



## Contenido

PROTOCOLOS DE COMUNICACION .....	2
CLASIFICACION DE PROTOCOLOS .....	2
LOS PROTOCOLOS ORIENTADOS A CONEXIÓN.....	2
LOS PROTOCOLOS NO ORIENTADOS A CONEXIÓN .....	3
TCP .....	3
UDP .....	3
MQTT .....	6
CARACTERISTICAS PRINCIPALES .....	7
PROTOCOLOS Y SEGURIDAD EN INTERNET .....	7
¿ES MQTT SEGURO? .....	8
ARQUITECTURA PUBLICACIÓN/SUSCRIPCIÓN DE MQTT .....	8
ESTRUCTURA DE UN MENSAJE MQTT.....	9
SINTAXIS DE LOS TOPIC .....	10
IMPLEMENTACION EN <a href="http://www.cloudmqtt.com">www.cloudmqtt.com</a> , con Software Mosquitto Client.....	12
¿CÓMO FUNCIONA LA ARQUITECTURA MQTT?.....	21
SERVICIO DE CALIDAD O QOS .....	22
QoS 0 [2] .....	22
QoS1 (al menos una vez) [2] .....	23
QoS2 (exactamente una vez) [2].....	23
UTILIZACION DE INTERFACES GRAFICAS MOBILE.....	25
APP MQTT Dash.....	25
REFERENCIAS.....	31



## PROTOCOLOS DE COMUNICACION

Un **protocolo** es un método estándar que permite la comunicación entre procesos, que se ejecutan en diferentes equipos, es decir, es un conjunto de reglas que no sólo incluyen el formato de los datos sino también los niveles de tensión, y otras características físicas que son importantes al momento de establecer una comunicación, también incluyen procedimientos que deben respetarse para el envío y la recepción de datos a través de una red.

Existen diversos protocolos de acuerdo a cómo se espera que sea la comunicación. Algunos protocolos, por ejemplo, se especializarán en el intercambio de archivos (FTP); otros pueden utilizarse por ejemplo para la petición de datos desde un servidor HTTP, y diversos otras más utilidades y aplicaciones.

En Internet, los protocolos utilizados pertenecen a una sucesión de protocolos o a un conjunto de protocolos relacionados entre sí. Este conjunto de protocolos se denomina TCP/IP.

Entre otros, contiene los siguientes protocolos: HTTP, FTP, ARP, ICMP, IP, TCP, UDP, SMTP, Telnet.

## CLASIFICACION DE PROTOCOLOS

### LOS PROTOCOLOS ORIENTADOS A CONEXIÓN

Son protocolos que controlan la transmisión de datos **durante** una comunicación establecida entre dos máquinas. En tal esquema, el equipo receptor envía acuse de recepción durante la comunicación (ACK), por lo cual el equipo remitente es responsable de la validez de los datos que está enviando. Los datos se envían entonces como flujo de datos.

TCP es un protocolo orientado a conexión.

Este tipo de protocolos, es utilizado siempre que sea necesario validar la llegada de un dato al terminal remoto, es un protocolo que garantiza que el paquete de datos, previamente dividido para ser enviado a través de la red llegue a destino, un caso de aplicación podría ser un programa de chat donde sólo se envían caracteres, es decir no hay posibilidad de montar voz y/o imagen en la misma.

Cada paquete de datos lleva consigo la necesidad de recibir un ACK desde el receptor, para que el mismo sea reconocido desde el emisor, como paquete entregado.

Sin embargo existen también protocolos que se aplican donde no es tan necesario validar la recepción del paquete de datos enviado, algo que puede resultar incoherente a la hora de la comunicación, es decir, uno se comunica con la intención que del otro lado el dato haya llegado, esto tiene explicación en la cantidad de datos que conforman la información final.



## LOS PROTOCOLOS NO ORIENTADOS A CONEXIÓN

Son un método de comunicación en el cual el equipo remitente envía datos sin avisarle al equipo receptor, y este recibe los datos sin enviar una notificación de recepción al remitente. Los datos se envían entonces como bloques (datagramas). UDP es un protocolo no orientado a conexión.

Una comunicación posible sobre UDP, es una comunicación de voz y video, cuando alguien tiene una teleconferencia por ejemplo a través de algún software con estas características, si uno de los datos no llegara, o varios de ellos no llegaran, a lo sumo se podría ver pixeles verdes sobre el rostro de alguna persona, está claro que el reconocimiento de aquella persona, o del bloque total de datos, no se verá alterada con una pérdida de información.

En síntesis.

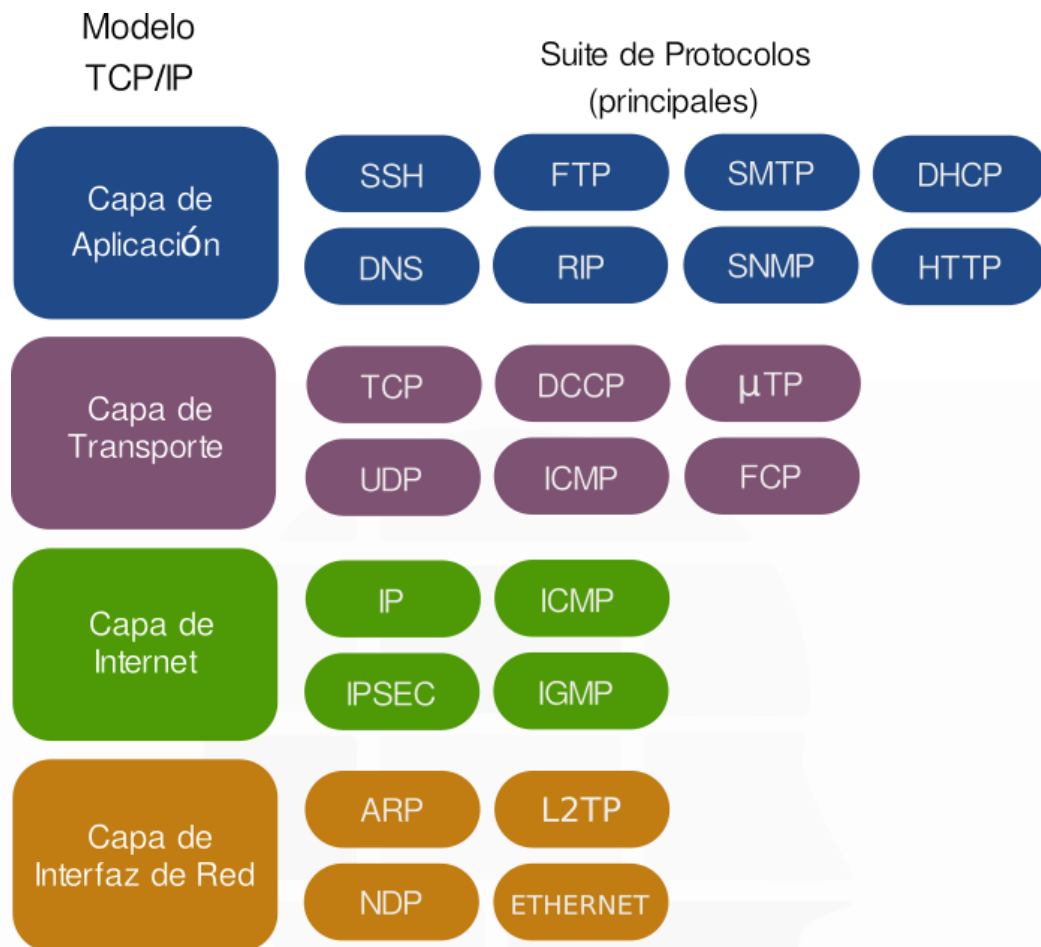
### TCP

En diferencia a UDP, el protocolo TCP está orientado a conexión. Cuando una máquina A envía datos a una máquina B, la máquina B es informada de la llegada de datos, y confirma su buena recepción. Aquí interviene el control CRC de datos que se basa en una ecuación matemática que permite verificar la integridad de los datos transmitidos. De este modo, si los datos recibidos son corruptos, el protocolo TCP permite que los destinatarios soliciten al emisor que vuelvan a enviar los datos corruptos.

