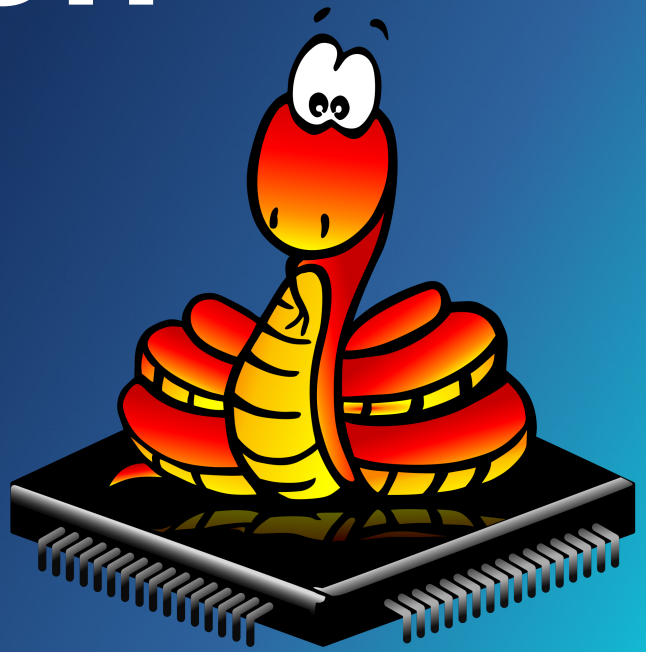
Micropython



Índice

¿Qué es **Micropython**?

- Es una implementación de Python 3 optimizada para funcionar en microcontroladores
- Contiene muchas de las funciones de Python para ordenadores, incluyendo el intérprete y versiones ligeras de algunas de las librerías nativas
- GitHub: https://github.com/micropython/micropython
- Página oficial: https://www.micropython.org
- Proyecto micropython-lib (ports de librerías a Micropython): https://github.com/micropython/micropython-lib

Historia de Micropython

- Creado por Damien George en 2013
- Primera versión publicada el 3 de mayo de 2014
- Última versión estable: 1.9.4 (11 de mayo de 2018)

¿Qué es un **microcontrolador**?

- Es un pequeño ordenador en un circuito integrado que puede ser programado para realizar ciertas tareas
- Contiene su propio procesador, memoria RAM y memoria flash
- Suele comunicarse con dispositivos y componentes electrónicos externos mediante pines GPIO (entrada y salida)
- Entre los más populares en la actualidad están los utilizados por las placas Arduino

¿Qué **microcontroladores** están **soportados por Micropython**?

- Placa oficial: STM32F405RG
- ESP8266 (placas Wemos D1 Mini)
- Arduino Due
- Teensy
- Otras de la familia STM32

Listado completo de placas compatibles:

https://github.com/micropython/micropython/wiki/Boards-Summary

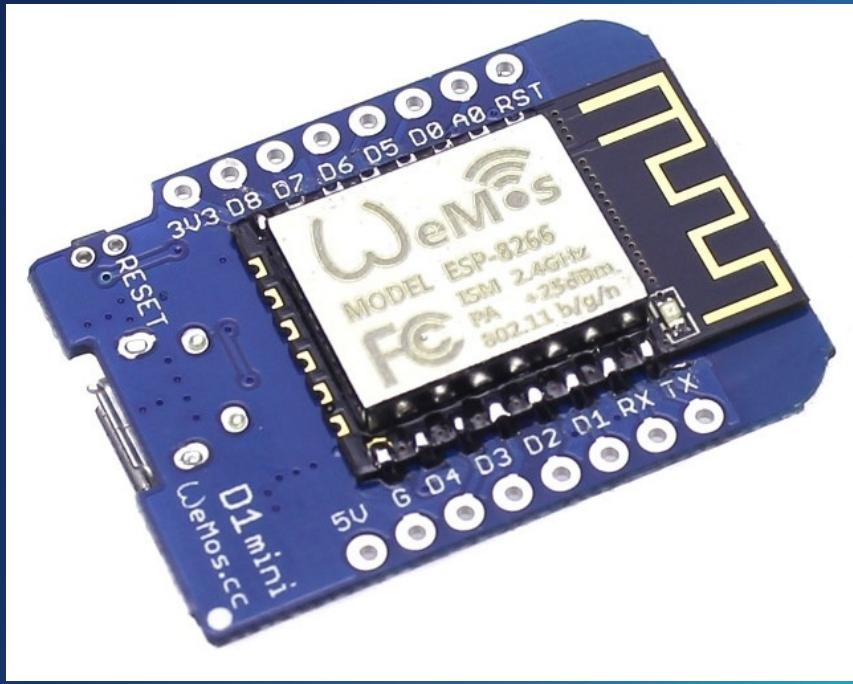
Micropython requiere que el microcontrolador tenga al menos 1MB de Flash, aunque existe una versión para micros de 512KB

