

R-Pi

Team Emertxe



IoT Protocols

MQTT



MQTT

Introduction

- Machine-to-Machine (M2M) and IoT connectivity protocol
- Lightweight messaging protocol which works with a server-based publish subscribe mechanism
- Runs on the top of TCP/IP protocol suite
- Lighter than HTTP 1.1 and HTTP/2 protocols
- Popular among IoT, M2M, Embedded Projects
- Suitable for the following applications for the data exchange
 - ✓ In-vehical Infotainment (IVI)
 - ✓ POS kiosks
 - ✓ Radio frequency identification

MQTT

Introduction

- MQTT was designed to support the challenges in M2M, IoT, Embedded and Mobile Applications
 - Be lightweight to make it possible to transmitt high volumes of data without huge overheads
 - Distribute minimal packets with huge volumes of data
 - Easily emit data from one client to many clients
 - Support an event-oriented paradigm with asynchronous, bidirectional, low-latency push delivery of messages
 - Make it possible to listen for the events whenever they happen

MQTT

Introduction

- MQTT was designed to support the challenges in M2M, IoT, Embedded and Mobile Applications ...
 - Publish information over unreliable networks
 - Provide reliable deliveries over fragile connections
 - Work well with battery-powered devices or require low power consumption
 - Provide responsiveness to make it possible to achieve near real time delivery of information
 - Offer security and privacy for all data
 - Be able to provide necessary scalability to distribute data to hundreds of thousands of clients

MQTT

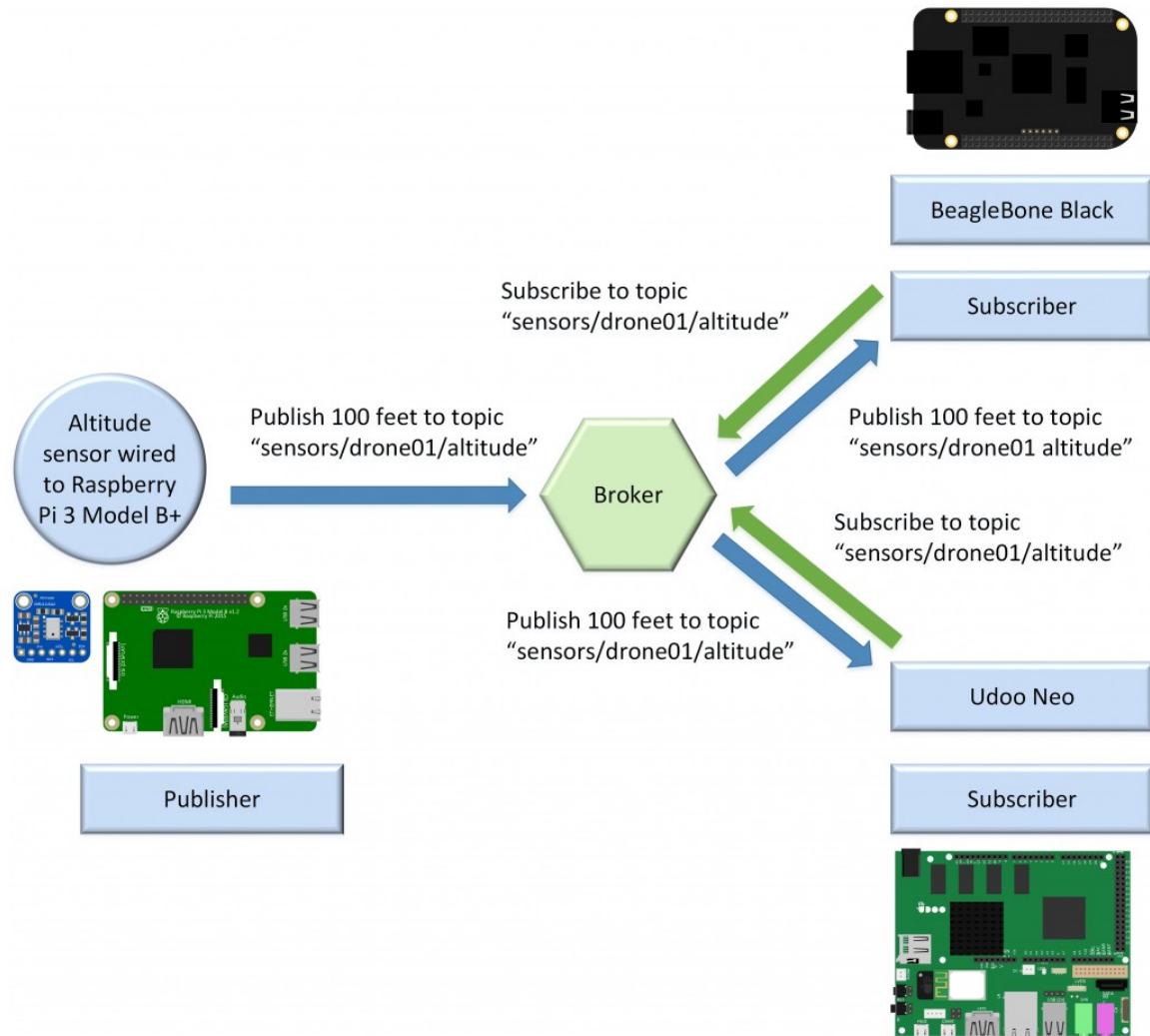
Understanding: publish-subscribe pattern



