Home Automation Project Report

TABLE OF CONTENTS

1. STUDENT INFORMATION

- **Name:** CHALLA GNAN PAVAN
- **University Serial Number:** 23H75A0406
- **Institution:** DVR & Dr.HS MIC COLLEGE OF TECHNOLOGY

  (kanchikacherla – 521180, NTR Dist., AP, india)
- **Submission Date:** 30 JUN 2025

2. INTRODUCTION

Home automation systems are transforming modern living by offering enhanced convenience, energy efficiency, and security. This project focuses on designing and implementing a cost-effective home automation system using widely available hardware components. The system automates the control of a fan and a lamp based on real-time environmental data, such as temperature, humidity, light levels, and presence detection, making it an ideal solution for small-scale smart home applications.

Motivation:

The rise of smart home technologies and the affordability of microcontrollers like the ESP32 motivated this project. The aim was to create a practical system that demonstrates the potential of IoT in automating daily tasks while remaining accessible to hobbyists and students.

Objectives:

- Automatically control a fan based on temperature thresholds.
- Manage a lamp based on ambient light levels and room occupancy.
- Monitor environmental conditions in real-time.
- Integrate multiple sensors and actuators into a cohesive automation system.

3. PROJECT SPECIFICATIONS

3.1 FUNCTIONAL REQUIREMENTS

- **Temperature-based Fan Control:** The system must turn on the fan when the temperature exceeds 28°C and turn it off when it falls below this threshold.
- **Light-based Lamp Control:** The lamp should activate when ambient light levels drop below a set threshold (e.g., in dark conditions).
- **Presence Detection:** The system should detect a person within 50 cm and turn on the lamp in low-light conditions if presence is detected.
- **Environmental Monitoring:** The system should continuously monitor temperature, humidity, light levels, and distance, with the ability to display or log these values.

3.2 NON-FUNCTIONAL REQUIREMENTS

- **Reliability:** The system should function consistently with minimal errors or sensor malfunctions.
- **Response Time:** Actions (e.g., turning on the fan or lamp) should occur within 2 seconds of detecting a change in conditions.
- **Power Efficiency:** Total power consumption should not exceed 10W during normal operation.
- **Ease of Installation:** The system should be simple to assemble and configure using common tools and software.
- **Scalability:** The design should support future enhancements, such as additional sensors or remote access features.

4. HARDWARE ARCHITECTURE

The hardware architecture integrates multiple sensors and actuators, controlled by the ESP32 microcontroller, to create an efficient automation system.

