Práctica 2

Aplicación MQTT

Jorge Alejandro Flores Triana

ID: 714510

ITESO, Universidad Jesuita de Guadalajara

Instituto Tecnológico y de Estudios Superiores de Occidente

Profesor – Sergio Nicolás Santana

Tópico: Sistemas de Comunicación

Aplicación MQTT

Link Código: https://github.com/a01633675/SistemasComunicacion

Objetivo:

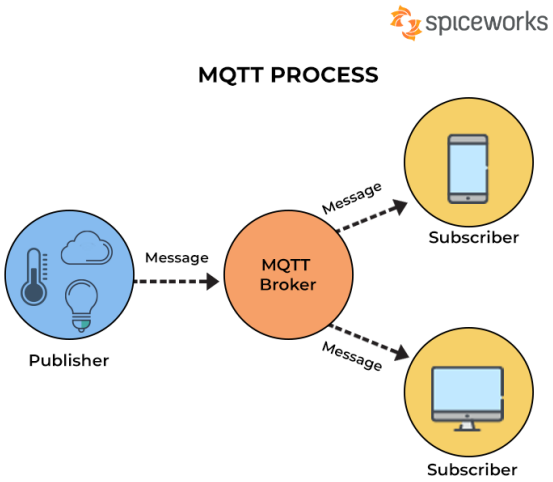
El objetivo de esta práctica es implementar un dispositivo conectado a la nube con MQTT y controlar dicho dispositivo utilizando una aplicación en un dispositivo móvil.

Marco de Referencia:

MQTT es un protocolo de mensajería estándar de OASIS para Internet de las cosas (IoT).

Está diseñado como un transporte de mensajería de publicación/suscripción extremadamente ligero que es ideal para conectar dispositivos remotos con un espacio de código pequeño y un ancho de banda de red mínimo. MQTT hoy en día se utiliza en una amplia variedad de industrias, como la automotriz, la manufactura, las telecomunicaciones, el petróleo y el gas, etc [1].

MQTT implementa el modelo de Publisher y/o Subscriber mediante la definición de clientes y brokers, tal y como se muestra a continuación [2].



- Cliente MQTT

Un cliente MQTT es cualquier dispositivo, desde un servidor hasta un microcontrolador, que ejecuta una biblioteca MQTT. Si el cliente envía mensajes, actúa como Publisher, y si recibe mensajes, actúa como Subscriber.

- Broker MQTT

El Broker MQTT es el sistema que coordina los mensajes entre los diferentes clientes. Las responsabilidades del Broker incluyen recibir y filtrar mensajes, identificar a los clientes suscritos a cada mensaje y enviarles los mensajes.

- Conexión MQTT

Los clientes y los Brokers comienzan a comunicarse mediante una conexión MQTT. Los clientes inician la conexión. Tanto el cliente MQTT como el Broker requieren una stack TCP/IP para comunicarse. Los clientes nunca se conectan entre sí, solo con el Broker.

- Tópico de MQTT

El término “tópico” se refiere a las palabras clave que utiliza el agente MQTT a fin de filtrar mensajes para los clientes de MQTT. Los temas están organizados jerárquicamente, de forma similar a un directorio de archivos o carpetas.

Implementación:

La aplicación que se busca implementar es para monitorear un robot de rescate. Los robots de rescate se han utilizado desde hace tiempo ya que su tecnología ha avanzado de manera importante a lo largo de los años gracias a su fiabilidad mecánica cuando se emplean sobre el terreno.

Los robots de rescate están diseñados especialmente para ayudar en la búsqueda y el rescate de personas tras un desastre natural como los terremotos. Por lo que deben monitorear distintas variables en el entorno:

- Velocidad de los motores [Duty Cycle %]: Ayuda a controlar la forma en la que se desplaza el robot, gracias a un enconder. Información de la aplicación al microcontrolador.

- Sentido de Giro: Ayuda a controlar la dirección del robot. Información de la aplicación al microcontrolador.

