



# PRACTICAL MQTT FOR THE INTERNET OF THINGS

## About me:

Senior software engineer and open source enthusiast

- Maker of certified open source hardware for Internet of Things
- 12+ years of professional experience
- Master of Information Technologies and Bachelor of Computer Systems and Technologies of Technical University Sofia
- Speaker at various open source events in San Francisco, Portland (OR), Hong Kong, Shanghai, Shenzhen, Brussels, Berlin, Bratislava, Prague, Sofia and Plovdiv



# MQTT Course

1. 2 Days

2. 6 Sections

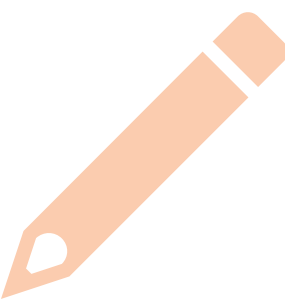
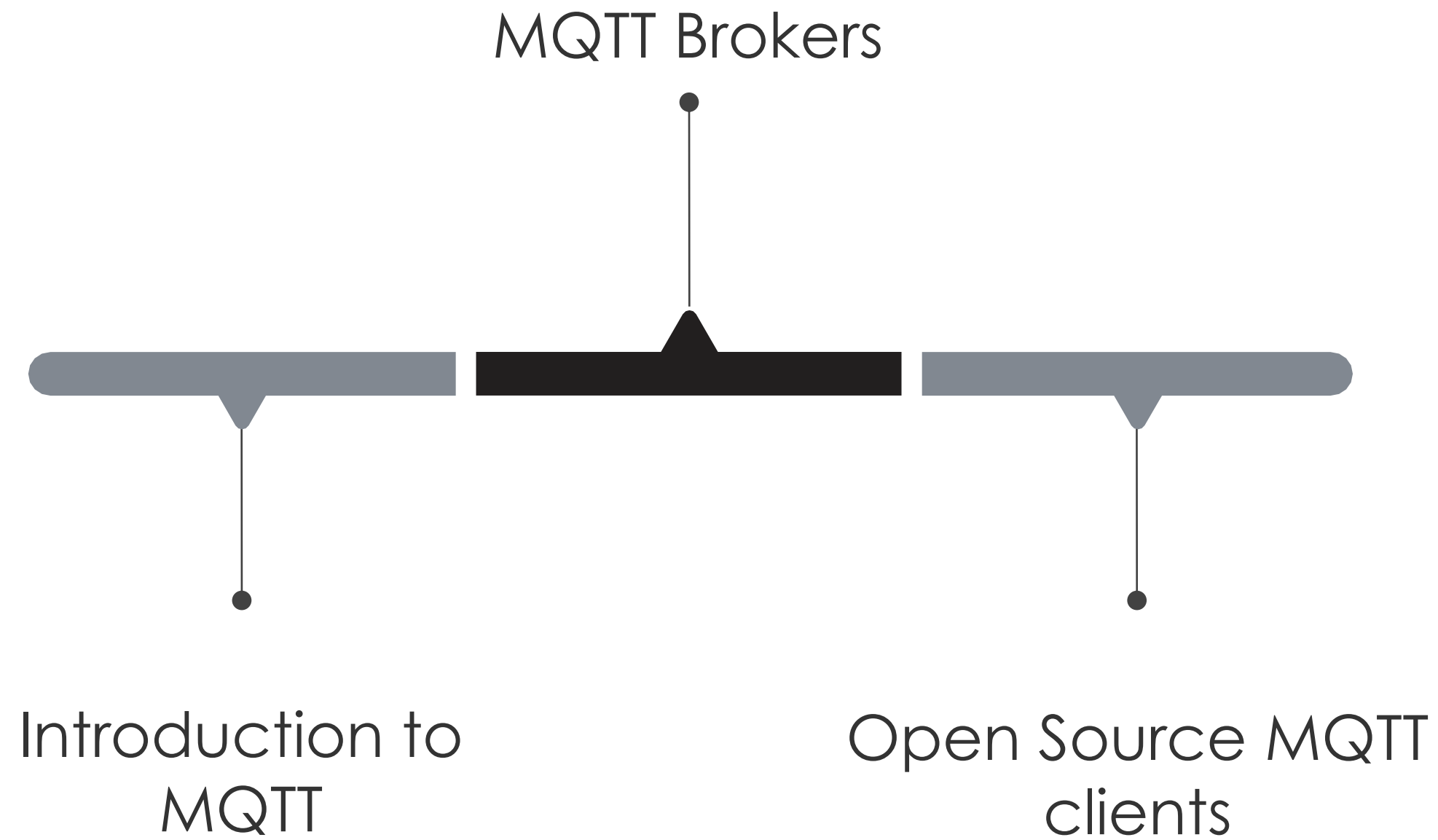
3. 6 Lab exercises

