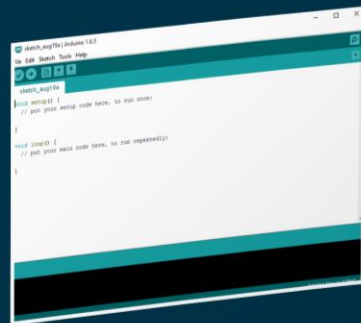
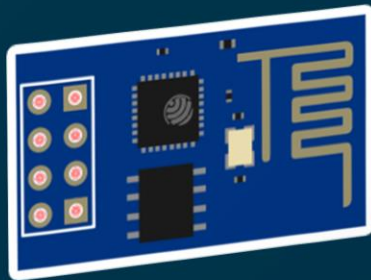


RANDOM NERD TUTORIALS - PREMIUM PROJECT

Password Protected Web Server Accessible from Anywhere using ESP8266 and Arduino IDE



Written by Rui Santos

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/products>

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

I.	Disclaimer	2
II.	Security Notice	3
III.	Table of Contents	4
IV.	About the Author	5
V.	Getting Started with ESP8266	7
VI.	Preparing Your Arduino IDE	15
VII.	Establishing a Serial Communication	22
VIII.	Building Your First Blinking LED Project with Arduino IDE	28
IX.	Reference for ESP8266 using Arduino IDE	35
X.	Password Protected Web Server with ESP8266	50
XI.	Making Your Web Server Accessible from Anywhere in the World	69
XII.	Recommended Tools	76
XIII.	Final Thoughts.....	78
XIV.	Time Sensitive Offer	Error! Bookmark not defined.
XV.	Download Other RNT Products	80

About the Author

Hey There,

Thank you for purchasing this [Random Nerd Tutorials Premium Project](#) and supporting my work!



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

If you're just getting started with the ESP8266, this eBook is perfect for you!

This eBook contains a project that you're going to build from scratch. It's a powerful password protected web server that you can access from anywhere to control your lights, a relay or any output that you want.

This low cost WiFi module (costs \$4) and the friendly Arduino IDE environment are the perfect solution for a Home Automation system.

Thanks for reading,

-Rui

P.S. If you would like the longer version of my story, you can find it over [here](#).

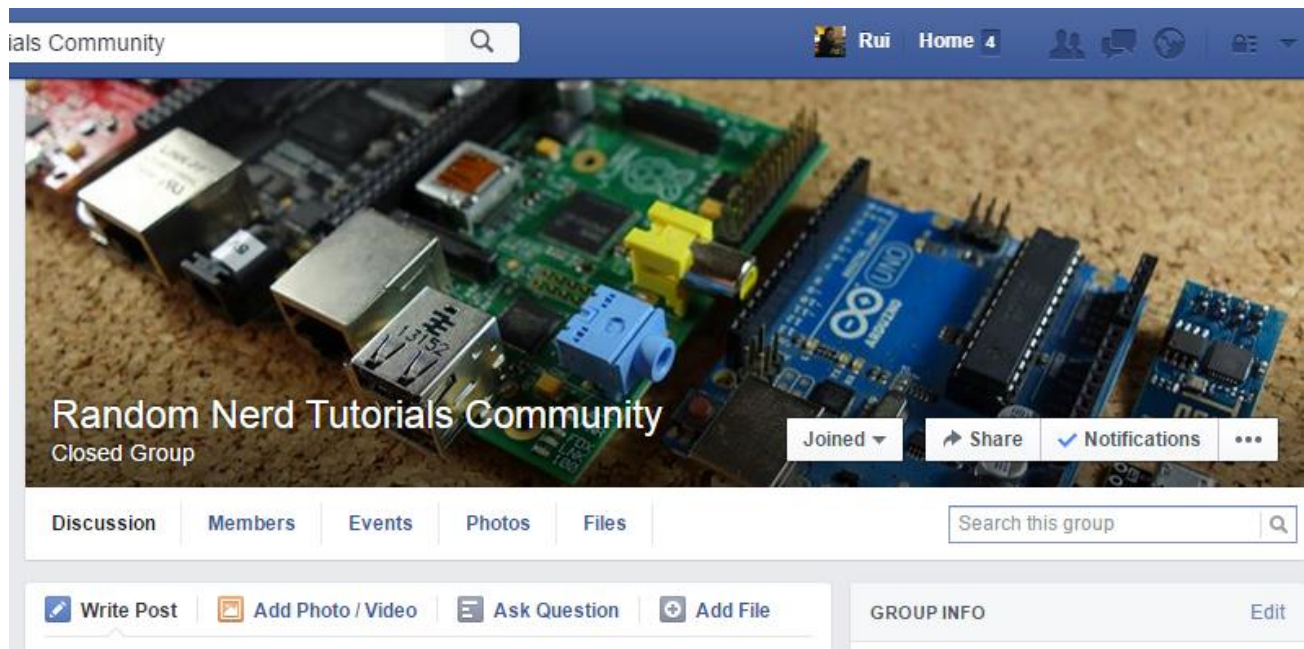
Join the Private Facebook Group

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!

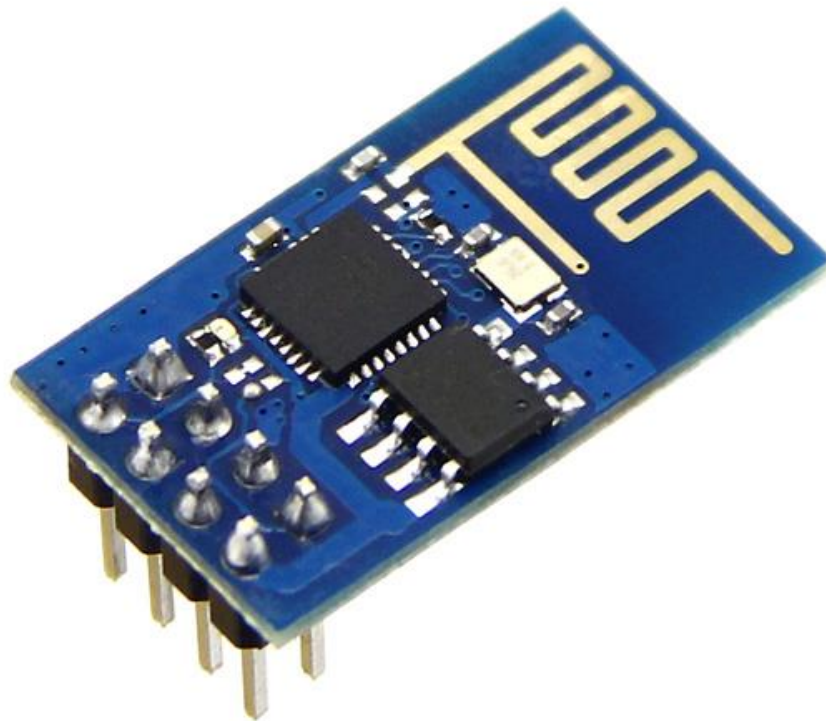
- Step #1: Go to -> <http://randomnerdtutorials.com/fb/>
- Step #2: Click “Join Group”
- Step #3: I’ll approve your request within less than 24 hours.



DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

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 a project that can be applied to automate your home.

This eBook covers:

- Technical specifications the ESP8266
- Where to buy the ESP8266
- How to install the Arduino IDE
- How to install the ESP8266 board on the Arduino IDE
- How to establish a serial communication with the ESP8266
- How to blink an LED with ESP8266 using Arduino IDE
- How to create a web server
- How make your web server password protected
- How to access your web server from anywhere
- And a lot more...

Let's Begin!

This first Unit gives you an overview of what you can do with your ESP8266, the ESP 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 board. You just need an ESP, 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 why 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, post tweets, etc.

Here's a list of tutorials that I've already created that you might find useful:

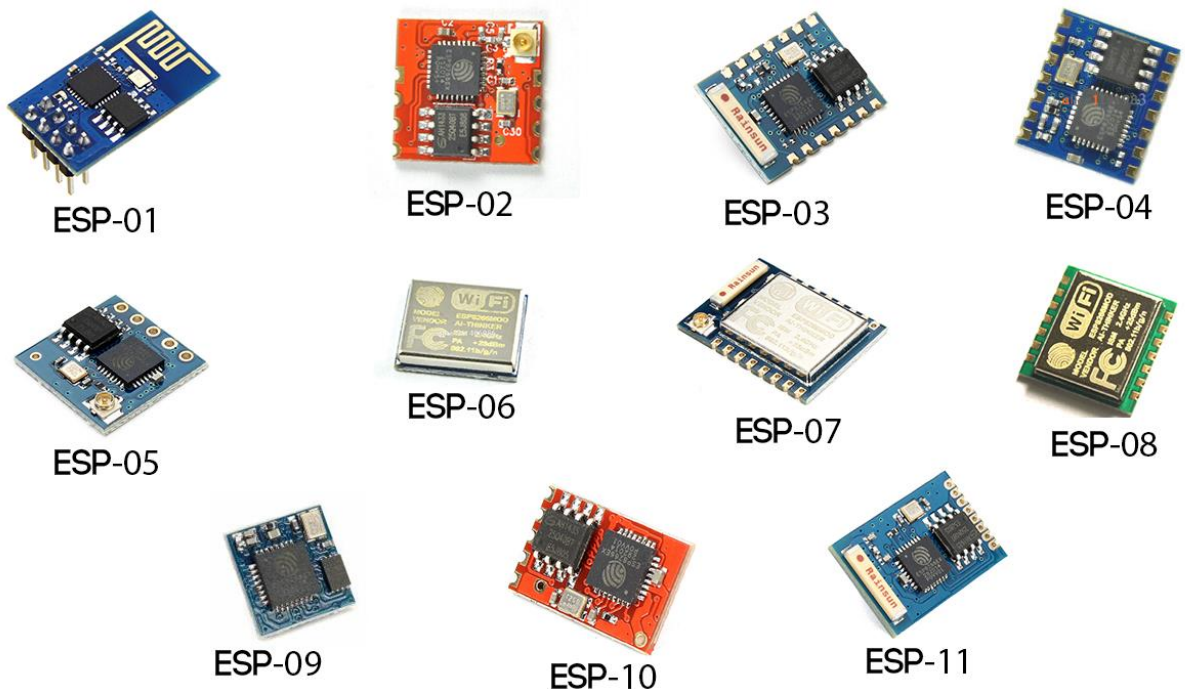
- [Posting a Tweet with the ESP8266](#)
- [How to Control Your ESP8266 From Anywhere in the World](#)
- [Retrieving Bitcoin Price Using ESP8266 WiFi Module](#)
- [ESP8266 Web Server Tutorial using NodeMCU](#)

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

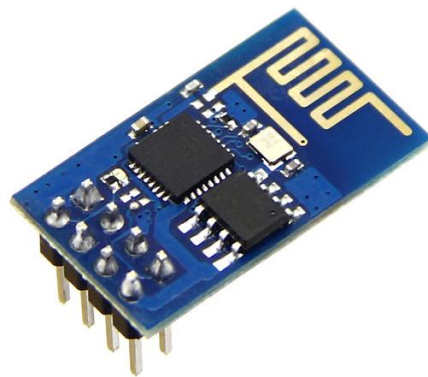
The ESP8266 comes in a wide variety of versions (see the following figure). The ESP-01 is the most common and that's the module we'll be using throughout this entire eBook.



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

This module is available through most electronics stores.

But still... The cheapest place to get one is eBay, you can 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 boards.

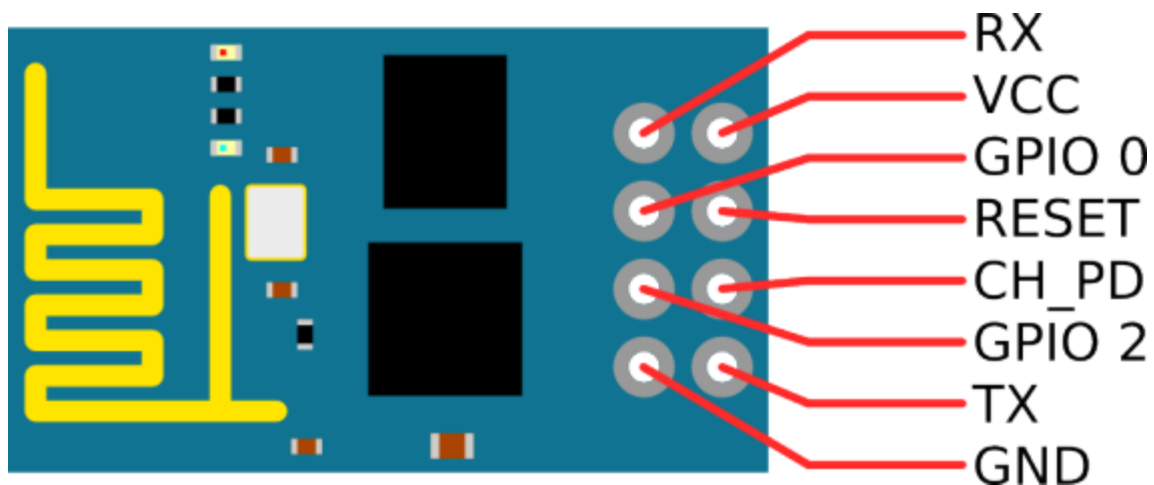
If you're using the ESP-07 or ESP-12 you need to connect their GPIO 15 to GND for all the schematics in this eBook.



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



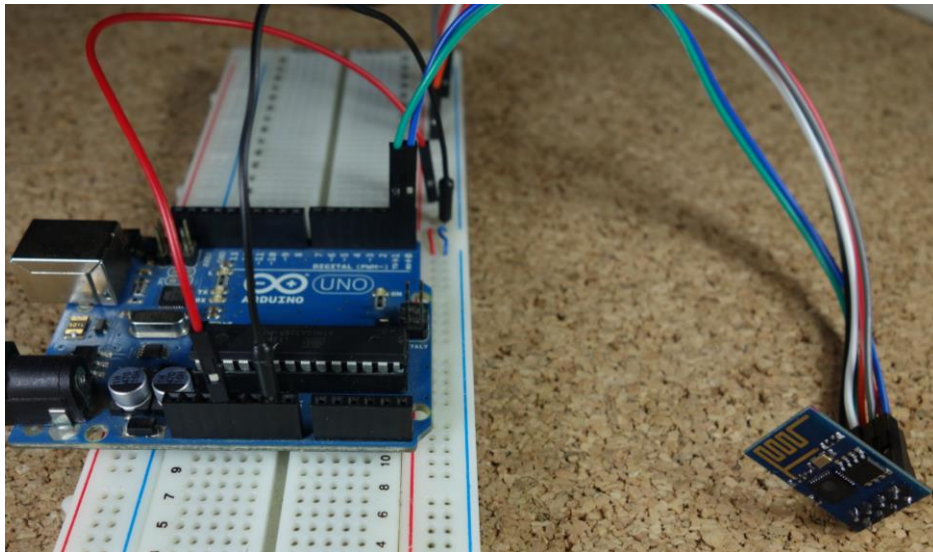
DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
 FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
 QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

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.

