MAVEN SILICON

PROJECT TITLE :

HOME AUTOMATION

STUDENT INFORMATION

NAME: HEMA HARINI G

UNIVERSITY: VELLORE INSTITUTE OF TECHNOLOGY, VELLORE

DATE OF SUBMISSION: 11/07/2024

INTRODUCTION

Embedded systems are specialized computing systems that carry out specific activities within broader mechanical or electrical systems. They are essential for current home automation due to their small size, low power consumption, and real-time processing capabilities. We can accomplish seamless automation and control by incorporating embedded components such as microcontrollers and sensors into home appliances and systems.

The goal of this project is to create and execute a comprehensive Home Automation system that will allow users to remotely monitor and control many aspects of their home environment. This includes automatic lighting, temperature control, appliance management. The system is intended to be user-friendly, scalable, and adaptable to various residential environments.

Scenario Application

1.Automated Lighting Control

Automation of lighting is a major factor in increasing energy efficiency and convenience in a smart home setting. Using motion and light sensors, our home automation system automatically modifies the lighting according to occupancy and ambient light levels. For example, the motion sensor in a room recognizes the presence of a person and turns on the lights. In contrast, the lights are switched off to save energy when the room is empty. To ensure ideal illumination at all times, it can be improvised with light sensors that also modify the lights' intensity in response to the availability of natural light.

2.Automated fan keeping Temperature in Control

Keeping the inside temperature at a comfortable level is necessary for a comfortable living space. To control heating and cooling systems, our system incorporates temperature sensors. The Fan will automatically adapt to maintain the desired temperature ranges that users have chosen. This not only guarantees comfort but also contributes to energy conservation by preventing needless heating or cooling of the house while it is unoccupied

3. Automated and Manual mode

Our home automation system's remote control and environment monitoring capabilities are among its main benefits. Users can access all system functionalities from their smartphones or tablets by using a dedicated mobile application. This includes monitoring live video feeds, controlling the temperature, turning on and off lights, and getting security warnings. With remote access, homeowners may control their home environment from anywhere in the world, which adds convenience and peace of mind.

Functional 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 WiFi network.

• The fan and bulb are controlled by a mobile connected to home WiFi network.

• Auto or mobile control mode can be selected on the mobile phone.

Non-Functional requirements

Performance requirements:

Automatic mode:

- bulb should switch on after the entry of human in the room within 1 minute.
- Fan should switch on as soon as the threshold is detected within 1 minute.

Manual control mode:

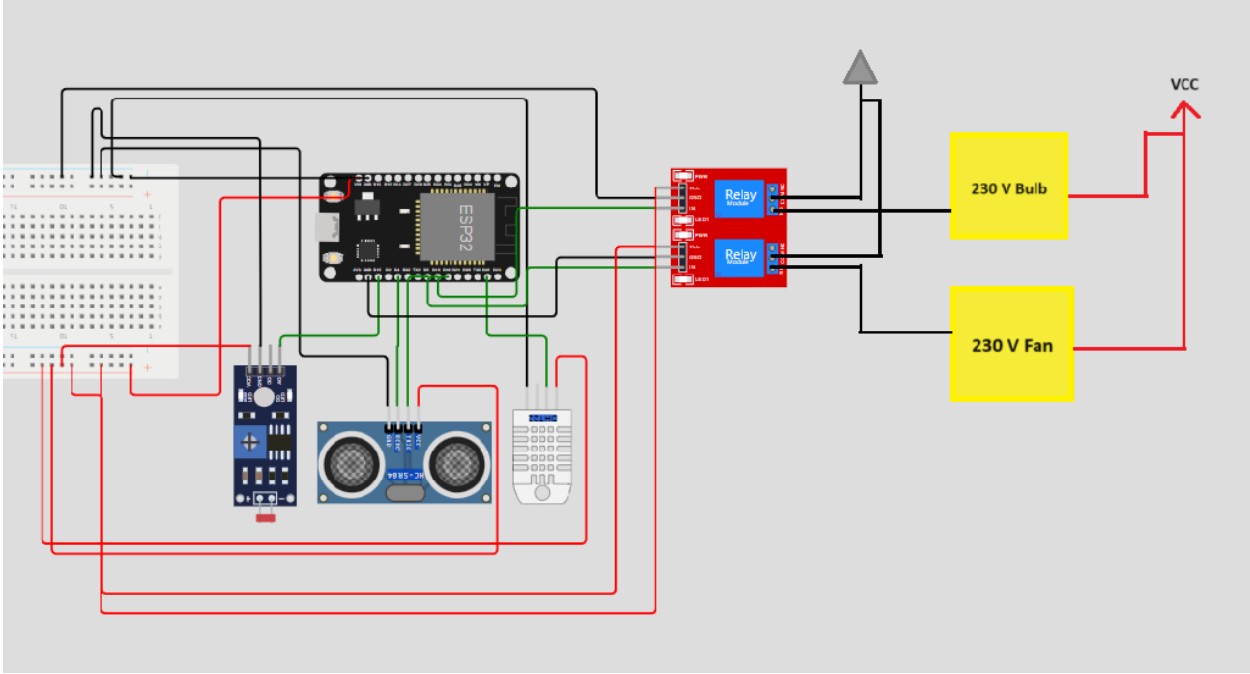
- The bulb should switch on within 1 minute of pressing the button on the mobile phone.
- Fan should switch on within 1 minute of pressing the button on the mobile phone.

