1. void setup() {

// Set the LED pin as an output

pinMode(13, OUTPUT);

}

void loop() {

// Turn the LED on

digitalWrite(13, HIGH);

// Wait for 1 second

delay(1000);

// Turn the LED off

digitalWrite(13, LOW);

// Wait for 1 second

delay(1000);

}

2. const int ledPin = 2; // Define the GPIO pin for the LED

void setup() {

// Set the LED pin as an output

pinMode(ledPin, OUTPUT);

}

void loop() {

// Turn the LED on

digitalWrite(ledPin, HIGH);

// Wait for 1 second

delay(1000);

// Turn the LED off

digitalWrite(ledPin, LOW);

// Wait for 1 second

delay(1000);

}

3. const int ledPin = D2; // Define the GPIO pin for the LED

void setup() {

// Set the LED pin as an output

pinMode(ledPin, OUTPUT);

}

void loop() {

// Turn the LED on

digitalWrite(ledPin, HIGH);

// Wait for 1 second

delay(1000);

// Turn the LED off

digitalWrite(ledPin, LOW);

// Wait for 1 second

delay(1000);

}

4. const int ledPin1 = D2; // GPIO pin for the first LED (Color 1)

const int ledPin2 = D3; // GPIO pin for the second LED (Color 2)

void setup() {

// Set the LED pins as outputs

pinMode(ledPin1, OUTPUT);

pinMode(ledPin2, OUTPUT);

}

void loop() {

// Turn both LEDs on

digitalWrite(ledPin1, HIGH);

digitalWrite(ledPin2, HIGH);

// Wait for 1 second

delay(1000);

// Turn both LEDs off

digitalWrite(ledPin1, LOW);

digitalWrite(ledPin2, LOW);

// Wait for 1 second

delay(1000);

}

5. const int ledPin1 = D2; // GPIO pin for the first LED (Color 1)

const int ledPin2 = D3; // GPIO pin for the second LED (Color 2)

const int buzzerPin = D4; // GPIO pin for the buzzer

void setup() {

// Set the LED pins and buzzer pin as outputs

pinMode(ledPin1, OUTPUT);

pinMode(ledPin2, OUTPUT);

pinMode(buzzerPin, OUTPUT);

}

void loop() {

// Turn on the first LED and the buzzer

digitalWrite(ledPin1, HIGH);

digitalWrite(buzzerPin, HIGH);

delay(500); // Adjust the delay as needed

// Turn off the first LED and the buzzer

digitalWrite(ledPin1, LOW);

digitalWrite(buzzerPin, LOW);

delay(500); // Adjust the delay as needed

// Turn on the second LED and the buzzer

digitalWrite(ledPin2, HIGH);

digitalWrite(buzzerPin, HIGH);

delay(500); // Adjust the delay as needed

// Turn off the second LED and the buzzer

digitalWrite(ledPin2, LOW);

digitalWrite(buzzerPin, LOW);

delay(500); // Adjust the delay as needed

}

6. // Define the buzzer pin

const int buzzerPin = 8; // Change this to the actual pin you've connected the buzzer to

void setup() {

// Set the buzzer pin as an output

pinMode(buzzerPin, OUTPUT);

}

void loop() {

// Play the "Happy Birthday" melody

playHappyBirthday();

delay(2000); // Wait for 2 seconds before repeating the melody

}

// Function to play the "Happy Birthday" melody

void playHappyBirthday() {

// Notes in the melody

int melody[] = {

NOTE\_G4, NOTE\_G4, NOTE\_A4, NOTE\_G4, NOTE\_C5, NOTE\_B4,

NOTE\_G4, NOTE\_G4, NOTE\_A4, NOTE\_G4, NOTE\_D5, NOTE\_C5,

NOTE\_G4, NOTE\_G4, NOTE\_G5, NOTE\_E5, NOTE\_C5, NOTE\_B4, NOTE\_A4,

NOTE\_F5, NOTE\_F5, NOTE\_E5, NOTE\_C5, NOTE\_D5, NOTE\_C5

};

