

HOME AUTOMATION USING ESP8266 AND BLYNK IOT

INTERNSHIP PROJECT REPORT

SUBMITTED BY

M Ramakrishna Reddy-22ECB0A14, NITW

G Sai Darshith Yadav -22ECB0A32, NITW

S Sai Sri Charan -22ECB0A35, NITW



Under the guidance of

Prof L. Anjaneyulu

DEPARTMENT OF ELECTRONICS AND COMMUNICATIONS
ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, WARANGAL

JULY 2024

NATIONAL INSTITUTE OF TECHNOLOGY
WARANGAL - 506 004

CERTIFICATE

This is to certify that Rama Krishna, Sai Darshith, Sai Sri Charan of National Institute of Technology Warangal have successfully completed a Project titled “Home automation using ESP8266 micro-controller and BLYNK IOT WEB CONSOLE”, as part of Summer Internship Programme under my guidance at National Institute of Technology, Warangal, Telangana during 20-05-2024 to 04-07 2024.

Prof L. Anjaneyulu, ECE dept

Visvesvaraya Centre for Skill Development

NIT Warangal

CONTENTS

Topic	Page.no
1.INTRODUCTION -----	05-06
1.1 THEORY	05
1.2 BACKGROUND OF THIS PROJECT	06
2.PROBLEM STATEMENT-----	06
3.PROJECT DESCRIPTION -----	07-15
3.1 MAIN OBJECTIVE	07
3.2 BLOCK DIAGRAM	08
3.3 HARDWARE DESCRIPTION	08-13
3.3.1 <i>ESP8266 micro-controller</i>	08
3.3.2 <i>DHT11 sensor</i>	10
3.3.3 <i>4 channel relay module (5V)</i>	11
3.3.4 <i>7805 IC</i>	12
3.3.5 <i>BLYNK WEB CONSOLE APPLICATION</i>	12
3.4 CIRCUIT DIAGRAM	13
3.5 CREATING BLYNK WEB DASHBOARD	14
TO CONTROL HOME APPLIANCES THROUGH	
BLYNK IOT PLATFORM	
4.ARDUINO IDE CODE -----	15-25
5.RESULTS-----	26-27

CONTENTS

Topic

Page.no

6.CONCLUSIONS-----28-29

6.1OVERALL RESULT	28
6.2 LIMITATIONS	28
6.3 FUTURE SCOPE	28
6.4 CONCLUSION	29

1.INTRODUCTION

1.1 THEORY

The Internet of Things (IoT) represents the next stage in technological development following the era of interconnected computers. This phase involves the integration and management of virtually all objects in the digital realm, connecting them to the internet to create a seamlessly networked world. This is a concept where each device is assigned to an IP address and through that IP address anyone makes that device detectable on internet. Initially, it has been started as the “Internet of Computers.” Research studies have forecast an explosive growth in the number of “things” or “devices” that will be connected to the Internet. The resulting network is called the “Internet of Things” (IoT). The most significant expansion of the Internet is the IoT universe, which has transformative effects across every industry and in our daily lives. In coming future, an increase in number of devices and systems will be able to send and receive information via the Internet. This will drive various new developments with huge potential, particularly in business markets. The IoT concept is not far vision of the future but is already used in many fields and has a great impact on the development of technology today.

Recent technological advancements, such as Bluetooth and Wi-Fi, enable wireless control environments, allowing different devices to connect and communicate with each other effortlessly. Using a Wi-Fi shield to serve as a micro web server for the Arduino or any other microcontroller eliminates the need for wired connections between the Arduino board and a computer, reducing costs and enabling it to function independently. The Wi-Fi requires connection to the internet through a wireless router or hotspot, serving as the portal for the Arduino to communicate with the internet. With this capability, an internet-based home automation system is designed for remote monitoring and control of home appliances.

1.2 BACKGROUND OF THIS PROJECT

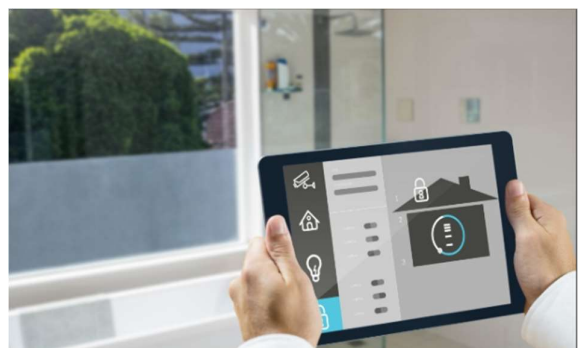
Home automation has evolved over the years, with terms like "smart home" and "intelligent home" emerging to describe the networking of household appliances. These systems, known as Home Automation Systems (HASs), offer centralized control and remote monitoring of lighting, security, and other home appliances. They enhance energy efficiency, improve security, and significantly increase user comfort and convenience.

Despite its growing popularity in the market, HASs face some challenges. Many end users, especially elderly and disabled individuals despite their benefits, hesitate due to concerns about complexity and cost.

2.PROBLEM STATEMENT

In these days, where there is a vast advancement of technology, controlling all our home appliances and our security systems manually always is not quite easy and comfortable. Our main aim is to develop a prototype enabling wireless remote control of a network of home appliances and to access our room environment conditions (such as temperature; humidity), accessible using an Android application. Key features include switch mode control, voice command functionality, and real-time device status monitoring within the application interface. And the main components which were used in this project such as

- ESP8266 micro-controller (Wi-Fi module)
- DHT11 sensor
- 4 channel relay-module (5v)
- 9v battery
- 7805IC (to regulate voltage from 9v to 5v)
- Bulbs, switches and connecting wires
- 2-0.01uf non polar capacitors



3. PROJECT DESCRIPTION

3.1 MAIN OBJECTIVES

Design of an independent Home Automation System

To design an interconnected network of home appliances integrated into a comprehensive Home Automation System.

The ultimate goal is to automate and seamlessly integrate control for each appliance within the network, enhancing overall system functionality and efficiency.

Wireless control of home appliances (Switch and Voice mode)

To develop the application that would include some special features of switch and/or voice modes to control the home appliances.

