Why we use microPython

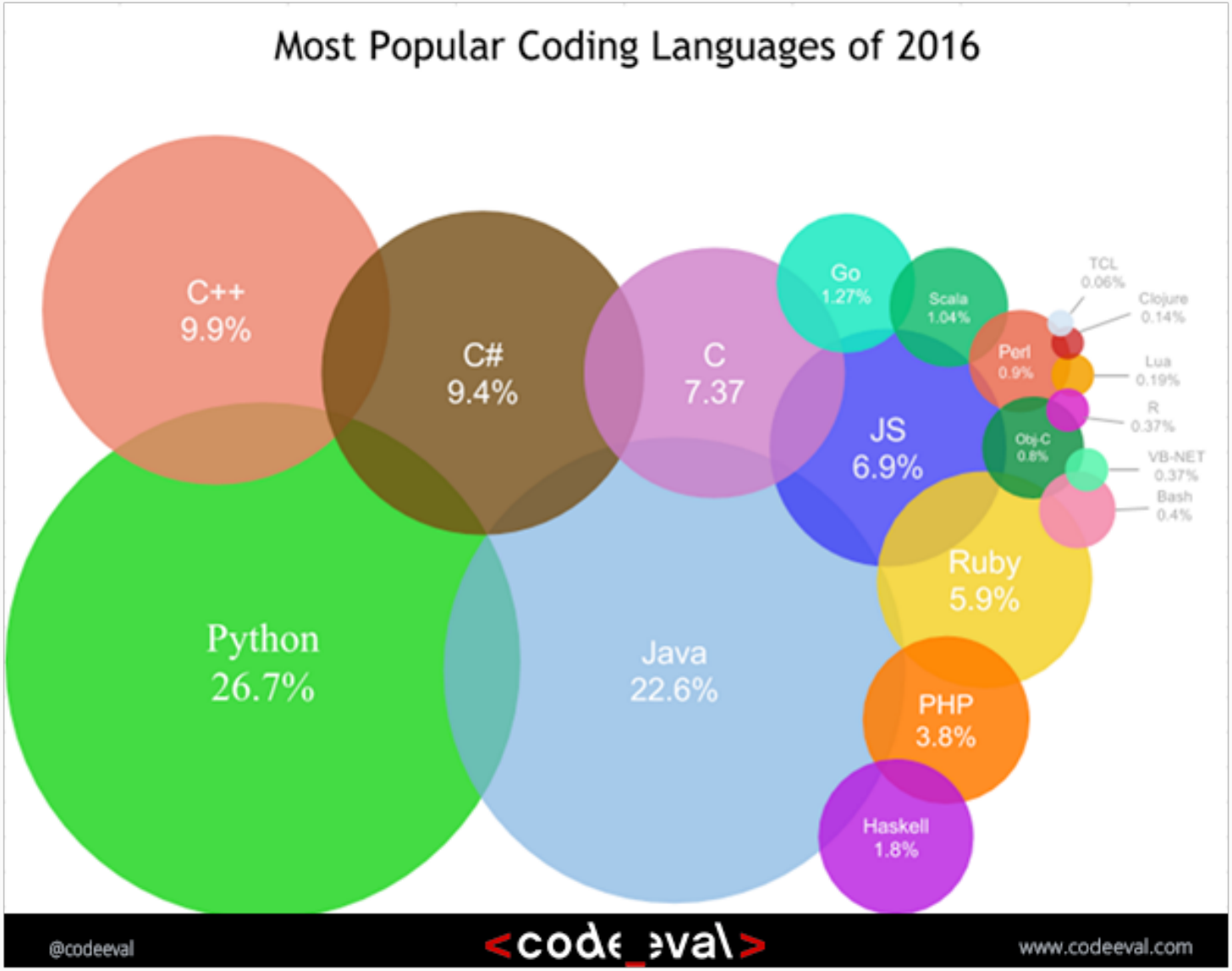
Rapid prototyping with microPython devices

Marco Zennaro, ICTP

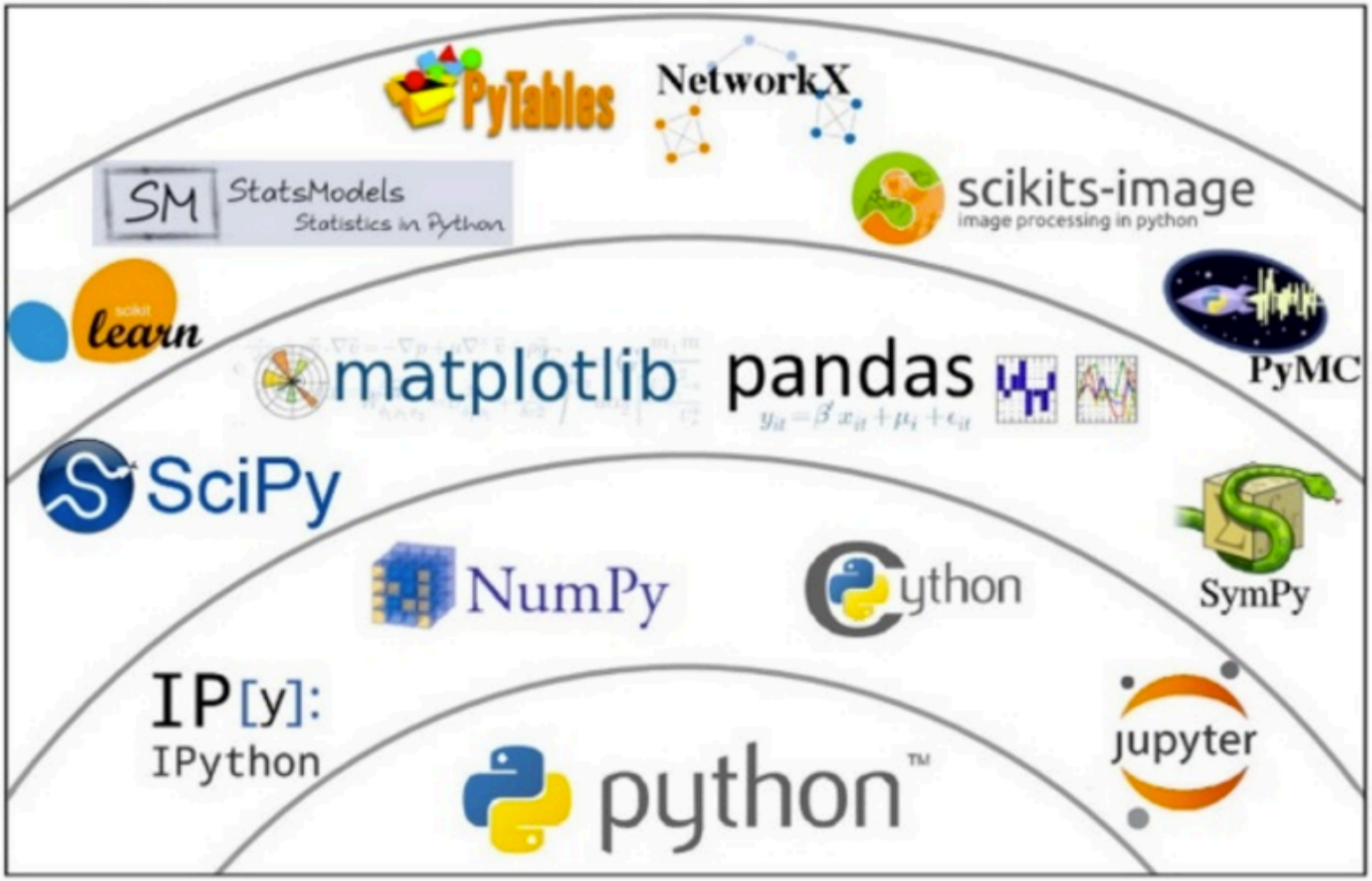
February 18, 2018

Why micropython?

python


Most Popular Coding Languages of 2016
C++
9.9%
C#
9.4%
C
7.37
Go
1.27%
Scala
1.04%
TCL
0.06%
Clojure
0.14%
Perl
0.9%
Lua
0.19%
R
0.37%
Obj-C
0.8%
VB-NET
0.37%
Bash
0.4%
JS
6.9%
Ruby
5.9%
Python
26.7%
Java
22.6%
PHP
3.8%
Haskell
1.8%
@codeeval
<code_eval>
www.codeeval.com

python ecosystem



micropython

MicroPython is a lean and fast implementation of the Python 3 programming language that is optimised to run on a microcontroller. MicroPython was successfully funded via a Kickstarter campaign and the software is now available to the public under the MIT open source license.

