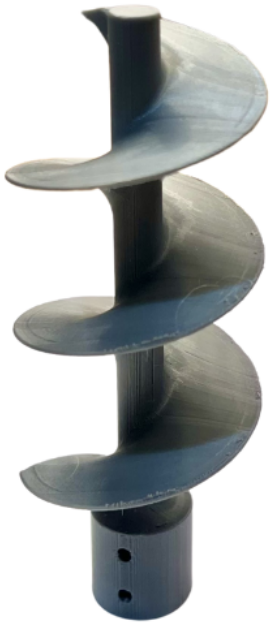

Fig. 6.2.2.a: Real 3D Printed Auger.

Fig. 6.2.2.b: Front View.

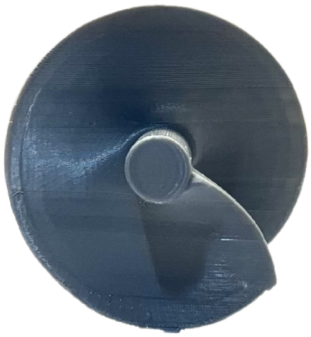

Fig. 6.2.2.c: Top View.

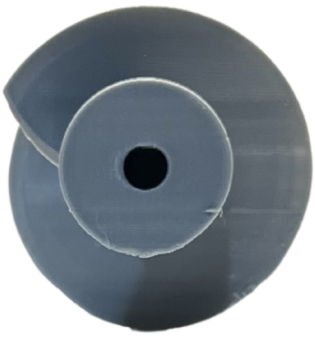

Fig. 6.2.2.d: Bottom View.

6.2.3 3D Printed Enclosure

Fig. 6.2.3.a: Left Side.

Fig. 6.2.3.b: Right Side.

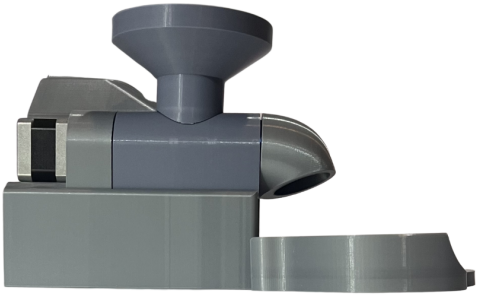

Fig. 6.2.3.c: Left Quarter View.

Fig. 6.2.3.d: Right Quarter View.

Fig. 6.2.3.c: Front View.

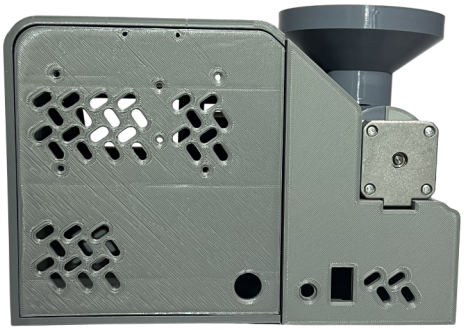

Fig. 6.2.3.d: Rear View.

6.2.4 Food Container

Fig. 6.2.4.a: Food Container.

Fig. 6.2.4.b: Front View.

Fig. 6.2.4.c: Top View.

Fig. 6.2.4.d: Bottom View.

6.2.5 Food Tray

Fig. 6.2.5.a: Food Tray.

Fig. 6.2.5.b: Front View.

Fig. 6.2.5.c: Top View.

Fig. 6.2.5.d: Bottom View.

6.2.6 Final Product

Fig. 6.2.6.a: Left Side.

Fig. 6.2.6.b: Right Side.

Fig. 6.2.6.c: Left Quarter View.

Fig. 6.2.6.d: Right Quarter View.

Fig. 6.2.6.c: Front View.

Fig. 6.2.6.d: Rear View.

7 User Manual

A two-page user manual is provided to the user to be more understanding about the way to use this machine properly.

Fig. 7.a: User Manual Page 1.

