Lab1:
getting started with LoPy

Marco Zennaro, PhD
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

microUSB Cable

Antenna + Pigtail

Enclosure

Pycom LoPy 4

PySense

Our Lab equipment

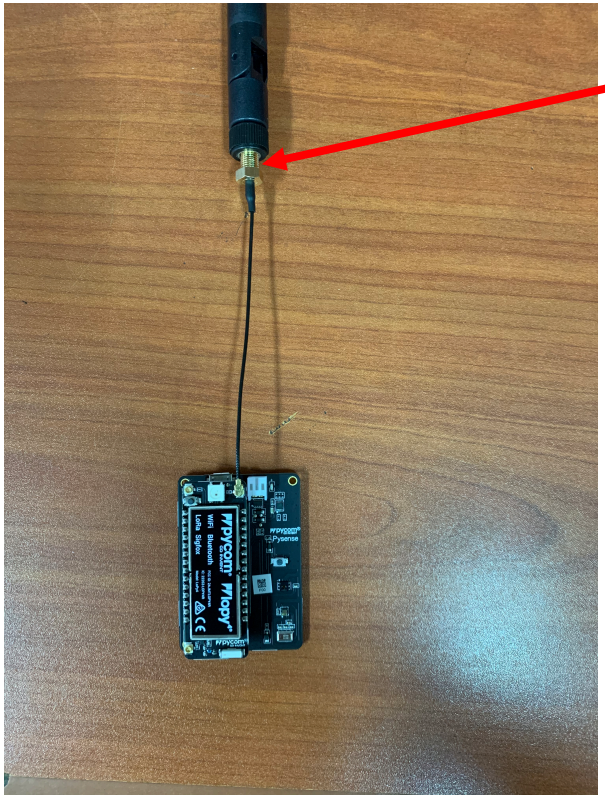
Our Lab equipment

Pigtail has to be connected on the right!

Pigtail are delicate!

Handle with care!

Our Lab equipment



Connect the antenna

Never operate the LoPy without the antenna connected!

Updating the firmware

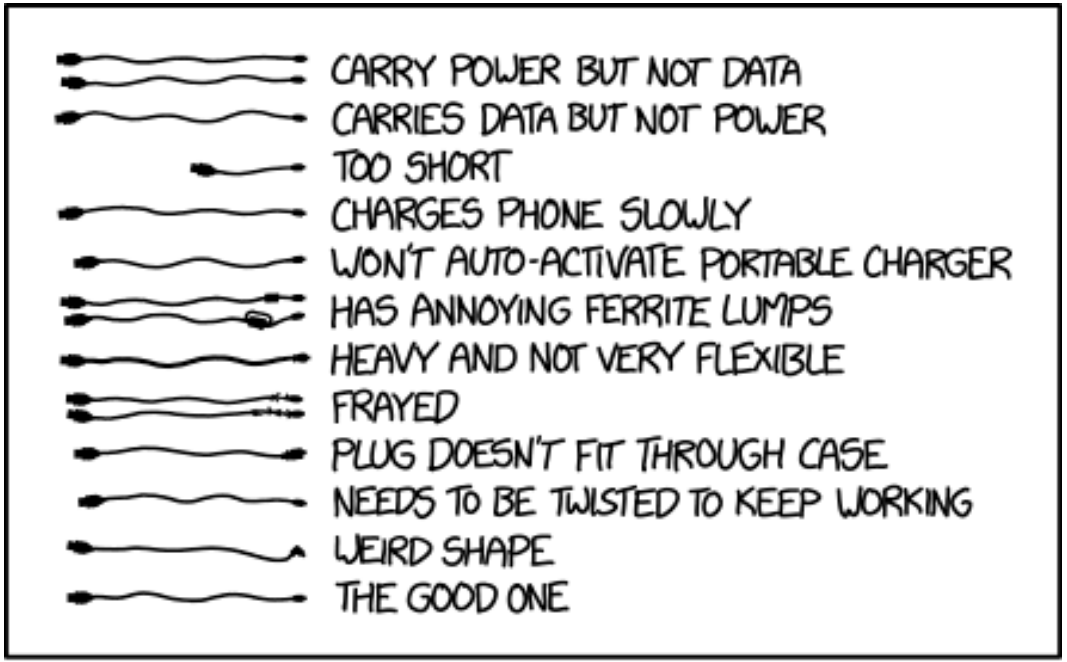
Please visit:

https://software.pycom.io/downloads/linux-1.16.2.html

Download the .deb file and install it.

