Programming (Flashing) and Using the ESP8266

Fernando G. Tinetti*

Technical Report TR-RT-01-2018
III-LIDI, Fac. de Informática, UNLP
CIC, Prov. de Buenos Aires
Argentina
contact e-mail: fernando@info.unlp.edu.ar
November 2018

Abstract. This document is intended to define a step-by-step guide for successful programming of the ESP8266-01 module with an Arduino board and Arduino IDE. There are many Internet sites/blogs/etc. documenting similar tasks, and this report is trying to simplify most of the electrical and interface requirements, while maintaining most of the flexibility of interfacing the ESP8266 for a wide number of possible applications. The main underlying idea is devoting the ESP8266-01 for Wi-Fi networking along with simple HTTP server, and an Arduino board for taking advantage of its large number of applications and already proven flexibility.

1.- Introduction

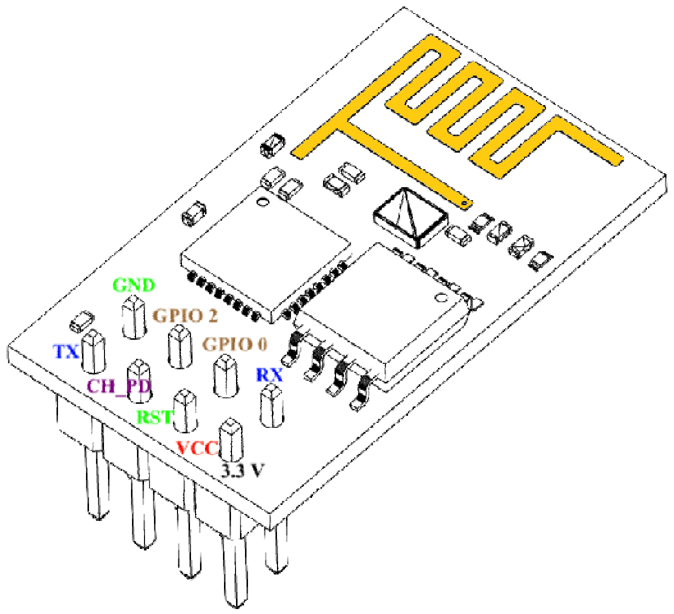
Programming the ESP8266 [6] [3] in its several models (e.g. ESP8266-0x, x = 01, …, 12) [5] is documented in several Internet sites/blogs/user forums. As expected, there is huge heterogeneity depending on models as well as on applications. However, the heterogeneity as well as the successive upgrades made by breakout manufacturer/s has led to confusing and out of date information. The focus of this technical report is the ESP8266-01 for various reasons, among which:

- It is the lowest cost model. The underlying idea behind using the ESP8266-01 is as simple as adding Wi-Fi facility to the also low cost Arduino development boards without compromising and/or adapting already designed and used Arduino developments/applications.
- Being the first popular model (or, at least, one of the popular models), there is a large amount of non-accurate/complete and out of date information. Selecting the correct source of information is a huge task, many times including high risks. This report includes usage and information proven to work in real modules available (not on all of them, though).
- More complex ESP8266 based breakouts/boards such as the NodeMCU [9] [4] and those referred to as WEMOS D1 boards [10] are easier to use, but also more expensive and in some specific cases change (some) Arduino defined pin number, assignments, and electrical specifications. Besides, those boards usually have more accurate documentation and usage reports.
- Manufacturers usually include "official" documentation on current designs, and its own development tools, SDK and/or IDE, not always (or never) taking into account the Arduino platform.

Fig. 1 shows the ESP8266-01 breakout model, which basically has the "minimal" pinout required for operation as well as the printed/PCB antenna. In this report, the pinout is basically used for:

- Vcc (3.3v), GND, CH_PD: power and "nomal" operation
- R, Tx: programming/flash or data communication
- GPIO0: for selecting either "UART" or "flash" mode at ESP8266-01 startup

Fig. 1 also shows two of the most common modules available for purchase since several years.



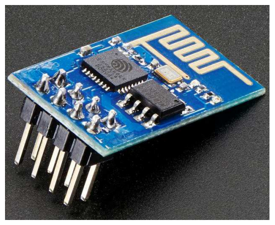
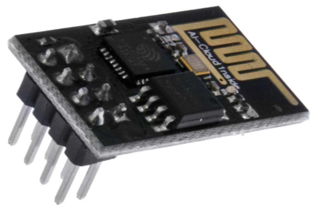



Figure 1: ESP8266-01 Pinout and Commonly Available Modules.