### UDP

UDP es un protocolo no orientado a conexión. Es decir, cuando una maquina A envía paquetes a una maquina B, el flujo es unidireccional. La transferencia de datos es realizada sin haber realizado previamente una conexión con la máquina de destino (maquina B), y el destinatario recibirá los datos sin enviar una confirmación al emisor (la maquina A). Esto es debido a que la encapsulación de datos enviada por el protocolo UDP no permite transmitir la información relacionada al emisor. Por ello el destinatario no conocerá al emisor de los datos excepto su IP.

Cada protocolo es aplicado en distintas capas, tal como lo indica la figura siguiente, cuando conectamos una PC mediante cable a nuestro router, a nivel físico estamos implementando ETHERNET, en la capa de internet por ejemplo IP, en la capa de transporte TCP, y en la capa de aplicación HTTP por ejemplo. (siguiendo un esquema similar al del modelo OSI)



Fuente: [https://es.wikipedia.org/wiki/Familia\\_de\\_protocolos\\_de\\_internet](https://es.wikipedia.org/wiki/Familia_de_protocolos_de_internet)



## LA PILA OSI

**Nivel de Aplicación**  
Servicios de red a aplicaciones

**Nivel de Presentación**  
Representación de los datos

**Nivel de Sesión**  
Comunicación entre dispositivos de la red

**Nivel de Transporte**  
Conexión extremo-a-extremo y fiabilidad de los datos

**Nivel de Red**  
Determinación de ruta e IP (Direccionamiento lógico)

**Nivel de Enlace de Datos**  
Direccionamiento físico (MAC y LLC)

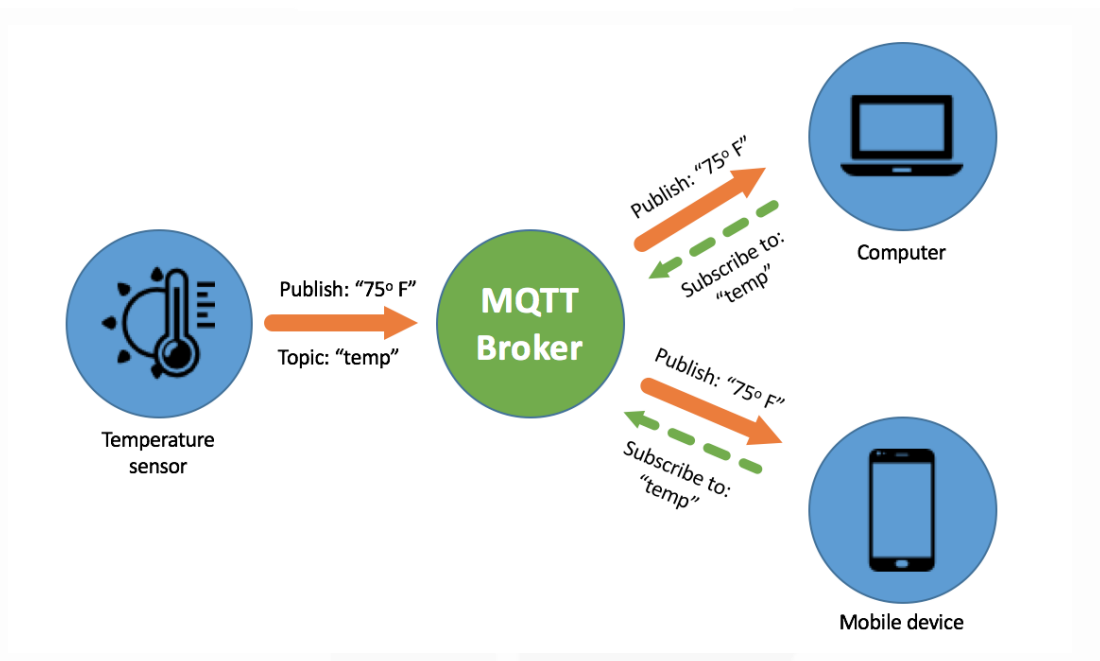
**Nivel Físico**  
Señal y transmisión binaria



## MQTT

Este es un protocolo reducido que corre sobre TCP/IP que permite que las “máquinas” hablen entre sí, de ahí su tipo, M2M (del inglés *Machine to Machine*).

En un extremo tenemos un usuario final, o endpoint, es decir un dispositivo capaz de capturar información a través de sensores, información como la temperatura, presión, humedad, niveles o cualquier otra magnitud que uno desee medir de manera directa o indirecta; y del otro lado, un terminal que recolecta dichos datos, o broker, y que además vincula a otros endpoints, que desean conocer que paso con aquella medición de temperatura, presión o humedad realizado por el primero, tal como lo muestra la figura.



Dada la gran cantidad de dispositivos IoT, hay una necesidad real de un protocolo muy ligero, que consuma muy poco ancho de banda y que permita comunicarse a través de la publicación/suscripción para tener una comunicación bidireccional real con acuses de recibo, ágil.

Esto permite que los dispositivos pasen de tener una iteración punto a punto a una iteración más sofisticada donde se establezcan verdaderos diálogos entre las máquinas.

Aquí es donde MQTT cobra sentido, dado que es un protocolo que permite eso: publicación y suscripción de mensajes, comunicación bidireccional y acuses de recibo de dichos mensajes.



## CARACTERISTICAS PRINCIPALES

- ✓ Protocolo de comunicación asincrónico.
- ✓ Baja cantidad de bits en los encabezados
- ✓ Modelo de Publish/Subscribe (PubSub model)
- ✓ Corre sobre protocolo orientado a la comunicación (TCP)

Uno de sus puntos fuertes es que es extremadamente simple y ligero.

Por este motivo es muy interesante para sistemas que requieren poco ancho de banda, tienen una alta latencia y requieren de poco consumo de los dispositivos.

Los objetivos del protocolo MQTT es minimizar el ancho de banda, la comunicación bidireccional entre dispositivos, y además minimizar los requerimientos de los dispositivos tanto recursos como consumo y garantizar la fiabilidad, dado que no es necesario que constantemente este peticionando sobre una base de datos, a esto se le agrega la seguridad, en el caso que desee implementarse de este modo.

## PROTOCOLOS Y SEGURIDAD EN INTERNET

Los protocolos son desarrollados para transmitir información, dicha información en muchos casos es crítica por lo que existen estándares de seguridad a los cuales someter dichos protocolos, los siguientes principios están elaborados a partir de las recomendaciones del **U.S. Department of Homeland Security, OWASP IoT Project** y la **IoT Security Foundation**.

Estos son un conjunto de mejores prácticas de carácter general cuyo objetivo es reducir el riesgo de que un despliegue IoT se vea comprometido y en consecuencia se pueda revelar información sensible, se interrumpa el funcionamiento del sistema o se utilice su infraestructura para enviar un ataque de DDoS a un tercero.