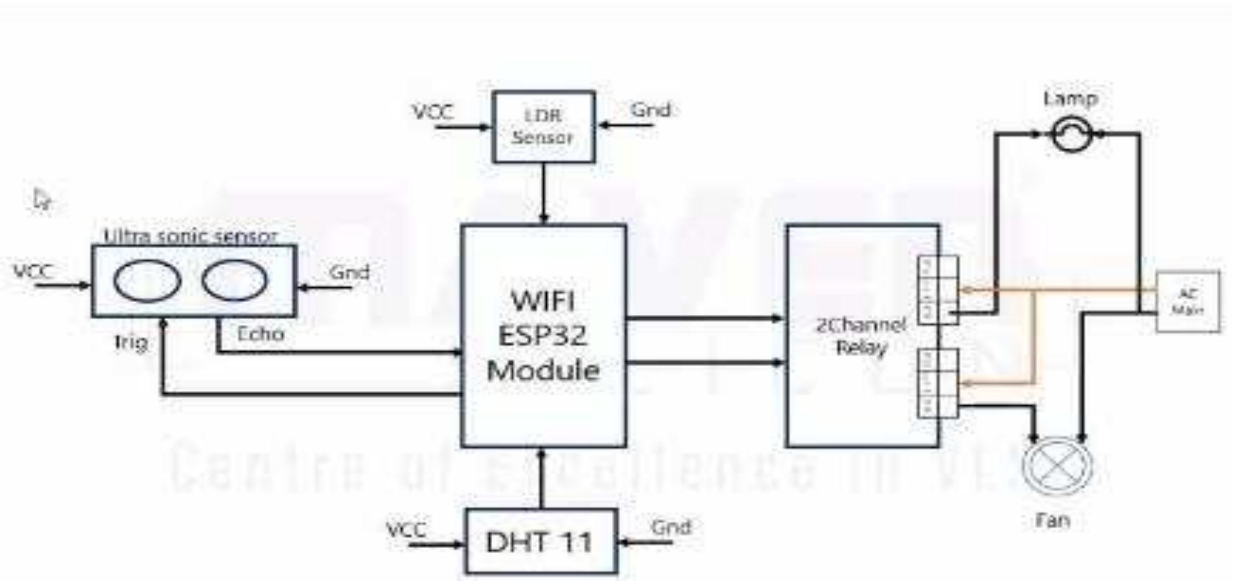
4.1 LIST OF HARDWARE COMPONENTS USED

- **ESP32 Controller:**
  - **Description:** A powerful dual-core microcontroller with Wi-Fi and Bluetooth capabilities.
  - **Role:** Acts as the central processing unit, collecting sensor data and controlling the fan and lamp via the relay.
  - **Specifications:** 32-bit, 240 MHz, 520 KB SRAM.
- **DHT11 Sensor:**
  - **Description:** A digital sensor for temperature and humidity measurement.
  - **Role:** Monitors room climate to trigger fan operation.
  - **Specifications:** Temperature: 0-50°C (±2°C); Humidity: 20-80% (±5%).
- **LDR Sensor (Light-Dependent Resistor):**
  - **Description:** A light-sensitive resistor used in a voltage divider circuit.
  - **Role:** Detects ambient light levels to control the lamp.
  - **Specifications:** Resistance varies inversely with light intensity.
- **Ultrasonic Sensor (HC-SR04):**
  - **Description:** A sensor that uses sound waves to measure distance.
  - **Role:** Detects presence within a specified range for lamp control.
  - **Specifications:** Range: 2-400 cm; Accuracy: ±3 mm.
- **2-Channel Relay Module:**

    - **Description:** An electrically operated switch for controlling high-voltage devices.
    - **Role:** Switches the fan and lamp on or off based on ESP32 signals.
    - **Specifications:** 5V control, 10A/250V AC switching capacity.

- **Fan:**
    - **Description:** A small cooling device (DC for demonstration).
    - **Role:** Activates to reduce temperature when thresholds are exceeded.
    - **Specifications:** 5V DC (AC fan could be used with relay in practice).
- **Lamp:**
    - **Description:** A light source (DC for demonstration).
    - **Role:** Provides illumination in low-light or occupancy scenarios.
    - **Specifications:** 5V DC (AC lamp could be used with relay in practice).

Block Diagram:



5. SOFTWARE ARCHITECTURE

The software is developed using the Arduino IDE, with the ESP32 programmed in C++. It features a modular design with distinct components for sensor reading, decision-making, and output control.

- **Main Components:**
  - **Initialization:** Configures pins and sensors at startup.
  - **Sensor Reading:** Functions to collect data from DHT11, LDR, and ultrasonic sensors.
  - **Control Logic:** Decision-making based on sensor thresholds.
  - **Output Management:** Controls relay channels for the fan and lamp.
- **Control Flow:**

1. Read temperature and humidity from DHT11.
2. Measure light level using LDR.
3. Calculate distance with the ultrasonic sensor.
4. Apply logic to activate/deactivate the fan and lamp.
5. Repeat every 2 seconds.

