Arduino Beginner's Guide

ESP8266 Setup Tutorial using Arduino

NOVEMBER 16, 2016 Rahul Iyer

The ESP8266 is a low-cost WiFi module that can be integrated easily into IoT devices. We've featured several projects using this module, such as **How To Make Smart Home Electronics: A Smart Mailbox** and **How To Read Your Arduino's Mind: Building A Childproof Lock**. This tutorial will walk you through setting up ESP8266 Wifi module which can be used with Arduino. The ESP8266 comes in many models with different functionalities. We'll be focusing on the ESP8266 ESP-01 module, the most common and basic one available.

What is ESP8266?

The ESP8266 is a small WiFi module built around the ESP8266 chip that can connect your microcontroller to the internet wirelessly for a very small cost. It can be a great option for Internet of Things (IoT) projects, but can be difficult to work with for beginner hobbyists who do not have prior experience with the module. In this tutorial, we hope to show you how to interface the ESP8266 with an Arduino and perform some basic functions like connecting it to a WiFi network.

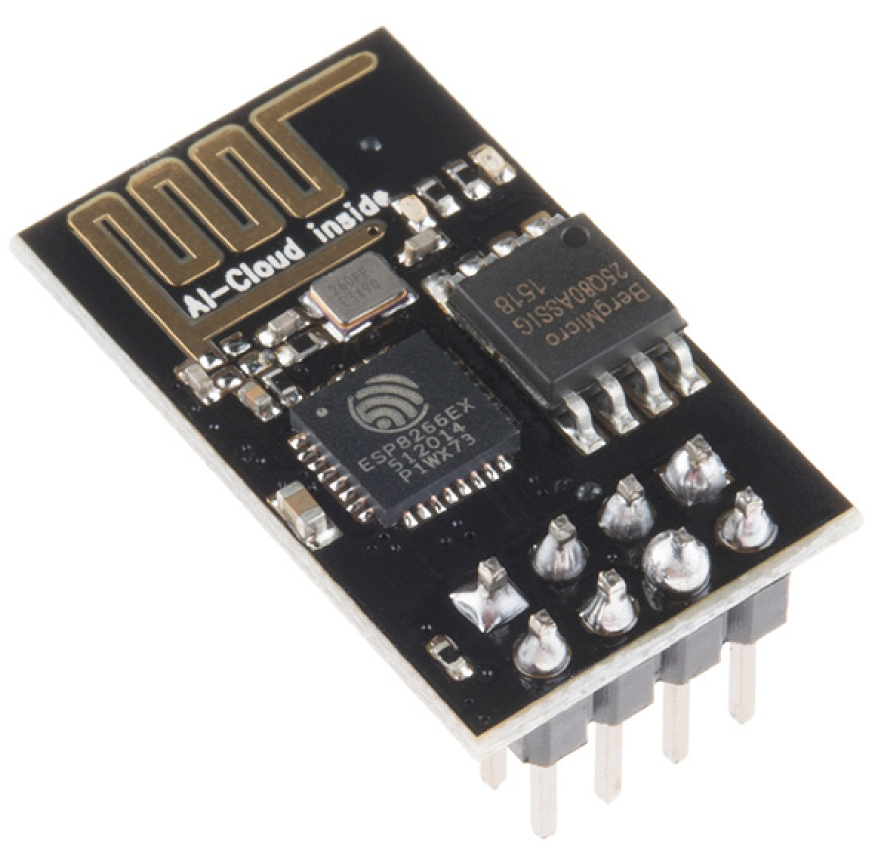


Figure 1. ESP8266 ESP-01 module / ©Sparkfun

ESP-01 Features – Sparkfun**:**

- 802.11 b/g/n
- Wi-Fi Direct (P2P), soft-AP
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLLs, regulators, DCXO and power management units
- +19.5dBm output power in 802.11b mode
- Power down leakage current of <10uA
- 1MB Flash Memory
- Integrated low power 32-bit CPU could be used as application processor
- SDIO 1.1 / 2.0, SPI, UART
- STBC, 1×1 MIMO, 2×1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)

The first feature to notice about the ESP8266 is its awkwardly spaced header pins. The module has 8 pins that serve different functions, but they are packed in a 4×2 arrangement that makes plugging the module into a breadboard impossible. This means that to prototype projects on a breadboard, you'll need male-female jumper wires to connect the pins on the ESP8266 to rows on the breadboard. If you'd like to make your prototyping more compact, you can also purchase breadboard breakouts for the ESP8266 such as this one. For prototyping, I chose to just use jumper wires.

