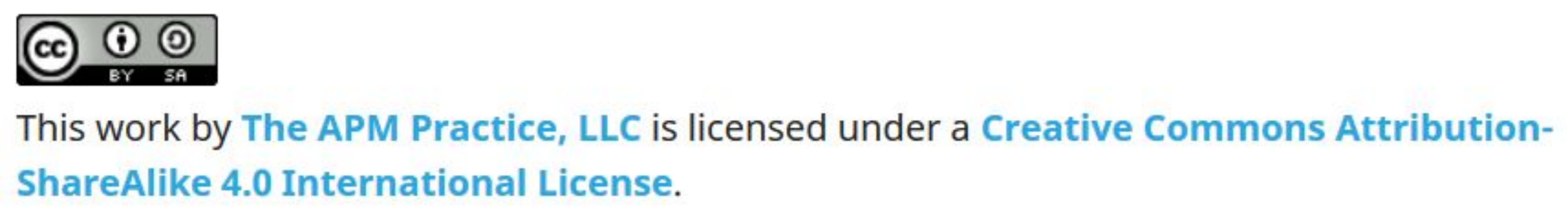
Using Messaging with Raspberry Pi

Tutorial /Lab

Mike Sydor, Software Performance Consultant



GOAL

- Do some messaging and start building the IoT (Internet of Things)
- Introduce both Clients and Brokers
- Work with Python
- Practice some software engineering concepts

How do you send data to another device?

What if that device isn’t ready?

> What happens to my data?

MOTIVATION

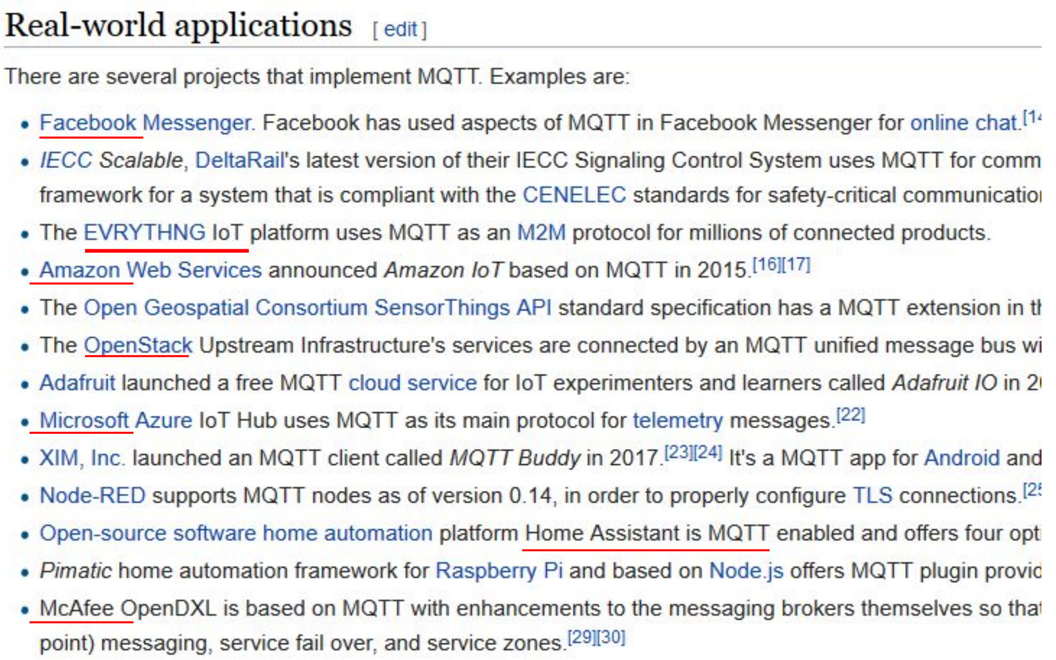
- The 'Pizza Shop' Monitor
  - Replace failed CCTV with Wireless solution
    - Need a Buzzer, as well as video feed - (2) raspberry pi
    - (1) device hosts the camera and motion sensor
    - (1) device hosts the monitor and buzzer
  - Communication via ad-hoc networking (no router or internet)
- Defensive Programming
  - What does my user know about computers?
  - When does the buzzer become annoying?
  - Fully ‘hands-free’ operation
    - Shutdown button

DIFFERENT WAYS TO GET IT DONE

- Primitive - run a long length of wire
  - Physics is not helping us! (signal loss)
  - Pulling cable - not really fun.
    - For a commercial facility, you need to be licensed...
- Classic - Configure a Web Server
  - publish a page with the sensor status
    - lots of problems
    - lots of software to configure and install
    - lots of latency
    - lots of missed posts (potentially)
- Efficient - Just notify me.