Environment requirements:

- The system should work in the temperature range 0 degree centigrade to 60 degree centigrade.

HARDWARE ARCHITECTURE:

Schematic :



The selection of components for the Home Automation project was driven by the need to create a versatile, reliable, and efficient system capable of managing various aspects of a modern home environment. Below is an explanation of each component chosen and the rationale behind their selection:

ESP32 Microcontroller: The ESP32 microcontroller was chosen as the central processing unit for this project due to its capabilities and versatility. It offers integrated Wi-Fi and Bluetooth connectivity, making it ideal for remote monitoring

and control. Its low power consumption and high processing power enable efficient handling of multiple sensors and actuators.

DHT11 Sensor Module (Temperature and Humidity Sensor)

The DHT11 sensor module was selected to monitor temperature and humidity levels within the home. This sensor is known for its ease of use, reliability, and affordability. It provides digital output, simplifying the integration with the ESP32 microcontroller. The data from the DHT11 sensor enables the system to maintain optimal climate conditions by controlling fan based on real-time environmental readings.

LDR Sensor Module (Light Dependent Resistor Sensor)

The LDR sensor module was chosen to manage the lighting system based on ambient light levels. This sensor is highly sensitive to light and provides analog output that can be easily read by the ESP32. By incorporating the LDR sensor, the system can automatically adjust indoor lighting based on natural light availability, ensuring energy efficiency and maintaining appropriate illumination levels.

Ultrasonic Sensor

The ultrasonic sensor was chosen for its ability to detect motion and measure distance accurately. This sensor operates by emitting ultrasonic waves and measuring the time it takes for the waves to return after hitting an object. It is used in the project to detect the presence of individuals in a room, enabling automated lighting. The ultrasonic sensor's reliability in various lighting conditions and its non-intrusive nature make it an ideal choice for people detection.

2-Channel Relay Module

The 2-channel relay module was selected to control high-voltage appliances such as the 230V bulb and the DC fan. This module allows the microcontroller to safely switch these devices on and off. The use of relays ensures that the low-power signals from the ESP32 can effectively manage high-power devices without any risk of damage to the microcontroller or the appliances.

To ensure all requirements were met, I adopted a systematic approach.

I first connected each of my sensor modules and other components and tested their individual working coding them separately ensuring their proper stand alone working.

