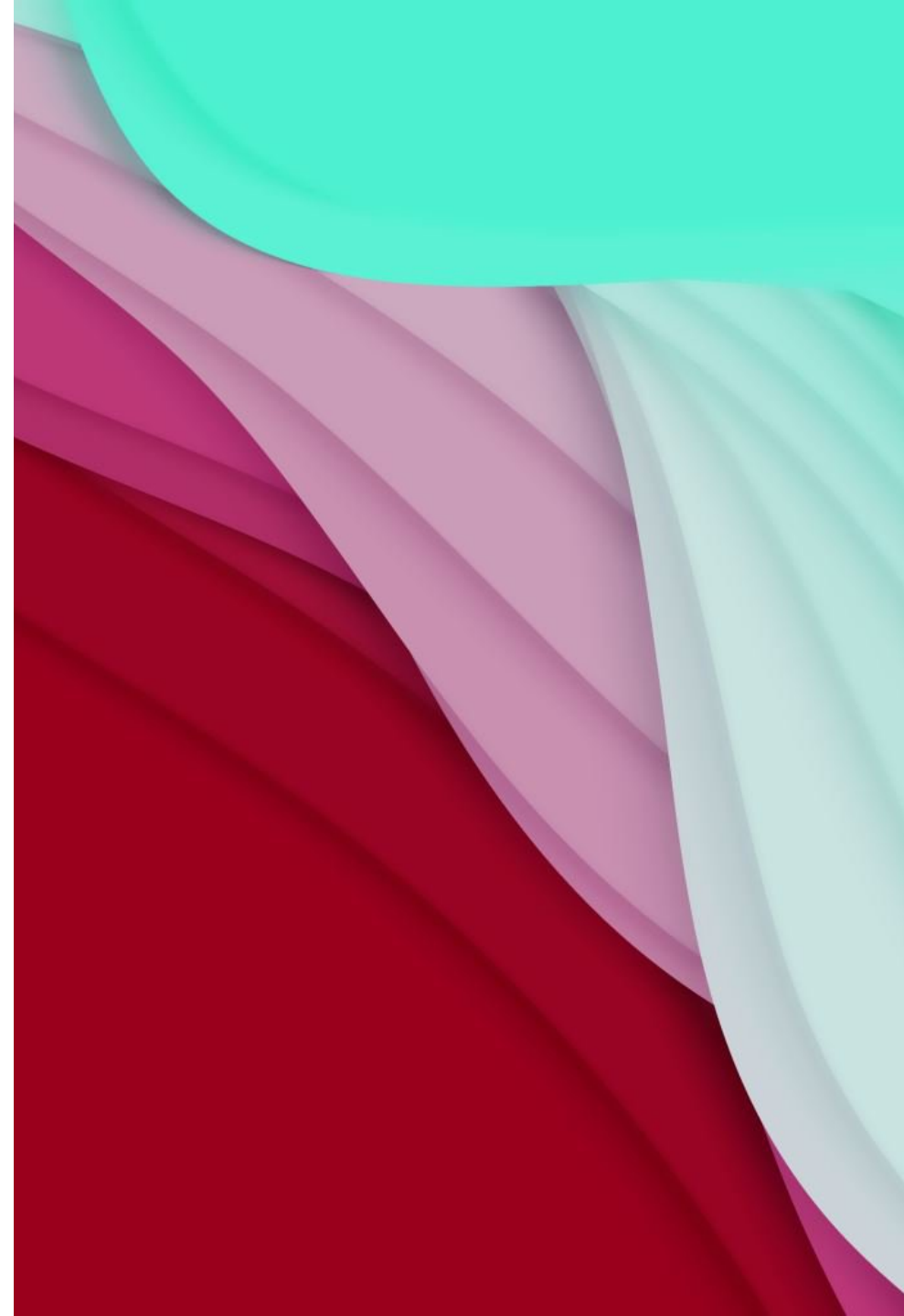


Estación Meteorológica IoT con ESP32 en MicroPython



Objetivos



Objetivos

Objetivo general:

Diseñar e implementar una estación meteorológica utilizando un microcontrolador ESP32 programado en MicroPython, que recoja datos de sensores ambientales y los muestre en una página web accesible por Wi-Fi.