MESSAGING VIA MQTT

- *asynchronous* communication
  - just like texting via smartphone, or email - but faster!
  - client A via Verizon via Client B
- Publish/Subscribe
  - https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern
- From an IT perspective, the #1 strategy to integrate disparate systems
  - Data Integration
  - Process Integration
  - Event-driven Applications
    - Trading Systems



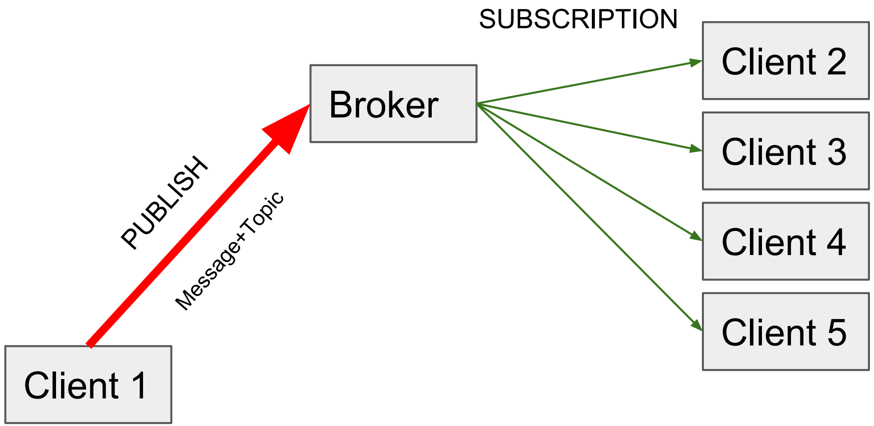
Real-world applications [edit]

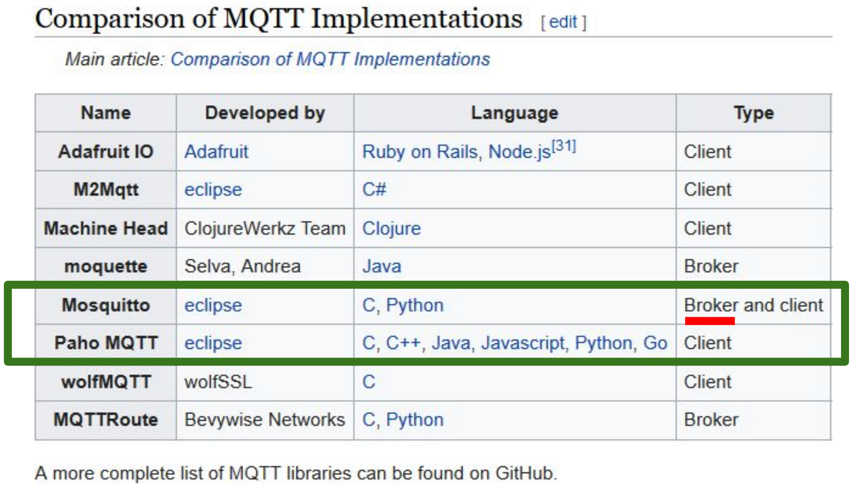
There are several projects that implement MQTT. Examples are:

- Facebook Messenger. Facebook has used aspects of MQTT in Facebook Messenger for online chat.[1
- *IECC Scalable*, DeltaRail's latest version of their IECC Signaling Control System uses MQTT for comm framework for a system that is compliant with the CENELEC standards for safety-critical communicatio
- The EVRYTHNG IoT platform uses MQTT as an M2M protocol for millions of connected products.
- Amazon Web Services announced *Amazon IoT* based on MQTT in 2015.[16][17]
- The Open Geospatial Consortium SensorThings API standard specification has a MQTT extension in th
- The OpenStack Upstream Infrastructure's services are connected by an MQTT unified message bus wi
- Adafruit launched a free MQTT cloud service for IoT experimenters and learners called *Adafruit IO* in 2
- Microsoft Azure IoT Hub uses MQTT as its main protocol for telemetry messages.[22]
- XIM, Inc. launched an MQTT client called *MQTT Buddy* in 2017.[23][24] It's a MQTT app for Android and
- Node-RED supports MQTT nodes as of version 0.14, in order to properly configure TLS connections.[2
- Open-source software home automation platform Home Assistant is MQTT enabled and offers four opt
- *Pimatic* home automation framework for Raspberry Pi and based on Node.js offers MQTT plugin provid
- McAfee OpenDXL is based on MQTT with enhancements to the messaging brokers themselves so tha point) messaging, service fail over, and service zones.[29][30]

