

Introduction to MQTT

Marco Zennaro, PhD

TinyML4D Academic Network Co-Chair



All the credit goes to Prof. Pietro Manzoni:



IoT and edge computing: stories of a necessary relationship

Pietro Manzoni

- Universitat Politècnica de València (UPV)
- Valencia - SPAIN
- pmanzoni@disca.upv.es

<http://grc.webs.upv.es/>

The screenshot shows the top part of the GRC website. It includes the GRC logo (GRUPO DE REDES DE COMPUTADORES) and the Universitat Politècnica de València logo. A navigation menu contains links for Home, Members, Papers, Projects, Teaching, Software, and Internal. Below the menu, there is a paragraph about the group's history and a list of research areas. There are also sections for 'Infos and News' and 'Events and CfPs' with various links and dates.

The **Networking Research Group** (GRC - Grupo de Redes de Computadores) of the Universitat Politècnica de València (UPV) was founded in 2000 and it is mainly composed of researchers of the Computer Engineering Department (DISCA). It keeps strong bonds and collaborations with other researchers in the same area in Spain and in the rest of the world.

The group research efforts are focused on offering **Data Communication Solutions for Mobile Systems**. The main areas of application are:

- AIoT infrastructures for environmental sustainability
- Drone-based networks
- Efficient IoT infrastructures development
- Intelligent Transport Systems
- LPWAN-based networks
- Mobile edge computing
- Pub/Sub systems
- Social sensing

Infos and News:

- [Overview of GRC research \[Sept. 2021\]](#)
- [GRC YouTube channel](#)
- [COVIDsensing: a tool to analyze COVID spreading using AI](#)

Events and CfPs:

Conferences:

- GoodIT, International Conference on Information Technology for Social Good, September 7-9, 2022, Limassol (CYPRUS).
- NET4us, Workshop on Networked sensing systems for a sustainable society, during SIGCOMM 2022, August 22-26, 2022, Amsterdam (Netherlands).

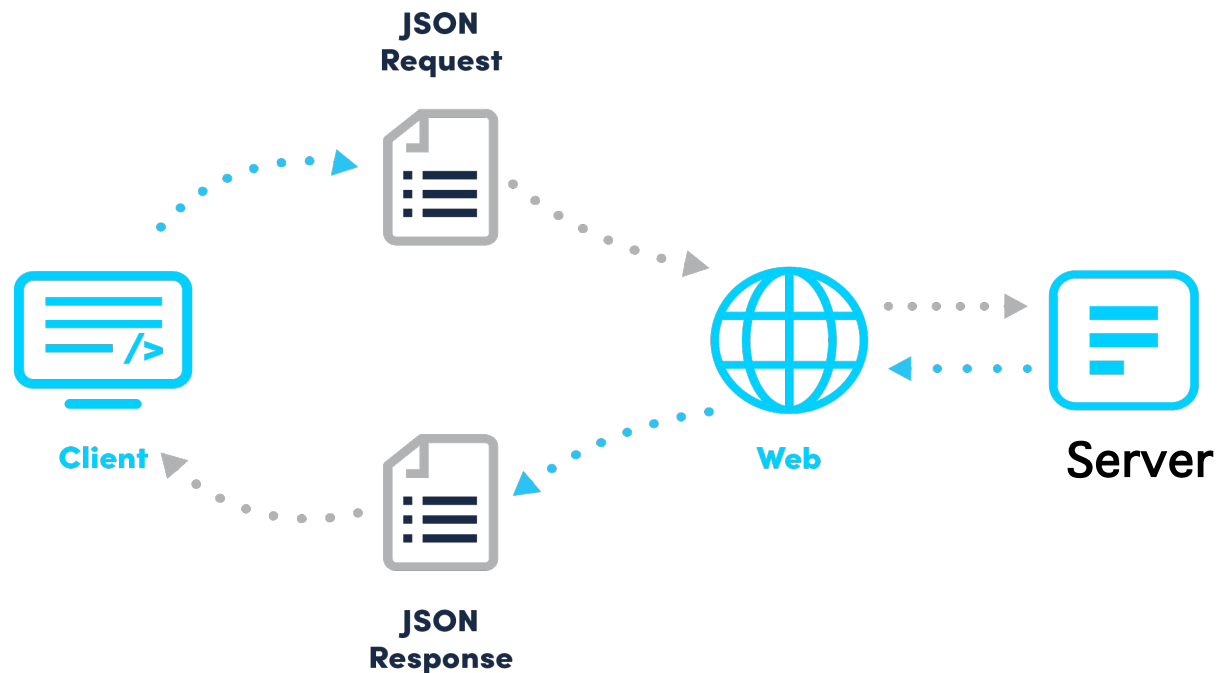
Journals Special Issues:

- Computer Networks, Elsevier, Special Issue on "Pub/sub solutions for interoperable and dynamic IoT systems". Submission deadline: to be opened soon.
- Frontiers, Loop, Research Topic on "SDN migration challenges and practices in ISP/Telcos" Networks. Submission deadline: open.
- ITU Journal, ITU, Special issue on "Network virtualization, slicing, orchestration, fog and edge

