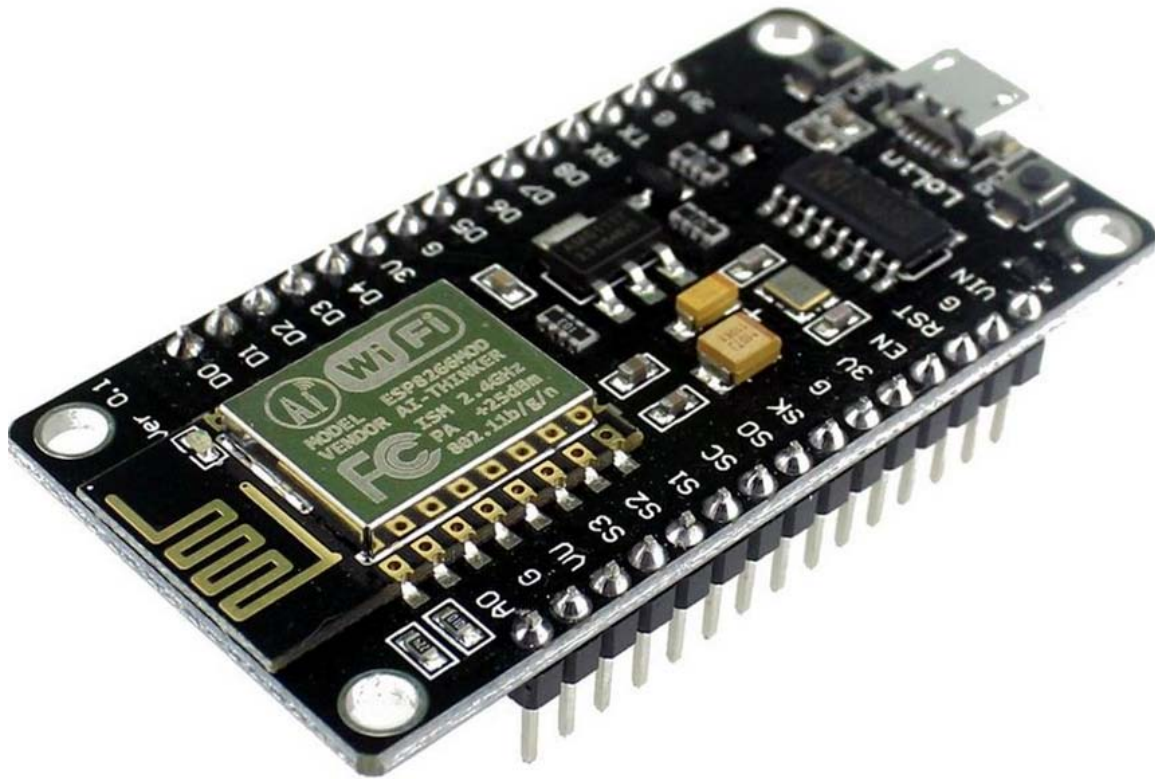


ESP8266 NodeMCU WiFi Devkit



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.

