

Práctica 5: Protocolo MQTT



Sistemas Embebidos

Francisco Javier Pérez Martínez
Alex Navarro Soria

Francisco Joaquín Murcia Gómez
Elvi Mihau Sabau Sabau

30 de abril de 2022



Índice

1. Abstracto	3
2. Descripción	3
3. Parte A. Instalación y configuración de Arduino.	4
3.1. Configuración general	4
3.2. Productores	5
3.3. Consumidor	6
4. Parte B. Conexión del dispositivo consumidor a la plataforma Cloud.	7
4.1. Problemas de la implementación	7
4.2. Primera solución	7
4.3. Cliente de ArduinoCloud incompatible con la librería ArduinoMqttClient.	8
4.4. Segunda solución	8
4.5. Desarrollo de la práctica.	10
5. Conclusiones	14



1. Abstracto

En esta práctica usaremos un arduino nano 33 IoT con headers para conectarnos y emitir mensajes a un broker, usando la arquitectura productor / consumidor mediante el protocolo MQTT.

Keywords: MQTT, Arduino, Mosquitto, Producer, Consumer

2. Descripción

Durante esta práctica implementaremos una arquitectura de comunicación entre 4 arduinos, haciendo que uno sea un consumidor, y los otros 3 haciendo de productores.

Usaremos el broker MQTT de mosquitto^a, este recibirá los mensajes de los productores, y los consumidores al suscribirse, recibirán los mensajes.

Estos mensajes se envían y se leen todos de un tópico en concreto, este tópico se usa para diferenciar un canal de mensajes de otros.

El esquema de la arquitectura sería el siguiente:

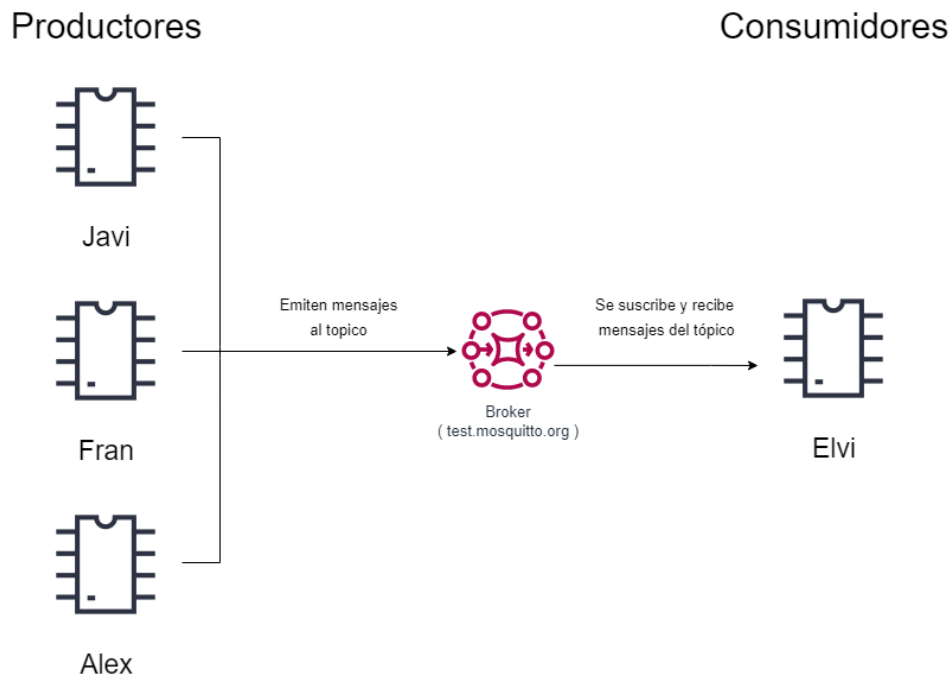


Figura 1: Esquema de la estructura del canal de comunicaciones.