Then to ensure combined working I integrated a 2-channel relay module with the ESP32 microcontroller to control a 230V light bulb and fan, verified reliable WiFi connectivity, and developed a html webpage for manual control. The webpage facilitated seamless switching between auto and manual modes. Performance requirements were addressed by calibrating the ultrasonic sensor to activate the bulb within 1 minute of detecting human presence and configuring the DHT11 sensor to activate the fan within 1 minute of threshold detection. Environmental requirements were met by selecting components capable of operating between 0°C and 60°C, validated through testing requirements

List of Hardware components used:

S.NO.	HARDWARE DESCRIPTION	QUANTITY
1	ESP32 (Microcontroller)	1
2	DHT11 sensor Module (Temperature and Humidity sensor).	1
3	LDR sensor Module (Light Dependent Resistor sensor)	1
4	2 – Channel Relay module	1
5	Ultrasonic sensor	1
6	230 V Bulb	1
7	DC Fan	1
8	Breadboard	1
9	Jumper wires	
10	230 V power supply plug	

SOFTWARE ARCHITECTURE

Software used : ARDUINO IDE

Functional Blocks and their functionality

I went about dividing the code into different blocks each with specific functions making it more easier to check debug. To measure and calibrate various parameters required for the functioning of the system I created individual setup functions which are then called upon whenever required.

1.Header Inclusions and Pin Definitions :

This block includes the necessary libraries for DHT sensor communication and WiFi connectivity. It also defines the pins for the DHT11 sensor, ultrasonic sensor, LDR sensor, relays for the light and fan, and the DHT sensor type.

2. DHT Sensor Setup and Temperature Reading:

Initializes the DHT11 sensor and defines the getTemp() function to read and return the current temperature. This temperature data is used for controlling the fan.

3. LDR Sensor Setup and Light Reading:

Configures the LDR sensor to read ambient light levels. The getLight() function reads the light intensity which is used to decide whether to turn on or off the light bulb.

5. Relay Setup for Appliances:

Configures the pins connected to the relays for controlling the light bulb and fan. This setup allows the microcontroller to switch these appliances on or off.

6. WiFi Connectivity Setup:

Connects the ESP32 to the specified WiFi network and starts an HTTP server. The server listens for incoming client requests and serves the web-based control interface. It gives out the IP address of the system through which we can access the html webpage that we have created for controlling.

7. **Automatic Mode Control:**

Implements the automatic mode where the system controls the light and fan based on sensor data. The light is turned on if ambient light is low, and the fan is activated if the temperature is below the threshold. If no one is detected (distance > 6 inches), both appliances are turned off.

8. **Setup Function**:

Initializes serial communication, sets up all sensors, relays, and WiFi, and prepares the system for operation.

9. **Main Loop for Manual and Automatic Control**:

This loop handles incoming HTTP requests for both manual and automatic modes. It processes commands to turn the light and fan on or off and switches between manual and automatic modes based on user input. In automatic mode, it continuously checks sensor data and adjusts appliances accordingly.

Pseudocode :

Include the required Libraries

Define Pins

-Set pins for DHT sensor, ultrasonic sensor, LDR sensor, light bulb, and fan

Define WiFi Credentials

- SSID and password for the WiFi network

Initialize:

- Set up DHT sensor
- Set up LDR sensor pin
- Set up ultrasonic sensor pins
- Set up relay pins for light bulb and fan
- Connect to WiFi and start HTTP server

Function setup():

Initialize Serial Communication

```
    Call setup functions for sensors and relays
    Call WiFi setup function
Function loop():
    Check for incoming HTTP requests from the client
    If a request is received:
        Read the request
        Process commands for manual control (light and fan)
        Process commands for changing modes (Manual/Auto)
        Send HTML response with control options to the client
    If in Auto Mode:
        Call Automode() to control appliances based on sensor data
    Else:
        Display "Manualmode" message
Function DHTsetup():
    Initialize DHT sensor
Function getTemp():
    Read temperature from DHT sensor
    Print temperature
    Return temperature
Function LDRsetup():
    Set LDR pin as INPUT
Function getLight():
    Read light level from LDR sensor
    Return light level
Function UltrasonicSensorsetup():
```

```
    Set trigger pin as OUTPUT
    Set echo pin as INPUT
Function getdist():
    Send pulse to ultrasonic sensor
    Measure pulse duration
    Calculate distance
    Print distance
    Return distance
Function relaySetup():
    Set pins for light bulb and fan as OUTPUT
Function wifiSetup():
    Connect to WiFi network using SSID and password
    Wait for connection
    Print local IP address
    Start HTTP server
Function Automode():
    Get temperature, light level, and distance
    If distance <= 6 inches:
        If light level < 800:
            Turn light ON
        Else:
            Turn light OFF
        If temperature >= 20°C:
            Turn fan OFF
        Else:
```