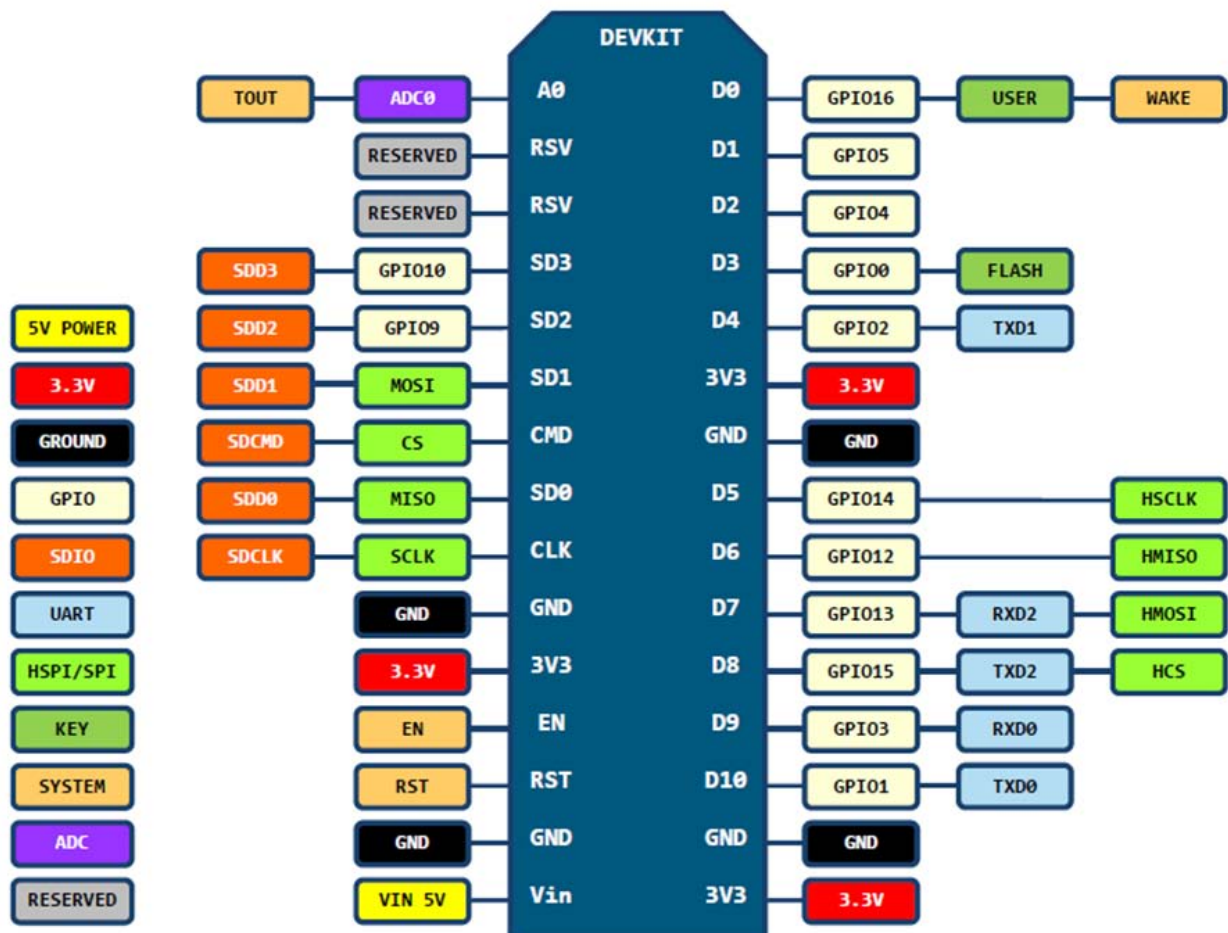
Table of Contents

1. Specification:.....	3
2. Pin Definition:	3
3. Using Arduino IDE	3
3.1 Install the Arduino IDE 1.6.4 or greater	4
3.2 Install the ESP8266 Board Package	4
3.3 Setup ESP8266 Support.....	5
3.4 Blink Test	7
3.5 Connecting via WiFi.....	9

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.

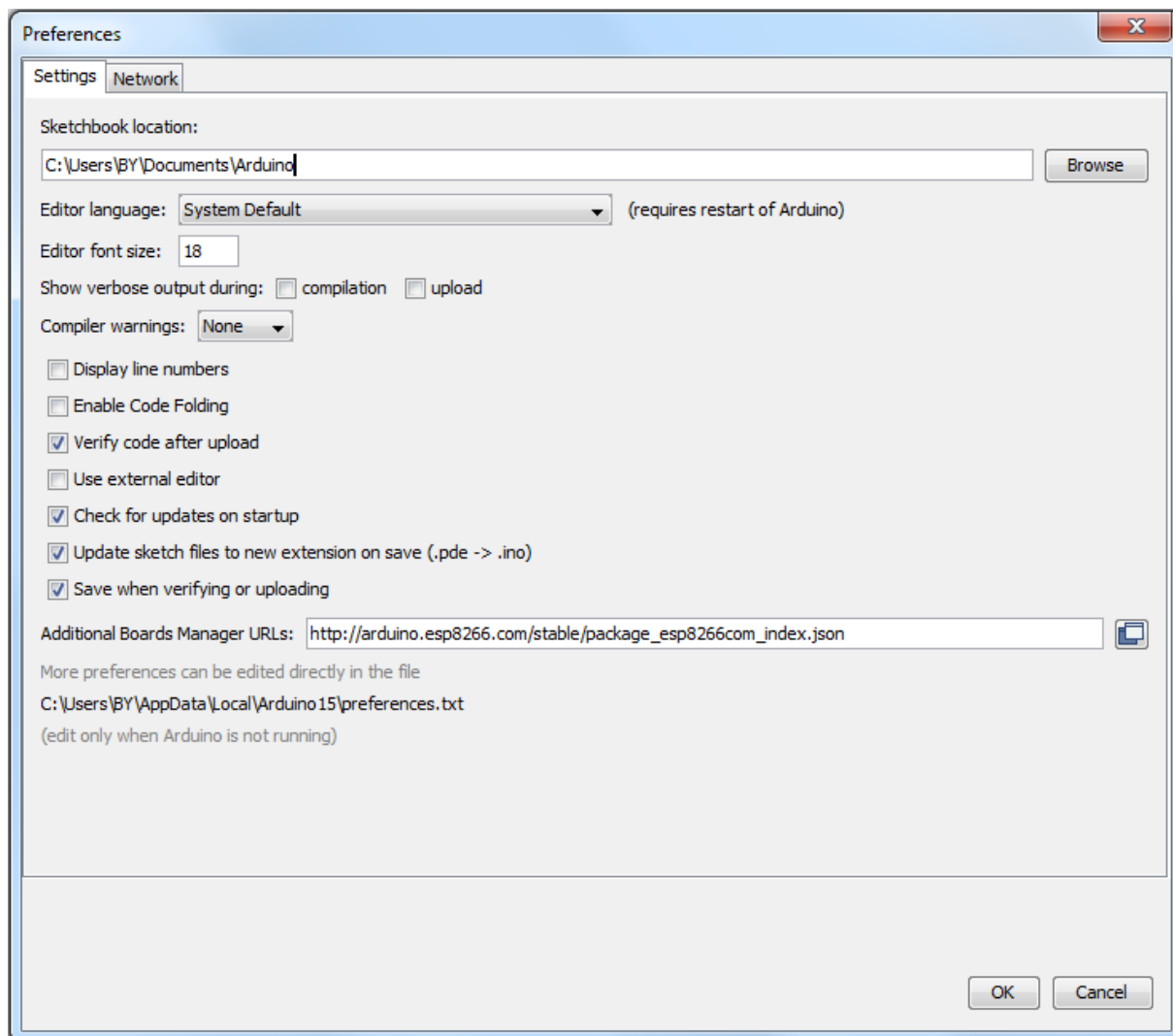
3.1 Install the Arduino IDE 1.6.4 or greater

