Smart Pet Feeding System

Sumiya Tabussum Dristy
United International University
011181237

AHM Emramul Pasha
United International University
011181243

Raj Shekhar Karmaker
United International University
011193149

Kawsar Newaz Chowdhury
United International University
011201454

***Abstract*—This paper is an exploratory study to share information on raising knowledge on how to manage a pet feeding system. This study explains how humans' personal information might be compromised due to their daily behaviors. This project is associated with and eliminating inconsistent and manual pet feeding routines by introducing an automated feeding system controlled by a microprocessor.**

The automated pet feeding system developed utilizes a micro controller to control the scheduling and portions of pet food. It integrates a microprocessor to manage user-defined feeding times and portion sizes, ensuring accurate and timely dispensing while allowing remote adjustments via a dedicated app.

The system's performance in an automated pet feeding system, which is controlled by a micro controller and a microprocessor, shows precise and reliable feeding accuracy, delivering meals on schedule, and accommodating real-time tweaks for the convenience of pet owners.

I. Project Overview

The "Automated Pet Feeding System" is a cutting-edge project aimed at revolutionizing the way pets are cared for by introducing a smart and efficient feeding solution. This project leverages microprocessor technology to create a hands-free, customizable, and reliable feeding system for pets, offering numerous benefits to both pet owners and their furry companions.

The primary objective of this project is to design and develop an automated pet feeding system that simplifies the daily feeding routine for pet owners while ensuring the health and well-being of their pets. By utilizing a microprocessor-controlled mechanism, this system will accurately dispense appropriate portions of pet food at scheduled intervals, all while addressing safety concerns and promoting convenience.

In conclusion, the Automated Pet Feeding System project brings innovation to pet care by introducing a microprocessor-driven solution that provides convenience, accuracy, and improved pet health. This system represents a significant step forward in the domain of pet care technology, offering a smarter way to nurture our beloved animal companions.

II. Background

To take care of pets like cats and dogs, an automatic pet feeder system was created. The pet feeder system may provide food and water while also keeping an eye on the creature's movement. This gadget has several embedded parts that make it possible to administer water and feed meals without the need for human assistance. Unfortunately, they have mistreated and starved their pets because of their busy schedules and little free time at work. By giving their pets the proper food at the right time and keeping tabs on them via the designated application, folks can save time and energy by participating in this project. Because the device has an internet connection, it has been possible for the owner to monitor the pet's feeding on the appropriate Thing Talk cloud. One of the machine's innovative parts is the Arduino UNO, which is used in a bottle that may last for a week with electrical connections. Another is the server motor. This is designed to automatically provide food to pets based on the amount of food available and the time. As a result, the majority of pet feeders on the market are operated manually and do not even employ IoT techniques. This pet feeder may be used automatically, so owners won't have to worry about their dogs. Those who adore animals will adore choosing this kind of device for their pets at home. Customers who use this kind of device will also feel less anxious about leaving their dogs for brief periods, such as when going back to their hometown or starting a new job. Finally, it is a great way to recruit and retain top local users, particularly in the mechanical industry. This is very useful for pet owners, and it makes use of the most recent technology, which stimulates the Internet of Things and embedded systems since they may be used to upgrade local industrial pet care.

III. Benefits

The automated pet feeding system offers several benefits:

Convenience: Pet owners can maintain consistent feeding schedules even when they are not physically present.

Health: Portion control and balanced feeding contribute to the pet's overall health and well-being.

Peace of Mind: Owners can travel or work longer hours without worrying about their pet's feeding routine.

Data-Driven Insights: The system's data collection helps pet owners understand their pet's dietary habits better.

