

## IoT…Internet of Things モノのインターネット

《何ができる?》ESP32 につながったセンサーや LED、スイッチなどの入出力が、インターネットと連動する仕組みを作っていく。

インターネット・サイトから情報を入手する

- HTML データそのものをダウンロードして処理
- API を利用して JSON (XML、CSV など) 形式の情報をダウンロードして処理

インターネット・サイトへ情報をエントリーする

- GET クエリ、POST クエリで文字列もしくはキーバリュー (項目=値) ペアを送信
- mBaaS (mobile backend as a Service) や IoT プラットフォームを利用して、JSON 形式の情報を送信

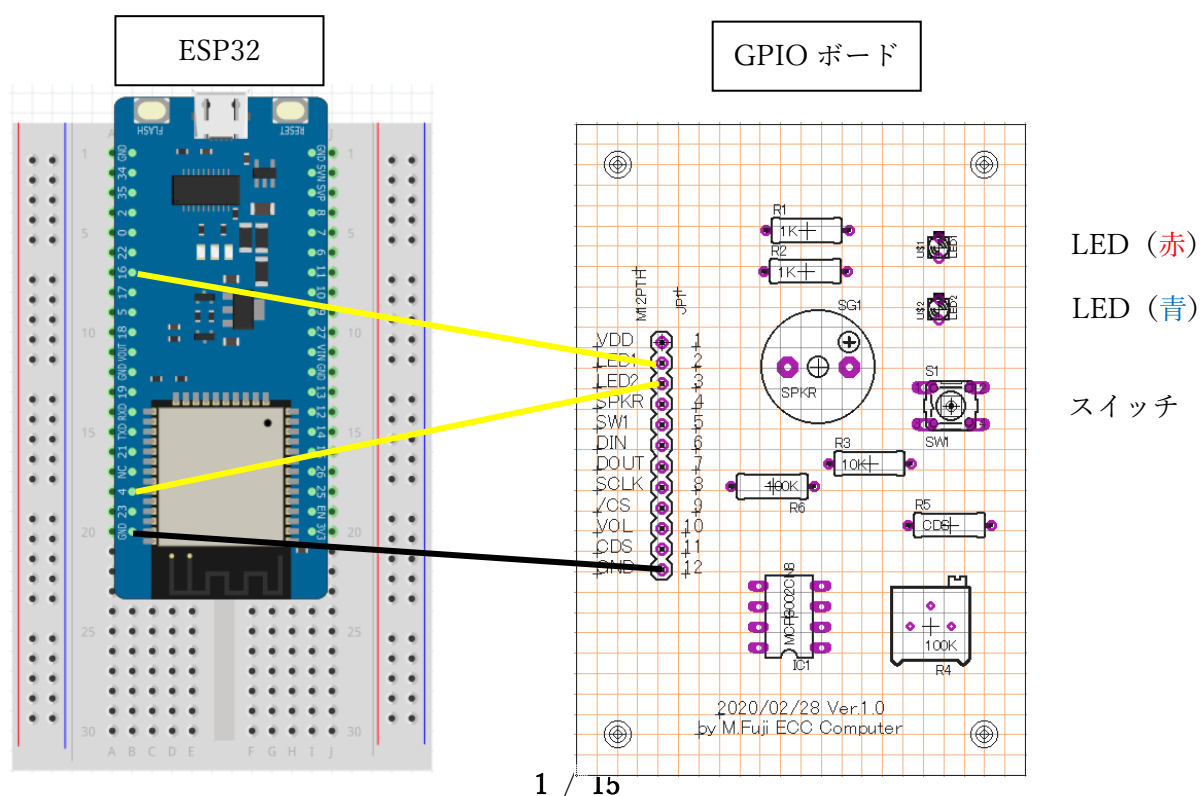
演習

### 【目標】

WiFi ステーション (クライアント) 接続で、インターネット・サイトに接続し、HTML テキストや JSON データを受信する。受信した JSON データの利用方法について理解する。