[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.](#)

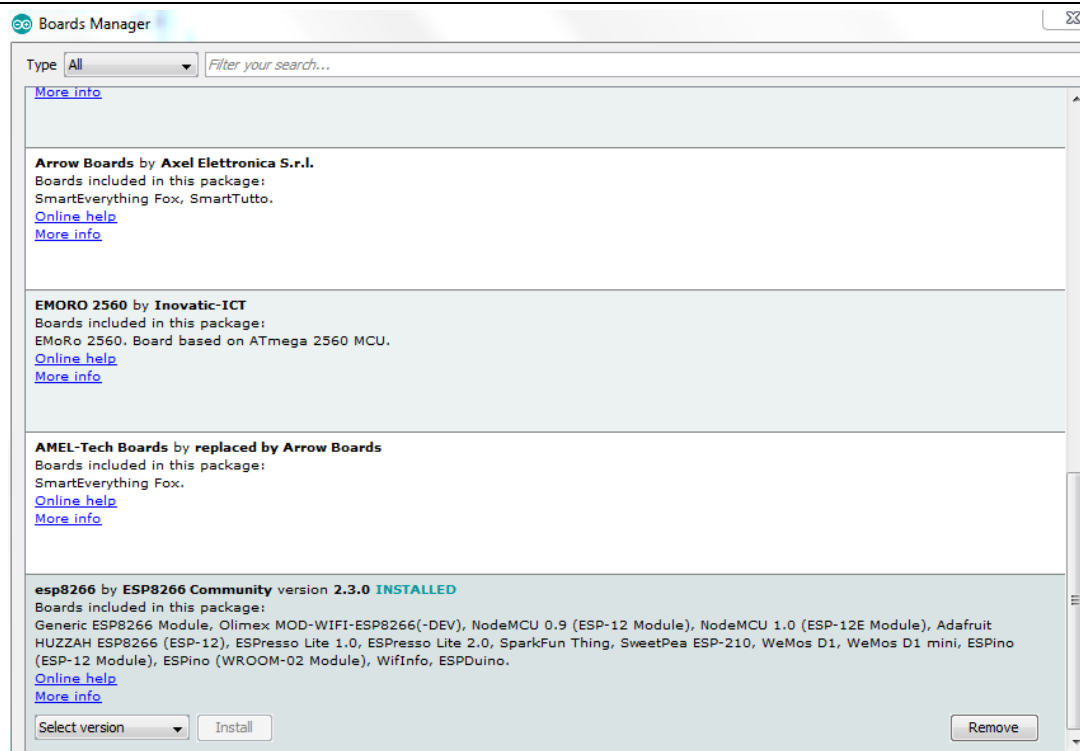
3.2 Install the ESP8266 Board Package

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



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

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

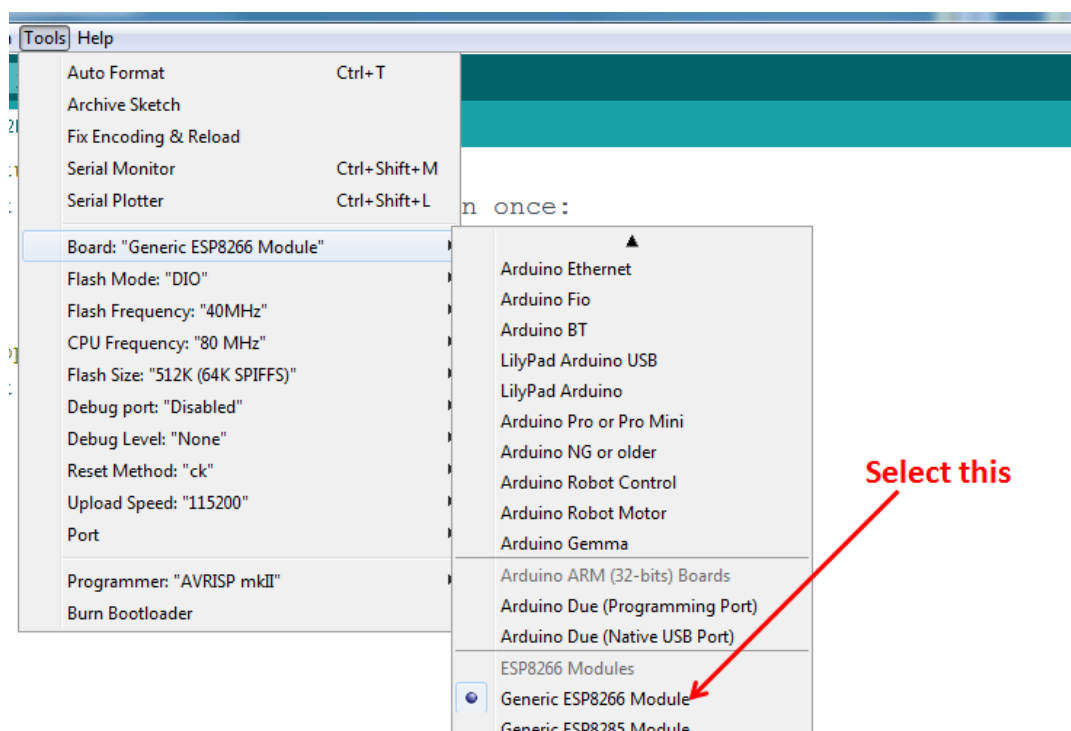


