

```
1
2 Programming 'Language' = {
3
4     Introducción = [Python,
5                     MicroPython]
6
7
8
9
10
11 }
12
13
14
```



```
1
2
3 Clase 06 = {
4
5     Presentación = [Les
6                     damos la bienvenida al
7                     curso]
8
9
10
11
12 }
13
14
```



Clases = {

Clase 05

Conceptos Básicos de P00. Clases y objetos. Métodos y atributos.



Clase 06

Instalación de MicroPython en la placa ESP32.
Introducción a la herramientas de desarrollo Thonny.
Conexión y configuración de la placa.

Clase 07

Control de Hardware Básico. Manejo de pines GPIO.
Lectura de sensores y actuadores.

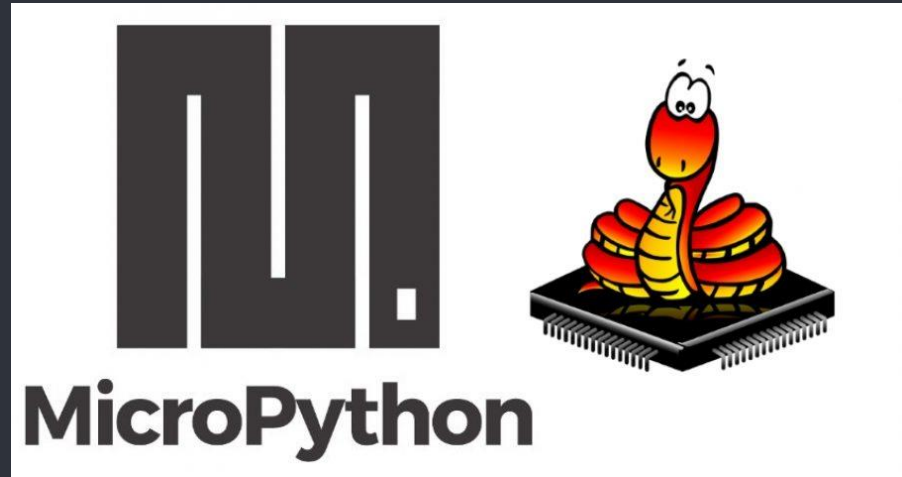
Clase 08

Comunicación Serial. UART, I2C, SPI.
Comunicación entre dispositivos.



MicroPython {

Damien George, físico australiano que desarrolló MicroPython. En 2013 hizo una campaña en Kickstarter en la cual los fondos recaudados sobrepasaron por mucho el objetivo.



}



MicroPython {

Como ya vimos en la primera clase, MicroPython es una implementación ligera de Python 3 diseñada para ejecutarse en microcontroladores y sistemas embebidos. Algunos de sus puntos fuertes son:

Ligero y Eficiente: Diseñado para ser eficiente en términos de memoria y recursos, lo que lo hace ideal para dispositivos con recursos limitados.

Compatibilidad con Python: Aunque es una versión reducida, mantiene una alta compatibilidad con Python 3, permitiendo a los desarrolladores usar muchas de las características y bibliotecas estándar de Python.

Interacción con Hardware: Proporciona módulos específicos para interactuar con el hardware, como machine para controlar pines GPIO, PWM, ADC, etc.

REPL: Ofrece un entorno REPL (Read-Eval-Print Loop) que permite a los desarrolladores escribir y probar código en tiempo real directamente en el dispositivo.

Portabilidad: Puede ejecutarse en una variedad de microcontroladores, incluyendo ESP8266, ESP32, STM32, y otros.

}



Sistemas compatibles {

MicroPython es compatible con una amplia variedad de microcontroladores.

ESP: ESP8266 y ESP32 en sus varias versiones. Tienen WiFi y Bluetooth.

STM32: Una familia de microcontroladores de STMicroelectronics.

RP2040 y **RP2350**: Microcontroladores de Raspberry Pi, utilizado en la Raspberry Pi Pico y Pico 2.

SAMD21: Utilizado en algunas placas de Adafruit como las Feather.

nRF52: Microcontroladores de Nordic Semiconductor con BLE.

ARM: Placas de desarrollo como las Teensy.

Pyboard: La placa de desarrollo oficial de MicroPython basada en STM32.

RISC-V: Microsemi hizo un fork de MicroPython para sus RV32 y RV64.

Cada uno de estos microcontroladores tiene sus propias características y capacidades, lo que permite a los desarrolladores elegir el más adecuado para su proyecto específico. La documentación oficial de MicroPython proporciona guías detalladas para configurar y programar cada uno de estos microcontroladores.

