Why we will use microPython

Rapid prototyping with microPython devices

Marco Zennaro, ICTP

June 14, 2018

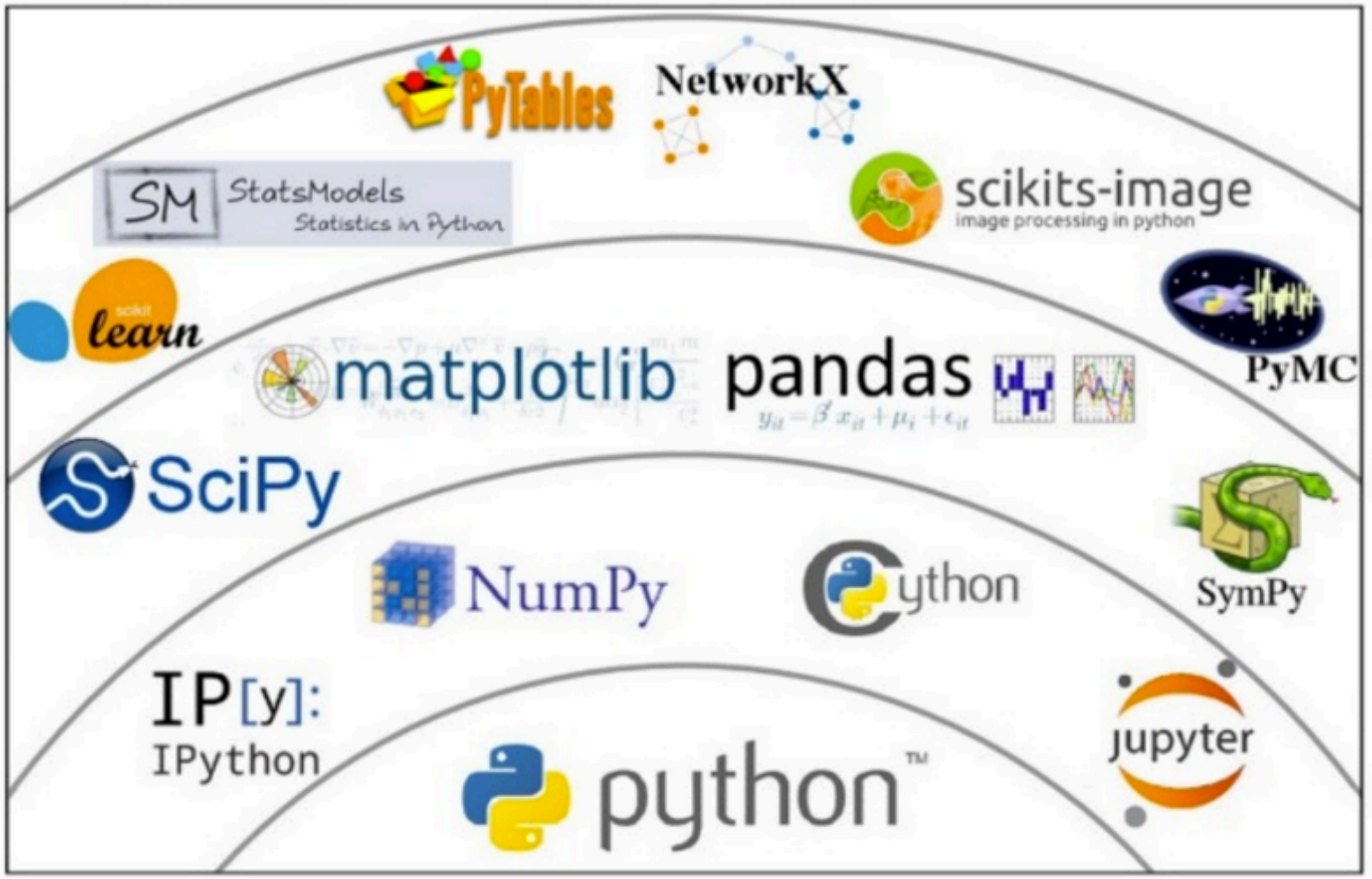
Why micropython?

python

Worldwide, Apr 2018 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Java	22.62 %	-0.8 %
2		Python	22.05 %	+5.2 %
3	↑↑	Javascript	8.56 %	+0.2 %
4	↓	PHP	8.22 %	-1.8 %
5	↓	C#	7.95 %	-0.7 %
6		C	6.38 %	-1.1 %
7	↑	R	4.26 %	+0.4 %
8	↓	Objective-C	3.7 %	-1.0 %

