Why we will use microPython

Rapid prototyping with microPython devices

Marco Zennaro, ICTP

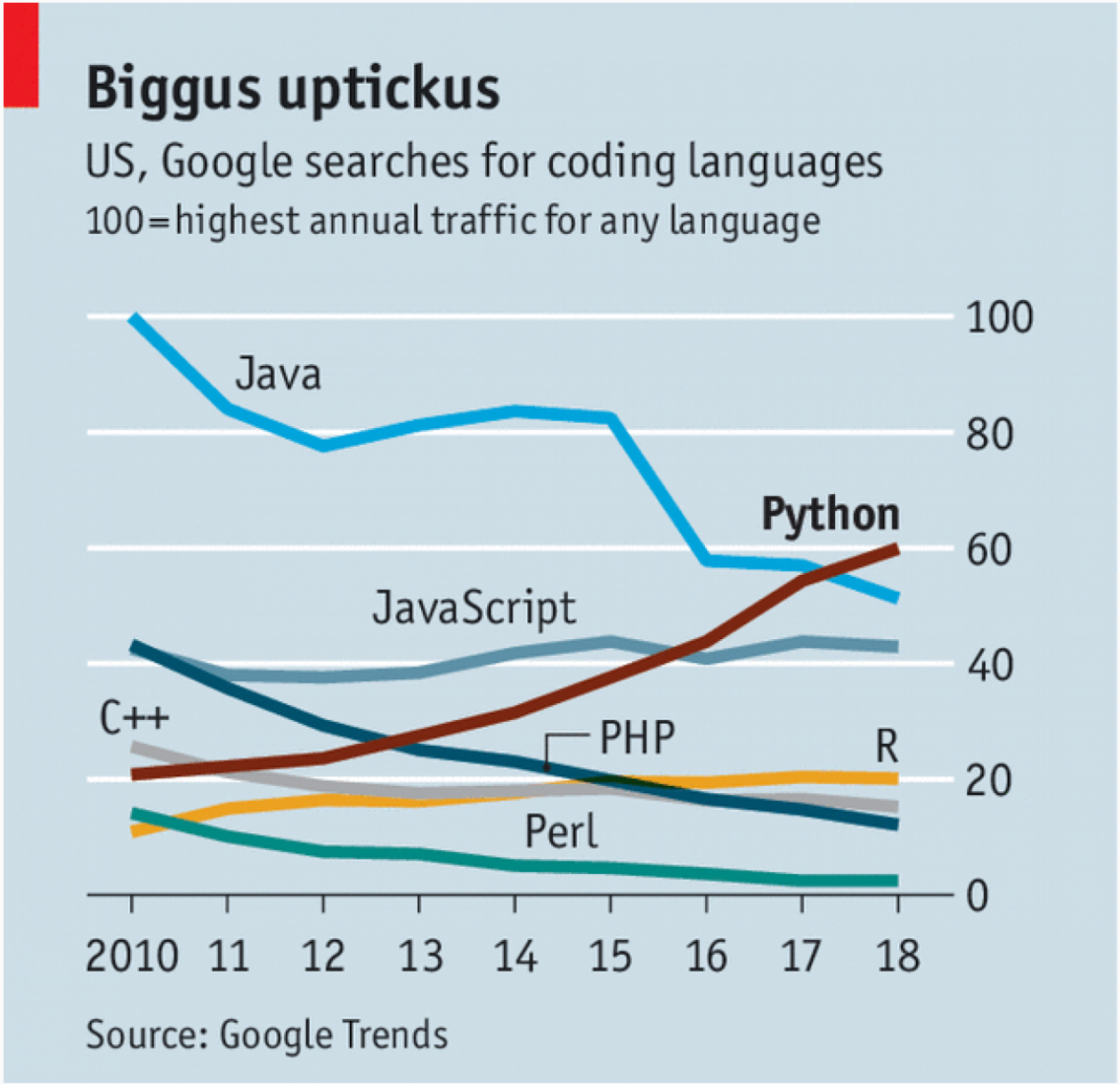
January 21, 2019

Why micropython?

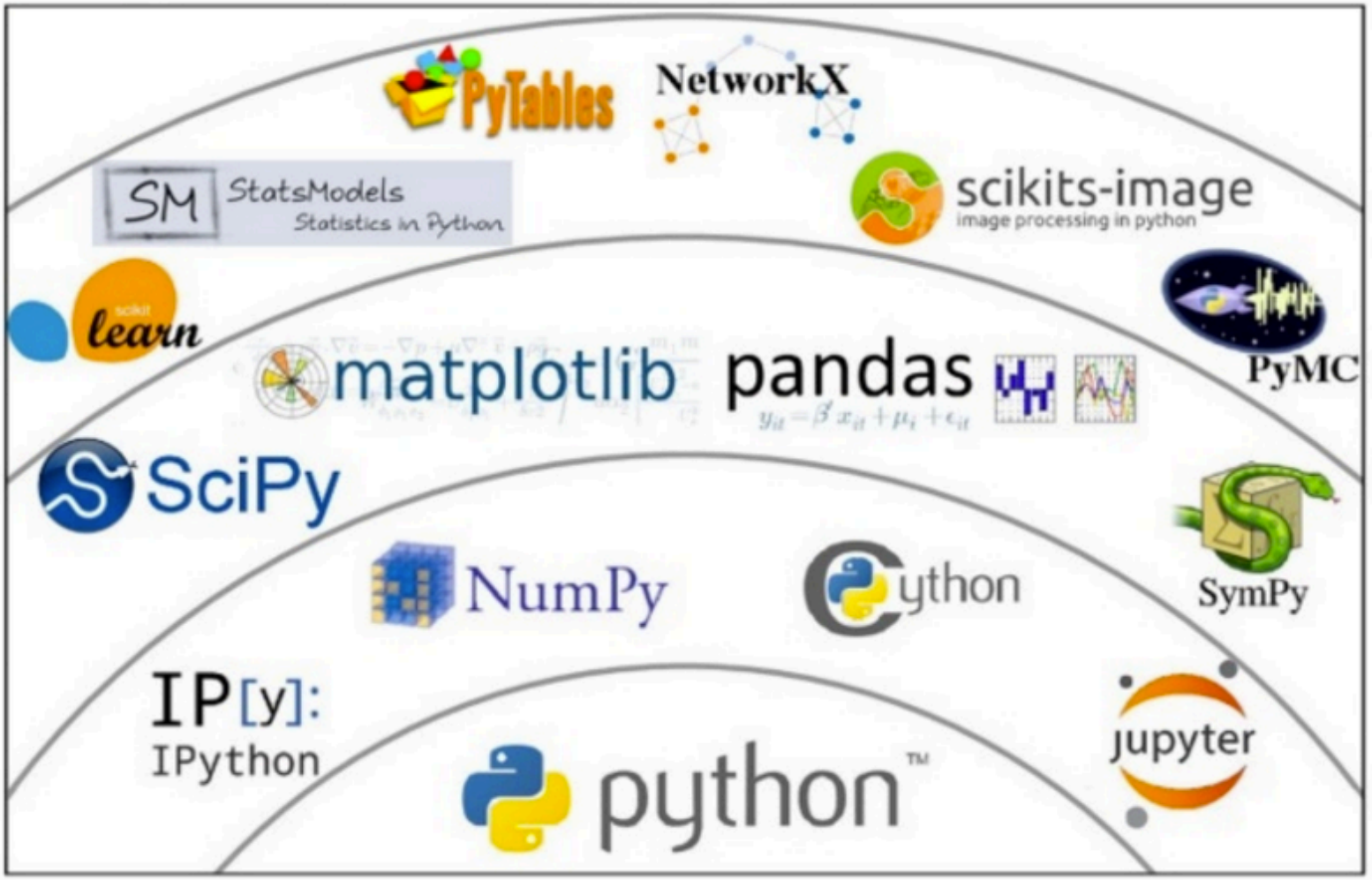
Programming languages

Python has brought computer programming to a vast new audience

And its inventor has just stepped down


Biggus uptickus
US, Google searches for coding languages
100=highest annual traffic for any language
Java
Python
JavaScript
C++
PHP
R
Perl
100
80
60
40
20
0
2010
11
12
13
14
15
16
17
18
Source: Google Trends
Economist.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

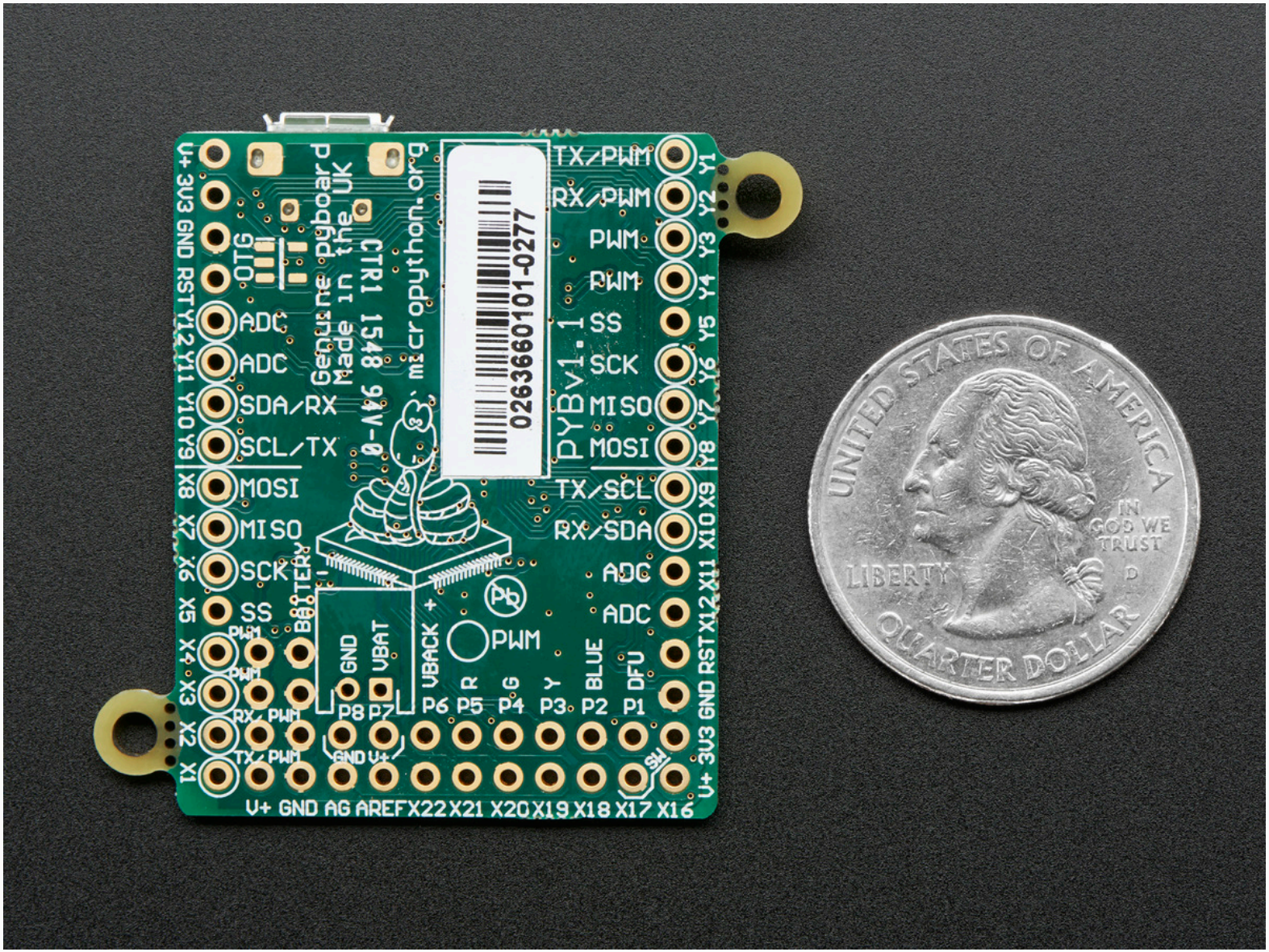
The MicroPython **pyboard** is a compact electronic circuit board that runs MicroPython on the bare metal, giving you a low-level Python operating system that can be used to control all kinds of electronic projects.

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


PYBv1.1
Genuine pyboard
Made in the UK
micropython.org
0263660101-0277
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

pyboard

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

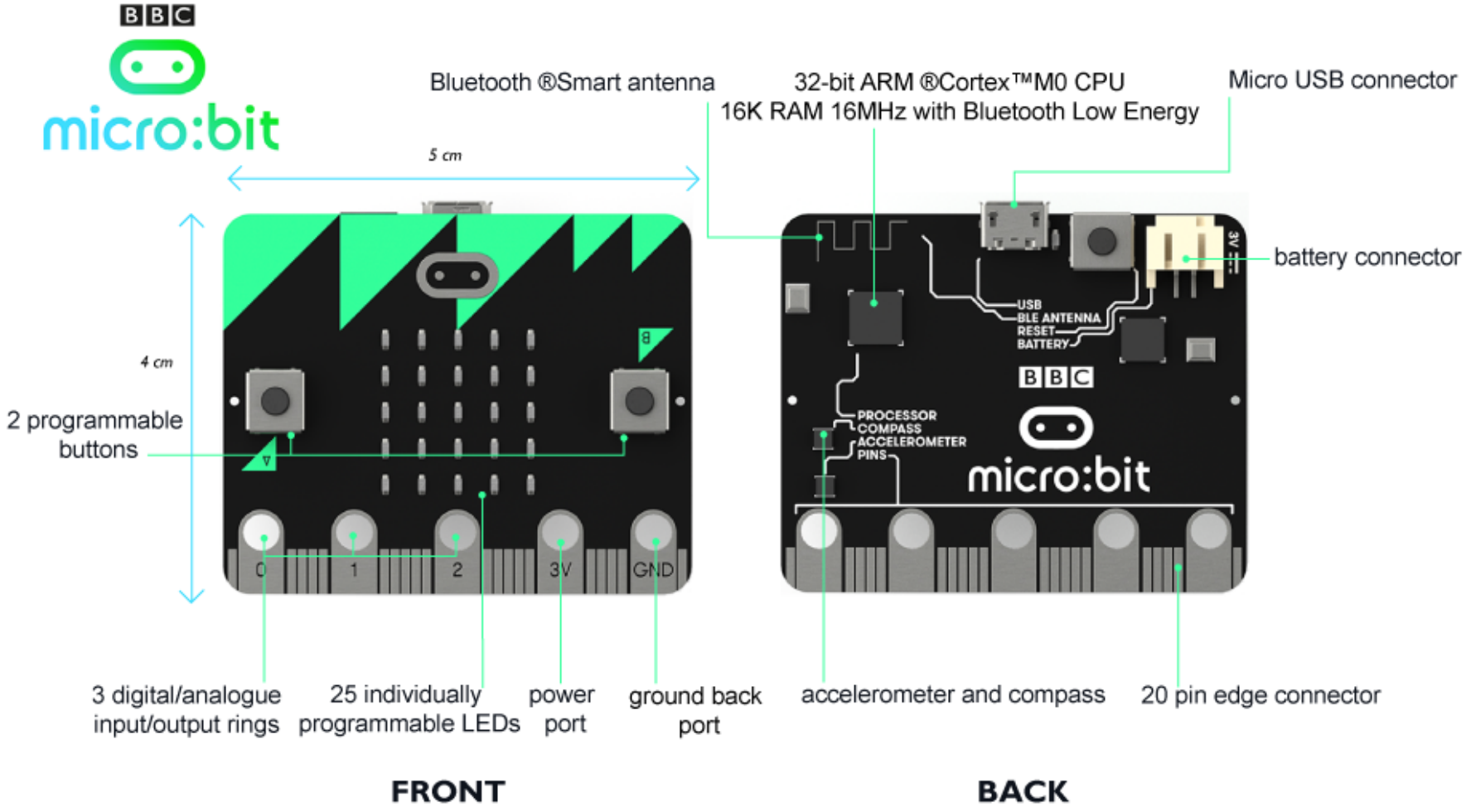
PERIPHERALS			
independent timers	13	7	7
hardware random number generator	yes	no	no
UART	5	3	3
I2C	2	2	2
SPI	2	2	2
CAN	2	0	0
MICROPYTHON CAPABILITIES			
internal flash fs	112k (94k usable)	64k (46k usable)	64k (46k usable)
approx heap size	100k	83k	83k
ADD-ONS			
LCD+touch skin compatible (LCD32MKv1.0)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

