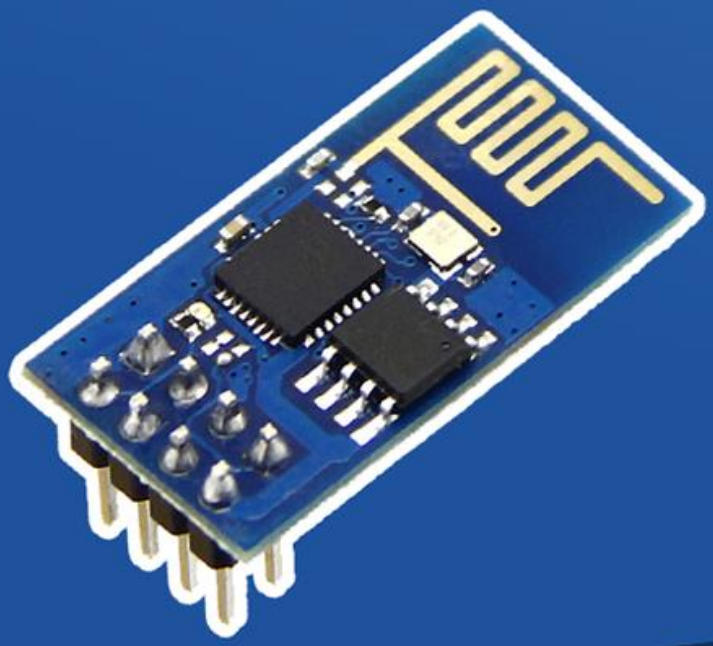
HOME AUTOMATION USING ESP8266

Complete Guide for the ESP8266



How to control devices and read sensors using the low cost ESP8266 WiFi Module

Written by Rui Santos

Disclaimer

This eBook has been written for information purposes only. Every effort has been made to make this eBook as complete and accurate as possible. The purpose of this eBook is to educate. The author (Rui Santos) does not warrant that the information contained in this eBook is fully complete and shall not be responsible for any errors or omissions.

The author (Rui Santos) shall have neither liability nor responsibility to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by this eBook.

Throughout this eBook you will find some links and some of them are affiliate links. This means the author (Rui Santos) earns a small commission from each purchase with that link. Please understand that the author has experience with all of these products, and he recommends them because they are useful, not because of the small commissions he makes if you decide to buy something. Please do not spend any money on these products unless you feel you need them.

Other Helpful Links:

- Join Private Facebook Group
- Contact Support Via Email
- Terms and Conditions

Security Notice

This is the kind of thing I hate to have to write about but the evidence is clear: piracy for digital products is over all the internet.

For that reason I've taken certain steps to protect my intellectual property contained in this eBook.

This eBook contains hidden random strings of text that only apply to *your* specific eBook version that is unique to your email address. You probably won't see anything different, since those strings are hidden in this PDF. I apologize for having to do that – **but it means if someone were to share this eBook I know exactly who shared it and I can take further legal consequences.**

You cannot redistribute this eBook. This eBook is for personal use and is only available for purchase at:

- http://randomnerdtutorials.com/home-automation-using-esp8266

Please send an email to the author (Rui Santos - hello@ruisantos.me), if you found this eBook anywhere else.

What I really want to say is thanking your purchasing this eBook and I hope you have fun with it!

Table of Contents

About the Author

Hey There,

Thank you for purchasing my eBook "Home Automation Using ESP8266"!

I'm Rui Santos, founder of the Random Nerd Tutorials blog, founder of Blog1K.com and author of BeagleBone For Dummies.

If you're new to the world of ESP8266, this eBook is perfect for you! If you already used the ESP8266 before, I'm sure you'll also learn something new.

This eBook contains the information you need to get up to speed quickly and start your own venture with the ESP8266!

Thanks for reading,

-Rui

P.S. If you would like the longer version of my story, you can find it over here.

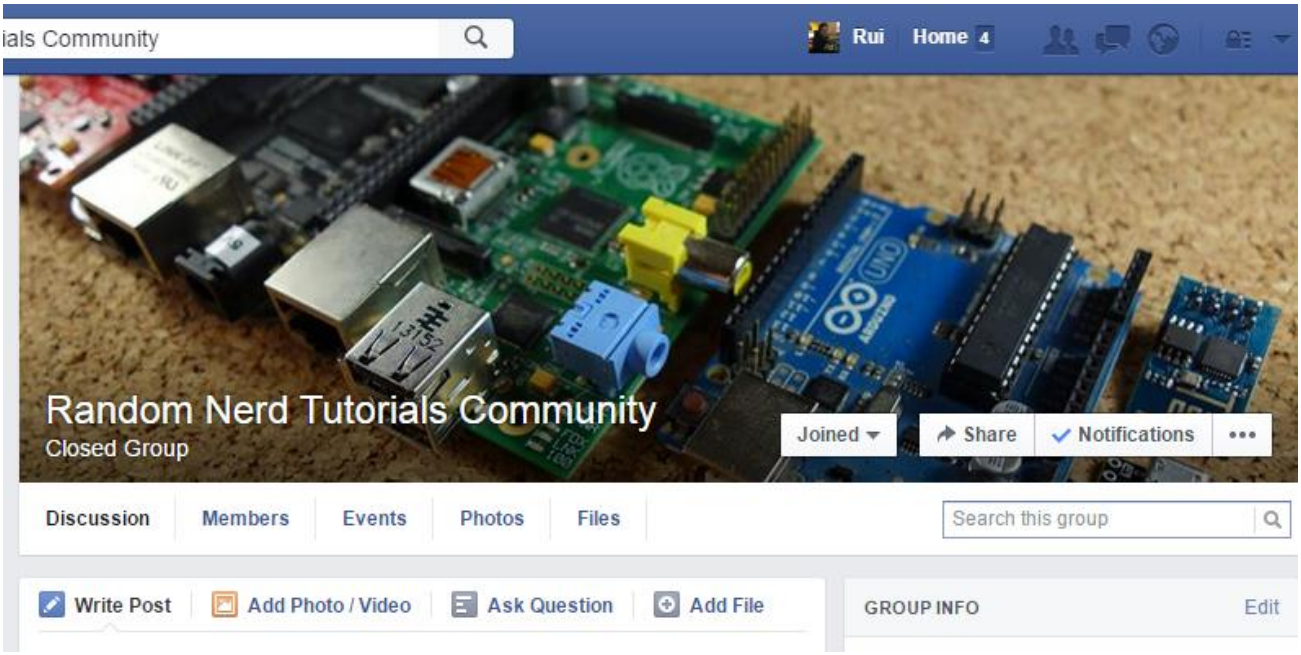
Join the Private Facebook Group

This eBook comes with an opportunity to join a private community of like-minded people. If you purchased this eBook, you can join our private Facebook Group today!

Inside that group you can ask questions and create discussions about everything related to ESP8266, Arduino, BeagleBone, Raspberry Pi, etc.

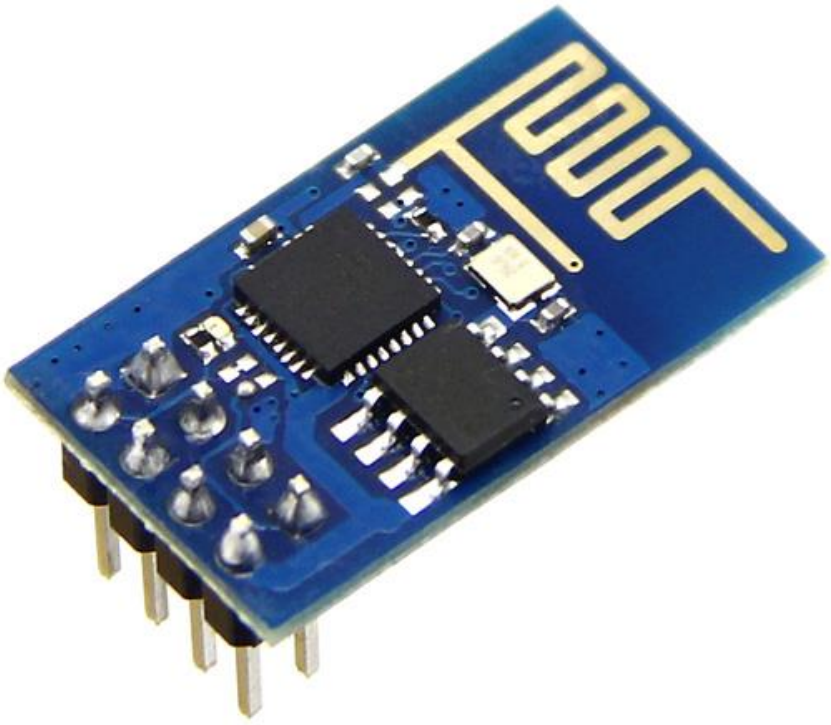
See it for yourself!

- Step #1: Go to -> http://randomnerdtutorials.com/fb/
- Step #2: Click "Join Group"
- Step #3: I'll approve your request within less than 24 hours.



Unit 1

Getting Started with ESP8266



Getting Started with ESP8266

Hello and thank you for purchasing this eBook!

This eBook is my step-by-step guide designed to help you get started with this amazing WiFi module called ESP8266 and build projects that can be applied to automate your home.

This eBook covers:

- Technical specifications the ESP8266
- Where to buy the ESP8266
- How to establish a serial communication with the ESP8266
- What's NodeMCU firmware
- How to flash and install a new firmware
- How to install ESPlorer and how it works
- How to blink an LED with NodeMCU
- Lua programming language
- How to interact with the ESP8266 GPIOs
- How to create a web server
- How to send emails with the ESP8266
- How to create an email notifier with a PIR Motion Sensor
- And a lot more…

Let's Begin!

This first Unit gives you an overview of what you can do with your ESP8266, the ESP8266 technical specifications and where you can buy one.

About the ESP8266

The ESP8266 is a $4 WiFi module with an ARM processor that is great to extend the functionality of a microcontroller such as an Arduino. It talks with your microcontroller via serial.

This entire eBook was designed to take the most of your ESP8266, so you don't even need an Arduino. You just need an ESP8266, a cheap FTDI programmer and a few components!

Comparing with other WiFi solutions in the market, this is definitely a great option for most "Internet of Things" projects! And it's easy to see what it's so popular: it only costs a few dollars and can be integrated in advanced projects.

So what can you do with this low cost module?

You can create a web server, send HTTP requests, control outputs, read inputs and interrupts, send emails, etc.

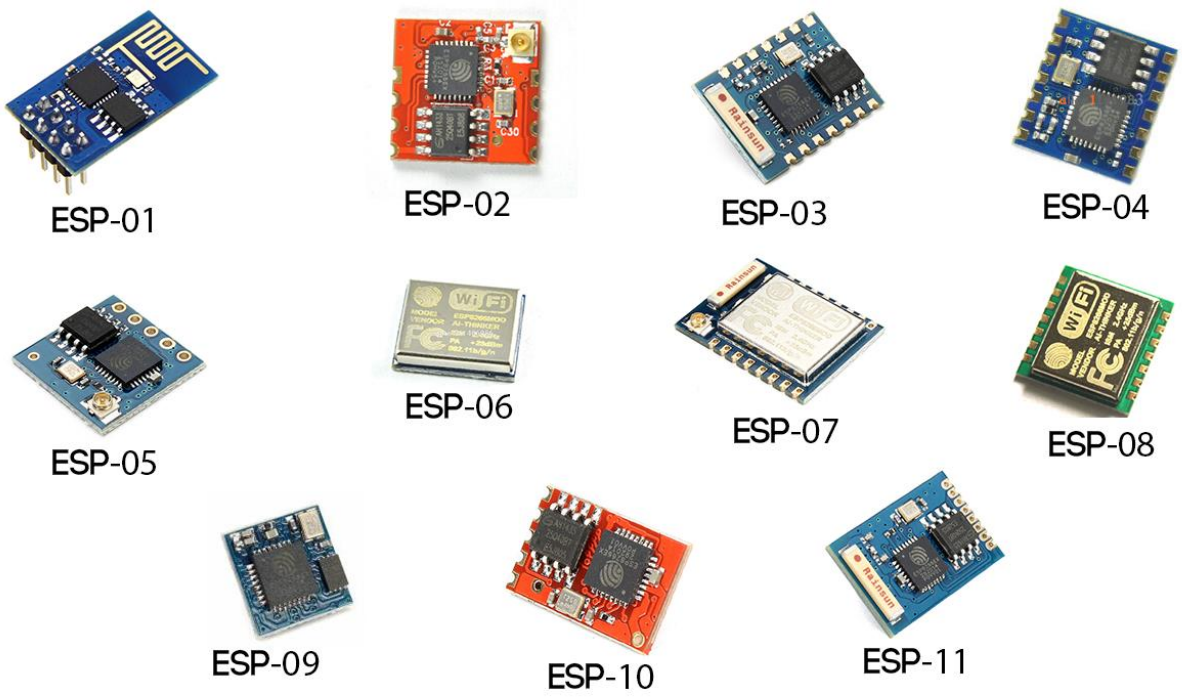
Let's take a look at the ESP8266 specs:

- 802.11 b/g/n protocol
- Wi-Fi Direct (P2P), soft-AP

- Integrated TCP/IP protocol stack
- Built-in low-power 32-bit CPU
- SDIO 2.0, SPI, UART

Finding Your ESP8266

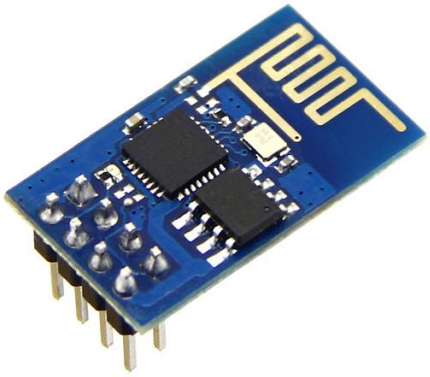
The ESP8266 comes in wide variety of versions (see the following figure). The ESP-01 is the most common and that's the module we'll be using throughout this entire eBook.



Note: I cannot ensure that all the projects presented in this eBook will work with other versions of the ESP8266, but other boards should be compatible with the projects in this eBook.

This module is becoming available through most electronics stores.

But still… The cheapest place to get one is eBay, you purchase one ESP8266 version 01 for less than $4. [Click here to buy this module on eBay](http://randomnerdtutorials.com/ebay-esp8266) (http://randomnerdtutorials.com/ebay-esp8266)



Recommended Alternative Boards

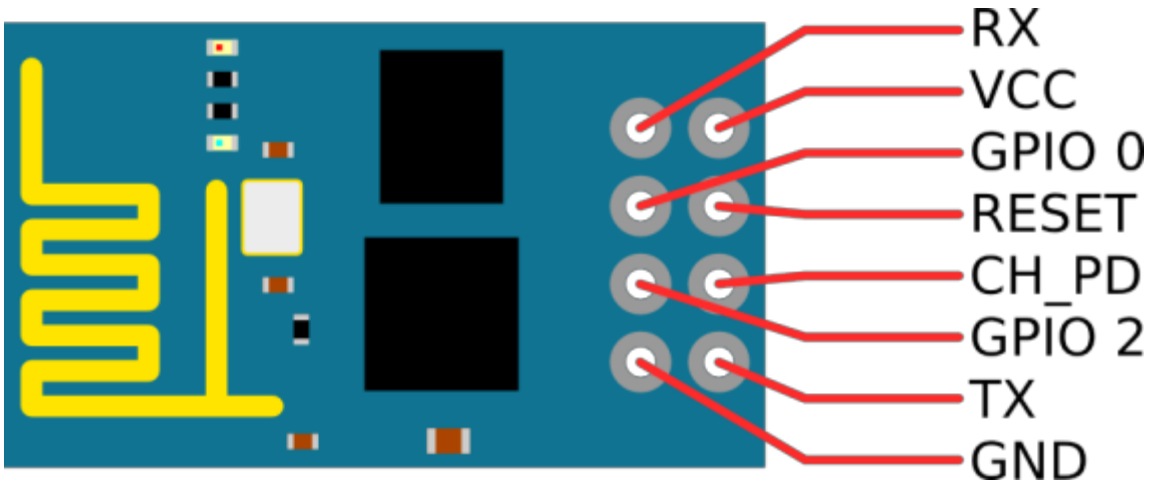
This eBook was also tested with ESP-07 and ESP-12. So you can make all the projects presented in this eBook using those two boards.

If you're using the ESP-07 or ESP-12 you need connect their GPIO 15 to GND for all the schematics in this eBook.

ESP-01 Pinout

Here's a quick overview of the pinout of your ESP8266 version 01:

- RX
- VCC (3.3V)
- GPIO 0
- RESET
- GPIO 2
- CH_PD
- GND
- TX

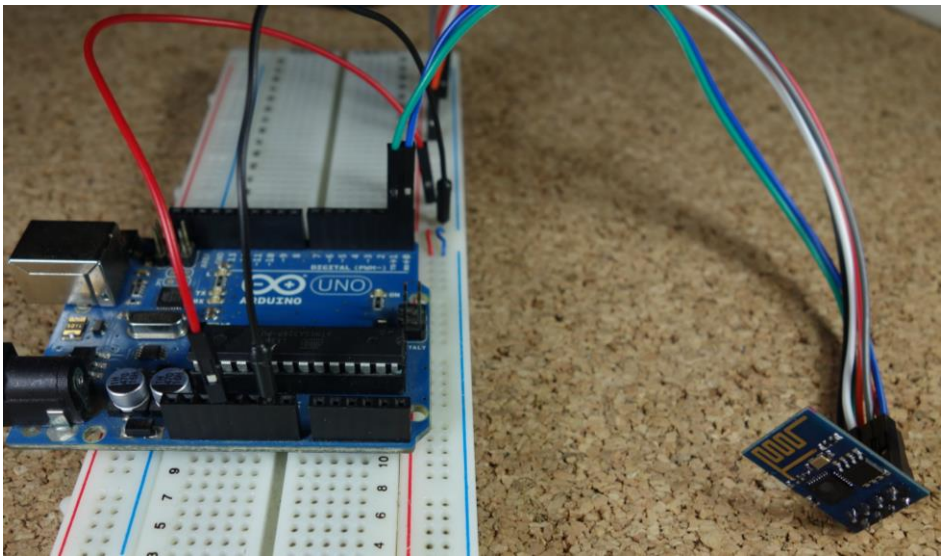


Warning: Before applying power to your module, please note that this module operates at 3.3V. If you plug it up to 5V, it will fry.

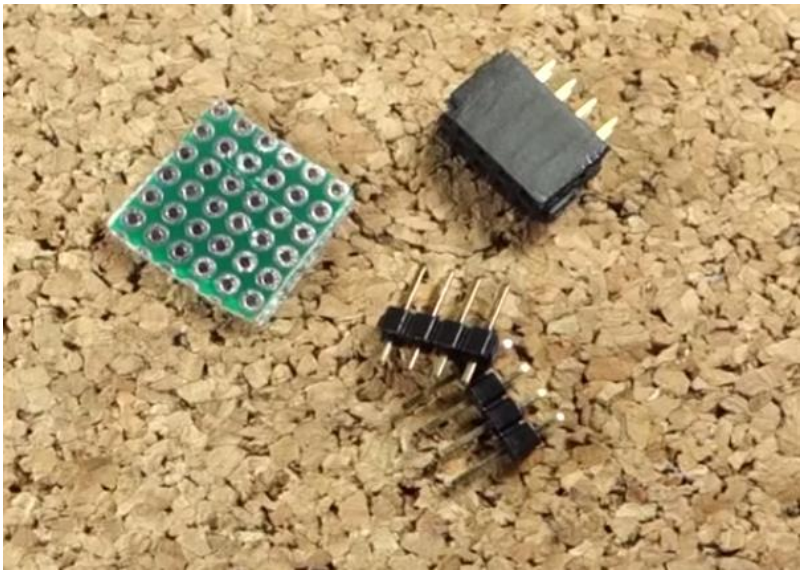
Your ESP8266 is Not Breadboard Friendly…

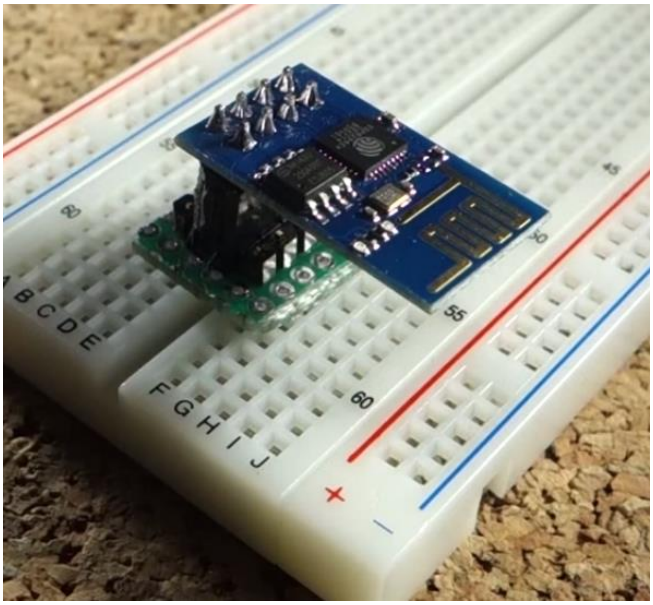
Sadly out of the box your ESP8266 is not breadboard friendly… So you can either use some female to male jumper wires (described below as Option #1) or you can go an extra step and design a small PCB that fits nicely in your breadboard (described below as Option #2). Both ways work fine!

Option #1 – Jumper Wires



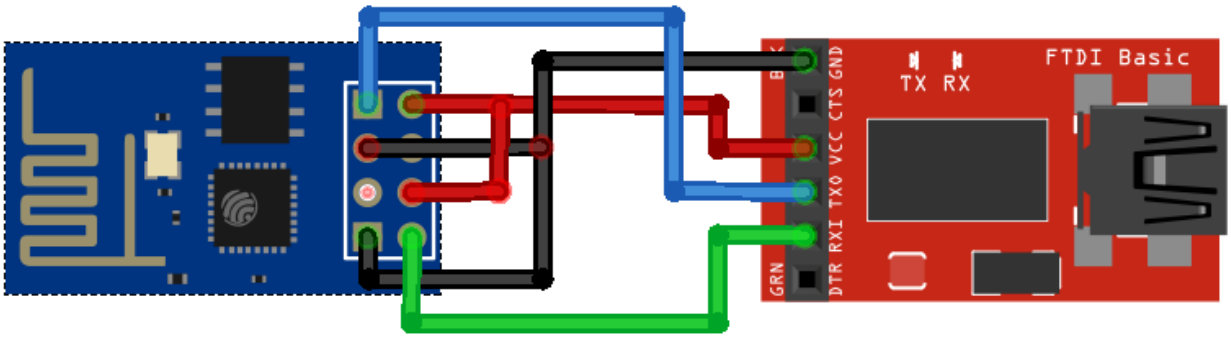
Option #2 – Small PCB





Unit 2

Flashing NodeMCU



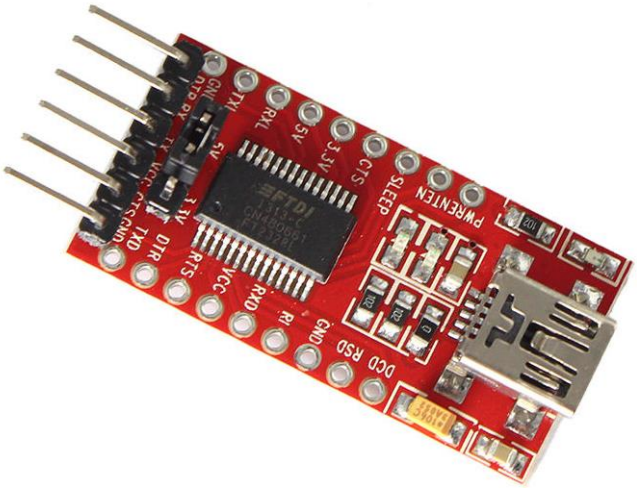
Flashing NodeMCU

Unit 2 covers how you can establish a serial communication with your ESP8266 using an FTDI Programmer or an Arduino. It also explains what NodeMCU is, what it does and how to install it.

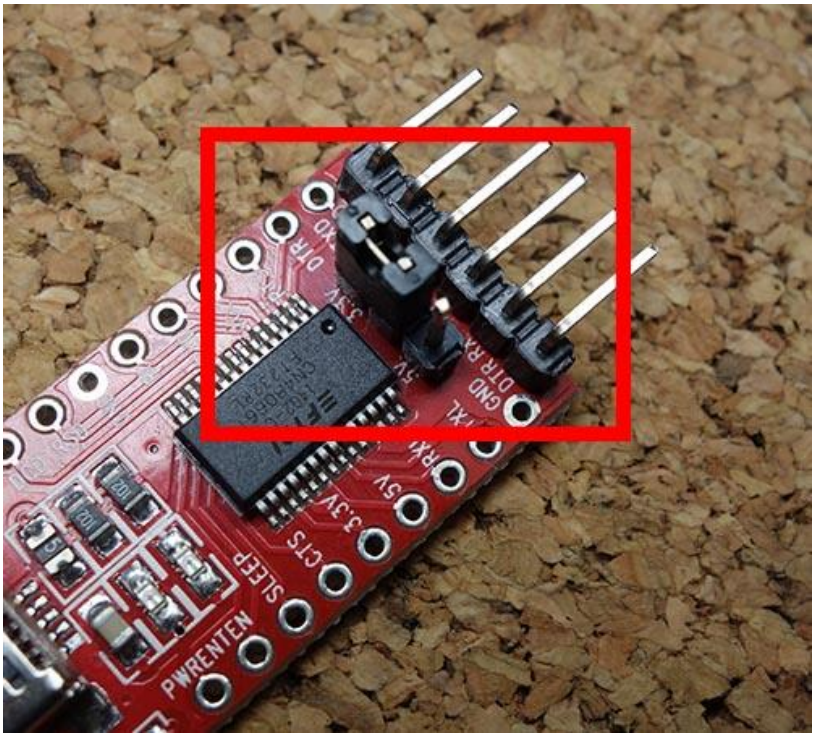
Using FTDI Programmer 3.3V

If you have a 3.3V FTDI Programmer it's really easy to get started and establish a serial communications with your ESP8266 WiFi module.

