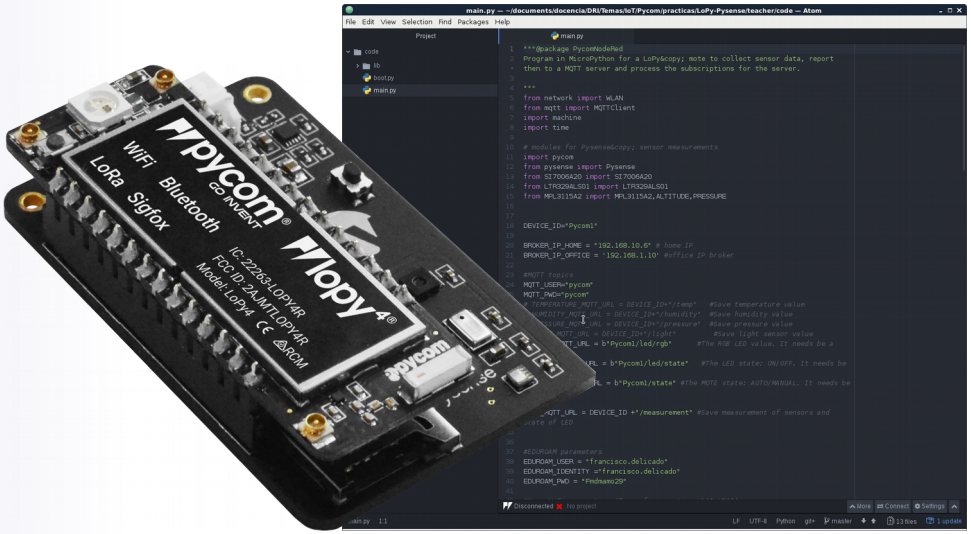
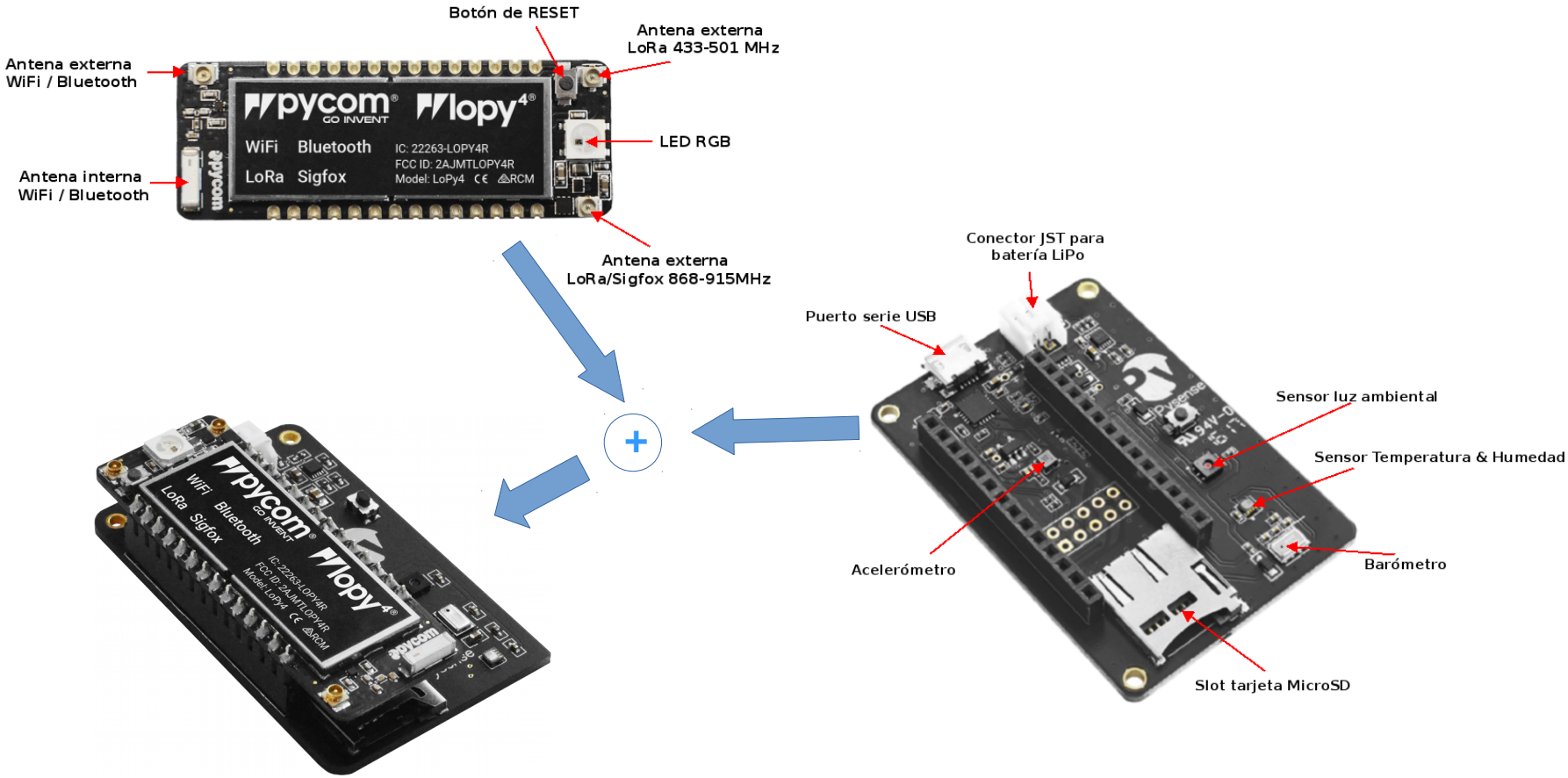
Programación del LoPy4 - Pysense