// Note durations: 4 = quarter note, 8 = eighth note, etc.

int noteDurations[] = {

4, 4, 4, 4, 4, 2,

4, 4, 4, 4, 4, 2,

4, 4, 4, 4, 4, 4, 2,

4, 4, 4, 4, 4, 2

};

// Play the melody

for (int i = 0; i < sizeof(melody) / sizeof(melody[0]); i++) {

int noteDuration = 1000 / noteDurations[i];

tone(buzzerPin, melody[i], noteDuration);

int pauseBetweenNotes = noteDuration \* 1.30; // Add a pause between notes

delay(pauseBetweenNotes);

noTone(buzzerPin); // Stop the tone

}

}

7. const int buttonPin = 2; // Digital pin connected to the push-button

const int ledPin = 13; // Digital pin connected to the LED

void setup() {

pinMode(buttonPin, INPUT\_PULLUP); // Set the button pin as input with internal pull-up resistor

pinMode(ledPin, OUTPUT); // Set the LED pin as an output

}

void loop() {

// Read the state of the button

int buttonState = digitalRead(buttonPin);

// If the button is pressed (LOW state because of the pull-up resistor)

if (buttonState == LOW) {

digitalWrite(ledPin, HIGH); // Turn the LED on

} else {

digitalWrite(ledPin, LOW); // Turn the LED off

}

}

8. #include <DHT.h>

// Constants

#define DHTPIN 2 // Digital pin connected to the DHT11 sensor

#define DHTTYPE DHT11 // Type of DHT sensor (DHT11 or DHT22)

// Initialize DHT sensor

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600); // Initialize serial communication

dht.begin(); // Initialize DHT sensor

}

void loop() {

// Read temperature and humidity from DHT sensor

float temperature = dht.readTemperature(); // Read temperature in Celsius

float humidity = dht.readHumidity(); // Read humidity in percentage

// Check if the readings are valid

if (isnan(temperature) || isnan(humidity)) {

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

} else {

// Print temperature and humidity to the serial monitor

Serial.print("Temperature: ");

Serial.print(temperature);

Serial.println(" °C");

Serial.print("Humidity: ");

Serial.print(humidity);

Serial.println(" %");

}

delay(2000); // Delay for 2 seconds before taking another reading

}

9. #include <DHT.h>

// Constants

#define DHTPIN 2 // Digital pin connected to the DHT11 sensor

#define DHTTYPE DHT11 // Type of DHT sensor (DHT11 or DHT22)

#define RED\_LED\_PIN 7 // Digital pin for the red LED

#define GREEN\_LED\_PIN 6 // Digital pin for the green LED

#define BUZZER\_PIN 8 // Digital pin for the buzzer

// Initialize DHT sensor

DHT dht(DHTPIN, DHTTYPE);

void setup() {

pinMode(RED\_LED\_PIN, OUTPUT);

pinMode(GREEN\_LED\_PIN, OUTPUT);

pinMode(BUZZER\_PIN, OUTPUT);

dht.begin(); // Initialize DHT sensor

}

void loop() {

// Read temperature from DHT sensor

float temperature = dht.readTemperature();

// Check if the temperature is above 30°C

if (temperature > 30.0) {

digitalWrite(RED\_LED\_PIN, HIGH); // Turn on the red LED

digitalWrite(GREEN\_LED\_PIN, LOW); // Turn off the green LED

digitalWrite(BUZZER\_PIN, HIGH); // Turn on the buzzer

} else {

digitalWrite(RED\_LED\_PIN, LOW); // Turn off the red LED

digitalWrite(GREEN\_LED\_PIN, HIGH); // Turn on the green LED

digitalWrite(BUZZER\_PIN, LOW); // Turn off the buzzer

}

delay(1000); // Delay for 1 second between readings (adjust as needed)

}

10. #include <DHT.h>

// Constants

#define DHTPIN 2 // Digital pin connected to the DHT11 sensor

#define DHTTYPE DHT11 // Type of DHT sensor (DHT11 or DHT22)

