Connect the ESP8266 WiFi Chip to your Raspberry Pi

In this project, we are going to make a typical home automation project: a WiFi weather measurement station. We will connect a DHT11 sensor to an ESP8266 board, and access the data via WiFi. However, here, we are actually going to grab that data from a Raspberry Pi, and make the Pi display the data on a simple graphical interface.

To do so, we will run a simple web server on the ESP8266 chip. The Raspberry Pi will then access this data via WiFi, and display it graphically. The nice thing is can you can apply this in several other projects, to make the Raspberry Pi the 'hub' of your home, with several ESP8266-based devices connected to it. Let's start!

Hardware & Software Requirements

For this project, you will of course need an ESP8266 chip. Here, I used the Adafruit ESP8266 board.

You will also need a fully configured Raspberry Pi board, with the Raspbian operating system installed on it.

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.

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

- Raspberry Pi board
- Adafruit ESP8266 breakout board
- USB 5V FTDI converter
- DHT11 sensor
- Breadboard
- Jumper wires

You will need to have the latest version of the Arduino IDE installed. You will also need the DHT library. You can install it from the Arduino library manager, that you can access from **Sketch>Include Libraries>Manage Libraries** inside the Arduino IDE.

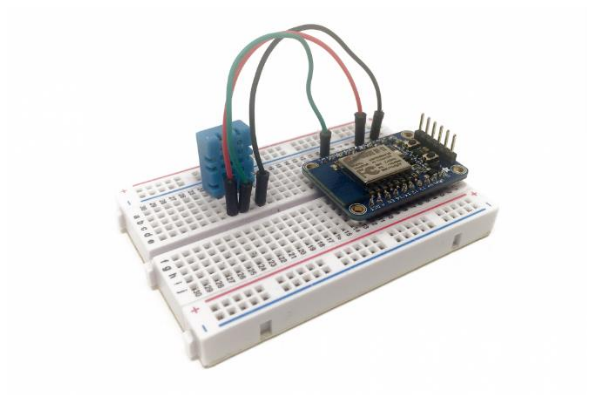
On your Raspberry Pi, you will also need to have Node.js installed. To do so, follow the instructions from:

https://learn.adafruit.com/node-embedded-development/installing-node-dot-js

Hardware Configuration

We are first going to see how to configure the hardware to use the ESP8266 board. First, place the Adafruit ESP8266 board on the breadboard.

Once this is done, simply insert the DHT11 sensor on the breadboard. Then, connect the left pin of the sensor to the 3.3V pin of the ESP8266, the right pin to GND (blue power rail), and the pin next to VCC to the GPIO5 pin on your ESP8266 board. This is the final result, not showing the USB-to-Serial FTDI converter:



Make sure that you connected everything according to the instructions above, or you won't be able to continue.

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.

Note that the complete code can also be found inside the GitHub repository of the book:

https://github.com/openhomeautomation/connect-esp8266-rpi
You can now paste this code in the Arduino IDE. Then, go in **Tools>Boards**, and select the Adafruit ESP8266 board from the list.

After that, we need to put the board in bootloader mode, so we can program it. To do so, refer to the first chapter of this part of the book.

Now, upload the code to the board, and open the Serial monitor when this is done. Also set the Serial monitor speed to 115200.

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 make the measurements accessible by the Raspberry Pi.

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.

It starts by including the required libraries for this project:

```
#include "ESP8266WiFi.h"
#include <aREST.h>
#include "DHT.h"
```

Then, 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";
```

We also declare the aREST API, that will help us control the board remotely from the Raspberry Pi:

```
aREST rest = aREST();
```

Now, we also declare two variables that will contain the measurements made by the sensor:

```
int temperature;
int humidity;
```

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 expose the two measurement variables to the aREST API, so they can be access from the outside:

```
rest.variable("temperature",&temperature);
rest.variable("humidity",&humidity);
```

After that, we set the name & ID of the board:

```
rest.set_id("1");
rest.set_name("sensor_module");
```

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, and handle the request:

```
WiFiClient client = server.available();
if (!client) {
```

```
return;
}
while(!client.available()){
delay(1);
}
rest.handle(client);
```

Then, we read data from the sensor:

```
temperature = dht.readTemperature();
humidity = dht.readHumidity();
```

It's now time to upload the sketch the board. You can grab the complete code from:

https://github.com/openhomeautomation/connect-esp8266-rpi

Follow the instructions we saw in the previous section to upload the code to the board. Then, open the Serial monitor to get the IP address of the board, you will need it soon.

Connecting the Board to Your Raspberry Pi

We are now going to see how to connect the board to your Raspberry Pi, so it can display the data in a nice graphical interface.

The only thing that you need to change to run this interface is the IP address of your ESP8266 board inside the app.js file:

```
rest.addDevice('http','192.168.0.100');
```

You can get all the code from:

https://github.com/openhomeautomation/connect-esp8266-rpi

Place all the files into a folder on your Pi, navigate to this folder with a terminal, and type:

```
sudo npm install express jade arest
```

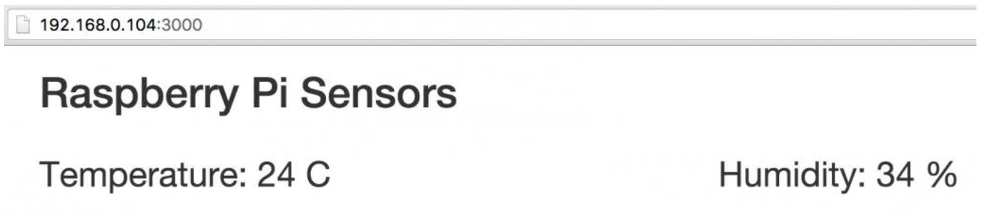
Be patient, this step can take a while. After that, type:

```
sudo node app.js
```

This will start the application on your Raspberry Pi. Then, navigate to the URL of your Raspberry Pi on port 3000, for example:

http://192.168.0.104:3000

You should immediately see the interface showing the temperature & humidity readings made by the ESP8266 board:


192.168.0.104:3000

Raspberry Pi Sensors

Temperature: 24 C

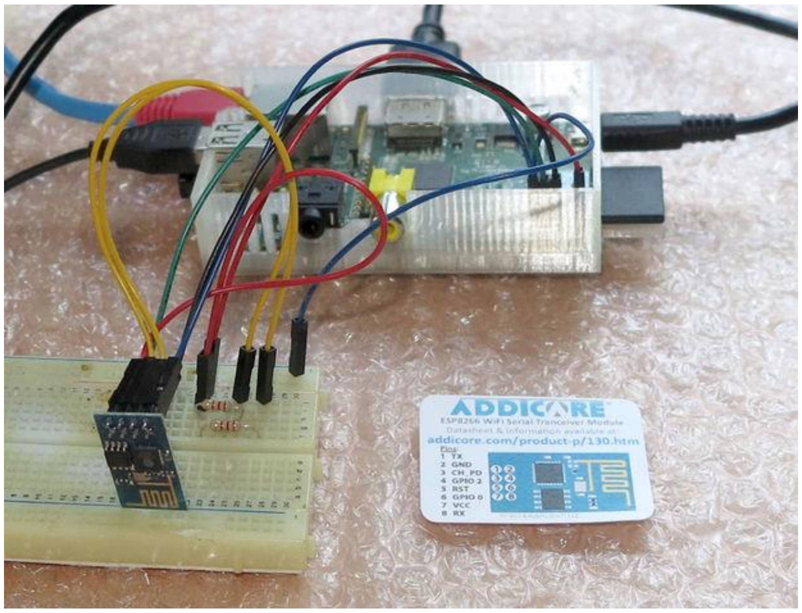
Humidity: 34 %

How to Go Further

Let’s summarise what we achieved in this project. We built a WiFi measurement station based on the ESP8266 WiFi chip. We used a sensor to measure the local temperature & humidity. Then, we displayed all the measurements on a web page that was served by the Raspberry Pi.

There are many things you can do to improve this project. You can for example deploy more of those sensors boards in your home, and display all the measurements on a single page on your Pi.

Connect an ESP8266 to your RaspberryPi


ADDICORE
ESP8266 WiFi Serial Tranceiver Module
addicore.com/product-p/130.htm
Pins:
1 TX
2 GND
3 CH_PD
4 GPIO 2
5 RST
6 GPIO 0
7 VCC
8 RX

ESP8266 boards are pretty neat, but if you just bought one (And why wouldn't you for only $5?) and have realized that you don't have any obvious means (3.3V TTL USB serial device) to communicate with it, you can talk to it directly with a Raspberry Pi. Both use 3.3V signaling, so no level converting is required.

You will need:

Raspberry Pi desktop setup (just about any Pi should do, but with power, keyboard, screen, etc. network access is a plus)

ESP8266-01 (or equivalent)

Some jumper wire and nipper/strippers

Solderless Breadboard (or You can solder this instead)

2x Pull-up/down resistors

(optional) push-button switch

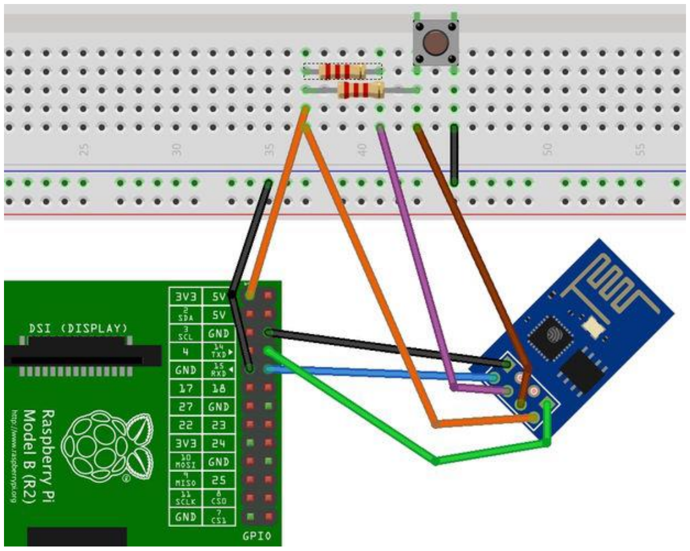
The process is roughly two steps,

1) Wire them together

2) Configure Raspbian

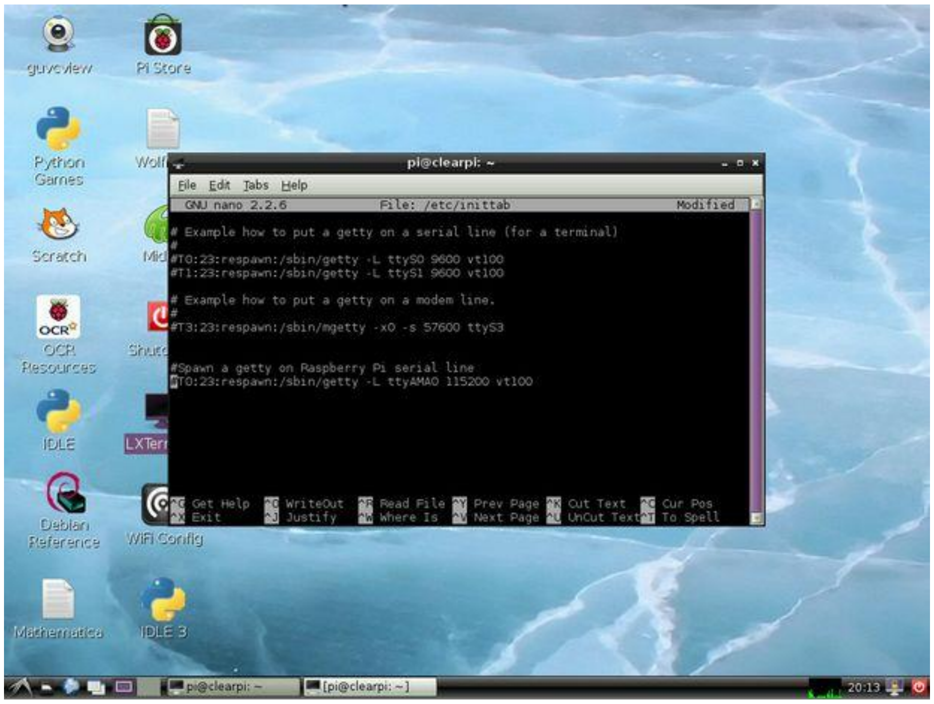
3) Start talking to your ESP8266

Step 1: Wire them together



With the power off, connect the 3.3V and ground pins to one another. Similarly connect the RX to TX and vice versa. You will also want some pull down resistors and optionally a reset button. I've included a Fritzing diagram (above).

Step 2: Configure Raspbian Linux