Objetivos específicos:

Configurar sensores ambientales (temperatura, humedad y presión) y conectarlos al ESP32.

Establecer la conexión Wi-Fi para la transmisión de datos.

Programar un servidor web embebido que visualice los datos en tiempo real.

Diseñar una interfaz web sencilla, compatible con navegadores de computadoras y dispositivos móviles.

Introducción



En la actualidad, el monitoreo ambiental en tiempo real es una herramienta clave en campos como la agricultura, la educación, la domótica y la meteorología. Gracias a los avances en tecnologías de bajo costo, es posible construir estaciones meteorológicas compactas y económicas usando plataformas como el ESP32. Este microcontrolador no solo permite recolectar información del ambiente a través de sensores, sino también publicar esos datos a través de una red Wi-Fi, haciendo posible su consulta desde cualquier navegador web.

Este proyecto propone una solución sencilla para visualizar datos meteorológicos localmente, sin necesidad de servicios en la nube, utilizando MicroPython como entorno de desarrollo por su simplicidad y eficiencia en dispositivos embebidos.



Justificación

Implementar una estación meteorológica con ESP32 tiene múltiples beneficios tanto educativos como prácticos:

Educativos: Permite a estudiantes y desarrolladores comprender conceptos de sensores, comunicación de red, servidores web y programación embebida.

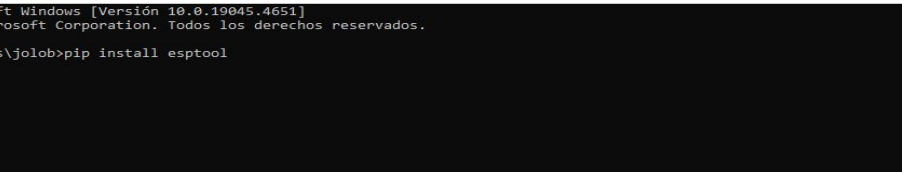
Prácticos: Proporciona una herramienta útil para monitorear condiciones ambientales en tiempo real sin depender de internet, ideal para lugares rurales o espacios cerrados.

Económicos: El uso de componentes de bajo costo como el ESP32, sensores DHT22 y BMP280 permite crear una solución funcional a una fracción del precio de una estación meteorológica comercial.

Además, utilizar MicroPython reduce la complejidad de desarrollo, facilitando la comprensión del código y acelerando el prototipado.

MicroPython en ESP32

- Instalar esptool. Usar comando :
C:\Users\usuario\pip install esptool



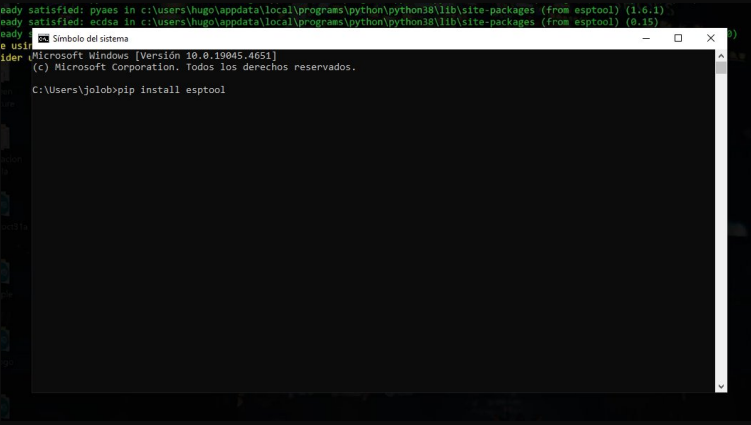
```
Microsoft Windows [Versión 10.0.19045.4651]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\jolo>pip install esptool
```

