Build a WiFi Weather Station with the ESP8266

by Marco Schwartz of Open Home Automation
http://www.openhomeautomation.net/

I love to build DIY home automation systems using open-source hardware, especially with the amazing ESP8266 WiFi chip. However, it can be a bit tricky to get started, and you might feel lost in front of all the options that are available to you.

This is why I wrote this guide to help you build your first home automation system using the ESP8266: a simple weather measurement station that works via WiFi. We will connect a DHT11 sensor to the ESP8266 board, and access the data via WiFi.

To do so, we will run a simple web server on the ESP8266 chip, that will display the results inside a web page. We will also make this web page responsive, so it looks nice even if you are using a smartphone. Finally, we will use the fact that the ESP8266 is already connected to the web to grab weather measurements online & display it in the interface as well.

Hardware & Software Requirements

For this project, you will of course need a board with the ESP8266 chip. I will for example use an Adafruit Feather ESP8266 board, but you can of course use any board based on the ESP8266 chip.

However, I recommend using development boards that have USB-to-Serial converter onboard as you just need to plug them to your computer via a USB cable.

You will also need a temperature sensor. I used a DHT11 sensor, which is cheap, very easy to use & that will allow us to measure the ambient temperature & humidity.

If you are not using a development board, you will also need a 3.3V FTDI USB module to program the ESP8266 chip, as well as a breadboard power supply to power the chip. Finally, you will also need some jumper wires & a breadboard.

This is a list of all the components that will be used in this chapter:

- Adafruit Feather ESP8266 WiFi board
- DHT11 sensor

- Breadboard
- Jumper wires

On the software side, you will need the latest version of the Arduino IDE that you can get from:

http://www.arduino.cc/en/Main/Software

Then, follow this procedure to add the ESP8266 board to the Arduino IDE:

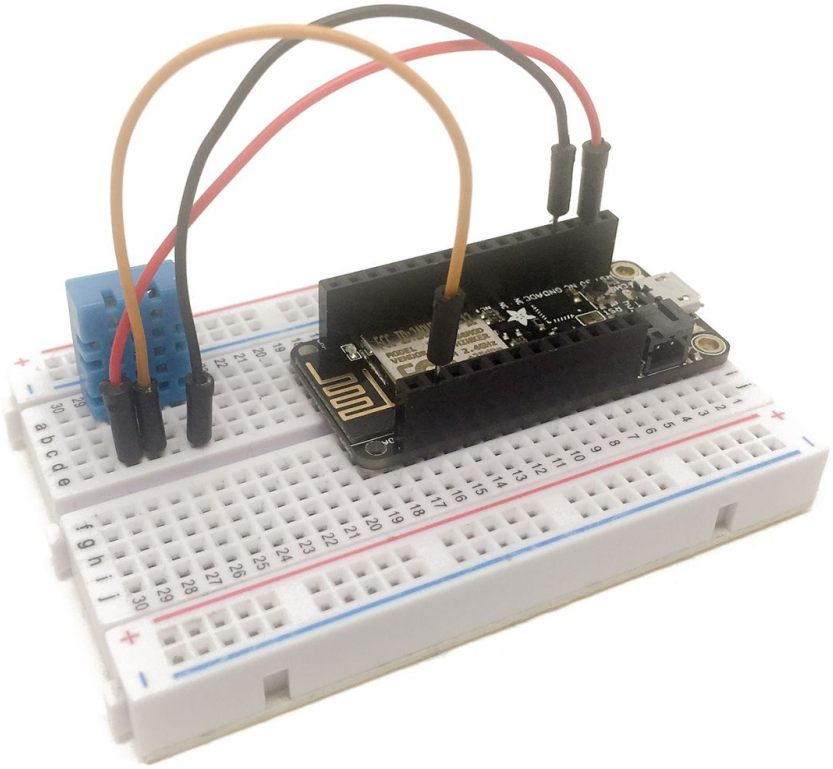
- Start the Arduino IDE and open the Preferences window.
- Enter the following URL into the **Additional Board Manager URLs** field: http://arduino.esp8266.com/package_esp8266com_index.json
- Open Boards Manager from Tools > Board menu and install the esp8266 platform.

You will also need the Adafruit DHT library that you can get from the Arduino library manager.

Hardware Configuration

We are first going to see how to configure the hardware to use the ESP8266 board. If you are using a development board, you simply need to plug the board to your computer via a USB cable.

Once this is done, simply insert the DHT11 sensor on the breadboard. Then, connect the left pin to VCC of the board, the right pin to GND, and the pin next to VCC to the GPIO5 pin on your ESP8266 development. This is the final result:



Testing the Sensor

We are now going to test the sensor. Again, remember that we are using the Arduino IDE, so we can code just like we would do using an Arduino board. Here, we will simply print the value of the temperature inside the Serial monitor of the Arduino IDE.