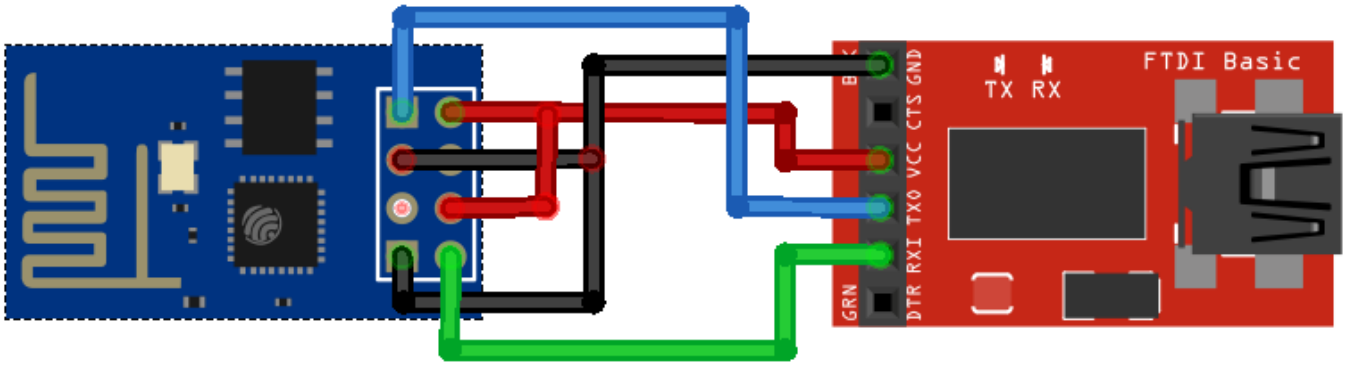
You can click here to purchase one FTDI Programmer from eBay for $3 (http://randomnerdtutorials.com/ebay-ftdi-programmer).



Important: Most FTDI programmers have a jumper to convert from 5V to 3.3V. So make sure your FTDI programmer is set to 3.3V operation (as shown in the following Figure).



Follow the circuit in the figure below to connect your ESP8 to your FTDI Programmer to establish a serial communication.



Note: The circuit above has GPIO 0 connected to GND, that's because we want to update the firmware. In normal usage (if you're not flashing your ESP with a new firmware) it would be connected to VCC.

Extra Tips for Windows Users

In this section I have two tips that might help you if you're on a Windows PC.

If you have a brand new FTDI Programmer and you need to install your FTDI drivers on Windows, visit this website for the official drivers: http://www.ftdichip.com/Drivers/VCP.htm. In alternative you can contact the seller that sold you the FTDI Programmer.

If you're having trouble installing the FTDI drivers on Windows 7/8/8.1 follow this tutorial: http://youtu.be/SPdSKT6KdF8.

In the video I mentioned earlier, the guy tells you to download the drivers from the FTDI website, read carefully the YouTube description of his video to find all the links. Here's the drivers you need:

http://www.ftdichip.com/Drivers/CDM/CDM%20v2.12.00%20WHQL%20Certified.zip

Arduino Alternative (Not Recommended)

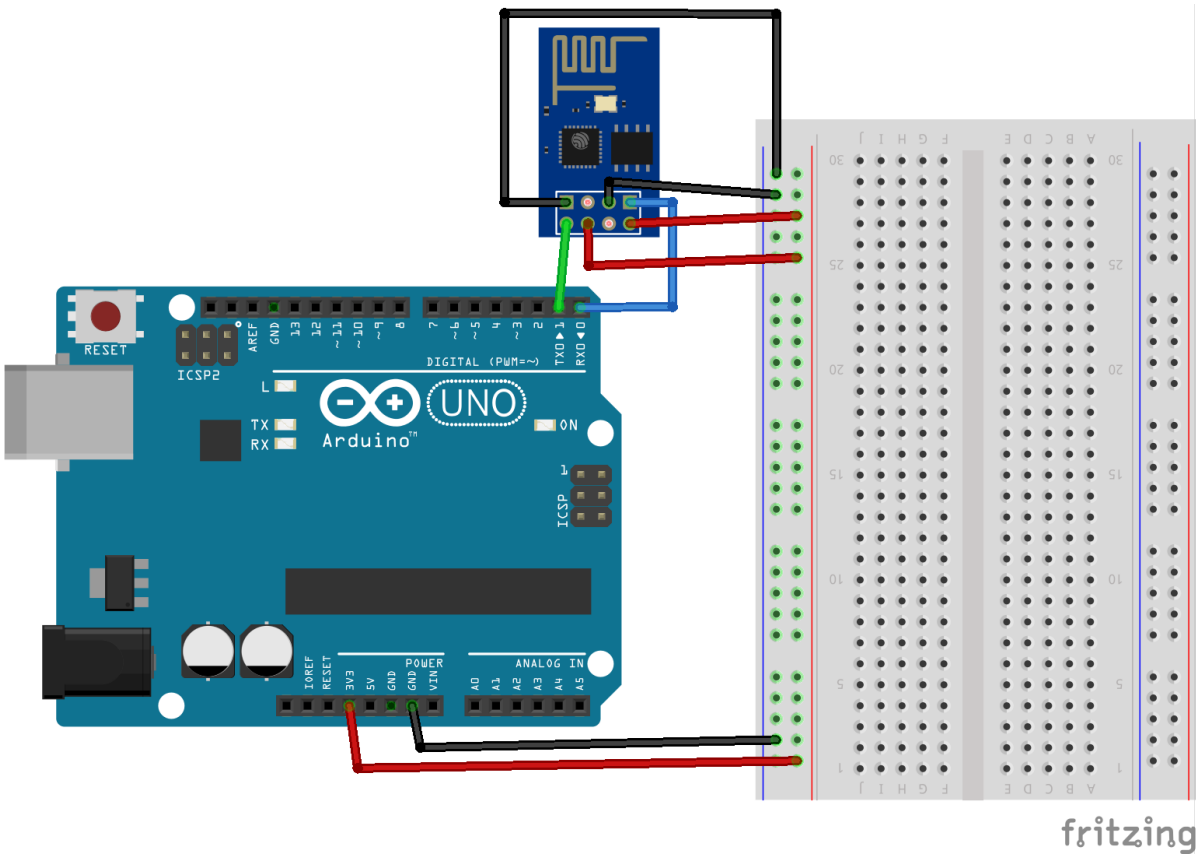
I don't recommend using an Arduino to power your ESP8266 and establish a serial communication. You may break both your Arduino and your ESP8266. Since the ESP does not have 5V tolerant pins.

Since this module is so cheap I've decided to give it a try and use an Arduino. And it works just fine for me... But again it's not very stable and might not work for you. **So use an FTDI Programmer instead.**

If you decide to try with an Arduino here's what you need to do:

1. Unplug everything from your Arduino (disconnect your old circuits and don't connect your ESP8266, yet)
2. Connect your Arduino UNO to your computer with a USB cable
3. Open your Arduino IDE
4. Go to File > Examples > Basics > BareMinimum
5. Upload that sketch to your Arduino
6. Follow the circuit below (after reading the warning)

Warning: The following circuit works fine for me and I've been playing with the same ESP module for a long time without any problem. But as a final project you shouldn't power your ESP8266 module with the 3.3V from your Arduino, because it might not supply enough current. So I advise you to use an FTDI programmer or an external power supply to ensure that it works fine for you. If you decide to use an Arduino I encourage you to add a 3.3v level shifter or a voltage divider to your ESP8266 RX pin.



Note: The circuit above has GPIO 0 connected to GND, that's because we want to update the firmware. In normal usage (if you're not flashing your ESP8266 with a new firmware) it would be connected to VCC.

Why Flashing Your ESP8266 with NodeMCU?

NodeMCU is a firmware package that allows you to program the ESP8266 modules with LUA script. And you will find it very similar to the way you program your Arduino.

With just a few lines of code you can establish a WiFi connection, control the ESP8266 GPIOs, turning your ESP8266 into a web server and a lot more.

With this firmware your ESP8266 obtains functionalities of the ESP8266's internal microcontroller, such as, SPI, UART, I2C, PWM, GPIO and more...

External Resources:

- NodeMCU: http://nodemcu.com/index_en.html
- Firmware: https://github.com/nodemcu/nodemcu-firmware
- Firmware Flasher: https://github.com/nodemcu/nodemcu-flasher
- Node MCU API: https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en

Downloading NodeMCU Flasher For Windows

At this point (when I wrote this eBook) the official place to download the latest firmware flasher is here: https://github.com/nodemcu/nodemcu-flasher/blob/master/Win64/Release/ESP8266Flasher.exe

To download the NodeMCU firmware flasher you go to the preceding URL and click "Raw" (as shown in the following figure).

If they move the flasher to another URL you might need to browse through this GitHub repository: https://github.com/nodemcu/nodemcu-flasher.

If this URL changes I'll update this eBook. Post a comment here if you find that this URL has changed: http://randomnerdtutorials.com/fb. Thank you!

Flashing NodeMCU on Windows PC

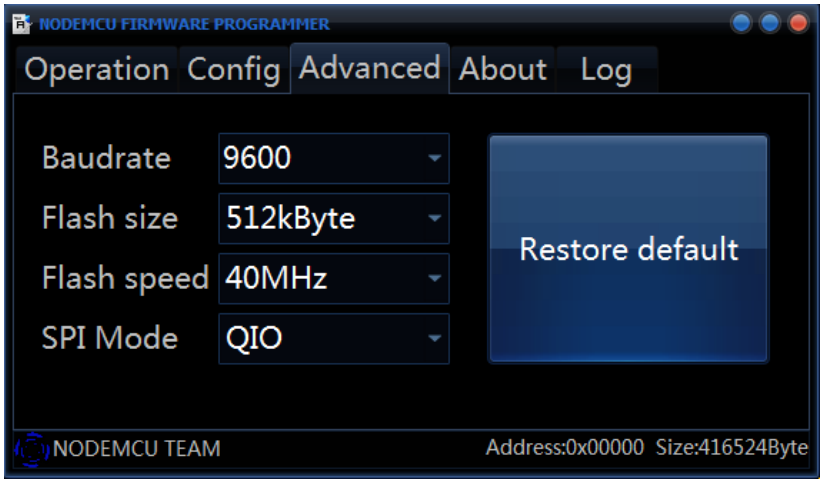
So... At this point you have the circuit prepared to flash your ESP8266 and the NodeMCU firmware flasher for Windows. How do we load this firmware on the ESP8266 chip?

It's pretty simple. Having the serial communication established between your ESP8266 and your FTDI Programmer, you just need to follow these instructions:

1. Connect your FTDI Programmer to your computer
2. Open the firmware flasher for windows
3. Go to the Advanced tab (Figure below)
4. Set your baudrate to 9600

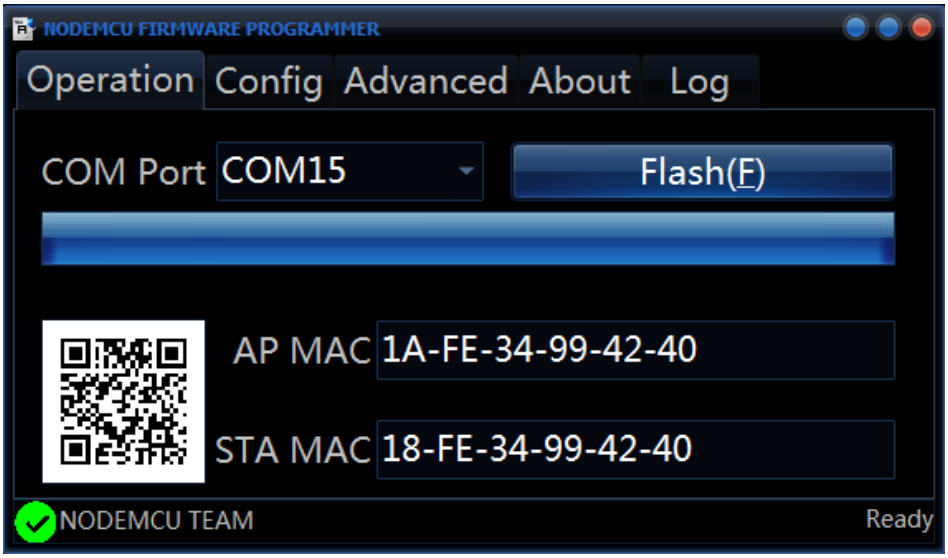
Warning: Some ESP8266 modules by default work at baudrate 115200. If you're trying to flash and nothing happens, try to change the baudrate to 115200.

Here's how your firmware flasher should look in the Advanced tab:



Then go to the Operation tab and follow these instructions:

1. Select the COM Port of your FTDI programmer
2. Press Flash
3. That should start flashing the firmware to the ESP
4. When it's finished you should see a green Check icon at the bottom left corner



While the flash process is occurring, you're going to see the blue LED from the ESP8266 blinking, if you don't see it blinking and you flasher is stuck, repeat this process again:

1. Check your connections
2. Disconnect the FTDI Programmer from the computer and recoonect it
3. Re-open your NodeMCU flasher
4. Check the NodeMCU flasher Advanced Settings
5. Try a different baudrate (9600 or 115200)
6. Try to flash your firmware again

When the flash process finishes, remove power from the ESP and disconnect the GPIO 0 pin from GND.

Flashing NodeMCU on Mac OS X or Linux

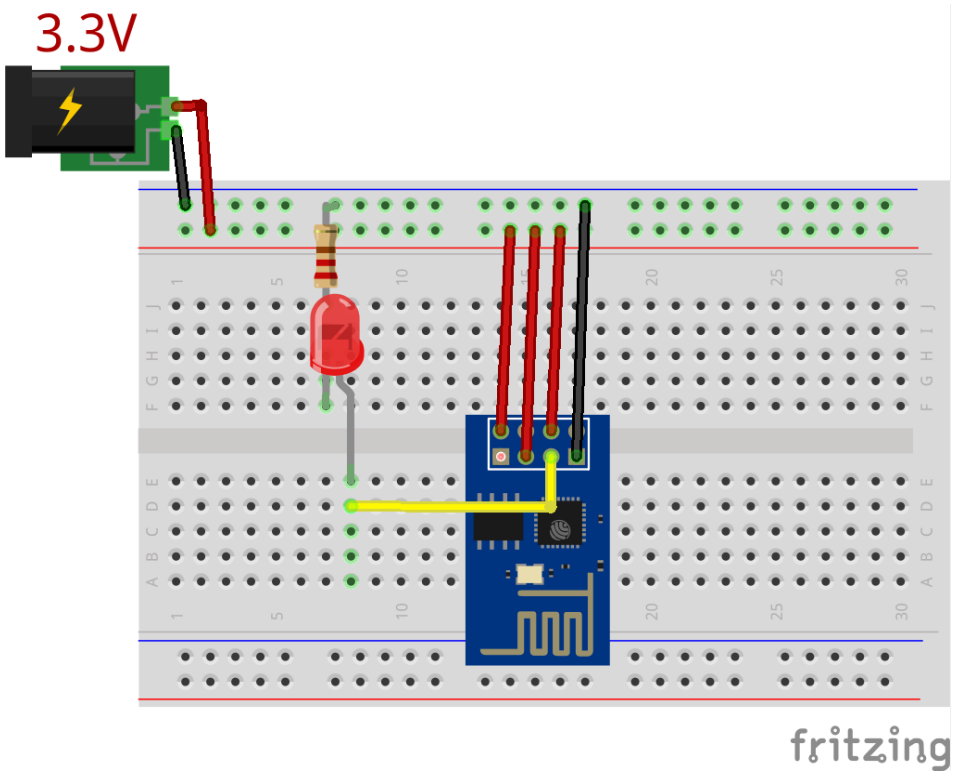
I want to be honest with you, I personally didn't try to flash NodeMCU in Mac OS X or Linux, but here's a good tutorial to do that:

- http://www.whatimade.today/flashing-the-nodemcu-firmware-on-the-esp8266-linux-guide/

If you have problems flashing the firmware in Mac OS X or Linux, please try to use a Windows PC or post a comment in the Facebook group.

Unit 3

Building Your First Blinking LED Project with NodeMCU



Building Your First Blinking LED Project with NodeMCU

In this Unit you're going to download and install ESPlorer (which is an open-source IDE for your ESP). You're also going to design a simple circuit to blink an LED with NodeMCU.

Why do we always blink an LED first?

That's a great question! If you can blink an LED you can pretty much say that you can turn any electronic device on or off. Whether is an LED a lamp or your toaster.

What's the ESPlorer?

The ESPlorer is an IDE for ESP8266 developers. It's multiplatform, this simply means that it runs on Windows, Mac OS X or Linux (it was created in JAVA).

This software allows you to easily establish a serial communications with your ESP8266, send commands, upload code and much more.

Requirements:

You need to have JAVA installed in your computer. If you don't have, go to this website: http://java.com/download, download and install the latest version.

External Resources:

- Download ESPlorer: http://esp8266.ru/esplorer/
- GitHub Repository: https://github.com/4refr0nt/ESPlorer

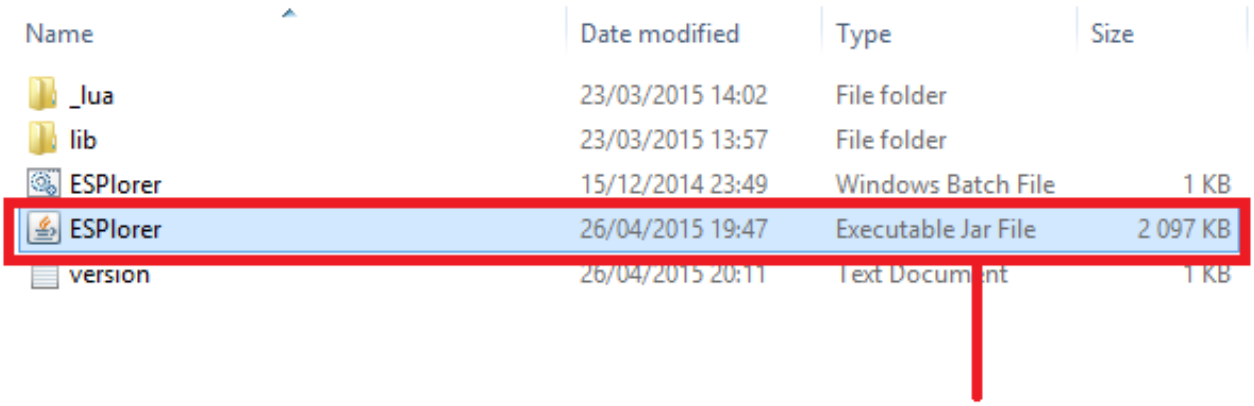
Downloading ESPlorer

Now let's download the ESPlorer IDE, visit the following URL: http://esp8266.ru/esplorer/#download

Then click that big "Download Now" button (as shown below).

Installing ESPlorer

Grab the folder that you just downloaded. It should be named "ESPlorer.zip" and unzip it. Inside that folder you should see the following files:

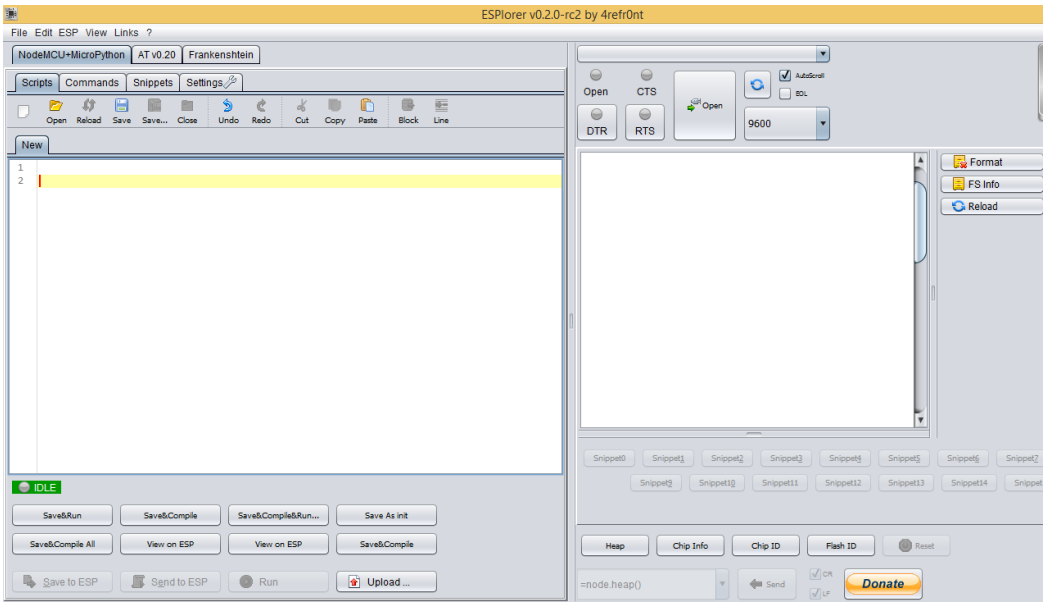


Execute the "ESPlorer.jar" file and the ESPlorer IDE should open after a few seconds (the "ESPlorer.jar" file is what you need to open every time you want to work with the ESPlorer IDE).

Note: If you're on Mac OS X or Linux, just type this command line in your terminal to run the ESPlorer: *sudo java –jar ESPlorer.jar*.

ESPlorer IDE

When the ESPlorer first opens, this is what you should see:



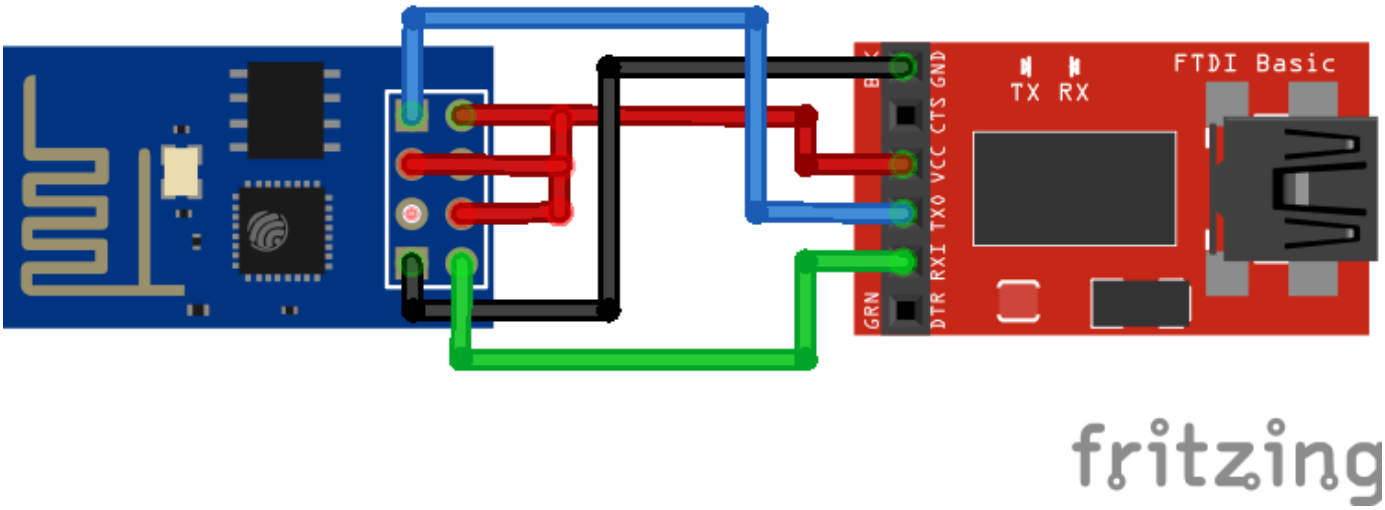
ESPlorer has a lot of options you don't need and you might feel overwhelmed with all those buttons and menus.

But don't worry, I'll go over each of that features you will need to complete all the projects in this eBook.

Do not close the ESPlorer IDE, you're going to use it in just a few minutes.

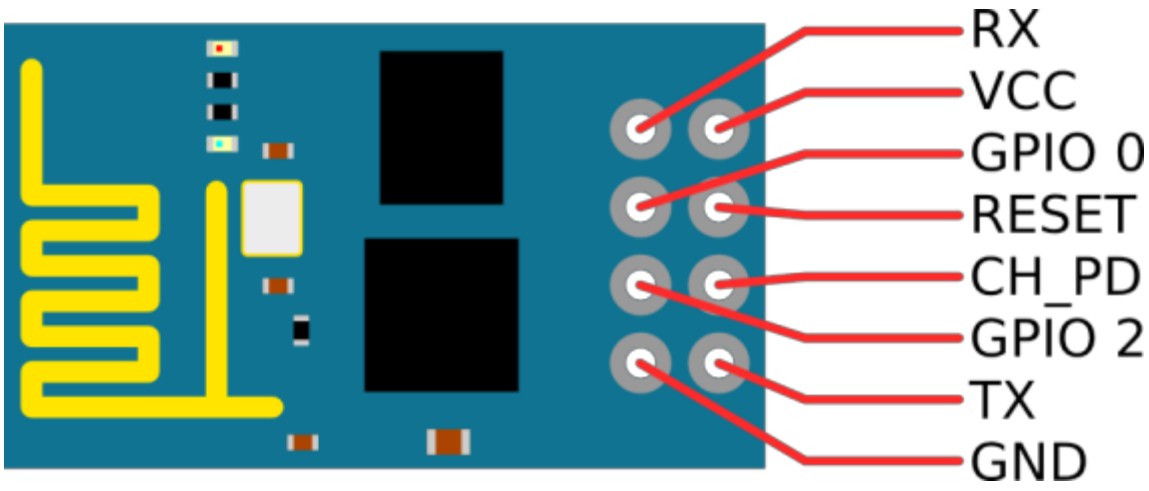
Schematics

To upload code to your ESP8266, you should connect your ESP8266 to your FTDI Programmer like the figure below:



About GPIOs Assignment

Just a quick recap, here's the ESP-01 pinout (all pins operate at 3.3V):



Important: In the next section called "Writing Your Lua Script" when we define:

pin = 3

we are referring to GPIO 0, and if we define:

pin = 4

we are referring to GPIO 2. This is how NodeMCU Firmware is defined. You don't need to worry about this, simply remember that 3 refers to GPIO 0 and 4 refers to GPIO 2. I'll explore this concept in more detail later in this eBook.

