# AUTOMATED PET FEEDER USING IOT

**A PROJECT SUBMITTED TO MAHATMA GANDHI UNIVERSITY, IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE MASTERS DEGREE IN COMPUTER SCIENCE**

**SUBMITTED BY,**

**ALAN CYRIL SUNNY**
**Reg No : 220011018071**

**POST GRADUATE DEPT. OF COMPUTER SCIENCE**
**BVM HOLY CROSS COLLEGE, CHERPUNKAL**
**KOTTAYAM, 68658**

**POST GRADUATE DEPT. OF COMPUTER SCIENCE**

**BVM HOLY CROSS COLLEGE**

# Certificate

Certified that the report entitled **AUTOMATED PET FEEDER** is the bonafide record of the project work done by **Mr. ALAN CYRIL SUNNY. (Register Number: 220011018071)** under our guidance and is submitted in partial fulfillment of the Master's degree in Computer Science, awarded by Mahatma Gandhi University Kerala.

Mr. Bilas Joseph
Project Guide

Mr. Binu M B
Head of the Department

Rev. Dr. Baby Sebastian Thonikuzhy

Principal

Submitted for viva- voce on----------

External Examiners

1.

2.

# DECLARATION

I hereby declare that the project work entitled **AUTOMATED PET FEEDER** submitted in partial fulfillment of the requirements for the award of the Master's degree in Computer Science from BVM Holy Cross College, Cherpunkal is record of bonafide work done under the guidance of Mr. Bilas Joseph.

Place: ALAN CYRIL SUNNY

Date: Reg. No: 220011018071

# ACKNOWLEDGEMENT

I am grateful to those people who have helped me in completing this project work. I take this opportunity to express my gratitude to all of them. First and foremost, I would like to thank God Almighty, who has showered his blessings on me to complete this project successfully.

I owe a deep sense of gratitude to Rev. Dr. Baby Sebastian Thonikuzhy, Principal, BVM Holy Cross College, Cherpunkal for his able leadership and guidance in all the official matters regarding this project.

I extent my sincere thanks to Mr. Binu M.B, Head of the department of Computer Science, who inspired me all through the preparatory stages of this project work. I am extremely happy to express my deep sense of gratitude to Mr. Bilas Joseph, Assistant Professor Department of Computer Science, for his inspiring and commendable support throughout the all stages of this project work. I express my deep sense of regard to all the faculty members of the Department of Computer Science, for their encouragement and excellent suggestions from time to time.

It is my pleasure to express my hearty thanks to my parents who provide all the support for the completion of the project both directly and indirectly. I am extremely thankful to all my classmates who have been a constant support during the development of my project.

Finally, I thank all the authorities and staffs of BVM Holy Cross College, Cherpunkal for letting me to do this project work.

Place:

Date

# CONTENT

# ABSTRACT

This mini project presents an Internet of Things (IoT) solution for remotely controlling a pet feeder using an ESP8266 microcontroller. The system integrates a web interface accessible over WiFi, allowing users to interact with the pet feeder. Key features include the ability to turn the feeder on/off, toggle a webcam feed, and customize feeding intervals. The ESP8266 microcontroller establishes a connection to a local WiFi network, and an embedded web server is created using the ESP8266WebServer library. A servo motor controls the pet feeder's mechanism, simulating the feeding action for the pet. The web interface provides a user-friendly experience, displaying real-time information about the servo and webcam status. Users can dynamically control the feeding process through the web interface, adjusting the feeding frequency and enabling/disabling the webcam feed. The project aims to offer convenience for pet owners, allowing them to monitor and feed their pets remotely. The project also emphasizes collaboration, as indicated by the team member names in the web interface. The system is designed to be extensible, and the code includes placeholders for additional

JavaScript logic to asynchronously communicate with the server for real-time status updates. Overall, this IoT mini project showcases a practical application of connected devices, demonstrating how technology can enhance the interaction between pet owners and their pets. The web-based control interface provides an intuitive and accessible means for managing the pet feeder's functionality.

# CHAPTER 1

## INTRODUCTION

The Automated Pet Feeder Control System is a mini-project designed to remotely control a pet feeder using an ESP8266 microcontroller. The system allows users to schedule feeding times for their pets and provides a web interface for monitoring and controlling the pet feeder. Additionally, it includes the capability to enable or disable a webcam feed for live monitoring of the pet. The Arduino Mini Project on Automated Pet Feeder introduces an innovative solution tailored to address the needs of pet owners seeking to enhance the feeding experience for their beloved companions. This project leverages Arduino microcontroller technology to create a customizable and user-friendly pet feeding system. By combining precise scheduling, portion control, and remote access capabilities, this project offers an efficient and reliable solution for pet care enthusiasts. IOT is a platform which can embed both software and hardware. It is obivious from that IoT is an efficient way to acces data. Automated Pet Feeder system was planned to ensure the time to time feeding of pet in absence of it's master, so that master can do his other work or tasks without worrying about feeding. This system has attractive design and aesthetic model. Arduino and IoT add's automation in the system. ESP8266 circuit to control the functions of the system.

# CHAPTER 2

# FEASIBILITY STUDY

## 2.1: Existing System

Current automated pet feeders on the market meet various parts of our design objectives but are quite costly. The pet want automatic feeder permits scheduled feeding and system configuration via a smartphone app. The Automatic Pet Feeder by wireless utilizes is to release food for the pet when close by and it is at the appropriate feed time.

It focuses on delivering smarter meal portions for pet, and it comes with a smartphone application as the main user interface. The smartphone application can modify a pet's feeding schedules and monitor a pet's overall dietary habits but it can monitor only one pet and does not support multiple pets.

It is a wifi enabled automated pet feeder that allows a user to remotely feed a pet. And it also comes with a smartphones application that can modify a pets feeding schedule and food amount. It is also supported as web, android and ios application.

## 2.2: Proposed System

The proposed system is also refered on smart home technology, including the smart pet door and pet feeder. The results not only present the key improvement of the pet monitor system involved in the IOT technology, but also demand of pet owners.

The basic vision behind the IOT, it may have a new way of operational method, it may have a new method of connecting devices, and there might be the even complete clean state approach. As the full operational definition is finialized, but there are numerous research issues that can be worked on. As a

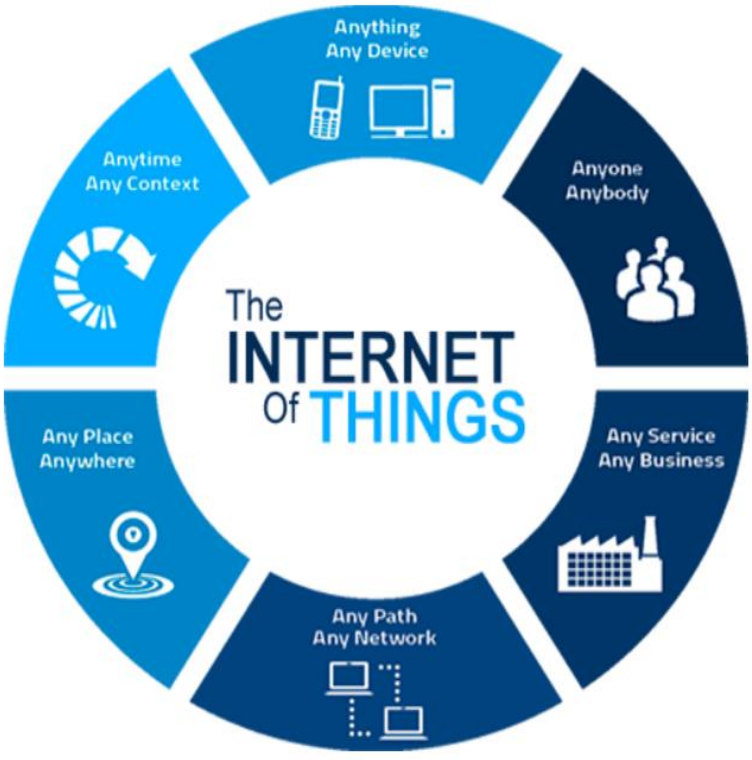
next step, we will fully integrate the other pet care devices into our system. With that, diverse needs of the owners can be met, and health monitor and entertainment topics for pets are all covered

# CHAPTER 3

# INTERNET OF THINGS (IOT)

## 3.1: INTRODUCTION

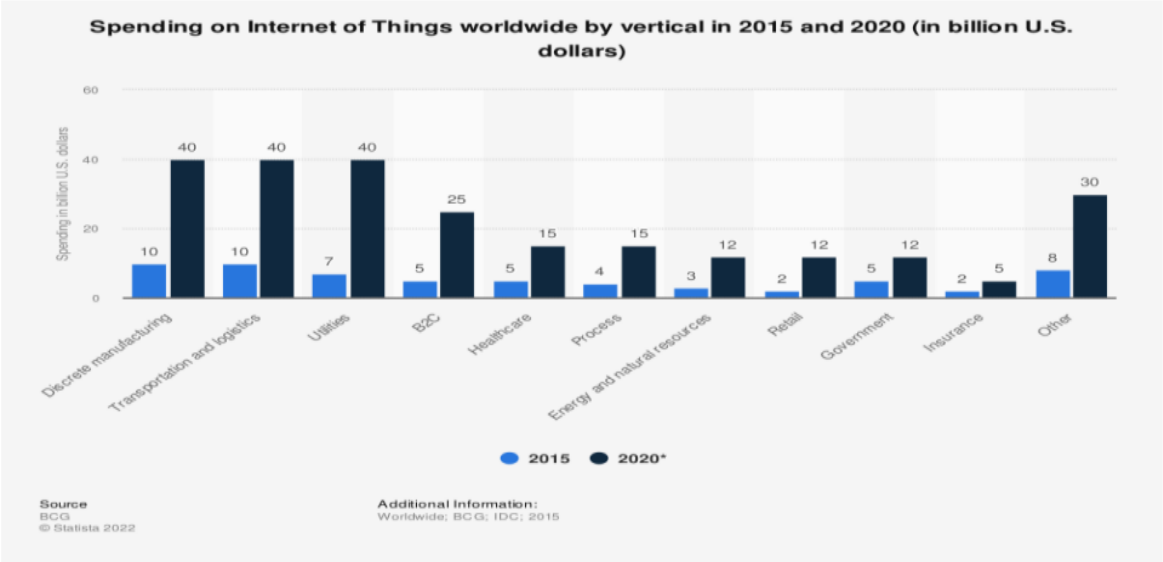
The Internet of things refers to a type of network to connect anything with the Internet based on stipulated protocols through information sensing equipment's to conduct information exchange and communications in order to achieve smart recognitions, positioning, tracing, monitoring, and administration. In this paper we briefly discussed about what IOT is, how IOT enables different technologies, about its architecture, characteristics & applications, IOT functional view & what are the future challenges for IOT.



