Instituto Tecnológico Autónomo de México

Departamento Académico de Sistemas Digitales

*Nuevas Tecnologías Inalámbricas*

***Comunicación en IoT con MQTT***

Integrantes:

Alfonso Venancio- 149211

Efraín Aguilar- 149643

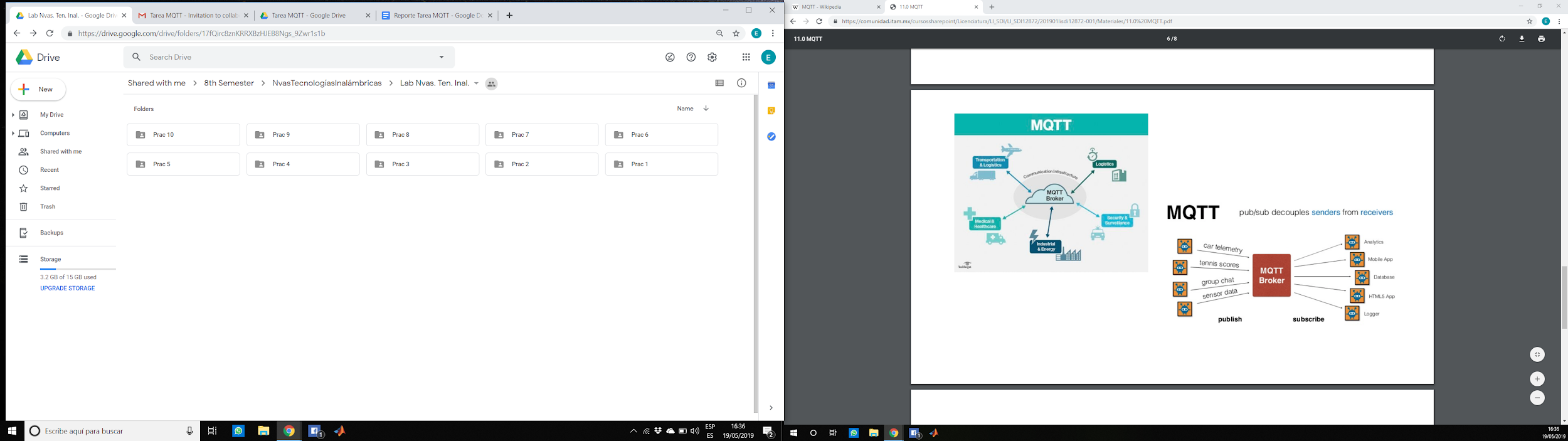
Leandro Pantoja- 150883

Ulises Alejandre - 159235

Mayo de 2019

**Introducción a MQTT**

MQTT (Message Queuing telemetry transport). es un protocolo de mensaje cliente-servidor donde se publica y subscribe. Está diseñado para ser usado con facilidad y ser simple. Su aplicación principal es IoT donde se tiene un límite en ancho de banda y una porción pequeña de código.



Su diseño permite la comunicación con dispositivos con poca memoria y procesamiento ya que utiliza comandos simples. Facilita la distribución uno a muchos y comparando con el protocolo HTTP es 93 veces más rápido para redes 3G si es que el objetivo de la aplicación tiene limitante de recursos y la información intercambiada está delimitada por mensajes cortos.

**Implementación de los clientes**

Como no fue posible hacer la publicación de mensajes usando el broker de Losant y la librería MQTT de python (aparecía un mensaje que decía que no se tenía permitido publicar y que se conectó de forma insegura a Losant). Por otro lado, la librería losant-mqtt de python permitía publicar sin problemas a la plataforma, pero sólo al tópico /state. Por estas razones, se decidió que los clientes usaran un broker local.

Para la implementación de los clientes de MQTT se necesitó instalar la librería paho-mqtt de python; la cual se instaló mediante pip, usando el comando *pip install paho-mqtt*. Para poder tener un broker local de MQTT se instaló mosquitto (de la página <https://mosquitto.org/download/>). Para Ubuntu, *se usó el comando sudo apt-get install mosquitto*.

A partir de ejemplos encontrados en internet, se escribieron los clientes que simulan a los sensores. Por ejemplo, el cliente de humedad es el siguiente:

import random

import paho.mqtt.client as mqtt

import time

def sim\_hum():

#Función que simula un sensor de humedad.

