Embedded System Software

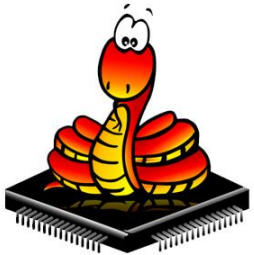
Lecture 11 : Introduction to MicroPython

Nattapong Wattanasiri

What is MicroPython?

- Reimplementation of **Python 3** that can be run on **MCU**. (MCU version of the Python)
- Small footprint interpreter as firmware.

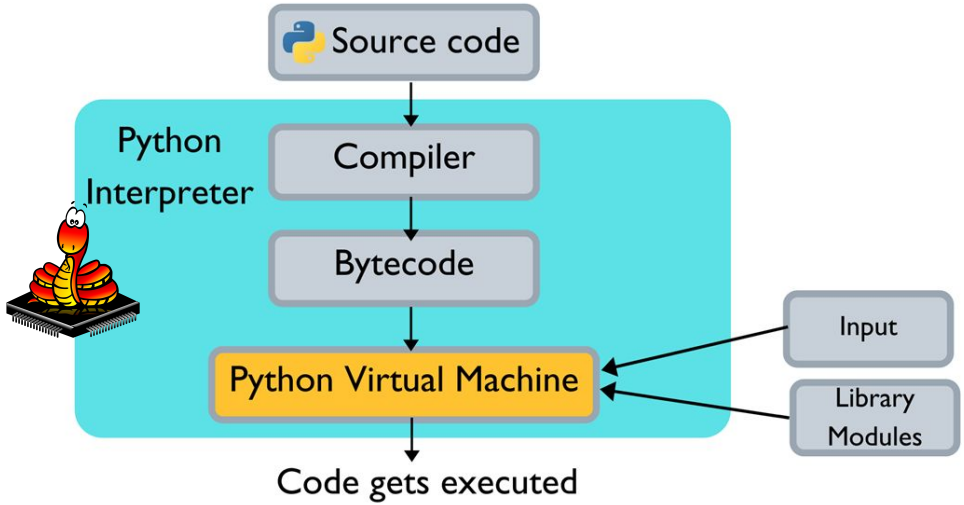
MicroPython



"it is compact enough to fit and run within just 256k of code space and 16k of RAM"

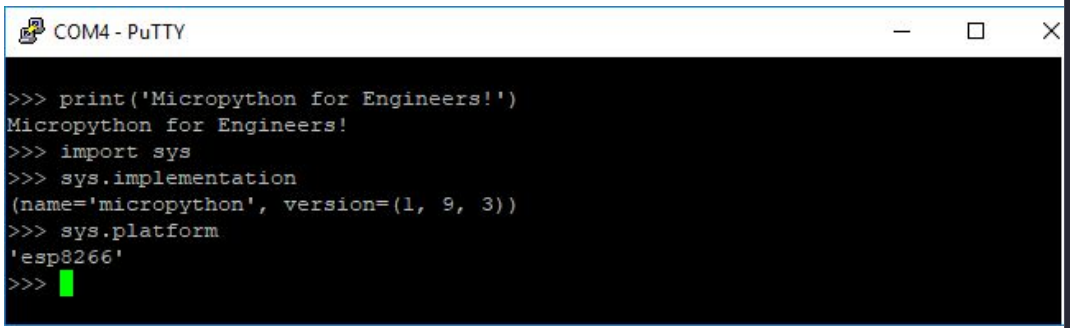
What is MicroPython?

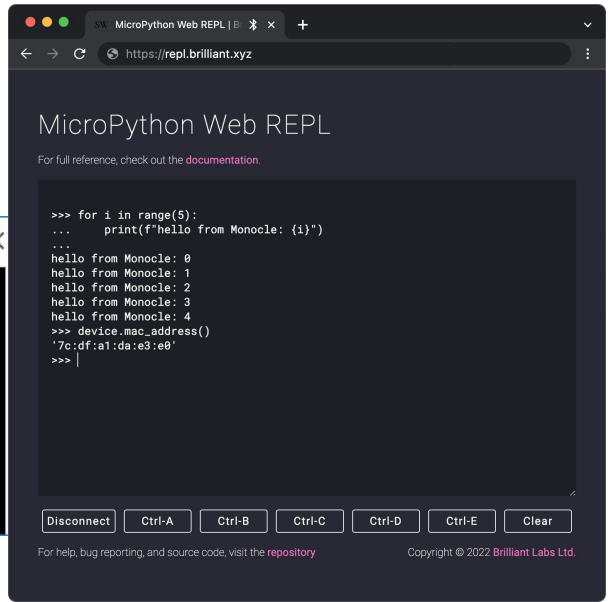
- Interpreter as firmware.
- Run “bare-metal” directly on hardware.
- View as **Operating System**



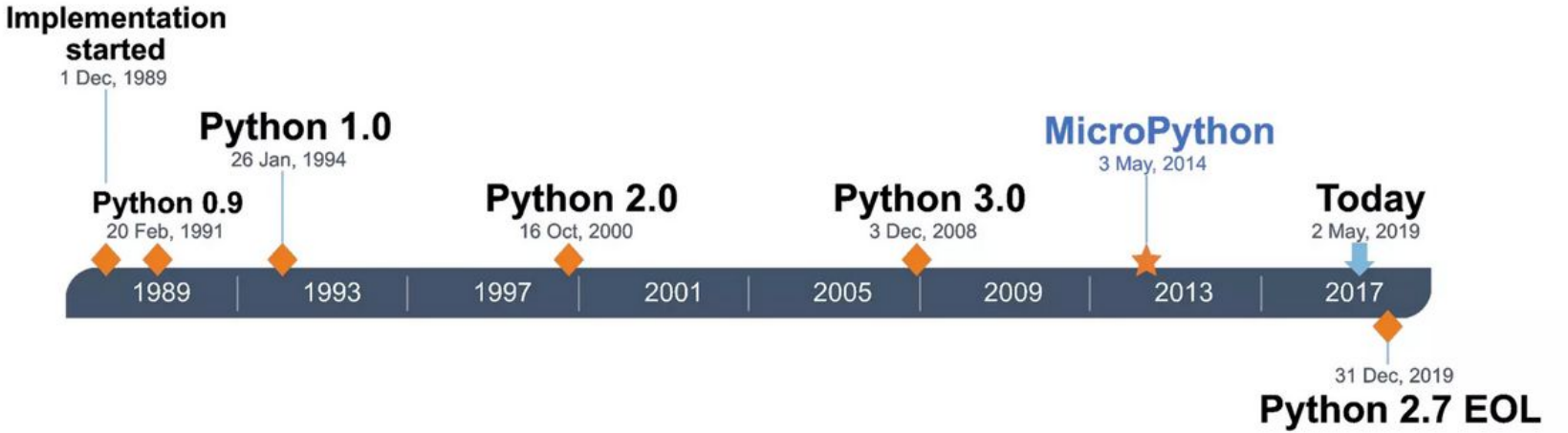
REPL

- Read-Evaluate-Print-Loop
- Interactive MicroPython Prompt for User
- Via wire (UART) or wireless (WIFI)
- Active when the main script (main.py) has ended.





