LAMPIRAN - 1

Listing Program

```
#include <Adafruit_VC0706.h>
#include <SPI.h>
#include <SD.h>
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <DFPlayer_Mini_Mp3.h> //memasukan library DFPlayermini
#define min(a,b) ((a)<(b)?(a):(b))

const char* ssid = "andre";
const char* password = "112345678";

const char* host = "api.telegram.org";
const int httpsPort = 443;

const char* token = "1344956820:AAHmTPffWoTiV94JlRvDibrc1MdHofvG9MA";
const char* chat_id = "1118834794";

const char* boundry = "<delimitador_conteudo>";

WiFiClientSecure client;
```

```
UniversalTelegramBot bot(token, client);

int Bot_mtbs = 1000;
long Bot_lasttime;

#define chipSelect D0
#define doorlock 2
int state = 0;
int statusMode = 0;
SoftwareSerial cameraconnection = SoftwareSerial(0,4);
Adafruit_VC0706 cam = Adafruit_VC0706(&cameraconnection);


void setup()
{client.setInsecure();

  pinMode(D0, OUTPUT);
  pinMode(5, INPUT_PULLUP);
  pinMode(doorlock, OUTPUT);
  digitalWrite(doorlock,HIGH);
  Serial.begin(115200);
   Serial.begin(9600);
  Serial.println("VC0706 Camera snapshot test");
  mp3_set_serial (Serial); //baud komunikasi pada 9600
  delay(1);
  mp3_set_volume (90);
```

```
   if (!SD.begin(chipSelect)) {
   Serial.println("Card failed, or not present");
   return;
  }
  else {
    Serial.println("MASUK");
  }
    initWifiConnection();
  statusMode = 0;
}

void loop()
{ utama();
 if (millis() > Bot_lasttime + Bot_mtbs)  {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

    while(numNewMessages) {
      Serial.println("RESPON BAIK");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }

    Bot_lasttime = millis();
 }
}
```

```
void utama() {
   int bacaSensor = digitalRead(5);
  if(statusMode==0){
  if (bacaSensor == HIGH ) {
    Serial.println("BAHAYA! PINTU TERBUKA!");
    bot.sendMessage(chat_id,"BAHAYA! PINTU TERBUKA!");
    capture();
    delay(100);
    mp3_play (1);
  }

  else if (bacaSensor ==  LOW ) {
    Serial.println("PINTU AMAN!");
  }
  }

 if(statusMode==1){
 if (bacaSensor == HIGH ) {
    Serial.println("BAHAYA! PINTU TERBUKA!");
  }
    else if (bacaSensor ==  LOW ) {
    Serial.println("PINTU AMAN!");
  }
  }
 delay(100);
```

```
}

void capture(){
   if (cam.begin()) {
    Serial.println("Camera Found:");

  } else {
    Serial.println("KAMERA BERMASALAH!");
   // bot.sendMessage(chat_id,"KAMERA BERMASALAH");
    return;
  }

  char *reply = cam.getVersion();
  if (reply == 0) {
    Serial.print("Failed to get version");
    //disini
  } else {
    Serial.println("-----------------");
    Serial.print(reply);
    Serial.println("-----------------");
  }
  cam.setImageSize(VC0706_640x480);         // small

  uint8_t imgsize = cam.getImageSize();
  Serial.print("Image size: ");
  if (imgsize == VC0706_640x480) Serial.println("640x480");
  if (imgsize == VC0706_320x240) Serial.println("320x240");
  if (imgsize == VC0706_160x120) Serial.println("160x120");

  Serial.println("Snap in 3 secs...");
  delay(4000);

  if (! cam.takePicture())
    Serial.println("Failed to snap!");
  else
    Serial.println("Picture taken!");

  char filename[13];
  strcpy(filename, "IMAGE00.JPG");
  SD.remove(filename);
  for (int i = 0; i < 100; i++) {
    filename[5] = '0' + i/10;
    filename[6] = '0' + i%10;
     if (! SD.exists(filename)) {
      break;
    }
  }
  Serial.print("Saving ");
  Serial.println(filename);

  File imgFile = SD.open(filename, FILE_WRITE);
  uint16_t jpglen = cam.frameLength();
```

```
Serial.print("Storing ");
Serial.print(jpglen, DEC);
Serial.print(" byte image.");

int32_t time = millis();

byte wCount = 0;
while (jpglen > 0) {
  uint8_t *buffer;
  uint8_t bytesToRead = min(32, jpglen);
  buffer = cam.readPicture(bytesToRead);
  imgFile.write(buffer, bytesToRead);

if(++wCount >= 64) {
    Serial.print('.');
    wCount = 0;
  }

   jpglen -= bytesToRead;
   delay(10);
}
imgFile.close();

time = millis() - time;

Serial.println("done!");
```

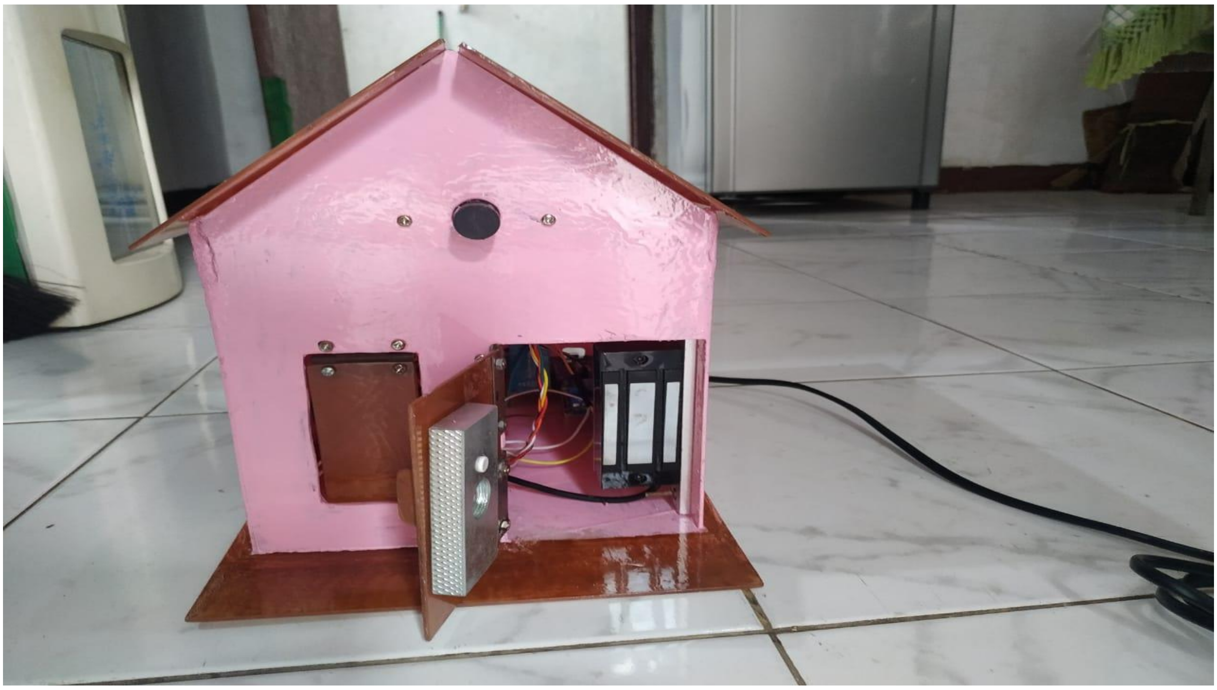
```
// Serial.print(time); Serial.println(" ms elapsed");
  sendPhotoToTelegram(filename);
}

void initWifiConnection()
{
  Serial.println();
  Serial.printf("Connecting to Wifi [%s]...\r\n", ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi Connected");
}

void receiveDataFromTelegram()
{
    // Espera por tempo de resposta do Servidor
    unsigned long timeout = millis();
    while (client.available() == 0) {
      if (millis() - timeout > 5000) {
        Serial.println("Response From Telegram!");
        client.stop();
    return;
  }
}

Serial.println();
Serial.println("Receiving from telegram...");

int responseContentLength = 0;
while (client.available()) {

  String line = client.readStringUntil('\r');
  client.read(); // lê o caracter '\n'

  Serial.println(line);

  if (line.startsWith("Content-Length:")) {
    int index = line.indexOf(':');
    responseContentLength = line.substring(index + 1).toInt();
  }

  if (line.length() == 0)
    break;
}

while (responseContentLength > 0)
{
      char ch = client.read();
      Serial.print(ch);
      responseContentLength--;
    }

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

void handleNewMessages(int numNewMessages) {
  Serial.println("handleNewMessages");
  Serial.println(String(numNewMessages));

  for (int i=0; i<numNewMessages; i++) {
    String chat_id = String(bot.messages[i].chat_id);
    String chat_id_1 = String(bot.messages[i].chat_id);
    String text = bot.messages[i].text;

     if (text == "/pantau") { //instruksi untuk memanggil foto
      capture();
    }
    if (text == "/ON"){
     Serial.println("HIDUPKAN KEAMANAN RUMAH");
     bot.sendMessage(chat_id,"KEAMANAN RUMAH AKTIP");
      statusMode = 0;
}
```

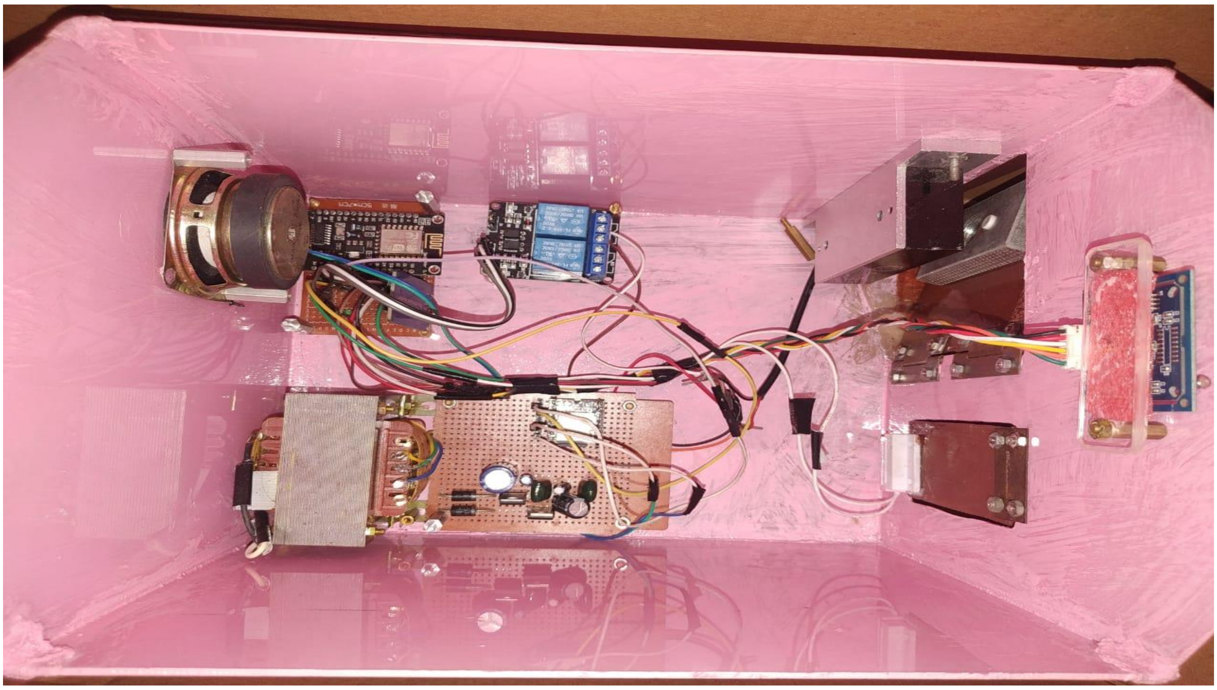
```
    if (text == "/OFF"){
    Serial.println("MATIKAN KEAMANAN RUMAH");
    bot.sendMessage(chat_id,"KEAMANAN RUMAH TIDAK AKTIF");
     statusMode = 1;
  }
  if (text == "/KunciRumah"){
    digitalWrite(doorlock,LOW);
    Serial.println("RELAY ON");
    bot.sendMessage(chat_id,"RUMAH DIKUNCI");
}
    if (text == "/MatikanKunciRumah"){
    digitalWrite(doorlock,HIGH);
    Serial.println("RELAY OFF");
    bot.sendMessage(chat_id,"RUMAH TIDAK DIKUNCI");
  }
 if (text == "/mulai") { // Text Pembuka saat Program dijalankan
   String welcome = "TABIK PUN!\n\n";
   welcome += "SISTEM KEAMANAN RUMAH DENGAN MENGGUNAKAN SENSOR MEGHNET BERBASIS INTERNET OFF THING \n\n";
   welcome += "OLEH : ANDRE EFENDI\n\n";
   welcome += "SILAHKAN PILIH PERINTAH DI BAWAH INI : \n\n";
   welcome += "/mulai    : PANDUAN\n";
   welcome += "/pantau   : MELIHAT SITUASI RUMAH\n";
   welcome += "/ON    : KEAMANAN RUMAH AKTIP\n";
   welcome += "/OFF   : KEAMANAN RUMAH TIDAK AKTIF\n\n";
   welcome += "/KunciRumah    : RUMAH DIKUNCI\n";
   welcome += "/MatikanKunciRumah   : RUMAH TIDAK DIKUNCI\n\n";
  bot.sendMessage(chat_id, welcome, "Markdown");
   }
  }
}

void sendPhotoToTelegram(String filename)
{
    Serial.printf("Connecting Telegram %s:%d... ", host, httpsPort);

    if (!client.connect(host, httpsPort)) {
      Serial.println("Failde To Conenct!");
      return;
    }
    sendDataToTelegram(filename);
    receiveDataFromTelegram();
}

void sendDataToTelegram(String file_name)
{
    String start_request = "";
    String end_request = "";

//    String chat_id = String(bot.messages[0].chat_id);
//    String chat_id_1 = String(bot.messages[0].chat_id);

    start_request = start_request + "--" + boundry + "\r\n";
    start_request = start_request + "content-disposition: form-data; name=\"chat_id\"" + "\r\n";
    start_request = start_request + "\r\n";
    start_request = start_request + chat_id +"\r\n";

    start_request = start_request + "--" + boundry + "\r\n";
    start_request = start_request + "content-disposition: form-data; name=\"photo\"; filename=\"foto.jpg\"\r\n";
    start_request = start_request + "Content-Type: image/jpeg\r\n";
    start_request = start_request + "\r\n";

    end_request = end_request + "\r\n";
    end_request = end_request + "--" + boundry + "--" + "\r\n";

//  String file_name = "IMAGE00.jpg";

    Serial.print("Sending ");
    Serial.println(file_name);

    File file = SD.open(file_name);
    int contentLength = (int)file.size() + start_request.length() + end_request.length();

    String headers = String("POST /bot") + token + "/sendPhoto HTTP/1.1\r\n";
    headers = headers + "Host: " + host + "\r\n";
    headers = headers + "User-Agent: ESP8266" + String(ESP.getChipId()) + "\r\n";
    headers = headers + "Accept: */*\r\n";
    headers = headers + "Content-Type: multipart/form-data; boundary=" + boundry + "\r\n";
    headers = headers + "Content-Length: " + contentLength + "\r\n";
    headers = headers + "\r\n";
    headers = headers + "\r\n";

    Serial.println();
    Serial.println("Mengirim data ke Telegram ...");

    Serial.print(headers);
    client.print(headers);
    client.flush();

    Serial.print(start_request);
    client.print(start_request);
    client.flush();

    Serial.println("sendFile");
    sendFile(&file);

    file.close();
    client.flush();

    Serial.print(end_request);
    client.print(end_request);
    client.flush();
}

void sendFile(Stream* stream)
```

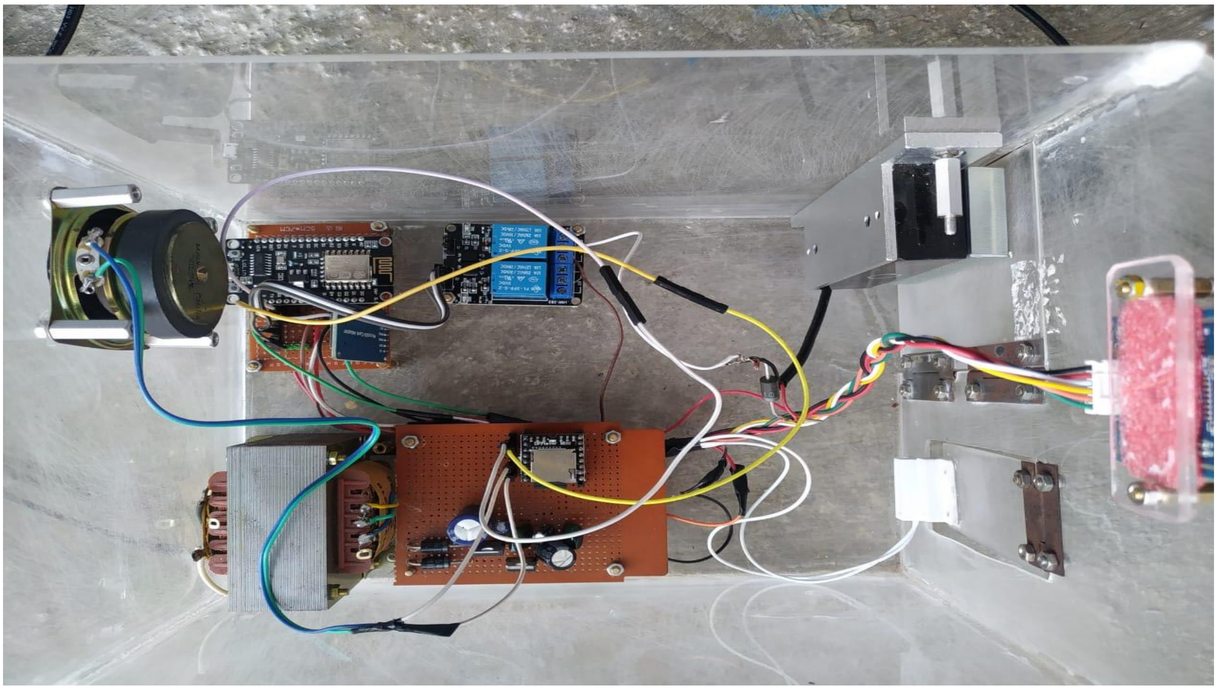
```
{
   size_t bytesReaded;
   size_t bytesSent;
   do {
      uint8_t buff[1024];
      bytesSent = 0;
      bytesReaded = stream->readBytes(buff, sizeof(buff));
      if (bytesReaded) {
         bytesSent = client.write(buff, bytesReaded);
         client.flush();
      }
   } while ( (bytesSent == bytesReaded) && (bytesSent > 0) );
}
```

