Lab1: getting started with LoPy

Marco Zennaro, PhD
ICTP

ICTP

Labs

⅓   Ready to use, tested examples

⅓   Exercise based on the examples

⅓   Your imagination → create new applications

Lab alert

The number of variables in the lab settings is huge (computer operating system, firewall, device firmware version, code version, network, etc)

Things will go wrong :-)

Be patient, we will solve all issues!

Found a bug? Let me know! Feedback is welcome.

Our Lab equipment

LoPy

PySense

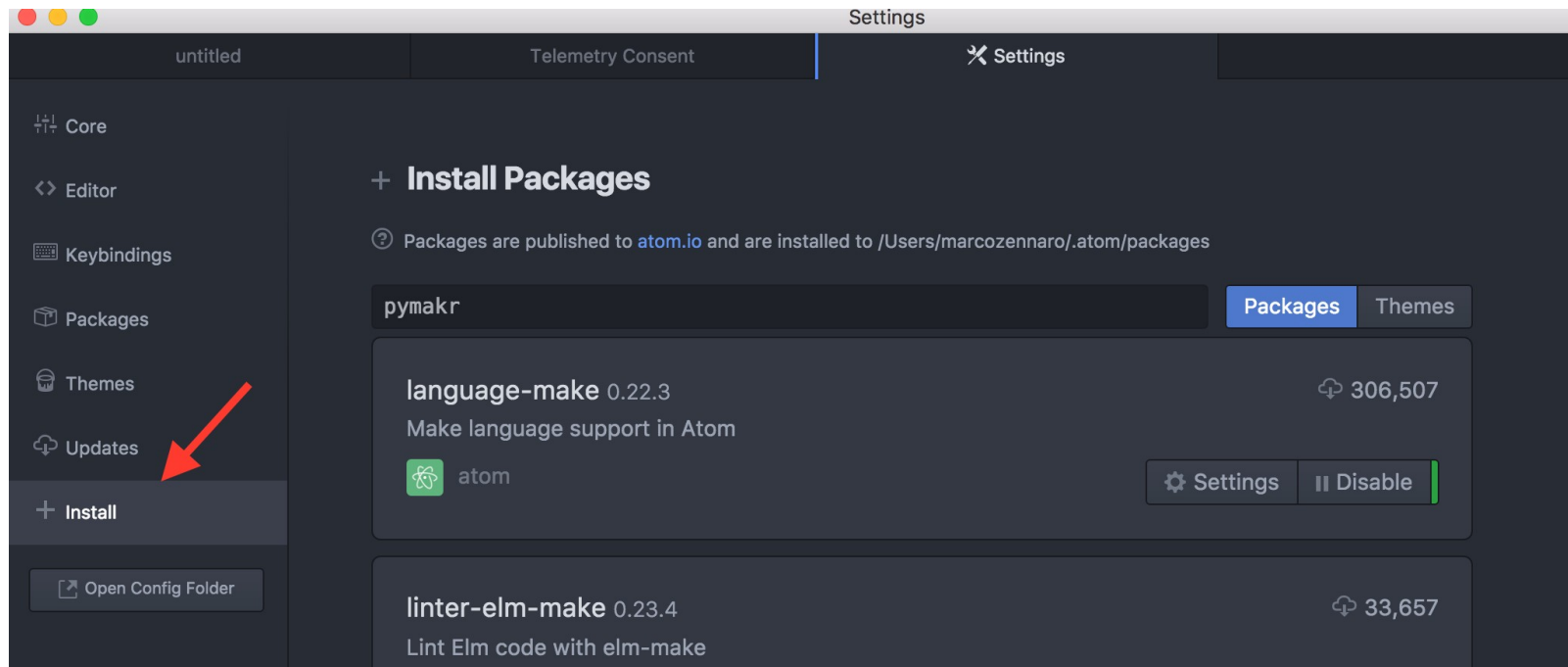Pycom LoPy 4

PySense

microUSB Cable

Workflow: Atom

Please install from www.atom.io or download the installer from our local server.

If you have it already, upgrade to the latest version!

