WiFi Module HowTo

COEN-4720 Embedded Systems

Cristinel Ababei

Dept. of Electrical and Computer Engineering, Marquette University

1. Objective

The objective of this lab is to get to know the ESP8266 WiFi module.

2. Introduction to WiFi

TODO. See lecture slides.

3. ESP8266 Module

TODO: Clean this up.

The mostly discontinued ESP8266 module [1] is a TTL "Serial to Wireless Internet" device. This module is very low cost, but very simple.
This is the ESP8266, a WiFi SoC (System on a Chip) that can connect to 802.11b/g/n networks on the 2.4GHz band. The ESP8266 behaves like any other wifi client. If your router can handle 128 laptops or 128 cell phones, it can also handle 128 ESP8266s.

It is an impressive, low cost WiFi module suitable for adding WiFi functionality to an existing microcontroller project via a UART serial connection. The module can even be reprogrammed to act as a standalone WiFi connected device–just add power! The feature list is impressive and includes:

- 802.11 b/g/n protocol
- Wi-Fi Direct (P2P), soft-AP
- Integrated TCP/IP protocol stack

It doesn't support SSL, or communication over SPI - just UART!
It doesn't have 5V to 3.3V logic level shifting so you'll probably want to use a logic level shifter if your microcontroller does not have 3.3V capability, such as this one [2].

You could even connect the module directly to your PC if you have a TTL-Serial-to-USB adapter. *Don't try to connect the module to a PC serial port directly, you could cause damage to the module or your computer*. This module requires a 3.3 volt supply for VCC, and 3.3V logic. **It is not 5V tolerant.** Do not connect RX or TX on 5V Arduino to it as that will destroy this module. You must use a logic level converter or a 3.3V Arduino. The 3.3V supply on the Arduino Uno has inadequate current capability to power this module. You must provide a separate, higher 3.3V supply (about 300mA or better). Note that one can find though suggestions of wiring the ESP8266 to an Arduino different ways [5], using a level converter and a different power supply for the module would be safer.

4. Simple testing of the ESP8266 Module

In this experiment, we only connect to the ESP8266 module to check if it is ok. The schematic diagram of this experiment is shown in Fig.1 below. The information here has been adapted from the nice getting-up-to-speed guide from:
http://rancidbacon.com/files/kiwicon8/ESP8266_WiFi_Module_Quick_Start_Guide_v_1.0.4.pdf

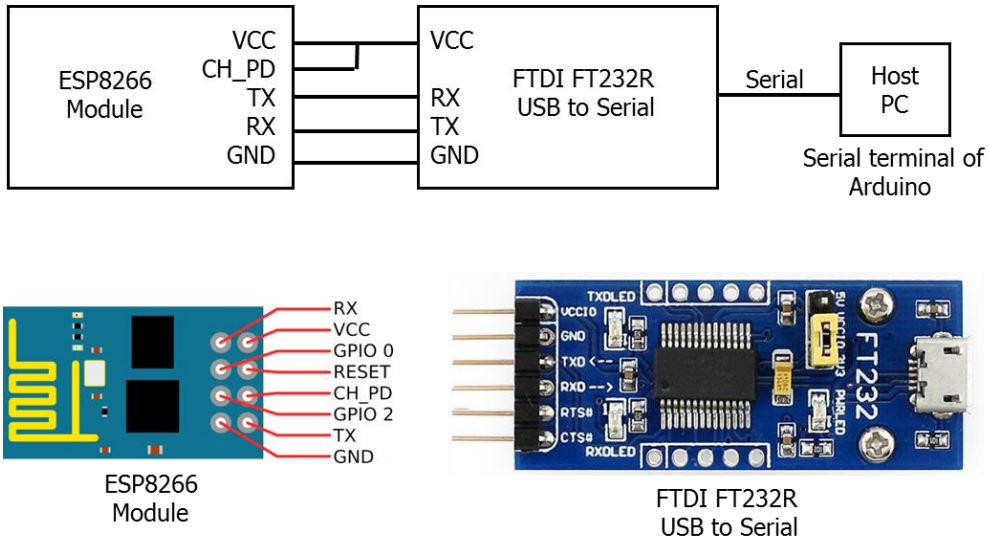

Figure 1 Schematic diagram of 3.3V FTDI board connected to the ESP8266 WiFi module.

