Flash ESP8266 WiFi module

Virgile Neu

May 27, 2017
version 1.0

1 Material needed

In order to flash the ESP8266 Module, you will need :

- A ESP8266 Module,
- A USB-UART cable with `GND`, `Rx` and `Tx`,
- The given archive `esptool-master`,
- A `DE0_nano SoC` board from Altera with the custom extension board from the LAP,
- Three female-female cables for the board,
- A computer with a GNU/Linux distribution with Python2 installed and at least 1 USB port.

2 setup

Here are the ports of the module for connecting the cables :

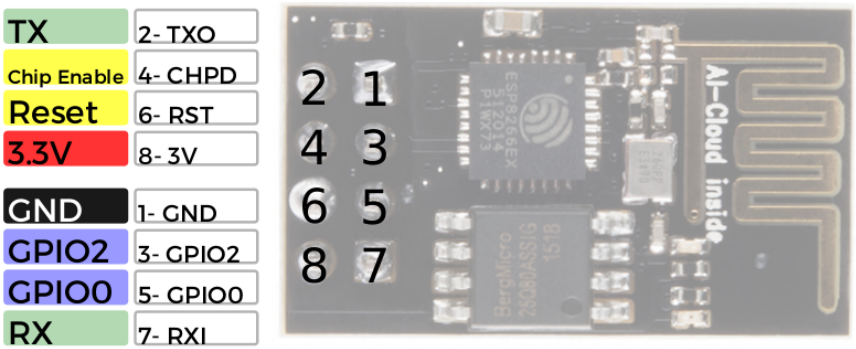

Figure 1: ESP8266 pinout guide.

1. Connect the USB-UART cable to the module : The `GND` (black) on the pin1, the `Rx` (yellow) on the `Tx` and the Tx (Orange) on the Rx of the module. see figure 2.
2. Plug the USB end of the cable into a USB port of the computer.
3. Power on and enable the board : Take two female-female cables, connect them to two 3V3 pins, one is the default `ESP8266_VCC`, the other can be `VCC3P3` on the Arduino Extension. see figure 3.

4. Connect the two cables to the `3.3V` (pin 8) and the `Chip Enable` (pin 4) of the module. A red led should be on.

5. Pull-down the `GPIO0` pin of the ESP8266 (pin 5) : take a third cable, connect it to a ground on the board.

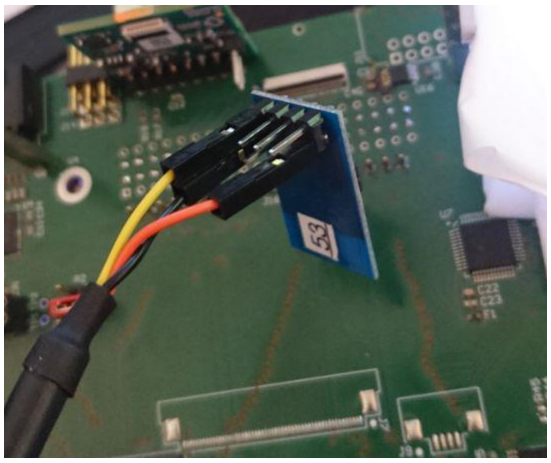

Figure 2: Connection of the USB-UART cable.

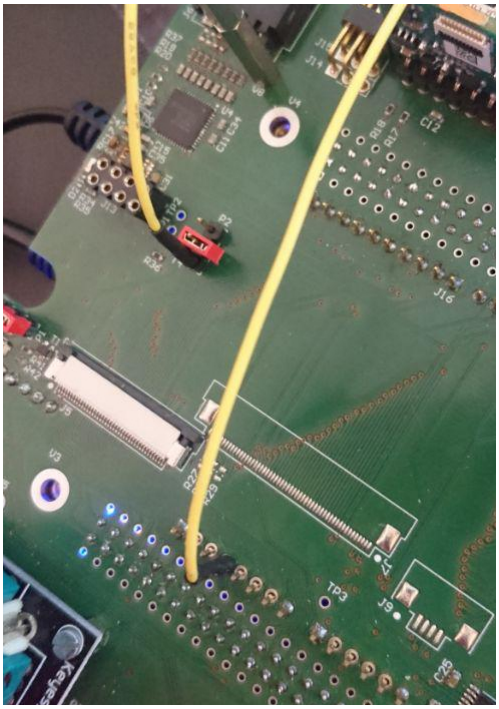

Figure 3: Connection of the two female-female cables to 3.3 volts sources.

You should have the following setup :

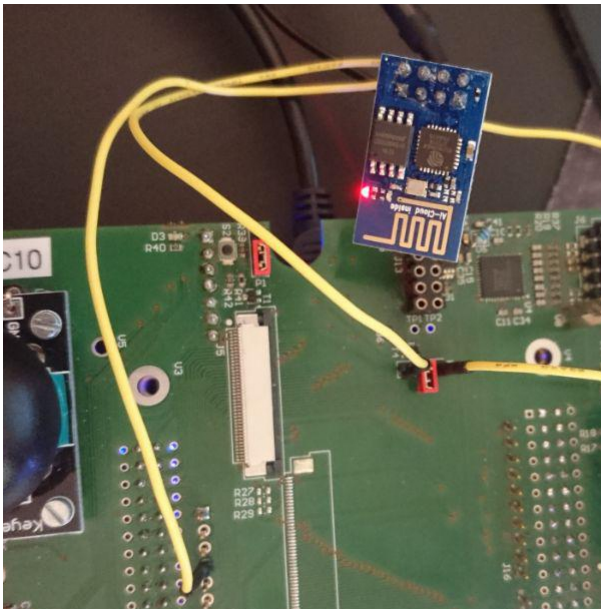

Figure 4: Complete setup for flashing the module.

