Smart SyncEdge Platform

Embedded Systems

Submitted in partial fulfillment for the award of the course

Of

BACHELOR OF ENGINEERING

In

COMPUTER SYSTEMS ENGINEERING

Submitted By:

ZAIN UL ABIDIN[133-22-0001] ,

SOYAM KAPOOR[133-22-0041]

Guided by:

Dr: Junaid Ahmed, and Engr. Asif Ali

Department of Computer Systems Engineering

Sukkur IBA University

Abstract

The **Smart SyncEdge Platform** is a smart home automation IoT system designed to enable seamless control of home appliances using ESP32, Arduino IoT Cloud, and voice assistants such as Google Home and Alexa. The system supports internet-based, and manual switch control of appliances. With features like four relays for multiple appliances, a DHT11 sensor for temperature monitoring, the platform offers versatile functionality both online and offline. This solution is user-friendly, costeffective, and compatible with multiple environments, providing efficient and modern home automation for convenience, accessibility, and energy savings.

I. INTRODUCTION

Home automation represents a modern shift in how individuals interact with their home environments, enabling enhanced control, monitoring, and efficiency through IoT technologies. The **Smart SyncEdge Platform** aims to streamline the management of home appliances, making it accessible and intuitive for users.

At its core, this project leverages an **ESP32 microcontroller** integrated with the **Arduino IoT Cloud** for seamless connectivity and control. The platform supports voice assistants like Google Home and Alexa, enabling effortless appliance management through voice commands. Offline control is ensured via manual switches enhancing flexibility and reliability.

This project's objectives include:

- Providing centralized control of appliances using a mobile/web interface and voice commands.
- Offering real-time temperature monitoring via a **DHT11 sensor**.
- Supporting offline functionality to ensure usability during internet outages.
- Introducing energy efficiency by automating and monitoring appliance usage.

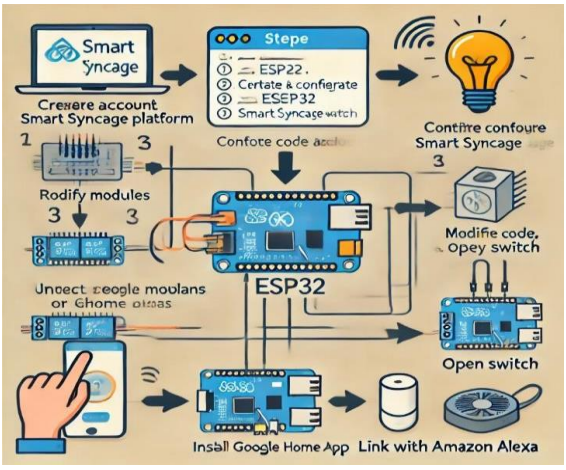

Figure 1: Home Automation

II. SYSTEM METHODOLOGY

A. System Components:

1. **ESP32 Microcontroller:**
   The ESP32 acts as the system's core, featuring built-in Wi-Fi and Bluetooth capabilities. It communicates with the Arduino IoT Cloud, enabling device control over the internet.
2. **Arduino IoT Cloud:**
   The cloud platform provides a user-friendly interface for managing appliances, monitoring device status, and configuring automation routines.
3. **DHT11 Temperature Sensor:**
   This sensor collects temperature and humidity data in real-time, displayed on the web/mobile interface.
4. **4-Channel Relay Module:**
   The relay module connects to the ESP32, enabling the control of high-voltage appliances such as lights and fans.
5. **Manual Switches:**
   These switches provide direct physical control over appliances, with their state updates reflected in the online dashboard.
6. **Voice Assistants (Google Home & Alexa):**
   Integrated with the platform, these assistants enable hands-free control of appliances through voice commands.

B. Working procedure:

1. **Cloud Integration:**
   The ESP32 is programmed using the Arduino IDE, with configurations for Wi-Fi credentials and connection to the Arduino IoT Cloud. Appliances are registered as virtual devices on the cloud platform.
2. **Data Flow:**
   - When a command is issued via voice, app, the ESP32 processes it and controls the respective relay.
   - Temperature data from the DHT11 sensor is continuously sent to the Arduino IoT Cloud for real-time monitoring.
3. **Offline Mode:**
   - Manual switches provide direct control of appliances.
   - State changes made offline are synced to the cloud when the internet is restored.
4. **Energy Efficiency:**
   Automation routines, such as turning off appliances when not in use, are configured through the Arduino IoT Cloud, promoting energy savings.