MQTT[1] (**MQ Telemetry Transport** or **Message Queue Telemetry Transport**)

is an ISO standard (ISO/IEC PRF 20922)[2] publish-subscribe-based "**lightweight**" messaging protocol for use on top of the TCP/IP protocol. It is designed for *connections with remote locations* where a "**small code footprint**" is required or the network ***bandwidth is limited.*** The publish-subscribe messaging pattern requires a message broker. The broker is responsible for distributing messages to interested clients based on the topic of a message. Andy Stanford-Clark and Arlen Nipper of Cirrus Link authored the first version of the protocol in **1999**.[3]





Comparison of MQTT Implementations [edit]

Main article: Comparison of MQTT Implementations

Name	Developed by	Language	Type
Adafruit IO	Adafruit	Ruby on Rails, Node.js[31]	Client
M2Mqtt	eclipse	C#	Client
Machine Head	ClojureWerkz Team	Clojure	Client
moquette	Selva, Andrea	Java	Broker
Mosquitto	eclipse	C, Python	Broker and client
Paho MQTT	eclipse	C, C++, Java, Javascript, Python, Go	Client
wolfMQTT	wolfSSL	C	Client
MQTTRoute	Bevywise Networks	C, Python	Broker

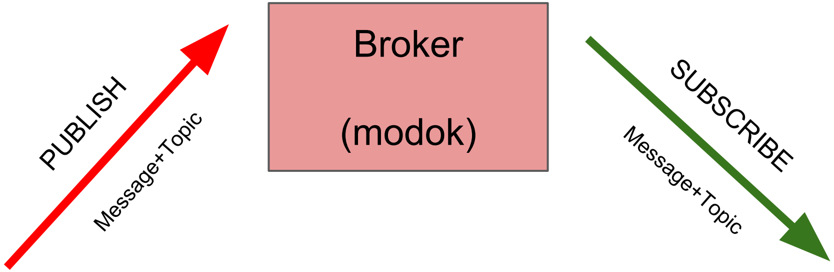
A more complete list of MQTT libraries can be found on GitHub.

https://en.wikipedia.org/wiki/MQTT

INSTALLATION

```
sudo apt-get install python3-pip          !! Only if you are missing pip

sudo pip install paho-mqtt

sudo apt-get install mosquitto            !! Only for configuring a BROKER
```

SOLUTION ARCHITECTURE



PUBLISH

Message+Topic

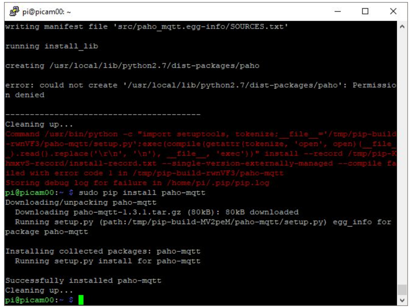
Broker

(modok)

SUBSCRIBE

Message+Topic

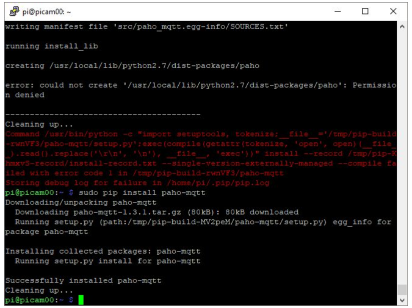
Window 1 - Publisher



```
writing manifest file 'src/paho_mqtt.egg-info/SOURCES.txt'

running install_lib

creating /usr/local/lib/python2.7/dist-packages/paho

error: could not create '/usr/local/lib/python2.7/dist-packages/paho': Permissio
n denied

----------------------------------------
Cleaning up...
Command /usr/bin/python -c "import setuptools, tokenize;__file__='/tmp/pip-build
-rwnVF3/paho-mqtt/setup.py';exec(compile(getattr(tokenize, 'open', open)(__file_
_).read().replace('\r\n', '\n'), __file__, 'exec'))" install --record /tmp/pip-K
hmxv5-record/install-record.txt --single-version-externally-managed --compile fa
iled with error code 1 in /tmp/pip-build-rwnVF3/paho-mqtt
Storing debug log for failure in /home/pi/.pip/pip.log
pi@picam00:~ $ sudo pip install paho-mqtt
Downloading/unpacking paho-mqtt
  Downloading paho-mqtt-1.3.1.tar.gz (80kB): 80kB downloaded
  Running setup.py (path:/tmp/pip-build-MV2peM/paho-mqtt/setup.py) egg_info for
package paho-mqtt

Installing collected packages: paho-mqtt
  Running setup.py install for paho-mqtt

Successfully installed paho-mqtt
Cleaning up...
pi@picam00:~ $
```