### 【1, ESP32 と電子工作部品との接続】

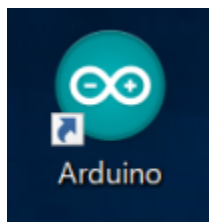
今回新たな電子工作部品はなく、以下の様に LED 2 つを使います。



上記実体配線図を元に、5本のジャンパーコードの配線を行う。接続するピンは以下の一覧の通り。

- ① ジャンパーコード（黄）ESP32のI016 — GPIOボードLED1（2番ピン）
- ② ジャンパーコード（黄）ESP32のI04 — GPIOボードLED2（3番ピン）
- ③ ジャンパーコード（黒）ESP32のGND — GPIOボードGND（12番ピン）

## 【2. Arduino スケッチのサンプルプログラムを実行】

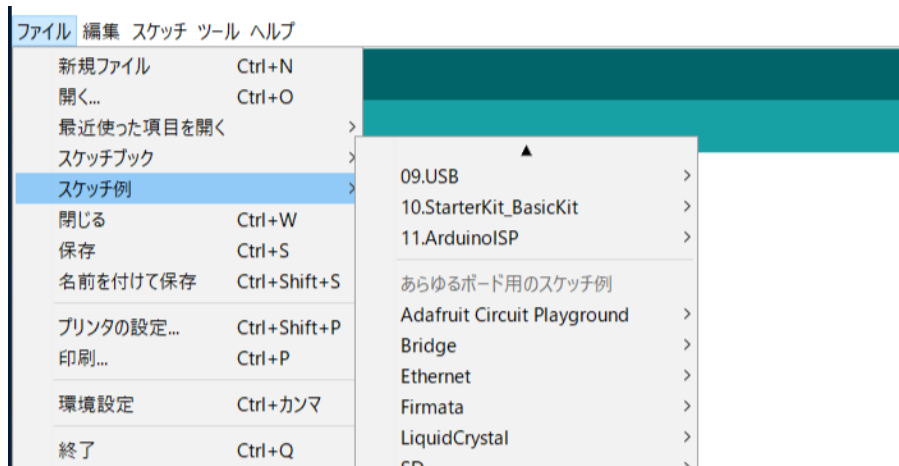


デスクトップのアイコンをダブルクリックして  
Arduino IDE を起動する。

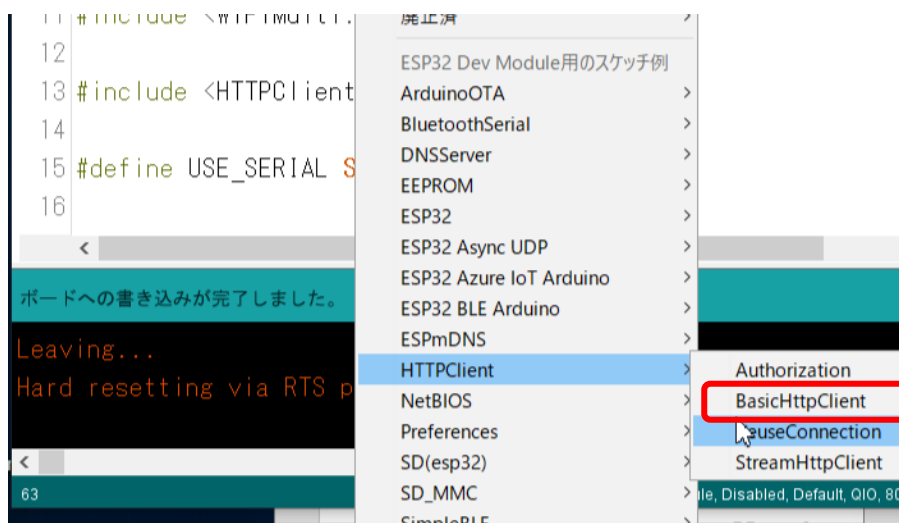
### 【課題】プロジェクト名「kad23\_BasicHttpClient」

まずは、サンプルを呼び出して書き換え、動作確認をします。

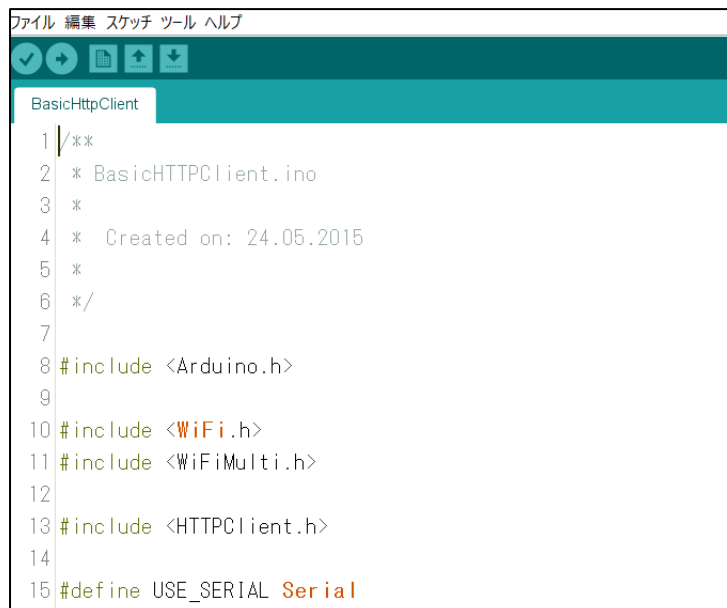
[ファイル] - [スケッチ例] - [HttpClient] - [BasicHttpClient] を選ぶ。



(中略)



以下の様に、サンプルソースの「BasicHttpClient」が表示される。このままでは書き換えても保存できないので、[ファイル]-[名前をつけて保存]で「kad23\_BasicHttpClient」にして保存する。



```
1 /**
2  * BasicHttpClient.ino
3  *
4  * Created on: 24.05.2015
5  *
6  */
7
8  #include <Arduino.h>
9
10 #include <WiFi.h>
11 #include <WiFiMulti.h>
12
13 #include <HttpClient.h>
14
15 #define USE_SERIAL Serial
```

書き換える部分は以下の通り。注意しないといけないのは、サンプルソースの「BasicHttpClient」は、前回の課題と違い、複数の WiFi アクセスポイントを接続できるマルチポイントアクセスの WiFiMulti ライブラリを使っている点である。

WiFiMulti ライブラリはハッシュ化されたパスワードを受け付けないため、代わりに前回の課題 20～22 で使った WiFi ライブラリを用いることにする。

「kad23\_BasicHttpClient.ino」書きかえ前の setup() メソッド

```
50 void setup() {
51
52     USE_SERIAL.begin(115200);
