Everything about Internet of Things and more

ESP8266 ARDUINO TUTORIAL – WIFI MODULE COMPLETE REVIEW

Technology goes ahead exponentially with each year whether we do something or not. With the same speed engineers work hard to reduce the size of every electronic device or component and loose most of the wiring. If you look back 20 years ago a device was called portable if you could rise it in your arms and carry up. Today nothing is portable if it can't fit in your pocket.

Nowadays devices, either they are made for regular or industrial use, are based on wireless communication technology and the main reason is not to get rid of wires, but to be able to interconnect between them. Meanwhile buying a wireless device became natural for everyone and price for WiFi Ready equipment lowered with time passing.

While you read this, a WiFi microchip has no more than 5mm length and can be powered with as low as 10 micro Amps during sleep period. In this article we are going to test one of the cheapest and easy to use WiFi development platform, the ESP8266 arduino compatible device.

Cheap WiFi solution for your first IoT project – ESP8266 Arduino compatible wifi module with 1Mb flash upgrade

The ESP8266 arduino compatible module is a **low-cost Wi-Fi chip** with **full TCP/IP capability**, and the amazing thing is that this little board has **a MCU (Micro Controller Unit) integrated** which gives the possibility to **control I/O digital pins** via simple and almost pseudo-code like programming language. This device is produced by Shanghai-based Chinese manufacturer, **Espressif Systems**.

This chip was first time seen in **August 2014**, in **ESP-01** version module, made by **AI-Thinker**, a third-party manufacturer. This little module allows the MCU to connect to WiFi network and create simple TCP/IP connections.

His **His very low price (1.7$ – 3.5$)** and the incredible small size attracted many **geeks** and **hackers** to explore it and use it in a **large variety of projects**. Being a true success, **Espressif** produces now many versions having different dimensions and technical specifications. One of the successors is **ESP32**.

You can find over the internet hundreds of projects and various implementations like home automation, data logging solutions, robotics, controlling things over the internet, even drones or copters.

ESP8266-01Technical specifications

- 32-bit RISC CPU: Tensilica Xtensa LX106 running at 80 MHz **
- 64 KiB of instruction RAM, 96 KiB of data RAM
- External QSPI flash – 512 KiB to 4 MiB* (up to 16 MiB is supported)
- IEEE 802.11 b/g/n Wi-Fi
- Integrated TR switch, balun, LNA, power amplifier and matching network
- WEP or WPA/WPA2 authentication, or open networks
- 16 GPIO pins **
- SPI, I²C,
- I²S interfaces with DMA (sharing pins with GPIO)
- UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
- 1 10-bit ADC

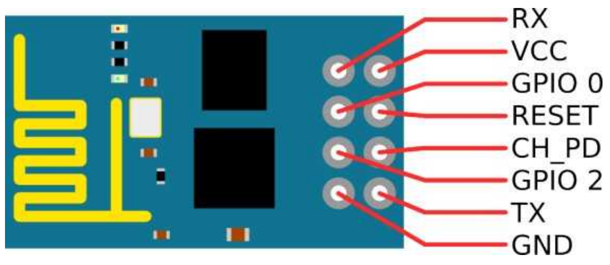
*** CPU and flash clock speeds can be raised via overclocking on some devices and the 16 I/O are not available in all versions.*

ESP8266 Arduino module comes with **PCB trace antenna** which seems to have a **very good coverage** (*I saw a demonstration with more than 1km range!!!*). Other version can have ***on-board ceramic antenna or an external connector*** which allows you to attach external WiFi antennas modules. ESP-01 has only 6 active pins, although the MCU can support up to 16 I/O. Board dimensions are **14.3 x 24.8 mm.**

Over the internet i found that ESP8266 arduino module, version 01, is sold in two or more versions, which at first glance seem quite the same. After buying both of them i saw that there is a difference in size of the flash memory. You may encounter issues while flashing if you don't make the proper settings according to board specifications.

Although the board default has 2 available GPIOs, you can do some workarounds and use other MCU available pins if you have the proper soldering tools. I managed to use GPIO 16 in order to wake up the device after DEEP SLEEP mode (*explained later in SLEEP MODES*).

Module pin description (pinout)



Pins are arranged in two rows, having 4 on each row. Some models have pin description on the PCB, which make it simple. On the top row you can find following pins from the left to the right:

1. **GND** (Ground from power supply)
2. **GPIO2** (Digital I/O programmable)
3. **GPIO0** (Digital I/O programmable, also used for BOOT modes)
4. **RX** – UART Receiving channel

On the bottom (second row) you can find:

1. **TX** – UART Transmitting channel
2. **CH_PD** (enable/power down, must be pulled to 3.3v directly or via resistor)
3. **REST** – reset, must be pulled to 3.3v)
4. **VCC** -3.3v power supply

Power supply and current consumption

All esp8266 arduino compatible modules must be powered with **DC current** from any kind of source that can deliver **stable 3.3V** and **at least 250mA**. Also **logic signal** is rated at **3.3v** and the RX channel should be protected by a 3.3v divisor step-down. You should be careful when using this module with Arduino or other boards which supplies 5v, because this module usually **do not come with overpower protection** and can be easily destroyed.

Here is the declared power consumption from Espressif:

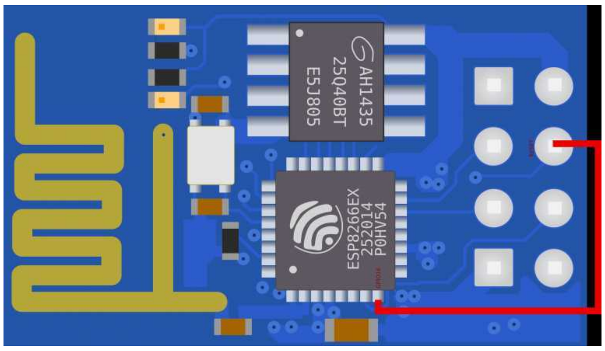
Parameters	Typical	Unit
Tx802.11b, CCK 11Mbps, P OUT=+17dBm	170	mA
Tx 802.11g, OFDM 54Mbps, P OUT =+15dBm	140	mA
Tx 802.11n, MCS7, P OUT =+13dBm	120	mA
Rx 802.11b, 1024 bytes packet length , -80dBm	50	mA
Rx 802.11g, 1024 bytes packet length, -70dBm	56	mA
Rx 802.11n, 1024 bytes packet length, -65dBm	56	mA
Modem-Sleep	15	mA
Light-Sleep	0.9	mA
Deep-Sleep	10	uA
Power Off	0.5	uA

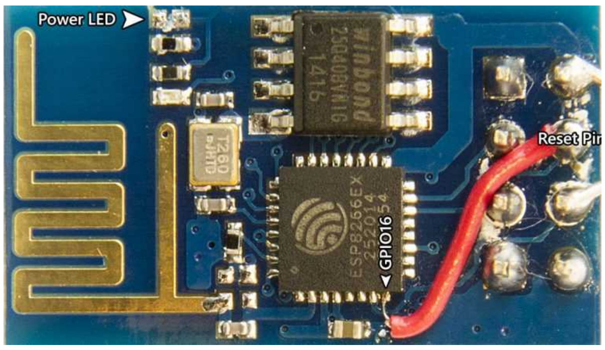
If you are going to use **ESP-01** in a project that is **powered by batteries or by solar power** it is mandatory to know everything about **ESP8266 arduino Sleep modes**. Current version offers 3 different sleep modes which can be triggered programmatically.

ESP8266WiFi library offers specific functions to call sleep modes which can take settings parameters that change the callback jobs after wake-up like waking up with RF module powered off or on.

The most important mode is **DEEP_SLEEP** because of the very low power consumption rates during sleep. Deep sleep mode is very common in projects that do data-logging at specific intervals and idle between measurements.

