Message

Queuing

Telemetry

Transport

Long time ago...

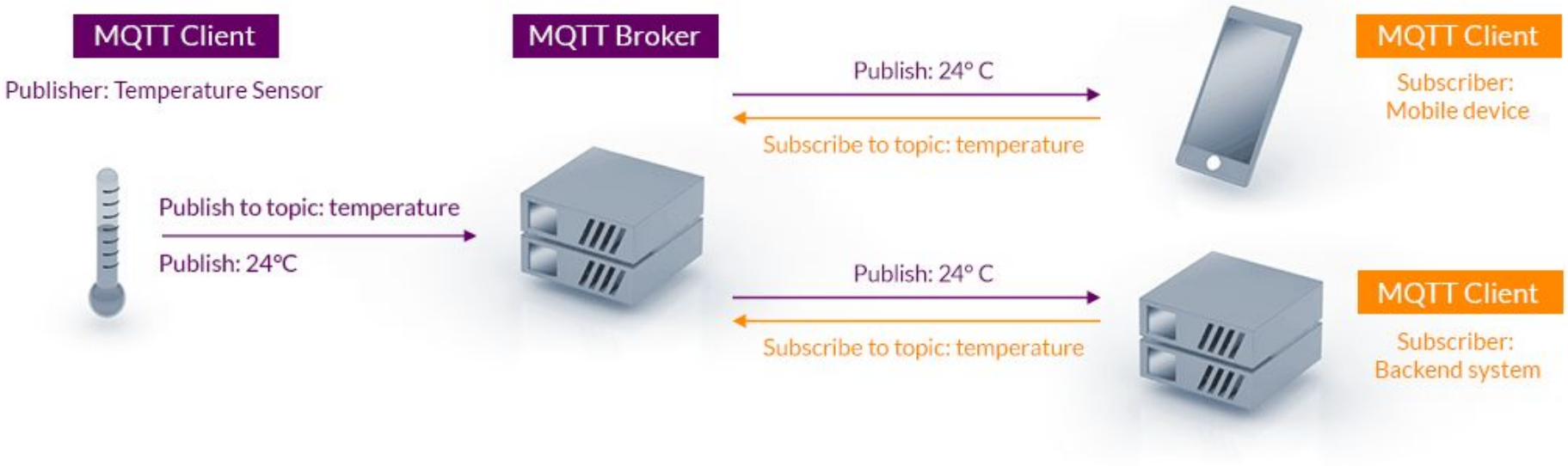
Engineers needed a protocol for use in the oil and gas industry.

Requirements were simple - minimal bandwidth and minimal battery loss

to monitor oil pipelines via satellite.

MQTT (Message Queuing Telemetry Transport) was invented.

How it works


MQTT Client
Publisher: Temperature Sensor
MQTT Broker
Publish: 24° C
Subscribe to topic: temperature
MQTT Client
Subscriber:
Mobile device
Publish to topic: temperature
Publish: 24°C
Publish: 24° C
Subscribe to topic: temperature
MQTT Client
Subscriber:
Backend system

How it works

- Protocol that works on top of the TCP/IP protocol.
- Sending HTTP requests with your IoT device is MQTT messages can be as small as 2 bytes, whereas HTTP requires headers which contains a lot of information that other devices might not care about.
- With MQTT, when a server receives information from one client, it will automatically distribute that information to each of the interested clients. If you have multiple devices waiting for a request with HTTP, you'll need to send eg. a POST action to each client.

Basic information

- MQTT specification — mqtt.org/mqtt-specification
- MQTT software — mqtt.org/software/
- MQTT use cases — mqtt.org/use-cases/