Turn fan ON

Else:

Turn both light and fan OFF

Function ManualControl():

Process commands from the webpage

- Turn light or fan ON/OFF

- Switch between Manual and Auto modes

CODE:

```
#include "DHT.h"
#include <WiFi.h>
#include <WiFiClient.h>

#define dhtPin 4
#define trigPin 17
#define echoPin 16
#define ldrPin 22
#define led 15
#define fan 5
#define DHTTYPE DHT11

const char* ssid = "vivo";
const char* password = "hemaharini";
WiFiServer server(80);

DHT dht(dhtPin,DHTTYPE);

void DHTsetup()
{
Serial.println("DHTxx test!");
dht.begin();
}
float getTemp()
{
 float temp = dht.readTemperature();
 Serial.print("Temperature: ");
 Serial.print(temp);
 Serial.println("C");
 return temp;
}
```

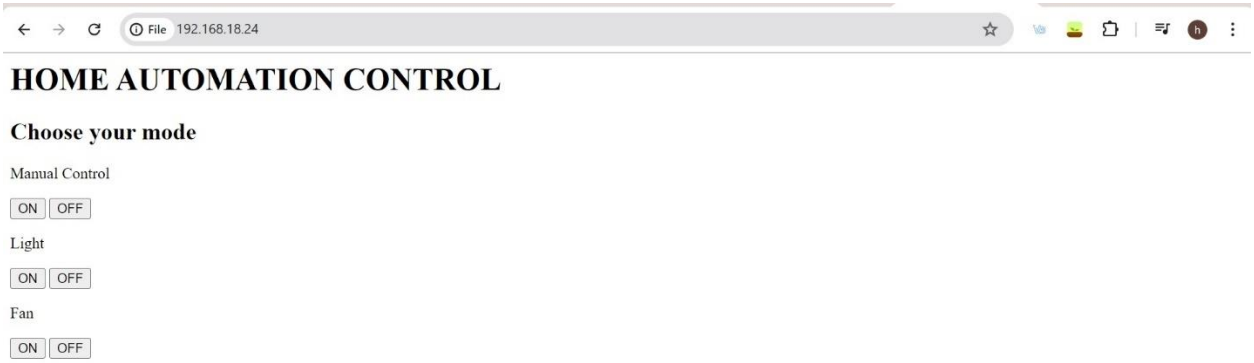
```
float LDRsetup()
{ pinMode(ldrPin,INPUT);
}
float getLight(){
  int light = analogRead(ldrPin);
  return light;
}

float UltrasonicSensorsetup(){
  pinMode(trigPin,OUTPUT);
  pinMode(echoPin,INPUT);
}
float getdist(){
 digitalWrite(trigPin,LOW);
vTaskDelay(1000);

digitalWrite(trigPin,HIGH);
vTaskDelay(1000);
digitalWrite(trigPin,LOW);

float time=pulseIn(echoPin,HIGH);
float dist = time*0.034/2;
float inch= dist*39.37;
Serial.print("Distance in cm: ");
Serial.println("dist");
Serial.print("Distance in inch: ");
Serial.println("inch");
return inch;
}

void relaySetup(){
  pinMode(led,OUTPUT);
  pinMode(fan,OUTPUT);
}

 void wifiSetup()
{
  Serial.print("Connecting to ");
  Serial.print(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED)
  {
    Automode();
```

