**Embedded Systems Internship**

**Home Automation**

SUMMER INDUSTRIAL INTERNSHIP TRAINING REPORT

*Submitted by*

**22BEC1205**

**BECE399J – SUMMER INDUSTRIAL INTERNSHIP**

*in partial fulfilment for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

in

**ELECTRONICS AND COMMUNICATION ENGINEERING**

****

November 2024

**School of Electronics Engineering**

**DECLARATION BY THE CANDIDATE**

I hereby declare that the Industrial Internship Report entitled “**Embedded System Design Internship Program​**” submitted by me to VIT University, Chennai in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in **Electronics and Communications Engineering** is a record of Bonafide industrial training undertaken by me under the supervision of **Mr. Raghavendra, Maven-Silicon, Bengaluru**. I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Location: Chennai Signature of the Candidate

Date: 12.11.2024



****

**School of Electronics Engineering**

**BONAFIDE CERTIFICATE**

This is to certify that the Industrial Internship Report entitled “**Embedded Systems Internship: HOME AUTOMATION​**” submitted by **Jayakrishnan Menon (22BEC1205)** to VIT, Chennai in partial fulfilment of the requirement for the award of the degree of **Bachelor of Technology** in **Electronics and Communication Engineering** is a record of Bonafide industrial internship undertaken by him/her fulfils the requirements as per the regulations of this institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Signature of the Examiner** **Signature of the Examiner**

Date: Date:

**Head of the Department (B.Tech ECE)**

**ACKNOWLEDGEMENT**

I extend my sincere gratitude to Mr. Raghavendra and Maven Silicon for their continuous guidance and support throughout the duration of this project. Mr. Raghavendra's deep understanding and expertise in IoT and home automation were crucial in helping me overcome technical challenges and refine my approach to developing this system. The professional environment and resources provided by Maven Silicon, combined with Mr. Raghavendra's mentorship, not only provided me with valuable insights into the practical aspects of IoT integration but also encouraged me to think critically and enhance my problem-solving skills. I am truly thankful for their dedication, patience, and willingness to share knowledge, which greatly contributed to the successful completion of this project.

Jayakrishnan Menon

**(22BEC1205)**

PROJECT REPORT FOR HOME AUTOMATION

Table of Contents

[1. Abstract 8](#_Toc182381691)

[2. Introduction 9](#_Toc182381692)

[3. Literature Review 10](#_Toc182381693)

[4. Problem Statement and Project Specifications 12](#_Toc182381694)

[4.1 Function Requirements 12](#_Toc182381695)

[4.2 Non-function Requirements 12](#_Toc182381696)

[5. Hardware Components 13](#_Toc182381697)

[5.1 Description of Hardware used 13](#_Toc182381698)

[5.2 List of Hardware Components 15](#_Toc182381699)

[6. Software Description 16](#_Toc182381700)

[6.1 Functionality and requirements of the Main and Sub Functions: 16](#_Toc182381701)

[6.2 Main Functions Used: 17](#_Toc182381702)

[6.2 Secondary Functions Used: 18](#_Toc182381703)

[7. Code 21](#_Toc182381704)

[8. Testing and Results Obtained 25](#_Toc182381705)

[8.1 Methods of Testing: 25](#_Toc182381706)

[8.2 Observation Table for Auto Mode Testing 25](#_Toc182381707)

[8.3 Relevant images from Testing and Results 26](#_Toc182381708)

[9. Conclusion 29](#_Toc182381709)

[10. Future work 31](#_Toc182381710)

# 1. Abstract

This project explores the development and implementation of a smart home automation system utilizing the ESP32 microcontroller. As IoT technology increasingly shapes modern living, this project aims to harness the capabilities of the ESP32 to create a cost-effective, and user-friendly solution for remotely managing home appliances. The system is designed to connect with various sensors and actuators, allowing users to control and monitor devices such as lights and fans, through a web interface. Key functionalities include real-time monitoring, automated device control, and customizable settings to enhance convenience, energy efficiency, and security within the home environment. This report covers the project's specifications, hardware components and software description, code implementation, testing and results, as well as future enhancements. Through this project, the potential of IoT in transforming home environments into smart, efficient, and secure spaces is demonstrated.

# 2. Introduction

Convenience and efficiency are highly valued in the modern world. Embedded design-based home automation has become a popular technology that turns living areas into intelligent systems, improving comfort and saving energy. Embedded software acts as the brain of these systems, which is stored in the microcontrollers that regulate and observe the devices. Its significance stems from its capacity to streamline repetitive operations, maximize energy efficiency, and customize the user experience. Essentially, embedded design integrates hardware components and makes intelligent functionalities possible, acting as the unseen power that gives home automation life. A smarter, more convenient, and sustainable way of living is made possible by embedded design. With this project, we work on an ESP32 microcontroller board to attempt at making a simple home automation system.

# 3. Literature Review

Home automation has become an increasingly popular topic in recent years as the Internet of Things (IoT) continues to revolutionize the way we interact with our living spaces. Numerous researchers and industry leaders have explored the potential of embedded systems and smart home technologies to enhance comfort, energy efficiency, and security in residential environments.

One prominent study by Gupta et al. [1] investigated the design and implementation of a low-cost, modular home automation system using the Arduino platform. Their work emphasized the importance of affordability and scalability in driving wider adoption of such technologies. Similarly, Rashid et al. [2] presented a Raspberry Pi-based home automation system that integrated various sensors and actuators to enable remote monitoring and control capabilities.

The integration of machine learning and artificial intelligence has also been a focus area for enhancing home automation. Researchers explored the use of deep learning algorithms to predict user preferences and automatically adjust lighting, temperature, and other environmental factors. Such adaptive systems hold the promise of creating truly personalized and intelligent living spaces.

Several works have also explored the potential of voice assistants and natural language processing in home automation. Perumal et al. [3] developed a system that allowed users to control home appliances through voice commands, demonstrating the improved accessibility and convenience offered by such interfaces.

