



ITESO, Universidad
Jesuita de Guadalajara

PRÁCTICA 2

MQTT y la nube

Desarrollo de software de comunicaciones en ambientes embebidos

Instituto Tecnológico y de Estudios Superiores de Occidente
Especialidad en Sistemas Embebidos

14-Marzo-2022

MERCADO, IXCHEL DAYANARA

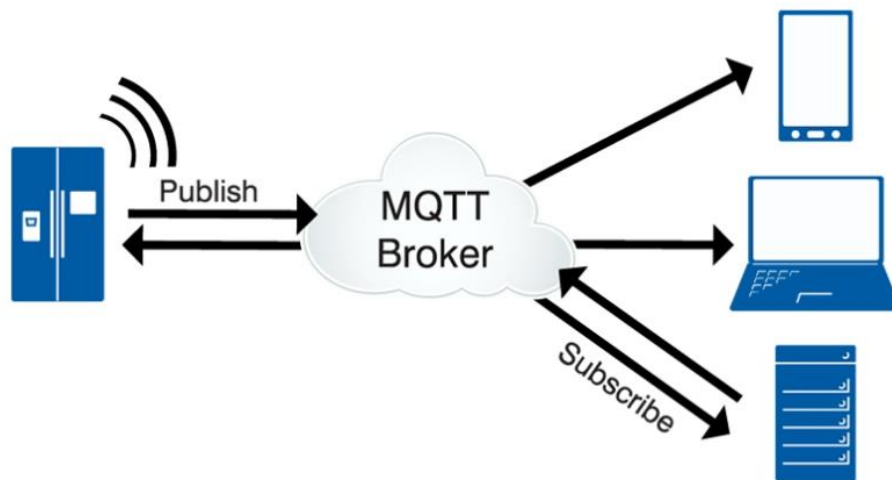
lxchel.mercado@iteso.mx

Índice

Protocolo MQTT.....	2
Estructura de un mensaje MQTT.....	3
Calidad del servicio (QoS)	3
Seguridad en MQTT	3
Implementación de práctica 2.....	4
Requerimientos	4
Implementación	4
Resultados de la implementación	7
Problemas enfrentados durante la implementación	8
Conclusiones.....	8

Protocolo MQTT

MQTT son las siglas MQ Telemetry Transport. Es un protocolo de comunicación M2M (machine-to-machine) de tipo message queue. Está basado en la pila TCP/IP como base para la comunicación. En el caso de MQTT cada conexión se mantiene abierta y se "reutiliza" en cada comunicación. El funcionamiento del MQTT es un servicio de mensajería push con patrón publicador/suscriptor (pub-sub). Para filtrar los mensajes que son enviados a cada cliente los mensajes se disponen en topics organizados jerárquicamente. Un cliente puede publicar un mensaje en un determinado topic. Otros clientes pueden suscribirse a este topic, y el broker le hará llegar los mensajes suscritos.



Los clientes inician una conexión TCP/IP con el broker, el cual mantiene un registro de los clientes conectados. Esta conexión se mantiene abierta hasta que el cliente la finaliza. Por defecto, MQTT emplea el puerto 1883 y el 8883 cuando funciona sobre TLS.

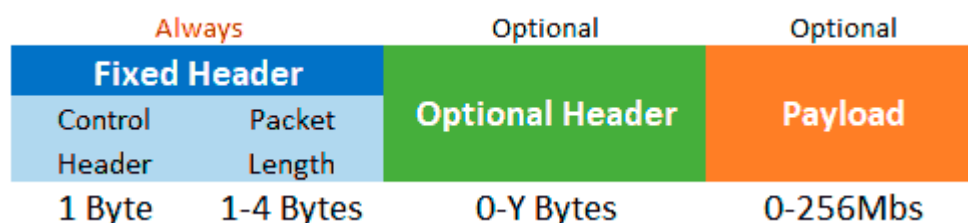
Para ello el cliente envía un mensaje CONNECT que contiene información necesaria (nombre de usuario, contraseña, client-id...). El broker responde con un mensaje CONNACK, que contiene el resultado de la conexión (aceptada, rechazada, etc).