III. SYSTEM DESIGN AND SIMULATION

A. Hardware Configuration

The hardware setup consists of:

- The ESP32 microcontroller connected to a 5V DC power supply.
- Four relays connected to appliances such as lights and fans.
- Manual switches interfaced with the ESP32 to operate the relays directly.

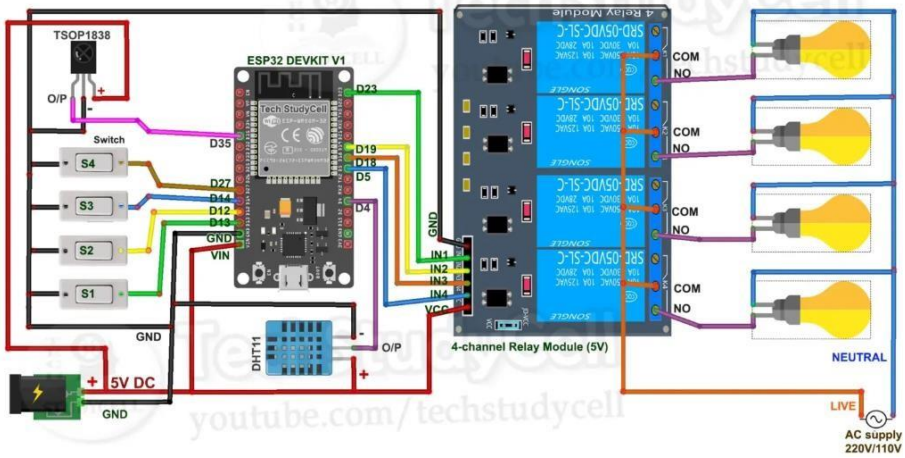

Figure 2: Hardware Configuration Diagram

B. Software Implementation

1. Arduino IDE:

- Libraries such as WiFi, ArduinoIoTCloud, and DHT are utilized for programming.
- The code includes routines for handling cloud communication, relay control, and sensor data processing.

2. Arduino IoT Cloud Setup:

- Devices are registered, and virtual switches are mapped to appliance relays.
- Automation rules, such as scheduled appliance control, are configured via the cloud dashboard.

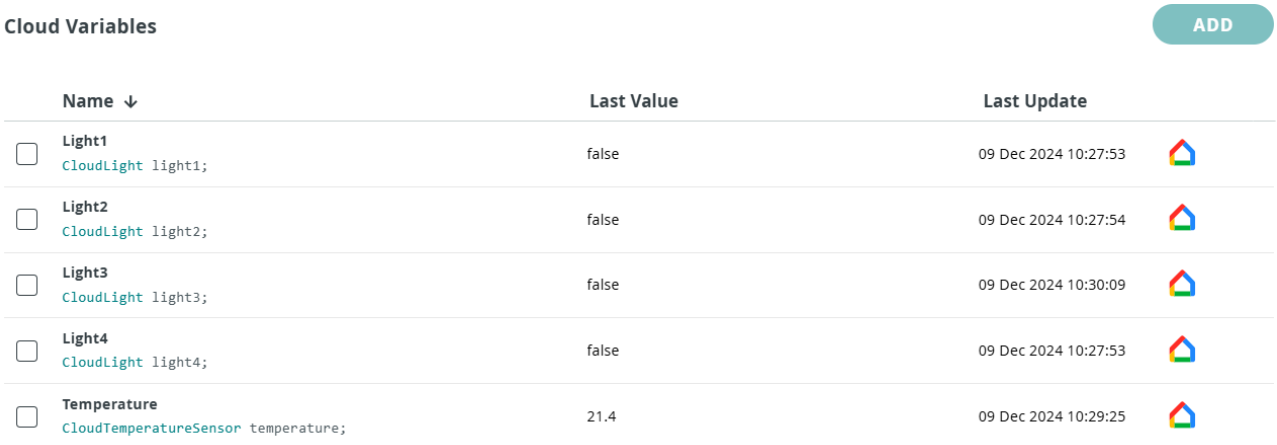

Figure 3: Arduino IoT Cloud Interface

3. **Google Home Setup:**

The system supports up to **6 users** who can connect and gain access to control the appliances. Each user can access the system via Google Home app interface, which is secured by a login system. Once logged in, users can remotely control the connected devices (such as bulbs, fan, and open port switch) using their individual credentials. The system is designed to handle multiple simultaneous connections, ensuring that all 6 users can control the devices independently without interference.