// Initialize DHT sensor

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600); // Initialize serial communication

dht.begin(); // Initialize DHT sensor

}

void loop() {

// Variables for temperature and humidity readings

float temperature1, temperature2, temperature3;

float humidityReadings[10];

float humiditySum = 0.0;

int temperatureSum = 0;

// Read three temperature values

temperature1 = dht.readTemperature();

delay(1000);

temperature2 = dht.readTemperature();

delay(1000);

temperature3 = dht.readTemperature();

delay(1000);

// Read ten humidity values and calculate the sum

for (int i = 0; i < 10; i++) {

humidityReadings[i] = dht.readHumidity();

humiditySum += humidityReadings[i];

delay(1000);

}

// Calculate the average humidity

float humidityAverage = humiditySum / 10;

// Calculate the sum of three temperatures

temperatureSum = int(temperature1) + int(temperature2) + int(temperature3);

// Check if the sum of temperatures is odd or even

Serial.print("Sum of temperatures: ");

Serial.println(temperatureSum);

if (temperatureSum % 2 == 0) {

Serial.println("The sum of temperatures is even.");

} else {

Serial.println("The sum of temperatures is odd.");

}

// Print the average humidity

Serial.print("Average humidity: ");

Serial.print(humidityAverage);

Serial.print(" %");

delay(5000); // Delay for 5 seconds before repeating the readings

}

11. #include <DHT.h>

// Constants

#define DHTPIN 2 // Digital pin connected to the DHT11 sensor

#define DHTTYPE DHT11 // Type of DHT sensor (DHT11 or DHT22)

// Initialize DHT sensor

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600); // Initialize serial communication

dht.begin(); // Initialize DHT sensor

}

void loop() {

// Variables for temperature readings

float temperatureReadings[5];

float temperatureSum = 0.0;

int averageTemperature;

// Read five temperature values and calculate the sum

for (int i = 0; i < 5; i++) {

temperatureReadings[i] = dht.readTemperature();

temperatureSum += temperatureReadings[i];

delay(1000);

}

// Calculate the average temperature

averageTemperature = int(temperatureSum / 5);

// Print the average temperature

Serial.print("Average temperature: ");

Serial.println(averageTemperature);

// Print the Fibonacci series up to the average temperature value

int a = 0, b = 1, c;

Serial.print("Fibonacci series up to the average temperature: ");

Serial.print(a); // Print the first number

while (a + b <= averageTemperature) {

c = a + b;

Serial.print(", ");

Serial.print(c);

a = b;

b = c;

}

Serial.println();

delay(5000); // Delay for 5 seconds before repeating the readings

}

12. // Define the pin connected to the IR sensor

const int irSensorPin = 7;

void setup() {

Serial.begin(9600); // Initialize serial communication

pinMode(irSensorPin, INPUT); // Set the IR sensor pin as an input

}

void loop() {

// Read the state of the IR sensor

int irSensorState = digitalRead(irSensorPin);

// Print the IR sensor reading to the serial monitor

Serial.print("IR Sensor Reading: ");

Serial.println(irSensorState);

delay(1000); // Delay for 1 second before taking another reading

}

13. // Define pin connections

const int irTransmitterPin = 9;

const int irReceiverPin = 8;

const int redLedPin = 7;

const int whiteLedPin = 6;

const int buzzerPin = 5;

void setup() {

pinMode(irTransmitterPin, OUTPUT); // IR transmitter as an output

pinMode(irReceiverPin, INPUT); // IR receiver as an input

pinMode(redLedPin, OUTPUT); // Red LED as an output

pinMode(whiteLedPin, OUTPUT); // White LED as an output

pinMode(buzzerPin, OUTPUT); // Buzzer as an output

digitalWrite(irTransmitterPin, LOW); // Initially turn off the IR transmitter

digitalWrite(redLedPin, LOW); // Initially turn off the red LED

digitalWrite(whiteLedPin, LOW); // Initially turn off the white LED

digitalWrite(buzzerPin, LOW); // Initially turn off the buzzer

Serial.begin(9600); // Initialize serial communication

}

void loop() {

// Read the state of the IR receiver

int irReceiverState = digitalRead(irReceiverPin);