A brief introduction to MQTT

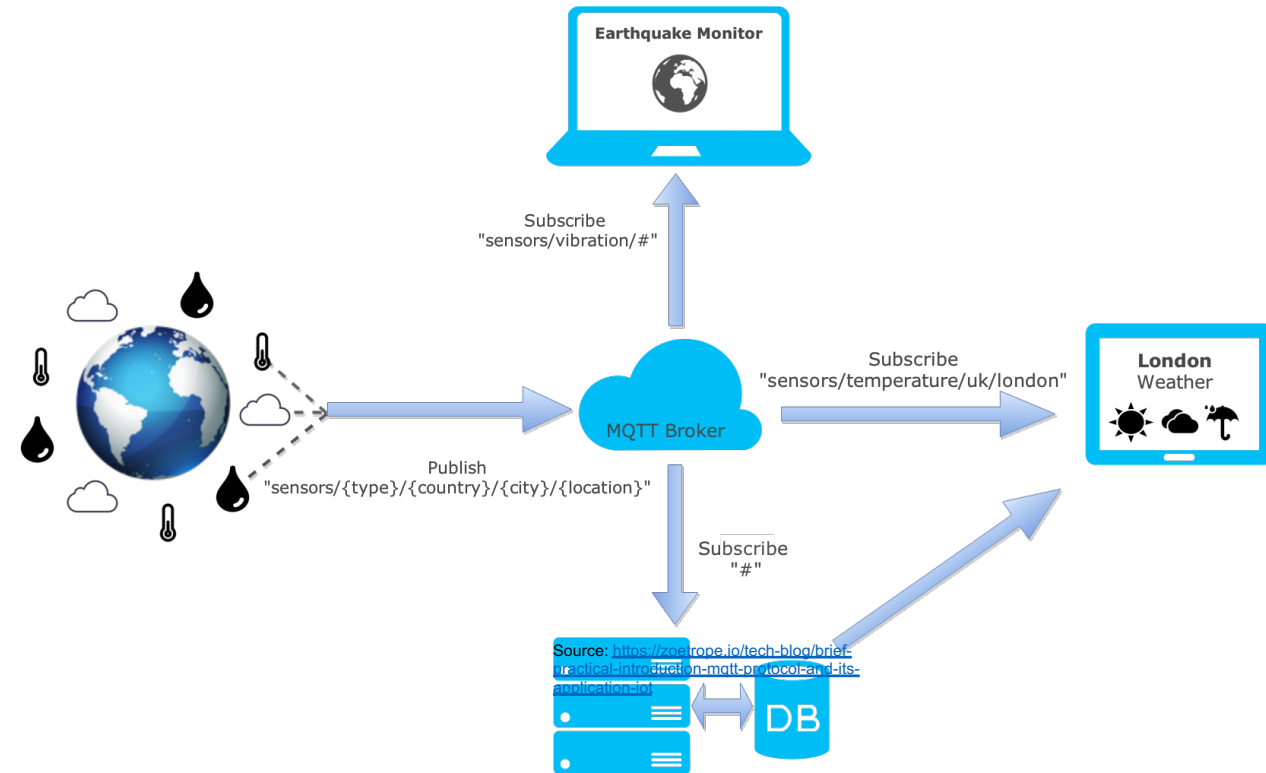
Request/response approach

- **REST**: Representational State Transfer
- Widely used; based on HTTP
- *Lighter version: **CoAP** (Constrained Application Protocol)*



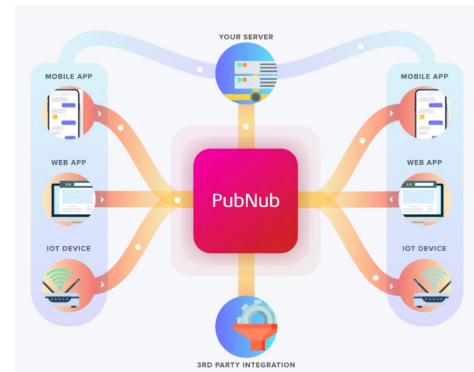
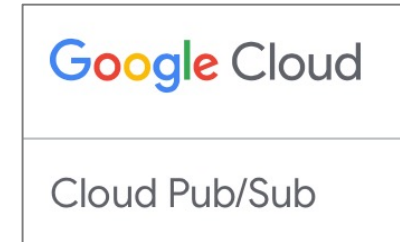
Pub/sub approach: an example

- Pub/Sub separate a client (or more clients), who is sending a message about a specific **topic**, called **publisher**, from another client (or more clients), who is receiving the message, called **subscriber**.
- There is a third component, called **broker**, which is known by both the publisher and subscriber, which filters all incoming messages and distributes them accordingly.

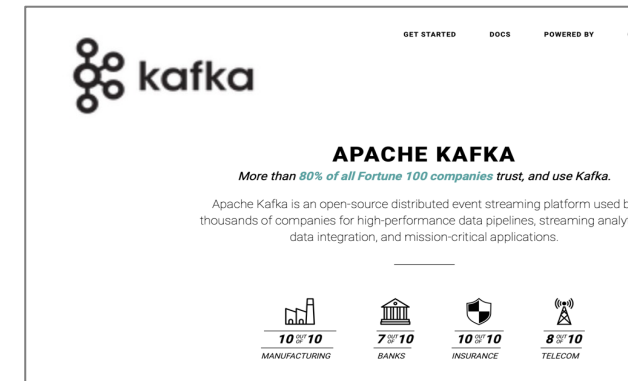


Pub/sub approach: a growing approach

- Various protocols:
 - AMQP, XMPP (was Jabber), ... **MQTT**



<https://www.pubnub.com/>



<https://kafka.apache.org>

Message Queuing Telemetry Transport



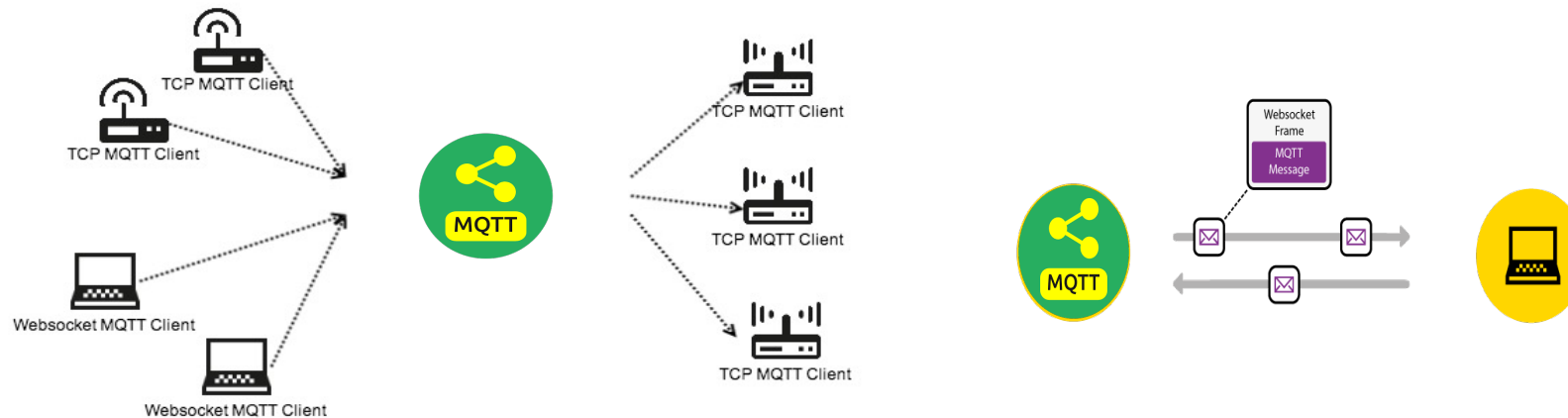
- A **lightweight publish-subscribe protocol** that can run on embedded devices and mobile platforms → <http://mqtt.org/>
 - Low power usage.
 - Binary compressed headers
 - Maximum message size of 256MB
 - not really designed for sending large amounts of data
 - better at a high volume of low size messages.
- Documentation sources:
 - The MQTT community wiki:
 - <https://github.com/mqtt/mqtt.github.io/wiki>
 - A very good tutorial:
 - <http://www.hivemq.com/mqtt-essentials/>

Some details about versions

- **MQTT 3.1.1 is the current version of the protocol.**
 - Standard document here:
 - <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>
 - October 29th 2014: MQTT was officially approved as OASIS Standard.
 - https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt
- MQTT v5.0 is the successor of MQTT 3.1.1
 - Current status: Committee Specification 02 (7 March 2019)
 - <http://docs.oasis-open.org/mqtt/mqtt/v5.0/cs02/mqtt-v5.0-cs02.html>
 - **Not backward compatible**; too many new things are introduced so existing implementations have to be revisited, for example:
 1. More extensibility → user properties
 2. Improved error reporting (Reason Code & Reason String)
 3. Performance improvements and improved support for small clients
 - shared subscriptions
 - topic alias
 4. Formalized common patterns → payload format description
 5. Improved authentication