NEW

Add to compare

Digi XBee3™ Cellular LTE CAT 1

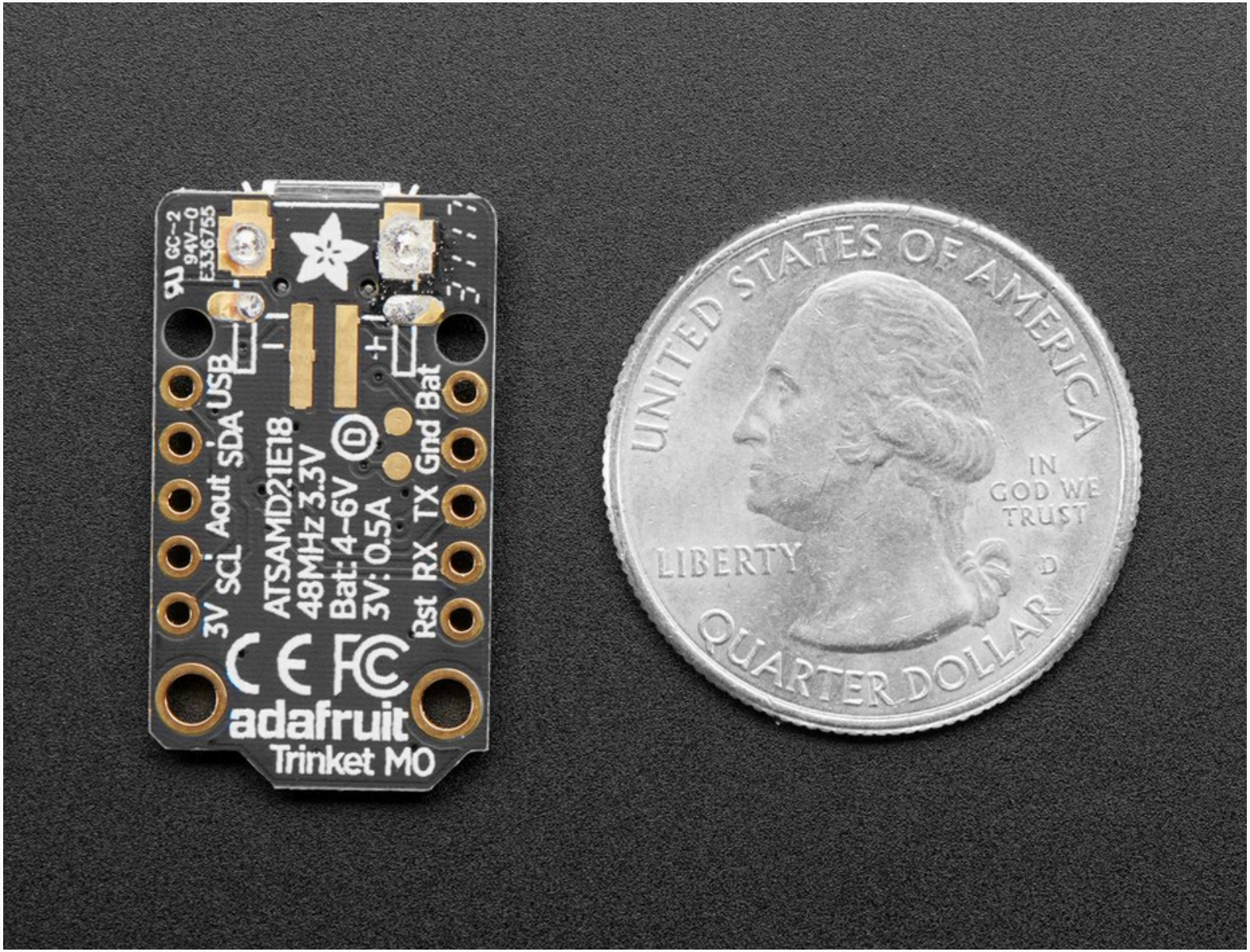
Digi XBee3™ smart modems offer the easiest way to integrate cellular...