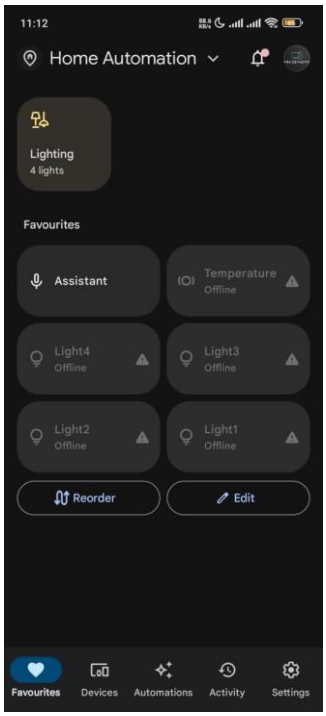

Figure 4: Google Home Interface

C. Simulation

The system is tested in simulation software to ensure all components function as expected. Temperature values from the DHT11 sensor are displayed in real-time, and relay operations are validated against cloud commands and manual inputs.

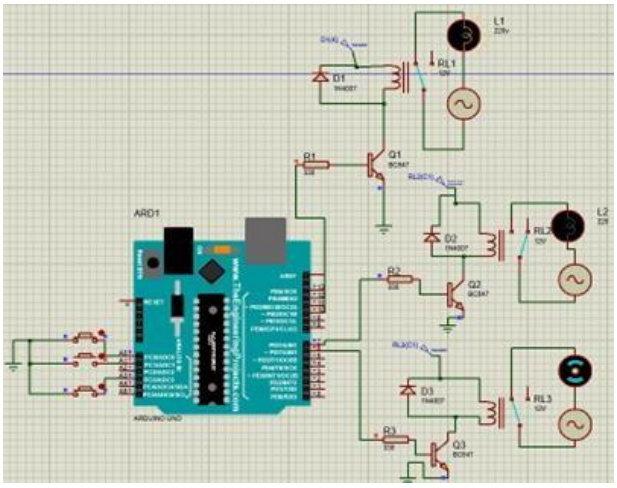

Figure 5: Simulation

IV. HARWARE DEVELOPMENT AND TESTING

A. Power Supply:

The ESP32 is powered by a regulated 5V supply, typically provided via a USB cable or a mobile charger adapter. The relay module and connected appliances receive appropriate power through separate circuits.

B. Load:

Appliances such as bulbs, motors, and other AC loads are connected to the relay module. The relays successfully manage the on/off states of these devices based on commands received from the ESP32 microcontroller. The system demonstrates reliable performance in handling varying loads, ensuring seamless operation under normal household conditions.

C. Manual Switch Integration:

Manual switches are interfaced with the ESP32 GPIO pins configured as inputs. The state of these switches is continuously monitored, allowing manual control without internet dependency.

D. Hardware Setup:

The system uses the **ESP32** microcontroller as the central control unit, responsible for managing the operation of the appliances. It interfaces with the **relay module** which controls the appliances. The ESP32 communicates with the relay module to manage the on/off states of 2 bulbs, a fan, and an open port switch connected to it. The relay module operates the high-voltage appliances safely, while the ESP32 handles control signals sent from the user interface. GPIO pins from the ESP32 board are connected to the relays to control the appliances. The setup ensures secure control of connected devices through **both online and offline modes** via Wi-Fi or manual switches.