#Regresa un float aleatorio entre 0 y 100

temperatura = random.random()\*100

return temperatura

def on\_publish(client,userdata,result):

#Función que determina la acción del cliente al publicar un mensaje

print("Datos publicados en TempHum")

pass

broker="localhost" #dominio o IP del broker mqtt

port=1884 #puerto que se usará

client1= mqtt.Client(client\_id="Humedad") #instancia del cliente mqtt

client1.on\_publish = on\_publish #asignar acciones al publicar

client1.connect(broker,port) #conectarse al broker

#Correr cliente indefinidamente, publicando un valor aleatorio

# de temperatura al tópico /TempHum cada 10 segundos

while True:

ret= client1.publish("TempHum",sim\_hum())

time.sleep(10)

Los otros clientes que simulan sensores tienen una estructura similar, sólo cambian los parámetros de publicación y la simulación del sensor.

Los clientes que están suscritos a los tópicos de los sensores también se basaron de ejemplo que se encontraron en internet. El siguiente código es el respectivo al cliente correspondiente a los sensores de temperatura y humedad:

import ssl

import sys

import paho.mqtt.client

def on\_connect(client, userdata, flags, rc):

#Función que determina que hace el cliente al conectarse

print('Conectado a (%s)' % client.\_client\_id)

client.subscribe(topic='TempHum', qos=2) #conectarse al tópico /Pres

def on\_message(client, userdata, message):

#Función que determina que hará el cliente al detectar

# que se envió un mensaje al tópico al que está suscrito

print('------------------------------')

print('Topico: %s' % message.topic)

print('Carga: %s' % message.payload)

print('Nivel QoS: %d' % message.qos)

def main():

#Instanciar el cliente

client = paho.mqtt.client.Client(client\_id='CliTempHum', clean\_session=False)

client.on\_connect = on\_connect #asignar acciones al conectarse

client.on\_message = on\_message #asignar acciones al detectar el envío de mensajes

client.connect(host='localhost', port=1884) #conectarse al host por el puerto asignado

client.loop\_forever() #dejar el cliente activo indefinidamente

main()

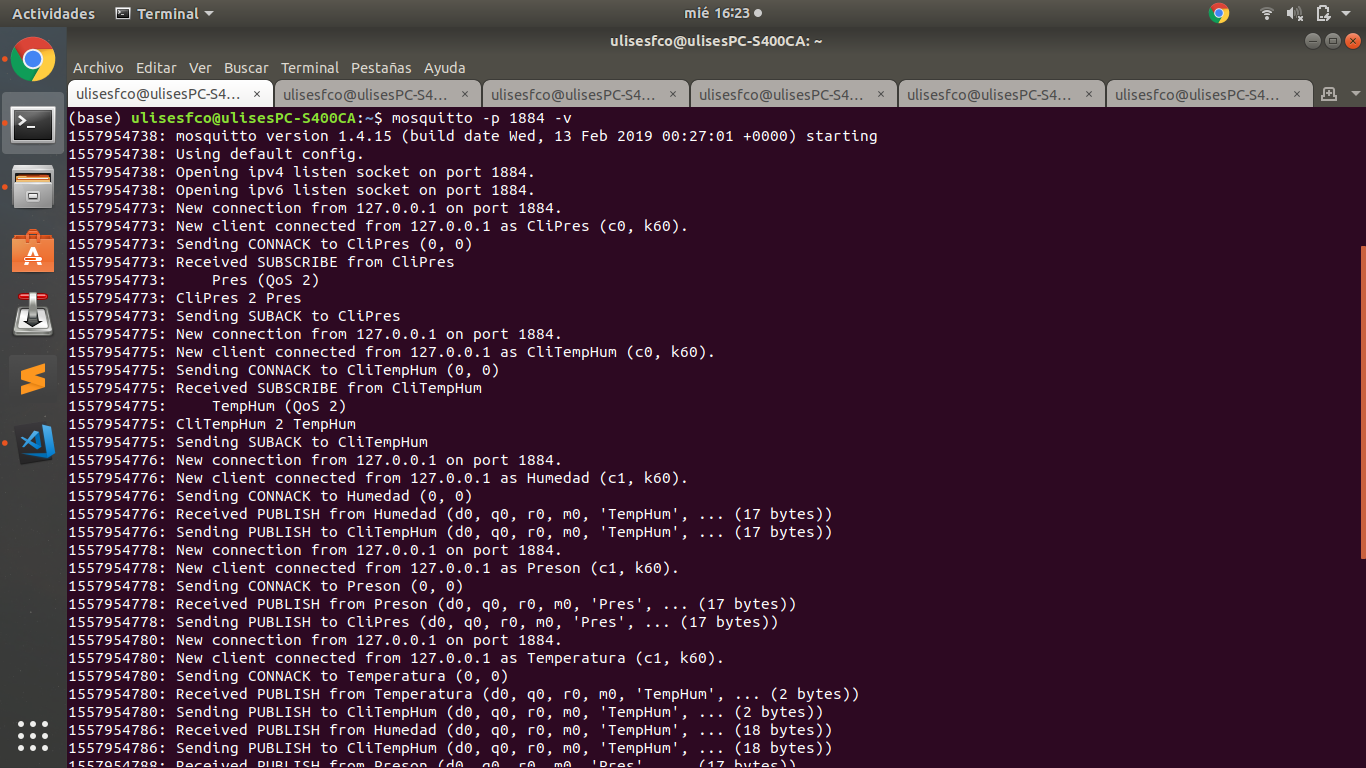
sys.exit(0)