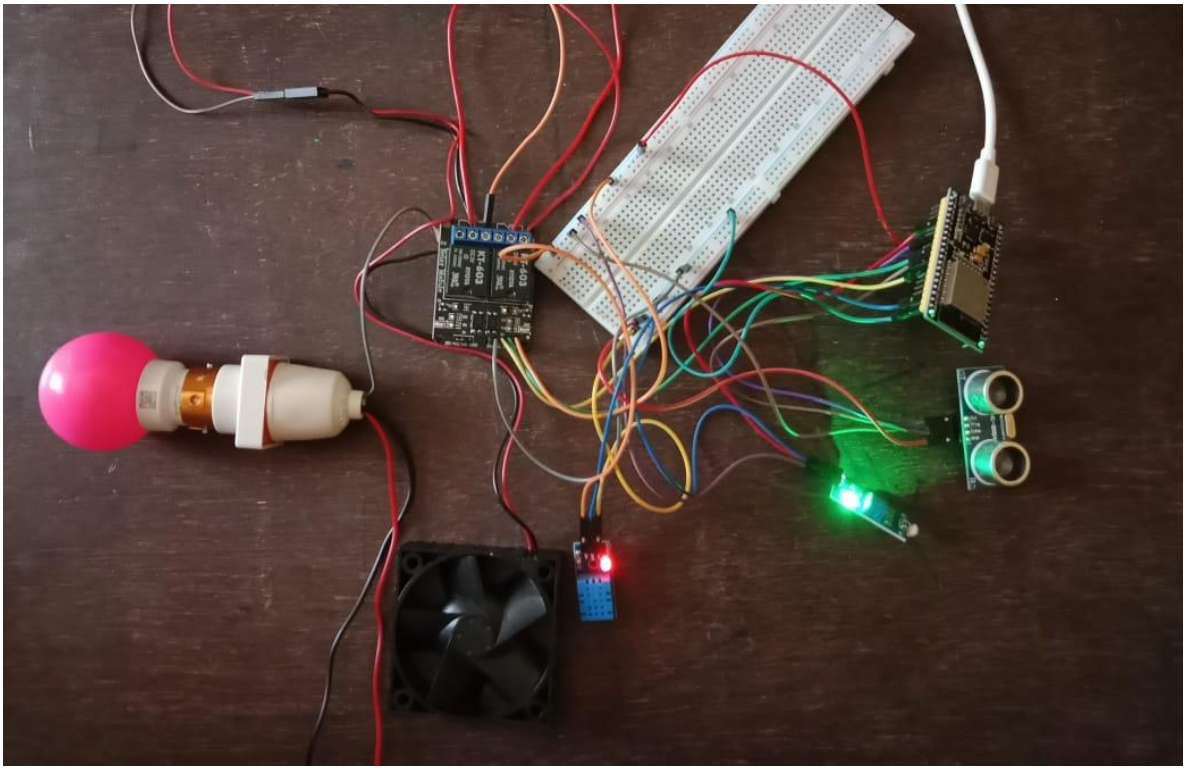
```
    vTaskDelay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected..!");
  Serial.print("Got IP: ");
  Serial.println(WiFi.localIP());

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

void Automode(){
  float temp = getTemp();
  float light = getLight();
  float inch = getdist();

  if(inch<=6)
{
   if (light < 800) {
      digitalWrite(led,HIGH);
      }
   else {
      digitalWrite(led,LOW);
      }
   if (temp >= 20 ) {
      digitalWrite(fan,LOW);
      }
   else{
      digitalWrite(fan, HIGH);
      }
}
else {
  digitalWrite(led, LOW);
  digitalWrite(fan,LOW);
   }
}

void setup()
{
  Serial.begin(9600);
  DHTsetup();
  LDRsetup();
  UltrasonicSensorsetup();
  relaySetup();
```