Security and privacy concerns have been another critical aspect of home automation research. Researchers like Sicari et al. [4] have investigated the challenges of ensuring data privacy and protection in IoT-based home automation systems, emphasizing the need for robust security measures and protocols.

The literature reviewed in this section provides valuable insights into the state-of-the-art home automation, covering aspects such as hardware design, software architectures, user interaction methods, and security considerations. These works have informed the development of the home automation system presented in this report, guiding the selection of technologies and the overall system design.

References:

[1] Gupta, A., Verma, T., & Kulshrestha, A. (2016). Design and implementation of low-cost, modular home automation system. 2016 International Conference on Connectivity and Embedded Systems (ICCES), 1-4.

[2] Rashid, T. A., Fadhil, H. A., & Zebari, R. R. (2017). Raspberry Pi-based home automation system. 2017 International Conference on Advanced Science and Engineering (ICOASE), 80-85.

[3] Perumal, T., Sulaiman, M. N., & Leong, C. Y. (2015). Interoperability for smart home environment using web services. 2015 IEEE International Conference on Consumer Electronics, 54-55.

[4] Sicari, S., Rizzardi, A., Grieco, L. A., & Coen-Porisini, A. (2015). Security, privacy and trust in Internet of Things: The road ahead. Computer Networks, 76, 146-164.

# 4. Problem Statement and Project Specifications

This section describes the specifications and constraints within which we had to make a working model of our embedded systems project

AUTO MODE:

* When a person enters the room and if there is no daylight in the room, the bulb will switch on
* When the person moves out or daylight returns, the bulb will switch off
* When the temperature of the room goes above a certain threshold and if a person is there in the room, the fan will start
* If the person moves our if the room or the temperature drops below the threshold, the fan will stop

MANUAL MODE:

* The fan and the bulb can be switched on and off through a browser in mobile or desktop when connected to the same home network through an appropriate IP address
* The bulb or the fan cannot be controlled manually if the system is in auto mode

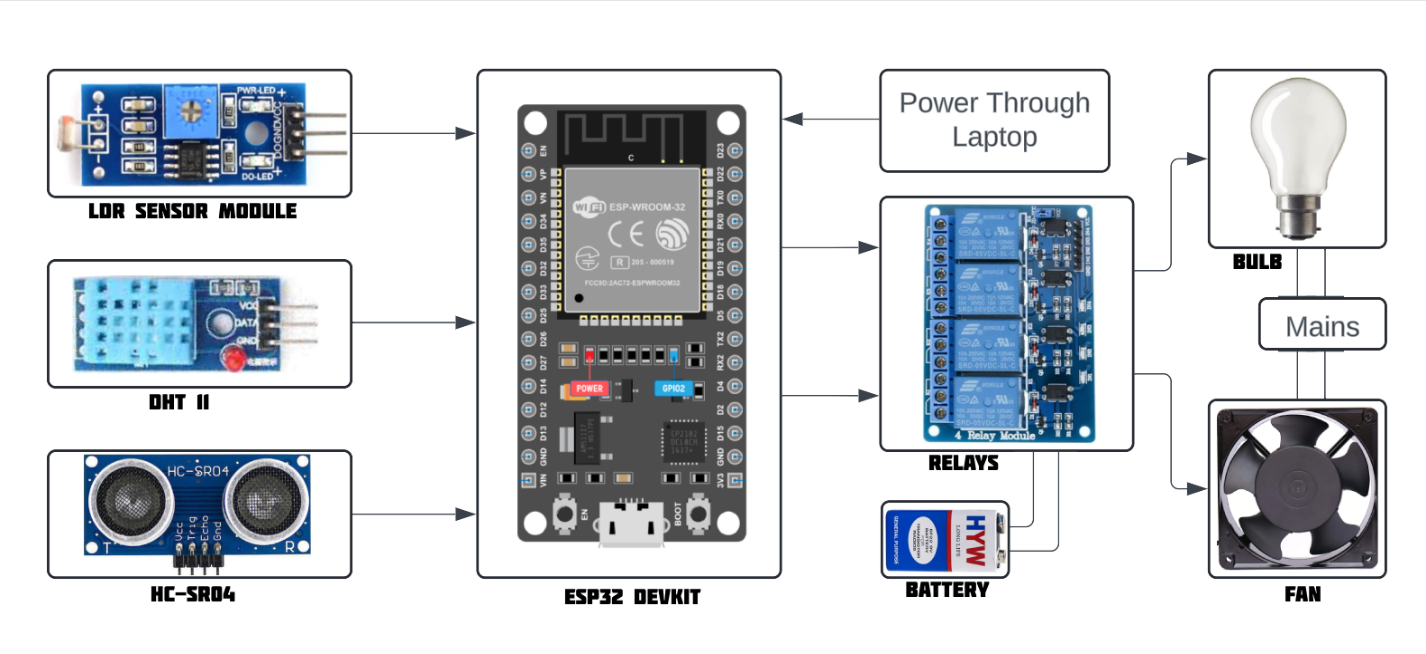
## 4.1 Function Requirements

* This project controls two home appliances, a light bulb and a fan connected to a 230V supply through relays
* The controller board is connected to the home Wi-Fi network
* The fan and bulb are controlled by a mobile connected to home Wi-Fi network
* Auto or Manual control mode can be selected on the mobile phone

## 4.2 Non-function Requirements

* The bulb and fab should switch on within 1 minute of pressing the button on the mobile phone in Manual mode
* The bulb and fab should switch on within 1 minute of the entry of a human into the room in the Auto mode
* The system should work in the temperature range – 0 to 60 degrees centigrade

# 5. Hardware Components



**Block diagram of Home automation Project**

## 5.1 Description of Hardware used

This section will provide a brief explanation of each hardware component used in the home automation system and how the various Functional requirements were met:

* **ESP32 Microcontroller Development Board:**

The ESP32 is the heart of the system. It's a single-board computer containing a powerful microcontroller unit (MCU) with built-in Wi-Fi and Bluetooth connectivity. This board allows you to program the embedded software that controls and interacts with all the other devices. Using this board was more efficient compared to using a board from the Arduino family due to its inbuilt Wi-Fi capability and processing speed.

