LAMPIRAN A

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id E-mail : info@polsri.ac.id

PELAKSANAAN REVISI LAPORAN AKHIR

Mahasiswa berikut,

Nama	:	Alifia Sava Salsabilla
NIM	:	062030321032
Jurusan/Program Studi	:	Teknik Elektro / DIII Teknik Elektronika
Judul Laporan Akhir	:	Rancang bangun alat monitoring pemberi makan dan minum kucing otomatis Menggunakan mikrokontroller NodeMCU berbasis *Internet of things (IOT).*

Telah melaksanakan revisi terhadap Laporan Akhir yang diujikan pada hari Selasa, 8 Agustus 2023 Pelaksanaan revisi terhadap Laporan Akhir tersebut telah disetujui oleh Dosen Penguji yang memberikan revisi:

No.	Komentar	Nama Dosen Penguji *)	Tanggal	Tanda Tangan
	Acc	Dr. Rd. Kusumanto, S.T.,M.M.	29/8/23	
	Acc	Ir. Iskandar Lutfi, M.T.	8/8/2023	
	Acc	Yeni Irdayanti, S.T.,M,Kom.	28/8-23	
	Sudah revisi	Niksen Alfarizal, S.T.,M.Kom.	1/9 23	
	Acc (Sudah Revisi)	Ibnu Maja, S.Si.,M.M.	28/8-2023	

Palembang, ...

Ketua Penguji **),

(Dr.Rd. Kusumanto, S.T.,M.M.)

NIP .....

Catatan:
*) Dosen penguji yang memberikan revisi saat ujian laporan akhir.
**) Dosen penguji yang ditugaskan sebagai Ketua Penguji saat ujian LA.
Lembaran pelaksanaan revisi ini harus dilampirkan dalam Laporan Akhir.

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id E-mail : info@polsri.ac.id

LEMBAR BIMBINGAN LAPORAN AKHIR

Nama : Alifia Sava Salsabilla
NIM : 062030321032
Jurusan/Program Studi : Teknik Elektro/ DIII Teknik Elektronika
Judul Laporan Akhir : .....
.....
.....
Pembimbing I : Ir.A. Rahman.,M.T

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
1.	3/2-2023	Konsultasi masalah judul Lap akhir dan " alat Rancang kancing ato matis	A. RAHMAN.
2.	22/05-203.	Pengesahan judul Lap akhir	
3.	20/06-2023	Konsultasi Bab. I dan Bab. III	
4.	22/07 '23	Diskusi Lap akhir Bab. I	
5.	24/07. '23	Konsultasi bab III .	
6.	28/07. '23	Diskusi data IV	
7.	31/07.'23	Meminjam Peralatan. Setiap bab.	

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
8.	1/08 - '23	Disetujui persiapan untuk Ujian Up akhir	
9.	3/08 - '23	Rekomendasi untuk ujian akhir	
10.			
11.			
12.			

Palembang, ...

Ketua Jurusan/KPS,

(.....)
NIP .....

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id E-mail : info@polsri.ac.id

LEMBAR BIMBINGAN LAPORAN AKHIR

Nama : Alifia Sava Salsabilla
NIM : 062030321032
Jurusan/Program Studi : Teknik Elektro/ DIII Teknik Elektronika
Judul Laporan Akhir : .....
.....
.....

Pembimbing II : Ibnu Maja, S.Si., M.M.

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
1.	3/01-2023	Konsultasi Masalah Judul Laporan akhir alat Rancang Pemberi Makan kucing otomatis"	
2.	22/05-2023.	Proposal LA.	
3.	25/05-2023.	Revisi Proposal LA.	
4.	22/06-2023.	Laporan Bab I - Bab III.	
5.	26/06-2023	Revisi Bab I	
6.	30/06-2023	Revisi Bab II	
7.	3/06-2023	Revisi Bab. III.	

No.	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
8.	28/-23/07	Revisi BAB IV. (Pembahasan).	
9.	1/-2023/08	Revisi Keseluruhan. (Bab 1-5).	
10.	3/-2023/08	Revisi Kesimpulan & Saran	
11.			
12.			

Palembang, ...

Ketua Jurusan/KPS,

(.....)

NIP .....

KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp 0711-353414 Fax 0711-355918
Website : www.polisriwijaya.ac.id E-mail : info@polsri.ac.id

KESEPAKATAN BIMBINGAN LAPORAN AKHIR (LA)

Kami yang bertanda tangan di bawah ini,

Pihak Pertama

Nama	: Alfia Sava Salsabilla
NIM	: 062030321032
Jurusan	: Teknik Elektro
Program Studi	: DIII Teknik Elektronika

Pihak Kedua

Nama	: Ir.A. Rahman.,M.T.
NIP	: 196202052993031002
Jurusan	: Teknik Elektro
Program Studi	: DIII Teknik Elektronika

Pada hari ini Selasa tanggal 29-08-2023 telah sepakat untuk melakukan konsultasi bimbingan Laporan Akhir.

Konsultasi bimbingan sekurang-kurangnya 1 (satu) kali dalam satu minggu. Pelaksanaan bimbingan pada setiap hari ..... pukul 10.00....., tempat di Politeknik Negeri Sriwijaya.

Demikianlah kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Laporan Akhir.

Palembang, 29-08-2023

Pihak Pertama,

(Alfia Sava Salsabilla)
NIM. 062030321032

Pihak Kedua,

(Ir.A. Rahman.,M.T.)
NIP. 196202052993031002

Mengetahui,
Ketua Jurusan

(Ir. Iskandar Lutfi, M.T)
NIP. 196501291991031002

KEMENTERIAN RISET,TEKNOLOGI DAN PENDIDIKAN TINGGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp 0711-353414 Fax 0711-355918
Website : www.polisriwijaya.ac.id E-mail : info@polsri.ac.id

KESEPAKATAN BIMBINGAN LAPORAN AKHIR (LA)

Kami yang bertanda tangan di bawah ini,

Pihak Pertama

Nama	: Alifia Sava Salsabilla
NIM	: 062030321032
Jurusan	: Teknik Elektro
Program Studi	: DIII Teknik Elektronika

Pihak Kedua

Nama	: Ibnu Maja, S.Si.,M.M.
NIP	: 197604052005011002
Jurusan	: Teknik Elektro
Program Studi	: DIII Teknik Elektronika

Pada hari ini ..Selasa..... tanggal ..29-08-2023..... telah sepakat untuk melakukan konsultasi bimbingan Laporan Akhir.

Konsultasi bimbingan sekurang-kurangnya 1 (satu) kali dalam satu minggu. Pelaksanaan bimbingan pada setiap hari ..Senin & Rabu..... pukul ..10:00....., tempat di Politeknik Negeri Sriwijaya.

Demikianlah kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Laporan Akhir.

Palembang, 29-08-2023

Pihak Pertama,

(Alifia Sava Salsabilla)
NIM. 062030321032

Pihak Kedua,

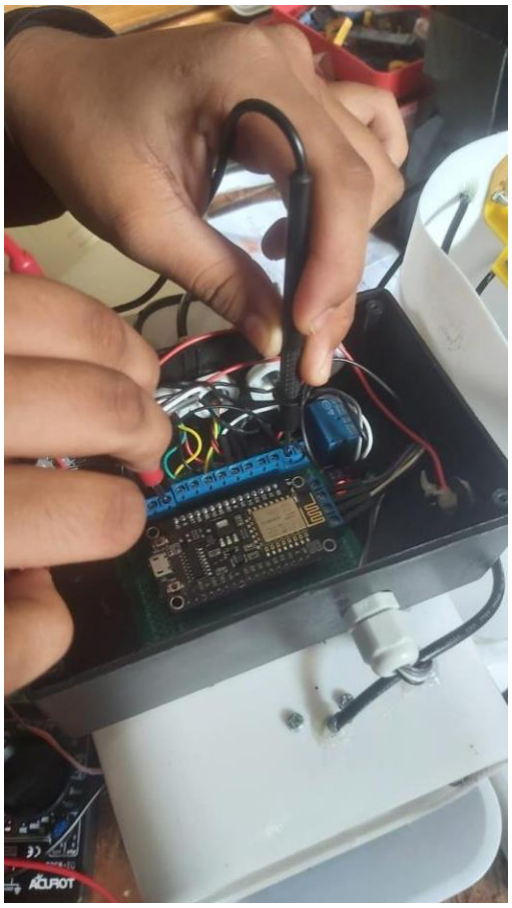
(Ibnu Maja, S.Si.,M.M.)
NIP. 197604052005011002

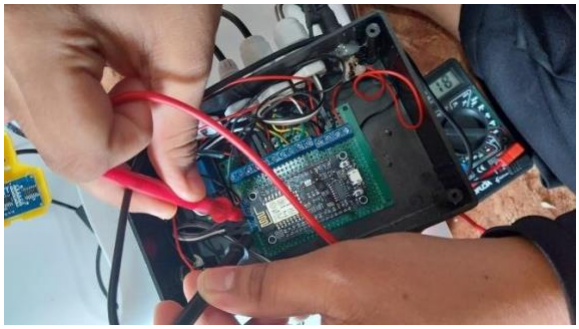
Mengetahui,
Ketua Jurusan

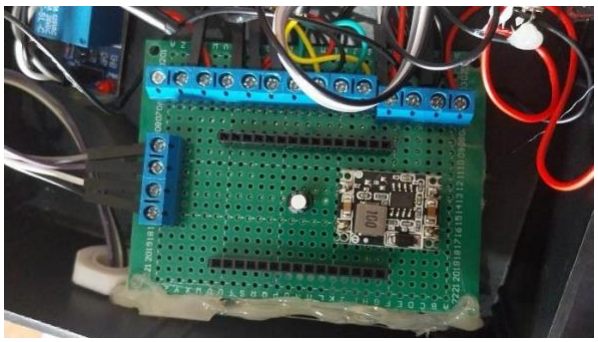
(Ir. Iskandar Lutfi, M.T)
NIP. 196501291991031002

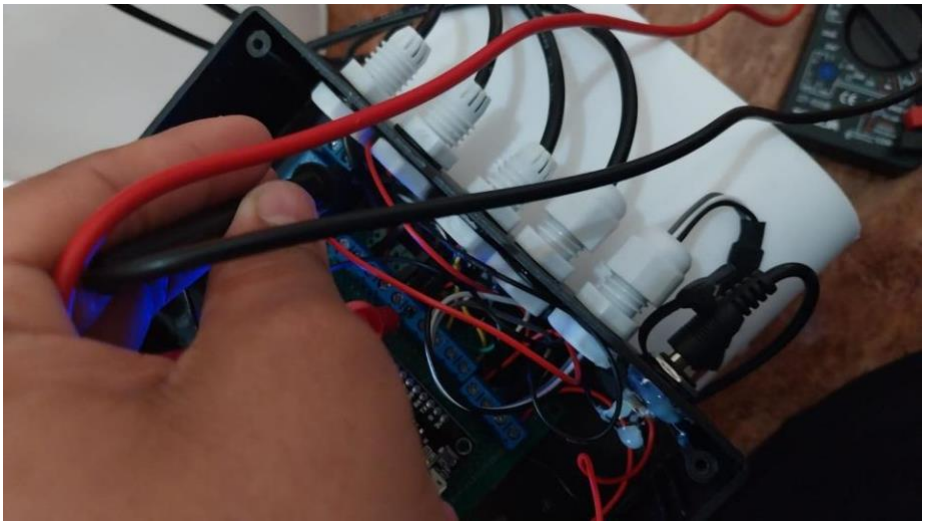
LAMPIRAN B


Kap. Rakamis ON
Kap. Minumis ON









LAMPIRAN C

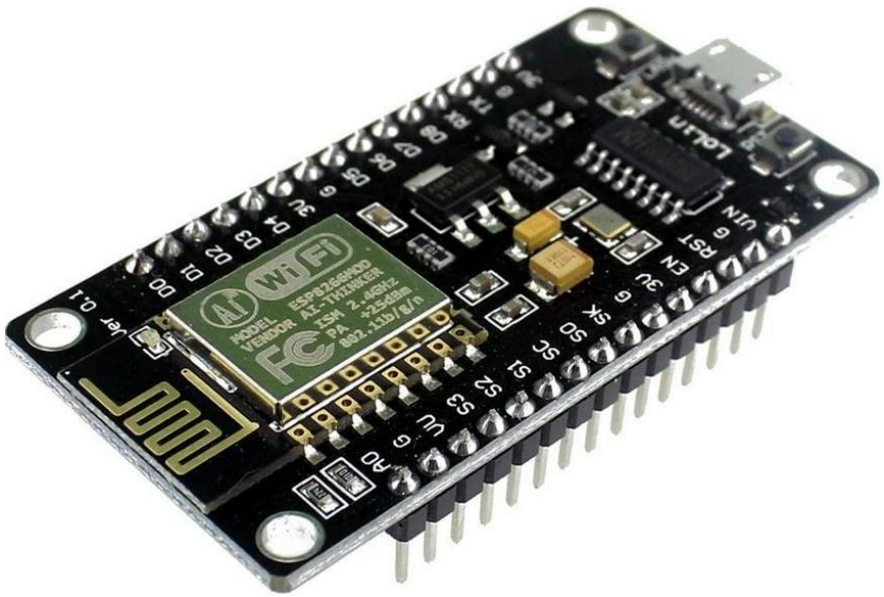
Handson Technology

User Manual V1.3

ESP8266 NodeMCU WiFi Development Board

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.