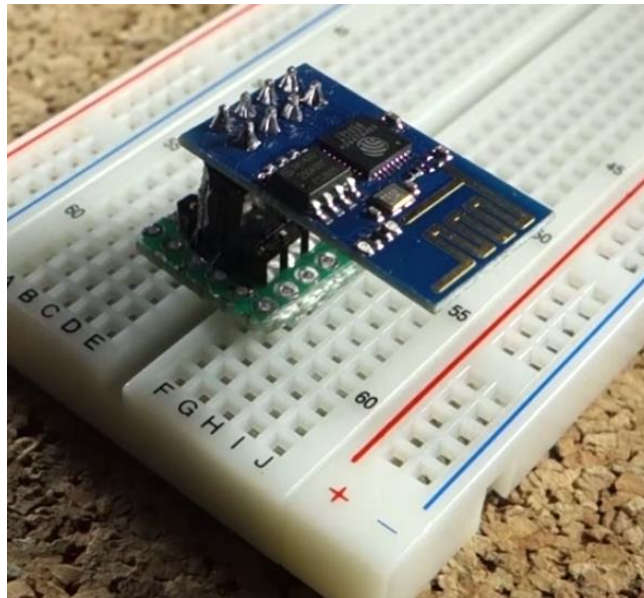
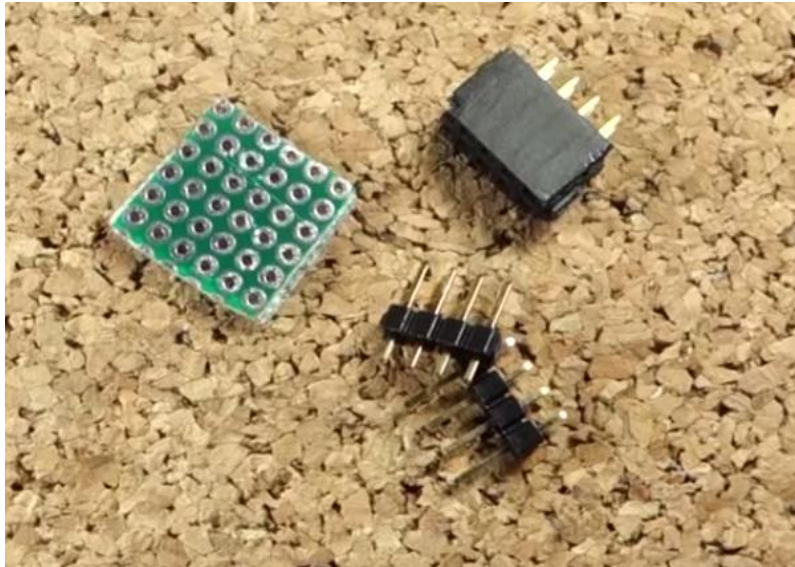
Your ESP8266 is Not Breadboard Friendly...

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



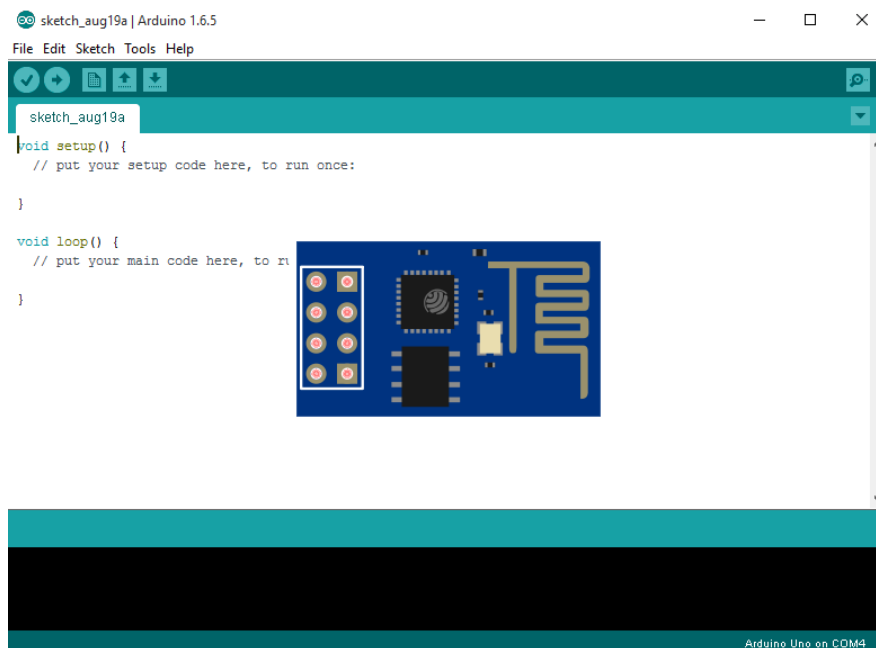
Option #2 – Small PCB



DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

Unit 2

Preparing Your Arduino IDE



Preparing Your Arduino IDE

In this Unit you're going to download, install and prepare your Arduino IDE to work with the ESP8266. This means you can program your ESP using the friendly Arduino programming language.

What's the Arduino IDE?

The Arduino IDE is an open-source software that makes it easy to write code and upload it to the Arduino board. Recently [this](#) GitHub repository added support for the ESP board to integrate with the Arduino IDE.

The Arduino IDE is a multiplatform software, this simply means that it runs on Windows, Mac OS X or Linux (it was created in JAVA).

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.

Downloading Arduino IDE

Now let's download the Arduino IDE, visit the following URL: <https://www.arduino.cc/en/Main/Software>.

Then select your operating system and download the software (as shown below).



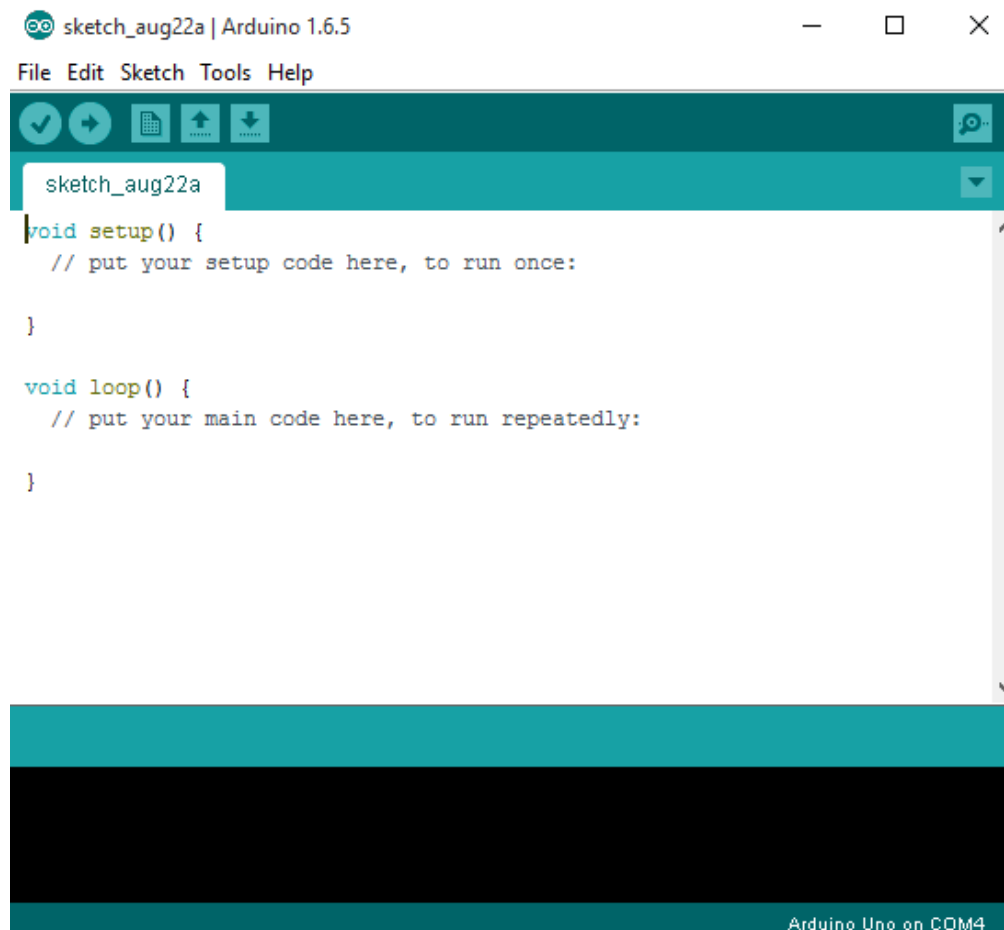
Installing Arduino IDE

Grab the file that you have just downloaded which is named “arduino-(...).zip”. Run that file and follow the installation wizard that shows on your screen.

Open the Arduino IDE application file (see Figure below).

Name	Date modified	Type	Size
dist	13/08/2015 20:53	File folder	
drivers	13/08/2015 20:53	File folder	
examples	13/08/2015 20:54	File folder	
hardware	13/08/2015 20:54	File folder	
java	13/08/2015 20:57	File folder	
lib	13/08/2015 20:59	File folder	
libraries	13/08/2015 21:00	File folder	
reference	13/08/2015 21:03	File folder	
tools	13/08/2015 21:03	File folder	
arduino	14/08/2015 17:42	Application	393 KB
arduino.l4j	13/08/2015 20:53	Configuration sett...	1 KB
arduino_debug	13/08/2015 20:53	Application	390 KB
arduino_debug.l4j	13/08/2015 20:53	Configuration sett...	1 KB
libusb0.dll	13/08/2015 20:53	Application extens...	43 KB
msvcp100.dll	13/08/2015 20:53	Application extens...	412 KB
msvcr100.dll	13/08/2015 20:53	Application extens...	753 KB
revisions	13/08/2015 20:53	Text Document	66 KB

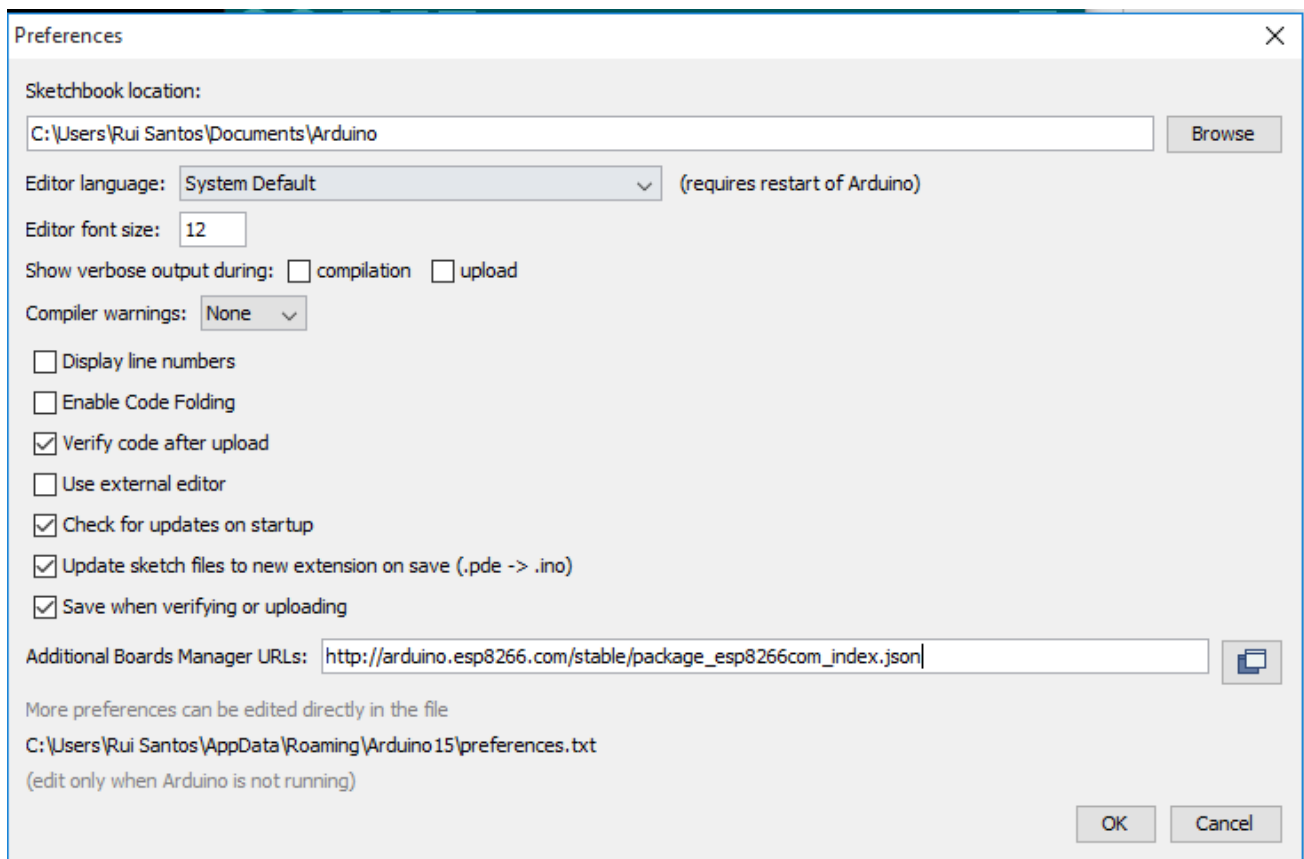
When the Arduino IDE first opens, this is what you should see:



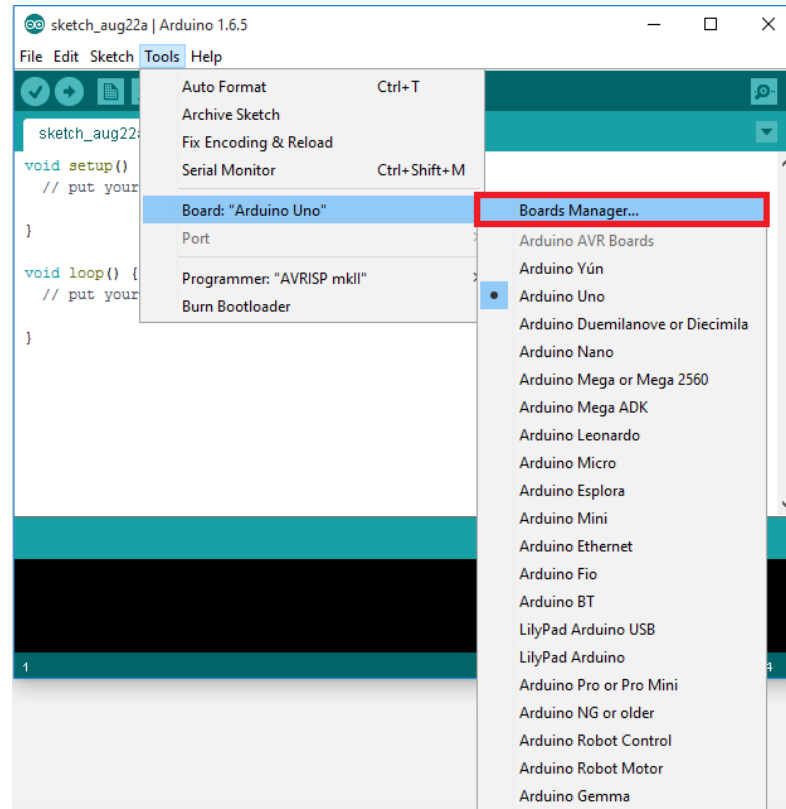
Installing the ESP8266 Board

To install the ESP8266 board in your Arduino IDE, follow these next instructions:

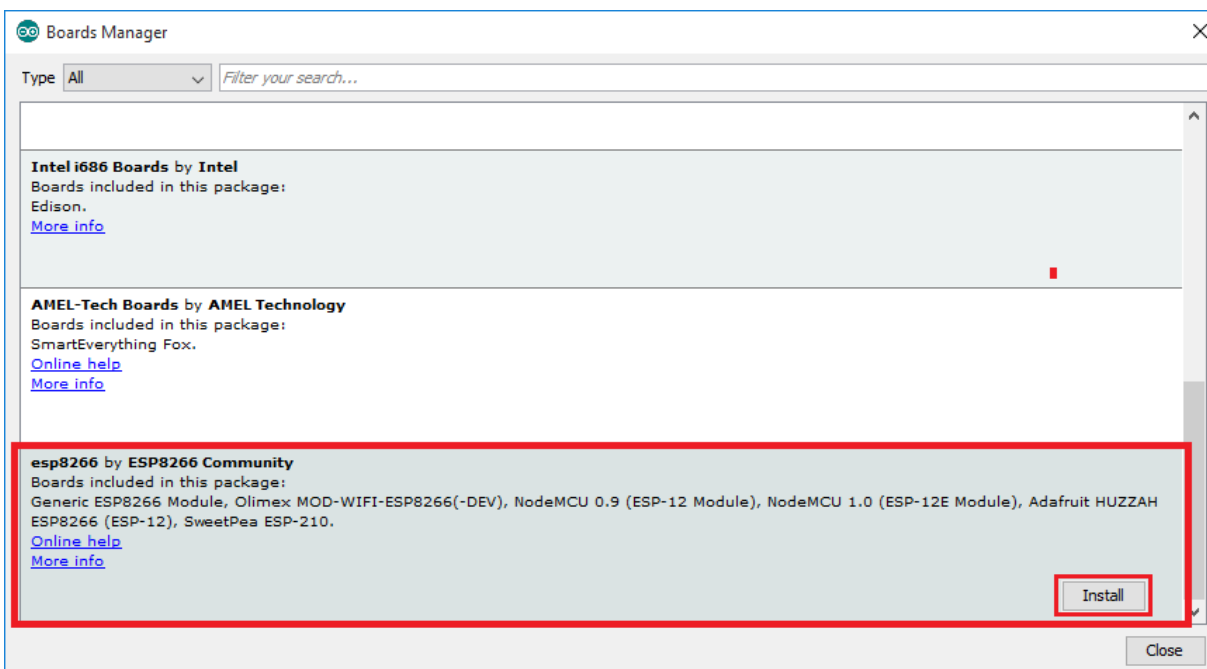
1. Open the preferences window from the Arduino IDE. Go to File > Preferences
2. Enter http://arduino.esp8266.com/stable/package_esp8266com_index.json into Additional Board Manager URLs field and click the “OK” button



3. Open boards manager. Go to Tools > Board > Boards Manager...

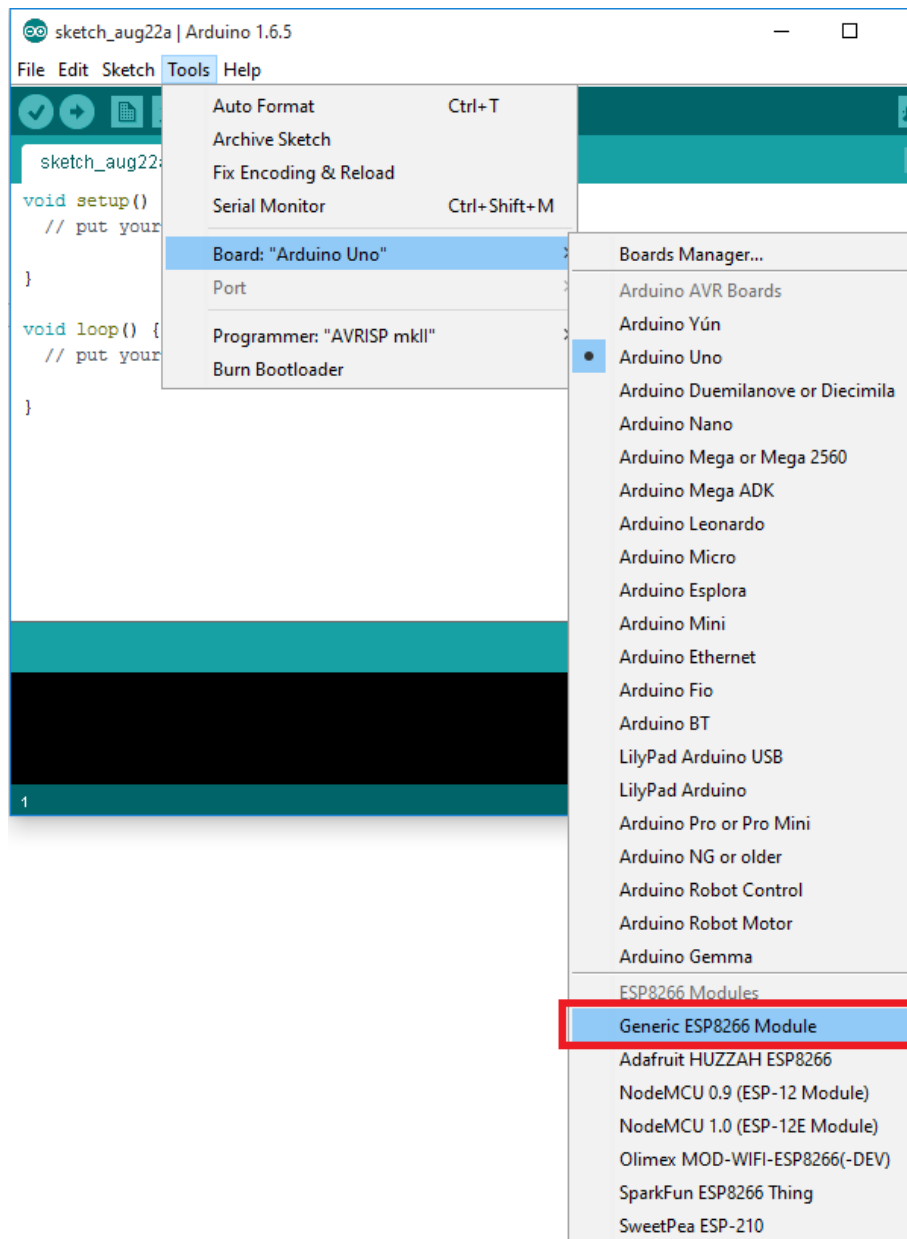


4. Scroll down, select the ESP8266 board menu and install “esp8266 platform”



DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

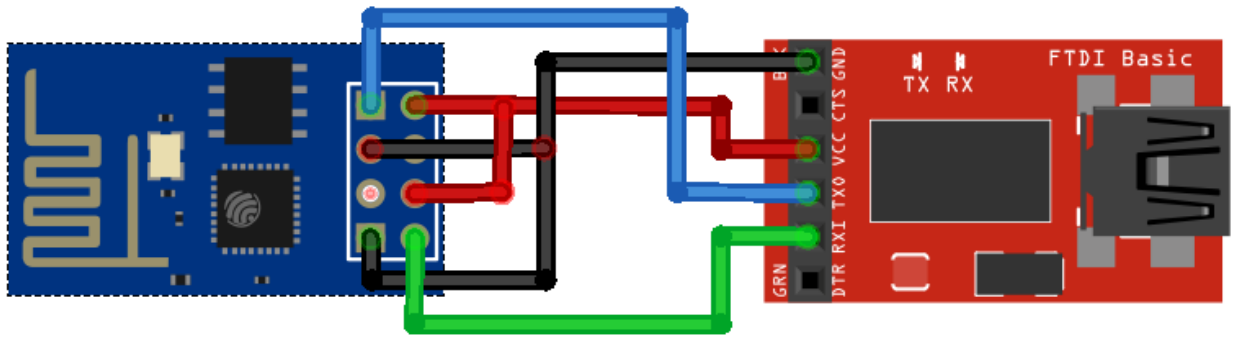
5. Choose your ESP8266 board from Tools > Board > Generic ESP8266 Module



6. Finally, re-open your Arduino IDE

Unit 3

Establishing a Serial Communication



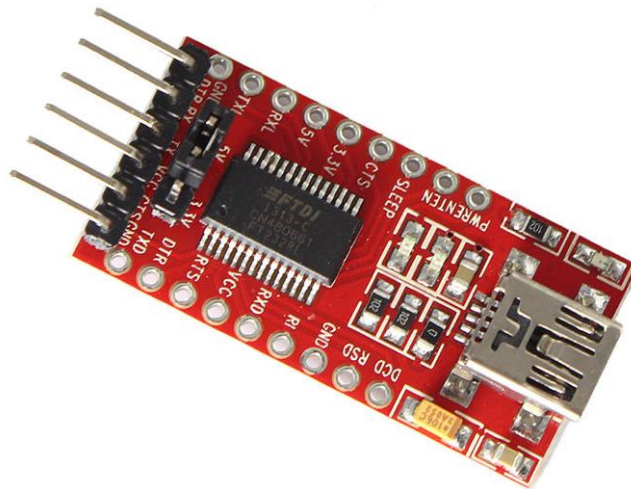
Establishing a Serial Communication

Unit 3 covers how you can establish a serial communication with your ESP8266 using an FTDI Programmer or an Arduino.

Using FTDI Programmer 3.3V

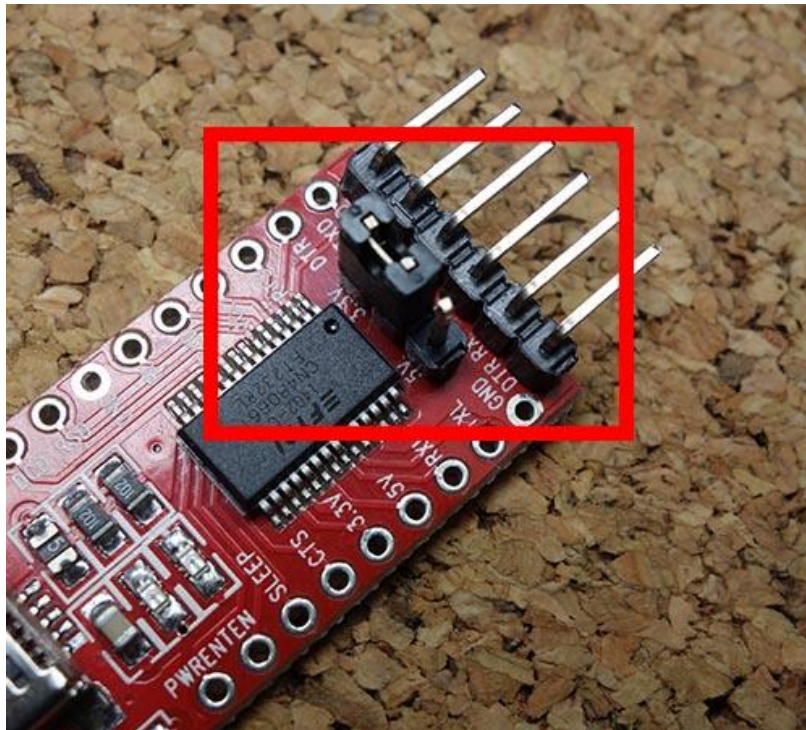
If you have a 3.3V FTDI Programmer it's really easy to get started and establish a serial communications with your ESP.

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

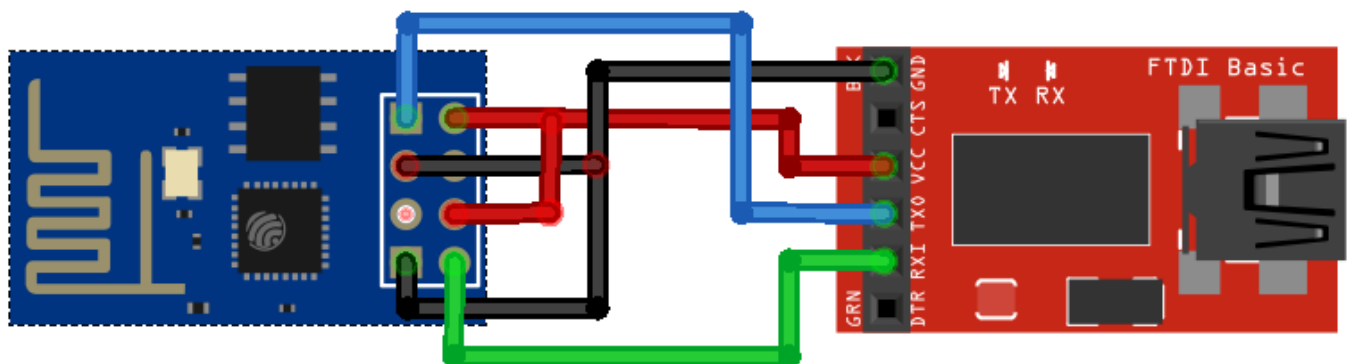


Important: Most FTDI programmers have a jumper to convert from 5V to 3.3V. So make sure your FTDI programmer is set to 3.3V operation (as shown in the following Figure).

DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)



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 upload code. When you upload a new sketch into your ESP it requires the ESP to flash a new firmware. In normal usage (if you're not flashing your ESP with a new firmware) it would be connected to VCC.

Unbricking the FTDI Programmer on Windows

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 7/8/8.1/10 it's very likely that FTDI is bricked. Follow this tutorial to fix that: <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)

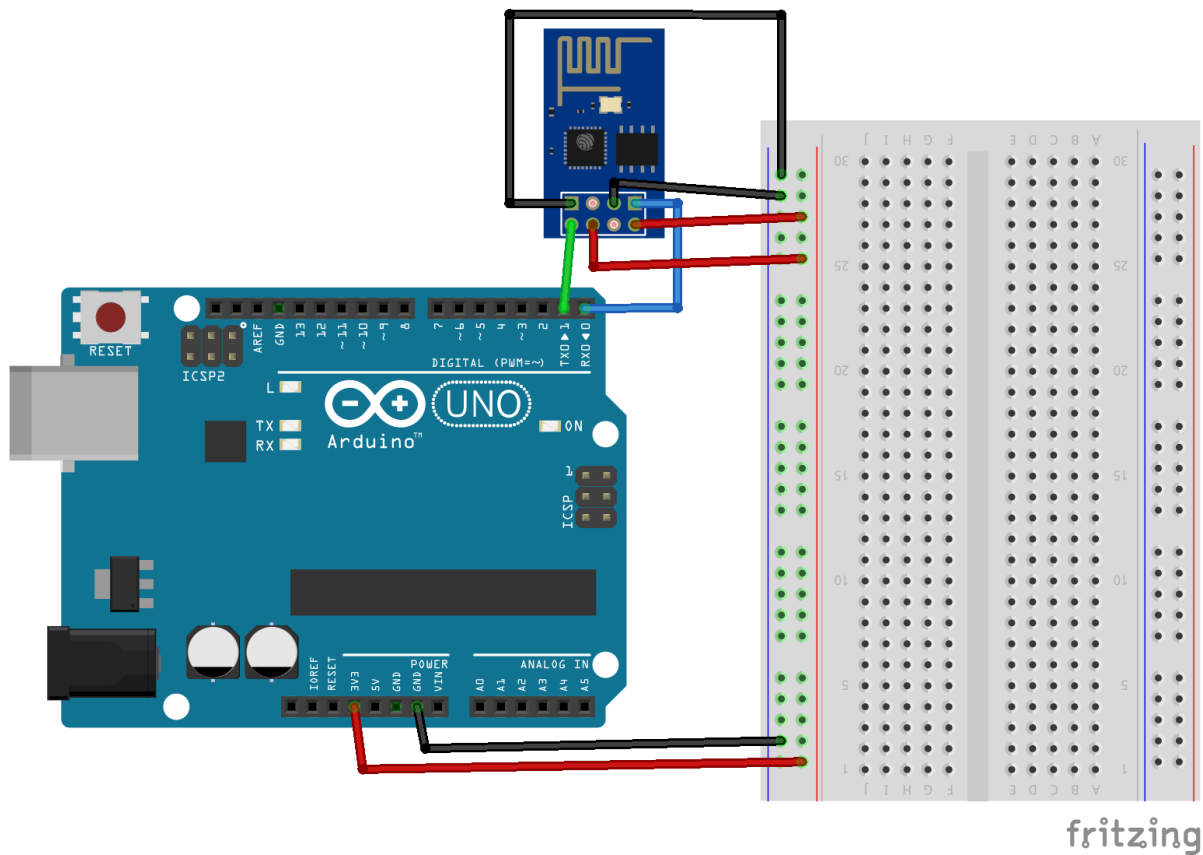
I don't recommend using an Arduino to power your ESP8266 and establish a serial communication. You may break both your Arduino and your ESP. Since the ESP does not have 5V tolerant pins.