Add to compare

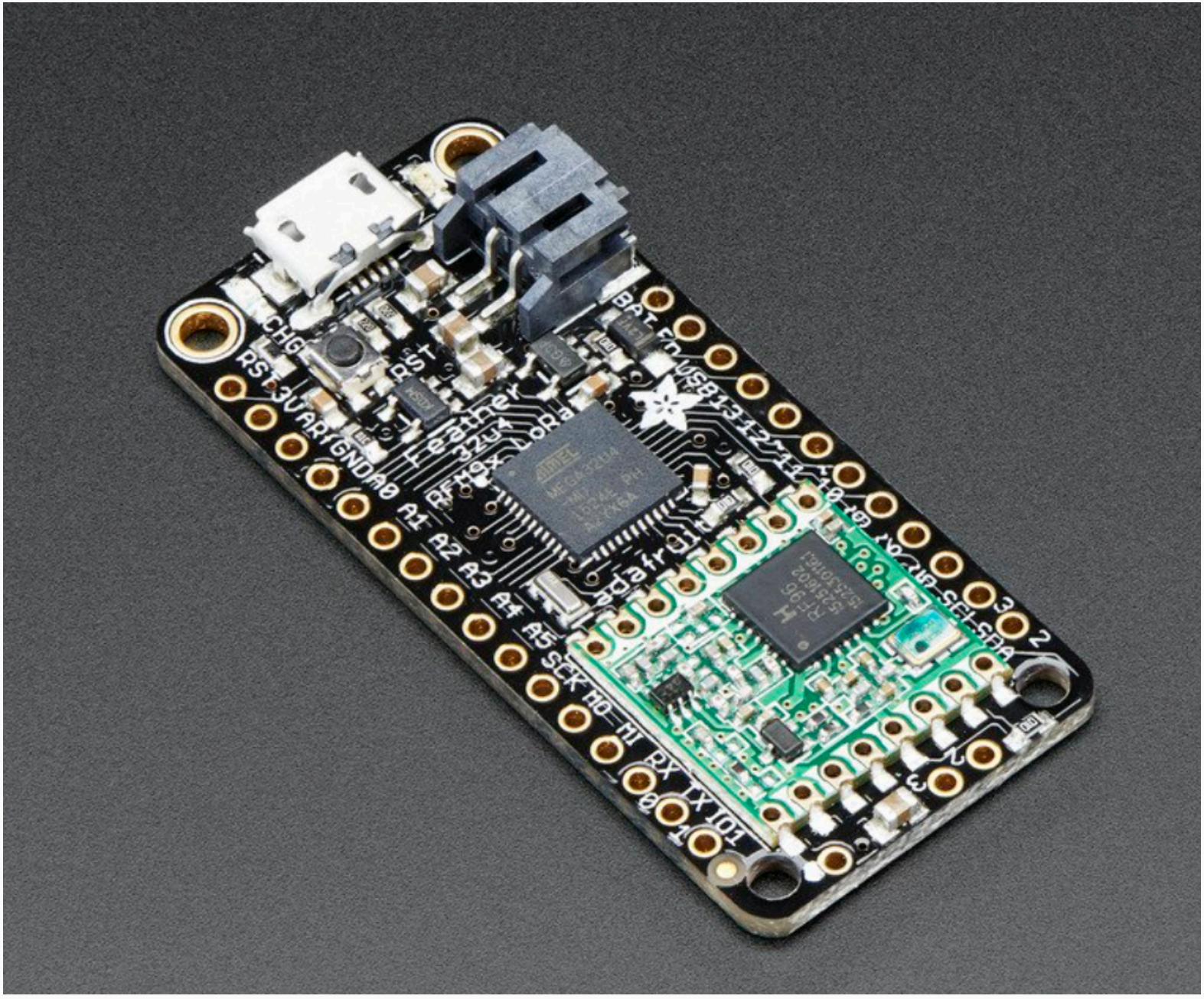
Digi XBee® Cellular LTE Cat 1

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

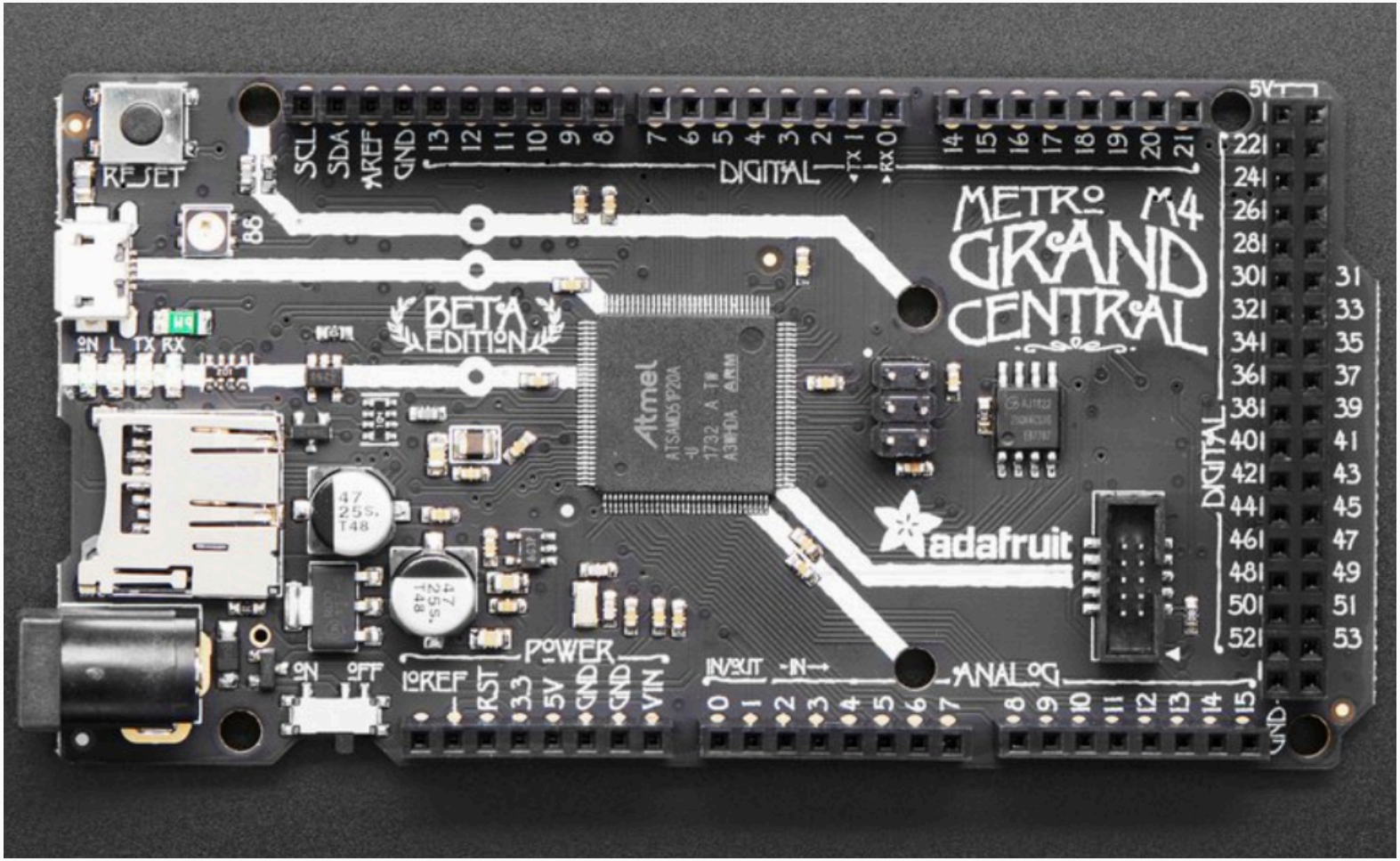
Trinket


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

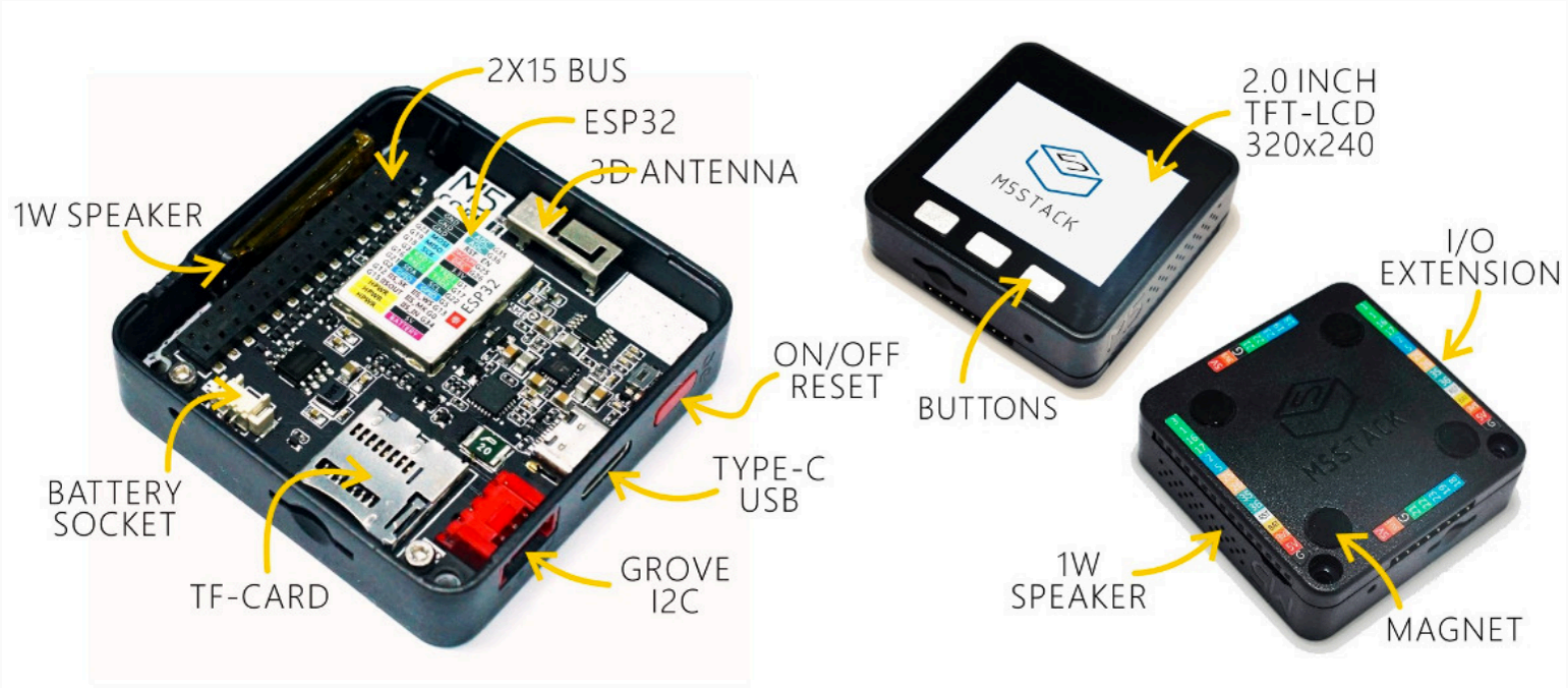
Feather 32u4 RFM95



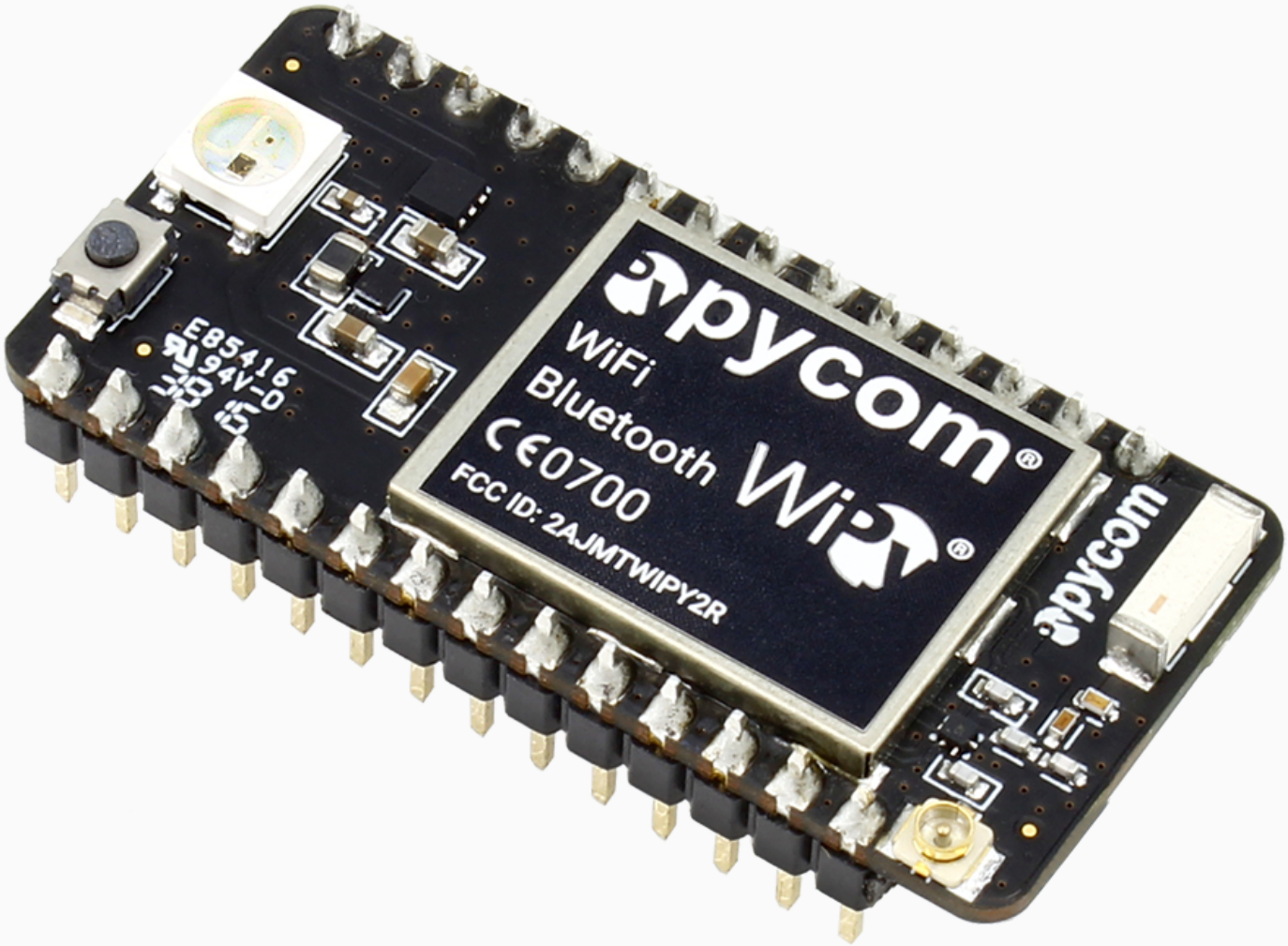
Grand Central M4 Express



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

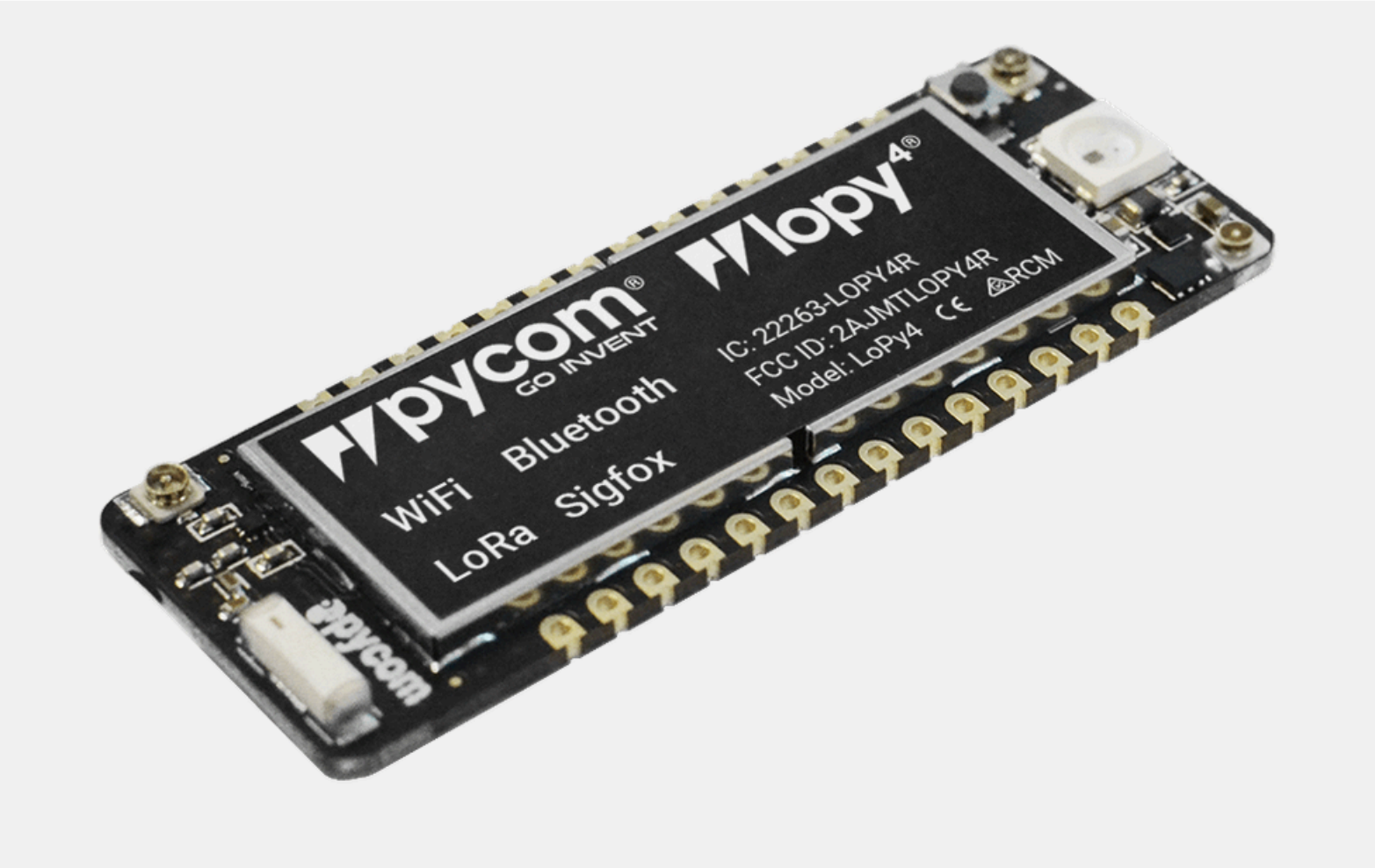
pycom: WiPy


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: LoPy4

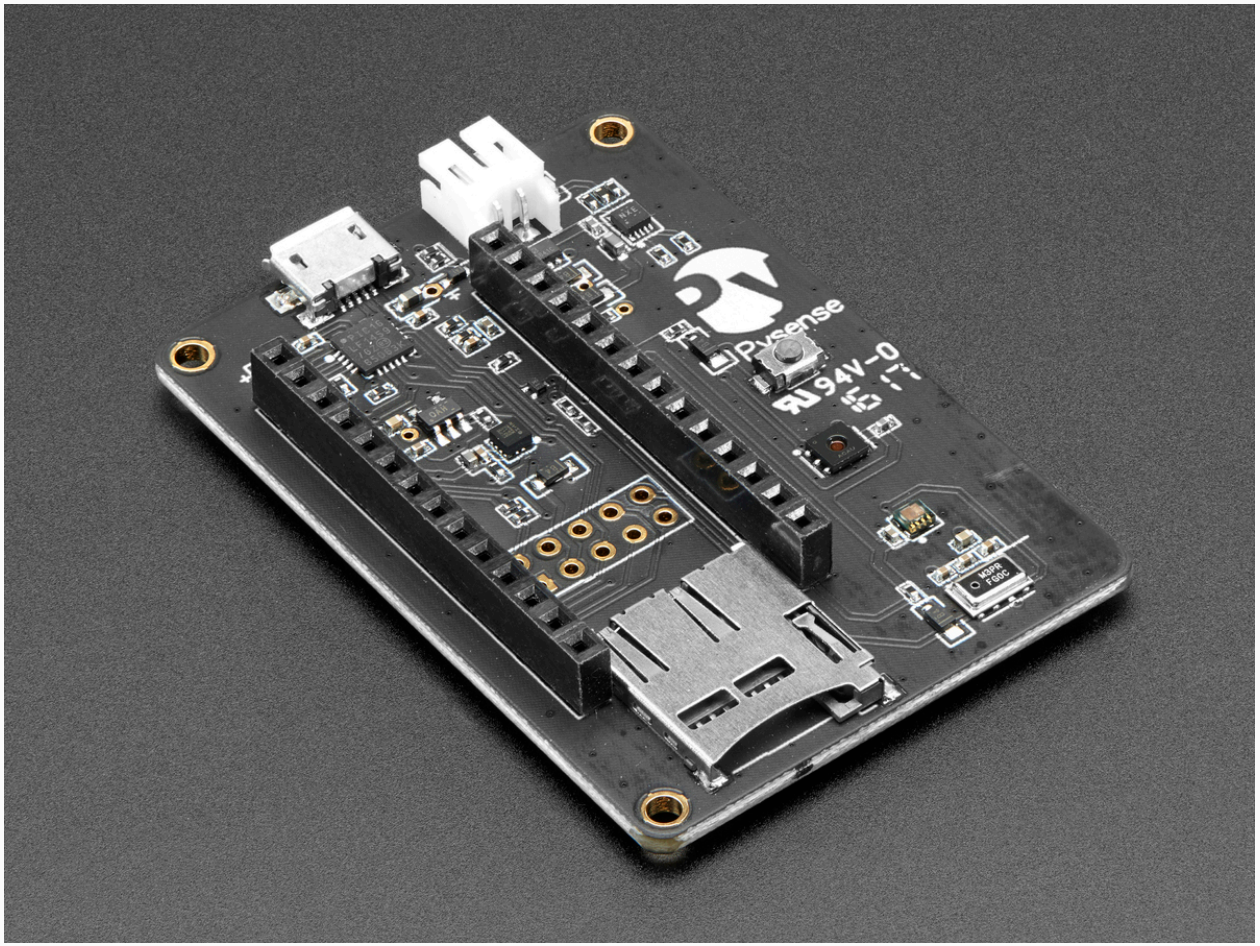

pycom
GO INVENT
lopy4
WiFi
Bluetooth
LoRa
Sigfox
IC: 22263-LOPY4R
FCC ID: 2AJMTLOPY4R
Model: LoPy4
RCM