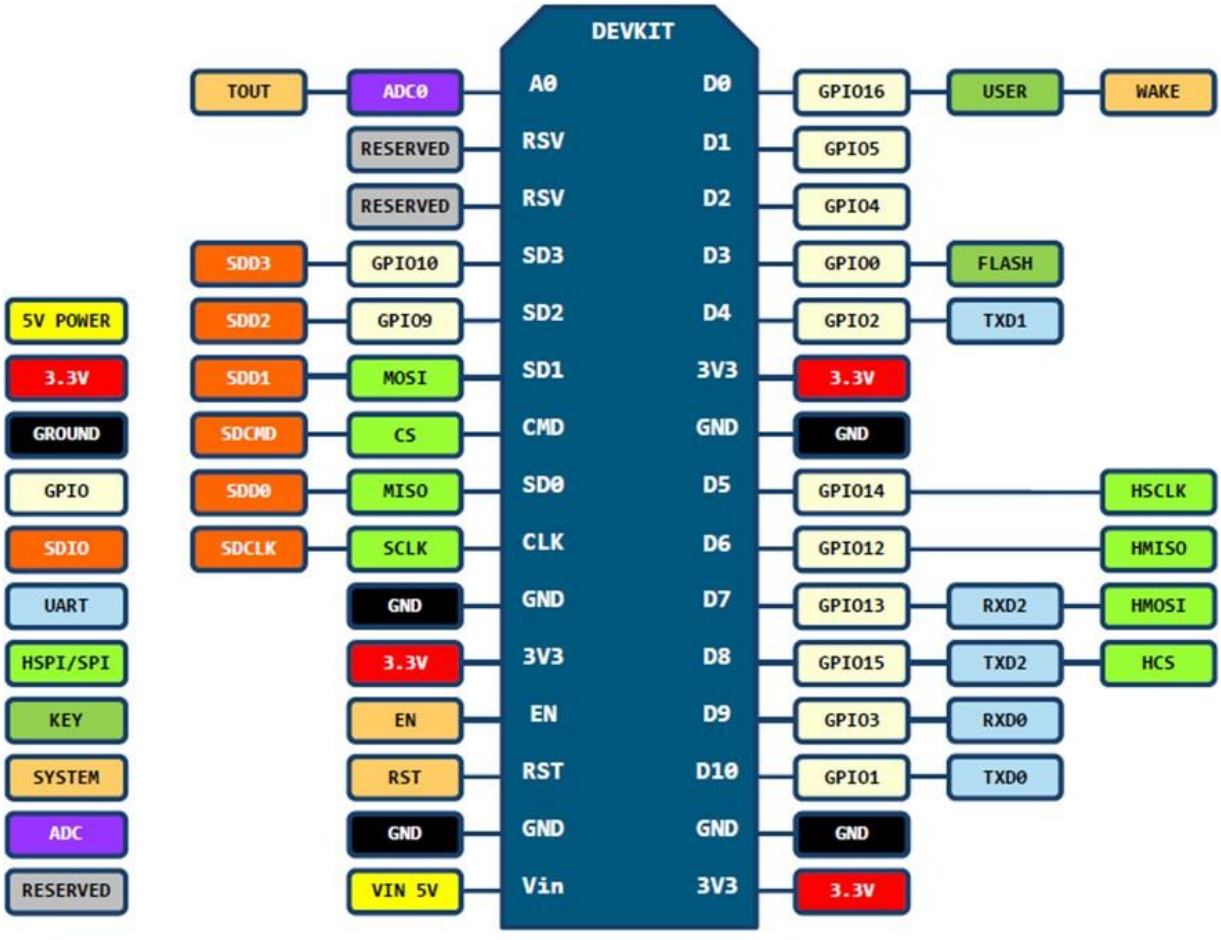
Getting Started User Guide

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.

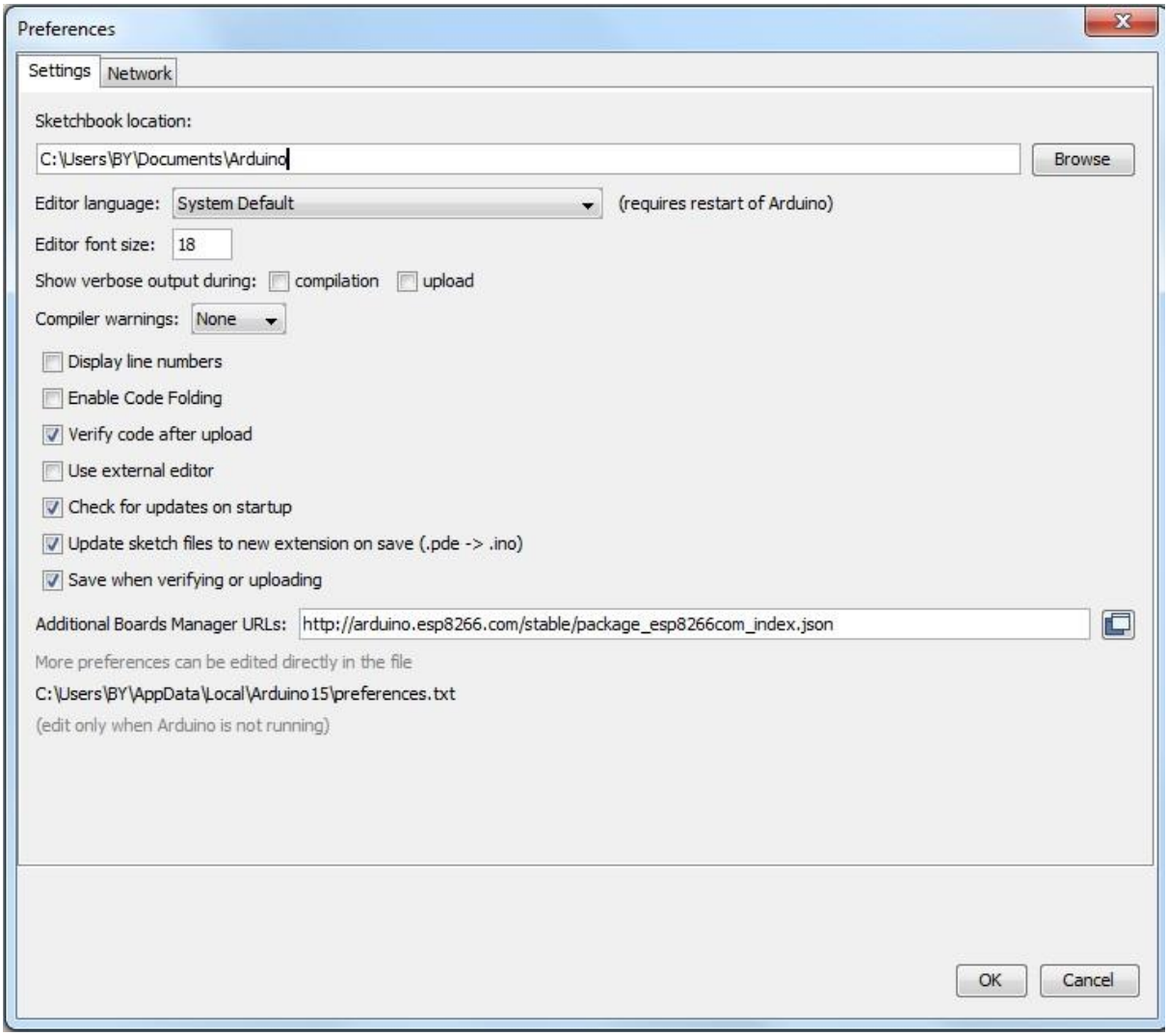
3.1 Install the Arduino IDE 1.6.4 or greater

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.

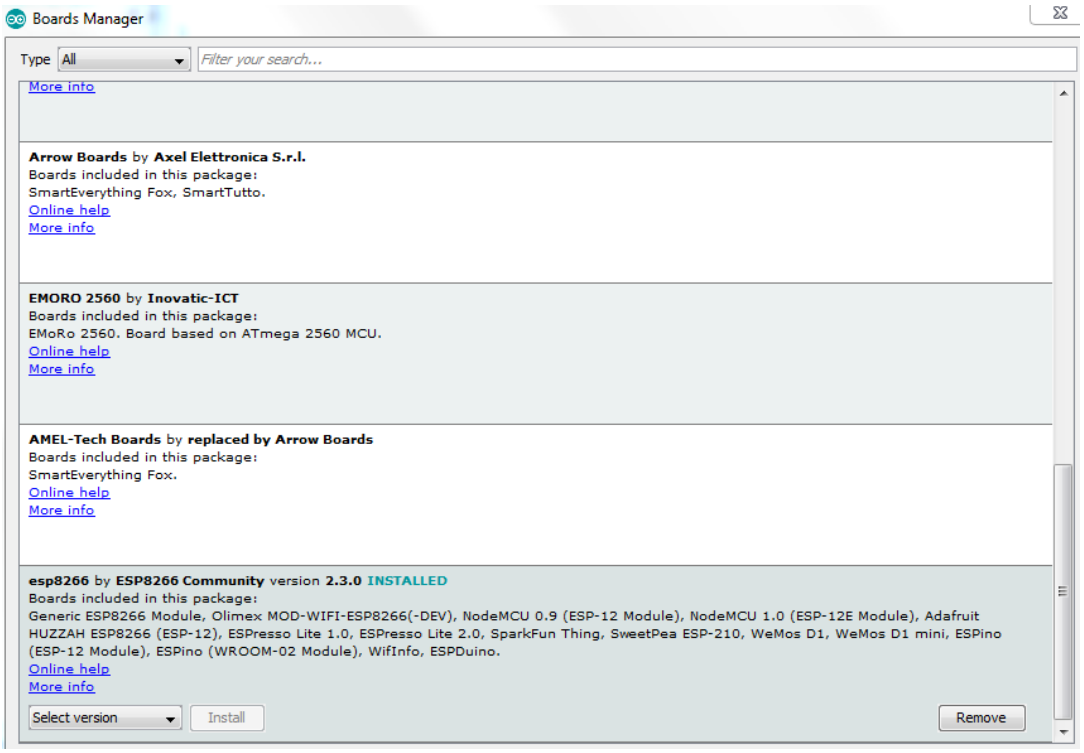
3.2 Install the ESP8266 Board Package

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.