}



Sistemas compatibles {

```
# Lista completa
# https://MicroPython.org/download/

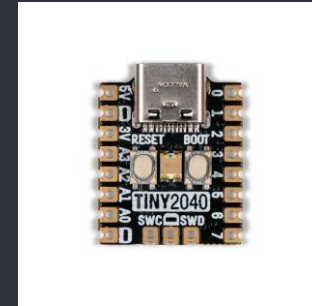
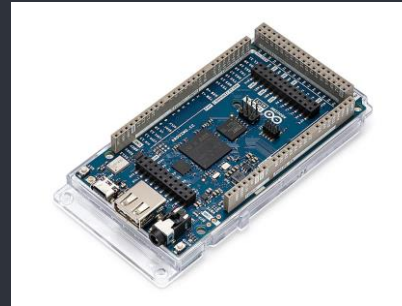
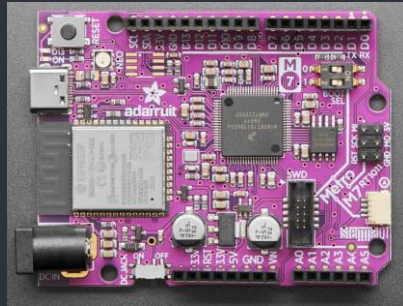
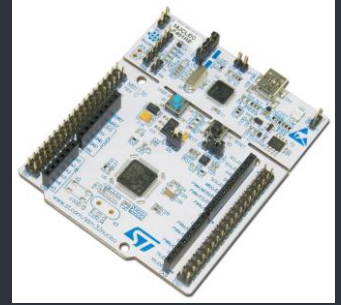
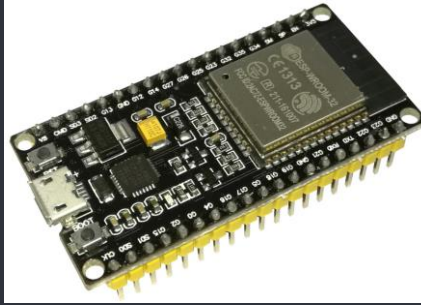
# Vendor: Actinius, Adafruit, Arduino, BBC, Espressif, Espruino, Fez,
George Robotics, HydraBus, I-SYST, LEGO, LILYGO, Laird Connectivity,
LimiFrog, M5 Stack, M5Stack, Makerdiary, McHobby, Microchip,
MikroElektronika, MiniFig Boards, NXP, Netduino, Nordic Semiconductor,
OLIMEX, PJRC, Particle, Pimoroni, Pololu, Pycom, Raspberry Pi, Renesas
Electronics, ST Microelectronics, Seeed Studio, Silicognition,
Silicognition LLC, Sparkfun, Unexpected Maker, VCC-GND Studio, Vekatech,
WeAct, Wemos, Wireless-Tag, Wiznet, nullbits, u-blox

# Microcontrolador: RA6M5, cc3200, esp32, esp32c3 (Risc-V), esp32c6,
esp32s2, esp32s3, esp8266, mimxrt, nrf51, nrf52, nrf91, ra4m1, ra4w1,
ra6m1, ra6m2, ra6m5, rp2040, rp2350, samd21, samd51, stm32f0, stm32f4,
stm32f7, stm32g0, stm32g4, stm32h5, stm32h7, stm32l0, stm32l1, stm32l4,
stm32wb, stm32wl
```

}



Sistemas compatibles {



}



Documentación {

La pagina oficial de MicroPython tiene todo lo necesario para comenzar..

<https://micropython.org/>

La documentación oficial tiene ejemplos y códigos listos para usar.
Consultarla es importantísimo.

<https://docs.micropython.org/en/latest/>

Para descargar MicroPython para la placa que dispongamos, vamos a
Download.

<https://micropython.org/download/>



MicroPython

DOWNLOAD

DOCS

DISCORD

DISCUSSIONS

WIKI

STORE



ESP32 {

El ESP32 es un microcontrolador de bajo costo y alto rendimiento desarrollado por Espressif Systems. Es ampliamente utilizado en proyectos de Internet de las Cosas (IoT) debido a sus capacidades integradas de WiFi y Bluetooth.