```
  GNU nano 2.2.6            File: /etc/inittab                        Modified

# Example how to put a getty on a serial line (for a terminal)
#
#T0:23:respawn:/sbin/getty -L ttyS0 9600 vt100
#T1:23:respawn:/sbin/getty -L ttyS1 9600 vt100

# Example how to put a getty on a modem line.
#
#T3:23:respawn:/sbin/mgetty -x0 -s 57600 ttyS3


#Spawn a getty on Raspberry Pi serial line
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100






^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

Boot up the Pi and with super user privs make some edits (e.g. “sudo nano”).

Disable the Kernel serial console

Edit /boot/cmdline.txt to **remove** the underlined text:

dwc_otg.lpm_enable=0 rpitestmode=1 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 rootwait

Disable serial logins

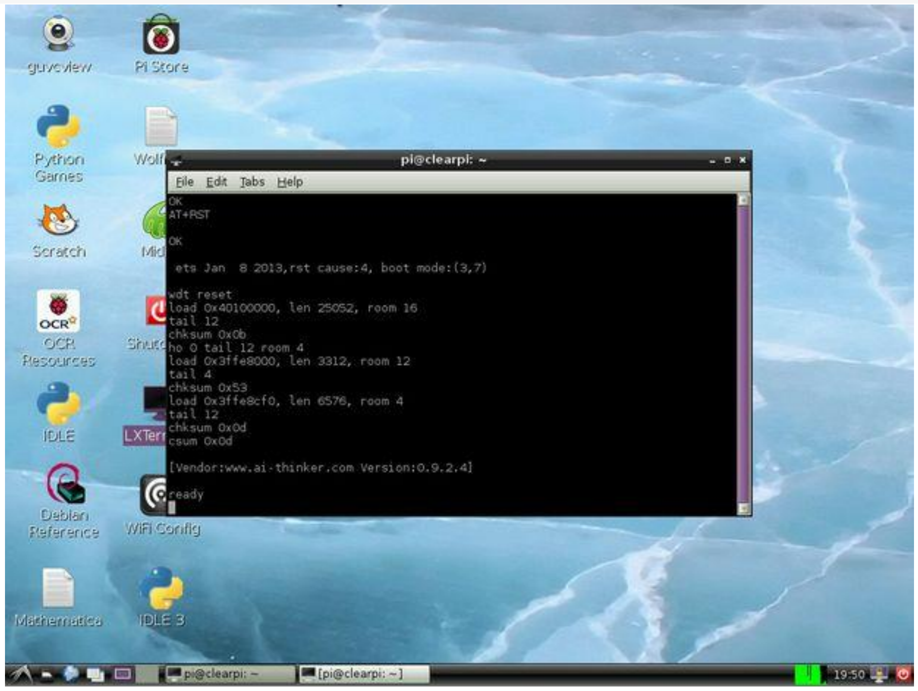
Edit /etc/inittab and **remove** the (usually) last line:

2:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

Reboot

sudo shutdown -r now

Step 3: Start talking to your ESP8266



```
OK
AT+RST

OK

 ets Jan  8 2013,rst cause:4, boot mode:(3,7)

wdt reset
load 0x40100000, len 25052, room 16
tail 12
chksum 0x0b
ho 0 tail 12 room 4
load 0x3ffe8000, len 3312, room 12
tail 4
chksum 0x53
load 0x3ffe8cf0, len 6576, room 4
tail 12
chksum 0x0d
csum 0x0d

[Vendor:www.ai-thinker.com Version:0.9.2.4]

ready
```

Connect to the serial port

You should now be able to interact with the ESP8266 via a terminal emulator program. I suggest GNU Screen for this ("sudo apt-get install screen"). Screen passes through your keystrokes unless you start with "CTRL-A" in which case you can then make screen do things. For example "CTRL-A" then "k" will close (kill) the session. screen /dev/ttyAMA0 115200 (the speed may vary depending on your ESP8266 board) (you may need to sudo this too, depending permissions) You can now send your ESP8266 "AT" commands. You'll still need "return" and "CTRL-J" after each one, though.

Notes/Caveats

The Pi does not seem to recover well if you try to use the reset button. Best to shutdown the pi, then remove power, then power/boot back up rather than try the reset button.

Don't mess with the wires while power is on. Shutdown the Pi ("sudo shutdown -h now") and wait for any blinking LEDs to stop and disconnect power before trying to connect/disconnect wires.

This linking won't by itself give your Pi internet access via the ESP8266. But it does at least let you try out, configure, maybe even reprogram it.

Screen won't relinquish the serial port if you just close a window or disconnect. This is on purpose, but can take some getting used to. If you think you disconnected from Screen but it is still running try "screen -ddR" to reconnect, and then you can to the kill as above. [Or reboot.]

The Pi and the ESP8266 can use a lot of power. This setup relies on the Pi's 3.3V supply alone, which may not be up to the task if both devices are going to be going full blast. A more reliable/permanent solution would be to rig up separate power for the ESP8266.

Getting Started With the ESP8266 Chip

The ESP866 is a tiny WiFi chip that has the huge advantage to come at a ridiculous price point: $5. And the best is that this chip also has a small processor onboard, so it can actually function in complete autonomy, without an additional Arduino board for example. Compare that to the cost of a traditional Arduino + WiFi module solution (around $40), and you will immediately see why this chip is receiving so much interest these days.

All of this makes it the perfect piece of hardware for connected home automation projects. In this article, which is the first one in a series of articles dedicated to the ESP8266, we'll stick to the basics. We will see how to choose an ESP8266 module, what other components you need, how to configure the chip, and finally how to connect it to your WiFi network. To program the chip, we'll simply be using the well-known Arduino IDE. Let's start!

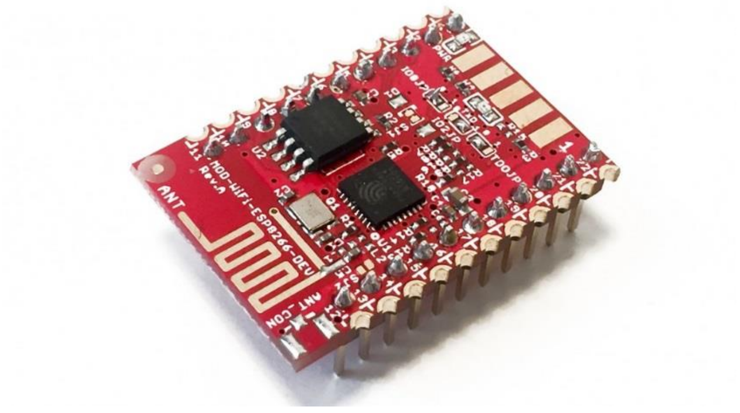
How to Choose Your ESP8266 Module

We are first going to see how to choose the right ESP8266 module for your project. Indeed, there are many on the market, and it's quite easy to get lost between all the choices.

The first one that you probably heard of is the small ESP8266 Serial Wireless Transceiver module:

This module is the most famous one, as it is really small and only costs $5. However, the number of accessible GPIO pins (outputs or inputs pins) are quite limited. It is also difficult to plug it on a breadboard.

But there are many other modules on the market, that gives you access to all the pins of the ESP8266. For example, I really like the ESP8266 Olimex module which is also cheap (around $10):

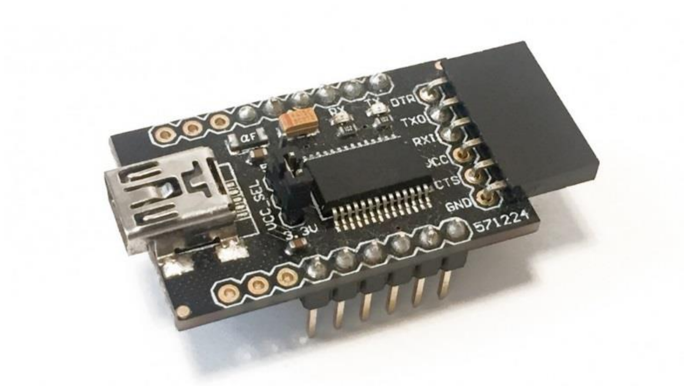


This module can easily be mounted on a breadboard, and you can easily access all the pins of the ESP8266. This is the one I will use for the rest of this article, but you can perfectly follow along with any ESP8266 module.