Monitoring status of appliances

Being able to view the present status of home appliances and environment conditions on the application interface, in order have a better Home Automation System.

Controlled by any device capable of Wi-Fi (Android, iOS, PC)

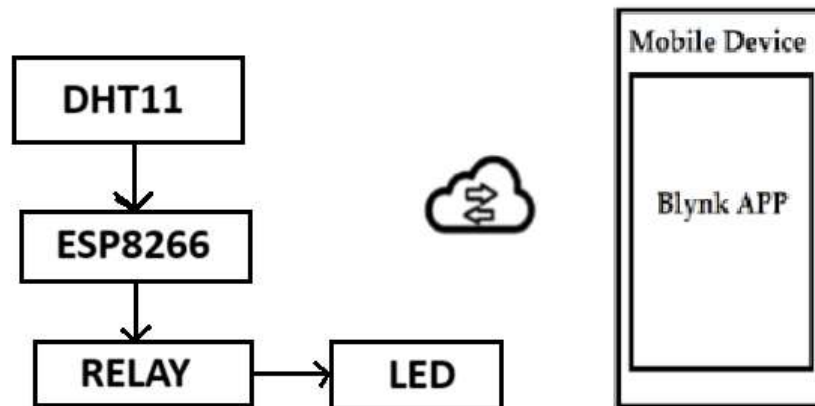
To achieve flexibility in control of the home appliances, and device capable of Wi-Fi connectivity will be able to obtain a secure control on the Home Automation System.

Flexible platform for future enhancement

With a strong existing possibility of adding and integrating more features and appliances to the system, the designed system needs to be highly flexible in nature.



3.2 BLOCK DIAGRAM



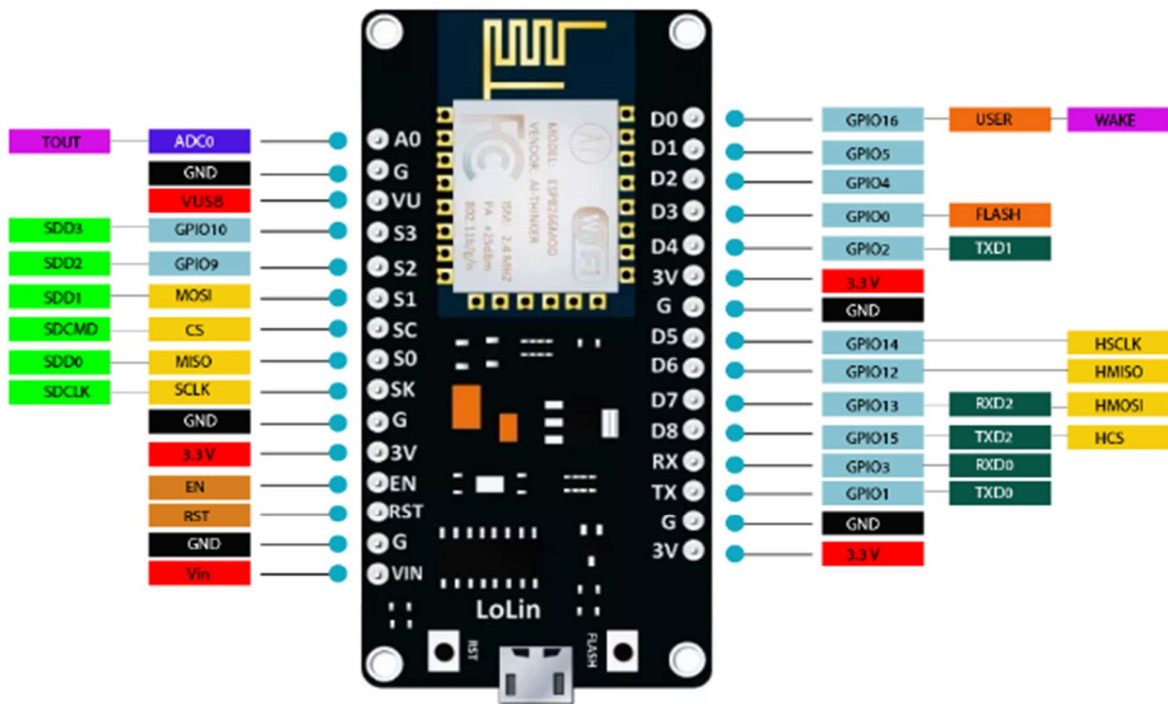
3.3 HARDWARE DESCRIPTION

3.3.1 ESP8266 micro-controller



The ESP8266 microcontroller is a highly integrated microcontroller with built-in Wi-Fi capability, designed for cost-effective and efficient and effective Internet of Things (IoT) applications. Developed by Espressif Systems, this microcontroller has gained widespread popularity due to its vast versatility, better affordability, and robust performance.

It is a low-cost microcontroller designed for embedded applications that require wireless connectivity. It features a Tensilica L106 32-bit RISC processor that operates at a clock speed of 80 MHz, with the capability to overclock up to 160MHz. The device includes built-in Wi-Fi, making it suitable for IoT projects where wireless communication is essential.



ESP8266 NodeMCU V3

Fig: The above diagram describes the pin configuration of ESP8266 microcontroller.

ESP8266 specifications:

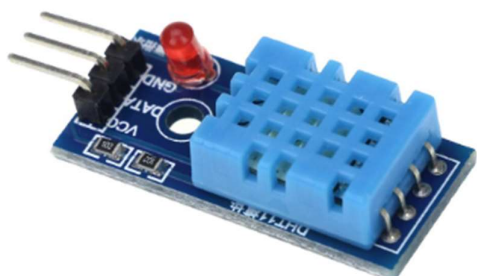
- 32-bit RISC-CPU
- 80 MHz – 160 MHz clock speed
- 64 KB SRAM
- 4 MB FLASH
- Wi-Fi 802.11 b/g/n, Bluetooth 4.2 and BLE
- 16 digital GPIO pins
- Serial connectivity: SPI, I2C, I2S, UART
- USB-TTL CP2102
- 1 Analog input
- Operating voltage: 3.3 V D.C

Which ESP8266 GPIOs are safe to use?