(separate terminal sessions)

Window 2 - Subscriber

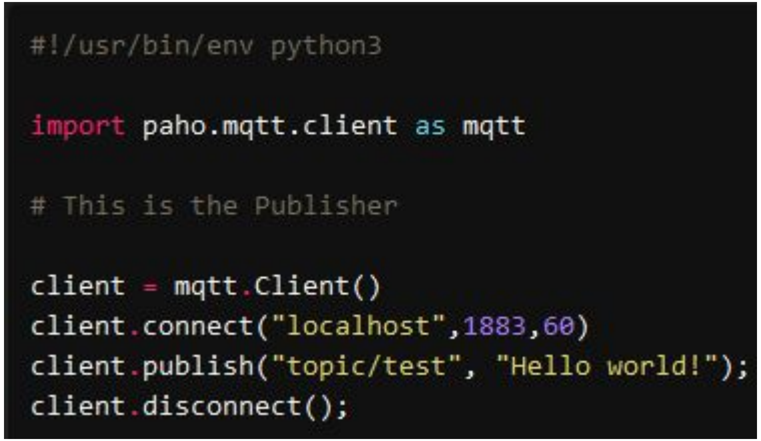


```
writing manifest file 'src/paho_mqtt.egg-info/SOURCES.txt'

running install_lib

creating /usr/local/lib/python2.7/dist-packages/paho

error: could not create '/usr/local/lib/python2.7/dist-packages/paho': Permissio
n denied

----------------------------------------
Cleaning up...
Command /usr/bin/python -c "import setuptools, tokenize;__file__='/tmp/pip-build
-rwnVF3/paho-mqtt/setup.py';exec(compile(getattr(tokenize, 'open', open)(__file_
_).read().replace('\r\n', '\n'), __file__, 'exec'))" install --record /tmp/pip-K
hmxv5-record/install-record.txt --single-version-externally-managed --compile fa
iled with error code 1 in /tmp/pip-build-rwnVF3/paho-mqtt
Storing debug log for failure in /home/pi/.pip/pip.log
pi@picam00:~ $ sudo pip install paho-mqtt
Downloading/unpacking paho-mqtt
  Downloading paho-mqtt-1.3.1.tar.gz (80kB): 80kB downloaded
  Running setup.py (path:/tmp/pip-build-MV2peM/paho-mqtt/setup.py) egg_info for
package paho-mqtt

Installing collected packages: paho-mqtt
  Running setup.py install for paho-mqtt

Successfully installed paho-mqtt
Cleaning up...
pi@picam00:~ $
```