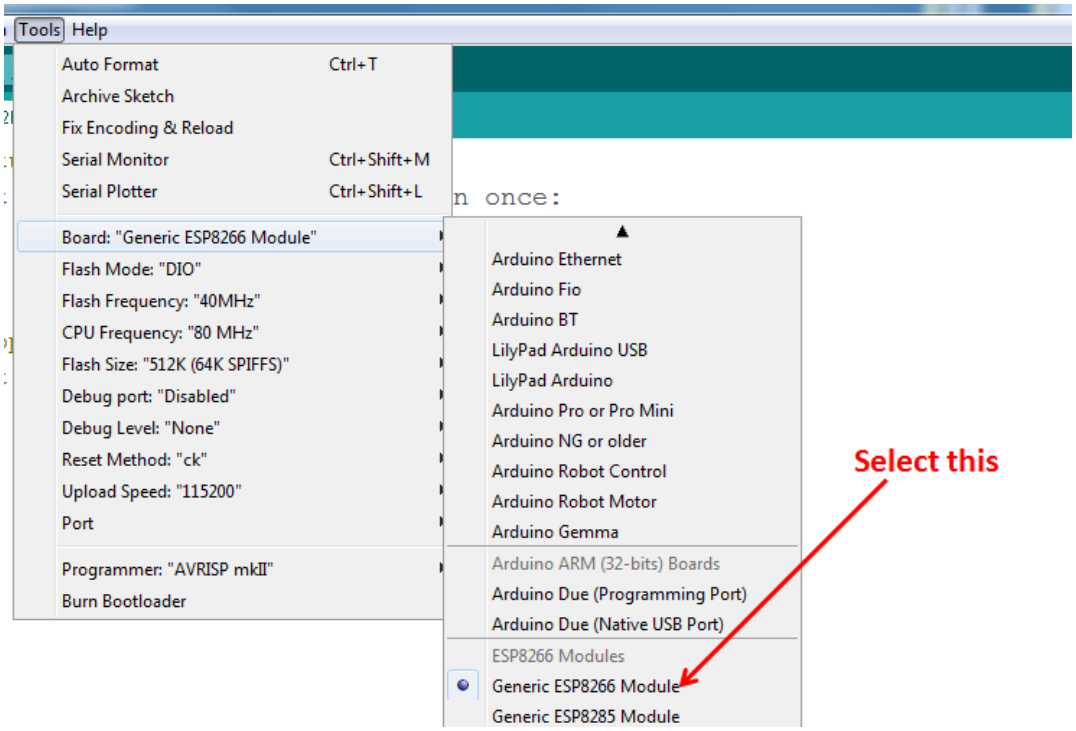


Click 'Tools' -> 'Board:' -> 'Board Manager…' to access this panel.

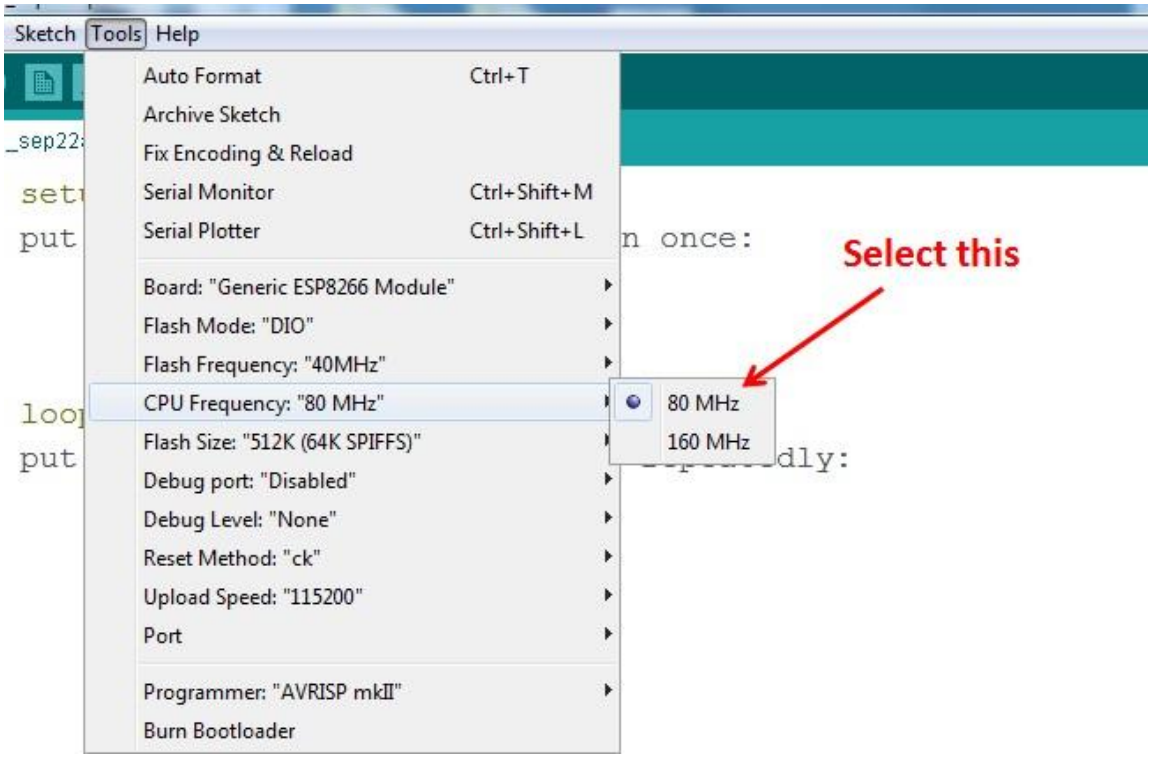
Scroll down to 'esp8266 by ESP8266 Community' and click "Install" button to install the ESP8266 library package. Once installation completed, close and re-open Arduino IDE for ESP8266 library to take effect.

3.3 Setup ESP8266 Support

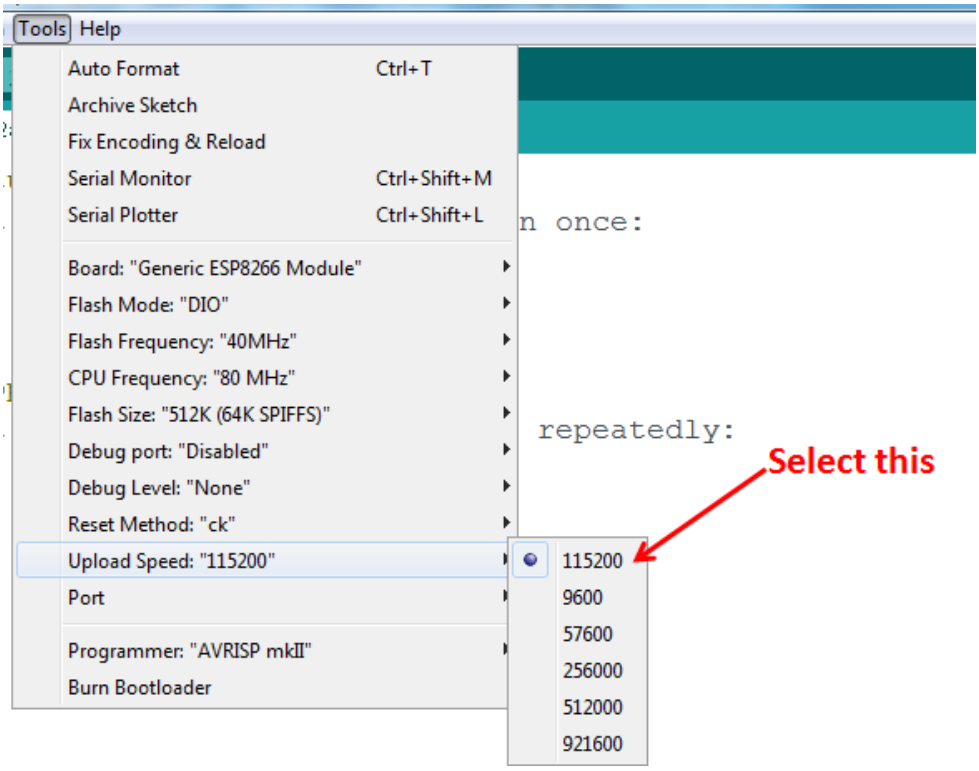
When you've restarted Arduino IDE, select 'Generic ESP8266 Module' from the 'Tools' -> 'Board:' dropdown menu.



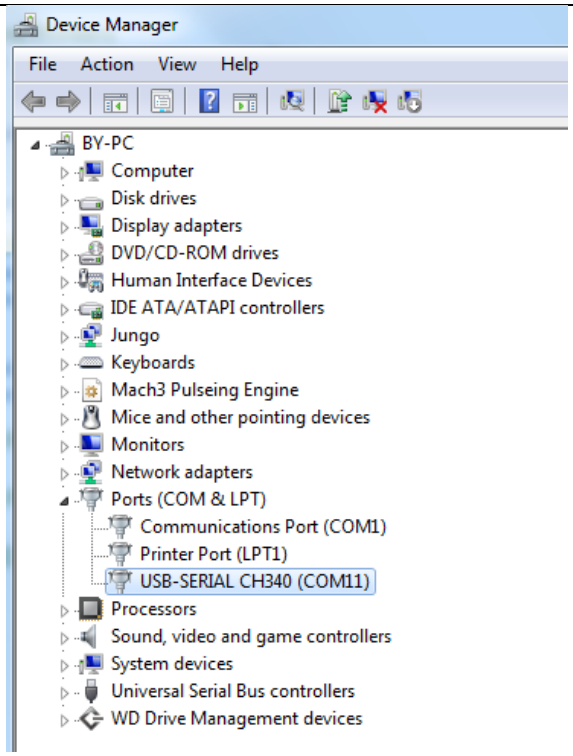
Select 80 MHz as the CPU frequency (you can try 160 MHz overclock later)



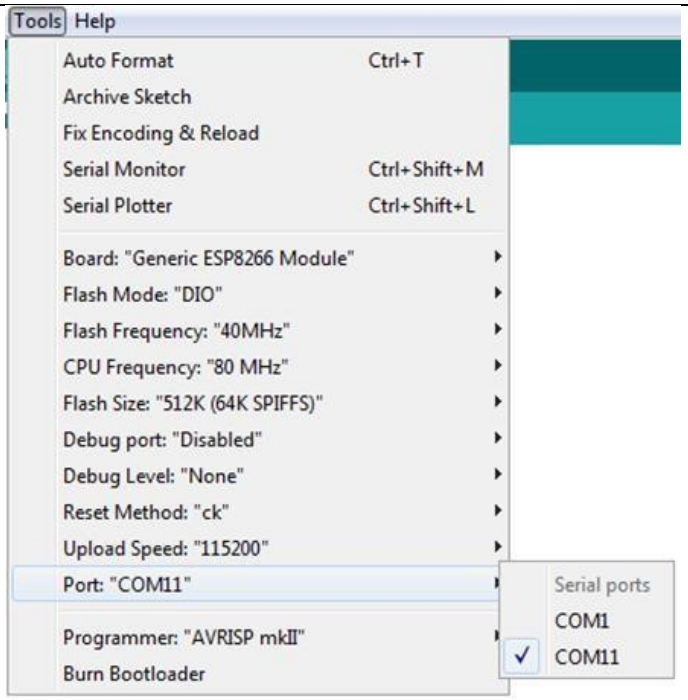
Select '115200' baud upload speed is a good place to start - later on you can try higher speeds but 115200 is a good safe place to start.



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.