LAMPIRAN - 2

Gambar Bentuk Fisik Alat







LAMPIRAN - 3

Data Alat

1. NodeMcu ESP8266

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

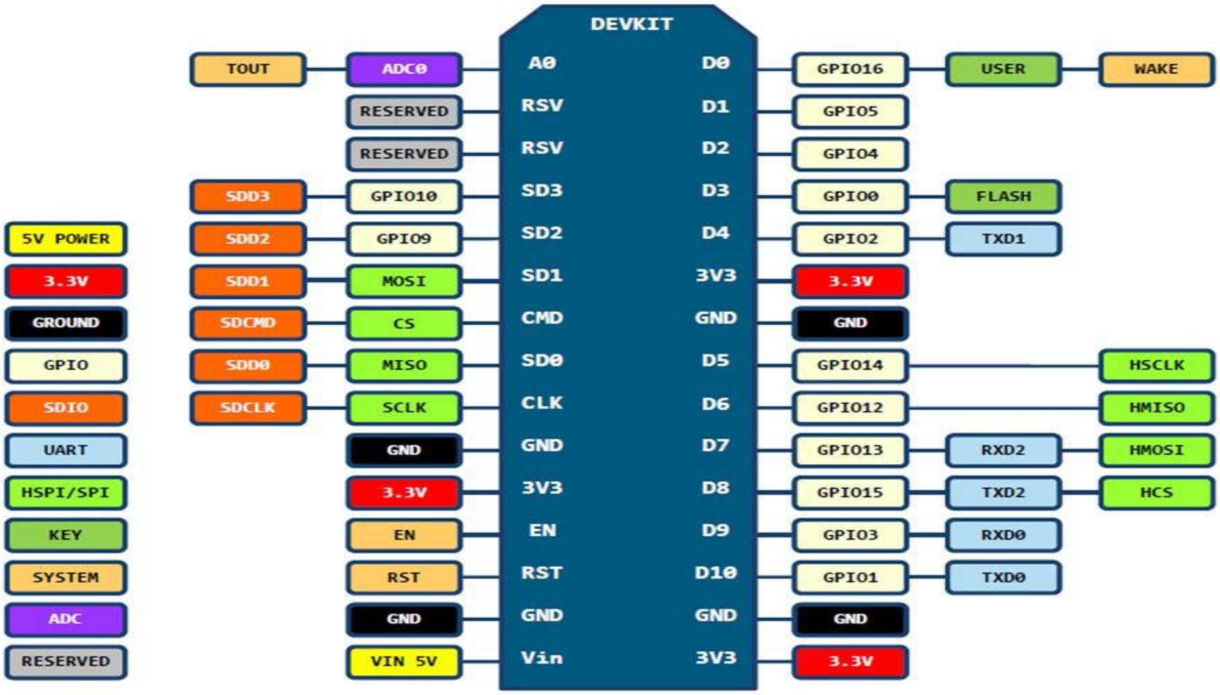
a) Specification:

- 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). mode
- +19.5dBm output power in 802.11b
- Analog to Digital: 1 input with 1024 step resolution.
- 802.11 support: b/g/n.
- Maximum concurrent TCP connections: 5.

b) Pin Definition:



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

c) 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.

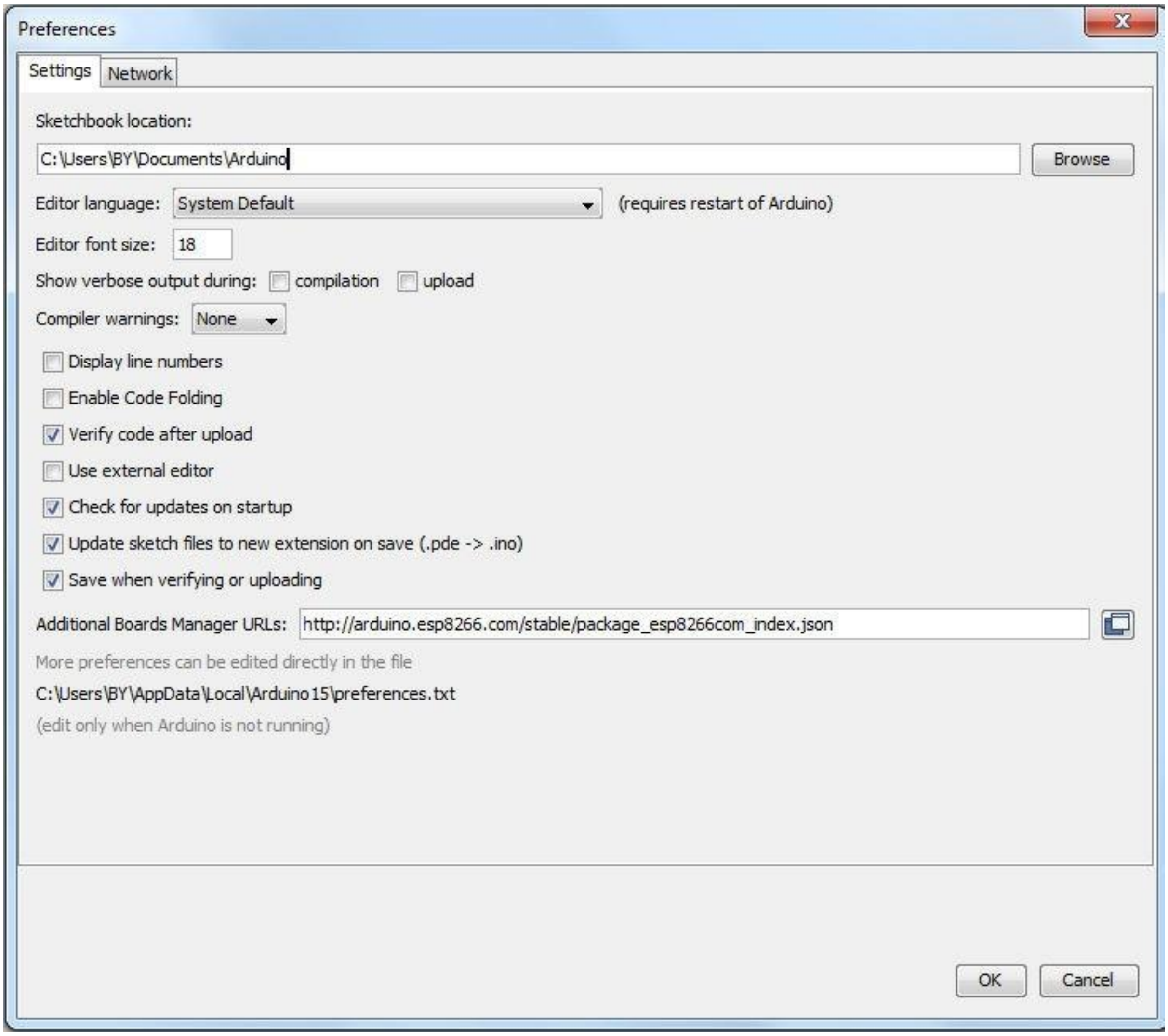
1. Install the Arduino IDE 1.6.4 or greater

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.

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.



Preferences

Settings | Network

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

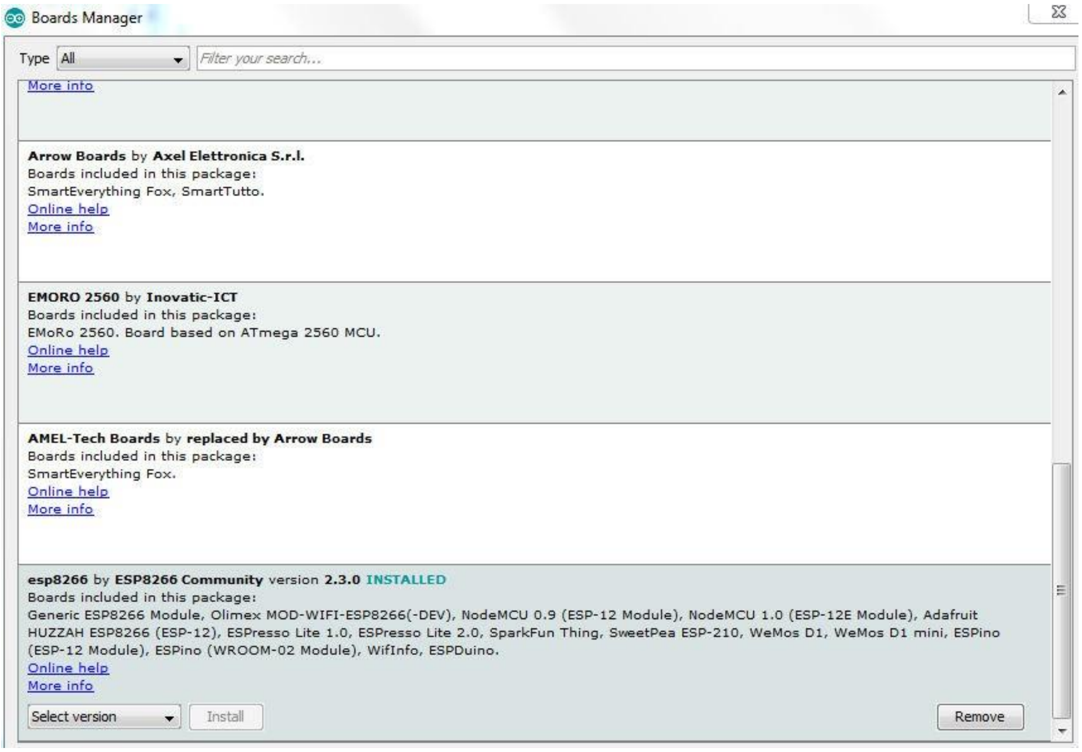
More preferences can be edited directly in the file

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.



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

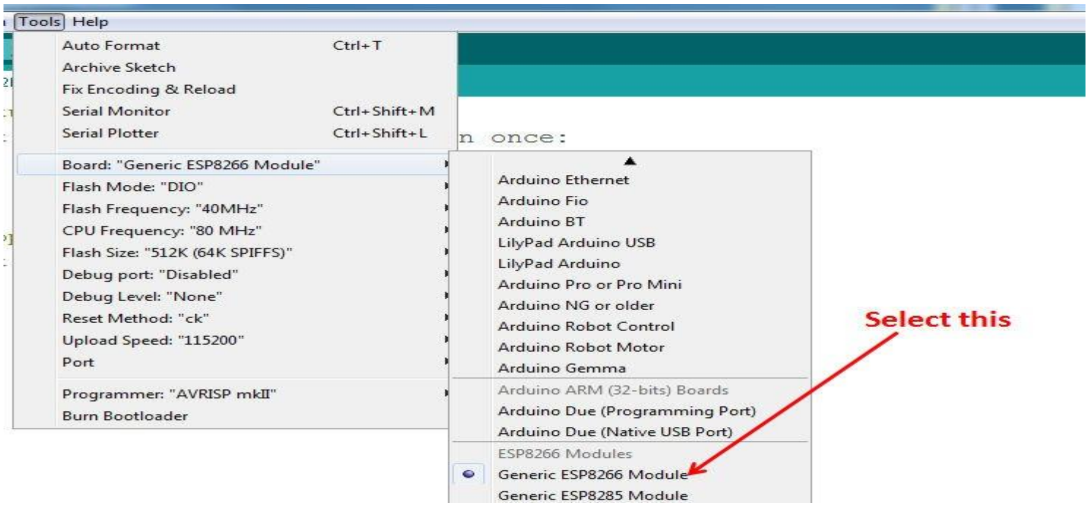
AMEL-Tech Boards by **replaced by Arrow Boards**
Boards included in this package:
SmartEverything Fox.
Online help
More info

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

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

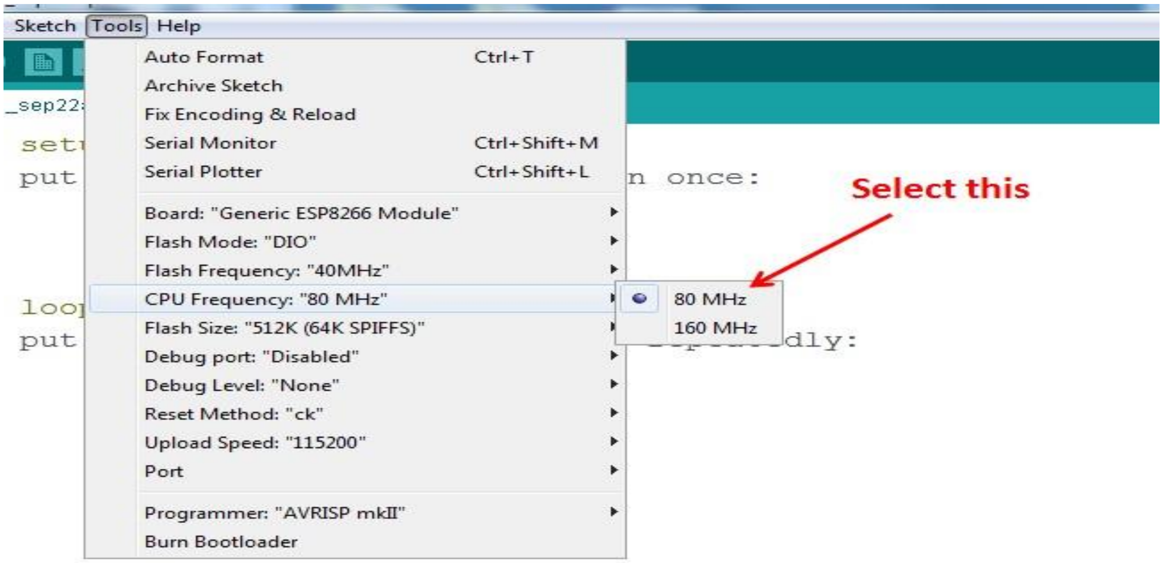
Scroll down to ‘ esp8266 by ESP8266 Community ’ and click “Install” button to install the ESP8266 library package. Once installation completed, close and re-open Arduino IDE for ESP8266 library to take effect.

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



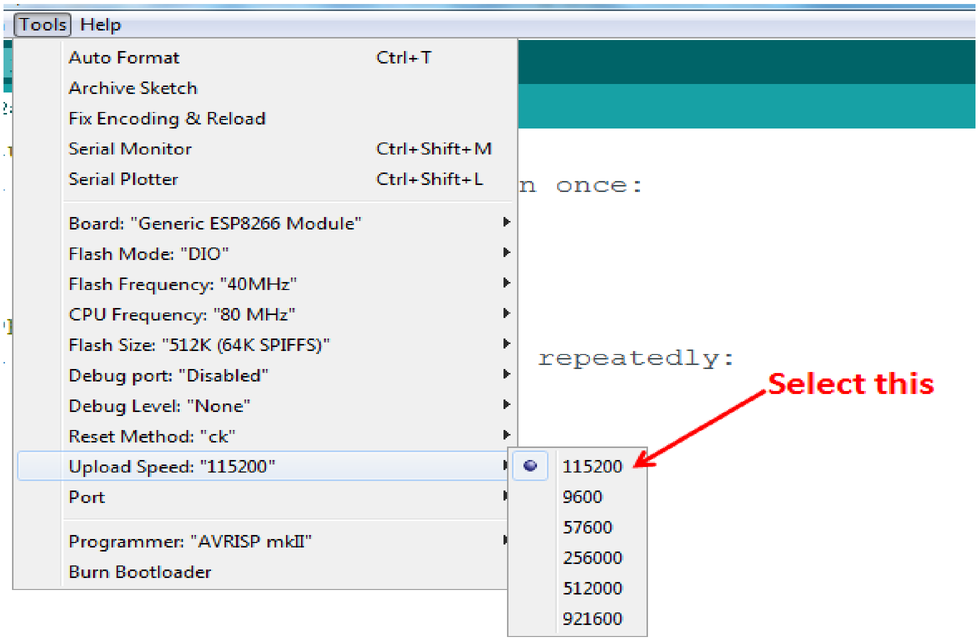
Select this

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



Select this

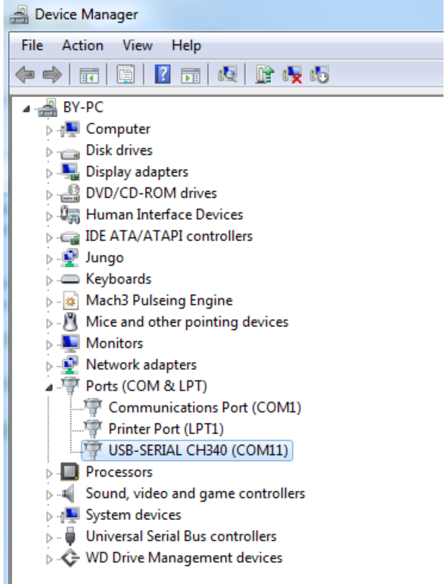
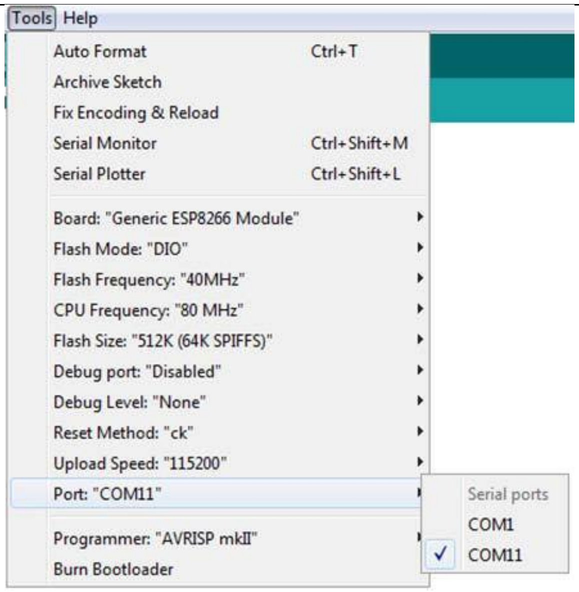
Select ‘115200’ baud upload speed is a good place to start - later on you can try higher speeds but 115200 is a good safe place to start.