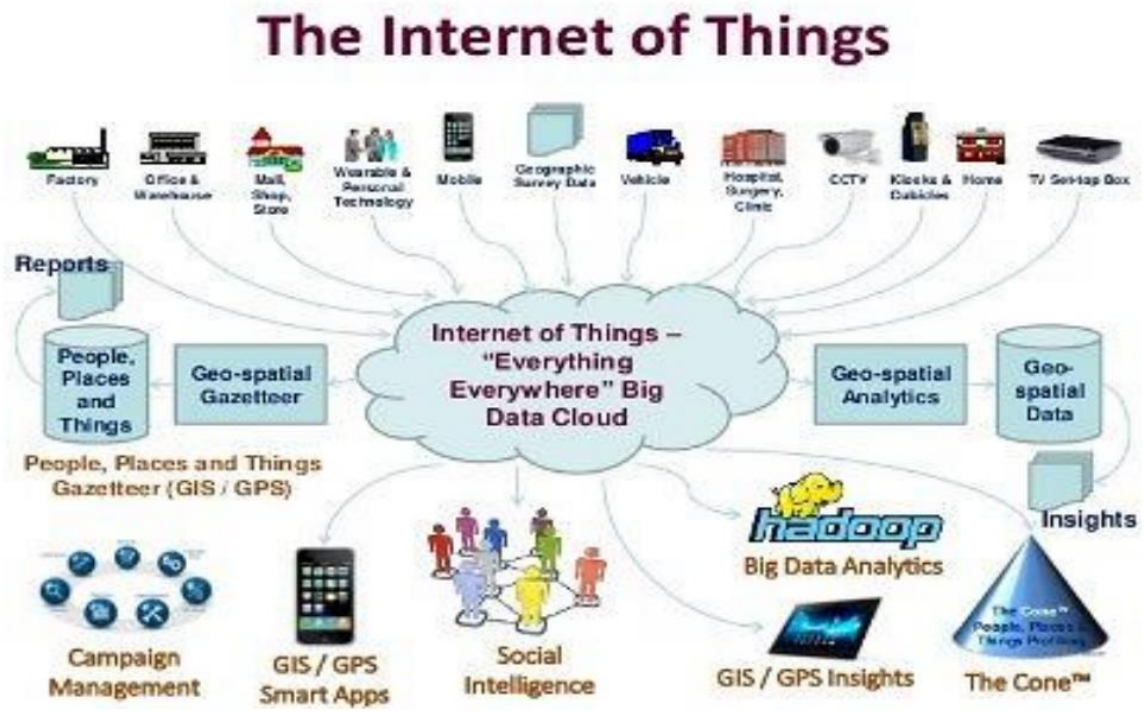
## 3.2: HISTORY OF IOT

The Internet of Things may be a hot topic in the industry but it's not a new concept. In the early 2000's Kevin Ashton was laying the groundwork for what would become the Internet of Things (IoT) at MIT's AutoID lab. Ashton was one of the pioneers who conceived this notion as he searched for ways that Proctor & Gamble could improve its business by linking RFID information to the Internet. The concept was simple but powerful. If all objects in daily life were equipped with identifiers and wireless connectivity, these objects could be communicate with each other and managed by computers. In a 1999 article for the RFID Journal Ashton wrote: "If we had computers that knew everything there was to know about things- using data they gathered without any help from us – we would be able to track and count everything, and greatly reduce waste, loss and cost. we would know when things needed replacing, repairing, or recalling, and whether they were fresh or past their best. We need to empower computers with their own means of gathering information, so they can see, hear and smell the world for themselves, in all its random glory. RFID and sensor technology enable computers to observe, identify and understand the world – without the limitations of human-entered data." At the time, this vison required major technology improvements. After all, how would we conduct everything on the planet? What type of wireless communications could be built into devices? What changes would need to be made to the existing Internet Infrastructure to support billons of new devices communicating? What would power these devices? What must be developed to make the solutions cost effective? There were more questions than answers to the IoT concepts in 1999. Today, many of these obstacles have been solved. The size and cost of wireless radios has dropped tremendously. IPv6 allows us to assign a communications address to billons of devices. Electronics companies are building Wi-Fi and cellular wireless connectivity into a wide range of devices. ABI Research estimates over five billon wireless chips will ship in 2013.2 Mobile data coverage has improved significantly with many networks offering broadband speeds. While not perfect, battery technology has improved and solar recharging has been built into numerous devices. There will be billions of objects connecting to the network with the next several years. For example, Cisco's Internet of Things Group(IOTG) predicts there will be over 50 billion connected devices by 2020. Internet of Things technologies are something more than just another innovation. They can change our life completely by making every single action we take easier and faster. Do you remember that just several

years ago automated doors were something unusual and incomprehensible? Now, we are used to them. Technologies keep being developed, and more significant changes are yet to come. IoT is one such change. The idea of devices being wirelessly connected to one global network reminds me of the Internet a bit. In this case, however, we have to deal with a concept that is more than that. The simplest example can be taken from our everyday lives. The coffee machines start preparing your favorite drink right at the moment you turn off the alarm clock. With the help of the same technology, the Uber driver receives the notification that it is time to drive to your place and pick you up 15 minutes before you leave. The opportunities for changes are unlimited; now, we just have to discover the possibilities and use them. So, let's discuss how to build an IoT application and join this type.




This is a rapidly growing technology in a world that is expecting 60 billon devices to be connected through the internet by 2020. Numerous SMART applications are and will be developed which will allow us to live a SMARTER, BETTER, EFFECTIVE, and more EFFICIENT life.




Work has already started and is moving with the speed of thought in the fields of SMART City, SMART Transportation, SMART Agriculture, SMART Health care etc. name any field and there are applications under development. There is a need and people are looking for young talent that has knowledge of this integration of sensors, embedded systems, and communication devices and their protocols. Computer Science topics, specially data aggregation, data analysis, and the subsequent development of algorithms for decision making, are in high demand as well. Is IoT difficult to learn? Whether one is from a hardware or software background, a school student or an accomplished coder, is it hard to gain expertise in this multi-disciplinary field? Then I will say NO! There is a platform available which enables you to not only learn, train yourself, and gain expertise but also equips you to shape-up your own innovative ideas, test them, and deploy them as an IoT enabled system. Pet feeding is a critical issue faced by various countries. Health problems have been growing at drastic rate especially in urban areas of developing countries due to industrialization and growing number of vehicles. It leads to release of a lot of pet foods. Harmful effects of pet food include mild allergic reactions such as irritation of the throat, eyes, and nose as well as some serious problems like bronchitis, heart diseases, pneumonia, lung and aggravated asthma. Premature deaths occur due to the presence of food poisonous.

## 3.3: LITERATURE REVIEW

IoT applications have become increasingly prevalent in the domain of pet care, offering innovative solutions for remote monitoring and automated systems. Within this context, researchers have explored the development of smart pet feeding systems that leverage IoT technologies. These systems often integrate sensors to monitor pet activities, facilitating personalized and automated feeding schedules. The integration of web interfaces allows users to remotely control and monitor these systems, contributing to a holistic approach to pet care.

The ESP8266 microcontroller has emerged as a versatile component in various IoT projects, particularly those involving web-enabled applications. Its cost-effectiveness and flexibility make it suitable for creating low-cost and efficient solutions for remote device control and monitoring. In the specific case of the Pet Feeder IoT System, the ESP8266 is employed to facilitate the web-based control interface, allowing users to manage feeding schedules and toggle webcam feeds remotely.

Web-based control interfaces have garnered attention in the literature due to their impact on user experience. The Pet Feeder IoT System's interface aligns with principles of accessibility and intuitiveness, providing users with a seamless way to interact with and control the device. This emphasis on user-friendly design is crucial in ensuring effective communication between pet owners and their automated pet care systems.

Servo motors play a pivotal role in the automation of pet care solutions, specifically in controlling feeding mechanisms. These motors offer precision and reliability in dispensing food, contributing to the accuracy of the overall system. The literature on servo motor applications in automation underscores their significance in creating controlled and efficient actions within IoT-enabled devices.

Collaboration in IoT projects has been explored, emphasizing the inclusion of diverse skill sets and innovative problem-solving. The Pet Feeder IoT System incorporates a collaborative approach, evident in the inclusion of team member names in the web interface. This collaborative effort enhances the overall development process and contributes to the project's success.

While IoT-based pet care solutions show promise, challenges such as power efficiency, security, and real-time communication require further exploration. Future research directions may focus on addressing these

challenges to enhance the reliability and security of such systems, paving the way for more robust implementations of IoT in the field of pet care. In summary, the literature reviewed highlights the growing interest in IoT applications for pet care, showcasing innovative solutions and paving the way for future advancements in the field.