It ensures that the memory size/microcontroller performance is optimised and fit for purpose for the application it serves. Many sensor reading and reporting applications do not require a PC based processor as this would make the total application over priced and under-efficient.

micropython options

pyboard

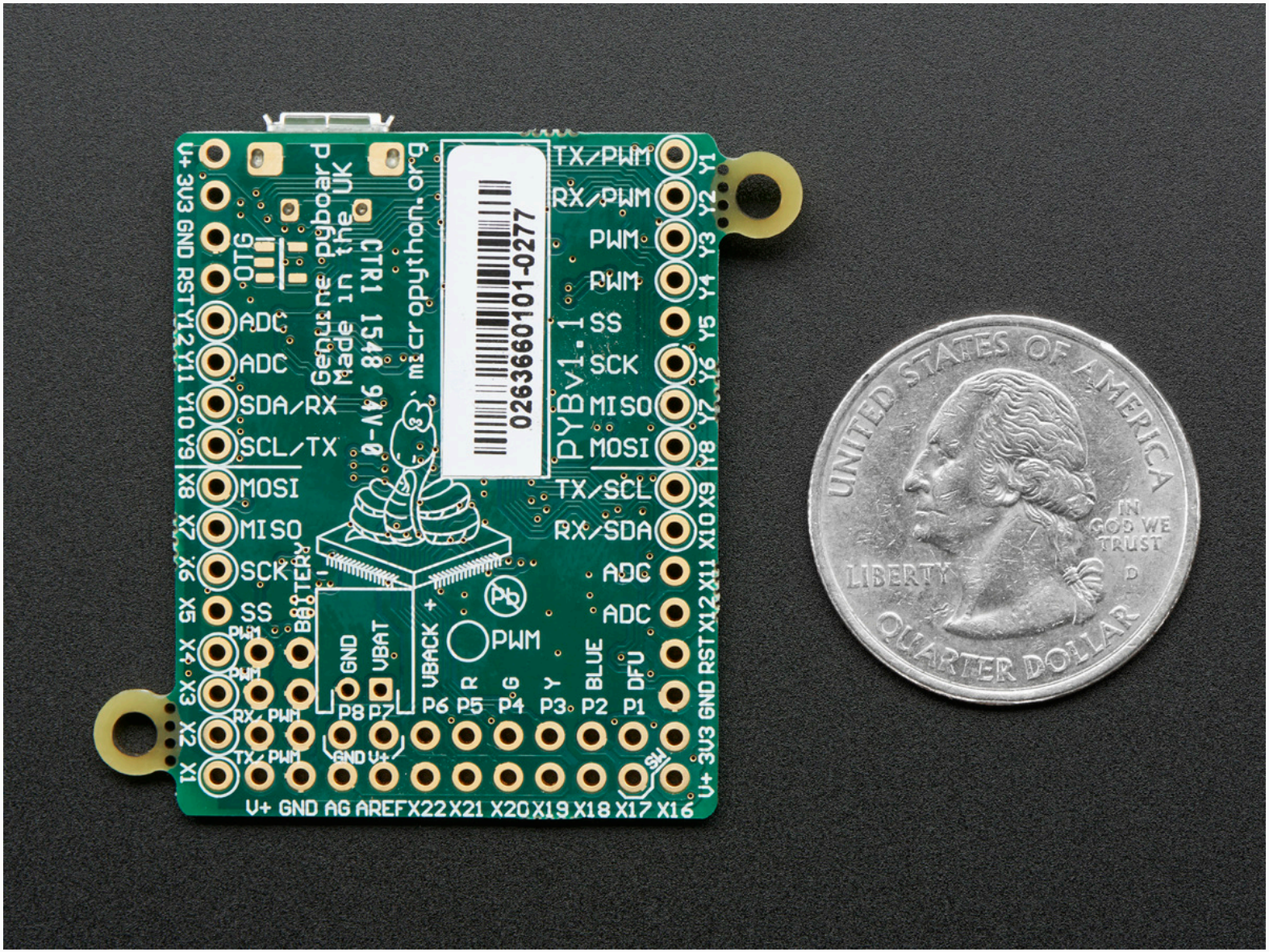
The MicroPython **pyboard** is a compact electronic circuit board that runs MicroPython on the bare metal, giving you a low-level Python operating system that can be used to control all kinds of electronic projects.

MicroPython is packed full of advanced features such as an interactive prompt, arbitrary precision integers, closures, list comprehension, generators, exception handling and more. Yet it is compact enough to fit and run within just 256k of code space and 16k of RAM.

MicroPython aims to be as compatible with normal Python as possible to allow you to transfer code with ease from the desktop to a microcontroller or embedded system.

Credit micropython.org

pyboard



TX/PWM
RX/PWM
PWM
PWM
SS
SCK
MISO
MOSI
TX/SCL
RX/SDA
ADC
ADC
ADC
ADC
SDA/RX
SCL/TX
MOSI
MISO
SCK
SS
PWM
PWM
RX/PWM
TX/PWM
V+ GND AG AREF X22 X21 X20 X19 X18 X17 X16
V+ 3V3 GND RST Y12 Y11 Y10 Y9 X8 X7 X6 X5 X4 X3 X2 X1
Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8 X9 X10 X11 X12 RST GND 3V3 V+
OTG
Genuine pyboard
Made in the UK
CTR1 1548 94V-0
micropython.org
0263660101-0277
PYBv1.1
BATTERY
GND
VBAT
VBACK +
PWM
R
G
Y
BLUE
DFU
P8 P7 P6 P5 P4 P3 P2 P1
UNITED STATES OF AMERICA
IN GOD WE TRUST
LIBERTY
QUARTER DOLLAR

MicroPython pyboard feature table

BOARD			
description	The original pyboard v1.1	Pyboard lite v1.0 with accelerometer	Pyboard lite v1.0
SKU	PYBv1.1	PYBLITEv1.0-AC	PYBLITEv1.0
PRICE			
GBP incl. tax	£28.00	£22.60	£19.60
approx EUR incl. tax	€39.20	€31.60	€27.40
approx USD excl. tax	$35.00	$28.25	$24.50
MICROCONTROLLER			
MCU	STM32F405RGT6	STM32F411RET6	STM32F411RET6
CPU	Cortex-M4F	Cortex-M4F	Cortex-M4F
internal flash	1024k	512k	512k
RAM	192k	128k	128k
maximum frequency	168MHz	96MHz	96MHz
hardware floating point	single precision	single precision	single precision

BOARD FEATURES			
micro USB connector	yes	yes	yes
micro SD card slot	yes	yes	yes
accelerometer (MMA7660)	yes	yes	no
real time clock	32kHz crystal	internal oscillator; pads to solder 32kHz crystal	internal oscillator; pads to solder 32kHz crystal
switches	USR+RST	USR+RST	USR+RST
leds	R+G+Y+B	R+G+Y+B	R+G+Y+B
hobby servo ports	4	4	4
DFU mode for firmware upgrade	yes	yes	yes
POWER SUPPLY			
supply options	USB/V+/VBAT	USB/V+/VBAT	USB/V+/VBAT
input range on V+/VBAT	3.6v-16v	3.6v-16v	3.6v-16v
max output of regulated 3.3v	250mA	250mA	250mA
place for JST connector	yes	yes	yes
backup battery input (VBACK)	yes	yes	yes

POWER CONSUMPTION			
running at 168MHz	56mA	-	-
running at 96MHz	37mA	23mA	23mA
running at 48MHz	21mA	13mA	13mA
idling at 168MHz	16mA	-	-
idling at 96MHz	12mA	5mA	5mA
idling at 48MHz	7mA	4mA	4mA
sleep (full RAM retention)	360uA	180uA	180uA
deepsleep (backup retention only)	6uA	6uA	6uA
IO CAPABILITIES			
IO pins	30	30	30
pins with PWM	20	18	18
pins with A/D	16 (4 shielded)	16 (4 shielded)	16 (4 shielded)
pins with D/A	2	0	0

pyboard

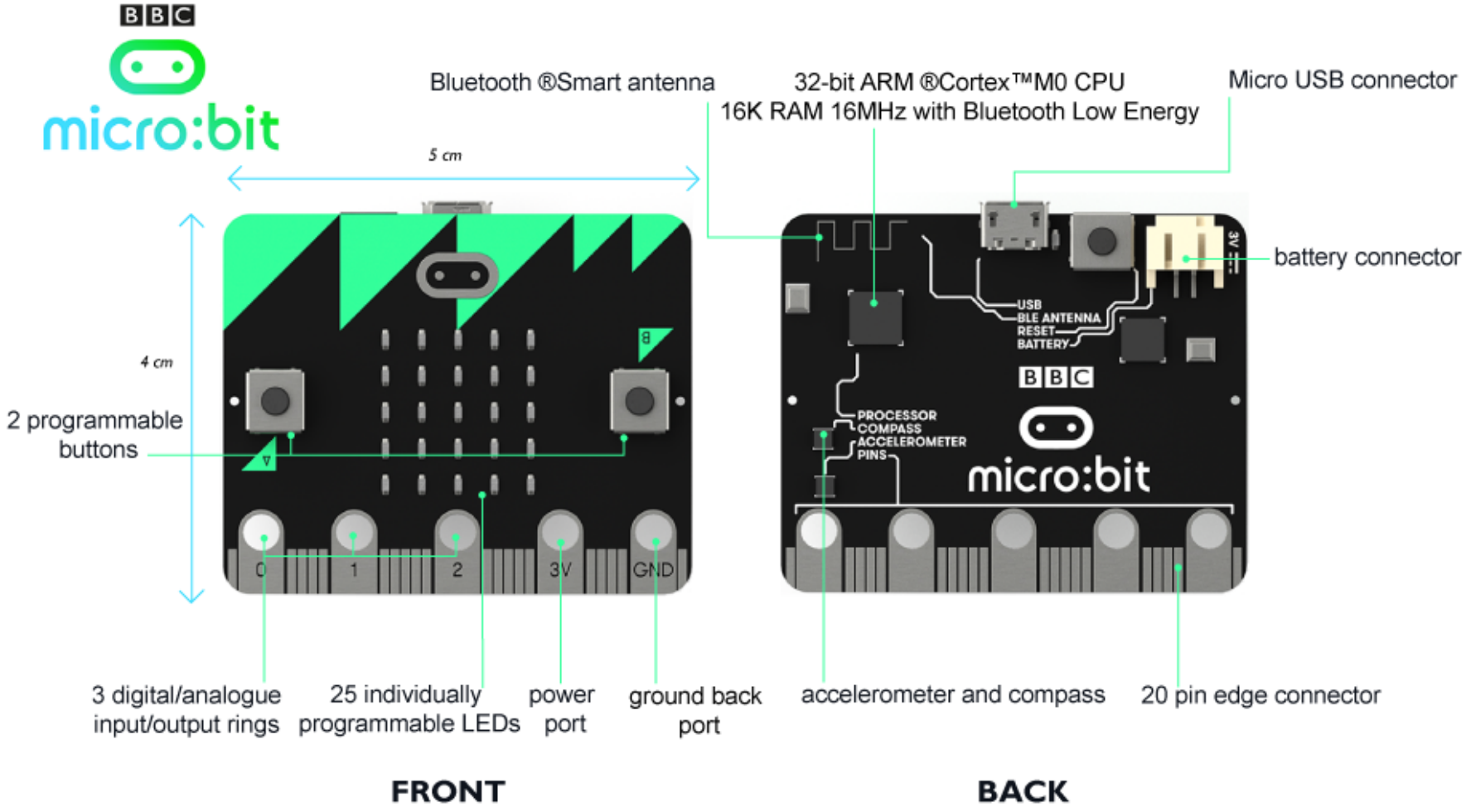
PERIPHERALS			
independent timers	13	7	7
hardware random number generator	yes	no	no
UART	5	3	3
I2C	2	2	2
SPI	2	2	2
CAN	2	0	0
MICROPYTHON CAPABILITIES			
internal flash fs	112k (94k usable)	64k (46k usable)	64k (46k usable)
approx heap size	100k	83k	83k
ADD-ONS			
LCD+touch skin compatible (LCD32MKv1.0)	yes	yes	yes
Audio skin compatible (AMPv1.0)	yes	no	no

ESP8266: low cost

ESP8266: characteristics

- 802.11 b/g/n
- Built-in TCP / IP protocol stack
- Built-in PLL, voltage regulator and power management components
- 802.11b mode + 19.5dBm output power
- Built-in temperature sensor
- off leakage current is less than 10uA
- Built-in low-power 32-bit CPU: can double as an application processor
- SDIO 2.0, SPI, UART
- standby power consumption of less than 1.0mW

BBC Micro:bit



BBC Micro:bit

The Micro Bit is an ARM-based embedded system designed by the BBC for use in computer education in the UK.