Pseudo Code For Home Automation System:

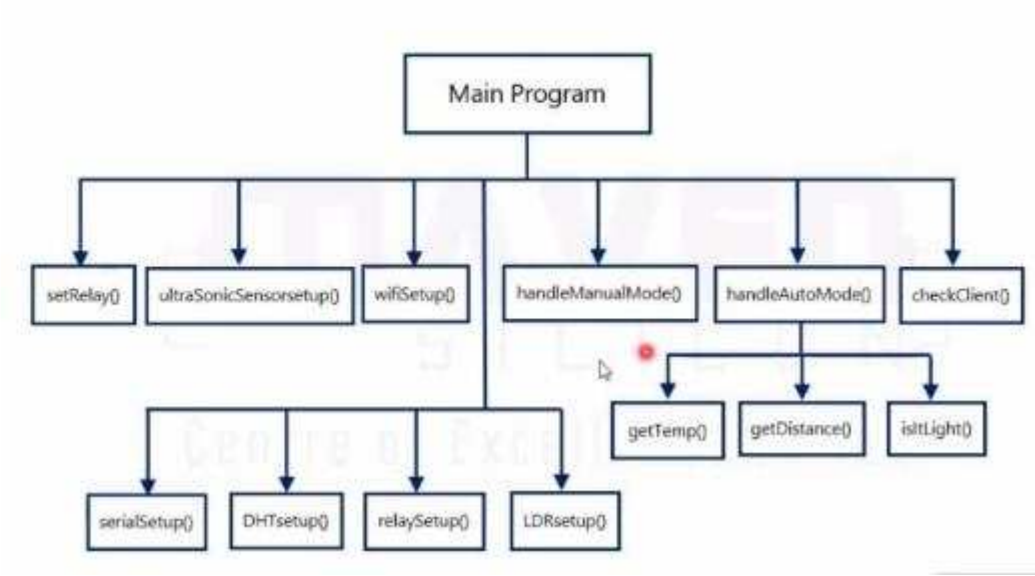
```
DEFINE constants and pins:
    DHT sensor pin and type
    LDR pin
    Ultrasonic trig and echo pins
    Relay pins
    WiFi credentials
```

```
DECLARE global variables:
    Server at port 80
    Relay previous states
    Flags for manual mode

FUNCTION SerialSetup:
    Start serial communication at 115200 baud rate
    Print startup message

FUNCTION DHTsetup:
    Print test message
    Initialize DHT sensor

FUNCTION getTemp:
    Wait for sensor reading delay
    Read humidity and temperature from DHT sensor
    IF any reading fails:
        Print error
        RETURN 0
    ELSE:
        Compute heat index
        Print temperature in Celsius
        RETURN temperature
```

```
FUNCTION LDRsetup:
    Set LDR pin as INPUT

FUNCTION isitLight:
    Read digital value from LDR
    IF HIGH:
        Print "it is dark"
        RETURN false
    ELSE:
        Print "it is light"
        RETURN true

FUNCTION ultraSonicSensorsetup:
    Set trig pin as OUTPUT
    Set echo pin as INPUT

FUNCTION getDistance:
    Trigger ultrasonic pulse
    Measure echo duration
    Calculate distance in cm
    Print distance
    RETURN distance

FUNCTION relaySetup:
    Set relay pins as OUTPUT
    Initialize relays to OFF state

FUNCTION setRelay(relay_number, value):
    IF relay_number is for Light:
        Print Light Relay state
        Invert logic for relay (LOW=ON)
        IF state changed:
            Write to relay pin
        Update previous state
    IF relay_number is for Fan:
        Print Fan Relay state
        Invert logic
        IF state changed:
            Write to relay pin
        Update previous state

FUNCTION WiFiSetup:
    Print WiFi connection message
    Connect to WiFi
    WHILE not connected:
        Run AutoMode logic
        Wait and print dot
    Print IP address
    Start WiFi server

FUNCTION handleAutoMode:
    Read distance, temperature, and light status
    IF person is detected (distance <= 10):
        IF temperature >= 20:
            Turn ON fan
        ELSE:
            Turn OFF fan
        IF it is light (day):
            Turn OFF light
        ELSE:
            Turn ON light
    ELSE:
        Turn OFF both light and fan

FUNCTION setup:
    Call SerialSetup
    Call DHTsetup
    Call LDRsetup
    Call ultraSonicSensorsetup
    Call relaySetup
    Call WiFiSetup
```

```
LOOP forever:
    WAIT for client request via WiFi
    IF client connects:
        Read data from client request
        IF request ends (new line detected):
            Send HTML page to control relays manually
            BREAK out of request handling loop
        IF request contains specific keywords:
            Handle light ON/OFF
            Handle fan ON/OFF
            Handle manual mode ON/OFF
    END client session

    WAIT for 2 seconds

    IF not in manual mode:
        Print "AutoMode"
        Call handleAutoMode
    ELSE:
        Print "Waiting for manual control"

END
```