Para enviar los mensajes el cliente emplea mensajes PUBLISH, que contienen el topic y el payload.

Para suscribirse y desuscribirse se emplean mensajes SUBSCRIBE y UNSUBSCRIBE, que el servidor responde con SUBACK y UNSUBACK.

Por otro lado, para asegurar que la conexión está activa los clientes mandan periódicamente un mensaje PINGREQ que es respondido por el servidor con un PINGRESP. Finalmente, el cliente se desconecta enviando un mensaje de DISCONNECT.

Estructura de un mensaje MQTT



- **Cabecera fija:** Ocupa 2 a 5 bytes, obligatorio. Consta de un código de control, que identifica el tipo de mensaje enviado, y de la longitud del mensaje. La longitud se codifica en 1 a 4 bytes, de los cuales se emplean los 7 primeros bits, y el último es un bit de continuidad.
- **Cabecera variable:** Opcional, contiene información adicional que es necesaria en ciertos mensajes o situaciones.
- **Contenido(payload):** Es el contenido real del mensaje. Puede tener un máximo de 256 Mb aunque en implementaciones reales el máximo es de 2 a 4 kB.

Calidad del servicio (QoS)

MQTT dispone de un mecanismo de calidad del servicio o QoS, entendido como la forma de gestionar la robustez del envío de mensajes al cliente ante fallos.

MQTT tiene tres niveles QoS posibles:

- QoS 0 unacknowledged (at most one): El mensaje se envía una única vez. En caso de fallo por lo que puede que alguno no se entregue.
- QoS 1 acknowledged (at least one): El mensaje se envía hasta que se garantiza la entrega. En caso de fallo, el suscriptor puede recibir algún mensaje duplicados.
- QoS 2 assured (exactly one). Se garantiza que cada mensaje se entrega al suscriptor, y únicamente una vez.

Seguridad en MQTT

El protocolo MQTT dispone de distintas medidas de seguridad que podemos adoptar para proteger las comunicaciones.

Esto incluye transporte SSL/TLS y autenticación por usuario y contraseña o mediante certificado. Sin embargo, hay que tener en cuenta que muchos de los dispositivos IoT disponen de escasa capacidad, por lo que el SLL/TLS puede suponer una carga de proceso importante.

En muchos casos, la autenticación consiste en una contraseña y usuario que son enviados como texto plano. Por último, también es posible configurar el broker para aceptar conexiones anónimas.

Implementación de práctica 2

Requerimientos

Los requerimientos de esta práctica eran los siguientes:

1. Implementar un dispositivo conectado a la nube con MQTT.
 - a) El dispositivo debe contar con al menos 3 tópicos (2 poster, 1 subscriber viceversa).
 - b) Se debe de presentar la aplicación funcionando en la tarjeta FRDM-K64 conectada a Internet a través de una red ethernet.
 - c) La aplicación del dispositivo es libre, solo debe respetarse el requerimiento 1a.
 - d) Como servidor en la nube se debe utilizar CloudMQTT.
2. Controlar el dispositivo del requerimiento 1 utilizando una aplicación en un dispositivo móvil. (Android: IoTMQTT Panel).

Implementación

La implementación comenzó con base al programa de ejemplo de NXP para MQTT con FreeRtos. A partir de eso el programa para la práctica debía cumplir con el siguiente concepto elegido: Una aplicación que se ocupe de cuidar tus plantas, regándolas, dándoles luz ultravioleta y checando la humedad del ambiente. A partir de este concepto, comencé la modificación del ejemplo. Lo primero era subscribirse a los 3 tópicos que se iban a usar:

```
static void mqtt_subscribe_topics(mqtt_client_t *client)
{
    static const char *topics[] = {"humidity/#", "water/#", "uvlight/#"};
    int qos[] = {0, 1, 1};
    err_t err;
    int i;
```