The board has an ARM Cortex-M0 processor, accelerometer and magnetometer sensors, Bluetooth and USB connectivity, a display consisting of 25 LEDs, two programmable buttons, and can be powered by either USB or an external battery pack. The device inputs and outputs are through five ring connectors that are part of the 23-pin edge connector.

Digi

Add to compare

Digi XBee3™ Cellular LTE CAT 1

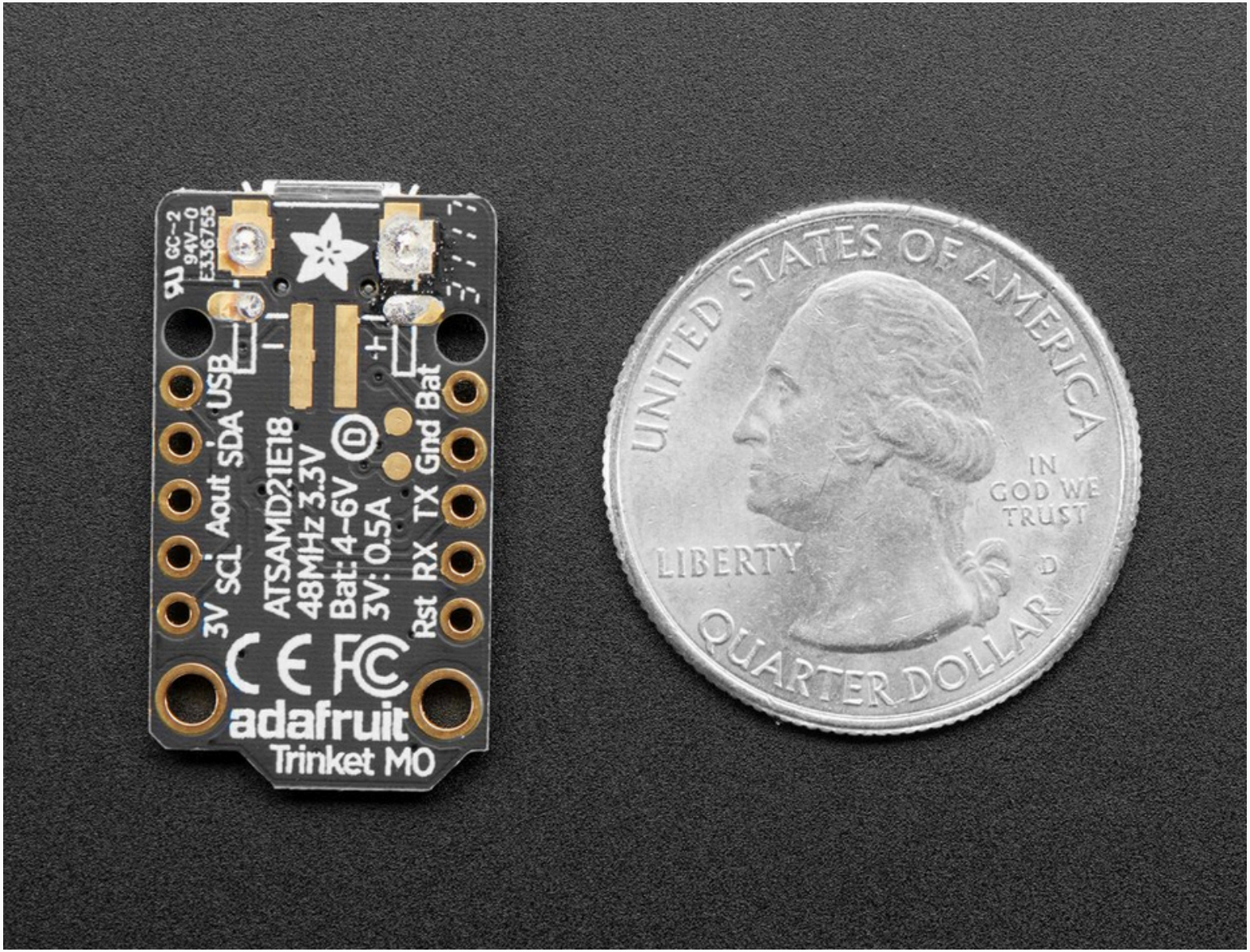
Digi XBee3™ smart modems offer the easiest way to integrate cellular...

Add to compare

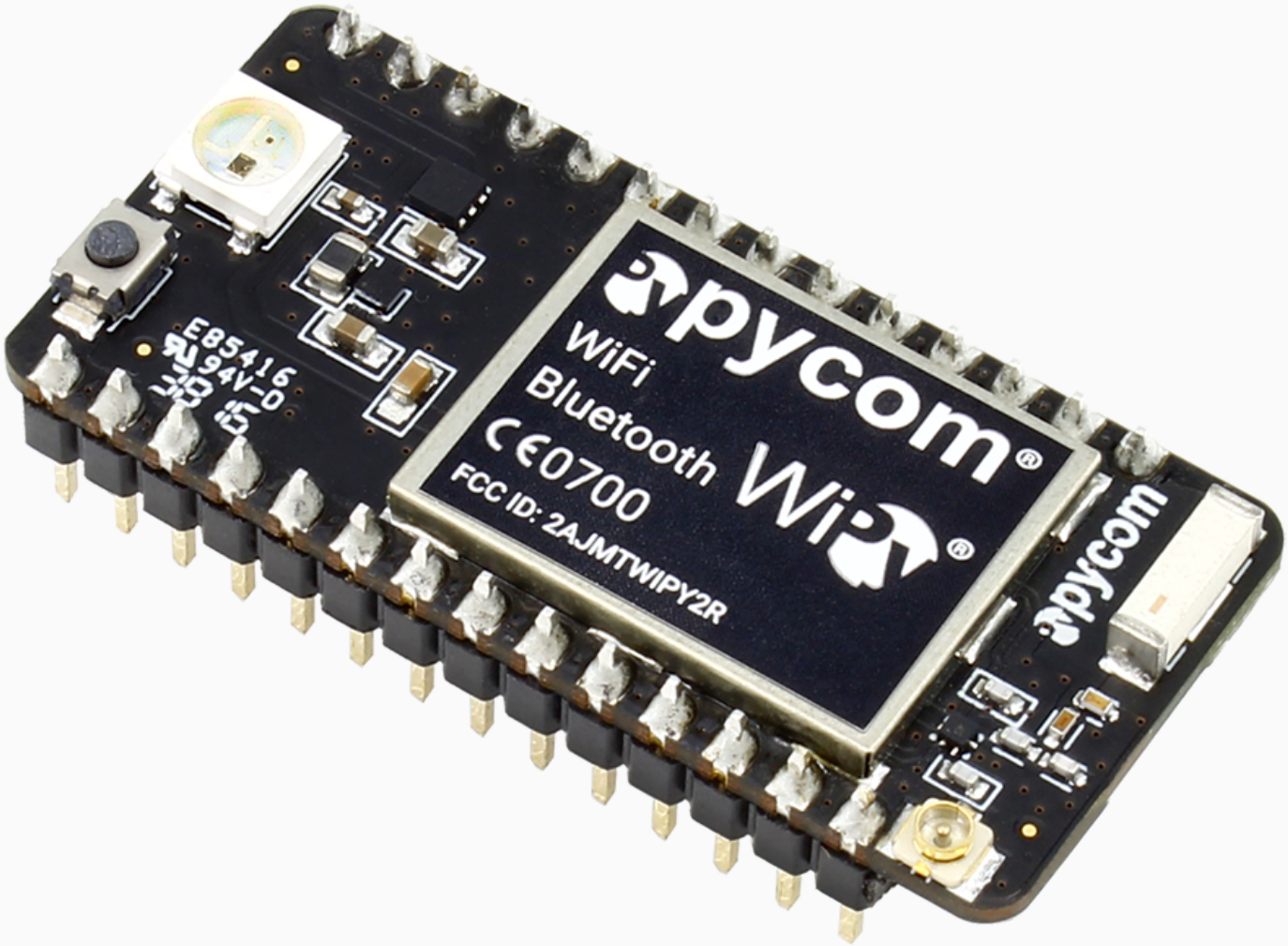
Digi XBee® Cellular LTE Cat 1

Digi XBee Cellular LTE Cat 1 embedded modems provide OEMs with a...