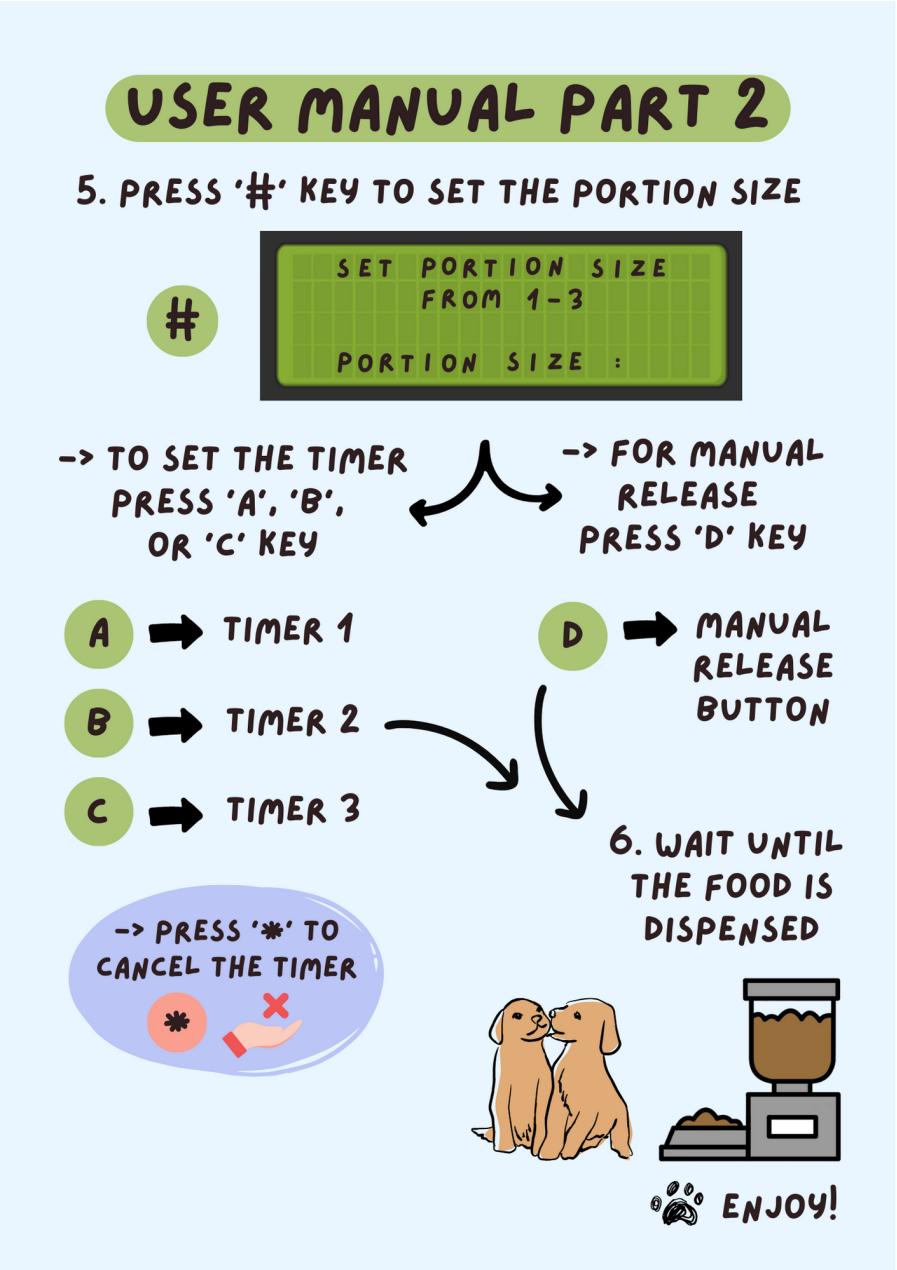

Fig. 7.b: User Manual Page 2.

8 Testing

8.0.1 Electrical Power Testing

An electrical power testing is conducted to ensure the safety of the electrical systems and components by identifying potential hazards such as loose connections, or overloading. This helps ensure that the electrical system in our project is functioning reliably and efficiently, reducing the risk of power outages, system failures, and other disruptions [20].

Table 8.0.1: Electrical Power Testing Table.

	Stepper Motor OFF	**Stepper Motor ON**
1 Portion	V = 12.00V	V = 12.00V
	I = 0.694A	I = 0.573A
	P = 8.328W	P = 6.876W
2 Portion	V = 12.00V	V = 12.00V
	I = 0.693A	I = 0.449A
	P = 8.316W	P = 5.388W
3 Portion	V = 12.00V	V = 12.00V
	I = 0.685A	I = 0.452A
	P = 8.220W	P = 5.424W

8.0.2 Weight per portion Testing

To test if the auger mechanism is accurate or not. The Weight per portion is tested. This serves the purpose of ensuring that the feeder dispenses the correct amount of food for each serving. This test involves measuring the weight of a single portion of dog food that is dispensed by the feeder and comparing it to the desired weight per portion.