Connect the LoPy to PC via USB

Micro USB cables must carry data



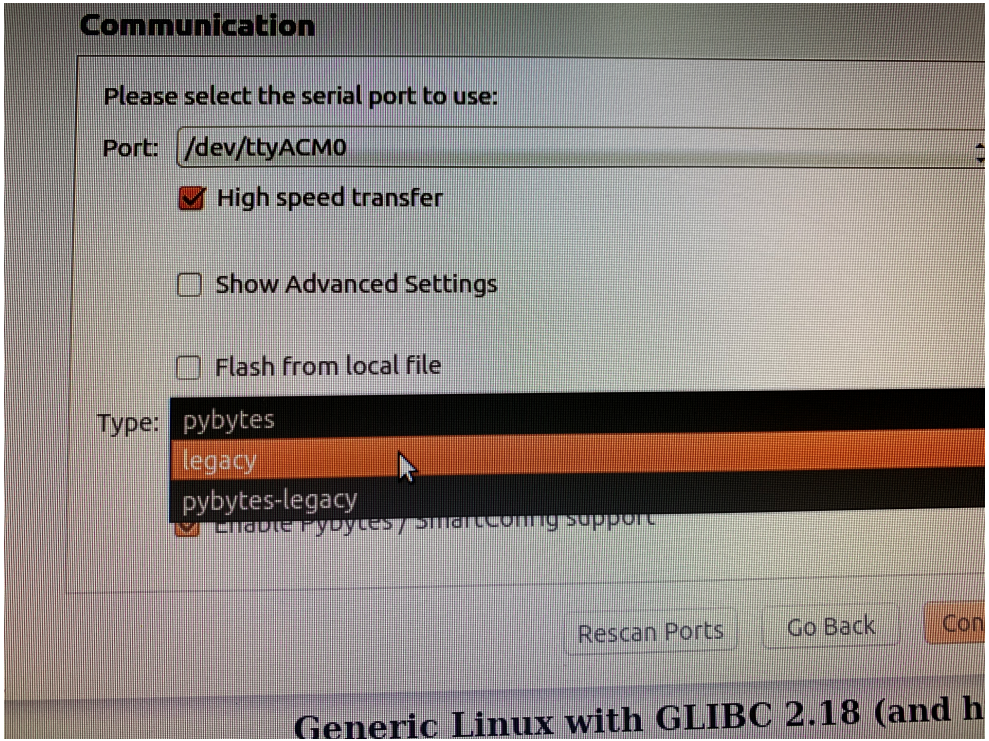
THE LAW OF USB CABLES:
NO MATTER HOW MANY YOU GET,
YOU ONLY EVER HAVE ONE GOOD ONE.

Workflow: launch the firmware updater

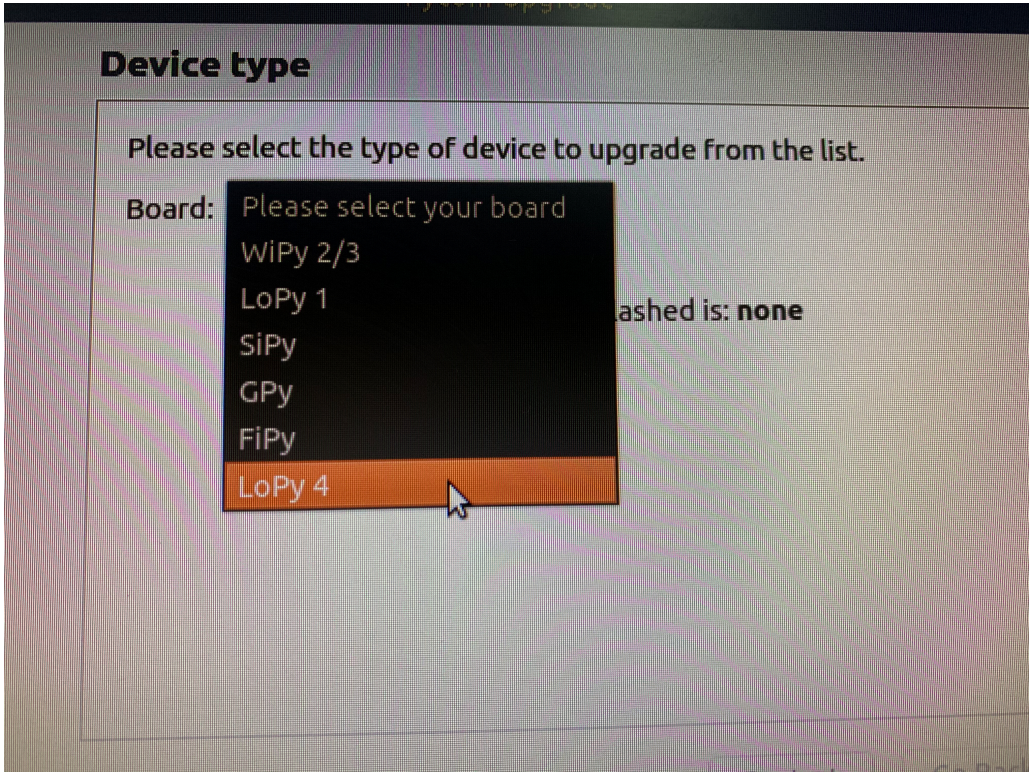
From the Terminal:

```
> sudo chmod 666 /dev/ttyACM0
> pycom-fwtool
```