- Temperatura del Ambiente [ºC]: Ayuda a determinar si hay fuentes de calor cercanas que representen una amenaza. Información del microcontrolador a la aplicación.

- Sensor de Humo/Gas: Ayuda a saber si hay una fuga de gas cercana o algún otro material tóxico en el aire, así como posibles incendios. Información del microcontrolador a la aplicación.

- Sensor de Corriente [A]: Ayuda a conocer el consumo de corriente de los motores para saber si hay algún riesgo de dañar la circuitería interna. Esto es común ya que el robot se mueve en terrenos abruptos y muchas veces puede quedar atorado, causado estrés en los motores. Información del microcontrolador a la aplicación.

Cada variable descrita anteriormente está relacionada con un tópico establecido:

- rescue/speed
- rescue/direction
- rescue/temperature
- rescue/gas
- rescue/current

Finalmente, varias funciones que venían del ejemplo de FreeRTOS MQTT fueron utilizadas. Las más importantes son:

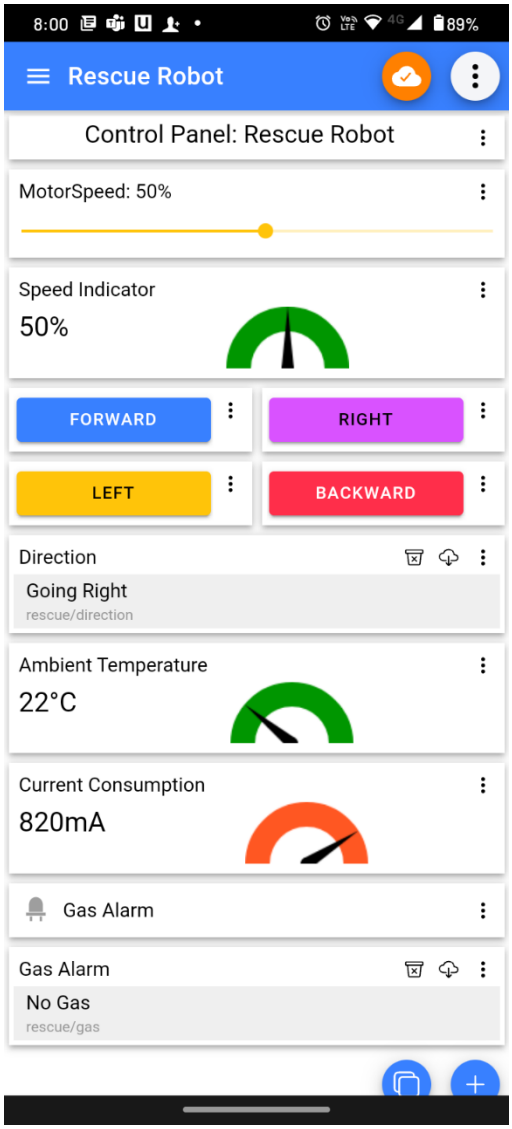
- `connect_to_mqtt ( )`: Esta función conecta el cliente (FRDM-K64) con un broker específico a través de un puerto determinado.

- `mqtt_subscribe_topics ( )`: Esta función se subscribe a un tópico determinado e imprime en la terminal lo que recibe.

- `publish_message ( )`: Esta función publica información a un tópico determinado. Hay que ser cuidadoso al momento de castear los datos antes de ser enviados.

Resultados:

Para probar el código, se realizó una pequeña aplicación móvil en IoT MQTT Panel para simular el control del robot de rescate.



