TempCity: Using ESP32 Feather board and OpenWeather to read city temperature

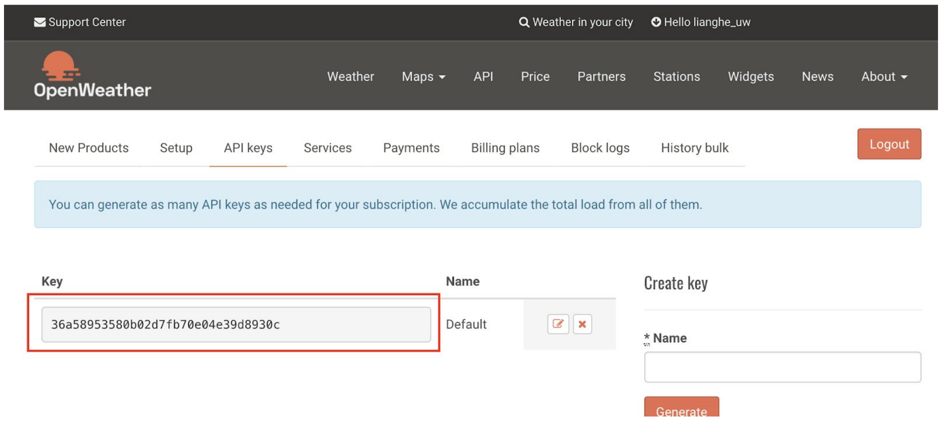
In this tutorial, we show how to use Adafruit HUZZAH32 (ESP32 Feather board) and OpenWeather API to access a specific city's weather information, in this case, temperature. The rest of this tutorial is structured as follows: connect Adafruit HUZZAH32 to Internet, access OpenWeather data using provided API, build a simple physical system to check a specific city's temperature.

Get Adafruit HUZZAH32 Connected to Internet

Check out the tutorial of configuring programming environment and connecting your ESP32 Feather board to the campus network on this GitHub wiki page.

Register an Account on OpenWeather and Obtain Free API Key

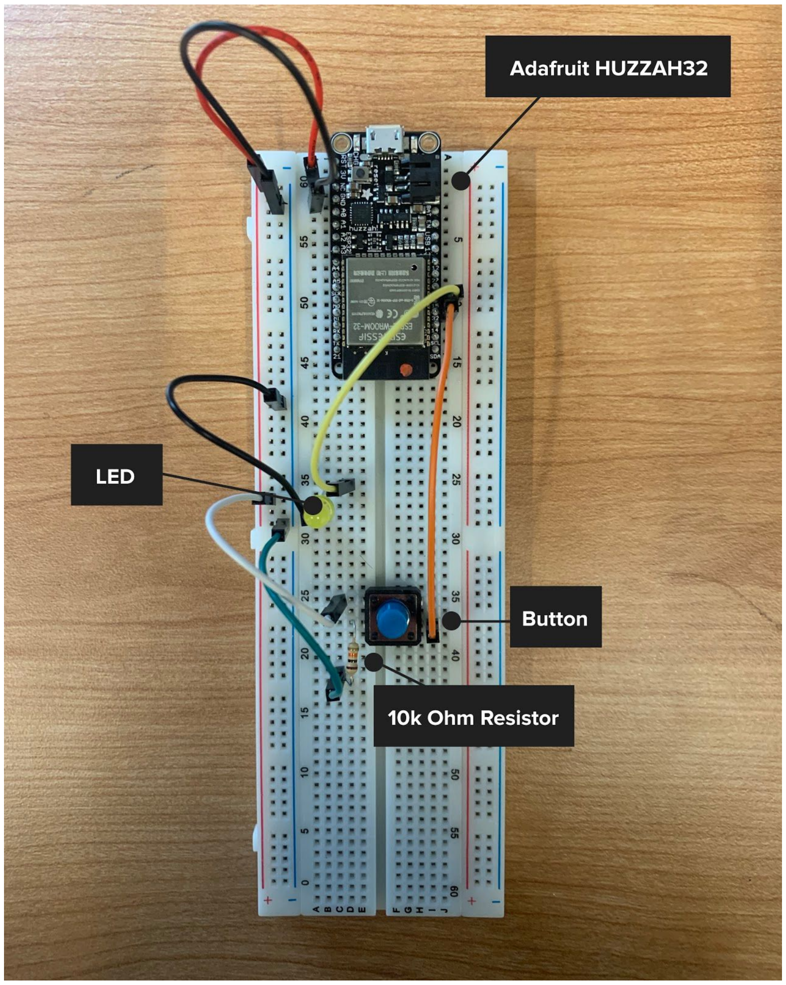
1. Go to OpenWeather website and sign up a free account.
2. Get the free API key: https://home.openweathermap.org/api_keys. Copy and save the key (see the example key in the red rectangle in the screenshot below) in a txt file.