# CHAPTER 4

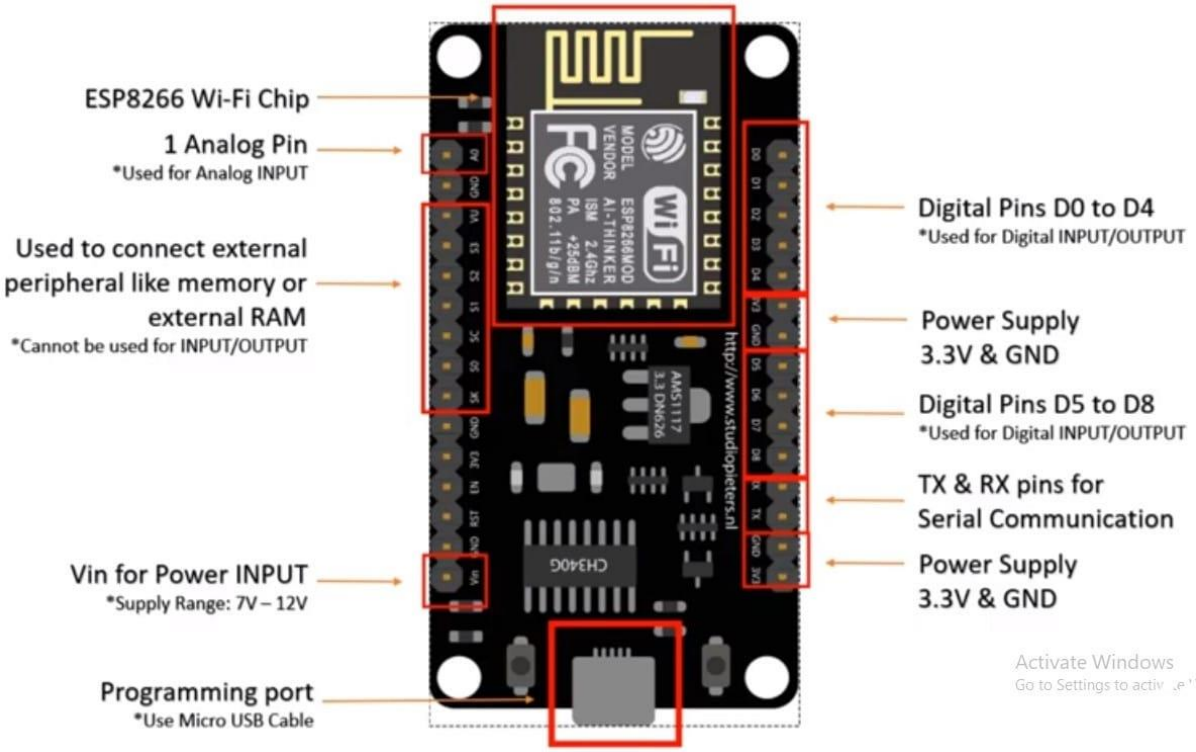
# HARDWARE REQUIREMENTS

## Hardware Components

1. ESP8266 Microcontroller
2. Servo Motor
3. Webcam
4. Power Supply

### 1. ESP8266 Microcontroller

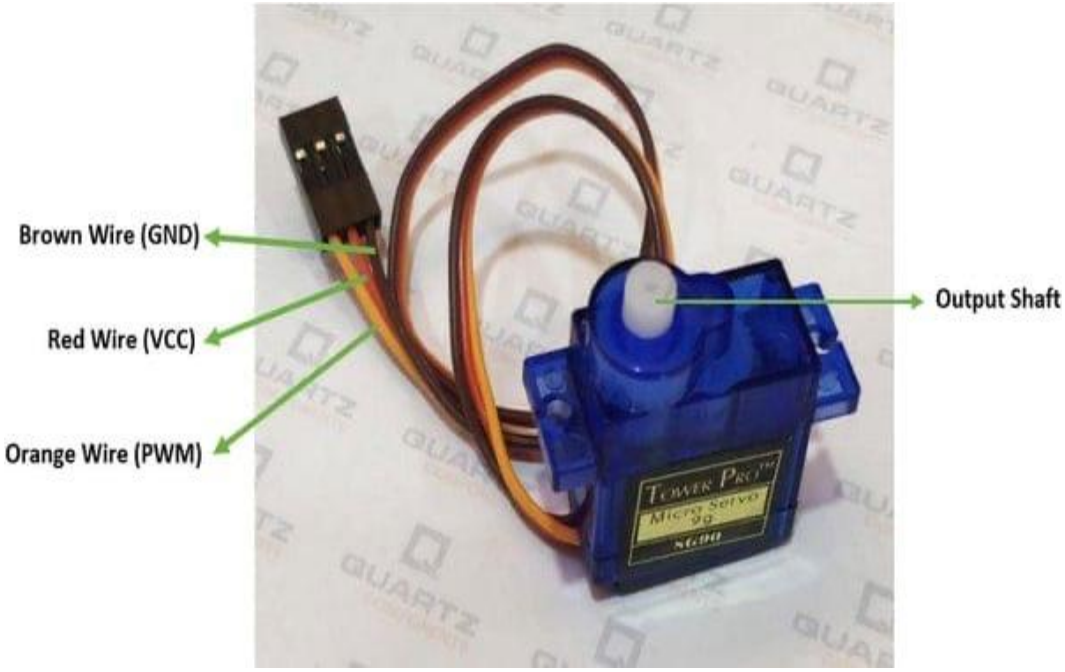
The ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP Protocol stack that can give any microcontroller access to your WiFi network.the ESP8266 is capable of either hosting an application or offloading all WiFi networking functions from another application processor. ESP8266 is complete self contained Wi-fi network solution that can carry software applications or through another applications processor uninstall all Wi-Fi networking capabilities. Built in cache Memory will help improve system performance and reduce memory requirement. Another situation is when wireless internet access assumes the task of Wi-Fi adapter; we can add it to any microcontroller-based design. The ESP8266 is highly integrated chip, including antenna switch balun, power management converter. So with minimal external circuitary, we include front end module, including the entire solution designed to minimize the space occupied by PCB. The central processing unit that controls the entire system. It manages WiFi connectivity and serves as the brain for processing requests.




## 2. Servo Motor

Mechanism for controlling the pet feeder's opening and closing action. Attached to a designated pin on the ESP8266 for control. It is a type of rotary actuator that allows for precise control of angular position. It consists of a small DC motor, gears and feedback mechanism. The feedback device often a potentiometer, continuously sends information about the motors current position to a controller. This feedback loops enables the controller to adjust the motors movement , ensuring accurate positioning.

Servo motors are commonly used in robotics, remote controlled vehicles, automation systems, and various applications where precise control of angular position is crucial.




## 3. Webcam

Captures live video feed for remote monitoring. Connected to a separate system, and its status is toggled through the ESP8266. An IP camera often reffered to as an IP cam or network camera, is a type of webcam that connects directly to a computer network, such as the internet or a local intranet. Unlike traditional webcams, which are usually usb based and connect to a specific device, IP cameras can operate independently and accessible over a network.

## 4. Power Supply

Provides the necessary power to the ESP8266, servo motor, and webcam. A power supply is an electronic device or system that provides electrical energy to another device or circuit. Its primary function is to convert input power from a source such as electrical outlet or a battery, into a form suitable for the operation of electronic devices. These are crucial components in various electronic systems, ensuring that connected devices receive the correct voltage, current and sometimes specific waveforms.

# CHAPTER 5

# SOFTWARE REQUIREMENTS

## Software Components

1. Arduino Sketch
2. ESP8266WiFi Library
3. ESP8266WebServer Library
4. Servo Library
5. ESP8266HTTPClient Library
6. Web Interface(HTML,CSS,JS)

### 1. Arduino Sketch

An Arduino sketch refers to the code written to program an Arduino microcontroller. Arduino sketches are typically written in the Arduino programming language, which is a simplified version of C and C++. The sketch contains instructions that define how the Arduino board should function and interact with connected hardware components.

The main structure of an Arduino sketch includes two essential functions: `setup()` and `loop()`. The `setup()` function is executed once when the Arduino starts, and it is used for initializing variables, configuring pins, and setting up initial conditions. The `loop()` function, as the name suggests, runs continuously in a loop after the `setup()` completes. It contains the main program logic, determining how the Arduino behaves during its operation.

Arduino sketches can read sensors, control actuators, communicate with other devices, and perform various tasks based on the programmed instructions. Once the sketch is written, it is uploaded to the Arduino board, allowing the microcontroller to execute the defined actions.

## 2. ESP8266Wifi Library:

It is a part of the Arduino core for the ESP8266 Wifi module. It provides functions and methods that simplify the process of connecting an ESP8266 microcontroller to Wifi networks. And handling network related tasks. This library is crucial when developing IOT projects using the ESP8266 module.

Key functions of the ESP8266WiFi library include:

1. WiFi.begin(ssid, password): Initiates a connection to a Wi-Fi network with the specified SSID and password.
2. WiFi.status(): Checks the current connection status, indicating whether the ESP8266 is connected, disconnected, or in the process of connecting.
3. WiFi.localIP(): Retrieves the local IP address assigned to the ESP8266 on the connected Wi-Fi network.
4. WiFi.scanNetworks(): Scans for available Wi-Fi networks and returns information about them, such as SSID and signal strength.
5. WiFiServer and WiFiClient classes: These classes enable the creation of a simple web server on the ESP8266, allowing it to respond to HTTP requests.

The ESP8266WiFi library simplifies the process of incorporating Wi-Fi capabilities into Arduino sketches, making it easier for developers to create projects that involve wireless communication and internet connectivity.

## 3. ESP8266WebServer library

The ESP8266WebServer library is an essential part of the Arduino core for the ESP8266 WiFi module. It facilitates the creation of web servers on the ESP8266, allowing the microcontroller to handle HTTP requests and responses. This is particularly useful for building IoT (Internet of Things) projects where web-based interfaces or control mechanisms are required.

Key features and functionalities of the ESP8266WebServer library include:

1. Server Handling: It enables the ESP8266 to act as a web server, handling incoming HTTP requests.

2. Routing: Developers can define routes or URL patterns and associate them with specific functions to execute when a corresponding request is received.
3. Request Handling: The library provides methods to access information about incoming HTTP requests, such as the HTTP method (GET, POST, etc.), parameters, headers, and client IP address.
4. Response Generation: It supports the generation of HTTP responses, making it possible to send HTML pages, JSON data, or other content back to the client.
5. Authentication: Allows for setting up basic authentication to restrict access to certain parts of the web server.

By using the ESP8266WebServer library, developers can create interactive web interfaces for their ESP8266-based projects, enabling users to control or monitor devices through a web browser. This makes it a powerful tool for building IoT applications with user-friendly interfaces.

## 4. Servo library

The Servo library in Arduino provides functions and methods to control servo motors easily. Servo motors are commonly used in robotics and other applications where precise control of angular position is required. The library simplifies the process of interfacing with and controlling these motors using an Arduino board.

Key functions and features of the Servo library include:

1. Servo: This is the main class in the library, representing a servo motor. Developers can create instances of this class to control multiple servo motors simultaneously.
2. attach(pin): Associates a servo motor with a specific Arduino pin. This informs the library which pin is connected to the control wire of the servo.
3. write(angle): Sets the angle of the servo motor. Servo motors typically have a limited range of motion, often 0 to 180 degrees.
4. read(): Retrieves the current angle set for the servo.
5. writeMicroseconds(microseconds): Sets the position of the servo based on microseconds. This provides more precise control over the servo's position.

The Servo library simplifies the programming of servo motors, allowing developers to easily integrate them into their Arduino projects. This is particularly useful for applications such as robotic arms, remote-controlled vehicles, and other projects requiring controlled and precise movement.

## 5. ESP8266HTTPClientLibrary:

The ESP8266HTTPClient library is a part of the Arduino core for the ESP8266 WiFi module. It enables the ESP8266 microcontroller to make HTTP requests to communicate with web servers or APIs over the internet. This is essential for IoT (Internet of Things) projects where data exchange with online services is required.