^a<https://test.mosquitto.org/>



3. Parte A. Instalación y configuración de Arduino.

3.1. Configuración general

Los arduinos usarán la librería `mqttClient` utilizando el protocolo MQTT. Este a la hora de inicializarse requiere de un cliente de red para su uso, da igual si es `WiFiClient`^b o `EthernetClient`^c.

Para la conexión vía WiFi, usaremos `WiFiNina`, como en las prácticas anteriores.

Dependiendo de si el dispositivo es un productor o consumidor, este realizará diferentes tareas, pero utilizando la misma librería `mqttClient`, ya que esta permite la transmisión de datos a través del protocolo MQTT de forma genérica.

^b<https://www.arduino.cc/en/Reference/WiFiClient>

^c<https://www.arduino.cc/en/Reference/EthernetClient>



3.2. Productores

Para la parte de los 3 clientes MQTT, se ha cargado en el entorno la biblioteca "ArduinoMqtt-Client". Una vez instalada dicha librería, cargaremos el ejemplo "WifiSimpleSender" y modificaremos el mensaje que publicará cada cliente.

Mensaje a enviar : Nombre + FechaHora + número aleatorio [0..100]

Para obtener la fecha y la hora hemos tenido que incluir la librería "NTPClient" para conectar-nos a un servidor de tiempo y así mantener ambos datos sincronizados.

Concretamente, para la hora utilizamos el siguiente enlace^d Sin embargo, nos surgió un problema para obtener la fecha y es que la función `getFormattedDate()` no la reconocía y por tanto tuvimos que emplear otro método. Realizando otra búsqueda, hallamos que el cliente NTP instalado no viene con funciones para obtener la fecha, por lo que debemos crear una estructura de tiempo (`struct tm`) y luego, acceder a sus elementos para obtener información sobre la fecha.^e

```
1  #include <NTPClient.h>
2  #include <WiFiUdp.h>
3  #include "time.h"
4
5  WiFiUDP ntpUDP;
6  NTPClient timeClient(ntpUDP, "europe.pool.ntp.org", 7200);
7  ...
8  void setup() {
9      ...
10     timeClient.begin();
11     ...
12 }
13 void loop() {
14     // time structure
15     time_t epochTime = timeClient.getEpochTime();
16     struct tm *ptm = gmtime ((time_t *)&epochTime);
17     int monthDay = ptm->tm_mday;
18     int currentMonth = ptm->tm_mon+1;
19     int currentYear = ptm->tm_year+1900;
20     String currentDate = String(monthDay) + "/" + String(currentMonth) + "/" + String(currentYear);
21
22     mqttClient.print("Javi | Fecha y Hora: ");
23     mqttClient.print(currentDate); // publicar fecha
24     mqttClient.print(" - ");
25     mqttClient.print(timeClient.getFormattedTime()); // publicar hora obtenida en getFormattedTime()
26     mqttClient.print(" | nº random: ");
27     mqttClient.println(nrand); // publicar nº random
28 }
```

^d<https://randomnerdtutorials.com/esp32-ntp-client-date-time-arduino-ide/>

^e<https://randomnerdtutorials.com/esp8266-nodemcu-date-time-ntp-client-server-arduino/>



3.3. Consumidor

El consumidor se conectará a la red WiFi, y posteriormente se intentará conectar al broker de mosquitto, especificando la dirección del broker `test.mosquitto.org` y el puerto por defecto 1883. Al confirmar la conexión, este se subscribirá al tópico `SE/practicaUA2022/murcia`, donde esperará a recibir mensajes de los productores.

Cada vez que un productor emita un mensaje al tópico, el consumidor recibirá dicho mensaje, y lo mostrará por el monitor serial, como se muestra a en la figura 2:

```
Received a message with topic 'SE/practicaUA2022/murcia', length 59 bytes:
Javi | Fecha y Hora: 2/4/2022 - 18:10:10 | n° random: 85

Received a message with topic 'SE/practicaUA2022/murcia', length 59 bytes:
Alex | Fecha y Hora: 2/4/2022 - 18:10:11 | n° random: 19

Received a message with topic 'SE/practicaUA2022/murcia', length 59 bytes:
Fran | Fecha y Hora: 2/4/2022 - 18:10:10 | n° random: 80

Received a message with topic 'SE/practicaUA2022/murcia', length 59 bytes:
Javi | Fecha y Hora: 2/4/2022 - 18:10:11 | n° random: 93
```

Figura 2: Captura del Monitor Serial del Arduino Consumidor.