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

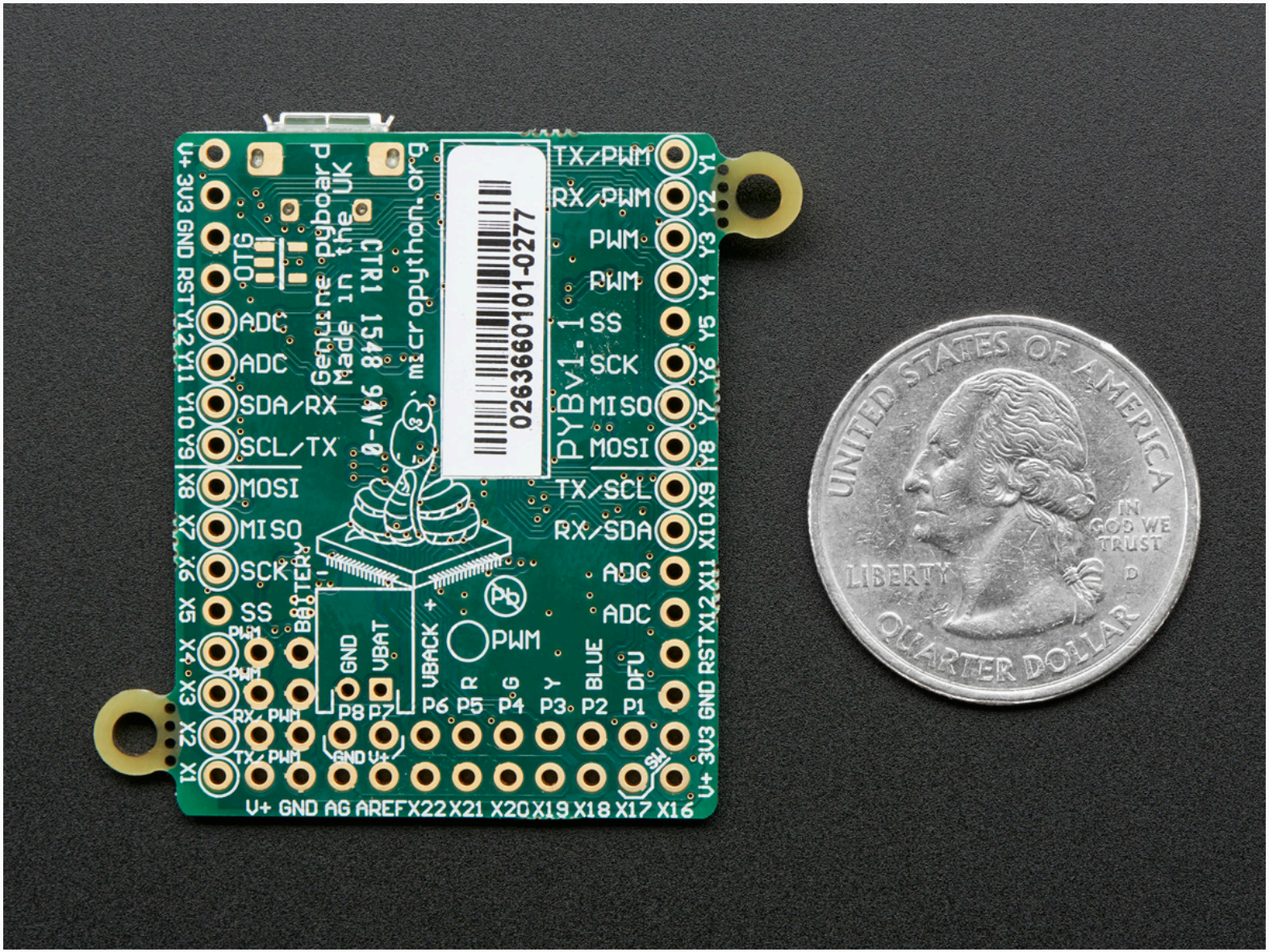
The MicroPython **pyboard** is a compact electronic circuit board that runs MicroPython on the bare metal, giving you a low-level Python operating system that can be used to control all kinds of electronic projects.