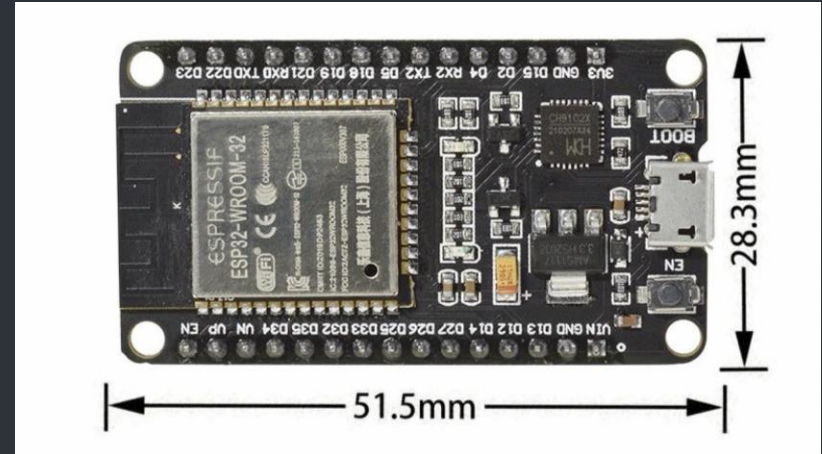
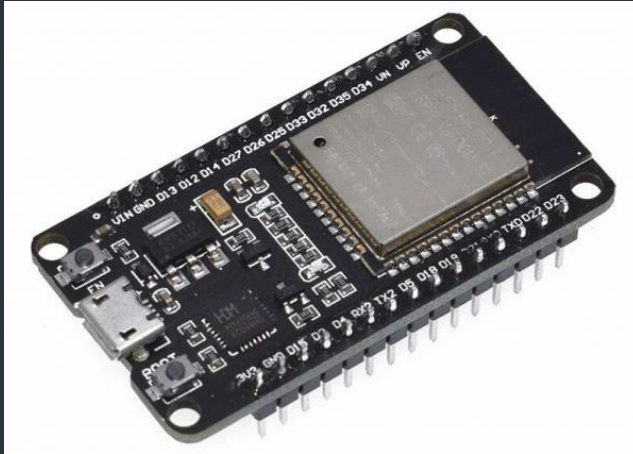
- Procesador **Xtensa LX6 de 32 bits de doble núcleo**
- Velocidad de **160Mhz** (máximo **240 Mhz**)
- Memoria **520 KiB SRAM**
- Memoria **448 KiB ROM**
- Memoria **flash** externa hasta **16MiB**
- Stack **TCP/IP integrada**
- **Wifi** 802.11 b/g/n **2.4GHz** (soporta WFA/WPA/WPA2/WAPI)
- **Bluetooth v4.2** BR/EDR y BLE
- 32 pines GPIO
- Conversor analógico digital (**ADC**) de 12bits y 18 canales
- 2 conversores digital analógico (**DAC**) de 8bits

- 16 salidas **PWM** (LED PWM)
- **Sensores capacitivos** (en GPIO)
- 3x UART, 4x SPI, 2x I2S, 2x I2C, CAN bus 2.0
- Controladora host SD/SDIO/CE-ATA/MMC/Emmc
- Sensor de temperatura
- Sensor de efecto Hall
- Reloj tiempo real (RTC)
- Conector Micro Usb
- Lenguajes de programación: LUA, C++, MicroPython, entre otros.
- **Tensión de trabajo 3V3 (;CUIDADO!)**
- **Corriente GPIO max de salida 40mA (recomendado 20mA) o 1200mA max en total.**