ISO/IEC PRF 20922

Basic concepts

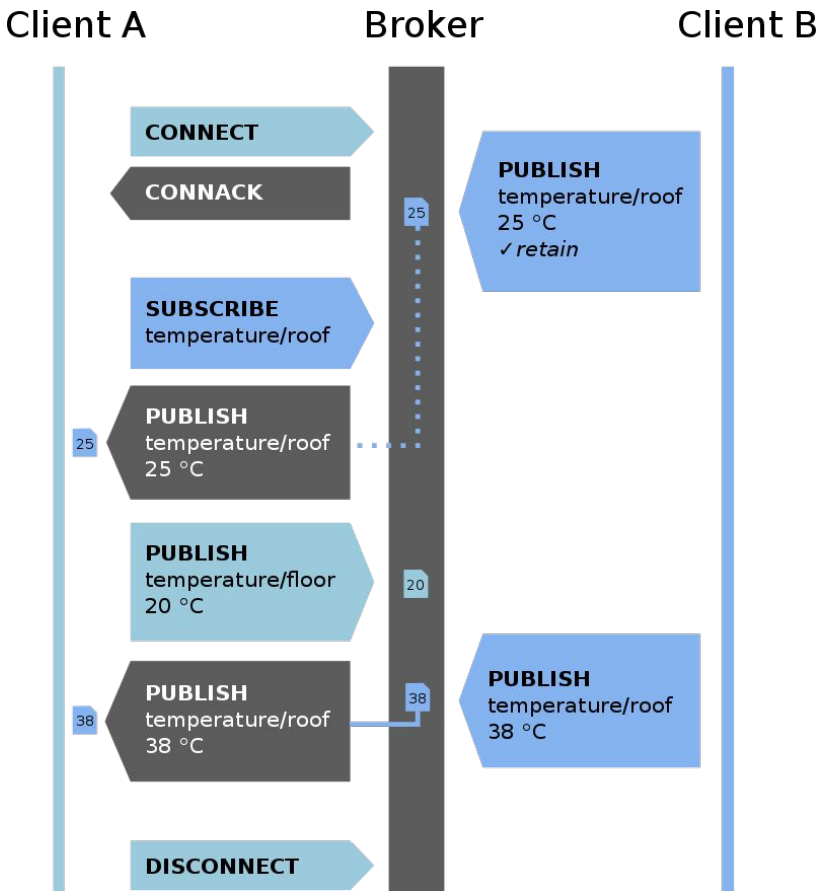
- Publish/Subscribe
- Message
- Topic
- Broker

Publish/Subscribe

In a publish and subscribe system, a device can publish a message on a topic, or it can be subscribed to a particular topic to receive messages.

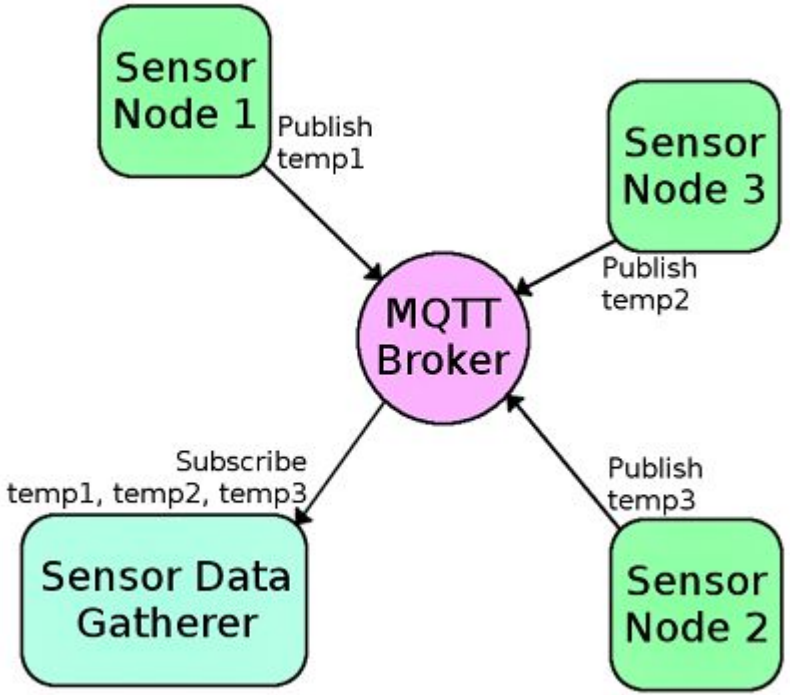
- **Device 1** publish on a topic.
- **Device 2** is subscribed to the same topic as **device 1** is publishing in.
- **Device 2** receives the messages.

Publish/Subscribe



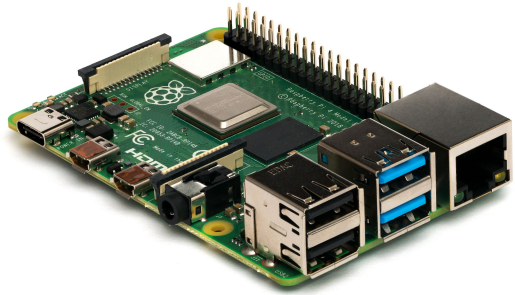
Broker

The broker is primarily responsible for receiving all messages, filtering the messages, decide who is interested in them and then publishing the message to **all subscribed clients**.



