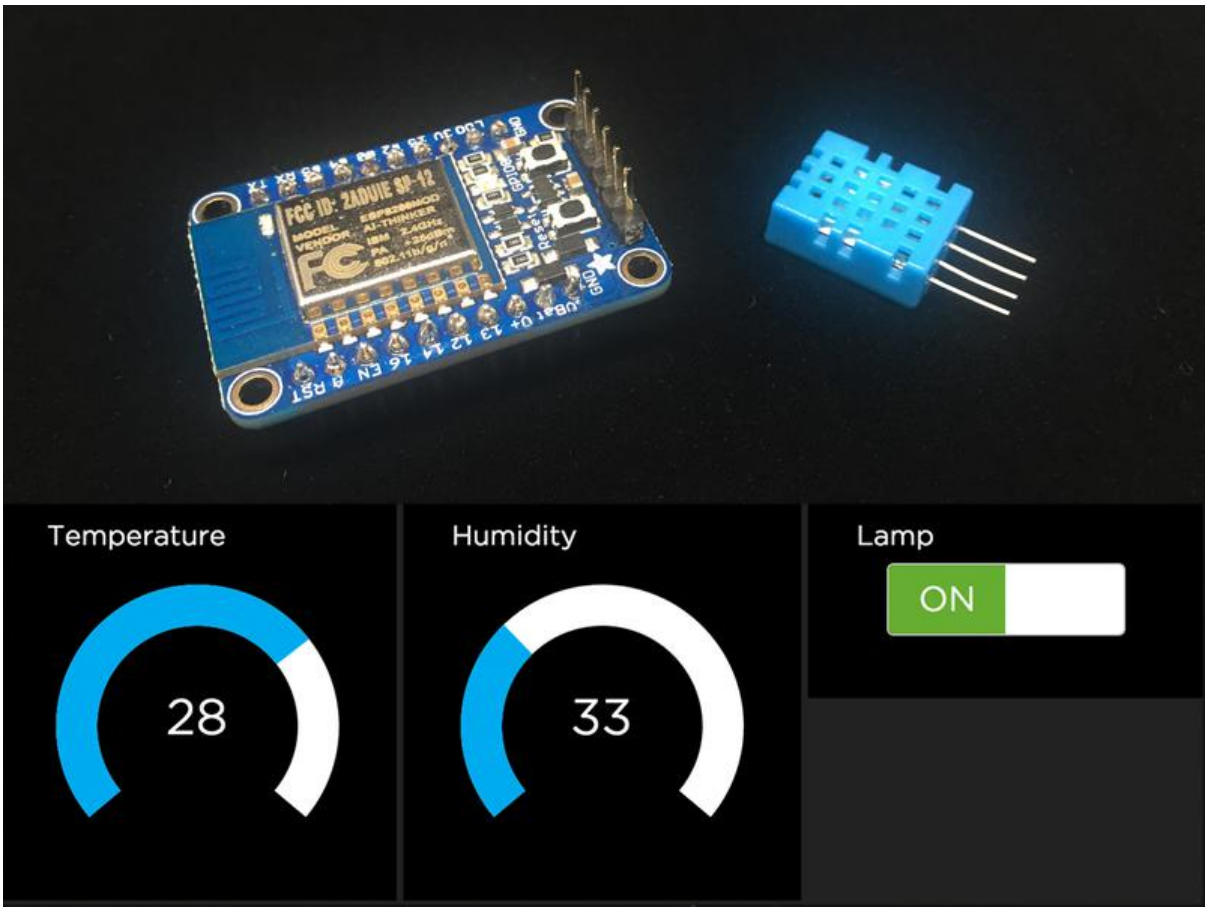
Home Automation in the Cloud with the ESP8266 & Adafruit IO

Created by Marc-Olivier Schwartz



https://learn.adafruit.com/home-automation-in-the-cloud-with-the-esp8266-and-adafruit-io

