MAE 6291
Internet of Things for Engineers

Prof. Kartik Bulusu, MAE Dept.

Week 11 [04/09/2025]	• Guest lecture: FPGAs…What are they, How do they work, What do we do with them today, and Where will they go tomorrow? by Prof. Thomas Farmer, U Penn. • Introduction to MicroPython	• Introduction to the ESP32 microcontroller • Setting up Micropython interpreter in Thonny IDE • Flashing the firmware on the ESP32 microcontroller • In-class lab set-up of ESP32 microcontroller with the Raspberry Pi 4B [Graded Lab Activity]

git clone https://github.com/gwu-mae6291-iot/spring2025_codes.git

School of Engineering & Applied Science
THE GEORGE WASHINGTON UNIVERSITY

Spring 2025

Photo: Kartik Bulusu

Differences Between a Microcontroller and a Microprocessor

Source:
https://raspberrytips.com/is-raspberry-pi-a-microcontroller/
https://libre.computer/products/aml-s905x-cc/
https://www.raspberrypi.com/products/raspberry-pi-4-model-b/
https://learn.sparkfun.com/tutorials/esp32-thing-plus-usb-c-hookup-guide/introduction

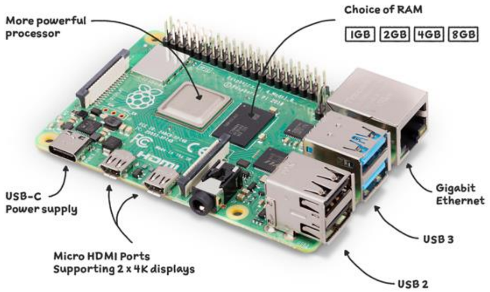
A microprocessor is the controlling unit of a micro-computer wrapped in a small chip, and it contains all the functions of a central processing unit of a computer.

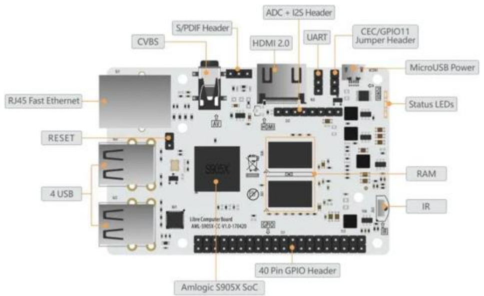
It performs Arithmetic Logical Unit (ALU) operations, and it communicates with other connected devices.

	Microprocessor	Microcontroller
Memory	Over 512 MB in general	Minimal, generally under 256 KB
Clock speed	Between 1 an 4 Ghz on average	Under 300 MHz in general
Power consumption	High	Low
Application	Complex tasks requiring calculations	Fixed and predefined task
Architecture	32 or 64 bits	8, 16 or 36 bits

A microcontroller is a chip that is optimized to control electronic devices. It is found in a single integrated circuit that is dedicated to performing a specific task.

It controls other portions of an electronic system, usually via a microprocessor unit.

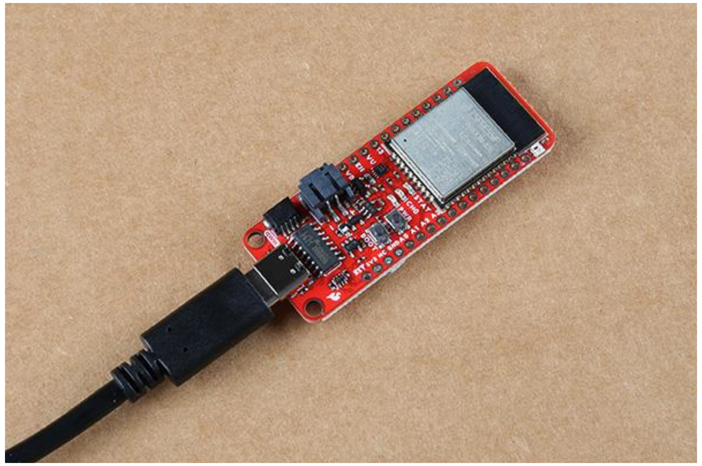


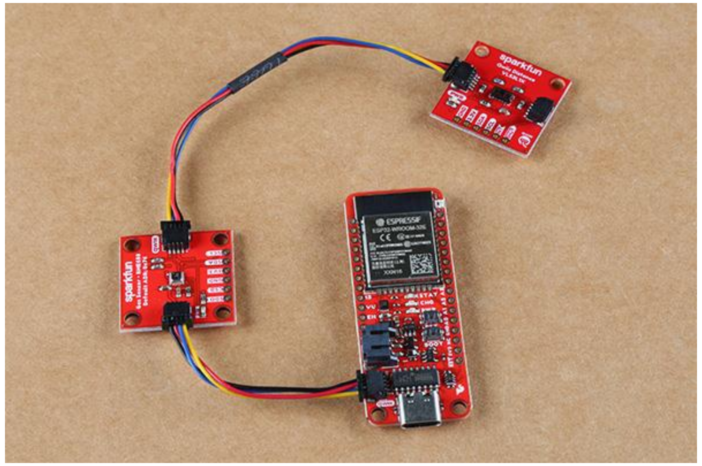


Setting up the ESP32 microcontroller with the Raspberry Pi 4B

Source:
https://learn.sparkfun.com/tutorials/esp32-thing-plus-usb-c-hookup-guide/introduction
https://www.sunfounder.com/products/esp-wroom-32-esp32-esp-32s-development-board?srsltid=AfmBOopfjt-0zffmxHW-zWDQZrbQpBt6vjv9RWp3Vs8OjC6g-_ldRPkz