This is the complete code for this part:

```
// Libraries
#include "DHT.h"

// Pin
#define DHTPIN 5

// Use DHT11 sensor
#define DHTTYPE DHT11

// Initialize DHT sensor
DHT dht(DHTPIN, DHTTYPE, 15);

void setup() {
```

```
 // Start Serial
 Serial.begin(115200);

 // Init DHT
 dht.begin();
}

void loop() {

 // Reading temperature and humidity
 float h = dht.readHumidity();
 // Read temperature as Celsius
 float t = dht.readTemperature();

 // Display data
 Serial.print("Humidity: ");
 Serial.print(h);
 Serial.print(" %\t");
 Serial.print("Temperature: ");
 Serial.print(t);
 Serial.println(" *C ");

 // Wait a few seconds between measurements.
 delay(2000);

}
```

Let's see the details of the code. You can see that all the measurement part is contained inside the loop() function, which makes the code inside it repeat every 2 seconds.

Then, we read data from the DHT11 sensor, print the value of the temperature & humidity on the Serial port.

You can now paste this code in the Arduino IDE. Then, go in Tools>Boards, and select the correct board from the list. If you can't find the board you are using, select "Generic ESP8266 Module". Also select the correct Serial port that corresponds to the board or FTDI converter you are using.

You should immediately see the temperature & humidity readings inside the Serial monitor. My sensor was reading around 24 degrees Celsius when I tested it, which is a realistic value.

Accessing the Sensor via WiFi

At this point, we are sure that the sensor is working and that data can be read by the ESP8266 chip. Now, we are going to build the sketch that will connect to your WiFi network, and then serve a web page that will display the results in live.

As this sketch is quite long, I will only detail the most important parts here. You can of course find the complete code for this project inside the GitHub repository of the project:

https://github.com/openhomeautomation/esp8266-weather-station

First, you need to set up your own WiFi network name & password in the code:

```
const char* ssid = "your_wifi_network_name";
const char* password = "your_wifi_network_password";
```

After that, we create a web server on port 80:

```
WiFiServer server(80);
```

Then, inside the setup() function of the sketch, we connect the ESP8266 to the WiFi network:

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

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

Then, we start the server, and print the IP address on the Serial port:

```
// Start the server
server.begin();
Serial.println("Server started");

// Print the IP address
Serial.println(WiFi.localIP());
```

Inside the loop() function of the sketch, we check if a client is connected to the ESP8266:

```
WiFiClient client = server.available();
```

Then, we read data from the sensor:

```
// Reading temperature and humidity
float h = dht.readHumidity();

// Read temperature as Celsius
float t = dht.readTemperature();
```

After that, we read the incoming request from the client:

```
String req = client.readStringUntil('\r');
Serial.println(req);
client.flush()
```

Then, we prepare our answer. What we want here is to serve the data to the client in an elegant way. That's why we will use the Bootstrap CSS framework, that will make our page looks pretty. It will also makes it responsive, so it will look great on mobile devices as well.

The first part is to send the Head tag of the HTML page, which includes the Bootstrap CSS file. We also set in this part the refresh rate of the page, which will be automatically refreshed every minute:

```
String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n";
s += "<head>";
s += "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">";
s += "<meta http-equiv=\"refresh\" content=\"60\" />";
s += "<script src=\"https://code.jquery.com/jquery-2.1.3.min.js\"></script>";
s += "<link rel=\"stylesheet\" href=";
s += "\"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css\">";
s += "<style>body{font-size: 24px;} .voffset {margin-top: 30px;}</style>";
s += "</head>";
```

Then, we send the core of the page, which consists in simply displaying the temperature & humidity data:

```
s += "<div class=\"container\">";
s += "<h1>WiFi Weather Station</h1>";

s += "<div class=\"row voffset\">";
s += "<div class=\"col-md-3\">Temperature: </div>";
s += "<div class=\"col-md-3\">" + String(t) + "</div>";
s += "<div class=\"col-md-3\">Humidity: </div>"
s += "<div class=\"col-md-3\">" + String(h) + "</div>";
s += "</div>";

s += "</div>";
```

Finally, we send this to the client, and wait until the client disconnects from our board:

```
client.print(s);
delay(1);
Serial.println("Client disconnected");
```

It's now time to test the project. Get the code from the GitHub repository of this project:

https://github.com/openhomeautomation/esp8266-weather-station

Then, modify it with your own parameters, and upload the code to the board, using the instructions we saw earlier.

After that, open the Serial monitor of the Arduino IDE. You should see that the IP address is displayed inside the Serial monitor.

Then, simply go to a web browser and type this IP address. You should immediately see the measured data in your browser:

Note that you can also do the same from your mobile phone or tablet, and it will work just as well!

This is already the end of this guide about open-source home automation! I hope this simple project gave you an idea of what you can do with the ESP8266 for home automation applications.

If that's not done yet, you can of course follow my website on Facebook & on Twitter.

Thanks again, and all the best for your home automation projects!

Marco Schwartz
contact@openhomeautomation.net