- ✓ – Your top priority pins. They are perfectly safe to use.
- ⚠ – Pay close attention because their behaviour, particularly during boot, can be unpredictable. Use them only when absolutely necessary.
- ✗ – It is recommended that you avoid using these pins.

Label	GPIO	Safe to use?	Reason
D0	GPIO16	!	HIGH at boot, used to wake up from deep sleep
D1	GPIO5	✓	
D2	GPIO4	✓	
D3	GPIO0	!	connected to FLASH button, boot fails if pulled LOW
D4	GPIO2	!	HIGH at boot, boot fails if pulled LOW
D5	GPIO14	✓	
D6	GPIO12	✓	
D7	GPIO13	✓	
D8	GPIO15	!	Required for boot, boot fails if pulled HIGH
RX	GPIO3	✗	Rx pin, used for flashing and debugging
TX	GPIO1	✗	Tx pin, used for flashing and debugging
CLK	GPIO6	✗	Connected to Flash memory
SDO	GPIO7	✗	Connected to Flash memory
CMD	GPIO11	✗	Connected to Flash memory
SD1	GPIO8	✗	Connected to Flash memory
SD2	GPIO9	✗	Connected to Flash memory
SD3	GPIO10	✗	Connected to Flash memory
A0	ADC0	!	Analog input pin, cannot be configured as output

3.3.2 DHT11 Sensor

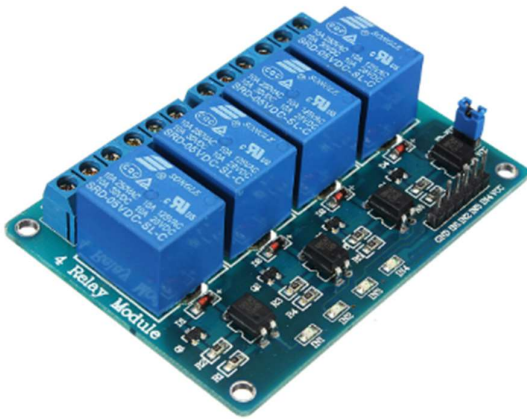


The DHT11 is a basic, low-cost digital sensor used for measuring temperature and humidity. It provides reliable readings and is widely utilized in various environmental monitoring and control applications. The sensor is popular among hobbyists and professionals for its simplicity, ease of integration, and affordability.

DHT11 specifications:

- Power supply: 3.3 to 5V DC
- Current consumption: max 2.5mA
- Operating range: 20-80% RH, 0-50°C
- Humidity measurement range: 20-90% RH
- Humidity measurement accuracy: $\pm 5\%$ RH
- Temperature measurement range: 0-50°C
- Temperature measurement accuracy: $\pm 2^\circ\text{C}$
- Response time: 1s
- Sampling rate: 1Hz (1 sample per second)
- Data transmission distance: 20-30m (at open air)
- Dimensions: 15mm x 12mm x 5.5mm

3.3.3 4-CHANNEL RELAY MODULE (5V)



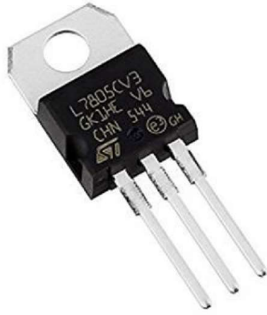
The 4-channel relay module is a versatile electronic component that allows a microcontroller to control high-voltage and high-current devices such as lights, fans, and motors. The module is used to start or stop a circuit that uses a voltage or current much higher than the one supported by ESP8266. There is no direct contact between the low voltage electronic circuit and the high voltage electrical circuit because they are protected from each other. Each of the 4-channels

has 3 types of connections called NC (Normal Closed), NO (Normal Open) and COM (Common). The relay module is used for high current switching, isolated power delivery and home automation.

4-CHANNEL RELAY MODULE specifications:

- Operating voltage: 3.75V – 6V
- Quiescent current consumption: 2mA
- Current consumption when relay is active: $\sim 70\text{mA}$
- Maximum contact voltage: 250V A.C, 30V D.C
- Maximum contact current: 10A
- Number of channels: 4

3.3.4 7805 IC

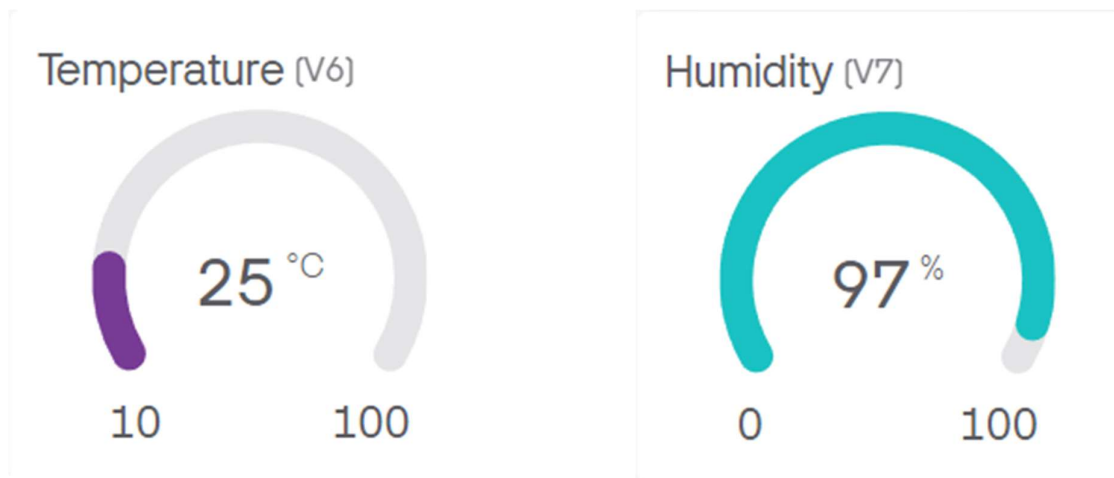


The 7805 is a popular and widely used linear voltage regulator IC that provides a stable 5V output from a higher input voltage. It is part of the 78xx series of regulators, which offer various fixed output voltages. The 7805 IC is commonly used in electronic circuits to ensure a consistent and reliable 5V power supply for components and devices.