ESP32 Microcontroller – A first look





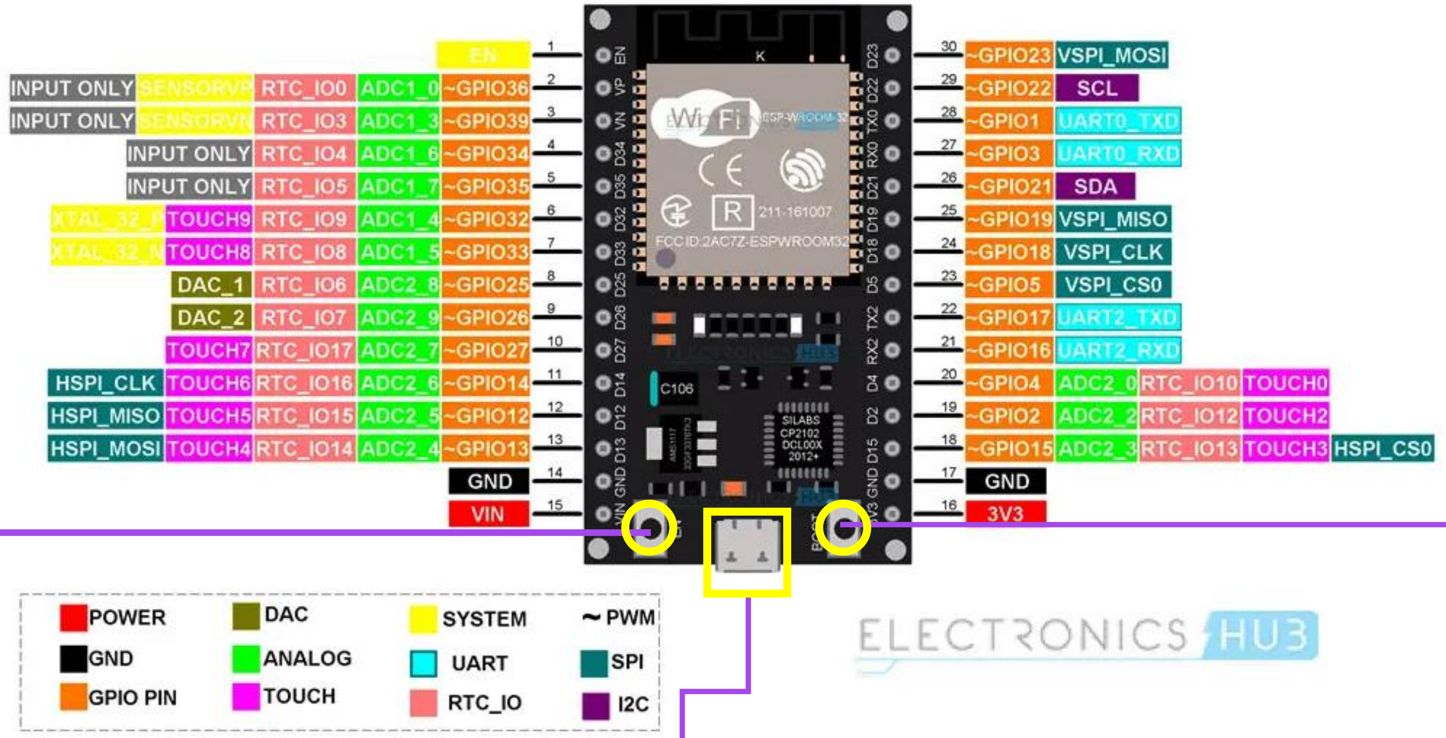
Source:
https://learn.sparkfun.com/tutorials/esp32-thing-plus-usb-c-hookup-guide/introduction
https://www.electronicshub.org/esp32-pinout/

ESP32-WROOM Microcontroller – Pin outs

ESP32-WROOM only requires 3.3V to power the board.

- The simplest method to power the board is through the USB-C connector.
- The VIN pin is designed to supply power to the board when not using USB power.
- When powered via USB, the VIN pin will output 5V, which can be used to power other modules.

ESP32-WROOM is only compatible with **2.4GHz WiFi** networks; it will not work on the 5GHz bands.



The Reset pin on all ESP32 chips (also known as the EN pin) is a power shutdown pin, that when pulled low (pulled to GND) cuts the power off to the ESP32

USB-C connector on the ESP32-WROOM

The BOOT button is also connected to GPIO 0
Note: GPIO0 shares the same pin as GPIO36 on this board

Sources:
https://lastminuteengineers.com/esp32-pinout-reference/

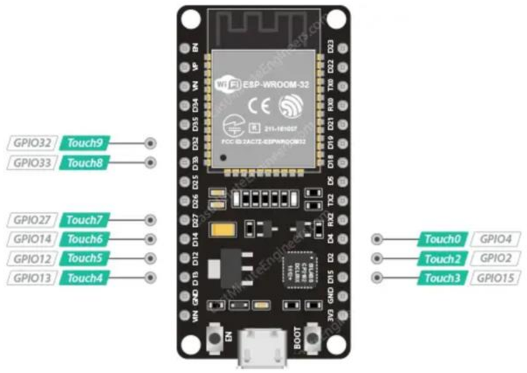
ESP32 Peripherals and I/O

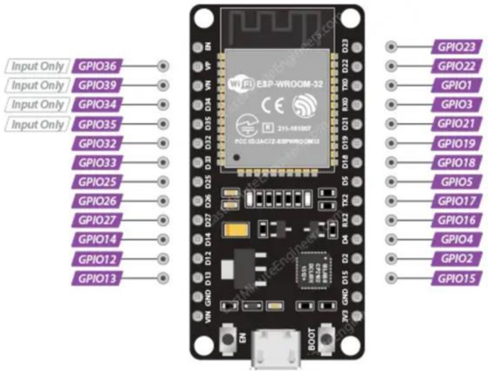
Although the ESP32 has 48 GPIO pins in total, only 25 of them are broken out to the pin headers on both sides of the development board.
These pins can be assigned a variety of peripheral duties, including:

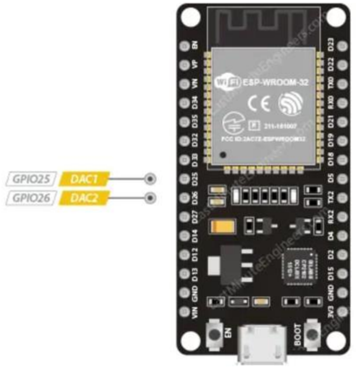
15 ADC channels	15 channels of 12-bit SAR ADC with selectable ranges of 0-1V, 0-1.4V, 0-2V, or 0-4V
2 UART interfaces	2 UART interfaces with flow control and IrDA support
25 PWM outputs	25 PWM pins to control things like motor speed or LED brightness
2 DAC channels	Two 8-bit DACs to generate true analog voltages
SPI, I2C and I2S interface	Three SPI and one I2C interfaces for connecting various sensors and peripherals, as well as two I2S interfaces for adding sound to your project
9 Touch Pads	9 GPIOs with capacitive touch sensing

– Pay close attention because their behavior, particularly during boot, can be unpredictable. Use them only when absolutely necessary.