Después, implementé la interrupción para el botón 3, y también implementé el GPIO para el led azul, esto con la finalidad de usar el botón para iniciar la acción de publicar en alguno de los tópicos, y el led para reflejar el funcionamiento de la luz ultravioleta.

```
140
141
142 void BOARD_SW_IRQ3_HANDLER(void)
143 {
144     #if (defined(FSL_FEATURE_PORT_HAS_NO_INTERRUPT) && FSL_FEATURE_PORT_HAS_NO_INTERRUPT)
145         /* Clear external interrupt flag. */
146         GPIO_GpioClearInterruptFlags(BOARD_SW_GPIO3, 1U << BOARD_SW_GPIO3_PIN);
147     #else
148         /* Clear external interrupt flag. */
149         GPIO_PortClearInterruptFlags(BOARD_SW_GPIO3, 1U << BOARD_SW_GPIO3_PIN);
150     #endif
151     /* Change state of button. */
152     PRINTF(" %s is pressed \r\n", BOARD_SW_NAME3);
153     g_Button3Press = true;
154     SDK_ISR_EXIT_BARRIER;
155 }
156
```

```

/* Init output LED GPIO. */
GPIO_PinInit(BOARD_LEDB_GPIO, BOARD_LEDB_GPIO_PIN, &led_config);
////////////////////////////////////
PRINTF("LED and Buttons Initialized \n"); //Water for user

```

Una vez teniendo el hardware listo, seguí con la implementación de las funciones que iban a reflejarse según el tópico del que llegara mensaje, una para cada funcionamiento:

```

41
42 void water_the_plants()
43 {
44     static const char *topic = "water/#";
45     static const char *message = "plants were watered";
46
47     mqtt_publish(mqtt_client, topic, message, strlen(message), 1, 0, mqtt_message_published_cb, (void *)topic);
48 }
49 void uv_light_toggle()
50 {
51     static const char *topic = "uvlight/#";
52     static const char *message = "UV light led was toggled";
53     GPIO_PortToggle(BOARD_LEDB_GPIO, 1U << BOARD_LEDB_GPIO_PIN);
54     mqtt_publish(mqtt_client, topic, message, strlen(message), 1, 0, mqtt_message_published_cb, (void *)topic);
55 }
56 void display_humidity()
57 {
58     static const char *topic = "humidity/#";
59     static const char *message = "80% humidity";
60     mqtt_publish(mqtt_client, topic, message, strlen(message), 1, 0, mqtt_message_published_cb, (void *)topic);
61 }
62 #endif

```

Y este código iba a ser llamada a partir del cambio de bandera del botón, activando la función de choose_topic, dónde podrías escoger a cuál tópico quieres publicar y qué funcionamiento quieres activar.

```

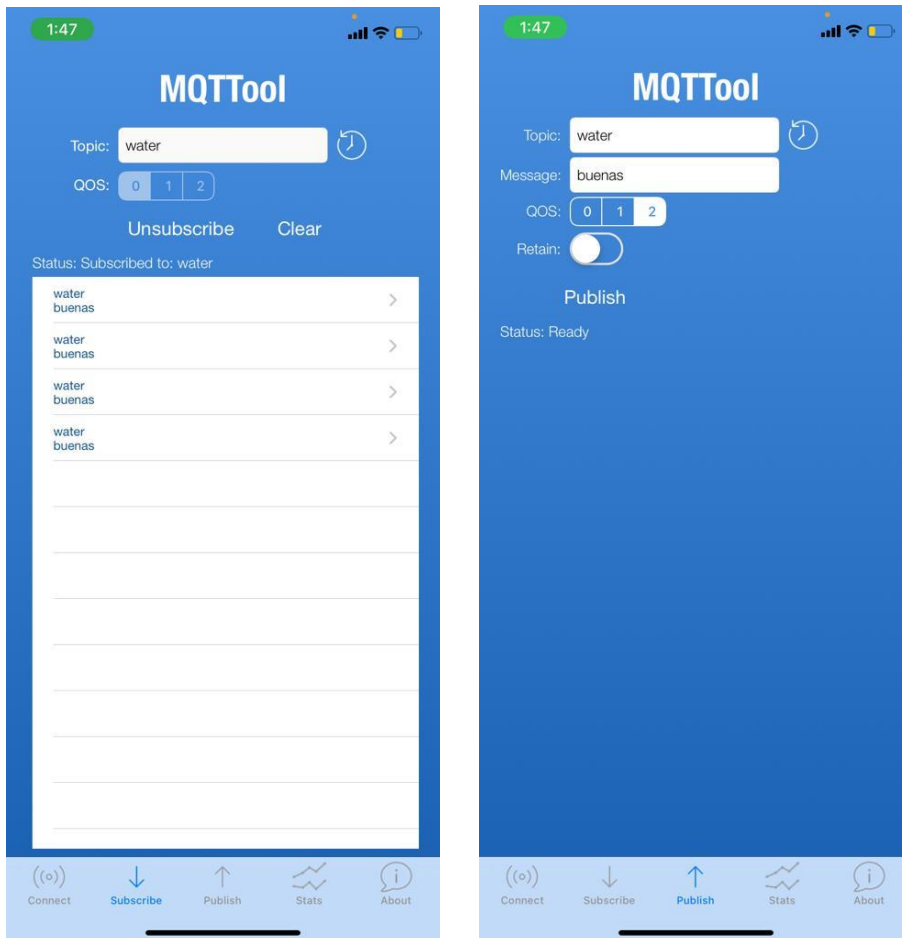
615 }
616 void choose_topic()
617 {
618     int t;
619     PRINTF("\nChoose the topic you want to publish in:(1: water the plants, 2: uv light toggle, 3: humidity)");
620     SCANF("%d",&t);
621     if(t==1)
622     {
623         PRINTF("\nTopic chose: Water");
624         water_the_plants();
625     }
626     if(t==2)
627     {
628         PRINTF("\nTopic chose: UV light");
629         uv_light_toggle();
630     }
631     if(t==3)
632     {
633         PRINTF("\nTopic chose: Humidity");
634         display_humidity();
635     }
636     else
637     {
638         PRINTF("\nInvalid choice");
639     }
640 }
641

