EN.605.715.81.FA25 Project 5 - WiFi Temp & Humidity

Kelly “Scott” Sims

YouTube Link: https://youtu.be/wS_7oo0JA1U
Github Link:
https://github.com/Mooseburger1/johns_hopkins_masters/tree/main/software_development_for_rt_embedded_systems_EN_605_715_81/project_5/esp8226_weather_station/src

Requirements

Project Key Requirements

1. Using ESP8266 transmit / Receive Temp over WiFi
2. Using ESP8266 transmit / Receive Humidity over WiFi
3. Display temp and humidity values

Requirement Alterations

1. Data will be retrieved via weatherapi.com and its “current weather” endpoint
2. Data will be rendered via OLED screen wired to ESP8266

Hardware & Protocol Considerations

Microcontroller: ESP8266 - platform espressif8266
Communication: WiFi
Hardware: LED Screen

Design

WiFi Connection & Weather API Access

WiFi credentials are instantiated as global constants; SSID and Password. They are used in conjunction with the WiFiClient.h library to provide internet access to the ESP8266 board.

An account is created on weatherapi.com in order to retrieve an API key to make calls to the API endpoints. This too is instantiated as a global constant. The endpoint utilized is

http://api.weatherapi.com/v1/current.json?key=**<api_key>**

Where <api_key> is the generated API key. Extra query parameters are added to pinpoint the specific city in which to retrieve weather data from; “q=<city>” such that the exact called endpoint is:

*http://api.weatherapi.com/v1/current.json?key=**<api_key>**&q=**<city>**&aqi=no*

The “aqi=no” query parameter just states we don’t want the air quality index data.

API Calls

To make http calls to the endpoint, the ESP8266HTTPClient.h library is utilized. This library sends and receives traditional http/1.1 calls. To parse the API response, the ArduinoJson.h library is utilized. The full API response schema can be seen using the API explorer at https://www.weatherapi.com/api-explorer.aspx. Below is an example response from the explorer to illustrate a response schema.

Call

```
http://api.weatherapi.com/v1/current.json?key=[redacted]&q=zurich&aqi=no
```

Response Code

```
200
```

Response Headers


```
{
  "Transfer-Encoding": "chunked",
  "Connection": "keep-alive",
  "Vary": "Accept-Encoding",
  "CDN-PullZone": "93447",
  "CDN-Uid": "8fa3a04a-75d9-4707-8056-b7b33c8ac7fe",
  "CDN-RequestCountryCode": "GB",
  "Age": "0",
  "x-weatherapi-qpm-left": "4999999",
  "x-varnish": "735205181",
  "CDN-ProxyVer": "1.39",
  "CDN-RequestPullSuccess": "True",
  "CDN-RequestPullCode": "200",
  "CDN-CachedAt": "10/19/2025 20:09:45",
  "CDN-EdgeStorageId": "1334",
  "CDN-RequestId": "db1dad7afd7a58c671743f365452bbe4",
  "CDN-Cache": "MISS",
  "CDN-Status": "200",
  "CDN-RequestTime": "0",
  "Cache-Control": "public, max-age=180",
  "Content-Type": "application/json",
  "Date": "Sun, 19 Oct 2025 20:09:45 GMT",
  "Server": "BunnyCDN-DE1-860",
  "Via": "1.1 varnish (Varnish/7.1)"
}
```


