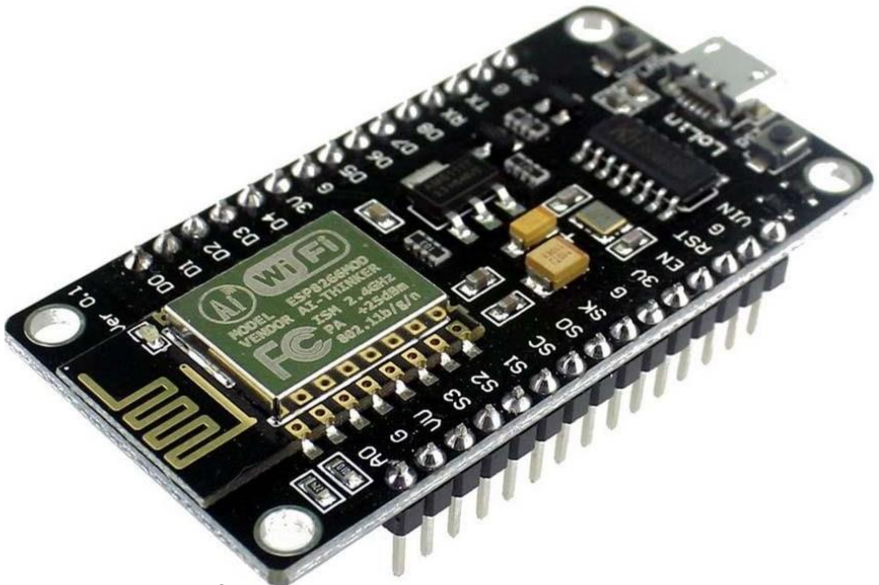
ESP8266 WiFi



The ESP8266 is the name of a micro controller designed by Espressif Systems. The ESP8266 itself is a self-contained WiFi networking solution offering as a bridge from existing micro controller to WiFi and is also capable of running self-contained applications.

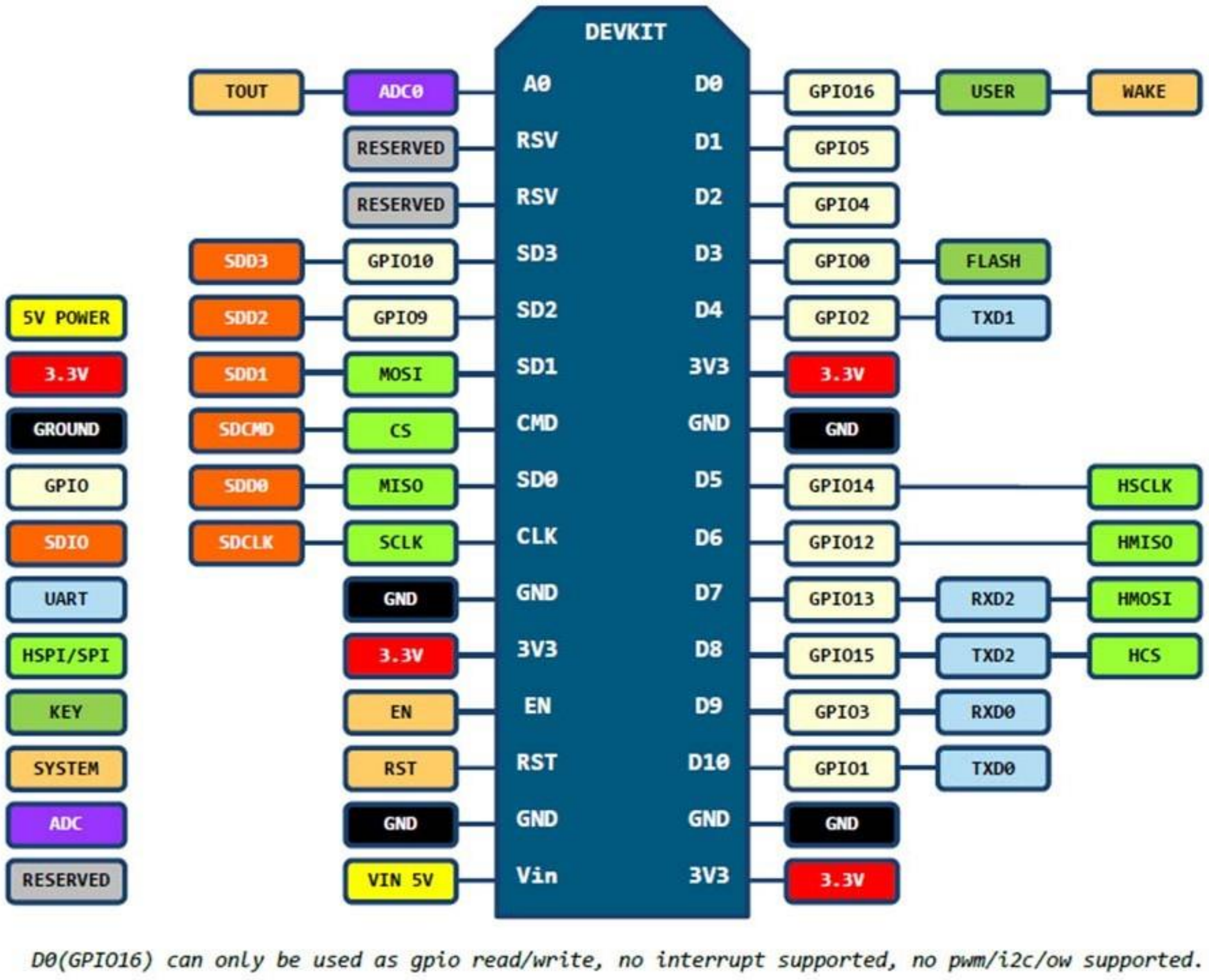
This module comes with a built in USB connector and a rich assortment of pin-outs. With a micro USB cable, you can connect NodeMCU devkit to your laptop and flash it without any trouble, just like Arduino. It is also immediately breadboard friendly.

Table of Contents

1. Specification:

- Voltage:3.3V.
- Wi-Fi Direct (P2P), soft-AP.
- Current consumption: 10uA~170mA.
- Flash memory attachable: 16MB max (512K normal).
- Integrated TCP/IP protocol stack.
- Processor: Tensilica L106 32-bit.
- Processor speed: 80~160MHz.
- RAM: 32K + 80K.
- GPIOs: 17 (multiplexed with other functions).
- Analog to Digital: 1 input with 1024 step resolution.
- +19.5dBm output power in 802.11b mode
- 802.11 support: b/g/n.
- Maximum concurrent TCP connections: 5.

2. Pin Definition:



D0(GPIO16) can only be used as gpio read/write, no interrupt supported, no pwm/i2c/ow supported.

3. Using Arduino IDE

The most basic way to use the ESP8266 module is to use serial commands, as the chip is basically a WiFi/Serial transceiver. However, this is not convenient. What we recommend is using the very cool Arduino ESP8266 project, which is a modified version of the Arduino IDE that you need to install on your computer. This makes it very convenient to use the ESP8266 chip as we will be using the well-known Arduino IDE. Following the below step to install ESP8266 library to work in Arduino IDE environment.

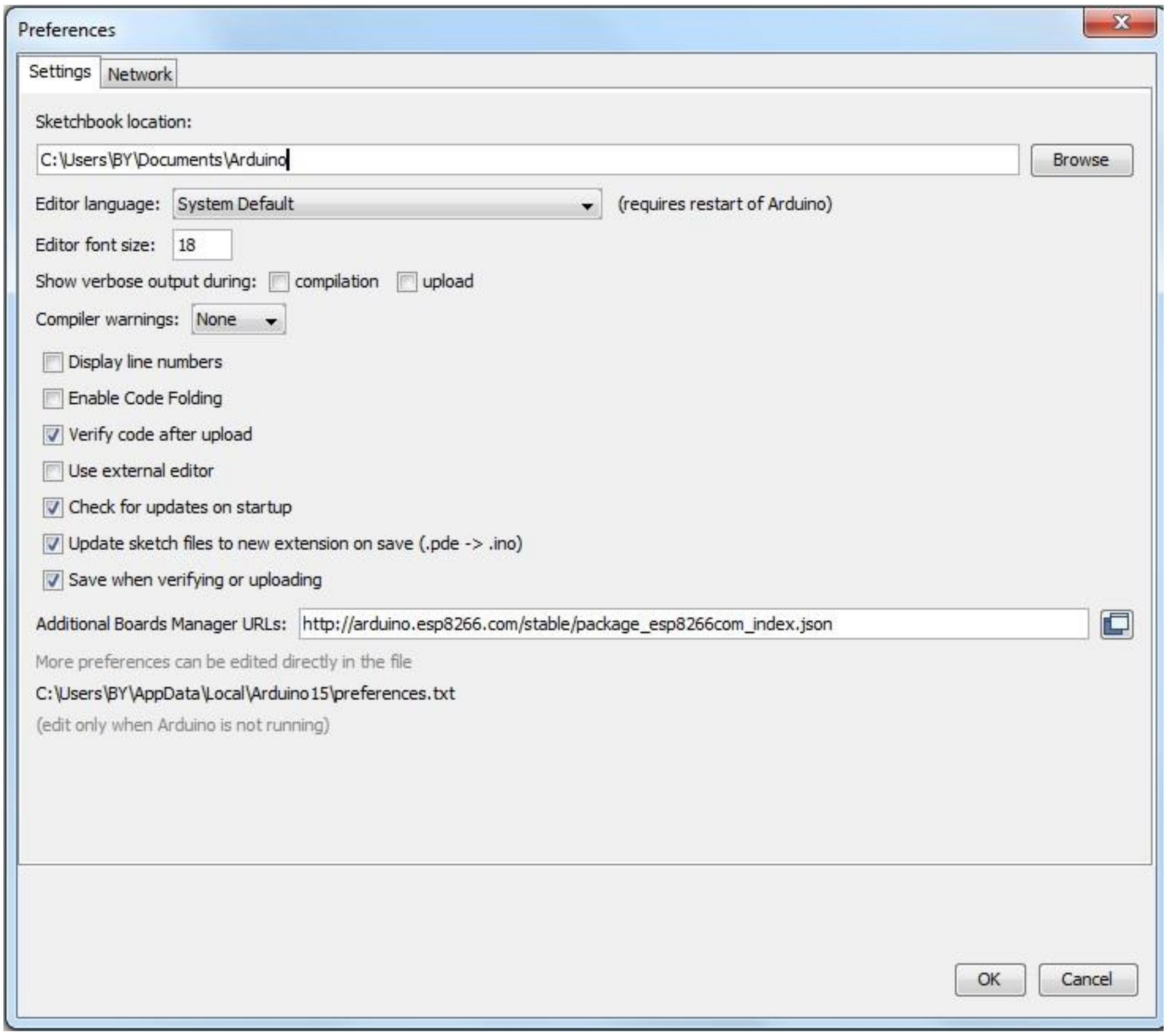
Install the Arduino IDE 1.6.4 or greater

Download Arduino IDE from Arduino.cc (1.6.4 or greater) - don't use 1.6.2 or lower version! You can use your existing IDE if you have already installed it.

You can also try downloading the ready-to-go package from the ESP8266-Arduino project, if the proxy is giving you problems.