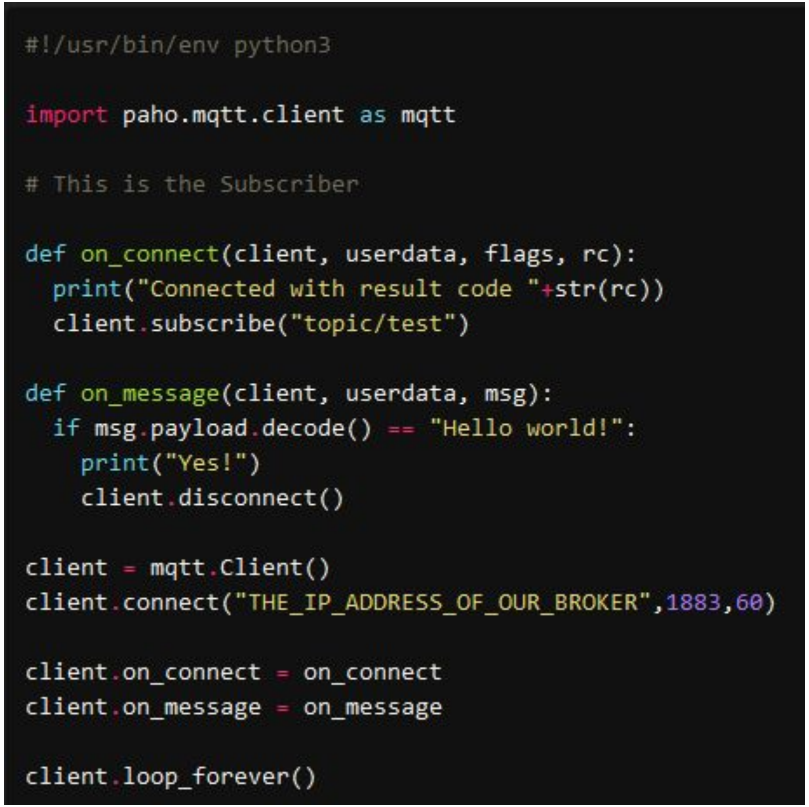
PSEUDOCODE

```
# create new instance of client object

# connect to broker

# Wait for connection

while CONNECTED != True:
      lightSleep(0.1)

# spin, waiting for messages
try:
      while True:
        get_message()
        lightSleep(1)

except KeyboardInterrupt:
      print ("exiting")

# clean up
#
```

IMPLEMENTATION



```
#!/usr/bin/env python3

import paho.mqtt.client as mqtt

# This is the Publisher

client = mqtt.Client()
client.connect("localhost",1883,60)
client.publish("topic/test", "Hello world!");
client.disconnect();
```



```
#!/usr/bin/env python3

import paho.mqtt.client as mqtt

# This is the Subscriber

def on_connect(client, userdata, flags, rc):
  print("Connected with result code "+str(rc))
  client.subscribe("topic/test")

def on_message(client, userdata, msg):
  if msg.payload.decode() == "Hello world!":
    print("Yes!")
    client.disconnect()

client = mqtt.Client()
client.connect("THE_IP_ADDRESS_OF_OUR_BROKER",1883,60)

client.on_connect = on_connect
client.on_message = on_message

client.loop_forever()
```

Challenge #1

- Create each script (publisher, subscriber)
- Make executable (chmod +x filename.py)
- Add/modify print statement to show what you have received
- The Broker is at 192.168.0.50 (modok)
- Open a separate window for each to run in
  - Publisher
  - Subscriber
- Send a message!

Challenge #2

- Change the client so it will accept messages continuously

Ad-hoc Networking

1. Create a backup of the wireless config
   a. `cp /etc/network/interfaces /etc/network/interface.` **`orig`**
   b. `cp /etc/network/interfaces /etc/network/interface.` **`wireless`**
   c. `cp /etc/network/interfaces /etc/network/interface.` **`adhoc`**
2. Modify the file `interfaces.adhoc` as follows

```
auto wlan0
iface wlan0 inet static
address 10.0.0.n
netmask 255.255.255.0
wireless-channel 1
wireless-essid scalici-video
```

n is any unique integer between 2 and 255

Everything else gets commented out...

To Go *adhoc*: `cp /etc/network/interfaces.` **`adhoc`** `to /etc/interfaces`

To Go *wireless*: `cp /etc/network/interfaces.` **`wireless`** `to /etc/interfaces`

Challenge #3

- Join the ad-hoc network @ scalici-video
- Publish a message, every 5 seconds, as follows:
    - MQ_BROKER='10.0.0.1'
    - MQ_TOPIC='pcam/motion'
    - message :: ‘Event’