```

El punto de esto era que, aunque mandaras el funcionamiento desde la tarjeta, dieras un aviso a tu otro dispositivo de lo que estaba pasando del lado de la tarjeta, por eso publica en cada función.

Ahora, para la parte de subscripción, también nos subscribimos a los mismos tópicos, donde desde la aplicación para iOS “MQTTTool”. En la imagen de la izquierda, podemos ver un ejemplo de cómo estamos suscritos al tópico “wáter” y recibimos mensajes, en la imagen de

la derecha, podemos ver cómo se publica el mensaje en el tópico, al no tener un teléfono Android, no pude usar la app sugerida, por lo que opté por esta, aunque sea menos gráfica.

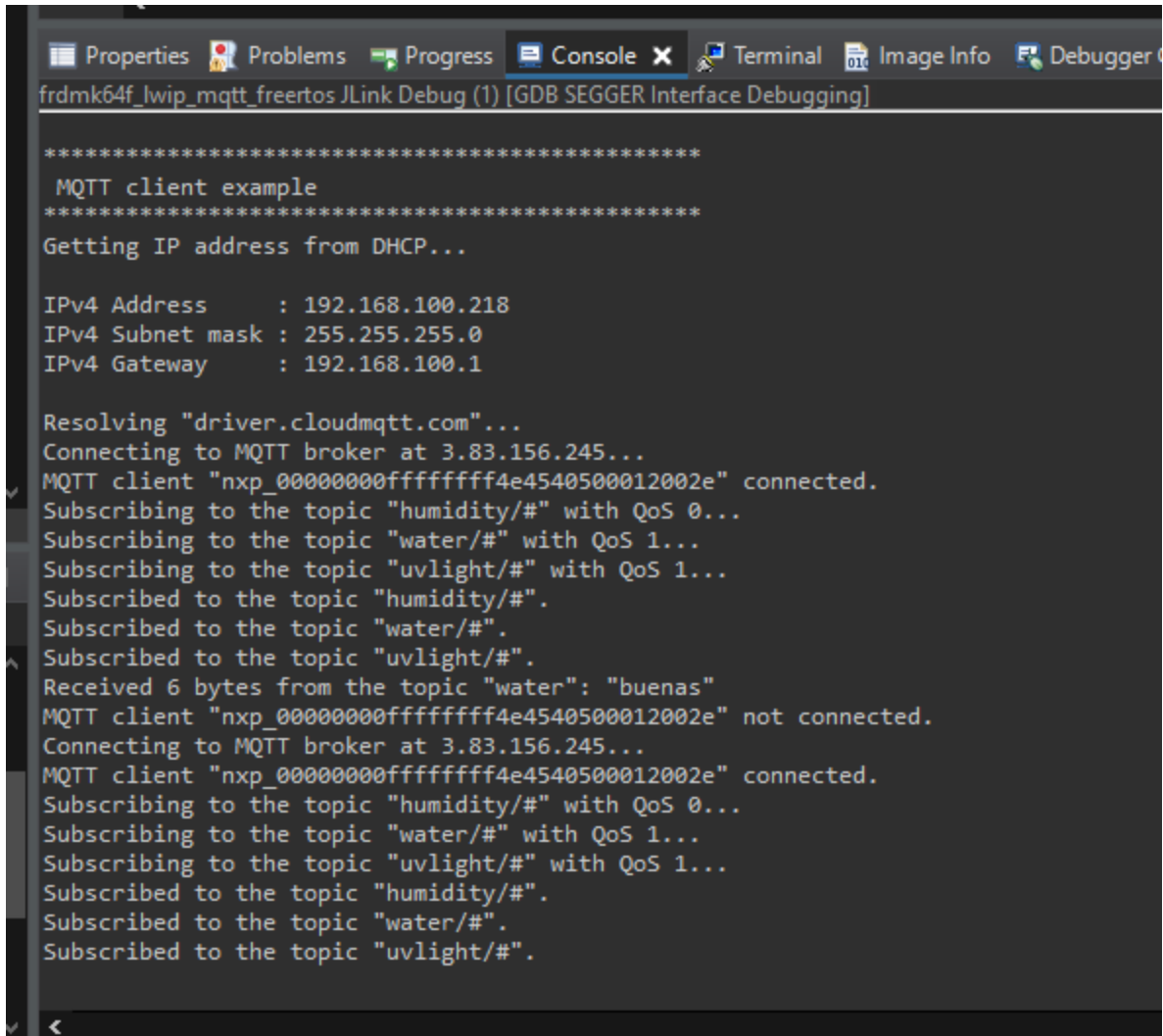


Para identificar a qué tópico se estaban recibiendo mensajes, modifiqué la función `mqtt_incoming_publish_cb`, agregué una condiciones, y según el tópico, mandar llamar la acción.

```
220  * @brief Called when there is a message on a subscribed topic.
221  */
222  static void mqtt_incoming_publish_cb(void *arg, const char *topic, u32_t tot_len)
223  {
224      char *t = (char*)topic;
225      LWIP_UNUSED_ARG(arg);
226      if(*t=='w')
227      {
228          water_the_plants();
229      }
230      if(*t=='u')
231      {
232          uv_light_toggle();
233      }
234      if(*t=='h')
235      {
236          display_humidity();
237      }
238      PRINTF("Received %u bytes from the topic \"%s\": \", tot_len, topic);
239  }