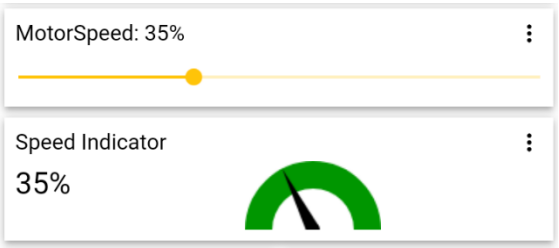
Por otro lado, la tarjeta FRDM-K64 se comunica con el Broker de MQTT (al igual que la aplicación móvil) con los siguientes parámetros

```
/*! @brief MQTT server host name or IP address. */
#define EXAMPLE_MQTT_SERVER_HOST "driver.cloudmqtt.com"


/*! @brief MQTT client information. */
static const struct mqtt_connect_client_info_t mqtt_client_info = {
    .client_id   = (const char *)&client_id[0],
    .client_user = "Jorge",
    .client_pass = "clase2023",
    .keep_alive  = 100,
    .will_topic  = NULL,
    .will_msg    = NULL,
    .will_qos    = 0,
    .will_retain = 0,
#if LWIP_ALTCP && LWIP_ALTCP_TLS
    .tls_config = NULL,
#endif
};
```

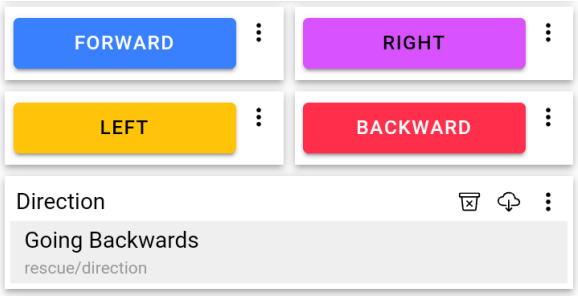
Para el tópico de la velocidad, hay un slider en la aplicación móvil donde se configura el duty cycle de los motores. La tarjeta FRDM-K64 se subscribe al tópico e imprime los valores en la terminal

```
COM34 ×
Resolving "driver.cloudmqtt.com"...
Connecting to MQTT broker at 3.83.156.245...
MQTT client "nxp_00000000ffffffff4e45467250170023" connected.
Subscribing to the topic "rescue/speed" with QoS 1...
Subscribing to the topic "rescue/direction" with QoS 1...
Subscribed to the topic "rescue/speed".
Subscribed to the topic "rescue/direction".
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 35%"
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 60%"
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 15%"
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 85%"
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 60%"
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 20%"
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 85%"
```

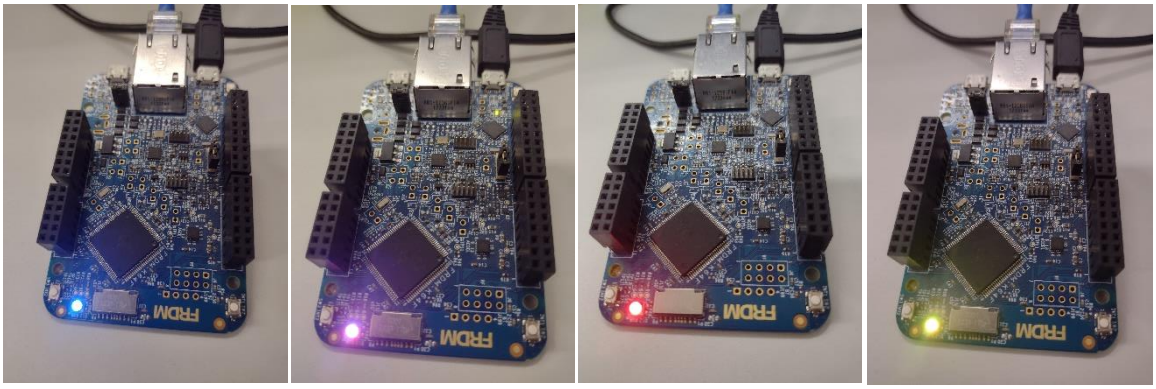


Para el tópico de la dirección, hay unos botones en la aplicación móvil donde se configura la dirección del robot. La tarjeta FRDM-K64 se subscribe al tópico e imprime los valores en la terminal.

```
COM34 ×
Subscribed to the topic "rescue/direction".
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 35%"
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 60%"
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 15%"
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 85%"
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 60%"
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 20%"
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 85%"
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 60%"
Received 2 bytes from the topic "rescue/speed": "Motor Speed: Duty Cycle = 35%"
Received 13 bytes from the topic "rescue/direction": "Going Forward"
Received 11 bytes from the topic "rescue/direction": "Going Right"
Received 10 bytes from the topic "rescue/direction": "Going Left"
Received 15 bytes from the topic "rescue/direction": "Going Backwards"
```