53
54     USE_SERIAL.println();
55     USE_SERIAL.println();
56     USE_SERIAL.println();
57
58     for(uint8_t t = 4; t > 0; t--) {
59         USE_SERIAL.printf("[SETUP] WAIT %d...\n", t);
60         USE_SERIAL.flush();
61         delay(1000);
62     }
63
64     wifiMulti.addAP("SSID", "PASSWORD");
65
66 }
```

54～64 行目をコメントアウトして、以下の様に書き加える。

「kad23 BasicHttpClient.ino」書き換え後の setup() メソッドと ssid/ハッシュ化パスワード変数

```
49 // 接続先のSSIDとパスワード 学内CampusIoT
50 const char ssid[] = "CampusIoT-WiFi";
51 const char passwd[] = "0b8b413f2c0fa6aa90e085e9431abbbf1fa1b2bd2db0ecf4
52
53 void setup() {
54
55     USE_SERIAL.begin(115200);
56
57     // USE_SERIAL.println();
58     // USE_SERIAL.println();
59     // USE_SERIAL.println();
60     //
61     // for(uint8_t t = 4; t > 0; t--) {
62     //     USE_SERIAL.printf("[SETUP] WAIT %d...\n", t);
63     //     USE_SERIAL.flush();
64     //     delay(1000);
65     // }
66     //
67     // wifiMulti.addAP("SSID", "PASSWORD");
68
69     // WiFi接続シーケンス
70     Serial.print("Connecting...");
71     WiFi.begin(ssid, passwd);
72     while (WiFi.status() != WL_CONNECTED) {
73         delay(100);
74         Serial.print(".");
75     }
76     Serial.println("connected");
77     Serial.println(WiFi.localIP());
78 }
```