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"
Programmer: "AVRISP mkII"

● 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.

	
Find out which Com Port	Select the correct Com Port as

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

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"

Serial ports
COM1
✓ COM11

4. Blink Test

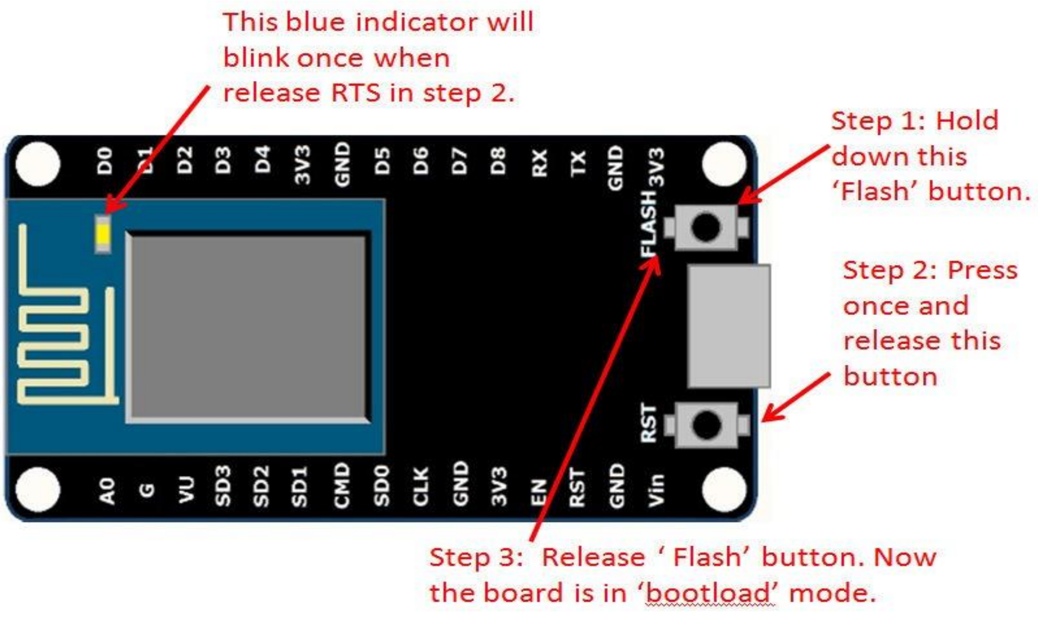
We'll begin with the simple blink test.

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

}
```

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.

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

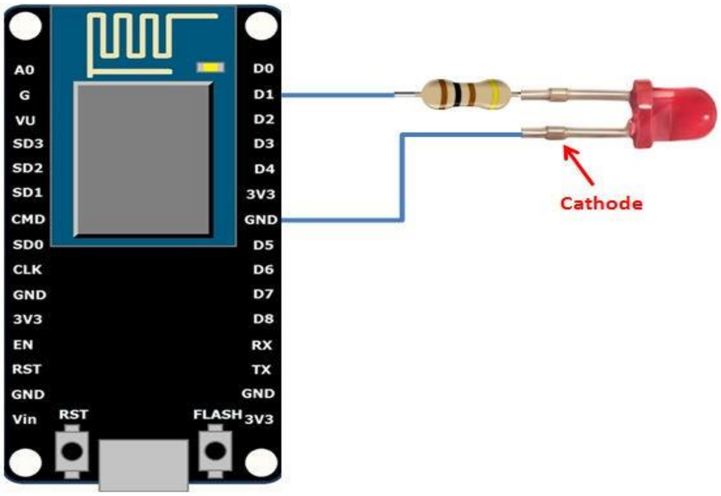
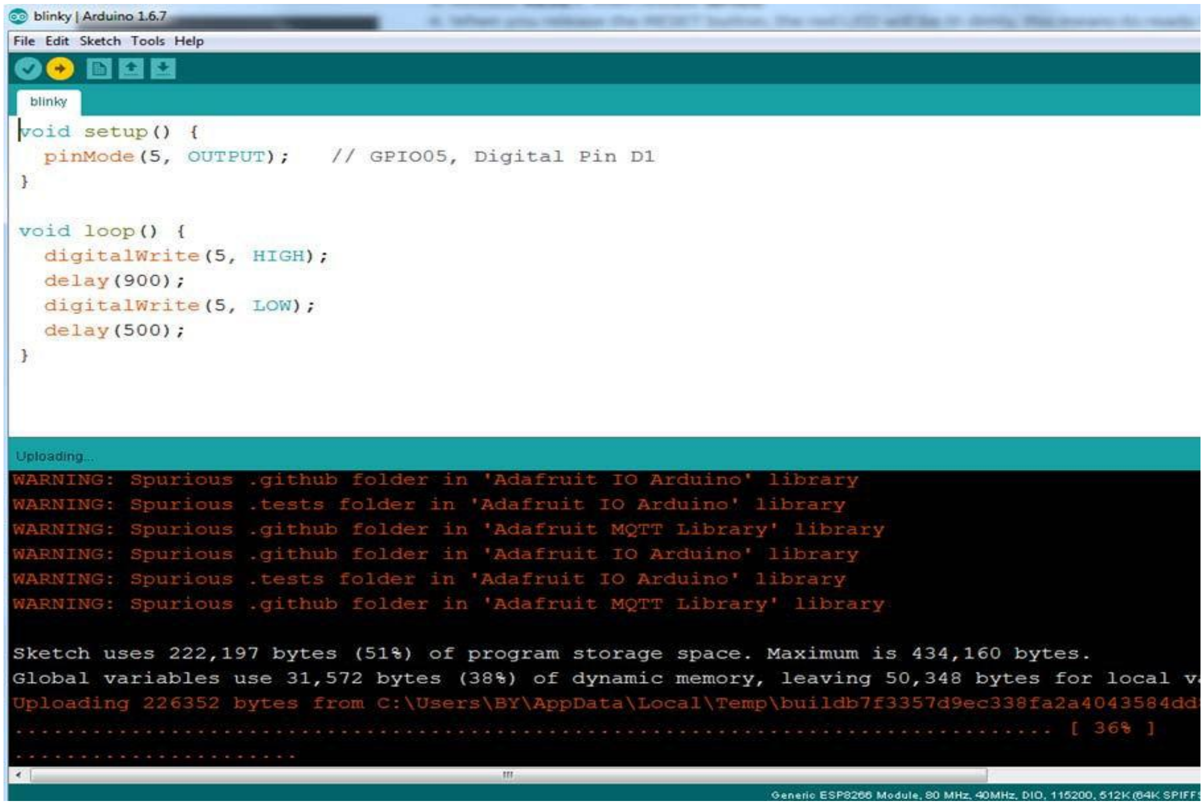


Figure3-1: Connection diagram for the blinking test


blinky | Arduino 1.6.7
File Edit Sketch Tools Help
blinky
Uploading...
Generic ESP8266 Module, 80 MHz, 40MHz, DIO, 115200, 512K (64K SPIFF

```
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% ]
......................
```

Figure 3.2: Uploading the sketch to ESP8266 NodeMCU module.

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

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


                                         // key in your own SSID
const char* password = "abc1234";
```

```
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**()){**

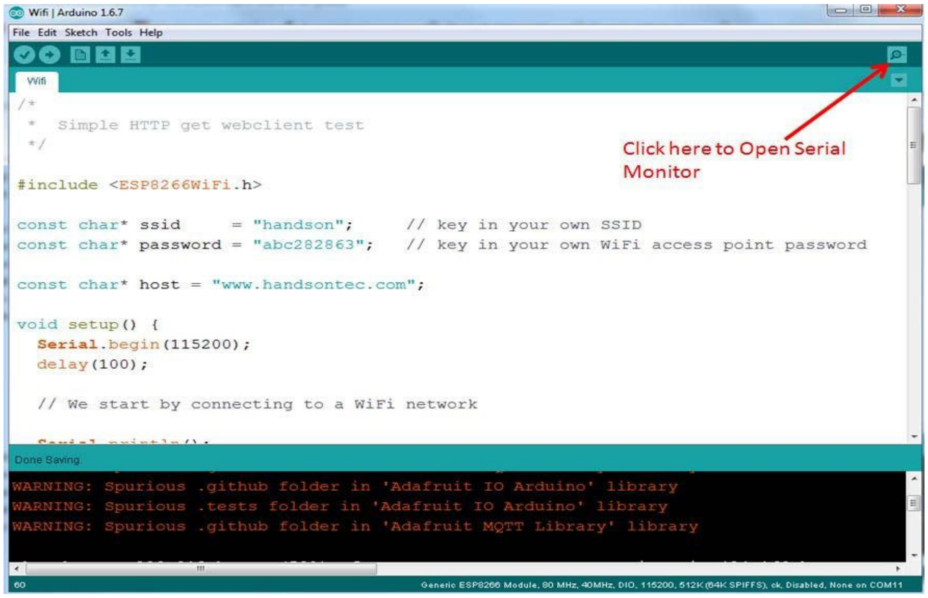
String line = client**.**readStringUntil**(**'\r'**);** Serial**.**print**(**line**);**

}

Serial**.**println**();** Serial**.**println**(**"closing connection"**);**

}

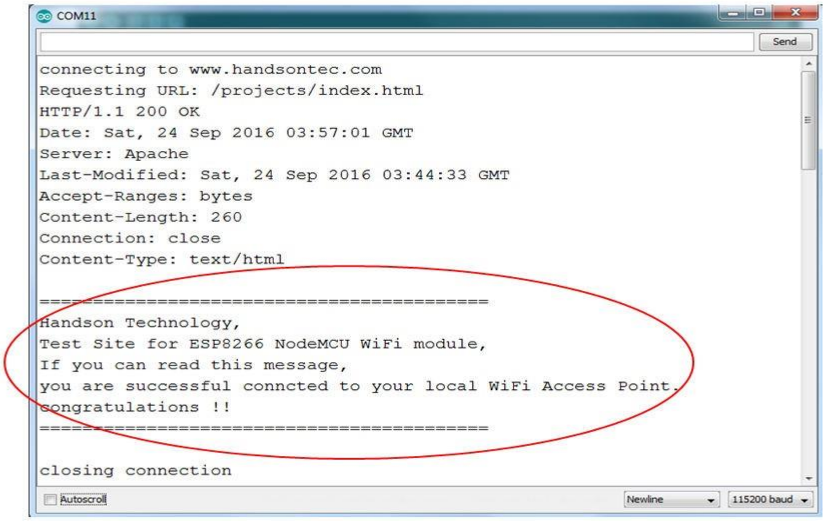
Open up the IDE serial console at 115200 baud to see the connection and webpage printout!


Wifi | Arduino 1.6.7
File Edit Sketch Tools Help
Wifi
Click here to Open Serial Monitor
Done Saving.
60

```
/*
 *  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
```

```
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
```

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



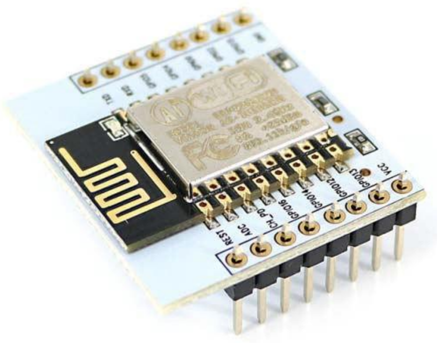
```
connecting to www.handsontec.com
Requesting URL: /projects/index.html
HTTP/1.1 200 OK
Date: Sat, 24 Sep 2016 03:57:01 GMT
Server: Apache
Last-Modified: Sat, 24 Sep 2016 03:44:33 GMT
Accept-Ranges: bytes
Content-Length: 260
Connection: close
Content-Type: text/html

==========================================
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.

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.

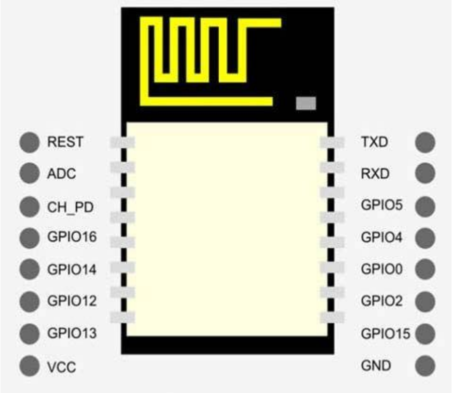


ESP8266 Module Breadboard Friendly with Header Connector

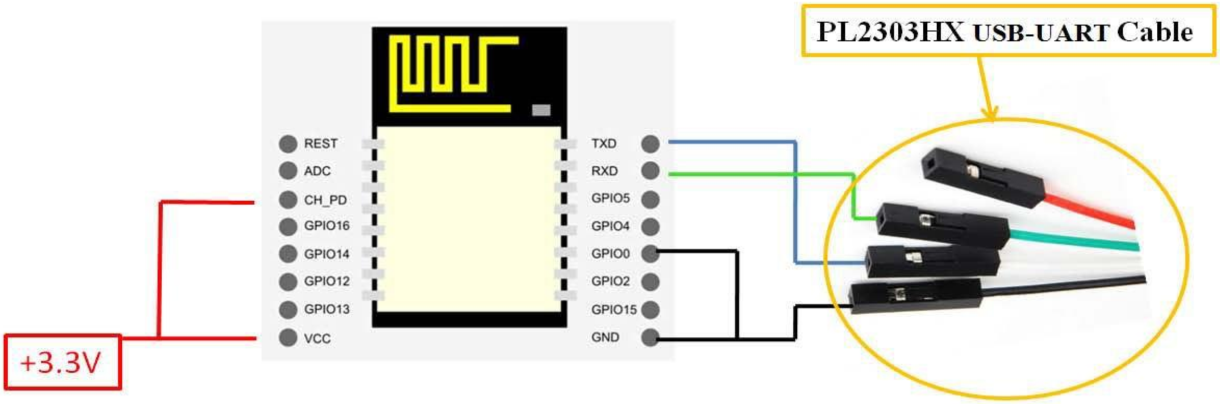
4.1 Parts Required:

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

4.2 Pin Assignment:



4.3 Wiring:



E	Description
C	Pull high, connect to Vcc +3.3V
V	Power Supply +3.3V
T	Connect to RXD (white) of PL2303HX USB-
R	Connect to TXD (Green) of PL2303HX USB-
G	Pull low, connect to GND pin
G	Power Supply ground

4.4 Downloading NodeMCU Flasher for Windows

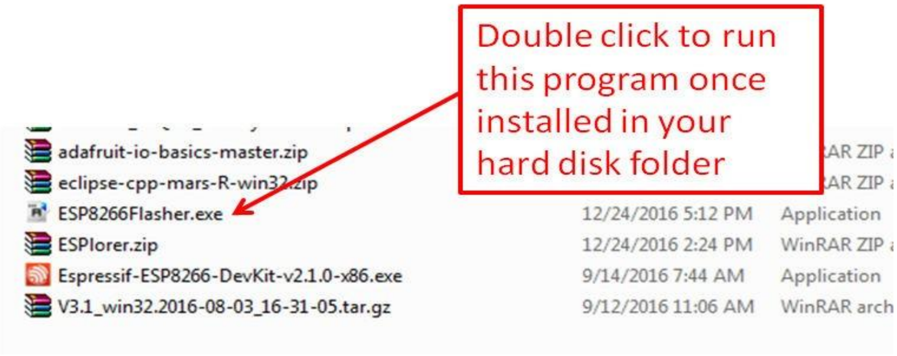
After wiring your circuit, you have to download the NodeMCU flasher. This is a .exe file that you can download using one of the following links:

- Win32 Windows Flasher
- Win64 Windows Flasher

You can find all the information about NodeMCU flasher here.