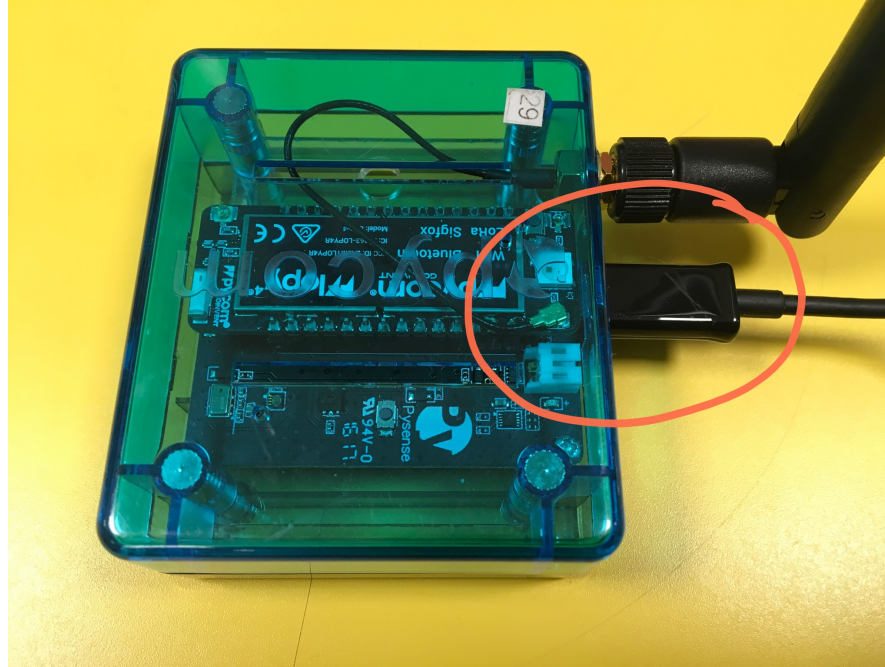# Workflow: install the **pymakr** package