Micro-Python



LoPy^4 & Pysense



Programando LoPy4

MicroPython

- Implementación Python 3.5 para microcontroladores.
  - Sintaxis similar Python 3.5.
  - Compatibilidad total de MicroPython a Python 3.5, no al revés.
- Estructura proyecto MicroPython:

cert		Directory
lib		Directory
sys		Directory
boot.py	1734	Python
main.py	14	Python

```
#boot.py File
from machine import UART
import machine
import os
uart = UART(0, baudrate=115200)
os.dupterm(uart)

machine.main('main.py')
```

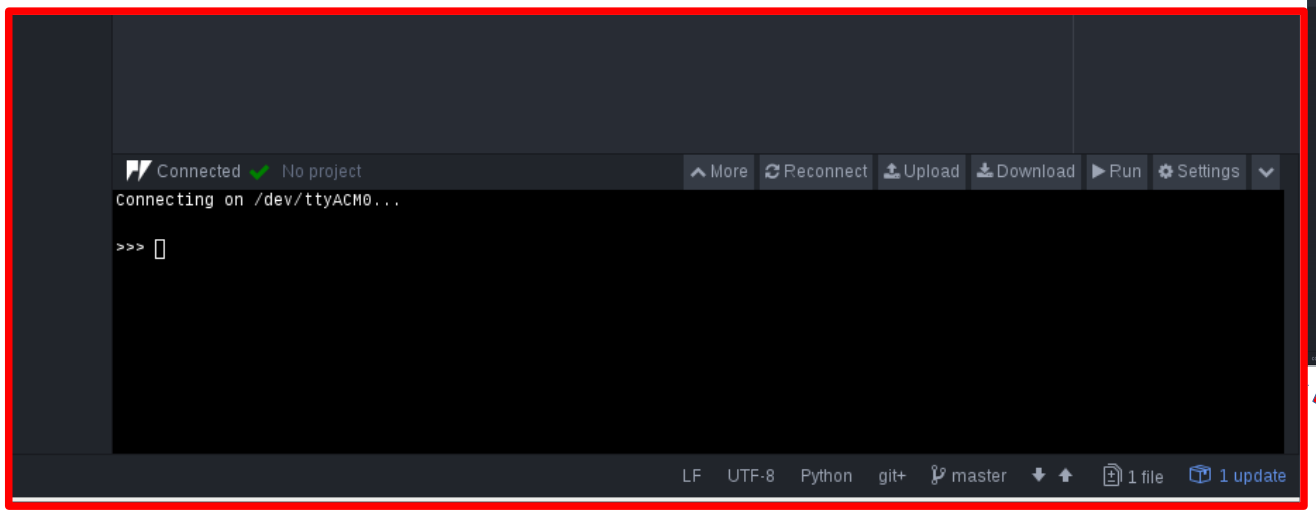
```
# main.py File
from network import WLAN
from mqtt import MQTTClient
...
STATE_AUTO = 0
STATE_MANUAL = 1
STATE_LEN_ON = 1
...
def sub_cb(topic, msg):
    print(topic+' Subscription message: '+ msg)
    if topic == MOTE_STATE_MQTT_URL:
        state_mote = int(msg)
        print("\tMOTE_STATE = %d"%state_mote)
    elif topic == LED_RGB_MQTT_URL:
...
pycom.heartbeat(False)
while True:
    temperature = si.temperature();
    msg_tmp = "\"temperature\":%.2f" % temperature
    light = li.light()
    msg_light = "\"light\":%.2f" % light[0]
...
```