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


Genuine pyboard
Made in the UK
micropython.org
PYBv1.1
TX/PWM
RX/PWM
PWM
PWM
SS
SCK
MISO
MOSI
TX/SCL
RX/SDA
ADC
ADC
ADC
SDA/RX
SCL/TX
MOSI
MISO
SCK
SS
PWM
PWM
PWM
GND
VBAT
VBACK
R
G
Y
BLUE
DFU
SW
BATTERY
OTG
LIBERTY
IN GOD WE TRUST
QUARTER DOLLAR
UNITED STATES OF AMERICA

MicroPython pyboard feature table

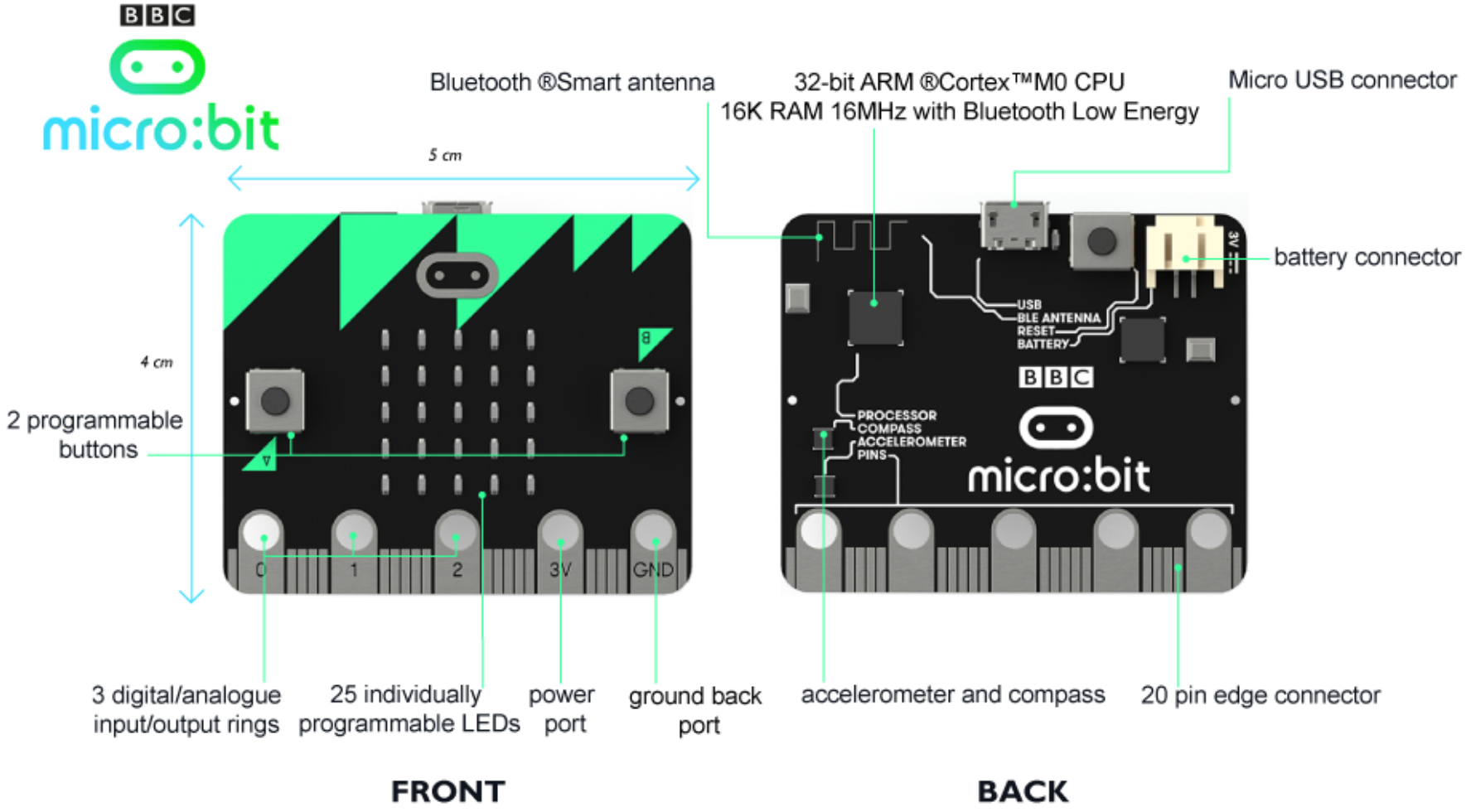
BOARD			
description	The original pyboard v1.1	Pyboard lite v1.0 with accelerometer	Pyboard lite v1.0
SKU	PYBv1.1	PYBLITEv1.0-AC	PYBLITEv1.0
PRICE			
GBP incl. tax	£28.00	£22.60	£19.60
approx EUR incl. tax	€39.20	€31.60	€27.40
approx USD excl. tax	$35.00	$28.25	$24.50
MICROCONTROLLER			
MCU	STM32F405RGT6	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