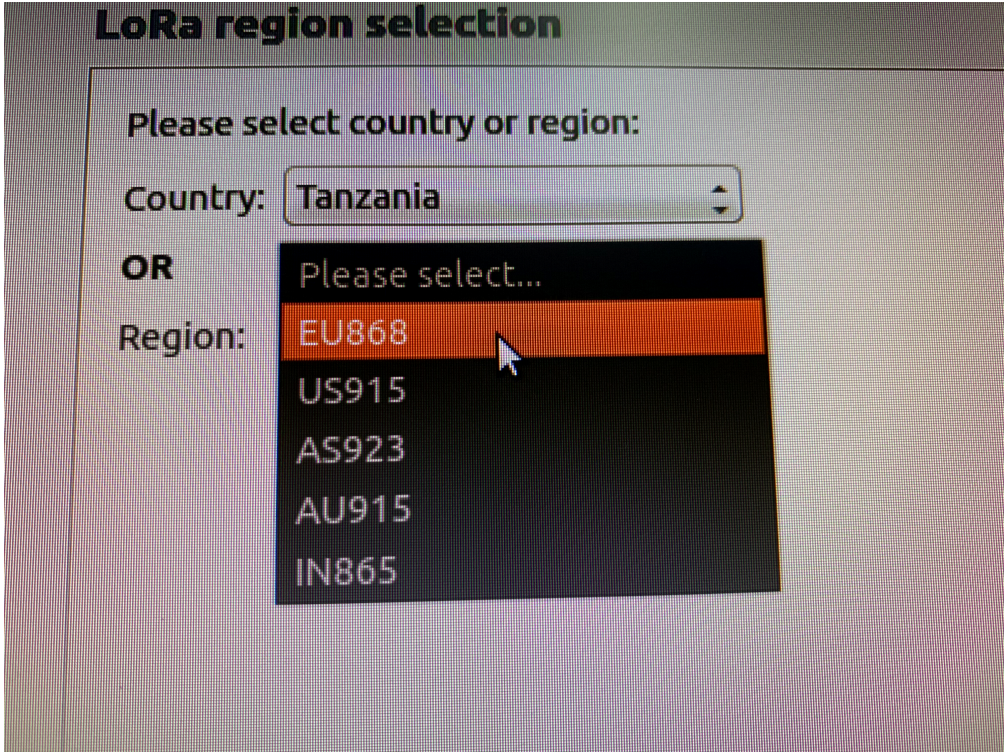
Workflow: launch the firmware updater



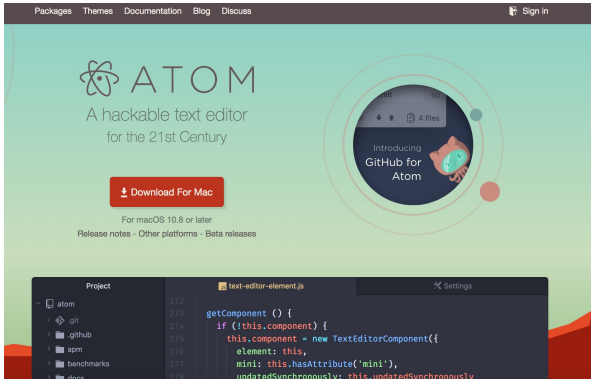
Workflow: launch the firmware updater



Workflow: launch the firmware updater



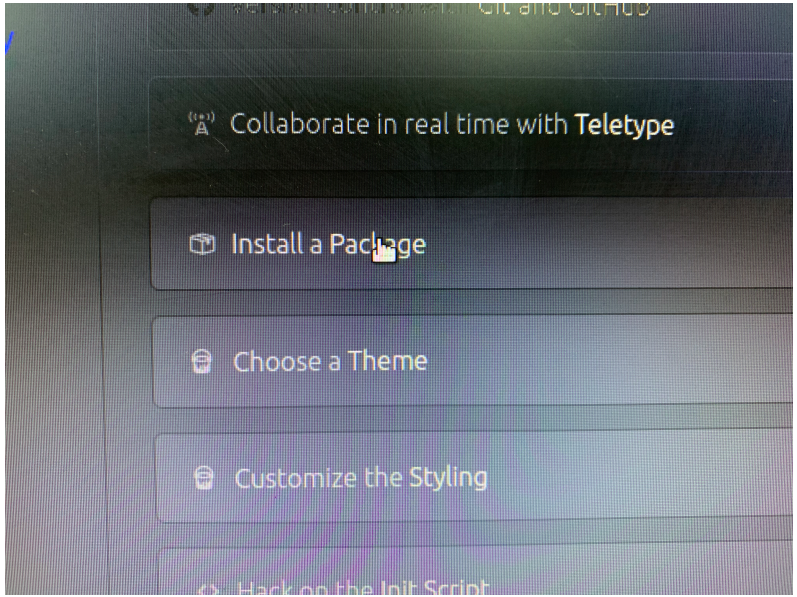
Workflow: Atom