Atom > Preferences > Install
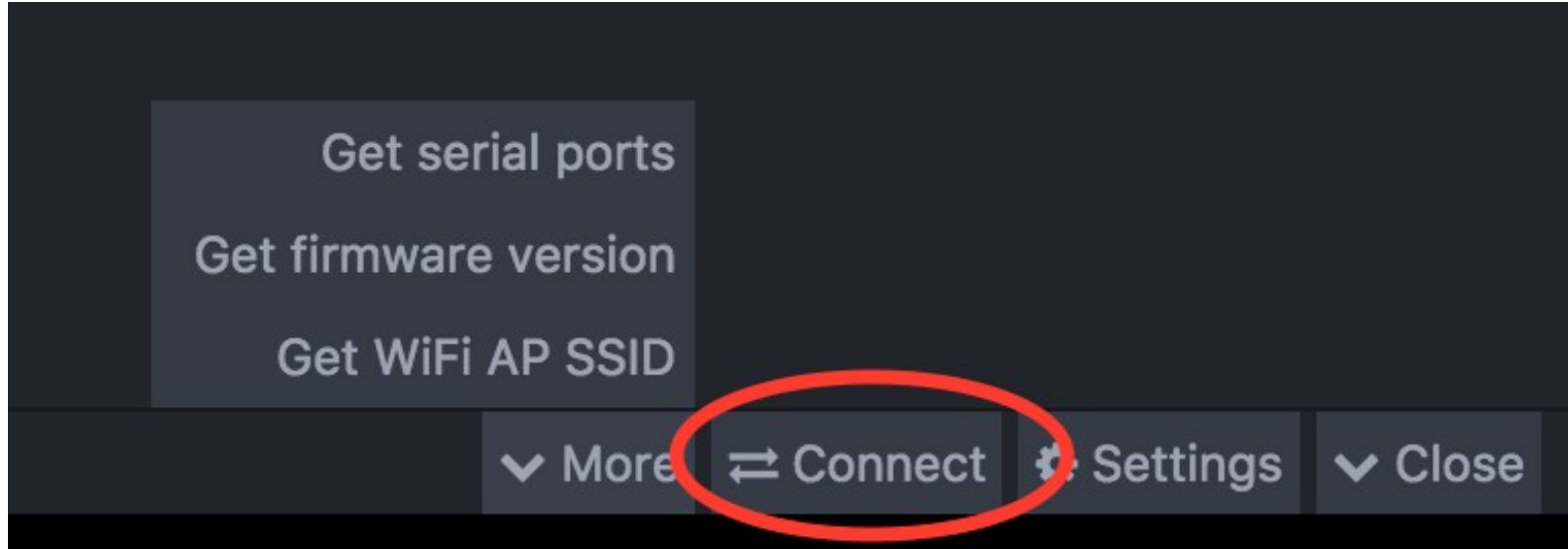Different on Windows, Mac, Linux
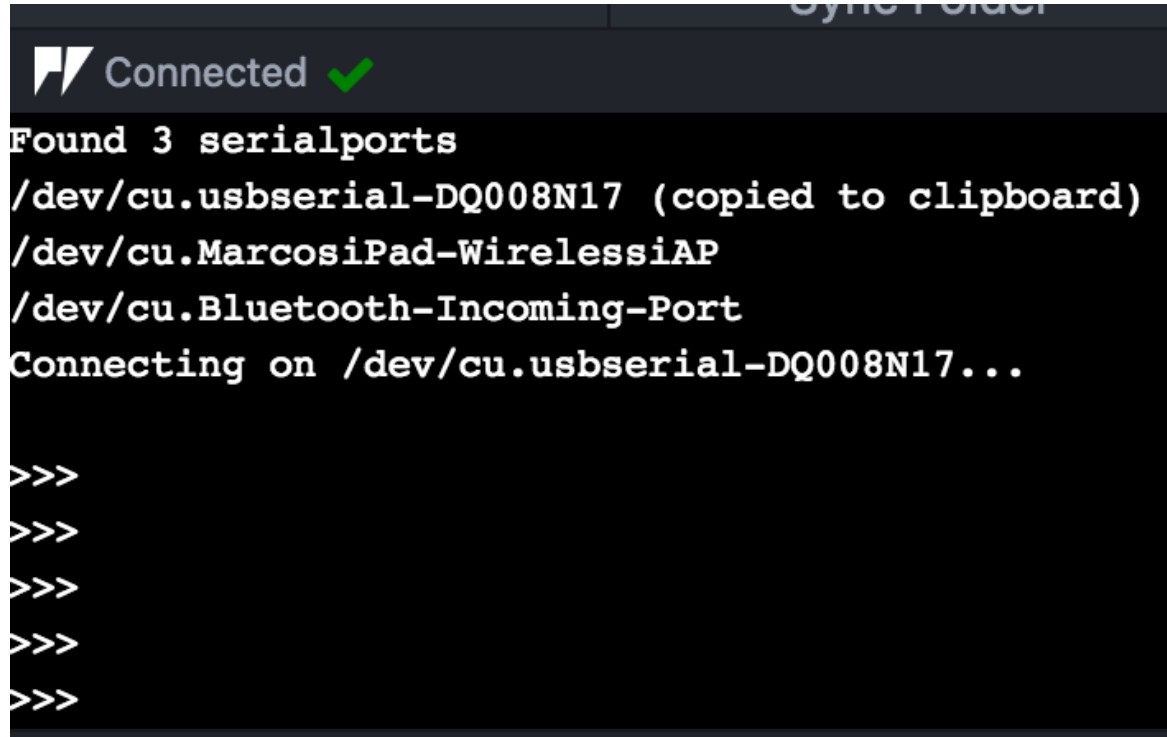
Workflow: connect board via USB

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

Workflow: connect!

Click on 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.

REPL example

The os module can be used for further control over the filesystem. First import the module:

```
>>> import os
```

Then try listing the contents of the filesystem:

```
>>> os.listdir()
```

REPL example

You can make directories:

>>> os.mkdir('dir_iotlab')

And remove entries:

>>> os.remove('data.txt')

REPL example

If a device's filesystem gets corrupted, you can format it by running:

```
>>> import os
```

```
>>> os.mkfs('/flash')
```

Executing code

2) Using the **Run** button. Code that is open on the Atom editor will be executed, but will **not be stored in the device**.

⚠️

If you reboot, the code will not be executed again.

