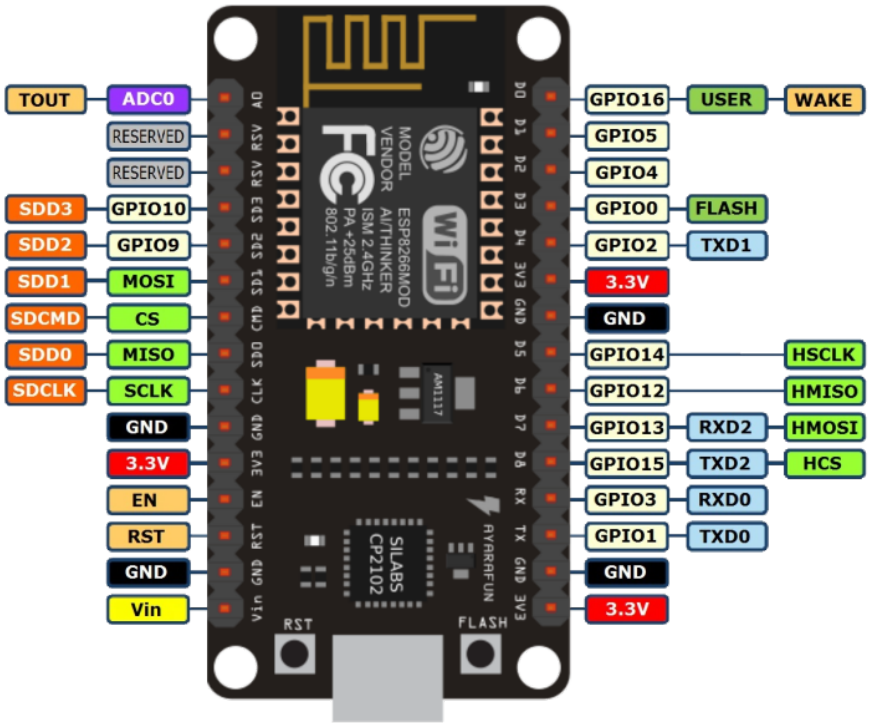
Python en microcontroladores

Nodemcu:

NodeMCU es una plataforma IoT de código abierto. Incluye firmware que se ejecuta en el ESP8266 Wi-Fi SoC de Espressif Systems , y el hardware que se basa en el módulo ESP-12. El término "NodeMCU" por defecto se refiere al firmware en lugar de los kits de desarrollo. El firmware utiliza el lenguaje de secuencias de comandos Lua . Se basa en el proyecto eLua , y se basa en el SDK Espressif OS para ESP8266 . Utiliza muchos proyectos de código abierto, como lua-cjson, y spiffs.



MICROCONTROLLER	ESP-8266EX
Operating Voltage	**3.3V**
Digital I/O Pins	11
Analog Input Pins	**1**
Clock Speed	80MHz/160MHz
Flash	**4M bytes**

Pasos:

1) Borrar el firmware del NODEMCU:

```
$ sudo pip install esptool
$ esptool.py --port /dev/ttyUSB0 erase_flash
```

También se puede descargar aquí:

```
$ git clone https://github.com/themadinventor/esptool.git
$ cd esptool
$ python esptool.py -h
```

2) Instalar el firmware del micro-python:

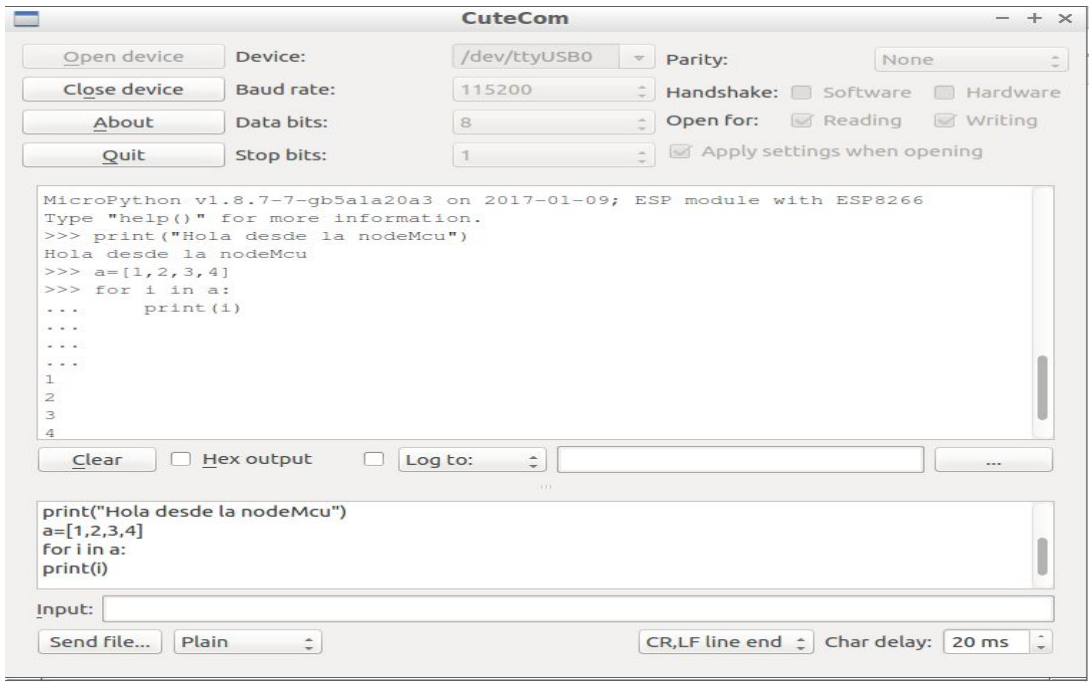
El firmware se encuentra en la siguiente dirección de internet:
http://micropython.org/download#esp8266