Key functionalities of the ESP8266HTTPClient library include:

1. begin(url): Initializes an HTTP request, specifying the URL of the server or API endpoint.
2. GET() and POST(): These methods perform HTTP GET and POST requests, respectively. GET is used to retrieve data from a server, while POST is used to send data to a server.
3. addHeader(name, value): Allows the inclusion of custom headers in the HTTP request, providing additional information to the server.
4. sendRequest(): Sends the HTTP request to the server.
5. responseStatusCode() and responseHeaders(): Retrieve information about the server's response, such as the HTTP status code and headers.
6. getString(): Retrieves the response body as a string.
7. writeToStream(stream): Writes the response body directly to a specified stream, which can be useful for handling large amounts of data efficiently.

The ESP8266HTTPClient library simplifies the process of integrating HTTP communication into ESP8266-based projects, enabling the microcontroller to interact with web servers and APIs for data exchange.

## 6. Web Interface

A web interface is a user interface for interacting with a software application or system through a web browser. It typically consists of three core technologies: HTML (Hypertext Markup Language), CSS (Cascading Style

Sheets), and JavaScript. Here's a brief explanation of each:

1. HTML (Hypertext Markup Language): HTML is the standard markup language used to structure content on the web. It provides a set of elements or tags that define the structure of a web page, such as headings, paragraphs, images, links, and forms. HTML creates the basic skeleton of a web page.
2. CSS (Cascading Style Sheets): CSS is a styling language that controls the presentation and layout of HTML elements. It allows developers to define styles such as colors, fonts, spacing, and positioning. CSS enables the separation of content and presentation, making it easier to maintain and update the visual aspects of a web page.
3. JavaScript: JavaScript is a scripting language that adds interactivity and dynamic behavior to web pages. It can be used to manipulate the HTML and CSS of a page in response to user actions. JavaScript is often employed to create features like form validation, animations, and real-time updates without requiring a page reload.

When combined, HTML, CSS, and JavaScript allow developers to create visually appealing and interactive web interfaces. HTML structures the content, CSS styles it, and JavaScript adds functionality. This trio is fundamental to modern web development, enabling the creation of engaging and user-friendly experiences across various devices.

# CHAPTER 6

# WORKING OF AUTOMATED PET FEEDER

This Automated Pet Feeder Control System utilizes an ESP8266 microcontroller for WiFi connectivity and hosts a web interface. The servo motor controls the pet feeder door based on scheduled feeding times. Users can toggle a webcam feed for live monitoring through the web interface. The system follows a client-server architecture, with the ESP8266 acting as the server. The code structure includes WiFi configuration, web server setup, servo control, and webcam feed toggling. Users can interact with the web interface to control the servo, toggle the webcam, and update feeding frequency settings. The project successfully provides a remotely controllable pet feeder with scheduling capabilities, and future improvements could include implementing security features and enhancing user interface friendliness. Overall, the system demonstrates a practical application of IoT technology in pet care.

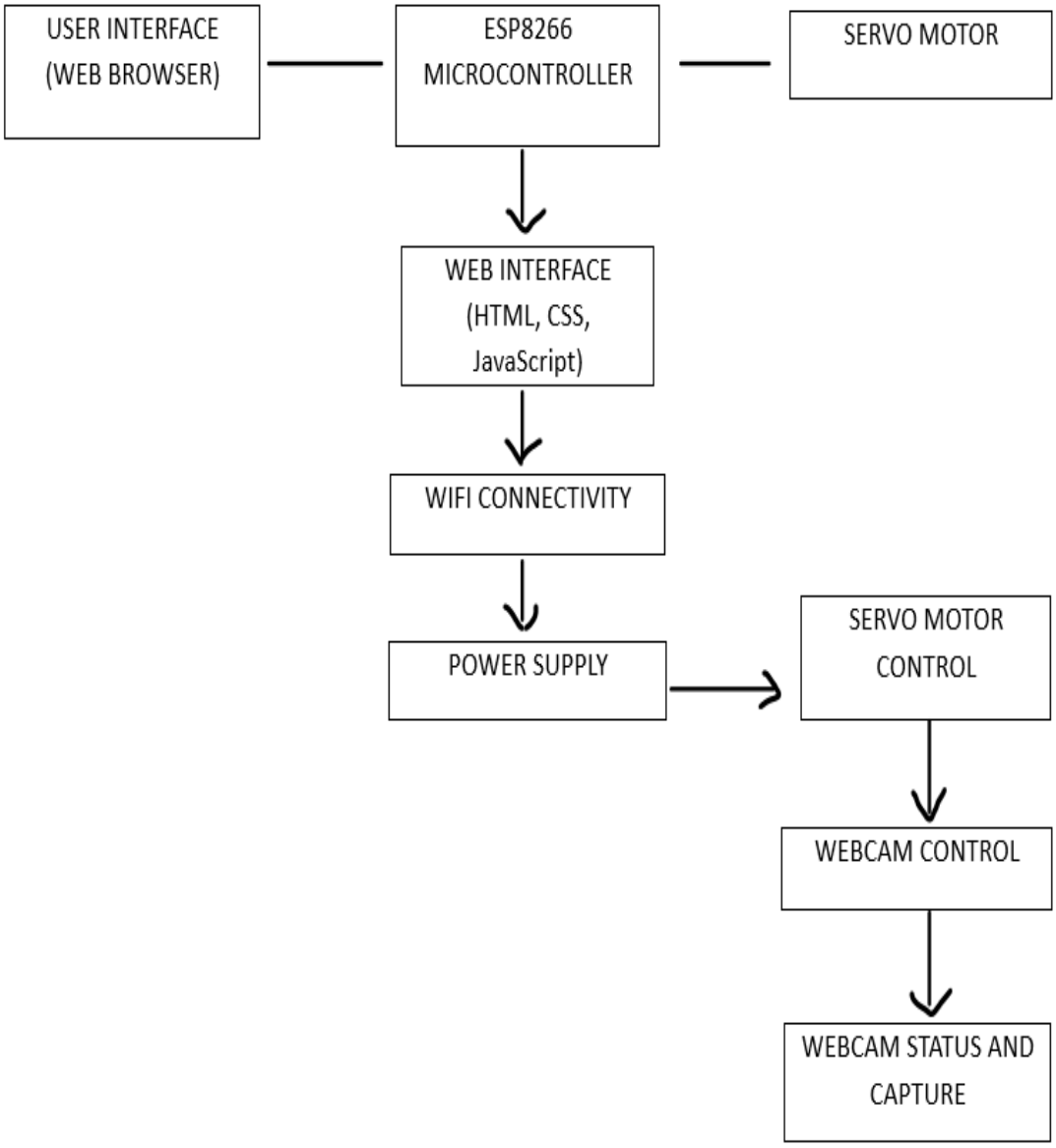

SET WET
FOR REGULAR USE:
For user manual and other technical details, visit www.nvartech.com/downloads
TEC CERTIFIED


ALWIN PAUL

# CHAPTER 7

# BLOCK DIAGRAM

## 7.1: Block diagram



## 7.2: Explanation

The automated pet feeding system involves connecting a web-based interface hosted on the ESP8266 microcontroller. The ESP8266 manages core logic, controls a servo motor for pet feeder actions, and governs webcam status for remote monitoring. The web interface, dynamically generated using HTML, CSS, and JavaScript, facilitates user interaction. The ESP8266 connects via WiFi and relies on a dedicated power supply for uninterrupted operation.

For setup:

- Connect the ESP8266 to a power source and ensure it's programmed with the Arduino sketch.
- Establish a WiFi connection for internet access.
- Wire the servo motor to the ESP8266, connecting power, ground, and control wires.
- Connect the webcam, ensuring compatibility and power supply.
- Create a dynamic web interface with controls for servo and webcam actions.
- Integrate the web interface code with the ESP8266 programming.
- Connect a dedicated power supply to ensure stable operation.
- Test the system, verifying user commands and power consistency.
- Implement safety measures, such as emergency stops and sensors for malfunctions.

This integrated setup enables a comprehensive automated pet feeding system with remote monitoring and control capabilities.

# CHAPTER 8

# CODE AND RESULTS

## 8.1:PET FEEDER

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <Servo.h>
#include <ESP8266HTTPClient.h>

const char* ssid = "KRAKEN"; // Enter your Wi-Fi SSID here
const char* password = "Ramboo1234@#"; // Enter your Wi-Fi password here

ESP8266WebServer server (80);

Servo servo;   // Create a servo object

int servoPin = 2; // D4 (GPIO2) for the servo control
int servoOpenAngle = 90; // Adjust as needed
int servoClosedAngle = 0; // Adjust as needed

unsigned long lastFeedingTime = 0;  // Variable to store the last feeding time
unsigned long feedingInterval = 3600;  // Default feeding interval in seconds (1 hour)
unsigned long feedingFrequency = 3600;  // Default feeding frequency in seconds (1 hour)
bool isServoOn = false;  // Flag to track the servo status
bool isWebcamOn = false;  // Flag to track the webcam status

const char* webcamUrl = "http://192.168.1.34:8080";   // Replace with your IP webcam URL

WiFiClient client;  // Create a WiFiClient object for HTTPClient
HTTPClient http;
```

```
unsigned long lastWebcamRefresh = 0; // Declare the variable
unsigned long webcamRefreshInterval = 10000; // Refresh the webcam feed every 10 seconds

void setup() {
  Serial.begin(115200);
  delay(100);

  Serial.println("Connecting to ");
  Serial.println(ssid);

  // Connect to your local Wi-Fi network
  WiFi.begin(ssid, password);

  // Check if Wi-Fi is connected to the network
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected..!");
  Serial.print("Got IP: ");
  Serial.println(WiFi.localIP());

  server.on("/", handle_OnConnect);
  server.on("/servoon", handle_servo_on);
  server.on("/servooff", handle_servo_off);
  server.on("/togglewebcam", handle_toggle_webcam);
  server.on("/updateinterval", handle_update_interval);
  server.on("/updatefrequency", handle_update_frequency);
  server.onNotFound(handle_NotFound);

  servo.attach(servoPin);

  server.begin();
```

```
  Serial.println("HTTP server started");
}

void loop() {
  server.handleClient();

  if (isServoOn) {
    // Check if it's time to feed the pet based on the feeding interval
    if (millis() - lastFeedingTime >= feedingFrequency * 1000) {
      // Perform the feeding action
      servo.write(servoOpenAngle);
      delay(1000); // Adjust as needed
      servo.write(servoClosedAngle);