Response Body

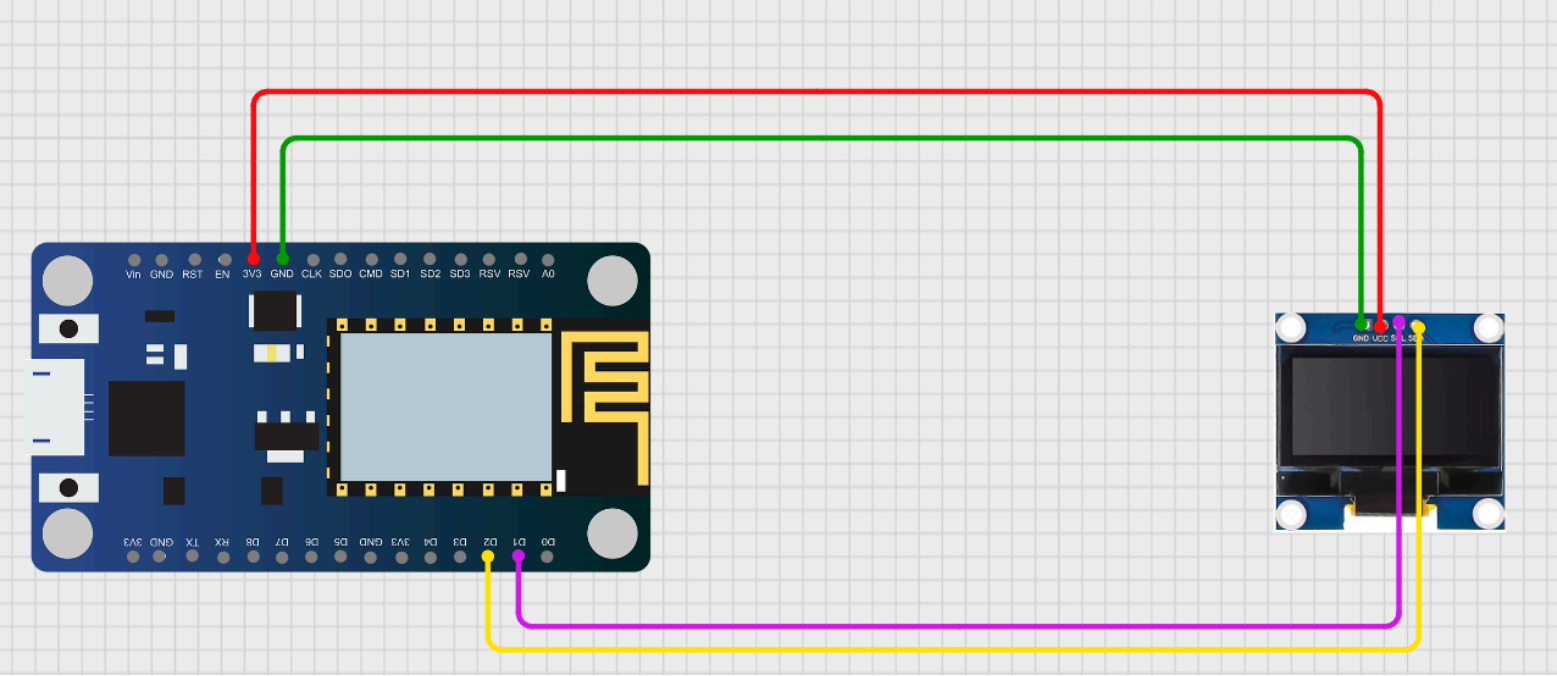

```
{
   "location": {
      "name": "Zurich",
      "region": "",
      "country": "Switzerland",
      "lat": 47.3667,
      "lon": 8.55,
      "tz_id": "Europe/Zurich",
      "localtime_epoch": 1760904827,
      "localtime": "2025-10-19 22:13"
   },
   "current": {
      "last_updated_epoch": 1760904000,
      "last_updated": "2025-10-19 22:00",
      "temp_c": 10.1,
      "temp_f": 50.2,
      "is_day": 0,
      "condition": {
         "text": "Overcast",
         "icon": "//cdn.weatherapi.com/weather/64x64/night/122.png",
         "code": 1009
      },
      "wind_mph": 2.2,
      "wind_kph": 3.6,
      "wind_degree": 148,
      "wind_dir": "SSE",
      "pressure_mb": 1014.0,
      "pressure_in": 29.94,
      "precip_mm": 0.0,
      "precip_in": 0.0,
      "humidity": 87,
      "cloud": 100,
      "feelslike_c": 10.4,
      "feelslike_f": 50.6,
      "windchill_c": 7.9,
      "windchill_f": 46.3,
      "heatindex_c": 7.8,
      "heatindex_f": 46.0,
      "dewpoint_c": 4.6,
      "dewpoint_f": 40.3,
      "vis_km": 10.0,
      "vis_miles": 6.0,
      "uv": 0.0,
      "gust_mph": 4.2,
      "gust_kph": 6.8,
      "short_rad": 0,
      "diff_rad": 0,
      "dni": 0,
      "gti": 0
   }
}
```

Because we only care about temperature and humidity, we only need to extract those values from the response JSON. To do this we use the ArduinoJson.h library, and focus on the temp and humidity fields. We also extract the city, country, and temps in celsius and fahrenheit

```
float temp_f = doc["current"]["temp_f"];
float temp_c = doc["current"]["temp_c"];
int humidity = doc["current"]["humidity"];
String city = doc["location"]["name"];
String country = doc["location"]["country"];
```