Executing code
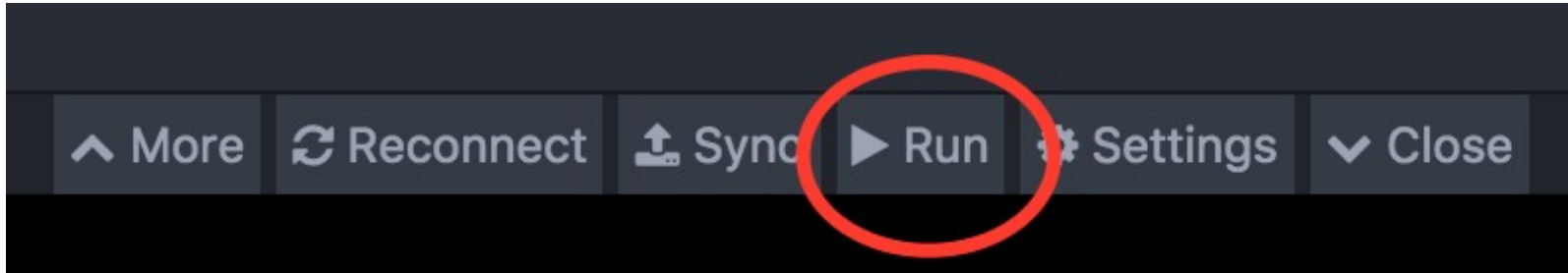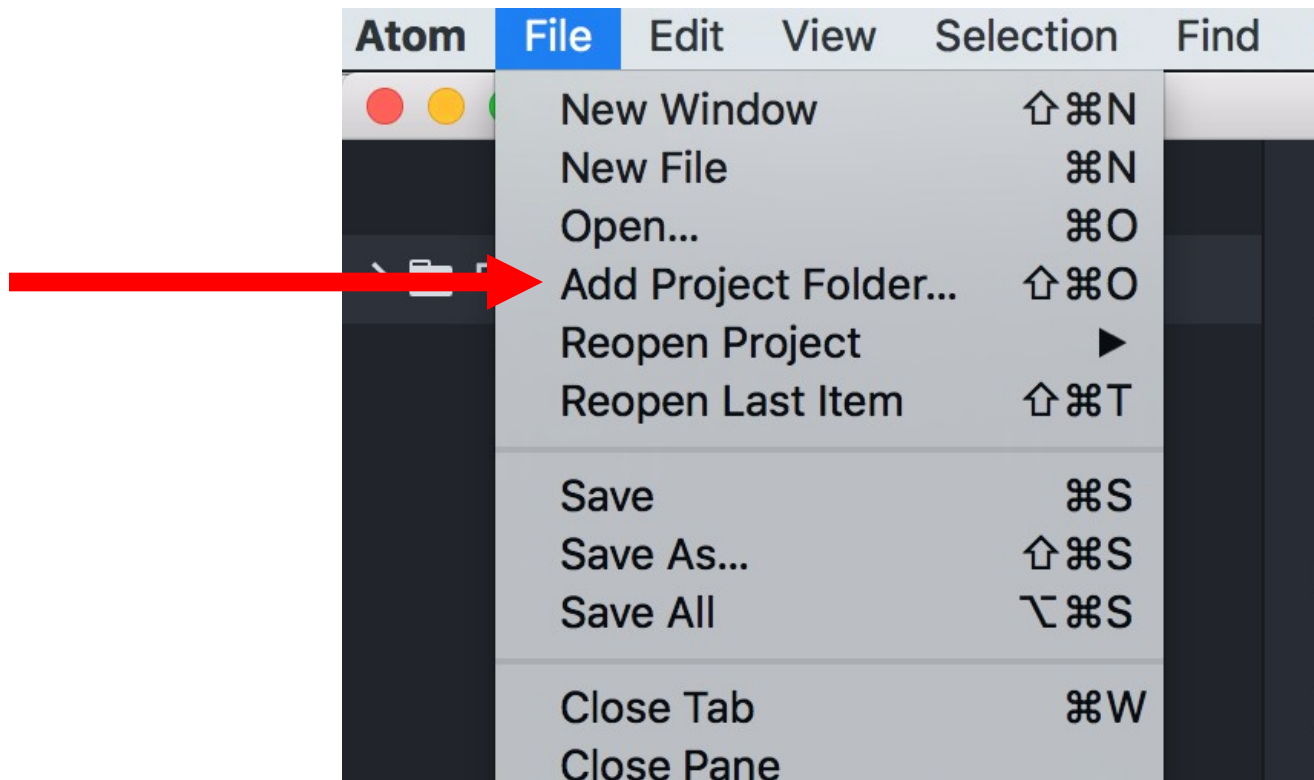
Executing code

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.