Install the ESP8266 Board Package

Enter ***http://arduino.esp8266.com/stable/package_esp8266com_index.json*** into *Additional Board Manager URLs* field in the Arduino v1.6.4+ preferences.



Sketchbook location: C:\Users\BY\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 18

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☐ Display line numbers

☐ Enable Code Folding

☑ Verify code after upload

☐ Use external editor

☑ Check for updates on startup

☑ Update sketch files to new extension on save (.pde -> .ino)

☑ Save when verifying or uploading

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

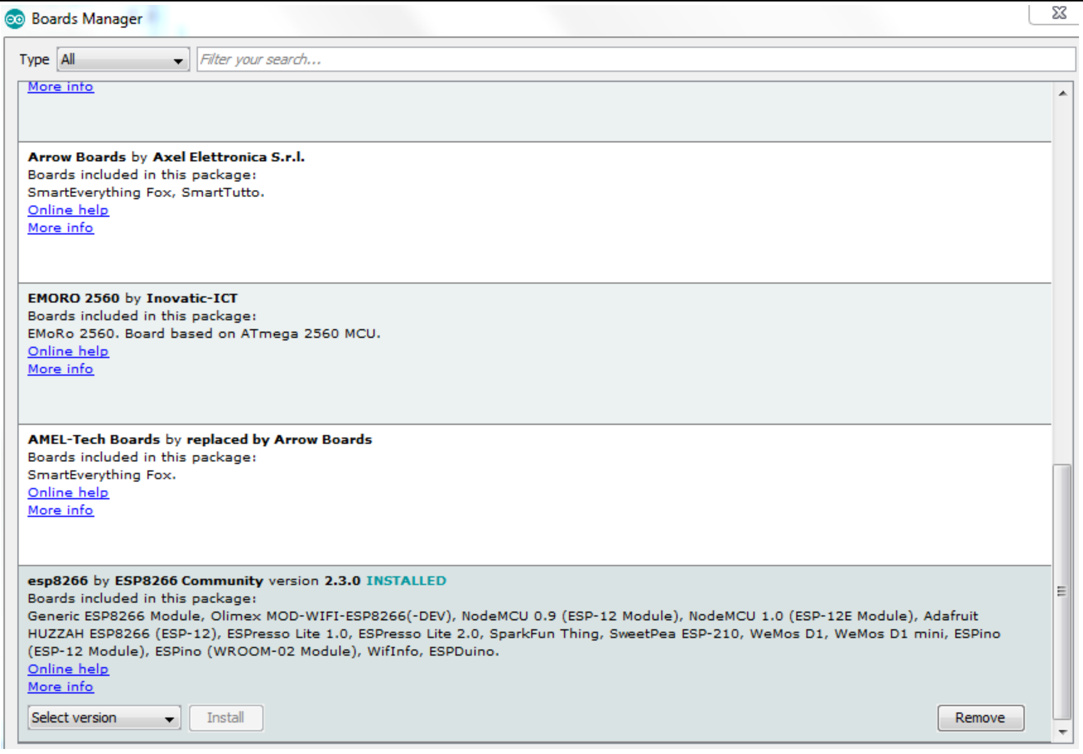
More preferences can be edited directly in the file

C:\Users\BY\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

Click 'File' -> 'Preferences' to access this panel.

Next, use the Board manager to install the ESP8266 package.


Boards Manager

Type All | Filter your search...

More info

Arrow Boards by **Axel Elettronica S.r.l.**
Boards included in this package:
SmartEverything Fox, SmartTutto.
Online help
More info

EMORO 2560 by **Inovatic-ICT**
Boards included in this package:
EMoRo 2560. Board based on ATmega 2560 MCU.
Online help
More info

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

esp8266 by **ESP8266 Community** version **2.3.0** **INSTALLED**
Boards included in this package:
Generic ESP8266 Module, Olimex MOD-WIFI-ESP8266(-DEV), NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Adafruit HUZZAH ESP8266 (ESP-12), ESPresso Lite 1.0, ESPresso Lite 2.0, SparkFun Thing, SweetPea ESP-210, WeMos D1, WeMos D1 mini, ESPino (ESP-12 Module), ESPino (WROOM-02 Module), WifInfo, ESPDuino.
Online help
More info

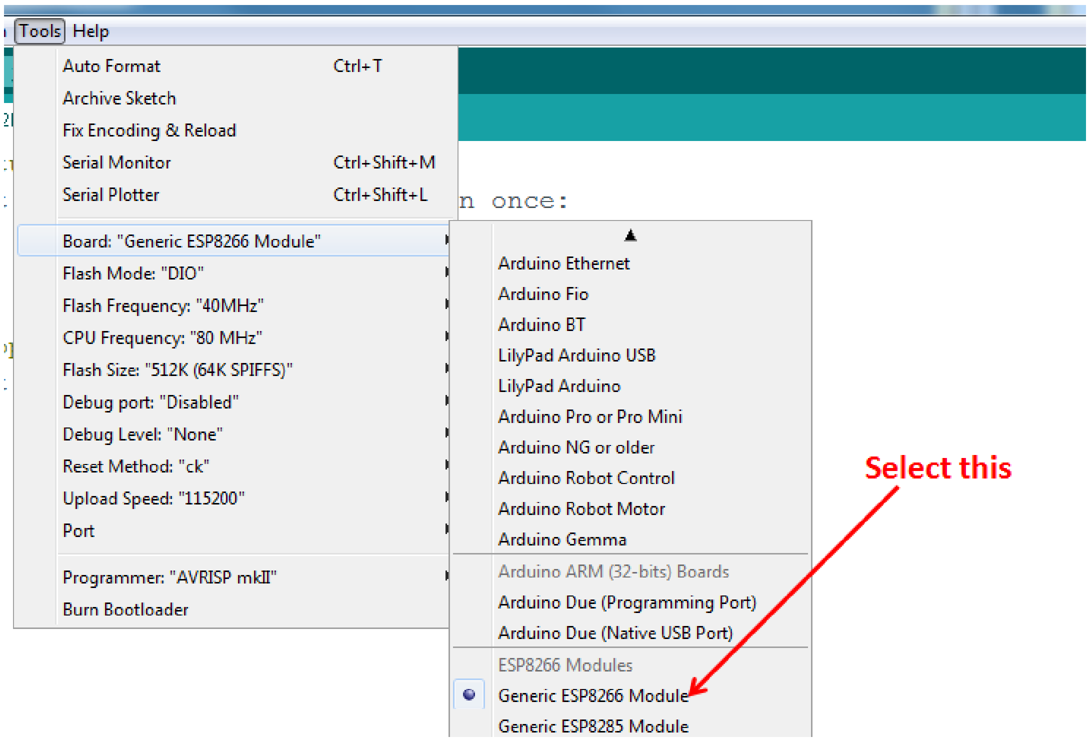
Select version | Install | Remove

Click 'Tools' -> 'Board:' -> 'Board Manager…' to access this panel.

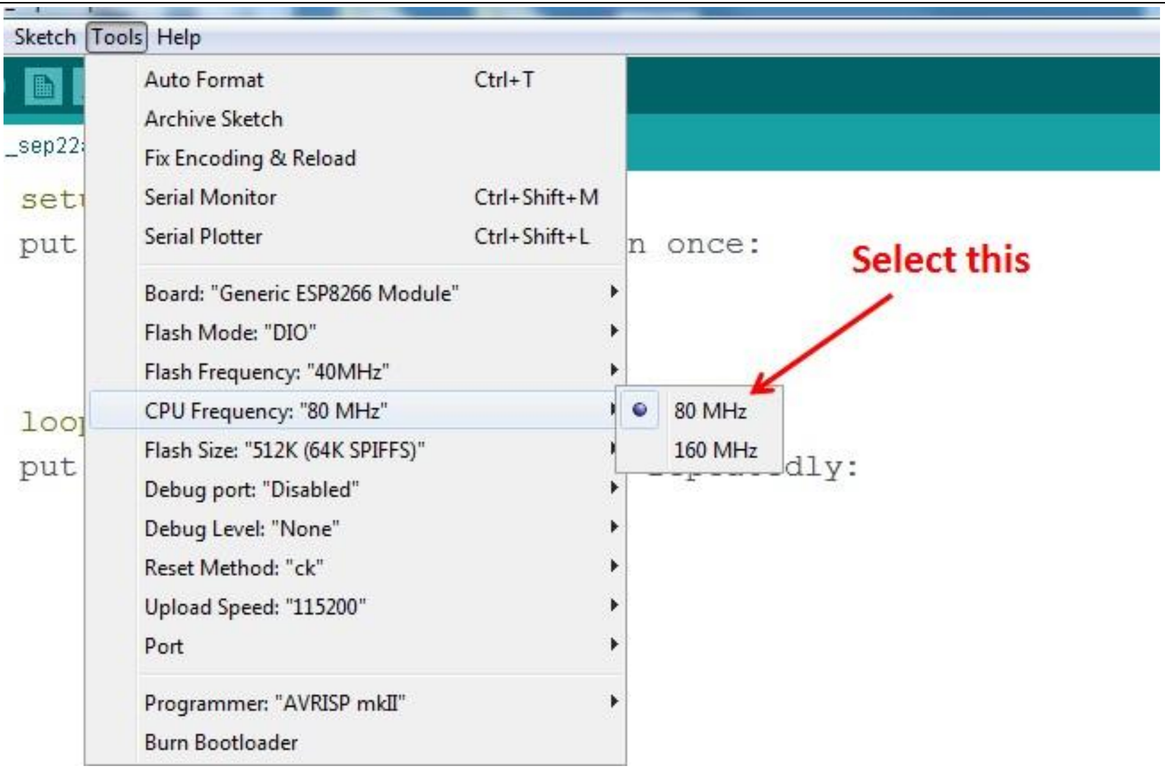
Scroll down to ' esp8266 by ESP8266 Community ' and click "Install" button to install the ESP8266 library package. Once installation completed, close and re-open Arduino IDE for ESP8266 library to take effect.

Setup ESP8266 Support

When you've restarted Arduino IDE, select 'Generic ESP8266 Module' from the 'Tools' -> 'Board:' dropdown menu.