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

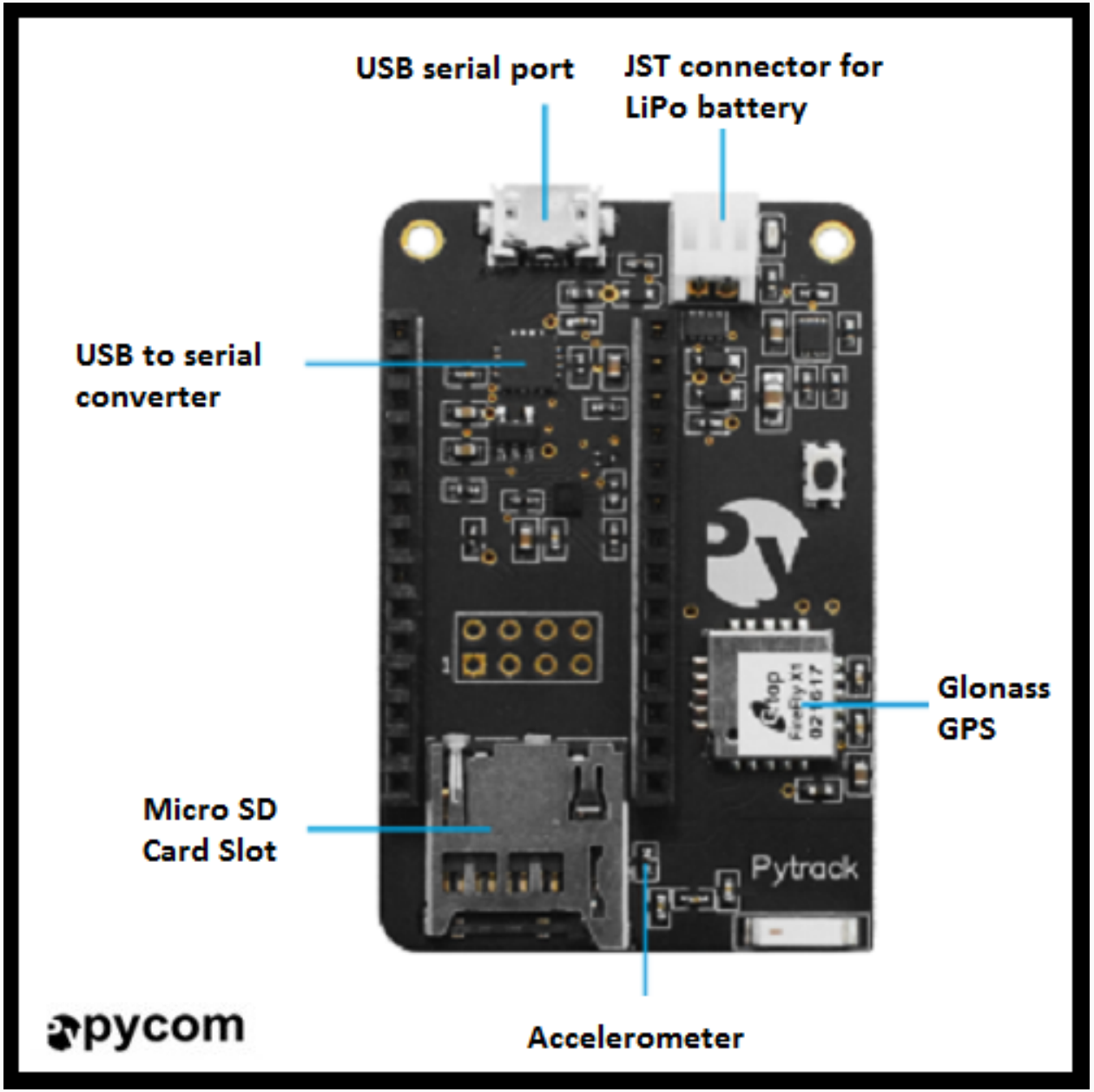
pycom: PySense



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

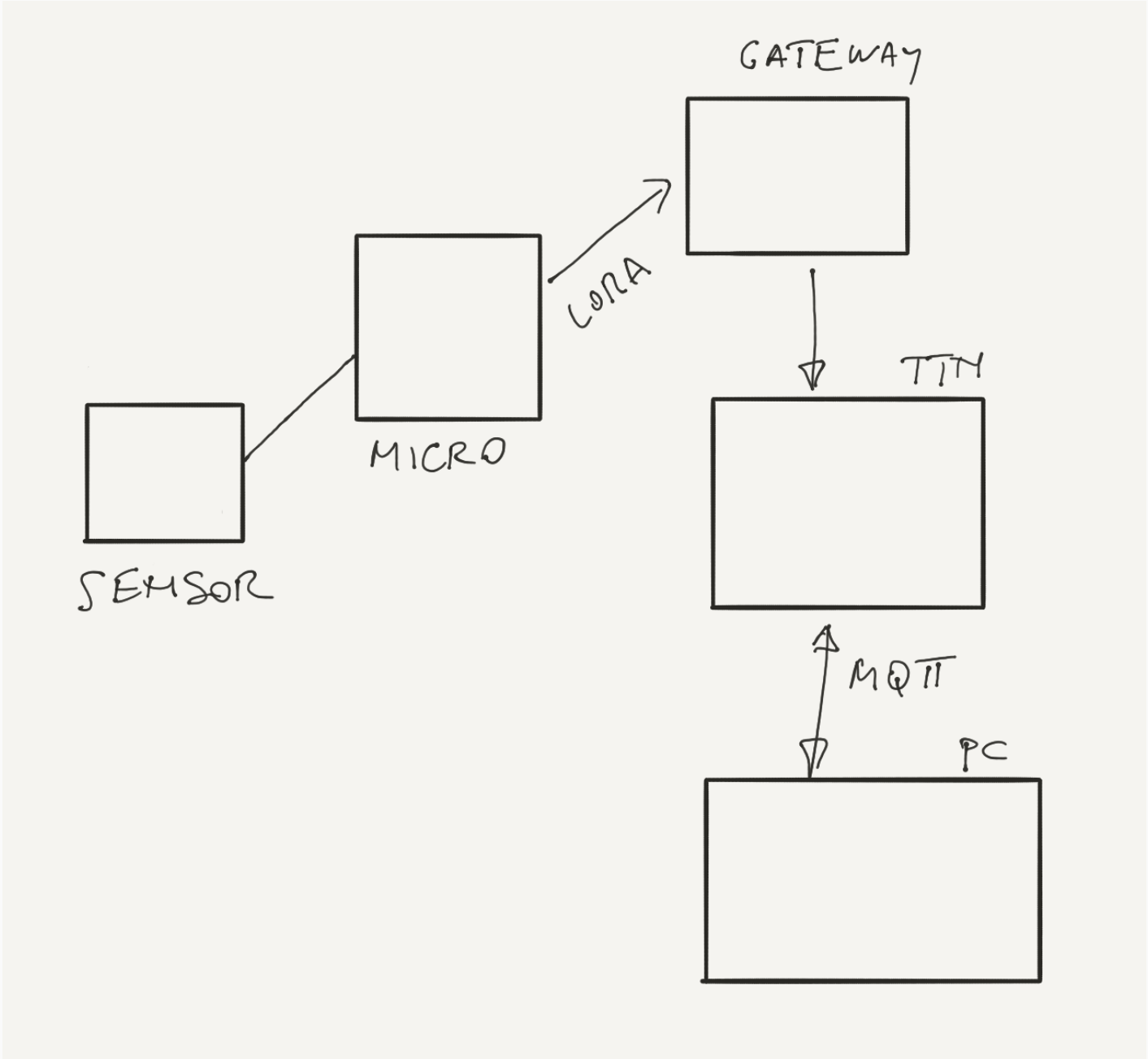
- GNSS + Glonass GPS
- 3 axis 12-bit accelerometer
- USB port with serial access
- LiPo battery charger
- MicroSD ard compatibility
- Ultra low power operation (1uA in deep sleep)

Our Lab equipment

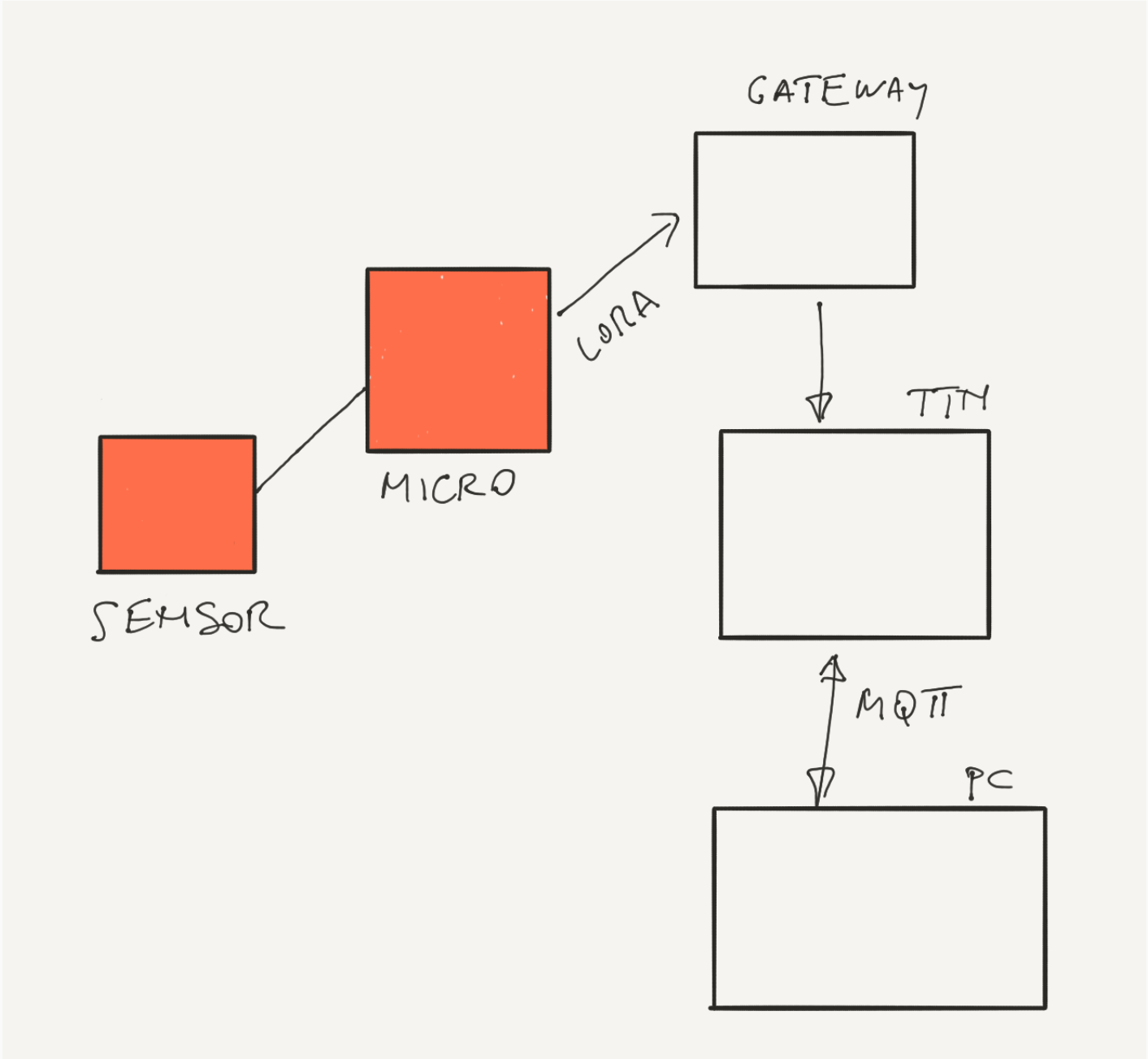
- Pycom LoPy4
- PySense
- microUSB cable

Plan of the week

overall plan



plan for today



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 internal flash
4. Reading sensors using the PySense
5. Using LoRaWAN
6. Using MQTT