– It is recommended that you avoid using these pins.







D2 / Pin 2 / GPIO2: Logic LOW during boot and **also connected to the on-board LED**

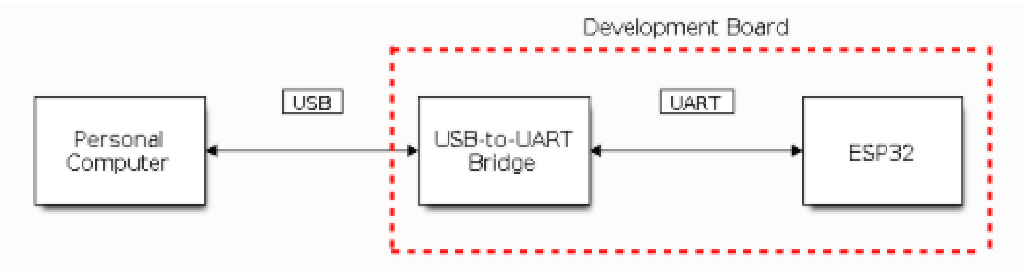
Step 0: Install CP210x USB-to-UART bridge drivers for ESP32 WROOM

Source:
https://cdn.sparkfun.com/assets/5/0/a/8/5/CH340DS1.PDF
https://learn.sparkfun.com/tutorials/esp32-thing-plus-usb-c-hookup-guide/introduction
https://docs.espressif.com/projects/esp-idf/en/stable/esp32/get-started/establish-serial-connection.html

>>> sudo apt-get update
>>> sudo apt-get upgrade

>>> git clone https://github.com/gwu-mae6291-iot/spring2025_codes.git

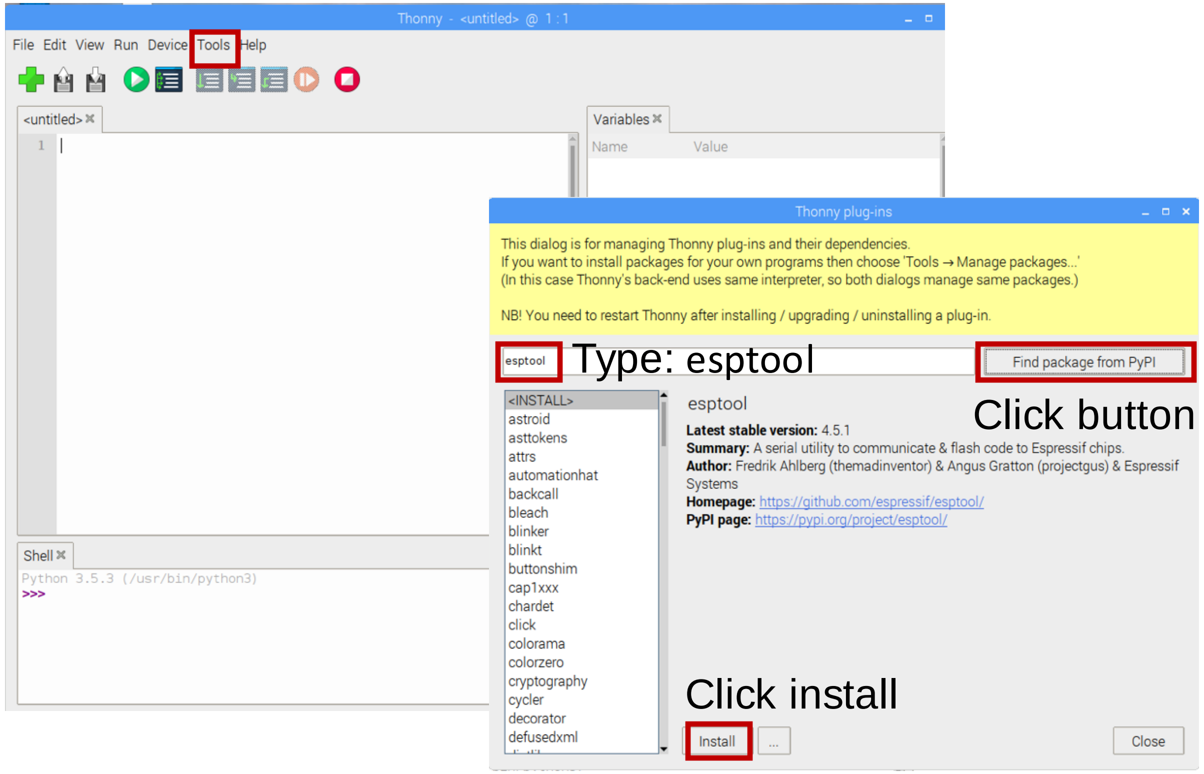
CP210x is a USB-to-UART bridge chip installed ESP32 development boards