### REQUISITOS [1]

#### Incorporar la seguridad en la fase de diseño

- ✓ Evitar usuarios y contraseñas por defecto, especialmente si estas credenciales son iguales
- ✓ para todos los dispositivos.
- ✓ En el caso de dispositivos con sistema operativo, utilizar distribuciones actualizadas.

#### Considera la seguridad física del dispositivo

- ✓ Implementar medidas anti-tampering en el dispositivo
- ✓ No guardar información sensible o personal en elementos de memoria extraíble
- ✓ Guardar de forma segura credenciales en el dispositivo.

#### Permitir actualizaciones remotas de software

- ✓ Asignar una identidad segura a los dispositivos.



- ✓ Disponer de un canal de comunicación seguro para notificar la actualización.
- ✓ Proporcionar un canal seguro para realizar la descarga de la actualización.
- ✓ Incluir un mecanismo de validación de la integridad del paquete de actualización.

#### **Implementar mejores prácticas de la industria**

- ✓ Conocer y participar en los foros de la industria para intercambio de mejores prácticas.
- ✓ Participar en las plataformas de difusión de vulnerabilidades.

#### **Conectar de forma consciente**

- ✓ Deshabilitar las interfaces que no tengan utilidad en el despliegue.
- ✓ Proporcionar información a los usuarios de la utilidad de la conectividad y las interfaces.
- ✓ Permitir al usuario deshabilitar interfaces de comunicación cuando no sean necesarias.

### **¿ES MQTT SEGURO?**

Si un equipo esté conectado a Internet, no podemos garantizar la seguridad al 100%. Sobre esa base, MQTT soporta cifrado mediante SSL.

Por lo tanto, podemos enviar datos cifrados a través de la red utilizando este protocolo. Eso sí, se debe pagar un precio.

Cuando se utiliza cifrado SSL se añade una sobrecarga debido a un extra en la cantidad de datos a la red y los microcontroladores, que se utilicen para implementar el sistema embebido, haciendo que MQTT ya no sea un protocolo tan ligero y simple, sin embargo, el más apropiado para este tipo de aplicaciones.

### **ARQUITECTURA PUBLICACIÓN/SUSCRIPCIÓN DE MQTT**

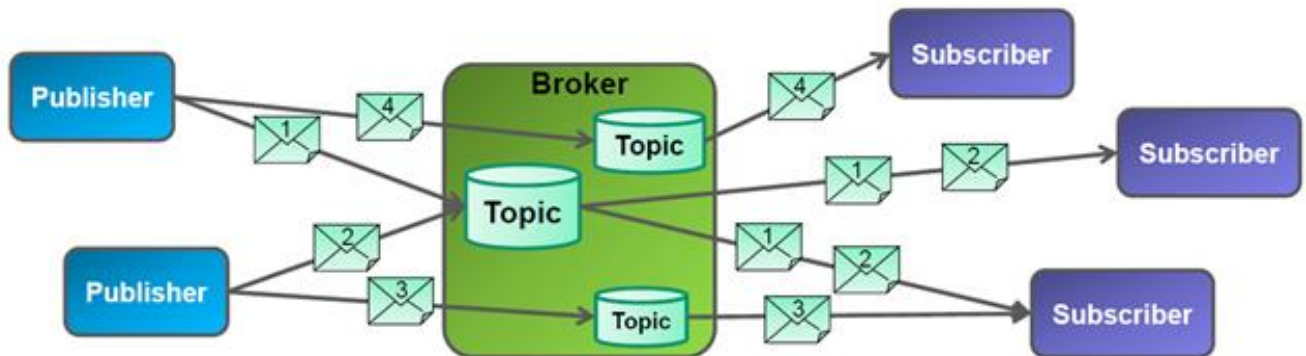
Se trata de una arquitectura basada en eventos. Cada mensaje se envía a los receptores que se hayan suscrito a una publicación concreta. El Broker se encarga de distribuir los mensajes a los receptores.

El topic es el tema donde se suscriben los receptores para recibir el mensaje. Veamos un ejemplo de una comunicación típica para entender el concepto de arquitectura publicación/suscripción.

Un cliente se suscribe a un topic que es como el tema, a quién va dirigido el mensaje. Podemos entenderlo como el asunto de un email. Un mensaje no tiene destinatario concreto, puede haber uno, muchos o ninguno.

El emisor, el que envía el mensaje, no sabe a quién va dirigido dicho mensaje, sólo el broker lo sabe. Cuando llega un nuevo mensaje se lo envía a aquellos que se han suscrito al topic.





El mensaje puede ser cualquier cosa, una temperatura, V o F, un texto, ON OFF, el formato es totalmente libre, lo decide el programador y el tamaño máximo depende de la implementación del protocolo MQTT y no de la especificación.

### ESTRUCTURA DE UN MENSAJE MQTT

Lo más importante dentro del protocolo MQTT son los mensajes. Se envían de forma asíncrona, es decir, no hay un reloj indicando una base de tiempo para sincronizar la llegada y salida de información y no hay que esperar respuesta una vez que se envía un mensaje.

Cada mensaje consta de 3 partes:

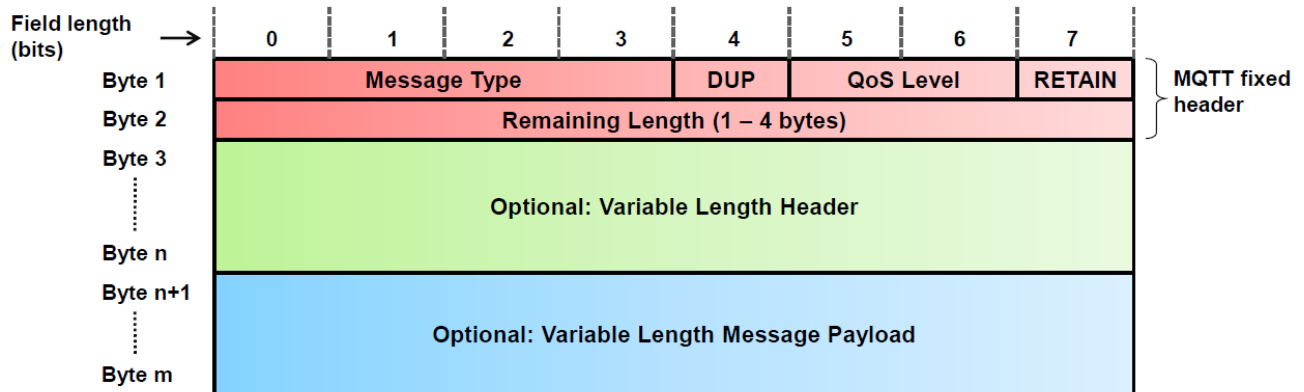
**Encabezado fijo.** Ocupa sólo 2 bytes y es obligatorio enviar esta parte en todos los mensajes.

**Encabezado variable.** Ocupa 4 bytes y no es obligatorio que esté en todos los mensajes.

**Mensaje o carga útil (*payload*).** Puede tener un máximo de 256 Mb, aunque en implementaciones reales el máximo es de 2 a 4 kB.



MQTT uses network byte and bit ordering.



El MQTT SERVER => **BROKER**

Es quien corre los topics , recibe pedidos de suscripción de los clientes a los topics, recibe mensajes desde los clientes y re direccionan , basado en la suscripción de los clientes

El MQTT CLIENT => **PUBLISHER, SUBSCRIBER**

Los clientes se subscriben a los topics, para publicar y recibir mensajes.