2.- ESP8266-01 Power Supply (and Disclaimer)

There are lots of suggestions, warnings, and specification sources so in this report the documented power connection is the one used for years without any major incident due to the power consumption. Fig. 2 schematically shows the pins used for the ESP8266-01 power supply using an Arduino UNO board, which

- It is not expected to be the best power supply.
- It is not granted to always work and/or under all circumstances/usage.
- It is not guaranteed to be compliant with any hardware specification (neither Arduino board current that can be drawn from the 3.3v nor the ESP8266-01 Vcc).

Warning/Disclaimer: Use the power supply shown in Fig. 1 under your own risk, and take into account that this scheme has worked for several years in very simple development environments and very simple application prototypes. It has not been used in production environments of any application. A capacitor as the one shown in Fig. 2 is repeatedly suggested in many sites due to several ESP8266 power consumption spikes as well as the rather limited 3.3v power source provided by the Arduino boards. The capacitor nominal capacitance specification (e.g. either 1μF or 10μF) depends on the (mostly Internet, non-official) source, though. The work documented in this technical report has been made with 1μF and 10μF nominal capacitance capacitors. There are several suggestions of using a power source independent of Arduino boards, too. The power source in Fig. 1, however,

- It is the one successfully used in all the experiments, as noted above.
- It has not been experimented serious (e.g. causing hardware damage) problems.
- It is possibly working because of the low requirements of the programs running in the ESP8266-01 reported in this document.
- It is the simplest power supply to an ESP8266-01 from an Arduino board, considering the above consideration: the ESP8266-01 is used as a way of providing Wi-Fi communication to an Arduino board and Arduino boards (maybe existing) applications.

The ESP8266-01 Rx-Tx communication pins will be usually connected to the Arduino board pins Rx-Tx, but the way in which they will be connected (e.g. Rx ESP8266 to Rx Arduino board and Tx ESP8266 to Tx Arduino board or vice-versa) will depend on the way in which the Arduino will be used as an interface to the ESP8266-01.

