3️⃣

INFORME 3 - Enrique Sopeña urbano

Pregunta 1

Al ejecutar el código `sisub.py`, el cliente MQTT se conecta correctamente al broker público `test.mosquitto.org` y se suscribe al topic `$SYS/#`.

La salida muestra una secuencia continua de mensajes del sistema, como número de clientes conectados, bytes enviados o recibidos, y estados de conexión de distintos clientes o brokers enlazados.

Estos mensajes provienen del canal de administración `$SYS/#`, que publica periódicamente información interna del broker.

En conclusión, el código funciona correctamente y permite observar en tiempo real el estado y actividad del servidor MQTT, actuando como un **monitor del broker**.

Resultado

```
(venv) root@73cb0c9cf0d8:/var/persist/practica3# python3 sisub.py
Successfully connected to test.mosquitto.org port: 1883
Flags: ConnectFlags(session_present=False)
Properties: []
Subscribed: 1 [ReasonCode(Suback, 'Granted QoS 0')]
$SYS/broker/bytes/received 0 1165934037123
$SYS/broker/bytes/sent 0 28279982879732
$SYS/broker/clients/active 0 11951
```

Pregunta 2

Al ejecutar el código `sipub.py`, el cliente MQTT se conecta al broker público `test.mosquitto.org` y comienza a publicar cada 5 segundos un valor aleatorio (entre

0 y 100) en el tópico `PMtest/rndvalue`.

En la consola se muestran los mensajes enviados junto con su identificador (`mid`), confirmando que cada publicación ha sido procesada correctamente por el broker.

Si en paralelo se ejecuta un suscriptor (`sisub.py`), no se observa la recepcion de los mensajes producidos por el publicador porque no esta suscrito al tópico `PMtest/rndvalue` .

En conclusión, el programa `sipub.py` genera un flujo periódico de datos y demuestra cómo un cliente MQTT puede **publicar información de forma asíncrona y continua** en un broker remoto.

Resultado

Terminal del publicador

```
(venv) root@e4f891c19ca4:/var/persist/practica3# python sipub.py
publishing: 55
mid: 1
Successfully connected to test.mosquitto.org port: 1883
Flags: ConnectFlags(session_present=False)
Properties: []
publishing: 44
mid: 2
publishing: 15
mid: 3
publishing: 70
```

Terminal del suscriptor

```
(venv) root@73cb0c9cf0d8:/var/persist/practica3# python3 sisub.py
Successfully connected to test.mosquitto.org port: 1883
Flags: ConnectFlags(session_present=False)
Properties: []
Subscribed: 1 [ReasonCode(Suback, 'Granted QoS 0')]
$SYS/broker/bytes/received 0 1165934037123
$SYS/broker/bytes/sent 0 28279982879732
$SYS/broker/clients/active 0 11951
```

Pregunta 3

Se modificó el código `sisub.py` para que en lugar de suscribirse al topic `$SYS/#` —que solo muestra información interna del broker—, escuche los mensajes publicados por el productor `sipub.py`.

En concreto, se cambió la línea:

```
THE_TOPIC = "$SYS/#"
```

por:

```
THE_TOPIC = "PMtest/rndvalue"
```

con el objetivo de recibir todos los mensajes publicados bajo el topic `PMtest/rndvalue`.

Tras los cambios, el suscriptor recibe correctamente los valores aleatorios enviados por `sipub.py`, confirmando la comunicación entre ambos programas mediante el broker público

MQTT, aunque los valores recibidos por el suscriptor aparecen en un orden distinto al de los publicados.

Esto ocurre porque el código utiliza el nivel de calidad de servicio **QoS 0**, que corresponde a una entrega “best effort” (*como máximo una vez*).

En este modo, el broker MQTT no garantiza el orden ni la confirmación de entrega de los mensajes, por lo que algunos pueden llegar fuera de orden o incluso perderse.