      // Update the last feeding time
      lastFeedingTime = millis();
    }
  }

  // Periodically refresh the webcam feed
  if (isWebcamOn && millis() - lastWebcamRefresh >= webcamRefreshInterval)
  {
    refreshWebcamFeed();
    lastWebcamRefresh = millis();
  }
}

void refreshWebcamFeed() {
  http.begin(client, webcamUrl); // Use the WiFiClient object
  int httpCode = http.GET();
  if (httpCode == HTTP_CODE_OK)
  {
    // Successfully retrieved the webcam feed, do nothing
  } else {
    Serial.println("Failed to fetch the webcam feed.");
  }
  http.end();
```

```
}

void handle_OnConnect()
{
  Serial.print("Servo Status: ");
  Serial.println(isServoOn ? "ON" : "OFF");
  Serial.print("Webcam Status: ");
  Serial.println(isWebcamOn ? "ON" : "OFF");

  String response = SendHTML(isServoOn, isWebcamOn, feedingFrequency);
  server.send(200, "text/html", response);
}

void handle_servo_on()
{
  servo.write(servoOpenAngle);
  Serial.println("Servo Status: ON");
  isServoOn = true;
  String response = SendHTML(true, isWebcamOn, feedingFrequency);
  server.send(200, "text/html", response);
}

void handle_servo_off()
{
  servo.write(servoClosedAngle);
  Serial.println("Servo Status: OFF");
  isServoOn = false;
  String response = SendHTML(false, isWebcamOn, feedingFrequency);
  server.send(200, "text/html", response);
}

void handle_toggle_webcam()
{
  isWebcamOn = !isWebcamOn;
  Serial.print("Webcam Status: ");
  Serial.println(isWebcamOn ? "ON" : "OFF");
  String response = SendHTML(isServoOn, isWebcamOn, feedingFrequency);
```

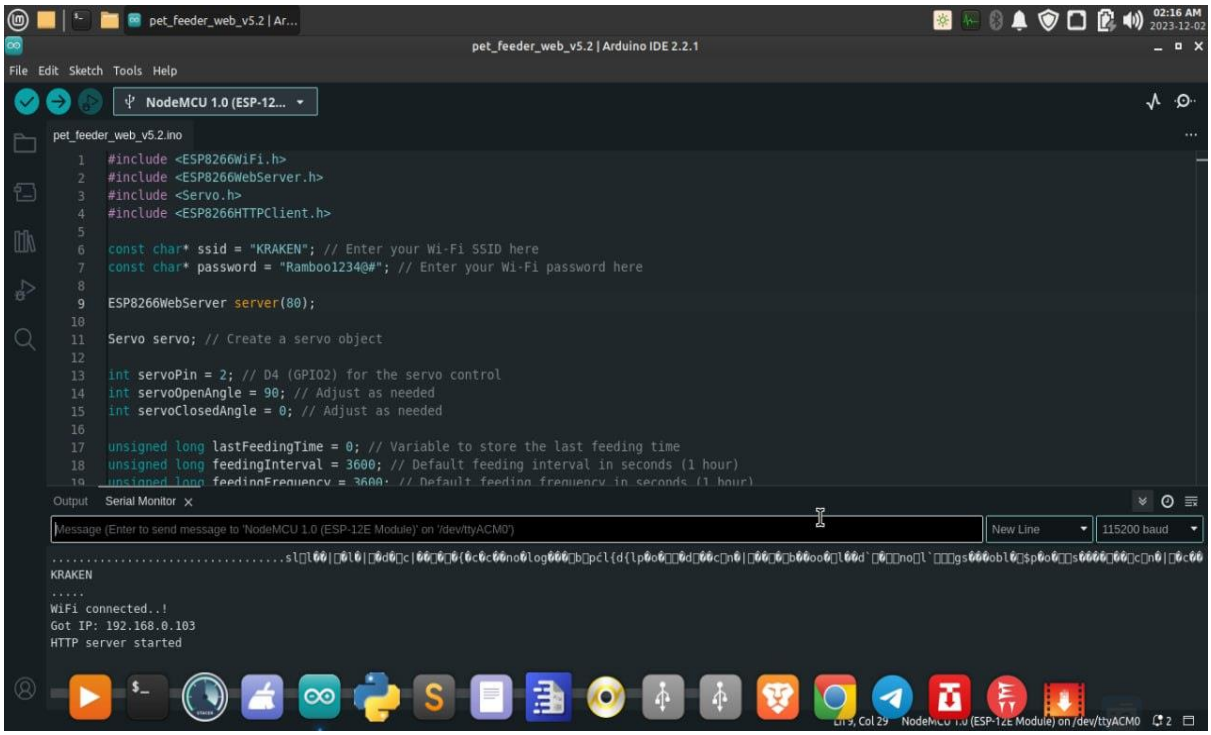
```
  server.send(200, "text/html", response);
}

void handle_update_interval()
{
  if (server.hasArg("interval"))
  {
    feedingFrequency = server.arg("interval").toInt();
   }
  String response = SendHTML(isServoOn, isWebcamOn, feedingFrequency);
  server.send(200, "text/html", response);
}

void handle_update_frequency()
{
  if (server.hasArg("frequency"))
  {
    feedingFrequency = server.arg("frequency").toInt();
  }
  String response = SendHTML(isServoOn, isWebcamOn, feedingFrequency);
  server.send(200, "text/html", response);
}

void handle_NotFound()
{
  server.send(404, "text/plain", "Not found");
}

String SendHTML(bool servoStatus, bool webcamStatus, unsigned long currentFrequency)
{
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr += "<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";
  ptr += "<title>Pet Feeder Control</title>\n";
  ptr += "<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}\n";
```

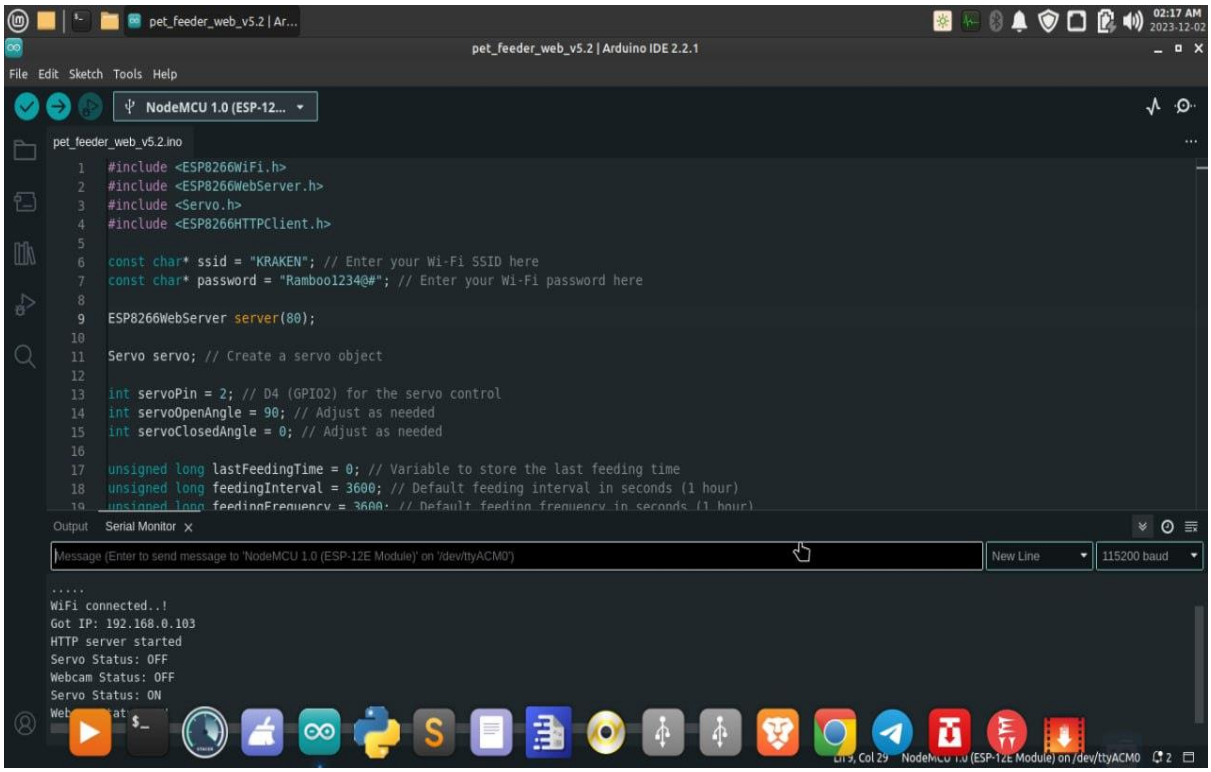
```
  ptr+="body{margin-top: 50px; background-color: #f4f4f4;} h1 {color: #444444; margin: 50px auto 30px;} h2, h3 {color: #1abc9c; margin: 30px auto;} h3 {margin-bottom: 10px;}\n";
  ptr+=".button {display: block; width: 200px; background-color: #1abc9c; border: none; color: white; padding: 13px 30px; text-decoration: none; font-size: 20px; margin: 0px auto 35px; cursor: pointer; border-radius: 4px;}\n";
  ptr += ".button.button-on {background-color: #16a085;}\n";
  ptr += ".button.button-on:active {background-color: #16a085;}\n";
  ptr += ".button.button-off {background-color: #e74c3c;}\n";
  ptr += ".button.button-off:active {background-color: #c0392b;}\n";
  ptr += ".menu {display: flex; justify-content: space-around; background-color: #333; padding: 10px;}\n";
  ptr += ".menu a {color: #fff; text-decoration: none; font-size: 18px;}\n";
  ptr += ".menu a:hover {text-decoration: underline;}\n";
  ptr += "p {font-size: 14px; color: #888; margin-bottom: 10px;}\n";
  ptr += ".iframe-container {height: 100vh;}\n"; // Updated container height to fill the viewport height
  ptr += "iframe {width: 100%; height: 100%; border: 0;}\n"; // Updated CSS for full-screen iframe
  ptr += ".left {float: left; width: 50%;}\n";
  ptr += ".right {float: right; width: 50%;}\n";
  ptr += "input {width: 60px; text-align: center;}\n";
  ptr += "form {margin-top: 20px;}\n";
  ptr += "footer {margin-top: 50px; color: #777;}\n";
  ptr += "</style>\n";
  ptr += "</head>\n";
  ptr += "<body>\n";

  // Navigation Menu
  ptr += "<div class='menu'><a href='#servo'>Servo</a> | <a href='#webcam'>Webcam</a> | <a href='#feeding'>Feeding Time</a> | <a href='#settings'>Settings</a> | <a href='#about'>About</a></div>";