240  }
241
```

Resultados de la implementación

A continuación, dejaré unas imágenes del output del programa ya funcionando:



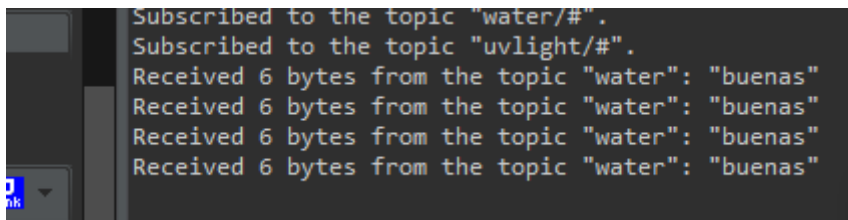
The screenshot shows a debugger interface with a console window titled "frdmk64f_lwip_mqtt_freertos JLink Debug (1) [GDB SEGGER Interface Debugging]". The console output is as follows:

```
*****
MQTT client example
*****

Getting IP address from DHCP...

IPv4 Address      : 192.168.100.218
IPv4 Subnet mask  : 255.255.255.0
IPv4 Gateway      : 192.168.100.1

Resolving "driver.cloudmqtt.com"...
Connecting to MQTT broker at 3.83.156.245...
MQTT client "nxp_00000000ffffffff4e4540500012002e" connected.
Subscribing to the topic "humidity/#" with QoS 0...
Subscribing to the topic "water/#" with QoS 1...
Subscribing to the topic "uvlight/#" with QoS 1...
Subscribed to the topic "humidity/#".
Subscribed to the topic "water/#".
Subscribed to the topic "uvlight/#".
Received 6 bytes from the topic "water": "buenas"
MQTT client "nxp_00000000ffffffff4e4540500012002e" not connected.
Connecting to MQTT broker at 3.83.156.245...
MQTT client "nxp_00000000ffffffff4e4540500012002e" connected.
Subscribing to the topic "humidity/#" with QoS 0...
Subscribing to the topic "water/#" with QoS 1...
Subscribing to the topic "uvlight/#" with QoS 1...
Subscribed to the topic "humidity/#".
Subscribed to the topic "water/#".
Subscribed to the topic "uvlight/#".
```



