Arduino code :

#include "MQ135.h"

const int MQ135\_1 = A0;

const int MQ135\_2 = A1;

MQ135 gasSensor1(MQ135\_1);

MQ135 gasSensor2(MQ135\_2);

// Calibrated RZero values for each sensor

const float CALIBRATED\_RZERO\_1 = 15.3; // For MQ135\_1 (A0)

const float CALIBRATED\_RZERO\_2 = 30.27; // For MQ135\_2 (A1)

void setup() {

Serial.begin(9600);

gasSensor1.setRZero(CALIBRATED\_RZERO\_1);

gasSensor2.setRZero(CALIBRATED\_RZERO\_2);

}

void loop() {

float ppm1 = gasSensor1.getPPM();

float ppm2 = gasSensor2.getPPM();

Serial.print(ppm1, 1); Serial.print(","); Serial.println(ppm2, 1);

delay(2000);

}

ESP32 code:

#include <WiFi.h>

#include "DHT.h"

// Pin Definitions

#define DHTPIN 9          // DHT11 connected to IO2

#define DHTTYPE DHT11     // Sensor type

#define RELAY\_PIN 3       // Relay control pin

#define RXD2 44           // UART2 RX pin

#define TXD2 43           // UART2 TX pin

DHT dht(DHTPIN, DHTTYPE);

HardwareSerial mySerial(2); // Use UART2

// WiFi AP credentials

const char \*ssid = "AirPurifier";

const char \*password = "12345678";

WiFiServer server(80);

// Sensor values

float ppm1 = 0, ppm2 = 0, avg\_ppm = 0;

float h = 0, t = 0;

bool fanAuto = true;

bool fanManualState = false;

bool fanState = false;

float baseline\_ppm = 0;

#define BASELINE\_READS 10

void setup() {

  Serial.begin(115200);

  dht.begin();

  pinMode(RELAY\_PIN, OUTPUT);

  digitalWrite(RELAY\_PIN, HIGH); // Fan OFF initially

  // UART2 for Arduino communication

  mySerial.begin(9600, SERIAL\_8N1, RXD2, TXD2);

  // Reset WiFi and set mode explicitly

  WiFi.disconnect();

  WiFi.mode(WIFI\_AP);

  // Configure AP with fixed IP

  IPAddress local\_ip(192, 168, 4, 1);

  IPAddress gateway(192, 168, 4, 1);

  IPAddress subnet(255, 255, 255, 0);

  WiFi.softAPConfig(local\_ip, gateway, subnet);

  WiFi.softAP(ssid, password);

  IPAddress myIP = WiFi.softAPIP();

  Serial.print("AP IP address: ");

  Serial.println(myIP);

  server.begin();

  Serial.println("Server started");

  // Get baseline readings for air quality

  Serial.println("Measuring clean air baseline...");

  calculateBaseline();

}

void calculateBaseline() {

  float total = 0;

  int count = 0;

  unsigned long startTime = millis();

  // Try to get baseline readings with timeout

  while (count < BASELINE\_READS && (millis() - startTime < 30000)) {

    if (mySerial.available()) {

      String line = mySerial.readStringUntil('\n');

      int comma = line.indexOf(',');

      if (comma > 0) {

        float p1 = line.substring(0, comma).toFloat();

        float p2 = line.substring(comma + 1).toFloat();

        total += (p1 + p2) / 2.0;

        count++;

        Serial.print("Baseline reading ");

        Serial.print(count);

        Serial.print(": ");

        Serial.println((p1 + p2) / 2.0);

      }

    }

    delay(100);

  }

  if (count > 0) {

    baseline\_ppm = total / count;

  } else {

    baseline\_ppm = 400; // Default clean air assumption if no readings

  }

  Serial.print("Baseline PPM: ");

  Serial.println(baseline\_ppm);

}

void loop() {

  // Read DHT11

  h = dht.readHumidity();

  t = dht.readTemperature();

  // Process any sensor data from Arduino

  processSerialData();

  // Fan control logic (MODIFIED FOR YOUR REQUIREMENT)

  // Automatic mode: Fan ON if MQ135 #1 (input) > 500 ppm

  bool autoFan = (ppm1 > 1000);

  fanState = fanAuto ? autoFan : fanManualState;

  digitalWrite(RELAY\_PIN, fanState ? LOW : HIGH);

  // Handle web server connections

  WiFiClient client = server.available();

  if (client) {

    Serial.println("New Client");

