Open Weather Map.Org Based Weather Data Monitoring System from Anywhere with Internet Access

Lwin Lwin Yee
Kyaukse University
lwinlwinyee1967@gmail.com

Yin Min Swe
Kyaukse University
yinminswe08@gmal.com

May Soe Myint
Myeik University
teachermaysoe@gmal.com

Abstract

Weather data such as temperature, humidity, wind speed, pressure and etc., are received from Open Weather Map.org using the ESP32. The Open Weather Map.org gives out to the users the application programming interface (API) tool which can be used to request the weather data for the specified location. By signing up for the Open Weather Map.org and creating theown account, the application programming interface identification (APIID) or API keycan be obtained. The city name and country code are filled up the API URL (Uniform Resource Locator) with API key to obtain the weather data of that specified location. The same manner can repeat for the different city and country.Using Open Weather Map.organd ESP32, nomorecomplicated sensors are requiredto obtain the variety of weather data. And then, the cost for weather monitoring system usingOpen Weather Map.org is less than that using actual sensors. The weather data can be obtained at anywhere with internet access from a certain place.

1. Introduction

In the conventional weather data monitoring system, many sensors that can be measured for physical quantities are required and the visualization system such as LCD, TFT LCD, are also needed. Moreover, for the weather data for specific locations, the apparatus will be carried to that location, the manpower is used for monitoring the weather data record. So, the expenditure for measurable facilities, travelling allowance and manpower are used for the conventional weather data monitoring system. In this system, no sensors are required and data can be saved (recorded) in the hard drive of the PC with the conventional manner of the computer. The weather data from the different locations on the earth can be obtained from a place with internet access. For hardware requirement, it needs only one ESP32 which combines the microcontroller and WiFi module. API key of Open Weather Map.org is needed as software requirement.

2. Materials and Method

ESP32 can be operated with an Arduino IDE but it is not included in the Arduino IDE. ESP32 boards must be installed in Arduino IDE.The ESP 32 is functioned as a client and it makes HTTP GET requests to “Open Weather Map.org”. The server responds back with weather information to theclient(ESP32). These data can be monitored on the serial monitor.

2.1. ESP 32 Microprocessor

ESP32 is the microcontroller attached with WiFi / bluetooth / bluetooth low energy(BLE). ESP32 is Tensilica Xtensa 32-bit microprocessor. It is a dual core microcontroller of which speed can run up to 240 MHz and its performance is up to 600 DMIPS (Dhrystone Million Instructions per Second). There are two wireless communications; WiFi and bluetooth. There are two types of bluetooth; blueooth and bluetooth low energy(BLE). It consists of ROM 448KiB, SRAM 510KiB, RTC fast SRAM 8KiB and RTC slow SRAM 8KiB . The I/O pin diagram of ESP32 is illustrated in figure. 1. ESP32 has an input/output I/O bus with multi-functions as follows;.

1) ADC(analog-to-digital converter)
2) DAC(Digital-to- analog converter)
3) I2C (inter-integrated circuit)
4) UART (universal asynchronous receiver/transmitter)
5) CAN2.0 (controller area network)
6) SPI (serial peripheral interface)
7) I2S (inter-integrated sound)
8) RMII (reduced media-independent interface)
9) PWM (pulse width modulation)

Moreover, hall sensor, temperature sensor, touch sensor,digital to analog converter(DAC), analog to digital converter(ADC) and pulse width modulation (PMW) are built in ESP32 [1]. The function block diagram of ESP32 is shown in figure.2.