4. Parte B. Conexión del dispositivo consumidor a la plataforma Cloud.

4.1. Problemas de la implementación

Debido a que la propia librería de ArduinoCloud se encarga automáticamente de gestionar la conexión a la red, no hemos podido obtener el cliente de red que esta genera, y sin el cliente de red no podemos inicializar la librería `mqttClient`.

Si intentamos sobre escribir la conexión conectándonos manualmente a la red como hicimos en el apartado anterior experimentaremos una falla, ya que habrían 2 librerías que estarían usando la antena del dispositivo, la nuestra (WiFiNina) y la integrada en el sketch de ArduinoCloud.

De momento, estamos aún investigando si hay alguna manera de obtener el objeto instanciado del cliente de red que genera ArduinoCloud para usarlo con la librería `mqttClient`, incluso hemos preguntado por el foro de Arduino^f y por el repositorio oficial de ArduinoIoTCloud^g.

4.2. Primera solución

Después de 12 días, al preguntar por el repositorio de ArduinoIoTCloud, la comunidad nos respondió, mostrándonos la manera de obtener el cliente wifi desde el objeto `ArduinoIoTPreferredConnection`.

Además, nuestra pregunta en el repositorio se ha etiquetado como una imperfección en la documentación de ArduinoCloud, así que de paso en esta práctica estamos ayudando a un proyecto open source.

La manera en la que obtendremos el cliente wifi y lo engancharemos a nuestro cliente `mqtt` es la siguiente:

```
1  #include <ArduinoMqttClient.h>
2  #include "thingProperties.h";
3  ...
4  void setup() {
5      ...
6
7      initProperties();
8      ArduinoCloud.begin(ArduinoIoTPreferredConnection);
9      mqttClient = new MqttClient(ArduinoIoTPreferredConnection.getClient());
10     ...
11 }
12 ...
```

^f<https://forum.arduino.cc/t/getting-wificlient-from-arduinocloud-object/973385>

^g<https://github.com/arduino-libraries/ArduinoIoTCloud/issues/314>



4.3. Cliente de ArduinoCloud incompatible con la librería ArduinoMqttClient.

Por desgracia, el cliente wifi que nos proporciona ArduinoCloud no es compatible con el uso de la librería, así que tendremos que pensar en otra estrategia.

4.4. Segunda solución

Así que probamos lo mismo, pero al revés. Hicimos uso de WIFInina, que sabemos que es compatible con la librería mqtt, y la enganchamos como `ArduinoIoTPreferredConnection` a ArduinoCloud, y probamos a ver si funciona.

```
1  ...
2  WiFiClient wifiClient;
3  MqttClient mqttClient(wifiClient);
4
5  void setup() {
6      Serial.begin(9600);
7      delay(1500);
8
9      Serial.print("Attempting to connect to WPA SSID: ");
10     Serial.println(SSID);
11     while (WiFi.begin(SSID, PASS) != WL_CONNECTED) {
12         // failed, retry
13         Serial.print("Failed connection to WiFi, retrying...\n");
14         delay(5000);
15     }
16
17     Serial.println("You're connected to the network");
18     Serial.println();
19
20     initProperties();
21     ArduinoCloud.begin(wifi);
22     setDebugMessageLevel(2);
23     ArduinoCloud.printDebugInfo();
24     Serial.println();
25     ...
```