Writing Your Lua Script

Below is your script to blink an LED. You can download the Lua Script in the following link: https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/blob/master/Projects/ESP8266/blink.lua

init.lua

```
1  lighton=0
2  pin=4
3  gpio.mode(pin,gpio.OUTPUT)
4  tmr.alarm(1,2000,1,function()
5      if lighton==0 then
6          lighton=1
7          gpio.write(pin,gpio.HIGH)
8      else
9          lighton=0
10          gpio.write(pin,gpio.LOW)
11     end
12 end)
```

How this script works:

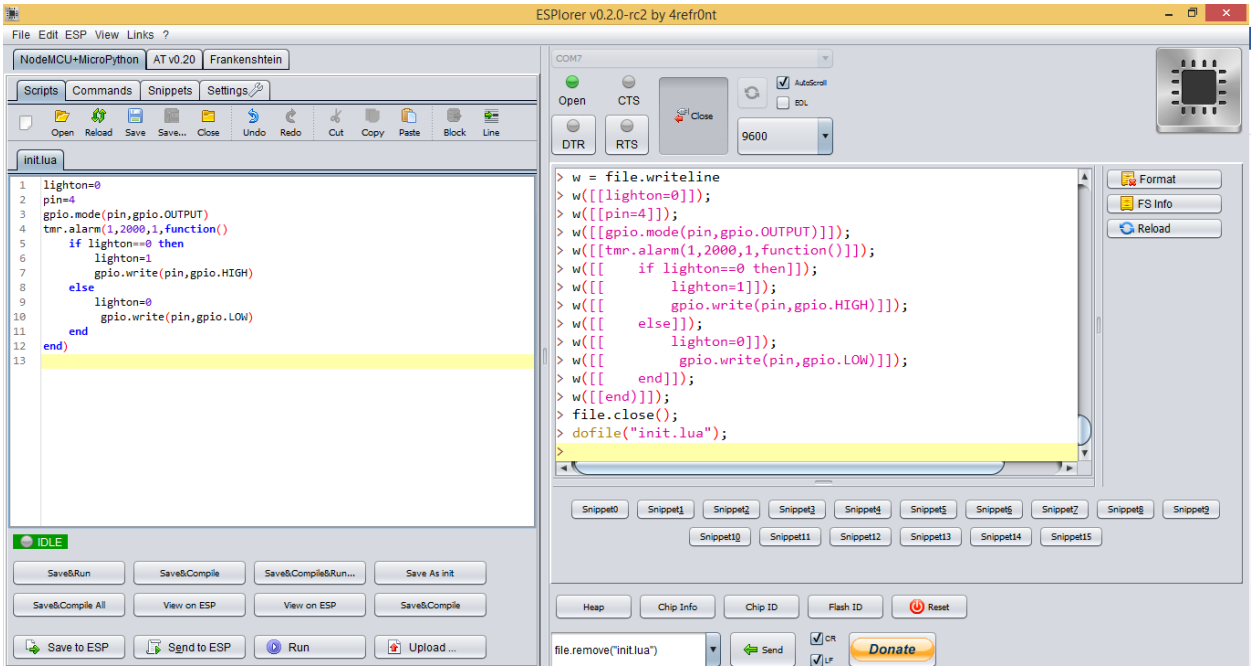
1. Create a variable called *lighton* to control the current state of your LED
2. Define *pin = 4* (4 refers to GPIO 2) as an *OUTPUT*
3. Next create a *tmr.alarm()* function that is executed every 2 seconds (2000 milliseconds)
4. The script checks the value of the variable *lighton*. *If* the variable contains 0, it means the output pin is *LOW* and the LED is off. The program will then change the variable *lighton* to 1, set the pin to *HIGH*, and the LED will go on.

5. If the variable *lighton* did not contain 0 it means the LED is on. Then the script would run the statements in the *else* section. It would first reassign the variable *lighton* to contain 0, and then change the pin to *LOW* which turns the LED off.
6. The program repeats steps 4. and 5. every two seconds, which causes the LED to blink!

Note: You should name your Lua script "init.lua", that ensures that your ESP8266 executes your script every time it restarts.

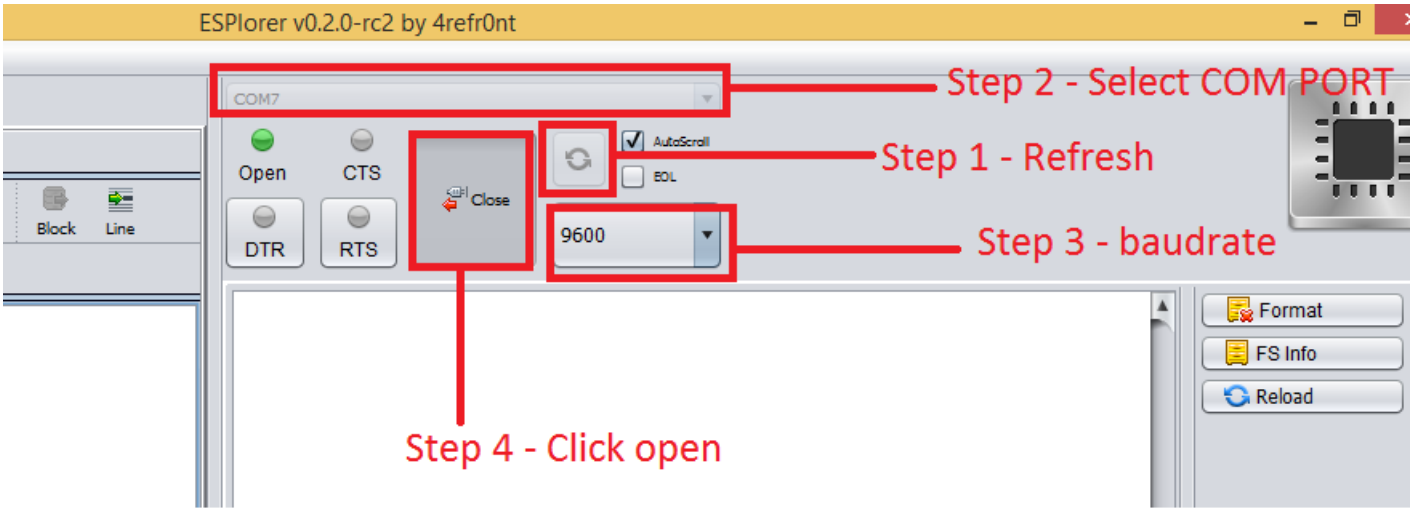
Uploading Code

Having your ESP8266+FTDI Programmer connected to your computer, go to the ESPlorer IDE:



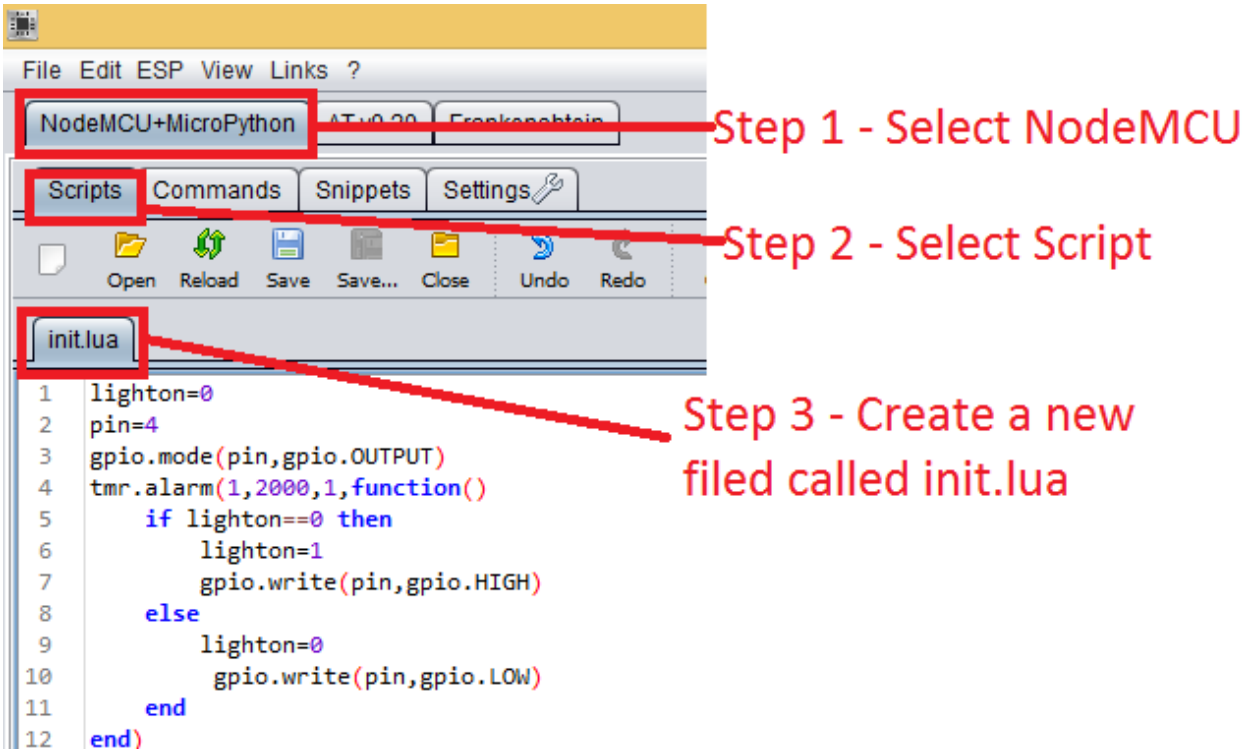
Look at the top right corner of your ESPlorer IDE and follow these instructions:

1. Press the Refresh button
2. Select the COM port for your FTDI programmer
3. Select 9600 as your baudrate
4. Click Open

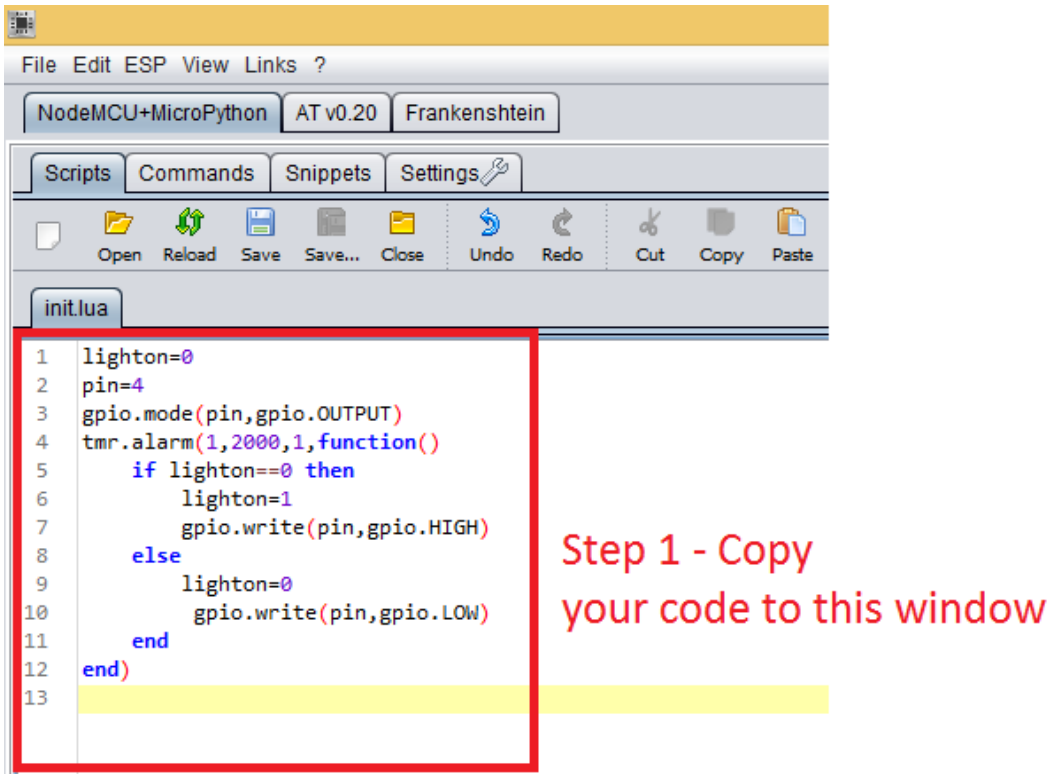


Then in the top left corner of the ESPlorer IDE, follow these instructions:

1. Select NodeMCU
2. Select Scripts
3. Create a new file called “init.lua”



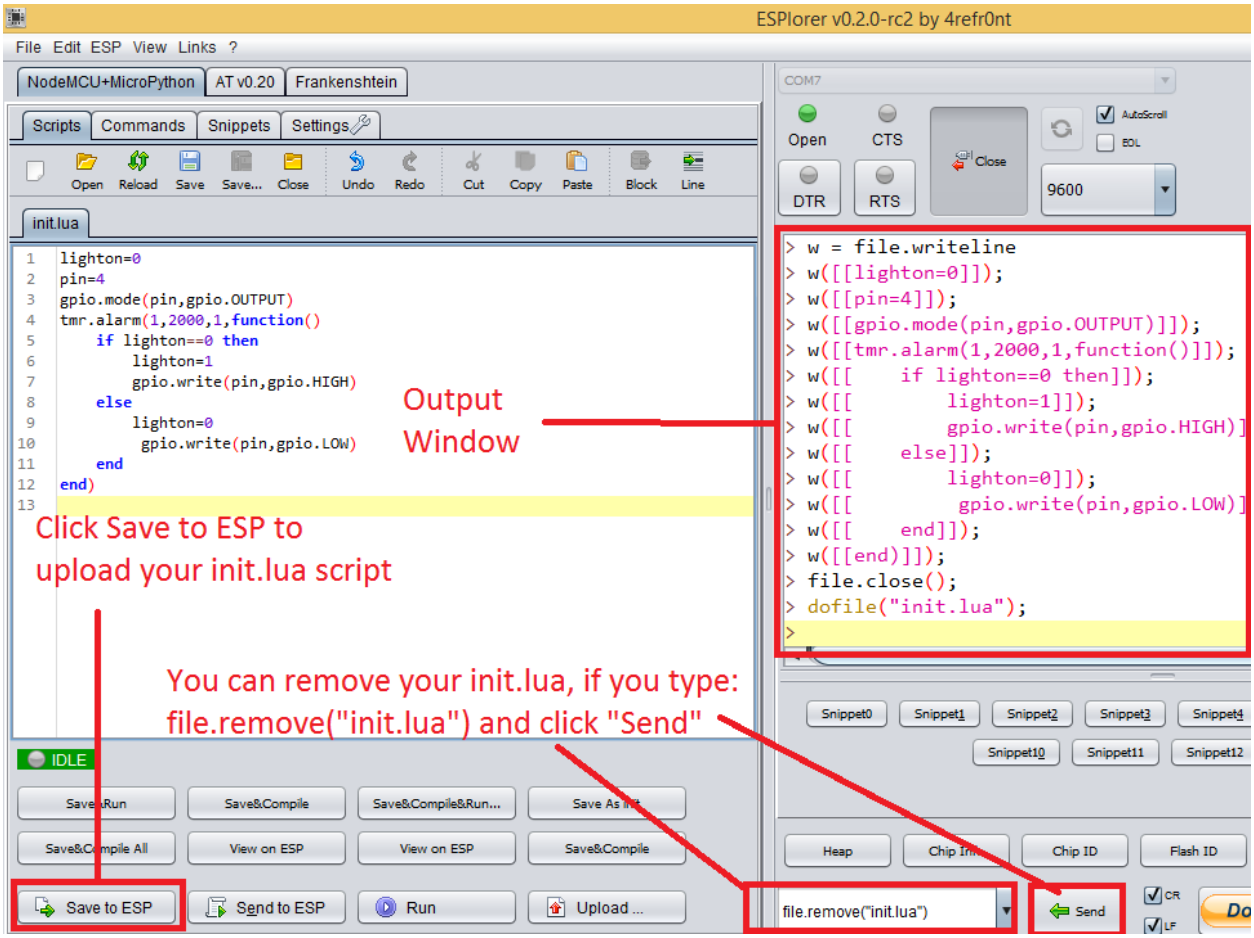
Copy the Lua script (which was created in the previous section) to the code window (as you can see in the Figure below):



The next step is to save your code to your ESP8266.

At the left bottom corner click the button “Save to ESP”.

The output window will display the commands being sent to the ESP8266. It should look similar to the Figure below.

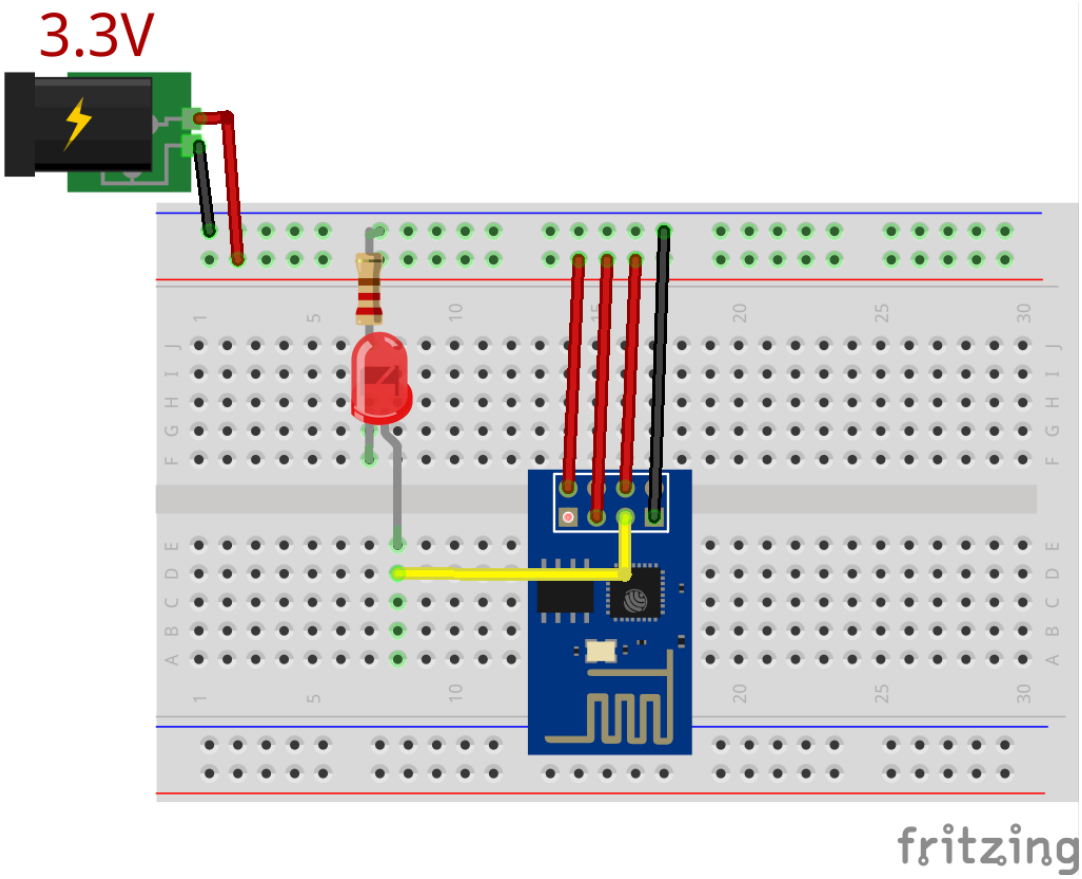


Note: You can easily delete the “init.lua” file from the ESP. Simply type *file.remove(“init.lua”)* and press the button “Send” (see Figure above). Or you can type the command *file.format()* to remove all the files saved in your ESP8266.

Here's Your Final Circuit

After uploading the code to the module, unplug it from your computer. Next, change the wiring to match the following diagram.

Then, apply power from a 3.3V source to the ESP.



Restart your ESP8266!

Congratulations, you've made it! Your LED should be blinking every 2 seconds!

Unit 4

Lua Programming Language – The Basics

Lua Programming Language – The Basics

Before diving deeper into more projects with the ESP8266 I thought it would be helpful to create a Unit dedicated to Lua Programming language. Lua is light weight programming language written in C. It started as an in-house project in 1993 by Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Cele.

More details about this program language can be found here: http://en.wikipedia.org/wiki/Lua_%28programming_language%29

NodeMCU is a Lua based firmware package for the ESP8266, so it's important that you know the Lua basics in order to write your own scripts for the ESP8266.

Variables

In Lua, though we don't have variable data types, we have three types based on the scope of the variable. The scope means that a variable can be either of global or local scope.

- **Global variables:** All variables are considered global (unless it is declared as a local)

```
pin = 3
test = "It works!"
```

- **Local variables:** When the type is specified as local for a variable, its scope is limited with the functions inside their scope

```
local pin = 3
local test = "It works!"
```

- **Table fields:** This is a special type of variable that can hold anything except *nil* (we won't cover table fields)

Note: Lua is case-sensitive. So a variable called *PIN* is different from *Pin* or *pin*.

Data Types aka Value Type

Lua is a dynamically typed language, so the variables don't have types, only the values have types. Values can be stored in variables, passed as parameters and returned as results.

The list of data types for values are given below.

Value Type	Description
string	Arrays of characters
number	Represents real (double precision floating point) numbers
boolean	Includes true and false as values. Generally used for condition checking.
function	A method that is written in Lua
nil	No data stored in the variable
table, userdata and thread	We won't cover these 3 value types in this eBook.

Here's a great illustration of the value types in action:

```
print(type("Hello World!"))    -- string
print(type(7))                 -- number
print(type(true))              -- boolean
print(type(print))             -- function
print(type(nil))               -- nil
```

Note: When working with NodeMCU in your ESP8266, you'll see the value *nil* come up once in a while. It simply means that a variable is not defined. Also, if you want to delete a variable, simply set that variable to the *nil* value.

Comments

Comments are plain text that explains how the code works. Anything designated as a comment is ignored by the ESP module. Comments start with two dashes: --. There are two types of comments:

- Single-line comment

```
print("Hello World!") -- That's how you make a comment
```

- Multi-line comment

```
--[[
print("Hello World!") this is a multi line comment
--]]
```

Operators

An operator is a symbol that tells the interpreter to perform specific mathematical or logical manipulations. Lua language is rich in built-in operators and provides following type of operators:

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Misc Operators

For all the following tables and examples in this section assume that you have two variables: *A* that stores number 1 and a variable *B* that stores number 2.

```
A = 1
B = 2
```

Arithmetic Operators

Operator	Example	Result
+	A + B	3
-	A - B	-1
*	A * B	2
/	B / A	2
%	B % A	0
^	B^2	4
-	-A	-1

Relational Operators

Operator	Example	Result
==	(A == B)	not true
~=	(A ~= B)	true
>	(A > B)	not true
<	(A < B)	true
>=	(A >= B)	not true
<=	(A <= B)	true

Logical Operators

Operator	Example	Result
and	(A and B)	false
or	(A or B)	true
not	!(A and B)	true

Concatenation Operator

Now imagine that you have two new variables:

```
a = "Hello "
b = "World!"
```

Operator	Example	Result
..	a..b	"Hello World!"

Loops

A loop allows us to execute a block of code multiple times for as long as the condition (*bolean_value*) is true.

```
--While Loop
while bloean_value
do
 -- executes this code while is true
end

-- or Foor Loop
for init,max/min value, increment
do
  -- executes this code while is true
end
```

if... else statements

if... else statements are one the most important coding tools for adding control to your program. *if... else* statements are used as follows:

```
if bolean_value then
    -- if the bolean_value is true
else
    -- if the bolean_value is false
end
```

Their use is just as the words describe them: If a certain condition is met (*bolean_value=true*), then the code inside the *if* statement runs. If the condition is false (*bolean_value=false*), the code inside the *else* statement runs.

Functions

Functions are great ways to organize your code. If you want to do something multiple times, instead of repeating your code several times, you create a separate function that you can call and execute any time.

This is how you create a new function that takes one parameter (the temperature in Kelvin) and converts that temperature to both Celsius and Fahrenheit:

```
function displayTemperature(kelvin)
  celsius = kelvin - 273.15
  print("Temperature in Celsius is: ", celsius)

  fahrenheit = (celsius*9/5+32)
  print("Temperature in Fahrenheit is: ", fahrenheit)
end

k = 294 --temperature in Kelvin
displayTemperature(k) -- calls function
```

Unit 5

Interacting with the ESP8266 GPIOs using NodeMCU Firmware

Interacting with the ESP8266 GPIOs using NodeMCU Firmware

GPIO stands for *general purpose input/output*, which sums up what pins in this mode can do: they can be either inputs or outputs for the vast majority of applications.

In this Unit we're going to explore the NodeMCU GPIO API. If you want learn more about this API, you can visit the official wiki:

- https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en.

This Unit explains how to set the different modes for each GPIO. I will share snippets of code that can be applied to your projects.

Since these snippets work in the ESP8266, feel free to test them!

Pin Mode

When using a GPIO pin we first need to specify it's mode of operation. There are three possible modes that you can assign to each pin (one mode at a time for any pin):

Mode	Reference	Description
OUTPUT	gpio.OUTPUT	You set the pin to HIGH or LOW
INPUT	gpio.INPUT	You read the current state of the pin
INTERRUPT	gpio.INT	Similar to INPUT, you're constantly checking for a change in a pin. When a change occurs it executes a function.

Here's How to Assign Pins

The table below shows the GPIO pin index assignments for the ESP8266. The ESP version 01 has only two: GPIO 0 and GPIO 2:

IO index (in code)	ESP8266 GPIO
0 [*]	GPIO 16
1	GPIO 5
2	GPIO 4
3	**GPIO 0**
4	**GPIO 2**
5	GPIO 14
6	GPIO 12

7	GPIO 13
8	GPIO 15
9	GPIO 3
10	GPIO 1
11	GPIO 9
12	GPIO 10

OUTPUT Mode

Using *gpio.write()* you can set any GPIO to HIGH (3.3V) or LOW (0V). That's how you turn an LED on or off.

Here is how to make the pin GPIO 2 put out a HIGH (3.3V):

```
pin = 4
gpio.mode(pin, gpio.OUTPUT)
gpio.write(pin, gpio.HIGH)
```

Here is how to make the pin GPIO 2 put out a LOW (0V):

```
pin = 4
gpio.mode(pin, gpio.OUTPUT)
gpio.write(pin, gpio.LOW)
```

INPUT Mode

Using *gpio.read()* you can read the current state of any GPIO. For example, here is how you would check if a button was pressed.

```
pin = 4
gpio.mode(pin, gpio.INPUT)
print (gpio.read(pin))
```

If *print(gpio.read(pin))* = 1 the button was being pressed and if *print(gpio.read(pin))* = 0 the button was released, or was not pressed.