Last updated on 2021-11-15 06:30:41 PM EST

Table of Contents

Introduction 3

Hardware & Software Requirements 3

Building the Sensor Module 4

Building the Lamp Controller Module 5

Programming the Modules 6

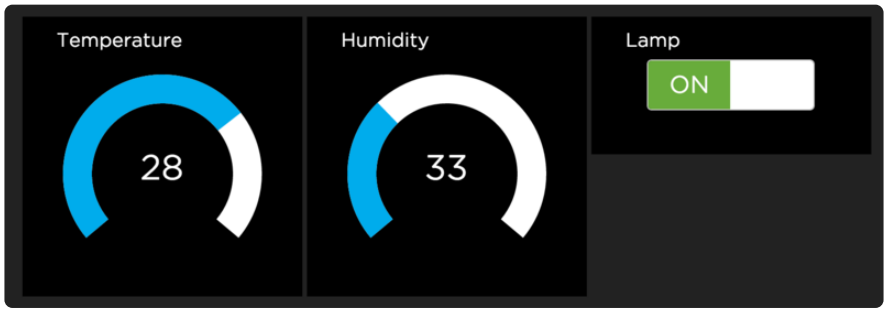
Controlling the Project from Adafruit IO 10

How to Go Further 12

Introduction

The ESP8266 is an amazing little chip which has onboard WiFi capabilities, an integrated processor, and also comes at less than $10.

This makes it the perfect chip for DIY electronics projects, and especially in the home automation field. There are many breakout boards available for the ESP8266, but in this guide we are going to use the Adafruit HUZZAH ESP8266 breakout, which is a very convenient ESP8266 breakout board.



In this project, we are going to use the ESP8266 to build two components which are very useful in home automation: a sensor module, and a lamp controller.

We are also going to link these two projects to Adafruit IO, which will allow you to control this small home automation system from anywhere in the world. Let's start!

Hardware & Software Requirements

Let's first see what are the required hardware modules for this project. Remember that we are going to build two different modules: a sensor board, and a lamp controller.

For both modules, you will need one Adafruit HUZZAH ESP8266 breakout, which will be the central component of each module. You will also need a breadboard and some male/male jumper wires to make the necessary connections.

For the sensor board, you will also need a DHT22 sensor, which we will use to measure the temperature & humidity. You can also use a DHT11 sensor which is cheaper but less precise (which is the one you will see on the pictures), you will only have to change one line of code.

For the lamp controller, we are going to use a PowerSwitch Tail 2, which makes it really easy to attach a lamp (or any electrical device) to your project.

Finally, you will need one FTDI friend board (or a FTDI/USB cable) to program the ESP8266 breakout board.

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

https://www.arduino.cc/en/main/software (https://adafru.it/fGz)

After it is installed, add the ESP8266 package by following these steps:

- Open the Preferences window in the Arduino IDE
- Enter `http://arduino.esp8266.com/stable/package_esp8266com_index.json` into Additional Board Manager URLs field.
- Open the Boards Manager from Tools > Board menu and install the esp8266 platform

You will also need to install the following Arduino libraries:

- Adafruit MQTT library (https://adafru.it/fp6)
- DHT sensor library (https://adafru.it/aJX)

Finally, also create an Adafruit IO account if it is not done yet. You can do so by going to:

http://io.adafruit.com/ (https://adafru.it/fH9)

You will need your Adafruit account name & Adafruit AIO key later in this tutorial.

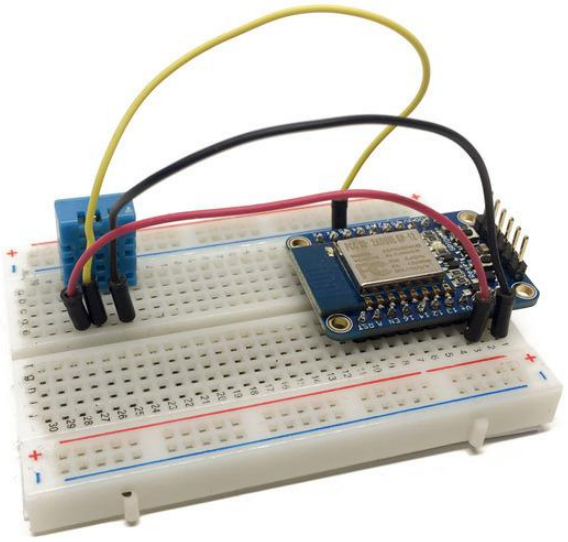
Building the Sensor Module

Let's now see how to configure the hardware for the sensor module. Thanks to the Adafruit ESP8266 board, it is really simple to build this module on a breadboard.