MicroPython Timeline



Why Python on MCU?

- Python is an **easy-to-learn**. Easy to write.
- Reduce a **development time** for Proof-of-Concept & Prototyping.
  -> Best for Rapid Prototyping!

Why Python on MCU?

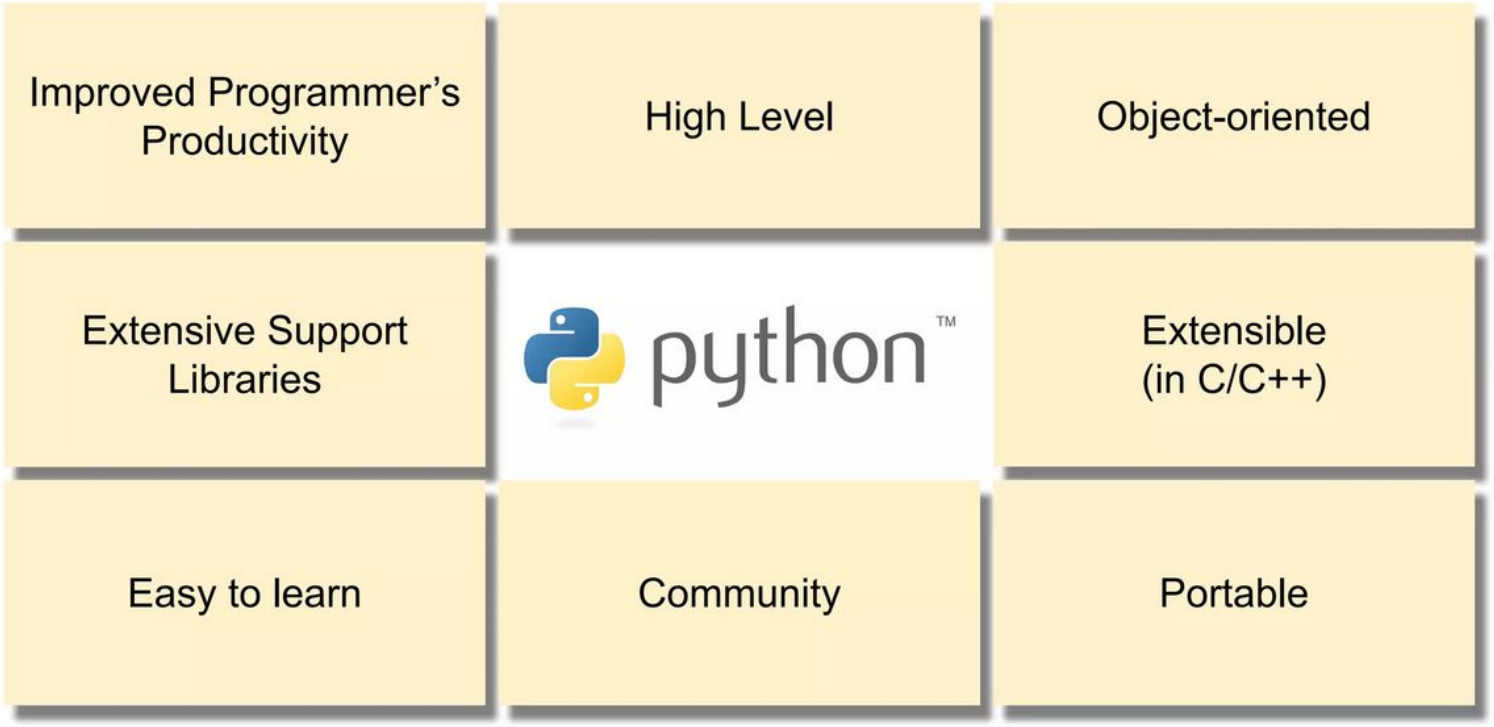
Since MicroPython is Python 3, you get:

- Python's style of object orientation (but without metaclasses)
- Data types (like unicode strings, integers, and floating-point numbers) and data structures (like lists, sets, and dictionaries)
- The highly dynamic nature of Python objects
- Functions as first-class objects
- Exception handling (`try`, `except`, `finally`, and the standard built-in exception classes)
- Fun features like generator functions (using the `yield` keyword), generator expressions, and list comprehensions
- The new `async` and `await` keywords in the very latest versions of MicroPython
- A comprehensive number of Python's built-in functions

Why Python on MCU?

Figure 1-1. In 2016 Python was ranked the third most popular programming language in the world by the IEEE. Guido van Rossum (the inventor of Python) correctly points out the omission of the "Embedded" flag thanks to MicroPython.[4]

Why Python on MCU?

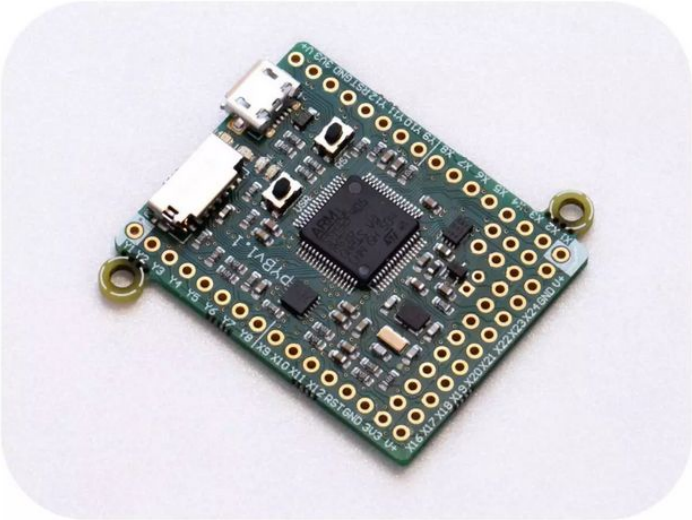


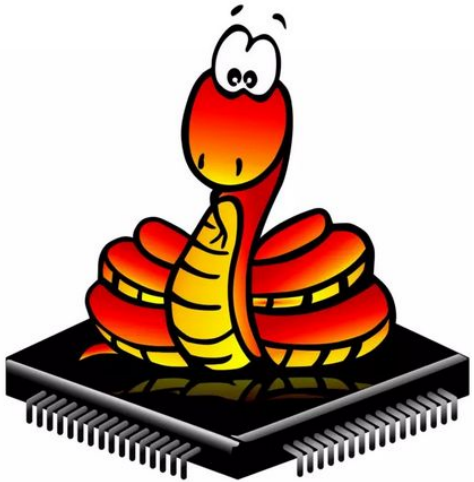
MicroPython Genesis

- Created by **Damien George**, Australian physicist and Programmer.
- He wondered if it would be possible to write a version of Python for microcontrollers.