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)

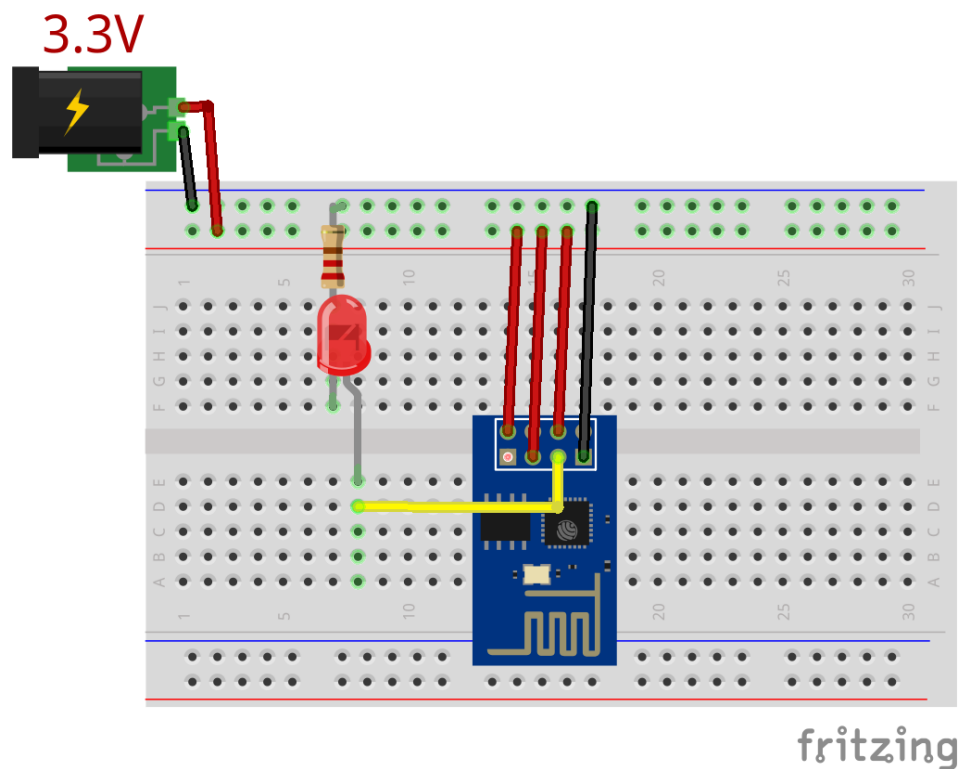
Warning: The following circuit works fine for me and I've been playing with the same ESP module for a long time without any problem. But as a final project you shouldn't power your ESP8266 module with the 3.3V from your Arduino, because it might not supply enough current. So I advise you to use an FTDI 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 upload code. When you upload a new sketch into your ESP it requires the ESP to flash a new firmware. In normal usage (if you're not flashing your ESP with a new firmware) it would be connected to VCC.

Unit 4

Building Your First Blinking LED Project with Arduino IDE



Building Your First Blinking LED Project with Arduino IDE

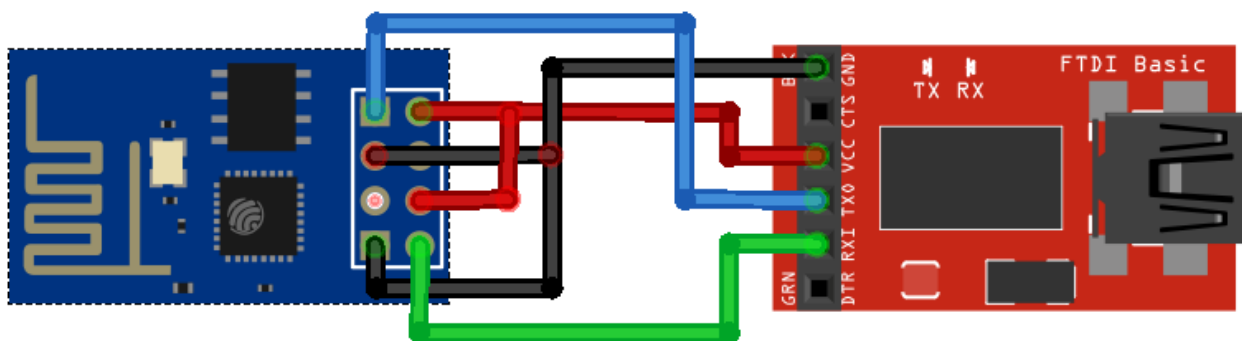
In this Unit you're going to design a simple circuit to blink an LED with ESP and using the Arduino IDE.

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.

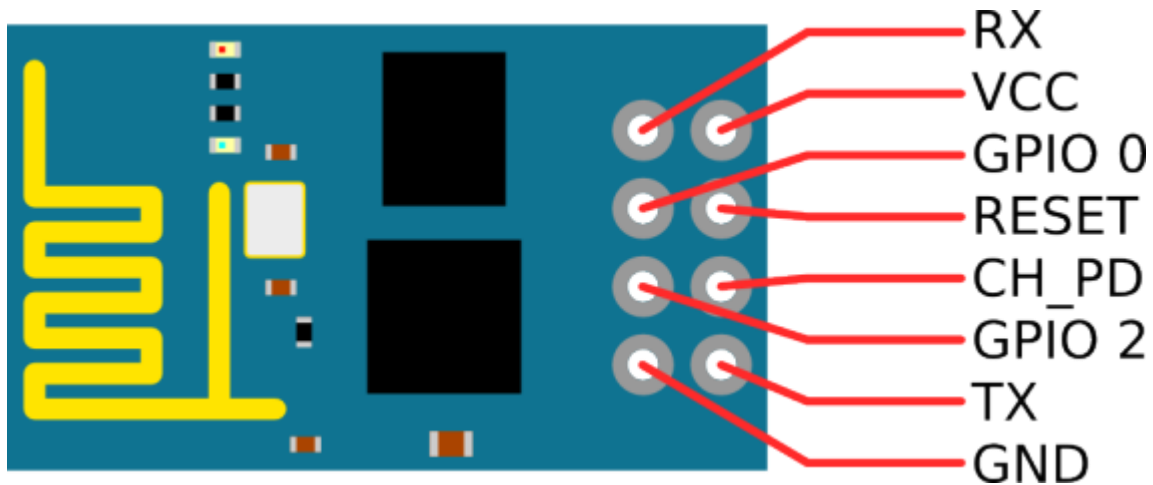
Schematics

To upload code to your ESP8266, you should connect your ESP 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 Arduino Sketch” when we define:

```
pin = 0
```

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

```
pin = 2
```

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

Writing Your Arduino Sketch

Below is your sketch to blink an LED. You can download the Arduino Sketch in the following link:

<https://gist.github.com/RuiSantosdotme/789861277da9680e9cfb>

```
int pin = 2;

void setup() {
  // initialize GPIO 2 as an output.
  pinMode(pin, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(pin, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(pin, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

How this sketch works:

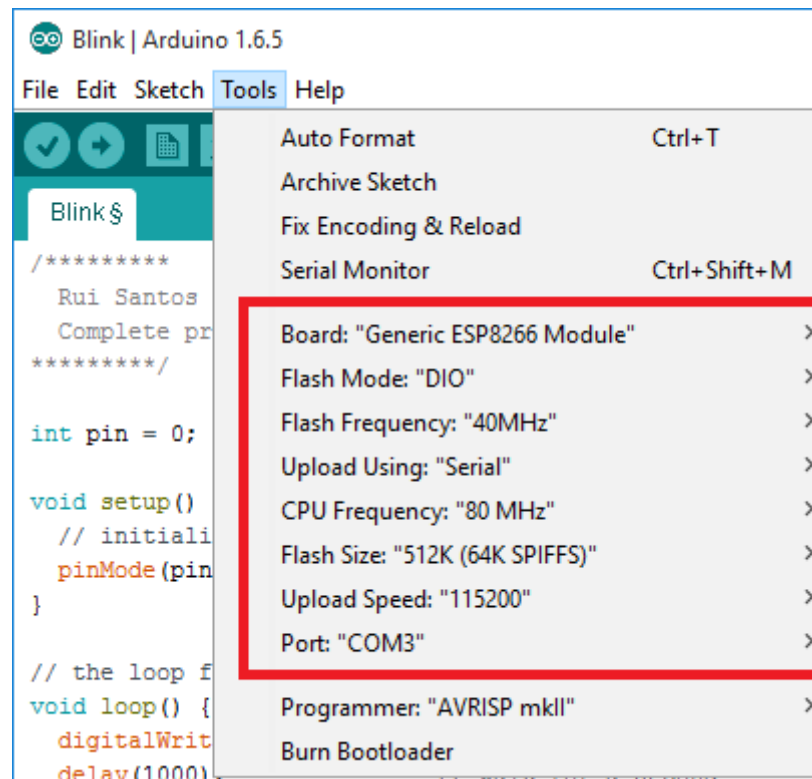
1. Create a *integer (int)* variable called *pin = 2* which refers to GPIO 2
2. In you *setup()*, you use the function *pinMode(pin, OUTPUT)* to set your GPIO 2 as an *OUTPUT* . This code only runs once.
3. Next in your *loop()*, you use two functions *digitalWrite()* and *delay()*. This section of code will run over and over again until you unplug your ESP.
4. First you turn the LED on for 1 second (1000 milliseconds) using *digitalWrite(pin, HIGH)* and *delay(1000)*.
5. Then you turn the LED off using *digitalWrite(pin, LOW)* and wait 1 second with *delay(1000)*.

6. The program keeps repeating steps 4. and 5. which causes the LED to blink!

Uploading Code

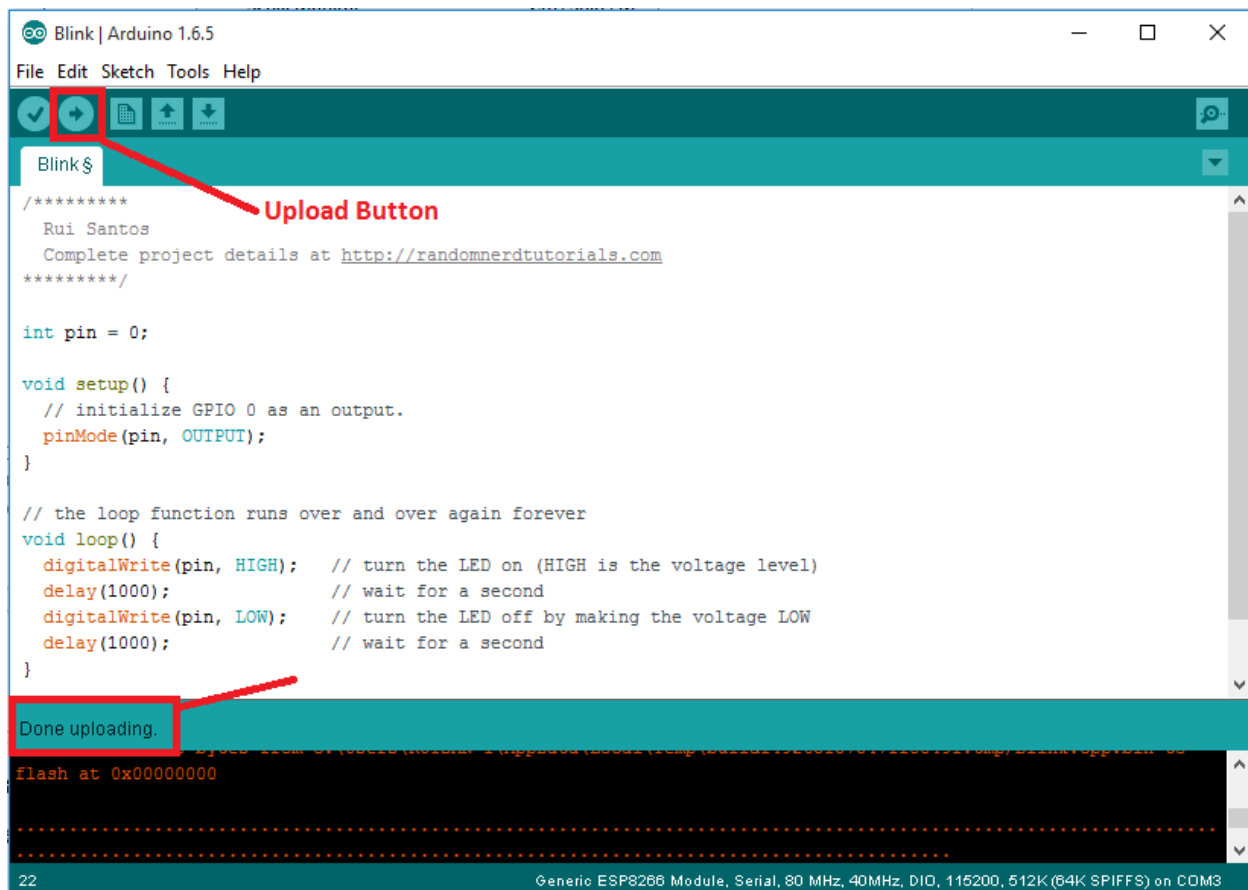
Once you have your ESP8266+FTDI Programmer connected to your computer, go to the Arduino IDE.

Look at the Tools menu and check all the configurations, by default they should look like this:



Important: Your COM port is very likely to be different from the screenshot above (Port: “COM3”). That’s fine, because it doesn’t interfere with anything. On the other hand all the other configurations should look exactly like mine.

After checking the configurations click the “Upload Button” and wait a few seconds until you see the message “Done uploading.” in the bottom left corner.

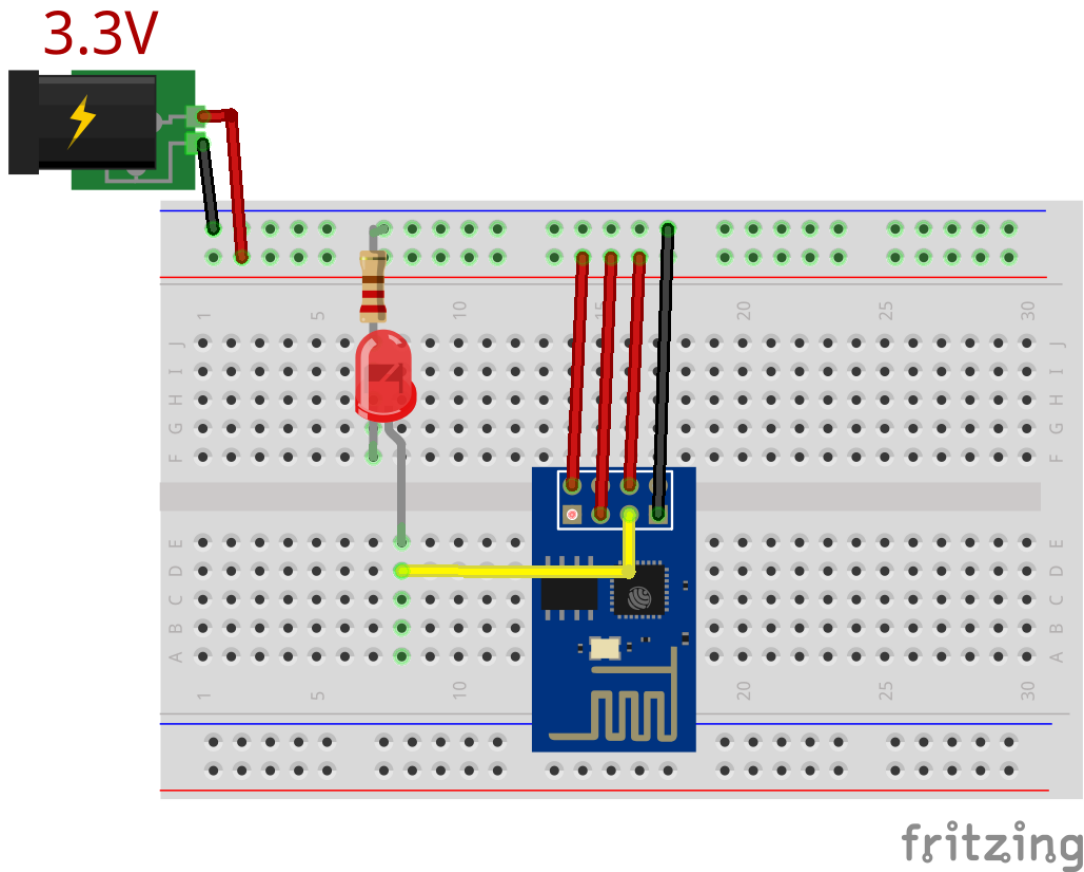


Here's Your Final Circuit

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

DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

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

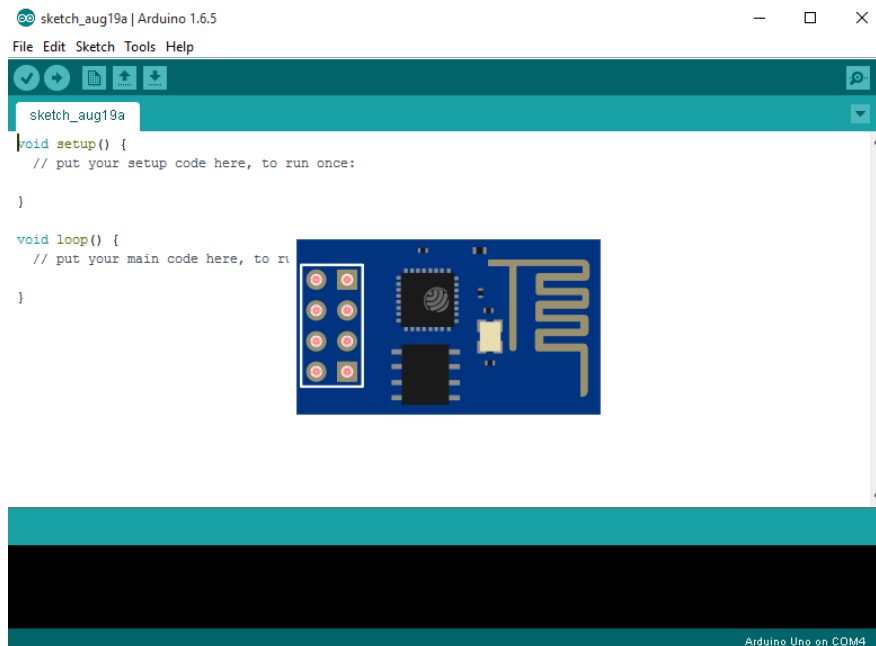


Restart your ESP8266.

Congratulations, you've made it! Your LED should be blinking every 1 second!

Unit 5

Reference for ESP8266 using Arduino IDE



Reference for ESP8266 using Arduino IDE

Before diving deeper into your web server project I thought it would be helpful to create a Unit dedicated on how to use the Arduino IDE with the ESP.

I encourage you to read the “Digital IO” section, the rest of this Unit is optional, so feel free to skip it and use it as a reference guide for later projects.

More documentation and details about this reference can be found here:
<http://arduino.esp8266.com/versions/1.6.5-947-g39819fo/doc/reference.html>

Note: This Unit might be subject to change in the future, since I don't control updates and changes made by the authors of this ESP integration with the Arduino IDE. Please check the preceding URL for the latest information. This eBook will be updated if necessary.

Digital IO

Let's talk about GPIOs. 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.

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

Mode	Description
OUTPUT	You set the pin to HIGH or LOW
INPUT	You read the current state of the pin
INPUT_PULLUP	Similar to INPUT, but you use internal pull-up resistors
INPUT_PULLDOWN	Similar to INPUT, but you use internal pull-down resistors
INTERRUPT	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. Pin numbers in Arduino IDE correspond directly to the ESP8266 GPIO pin numbers.

The ESP version 01 has only two: GPIO 0 and GPIO 2.

IO index (in code)	ESP8266 GPIO	Available Modes
0	GPIO 0	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
1	GPIO 1	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
2	GPIO 2	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT

3	GPIO 3	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
4	GPIO 4	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
5	GPIO 5	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
6	GPIO 6	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
7	GPIO 7	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
8	GPIO 8	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
9	GPIO 9	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
10	GPIO 10	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
11	GPIO 11	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
12	GPIO 12	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
13	GPIO 13	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
14	GPIO 14	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
15	GPIO 15	INPUT, OUTPUT, INPUT_PULLUP or INTERRUPT
16	GPIO 16	INPUT, OUTPUT, INPUT_PULLDOWN

Important: At startup, pins are configured as INPUT.

Note: Pins may also serve other functions, like Serial, I2C, SPI. These functions are normally activated by the corresponding library.

OUTPUT Mode

Using *digitalWrite()* 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):

