ASSIGNMENT -5

#include <WiFi.h>

#include <HTTPClient.h>

const char\* ssid = "Redmi";

const char\* password = "12345671";

WiFiServer server(80);

String header;

// Current time

unsigned long currentTime = millis();

// Previous time

unsigned long previousTime = 0;

// Define timeout time in milliseconds (example: 2000ms = 2s)

const long timeoutTime = 2000;

void setup() {

Serial.begin(115200);

delay(4000);

pinMode(2, OUTPUT);

digitalWrite(2, HIGH);

WiFi.begin(ssid, password);

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

delay(1000);

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

}

Serial.println("");

Serial.println("WiFi connected.");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

server.begin();

}

void loop() {

if ((WiFi.status() == WL\_CONNECTED)) { //Check the current connection status

HTTPClient http;

http.begin("http://api.openweathermap.org/data/2.5/weather?q=Hyderabad,IN&appid=f40b01b7e49afa7f9bad6ff0f603ea37"); //Specify the URL

int httpCode = http.GET(); //Make the request

if (httpCode > 0) { //Check for the returning code

String payload = http.getString();

Serial.println(httpCode);

String temp=payload.substring(157,160);

Serial.println(“temperature”);

Serial.println(temp);

Serial.println(“humidity”);

String hum=payload.substring(246,249);

//int hum=main12.get();

Serial.println(hum);

WiFiClient client = server.available(); // Listen for incoming clients

if (client) { // If a new client connects,

currentTime = millis();

previousTime = currentTime;

Serial.println("New Client."); // print a message out in the serial port

String currentLine = ""; // make a String to hold incoming data from the client

while (client.connected() && currentTime - previousTime <= timeoutTime) { // loop while the client's connected

currentTime = millis();

if (client.available()) { // if there's bytes to read from the client,

char c = client.read(); // read a byte, then

Serial.write(c); // print it out the serial monitor

header += c;

if (c == '\n') { // if the byte is a newline character

// if the current line is blank, you got two newline characters in a row.

// that's the end of the client HTTP request, so send a response:

if (currentLine.length() == 0) {

// HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)

// and a content-type so the client knows what's coming, then a blank line:

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

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

client.println("Connection: close");

client.println();

// Display the HTML web page

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

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

client.println("<link rel=\"icon\" href=\"data:,\">");

client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");

client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px;");

client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");

client.println(".button2 {background-color: #555555;}</style></head>");

// Web Page Heading

client.println("<body><h1>Weather Data</h1>");

client.println("<body><p1>City : Hyderabad<br>Temaprature :"+temp+" F<br>Humidity :"+hum+"</p1>");

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

// The HTTP response ends with another blank line

client.println();

// Break out of the while loop

break;

} else { // if you got a newline, then clear currentLine

currentLine = "";

}

} else if (c != '\r') { // if you got anything else but a carriage return character,

currentLine += c; // add it to the end of the currentLine

}

}

}

// Clear the header variable

header = "";

// Close the connection

client.stop();

Serial.println("Client disconnected.");

Serial.println("");

}

}

else {

Serial.println("Error on HTTP request");

}

http.end(); //Free the resources

}

delay(10000);

}