Step 1: Check and Install **esptool.py** plug-in

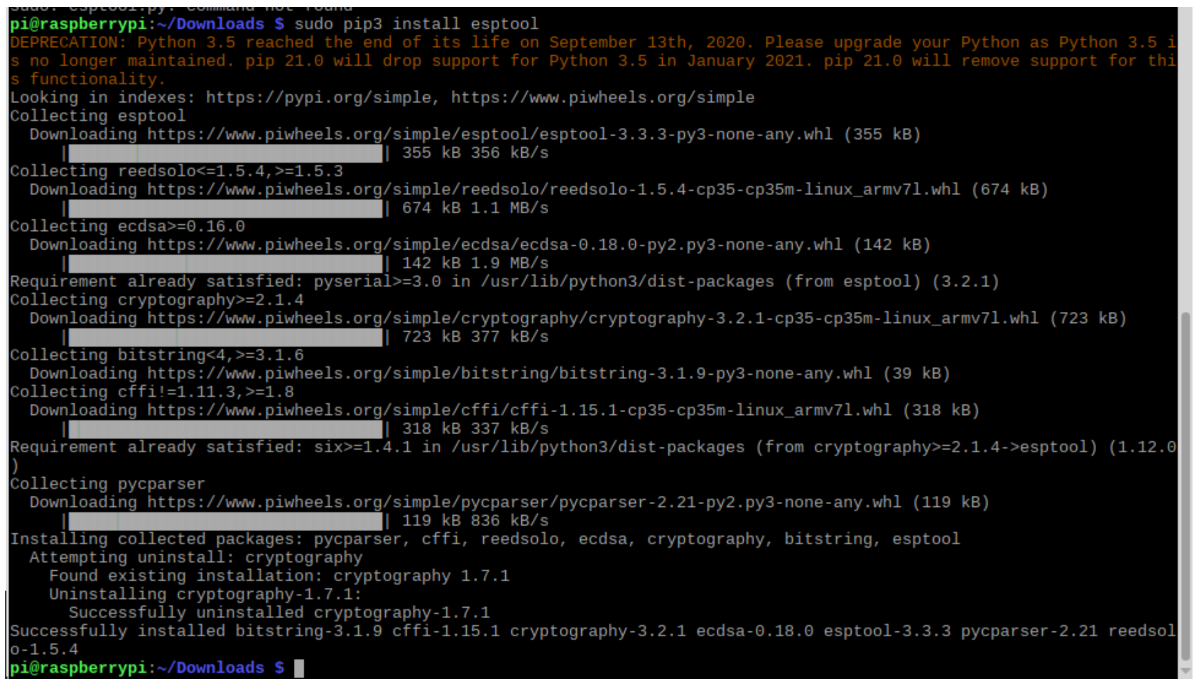
In Thonny

Click: Tools



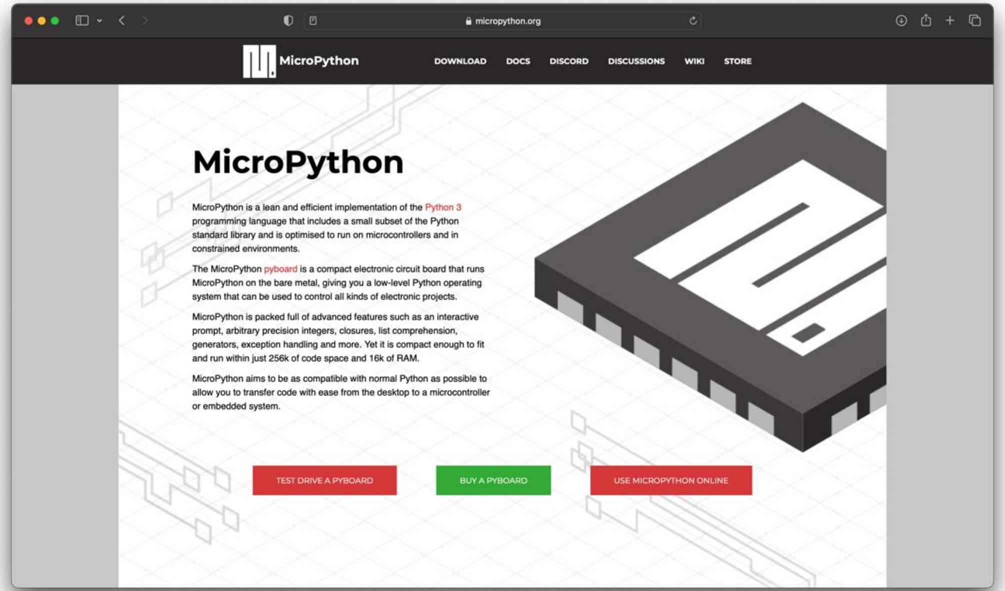
In Terminal

~~sudo~~ pip install esptool



Source:
Folder by Colourcreatype from https://thenounproject.com/browse/icons/term/folder/
Micropython https://micropython.org

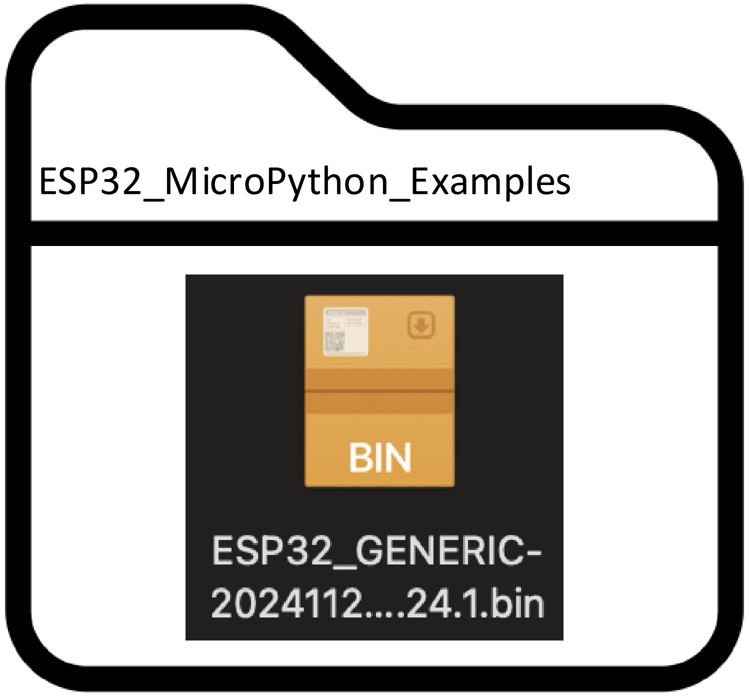
Step 2: Download generic ESP32 firmware from https://micropython.org/download/esp32/ [Provided to you with the github repo]



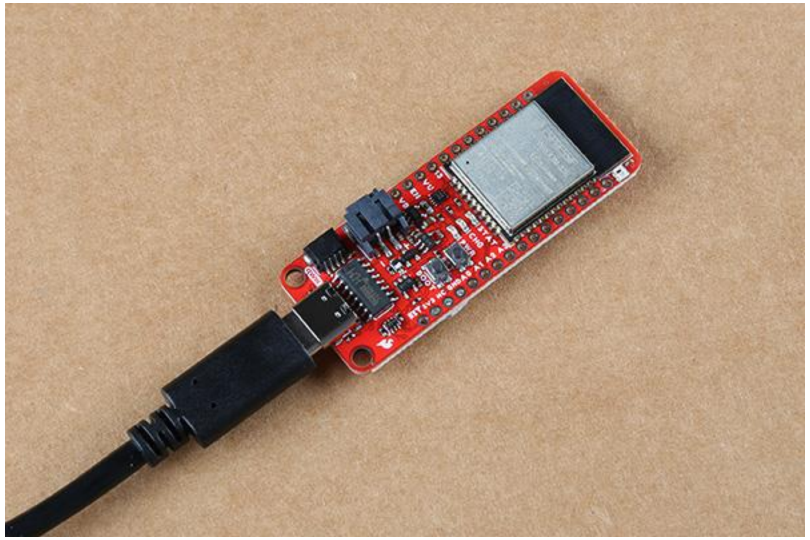
Firmware

Releases

v1.24.1 (2024-11-29) .bin / [.app-bin] / [.elf] / [.map] / [Release notes] (latest)
v1.24.0 (2024-10-25) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
v1.23.0 (2024-06-02) .bin / [.app-bin] / [.elf] / [.map] / [Release notes]
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]
v1.20.0 (2023-04-26) .bin / [.elf] / [.map] / [Release notes]
v1.19.1 (2022-06-18) .bin / [.elf] / [.map] / [Release notes]