Click 'Tools' -> 'Board:' -> 'Board Manager...' to access this panel.

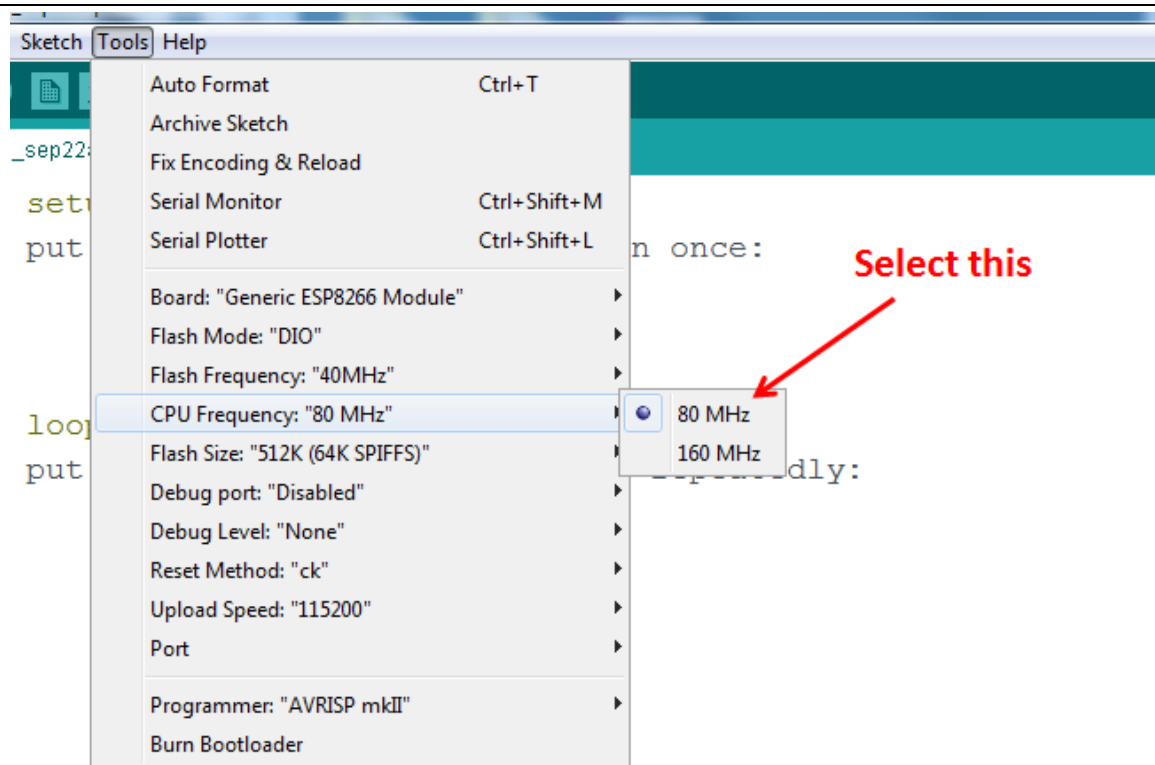
Scroll down to 'esp8266 by ESP8266 Community' and click "Install" button to install the ESP8266 library package. Once installation completed, close and re-open Arduino IDE for ESP8266 library to take effect.