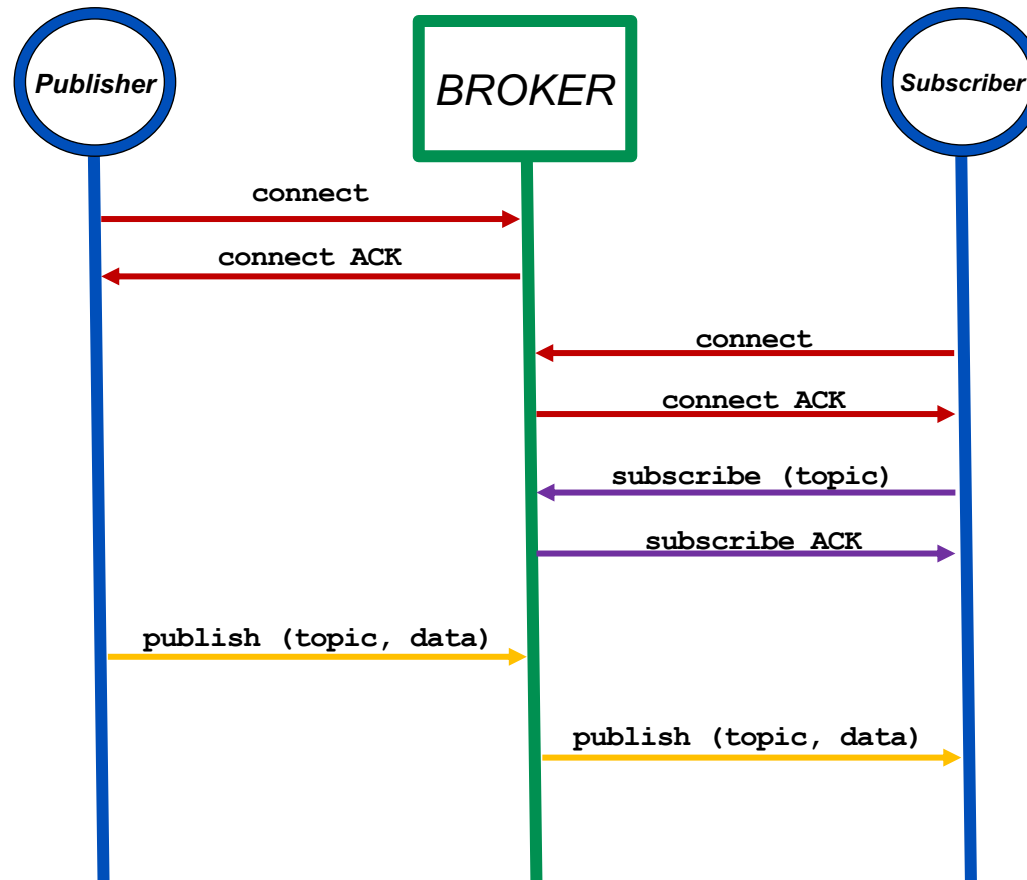
MQTT works on top of...

- mainly of TCP
 - There is also the closely related [MQTT for Sensor Networks \(MQTT-SN\)](#) where TCP is replaced by UDP → TCP stack is too complex for WSN
- websockets can be used, too!
 - Websockets allows you to receive MQTT data directly into a web browser.



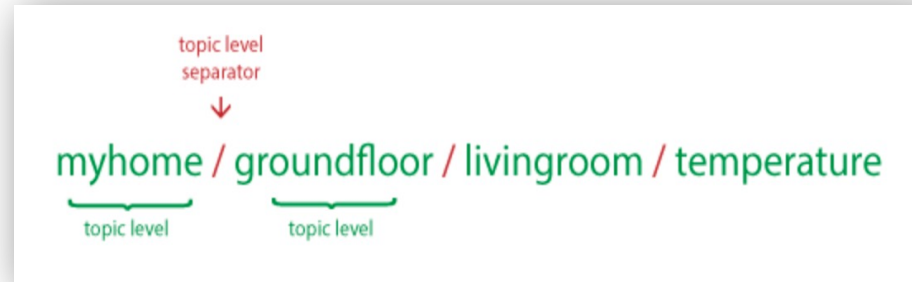
- Both, TCP & websockets can work on top of “Transport Layer Security (TLS)” (and its predecessor, Secure Sockets Layer (SSL))

Publish/subscribe interactions sequence



Topics

- MQTT Topics are structured in a hierarchy similar to folders and files in a file system using the forward slash (/) as a delimiter.
- Allow to create a user friendly and self descriptive **naming structures**



- Topic names are:
 - Case sensitive
 - use UTF-8 strings.
 - Must consist of at least one character to be valid.
- Except for the \$SYS topic **there is no default or standard topic structure.**

Special \$SYS/ topics

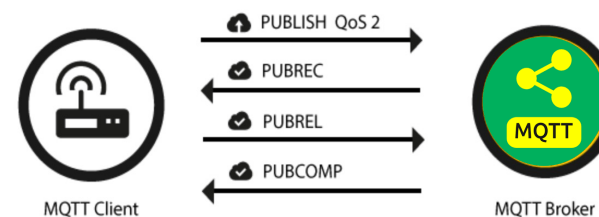
- \$SYS/broker/clients/connected
- \$SYS/broker/clients/disconnected
- \$SYS/broker/clients/total
- \$SYS/broker/messages/sent
- \$SYS/broker/uptime

Topics wildcards

- Topic subscriptions can have wildcards. These enable nodes to subscribe to groups of topics that don't exist yet, allowing greater flexibility in the network's messaging structure.
 - '+' matches anything at a given tree level
 - '#' matches a whole sub-tree
- Examples:
 - Subscribing to topic `house/#` covers:
 - `house/room1/main-light/left/red`
 - `house/room1/alarm`
 - `house/garage/main-light`
 - `house/main-door/lock/upper`
 - Subscribing to topic `house/+/main-light` covers:
 - `house/room1/main-light`
 - `house/room2/main-light`
 - `house/garage/main-light`
 - but doesn't cover
 - `house/room1/side-light`
 - `house/room2/front-light`

Quality of Service (QoS)

- Messages are published with a **Quality of Service (QoS)** level, which specifies delivery requirements.
- A **QoS 0 (“at most once”)** message is fire-and-forget.
 - For example, a notification from a doorbell may only matter when immediately delivered.
- With **QoS 1 (“at least once”)**, the broker stores messages on disk and retries until clients have acknowledged their delivery.
 - (Possibly with duplicates.) It’s usually worth ensuring error messages are delivered, even with a delay.
- **QoS 2 (“exactly once”)** messages have a second acknowledgement round-trip, to ensure that **non-idempotent messages** can be delivered exactly once.



Retained Messages


- A retained message is a normal MQTT message with the **retained flag set to true**. The broker will store the last retained message and the corresponding QoS for that topic
 - Each client that subscribes to a topic pattern, which matches the topic of the retained message, will receive the message immediately after subscribing.
 - For each topic **only one retained message** will be stored by the broker.
- **Retained messages can help newly subscribed clients to get a status update immediately after subscribing to a topic and don't have to wait until a publishing clients send the next update.**
 - In other words, a retained message on a topic is the last known good value, because it doesn't have to be the last value, but it certainly is the last message with the retained flag set to true.

MQTT Keep alive

- The keep alive functionality assures that the connection is still open and both broker and client are connected to one another.
- The client specifies a time interval in seconds and communicates it to the broker during the establishment of the connection.
 - The interval is the longest possible period of time which broker and client can endure without sending a message.
 - If the broker doesn't receive a PINGREQ or any other packet from a particular client, it will close the connection and send out the last will and testament message (if the client had specified one).
- Good to Know
 - The MQTT client is responsible of setting the right keep alive value.
 - The maximum keep alive is 18h 12min 15 sec.
 - If the keep alive interval is set to **0**, the keep alive mechanism is deactivated.