Resultado

Terminal del publicador

```
(venv) root@e4f891c19ca4:/var/persist/practica3# python sipub.py
publishing: 55
mid: 1
Successfully connected to test.mosquitto.org port: 1883
Flags: ConnectFlags(session_present=False)
Properties: []
publishing: 44
mid: 2
```

```
publishing: 15
mid: 3
publishing: 77
```

Terminal del suscriptor

```
(venv) root@73cb0c9cf0d8:/var/persist/practica3# python3 sisub.py
Successfully connected to test.mosquitto.org port: 1883
Flags: ConnectFlags(session_present=False)
Properties: []
Subscribed: 1 [ReasonCode(Suback, 'Granted QoS 0')]
PMtest/rndvalue 0 44
PMtest/rndvalue 0 77
PMtest/rndvalue 0 15
```

Pregunta 4

Al ejecutar el suscriptor configurado con el tópico `Spain/Valencia/UPV` y el publicador con el tópico `spain/valencia/upv`, ambos clientes se conectan correctamente al broker público (`test.mosquitto.org`), pero **el suscriptor no recibe ningún mensaje**.

Aunque el publicador muestra que los mensajes se envían correctamente, en el terminal del suscriptor no aparece ninguna recepción.

Esto ocurre porque en MQTT los **tópicos son sensibles a mayúsculas y minúsculas** (*case-sensitive*).

Por tanto, `Spain/Valencia/UPV` y `spain/valencia/upv` son considerados **dos tópicos completamente diferentes** por el broker.

En consecuencia, los mensajes publicados en el tópico `spain/valencia/upv` no coinciden con la suscripción del cliente que escucha en `Spain/Valencia/UPV`, y por eso **no se recibe ningún dato**.

Resultado

Terminal del publicador

```
(venv) root@e4f891c19ca4:/var/persist/practica3# python sipub.py
publishing: hola desde el publicador
```

```
mid: 1
Successfully connected to test.mosquitto.org port: 1883
Flags: ConnectFlags(session_present=False)
Properties: []
publishing: hola desde el publicador
mid: 2
publishing: hola desde el publicador
mid: 3
```

Terminal del suscriptor

```
(venv) root@73cb0c9cf0d8:/var/persist/practica3# python3 sisub.py
Successfully connected to test.mosquitto.org port: 1883
Flags: ConnectFlags(session_present=False)
Properties: []
Subscribed: 1 [ReasonCode(Suback, 'Granted QoS 0')]
Spain/Valencia/UPV 0 HOLAAAA
```

Pregunta 5

Se realizaron tres pruebas para analizar el comportamiento del parámetro `retain` en MQTT:

- **Caso 1 (`retain=False`):** El suscriptor no recibe nada, ya que el mensaje no se almacena en el broker y solo se entrega a clientes conectados en ese momento.
- **Caso 2 (`retain=True`):** El suscriptor recibe el mensaje inmediatamente al conectarse, incluso si el publicador ya ha terminado su ejecución. El broker conserva una copia del último mensaje publicado en ese tópico.
- **Caso 3 (varios mensajes con `retain=True`):** El suscriptor solo recibe el **último mensaje retenido**, porque el broker guarda únicamente un mensaje por tópico, sobrescribiendo los anteriores.

En conclusión, la opción `retain=True` permite que el último mensaje de un tópico quede disponible para nuevos suscriptores, mientras que con `retain=False` los mensajes no se conservan una vez enviados.

Resultado

1. **Caso 1:**

   Terminal del publicador

   ```
   (venv) root@e4f891c19ca4:/var/persist/practica3# python sipub.py
   publishing: hola desde el publicador, mensaje numero 0
   mid: 1
   Successfully connected to test.mosquitto.org port: 1883
   Flags: ConnectFlags(session_present=False)
   Properties: []
   publishing: hola desde el publicador, mensaje numero 1
   mid: 2
   publishing: hola desde el publicador, mensaje numero 2
   mid: 3
   publishing: hola desde el publicador, mensaje numero 3
   mid: 4
   publishing: hola desde el publicador, mensaje numero 4
   mid: 5
   publishing: hola desde el publicador, mensaje numero 5
   mid: 6
   publishing: hola desde el publicador, mensaje numero 6
   ```

   Terminal del suscriptor

   ```
   (venv) root@73cb0c9cf0d8:/var/persist/practica3# python3 sisub.py
   Successfully connected to test.mosquitto.org port: 1883
   Flags: ConnectFlags(session_present=False)
   Properties: []
   Subscribed: 1 [ReasonCode(Suback, 'Granted QoS 0')]
   Spain/Valencia/UPV 0 mensaje_prueba3
   Spain/Valencia/UPV 0 hola desde el publicador, mensaje numero 4
   Spain/Valencia/UPV 0 hola desde el publicador, mensaje numero 5
   Spain/Valencia/UPV 0 hola desde el publicador, mensaje numero 6
   ```