Trinket


3V SCL Aout SDA USB
ATSAMD21E18
48MHz 3.3V
Bat: 4-6V
3V: 0.5A
Rst RX TX Gnd Bat
adafruit
Trinket M0
UNITED STATES OF AMERICA
IN GOD WE TRUST
LIBERTY
QUARTER DOLLAR

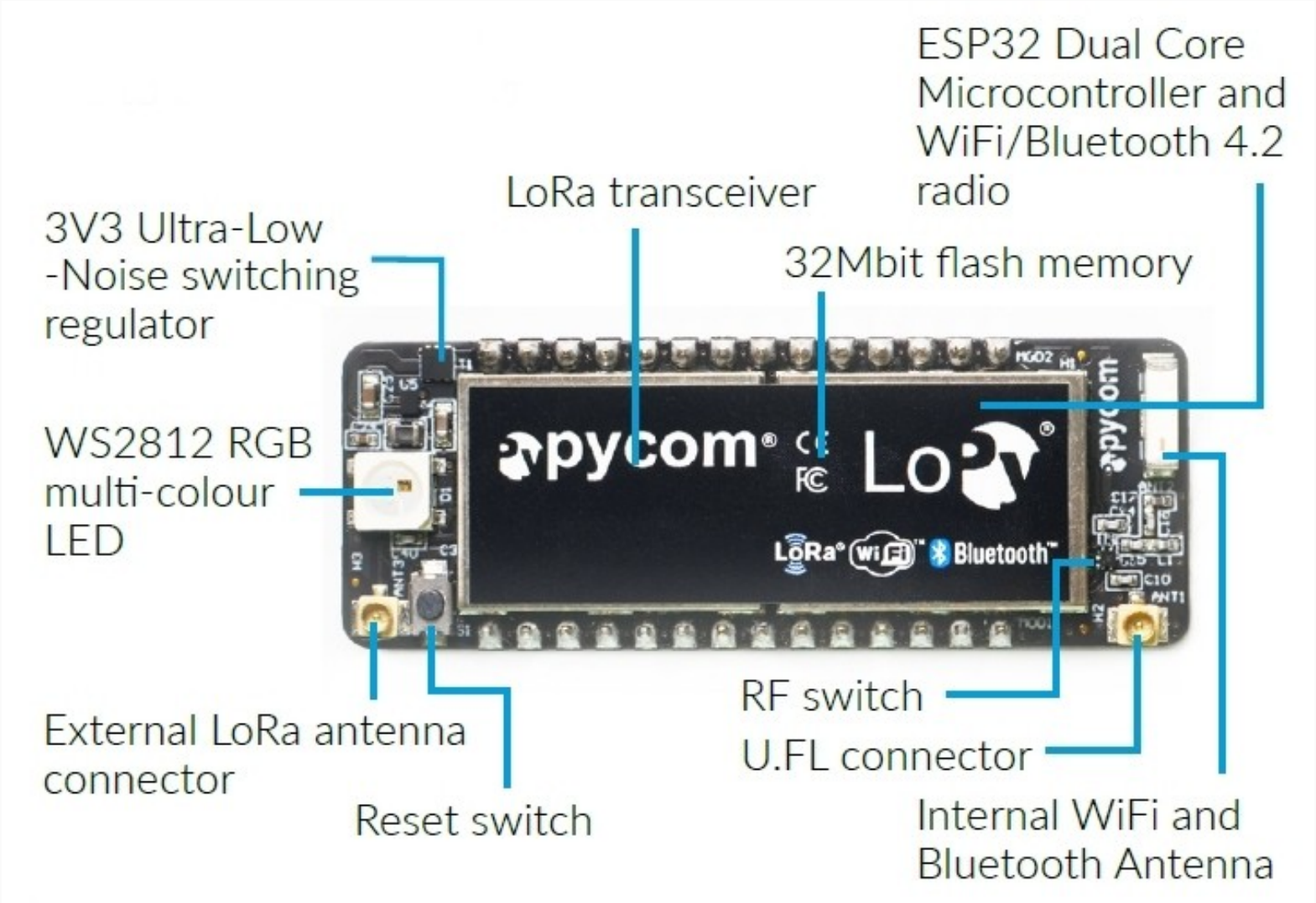
pycom: WiPy


pycom
WiFi
Bluetooth
0700
FCC ID: 2AJMTWIPY2R
E85416
94V-0
pycom

pycom: WiPy

- Espressif ESP32 chipset
- Dual processor + WiFi radio system on chip
- consuming 25uA
- 2 x UART, 2 x sPI, I2C, I2S, micro SD card
- Analog channels: 8×12 bit ADCs
- Hash/Encryption: SHA, MD5, DES, AES
- Bluetooth
- Memory, RAM: 512KB, External flash: 4MB
- Hardware floating point acceleration

pycom: LoPy

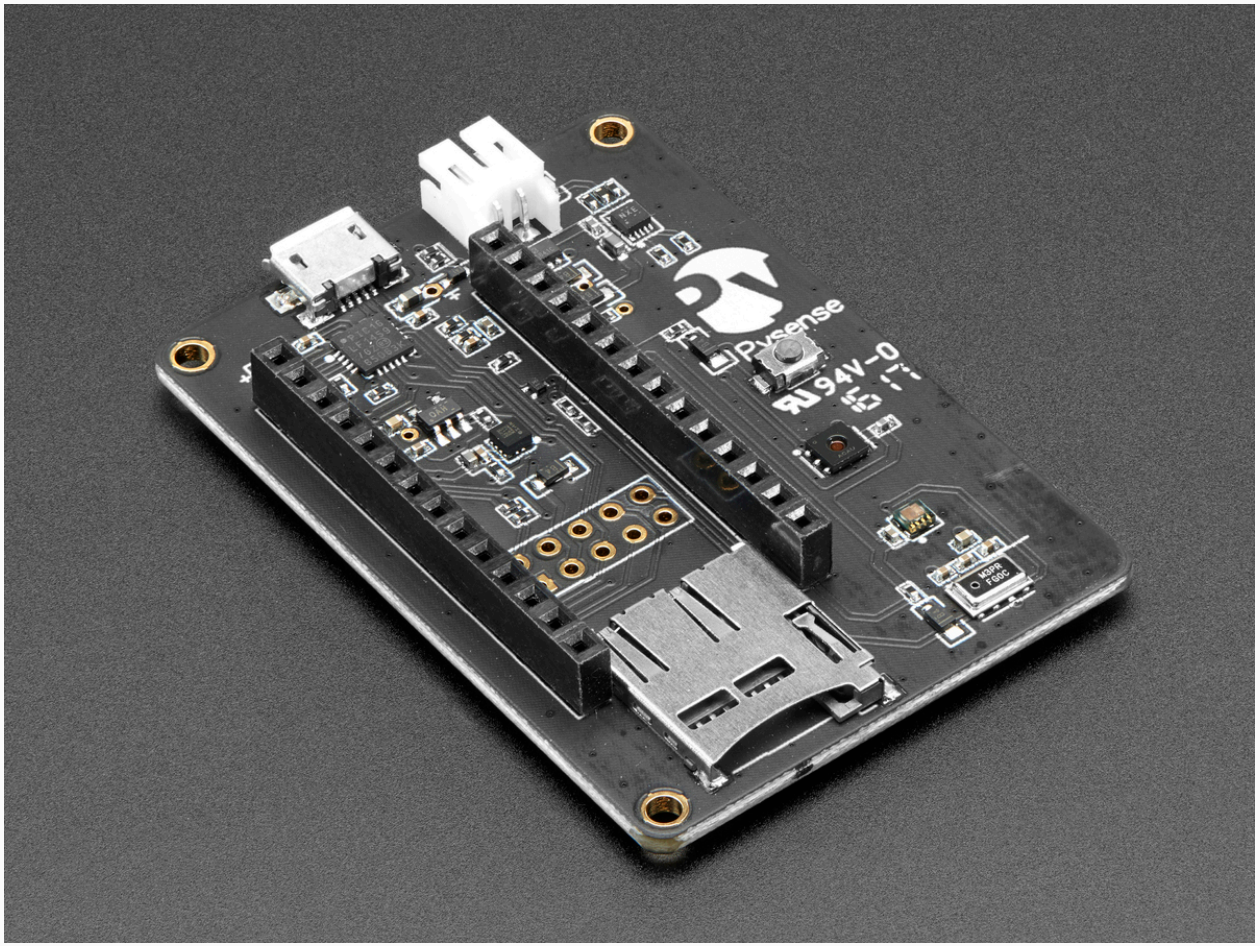

ESP32 Dual Core Microcontroller and WiFi/Bluetooth 4.2 radio
LoRa transceiver
3V3 Ultra-Low -Noise switching regulator
32Mbit flash memory
WS2812 RGB multi-colour LED
RF switch
U.FL connector
External LoRa antenna connector
Reset switch
Internal WiFi and Bluetooth Antenna