El microcontrolador **ESP8266**

- La familia de SoC ESP8266 se impuso rápidamente como una alternativa a los microcontroladores utilizados por Arduino, al ofrecer:
  - Mejor potencia y prestaciones
  - Conectividad WiFi y stack TCP/IP incorporados
  - Bajo coste
- Existen diferentes versiones del ESP8266, que se diferencian principalmente por sus prestaciones y tamaño
- Estos micros pueden programarse con algún firmware (Micropython, Tasmota) o subiendo código creado con algún framework (Arduino).

La placa **Wemos D1 Mini**

- Para nuestros ejemplos, vamos a usar la placa Wemos D1 Mini



- Integra el SoC ESP-8266-EX

- Estas placas ofrecen:
  - bastantes pines
  - tamaño reducido
  - breadboard friendly
  - se pueden adquirir por poco dinero (clónicos), ~3€
  - Serial a USB incorporado

Especificaciones del SoC **ESP8266EX**

- Tensión de alimentación: 3.3V o 5V (con regulador integrado y USB)
- Tensión lógica: 3.3V (cuidado al utilizar módulos y componentes que funcionen a 5V)
- 11 pines digitales de entrada/salida, con soporte para PWM, I2C e interrupciones
- 1 entrada analógica (máx 3.2V)
- CPU: 32bit RISC, 80MHz/160MHz
- RAM: 160KB (64KB para instrucciones, 96KB para datos)
- Flash: 4MB

Instalando Micropython en un **Wemos D1 Mini**

(Los siguientes pasos son realizados desde un equipo con GNU/Linux)

1) Descargar firmware (bin): http://micropython.org/download/#esp8266
2) Instalar esptool desde pip
3) Dar permiso al usuario actual para que pueda usar el Serial en USB:

```
sudo adduser <user> dialout
sudo reboot
```

(o en su defecto ejecutar esptool como root)

github.com/enforcerzhukov

Instalando Micropython en un **Wemos D1 Mini**

4) Conectar el Wemos al ordenador, por USB
5) Borrar Flash del microcontrolador:

```
esptool.py --port /dev/ttyUSB0 erase_flash
```

(reemplazar `dev/ttyUSB0` por el puerto Serial pertinente)

6) Subir firmware descargado:

```
esptool.py --port /dev/ttyUSB0 --baud 460800
write_flash --flash_size=detect -fm dio 0
<archivoFirmwareDescargado.bin>
```

github.com/enforcerzhukov

Conexión al intérprete Micropython por **USB**

- Una vez Micropython está descargado, podemos conectarnos por USB al Serial y accederemos al intérprete de Micropython.
- Desde este intérprete podemos ejecutar comandos Python en tiempo real en el microcontrolador.
- Podemos usar Putty para conectarnos por Serial:
  - Puerto: /dev/ttyUSB0 (si no hay más dispositivos)
  - Baudrate: 115200

github.com/enforcerzhukov

El **sistema de archivos** de Micropython

- La instalación predeterminada de Micropython crea un sistema de archivos en la memoria Flash del microcontrolador
- Aquí podremos almacenar:
  - los scripts Python que se ejecutarán
  - otros ficheros que sean necesarios (pueden ser leídos y escritos desde Python)
- Existen dos scripts Python propios de Micropython:
  - boot.py
  - main.py