* **LDR Sensor Module:**

An LDR (Light Dependent Resistor) module detects and measures ambient light levels. Its resistance changes based on the light intensity. This module can be used to trigger automated lighting or adjust settings based on sunlight availability. This sensor sends a low signal when there is light present above a threshold and sends a high signal when the ambient light goes below the threshold. In our project, the D0 pin of the LDR module was connected to the D18 pin of the ESP32 and the Vcc and GND pins of the sensor were given to the 3.3V and GND pin of the ESP32 respectively. The inbuilt potentiometer was also slightly adjusted to suit the threshold

* **DHT 11 Module:**

The DHT 11 module is a digital temperature and humidity sensor. It provides digital signals representing the surrounding temperature and humidity levels. This data can be used to automate climate control systems or for environmental monitoring purposes. In our project, the DATA pin of the DHT 11 was connected to the D15 pin of the ESP32 and the Vcc and GND pins of the sensor were given to the 3.3V and GND pin of the ESP32 respectively.

* **HC-SR04 Ultrasonic Sensor Module:**The HC-SR04 is a sensor that uses ultrasonic sound waves to detect the distance to objects. It emits a high-frequency sound pulse and measures the time it takes for the echo to return. This allows the microcontroller to calculate the distance between the sensor and any object within its range. In our project, the ECHO and TRIG pins of the HC-SR04 were connected to the D19 and D21 pins of the ESP32 respectively. The Vcc and GND pins of the sensor were given to the 3.3V and GND pin of the ESP32 respectively.
* **Relays:**

Relays act as electronic switches. They control the flow of current to higher-power devices like bulbs or fans. The ESP32 sends control signals to the relay, which then turns the connected device on or off. The relay we used was a 4-channel relay with optocouplers and worked on negative logic. The D2 and D4 pins of the ESP32 controlled the fan and light on the other side of the relay module. Additionally, the Vcc and GND pins of the module were given to the 3.3V and GND pin of the ESP32 respectively.

* **Battery (Optional):**

A battery was used as a power source to power the electromagnet of the relay since ESP32 works on a digital logic between 0 and 3.3volts, which is insufficient to drive the electromagnets present in the relays.

* **Bulb:**

This can be any standard incandescent bulb or a more energy-efficient LED bulb that works on 230V 50Hz. On our project we used an LCD bulb.

* **Fan:**

We used a 230V 50Hz fan of dimensions of 12cm by 12cm.

## 5.2 List of Hardware Components

|  |  |  |
| --- | --- | --- |
| **Serial No.** | **Hardware** | **Quantity** |
| 1. | ESP32 DEVKIT V1 | 1 |
| 2. | DHT 11 | 1 |
| 3. | HC-SR04 | 1 |
| 4. | LDR Sensor Module | 1 |
| 5. | Jumper Wires | As per need |
| 6. | Breadboard | 2 |
| 7. | Relay Modules (4 in 1) | 1 |
| 8. | 230V Fan (12cm x 12cm) | 1 |
| 9. | Light Bulb | 1 |
| 10. | Light Bulb Holder | 1 |
| 11. | Twin twisted wire | 2 |
| 12. | Plugs | 2 |
| 13. | Battery | 1 |

# 6. Software Description

## 6.1 Functionality and requirements of the Main and Sub Functions:

**Functions related to Automatic Mode:**

* **readDarkness():**
  + **Functionality:** Checks the LDR sensor state to determine if it's dark.
  + **Requirement:** Ensures the bulb turns on in auto mode when there's no daylight (darkness detected).
* **readDist():**
  + **Functionality:** Measures distance using the ultrasonic sensor to detect a person's presence.
  + **Requirements:**
    - Ensures the bulb turns on in auto mode when a person enters the room (object detected using distance).
    - Ensures the fan turns on when a person is present and the temperature is high.
* **readTemp():**
  + **Functionality:** Reads temperature data from the DHT sensor.
  + **Requirement:** Ensures the fan turns on in auto mode when the temperature goes above a certain threshold.
* **lightCtrlAuto():**
  + **Functionality:** Controls the light based on darkness and object detection flags.
  + **Requirement:** Turns on the bulb in auto mode when it's dark and a person is present (based on flags).
  + **Requirement:** Turns off the bulb in auto mode when it's not dark or no person is detected (based on flags).
* **fanCtrlAuto():**
  + **Functionality:** Controls the fan based on object detection and temperature flags.
  + **Requirement:** Turns on the fan in auto mode when a person is present and the temperature is high (based on flags).
  + **Requirement:** Turns off the fan in auto mode when no person is detected or the temperature drops below the threshold (based on flags).

**Functions related to Manual Mode:**

* **loop() (client handling section):**
  + **Functionality:** Listens for incoming client connections and handles commands received through a web browser on a mobile or desktop device.
  + **Requirement:** Enables manual control of the fan and bulb through a web interface.
  + **Requirement:** Disables manual control buttons on the web interface when the system is in auto mode (h flag).
* **loop() (button handling section):**
  + **Functionality:** Checks for specific commands in received data (GET /fanOn, etc.) and sets corresponding pins (HIGH/LOW) to control the fan and bulb.
  + **Requirement:** Allows turning the fan and bulb on/off manually through web interface buttons.

**Other Functions:**

* **sensor\_IO\_init():**
  + **Functionality:** Initializes sensors and sets pin modes for Fan and Light control.
  + **Requirement:** Sets up the hardware components for sensor readings and device control.
* **wifi\_server\_init():**
  + **Functionality:** Initializes WiFi connection and starts a web server for handling client requests.
  + **Requirement:** Enables the controller board to connect to the home WiFi network for communication with the mobile device.
* **lightStatus() and fanStatus():**
  + **Functionality:** Not directly related to requirements, but provide debugging information by printing the current status of the bulb and fan to the serial monitor.