7805IC specifications/Features:

- 5V Positive Voltage Regulator
- Minimum Input Voltage is 7V
- Maximum Input Voltage is 25V
- Operating current (IQ) is 5mA
- Internal Thermal Overload and Short circuit current limiting protection is available.
- Junction Temperature maximum 125 degree Celsius

3.3.5 BLYNK WEB CONSOLE APPLICATION



The top part of the Blynk application is made from 2 widgets. The VIOLET widget is used for showing the temperature measured by the system, the blue one is used for showing the humidity measured by the system.

SWITCH 1 (V1)



SWITCH 2 (V2)

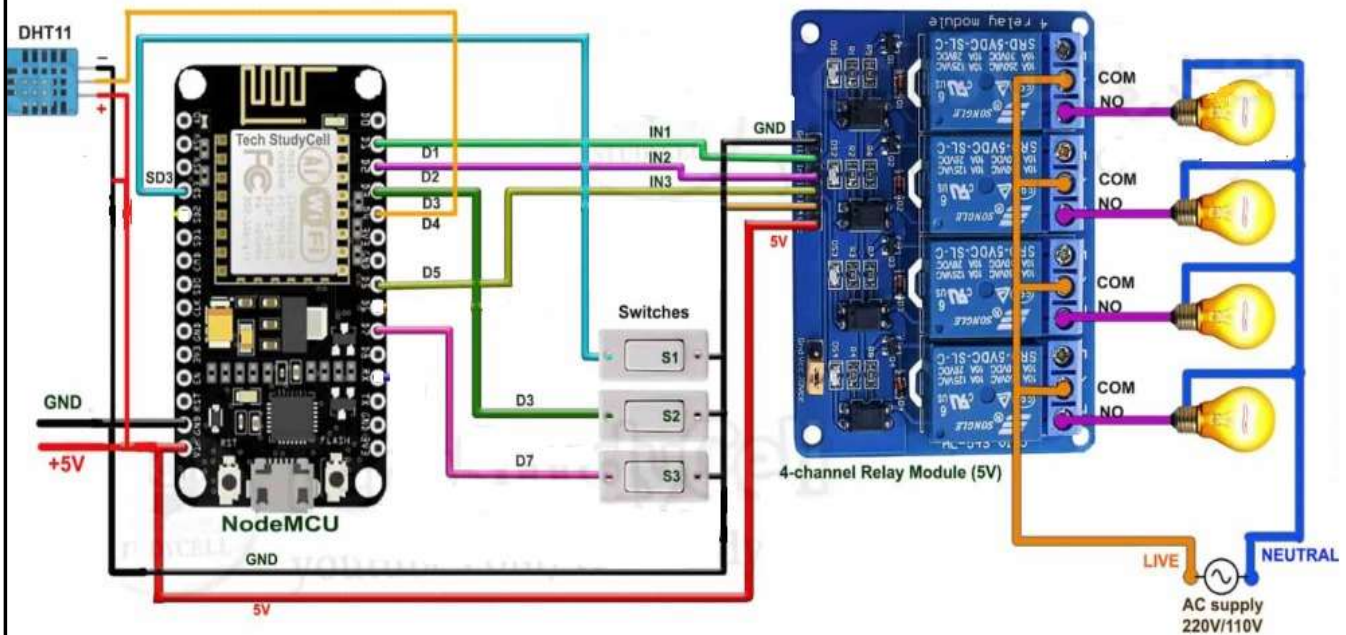


SWITCH 3 (V3)



The above part of the Blynk application is made from 3 buttons. Each button is used to operate one of the 4 channels on the relay module.

3.4 CIRCUIT DIAGRAM



In the above circuit diagram, as there is only availability of 9V battery or less than 5V, I've made another circuit to create 5V using 9V battery with the help of 7805 regulator IC.

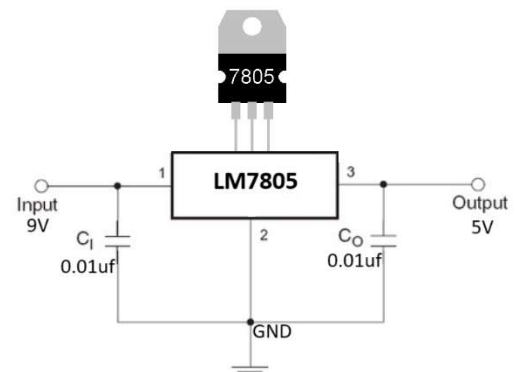
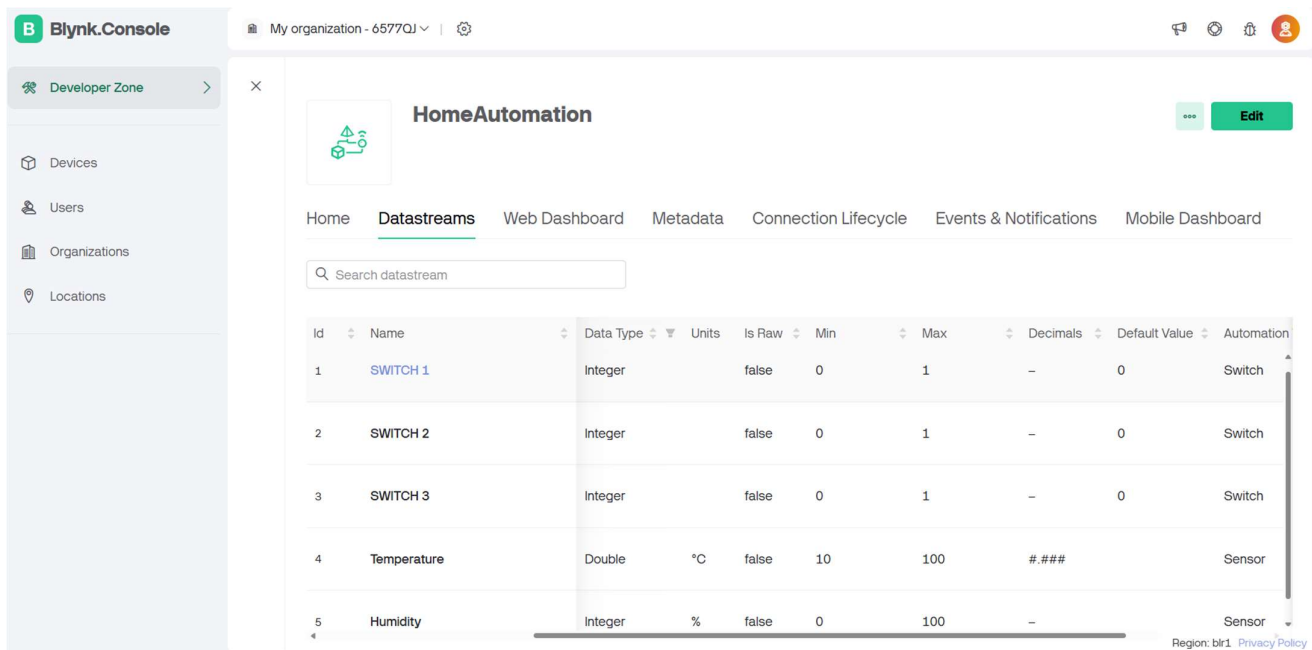