Implementation

ESP8266	OLED SCREEN
3V3	VCC
GND	GND
D1 (Clock)	SCL
D2 (Data)	SDA




```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <ArduinoJson.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// Refresh weather data every hour
```

```
// 1000ms x 60s x 60m
const unsigned long interval = 1000 * 60 * 60;

// WIFI Credentials
const char* ssid = "your_internet_here";
const char* password = "your_internet_password_here";

// LED Display Dimensions
const int SCREEN_WIDTH = 128;
const int SCREEN_HEIGHT = 64;
const int OLED_RESET = -1;

unsigned long previousMillis = 0;

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// API Key from Weather API
String api_key = "you_key_here";
String city = "Zurich";

// API Endpoint for Weather API's "current weather" endpoint
String api_endpoint = "http://api.weatherapi.com/v1/current.json?key=" + api_key + "&q=" +
city + "&aqi=no";
String UNITS = "metric"; // Can be "metric" or "imperial"

void getWeatherData();

void setup() {
  Serial.begin(9600);

  // --- Initialize OLED Display ---
  while(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    Serial.println("You are trapped in an infinite loop because the display failed to
initialize");
  }

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0,0);
  display.println("Initializing...");
  display.display();
  delay(1000);


  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi...");
  display.setCursor(0,10);
```

```
  display.print("Connecting to WiFi...");
  display.display();

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    display.print(".");
    display.display();
  }
  Serial.println("\nConnected!");

  display.setCursor(0,20);
  display.println("Connected!");
  display.display();

  // Get initial weather data to kick things off
  getWeatherData();
}

void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    getWeatherData();
  }
}


void updateDisplay(float temp, int humidity, String location, String units = UNITS) {
  display.clearDisplay();

  // Display Location
  display.setTextSize(2);
  display.setCursor(0, 0);
  display.print(location);

  // Display Temperature
  display.setTextSize(2);
  display.setCursor(0, 25);
   // Print with 1 decimal place
  display.print(temp, 1);
  display.print(" ");

  // Add Degree Symbol
  display.setTextSize(1);
  display.drawCircle(display.getCursorX() - 10, 27, 2, SSD1306_WHITE);
  display.setTextSize(2);
  display.print(units == "metric" ? "C" : "F");

  // Display Humidity
```

```
  display.setTextSize(2);
  display.setCursor(0, 48);
  display.print(humidity);
  display.print("% Hum");

  display.display();
}

void getWeatherData() {
  Serial.println("Getting Weather Data");
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("WiFi Disconnected.");
    return;
  }

  WiFiClient client;
  HTTPClient http;

  http.begin(client, api_endpoint);
  int httpResponseCode = http.GET();

  if (httpResponseCode == HTTP_CODE_OK) {
    String payload = http.getString();

    JsonDocument doc;
    DeserializationError error = deserializeJson(doc, payload);

    if (error) {
      Serial.print(F("deserializeJson() failed: "));
      Serial.println(error.c_str());
      return;
    }

    // Extract values
    // See WeatherAPI Explorer for the return JSON format
    // https://www.weatherapi.com/api-explorer.aspx
    float temp_f = doc["current"]["temp_f"];
    float temp_c = doc["current"]["temp_c"];
    int humidity = doc["current"]["humidity"];
    String city = doc["location"]["name"];
    String country = doc["location"]["country"];


    Serial.println("--- Weather Data ---");
    Serial.print(" Temperature: ");
    Serial.println(temp_c);
    Serial.println(temp_f);
    Serial.print(" Humidity: ");
    Serial.print(humidity);
    Serial.println("%");
```

```
    Serial.println("--- Location Data ---");
    Serial.print("City: ");
    Serial.println(city);
    Serial.print("Country: ");
    Serial.println(country);

    updateDisplay(UNITS == "metric" ? temp_c : temp_f, humidity, city);

  } else {
    Serial.printf("HTTP Error code: %d\n", httpResponseCode);
    Serial.println(http.getString());
    display.clearDisplay();
    display.setCursor(0,0);
    display.println("API Request Failed");
    display.println("Check API Key or City");
    display.display();
  }

  http.end();
}
```