## SINTAXIS DE LOS TOPIC

El símbolo / es un separador de niveles. Siempre se pone para separar cada uno de los niveles. Cuando trabajamos con un dispositivo IoT utilizamos otros nombres para los topics. Por ejemplo, podríamos tener algo como esto para medir la temperatura y/o humedad:

/casa/comedor/temperatura  
/casa/comedor/humedad  
/casa/cocina/temperatura  
/casa/cocina/ humedad  
/casa/dormitorio/temperatura  
/casa/dormitorio/humedad

Es un protocolo intuitivo desde la escritura, el primero de los topic, indica

Además, existen comodines como el símbolo + y #.

El símbolo + se sustituye por cualquier nivel. Si por ejemplo nos queremos suscribir a todos



los topic de temperatura podemos hacerlo de la siguiente manera

**/casa/+/temperatura**

El símbolo + se sustituirá por cada nivel que tenga como nivel superior casa y como nivel inferior temperatura.

El símbolo # también es un comodín. Este símbolo cubre los niveles que estén por debajo. Por ejemplo, si quieres suscribirte a todos los mensajes que se envían a casa sólo tienes que hacer lo siguiente:

**/casa/#**

El topic anterior nos enviará cualquier mensaje que se envíe a casa o a cualquier nivel que esté por debajo. Es recomendable utilizar topics, del formato anterior ya que nos permite escalabilidad.

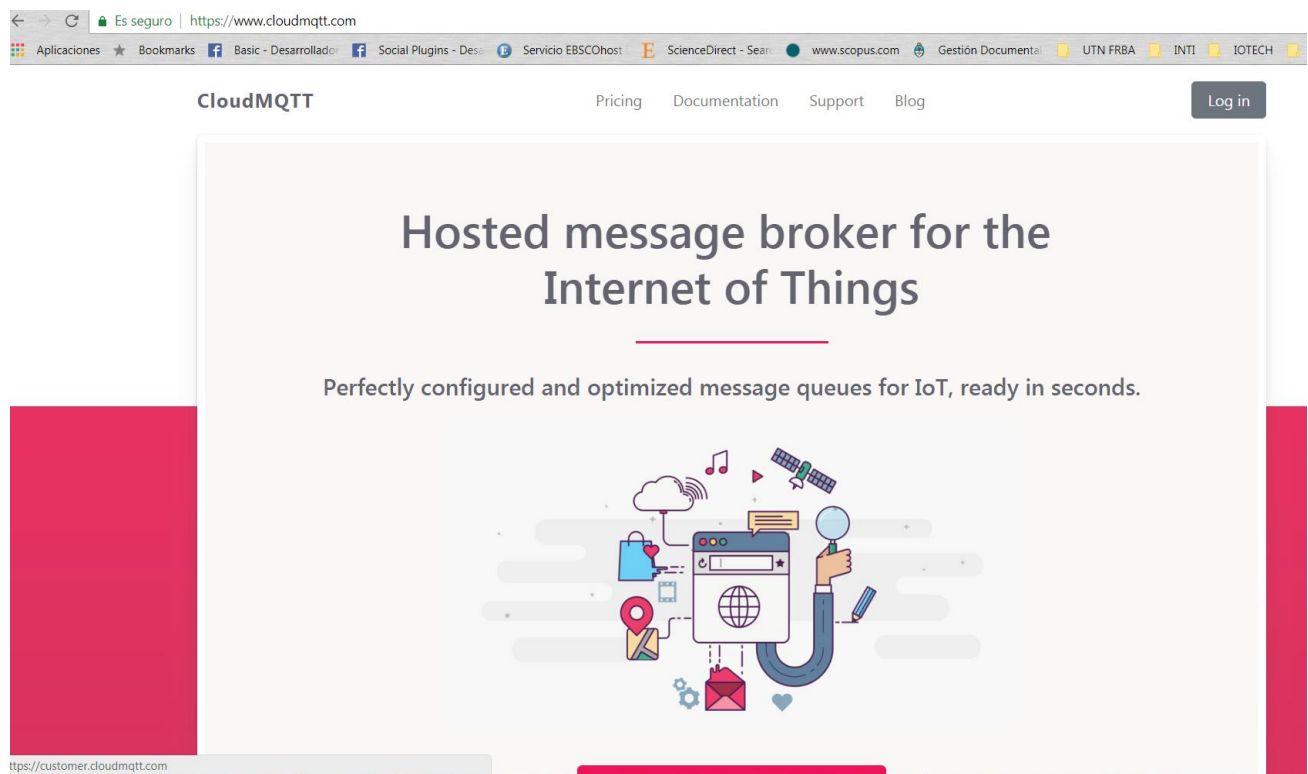
Cada bit del mensaje está estudiado cuidadosamente para que se consuma el menor ancho de banda posible. No hace falta aprender todo esto de memoria. Cuando utilizamos una librería o un software, él ya se encarga de crear el mensaje según una configuración por nosotros.

De lo único que tenemos que preocuparnos es del mensaje o payload, es decir, qué queremos enviar al receptor y recibir de él, a través del Broker.



## IMPLEMENTACION EN [www.cloudmqtt.com](https://www.cloudmqtt.com) , con Software Mosquitto Client

Para poder realizar pruebas y no quedarnos sólo con los conceptos teóricos vamos a utilizar las cuentas gratuitas que nos ofrecen desde el sitio <https://www.cloudmqtt.com/>




Primeramente, debemos amarnos una cuenta, asociando un nombre y una dirección de email, valido para que luego de la validación podamos utilizar este bróker.



Es seguro | <https://customer.cloudmqtt.com/instance/create>

Aplicaciones Bookmarks Basic - Desarrollador Social Plugins - Des Servicio EBSCOhost ScienceDirect - Sea www.scopus.com Gestión Documental UTN FRBA INTI IOTech

**CloudMQTT** List all instances  marcospoliti@outlook.com

### Create new instance

**Name** casa001


**Plan** Cute Cat (Free)

**Data center** US-East-1 (Northern Virginia)

**Tags** Type or click here  
Admins can manage tag access control.

Create New Instance

**Plan**



Cute Cat

See the [plan page](#) to learn about the different plans.

**MENU**

- Home
- Plans
- Documentation
- Blog

**MORE**

- Status
- Terms of Service
- Program Policies
- Privacy Policy

**CloudMQTT**

Contact Support

Una vez dentro, creamos una nueva instancia con el nombre que deseamos en este caso **casa001**.

Utilizamos la cuenta CuteCat (free), estas cuentas tienen sus limitaciones en cantidad de pedidos por minuto, con lo cual, podemos observar que la misma no funciona tan rápidamente como deseamos si agotamos la cantidad de pedidos que el bróker nos autoriza por brindarnos la cuenta gratuita.


Actualmente se encuentre suspendido el dispendio de cuentas Free, por lo que salvo que la Instancia haya sido creada antes de Abril de 2020, se deberá pagar 5 dólares por mes para el primer servicio compartido.

cloudmqtt.com/plans.html

ScienceDirect - Sea... www.scopus.com UTN FRBA IOT IOTECH Marcos Politi uC Programacion Web ELT ER ScienceDirect CETAR UNSAM

### Shared Instances

For development or small hobby projects. Not recommended for production due to variable performance.



#### Humble Hedgehog

- 25 users/acl rules/connections
- 20 Kbit/s
- 3 bridges
- Support by e-mail

**\$ 5**  
PER MONTH

Get Now



Si deseáramos mejorar, pues sólo debemos actualizar el plan, a partir de allí debemos comenzar a pagar, con lo cual es obligatorio la carga de los datos de una tarjeta de crédito.