書き換えたら保存して、マイコンボードに書き込む。

【問題】 もう一か所、loop() メソッド内にある行を書きかえないと正常な動きになりません。WiFiMulti ライブラリのインスタンスを使っている行を書きかえましょう(ヒント:上記書き換えソースの72行目)。

以下の様に HttpClient 接続が行われ、レスポンスコード 200 で HTML 文が表示される。

シリアルモニタ

Connecting.....connected

10.101.0.x

[HTTP] begin...

[HTTP] GET...

[HTTP] GET... code: 200

```
<!doctype html>
```

```
<html>
```

```
<head>
```

```
  <title>Example Domain</title>
```

```
  <meta charset="utf-8" />
```

```
  <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1" />
```

```
  <style type="text/css">
```

```
    body {
```

```
      background-color: #f0f0f2;
```

```
      margin: 0;
```

```
      padding: 0;
```

```
      font-family: "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
```

```
    }
```

```
    div {
```

```
      width: 600px;
```

```
      margin: 5em auto;
```

```
      padding: 50px;
```

```
      background-color: #fff;
```

```
      border-radius: 1em;
```

```
    }
```

```
... 《中略》 ...
```

```
  </style>
```

```
</head>
```

```
<body>
```

```
<div>
```

```
  <h1>Example Domain</h1>
```

```
  <p>This domain is established to be used for illustrative examples in documents. You may use this
```

```
  domain in examples without prior coordination or asking for permission.</p>
```

```
  <p><a href="http://www.iana.org/domains/example">More information...</a></p>
```

```
</div>
```

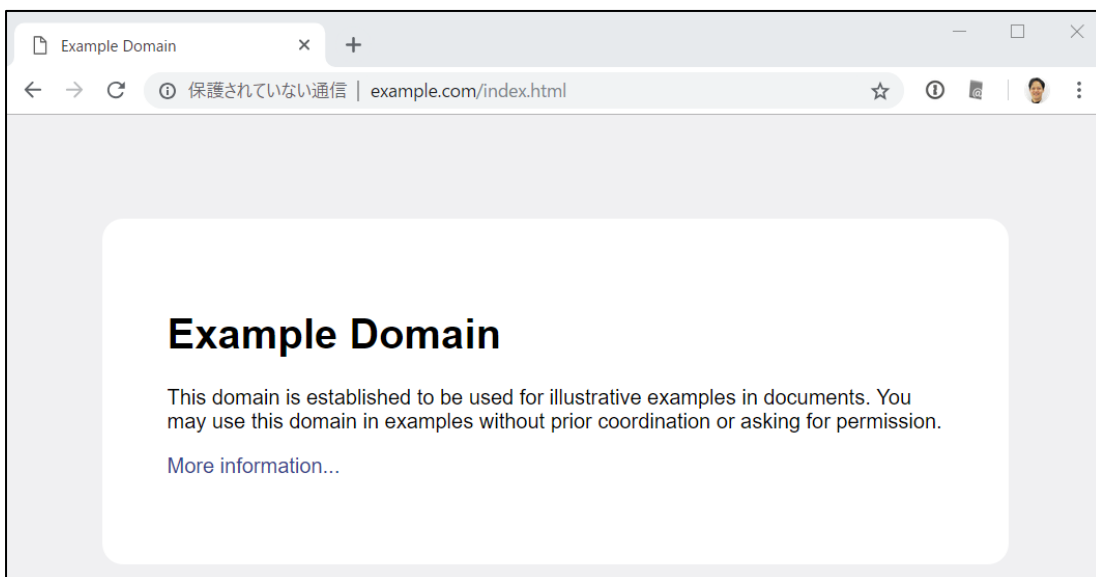
```
</body>
```

```
</html>
```

```
[HTTP] begin...
[HTTP] GET...
[HTTP] GET... code: 200
<!doctype html>
<html>
<head>
  <title>Example Domain</title>
```

