```
#include <ESP8266WiFi.h>
#include <EEPROM.h>
#include <TM1637Display.h>

// Initial AP settings
String currentAPName = "NEW_AP_NAME"; // Default AP Name
String currentPassword = "987654321"; // Default Password
WiFiServer server(80);
String request = "";
String newPassword = "";
String newAPName = "";

// Define relay pins for ESP-01
const int relay1Pin = 0; // GPIO0 (D3)
const int relay2Pin = 2; // GPIO2 (D4)

// Define EEPROM addresses
const int apNameAddress = 0;
const int passwordAddress = 32;

// Define TM1637 display pins for ESP8266
#define CLK 14  // GPIO 14 (D5 on some ESP8266 boards)
#define DIO 12  // GPIO 12 (D6 on some ESP8266 boards)
TM1637Display display(CLK, DIO);

// Pin assignment for potentiometer
const int potPin = A0; // Analog pin to read from the potentiometer
```

```
const int minPrecision = 0; // Minimum precision value
const int maxPrecision = 320; // Maximum precision value

// Variables for potentiometer
float potValue = 0.0; // Raw analog value from the potentiometer
float precisionValue = 0.0; // Calculated precision value based on potentiometer reading
float lastDummyValue = -1.0; // Store the last dummy value to detect changes
const float threshold = 0.3; // Threshold to determine significant change in reading

// Moving average variables for potentiometer readings
const int numReadings = 100; // Number of readings to average
float readings[numReadings]; // Array to store readings
int readIndex = 0; // Current index for readings
float total = 0; // Total of readings
float average = 0; // Average value

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

  // Initialize relay pins as outputs
  pinMode(relay1Pin, OUTPUT);
  pinMode(relay2Pin, OUTPUT);

  // Start with relays off
  digitalWrite(relay1Pin, HIGH);
  digitalWrite(relay2Pin, HIGH);
```

```
// Initialize EEPROM
EEPROM.begin(512);

// Read AP settings from EEPROM
currentAPName = readStringFromEEPROM(apNameAddress);
currentPassword = readStringFromEEPROM(passwordAddress);

// Use default values if EEPROM does not contain valid settings
if (currentAPName.length() == 0 || currentPassword.length() == 0) {
  currentAPName = "NEW_AP_NAME";
  currentPassword = "987654321";
  writeStringToEEPROM(apNameAddress, currentAPName);
  writeStringToEEPROM(passwordAddress, currentPassword);
}

// Start the Access Point
WiFi.disconnect();
WiFi.softAP(currentAPName.c_str(), currentPassword.c_str());
server.begin();

Serial.println("Access Point Started");
Serial.print("IP Address: ");
Serial.println(WiFi.softAPIP());

// Initialize TM1637 display
display.setBrightness(0x0f); // Set TM1637 brightness
```

```
  // Initialize readings array for potentiometer
  for (int i = 0; i < numReadings; i++) {
    readings[i] = 0; // Initialize all readings to 0
  }
}

void loop() {
  // Handle client requests
  WiFiClient client = server.available();
  if (client) {
    request = client.readStringUntil('\r');
    client.flush();

    if (request.indexOf("RELAY1ON_RELAY2OFF") > 0) {
      digitalWrite(relay1Pin, LOW); // Turn Relay 1 ON
      digitalWrite(relay2Pin, HIGH); // Turn Relay 2 OFF
      client.println("HTTP/1.1 200 OK\r\n");
      client.println("RELAY 1 ON, RELAY 2 OFF");
      Serial.println("Relay 1 ON, Relay 2 OFF");
    } else if (request.indexOf("RELAY1OFF_RELAY2ON") > 0) {
      digitalWrite(relay1Pin, HIGH); // Turn Relay 1 OFF
      digitalWrite(relay2Pin, LOW); // Turn Relay 2 ON
      client.println("HTTP/1.1 200 OK\r\n");
      client.println("RELAY 1 OFF, RELAY 2 ON");
      Serial.println("Relay 1 OFF, Relay 2 ON");
    } else if (request.indexOf("BOTH_RELAYS_OFF") > 0) {
      digitalWrite(relay1Pin, HIGH); // Turn Relay 1 OFF
```

```
    digitalWrite(relay2Pin, HIGH); // Turn Relay 2 OFF
    client.println("HTTP/1.1 200 OK\r\n");
    client.println("Both Relays OFF");
    Serial.println("Both Relays OFF");
  } else if (request.indexOf("CHANGE_AP_SETTINGS") > 0) {
    handleAPChangeRequest(request, client);
  } else if (request.indexOf("GET /") >= 0) {
    sendHTMLForm(client);
  }
}

// Read the analog value from the potentiometer
potValue = analogRead(potPin); // Read the ADC value (0 to 1023)