pycom: SiPy

pycom®
Bluetooth
WiFi
sigfox
SiPy®
Model: SiPy 1.0
FCC ID: 2AJMTSIPY1R
CE 0700

pycom: Expansion Board

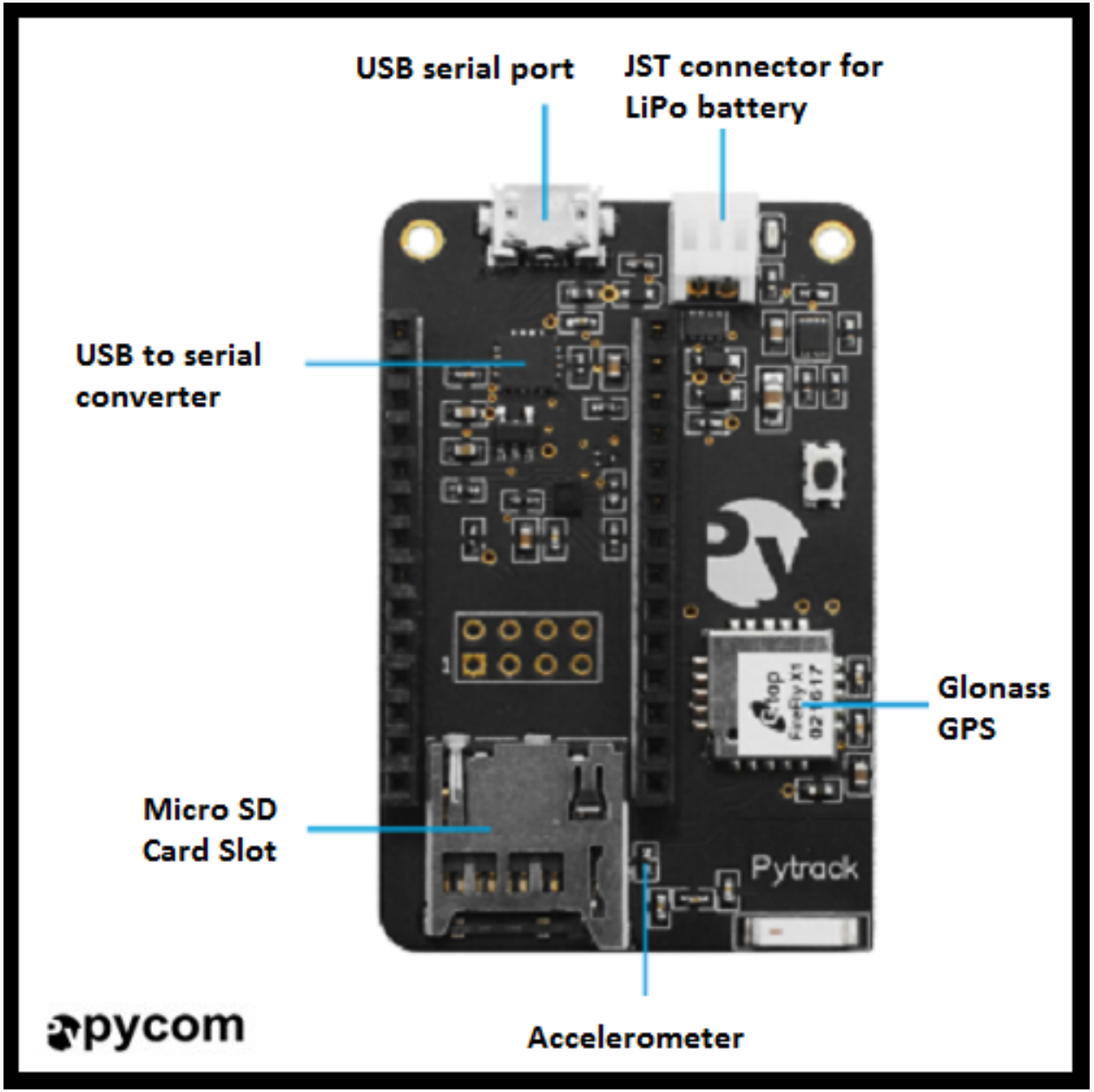
pycom: PySense



pycom: PySense

- Ambient light sensor
- Barometric pressure sensor
- Humidity sensor
- 3 axis 12-bit accelerometer
- Temperature sensor
- USB port with serial access
- LiPo battery charger
- MicroSD card compatibility
- Ultra low power operation (1uA in deep sleep)

pycom: PyTrack



pycom: PyTrack

- GNSS + Glonass GPS
- 3 axis 12-bit accelerometer
- USB port with serial access
- LiPo battery charger
- MicroSD ard compatibility
- Ultra low power operation (1uA in deep sleep)

Plan of the week

plan for the week

During the lab sessions we will cover:

1. Pycom workflow
2. Hello World for IoT: LED switching
3. Saving data to a microSD or to internal flash
4. Reading sensors using the PySense
5. Reading position using the PyTrack
6. Connecting to WiFi and measuring the signal strength
7. Using MQTT
8. Using LoRaWAN
9. Project presentation

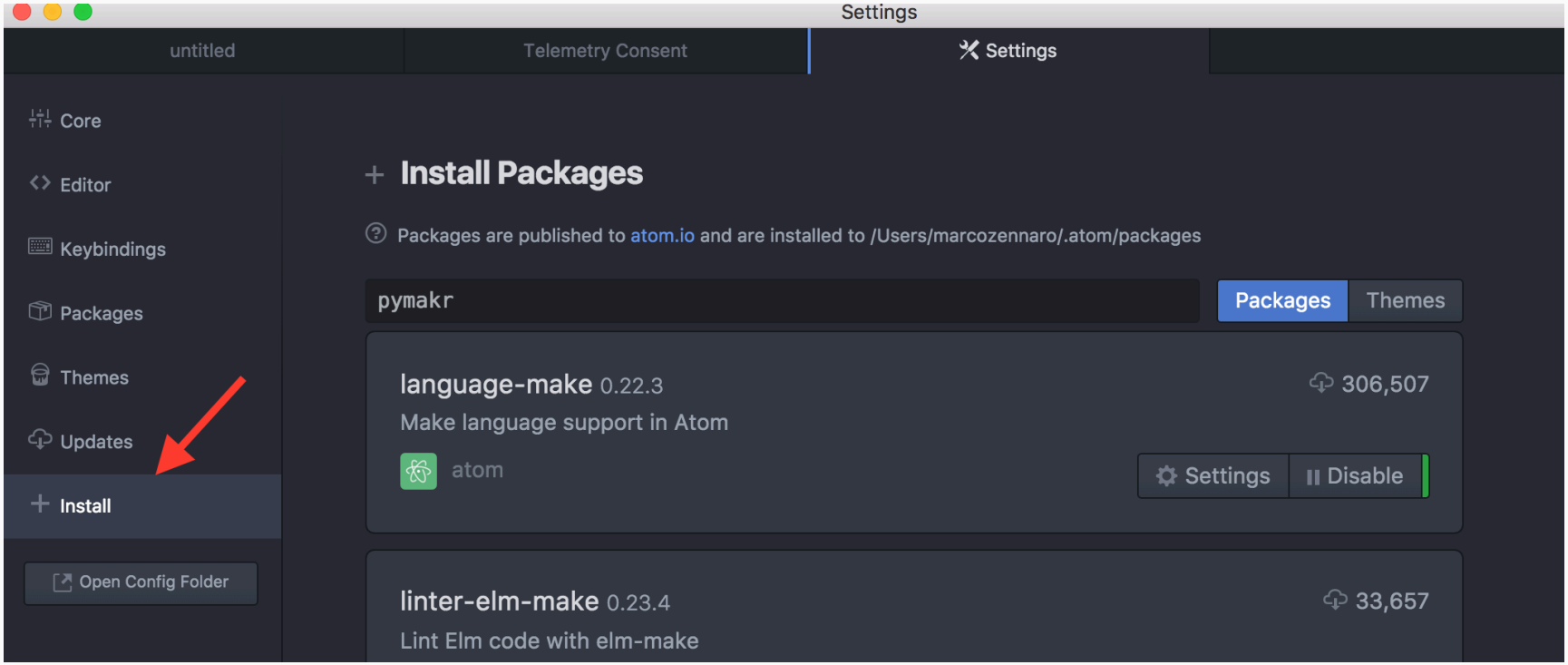
You will have simple code snippets and will develop more complex code as exercise.