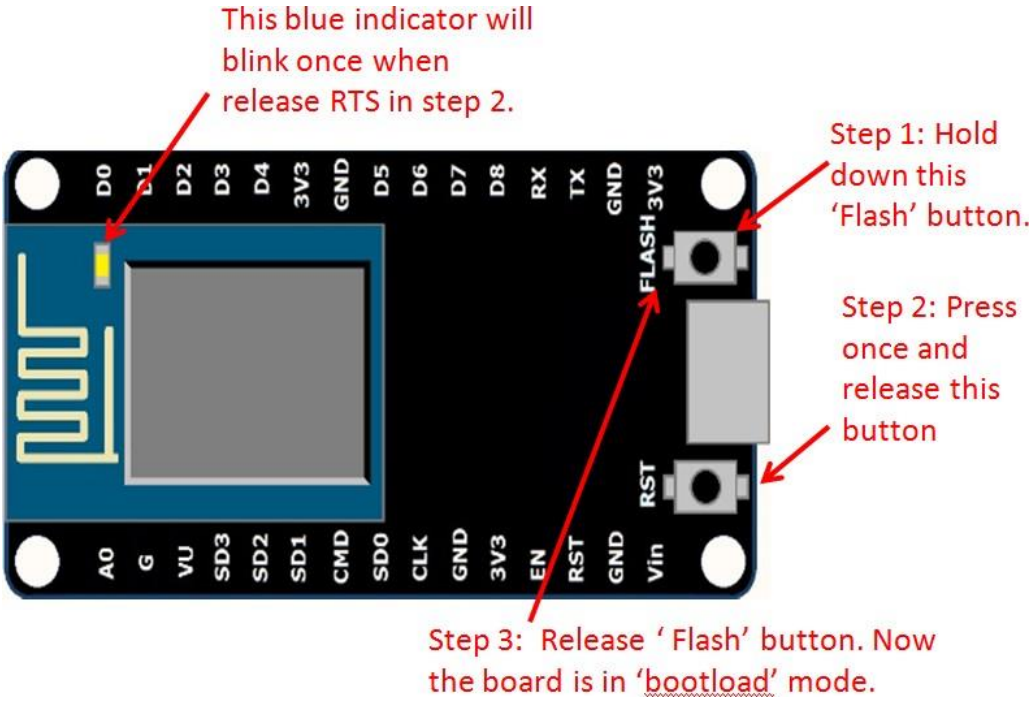
Enter this into the sketch window (and save since you'll have to). Connect a LED as shown in Figure3-1.

```
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 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.



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

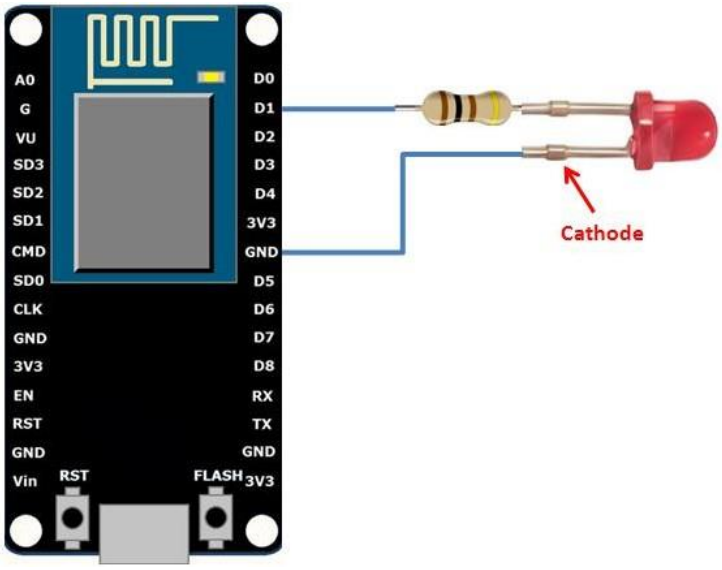


Figure3-1: Connection diagram for the blinking test

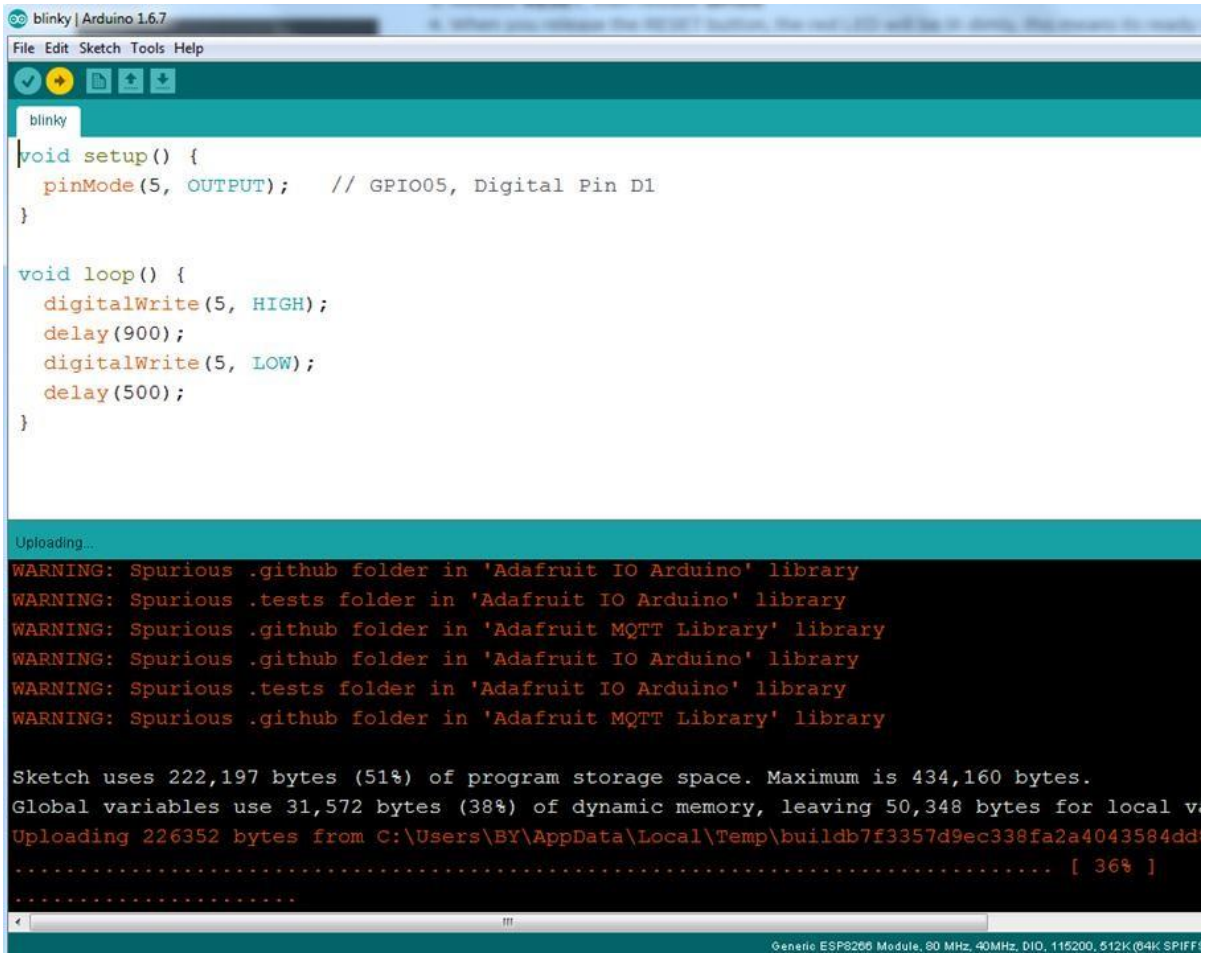


Figure 3.2: Uploading the sketch to ESP8266 NodeMCU module.

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";

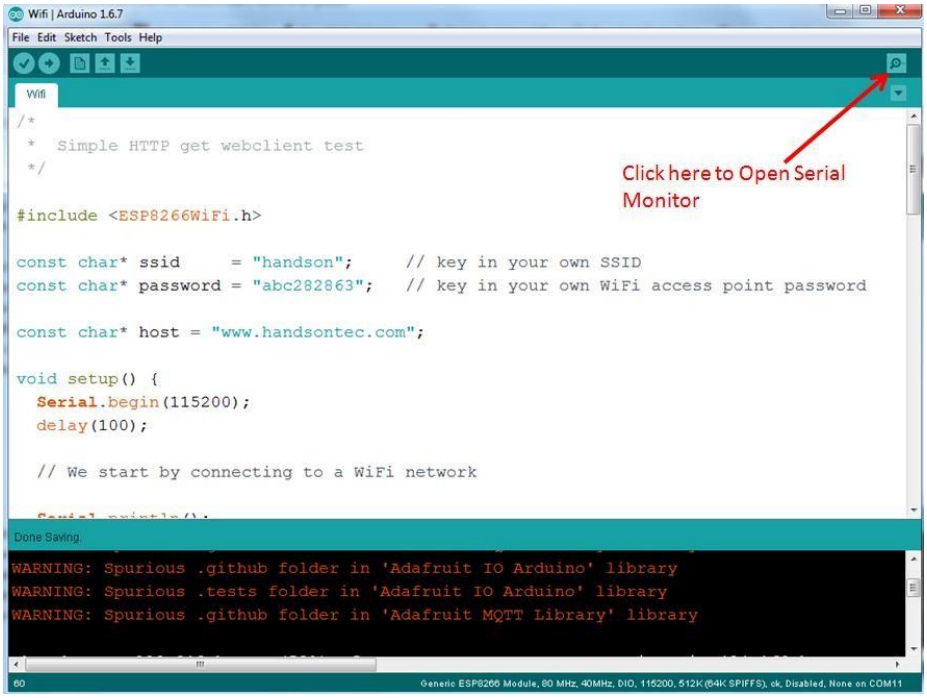
const char* password = "yourpassword";

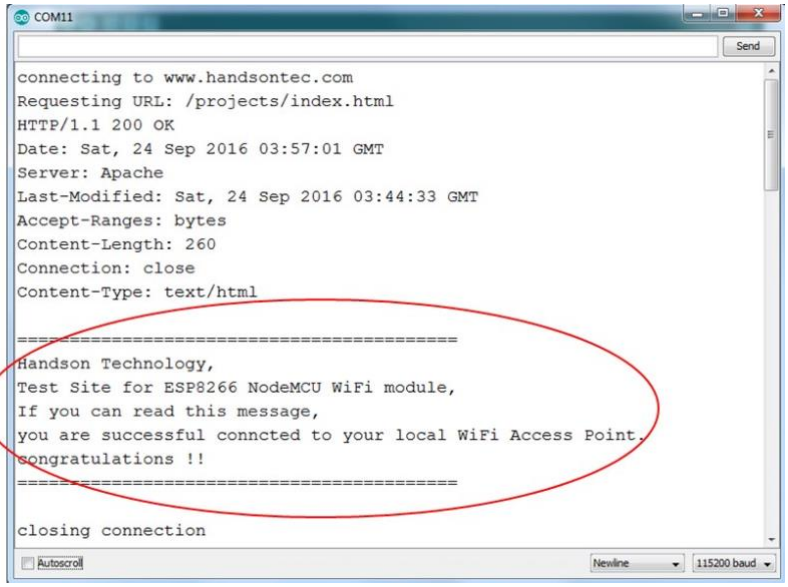
to your WiFi access point and password, then upload the same way: get into bootload mode, then upload code via IDE.

```
/*
 *  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.

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.

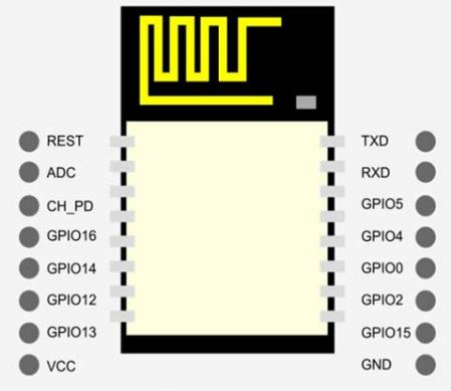
In this tutorial we are going to use another ESP8266 module with pin header adapter board which is breadboard friendly.

