Working Title: Plant Monitoring System

Abstract:

Smart Plant Monitoring System helps with taking care of indoor plants as well as outdoor plants. It uses Arduino boards with sensors to measure light, soil moisture, temperature, and humidity around each plant or a general area of soil if using it outdoors. This information is sent to a main Arduino unit and then to an easy-to-use web app. This lets users watch and compare the health of their plants in real time. The goal is to make taking care of plants easier by maintaining the best growing conditions and meeting the needs of plant lovers effectively.

Project Ideas:

Overall Description of Project Idea

The Smart Plant Monitoring System uses an arduino with a different suite of sensors to improve indoor and outdoor plant care. It connects two or more Arduino boards, each linked to a plant, to gather data from different sensors and then send that data to a Central Hub / Server for processing and display the data to the user. This will offer a detailed understanding of the health and requirements of each plant.

Initial Project Design stating how Multiple Arduinos will be used

3 Arduino boards connected to 3 plants along with their respective sensors. These individual Arduino boards are both connected to another arduino which is the central system for receiving and computation of information by the sensors. The central unit will also redirect to the front end of an application which displays the day to the end user.

Expected Plan for Use and Communication between the multiple Arduinos

Communication between the individual Arduino boards and the central unit will be facilitated through WiFi (ESP8266 Serial Wifi Module). This setup allows for the easy addition of more plants to the system and ensures data is consistently and reliably transmitted to the user's interface.

Initial Project Design stating Expected Inputs/Outputs

1. Inputs:
    a. Light sensors x3
    b. Soil Moisture sensors x3
    c. Heat and Humidity sensor (DHT11) x3
    d. ESP 8266 NodeMCU CP2102 ESP-12E x4
    e. Potentiometer x3

2. Outputs:
    a. Real-time data streaming to a web app interface
    b. ESP 8266 NodeMCU CP2102 ESP-12E x4
    c. LCD x3

Our project introduces an approach to plant care by integrating a scalable network of Arduino-based sensors and a central processing unit. This design simplifies the monitoring of multiple plants. Each plant gets its own set of sensors that check things like soil moisture, temperature, humidity, and light. These sensors send information to a main Server powered by ESP8266, which acts like the brain of the system. The main board looks at the data and send it to the web app that will display the data in real-time for the user to see

How to build your project

Step 1 (Connect server board)
- Plug micro usb in to provide power

Step 2 (Connect ESP 8266 NodeMCU on sensor board):
- Place the wifi module near the board or place onto the breadboard
- Connect the GND pin on wifi board to the ground
- Connect the SDA/D1 pin on wifi board to the digital pin SDA on the sensor board
- Connect the 3v3 on the wifi board to the 3.3v supply
- Connect the SCL/D2 pin on the wifi board to the digital pin SCL sensor board.

Step 3 (connecting the humidity and temperature sensor)
- Place the humidity and heat sensor on the breadboard
- connect the ground pin to the ground
- Connect the power in to 5v
- Connect the data pin to digital pin 7

Step 4 (connecting the soil sensor)
- Connect the ground pin to the the ground
- Connect the power pin to 5v
- Connect the Analog pin to A0 on the sensor board

Step 5 (connecting the Photoresistor)
- Place the Photoresistor on the board preferably on a far corner so that the light can reach it without being obstructed
- Place the potentiometer on the board and connect it to the ground and 5v rails
- Add a 220 ohm resistor, connect one end to the ground rail and another on the same rail of the Photoresister the breadboard.