Board: "Generic ESP8266 Module"
Flash Mode: "DIO"
Flash Frequency: "40MHz"
CPU Frequency: "80 MHz"
Flash Size: "512K (64K SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
Reset Method: "ck"
Upload Speed: "115200"
Port
Programmer: "AVRISP mkII"

Select 80 MHz as the CPU frequency (you can try 160 MHz overclock later)

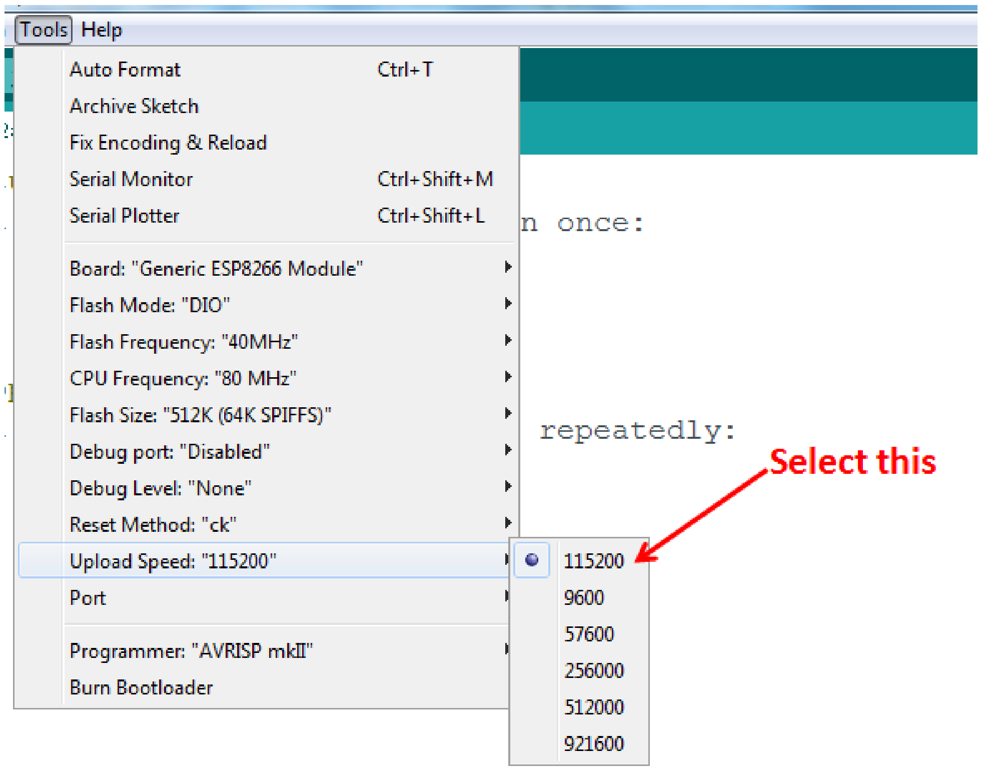


Sketch Tools Help

Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
Board: "Generic ESP8266 Module"
Flash Mode: "DIO"
Flash Frequency: "40MHz"
CPU Frequency: "80 MHz"
Flash Size: "512K (64K SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
Reset Method: "ck"
Upload Speed: "115200"
Port
Programmer: "AVRISP mkII"
Burn Bootloader

● 80 MHz
160 MHz

Select '115200' baud upload speed is a good place to start - later on you can try higher speeds but 115200 is a good safe place to start.

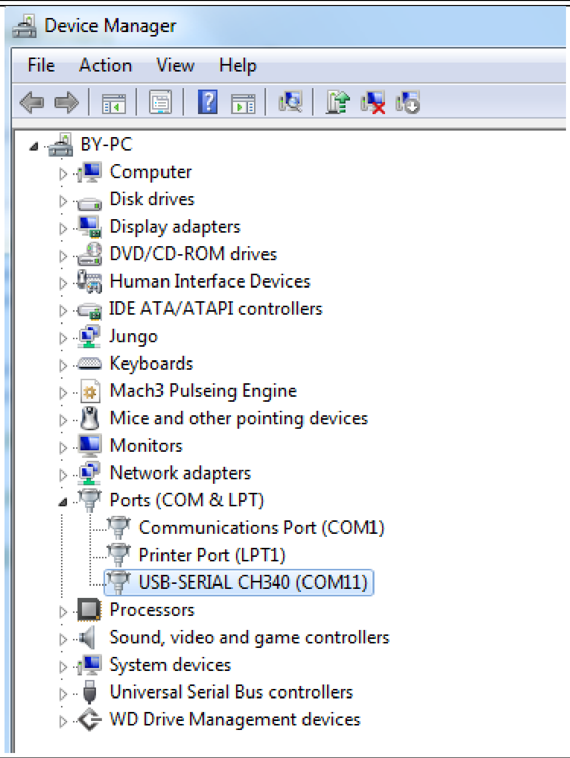


Tools Help

Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
Board: "Generic ESP8266 Module"
Flash Mode: "DIO"
Flash Frequency: "40MHz"
CPU Frequency: "80 MHz"
Flash Size: "512K (64K SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
Reset Method: "ck"
Upload Speed: "115200"
Port
Programmer: "AVRISP mkII"
Burn Bootloader

● 115200
9600
57600
256000
512000
921600

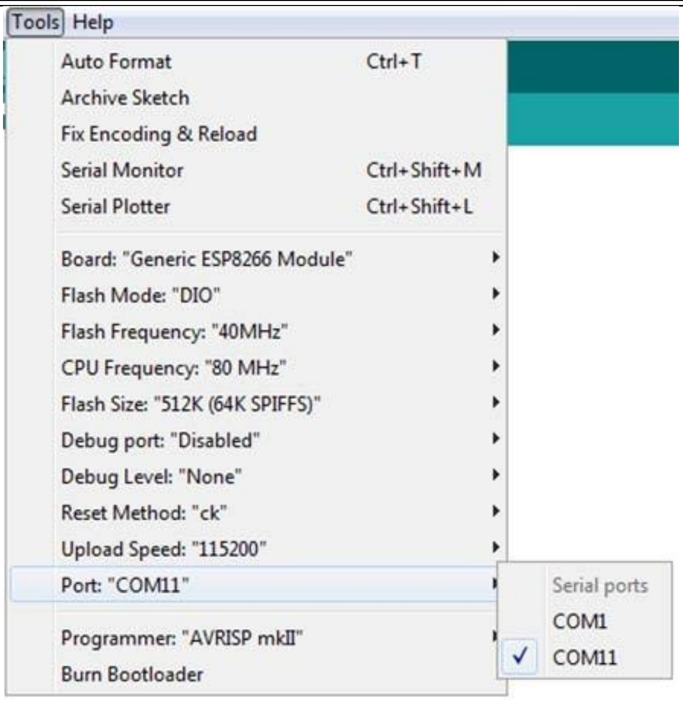
Go to your Windows 'Device Manager' to find out which Com Port 'USB-Serial CH340' is assigned to. Select the matching COM/serial port for your CH340 USB-Serial interface.



Device Manager

File Action View Help

- BY-PC
  - Computer
  - Disk drives
  - Display adapters
  - DVD/CD-ROM drives
  - Human Interface Devices
  - IDE ATA/ATAPI controllers
  - Jungo
  - Keyboards
  - Mach3 Pulseing Engine
  - Mice and other pointing devices
  - Monitors
  - Network adapters
  - Ports (COM & LPT)
    - Communications Port (COM1)
    - Printer Port (LPT1)
    - USB-SERIAL CH340 (COM11)
  - Processors
  - Sound, video and game controllers
  - System devices
  - Universal Serial Bus controllers
  - WD Drive Management devices



Tools Help

Auto Format	Ctrl+T
Archive Sketch	
Fix Encoding & Reload	
Serial Monitor	Ctrl+Shift+M
Serial Plotter	Ctrl+Shift+L
Board: "Generic ESP8266 Module"	
Flash Mode: "DIO"	
Flash Frequency: "40MHz"	
CPU Frequency: "80 MHz"	
Flash Size: "512K (64K SPIFFS)"	
Debug port: "Disabled"	
Debug Level: "None"	
Reset Method: "ck"	
Upload Speed: "115200"	
Port: "COM11"	
Programmer: "AVRISP mkII"	
Burn Bootloader	

Serial ports

COM1

✓ COM11

Find out which Com Port is assign for CH340	Select the correct Com Port as indicated on 'Device Manager"

Note: if this is your first time using CH340 " USB-to-Serial " interface, please install the driver first before proceed the above Com Port setting. The CH340 driver can be download from the below site:

https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers

Blink Test

We'll begin with the simple blink test.

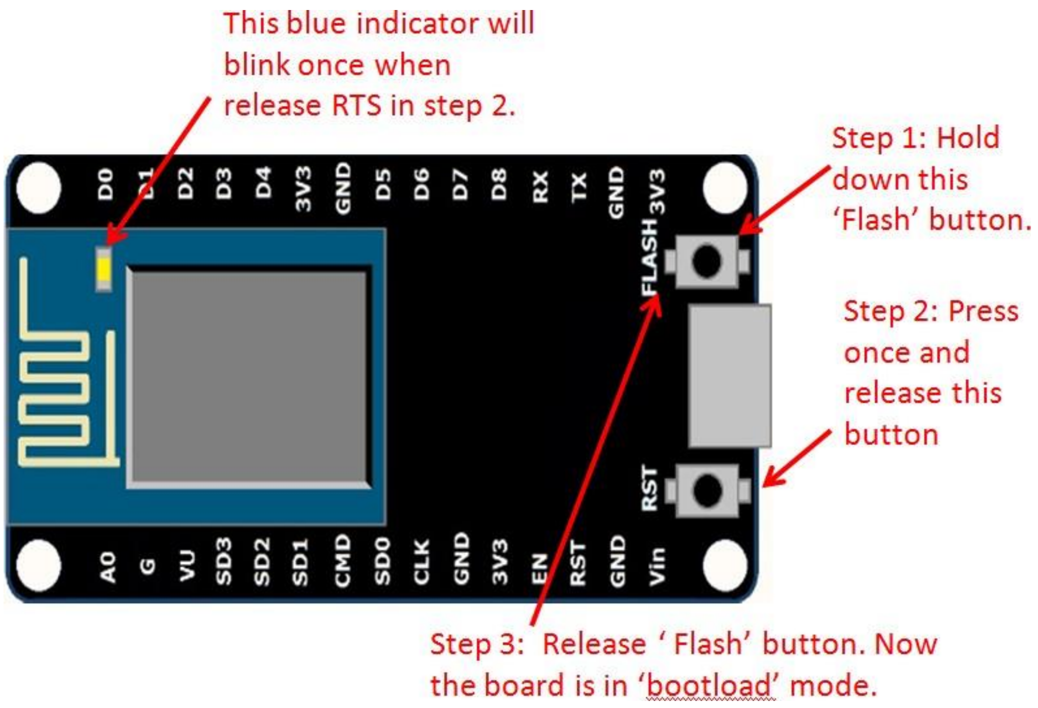
Enter this into the sketch window (and save since you'll have to). Connect a LED as shown in Figure3-1.

```
void setup() {
  pinMode(5, OUTPUT);    // GPIO05, Digital Pin D1
}

void loop() {
  digitalWrite(5, HIGH);
  delay(900);
  digitalWrite(5, LOW);
  delay(500);
}
```

Now you'll need to put the board into bootload mode. You'll have to do this before each upload. There is no timeout for bootload mode, so you don't have to rush!

- Hold down the 'Flash' button.
- While holding down ' Flash', press the 'RST' button.
- Release 'RST', then release 'Flash'

- When you release the 'RST' button, the blue indication will blink once, this means its ready to bootload.


This blue indicator will blink once when release RTS in step 2.
Step 1: Hold down this 'Flash' button.
Step 2: Press once and release this button
Step 3: Release ' Flash' button. Now the board is in 'bootload' mode.
D0
D1
D2
D3
D4
3V3
GND
D5
D6
D7
D8
RX
TX
GND
3V3
FLASH
RST
A0
G
VU
SD3
SD2
SD1
CMD
SD0
CLK
GND
3V3
EN
RST
GND
Vin

Once the ESP board is in bootload mode, upload the sketch via the IDE, Figure 3-2.