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

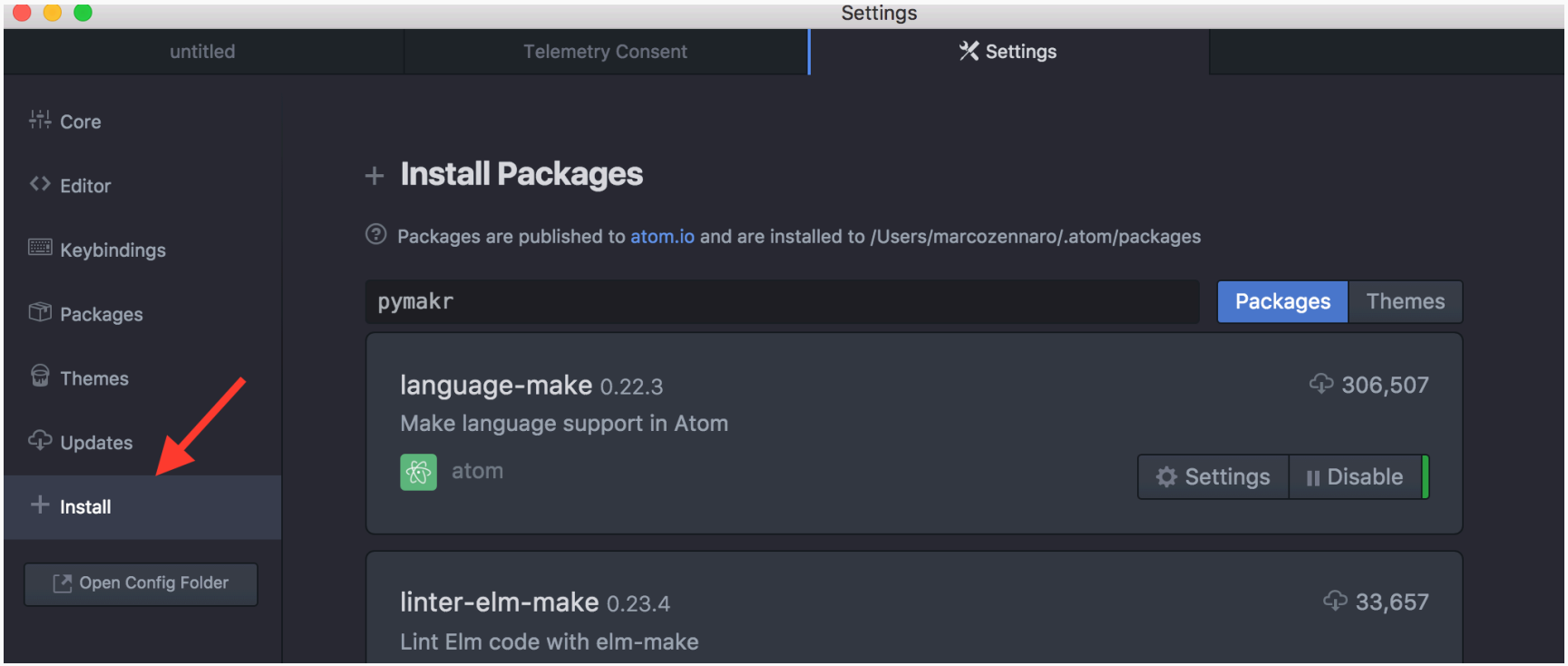
workflow: Atom

Please install Atom from

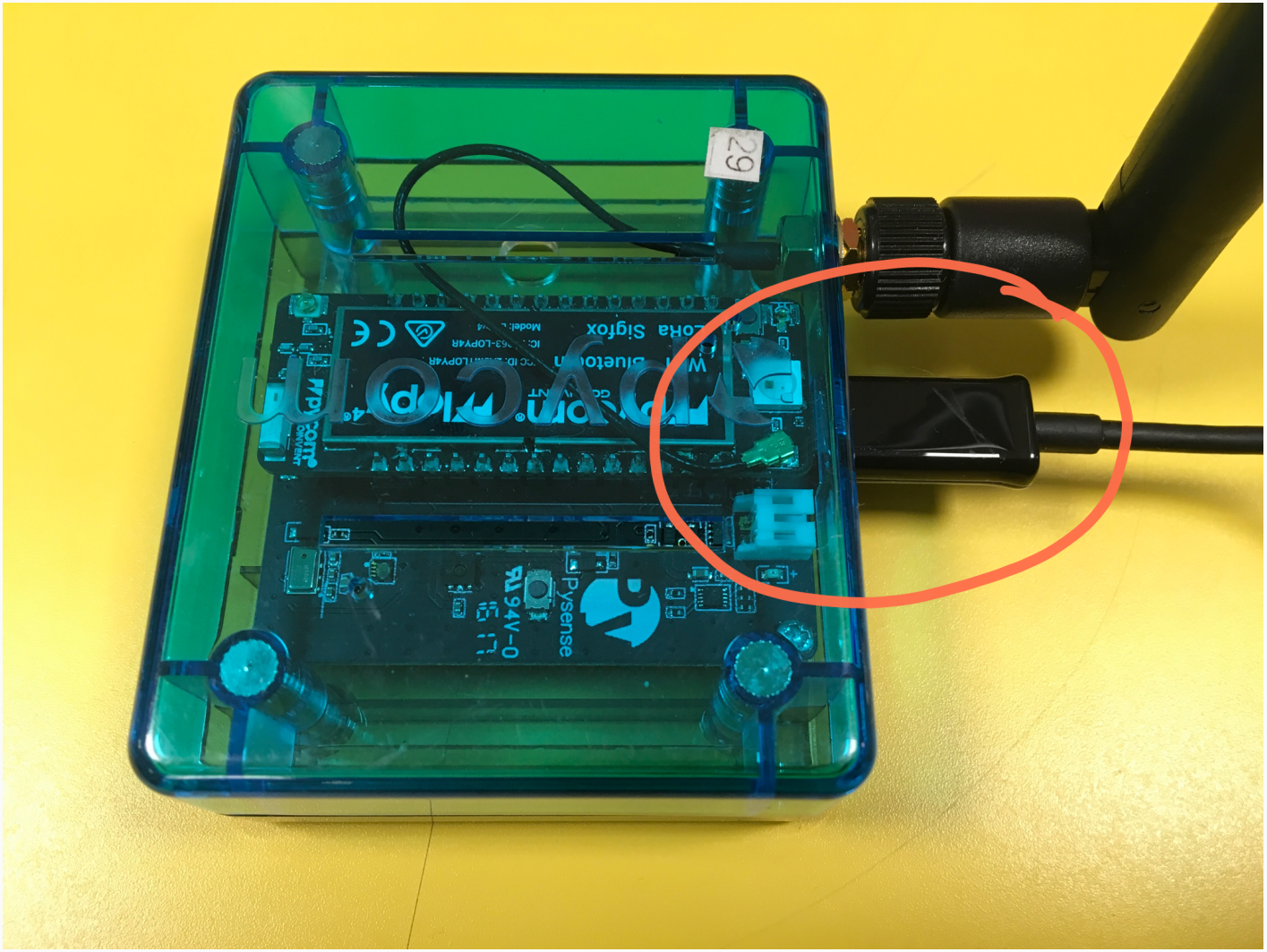
`www.atom.io`

workflow: install the pymakr package

Preferences -> Settings -> Install -> search **Pymakr**

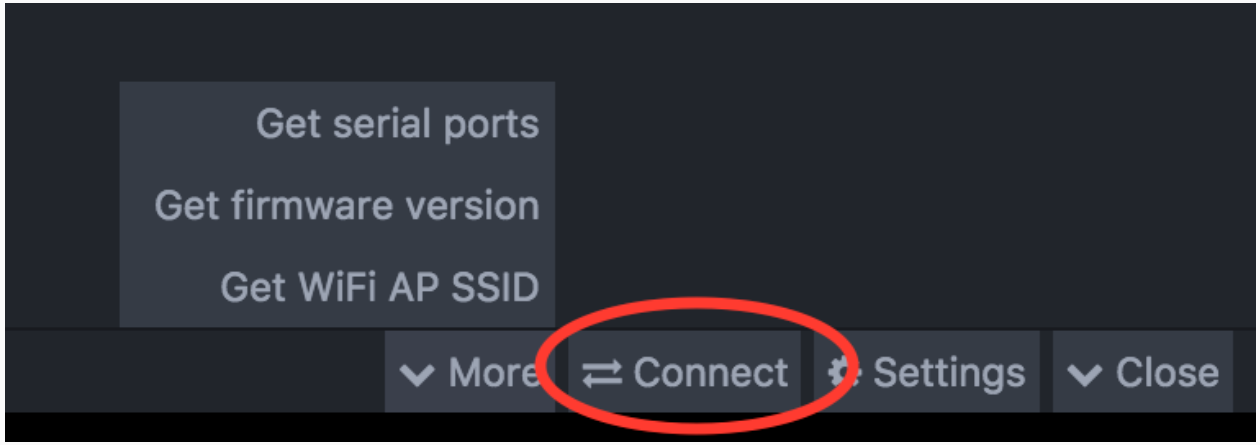


workflow: connect board via USB

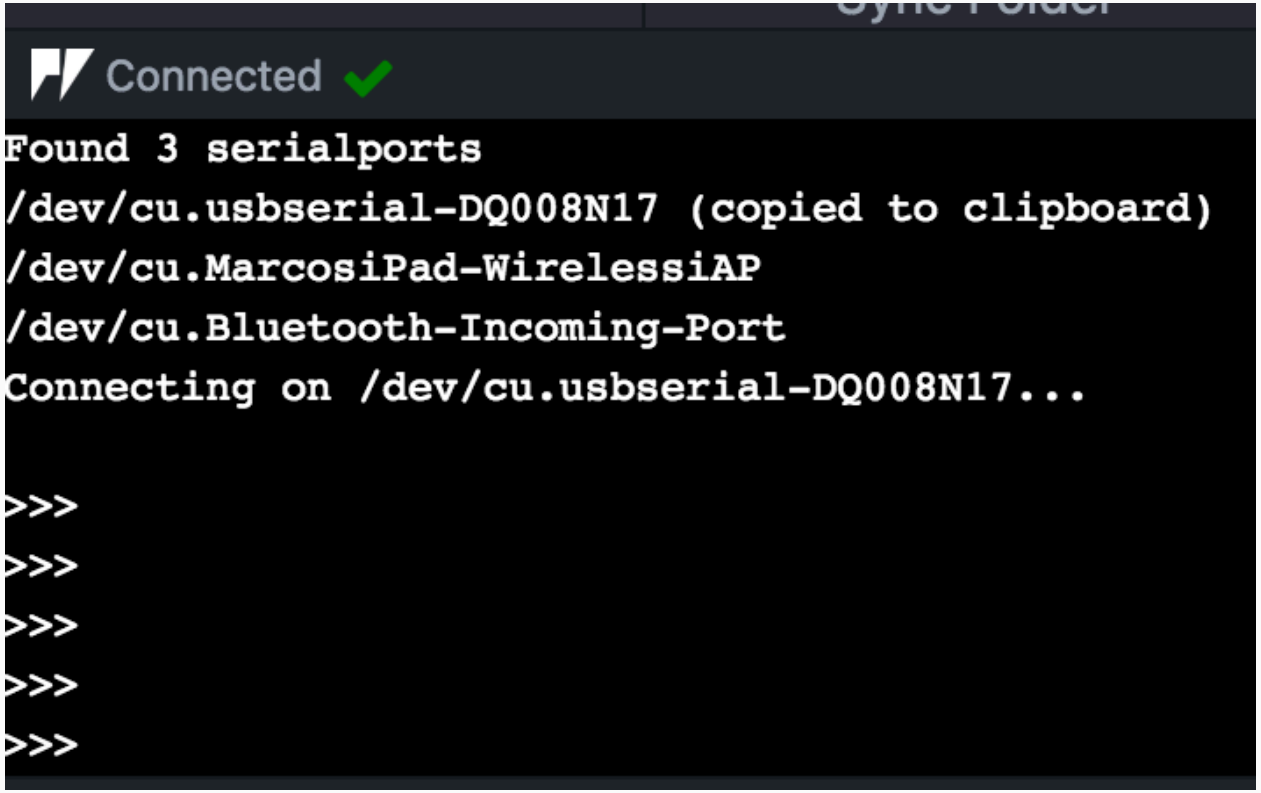


Make sure the LED and the microUSB are on the same side!

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:

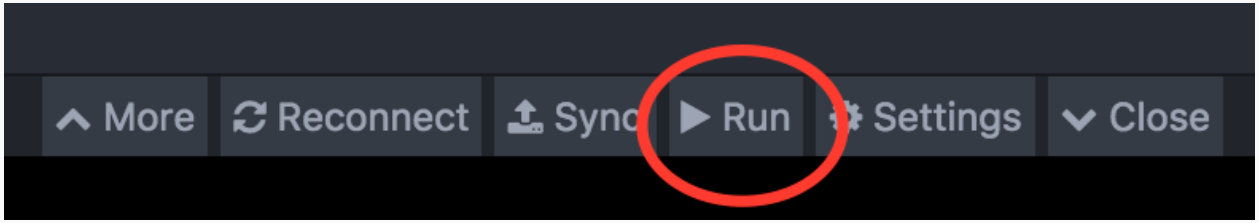
print("Hi! I am a python shell!")

executing code

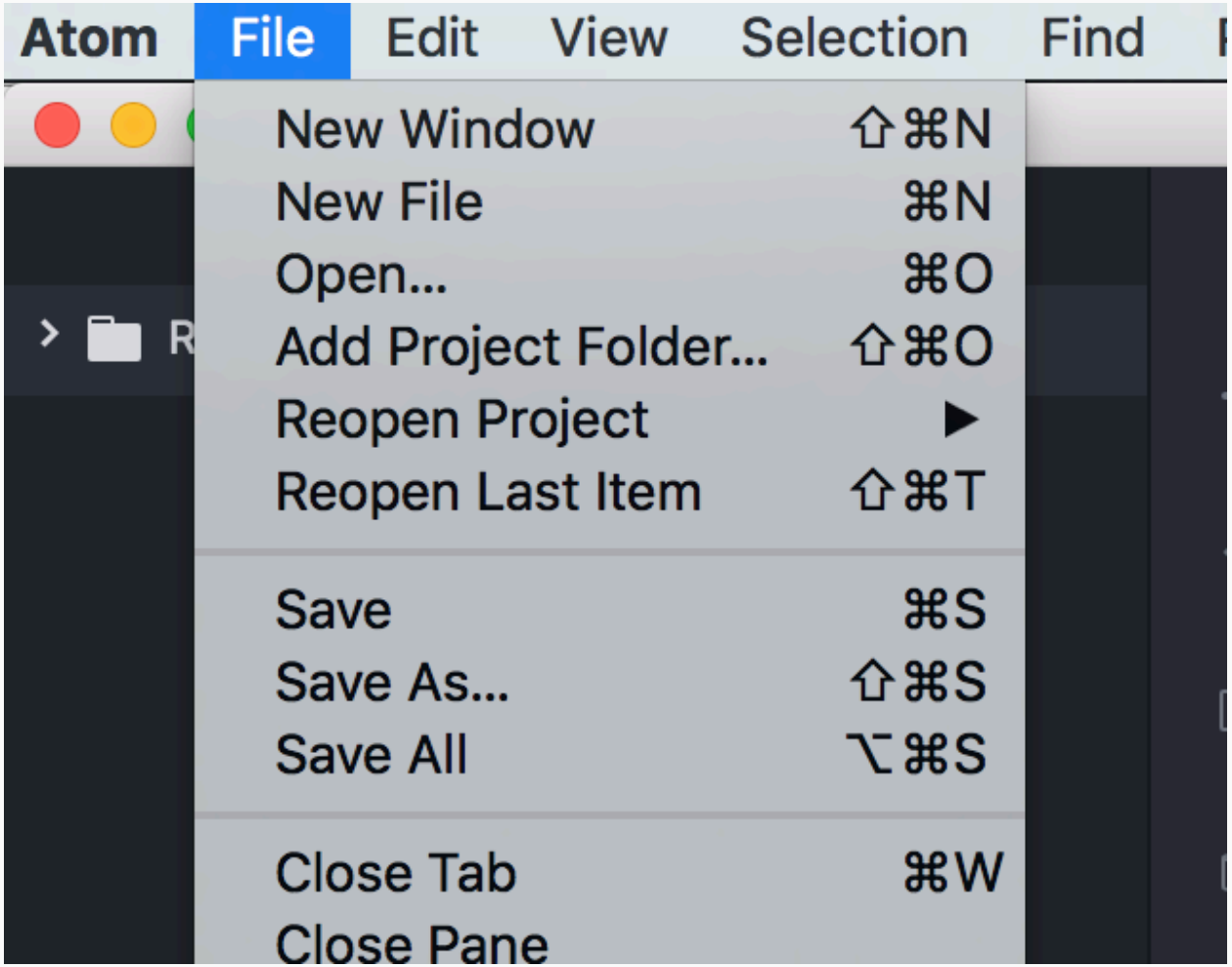
There are three ways to execute code on a Pycom device:

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. In this way, the code is stored in the Pycom device and will be executed again if you reboot the device.

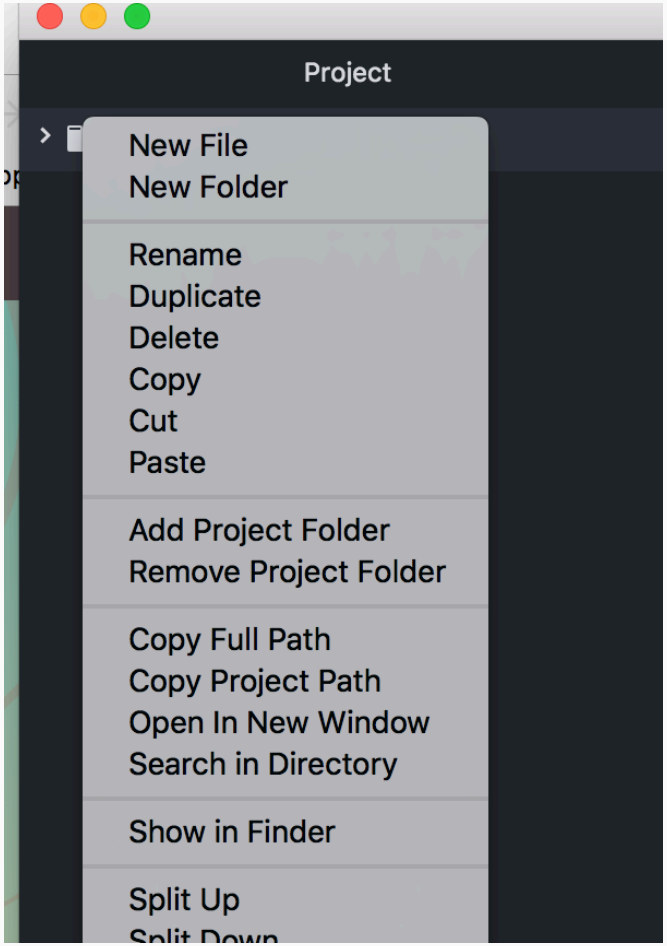
workflow: Run



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.

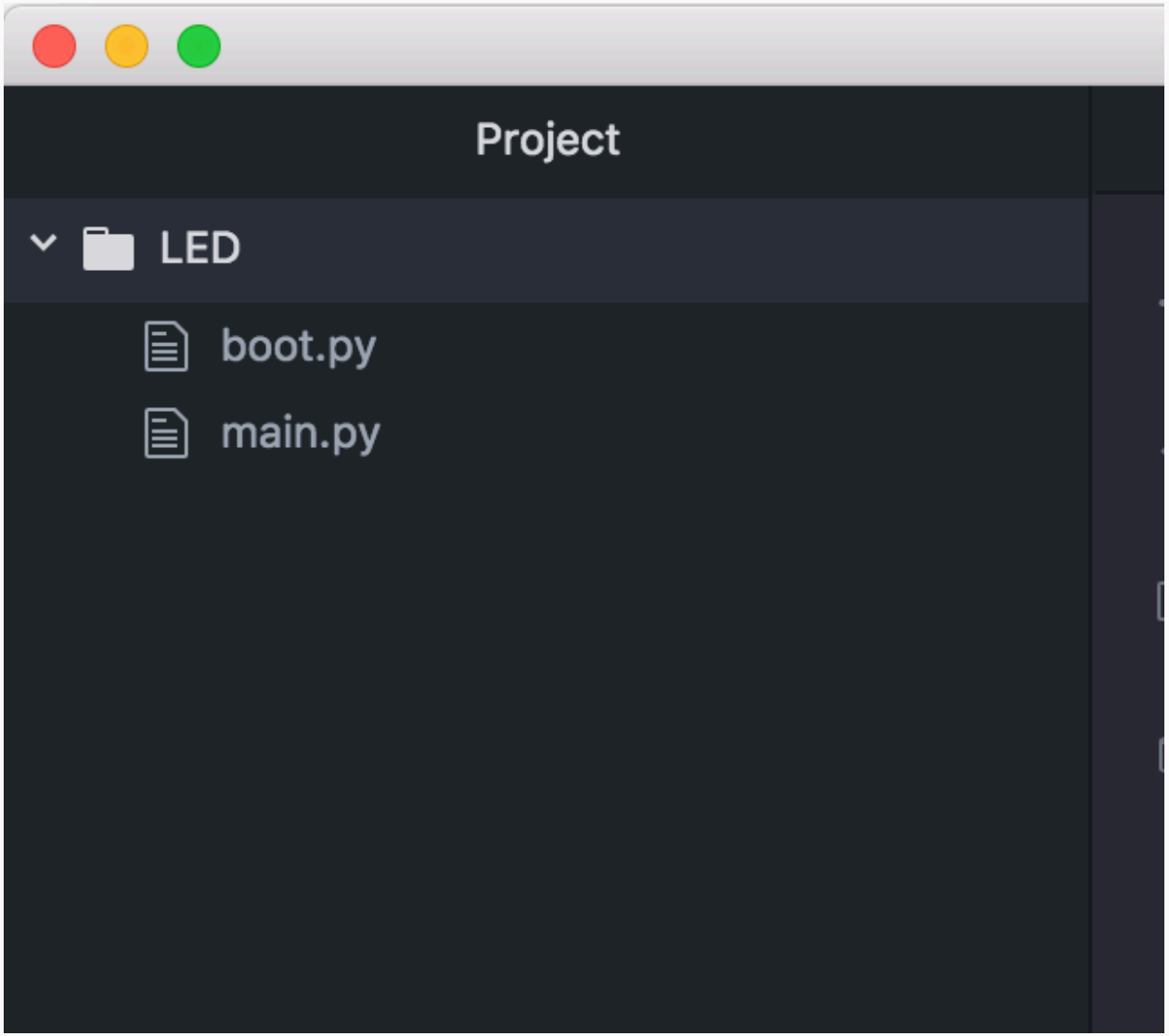
workflow: Project folder

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

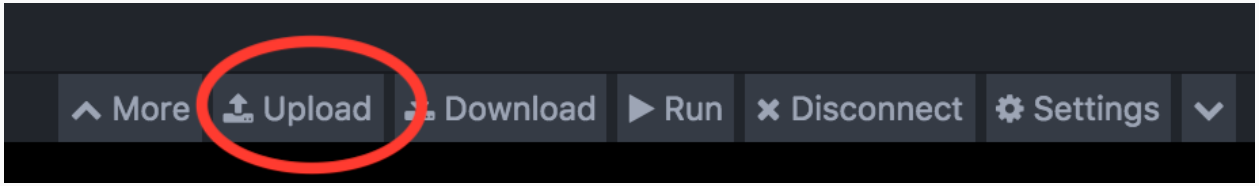
You should always have two files: **boot.py** (executed at boot time) and **main.py** (containing the main code).