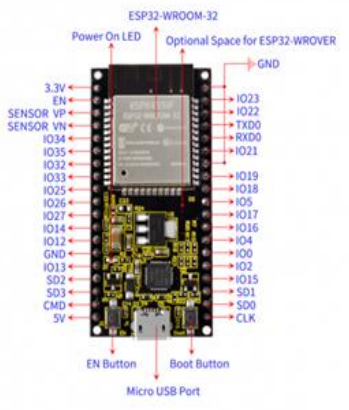

Figure 1. Pin assignment of ESP32microprocessor

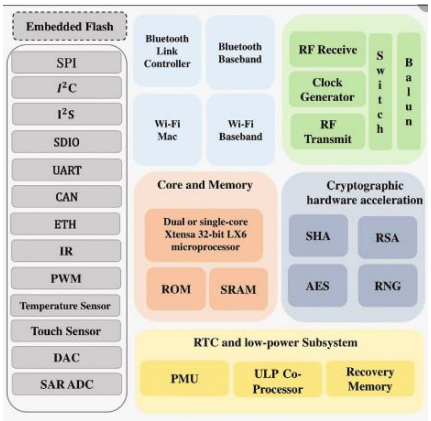

Figure 2. Function block of ESP32

2.2. ESP32 Board Installing

As the ESP32 board and library are not included in ArduinoIDE, they must be installed in Arduino IDE. In the Arduino IDE, File is open and goes to Preferences.Https://dl.espressif.com/dl/package_esp32_index.json is pasted into the "Additional Board Manager URLs" of preference window box as shown in figure 3. And then, open the "tool" in the status bar, then, board manager in the manager window box. The word "ESP32" is written in blank and installed it as shown in figure 4.

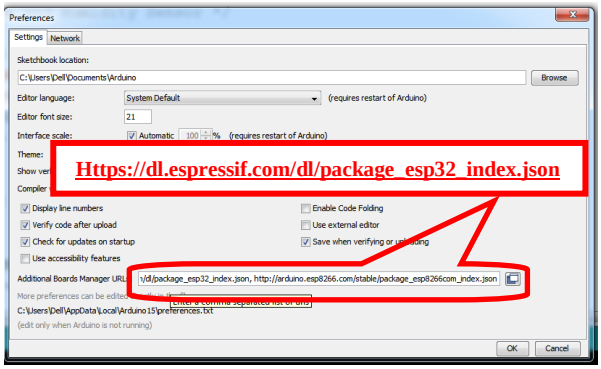

Figure 3. URL filling to install ESP32 board

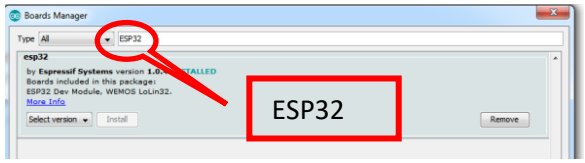

Figure 4. Installing ESP32

3. Operation

In this research, the data string with a json object is received by ESP32 from Open Weather Map.org. That data string contains the information of the weather for optional location. The API key and Arduinojson library are required to get data from Open Weather Map. After they are completed, data string is received from Open Weather Map.org using API key and httpGETRequest() function.

3.1. Arduino JASON Library Installing

After opening IDE, go to "sketch", then to include the library and choose managed libraries. And then, Arduino_JSON library is installed as shown in figure 5.

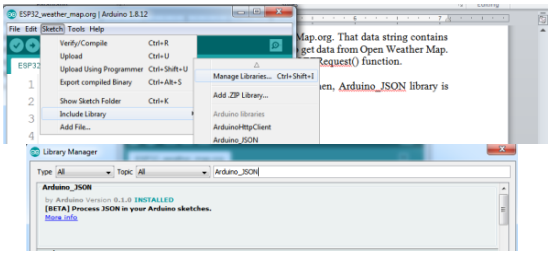

Figure 5. Installing Arduino_JSON library

3.2. Registration API Key of Open Weather Map.org

First of registration process,Visit http://openweathermap.org/appid and then, Sign Up.An account is created by own mail address and then click Create Account.An API can be found on 'API keys' tab in personal account.Step-by-step registration is illustrated in figure 6. Free users can make up to 60 API calls per minute.The current weather data can be accessed for any location on Earth including over 200,000 cities. Current weather is frequently updated based on global models and data from more than 40,000 weather stations. Data is available in JSON, XML, or HTML format. API can be coded by city name, by city ID, by geographic coordinates and by ZIP code.