The pinout for the ESP8266's pins are according to the following diagram:

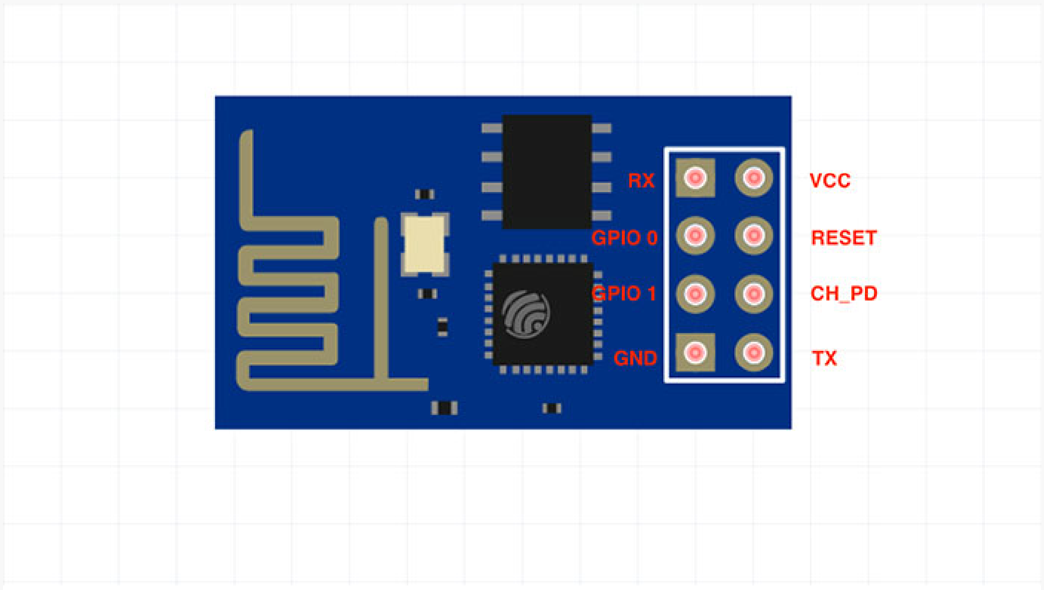


Figure 2. ESP8266 Pinout / © Github.com/ydonnelly

RX: UART serial communication receive pin

GPIO 0: GPIO pin (unused in this project)

GPIO 1: GPIO pin (unused in this project)

GND: Connection to Ground

VCC: Connection to 3.3V Vcc (Vcc cannot exceed 3.3V!)

RESET: Reset pin (pull down to reset)

CH_PD: Chip enable and power down pin

TX: UART serial communication transmit pin

Note that the maximum voltage input for the ESP8266 is 3.3V. Any input voltage greater than 3.3V will damage the module! To program settings on the ESP8266, we'll first need to connect it to a serial terminal on a computer through which we can send it special commands. Settings that we'll have to program include, for example, the SSID and password for the wifi network the module will be connected to. To connect the ESP8266 to a computer and configure its settings, we'll need a USB to serial adapter with 3.3V logic, along with a serial terminal program. Fortunately for us, we have the Arduino and the Arduino IDE's serial monitor! This means that we'll just have to connect the ESP8266 module to the Arduino and upload a custom sketch to the Arduino.

Use the following diagram to connect the ESP8266 module to the Arduino:

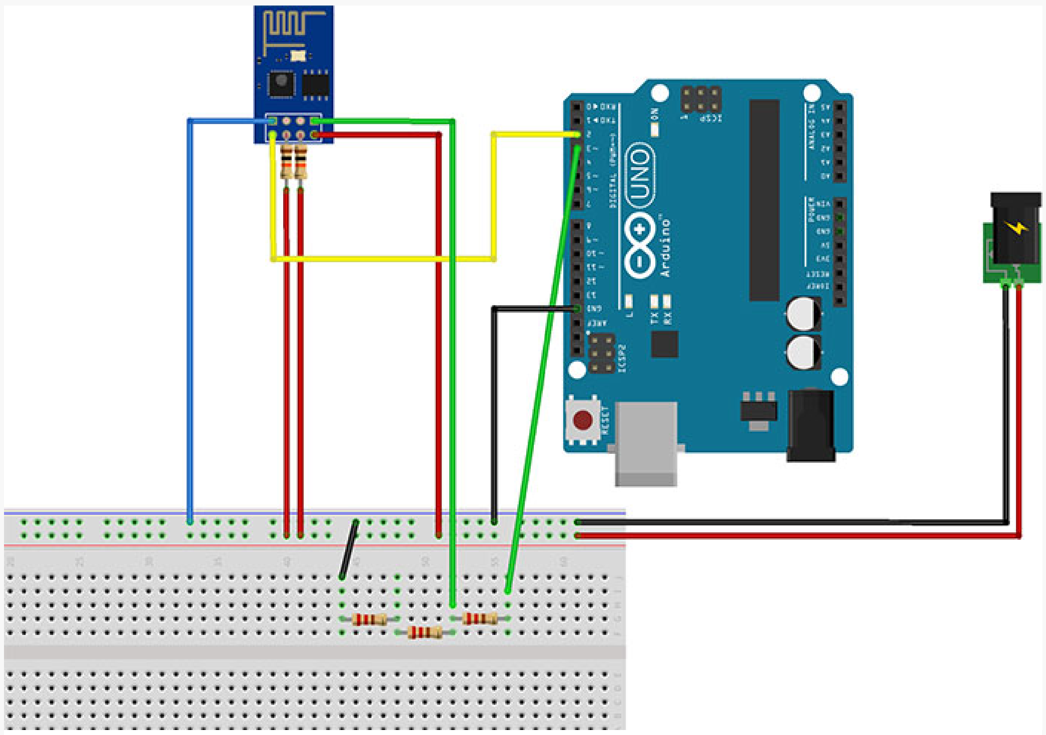


Figure 3. Connection between ESP8266 and Arduino Uno

You will need the following parts:

- Arduino Uno or similar module
- ESP8266 ESP-01 module
- 2 x 1kΩ resistors for CH_PD and RESET pull-up
- 3 x 220Ω resistors for serial line voltage divider
- 3.3V regulated power supply for ESP8266 (you cannot plug the power line into the Arduino’s 3.3V out pin because the ESP8266 draws a lot of current!)
    - You can use something like this: https://www.sparkfun.com/products/114
- Breadboard and jumper wires

A couple of features of this circuit stand out immediately.

First, the ESP8266 module is powered by a 3.3V regulated power supply. As described earlier, the ESP8266 has a maximum voltage input rating of 3.3 volts. Using a power supply rated higher than this recommended 3.3 volts could fry the module! When connecting the ESP8266 to this power supply, you should not only connect the Vcc and Ground pins to the power supply lines, but also connect the power supply’s ground line to the Arduino’s ground pin. Since the Arduino will be powered by the USB connection to the laptop, creating a common ground essentially creates a common reference (you can think of it as a baseline) to compare voltages and thereby interpret digital high and low signals.

Second, the receive (RX) line for the ESP8266 module is connected to the output of a resistor voltage-divider circuit. We do this to shift the serial communication logic level (the highs and lows of the digital signals that make up the serial communications) from a logic high of 5 volts on the Arduino to a logic high of 3.3 volts on the ESP8266 module. Again, the ESP8266 is specified with 3.3V logic, so connecting the module’s receive line directly to the Arduino’s transmit line could damage the device. We do not need to shift levels on the ESP8266’s transmit line because the module’s 3.3V logic high is a high enough voltage to also register as a logic high on the Arduino.

Finally, there are two pull-up resistors on the ESP8266’s CH_PD and RESET pins. These are specified by the module’s creators to ensure that the device functions properly while it is being used. To reset the ESP8266 or disable it, these pins must be pulled down, but since we do not want those processes to occur, we will instead pull these pins high through some 10kΩ pull-up resistors.