4. 180+ slides

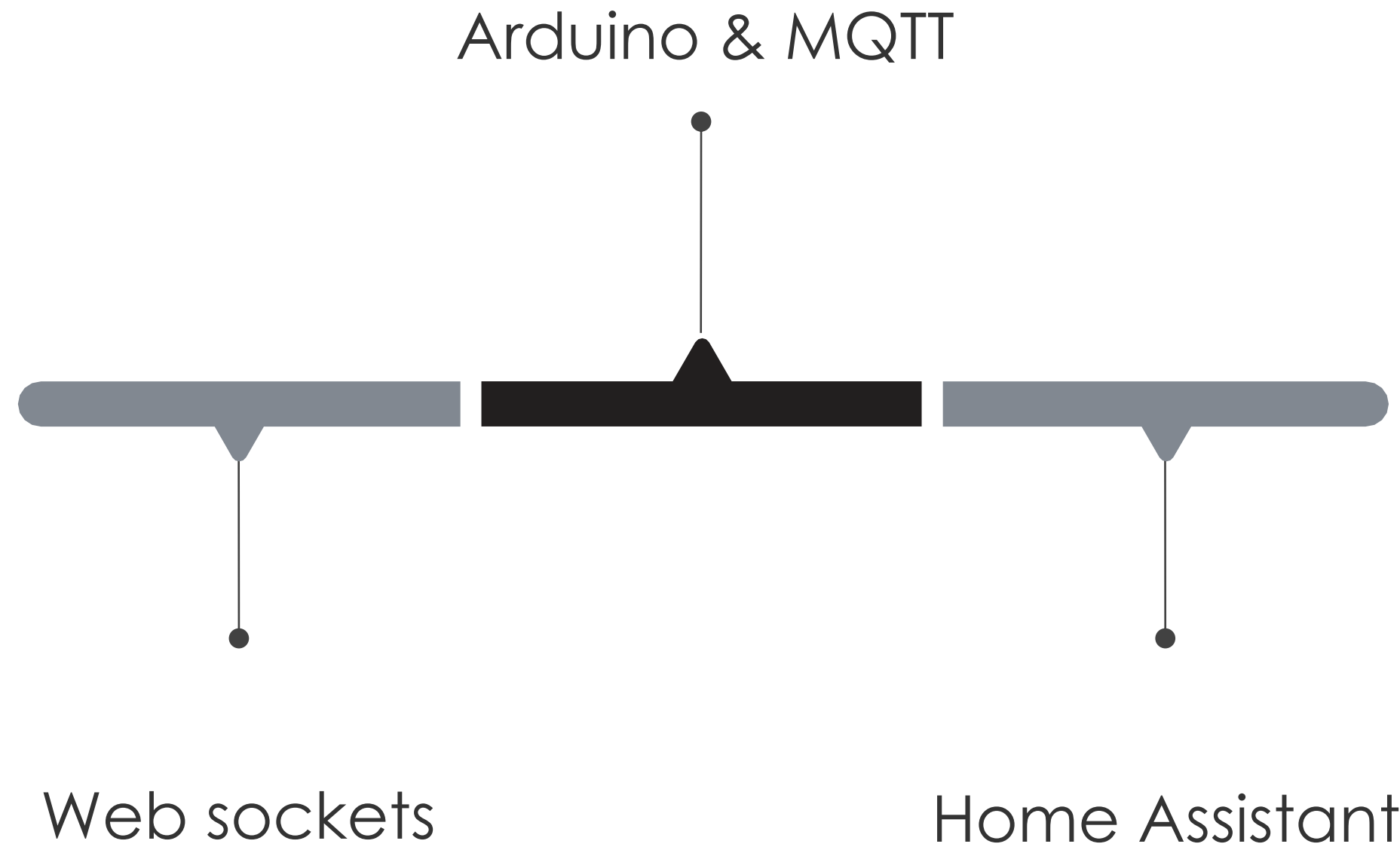
5. Examples in  
GitHub



# Roadmap Day 1



## Roadmap Day 2





# Section 1: Introduction to MQTT

## In this section you will learn...



Overview of popular protocols for real-time communication between Internet of Things



Understanding how MQTT works



Reviewing MQTT key features



# Popular Internet messaging protocols for Internet of Things (IoT)

- Message Queue Telemetry Transport (MQTT)
- MQTT for Sensor Networks (MQTT-SN)
- Constrained Application Protocol (CoAP)
- Advanced Message Queuing Protocol (AMQP)
- Data Distribution Service (DDS)





## What is MQTT?

- Lightweight publish/subscribe machine-to-machine protocol on top of TCP/IP
- Near real-time communication between clients through MQTT message broker
- Small source code footprint for embedded devices
- Protocol versions 3.1, 3.1.1 and 5.0
- ISO standard (ISO/IEC 20922)
- MQTT-SN (MQTT for Sensor Networks) uses UDP

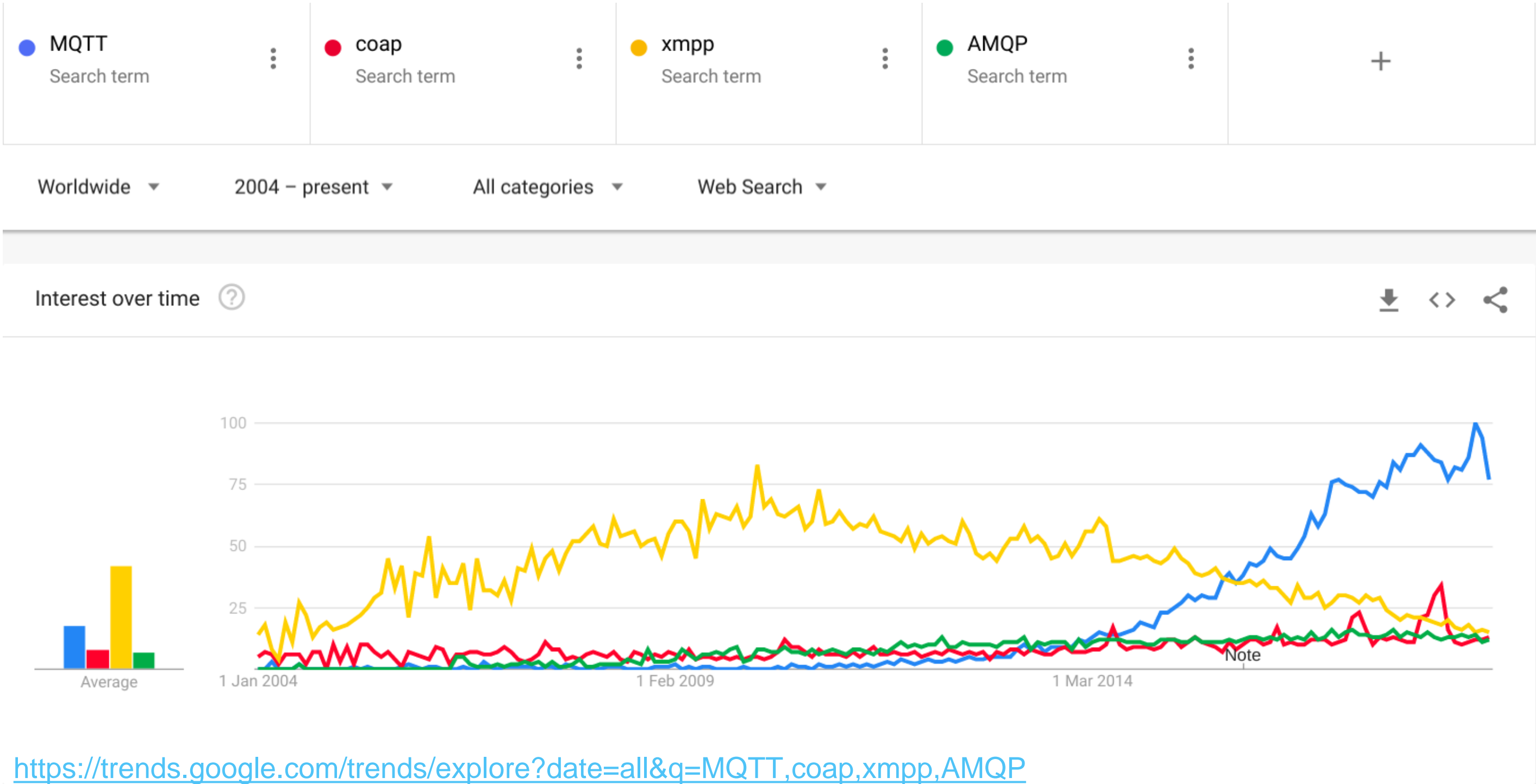


## Brief History of MQTT

- Created by Dr Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom (now Eurotech) in 1999
- Available under a royalty free license as protocol version 3.1 since 2010
- OASIS standard since 2014
- ISO standard (ISO/IEC 20922) since 2016
- MQTT version 5 was released as a specification in 2018