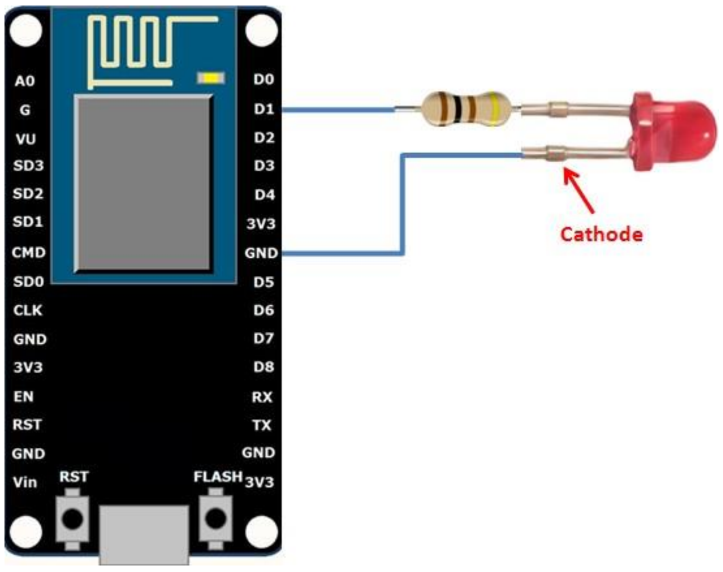
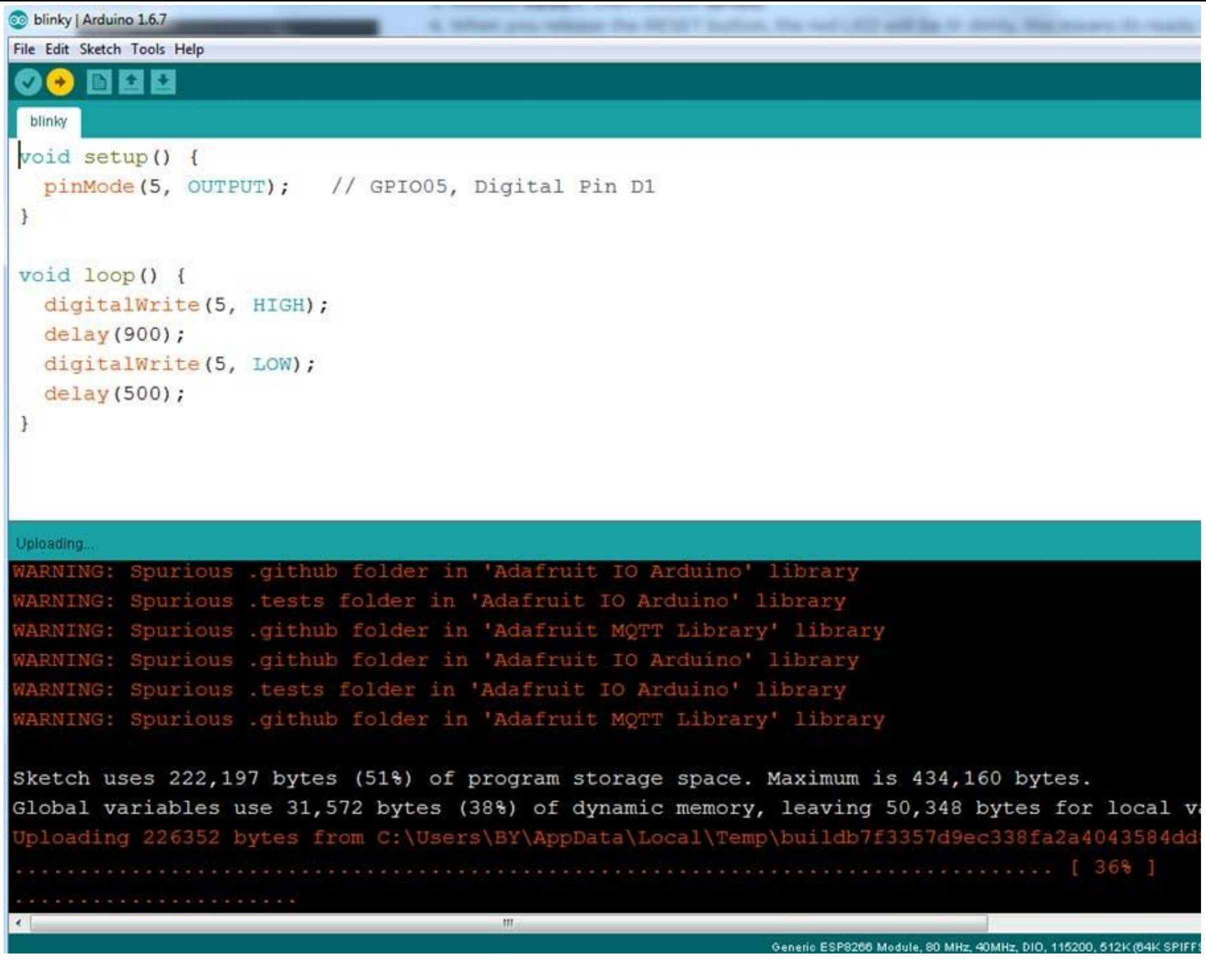


Figure3-1: Connection diagram for the blinking test



```
void setup() {
  pinMode(5, OUTPUT);    // GPIO05, Digital Pin D1
}

void loop() {
  digitalWrite(5, HIGH);
  delay(900);
  digitalWrite(5, LOW);
  delay(500);
}
```

```
WARNING: Spurious .github folder in 'Adafruit IO Arduino' library
WARNING: Spurious .tests folder in 'Adafruit IO Arduino' library
WARNING: Spurious .github folder in 'Adafruit MQTT Library' library
WARNING: Spurious .github folder in 'Adafruit IO Arduino' library
WARNING: Spurious .tests folder in 'Adafruit IO Arduino' library
WARNING: Spurious .github folder in 'Adafruit MQTT Library' library

Sketch uses 222,197 bytes (51%) of program storage space. Maximum is 434,160 bytes.
Global variables use 31,572 bytes (38%) of dynamic memory, leaving 50,348 bytes for local v
Uploading 226352 bytes from C:\Users\BY\AppData\Local\Temp\buildb7f3357d9ec338fa2a4043584dd
................................................................................ [ 36% ]
......................
```

Generic ESP8266 Module, 80 MHz, 40MHz, DIO, 115200, 512K (64K SPIFF

Figure 3.2: Uploading the sketch to ESP8266 NodeMCU module.

The sketch will start immediately - you'll see the LED blinking. Hooray!

Connecting via WiFi

OK once you've got the LED blinking, let's go straight to the fun part, connecting to a webserver. Create a new sketch with this code:

Don't forget to update:

const char* ssid = "yourssid";

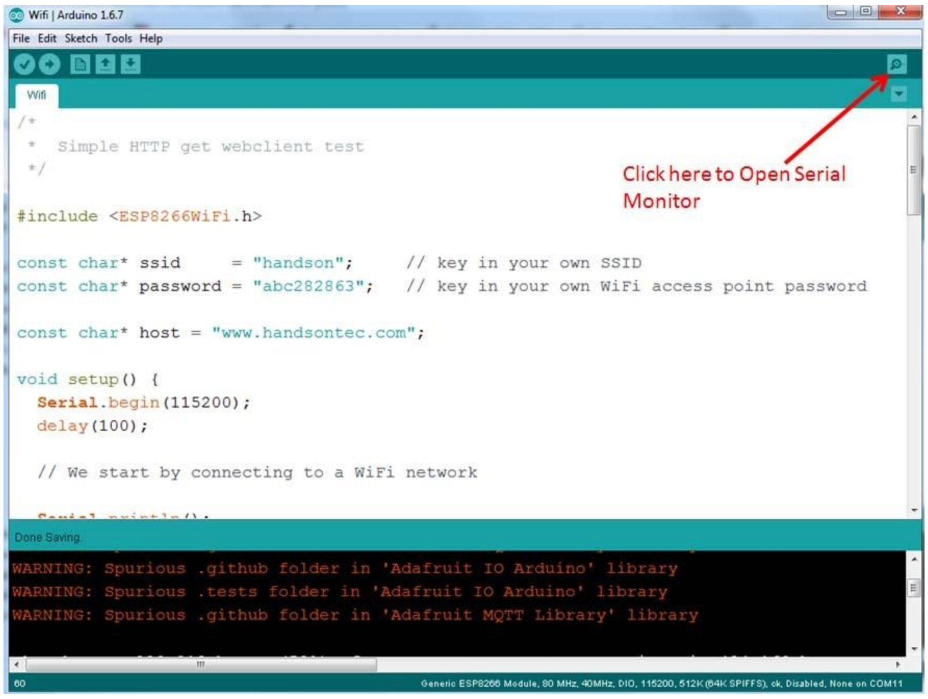
const char* password = "yourpassword";

to your WiFi access point and password, then upload the same way: get into bootload mode, then upload code via IDE.

```
/*
 *  Simple HTTP get webclient test
 */

#include <ESP8266WiFi.h>

const char* ssid     = "handson";      // key in your own SSID
const char* password = "abc1234";      // key in your own WiFi access point
password
```

```
const char* host = "www.handsontec.com";

void setup() {
  Serial.begin(115200);
  delay(100);

  // We start by connecting to a WiFi network

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

  WiFi.begin(ssid, password);

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

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

int value = 0;

void loop() {
  delay(5000);
  ++value;

  Serial.print("connecting to ");
  Serial.println(host);

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }

  // We now create a URI for the request
  String url = "/projects/index.html";
  Serial.print("Requesting URL: ");
  Serial.println(url);

  // This will send the request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  delay(500);

  // Read all the lines of the reply from server and print them to Serial
  while(client.available()){
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }

  Serial.println();
  Serial.println("closing connection");
}
```

Open up the IDE serial console at 115200 baud to see the connection and webpage printout!