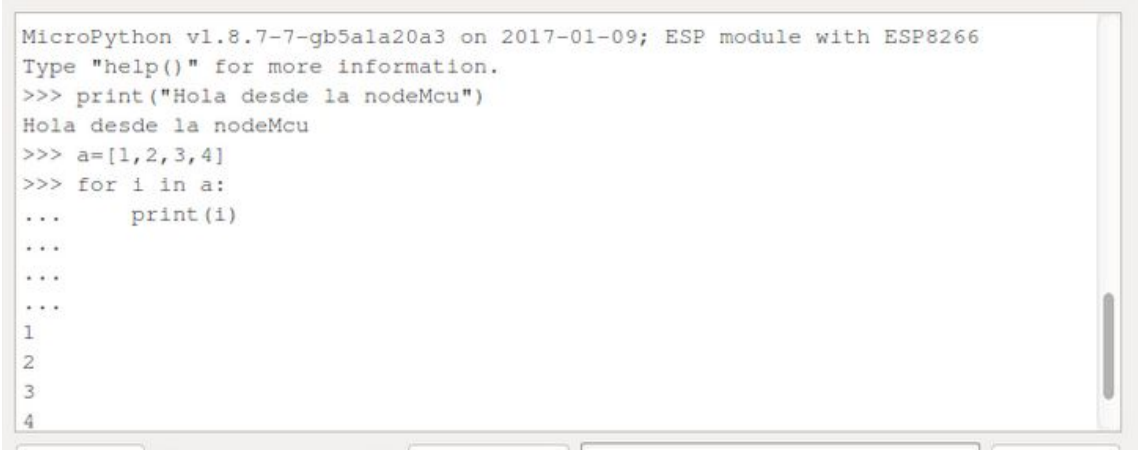
```
$ esptool.py --port /dev/ttyUSB0 --baud 460800 write_flash --flash_size=detect 0 esp8266-2016-05-03-v1.8.bin
```

3) Instalar algún software de monitoreo de protocolo serial, como el cutecom:

```
$ sudo apt-get install cutecom
```

4) **Abrir cutecom a la velocidad de 115200 y en el puerto correcto:**





5) Probar la sintaxis de python:

```
>>> print('hello esp8266!')
>>> 1 + 2
3
>>> 1 / 2
0.5
>>> 12**34
49222352429520267040371132431220080 64
```

Podemos usar sus pines como salida o entrada digital:

```
>>> import machine
>>> pin = machine.Pin(2, machine.Pin.OUT)
>>> pin.high()
>>> pin.low()
```

Más información de la sintaxis:
http://docs.micropython.org/en/latest/esp8266/esp8266/quickref.html

Librerías de MicroPython:
https://github.com/micropython/micropython-lib

6) Cargar script en la NODEMCU:

Instalar ampy de adafruit:

```
$ git clone https://github.com/adafruit/ampy.git
$ sudo python setup.py install
$ ampy --help
```

Probar algún script:

Se resalta que el script puede estar en cualquier parte de nuestra PC

```
$ ampy --port /serial/port run test.py
```

Probar script sin mensaje en terminal:

```
$ ampy --port /serial/port run --no-output test.py
```

7) Copiar, borrar y crear archivos a la tarjeta:

Con “put” podemos copiar archivos a la nodemcu.

```
$ ampy --port /serial/port put test.py
```

También podemos copiar carpetas:

```
$ ampy --port /serial/port put miCarpetaEnNodeMcu
```

Podemos leer, borrar y crear archivos:

```
$ ampy --port /serial/port get boot.py                #lee archivo boot.py
$ ampy --port /serial/port mkdir foo                  #crea una carpeta
$ ampy --port /serial/port mkdir /foo/bar             # crea carpeta recursiva
$ ampy --port /serial/port ls                         # lista los archivos y directorios del micro
$ ampy --port /serial/port rm test.py                 # borrar un archivo
$ ampy --port /serial/port rmdir /foo/bar             # borra un directorio
```

8) Grabar un archivo en la NODEMCU como archivo principal:

Secuencias de inicio

Hay dos archivos importantes que MicroPython busca en la raíz de su sistema de archivos. Estos archivos contienen el código de MicroPython que se ejecutará cada vez que la placa se enciende o restablece (es decir, se inicia). Estos archivos son:

/boot.py - Este archivo se ejecuta primero al encender / restablecer y debe contener código de bajo nivel que configura la placa para finalizar el arranque. Por lo general, no necesita modificar boot.py a menos que esté personalizando o modificando MicroPython. Sin embargo, es interesante ver el contenido del archivo para ver qué sucede cuando la placa se inicia. Recuerde que puede usar el comando ampy get para leer esto y cualquier otro archivo!

/main.py - Si este archivo existe, se ejecuta después de boot.py y debe contener cualquier script principal que desee ejecutar cuando la placa se encienda o restablezca.

```
$ ampy --port /serial/port put test.py /main.py
```