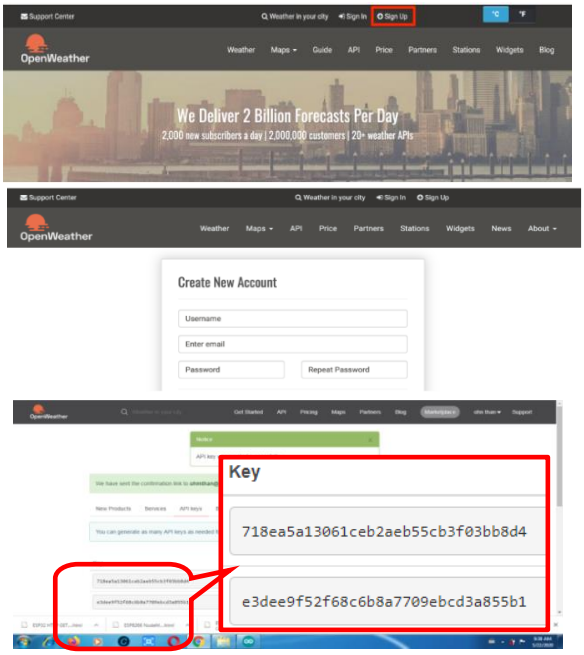

Figure 6. Open weather map registration process

3.3. Getting Weather Data

WiFi.h library is needed to communicate between ESP32 and WiFi network using WiFi network credentials such as network name (SSID) and password [2]. Additionally, HTTPClient.h library is functioned as the HTTP request.Using API key, insert the city and country code as follow URL;

String openWeatherMapApiKey=”xxxxxxxxxxxxxxxxx”;
String city=”Yangon”;
String countryCode=”mm”;
String serverPath=http://api.openweathermap.org/data/2.5/weather?q=+city+”,”+countryCode+”&APPID=”+openWeatherMapApiKey;

Data request to server and response to client are illustrated in figure 7.

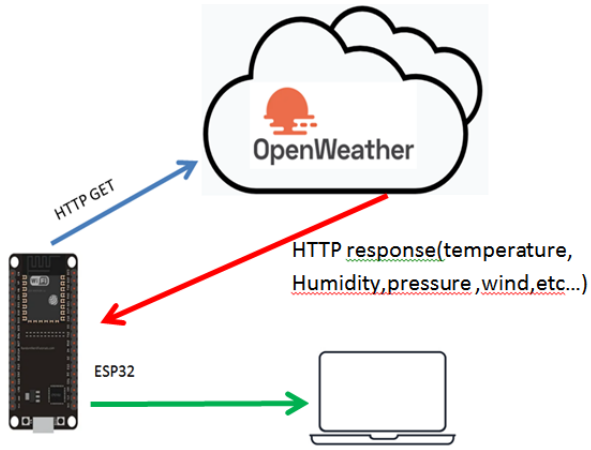

Fig. 7 HTTP get and response

4. Results, Discussion and Conclusion

4.1. Results

The name of city and country code are filled up in the code and the program is run. The Json objects are decoded and stored in the *jsonBuffer* array[3,4]. These values are displayed on the serial monitor. The name of the city, GPS location, current temperature, feeling temperature(heat index), minimum temperature, maximum temperature, visibility(in metre), pressure, humidity, wind speed, wind direction, gust and cloudy(in %) are displayed on the serial monitor. The weather data from the different cities and the different countries are illustrated in figure 8 and figure 9(a) to 9(d) .