In order to take advantage of this mode when using esp8266 arduino compatible module, ESP-01 standard, you need to make a little workaround and connect REST pin with the GPIO16 pin (which is not available in default 6 six pins).
Here is an example how to do it



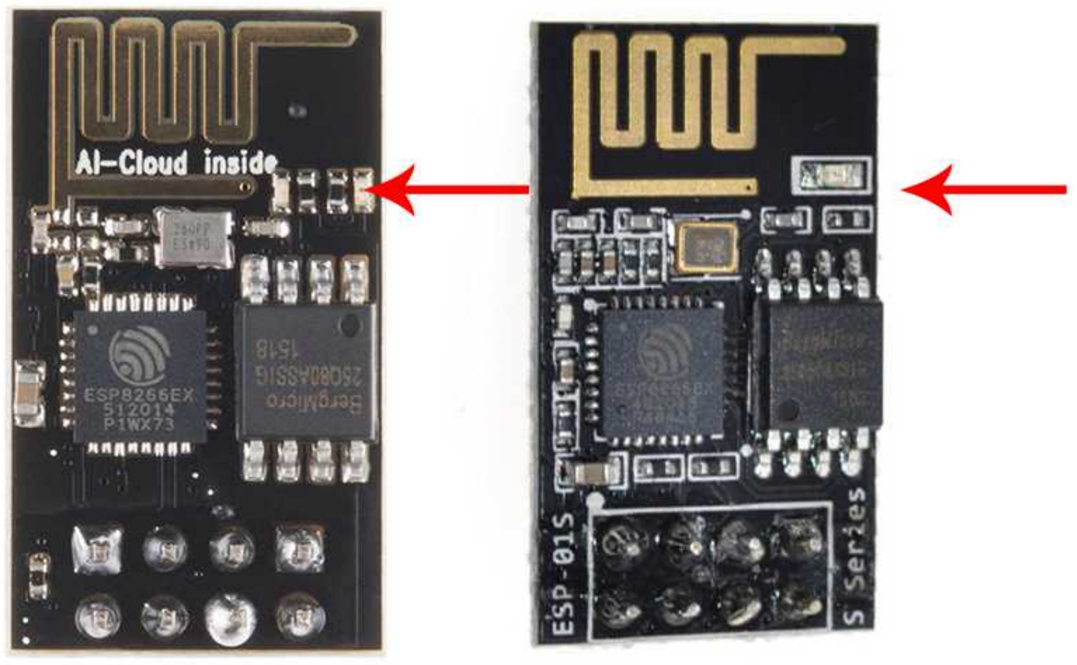


After doing this connection you can use the following command to trigger the deep sleep mode:

```
ESP.deepSleep(sleepTimeInSeconds * 1000000);
```

Another thing that you need to know about the esp8266 arduino 01 version is that usually comes with **two LEDs,** a **red** one for power, and a **blue** one for data transmitting. The **red LED is always on** when the **module** is **powered** on and the blue led blinks during serial activities. For newer versions the producers eliminated the RED pin because of the continuous power consumption, so if you are going to buy one, **try to find a version which has only the serial blue LED** especially if you are going to use a **battery power supply** in your project.

The differences can be seen in the following pictures:



While in the left picture you can see **two smd LEDs** near the antenna, one for power indicator and one for data indicator, the right module has a **single LED** just to indicate the data transmission. PCB design can differ from one to another because producers tend to reduce the costs by cutting down the materials.

Talking with ESP-01 (AT / LUA / Arduino)

ESP8266-01 gives you **many methods to communicate** with it **trough the RX/TX** pins or **over the air (OTA)**. The differences are not only in hardware but can be also in what kind of **firmware is flashed out of the box**. No matter what firmware comes default installed, **you** should be able to **flash your preferred firmware** by following the firmware flashing instruction from the datasheet.

This module can be programmed using **LUA code, Arduino code or directly trough AT commands** and this gives us more freedom when embedding this device in our projects. Also there are few **python firmware modes** but i haven't had the chance to test them. I personally choose to work with **Arduino** because of the past experience and **tones of libraries available**.

As it comes, **out of the box**, this module **is ready to talk** via **AT commands** without any other extra settings or configurations. There are many software applications which you can use to communicate via **AT** and have tones of **ready made tools** and **functions** which will make everything easier. I used **ESPlorer** and i totally recommend it, you can find it here. After booting, **to be able to use AT** commands, **module** should **display *"ready"*** on the serial monitor.

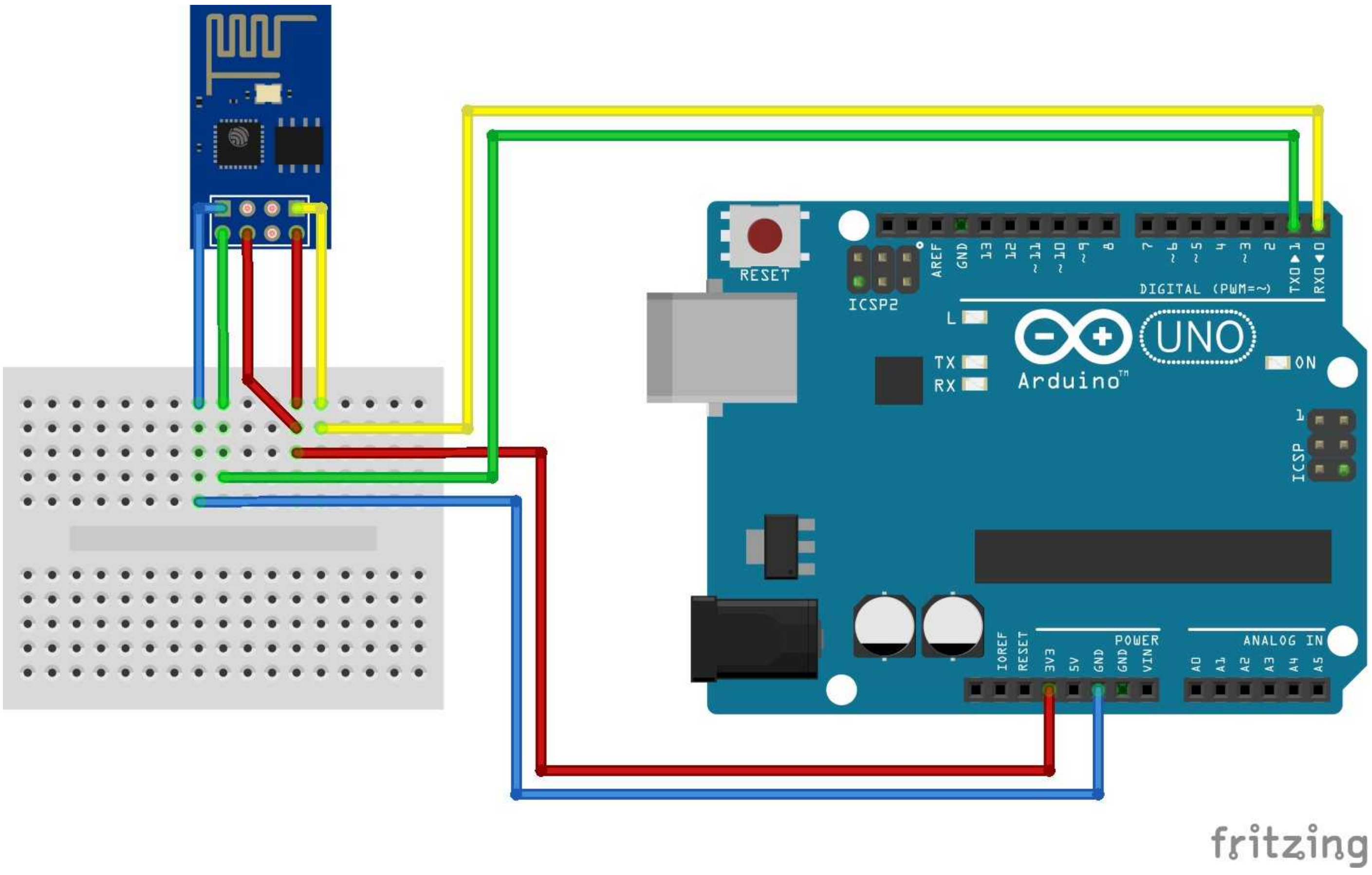
Few basic AT commands examples:

- AT – response OK
- AT+CWLAP – list nearby available WiFi networks
- AT+GMR – check the firmware version
- AT+CWJAP="<access_point_name>","<password>" – join WiFi network using credentials
- AT+CIFSR – get current allocated IP address

See here a complete list with AT instruction set.

In order to be able to talk with the ESP8266 arduino compatible module, you need to choose a way to **connect it with your computer.** You can communicate with the module via **standard Serial communication RS232** by using an **Arduino** board as a ***proxy/bridge***, by default Arduino having a USB to Serial converter integrated. If you are a beginner in development boards I totally recommend you one of the best Arduino books by Jeremy Blum, the best combination of a formaly trained electronics engineer and a Maker/Hacker.

Arduino Uno differs from all preceding boards in that it does not use the **FTDI USB-to-serial** driver chip. Instead, it features the *Atmega16U2* (Atmega8U2 up to version R2) programmed as a USB-to-serial converter. In order to use Arduino as a bridge, first you need to load an empty program on it. After doing that, you need to make the following connections in order to work:



UNO	ESP-01
RX	RX
TX	TX
3.3v	VCC
GND	GND
RST	3.3v / float
CH_PD	3.3v

After that you should be able to see data and send AT commands in Serial Monitor by selecting Arduino's COM port, **setting** a proper **baudrate**, default should be **115200**, and make the additional settings to read "**Both NL & CR**".

Firmware Over The Air (FOTA) solution in every embedded **DIY** or commercial project is a highly desirable if not a required feature today, when every project core needs to scalable. So the possibility to upload your code from a remote computer via Wi-Fi connection rather then Serial is a **high advantage in every project**. First you need FOTA needs needs prerequisites. First firmware upload needs to be done via Serial, and if the OTA routines are correctly implemented in the program next uploads can be done over the air.

Because the module needs to be exposed wirelessly, the chance to being hacked and loaded with maleficent code exists. You can improve your security by setting custom port and password. Check functionalities from the ArduinoOTA library that may help you to boost security. Because of the complexity of this procedures we will cover the full story in a future article, but for now be aware that this option exists and it works pretty good.