↑ 繰り返して HTML テキストが表示される。

上記 HTML テキストは、<http://example.com/index.html> サイトで HTTP 接続ができるサンプルサイト。ブラウザで表示させてみてソースを表示させてみる。シリアルモニタの表示と同じはず。



example.com サイトのトップページが表示されたら、アクセス先のサイトを変更する。

- <http://example.com/index.html> サイトから
- <http://arduinojson.org/example.json> サイトへ変更

「kad23 BasicHttpClient.ino」 の loop() メソッド内を書き換える。

書きかえた後、マイコンボードに書き込んだら、シリアルモニタは以下の様に表示が変わる。

シリアルモニタ 課題 「kad23 BasicHttpClient」 完成時の出力

```
Connecting....connected
10.101.0.x
[HTTP] begin...
[HTTP] GET...
[HTTP] GET... code: 200
{
  "sensor": "gps",
```

```
"time": 1351824120,
"data": [
  48.756080,
  2.302038
]
}
[HTTP] begin...
[HTTP] GET...
[HTTP] GET... code: 200
{
  "sensor": "gps",
  "time": 1351824120,
  "data": [
    48.756080,
    2.302038
  ]
}
```

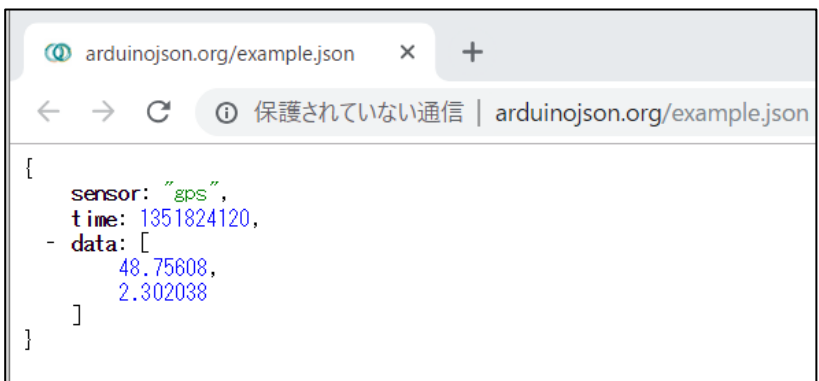
↑繰り返して JSON テキストが表示される。

上記のような出力になれば OK!!

上記 JSON テキストは、<http://arduinojson.org/example.json> サイトで HTTP 接続ができる JSON サンプルのサイトで、ブラウザで表示させてみる。

シリアルモニタの表示と同じはず。

※右のような整形された JSON 表示のため、Chrome 機能拡張の「JSON View」を入れることをおススメする。



#### 【課題】 プロジェクト名「kad24\_BasicHttpClientJson」

前の課題のソースをコピーし JSON テキストをプログラム内で取り扱えるよう書きかえるが、Arduino 言語は JSON をそのまま取り扱えない。JSON を扱うライブラリ「ArduinoJSON」をインクルードしておく必要がある。

以下のサイトを表示させ、「ArduinoJSON」ライブラリを追加しておく。

Arduino で JSON を扱う「ArduinoJSON」 - Bye Bye Moore  
<http://shuzo-kino.hateblo.jp/entry/2016/05/06/203603>



☆Arduino スケッチは、まず前の課題からコピー＆ペーストすればよい。loop() 関数の内容を変更して、受け取った JSON をパースして、配列に格納して単なる文字列から、コード内で使いやすいデータとして利用するやりかたを学ぶ。

前の課題からコピーしたスケッチプロジェクトを開いたまま、サンプルソースを見て、必要部分を書き換える。

[ファイル]-[スケッチ例]-[ArduinoJson]-[JsonParserExample]を選ぶ。

ArduinoJson ライブラリは、JSON データの DOM を扱うライブラリ。文字列をシリアル化して JSON データとして扱い、クラウドに向けて送信したり JSON データをデシリアル化して Arduino スケッチの中で利用できるようにしたりする。