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

workflow: example of Project folder



workflow: upload Project

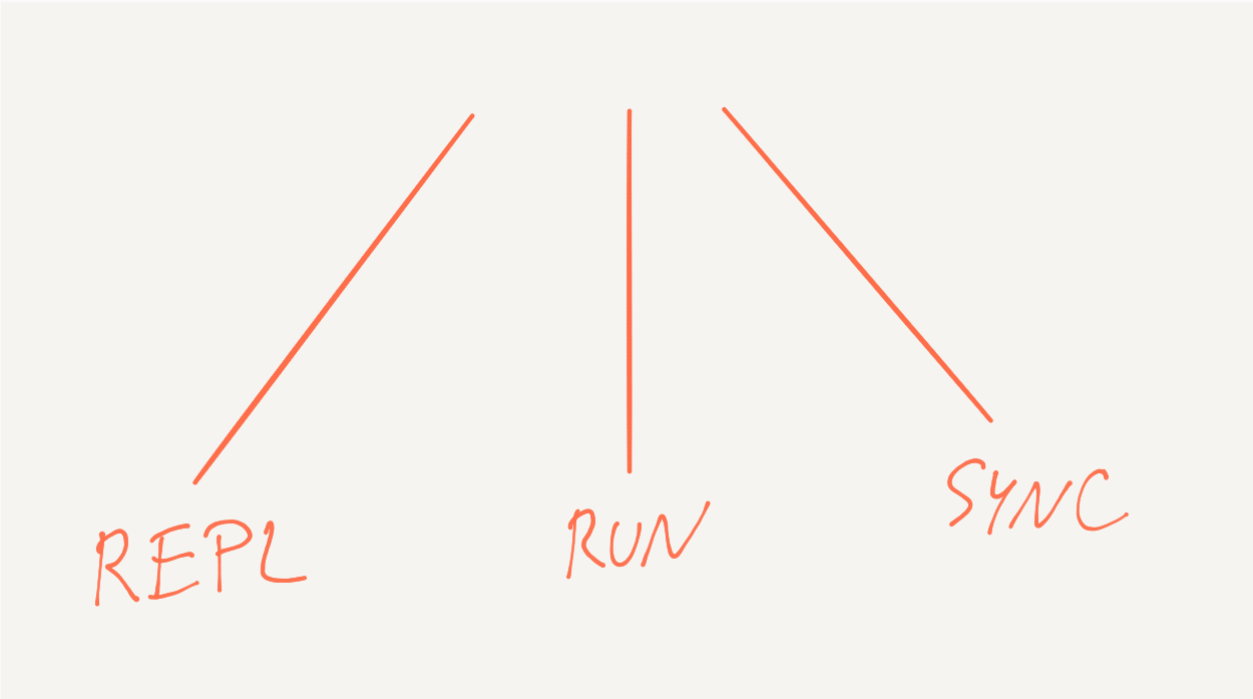


workflow: boot.py

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)
```

workflow: summary



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

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

Exercises:

1) 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.

2) Try to change the color of the LED gradually (from yellow to red, for example).

Writing data on Flash memory

Flash

In this example, we will learn how to:

1. access and operate the device file system;
2. create and write a file in the /flash folder;

Folder structure

Connect to a Lopy via the Atom console and import the basic operating system module (os): `import os`.

Once imported:

to know you current working directory: `os.getcwd()` (most probably the /flash folder);

to list folders and files in your current working directory: `os.listdir()`;

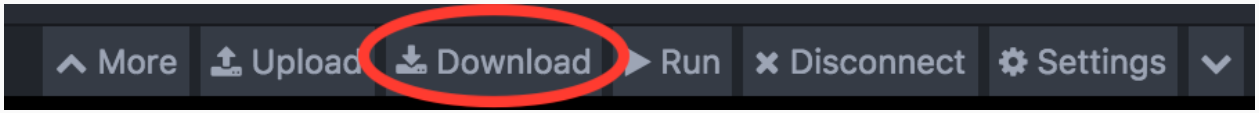
to create a new folder/directory named "log": `os.mkdir('log')`;

Writing and reading

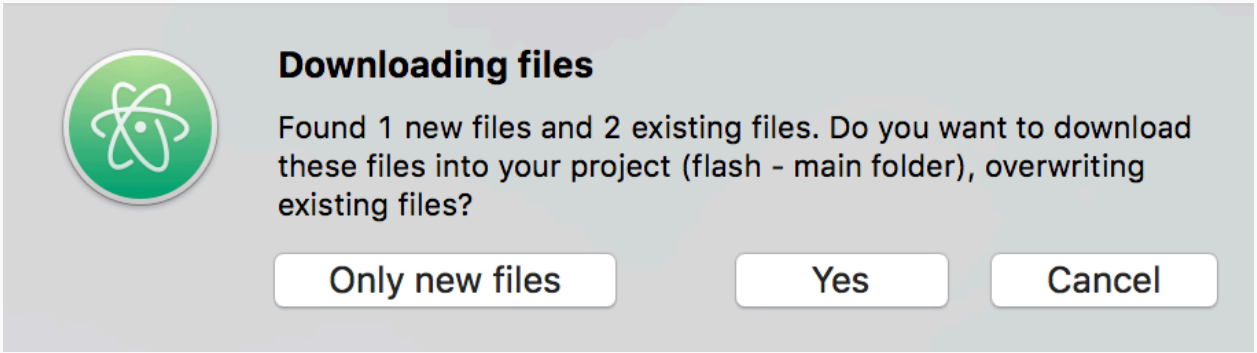
In the simplest case, to create and write a new file:

```
os.listdir('/flash')

# create/open, write, close a file
f = open('log/my\_first\_file.log', 'w')
f.write('Testing␣write␣operations␣in␣a␣file.')
f.close()