The first step is to place the ESP8266 breakout board on the breadboard, as well as the DHT sensor. Then, connect the V+ (or VCC) pin from the ESP8266 board to the first pin of the DHT sensor (look at the picture below).

After that, connect the GND pin of the ESP8266 to the last pin of the DHT. Finally, connect pin number 5 of the ESP8266 breakout board to the pin number 2 of the DHT sensor.

This is the final result:



As mentionned before, you will see that I used a DHT11 sensor on this picture. However, the code you will find on GitHub is made for the DHT22 sensor, and I really suggest using a DHT22 sensor as it is more precise.

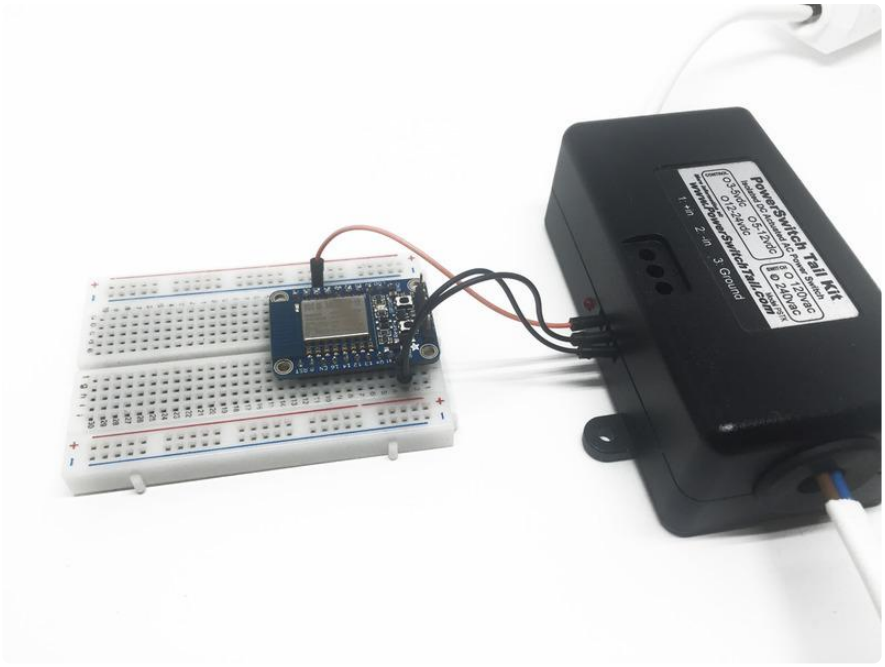
Building the Lamp Controller Module

We are now going to see how to build the lamp controller module. This first step is to place the ESP8266 breakout board on the breadboard.

Then, the only thing you need to do is to connect the PowerSwitch Tail Kit. Connect the two pins on the right (-in and Ground) on the GND pin of the ESP8266 board and the +in pin to the pin number 5 of the ESP8266.

Then, also connect a lamp or any electrical device to the PowerSwitch, and the other end of the PowerSwitch to the mains electricity.