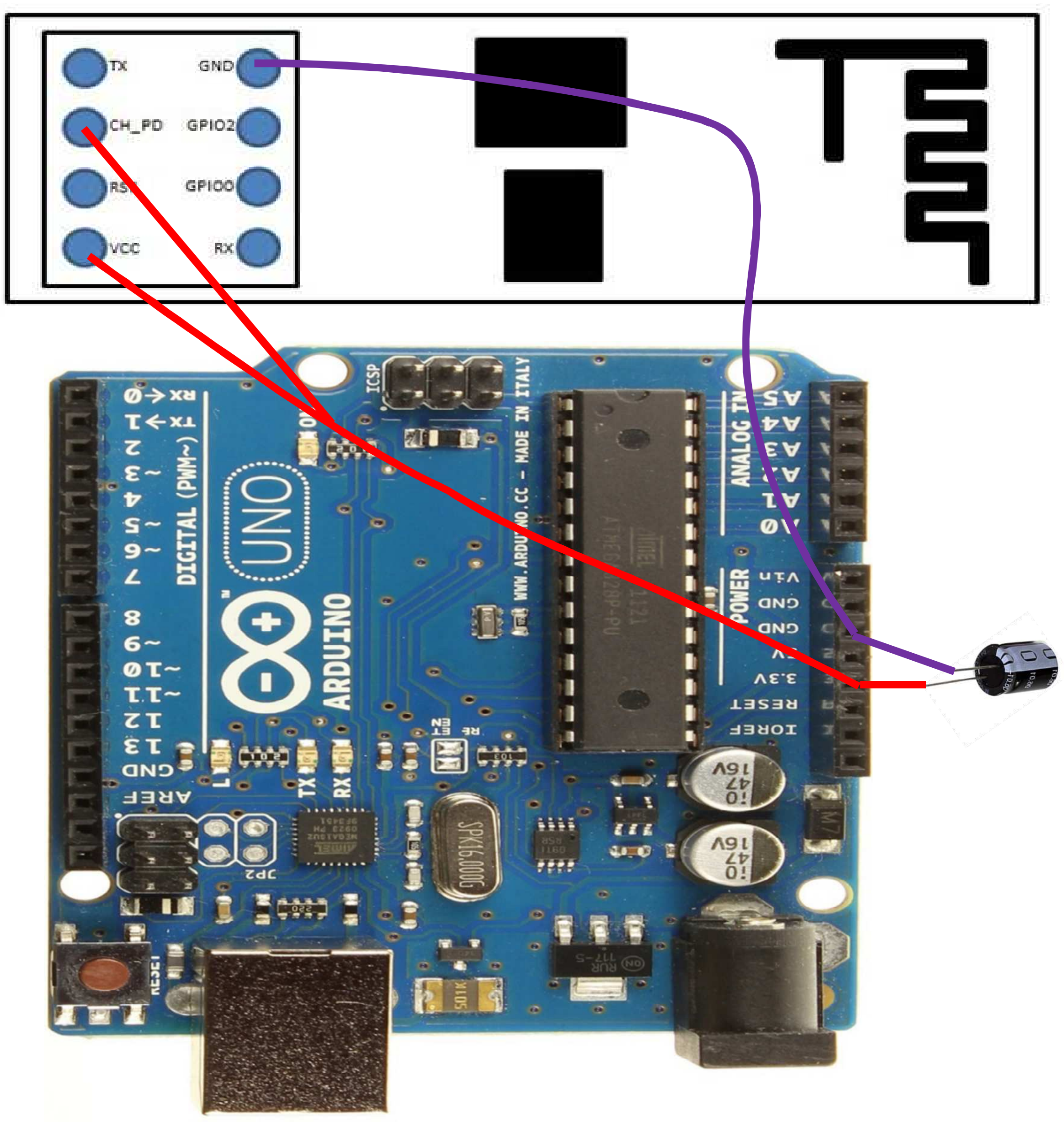


Figure 2: ESP8266-01 Power from Arduino Board.

3.- ESP8266-01 Startup Operation Modes

Depending on how the ESP8266-01 GPIO0 is connected at startup (either connected to GND or not), the module starts operating in either "UART mode" or "flash mode" [8] [2]:

- UART Mode: GPIO0 connected to GND. In this mode, the ESP8266-01 expects to receive the program to flash and execute. The ESP8266-01 startup sequence would be:

1) Receive the program to flash.
2) Execute the recently received and flashed program.

- Flash Mode: GPIO00 not connected. In this mode, the ESP8266 starts executing the program stored in its flash. By default, they are flashed (i.e. as a default factory setting) for being used with "standard" (well-known) AT commands.

The Arduino IDE along with an Arduino board (e.g. Arduino UNO) can be used as an interface in all operating modes. In UART mode, the Arduino board is used to upload the flash contents to the ESP8266-01, and in Flash Mode the Arduino board is used as an interface, for example to send commands to the ESP8266-01 and receive the replies from the module. Furthermore, if the ESP8266-01 contains the factory flash contents for processing AT commands, in Flash mode the user can take advantage of the Arduino board by using

- The Serial Monitor, i.e. interactively sending AT commands by typing them in the Arduino IDE Serial Monitor. This usage is rather limited, mainly to check/experiment with ESP8266 basic operation, because interactive operation of the ESP8266 is limited by itself.
- The Arduino board directly operating with the ESP8266-01 via the Arduino Serial library, i.e. the Arduino board is programmed for sending the AT commands and/or the communications/data expected by the ESP8266-01 in order to (inter)operate.

The serial connection Rx-Tx between the Arduino board and the ESP8266-01 will be determined by the way in which the Flash mode will be used.

4.- Programming (*flashing*) the ESP8266-01 with Arduino

There are two requirements for programming the ESP8266-01:

1) A specific physical connection, defining startup operating mode as explained in the previous section, as "UART mode".
2) The program/environment for sending the flash contents via the serial interface (Rx-Tx ESP8266-01 pins) to be stored in the module.

The physical connections are required by the module itself, and there are several programs/environments for interacting with the module in order to send the flash contents to the module. Since this report is focused in the Arduino Environment, then the physical connection will be made with an Arduino board, and the program for interacting with the ESP8266-01 module will be the Arduino IDE.

Fig. 3 shows the physical connections used for programming (*flashing*) the ESP8266-01 with an Arduino UNO board. Take into account that the power supply connection is the one shown in Fig. 2 above, and there are a few more physical connections, basically for:

- Defining the ESP8266-01 startup mode as "UART mode", by connecting the ESP8266-01 GPIO0 to GND. This "extra" connection of the GPIO0 to GND, is used only for flashing the ESP8266-01 module, and it is sometimes suggested to be implemented via a physical switch. The physical connection used for the experiments in this report is simpler, as will be explained shortly below.
- ESP8266-01 communication (for the ESP8266-01 receiving the contents to be stored in its flash memory), by connecting the ESP8266-01 Rx to the Arduino UNO Rx, and the ESP8266-01 Tx to the Arduino UNO Tx. These physical connections basically make the Arduino UNO a "bridge" among a computer and the ESP8266-01.