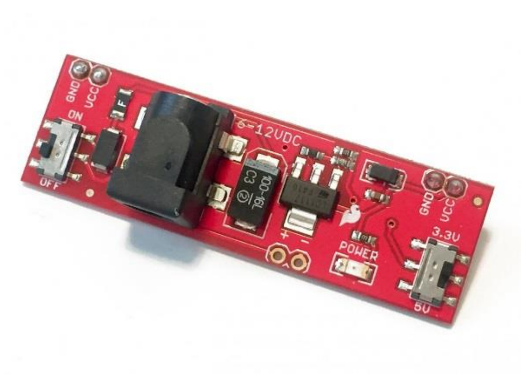
Hardware Requirements

Let's now see what we need to make the ESP8266 chip work. Indeed, it is usually wrongly assumed that you just need this little chip and nothing else to make it work, and we are going to see that it is not true.

First, you will need some way to program the ESP8266. You can use an Arduino board for that, but for me the really great thing about the ESP8266 is that it can function completely autonomously. So for to program the chip, I will use a USB FTDI programmer. Note that it has to be compatible with the logic level of the ESP8266 chip, so 3.3V. I used a module that can be switched between 3.3V and 5V:



You will also need a dedicated power supply to power the chip. This is a point that is often forgotten and that leads to a lot of issues. Indeed, if you are for example trying to power the ESP8266 chip from the 3.3V coming from the FTDI board or from an Arduino board, it simply won't work correctly. Therefore, you need a dedicated power supply that can deliver at least 300 mA to be safe. I used this breadboard power supply that can deliver up to 500 mA at 3.3V:

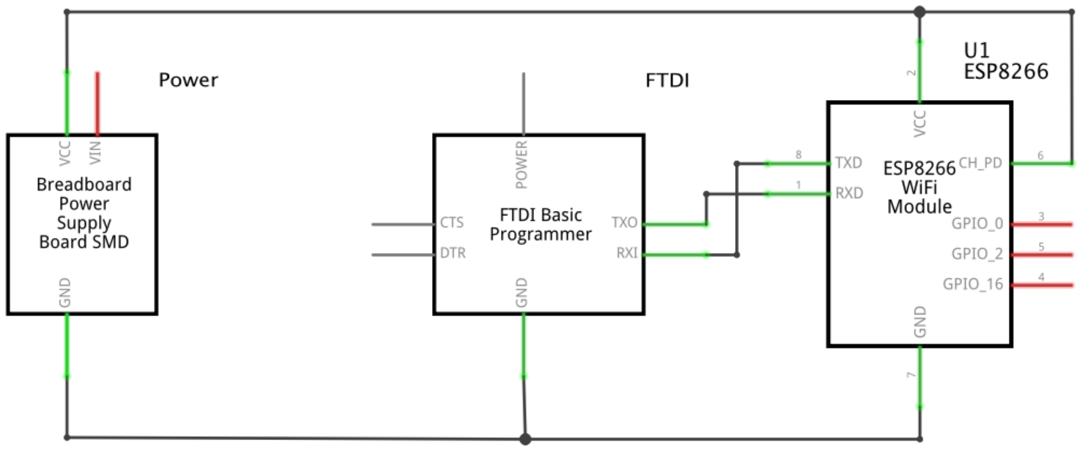


This is a list of all the components that will be used in this guide:

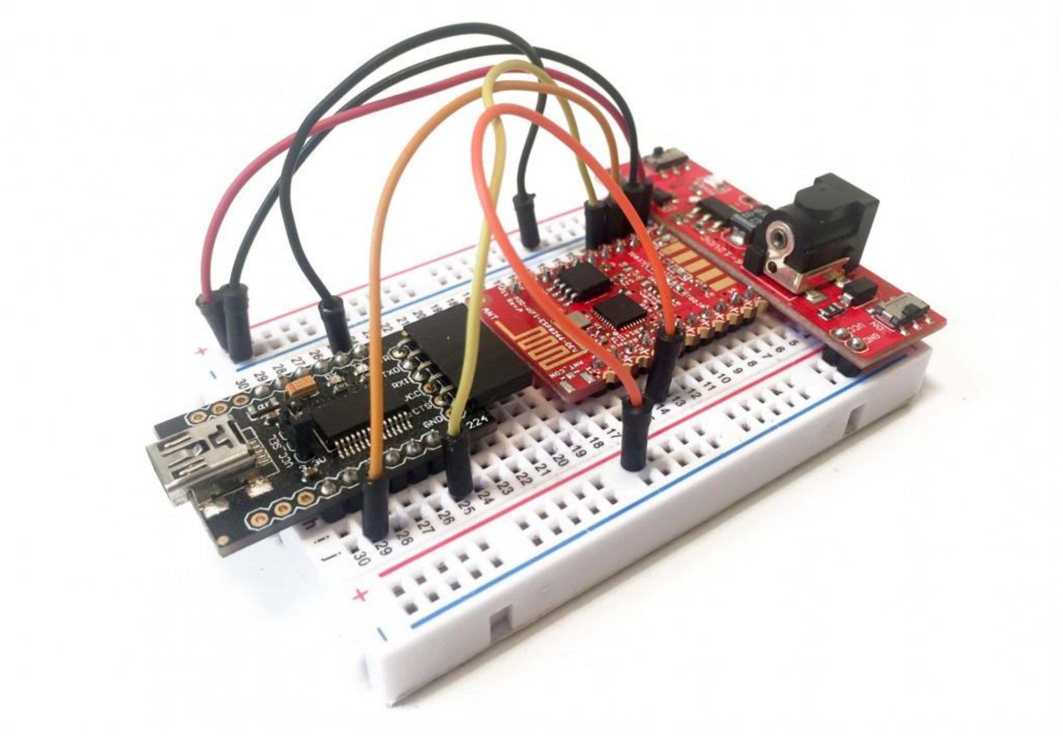
- ESP8266 Olimex module
- Breadboard 3.3V power supply
- 3.3V FTDI USB module
- Breadboard
- Jumper wires

Hardware Configuration

We are now going to see how to configure the hardware for the first use of your ESP8266 board. This is how to connect the different components:



And this is how it will look like at the end:



Make sure that you connected everything according to the schematics, or you won't be able to continue. Also make sure that all the switches of your components (FTDI module & power supply) are set to 3.3V, or it will damage your chip.

Also, connect one wire to the pin 0 (GPIO 0) of the ESP8266. Don't connect it to anything else for now, but you will need it later to put the chip in programming mode.

Configuring Your ESP8266 Chip

We are now going to configure your ESP8266 chip using the Arduino IDE. This is a great way to use the chip as you will be able to program it using the well-known Arduino IDE, and also re-use several existing Arduino libraries.

If this is not done yet, install the latest version of the Arduino IDE. You can get it from:

http://www.arduino.cc/en/main/software

Then, you need to take a few steps to be able to configure the ESP8266 with the Arduino IDE:

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

Now, we are going to check that the Arduino IDE is correctly working, and connect your chip to your local WiFi network.

To do so, we need to write the code first, and then upload it to the board. The code is will be quite simple: we just want to connect to the local WiFi network, and print the IP address of the board. This is the code to connect to the network:

```
// Import required libraries
#include "ESP8266WiFi.h"

// WiFi parameters
const char* ssid = "your_wifi_name";
const char* password = "your_wifi_password";

void setup(void)
{
// Start Serial
Serial.begin(115200);

// Connect to WiFi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

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

}

```

```
void loop() {

}
```

You can simply copy the lines of code above, and copy them into the ESP8266 Arduino IDE that you downloaded before. Of course, put your own WiFi name & password in the code. Save this file with a name of your choice.

Now, also go in Tools>Boards, and select the board you are using. If you are not sure about which board to choose, simply choose "Generic ESP8266 Module". Also select the correct Serial port that corresponds to the FTDI converter your are using.

After that, we need to put the board in bootloader mode, so we can program it. To do so, connect the pin GPIO 0 to the ground, via the cable we plugged into GPIO 0 before. Then, power cycle the board but switching the power supply off & then on again.

Now, upload the code to the board, and open the Serial monitor when this is done. Also set the Serial monitor speed to 115200. Now, disconnect the cable between GPIO 0 and GND, and power cycle the board again. You should see the following message:

```
WiFi connected
192.168.1.103
```

If you can see this message and an IP, congratulations, your board is now connected to your WiFi network! You are now ready to build your first projects using the ESP8266 chip & this modified Arduino IDE.

How to Go Further

You now have a completely usable ESP8266 module, so basically what you can do next only depends on your imagination! You can for example use this chip to build a WiFi motion sensor, to control a relay remotely, and also to send data to a remote cloud platform.

I will post articles about applications of this board in the coming weeks, but in the meantime don't hesitate to experiment with this amazing little WiFi board and share your projects in the comments!

ESP8266 WiFi Module for Dummies

Overview of the ESP8266 WiFi Module

The ESP8266 is a really useful, cheap WiFi module for controlling devices over the Internet. It can work with a micro-controller like the Arduino or it can be programmed to work on its own.

The Internet of Things (IoT) has just been made a whole lot cheaper and easier!

The ESP8266 comes with factory installed firmware allowing you to control it with standard "AT commands". You can also easily create and upload your own code and this makes it hugely powerful and flexible.

The ESP8266 was launched in 2014 and is rapidly growing in popularity. There is only one ESP8266 processor but it is found on many different breakout boards. These all differ in terms of which pins are exposed and the size of the flash memory etc. These breakout boards have evolved rapidly over the years and there is a lot of information to be found on the web. This is both a blessing and a curse as some of the advice is outdated or just plain wrong.

As a result it took me 3 days before I could get it to do anything! Some people are lucky but for me it was a battle to get anything working. At times I thought I had broken it but with some perseverance it always came back to life!

I've finally got it singing and dancing and it turns out to be easier than I thought!

I've consolidated all the useful info I found into one place and I thought I'd share this in an instructable. This will hopefully get you quickly up and running with all the basic functionality that you'll need.

In this instructable were are looking at the ESP-01 breakout board.

This same approach should however apply to most (if not all) of the variants of the ESP out there.