This is the completey assembled lamp controller:



Programming the Modules

In this section, we are going to see how to program the two modules that you just built. We will start by looking at the most important pieces of the code, first for the sensor module, and then for the lamp controller. Note that you can find the whole code for this guide on the corresponding GitHub repository:

https://github.com/openhomeautomation/adafruit-io-esp8266 (https://adafru.it/fHb)

The sketch for the ESP8266 sensor module starts by including the right libraries:

```
#include &lt;ESP8266WiFi.h&gt;
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include "DHT.h"
```

Then, we define on which pin the DHT sensor is connected to:

```
#define DHTPIN 5
#define DHTTYPE DHT22
```

After that, there is the section of the code where you need to modify some data. You need to enter your WiFi network name & password, your Adafruit account name, and your AIO key:

```
// WiFi parameters
#define WLAN_SSID       "WLAN_SSID"
#define WLAN_PASS       "WLAN_PASS"
```

```
// Adafruit IO
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "AIO_USERNAME"
#define AIO_KEY         "AIO_KEY"
```

We also create the DHT sensor instance:

```
DHT dht(DHTPIN, DHTTYPE, 15);
```

We also define on which feed we want to send the data to. By default, the sketch will simply send temperature measurements to a feed called temperature, and humidity measurements to a feed called humidity.

You can of course change this by changing the following line:

```
const char TEMPERATURE_FEED[] PROGMEM = AIO_USERNAME "/feeds/temperature";
```

In the setup() function of the sketch, we initialise the DHT sensor:

```
dht.begin();
```

Then, in the loop() function of the sketch, we constantly check if we are still connected to Adafruit IO. If that's not the case, we reconnect the board to Adafruit IO:

```
if(! mqtt.ping(3)) {
  // reconnect to adafruit io
  if(! mqtt.connected())
    connect();
}
```

After that, we make the temperature & humidity measurements:

```
int humidity_data = (int)dht.readHumidity();
int temperature_data = (int)dht.readTemperature();
```

We then send these measurements to their respective feed:

```
if (! temperature.publish(temperature_data))
  Serial.println(F("Failed to publish temperature"));
else
  Serial.println(F("Temperature published!"));

if (! humidity.publish(humidity_data))
  Serial.println(F("Failed to publish humidity"));
else
  Serial.println(F("Humidity published!"));
```

Finally, we repeat the operation every 10 seconds:

```
delay(10000);
```

Let's now see the specificities of the lamp controller sketch. We have to define on which pin the PowerSwitch Tail is connected to:

```
const int lamp_pin = 5;
```

We also define a new feed for the project, simply called lamp:

```
const char LAMP_FEED[] PROGMEM = AIO_USERNAME "/feeds/lamp";
```

In the setup() function, we set the pin of the PowerSwitch Tail to an output:

```
pinMode(lamp_pin, OUTPUT);
```

We also make the project subscribe to the lamp feed:

```
mqtt.subscribe(&amp;lamp);
```

Then, in the loop() function, we have to listen to incoming messages from Adafruit IO, to know if we need to turn the lamp on or off. It starts by creating a subscription instance:

```
Adafruit_MQTT_Subscribe *subscription;
```

Then, we listen for incoming messages. If the message is 'ON', we switch the lamp on. And if it's 'OFF', we simply switch the lamp off again. This is done by the following piece of code:

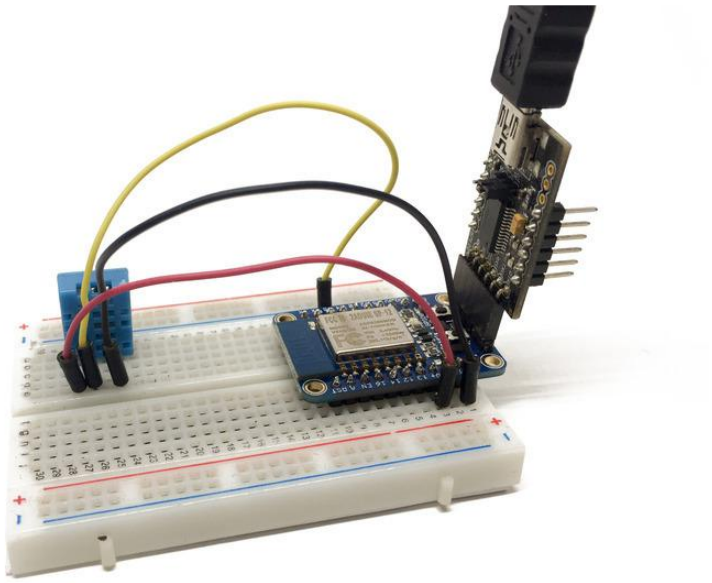
```
while (subscription = mqtt.readSubscription(1000)) {
  // we only care about the lamp events
```

```
  if (subscription == &amp;lamp) {

    // convert mqtt ascii payload to int
    char *value = (char *)lamp.lastread;
    Serial.print(F("Received: "));
    Serial.println(value);

    // Apply message to lamp
    String message = String(value);
    message.trim();
    if (message == "ON") {digitalWrite(lamp_pin, HIGH);}
    if (message == "OFF") {digitalWrite(lamp_pin, LOW);}

  }
}
```