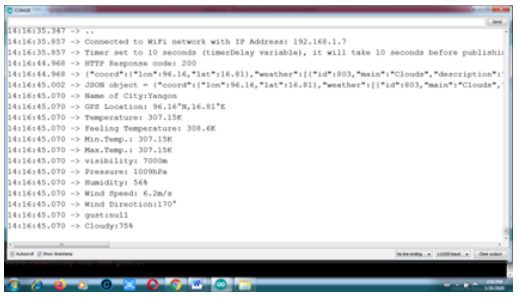

Figure 8. Weather data for Yangon

```
13:58:24.015 -> JSON object = {"coord":{"lon"
13:58:24.083 -> Name of City:Putao
13:58:24.083 -> GPS Location: 97.42°N,27.31°E
13:58:24.083 -> Temperature: 293.73K
13:58:24.083 -> Feeling Temperature: 296.44K
13:58:24.083 -> Min.Temp.: 293.73K
13:58:24.083 -> Max.Temp.: 293.73K
13:58:24.083 -> visibility: nullm
13:58:24.083 -> Pressure: 1011hPa
13:58:24.083 -> Humidity: 90%
13:58:24.083 -> Wind Speed: 0.67m/s
13:58:24.083 -> Wind Direction:317°
13:58:24.083 -> gust:null
13:58:24.083 -> Cloudy:100%
```

```
13:51:33.354 -> JSON object = {"coord":{"lon'
13:51:33.388 -> Name of City:Myitkyina
13:51:33.388 -> GPS Location: 97.4°N,25.38°E
13:51:33.388 -> Temperature: 309.07K
13:51:33.388 -> Feeling Temperature: 310.99K
13:51:33.422 -> Min.Temp.: 309.07K
13:51:33.422 -> Max.Temp.: 309.07K
13:51:33.422 -> visibility: nullm
13:51:33.422 -> Pressure: 1007hPa
13:51:33.422 -> Humidity: 38%
13:51:33.422 -> Wind Speed: 2.1m/s
13:51:33.422 -> Wind Direction:201°
13:51:33.422 -> gust:null
13:51:33.422 -> Cloudy:51%
```

Figure 9(a). Weather data for different cities (Putao and Myitkyina)

```
14:07:44.295 -> JSON object = {"coord":{"lon"
14:07:44.329 -> Name of City:Falam
14:07:44.363 -> GPS Location: 93.68°N,22.92°E
14:07:44.363 -> Temperature: 302.43K
14:07:44.363 -> Feeling Temperature: 299.96K
14:07:44.363 -> Min.Temp.: 302.43K
14:07:44.363 -> Max.Temp.: 302.43K
14:07:44.363 -> visibility: nullm
14:07:44.363 -> Pressure: 1005hPa
14:07:44.363 -> Humidity: 30%
14:07:44.363 -> Wind Speed: 3.55m/s
14:07:44.363 -> Wind Direction:260°
14:07:44.363 -> gust:null
14:07:44.363 -> Cloudy:41%
```

```
14:03:10.478 -> JSON object = {"coord":{"lon"
14:03:10.511 -> Name of City:Tachileik
14:03:10.511 -> GPS Location: 99.88°N,20.45°E
14:03:10.511 -> Temperature: 310.79K
14:03:10.511 -> Feeling Temperature: 312.25K
14:03:10.511 -> Min.Temp.: 310.15K
14:03:10.511 -> Max.Temp.: 311.48K
14:03:10.545 -> visibility: 10000m
14:03:10.545 -> Pressure: 1007hPa
14:03:10.545 -> Humidity: 39%
14:03:10.545 -> Wind Speed: 4.1m/s
14:03:10.545 -> Wind Direction:230°
14:03:10.545 -> gust:null
14:03:10.545 -> Cloudy:20%
```

Figure 9(b). Weather data for different cities (Falam and Tachileik)

```
14:04:59.096 -> JSON object = {"coord":{"lon"
14:04:59.130 -> Name of City:Sittwe
14:04:59.130 -> GPS Location: 92.9°N,20.15°E
14:04:59.130 -> Temperature: 304.18K
14:04:59.130 -> Feeling Temperature: 306.99K
14:04:59.130 -> Min.Temp.: 304.18K
14:04:59.130 -> Max.Temp.: 304.18K
14:04:59.130 -> visibility: nullm
14:04:59.130 -> Pressure: 1008hPa
14:04:59.130 -> Humidity: 66%
14:04:59.130 -> Wind Speed: 4.23m/s
14:04:59.130 -> Wind Direction:227°
14:04:59.130 -> gust:null
14:04:59.130 -> Cloudy:0%
```

```
14:00:53.701 -> JSON object = {"coord":{"lon"
14:00:53.734 -> Name of City:Myeik
14:00:53.734 -> GPS Location: 98.6°N,12.43°E
14:00:53.734 -> Temperature: 301.43K
14:00:53.734 -> Feeling Temperature: 305.01K
14:00:53.734 -> Min.Temp.: 301.43K
14:00:53.734 -> Max.Temp.: 301.43K
14:00:53.734 -> visibility: nullm
14:00:53.734 -> Pressure: 1009hPa
14:00:53.734 -> Humidity: 78%
14:00:53.734 -> Wind Speed: 3.25m/s
14:00:53.734 -> Wind Direction:275°
14:00:53.768 -> gust:null
14:00:53.768 -> Cloudy:100%
```