Step 1: The ESP-01 in a nutshell

1. It supports the 802.11 b/g/n protocol
2. It can connect to your router and work as a client or it can be an access point itself or both!
3. It is IP addressable and can be a Web Server
4. The "standard" version has 2 digital pins that can be used for input or output. Eg: to drive LED's or relays. These pins can also be used for PWM. Other versions have more pins exposed.
5. Analog input is also available on the ESP8266 chip (ADC/TOUT) but it's not wired up on the ESP-01.
6. It can be combined with an Arduino or it can be programmed to work on its own
7. There are various tools and development environments (IDE's) to program it.
8. And it has lots more….

In this demo the ESP8266 acts as a Webserver hosting a Web Page that you can interact with over WiFi.

With this demo you can start your own home automation project and control it from the web!

Step 2: What you'll need

Firstly let's see if we can get it up and running.

A lot of sites recommend re-flashing the firmware and using various tools. You don't need to do that just yet.

We'll be replacing the firmware with our own code. In another instructable: The Simple Guide to Flashing Your ESP8266 Firmware I show you how to revert it back to an updated factory version. If you'll only be programming the ESP8266 with your own code - as we will be doing in this demo - then you don't need factory firmware. Your code is the firmware.

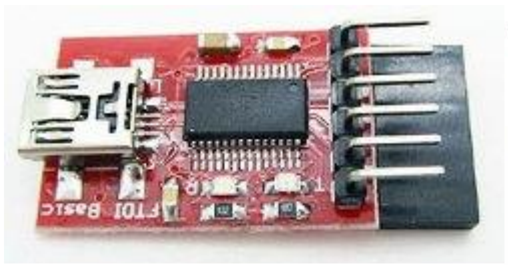
We'll be using the Arduino IDE as a programming tool but we don't need an Arduino itself. The ESP8266 has all the functionality we need.

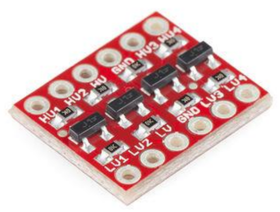
NB: It's a 3.3V module and not 5V. The ESP-01 is more robust than I thought but you should set it up properly, especially if you're using it with a 5V USB to Serial Programmer or micro-controller, otherwise you may fry it!

What you'll need:

1. An ESP8266 of course (use the ESP-01 variant or adapt your wiring accordingly)
2. Breadboard
3. 3.3V USB to Serial Programmer like the FTDI Basic (sometimes known as a UART: Universal Asynchronous Receiver/Transmitter). Some come with a jumper to switch between 5V and 3.3V.
4. Breadboard Jumper Wires
5. A separate 3.3V Power Supply. You **do** need one as the ESP8266 can chew a lot of mA (>250mA at times). The 3.3V output from the Serial Programmer or the Arduino **is not** sufficient. Without a separate power supply your ESP may be erratic and you will waste time trying to upload code.
6. Led, push button, 330 ohm resistor, 10k ohm resistor, .1μF capacitor across your power supply (as standard practice).
7. Add a Logic Level Converter into the circuit if you're using a micro-controller or FTDI programmer at 5V

Step 3: The components in more detail





ESP-01

I got my ESP-01 from: http://netram.co.za/2720-wifi-serial-transceiver-module-esp8266.html

You can have a look at the various versions here:http://www.esp8266.com/wiki/doku.php?id=esp8266-module-family

Breadboard Power Supply

A breadboard power supply is really useful, especially when you're dealing with mixed voltages. The one shown here supplies 3.3V and 5.5V and plugs right into your breadboard. I should have got one a long time ago! The ESP8266 can chew a lot of mA (>500mA at times). If your ESP reboots or seems unreliable then you need a separate power supply for the ESP (keep a common ground though).

http://netram.co.za/938-breadboard-power-supply-5v33v.html

USB to Serial programmer

Get one that supports 3.3V or has a 3.3V/5V jumper otherwise if you only have 5V then you should use a bi-directional Logic Level Converter in between the programmer and the ESP8266. The FTDI Basic has a 3.3V/5V jumper on the back:

http://netram.co.za/2842-ftdi-basic-usb-to-ttl-programmer.html

http://netram.co.za/2679-logic-level-converter-bi-directional.html

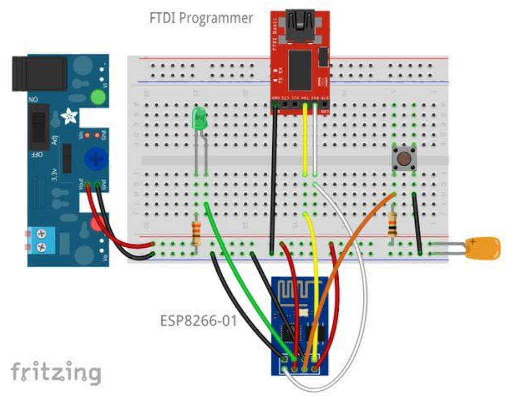
Logic Level Converter

Bi-Directional 5V & 3.3V. This is only required if you are using a 5V serial programmer.

Breadboard Jumpers

Include some JST jumpers as they're handy for plugging onto modules with pins and then joining them to the breadboard. Include some male to female cables for this purpose. The ESP-01 isn't breadboard friendly so you'll need these.

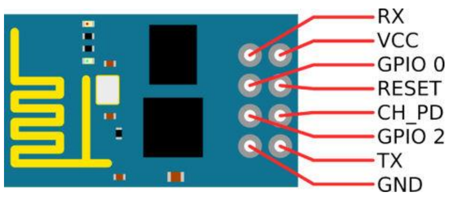
Step 4: The Breadboard Layout



Looking at the breadboard, the FTDI programmer is the red board on top and the ESP8266-01 at the bottom. The FTDI programmer only needs 3 wires connected: the ground wire and the RX to TX and TX to RX on the ESP. On the left is the breadboard power supply (it would normally plug into the breadboard).

NB: The quality of the power supply to the ESP is one of the most important factors. There is a small capacitor (.1uF) across 3.3V and Ground to minimise spikes. If you are having problems with programming your ESP it may well be your power supply. What will help is to add another larger capacitor across 3.3V and Ground. eg: 10uF.

Step 5: The ESP-01 Pinouts



The picture above of the ESP8266 pinouts is found all over the net. I think it may have originated from Neil "Kolban's Book on ESP8266". This is a fantastic resource and is available as a pdf download from:https://leanpub.com/ESP8266_ESP32. It's 400 pages long, was updated in January 2016, and if you're into serious ESP8266 development then this is the book to get. It's free but make a donation if you can.

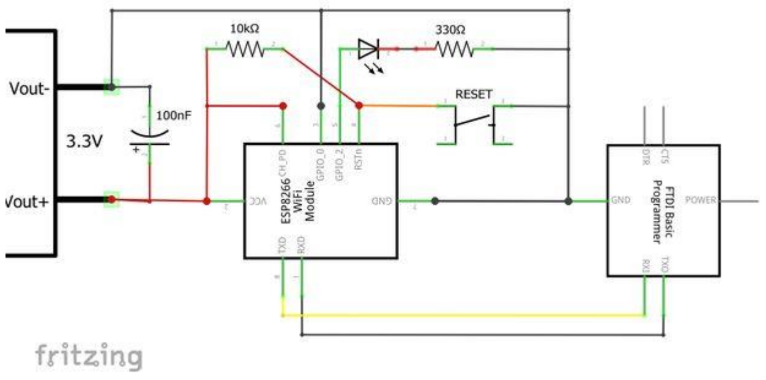
Breadboard wiring notes:

- The wires are shown connecting to the front of the ESP-01 for clarity. Attach them to the same pins from behind. The ESP-01 is not breadboard friendly and JST connector male-female jumpers are handy for connecting it.
- GPIO 0 must be held low for programming. This should be removed if you remove the FTDI and operate the ESP8266 on its own power supply.
- CH_PD should be held high at all times.
- The RESET pin is held high with a 10k ohm resistor and brought to ground by the RESET button which will reboot the chip. Press RESET every time before you upload code, and every time you connect or disconnect GPIO 0. The reset button will save you a lot of hassle.

When you power the circuit up the red LED on the ESP-01 should light up and the blue LED should flash briefly. Later when uploading code to the ESP the blue LED should flash continuously during the process. Any other LED combinations suggests that there is a problem.

NB: If you want to remove the Serial programmer and run the ESP on its own, make sure you disconnect GPIO 0 from ground and push the RESET button otherwise the ESP may not work properly.

Step 6: Schematic



Step 7: Using the Arduino instead of a Serial Programmer

You can also use an Arduino instead of your USB to Serial programmer.

A few things to note:

1. On a 5V Arduino the digital output will be at 5V and this could fry your ESP2866. So either use a 3.3V Arduino or a Logic Converter chip
2. The 3.3V Vout on the Arduino is insufficient to drive your ESP8266 and you could get unreliable results. Preferably use another 3.3V supply.
3. As a recommended practice always upload a blank sketch to your Arduino before connecting any other devices to avoid any unpredictable results. A blank sketch has empty setup() and loop() functions.
4. If you are using an Arduino Uno it only has one hardware TX&RX pair. To communicate with both the ESP8266 and the Serial Monitor at the same time you will also need to use the SoftwareSerial include file. It used to have a limitation on the baud rate but can now theoretically support a baud rate of up to 115200. A Mega 2560 has more hardware TX and RX pins so this could be a better way to go.

5. And **finally and importantly**: Look closely at the TX and RX pins on the Arduino. You will notice that they have arrows next to them. The TX pin has an arrow pointing away from it. What this says is TX IN! What this really means is that **the TX pin is actually the RX pin!** And likewise **the RX pin is actually the TX pin!** So when using the Arduino as a serial adapter with another device you must connect RX to RX and TX to TX. I only discovered this nuance after several hours of pain! Why-oh-why did they label it like this?

Whilst I'm asking questions: **why** the ESP-01 was produced with so few of the useful pinouts exposed also baffles me! It is still a great starting point but if you need something more then go for the ESP-12.