Note that you can find the complete code for this guide inside the GitHub repository of the project:

https://github.com/openhomeautomation/adafruit-io-esp8266 (https://adafru.it/fHb)

We are now going to program the board. The first step is to plug the FTDI friend board to the ESP8266 breakout:



Open the code with your Arduino IDE, and select 'Adafruit HUZZAH ESP8266' from the Tools>Boards menu of the Arduino IDE. Also select the Serial port corresponding to the FTDI board you are using.

Then, put the ESP8266 board in bootloader mode so you can program it. On the Adafruit ESP8266 board, it is really simple: just hold the GPIO0 button pressed, and then press the Reset button.

Make sure that you also modified the code with your own data (WiFi network credentials & Adafruit IO credentials).

After that, upload the code to the board. When it is finished, open the Serial monitor, and reset the board: you should see that the board automatically connects to Adafruit IO.

Of course, you need to repeat the operation for both modules in the project.

If your ESP8266 resets when you run the progem, note that there have been recent changes to the Adafruit_IO library. You will have to remove references to PROGMEM. Edit your sketch to match the code sample below:

```
WiFiClient client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server and
login details.
Adafruit_MQTT_Client mqtt(&amp;client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME,
AIO_KEY);

/************************** Feeds ***********************************/

// Setup a feed called 'lamp' for subscribing to changes.
// Notice MQTT paths for AIO follow the form: &lt;username&gt;/feeds/
&lt;feedname&gt;
Adafruit_MQTT_Subscribe lamp = Adafruit_MQTT_Subscribe(&amp;mqtt, AIO_USERNAME "/
feeds/lamp");
```

(if you're using the sensor sketch, make the appropriate changes):

```
// Setup feeds for temperature &amp; humidity
Adafruit_MQTT_Publish temperature = Adafruit_MQTT_Publish(&amp;mqtt, AIO_USERNAME "/
feeds/temperature");

Adafruit_MQTT_Publish humidity = Adafruit_MQTT_Publish(&amp;mqtt, AIO_USERNAME "/
feeds/humidity");
```

Controlling the Project from Adafruit IO

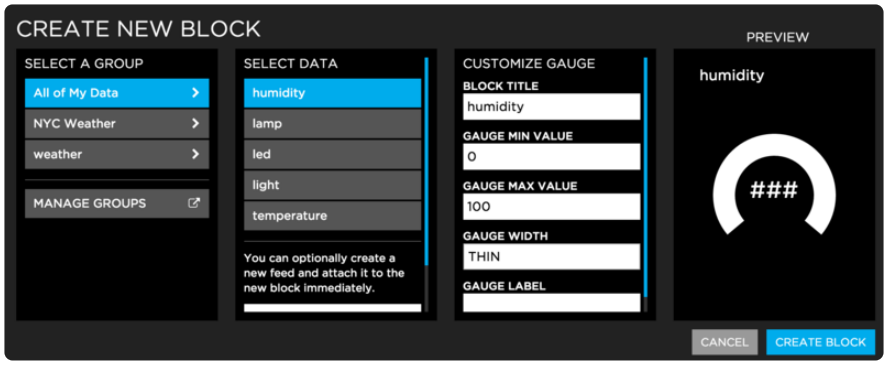
It's now time to create our Adafruit IO dashboard & control our modules from anywhere in the world.