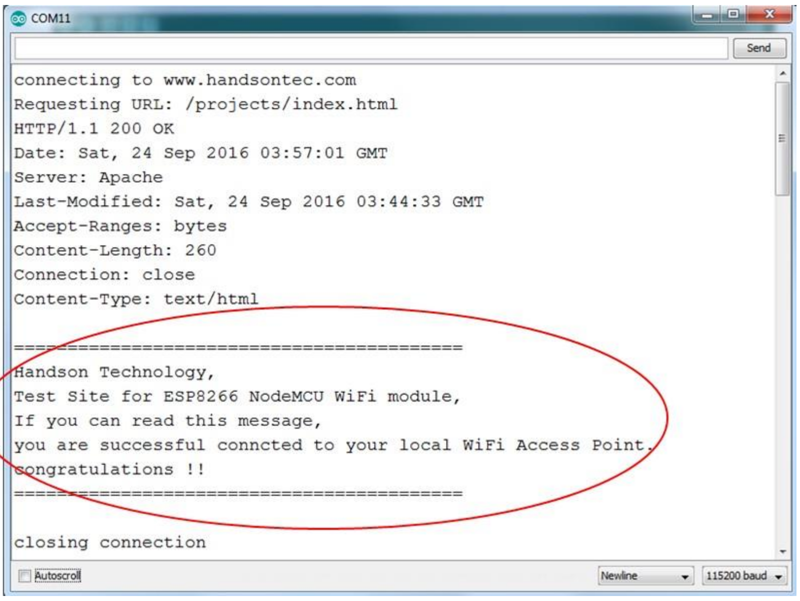

Wifi | Arduino 1.6.7
File Edit Sketch Tools Help
Wifi
Click here to Open Serial Monitor

```
/*
 *  Simple HTTP get webclient test
 */

#include <ESP8266WiFi.h>

const char* ssid     = "handson";      // key in your own SSID
const char* password = "abc282863";    // key in your own WiFi access point password

const char* host = "www.handsontec.com";

void setup() {
  Serial.begin(115200);
  delay(100);

  // We start by connecting to a WiFi network
```

Done Saving.

```
WARNING: Spurious .github folder in 'Adafruit IO Arduino' library
WARNING: Spurious .tests folder in 'Adafruit IO Arduino' library
WARNING: Spurious .github folder in 'Adafruit MQTT Library' library
```

60 Generic ESP8266 Module, 80 MHz, 40MHz, DIO, 115200, 512K (64K SPIFFS), ck, Disabled, None on COM11



```
connecting to www.handsontec.com
Requesting URL: /projects/index.html
HTTP/1.1 200 OK
Date: Sat, 24 Sep 2016 03:57:01 GMT
Server: Apache
Last-Modified: Sat, 24 Sep 2016 03:44:33 GMT
Accept-Ranges: bytes
Content-Length: 260
Connection: close
Content-Type: text/html

==========================================
Handson Technology,
Test Site for ESP8266 NodeMCU WiFi module,
If you can read this message,
you are successful conncted to your local WiFi Access Point.
congratulations !!
==========================================

closing connection
```

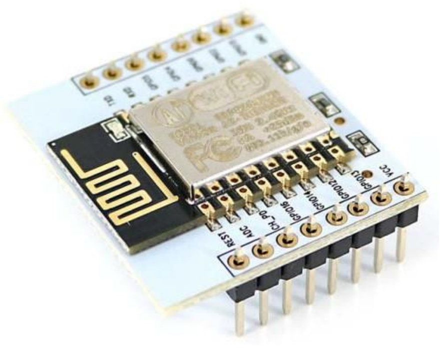
That's it, pretty easy right ! This section is just to get you started and test out your module.

4. Flashing NodeMCU Firmware on the ESP8266 using Windows

Why flashing your ESP8266 module with NodeMCU?

NodeMCU is a firmware that allows you to program the ESP8266 modules with LUA script. And you'll 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.

In this tutorial we are going to use another ESP8266 module with pin header adapter board which is breadboard friendly.

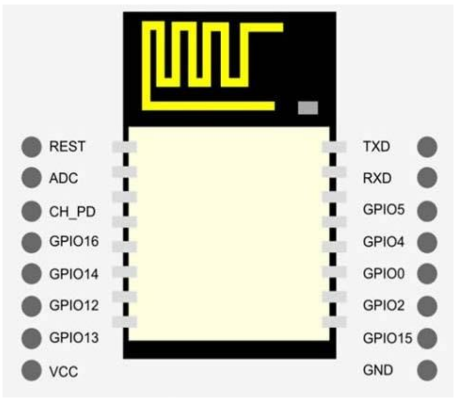


ESP8266 Module Breadboard Friendly with Header Connector

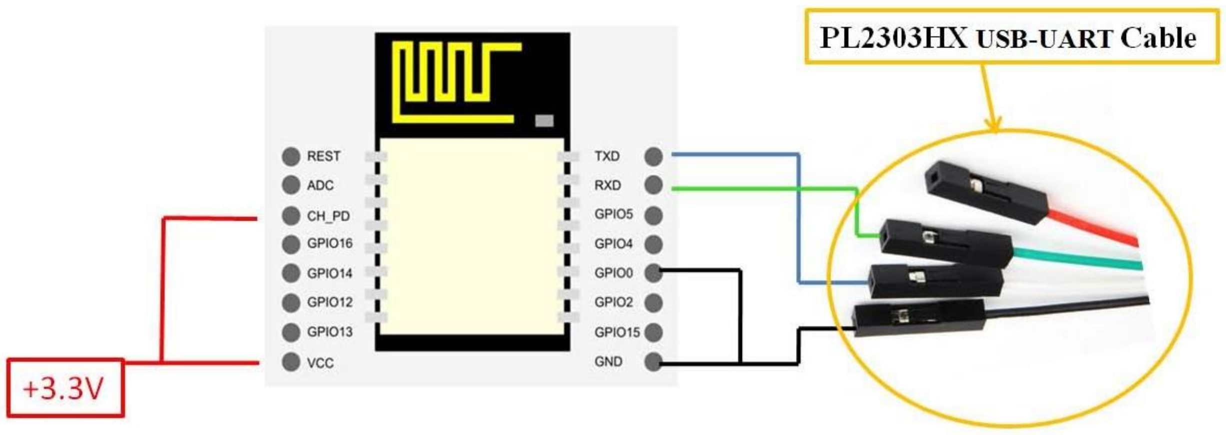
Parts Required:

- ESP8266 Module Breadboard Friendly
- PL2303HX USB-UART Converter Cable
- Some Male-to-Female JumperWires

Pin Assignment:



Wiring:



ESP8266 Pin	Description
CH_PD	Pull high, connect to Vcc +3.3V
Vcc	Power Supply +3.3V
TXD	Connect to RXD (white) of PL2303HX USB-Serial converter cable
RXD	Connect to TXD (Green) of PL2303HX USB-Serial converter cable
GPIO0	Pull low, connect to GND pin
GND	Power Supply ground

Downloading NodeMCU Flasher for Windows

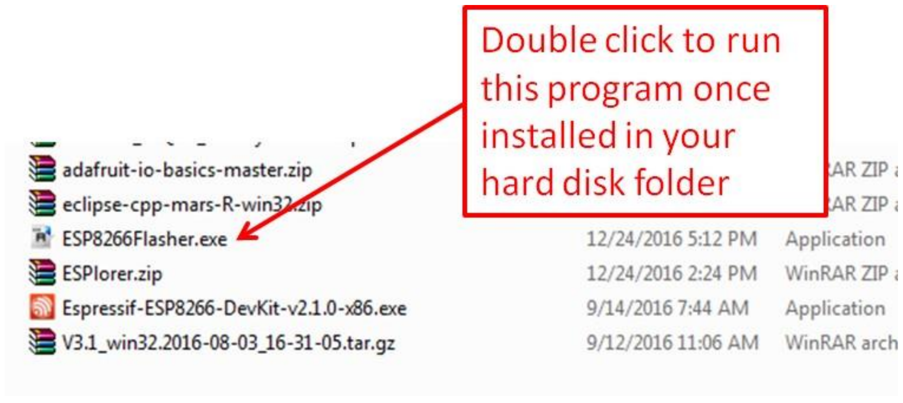
After wiring your circuit, you have to download the NodeMCU flasher. This is a .exe file that you can download using one of the following links:

- Win32 Windows Flasher
- Win64 Windows Flasher

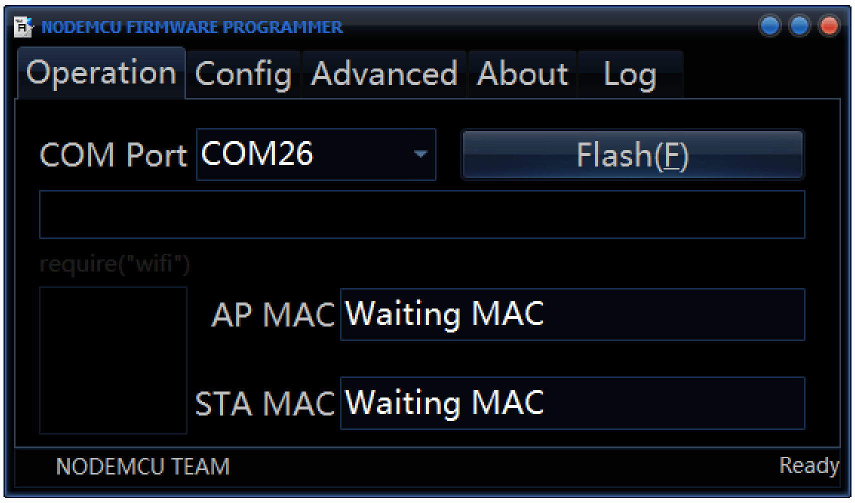
You can find all the information about NodeMCU flasher here.

Flashing your ESP8266 using Windows

Open the flasher that you just downloaded and a window should appear (as shown in the following figure).


Double click to run this program once installed in your hard disk folder

adafruit-io-basics-master.zip | | WinRAR ZIP
eclipse-cpp-mars-R-win32.zip | | WinRAR ZIP
ESP8266Flasher.exe | 12/24/2016 5:12 PM | Application
ESPlorer.zip | 12/24/2016 2:24 PM | WinRAR ZIP
Espressif-ESP8266-DevKit-v2.1.0-x86.exe | 9/14/2016 7:44 AM | Application
V3.1_win32.2016-08-03_16-31-05.tar.gz | 9/12/2016 11:06 AM | WinRAR arch



COM Port COM26

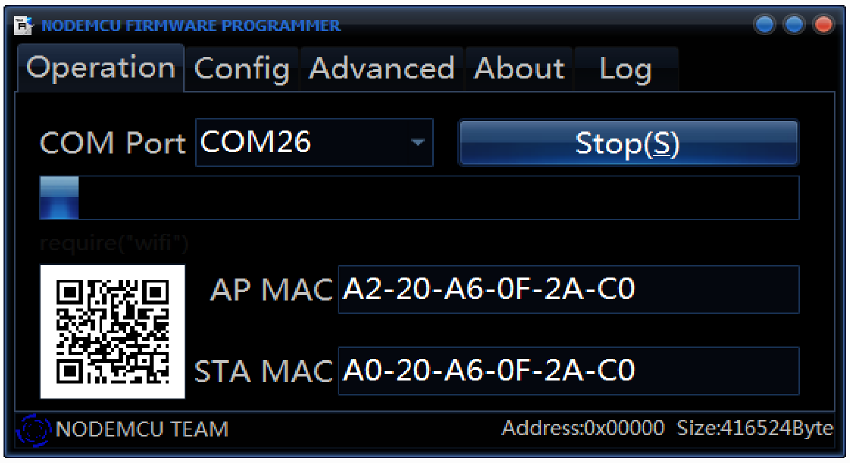
require("wifi")

AP MAC Waiting MAC

STA MAC Waiting MAC

NODEMCU TEAM Ready

Press the button "Flash" and it should start the flashing process immediately, showing the Module MAC address if successful connected.



