UNIVERSIDAD POLITÉCNICA DE MADRID

*Departamento de Ingeniería de Sistemas Telemáticos*
*Universidad Politécnica de Madrid*

**CLIENTE MQTT**

# Introducción

En esta práctica[1][2] vamos a crear un cliente MQTT `client.py` que enviará constantes vitales de un Sensor. Para ello se proporciona el programa `reporter.sh` (disponible en Moodle) que va añadiendo líneas a un archivo `report.csv` que almacena las constantes vitales de un sensor.

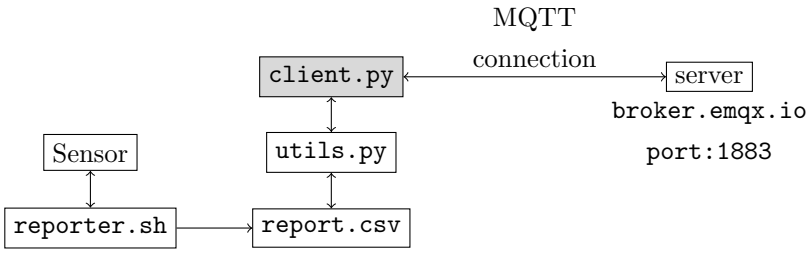


Figura 1: Escenario de la práctica.

# Instalación de Dependencias

Antes de comenzar la práctica debemos instalar las dependencias necesarias para python y MQTT. Descargue el fichero `practica-cliente.zip` de Moodle y descomprímalo en su carpeta personal. Abra una terminal y ejecute las siguientes líneas

```
$ cd ~/practica-cliente
$ ./dependencies.sh
```

[1]Todas las preguntas tienen el mismo valor/puntuación. Cada apartado del Problema 2 tiene el mismo valor que el resto de Problemas.

## Generación de Constantes Vitales

Para generar el archivo `report.csv` con las constantes vitales, ejecute el programa que reporta las medidas del sensor:

```
$ cd ~/practica-cliente
$ ./reporter.sh
```

Tras ejecutar las líneas de arriba, verá que se ha generado un archivo `report.csv` con las siguientes columnas (separadas por comas):

```
Idx, Time (s), HR (BPM), RESP (BPM), SpO2 (%),TEMP (*C),OUTPUT
0,0,94,21,97,36.2,Normal
1,1,94,25,97,36.2,Normal
2,2,101,25,93,38,Abnormal
3,3,55,11,100,35,Abnormal
```

*Nota*: el archivo `report.csv` se genera de cero para cada ejecución del el `reporter.sh`.

## Lectura de Constantes Vitales

A continuación va a programar la función `read_report()` encargada de leer constantes vitales del reporte. Esta función se encuentra en el archivo `utils.py` y se encarga de leer el archivo de reporte `report.csv`. Por ejemplo, podemos empezar en la tercera línea del archivo y leer dos líneas usando

```
$ python3
>>> import utils
>>> utils.read_report('report.csv',10,2)
['9,9,94,26,97,29,Normal\n', '10,10,94,26,97,42,Abnormal\n']
```

Para más detalles sobre el funcionamiento de `read_report()`, lea la descripción de la función en el archivo `utils.py`. Tendrá que utilizar las funciones `open()` y `read_lines()` para programar la función.

**Problema** 1. Programe la función `read_report()` y ejecútela usando como parámetros `LAST_LINE=`$X$ y `num_lines=`$\lceil \frac{X}{2} \rceil$, donde $X$ es el número de grupo asignado por el profesor. Copie el resultado en el campo `lectura` del archivo `respuestas-X.json` usando dobles comillas:

```
{
    "lectura": [ "9,9,94,26,97,29,Normal\n",
      "10,10,94,26,97,42,Abnormal\n"]
}
```

*Nota*: deje corriendo `reporter.sh` para que genere suficientes reportes.

# Cliente MQTT

## Conexión

A continuación va a programar parte de la lógica del cliente MQTT. En concreto se va a programar la conexión del `client.py` con el server ilustrada en la Figura. 1, donde se especifica la dirección y puerto del servidor.

Para la comunicación con el servidor mediante MQTT, el programa `client.py` utiliza la librería `PAHO` de python. Los pasos para abrir una conexión (consulte la plantilla del cliente `client.py`) son los siguientes:

1. Conectarse con el servidor MQTT: función `connect_mqtt()`; e
2. invocar el bucle de publicación de mensajes: función `publish()`.

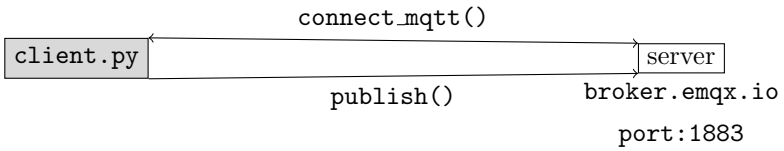


Figura 2: Conexión y publicación del cliente.

Modifique la plantilla de `client.py` para conectarse al servidor y responda a las siguientes preguntas.