Broker

+



mosquitto

HIVEMQ CLOUD

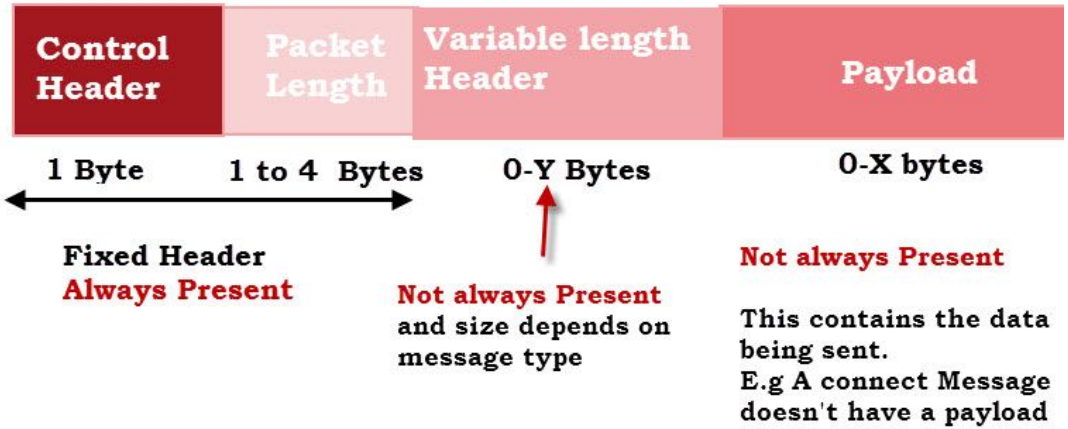
verne MQ

MOSCA

EMQ

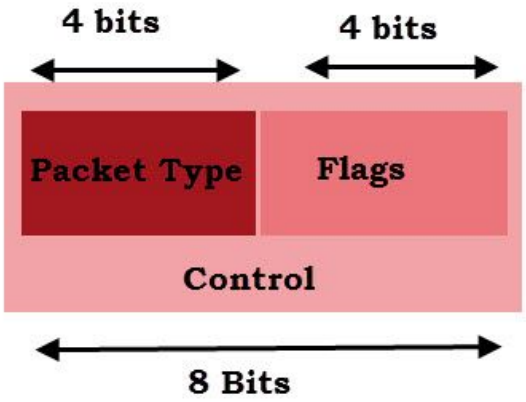
Message

Messages are the information that you want to exchange between your devices. Whether it’s a command or data.



MQTT Standard Packet Structure

Message



Packet Type Examples:

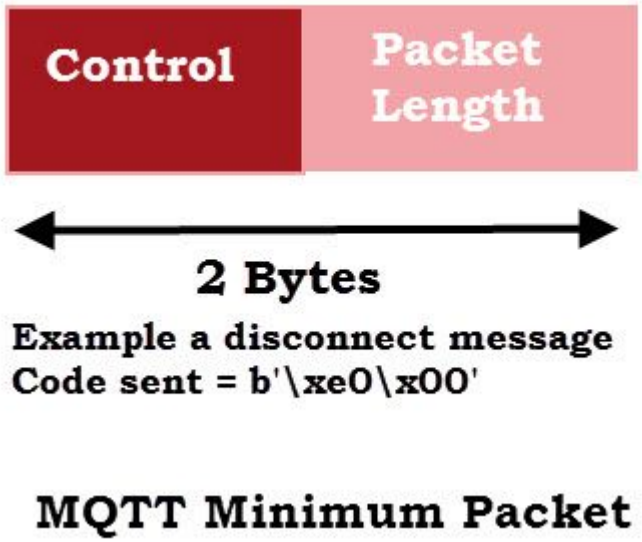
Connect =0001 =1
Connack=0010 =2

Disconnect=1110=14

MQTT Control Field Structure

www.steves-internet-guide.com/mqtt-protocol-messages-overview/

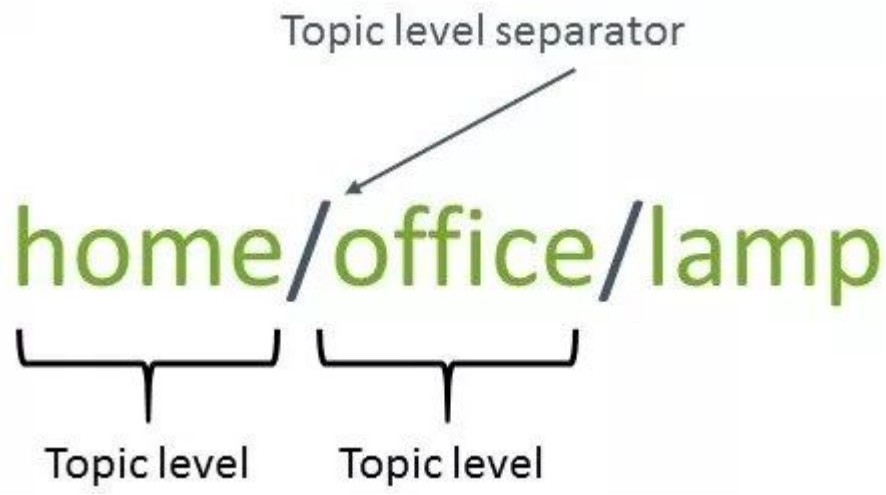
Message



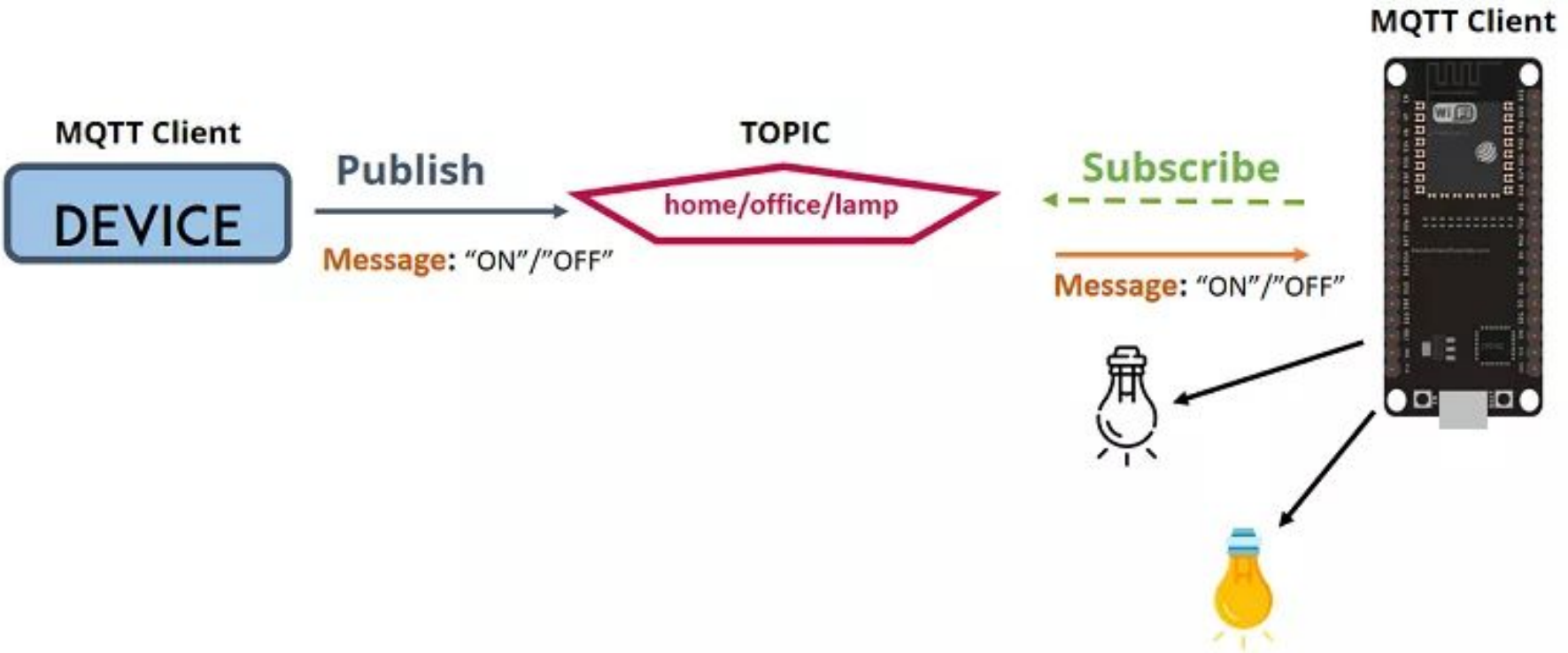
Max. packet length is 256MB.