Source:
Folder by Colourcreatype from https://thenounproject.com/browse/icons/term/folder/
Micropython https://micropython.org
https://learn.sparkfun.com/tutorials/esp32-thing-plus-usb-c-hookup-guide/introduction



ls /dev/tty*

pi@raspberrypi: ~

File Edit Tabs Help

```
pi@raspberrypi:~ $ ls /dev/tty*
/dev/tty    /dev/tty19  /dev/tty3   /dev/tty40  /dev/tty51  /dev/tty62
/dev/tty0   /dev/tty2   /dev/tty30  /dev/tty41  /dev/tty52  /dev/tty63
/dev/tty1   /dev/tty20  /dev/tty31  /dev/tty42  /dev/tty53  /dev/tty7
/dev/tty10  /dev/tty21  /dev/tty32  /dev/tty43  /dev/tty54  /dev/tty8
/dev/tty11  /dev/tty22  /dev/tty33  /dev/tty44  /dev/tty55  /dev/tty9
/dev/tty12  /dev/tty23  /dev/tty34  /dev/tty45  /dev/tty56  /dev/ttyAMA0
/dev/tty13  /dev/tty24  /dev/tty35  /dev/tty46  /dev/tty57  /dev/ttyprintk
/dev/tty14  /dev/tty25  /dev/tty36  /dev/tty47  /dev/tty58  /dev/ttyS0
/dev/tty15  /dev/tty26  /dev/tty37  /dev/tty48  /dev/tty59  /dev/ttyUSB0
/dev/tty16  /dev/tty27  /dev/tty38  /dev/tty49  /dev/tty6
/dev/tty17  /dev/tty28  /dev/tty39  /dev/tty5   /dev/tty60
/dev/tty18  /dev/tty29  /dev/tty4   /dev/tty50  /dev/tty61
pi@raspberrypi:~ $
```

Flashing ESP32 using Thonny IDE

- Step 3: Connect the ESP32 microcontroller using the USB cable provided and "erase the flash"

~~sudo~~ esptool.py --port /dev/ttyUSB0 erase_flash

pi@raspberrypi: ~

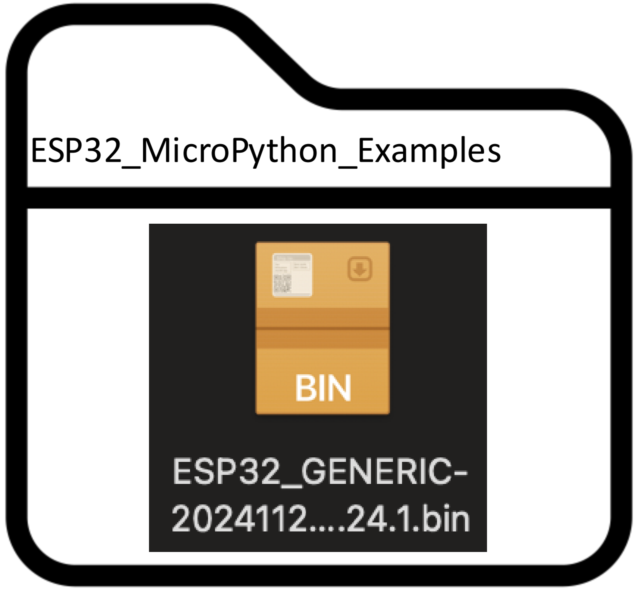
File Edit Tabs Help

```
pi@raspberrypi:~ $ sudo esptool.py --port /dev/ttyUSB0 erase_flash
esptool.py v4.7.0
Serial port /dev/ttyUSB0
Connecting....
Detecting chip type... Unsupported detection protocol, switching and trying agai
n...
Connecting....
Detecting chip type... ESP32
Chip is ESP32-D0WD-V3 (revision v3.0)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme
None
Crystal is 40MHz
MAC: 94:e6:86:92:d2:7c
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 55.3s
Hard resetting via RTS pin...
```

Source:
Folder by Colourcreatype from https://thenounproject.com/browse/icons/term/folder/
Micropython https://micropython.org

Step 4: Deploy the new firmware using the downloaded binary file

sudo esptool.py --chip esp32 --port /dev/ttyUSB0 --baud 460800 write_flash -z 0x1000 ESP32_GENERIC-20241129-v1.24.1.bin