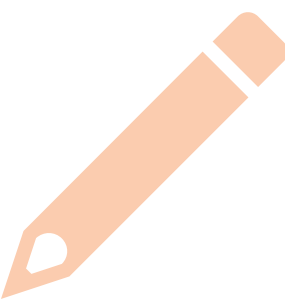


# IoT Communication Trends



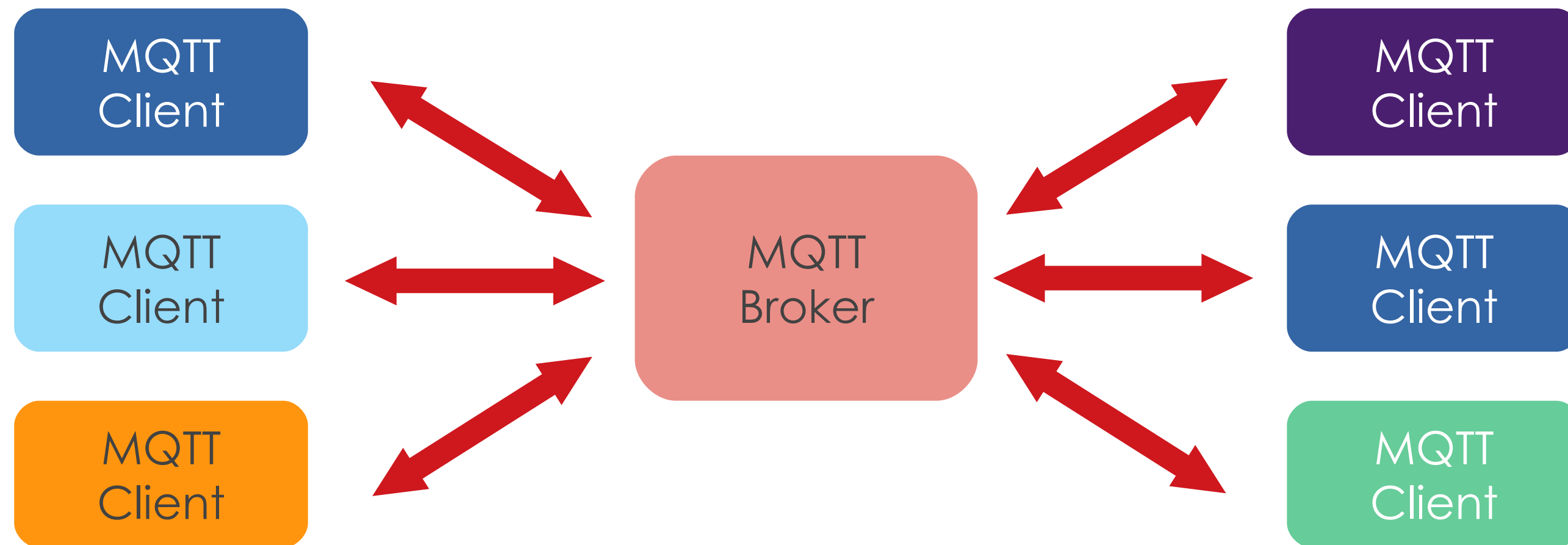
# MQTT Operations

- Connect
- Disconnect
- Subscribe
- Unsubscribe
- Publish



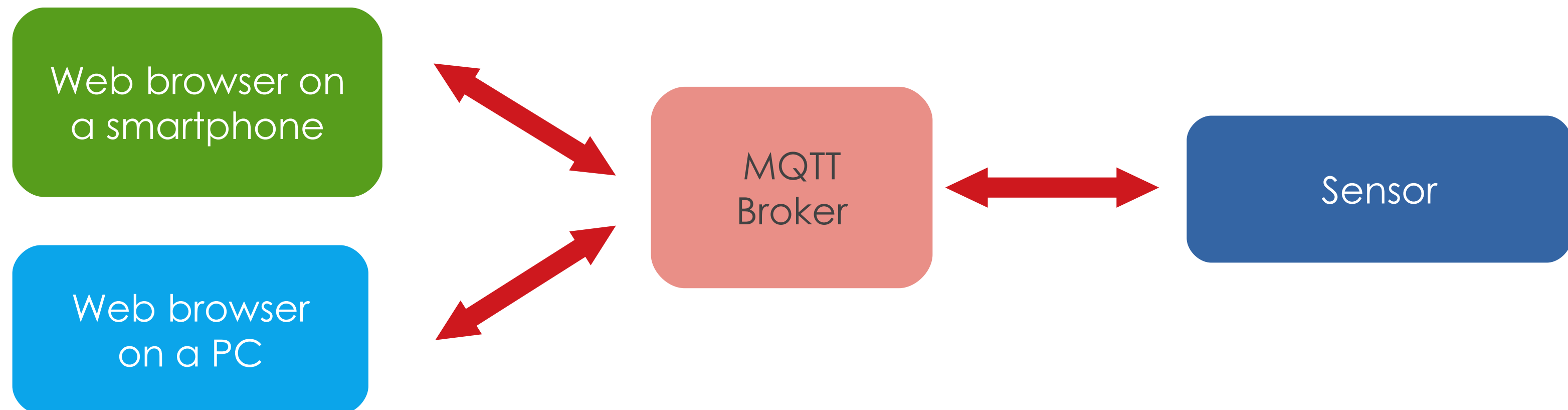
# MQTT Broker

- Manages messages between numerous connected MQTT clients
- Deliver published messages to the subscribed MQTT clients based on the topic of the message



## Example usage

- Web page, loaded on web browsers, connect as MQTT clients via web sockets and subscribe to a topic
- Application that reads data from a sensor and publishes message which the MQTT broker delivers to the subscribed clients



# MQTT Message

- Topic
- Payload (text or binary)
- Quality of service (0, 1 or 2)
- Retain (true or false)
- Furthermore, the MQTT protocol supports Last Will & Testament (LWT) which allows the broker to notify interested clients about an ungracefully disconnected client by publishing a message on his behalf

## Example message

- |           |                       |
|-----------|-----------------------|
| • Topic   | "hello/1"             |
| • Payload | { "temperature": 20 } |
| • QoS     | 2                     |
| • Retain  | false                 |



## MQTT Topics and Wildcards

- Wildcards define regular expressions used by MQTT client to subscribe for multiple similar topics simultaneously
- Topic
- **home / bedroom / temperature**
- Single level wildcards
- **home / + / temperature**
- Multi level wildcards
- **home / #**





## MQTT Quality of Service (QoS)

- .**0** - at most once (no delivery guarantee)
- .**1** - at least once
- .**2** - exactly once



## Retained messages

- Message published with retain flag set to **true**
- Useful for storing the last known good value
- The MQTT broker is responsible for transmitting the retained message to all MQTT clients interested in this topic, even if they have connected after the time when the message has been published
- Deletes a retained message by publishing another message with the same topic and an empty payload



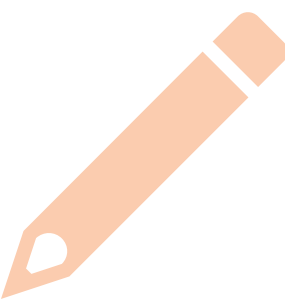
## Last Will & Testament

- Last Will & Testament (LWT) allows the broker to notify interested clients about an ungracefully disconnected client by publishing a message on his behalf
- The MQTT client should register the will message when connecting to the MQTT broker
- The minimum requirement for a will message is to specify at least a topic



# MQTT Security

- Transport encryption with TLS/SSL
- Authentication: username/password
- Authorization: Access Control Lists (ACL)



# MQTT Protocol Versions

- **3.1** (protocol standard from 2010)
- **3.1.1** (protocol standard from 2014)
- **5.0** (protocol standard from 2018)

For more details visit: <http://mqtt.org/documentation>



## What Happened to MQTT 4?

- There is no MQTT 4
- 8 bit unsigned value represents the protocol level (aka version) in the variable header of the CONNECT Packet
- “The value of the Protocol Level field for the version 3.1.1 of the protocol is 4 (0x04)”  
<http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/csprd02/mqtt-v3.1.1-csprd02.html>
- “The value of the Protocol Level field for the version 5.0 of the protocol is 5 (0x05)”  
<http://docs.oasis-open.org/mqtt/mqtt/v5.0/cs02/mqtt-v5.0-cs02.html>



## What is new in MQTT 5?

- Improvements and provisioning features for large scale systems
- Metadata and user properties
- Request/response interactions
- Improvements for authentication, error handling, lower bandwidth consumption and performance on clients with restrained hardware