RESOURCES

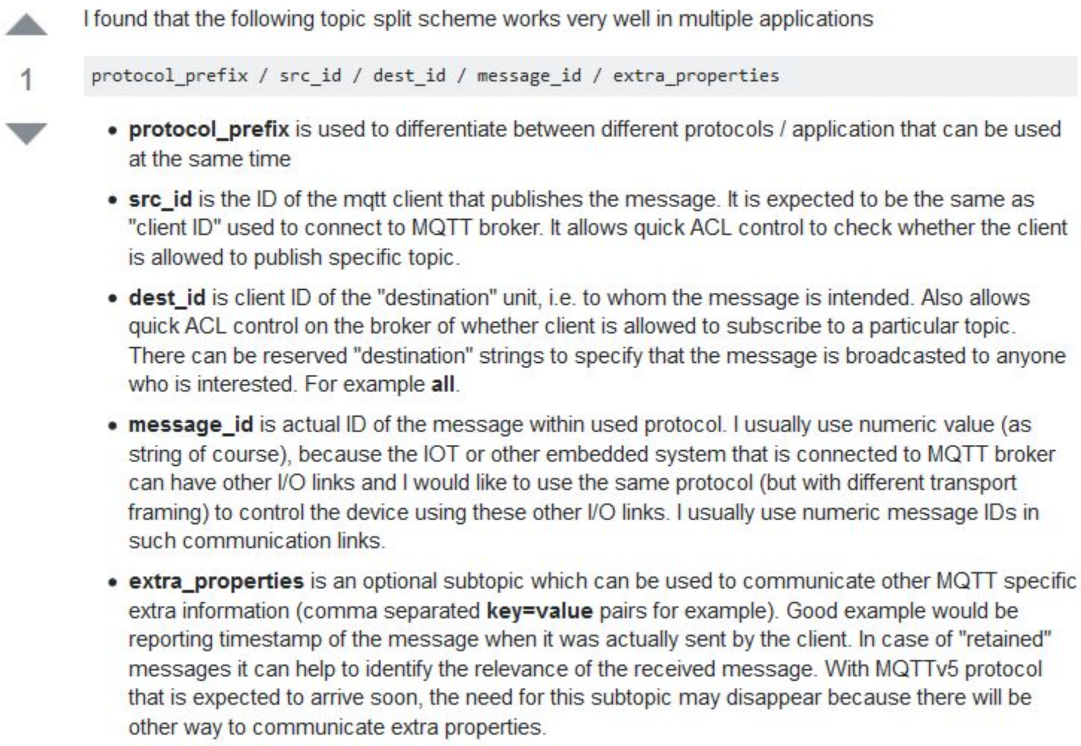
http://www.ev3dev.org/docs/tutorials/sending-and-receiving-messages-with-mqtt/

Red Pill

https://www.home-assistant.io/getting-started/

https://www.home-assistant.io/addons/google_assistant/

Message Scheme



I found that the following topic split scheme works very well in multiple applications

```
protocol_prefix / src_id / dest_id / message_id / extra_properties
```