```
int pin = 2;
pinMode(pin, OUTPUT);
digitalWrite(pin, HIGH);
```

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

```
int pin = 2;
pinMode(pin, OUTPUT);
digitalWrite(pin, LOW);
```

INPUT Mode

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

```
int pin = 2;
pinMode(pin, INPUT);
digitalRead(pin);
```

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

INTERRUPT Mode

Pin interrupts are supported in the ESP through *attachInterrupt()* and *detachInterrupt()* functions.

Similar to the Arduino. Interrupts may be attached to any GPIO pin, except GPIO 16. Standard Arduino interrupt types are supported: CHANGE, RISING and FALLING. More information can be found here:

- <https://www.arduino.cc/en/Reference/AttachInterrupt>

- <https://www.arduino.cc/en/Reference/DetachInterrupt>

Analog Input

The ESP-01 doesn't offer any analog inputs.

On the other hand, other versions such as the ESP-12 has a single ADC channel available to users. It may be used either to read voltage at ADC pin, or to read module supply voltage (VCC).

To read external voltage applied to ADC pin, use *analogRead(Ao)*.

Important: Input voltage range is 0 — 1.0V.

To read VCC voltage, ADC pin must be kept unconnected. Additionally, the following line has to be added to the sketch:

```
ADC_MODE(ADC_VCC);
```

This line has to appear outside of any functions, for instance right after the #include lines of your sketch.

Analog Output

analogWrite(pin, value) enables software PWM on the given pin. PWM may be used on pins 0 to 16. Call *analogWrite(pin, 0)* to disable PWM on the pin. value may be in range from 0 to PWM_RANGE, which is equal to 1023 by default. PWM range may be changed by calling *analogWriteRange(new_range)*.

DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

PWM frequency is 1kHz by default. Call *analogWriteFreq(new_frequency)* to change the frequency.

Timing and Delays

millis() and *micros()* return the number of milliseconds and microseconds elapsed after reset, respectively.

delay(ms) pauses the sketch for a given number of milliseconds and allows WiFi and TCP/IP tasks to run. *delayMicroseconds(us)* pauses for a given number of microseconds.

Remember that there is a lot of code that needs to run on the chip besides the sketch when WiFi is connected. WiFi and TCP/IP libraries get a chance to handle any pending events each time the *loop()* function completes, OR when delay is called. If you have a loop somewhere in your sketch that takes a lot of time (>50ms) without calling *delay()*, you might consider adding a call to delay function to keep the WiFi stack running smoothly.

There is also a *yield()* function which is equivalent to *delay(0)*. The *delayMicroseconds* function, on the other hand, does not yield to other tasks, so using it for delays more than 20 milliseconds is not recommended.

Serial

Serial object works much the same way as on a regular Arduino. Apart from hardware FIFO (128 bytes for TX and RX) HardwareSerial has additional 256-byte TX and RX buffers. Both transmit and receive is interrupt-driven.

DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

Write and read functions only block the sketch execution when the respective FIFO/buffers are full/empty.

Serial uses UART 0, which is mapped to pins GPIO 1 (TX) and GPIO 3 (RX). Serial may be remapped to GPIO 15 (TX) and GPIO 13 (RX) by calling *Serial.swap()* after *Serial.begin()*. Calling swap again maps UART 0 back to GPIO 1 and GPIO 3.

Serial1 uses UART 1, TX pin is GPIO 2. UART 1 can not be used to receive data because normally it's RX pin is occupied for flash chip connection. To use *Serial1*, call *Serial1.begin(baudrate)*.

By default the diagnostic output from WiFi libraries is disabled when you call *Serial.begin()*. To enable debug output again, call *Serial.setDebugOutput(true)*. To redirect debug output to *Serial1* instead, call *Serial1.setDebugOutput(true)*.

You also need to use *Serial.setDebugOutput(true)* to enable output from *printf()* function.

Both Serial and Serial1 objects support 5, 6, 7, 8 data bits, odd (O), even (E), and no (N) parity, and 1 or 2 stop bits. To set the desired mode, call *Serial.begin(baudrate,SERIAL_8N1)*, *Serial.begin(baudrate,SERIAL_6E2)*, etc.

Progmem

The Program memory features work much the same way as on a regular Arduino; placing read only data and strings in read only memory and freeing

heap for your application. The important difference is that on the ESP8266 the literal strings are not pooled. This means that the same literal string defined inside `aF("")` and/or `PSTR("")` will take up space for each instance in the code. So you will need to manage the duplicate strings yourself.

WiFi (ESP8266WiFi library)

This is mostly similar to WiFi shield library. Differences include:

- `WiFi.mode(m)`: set mode to `WIFI_AP`, `WIFI_STA`, or `WIFI_AP_STA`.
- call `WiFi.softAP(ssid)` to set up an open network
- call `WiFi.softAP(ssid, password)` to set up a WPA2-PSK network (password should be at least 8 characters)
- `WiFi.macAddress(mac)` is for STA, `WiFi.softAPmacAddress(mac)` is for AP.
- `WiFi.localIP()` is for STA, `WiFi.softAPIP()` is for AP.
- `WiFi.RSSI()` doesn't work
- `WiFi.printDiag(Serial)` will print out some diagnostic info
- `WiFiUDP` class supports sending and receiving multicast packets on STA interface. When sending a multicast packet, replace `udp.beginPacket(addr,port)` with `udp.beginPacketMulticast(addr, port, WiFi.localIP())`. When listening to multicast packets, replace `udp.begin(port)` with `udp.beginMulticast(WiFi.localIP(), multicast_ip_addr,port)`. You can use `udp.destinationIP()` to tell whether the packet received was sent to the multicast or unicast address. Also note that multicast doesn't work on softAP interface.

WiFiServer, *WiFiClient* and *WiFiUDP* behave mostly the same way as with WiFi shield library. Four samples are provided for this library. You can see more commands here: <http://www.arduino.cc/en/Reference/WiFi>

Ticker

Library for calling functions repeatedly with a certain period. Two examples included.

It is currently not recommended to do blocking IO operations (network, serial, file) from Ticker callback functions. Instead, set a flag inside the ticker callback and check for that flag inside the loop function.

EEPROM

This is a bit different from standard EEPROM class. You need to call *EEPROM.begin(size)* before you start reading or writing, size being the number of bytes you want to use. Size can be anywhere between 4 and 4096 bytes.

EEPROM.write() does not write to flash immediately, instead you must call *EEPROM.commit()* whenever you wish to save changes to flash. *EEPROM.end()* will also commit, and will release the RAM copy of EEPROM contents.

EEPROM library uses one sector of flash located at 0x7b000 for storage.

I2C (Wire library)

Wire library currently supports master mode up to approximately 450KHz. Before using I2C, pins for SDA and SCL need to be set by calling *Wire.begin(int sda, int scl)*, i.e. *Wire.begin(0, 2)* on ESP-01, else they default to pins 4(SDA) and 5(SCL).

SPI

SPI library supports the entire Arduino SPI API including transactions, including setting phase (CPHA). Setting the Clock polarity (CPOL) is not supported, yet (SPI_MODE2 and SPI_MODE3 not working).

ESP-specific APIs

APIs related to deep sleep and watchdog timer are available in the ESP object, only available in Alpha version.

- *ESP.deepSleep(microseconds, mode)* will put the chip into deep sleep. mode is one of *WAKE_RF_DEFAULT*, *WAKE_RFCAL*, *WAKE_NO_RFCAL*, *WAKE_RF_DISABLED*. (GPIO16 needs to be tied to RST to wake from deepSleep.)
- *ESP.restart()* restarts the CPU.
- *ESP.getFreeHeap()* returns the free heap size.
- *ESP.getChipId()* returns the ESP8266 chip ID as a 32-bit integer.

Several APIs may be used to get flash chip info:

- *ESP.getFlashChipId()* returns the flash chip ID as a 32-bit integer.
- *ESP.getFlashChipSize()* returns the flash chip size, in bytes, as seen by the SDK (may be less than actual size).
- *ESP.getFlashChipSpeed(void)* returns the flash chip frequency, in Hz.
- *ESP.getCycleCount()* returns the cpu instruction cycle count since start as an unsigned 32-bit. This is useful for accurate timing of very short actions like bit banging.
- *ESP.getVcc()* may be used to measure supply voltage. ESP needs to reconfigure the ADC at startup in order for this feature to be available. Add the following line to the top of your sketch to use *getVcc*:

```
ADC_MODE(ADC_VCC);
```

Important: TOUT pin has to be disconnected in this mode.

Note: that by default ADC is configured to read from TOUT pin using *analogRead(Ao)* and *ESP.getVCC()* is not available.

OneWire

from https://www.pjrc.com/teensy/td_libs_OneWire.html

Library was adapted to work with ESP8266 by including register definitions into OneWire.h Note that if you already have OneWire library in your Arduino/libraries folder, it will be used instead of the one that comes with this package.

mDNS and DNS-SD responder (ESP8266mDNS library)

Allows the sketch to respond to multicast DNS queries for domain names like "foo.local", and DNS-SD (service discovery) queries. Currently the library only works on STA interface, AP interface is not supported. See attached example for details.

SSDP responder (ESP8266SSDP)

SSDP is another service discovery protocol, supported on Windows out of the box. See attached example for reference.

DNS server (DNSServer library)

Implements a simple DNS server that can be used in both STA and AP modes. The DNS server currently supports only one domain (for all other domains it will reply with NXDOMAIN or custom status code). With it clients can open a web server running on ESP8266 using a domain name, not an IP address. See attached example for details.

Servo

This library exposes the ability to control RC (hobby) servo motors. It will support upto 24 servos on any available output pin. By default the first 12

servos will use Timero and currently this will not interfere with any other support. Servo counts above 12 will use Timer1 and features that use it will be effected. While many RC servo motors will accept the 3.3V IO data pin from a ESP8266, most will not be able to run off 3.3v and will require another power source that matches their specifications. Make sure to connect the grounds between the ESP8266 and the servo motor power supply.

Other supported libraries (not included with the Arduino IDE)

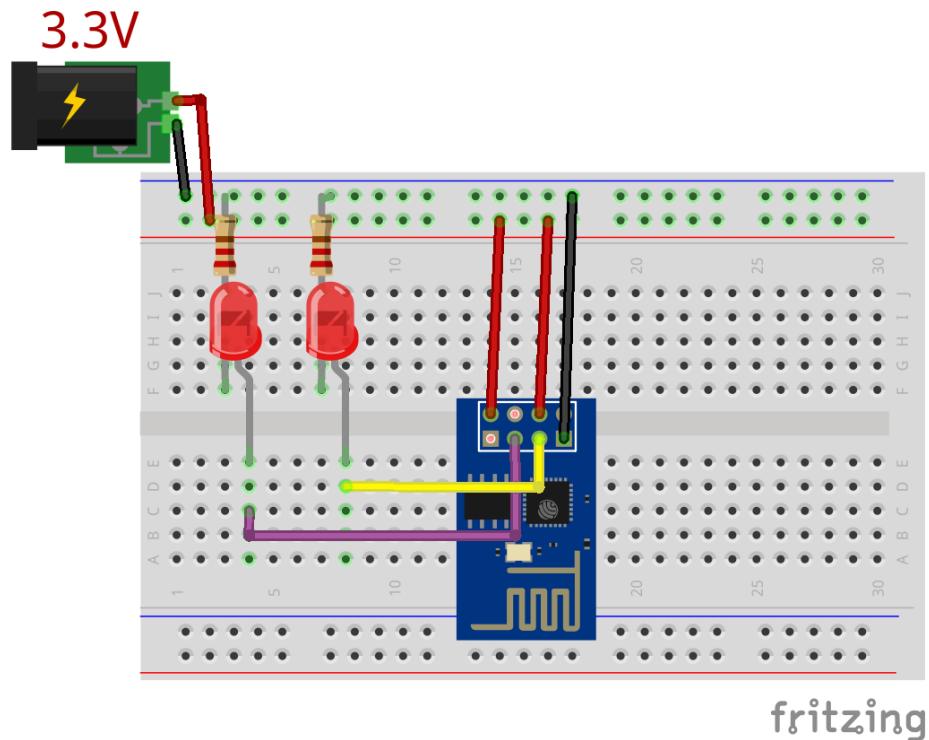
Libraries that don't rely on low-level access to AVR registers should work well. Here are a few libraries that were verified to work:

- [arduinoWebSockets](#) - WebSocket Server and Client compatible with ESP8266 (RFC6455)
- [aREST](#) - REST API handler library
- [Blynk](#) - easy IoT framework for Makers
- [DallasTemperature](#)
- [DHT11](#) - Download latest v1.1.0 library and no changes are necessary
- [NeoPixel](#) - Adafruit's NeoPixel library, now with support for the ESP8266
- [NeoPixelBus](#) - Arduino NeoPixel library compatible with ESP8266.
- [PubSubClient](#) MQTT library
- [RTC](#) - Arduino Library for Ds1307 & Ds3231 compatible with ESP8266
- [Souliss, Smart Home](#) - Framework for Smart Home based on Arduino, Android and openHAB

- [ST7735](#) - Adafruit's ST7735 library modified to be compatible with ESP8266

Unit 6

Password Protected Web Server with ESP8266



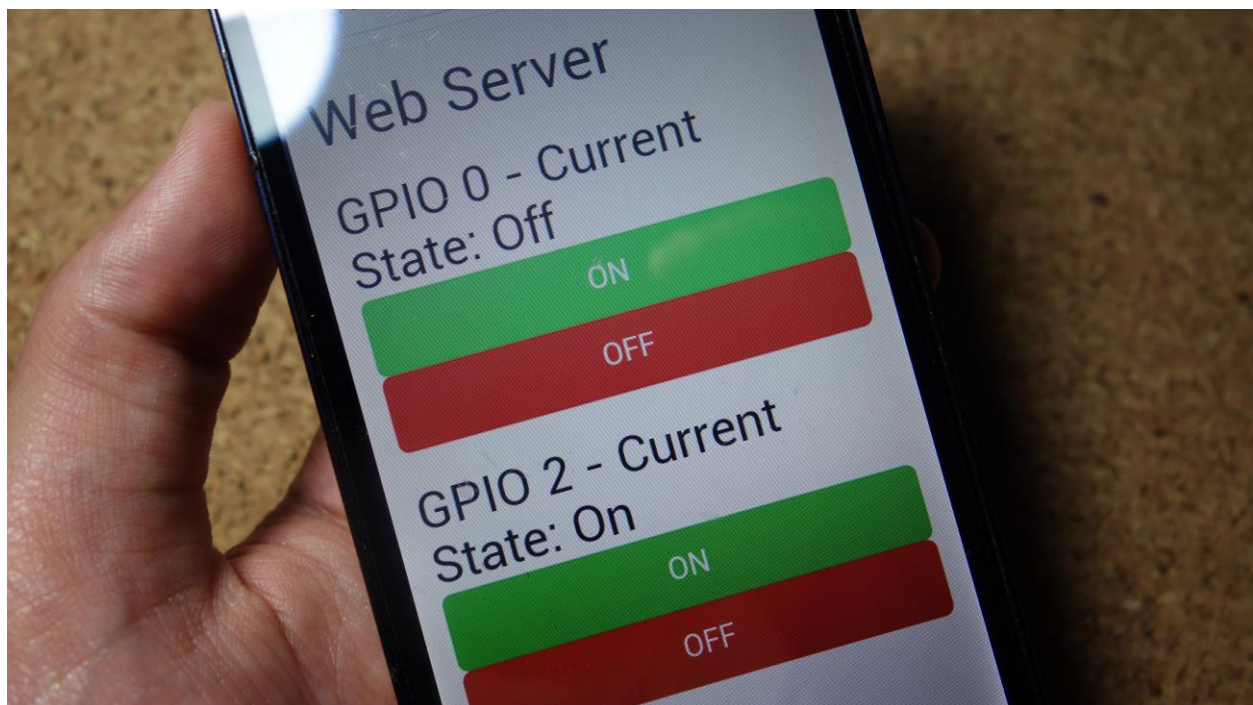
Password Protected Web Server with ESP8266

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.

This web server will be password protected and accessible from anywhere in the world.

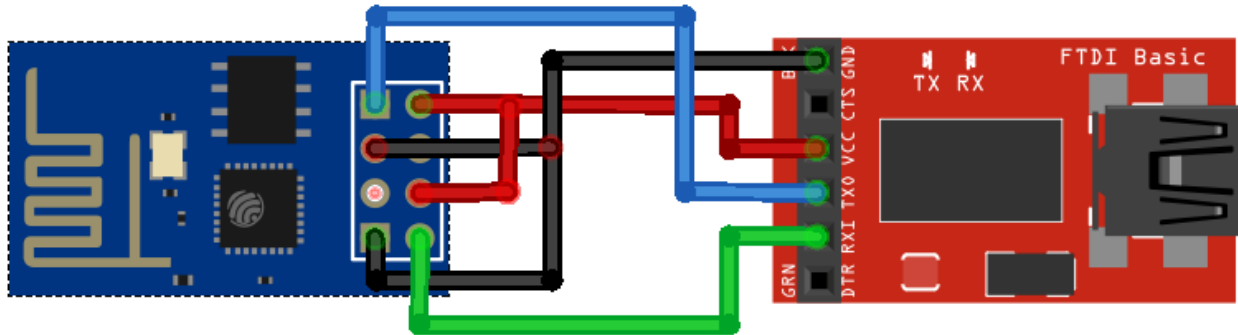
Spoiler Alert: This is what you're going to achieve at the end of this project!



DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

Schematics

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



Writing Your Arduino Sketch

Let's create a web server that controls two outputs (GPIO 0 and GPIO 2).

You can download the Sketch for this project in the following link:

<https://gist.github.com/RuiSantosdotme/92d9a5775585f80cc97e>

The snippet of code below starts by including the ESP8266 library. Then, you configure your ESP8266 with your own credentials (*ssid* and *password*). You actually need to replace those two lines with your credentials, so that your ESP can connect to your network.

```
// Including the ESP8266 WiFi library
#include <ESP8266WiFi.h>

// Replace with your network details
const char* ssid = "YOUR_NETWORK_NAME";
const char* password = "YOUR_NETWORK_PASSWORD";
```

DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

The next thing to do is declaring your web server on port 8888. You do it like this:

```
// Web Server on port 8888
WiFiServer server(8888);
```

Create one variable *header* to store the header response of the request. Two variables *gpio0_state* and *gpio2_state* to store the current state of the ESP GPIOs. Two variables (*gpio0_pin* and *gpio2_pin*) which refer to GPIO 0 and GPIO 2 respectively.

```
// variables
String header;
String gpio0_state = "Off";
String gpio2_state = "Off";
int gpio0_pin = 0;
int gpio2_pin = 2;
```

Now, let's go ahead and create your *setup()* function (it only runs once when your ESP first boots):

```
void setup() {

}
```

Start a serial communication at a 115200 baudrate for debugging purposes. Then define your GPIOs as *OUTPUTs* and set them *LOW* by default.

```
// only runs once
void setup() {
  // Initializing serial prt for debugging purposes
  Serial.begin(115200);
  delay(10);

  // preparing GPIOs
  pinMode(gpio0_pin, OUTPUT);
  digitalWrite(gpio0_pin, LOW);
  pinMode(gpio2_pin, OUTPUT);
  digitalWrite(gpio2_pin, LOW);
}
```

Now with the snippet of code below you begin the WiFi connection, you give time for your ESP to connect to your network and to print its IP address in the Serial Monitor.

```
// Connecting to WiFi network
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");

// Starting the web server
server.begin();
Serial.println("Web server running. Waiting for the ESP IP...");
delay(10000);

// Printing the ESP IP address
Serial.println(WiFi.localIP());
}
```

Inside your *loop()* function you tell exactly what happens when a new client establishes a connection with the web server.

DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
 FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
 QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

Your code is always listening for new clients. When a new client connects, it starts a connection.

```
// runs over and over again
void loop() {
  // Listenning for new clients
  WiFiClient client = server.available();

  if (client) {
    Serial.println("New client");
```

There's a boolean variable *blank_line* to act as a control to help determine when the HTTP request ends. You also have a *while()* loop that will be running for as long as the client stays connected.

```
if (client) {
  Serial.println("New client");
  // boolean to locate when the http request ends
  boolean blank_line = true;
  while (client.connected()) {
    if (client.available()) {
```

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.

Right now your username is *user* and your password is *pass*. I'll show you how to change that in just a moment. If the user enters the correct username and password it shows your web page to control the ESP.

```
// Finding the right credential string
if(header.indexOf("dXNlcjpwYXNz") >= 0) {
  //successful login
```

This next snippet of code is what checks which button in your web page 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://192.168.1.70:8888/gpio0off>. Your code checks that URL and with some *if... else* statements it knows that you want your GPIO 0 (which is defined as *gpio0_pin*) to go *LOW*.

```
// Finding the right credential string
if(header.indexOf("dXNlcjpwYXNz") >= 0) {
    //successful login
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/html");
    client.println("Connection: close");
    client.println();
    // turns the GPIOs on and off
    if(header.indexOf("GET / HTTP/1.1") >= 0) {
        Serial.println("Main Web Page");
    }
    else if(header.indexOf("GET /gpio0on HTTP/1.1") >= 0){
        Serial.println("GPIO 0 On");
        gpio0_state = "On";
        digitalWrite(gpio0_pin, HIGH);
    }
    else if(header.indexOf("GET /gpio0off HTTP/1.1") >= 0){
        Serial.println("GPIO 0 Off");
        gpio0_state = "Off";
        digitalWrite(gpio0_pin, LOW);
    }
    else if(header.indexOf("GET /gpio2on HTTP/1.1") >= 0){
        Serial.println("GPIO 2 On");
        gpio2_state = "On";
        digitalWrite(gpio2_pin, HIGH);
    }
    else if(header.indexOf("GET /gpio2off HTTP/1.1") >= 0){
        Serial.println("GPIO 2 Off");
        gpio2_state = "Off";
        digitalWrite(gpio2_pin, LOW);
    }
}
```

You send your web page to the client using the `client.println()` function. 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*).

```
// your web page
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("<head>");
client.println("<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
client.println("<link rel=\"stylesheet\" href=\"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css\">");
client.println("</head><div class=\"container\">");
client.println("<h1>Web Server</h1>");
client.println("<h2>GPIO 0 - Current State: " + gpio0_state);
client.println("<div class=\"row\">");
client.println("<div class=\"col-md-2\"><a href=\"/gpio0on\" class=\"btn btn-block btn-lg btn-success\" role=\"button\">ON</a></div>");
client.println("<div class=\"col-md-2\"><a href=\"/gpio0off\" class=\"btn btn-block btn-lg btn-danger\" role=\"button\">OFF</a></div>");
client.println("</div>");
client.println("<h2>GPIO 2 - Current State: " + gpio2_state);
client.println("<div class=\"row\">");
client.println("<div class=\"col-md-2\"><a href=\"/gpio2on\" class=\"btn btn-block btn-lg btn-success\" role=\"button\">ON</a></div>");
client.println("<div class=\"col-md-2\"><a href=\"/gpio2off\" class=\"btn btn-block btn-lg btn-danger\" role=\"button\">OFF</a></div>");
client.println("</div></div></html>");
```

If for any reason you've entered the wrong credentials, it prints a text message in your browser saying "Authentication failed".

```
else {
    client.println("HTTP/1.1 401 Unauthorized");
    client.println("WWW-Authenticate: Basic realm=\"Secure\"");
    client.println("Content-Type: text/html");
    client.println();
    client.println("<html>Authentication failed</html>");
}
```

The final lines of code do the following: cleans the header variable, stops the loop and closes the connection.