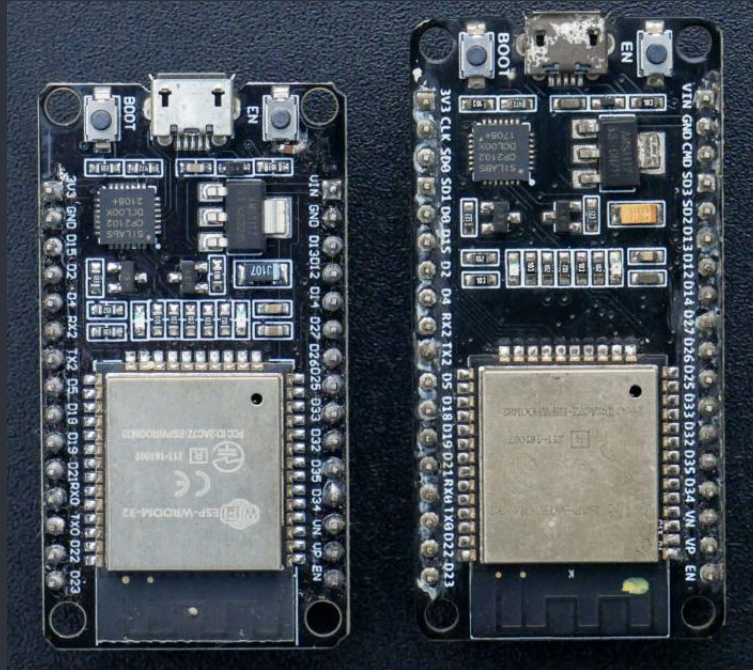


ESP32 DevKit V1 {

La placa ESP32 DevKit V1 es una plataforma de desarrollo basada en el módulo ESP32-WROOM-32, que integra conectividad WiFi y Bluetooth. La que usaremos usa un integrado CH9102X para poder conectar el puerto USB con la computadora y programarla, además de un chip de 4MiB de flash SPI.



ESP32 DevKit V1 {



Aquí vemos dos versiones del DevKit V1.

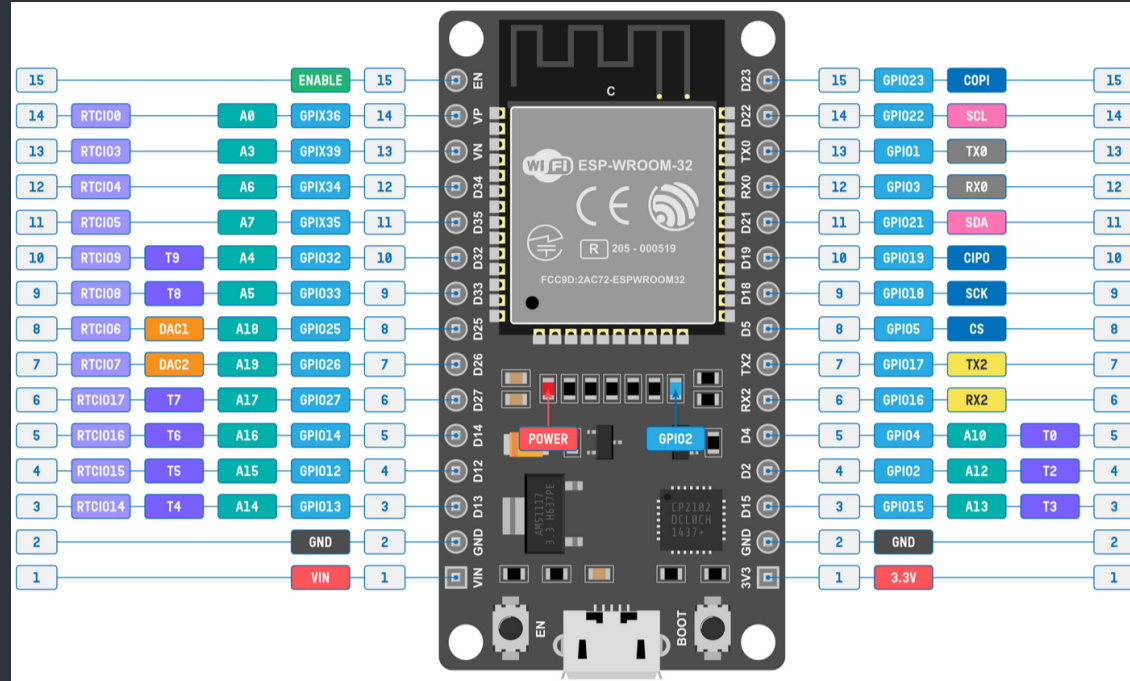
Nosotros vamos a usar el modelo de la izquierda que tiene 30 pines.

También vienen modelos con 36 pines.

