```
/*
 * Rui Santos
 * Complete Project Details http://randomnerdtutorials.com
 * Based on the Arduino Ethernet Web Client Example
 * and on the sketch "Sample Arduino Json Web Client" of the Arduino JSON
library by Benoit Blanchon (bblanchon.github.io/ArduinoJson)
 */
#include <ArduinoJson.h>
#include <Ethernet.h>
#include <SPI.h>
EthernetClient client;
// Name address for Open Weather Map API
const char* server = "api.openweathermap.org";
// Replace with your unique URL resource
const char* resource = "REPLACE_WITH_YOUR_URL_RESOURCE";
// How your resource variable should look like, but with your own COUNTRY CODE,
CITY and API KEY (that API KEY below is just an example):
//const char* resource = "/data/2.5/weather?
q=Porto,pt&appid=bd939aa3d23ff33d3c8f5dd1";
const unsigned long HTTP_TIMEOUT = 10000;  // max respone time from server
const size_t MAX_CONTENT_SIZE = 512;       // max size of the HTTP response
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
// The type of data that we want to extract from the page
struct clientData {
  char temp[8];
  char humidity[8];
};
// ARDUINO entry point #1: runs once when you press reset or power the board
void setup() {
  Serial.begin(9600);
  while (!Serial) {
    ;  // wait for serial port to initialize
  }
  Serial.println("Serial ready");
  if(!Ethernet.begin(mac)) {
    Serial.println("Failed to configure Ethernet");
    return;
  }
  Serial.println("Ethernet ready");
  delay(1000);
}
// ARDUINO entry point #2: runs over and over again forever
void loop() {
  if(connect(server)) {
    if(sendRequest(server, resource) && skipResponseHeaders()) {
      clientData clientData;
      if(readReponseContent(&clientData)) {
        printclientData(&clientData);
      }
    }
  }
  disconnect();
  wait();
}
// Open connection to the HTTP server
bool connect(const char* hostName) {
  Serial.print("Connect to ");
  Serial.println(hostName);
  bool ok = client.connect(hostName, 80);
  Serial.println(ok ? "Connected" : "Connection Failed!");
  return ok;
}
// Send the HTTP GET request to the server
bool sendRequest(const char* host, const char* resource) {
```

```
  Serial.print("GET ");
  Serial.println(resource);
  client.print("GET ");
  client.print(resource);
  client.println(" HTTP/1.1");
  client.print("Host: ");
  client.println(host);
  client.println("Connection: close");
  client.println();
  return true;
}
// Skip HTTP headers so that we are at the beginning of the response's body
bool skipResponseHeaders() {
  // HTTP headers end with an empty line
  char endOfHeaders[] = "\r\n\r\n";
  client.setTimeout(HTTP_TIMEOUT);
  bool ok = client.find(endOfHeaders);
  if (!ok) {
    Serial.println("No response or invalid response!");
  }
  return ok;
}
// Parse the JSON from the input string and extract the interesting values
// Here is the JSON we need to parse
/*{
    "coord": {
        "lon": -8.61,
        "lat": 41.15
    },
    "weather": [
        {
            "id": 800,
            "main": "Clear",
            "description": "clear sky",
            "icon": "01d"
        }
    ],
    "base": "stations",
    "main": {
        "temp": 296.15,
        "pressure": 1020,
        "humidity": 69,
        "temp_min": 296.15,
        "temp_max": 296.15
    },
    "visibility": 10000,
    "wind": {
        "speed": 4.6,
        "deg": 320
    },
    "clouds": {
        "all": 0
    },
    "dt": 1499869800,
    "sys": {
        "type": 1,
        "id": 5959,
        "message": 0.0022,
        "country": "PT",
        "sunrise": 1499836380,
        "sunset": 1499890019
    },
    "id": 2735943,
    "name": "Porto",
```

```
    "cod": 200
}*/
bool readReponseContent(struct clientData* clientData) {
  // Compute optimal size of the JSON buffer according to what we need to parse.
  // See https://bblanchon.github.io/ArduinoJson/assistant/
  const size_t bufferSize = JSON_ARRAY_SIZE(1) + JSON_OBJECT_SIZE(1) +
      2*JSON_OBJECT_SIZE(2) + JSON_OBJECT_SIZE(4) + JSON_OBJECT_SIZE(5) +
      JSON_OBJECT_SIZE(6) + JSON_OBJECT_SIZE(12) + 390;
  DynamicJsonBuffer jsonBuffer(bufferSize);
  JsonObject& root = jsonBuffer.parseObject(client);
  if (!root.success()) {
    Serial.println("JSON parsing failed!");
    return false;
  }
  // Here were copy the strings we're interested in using to your struct data
  strcpy(clientData->temp, root["main"]["temp"]);
  strcpy(clientData->humidity, root["main"]["humidity"]);
  // It's not mandatory to make a copy, you could just use the pointers
  // Since, they are pointing inside the "content" buffer, so you need to make
  // sure it's still in memory when you read the string
  return true;
}
// Print the data extracted from the JSON
void printclientData(const struct clientData* clientData) {
  Serial.print("Temp = ");
  Serial.println(clientData->temp);
  Serial.print("Humidity = ");
  Serial.println(clientData->humidity);
}
// Close the connection with the HTTP server
void disconnect() {
  Serial.println("Disconnect");
  client.stop();
}
// Pause for a 1 minute
void wait() {
  Serial.println("Wait 60 seconds");
  delay(60000);
}
```

view raw Projects/Arduino-JSON/Arduino_HTTP_GET_Client.ino

Note: make sure your replace the resource variable with your unique OpenWeatherMap URL resource:

```
const char* resource = "REPLACE_WITH_YOUR_URL_RESOURCE";
```