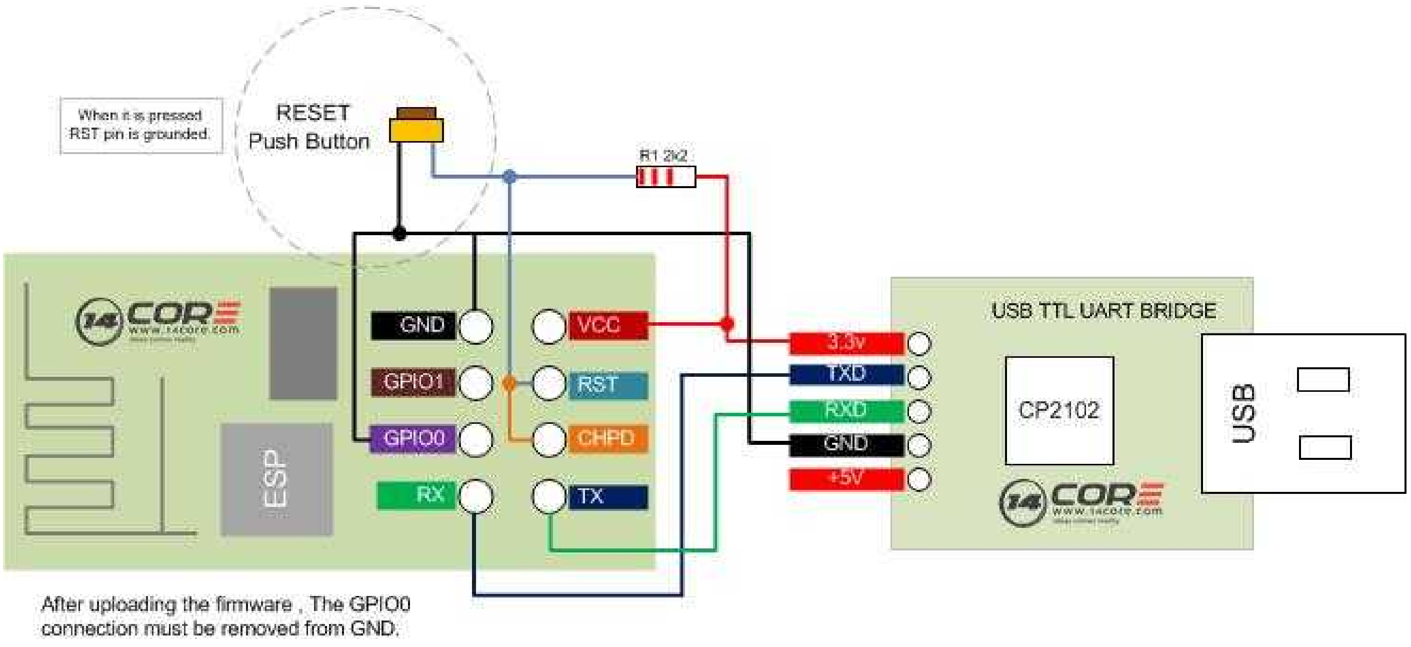
Another way to connect the esp8266 arduino module to computer is to use a **TTL or FTDI USB-to-serial** dedicated module. There are plenty of them on the market and there are quite cheap, but make no mistake, here quality does matter. You may encounter problems when working with it if ending up with a cheap one because of the differences in connections and also driver compatibility.

Most used TTL / FTDI converters chips are **CH340G**, **CP2102** and **FT232RL**. I personally used the first two ones and i have no problem when loading programs. Following connections need to be done:

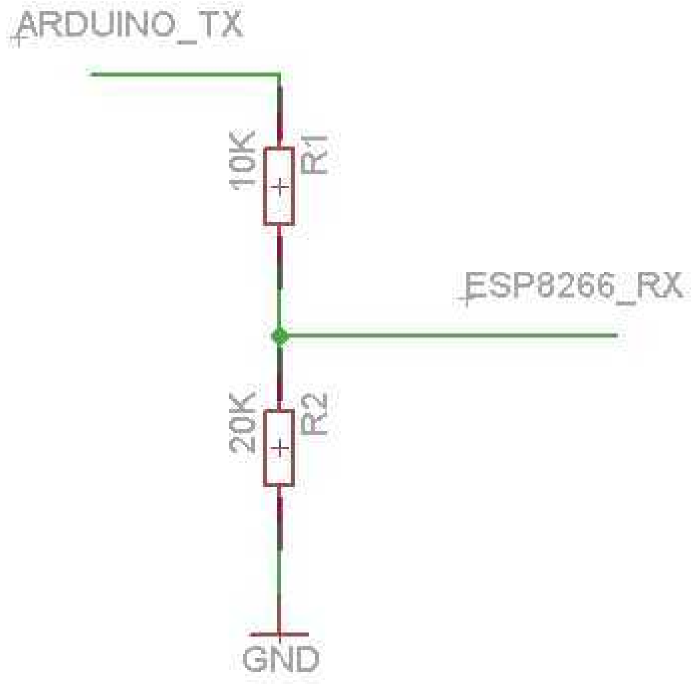
ESP-01	TTL / FTDI
RX	TX
TX	RX
VCC	3.3v
GND	GND
RST	3.3v / float
CH_PD	3.3v

I **highly recommend you not to use the TTL 3.3v power supply** because most of them are not able to provide enough power to handle the esp8266 arduino compatible device. The embedded voltage regulator used on this modules are not the happiest choice and you may get in trouble if it cannot support ESP peeks. If you choose to use an **external power supply** don't forget to setup a **common ground** in order to have a working circuit.

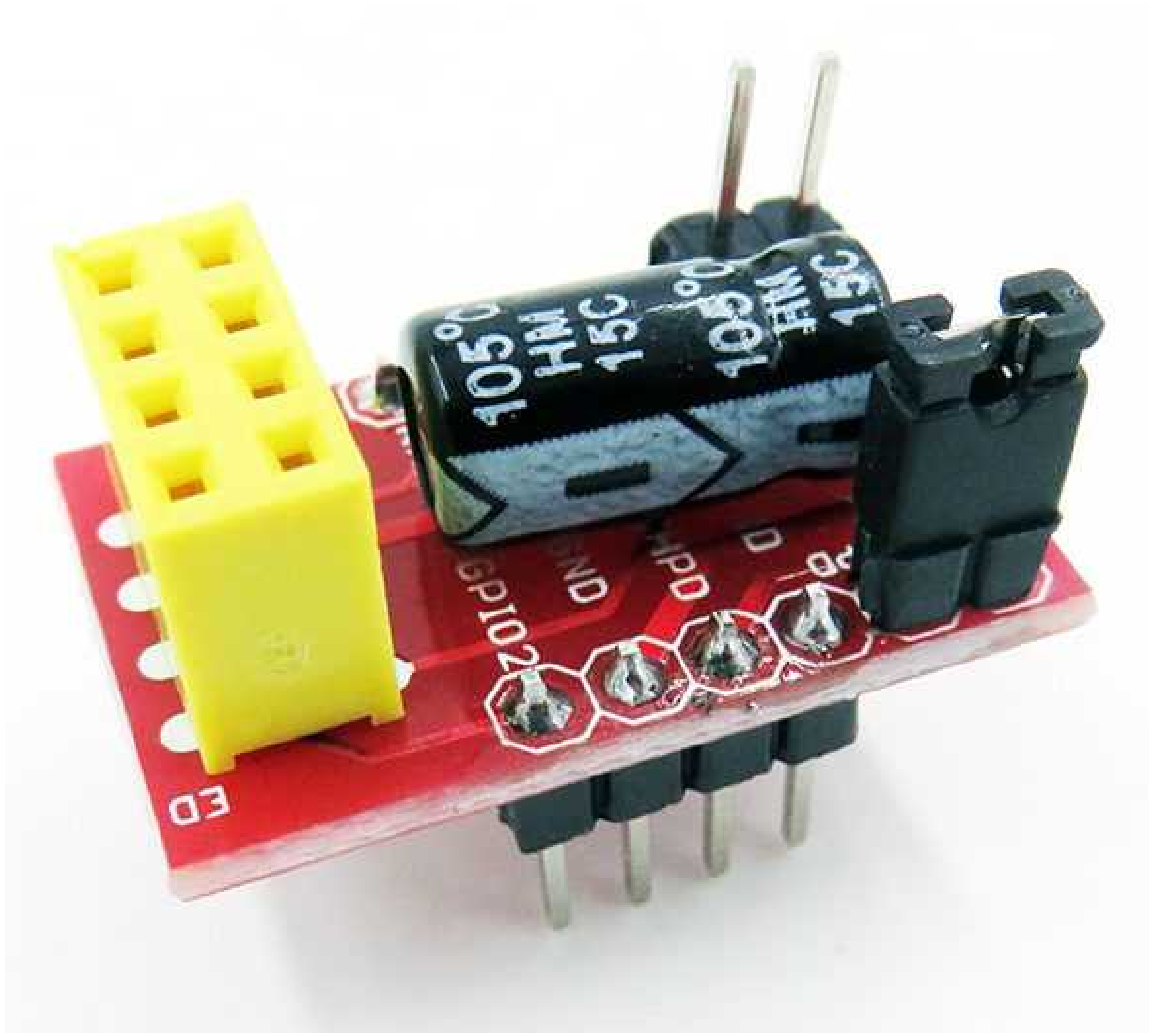
You can find TTL modules that have TX rated at 3.3v, if not, you shod step-down the TX channel to protect your ESP-01 module. You can see bellow a wiring scheme between ESP-01 and CP2102 which includes a reset button connected to ground, and also GPIO0 for boot switch.



