

**1) ¿Qué respuesta recibe? ¿Qué información le da la respuesta detallada?**

```
Terminal
asus@asus-K53SC ~ $ mosquitto_pub -h 127.0.0.1 -t "Test" -m "Hola Mundo"
asus@asus-K53SC ~ $
```

```
Terminal
^C
asus@asus-K53SC ~ $ mosquitto_sub -d -h 127.0.0.1 -t "Test"
Received CONNACK
Received SUBACK
Subscribed (mid: 1): 0
Received PUBLISH (d0, q0, r0, m0, 'Test', ... (10 bytes))
Hola Mundo
Sending PINGREQ
Received PINGRESP
Received PUBLISH (d0, q0, r0, m0, 'Test', ... (10 bytes))
Hola Mundo
```

**2) Pruebe hacer lo mismo utilizando un dispositivo embebido como cliente. ¿Se presentó alguna dificultad?**

No presentó dificultad fue totalmente transparente y sencilla.

**3) Utilizando un software sniffer de red (por ejemplo wireshark), visualice el flujo de datos capturados. ¿Cómo es la comunicación? ¿Cuántos paquetes son necesarios y por qué? ¿Cuántos partes intervinientes hay en la comunicación (ojo, recuerde modelo pub/sub)? ¿Cómo es la trama del paquete que lleva el mensaje? ¿Cuántos bytes mide el paquete (recuerde: se le llama “mosquitto”)?**

La comunicación entre los dispositivos intervinientes en el protocolo de mensajes al ser orientada a la conexión (TCP) genera un excedente de paquetes utilizados para emanar un simple mensaje en el modelo pub/sub.

Esto lo podemos visualizar con la herramienta wireshark que nos provee de los datos numéricos para conocer que la trama de paquetes intervinientes son más que el paquete de datos, por ejemplo en la figura 3, el “hello world” genera un mensaje de 1344 bits, donde, de los datos completos emanados representa el 60% (816 bits), el porcentaje restante es cabeceras propias del protocolo.

Capturing from eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: `ip.src==192.168.11.11` Expression... Clear Apply Guardar

No.	Time	Source	Destination	Protocol	Length	Info
1600	791.538639000	192.168.11.11	192.168.11.39	TCP	68	ibm-mqisdp > 40608 [PSH, ACK] Seq=206 Ack=29 Win=227 Len=2 TSva
1853	851.600949000	192.168.11.11	192.168.11.39	TCP	68	ibm-mqisdp > 40608 [PSH, ACK] Seq=208 Ack=31 Win=227 Len=2 TSva
2011	911.661711000	192.168.11.11	192.168.11.39	TCP	68	ibm-mqisdp > 40608 [PSH, ACK] Seq=210 Ack=33 Win=227 Len=2 TSva
2164	971.722853000	192.168.11.11	192.168.11.39	TCP	68	ibm-mqisdp > 40608 [PSH, ACK] Seq=212 Ack=35 Win=227 Len=2 TSva
2351	1031.786109000	192.168.11.11	192.168.11.39	TCP	68	ibm-mqisdp > 40608 [PSH, ACK] Seq=214 Ack=37 Win=227 Len=2 TSva
2527	1091.848519000	192.168.11.11	192.168.11.39	TCP	68	ibm-mqisdp > 40608 [PSH, ACK] Seq=216 Ack=39 Win=227 Len=2 TSva
2729	1151.911811000	192.168.11.11	192.168.11.39	TCP	68	ibm-mqisdp > 40608 [PSH, ACK] Seq=218 Ack=41 Win=227 Len=2 TSva
2880	1211.975434000	192.168.11.11	192.168.11.39	TCP	68	ibm-mqisdp > 40608 [PSH, ACK] Seq=220 Ack=43 Win=227 Len=2 TSva
3008	1272.039983000	192.168.11.11	192.168.11.39	TCP	68	ibm-mqisdp > 40608 [PSH, ACK] Seq=222 Ack=45 Win=227 Len=2 TSva
3115	1332.101855000	192.168.11.11	192.168.11.39	TCP	68	ibm-mqisdp > 40608 [PSH, ACK] Seq=224 Ack=47 Win=227 Len=2 TSva
3228	1392.166925000	192.168.11.11	192.168.11.39	TCP	68	ibm-mqisdp > 40608 [PSH, ACK] Seq=226 Ack=49 Win=227 Len=2 TSva
3319	1452.229641000	192.168.11.11	192.168.11.39	TCP	68	ibm-mqisdp > 40608 [PSH, ACK] Seq=228 Ack=51 Win=227 Len=2 TSva