fig:5V using 7805 IC

3.5 CREATING BLYNK WEB DASHBOARD TO CONTROL HOME APPLIANCES THROUGH BLYNK IOT PLATFORM

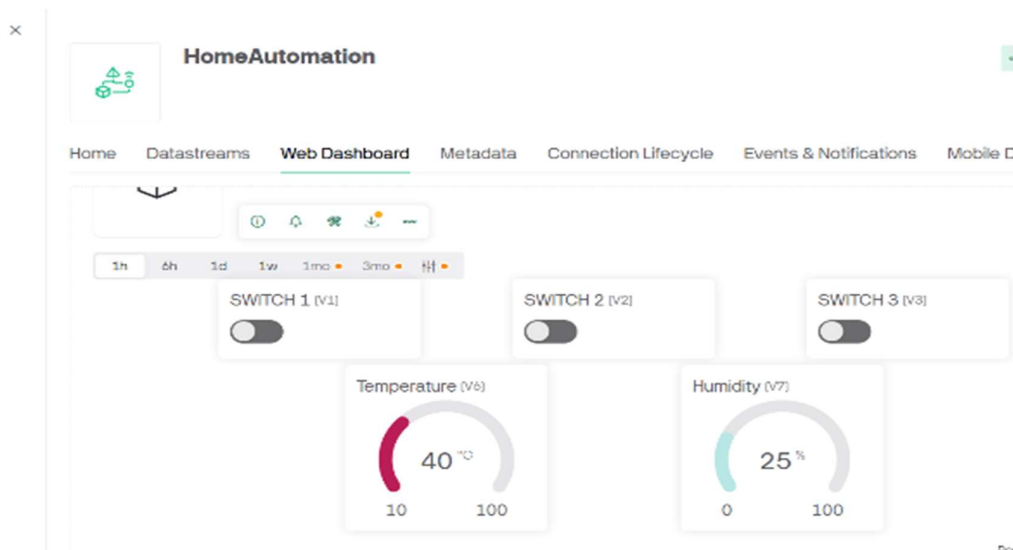
- After logging into blynk iot platform, I've created a new template where I can get an authorized blynk ID and NAME which should be added in our code at the beginning.
- Then I created my required datastreams such as switches and sensors.



The screenshot shows the Blynk Console interface. On the left is a sidebar with navigation options: Developer Zone, Devices, Users, Organizations, and Locations. The main area displays the 'HomeAutomation' project with tabs for Home, Datastreams (selected), Web Dashboard, Metadata, Connection Lifecycle, Events & Notifications, and Mobile Dashboard. Below the tabs is a search bar and a table of datastreams.

Id	Name	Data Type	Units	Is Raw	Min	Max	Decimals	Default Value	Automation
1	SWITCH 1	Integer		false	0	1	-	0	Switch
2	SWITCH 2	Integer		false	0	1	-	0	Switch
3	SWITCH 3	Integer		false	0	1	-	0	Switch
4	Temperature	Double	°C	false	10	100	###		Sensor
5	Humidity	Integer	%	false	0	100	-		Sensor

- Then I've created my own widgets in dashboard and if needed we can create automation for required appliances and then create a schedule whenever you want.



The screenshot shows the 'Web Dashboard' tab for the 'HomeAutomation' project. It features a top navigation bar with tabs: Home, Datastreams, Web Dashboard (selected), Metadata, Connection Lifecycle, Events & Notifications, and Mobile C. Below the tabs is a toolbar with icons for time range (1h, 6h, 1d, 1w, 1mo, 3mo, All) and a dropdown menu. The dashboard displays five widgets: three toggle switches labeled 'SWITCH 1 [v1]', 'SWITCH 2 [v2]', and 'SWITCH 3 [v3]', and two gauge sensors labeled 'Temperature [v4]' and 'Humidity [v7]'. The Temperature gauge shows a value of 40 °C, and the Humidity gauge shows a value of 25 %.

- For automation, we need to turn on condition and action while creating datastreams, and then by adding schedules, it may be based on time or by temperature, that particular relay will operate with that schedule irrespective of our command.
- After saving it and applying, we'll receive a token which should be placed in our code at the beginning as it connects our circuit and our Blynk device.