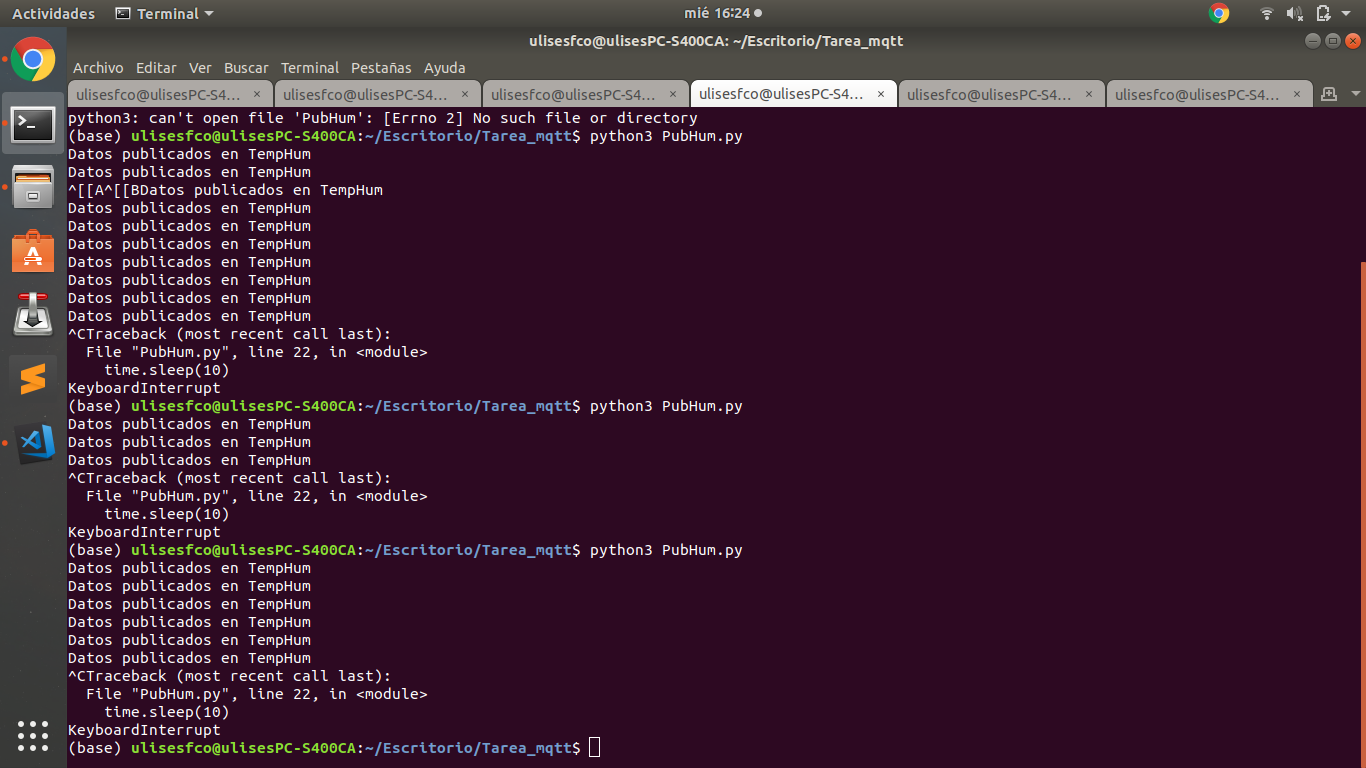
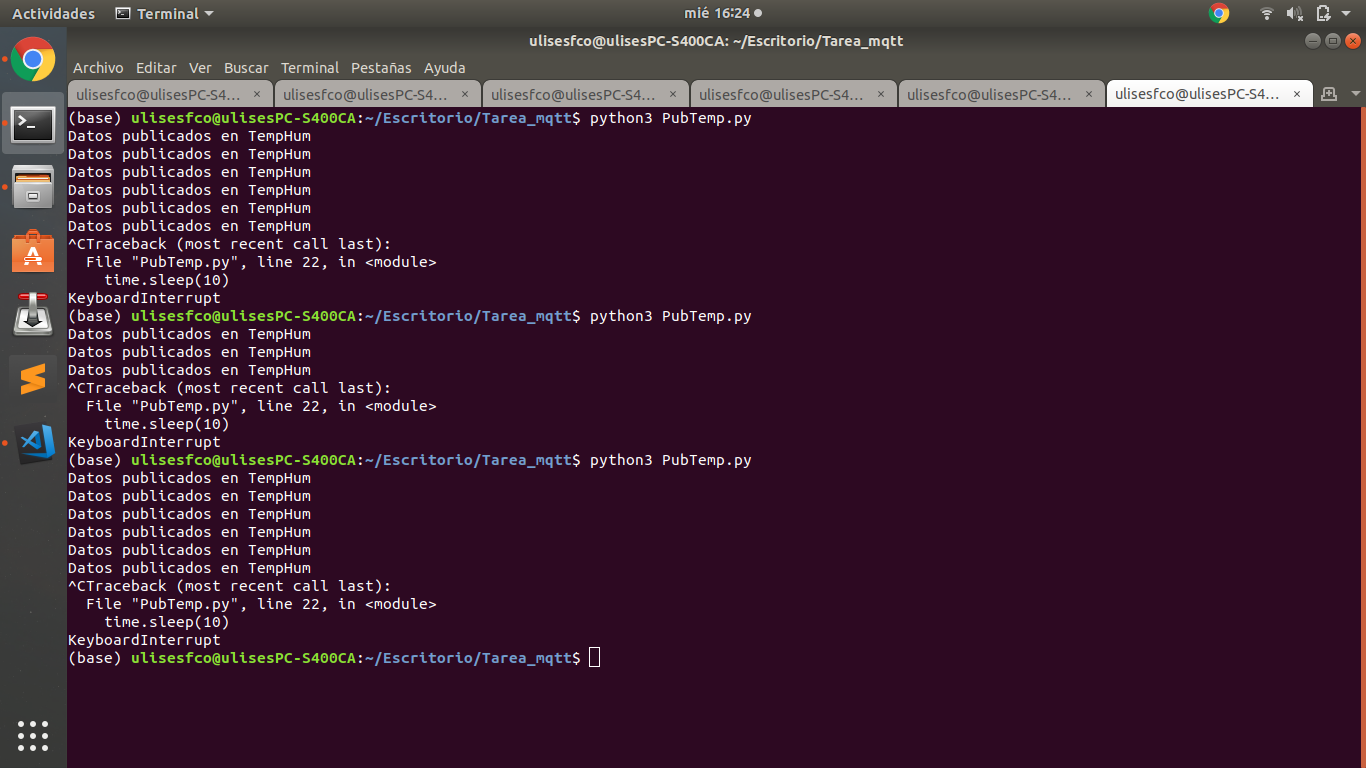
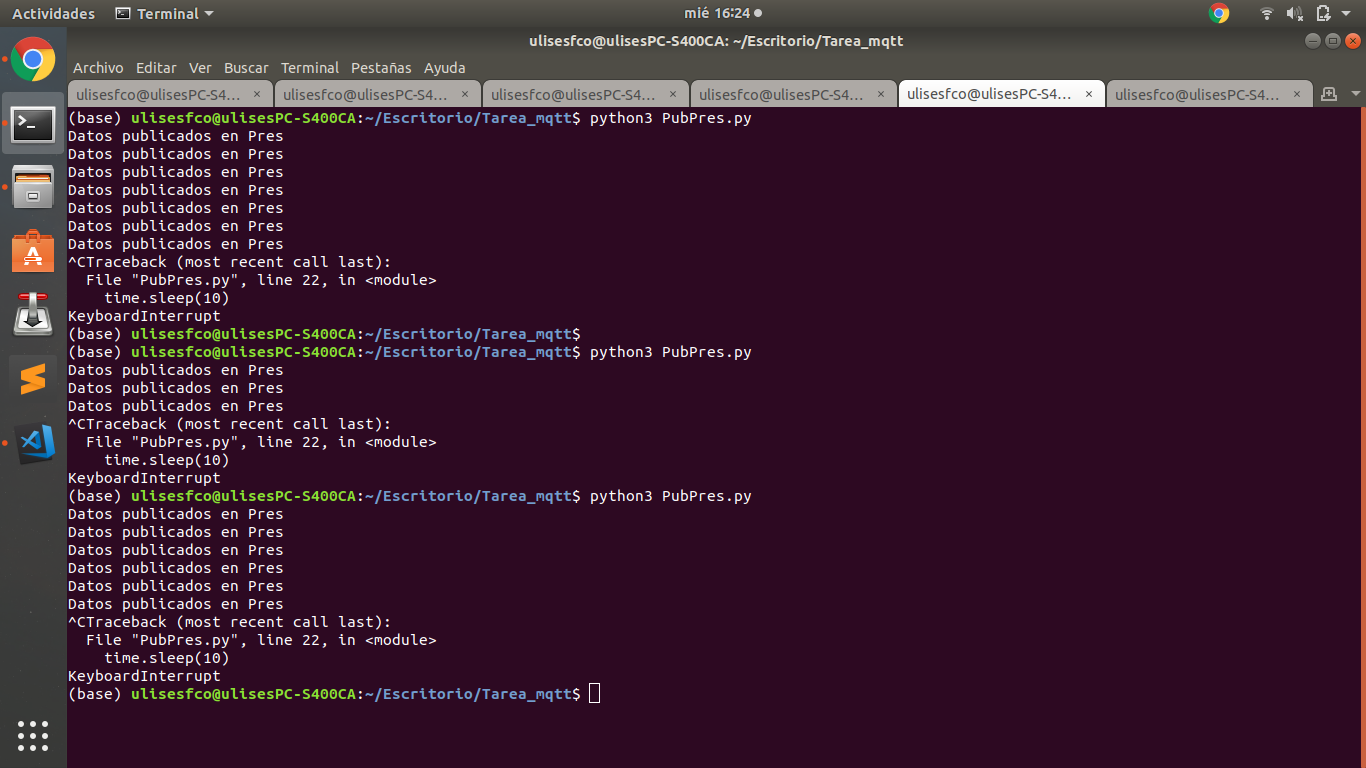
Los cambios principales a ambos tipos de clientes fueron los correspondientes al client\_id, host usado, tópico al que se suscribe o se publica, puerto usado y los datos que se publican en los tópicos respectivos a cada sensor. Cada cliente es un script de python distinto.