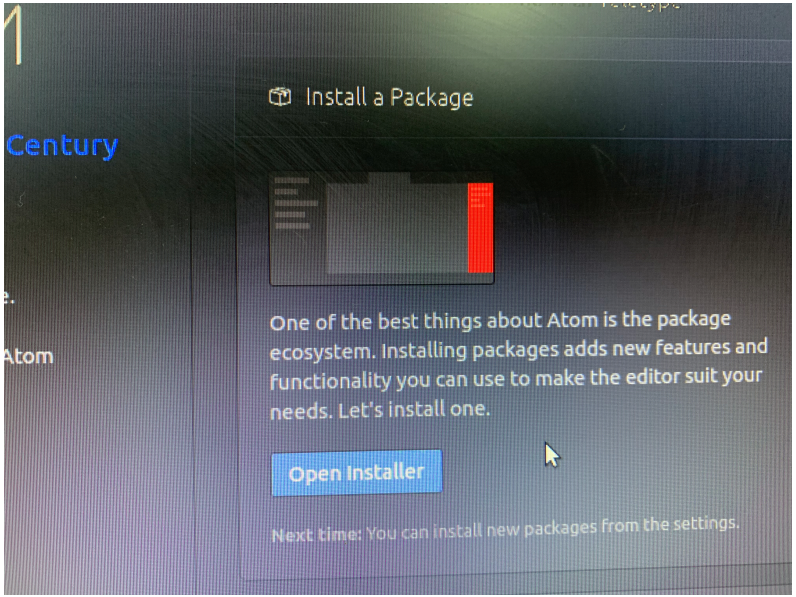


Please install Atom from www.atom.io

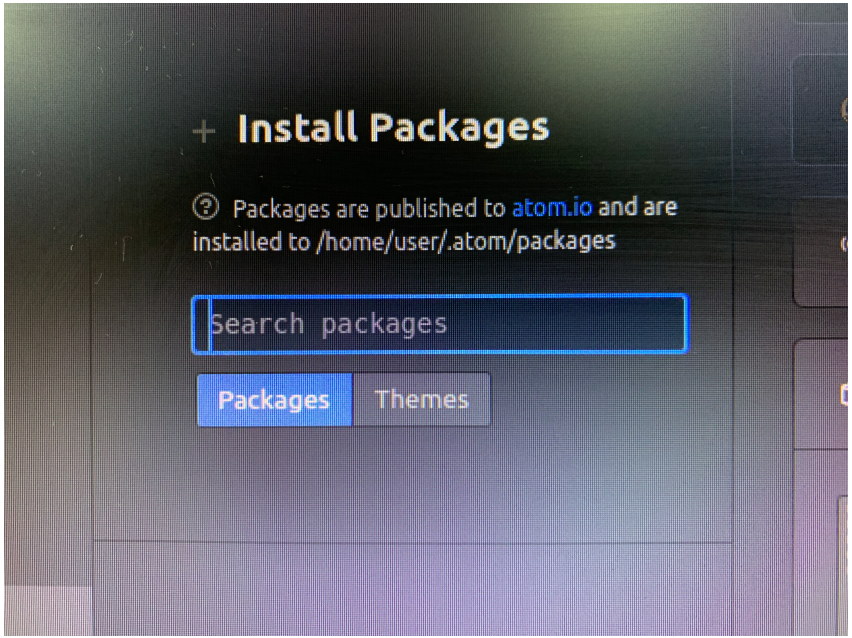
If you have it already, upgrade to the latest version!

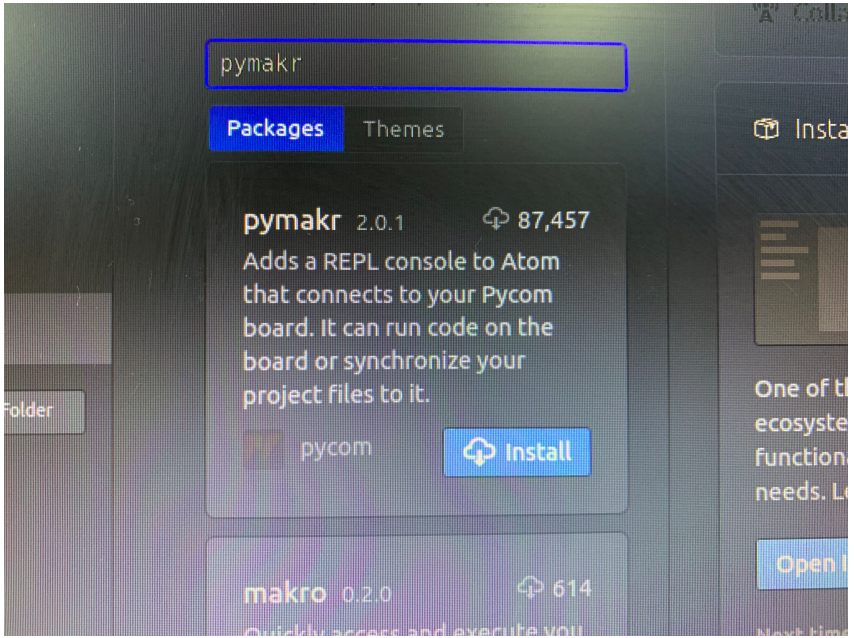
Workflow: install the **pymakr** package