Now it is time to upgrade the Firmware. We assume the USB-UART serial cable is identified as `\dev\ttyUSB0`.

1. unzip the archive : `$ unzip esptool-master.zip`,
2. go to the extrated folder : `$ cd esptool-master`,
3. run the program the with flash memory parameters already located in the folder :
   ```
   $ ./esptool.py -p /dev/ttyUSB0 write_flash 0x0000 \
   boot_v1.7.bin 0x01000 user1.1024.new.2.bin 0x7C000 \
   esp_init_data_default.bin 0x7E000 blank.bin
   ```

You should see the following output on the terminal :

```
                        esptool-master$ ./esptool.py -p /dev/ttyUSB0 write_flash 0x0000 boot_
v1.7.bin 0x01000 user1.1024.new.2.bin 0x7C000 esp_init_data_default.bin 0x7E000 blank.bin
esptool.py v2.0-beta3
Connecting....
Detecting chip type... ESP8266
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 1MB
Flash params set to 0x0020
Compressed 4080 bytes to 2936...
Wrote 4080 bytes (2936 compressed) at 0x00000000 in 0.3 seconds (effective 120.0 kbit/s)...
Hash of data verified.
Compressed 427060 bytes to 305755...
Wrote 427060 bytes (305755 compressed) at 0x00001000 in 27.0 seconds (effective 126.6 kbit/s)...
Hash of data verified.
Compressed 128 bytes to 75...
Wrote 128 bytes (75 compressed) at 0x0007c000 in 0.0 seconds (effective 64.0 kbit/s)...
Hash of data verified.
Compressed 4096 bytes to 26...
Wrote 4096 bytes (26 compressed) at 0x0007e000 in 0.0 seconds (effective 2050.0 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting...
```

Figure 5: Output of the flashing program.

If there is any missing library from Python2, you should install them. In particular, it needs `pyserial2.7`. If the program keep trying to connect without managing, check the connectivity and/or try to unplug both 3.3V cables (`VCC` and `Chip enable`) and then replug them.

3 Testing

Now we want to test if the flash was successful. Unplug the wire to the GPIO0 pin of the module
Using your favourite USB-UART communication tool (here we will use PuTTY), open a serial connection with the following settings :

- Speed (baud) = 115200,
- Data bits = 8,
- Stop bits = 1,
- No Parity or Flow Control.

Now you can try to send any command in the AT commands set.
The commands should end by "\r\n", which can be done by entering 'ˆM'

and ’^J’ (ctrl+’m’ and ctr+’j’).
By default, the module will echo whatever you input on it’s Rx line to the Tx, that’s why you see what you type. It can be turned off by entering the ”ATE0\r\n” command.

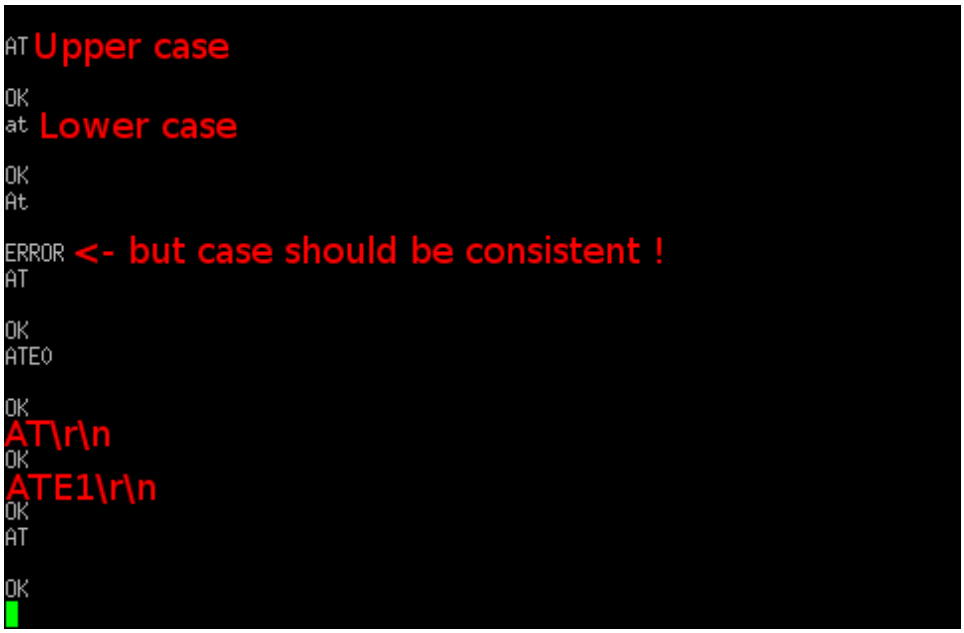

Figure 6: Exemple of test using PuTTY.