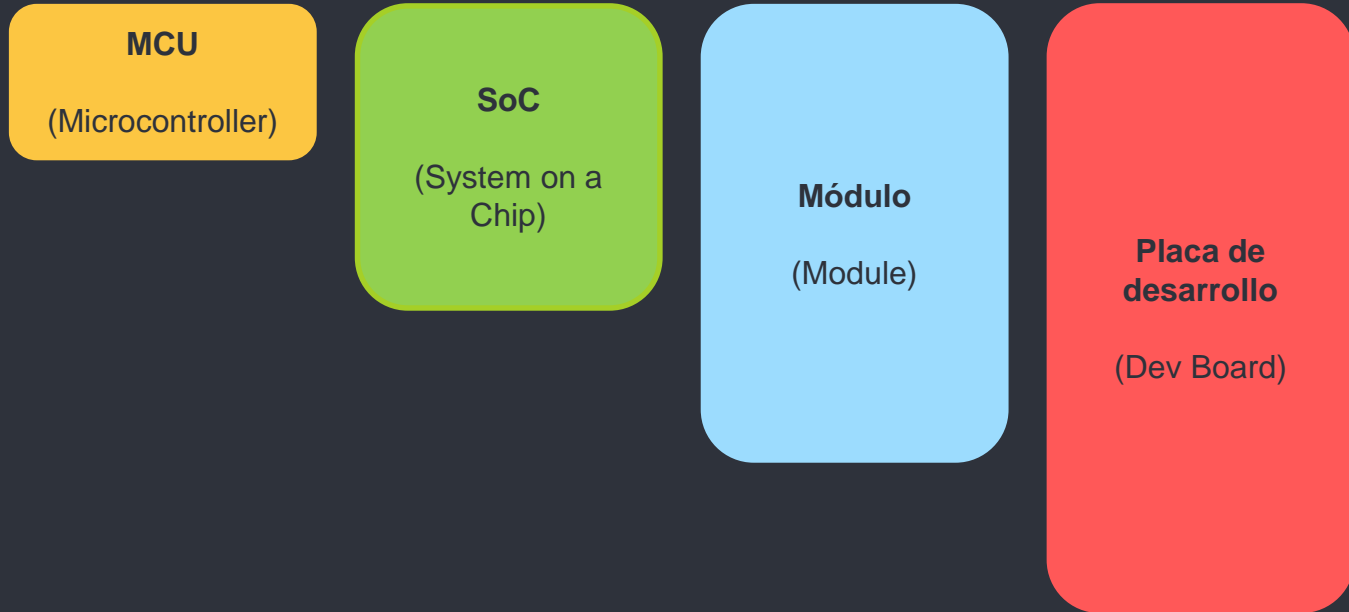
El pinout es distinto en ambos así que hay que tener eso en cuenta.



ESP32 DevKit V1 Pinout {



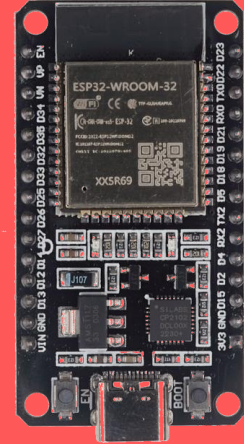
Conjuntos para desarrollar {



Conjuntos para desarrollar {



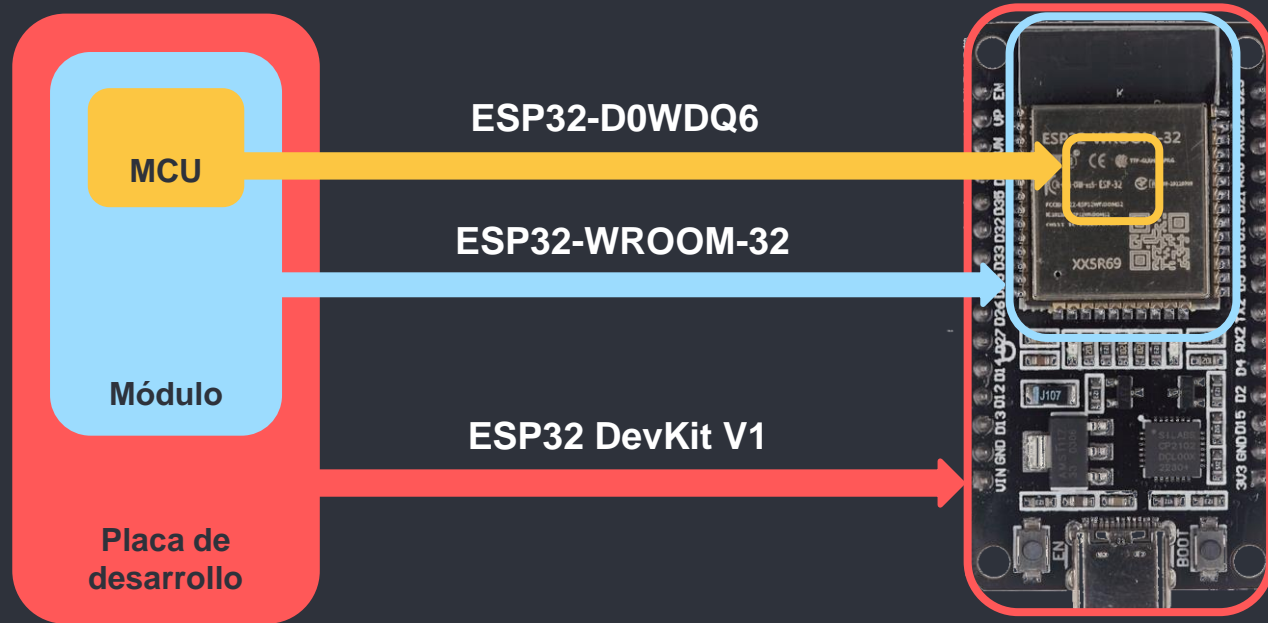
SoC
(System on a Chip)