“Will” message

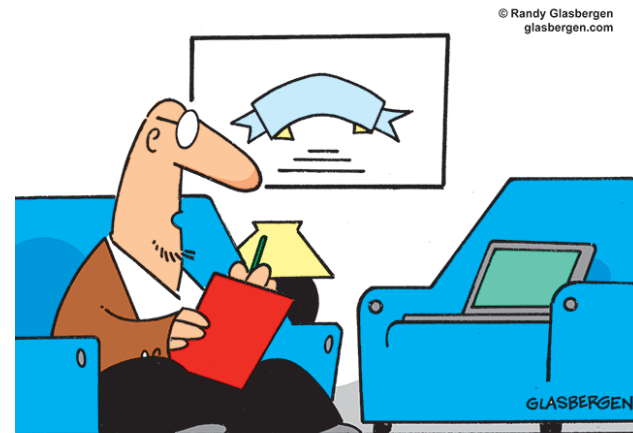
- When clients connect, they can specify an optional “will” message, to be delivered if they are unexpectedly disconnected from the network.
 - (In the absence of other activity, a 2-byte ping message is sent to clients at a configurable interval.)
- This “last will and testament” can be used to notify other parts of the system that a node has gone down.

MQTT-Packet:	
CONNECT 	
contains:	Example
clientId	"client-1"
cleanSession	true
username (optional)	"hans"
password (optional)	"letmein"
lastWillTopic (optional)	"/hans/will"
lastWillQos (optional)	2
lastWillMessage (optional)	"unexpected exit"
lastWillRetain (optional)	false
keepAlive	60

WHEN?

A few words on security

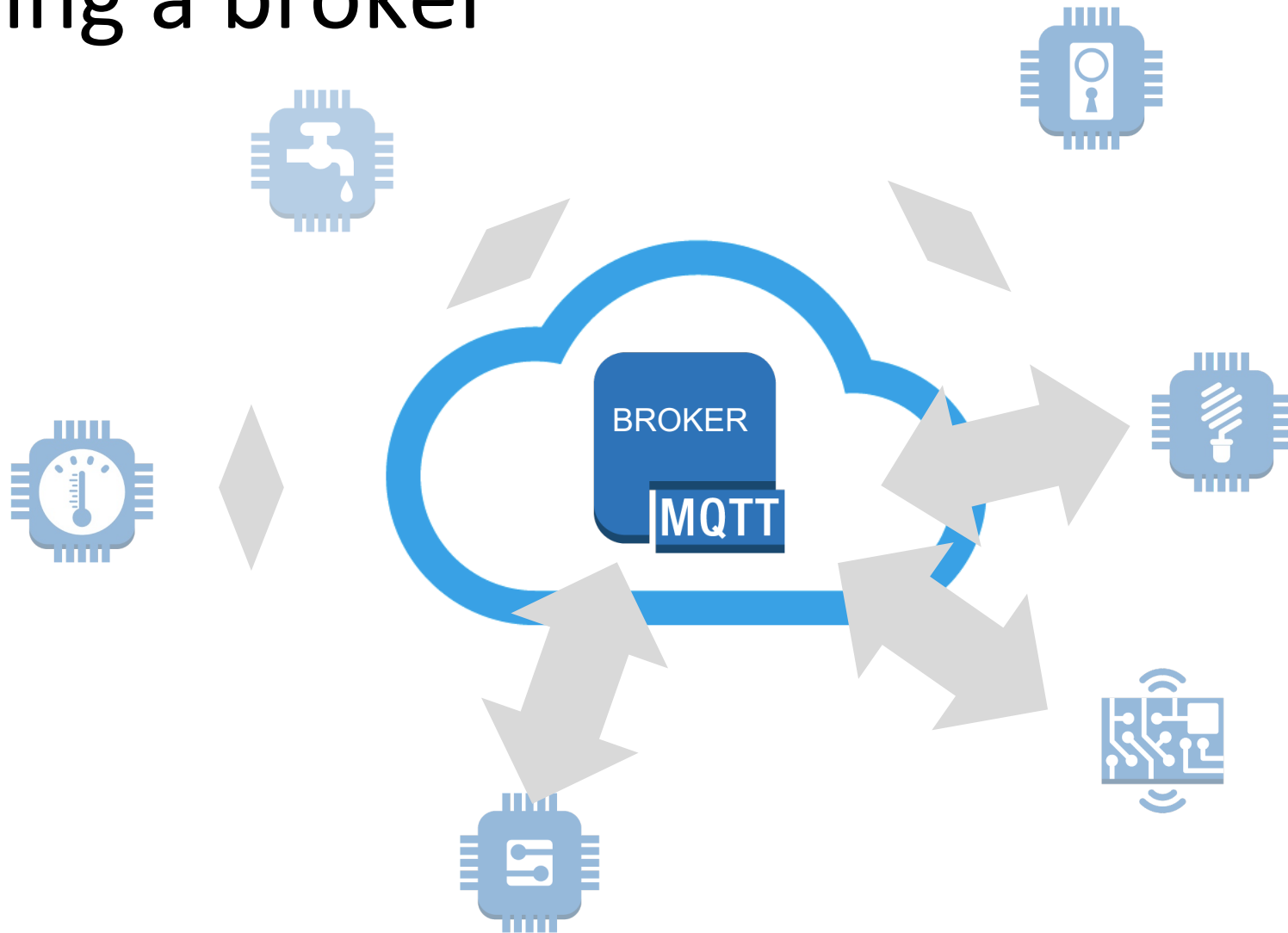
- MQTT has the option for Transport Layer Security (TLS) encryption.
- MQTT also provides username/password authentication with the broker.
 - Note that the password is transmitted in clear text. Thus, be sure to use TLS encryption if you are using authentication.



"It's not just you. We're all insecure in one way or another."

MQTT in practice

Creating a broker



Available MQTT brokers

- The most widely used are:
 - <http://mosquitto.org/>
 - man page: <https://mosquitto.org/man/mosquitto-8.html>
 - <http://www.hivemq.com/>
 - The standard trial version only supports 25 connections.
- And also:
 - <https://www.rabbitmq.com/mqtt.html>
 - <http://activemq.apache.org/mqtt.html>
- A quite complete list can be found here:
 - <https://github.com/mqtt/mqtt.github.io/wiki/servers>

Cloud based MQTT brokers: CloudMQTT

<https://www.cloudmqtt.com/> → based on Mosquitto

The screenshot shows the CloudMQTT website with a navigation bar containing 'Pricing', 'Documentation', 'Support', and 'Blog'. The main heading reads 'Hosted message broker for the Internet of Things'. Below this, there are two pricing cards. The 'Power Pug' plan, featuring a pug icon, is priced at \$299 per month and includes up to 10,000 connections, no artificial limitations, and support by e-mail and phone. The 'Humble Hedgehog' plan, featuring a hedgehog icon, is priced at \$5 per month and includes 25 users/acl rules/connections, 20 Kbit/s, 3 bridges, and support by e-mail. A large orange arrow points from the Power Pug plan to the Humble Hedgehog plan. The background features a large illustration of IoT-related icons like a cloud, satellite, and computer monitor.

