23-March-2018

**ESP8266-01 common problems.**

**POWER (Vcc and CH\_PD) Must be 3.3V. Make sure it is not 5V! You will kill the module.**

**Use a dedicated power source that can output 3.3V e.g. the breadboard power modules, which use the LM1117 3.3 Voltage regulator (mine has an “AMS1117-3.3” which is equivalent.**

*Keep in mind: 97% of circuit building errors come from wring wiring. The more circuits you build the lower this problem becomes. You will learn from your mistakes. If you still have errors, a complete circuit rebuild may do the job.*

The ESP’s CH\_PD or just CHPD pin (chip Power down) must be connected to 3.3V also \*I suggest using a 10K resistor to connect CHPD to 5V in case CHPD draws current, so you can minimise current ‘loss’ using the resistor instead of a plain jumper wire) but you can just use a jumper wire to the 3.3V line if necessary.

**GROUND**. Make sure you connect the Ground pin to the same ground as all other devices and power supplies in your system (inc. an Arduino if you are using one).

If you want to flash. Power up the ESP with GPIO0 connected to ground. Once power-up is complete remove the ground connection to GPIO0 (this will stop problems with operation later).

If you are using an Arduino to connect between your PC and the ESP, make sure the Arduino has no ‘active sketch’ in it. Remove old sketches by uploading a blank sketch.

Make sure you don’t have the serial port open as sometimes happens when working with other sketches etc. So close ALL serial communications programs to ‘free up’ the Serial Comms channel.

Again, if you are using an Arduino to connect between your PC and the ESP, make sure you have the right PORT selected. The

Make sure you have the correct BAUD rate in your Serial Comms program e.g. Serial Monitor in the Arduino IDE, or PUTTY, or CoolTerm, or Terminal etc…. Common bauds are 115200, older ones are 57600, 9600. The Baud rate of 74880 often gives a “status report” for the ESP but may not be the BAUD which is needed for sending and receiving data to and from the ESP like what’s done when using AT commands.

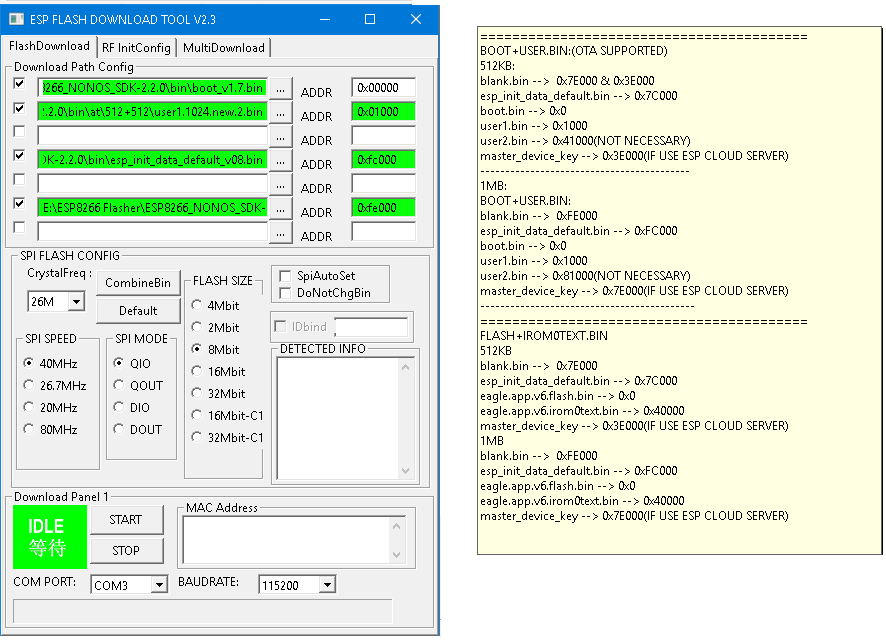
There are two ESP8266’s. One with 512KB (half a MegaBytes, 4.096Megabits) which is usually blue coloured, and one that has 1MB (8 Megabits). If you are going to flash the ESP, then make sure you get the appropriate binary files (.bin) and correct memory addresses to put these bin files into.

If flashing the ESP-01, make sure the Crystal Frequency is set to 26MHz, while the SPI frequency is set to 40MHz.

When trying to flash, the COM port is open somewhere (often by another program, e.g. Arduino IDE Serial Monitor window is open). Close all COM ports before flashing. Sounds like the advice above but it’s easy to forget this when flashing as opposed to just normal operation.

If you are using an Arduino as a “go between” from the computer to your ESP, make sure you have the RX and TX lines connected properly. Pin 0 on the Arduino is called 0 -> RX. This is means pin 0 on the Arduino goes to the RX of the other device. I.e. pin 0 of the Arduino is the Arduino's TX.  
This TX signal comes out at 5V !!. Using a jumper make this line go to a 2.2K resistor. At the other end of the resistor have a 4.7K resistor going to ground AND have the end of the 2.2K resistor go to the RX of the other device (the ESP). This is a voltage divider and steps down the voltage from 5V to about 3.2V which is close enough to 3.3V.  Pin 1 of the Arduino says 1 -> TX. This means pin 1 of the Arduino goes to the TX of the other device, i.e. the Arduino pin 1 is its RX line (it takes in the other devices TX signals). The ESP gives its TX signals at 3.3. which the Arduino can read so there is no need to adjust those voltages.

Flash settings….



Or you can use the “simple flash” tool and a combined binary file to do it, which is included in MichaelAllanGit github depository. I recommend this.