  // Main Content
  ptr += "<h1>Pet Feeder</h1>\n";
  ptr += "<section id='controls'>";
  ptr += "<h2>Controls</h2>";
```

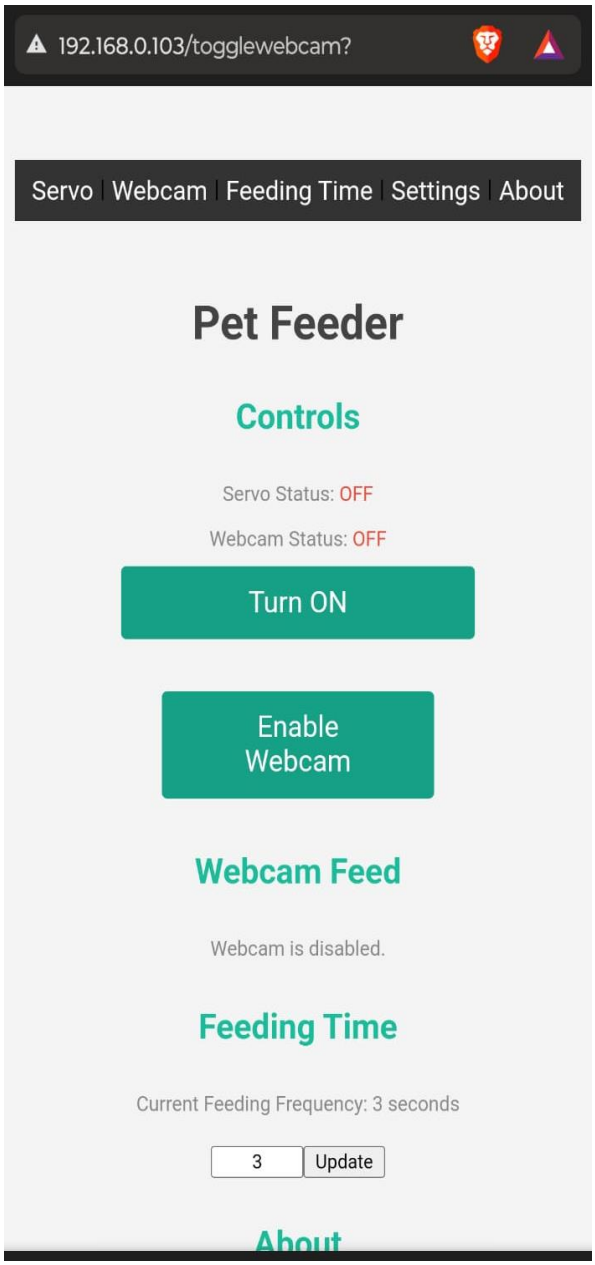
```
  ptr += "<p>Servo Status: ";
  ptr += servoStatus ? "<span style='color:#16a085;'>ON</span>" : "<span
style='color:#e74c3c;'>OFF</span>";
  ptr += "</p>";
  ptr += "<p>Webcam Status: ";
  ptr += webcamStatus ? "<span style='color:#16a085;'>ON</span>" : "<span
style='color:#e74c3c;'>OFF</span>";
  ptr += "</p>";
  ptr += "<a class='button ";
  ptr += servoStatus ? "button-off' href='/servooff'>" : "button-on'
href='/servoon'>";
  ptr += servoStatus ? "Turn OFF" : "Turn ON";
  ptr += "</a>\n";

  // Toggle Webcam Button
  ptr += "<form action='/togglewebcam' method='GET'>";
  ptr += "<button class='button ";
  ptr += webcamStatus ? "button-off'>" : "button-on'>";
  ptr += webcamStatus ? "Disable Webcam" : "Enable Webcam";
  ptr += "</button></form>\n";

  ptr += "</section>";

  ptr += "<section id='webcam'>"; // Centered
  ptr += "<h2>Webcam Feed</h2>";
  if (webcamStatus) {
    // Adjusted CSS for center positioning
    ptr += "<div class='iframe-container'>";
    ptr += "<iframe src='" + String(webcamUrl) + "'
frameborder='0'></iframe>\n";
    ptr += "</div>";
  } else {
    ptr += "<p>Webcam is disabled.</p>";
  }
  ptr += "</section>";
  ptr += "<section id='feeding'>";
  ptr += "<h2>Feeding Time</h2>\n";
```

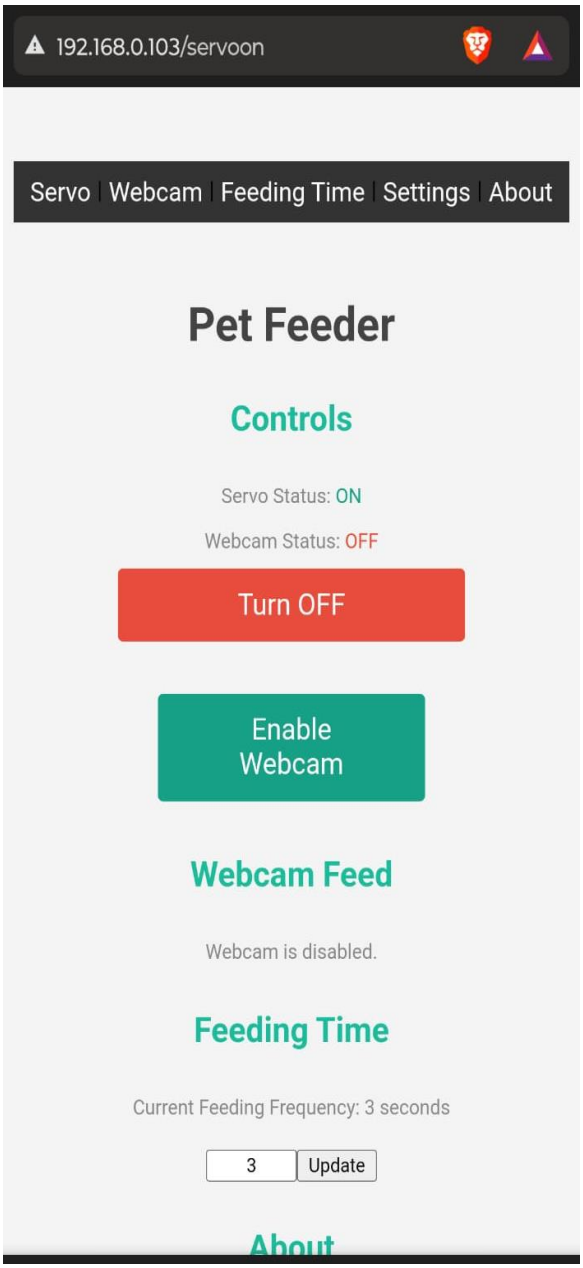
```
  ptr += "<p>Current Feeding Frequency: " + String(currentFrequency) + " seconds</p>\n";
  ptr += "<form action='/updatefrequency' method='GET'><input type='number' name='frequency' value='" + String(currentFrequency) + "'><input type='submit' value='Update'></form>\n";
  ptr += "</section>";

  ptr += "<section id='about'>";
  ptr += "<h2>About</h2>\n";
  ptr += "<p>This IoT Mini project controls a pet feeder remotely using a web interface. It allows you to feed your pet on a schedule and view a live webcam feed of your pet.</p>";
  ptr += "<p>Team Members: ALAN, BIJALI, LITTY, NANDHANA</p>\n";
  ptr += "</section>";

  ptr += "<footer>Designed with love by alan_cyril for IoT Mini Project - Version 5.2</footer>\n";

  ptr += "<script>\n";
  ptr += "function toggleControls() {\n";
  ptr += "var servoStatusElement = document.getElementById('servo-status');\n";
  ptr += "var webcamStatusElement = document.getElementById('webcam-status');\n";
  ptr += "var controlsButtonElement = document.getElementById('controls-button');\n";
  ptr += "var currentServoStatus = servoStatusElement.innerHTML;\n";
  ptr += "var currentWebcamStatus = webcamStatusElement.innerHTML;\n";
  ptr += "var newServoStatus = (currentServoStatus === 'OFF') ? 'ON' : 'OFF';\n";
  ptr += "var newWebcamStatus = (currentWebcamStatus === 'OFF') ? 'ON' : 'OFF';\n";
  ptr += "servoStatusElement.innerHTML = newServoStatus;\n";
  ptr += "webcamStatusElement.innerHTML = newWebcamStatus;\n";
  ptr += "controlsButtonElement.innerHTML = newServoStatus + ' | ' + newWebcamStatus;\n";
  ptr += "// Additional logic to send requests to the server to toggle the servo and webcam status\n";
  ptr += "// You can use AJAX or fetch to make asynchronous requests to the
```

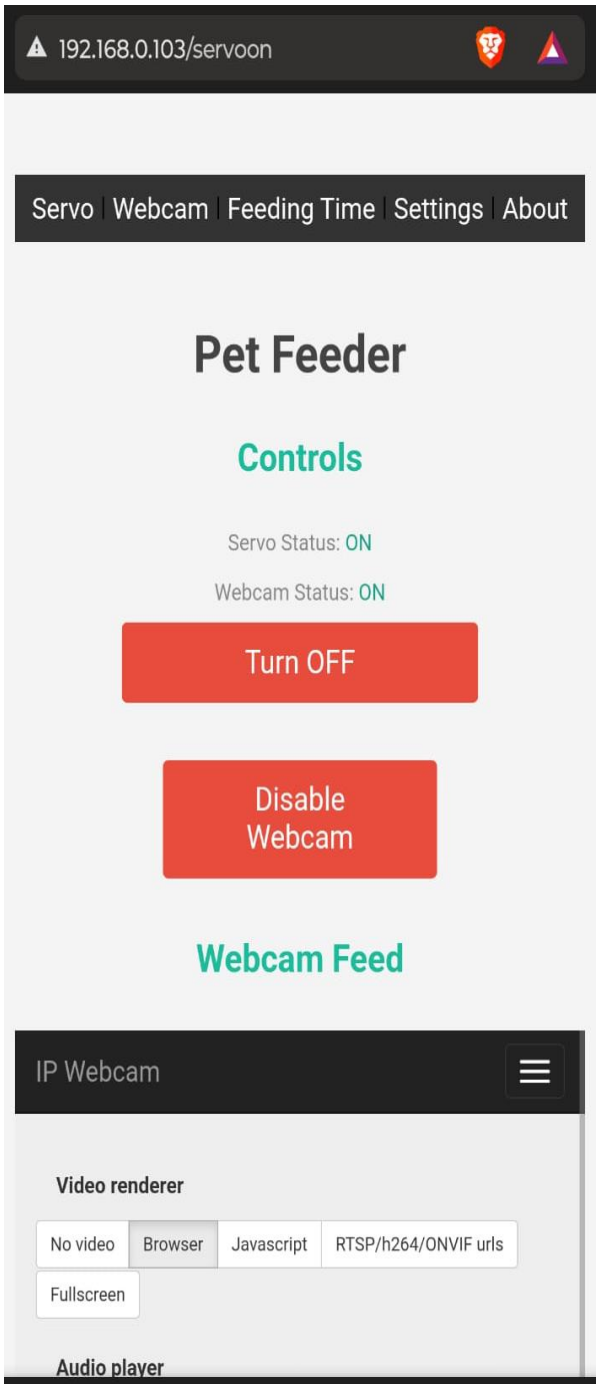
```
server here\n";
  ptr += "}\n";
  ptr += "</script>\n";
  ptr += "</body>\n";
  ptr += "</html>\n";