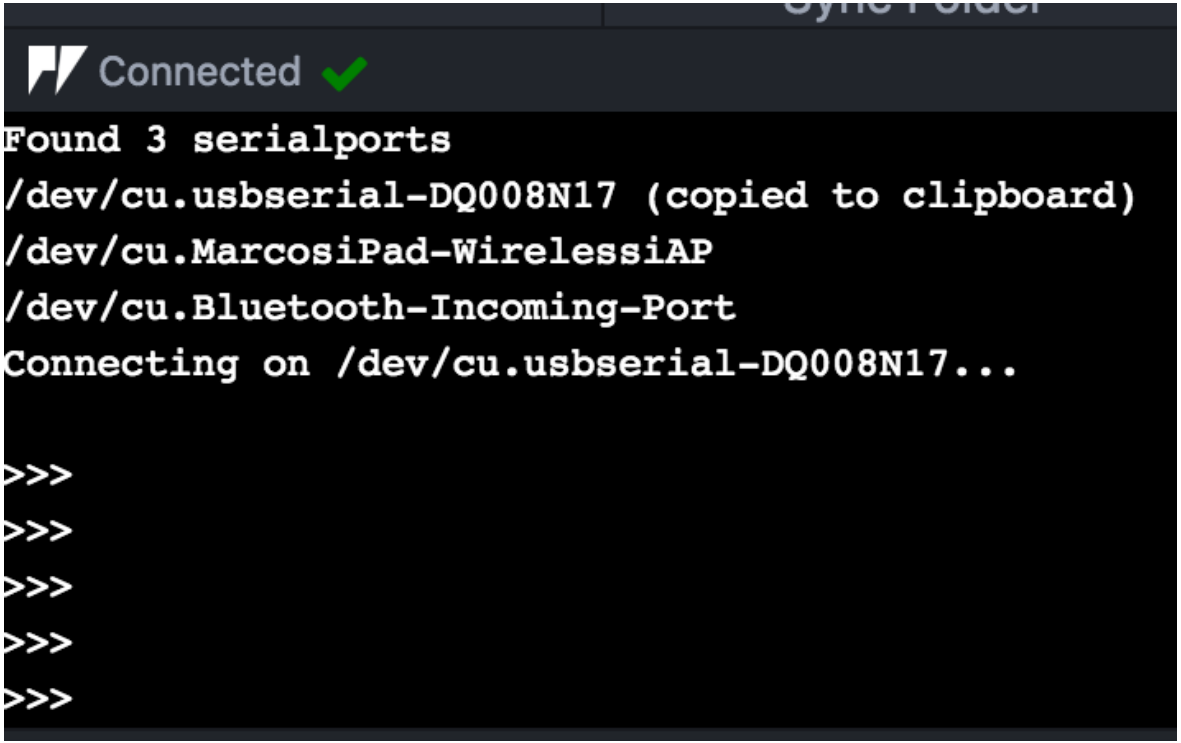


Workflow: install the **pymakr** package





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 uos
```

Then try listing the contents of the filesystem:

```
>>> uos.listdir()
```

REPL example

You can make directories:

```
>>> uos.mkdir('dir_iotlab')
```

You can remove directories:

```
>>> uos.rmdir('dir_iotlab')
```

REPL example

You can print the free space in kB:

```
>>> uos.getfree('/flash')
```

REPL example

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

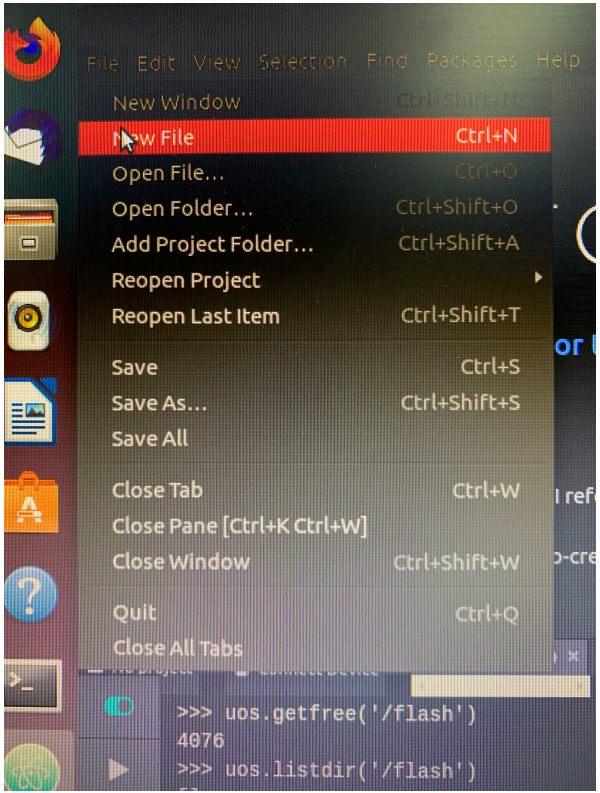
```
>>> uos.fsformat('/flash')
```

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**.

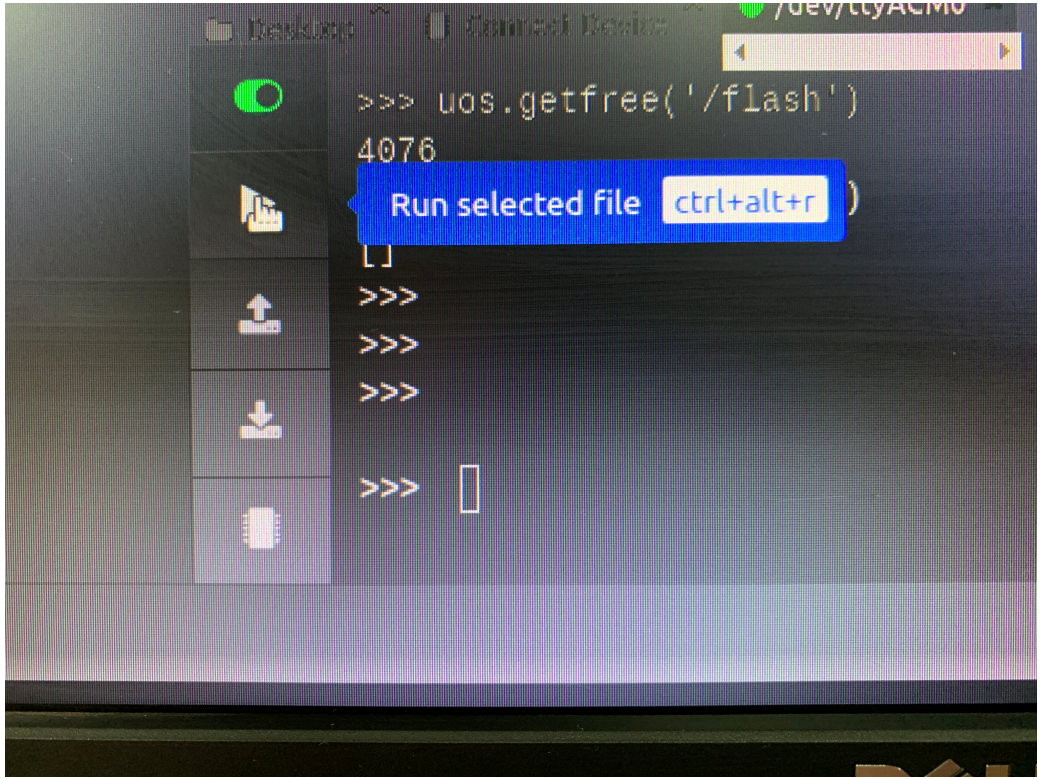
If you reboot, the code will not be executed again.

Executing code



Make sure you safe the file with a .py extension.

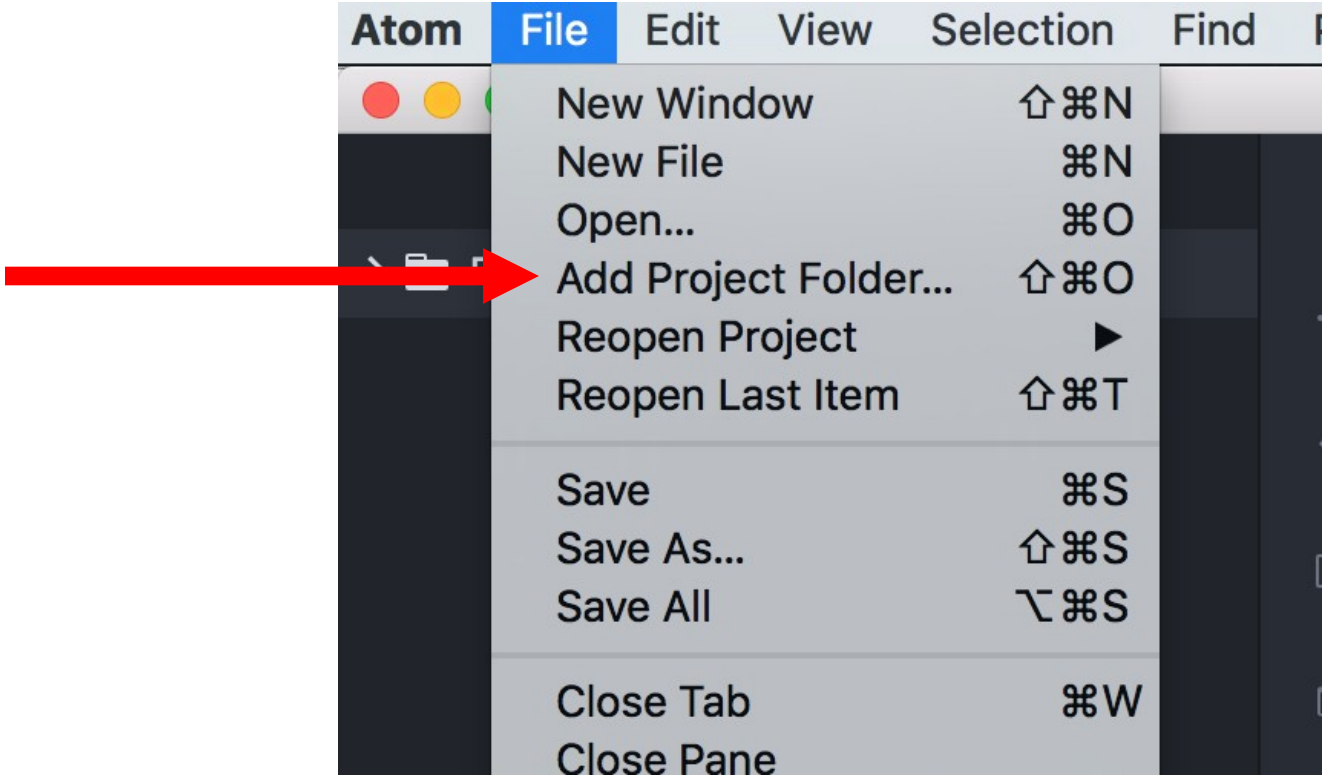
Executing code



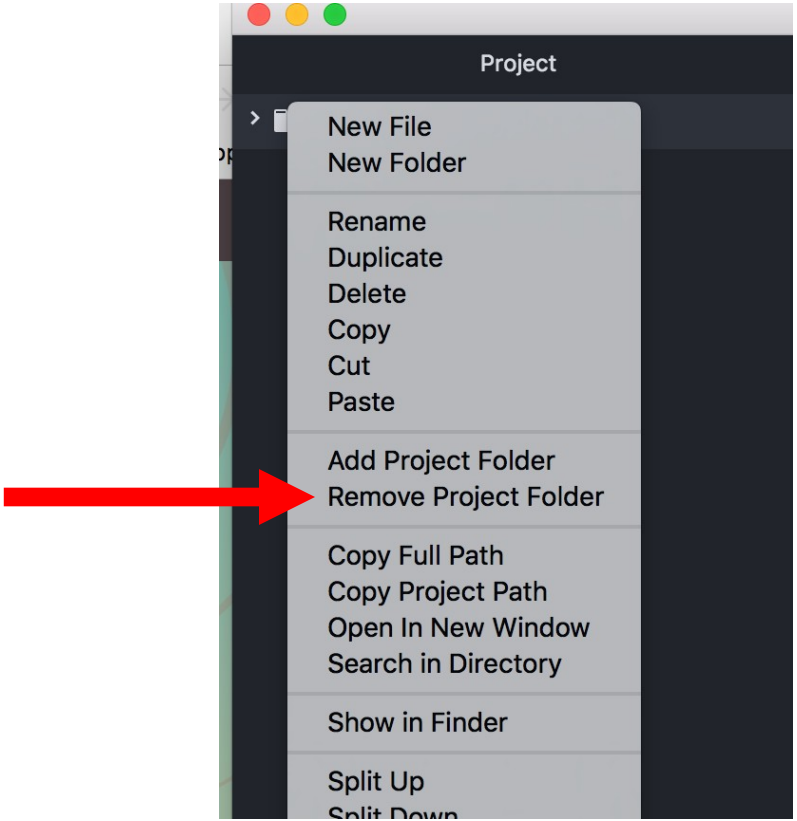
Executing code

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.

