Inicio › Ingeniería › Cursos › Curso MQTT

Enviar y recibir mensajes por MQTT con Arduino y la librería PubSubClient

Mié, 4 Ago 2021 | ~ 6 minutos

Continuamos con las entradas dedicadas a la comunicación por MQTT viendo cómo **enviar o recibir mensajes por MQTT desde un procesador como Arduino** gracias a la librería PubSubClient.

Ya llevamos varias entradas dentro de la **serie dedicada a MQTTT**, viendo **Qué es MQTT y su importancia en el IoT**, **Qué son los Topics y cómo usarlos** y aprendido a **instalar Mosquitto**, uno de los broker de MQTT más populares.

Siendo como es este blog, se veía venir de lejos que que el objetivo final era aplicarlo en un microprocesador. Después de todo, una de las funciones principales de MQTT es aplicarse a MCUs para hacer dispositivos IoT.

Si os parece que va a ser muy difícil ¡ya podéis ir quitándoos el miedo!. La comunicación MQTT **es una de las formas de comunicación más fácil de usar** que podemos emplear en nuestros proyectos. De hecho, como vimos al presentar el protocolo, la facilidad y la robustez es uno de los puntos fuertes de MQTT.

En el caso de MCUs, afortunadamente, **integrar MQTT en un procesador como Arduino es muy sencillo** gracias a la existencia de varias librerías. La más popular y conocida conocida es la genial librería **PubSubClient** desarrollada por Nick O'Leary.

PubSubClient es compatible con una gran variedad de dispositivos e interfaces web. Cuando fue desarrollada, fue principalmente pensada para una combinación de Arduino junto con un shield Ethernet o un shield Wifi. Por tanto, lo que veréis que muchos ejemplos y tutoriales hacen referencia a estos shields.

No obstante, a estas alturas, resulta más frecuente es emplear PubSubClient en **procesadores más avanzados que incorporan WiFi** de forma nativa. Como, por ejemplo, nuestros queridos ESP8266 y ESP32, que son buenos conocidos del blog en **su propia sección**.

> En esta entrada veremos el funcionamiento de la librería PubSubClient en su forma general. En la medida de lo posible vamos a **evitar hacer referencia a un procesador o interface en particular**. Volveremos a hablar de ella en más profundidad en la sección dedicada al **ESP8266/ESP32**, donde sí entraremos en más detalle..

PubSubClient es un cliente MQTT para microprocesadores y dispositivos IoT. Por defecto usa MQTT 3.1.1, aunque puede ser cambiada cambiando la variable MQTT_VERSION en el archivo PubSubClient.h. Permite suscribirse a mensajes en QoS 0 o QoS 1, aunque únicamente es posible publicar mensajes en QoS 0.

COMUNIDAD

DISCORD

CONTENIDO DEL ARTICULO

Enviar y recibir mensajes por MQTT con Arduino y la librería PubSubClient
Usando la librería PubSub
Constructores
Suscribirse a un topic
Enviar un mensaje
Funciones importantes
Ejemplo de código
Descarga el código

CURSO DE MQTT

MQTT
Ejemplos ESP32
Librería PubSub
Comunicar Web con MQTT
ESP32 y MQTT
ESP32 y AsyncMQTT
ESP32 + MQTT + Json
ESP32 + MQTT + Vue
Interface ESP32 + MQTT
Ejemplos C#

RELACIONADO

Curso ESP8266/ESP32
Curso iniciación Arduino
Curso Arduino avanzado

El tamaño máximo del mensaje a enviar, incluido la cabecera, es de 256 bytes. Esto es suficiente para la mayoría de proyectos. No obstante, es posible variarlo cambiando la constante MQTT_MAX_PACKET_SIZE en el fichero PubSubClient.h.

El intervalo de KeepAlive es de 15 segundos por defecto, aunque también es posible cambiarlo mediante la constante MQTT_KEEPALIVE, o llamando a la función estática PubSubClient::setKeepAlive(keepAlive).

La librería PubSubClient es Open Source, y todo el código está disponible en **https://github.com/knolleary/pubsubclient**. También la tenéis disponible a través del gestor de librerías del IDE de Arduino.

Usando la librería PubSub

Constructores

Podemos iniciar la librería mediante uno de sus constructores

```
PubSubClient ()
PubSubClient (client)
PubSubClient (server, port, [callback], client, [stream])
```

Siendo:

- Client, interface de comunicación que estamos usando
- Server, nombre del broker y su IP
- Port, puerto del broker, por defecto 1883
- Callback, función a ejecutar al recibir un mensaje
- Stream, instancia de un stream para almacenar los mensajes

Suscribirse a un topic

Para suscribirnos a un topic usamos la siguiente función:

```
mqttClient.subscribe("hello/world");
```

Enviar un mensaje

Para enviar un mensaje en un topic simplemente hacemos,

```
mqttClient.publish("hello/world", (char*)payload.c_str());
```

Funciones importantes

La otra función fundamental de la librería pubsub es loop(), que debe ser llamada frecuentemente para permitir al cliente gestionar las solicitudes y envíos MQTT.

```
mqttClient.loop();
```

