Arduino IDE Code

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

#include <ESP8266WebServer.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#include <DHT.h>

#include <ArduinoJson.h>

#include <NTPClient.h>

#include <WiFiUdp.h>

// Pin Definitions

#define DHTPIN D4

#define RELAY\_PIN D5

#define SOIL\_SENSOR A0

#define OLED\_SDA D2

#define OLED\_SCL D1

// WiFi credentials

const char\* ssid = "OnePlus Nord CE 3 Lite 5G";

const char\* password = "onepluse";

// NTP Settings

WiFiUDP ntpUDP;

NTPClient timeClient(ntpUDP, "pool.ntp.org", 19800, 60000); // UTC+5:30 (19800 seconds)

// Display settings

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1);

// DHT Sensor

DHT dht(DHTPIN, DHT11);

// Soil moisture calibration

const int AirValue = 790;

const int WaterValue = 390;

int soilMoistureValue = 0;

int soilmoisturepercent = 0;

// Web server

ESP8266WebServer server(80);

// Pump control

bool manualOverride = false;

unsigned long manualOverrideTime = 0;

const unsigned long MANUAL\_OVERRIDE\_TIMEOUT = 3600000; // 1 hour timeout

bool pumpCurrentlyOn = false;

unsigned long pumpStartTime = 0;

// History tracking

const int MAX\_HISTORY = 50;

struct PumpEvent {

  String startTime;

  String endTime;

  String duration;

  String mode;

  bool isComplete;

};

PumpEvent pumpEvents[MAX\_HISTORY];

int eventIndex = 0;

void setup() {

  Serial.begin(115200);

  // Initialize OLED

  Wire.begin(OLED\_SDA, OLED\_SCL);

  if(!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) {

    Serial.println("OLED allocation failed");

    while(1);

  }

  display.clearDisplay();

  // Initialize relay

  pinMode(RELAY\_PIN, OUTPUT);

  digitalWrite(RELAY\_PIN, HIGH);

  // Initialize DHT

  dht.begin();

  // Connect to WiFi

  WiFi.begin(ssid, password);

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

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

    delay(500);

    Serial.print(".");

  }

  Serial.println("\nWiFi connected");

  Serial.println("IP address: ");

  Serial.println(WiFi.localIP());

  // Initialize NTP client

  timeClient.begin();

  timeClient.update();

  // Set up server routes

  server.on("/", handleRoot);

  server.on("/api/sensor-data", handleSensorData);

  server.on("/api/pump-history", handlePumpHistory);

  server.on("/api/control-pump", handleControlPump);

  server.on("/api/clear-history", HTTP\_POST, handleClearHistory);

  server.begin();

  Serial.println("HTTP server started");

}

void loop() {

  server.handleClient();

  timeClient.update();

  // Read sensors

  float h = dht.readHumidity();

  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {

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

    return;

  }

  // Read soil moisture

  soilMoistureValue = analogRead(SOIL\_SENSOR);

  soilmoisturepercent = constrain(map(soilMoistureValue, AirValue, WaterValue, 0, 100), 0, 100);

  // Update display

  updateDisplay(h, t);

  // Automatic control if not in manual mode

  controlPumpAutomatically();

  delay(2000);

}

void updateDisplay(float humidity, float temperature) {

  display.clearDisplay();

  display.setTextColor(WHITE);

  display.setCursor(0, 15);

  display.print("Moisture: ");

  display.print(soilmoisturepercent);

  display.println("%");

  display.setCursor(0, 35);

  display.print("Humidity: ");

  display.print(humidity, 0);

  display.println("%");

  display.setCursor(0, 55);

  display.print("Temp: ");

  display.print(temperature, 0);

  display.println("C");

  display.display();

}

void controlPumpAutomatically() {

  if (manualOverride) {

    // Check if manual override timeout has expired

    if (millis() - manualOverrideTime > MANUAL\_OVERRIDE\_TIMEOUT) {

      manualOverride = false;

      logPumpStateChange(digitalRead(RELAY\_PIN), "Auto-Resume");

    }

    return;

  }

  static bool lastPumpState = digitalRead(RELAY\_PIN);

  bool newPumpState = soilmoisturepercent <= 30 ? LOW :

                     soilmoisturepercent >= 50 ? HIGH :

                     lastPumpState;

  if (newPumpState != lastPumpState) {

    digitalWrite(RELAY\_PIN, newPumpState);

    logPumpStateChange(newPumpState, "Auto");

    lastPumpState = newPumpState;

  }

}

void logPumpStateChange(bool pumpState, String mode) {

  if (pumpState == LOW) { // Pump turned ON

    pumpStartTime = millis();

    pumpCurrentlyOn = true;

    pumpEvents[eventIndex].startTime = getCurrentTime();

    pumpEvents[eventIndex].endTime = "";

    pumpEvents[eventIndex].duration = "";

    pumpEvents[eventIndex].mode = mode;

    pumpEvents[eventIndex].isComplete = false;

