

# ESP8266 - MICROPYTHON

Jacopo Rodeschini 1046083  
Department of Computer Science  
Università degli studi Bergamo  
Dalmine, 24044<sup>1</sup>

<sup>1</sup>Univerità degli studi di Bergamo UniBg

<2019-12-09>

## **Why esp8266?**

Simple - cheaper - Wifi - Fast - 1Mb flash size.

## **Why micropython?**

MicroPython is a software implementation of a programming language largely compatible with Python 3, written in C, that is optimized to run on a microcontroller.

MicroPython is a full Python compiler and runtime that runs on the microcontroller hardware. In addition to implementing a selection of core Python libraries, MicroPython includes modules such as "machine" for accessing low-level hardware

Forst part of guide are centered to load micropython on NodeMcu v3 board, next show hoe to build custom PCB with ESP8266.

## 1 Dictionary

### **Firmware:**

E' un programma, ovvero una sequenza di istruzioni, integrato direttamente in un componente elettronico programmato (es. BIOS su ROM). I dispositivi più recenti consentono l'aggiornamento del firmware, in una scheda elettronica esso generalmente trova posto all'interno di una memoria ROM o flash. Il suo scopo è quello di avviare il componente stesso e consentirgli di interagire con altri componenti hardware tramite l'implementazione di protocolli di comunicazione o interfacce di programmazione. Rappresenta di fatto il punto di incontro fra le componenti logiche e fisiche di un dispositivo elettronico, ossia tra software e hardware.

### **Bootloader:**

Spesso esiste un altro componente software più semplice e di livello più basso, che si occupa delle funzioni minimali necessarie a gestire la memoria non volatile e a caricare il firmware, denominato bootloader.

### **Driver:**

Esso permette al sistema operativo di utilizzare l'hardware senza sapere come esso funzioni, ma dialogandoci attraverso un'interfaccia standard. Ne consegue

che un driver è specifico sia dal punto di vista dell'hardware che pilota, sia dal punto di vista del sistema operativo per cui è scritto. Non è possibile utilizzare driver scritti per un sistema operativo su uno differente, perché l'interfaccia è generalmente diversa.

## 2 System:

This guide are joined with Ubuntu 18.04

```
echo "hi, I'm:"  
uname -s  
echo "my brain is:"  
uname -v
```

```
hi, I'm:  
Linux  
my brain is:  
#46~18.04.1-Ubuntu SMP Fri Jul 10 07:21:24 UTC 2020
```

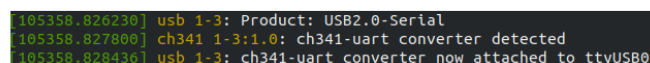
First step for configuration system is set user permission. Default user can't use tty\* (serial interface, COM) of machine. For this topic see below:  
use this command to find your permission

groups

if not see tty type of permission

sudo usermod -a -G tty yourname

Now plug and unplug your CH340 device from the USB port and use *dmesg* command to see what appened.  
if you see also like this:



```
105358.826230] usb 1-3: Product: USB2.0-Serial  
105358.827800] ch341 1-3:1.0: ch341-uart converter detected  
105358.828436] usb 1-3: ch341-uart converter now attached to ttyUSB0
```

Figure 1: dmesg output

the image show that esp8266 use ch340g driver (usb-uart converter) <sup>1</sup>. New version or different version of NodeMcu use cp12... converter. the procedure don't change, for driver installations see next sections.

---

<sup>1</sup>Note that for reason first you need to install ch340g driver on your laptop or device. In ubuntu this are just in normal installations. for other OS please check the driver. In second instance check the usb cable, this is the most cause of failure of procedure.

### 3 Connections:

Connctiong with machines were ESP are attached and controll if you have instal-lad **driver for esp8266 (ch340)** (install driver are the most critical operation, for more info follow [link](#))

```
ssh pi@<ip> ;; (view ./ssh/config)
```

```
:: for rasbian use this to downlaod driver
sudo apt update
sudo apt upgrade
```

:: attached esp8266 with (ch340) now attach nodeMcu end type dmesg if you see 1 output you have success install the driver

```
dmesg          :: view where esp are attached
ls /dev/tty*   :: look for /dev/ttyUSB0
```

Now NodeMcu are correctly connect with our machine (Ubuntu desktop).