MicroPython Genesis

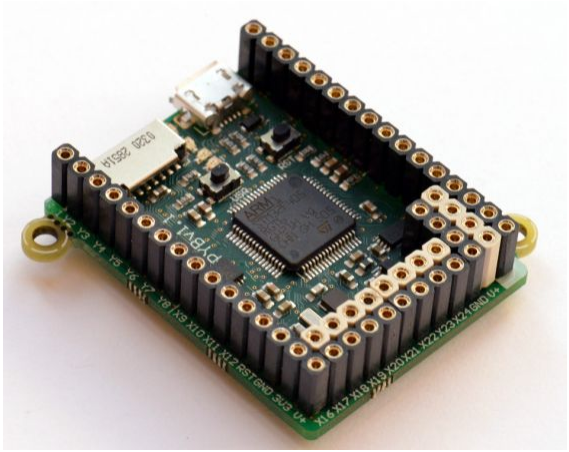
Kickstarter 2013 : PyBoard + MicroPython





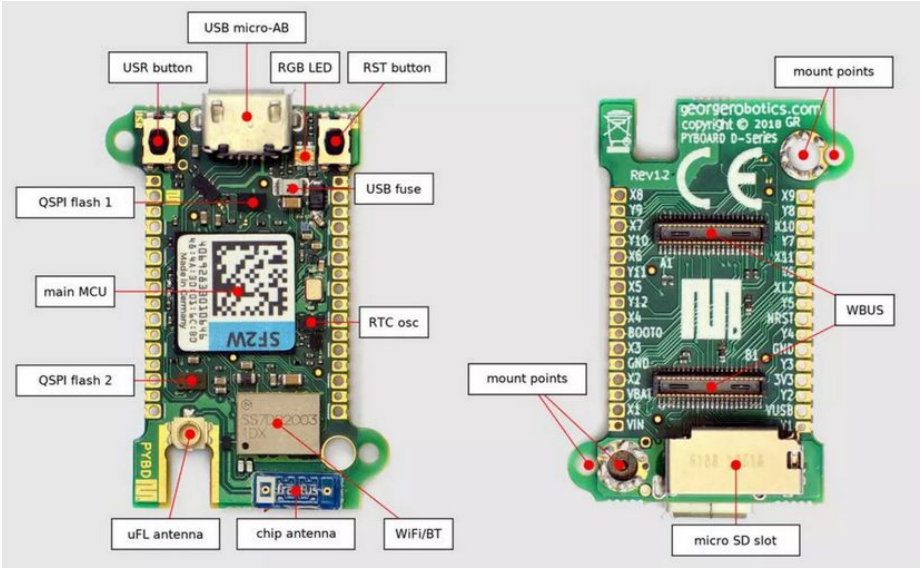
PyBoard

- STM32F405RG microcontroller, 168 MHz Cortex M4 CPU
- 1,024 Kb flash ROM, and 192 Kb RAM
- USB interface
- LEDs, Switches, RTC, Accelerometer Sensor



PyBoard-D series (2019)

- More powerful
- 216 MHz Cortex M7 CPU with double-precision hardware floating point
- 2048KiB flash ROM, 512KiB RAM



Flashing Micropython into PyBoard

- Use STM32's **DFU** (Device Firmware Update) feature.
- Connect DFU pin to the 3.3V to upload new firmware via USB port.

Figure 2-4. The adjacent DFU and 3.3 V pins on the front of the PyBoard

Flashing Micropython into PyBoard

- DFU Utility Software is used to upload MicroPython firmware. Such as **dfu-util**

```
$ sudo dfu-util --alt 0 -D firmware.dfu
```

Downloaded MicroPython firmware from website.

DFU Mechanism

- Ref. from STMicro

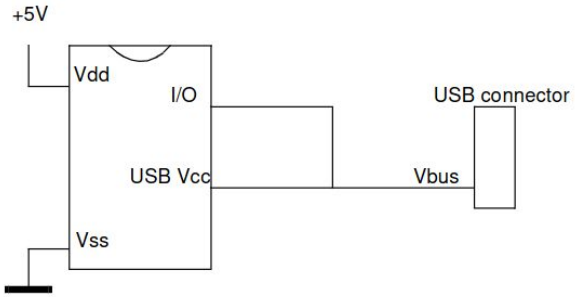
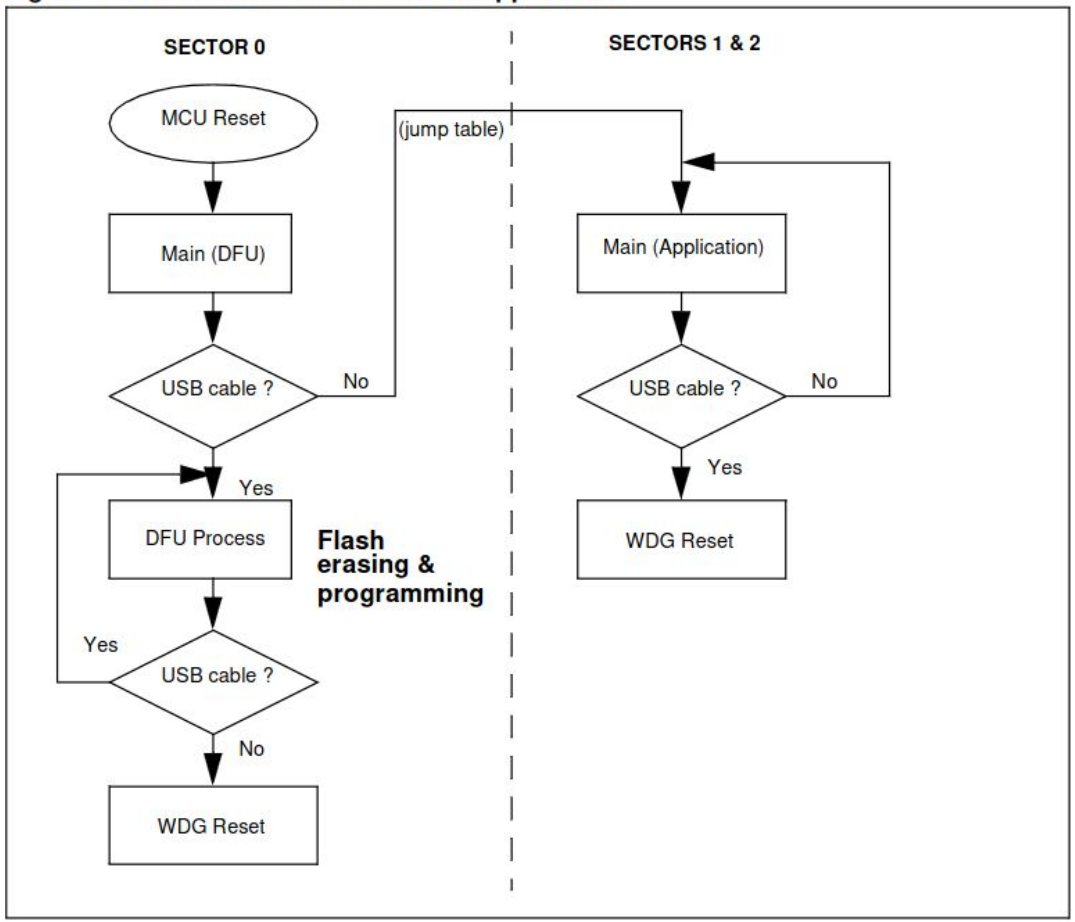