You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.

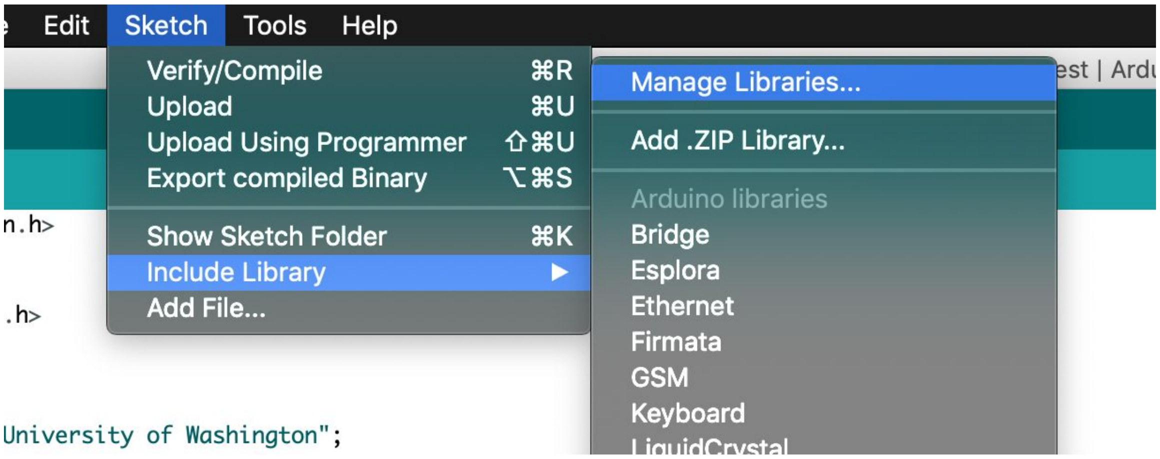
Key	Name
36a58953580b02d7fb70e04e39d8930c	Default

Circuit Building



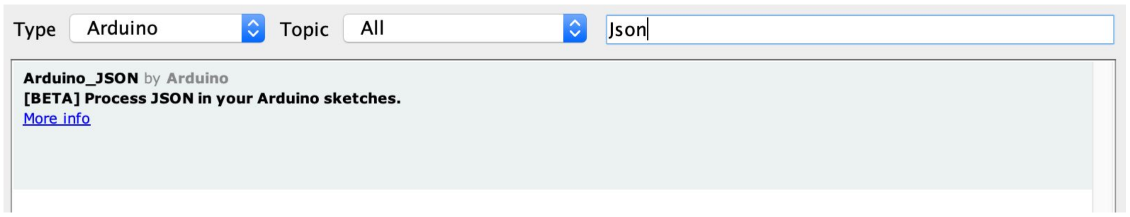
Programming in Arduino IDE

1. Install ArduinoJSON library: Go to “Sketch -> Include Library -> Manage Libraries”:



```
n.h>
.h>
University of Washington";
```

Search for “Json” and install “Arduino_JSON” library:



Arduino_JSON by **Arduino**
[BETA] Process JSON in your Arduino sketches.
More info

2. Include all required libraries in the code:

```
#include <ArduinoJson.h>

#include <WiFi.h>
#include <HTTPClient.h>
```

3. Set up the WiFi connection:

```
// WiFi related
const char* ssid = "University of Washington";
const char* password =  "";
```

4. Set up the LED related initialization:

```
/*** LED related ***/
const int ledPin = 15;  // 15 corresponds to GPIO15
// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8; // 8 bits used for brightness: 0 ~ 255
```

5. Set up the city selection (button control part):

```
/*** City selection related ***/
String cities [7] = {"Beijing,cn","Tokyo,jp","Seattle,us","Pittsburgh,us","Honolulu,us","atlanta,us","alaska,us"};
int cityIdx = 0;
const int buttonPin = 32;
int buttonState = 0;
int lastButtonState = LOW;   // the previous reading from the input pin
unsigned long lastDebounceTime = 0;  // the last time the output pin was toggled
unsigned long debounceDelay = 50;    // the debounce time; increase if the output flickers
```

6. Set up the OpenWeather connection:

```
/***  OpenWeather access related ***/
const String endpoint = "http://api.openweathermap.org/data/2.5/weather?q=";
const String key = "&APPID=36a58953580b02d7fb70e04e39d8930c"; ← replace this with your key
```

7. Inside *step()* function:

```
void setup() {
   Serial.begin(115200);
   delay(10);

   // We start by connecting to a WiFi network
   Serial.println();
   Serial.println();
   Serial.print("Connecting to ");
   Serial.println(ssid);

   WiFi.begin(ssid, password);
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  // the network is successfully connected
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  /*** LED setup ***/
  ledcSetup(ledChannel, freq, resolution);
  ledcAttachPin(ledPin, ledChannel);

  /*** city selection ***/
  pinMode(buttonPin, INPUT);
}
```

8. Inside *loop()* function:
    a. Check the button click with debouncing (more information about debouncing: https://www.arduino.cc/en/tutorial/debounce)

```
/*** check if the city has been changed ***/
  int reading = digitalRead(buttonPin);
  // If the switch changed, due to noise or pressing:
  if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    // whatever the reading is at, it's been there for longer than the debounce
    // delay, so take it as the actual current state:

    // if the button state has changed:
    if (reading != buttonState) {
      buttonState = reading;

      if (buttonState == HIGH) {
        // change a new city
        cityIdx = (cityIdx + 1)%7;
        Serial.println();
        Serial.print("City: ");
        Serial.println(cities[cityIdx]);

            …..

      }
      else {
        Serial.println("Error on HTTP request");
      }

      http.end();
    }
```

```
        }
```

b. Parse the temperature of the selected city and control the brightness of the LED

```
/*** Obtain the temperature of the new city ***/
    HTTPClient http;
    http.begin(endpoint + cities[cityIdx] + key);

    int httpCode = http.GET();
    //Check for the returning code
    if (httpCode > 0) {
      String payload = http.getString();
      // Serial.println(httpCode);
      // Serial.println(payload);

       /*** Change the LED brightness based on the temperature ***/
      DynamicJsonDocument doc(1024);
      DeserializationError error = deserializeJson(doc, payload);
      if (error)
        return;

      int temp = doc["main"]["temp"];
      Serial.print("temp: ");
      Serial.println(temp);

      int brightness = map(temp, 280, 310, 0, 255);
      Serial.print("LED brightness: ");
      Serial.println(brightness);
      Serial.println();

      ledcWrite(ledChannel, brightness);
```

The source code can be found on GitHub:
https://github.com/jonfroehlich/CSE599Au2019/tree/master/Code/ESP32/TempCityApp

Functionality & Interaction

A library of 7 cities is provided. When the user pushes the button, the next city's temperature is displayed through the LED. The brightness represents the temperature of the selected city. The user can see which city is selected in the serial monitor.

Demo