We'll connect to the ESP8266 module via a serial connection, which connects our host PC (running a terminal) to the module via a 3.3V FTDI Serial to USB board. In this way, we'll send commands to the ESP8266 module in order to control it. In my case, I use the FTDI board that I got off Amazon, which unfortunately was set by default to work at 5V output levels:
http://www.amazon.com/gp/product/B00RMXM4JI?psc=1&redirect=true&ref_=oh_aui_detailpage_o05_s01
In other words, this FTDI board is not converting voltage levels to 3.3V by default. Because of that, before connecting it to the ESP8266 module, I had to manually scratch the solder-jumper of the back of the FTDI board and solder the jumper so that the 3.3V operation mode is set. This is not a big deal but an annoyance that I had to take care of. To see exactly how that could be done (on a different FTDI board, but in principle it's the same thing), you can read this Instructable:
http://www.instructables.com/id/FTDI-USB-To-Serial-Converter-Guide/step5/Finished/
To avoid all that, you could just buy a smarter FTDI board that had a real jumper that one can set to select between 5V and 3.3V operation modes. One such example is:
http://www.amazon.com/gp/product/B00HSX3CXE?psc=1&redirect=true&ref_=oh_aui_detailpage_o00_s00

Once you wired up the whole thing, it should look like the one in the figure below.

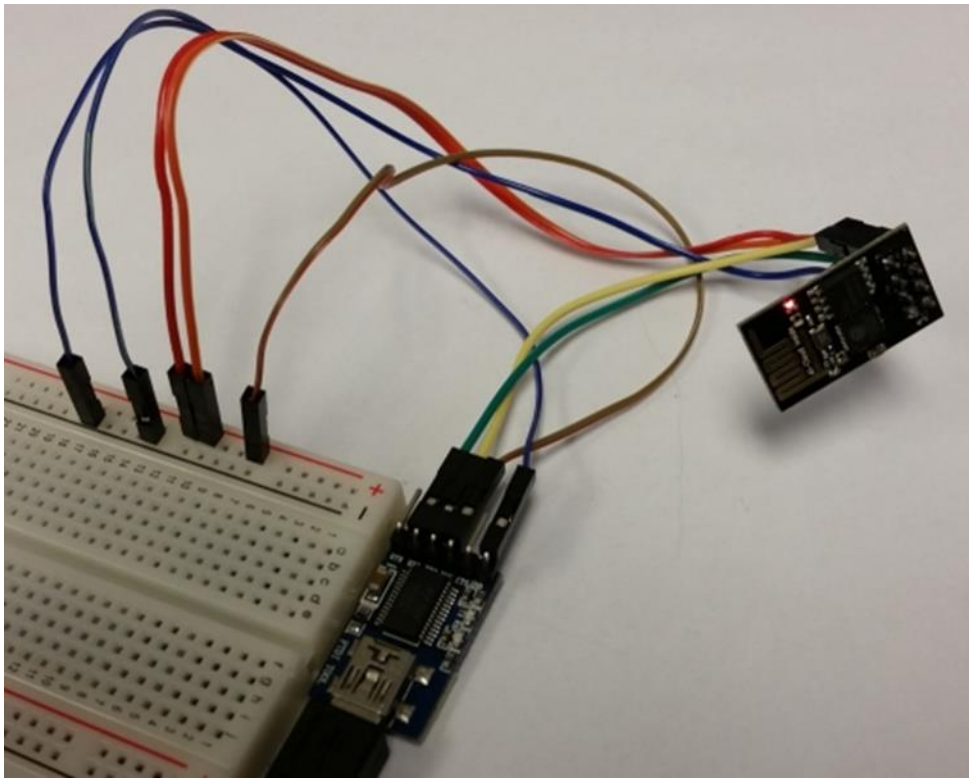

Figure 2: Experimental setup to connect host PC to ESP8266 module via 3.3V FTDI board.