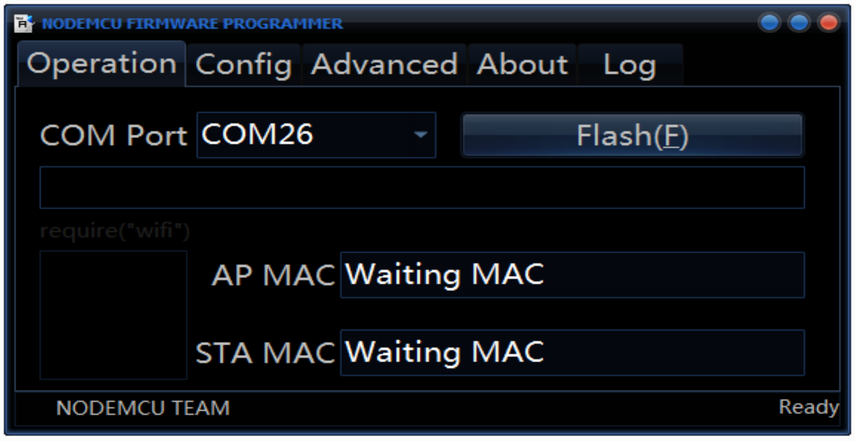
4.5 Flashing your ESP8266 using Windows

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



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

```
adafruit-io-basics-master.zip                              AR ZIP
eclipse-cpp-mars-R-win32.zip                               AR 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
```



NODEMCU FIRMWARE PROGRAMMER

COM Port: COM26

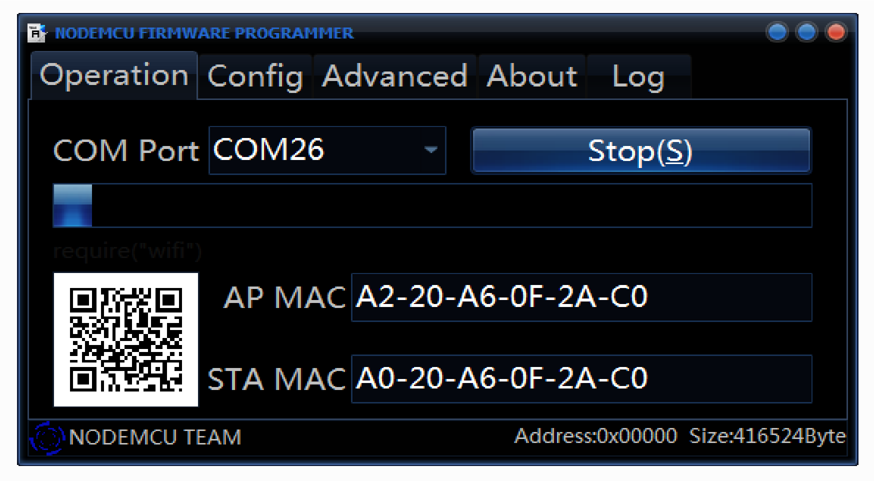
require("wifi")

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

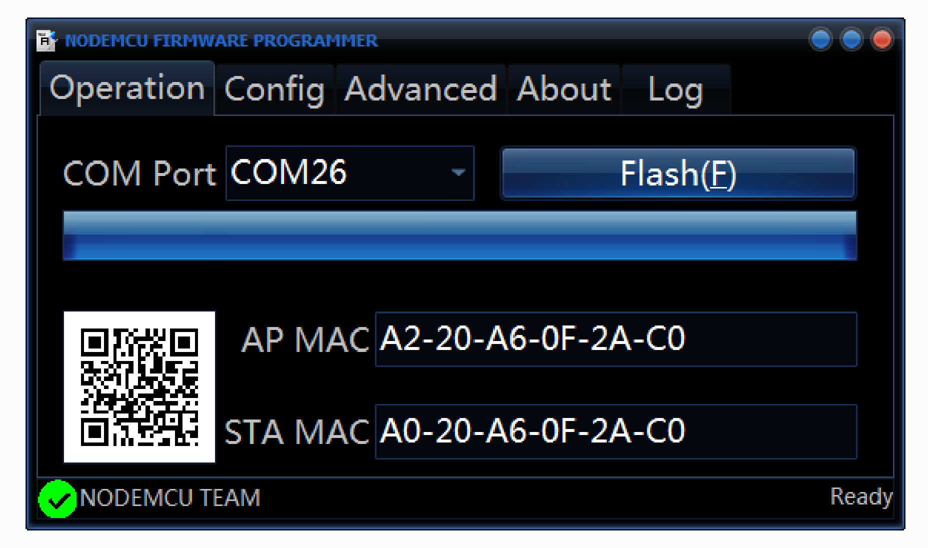
require("wifi")

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.

d) 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.

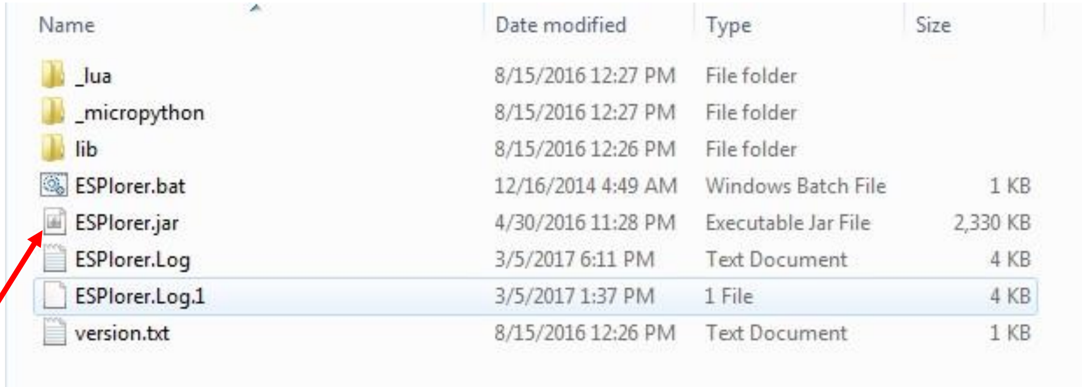
Main Resources:

- 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:

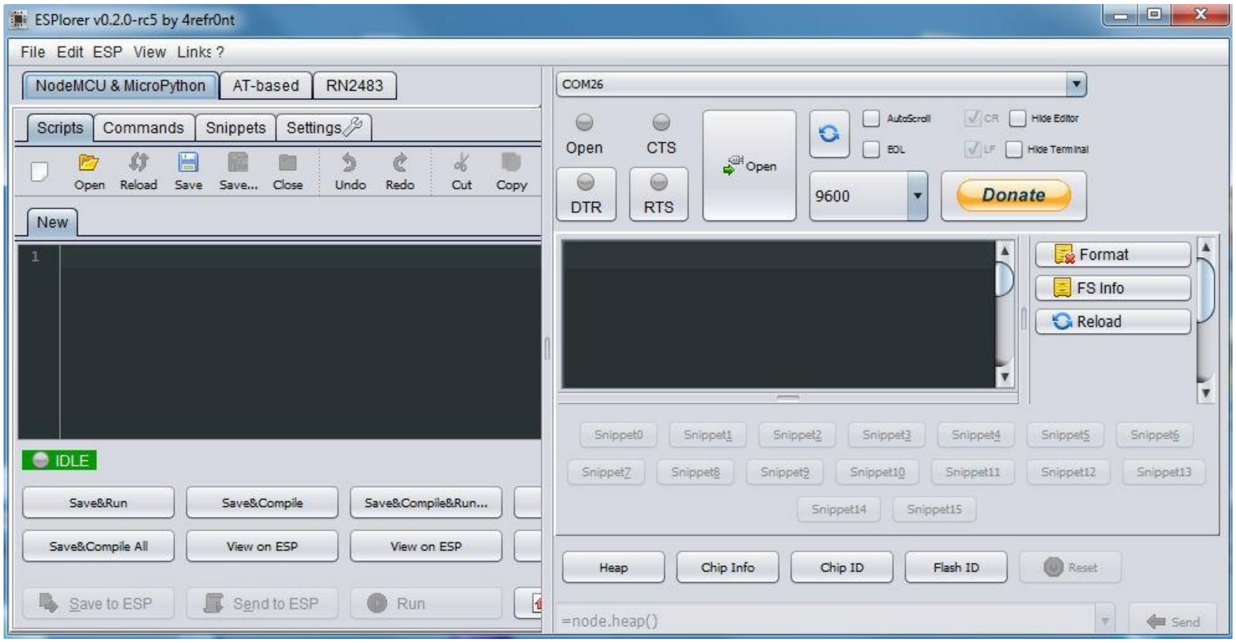


Name	Date modified	Type	Size
_lua	8/15/2016 12:27 PM	File folder	
_micropython	8/15/2016 12:27 PM	File folder	
lib	8/15/2016 12:26 PM	File folder	
ESPlorer.bat	12/16/2014 4:49 AM	Windows Batch File	1 KB
ESPlorer.jar	4/30/2016 11:28 PM	Executable Jar File	2,330 KB
ESPlorer.Log	3/5/2017 6:11 PM	Text Document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

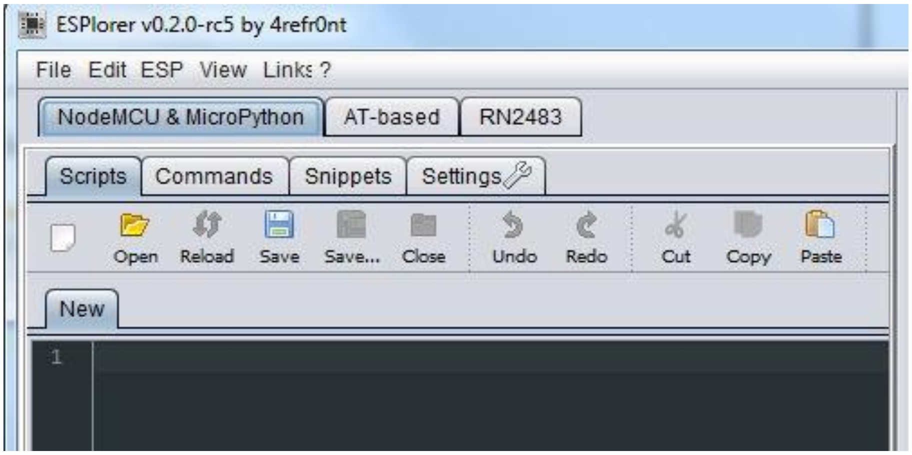


Here's a rundown of the features the ESPlorer IDE includes:

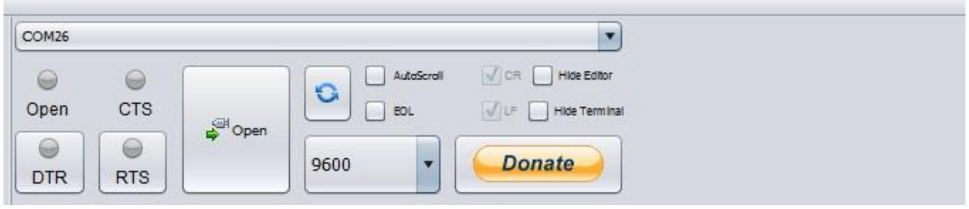
- Syntax highlighting LUA and Python code.
- Code editor color themes: default, dark, Eclipse, IDEA, Visual Studio.
- Undo/Redo editors features.
- Code Autocomplete (Ctrl+Space).
- Smart send data to ESP8266 (without dumb send with fixed line delay), check correct answer from ESP8266 after every lines.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

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

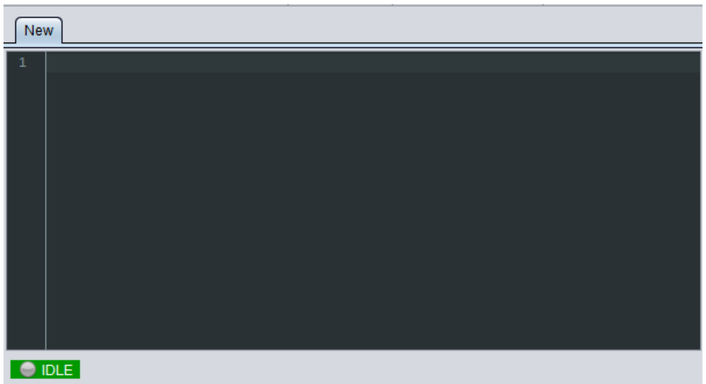
In the top left corner you can see all the regular options that you find in any software. Create a New file, Open a new file, Save file, Save file as, Undo, Redo, etc.



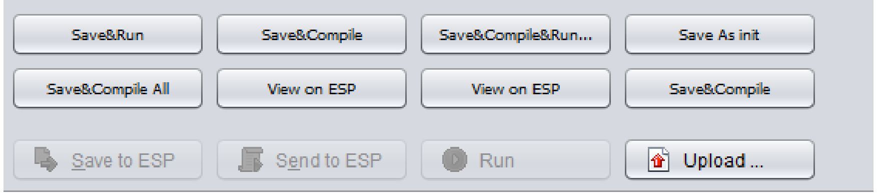
In the top right corner you have all the options you need to establish a serial communication (you're going to learn how to use them later in this Guide).



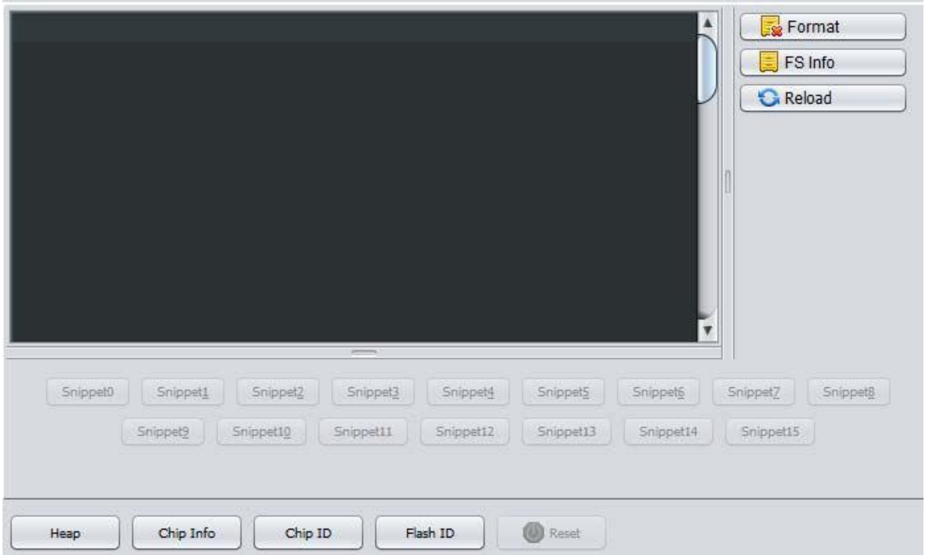
This next screenshot shows your Code Window, that's where you write your scripts (your scripts are highlighted with your code syntax).



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".

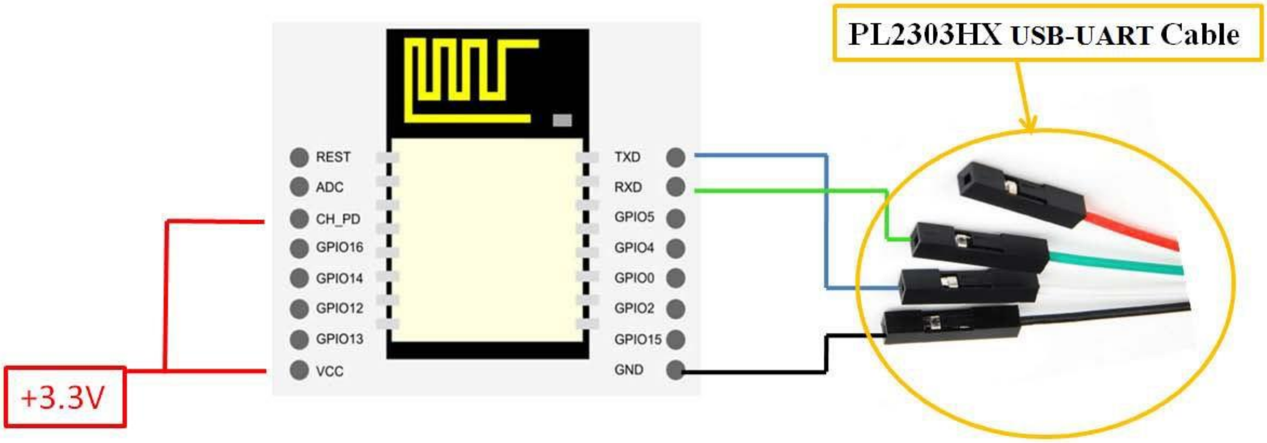


This screenshot shows the Output Window which tells you exactly what's going on in your ESP8266. You can see errors and use prints in your code to debug your projects.



5.2 Schematics

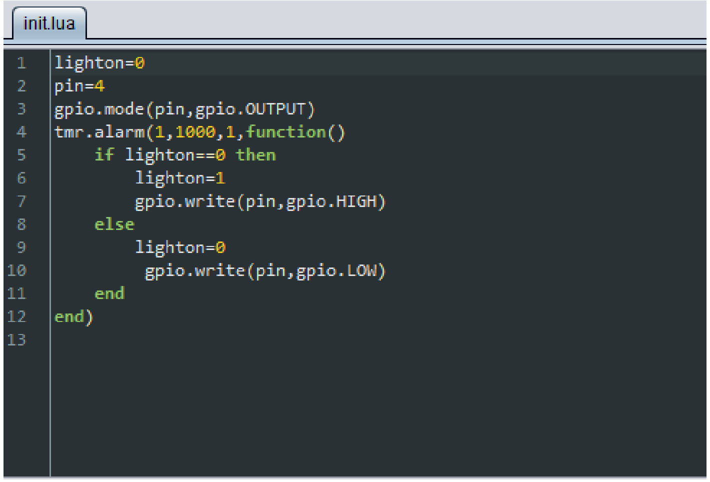
To upload code to your ESP8266, you should connect your ESP8266 to your PL2303HX USB-UART Programming Cable like the figure below:



5.3 Writing Your Lua Script

Below is your script to blink an LED.

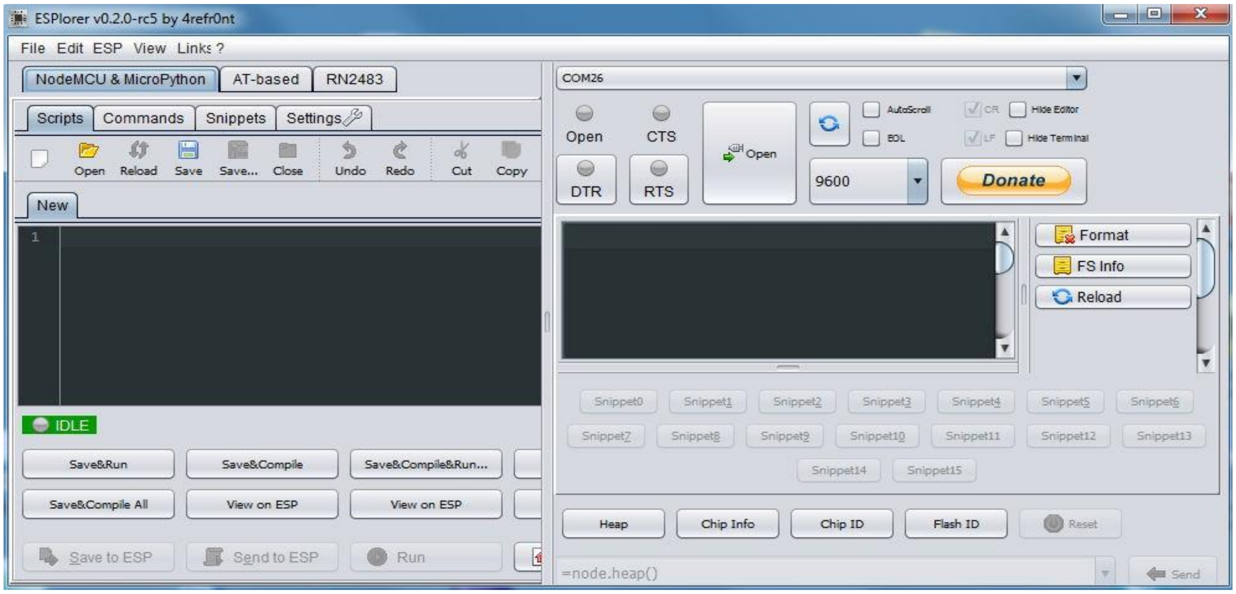
```
lighton=0
pin=4

gpio.mode(pin,gpio.OUTPUT)
tmr.alarm(1,2000,1,function()

    if   lighton==0   then
        lighton=1
```



```
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)
```

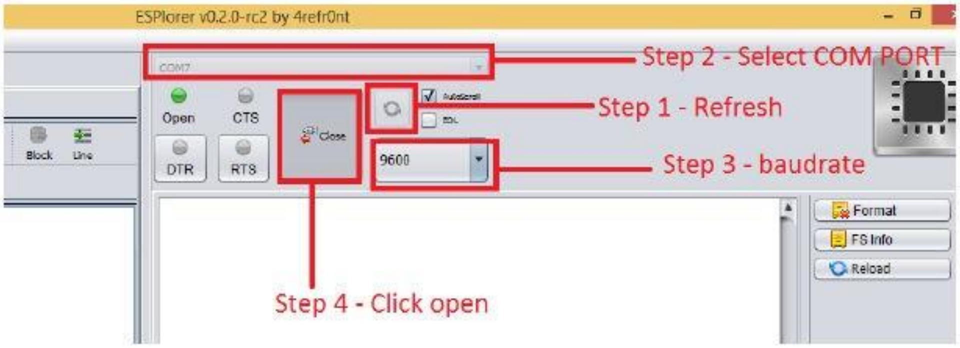
Right now you don’t need to worry how this code works, but how you can upload it to your ESP8266.