    String header = "";

    unsigned long currentTime = millis();

    unsigned long previousTime = currentTime;

    const long timeoutTime = 2000; // 2 second timeout

    // Loop while the client's connected

    while (client.connected() && currentTime - previousTime <= timeoutTime) {

      currentTime = millis();

      if (client.available()) {

        char c = client.read();

        header += c;

        // If the HTTP request ends

        if (header.indexOf("\r\n\r\n") >= 0) {

          // Check for specific requests

          if (header.indexOf("GET /fan/on") >= 0) {

            fanAuto = false;

            fanManualState = true;

          } else if (header.indexOf("GET /fan/off") >= 0) {

            fanAuto = false;

            fanManualState = false;

          } else if (header.indexOf("GET /fan/auto") >= 0) {

            fanAuto = true;

          }

          // Send HTTP response

          client.println("HTTP/1.1 200 OK");

          client.println("Content-type:text/html");

          client.println("Connection: close");

          client.println();

          // HTML content with improved styling

          client.println("<!DOCTYPE html><html>");

          client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

          client.println("<meta http-equiv=\"refresh\" content=\"5\">");

          client.println("<title>Air Purifier</title>");

          client.println("<style>");

          client.println("body { font-family: Arial; text-align: center; margin-top: 50px; }");

          client.println(".card { background-color: #f8f9fa; border-radius: 10px; margin: 10px auto; padding: 10px; max-width: 400px; box-shadow: 0 4px 8px rgba(0,0,0,0.1); }");

          client.println("button { background-color: #4CAF50; color: white; padding: 10px 20px; margin: 5px; border: none; border-radius: 5px; cursor: pointer; }");

          client.println("button:hover { background-color: #45a049; }");

          client.println(".off { background-color: #f44336; }");

          client.println(".off:hover { background-color: #d32f2f; }");

          client.println(".auto { background-color: #2196F3; }");

          client.println(".auto:hover { background-color: #0b7dda; }");

          client.println(".on { color: green; } .off-text { color: red; }");

          client.println("</style></head>");

          client.println("<body>");

          client.println("<h2>Air Purifier Dashboard</h2>");

          client.println("<div class=\"card\">");

          client.println("<h3>Temperature & Humidity</h3>");

          client.print("<p><b>Temperature:</b> "); client.print(t); client.println(" &deg;C</p>");

          client.print("<p><b>Humidity:</b> "); client.print(h); client.println(" %</p>");

          client.println("</div>");

          client.println("<div class=\"card\">");

          client.println("<h3>Air Quality (PPM)</h3>");

          client.print("<p><b>MQ135 #1 (Input):</b> "); client.print(ppm1, 1); client.println(" ppm</p>");

          client.print("<p><b>MQ135 #2 (Output):</b> "); client.print(ppm2, 1); client.println(" ppm</p>");

          client.print("<p><b>Average:</b> "); client.print(avg\_ppm, 1);

          // Display air quality status based on input sensor

          if (ppm1 < 700) {

            client.print(" <span style='color:green'>(Good)</span>");

          } else if (ppm1 < 1000) {

            client.print(" <span style='color:orange'>(Moderate)</span>");

          } else {

            client.print(" <span style='color:red'>(Poor)</span>");

          }

          client.println("</p>");

          client.println("</div>");

          client.println("<div class=\"card\">");

          client.println("<h3>Fan Control</h3>");

          client.print("<p><b>Mode:</b> ");

          client.print(fanAuto ? "<span style='color:blue'>AUTOMATIC</span>" : "<span>MANUAL</span>");

          client.println("</p>");

          client.print("<p><b>Status:</b> ");

          client.print(fanState ? "<span class='on'>ON</span>" : "<span class='off-text'>OFF</span>");

          client.println("</p>");

          client.println("<form action='/fan/on'><button type='submit'>Manual ON</button></form>");

          client.println("<form action='/fan/off'><button class='off' type='submit'>Manual OFF</button></form>");

          client.println("<form action='/fan/auto'><button class='auto' type='submit'>Auto Mode</button></form>");

          client.println("</div>");

          client.println("</body></html>");

          break;

        }

      }

    }

    // Clear the header variable

    header = "";

    // Close the connection

    client.stop();

    Serial.println("Client disconnected");