IV. Component List

Arduino uno
NodeMcu ESP8266
Ultrasonic Sensor
Digital Electronic Weighing Sensor, load-cell (scale 1 kg)
Servo Motor
Speaker and voice recording tool
Rtc module
Breadboard
Wires

- Arduino uno

Fig. 1. Arduino Uno

A microcontroller board called Arduino Uno is based on the ATmega328P (datasheet). It has a 16 MHz ceramic resonator (CSTCE16M0V53-R0), 6 analog inputs, 14 digital input/output pins (of which 6 can be used as PWM outputs), a USB port, a power jack, an ICSP header, and a reset button. It comes with everything required to support the micro controller; to get started, just use a USB cable to connect it to a computer, or an AC-to-DC adapter or battery to power it. You can experiment with your Uno without being overly concerned that you'll make a mistake; in the worst case, you can replace the chip for a few dollars and start over. **Specifications** Micro controller ATmega328P Operating Voltage 5V Input Voltage (recommended) 7-12V Input Voltage (limit) 6-20V Digital I/O Pins 14 (of which 6 provide PWM output) PWM Digital I/O Pins 6 Analog Input Pins 6 DC Current per I/O Pin 20 mA DC Current for 3.3V Pin 50 mA Flash Memory 32 KB (ATmega328P) of which 0.5 KB used by boot loader SRAM 2 KB (ATmega328P) EEPROM 1 KB (ATmega328P) Clock Speed 16 MHz $LED_BUILTIN13Length68.6mmWidth53.4mmWeight25g$

- Arduino uno

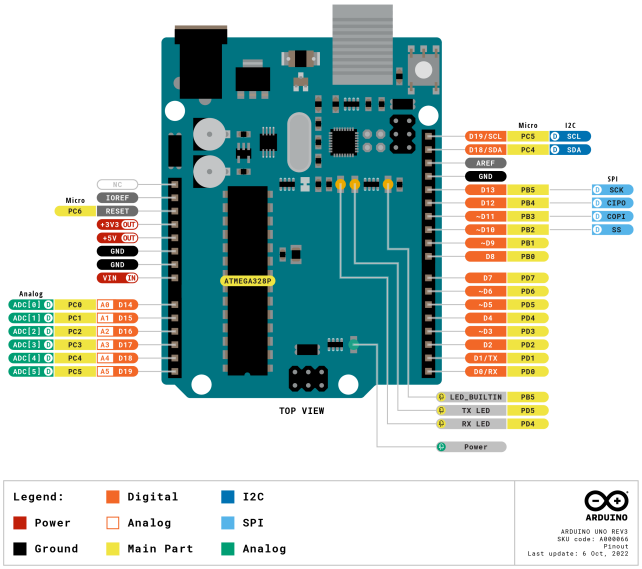

Fig. 2. Arduino Uno

– NodeMcu ESP8266

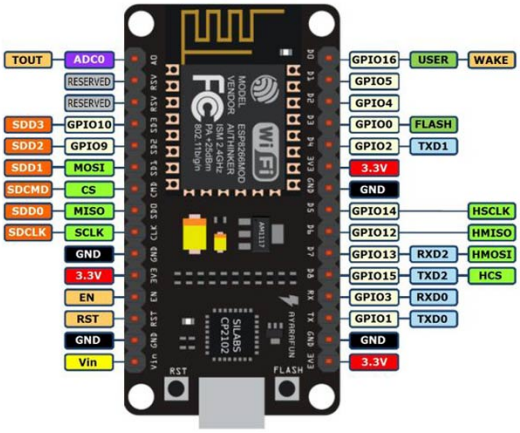

Fig. 3. NodeMcu ESP8266

NodeMCU is a development board and open-source Lua-based firmware that is specifically designed for Internet of Things (IoT) applications. It has hardware based on the ESP-12 module and firmware that runs on Espressif Systems' ESP8266 Wi-Fi SoC.

NodeMCU ESP8266 Specifications Features Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106 Operating Voltage: 3.3V Input Voltage: 7-12V Digital I/O Pins (DIO): 16 Analog Input Pins (ADC): 1 UARTs: 1 SPIs: 1 I2Cs: 1 Flash Memory: 4 MB SRAM: 64 KB Clock Speed: 80 MHz USB-TTL based on CP2102 is included onboard, Enabling Plug n