CloudMQTT Pricing Documentation Support Blog

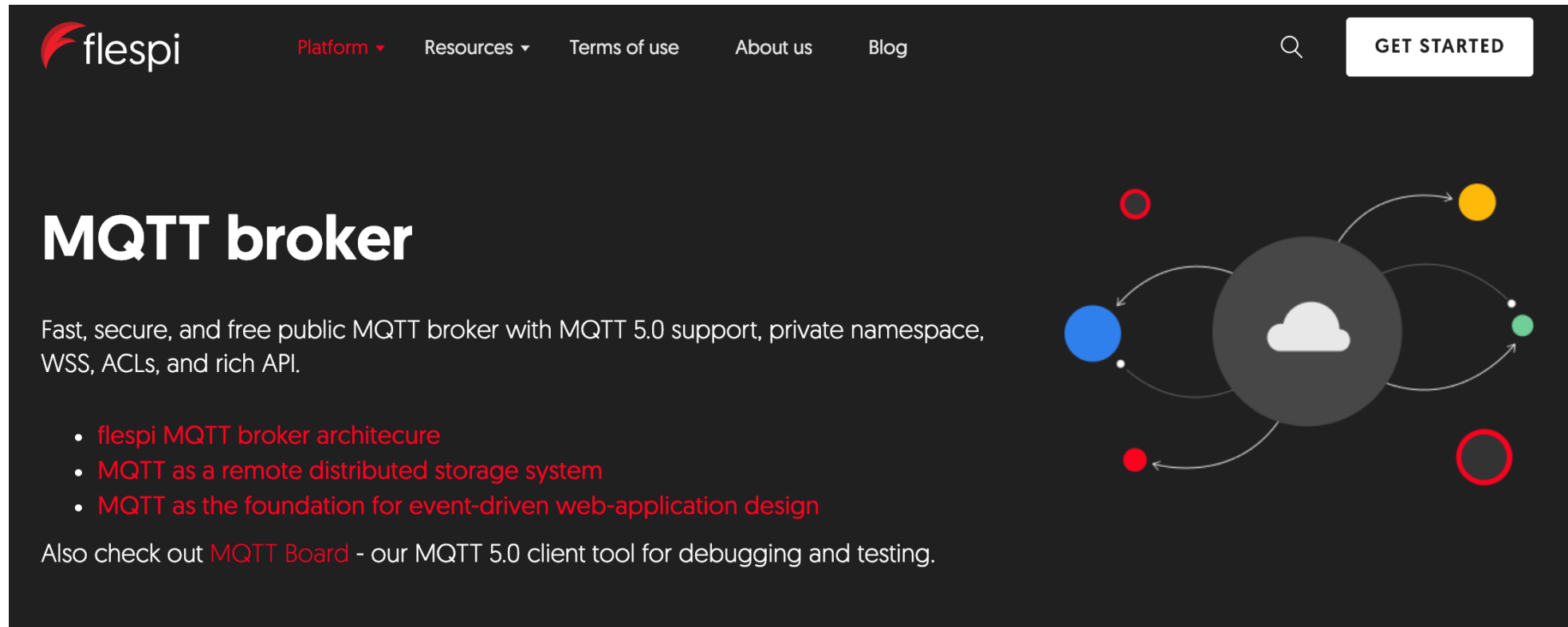
Hosted message broker for the Internet of Things

Perfectly configured and optimized message queues for IoT, ready in seconds.

Plan Name	Price (Per Month)	Key Features
Power Pug	\$299	Up to 10 000 connections, No artificial limitations, Support by e-mail, Support by phone
Humble Hedgehog	\$5	25 users/acl rules/connections, 20 Kbit/s, 3 bridges, Support by e-mail

Cloud based brokers: flespi

<https://flespi.com/mqtt-broker>



The screenshot shows the landing page for the flespi MQTT broker. The header features the flespi logo, navigation links for Platform, Resources, Terms of use, About us, and Blog, a search icon, and a GET STARTED button. The main heading is 'MQTT broker', followed by a description: 'Fast, secure, and free public MQTT broker with MQTT 5.0 support, private namespace, WSS, ACLs, and rich API.' Below this is a bulleted list of features: 'flespi MQTT broker architecture', 'MQTT as a remote distributed storage system', and 'MQTT as the foundation for event-driven web-application design'. At the bottom, it mentions 'MQTT Board' as a client tool. A diagram on the right illustrates a central cloud icon connected to several colored nodes (blue, red, yellow, green, red) via arrows, representing a distributed network.

flespi Platform Resources Terms of use About us Blog GET STARTED

MQTT broker

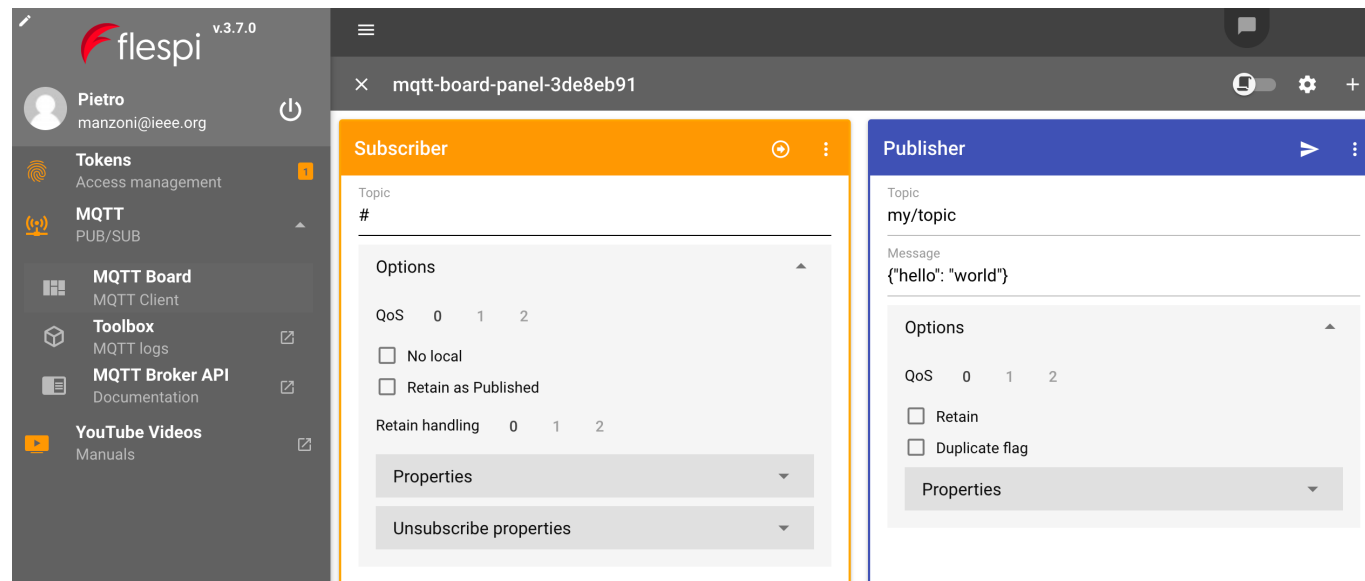
Fast, secure, and free public MQTT broker with MQTT 5.0 support, private namespace, WSS, ACLs, and rich API.

- flespi MQTT broker architecture
- MQTT as a remote distributed storage system
- MQTT as the foundation for event-driven web-application design

Also check out [MQTT Board](#) - our MQTT 5.0 client tool for debugging and testing.