INTERRUPT Mode

Using *gpio.trig()* allows us to detect when something occurs in a pin (when its state changes from HIGH to LOW or vice-versa). When that event occurs it executes a function.

Imagine that you have a motion sensor. When motion is detected you want to trigger a specific function that does something (send an email for example).

To create an interrupt, this is what you would do:

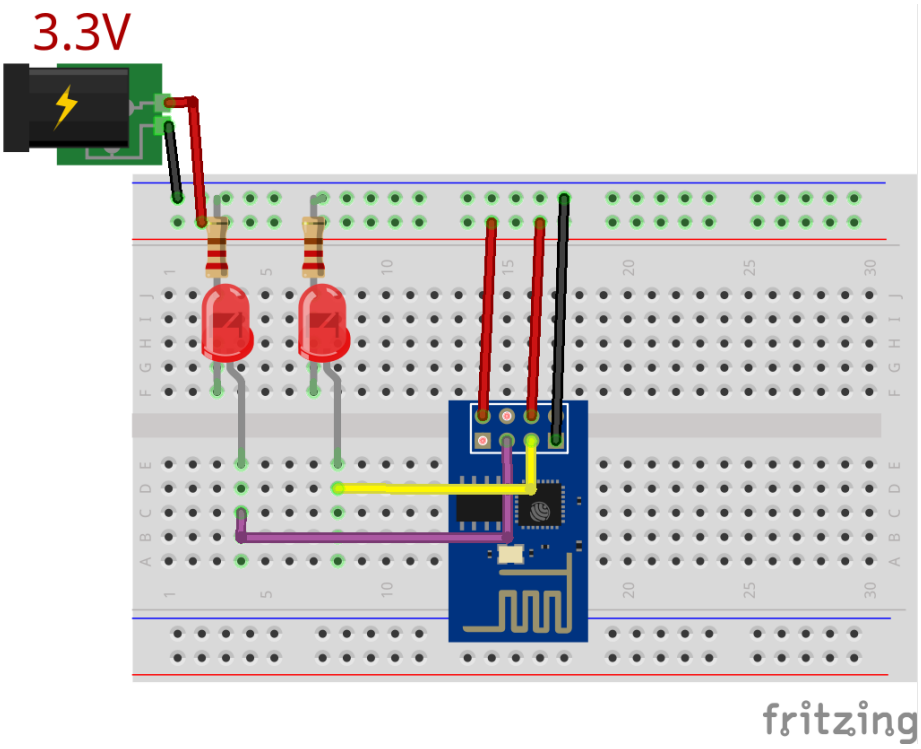
```
pin = 4
gpio.mode(pin, gpio.INT)

function onChange ()
    print('Motion Detected')
end

gpio.trig(pin, 'up', onChange)
```

The second parameter of the function called *gpio.trig()* can take these 5 types of events:

Event Name	When Occurs
up	Pin goes to HIGH
down	Pin goes to LOW
both	Pin goes LOW->HIGH or HIGH->LOW
low	While Pin is LOW
high	While Pin is HIGH

Unit 6

Web Server with ESP8266 – Controlling Outputs

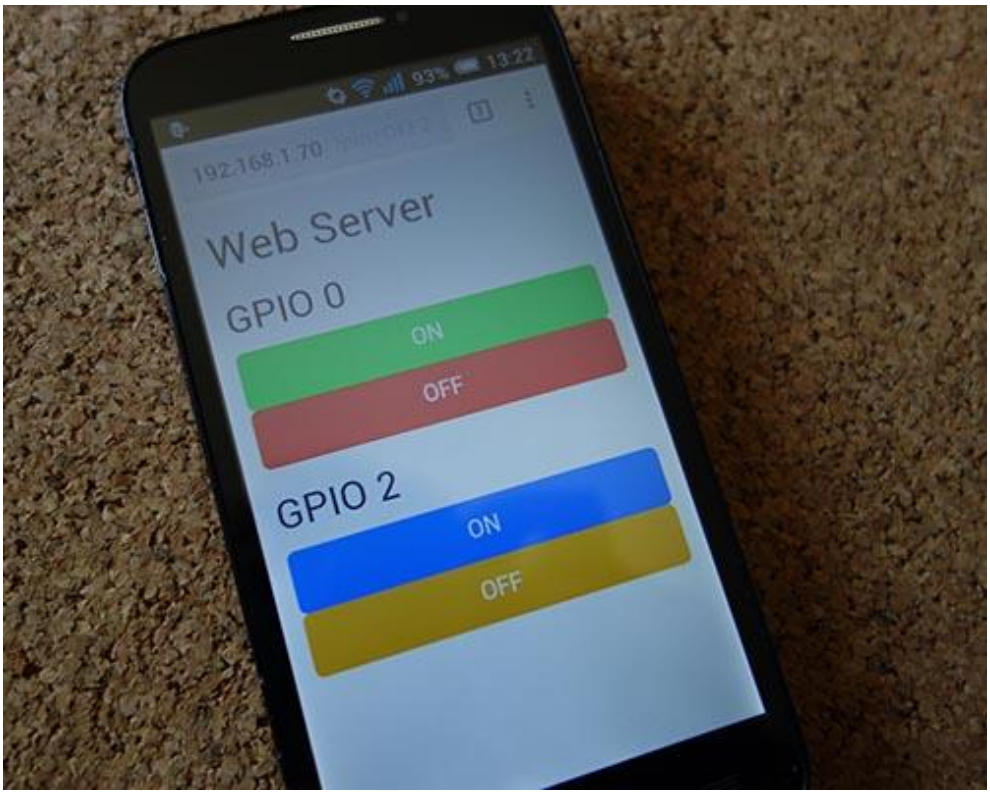


Web Server with ESP8266 – Controlling Outputs

In this Unit you're going to create a web server with your ESP8266 that can be accessed with any device that has a browser. This means you can control the ESP GPIOs from your laptop, smartphone, tablet and so on!

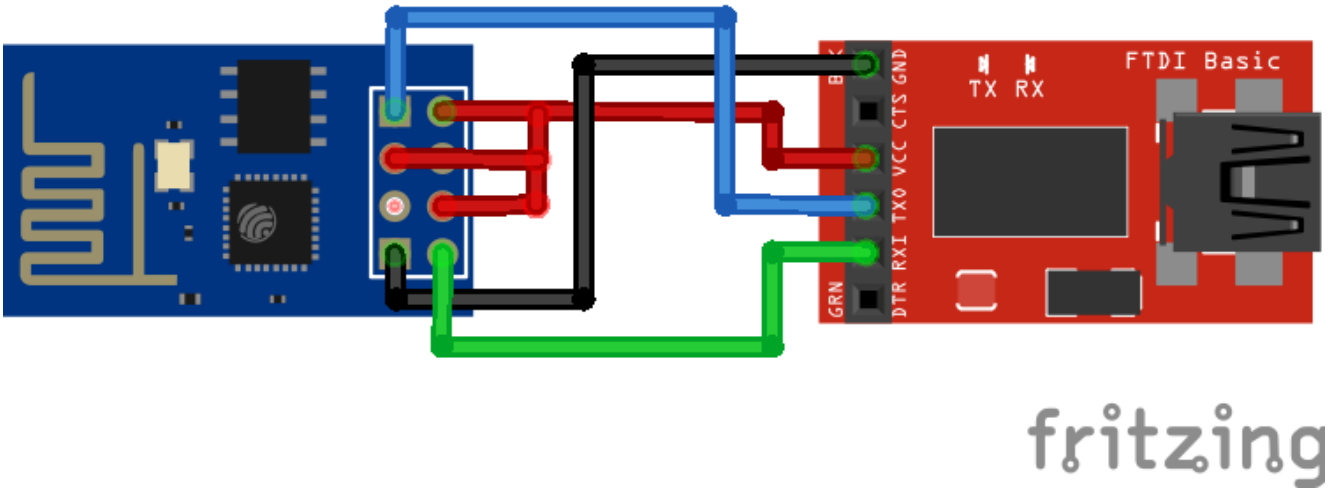
In this project we're going to control two LEDs, but this is just an example, the idea is to replace those LEDs with a Power Switch Tail or a relay to control any electronic devices that you want.

Spoiler Alert: This is what you're going to achieve at the end of this project!



Schematics

To upload code to your ESP8266, you should connect your ESP8 to your FTDI Programmer like the figure below:



Writing Your Lua Script

Below is the script to create a web server that controls two outputs (GPIO 0 and GPIO 2). You can download the Lua Script for this project in the following link: https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/blob/master/Projects/ESP8266/web-server.lua

Let's see how this code works!

The snippet of code below starts by setting the mode of your ESP8266 to a station. Then, you configure your ESP8266 with your own credentials (network name and password). You actually need to replace that second line with your credentials, so that your ESP can connect to your network.

The *print()* function in line 3 prints your ESP8266 IP address in the output window of the ESPlorer IDE (you need that IP to access your web server).

Next, you create two variables (*led1* and *led2*) which refer to GPIO 0 and GPIO 2 respectively and define them as *OUTPUTs*.

```
1  wifi.setmode(wifi.STATION)
2  wifi.sta.config("YOUR_NETWORK_NAME","YOUR_NETWORK_PASSWORD")
3  print(wifi.sta.getip())
4  led1 = 3
5  led2 = 4
6  gpio.mode(led1, gpio.OUTPUT)
7  gpio.mode(led2, gpio.OUTPUT)
```

The next thing to do is creating your web server on port 80. You do it like this:

```
 8
 9  srv=net.createServer(net.TCP)
10  srv:listen(80,function(conn)
11
12      end)
13  end)
```

Inside your web server you tell exactly what happens when a connection is established with a client *conn:on()*. You also create a few local variables that store your web page and your current URL path.

```
    conn:on("receive", function(client,request)
        local buf = "";
        local _, _, method, path, vars = string.find(request, "([A-Z]+) (.+)?(.+) HTTP");
        if(method == nil)then
            _, _, method, path = string.find(request, "([A-Z]+) (.+) HTTP");
        end
        local _GET = {}
        if (vars ~= nil)then
            for k, v in string.gmatch(vars, "(%w+)=(%w+)&*") do
                _GET[k] = v
            end
        end
```

The *buf* variable stores your web page. It's just a basic web page that uses the Bootstrap framework (see code below).

Learn more about the Bootstrap framework: http://getbootstrap.com/.

Your web page has four buttons to turn your LEDs *HIGH* and *LOW*. Two buttons for GPIO 0 and the other two for GPIO 2.

Your buttons are simply HTML tags with a CSS class that gives them that look. So when you press a button, you open *another* web page that has a different URL. And that's how your ESP8266 knows what it needs to do (whether is to turn your LEDs *HIGH* or *LOW*).

```
        buf = buf.."<head>";
        buf = buf.."<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">";
        buf = buf.."<script src=\"https://code.jquery.com/jquery-2.1.3.min.js\"></script>";
        buf = buf.."<link rel=\"stylesheet\" href=\"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.
        buf = buf.."</head><div class=\"container\">";

        buf = buf.."<h1>Web Server</h1>";
        buf = buf.."<h2>GPIO 0</h2>";
        buf = buf.."<div class=\"row\">";
        buf = buf.."<div class=\"col-md-2\"><a href=\"?pin=ON1\" class=\"btn btn-block btn-lg btn-
        buf = buf.."<div class=\"col-md-2\"><a href=\"?pin=OFF1\" class=\"btn btn-block btn-lg btn
        buf = buf.."</div>";
        buf = buf.."<h2>GPIO 2</h2>";
        buf = buf.."<div class=\"row\">";
        buf = buf.."<div class=\"col-md-2\"><a href=\"?pin=ON2\" class=\"btn btn-block btn-lg btn-
        buf = buf.."<div class=\"col-md-2\"><a href=\"?pin=OFF2\" class=\"btn btn-block btn-lg btn
        buf = buf.."</div></div>";

```

This final snippet of code is what checks which button in your webpage was pressed. Basically, it checks the URL that you have just clicked.

Let's see an example. When you click the button OFF from the GPIO 0 you open this URL: http://192.168.7.2/?pin=OFF1. Your Lua code checks that URL and with some *if… else* statements it knows that you want your GPIO 0 (which is defined as *led1*) to go *LOW*.

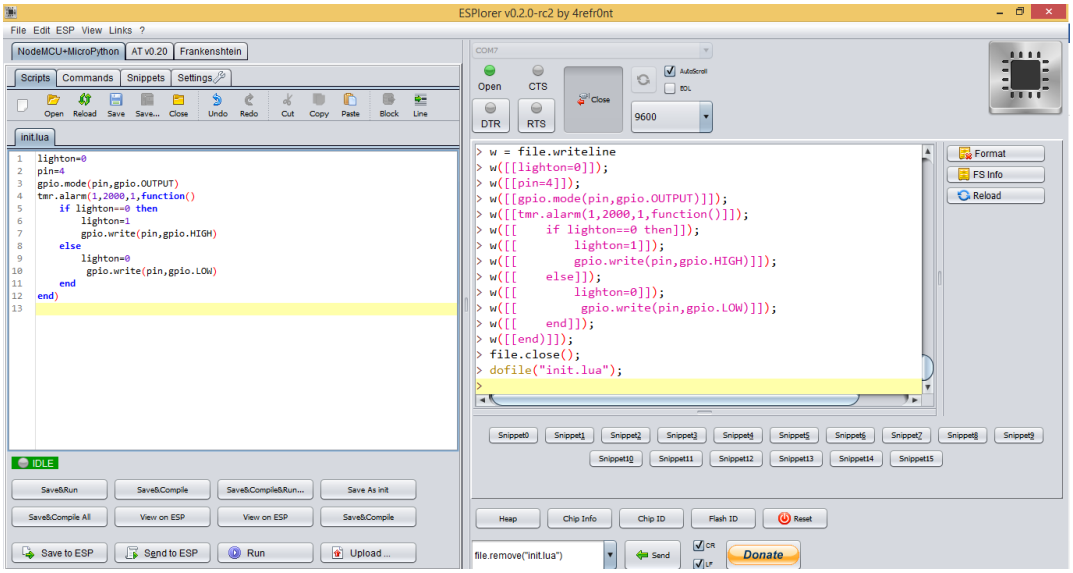
```
40
41          local _on,_off = "",""
42          if(_GET.pin == "ON1")then
43                gpio.write(led1, gpio.HIGH);
44          elseif(_GET.pin == "OFF1")then
45                gpio.write(led1, gpio.LOW);
46          elseif(_GET.pin == "ON2")then
47                gpio.write(led2, gpio.HIGH);
48          elseif(_GET.pin == "OFF2")then
49                gpio.write(led2, gpio.LOW);
50          end
51          client:send(buf);
52          client:close();
53          collectgarbage();
54      end)
55  end)
```

Uploading Code

Now you need to upload the code you just wrote to your ESP8266 (your file should be named as "init.lua").

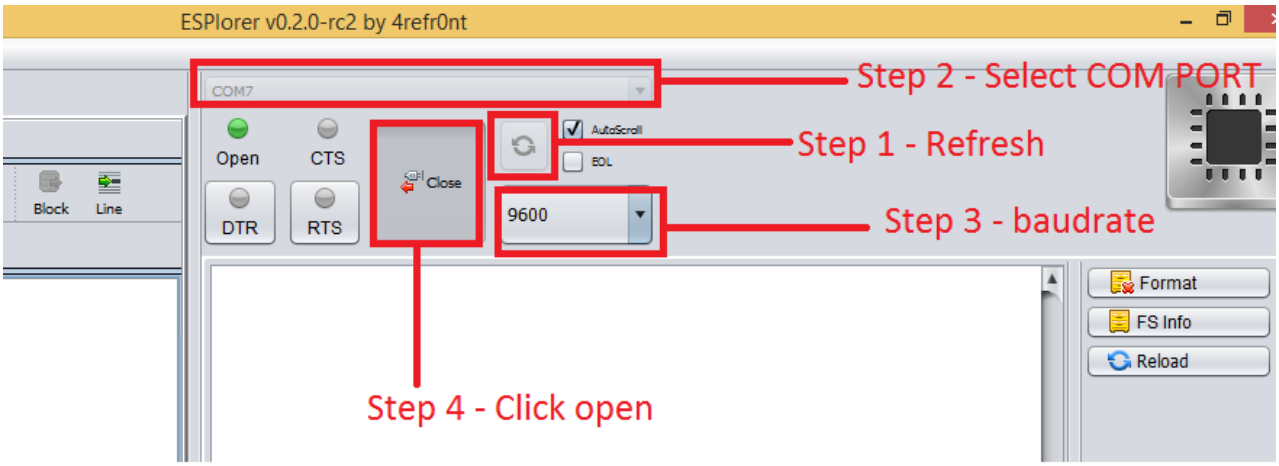
If you've followed Unit 3, you already know how to upload code to your ESP8266. **Feel free to skip this section** and go to the next headline called "ESP8266 IP".

Having your ESP8266+FTDI Programmer connected to your computer, go to the ESPlorer IDE:



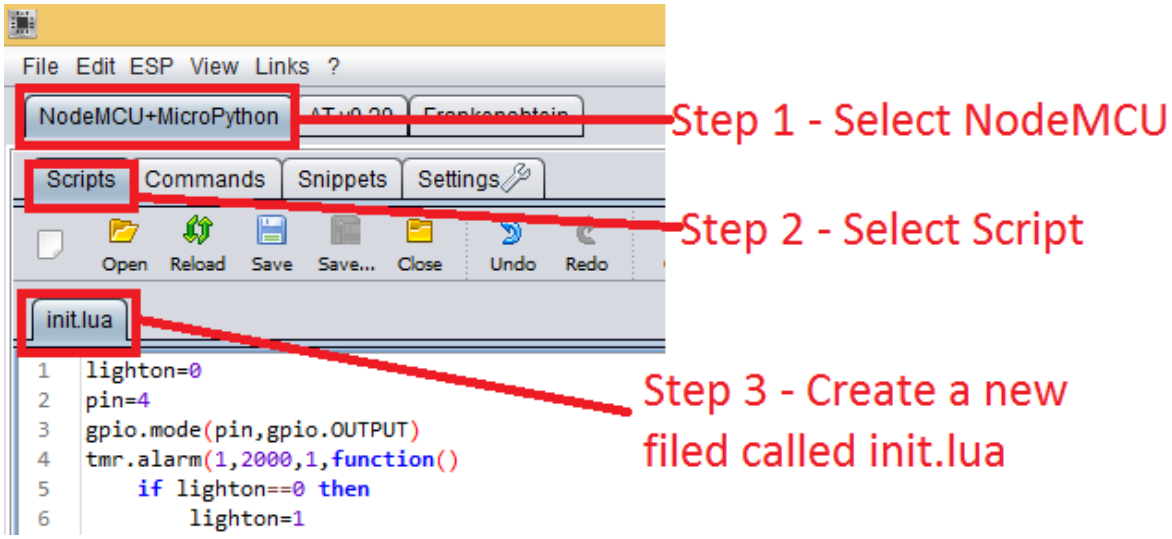
Look at the top right corner of your ESPlorer IDE and follow these instructions:

1. Press the Refresh button
2. Select the COM port for your FTDI programmer
3. Select 9600 as your baudrate
4. Click Open

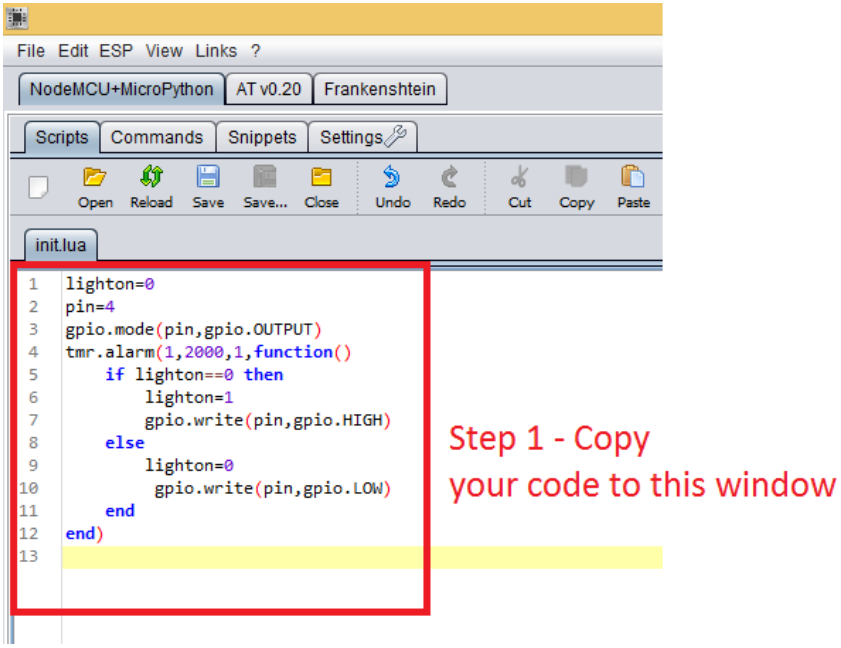


Then in the top left corner of your ESPlorer IDE, follow these instructions:

1. Select NodeMCU
2. Select Scripts
3. Create a new filled called “init.lua”



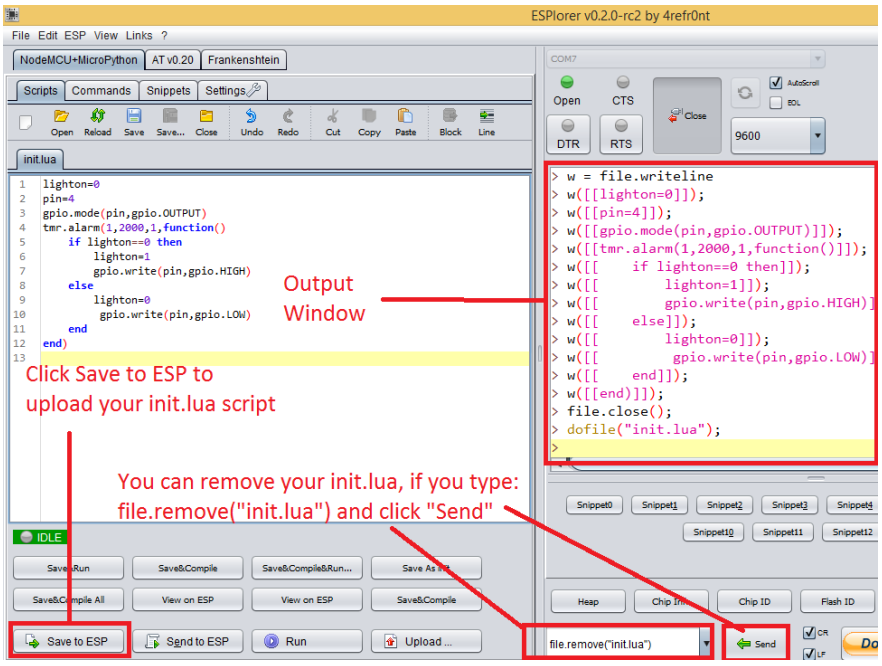
Copy your Lua script (which you created in the previous section) to the code window (as you can see in the figure below):



The next step is to save your code to your ESP8266!

At the left bottom corner click the button “Save to ESP”.

In your output window, it should start showing exactly which commands are being sent to your ESP8266 and it should look similar to the Figure below.

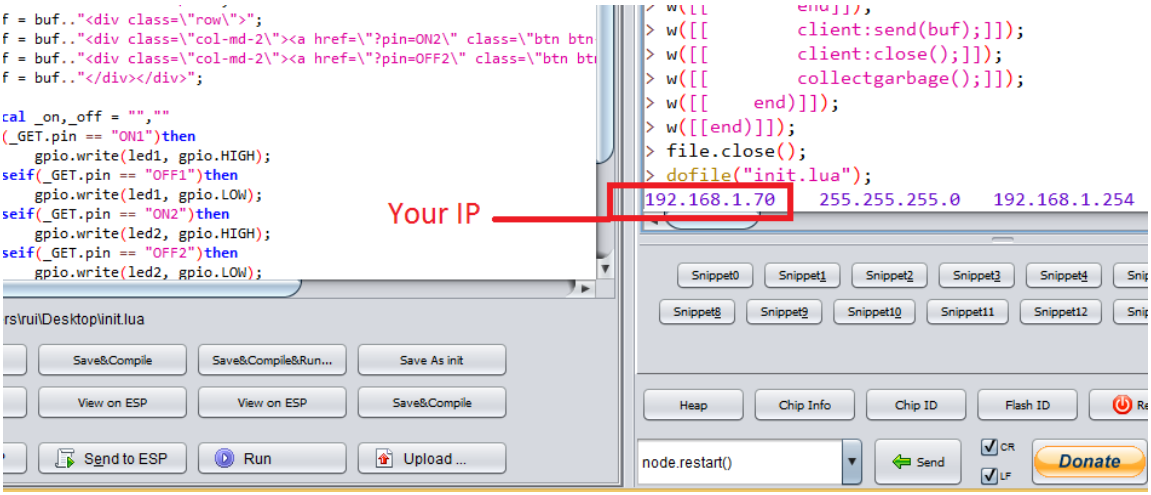


Note: You can easily delete the “init.lua” file from the ESP. Simply type *file.remove(“init.lua”)* and press the button “Send” (see Figure above). Or you can type the command *file.format()* to remove all the files saved in your ESP8266.

ESP8266 IP

After uploading your web server Lua script to your ESP8266 ,in your output window you're going to see 3 IP addresses.