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



Step 2 - Select COM PORT

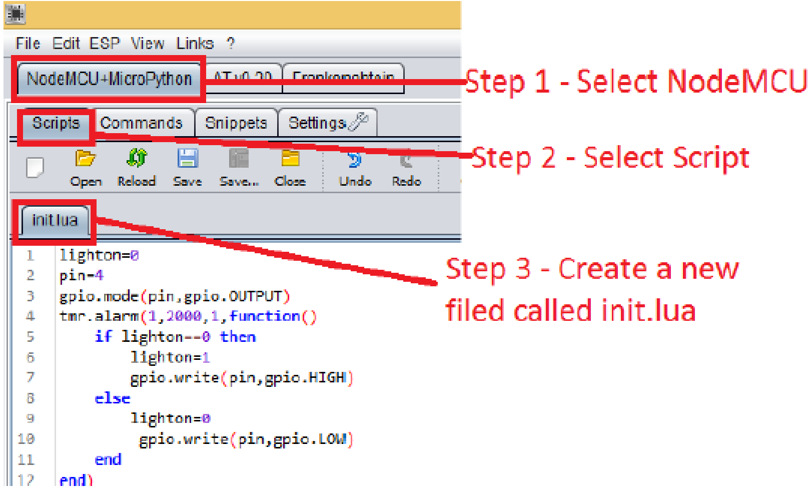
Step 1 - Refresh

Step 3 - baudrate

Step 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”



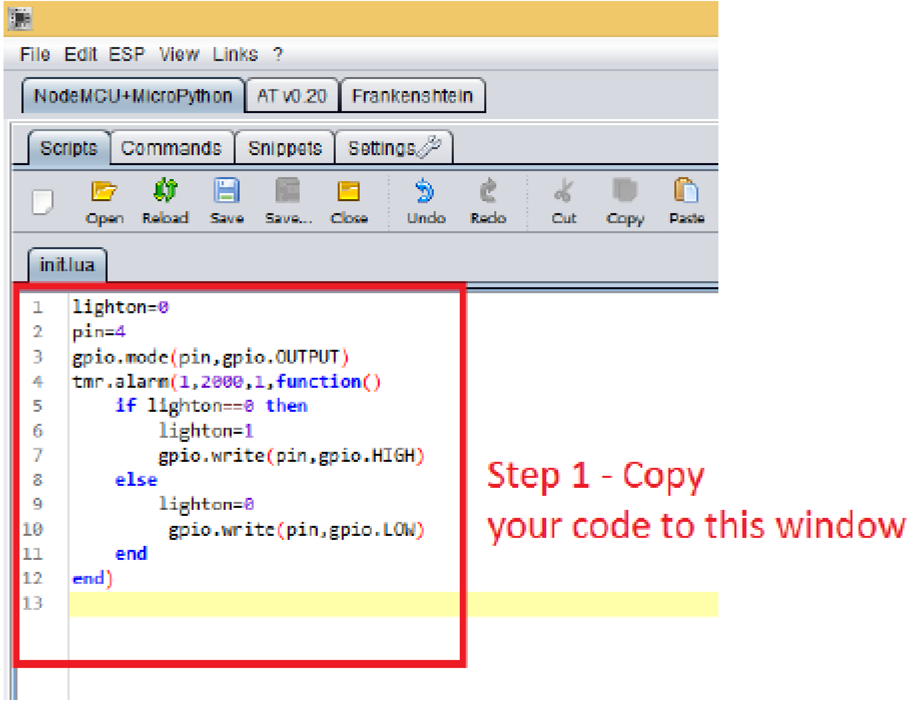
Step 1 - Select NodeMCU

Step 2 - Select Script

Step 3 - Create a new filed 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):



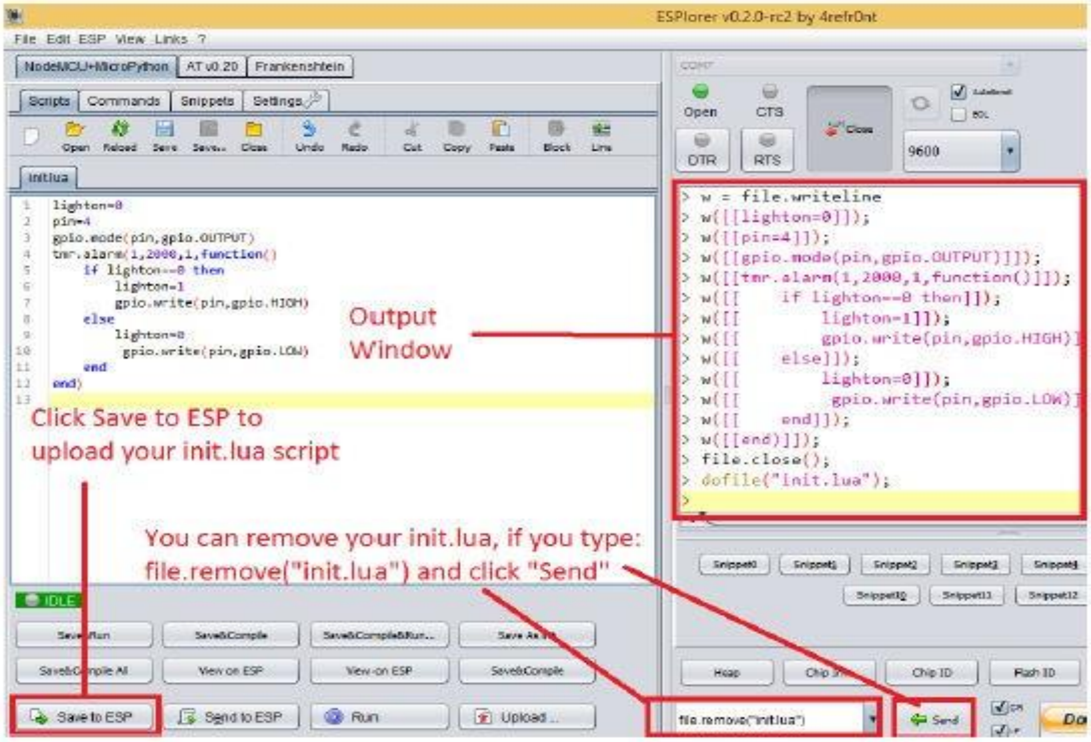
```
lighton=0
pin=4
gpio.mode(pin,gpio.OUTPUT)
tmr.alarm(1,2000,1,function()
    if lighton==0 then
        lighton=1
        gpio.write(pin,gpio.HIGH)
    else
        lighton=0
         gpio.write(pin,gpio.LOW)
    end
end)
```

Step 1 - Copy your code to this window

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");
>
```

```
file.remove("init.lua")
```

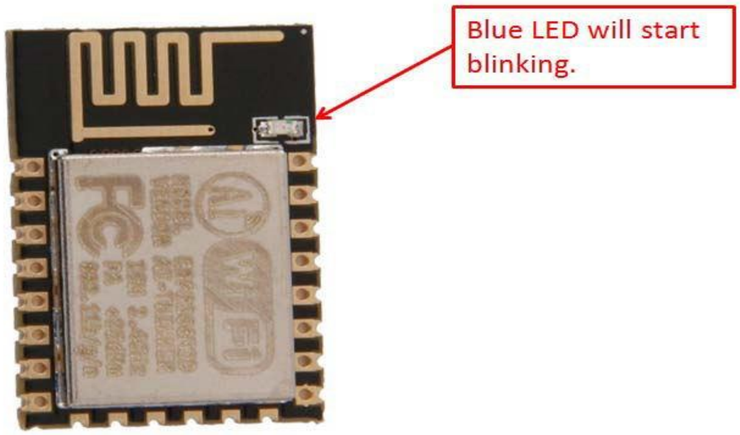
Output Window

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"

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.



Blue LED will start blinking.

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

No	**ESP8266 Pin**	**No**	**ESP8266 Pin**
D	GPIO16	D	GPIO13
D	GPIO5	D	GPIO15
D	GPIO4	D	GPIO3
D	GPIO0	D	GPIO1
D	GPIO2	D	GPIO9
D	GPIO14	D	GPIO10
D	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.

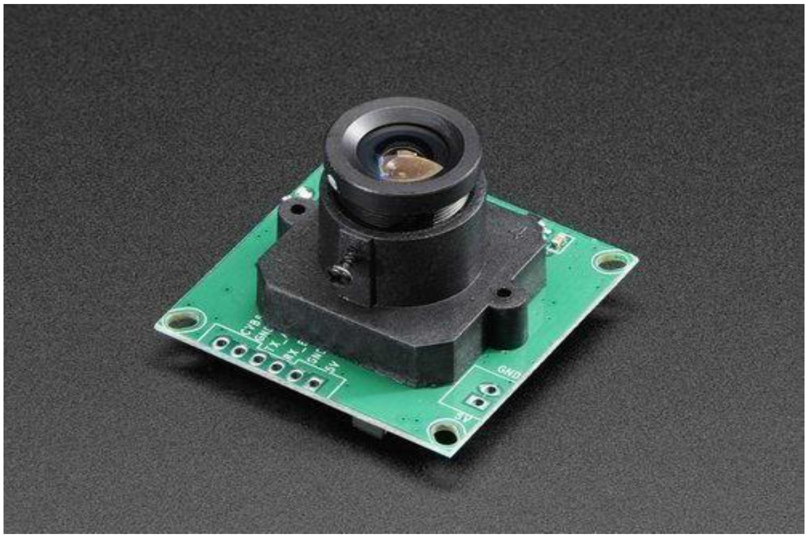
In our example in chapter 5 on blinking the blue LED, the blue LED in connected to GPIO2, it is defined as Pin4 (D4) in Lua script.

e) Web Resources:

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

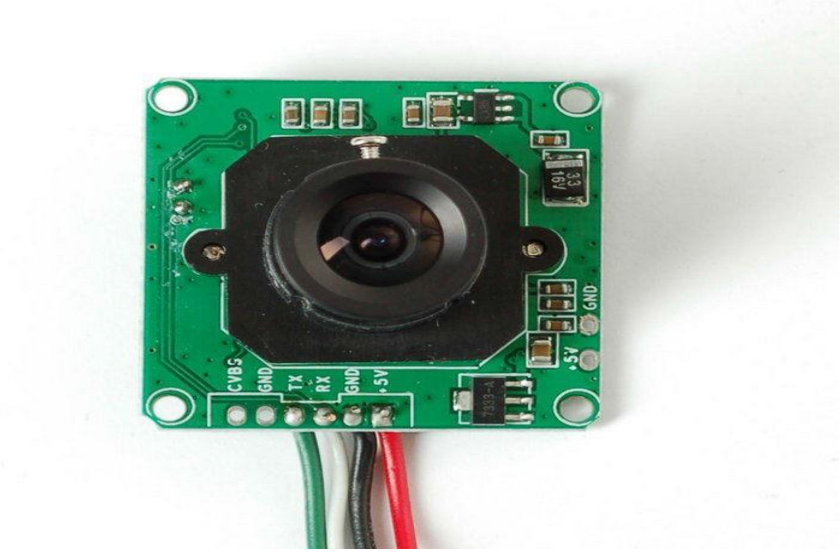
2. Kamera VC0706

TTL Serial Camera
Created by lady ada



Last updated on 2019-11-01 08:44:02 PM UTC

This tutorial is for our new TTL serial camera module with NTSC video output. These modules are a nice addition to a microcontroller project when you want to take a photo or control a video stream. The modules have a few features built in, such as the ability to change the brightness/saturation/hue of images, auto-contrast and auto-brightness adjustment, and motion detection.



Since it is a little confusing how this is both a snapshot and video camera, we'd like to explain it in detail now. The module was initially designed for surveillance purposes. Its meant to constantly stream TV-resolution video out of the Video pin (this is NTSC monochrome format) and also take commands from the serial port. The serial port commands can request that the module freeze the video and then download a JPEG color image. So for example, normally its just displaying video to a security monitor. When motion is detected, it would take a photo and save it to a disk for later analysis.

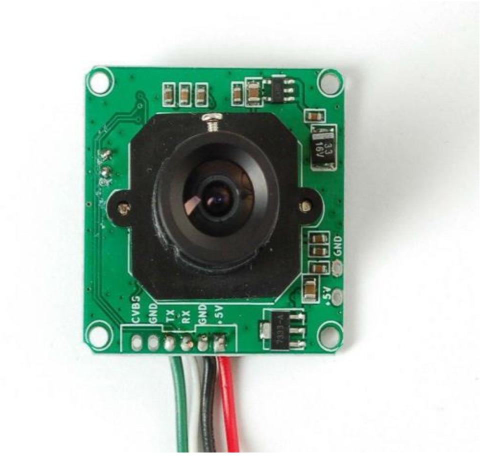
The module is admittedly not extremely high resolution - the maximum image size it can take is 640x480 pixels. And it is sensitive to infrared light, which alters the color rendition somewhat. The reason for all this is that it's meant for surveillance, not for nature photography. However, as far as we can tell, this is the best module on the market.

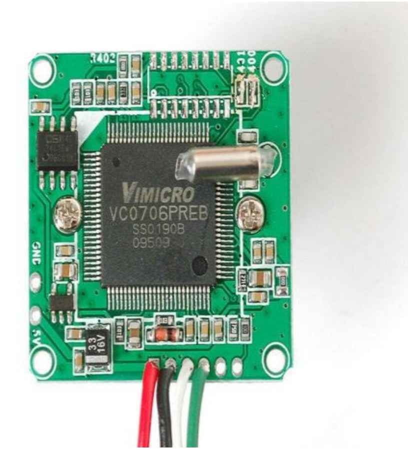
- Module size: 32mm x 32mm Image sensor: CMOS 1/4 inch CMOS Pixels: 0.3M
- Pixel size: 5.6um*5.6um
- Output format: Standard JPEG/M-JPEG
- White balance: Automatic Exposure: Automatic Gain: Automatic
- Shutter: Electronic rolling shutter
- SNR: 45DB
- Dynamic Range: 60DB
- Max analog gain: 16DB
- Frame speed: 640*480 30fps Scan mode: Progressive scan Viewing angle: 60 degrees
- Monitoring distance: 10 meters, maximum 15meters (adjustable)
- Image size: VGA (640*480), QVGA (320*240), QQVGA (160*120)
- Baud rate: Default 38400 (the datasheet claims you can change the baud rate with a command but it does not work reliably)
- Current draw: 75mA
- Operating voltage: DC +5V
- Communication: 3.3V TTL (Three wire TX, RX, GND)

Sample Images

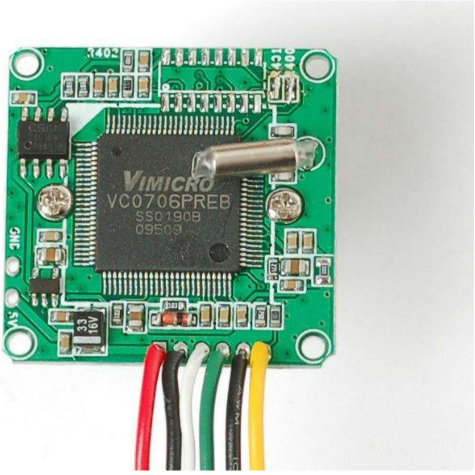
Here are two example images, one of outside during a cloudy day, and one inside on a sunny day.

The module comes without any connector so you'll need to solder wires into the connection pads. The good news is the pads are not too close togehter (about 2mm) and you can use any stranded or solid-core wire. If you aren't planning to use the video output abilities, you can use 4 wires. We will use red for the +5V pin, black for the Ground pin, white for the RX pin (data *into* the module) and green for the TX pin (data from the module)





If you'd like to get NTSC video out to connect to a TV or monitor, solder another black wire to the second Ground pin, and a yellow wire to the CVBS pin.



If you have the weatherproof version of this camera, it comes prewired with the following: Red is connected to +5Vin

- Black is connected to Ground
- Green is RX
- White is TX
- Yellow is NTSC Video signal out
- Brown is NTSC Video ground.

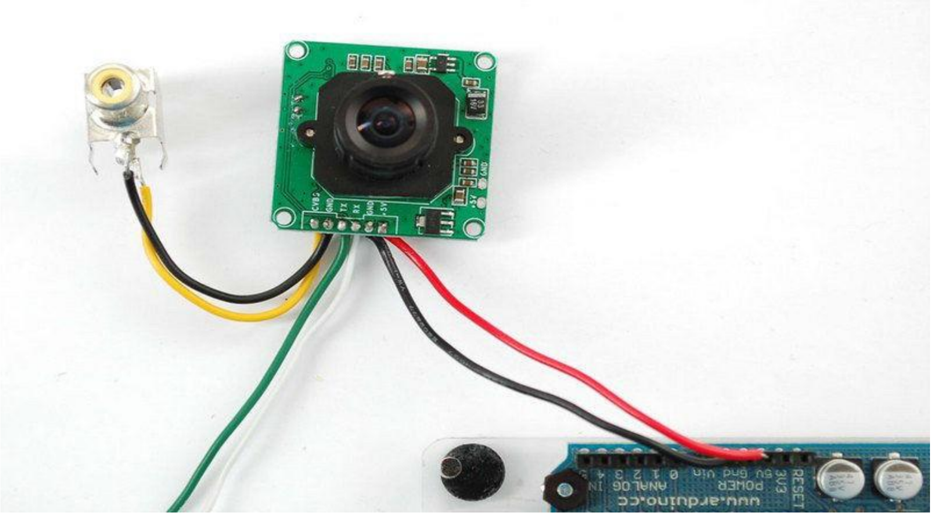
The quickest way to test out the modules is to use the NTSC video out connection. That way, when you adjust the view & focus you can immediately see the results. Paired with the next section (using the Comm Tool), its the ideal method of introducing yourself to the module.

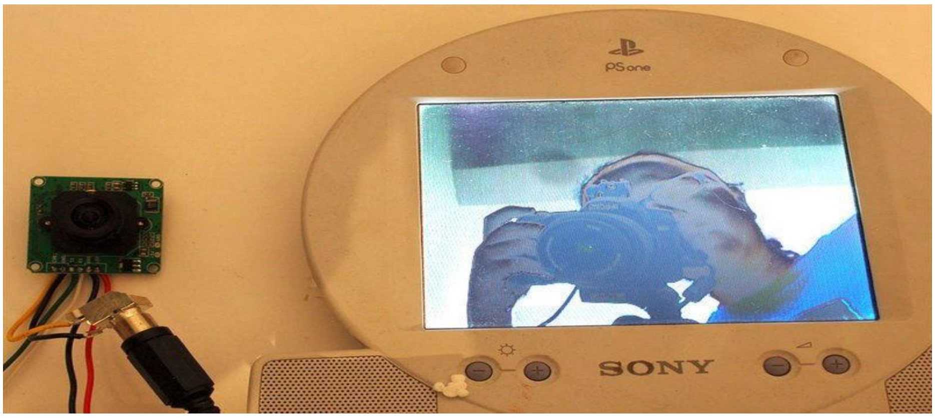
Most TV's and monitors require an RCA jack or plug input. We just soldered a spare RCA jack to the camera, with black being the case ground and yellow signal. You can get RCA cables and accessories in any hobby/electronics shop like Radio Shack.

Unfortunately, it is not possible to change the camera from NTSC to PAL - its hardcoded by a pin soldered to the board and there's no easy way to extract it and change it (we tried!)

Plug in the NTSC cable to your monitor, and connect the red and black power wires to +5V supply - you should get monochrome video output on the monitor immediately!

We have some NTSC television modules in the Adafruit shop you can use to test with (https://adafru.it/aM5)





To use the Comm Tool, a windows utility, we need to set up a serial link to the camera. There's two ways we suggest doing this. One is to use something like an FTDI friend or other USB/TTL serial converter. If you have an Arduino you can 'hijack' the serial chip (FTDI chip or similar) by uploading a blank sketch to the Arduino:

```
// empty sketch



void setup()
```

If you're using a Leonardo, Micro, Yun, or other ATmega32U4-based controller, use this Leo_passthru sketch instead of the "blank" sketch.

```
//Leo_passthru

// Allows Leonardo to pass serial data between

// fingerprint reader and Windows.

//

// Red connects to +5V

// Black connects to Ground

// Green goes to Digital 0
```

Now, wire it up as follows: Module SD Card Mini

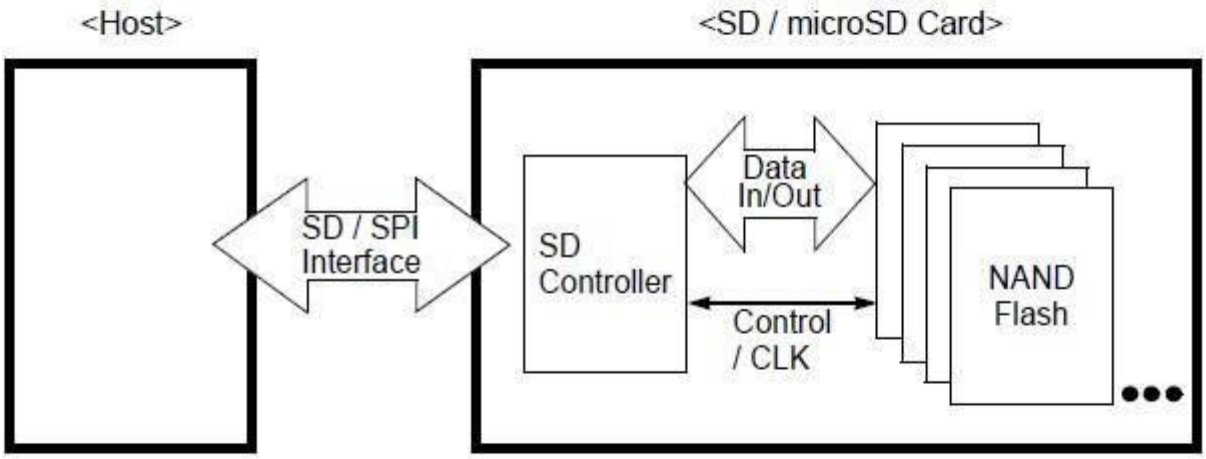
a) General Description

The eekoo The SD/microSD is a memory card that is specifically designed to meet the security, capacity, performance and enviroment requirements inherent in newly emerging audio and video consumer electronic devices. microSD cards are based on a 8-pin interface designed to operate in a maximum operating frequency of 100 MHz. The interface for microSD card products allows for easy integration into any design, regardless of which type of microprocessor is used. In addition to the interface, microSD card products offer an alternate communicationprotocol based on the SPI standard.

b) Product Features

- Up to 128GB of data storage
- High transmission speed (Class 10)
- SD - protocol compatible
- Supports SPI Mode
- Voltage range of 2.7 to 3.6V
- Correction of memory field errors
- Card removal during read operation will never harm the content
- Memory field error correction
- Dimension : 15mm(L) x 11mm(W) x 1mm(H)

c) System Block Diagram



d) Product Specifications

d.1 Reliability and Durability Specifications

Temperature	Operating: -25℃ to 85℃ Storage: -40℃(168h) to 85℃(500h)

moisture and corrosion	Operating: 25°C / 95% rel. humidity Non-Operating: 40°C / 93% rel. hum./500h salt water spray: 3% NaCl/35C; 24h acc. MIL STD Method 1009
Durability	10,000 mating cycles
Bending	10N
Torque	0.10N*m. ±2.5◦max
Drop Test	1.5m free fall
Visual Inspection/Shape and Form	No warp age; no mold slim; complete form; no cavities; surface smoothness≦-0.1mm/ cm2 within contour; no cracks; no pollution (oil,

d.2 System Reliability and Maintenance

MTBF	>1,000,000 hours
Preventive Maintenance	None
Data Reliability	< 1 non-recoverable error in 1014 bits read
Endurance	3,000~10,000 write/erase cycles

d.3 Electrical Static Discharge (ESD) requirement

ESD Protection	**Contact Discharge:** **Air Discharge;**	**±4KV, Human body model according to IEC61000-4-2.EN55024** **±8KV, Human body model according to IEC61000-4-2.EN55024**

e) Interface Description

e.1 General Description of Pins and Registers

The Micro SDHC has nine exposed contacts on one side. The host is connected to the SD Memory Card using a eight pin connector.

Pin Assignment in SD Bus Mode Pad Definition

P	Na	Ty	Micro SD
1	DA	I/	Card Detect/ Data
2	CD/	I/	Card Detect / Data
3	CM	P	Command /
4	VD	S	Supply voltage
5	CL	I	Clock
6	VSS	S	Supply Voltage
7	DA	I/	Data Line [Bit 0]
8	DA	I/	Data Line [Bit 1]

Note:

1) **S=power supply; I=input; O=output using push-pull drivers.**

2) **The extended DAT lines (DAT1-DAT3) are input on power up; they start to operate as DAT lines after the SET _BUS_WIDTH command.**

3) **After power up, this line is input with 50Kohm pull-up (can be used for card detection or SPI mode selection).**

The pull-up should be disconnected by the user, during regular data transfer, with SET_CLR_CARD_DETECT(ACMD42) command.

Pin Assignment in SPI Bus Mode Pad Definition

P	Na	Ty	Micro SD
1	RS	I	Reserved
2	CS	I	Chip Select (neg true)
3	DI	S	Data In
4	VD	S	Supply Voltage
5	SC	I	Clock
6	VSS	S	Supply Voltage
7	DO	O	Data Out
8	RS	I	Reserved

Micro SD memory Card Pin Assignment

e.2 SD Bus Topology

The SD bus has six communication lines and three supply lines:

- CMD: Command is bi-directional signal.(Host and card drivers are operating in push pull mode.)
- DAT0-3: Data lines are bi-directional signals. (Host and card drivers are operating in push

pull mode.).

- CLK: Clock is a host to cards signal. (CLK operates in push pull mode.)
- VDD: VDD is the power supply line for all cards.
- VSS [1:2]: VSS are two ground lines.

·The following figure shows the bus topology of several cards with one host in SD Bus mode.

Micro SD Memory Card System Bus Topology

During the initialization process, commands are sent to each card individually, allowing the application to detect the cards and assign logical addresses to the physical slots. Data is always sent to each card individually. However, to simplify the handling of the card stack, after initialization, all commands may be sent concurrently to all cards. Addressing information is provided in the command packet. Power Protection

Card can be inserted into or removed from the bus without damage. If one of the supply pins (VDD or Vss) is not connected properly, then the current is drawn through a data line to supply the card. Data transfer operations are protected by CRC codes; therefore, any bit changes induced by card insertion and removal can be detected by the Micro SD bus master. The inserted card must be properly reset also when CLK carries a clock frequency fpp.

If the hot insertion feature is implemented in the host, than the host has to withstand a shortcut between VDD and Vss without damage.

e.3 SPI Bus Topology

The memory Card SPI interface is compatible with SPI hosts available on the market. As any other SPI device the Micro SD Memory Card SPI channel consists of the following 4 signals:

1) CS: Host to card Chip Select signal.

2) SCLK: Host to card clock signal.

3) Data In: Host to card data signal.

4) Data Out: Card to host data signal.

Another SPI common characteristic, which is implemented in the Memory Card as well, is byte transfers. All data tokens are multiples of 8 bit bytes and always byte aligned to th9e CS signal.

The SPI standard defines the physical link only and not the complete data transfer protocol. In SPI Bus mode, the Micro SD Memory Card uses a subset of the Micro SD Memory Card protocol and command set. The Micro SD Memory Card identification and addressing algorithms are replaced by a hardware Chip Select (CS) signal.

A card (slave) is selected, for every command, by asserting (active low) the CS signal.The

CS signal must be continuously active for the duration of the SPI transaction (command, response and data). The only exception is card programming time. At this time the host can de-assert the CS signal without affecting the programming process.

The bi-directional CMD and DAT lines are replaced by uni-directional data In and data Out signals. This eliminates the ability of executing commands while data is being read or written. An exception is the multi read/write operations. The Stop Transmission command can be sent during data read. In the multi block write operation a Stop Transmission token is sent as the first byte of the data block.

f) Mechanical Form Factor

g) Contact Information

Shenzhen eekoo Electronics Co.,Ltd

Add: Room 1504,Block C,Jialin House,No.2001,Shennan Avenue,Futian District,Shenzhen China
Tel : +86-0755-8860 8670
Website: www.eekoo.com.cn eShop:eekoo.tmall.com

3. DF Player Mini

a) Summary

1.1 .Brief Instruction

DFPLayer Mini module is a serial MP3 module provides the perfect integrated MP3, WMV hardware decoding. While the software supports TF card driver, supports FAT16, FAT32 file system. Through simple serial commands to specify music playing, as well as how to play music and other functions, without the cumbersome underlying operating, easy to use, stable and reliable are the most important features of this module.

1.2 .Features

Support Mp3 and WMV decoding
Support sampling rate of
8KHz,11.025KHz,12KHz,16KHz,22.05KHz,24KHz,32KHz,44.1KHz,48KHz
24-bit DAC output, dynamic range support 90dB, SNR supports 85dB
Supports FAT16, FAT32 file system, maximum support 32GB TF card
A variety of control modes, serial mode, AD key control mode
The broadcast language spots feature, you can pause the background music being played
Built-in 3W amplifier
The audio data is sorted by folder; supports up to 100 folders, each folder can be assigned to 1000 songs
30 levels volume adjustable, 10 levels EQ adjustable.

1.3 .Application

Car navigation voice broadcast
Road transport inspectors, toll stations voice prompts
Railway station, bus safety inspection voice prompts
Electricity, communications, financial business hall voice prompts
Vehicle into and out of the channel verify that the voice prompts
The public security border control channel voice prompts
Multi-channel voice alarm or equipment operating guide voice
The electric tourist car safe driving voice notices
Electromechanical equipment failure alarm
Fire alarm voice prompts

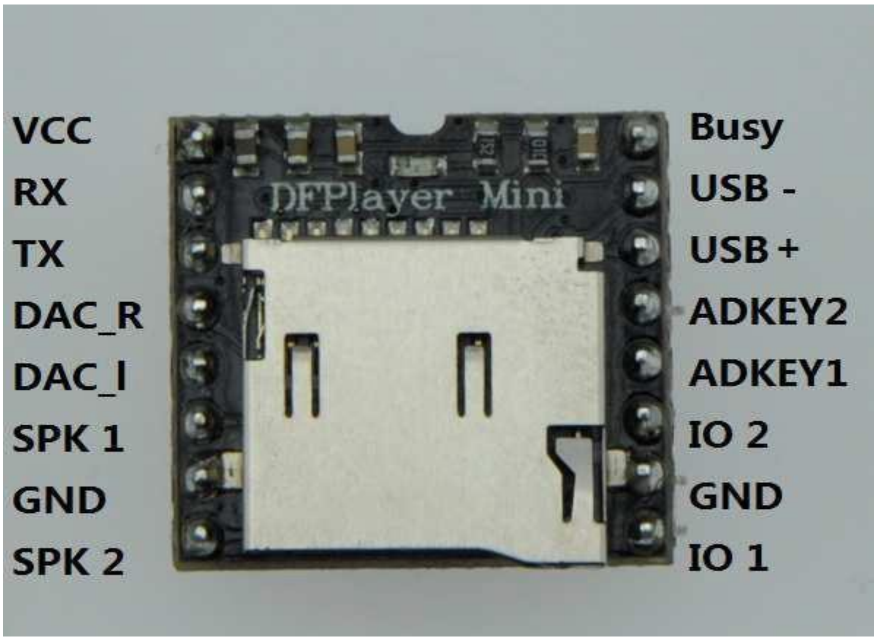
The automatic broadcast equipment, regular broadcast.

b) Module Application Instruction

b.1. Specification Description

Item	Description
MP3Format	1、Support 11172-3 and ISO13813-3 layer3 audio decoding
	2、Support sampling rate (KHZ):8/11.025/12/16/22.05/24/32/44.1/48
	3、Support Normal、Jazz、Classic、Pop、Rock etc
UART Port	Standard Serial; TTL Level; Baud rate adjustable(default baud rate is 9600)
Working Voltage	DC3.2~5.0V; Type :DC4.2V
Standby Current	20mA
Operating Temperature	-40~+70
Humidity	5% ~95%

Table 2.1 Specification Description



2.2 .Pin Descrition

Figure 2.

No	Pin	Description	Note
1	VCC	Input Voltage	DC3.2~5.0V;Type: DC4.2V
2	RX	UART serial input	
3	TX	UART serial output	
4	DAC_R	Audio output right channel	Drive earphone and amplifier
5	DAC_L	Audio output left channel	Drive earphone and amplifier
6	SPK2	Speaker-	Drive speaker less than 3W
7	GND	Ground	Power GND
8	SPK1	Speaker+	Drive speaker less than 3W
9	IO1	Trigger port 1	Short press to play previous（long press
10	GND	Ground	Power GND
11	IO2	Trigger port 2	Short press to play next（long press to increase volume）
12	ADKEY1	AD Port 1	Trigger play first segment
13	ADKEY2	AD Port 2	Trigger play fifth segment
14	USB+	USB+ DP	USB Port
15	USB-	USB- DM	USB Port
16	BUSY	Playing Status	Low means playing \High means no

Table 2.2 Pin Description

3. Serial Communication Protocol

Serial port as a common communication in the industrial control field, we conducted an industrial level of optimization, adding frame checksum, retransmission, error handling, and other measures to significantly strengthen the stability and reliability of communication, and can expansion more powerful RS485 for networking functions on this basis, serial communication baud rate can set as your own, the default baud rate is 9600

3.1. Serial Communication Format

Support for asynchronous serial communication mode via PC serial sending commands
Communication Standard:9600 bps
Data bits :1 Checkout :none Flow Control :none