Cloud based brokers: flespi

<https://flespi.io/#/panel/mqttboard>



flespi MQTT broker connection details

- **Host** — `mqtt.flespi.io`.
- **Port** — **8883 (SSL)** or **1883 (non-SSL)**; for MQTT over WebSockets: **443 (SSL)** or **80 (non-SSL)**.
- **Authorization** — use a **flespi platform token** as MQTT session username; no password.
- **Client ID** — use any unique identifier within your flespi user session.
- **Topic** — you can publish messages to any topic except **flespi/**.
- **ACL** — both **flespi/** and **MQTT pub/sub** restrictions **determined by the token**.

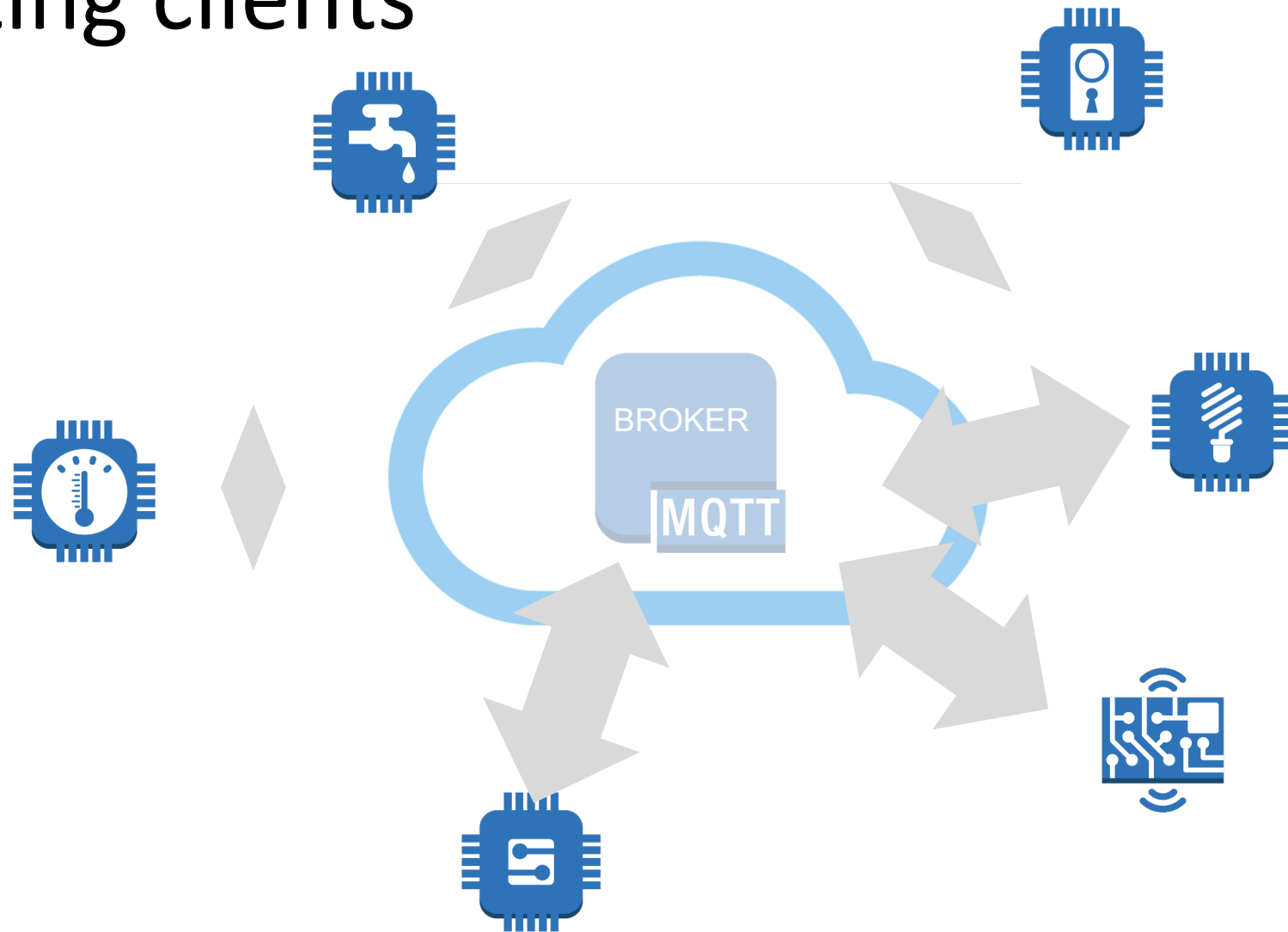
Open brokers (“Sandboxes”)

- **TCP based:**
 - <https://iot.eclipse.org/getting-started/#sandboxes>
 - Hostname: **iot.eclipse.org**
 - <http://test.mosquitto.org/>
 - Hostname: **test.mosquitto.org**
 - <https://www.hivemq.com/mqtt-demo/>
 - Hostname: **broker.hivemq.com**
 - <http://www.mqtt-dashboard.com/>
 - Ports:
 - standard: 1883
 - encrypted: 8883 (*TLS v1.2, v1.1 or v1.0 with x509 certificates*)
- **Websockets based:**
 - broker.mqttdashboard.com port: 8000
 - test.mosquitto.org port: 8080
 - broker.hivemq.com port: 8000
- https://github.com/mqtt/mqtt.github.io/wiki/public_brokers

Installing Mosquitto

- It takes only a few seconds to install a Mosquitto broker on a Linux machine. You need to execute the following steps:
 - *sudo apt-get update sudo apt-get install mosquitto mosquitto-clients*
- To start and stop the broker execution use:
 - *sudo /etc/init.d/mosquitto start/stop*
- Verbose mode:
 - *sudo mosquitto -v*
- To check if the broker is running you can use the command:
 - *sudo netstat -tanlp | grep 1883*

Creating clients



MQTT clients: iOS



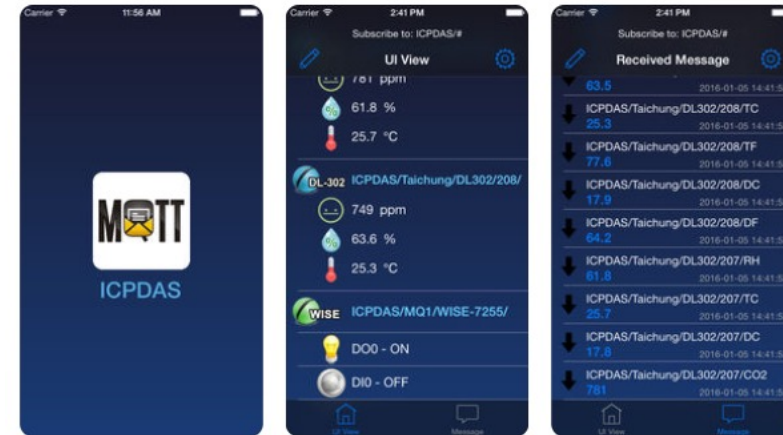
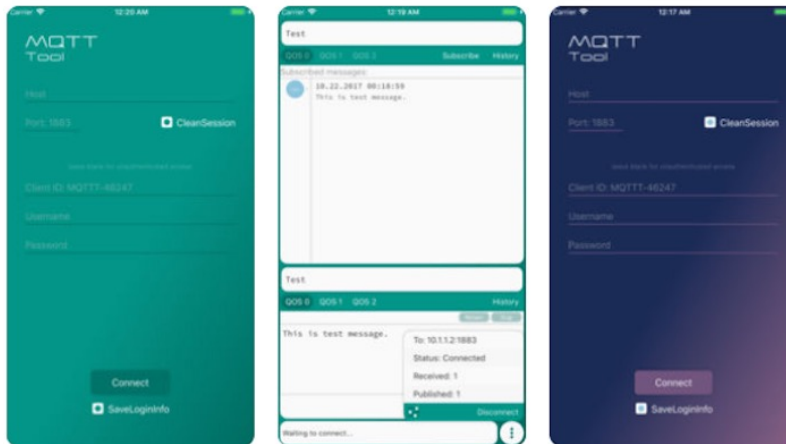
Mqttt
Utilidades

