```
# import os
# os.fsformat('/flash')

import network
import time
import pycom
from pysense import Pysense
from network import Bluetooth

from mqtt import MQTTClient
import usocket
import sys
import machine
from machine import Pin
from machine import Timer

from LIS2HH12 import LIS2HH12
from SI7006A20 import SI7006A20
from LTR329ALS01 import LTR329ALS01
from MPL3115A2 import MPL3115A2,ALTITUDE,PRESSURE

#*****************************************************************************
#
#                   iniciação
#*****************************************************************************

py = Pysense()

pycom.heartbeat(False)

pycom.rgbled(0xFF0000) # RED

#*****************************************************************************
#
#                   ação face a informação recebida
#*****************************************************************************

def sub_cb(topic, msg):
    print(msg)
    if (msg == b"OFF" or msg == b"off" or msg == b"Off"):
        pycom.rgbled(0x000000) # OFF
        MQTT_C.publish(MQTT_Topic_luz,str(0))
    if (msg == b"ON"or msg == b"on" or msg == b"On"):
        pycom.rgbled(0x00FF00) # Green
        MQTT_C.publish(MQTT_Topic_luz,str(1))
    if (msg == b"CHECK" or msg == b"check" or msg == b"Check"): #control
        _timmer_control(None)


#*****************************************************************************
#
#                   CONFIG NETWORK
#*****************************************************************************

wlan = network.WLAN(mode=network.WLAN.STA)
# wlan.connect('NOS-AE6C', auth=(network.WLAN.WPA2, 'AX7AUAHC'))
wlan.connect('labs', auth=(network.WLAN.WPA2, 'robot1cA!ESTG'))

while not wlan.isconnected():
    time.sleep_ms(50)

print("%============================%")
print("% WLAN Connected             %")
print("%============================%")

print("Config:", wlan.ifconfig())

pycom.rgbled(0x00FF00) # Green
```

```
#****************************************************************************#
#
#                                CONFIG MQTT                                 #
#****************************************************************************#

MQTT_Server = "io.adafruit.com"                     # Broker
MQTT_Port = 1883                                      # Porta de acesso
MQTT_Client_ID = '01e8e120-19e0-11eb-a2e4-b32ea624e442'  # Identificação do cliente
MQTT_USER = "Rs_Drumond_Paulo"                             # Nome do Topico
MQTT_PASSWORD="aio_zUkR87Ml8QYeWpOTFPfKxG0MxEvf"
MQTT_Temp = "Rs_Drumond_Paulo/feeds/temp-sala"
MQTT_Sala_OnOff = "Rs_Drumond_Paulo/feeds/sala-on-of"
MQTT_Topic_humidade = "Rs_Drumond_Paulo/feeds/humidade-sala"
MQTT_Topic_luminosidade = "Rs_Drumond_Paulo/feeds/luminosidade-sala"
MQTT_Topic_luz ="Rs_Drumond_Paulo/feeds/luz-sala"
MQTT_Topic_acelerometro ="Rs_Drumond_Paulo/feeds/alarm-text"

# MQTT Server Conection and Subscription
MQTT_C = MQTTClient(MQTT_Client_ID, MQTT_Server,MQTT_USER,MQTT_PASSWORD)     # Definição do
cliente
MQTT_C.set_callback(sub_cb)
MQTT_C.connect()                                      # Registo de conexão

MQTT_C.subscribe(MQTT_Temp)                          # Subscrição do topico
MQTT_C.subscribe(MQTT_Sala_OnOff)
MQTT_C.subscribe(MQTT_Topic_humidade)
MQTT_C.subscribe(MQTT_Topic_luminosidade)
MQTT_C.subscribe(MQTT_Topic_luz)
MQTT_C.subscribe(MQTT_Topic_acelerometro)

#**************************************************************************

# Button definition
p_in_Button = Pin('P14', mode=Pin.IN, pull=Pin.PULL_UP)

#*****************************************************************************
#
#                  CONFIG and CHEK SENSORES
#*****************************************************************************

mp = MPL3115A2(py,mode=ALTITUDE) # Returns height in meters. Mode may also be set to
PRESSURE, returning a value in Pascals
print("MPL3115A2 temperature: " + str(mp.temperature()))
print("Altitude: " + str(mp.altitude()))
mpp = MPL3115A2(py,mode=PRESSURE) # Returns pressure in Pa. Mode may also be set to
ALTITUDE, returning a value in meters
print("Pressure: " + str(mpp.pressure()))
#
si = SI7006A20(py)
print("Temperature: " + str(si.temperature())+ " deg C and Relative Humidity: " +
str(si.humidity()) + " %RH")
print("Dew point: "+ str(si.dew_point()) + " deg C")
t_ambient = 24.4
print("Humidity Ambient for " + str(t_ambient) + " deg C is " +
str(si.humid_ambient(t_ambient)) + "%RH")
#
lt = LTR329ALS01(py)
print("Light (channel Blue lux, channel Red lux): " + str(lt.light()))
#
li = LIS2HH12(py)
print("Acceleration: " + str(li.acceleration()))
print("Roll: " + str(li.roll()))
print("Pitch: " + str(li.pitch()))

#*****************************************************************************
#
#                  CONFIG Bluetooth
#*****************************************************************************
bluetooth = Bluetooth()
```

```
bluetooth.set_advertisement(name='LoPy Enigma', service_uuid=b'1234567890123456')

def conn_cb (bt_o):
    events = bt_o.events()
    if  events & Bluetooth.CLIENT_CONNECTED:
    print("Client connected")
    elif events & Bluetooth.CLIENT_DISCONNECTED:
    print("Client disconnected")

bluetooth.callback(trigger=Bluetooth.CLIENT_CONNECTED | Bluetooth.CLIENT_DISCONNECTED,
handler=conn_cb)
bluetooth.advertise(True)
srv1 = bluetooth.service(uuid=b'1234567890123456', isprimary=True)
chr1 = srv1.characteristic(uuid=b'ab34567890123456', value=2)
#chr2 = srv1.characteristic(uuid=b'ab34567890123465', value=2)

def char1_cb(chr,xx):
    print("Write request with value = {}".format(chr.value()))
    topic=''
    sub_cb(topic, chr.value())


char1_cb=chr1.callback(trigger=Bluetooth.CHAR_WRITE_EVENT, handler=char1_cb,arg=chr1)

print("Bluetooth config ok")
pycom.rgbled(0x0000FF) # Blue

#*****************************************************************
#
#  Timer control
#*****************************************************************

def _timmer_control(alarm):
    MQTT_C.publish(MQTT_Temp,str(mp.temperature()))
    time.sleep_ms(100)
    MQTT_C.publish(MQTT_Topic_humidade,str(si.humid_ambient(t_ambient)))
    time.sleep_ms(100)
    MQTT_C.publish(MQTT_Topic_luminosidade,str(lt.light()[0]))
    time.sleep_ms(100)
    MQTT_C.publish(MQTT_Topic_acelerometro,str(li.acceleration()))
    time.sleep_ms(100)

alarm=Timer.Alarm(_timmer_control,300,periodic=True)  # desperta apos 5 min


# Main Program
while True:
    if p_in_Button()== 0:
        _timmer_control(None)

    MQTT_C.check_msg()
```