Given the simple connections as well as the experimental/development nature of the work in this technical report a PCB has been avoided. Furthermore, it has not been necessary even a breadboard, the EXP8266-01 module is directly connected through cables to the Arduino UNO board.

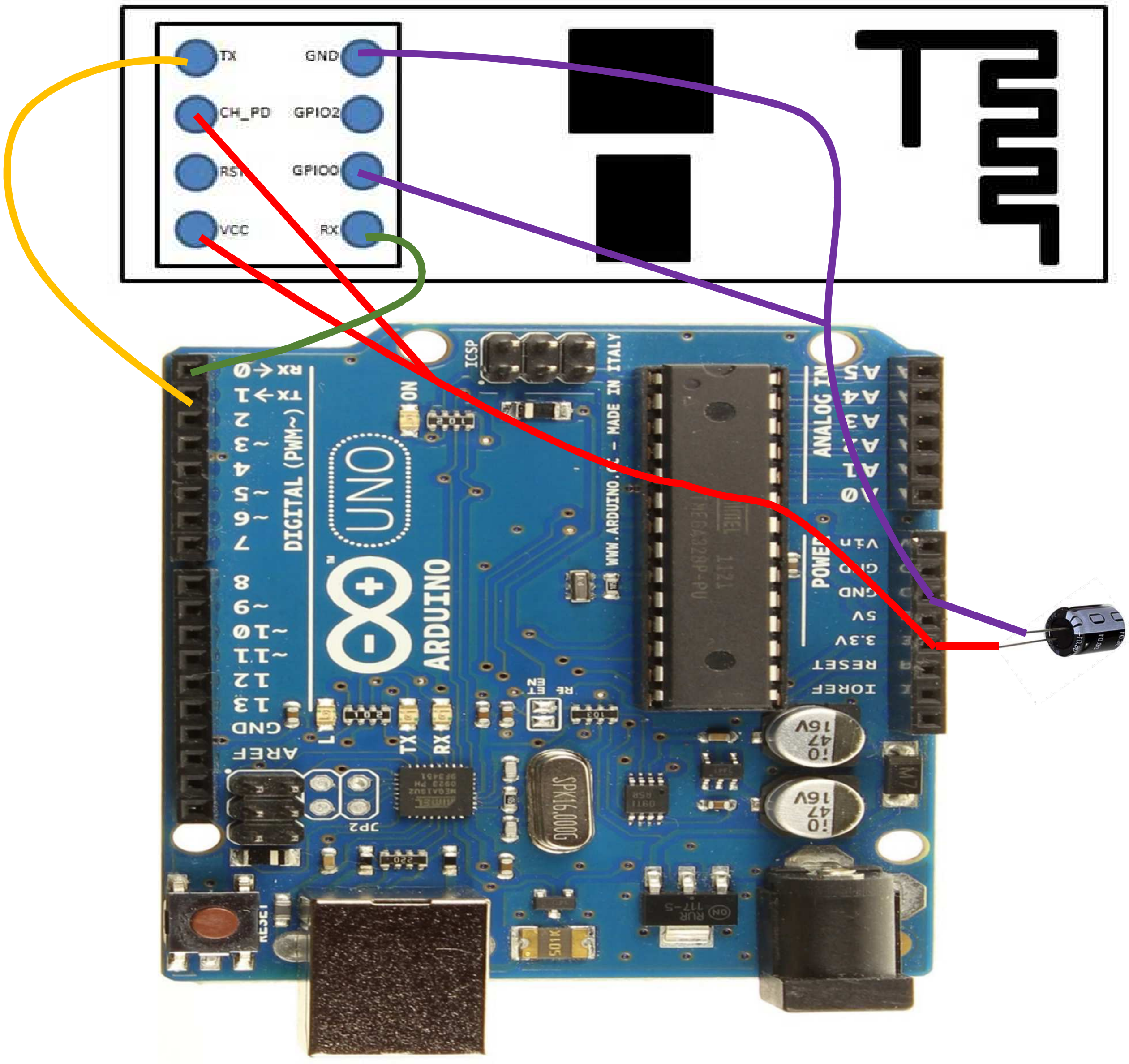


Figure 3: ESP8266-01 Connections for *Flashing*.