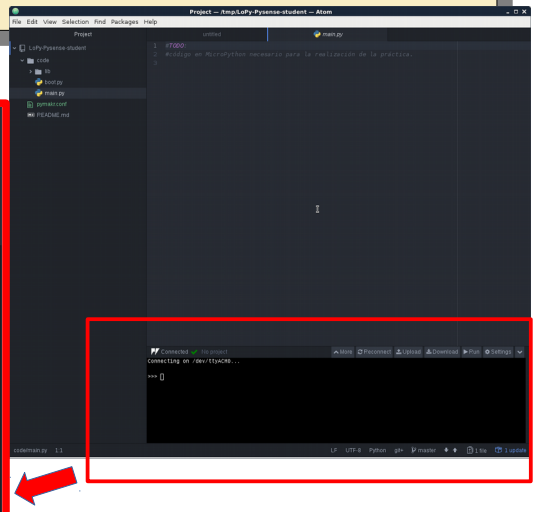
LoPy^4 & Atom

Pymakr: Plugin para Atom para interactuar con LoPy^4

- **Comunicación serie** a través **USB** Pysense o Pytrack.
- Subida de código.
- Ejecución directa de código.
- Uso de proyectos.
  - **pymakr.conf**: configura la comunicación.
- **Consola** (**REPL**, *Read Event Print Line*).

```
#pymakr.conf
{
  "address": "/dev/ttyACM0",
  "username": "micro",
  "password": "python",
  "sync_folder": "LoPy",
  "open_on_start": true,
  "sync_file_types": "py, txt, log, json, xml",
  "ctrl_c_on_connect": false
}
```





LoPy^4 - *Boot Modes*

https://docs.pycom.io/chapter/toolsandfeatures/bootmodes.html Consola REPL

Reinicio

- **Soft-reset**: Pulsar "**Ctrl+D**" en consola REPL, o

```
>>> import sys
>>> sys.exit()
```

- **Hard-reset**: Pulsar el botón de "reset" del LoPy^4, o

```
>>> import machine
>>> machine.reset()
```

Borrado "*flash*": Elimina todos los ficheros de la "*flash*"

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

Consejo: si el LoPy^4 tiene un mal comportamiento, realizar un borrado de la "*flash*" antes de hacer un reinicio del mismo.

LoPy4 - MicroPython

Estructura del main.py

#main.py (I)

```
#importación de módulos
from network import WLAN
from pysense import Pysense
...
from MPL3115A2 import MPL3115A2,ALTITUDE,PRESSURE

#Definición variables entorno: Constantes
DEVICE_ID="Pycom1"
BROKER_IP_HOME = "192.168.10.6"
BROKER_IP_OFFICE = '192.168.1.10'
STATE_LEN_ON = 1
STATE_LEN_OFF = 0

#Definición de funciones
def sub_cb(topic, msg):
    print(topic+' Subscription message: '+ msg)
    if topic == MOTE_STATE_MQTT_URL:
        state_mote = int(msg)
        print("\tMOTE_STATE = %d"%state_mote)
    elif topic == LED_RGB_MQTT_URL:
     ...

def sum_var(var1, var2):
    result = var1 + var2
    print(var1+' + '+ var2 +' = ' + result)
    return result
```

#main.py (II)

```
#Inicialización del dispositivo
wlan = WLAN(mode=WLAN.STA)
wnets = wlan.scan()
for net in wnets:
    ....
py = Pysense();
li = LTR329ALS01(py) # Returns light level in lux.
si = SI7006A20(py) #Return percentage relative humidity and
temperature.
pycom.heartbeat(False) #Stop automatic LED flashing

#Bucle infinito: Cuerpo programa
while True:
    temperature = si.temperature();
    msg_tmp = "\"temperature\":%.2f" % temperature
    ....
```