Figure 1. DFU Flowchart in a non-USB application

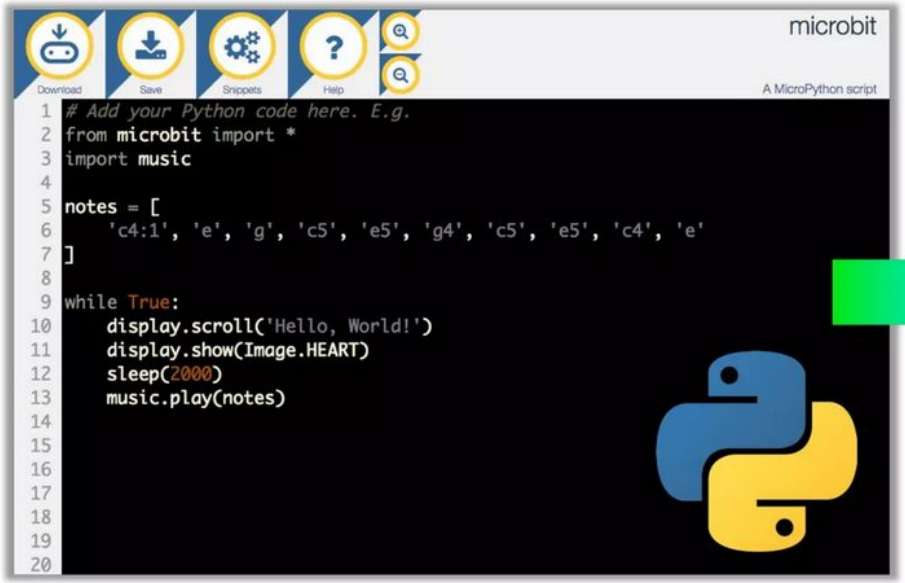


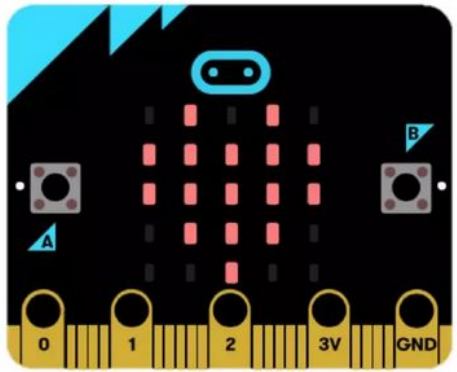
.dfu vs .uf2 file?

- Instead of using dfu-util, Just **Drag-and-Drop** a firmware like how the USB-drive works.
- UF2 is a file format designed by Microsoft that stands for **USB Flashing Format**.
- **UF2 Bootloader** is needed to be installed in MCU first.

micro:bit 2016

- Beginner-friendly Editor, Good for education tools

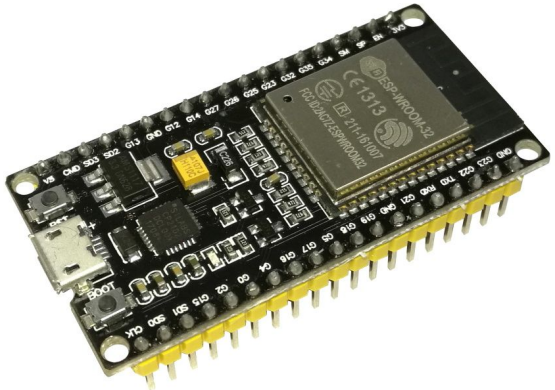




nRF51822, 16 MHz ARM Cortex-M0

Micropython on ESP32

- Upload micropython firmware with “**esptool**”
  (pip install esptool, or see esp-idf installation)



Esptool.py Documentation

This is the documentation for `esptool.py` - a Python-based, open source, platform independent utility to communicate with the ROM bootloader in Espressif SoCs.

`esptool.py`, `espefuse.py` and `espsecure.py` are a complete toolset for working with Espressif chips. They can do a number of things, for example:

- Read, write, erase, and verify binary data stored in flash.
- Read chip features and other related data such as MAC address or flash chip ID.
- Read and write the one-time-programmable efuses.
- Prepare binary executable images ready for flashing.
- Analyze, assemble, and merge binary images.

Micropython on ESP32

https://micropython.org/download/esp32

Firmware

Releases

v1.19.1 (2022-06-18) .bin [.elf] [.map] [Release notes] (latest)
v1.18 (2022-01-17) .bin [.elf] [.map] [Release notes]
v1.17 (2021-09-02) .bin [.elf] [.map] [Release notes]
v1.16 (2021-06-23) .bin [.elf] [.map] [Release notes]
v1.15 (2021-04-18) .bin [.elf] [.map] [Release notes]

Installation instructions

Program your board using the esptool.py program, found here.

If you are putting MicroPython on your board for the first time then you should first erase the entire flash using:

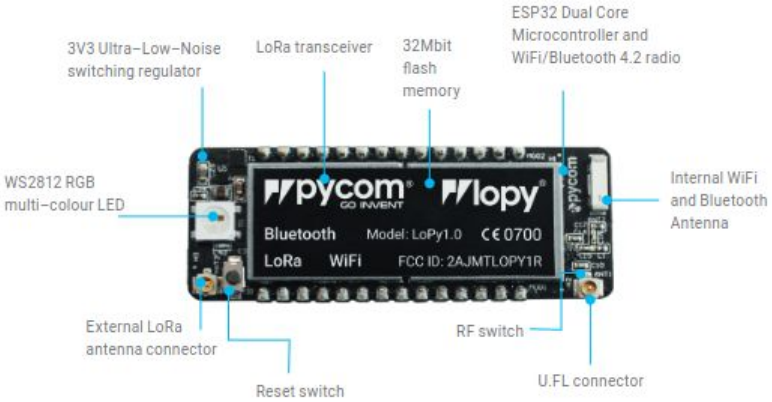
```
esptool.py --chip esp32 --port /dev/ttyUSB0 erase_flash
```

From then on program the firmware starting at address 0x1000:

```
esptool.py --chip esp32 --port /dev/ttyUSB0 --baud 460800 write_flash -z 0x1000 esp3
2-20190125-v1.10.bin
```

