MANIPAL UNIVERSITY
DUBAI CAMPUS

Mini Project (PROJECT)

Home automation system using nodemcu esp8266

Khalid ALI

Dept. of Electrical and electronics Engineering, School of Engg. & IT, Manipal Academy of Higher Education, Dubai.

<u>Abstract:</u>

This project shows a simple home automation system that uses two ESP8266 NodeMCU boards to wirelessly monitor and operate household appliances and sensors. The configuration consists of a primary NodeMCU (A) connected to a laptop, which serves as the main controller, and a secondary NodeMCU (B), which is linked to a gas sensor, PIR motion sensor, ultrasonic sensor, and buzzer. NodeMCU (B) serves as a Wi-Fi access point and web server, allowing it to accept commands and relay real-time sensor data via Wi-Fi.

NodeMCU (A) connects to NodeMCU (B's) access point and obtains sensor data (such as gas levels, motion, and distance) using HTTP requests before controlling the buzzer based on the results. Data is shown on the laptop, allowing for real-time monitoring and remote operation. This arrangement provides a flexible, scalable foundation for

smart home systems, focussing wireless connection between IoT devices to enable successful home automation.

Introduction:

In this project, we look at a basic home automation system built around two ESP8266 NodeMCU microcontrollers that interact wirelessly to monitor and control various appliances and sensors in a home. NodeMCU (A) serves as the principal controller and is linked to a laptop, whilst NodeMCU (B) interfaces with sensors and actuators like a gas sensor, PIR motion detector, ultrasonic distance sensor, and buzzer. NodeMCU (B) can receive commands and transmit sensor data to NodeMCU (A) via a wireless network when configured as a WiFi access point and web server.

The system provides real-time monitoring and remote control of appliances, with the NodeMCU (A) capable of sending requests to turn devices on or off based on sensor data. This project shows how IoT devices such as the ESP8266 may be used to form a basic, adaptable smart home system, increasing the flexibility and capability of home automation via simple yet effective wireless communication.

Objectives

1. Develop a cost-effective and efficient home automation system.
2. Utilize IoT technology to control and monitor devices wirelessly.
3. Provide an interactive system using HTTP requests for communication between microcontrollers.

System Overview

- **NodeMCU A (Client):** Handles user commands entered via the Serial Monitor and communicates with NodeMCU B over HTTP.
- **NodeMCU B (Server):** Receives commands from NodeMCU A to control devices and provide sensor readings.

Components Used

1. NodeMCU ESP8266 (x2):
2. LED:
3. Buzzer:
4. PIR Motion Sensor:
5. Ultrasonic Sensor (HC-SR04):
6. Gas/Smoke Detector (MQ-2):
7. JUMPER WIRES
8. Breadboard

Software

1. **Arduino IDE:** ○ For programming NodeMCU A and B.
2. **ESP8266WiFi Library:** ○ Manages Wi-Fi connection.
3. **ESP8266WebServer Library:**
   ○ Enables NodeMCU B to act as a web server.

Circuit Diagram

1. **LED**:
   Positive → D2 (GPIO 4) Negative
   → 220Ω resistor → GND
2. **Buzzer**:
   Positive → D3 (GPIO 0)
   Negative → GND
3. **Ultrasonic Sensor**:
   VCC → 5V
   GND → GND
   TRIG → D5 (GPIO 14)
   ECHO → D6 (GPIO 12) 4.