Step 8: Programming using the Arduino IDE

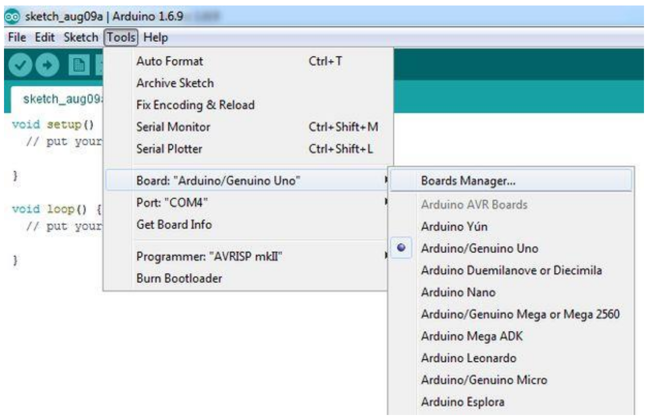
There are various tools to program the ESP8266 but the Arduino IDE (Integrated Development Environment) is by far the easiest for most folk.

To program the ESP8266 with the Arduino IDE you'll need to download the latest version if you don't already have it and then install the ESP8266 support. Note that you are only using the Arduino programming environment you don't need any Arduino hardware itself.

To download the Arduino IDE go to: https://www.arduino.cc/en/Main/Software

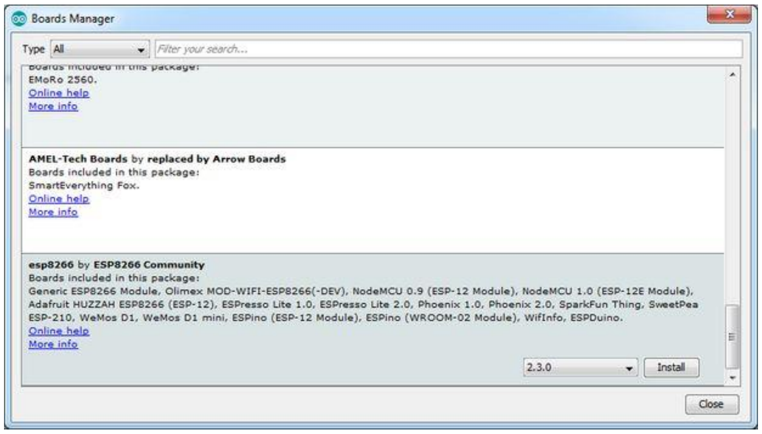
Start up the Arduino IDE and select the Boards Manager…

Step 9: Setup the ESP2866 Board Manager



Start the Arduino IDE and select the Boards Manager under Tools / Board ..

Step 10: Install the ESP2866 Board Manager



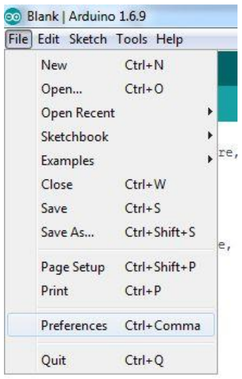
Boards included in this package:
EMoRo 2560.
Online help
More info

AMEL-Tech Boards by **replaced by Arrow Boards**
Boards included in this package:
SmartEverything Fox.
Online help
More info

esp8266 by **ESP8266 Community**
Boards included in this package:
Generic ESP8266 Module, Olimex MOD-WIFI-ESP8266(-DEV), NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Adafruit HUZZAH ESP8266 (ESP-12), ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, SparkFun Thing, SweetPea ESP-210, WeMos D1, WeMos D1 mini, ESPino (ESP-12 Module), ESPino (WROOM-02 Module), WifInfo, ESPDuino.
Online help
More info

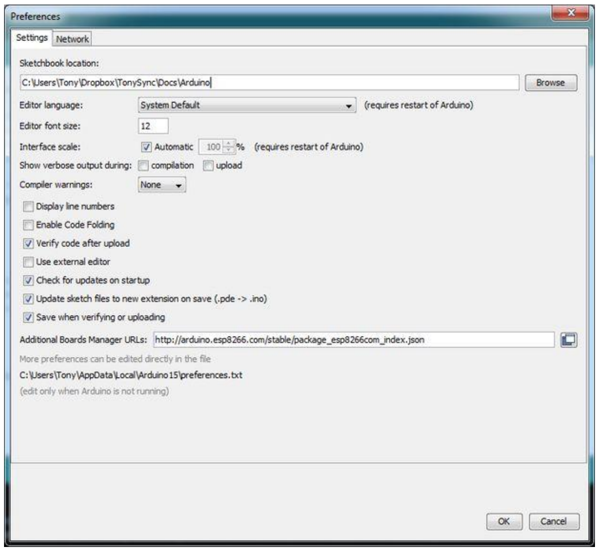
Find the esp8266, select it and click Install.

Step 11: Arduino Preference Setup



Go to File / Preferences

Step 12: Boards Manager URL's



Preferences

Settings | Network

Sketchbook location:
C:\Users\Tony\Dropbox\TonySync\Docs\Arduino — Browse

Editor language: System Default (requires restart of Arduino)
Editor font size: 12
Interface scale: [✓] Automatic 100 % (requires restart of Arduino)
Show verbose output during: [ ] compilation [ ] upload
Compiler warnings: None

- [ ] Display line numbers
- [ ] Enable Code Folding
- [✓] Verify code after upload
- [ ] Use external editor
- [✓] Check for updates on startup
- [✓] Update sketch files to new extension on save (.pde -> .ino)
- [✓] Save when verifying or uploading

Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

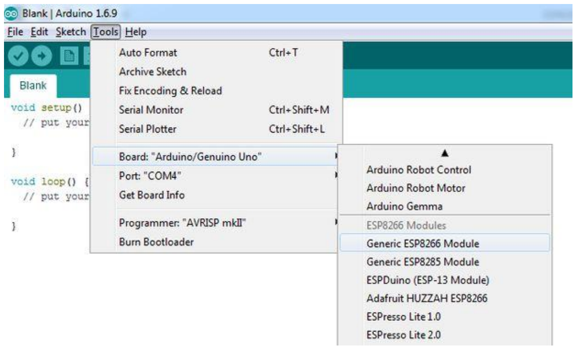
More preferences can be edited directly in the file
C:\Users\Tony\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

OK Cancel

In the Additional Boards Manager URLs field enter the URL for the ESP8266 package which is:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

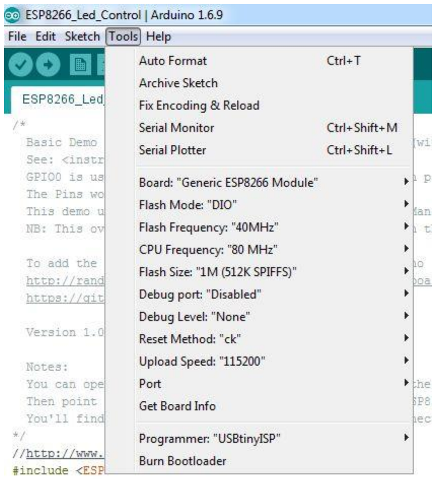
Step 13: ESP8266 Setup in IDE



```
void setup() {
  // put your
}

void loop() {
  // put your
}
```

Select the “Generic ESP8266 Module” and go back to the Boards Manager again

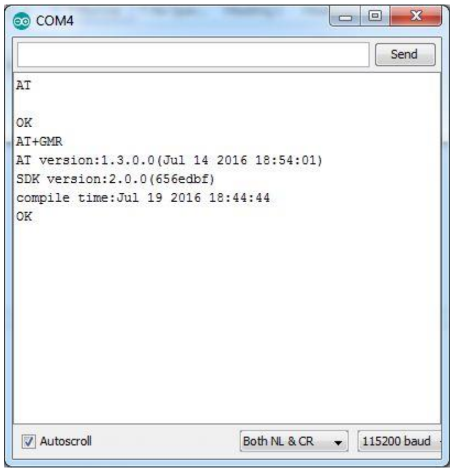
Step 14: ESP8266 Options


ESP8266_Led_Control | Arduino 1.6.9
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
Port
Get Board Info
Burn Bootloader

```
Board: "Generic ESP8266 Module"
Flash Mode: "DIO"
Flash Frequency: "40MHz"
CPU Frequency: "80 MHz"
Flash Size: "1M (512K SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
Reset Method: "ck"
Upload Speed: "115200"
Programmer: "USBtinyISP"
```

Setup you ESP-01 options. Select the appropriate COM port according to your PC.

Step 15: Talking to ESP8266



```
AT

OK
AT+GMR
AT version:1.3.0.0(Jul 14 2016 18:54:01)
SDK version:2.0.0(656edbf)
compile time:Jul 19 2016 18:44:44
OK
```

If you still have the original firmware on your ESP8266 you can try talking to it with the Serial Monitor.

1. Open the Arduino Serial Monitor:
2. Select the appropriate baud rate which is usually 115200 but could be anything down to 9600 depending on the firmware on the ESP.
3. Also select “Both NL & CR”. Ie: New Line and Carriage Return characters after each “Send”.
4. Type AT and press enter. The ESP should come back with OK. If not press the RESET button or try unplugging the USB cable and starting again. Make sure also that you have the right COM port selected. Also if you don’t have the “standard” firmware on your ESP it may not understand the AT command set. This is not necessarily a problem yet as you’ll be overwriting this with Arduino code later in this instructable.
5. Type AT+GMR. This should come back with the version numbers of the firmware on your ESP. In another instructable I show you how you can restore the latest version of the “factory” firmware (if you want to).

Step 16: ESP8266 Arduino Code

Download code file (ESP8266_LED_Control.zip), unzip it and open it in the Arduino IDE.

Change the Router setting to yours. ie: change the SSID and PASSWORD fields.

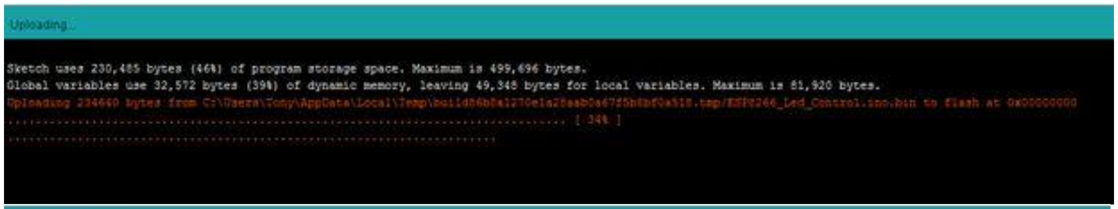
Make sure your breadboard is setup correctly, and connect your USB cable to the Serial Programmer.