Topic

Topics are the way you register interest for incoming messages or how you specify where you want to publish the message.



Topic


MQTT Client
DEVICE
Publish
Message: "ON"/"OFF"
TOPIC
home/office/lamp
Subscribe
Message: "ON"/"OFF"
MQTT Client

Topic

- topics are represented with strings separated by a forward slash
- each forward slash indicates a topic level
- topics are case sensitive
- UTF-8 encoding
- must contain at least 1 character

home/office/lamp

≠

home/office/LAmp

Topic

Do not use $SYS topic for Your own purposes.

These topics are special meta topics that the broker can use to publish information about the broker itself and its MQTT client sessions.

$SYS/broker/clients/total

The total number of active and inactive clients currently connected and registered on the broker

Topic

Subscribe to **house/#** covers:

- house/room1/main-light
- house/room1/alarm
- house/garage/main-light
- house/main-door

Subscribe to **house/+/main-light** covers:

- house/room1/main-light
- house/room2/main-light
- house/garage/main-light

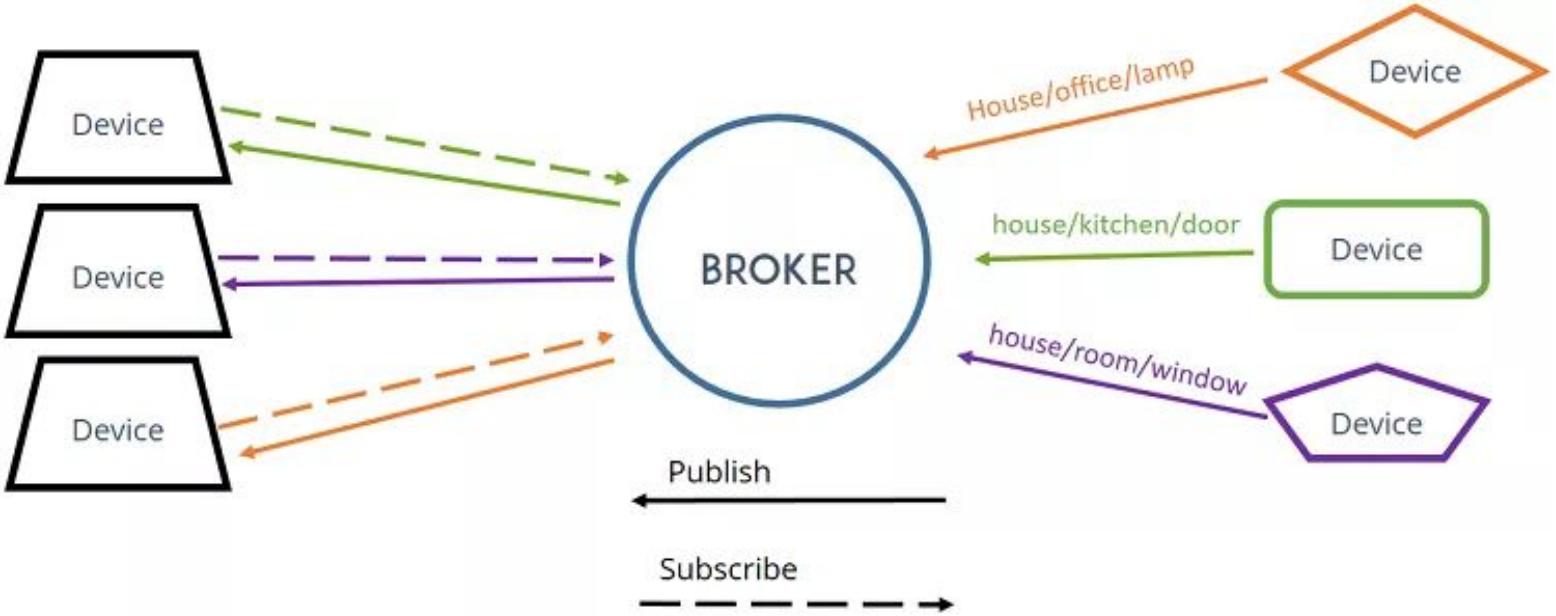
Topic

Incorrect topics:

- house+ – Reason - **no topic level**
- house# – Reason - **no topic level**

Client may publish to one topic. Publishing a message to many topics requires multiple publish operations.