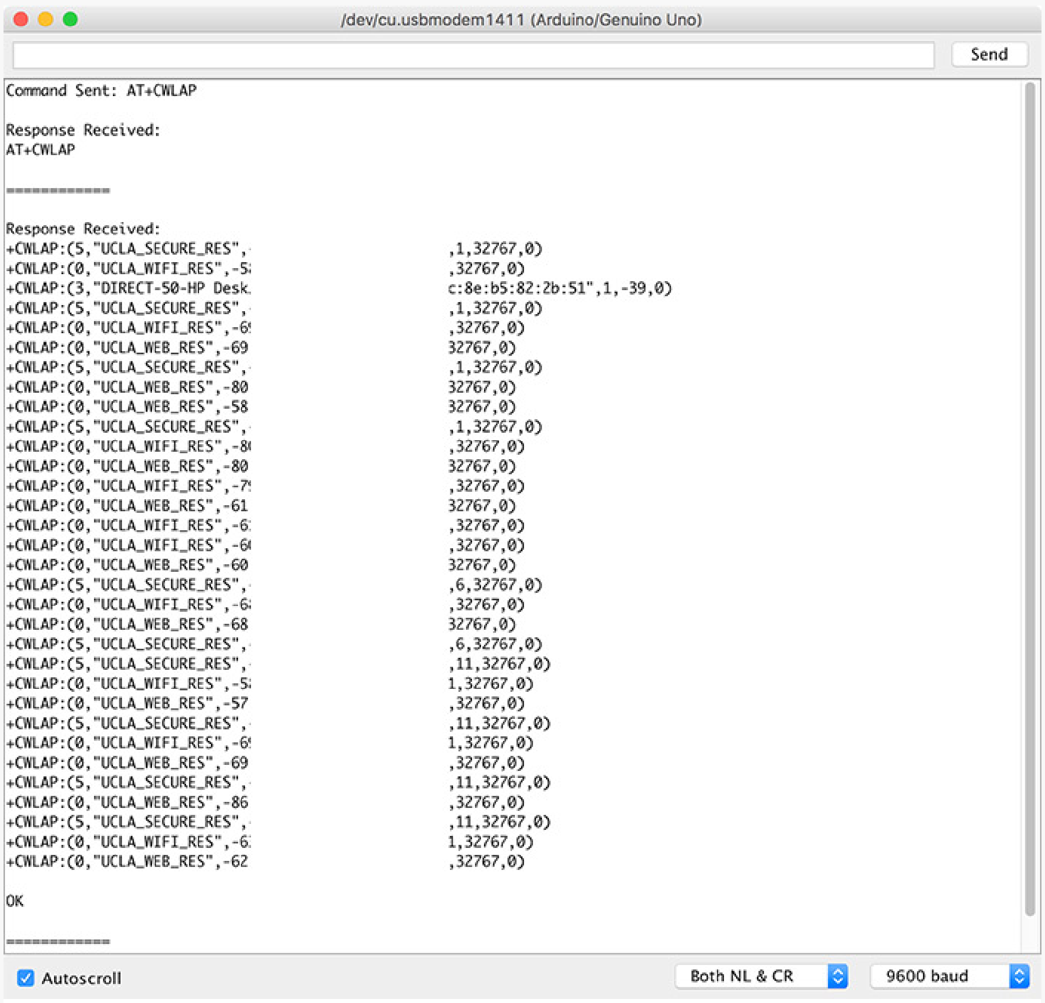
After you have made all the hardware connections, upload the following sketch to the Arduino. The sketch copies commands typed into the Serial Monitor and sends them to the ESP8266, and displays responses from the ESP8266 in the Serial Monitor window. We will be using the Arduino Serial Monitor both to send commands to the module and to view responses that it sends back! For the very first run of the sketch, you may need to uncomment the block that says “you may need to uncomment this block for the first run.” The ESP8266 modules come pre-set out of the factory at different BAUD rates, depending on the manufacturer. My module came set at 115200 BAUD out of the box. Because we are going to use SoftwareSerial to communicate with the module, we can slow the BAUD rate down to 9600 to ensure reliable communication. That setting is enforced with the AT+IPR=9600 command sent to the device (keep reading to know what this means!)

```
#include <SoftwareSerial.h>

const byte rxPin = 2; // Wire this to Tx Pin of ESP8266
const byte txPin = 3; // Wire this to Rx Pin of ESP8266

// We'll use a software serial interface to connect to ESP8266
SoftwareSerial ESP8266 (rxPin, txPin);

void setup() {
  Serial.begin(9600);

// You may need to uncomment this block for the first run:
//  ESP8266.begin(115200); // Change this to the factory baudrate used by ESP8266
//  delay(1000);
//
//  Serial.println("Setting BAUDRATE to 9600");
//  ESP8266.println("AT+IPR=9600");
//
  ESP8266.begin(9600);

}
bool okReceived = false;

void loop() {

  if (Serial.available() > 0)
  {
    String command = Serial.readStringUntil('\n');
    Serial.println("Command Sent: " + command);
    Serial.println();
    ESP8266.println(command);
  }

  int responseCounter = 0;
  if (ESP8266.available() > 0)
  {
    while (ESP8266.available() > 0)
    {
      if (responseCounter == 0)
      {
        Serial.println("Response Received:");
      }

      String response = ESP8266.readStringUntil('\n');
      Serial.println(response);
      responseCounter++;
    }
    Serial.println();
    Serial.println("============");
    Serial.println();
  }


}
```