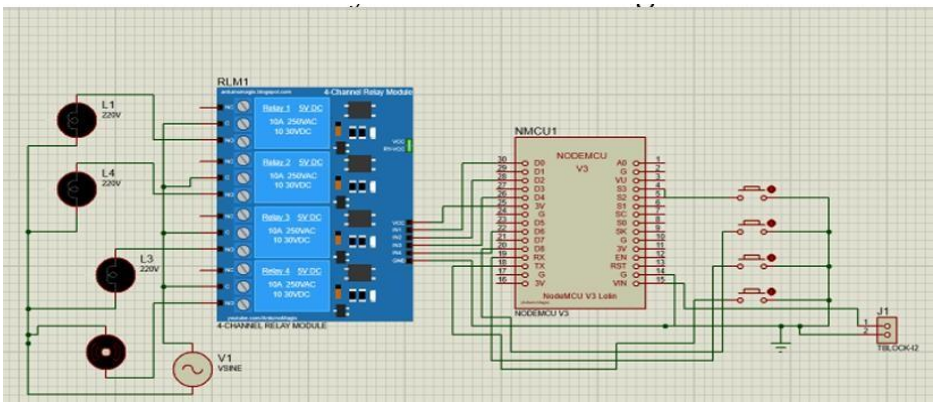

Figure 6: Hardware Setup

V. RESULT AND DISCUSSION

A. Testing Results:

1. Voice commands via Google Home/Alexa successfully controlled appliances.
2. The Arduino IoT Cloud dashboard reflected real-time updates and allowed device control.
3. Manual switches operations worked seamlessly in offline mode.
4. Temperature data from the DHT11 sensor was accurately displayed on the web interface.