}



Conjuntos para desarrollar {



Comparativa entre placas {

Característica	Arduino Uno	Arduino Mega	Raspberry Pi Pico (RP2040)	ESP32
Microcontrolador	ATmega328P	ATmega2560	RP2040 (Dual-core ARM Cortex-M0+)	Dual-core Tensilica Xtensa LX6
Velocidad de reloj	16 MHz	16 MHz	133 MHz	160 MHz o 240 MHz
Memoria Flash	32 KiB	256 KiB	2 MiB (QSPI Flash)	4 MiB o más
RAM	2 KiB SRAM	8 KiB SRAM	264 KiB SRAM	520 KiB SRAM
Wi-Fi	No	No	Sí (modelo W)	Sí
Bluetooth	No	No	Sí (modelo W)	Sí (Bluetooth y BLE)
Pines GPIO	14 pines digitales, 6 analógicos	54 pines digitales, 16 analógicos	26 GPIO	36 GPIO
ADC	6 canales de 10 bits	16 canales de 10 bits	3 canales de 12 bits	18 canales de 12 bits
DAC	No	No	No	2 canales de 8 bits
UART	1	4	2	3
I2C	1	1	2	2
SPI	1	1	2	4
Alimentación	5V o 3.3V	5V o 3.3V	1,8 V a 5.5V	3.3V
Soporte para RTOS	No	No	Sí (FreeRTOS)	Sí
Precio aproximado	~10 USD	~12-15 USD	~4-8 USD	~8-15 USD



Entorno de trabajo {

```
# Lo primero que tenemos que instalar son los drivers de los diferentes  
tipos de conversores usb que tenga nuestro dispositivo (por lo general son  
CH340, CP2102 o como en nuestra placa el CH9102X).
```

```
# https://www.wch-ic.com/downloads/CH343SER\_ZIP.html
```

Un problema común puede ser el cable. Hay dos tipos: aquellos que solo proporcionan energía (sirven únicamente para cargar dispositivos como teléfonos móviles) y están los que suministran energía y permiten la transferencia de datos. Es este último el cable que debemos usar.

Detectar este problema es fácil. Si tu placa se enciende (deberías de ver un LED iluminado), pero si no ves el dispositivo como una unidad flash USB conectada, entonces tengas el tipo incorrecto.

```
}
```



Thonny {



```
# Hay distintos IDEs con los cuales trabajar con  
MicroPython directamente. Entre ellos Thonny,  
uPyCraft, Mu, el propio Visual Studio Code con la  
extensión PlatformIO, etc.
```

```
# Para nuestras practicas usaremos Thonny, por su  
facilidad de uso y por ser el que mas opciones nos va  
a dar para trabajar en un principio. Una vez que  
tengamos en claro la metodología de trabajo, podríamos  
volver a Visual Studio Code con PlatformIO para  
disponer de todas las ventajas pero sabiendo que los  
procedimientos se hacen un poco mas complejos.
```

```
# Para instalar Thonny, nos dirigimos a su pagina web  
y descargamos la ultima versión disponible
```

```
# https://thonny.org/
```

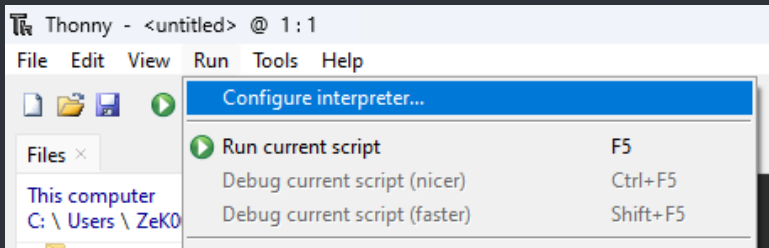


Firmware {

El firmware de MicroPython para ESP32 es una versión del interprete diseñada específicamente para funcionar en microcontroladores ESP32.