# One more thing...Day 1 Lab exercises

Day 1 **Lab exercises** can be done with real hardware or with a virtual machine:

- VirtualBox with Ubuntu 18.04
- Raspberry Pi with Raspbian





## Oracle VM VirtualBox

- General-purpose full virtual machine for x86 hardware
- Suitable for server, desktop and embedded use
- Available for free download for MS Windows, OS X and Linux distributions
- <https://www.virtualbox.org/>



## Raspberry Pi

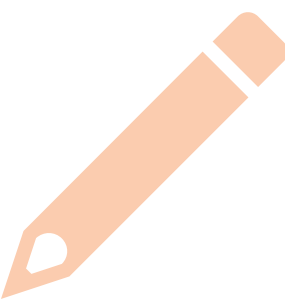
- Low cost single board computer developed in the UK by the Raspberry Pi Foundation
- Uses Broadcom ARM SoC
- Available with size of credit card (85x56mm), even smaller (65x30mm) or as a industrial compute module
- As of mid 2018 more than 19 million units have been sold worldwide





## Raspberry Pi Generations

- Raspberry Pi Model A and Model B
- Raspberry Pi Model B+
- Raspberry Pi 2 Model B
- Raspberry Pi 3 Model B
- Raspberry Pi 3 Model A+ and Model B+
- Raspberry Pi 0
- Raspberry Pi 0 W



# Industrial Compute Modules

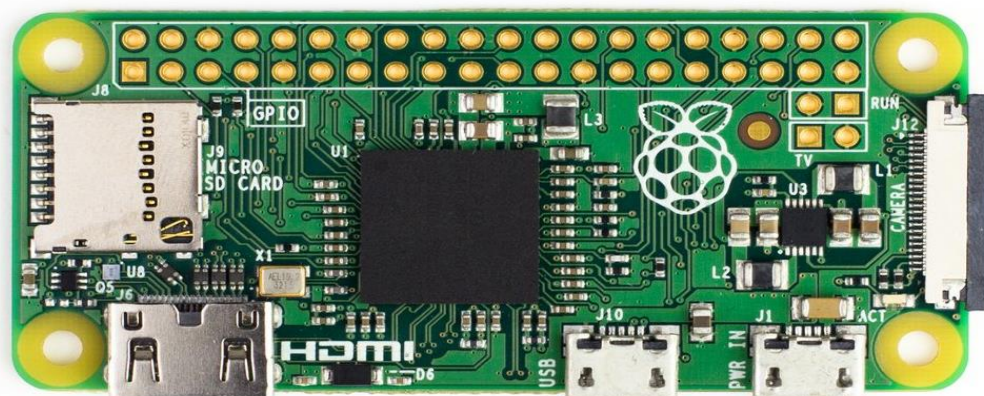
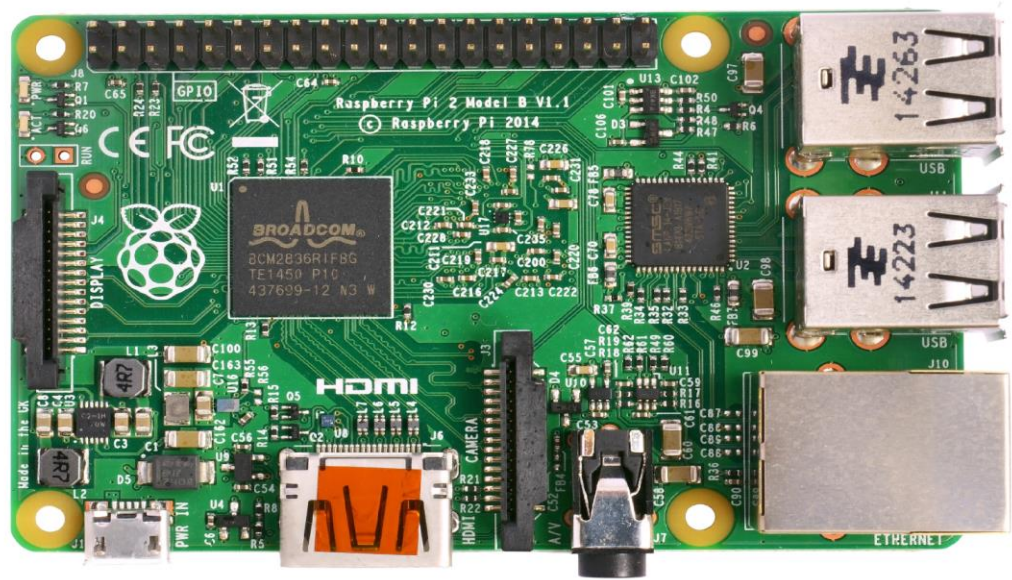
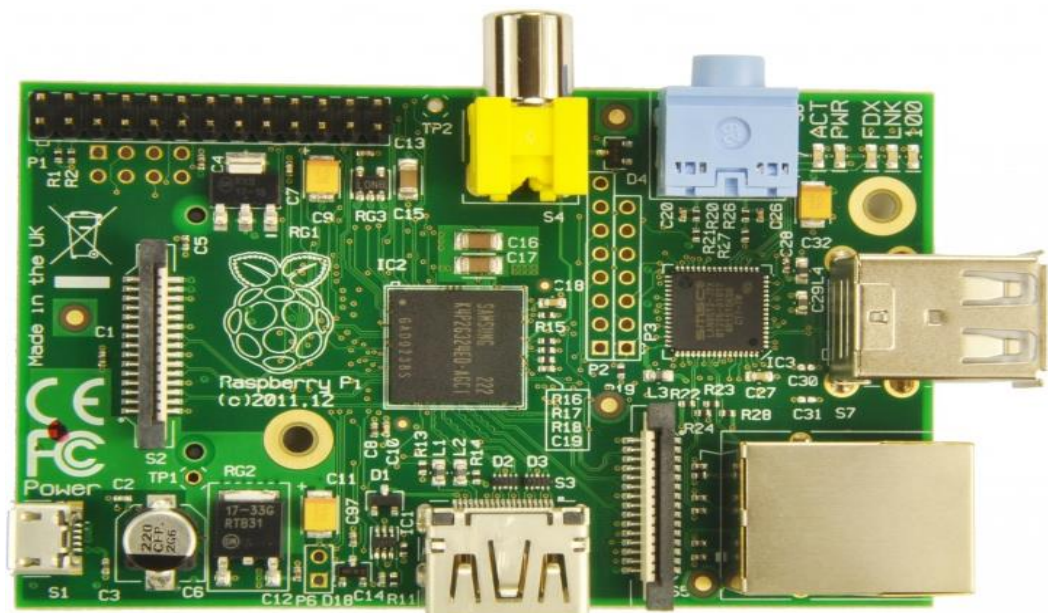
## Raspberry Pi Compute Module 1

- Raspberry Pi Compute Module 3 Lite
- Raspberry Pi Compute Module 3
- Raspberry Pi Compute Module IO Board