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

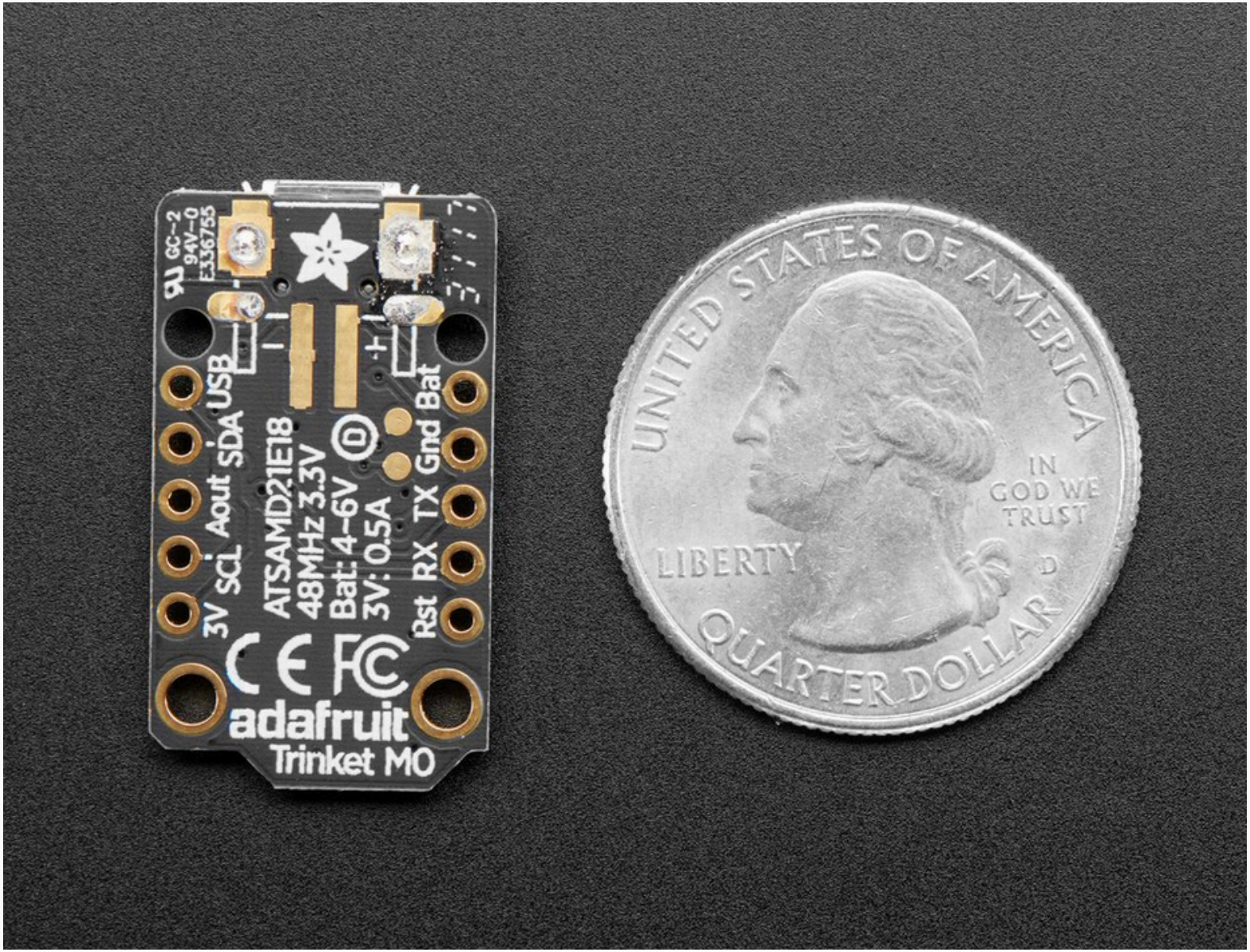


BBC Micro:bit

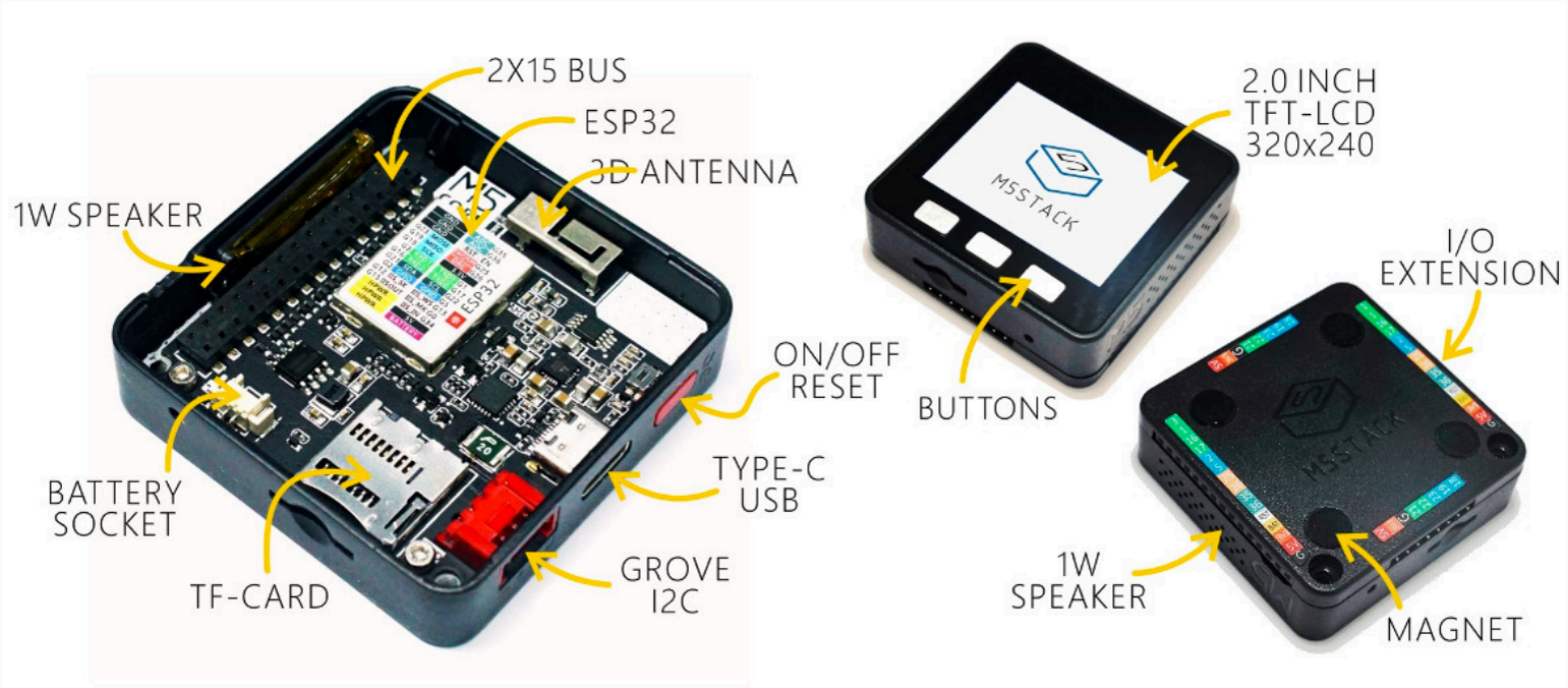
The Micro Bit is an ARM-based embedded system designed by the BBC for use in computer education in the UK.

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.