  return ptr;
}

// ... (Rest of the code remains unchanged)
```

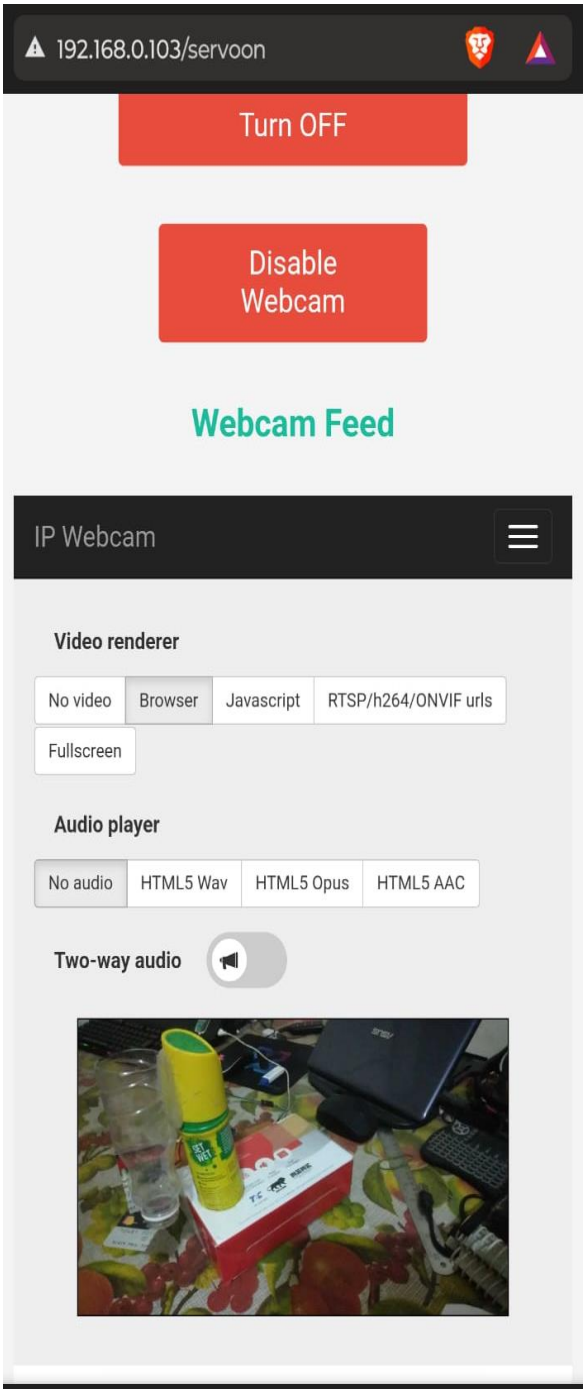

pet_feeder_web_v5.2 | Arduino IDE 2.2.1
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12...
pet_feeder_web_v5.2.ino
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <Servo.h>
#include <ESP8266HTTPClient.h>
const char* ssid = "KRAKEN"; // Enter your Wi-Fi SSID here
const char* password = "Ramboo1234@#"; // Enter your Wi-Fi password here
ESP8266WebServer server(80);
Servo servo; // Create a servo object
int servoPin = 2; // D4 (GPIO2) for the servo control
int servoOpenAngle = 90; // Adjust as needed
int servoClosedAngle = 0; // Adjust as needed
unsigned long lastFeedingTime = 0; // Variable to store the last feeding time
unsigned long feedingInterval = 3600; // Default feeding interval in seconds (1 hour)
Output Serial Monitor
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on '/dev/ttyACM0')
New Line
115200 baud
KRAKEN
.....
WiFi connected..!
Got IP: 192.168.0.103
HTTP server started

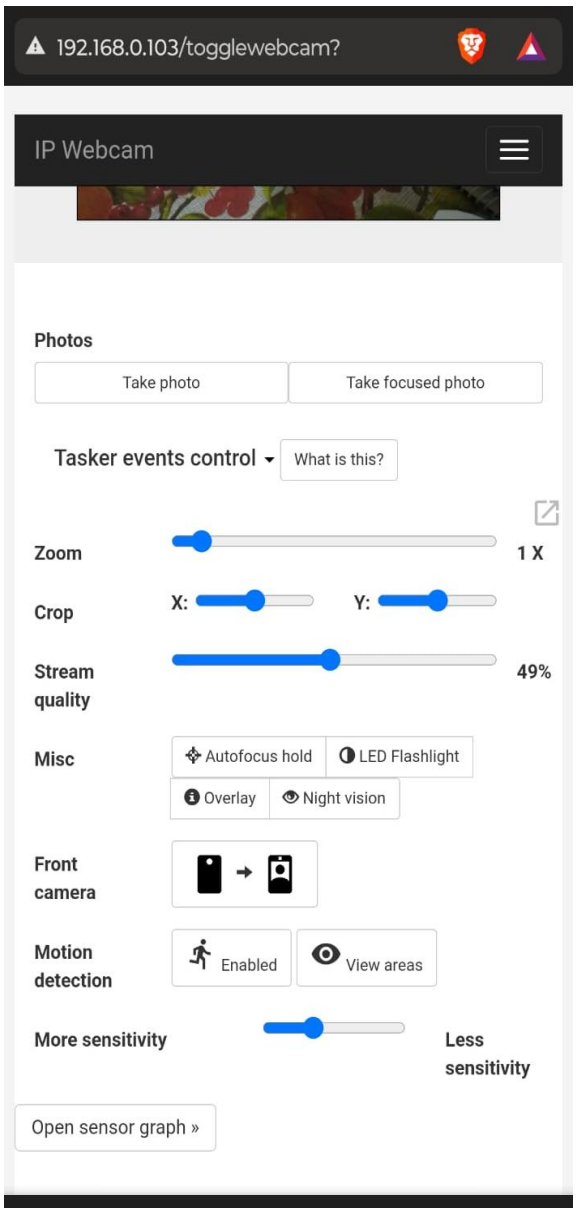

pet_feeder_web_v5.2 | Arduino IDE 2.2.1
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12...
pet_feeder_web_v5.2.ino
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <Servo.h>
#include <ESP8266HTTPClient.h>
const char* ssid = "KRAKEN"; // Enter your Wi-Fi SSID here
const char* password = "Ramboo1234@#"; // Enter your Wi-Fi password here
ESP8266WebServer server(80);
Servo servo; // Create a servo object
int servoPin = 2; // D4 (GPIO2) for the servo control
int servoOpenAngle = 90; // Adjust as needed
int servoClosedAngle = 0; // Adjust as needed
unsigned long lastFeedingTime = 0; // Variable to store the last feeding time
unsigned long feedingInterval = 3600; // Default feeding interval in seconds (1 hour)
Output Serial Monitor
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on '/dev/ttyACM0')
New Line
115200 baud
.....
WiFi connected..!
Got IP: 192.168.0.103
HTTP server started
Servo Status: OFF
Webcam Status: OFF
Servo Status: ON

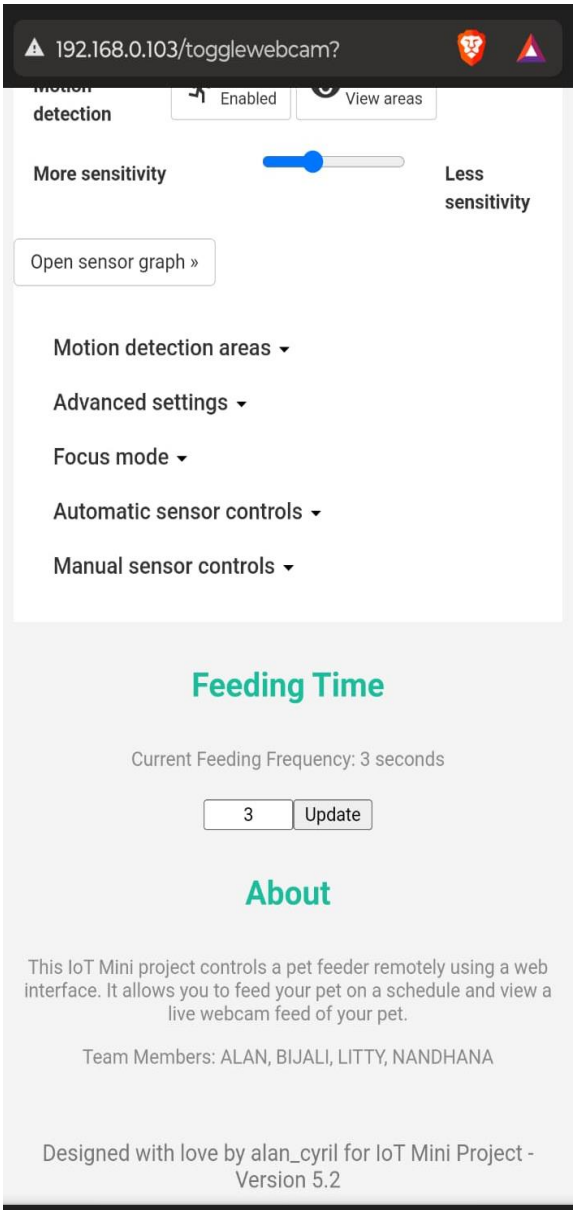

192.168.0.103/togglewebcam?
Servo | Webcam | Feeding Time | Settings | About
Pet Feeder
Controls
Servo Status: OFF
Webcam Status: OFF
Turn ON
Enable Webcam
Webcam Feed
Webcam is disabled.
Feeding Time
Current Feeding Frequency: 3 seconds
3
Update
About


192.168.0.103/servoon
Servo | Webcam | Feeding Time | Settings | About
Pet Feeder
Controls
Servo Status: ON
Webcam Status: OFF
Turn OFF
Enable Webcam
Webcam Feed
Webcam is disabled.
Feeding Time
Current Feeding Frequency: 3 seconds
3
Update
About


192.168.0.103/servoon
Servo | Webcam | Feeding Time | Settings | About
Pet Feeder
Controls
Servo Status: ON
Webcam Status: ON
Turn OFF
Disable Webcam
Webcam Feed
IP Webcam
Video renderer
No video
Browser
Javascript
RTSP/h264/ONVIF urls
Fullscreen
Audio player


192.168.0.103/servoon
Turn OFF
Disable Webcam
Webcam Feed
IP Webcam
Video renderer
No video
Browser
Javascript
RTSP/h264/ONVIF urls
Fullscreen
Audio player
No audio
HTML5 Wav
HTML5 Opus
HTML5 AAC
Two-way audio


192.168.0.103/togglewebcam?
IP Webcam
Photos
Take photo
Take focused photo
Tasker events control
What is this?
Zoom
1 X
Crop
X:
Y:
Stream quality
49%
Misc
Autofocus hold
LED Flashlight
Overlay
Night vision
Front camera
Motion detection
Enabled
View areas
More sensitivity
Less sensitivity
Open sensor graph »

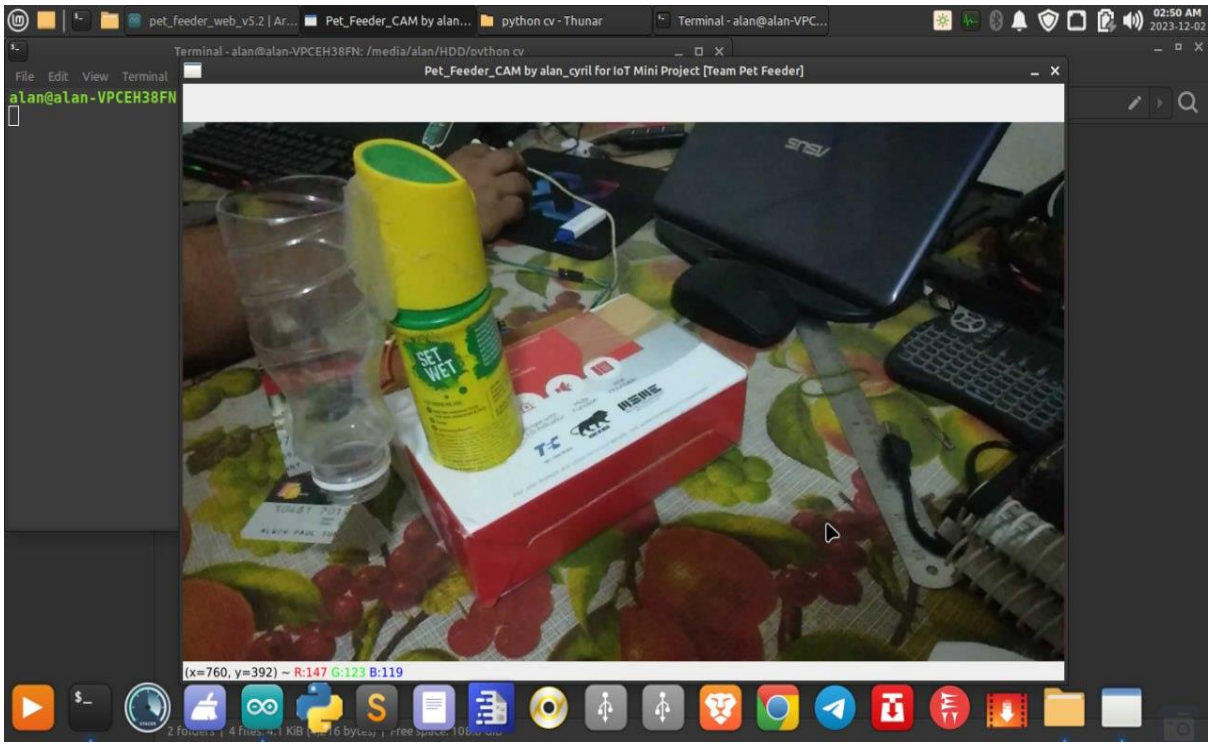

192.168.0.103/togglewebcam?
detection
Enabled
View areas
More sensitivity
Less sensitivity
Open sensor graph »
Motion detection areas
Advanced settings
Focus mode
Automatic sensor controls
Manual sensor controls
Feeding Time
Current Feeding Frequency: 3 seconds
3
Update
About
This IoT Mini project controls a pet feeder remotely using a web interface. It allows you to feed your pet on a schedule and view a live webcam feed of your pet.
Team Members: ALAN, BIJALI, LITTY, NANDHANA
Designed with love by alan_cyril for IoT Mini Project - Version 5.2

## 8.2:PYTHON CAM APP