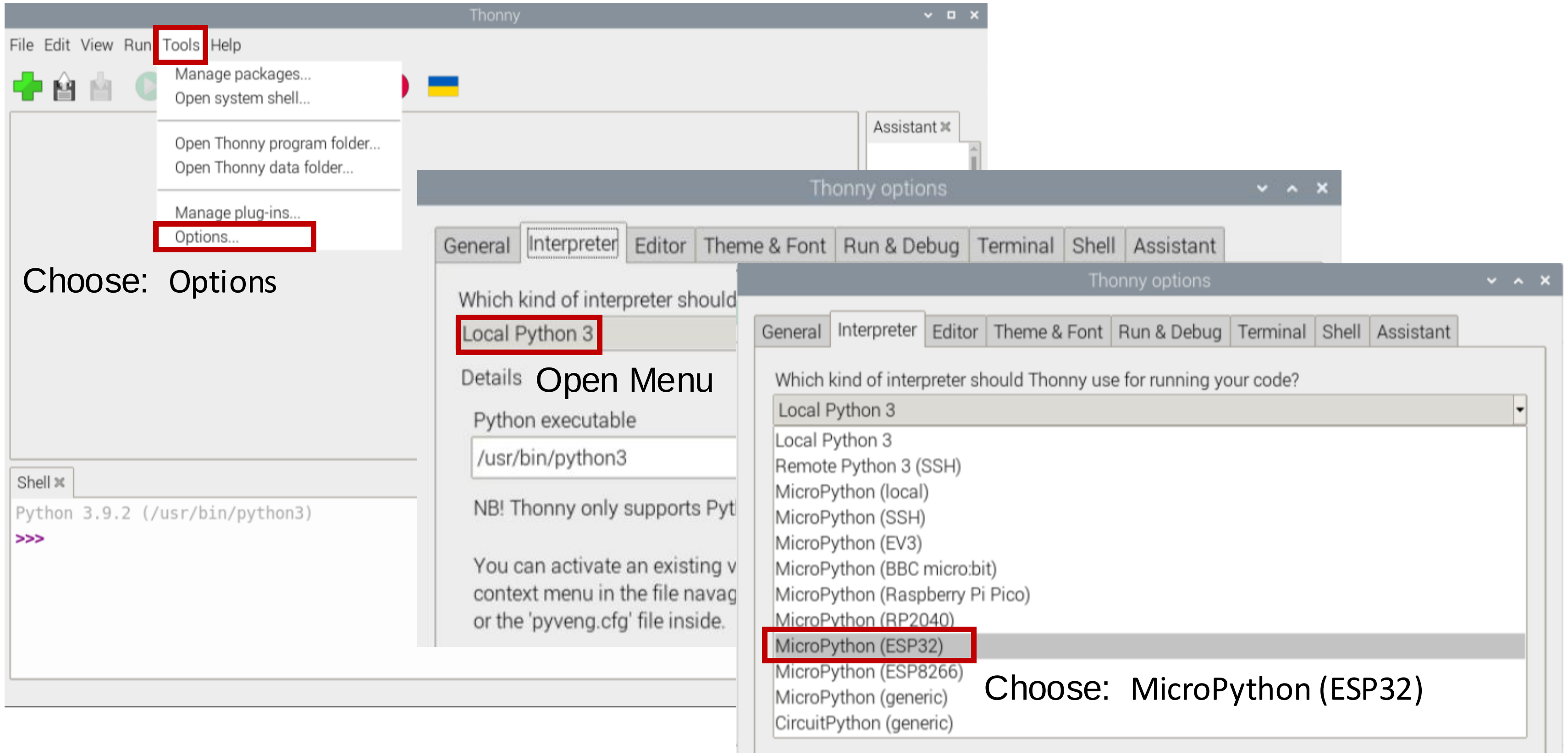

```
pi@raspberrypi:~/Desktop/Spring2024/MicropythonESP32 $ esptool.py --chip esp32 --port /dev/ttyUSB0 --baud 46080
0 write_flash -z 0x1000 ESP32_GENERIC-IDF3-20210202-v1.14.bin
esptool.py v4.6.2
Serial port /dev/ttyUSB0
Connecting....
Chip is ESP32-D0WD-V3 (revision v3.0)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 94:e6:86:92:d2:7c
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Flash will be erased from 0x00001000 to 0x00161fff...
Compressed 1445632 bytes to 925476...
Wrote 1445632 bytes (925476 compressed) at 0x00001000 in 21.8 seconds (effective 531.0 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
pi@raspberrypi:~/Desktop/Spring2024/MicropythonESP32 $
```

Sources:
https://micropython.org
https://en.wikipedia.org/wiki/MicroPython
https://upload.wikimedia.org/wikipedia/commons/4/4e/Micropython-logo.svg

Step 5: Start MicroPython interpreter on Thonny ID

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.

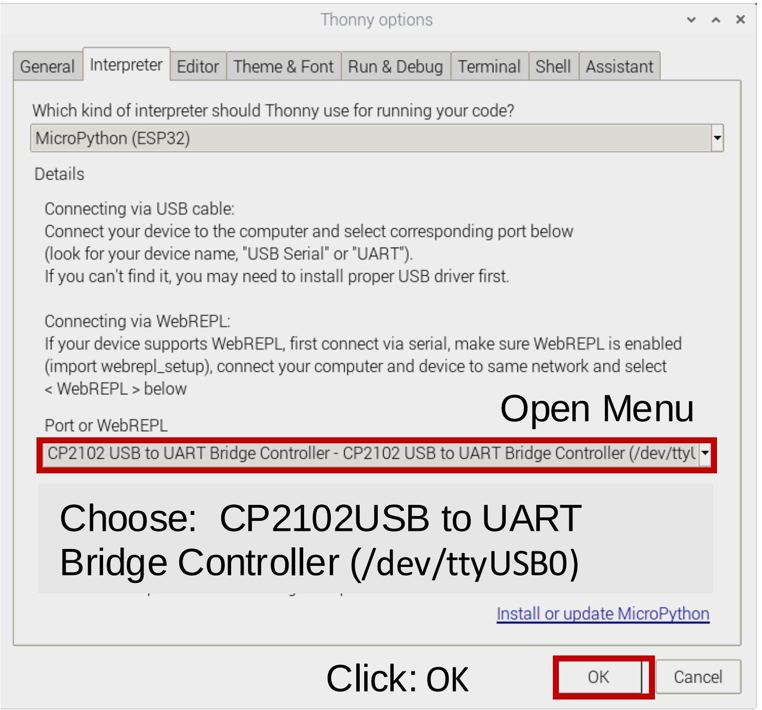
Click: Tools



Choose: Options

Open Menu

Choose: MicroPython (ESP32)