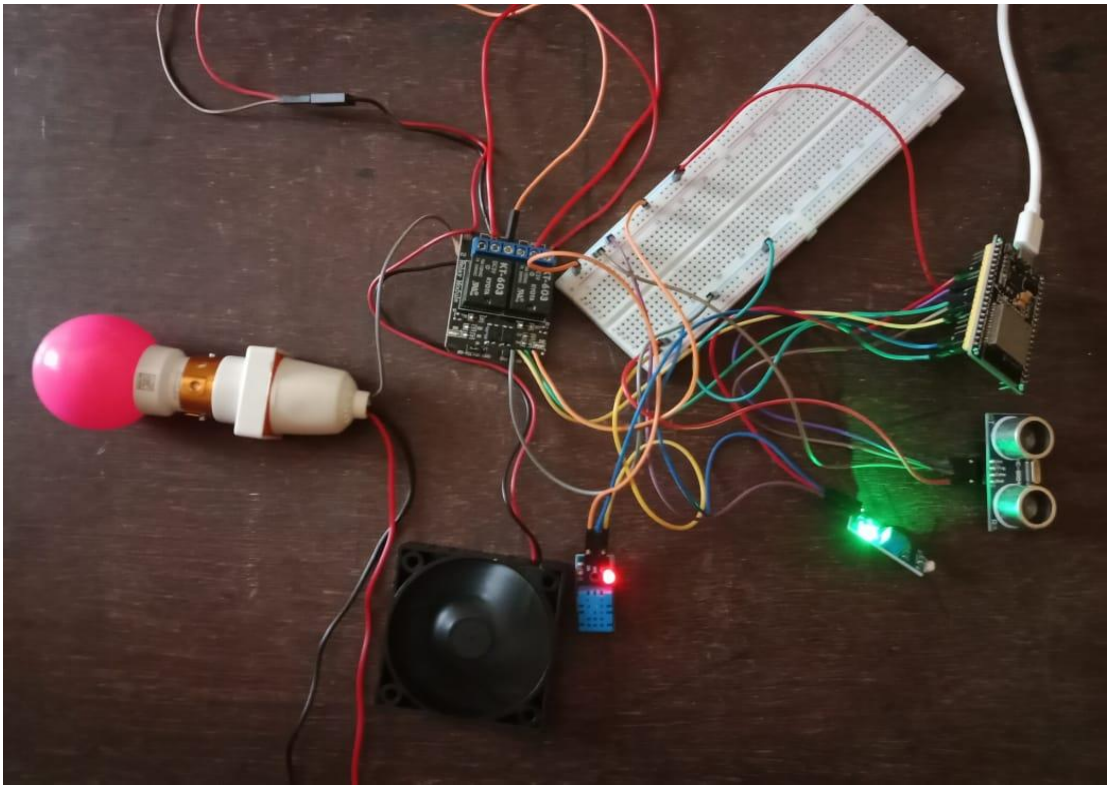
```
  wifiSetup();
}

bool Manualmode = false;

void loop()
{
  WiFiClient client = server.available();
  if (client)
  {
    bool currentlineisBlank = true;
    String buffer = "";
    while(client.connected())
    {
      if (client.available())
      {
        char c = client.read();
        buffer += c;
        if (c == '\n' && currentlineisBlank)
        {
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println();
          client.println("<HTML><title>ESP32</title>");
          client.println("<body><h1>HOME AUTOMATION CONTROL</h1>");
          client.println("<body><h1>Choose your mode</h1>");

          client.println("<p>Manual Control</p>");
          client.println("<a href=\?manualOn\"\"><button>On</button></a>");
          client.println("<a href=\?manualOff\"\"><button>Off</button></a>");

          client.println("<p>Light</p>");
          client.println("<a href=\?lightOn\"\"><button>On</button></a>");
          client.println("<a href=\?lightOff\"\"><button>Off</button></a>");

          client.println("<p>Fan</p>");
          client.println("<a href=\?fanOn\"\"><button>On</button></a>");
          client.println("<a href=\?fanOff\"\"><button>Off</button></a>");

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

        else if (c=='\n')
        {
```