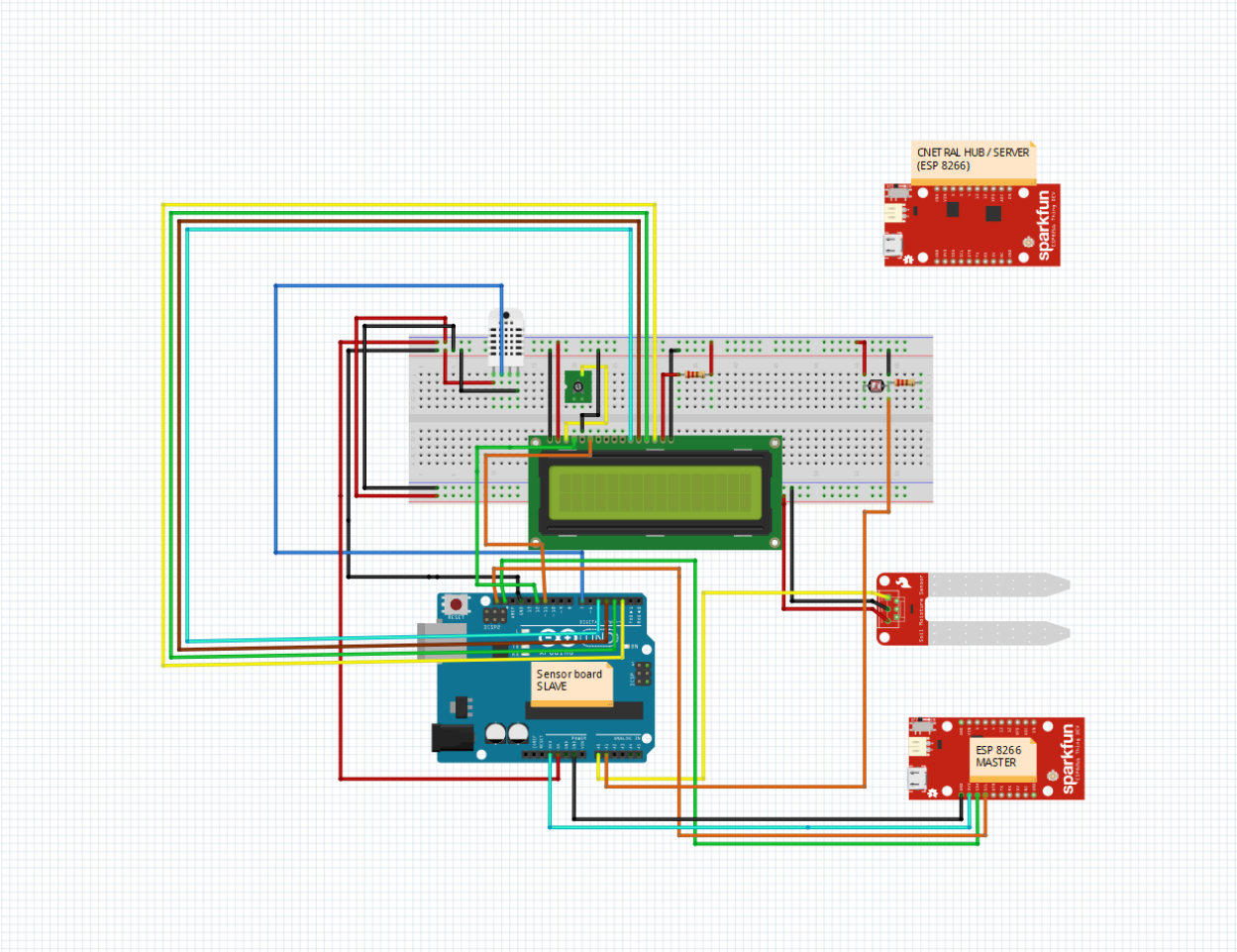
- Add a wire in between the 220 ohm resistor and the photoresistor and connect it to the Analog pin 1(A1) on the arduino board
- Add a wire to the other end of the photoresistor and connect it to the 5v rail