Play PCB Antenna Small Sized module to fit smartly inside your IoT projects

– Ultrasonic Sensor

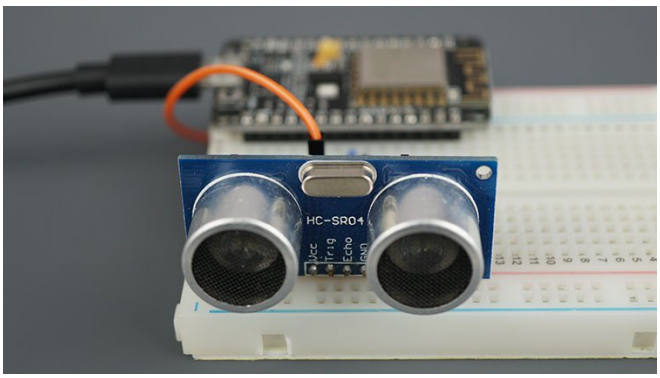

Fig. 4. Ultrasonic Sensor

According to the illustration above, the HC-SR04 Ultrasonic (US) sensor is a 4-pin module with the pin designations Vcc, Trigger, Echo, and Ground. This sensor is quite widespread and is used in many applications where it is important to sense objects or measure distance. On the front of the module, two projections that resemble eyeballs serve as the ultrasonic transmitter and receiver. The sensor uses a simple high school algorithm to function.

∗ · Digital Electronic Weighing Sensor, load-cell (scale 1 kg)

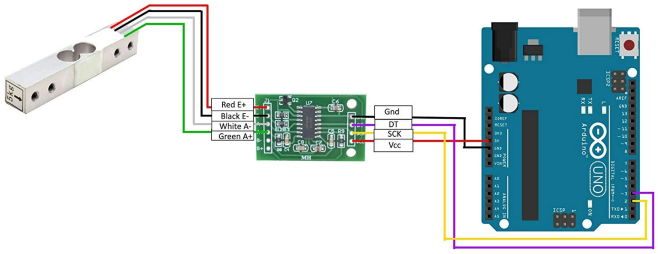

Fig. 5. Digital Electronic Weighing Sensor, load-cell (scale 1 kg)

This strain gauge-style load cell sensor for weighing can convert up to 1 kg of pressure (force) into an electrical signal. The electrical resistance that varies in reaction to and proportionate to the strain (such as pressure or force) applied to the bar may be measured by each load cell. It may be linked to the HX711 A/D Pressure Sensor via its four lead wires. It produces the output voltage according to the force changes over it and is simple to use with driving voltages of 5 to 10 volts.
General Specification Model YZC-131 Material Aluminium Dimensions in mm (LxWxH) 75 x 12.6 x 12.6 Weighing Range (Kg) 0 1 Rated Output 1.0 ± 0.1mV / V Non Linear Output ± 0.03Impedance () 1066 ± 10Output Impedance 1000 ± 10Insulation Resistance (M) 2000 Hysteresis 0.03Repeatability 0.03Creep (5 minutes) 0.05Zero Balance ± 0.1 mV / V Cable Length (cm) 18 Operating Temperature Range (°C) -20 to 65 Weight (gm) 26 Shipment Weight 0.08 kg Shipment Dimensions 12 × 5 × 4 cm

· Servo Motor

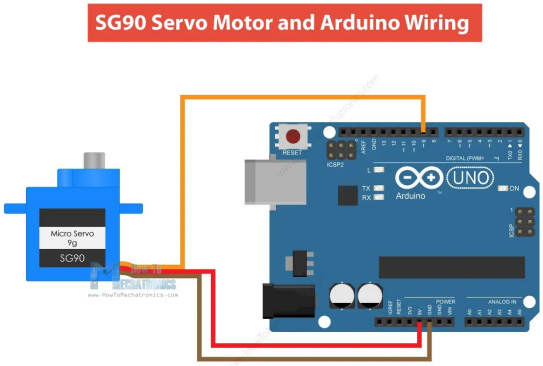

Fig. 6. Servo Motor

· A servo motor is a kind of motor that has extremely precise rotational capabilities. This type of motor often has a control circuit that gives feedback on the motor shaft's present location. This feedback enables the servo motors to rotate very precisely. A servo motor is used to rotate an item at predetermined angles or distances. It consists only of a straightforward motor that drives a servo mechanism. A motor is referred to as a DC servo motor if it is powered by a DC power source, and an AC servo motor if it is driven by an AC power source. The control wires offer PWM (Pulse with Modulation), which is used to drive the servo motor. There are three different pulse ranges: minimum, maximum, and repetition rate. A servo motor's neutral state allows it to turn 90 degrees in any direction. Every 20 milliseconds (ms), the servo motor expects