Figure 9(c). Weather data for different cities(Sittwe and Myeik)

```
14:29:29.504 -> {"coord":{"lon":103.85,"lat":
14:29:29.540 -> JSON object = {"coord":{"lon"
14:29:29.574 -> Name of City:Singapore
14:29:29.574 -> GPS Location: 103.85°N,1.29°E
14:29:29.574 -> Temperature: 303.11K
14:29:29.574 -> Feeling Temperature: 307.59K
14:29:29.608 -> Min.Temp.: 300.15K
14:29:29.608 -> Max.Temp.: 306.15K
14:29:29.608 -> visibility: 8000m
14:29:29.608 -> Pressure: 1009hPa
14:29:29.608 -> Humidity: 74%
14:29:29.608 -> Wind Speed: 2.6m/s
14:29:29.608 -> Wind Direction:10°
14:29:29.608 -> gust:null
14:29:29.608 -> Cloudy:75%
```

```
14:32:44.415 -> {"coord":{"lon":-71.45,"lat":
14:32:44.449 -> JSON object = {"coord":{"lon"
14:32:44.483 -> Name of City:Manchester
14:32:44.483 -> GPS Location: -71.45°N,43°E
14:32:44.483 -> Temperature: 285.33K
14:32:44.483 -> Feeling Temperature: 284.21K
14:32:44.483 -> Min.Temp.: 284.26K
14:32:44.483 -> Max.Temp.: 286.48K
14:32:44.517 -> visibility: 16093m
14:32:44.517 -> Pressure: 1023hPa
14:32:44.517 -> Humidity: 93%
14:32:44.517 -> Wind Speed: 2.1m/s
14:32:44.517 -> Wind Direction:150°
14:32:44.517 -> gust:null
14:32:44.517 -> Cloudy:90%
```

Figure 9(d). Weather data for different cities (Singapore and Manchester)

4.2. Discussion

The data are refreshed every ten seconds. If the name of city and country code is correct, data will be generated. Some parameter such as visibility and gust are not displayed in the some city. In this coding, the weather parameters are measured in SI unit system. There is a problem in program uploading to ESP32. "Failed to connect to ESP32" is occurs when it is connected to IDE properly. BOOT switch is pressed while program is compiling. When program is uploaded, BOOT switch is released [5].

4.3. Conclusion

This system can be applied to monitor the weather conditions of worldwide locations without any sensors and manpower from a place. But it needs the internet access. WiFi hotspot or other network can be used as internet service provider (ISP).So that, this system is useful and cost effective one. It can monitor the current weather data at anywhere. This system can be upgraded using the "thingspeak" cloud server. By connecting the "thingspeak" cloud server, the data with timestamp can be logged and plotted in live graph. The data can be visualized on PC as well as smart phone.

The system in this research used the open weather map.org for free charges. If the paid subscriptions is used, the following advantages will be obtained; (1) quick data update, (2) extended limit of API calls and (3) high data availability. That means that the business-class servers infrastructure is significantly higher reliability, capacity and data precision.

Acknowledgements

I most respectfully express the gratitude to Rector DrAungKhinMyint, Kyaukse University for permitting the opportunity to present this paper.

I am indebted to Professor Dr Cherry Than, Head of Department of Physics, Kyaukse University for giving advice and encouragement through the research period. Also, Thankful to all my colleagues for their participation in this research work.

References

[1] *https://en.wikipedia.org/wiki/ESP32*
[2] *https://en.wikipedia.org/wiki/Web server*
[3] *https://randomnerdtutorials.com/decoding-and-encoding-json- with-arduino-or-esp8266/*
[4] J Purdun, "Beginning C for Ard/doccuino" Apress, NY , 201
[5] R Santos,"Learn ESP32 with Arduino IDE" Random Nerd Tutorial Lab,2020
[6] https://openweathermap.org/appid
[7] https://openweathermap.org/guide
[8] https://www.balbooa.com/gridbox-documentation/openweathermap-api-key