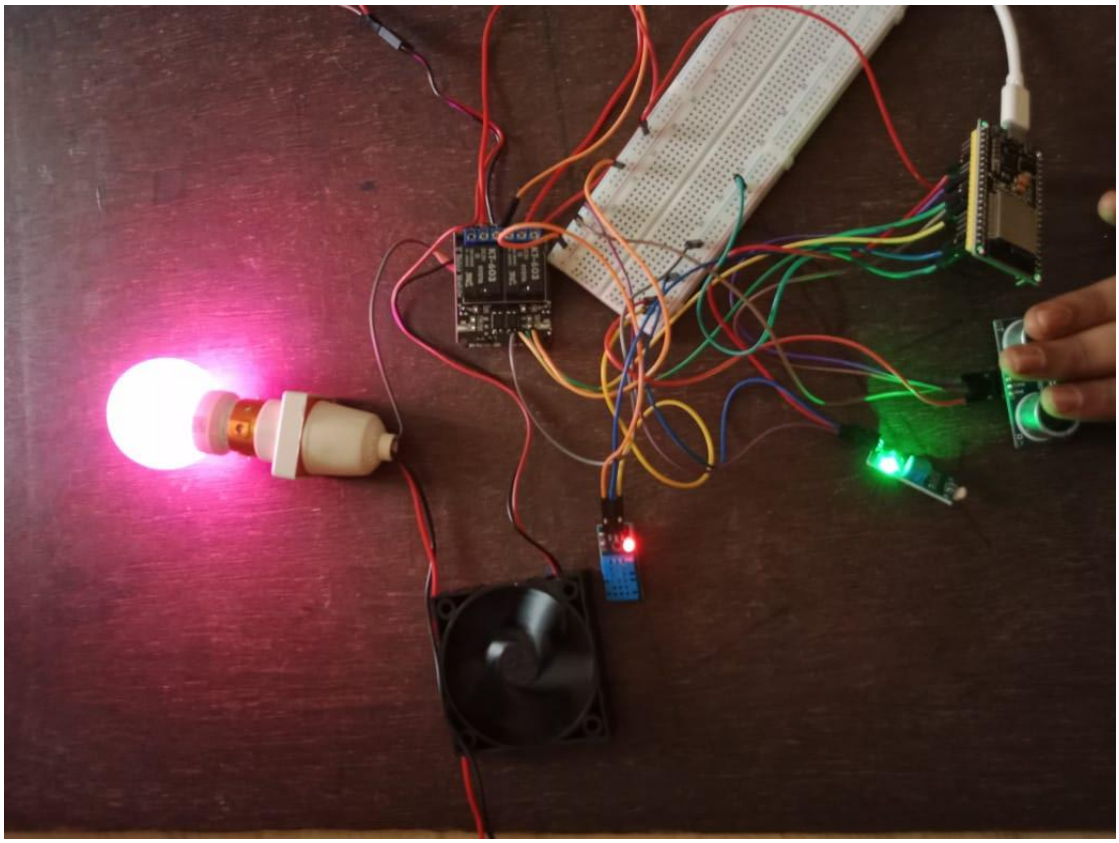
```
          currentlineisBlank = true;
          buffer = "";
        }
        else if (c == '\r');
        {
          if(buffer.indexOf("GET /?lightOff")>=0)
           digitalWrite(led, LOW);
          if(buffer.indexOf("GET /?lightOn")>=0)
           digitalWrite(led,HIGH);

          if(buffer.indexOf("GET /?fanOff")>=0)
           digitalWrite(fan,LOW);
          if(buffer.indexOf("GET /?fanOn")>=0)
           digitalWrite(fan,HIGH);

          if(buffer.indexOf("GET /?manualOn")>=0)
          {
           Manualmode = true;
           Serial.print("Manualmode: ");
           Serial.print(Manualmode);
          }
          if(buffer.indexOf("GET /?manualOff")>=0)
          {
           Manualmode = false;
           Serial.print("Manualmode: ");
           Serial.print(Manualmode);
          }
        }
      }
    }
}
else
{
   bool currentlineisBlank = false;
}
client.stop();
vTaskDelay(2000);
if(Manualmode == false)
{
  Serial.println("Automode");
  Automode();
}
else
{
  Serial.println("Manualmode"); } }
```