# open, read, close an existing file
f = open('log/my\_first\_file.log', 'r')
f.readall()
f.close()
```

workflow: download file from flash



workflow: download file from flash



exercise: flash

Write a code that creates a file named "log.csv" in /flash/log/.

In this file you will:

write "start", write a string for ten times, write "finish" and repeat this for five times.

PySense

PySense high-level modules

In this lab, we will provide a series of examples:

- accelerometer in src/pysense/acceloremeter

- measuring ambient light in src/pysense/ambient-light

- measuring temperature and atmospheric pressure in src/pysense/temp-bar

- measuring temperature and humidity in src/pysense/temp-hum

Pycom provides a library abstracting the implementation details of sensor chips. This library is already included in labs source code under the lib folder of each example.

Exercises

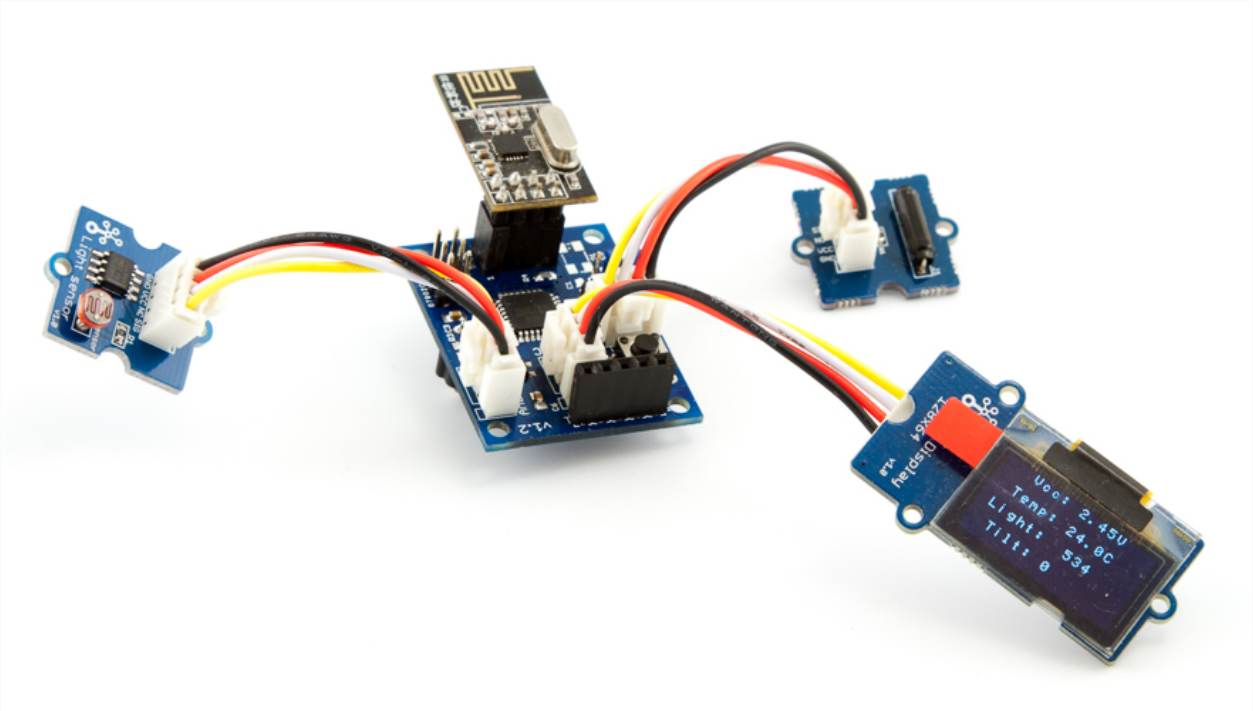
- Change the color of the LED based on accelerometer measurements (green, orange, red if the values of acceleration are small, medium or large)
- Find where is the temperature sensor and where is the light sensor
- Log the measurements of temperature every 10 seconds and the measurements of humidity every 30 seconds into the /flash/log folder (while LED blinking green)

ICTP-IAEA Grove shield

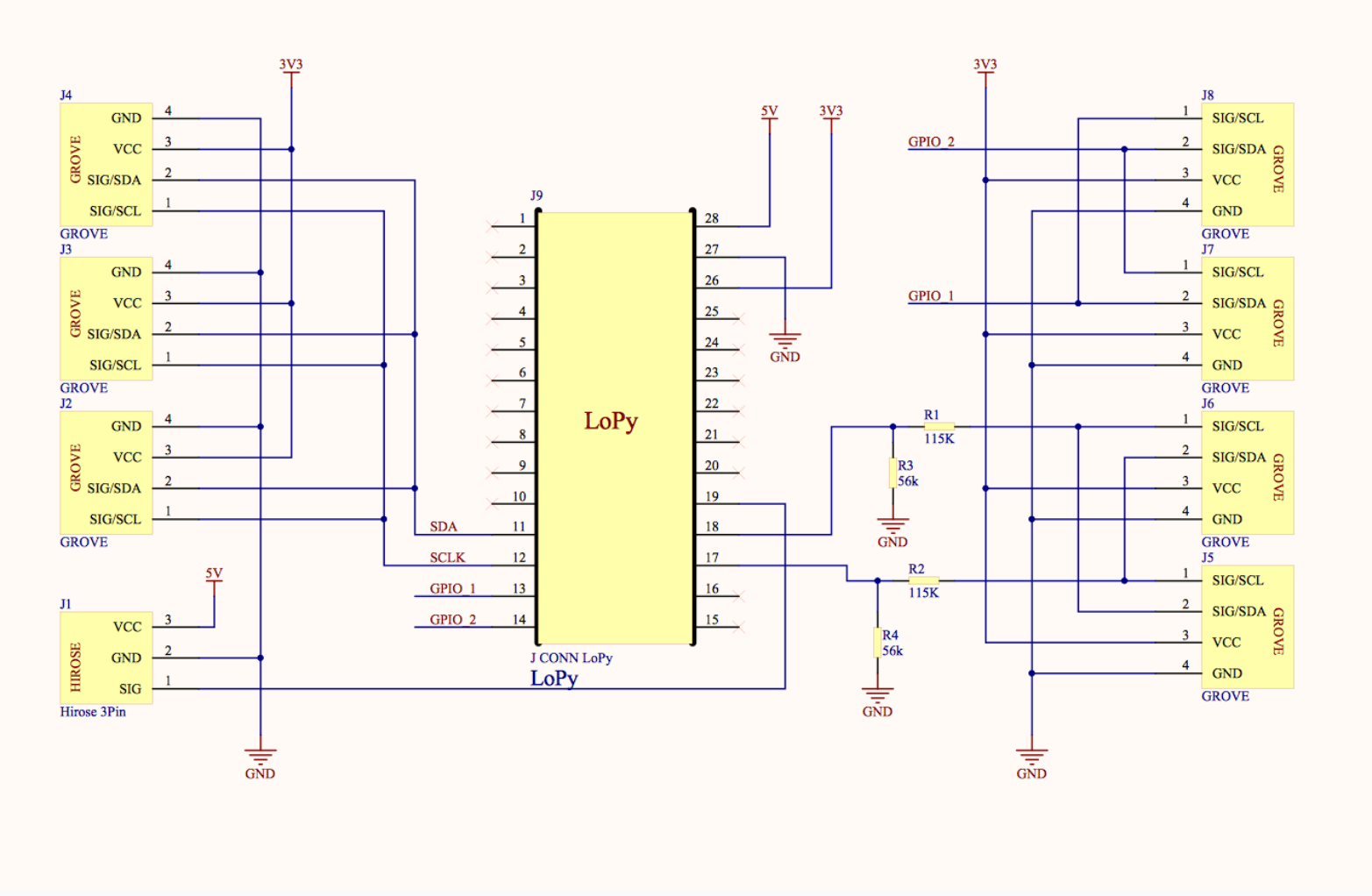
Grove sensors

www.seeedstudio.com/category/Sensor-for-Grove-c-24.html

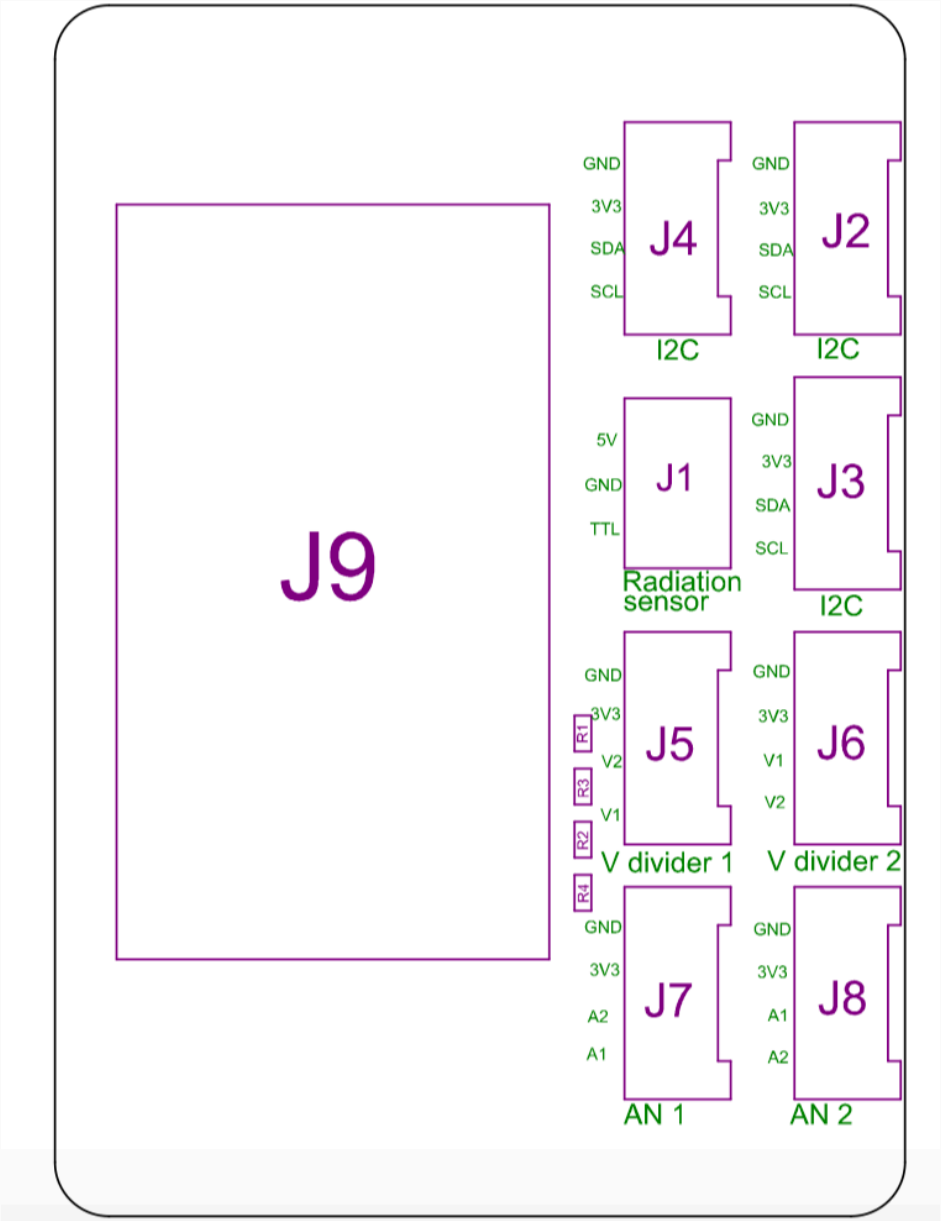
Grove sensors



ICTP-IAEA Grove shield



ICTP-IAEA Grove shield



ICTP-IAEA Grove shield

J4: I2C

J2: I2C

J1: radiation sensor