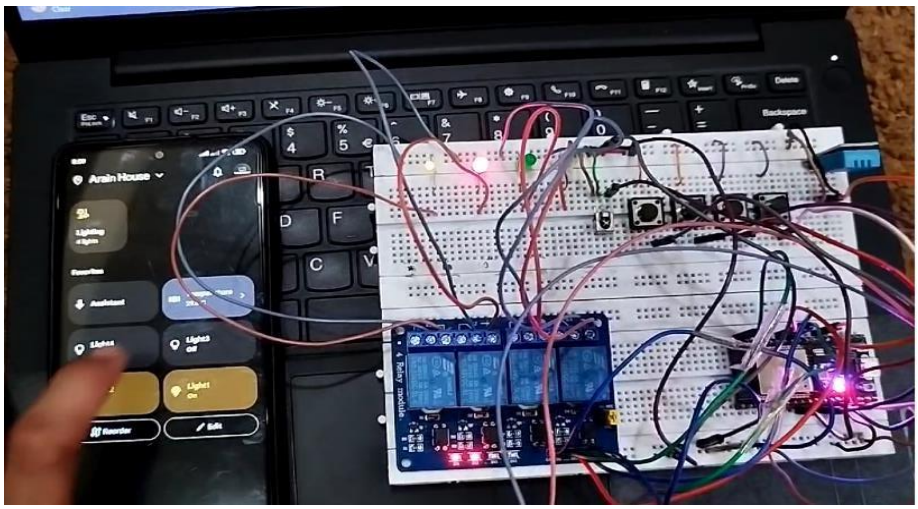

Figure 7: Prototype Design and Google home Setup

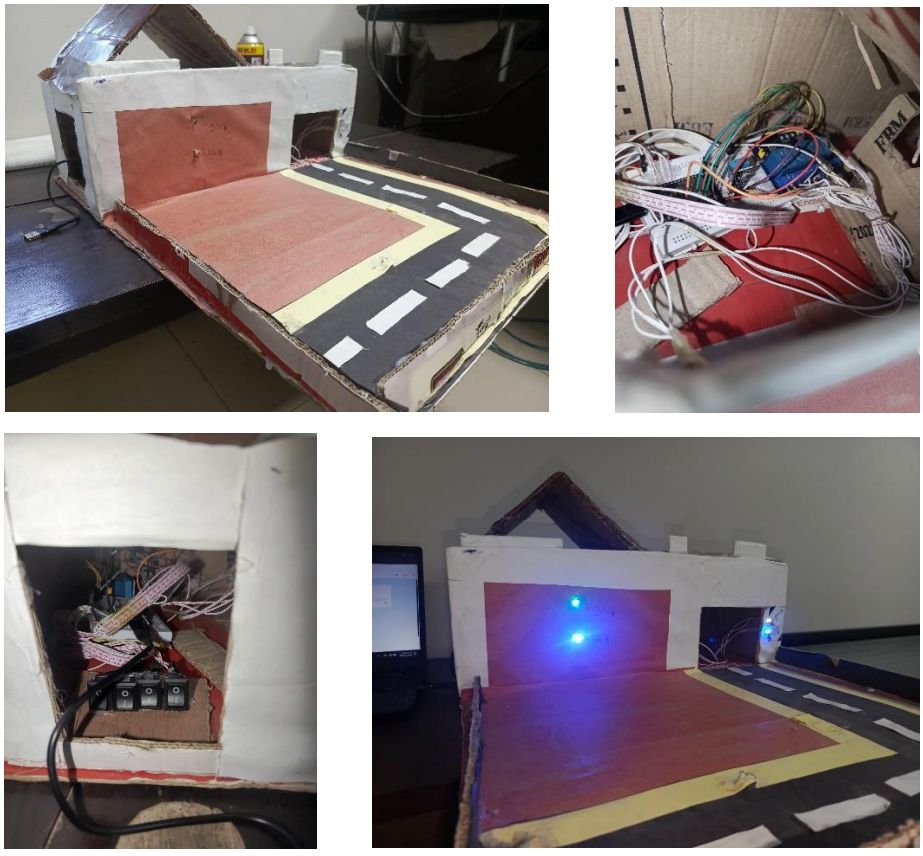

Figure 8: Home Automation Design Setup

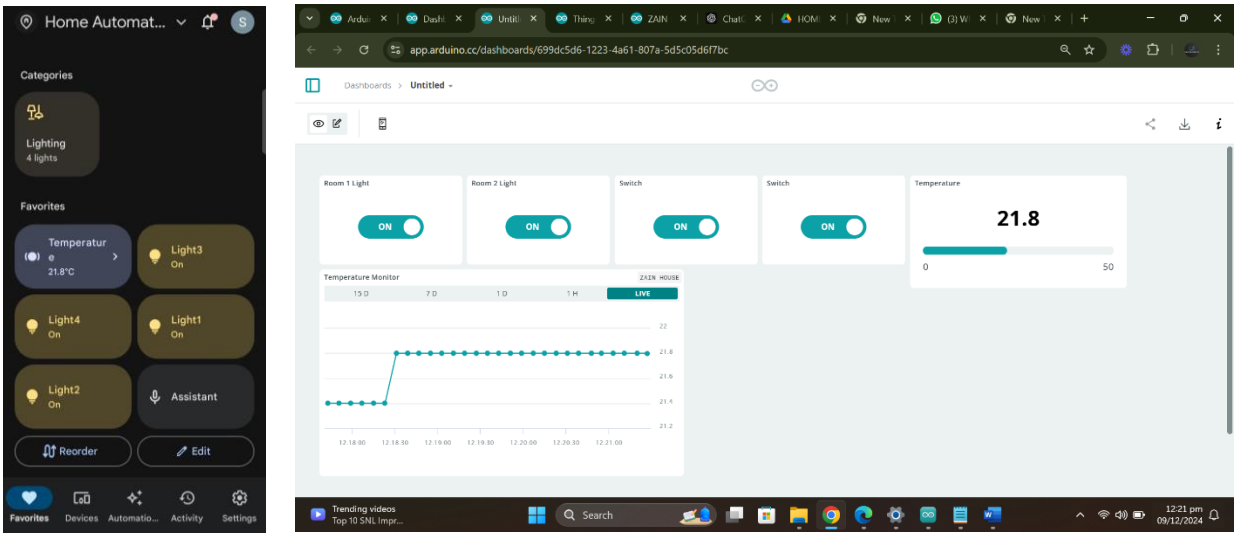

Figure 9: Arduino Web Dashboard and Google home Setup

B. Observations:

- The system demonstrated high reliability in both online and offline modes.
- Voice integration significantly enhanced user convenience.
- The use of manual switches ensured usability for all demographics, including those unfamiliar with smart technology.

C. Data Table:

S/N	Load	Output	Control (Manual, Virtual, Voice command)		
1	**Light 1**	on	ok	ok	ok
2	**Light 2**	on	ok	ok	ok
3	**Light 3**	on	ok	ok	ok
4	**Light 4**	on	ok	ok	ok
5	**Temperature Sensor Data**	on	ok	ok	ok

Table 1: System Working Data Table

VI. COST ANALYSIS & IMPORTANCE

A. Project Cost Breakdown:

Component	Quantity	Cost per Unit (PKR)	Total Cost (PKR)
ESP32 Microcontroller	1	1500	1500
Relay Module (4-channel)	1	500	500
Miscellaneous (Switches, Wires, etc.)	-	500	500
Total Cost			**2500**

Table 2: Cost Analyzation Table

B. Who Will Buy?

- Homeowners looking for a cost-effective and versatile smart home solution.
- Tech enthusiasts who value convenience and modern automation.