4.ARDUINO IDE CODE

* Download the libraries (can be downloaded directly from libraries or use below links)

* Blynk Library : <https://github.com/blynkkk/blynk-library>

* DHT Library : <https://github.com/adafruit/DHT-sensor-library>

```
#define BLYNK_TEMPLATE_ID "TMPL3gztglFqx"//only if web dashboard is used
```

```
#define BLYNK_TEMPLATE_NAME "HomeAutomation"
```

```
#define BLYNK_AUTH_TOKEN "yawWYczQldU_7xuqn3fBYtaxKoCTejsP"
```

```
// Your WiFi credentials.
```

```
// Set password to "" for open networks.
```

```
char ssid[] = "Naga";
```

```
char pass[] = "85979161";
```

```
bool fetch_blynk_state = true; //true or false
```

```
//#define BLYNK_PRINT Serial
```

```
#include <ESP8266WiFi.h>
```

```
#include <BlynkSimpleEsp8266.h>
```

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

```
#define DHTPIN      2 //D4 pin connected with DHT
```

```
// Uncomment whatever type you're using!
```

```
#define DHTTYPE DHT11  // DHT 11
```

```
//#define DHTTYPE DHT22  // DHT 22, AM2302, AM2321
```

```
//#define DHTTYPE DHT21  // DHT 21, AM2301
```

```
#define RelayPin1 5 //D1
```

```
#define RelayPin2 4 //D2
```

```
#define RelayPin3 14//D5
```

```
#define SwitchPin1 12 //D6
```

```
#define SwitchPin2 0 //D3
```

```
#define SwitchPin3 13 //D7
```



```
#define wifiLed 16 //D0
```

```
//Change the virtual pins according the rooms
```

```
#define VPIN_BUTTON_1 V1
```

```
#define VPIN_BUTTON_2 V2
```

```
#define VPIN_BUTTON_3 V3
```

```
#define VPIN_TEMPERATURE V6
```

```
#define VPIN_HUMIDITY V7
```

```
// Relay State
```

```
bool toggleState_1 = LOW; //Define integer to remember the toggle state for  
relay 1
```

```
bool toggleState_2 = LOW; //Define integer to remember the toggle state for  
relay 2
```

```
bool toggleState_3 = LOW; //Define integer to remember the toggle state for  
relay 3
```

```
// Switch State
```

```
bool SwitchState_1 = LOW;
```

```
bool SwitchState_2 = LOW;
```

```
bool SwitchState_3 = LOW;

int wifiFlag = 0;

float temperature1 = 0;

float humidity1 = 0;


char auth[] = BLYNK_AUTH_TOKEN;


BlynkTimer timer;

DHT dht(DHTPIN, DHTTYPE);


// When App button is pushed - switch the state


BLYNK_WRITE(VPIN_BUTTON_1) {

    toggleState_1 = param.asInt();

    digitalWrite(RelayPin1, !toggleState_1);

}

BLYNK_WRITE(VPIN_BUTTON_2) {

    toggleState_2 = param.asInt();

    digitalWrite(RelayPin2, !toggleState_2);

}

BLYNK_WRITE(VPIN_BUTTON_3) {
```

```

toggleState_3 = param.asInt();

digitalWrite(RelayPin3, !toggleState_3);

}

void all_SwitchOff(){

    toggleState_1      =      0;      digitalWrite(RelayPin1,      HIGH);
    Blynk.virtualWrite(VPIN_BUTTON_1, toggleState_1); delay(100);

    toggleState_2      =      0;      digitalWrite(RelayPin2,      HIGH);
    Blynk.virtualWrite(VPIN_BUTTON_2, toggleState_2); delay(100);

    toggleState_3      =      0;      digitalWrite(RelayPin3,      HIGH);
    Blynk.virtualWrite(VPIN_BUTTON_3, toggleState_3); delay(100);

    Blynk.virtualWrite(VPIN_HUMIDITY, humidity1);

    Blynk.virtualWrite(VPIN_TEMPERATURE, temperature1);

}

void checkBlynkStatus() { // called every 2 seconds by SimpleTimer

    bool isconnected = Blynk.connected();

    if (isconnected == false) {

        wifiFlag = 1;

        Serial.println("Blynk Not Connected");

        digitalWrite(wifiLed, HIGH);

    }

    if (isconnected == true) {

        wifiFlag = 0;