The screenshot shows the CloudMQTT console interface. The browser address bar displays <https://api.cloudmqtt.com/console/9849069/details>. The page title is "Details" and the instance name is "casa001". The left sidebar contains a menu with options: DETAILS, SETTINGS, CERTIFICATES, USERS & ACL, BRIDGES, AMAZON KINESIS STREAM, WEBSOCKET UI, CONNECTIONS, and LOG. The main content area is titled "Details" and contains an "Instance info" section with the following details:

Field	Value	Action
Server	m12.cloudmqtt.com	
User	xawuhkig	<button>Restart</button>
Password	PL-GXqebqWJp	<button>Rotate</button>
Port	11321	
SSL Port	21321	
Websockets Port (TLS only)	31321	
Connection limit	5	


Below the instance info, there is a "Reset DB" section with a warning: "This will erase all stored messages and sessions. The instance will be restarted." and a Reset DB.

On the right side, there is an "Active Plan" section featuring a cat icon, the name "Cute Cat", and an Upgrade Instance.

Una vez allí podemos observar los datos más importantes de la cuenta. Y con los cuales nos vamos a comunicar con dicho Broker.



## Instance info

**Server** m12.cloudmqtt.com**User** xawuhkig Restart**Password** PL-GXqebqWJp Rotate**Port** 11321**SSL Port** 21321**Websockets Port (TLS only)** 31321**Connection limit** 5

Luego de esto debo generar un usuario.

### Users and ACL

API Access

#### Users

**Name**  

casa001

EditDelete

En nuestro caso **casa001**, para luego poder escalar, luego más abajo en la misma página, vamos agregando los distintos topics,



• Use # for multi level wildcard acl.  
• Use + for single level wildcard acl.  
• Creating and deleting users and ACLs are asynchronous tasks and may take up to a minute. Poll list APIs to see when ready.  
For API docs look at HTTP API

Type	Pattern	Read/Write	
topic	casa001 - /cocina/temperatura	true/true	Delete
topic	casa001 - /cocina/humedad	true/true	Delete

PatternTopic

casa001

/comedor/humedad

☒ Read Access?  
☒ Write Access?

+ Add

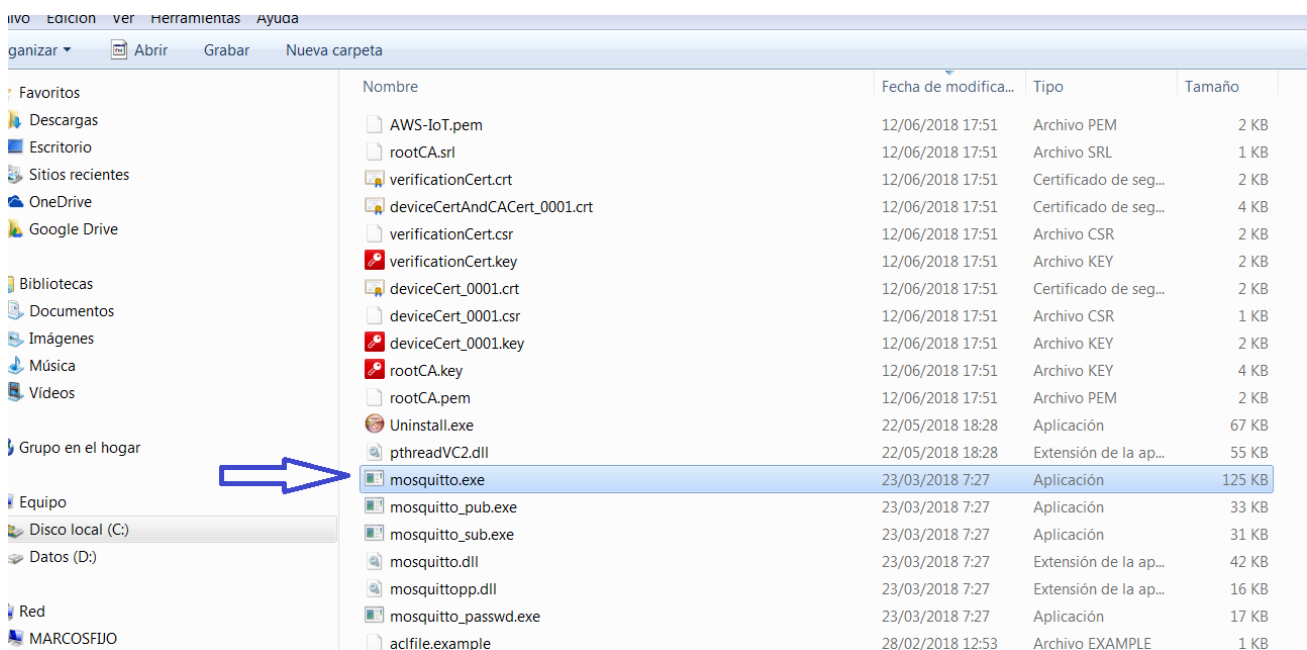
Así sucesivamente con todos los topics, como es una cuenta CuteCat, sólo nos dejan establecer 5 topics. En otro caso la cantidad de topics es ilimitada.





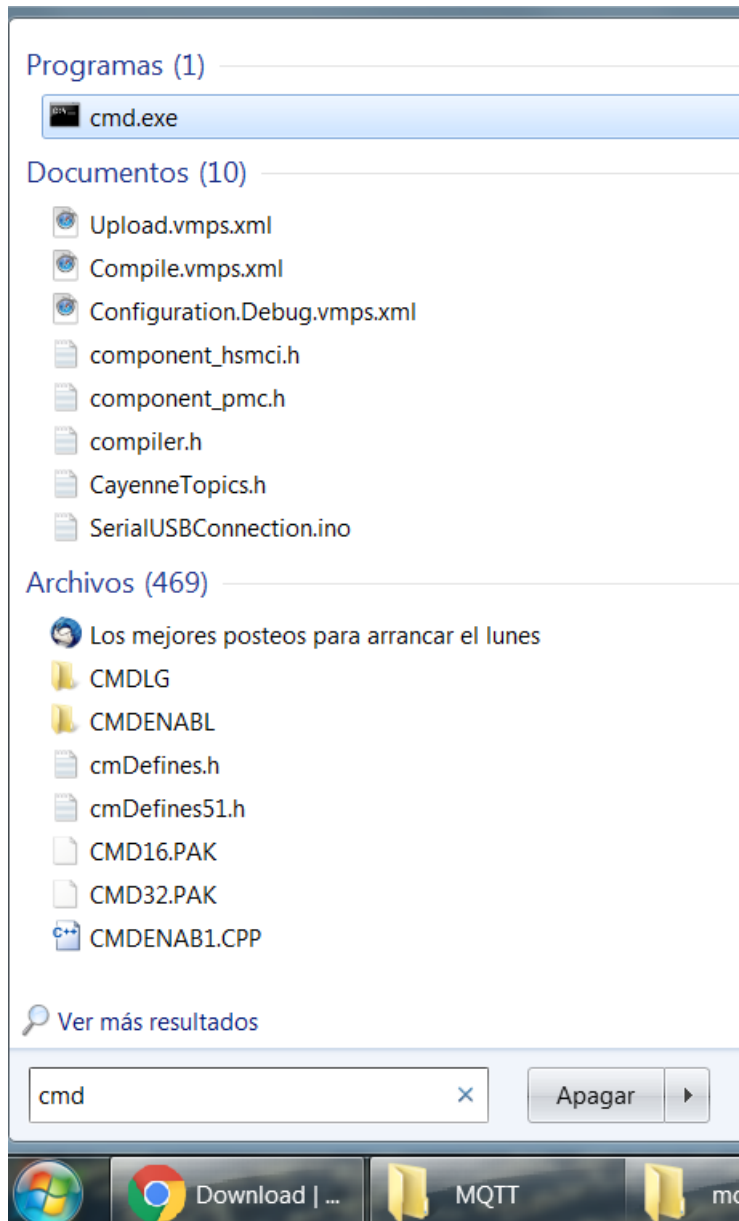
Una vez establecidos debemos proceder a instalar el mosquito en nuestra PC, para tal fin deben acceder al sitio, <https://mosquitto.org/download/> y de ahí descargar la versión que corresponda a su sistema operativo.