COM Port COM26

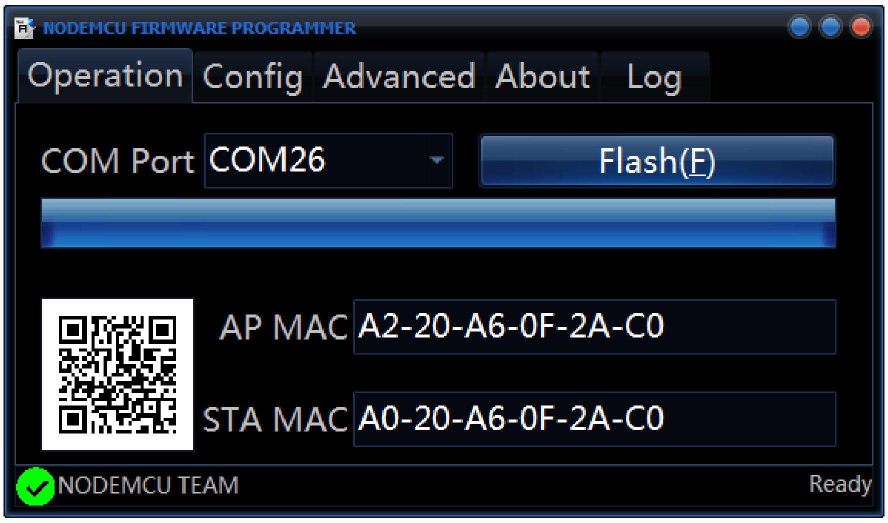
require("wifi")

AP MAC A2-20-A6-0F-2A-C0

STA MAC A0-20-A6-0F-2A-C0

NODEMCU TEAM Address:0x00000 Size:416524Byte

After finishing this flashing process, it should appear a green circle with a check icon at lower left corner.



COM Port COM26

AP MAC A2-20-A6-0F-2A-C0

STA MAC A0-20-A6-0F-2A-C0

NODEMCU TEAM Ready

Your ESP8266 module is now loaded with NodeMCU firmware.

5. Getting Started with the ESPlorer IDE

ESPlorer is an IDE (Integrated Development Environment) for ESP8266 devices. It's a multi platform IDE, can be used in any OS environment, this simply means that it runs on Windows, Mac OS X or Linux.

Supported platforms:

- Windows(x86, x86-64)
- Linux(x86, x86-64, ARM soft & hard float)
- Solaris(x86, x86-64)
- Mac OS X(x86, x86-64, PPC, PPC64)

This software allows you to establish a serial communications with your ESP8266 module, send commands, and 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. It requires JAVA (SE version 7 and above) installed.
- In order to complete the sample project presented in this Guide you need to flash your ESP8266 with NodeMCU firmware. Refer to chapter-4 in this guide on how to flash the NodeMCU firmware.

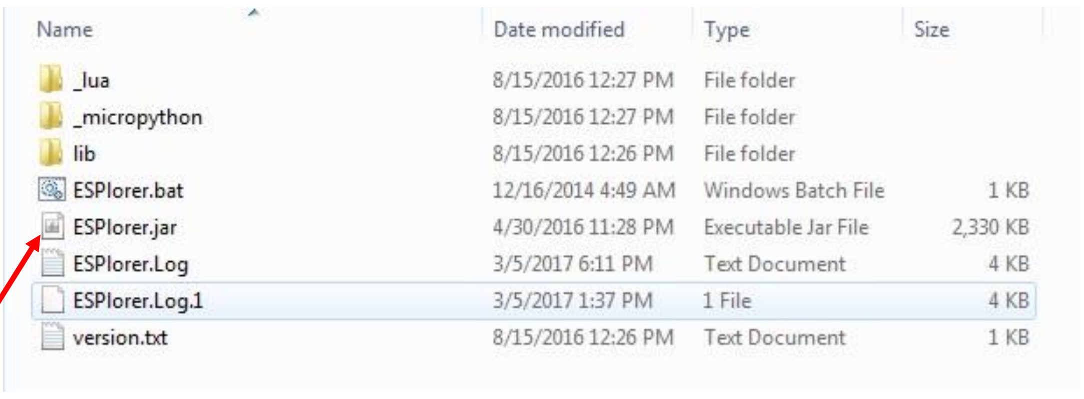
Main Resources:

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

Installing ESPlorer

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

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:

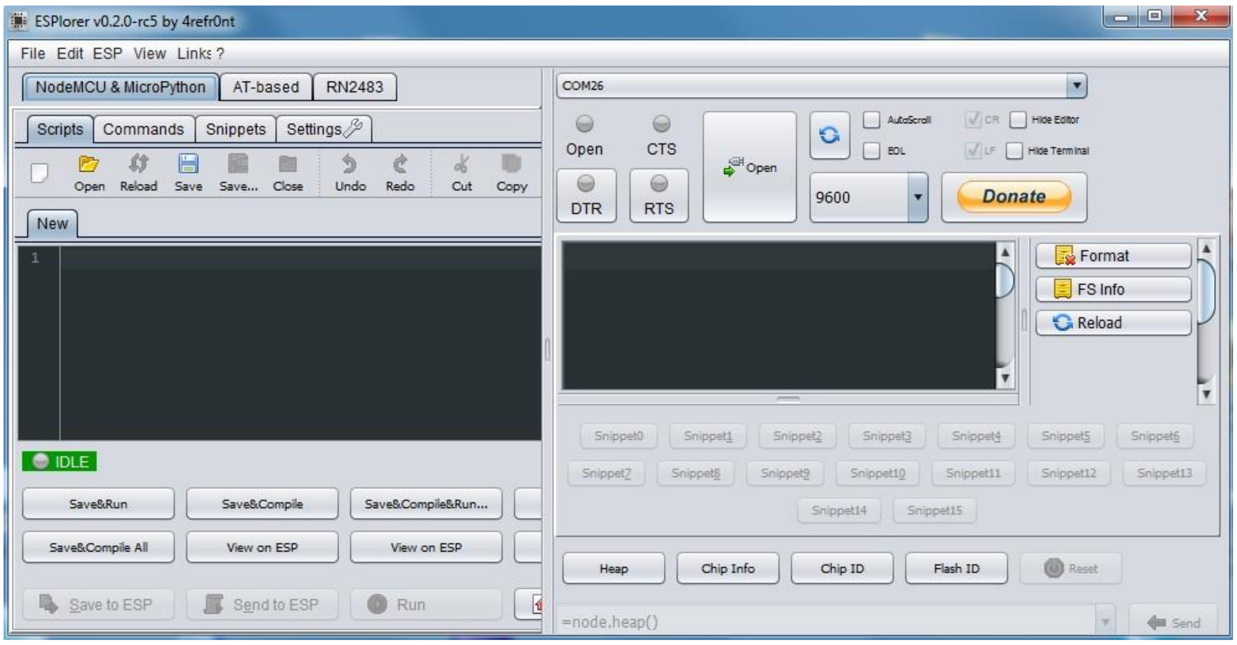


Name	Date modified	Type	Size
_lua	8/15/2016 12:27 PM	File folder	
_micropython	8/15/2016 12:27 PM	File folder	
lib	8/15/2016 12:26 PM	File folder	
ESPlorer.bat	12/16/2014 4:49 AM	Windows Batch File	1 KB
ESPlorer.jar	4/30/2016 11:28 PM	Executable Jar File	2,330 KB
ESPlorer.Log	3/5/2017 6:11 PM	Text Document	4 KB
ESPlorer.Log.1	3/5/2017 1:37 PM	1 File	4 KB
version.txt	8/15/2016 12:26 PM	Text Document	1 KB

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 you simply use this command line in your terminal to run the ESPlorer: sudo java –jar ESPlorer.jar.

When the ESPlorer first opens, that's what you should see:

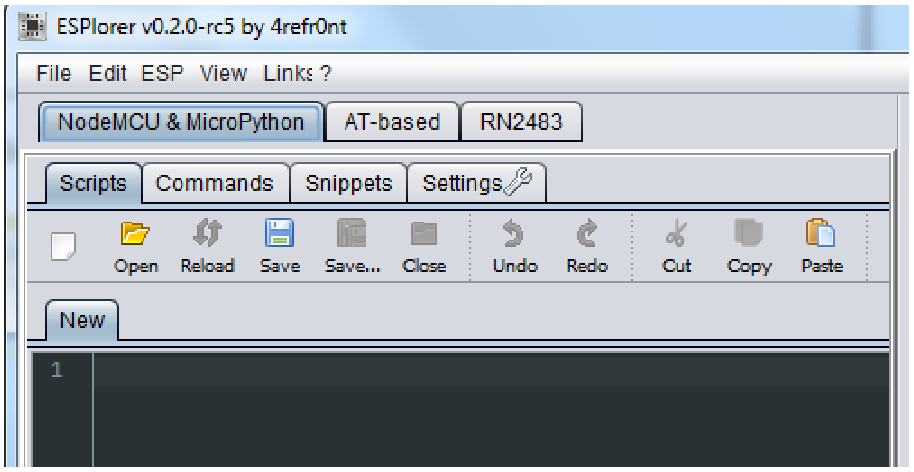


Here's a rundown of the features the ESPlorer IDE includes:

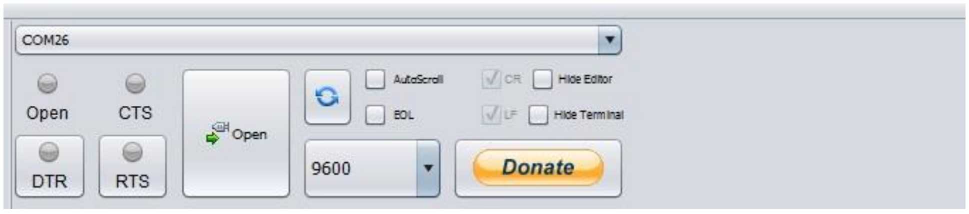
- Syntax highlighting LUA and Python code.
- Code editor color themes: default, dark, Eclipse, IDEA, Visual Studio.
- Undo/Redo editors features.
- Code Autocomplete (Ctrl+Space).
- Smart send data to ESP8266 (without dumb send with fixed line delay), check correct answer from ESP8266 after every lines.
- Code snippets.
- Detailed logging.
- And a lot more…

The ESPlorer IDE has a couple of main sections, let's break it down each one.

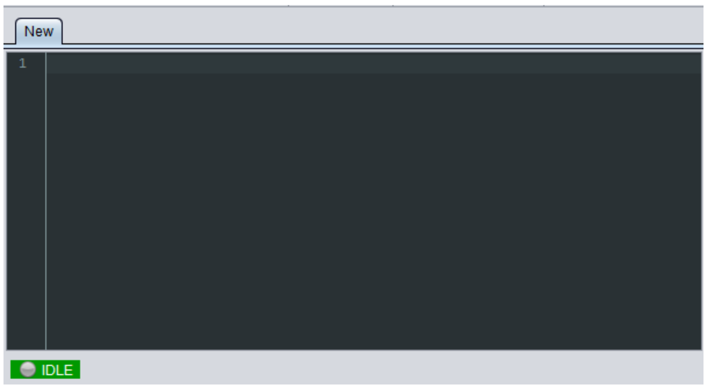
In the top left corner you can see all the regular options that you find in any software. Create a New file, Open a new file, Save file, Save file as, Undo, Redo, etc.