3.3 Setup ESP8266 Support

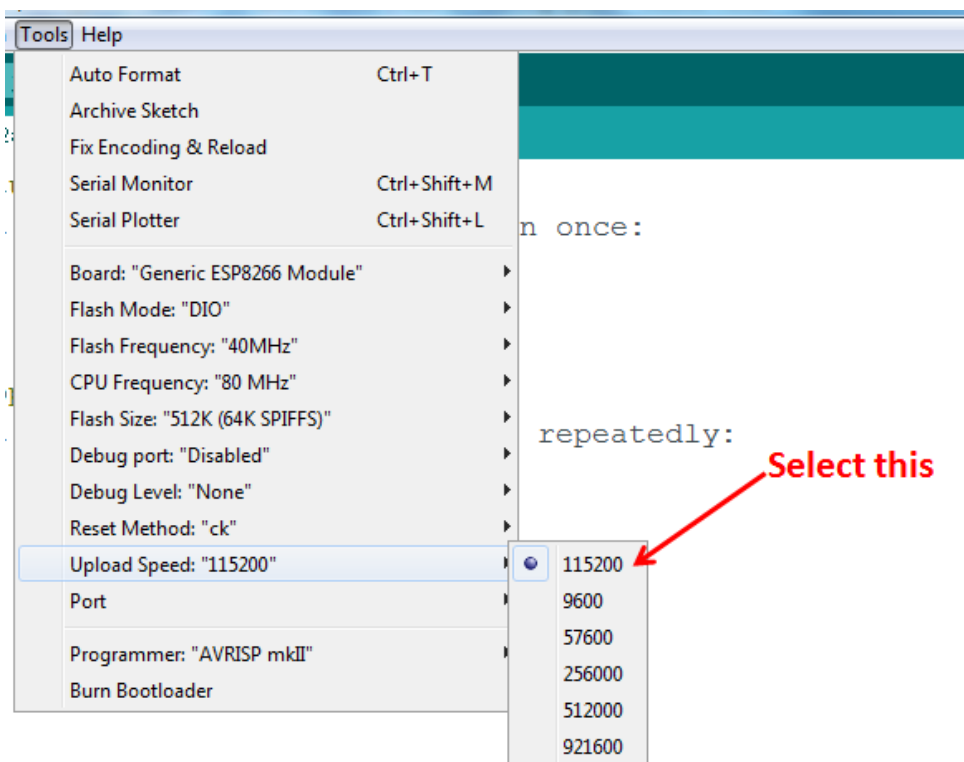
When you've restarted Arduino IDE, select 'Generic ESP8266 Module' from the 'Tools' -> 'Board:' dropdown menu.



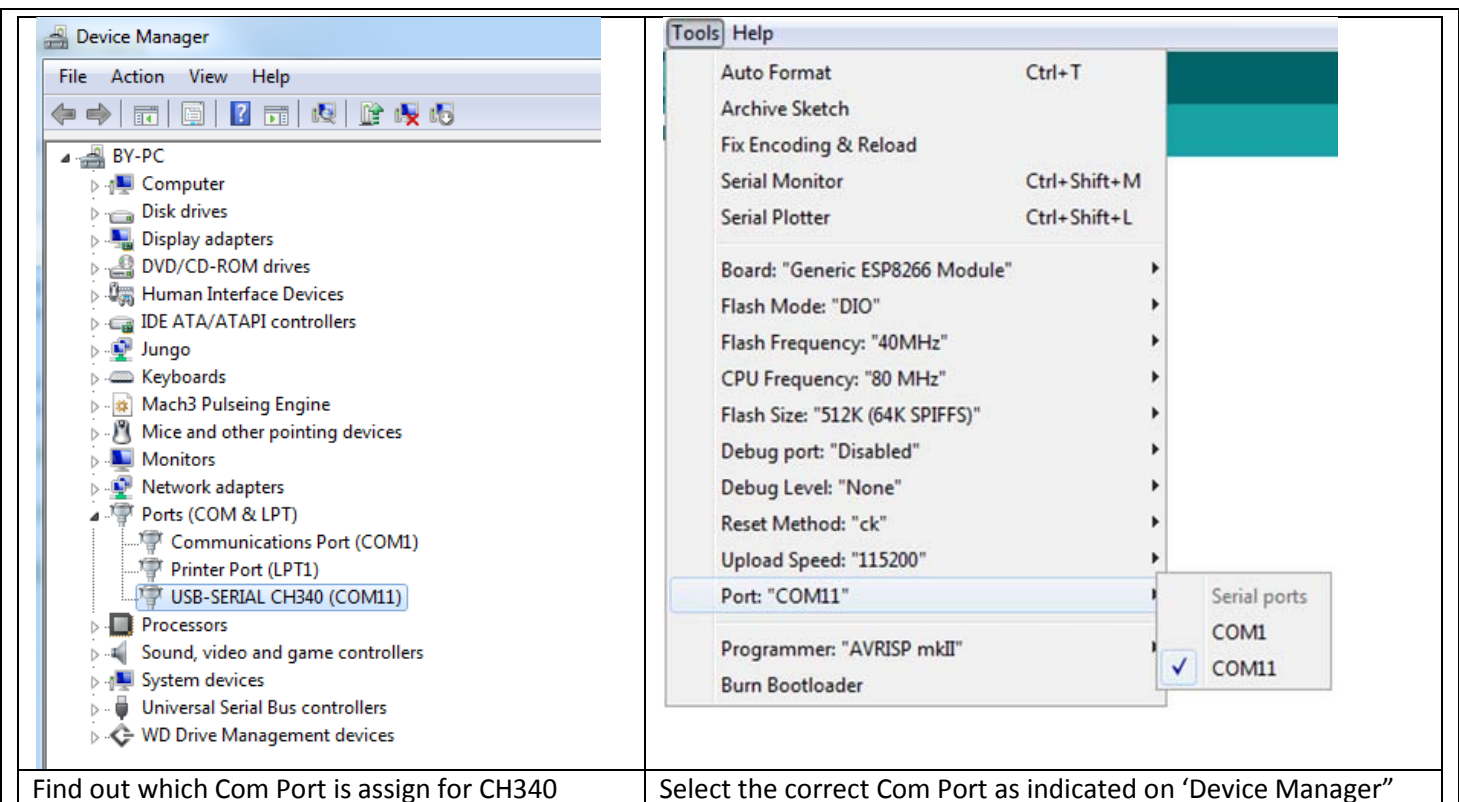
Select 80 MHz as the CPU frequency (you can try 160 MHz overclock later)



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.