Una vez instalado, deben cerciorarse de que el mismo esté en funcionamiento, para eso, debemos acceder a la carpeta donde fue instalado el mismo y ejecutar **mosquito.exe**, el mismo pareciera que desaparece, pero queda funcionando en segundo plano.



Luego de esto comenzaremos a proceder para correr el **mosquitto\_pub** para poder publicar en el bróker de MQTT elegido, (cabe destacar que este proceso, es la emulación de la generación de un dato de un sistema embebido, luego no tendremos que usar más este software, sino que los datos serán enviados por nuestros microcontroladores conectados a internet esto se detallará en las próximas clases)

Luego de esto debemos escribir **cmd** en nuestro buscador





Y escribir

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Marcos>cd..

C:\Users>cd..

C:\>cd Program Files (x86)

C:\Program Files (x86)>cd mosquito

C:\Program Files (x86)\mosquito>_
```

Luego una vez posicionado en la carpeta donde se instaló el emulador, se tipea:

```
C:\Program Files (x86)\mosquito>mosquito_pub -h "m12.cloudmqtt.com" -p "11321"
-u "xawuhkig" -P "PL-GXqebqWJp" -t "/cocina/temperatura " -m "25"

C:\Program Files (x86)\mosquito>mosquito_pub -h "m12.cloudmqtt.com" -p "11321"
-u "xawuhkig" -P "PL-GXqebqWJp" -t "/cocina/temperatura " -m "30"

C:\Program Files (x86)\mosquito>mosquito_pub -h "m12.cloudmqtt.com" -p "11321"
-u "xawuhkig" -P "PL-GXqebqWJp" -t "/cocina/temperatura " -m "HOLA UTN!!!"

C:\Program Files (x86)\mosquito>
```

Observen, que para que funcione el emulador necesita:

- h: Es el HOST.
- p: Es el PUERTO.
- u: Es el USUARIO.
- P: Es el PASSWORD.
- t: Es el TOPIC donde publicar.
- m: Es el MENSAJE a enviar.



Y recuerden que estos salieron de aquí.

## Instance info

Server	m12.cloudmqtt.com	
User	xawuhkig	<button>Restart</button>
Password	PL-GXqebqWJp	<button>Rotate</button>
Port	11321	
SSL Port	21321	
Websockets Port (TLS only)	31321	
Connection limit	5	

Luego en [www.cloudmqtt.com](http://www.cloudmqtt.com) abro la consola **Websocket** y ahí puedo observar,

### Websocket

```
C:\Program Files (x86)\mosquitto>mosquitto_pub -h "m12.cloudmqtt.com" -p "11321" -u "xawuhkig" -P "PL-GXqebqWJp" -t "/cocina/temperatura" -m "25"
C:\Program Files (x86)\mosquitto>mosquitto_pub -h "m12.cloudmqtt.com" -p "11321" -u "xawuhkig" -P "PL-GXqebqWJp" -t "/cocina/temperatura" -m "30"
C:\Program Files (x86)\mosquitto>mosquitto_pub -h "m12.cloudmqtt.com" -p "11321" -u "xawuhkig" -P "PL-GXqebqWJp" -t "/cocina/temperatura" -m "HOLA UTN!!!"
C:\Program Files (x86)\mosquitto>
```

### Received messages

Topic	Message
/cocina/temperatura	25
/cocina/temperatura	30
/cocina/temperatura	HOLA UTN!!!

Obteniendo así la conexión a dicho bróker.



Ahora supongamos que otro sistema embebido esta funcionando como suscriptor de estos topics, es decir, a este, le llegaran estos cambios en los valores. Y no como en el caso anterior que el emulador de sistemas embebidos publicaba en él.

Para este caso debemos ejecutar en otra ventana **mosquitto\_sub**, con los mismos atributos que en **\_pub**

The screenshot shows the CloudMQTT Websocket interface. On the right, a table titled 'Received messages' displays the following data:

Topic	Message
/cocina/temperatura	20
/cocina/temperatura	45
/cocina/temperatura	SI
/cocina/temperatura	ANDA!!!

On the left, a terminal window titled 'Administrador: Símbolo del sistema - mosquitto\_sub' shows the command execution and the received messages:

```
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\Marcos>cd..
C:\Users>cd..
C:\>cd Program Files (x86)
C:\Program Files (x86)>cd Mosquitto
C:\Program Files (x86)\Mosquitto>mosquitto_sub -h "m12.cloudmqtt.com" -p "11321"
-u "xawuhkig" -P "PL-GXqebqlJp" -t "/cocina/temperatura"
20
45
SI
ANDA!!!
```

## ¿CÓMO FUNCIONA LA ARQUITECTURA MQTT?

Una de las características más importantes del protocolo MQTT es que los clientes o nodos no dependen unos de otros, dado que no tienen conocimiento de quién está al otro lado. Puede incluso que no haya nadie en el otro extremo, y uno de ellos esté publicando constantemente a un bróker, del mismo modo podría un suscriptor estar online, y no recibir datos dado que no existe un publicador conectado.

Esto permite algo muy importante en proyectos de IoT este tipo: la escalabilidad.

Al contrario de lo que ocurre con el protocolo HTTP, no hay que hacer una petición para recibir información desde un cliente, en muchos casos esto es realizado a través de una API, pero en el caso de MQTT, Cada cliente MQTT abre una conexión permanente TCP con el broker.

Estas peticiones a través de los métodos GET por ejemplo, además de ser poco eficientes desde el punto de vista temporal, requieren un sistema dedicado constantemente al refresco de la información.

El Broker tiene la capacidad de hacer que los mensajes sean persistentes, guardando el mensaje hasta que se conecte el cliente al que va dirigido, aunque no necesariamente la aplicación utilizada aprovecha dicha situación.



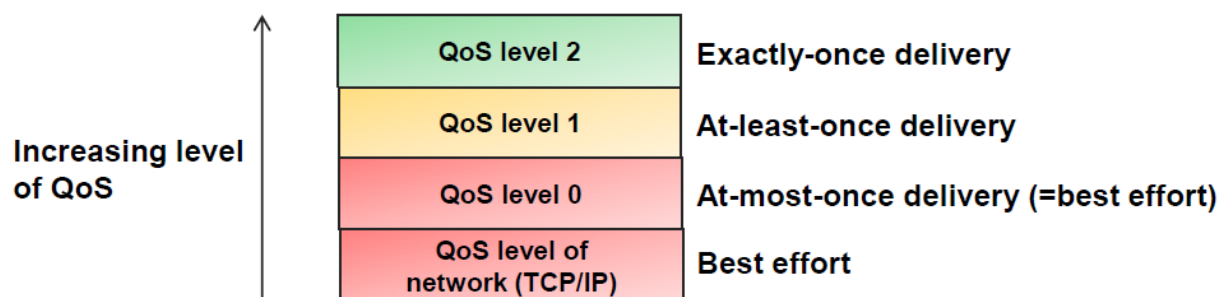
## SERVICIO DE CALIDAD O QOS

MQTT es un protocolo fiable. Eso se debe a que tiene implementado un Servicio de Calidad o QoS (del inglés *Quality of Service*), este servicio determina cómo se entrega el mensaje a los receptores.

El QoS se especifica en cada mensaje que se envía y puede haber 3 grados de calidad:

- ✓ QoS 0: como máximo una vez. Esto implica que puede que no se entregue.
- ✓ QoS 1: al menos una vez. Se garantiza la entrega, pero puede que duplicados.
- ✓ QoS 2: exactamente una vez. Se garantiza que llegará una vez el mensaje.

Utilizar un grado de calidad u otro dependerá de la fiabilidad que queramos tener en nuestro sistema. Eso sí, se debe tener en cuenta que cuanto más calidad menor será el rendimiento.



### QoS 0 [2]

En este nivel no hay garantía de entrega. El destinatario no acusa recibo del mensaje y el remitente no almacena ni transmite el mensaje.

El nivel de calidad de servicio 0 a menudo se denomina "disparar y olvidar".



Fuente: HiveMQ



### QoS1 (al menos una vez) [2]

QoS nivel 1 garantiza que el mensaje se entrega al menos una vez al receptor.

El remitente almacena el mensaje hasta que recibe un paquete PUBACK del receptor que acusa recibo del mensaje.

Es posible que se envíe o entregue un mensaje varias veces.

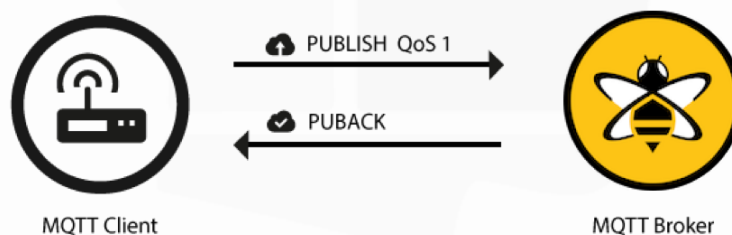
El remitente utiliza el identificador de paquete en cada paquete, para verificar si coincide el del PUBLISH con el del paquete PUBACK correspondiente.

Si el remitente no recibe un paquete PUBACK en un período de tiempo razonable, el remitente reenvía el paquete PUBLISH.

Cuando un receptor recibe un mensaje con QoS 1, puede procesarlo de inmediato. Por ejemplo, si el receptor es un intermediario, el intermediario envía el mensaje a todos los clientes suscriptores y luego responde con un paquete PUBACK.

Si el cliente de publicación envía el mensaje nuevamente, establece un indicador de duplicado (DUP).

En QoS 1, este indicador DUP solo se usa para fines internos y no es procesado por el broker o el cliente. El receptor del mensaje envía un PUBACK, independientemente de la bandera DUP.



Fuente: HiveMQ

### QoS2 (exactamente una vez) [2]

QoS 2 es el nivel más alto de servicio en MQTT. Este nivel garantiza que cada mensaje sea recibido solo una vez por los destinatarios previstos.

QoS 2 es el nivel de calidad de servicio más seguro y lento.

La garantía es proporcionada por al menos dos flujos de solicitud / respuesta (un handshake de cuatro etapas) entre el remitente y el receptor.



El remitente y el receptor usan el identificador de paquete del mensaje PUBLISH original para coordinar la entrega del mensaje.

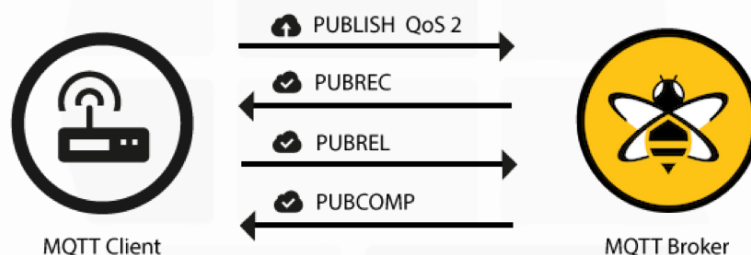
Una vez que el remitente recibe un paquete PUBREC del receptor, el remitente puede descartar de manera segura el paquete PUBLISH inicial.

El remitente almacena el paquete PUBREC del receptor y responde con un paquete PUBREL.

Después de que el receptor recibe el paquete PUBREL, puede descartar todos los estados almacenados y responder con un paquete PUBCOMP (lo mismo ocurre cuando el remitente recibe el PUBCOMP).