This is a close-up view of the console output from the previous image, showing the final state of the MQTT client:

```
Subscribed to the topic "water/#".
Subscribed to the topic "uvlight/#".
Received 6 bytes from the topic "water": "buenas"
Received 6 bytes from the topic "water": "buenas"
Received 6 bytes from the topic "water": "buenas"
Received 6 bytes from the topic "water": "buenas"
```


Problemas enfrentados durante la implementación

Los problemas enfrentados durante la implementación de la práctica fueron los siguientes:

- El primero fue el cable ethernet, esto me hizo perder mucho tiempo, no podía probar los cambios que hacía pues el programa no podía avanzar como correspondía si la conexión a ethernet no era estable, esto lo resolví al final, intenté resolverlo conectándolo directamente al modem en vez de a mi laptop, pero no funcionó, ese cable estaba defectuoso, con el segundo cable no lo intenté porque la primera vez me funcionó directamente a mi laptop, pero a veces servía y a veces no, por lo que, en cada prueba, pensaba que el cambio que le había hecho al programa era lo que había provocado que no corriera como debía, al final me di cuenta que era mi puerto ethernet el que tenía falso, con un tercer cable, después de intentar con el puerto, decidí reintentar con el modem.
- Una vez resuelto ese primer problema, era probar lo que ya había escrito para la práctica pero no había tenido oportunidad de probar, y es donde se me presentan dos problemas que realmente no pude resolver, uno de ellos fue que a pesar de que mi interrupción con el botón funcionaba bien, la tarea para checar si la bandera del botón se activaba nunca me funcionó, a pesar de tener la misma prioridad que las demás, esto lo culpo a mi inexperiencia con FreeRTOS.
- El otro problema fue al intentar enviar un mensaje desde mi celular a la tarjeta, a pesar de que sí se enviaba el mensaje, no logré que las condiciones funcionara correctamente, intenté de varias maneras, pero no me alcanzó el tiempo a probar de diferentes formas debido al primer problema que tuve. Así que estoy entregando una práctica que no funciona como debería pero sí tengo una idea de cuáles son las razones por las que no lo hace.
- El último problema fue que, a pesar de que había creado un dashboard en adafruit, nunca logré que se conectara, en retrospectiva no sabría decir si fue por el problema del ethernet o porque adafruit tiene mucha demanda, a pesar de esto, traté de utilizar diferentes dashboard en el navegador y en aplicaciones de escritorio pero no logré conectarlas, por eso terminé optando por la app de iOS, y no fue la única app de iOS que intenté, probé una de paga que tenía un dashboard pero la documentación no estaba muy clara por lo que terminé efectivamente usando MQTTTool.

Conclusiones

Esta práctica a pesar de no ser complicada, me dio muchos dolores de cabeza por problemas ajenos a mí, creo que de haberme dado cuenta de antes del problema de mi computadora me hubiera dado tiempo de corregir el mal funcionamiento de la práctica. A pesar de esto, creo que aun así entiendo cómo funciona el protocolo MQTT y quiero poder utilizarlo como es debido, seguiré tratando de corregir esta práctica después, pero quería dejar una evidencia de lo que trabajé en ella.