  } else { // Pump turned OFF

    if (pumpCurrentlyOn) {

      pumpCurrentlyOn = false;

      unsigned long duration = (millis() - pumpStartTime) / 1000; // in seconds

      pumpEvents[eventIndex].endTime = getCurrentTime();

      pumpEvents[eventIndex].duration = formatDuration(duration);

      pumpEvents[eventIndex].isComplete = true;

      eventIndex = (eventIndex + 1) % MAX\_HISTORY;

    }

  }

}

String formatDuration(unsigned long seconds) {

  unsigned long minutes = seconds / 60;

  seconds %= 60;

  unsigned long hours = minutes / 60;

  minutes %= 60;

  return String(hours) + "h " + String(minutes) + "m " + String(seconds) + "s";

}

String getCurrentTime() {

  timeClient.update();

  return timeClient.getFormattedTime();

}

void handleRoot() {

  String html = "<html><head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">";

  html += "<style>body { font-family: Arial; text-align: center; margin: 0 auto; padding: 20px; }";

  html += ".card { border: 1px solid #ddd; border-radius: 5px; padding: 15px; margin: 10px; display: inline-block; }";

  html += "</style></head><body>";

  html += "<h1>Smart Irrigation System</h1>";

  html += "<div class=\"card\"><h2>Sensor Data</h2>";

  html += "<p>Soil Moisture: " + String(soilmoisturepercent) + "%</p>";

  html += "<p>Temperature: " + String(dht.readTemperature(), 1) + "°C</p>";

  html += "<p>Humidity: " + String(dht.readHumidity(), 0) + "%</p>";

  html += "<p>Pump Status: " + String(digitalRead(RELAY\_PIN) ? "OFF" : "ON") + "</p>";

  html += "<p>Mode: " + String(manualOverride ? "Manual" : "Auto") + "</p></div>";

  html += "<div class=\"card\"><h2>Controls</h2>";

  html += "<p><a href=\"/api/control-pump?state=on\">Turn ON</a></p>";

  html += "<p><a href=\"/api/control-pump?state=off\">Turn OFF</a></p>";

  html += "<p><a href=\"/api/control-pump?state=auto\">Auto Mode</a></p></div>";

  html += "</body></html>";

  server.send(200, "text/html", html);

}

void handleSensorData() {

  DynamicJsonDocument doc(200);

  doc["soil\_moisture"] = soilmoisturepercent;

  doc["temperature"] = dht.readTemperature();

  doc["humidity"] = dht.readHumidity();

  doc["pump\_status"] = digitalRead(RELAY\_PIN) ? "OFF" : "ON";

  doc["control\_mode"] = manualOverride ? "manual" : "auto";

  String response;

  serializeJson(doc, response);

  server.send(200, "application/json", response);

}

void handlePumpHistory() {

  DynamicJsonDocument doc(4096);

  JsonArray history = doc.createNestedArray("history");

  for (int i = 0; i < MAX\_HISTORY; i++) {

    int index = (eventIndex + MAX\_HISTORY - i - 1) % MAX\_HISTORY;

    if (pumpEvents[index].isComplete) {

      JsonObject event = history.createNestedObject();

      event["start\_time"] = pumpEvents[index].startTime;

      event["end\_time"] = pumpEvents[index].endTime;

      event["duration"] = pumpEvents[index].duration;

      event["mode"] = pumpEvents[index].mode;

    }

  }

  String response;

  serializeJson(doc, response);

  server.send(200, "application/json", response);

}

void handleClearHistory() {

  // Reset all history entries

  for (int i = 0; i < MAX\_HISTORY; i++) {

    pumpEvents[i].isComplete = false;

  }

  eventIndex = 0;

  DynamicJsonDocument doc(200);

  doc["status"] = "success";

  doc["message"] = "History cleared";

  String response;

  serializeJson(doc, response);

  server.send(200, "application/json", response);

}

void handleControlPump() {

  DynamicJsonDocument doc(200);

  if (server.hasArg("state")) {

    String state = server.arg("state");

    if (state == "on") {

      manualOverride = true;

      manualOverrideTime = millis();

      digitalWrite(RELAY\_PIN, LOW);

      logPumpStateChange(LOW, "Manual");

      doc["status"] = "success";

      doc["message"] = "Pump turned ON";

      doc["pump\_state"] = "ON";

    }

    else if (state == "off") {

      manualOverride = true;

      manualOverrideTime = millis();

      digitalWrite(RELAY\_PIN, HIGH);

      logPumpStateChange(HIGH, "Manual");

      doc["status"] = "success";

      doc["message"] = "Pump turned OFF";

      doc["pump\_state"] = "OFF";

    }

    else if (state == "auto") {

      manualOverride = false;

      doc["status"] = "success";

      doc["message"] = "Auto mode enabled";

    }

    else {

      doc["status"] = "error";

      doc["message"] = "Invalid state";

      String response;

      serializeJson(doc, response);

      server.send(400, "application/json", response);

      return;

    }

    String response;

    serializeJson(doc, response);

    server.send(200, "application/json", response);

  } else {

    doc["status"] = "error";

    doc["message"] = "Missing state";

    String response;

    serializeJson(doc, response);

    server.send(400, "application/json", response);

  }

}