Okay, so we've connected the ESP8266 module to the Arduino. Now we can view and configure settings on the ESP8266 as we like. To do this, we must send a specific string command that the module can interpret and respond to. The string commands that we send are called "AT commands" because they all begin with the string "AT" and because they call *ATtention* to the device. A full list of all the AT commands you can use with the ESP8266 module can be found here: https://cdn.sparkfun.com/assets/learn_tutorials/4/0/3/4A-ESP8266__AT_Instruction_Set__EN_v0.30.pdf

I'll focus on a couple of examples for this tutorial. You can configure the rest of your ESP8266's settings by referencing the AT commands in the documentation!

The first command to run is "**AT+CWLAP**", the command to scan and list the wifi networks available to join. Typing in the string into the Serial Monitor and hitting enter produces this result!

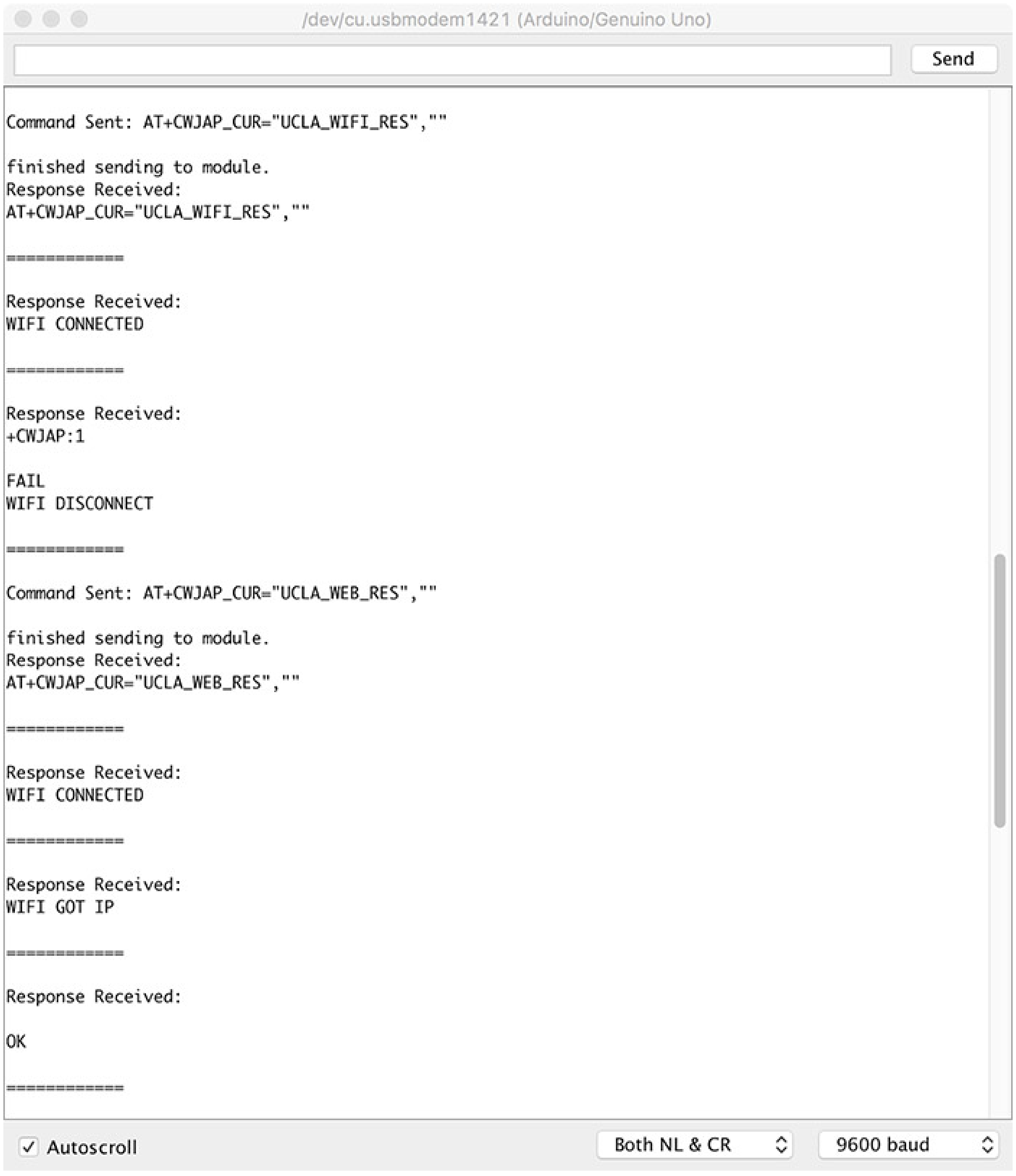


```
Command Sent: AT+CWLAP

Response Received:
AT+CWLAP

============

Response Received:
+CWLAP:(5,"UCLA_SECURE_RES",                      ,1,32767,0)
+CWLAP:(0,"UCLA_WIFI_RES",-5                      ,32767,0)
+CWLAP:(3,"DIRECT-50-HP Desk                      c:8e:b5:82:2b:51",1,-39,0)
+CWLAP:(5,"UCLA_SECURE_RES",                      ,1,32767,0)
+CWLAP:(0,"UCLA_WIFI_RES",-6                      ,32767,0)
+CWLAP:(0,"UCLA_WEB_RES",-69                      32767,0)
+CWLAP:(5,"UCLA_SECURE_RES",                      ,1,32767,0)
+CWLAP:(0,"UCLA_WEB_RES",-80                      32767,0)
+CWLAP:(0,"UCLA_WEB_RES",-58                      32767,0)
+CWLAP:(5,"UCLA_SECURE_RES",                      ,1,32767,0)