**Meeting Response Time Requirements:**

* The response time for manual control (within 1 minute) is not directly addressed by the code itself. It depends on factors like network latency and server processing time. However, the code ensures immediate action (setting pins HIGH/LOW) upon receiving commands from the web interface.
* The response time for automatic mode (within 1 minute of entry/exit) is also not directly controlled by the code. It depends on the chosen sensor reading intervals (e.g., delay() in loop()) and how frequently these readings trigger changes. By adjusting these delays, the response time for automatic control can be optimized.

## 6.2 Main Functions Used:

**1. setup():**

* **Serial Communication:** Initializes serial communication for debugging purposes.
* **Sensor and Pin Setup:**
  + Calls sensor\_IO\_init() to perform the following:
    - Initializes the DHT temperature sensor.
    - Sets Fan and Light control pins as outputs and ensures they're initially off.
    - Sets pin modes for ultrasonic sensor (trigger and echo) and LDR sensor (input).
* **WiFi Connection:**
  + Calls wifi\_server\_init() to:
    - Connect to the specified WiFi network.
    - If connected, start the server and print the IP address.
    - If not connected, print a message indicating failure.

**2. loop():**

* **Client Handling:**
  + Continuously checks for incoming client connections (server.accept()).
  + If a client connects:
    - Loops while the client is connected (while (client.connected())):
    - Checks for data from the client (client.available()).
    - Reads data byte by byte (char c = client.read()).
    - Processes the data based on received commands:
    - GET /fanOn: Turns on the Fan (sets pin HIGH).
    - GET /fanoff: Turns off the Fan (sets pin LOW).
    - Similar logic for lightOn, lightoff, Manualoff, and ManualOn commands, including turning off Fan and Light (fanlightoff()) when switching to Manual mode.
    - If the end of the HTTP request is reached:
    - Sends an appropriate HTTP response with current mode, control buttons (enabled/disabled based on mode), and a mode switching button.
    - Closes the connection.
* **Automatic Mode (h == 0):**
  + Introduces a delay (e.g., 1 second) for stability.
  + Reads sensor data:
    - Calls readTemp() to read temperature.
    - Calls readDarkness() to check for darkness.
    - Calls readDist() to measure distance (presence detection).
  + Based on sensor readings:
    - Calls lightCtrlAuto(): Turns on light if dark and someone is present, otherwise turns it off.
    - Calls fanCtrlAuto(): Turns on fan if someone is present and temperature is high, otherwise turns it off.
* **Manual Mode (h == 1):**
  + Introduces a delay (e.g., 1 second) for stability.
  + Prints a message indicating waiting for manual control.
* **Status Reporting (for debugging):**
  + Prints the current mode (Automatic or Manual).
  + Calls lightStatus() to print the Light status (ON/OFF).
  + Calls fanStatus() to print the Fan status (ON/OFF).

## 6.2 Secondary Functions Used:

**1. fanlightoff():**

* This function turns off both the Fan and Light by setting the corresponding pins (Fanpin and Lightpin) to HIGH.

**2. sensor\_IO\_init():**

* Initializes the DHT sensor (dht) with the data pin (DHTinput) and sensor type (DHT11).
* Sets a delay of 2 seconds (delay(2000)) for sensor initialization.
* Sets the Fan (Fanpin) and Light (Lightpin) pins as outputs using pinMode.
* Calls fanlightoff() to ensure both Fan and Light are initially off.
* Sets the echo (echoPin) and trigger (trigPin) pins for the ultrasonic sensor as input and output respectively using pinMode.
* Sets the Light Dependent Resistor (LDR) input pin (LDRinput) as input using pinMode.

**3. wifi\_server\_init():**

* Prints information about connecting to the specified WiFi network (ssid) with password (password) to the serial monitor.
* Attempts to connect to the WiFi network in a loop for a maximum of 10 seconds (x = 10). It prints a dot (.) every second to indicate progress.
* If connected successfully, prints the IP address obtained and starts the server (server.begin()) with a delay of 5 seconds (delay(5000)) .
* If connection fails, prints a message indicating WiFi connection failure.

**4. readTemp():**

* Reads the temperature data from the DHT sensor using dht.readTemperature().
* Prints the temperature value along with the unit (Celsius) to the serial monitor.
* Sets a flag b0 to 1 if the temperature is greater than 32 degrees Celsius, otherwise sets it to 0.

**5. readDist():**

* Triggers the ultrasonic sensor by setting the trigger pin (trigPin) LOW for 2 microseconds, then HIGH for 10 microseconds, and finally back to LOW.
* Measures the pulse duration (duration) on the echo pin (echoPin) while it's HIGH, indicating the time taken for the sound wave to travel to the object and back.
* Calculates the distance (distance) in centimeters using the formula distance = duration / 58.2.
* Prints the distance value along with the unit (centimeters) to the serial monitor.
* Sets a flag b2 to 1 if the distance is less than 6 centimeters (object detected), otherwise sets it to 0.

**6. readDarkness():**

* Reads the state of the LDR sensor on the input pin (LDRinput).
* If the pin is HIGH (indicating darkness), sets a flag b1 to 1 and prints a message indicating darkness detected.
* Otherwise, sets b1 to 0 and prints a message indicating it's not dark.

**7. lightCtrlAuto():**

* Performs a logical AND operation on flags b1 (darkness) and b2 (object detected) to determine if light control is needed in automatic mode.
* Sets the Light pin (Lightpin) to the opposite of the combined flag value (!b3) to turn the light ON if it's dark or an object is close, and vice versa.
* Introduces a small delay (delay(100)) to avoid rapid switching.

**8. fanCtrlAuto():**

* Performs a logical AND operation on flags b2 (object detected) and b0 (high temperature) to determine if fan control is needed in automatic mode.
* Sets the Fan pin (Fanpin) to the opposite of the combined flag value (!b4) to turn the fan ON if an object is close or the temperature is high, and vice versa.
* Introduces a small delay (delay(100)) to avoid rapid switching.