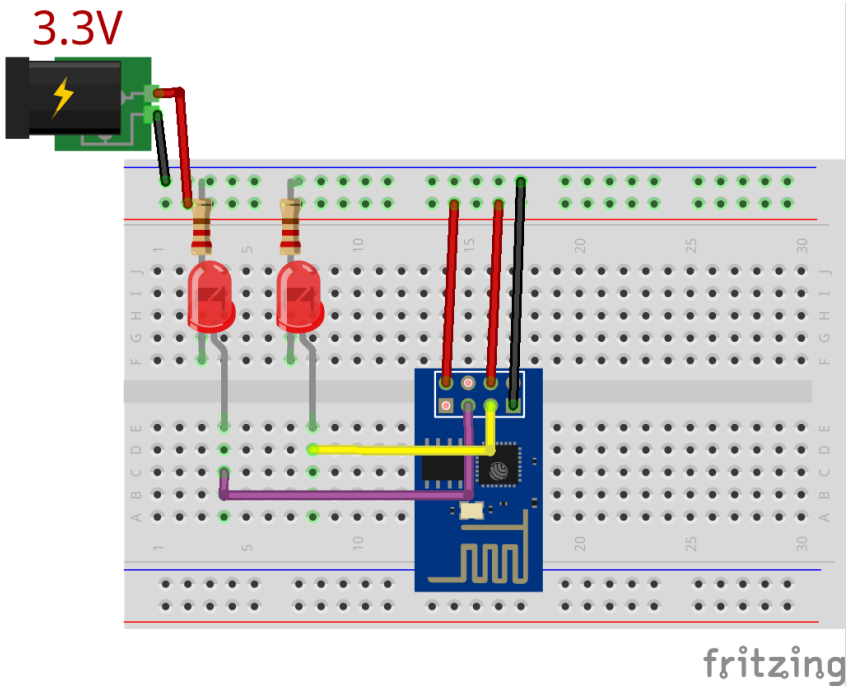
The IP that matters is the first one, in my case it's: 192.168.1.70. Your IP should be different, **save your ESP8266 IP** so you can access it later in this Unit.



Note: If your IP address doesn't appear in your output window you can send the command *print(wifi.sta.getip())* to print your ESP8266 IP.

Here's Your Final Circuit

After uploading your code to your ESP8266, follow the next schematics (you can use 220 ohm resistors for the LEDs).

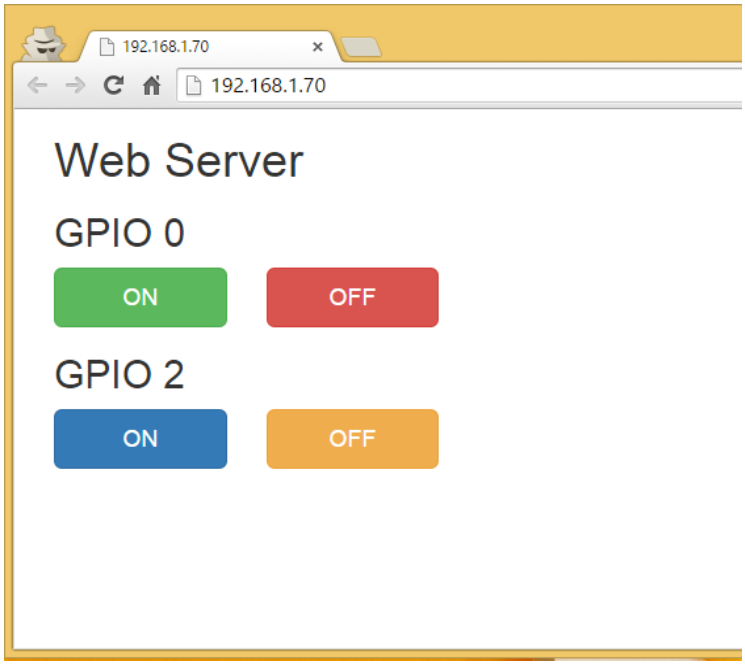


Accessing Your Web Server

Now follow the next instructions before accessing your web server:

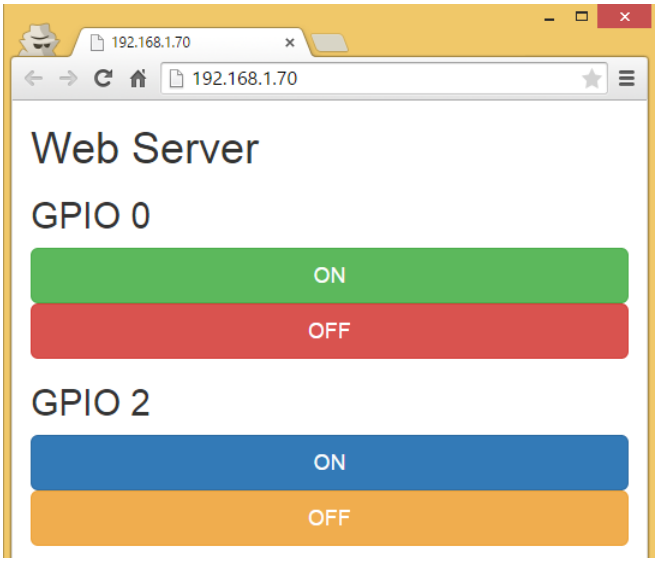
1. Restart your ESP8266 module
2. Open a browser
3. Type the IP address that you've previously saved (in my case: 192.168.1.70) in the URL bar

A web page like the one below should appear.



Note: In order to access your web server, you need to be connected to the same router that your ESP8266 is.

That was fun! Having a $4 WiFi module that can act as a web server and serves mobile responsive web pages is pretty amazing!



Making Your Web Server Password Protected

At this point your web server is running on your local network and anyone that is connected to your router can type the IP address of your ESP into their browser and access your web server.

To make your web server more secure let's add an authentication mechanism. After you implement this feature, when anyone wants to access your web server they need to enter a username and a password.

You only need to add those 6 lines of code below to your existing web server:

```
local _, _, auth = string.find(request, "%cAuthorization: Basic ([%w=\+\/]+)");--Authorization:
  if (auth == nil or auth ~= "dXNlcjpwYXNz")then --user:pass
        client:send("HTTP/1.0 401 Authorization Required\r\nWWW-Authenticate: Basic realm=\"ESP8266
        client:close();
        return;
end
```

Go to the following link and download the web server script with the authentication mechanism: https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/blob/master/Projects/ESP8266/web-server-authentication.lua

Encoding Your Username and Password

At this point if you upload the code I've mentioned in the preceding section, your username is *user* and your password is *pass*. I'm sure you want to change and customize this example with your own credentials.

Go to the following URL: https://www.base64encode.org. In the first field, type the following:

your_username:your_password

Note: You actually need to type the ":" between your username and your password.

In my example, I've entered *user:pass* (as you can see in the Figure below):

Then Press the green "Encode" button to generate your base64 encoded string. In my example is *dXNlcjpwYXNz*.

Copy your string and replace it in this line of the Lua Script that you downloaded.

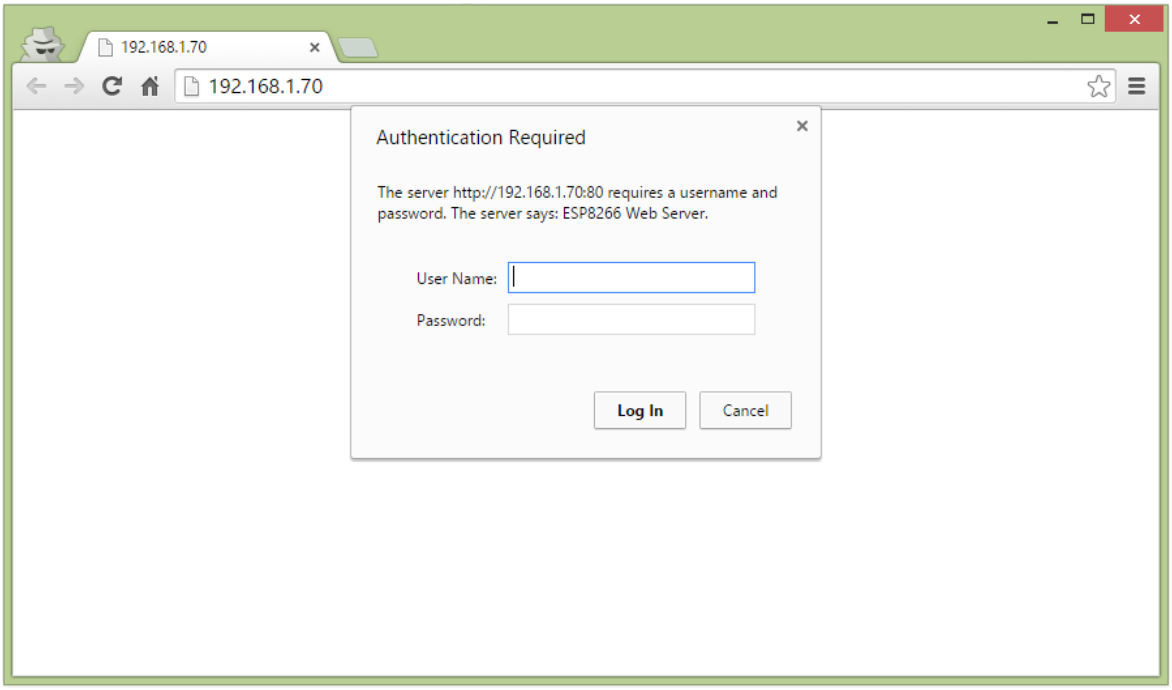
```
if (auth == nil or auth ~= "dXNlcjpwYXNz")then --user:pass
```

Uploading Your New Web Server to Your ESP

Now that you have your code ready, you need to upload your script to your ESP8266 like you did earlier in this unit.

After a successful upload. Access your ESP web server by entering your ESP IP address in your browser.

It should require that you enter your username and password in order to access your web server. As this is what you should see:



Taking It Further

I hope you're happy about seeing that LED turning on and off! I know that it's just an LED, but creating a web server just like you did is an extremely useful concept.

Controlling some house appliances may be more exciting than lighting up an LED. You can easily and immediately replace the LED with a new component that allows you to control any device that connects directly to the sockets on the wall. You have a few options...

Option #1 – PowerSwitch Tail II

The easiest route is to get yourself a PowerSwitch Tail II (www.powerswitchtail.com), which provides a safe way of dealing with high-voltage devices

The way this bulky component works is quite straightforward. Rather than connecting a house appliance directly to the wall, you connect it to the PowerSwitch Tail II which plugs into the wall.

The PowerSwitch Tail II has three pins that enable it to behave like a simple digital logic device. You connect the PowerSwitch Tail to an output GPIO of the ESP8266.

Your output pin will send a signal that's either HIGH or LOW. Whenever the signal is HIGH, there's a connection to the wall socket; when it's LOW, the connection is broken, as though the device were unplugged.

Here's how you should connect your PowerSwitch Tail II

PowerSwitch Tail II Pin Number	**Signal Name**	**ESP8266 Pins**
1	+in	GPIO 0 or GPIO 2
2	-in	GND
3	GND	Not used

Search for the PowerSwitch Tail II's instruction sheet for more details on how to wire it up.

Option #2 – Relay

There's another way to have your ESP8266 control a house appliance, but that method is more complicated. It requires a bit of extra knowledge and wariness because you're dealing with alternating current, and it involves relay modules.

I won't discuss this project in great detail, but here's a good tutorial: http://www.instructables.com/id/Web-Controlled-8-Channel-Powerstrip/?ALLSTEPS

The preceding link takes you to an Instructable that shows how to control sockets using a Raspberry Pi, but you can apply the same concepts to your ESP module.

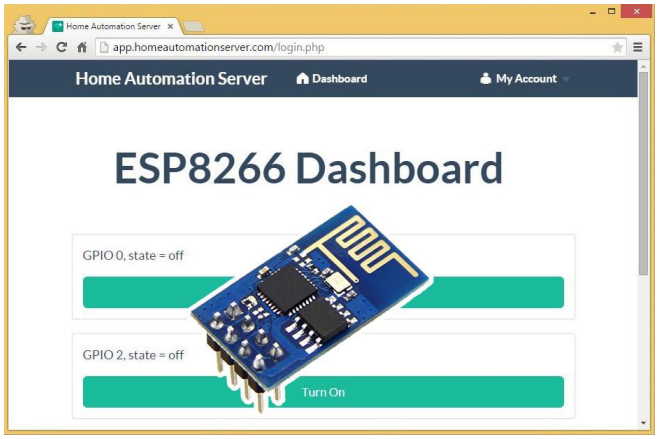
WARNING: Always be safe when dealing with high voltages, if you don't know what you're doing ask someone who does.

Controlling Your ESP8266 From Anywhere

The most common question I get about the ESP8266 WiFi Module is: "Is it possible to control my ESP8266 from anywhere in the world?".

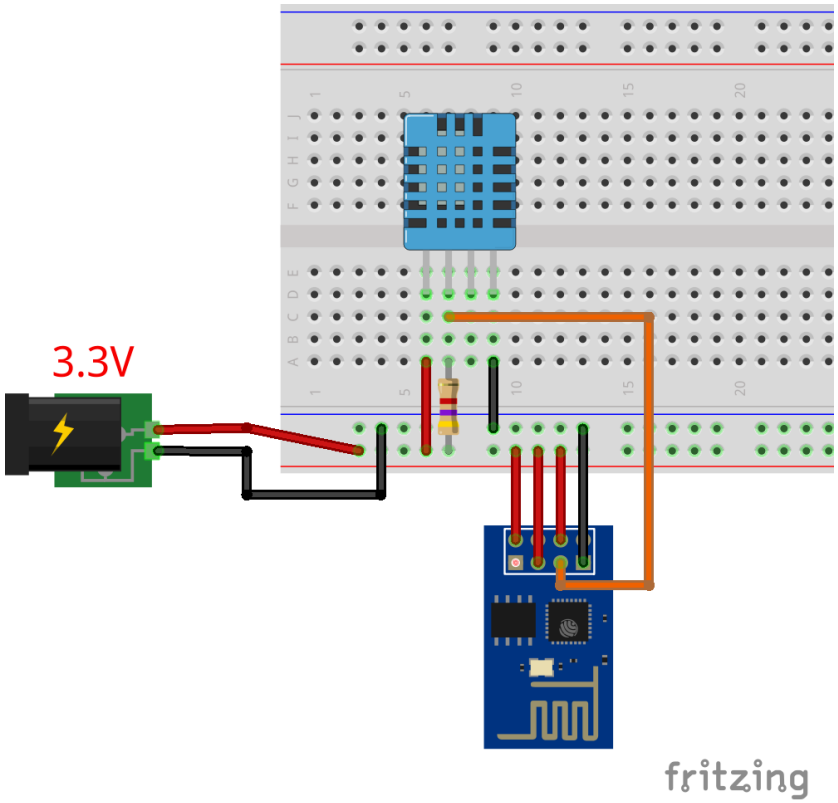
So I've decided to create a free platform that allows you to control your ESP8266 from anywhere.

Here's the complete tutorial: http://randomnerdtutorials.com/has-esp



Unit 7

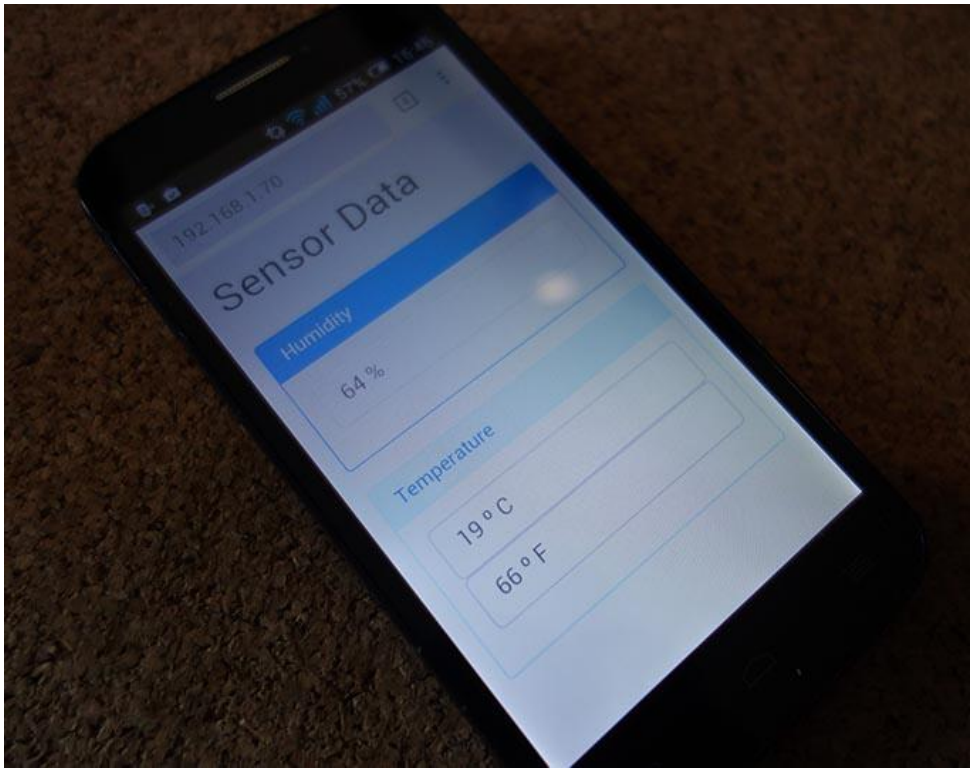
Displaying Temperature and Humidity on a Web Page



Displaying Temperature and Humidity on a Web Page

In this Unit you're going to create a web server with your ESP8266 that can be accessed with any device that has a browser. This will serve a web page with the current temperature and humidity. You'll use a DHT11 sensor to measure the temperature and humidity.

You can access your web server from your laptop, smartphone, tablet and so on! Here's the final result of this project!

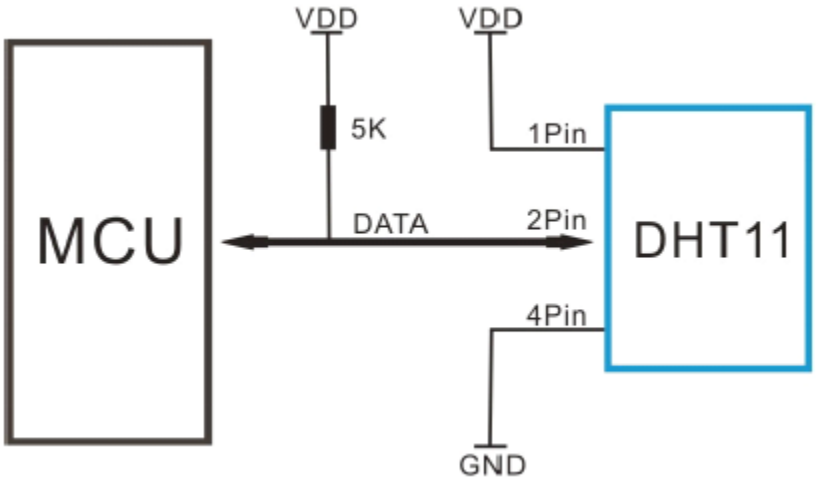


Using DHT11 Temperature and Humidity Sensor

The DHT11 sensor is very popular among the Arduino Tinkerers. The DHT sensors are relative cheap sensors for measuring temperature and humidity.

These sensors inside contain a chip that does analog to digital conversion and spits out a digital signal with the temperature and humidity.

With any microcontroller (MCU) these signals are fairly easy to read.



DHT11 specs:

- Range: 20-90%
- Absolute accuracy: ±5%
- Repeatability: ±1%
- Long term stability: ±1% per year
- Price: $1 to $5

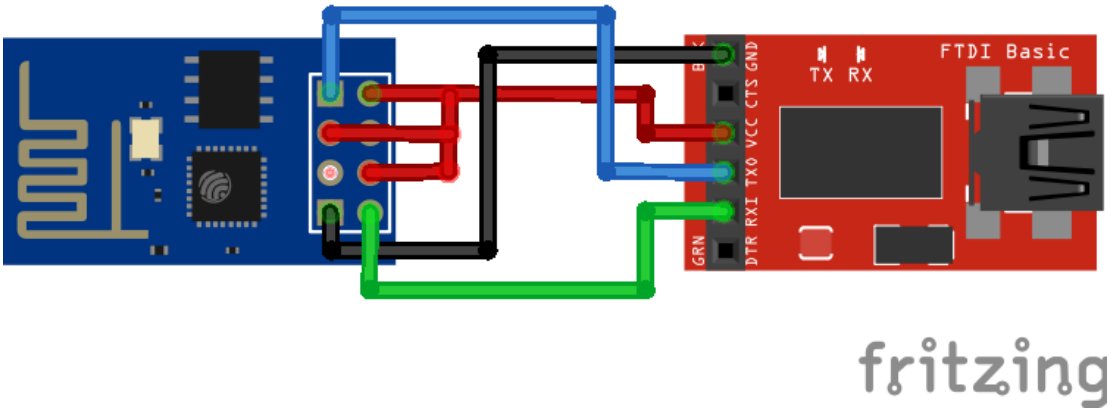
Pins:

- VCC (3.3V)
- Data OUT
- Don't connect
- GND

Visit the following link to buy a DHT11 Humidity and Temperature sensor on eBay: http://randomnerdtutorials.com/ebay-dht11

Schematics

To upload code to your ESP8266, you should connect your ESP to your FTDI Programmer like the figure below:



Writing Your init.lua Script

Below is the script to create a web page that displays the temperature and humidity from a DHT11 sensor. You can download the Lua script in the following link: https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/blob/master/Projects/ESP8266/maindht.lua

Let's look at how this code works.

The snippet of code below starts by setting the mode of your ESP to a station. Then it configures the module with your own credentials (network name and password). You actually need to replace that second line with your credentials, so that your ESP8266 can connect to your network.

We use a function called *tmr.delay(5000)* to delay 5 seconds your code. That allows time for the module to connect with the network.

The *print()* function in line 4 prints your ESP8266 IP address in the output window of the ESPlorer IDE (you need that IP to access your web server).

Next, you tell the sketch which sensor is connected to the module (in this case, the dht11), and you also create a few variables to store the data.

```
wifi.setmode(wifi.STATION)
wifi.sta.config("YOUR_NETWORK_NAME","YOUR_NETWORK_PASSWORD")
print(wifi.sta.getip())
tmr.delay(5000)

sensorType="dht11"              -- set sensor type dht11 or dht22
pin = 4 --  data pin, GPIO2
humi=0
temp=0
fare=0
bimb=1
```

Then, you create a function called *ReadDHT11()* that does exactly what it sounds like. It reads the humidity and temperature data from your sensor and stores it in the right variables. It also prints that data in your ESPlorer IDE output window. You can delete lines 30 to 32. They are just for debugging purposes.

```
function ReadDHT11()
    dht=require("dht")
    dht.read(pin)
    chck=1
    h=dht.getHumidity()
    t=dht.getTemperature()
    if h==nil then h=0 chck=0 end
    if sensorType=="dht11"then
        humi=h/256
        temp=t/256
    else
        humi=h/10
        temp=t/10
    end
    fare=(temp*9/5+32)
    print("Humidity:    "..humi.."%")
    print("Temperature: "..temp.." deg C")
    print("Temperature: "..fare.." deg F")
    -- release module
    dht=nil
    package.loaded["dht"]=nil
end
ReadDHT11()
```

You also need to create a *tmr.alarm()* function that will run the *ReadDHT11()* function every 5 seconds. Remember, that function will retrieve the current humidity and temperature readings every time it is run.

Next, in line 41 below, we create the web server on port 80. In line 46 below, we call the function *conn:send()*.

```
38
39  tmr.alarm(1,5000, 1, function() ReadDHT11() bimb=bimb+1 if bimb==5
40
41  srv=net.createServer(net.TCP) srv:listen(80,function(conn)
42      conn:on("receive",function(conn,payload)
43     --print(payload) -- for debugging only
44      --generates HTML web site
45      conn:send()
46      conn:on("sent",function(conn) conn:close() end)
47      end)
48  end)
```

Inside the function *conn:send()* you add your HTML page (see code below). It's just a basic web page that uses the Bootstrap framework. Learn more about the Bootstrap framework: http://getbootstrap.com/.

```
45      conn:send('HTTP/1.1 200 OK\r\nConnection: keep-alive\r\nCache-Control: privat
46          <!DOCTYPE HTML>\
47          <html><head><meta name="viewport" content="width=device-width, initial-sc
48          <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
49          <meta http-equiv="refresh" content="6">\
50          </head><div class="container">\
51          <h1>Sensor Data</h1></br><div class="row">\
52          <div class="col-md-4"><div class="panel panel-primary"><div class="panel-
53          </div><div class="panel-body">\
54          <div class="form-group form-group-lg"><input type="text" class="form-cont
55          </div></div></div></div>\
56          <div class="col-md-4"><div class="panel panel-info"><div class="panel-hea
57          </div><div class="panel-body">\
58          <div class="form-group form-group-lg"><input type="text" class="form-cont
59          <input type="text" class="form-control" value="'..fare..' º F">\
60          </div></div></div></div></div></div></html>')
```

The above Figure will display the temperature and humidity in a nice mobile responsive web page.

Downloading Your dht.lua Script

The library that we're going to use to read the data that comes from the DTH11 sensor was written by Javier Yanez and based on the script of Pig Fly from the ESP8266 forum. Thanks to them, it's easy to use the DHT11 sensor!

You can go to this link and download the "dht.lua" file: https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/blob/master/Projects/ESP8266/dht.lua

Uploading dht.lua and init.lua

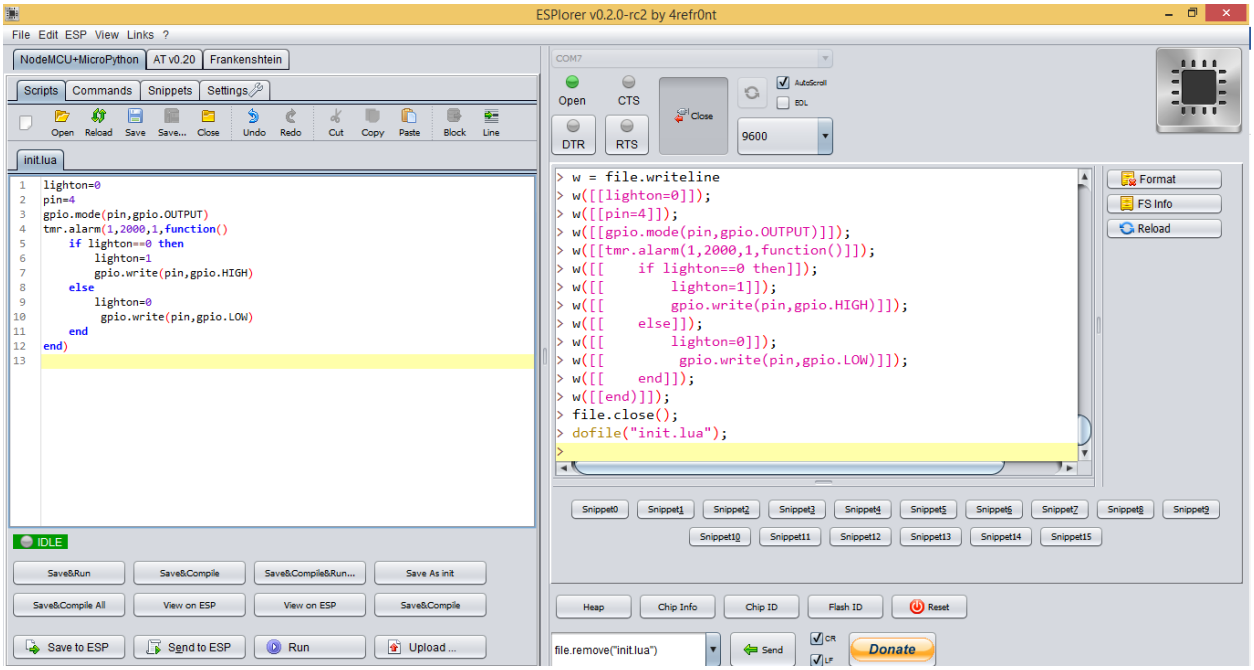
Now you need to upload the code you just wrote and downloaded.