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.



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>

3.4 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.

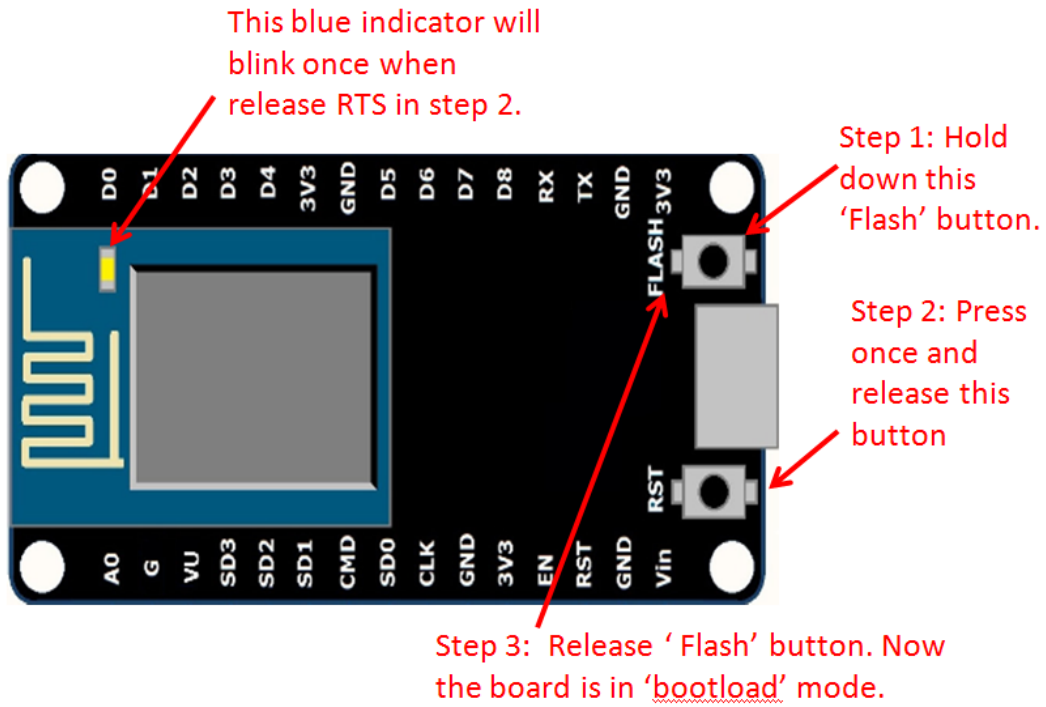
```
void setup() {
  pinMode(5, OUTPUT);    // GPIO05, Digital Pin D1
}

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

Now you'll need to put the board into bootloader mode. You'll have to do this before each upload. There is no timeout for bootloader 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 indicator will blink once, this means its ready to bootload.