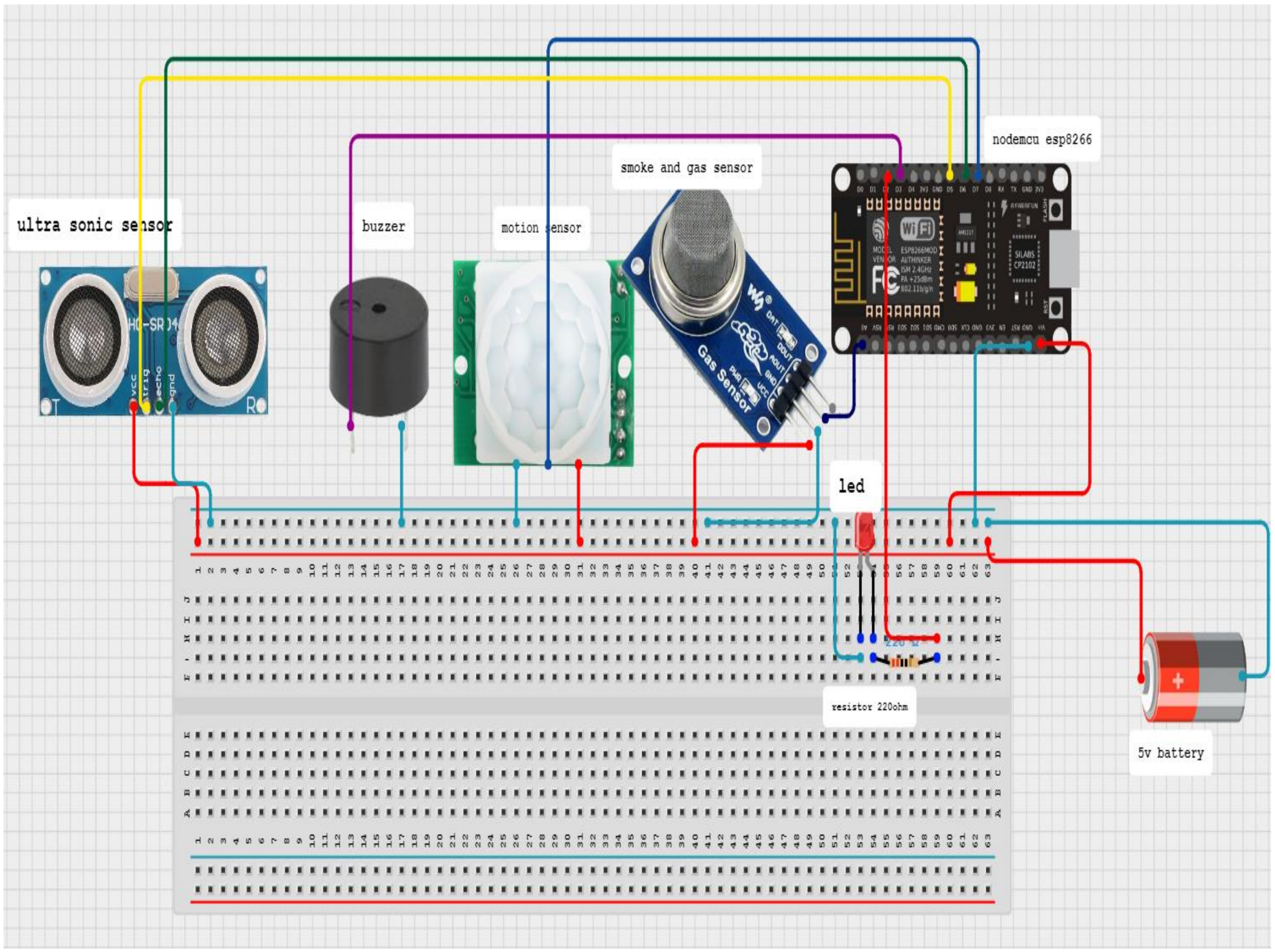
Smoke Detector (MQ2):
   VCC → 3.3V
   GND → GND Analog
   OUT → A0

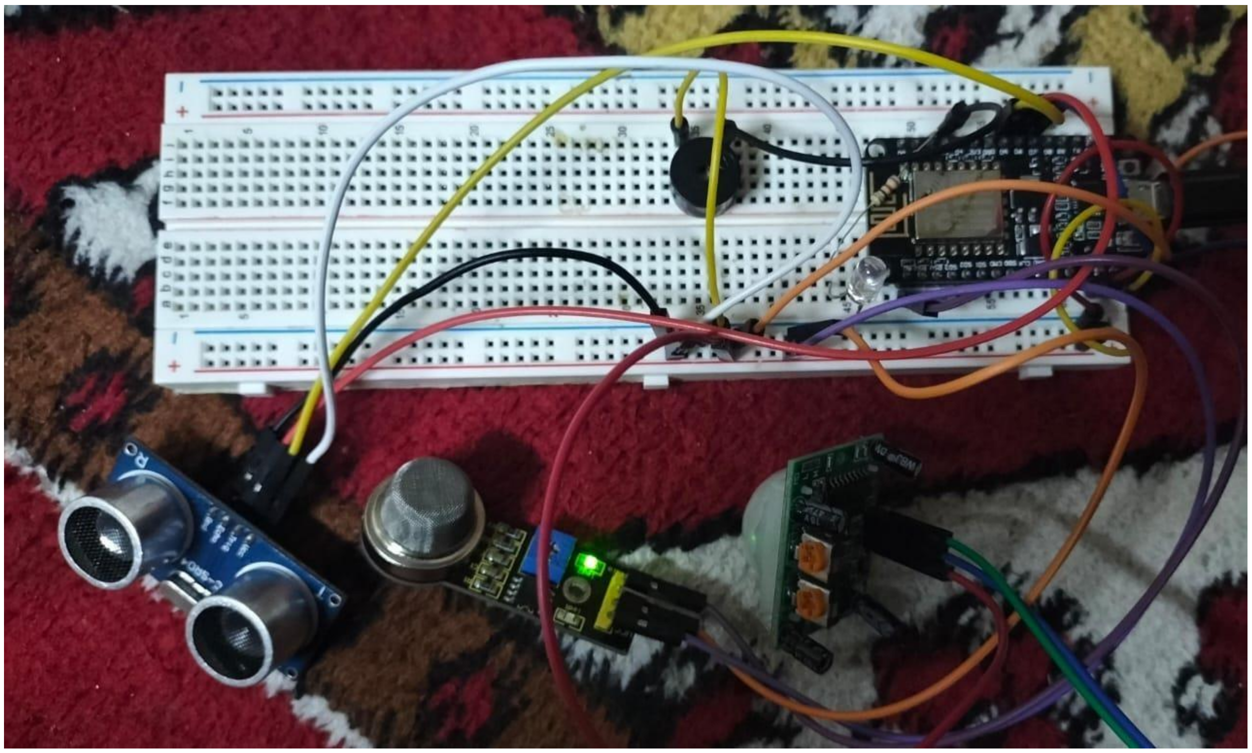
5. **PIR Sensor**:

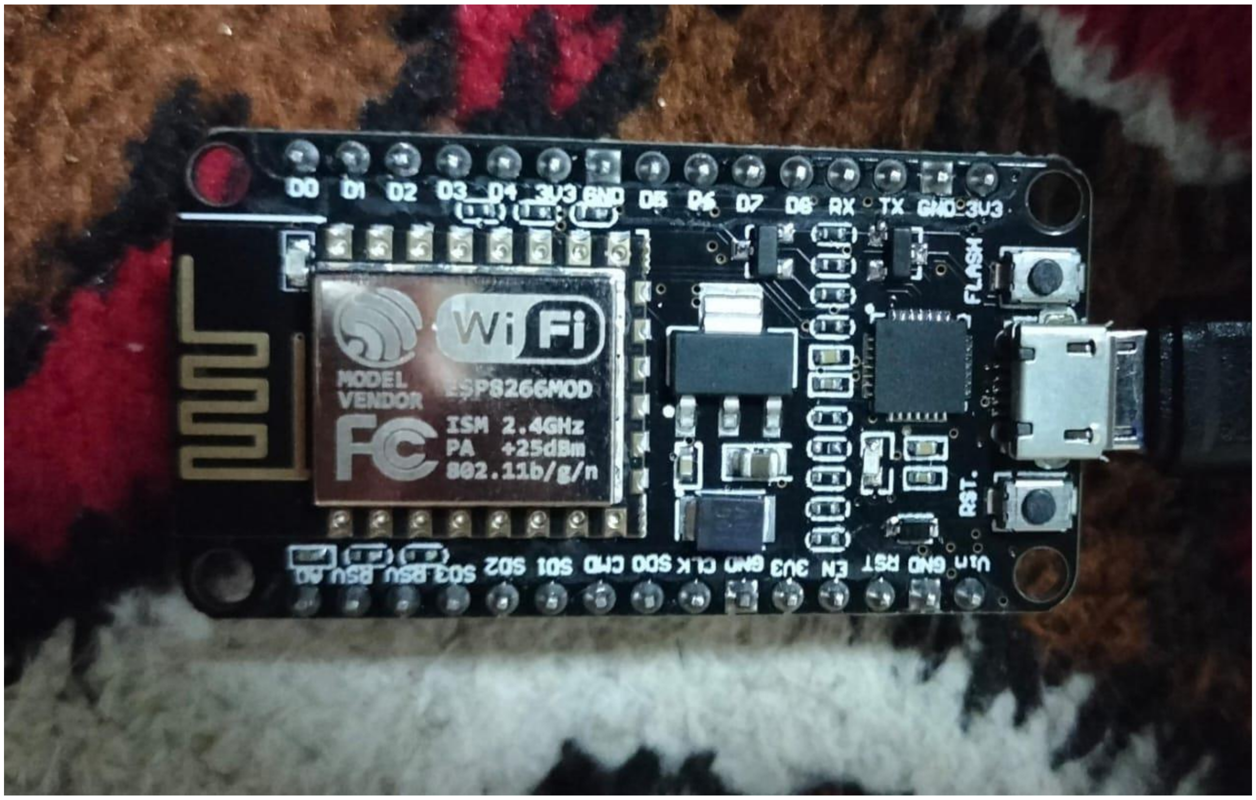
VCC → 5V

GND → GND