シリアルモニタ 課題「kad24\_BasicHttpClientJson」完成時の出力

```
Connecting....connected
10.101.0.x
[HTTP] begin...
[HTTP] GET...
[HTTP] GET... code: 200
{  ←String 型の payload 変数の中身(ここから)
  "sensor": "gps",
  "time": 1351824120,
  "data": [
    48.756080,
    2.302038
  ]
}  ←String 型の payload 変数の中身(ここまで)
gps                ←JSON キーsensor の中身
1351824120          ←JSON キーtime の中身
48.756081           ←JSON キーdata 配列の 0 番目の要素の中身
2.302038            ←JSON キーdata 配列の 1 番目の要素の中身
[HTTP] begin...
[HTTP] GET...
[HTTP] GET... code: 200
{
  "sensor": "gps",
  "time": 1351824120,
  "data": [
    48.756080,
    2.302038
  ]
}
gps
1351824120
48.756081
2.302038
[HTTP] begin...
↑繰り返して JSON テキストとデシリアライズされた JSON データが表示される。
```

《ヒント》 サンプルスケッチ「JsonParserExample」の 9 行目、`#include <ArduinoJson.h>`を入れることで、JSON のデータを取り扱うクラスが利用可能になる。

```

31 char json[] =
32     "{\"sensor\":\"gps\",\"time\":1351824120,\"data\":[48.756080,2.302038]}";
33
34 // Deserialize the JSON document
35 DeserializationError error = deserializeJson(doc, json);

```

40 行目の `deserializeJson()` メソッドを JSON テキストが入っている `payload` 変数に用いてみよう。`doc` 変数に JSON が入ることになる。

### 【課題】 プロジェクト名「kad25 JsonWeatherOpenMap」

インターネットに接続して、JSON データを Arduino スケッチ内でダウンロードし、プログラムとして利用できるようなれば、API を活用したシステムをつくることができる。

天気 API として良く用いられる Open Weather Map を使って、大阪市の天気の JSON データを入手して表示させてみよう。

以下のサンプルソースを用いて実行してみる。配布するテキストファイルからコピー＆ペーストすればよい。 **☆テキストファイルを渡します**

すでに API キーを入れているのですぐにアクセス可能だが、提出する際には、自身で取得した API キーを使うこと。

```

#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h> //JasonParserExample からコピー
// 接続先の SSID とパスワード 学内 CampusIoT
const char ssid[] = "CampusIoT-WiFi";
const char passwd[] = "0b8b413f2c0fa6aa90e085e9431abbf1fa1b2bd2db0ecf4ae9ce4b2e87da770c";

void setup() {
    Serial.begin(115200);
    // WiFi 接続シーケンス
    Serial.print("Connecting...");
    WiFi.begin(ssid, passwd);
    while (WiFi.status() != WL_CONNECTED) {
        delay(100);
        Serial.print(".");
    }
    Serial.println("connected");
    Serial.println(WiFi.localIP());
}

```

```

void loop() {
    // wait for WiFi connection
    //書き換え
    // if ((wifiMulti.run() == WL_CONNECTED)) {
    if ((WiFi.status() == WL_CONNECTED)) {
        HTTPClient http;
        Serial.print("[HTTP] begin...\n");
        // configure traged server and url

        String api_key = "0242cd04f4f84f903f92820b5e579a73";
        String base_url = "http://api.openweathermap.org/data/2.5/weat
her";
        String url = base_url + "?zip=530-
0015,JP&units=metric&APPID=" + api_key;
        //          String url = base_url + "?q=Osaka,jp&units=metric&AP
PID="+ api_key;

        Serial.println(url);
        http.begin(url); //HTTP
        Serial.print("[HTTP] GET...\n");
        // start connection and send HTTP header
        int httpCode = http.GET();

        // httpCode will be negative on error
        if (httpCode > 0) {
            // HTTP header has been send and Server response header has
            been handled

            Serial.printf("[HTTP] GET... code: %d\n", httpCode);
            // file found at server
            if (httpCode == HTTP_CODE_OK) {
                String payload = http.getString();
                //Serial.println(payload);

                //JasonParserExample からコピー
                StaticJsonDocument<2000> doc;
                // Deserialize the JSON document