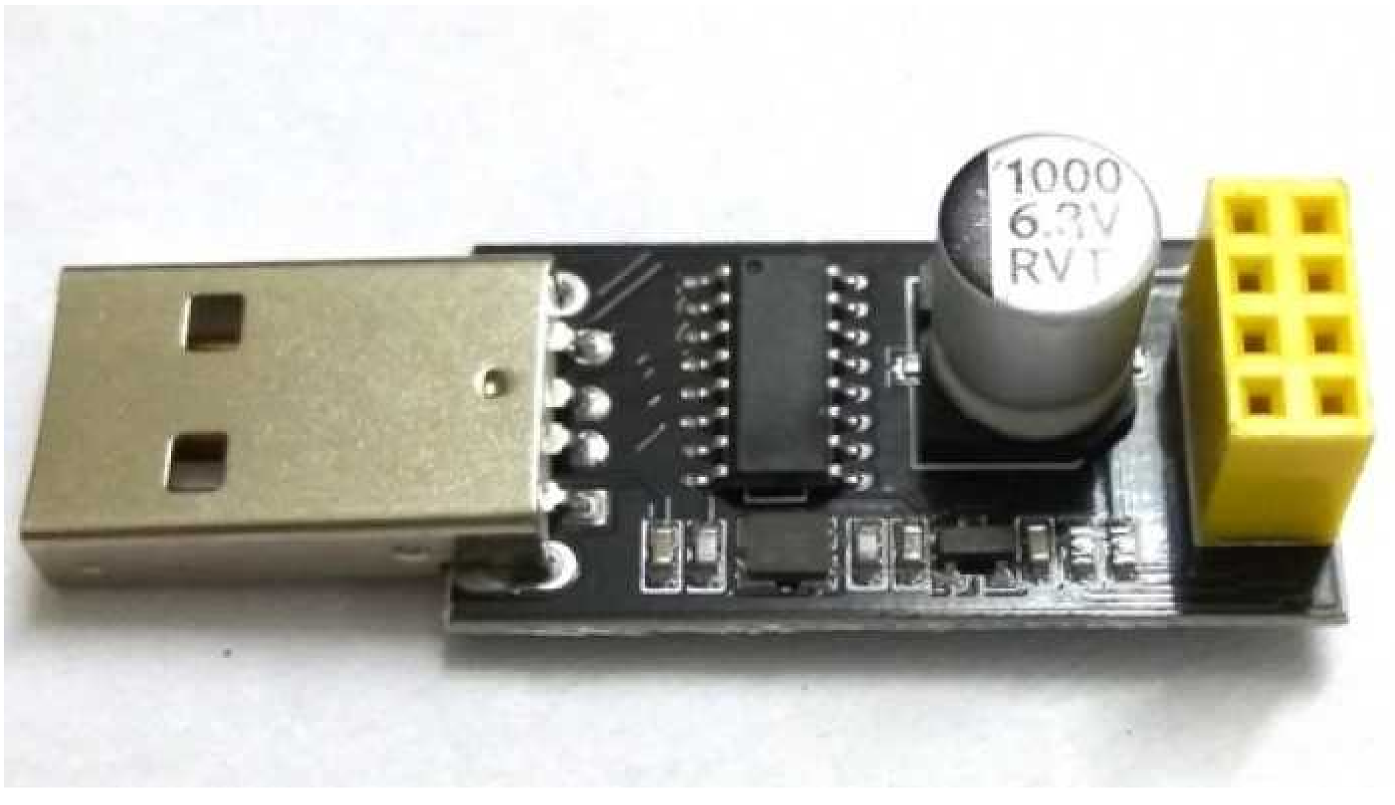
Here is a simple 3.3v divisor sketch using resistors:



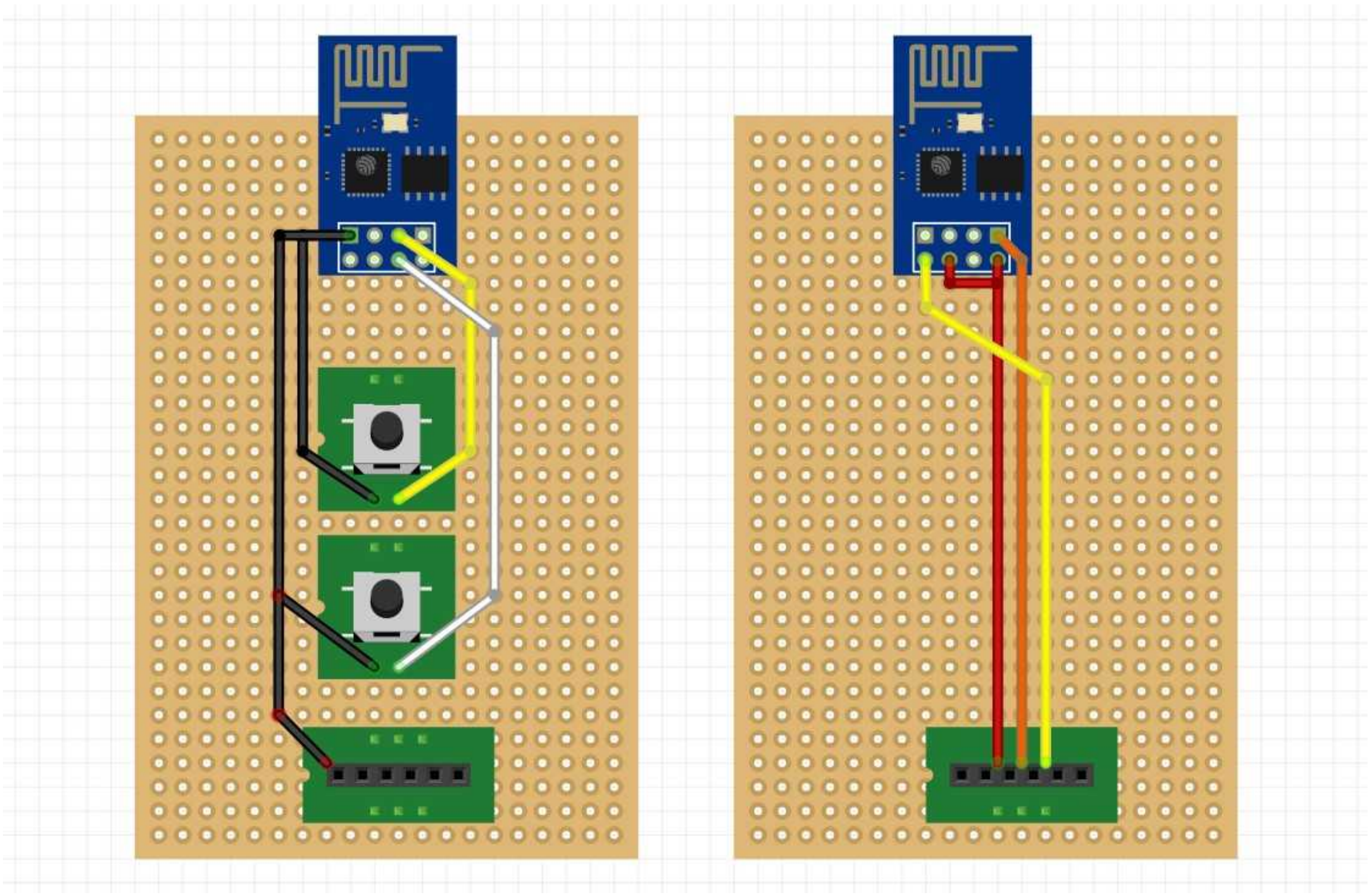
The most comfortable way to communicate with the ESP-01 is by using a **dedicated ESP01 programmer module**, which can be bought also from the same producers and the costs are very low. This kind of devices can have buttons integrated for RESET and BOOT switching and also come with a dedicated slot. All you need to do is to plug your esp and your done.