```

```
if (!fetch_blynk_state){

  Blynk.virtualWrite(VPIN_BUTTON_1, toggleState_1);

  Blynk.virtualWrite(VPIN_BUTTON_2, toggleState_2);

  Blynk.virtualWrite(VPIN_BUTTON_3, toggleState_3);

}

digitalWrite(wifiLed, LOW);

//Serial.println("Blynk Connected");

}

}

BLYNK_CONNECTED() {

  // Request the latest state from the server

  if (fetch_blynk_state){

    Blynk.syncVirtual(VPIN_BUTTON_1);

    Blynk.syncVirtual(VPIN_BUTTON_2);

    Blynk.syncVirtual(VPIN_BUTTON_3);

  }

  Blynk.syncVirtual(VPIN_TEMPERATURE);

  Blynk.syncVirtual(VPIN_HUMIDITY);

}

void readSensor(){

  float h = dht.readHumidity();
```

```
float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit

if (isnan(h) || isnan(t)) {

    Serial.println("Failed to read from DHT sensor!");

    return;

}

else {

    humidity1 = h;

    temperature1 = t;

    Serial.println(temperature1);

    Serial.println(humidity1);

}

}

void sendSensor()

{

    readSensor();

    Blynk.virtualWrite(VPIN_HUMIDITY, humidity1);

    Blynk.virtualWrite(VPIN_TEMPERATURE, temperature1);

}

void manual_control()

{

    if (digitalRead(SwitchPin1) == LOW && SwitchState_1 == LOW) {
```

```
digitalWrite(RelayPin1, LOW);

toggleState_1 = HIGH;

SwitchState_1 = HIGH;

Blynk.virtualWrite(VPIN_BUTTON_1, toggleState_1);

Serial.println("Switch-1 on");

}

if (digitalRead(SwitchPin1) == HIGH && SwitchState_1 == HIGH) {

    digitalWrite(RelayPin1, HIGH);

    toggleState_1 = LOW;

    SwitchState_1 = LOW;

    Blynk.virtualWrite(VPIN_BUTTON_1, toggleState_1);

    Serial.println("Switch-1 off");

}

if (digitalRead(SwitchPin2) == LOW && SwitchState_2 == LOW) {

    digitalWrite(RelayPin2, LOW);

    toggleState_2 = HIGH;

    SwitchState_2 = HIGH;

    Blynk.virtualWrite(VPIN_BUTTON_2, toggleState_2);

    Serial.println("Switch-2 on");

}

if (digitalRead(SwitchPin2) == HIGH && SwitchState_2 == HIGH) {
```

```
digitalWrite(RelayPin2, HIGH);

toggleState_2 = LOW;

SwitchState_2 = LOW;

Blynk.virtualWrite(VPIN_BUTTON_2, toggleState_2);

Serial.println("Switch-2 off");

}

if (digitalRead(SwitchPin3) == LOW && SwitchState_3 == LOW) {

    digitalWrite(RelayPin3, LOW);

    toggleState_3 = HIGH;

    SwitchState_3 = HIGH;

    Blynk.virtualWrite(VPIN_BUTTON_3, toggleState_3);

    Serial.println("Switch-3 on");

}

if (digitalRead(SwitchPin3) == HIGH && SwitchState_3 == HIGH) {

    digitalWrite(RelayPin3, HIGH);

    toggleState_3 = LOW;

    SwitchState_3 = LOW;

    Blynk.virtualWrite(VPIN_BUTTON_3, toggleState_3);

    Serial.println("Switch-3 off");

}

}
```

```
void setup()

{

  Serial.begin(9600);

  pinMode(RelayPin1, OUTPUT);

  pinMode(RelayPin2, OUTPUT);

  pinMode(RelayPin3, OUTPUT);


  pinMode(wifiLed, OUTPUT);


  pinMode(SwitchPin1, INPUT_PULLUP);
  pinMode(SwitchPin2, INPUT_PULLUP);
  pinMode(SwitchPin3, INPUT_PULLUP);


  //During Starting all Relays should TURN OFF

  digitalWrite(RelayPin1, !toggleState_1);

  digitalWrite(RelayPin2, !toggleState_2);

  digitalWrite(RelayPin3, !toggleState_3);


  dht.begin();  // Enabling DHT sensor

  digitalWrite(wifiLed, HIGH);
```



```
//Blynk.begin(auth, ssid, pass);

WiFi.begin(ssid, pass);

timer.setInterval(2000L, checkBlynkStatus); // check if Blynk server is connected
every 2 seconds

timer.setInterval(1000L, sendSensor); // Sending Sensor Data to Blynk Cloud
every 1 second

Blynk.config(auth);

delay(1000);

if (!fetch_blynk_state){

    Blynk.virtualWrite(VPIN_BUTTON_1, toggleState_1);

    Blynk.virtualWrite(VPIN_BUTTON_2, toggleState_2);

    Blynk.virtualWrite(VPIN_BUTTON_3, toggleState_3);

}

}

void loop()

{

    manual_control();

    readSensor();

    sendSensor();

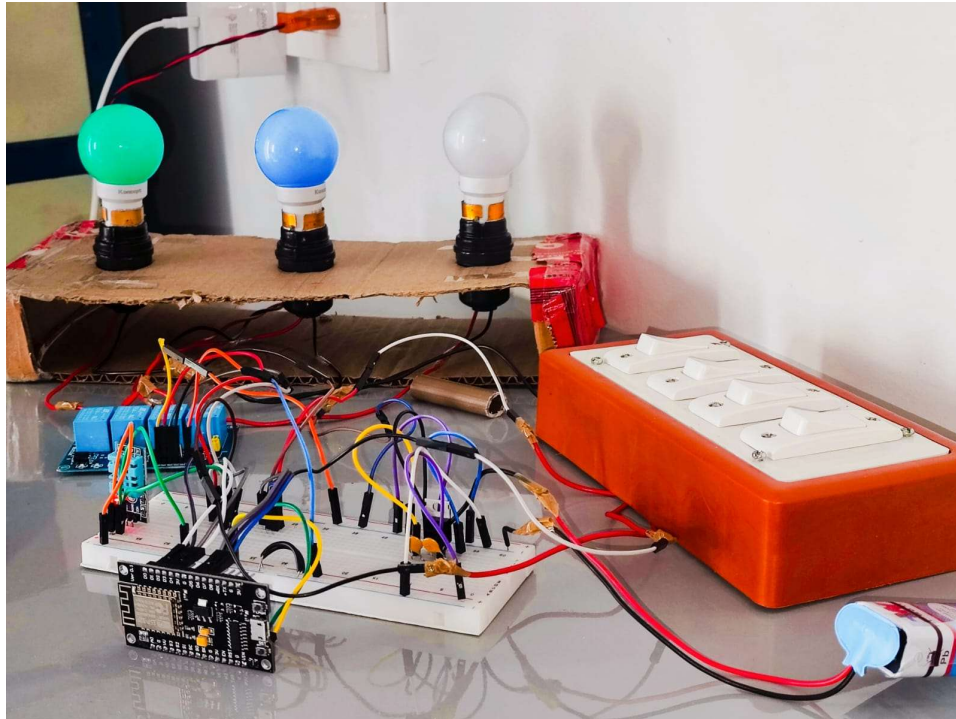
    Blynk.run();

    timer.run();

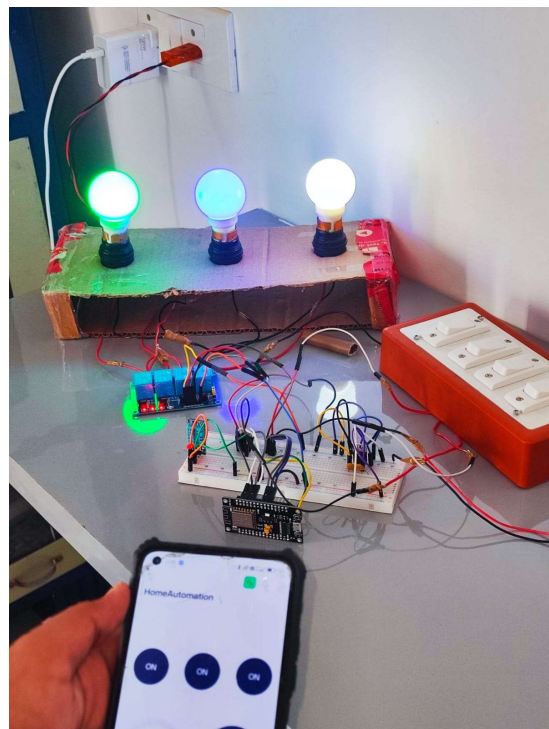
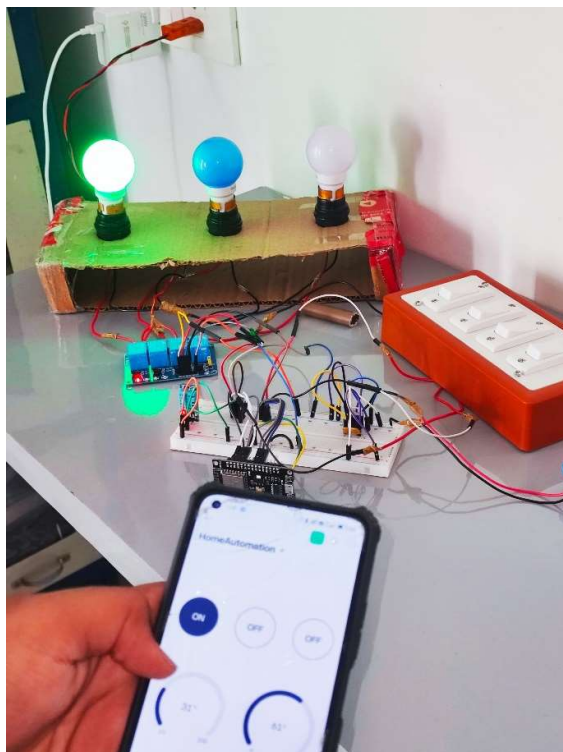
}
```