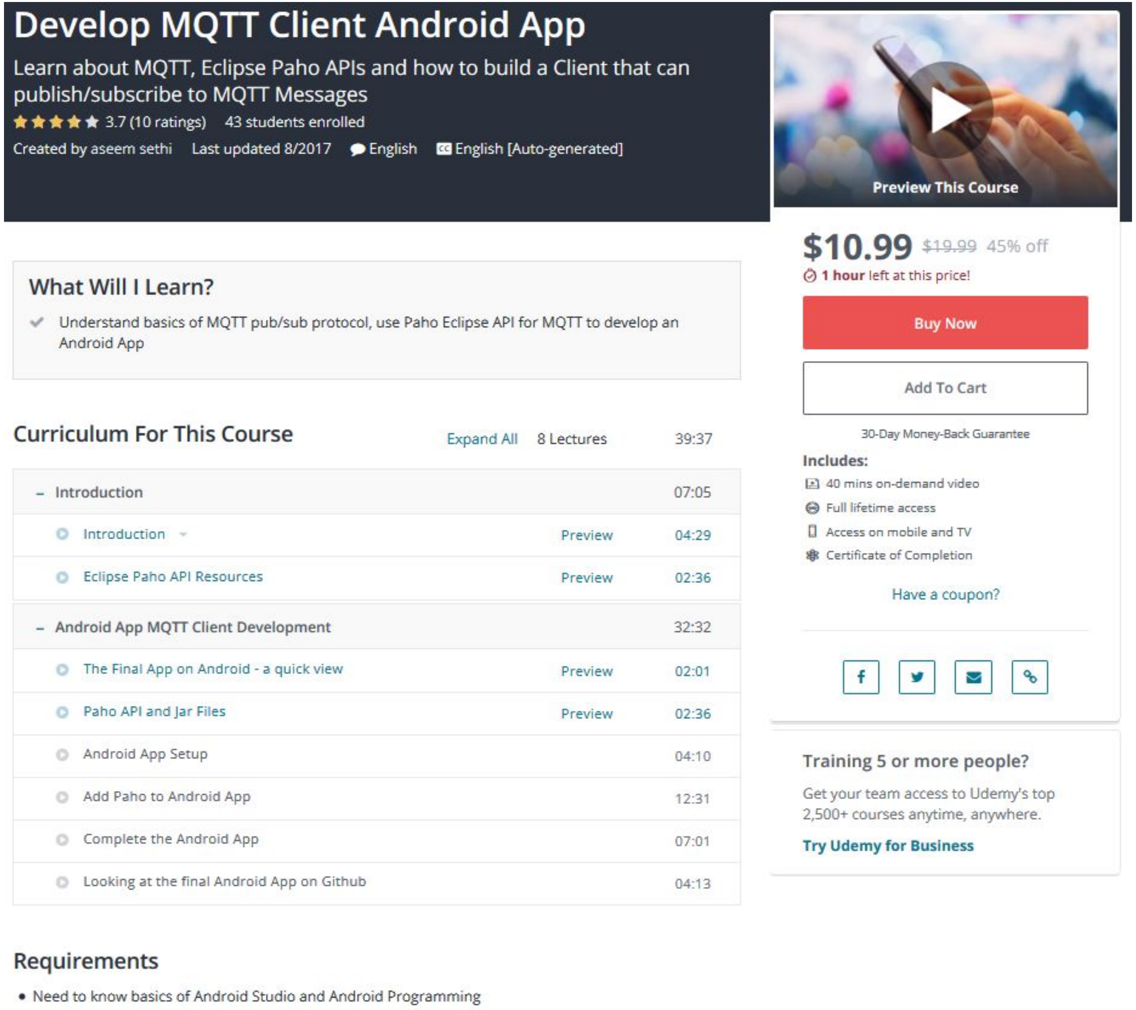
- **protocol_prefix** is used to differentiate between different protocols / application that can be used at the same time
- **src_id** is the ID of the mqtt client that publishes the message. It is expected to be the same as "client ID" used to connect to MQTT broker. It allows quick ACL control to check whether the client is allowed to publish specific topic.
- **dest_id** is client ID of the "destination" unit, i.e. to whom the message is intended. Also allows quick ACL control on the broker of whether client is allowed to subscribe to a particular topic. There can be reserved "destination" strings to specify that the message is broadcasted to anyone who is interested. For example **all**.
- **message_id** is actual ID of the message within used protocol. I usually use numeric value (as string of course), because the IOT or other embedded system that is connected to MQTT broker can have other I/O links and I would like to use the same protocol (but with different transport framing) to control the device using these other I/O links. I usually use numeric message IDs in such communication links.
- **extra_properties** is an optional subtopic which can be used to communicate other MQTT specific extra information (comma separated **key=value** pairs for example). Good example would be reporting timestamp of the message when it was actually sent by the client. In case of "retained" messages it can help to identify the relevance of the received message. With MQTTv5 protocol that is expected to arrive soon, the need for this subtopic may disappear because there will be other way to communicate extra properties.

https://stackoverflow.com/questions/48238365/good-way-to-design-mqtt-topic

WRAP-UP

- Messaging - the foundation of IoT
- A few moving parts - but not that difficult to use
- You need to give some thought to what messages you want to support

Thanks for Your Attention



Develop MQTT Client Android App

Learn about MQTT, Eclipse Paho APIs and how to build a Client that can publish/subscribe to MQTT Messages

★★★★★ 3.7 (10 ratings) 43 students enrolled

Created by aseem sethi Last updated 8/2017 English English [Auto-generated]

$10.99 ~~$19.99~~ 45% off

1 hour left at this price!

30-Day Money-Back Guarantee

Includes:

- 40 mins on-demand video
- Full lifetime access
- Access on mobile and TV
- Certificate of Completion

Have a coupon?

What Will I Learn?

✓ Understand basics of MQTT pub/sub protocol, use Paho Eclipse API for MQTT to develop an Android App

Curriculum For This Course

Expand All	8 Lectures	39:37
– Introduction		07:05
Introduction	Preview	04:29
Eclipse Paho API Resources	Preview	02:36
– Android App MQTT Client Development		32:32
The Final App on Android - a quick view	Preview	02:01
Paho API and Jar Files	Preview	02:36
Android App Setup		04:10
Add Paho to Android App		12:31
Complete the Android App		07:01
Looking at the final Android App on Github		04:13

Training 5 or more people?

Get your team access to Udemy's top 2,500+ courses anytime, anywhere.

Try Udemy for Business

Requirements

- Need to know basics of Android Studio and Android Programming