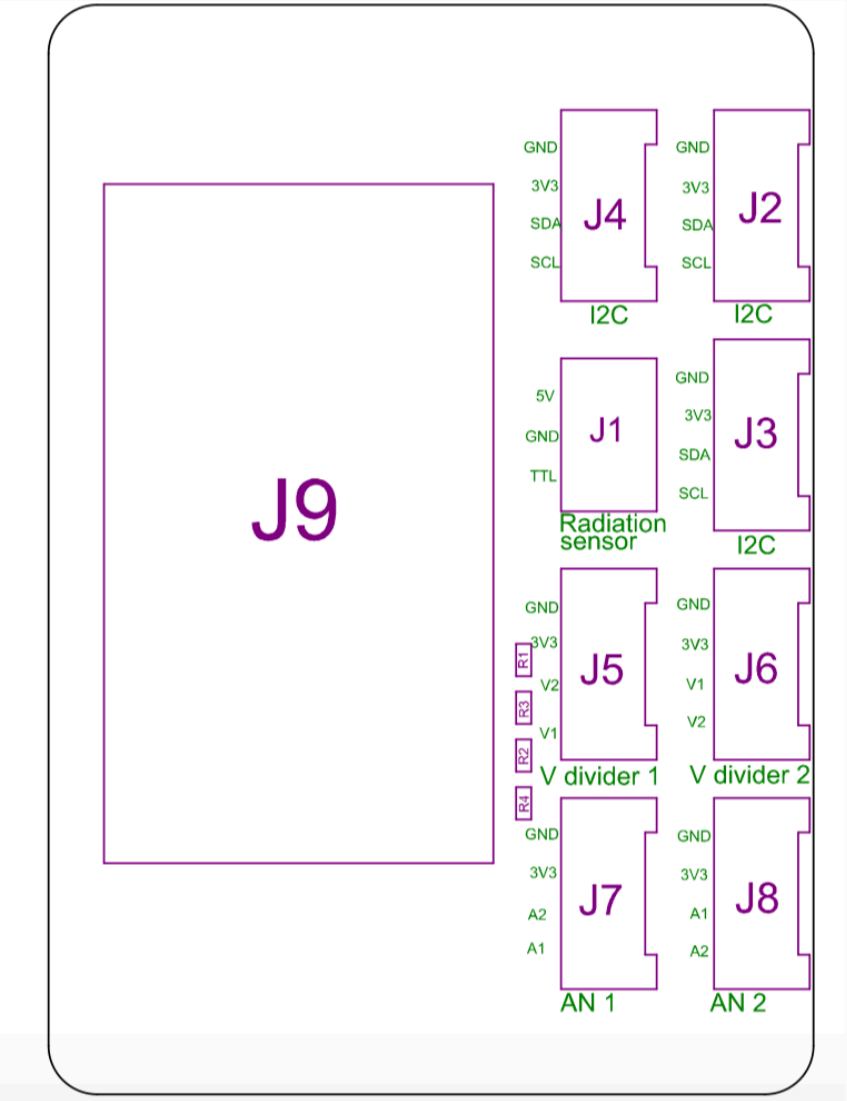
J3: I2C

J5: analog

J6: analog

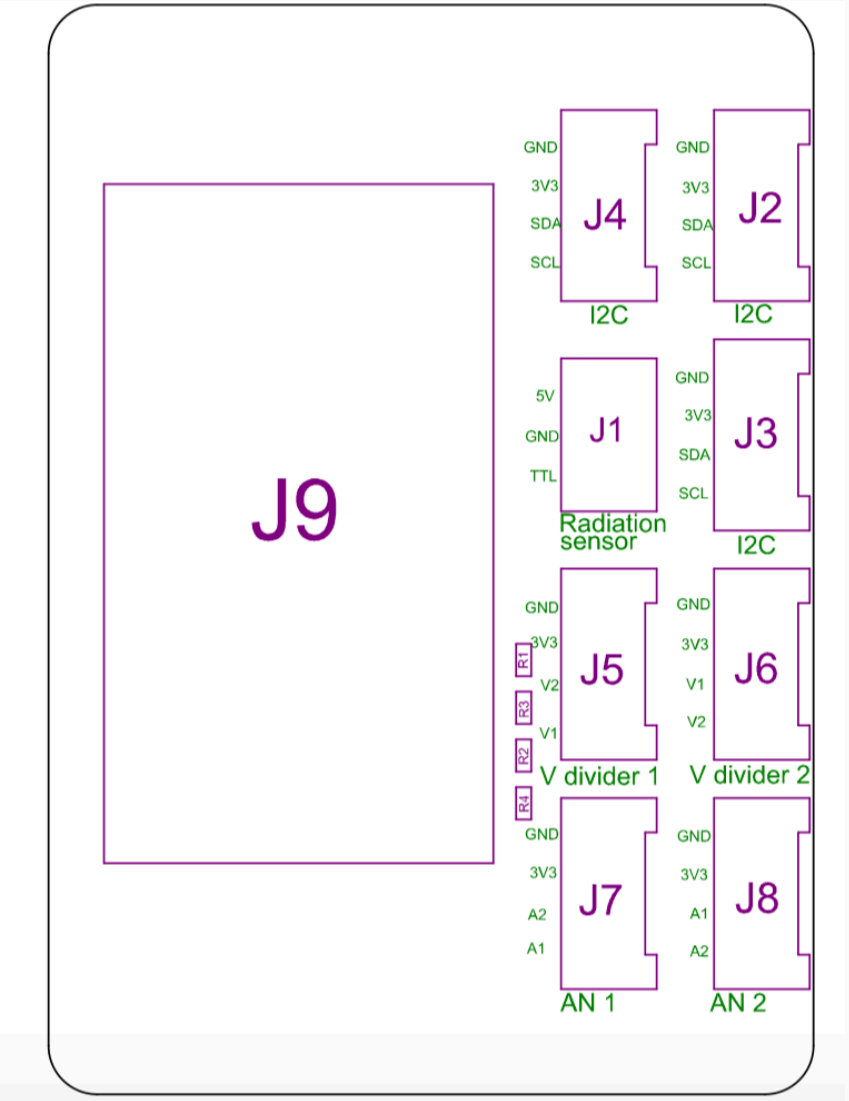
J7: digital

J8: digital

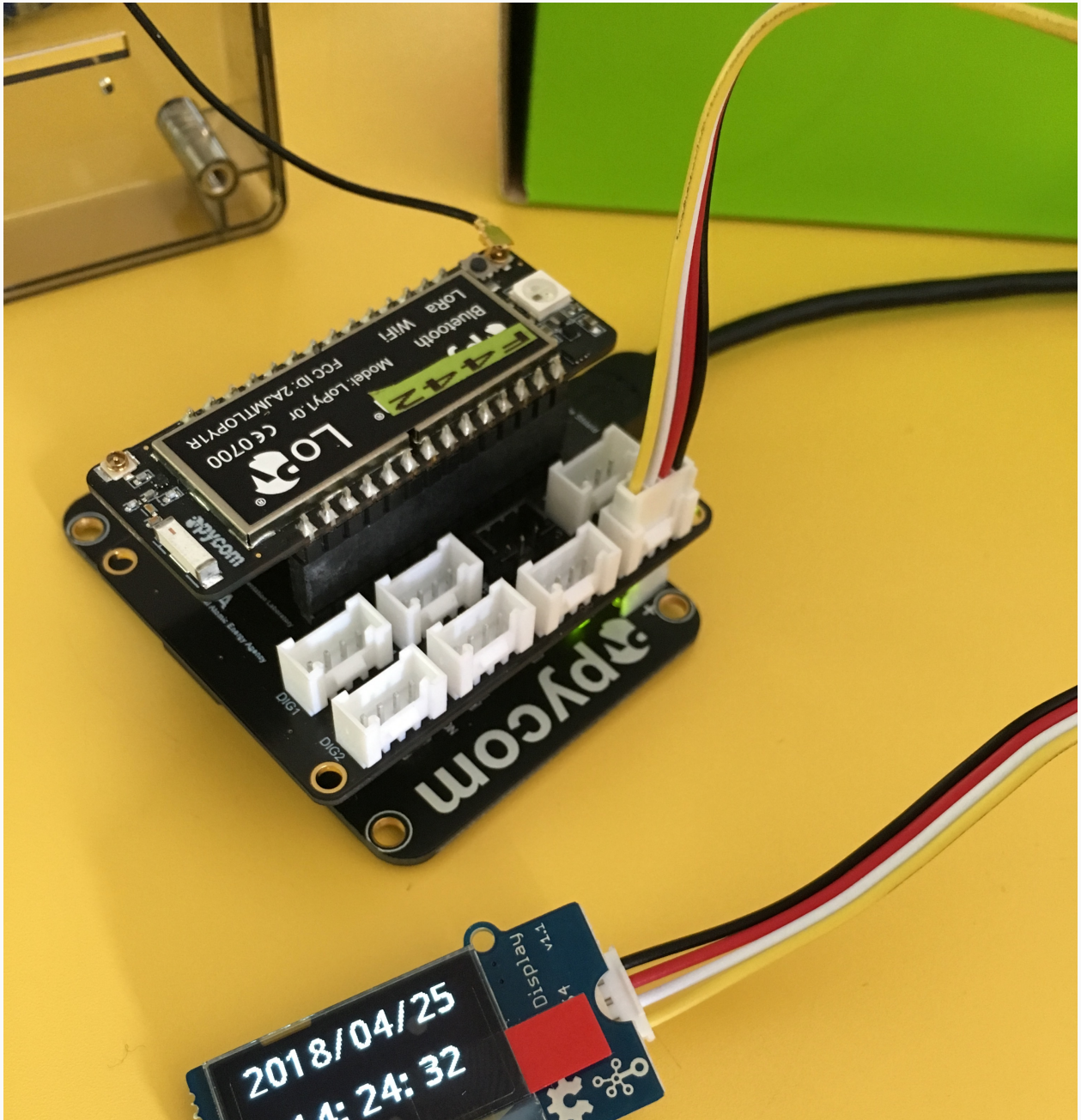


ICTP-IAEA Grove shield

- J4: Sda on Pin11, Scl on Pin12
- J2: Sda on Pin11, Scl on Pin12
- J1: Pin17
- J3: Sda on Pin11, Scl on Pin12
- J5: P16 on Pin18
- J6: P15 on Pin17
- J7: P12 on Pin14
- J8: P11 on Pin13



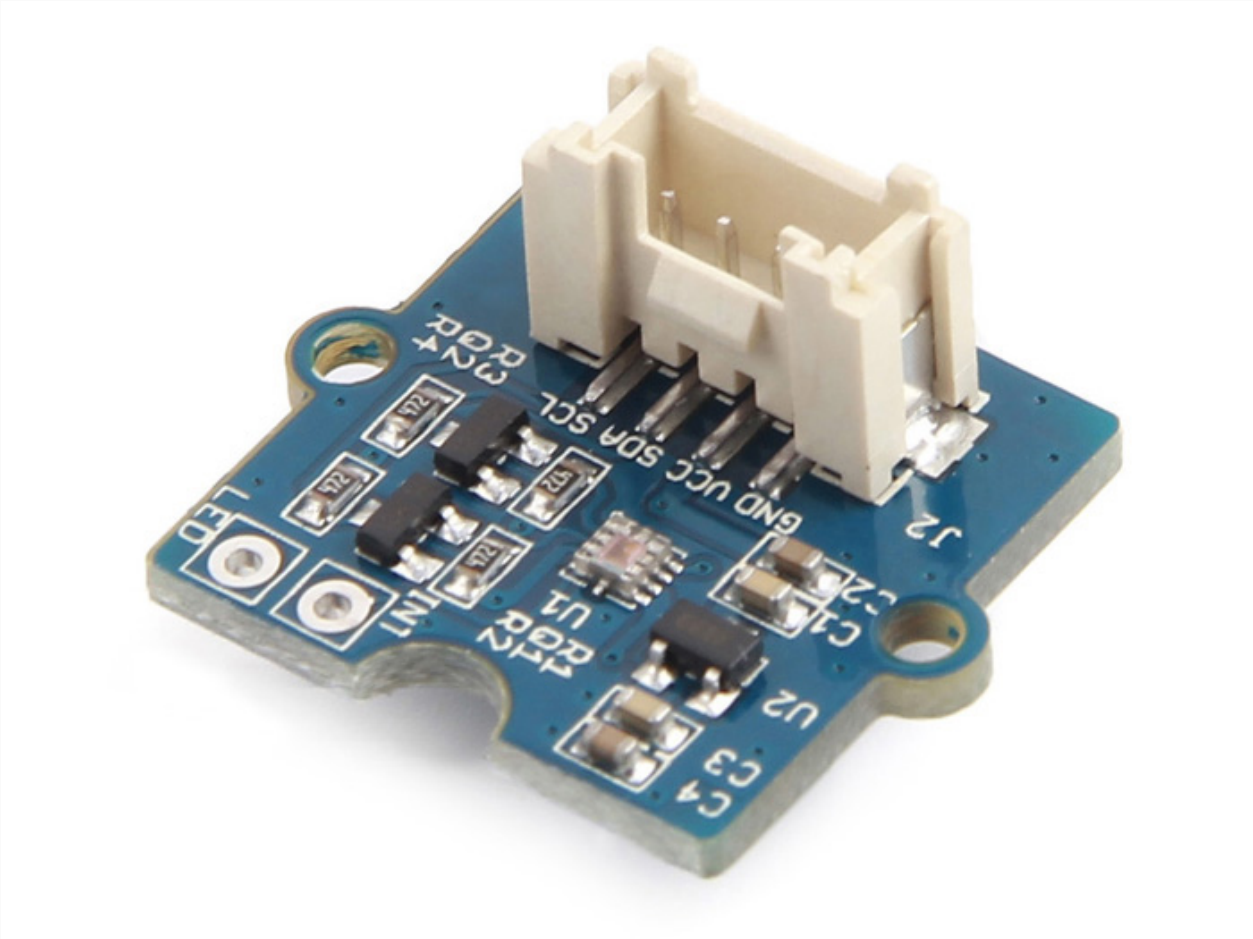
ICTP-IAEA Grove shield



Grove - OLED Display 0.96”: I2C sensor

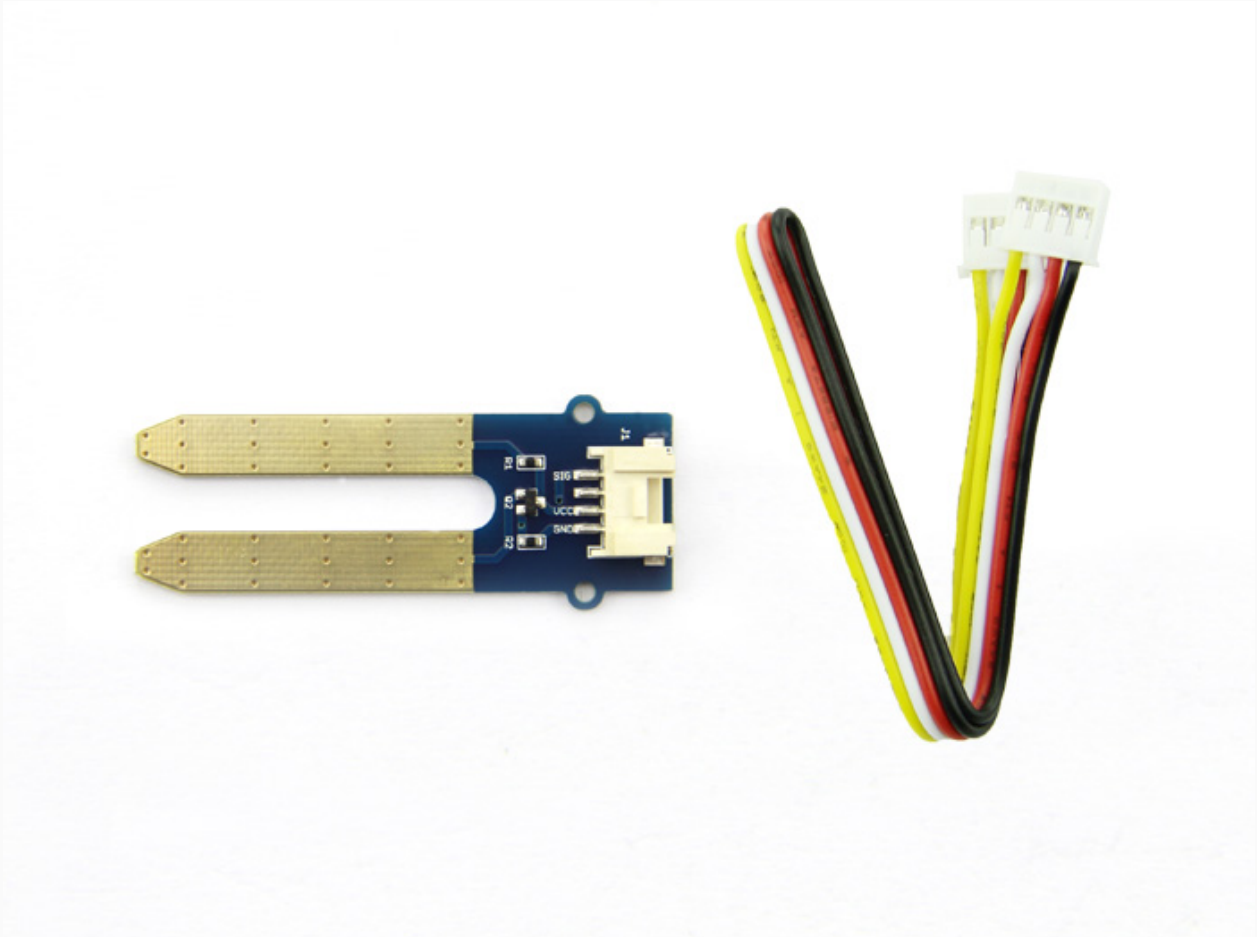
https://www.seeedstudio.com/Grove-OLED-Display-0.96%26quot%3B-p-781.html

Grove - Sunlight Sensor: I2C sensor



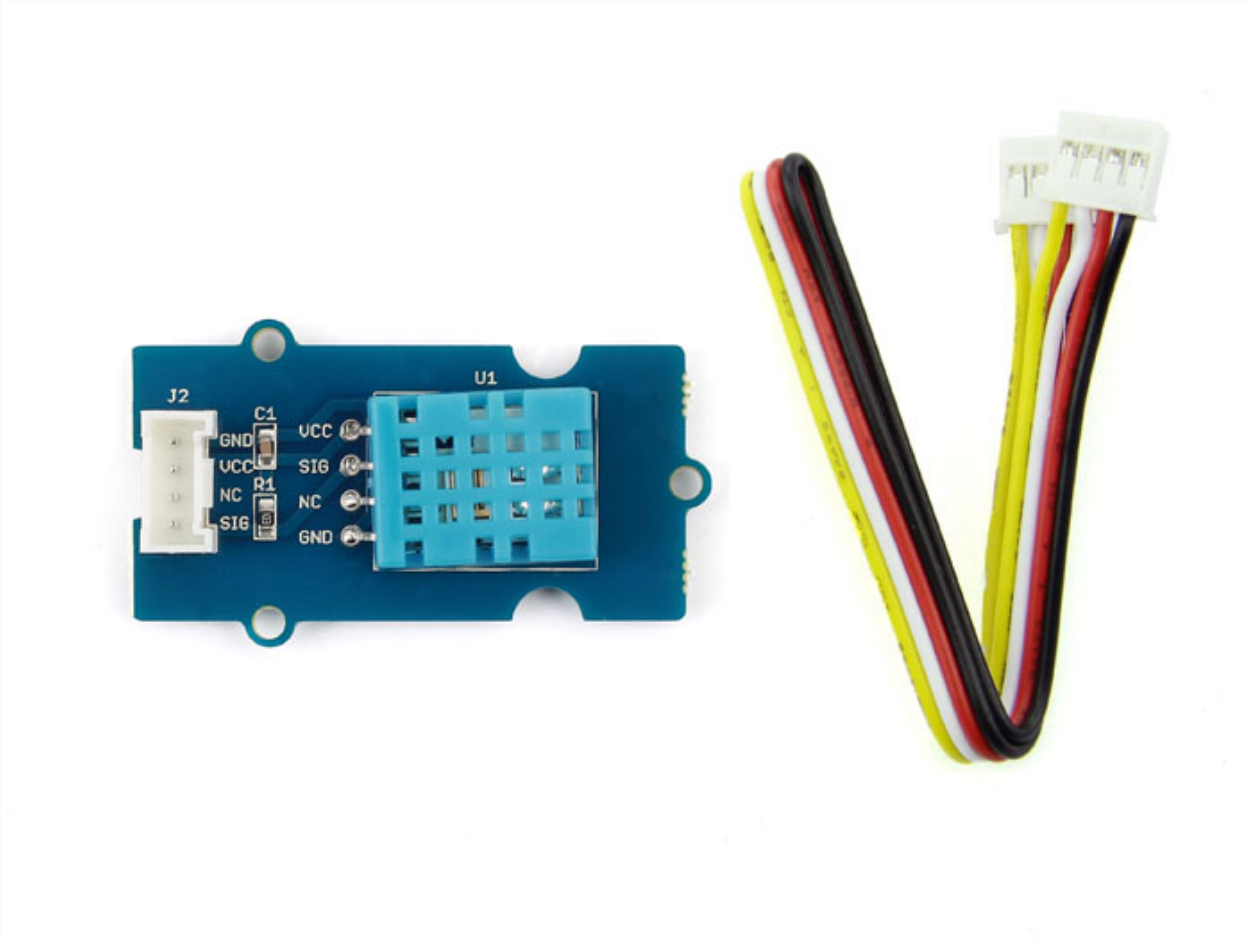
https://www.seeedstudio.com/Grove-Sunlight-Sensor-p-2530.html

Grove - Moisture Sensor: Analog sensor



https://www.seeedstudio.com/Grove-Moisture-Sensor-p-955.html

Grove - T and H Sensor DHT11: Digital sensor

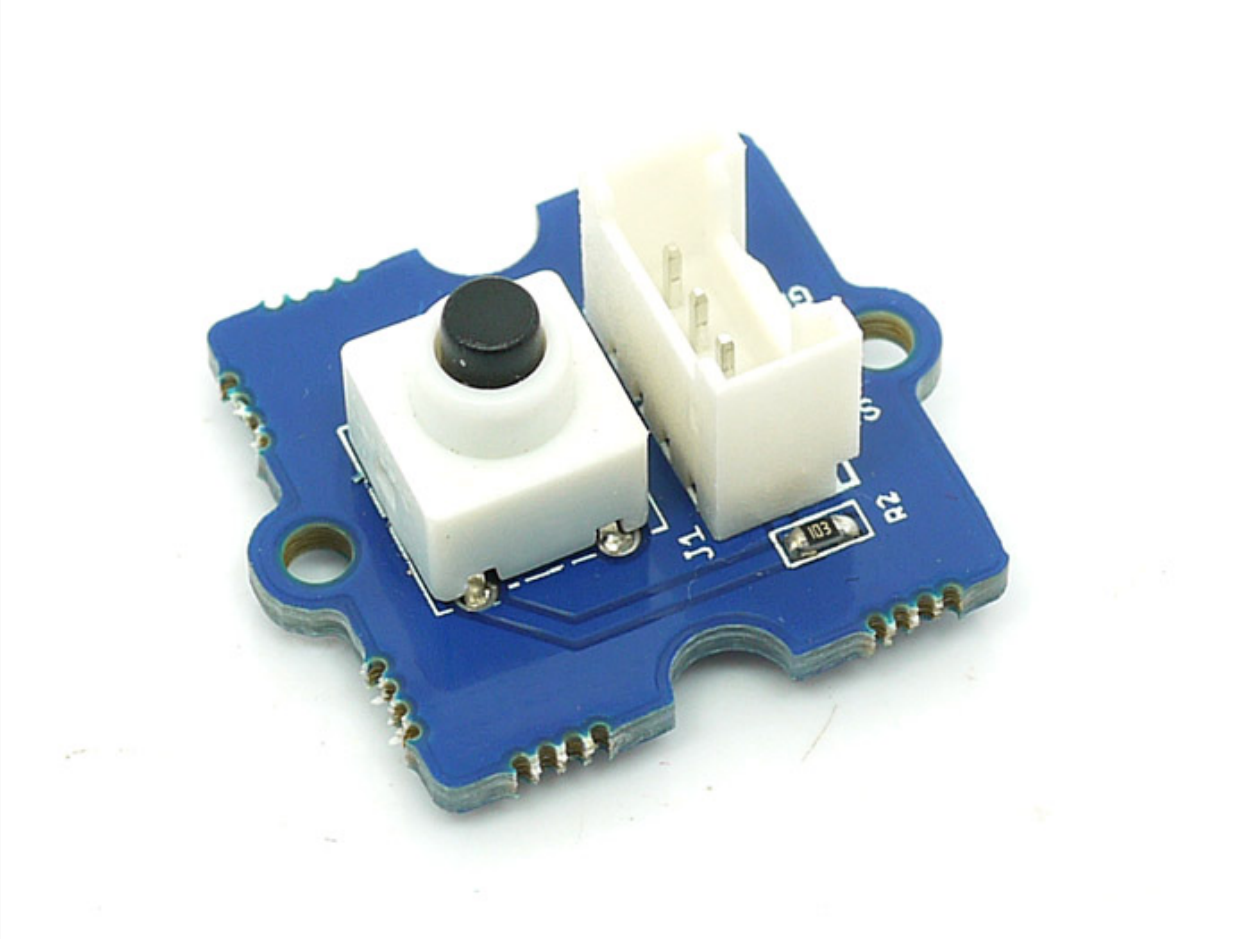


https://www.seeedstudio.com/Grove-Temperature-%26amp%3B-Humidity-Sensor-%EF%BC%88DHT11%EF%BC%89-p-745.html

Grove Buzzer: Digital Sensor

https://www.seeedstudio.com/Grove-Buzzer-p-768.html

Grove Button: Digital sensor



https://www.seeedstudio.com/Grove-Button-p-766.html

Exercises

- Check all the examples (in /grove_board)
- Show temperature and humidity on the display for 5 seconds, then light for 5 seconds, then temperature and humidity again, and so on.
- Design a sensor node for agriculture. Measure temperature, humidity, light and soil moisture. Save the data and time in the internal flash memory. Test your device outdoors!