workflow: Atom

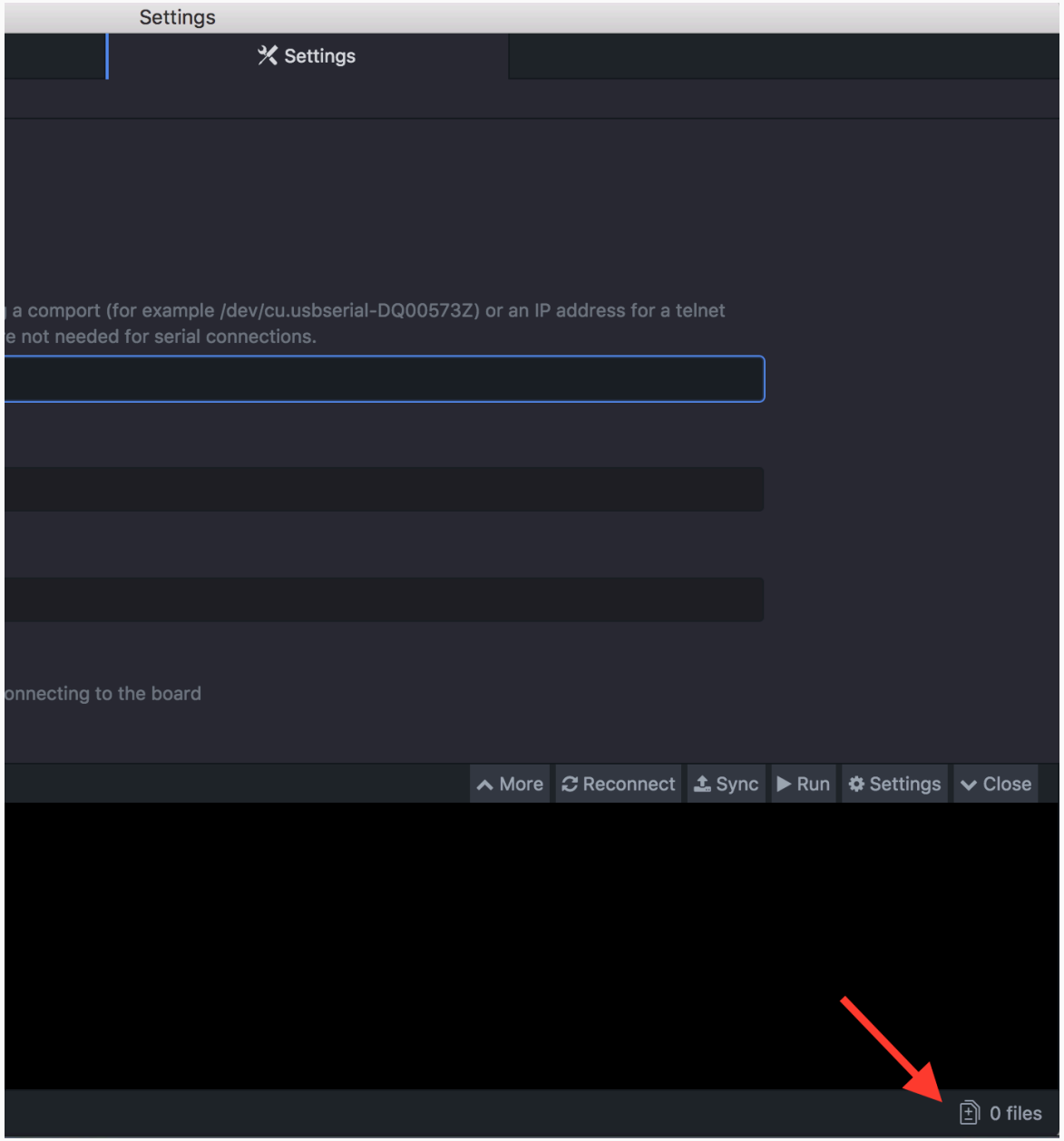
Please install Atom from

`www.atom.io`

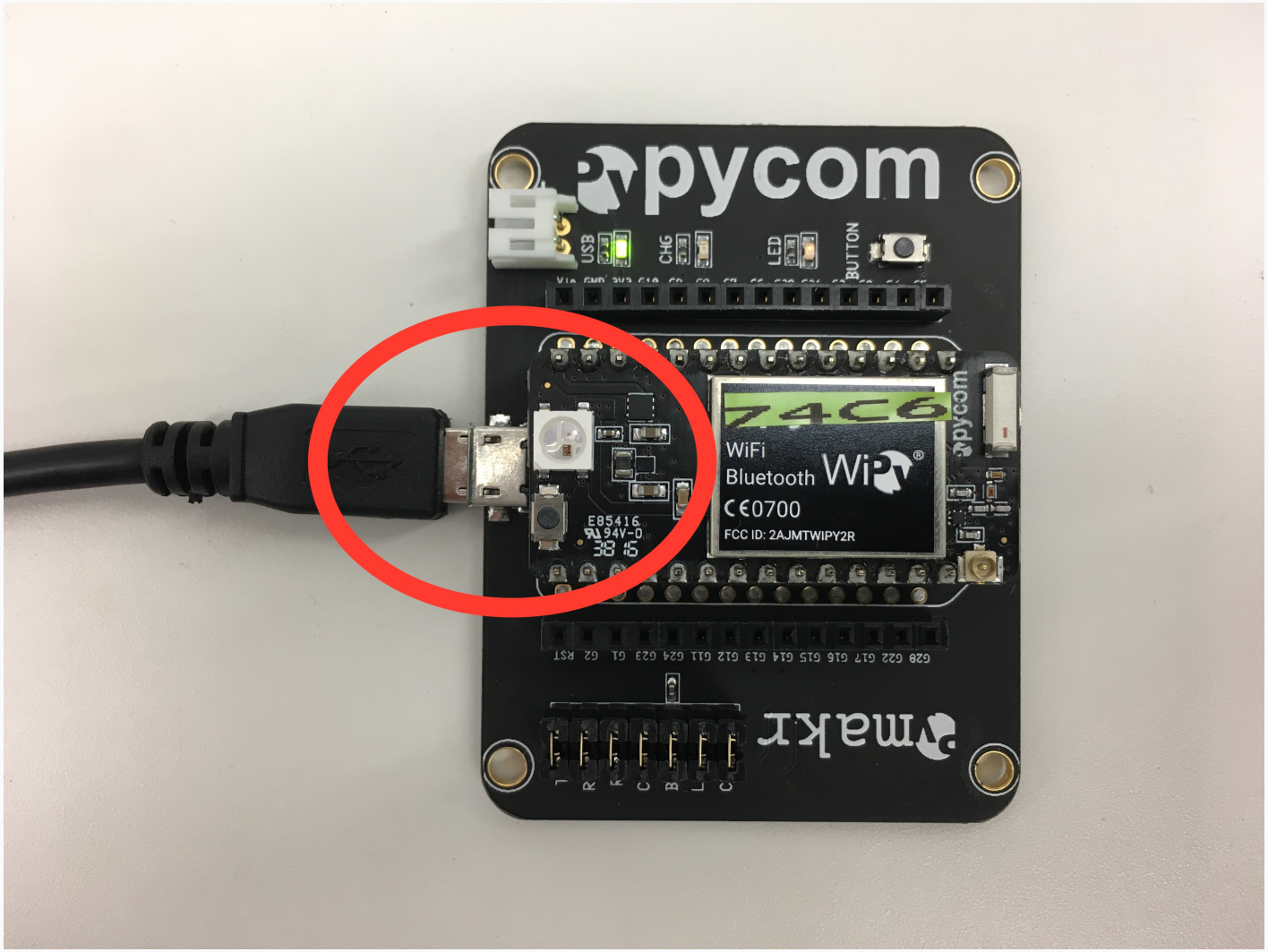
workflow: install the pymakr package



workflow: update packages if necessary

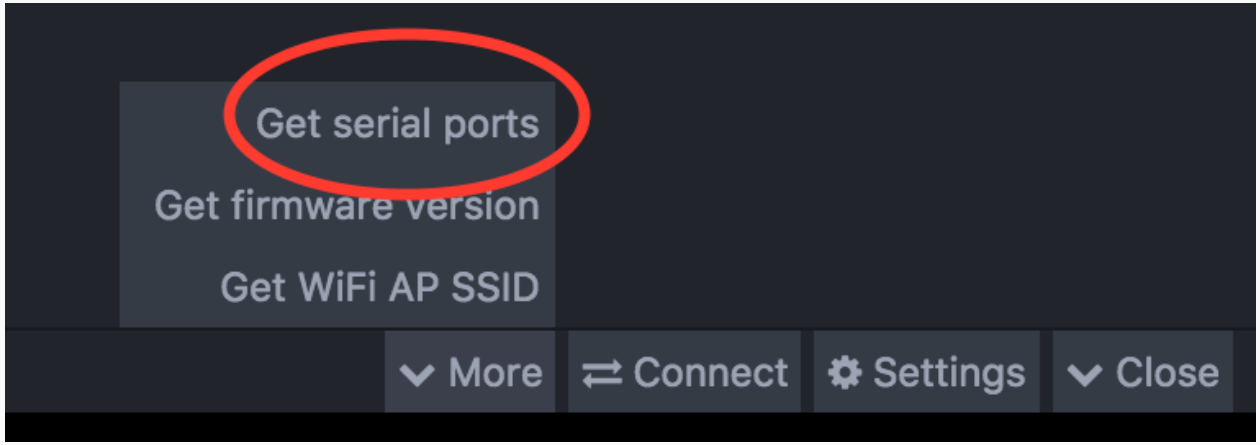


workflow: connect board via USB

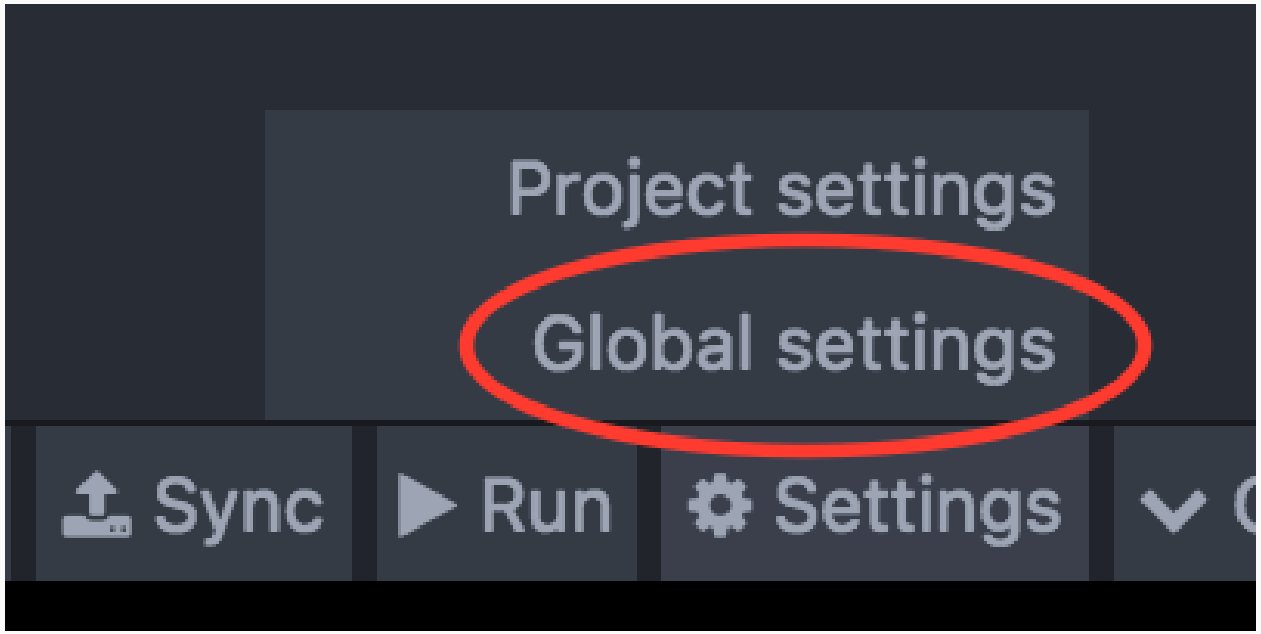


Make sure the LED and the microUSB are on the same side!