Al fin, con esta solución, podemos recibir los datos vía mqtt y mandarlos a ArduinoCloud.

```
Failed connection to WiFi, retrying...
You're connected to the network

***** Arduino IoT Cloud - configuration info *****
Device ID: 75db6bc5-98b8-45ce-8288-543d736a1f95
MQTT Broker: mqttts-sa.iot.arduino.cc:8883

Attempting to connect to the MQTT broker: test.mosquitto.org
You're connected to the MQTT broker!

Subscribing to topic: SE/practicaUA2022/murcia

Waiting for messages on topic: SE/practicaUA2022/murcia

WiFi.status(): 3
Current WiFi Firmware: 1.4.8
Connected to "arduino"
ArduinoIoTCloudTCP::handle_ConnectMqttBroker could not connect to mqttts-
sa.iot.arduino.cc:8883
ArduinoIoTCloudTCP::handle_ConnectMqttBroker 1 connection attempt at tick
time 29877
Connected to Arduino IoT Cloud
Thing ID: e7159ff3-9b41-4ded-82ff-d1e61fddcce8
Javi:33
Alex:62
Javi:0
Javi:52
Fran:56
```

Figura 3: Captura del Monitor Serial de ArduinoCloud.





4.5. Desarrollo de la práctica.

Procedemos a crear la variable que se nos pide en el enunciado:


Add variable


Name
ejercicio

 Sync with other Things 


Floating Point Number eg. 1.55 ▼

Declaration
`float ejercicio;`



Variable Permission 

☒ Read & Write ☐ Read Only

Variable Update Policy 

☒ On change ☐ Periodically

Figura 4: Captura de ArduinoIoTCloud, creando la variable `ejercicio`.



Una vez creada la variable, la enganchamos al mensaje recibido vía MQTT en nuestro Sketch.

Lo que hemos hecho es que, por cada mensaje recibido, como con `mqttClient.read()` leemos cada carácter, usaremos un carácter delimitador para separar el dispositivo del numero, el formato del mensaje será entonces el siguiente "nombre:numero" siendo los dos puntos (:) el carácter delimitante.

Una vez que tenemos las dos partes del mensaje separadas, podremos "montar" carácter a carácter cada parte del mensaje, una vez hecho esto, dependiendo de cual arduino esta emitiendo el numero, nos guardaremos dicho numero en el Array de números.

Cuando tengamos todos los Arrays de números llenos, sacaremos la media de estos, y la asignaremos a la variable, después resetearemos el valor de dichos Arrays.

```
1 String numbers[3];
2 String arduino;
3 ...
4 void loop() {
5     ArduinoCloud.update();
6
7
8
9
10    if (mqttClient.parseMessage()) {
11        bool nextIsNumber = FALSE;
12        char character;
13
14        // Obtain data per character.
15        while (mqttClient.available()) {
16            character = (char)mqttClient.read();
17
18            // Message format is device:number.
19            // Ex: Fran:23.
20            if (character != ':' && !nextIsNumber) {
21                arduino += (char)mqttClient.read();
22            } else if (character == ':') {
23                nextIsNumber = TRUE;
24            } else if ( character != ':' && nextIsNumber) {
25                switch (arduino) {
26                    "fran": numbers[0] += character; break;
27                    "alex": numbers[0] += character; break;
28                    "javi": numbers[0] += character; break;
29                }
30            }
31
32        }
33    }
```



```
34     // Do median, and publish to observable.
35     if (numbers[0] && numbers[1] & numbers[2]) {
36         float sum = (numbers[0].toFloat() + numbers[1].toFloat() + numbers[2].toFloat());
37         ejercicio = sum / 3;
38
39         // Reset array of numbers
40         numbers = {"", "", ""};
41     }
42 }
43 }
```



A continuación, creamos el dashboard, y un gráfico que enlazaremos al numero aleatorio recibido, para representar de manera visual dichos valores recibidos.