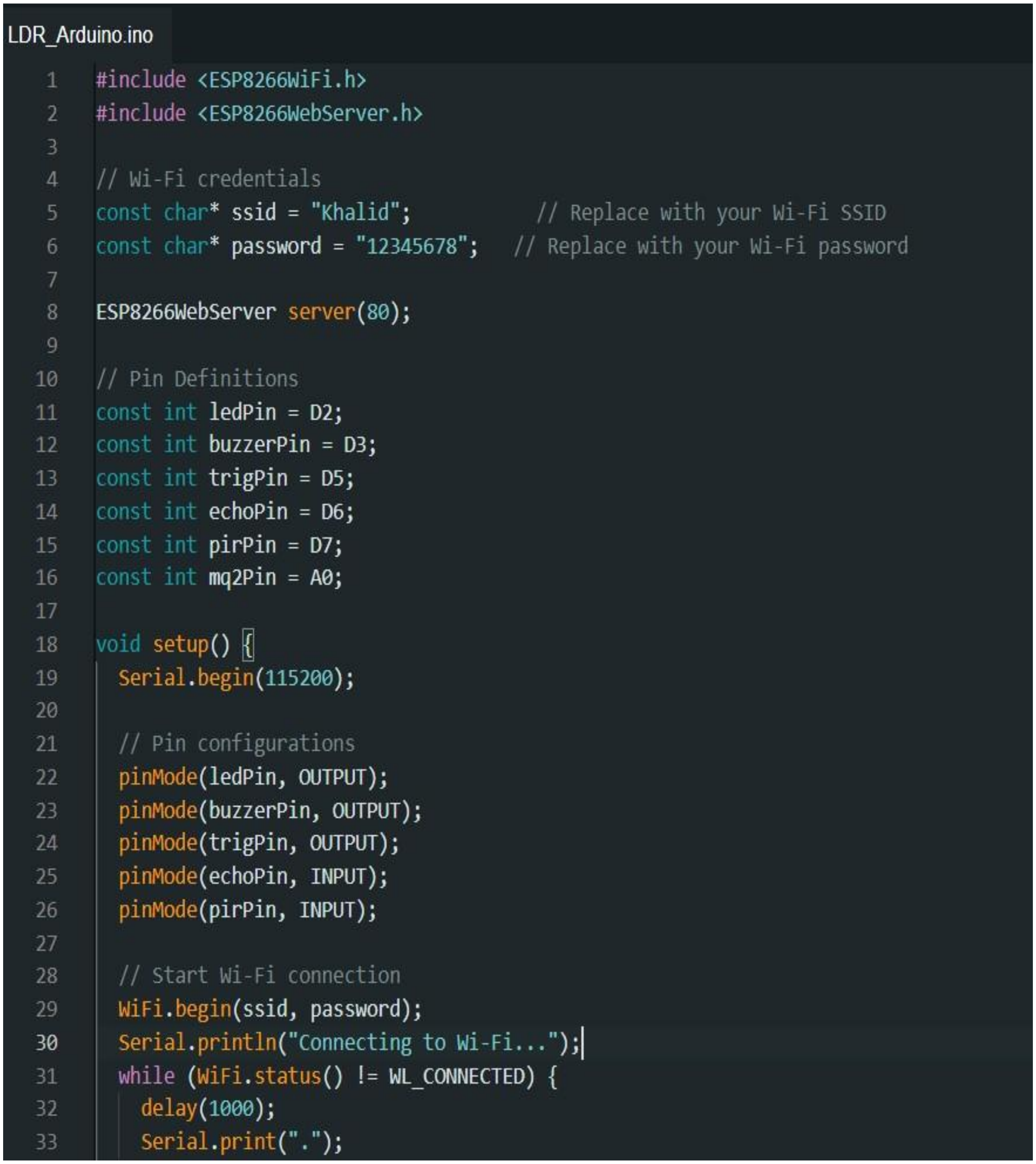
OUT → D7 (GPIO 13)