1. First upload the "dht.lua"
2. Then upload the "init.lua"

Note: You need to upload the files in that order, because the "init.lua" needs the "dht.lua" file to work properly.

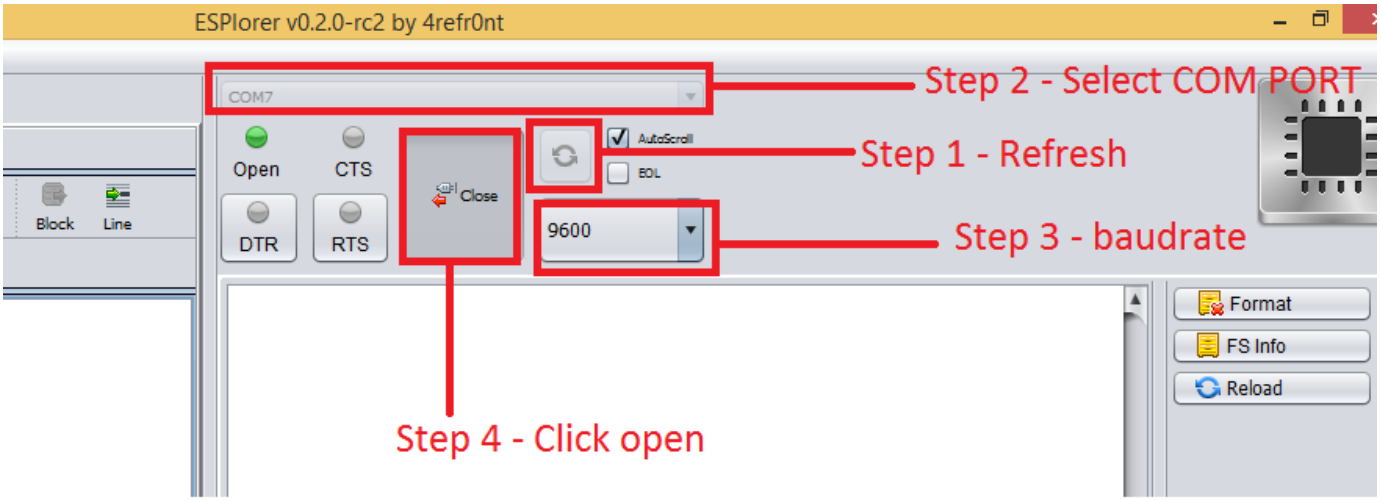
If you've followed Unit 3 or 6, you already know how to upload code to your ESP8266. **Feel free to skip this section** and go to the next headline called "ESP8266 IP".

Having your ESP8266+FTDI Programmer connected to your computer, go to the ESPlorer IDE:



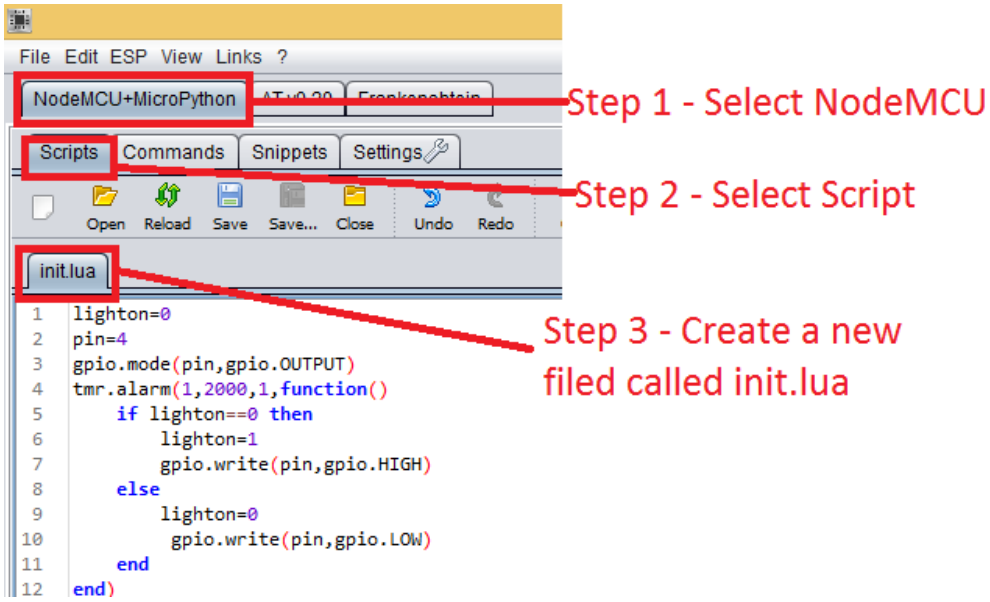
Look at the top right corner of your ESPlorer IDE and follow these instructions:

1. Press the Refresh button
2. Select the COM port for your FTDI programmer
3. Select 9600 as your baudrate
4. Click Open



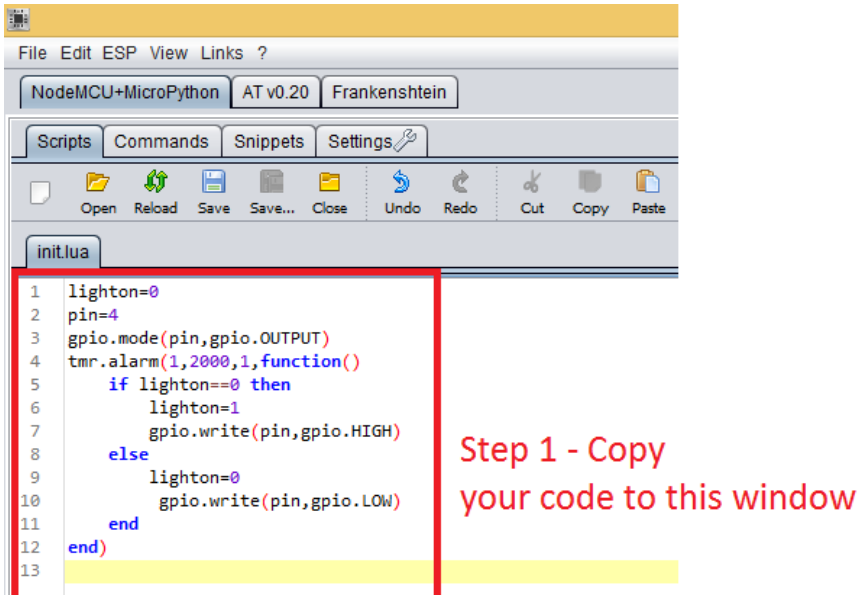
Then in the top left corner of your ESPlorer IDE, follow these instructions:

1. Select NodeMCU
2. Select Scripts
3. Create a new filed called “init.lua”




```
1  lighton=0
2  pin=4
3  gpio.mode(pin,gpio.OUTPUT)
4  tmr.alarm(1,2000,1,function()
5      if lighton==0 then
6          lighton=1
7          gpio.write(pin,gpio.HIGH)
8      else
9          lighton=0
10          gpio.write(pin,gpio.LOW)
11     end
12 end)
```

Copy your Lua script (which you created in the previous section) to the code window (as you can see in the figure below):

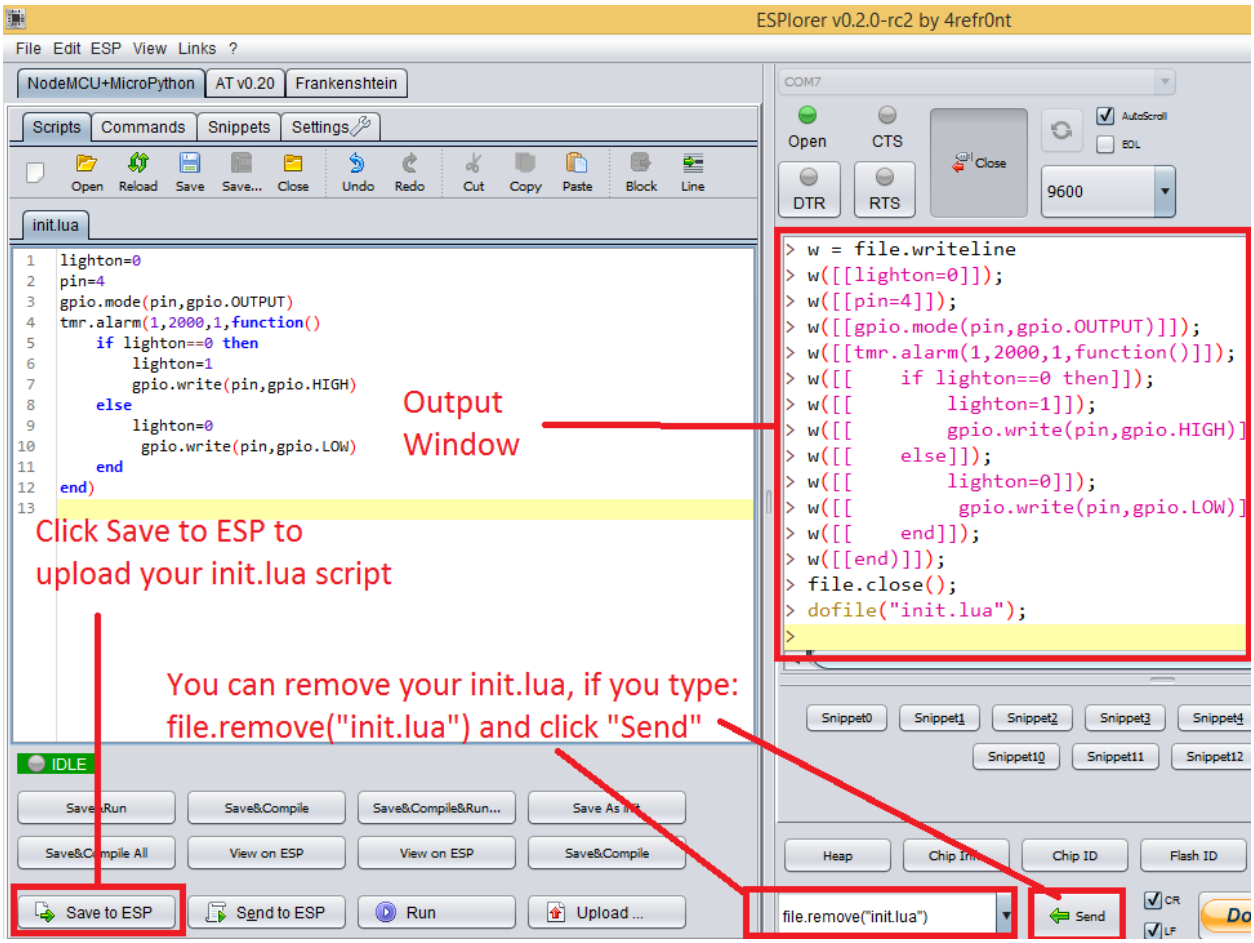



```
1  lighton=0
2  pin=4
3  gpio.mode(pin,gpio.OUTPUT)
4  tmr.alarm(1,2000,1,function()
5      if lighton==0 then
6          lighton=1
7          gpio.write(pin,gpio.HIGH)
8      else
9          lighton=0
10          gpio.write(pin,gpio.LOW)
11     end
12 end)
13
```

The next step is to save your code to your ESP8266!

At the left bottom corner click the button “Save to ESP”.

In your output window, it should start showing exactly which commands are being sent to your ESP8266 and it should look similar to the Figure below.

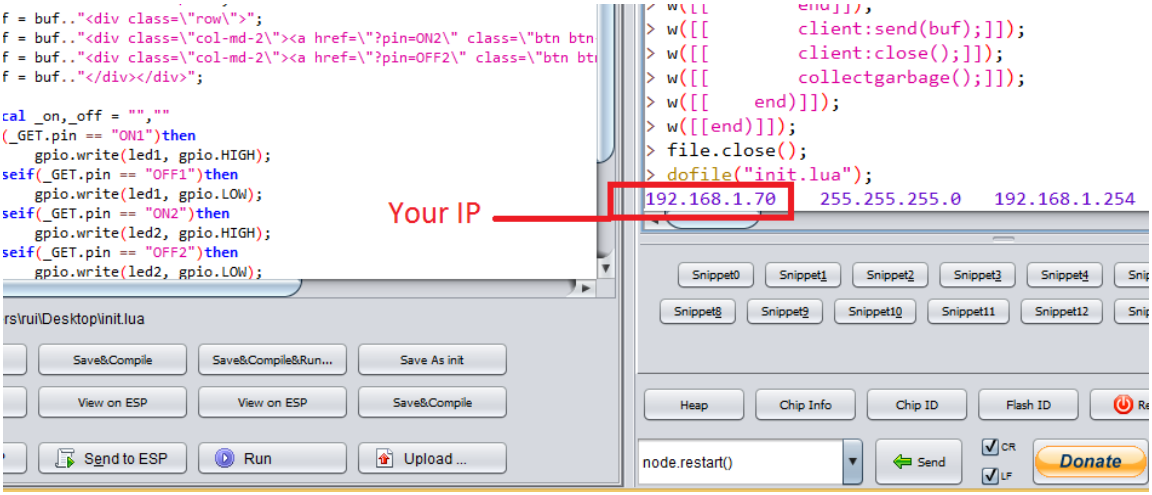


Note: You can easily delete the “init.lua” file from the ESP. Simply type *file.remove(“init.lua”)* and press the button “Send” (see Figure above). Or you can type the command *file.format()* to remove all the files saved in your ESP8266.

ESP8266 IP

After uploading your web server Lua script to your ESP8266 ,in your output window you're going to see 3 IP addresses.

The IP that matters is the first one, in my case it's: 192.168.1.70. Your IP should be different, **save your ESP8266 IP** so you can access it later in this Unit.

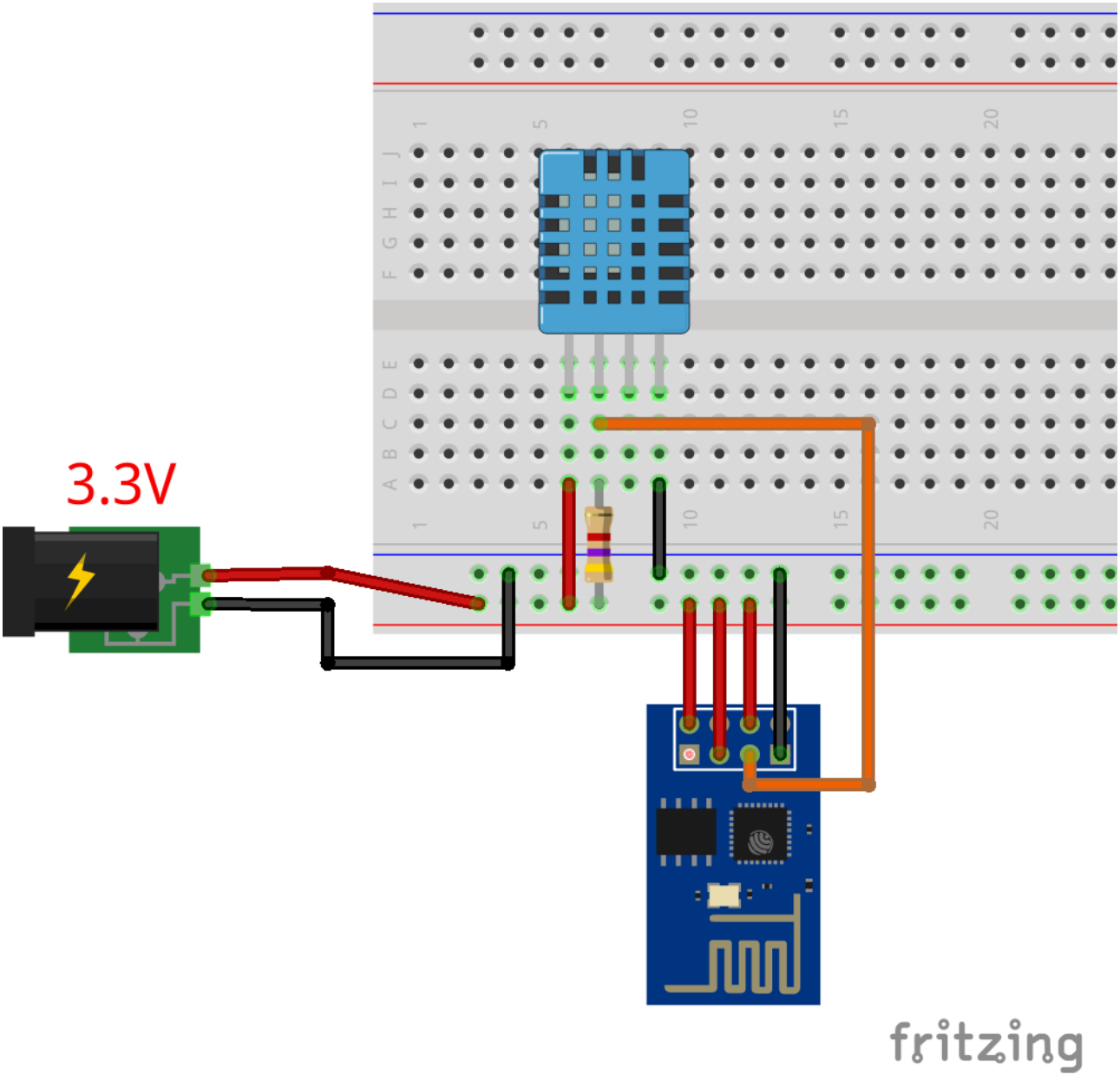


Note: If your IP address doesn't appear in your output window you can send the command *print(wifi.sta.getip())* to print your ESP8266 IP.

Here's Your Final Circuit

After uploading your code to your ESP8266, follow the next schematics. You need to use a 4700 ohm resistor with your DHT11 sensor as pull-up resistor.

Warning: The 3.3V power supply should supply 250mA to ensure that your circuit works.

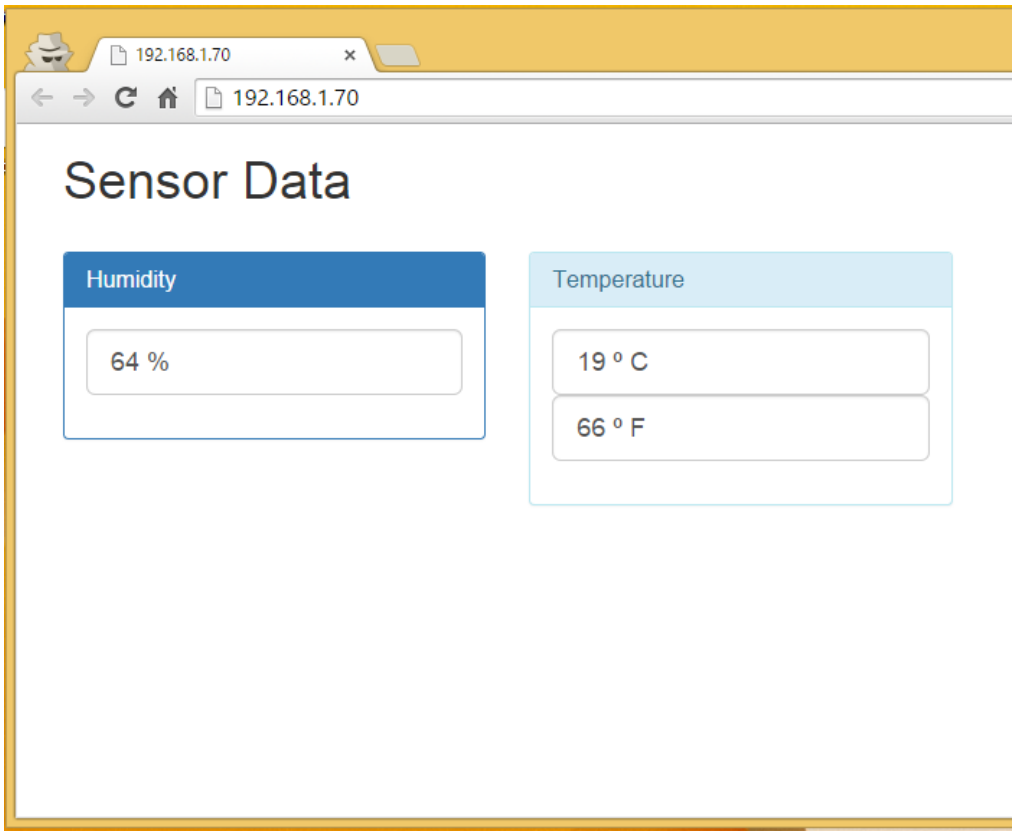


Accessing Your Web Server

Follow the next instructions before accessing your web server:

1. Restart your ESP8266 module
2. Open a browser
3. Type the IP address that you've previously saved (in my case: 192.168.1.70) in the URL bar

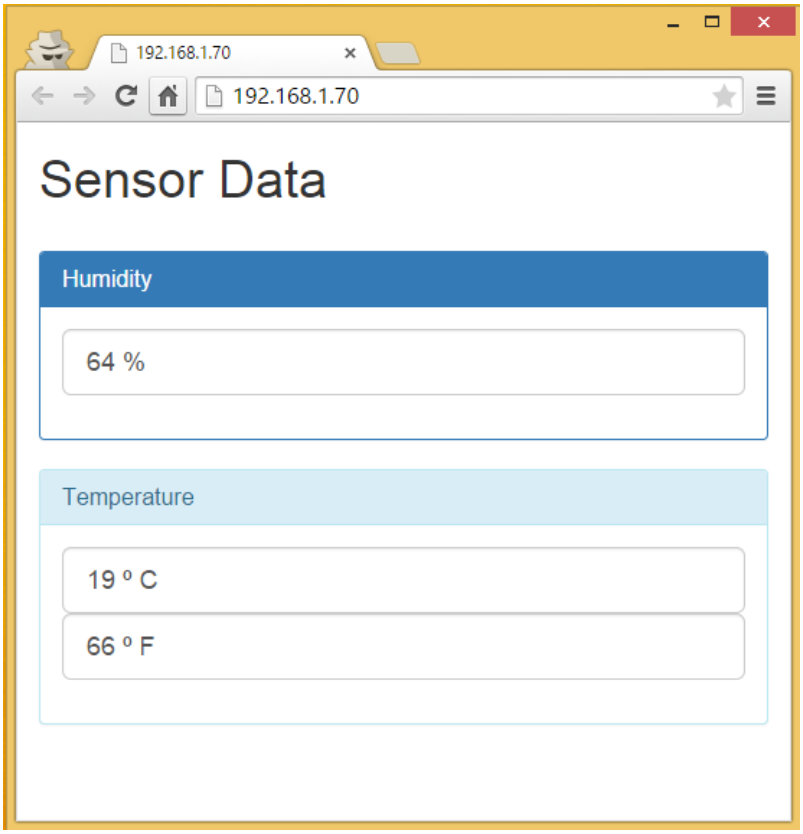
And a web page like the one below should appear.



Note: In order to access your web server, you need to be connected to the same router that your ESP8266 is.

Pretty cool, huh??! Feel free to modify or add your own twist to this code. And remember this page updates automatically every 6 seconds.

This page is mobile responsive so it looks great in any device!



Taking it Further

The ESP8266 doesn't offer analog pins. That's very limiting if you want to add analog sensors to your project...

So what can you do to overcome that problem? I have an idea for you.

You know how to establish a serial communication with the ESP8266 (Unit 2). You can attach a bunch of sensors to your Arduino and connect your Arduino to your ESP8266.

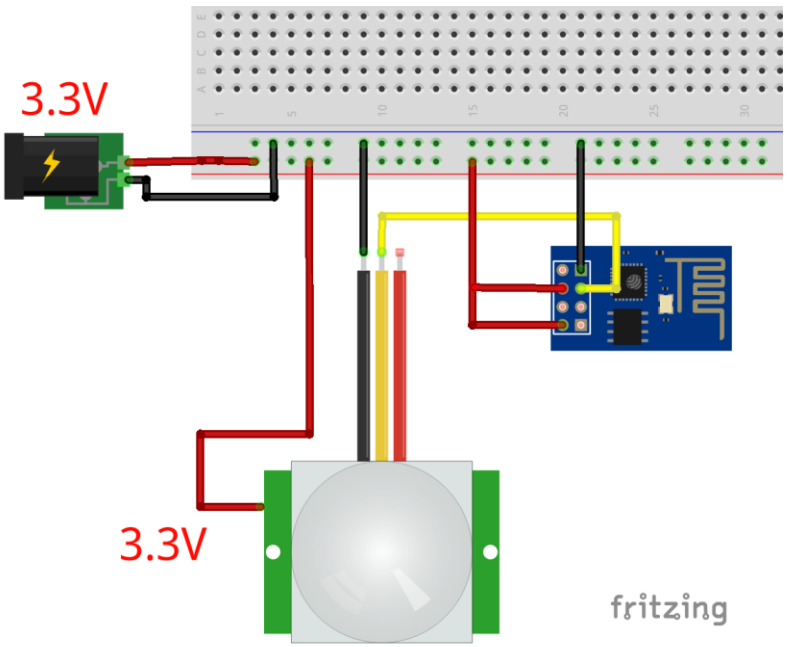
Your Arduino sends all the sensors readings to your ESP8266 via serial communication, then your ESP8266 grabs all the data and displays it in your neat web server!