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

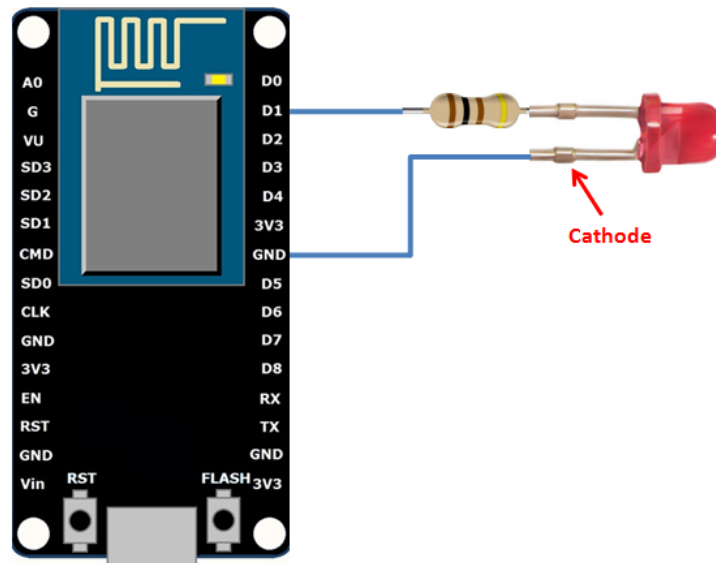


Figure3-1: Connection diagram for the blinking test

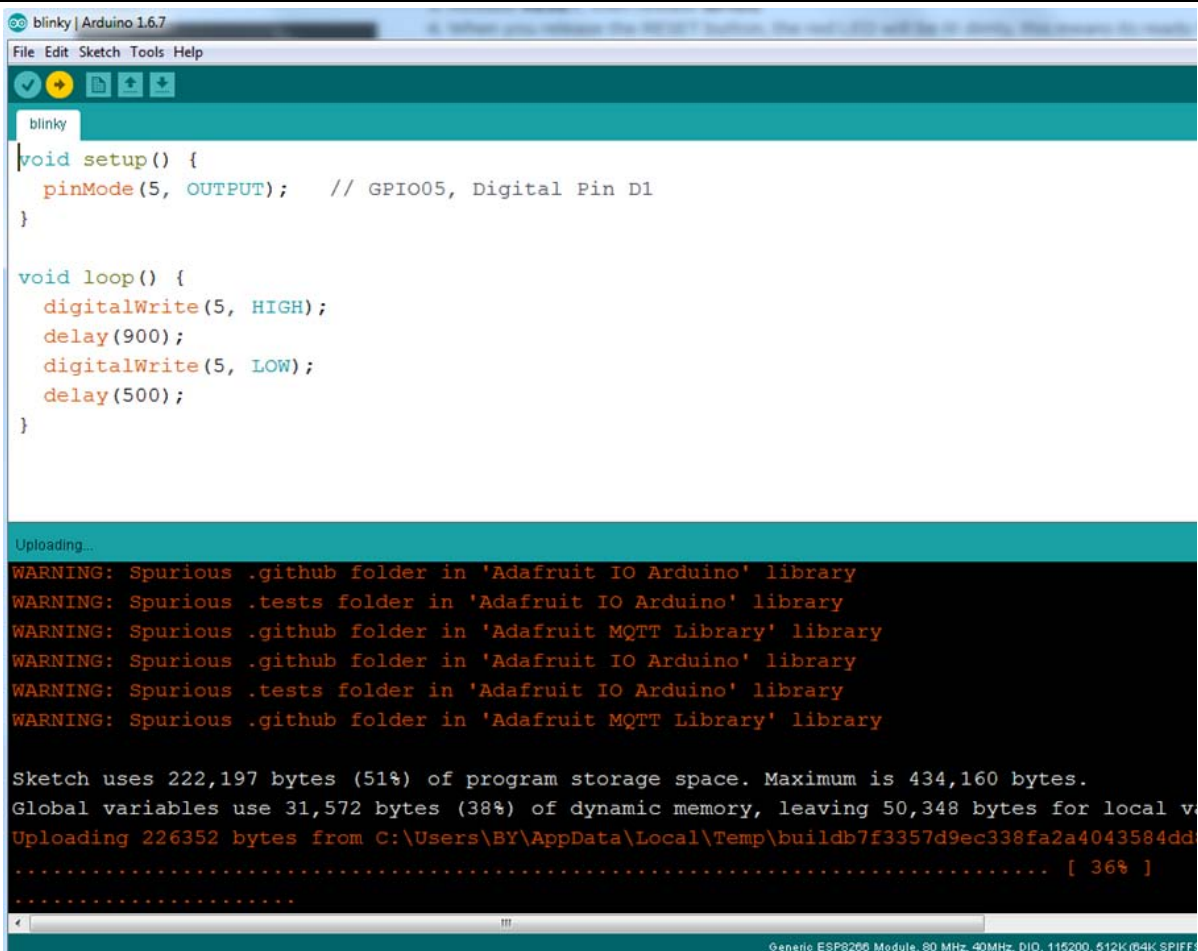


Figure 3.2: Uploading the sketch to ESP8266 NodeMCU module.

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

3.5 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);

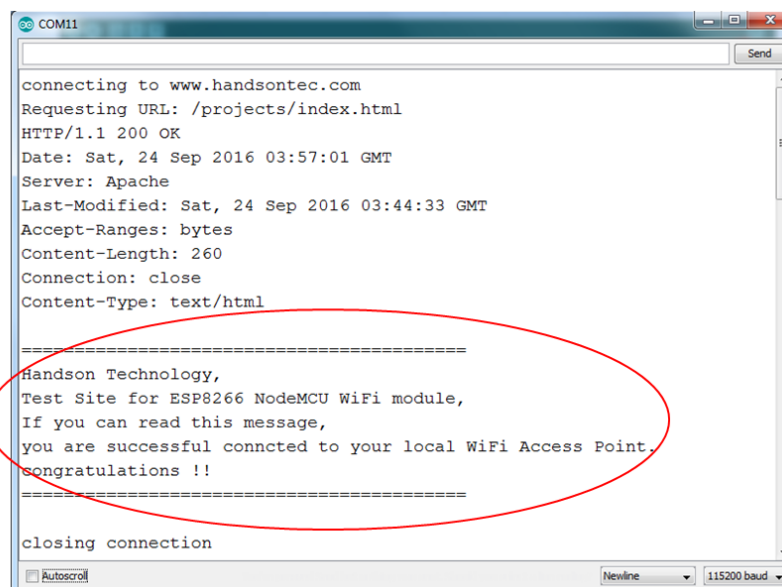
  // This will send the request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
    "Host: " + host + "\r\n" +
    "Connection: close\r\n\r\n");
  delay(500);

  // 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");
}