- **ESP8266_LED_Control.zip**

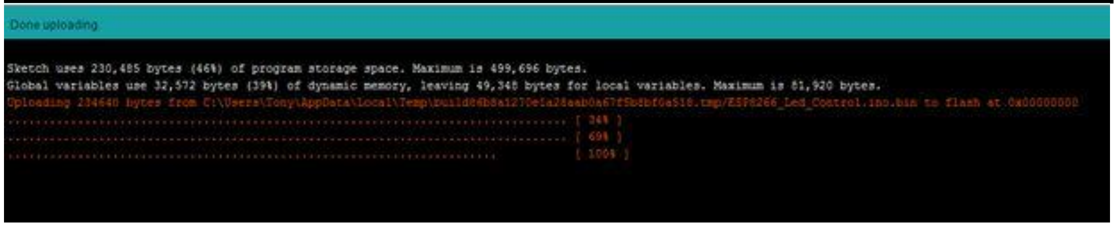
```
/*
  This is a demo of ESP8266 WiFi Module in standalone mode (without Arduino) controlling an LED via a Web page.
  This code also demostrates that PWM is available on the ESP8266, so we can dim the LED.
  Analog Input is available on the ESP8266 but this pin is not wired up on ESP-01 board.
  The ESP can also support interrupts.

  This demo uses the Arduino IDE and the Arduino Boards Manager to upload the program to the ESP2866 with an FTDI
Programmer.
  NB: This overwrites any Firmware that was previously on the ESP8266.

  See: http://www.instructables.com/id/ESP8266-WiFi-Module-for-Dummies/ for the details on how to set the board up and
upload the code to the ESP8266
  Also see: http://www.instructables.com/id/The-Simple-Guide-to-Flashing-Your-ESP8266-Firmware/
*/

// Include the ESP8266 Library. This library is automatically provided by the ESP8266 Board Manager and does not need to be
installed manually.
#include <ESP8266WiFi.h>

String codeVersion = "Version 1.0  Aug 2016 by TonesB";

// WiFi Router Login - change these to your router settings
const char* SSID = "YourSSID";
const char* password = "YourPassword";

// Setup GPIO2
int pinGPIO2 = 2; //To control LED
int ledStatus = 0; //0=off,1=on,2=dimmed

// Create the ESP Web Server on port 80
WiFiServer WebServer(80);
// Web Client
WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(10);
  Serial.println();
  Serial.println();
  Serial.println(codeVersion);

  // Setup the GPIO2 LED Pin - this demo also uses PWM to dim the LED.
  pinMode(pinGPIO2, OUTPUT);
  digitalWrite(pinGPIO2, LOW);
  ledStatus = 0;

  // Connect to WiFi network
  Serial.println();
  WiFi.disconnect();
  WiFi.mode(WIFI_STA);
  Serial.print("Connecting to ");
  Serial.println(SSID);
  WiFi.begin(SSID, password);

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

  // Start the Web Server
  WebServer.begin();
  Serial.println("Web Server started");

  // Print the IP address
  Serial.print("You can connect to the ESP8266 at this URL: ");
  Serial.print("http://");
  Serial.print(WiFi.localIP());
  Serial.println("/");
}

void loop() {
  // Check if a user has connected
  client = WebServer.available();
  if (!client) {//restart loop
    return;
  }

  // Wait until the user sends some data
  Serial.println("New User");
```

```
while (!client.available()) {
  delay(1);
}

// Read the first line of the request
String request = client.readStringUntil('\r\n');
Serial.println(request);
client.flush();

// Process the request:
if (request.indexOf("/LED=ON") != -1) {
  analogWrite(pinGPIO2, 1023);
  ledStatus = 1;
}
if (request.indexOf("/LED=OFF") != -1) {
  analogWrite(pinGPIO2, 0);
  ledStatus = 0;
}
if (request.indexOf("/LED=DIM") != -1) {
  analogWrite(pinGPIO2, 512);
  ledStatus = 2;
}

// Return the response
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html; charset=UTF-8");
client.println("");
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("<head>");
client.println("<title>ESP8266 Demo</title>");
client.println("</head>");
client.println("<body>");
client.println("<a href=\"/\">Refresh Status</a>");
client.println("</br></br>");

//check the LED status
if (ledStatus == 0) {
  client.print("LED is Off</br>");
  client.println("Turn the LED <a href=\"/LED=ON\">ON</a></br>");
  client.println("Set LED to <a href=\"/LED=DIM\">DIM</a></br>");
} else if (ledStatus == 1) {
  client.print("LED is On</br>");
  client.println("Turn the LED <a href=\"/LED=OFF\">OFF</a></br>");
  client.println("Set LED to <a href=\"/LED=DIM\">DIM</a></br>");
} else if (ledStatus == 2) {
   client.print("LED is Dimmed</br>");
   client.println("Turn the LED <a href=\"/LED=OFF\">OFF</a></br>");
   client.println("Turn the LED <a href=\"/LED=ON\">ON</a></br>");
 }

 client.println("</br>");
 client.println("<a href=\"http://www.instructables.com/id/ESP8266-WiFi-Module-for-Dummies/\" target=\"_blank\">See the Instructables Page: ESP8266 WiFi Module for Dummies</a></br>");
 client.println("<a href=\"http://www.instructables.com/id/The-Simple-Guide-to-Flashing-Your-ESP8266-Firmware/\" target=\"_blank\">See the Instructables Page: The Simple Guide to Flashing Your ESP8266 Firmware</a></br>");

 client.println("</br>");
 client.println("</body>");
 client.println("</html>");

 delay(1);
 Serial.println("User disconnected");
 Serial.println("");

}
```

Step 17: Uploading your code to the ESP8266



```
Sketch uses 230,485 bytes (46%) of program storage space. Maximum is 499,696 bytes.
Global variables use 32,572 bytes (39%) of dynamic memory, leaving 49,348 bytes for local variables. Maximum is 81,920 bytes.
Uploading 234640 bytes from C:\Users\Tony\AppData\Local\Temp\build86b8a1270e1a28aab0a67f5b8bf0a518.tmp/ESP8266_Led_Control.ino.bin to flash at 0x00000000
................................................................................ [ 34% ]
......................................................................
```



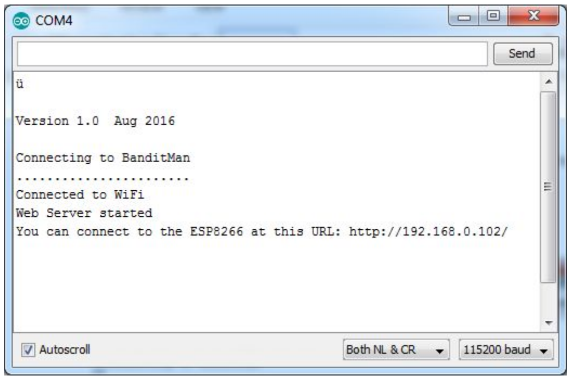
```
Sketch uses 230,485 bytes (46%) of program storage space. Maximum is 499,696 bytes.
Global variables use 32,572 bytes (39%) of dynamic memory, leaving 49,348 bytes for local variables. Maximum is 81,920 bytes.
Uploading 234640 bytes from C:\Users\Tony\AppData\Local\Temp\build86b8a1270e1a28aab0a67f5b8bf0a518.tmp/ESP8266_Led_Control.ino.bin to flash at 0x00000000
................................................................................ [ 34% ]
................................................................................ [ 69% ]
......................................................................           [ 100% ]
```

Make sure you've closed ALL Serial port monitors on the same port, if you you're using them. Eg: tools like CoolTerm. Open the Arduino Serial Monitor again, as you will need to see the IP address of the ESP after it loads.

NB: To upload your code you must have GPIO 0 grounded. You can leave it there whilst running your code in the browser. Press the RESET button before each load. If the upload fails, check your settings and try disconnecting and reconnecting the USB cable. Try Ctrl&U a couple of times. If all else fails try rebooting your PC. Sometimes the COM ports get a bit messed up.

The blue LED on the ESP should flash if the code is being uploaded successfully. The progress of the upload should look like the images above.

Step 18: Let see what it's doing



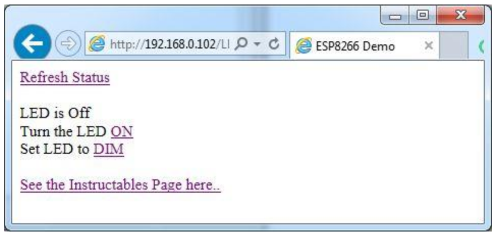
```
ü

Version 1.0  Aug 2016

Connecting to BanditMan
.......................
Connected to WiFi
Web Server started
You can connect to the ESP8266 at this URL: http://192.168.0.102/
```

Switch over to the Arduino Serial Monitor and you should see the IP Address of the ESP. You could also log into your Router and check what’s connected there. The device will show up in the list of connected IP’s with a name like: ESP_1A6C4A.

Whilst you are in your router you could always reserve the ESP's IP address and set up some Port Forwarding so that you can connect to the ESP from outside your network - like the office..

You should see something like the above. Use that IP address in your browser.

Step 19: Let's try it out



Refresh Status

LED is Off
Turn the LED ON
Set LED to DIM

See the Instructables Page here..

After clicking on a few links you’ll hopefully see the LED going on and off!

You also see that it can do PWM and dim the LED.

The ESP8266 can also handle Interrupts on the digital pins and also has analog input. The analog input pin though is not exposed on the ESP-01,

Internet Explorer works fine but for some reason FireFox seems very slow with the ESP. This may have something to do with the setup of the HTTP header.

Step 20: ESP Resources

"Kolban’s Book on ESP8266" updated in January 2016 is for serious enthusiasts.

It is 400 pages long and has an incredible amount of detail. You can get it for free but make a donation if you can.

https://leanpub.com/ESP8266_ESP32

ESP Sites

http://www.esp8266.com/wiki/doku.php?id=esp8266-module-family http://www.esp8266.com/

https://iotbytes.wordpress.com/esp8266-pinouts/

http://espressif.com/en/products/hardware/esp8266ex/resources

AT Commands

https://www.itead.cc/wiki/ESP8266_Serial_WIFI_Module#AT_Commands

https://espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf

Arduino Boards Manager

http://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide/

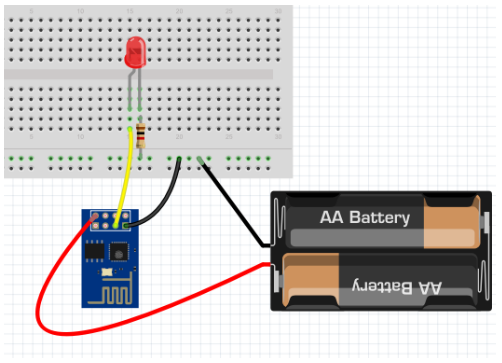
https://github.com/esp8266/Arduino

Notes on Flashing the Firmware:

I have published an instructable showing you how to flash the factory firmwareonto your ESP8266.