**9. lightStatus():**

* Reads the state of the Light pin (Lightpin).
* Prints a message to the serial monitor indicating whether the light is ON or OFF based on the pin state.

**10. fanStatus():**

* Reads the state of the Fan pin (Fanpin).
* Prints a message to the serial monitor indicating whether the fan is ON or OFF based on the pin state.

# 7. Code

#include <DHT.h>

#include <Wire.h>

#include <WiFi.h>

#define echoPin 19

#define trigPin 21

#define LDRinput 18

#define DHTinput 15

#define Fanpin 2

#define Lightpin 4

  bool h=0;

  float temp;

  long duration, distance;

  bool b0,b1,b2,b3,b4;

  const char \*ssid = "MY\_SSID";

  const char \*password = "MY\_PASSWORD";

  int x = 10; //10 seconds time to connect to a Wifi network

DHT dht(5, DHT11);

NetworkServer server(80);

void fanlightoff(){ //Fan and light off if function called

  digitalWrite(Fanpin, HIGH);

  digitalWrite(Lightpin, HIGH);

}

void sensor\_IO\_init(){

  dht.begin();

  delay(2000);

  pinMode(Fanpin, OUTPUT); //FAN

  pinMode(Lightpin, OUTPUT); //LIGHT

  fanlightoff();

  pinMode(echoPin, INPUT);  //Ultrason i/p

  pinMode(trigPin, OUTPUT); //Ultrason o/p

  pinMode(LDRinput, INPUT); //Ldr i/p

  pinMode(DHTinput, INPUT); //DHT i/p

}

void wifi\_server\_init(){

  Serial.println();

  Serial.println();

  Serial.print("Connecting to ");

  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while(WiFi.status() != WL\_CONNECTED && x!=0) {

    delay(1000);

    Serial.print(".");

    x--;}

  if(WiFi.status() == WL\_CONNECTED){

  Serial.println("");

  Serial.println("WiFi connected.");

  Serial.println("IP address: ");

  Serial.println(WiFi.localIP());

  server.begin();

  delay(5000);}

  else{

  Serial.println("");

  Serial.println("WiFi Not connected.");}

}

void readTemp(){

  temp = dht.readTemperature();

  Serial.print("\nTemperature: ");

  Serial.print(temp);

  Serial.print(" C");

  if(temp>32)

  {b0=1;}

  else{b0=0;}

}

void readDist(){

  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);

  distance = duration / 58.2;

  Serial.print("\nDistance: ");

  Serial.print(distance);

  Serial.print(" cm");

  if(distance<6)

  {b2=1;}

  else{b2=0;}

}

void readDarkness(){

  if(digitalRead(18)==HIGH)

  {b1=1; Serial.print("\nDarkness Detected");}

  else{b1=0; Serial.print("\nIt is not Dark");}

}

void lightCtrlAuto(){

  b3 = b1 & b2;

  delay(100);

  digitalWrite(Lightpin, !b3);

}

void fanCtrlAuto(){

  b4 = b2 & b0;

  delay(100);

  digitalWrite(Fanpin, !b4);

}

void lightStatus(){

  if(digitalRead(Lightpin)==0){Serial.print("\nLight is On");}

  else{Serial.print("\nLight is Off");}

}

void fanStatus(){

  if(digitalRead(Fanpin)==0){Serial.print("\nFan is On");}

  else{Serial.print("\nFan is Off");}

}

void setup() {

  Serial.begin(115200);

  sensor\_IO\_init();

  wifi\_server\_init();

}

void loop() {

  NetworkClient client = server.accept();  // listen for incoming clients

    if (client) {                     // if you get a client,

      //Serial.print("\nNew Client.");  // print a message out the serial port

      String currentLine = "";        // make a String to hold incoming data from the client

      while (client.connected()) {    // loop while the client's connected

        if (client.available()) {     // if there's bytes to read from the client,

          char c = client.read();     // read a byte, then

          //Serial.write(c);            // print it out the serial monitor

          if (c == '\n') {            // if the byte is a newline character

            // if the current line is blank, you got two newline characters in a row.

            // that's the end of the client HTTP request, so send a response:

            if (currentLine.length() == 0) {

              // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)

              // and a content-type so the client knows what's coming, then a blank line:

              client.println("HTTP/1.1 200 OK");

              client.println("Content-type:text/html");

              client.println();

              if(h==1){

              // the content of the HTTP response follows the header:

              client.println("<HTML><title>ESP32</title>");

              client.println("<body><h1><center>ESP32 PROJECT JAYAKRISHNAN MENON</center></h1>");

              client.println("<h2><center>You are in Manual Mode</center></h2>");

              client.println("<p><center>Light Control</center></p>");

              client.println("<a href=\"/lightOn\"\"><center><button>ON</button></a> <a href=\"/lightoff\"\"><button>OFF</button></center></a>");

              client.println("<p><center>Fan Control</center></p>");

              client.println("<a href=\"/fanOn\"\"><center><button>ON</button></a> <a href=\"/fanoff\"\"><button>OFF</button></center></a>");

              client.println("<p><center>Manual Mode</center></p>");

              client.println("<center><button disabled>ON</button> <a href=\"/Manualoff\"\"><button>OFF</button></center></a>");

              client.println("</body></HTML>");}

              if(h==0){

              // the content of the HTTP response follows the header:

              client.println("<HTML><title>ESP32</title>");

              client.println("<body><h1><center>ESP32 PROJECT JAYAKRISHNAN MENON</center></h1>");

              client.println("<h2><center>You are in Auto Mode</center></h2>");

              client.println("<p><center>Light Control</center></p>");

              client.println("<center><button disabled>ON</button> <button disabled>OFF</button></center>");

              client.println("<p><center>Fan Control</center></p>");

              client.println("<center><button disabled>ON</button> <button disabled>OFF</button></center>");

              client.println("<p><center>Manual Mode</center></p>");

              client.println("<a href=\"/ManualOn\"\"><center><button>ON</button></a> <button disabled>OFF</button></center>");

              client.println("</body></HTML>");}

              // The HTTP response ends with another blank line:

              client.println();

              break; // break out of the while loop:

            } else{currentLine = "";}// if you got a newline, then clear currentLine:

          } else if (c != '\r'){currentLine += c;}// if you got anything else but a carriage return character, add it to the end of the currentLine

          // Check to see the client:

          if (currentLine.endsWith("GET /fanOn")) {

            digitalWrite(Fanpin, LOW);}

          if (currentLine.endsWith("GET /fanoff")) {

            digitalWrite(Fanpin, HIGH);}

          if (currentLine.endsWith("GET /lightOn")) {

            digitalWrite(Lightpin, LOW);}

          if (currentLine.endsWith("GET /lightoff")) {

            digitalWrite(Lightpin, HIGH);}

          if (currentLine.endsWith("GET /Manualoff")) {

            h=0;

            fanlightoff();}

          if (currentLine.endsWith("GET /ManualOn")) {

            h=1;

            fanlightoff();}

        }

      }

      // close the connection:

      client.stop();

      //Serial.print("\nClient Disconnected.");

    }

    if(h==0){

      delay(1000);

      Serial.println();

      Serial.print("\nAuto Mode");

      readTemp();

      readDarkness();

      readDist();

      lightCtrlAuto();

      lightStatus();

      fanCtrlAuto();

      fanStatus();}

    else{

      delay(1000);

      Serial.println();

      Serial.print("\nWaiting for Manual Control");

      lightStatus();

      fanStatus();}

}

# 8. Testing and Results Obtained

Having established the design and implementation of the home automation system, this section focuses on the testing procedures conducted to validate its functionalities. The system's performance in both automatic and manual modes, ensuring its ability to respond to sensor data and user inputs within the specified requirements is observed here.

## 8.1 Methods of Testing:

The testing methodology employed to assess the home automation system's functionality differed based on the operational mode - automatic or manual. This ensured a thorough evaluation of the system's response to sensor data and user commands.

For testing in automatic mode, we focused on simulating real-world conditions by manipulating the sensor inputs artificially. The DHT sensor, responsible for temperature detection, was placed near hot and cold-water bottles to mimic temperature fluctuations throughout the day. Similarly, the LDR sensor, detecting light levels, was covered with hands to simulate darkness entering the room. Finally, the HC-SR04 sensor, used for presence detection, was tested by placing hands in front of it at varying distances, mimicking human movement within the room.