// If the IR receiver detects infrared rays (HIGH state), glow the white LED

if (irReceiverState == HIGH) {

digitalWrite(whiteLedPin, HIGH);

digitalWrite(redLedPin, LOW);

digitalWrite(buzzerPin, LOW); // Turn off the buzzer

} else {

// If no infrared rays detected, glow the red LED and play the buzzer

digitalWrite(whiteLedPin, LOW);

digitalWrite(redLedPin, HIGH);

digitalWrite(buzzerPin, HIGH); // Turn on the buzzer

}

delay(100); // Delay for a short period before checking again

}

14. // Define pin connections

const int trigPin = 9; // Trig pin of the ultrasonic sensor

const int echoPin = 10; // Echo pin of the ultrasonic sensor

// Variables to store distance and duration

long duration;

int distance;

void setup() {

Serial.begin(9600); // Initialize serial communication

pinMode(trigPin, OUTPUT); // Set the trigPin as an output

pinMode(echoPin, INPUT); // Set the echoPin as an input

}

void loop() {

// Trigger a pulse to the ultrasonic sensor

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Measure the duration of the echo pulse

duration = pulseIn(echoPin, HIGH);

// Calculate the distance in centimeters (cm)

distance = duration \* 0.034 / 2;

// Print the distance to the serial monitor

Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

delay(1000); // Delay for 1 second before taking another reading

}

15. // Define pin connections

const int trigPin = 9; // Trig pin of the ultrasonic sensor

const int echoPin = 10; // Echo pin of the ultrasonic sensor

// Variables for ultrasonic sensor

long duration;

int distance;

// Function to calculate the cube of a number

int cube(int num) {

return num \* num \* num;

}

void setup() {

Serial.begin(9600); // Initialize serial communication

pinMode(trigPin, OUTPUT); // Set the trigPin as an output

pinMode(echoPin, INPUT); // Set the echoPin as an input

}

void loop() {

int readings[3]; // Array to store three readings

int sum = 0;

// Take three readings at 2-second intervals

for (int i = 0; i < 3; i++) {

// Trigger a pulse to the ultrasonic sensor

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Measure the duration of the echo pulse

duration = pulseIn(echoPin, HIGH);

// Calculate the distance in centimeters (cm)

distance = duration \* 0.034 / 2;

// Store the reading in the array

readings[i] = distance;

// Print the reading to the serial monitor

Serial.print("Reading ");

Serial.print(i + 1);

Serial.print(": ");

Serial.print(readings[i]);

Serial.println(" cm");

delay(2000); // Wait for 2 seconds before the next reading

}

// Calculate the average of the three readings

for (int i = 0; i < 3; i++) {

sum += readings[i];

}

int average = sum / 3;

// Check if the average is an "Armstrong-like" number

int original = average;

int armstrongSum = 0;

while (original > 0) {

int digit = original % 10;

armstrongSum += cube(digit);

original /= 10;

}

if (armstrongSum == average) {

Serial.print("Average (");

Serial.print(average);

Serial.println(") is an Armstrong-like number.");

} else {

Serial.print("Average (");

Serial.print(average);

Serial.println(") is not an Armstrong-like number.");

}

// Stop further readings

while (true) {

// Do nothing

}

}

16. // Define segment pins

const int segA = 2;

const int segB = 3;

const int segC = 4;

const int segD = 5;

const int segE = 6;

const int segF = 7;

const int segG = 8;

// Define the common cathode pin

const int commonCathode = 9;