```

```

        DeserializationError error = deserializeJson(doc, payload)
;

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

    // Get the root object in the document
    JsonObject root = doc.as<JsonObject>();

    // Fetch values.
    // Most of the time, you can rely on the implicit casts.
    // In other case, you can do root["time"].as<long>();
    const char* cityName = root["name"];
    const char* icon = root["weather"][0]["icon"];
    float temp = root["main"]["temp"];
    int pres = root["main"]["pressure"];
    int hum = root["main"]["humidity"];
    // Print values.
    Serial.printf("都市名:%s 天気アイコン:%s¥n", cityName, icon);
    Serial.printf("温度:%.1f°C 気圧:%dPa 湿度%d%¥n", temp, pres, hum);
    }
    else {
        Serial.printf("[HTTP] GET... failed, error: %s¥n", http.errorToString(httpCode).c_str());
    }
    http.end();
}

delay(5000);
}

```

#### 【実行結果】

シリアルモニタ 課題「[k255 JsonOpwnWeatherMap](#)」完成時の出力

Connecting.....connected

10.101.0.x

[HTTP] begin...

```

http://api.openweathermap.org/data/2.5/weather?zip=530-0015,JP&units=metric&APPID= API キー
[HTTP] GET...
[HTTP] GET... code: 200
都市名:Osaka 天気アイコン:04n
温度:15.0°C 気圧:1013hPa 湿度 82%
[HTTP] begin...
http://api.openweathermap.org/data/2.5/weather?zip=530-0015,JP&units=metric&APPID=API キー
[HTTP] GET...
[HTTP] GET... code: 200
都市名:Osaka 天気アイコン:04n
温度:15.0°C 気圧:1013hPa 湿度 82%
[HTTP] begin...
↑ 繰り返してデシリアライズされた JSON データによる天気データが表示される。

```

上記のサンプルソースは、入手した API キーを使って、以下の様なクエリ文字付きの URL 文字列で Open Weather Map の API サイトにアクセスしている。

```
http://api.openweathermap.org/data/2.5/weather?zip=530-0015,JP&units=metric&APPID=API キー
```