Flowchart:



6. CODE

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

void handleAutoMode();

/*ALL SERIAL COMMUNICATION DATA STERT*/
void SerialSetup() {
  Serial.begin(115200);
  Serial.println("");
}
/*ALL SERIAL COMMUNICATION DATA ENDS*/

/*ALL TEMPERATURE SENSOR DATA STARTS*/
#define DHTPIN 14       //DHT SENSOR IS CONNECTED TO D14 (DIGITAL PIN 14)
```

```
#define DHTTYPE DHT11  //TYPE IS DHT11

/*INITIALIZE DHT11 SENSOR*/
DHT dht(DHTPIN, DHTTYPE);

void DHTsetup() {
 Serial.println("DHTxx test");
 dht.begin();
}

float getTemp() {
 //wait a few seconds between measurements
 delay(2000);
 //READING TEMPERATURE OR HUMIDITY TAKES ABOUT 250 ms
 //sensor readings may also be up to 2 sec 'old' (it is very slow sensor)
 float h = dht.readHumidity();
 float t = dht.readTemperature();      //read temperature as celcious
 float f = dht.readTemperature(true);  //read temperature as fahrenheit
 //check if any read failed and exit early (to try again)
 if (isnan(h) || isnan(t) || isnan(f))
 {
  Serial.println("Failed to read from DHT11 SENSOR");
  return 0;
 }
```

```
  float hif = dht.computeHeatIndex(f,h);     //compute heat index in fahrenheit
  float hic = dht.computeHeatIndex(t,h,false);   //compute heat index in celcious (i
fahrenheit - false)

  Serial.println("Temperature: ");
  Serial.println(t);
  Serial.println("°C");

  return t;
}
#define LDR_PIN 33
void LDRsetup()
{
  pinMode(LDR_PIN, INPUT);
}

bool isitLight()
{
  int lightState = digitalRead(LDR_PIN);
  if (lightState == HIGH) {
    Serial.println("it is dark");
  } else {
    Serial.println("it is light");
  }
  return !lightState;
}

const int trigpin = 12;
const int ecopin = 13;

#define sound_speed 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;


void ultraSonicSensorsetup()
{
  pinMode(trigpin, OUTPUT);  //sets the trig pin as output
  pinMode(ecopin, INPUT);   //stes the echo pin as input
}
float getDistance() {
  digitalWrite(trigpin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin, LOW);
  //Reads the echo, return the sound wave travel time in microseconds
  duration = pulseIn(ecopin, HIGH);

  //calculates the distance

  distanceCm = duration * sound_speed / 2;

  //CONVERT TO INCH
  distanceInch = distanceCm * CM_TO_INCH;

  //PRINT THE DISTANCE IN THE SERIAL MONITOR
  Serial.print("Distance (Cm): ");
  Serial.println(distanceCm);

  delay(1000);
  return distanceCm;
}
/*all ultrasonic sensor data ends*/



/*ALL Relay data starts*/
const int relay_pin1 = 27;   //relay in1
const int relay_pin2 = 26; //relay in2
```