to receive a pulse; the pulse's duration determines how far the motor will revolve. For instance, a 1.5ms pulse will cause the motor to turn to the 90° position; a pulse less than 1.5ms will cause the shaft to move to 0°, while a pulse longer than 1.5ms will cause the servo to turn to 180°.
The duration of the pulse supplied to the servo motor's Control PIN determines the angle at which it rotates, according to the PWM (Pulse Width Modulation) concept. The basic components of a servo motor are a DC motor, a variable resistor (potentiometer), and some gears. Gears transform a DC motor's high speed force into torque. Knowing that WORK = FORCE X DISTANCE, a DC motor has a low force and a high distance (speed), but a servo has a high force and a low distance. To determine the angle and stop the DC motor at the necessary angle, the potentiometer is linked to the servo's output shaft.

· Servo Motor Rotation

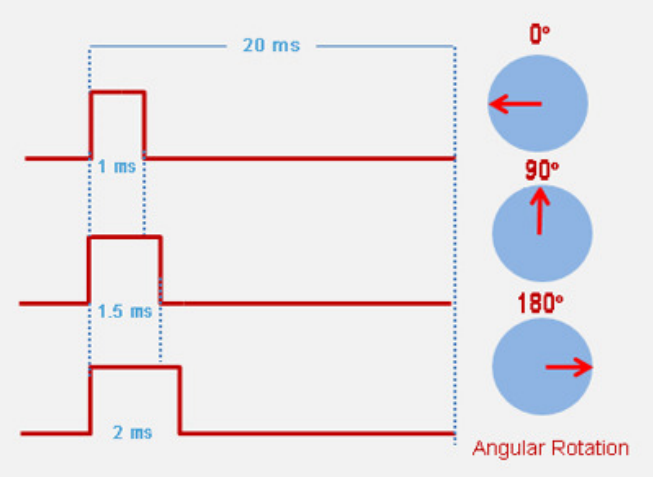

Fig. 7. Servo Motor Rotation

· Servo motors can rotate from 0 to 180 degrees, however depending on the manufacturing process, they can also rotate up to 210 degrees. Applying the appropriate width electrical pulse to its regulate pin will regulate the degree of rotation. Every 20 ms, the servo examines the pulse. The servo may rotate to 0 degrees with a pulse of 1 ms (1 millisecond) width, 90 degrees (the neutral position) with a 1.5 ms pulse, and 180 degrees with a 2 ms pulse.
All servo motors are powered by your +5V supply rails directly.