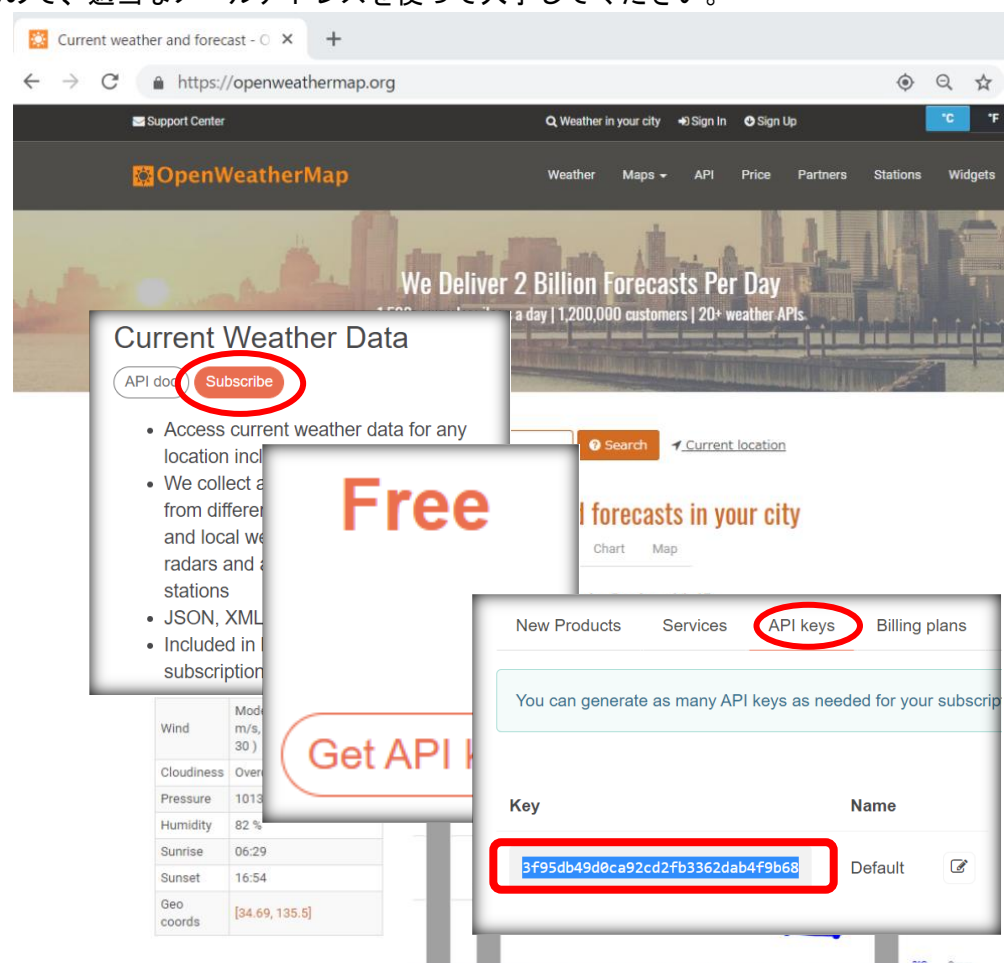
API キーの入手は無料なので、適当なメールアドレスを使って入手してください。

※詳細は Google 検索などで調べましょう。

とりあえず以下のサイトがおすすめ。見てみる。

- [OpenWeatherMap](#)  
で気象情報を  
トしよう | SG  
Labs

API キーを取得出来たら、ブラウザの URL 欄



に以下のクエリ文字付きの URL 文字列で Open Weather Map の API サイトにアクセスしてみましょう。

`http://api.openweathermap.org/data/2.5/weather?zip=530-0015,JP&units=metric&APPID=API キー`

以下の様な、JSON データが表示されます。矢印の部分を Arudino スケッチ内で使っています。



```
{
  - coord: [
    lon: 135.5,
    lat: 34.71
  ],
  - weather: [
    - [
      id: 804,
      main: "Clouds",
      description: "overcast clouds",
      icon: "04n"
    ]
  ],
  base: "stations",
  - main: {
    temp: 15,
    pressure: 1013,
    humidity: 82,
    temp_min: 15,
    temp_max: 15
  },
  visibility: 10000,
  - wind: {
    speed: 6.7,
    deg: 30
  },
  - clouds: {
    all: 90
  },
  dt: 1542049200,
  - sys: {
    type: 1,
    id: 7514,
    message: 0.0196,
    country: "JP",
    sunrise: 1541971751,
    sunset: 1542009285
  },
  id: 740093310,
  name: "Osaka",
  cod: 200
}
```

※上記のような整形された JSON 表示のため、Chrome 機能拡張の「JSON View」を入れることをおすすめします。

**【課題】 プロジェクト名「kad26\_JsonWeatherOpenMapLed」**

前の課題のソースコードをそのままコピーして、天気の状態に応じて、赤色 LED と青色 LED の点灯を制御するプログラムを作れ。仕様としては icon データを利用する。

Day icon	Night icon	Description	LED の点灯
01d.png 	01n.png 	clear sky	赤色 LED のみ点灯 
02d.png 	02n.png 	few clouds	赤 + 青色 LED 点灯  
03d.png 	03n.png 	scattered clouds	青色 LED のみ点灯 
04d.png 	04n.png 	broken clouds	
09d.png 	09n.png 	shower rain	
10d.png 	10n.png 	rain	
以下省略			

icon データ `const char* icon = root["weather"][0]["icon"];` が ”01d” または ”01n” の Clear Sky（快晴）、あるいは 02d” または ”02n” の few clouds（晴れ）のときは赤色 LED のみ点灯、それ以外の時は青色 LED のみ点灯するようにせよ。