```
        header = "";
        break;
    }
    if (c == '\n') {
        // when starts reading a new line
        blank_line = true;
    }
    else if (c != '\r') {
        // when finds a character on the current line
        blank_line = false;
    }
}
}
// closing the client connection
delay(1);
client.stop();
Serial.println("Client disconnected.");
}
}
```

Encoding Your Username and Password

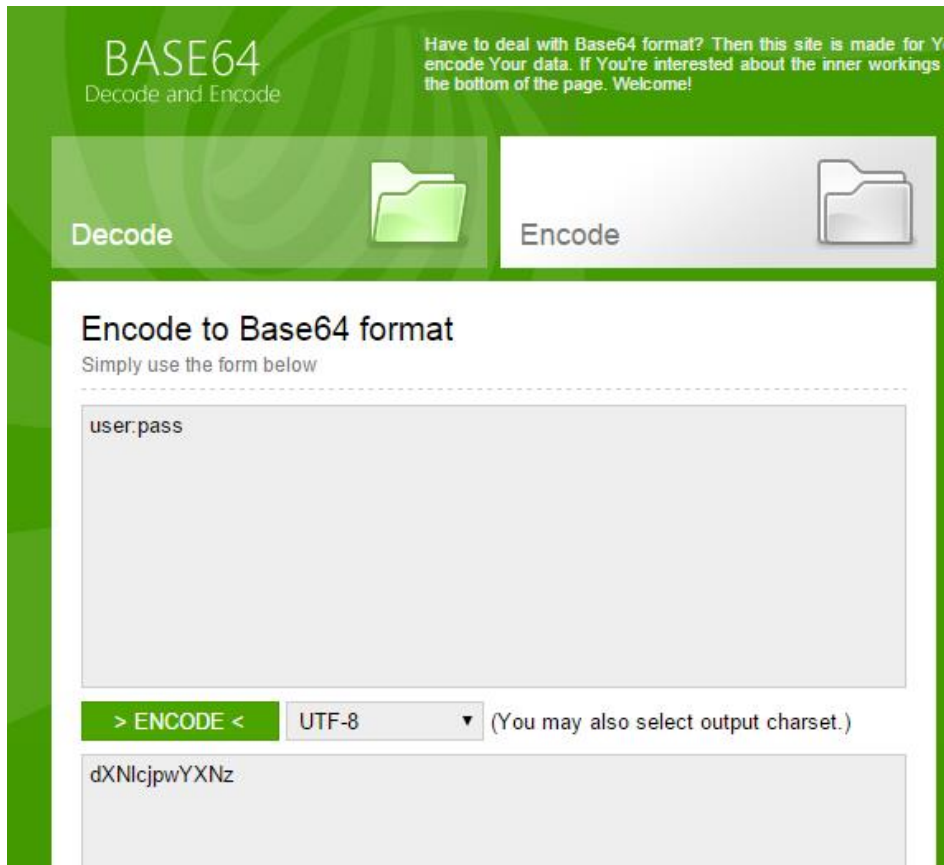
At this point if you upload the code created in the preceding section, your username is *user* and your password is *pass*. I'm sure you want to change and customize this example with your own credentials.

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 sketch that you downloaded.

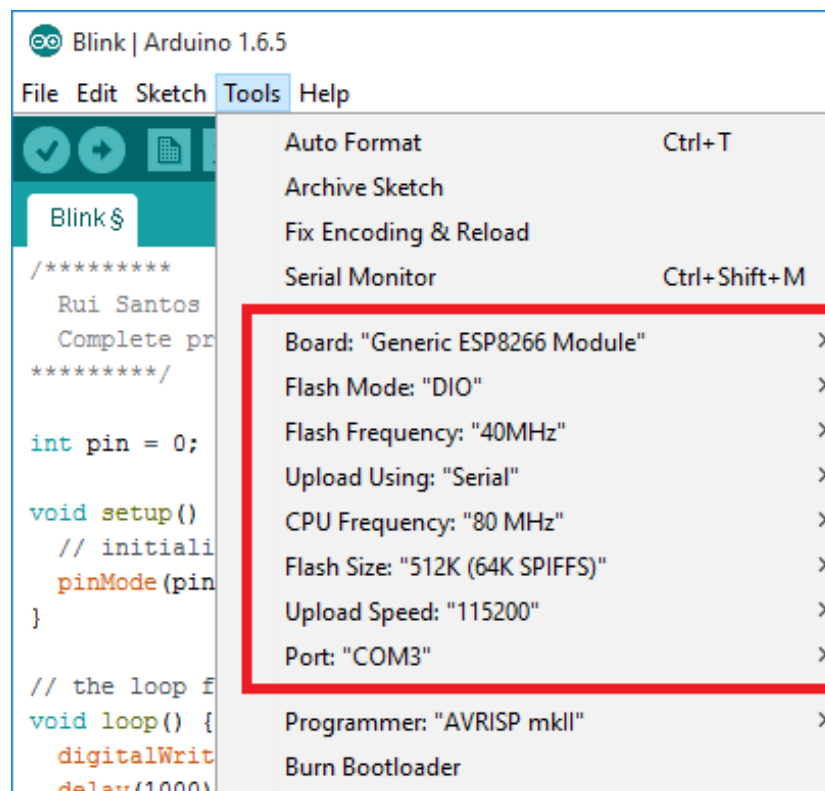
```
if(header.indexOf("dXNlcjpwYXNz") >= 0)
```

Tip: You can find that *if* statement above on line 80 from this Gist:
<https://gist.github.com/RuiSantosdotme/92d9a5775585f80cc97e>.

Uploading Code

Once you have your ESP8266+FTDI Programmer connected to your computer, go to the Arduino IDE.

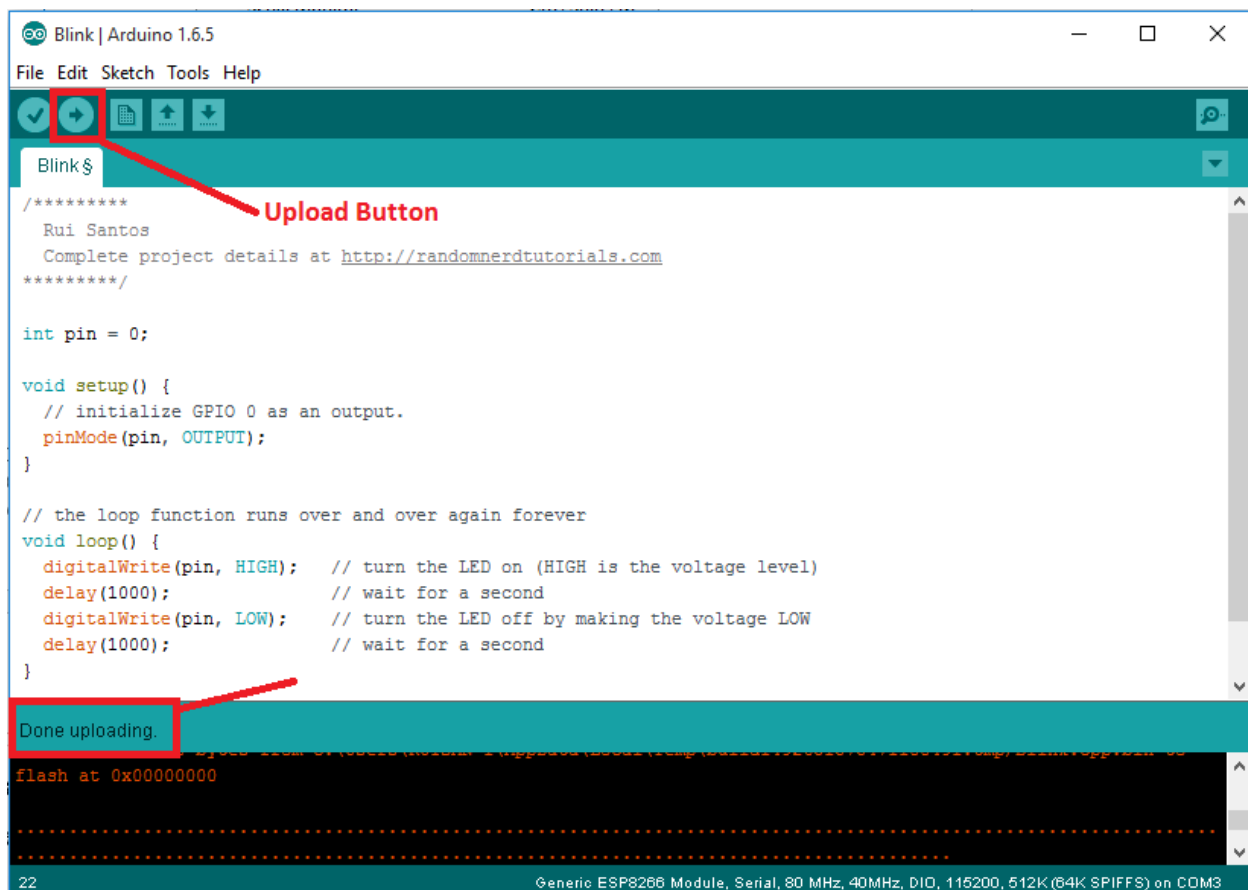
Look at the Tools menu and check all the configurations, by default they should look like this:



Important: Your COM port is very likely to be different from the screenshot above (Port: "COM3"). That's fine, because that doesn't interfere with anything. All the other configurations should look exactly like mine.

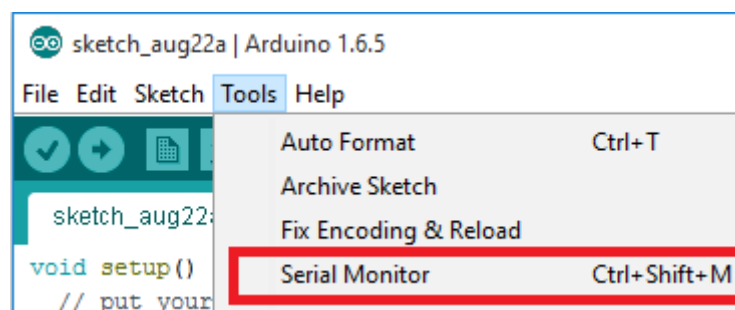
After checking the configurations click the "Upload Button" and wait a few seconds until you see the message "Done uploading." in the bottom left corner.

DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

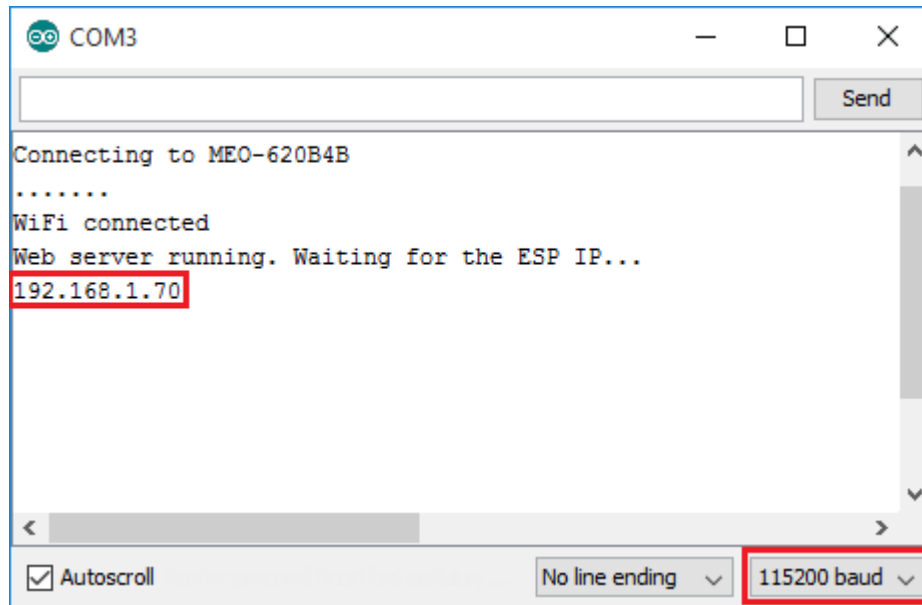


ESP8266 IP Address

After uploading your web server Sketch to your ESP, go to Tools > Serial Monitor. In your Serial Monitor window you're going to see your ESP IP address appearing when your ESP first boots.



The IP in my case it's: 192.168.1.70 (as shown in the Figure below). Your IP should be different, **save your ESP8266 IP** so you can access it later in this Unit.



Important: Set the Serial Monitor baudrate to 115200, otherwise you won't see anything.

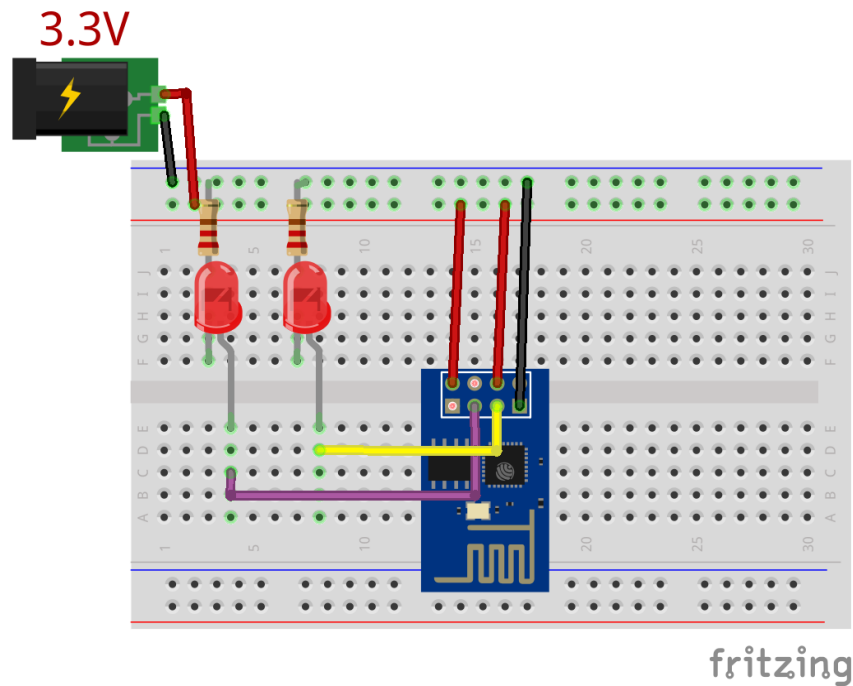
Troubleshooting: If your IP address doesn't show in your Serial Monitor Here's what you need to do:

1. Let the Serial Monitor stay opened
2. The FTDI Programmer should remain connected to your computer
3. Remove the power from your ESP (remove the 3.3V jumper wire).
4. Then plug it again to your ESP.

That restarts the ESP. The IP address should appear in your Serial Monitor screen after 10 seconds.

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).

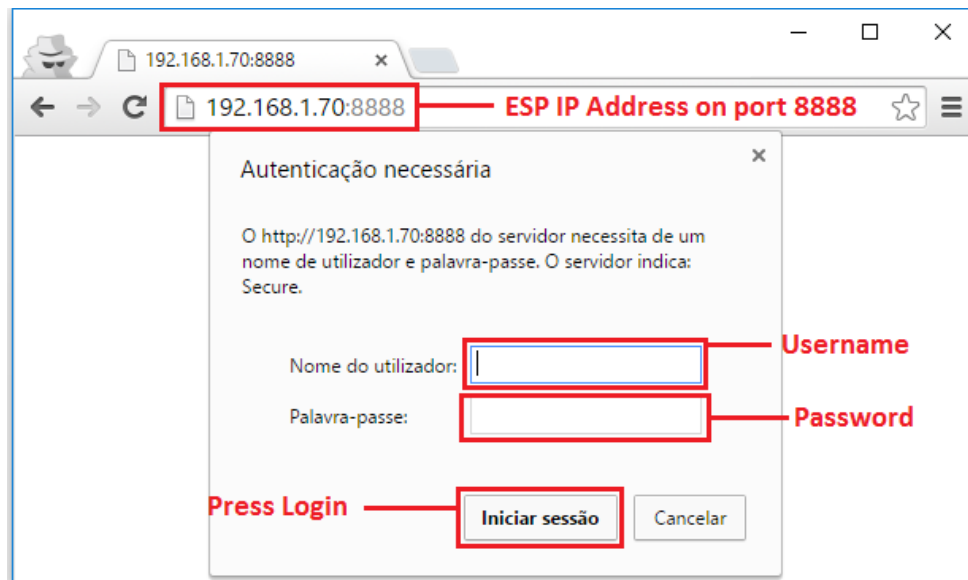


Accessing Your Web Server

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 the URL bar followed by :8888 (in my case: <http://192.168.1.70:8888>)

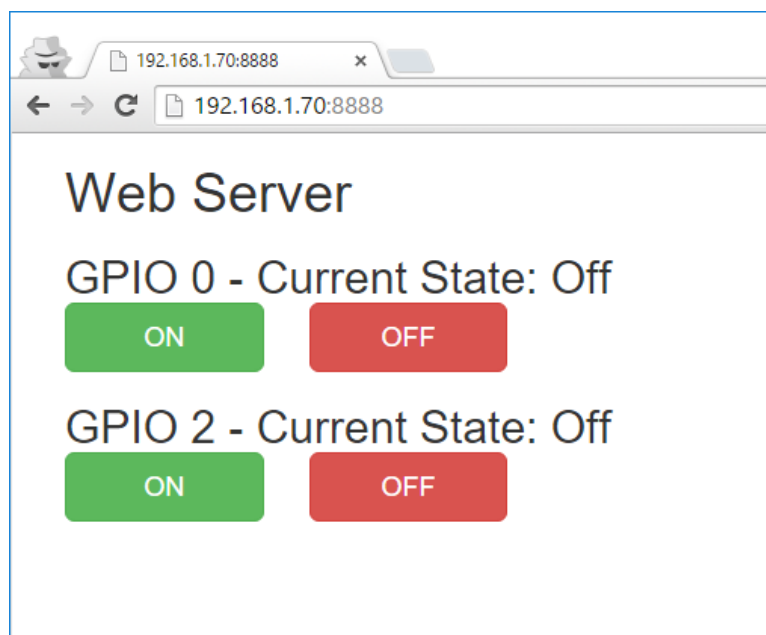
It should require that you enter your username and password in order to open your web server. And this is what you should see:



4. Enter your username and password

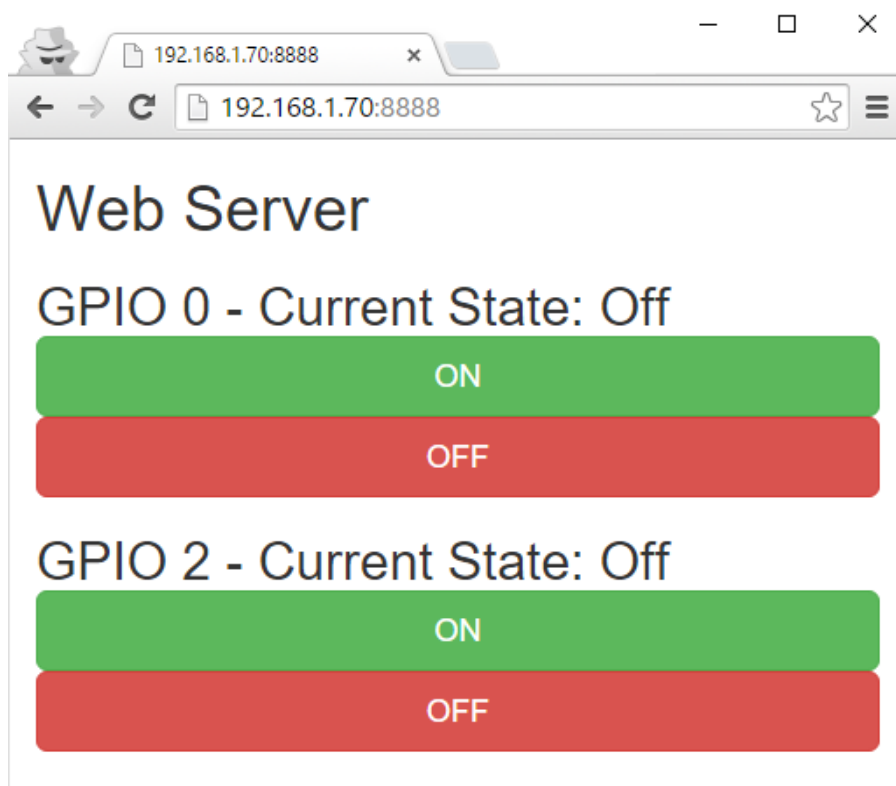
5. Press Login

A web page like the one below should appear.



Note: At this point, 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!



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>

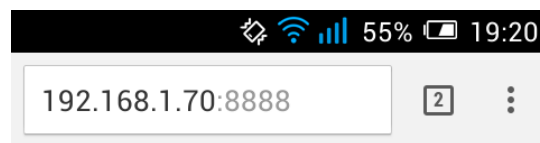
The preceding link takes you to an Instructable that shows how to control sockets using a Raspberry Pi, but you can apply the same concepts to your ESP module.

DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

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

Unit 7

Making Your Web Server Accessible from Anywhere in the World



Web Server

GPIO 0 - Current
State: Off

ON

OFF

GPIO 2 - Current
State: Off

ON

OFF

Making Your Web Server Accessible from Anywhere in the World

In this Unit you're going to make your web server accessible from anywhere in the world. You'll be using a free service to create a secure tunnel to your ESP which is running in your localhost. You don't need to do any port forwarding.

This service is called ngrok and it's free. Go to <https://ngrok.com/> to create your account. Click the green "Sign up" button:

ngrok

Home

Download

Docs

Product

FAQ

Login

Secure tunnels to localhost

"I want to expose a local server behind a NAT or firewall to the internet."

Download

Sign up

Enter your details in the forms in the left box (as shown in the Figure below).

Sign up

Your Name

Your Email

Confirm Email

Password

Sign up



Sign up with Github



Sign in with Google

Why should I sign up?

Signing up is free! Many useful features are only available after you sign up, including:

Password Protected

Set http auth credentials to protect access to your tunnel and those you share it with.