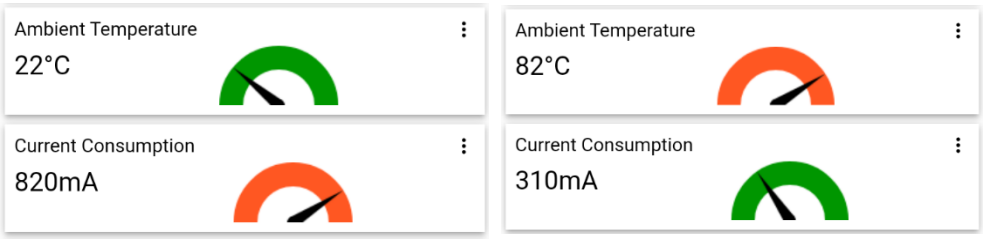


Además, los LEDs de la tarjeta se encienden acorde a un código de colores.

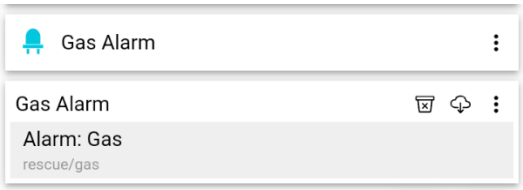


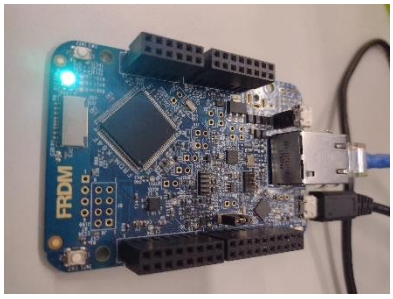
Para los tópico de temperatura y corriente, la aplicación se subscribe para mostrar los valores en un medidor gauge. El botón SW3 de la tarjeta, aumenta los valores de corriente y temperatura previamente definidos en un arreglo.

```
static char currentValues [MAX_MESSAGES][MAX_NUM_OF_CHARS] = {"10", "500", "750", "310", "200", "550", "820", "120", "70", "960"};
static char temperatureValues [MAX_MESSAGES][MAX_NUM_OF_CHARS] = {"10", "52", "45", "82", "73", "40", "22", "38", "65", "13"};
```



Finalmente, para el tópico de toxicidad, la aplicación se subscribe para mostrar con indicador LED si hay presencia de gas en el ambiente. El botón SW2 de la tarjeta controla si hay o no gas en el ambiente de manera simulada.





Conclusión:

Durante la práctica aprendí las bases de MQTT como un protocolo que forma parte de la capa de aplicación del modelo TCP. En resumen, creo que no hubo algo realmente complicado

con la práctica ya que gracias a la tarea anterior en donde teníamos que establecer la conexión con el Broker, gran parte de la práctica estaba avanzada. Creo que las actividades en clase y los tutoriales del profesor fueron de gran ayuda para facilitar la implementación de toda la aplicación.

Una de las cosas que es importante tener en cuenta al momento de publicar los datos es el tipo de variable que queremos transmitir, ya que es probable tener que hacer un casting de entero a ASCII. Por otro lado, también es importante tener una computadora con los puertos habilitados ya que un problema que tuve fue querer correr el cliente en la laptop del trabajo pero el firewall no me dejaba.

Finalmente, creo que hay espacio de mejora en mi aplicación ya que en la vida real es probable que se necesiten controles más dinámicos para manejar el robot, por lo que sería interesante continuar con el desarrollo de esta aplicación.

Referencias:

[1] MQTT. (2022). MQTT: The Standard for IoT Messaging. Recuperado de: https://mqtt.org/

[2] Amazon. (2022). ¿Qué es MQTT? Recuperado de: https://aws.amazon.com/es/what-is/mqtt/