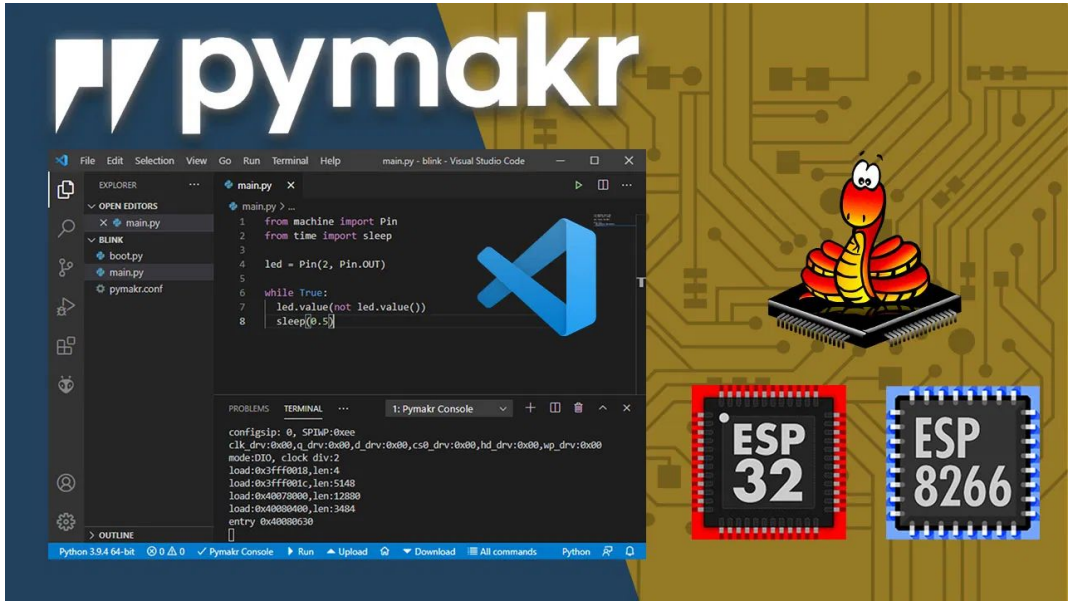
Pycom Lopy

- Xtensa® dual-core 32-bit LX6 microprocessor(s) (Same as **ESP32**)
- WiFi, LoRa, BLE



Pycom Lopy

- Pymakr for update firmware.
- VSCode Plugin



Supported Device for MicroPython

---

https://micropython.org/download/

Firmware for various microcontroller ports and boards are built automatically on a daily basis and can be found below.

Filter by:

Port: cc3200, esp32, esp8266, mimxrt, nrf, renesas-ra, rp2, samd, stm32

Feature: 10/100 Ethernet Phy, 16MB Flash, 1MB SPI Flash, 2x80 pin HD connectors, 8MB SDRAM, ADC, AudioAmp, AudioCodec, BLE, Battery Charging, Battery Management, Bluetooth 5.0, Bluetooth Nina-W102, Breadboard Friendly, Breadboard friendly, CAN, CYW43 WiFi/BT Module, Camera, Castellated Pads, Crypto IC ARM CC310, Crypto IC ATECC608A-MAHDA-T, Display, DisplayPort over USB-C, Dual-core processor, Ethernet, Feather, Grove, Humidity sensor HTS221, I2C, IMU, IMU LSM6DS3TR, IMU LSM6DSOXTR, IMU LSM9DS1, Infrared, JLink, LoRa, Micro USB, MicroSD, MicroUSB, Microphone, Microphone MP34DT05, Microphone MPM3610, Microphone MSM261D3526H1CPM, NXP SE050 crypto device, OLED, OpenSDA, PoE, Pressure sensor LPS22H, Proximity, Light, RGB sensor APDS-9960, QSPI Flash, QWIIC, RGB LED, Red/green/orange/blue leds, Reset/User button, SDCard, SDRAM, SIM Socket, SPDIF, SPI, SPI Flash, SPI Flash 16MB, SPIRAM, STEMMA QT/QWIIC, UART, USB Full speed, USB High Speed Phy, USB Stick form factor, USB-A, USB-C, USB-MICRO, WiFi, WiFi Nina-W102, mikroBUS

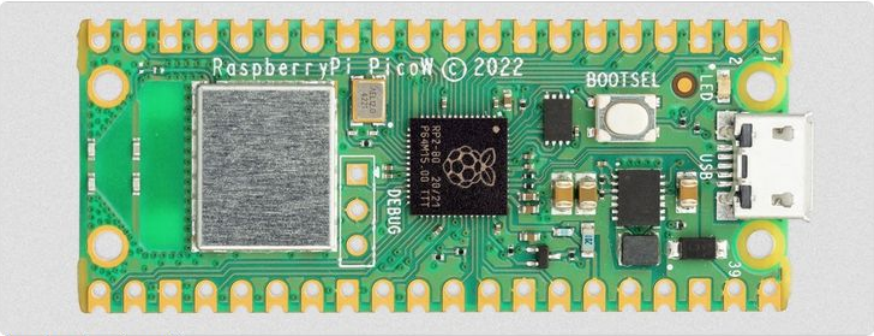
Vendor: Actinius, Adafruit, Arduino, BBC, Espressif, Espruino, George Robotics, I-SYST, LEGO, LILYGO, Laird Connectivity, Lego, M5 Stack, McHobby, Microchip, MikroElektronika, MiniFig Boards, NXP, Nordic Semiconductor, OLIMEX, PJRC, Particle, Pimoroni, Pycom, Raspberry Pi, Renesas Electronics, ST Microelectronics, Seeed Studio, Seeed Technology Co.,Ltd., Silicognition LLC, Sparkfun, Unexpected Maker, Unknown, VCC-GND Studio, WeAct, Wemos, Wireless-Tag, Wiznet, nullbits, u-blox

MCU: RA4M1, RA4W1, RA6M1, RA6M2, cc3200, esp32, esp32c3, esp32s2, esp32s3, esp8266, mimxrt, nrf51, nrf52, nrf91, rp2040, samd21, samd51, stm32f0, stm32f4, stm32f7, stm32g0, stm32g4, stm32h7, stm32l0, stm32l1, stm32l4, stm32wb, stm32wl

Upload MicroPython into Pico

- Download firmware for RP2040 (.uf2 file)
  https://micropython.org/download/rp2-pico-w/

Pico W



Vendor: Raspberry Pi
Features: Breadboard friendly, Castellated Pads, Micro USB, WiFi
Source on GitHub: rp2/PICO_W
More info: Website