Hasta que el receptor complete el procesamiento y envíe el paquete PUBCOMP al remitente, el receptor almacena una referencia al identificador del paquete PUBLISH original. Este paso es importante para evitar procesar el mensaje por segunda vez.

Después de que el remitente recibe el paquete PUBCOMP, el identificador del paquete del mensaje publicado queda disponible para su reutilización.



Fuente:HiveMQ





## UTILIZACION DE INTERFACES GRAFICAS MOBILE

### APP MQTT Dash

Lo anterior si bien es muy útil para comprender el tema MQTT, y a sabiendas que es la base del conocimiento para aplicar con microcontroladores como algún ATmel de la línea Arduino o el ESP8266, o con cualquier otro microcontrolador, cierto es que no se ve muy elegante.

Uno desea poder implementar este tipo de soluciones en plataformas gráficas acondicionadas para tal fin. Para eso vamos a valernos de App que se pueden descargar de manera gratuita en cualquier Store, por ejemplo, el de Google como es el caso de por ejemplo MQTT Dash.

Para tal fin debo crear un nuevo Dashboard, para controlar y monitorear mis equipos IoT, para tal fin debo contar con todos los datos de cloudmqtt.com en la instancia donde estoy trabajando.

Una vez configurada, debo proceder a agregar objetos con sus datos correspondientes.



Una vez completo los datos:

**MQTT Dash**

Default (automatically connect on start up).  
Note: this option is useful if you have just one connection configured.

☐ If you have more than one connection, you can create home screen shortcut for every connection.  
To create shortcut long press on any connection in connections list.

☒ Keep screen on when connected to this broker

Allow metrics management. If disabled, you can't add, edit, delete or rearrange metrics. This serves as protection from unintentional metrics changing.

Name  
Dashboard Casa

Address  
m12.cloudmqtt.com

Port  
11321

Enable connection encryption (SSL/TLS).  
Note: if server certificate is self-signed, you need to install it to your device or enable option below, otherwise connection will fail. If server

**MQTT Dash**

11321

Enable connection encryption (SSL/TLS).  
Note: if server certificate is self-signed, you need to install it to your device or enable option below, otherwise connection will fail. If server

☐ certificate issued by a known Certificate Authority (CA), it will work out of box, without installing to you device. Also don't forget, that MQTT servers have different ports for plain and SSL/TLS connections.

☐ This broker uses self-signed SSL/TLS certificate. I trust this certificate at my own risk.

User name  
xawuhkig

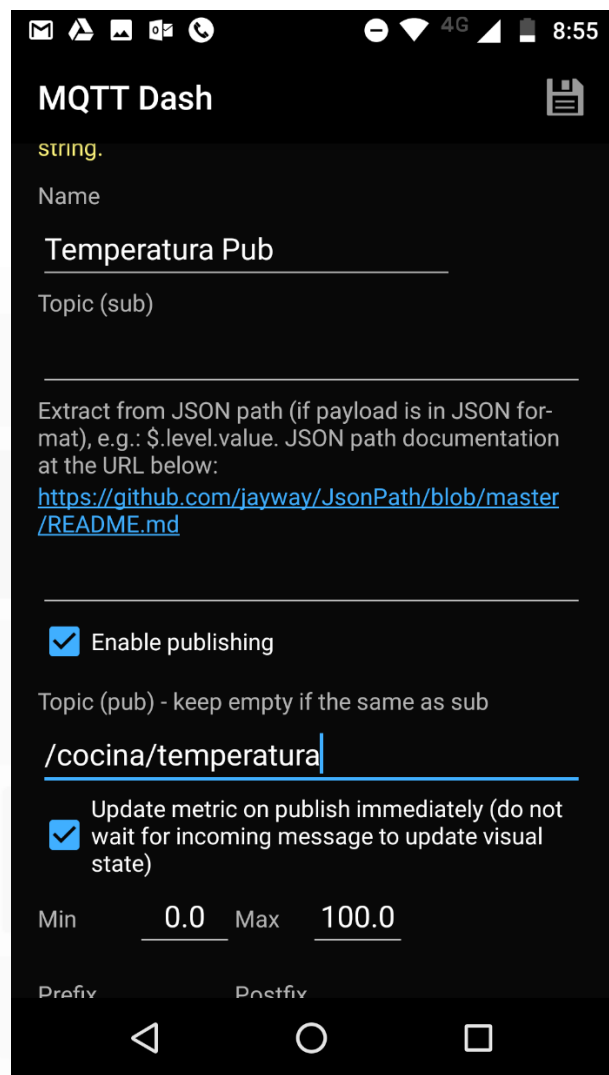
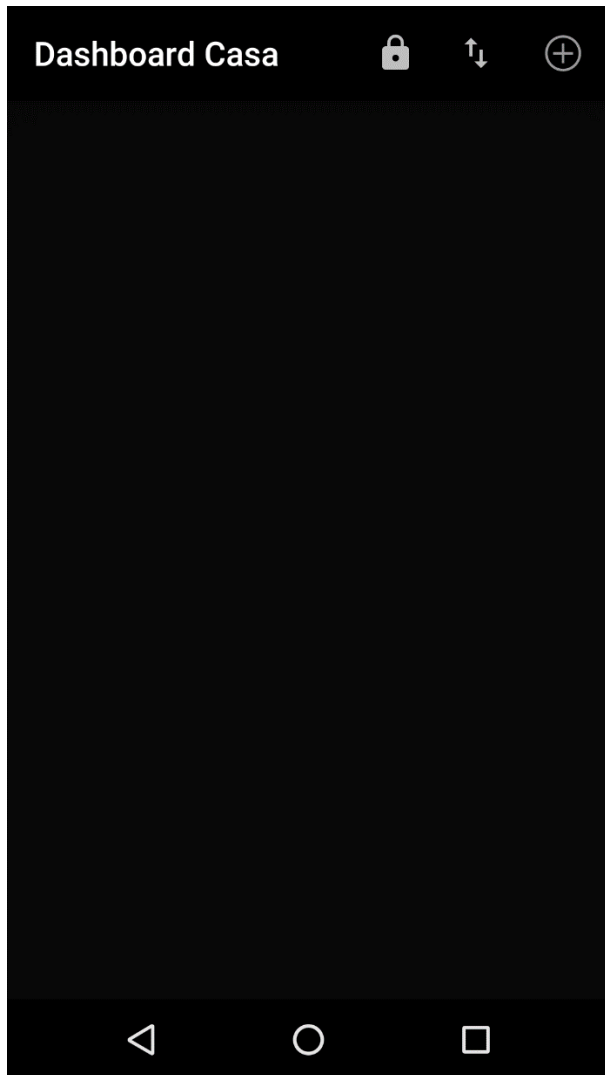
User password  
.....

Client ID (must be unique)  
mqttdash-3157358f

Tile size  
☐ Small  
☒ Medium

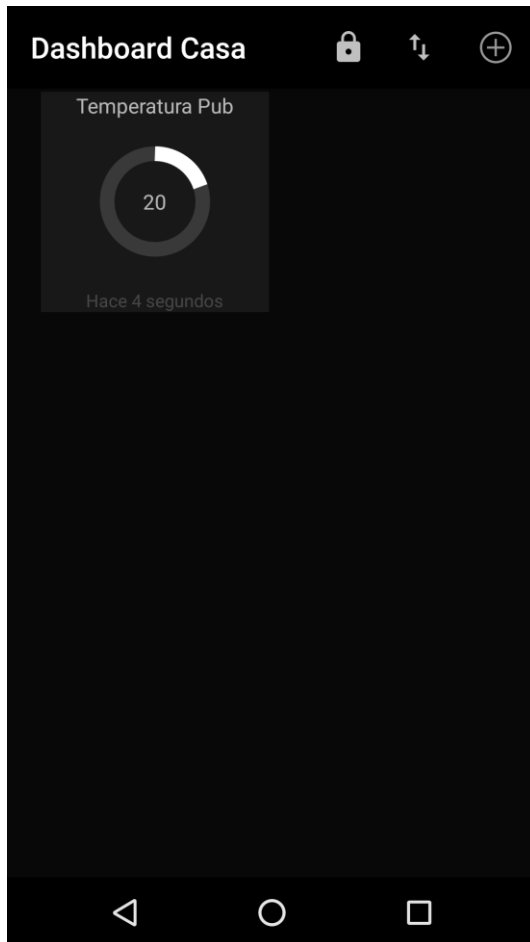


Pasamos al Dashboard creado, que en principio aparece vacío, con lo cual, presionamos sobre el icono +, y agregamos un objeto, en este caso para **Publicar** en mi instancia de **cloudmqtt.com**





Una vez creado, guardamos, y procedemos a cargar un valor:



Observando en el Websocket:

CloudMQTT casa001 marcospoliti@outlook.com

### Websocket

#### Send message

Topic

Message

Send

#### Received messages

Topic	Message
/cocina/temperatura	20



Posteriormente y con el objetivo de poder **Subscribirme** al topic **/cocina/temperatura**, creo otro objeto.

MQTT Dash

This metric is intended for displaying payload text (e.g. temperature displaying). Payload is expected to be string.

Name

Temperatura Sub

Topic (sub)

/cocina/temperatura

Extract from JSON path (if payload is in JSON format), e.g.: \$.level.value. JSON path documentation at the URL below:  
<https://github.com/jayway/JsonPath/blob/master/README.md>

☐ Enable publishing

Prefix Postfix

Main text size

☐ Small

☐ Medium





Observando nuevamente:

CloudMQTT casa001 marcospoliti@outlook.com

### Websocket

#### Send message

Topic

Message

Send

#### Received messages

Topic	Message
/cocina/temperatura	20
/cocina/temperatura	32

Clear session

Mg.Ing. Marcos Politi



## REFERENCIAS

- [1] Documentos, Maestría en Internet de las Cosas Universidad de Salamanca.
- [2] <https://www.hivemq.com/>
- [3] <http://mqtt.org/>
- [4] [http://www.future-internet.eu/fileadmin/documents/reports/Cross-ETPs\\_FI\\_Vision\\_Document\\_v1\\_0.pdf](http://www.future-internet.eu/fileadmin/documents/reports/Cross-ETPs_FI_Vision_Document_v1_0.pdf)
- [5] <https://www.computer.org/csdl/mags/ic/2013/04/mic2013040018-abs.html>
- [6] <http://www.domodesk.com/a-fondo-que-es-el-internet-de-las-cosas>
- [7] <http://ecixgroup.com/el-grupo/una-aproximacion-algunos-elementos-de-internet-de-las-cosas/>