IoT for entrepreneurs

the code explained (part 1)

Clément Levallois

2017-09-04

Table of Contents

last modified: 2017-09-20

The code, explained steps by steps

```
// libraries pre-installed with the board

#include <ESP8266WiFi.h> ①
#include <SoftwareSerial.h> ②
#include <Wire.h> ③
```

① The library for the Huzzah Feather (the specific board we use)

② The library enabling the board to send info to the computer (useful to debug!)

③ The library enabling Wifi

```
// libraries we added "manually"

#include <Adafruit_GFX.h> ①
#include <Adafruit_FeatherOLED.h> ②
#include <ArduinoJson.h> ③
```

① The library to help the board deal with graphics

② The library to help the board deal with the specific screen Feather OLED

③ The library that saves us time when dealing with text formatted in JSON

The next lines of code are used to store some useful information:

```
char* ssid     = "name of wifi network here"; ①
char* password = "password here"; ②
```

① means: we create the variable called `ssid`. We give it the value 'name of wifi network here'. This variable is of type `chars*` (characters)

② means: we create the variable called `password`. We give it the value 'password here'. This variable is of type `chars*` (characters)

Of coure, these values are just examples and you should put the name of your own wifi network and password.

Don’t forget the semi-colon at the end of the lines!

```
String location = "Saint-Etienne"; ①
```

① means: we create the variable called `location`. We give it the value "Saint-Etienne" (you can choose another place of course). This variable is of type `String` (`String` is the technical term for text).

```
Adafruit_FeatherOLED screen = Adafruit_FeatherOLED(); ①
```

① means: we create the variable called "screen". We give it the value returned by the function `Adafruit_FeatherOLED()`. The variable is of type `Adafruit_FeatherOLED`.

```
WiFiClient client; ①
```

① means: we create the variable named "client", of type "WifiClient". It has no value yet.

```
void setup() { ①
```

① This is called a function or method. The `setup` funcion is executed just once, when the board is powered up.

Don’t know what functions are? Check here.

```
// initialize the communication between the board and our computer.
// useful to display useful info.

Serial.begin(9600); ①
while (!Serial) ②
  { ③
    ; ④
  } ⑤
```

① means: we execute the method called `begin` on the variable `Serial`.

② means: as long as the variable "Serial" is not equal to "true", execute the lines of codes in the curly braces `{ }` in <3> and <5>. More info here.

③ The instructions in the curly braces are just this `;`, which do nothing. So the board is basically waiting until `Serial` switches to "true".

```
//initialize the screen and hides the battery sign

screen.init(); ①
screen.setBatteryVisible(false); ②
```

① means: we execute the method called `init` on the variable `screen`.

② means: we execute the method called `setBatteryVisible` on the variable `screen`. We use the parameter `false` for this method.

```
//initialize the wifi connection

WiFi.begin(ssid, password); ①
```

① means: a variable from the board (not created by us) is called "Wifi". We execute the method `begin` on it, which will start the connection to the wifi. So we give it 2 parameters: the name and password for the wifi, which we defined above.

```
// the program freezes as long as the wifi is not connected

while (WiFi.status() != WL_CONNECTED ) { ①
    delay(500); ②
}
```

① This another `while` loop, like the one above. The `status` method of `Wifi` returns a value which is `true` when the wifi is connected, and `false` otherwise.`WL_CONNECTED` is a variable with a value of `true`.

② `delay` is a simple function which freezes the board. For how long? For the duration shown in the parentheses, in milliseconds. So here, `delay(500)` freezes the board for 0.5 seconds before it retries to connect to the wifi.

```
} ①
```

① Do not forget this closing bracket, corresponding to the opening brack from the line above `void setup() {`

```
void loop() { ①
```

① the lines of code inside the loop function will be executed until the last, then start over again and again. This is where the interesting stuff happens! See here for more.

```
screen.clearDisplay(); ①
screen.clearMsgArea(); ②
```

① deletes what is on screen

② deletes the previous message

```
//get the quality of air at location
callAQI(location); ① ②
```

① means: execute the method `callAQI`. Where does it come from? You will write the code for this function in a separate file. The role of this function is to connect via wifi to http://waqi.info/ and return the air quality for a given location.

② `location` is a parameter to the function. It is a variable you defined above (scroll up).

```
String aqiNumber = readAQIResponse(); ①
```

① We create the variable `aqiNumber`, of type `String`. The value assigned to this variable is the number returned by the function `readAQIResponse()`.

Where does the `readAQIResponse` come from? It is a function written in another file by you. It picks the text returned by the previous function (`callAQI`), and extracts from it the number representing the air quality.

```
//display the air quality on the screen

screen.println("air quality"); ①
screen.println("at "+ location + ":"); ②
screen.println(aqiNumber); ③
screen.display();
```

① will write "air quality" on the 1st line of the screen

② will write "at Saint-Etienne" on the 2nd line of the screen (if the value of your location variable is "Saint-Etienne")

③ will write the air pollution index retrieved from above

④ will show the message on screen (if you forget this line, the message is not visible).

```
// wait 5 seconds
delay(5000);
} ①
```

① Don't forget this closing curly brace. It closes the function opened above with `void loop (){`

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: http://www.clementlevallois.net

Or get in touch via Twitter: @seinecle