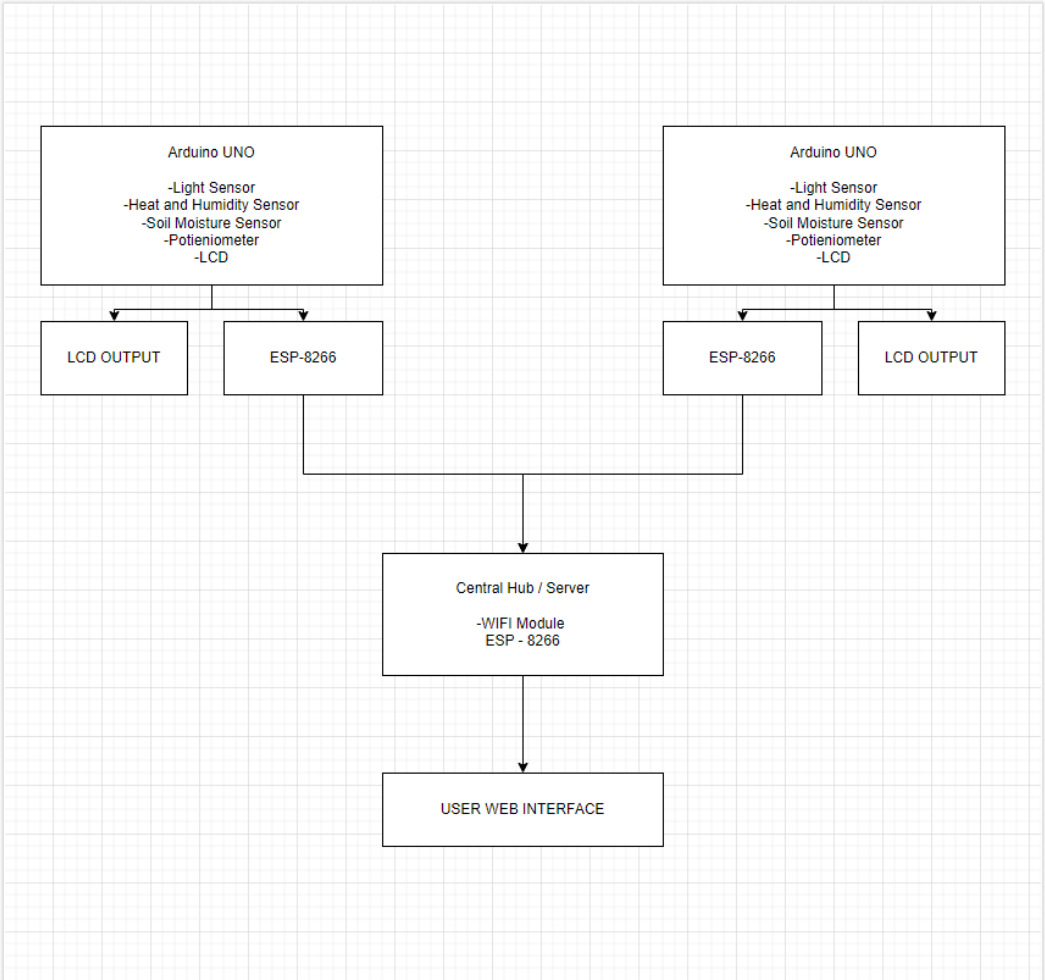
Step 6 (connecting LCD)

- Place the 16x2 LCD on the breadboard, preferably near the corner so you can have more room for other components.
- Place the potentiometer on the board and connect it to the ground and 5v rails
- To wire your LCD screen to your board, connect the following pins:
- LCD RS pin to digital pin 12
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4
- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2
- LCD R/W pin to GND
- LCD VSS pin to GND
- LCD VCC pin to 5V
- LCD LED+ to 5V through a 220 ohm resistor
- LCD LED- to GND