To talk to the ESP8266 module, open up a Serial terminal. My favorite such terminal these days is the **Termite**. It's super neat and easy to use. You can download it for free from here:
http://www.compuphase.com/software_termite.htm
Set it for a baudrate of 115200 and "Both NL & CR". Use the Serial terminal to send commands to and receive replies from the ESP8266 module as described at these resources:
https://cdn.sparkfun.com/assets/learn_tutorials/4/0/3/4A-ESP8266__AT_Instruction_Set__EN_v0.30.pdf
http://rancidbacon.com/files/kiwicon8/ESP8266_WiFi_Module_Quick_Start_Guide_v_1.0.4.pdf
http://www.electrodragon.com/w/index.php?title=Category:ESP8266&redirect=no

Note: *Another Serial terminal option is to use the Serial terminal of the Arduino IDE – in case you already have installed Arduino. Alternatively, you could use other terminals such as Putty or TeraTerm. For the latter, make sure that you do Setup-->Terminal you select New-Line for Transmit as CR+LF. Otherwise, commands sent to the ESP8266 would not be understood by the module. I also tried to use the popular Putty terminal, where I enabled 'Implicit LF in every CR' under Terminal options in order to send both '\r\n'. However, it did not work on my laptop. It might work on yours as it did for other people I read about online.*

Once the terminal is ready you could send first the command:

AT

If everything is all right you should see the ESP8266 module response as:

OK

To check firmware type the command:

AT+GMR

In my case, I got the response:

```
AT version:0.40.0.0(Aug  8 2015 14:45:58)
SDK version:1.3.0
Ai-Thinker Technology Co.,Ltd.
Build:1.3.0.2 Sep 11 2015 11:48:04
OK
```

Next, let's use the command AT+CWMODE, which must be used with an integer value of 1 (=station), 2 (=AP or access point), or 3 (=station + AP). If for example, we wanted to connect the ESP8266 module to our home network, and have it report back the IP address it had been assigned, then, it suffices to configure the module as a "station" only using the command:

AT+CWMODE=1

In my case, I got the response:

```
OK
```

Now, let’s query the local access points (i.e., WiFi networks around) with this command:

AT+CWLAP

In my case, I got the response:

```
+CWLAP:(4,"Wi believe I can Fi",-81,"00:24:b2:89:18:66",1,5)
+CWLAP:(0,"HP-Print-94-LaserJet Pro MFP",-87,"40:b8:9a:36:20:94",1,-17)
+CWLAP:(3,"MyCharterWiFi29-2G",-92,"80:37:73:eb:cb:29",1,3)
+CWLAP:(3,"OshkoshWolf",-56,"10:0d:7f:74:d7:14",6,-9)
+CWLAP:(4,"belkin.fa6",-88,"08:86:3b:3b:6f:a6",6,10)
+CWLAP:(3,"DIRECT-roku-1AD7BD",-68,"b8:3e:59:e6:85:c5",6,-9)
+CWLAP:(3,"NETGEAR02",-89,"04:a1:51:e1:d8:07",7,28)
+CWLAP:(1,"2WIRE387",-78,"b0:e7:54:0d:d9:39",9,-46)
+CWLAP:(3,"NETGEAR07",-84,"28:c6:8e:9d:c0:02",11,11)
+CWLAP:(4,"mikeyp",-89,"00:1d:d5:32:d8:30",11,-27)
OK
```

Let’s connect to the wireless router, which in my case is OshkoshWolf, using the command:

AT+CWJAP="OshkoshWolf","ThePassword"

In my case, I got the response:

```
WIFI CONNECTED
WIFI GOT IP
OK
```

To find the actual IP address that was assigned to the ESP8266 module, use the command:

AT+CIFSR

In my case, I got the response:

```
+CIFSR:STAIP,"192.168.1.16"
+CIFSR:STAMAC,"5c:cf:7f:11:e3:77"
OK
```

To double check that, I logged in into the router (opening http://www.routerlogin.net/start.htm) and checked the currently attached devices. As shown in the figure below, the ESP8266 module with the device name ESP_11E377 is connected and assigned indeed the above IP address.