```
void relaySetup()
{
  pinMode(relay_pin1, OUTPUT);
  pinMode(relay_pin2, OUTPUT);
  digitalWrite(relay_pin1, HIGH);
  digitalWrite(relay_pin2, HIGH);
}

uint8_t previousRelay1State = 1;
uint8_t previousRelay2State = 1;

void setRelay(uint8_t relay_num, uint8_t val)
{
  if (relay_num == relay_pin1)
  {
    Serial.print("Light Relay:");
    Serial.println(val);
    val = !val;
    if (previousRelay1State != val)
    {
      digitalWrite(relay_pin1, val);
    }
    previousRelay1State = val;
  }
```

```
 if (relay_num == relay_pin2)
 {
  Serial.print("Fan Relay: ");
  Serial.println(val);
  val = !val;
  if (previousRelay2State != val)
  {
   digitalWrite(relay_pin2, val);
  }
  previousRelay2State = val;
 }
}
/*all relay data ends*/


/*all WiFi data starts*/
const char* WiFi_name = "Redmi 8"; const char* WiFi_pass = "lucky123";
WiFiServer Server(80);  //default Server for WiFi is at port 80


void WiFiSetup()
{
 Serial.print("connecting to ");
 Serial.print("WiFi_name");
 WiFi.begin(WiFi_name, WiFi_pass);  //connecting to the WiFi network
```

```
  while (WiFi.status() != WL_CONNECTED)  // waiting for the respond of WiFi network
  {
    handleAutoMode();
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("connection successful");
  Serial.print("IP address");
  Serial.println(WiFi.localIP());  //getting the IP address
  Serial.println("Type the above IP address in Browser search bar");
  Server.begin();  //starting the Server
}
/*all WiFi data ends*/



void handleAutoMode() {
  // Read all seneor data
  float distance = getDistance();
  float temp = getTemp();
  bool islight = isitLight();



  if (distance <= 10)
```

```
{
  //person is present
  if (temp >= 20)
  {
    setRelay(relay_pin2, HIGH);


  }
  else
  {
    setRelay(relay_pin2, LOW);
  }
  if (islight == true) {
    //Day time
    setRelay(relay_pin1, LOW);


  }
  else
  {
    //Night time
    setRelay(relay_pin1, HIGH);
  }
}

else
```

```
  {
    //person is not present
    setRelay(relay_pin1, LOW);
    setRelay(relay_pin2, LOW);
  }
}



void setup() {
  SerialSetup();
  DHTsetup();
  LDRsetup();
  ultraSonicSensorsetup();
  relaySetup();
  WiFiSetup();
}

bool isManualMode = false;

void loop() {
  //handleAutoMode();
  WiFiClient client = Server.available();  //checking if any client request is available or not
  if (client)
```

```
{
 boolean currentlineisblank = true;
 String buffer = "";  //String is adata type in c++
 while (client.connected()) {
  if (client.available())  //if there is some client data available
  {
   char c = client.read();
   buffer+=c;  //read a byte
   if (c == '\n' && currentlineisblank)
    ;  //chack for new line charector
   {
    client.println("HTTP/1.1 200 OK");
    client.println("content.type: text/html");
    client.println();
    client.print("<HTML><titlt>ESP32</title>");
    client.print("<body><h1>ESP32 standalone ralay control </h1>");


    client.print("<p>Light control</p>");
    client.print("<a href=\"/?lightOn\"\"></button>ON</button></a>");
    client.print("<a href=\"/?lightoff\"\"></button>OFF</button></a>");


    client.print("<p>Fan control</p>");
  client.print("<a href=\"/?fanOn\"\"></button>ON</button></a>");
  client.print("<a href=\"/?fanoff\"\"></button>OFF</button></a>");



  client.print("<p>Manual control</p>");
  client.print("<a href=\"/?manualOn\"\"></button>ON</button></a>");
  client.print("<a href=\"/?manualoff\"\"></button>OFF</button></a>");


  client.print("<body></HTML>");
  break;  //break out of the whileloop
}
if (c == '\n') {
  currentlineisblank = true;
  buffer = "";