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.**

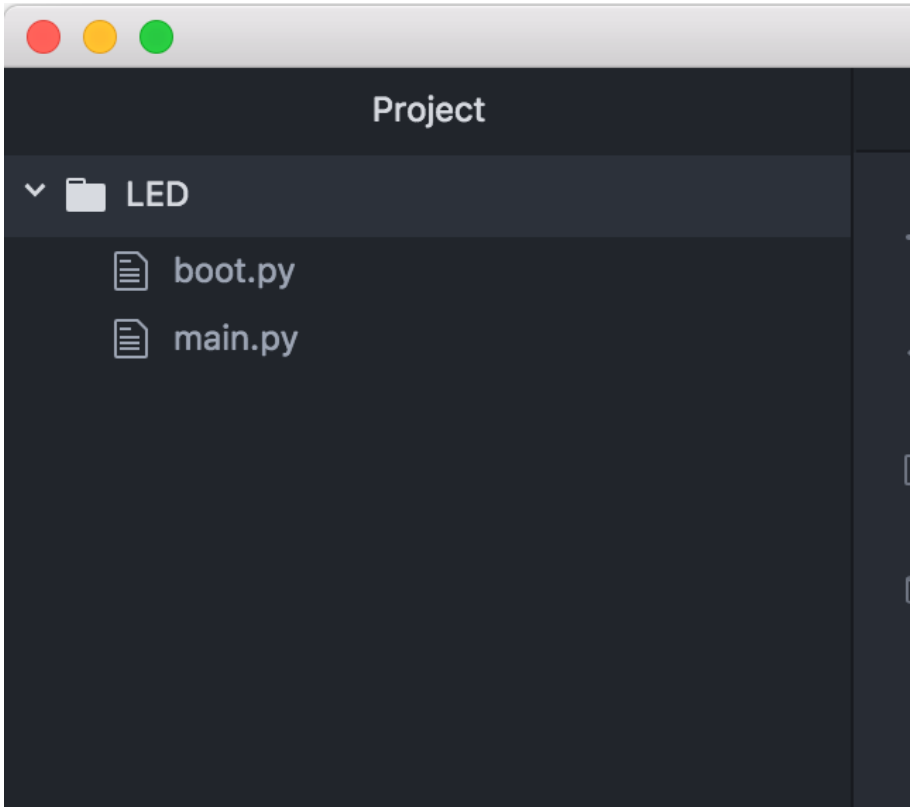
Workflow: project folder

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

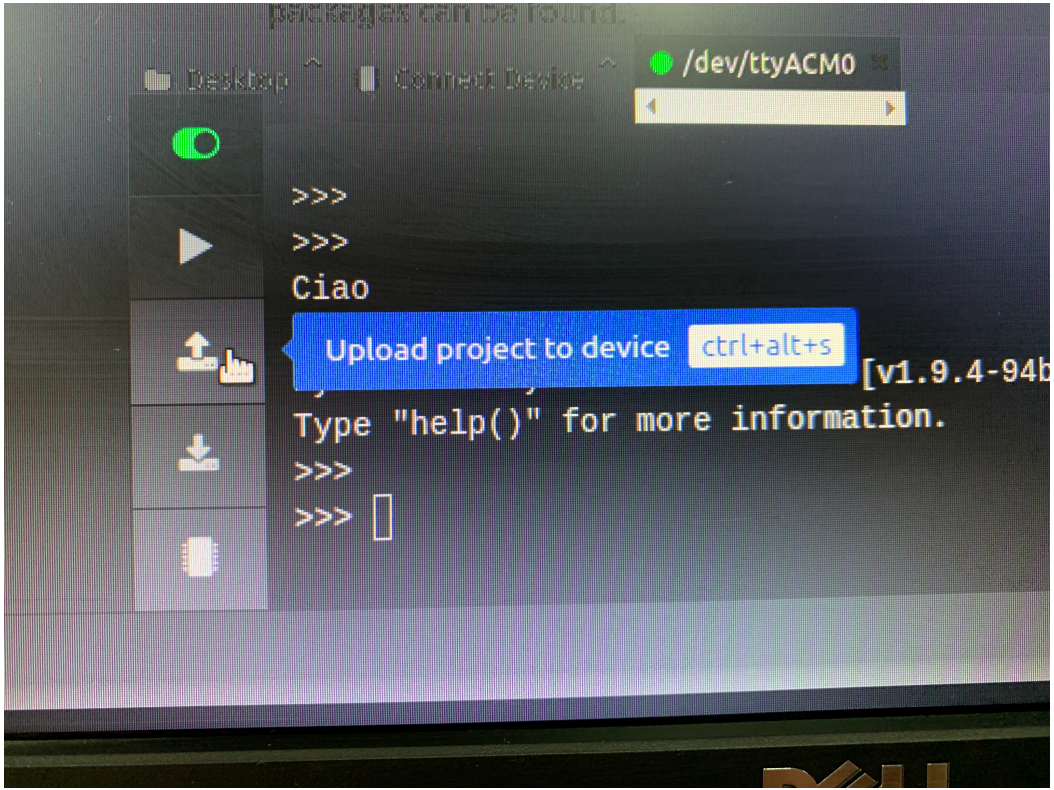
You should always have two files: **boot.py** (executed at boot time) and **main.py** (containing the main code).

The folder can also include libraries and other python files.

Workflow: example of project folder



Workflow: upload project folder



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 )
```