C. Why Buy This System?

- **Reliability**: Offline functionality ensures devices remain operational during connectivity issues.
- **Multi-User Support**: Supports up to six users, making it ideal for families.
- **Ease of Use**: Compatible with Google Assistant and a mobile app for seamless control.
- **Scalability**: The system can be expanded to include additional devices as needed.
- **Cost-Effective**: Offers advanced features at a fraction of the cost of traditional smart home systems.

VII. Impact of the Home Automation Project on Society

The implementation of a home automation system using ESP32, a relay module, a DHT temperature sensor, and LED indicators has far-reaching implications for society. The ability to control home appliances via Google Home, the Arduino IoT Cloud, and physical switches showcases innovation and practical utility. Below are the key impacts of this project on society:

1. Enhanced Convenience and Comfort

- The ability to control appliances using voice commands via Google Home or through an online platform eliminates the need for manual operation.
- Users can manage their home environment remotely, ensuring comfort even when they are away.

2. Energy Efficiency

- Automation enables users to monitor and control appliances efficiently, reducing unnecessary energy consumption.
- Integration with temperature sensors allows optimized energy use, such as turning off heating or cooling systems when not required, contributing to a sustainable future.

3. Improved Accessibility

- The system is highly beneficial for individuals with physical disabilities or the elderly. Voice commands and IoT-based controls empower them to operate devices without physical effort.
- The inclusion of physical switches ensures inclusivity, catering to all user preferences.

4. Safety and Security

- Automated control of appliances reduces the risk of electrical accidents caused by human errors, such as forgetting to turn off devices.
- Remote monitoring allows users to ensure their appliances are off when they leave the house, minimizing fire hazards.

5. Technological Awareness and Adoption

- This project promotes awareness and adoption of IoT technology in daily life. It serves as a stepping stone for society to embrace smart technologies for home management.

- Educates individuals on how to leverage technology for improving quality of life and environmental responsibility.

6. Cost Savings

- By minimizing energy waste and optimizing appliance usage, this system reduces electricity bills, making it economically beneficial for households.

7. Environmental Impact

- Smart control of devices contributes to a greener society by reducing the carbon footprint associated with excessive energy use.
- Encourages sustainable living practices among users.

8. Inspiration for Future Innovations

- This project can inspire students, hobbyists, and professionals to explore IoT, automation, and renewable energy integration.
- Demonstrates how accessible components like ESP32 and relay modules can be used to create impactful solutions.

VIII. CONCLUSION

The **Smart SynEdge Platform** effectively demonstrates the potential of IoT in enhancing home automation. By integrating ESP32, Arduino IoT Cloud, and voice assistants, this project offers a versatile and reliable solution for appliance control. The inclusion of offline functionality ensures usability in various scenarios, while energy efficiency and cost-effectiveness make it an ideal choice for modern households.

IX. REFERENCES

- Wright, P., & Manieri, A. (2014, April). Internet of Things in the Cloud. In *Proc. of the 4th Int. Conf. on Cloud Computing and Services Science*.
- Oner, V. O. (2021). *Developing IoT Projects with ESP32: Automate your home or business with inexpensive Wi-Fi devices*. Packt Publishing Ltd.
- Gay, W., & Gay, W. (2018). DHT11 sensor. *Advanced Raspberry Pi: Raspbian Linux and GPIO Integration*, 399-418.
- Voice Ass Sutar, P., Sarkar, S. A., Tagare, A., & Pawar, A. (2024, June). A Review on IoT-Enabled Smart Homes Using AI. In *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (pp. 1-6). IEEE.