Open Menu

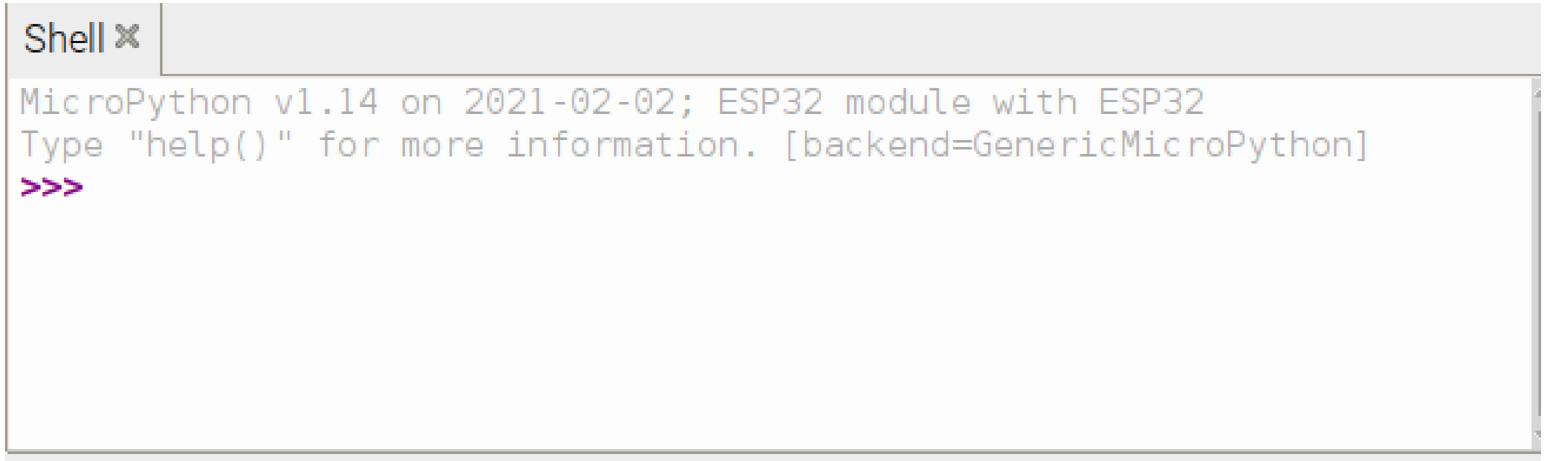
Choose: CP2102USB to UART Bridge Controller (/dev/ttyUSB0)

Click: OK

You are all set up to use MicroPython interpreter on Thonny IDE

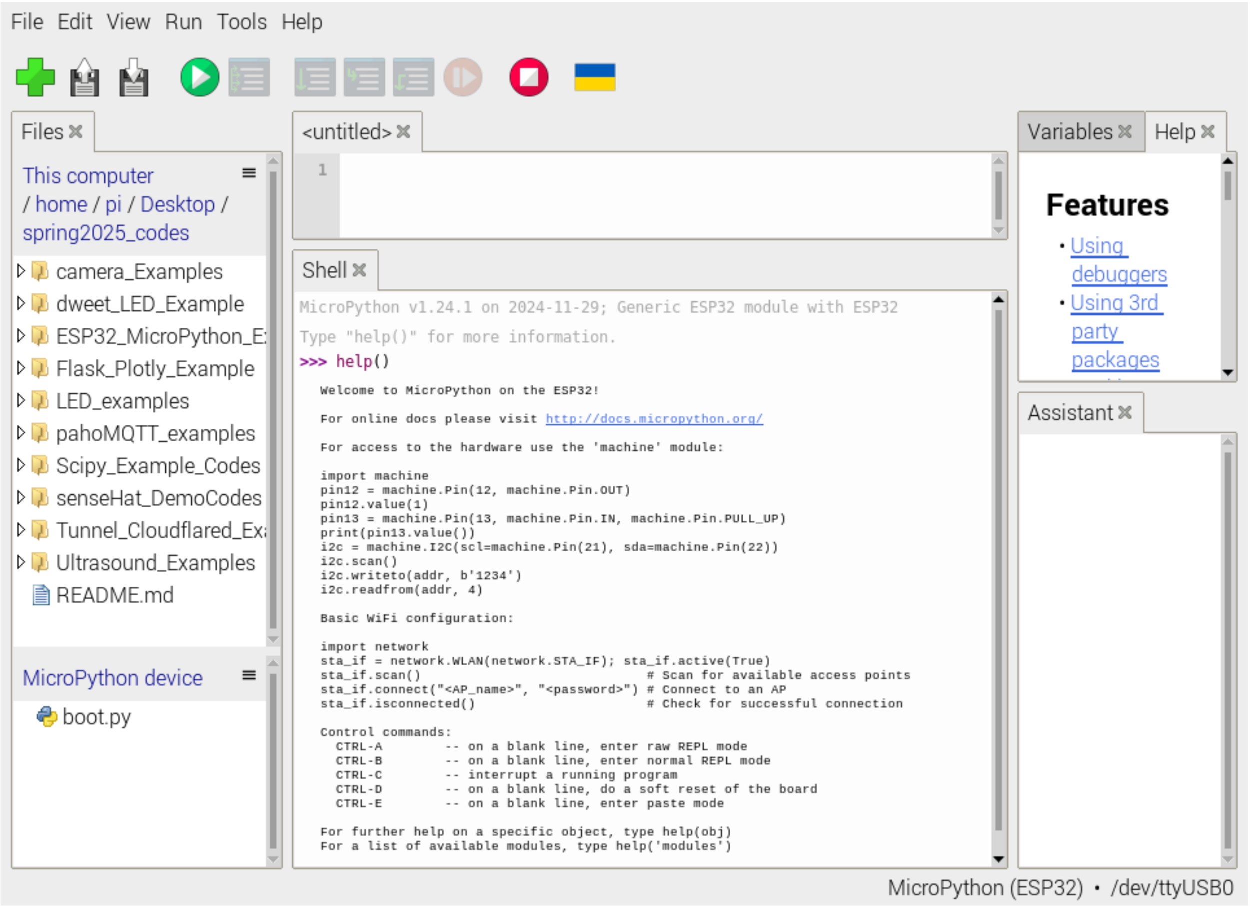
Sources:
https://micropython.org
https://en.wikipedia.org/wiki/MicroPython
https://upload.wikimedia.org/wikipedia/commons/4/4e/Micropython-logo.svg