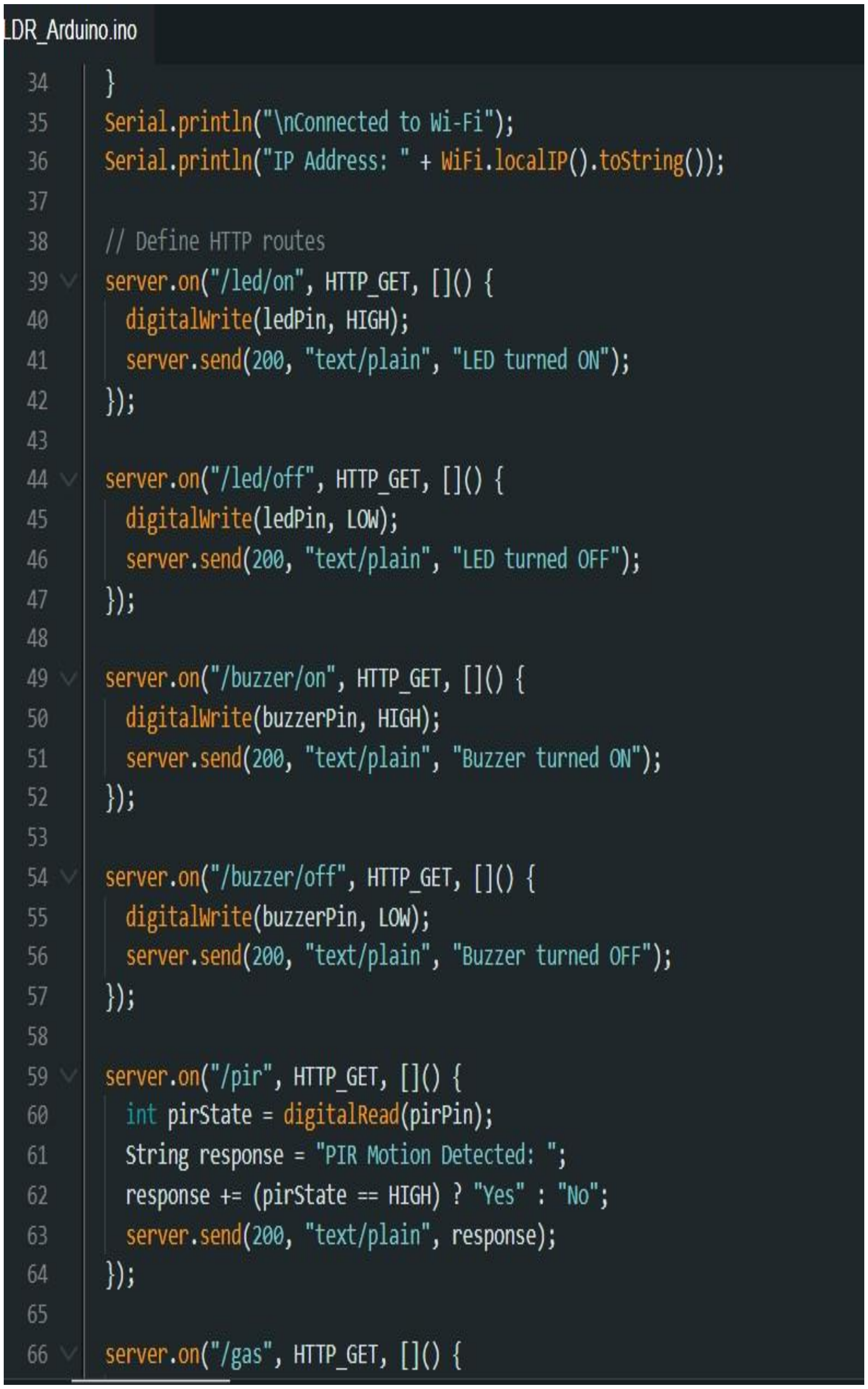




Code for nodemcu (b)