Given that the ESP8266-1 power supply is independent of the operation mode and that the GPIO0 is eventually connected to GND for programming (*flashing*) the cable in Fig. 4 (ESP is used as a short term for ESP8266-01) has been constructed and used throughout every experiment reported in this document. Basically, the cable is expected to connect:

- Arduino UNO 3.3v to ESP8266-01 Vcc and CH_PD.
- Arduino UNO GND to ESP8266-01 GND and eventually (when flashing) to ESP8266-01 GPIO0.

And the capacitor always connected as a way of enhancing power supply. Heat shrink has been used to maintain cables and capacitor connections isolated and fixed. Using the so called “dupont cables” (those shown in Fig. 4) provides the flexibility of connecting GPIO0 to GND for flashing as well as disconnect GPIO0 for module startup in “flash mode” and avoids using a breadboard for these simple interconnections of the ESP8266-01 with an Arduino board. The serial pins Rx-Tx are connected using M-F Dupont cables.

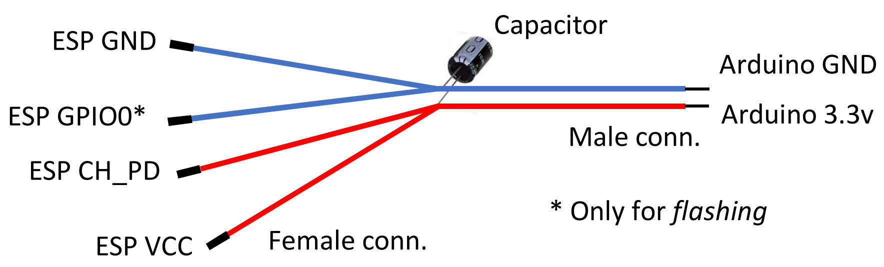


Figure 4: Cabling Used for Arduino <==> ESP8266-01 Connections.

Once the physical connections are defined, it is necessary to use the Arduino IDE for constructing and sending the flash contents to the ESP8266-01. The flash contents should be the firmware the ESP8266-01 will use for execution. The Arduino IDE needs to be configured for generating the firmware from a "standard" Arduino sketch. This is also referred to as "adding the plugin" for the ESP8266 in some Internet pages/tutorials. Beyond details, configuring the Arduino IDE for programming the ESP8266 is among the most stable documentation, and basically is taking advantage of the Arduino IDE board management [1] [7]. The procedure is relatively simple, requires Internet connection, and once successfully completed there are a number of boards for which the Arduino IDE can be used, including the one documented in this report: "Generic ESP8266 Module".

Since the Arduino UNO board will be used only as a "serial port bridge" to the ESP8266-01 it is usually needed that the board does not use the serial port, for avoiding "noise" in the communication between the computer where the Arduino IDE is running and the ESP8266 module. One of the most common ways of avoiding communication noise from the Arduino UNO is just uploading the sketch "BareMinimum" (found in File ==> Examples ==> 01.Basics). Another (more physical) way is just connecting the Arduino UNO Reset pin (which is between the 3.3v and IOREF pins) to GND. Some Arduino boards such as the Arduino Due require the latter to work.

Once the Arduino board is configured so that it does not access the serial pins, turn off the Arduino board and connect the pins as shown in Fig. 3 above before turning on the Arduino once again. The sketch in Fig. 5 will be used as a simple example to run in the ESP8266-01.