- `pub-sub` pattern requires a `server(broker)`
- All clients establish a connection with the server
- A client which sends a message through the server is known as `publisher`
- The server filters the incoming messages and distributes them to the clients that are interested
- Clients that register to the server as interested in specific types of messages are known as `subscribers`
- Both `publishers` and `subscribers` establish a connection with the server

MQTT

Understanding: publish-subscribe pattern



A topic is a named logical channel, also referred to as channel or subject.

MQTT

Working with message filtering

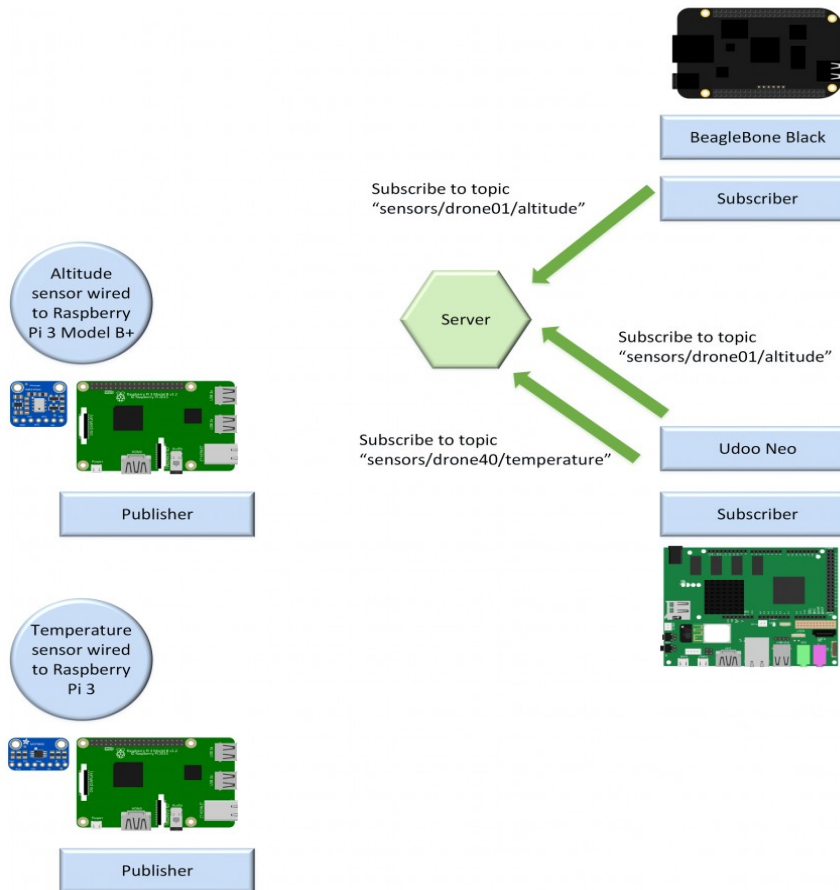
- The `server` will make sure that `subscribers` only receive the messages they are interested in
- Messages will get filter out based on different criteria in a `publish-subscribe` pattern
- One such filtering is `topic-based` also known as `subject-based` filtering
- `Example`
 - ✓ Consider each message belongs to a particular topic
 - ✓ When a publisher requests the server to publish a message, it must specify both the topic and the message
 - ✓ The server receives the messages and delivers it to all the subscribers that have subscribed to the topic to which the message belongs
- The subscriber can subscribe to more than one topic