X. APPENDIX

ESP32 Code :

```
// ************************* Include Libraries *************************
#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>
#include <DHT.h>

// ******************** WiFi and Device Configuration ********************
const char DEVICE_LOGIN_NAME[]  = "f5f5384c-9ac9-48f5-af57-275798f9795b"; // Enter DEVICE ID
const char SSID[]               = "Arain House";    // Enter WiFi SSID (name)
const char PASS[]               = "###########";    // Enter WiFi password
const char DEVICE_KEY[]         = "#ZIN?d672pLx?lhqWRx?Uv@kQ";    // Enter Secret device password (Secret Key)

// ***************************** Pin Definitions *****************************
#define DHTPIN            4 // D4  pin connected with DHT
#define RelayPin1         23 // D23
#define RelayPin2         19 // D19
#define RelayPin3         18 // D18
#define RelayPin4         5  // D5
#define SwitchPin1        13 // D13
#define SwitchPin2        12 // D12
#define SwitchPin3        14 // D14
#define SwitchPin4        27 // D27
#define wifiLed           2  // D2

// Uncomment the type of DHT sensor you're using
#define DHTTYPE DHT11       // DHT 11

// ************************* Global Variables *************************
DHT dht(DHTPIN, DHTTYPE);
int toggleState_1 = 0;
int toggleState_2 = 0;
int toggleState_3 = 0;
int toggleState_4 = 0;
bool SwitchState_1 = LOW;
bool SwitchState_2 = LOW;
bool SwitchState_3 = LOW;
bool SwitchState_4 = LOW;
float temperature1 = 0;
float humidity1   = 0;
int reconnectFlag = 0;

// ********************** Cloud Property Declarations **********************
CloudSwitch light1;
CloudSwitch light2;
CloudSwitch light3;
CloudSwitch light4;
CloudTemperatureSensor temperature;
```

```
// ********** Function Prototypes **********
void onLight1Change();
void onLight2Change();
void onLight3Change();
void onLight4Change();
void initProperties();
void readSensor();
void sendSensor();
void relayOnOff(int relay);
void manual_control();
void doThisOnConnect();
void doThisOnSync();
void doThisOnDisconnect();

// ********** Arduino IoT Cloud Properties Initialization **********
void initProperties() {
  ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
  ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
  ArduinoCloud.addProperty(light1, READWRITE, ON_CHANGE, onLight1Change);
  ArduinoCloud.addProperty(light2, READWRITE, ON_CHANGE, onLight2Change);
  ArduinoCloud.addProperty(light3, READWRITE, ON_CHANGE, onLight3Change);
  ArduinoCloud.addProperty(light4, READWRITE, ON_CHANGE, onLight4Change);
  ArduinoCloud.addProperty(temperature, READ, 8 * SECONDS, NULL); // Update temperature every 8
seconds
}

// ********** WiFi Connection Setup **********
WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);

// ********** Sensor Reading **********
void readSensor() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  } else {
    humidity1 = h;
    temperature = t;
  }
}

// ********** Sending Sensor Data **********
void sendSensor() {
  readSensor();
}

// ********** Manual Relay Control **********
```

```
void relayOnOff(int relay) {
  switch (relay) {
    case 1:
      if (toggleState_1 == 0) {
        digitalWrite(RelayPin1, LOW);
        toggleState_1 = 1;
        Serial.println("Device1 ON");
      } else {
        digitalWrite(RelayPin1, HIGH);
        toggleState_1 = 0;
        Serial.println("Device1 OFF");
      }
      delay(100);
      break;
    case 2:
      if (toggleState_2 == 0) {
        digitalWrite(RelayPin2, LOW);
        toggleState_2 = 1;
        Serial.println("Device2 ON");
      } else {
        digitalWrite(RelayPin2, HIGH);
        toggleState_2 = 0;
        Serial.println("Device2 OFF");
      }
      delay(100);
      break;
    case 3:
      if (toggleState_3 == 0) {
        digitalWrite(RelayPin3, LOW);