Thonny is set to use the Micropython interpreter

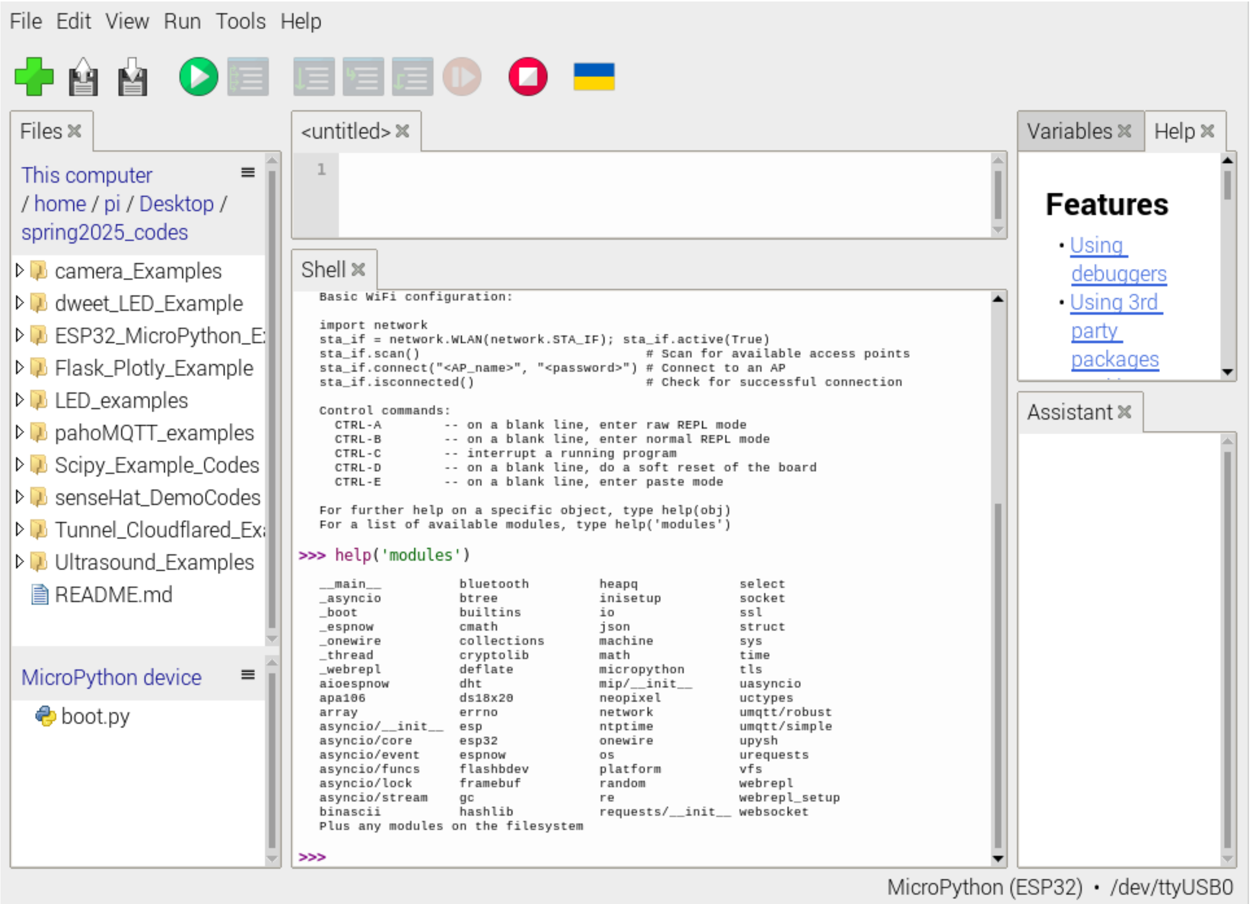


Step-6: Your first steps in MicroPython

>>> help()



>>> help('modules')

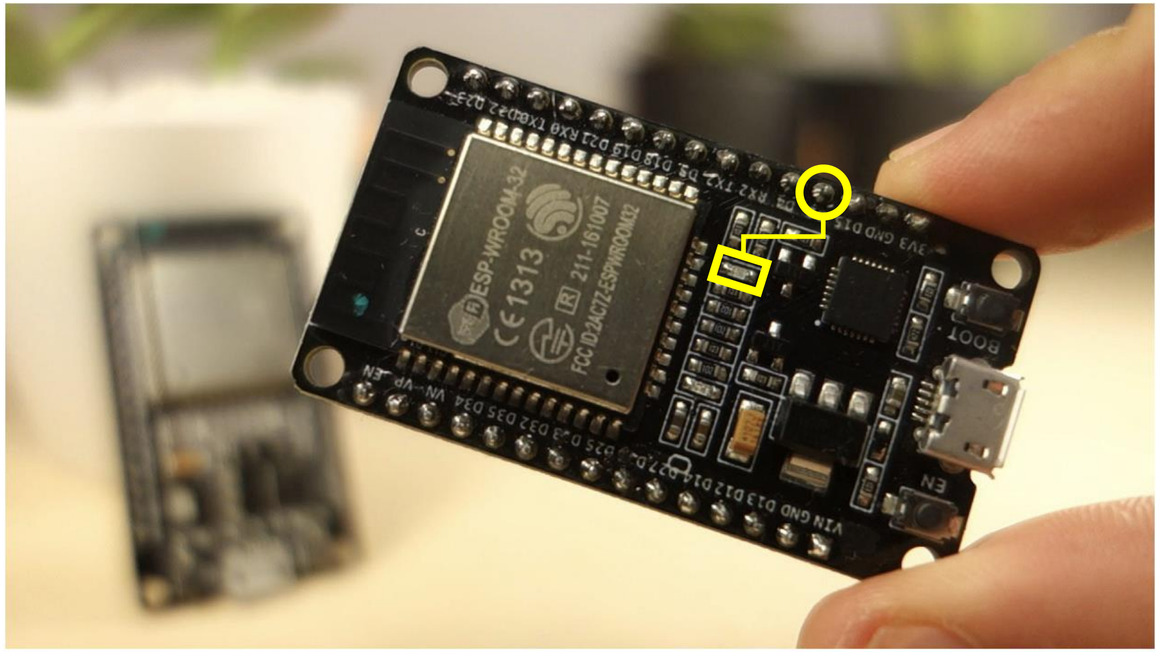


Step-7: Light up the on-board LED using MicroPython

Sources:
https://lastminuteengineers.com/esp32-pinout-reference/

D2 / Pin 2 / GPIO2: Logic LOW during boot and **also connected to the on-board LED**

```
MicroPython v1.24.1 on 2024-11-29; Generic ESP32 module with ESP32
Type "help()" for more information.
>>> from machine import Pin
>>> led = Pin(2, Pin.OUT)
>>> led.on()
>>> led.off()
>>> led.value(1)
>>> led.value(0)
>>> led.value(True)
>>> led.value(False)
```



Setting up the ESP32 Webserver and Access Point

- You will need to execute Python codes using the Micropython interpreter on Thonny
- Git-clone codes provided to you
- You will need two codes that should be flashed to the ESP32 from the Raspberry Pi 4B
  - boot.py
  - main.py
- You can work in groups if you like to complete the graded in-class exercise [10 points]