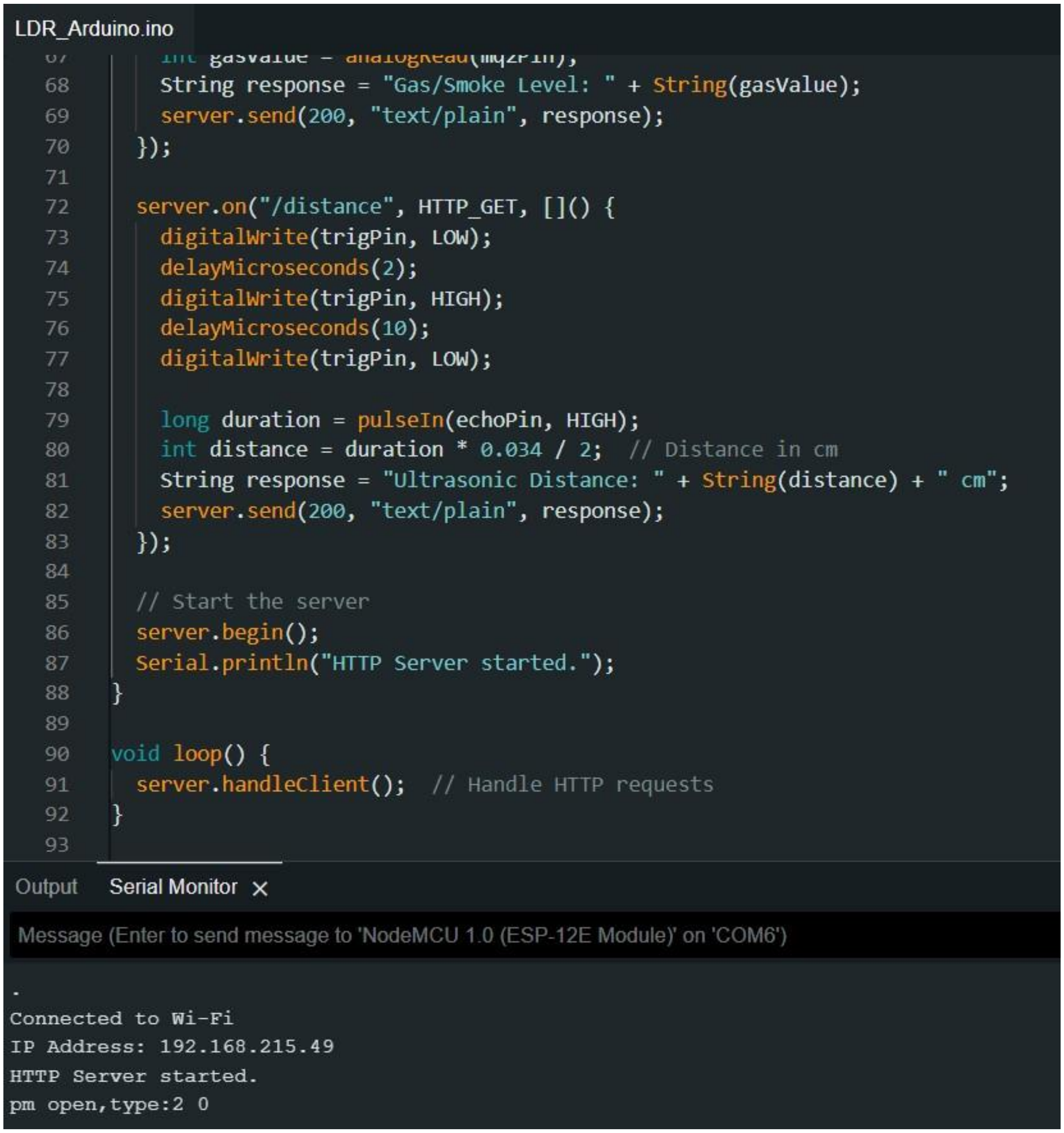

LDR_Arduino.ino

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

// Wi-Fi credentials
const char* ssid = "Khalid";           // Replace with your Wi-Fi SSID
const char* password = "12345678";   // Replace with your Wi-Fi password

ESP8266WebServer server(80);

// Pin Definitions
const int ledPin = D2;
const int buzzerPin = D3;
const int trigPin = D5;
const int echoPin = D6;
const int pirPin = D7;
const int mq2Pin = A0;

void setup() {
  Serial.begin(115200);

  // Pin configurations
  pinMode(ledPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(pirPin, INPUT);

  // Start Wi-Fi connection
  WiFi.begin(ssid, password);
  Serial.println("Connecting to Wi-Fi...");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
```



LDR_Arduino.ino

```
}
Serial.println("\nConnected to Wi-Fi");
Serial.println("IP Address: " + WiFi.localIP().toString());

// Define HTTP routes
server.on("/led/on", HTTP_GET, []() {
  digitalWrite(ledPin, HIGH);
  server.send(200, "text/plain", "LED turned ON");
});

server.on("/led/off", HTTP_GET, []() {
  digitalWrite(ledPin, LOW);
  server.send(200, "text/plain", "LED turned OFF");
});

server.on("/buzzer/on", HTTP_GET, []() {
  digitalWrite(buzzerPin, HIGH);
  server.send(200, "text/plain", "Buzzer turned ON");
});

server.on("/buzzer/off", HTTP_GET, []() {
  digitalWrite(buzzerPin, LOW);
  server.send(200, "text/plain", "Buzzer turned OFF");
});

server.on("/pir", HTTP_GET, []() {
  int pirState = digitalRead(pirPin);
  String response = "PIR Motion Detected: ";
  response += (pirState == HIGH) ? "Yes" : "No";
  server.send(200, "text/plain", response);
});