  }

}

void processSerialData() {

  if (mySerial.available()) {

    String line = mySerial.readStringUntil('\n');

    int comma = line.indexOf(',');

    if (comma > 0) {

      ppm1 = line.substring(0, comma).toFloat();

      ppm2 = line.substring(comma + 1).toFloat();

      avg\_ppm = (ppm1 + ppm2) / 2.0;

      Serial.print("PPM Values - MQ135 #1: ");

      Serial.print(ppm1);

      Serial.print(", MQ135 #2: ");

      Serial.print(ppm2);

      Serial.print(", Average: ");

      Serial.println(avg\_ppm);

    }

  }

}

ESP32 code:

#include <WiFi.h>

#include "DHT.h"

// Pin Definitions

#define DHTPIN 9          // DHT11 connected to IO2

#define DHTTYPE DHT11     // Sensor type

#define RELAY\_PIN 3       // Relay control pin

#define RXD2 44           // UART2 RX pin

#define TXD2 43           // UART2 TX pin

DHT dht(DHTPIN, DHTTYPE);

HardwareSerial mySerial(2); // Use UART2

// WiFi AP credentials

const char \*ssid = "AirPurifier";

const char \*password = "12345678";

WiFiServer server(80);

// Sensor values

float ppm1 = 0, ppm2 = 0, avg\_ppm = 0;

float h = 0, t = 0;

bool fanAuto = true;

bool fanManualState = false;

bool fanState = false;

float baseline\_ppm = 0;

#define BASELINE\_READS 10

void setup() {

  Serial.begin(115200);

  dht.begin();

  pinMode(RELAY\_PIN, OUTPUT);

  digitalWrite(RELAY\_PIN, HIGH); // Fan OFF initially

  // UART2 for Arduino communication

  mySerial.begin(9600, SERIAL\_8N1, RXD2, TXD2);

  // Reset WiFi and set mode explicitly

  WiFi.disconnect();

  WiFi.mode(WIFI\_AP);

  // Configure AP with fixed IP

  IPAddress local\_ip(192, 168, 4, 1);

  IPAddress gateway(192, 168, 4, 1);

  IPAddress subnet(255, 255, 255, 0);

  WiFi.softAPConfig(local\_ip, gateway, subnet);

  WiFi.softAP(ssid, password);

  IPAddress myIP = WiFi.softAPIP();

  Serial.print("AP IP address: ");

  Serial.println(myIP);

  server.begin();

  Serial.println("Server started");

  // Get baseline readings for air quality

  Serial.println("Measuring clean air baseline...");

  calculateBaseline();

}

void calculateBaseline() {

  float total = 0;

  int count = 0;

  unsigned long startTime = millis();

  // Try to get baseline readings with timeout

  while (count < BASELINE\_READS && (millis() - startTime < 30000)) {

    if (mySerial.available()) {

      String line = mySerial.readStringUntil('\n');

      int comma = line.indexOf(',');

      if (comma > 0) {

        float p1 = line.substring(0, comma).toFloat();

        float p2 = line.substring(comma + 1).toFloat();

        total += (p1 + p2) / 2.0;

        count++;

        Serial.print("Baseline reading ");

        Serial.print(count);

        Serial.print(": ");

        Serial.println((p1 + p2) / 2.0);

      }

    }

    delay(100);

  }

  if (count > 0) {

    baseline\_ppm = total / count;

  } else {

    baseline\_ppm = 400; // Default clean air assumption if no readings

  }

  Serial.print("Baseline PPM: ");

  Serial.println(baseline\_ppm);

}

void loop() {

  // Read DHT11

  h = dht.readHumidity();

  t = dht.readTemperature();

  // Process any sensor data from Arduino

  processSerialData();

  // Fan control logic

  bool autoFan = (avg\_ppm > baseline\_ppm + 200); // Adjust offset as needed

  fanState = fanAuto ? autoFan : fanManualState;

  digitalWrite(RELAY\_PIN, fanState ? LOW : HIGH);

  // Handle web server connections

  WiFiClient client = server.available();

  if (client) {

    Serial.println("New Client");

    String header = "";

    unsigned long currentTime = millis();

    unsigned long previousTime = currentTime;

    const long timeoutTime = 2000; // 2 second timeout

    // Loop while the client's connected

    while (client.connected() && currentTime - previousTime <= timeoutTime) {

      currentTime = millis();

      if (client.available()) {

        char c = client.read();

        header += c;