```
ngrok http -auth "user:password" 80
```

TCP Tunnels

Expose any networked service to the internet, even ones that don't use HTTP.

```
ngrok tcp 22
```

Multiple Simultaneous Tunnels

Run multiple tunnels simultaneously with a single ngrok client.

After creating your account, login and go to the main dashboard to find your Tunnel Authtoken. Copy your unique Authtoken to a safe place (you'll need them later in this Unit).

Rui / ngrok

Main Reserved Auth Team Admin Billing

Welcome to ngrok 2.0!

ngrok 2.0 is a ground-up rewrite of ngrok with a focus on quality, stability, better performance and highly-requested features that make it suitable for use from development all the way through to production use cases.

ngrok 2.0 introduces long-awaited features including support for TLS multiplexed tunnels, wildcard domains, reserved TCP addresses, RESTful APIs for tunnel status and dynamically controlling clients, plus many other improvements:

What's New in 2.0

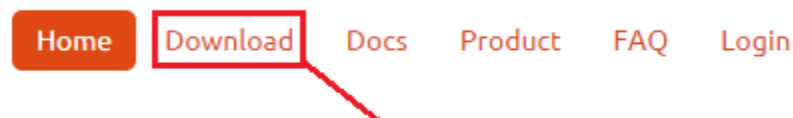
Your Tunnel Authtoken

3V1MfHcNMD9rhBizf8TRs_2whamY91tqX4 Copy

You only need to do this one time.

```
./ngrok authtoken 3V1MfHcNMD9rhBizf8TRs_2whamY91tqX4
```

Then go to the Download tab in the navigation bar and select “Download”:



Then choose your operating system and download the ngrok software.

Download and Installation

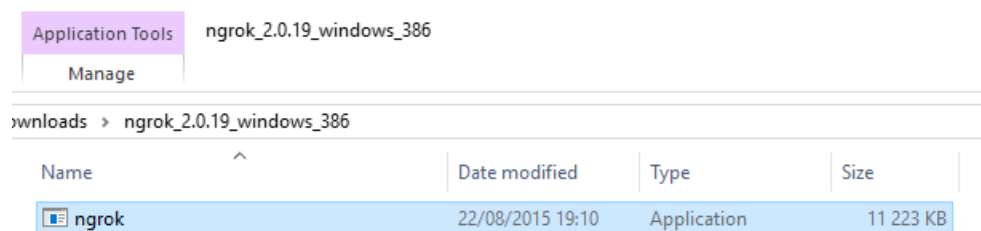
ngrok is easy to install. Download a single binary with *zero run-time dependencies* for any major platform. Unzip it and then run it from the command line.

Step 1: Download ngrok

Mac OS X	Download
Windows	Download
Linux	Download
Linux/ARM	Download
FreeBSD	Download

Unzip the folder that you have just downloaded and open the ngrok application.

Note: If you're on Linux open the terminal and type `./ngrok --help` to open the ngrok application.



You should see a window similar to the one below:

```
C:\Users\Rui Santos\Downloads\ngrok_2.0.19_windows_386\ngrok.exe

EXAMPLES:
  ngrok http 80                # secure public URL for port 80 web server
  ngrok http -subdomain=baz 8080 # port 8080 available at baz.ngrok.io
  ngrok http foo.dev:80        # tunnel to host:port instead of localhost
  ngrok tcp 22                 # tunnel arbitrary TCP traffic to port 22
  ngrok tls -hostname=foo.com 443 # TLS traffic for foo.com to port 443
  ngrok start foo bar baz      # start tunnels from the configuration file

VERSION:
  2.0.19

AUTHOR:
  inconshreveable - <alan@ngrok.com>

COMMANDS:
  authtoken  save authtoken to configuration file
  credits    prints author and licensing information
  http       start an HTTP tunnel
  start      start tunnels by name from the configuration file
  tcp        start a TCP tunnel
  test       test ngrok service end-to-end
  tls        start a TLS tunnel
  update     update to the latest version
  version    print the version string
  help       Shows a list of commands or help for one command

ngrok is a command line application, try typing 'ngrok.exe http 80'
at this terminal prompt to expose port 80.
C:\Users\Rui Santos\Downloads\ngrok_2.0.19_windows_386>
```

Now in your terminal enter the following command and replace the red text with your own IP address and ngrok's tunnel Authtoken:

```
ngrok tcp 192.168.1.70:8888 --authtoken 3V1MfHcNMD9rhBizf8TRs_2whamY91tqX4
```

Here's how it should look like, press Enter to run this command.

```
ers\Rui Santos\Downloads\ngrok_2.0.19_windows_386\ngrok.exe

s\Rui Santos\Downloads\ngrok_2.0.19_windows_386 ngrok tcp 192.168.1.70:8888 --authtoken 3V1MfHcNMD9rhBizf8TRs_2whamY91tqX4
```

If everything runs smoothly, you should notice that your Tunnel is online and a URL should be in your terminal.

```
C:\Users\Rui Santos\Downloads\nngrok_2.0.19_windows_386\nngrok.exe - ngrok tcp 192.168.1.70:8...
ngrok by @inconshreveable (Ctrl+C to quit)

Tunnel Status      online
Version            2.0.19/2.0.19
Web Interface       http://127.0.0.1:4040
Forwarding          tcp://0.tcp.ngrok.io:54626 -> 192.168.1.70:8888
Connections

  ttl    opn    rt1    rt5    p50    p90
    0     0     0.00   0.00   0.00   0.00
```

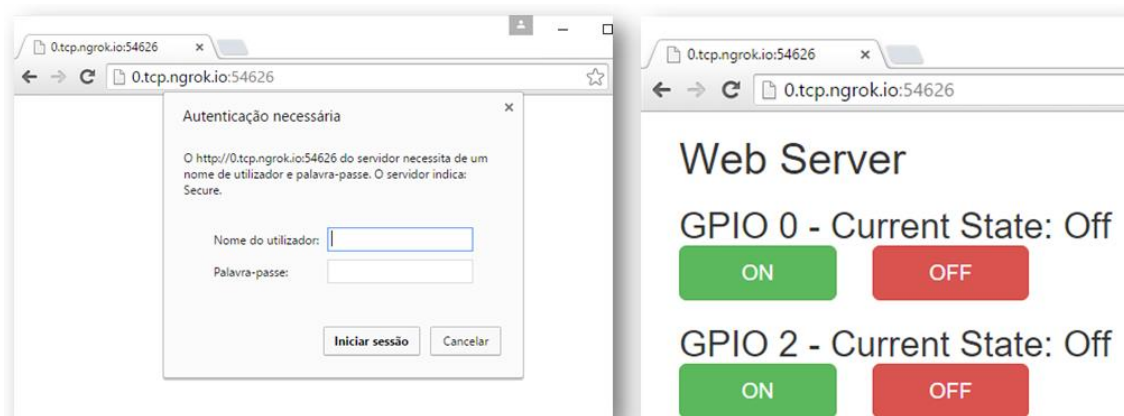
Now you can access your web server from anywhere in the world by typing your unique URL (in my case <http://0.tcp.ngrok.io:54626/>) in a browser.

Note: Even though it's a tcp connection you type http in your web browser.

Important: You need to let your computer on and running ngrok in order to maintain the tunnel online.

Troubleshooting: If you go to your ngrok.io URL and nothing happens. Open your ESP IP in your browser to see if your web server is still running. If it's still running make sure you've entered the right IP address and authtoken on the ngrok command executed earlier.

You'll always be asked to enter your username and password to open your web server.



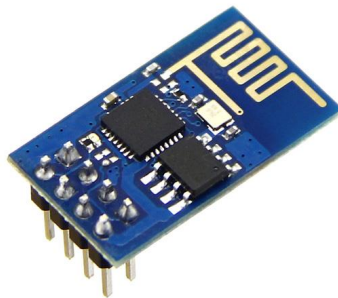
DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

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.

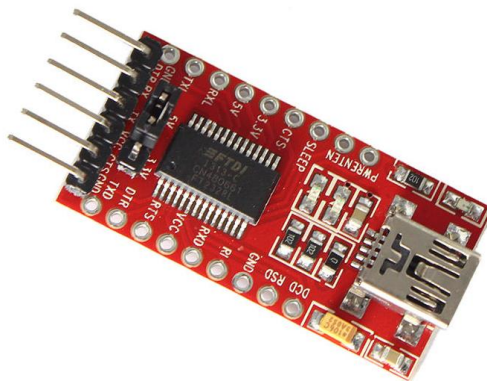
[Visit this link to buy ESP8266 on eBay:](#)

<http://randomnerdtutorials.com/ebay-esp8266>



[Visit this link to buy an FTDI Programmer on eBay:](#)

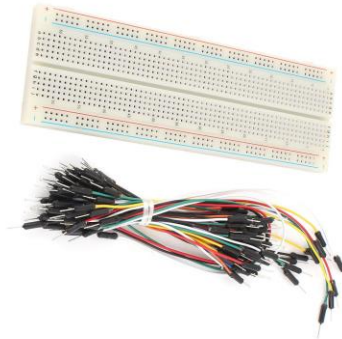
<http://randomnerdtutorials.com/ebay-ftdi-programmer>



DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

[Visit this link to buy a breadboard on eBay:](#)

<http://randomnerdtutorials.com/ebay-breadboard>



DOWNLOAD OTHER RNT PRODUCTS: [HTTP://RANDOMNERDTUTORIALS.COM/PRODUCTS](http://RANDOMNERDTUTORIALS.COM/PRODUCTS)
FOR MORE PROJECTS AND TUTORIALS GO TO: [HTTP://RANDOMNERDTUTORIALS.COM/](http://RANDOMNERDTUTORIALS.COM/)
QUESTIONS? VISIT OUR PRIVATE FACEBOOK GROUP: [HTTP://RANDOMNERDTUTORIALS.COM/FB](http://RANDOMNERDTUTORIALS.COM/FB)

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 password protected web server using the ESP8266.

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

- Use the ESP8266
- Establish a serial communication
- Create a password protected web server to control any output
- Make your web server accessible from anywhere in the world

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](#)).

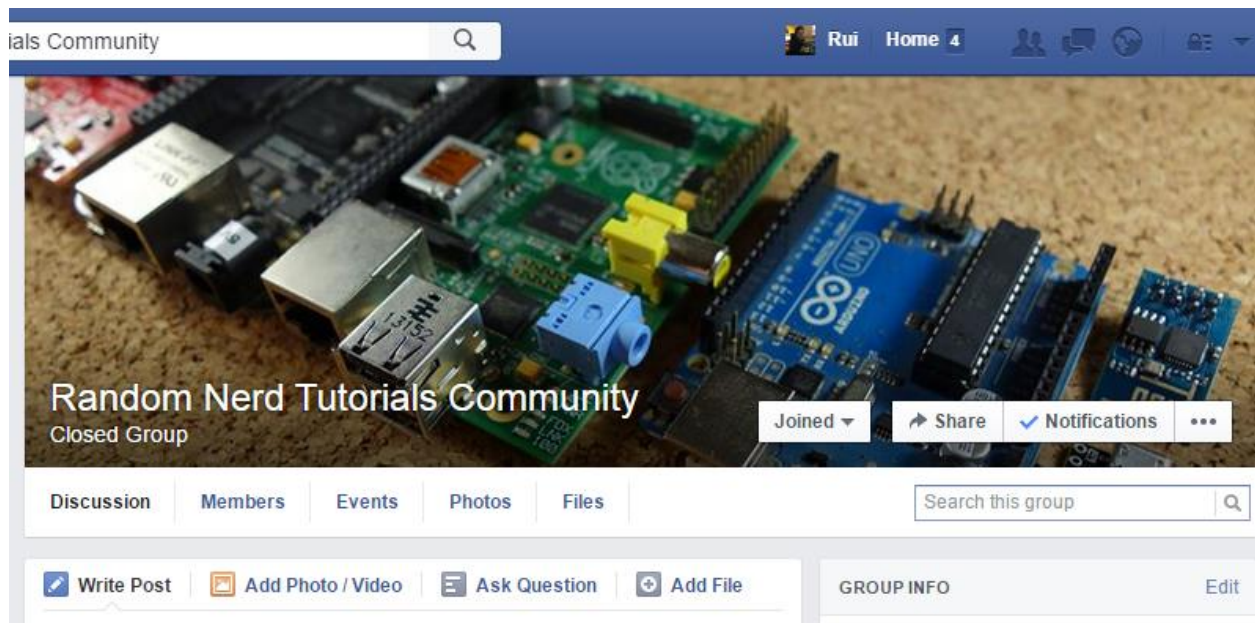
Good luck with all your projects,

-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/>

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>