ABRIR

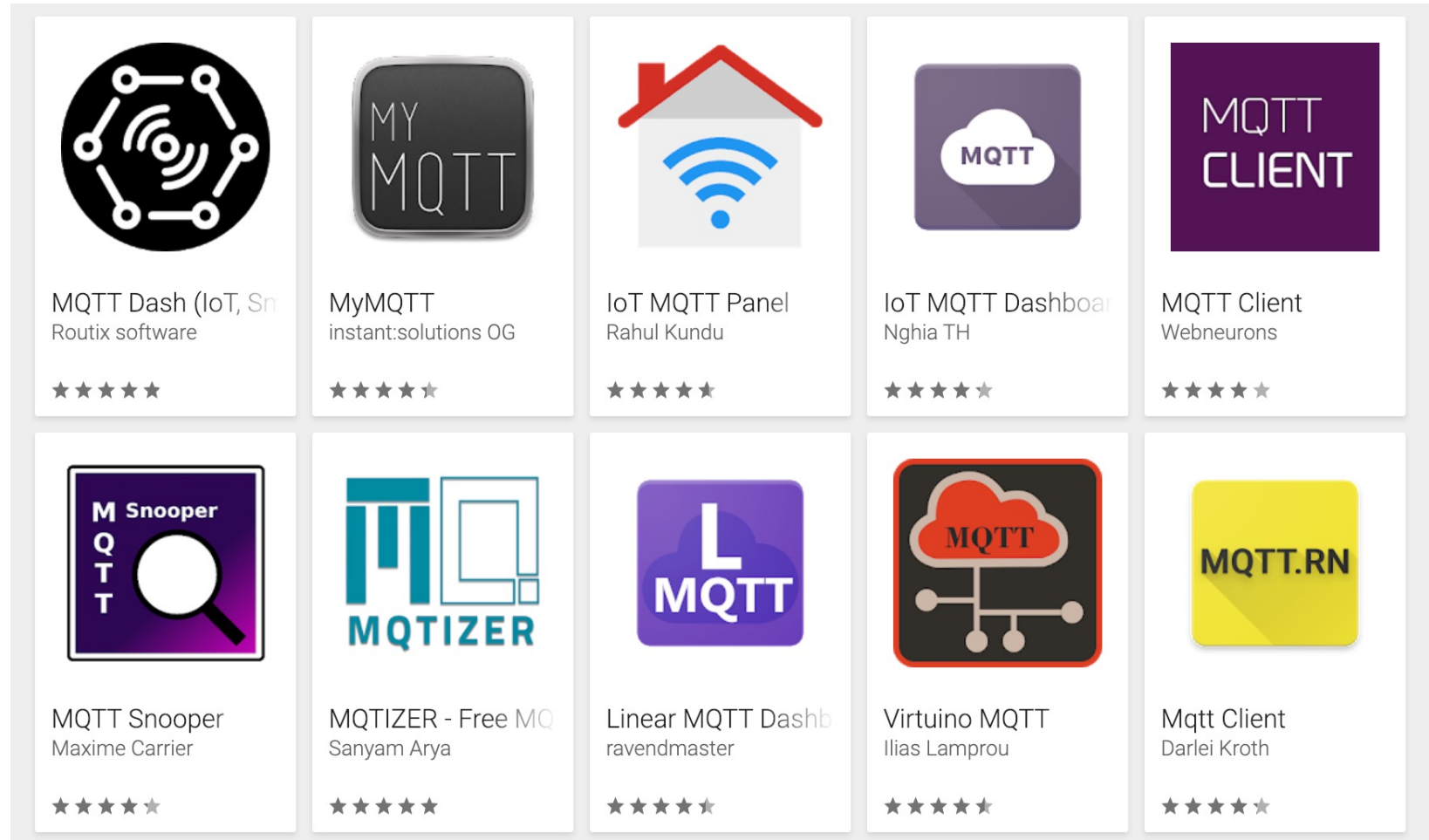


ICPDAS MQ...
Utilidades

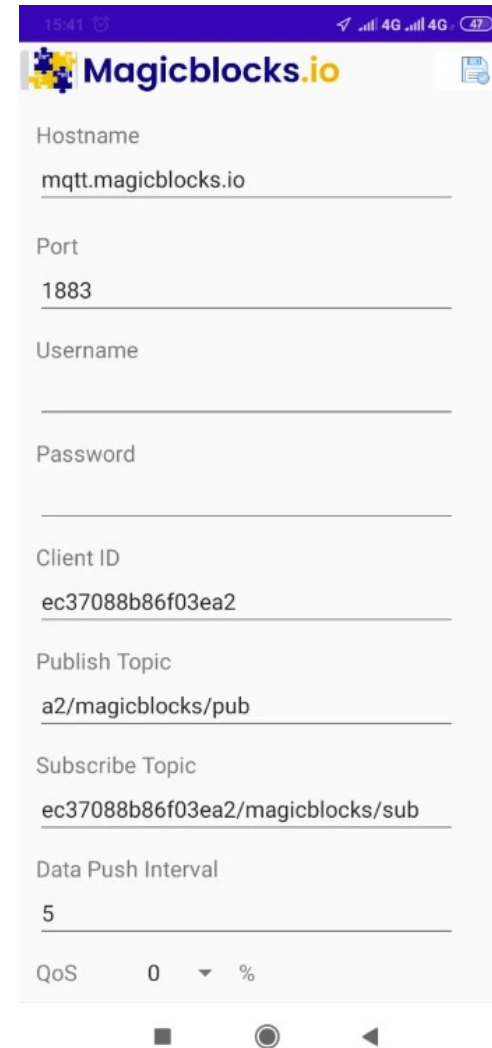
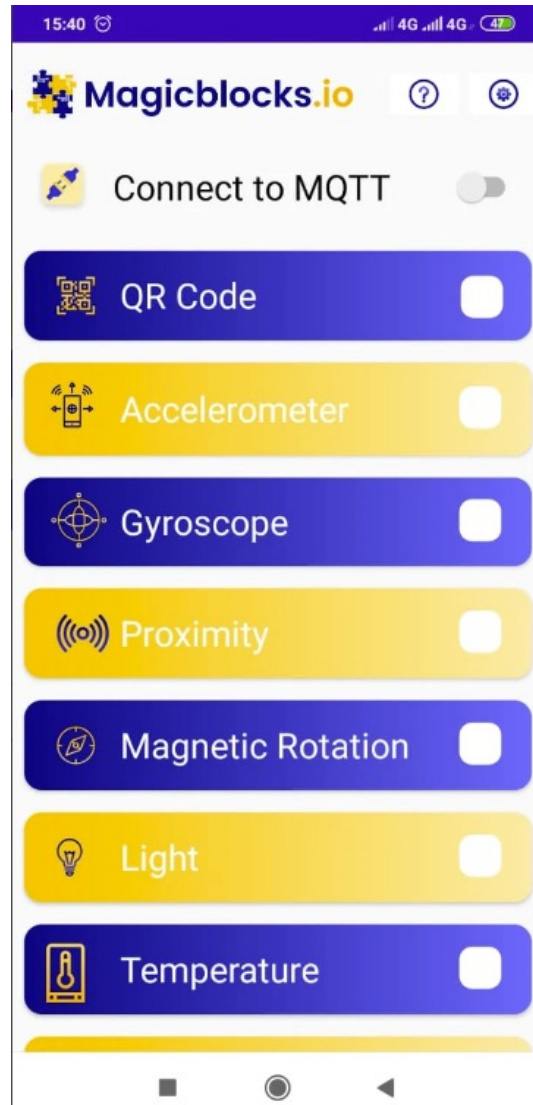
ABRIR