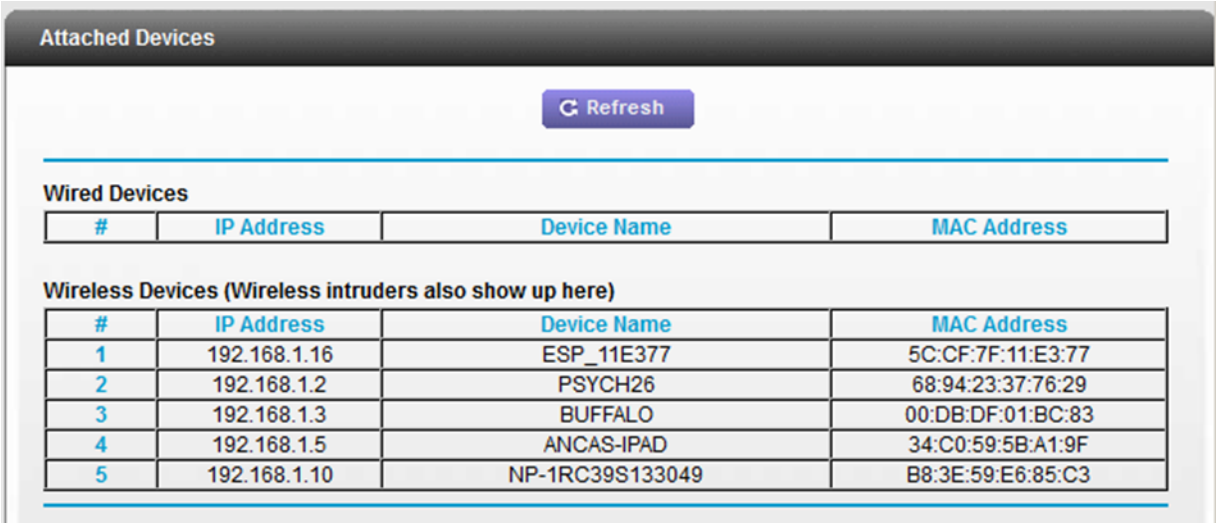

Figure 3 Router admin shows the ESP8266 module as the first attached device.

Now, you try to send some other commands that the ESP8266 module understand. A list of such commands can be found here:
https://cdn.sparkfun.com/assets/learn_tutorials/4/0/3/4A-ESP8266__AT_Instruction_Set__EN_v0.30.pdf
https://github.com/espressif/esp8266_at/wiki

Note (very important): *The ESP8266 module uses up to 200mA at 3.3V. It's possible that the USB converter used above may not be able to supply that. Because of that the ESP8266 module may respond with gibberish answers. To solve this issue, a separate 3.3V power supply must be used to power the module. Here, are some nice resources regarding this:*
https://www.youtube.com/watch?v=qU76yWHeQuw

5. ESP8266 Firmware Update

If for whatever reason you want to update the firmware of the ESP8266 module, then, here I describe how I did it. It must be noted that this may be a frustrating process as there are different ways to do it (according to various people out there) but most of the times those ways do not seem to work, at least not for me. If your ESP8266 works fine, then I would say that you should not go for such updates to spare yourself the pain of messing the module up and not being able to use it anymore. Do it only if you have spare modules and you wanna play.

First, I wanted to list the methods I tried and did not work. Some of these methods have been discussed on this nice webpage:
http://www.electrodragon.com/w/index.php?title=Category:ESP8266&redirect=no

Perhaps some of these you already found online in other places and tried them yourself. They may or may not work. Each of the methods discussed here, need an experimental setup that connects the ESP8266 module to the host PC via the 3.3V FTDI Serial to USB board as done in the previous section, **however, with one extra thing: the GPIO0 pin of the ESP8266 module must be connected to ground during the firmware upgrade!** So, here are first the three methods to upgrade firmware that did not work for me:
Method 1: Using the Espressif's official esp8266_flasher.exe program. I also downloaded the v0.9.2.2 AT Firmware.bin file, as described here:
http://www.xess.com/blog/esp8266-reflash/
http://www.instructables.com/id/Intro-Esp-8266-firmware-update/
Method 2: Using the nodemcu-flasher, which can be downloaded from here:
https://github.com/nodemcu/nodemcu-flasher
And done as described here for example:
http://blog.randypatterson.com/esp8266-firmware-updates-and-options/
Method 3: Using XTCOM utility, which can be downloaded from here:
http://iot-playground.com/swdownload/xtcom_util.zip
And done, as described here:
http://iot-playground.com/2-uncategorised/35-esp8266-firmware-update (has nice buying guide!)