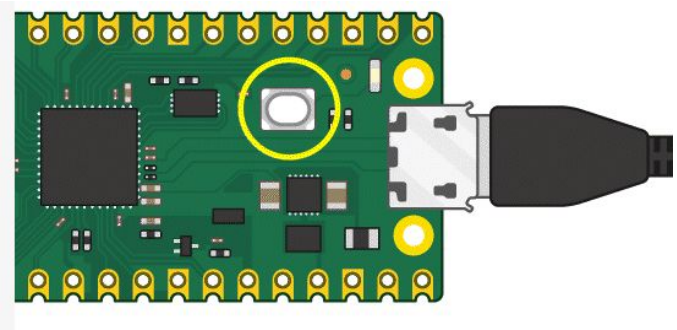
Firmware

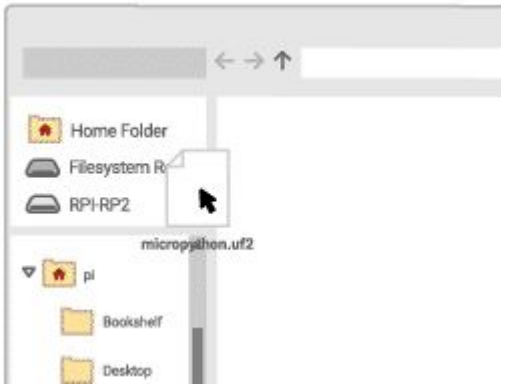
Nightly builds

v1.19.1-995-g0a3600a9a (2023-03-31) .uf2
v1.19.1-994-ga4672149b (2023-03-29) .uf2
v1.19.1-993-g283c1ba07 (2023-03-29) .uf2
v1.19.1-992-g38e7b842c (2023-03-23) .uf2

Upload MicroPython into Pico

- Plug USB as BOOTSEL mode
- Drag-and-Drop a downloaded .uf2 file to USB device





Test MicroPython with REPL

- Use screen (For MacOS), minicom, picocom or putty for the serial interface.

```
$ minicom -o -D /dev/ttyACM0
```

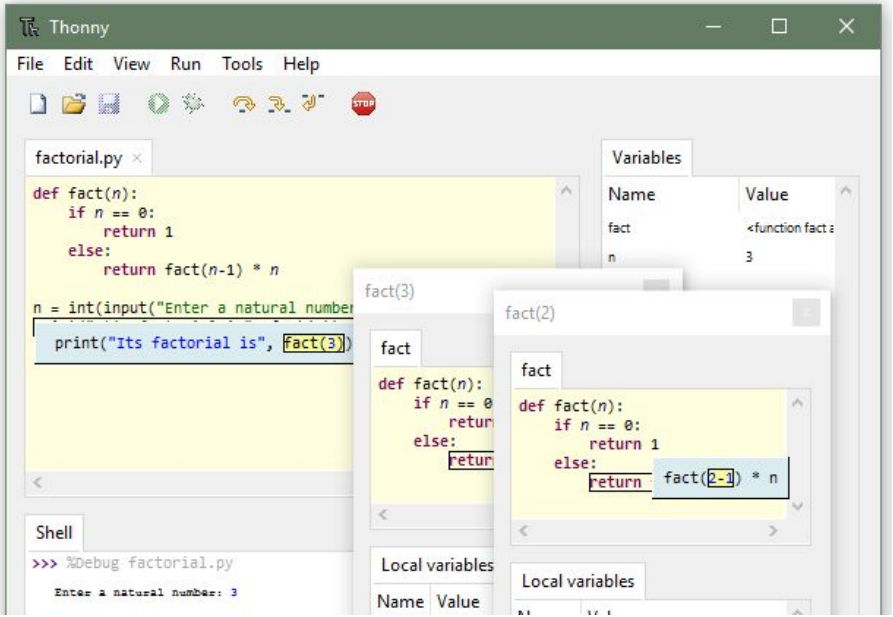
- If you are using a Mac computer, then the name of the Pico serial device is `/dev/tty.usbmodem0000000000001` .
- On a Linux computer the device name may vary, but it usually has the format `/dev/ttyACM<n>` ,

Test MicroPython with REPL

```
Welcome to minicom 2.8

OPTIONS: I18n
Port /dev/ttyACM0, 11:27:38

Press CTRL-A Z for help on special keys


>>>
>>>
>>> import utime
>>> from machine import Pin
>>> led_pin = 25
>>> LED = Pin(led_pin, Pin.OUT)
>>> LED.value(1)
>>> LED.value(0)
>>> for i in range(4):
...     LED.value(1)
...     utime.sleep(1)
...     LED.value(0)
...     utime.sleep(1)
...
>>>
```

Run Micropython Program as Files

- **Thonny** is the recommended way of working with a Pico and MicroPython as it has built-in support for running and uploading files. With GUI interface.



Run Micropython Program as Files

- **Ampy** is CLI tools version.

  To install Ampy
  $ pip3 install adafruit-ampy

  List file in Pi Pico
  $ ampy --port /dev/ttyXXXX ls

  Put file from host to MCU
  $ ampy --port /dev/ttyXXXX put main.py

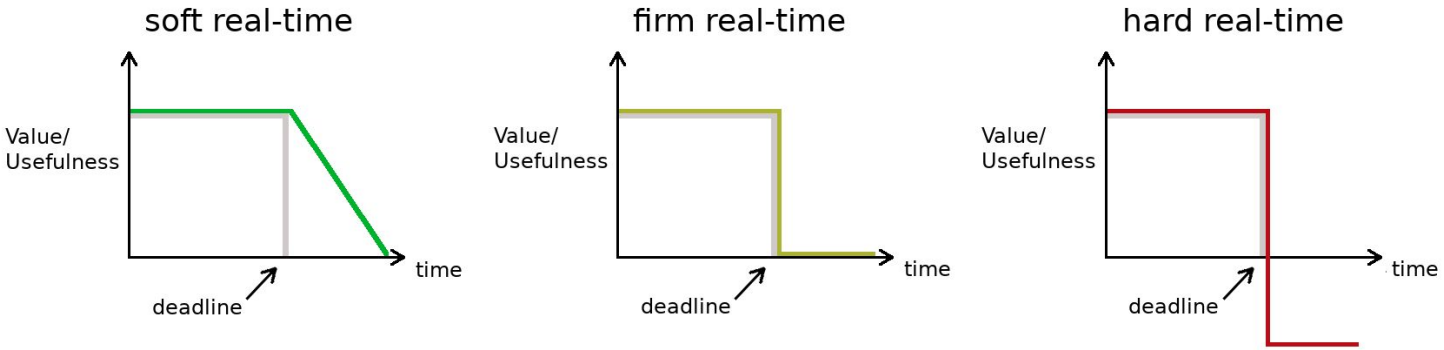
  Run the file
  ampy --port /dev/ttyXXXX run main.py