server.on("/gas", HTTP_GET, []() {
```



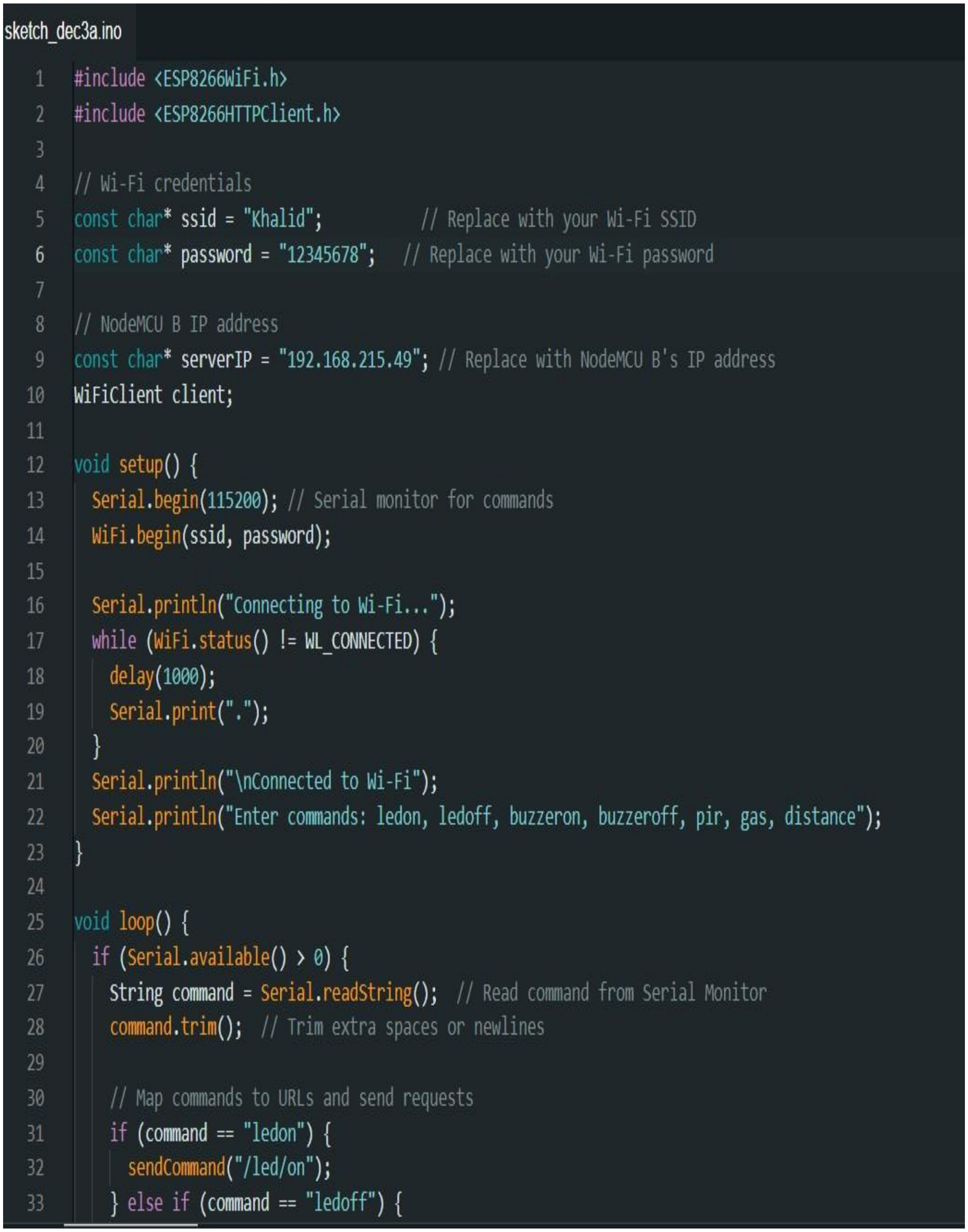
LDR_Arduino.ino

```
    int gasValue = analogRead(mq2Pin);
    String response = "Gas/Smoke Level: " + String(gasValue);
    server.send(200, "text/plain", response);
  });

  server.on("/distance", HTTP_GET, []() {
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    long duration = pulseIn(echoPin, HIGH);
    int distance = duration * 0.034 / 2;  // Distance in cm
    String response = "Ultrasonic Distance: " + String(distance) + " cm";
    server.send(200, "text/plain", response);
  });

  // Start the server
  server.begin();
  Serial.println("HTTP Server started.");
}