Frame 1005: 168 bytes on wire (1344 bits), 168 bytes captured (1344 bits) on interface 0

Ethernet II, Src: TexasIns\_be:68:27 (54:4a:16:be:68:27), Dst: AsustekC\_04:1d:f1 (54:04:a6:04:1d:f1)

Internet Protocol Version 4, Src: 192.168.11.11 (192.168.11.11), Dst: 192.168.11.39 (192.168.11.39)

Transmission Control Protocol, Src Port: ibm-mqisdp (1883), Dst Port: 40608 (40608), Seq: 96, Ack: 19, Len: 102

Data (102 bytes)

Data: 3064000474656d706d616e647361736461c3b1646b6173c3...

0000 54 04 a6 04 1d f1 54 4a 16 be 68 27 08 00 45 00 T.....TJ..h'..E.  
0010 00 9a ab 98 40 00 40 06 f7 42 c0 a8 0b 0b c0 a8 ...@.@..B.....  
0020 0b 27 07 5b 9e a0 59 f1 c5 a9 23 94 07 2c 80 18 '.[.Y. .#.,.,.  
0030 00 e3 a1 fe 00 00 01 01 08 0a 00 54 73 61 00 36 .....Tsa.6  
0040 4e da 30 64 00 04 74 65 6d 70 6d 61 6e 64 73 61 N.0d..te mpmandsa  
0050 73 64 61 c3 b1 64 6b 61 73 c3 b1 6c 64 6b 61 73 sda..dka s..ldkas  
0060 6c 64 6b 61 73 c3 b1 6b 64 c3 b1 61 73 64 6b c3 ldkas..k d..asdk.  
0070 b1 61 73 6c 6b 64 c3 b1 73 6c 61 64 6b c3 b1 6c .aslkd..sladk..l  
0080 61 73 6b 64 c3 b1 6c 6b 61 73 c3 b1 64 6b 61 73 askd..lk as..dkas  
0090 64 c3 b1 6c 6b 61 73 64 61 73 64 61 73 61 64 61 d..lkasd asdasada  
00a0 73 70 61 76 61 64 61 73 spavadas

eth0: <live capture in progress> File: /tmp/... Packets: 4274 - Displayed: 31 (0,7%) Profile: Default

Figura 3

4)1 Siguiendo los pasos propuestos por

<https://geekytheory.com/tutorialraspberrypigpioymqttnparte1/>, trate de prender y apagar un led en una placa con linux embebido publicando la orden desde su PC (el servidor puede seguir estando en la PC) Explique brevemente cómo funciona este escenario.

**Problema:** Prender led remoto por protocolo de mensajes.

**Solución:**

- Al momento de controlar un LED, nos suscribimos desde la Galileo al topic = led.
  - `mosquitto_sub -h 192.168.0.103 -t "led"`
- Configuramos el GPIO 3 como salida para comandar el led interno de la placa.
  - `echo -n "3" > /sys/class/gpio/export`
  - `echo -n "out" > /sys/class/gpio/gpio3/direction`
- Si enviamos desde la notebook '1' o '0' nos aparecerá en el terminal de la Galileo dicho mensaje, para lo cual se lo direccionamos como flag de on/off al GPIO 3
  - `mosquitto_sub -h 192.168.0.103 -t "led" > /sys/class/gpio/gpio3/value`
  - El operador '>' lo que hace es redireccionar lo que devolvería el comando por pantalla a otro fichero, en nuestro caso el fichero "value".

Ahora desde la terminal del PC, publicamos el mensaje para comandar el led:

- `mosquitto_pub -h 192.168.112.129 -t "GPIO" -m "1" // endendido`
- `mosquitto_pub -h 192.168.112.129 -t "GPIO" -m "0" // apagado`

5) ¿Cómo explicaría el funcionamiento de estos códigos? (Ver

<https://pypi.python.org/pypi/pahomqtt/1.1>)

El código del archivo "llamado paho\_mqtt\_pub.py" genera un publisher que emana cada dos segundos un hello world, al topic "hello/world", los subscriber al topic "hello/world" tienden a consumir el mensaje provisto .

6) Opcional: En vuestro hogar pruebe cambiar la dirección del broker por "iot.eclipse.org".