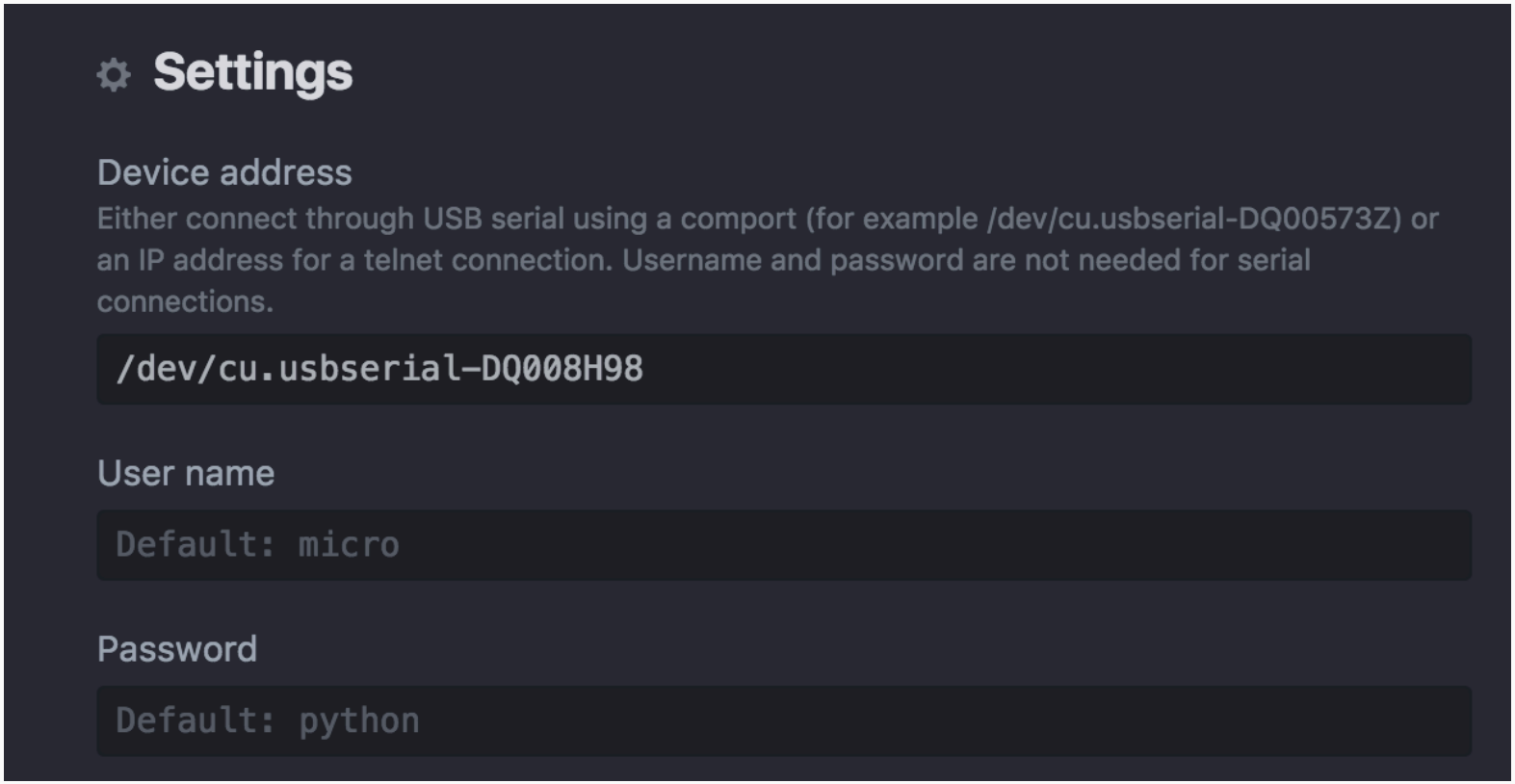
workflow: get serial port



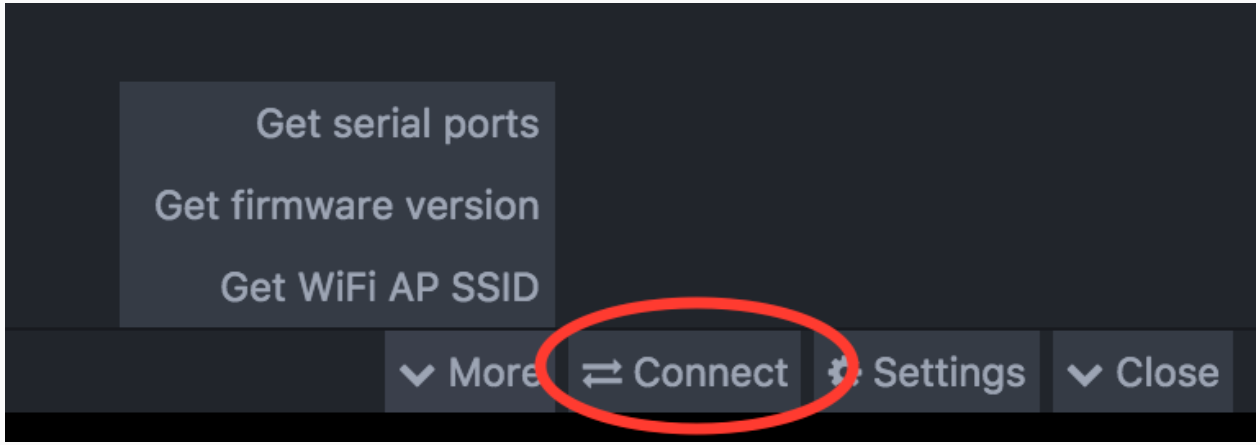
workflow: global settings



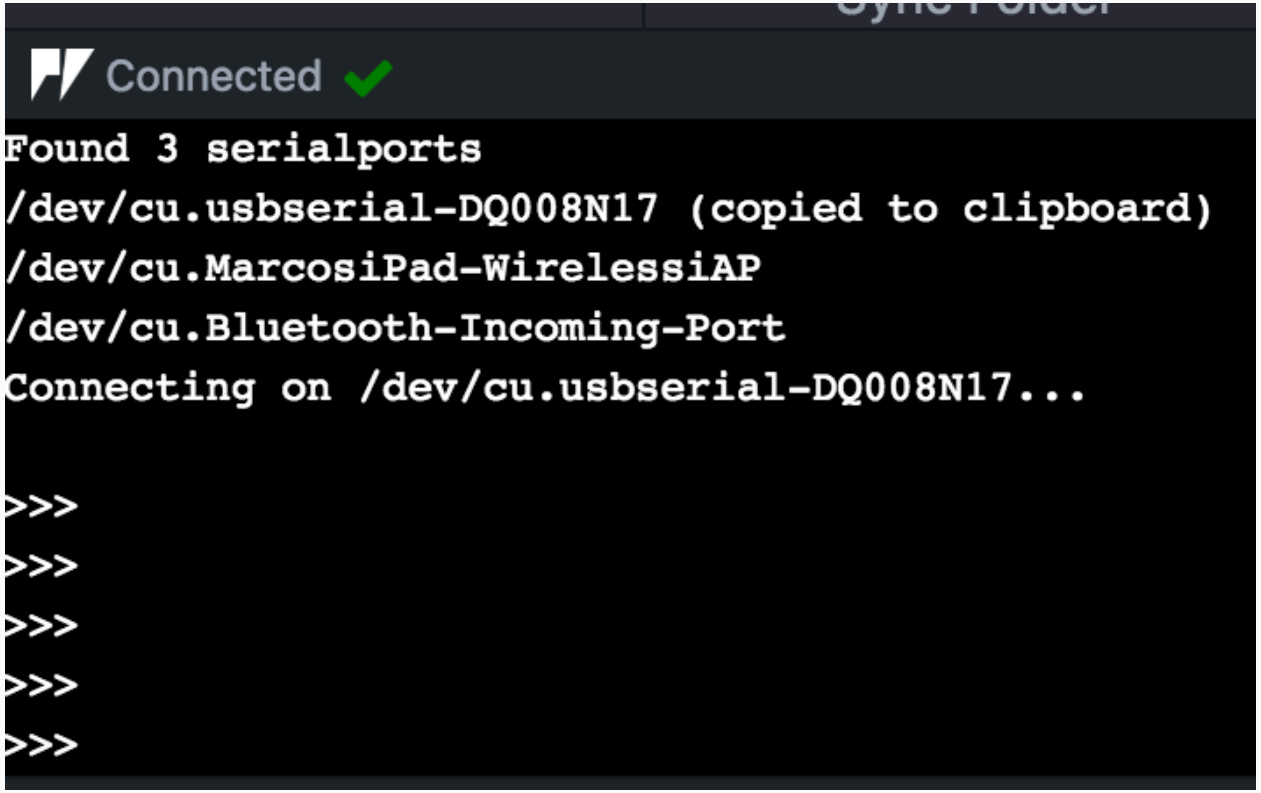
workflow: insert correct device address



workflow: connect!



workflow: REPL



REPL console

REPL stands for Read Print Eval Loop. Simply put, it takes user inputs, evaluates them and returns the result to the user.

You have a complete python console!

Try to enter 2+2 and press Enter.

Now enter:

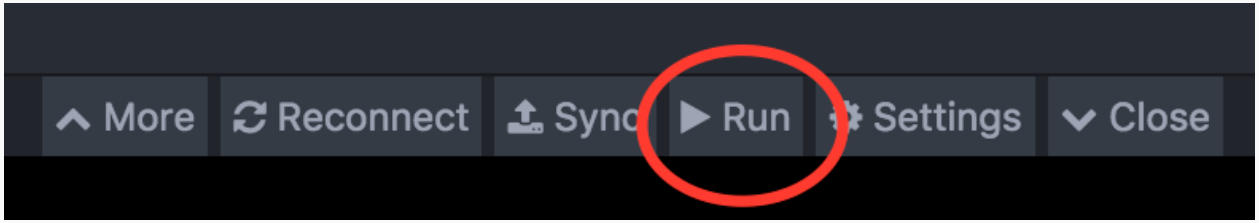
print("Hi! I am a python shell!")