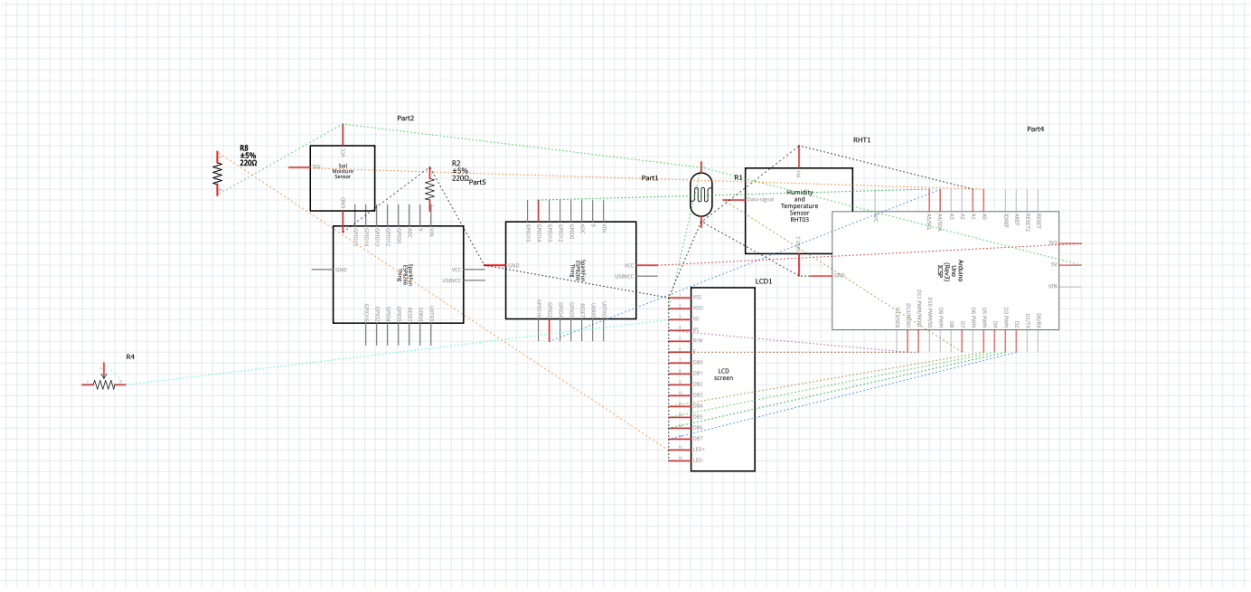
NOTE: Step 2 - 6 can be repeated if you would like to add more boards to the system for additional plants to monitor, nothing more needs to be done, the server will automatically receive the data and display them.

Diagram


CNET RAL HUB / SERVER
(ESP 8266)
sparkfun
Sensor board
SLAVE
ESP 8266
MASTER
sparkfun


Arduino UNO
-Light Sensor
-Heat and Humidity Sensor
-Soil Moisture Sensor
-Potieniometer
-LCD
Arduino UNO
-Light Sensor
-Heat and Humidity Sensor
-Soil Moisture Sensor
-Potieniometer
-LCD
LCD OUTPUT
ESP-8266
ESP-8266
LCD OUTPUT
Central Hub / Server
-WIFI Module
ESP - 8266
USER WEB INTERFACE

Schematics:


Part2
R8
±5%
220Ω
R2
±5%
220Ω
Part5
Part1
R1
RHT1
Part4
LCD1
R4

How to your project is to be used

The entire system will have at least one to many boards, it is completely up to the user to add as many boards he would like, one will be the server board the other will be a sensor suit board. The sensor suit board will include Arduino units equipped with sensors for monitoring soil moisture, light, humidity and temperature. To get started, simply position the sensor units in the soil and near your plants, connect the Arduino units to their power source, and ensure the central unit is within a suitable range for effective communication with your WiFi network. accessing the web application on any internet-enabled device will provide you with the dashboard overview of real-time data from all of your monitored plants. Detailed information on individual plants. Expanding your system to include more plants is effortlessly achieved by adding additional sensor suit units.

Supporting Materials

List of Materials Expected to be Needed:

- Ardiuno board x3
- 220 Ω Resistor x6
- Light sensors x3
- Soil moisture sensors x3
- Heat and humidity sensor x3
- Small house plant x3
- LCD x3
- Potentiometer x3
- ESP8266 Serial Wifi NODEMCU x4
- Wires

List of References:

- https://forum.arduino.cc/t/communication-between-two-boards-using-wifi/968591
- https://www.allaboutcircuits.com/projects/using-an-arduino-as-a-web-server/#:~:text=By%20equipping%20an%20Arduino%20with,Arduino%20as%20a%20web%20server.
- https://docs.arduino.cc/tutorials/uno-wifi-rev2/uno-wifi-r2-hosting-a-webserver/
- https://sensorkit.arduino.cc/sensorkit/module/lessons/lesson/08-the-temperature-sensor
- https://sensorkit.arduino.cc/sensorkit/module/lessons/lesson/05-the-light-sensor
- https://www.instructables.com/TMP36-Temperature-Sensor-Arduino-Tinkercad/
- https://www.circuitbasics.com/arduino-thermistor-temperature-sensor-tutorial/
- https://www.instructables.com/Arduino-Soil-Moisture-Sensor/
- https://docs.arduino.cc/learn/starting-guide/getting-started-arduino/
- https://docs.arduino.cc/tutorials/ethernet-shield-rev2/web-server/
- https://www.allaboutcircuits.com/projects/using-an-arduino-as-a-web-server/
- https://forum.arduino.cc/t/creating-a-front-end-to-arduino/230875
- https://www.instructables.com/Communication-Between-Two-Arduinos-I2C/

- https://forum.arduino.cc/t/how-to-read-sensor-value-analog-read-from-arduino-to-processing-ide/1065816

Code:

Plant Arduino Board's WIFI module (ESP 8266) code:

```
#include <Wire.h> //include the wire library for i2c communication
#include <ESP8266WiFi.h> //include the esp8266 wifi library

const char* ssid = "mastenn"; //ssid of the wifi network
const char* password = "123456789"; //password of the wifi network
const char* host = "192.168.65.148"; //ip address of the server to send data
String sensorData; //string to store the sensor data

void setup() {
  Serial.begin(9600); //initialize serial communication at 9600 baud rate
  Wire.begin(); //start i2c as master
  WiFi.begin(ssid, password); //begin wifi connection

  while (WiFi.status() != WL_CONNECTED) { //wait for wifi connection
    delay(500); //delay 500 ms
    Serial.print("."); //print dots to indicate waiting
  }

  Serial.println("\nWiFi connected"); //notify when wifi is connected
  Serial.print("IP Address: "); //print the local ip address
  Serial.println(WiFi.localIP());
  Serial.println("Master Device Ready"); //indicate the master device is ready
}

void loop() {
  sensorData = ""; //reset sensorData string
  Serial.println("Requesting data from Slave...");
  Wire.requestFrom(8, 128); //request 128 bytes from slave device at address 8

  while (Wire.available()) { //while data is available from slave
    char c = Wire.read(); //read a character from the buffer
    if (c >= 32 && c <= 126) { //filter to readable ascii range
      sensorData += c; //append character if it's printable
    }
  }
```

```
  if (!sensorData.isEmpty()) { //check if any sensor data was received
    Serial.print("Received sensor data: ");
    Serial.println(sensorData); //print the received data

    WiFiClient client; //create a wifi client
    if (client.connect(host, 80)) { //connect to server at given host and port
      Serial.println("Connected to server, sending data...");
      client.println("GET /data?" + sensorData + " HTTP/1.1"); //send GET request with sensor
data
      client.println("Host: " + String(host)); //send the host header
      client.println("Connection: close"); //close the connection after completion
      client.println(); //end of headers

      while (client.connected()) { //while connected to the server
        while (client.available()) { //while there's data from the server
          String line = client.readStringUntil('\n'); //read line by line
          Serial.println(line); //print each line received from the server
        }
      }
      client.stop(); //stop the client
    } else {
      Serial.println("Connection to server failed"); //print if connection failed
    }
  } else {
    Serial.println("No data received from slave"); //print if no data was received
  }

  delay(1000); //wait 5 seconds before next request
}
```