ESP8266 Module Breadboard Friendly with Header Connector

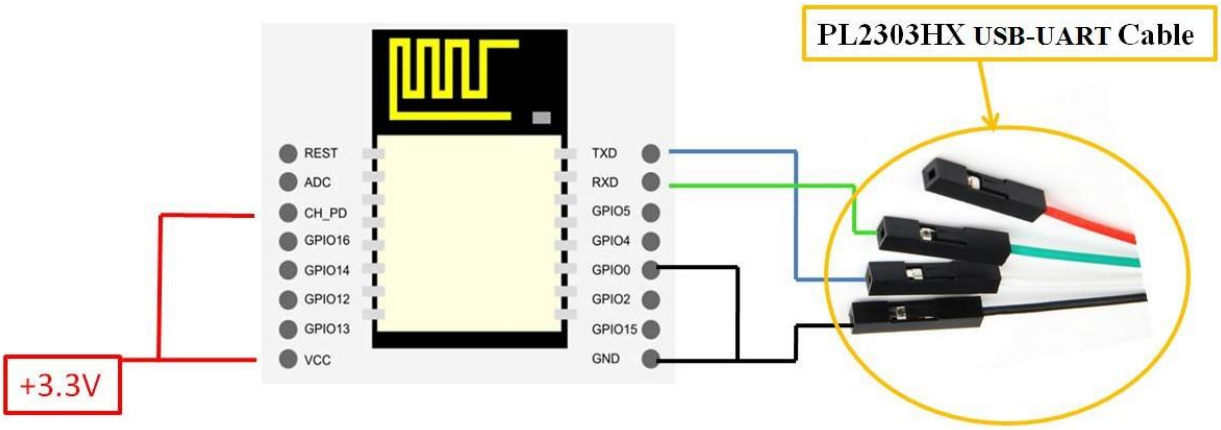
4.1 Parts Required:

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

4.2 Pin Assignment:



4.3 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

4.4 Downloading NodeMCU Flasher for Windows

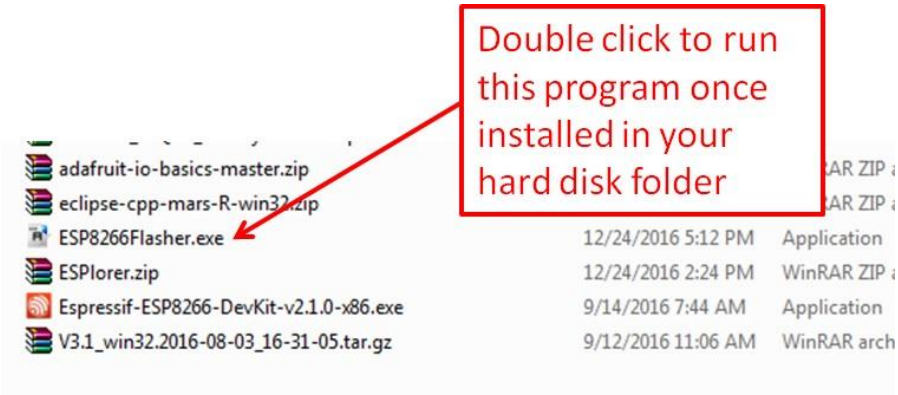
After wiring your circuit, you have to download the NodeMCU flasher. This is a .exe file that you can download using one of the following links:

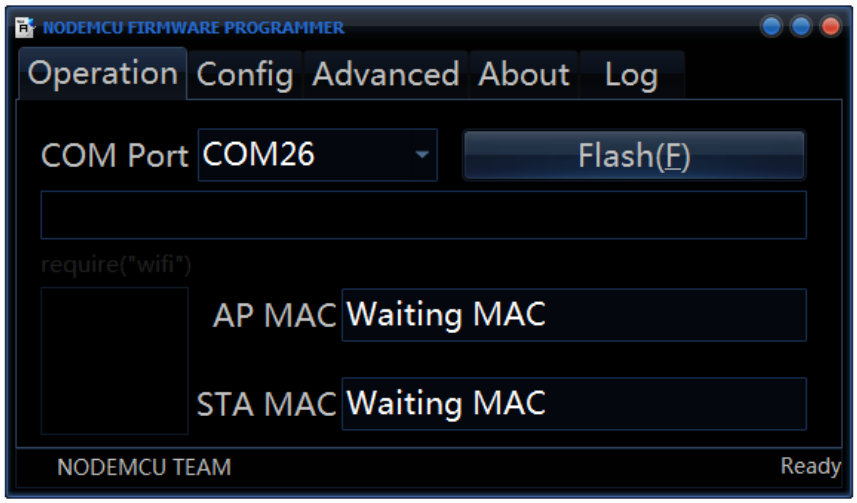
- Win32 Windows Flasher
- Win64 Windows Flasher

You can find all the information about NodeMCU flasher here.

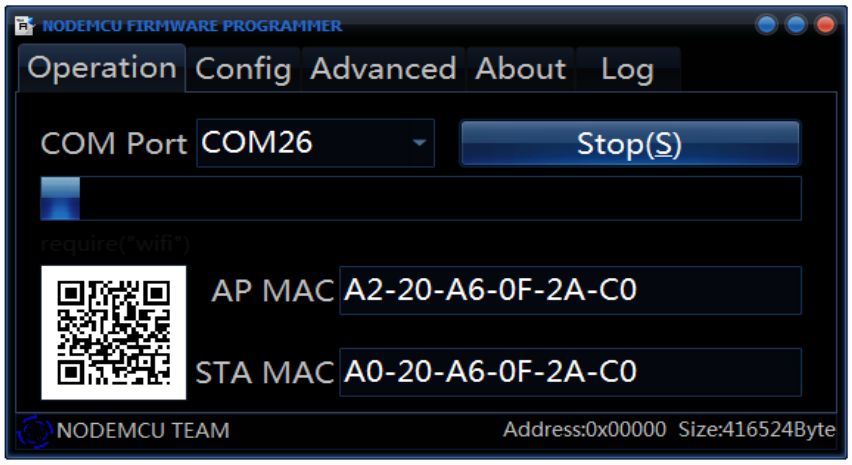
4.5 Flashing your ESP8266 using Windows

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

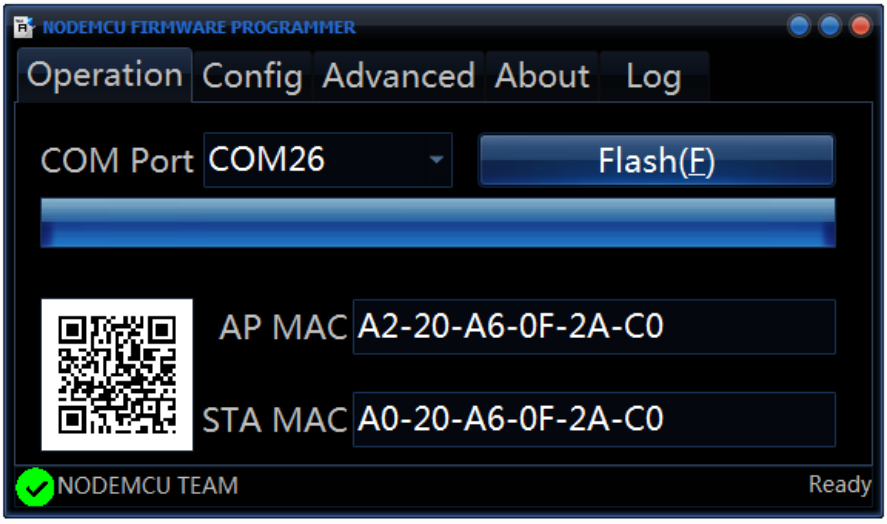




Press the button "Flash" and it should start the flashing process immediately, showing the Module MAC address if successful connected.



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



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.

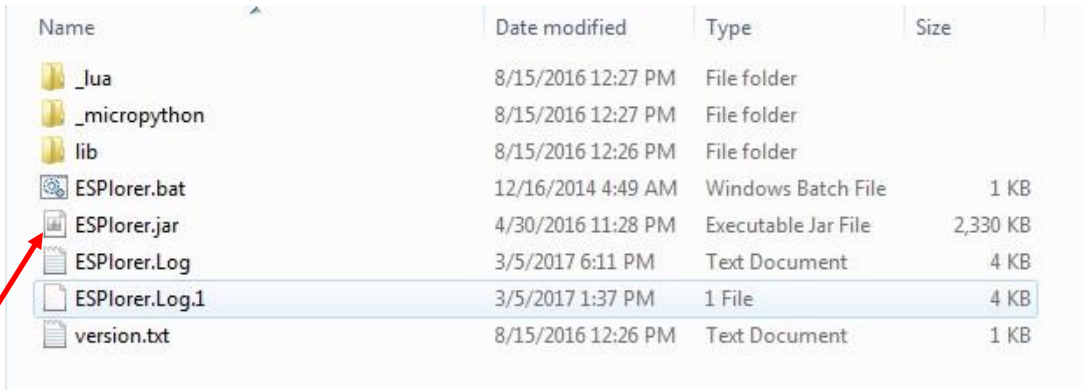
Main Resources:

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

5.1 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:



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.

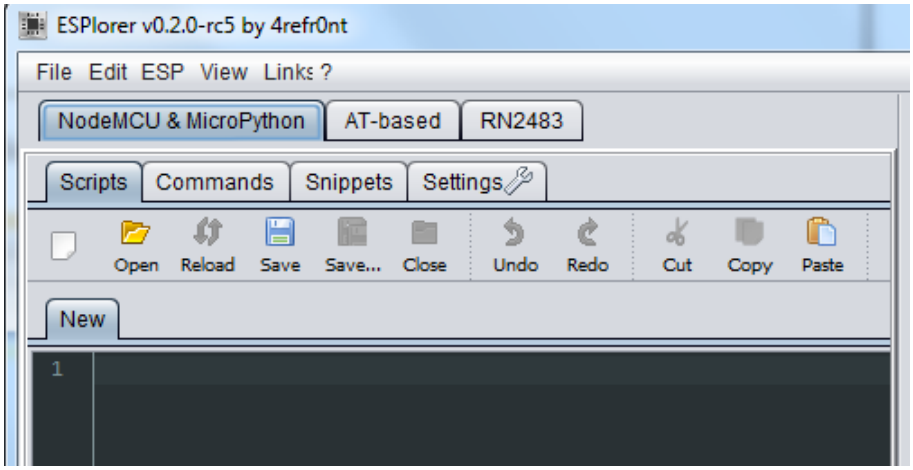
When the ESPlorer first opens, that's what you should see:

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…

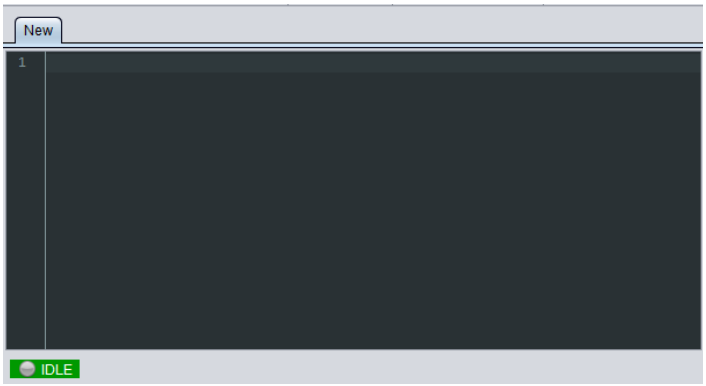
The ESPlorer IDE has a couple of main sections, let's break it down each one.

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.

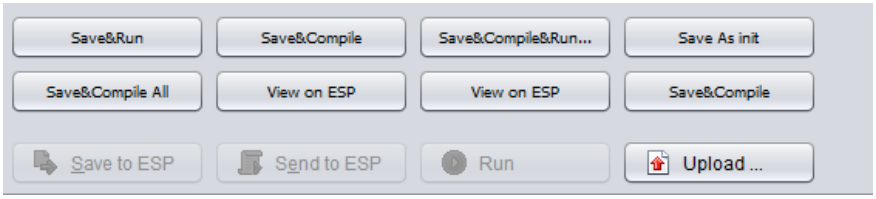


In the top right corner you have all the options you need to establish a serial communication (you're going to learn how to use them later in this Guide).

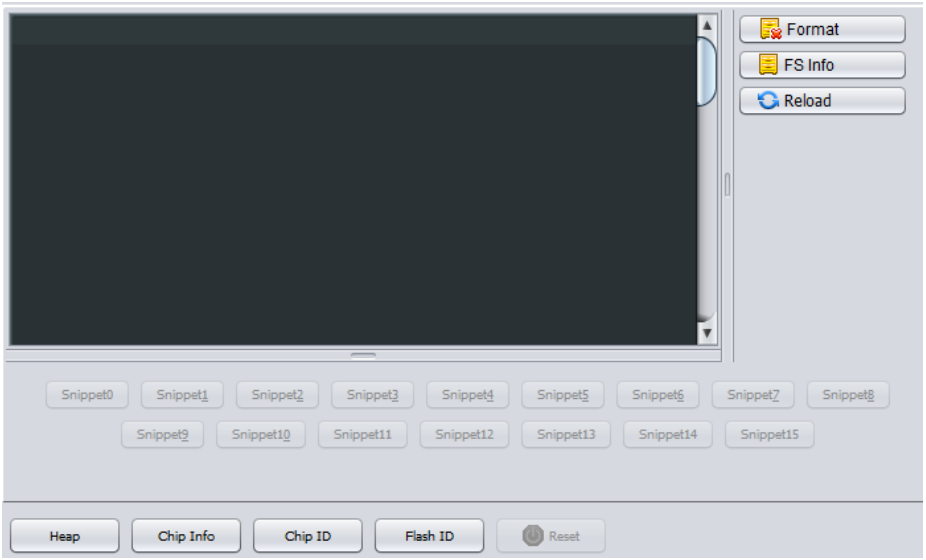
This next screenshot shows your Code Window, that's where you write your scripts (your scripts are highlighted with your code syntax).