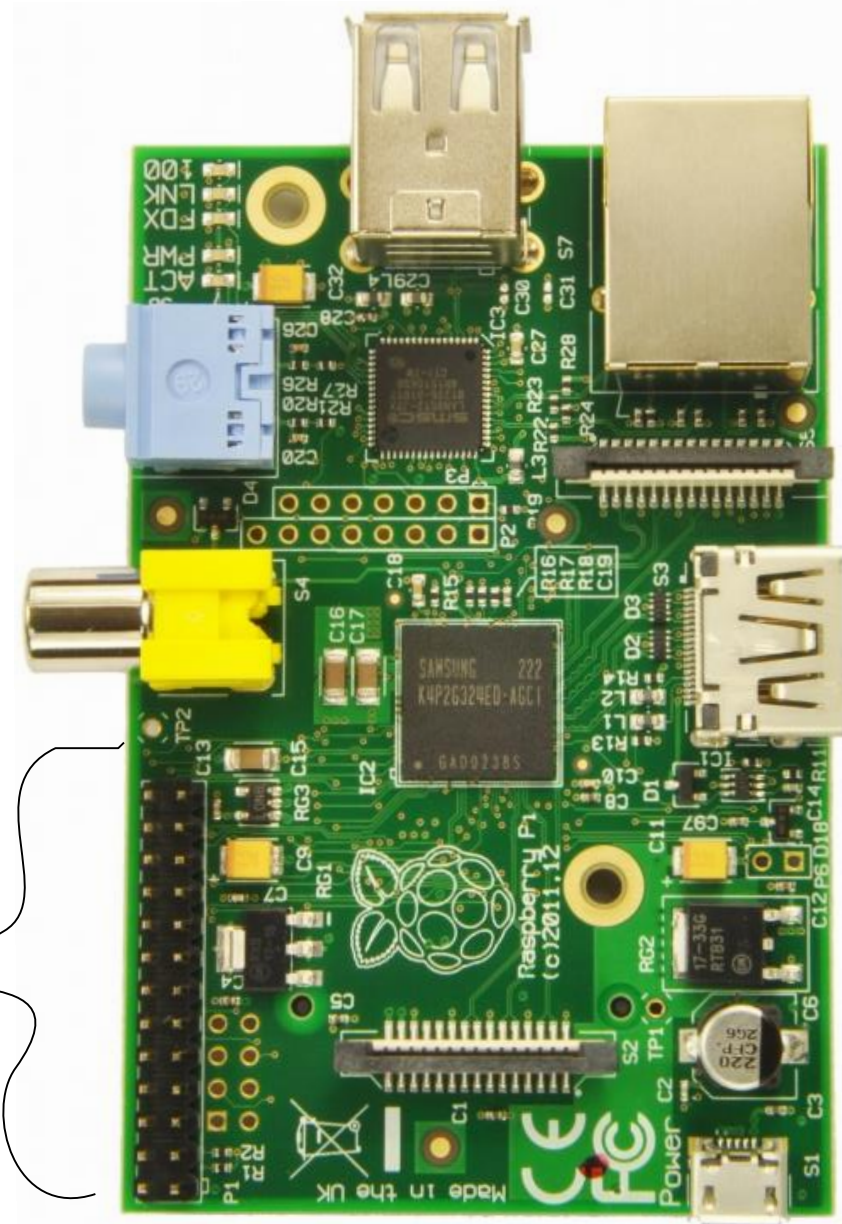
# Raspberry Pi Flavors



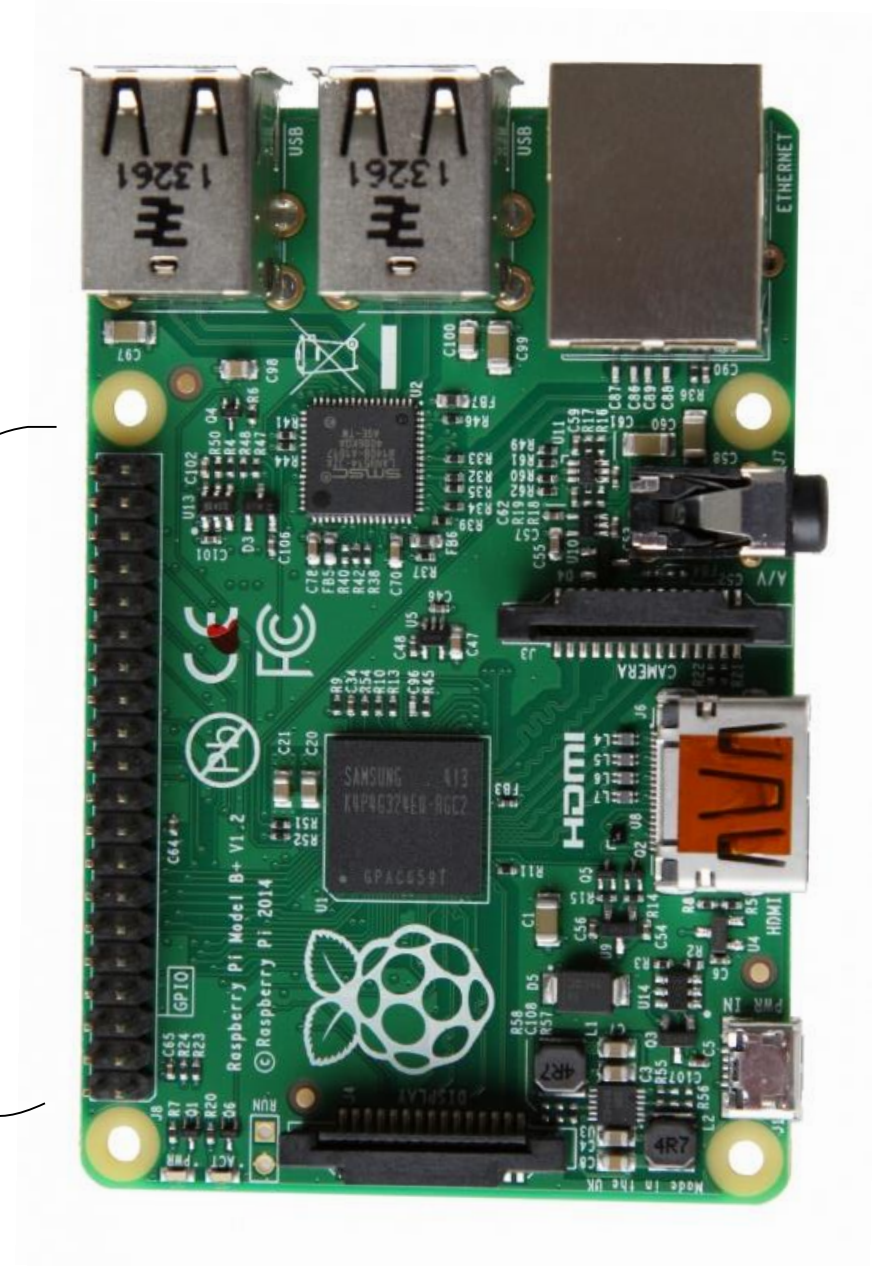


## Changes since B+

26 pins



40 pins



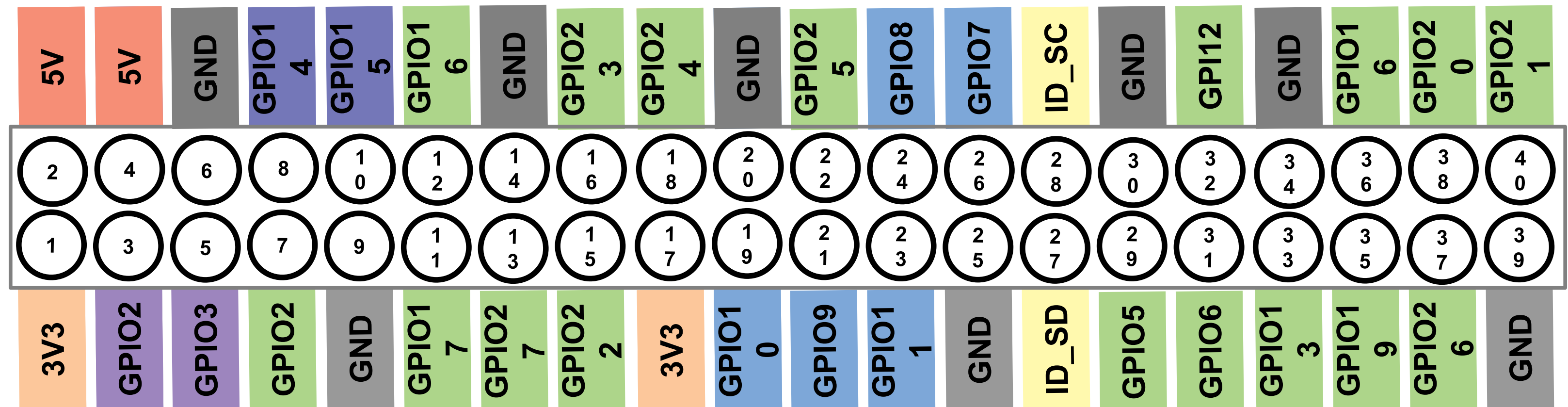
• Raspberry Pi Model B (2012)

Raspberry Pi Model B+ (2014)



# Raspberry Pi Pinout

- 40 pin header
- Specific pins for I2C, SPI, PWM and serial





## In this section you learned...

- What the popular protocols for real-time IoT communication are
- What MQTT is
- What the key features of MQTT are



# ? Questions?

# Lab 1:

## Installing Mosquitto on Raspberry Pi or VirtualBox

## Scenario:

1. Download Raspbian
2. Install to microSD card
3. Boot and configure Raspbian GNU/Linux distribution on Raspberry Pi.
4. Enable serial console output and I2C through raspi-config.
