# Práctica 2 - MQTT

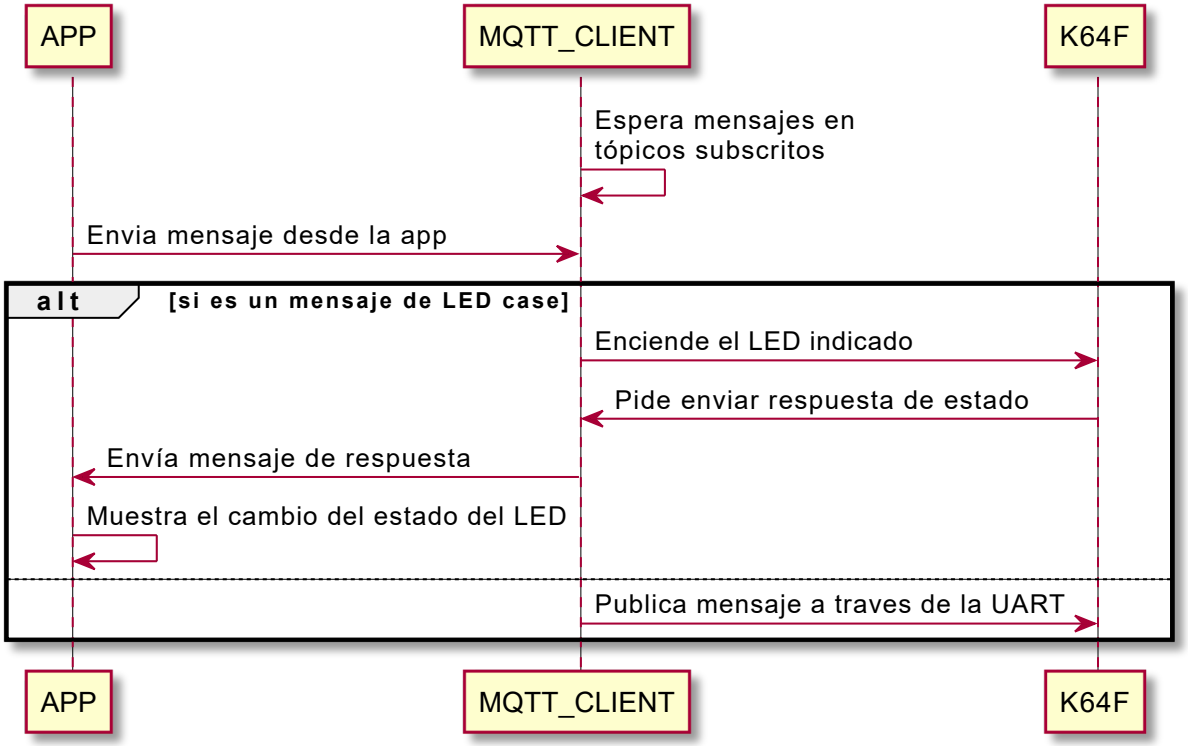
**Autor: Miguel Pérez García**

## Introducción

MQTT es un protocolo de mensajería diseñado inicialmente por IBM en 1999, el objetivo de este protocolo era comunicar dispositivos embebidos entre ellos o hacía un servicio central. Es un protocolo basado en el modelo de publicador y subscriptor, de modo que los clientes puedes subscribirse a un tópico y recibir los mensajes que los publicadores pongan en este tópico.

## Desarrollo de práctica

Para la práctica desarrollé una aplicación que permite controlar dos LED's de la tarjeta de desarrollo K64F, a continuación se explicará el código. El siguiente diagrama muestra el funcionamiento general del proyecto.