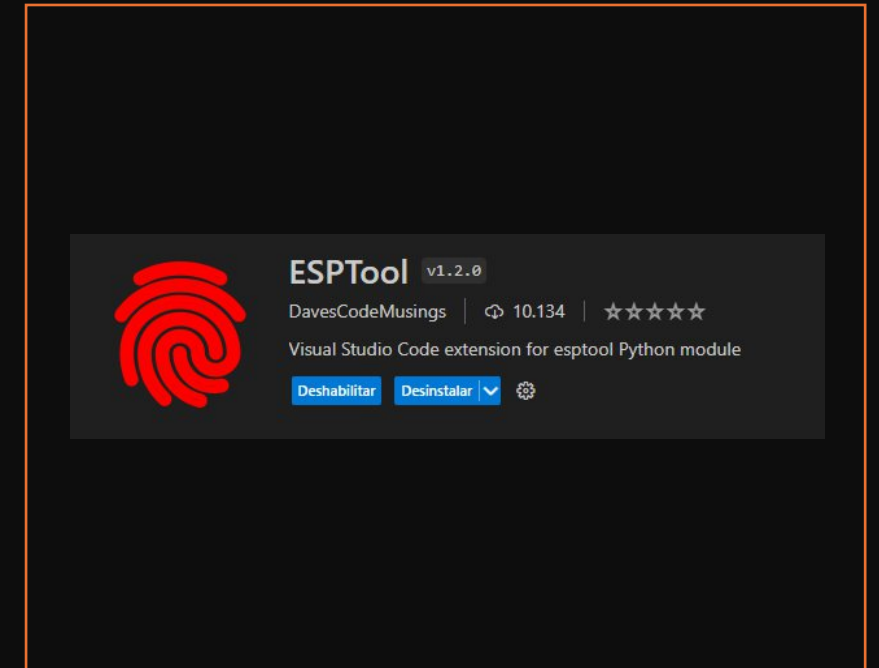


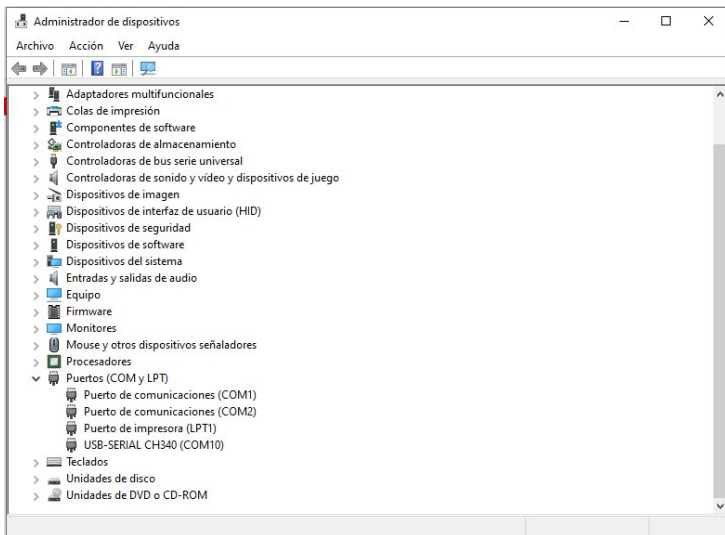
MicroPython en ESP32

- Si pip install esptool causa error, probar:
- python -m install esptool
- pip2 install esptool



```
Ready satisfied: pyaes in c:\users\hugo\appdata\local\programs\python\python38\lib\site-packages (from esptool) (1.6.1)
Ready satisfied: ecdsa in c:\users\hugo\appdata\local\programs\python\python38\lib\site-packages (from esptool) (0.16)
Ready
e usi
ider
Microsoft Windows [Versión 10.0.19045.4651]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\jolo>pip install esptool
```





Ubicar el puerto de comunicación
(COM 10)

Borrar Memoria, “Flashear”

Usar el comando:

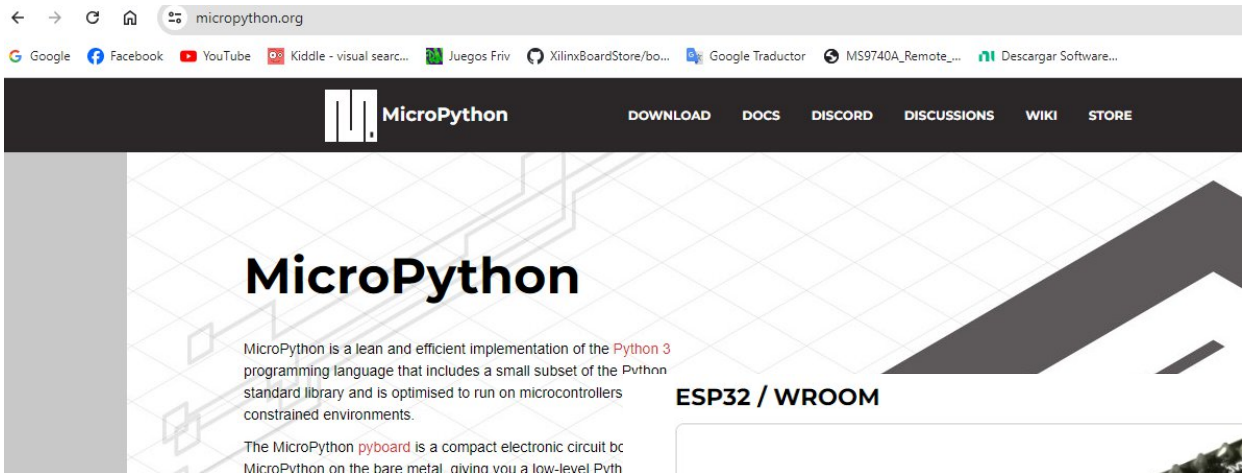
```
esptool --chip esp32 --port COM10 erase_flash
```

```
C:\Users\Hugo>esptool.py --chip esp32 --port COM5 erase_flash
esptool.py v2.8
Serial port COM5
Connecting....._....._
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: fc:f5:c4:2f:47:30
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 8.4s
Hard resetting via RTS pin...
```

Si presenta error, cambiar esptool.py, por esptool, sin la terminación
O mantener presionado el botón de boot del Arduino, mientras se
Hace la comunicación.

Instalar firmware

1. Vamos al sitio micropython.org y damos en Download



2. Escogemos nuestra tarjeta
Esp32 WROOM



Firmware

Releases

[v1.23.0 \(2024-06-02\)](#) [.bin](#) / [\[.app-bin\]](#) / [\[.elf\]](#) / [\[.map\]](#) / [\[Release notes\]](#) (latest)
[v1.22.2 \(2024-02-22\)](#) [.bin](#) / [\[.app-bin\]](#) / [\[.elf\]](#) / [\[.map\]](#) / [\[Release notes\]](#)
[v1.22.1 \(2024-01-05\)](#) [.bin](#) / [\[.app-bin\]](#) / [\[.elf\]](#) / [\[.map\]](#) / [\[Release notes\]](#)
[v1.22.0 \(2023-12-27\)](#) [.bin](#) / [\[.app-bin\]](#) / [\[.elf\]](#) / [\[.map\]](#) / [\[Release notes\]](#)
[v1.21.0 \(2023-10-05\)](#) [.bin](#) / [\[.app-bin\]](#) / [\[.elf\]](#) / [\[.map\]](#) / [\[Release notes\]](#)