```

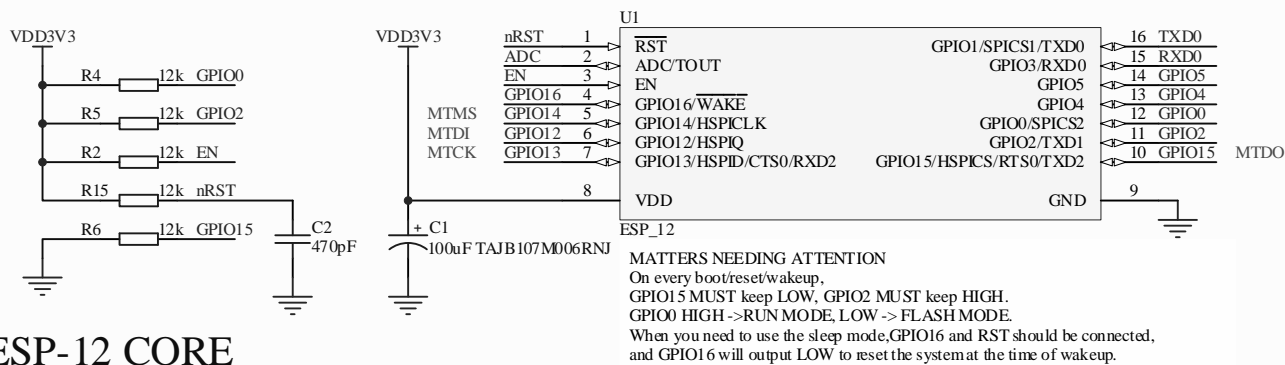
Open up the IDE serial console at 115200 baud to see the connection and webpage printout!



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

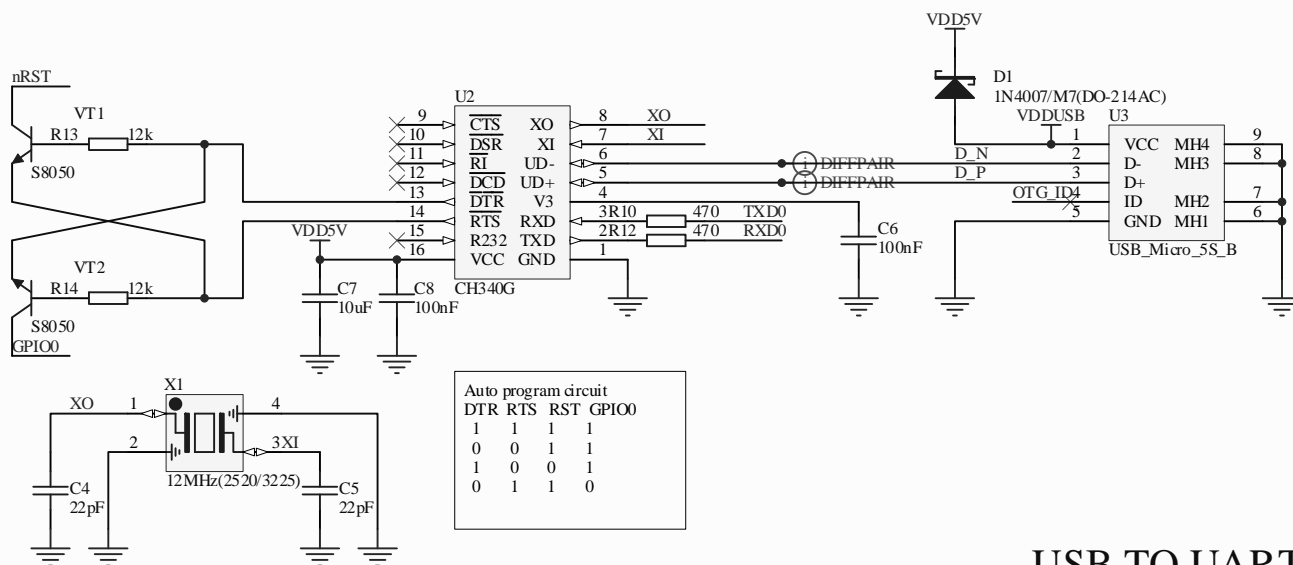
A

A



B

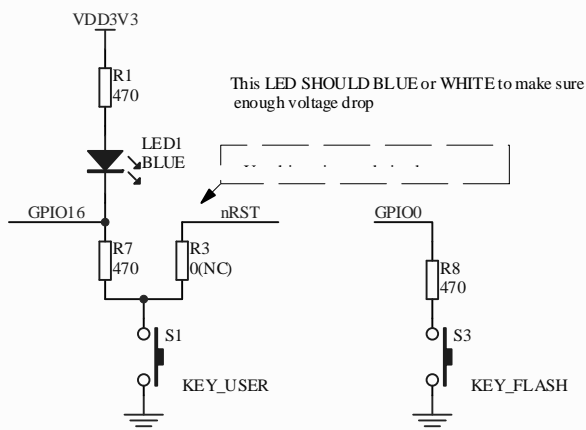
B



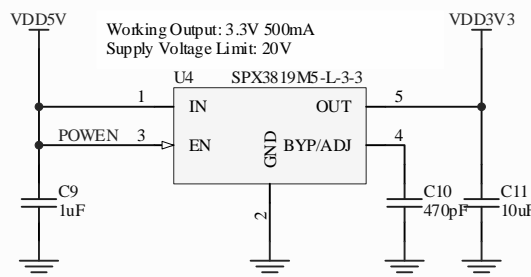
USB TO UART

C

C



KEY



POWER

D

D