Plant Arduino Board code:

```
#include <Wire.h> //include the wire library for i2c communication
#include <DHT.h> //include the dht library for temperature and humidity sensor
#include <LiquidCrystal.h>

#define DHTPIN 7 //define the pin where the dht sensor is connected
#define DHTTYPE DHT11 //define the type of dht sensor
DHT dht(DHTPIN, DHTTYPE); //initialize the dht sensor

const int dry = 595; //calibrated value for dry soil
```

```
const int wet = 239; //calibrated value for wet soil
int photoPin = A1; //pin for the light sensor

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

unsigned long lastSensorRead = 0; //stores the last time sensors were read
const long sensorReadInterval = 3000; //interval to read sensors in milliseconds

int humidity = 0, temperature = 0; //variables to store humidity and temperature values
int soilMoisture = 0, lightLevel = 0; //variables to store soil moisture and light level values

void setup() {
  Serial.begin(9600); //begin serial communication at 9600 baud rate
  dht.begin(); //start the dht sensor
  pinMode(photoPin, INPUT); //set photopin as input

  Wire.begin(8); //start i2c as slave on address 8
  Wire.onRequest(requestEvent); //register the event for i2c request
  Serial.println("Slave Device Ready"); //print device ready message

  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  unsigned long currentMillis = millis(); //get current time
  if (currentMillis - lastSensorRead >= sensorReadInterval) { //check if it's time to read sensors
    lastSensorRead = currentMillis; //update last sensor read time

    humidity = dht.readHumidity(); //read humidity
    temperature = dht.readTemperature(); //read temperature
    soilMoisture = map(analogRead(A0), wet, dry, 100, 0); //read and scale soil moisture
    lightLevel = map(analogRead(photoPin), 300, 0, 100, 0); //read and scale light level

    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.println("%, ");

    Serial.print("temperature: ");
    Serial.print(temperature);
    Serial.println("°C, ");
```

```
    Serial.print("Soil Moisture: ");
    Serial.print(soilMoisture);
    Serial.println("%, ");

    Serial.print("Light Level: ");
    Serial.print(lightLevel);
    Serial.println("%");

    Serial.println(" ");

    lcd.clear();
    lcd.setCursor(0,0);

    //prepare the humidity and temperature strings
    String humidityText = "H:" + String(humidity) + "%";
    String temperatureText = "T:" + String(temperature) + "C";

    // calculate the position to start the temperature text on the right
    int temperaturePosition = 16 - temperatureText.length(); //ensure it aligns right

    //print humidity and temperature on the same line with adjusted positions
    lcd.print(humidityText);
    lcd.setCursor(temperaturePosition, 0);
    lcd.print(temperatureText);

    //second line, prepare soil moisture and light level similarly
    lcd.setCursor(0,1); // Start of the second line
    String soilText = "S:" + String(soilMoisture) + "%";
    String lightText = "L:" + String(lightLevel) + "%";
    int lightPosition = 16 - lightText.length(); // calc right alignment for light

    lcd.print(soilText);
    lcd.setCursor(lightPosition, 1);
    lcd.print(lightText);
  }
}

void requestEvent() { //handle i2c request
    String sensorData; //using string for easier concatenation
    if (isnan(humidity) || isnan(temperature)) { //check if sensor values are not a number
        sensorData = "Temp=?&Hum=?&Soil=" + String(soilMoisture) + "&Light=" +
String(lightLevel); //handle missing temperature and humidity
    } else {
        sensorData = "Temp=" + String((int)temperature) + "&Hum=" + String((int)humidity) +
```

```
                "&Soil=" + String(soilMoisture) + "&Light=" + String(lightLevel); //concatenate valid
sensor values into a string
    }
    Wire.write(sensorData.c_str(), sensorData.length()); //send data over i2c
}
```