2. **Caso 2:**

   Terminal del publicador

   ```
   (venv) root@e4f891c19ca4:/var/persist/practica3# python sipub2.py
   publishing: hola desde el publicador, mensaje numero 0
   ```

```
mid: 1
Successfully connected to test.mosquitto.org port: 1883
Flags: ConnectFlags(session_present=False)
Properties: []
publishing: hola desde el publicador, mensaje numero 1
mid: 2
^CStopped by user.
```

Terminal del suscriptor

```
(venv) root@73cb0c9cf0d8:/var/persist/practica3# python3 sisub.py
Successfully connected to test.mosquitto.org port: 1883
Flags: ConnectFlags(session_present=False)
Properties: []
Subscribed: 1 [ReasonCode(Suback, 'Granted QoS 0')]
Spain/Valencia/UPV 0 hola desde el publicador, mensaje numero 1
```

3. **Caso 3:**

   Terminal del publicador

```
(venv) root@e4f891c19ca4:/var/persist/practica3# python sipub2.py
publishing: hola desde el publicador, mensaje numero 0
mid: 1
mid: 2
mid: 3
Successfully connected to test.mosquitto.org port: 1883
Flags: ConnectFlags(session_present=False)
Properties: []
publishing: hola desde el publicador, mensaje numero 1
mid: 4
mid: 5
mid: 6
publishing: hola desde el publicador, mensaje numero 2
mid: 7
mid: 8
mid: 9
publishing: hola desde el publicador, mensaje numero 3
mid: 10
```

```
mid: 11
mid: 12
publishing: hola desde el publicador, mensaje numero 4
mid: 13
mid: 14
mid: 15
publishing: hola desde el publicador, mensaje numero 5
mid: 16
mid: 17
mid: 18
^CStopped by user.
```

Terminal del suscriptor

```
(venv) root@73cb0c9cf0d8:/var/persist/practica3# python3 sisub.py
Successfully connected to test.mosquitto.org port: 1883
Flags: ConnectFlags(session_present=False)
Properties: []
Subscribed: 1 [ReasonCode(Suback, 'Granted QoS 0')]
Spain/Valencia/UPV 0 hola desde el publicador, mensaje numero 5
```

Pregunta 6

Se implementó una aplicación de chat simple en Python utilizando el protocolo MQTT y la librería `paho-mqtt`.

Todos los usuarios se conectan al mismo tópico (`upv/chat/grupo1`) y publican sus mensajes con `retain=False`.

Cada mensaje publicado es reenviado por el broker a todos los clientes suscritos, logrando comunicación bidireccional en tiempo real.

La lectura del texto se realiza con `input()`, y se usa `threading` implícitamente mediante `loop_start()` para gestionar la escucha de mensajes mientras el usuario escribe.

Los mensajes retenidos no se emplean, garantizando que solo los usuarios conectados al grupo reciban los mensajes en tiempo real, cumpliendo el requisito de que “solo los miembros del grupo” los vean.

Pregunta 7

Se desarrolló una aplicación en Python que se conecta al broker MQTT de **The Things Network (TTN)** para leer los datos publicados por los dispositivos LoRaWAN.

Se utilizaron los siguientes parámetros de conexión:

- **Broker:** eu1.cloud.thethings.network
- **Usuario:** lopys2ttn@ttn
- **Contraseña:** *token de acceso proporcionado por TTN*
- **Tópico:** `v3/+/devices/#` (permite escuchar todos los dispositivos de cualquier aplicación asociada).

El programa utiliza la librería `paho-mqtt` y se suscribe al tópico indicado.

Cada vez que se recibe un mensaje, intenta decodificarlo como JSON y muestra su contenido completo por pantalla. Si el mensaje no tiene formato JSON válido, se imprime el contenido en texto plano.

Este script permite visualizar en tiempo real los mensajes *uplink* publicados por TTN en formato JSON, mostrando toda la información enviada por los dispositivos conectados a la red LoRaWAN.