        // If the HTTP request ends

        if (header.indexOf("\r\n\r\n") >= 0) {

          // Check for specific requests

          if (header.indexOf("GET /fan/on") >= 0) {

            fanAuto = false;

            fanManualState = true;

          } else if (header.indexOf("GET /fan/off") >= 0) {

            fanAuto = false;

            fanManualState = false;

          } else if (header.indexOf("GET /fan/auto") >= 0) {

            fanAuto = true;

          }

          // Send HTTP response

          client.println("HTTP/1.1 200 OK");

          client.println("Content-type:text/html");

          client.println("Connection: close");

          client.println();

          // HTML content with improved styling

          client.println("<!DOCTYPE html><html>");

          client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

          client.println("<meta http-equiv=\"refresh\" content=\"5\">");

          client.println("<title>Air Purifier</title>");

          client.println("<style>");

          client.println("body { font-family: Arial; text-align: center; margin-top: 50px; }");

          client.println(".card { background-color: #f8f9fa; border-radius: 10px; margin: 10px auto; padding: 10px; max-width: 400px; box-shadow: 0 4px 8px rgba(0,0,0,0.1); }");

          client.println("button { background-color: #4CAF50; color: white; padding: 10px 20px; margin: 5px; border: none; border-radius: 5px; cursor: pointer; }");

          client.println("button:hover { background-color: #45a049; }");

          client.println(".off { background-color: #f44336; }");

          client.println(".off:hover { background-color: #d32f2f; }");

          client.println(".auto { background-color: #2196F3; }");

          client.println(".auto:hover { background-color: #0b7dda; }");

          client.println(".on { color: green; } .off-text { color: red; }");

          client.println("</style></head>");

          client.println("<body>");

          client.println("<h2>Air Purifier Dashboard</h2>");

          client.println("<div class=\"card\">");

          client.println("<h3>Temperature & Humidity</h3>");

          client.print("<p><b>Temperature:</b> "); client.print(t); client.println(" &deg;C</p>");

          client.print("<p><b>Humidity:</b> "); client.print(h); client.println(" %</p>");

          client.println("</div>");

          client.println("<div class=\"card\">");

          client.println("<h3>Air Quality (PPM)</h3>");

          client.print("<p><b>MQ135 #1:</b> "); client.print(ppm1, 1); client.println(" ppm</p>");

          client.print("<p><b>MQ135 #2:</b> "); client.print(ppm2, 1); client.println(" ppm</p>");

          client.print("<p><b>Average:</b> "); client.print(avg\_ppm, 1);

          // Display air quality status

          if (avg\_ppm < 700) {

            client.print(" <span style='color:green'>(Good)</span>");

          } else if (avg\_ppm < 1000) {

            client.print(" <span style='color:orange'>(Moderate)</span>");

          } else {

            client.print(" <span style='color:red'>(Poor)</span>");

          }

          client.println("</p>");

          client.println("</div>");

          client.println("<div class=\"card\">");

          client.println("<h3>Fan Control</h3>");

          client.print("<p><b>Mode:</b> ");

          client.print(fanAuto ? "<span style='color:blue'>AUTOMATIC</span>" : "<span>MANUAL</span>");

          client.println("</p>");

          client.print("<p><b>Status:</b> ");

          client.print(fanState ? "<span class='on'>ON</span>" : "<span class='off-text'>OFF</span>");

          client.println("</p>");

          client.println("<form action='/fan/on'><button type='submit'>Manual ON</button></form>");

          client.println("<form action='/fan/off'><button class='off' type='submit'>Manual OFF</button></form>");

          client.println("<form action='/fan/auto'><button class='auto' type='submit'>Auto Mode</button></form>");

          client.println("</div>");

          client.println("</body></html>");

          break;

        }

      }

    }

    // Clear the header variable

    header = "";

    // Close the connection

    client.stop();

    Serial.println("Client disconnected");

  }

}

void processSerialData() {

  if (mySerial.available()) {

    String line = mySerial.readStringUntil('\n');

    int comma = line.indexOf(',');

    if (comma > 0) {

      ppm1 = line.substring(0, comma).toFloat();

      ppm2 = line.substring(comma + 1).toFloat();

      avg\_ppm = (ppm1 + ppm2) / 2.0;

      Serial.print("PPM Values - MQ135 #1: ");

      Serial.print(ppm1);

      Serial.print(", MQ135 #2: ");

      Serial.print(ppm2);

      Serial.print(", Average: ");

      Serial.println(avg\_ppm);

    }

  }

}