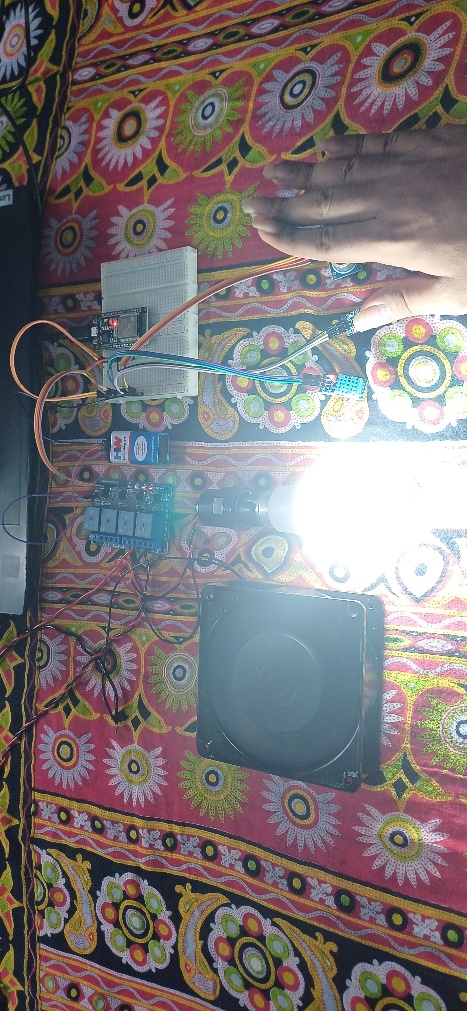
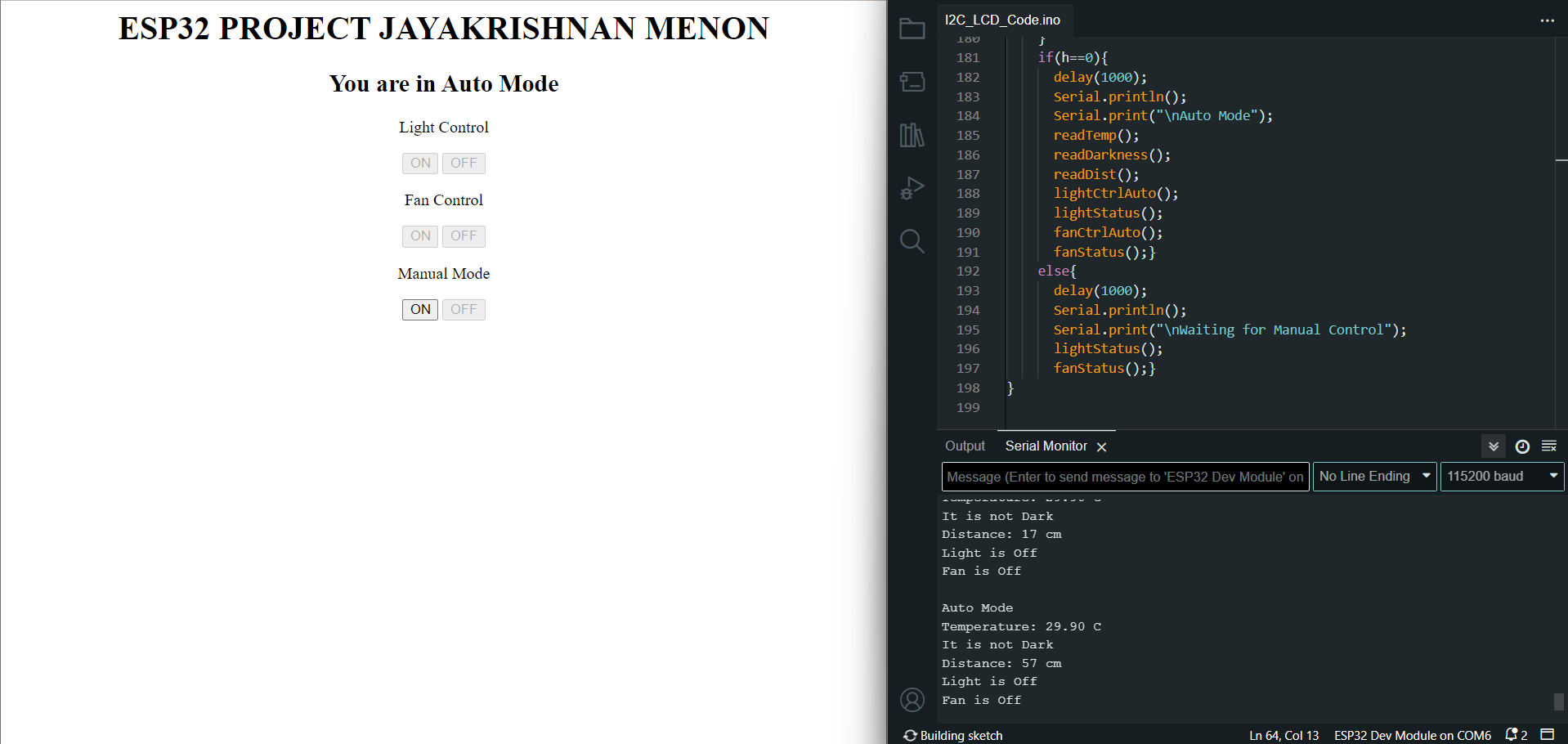
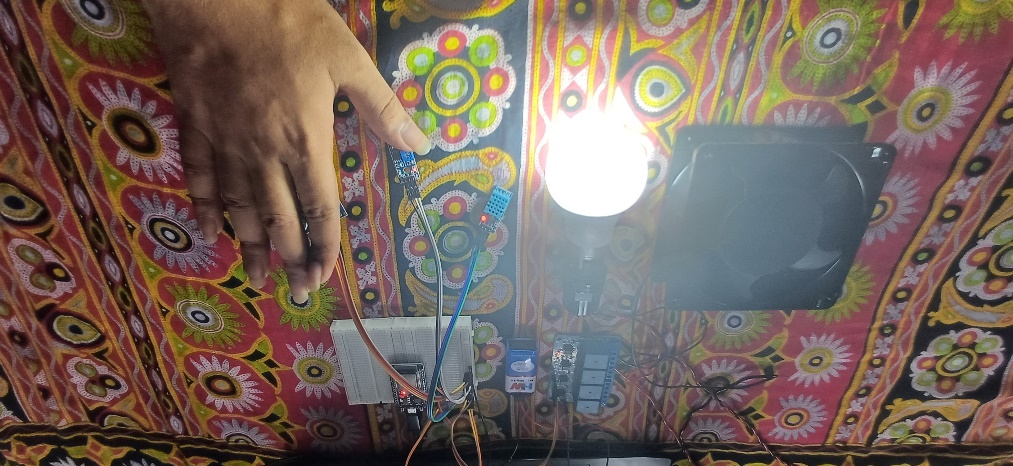
In manual mode, the focus shifted to evaluating the system's response to user commands and its ability to control the connected devices. A local webpage was accessed on a laptop, acting as the user interface for manual control of the bulb and fan. While commands were sent through the webpage, the system's response was monitored in two ways. Firstly, the output displayed on the serial monitor was observed, providing real-time system messages. Secondly, the physical state (on/off) of the bulb and fan was verified, ensuring their alignment with the user commands sent through the web interface. This two-pronged approach in manual mode testing confirmed that user input translated into the intended actions for controlling the connected devices.

## 8.2 Observation Table for Auto Mode Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Distance < 6cm | Temperature > 32C | Dark Environment | Light Status | Fan Status |
| YES | YES | YES | ON | ON |
| NO | YES | YES | OFF | OFF |
| YES | NO | YES | ON | OFF |
| NO | NO | YES | OFF | OFF |
| YES | YES | NO | OFF | ON |
| NO | YES | NO | OFF | OFF |
| YES | NO | NO | OFF | OFF |
| NO | NO | NO | OFF | OFF |