El fichero **boot.py** de Micropython

- Cuando el microcontrolador con Micropython arranca, se ejecuta automáticamente el script boot.py
- Contiene código de bajo nivel que no debería eliminarse ni alterarse
- Aunque a menudo se inicializan servicios como la conexión WiFi desde boot.py, no es recomendable modificarlo

El fichero **main.py** de Micropython

- Cuando la ejecución de boot.py finaliza, main.py entra en ejecución
- En este script podemos introducir nuestro propio código, con las funciones que deseamos implementar en el microcontrolador
- Es recomendable realizar la conexión a la red WiFi y el arranque de servicios (como Telnet) desde este fichero, en lugar de hacerlo desde boot.py

Conexión a una **red WiFi (cliente)**

```
import network

#Auto-conexión a Wifi
sta_if = network.WLAN(network.STA_IF)

if not sta_if.isconnected():
    print('connecting to network...')
    sta_if.active(True)
    sta_if.connect('SSID', 'PASS')
    while not sta_if.isconnected():
        pass

print('network config:', sta_if.ifconfig())

# Desactivar Wifi propia de Micropython (opcional)
network.WLAN(network.AP_IF).active(False)
```

Conexión mediante **WebREPL**

- Mediante WebREPL podemos conectarnos por WiFi al microcontrolador para:
  - Utilizar el intérprete de Python remotamente
  - Subir archivos al microcontrolador
  - Recuperar archivos del microcontrolador
- Activación de WebREPL:

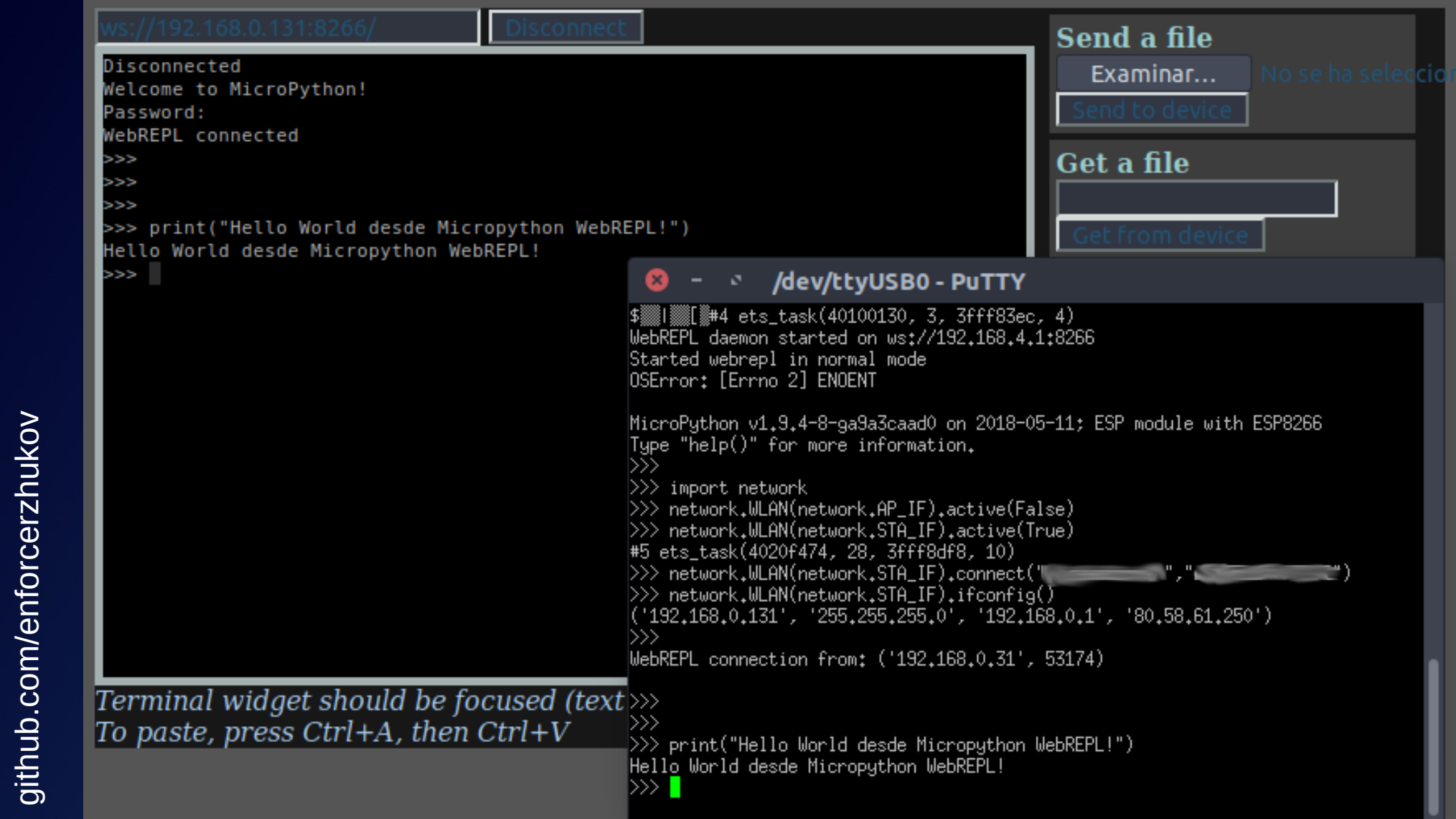
```
import webrepl_setup
```

- Se preguntará:
  - Si queremos habilitarlo para que se arranque automáticamente
  - La contraseña de acceso a WebREPL

Cliente **WebREPL**

- App web desde la que podemos conectarnos a Micropython remotamente por red, y:
  - Utilizar el intérprete Python
  - Subir y descargar ficheros
- Disponible desde: http://micropython.org/webrepl/

  (la comunicación con el microcontrolador es por LAN)
- Podemos descargarnos la página en HTML aquí: https://github.com/micropython/webrepl/blob/master/webrepl.html


ws://192.168.0.131:8266/
Disconnect
Disconnected
Welcome to MicroPython!
Password:
WebREPL connected
>>>
>>>
>>>
>>> print("Hello World desde Micropython WebREPL!")
Hello World desde Micropython WebREPL!
>>>
Send a file
Examinar...
No se ha seleccion
Send to device
Get a file
Get from device
/dev/ttyUSB0 - PuTTY
$▒|▒[▒#4 ets_task(40100130, 3, 3fff83ec, 4)
WebREPL daemon started on ws://192.168.4.1:8266
Started webrepl in normal mode
OSError: [Errno 2] ENOENT
MicroPython v1.9.4-8-ga9a3caad0 on 2018-05-11; ESP module with ESP8266
Type "help()" for more information.
>>>
>>> import network
>>> network.WLAN(network.AP_IF).active(False)
>>> network.WLAN(network.STA_IF).active(True)
#5 ets_task(4020f474, 28, 3fff8df8, 10)
>>> network.WLAN(network.STA_IF).connect("[redacted]","[redacted]")
>>> network.WLAN(network.STA_IF).ifconfig()
('192.168.0.131', '255.255.255.0', '192.168.0.1', '80.58.61.250')
>>>
WebREPL connection from: ('192.168.0.31', 53174)
>>>
>>>
>>> print("Hello World desde Micropython WebREPL!")
Hello World desde Micropython WebREPL!
>>>
Terminal widget should be focused (text
To paste, press Ctrl+A, then Ctrl+V
github.com/enforcerzhukov

Pinout del Wemos D1 Mini

- La numeración de los pines en la placa, y la numeración utilizada en Micropython, no coinciden
- Las equivalencias son las siguientes:

Pin placa	Pin Micropython	Pin placa	Pin Micropython
D0, LED2	16	D5	14
D1	5	D6	12
D2	4	D7	13
D2	4	D8	15
D3	0	SD2	9
D4, LED1	2	SD3	10

Ejemplo: **LED blink**

- El siguiente ejemplo hará parpadear el LED integrado en la placa Wemos D1 Mini (pin 2) cada segundo:

```
from machine import Pin
from utime import sleep

LED = Pin(2, Pin.OUT)
state = False

while True:
   LED.value(state)
   state = not state
   sleep(1)
```