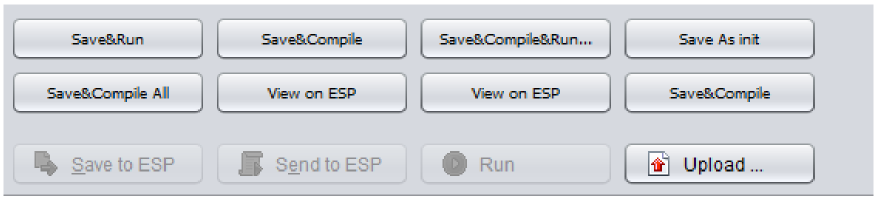
In the top right corner you have all the options you need to establish a serial communication (you're going to learn how to use them later in this Guide).



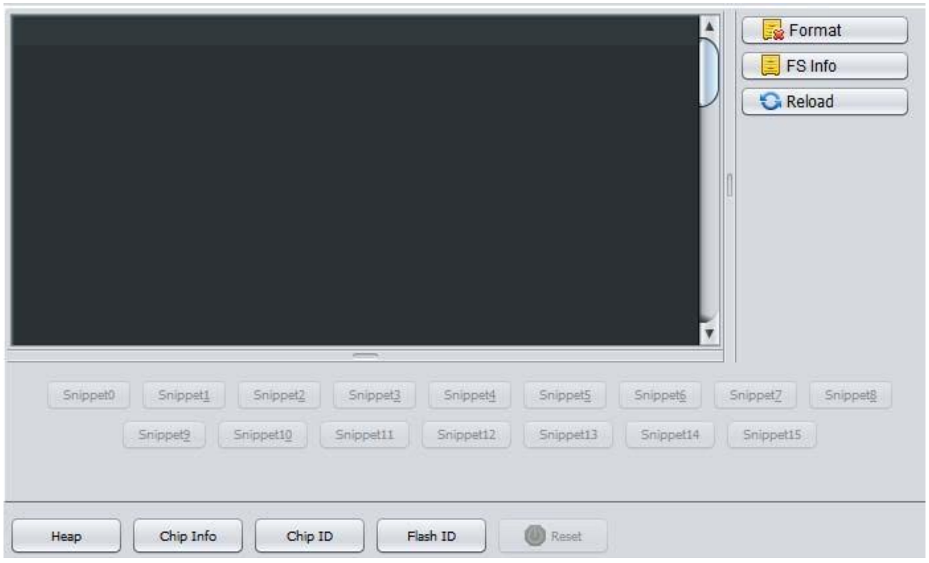
This next screenshot shows your Code Window, that's where you write your scripts (your scripts are highlighted with your code syntax).



Below the Code Window, you have 12 buttons that offer you all the functions you could possible need to interact with your ESP8266. Here's the ones you'll use most: "Save to ESP" and "Send to ESP".

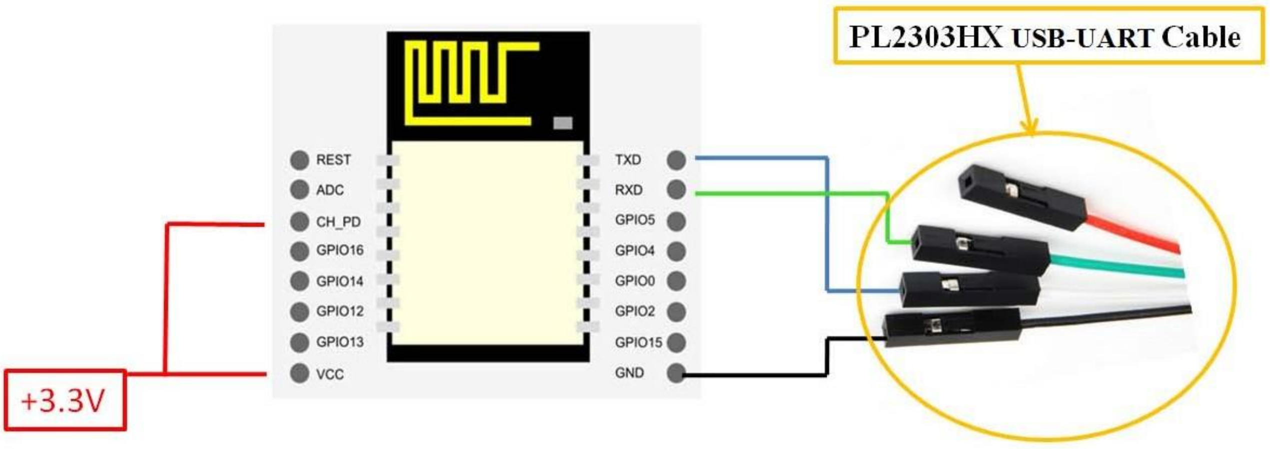


This screenshot shows the Output Window which tells you exactly what's going on in your ESP8266. You can see errors and use prints in your code to debug your projects.



Schematics

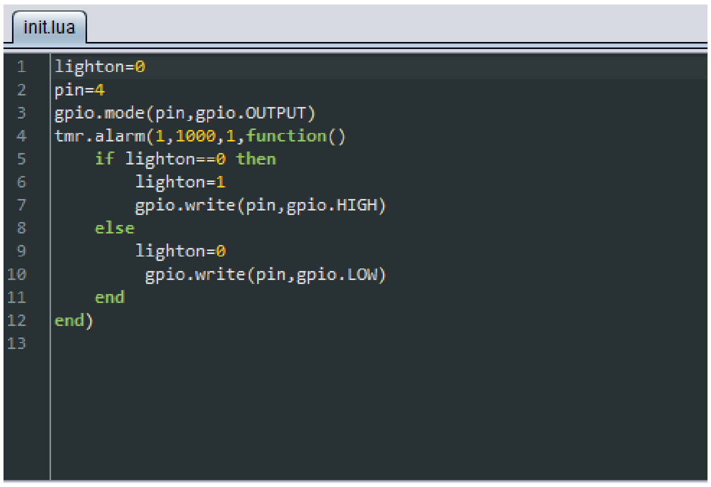
To upload code to your ESP8266, you should connect your ESP8266 to your PL2303HX USB-UART Programming Cable like the figure below:



Writing Your Lua Script

Below is your script to blink an LED.

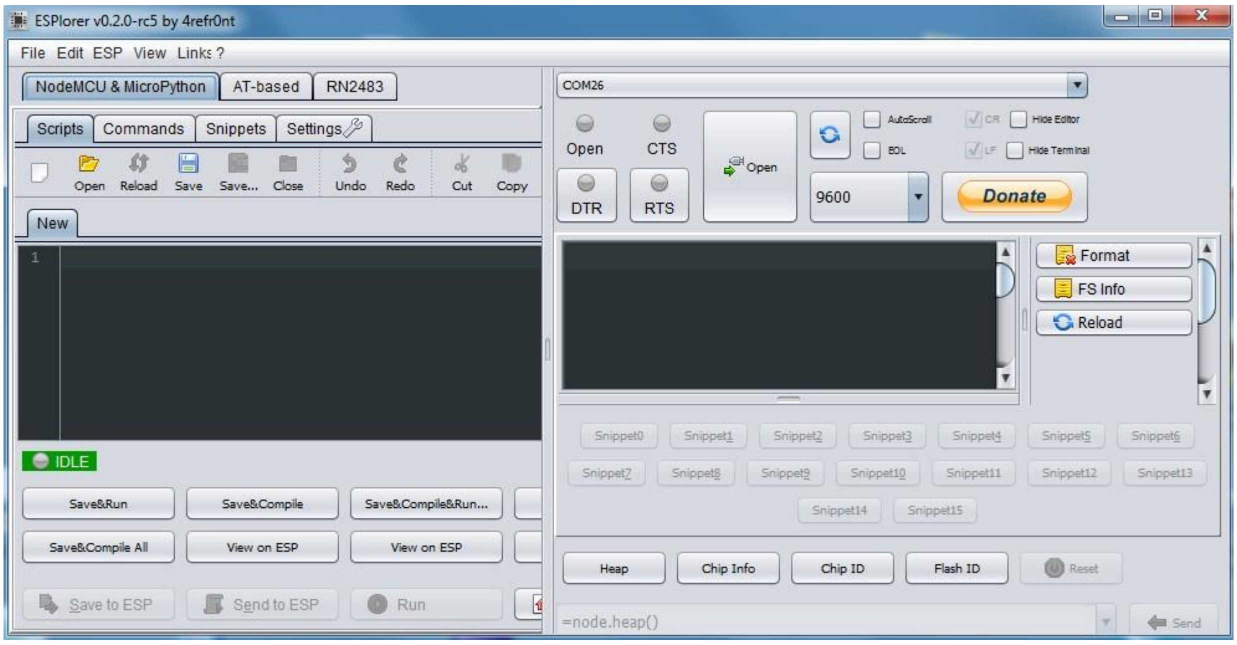
```
lighton=0
pin=4
gpio.mode(pin,gpio.OUTPUT)
tmr.alarm(1,2000,1,function()
    if lighton==0 then
        lighton=1
        gpio.write(pin,gpio.HIGH)
    else
        lighton=0
         gpio.write(pin,gpio.LOW)
    end
end)
```



```
lighton=0
pin=4
gpio.mode(pin,gpio.OUTPUT)
tmr.alarm(1,1000,1,function()
    if lighton==0 then
        lighton=1
        gpio.write(pin,gpio.HIGH)
    else
        lighton=0
         gpio.write(pin,gpio.LOW)
    end
end)
```

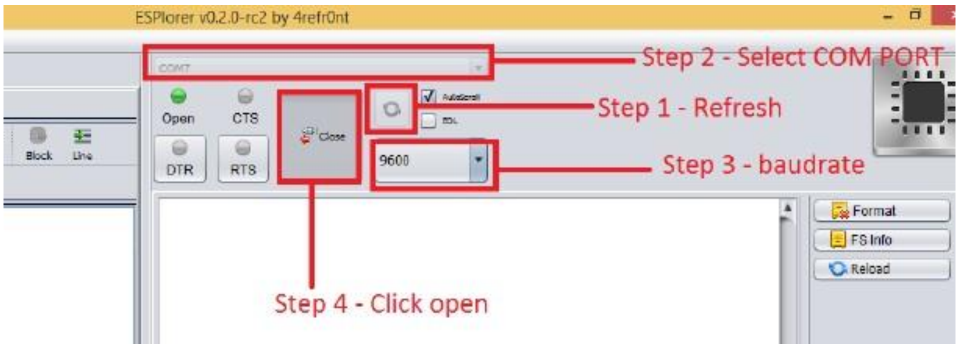
Right now you don’t need to worry how this code works, but how you can upload it to your ESP8266.