ICTP

Workflow: project folder

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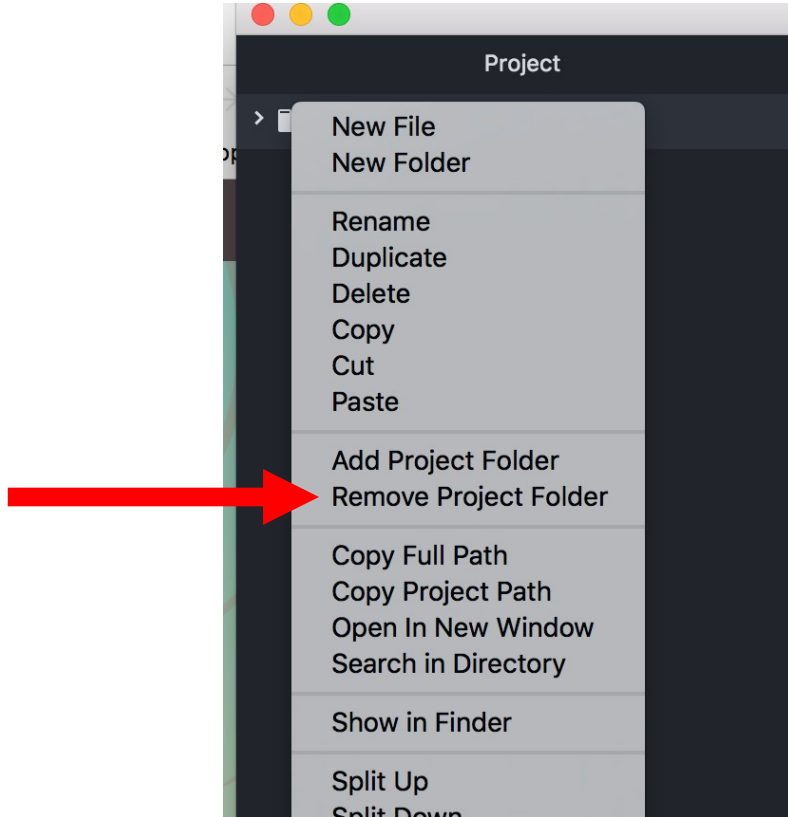
It is easier if you only have one Project folder. Make sure you Remove any other Project folders and **keep only the one you want to use.**

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 files.