MQTT clients: Android



MQTT clients: Android



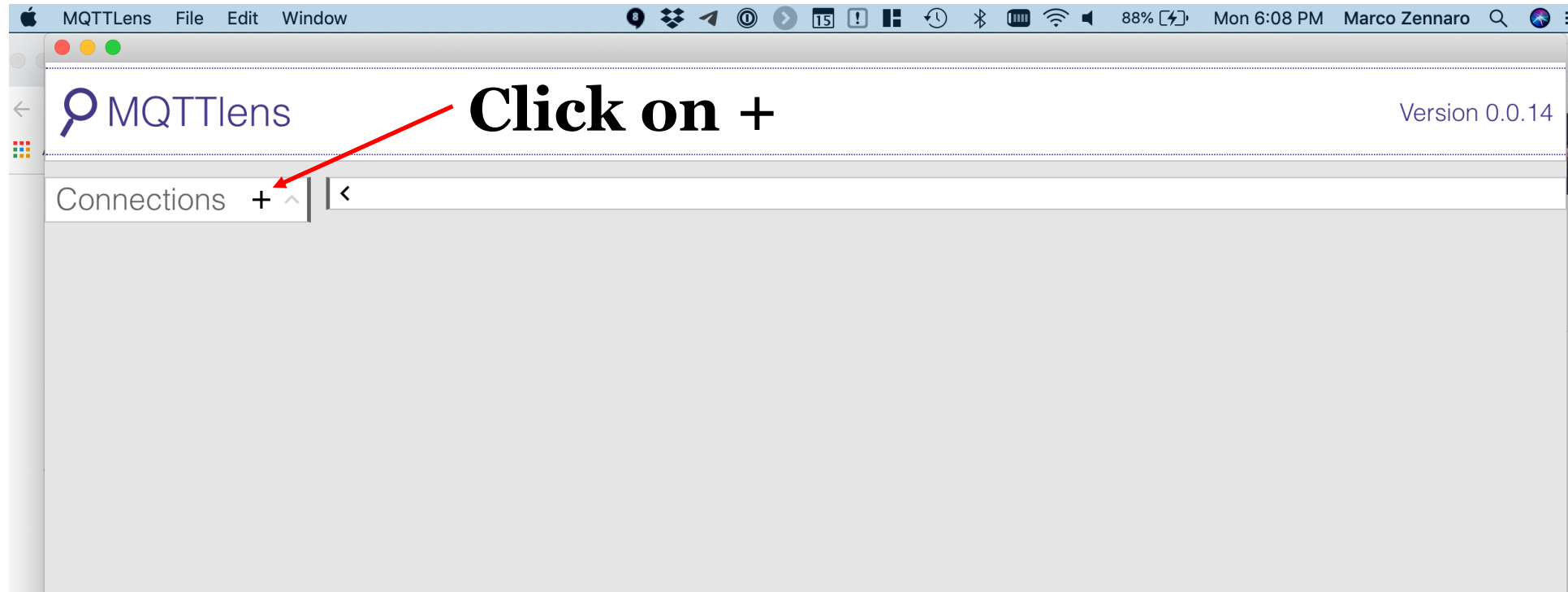
Mosquitto client

- The Mosquitto broker comes with a couple of useful commands to quickly publish and subscribe to some topic.
- Their basic syntax is the following.
 - *mosquitto_sub -h HOSTNAME -t TOPIC*
 - *mosquitto_pub -h HOSTNAME -t TOPIC -m MSG*
- More information can be found:
https://mosquitto.org/man/mosquitto_sub-1.html
https://mosquitto.org/man/mosquitto_pub-1.html

MQTT lens

- **MQTT Lens: a Google Chrome application**, which connects to a MQTT broker and is able to subscribe and publish to MQTT topics.
- Search for MQTT Lens in Google and install it in Chrome.

MQTT lens



MQTT lens

Name **Broker address** **Broker port #**

Connection name
Eclipse MQTT

Connection color scheme
[Green bar]

Hostname
tcp:// e.g. iot.eclipse.org

Port
1883

Client ID
lens_xUNxxwliiALGArFmSnbwCOpmt7Xp
Generate a random ID

Session **Automatic Connection** **Keep Alive**
 Clean Session Automatic Connection 120 seconds

Credentials

Username **Password**
Enter username Enter password

Last-Will [dropdown arrow]

CREATE CONNECTION

Create connection

MQTT lens: subscriber

<

Connection: Test

Topic name

Subscribe

topic 0 - at most once SUBSCRIBE

Publish

topic 0 - at most once Retained PUBLISH

Message

Subscriptions

MQTT lens: publisher

The screenshot shows the MQTT Lens publisher interface. It is divided into three main sections: 'Subscribe', 'Publish', and 'Subscriptions'. The 'Subscribe' section has a text input field containing 'topic', a dropdown menu set to '0 - at most once', and a green 'SUBSCRIBE' button. The 'Publish' section has a text input field containing 'topic', a dropdown menu set to '0 - at most once', a checkbox for 'Retained', and a green 'PUBLISH' button. The 'Subscriptions' section is currently empty. Two red arrows point from the text 'Topic name' to the 'topic' input field in the 'Publish' section, and from the text 'Message' to the 'Message' input field in the 'Publish' section.

<

Connection: Test

Subscribe ^

topic 0 - at most once SUBSCRIBE

Publish ^

topic **Topic name** 0 - at most once Retained **Message** PUBLISH

Message

Subscriptions

MQTT Lab

- **Exercise #1**
 - Split the class in two: half of the class will publish data using their smartphones and the other half will subscribe using smartphones or MQTT software (MQTT Lens, MQTT Explorer or others)
 - You must first agree on:
 - The broker you will use
 - The topic you will use to publish/subscribe

MQTT Lab

- **Exercise #2**
 - Download “Magicblocks” from the Google Play Store
 - Experiment with the app and send data from your phone’s sensors to an MQTT broker.
 - Can you see the data on an MQTT subscriber?

Telegram/Whatsapp Door alarm



Using the gyroscope or the accelerometer sensor, the Door alarm will send an alert message using Telegram/Whatsapp when the door opens. Use an MQTT-Telegram/Whatsapp bridge.

Thanks