Both Publishers and Subscribers are decoupled.

MQTT

Working with message filtering

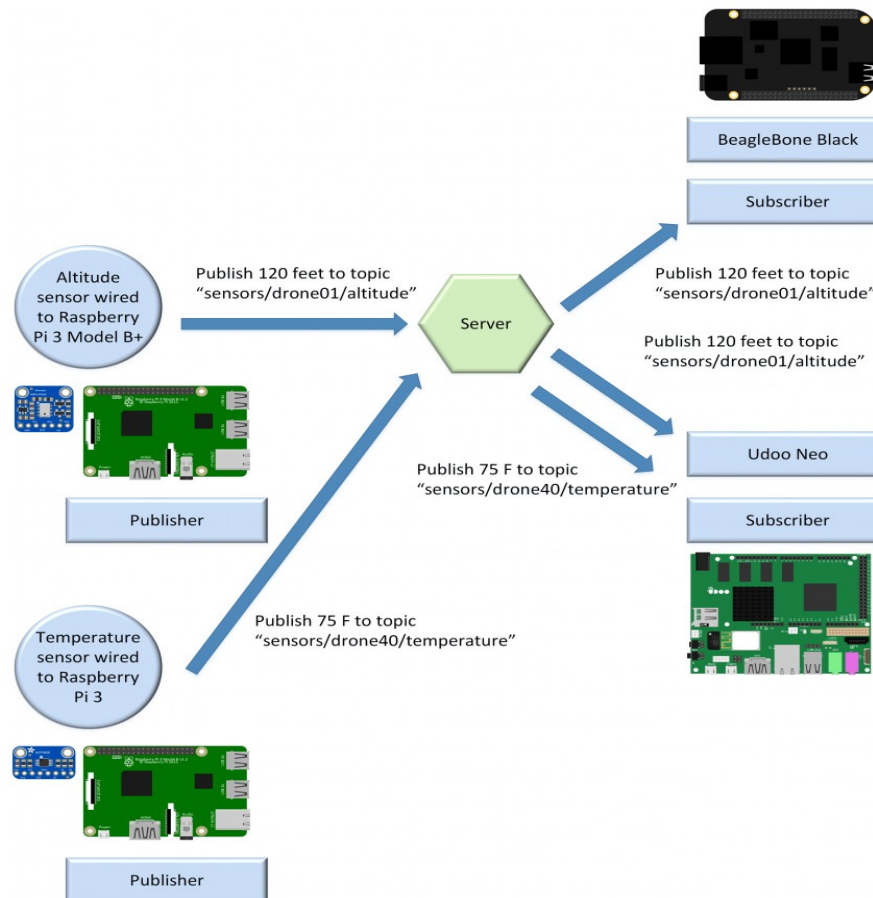
- The diagram shows two future publishers that haven't published any messages yet, a server and two subscribers connected to the server