Otras funciones interesantes del objeto PubSubClient son:

```
boolean connect(const char* id);
boolean connect(const char* id, const char* user, const char* pass);
void disconnect();

boolean publish(const char* topic, const char* payload);

boolean subscribe(const char* topic);
boolean subscribe(const char* topic, uint8_t qos); // qos es 0 o 1, por defecto 0
boolean unsubscribe(const char* topic);

PubSubClient& setServer(IPAddress ip, uint16_t port);
PubSubClient& setCallback(MQTT_CALLBACK_SIGNATURE);
```

```
PubSubClient& setClient(Client& client);
PubSubClient& setStream(Stream& stream);
PubSubClient& setKeepAlive(uint16_t keepAlive);
PubSubClient& setSocketTimeout(uint16_t timeout);

// prototipo de función callback
void OnMqttReceived(char* topic, byte* payload, unsigned int length)
```

Ejemplo de código

A continuación, tenemos un ejemplo de código, basado en los ejemplos de la propia librería. Este ejemplo está basado en un Shield de Ethernet, pero el funcionamiento es similar con cualquier otra conexión.

```
#include <Ethernet.h>

#include <PubSubClient.h>

// direcciones IP, servidor, y MAC
// necesarias para Ethernet Shield
IPAddress ip(172, 16, 0, 100);
IPAddress server(172, 16, 0, 2);
byte mac[] = {
    0xDE,
    0xED,
    0xBA,
    0xFE,
    0xFE,
    0xED};

// instanciar objetos
EthernetClient ethClient;
PubSubClient client(ethClient);

// constantes del MQTT
// direccion broker, puerto, y nombre cliente
const char *MQTT_BROKER_ADRESS = "192.168.1.150";
const uint16_t MQTT_PORT = 1883;
const char *MQTT_CLIENT_NAME = "ArduinoClient_1";

// realiza las suscripción a los topic
// en este ejemplo, solo a 'hello/world'
void SuscribeMqtt()
{
    mqttClient.subscribe("hello/world");
}

// callback a ejecutar cuando se recibe un mensaje
// en este ejemplo, muestra por serial el mensaje recibido
void OnMqttReceived(char *topic, byte *payload, unsigned int length)
{
    Serial.print("Received on ");
    Serial.print(topic);
    Serial.print(": ");

    String content = "";
    for (size_t i = 0; i < length; i++)
    {
        content.concat((char)payload[i]);
    }
    Serial.print(content);
    Serial.println();
}
```

```
50
51  // inicia la comunicacion MQTT
52  // inicia establece el servidor y el callback al recibir un mensaje
53  void InitMqtt()
54  {
55      mqttClient.setServer(MQTT_BROKER_ADRESS, MQTT_PORT);
56      mqttClient.setCallback(OnMqttReceived);
57  }
58
59  // conecta o reconecta al MQTT
60  // consigue conectar -> suscribe a topic y publica un mensaje
61  // no -> espera 5 segundos
62  void ConnectMqtt()
63  {
64      Serial.print("Starting MQTT connection...");
65      if (mqttClient.connect(MQTT_CLIENT_NAME))
66      {
67          SuscribeMqtt();
68          client.publish("connected","hello/world");
69      }
70      else
71      {
72          Serial.print("Failed MQTT connection, rc=");
73          Serial.print(mqttClient.state());
74          Serial.println(" try again in 5 seconds");
75
76          delay(5000);
77      }
78  }
79
80  // gestiona la comunicación MQTT
81  // comprueba que el cliente está conectado
82  // no -> intenta reconectar
83  // si -> llama al MQTT loop
84  void HandleMqtt()
85  {
86      if (!mqttClient.connected())
87      {
88          ConnectMqtt();
89      }
90      mqttClient.loop();
91  }
92
93  // únicamente inicia seria, ethernet y MQTT
94  void setup()
95  {
96      Serial.begin(9600);
97
98      Ethernet.begin(mac, ip);
99      delay(1500);
100     InitMqtt();
101 }
102
103 // únicamente llama a HandleMqtt
104 void loop()
105 {
106     HandleMqtt();
107 }
```

He puesto bastantes comentarios en el código para explicarlo, pero como puntos destacables:

- Hemos dividido el código divido en funciones “más o menos” reutilizables
- En el setup, únicamente es necesario llamar a la función InitMqtt
- InitMqtt establece el callback OnMqttReceived, que muestra por serial los mensajes recibidos

- Por otro lado, en el loop, llamamos a la función HandleMqtt
- La función HandleMqtt reconecta el clientes si es necesario, de lo contrario ejecuta el loop del cliente
- Si es necesario conectar/reconectar, llama a ConnectMqtt
- ConnectMqtt conecta el cliente, y se suscribe a los topic con SuscribeMqtt

Si lo ejecutamos, veremos en el puerto serial los mensajes recibidos. Para las pruebas, por ejemplo, podemos usar MQTT Explorer, un cliente genérico que vimos en **esta entrada**.

¡Así de fácil! Como habíamos anticipado, no es nada complicado conectar un microprocesador a una red MQTT. De hecho, es muy sencillo, siendo esta es una de las ventajas y fortalezas de este sistema.

Las próximas entradas sobre MQTT serán en la sección del ESP8266 y ESP32. Aunque haremos referencia a esta entrada cuando sea necesario, también veremos que estos dos disponen de otra librería para realizar la conexión MQTT de forma asíncrona. ¡Nos vemos en la próxima entrada.

Descarga el código

Todo el código de esta entrada está disponible para su descarga en Github.

Temas: Mqtt

Continuar leyendo: Curso de MQTT

Anterior
MQTT Explorer, un cliente genérico para MQTT

Siguiente
Cómo comunicar un ESP32 con una página web a través de MQTT

Luis Llamas
Ingeniería, informática y diseño