

# IoT for entrepreneurs

## *the code explained (part 2)*

Clément Levallois

2017-09-04

# Table of Contents

The code, explained steps by steps .....	1
The end .....	3



## The code, explained steps by steps

```
void callAQI(String location){ ①
```

- ① here starts the definition of the function which will connect the board to a website giving the air pollution in a given place (the parameter `location`). This function will be executed when called in the `loop` function that you have written in the file `code_for_ar_quality_screen.io`

```
char* token = "put your own token here"; ①
```

- ① create the variable `token`. The value for this variable is the token you can obtain for free on this website: <http://aqicn.org/data-platform/token/#/>



a token is a kind of password (sometimes called an API secret, or a secret token). It certifies your identity when accessing an API.

```
char* host = "api.waqi.info"; ①  
String webaddress = "/search/?token=" + String(token) + "&keyword=" + location; ②  
int httpPort = 80; ③
```

- ① the variable `host` contains the first part of the url (address of the website) where we will retrieve the air quality.
- ② this is the second part of the url. You see that it includes your token (transformed into a `String`) and the `location` where the air pollution is measured.
- ③ to send and receive data via http, even if we don't see it, we must specify the port - this is a bit like a gate number. 80 is the default port number for http. Check [this](#) reference.

```
if (!client.connect(host, httpPort)) { ①  
    Serial.println("connection failed"); ②  
    return ; ③  
}
```

- ① the variable "client" (which we have created in the first file) now tries to connect to the website. If it fails (see the `!` in the if condition), then:

② send a message saying the connection to the website has failed

③ **return**, which means stopping there and leaving the function **only if the connection has failed**.

The following lines of code will be executed only if our board could connect to <http://api.waqi.info>

```
client.print("GET ");①
client.println(" HTTP/1.1");
client.print(webaddress); ②
client.print("Host: ");
client.println(host);
client.println("Connection: close"); ③
client.println();
delay(500);

unsigned long timeout = millis(); ④
while (client.available() == 0) {
  if (millis() - timeout > 5000) {
    Serial.println(">>> Client Timeout !");
    client.stop();
    return ;
  }
}

return; ⑤
}
```

① all the lines of code starting with **client.print** contribute to connecting your board to the specific page of the website where the air quality for your location is available.

② this specific line is the one where your particular location and token will be taken into account: see the **webaddress** we defined above.

③ this tells the board to disconnect from the website when the connection will be over.

④ this tells the board to wait until the response from the website has been completely received by the client.

⑤ we are done and we can leave the function **callAQI**. The info retrieved on the website is now stored in our **client** variable.

To summarize so far:

- the function (method) **callAQI** connected the board to the website <http://api.waqi.info>
- on this website, it navigated to a page defined by our token and the location where we want to measure the air quality
- the result (info about the air quality) has been saved in the memory of our board, in the variable **client** we used for the connection.

We need more steps to extract the info we need from the **client** variable. We could have added extra lines to the function **callAQI**, but for a pure matter of making our code more readable, we put these lines of code in a separate function that we name **readAQIResponse**:

We now explain step by step what this function does:

```
String readAQIResponse(){ ①
    while (client.available()) {②

        String line = client.readStringUntil('\n'); ③
        Serial.println(line); ④
        if (line.indexOf("data") < 0) { ⑤
            continue; ⑥
        }
        DynamicJsonBuffer jsonBuffer (1000); ⑦
        JsonObject& root = jsonBuffer.parseObject(line); ⑧
        String aqiValue = root["data"][0]["aqi"]; ⑨
        return aqiValue; ⑩
    }
    Serial.println("the client was not available. Returning no data");
    return "no data received";⑪
}
```

- ① the function is called `readAQIResponse`. When it finishes, it provides (it "returns") a value of type `String`
- ② this `while` loop will read the info that the `client` variable received (see just above)
- ③ in code, the end of a line is marked by a symbol invisible to humans: it is `\n`. So the first line is all characters until `\n`.
- ④ To debug (help us understand when and why something is wrong), we print this line on the computer
- ⑤ we interested only in the line that contains the word "data". If this word is not present, its index will be negative. See more explanations [here](#).
- ⑥ so if "data" is not contained in the line, just move on to the next: ignore the rest of the function and move back to the beginning of the `while` loop.
- ⑦ Here, we prepare a new variable which will help us extract the info we need from this line.
- ⑧ The `root` variable now contains the line, formatted as a json object (see more on json objects [here](#)).
- ⑨ In this `root` object, we reach for the subelement `["data"][0]["aqi"]`. This is the value of the air pollution for our location.
- ⑩ we return this value, leave the `while` loop and the function.
- ⑪ if the client was not available, we received no data.

## The end

Find references for this lesson, and other lessons, [here](#).



This course is made by Clement Levallois.

Discover my other courses in data / tech for business: <https://www.clementlevallois.net>

Or get in touch via Twitter: [@seinecle](https://twitter.com/seinecle)