## 8.3 Relevant images from Testing and Results

Distance>6cm, Temperature<32C, No Darkness  
Light off, Fan off

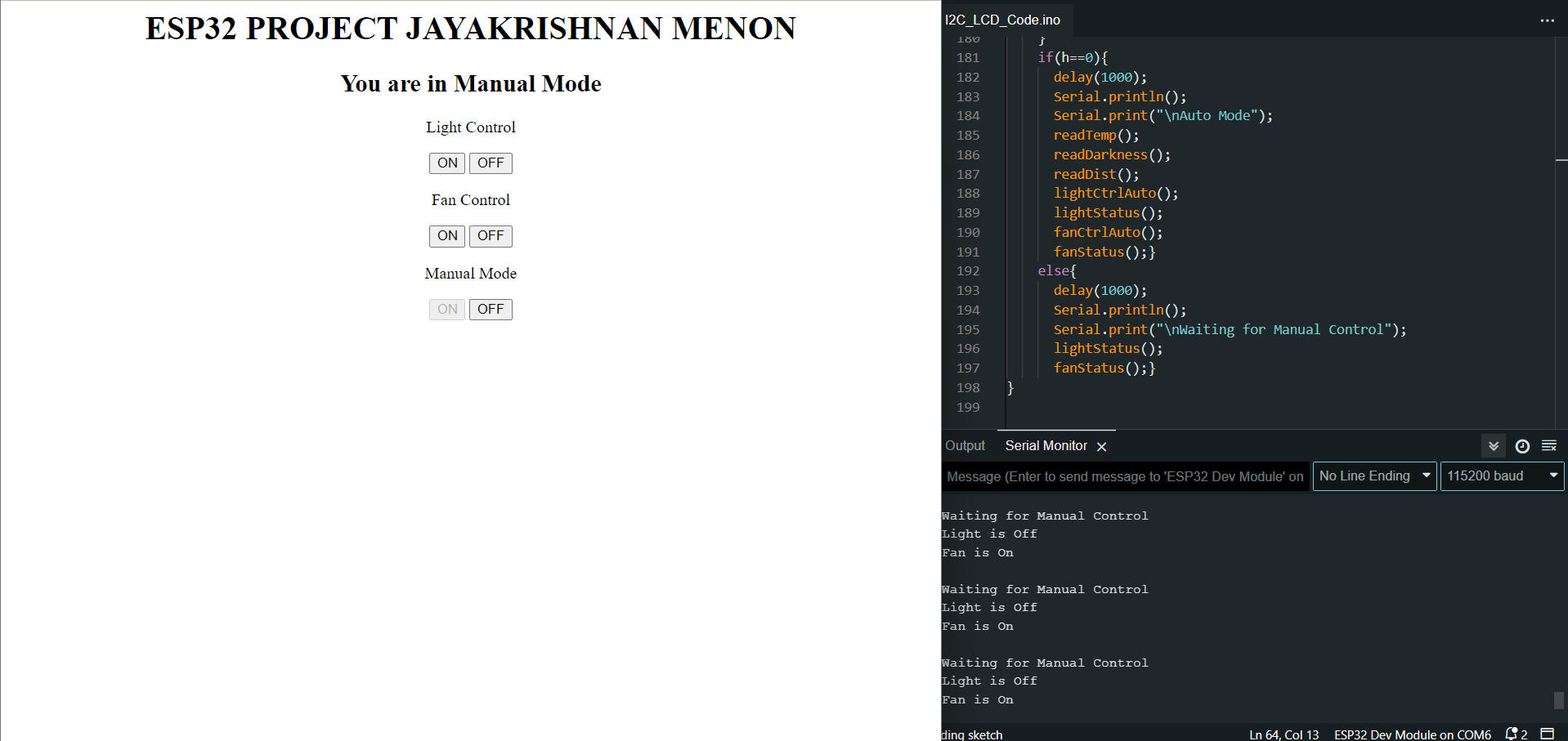


Distance<6cm, Temperature<32C, LDR covered  
Light On, Fan off

Distance<6cm, Temperature>32C, No Darkness  
Light off, Fan On

Appearance of webpage in Auto mode (Manual Light and Fan controls disabled in Auto mode)

Distance<6cm, Temperature>32C, LDR covered  
Light On, Fan On



Appearance of webpage in Manual mode (Manual Light and Fan controls enabled in Manual mode)

# 9. Conclusion

This project successfully demonstrated the implementation of a cost-effective and functional home automation system using the ESP32 microcontroller. The system effectively achieved its primary objectives of providing both automatic and manual control of household appliances through careful integration of various sensors and a user-friendly web interface.

Key achievements of the project include:

1. **Dual Mode Operation:**
   * Successfully implemented both automatic and manual control modes
   * Seamless switching between modes through the web interface
   * Reliable operation in both modes with appropriate safety controls
2. **Sensor Integration:**
   * Effective integration of multiple sensors (DHT11, HC-SR04, LDR)
   * Accurate environmental monitoring and presence detection
   * Reliable triggering of automated responses based on sensor inputs
3. **Network Connectivity:**
   * Stable Wi-Fi connection for remote device control
   * Responsive web interface for manual control
   * Secure local network implementation
4. **Performance Requirements:**
   * Met the response time requirement of device activation within 1 minute
   * Demonstrated reliable operation within the specified temperature range
   * Achieved consistent sensor readings and appropriate system responses
5. **User Interface:**
   * Created an intuitive web-based control interface
   * Implemented clear status indicators for current operating mode
   * Provided easy-to-use controls for manual operation

The testing phase validated the system's reliability and functionality across various scenarios, with the observation table demonstrating consistent and predictable behaviour in automatic mode. The system successfully handled different combinations of environmental conditions (temperature, light, and presence detection) while maintaining stable operation.

This project has established a solid foundation for home automation, demonstrating that affordable and effective smart home solutions can be implemented using readily available components. The modular nature of the design allows for future expansions and improvements, making it a valuable starting point for more sophisticated home automation applications.

# 10. Future work

This home automation system acts as a springboard for creating a truly intelligent home. Imagine a system that learns your habits and anticipates your needs. Machine learning algorithms could analyse sensor data, automatically adjusting lights and temperature for optimal comfort and energy efficiency. Voice assistants like Alexa or Google Assistant could allow you to control devices with just your voice, offering a truly hands-free experience. Advanced sensors could take automation a step further. Motion detectors could ensure lights turn on only when someone's present, and CO2 sensors could monitor air quality, triggering ventilation when necessary.

This system can also become part of a larger connected ecosystem. Integration with smart home platforms like SmartThings or IFTTT would allow your home automation system to interact with other smart devices. Imagine a scenario, where turning on the lights in the morning automatically triggers the coffee maker to brew a fresh pot, or the thermostat adjusts based on real-time weather data. Enhancements like fingerprint recognition could add a layer of security, ensuring only authorized users can control the system or switch modes.

Finally, the possibilities for remote management and advanced functionalities expand significantly by connecting the system to IoT platforms. Imagine monitoring your home environment remotely, analysing sensor data for insights, or even integrating cloud-based services for features like facial recognition or biometric monitoring to personalize the environment based on your health and well-being. By exploring these advancements, this project can evolve into a comprehensive and intelligent home automation system that seamlessly integrates with your life, offering unparalleled convenience, security, and personalized comfort.