ICTP

Workflow: example of project folder
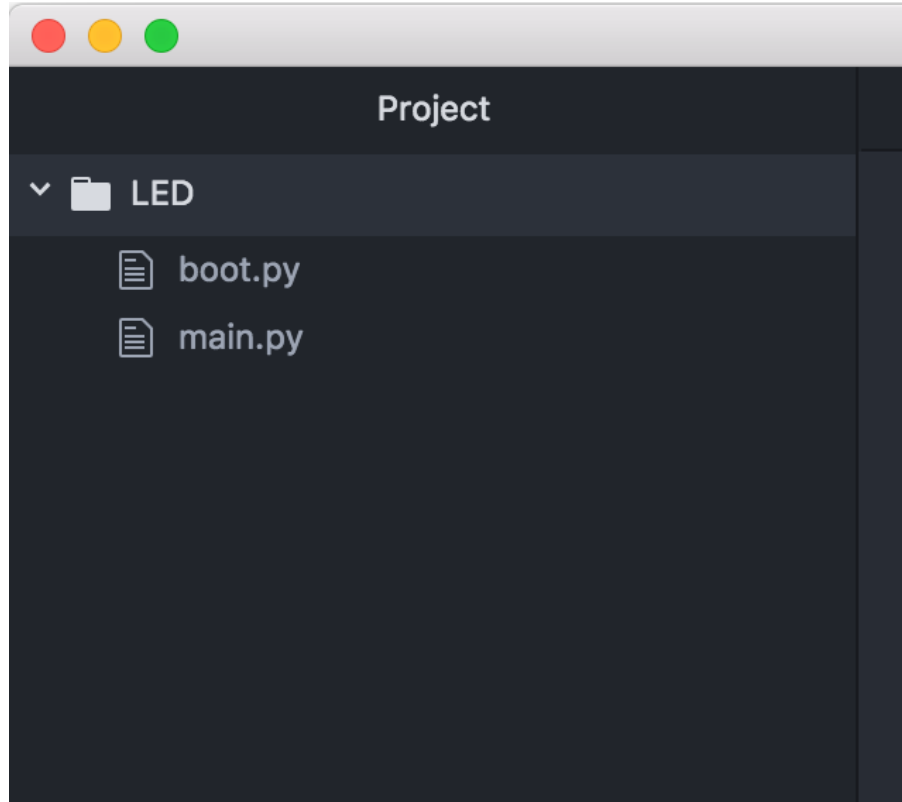
Workflow: upload project folder
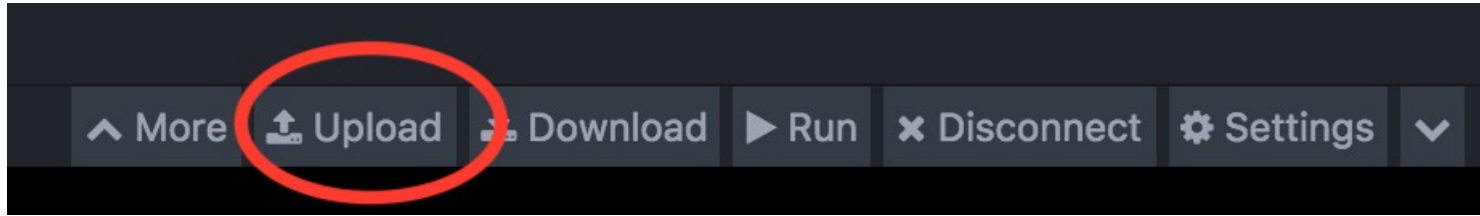
Workflow: useful Atom settings

Settings → Global Settings

☑ **Autoconnect on USB**
Ignores any 'device address' setting and automatically connects to the top item in the serialport list

☑ **Upload all file types**
If enabled, all files will be uploaded no matter the file type. The list of file types below will be ignored

Workflow: useful Atom settings

☑ Open on start
Automatically open the pymakr console and connect to the board after starting Atom

☑ Safe-boot before upload
[Only works with firmware v1.16.0.b1 and up.] Safe boots the board before uploading to prevent running out of memory while uploading. Especially useful on older boards with less memory, but adds about 2 seconds to the upload procedure

☑ Reboot after upload
Reboots your pycom board after any upload or download action

Workflow: boot.py

The boot.py file should always start with following code, so we can run our Python scripts over a serial connection (USB).

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

Resetting

They are different ways to reset your device. Pycom devices support both **soft and hard resets**.

A soft reset clears the state of the MicroPython virtual machine but leaves hardware peripherals unaffected. To do a **soft reset**:

press Ctrl+D on the REPL

Resetting

A **hard reset** is the same as performing a power cycle to the device. In order to hard reset the device, press the reset switch or run:

```
>>> import machine

>>> machine.reset()
```
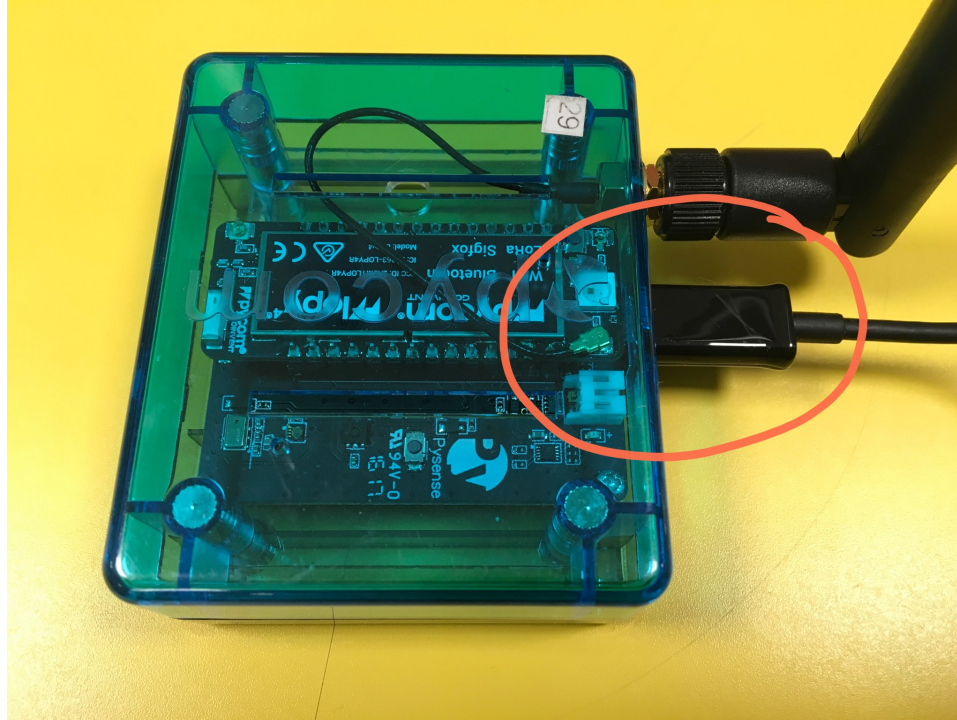
LED