Here's a good tutorial on how to build a weather station with an Arduino: http://www.instructables.com/id/Arduino-weather-station/?ALLSTEPS

You can re-purpose some of the pieces in that Instructable to read your sensors with an Arduino.

Unit 8

Email Notifier with ESP8266 and PIR Motion Sensor

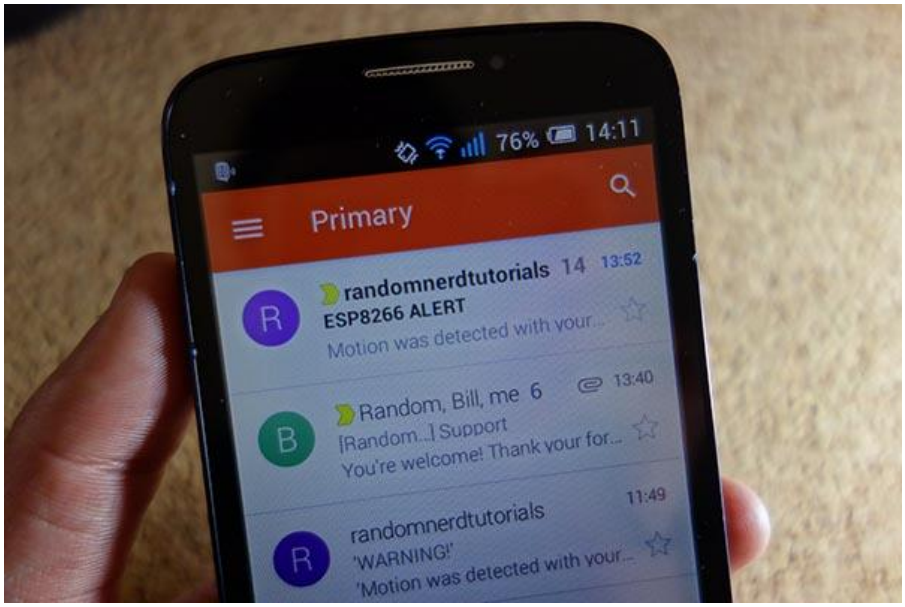


Email Notifier with ESP8266 and PIR Motion Sensor

In this Unit you're going to create an email notifier with an ESP8266 and a cheap PIR Motion sensor. You can set this project in a strategic place and when someone walks by, it sends you an email alert. I'll basically show how to build a home surveillance system for $6.

In order to accomplish this task you have to sign up for one free service called IFTTT which stands for "If This Then That". IFTTT is a platform that gives you creative control over dozens of products and apps. You can make apps work together. For example when you send a request to IFTTT, it triggers a recipe that sends you an email alert.

Here's a spoiler of the end result:

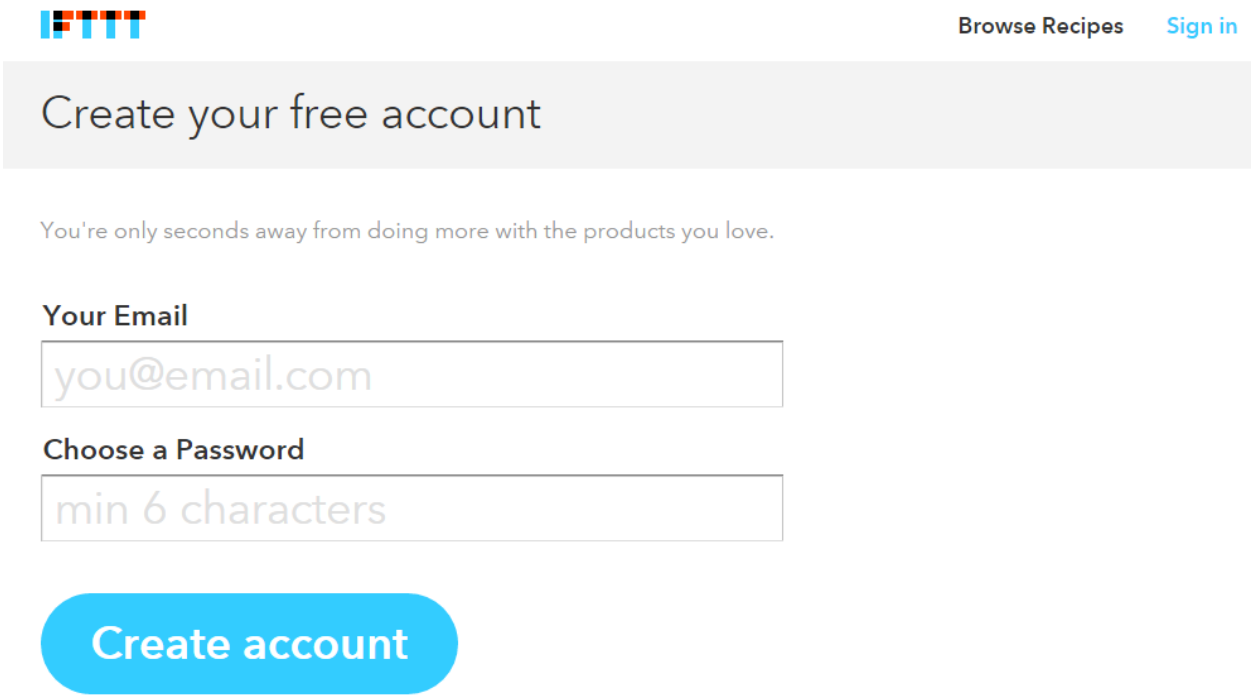


Creating Your IFTTT Account

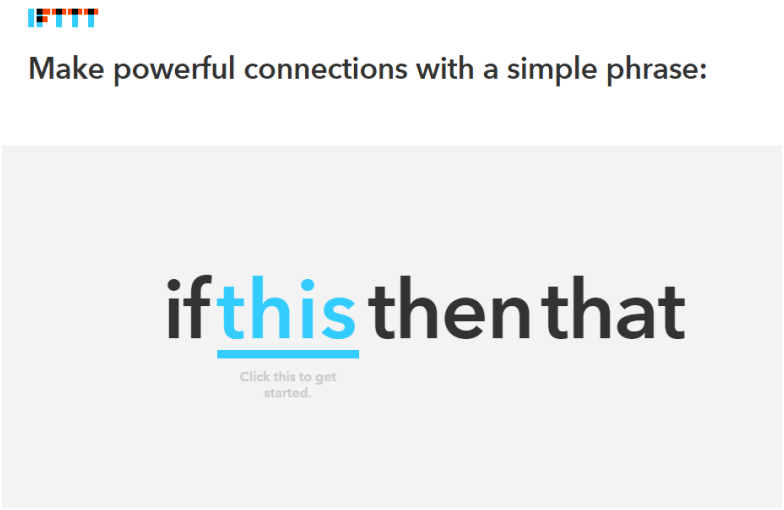
Creating an account on IFTTT is free!

Go the official site: https://ifttt.com/ and click the “Sign Up” button in the middle of the page.

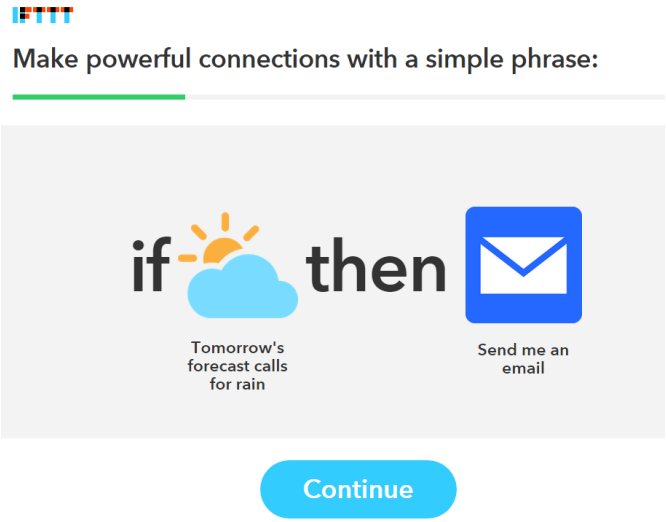
Complete the form with your personal information (see Figure below) and create your account.



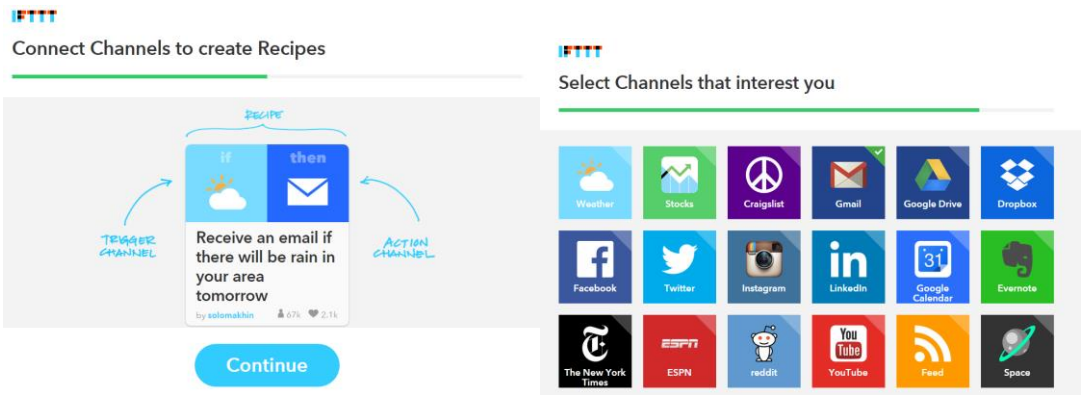
After creating your account, follow their getting started tutorial by clicking the word “this”:



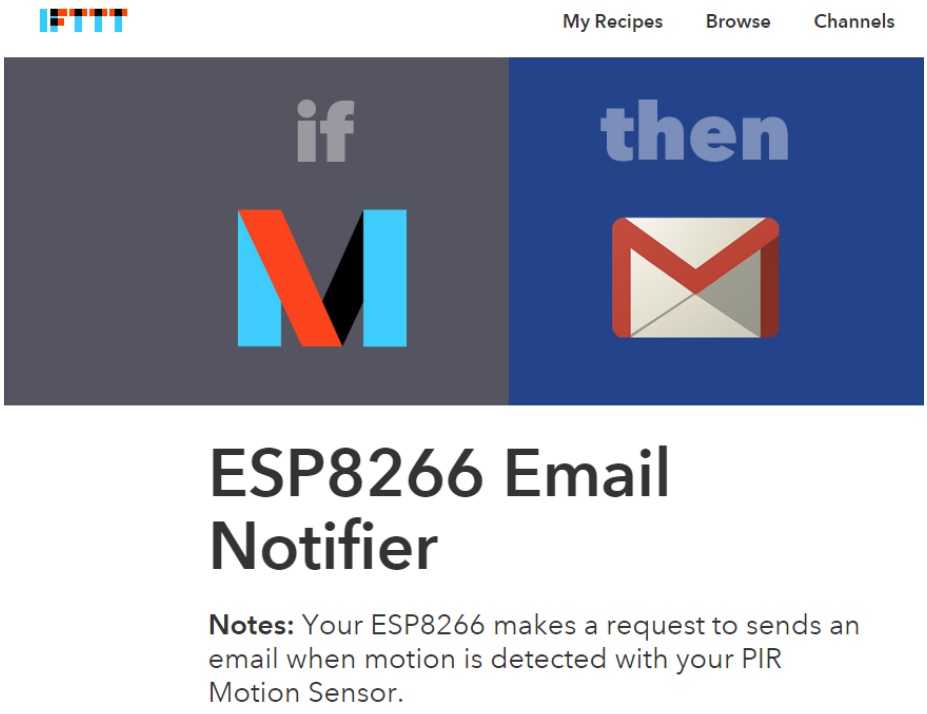
Then wait a few seconds and click the big blue button “Continue”:



Click “Continue” a few more times to finish their introductory tutorial:

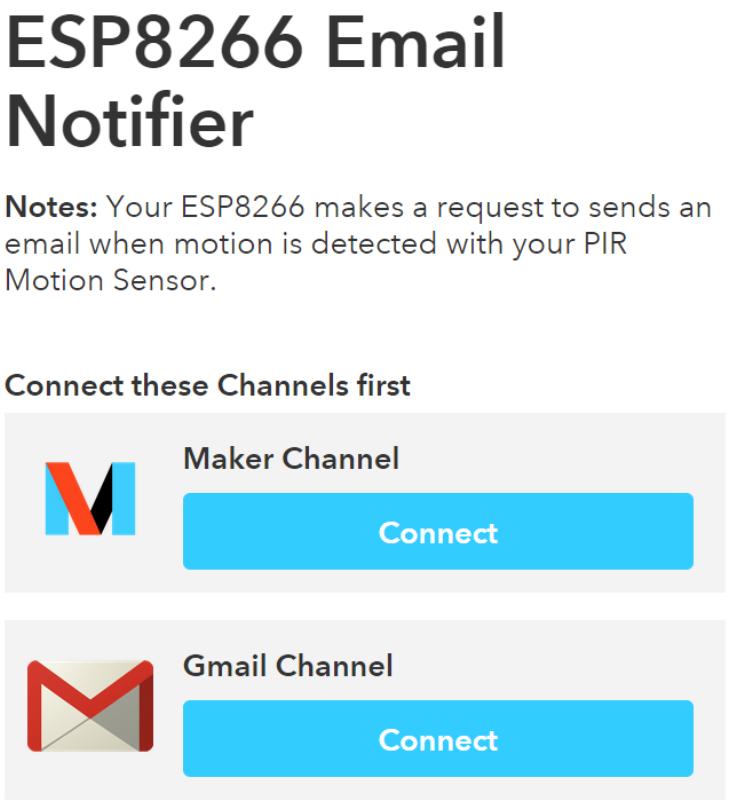


I've created a recipe that you can use that integrates perfectly in this project. If you're logged in at IFTTT and you open this URL: https://ifttt.com/recipes/315426-esp8266-email-notifier you can use my recipe instantly.

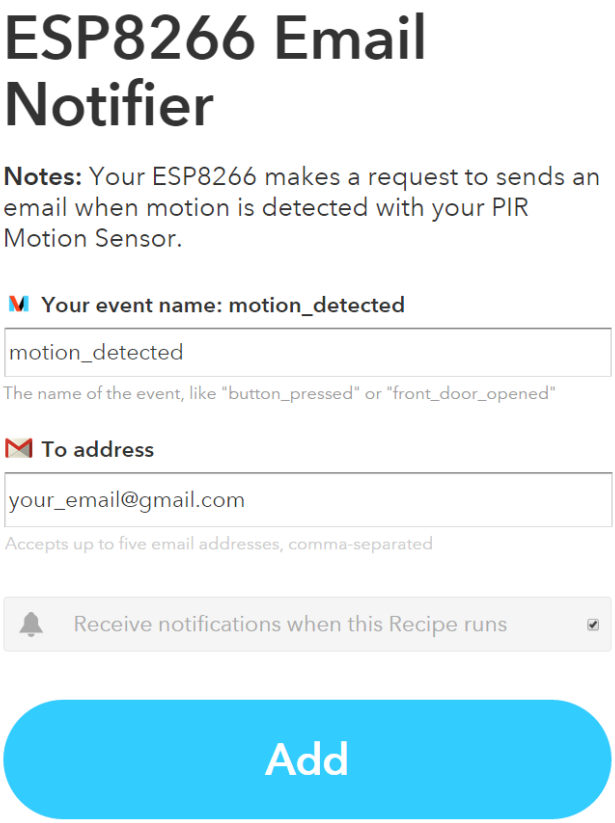


Next scroll down that page and follow these instructions to make it work for you:

1. Connect your account to the Maker Channel
2. Then connect your account to your Gmail Channel



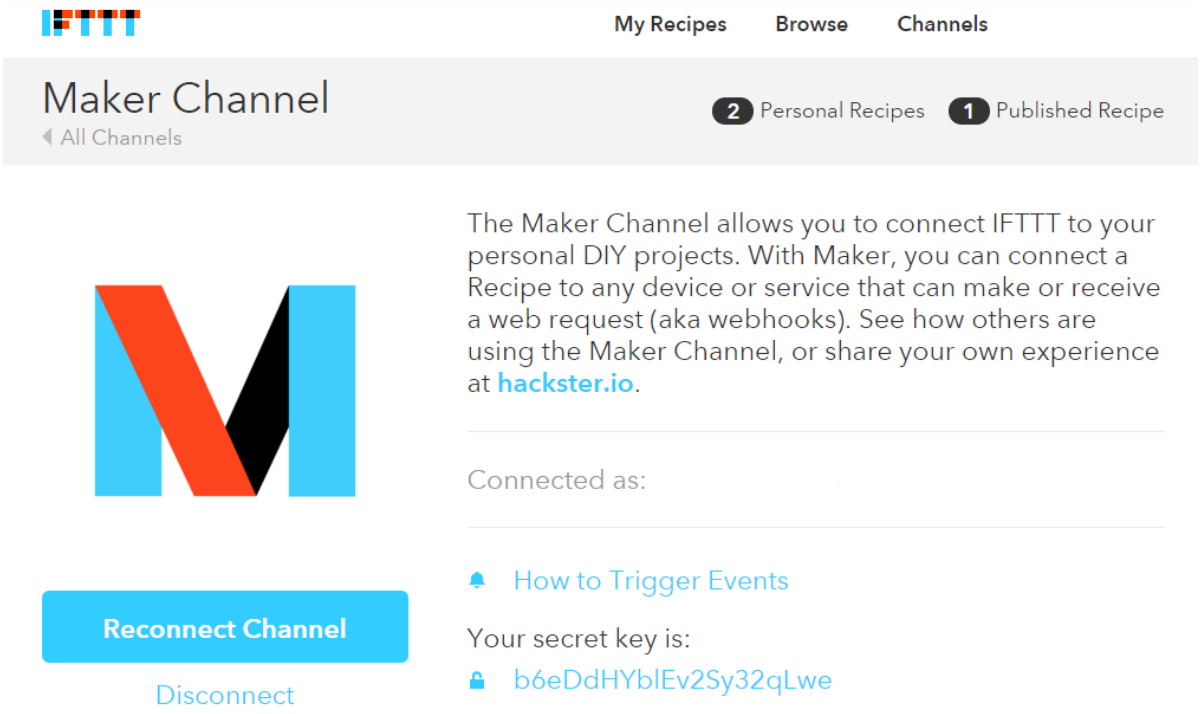
A new page loads (see Figure below) when you finish connecting your account to Make Channel and Gmail Channel.



Fill the recipe with your own information. Follow these instructions:

1. Type “motion_detected” in your event name
2. Replace with your email the to address field (that’s the email address where you’re going to receive the email alerts)
3. Press the “Add” button

Go to this URL: https://ifttt.com/maker. Copy you secret key to a safe place (you’ll need them later in this Unit). In my example my secret key is: *b6eDdHYblEv2Sy32qLwe*



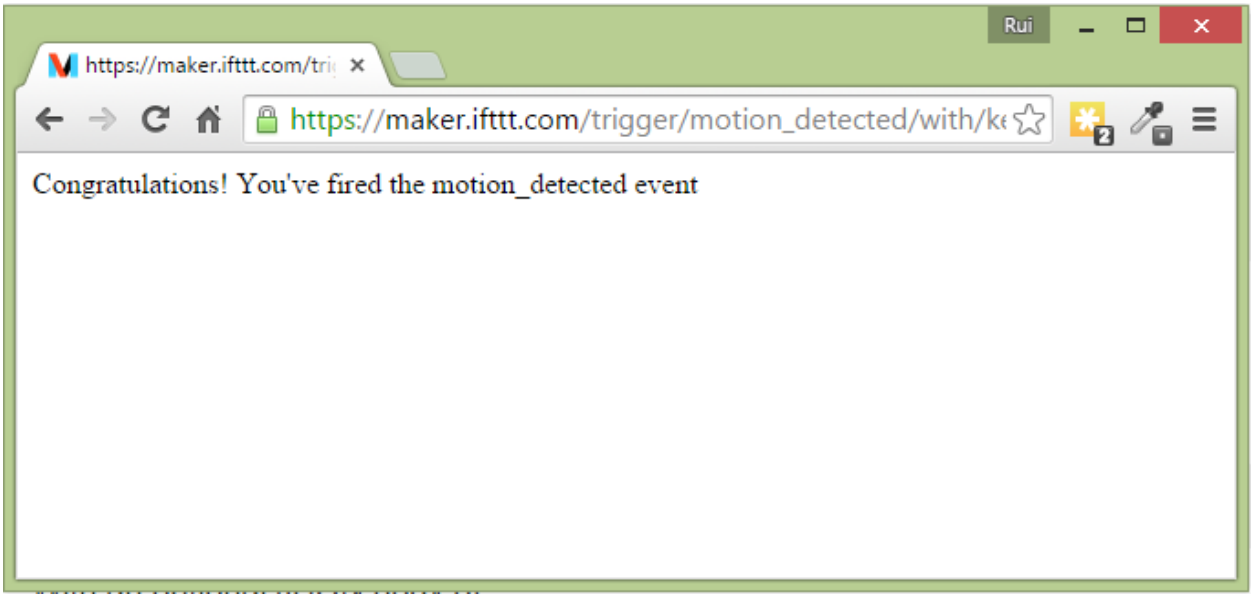
Let's test if your request is working properly. Replace YOUR_API_KEY from the following URL:

https://maker.ifttt.com/trigger/motion_detected/with/key/**YOUR_API_KEY**

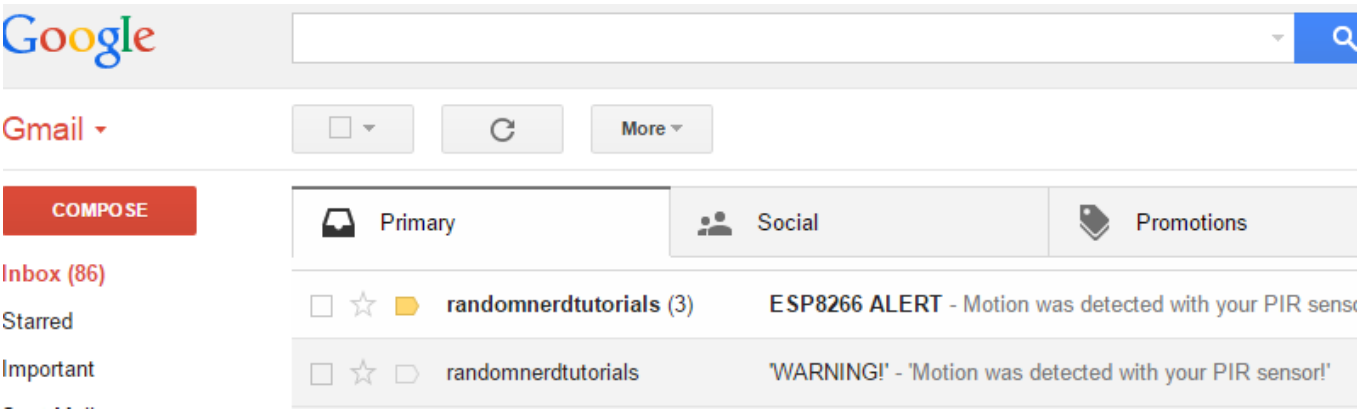
With your API KEY:

https://maker.ifttt.com/trigger/motion_detected/with/key/**b6eDdHYblEv2Sy32qLwe**

Open your URL with your API KEY in your browser.



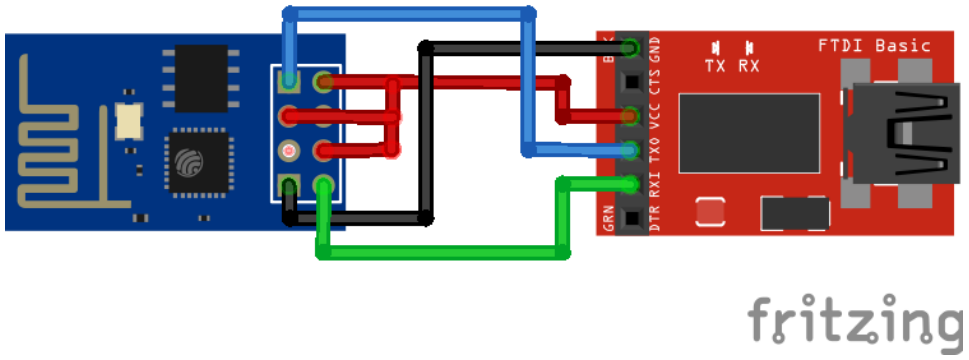
You should see something similar to the preceding Figure. Then go to your email and a new email should be there!



If you got the email in your inbox, lucky you! Most things don't work right at the first time. If you didn't receive the email, check your spam folder or re-check your IFTTT recipe. Now let's make your ESP8266 send emails for you.

Schematics

To upload code to your ESP8266, you should connect your ESP8266 to your FTDI Programmer like the figure below:



Writing Your init.lua Script

Below is the script to create your email notifier project. You can download the Lua script in the following link: https://github.com/RuiSantosdotme/Random-Nerd-Tutorials/blob/master/Projects/ESP8266/email_notifier_ifttt.lua

The snippet of code below starts by setting the mode of your ESP8266 to a station. Then you configure your ESP8266 with your own credentials (network name and password). You actually need to replace that second line with your credentials, so your ESP8266 can connect to your network.

Next, you create one variable (*pin*) which refers to GPIO 2 and define it as *INTERRUP*. That's where you're going to attach your PIR Motion sensor.

```
wifi.setmode(wifi.STATION)
wifi.sta.config("YOUR_NETWORK_NAME","YOUR_NETWORK_PASSWORD")
pin = 4
gpio.mode(pin, gpio.INT)
```

For this project we need to use an *INTERRUPT* that occurs when the PIR Motion sensor goes from LOW to HIGH, so we use the event "*up*" in the *gpio.trig(pin, 'up', onChange)* function. So when motion is detected it executes the function called *onChange()* which creates a TCP connections to IFTTT and makes your HTTP POST request.

```
function onChange ()
    -- A simple http client
    print('Motion Detected')
    conn = nil
    conn=net.createConnection(net.TCP, 0)
    conn:on("receive", function(conn, payload) end)
    conn:connect(80,"maker.ifttt.com")
    conn:on("connection", function(conn, payload)
        conn:send("POST /trigger/motion_detected/with/key/YOUR_API_KEY HTTP/1.1\r\n
    conn:close()
    print('Email Sent')
end
gpio.trig(pin, 'up', onChange)
```

You have to replace the line 14 with **your URL** from https://ifttt.com/maker.