3. Descargar en el directorio de trabajo

C:\Users\usuario

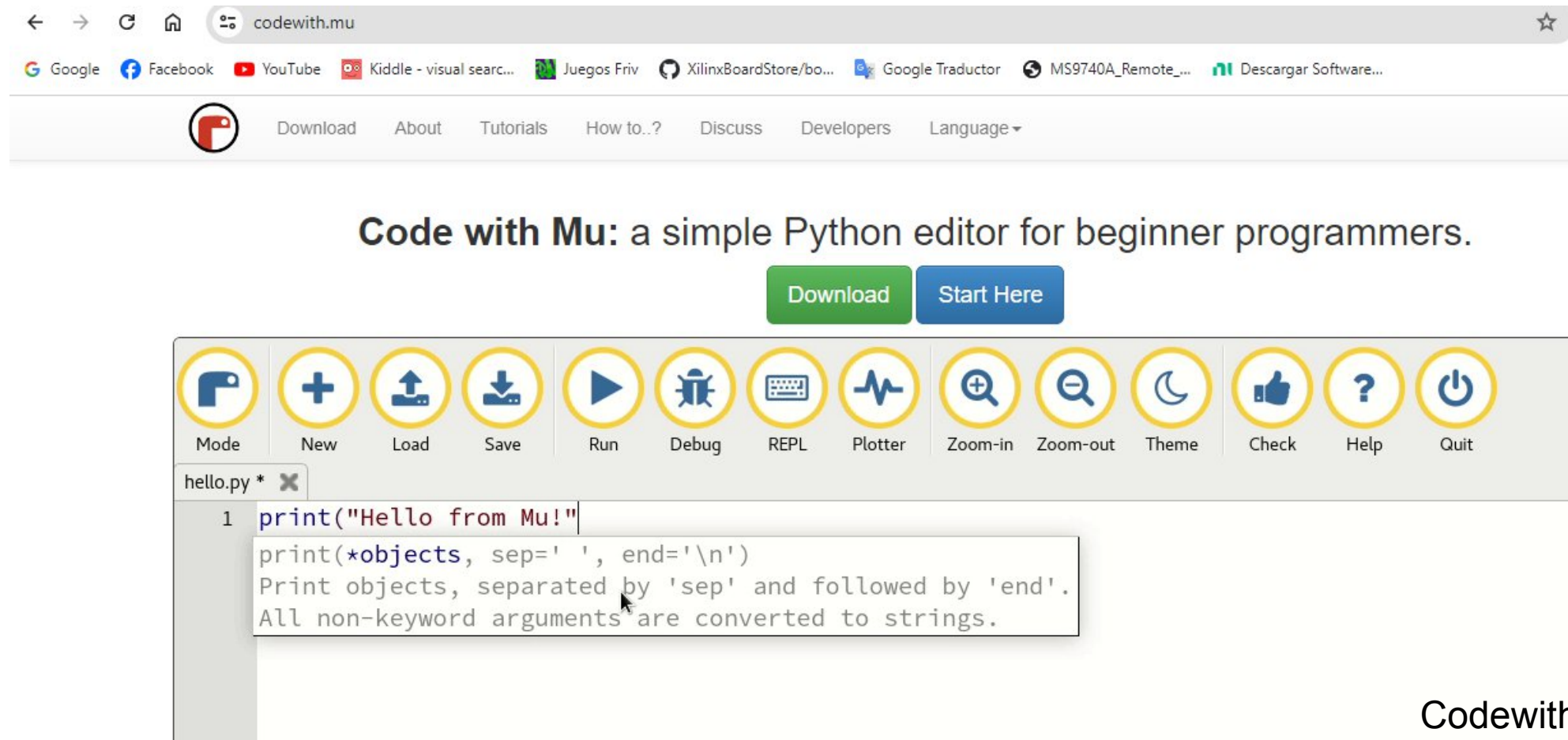
Escribir en el ESP32

```
C:\Users\jolob>esptool --chip esp32 --port COM10 --baud 460800 write_flash -z 0x1000 ESP32_GENERIC-20240602-v1.23.0.bin
esptool.py v4.7.0
Serial port COM10
Connecting....
Chip is ESP32-D0WD (revision v1.0)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 98:cd:ac:ab:e6:e0
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Flash will be erased from 0x00001000 to 0x001a8fff...
Compressed 1734240 bytes to 1142447...
Wrote 1734240 bytes (1142447 compressed) at 0x00001000 in 25.5 seconds (effective 543.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

esptool --chip esp32 --port COM10 --baud 460800 write_flash -z 0x1000 archivo.bin

Descargar e instalar MU Editor.



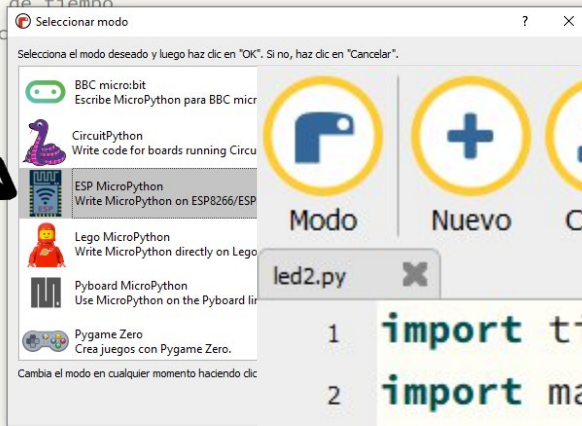
The screenshot displays the website codewith.mu in a web browser. The browser's address bar shows the URL, and the page title is "Code with Mu: a simple Python editor for beginner programmers." Below the title, there are two buttons: "Download" (green) and "Start Here" (blue). The main content area features a toolbar with 13 icons: Mode (a red 'f' in a circle), New (a plus sign), Load (an upload arrow), Save (a download arrow), Run (a play button), Debug (a bug icon), REPL (a keyboard icon), Plotter (a waveform icon), Zoom-in (a magnifying glass with a plus), Zoom-out (a magnifying glass with a minus), Theme (a moon icon), Check (a thumbs up), Help (a question mark), and Quit (a power button). Below the toolbar, a code editor window titled "hello.py" is open, showing the following Python code:

```
1 print("Hello from Mu!")
```

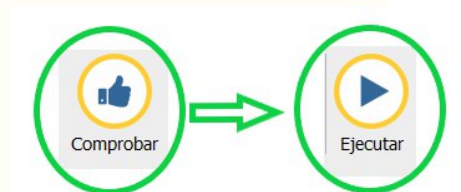
A tooltip is visible over the `print` function, displaying the following text:

```
print(*objects, sep=' ', end='\n')
Print objects, separated by 'sep' and followed by 'end'.
All non-keyword arguments are converted to strings.
```

The website's footer contains the text "Codewith.mu".



```
1 import time # configuracion de tiempo
2 import machine # configuracion de pines esp32
3 # Escribe tu código aquí :-)
4 tiempo = 500 # milisegundos
5 # configuracion de pines GPIO E/S
6 boton = machine.Pin(15, machine.Pin.IN)
7 led = machine.Pin(2, machine.Pin.OUT)
8 # codigo
9 while True:
10     if (boton.value()):
11         led.on()
12         time.sleep_ms(tiempo)
13         led.off()
14         time.sleep_ms(tiempo)
15
```



```

1 import time
2 import machine
3
4 led = machine.Pin(2, machine.Pin.OUT)
5 led.off()
6
7 def destellos(*arb):
8     suma = 0
9     for i in arb:
10         suma += i
11
12     print("El Led prendera ", suma, " veces")
13     for x in range(suma):
14         led.on()
15         time.sleep_ms(200)
16         led.off()
17         time.sleep_ms(200)

```

ESP MicroPython REPL

MPY: soft reboot

raw REPL; CTRL-B to exit

>OK

❏❏>

MicroPython v1.23.0 on 2024-06-02; Generic ESP32 module with ESP32

Type "help()" for more information.

>>> destellos(1,4,6)

El Led prendera 11 veces

>>> |

```

1 import time
2 import machine
3
4 led = machine.Pin(2, machine.Pin.OUT)
5 led.off()
6 def saludo():
7     print("Hola mundo")
8
9 def blink(n):
10     for x in range(n):
11         led.on()
12         time.sleep(1)
13         led.off()
14         time.sleep(1)

```

ESP MicroPython REPL

...

MPY: soft reboot

raw REPL; CTRL-B to exit

>OK

❏❏>

MicroPython v1.23.0 on 2024-06-02; Generic ESP32 module with ESP32

Type "help()" for more information.

>>> blink(10)

>>> |