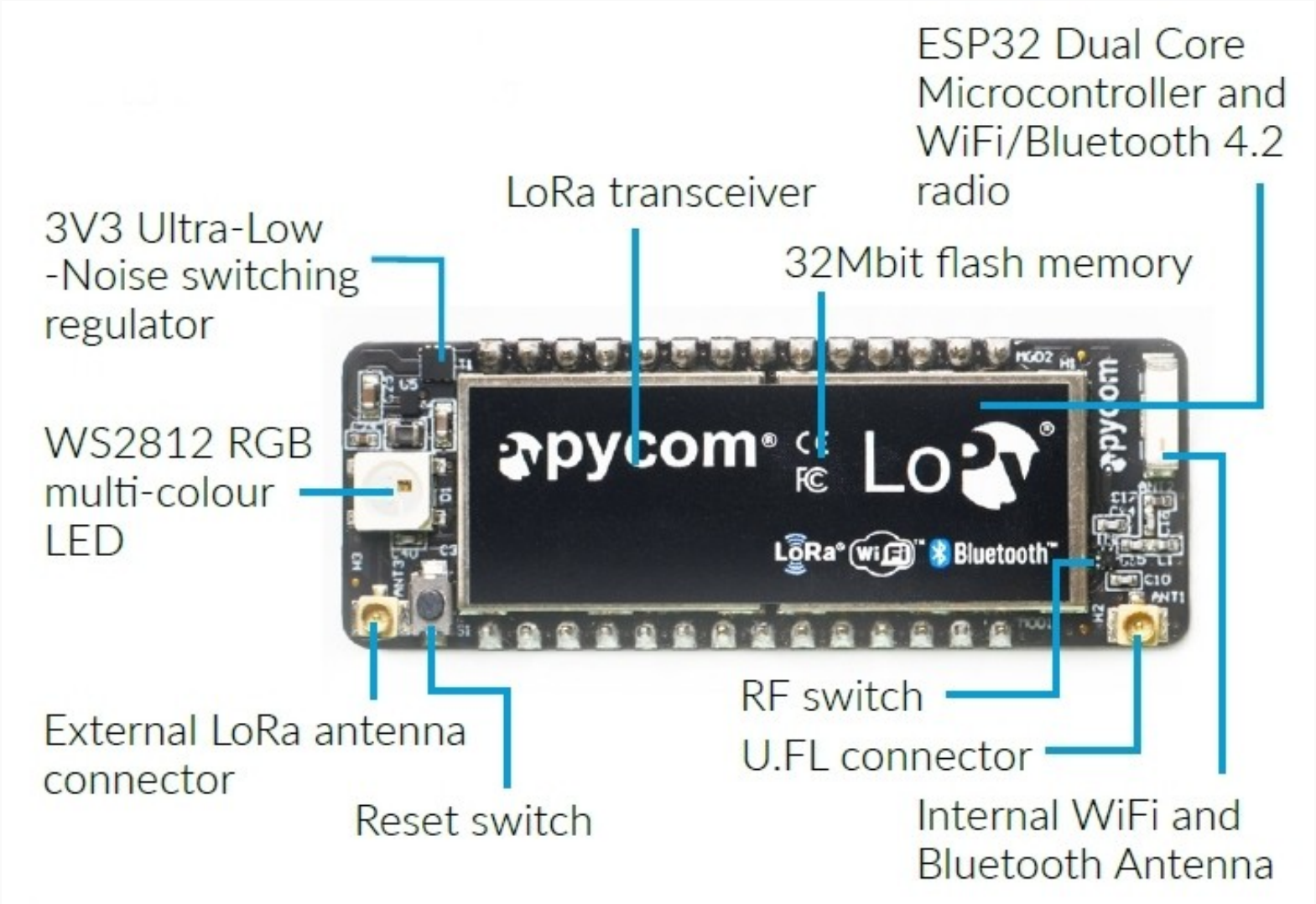
Trinket


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

M5stack


2X15 BUS
ESP32
3D ANTENNA
1W SPEAKER
ON/OFF
RESET
TYPE-C
USB
BATTERY
SOCKET
TF-CARD
GROVE
I2C
2.0 INCH
TFT-LCD
320x240
BUTTONS
I/O
EXTENSION
1W
SPEAKER
MAGNET