5. Install the open source MQTT broker mosquitto.

*Alternatively, you can use VirtualBox and Ubuntu 18.04 instead of Raspberry Pi*

## Aim:

Setup MQTT broker on Raspberry Pi



## Required hardware

- Raspberry Pi 3 Model B or B+
- 5V/2.5A USB power supply
- Monitor with HDMI input
- HDMI cable
- USB keyboard
- USB mouse
- MicroSD card
- Personal computer (with Windows, Mac OS or a Linux distribution)



## Steps

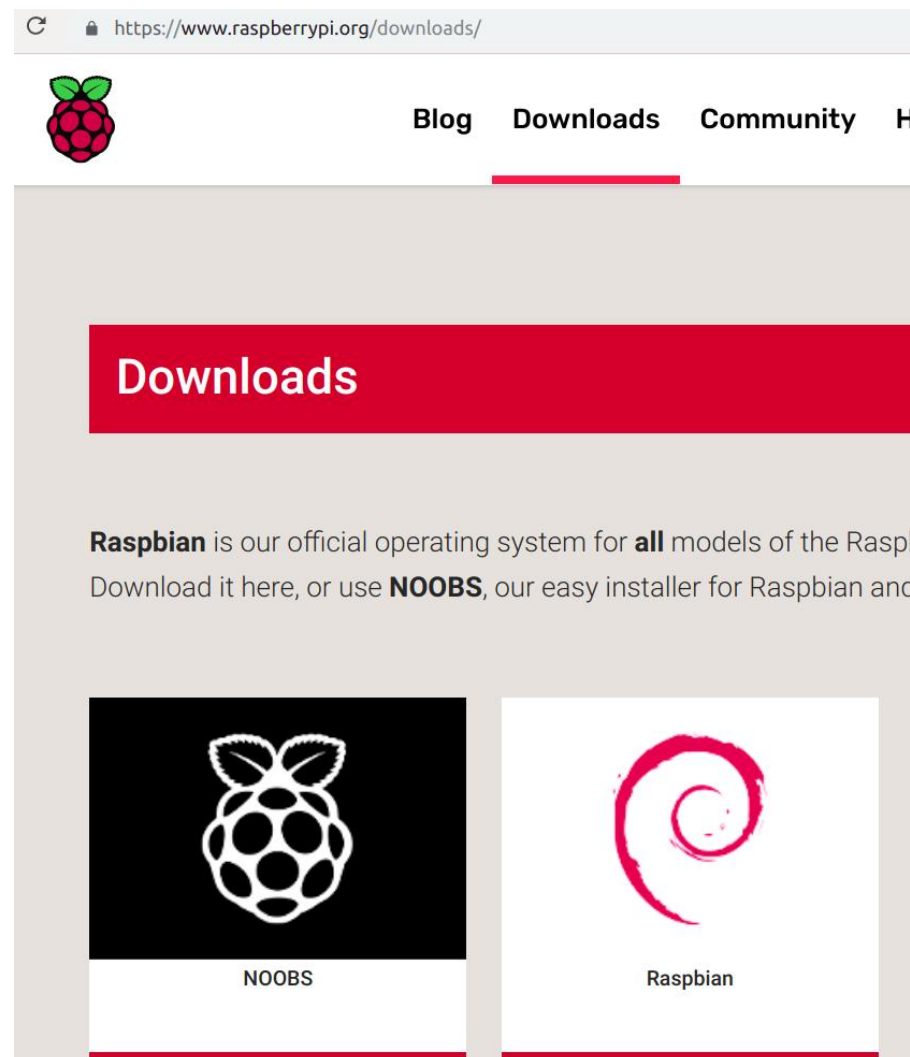
1. Download Raspbian
2. Install to microSD card
3. Boot and configure Raspbian GNU/Linux distribution on Raspberry Pi
4. Enable serial console output through raspi-config
5. Enable I2C through raspi-config
6. Install mosquitto
- 7. `sudo apt-get update`**
- 8. `sudo apt-get install -y mosquitto mosquitto-clients`**