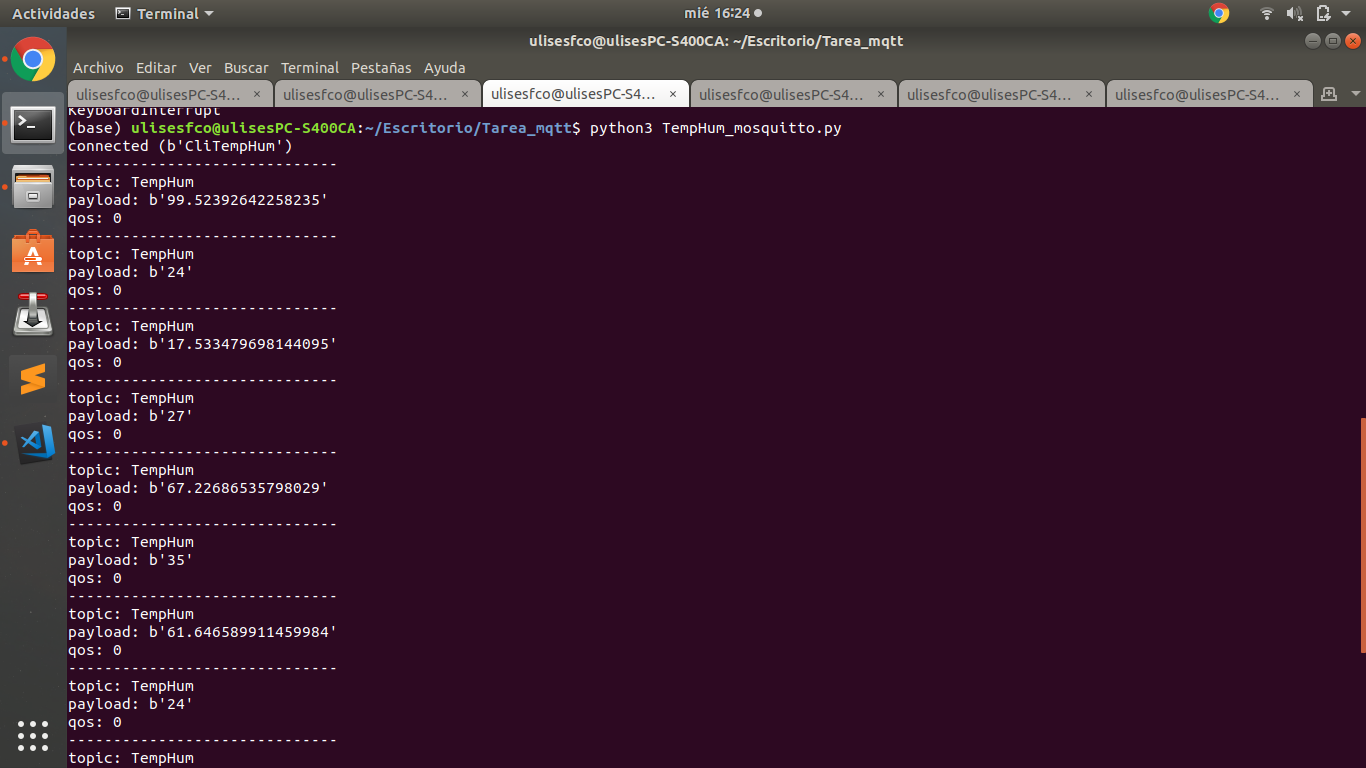
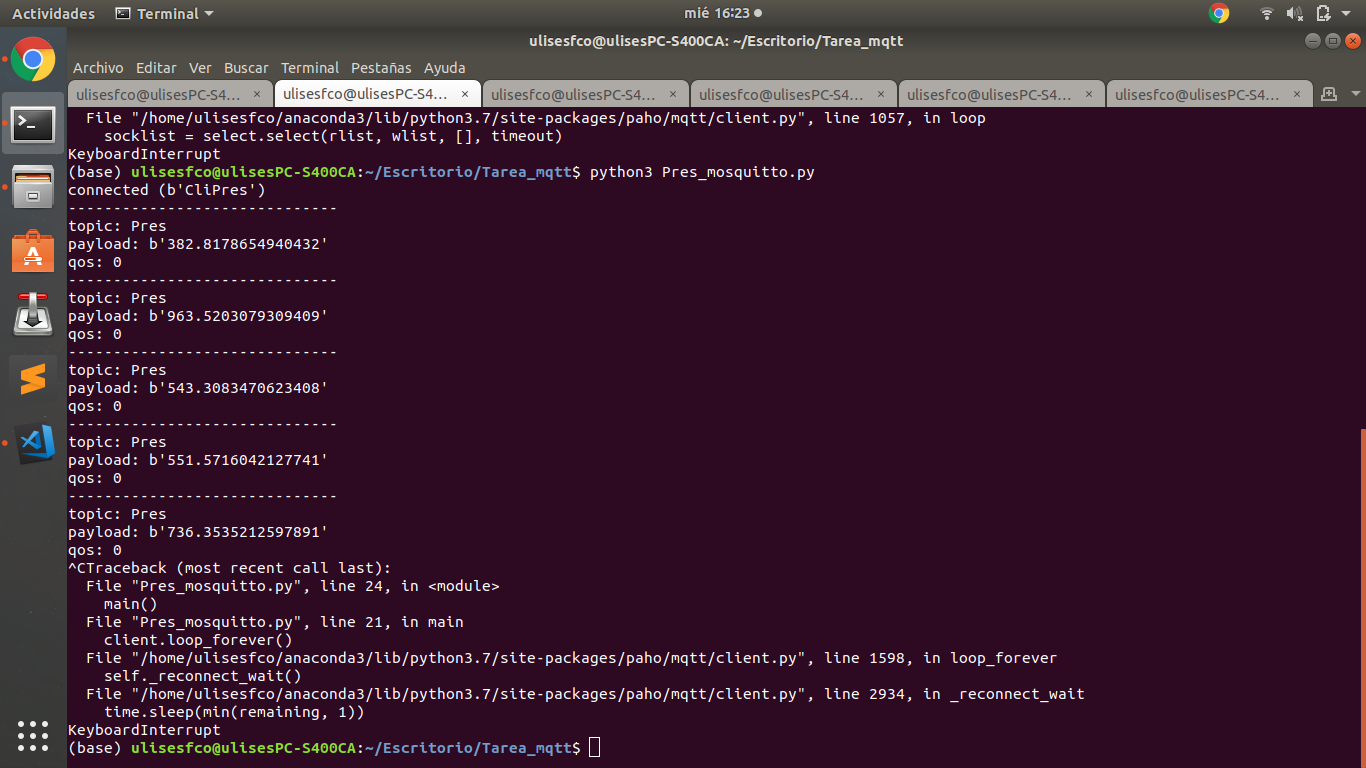
**Configuración y pruebas del sistema**