```
1 import time
2 import machine
3
4 def blink(duracion, cant, npin):
5     """Parpadea un led
6     duracion: tiempo de encendido y apagado
7     cant:      cantidad de veces
8     npin:      # del pin"""
9     led = machine.Pin(npin, machine.Pin.OUT)
10    for x in range(cant):
11        led.on()
12        time.sleep_ms(duracion)
13        led.off()
14        time.sleep_ms(duracion)
15
16    return ('DONE')
17
```

ESP MicroPython REPL

ipy. soft reboot

aw REPL; CTRL-B to exit

OK

18

MicroPython v1.23.0 on 2024-06-02; Generic ESP32 module with ESP32

Type "help()" for more information.

>>> blink(500,20,2)

DONE'

ESP8266 - blink1.py

Modo

Nuevo

Cargar

Guardar

Ejecutar

Archivos

REPL

Trazador

Acercar

Alejar

Tema

Comprobar

Tidy

Ayuda

Salir

1

import time

2

import machine

3

4

def blink(duracion, cant, npin):

5

"""Parpadea un led

6

duracion: tiempo de encendido y apagado

7

cant: cantidad de veces

8

npin: # del pin"""

9

led = machine.Pin(npin, machine.Pin.OUT)

10

for x in range(cant):

11

led.on()

12

time.sleep_ms(duracion)

13

led.off()

14

time.sleep_ms(duracion)

15

Filesystem on ESP8266/ESP32 board

Files on your device:

boot.py

led1.py

Archivos en tu ordenador:

blink1.py

led1.py

led2.py

ledFunction.py

ledFunction2.py

ESP 32

PC

Archivos

Mu 1.2.0

Modo

Nuevo

Cargar

Guardar

Ejecutar

Archivos

REPL

Trazador

Acercar

Alejar

Tema

Comprobar

Tidy

Ayuda

Salir

ESP MicroPython REPL

MicroPython v1.23.0 on 2024-06-02; Generic ESP32 module with ESP32

Type "help()" for more information.

>>>

>>> import blink1

>>> blink1.blink(500,20,2)

'DONE'

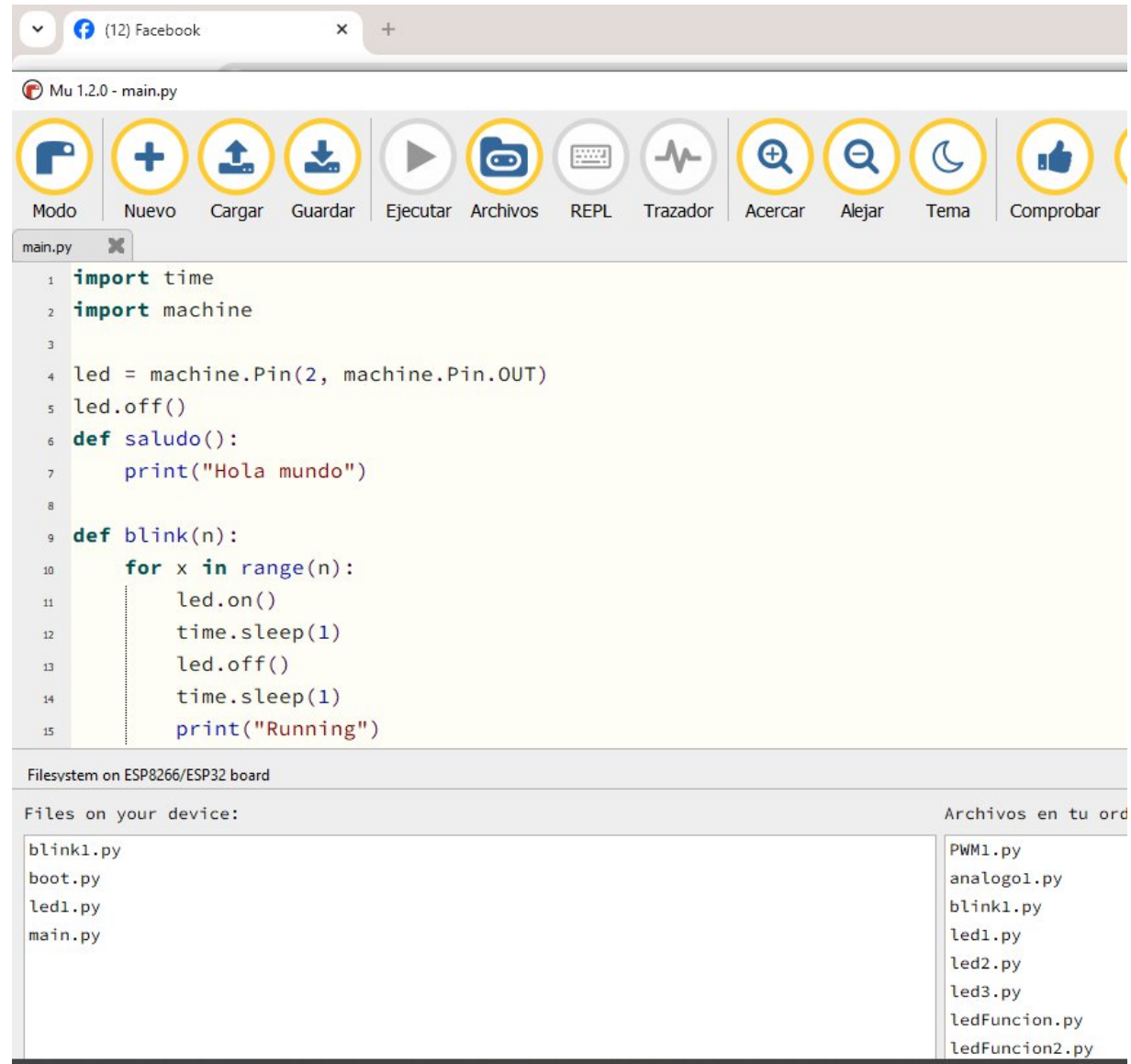
>>> |

Conversor ADC