The first step is to create a new dashboard at:

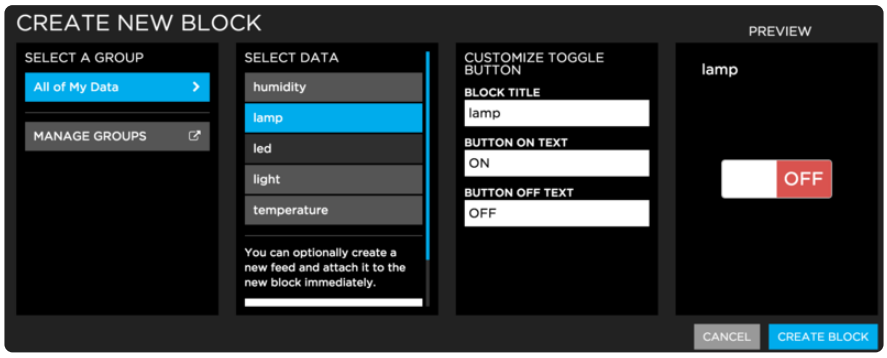
https://io.adafruit.com (https://adafru.it/eZ8)

If it is not done yet, create the feeds that we used in our project: temperature, humidity, and lamp.

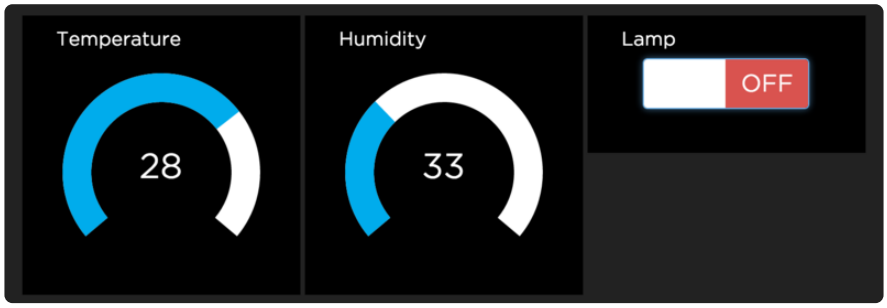
Then, add a new 'Gauge' widget, and link it to the temperature feed. Do the same for humidity:



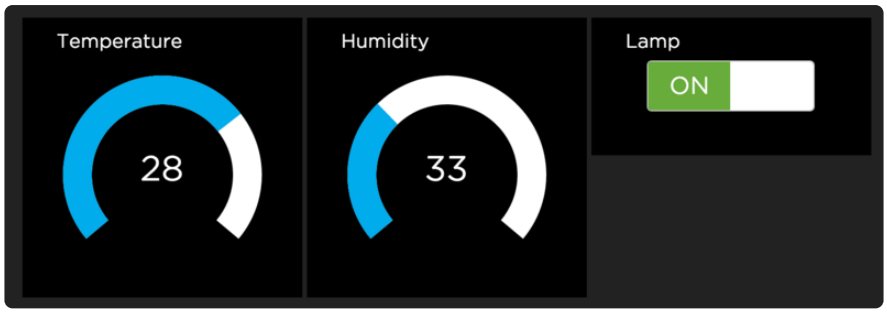
Then, create a 'toggle button' widget, and link it to the lamp feed:



Your dashboard is now ready to be used. If you followed all the instructions in this guide so far, you should see that the temperature & humidity gauges have already been updated, as the ESP8266 sensor module already sent data to Adafruit IO:



Finally, test the toggle button by clicking on it:



You should immediately see that the lamp (or the electrical device) connected to the PowerSwitch Tail is turning on. Simply click on the button again to switch it off.

Congratulations, you now learned the basics on how to build home automation modules with the ESP8266 & control them from anywhere with Adafruit IO!

How to Go Further

There are of course several ways to improve this project. One way is to connect more of these modules to your Adafruit IO dashboard. You can perfectly have several lamp modules for example, and control them all from your Adafruit IO dashboard.

You can also experiment with different kind of modules. For example, you can build cheap motion sensors based on the ESP8266, and monitor them as well from your Adafruit IO dashboard.