Los clientes se probaron en una computadora con Linux: ejecutando los 5 scripts desde la terminal, al igual que el broker mosquitto. Para ejecutar los scripts primero ; se inició el broker mosquitto en el puerto 1884 con el comando *mosquitto -p 1884 -v*; luego, en una terminal aparte, se cambió el directorio activo al lugar en donde se encontraban los archivos con el comando *cd <directorio>*; finalmente, se ejecutaron los 5 clientes, cada uno en una ventana de terminal distinta, con el comando *python3 <nombre de cliente>*.

Las siguiente imágenes muestran a los clientes y el broker en funcionamiento:







Igualmente, en el siguiente enlace se encuentra un video que muestra el funcionamiento de los clientes: <https://drive.google.com/file/d/1WGXXPxxyKK5jMUTOlXNWijfghQioHrVU/view?usp=sharing>

**Conclusiones**

Con esta tarea de IoT con MQTT se observó el funcionamiento de este protocolo para establecer un flujo de información entre un cliente (suscriptor) y un servidor (publicador). Para este ejercicio se simuló con valores aleatorios 3 sensores (temperatura, humedad y presión) y con ayuda de la plataforma Losant y librerías mqtt en python se integró una solución funcional de manera local. concluimos que este protocolo es muy útil para aplicaciones IoT con un sistema distribuido debido a que te permite implementar de manera rápida un servicio de multicast y establecer un flujo de información de muchos sensores.

**Referencias**

* <https://www.eclipse.org/paho/clients/python/docs/>
* <https://www.albertcoronado.com/2018/07/07/iot-con-mqtt-mosquitto-python/><http://www.steves-internet-guide.com/publishing-messages-mqtt-client/>
* <https://docs.losant.com/mqtt/overview/>
* <https://www.ibm.com/developerworks/ssa/library/iot-mqtt-why-good-for-iot/index.html>