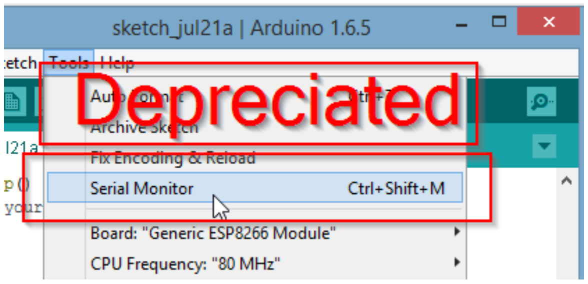
See: http://www.instructables.com/id/The-Simple-Guide-to-Flashing-Your-ESP8266-Firmware/

Blink (without support of Serial Monitor)



This will be the first example. Let's start with an easy one.
Let's blink a led.

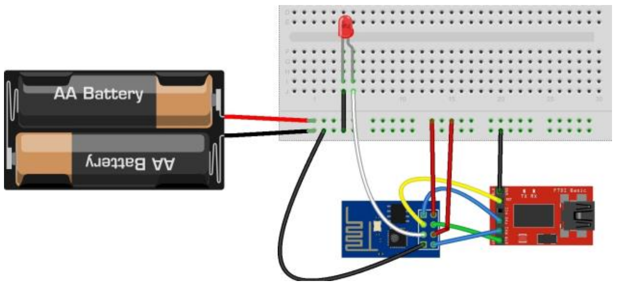
This example explains blink by using a USB-TTL connector that support DTR and Reset (RST).Use this blink_manual_flash example.



By using DTR, explained on this page, uploading will go smoothly. Please be informed that Serial monitor will not work anymore.
If you have 2 NPN transistors available. I recommend to use this example: blink

Shopping list
– 1 ESP8266 (for example version ESP01)
– 1 led (I use a red one)
– 1 USB to TTL (I use FT232RL)
– 10 wires
– 1 breadboard
– a 3.3V power source (I use an battery in this example, power from your USB TTL will most likely might NOT work)

Hardware



[table caption=”Wires” width=”500″ colwidth=”20|100|50″ colalign=”left|left|left|left|left”]
USB TTL, ESP8266 ESP-01
GND, GND
TX, RX
RX, TX
RTS, RST
DTR,GPIO0
,
+3.3V, CH_PD
battery -, GND
battery +, VCC
led-longleg, GPIO2
led-shortleg, GND

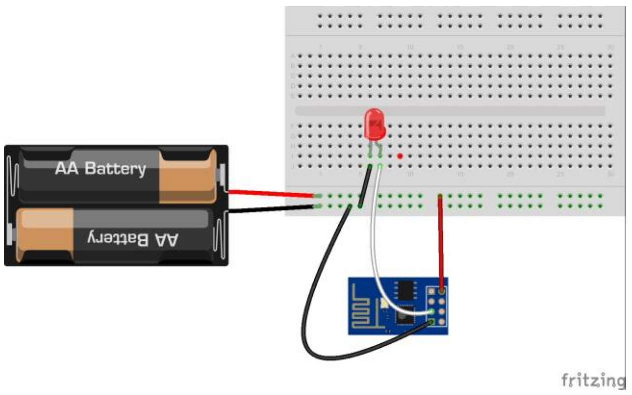
[/table]

Coding
Actually this is the same example as you may find in Files->Examples->01 Basics-> Blink.
ledPin = 2 means that we are using GPIO2.

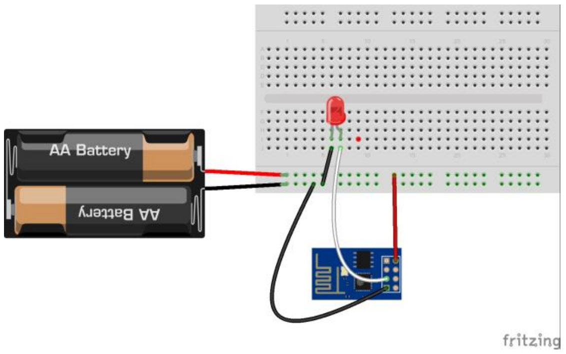
```
[cpp]
int ledPin = 2;
void setup() {
pinMode(ledPin, OUTPUT);
}

void loop() {
digitalWrite(ledPin, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(ledPin, LOW); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
}
[/cpp]
```

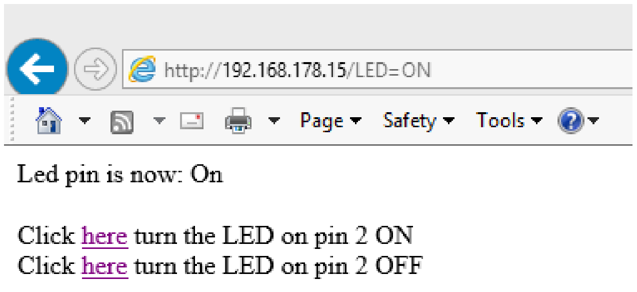
Uploading
Press upload in the Arduino IDE, arduino will compile and upload. After the sketch is uploaded, the led will blink for one second on and off.

Blinking
When you have uploaded the sketch, you may remove all the wires required for uploading.
This is what is left.



wifiwebserver



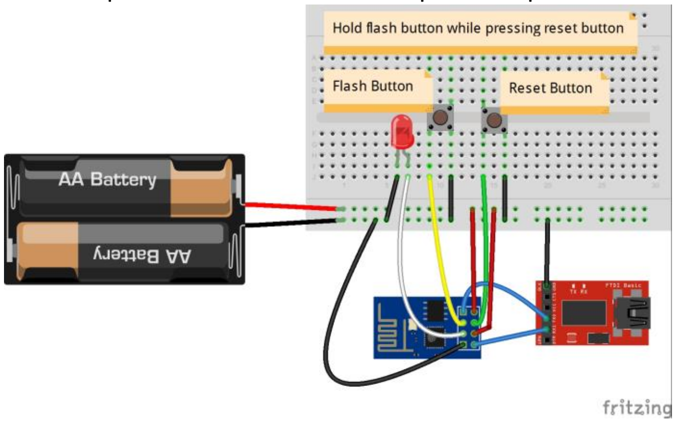


Led pin is now: On

Click here turn the LED on pin 2 ON
Click here turn the LED on pin 2 OFF

Shopping list
– 1 ESP8266 (for example version ESP01)
– 1 led (I use a red one)
– 1 USB to TTL (I use FT232RL)
– 2 push buttons
– 10 wires
– 1 breadboard
– a 3.3V power source (I use an battery in this example, power from your USB TTL will most likely might NOT work)

Hardware
The setup is the same as in the simple example blink



USB TTL ESP8266 ESP-01
—————————
GND — GND
TX — RX
RX — TX

————————
+3.3V — CH_PD
pushbutton — GPI0
resetbutton — RST
battery – — GND
battery + — VCC
led,longleg — GPIO2
led,shortleg — GND

Coding

Copy this code. Change ssid into your wifi accesspoint, and change the password into yours.
and compile.

ledPin = 2 means that we are using GPIO2.

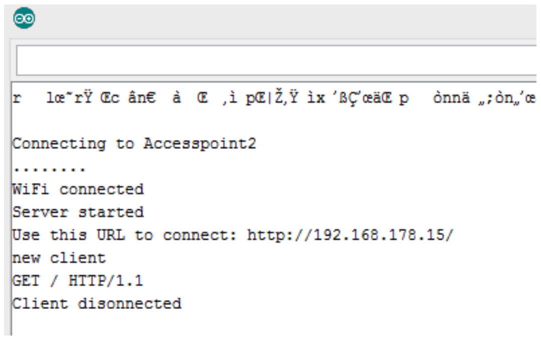
```
 [cpp]
#include <ESP8266WiFi.h>

const char* ssid = "your-ssid";
const char* password = "your-password";

int ledPin = 2; // GPIO2
WiFiServer server(80);

void setup() {
Serial.begin(115200);
delay(10);

pinMode(ledPin, OUTPUT);
digitalWrite(ledPin, LOW);

// Connect to WiFi network
Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);

WiFi.begin(ssid, password);

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

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

// Print the IP address
Serial.print("Use this URL to connect: ");
Serial.print("http://");
Serial.print(WiFi.localIP());
Serial.println("/");

}

void loop() {
// Check if a client has connected
WiFiClient client = server.available();
if (!client) {
```

```
return;
}

// Wait until the client sends some data
Serial.println("new client");
while(!client.available()){
delay(1);
}

// Read the first line of the request
String request = client.readStringUntil('\r');
Serial.println(request);
client.flush();

// Match the request

int value = LOW;
if (request.indexOf("/LED=ON") != -1) {
digitalWrite(ledPin, HIGH);
value = HIGH;
}
if (request.indexOf("/LED=OFF") != -1) {
digitalWrite(ledPin, LOW);
value = LOW;
}

// Set ledPin according to the request
//digitalWrite(ledPin, value);

// Return the response
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println(""); // do not forget this one
client.println("<!DOCTYPE HTML>");
client.println("<html>");

client.print("Led pin is now: ");

if(value == HIGH) {
client.print("On");
} else {
client.print("Off");
}
client.println("<br><br>");
client.println("Click <a
href=\"/LED=ON\">here</a> turn the LED on
pin 2 ON<br>");
client.println("Click <a
href=\"/LED=OFF\">here</a> turn the LED on
pin 2 OFF<br>");
client.println("</html>");

delay(1);
Serial.println("Client disonnected");
Serial.println("");

}
[/cpp]
```

Download source here: WifiWebServer

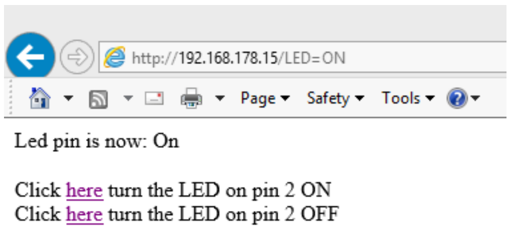
Uploading
Press reset button while holding flash button pressed. In other words. Press flash button, keep the flash button pushed while you once click on reset. You may now release the flash button. The ESP8266 is now in flash mode. You are able to upload the sketch.
Press upload in the Arduino IDE, arduino will compile and upload. After the sketch is uploaded, the led will blink for one second on and off.