Alternative to Atom: mpy-repl-tool

mpy-repl-tool is a software tool that you can use to connect and control a LoPy via the command line

http://mpy-repl-tool.readthedocs.io/en/latest/index.html

To install it you have to execute:

```
> sudo apt-get install python3-pip

> python3 -m pip install mpy-repl-tool
```

Alternative to Atom: mpy-repl-tool

This command will list the serial ports and “hopefully” will automatically find your LoPy board.

```
> python3 -m there detect
```

To get a list of the files on the LoPy run:

```
> python3 -m there ls -l
```

Alternative to Atom: mpy-repl-tool

To uload the code files from your computer to the LoPy:

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

To start a serial terminal and get access to the REPL:

```
> python3 -m there -i
```

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

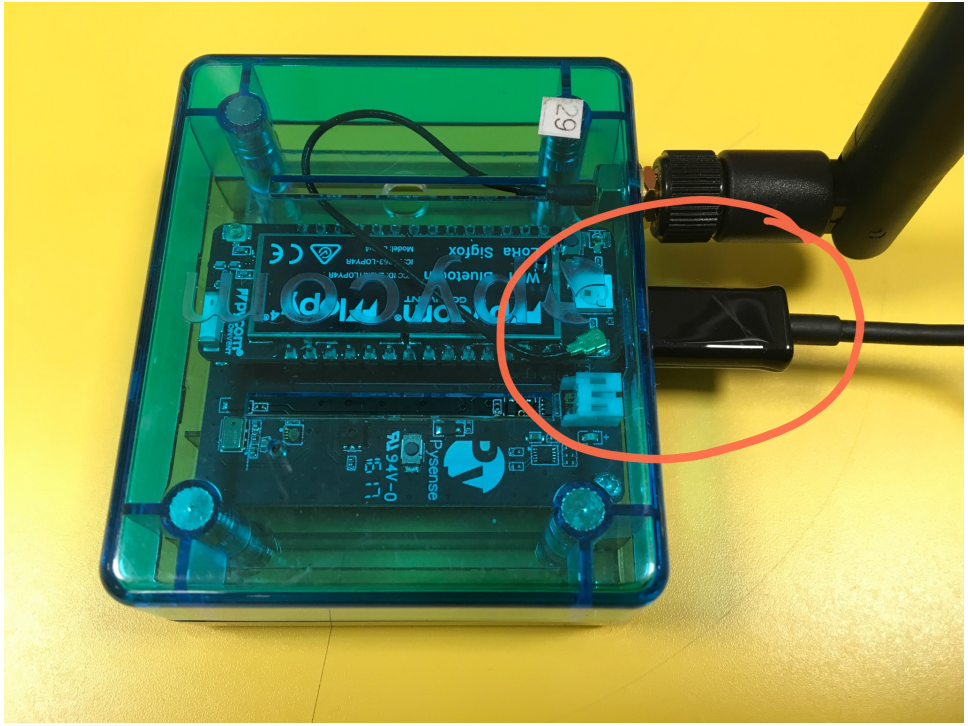
LED

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

LoPys don’t have screens!

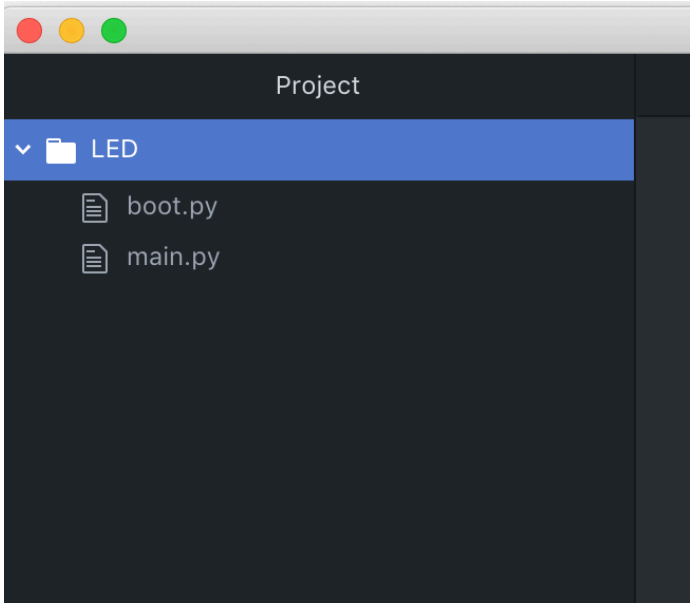
The LoPy module has one LED (big, white LED on the same side as the microUSB).

LED



LED

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



LED example, part 1

```
import pycom

import time

pycom.heartbeat(False)
```

Import the python libraries needed

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

LED example, part 1

```
for cycles in range(10):
    pycom.rgbled(0x007f00) # green
    time.sleep(5)
    pycom.rgbled(0x7f7f00) # yellow
    time.sleep(5)
   pycom.rgbled(0x7f0000) # red
    time.sleep(5)
```

Execute 10 times

Change LED color to green, yellow and red

Sleep for 5 seconds each time

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