executing code

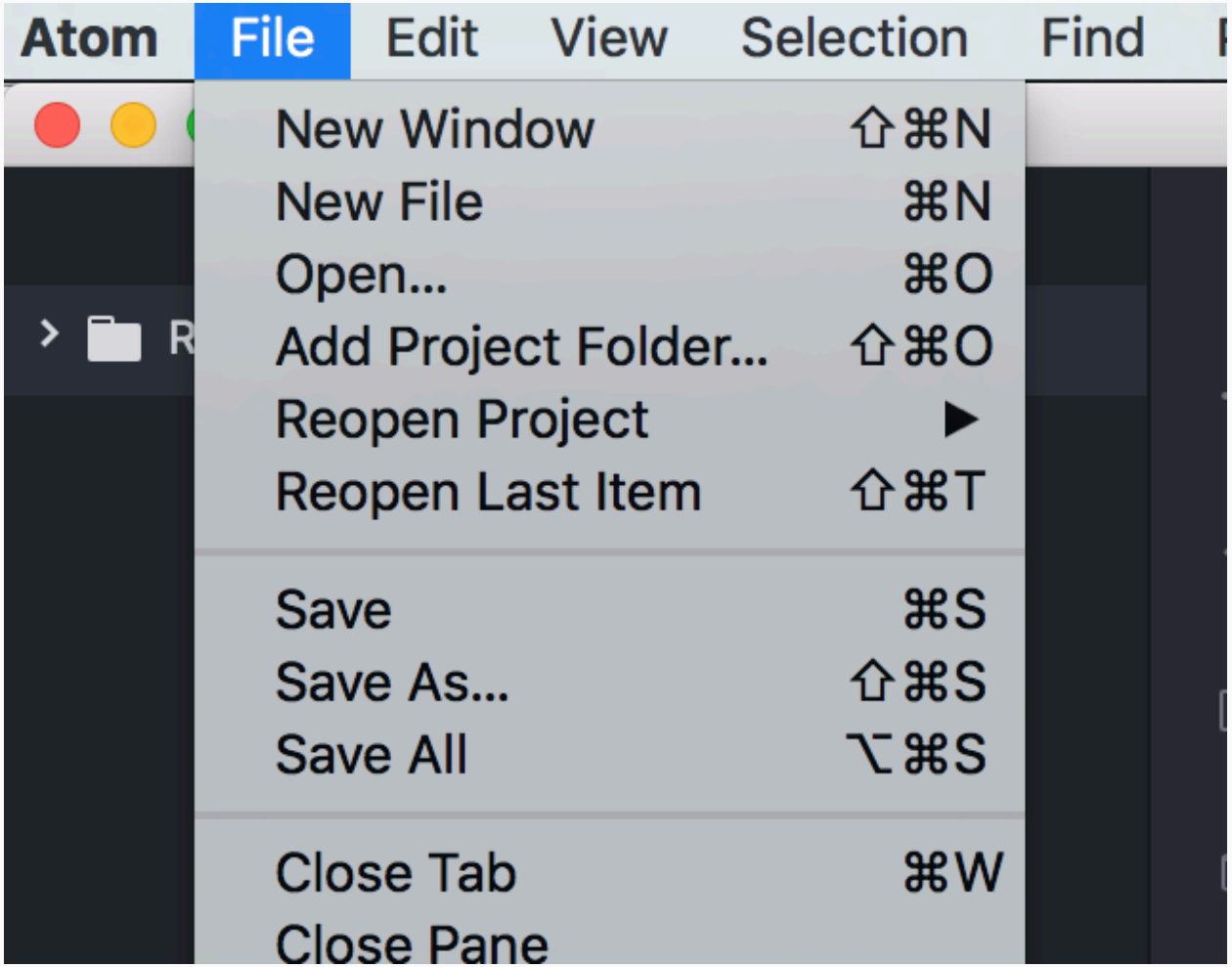
There are three ways to execute code on a Pycom device:

1. Via the REPL shell. Useful for single commands and for testing.
2. Using the Run button. Code in the Atom editor will be executed, but will not be stored in the device. If you reboot, the code will not be executed again.
3. Synching the device with the Project folder in Atom.

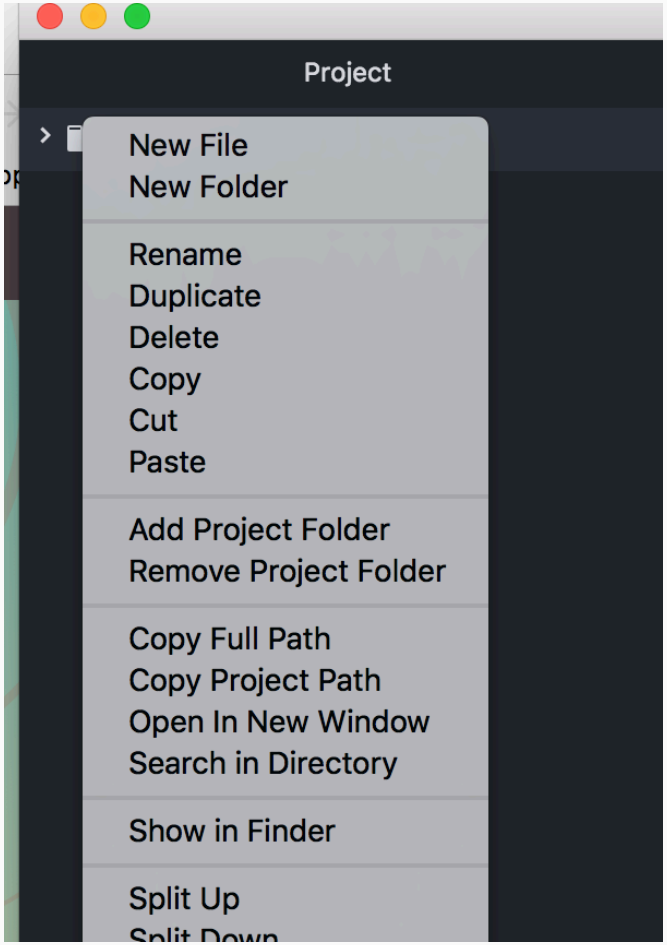
workflow: Run



workflow: add Project folder



workflow: ONE Project folder



It is easier if you only have one Project folder. Make sure you Remove any other Project folders and keep only the one you want to use.

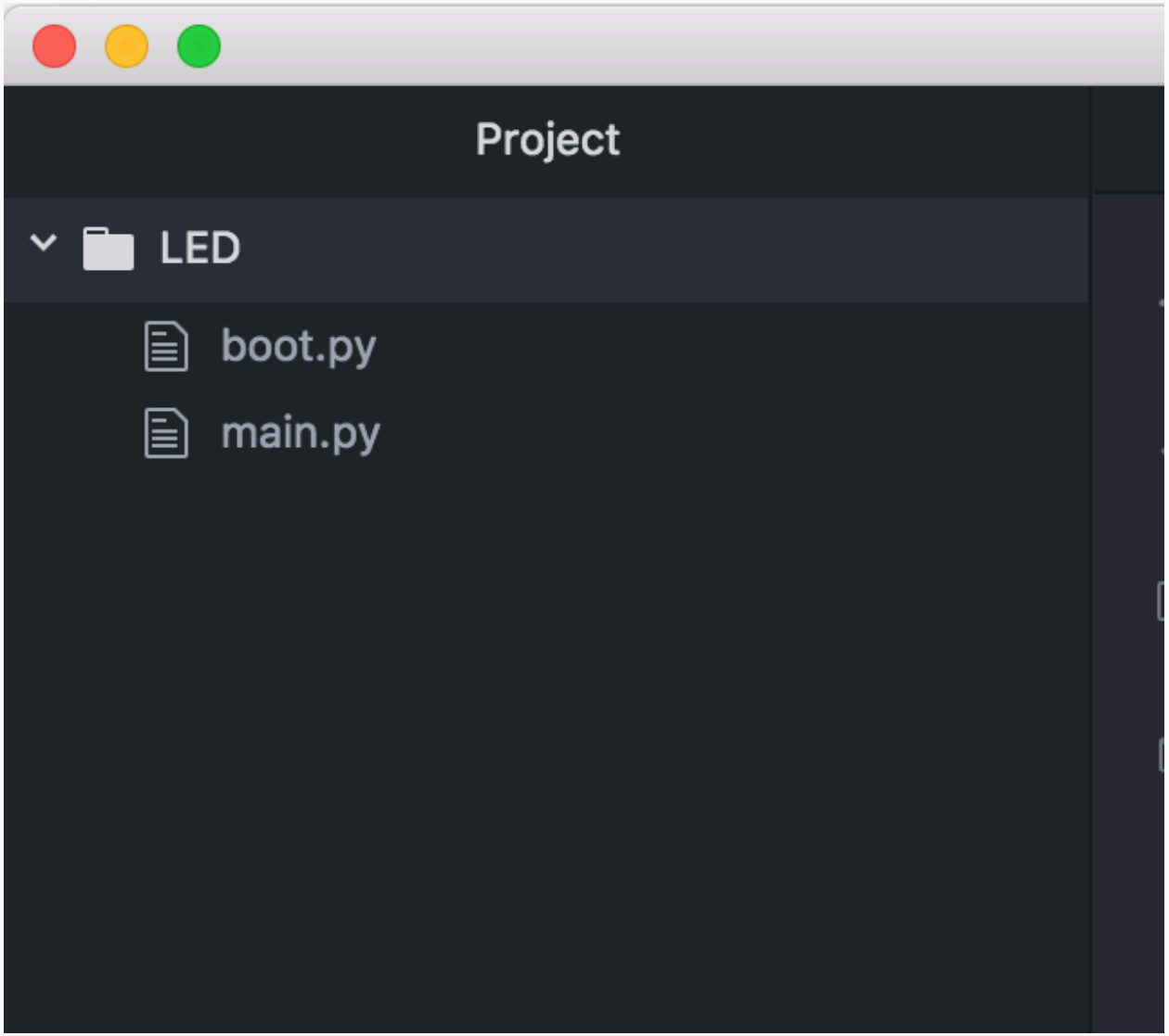
workflow: Project folder

The Project folder should contain all the files to be synched with the device.

You should always have two files: **boot.py** (executed at boot time) and **main.py** (containing the main code).

The folder can also include libraries and other python source code.

workflow: example of Project folder



workflow: boot.py

The boot.py file should always start with following code, so we can run our Python scripts over Serial or Telnet.

```
from machine import UART
import os
uart = UART(0, 115200)
os.dupterm(uart)
```

LED

LED

In this example, we will create and deploy the proverbial 1st app, “Hello, world!” to a Pycom device.

The WiPy module has one LED (big, white LED on the same side as the microUSB).

code: LED

Check the LED folder and sync the two files to your active project folder.

Try to send an SOS message using the LED. The SOS is line-line-line-dot-dot-dot-line-line-line in morse code, where a line is three times longer than a dot.