  Remove file
  $ ampy --port /dev/ttyXXXX rm main.py

  Get file from MCU to host
  ampy --port /dev/ttyXXXX get main.py

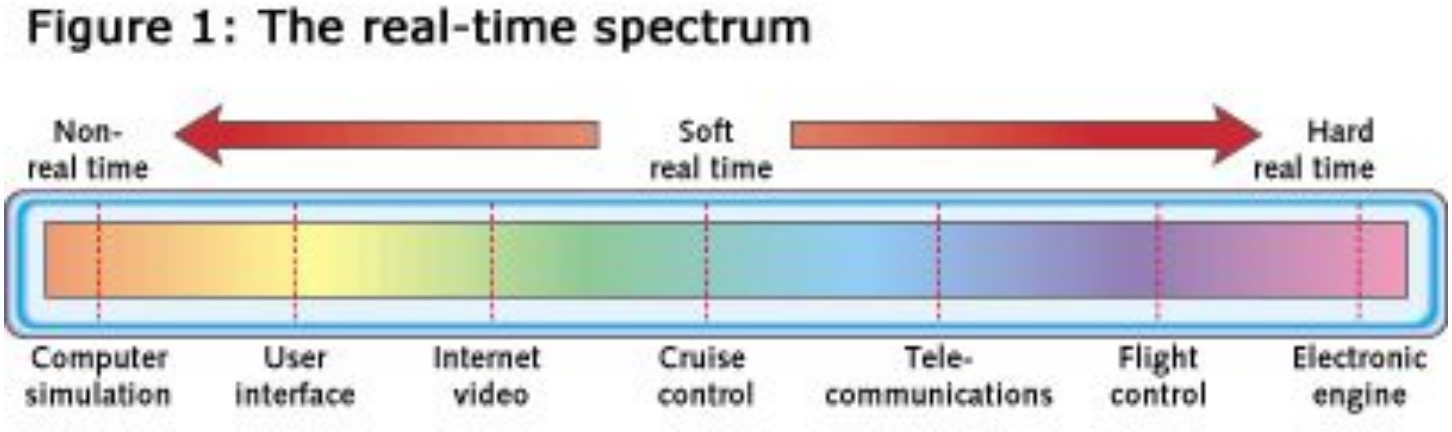
When we shouldn’t use micropython?

- Needs accurate timing (**Real-time**), Good performance and long life.
- Safety consideration, Life and death situation etc.



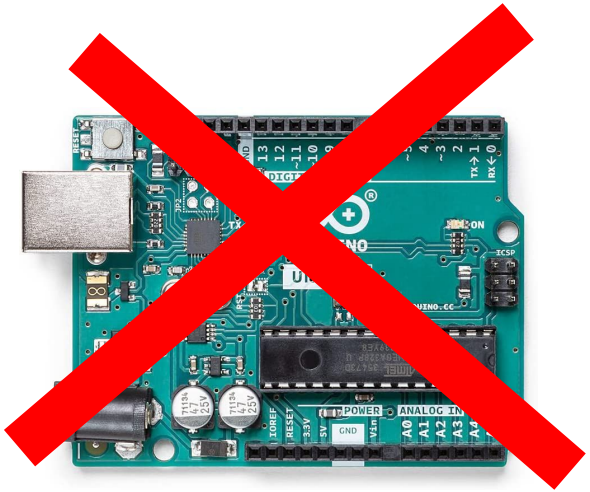
Real-time -> not always fast. but predictable and deterministic

When we shouldn't use micropython?

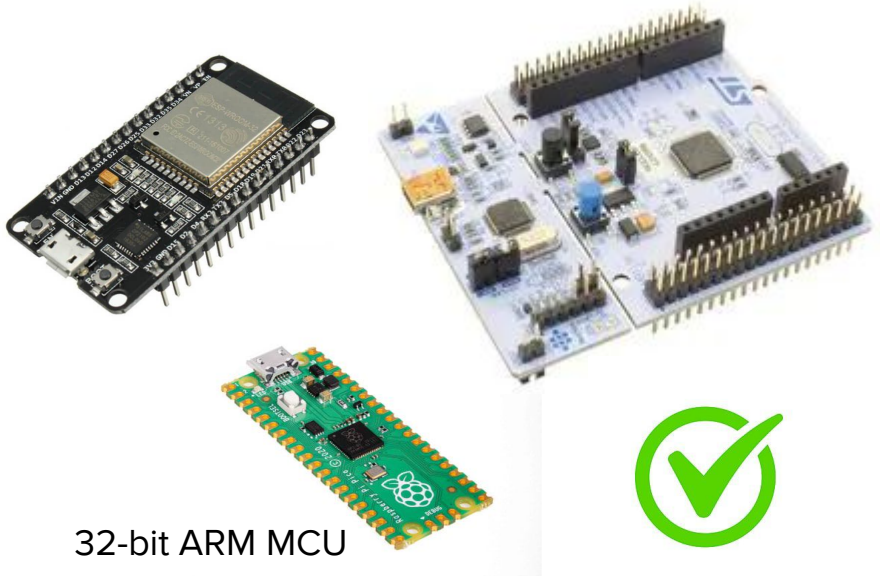


When we shouldn’t use micropython?

- Limit of the hardware resource. Memory and CPU speed.



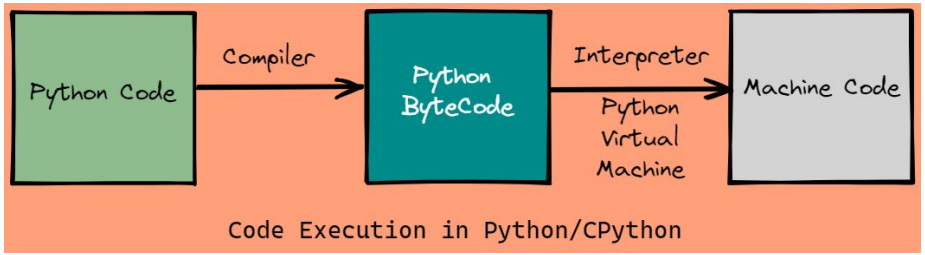
8/16-bit MCU

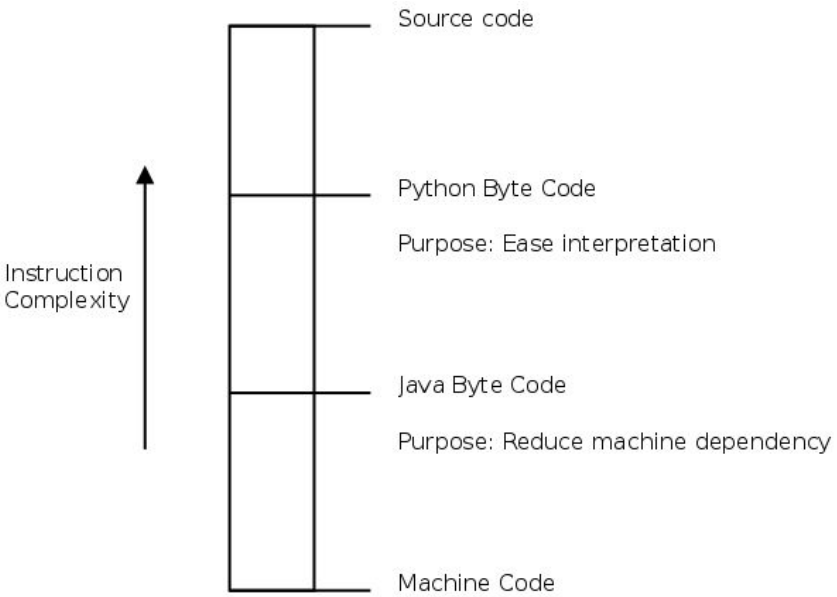


32-bit ARM MCU

When we shouldn’t use micropython?