Forma	V : C F p p c $	
$S	Start byte 0x7E	feedback begin with $
VER	Version	Version Information
Len	the number of bytes after	Checksums are not counted
CMD	Commands	specific operations, such as play / pause, etc.
Feedback	Command feedback	r feedback, 1: feedback, 0: no feedback
para1	Parameter 1	Query high data byte
para2	Parameter 2	Query low data byte
check	Checksum	tion and verification [not include start bit $]
$O	End bit	End bit 0xEF

For example, if we specify play NORFLASH, you need to send: 7E FF 06 09 00 00 04 FF DD EF Data length is 6, which are 6 bytes [FF 06 09 00 00 04]. Not counting the start, end, and verification.

3.2 .Serial Communication Commands

1).Directly send commands, no parameters returned

CMD	Function Description	Parameters(16 bit)
0x01	Next	
0x02	Previous	

0x03	Specify tracking(NUM)	0-2999
0x04	Increase volume	
0x05	Decrease volume	
0x06	Specify volume	0-30
0x07	Specify EQ(0/1/2/3/4/5)	Normal/Pop/Rock/Jazz/Classic/Base
0x08	Specify playback mode (0/1/2/3)	Repeat/folder repeat/single repeat/
0x09	Specify playback	U/TF/AUX/SLEEP/FLASH
0x0A	Enter into standby – low power	
0x0B	Normal working	
0x0C	Reset module	
0x0D	Playback	
0x0E	Pause	
0x0F	Specify folder to playback	1~10(need to set by user)
0x10	Volume adjust set	{DH = 1:Open volume adjust }{DL: set volume

2).Query the System Parameters

Commands	Function Description	Parameters(16 bit)
0x3C	STAY	
0x3D	STAY	
0x3E	STAY	
0x3F	Send initialization parameters	0 - 0x0F(each bit represent one device of the low-four bits)
0x40	Returns an error, request retransmission	
0x41	Reply	
0x42	Query the current status	
0x43	Query the current volume	
0x44	Query the current EQ	
0x45	Query the current playback mode	
0x46	Query the current software version	

0x47	Query the total number of TF card files	
0x48	Query the total number of U-disk files	
0x49	Query the total number of flash files	
0x4A	Keep on	
0x4B	Queries the current track of TF card	
0x4C	Queries the current track of U-Disk	
0x4D	Queries the current track of Flash	

3.3. Returned Data of Module

3.3.1. Returned Data of Module Power-on

1).The module power on, require a certain of the time initialization, this time is determined by U-disk, TF card, flash, etc. device 's file numbers, general situation in the 1.5 ~ 3Sec. If module initialization data has not been

sent out within the time, indicating that the module initialization error, please reset the module's power supply, and detect hardware connecting;

2).The module initialization data including online devices, such as sending 7E FF 06 3F 00 00 01 xx xx EF, DL

= 0x01 describe only the U-disk online during power-on, Other data are seen as the table below:

U-Disk on-line	7E FF 06 3F 00 00 01 xx xx EF	Each device are or relationship
TF Card on-line	7E FF 06 3F 00 00 02 xx xx EF	
PC on-line	7E FF 06 3F 00 00 04 xx xx EF	
FLASH on-line	7E FF 06 3F 00 00 08 xx xx EF	
U-disk & TF Card on-line	7E FF 06 3F 00 00 03 xx xx EF	

3).MCU will not send corresponding control commands until module initialization sending commands or the module will not process the commands sent by MCU, and will also affect the normal initialization of the module.

3.3.2 .Returned Data of Track Finished Playing

U-Disk finish playback 1st track	7E FF 06 3C 00 00 01 xx xx EF
U-Disk finish playback 2nd	7E FF 06 3C 00 00 02 xx xx EF
TF card finish playback 1st track	7E FF 06 3D 00 00 01 xx xx EF
rd finish playback 2nd track	7E FF 06 3D 00 00 02 xx xx EF
Flash finish playback 1st track	7E FF 06 3E 00 00 01 xx xx EF
Flash finish playback 2nd	7E FF 06 3E 00 00 02 xx xx EF

1.The module will enter into pause status automatically after being specified playing, if customers need such application, they can specify track to play ,the module will enter into pause status after finishing playing ,and wait for the commands sent by MCU.

2 In addition, we opened a dedicated I/O as decoding and pausing status indication.

See Pin 16, Busy 1).Output high level at playback status;

2).Output low level at pause status and module sleep;

3. For continuous playback applications, it can be achieved as below, if it finishes the first tracking of the TF card, it will return

7E FF 06 3D 00 00 01 xx xx EF

3D ---- U-disk command

00 01 ---- expressed finished playing tracks.

If the external MCU receives this command, please wait 100ms. And then sending the playback command [7E FF 06 0D 00 00 00 FF EE EF], because inside the module it will first initialize the next track information. In this case, the module can be played continuously.

4. If the currently finish playing the first song, the track pointer automatically point to second song, If you send a "play the next one” command, then the module will playback the third song. And, if the module finishes playing the last one, the player will automatically jump to the first pointer, and pause.

5. After specifying device, the module play pointer will point to device root directory of the first track, and enters the pause state, and wait MCU sending track

playing command.

3.3.3 .Returned Data of Module Responds

FLASH finish play the 1st track	7E FF 06 3E 00 00 01 xx xx EF

1). in order to strengthen the stability of the data communication, we have increased response processing; ACKB byte is set whether need to reply to response. So that to ensure each communication get handshake signals, which will indicate the module has been successfully received data sent by the MCU and process immediately.

2).For general applications, customers can freely choose, without this response processing is also ok.

3.3.4 .Returned Data of Module Error

Module is busy	7E FF 06 40 00 00 00 xx xx
A frame data are not all received	7E FF 06 40 00 00 01 xx xx
Verification error	7E FF 06 40 00 00 02 xx xx

1). In order to strengthen the stability of the data communication, we added data error handling mechanism. Module will responds information after receiving error data format.

2). In the case of relatively harsh environment, it is strongly recommended that customers process this command. If the application environment in general, you no need handle it;

3).The module returns busy, basically when module power-on initialization will return, because the modules need to initialize the file system.

3.3.5. Push-in and Pull-out information of Device

Push in U-disk	7E FF 06 3A 00 00 01 xx
Push in TF card	7E FF 06 3A 00 00 02 xx
Pull out U-disk	7E FF 06 3B 00 00 01 xx
Pull out TF card	7E FF 06 3B 00 00 02 xx

1).For the flexibility of the module, we particularly add command feedback of push-in and pull-out device. Let user know the working status of the module.

2).When push-in device, we default playback the first track of device root directory as audition, if users do not need this feature, you can wait 100ms after receiving the message of push –in serial device ,and then send pause command.

3.4 Serial Commands

3.4.1. Commands of Specify Track Play

Our instructions are given in support of the specified track is playing, the song selection ranges from 0 to 2999. Actually can support more, because it involves the reasons to the file system, support for the song too much, it will cause the system to operate slowly, and usually the application does not need to support so many files. If the customer has unconventional applications, please communicate with us in advance.

1).For example, select the first song played, serial transmission section: 7E FF 06 03 00 00 01 FF E6 EF 7E --- START command

FF --- Version Information

06 --- Data length (not including parity) 03 --- Representative No.

1 --- If need to acknowledge [0x01: need answering, 0x00: do not need to return the response] 00 --- Tracks high byte [DH]
2 --- Tracks low byte [DL], represented here is the first song played FF --- Checksum high byte

E6 --- Checksum low byte EF --- End Command

2).For selections, if choose the 100th song, first convert 100 to hexadecimal, the default is double-byte, it is 0x0064.

DH = 0x00; DL = 0x64

3).If you choose to play the 1000th, first convert 1000 to hexadecimal, the default is double-byte, it is 0x03E8 DH = 0x03; DL = 0xE8

4).And so on to the other operations, as in the embedded area in hexadecimal is the most convenient method of operating.

3.4.2 .Commands of Specify Volume

1). Our system power-on default volume is 30, if you want to set the volume, then directly send the corresponding commands.
2).For example, specify the volume to 15, serial port to send commands: 7E FF 06 06 00 00 0F FF D5 EF 3).DH = 0x00; DL = 0x0F, 15 is converted to hexadecimal 0x000F, can refer to the instructions of playing track section.

3.4.3 .Specify Device Play

1).The module default support four types of playback devices, the device must be on line, so it can specify playback. The software will automatically detect without user attention.
2).Refer the table as below to select the appropriate command to send
3).Module will automatically enter the Suspend state after the specified device, waiting for the user to specify a track playing. It will take about 200ms from specifying device to the module initialize file information. Please wait for 200ms and then send the specified track command.

Specify playback device	7E FF 06 09 00 00 01	xx xx：
fy playback device – TF Card	7E FF 06 09 00 00 02	
Specify playback device	7E FF 06 09 00 00 05	

3.4.4. Specify File to Play

Specify folder 01 of 001.mp3	7E FF 06 0F 00 01 01 xx xx EF
Specify folder 11 of 100.mp3	7E FF 06 0F 00 0B 64 xx xx EF
Specify folder 99 of 255.mp3	7E FF 06 0F 00 63 FF xx xx EF

1).Specify the folder playback is developed extensions, default folders are named as "01", "11" in this way because our module does not support Chinese characters identify the name of the folder name, in order to stabilize the system switching speeds and songs under each folder default maximum support up to 255 songs, up to 99 folders classification, if customers have special requirements, they need to classify according to the English name, we also can be achieved, but name only is "GUSHI", "ERGE" and other English name

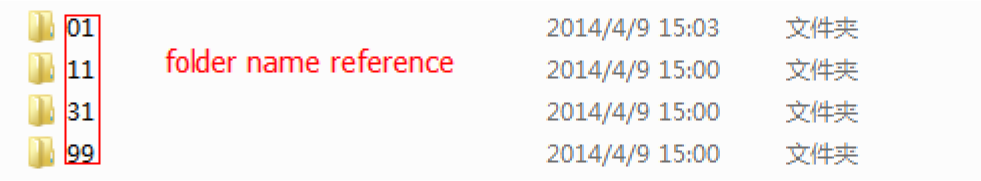
2).For example, specify "01" folder 100.MP3 file, serial port to send commands : 7E FF 06 0F 00 01 64 xx xx EF

DH: represents the name of the folder, the default support for 99 documents become 01 - 99 named DL: on behalf of the tracks, the default maximum of 255 songs that 0x01 ~ 0xFF

Please refer to the above set rules for setting tracks

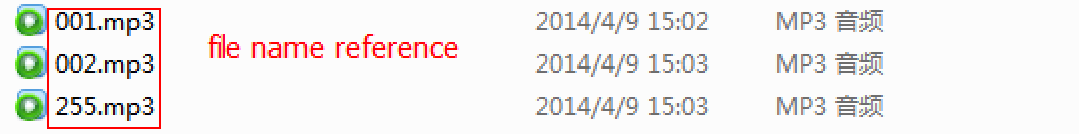
3).to the standard of the module, you must specify both the folder and file name, to lock a file. Individually specified folder or specify the file name alone is also possible, but the document management will be worse.