Having your ESP8266+PL2303HX Programmer connected to your computer, go to the ESPlorer IDE:



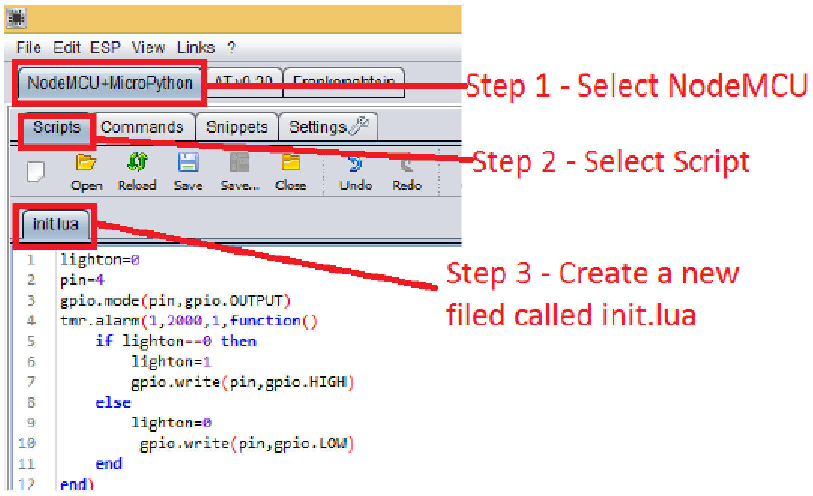
Look at the top right corner of your ESPlorer IDE and follow these instructions:

1. Press the Refreshbutton.
2. Select the COM port for your FTDI programmer.
3. Select your baudrate.
4. Click Open.



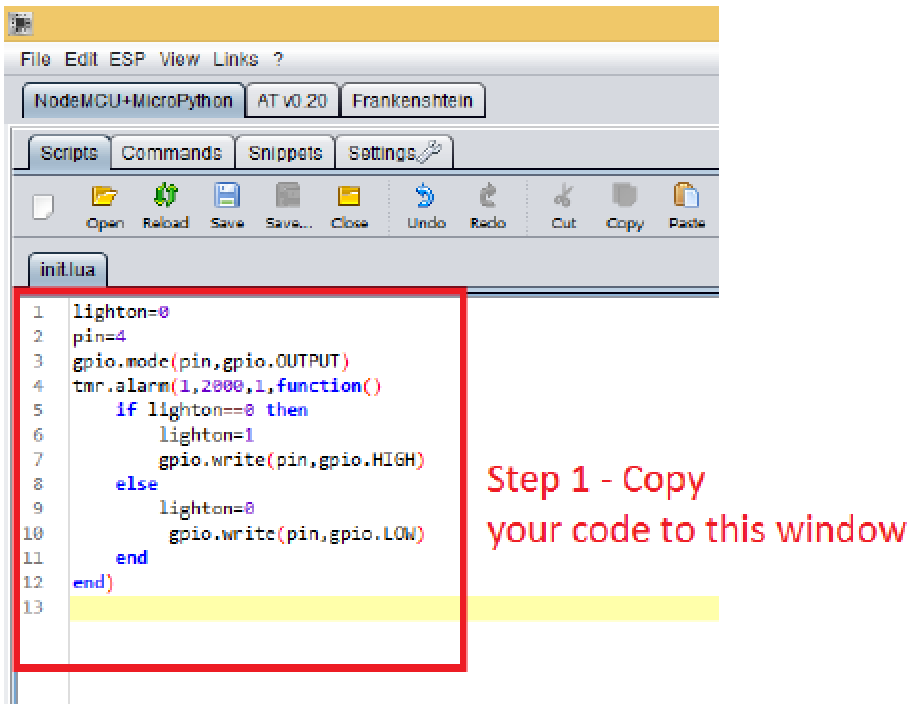
Then in the top left corner of your ESPlorer IDE, follow these instructions:

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



```
lighton=0
pin=4
gpio.mode(pin,gpio.OUTPUT)
tmr.alarm(1,2000,1,function()
    if lighton==0 then
        lighton=1
        gpio.write(pin,gpio.HIGH)
    else
        lighton=0
         gpio.write(pin,gpio.LOW)
    end
end)
```

Copy your Lua script to the code window (as you can see in the Figure below):

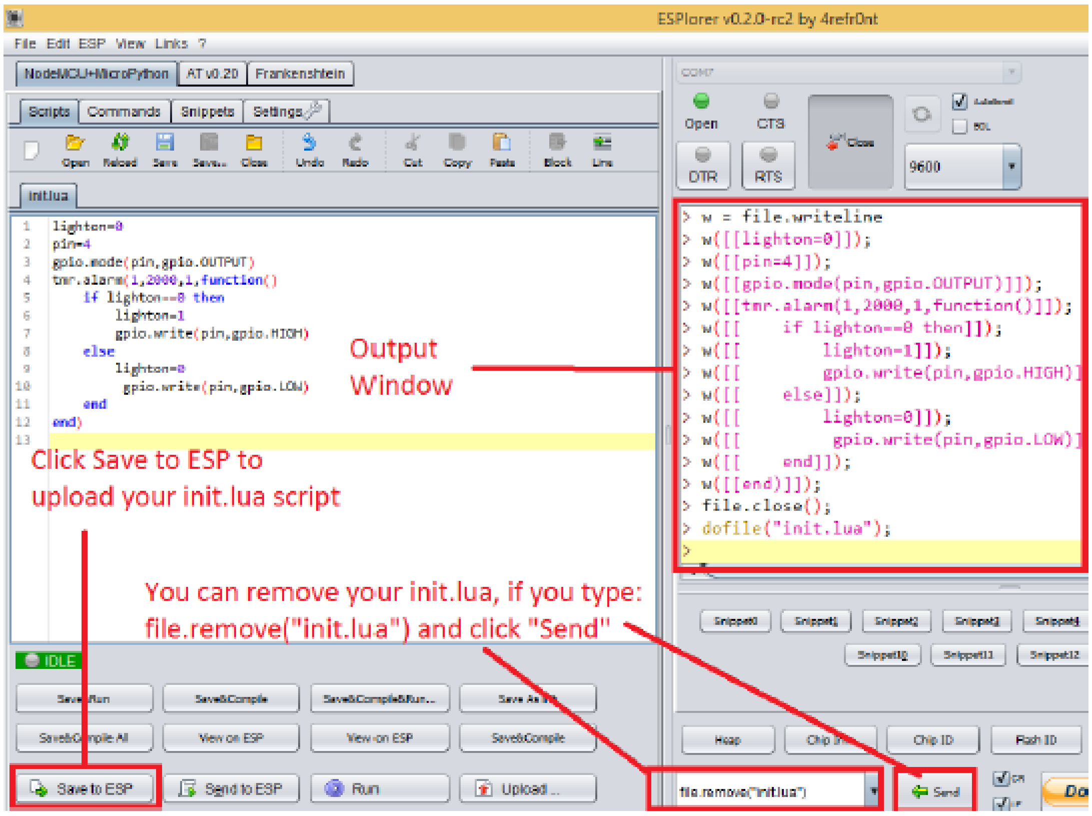


```
lighton=0
pin=4
gpio.mode(pin,gpio.OUTPUT)
tmr.alarm(1,2000,1,function()
    if lighton==0 then
        lighton=1
        gpio.write(pin,gpio.HIGH)
    else
        lighton=0
         gpio.write(pin,gpio.LOW)
    end
end)
```

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.



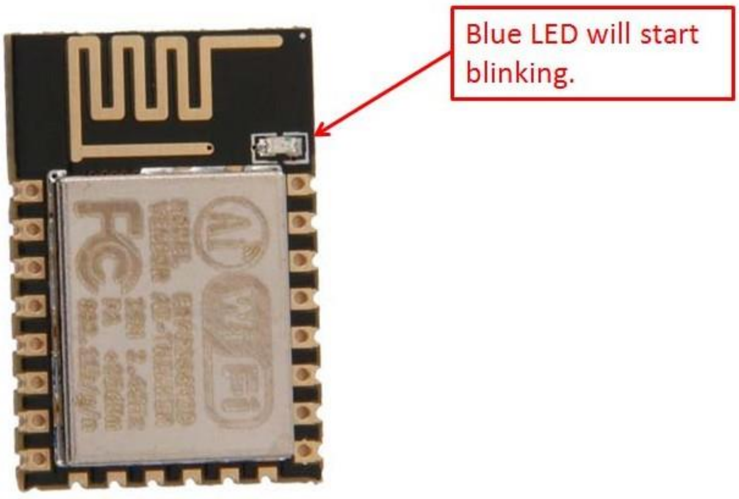
```
lighton=0
pin=4
gpio.mode(pin,gpio.OUTPUT)
tmr.alarm(1,2000,1,function()
    if lighton==0 then
        lighton=1
        gpio.write(pin,gpio.HIGH)
    else
        lighton=0
         gpio.write(pin,gpio.LOW)
    end
end)
```

```
> w = file.writeline
> w([[lighton=0]]);
> w([[pin=4]]);
> w([[gpio.mode(pin,gpio.OUTPUT)]]);
> w([[tmr.alarm(1,2000,1,function()]]);
> w([[    if lighton==0 then]]);
> w([[        lighton=1]]);
> w([[        gpio.write(pin,gpio.HIGH)]
> w([[    else]]);
> w([[        lighton=0]]);
> w([[         gpio.write(pin,gpio.LOW)]
> w([[    end]]);
> w([[end)]]);
> file.close();
> dofile("init.lua");
>
```

ESPlorer v0.2.0-rc2 by 4refr0nt
File Edit ESP View Links ?
NodeMCU+MicroPython
AT v0.20
Frankenshtein
Scripts
Commands
Snippets
Settings
init.lua
Open
CTS
DTR
RTS
Close
9600
Output Window
Click Save to ESP to upload your init.lua script
You can remove your init.lua, if you type: file.remove("init.lua") and click "Send"
IDLE
Save to ESP
Send to ESP
Run
Upload ...
file.remove("init.lua")
Send

Note: If you want to delete your "init.lua" file, you can do that easily. 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. You can type any commands and send them to your ESP8266 through that window.

After uploading your code to your ESP8266, unplug your ESP8266 from your computer and power up the ESP8288 module.



Congratulations, you've made it! The blue LED at the upper right corner should be blinking every 2 seconds!

6. NodeMCU GPIO for Lua

The GPIO(General Purpose Input/Output) allows us to access to pins of ESP8266 , all the pins of ESP8266 accessed using the command GPIO, all the access is based on the I/O index number on the NoddMCU dev kits, not the internal GPIO pin, for example, the pin 'D7' on the NodeMCU dev kit is mapped to the internal GPIO pin 13, if you want to turn 'High' or 'Low' that particular pin you need to called the pin number '7', not the internal GPIO of the pin. When you are programming with generic ESP8266 this confusion will arise which pin needs to be called during programming, if you are using NodeMCU devkit, it has come prepared for working with Lua interpreter which can easily program by looking the pin names associated on the Lua board. If you are using generic ESP8266 device or any other vendor boards please refer to the table below to know which IO index is associated to the internal GPIO of ESP8266.

Nodemcu dev kit	**ESP8266 Pin**	**Nodemcu dev kit**	**ESP8266 Pin**
D0	GPIO16	D7	GPIO13
D1	GPIO5	D8	GPIO15
D2	GPIO4	D9	GPIO3
D3	GPIO0	D10	GPIO1
D4	GPIO2	D11	GPIO9
D5	GPIO14	D12	GPIO10
D6	GPIO12		

D0 or GPIO16 can be used only as a read and write pin, no other options like PWM/I2C are supported by this pin.

In our example in chapter 5 on blinking the blue LED, the blue LED in connected to GPIO2, it is defined as Pin4 (D4) in Lua script.

7. Web Resources:

- ESP8266 Lua Nodemcu WIFI Module
- ESP8266 Breadboard FriendlyModule
- ESP8266 Remote Serial WIFI Module
- PL2303HX USB-UART Converter Cable