+CWLAP:(0,"UCLA_WIFI_RES",-8                      ,32767,0)
+CWLAP:(0,"UCLA_WEB_RES",-80                      32767,0)
+CWLAP:(0,"UCLA_WIFI_RES",-7                      ,32767,0)
+CWLAP:(0,"UCLA_WEB_RES",-61                      32767,0)
+CWLAP:(0,"UCLA_WIFI_RES",-6                      ,32767,0)
+CWLAP:(0,"UCLA_WIFI_RES",-6                      ,32767,0)
+CWLAP:(0,"UCLA_WEB_RES",-60                      32767,0)
+CWLAP:(5,"UCLA_SECURE_RES",                      ,6,32767,0)
+CWLAP:(0,"UCLA_WIFI_RES",-6                      ,32767,0)
+CWLAP:(0,"UCLA_WEB_RES",-68                      32767,0)
+CWLAP:(5,"UCLA_SECURE_RES",                      ,6,32767,0)
+CWLAP:(5,"UCLA_SECURE_RES",                      ,11,32767,0)
+CWLAP:(0,"UCLA_WIFI_RES",-5                      1,32767,0)
+CWLAP:(0,"UCLA_WEB_RES",-57                      ,32767,0)
+CWLAP:(5,"UCLA_SECURE_RES",                      ,11,32767,0)
+CWLAP:(0,"UCLA_WIFI_RES",-6                      1,32767,0)
+CWLAP:(0,"UCLA_WEB_RES",-69                      ,32767,0)
+CWLAP:(5,"UCLA_SECURE_RES",                      ,11,32767,0)
+CWLAP:(0,"UCLA_WEB_RES",-86                      ,32767,0)
+CWLAP:(5,"UCLA_SECURE_RES",                      ,11,32767,0)
+CWLAP:(0,"UCLA_WIFI_RES",-6                      1,32767,0)
+CWLAP:(0,"UCLA_WEB_RES",-62                      ,32767,0)

OK

============
```

Figure 4. “AT+CWLAP” command in Serial Monitor

I’ve blanked out the IP Addresses for the networks that the ESP8266 listed, but as you can see, the device finds several networks and lists their information neatly in rows. This will be useful when we want to connect to a specific network! We can do so through the command “**AT+CWJAP_CUR**”.