- Secure Issue : Production code can be hacked! With Bytecode Tracing.

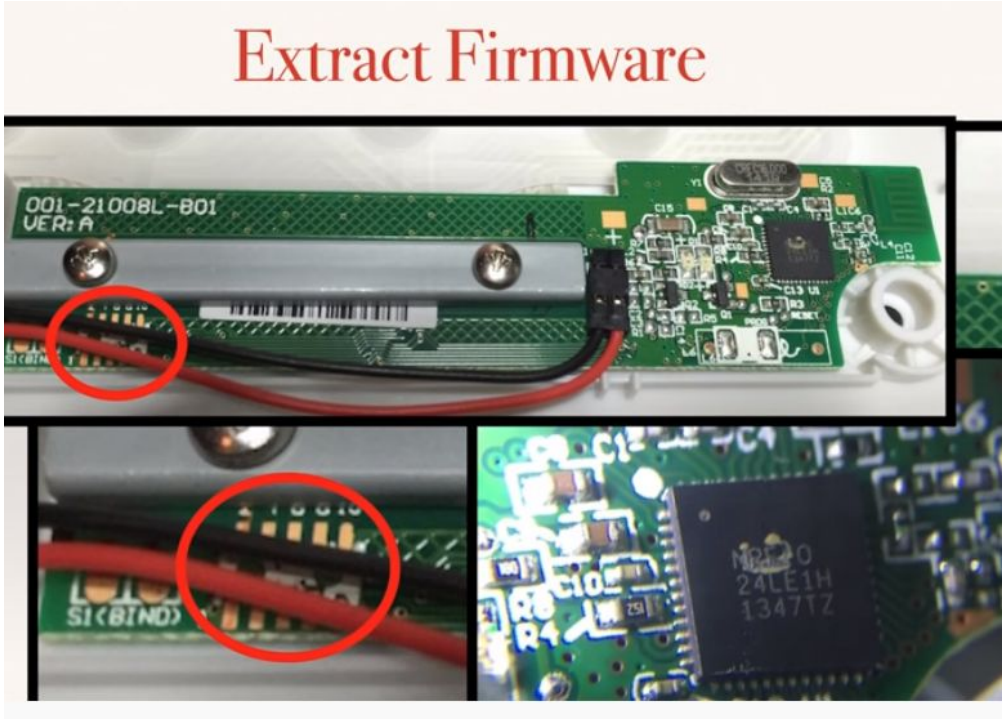




When we shouldn’t use micropython?

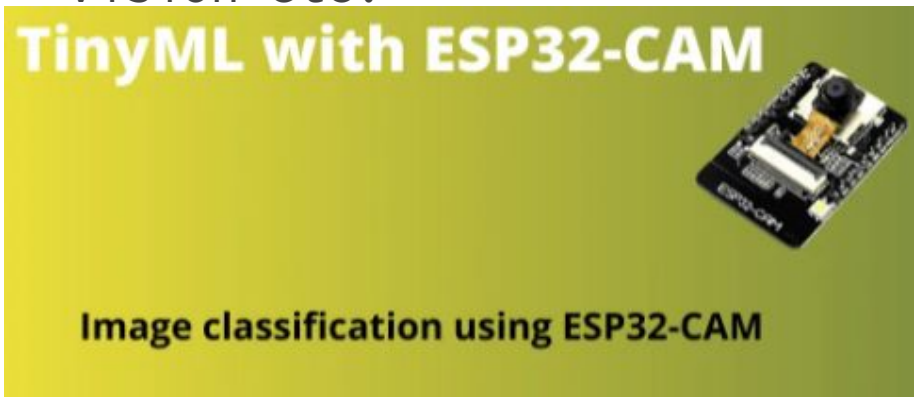
Some random consumer product can be hacked via programmer port (UART, SPI, USB etc)

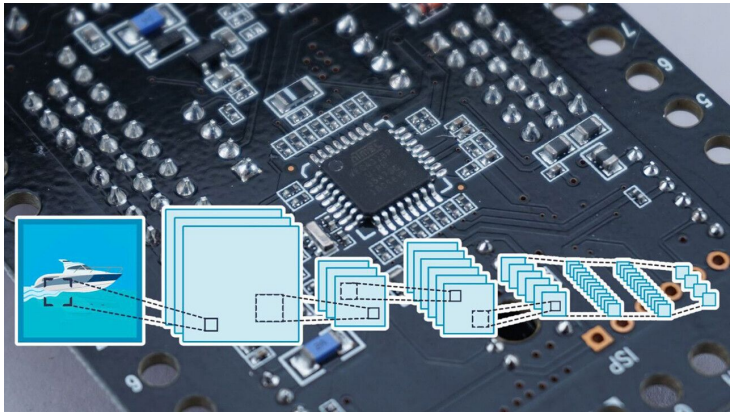
Bytecode make it simpler to extract the firmware.



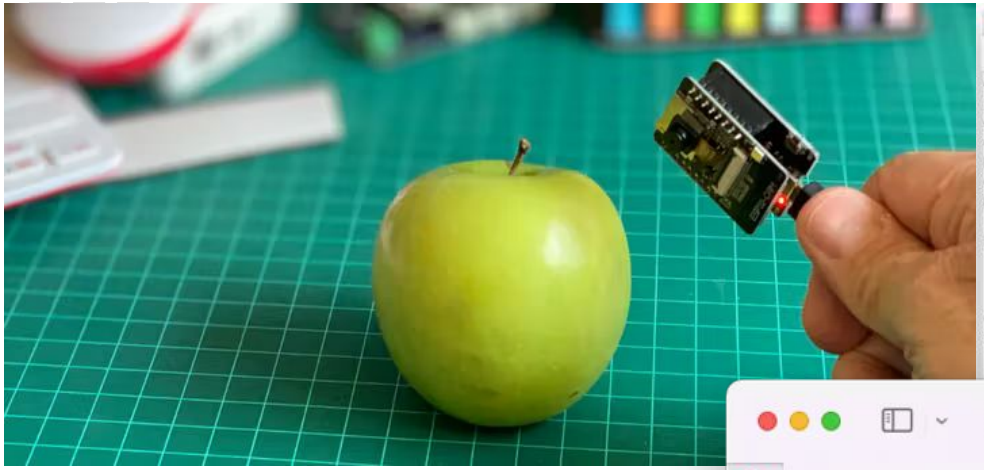
Future on Micropython?

- More hardwares that support Micropython
- Better performance
- IoT with AI on the Edge, ML on microcontroller, Computer vision etc.





Future on Micropython?



Reference Books