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".

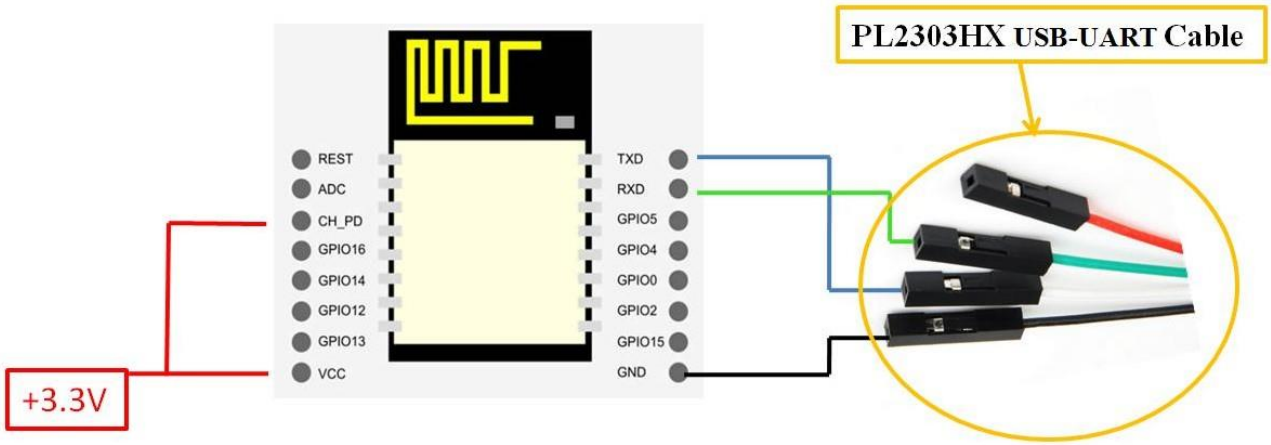


This screenshot shows the Output Window which tells you exactly what's going on in your ESP8266. You can see errors and use prints in your code to debug your projects.