// Define a 2D array to represent the segments required to display each digit (0-9)

const byte digitSegments[10][7] = {

{1, 1, 1, 1, 1, 1, 0}, // 0

{0, 1, 1, 0, 0, 0, 0}, // 1

{1, 1, 0, 1, 1, 0, 1}, // 2

{1, 1, 1, 1, 0, 0, 1}, // 3

{0, 1, 1, 0, 0, 1, 1}, // 4

{1, 0, 1, 1, 0, 1, 1}, // 5

{1, 0, 1, 1, 1, 1, 1}, // 6

{1, 1, 1, 0, 0, 0, 0}, // 7

{1, 1, 1, 1, 1, 1, 1}, // 8

{1, 1, 1, 1, 0, 1, 1} // 9

};

void setup() {

// Set segment pins as OUTPUT

pinMode(segA, OUTPUT);

pinMode(segB, OUTPUT);

pinMode(segC, OUTPUT);

pinMode(segD, OUTPUT);

pinMode(segE, OUTPUT);

pinMode(segF, OUTPUT);

pinMode(segG, OUTPUT);

// Set common cathode pin as OUTPUT

pinMode(commonCathode, OUTPUT);

// Initialize common cathode pin to LOW (display off)

digitalWrite(commonCathode, LOW);

// Initialize all segment pins to LOW (segments off)

digitalWrite(segA, LOW);

digitalWrite(segB, LOW);

digitalWrite(segC, LOW);

digitalWrite(segD, LOW);

digitalWrite(segE, LOW);

digitalWrite(segF, LOW);

digitalWrite(segG, LOW);

}

void displayDigit(int digit) {

// Display the specified digit on the seven-segment display

for (int segment = 0; segment < 7; segment++) {

digitalWrite(segment + segA, digitSegments[digit][segment]);

}

}

void loop() {

for (int i = 0; i <= 9; i++) {

// Display numbers 0 to 9

displayDigit(i);

// Turn on the common cathode (display on)

digitalWrite(commonCathode, HIGH);

// Display for a short time (adjust as needed)

delay(1000); // Display each digit for 1 second

// Turn off the common cathode (display off)

digitalWrite(commonCathode, LOW);

// Clear the display (all segments off)

for (int segment = 0; segment < 7; segment++) {

digitalWrite(segment + segA, LOW);

}

// Delay before displaying the next digit (adjust as needed)

delay(500); // Pause between digits for 0.5 seconds

}

}

17. #include <Wire.h>

#include <LiquidCrystal\_I2C.h>

// Initialize the LCD with I2C address 0x27

LiquidCrystal\_I2C lcd(0x27, 16, 2); // Change these values if you have a different LCD configuration

void setup() {

// Initialize the LCD

lcd.init();

lcd.backlight();

// Display your name on the LCD

lcd.setCursor(0, 0); // Set the cursor to the first row, first column

lcd.print("Your Name");

// You can also display additional text on the second line if you have a 16x2 LCD

// lcd.setCursor(0, 1); // Set the cursor to the second row, first column

// lcd.print("Additional Text");

}

void loop() {

// Your main program loop (if needed)

}

18. #include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

char name[] = "Your Name "; // Your name with extra spaces for scrolling

int nameLength = 16; // Length of the name including extra spaces

int scrollDelay = 200; // Delay between scrolling steps (milliseconds)

void setup() {

lcd.begin(16, 2);

lcd.setCursor(0, 0);

lcd.print("Your Name:");

lcd.setCursor(0, 1);

lcd.print(" "); // Clear the second line initially

delay(1000);

}

void loop() {

// Scroll left

for (int i = 0; i < nameLength - 16; i++) {

lcd.setCursor(0, 1);

for (int j = 0; j < 16; j++) {

lcd.print(name[i + j]);

}

delay(scrollDelay);

lcd.clear();

}

// Scroll right

for (int i = nameLength - 16; i >= 0; i--) {

lcd.setCursor(0, 1);

for (int j = 0; j < 16; j++) {

lcd.print(name[i + j]);

}

delay(scrollDelay);

lcd.clear();

}

}

19. #include <Wire.h>

#include <LiquidCrystal\_I2C.h>

// Ultrasonic sensor pins

const int trigPin = 9;

const int echoPin = 10;