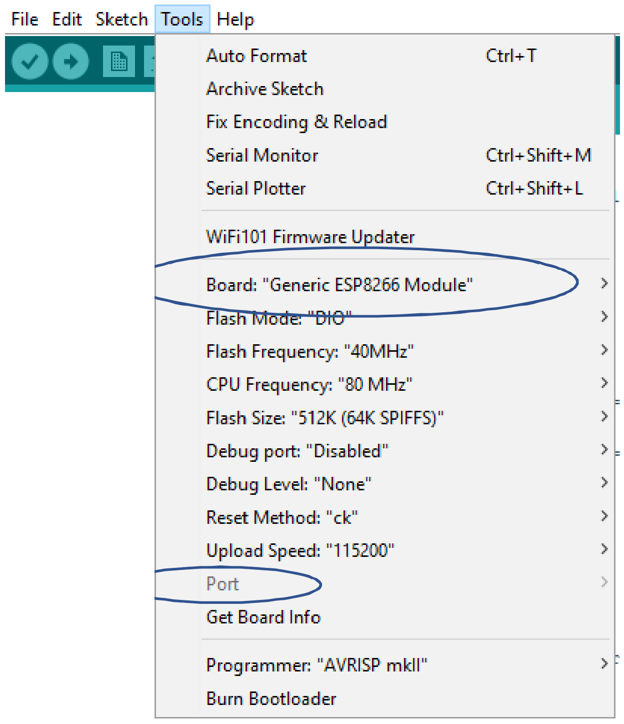
```
#include <ESP8266WebServer.h>

/* Set these to your desired credentials. */
const char *ssid = "simple";

ESP8266WebServer server(80);

void handleRoot(){
  server.send(200, "text/html", "<h1>Reply from ESP8266</h1>");
}

void setup(){
  delay(1000);
  WiFi.softAP(ssid);
  server.on("/", handleRoot);
  server.begin();
}

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

Figure 5: A Simple ESP8266-01 Program Example.

The program (Arduino sketch) example of Fig. 5 defines two main tasks for the ESP8266-01:

1) Configures the Wi Fi operation of the ESP8266-01 as an access point, setting the name of the Wi Fi network as "simple" (without quotes), and without any security (password) access.
2) Sets a (HTTP) server in TCP port 80 (the standard one for HTTP servers), which replies with a very simple HTML content (<h1>Reply from ESP8266</h1>) to every request of "/".

Before uploading the code, select the right board, so that the Arduino IDE will generate the corresponding flash content for the ESP8266-01 module: "Generic ESP8266Module". Fig. 6 shows the complete set of options automatically defined when the Tools ==> Board: "Generic ESP8266Module" is selected in the Arduino IDE.



Board: "Generic ESP8266 Module"
Flash Mode: "DIO"
Flash Frequency: "40MHz"
CPU Frequency: "80 MHz"
Flash Size: "512K (64K SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
Reset Method: "ck"
Upload Speed: "115200"
Port
Programmer: "AVRISP mkII"

Figure 6: Arduino IDE Board Selection for Flashing the ESP8266-01.

Take into account that unlike Fig. 6, the Port should be defined as the port in which the Arduino board is identified by the computer. Once defined the Board and Port, it is possible to upload the Arduino sketch, process that will make to store the program in the ESP8266-01 module flash. If the flash fails, it is recommended to unplug the power source (turn off) of the Arduino board and plug it again, so the Arduino UNO is restarted as well as the ESP8266-01 module. If the upload (flash) fails again, try with a different Upload Speed (115200 has always been used successfully in the experiments reported in this document, though).

Once the sketch is successfully uploaded, it is possible to check that the wi-fi network is "up & running", i.e. every device with a wi-fi interface would be able to

- Find a wi-fi network with name "simple" (without quotes).
- Connect to the "simple" network (thus receiving an IP address via DHCP).
- Make a request from an Internet browser to the IP 192.168.4.1 (the default IP of the ESP8266-01 module), which will reply with the HTML <h1>Reply from ESP8266</h1>. Thus, the browser will show the text "Reply from ESP8266" (without quotes) with HTML header 1 format.

Summarizing, the sequence of steps for uploading the sketch of Fig. 4 above and verify it is effectively working is:

1) Unplug the Arduino board power supply and unplug the ESP8266-01 module.
2) Either
   a. Connect the Arduino board Reset pin to GND
      or
   b. Plug in the power supply Arduino board, load the BareMinimum sketch, and unplug the Arduino board power supply
3) Connect the ESP8266-01 to the Arduino board as shown in Fig. 3 above, and connect the Arduino board power supply
4) Load the "simple" sketch of Fig. 5 above in the Arduino IDE