5.2 Schematics

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



5.3 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)
```

init.lua

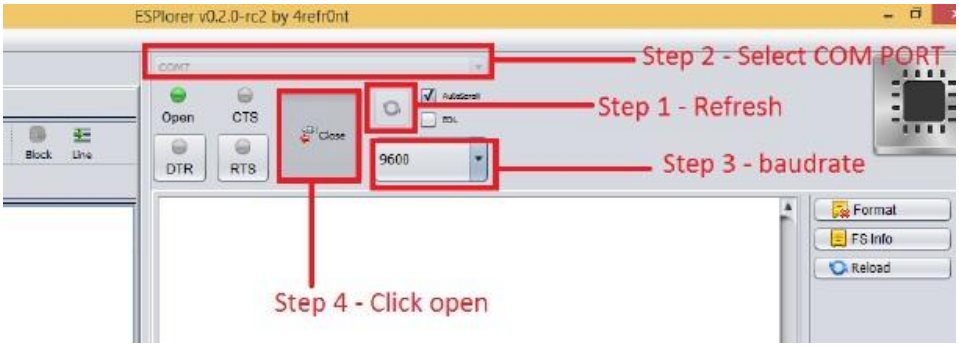
```
1  lighton=0
2  pin=4
3  gpio.mode(pin,gpio.OUTPUT)
4  tmr.alarm(1,1000,1,function()
5      if lighton==0 then
6          lighton=1
7          gpio.write(pin,gpio.HIGH)
8      else
9          lighton=0
10          gpio.write(pin,gpio.LOW)
11     end
12 end)
13
```

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:

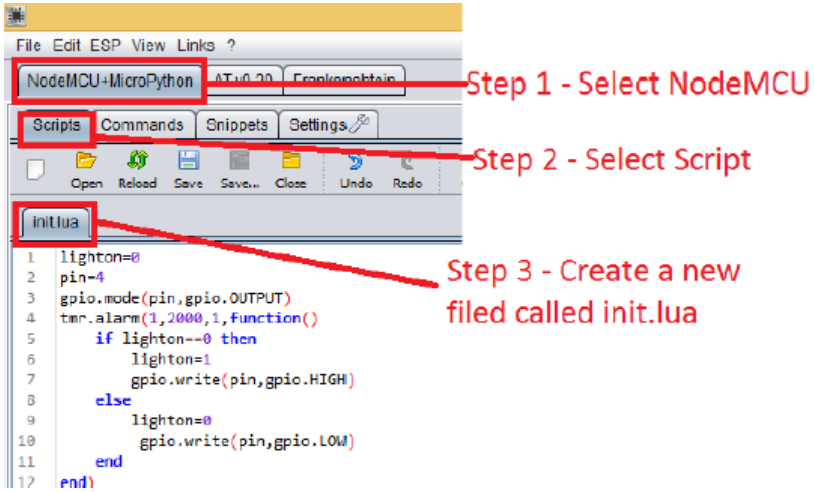
Look at the top right corner of your ESPlorer IDE and follow these instructions:

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

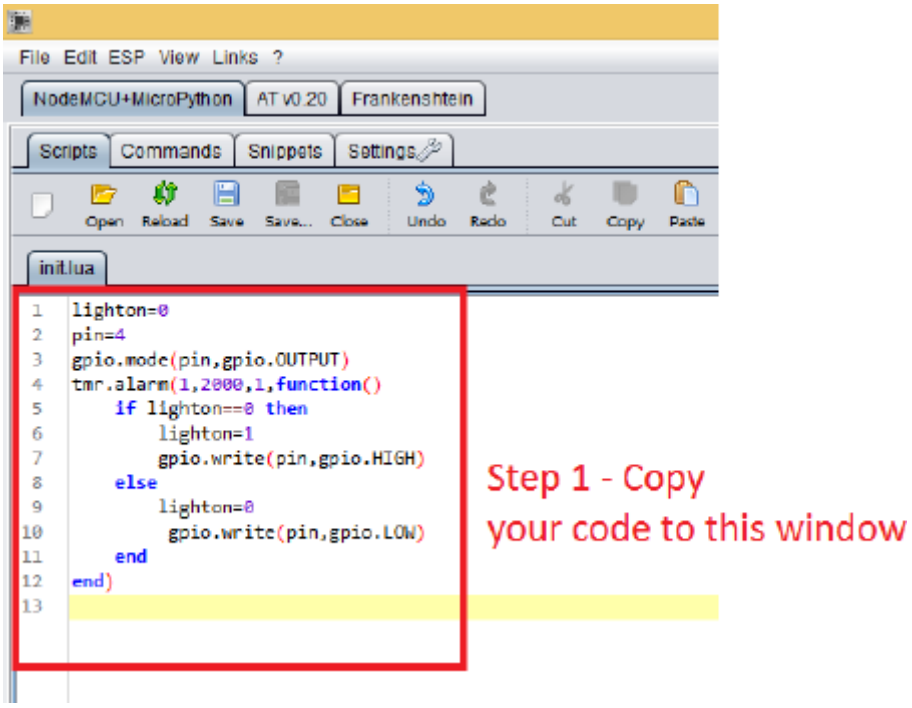


Then in the top left corner of your ESPlorer IDE, follow these instructions:

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



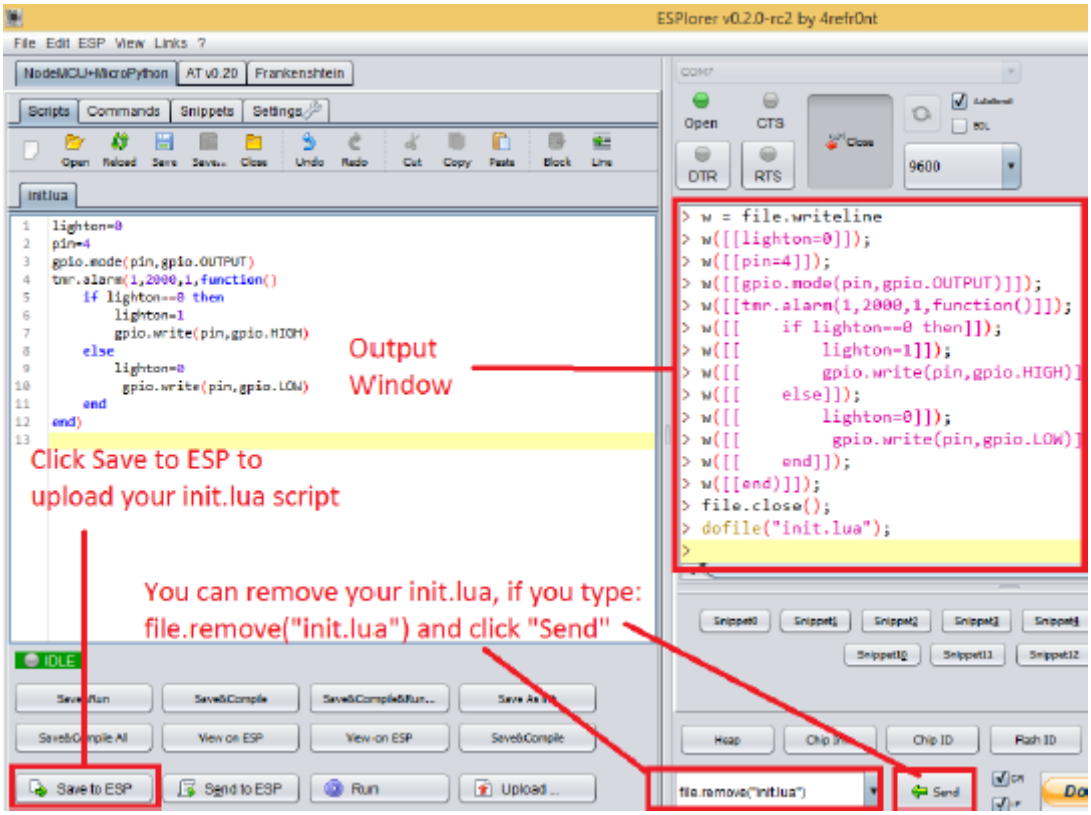
Copy your Lua script to the code window (as you can see in the Figure below):



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

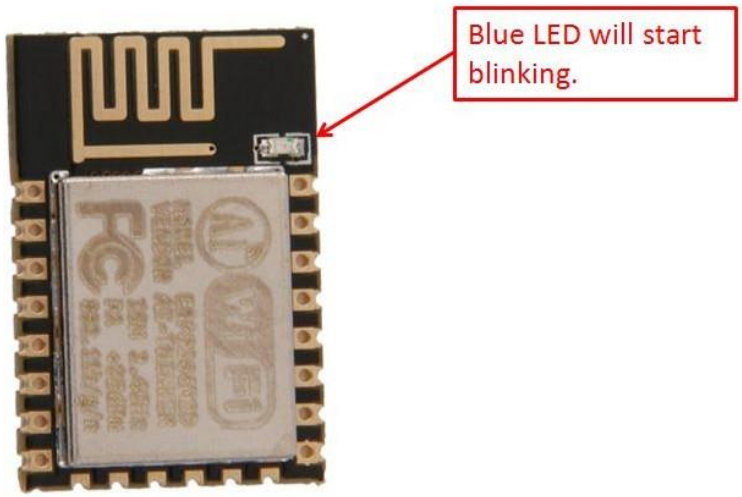
At the left bottom corner click the button “Save to ESP”.

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



Note: 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 Friendly Module
- ESP8266 Remote Serial WIFI Module
- PL2303HX USB-UART Converter Cable

HandsOn Technology provides a multimedia and interactive platform for everyone interested in electronics. From beginner to diehard, from student to lecturer. Information, education, inspiration and entertainment. Analog and digital, practical and theoretical; software and hardware.

open source hardware

Hands*On* Technology support Open Source Hardware (OSHW) Development Platform.

Learn : Design : Share

handsontec.com

The Face behind our product quality…

In a world of constant change and continuous technological development, a new or replacement product is never far away – and they all need to be tested.
Many vendors simply import and sell wihtout checks and this cannot be the ultimate interests of anyone, particularly the customer. Every part sell on Handsotec is fully tested. So when buying from Handsontec products range, you can be confident you're getting outstanding quality and value.

We keep adding the new parts so that you can get rolling on your next project.

Breakout Boards & Modules

Connectors

Electro-Mechanical Parts

Engineering Material

Mechanical Hardware

Electronics Components

Power Supply

Arduino Board & Shield

Tools & Accessory

HC-SR04 Ultrasonic Sensor

Elijah J. Morgan

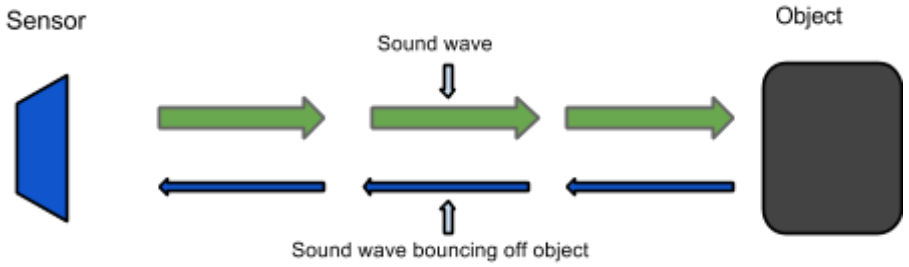
Nov. 16 2014

The purpose of this file is to explain how the HC-SR04 works. It will give a brief explanation of how ultrasonic sensors work in general. It will also explain how to wire the sensor up to a microcontroller and how to take/interpret readings. It will also discuss some sources of errors and bad readings.

1. **How Ultrasonic Sensors Work**
2. **HC-SR04 Specifications**
3. **Timing chart, Pin explanations and Taking Distance Measurements**
4. **Wiring HC-SR04 with a microcontroller**
5. **Errors and Bad Readings**

1. How Ultrasonic Sensors Work

Ultrasonic sensors use sound to determine the distance between the sensor and the closest object in its path. How do ultrasonic sensors do this? Ultrasonic sensors are essentially sound sensors, but they operate at a frequency above human hearing.



The sensor sends out a sound wave at a specific frequency. It then listens for that specific sound wave to bounce off of an object and come back (Figure 1). The sensor keeps track of the time between sending the sound wave and the sound wave returning. If you know how fast something is going and how long it is traveling you can find the distance traveled with equation 1.

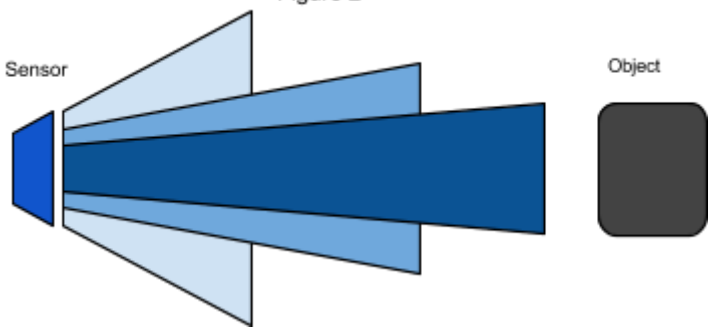
Equation 1. $d = v \times t$

The speed of sound can be calculated based on the a variety of atmospheric conditions, including temperature, humidity and pressure. Actually calculating the distance will be shown later on in this document.

It should be noted that ultrasonic sensors have a cone of detection, the angle of this cone varies with distance, Figure 2 show this relation. The ability of a sensor to

detect an object also depends on the objects orientation to the sensor. If an object doesn't present a flat surface to the sensor then it is possible the sound wave will bounce off the object in a way that it does not return to the sensor.

Figure 2



2. HC-SR04 Specifications

The sensor chosen for the Firefighting Drone Project was the HC-SR04. This section contains the specifications and why they are important to the sensor module. The sensor modules requirements are as follows.

- Cost
- Weight
- Community of hobbyists and support
- Accuracy of object detection
- Probability of working in a smoky environment
- Ease of use

The HC-SR04 Specifications are listed below. These specifications are from the Cytron Technologies HC-SR04 User's Manual (source 1).

- Power Supply: +5V DC
- Quiescent Current: <2mA
- Working current: 15mA
- Effectual Angle: <15º
- Ranging Distance: 2-400 cm
- Resolution: 0.3 cm
- Measuring Angle: 30º
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm
- Weight: approx. 10 g

The HC-SR04's best selling point is its price; it can be purchased at around $2 per unit.

3. Timing Chart and Pin Explanations

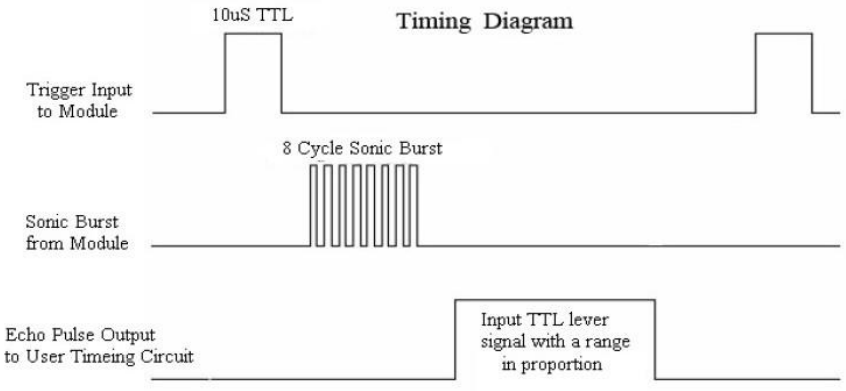
The HC-SR04 has four pins, VCC, GND, TRIG and ECHO; these pins all have different functions. The VCC and GND pins are the simplest -- they power the HC-SR04. These pins need to be attached to a +5 volt source and ground respectively. There is a single control pin: the TRIG pin. The TRIG pin is responsible for sending the ultrasonic burst. This pin should be set to HIGH for 10 μs, at which point the HC-SR04 will send out an eight cycle sonic burst at 40 kHZ. After a sonic burst has been sent the ECHO pin will go HIGH. The ECHO pin is the data pin -- it is used in taking distance measurements. After an ultrasonic burst is sent the pin will go HIGH, it will stay high until an ultrasonic burst is detected back, at which point it will go LOW.

Taking Distance Measurements

The HC-SR04 can be triggered to send out an ultrasonic burst by setting the TRIG pin to HIGH. Once the burst is sent the ECHO pin will automatically go HIGH. This pin will remain HIGH until the the burst hits the sensor again. You can calculate the distance to the object by keeping track of how long the ECHO pin stays HIGH. The time ECHO stays HIGH is the time the burst spent traveling. Using this measurement in equation 1 along with the speed of sound will yield the distance travelled. A summary of this is listed below, along with a visual representation in Figure 2.

1. Set TRIG to HIGH
2. Set a timer when ECHO goes to HIGH
3. Keep the timer running until ECHO goes to LOW
4. Save that time
5. Use equation 1 to determine the distance travelled

Figure 3
Source 2



Source 2

To interpret the time reading into a distance you need to change equation 1. The clock on the device you are using will probably count in microseconds or smaller. To use equation 1 the speed of sound needs to determined,which is 343 meters per second at standard temperature and pressure. To convert this into more useful form use equation 2 to change from meters per second to microseconds per centimeter. Then equation 3 can be used to easily compute the distance in centimeters.