· Rtc module

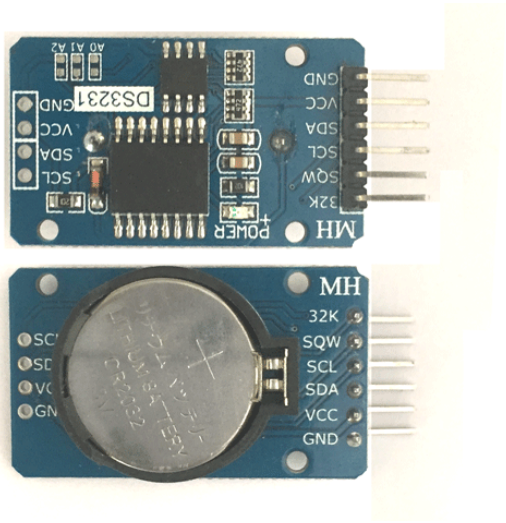

Fig. 8. Rtc module

A 10-bit temperature sensor with a precision of 0.25C and 32Kbit EEPROM are both included into the DS3231 RTC, which is a precise real-time clock module. A low-cost, incredibly precise I2C real-time clock (RTC) with an inbuilt temperature-compensated crystal oscillator (TCXO) and crystal is the DS3231 RTC module, or Precise Real-Time Clock Module. When the gadget's main power supply is interrupted, the device has a battery input and continues to retain precise time.
General Specification Operating Voltage (VDC) 2.7 5.5 Voltage Supply for RTC 2.2 V 5.5 V Accuracy ±2ppm from 0°C to +40°C. ±3.5ppm from -40°C to +85°C. Battery Holder 2032 Coin Battery. I2C interface Fast (400kHz) I2C Interface. EEPROM AT24C32 32Kbit Serial I2C. Time and Date Format Time: HH: MM: SS (12/24 hr). Date Format: YY-MM-DD-dd. Operating Temperature Range (°C) -40 to 85 Digital Temp Sensor Output 10 bit, ±3°C Accuracy and 0.25C resolution. Dimensions in mm (LxWxH) 38 x 22 x 14 Weight(gm) 8 Shipment Weight 0.012 kg Shipment Dimensions 12 × 8 × 5 cm

1) The VU pin of the NodeMCU is linked to the VCC pin of the ultrasonic sensor.
2) The NodeMCU's GND pin is linked to the GND pin of the ultrasonic sensor.

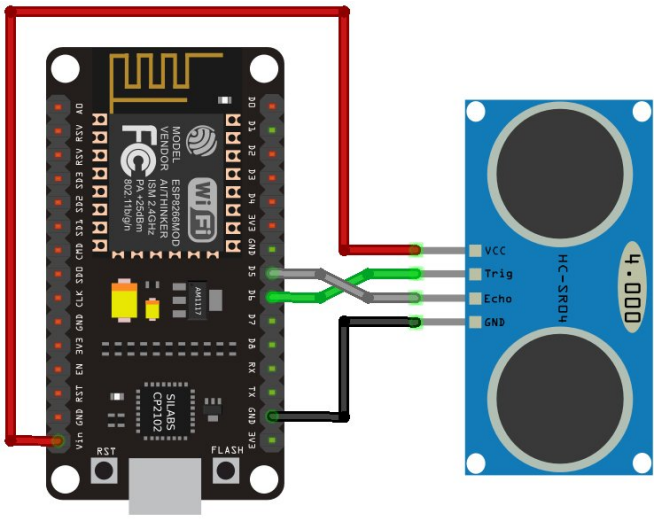

Fig. 9. NodeMcu ESP8266 to Ultrasonic Sensor

3) The NodeMCU's D5 pin is linked to the Ultrasonic sensor's TRIG pin.
4) The NodeMCU's D6 pin is linked to the Ultrasonic sensor's ECHO pin.

V. Features

1) Programmable Feeding Schedule Allow pet owners to set specific feeding times and portion sizes for their pets.
2) Portion Control Dispense precise amounts of food to prevent overfeeding or underfeeding.
3) Food Level Monitoring Implement a system to detect and display the remaining food level in the feeder.
4) Feeding History and Data Logging Log feeding times and portion sizes for tracking and analysis.
5) Customizable Alerts Send notifications or alerts to pet owners when the food level is low or when feeding is completed.

VI. Code

VII. Flowchart

```
weight+time+speaker+ultra + motor.ino
#include <Wire.h>
#include "RTClib.h"
#include "HX711.h"
#include <Servo.h>

#define DOUT_PIN 2
#define CLK_PIN 3

#define TRIG_PIN 4
#define ECHO_PIN 5
#define LED_PIN 6

#define P_E 8

Servo myservo;

HX711 scale;

RTC_Millis rtc;

int pos=0;
int speak=0;
Output
```

Fig. 10. Code1

```
weight+time+speaker+ultra + motor.ino
void setup () {
  Serial.begin(9600);
  scale.begin(DOUT_PIN, CLK_PIN);

Serial.println("Initializing...");

float calibration_factor = 796.55;

scale.set_scale(calibration_factor);

scale.tare();
pinMode(TRIG_PIN, OUTPUT);
pinMode(ECHO_PIN, INPUT);
pinMode(LED_PIN, OUTPUT);
pinMode(P_E, OUTPUT);
myservo.attach(9);


rtc.begin(DateTime(F(__DATE__), F(__TIME__)));
```