pycom: LoPy

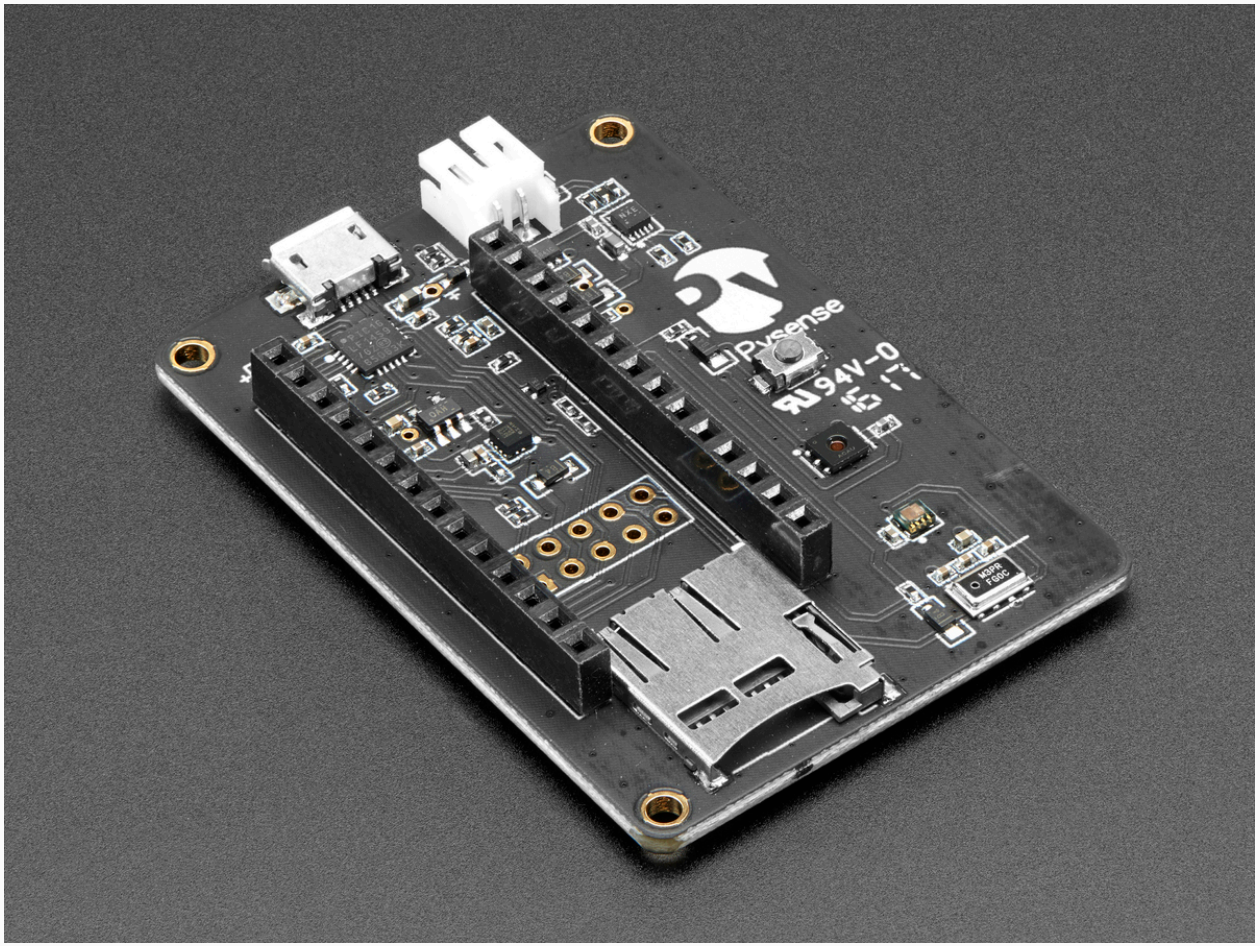

ESP32 Dual Core Microcontroller and WiFi/Bluetooth 4.2 radio
LoRa transceiver
32Mbit flash memory
3V3 Ultra-Low -Noise switching regulator
WS2812 RGB multi-colour LED
pycom
LoRa
WiFi
Bluetooth
RF switch
U.FL connector
External LoRa antenna connector
Reset switch
Internal WiFi and Bluetooth Antenna

pycom: LoPy4

- Espressif ESP32 chipset
- Quadruple network MicroPython enabled development board (LoRa, Sigfox, WiFi, Bluetooth)
- RAM: 4MB (vs 512KB)
- External flash: 8MB (vs 4MB)