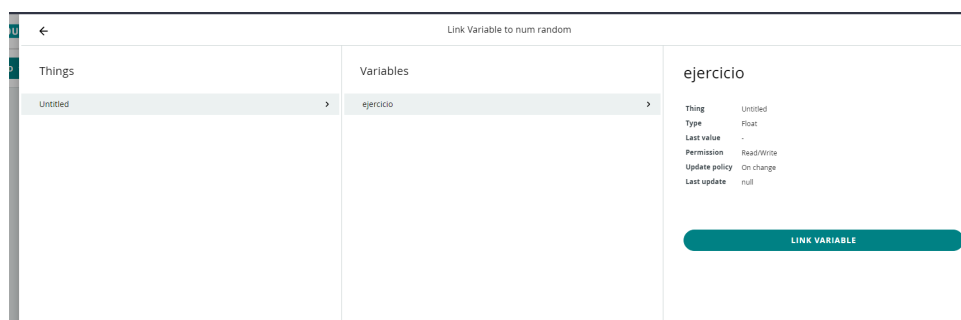


Figura 5: Captura de ArduinoIoTCloud, creación del gráfico enlazando la variable `ejercicio`.

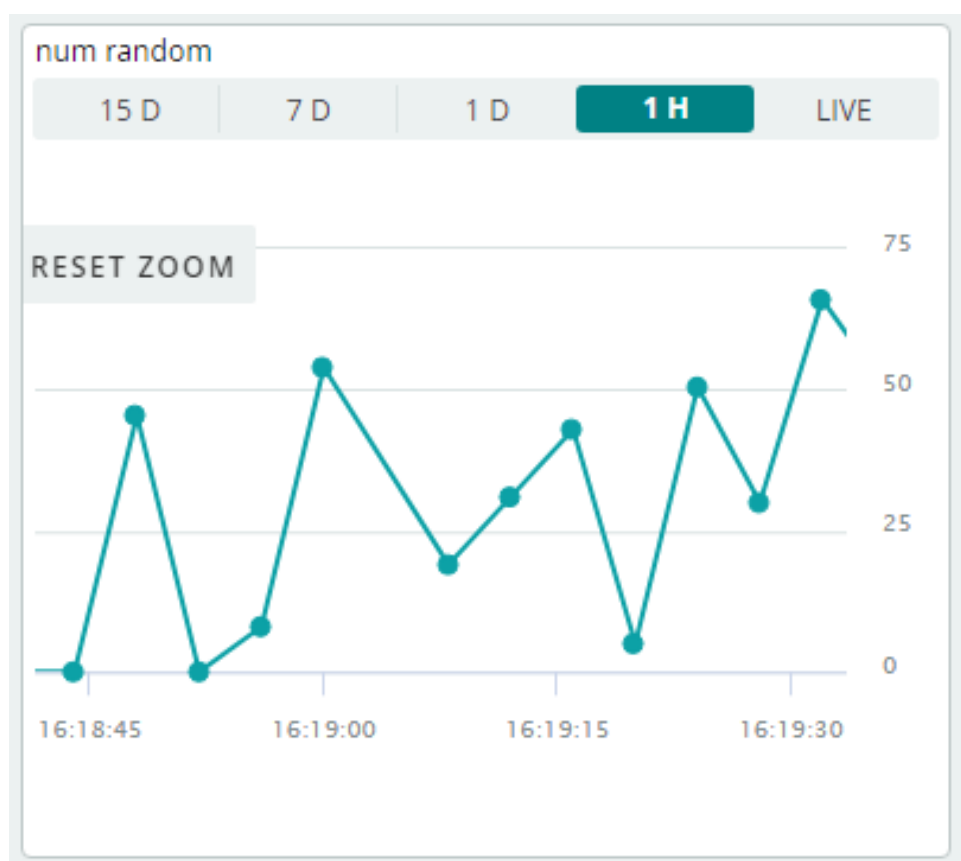


Figura 6: Captura de ArduinoIoTCloud, captura del gráfico.



5. Conclusiones

Por la experiencia que hemos tenido usando ArduinoCloud en esta práctica y la anterior, podemos decir con seguridad que tiene un gran potencial, pero hay muchas características mejorables.