### 4 Esptool

For next step, it's necessary install esptool.py (esp tool), this software python script that comunicate with NodeMcu rom and erase/update firmware. To install it, open terminal and use python packets maneger. (use latest python version 3.7)

```
pip install esptoo.py
```

for example, for see the chip specifications use:

```
esptool.py chip_id

esptool.py v2.8
Found 1 serial ports
Serial port /dev/ttyUSB0
Connecting....
Detecting chip type... ESP8266
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: 5c:cf:7f:3d:52:dd
Uploading stub...
Running stub...
Stub running...
Chip ID: 0x003d52dd
Hard resetting via RTS pin...
```

another useful optios are `-flash_id`

```
esptool.py flash_id
```

```

esptool.py v2.8
Found 1 serial ports
Serial port /dev/ttyUSB0
Connecting....
Detecting chip type... ESP8266
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: 5c:cf:7f:3d:52:dd
Uploading stub...
Running stub...
Stub running...
Manufacturer: c8
Device: 4016
Detected flash size: 4MB
Hard resetting via RTS pin...

```

with `flash_id` controll the memory of esp chip, with this information intall the true firmaware, add `-help` options too see full command.

## 5 Upload Firmware

in this step we cancell last firmaware, to upload micropython interpreter, download form this [Link](#) latest firmware, for thi issues check the flash memory size, in my case real flash size are 1Mb (Att: the `flash_id` return 4Mb, for nodeMcu this are divide by 4), then upload <sup>2</sup>.

```

esptool.py --port /dev/ttyUSB0 erase_flash // cancell previos firmware

esptool.py --port /dev/ttyUSB0 --baud 460800 write_flash
--flash_size=detect 0 esp8266-20180511-v1.9.4.bin --flash_mode=dout

```

for more info of this step and fix bug follow [firmware upload](#) (official documentation of micropython firmware upload).

## 6 RELP Terminal

RELP (inline interpreter micropython) is a more powerfull features of micropython (for me). You can open inline session with board (NodeMcu) and type upython code that will evalutate on board. The REPL has history, tab completion, auto-indent and paste mode for a great user experience. For open a new session type:

```
picocom /dev/ttyUSB0 -b115200
```

```
Terminal ready
```

---

<sup>2</sup>Note that line to upload new firmware there are `-flash_mode=dout` options, missing it cause failure of upload.

```
>>> // python interpreter
>>> 3 + 1
4
>>> print('Hello Word!!!')
Hello Word!!!
```

```
[C-a C-x] close session
#+END_SRC
```

```
now, put inside a python code e walla :).
in the following examples we turn on and off a simple pin named led
#+BEGIN_EXAMPLE
>>> from machine import Pin
>>> led = Pin(12,Pin.OUT)
>>> led.on()
>>> led.off()
>>> led.on()
```

Now with program ours nodemcu in micropython. For micropython documentation [doc](#) follow "Reference for ESP8266"

## 7 Run Script

An import task, is run a upython script (.py) that start run when board are powered. For this install ampy

```
sudo apt install ampy upgrade
```

then use this to run main.py script. The output was printed in command line.

```
ampy --port /dev/ttyUSB0 run main.py
```

when you use a infinite loop add `--no-output` options and open a repl terminal to see execution of program (picocom)

```
ampy --port /dev/ttyUSB0 run --no-output main.py
```

if you put our main file in memory, this run every times when board is powerd, use:

```
ampy --port /dev/ttyUSB0 put myfile.py /main.py
```

for more details on ampy tool or more oprions follow [ampy doc](#)

## 8 Custom board

8.1 TODO ESP-12 observations

8.2 TODO Power ESP-12

8.3 TODO Power boar

8.4 TODO UART connections

8.5 TODO Auto reset circuit

8.6 TODO Simple [°C] Sensor

8.7 TODO Simple Bjt Output