Equation 2. $Distance = \frac{Speed}{170.15\ m} \times \frac{Meters}{100\ cm} \times \frac{1e6\ \mu S}{170.15\ m} \times \frac{58.772\ \mu S}{cm}$

Equation 3. $Distance = \frac{time - \mu s}{58\ \mu s/cm} = cm$

4. Wiring the HC-SR04 to a Microcontroller

This section only covers the hardware side. For information on how to integrate the software side, look at one of the links below or look into the specific microcontroller you are using.

The HC-SR04 has 4 pins: VCC, GND, TRIG and ECHO.

1. VCC is a 5v power supply. This should come from the microcontroller
2. GND is a ground pin. Attach to ground on the microcontroller.
3. TRIG should be attached to a GPIO pin that can be set to HIGH
4. ECHO is a little more difficult. The HC-SR04 outputs 5v, which could destroy many microcontroller GPIO pins (the maximum allowed voltage varies). In order to step down the voltage use a single resistor or a voltage divider circuit. Once again this depends on the specific microcontroller you are using, you will need to find out its GPIO maximum voltage and make sure you are below that.

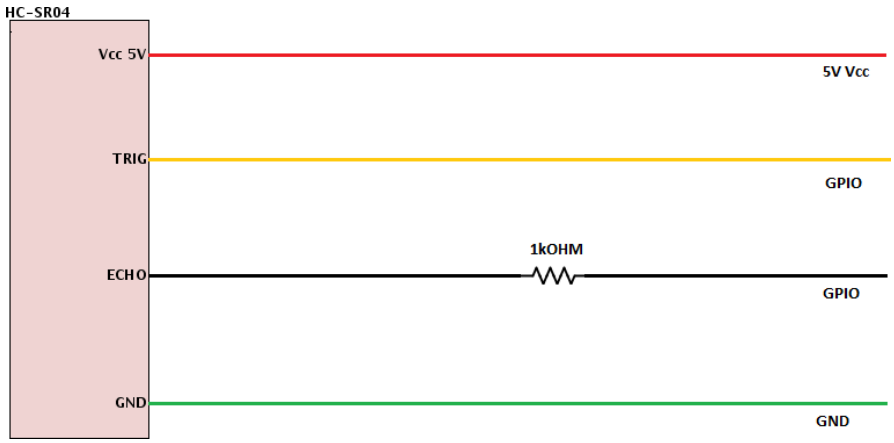
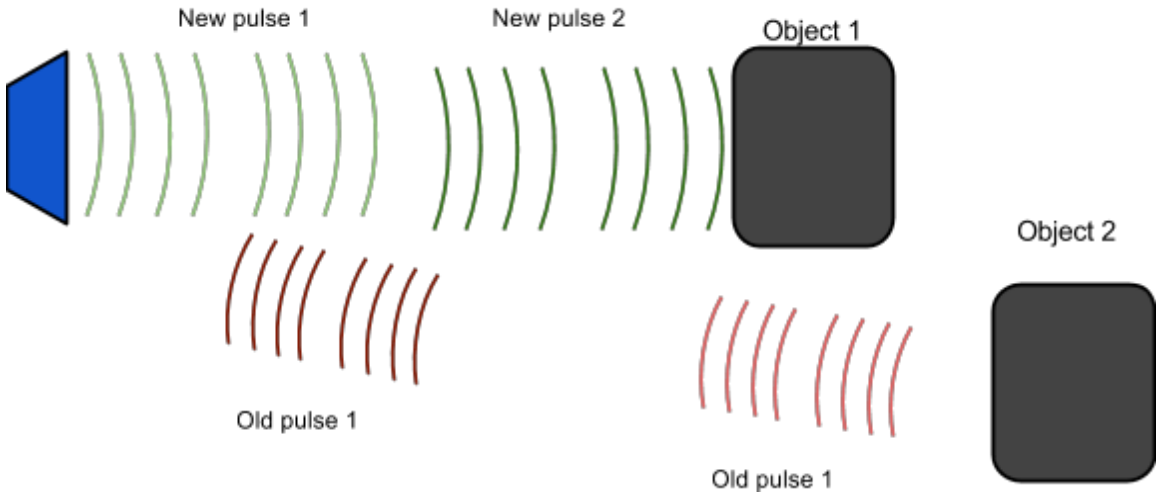


Figure 4

5. Errors and Bad Readings

Ultrasonic sensors are great sensors -- they work well for many applications where other types of sensors fall short. Unfortunately, they do have weaknesses. These weaknesses can be mitigated and worked around, but first they must be understood. The

first weakness is that they use sound. There is a limit to how fast ultrasonic sensors can get distance measurements. The longer the distance, the slower they are at reporting the distance. The second weakness comes from the way sound bounces off of objects. In enclosed spaces it is possible, if not probable that there will be unintended echos. The echos can very easily cause false short readings. In Figure 2 a pulse was sent out. It bounced off of object 1 and returned to the sensor. The distance was recorded and then a new pulse was sent. There was another object farther away, so that when the new pulse reaches object 1, the first signal will reach the sensor. This will cause the sensor to think that there is an object closer than is actually true. The old pulse is smaller than the new pulse because it has grown weaker. The longer the pulse exists the weaker it grows until it is negligible. If multiple sensors are being used, the number of echos will increase along with the number of errors. There are two main ways to reduce the number of errors. The first is to provide shielding around the sensor. This prevents echos coming in from angle outside what the sensor should actually pick up. The second is to reduce the frequency at which pulses are sent out. This gives more time for the echos to dissipate.



Works Cited

Source 1.
"HC-SR04 User's_Manual." *docs.google.* Cytron Technologies, May 2013 Web. 5 Dec. 2009.
<https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit>

Source 2.
"Attiny2313 Ultrasonic distance (HR-SR04) example." *CircuitDB.* n.a. 7 Sept. 2014 Web. 5 Dec. 2014. <http://www.circuitdb.com/?p=1162>

Links

These are not formatted; you will need to copy and paste them into your web browser.

Want to learn about Ultrasonic Sensors in general?
http://www.sensorsmag.com/sensors/acoustic-ultrasound/choosing-ultrasonic-sensor-proximity-or-distance-measurement-825

All about the HC-SR04

- http://www.circuitdb.com/?p=1162
- http://www.micropik.com/PDF/HCSR04.pdf
- http://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/
- http://www.ezdenki.com/ultrasonic.php
  (^fantastic tutorial, explains a lot of stuff)
- http://www.elecrow.com/hcsr04-ultrasonic-ranging-sensor-p-316.html
  (^ this one has some cool charts)

ESP32-CAM Development Board

SKU:DFR0602

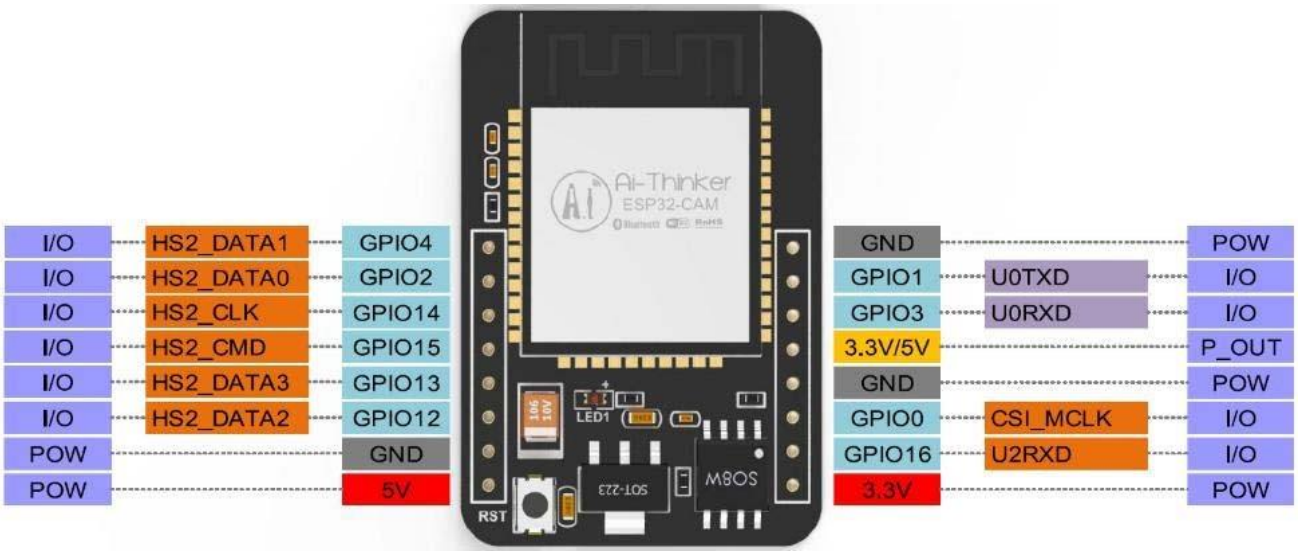
INTRODUCTION

ESP32-CAM is a low-cost ESP32-based development board with onboard camera, small in size. It is an ideal solution for IoT application, prototypes constructions and DIY projects.

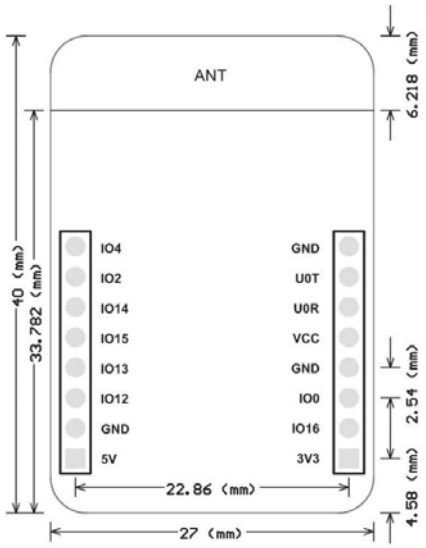
The board integrates WiFi, traditional Bluetooth and low power BLE , with 2 high-performance 32-bit LX6 CPUs. It adopts 7-stage pipeline architecture, on-chip sensor, Hall sensor, temperature sensor and so on, and its main frequency adjustment ranges from 80MHz to 240MHz.

Fully compliant with WiFi 802.11b/g/n/e/i and Bluetooth 4.2 standards, it can be used as a master mode to build an independent network controller, or as a slave to other host MCUs to add networking capabilities to existing devices

ESP32-CAM can be widely used in various IoT applications. It is suitable for home smart devices, industrial wireless control, wireless monitoring, QR wireless identification, wireless positioning system signals and other IoT applications. It is an ideal solution for IoT applications.



Schematic Diagram



Dimension Diagram

Notes:

1. Please be sure that the power supply for the module should be at least 5V 2A, otherwise maybe there would be water ripple appearing on the image.

2.ESP32 GPIO32 pin is used to control the power of the camera, so when the camera is in working, pull GPIO32 pin low.

3.Since IO pin is connected to camera XCLK, it should be left floating in using, and do not connect it to high/low level.

4.The product has been equipped with default firmware before leaving the factory, and we do not provide additional ones for you to download. So, please be cautious when you choose to burn other firmwares.

FEATURES

- Up to 160MHz clock speed，Summary computing power up to 600 DMIPS
- Built-in 520 KB SRAM, external 4MPSRAM
- Supports UART/SPI/I2C/PWM/ADC/DAC
- Support OV2640 and OV7670 cameras, Built-in Flash lamp.
- Support image WiFI upload
- Support TF card
- Supports multiple sleep modes.
- Embedded Lwip and FreeRTOS
- Supports STA/AP/STA+AP operation mode
- Support Smart Config/AirKiss technology
- Support for serial port local and remote firmware upgrades (FOTA)

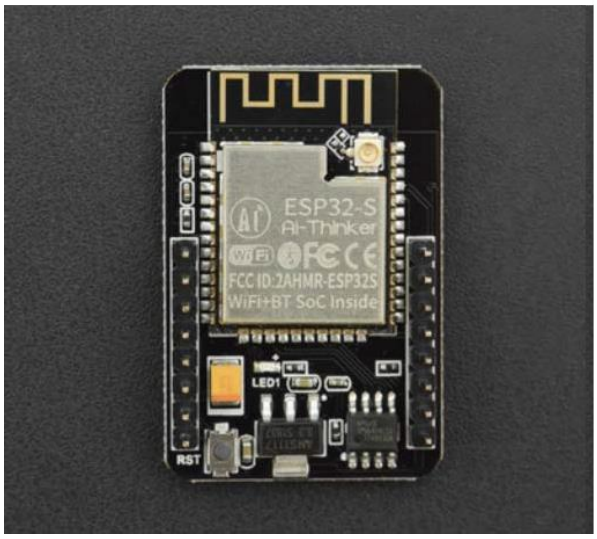
SPECIFICATION

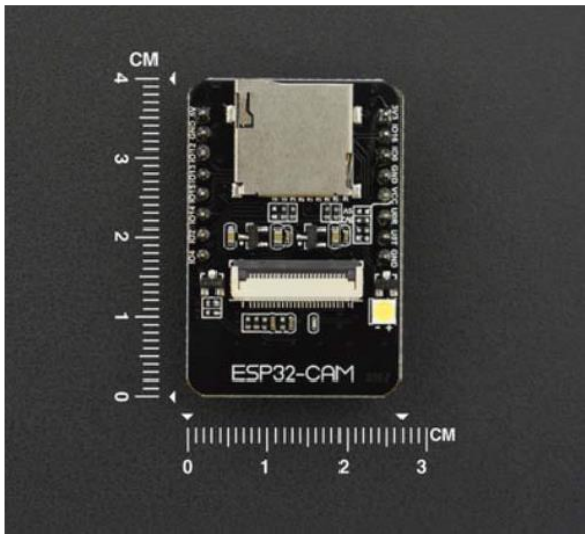
- SPI Flash: default 32Mbit
- RAM: built-in 520 KB+external 4MPSRAM
- Dimension: 27*40.5*4.5（±0.2）mm/1.06*1.59*0.18”
- Bluetooth: Bluetooth 4.2 BR/EDR and BLE standards
- Wi-Fi: 802.11b/g/n/e/i
- Support Interface: UART, SPI, I2C, PWM
- Support TF card: maximum support 4G
- IO port: 9
- Serial Port Baud-rate: Default 115200 bps

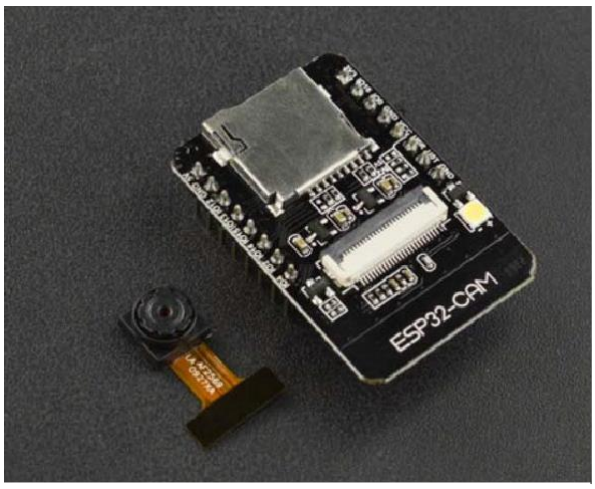
- Image Output Format: JPEG(OV2640 support only), BMP, GRAYSCALE
- Spectrum Range: 2412 ~2484MHz
- Antenna: onboard PCB antenna, gain 2dBi
- Transmit Power: 802.11b: 17±2 dBm (@11Mbps);
  802.11g: 14±2 dBm (@54Mbps);
  802.11n: 13±2 dBm (@MCS7)
- Receiving Sensitivity: CCK, 1 Mbps : -90dBm;
  CCK, 11 Mbps: -85dBm;
  6 Mbps (1/2 BPSK): -88dBm;
  54 Mbps (3/4 64-QAM): -70dBm;
  MCS7 (65 Mbps, 72.2 Mbps): -67dBm
- Power consumption: Turn off the flash: 180mA@5V
  Turn on the flash and adjust the brightness to the maximum: 310mA@5V
  Deep-sleep: the lowest power consumption can reach 6mA@5V
  Moderm-sleep: up to 20mA@5V
  Light-sleep: up to 6.7mA@5V
- Security: WPA/WPA2/WPA2-Enterprise/WPS
- Power supply range: 5V
- Operating temperature: -20 °C ~ 85 °C
- Sorage environment: -40 °C ~ 90 °C, < 90%RH
- Weight: 10g

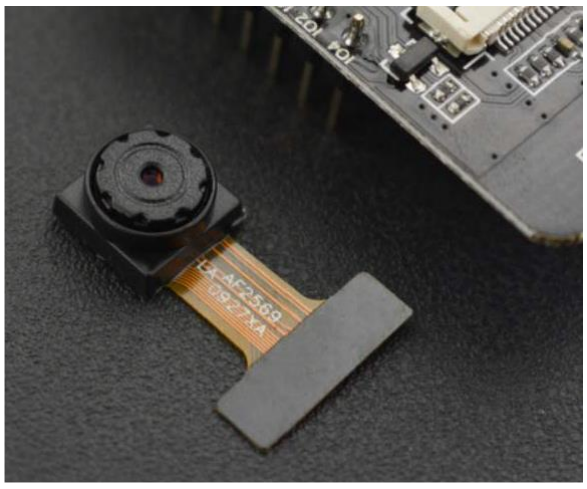
SHIPPING LIST

- ESP32-CAM Development Board x1









https://www.dfrobot.com/product-1876.html/8-15-19