void loop() {
  server.handleClient();  // Handle HTTP requests
}
```

Output
Serial Monitor
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM6')

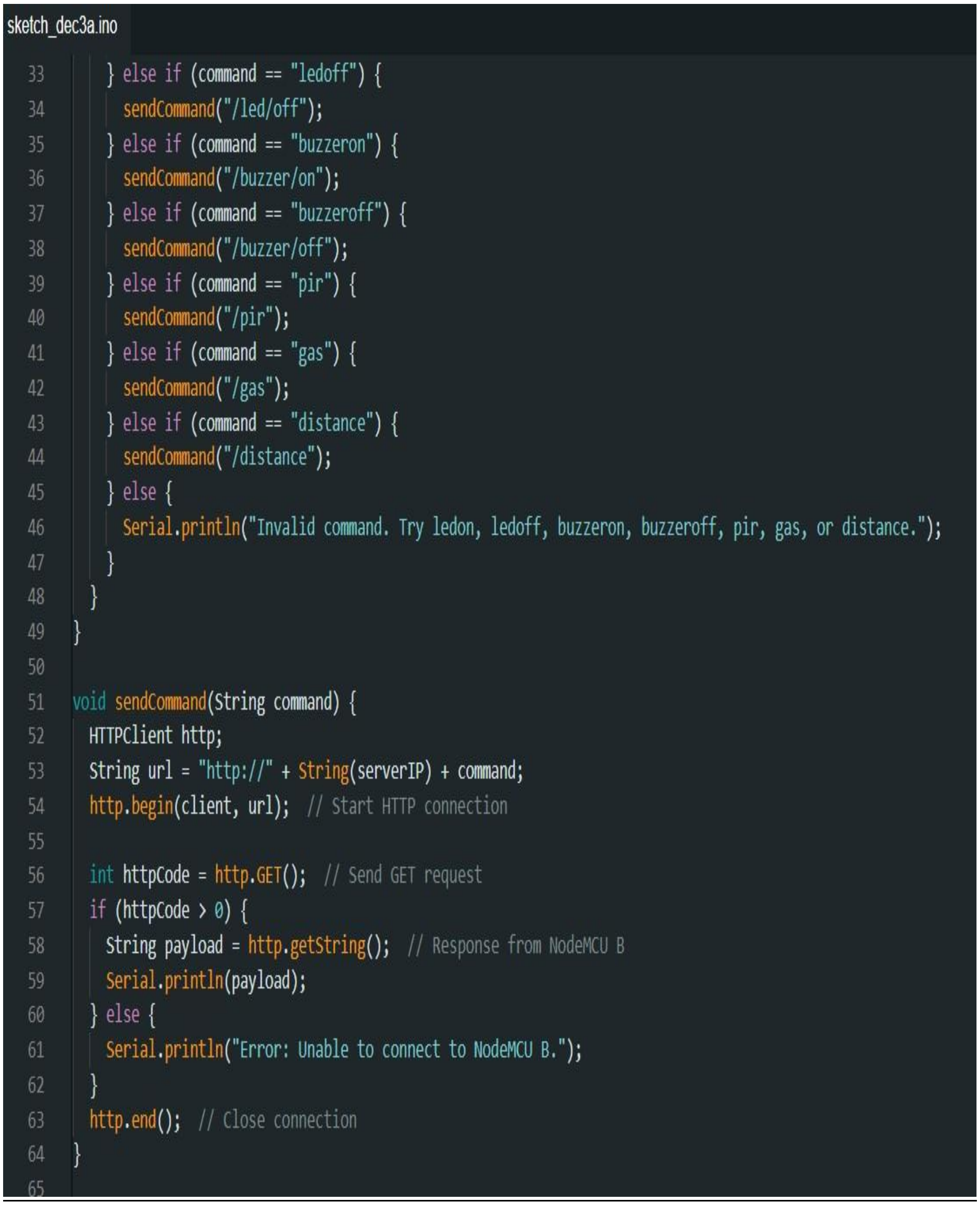
```
.
Connected to Wi-Fi
IP Address: 192.168.215.49
HTTP Server started.
pm open,type:2 0
```

Code for nodemcu (a)



sketch_dec3a.ino

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

// Wi-Fi credentials
const char* ssid = "Khalid";           // Replace with your Wi-Fi SSID
const char* password = "12345678";   // Replace with your Wi-Fi password

// NodeMCU B IP address
const char* serverIP = "192.168.215.49"; // Replace with NodeMCU B's IP address
WiFiClient client;

void setup() {
  Serial.begin(115200); // Serial monitor for commands
  WiFi.begin(ssid, password);

  Serial.println("Connecting to Wi-Fi...");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("\nConnected to Wi-Fi");
  Serial.println("Enter commands: ledon, ledoff, buzzeron, buzzeroff, pir, gas, distance");
}

void loop() {
  if (Serial.available() > 0) {
    String command = Serial.readString();  // Read command from Serial Monitor
    command.trim();  // Trim extra spaces or newlines

    // Map commands to URLs and send requests
    if (command == "ledon") {
      sendCommand("/led/on");
    } else if (command == "ledoff") {
```


sketch_dec3a.ino

```
    } else if (command == "ledoff") {
      sendCommand("/led/off");
    } else if (command == "buzzeron") {
      sendCommand("/buzzer/on");
    } else if (command == "buzzeroff") {
      sendCommand("/buzzer/off");
    } else if (command == "pir") {
      sendCommand("/pir");
    } else if (command == "gas") {
      sendCommand("/gas");
    } else if (command == "distance") {
      sendCommand("/distance");
    } else {
      Serial.println("Invalid command. Try ledon, ledoff, buzzeron, buzzeroff, pir, gas, or distance.");
    }
  }
}

void sendCommand(String command) {
  HTTPClient http;
  String url = "http://" + String(serverIP) + command;
  http.begin(client, url);  // Start HTTP connection

  int httpCode = http.GET();  // Send GET request
  if (httpCode > 0) {
    String payload = http.getString();  // Response from NodeMCU B
    Serial.println(payload);
  } else {
    Serial.println("Error: Unable to connect to NodeMCU B.");
  }
  http.end();  // Close connection
}

```

Applications

1. Smart homes for appliance control and monitoring.
2. Real-time safety and security systems using gas and motion sensors.
3. Industrial automation and monitoring.

Conclusion

This project demonstrates a scalable and efficient IoT solution for home automation. It leverages Wi-Fi communication, making it a costeffective and user-friendly system.