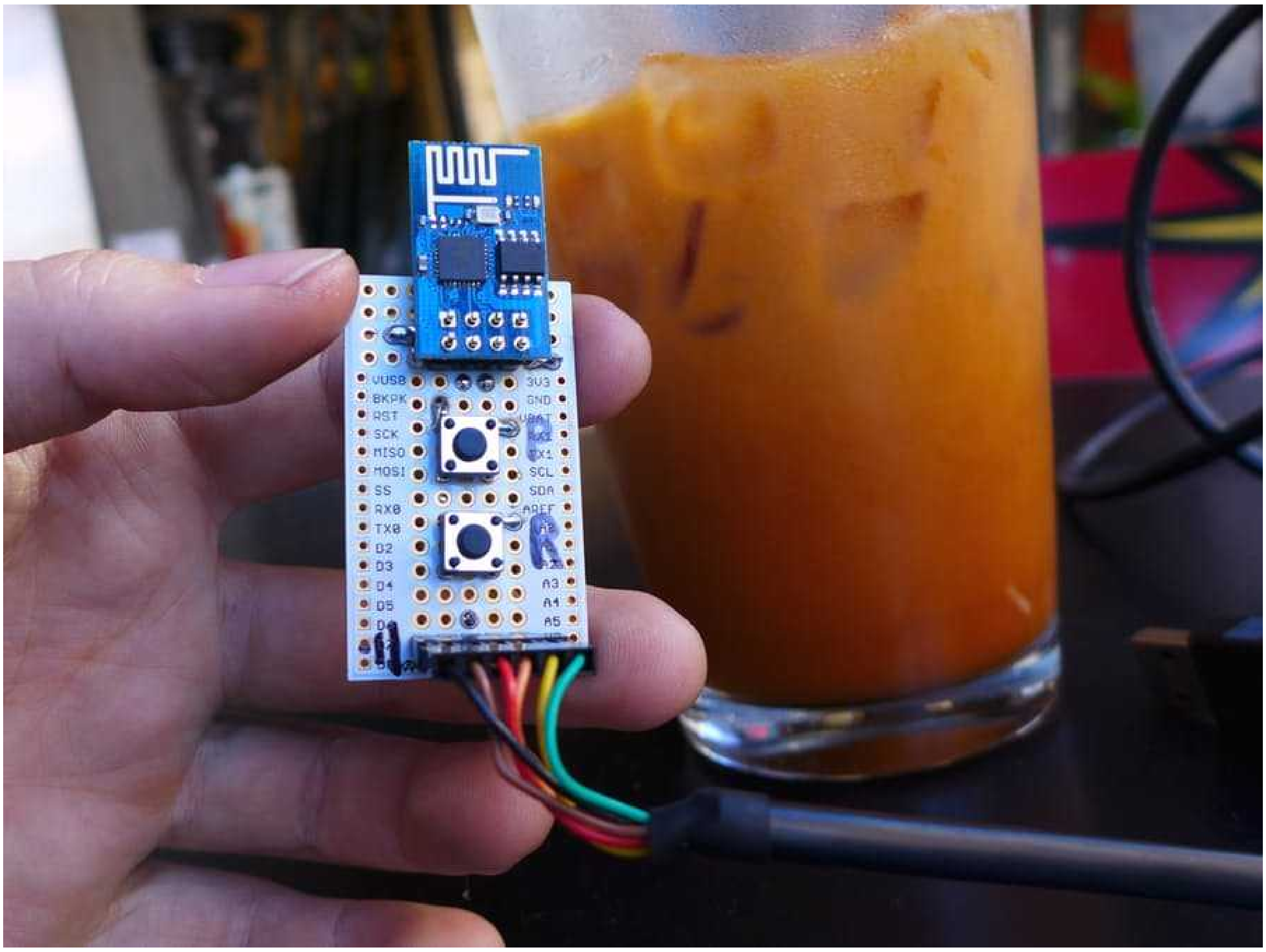


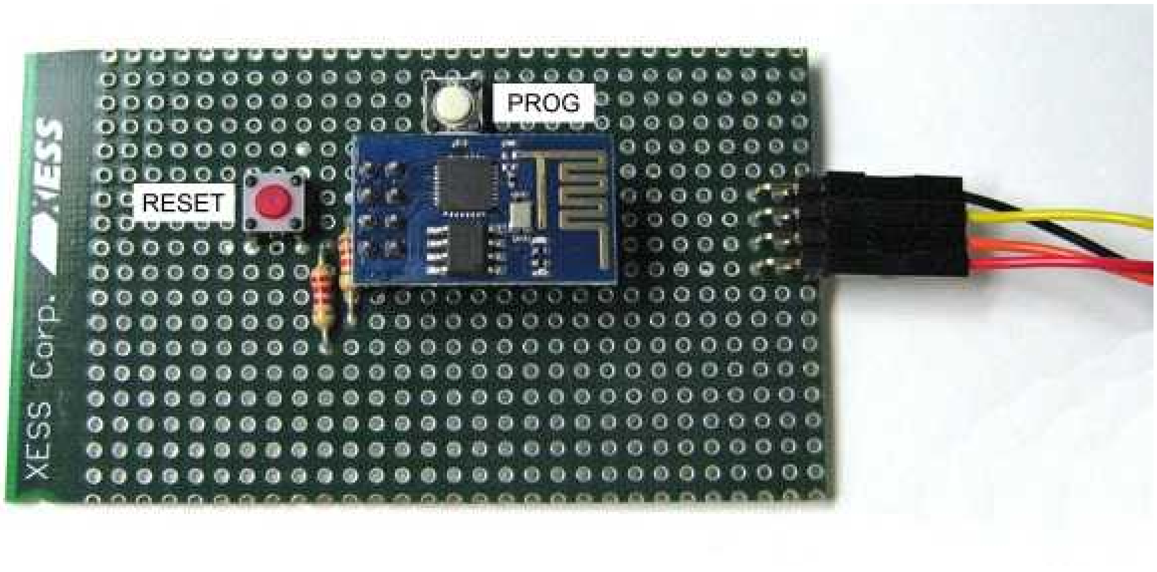
ESP-01
ESP-01 Adapter
TX
RX

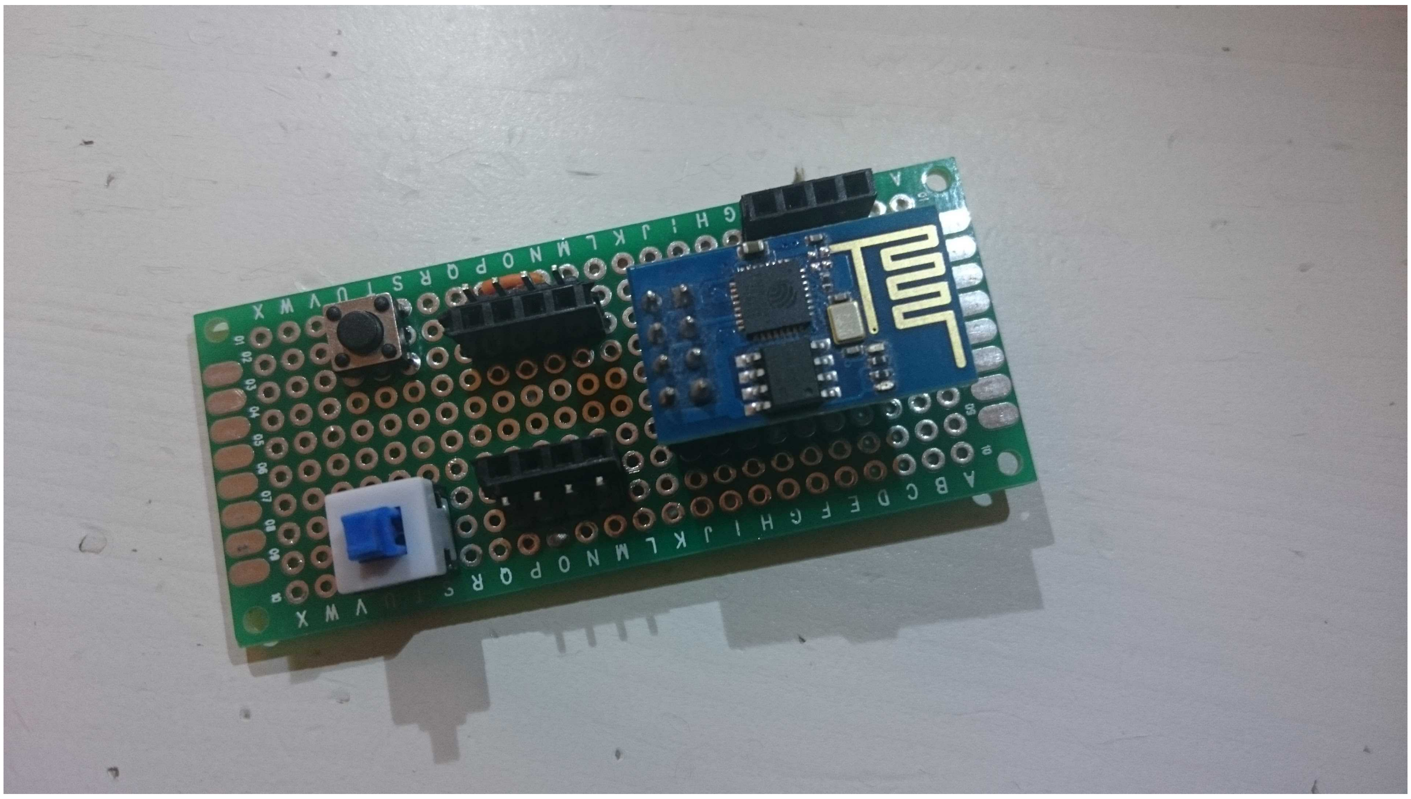

1000
6.3V
RVT

You can also build your own programmer if you have a bit of skills, you don't need to be an expert. Here are few homemade tryouts:



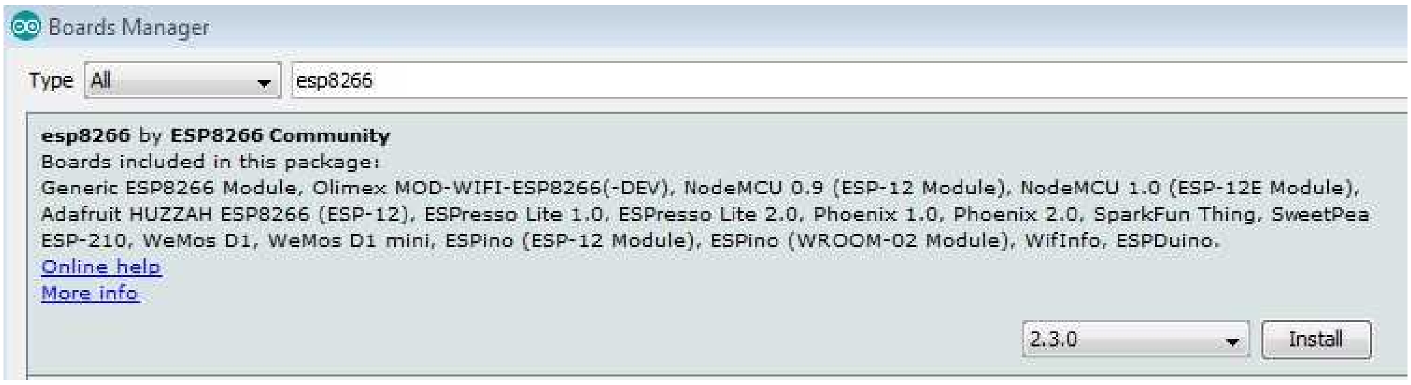






In order to **setup your Arduino IDE** to work with your esp8266 arduino compatible module you need to make the following steps:

1. Connect your ESP8266-01 Module to PC
2. Open your Arduino IDE
3. Go to File -> Preferences
4. Add this link to Additional Board Manager
5. Go to Tools -> Board Manager
6. Find ESP8266 board set and activate it
7. Select Generic ESP8266 board from Tools->Boards
8. Choose your programmer COM port
9. You are ready to go!



esp8266 by **ESP8266 Community**
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, Phoenix 1.0, Phoenix 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

Now, to be able to download the program to your ESP-01 module, you first need to put your device in the **proper BOOT mode** (*Download code from UART*). ESP8266-01 have the following **boot modes**:

MTDO / GPIO15	GPIO0	GPIO2	Mode	Description
L	L	H	UART	Download code from UART
L	H	H	Flash	Boot from SPI Flash
H	X	X	SDIO	Boot from SD-card

Note that L = LOW (putting to Ground / -3.3v) and H = HIGH (putting to 3.3v)

After resetting the module in *Download code from UART* you should see a message containing ***"boot mode:[1,6]"*** in the serial monitor, if you are on the correct baudrate. A wrong baudrate setting will display garbage text / characters or nothing at all. After that you should be able to upload your sketch to ESP8266. When upload is done, module should reset itself. Don't forget to **pull HIGH the GPI0** or the module will get in *Download mode* **again** and you will not be able to see it working. The module can be rebooted at anytime by pulling REST pin to LOW. After each reset it will follow the boot sequence and program loading.

Once the **ESP8266** board is installed and activated in Arduino IDE, you will be able to include **all ESP WiFi libraries and examples** that comes with the package. The most used library is **ESP8266WiFi** which offers many implementation examples like **WiFiClient, WiFiServer, WiFiAccessPoint** etc. You can find allot of projects examples over the internet, I for example, found great ideas on arduino.cc projecthub. Here is a simple **Arduino blink example** which you can use to test the esp module with the built in LED:

```
/*
 ESP8266 Arduino Blink by Simon Peter
 Blink the blue LED on the ESP-01 module
 This example code is in the public domain

 The blue LED on the ESP-01 module is connected to GPIO1
 (which is also the TXD pin; so we cannot use Serial.print() at the same time)

 Note that this sketch uses LED_BUILTIN to find the pin with the internal LED
*/

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);     // Initialize the LED_BUILTIN pin as an output
}

// the loop function runs over and over again forever
```

```
void loop() {
  digitalWrite(LED_BUILTIN, LOW);   // Turn the LED on (Note that LOW is the voltage
level
                                    // but actually the LED is on; this is because
                                    // it is acive low on the ESP-01)
  delay(1000);                      // Wait for a second
  digitalWrite(LED_BUILTIN, HIGH);  // Turn the LED off by making the voltage HIGH
  delay(2000);                      // Wait for two seconds (to demonstrate the active
low LED)
}
```

Off course, after that you can try a more complex example by loading a **ESP8266 Arduino WiFi Client example** program that sends data via WiFi to the data.spakfun.com iot platform:

```
/*
 *  This sketch sends data via HTTP GET requests to data.sparkfun.com service.
 *
 *  You need to get streamId and privateKey at data.sparkfun.com and paste them
 *  below. Or just customize this script to talk to other HTTP servers.
 *  ESP8266 Arduino example
 */

#include <ESP8266WiFi.h>

const char* ssid     = "your-ssid";
const char* password = "your-password";

const char* host = "data.sparkfun.com";
const char* streamId   = "....................";
const char* privateKey = "....................";

void setup() {
  Serial.begin(115200);
  delay(10);

  // 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 = "/input/";
  url += streamId;
  url += "?private_key=";
  url += privateKey;
  url += "&value=";
  url += value;

  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");
  unsigned long timeout = millis();
  while (client.available() == 0) {
    if (millis() - timeout > 5000) {
      Serial.println(">>> Client Timeout !");
      client.stop();
      return;
    }
  }

  // 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");
}
```

Or if you need to make a server in your network, you can try **ESP8266 Arduino Wifi Server** example program:

```
/*
 *  This sketch demonstrates how to set up a simple HTTP-like server.
 *  The server will set a GPIO pin depending on the request
 *    http://server_ip/gpio/0 will set the GPIO2 low,
 *    http://server_ip/gpio/1 will set the GPIO2 high
 *  server_ip is the IP address of the ESP8266 Arduino module, will be
 *  printed to Serial when the module is connected.
 */

#include <ESP8266WiFi.h>

const char* ssid = "your-ssid";
const char* password = "your-password";

// Create an instance of the server
// specify the port to listen on as an argument
WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  delay(10);

  // prepare GPIO2
  pinMode(2, OUTPUT);
  digitalWrite(2, 0);

  // Connect to 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");

  // Start the server
  server.begin();
  Serial.println("Server started");

  // Print the IP address
  Serial.println(WiFi.localIP());
}

void loop() {
  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  // Wait until the client sends some data
  Serial.println("new client");
  while(!client.available()){
    delay(1);
  }

  // Read the first line of the request
  String req = client.readStringUntil('\r');
  Serial.println(req);
  client.flush();

  // Match the request
  int val;
  if (req.indexOf("/gpio/0") != -1)
    val = 0;
  else if (req.indexOf("/gpio/1") != -1)
    val = 1;
  else {
    Serial.println("invalid request");
    client.stop();
    return;
  }

  // Set GPIO2 according to the request
  digitalWrite(2, val);

  client.flush();

  // Prepare the response
  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE
HTML>\r\n<html>\r\nGPIO is now ";
  s += (val)?"high":"low";
  s += "</html>\n";

  // Send the response to the client
  client.print(s);
  delay(1);
  Serial.println("Client disonnected");

  // The client will actually be disconnected
  // when the function returns and 'client' object is detroyed
}
```

And in the last example an **ESP8266 Arduino WiFi Access Point which hosts a web server** is created:

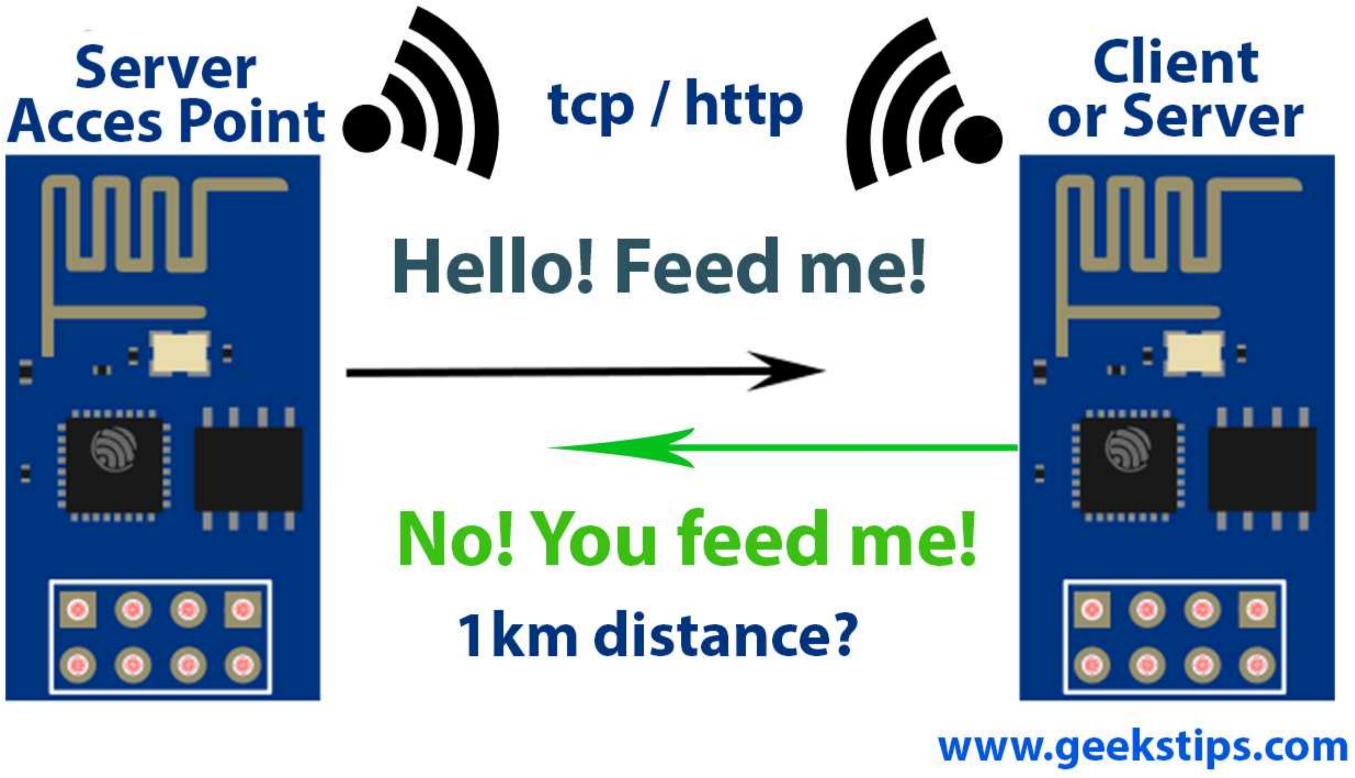
```
/* Create a WiFi access point and provide a web server on it.
ESP8266 Arduino example
 */

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

/* Set these to your desired credentials. */
const char *ssid = "ESPap";
const char *password = "thereisnospoon";

ESP8266WebServer server(80);

/* Just a little test message.  Go to http://192.168.4.1 in a web browser
 * connected to this access point to see it.
 */
void handleRoot() {
        server.send(200, "text/html", "<h1>You are connected</h1>");
}

void setup() {
        delay(1000);
        Serial.begin(115200);
        Serial.println();
        Serial.print("Configuring access point...");
        /* You can remove the password parameter if you want the AP to be open. */
        WiFi.softAP(ssid, password);

        IPAddress myIP = WiFi.softAPIP();
        Serial.print("AP IP address: ");
        Serial.println(myIP);
        server.on("/", handleRoot);
        server.begin();
        Serial.println("HTTP server started");
}

void loop() {
        server.handleClient();
}
```

Final thoughts

After playing with this module for few weeks, i find it being one of the coolest gadget i ever used in my projects and i have nothing than beautiful words about it. If you are even just a bit attracted about development boards and you haven't had the chance to play with ESP8266 series, this is one of the first thing that i recommend to get and play with. It easily checks every goal, the price is excellent, it has MCU integrated, it has Wi-Fi integrated, the signal range is pretty awesome, the size is excellent for embedding in most projects, and it can be powered by any average + batteries no problem.

In the further articles i will show you how i made to create a network of 5 ESP8266-01 in my house, and make them communicate each other to centralize data into a IoT web server. Also i want to come back with feedback about day to day behavior and especially power consumption.

Update: *Multiple ESP8266 talking each other article ready. Find here an incredible story about a farmer from Europe who managed to implement a huge ESP8266 WiFi topology using more than 40 ESP modules in a real life example.*



Update: Read the complete guide for the ESP32 module, newest upgrade to ESP8266, now with Bluetooth and integrated sensors. Full review with programing tutorials and code examples.

Hoping that this article inspired you, i kindly invite you **share this article**, **subscribe** my **YouTube** channel and **join** the communities on social networks. Please feel free to comment or send suggestions / remarks so i can **improve** the content quality!

Share on: Facebook Twitter Google+

48 Replies to "ESP8266 Arduino tutorial – WiFi module complete review"

1. *Ray* says:

   Nice job.
   Thanks for examples and notes.

   - *GeeksTips* says:

     Your welcome, feel free make remarks and also make proposals for the following articles ! Thank you!

   - *GeeksTips* says:

     Thanks for your time and interest! Stay close ☺

2. *Jonas* says:

   Good and thorough review. I'm looking forward to the multiple esp8266 communications tutorial.

   - *GeeksTips* says:

     Hello, I already implemented multiple ESP communication system in my house and I will write an tutorial about how to do that soon. Thank you for your interest!

3. *NoreenCocket* says:

   I see your website needs some fresh & unique articles.
   Writing manually is time consuming, but there
   is tool for this task. Just search for; Digitalpoilo's tools

4. *Jeepers01* says:

   Very good and informative article. Under point 4. above, the link to additional boards is missing. It should be
   http://arduino.esp8266.com/stable/package_esp8266com_index.json
   Feel free to email when the page is updated about the network of 5 ESP's as this sounds interesting. Will it be with MQTT?

   - *GeeksTips* says:

     Hello, thank you for appreciation. Check this article, you will find here a great WiFi topology in a real life example with more than 40 ESP modules connected each other.
     http://www.geekstips.com/two-esp8266-communication-talk-each-other/

5. *jackshowme* says:

   I am confused with the connection between Arduino and ESP8266 for RX.
   Why Arduino Rx(D0) is connected to ESP8266 RX and their TXs are also connected.
   I got the same result as you said here.
   In the begining of my experiment, I connected Arduino Rx/TX to ESP8266 TX/RX like as
   TTL / FTDI case. I got no response form ESP8266 when I give it a command.
   Next, I connected Arduino Rx/TX to ESP8266 RX/TX, and I found eveything is fine.
   I always think Arduino Rx(D0)/TX(D1) is for receive/transmit data to/from Arduino.
   Is it wrong?

   - *GeeksTips* says:

     When you use a dedicated TTL / FTDI the terminals needs to be aligned logical RX < -> TX, receiver < -> sender, but this is not the case when using Arduino board for programming. In this case the UNO acts like a repeater, a bridge. By running UNO with a blank sketch, you can use its serial port just to forward the signal. It actually does nothing but forwarding the packages, not programming. There are other modes to use UNO as a programmer, as a ISP, or by using emulated UART pins with SoftwareSerial library. I used this one, because is the simplest.

6. *Maxsuell Roger* says:

   Obrigado por seus artigos, estou lendo todos e estão me ajudando muito. Estarei inciando um projeto utilizando ESP8266, sou estudante de Engenharia Elétrica. Mais uma vez obrigado!

7. *Brundha sholaganga* says:

   Thank you it was very informative article ,by the way do we need to code the wifi module initially i mean after installation of library in IDE you mentioned it right , what i thought is wifi sheild need to be set in AT mode for changes in default settings and later dump the code in arduino as per required task of operation by arduino

8. *Mohammad Abdellatif* says:

   Hey,
   This is a great tutorial and i learned a lot from it.

Iam doing a project where i use a VC0706 camera with arduino and use esp8266 wifi to send the images to a webserver.
I managed to write a code that combines both of them. But when i use the method here to upload the data on the esp module, it connects to the internet but cannot communicate with the camera.
My question is: is there a way to modify the code so it works directly on the arduino? meaning i select the board as the arduino and not the esp?

9. *GeeksTips* says:

   You need to come with more details, otherwise I cannot know your setup. Good Luck!

10. *Chukwuka Okeke* says:

    I appreciate your articles, they are very useful. I'm trying to do a database project where an esp8266 and an Android app exchange data(in JSON format) on specific ports on the server. You may not necessarily bother about d Android. It goes this way, d esp8266 acts as an access point for d Android to connect to so could you pls guide me on how I could send data to the Android on specific ports?

11. *GeeksTips* says:

    This depends on how you want to receive the data, meaning on what channels. You could emulate a local webserver on the Android, and use it as an http Endpoint (or REST). After that you can just send requests to that endpoint from the ESP and attach the JSON. You can also use Blynk APP and send data over the serial, but its tricky.

12. *Batista* says:

    Sou novo na área e estou adorando a família ESP, muito obrigado e parabéns.
    ME AJUDE a elaborar um projeto, transmissão de dados para nível de água caixa dágua, via wifi com display no servidor indicando o nível.

13. *batista* says:

    elaborar sensor de nivel wifi e aparecer no display

14. *Ajay* says:

    after uploading any example from arduino IDE, getting below error in serial monitor, Please help me to resolve the error of esp8266 (esp01 black)

    ```
    Fatal exception (0):
    epc1=0x40218310, epc2=0x00000000, epc3=0x00000000, excvaddr=0x00000000, depc=0x00000000
    ```

    - *Lucian Vaduva* says:

      My piece of code should not cause such a Fatal Exception. Please make sure you have a proper power supply and you hooked up wiring correctly.