In the Serial Monitor, type in AT+CWJAP_CUR, followed by the SSID of the network you want to join in double quotes, followed by the password to that network in double quotes. The picture below shows a failed attempt followed by a successful one.



```
Command Sent: AT+CWJAP_CUR="UCLA_WIFI_RES",""

finished sending to module.
Response Received:
AT+CWJAP_CUR="UCLA_WIFI_RES",""

============

Response Received:
WIFI CONNECTED

============

Response Received:
+CWJAP:1

FAIL
WIFI DISCONNECT

============

Command Sent: AT+CWJAP_CUR="UCLA_WEB_RES",""

finished sending to module.
Response Received:
AT+CWJAP_CUR="UCLA_WEB_RES",""

============

Response Received:
WIFI CONNECTED

============

Response Received:
WIFI GOT IP

============

Response Received:

OK

============
```

✓ Autoscroll   Both NL & CR   9600 baud

Great work! Now that you successfully have your ESP8266 module running and communicating with both the WiFi networks and your Arduino, you can send it some other configuration commands as well. The documentation provides a detailed list of all the commands you can send the module. In the next tutorial, we'll show you how to use the ESP8266 module and the commands you just learned in a complete Arduino project!

5 Comments www.deviceplus.com 1 Login

Recommend Share Sort by Best

Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS ?

Name

EM Farih • 10 days ago
10k or 1k? I use 1k, but still it didn't work. Sometimes it print with random character..
• Reply • Share ›

Ziv Hadad • 5 months ago
Hi!
So it didn't work for me...
So i have wanted to ask, since i'm good at programming but still only beginning to understand how the electronics work here, is connecting 2 Arduino 3.3V outputs on a serial connection to the Esp8266 going to produce enough power? or is it not working this way?
Thanks in advance!
• Reply • Share ›

Joao Felipe • 5 months ago
Parabens.
Ótimo trabalho.
• Reply • Share ›

Peterson de Aquino • 9 months ago
Hi thanks for you post! Can you send me more informations about ESP8266? Exemplos or tutorial because my project don´t work! thanks
• Reply • Share ›

Nick Nick • 9 months ago
Great Tuto! Very clear and usefull
• Reply • Share ›

ALSO ON WWW.DEVICEPLUS.COM

How Does the Disney Snapbot Work? | Device Plus
1 comment • 3 months ago
Abhimanyu Singh — I would like to make one for our project, how can I procure the parts to assemble together, please contact me on my …

Using ESP-WROOM-02 Wifi Module As Arduino MCU
1 comment • 4 months ago
Sidharth Das — Hi there, really interesting DIY. I found your post while looking for a chip with WiFi and BLE. I have a question, I hope you would …

Arduino Long Range Comm. - LoRaLib library | Device Plus
60 comments • 10 months ago
Avatar Jan Gromes — Then you have installed an older version of the library. Install the latest version (1.0.1).

DIY Raspberry Pi 3 Smart Picture Frame - PT 2 | Device Plus
3 comments • a year ago
Avatar Purza — Great project!

Subscribe Add Disqus to your siteAdd DisqusAdd Disqus' Privacy PolicyPrivacy PolicyPrivacy

Rahul Iyer
Studying Electrical Engineering at UCLA, Rahul loves to work on electronics and robotics projects as a hobby. He is especially enthusiastic about electric vehicle technology and assistive robotics.

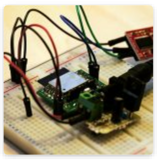
Check us out on Social Media

Facebook Twitter Linkedin

Reddit Google+ Follow on Feedly

Recommended Posts

ESP-WROOM-02 Setup Guide


ESP-WROOM-02 Wifi Setup Guide – AT Commands

How To Make Smart Home Electronics: A Smart Mailbox

Arduino Long Range Communication Tutorial – LoRaLib Library

DIY Arduino Home Security System using ROHM Sensor Kit Part 2 – Cayenne Setup

Using ESP-WROOM-02 Wifi Module As Arduino MCU

Receive update on new posts
Privacy Policy
Submit

About | Company | Privacy Policy | Terms of Service | Contact

Don't Forget to Follow Us!