Table 8.0.2: Weight Per Portion Testing Table.

	No.	**Weight(grams)**
1 Portion	1.	23
	2.	21
	3.	28
Average		24
2 Portion	1.	54
	2.	47
	3.	46
Average		49
3 Portion	1.	99
	2.	109
	3.	95
Average		101

9 Challenges

9.1 Heating Issues

The Arduino gets too hot when the stepper motor uses the same power supply as the Arduino, because the stepper motor needs 12V to be operated and the Arduino uses only 5V. To solve this problem we added an LM2596n module or a buck converter to step down the voltage from 12V DC to 7.5V DC. Additionally, heat sinks are attached to the LM2596 module and the Arduino.

9.2 Dispensing System Jamming

The food pellets gets stuck before entering the tube, because the amount of food pellets are going into one tiny hole in a substantial amount. This problem is solved by the funnel shape being optimized to be able to handle these amounts and the inlet size of the tube being reduced.

9.3 WIFI Connection Issues

The ESP32 is not able to connect to 5G wifi, it has to be connected to a 2.4G frequency only. Phones such as the Iphone 12 or newer versions are emitting 5G wifi as a personal hotspot, which cannot be used with the ESP32. To use the 2.4G wifi hotspot, the user has to turn on "Maximize compatibility" in the personal hotspot tab, this will set the wifi of that phone to be 2.4G which provides slower speed than 5G. The same goes to wifi modules at home. The user must choose the 2.4G connection from there wifi router in order to connect to ESP32 to their wifi module.

10 Applications

Our automatic pet feeder has a design that utilizes the auger for dispensing food. This design can be used to fill or dispense any type of substance that has shapes similar to grains or pellets. Below are five applications of our pet feeder.

10.1 Auger powder filling machine

With some adjustments to the design and enlarging the size, this pet feeder can be modified into an auger powder filling machine. This is a device that is used to fill any powdered products into pillow cases, glass jars, rice bags, etc.

10.2 Medicine dispenser

Keeping the same exact design, the container can be filled with medicine and be used in hospitals to dispense pills in a time efficient matter. This can be placed in the medicine room of the hospital, drug stores and can also be placed in the patient's room since this dispenser has a Timer Mode which dispenses at any time set by the user.

10.3 Seed dispenser

Since seeds are a similar shape to dog food, the food pellets in the container can be replaced with seeds. This can be used by farmers to get an accurate and constant weight of their seeds to be used in plantations of their crops.

10.4 Candy machine

Similar reasons to why we could replace the food pellets with seeds, the contents in the container can also be replaced with candies. Adding a system that accepts coins or other types of payment to this feeder can make this become a candy machine. This can be placed beside vending machines and are for people who prefer a small serving of candies.

10.5 Coffee bean dispenser

Replacing the contents of the container with coffee beans, this device can be used as a coffee bean dispenser. Baristas can get an accurate portion of coffee bean required to brew any type of coffee. This machine can be used in Cafés to help baristas brew coffee consistently due to the beans being dispensed accurately in terms of weight.

11 Conclusion

In conclusion, an automatic pet feeder is a valuable addition to the life of pet owners. This innovative device is designed to dispense food to pets that weigh less than or equal to 5 kg, and its auger system ensures that food is dispensed in a consistent and controlled manner. Additionally, the pet feeder notifies the user when the food in the container is running low, ensuring that food is always readily available for the pet. Lastly, the pet feeder alerts the user to check the food bowl, ensuring that the pet is well-fed and healthy. With its innovative design and reliable performance, an automatic pet feeder is an excellent investment for pet owners looking to provide the best care for their furry friends.

Acknowledgment

We would like to thank the Assumption University Engineering committee for giving us the chance and inspiration to create this Automatic Pet Feeder and Asst. Prof. Dr. Narong Aphiratsakun for guiding us through our journey in to completing the Automatic Pet Feeder. Finally, we would also like to thank A. Warongkidh Ganchnasopa and Mr.Utthaiwut Nontsungnoen for advising us about the electrical power testing.