**Alternatively:** Install Ubuntu 18.04 LTS and mosquitto on VirtualBox



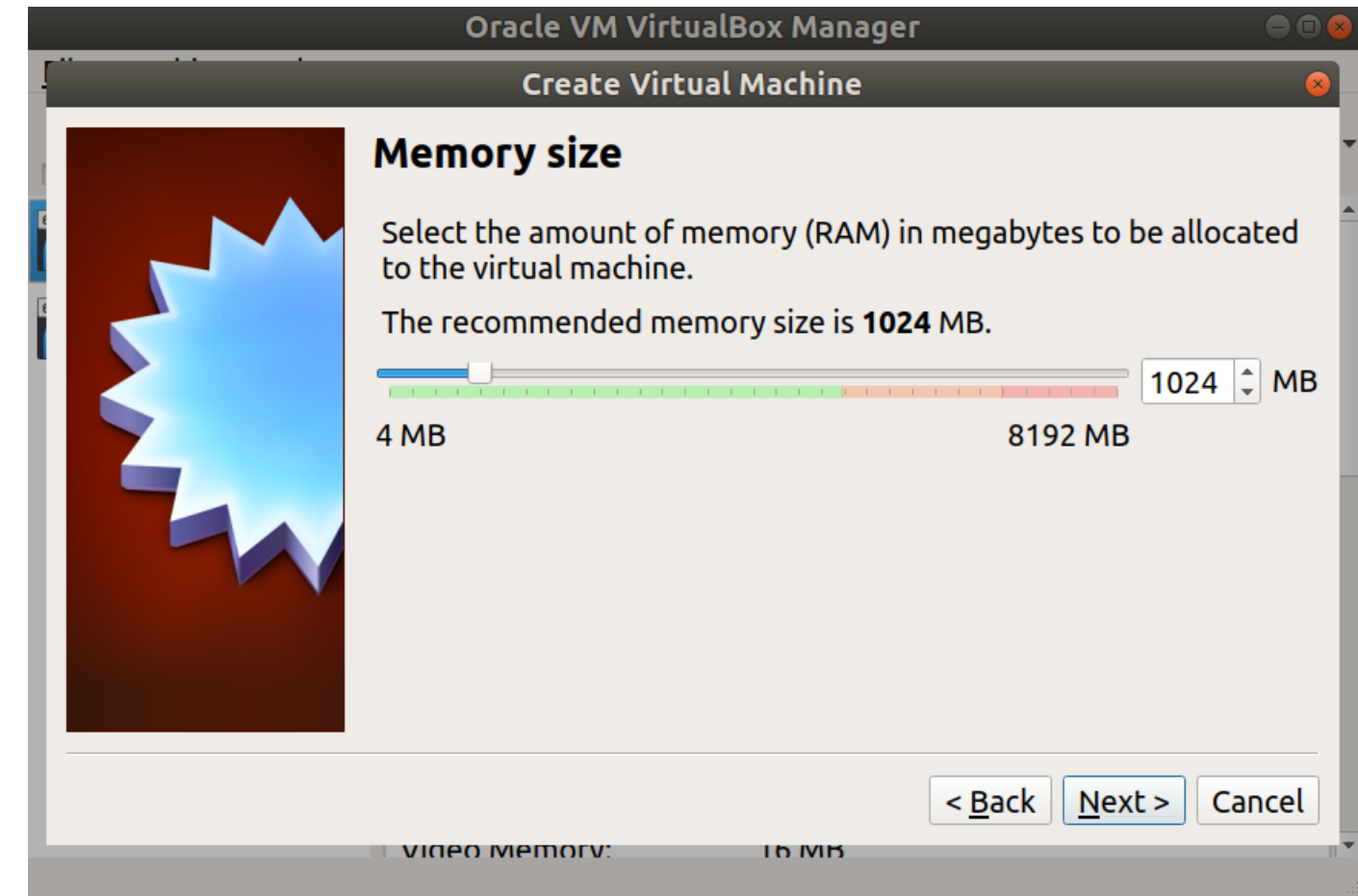
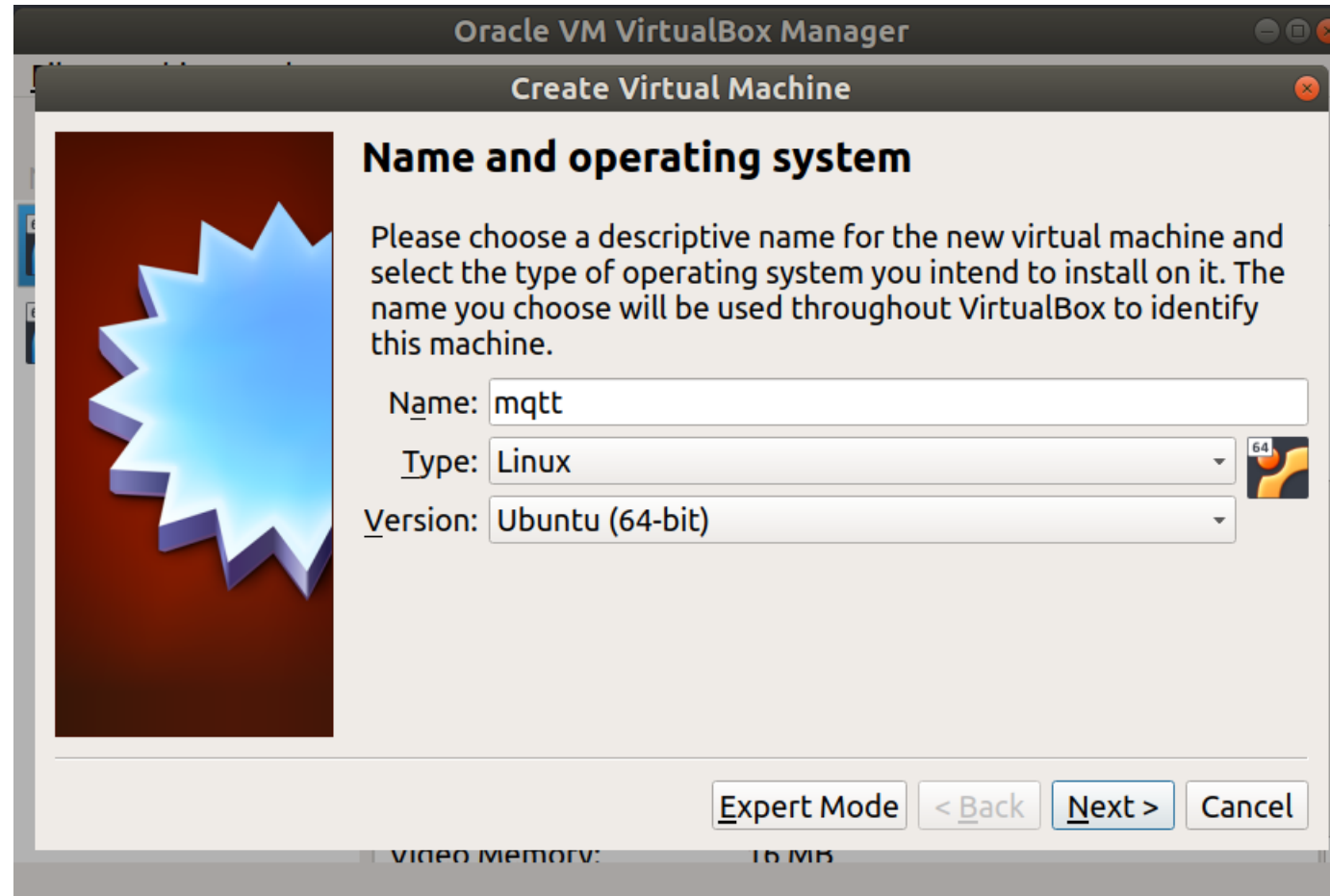
# Raspbian for Raspberry Pi

- 1) Download Raspbian from <https://www.raspberrypi.org/downloads/>
- 2) Use Balena Etcher (dd or another appropriate application) to flash the image on microSD card: <https://www.balena.io/etcher/>