**Problema** 2. Inicie una captura wireshark en todas las interfaces (`any`) poniendo filtro "mqtt". Ejecute en la consola `client.py` e identifique las tramas de conexión y ACK en wireshark. Detenga la captura y rellene las siguientes preguntas en el JSON de respuestas:

*a*) ¿Cuál es el ID de su cliente?

*b*) En la captura wireshark, ¿cuál es el tipo de mensaje MQTT del ACK de la conexión?

Responda rellenando lso campos `clientid` y `ackmessagetype` del JSON de respuestas, respectivamente.
Guarde[3] la captura en el fichero `captura-connect-grupoX.pcapng`. Siga el mismo procedimiento del pie de página para todas las capturas de la práctica.

[3]Para guardas solo los paquetes MQTT basta con poner en el filtro "mqtt", pinchar en File-Export Specified Packets, y en el cuadro "Packet Range" pinchar en la bola "Displayed".

## Publicación

A continuación debe modificar el `client.py` para leer los datos del sensor (archivo `report.csv`) y publicarlos en el topic `rserGX/vitals`, donde X es su número de grupo.

La función `publish()` contiene un bucle que continuamente publica mensajes MQTT. Modifique el bucle para que espere $(X \mod 5)+5$ segundos al final de cada iteración. Además, modifique `publish()` para invocar a `read_report()` y publicar *una a una* las constantes vitales.

Tras haber realizado las modificaciones indicadas, inicie una captura en wireshark filtrando paquetes MQTT y responda a las siguientes preguntas usando la captura entregada. Para ello responda en en el JSON de respuestas a:

**Problema** 3. ¿En qué instante se envía la muestra $X$? (valor "Time" en wireshark).

**Problema** 4. ¿Cuánto tiempo pasa entre dos Ping de MQTT?

Responda rellenando lso campos `instantemuestraX` (no sustituya la X por su número de grupo) y `tiempopings` del JSON de respuestas, respectivamente.

Guarde la captura en el fichero `captura-publish-grupoX.pcapng`.

## Desconexión y QoS

A continuación vamos a probar los distintos niveles de QoS ofrecidos por MQTT. Para ello basta con añadir el argumento `qos=`$q$ a la función `client.publish()`, donde $q \in \{0, 1, 2\}$ especifica el nivel QoS de MQTT.

En primer lugar vamos a probar qué sucede con un Qos de 0 si se cae la conexión. Para ello vamos a apagar la interfaz `eth0` usando el siguiente comando

```
$ ip link set down dev eth0
```

y la encenderemos con el siguiente comando

```
$ ip link set up dev eth0
```

## QoS 0

Ponga un QoS 0 y ejecute el `client.py` usando el siguiente comando para guardar la salida:

```
$ python3 client.py 2>&1 | tee /tmp/qos0-grupoX.log
```

Inicie una captura en wireshark filtrando tráfico MQTT y espere a que se envíen, al menos, dos tramas PUBLISH. Apague la interfaz `eth0` y espere a que el cliente imprima "Failed to send message [...]". Vuelva a encender la interfaz.

Responda a las siguientes preguntas en el JSON de respuestas:

**Problema** 5. Especifique el `Idx` de las muestras identificadas como perdidas por el cliente.

**Problema** 6. Especifique el `Idx` de las muestras que no se han enviado con éxito.

Responda rellenando los campos `perdidascliente` y `perdidas` del JSON de respuestas, respectivamente.

Detenga la captura de wireshark y guárdela en el fichero `qos0-grupoX.pcapng`.

### QoS 1

Ponga un QoS 1 y ejecute el `client.py` como en el apartado anterior, esta vez guardando el log en el archivo `qos1-grupoX.log`.

De nuevo, inicie una captura en wireshark filtrando tráfico MQTT y espere a que se envíen, al menos, dos tramas PUBLISH. Apague la interfaz `eth0` y espere de nuevo a que el cliente imprima "Failed to send message [...]". Vuelva a encender la interfaz.

**Problema** 7. Indique el `Message Identifier` de los mensajes duplicados.

**Problema** 8. Repita otra captura incrementando el `keepalive` al doble e indique el `Message Identifier` de los mensajes perdidos.

Responda rellenando los campos `duplicates` y `duplicatesx2` del JSON de respuestas, respectivamente.

Guarde ambas capturas en los ficheros `qos1-grupoX.pcapng` y `qos1-x2keepalive-grupoX.pcapng`.

## Entrega

Se subirá a moodle un archivo `clienteX.zip` (con `X` el número de grupo) que contenga:

1. el cliente `client.py`;
2. el archivo `utils.py`;
3. el JSON de respuestas `respuestas-X.json`;
4. las trazas de Wireskark `captura-connect-grupoX.pcapng`, `captura-publish-grupoX.pcapng`, `qos0-grupoX.pcapng`, `qos1-grupoX.pcapng`, `qos1-x2keepalive-grupoX.pcapng`; y

5. los logs `qos0-grupoX.log`, `qos1-grupoX.log`.

**Atención I**: las capturas deben contener *solamente* tráfico MQTT.
**Atención II**: una entrega sin los archivos especificados, o con archivos sin formato especificado tendrá un 0 en los **Problemas** correspondientes.