        toggleState_3 = 1;
        Serial.println("Device3 ON");
      } else {
        digitalWrite(RelayPin3, HIGH);
        toggleState_3 = 0;
        Serial.println("Device3 OFF");
      }
      delay(100);
      break;
    case 4:
      if (toggleState_4 == 0) {
        digitalWrite(RelayPin4, LOW);
        toggleState_4 = 1;
        Serial.println("Device4 ON");
      } else {
        digitalWrite(RelayPin4, HIGH);
        toggleState_4 = 0;
        Serial.println("Device4 OFF");
      }
      delay(100);
      break;
    default: break;
  }
```

```
}

// ********** Manual Switch Control **********
void manual_control() {
  if (digitalRead(SwitchPin1) == LOW && SwitchState_1 == LOW) {
    digitalWrite(RelayPin1, LOW);
    toggleState_1 = 1;
    SwitchState_1 = HIGH;
    light1 = toggleState_1;
    Serial.println("Switch-1 on");
  }
  // (similar code for other switches)
}

// ********** IoT Cloud Connection Callbacks **********
void doThisOnConnect() {
  Serial.println("Board successfully connected to Arduino IoT Cloud");
  digitalWrite(wifiLed, HIGH); // Turn on WiFi LED
}

void doThisOnSync() {
  Serial.println("Thing Properties synchronized");
}

void doThisOnDisconnect() {
  Serial.println("Board disconnected from Arduino IoT Cloud");
  digitalWrite(wifiLed, LOW); // Turn off WiFi LED
}

// ********** Arduino Setup Function **********
void setup() {
  Serial.begin(115200);
  delay(1500);

  initProperties();
  dht.begin();

  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  ArduinoCloud.addCallback(ArduinoIoTCloudEvent::CONNECT, doThisOnConnect);
  ArduinoCloud.addCallback(ArduinoIoTCloudEvent::SYNC, doThisOnSync);
  ArduinoCloud.addCallback(ArduinoIoTCloudEvent::DISCONNECT, doThisOnDisconnect);

  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

  pinMode(RelayPin1, OUTPUT);
  pinMode(RelayPin2, OUTPUT);
  pinMode(RelayPin3, OUTPUT);
  pinMode(RelayPin4, OUTPUT);

  pinMode(wifiLed, OUTPUT);
```

```
  pinMode(SwitchPin1, INPUT_PULLUP);
  pinMode(SwitchPin2, INPUT_PULLUP);
  pinMode(SwitchPin3, INPUT_PULLUP);
  pinMode(SwitchPin4, INPUT_PULLUP);

  digitalWrite(RelayPin1, HIGH);
  digitalWrite(RelayPin2, HIGH);
  digitalWrite(RelayPin3, HIGH);
  digitalWrite(RelayPin4, HIGH);
}

// ********** Arduino Loop Function **********
void loop() {
  ArduinoCloud.update();
  manual_control();
  sendSensor();

  // Optional: Print WiFi status periodically
  static unsigned long lastWiFiCheck = 0;
  if (millis() - lastWiFiCheck > 5000) {
    lastWiFiCheck = millis();
    if (WiFi.status() == WL_CONNECTED) {
      Serial.println("WiFi connected: " + String(WiFi.localIP()));
    } else {
      Serial.println("WiFi not connected!");
    }
  }
}

// ********** Cloud Property Change Handlers **********
void onLight1Change() {
  if (light1 == 1) {
    digitalWrite(RelayPin1, LOW);
    Serial.println("Device1 ON");
    toggleState_1 = 1;
  } else {
    digitalWrite(RelayPin1, HIGH);
    Serial.println("Device1 OFF");
    toggleState_1 = 0;
  }
}

void onLight2Change() {
  if (light2 == 1) {
    digitalWrite(RelayPin2, LOW);
    Serial.println("Device2 ON");
    toggleState_2 = 1;
  } else {
    digitalWrite(RelayPin2, HIGH);
    Serial.println("Device2 OFF");
    toggleState_2 = 0;
```

```
  }
}

void onLight3Change() {
  if (light3 == 1) {
    digitalWrite(RelayPin3, LOW);
    Serial.println("Device3 ON");
    toggleState_3 = 1;
  } else {
    digitalWrite(RelayPin3, HIGH);
    Serial.println("Device3 OFF");
    toggleState_3 = 0;
  }
}

void onLight4Change() {
  if (light4 == 1) {
    digitalWrite(RelayPin4, LOW);
    Serial.println("Device4 ON");
    toggleState_4 = 1;
  } else {
    digitalWrite(RelayPin4, HIGH);
    Serial.println("Device4 OFF");
    toggleState_4 = 0;
  }
}
```

Here’s the link: HOME code.txt - Google Drive

The End.