LED

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

LoPys don't have screens!

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

LED

LED

Check the **LED folder** and sync the two files to your active project folder.
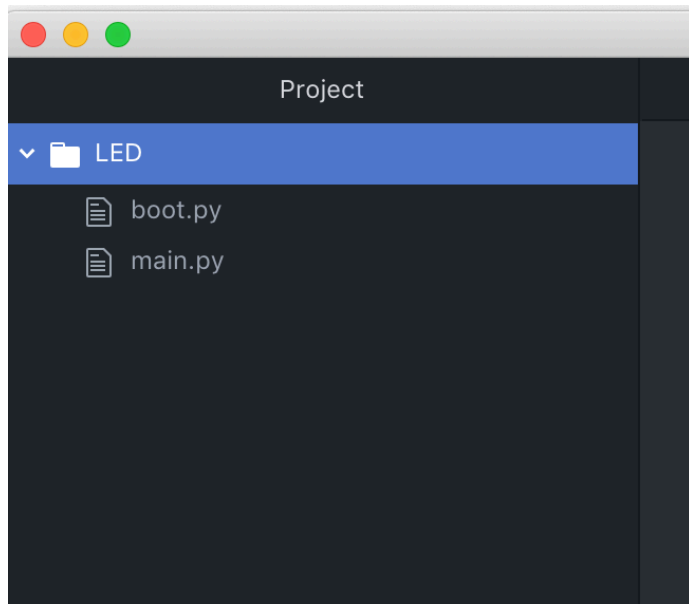
LED example, part 1

import pycom   Import the python libraries needed

import time

pycom.heartbeat(False)   Set OFF the heartbeat (the LED turns blue to show that the device is alive) as we will change the color of the LED

ICTP

LED example, part 1

```
for cycles in range(10):                    Execute 10 times

    pycom.rgbled(0x007f00) # green          Change LED color to green, yellow and red

    time.sleep(5)                           Sleep for 5 seconds each time

    pycom.rgbled(0x7f7f00) # yellow

    time.sleep(5)

  pycom.rgbled(0x7f0000) # red

    time.sleep(5)
```

LED: Exercises

1) Use the LED to simulate a traffic light (red, yellow and green lights)

2) How fast can the light blink? (modify the sleep time)

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

PySense

PySense: high-level modules

In this lab, we will provide a series of examples to use the sensors in the PySense module.

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

PySense: high-level modules

accelerometer in Code/pysense/acceloremeter

ambient light in Code/pysense/ambient-light

temperature and atmospheric pressure in Code/pysense/temp-bar

temperature and humidity in Code/pysense/temp-hum

PySense: Exercises

1) Change the color of the LED based on light measurements (no light → red, strong light → green)

2) Change the color of the LED based on the accelerometer measurement (slow acc → slow blink, fast acc→ fast blink)

PySense: Exercises

3) Change the color of the LED based on the temperature measurement (cooler → green, warmer → red)

4) Come up with a cool and useful application!

Summary

We introduced the Pycom workflow.

We learned how to change the color of the LED and read values from the PySense.

Feedback?

Email mzennaro@ictp.it