Replace line 14:

conn:send("POST **/trigger/motion_detected/with/key/YOUR_API_KEY** HTTP/1.1\r\nHost: maker.ifttt.com\r\nConnection: keep-alive\r\nAccept: */*\r\n\r\n") end)

With your API KEY:

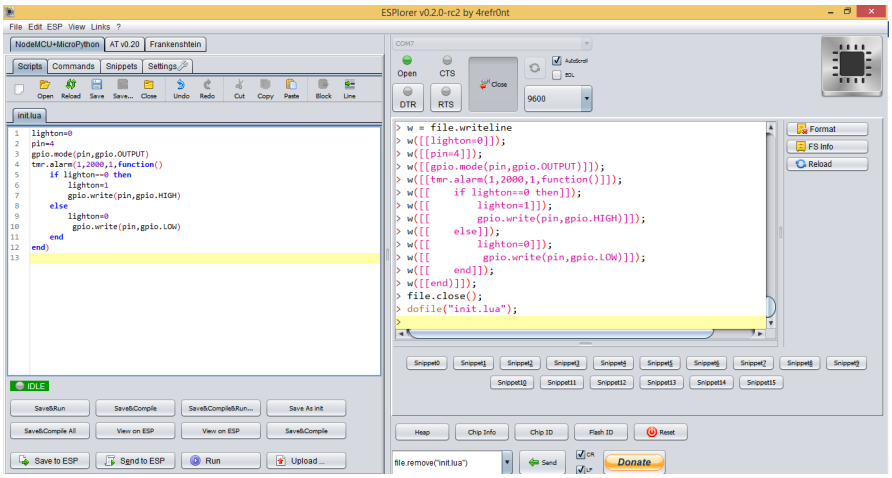
conn:send("POST
/trigger/motion_detected/with/key/b6eDdHYblEv2Sy32qLwe
HTTP/1.1\r\nHost: maker.ifttt.com\r\nConnection: keep-alive\r\nAccept:
/\r\n\r\n") end)

Uploading Code

Now you need to upload the code you just wrote to your ESP8266 (your file should be named as “init.lua”).

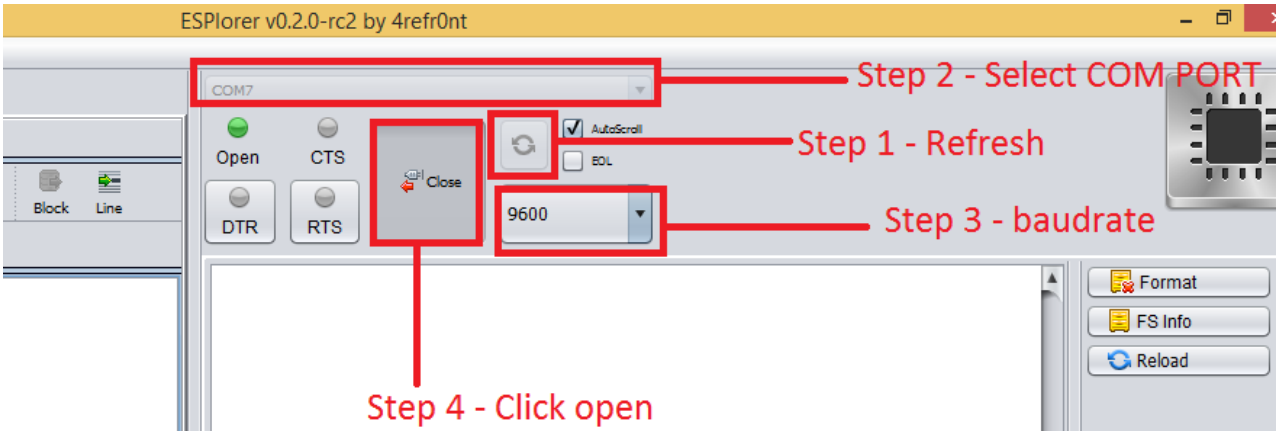
If you’ve followed Unit 3, 6 or 7, you already know how to upload code to your ESP8266. **Feel free to skip this section** and go to the next headline called “Modifying PIR Motion Sensor to Work at 3.3V”.

Having your ESP8266+FTDI Programmer connected to your computer, go to the ESPlorer IDE:



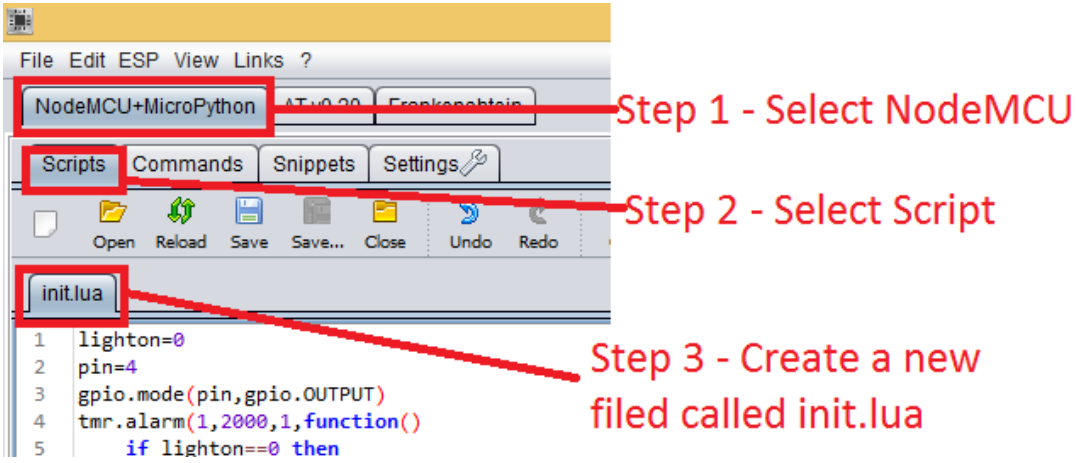
Look at the top right corner of your ESPlorer IDE and follow these instructions:

5. Press the Refresh button
6. Select the COM port for your FTDI programmer
7. Select 9600 as your baudrate
8. Click Open

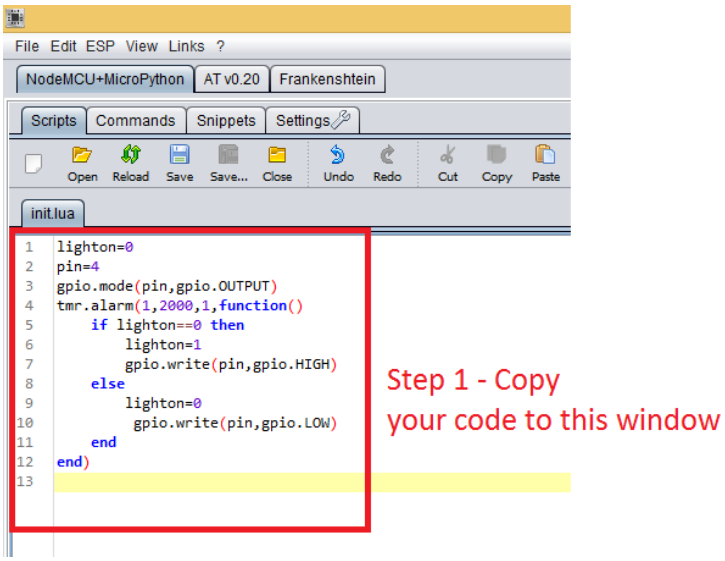


Then in the top left corner of your ESPlorer IDE, follow these instructions:

4. Select NodeMCU
5. Select Scripts
6. Create a new filled called “init.lua”



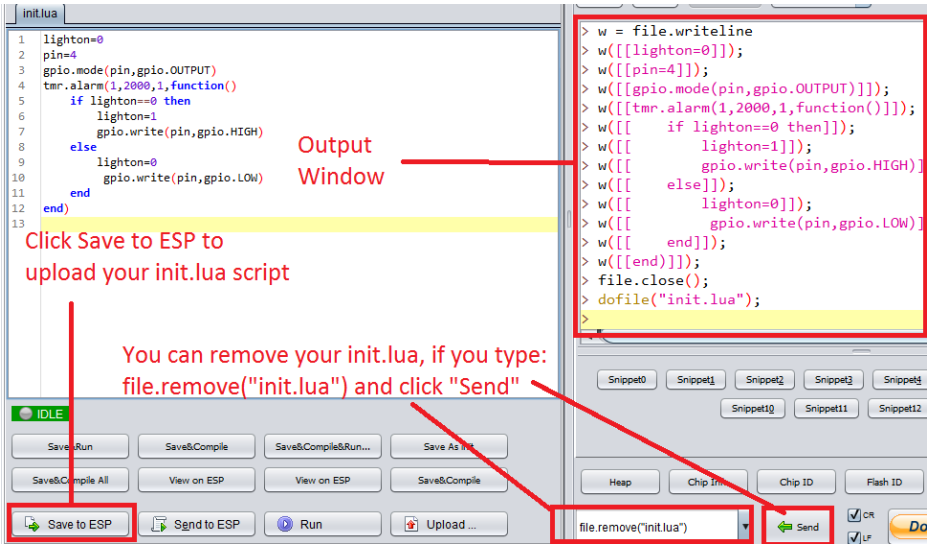
Copy your Lua script (which you created in the previous section) to the code window (as you can see in the figure below):



The next step is to save your code to your ESP8266!

At the left bottom corner click the button “Save to ESP”.

In your output window, it should start showing exactly which commands are being sent to your ESP8266 and it should look similar to the Figure below.



Note: You can easily delete the “init.lua” file from the ESP. Simply type *file.remove(“init.lua”)* and press the button “Send” (see Figure above). Or

you can type the command *file.format()* to remove all the files saved in your ESP8266.

Modifying PIR Motion Sensor to Work at 3.3V

This PIR Motion sensor is a $2 sensor that is used to detect movement from humans or pets. You can read a guide on my website on how to use this sensor with an Arduino.

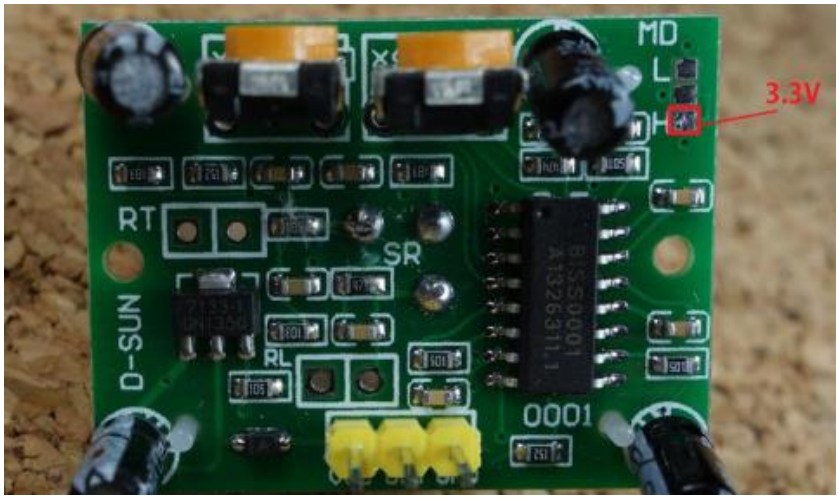
By default this module runs at 5V, but it has an on board voltage regulator that drops that voltage to 3.3V. With a quick online search I've found a blog post that explains how you can bypass the voltage regulator and use this module at 3.3V, which is exactly what I needed.

Where to buy one?

You can go to this link: http://randomnerdtutorials.com/ebay-pir to purchase one of these modules on eBay for less than $2.

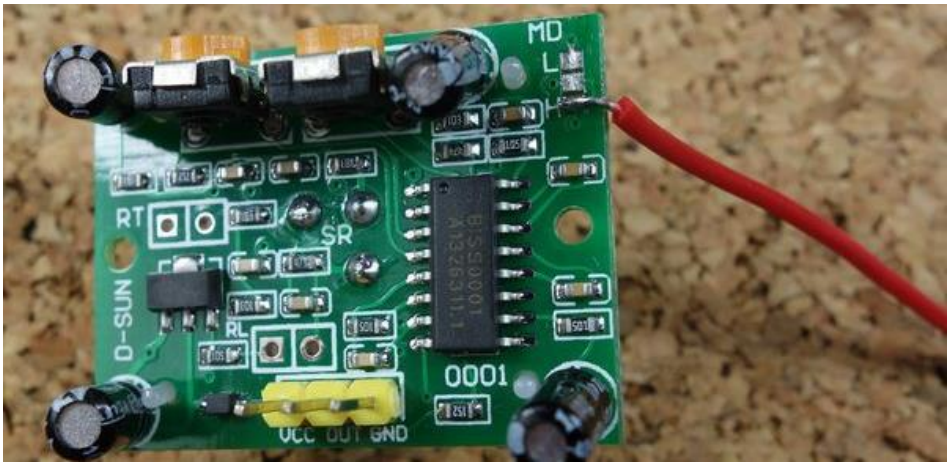
Before soldering

Some of these modules come with pins soldered in that top right corner, so you don't have to solder anything. You would simply connect a jumper wire to that pin that is highlighted in red (see Figure below). With my particular sensor I had to solder a small wire.



After soldering

Here's how it looks, now if you supply 3.3V through that red wire your module works at 3.3V.

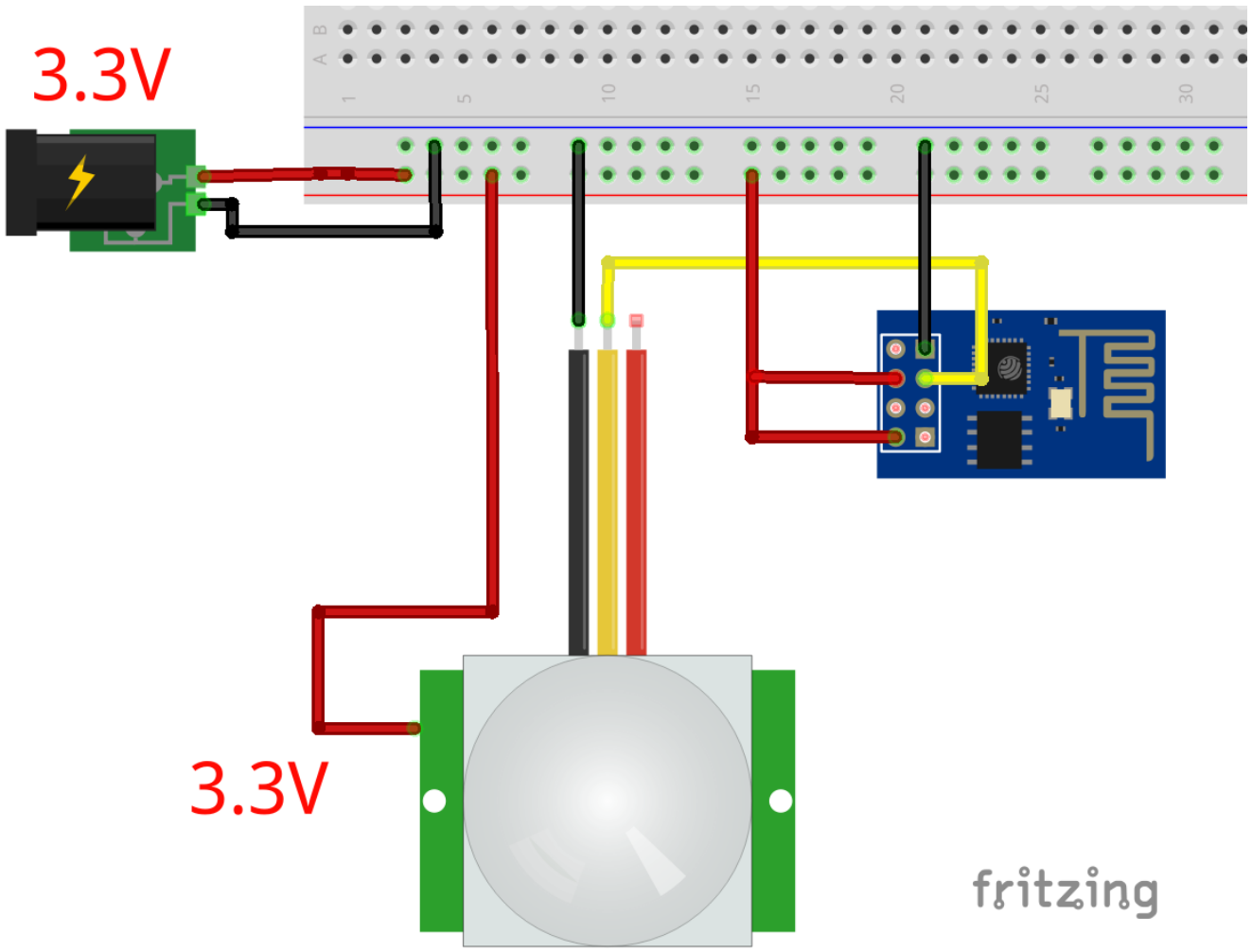


Testing

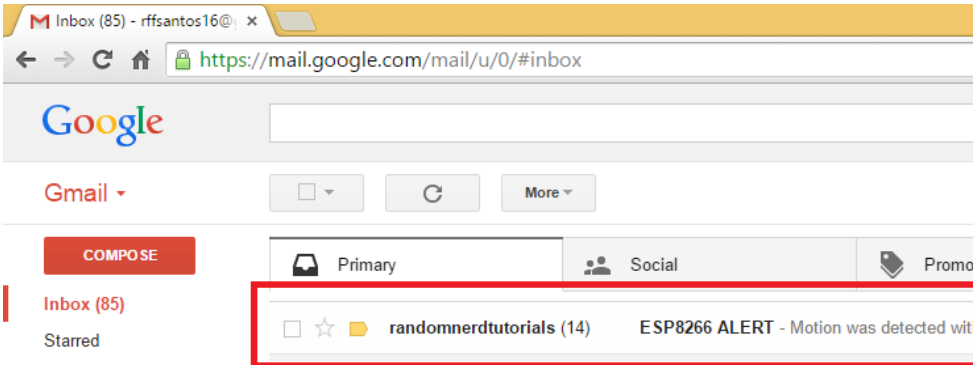
You can test this sensor with an Arduino using this code. The only difference now is that you can power this module with the 3.3V pin of the ESP8266!

Here's Your Final Circuit

This circuit is very quick to assemble, simply follow the schematics:



When you move your hand in front of the PIR Motion Sensor, or when someone passes in front of your sensor, an email is immediately sent to your inbox! How cool is that?



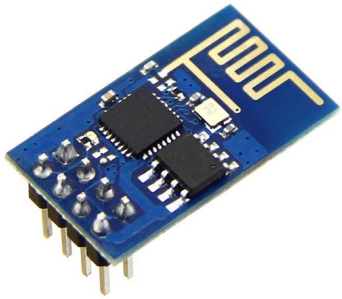
Taking It Further

The IFTTT platform is very powerful and you can integrate your ESP8266 projects with more than 200 apps. Here's a list of apps that you can now connect your ESP to: https://ifttt.com/channels.

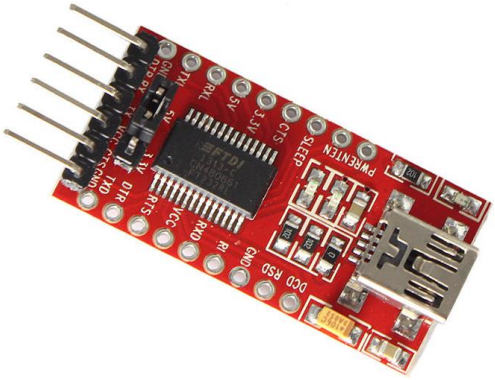
Recommended Tools

I thought it would be helpful to create this section to share the tools and modules I personally use for all these projects.

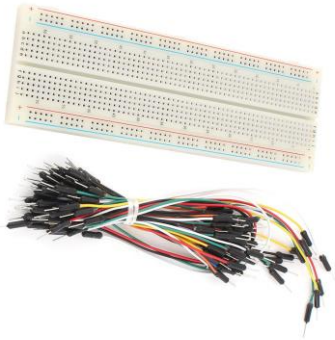
Visit this link to buy ESP8266 on eBay:
http://randomnerdtutorials.com/ebay-esp8266



Visit this link to buy an FTDI Programmer on eBay:
http://randomnerdtutorials.com/ebay-ftdi-programmer



Visit this link to buy a breadboard on eBay:
http://randomnerdtutorials.com/ebay-breadboard



Visit this link to buy a DHT11 Humidity and Temperature sensor on eBay:
http://randomnerdtutorials.com/ebay-dht11

Visit this link to buy a PIR Motion Sensor on eBay:
http://randomnerdtutorials.com/ebay-pir

Final Thoughts

Congratulations for completing this eBook!

If you followed all the projects presented in this eBook you now have the knowledge to build your Home Automation system using the ESP8266.

Let's see the key things that you've accomplished. You know how to:

- Create a password protected web server to control any output (Unit 6)
- Create a web page do display sensor data (Unit 7)
- Build an email alert system (Unit 8)

Now feel free to add multiple ESP8266s to your projects and build up on the snippets of code presented in this eBook to fit your own projects!

I hope you had fun following all these projects! If you have something that you would like to share let me know in the Facebook group (Join the Facebook group here).

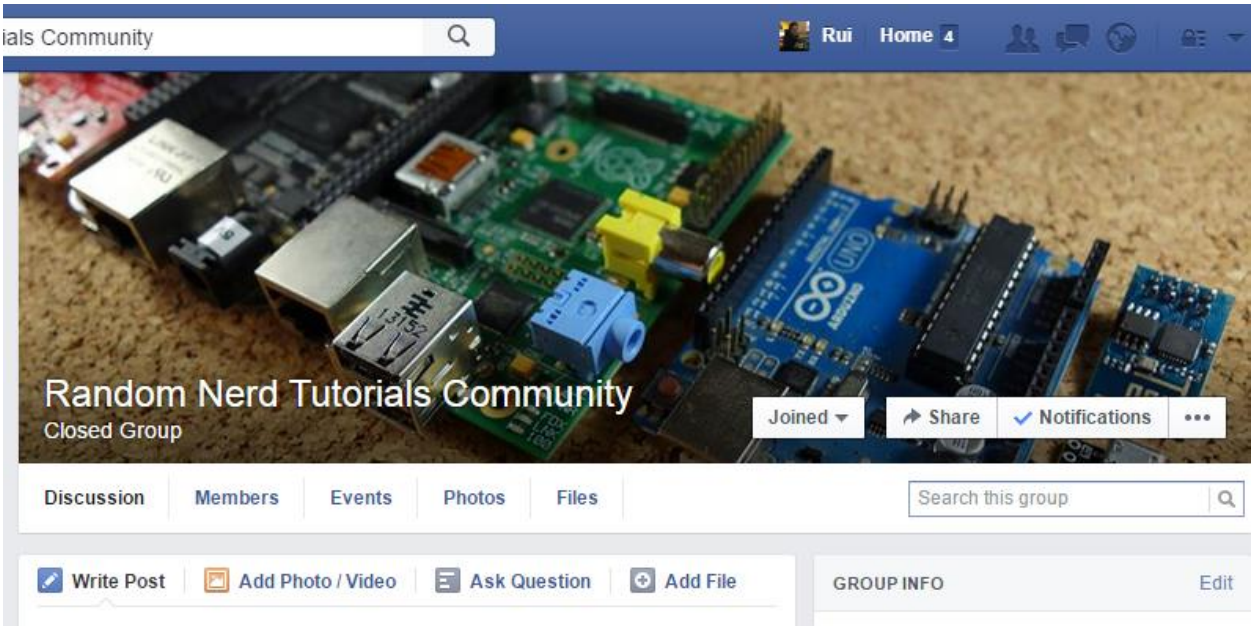
Good luck with all your projects,

-Rui

P.S. If you haven't already, you can download a FREE eBook with all my Arduino Projects by visiting -> http://randomnerdtutorials.com/ebook/.

Do You Have Any Questions?

If you completed this eBook and want more information visit the visit our Facebook group (http://randomnerdtutorials.com/fb/) or click the button below to join.



Click here to join our Facebook Group->

http://randomnerdtutorials.com/fb/

Time Sensitive Offer

I've launched a course called Blog1K. Blog1K is my proven step-by-step system designed to take you by the hand to create a blog that gets you a raise, lands you a new job or earns you an extra money on the side. It's literally step-by-step nothing is left out.

This Course Shows How Programmers and Engineers Can Make Extra Money On The Side, Land High-Paying Clients and Get a Raise Blogging.

Here's what this course can do for you:

- Help you earn more money as a Programmer or Engineer
- Having the flexibility to work from home
- Having clients come to you instead of you having to look for them
- Getting a raise in your current job
- Becoming an authority in your industry

This course usually sells for $77 as you can see in this page: http://blog1k.com/enroll-today. But I'm offering you a special deal for a very limited time.

You can get the complete course today for just one single payment of $27. Go to this page now: http://blog1k.com/special. This offer ends *soon*, so depending on when you're reading this eBook it might already expired. If you have any questions email me rui@blog1k.com.

Click here to take advantage of this special deal

http://blog1k.com/special

Download Other RNT Products

Random Nerd Tutorials is an online resource with electronics projects, tutorials and reviews.

Creating and posting new projects takes a lot of time. At this moment, Random Nerd Tutorials has nearly 100 free blog posts with complete tutorials using open-source hardware that anyone can read, remix and apply to their own projects: http://randomnerdtutorials.com

To keep free tutorials coming, there's also paid content or as I like to call "Premium Content".

To support Random Nerd Tutorials you can download Premium content here. If you enjoyed this eBook make sure you check all the others.

Thanks for taking the time to read my work!

Good luck with all your projects,

-Rui Santos

P.S. Click here for more Courses and eBooks like this one.

Click here to Download other Courses and eBooks

http://randomnerdtutorials.com/products