MQTT

Working with message filtering

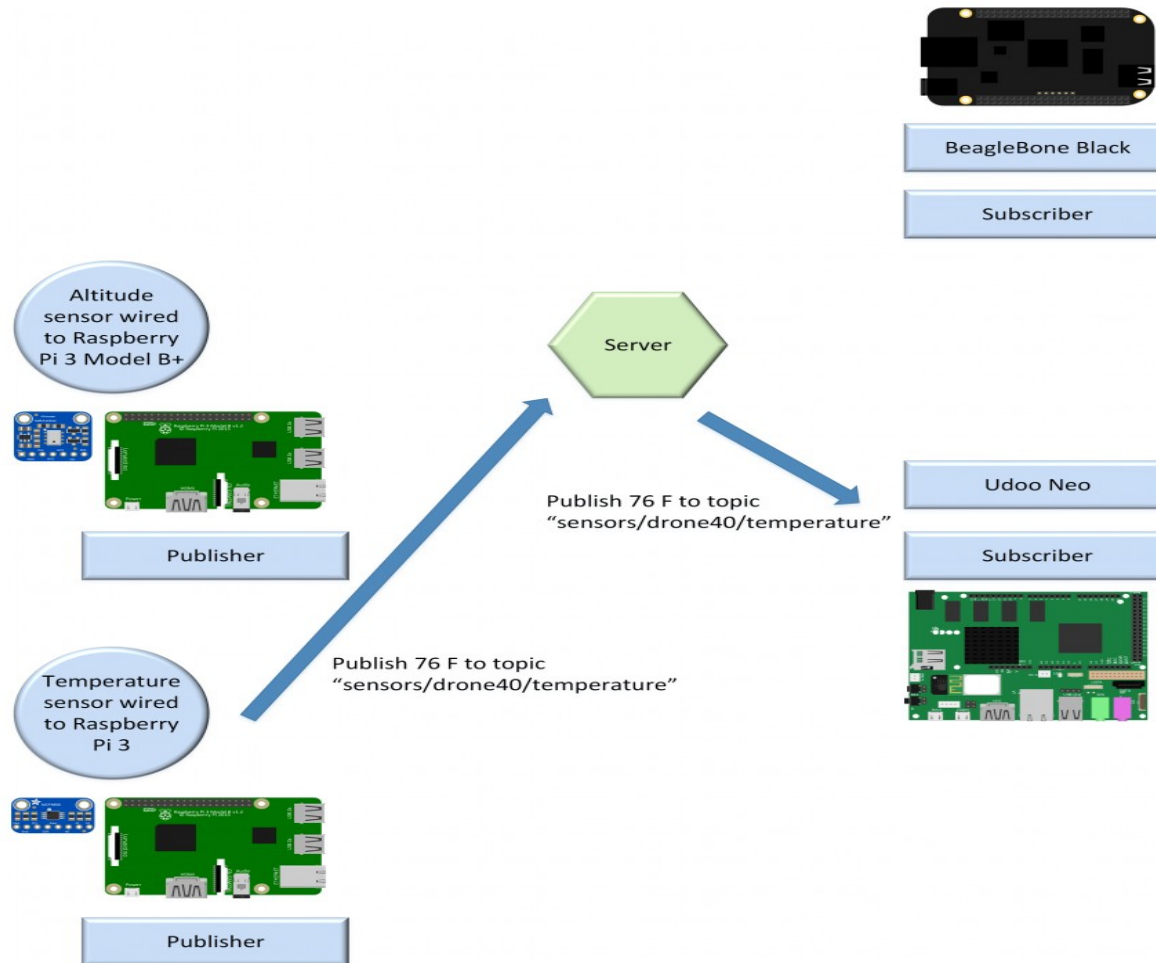
- The diagram shows what happens after two publishers connect and publish messages to different topics



MQTT

Working with message filtering

- The diagram shows what happens after one publisher publishes a message to a topic the server and the topic has only one subscriber