// LCD module setup

LiquidCrystal\_I2C lcd(0x27, 16, 2); // Change these values if your LCD setup is different

// Variables for ultrasonic sensor

long duration;

int distance;

// Function to calculate if a number is a palindrome

bool isPalindrome(int num) {

int original = num;

int reversed = 0;

while (num > 0) {

int digit = num % 10;

reversed = reversed \* 10 + digit;

num /= 10;

}

return original == reversed;

}

void setup() {

// Initialize serial communication

Serial.begin(9600);

// Initialize ultrasonic sensor pins

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

// Initialize LCD

lcd.init();

lcd.backlight();

lcd.setCursor(0, 0);

lcd.print("Reading...");

}

void loop() {

int readings[3]; // Array to store three readings

int sum = 0;

// Take three readings at 2-second intervals

for (int i = 0; i < 3; i++) {

// Trigger a pulse to the ultrasonic sensor

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Measure the duration of the echo pulse

duration = pulseIn(echoPin, HIGH);

// Calculate the distance in centimeters (cm)

distance = duration \* 0.034 / 2;

// Store the reading in the array

readings[i] = distance;

// Print the reading to the serial monitor

Serial.print("Reading ");

Serial.print(i + 1);

Serial.print(": ");

Serial.print(readings[i]);

Serial.println(" cm");

delay(2000); // Wait for 2 seconds before the next reading

}

// Calculate the average of the three readings

for (int i = 0; i < 3; i++) {

sum += readings[i];

}

int average = sum / 3;

// Check if the average is a palindrome and display on LCD

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Average: ");

lcd.print(average);

if (isPalindrome(average)) {

lcd.setCursor(0, 1);

lcd.print("Palindrome: Yes");

} else {

lcd.setCursor(0, 1);

lcd.print("Palindrome: No");

}

// Stop further readings

while (true) {

// Do nothing

}

}

20. #include <ESP8266WiFi.h>

const char\* ssid = "YourSSID"; // Replace with your network SSID

const char\* password = "YourPassword"; // Replace with your network password

void setup() {

Serial.begin(115200);

delay(10);

// Connect to Wi-Fi

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println("Connecting to WiFi...");

}

// Print IP and MAC address

Serial.println("Connected to WiFi");

Serial.print("IP address: ");

Serial.println(WiFi.localIP());

Serial.print("MAC address: ");

Serial.println(WiFi.macAddress());

}

void loop() {

// Your main code here (if any)

}

21. CREATE TABLE led\_status (

id INT AUTO\_INCREMENT PRIMARY KEY,

status TINYINT

);

<?php

// Connect to the database (replace with your database credentials)

$db = new mysqli("localhost", "username", "password", "database\_name");

if ($db->connect\_error) {

die("Connection failed: " . $db->connect\_error);

}

// API endpoint to get LED status

if ($\_SERVER['REQUEST\_METHOD'] === 'GET') {

$result = $db->query("SELECT status FROM led\_status ORDER BY id DESC LIMIT 1");

if ($result && $result->num\_rows > 0) {

$row = $result->fetch\_assoc();

echo json\_encode(['status' => $row['status']]);

} else {

echo json\_encode(['status' => null]);

}

}

// API endpoint to update LED status

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$status = $\_POST['status'];

$stmt = $db->prepare("INSERT INTO led\_status (status) VALUES (?)");

$stmt->bind\_param("i", $status);

if ($stmt->execute()) {

echo json\_encode(['message' => 'Status updated successfully']);

} else {

echo json\_encode(['message' => 'Status update failed']);

}

}

$db->close();

?>

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

const char\* ssid = "YourSSID";

const char\* password = "YourPassword";

const char\* serverUrl = "http://your\_server\_ip/led\_status.php";

const int ledPin = D1; // GPIO pin connected to LED (replace with your pin)

void setup() {

pinMode(ledPin, OUTPUT);