```
1 from machine import ADC, Pin
2 import time
3
4 adc = ADC(Pin(32))    # configura pin del ADC
5 adc.atten(ADC.ATTN_11DB) # rango de voltaje 0 a 3.6 V
6 """ADC.ATTN_0DB      : voltaje máximo 1 V
7   ADC.ATTN_2_5DB     : voltaje máximo 1.34 V
8   ADC.ATTN_6DB       : voltaje máximo 2 V
9   ADC.ATTN_11DB      : voltaje máximo 3.6 V
10 """
11
12 while True:
13     v = (adc.read() , 0, 4095)
14     print(v)
15     time.sleep_ms(200)
```


Inicio Automático.

- Para que un software se haga automático (al iniciar), el nombre del programa debe ser main.py



Estación Meteorológica

Configuración de librerías y
sensores

```
import network
import socket
import time
from machine import Pin, I2C
import dht
from bmp280 import BMP280
```

Inicio de Sensores y WIFI

```
# Configurar sensores
dht_sensor = dht.DHT22(Pin(4)) # Cambia por DHT11 si lo usas
i2c = I2C(0, scl=Pin(22), sda=Pin(21))
bmp = BMP280(i2c)

# Conexión Wi-Fi
ssid = 'TU_SSID'
password = 'TU_PASSWORD'

station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)

while not station.isconnected():
    pass

print('Conectado a WiFi:', station.ifconfig())
```



Página HTML

Página HTML

```
def generar_pagina(temp, hum, pres):  
    html = f"""<!DOCTYPE html>  
<html>  
<head>  
    <title>Estación Meteorológica ESP32</title>  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <style>  
        body {{ font-family: Arial; text-align: center; margin-top: 50px; }}  
        .data {{ font-size: 24px; margin: 20px; }}  
    </style>  
</head>  
<body>
```

```
    <h1>Estación Meteorológica</h1>  
    <div class="data">🌡 Temperatura: {temp:.2f} °C</div>  
    <div class="data">💧 Humedad: {hum:.2f} %</div>  
    <div class="data">📏 Presión: {pres:.2f} hPa</div>  
</body>  
</html>"""  
    return html
```

```
# Servidor web  
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]  
  
s = socket.socket()  
s.bind(addr)  
s.listen(1)  
  
print('Servidor web en http://%s' % station.ifconfig()[0])
```

Generar IP conexión

Código

```
while True:
    try:
        cl, addr = s.accept()
        print('Cliente conectado desde', addr)

        dht_sensor.measure()
        temp = dht_sensor.temperature()
        hum = dht_sensor.humidity()
        pres = bmp.pressure

        response = generar_pagina(temp, hum, pres)

        cl_file = cl.makefile('rwb', 0)
        while True:
            line = cl_file.readline()
            if not line or line == b'\r\n':
                break
```

```
        cl_file = cl.makefile('rwb', 0)
        while True:
            line = cl_file.readline()
            if not line or line == b'\r\n':
                break

        cl.send('HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n')
        cl.send(response)
        cl.close()
    except Exception as e:
        print('Error:', e)
```

Conexión

- El navegador (PC o Celular) debe estar en la misma red que el ESP32
- En el navegador escribe la IP que aparece al momento de conectarse
- Servidor web en <http://192.168.X.X>