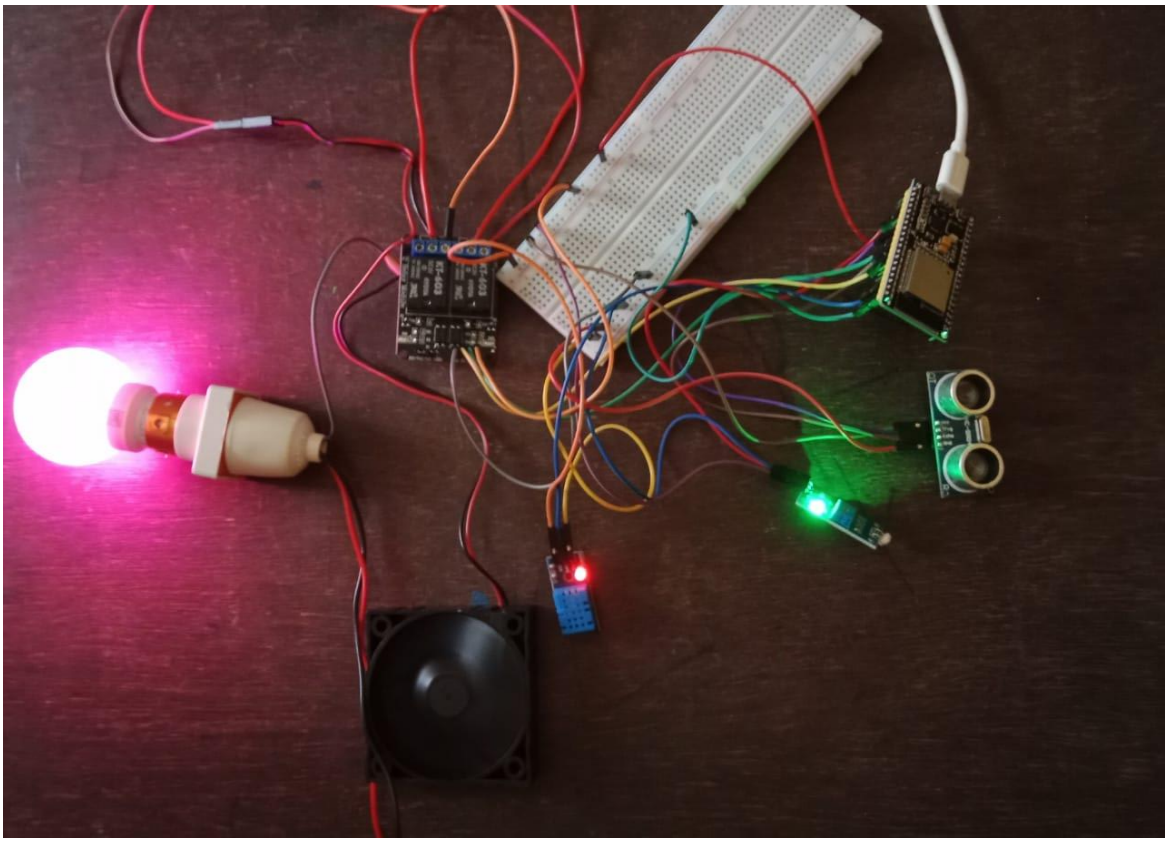
TESTING AND RESULTS











Testing Procedure:

The system was finally tested with various work cases based on the functional and non functional requirements mentioned at the start of the report.

Accordingly we have 2 modes of control

- Automode
- Manualmode

Default the sytem is set to automode, so the system started with automode when power supply was given initially and the code was run.

At the start the system established the wifi setup and provided us the IP address of the system which is later used to open the webpage.

- Automatic Control:

The light bulb turned on within 1 minute when the LDR sensor detected low light levels and a human presence was detected by the ultrasonic sensor

The fan turned on within 1 minute when the temperature read by the DHT11 sensor exceeded the specified threshold and a human presence was detected by the ultrasonic sensor.

- Manual Control:

The light bulb successfully turned on and off within 1 minute of pressing the corresponding button on the mobile application.

The fan turned on and off within 1 minute of pressing the corresponding button on the mobile application.

Future Improvements

While the current Home Automation that I have worked with during this internship project is just a small prototype integrating only 2 appliances Fan, light it can be improvised with adding other home appliances. The project can be broadened by using more well equipped industry level sensors along with booming technologies like IOT to make it more efficient and smart.

Below are some potential future improvements:

1. By enabling voice commands to operate household appliances, integrating the system with well-known voice assistants like Apple Siri, Google Assistant, or Amazon Alexa could improve customer convenience.
2. Including energy monitoring features can give consumers access to real-time data on how much energy they use.
3. Adding extra features to the mobile application, such notifications, custom scenes, and scheduling, will enhance user experience and offer more control possibilities.

4. By adding more sensors to the system—such as smoke, CO2, and motion detectors—it will be able to monitor and react to a wider range of environmental situations, hence increasing overall house safety.