5) Define the Board (Fig. 6 above) and Port in the Arduino IDE "Tools" menu
6) Upload the sketch
7) Verify that the "simple" wi-fi network is "Up & Running".

If everything is successfully working so far, it is possible to unplug the power supply Arduino UNO and check that the ESP8266-01 is working without having to be flashed again, as explained in the next section.

5.- Using the ESP8266-01 with Arduino

Once the ESP8266-01 has the right program stored in its flash memory, it is available to be used by just connecting the power supply. In general, there are two requirements for an ESP8266-01 module to be used as an "Arduino board wi-fi network":

1) A specific physical connection, defining startup operating mode as explained in the previous section, as "Flash mode".
2) The serial connection will be used for data communication between the Arduino board and the ESP8266-01 module.

The "Flash mode" of operation will make the ESP8266-01 module directly use program in the flash memory for execution. The serial communication will allow the Arduino board and the ESP8266-01 module to exchange data without intervention of any other device/module. Unlike the previous section, the data received in the ESP8266-01 module will be sent directly from the Arduino board, i.e. the Arduino board will not be used as a "serial communication bridge" but will handle every data transference from/to the ESP8266-01 module.

Fig. 7 shows the connections of the ESP8266-01 in order to be used as an Arduino board wi-fi network. Take into account that the power supply connection is the one shown in Fig. 2 above, and

- The GPIO0 pin is not connected, so the ESP8266-01 starts operating in "flash mode" (i.e. use the flash contents as the program to run).
- ESP8266-01 module serial port is connected to Arduino board serial port such as the data sent from the ESP8266-01 module is directly received by the Arduino board, and vice versa. In terms of physical connections, the ESP8266-01 module Rx pin is connected to the Arduino board Tx pin, and the ESP8266-01 module Tx pin is connected to the Arduino board Rx pin.

The sketch in Fig. 5 uploaded to the ESP8266-01 module does not use/need the serial connection for Arduino board (inter)operation, though. Clearly, that sketch is not very useful and does not provide a wi-fi facility to a low cost Arduino board. In cases the Arduino board needs to send and/or receive some data via Wi-Fi it will need the serial communication with the ESP8266-01 module via Rx-Tx pins connected as shown in Fig. 7.

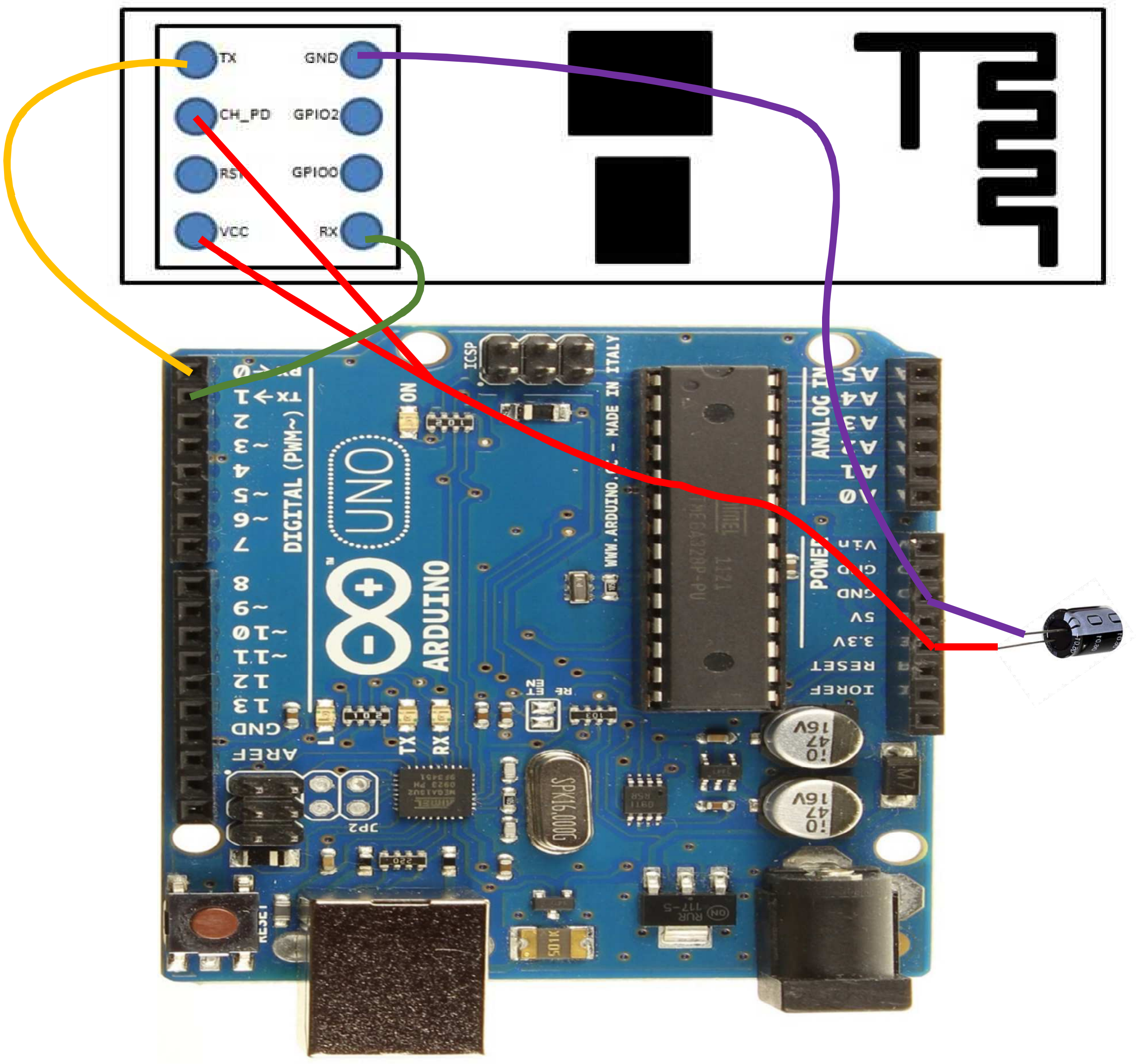


Figure 7: ESP8266-01 Connections for "Standard" (Flash Mode) Operation.

It is possible to verify the ESP8266-01 module runs with the sketch of Fig. 5 already uploaded by following the steps:

1) Unplug power the Arduino board power supply.
2) Connect the ESP8266-01 module as shown in Fig. 7 above.
3) Plug in the Arduino board power supply.
4) Verify that the "simple" wi-fi network is "Up & Running".

In order to successfully use the Serial interface, it has to be configured in both *sides* (the Arduino board and the ESP8266-01 module) with the same speed (e.g. 9600 bauds). One of the advantages of using the Arduino IDE for ESP8266-01 module programming is that many Arduino libraries such as the Serial one are available and can be used without any change/adaptation. Thus, communicating the Arduino board and the ESP8266-01 module would be as simple as:

- Initialize both serial ports using the Serial library and the same baud rate.
- Used the send/write operations of the Serial library for sending data from one platform, and use receive operations of the Serial library in the other, i.e. for each send there should be a corresponding receive.

Also, as in the case of the Serial communication of the Arduino boards with the computer to which it is connected to, it is not necessary to know in advance every data communication, because all of the Serial library is available. Thus, functions for testing serial communication operations can be used, e.g. Serial.available() can be used in the ESP8266-01 module as well as in the Arduino board.

6.- Conclusions and Further Work

This report documents a simple way of programming and the main idea of using an ESP8266-01 module as an Arduino board Wi-Fi facility. The whole process is *reduced* to a few cables connected to the Arduino board, even without the need of a breadboard. Furthermore, the whole process is as simple as

- Define a sketch for the ESP8266-01 module.
- Upload the ESP8266-01 module sketch.
- Define a sketch for the Arduino board.
- Upload the sketch to the Arduino board.

And the Arduino board communications with the ESP8266-01 module is “naturally” handled by the serial physical interconnection (Rx-Tx pins of each platform) along with the well-known Arduino Serial library. There is no need of other software or procedure. Unlike using other ESP8266-based boards, the operation of hardware from the Arduino board remains without any change. The only requirement for adding ESP8266-01 module to existing applications/projects is that the Arduino board serial pins are not already used. Actually, Arduino boards with more than one serial port are even more possible to take advantage of the procedure explained in this technical work.

References

[1] Adafruit, “Adafruit HUZZAH ESP8266 breakout, Using Arduino IDE”,
https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout/using-arduino-ide

[2] Components101, “ESP8266 - WiFi Module”,
https://components101.com/wireless/esp8266-pinout-configuration-features-datasheet

[3] Electrodragon, “esp8266-ElectroDragon”,
https://www.electrodragon.com/product-tag/esp8266/

[4] ElectronicWings, "Introduction to NodeMCU | NodeMCU",
https://www.electronicwings.com/nodemcu/introduction-to-nodemcu

[5] ESP8266 Community Wiki, “esp8266-module-family [ESP8266 Support WIKI]”
https://www.esp8266.com/wiki/doku.php?id=esp8266-module-family

[6] ESP8266 Community Wiki, “start [ESP8266 Support WIKI]”
https://www.esp8266.com/wiki/

[7] iot-playground, “Arduino ESP8266 IDE”,
https://iot-playground.com/blog/2-uncategorised/67-arduino-esp8266-ide

[8] OLIMEX Ltd., “Changing the Modes of MOD-WIFI-ESP8266-DEV. Reference. Revision B”, 2018. Available at
https://www.olimex.com/Products/IoT/ESP8266/MOD-WIFI-ESP8266-DEV/resources/MOD-WIFI-ESP8266-DEV_jumper_reference.pdf

[9] UpSkill Learning, ESP8266: Programming NodeMCU Using Arduino IDE - Get Started With ESP8266, CreateSpace Independent Publishing Platform, ISBN-10: 1534822666, 2016.

[10] WEMOS Electronics, “WEMOS wiki [WEMOS Electronics]”,
https://wiki.wemos.cc/doku.php