El código que se utilizó de base fue el ejemplo del SDK de la K64F de MQTT, este ejemplo ya tiene implementados los métodos para conectarse a cualquier servidor por lo que solo bastó añadir la parte de la aplicación que pretendía realizar. El código que se implementó es el siguiente:

```
uint8_t *last_topic;
//Este callback se ejecuta cuando llega un mensaje a un tópico al que se ha
subscrito
static void mqtt_incoming_publish_cb(void *arg, const char *topic, u32_t tot_len)
{
    LWIP_UNUSED_ARG(arg);
```

```
    // Obtengo la longitud del tópico
    size_t length = strlen(topic);
    if(NULL != last_topic) free(last_topic);
    // Creo espacio suficiente para almacenarlo
    last_topic = malloc(length * sizeof(uint8_t));
    memset(last_topic, 0, length);
    // Copio el tópico a una variable global para usarse posteriormente
    memcpy(last_topic, topic, length);

    PRINTF("Received %u bytes from the topic \"%s\": \"", tot_len, topic);
}

static void mqtt_incoming_data_cb(void *arg, const u8_t *data, u16_t len, u8_t
flags)
{
    LWIP_UNUSED_ARG(arg);

    size_t last_topic_size = strlen(last_topic);
    // Obtengo el identificador del tópico
    uint8_t device_number = last_topic[last_topic_size - 2];
    // El número 1 representa al LED rojo
    if('1' == device_number) {
        //Cambiamos el estado del LED
        GPIO_PortToggle(BOARD_LED_GPIO, 1 << BOARD_LED_GPIO_PIN);
        red_pin_status = !red_pin_status;
        uint8_t *message = red_pin_status ? "ON" : "OFF";
        // Enviamos la respuesta del cambio de estado
        publish_message_simp("device/led/1/r", message, 1);
    // El número 2 representa al LED verde
    }else if('2' == device_number) {
        //Cambiamos el estado del LED
        GPIO_PortToggle(BOARD_LED_GREEN_GPIO, 1 << BOARD_LED_GREEN_PIN);
        green_pin_status = !green_pin_status;
        uint8_t *message = green_pin_status ? "ON" : "OFF";
        // Enviamos la respuesta del cambio de estado
        publish_message_simp("device/led/2/r", message, 1);
    // La letra 'v' representa al botón 'virtual' de la aplicación
    }else if('v' == device_number){
        // En este caso solo imprimimos que se presionó el botón
        PRINTF("Virtual button pressed");
    }else{
        PRINTF("NO DEVICE RELATED\n\r");
    }

    if (flags & MQTT_DATA_FLAG_LAST)
    {
        PRINTF("\"\r\n");
    }
}
```

## Problemas enfrentados

El mayor inconveniente con el que me enfrenté fue con Adafruit, si bien su plataforma se ve robusta, me parece un poco restrictiva puesto que obliga a utilizar un esquema para las rutas y su documentación no me parece suficientemente clara para saber como se deben enviar paquetes para que se vean reflejados en sus propios dashboards, mi solución fue pasarme a un servidor propio, ya que en ocasiones anteriores había utilizado MQTT, tanto en mis estudios profesionales como en la especialidad, fue solo cuestión de encender de nuevo mi servidor y poner a correr mosquitto.

Otro problema al que llegué fue el uso de interrupciones, anteriormente había utilizado las interrupciones de los botones con FreeRTOS y MQTT, sin embargo en esta ocasión no pude realizar correctamente el lanzar el mensaje desde una tarea que solo se inicia con la interrupción, decidí quitarme el problema completamente y no utilizar las interrupciones, sin embargo será una tarea para el fin de semana.

## Conclusiones

MQTT me parece un protocolo muy interesante, anteriormente lo había utilizado para comunicar una tarjeta de Qualcomm con una aplicación móvil que fue el proyecto final de aplicaciones móviles en mi educación profesional. Lo que me gusta mucho es que al ser únicamente un protocolo, sin estar atado a nada, se pueden hacer cosas muy interesantes entre MQTT y AMQP, que es un protocolo similar pero mas robusto ya que fue diseñado para la industria bancaria, con brokers como RabbitMQ que permite la interacción entre dispositivos de estos protocolos.

El código se encuentra en este repositorio y el video