Timers (https://docs.pycom.io/chapter/tutorials/all/timers.html)

```
#Cronómetro
from machine import Timer
import time

chrono = Timer.Chrono()

chrono.start() # inicia crono
time.sleep(1.25) # stop 1.25 seconds
lap = chrono.read() # lee crono
time.sleep(1.5)
chrono.stop() # para crono
total = chrono.read()

print("the racer took %f seconds to finish the race" % total)
print("  %f seconds in the first lap" % lap)
print("  %f seconds in the last lap" % (total - lap))
```

```
#Alarma. Genera llamada a función de callback
from machine import Timer

class Clock:

    def __init__(self):
        self.seconds = 0
        self.__alarm = Timer.Alarm(self._seconds_handler, 1,
                                   periodic=True) #Define Alarma

    def _seconds_handler(self, alarm): #Función de callback
        self.seconds += 1
        print("%02d seconds have passed" % self.seconds)
        if self.seconds == 10:
            alarm.callback(None) # stop alarma tras 10 seconds

clock = Clock() #Instancia objeto
```

Threading (https://docs.pycom.io/chapter/tutorials/all/threading.html)

```
#Threading (Hilos)
import _thread
import time

def th_func(delay, id):
    while True:
        time.sleep(delay)
        print('Running thread %d' % id)

for i in range(3):
    _thread.start_new_thread(th_func, (i + 1, i))
```

RGB LED (https://docs.pycom.io/chapter/tutorials/all/rgbled.html)

```
#RGB LED
import pycom
import time

pycom.heartbeat(False) #Apago el LED.
pycom.rgbled(0xff0000)  #Pongo el LED a rojo
time.sleep(5)
pycom.rgbled(0x00ff00)  #Pongo el LED a Verde
```

Pysense - MicroPython API (I)

Acelerómetro 3-ejes (LIS2HH12)

https://docs.pycom.io/chapter/pytrackpysense/apireference/pysense.html

```
#Acelerómetro
from pysense import Pysense
from LIS2HH12 import LIS2HH12

py = Pysense()
acc = LIS2HH12(py)

while True:
  pitch = acc.pitch()  #Lee pitch
  roll = acc.roll()  #Lee roll
  accel = acc.acceleration() #Lee acceleración 3 ejes
  print('{},{}'.format(pitch,roll))
  print ("x: %0.2f,  y: %0.2f, z: %0.2f" % (accel[0],accel[1],accel[3])
  ...
```

Luminosidad(LTR-329ALS-01)

https://docs.pycom.io/chapter/pytrackpysense/apireference/pysense.html

```
#Luminosidad
from pysense import Pysense
from LTR329ALS01 import LTR329ALS01

py = Pysense()
li = LTR329ALS01(py)

while True:
  light = li.light() #Leo dos valores de luminosidad
  print ("light1: %0.2f,  light2: %0.2f" % (light[0],light[1])
  ....
```

Pysense - MicroPython API (II)

Humedad y temperatura (SI7006A20)

https://docs.pycom.io/chapter/pytrackpysense/apireference/pysense.html

```
#Humedad y Temperatura
from pysense import Pysense
from SI7006A20 import SI7006A20

py = Pysense()
si = SI7006A20(py)

while True:
  temp = si.temperature() #Leo temperatura
  hum = si.humidity() #Leo humedad
  print ("humedad: %0.2f,  temperatura: %0.2f" % (hum,temp)
  ....
```

Presión con Altitud (MPL3115A2)

https://docs.pycom.io/chapter/pytrackpysense/apireference/pysense.html

```
#Acelerómetro
from pysense import Pysense
from MPL3115A2 import MPL3115A2,ALTITUDE,PRESSURE

py = Pysense()
mpl = MPL3115A2(py, mode=ALTITUDE)
mpl2 = MPL3115A2(py, mode=PRESSURE)

while True:
  alt = mpl.altitude()  #Lee altitud
  bar = mpl2.pressure()  #Lee presion
  print ("altitude: %0.2f,  presión: %0.2f" % (alt,bar)
  ...
```

Bibliografía

- Pycom Documentation (https://docs.pycom.io/)