Serial.begin(115200);

// Connect to Wi-Fi

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println("Connecting to WiFi...");

}

Serial.println("Connected to WiFi");

// Fetch LED status from the server

int ledStatus = getLedStatus();

if (ledStatus == 1) {

digitalWrite(ledPin, HIGH); // Turn on the LED

} else {

digitalWrite(ledPin, LOW); // Turn off the LED

}

}

void loop() {

// Your main program loop (if any)

}

int getLedStatus() {

HTTPClient http;

http.begin(serverUrl);

int httpCode = http.GET();

if (httpCode == HTTP\_CODE\_OK) {

String payload = http.getString();

int status = payload.toInt();

Serial.print("LED Status from Server: ");

Serial.println(status);

return status;

} else {

Serial.println("Failed to fetch LED status from server");

return -1;

}

}

22. CREATE TABLE device\_status (

id INT AUTO\_INCREMENT PRIMARY KEY,

led\_status TINYINT,

buzzer\_status TINYINT

);

<?php

// Connect to the database (replace with your database credentials)

$db = new mysqli("localhost", "username", "password", "database\_name");

if ($db->connect\_error) {

die("Connection failed: " . $db->connect\_error);

}

// API endpoint to get LED and buzzer statuses

if ($\_SERVER['REQUEST\_METHOD'] === 'GET') {

$result = $db->query("SELECT led\_status, buzzer\_status FROM device\_status ORDER BY id DESC LIMIT 1");

if ($result && $result->num\_rows > 0) {

$row = $result->fetch\_assoc();

echo json\_encode(['led\_status' => $row['led\_status'], 'buzzer\_status' => $row['buzzer\_status']]);

} else {

echo json\_encode(['led\_status' => null, 'buzzer\_status' => null]);

}

}

// API endpoint to update LED and buzzer statuses

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$led\_status = $\_POST['led\_status'];

$buzzer\_status = $\_POST['buzzer\_status'];

$stmt = $db->prepare("INSERT INTO device\_status (led\_status, buzzer\_status) VALUES (?, ?)");

$stmt->bind\_param("ii", $led\_status, $buzzer\_status);

if ($stmt->execute()) {

echo json\_encode(['message' => 'Status updated successfully']);

} else {

echo json\_encode(['message' => 'Status update failed']);

}

}

$db->close();

?>

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

const char\* ssid = "YourSSID";

const char\* password = "YourPassword";

const char\* serverUrl = "http://your\_server\_ip/device\_status.php";

const int ledPin = D1; // GPIO pin connected to the LED (replace with your pin)

const int buzzerPin = D2; // GPIO pin connected to the buzzer (replace with your pin)

void setup() {

pinMode(ledPin, OUTPUT);

pinMode(buzzerPin, OUTPUT);

Serial.begin(115200);

delay(10);

// Connect to Wi-Fi

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println("Connecting to WiFi...");

}

Serial.println("Connected to WiFi");

// Fetch LED and buzzer statuses from the server

int ledStatus, buzzerStatus;

getStatusFromServer(ledStatus, buzzerStatus);

// Set initial LED and buzzer states based on fetched statuses

digitalWrite(ledPin, ledStatus);

digitalWrite(buzzerPin, buzzerStatus);

}

void loop() {

// Your main program loop (if any)

}

void getStatusFromServer(int& ledStatus, int& buzzerStatus) {

HTTPClient http;

http.begin(serverUrl);

int httpCode = http.GET();

if (httpCode == HTTP\_CODE\_OK) {

String payload = http.getString();

Serial.println("Received status from server: " + payload);

// Parse JSON response

int ledStatusParsed = 0;

int buzzerStatusParsed = 0;

sscanf(payload.c\_str(), "{\"led\_status\":%d,\"buzzer\_status\":%d}", &ledStatusParsed, &buzzerStatusParsed);

ledStatus = ledStatusParsed;

buzzerStatus = buzzerStatusParsed;

} else {

Serial.println("Failed to fetch status from server");

}

}