// Remove the oldest reading
total -= readings[readIndex];
readings[readIndex] = potValue;
total += readings[readIndex];

// Move to the next index
readIndex = (readIndex + 1) % numReadings; // Wrap around if at the end

// Calculate the average
average = total / numReadings;

// Map the average to the precision range
precisionValue = map(average, 0, 1023, minPrecision, maxPrecision);
```

```
  // Use map to calculate dummyValue
  int dummyValue = (precisionValue > 50)
    ? map(precisionValue, 50, maxPrecision, 0, maxPrecision - 50)
    : 0;

  // Update the display only if the dummy value has changed significantly
  if (abs(dummyValue - lastDummyValue) > threshold || lastDummyValue == -1.0) {
    lastDummyValue = dummyValue; // Update lastDummyValue to the current dummy value

    // Display dummy value on TM1637 display
    display.showNumberDec(dummyValue, false); // Display dummyValue as an integer

    // Print to Serial Monitor for debugging
    Serial.print("Precision Value: ");
    Serial.print((int)precisionValue);
    Serial.print(" => Display values: ");
    Serial.println(dummyValue);
  }

  // Small delay to allow for potentiometer rotation
  delay(10);
}

void handleAPChangeRequest(String request, WiFiClient &client) {
  int nameIndex = request.indexOf("newAPName=") + 10;
  int nameEndIndex = request.indexOf("&", nameIndex);
```

```
if (nameEndIndex == -1) nameEndIndex = request.indexOf(" ", nameIndex);
newAPName = request.substring(nameIndex, nameEndIndex);
newAPName.trim();

int passIndex = request.indexOf("newPassword=") + 12;
int passEndIndex = request.indexOf(" ", passIndex);
if (passEndIndex == -1) passEndIndex = request.length();
newPassword = request.substring(passIndex, passEndIndex);
newPassword.trim();

bool updated = false;
if (newAPName.length() > 0) {
  currentAPName = newAPName;
  writeStringToEEPROM(apNameAddress, currentAPName);
  Serial.print("Updated AP Name: ");
  Serial.println(currentAPName);
  updated = true;
}
if (newPassword.length() > 0) {
  currentPassword = newPassword;
  writeStringToEEPROM(passwordAddress, currentPassword);
  Serial.print("Updated Password: ");
  Serial.println(currentPassword);
  updated = true;
}
if (updated) {
  WiFi.softAPdisconnect(true);
```

```
    delay(1000);
    WiFi.softAP(currentAPName.c_str(), currentPassword.c_str());
    server.begin();
    client.println("HTTP/1.1 200 OK\r\n");
    client.println("AP Settings Updated. Please reconnect with the new credentials.");
  } else {
    client.println("HTTP/1.1 400 Bad Request\r\n");
    client.println("No new AP settings provided.");
  }
  client.flush();
}

void sendHTMLForm(WiFiClient &client) {
  String html = "<!DOCTYPE HTML><html><body>";
  html += "<h1>Change Access Point Settings</h1>";
  html += "<form action=\"/CHANGE_AP_SETTINGS\" method=\"GET\">";
  html += "New AP Name: <input type=\"text\" name=\"newAPName\"><br>";
  html += "New Password: <input type=\"text\" name=\"newPassword\"><br>";
  html += "<input type=\"submit\" value=\"Update AP Settings\">";
  html += "</form>";
  html += "<h2>Control Relays</h2>";
  html += "<form action=\"/RELAY1ON_RELAY2OFF\" method=\"GET\"><input type=\"submit\"
value=\"Relay 1 ON, Relay 2 OFF\"></form>";
  html += "<form action=\"/RELAY1OFF_RELAY2ON\" method=\"GET\"><input type=\"submit\"
value=\"Relay 1 OFF, Relay 2 ON\"></form>";
  html += "<form action=\"/BOTH_RELAYS_OFF\" method=\"GET\"><input type=\"submit\"
value=\"Both Relays OFF\"></form>";
  html += "</body></html>";
  client.println("HTTP/1.1 200 OK\r\n");
```

```
  client.println("Content-Type: text/html\r\n\r\n");
  client.println(html);
  client.flush();
}

String readStringFromEEPROM(int startAddress) {
  char data[64];
  int len = 0;
  for (int i = 0; i < 64; i++) {
    data[i] = EEPROM.read(startAddress + i);
    if (data[i] == '\0') break;
    len++;
  }
  return String(data).substring(0, len);
}

void writeStringToEEPROM(int startAddress, String value) {
  int len = value.length();
  for (int i = 0; i < 64; i++) {
    if (i < len) {
      EEPROM.write(startAddress + i, value[i]);
    } else {
      EEPROM.write(startAddress + i, 0);
    }
  }
  EEPROM.commit();
}
```