Topic


Device
Device
Device
BROKER
House/office/lamp
Device
house/kitchen/door
Device
house/room/window
Device
Publish
Subscribe

Quality of Services (QoS)

An agreement between sender and receiver on the guarantee of delivering a message:

- 0 - the client publishes the message, and there is no acknowledgement by the broker (fire and forget).
- 1 - the broker sends an acknowledgement back to the sender, but in the event that that the acknowledgement is lost the sender won't realise the message has got through, so will send the message again. The client will re-send until it gets the broker's acknowledgement.

Quality of Services (QoS)

- 2 - known as assured delivery. Sender and receiver clients use a 2-level handshake to ensure only one copy of the message is received. There is a sequence of 4 messages between the sender and the receiver, to confirm that the main message has been sent and that the acknowledgement has been received.

Quality of Services (QoS)

Summary:

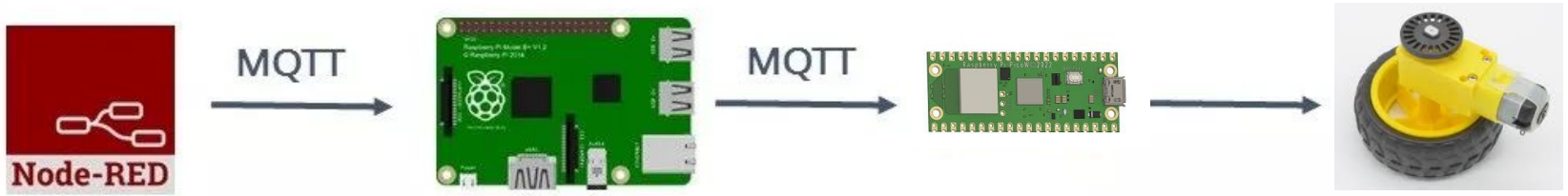
- QoS 0: at most once
- QoS 1: at least once
- QoS 2: exactly once

Understanding MQTT Quality of Service or also known as MQTT QoS

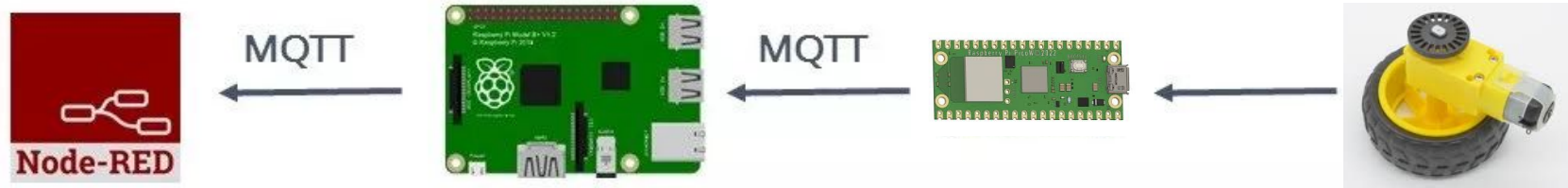
cedalo.com/blog/understanding-mqtt-qos/

Task

Send a command to control a DC motor - direction and rotational speed:



Read and publish data - angular velocity:



NodeRED on RPi

1.How to install NodeRED on Raspberry Pi:

nodered.org/docs/getting-started/raspberrypi

2. Aedes broker:

flows.nodered.org/node/node-red-contrib-aedes

3. Dashboard:

flows.nodered.org/node/node-red-dashboard

Mosquitto broker on RPi

1.How to install Mosquitto broker on Raspberry Pi:

```
sudo apt update
sudo apt install -y mosquitto mosquitto-clients
```

2. Mosquitto auto start on boot up:

```
sudo systemctl enable mosquitto.service
```

3. Testing installation (should prompt a warning message saying "*Error: Address already in use*"):

```
mosquitto -v
```

Mosquitto broker on RPi

4. RPi IP adres:

```
hostname -I
```

5. Subscribing to `testTopic` topic:

```
mosquitto_sub -d -t testTopic
```

6. Publishing `“It’s alive”` message to `testTopic` topic:

```
mosquitto_pub -d -t testTopic -m "It’s alive”
```

Project parameters