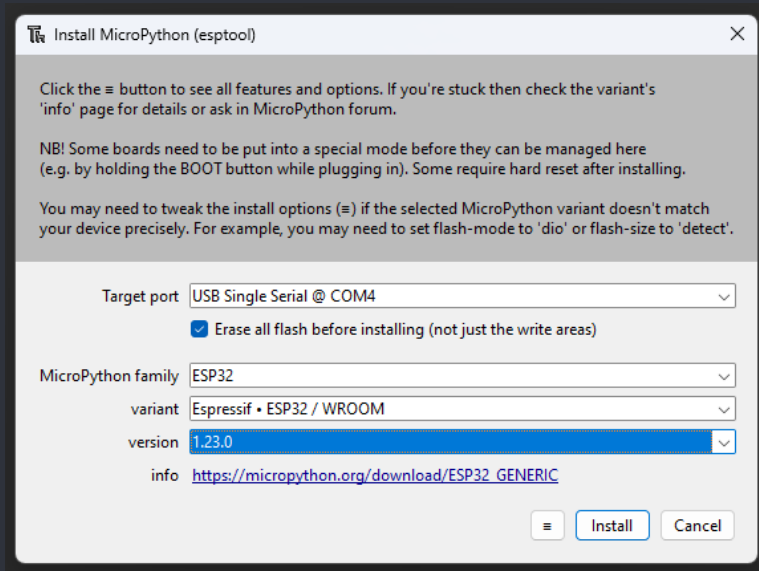
Para instalar el firmware tenemos varias alternativas, podemos descargarlo desde la pagina de MicroPython y luego grabarlo en la ESP32 mediante herramientas como ESP-IDF, esptools, etc. En nuestro caso, vamos a usar el IDE Thonny para hacer todo el proceso en un solo paso.

Para ello, vamos a iniciar Thonny y nos vamos a dirigir a Run -> Configure interpreter...



Firmware {

Luego seleccionamos el puerto COM que nos asigno la computadora cuando enchufamos la placa, completamos los demás datos y le damos click a Install.



Ejemplo {

```
# Vamos a verificar si todo funciona correctamente. Vamos a escribir  
nuestro primer programa.
```

```
from machine import Pin  
from time import sleep
```

```
# Configurar el pin GPIO 2 como salida (LED  
incorporado)  
led = Pin(2, Pin.OUT)
```

```
# Bucle infinito para parpadear el LED
```

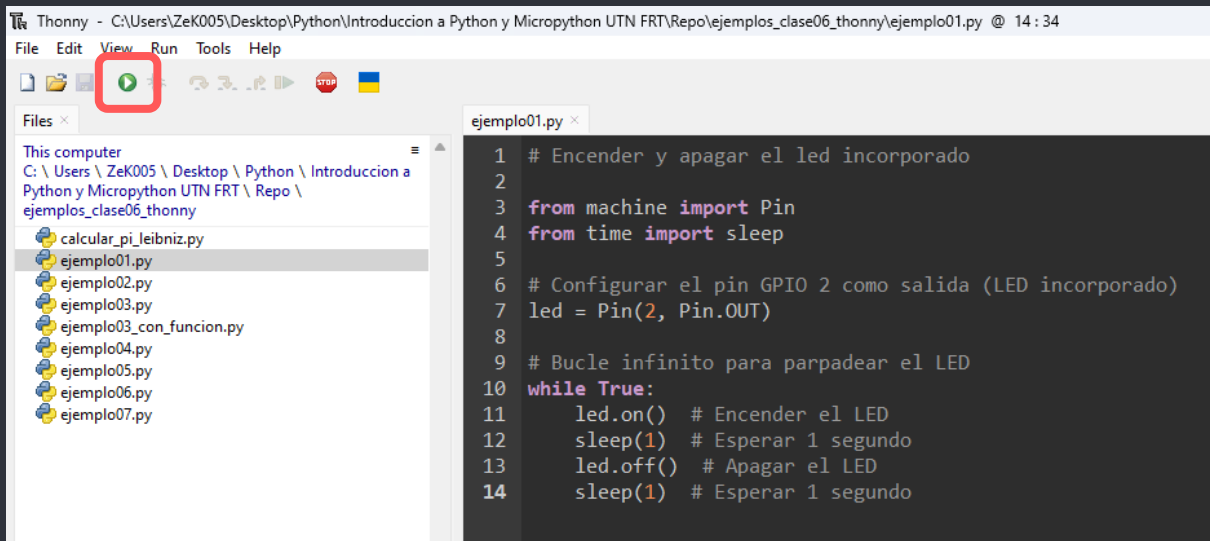
```
while True:  
    led.on()    # Encender el LED  
    sleep(1)    # Esperar 1 segundo  
    led.off()   # Apagar el LED  
    sleep(1)    # Esperar 1 segundo
```

```
}
```