pycom: Expansion Board

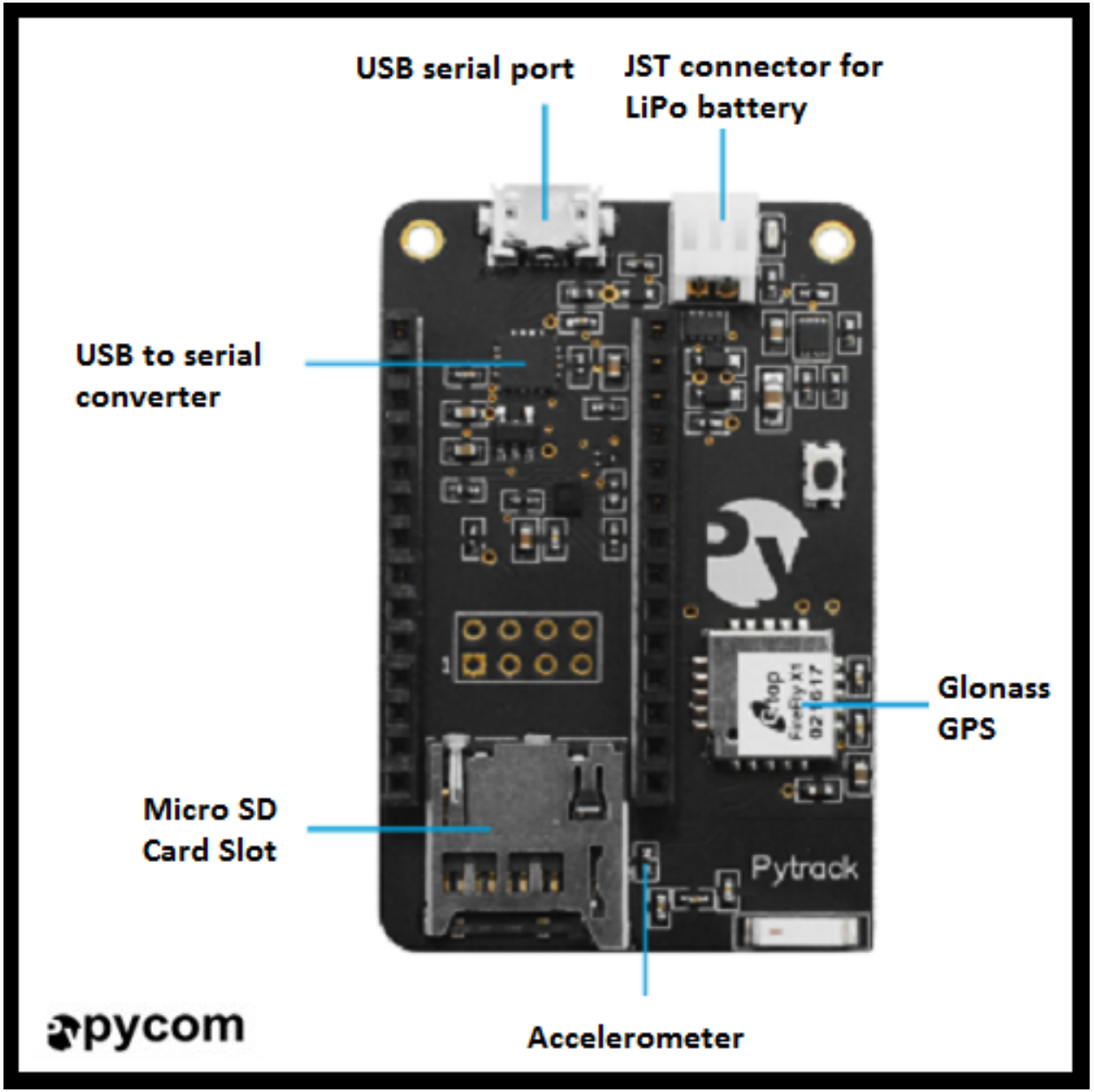
pycom: PySense



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)

Our Lab equipment

- Pycom LoPy4
- PySense
- microUSB cable

Plan of the lab session

plan for the lab session

During the lab session we will cover:

1. Pycom workflow
2. Hello World for IoT: LED switching
3. Reading sensors using the PySense and sending them to Ubidots via WiFi

You will have simple code snippets and will develop (a bit) more complex code as exercise.

workflow: using the command line tool

You will use a command tool to access the LoPy. The tool is already installed on the Linux machines in the Lab.

Plug in your LoPy device in a USB port and wait a few seconds until the LED starts blinking in blue.

Open a Terminal.

Then execute:

```
$ python3 -m there detect
```

This command will list the serial ports and will automatically find your LoPy board, listed as PySense.

workflow: uploading code + REPL

To upload code on the LoPy:

```
$ python3 -m there push *.py /flash
```

To upload the entire directory (including directories) on the LoPy:

```
$ python3 -m there push -r * /flash
```

Once you are done uploading, start a serial terminal and get access to the REPL:

```
$ python3 -m there -i
```

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. **Synching** the device with a folder on your PC. In this way, the code is stored in the Pycom device and will be executed again if you reboot the device.

workflow: Run

Now that you have synched the folder with your LoPy, you must soft-reboot it and the code will be executed.

Press **Control + D** to soft-reboot your LoPy.

You can also execute the code by calling from the REPL console:

```
import main
```

LED

LED

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

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

code: LED

Create a file called main.py on your PC, and copy the code found here:

https://github.com/pmanzoni/pythonMQTT_062018/tree/master/code

Now copy the file to the LoPy and execute it (by soft-reboot).

What do you see?