5.RESULTS

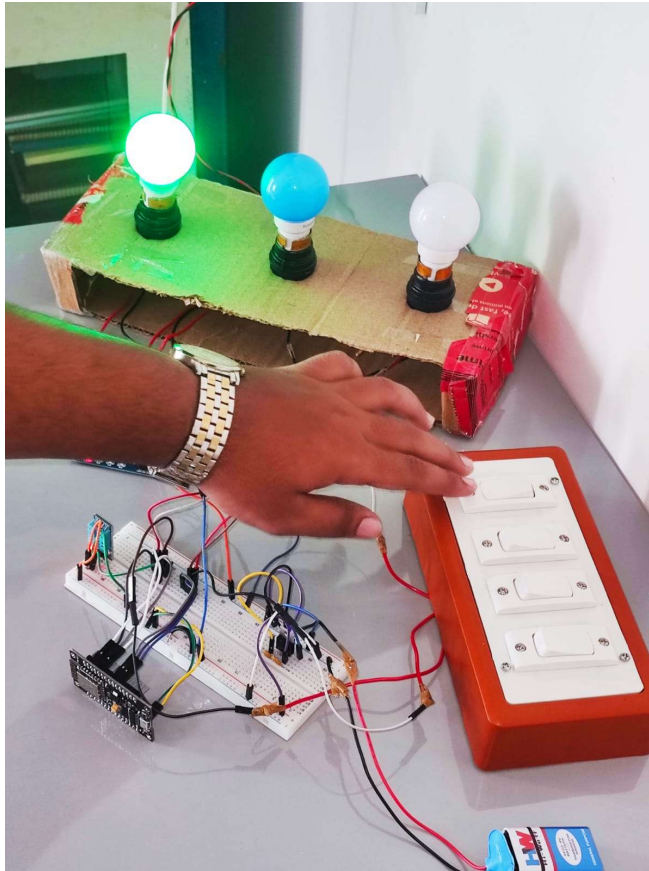
a)CONNECTION DIAGRAM



b)RESULT via BLYNK APPLICATION

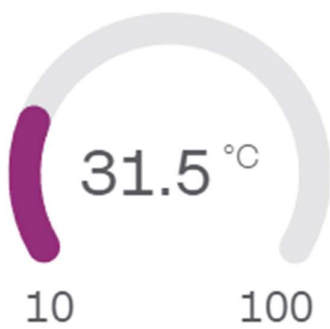


C) RESULT via MANUAL SWITCHES

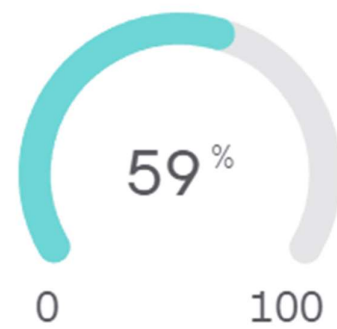


D) DHT11 Sensor RESULT

Temperature



Humidity



6.CONCLUSIONS

6.1 OVERALL RESULT

Experimental model was done according to circuit diagram and practical outcomes were as expected outcomes. All home appliances are connected to Wi-Fi and they are switching without much delay via android application. Even Blynk is also successful in showing precise results. This project was done on a breadboard. To avoid any loose connections, designing a PCB and conducting this project on it is preferable.

6.2 LIMITATIONS

This project was done based on android application, other platforms may not work in our application. When it is connected to voice applications like google assistant, the disturbance around us may affect while commanding over our voice. This may show ambiguity in results.

6.3 FUTURE SCOPE

Home automation systems are rapidly evolving with the advancements in technology, presenting numerous opportunities for future development and expansion. Implementing AI and machine learning to analyse user behaviour and predict needs, automatically adjusting settings for comfort, energy efficiency, and security, enhancing security and privacy like integrating biometric authentication methods such as facial recognition, fingerprint scanning, and voice recognition for secure access and control, Expanding this in many areas despite only home.

6.4 CONCLUSION

In conclusion, the Home Automation System project significantly improves the efficiency, security, and user-friendliness of living environments. Utilizing IoT, wireless communication, and intelligent control systems, it allows seamless integration and remote management of household appliances. This system offers excellent convenience, enabling users to control devices and monitor temperature, humidity, and brightness remotely via Wi-Fi on Android and iOS devices. As technology advances, the system will evolve, adding more features and applications, ultimately transforming home management. Hence, this system is both scalable and flexible.