}
else if (c == '\n')
{
  if (buffer.indexOf("GET /?lightoff") >= 0)  //if fails return -1 hence<=0
  {
    setRelay(relay_pin1, LOW);
```

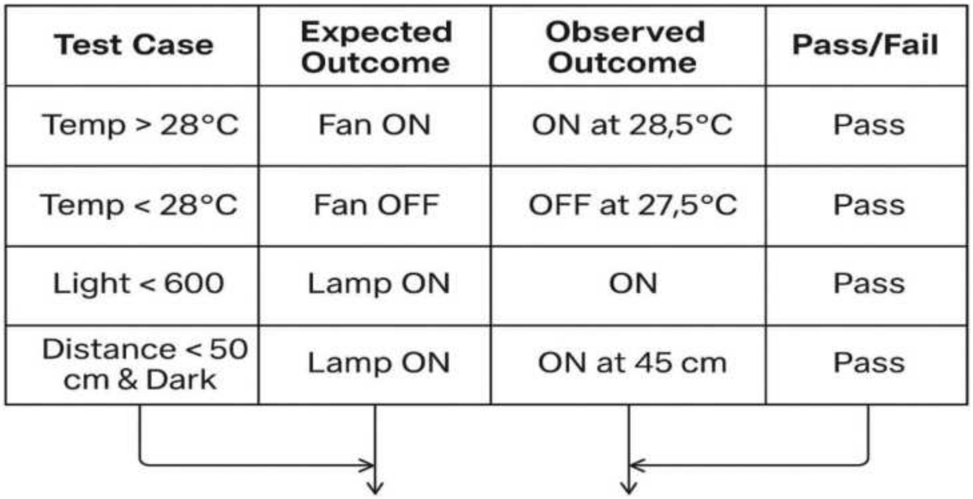
```
}

if (buffer.indexOf("GET /?lightOn") >= 0)
{
  setRelay(relay_pin1, HIGH);
}



if (buffer.indexOf("GET /?fanoff") >= 0)
{
  setRelay(relay_pin2, LOW);
}
if (buffer.indexOf("GET /?fanOn") >= 0)
{
  setRelay(relay_pin2, HIGH);
}



if (buffer.indexOf("GET /?manualOn") >= 0) {
  isManualMode = true;
  Serial.print("isManualMode:  ");
  Serial.println(isManualMode);
```

```
        }


        if (buffer.indexOf("GET /?manualoff") >= 0) {
          isManualMode = false;
          Serial.print("isManualMode:  ");
          Serial.println(isManualMode);
        }



      } else {
        currentlineisblank = false;
      }
    } else {
    }
  }
  client.stop();
}



delay(2000);
if (isManualMode == false) {
  Serial.println("AutoMode");
  handleAutoMode();
  } else {
    Serial.println("waiting for manual control");
  }
}
```

7. TESTING AND RESULTS

The system was tested under various conditions to verify functionality:

- **Test Cases:**
  - **Temperature Control:** Heated the DHT11; fan turned on at 28.5°C and off at 27.5°C.
  - **Light Control:** Covered LDR; lamp activated when light level < 600.
  - **Presence Detection:** Placed object at 45 cm in low light; lamp turned on.
- **Results Table:**



Test Case	Expected Outcome	Observed Outcome	Pass/Fail
Temp > 28°C	Fan ON	ON at 28,5°C	Pass
Temp < 28°C	Fan OFF	OFF at 27,5°C	Pass
Light < 600	Lamp ON	ON	Pass
Distance < 50 cm & Dark	Lamp ON	ON at 45 cm	Pass

- **Performance:** Response time averaged 1.5 seconds; power consumption was ~7W.

8. FUTURE WORK

- **Wi-Fi Integration:** Enable remote monitoring and control via a smartphone app.
- **Voice Control:** Add compatibility with Alexa or Google Assistant.
- **Expanded Sensors:** Include motion or air quality sensors for additional functionality.
- **Energy Tracking:** Monitor power usage of the fan and lamp for efficiency insights.