References

[1] S. Koley , S. Srimani, D.Nandy , P. Pal, S. Biswas, and I.Sakar (2021, February), "Smart Pet Feeder. In Journal of Physics", Conference Series (Vol. 1797, No. 1, p. 012018). IOP Publishing.

[2] M. Manoj, "Automatic Pet Feeder.", International Journal of Advances in Science Engineering and Technology,(2015) ISSN, 2321-9009.

[3] M.Ibrahim, H. Zakaria, and E. Xian, "Pet food autofeeder by using Arduino". In IOP Conference Series: Materials Science and Engineering (Vol. 670, No. 1, p. 012069). November 2019 , IOP Publishing.

[4] I. Kersbergen, J.Alexander German, C. Westgarth, and E. Robinsona, “Portion size and meal consumption in domesticated dogs: An experimental study” 2019 May 15; 204: 174–179, PMCID: PMC6488012 PMID: 30817974,
doi: 10.1016/j.physbeh.2019.02.034

[5] Robotshop, Arduino Mega 2560. Retrieved November, 2560 (2011).

[6] Espressif Systems, ESP32 Series Datasheet. Espressif Systems, 1–65 (2021).

[7] D. A. Devi and N. S. Rani, "Design and Implementation of custom IP for Real Time Clock on Reconfigurable Device," 2019 Third International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 2019, pp. 414-418, doi: 10.1109/ICISC44355.2019.9036428.

[8] D. Nedelkovski, How To Control a Stepper Motor with A4988 Driver and Arduino. 2015, 1 (2015).

[9] E. Maulana, C. S. Nugroho and A. B. Dianisma, "Animal Presence Detection for Elephants and Extruding Method Based on Bee Frequency," 2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), Batu, Indonesia, 2018, pp. 119-122, doi: 10.1109/EECCIS.2018.8692992.

[10] F. A. Silaban, S. Budiyanto, W. K. Raharja, Stepper motor movement design based on FPGA. International Journal of Electrical and Computer Engineering. 10, 151–159 (2020).

[11] A. Gundogdu, R. Celikel,NARMA-L2 controller for stepper motor used in single link manipulator with low-speed-resonance damping,Engineering Science and Technology, an International Journal,Volume 24, Issue 2,2021,Pages 360-371,ISSN 2215-0986.

[12] Handson Technology, Handson Technology User Guide I2C Serial Interface 20x4 LCD Module. LCD 2004 I2C datasheet, 1–8 (2021).

[13] SHENZHEN EONE ELECTRONICS CO.,LTD, "Specification For LCD Module 2004 A"; ORISE Technology Co.,Ltd., Version 0.1, SEP 21, 2007.

[14] T. S. Lan, Y. Jiao, X. J. Dai, H. S. Chen, Z. Yang, Portable pulse meter with simple design. Sensors and Materials. 33, 4077–4085 (2021).

[15] M. O. Alsumady, Y. K. Alturk, A. Dagamseh, M. Tantawi, Controlling of dc-dc buck converters using microcontrollers. International Journal of Circuits, Systems and Signal Processing. 15, 197–202 (2021).

[16] S. Chandrasekaran, Interface 4x4 Matrix Keypad With Microcontroller - EmbedJournal. Embed Journal. 2013, 1–8 (2013).

[17] Arduino.cc, Arduino Uno Rev3. Arduino.Cc, 1 (2020).

[18] K. Widarsono, M. Jauhari, and A. L. Dzuhuri. "Relay Protection of Over Voltage, Under Voltage and Unbalance Voltage Magnitude Based on Visual Basic Using Arduino Mega." In Seminar MASTER PPNS, vol. 4, no. 1, pp. 39-48. 2019.

[19] G. Hetsroni, et al. "A uniform temperature heat sink for cooling of electronic devices." International Journal of Heat and Mass Transfer 45.16 (2002): 3275-3286.

[20] H. Mizutani, R. Ishikawa, and K. Honjo. "InGaAs MMIC SPST switch based on HPF/LPF switching concept with periodic structure." IEEE Transactions on Microwave Theory and Techniques 64.9 (2016): 2863-2870.

[21] B. Ravi, P. Pavan Kumar, and P. G. Kuppusamy. "Arduino Mega based PET feeding automation." IOSR Journal of Electronics and Communication Engineering 14, no. 4 (2019): 13-16.

[22] P. Gill ,"Electrical power equipment maintenance and testing.", CRC press; 2016 Dec 19.