MQTT

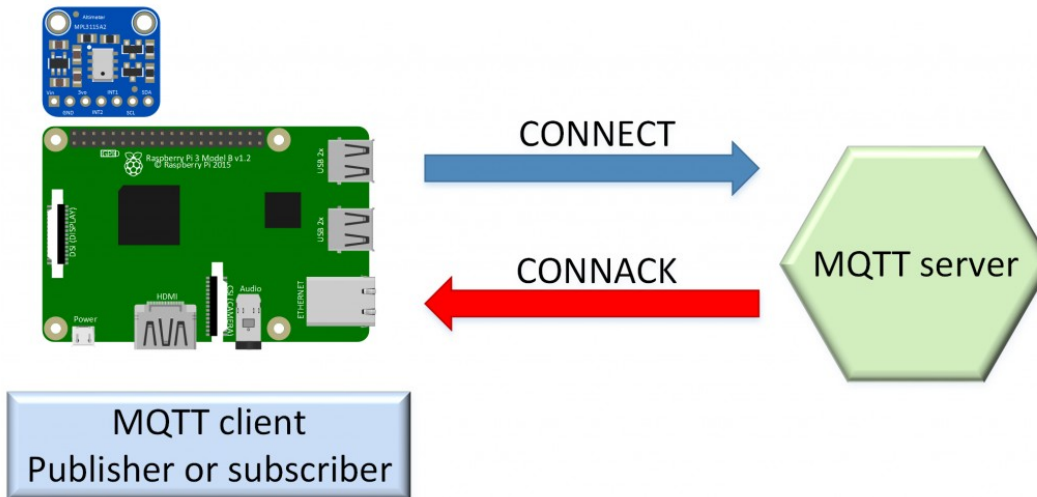
Understanding: MQTT puzzle

- MQTT servers, MQTT brokers, message brokers all are synonyms
- MQTT publishers and subscribers are completely decoupled, and also they are the MQTT clients which establish the connection with the server
- MQTT client can be both a publisher and subscriber at the same time
- Any device that has a TCP/IP stack and capable of using MQTT library can become an MQTT client i.e publisher or subscriber or both
- Selecting the right MQTT server is important
- The MQTT server is responsible for the authentication and authorization of MQTT clients that will be able to become publishers and / or subscribers

MQTT

Understanding: MQTT puzzle

- UNDERSTANDING MQTT CLIENT AND SERVER CONNECTION



After, successful connection has been established, the server will keep the connection open until the client loses the connection or sends a DISCONNECT packet to the server to close the connection

MQTT

Understanding: CONNECT packet

The CONNECT control packet contains the following fields in the payload,

ClientID:

- Is a string that identifies each MQTT client connected to the server
- If a client specifies the empty value as the client ID, then the server must generate a unique clientID

CleanSession:

- It is a flag with boolean value, which specifies what happens after MQTT client disconnect from the server and then reconnect
- If Flag = 1,
 - The client indicates to the server that the session will only last as long as the network connection is alive
 - After client disconnection, any information related to the previous session will be discarded

MQTT

Understanding: CONNECT packet

The CONNECT control packet contains the following fields in the payload,

CleanSession:

- If Flag = 0,
 - The server stores all the subscriptions for the client
 - When the client disconnects, the server stores all the messages that arrive with specific quality of service levels that match the subscriptions that the MQTT client had at the time of disconnection
 - When the client establishes a new connection, the client will have all the same subscriptions and continue to receive all the messages that it couldn't receive when it lost the connection

MQTT

Understanding: CONNECT packet

The CONNECT control packet contains the following fields in the payload,

Username

- If the client wants to specify the username to request authentication and authorization from the MQTT server, it must set the Username flag to 1 or True
- Specify the value of the Username field

Password

- If the client wants to specify the password to request authentication and authorization from the MQTT server, it must set the Password flag to 1 or True
- Specify the value of the Password field

MQTT

Understanding: CONNECT packet

The CONNECT control packet contains the following fields in the payload,

ProtocolLevel

- Indicates the MQTT protocol version that the MQTT client requests the MQTT server to use

KeepAlive

- Time interval expressed in seconds
- Client commits to send the control packets to the server within the time specified for KeepAlive
- If the client doesn't send any control packet then a PINGREQ control packet will be sent to server to indicate the client connection is Alive
- The server in turn, sends the PINGRESP control packet to the client to indicate the client connection is Alive
- The connection is closed, when there is an absence of these control packets
- If KeepAlive = 0, the keep alive mechanism is turned off

MQTT

Understanding: CONNACK packet

ReturnCode

ReturnCode Value	Description
0	The connection was accepted
1	The connection was refused because the MQTT server doesn't support the MQTT protocol version requested by the MQTT client in the CONNECT control packet
2	The connection was refused because the ClientID specified has been rejected
3	The connection refused because MQTT service isn't available even though network connection established
4	The connection refused because the username and the password values are malformed
5	The connection refused because the authorization failed

THANK YOU