Ejemplo {

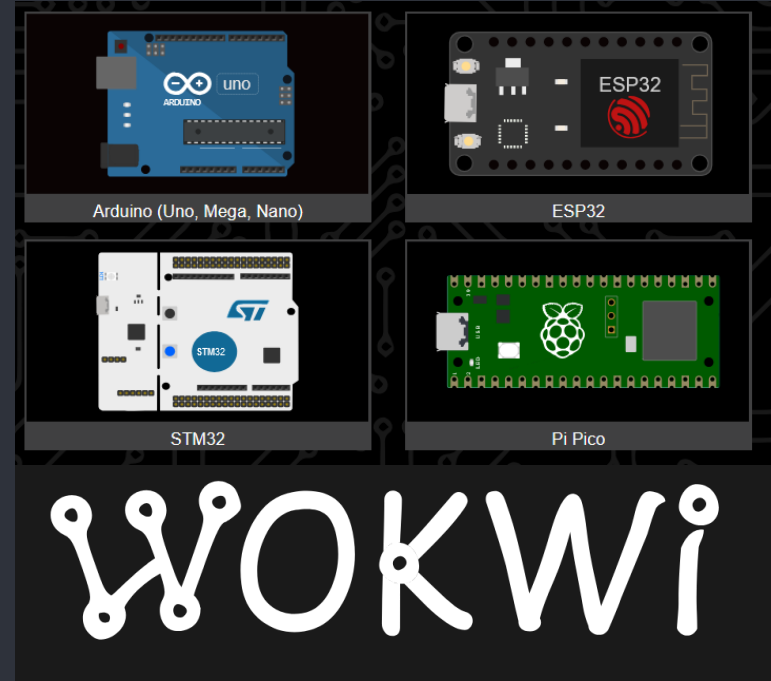
Una vez escrito el programa, vamos a hacer click al botón play verde y debería ejecutarse el código en la ESP32.



Wokwi {

Wokwi es una plataforma en línea que permite simular y desarrollar proyectos de electrónica y microcontroladores. Es especialmente útil para aquellos que desean experimentar y aprender sobre microcontroladores como el Arduino, ESP32, y otros, sin necesidad de hardware físico. Wokwi proporciona un entorno de simulación interactivo donde puedes escribir código, conectar componentes electrónicos y ver cómo interactúan en tiempo real. Entre los lenguajes soportados se encuentra MicroPython.

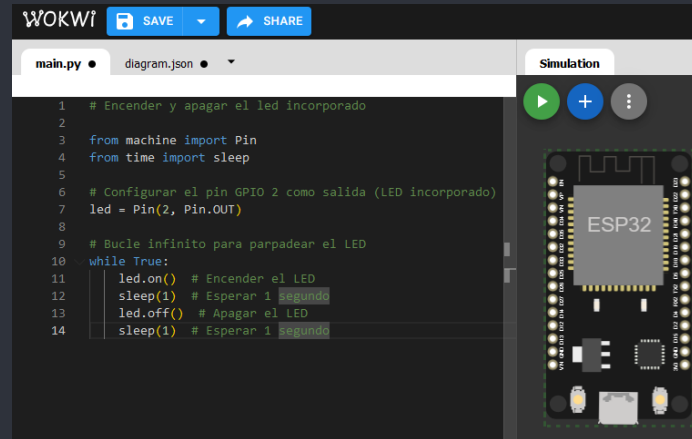
}



Wokwi {

Vamos a ir a la sección MicroPython y vamos a poder ver algún ejemplo ya armado o comenzar un diseño desde cero. Allí podremos probar nuestro código del ejemplo anterior y verificar si funciona correctamente.

Ejemplo listo para probar: <https://wokwi.com/projects/412197049127131137>



```
1
2
3 Aprender a programar
4
5 es aprender a pensar.
6
7
8
9
10
```

```
11 { Steve Jobs; }
12
13
14
```



```
1
2
3
4
5 { Nos vemos en la
6 proxima clase }
7
8
9
10
11
12
13
14
```