Fig. 11. Code2

```
weight+time+speaker+ultra + motor.ino
  rtc.adjust(DateTime(2023, 8, 24, 3, 29, 0));
}

void loop () {
  DateTime now = rtc.now();

  if (now.hour() == 3 && now.minute() >= 0 && now.minute() <= 30  || now.hour()==3 && now.minute() >=32 && now.minute() <=35 ) {


  long duration, distance_cm;

  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH);

  distance_cm = (duration / 2) / 29.1;

  Serial.print("Cat Distance: ");
```

Fig. 12. Code 3

```
weight+time+speaker+ultra + motor.ino
  Serial.print("Cat Distance: ");
  Serial.print(distance_cm);
  Serial.println(" cm");
  delay(2000);

  int threshold_distance = 10;

  if (distance_cm > threshold_distance && speak == 0) {


   digitalWrite(P_E, HIGH);
   delay(2000);
   digitalWrite(P_E, LOW);
   delay(1000);

  }


  else if(distance_cm < threshold_distance){

    speak=1;
```

Fig. 13. Code4

```
weight+time+speaker+ultra + motor.ino
    speak=1;

    digitalWrite(P_E, LOW);

    if (scale.is_ready()) {
    float weight = scale.get_units(10);
    Serial.print("Weight: ");
    Serial.print(weight);
    Serial.println(" grams");
    if (weight < 100){

      digitalWrite(LED_PIN, HIGH);


      for (pos = 0; pos <= 180; pos += 1) {

        myservo.write(pos);
        delay(15);

      }
      for (pos = 180; pos >= 0; pos -= 1) {
        myservo.write(pos);
Output
Using library Servo at version 1.2.0 in folder: C:\Users\Dristy\AppData\Local\Arduino15\libraries\Servo
Using library SPI at version 1.0 in folder: C:\Users\Dristy\AppData\Local\Arduino15\packages\arduino\hardware\avr\1.8.6\libraries\SPI
"C:\\Users\\Dristy\\AppData\\Local\\Arduino15\\packages\\arduino\\tools\\avr-gcc\\7.3.0-atmel3.6.1-arduino7/bin/avr-size" -A "C:\\Users\\Dristy\\AppData\\Local\\Temp\\
Sketch uses 8684 bytes (26%) of program storage space. Maximum is 32256 bytes.
Global variables use 464 bytes (22%) of dynamic memory, leaving 1584 bytes for local variables. Maximum is 2048 bytes.
```

Fig. 14. Code5

```
weight+time+speaker+ultra + motor.ino
109     for (pos = 180; pos >= 0; pos -= 1) {
110       myservo.write(pos);
111       delay(15);
112     }
113
114   }
115   Serial.println("food is given");
116   digitalWrite(LED_PIN, LOW);
117
118
119
120
121   delay(1000);
122   }
123
124   }
125
126
127
128
129   else
130   {
131     Serial.println("its not time");
```

Fig. 15. Code 6

```
weight+time+speaker+ultra + motor.ino
118
119
120
121   delay(1000);
122   }
123
124   }
125
126
127
128
129   else
130   {
131     Serial.println("its not time");
132     delay(3000);
133     speak=0;
134   }
135
136
137
138
139 }
140
Output
```

Fig. 16. Code 7

```
NodeMCU Food Container.ino
1   #include <ESP8266WiFi.h>
2   #include <WiFiClient.h>
3   #include <ESP8266HTTPClient.h>
4
5   WiFiClient client;
6   HTTPClient http;
7   String url;
8   String API = "JS6NQSAOIQ878ENJ";
9   String FieldNo = "1";
10
11  #define TRIG_PIN D7
12  #define ECHO_PIN D8
13  #define LED_PINGreen D4
14  #define LED_PINRed D3
15
16  void ConnectToWifi();
17  void SendGETRequest(float data);
18
19  int msg = 0;
20
21  void setup() {
22    Serial.begin(9600);
23    pinMode(LED_PINGreen, OUTPUT);
24    pinMode(LED_PINRed, OUTPUT);
25    pinMode(TRIG_PIN, OUTPUT);
26    pinMode(ECHO_PIN, INPUT);
27    ConnectToWifi();
28  }
```