```
# Import essential libraries
import requests
import cv2
import numpy as np
import imutils

# Replace the below URL with your own. Make sure to add "/shot.jpg" at last.
url = "http://192.168.0.104:8080/shot.jpg"

# While loop to continuously fetching data from the Url
while True:
        img_resp = requests.get(url)
        img_arr = np.array(bytearray(img_resp.content), dtype=np.uint8)
        img = cv2.imdecode(img_arr, -1)
        img = imutils.resize(img, width=1000, height=1800)
        cv2.imshow("Pet_Feeder_CAM", img)

        # Press Esc key to exit
        if cv2.waitKey(1) == 27:
                break

cv2.destroyAllWindows()
```

PET_FEEDER_CAM.py

```python
# Import essential libraries
import requests
import cv2
import numpy as np
import imutils

# Replace the below URL with your own. Make sure to add "/shot.jpg" at last.
url = "http://192.168.0.104:8080/shot.jpg"

# While loop to continuously fetching data from the Url
while True:
    img_resp = requests.get(url)
    img_arr = np.array(bytearray(img_resp.content), dtype=np.uint8)
    img = cv2.imdecode(img_arr, -1)
    img = imutils.resize(img, width=1000, height=1800)
    cv2.imshow("Pet_Feeder_CAM by alan_cyril for IoT Mini Project [Team Pet Feeder]", img)

    # Press Esc key to exit
    if cv2.waitKey(1) == 27:
        break

cv2.destroyAllWindows()
```



# CHAPTER 9

# CONCLUSIONS AND FUTURE SCOPE

## 9.1:CONCLUSION

The Pet Feeder IoT System represents a successful integration of Internet of Things (IoT) technologies to enhance pet care and monitoring. The project leverages the ESP8266 microcontroller, servo motors, and collaborative development principles to create a user-friendly and remotely accessible pet feeding solution. The key findings and conclusions are as follows:

1. Remote Pet Feeding Control:
   - The web-based control interface allows users to remotely manage pet feeding schedules. The integration of the ESP8266 microcontroller facilitates seamless communication between the user's web browser and the pet feeder, providing a convenient solution for pet owners.
2. Webcam Monitoring Feature:
   - The inclusion of a webcam monitoring feature adds an extra layer of interaction, enabling pet owners to visually check on their pets in real-time. This feature enhances the overall pet care experience and contributes to a comprehensive monitoring solution.
3. Collaborative Development Approach:
   - The collaborative development approach, evident in the inclusion of team member names in the web interface, highlights the benefits of teamwork in IoT projects. Diverse skill sets contribute to innovative problem-solving and a more robust implementation.
4. Servo Motor Precision:
   - The utilization of servo motors for controlling the feeding mechanism ensures precision and reliability in dispensing food. This enhances the accuracy of the pet feeding process and contributes to the overall effectiveness of the system.

## 9.2:FUTURE SCOPE

1. Enhanced Security Features:
   - Implementing additional security measures to safeguard the system against unauthorized access. This could include user authentication, secure communication protocols, and data encryption to protect user privacy.
2. Power Efficiency Improvements:
   - Exploring ways to enhance power efficiency, especially for prolonged operation. This may involve optimizing the power consumption of components, implementing sleep modes, or integrating alternative power sources such as solar panels.
3. Mobile Application Integration:
   - Developing a dedicated mobile application for convenient pet monitoring and control. A mobile app would provide users with flexibility and on-the-go access to the pet feeder system.
4. Machine Learning Integration:
   - Introducing machine learning algorithms for pet behavior analysis. This could enable the system to learn and adapt feeding schedules based on the pet's habits, contributing to a more personalized and responsive pet care solution.
5. Expandable Feeder Capacities:
   - Designing the system to accommodate a larger pet or multiple pets by enhancing the feeder's capacity. This ensures scalability and versatility to meet the needs of a broader range of pet owners.
6. User Feedback Integration:
   - Incorporating a feedback mechanism within the web interface for users to provide insights into their pet's behavior and preferences. This user feedback can be used to further refine and improve the system.
7. Integration with Pet Health Monitoring:
   - Exploring possibilities to integrate health monitoring features, such as weight sensors or health trackers, to provide a more comprehensive approach to pet well-being.

In conclusion, the Pet Feeder IoT System demonstrates the potential of IoT in pet care, and future developments can focus on enhancing security, power efficiency, and overall system capabilities to provide an even more sophisticated and user-centric pet care solution.

# BIBLIOGRAPHY


1. Tutorials and Documentation:
    - Espressif Systems. "ESP8266 Arduino Core."
    - Arduino. "Servo Library Reference."

2. Web Technologies:
    - MDN Web Docs. "HTML: Hypertext Markup Language."
    - MDN Web Docs. "CSS: Cascading Style Sheets."
    - MDN Web Docs. "JavaScript Guide."

3. Internet of Things (IoT) References:
   Shovic, John R. (2016). "Make: Getting Started with the Internet of Things." Maker Media, Inc.

4. Web Server Development:
   GitHub. (n.d.). "ESP8266WebServer Library."

5. Electronics and Hardware:
   Servo Motor Datasheet. (Manufacturer's Name, Year).

6. Project Collaboration:
   Team Collaboration Guide. "Effective Collaboration Strategies for IoT Projects."

7. Project Inspiration:
   Pet feeder by Circuit Digest

8. Python:
    - Learning OpenCV 5 Computer Vision with Python, Fourth Edition
      By Joseph Howse , Joe Minichino
    - AUTOMATE THE BORING STUFF WITH PYTHON: PRACTICAL PROGRAMMING FOR TOTAL BEGINNERS
      By Al Sweigart