On Central Hub/Server Arduino code:

```
#include <ESP8266WiFi.h> //include ESP8266 WiFi library
#include <ESP8266WebServer.h> //include ESP8266 Web Server library

const char* ssid = "mastenn"; //ssid of the wifi
const char* password = "123456789"; //password for wifi access

ESP8266WebServer server(80); //create a web server on port 80

struct ClientInfo {
  String ip; //client ip address
  int id; //client id
  bool isActive; //flag for client's activity
  String data; //data received from client
  unsigned long lastActive; //timestamp of last activity
};

ClientInfo clients[10]; //array to hold up to 10 clients
int clientCount = 0; //current number of registered clients
int nextClientID = 1; //next client id to assign

void setup() {
  Serial.begin(9600); //initialize baud rate for serial communication
  WiFi.begin(ssid, password); //start wifi connection

  while (WiFi.status() != WL_CONNECTED) {
    delay(500); //wait for connection
    Serial.print("."); //print dots on serial as progress indicator
  }

  Serial.println("\nConnected to WiFi"); //indicate wifi connection success
  Serial.print("IP address: "); //display ip address
  Serial.println(WiFi.localIP());
```

```
  server.on("/", handleRoot); //define route handler for root
  server.on("/data", handleData); //define route handler for data
  server.begin(); //start the server
  Serial.println("HTTP server started"); //server start confirmation
}

bool isWebBrowser(const String& userAgent) {
  //common web browser checks
  return userAgent.startsWith("Mozilla") || userAgent.startsWith("Chrome") ||
userAgent.startsWith("Safari");
}

int getClientId(String ip, String userAgent) {
  if (isWebBrowser(userAgent)) {
    Serial.println("Ignoring web browser connection from IP: " + ip); //ignore web browsers
    return -1; //return -1 to indicate no registration
  }

  unsigned long currentTime = millis(); //get current time
  for (int i = 0; i < clientCount; i++) {
    if (clients[i].ip == ip) {
      clients[i].isActive = true; //update client's active status
      clients[i].lastActive = currentTime; //update client's last active timestamp
      return clients[i].id; //return client id
    }
  }

  if (clientCount < 10) {
    clients[clientCount] = {ip, nextClientID, true, "", currentTime}; //register new client
    clientCount++;
    return nextClientID++; //increment for next client
  } else {
    Serial.println("Client array full!"); //inform that client array is full
    return -1;
  }
}

void handleRoot() {
  //create html content for root path
  String html = "<!DOCTYPE html><html><head><title>Sensor Data</title>";
  html += "<meta http-equiv=\"refresh\" content=\"1\">";
  html += "</head><body>";
  html += "<h1>Sensor Data:</h1>";
```

```
  for (int i = 0; i < clientCount; i++) {
    if (clients[i].isActive) {
      html += "<div id='client" + String(clients[i].id) + "'><h2>Client " + String(clients[i].id) +
"</h2><p>" + clients[i].data + "</p></div>";
    }
  }

  html += "</body></html>";
  server.send(200, "text/html", html); //send html content as response
}

void handleData() {
  String clientIP = server.client().remoteIP().toString(); //get client IP address
  String userAgent = server.header("User-Agent"); //get user agent

  if (isWebBrowser(userAgent)) {
    server.send(403, "Forbidden", "Access denied for web browsers"); //deny access to web
browsers
    return;
  }

  int clientId = getClientId(clientIP, userAgent); //get client id
  if (clientId == -1) {
    server.send(404, "Not Found", "Client not registered or array is full"); //handle unregistered
client
    return;
  }

  String data = "";
  if (server.args() > 0) {
    for (uint8_t i = 0; i < server.args(); i++) {
      if (i > 0) data += ", ";
      data += server.argName(i) + ": " + server.arg(i); //collect data from args
    }
    clients[clientId - 1].data = data; //store data for client
    Serial.println("Received data from Client ID " + String(clientId) + ": " + data); //log received
data
  }

  server.send(200, "application/json", generateClientDataJSON()); //send data as JSON
}

String generateClientDataJSON() {
  //generate JSON data string
```

```
  String jsonData = "[";
  for (int i = 0; i < clientCount; i++) {
    if (clients[i].isActive) {
      if (i > 0) jsonData += ",";
      jsonData += "{\"id\":" + String(clients[i].id) + ", \"data\":\"" + clients[i].data + "\"}";
    }
  }
  jsonData += "]";
  return jsonData; //return JSON data
}

void loop() {
  server.handleClient(); //handle incoming client requests

  unsigned long currentMillis = millis(); //get current millis
  const unsigned long timeout = 15000; //set timeout value

  for (int i = 0; i < clientCount; i++) {
    if (clients[i].isActive && (currentMillis - clients[i].lastActive > timeout)) {
      clients[i].isActive = false; //mark client as inactive if timed out
      Serial.println("Client " + String(clients[i].id) + " has been marked as inactive due to
inactivity."); //log inactive client
    }
  }
}
```