15. *Prasad M.* says:

    It is a good article, as i am using ESP8266-01 module for my project.
    I am in trouble for connecting and communicating multiple sensor output to single ESP board.(GPIO2) Please let me know if there are any alternative to above problem.

16. *marco* says:

hello your site and your youtube channel is very interesting compliments.
I would like to ask you a question and I hope you will answer.
I am developing a project to turn on a led connected to an esp8266 with a button connected to another esp8266.
could you help me?
thank you so much
I look forward to your kind reply.

17. *Loven S* says:

    Thanks alot ji…

    thanks for providing this tutorial to us!

18. *ferreteria on-line* says:

    each time i used to read smaller content which also clear their motive, and that is
    also happening with this piece of writing which
    I am reading at this place.

19. *CHETHAN B R* says:

    thanks for the article.. but i need help from your end … above proj. has no role of arduino -uno board..
    Please do upload some code which control's the ESP-8266-01 board from ARDUINO-UNO board with AT commands,,

    i need to send & receive data between 2 points via WiFi(ESP-8266)connected to Arduino boards

    Thanks in Advance..

20. *Doug Leary* says:

    Thanks very much for this thorough, informative writeup! Let me just add that TX and RX can be used as GPIO pins.
    TX=GPIO1, RX=GPIO3. I have been doing this without problems.

21. *gas* says:

    Hello,
    I have a WROOM-02 module that I want low power consumption, so it must go to sleep periodically. My goal is to connect the smartphone or PC, using Blynk, to this module and read the data that reads on the analog pin on other GPIOs. Also, if possible, send commands to the GPIOs. How it's possible?
    Thanks!

    - *Lucian Vaduva* says:

      Hi! Please read my Blynk articles. It covers all starting know-how for setting up Blynk with ESP8266. Is quite the same with ESP32 too. Thanks!
      https://www.geekstips.com/arduino-blynk-esp8266-iot-for-non-programmers/

22. *Samad* says:

    Hello everyone
    I want to communicate between labview and ESP8266 i want implementation ESP as a client and Labview as a server, How i program ESP and Labview ?
    please guide me!!!!…

- *Lucian Vaduva* says:

  Hi Samad, unfortunately I don't have any knowledge about connecting ESP8266 with Labview at this time, but I am going to document myself regarding this topic and come back with an article, and then ping you. Thank you.

23. *Daniel* says:

    Hi.
    I'm Daniel from Valencia (Spain- spain) I would like to tell you about a small project that I have in mind. How to connect two esp8266-01
    One as a client with gpio2 entry of a rain sensor, which sends order for 30 "to the other esp8266-01 as a server, with output from gpio2 to a relay, which causes the blind of a window to go down.
    Is it possible to do the project?
    I have looked a lot on the internet and I have not seen anything, could you give me a hand
    I have everything I need to start. THANK YOU
    Based on: ESP8266 Arduino IDE-Client Server Communication "Hello World"
    Telephone +34607330365

    - *Lucian Vaduva* says:

      Hi Daniel, There are many ways to make two ESP8266 communicate each other. One is by using Blynk Platform, with Bridge plugin. Another one is to make both ESP8266 servers and connect each other by using REST calls. I have covered both methods in my articles, you can find below more details:
      https://www.geekstips.com/blynk-bridge-another-way-to-communicate-between-two-esp8266/
      https://www.geekstips.com/two-esp8266-communication-talk-each-other/

24. *Swati* says:

    Arduino: 1.8.1 (Windows 8.1), Board: "Generic ESP8266 Module, 80 MHz, ck, 26 MHz, 40MHz, QIO, 512K (no SPIFFS), v2 Prebuilt (MSS=536), Disabled, None, 115200"

    ```
    Sketch uses 247051 bytes (49%) of program storage space. Maximum is 499696 bytes.
    Global variables use 32864 bytes (40%) of dynamic memory, leaving 49056 bytes for local variables. Maximum is 81920 bytes.
    warning: espcomm_sync failed
    error: espcomm_open failed
    error: espcomm_upload_mem failed
    error: espcomm_upload_mem failed
    ```

    We have tried various methods but we are unable to rectify it Please give us a proper solution to avoid this problem since the simple blink code for esp826601 is not uploading.

25. *Adam* says:

    I have problem.

    ```
    Sketch uses 247051 bytes (49%) of program storage space. Maximum is 499696 bytes.
    Global variables use 32864 bytes (40%) of dynamic memory, leaving 49056 bytes for local variables. Maximum is 81920 bytes.
    warning: espcomm_sync failed
    error: espcomm_open failed
    ```

error: espcomm_upload_mem failed
error: espcomm_upload_mem failed

We have tried various methods but we are unable to rectify it Please give us a proper solution to avoid this problem since the simple blink code for esp826601 is not uploading.

- *Lucian Vaduva* says:

  Hello Adam! From what I see in the error, this is not a sketch problem, but rather a communication issue between ESP and your PC. You need to make sure that your ESP-01 is in FLASH MODE before hitting upload button. Reset the ESP while pulling GPIO-0 LOW (to ground). You should should see in serial console something like mode 1-6 / 1-7.

26. *swg* says:

Great tutorial! just wanted to address an issue in one of your schematics — using 2 resistors as a voltage divider is a bad way to make a 3V3 supply for the ESP, since it draws nonzero current (and therefore the resistors are not a voltage divider). Instead, you should use a 3V3 regulator (or a buffer amplifier after the resistors).

27. *Type your name Ranjith* says:

This is good one for beginers.expecting more
Thankyou.

- *Lucian Vaduva* says:

  Thanks! need to find time to write ☺

28. *Dragan* says:

I have to admit this is the best tutorial on the entire Internet.

I've been bothering with esp 8266 5v relay (https://www.aliexpress.com/item/ESP8266-ESP-01S-5V-ESP01S-WiFi-Relay-Module-Things-Smart-Home-Remote-Control-Switch-For-Arduino/32849053934.html?spm=a2g0s.9042311.0.0.PhHZPp) for a couple of days. On esp8266 modul I have the latest firmware. I have communicated via arduino uno with esp modul on baudrate 9600. When I tried to control relay directly with android phone (App: Easy TCP,A00101A2 open relay, A00100A1 closed relay), nothing happens.

Would you be so kind to give me some instructions. Thank in advance.

29. *Dragan* says:

Problem solved.Thanks anyway.For any questions just ask!

30. *Florin* says:

Hello!
Can somebody help me please with this Error:
Arduino: 1.8.5 (Windows 10), Board: "Generic ESP8266 Module, 80 MHz, ck, 26 MHz, 40MHz, QIO, 512K (no SPIFFS), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200"

warning: espcomm_send_command: can't receive slip payload data
trying to connect
flush start

```
setting serial port timeouts to 1 ms
setting serial port timeouts to 1000 ms
flush complete
espcomm_send_command: sending command header
espcomm_send_command: sending command payload
espcomm_send_command: receiving 2013 bytes of data
read 0, requested 1
error: failed reading byte
warning: espcomm_send_command: can’t receive slip payload data
trying to connect
flush start
setting serial port timeouts to 1 ms
setting serial port timeouts to 1000 ms
flush complete
espcomm_send_command: sending command header
espcomm_send_command: sending command payload
espcomm_send_command: receiving 2013 bytes of data
read 0, requested 1
error: failed reading byte
warning: espcomm_send_command: can’t receive slip payload data
warning: espcomm_sync failed
error: espcomm_open failed
error: espcomm_upload_mem failed
```

I am using a dedicated ESP01 programmer module. I can’t upload even a blink file.
Here you got a photo with the module and support module:
https://ibb.co/gte9UH
Thank you!

31. *Giovanni* says:

Hello, congratulations for your article.
I’m reading a lot of articles on the web but what I get is a big mess in my brain.
My intent is to use an Arduino UNO connected to an ESP-01 and use an MQTT server to control various information that can be consulted on the internet from any web browser.
Unfortunately in none of the articles so far consulted I found one that explains step by step the unequivocal way how to implement the code correctly.
Below I ask you some questions to which I would like a clear answer:
To connect ESP-01 to the home router I first have to flash the ESP-01 to establish the connection parameters, then in a second step do I have to program arduino to connect to the MQTT server and manage the sensors etc.? or should I use arduino to do everything, including connection to the router?
How should the data transmission from arduino to ESP-01 be carried out and then to the MQTT server?
Can you kindly indicate a simple step by step list of things to do?
example:
1. flash esp01 ….
2. load the sketch on arduino …

3. …
4. …

If you can also enter the code is welcome

I use Arduino to flash ESP-01 through Arduino IDE.
I thank you in advance if you want to answer my questions.

Leave a Reply

Your email address will not be published.

Type your name

Type your email

Type your website url

Post Comment

- About Me
- Internet of Things
- Arduino
- Tips
- NEW Battery Life Calculator

Amazon.com - DISCOUNT

Arduino Advanced Starter Kit - Instructions, Servo, Remote, IR
Amazon's Choice

BEST SELLER

Make Arduino Fly - A guide to build drones with Arduino
Amazon's TOP

Updated on: Apr 1, 2018

Your information is secured

- About Me
- Privacy Policy

- Cookie Policy
- Contact Me