Fig. 17. node 1

```
NodeMCU Food Container.ino
27    ConnectToWifi();
28  }
29
30  void loop() {
31
32
33    digitalWrite(TRIG_PIN, LOW);
34    delayMicroseconds(2);
35
36    digitalWrite(TRIG_PIN, HIGH);
37    delayMicroseconds(10);
38    digitalWrite(TRIG_PIN, LOW);
39
40    unsigned long duration = pulseIn(ECHO_PIN, HIGH);
41    float distance = duration * 0.034 / 2;
42
43    Serial.print("Food level: ");
44    Serial.print(distance);
45    Serial.println(" cm");
46
47    if (distance > 10 && msg == 0) {
48      SendGETRequest(distance);
49      digitalWrite(LED_PINRed, HIGH);
50      digitalWrite(LED_PINGreen, LOW);
51
52      msg = 1;
53    }
54    else if (distance <= 10) {
55      msg = 0;
```

Fig. 18. Node 2

```
NodeMCU Food Container.ino
54    else if (distance <= 10) {
55      msg = 0;
56      digitalWrite(LED_PINGreen, HIGH);
57      digitalWrite(LED_PINRed, LOW);
58    }
59
60    delay(10000);
61  }
62
63  void SendGETRequest(float data) {
64    url = "http://api.thingspeak.com/update?api_key=";
65    url += API;
66    url += "&field";
67    url += FieldNo;
68    url += "=";
69    url += data;
70    http.begin(client, url);
71    Serial.println("Sending GET request...");
72    int httpCode = http.GET();
73
74    if (httpCode == HTTP_CODE_OK) {
75      Serial.println("Data sent successfully");
76    } else {
77      Serial.print("Error in sending. HTTP code: ");
78      Serial.println(httpCode);
79    }
80
81    http.end();
Output
```

Fig. 19. Node 3

```
NodeMCU Food Container.ino
71    Serial.println("Sending GET request...");
72    int httpCode = http.GET();
73
74    if (httpCode == HTTP_CODE_OK) {
75      Serial.println("Data sent successfully");
76    } else {
77      Serial.print("Error in sending. HTTP code: ");
78      Serial.println(httpCode);
79    }
80
81    http.end();
82  }
83
84  void ConnectToWifi() {
85    WiFi.mode(WIFI_STA);
86    WiFi.begin("Dexter_Net_By_Suborno", "918273645");
87
88    Serial.print("Connecting to WiFi");
89    while (WiFi.status() != WL_CONNECTED) {
90      Serial.print('.');
91      delay(200);
92    }
93
94    Serial.println();
95    Serial.print("Connected! IP Address: ");
96    Serial.println(WiFi.localIP());
97
98
```

Fig. 20. Node 4

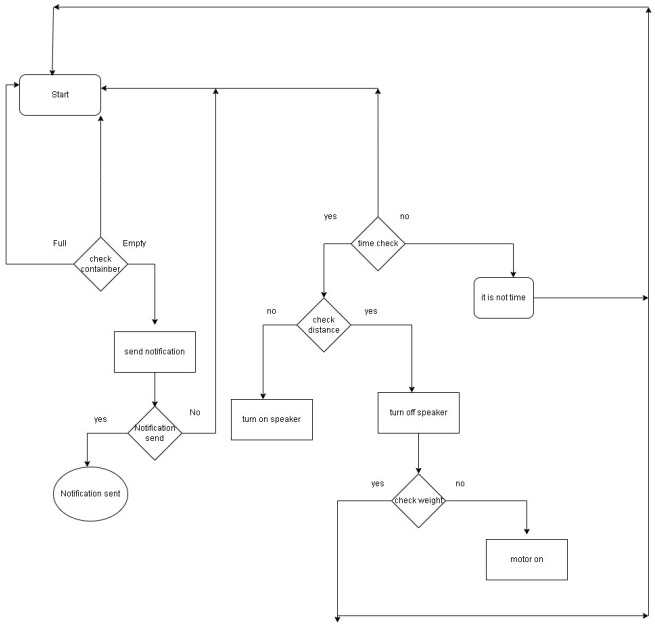

Fig. 21. Flow chart