4).The following diagram illustrates both the folders and file names are specified



01	2014/4/9 15:03	文件夹
11	2014/4/9 15:00	文件夹
31	2014/4/9 15:00	文件夹
99	2014/4/9 15:00	文件夹

Figure 3.1folder name



001.mp3	2014/4/9 15:02	MP3 音频
002.mp3	2014/4/9 15:03	MP3 音频
255.mp3	2014/4/9 15:03	MP3 音频

Figure 3.2 file name

3.5. Key Ports

We use the AD module keys, instead of the traditional method of matrix keyboard connection, it is to take advantage of increasingly powerful MCU AD functionality, Our module default configuration 2 AD port, 20 key resistance distribution, if used in strong electromagnetic interference or strong inductive, capacitive load of the occasion, please refer to our "Notes."

1).Refer diagram

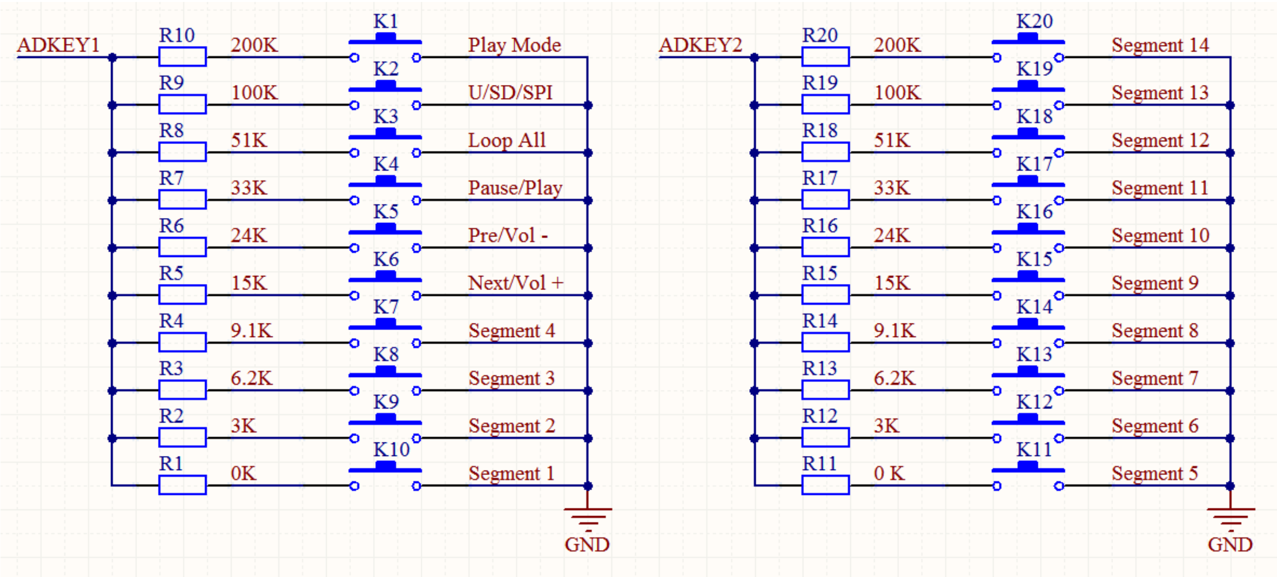


Figure 3.3 ad key refer

2)、20 function keys allocati

Key	Short Push	Long Push	Description
K1	Play Mode		Switch to interrupt / non interrupted
K2	Playback device switches		U/TF/SPI/Sleep
K3	Operating Mode		All cycle
K4	Play/Pause		
K5	Previous	Vol+	
K6	Next	Vol-	
K7	4	Repeat play tracking 4	Long push always to repeat play
K8	3	Repeat play tracking 3	Long push always to repeat play
K9	2	Repeat play tracking 2	Long push always to repeat play
K10	1	Repeat play tracking 1	Long push always to repeat play
K11	5	Repeat play tracking 5	Long push always to repeat play
K12	6	Repeat play tracking 6	Long push always to repeat play
K13	7	Repeat play tracking 7	Long push always to repeat play
K14	8	Repeat play tracking 8	Long push always to repeat play
K15	9	Repeat play tracking 9	Long push always to repeat play
K16	10	Repeat play tracking 10	Long push always to repeat play
K17	11	Repeat play tracking 11	Long push always to repeat play
K18	12	Repeat play tracking 12	Long push always to repeat play
K19	13	Repeat play tracking 13	Long push always to repeat play
K20	14	Repeat play tracking 14	Long push always to repeat play

4、Application Circuit

4.1 Serial Communication Connect

Module's serial port is 3.3V TTL level, so the default interface level is 3.3V. If the MCU system is 5V. It is recommended connect a 1K resistor in series.

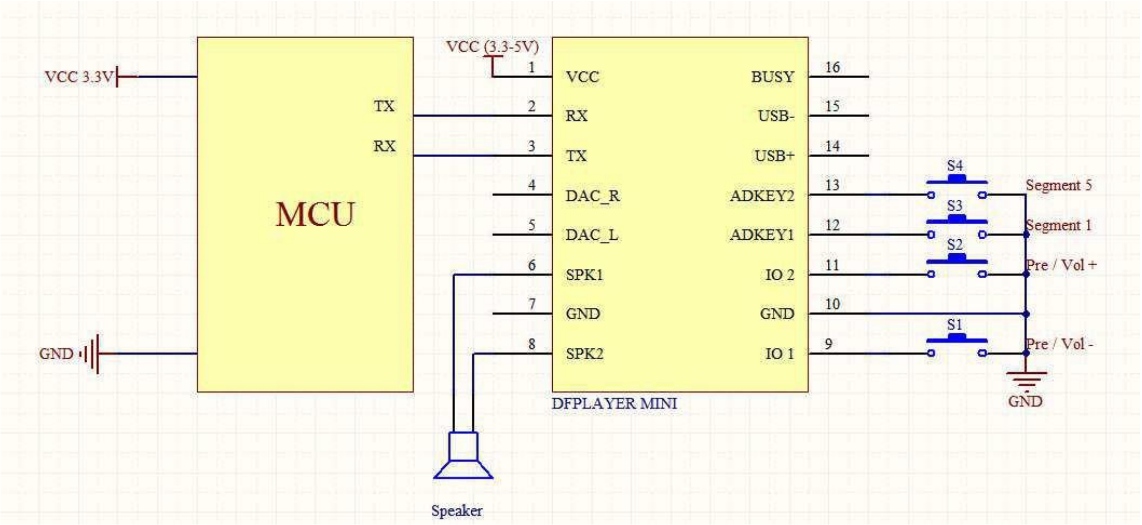


Figure 4.1 Serial Connect (3.3V)

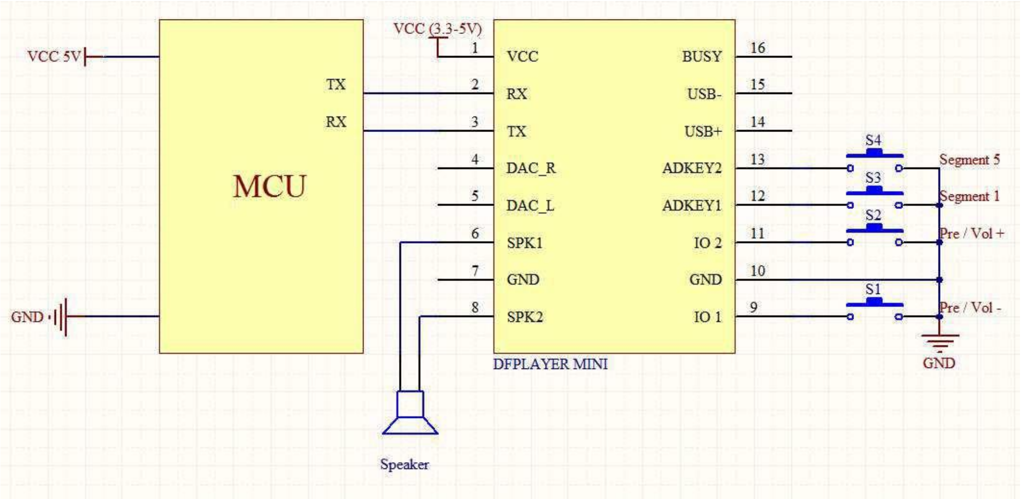


Figure 4.2 Serial Connect (5v)

4.2. Other Refer Diagram

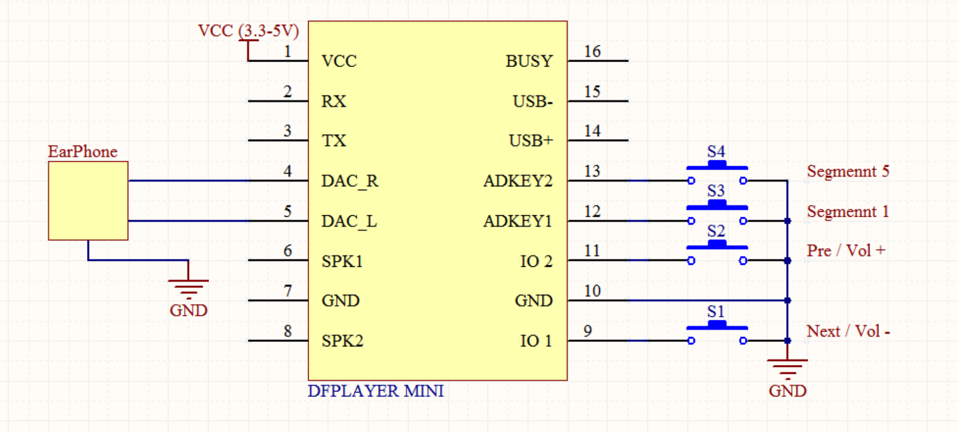


Figure 4.3 headset connect module

Between the headset and the module can string a 100R resistor, make a limiting

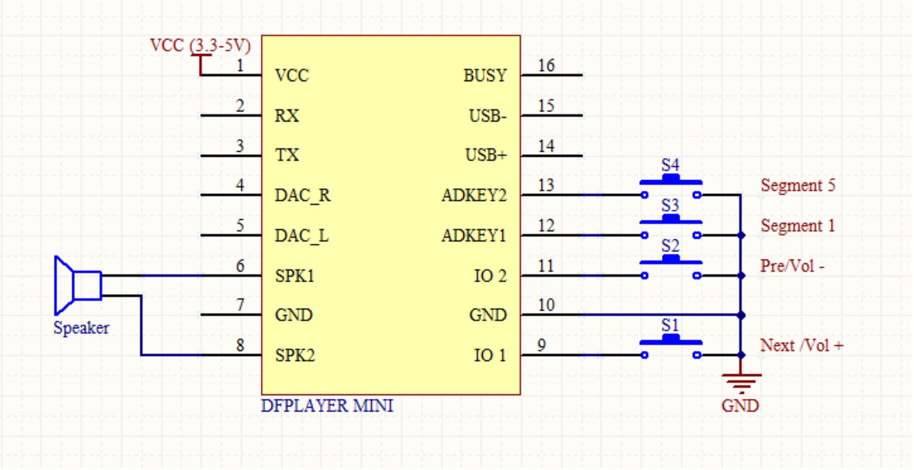


Figure 4.4 speaker connect module

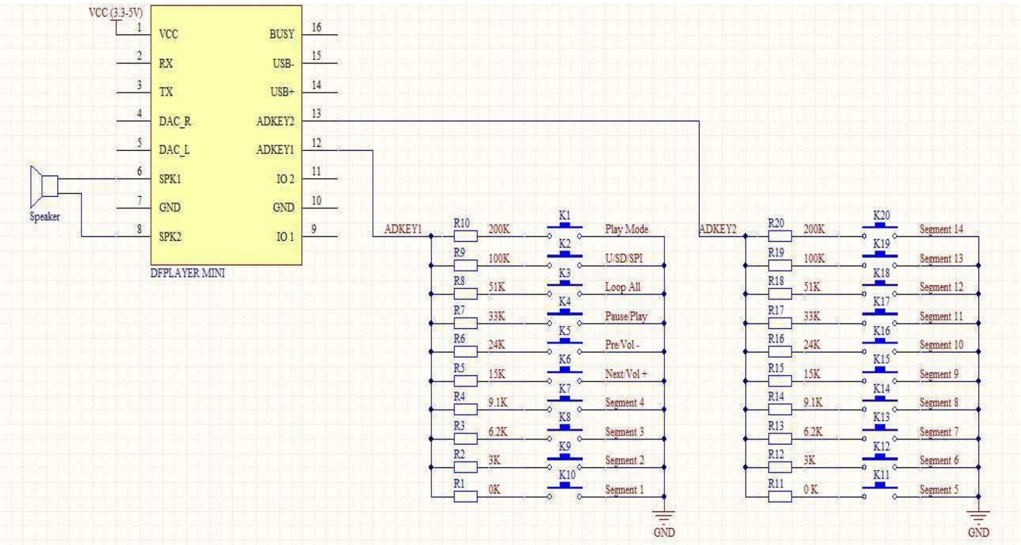


Figure 4.5 Ad key connect refer

5、 MP3-TF-16P Size (unit: mm)

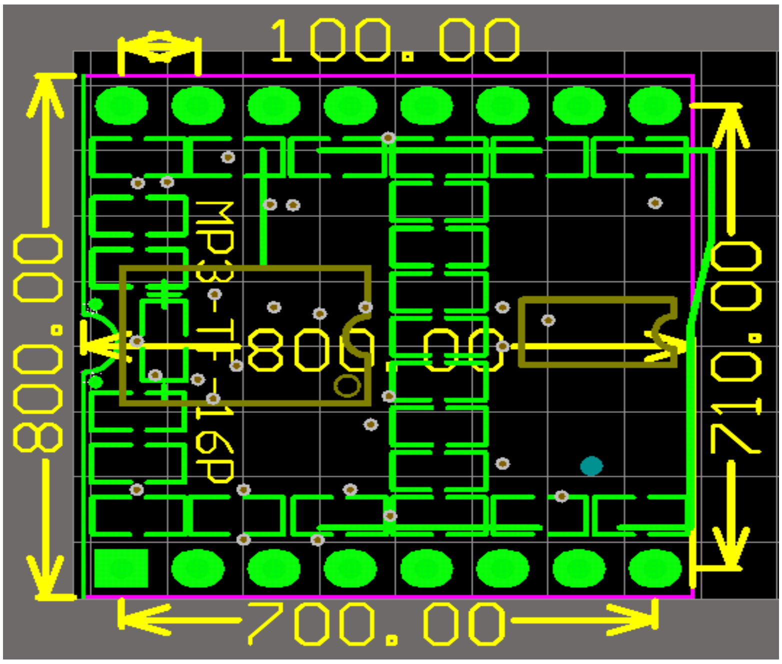


Figure 5.1 pcb size

6、 Note*

I/O Input Specification						
Item	Description	Min	Type	Max	Unit	Test Condition
VIL	Low-Level Input Voltage	-0.3	-	0.3*VDD	V	VDD=3.3V
VIH	High-Level Input Voltage	0.7VDD	-	VDD+0.3	V	VDD=3.3V
I/O Output Specification						
Item	Description	Min	Type	Max	Unit	Test Condition
VOL	Low-Level Output Voltage	-	-	0.33	V	VDD=3.3V
VOH	High-Level Output Voltage	2.7	-	-	V	VDD=3.3V

1. The module's external interfaces are 3.3V TTL level, so please note the level conversion during the hardware circuit design, also in strong interference environment, electromagnetic

compatibility note some protective measures, GPIO using opt coupler isolation, increasing TVS etc.

2, ADKEY key values are in accordance with the general use of the environment, if the strong inductive or capacitive load environment, please note that the module power supply is recommended to use a separate isolated power supply, another matched beads and inductors for power filtering, we must ensure that the input power as much as possible the stability and clean. If you really can not be guaranteed, please contact us to reduce the number of keys to redefine wider voltage distribution.

3. For general Serial communication, please pay attention to level conversion. If strong interference environment, or long distance RS485 applications, then please note that signal isolation, in strict accordance with industry standard design communication circuits.

4. Mickro SD

1. General Description

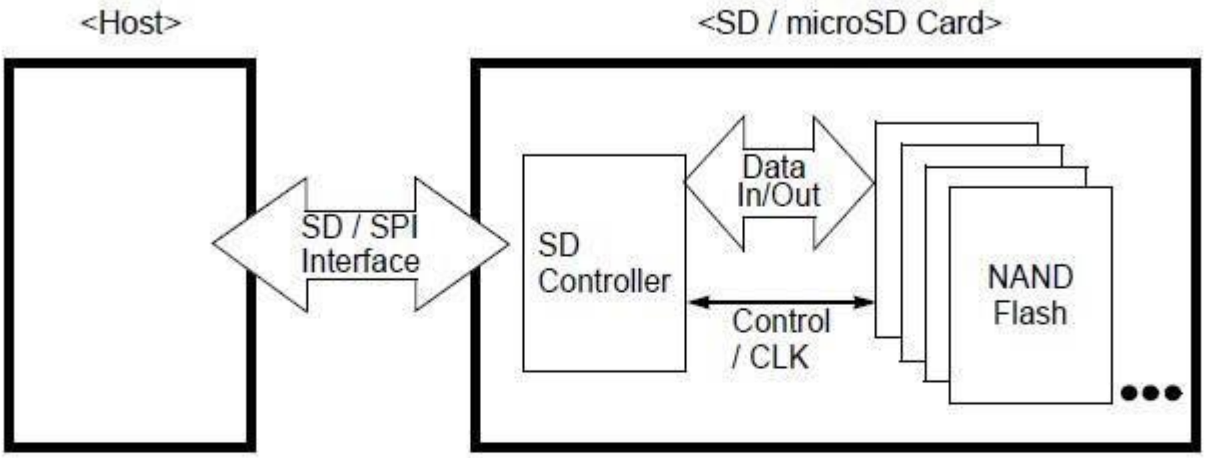
The eekoo The SD/microSD is a memory card that is specifically designed to meet the security, capacity, performance and enviroment requirements inherent in newly emerging audio and video consumer electronic devices. microSD cards are based on a 8-pin interface designed to operate in a maximum operating frequency of 100 MHz. The interface for microSD card products allows for easy integration into any design, regardless of which type of microprocessor is used. In addition to the interface, microSD card products offer

an alternate communicationprotocol based on the SPI standard.

2. Product Features

- Up to 128GB of data storage
- High transmission speed (Class 10)
- SD - protocol compatible
- Supports SPI Mode
- Voltage range of 2.7 to 3.6V
- Correction of memory field errors
- Card removal during read operation will never harm the content
- Memory field error correction
- Dimension : 15mm(L) x 11mm(W) x 1mm(H)

3. System Block Diagram



a. Reliability and Durability Specifications

Temperature	Operating: -25°C to 85°C Storage: -40°C(168h) to 85°C(500h)

moisture and corrosion	Operating: 25°C / 95% rel. humidity Non-Operating: 40°C / 93% rel. hum./500h salt water spray: 3% NaCl/35C; 24h acc. MIL STD Method 1009
Durability	10,000 mating cycles
Bending	10N
Torque	0.10N*m. ±2.5∘max
Drop Test	1.5m free fall
Visual Inspection/Shape and Form	No warp age; no mold slim; complete form; no cavities; surface smoothness≦-0.1mm/ cm2 within contour; no cracks; no pollution (oil, dust, etc.)

b. System Reliability and Maintenance

MTBF	>1,000,000 hours
Preventive Maintenance	None
Data Reliability	< 1 non-recoverable error in 1014 bits read
Endurance	3,000~10,000 write/erase cycles

c. Electrical Static Discharge (ESD) requirement

ESD Protection	**Contact** **Discharge: Air**	**±4KV, Human body model according to IEC61000-4-2.EN55024** **±8KV, Human body model**

5. Interface Description

a. General Description of Pins and Registers

The Micro SDHC has nine exposed contacts on one side. The host is connected to the SD Memory Card using a eight pin connector.

Pin Assignment in SD Bus Mode Pad Definition

PIN#	Name	Type	Micro SD Description
1	DAT2	I/O	Card Detect/ Data Lin [Bit 3]
2	CD/DAT3	I/O	Card Detect / Data Line
3	CMD	PP	Command / Response
4	VDD	S	Supply voltage
5	CLK	I	Clock
6	VSS	S	Supply Voltage Ground
7	DAT0	I/O	Data Line [Bit 0]
8	DAT1	I/O	Data Line [Bit 1]

Note:

4) S=power supply; I=input; O=output using push-pull drivers.

5) The extended DAT lines (DAT1-DAT3) are input on power up; they start to

operate as DAT lines after the SET _BUS_WIDTH command.

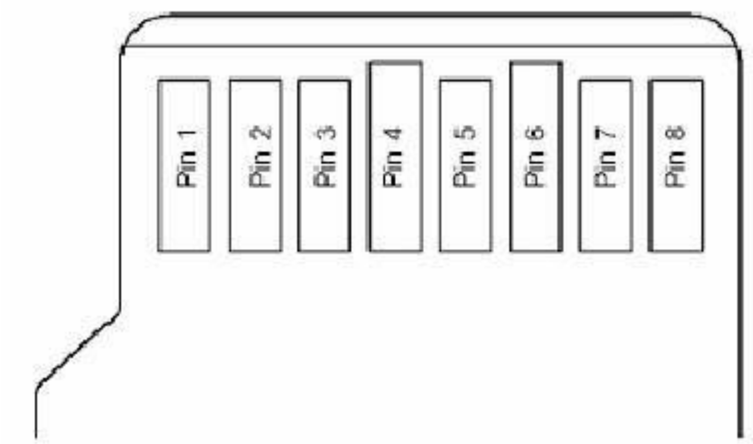
6) After power up, this line is input with 50Kohm pull-up (can be used for card detection or SPI mode selection).

The pull-up should be disconnected by the user, during regular data transfer, with SET_CLR_CARD_DETECT(ACMD42) command.

Pin Assignment in SPI Bus Mode Pad Definition

PIN#	Name	Type	Micro SD Description
1	RSV	I	Reserved
2	CS	I	Chip Select (neg true)
3	DI	S	Data In
4	VDD	S	Supply Voltage
5	SCLK	I	Clock
6	VSS	S	Supply Voltage Ground
7	DO	O	Data Out
8	RSV	I	Reserved

Micro SD memory Card Pin Assignment

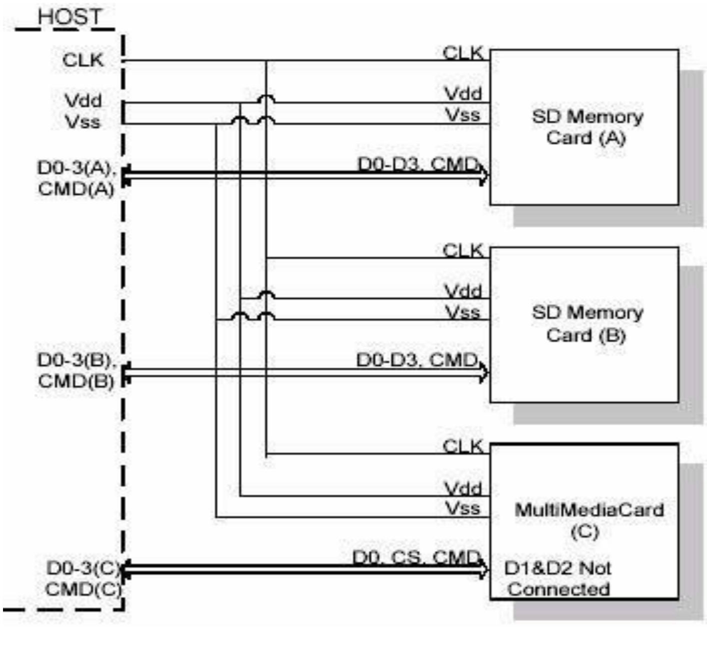


b. SD Bus Topology

The SD bus has six communication lines and three supply lines:

· CMD: Command is bi-directional signal.(Host and card drivers are operating in push pull mode.)
· DAT0-3: Data lines are bi-directional signals. (Host and card drivers are operating in push pull mode.).
· CLK: Clock is a host to cards signal. (CLK operates in push pull mode.)
· VDD: VDD is the power supply line for all cards.
· VSS [1:2]: VSS are two ground lines.
·The following figure shows the bus topology of several cards with one host

in SD Bus mode.



During the initialization process, commands are sent to each card individually, allowing the application to detect the cards and assign logical addresses to the physical slots. Data is always sent to each card individually. However, to simplify the handling of the card stack, after initialization, all commands may be sent concurrently to all cards. Addressing information is provided in the command packet.

c. Power Protection

Card can be inserted into or removed from the bus without damage. If one of the supply pins (VDD or Vss) is not connected properly, then the current is drawn through a data line to supply the card. Data transfer operations are protected by CRC codes; therefore, any bit changes induced by card insertion and removal can be detected by the Micro SD bus master. The inserted card must be properly reset also when CLK carries a clock frequency fpp.

If the hot insertion feature is implemented in the host, than the host has to withstand a shortcut between VDD and Vss without damage.

d. SPI Bus Topology

The memory Card SPI interface is compatible with SPI hosts available on the market. As any other SPI device the Micro SD Memory Card SPI channel consists of the following 4 signals:

5) CS: Host to card Chip Select signal.

6) SCLK: Host to card clock signal.

7) Data In: Host to card data signal.

8) Data Out: Card to host data signal.

Another SPI common characteristic, which is implemented in the Memory Card as well, is byte transfers. All data tokens are multiples of 8 bit bytes and always byte aligned to the CS signal.

The SPI standard defines the physical link only and not the complete data transfer protocol. In SPI Bus mode, the Micro SD Memory Card uses a subset of the Micro SD Memory Card protocol and command set. The Micro SD Memory Card identification and addressing algorithms are replaced by a hardware Chip Select (CS) signal.

A card (slave) is selected, for every command, by asserting (active low) the CS signal.The CS signal must be continuously active for the duration of the SPI transaction (command, response and data). The only exception is card programming time. At this time the host can de-assert the CS signal without affecting the programming process.

The bi-directional CMD and DAT lines are replaced by uni-directional data In and data Out signals. This eliminates the ability of executing commands while data is being read or written. An exception is the multi read/write operations. The Stop Transmission command can be sent during data read. In the multi block write operation a Stop Transmission token is sent as the first byte of the data block.