Testing
Open the serial monitor from the menu tools. You will see the URL



```
r   lœˆrŸ Œc ân€  à  Œ  ,ì pŒ|Ž,Ÿ ìx ’ßÇ’œäŒ p   ònnä „;òn„’œ

Connecting to Accesspoint2
........
WiFi connected
Server started
Use this URL to connect: http://192.168.178.15/
new client
GET / HTTP/1.1
Client disonnected
```

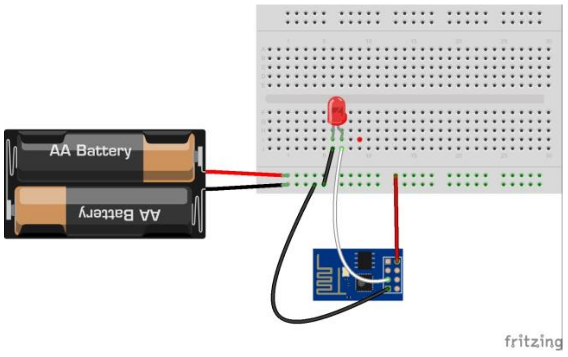
The strange characters on the first line after a reset is normal.
Copy the URL from the serial monitor into your browser.



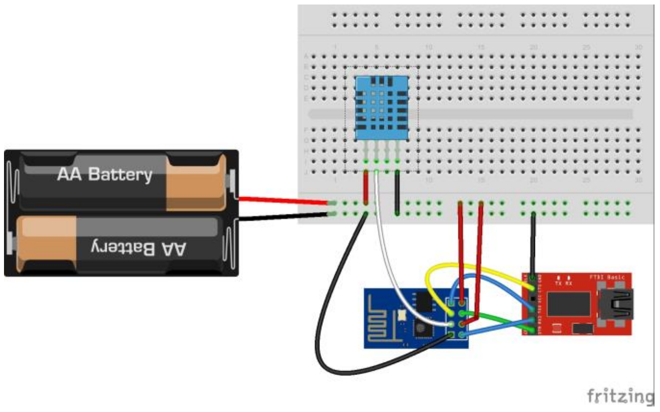
Led pin is now: On

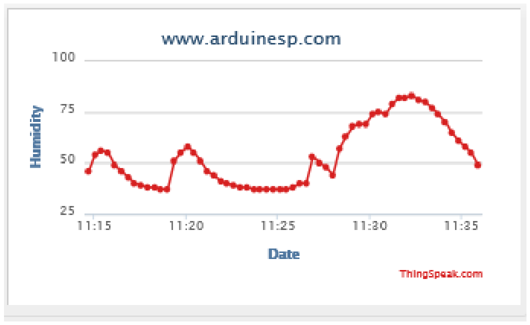
Click here turn the LED on pin 2 ON
Click here turn the LED on pin 2 OFF

Try to click on and the light will go on (or off)When you have uploaded the sketch, you may remove all the wires required for uploading.
This is what is left.



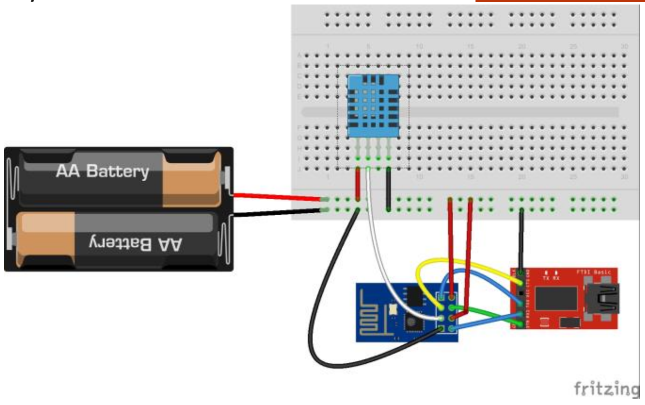
ThingSpeak





Shopping list
– 1 ESP8266 (for example version ESP01)
– DHT11
– 1 USB to TTL (I use FT232RL)
– 8 wires
– 1 breadboard
– a 3.3V power source (I use an battery in this example, power from your USB TTL will most likely might NOT work)

Hardware
The hardware is based on a USB to TTL with **DTR and RST**
If you do not have this have look at the Blink Manual Flash example, and add 2 buttons.

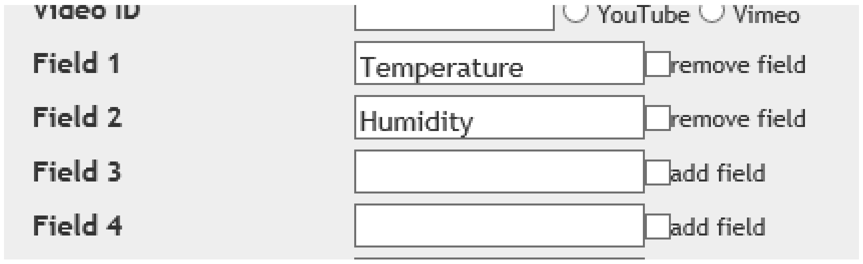


USB TTL ESP8266 ESP-01
——————————
GND — GND
TX   — RX
RX   — TX
RST — RST
DTR — GPIO0

————————
+3.3V   — CH_PD

battery –	— GND
battery +	— VCC
DHT11 VDD (1)	— VCC
DHT11 data (2)	— GPIO2
DHT11 (3)	— not used
DHT11 (4)	— GND

thingspeak
Create an account on www.thingspeak.com . It is very easy.
After login, create a new channel.



Video ID		◯ YouTube ◯ Vimeo
Field 1	Temperature	☐ remove field
Field 2	Humidity	☐ remove field
Field 3		☐ add field
Field 4		☐ add field

Add field1 and field2 in the tab "Chanel settings" like the picture above

Coding

Copy this code. Change ssid into your wifi accesspoint, and change the password into yours. And do not forget to change the API key into yours
and compile.

```
 [cpp]
// www.arduinesp.com
//
// Plot DTH11 data on thingspeak.com using an
ESP8266
// April 11 2015
// Author: Jeroen Beemster
// Website: www.arduinesp.com

#include <DHT.h>
#include <ESP8266WiFi.h>

// replace with your channel's thingspeak API
key,
String apiKey = "4XZ9NA14HQZ7BYD7";
const char* ssid = "ssid";
const char* password = "password";
const char* server = "api.thingspeak.com";
#define DHTPIN 2 // what pin we're connected
to

DHT dht(DHTPIN, DHT11,15);
WiFiClient client;

void setup() {
Serial.begin(115200);
delay(10);
dht.begin();

WiFi.begin(ssid, password);

Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);

WiFi.begin(ssid, password);

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

}

void loop() {

float h = dht.readHumidity();
float t = dht.readTemperature();
if (isnan(h) || isnan(t)) {
Serial.println("Failed to read from DHT
sensor!");
return;
}

if (client.connect(server,80)) { //
"184.106.153.149" or api.thingspeak.com
String postStr = apiKey;
postStr +="&field1=";
postStr += String(t);
postStr +="&field2=";
postStr += String(h);
postStr += "\r\n\r\n";

client.print("POST /update HTTP/1.1\n");
client.print("Host: api.thingspeak.com\n");
client.print("Connection: close\n");
client.print("X-THINGSPEAKAPIKEY:
"+apiKey+"\n");
client.print("Content-Type: application/x-www-
form-urlencoded\n");
client.print("Content-Length: ");
client.print(postStr.length());
client.print("\n\n");
client.print(postStr);

Serial.print("Temperature: ");
Serial.print(t);
Serial.print(" degrees Celcius Humidity: ");
Serial.print(h);
Serial.println("% send to Thingspeak");
}
client.stop();

Serial.println("Waiting…");
// thingspeak needs minimum 15 sec delay
between updates
delay(20000);
}
[/cpp]
```

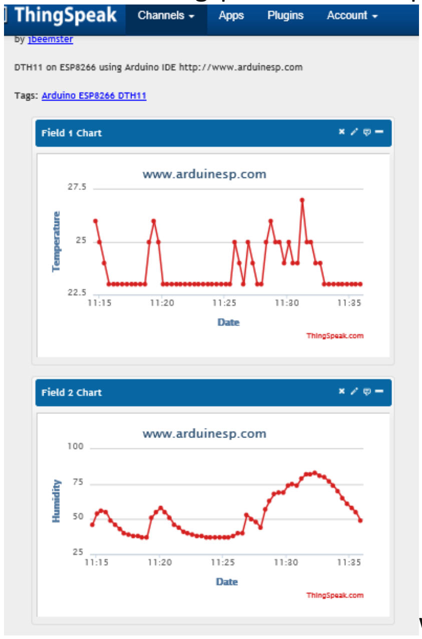
Download sorce code here: ThingspeakESP8266
DHT library (from adafruit) can be downloaded here

Uploading
Press upload in the Arduino IDE, arduino will compile and upload.

Testing
Open the serial monitor from the menu tools. You will see if the data will be send to www.thingspeak.com

The strange characters on the first line after a reset is normal.

Go to www.thingspeak.com and open your channel.

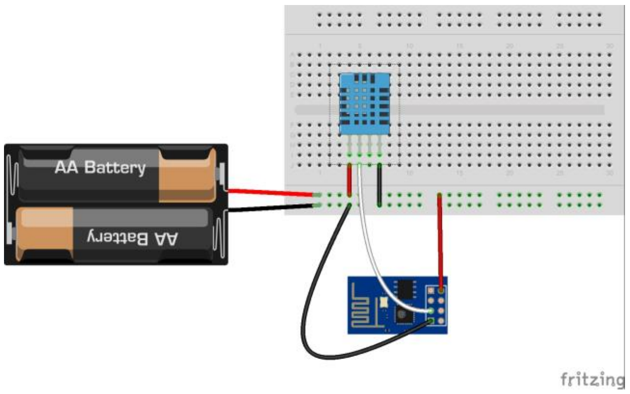


DTH11 on ESP8266 using Arduino IDE http://www.arduinesp.com

Tags: Arduino ESP8266 DTH11

When you have uploaded the sketch, you may remove all the wires required for uploading.

This is what is left.



Good luck.