Finally, here is the method that worked for me.
Method 4: Using Python tool esptool.py, which is meant to be used as an alternative to the XTCOM tool (discussed earlier).
Step1: Download the whole esptool master folder from here:
https://github.com/themadinventor/esptool
I placed a copy of the esptool-master tool folder as:
M:\ESP8266\esptool-master
Step2: I also made two changes to the file esptool.py. First, I replaced the lines:

```
ESP_RAM_BLOCK   = 0x1800
ESP_FLASH_BLOCK = 0x400
```

With these two lines (based on some suggestions found online):

```
ESP_RAM_BLOCK   = 0x180
ESP_FLASH_BLOCK = 0x40
```

Second, I changed the following function:

```
def __init__(self, port=0, baud = ESP_ROM_BAUD):
    self._port = serial.Serial(port)
    # setting baud rate in a separate step is a workaround for
    # CH341 driver on some Linux versions (this opens at 9600 then
    # sets), shouldn't matter for other platforms/drivers. See
    # https://github.com/themadinventor/esptool/issues/44#issuecomment-107094446
    self._port.baudrate = baud
```

with this:

```
def __init__(self, port=0, baud = ESP_ROM_BAUD):
        self._port = serial.Serial(port='COM2', baudrate=115200) # open COM2;
```

in order to hardcode the serial port to COM2 and the baudrate to 115200. In my case it is COM2 because this is how the FTDI Serial to USB board was assigned once plugged it into my laptop (Windows 7 Professional).

Step3: Download a copy of the binary firmware you want to use for your upgrade. While, you can find different versions of the firmware online, I have downloaded the one from here: https://github.com/JhonControl/ESP8266-Flasher/tree/master/Firmware%20ESP8266/Firmware%20ESP8266%20V0.9.5.2_AT
In my case, I downloaded "v0.9.5.2 AT Firmware.bin" and renamed it as "v0_9_5_2ATFirmware.bin" because file names that include blanks on Windows is always trouble, which you want to avoid. I placed this binary into:
M:\ESP8266\esptool-master\v0_9_5_2ATFirmware.bin
Step4: Install Python2.7 on your Windows machine if you do not have it yet. Here, I'll let you figure it out how to do it.
Step5: Open a cmd terminal and go to M:\ESP8266\esptool-master, where we can run the esptool.py Python script like this:
> C:\Python27\python.exe esptool.py write_flash 0x00000 v0_9_5_2ATFirmware.bin
If everything is ok, then you should see something like this:
> Connecting…
> Erasing flash…
> Wrote 520192 bytes at 0x00000000 in 129.8 seconds <32.1 kbit/s>…
> Leaving…
If you get a problem, then, the most likely outcome will be:
> Connecting…
> Erasing flash…
> Writing at 0x00000000… <6%>
> A fatal error occurred: Invalid head of packed
I got it too the first time, but at the second attempt it worked fine!
Step6: If all ok, disconnect the USB cable and disconnect pin GPIO0 of the ESP8266 module from ground. Then, check the module as described in the previous section. Send some AT commands and check that everything is fine! Particularly, if you want to change the baudrate at which the ESP8266 module works (default is usually 115200) to say 9600 in order to for example use a SoftwareSerial when playing with Arduino, you can send the AT command:

```
AT+CIOBAUD=9600
```

Note: *The above command works for version v.0.9.5 for example, but not for v1.5.0. For updated AT commands set see for example: https://cdn.sparkfun.com/assets/learn_tutorials/4/0/3/4A-ESP8266__AT_Instruction_Set__EN_v0.30.pdf*
To change the baudrate at which the ESP8266 module communicate in v1.5.0, we should use instead:

```
AT+UART_DEF=9600,8,1,0,0
```

References

[1] ESP8266 WiFi Module at Adafruit (discontinued), https://www.adafruit.com/products/2282
[2] SparkFun Logic Level Converter - Bi-Directional, https://www.sparkfun.com/products/12009
[3] ESP8266 WiFi Module Quick Start Guide, http://rancidbacon.com/files/kiwicon8/ESP8266_WiFi_Module_Quick_Start_Guide_v_1.0.4.pdf
[4] Adafruit FTDI Friend + extras - v1.0, https://www.adafruit.com/products/284
[5] ESP8266 wiring schemas, http://yaab-arduino.blogspot.co.uk/2015/03/esp8266-wiring-schemas.html