## Using VirtualBox (1/4)

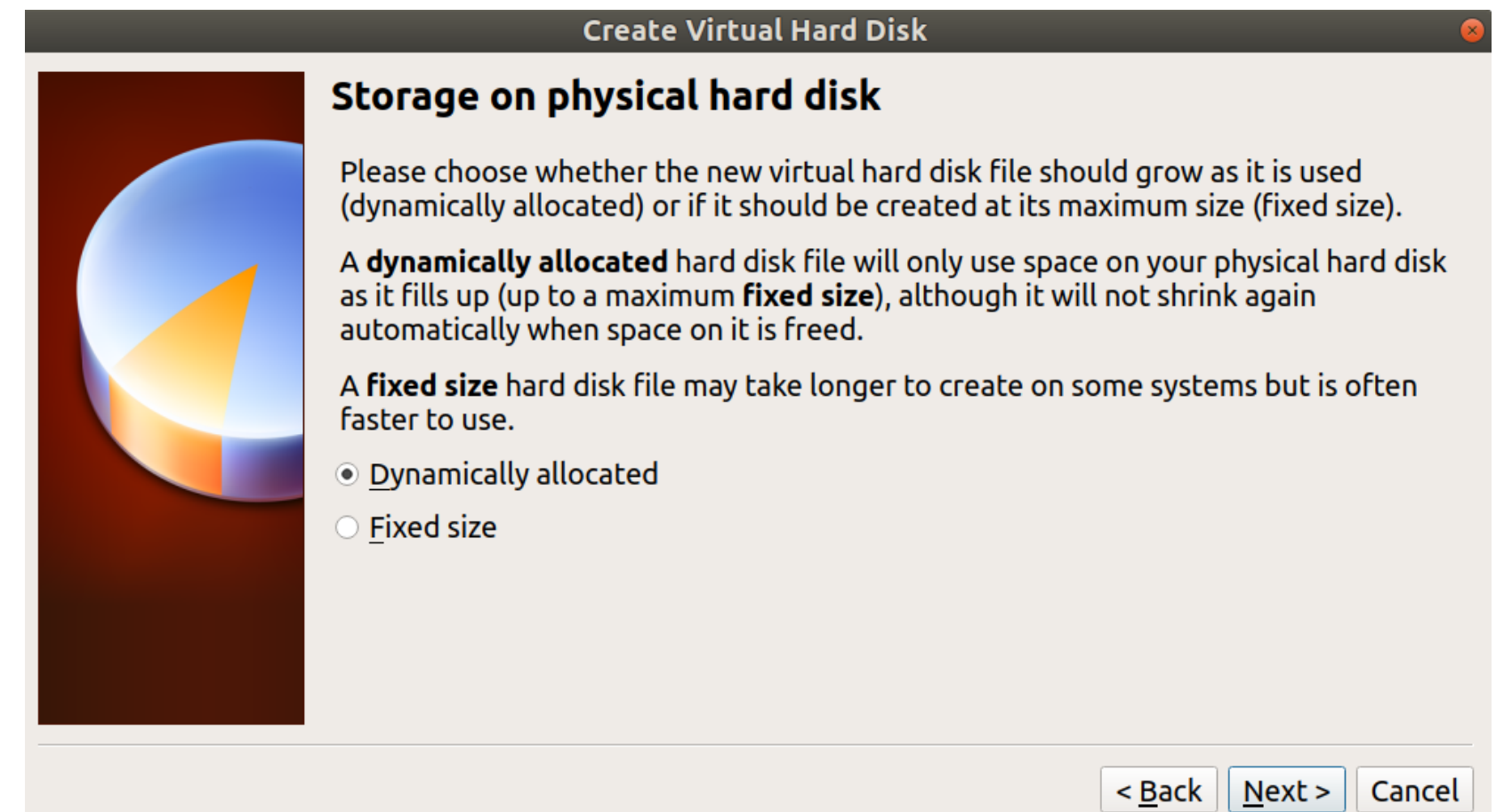
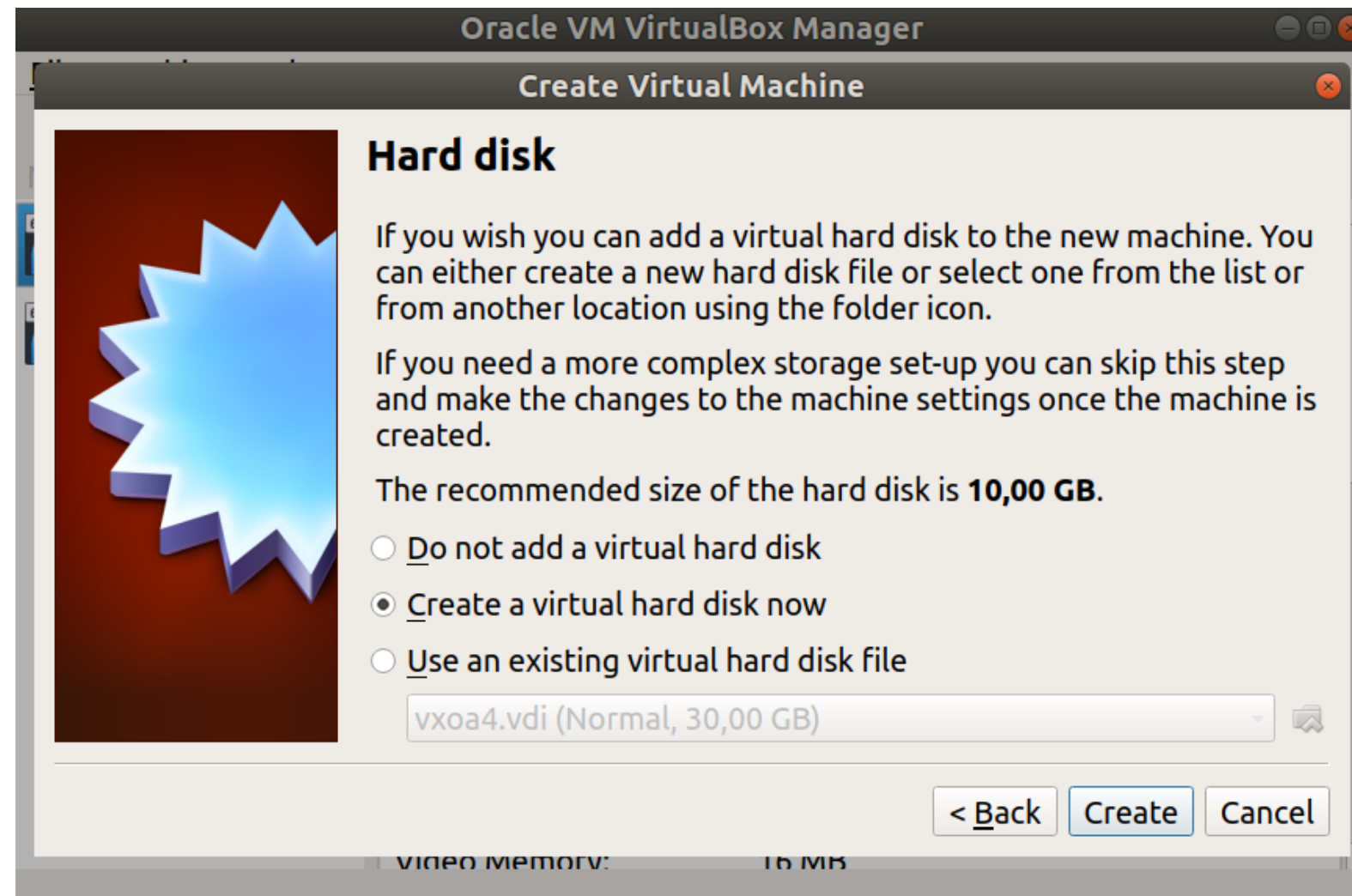
- 1) Download VirtualBox: <https://www.virtualbox.org/>
- 2) Download **Ubuntu Desktop 18.04 LTS**: <https://www.ubuntu.com/>
- 3) Launch VirtualBox and create new virtual machine with type **Linux** and version **Ubuntu (64-bit)**





## Using VirtualBox (2/4)

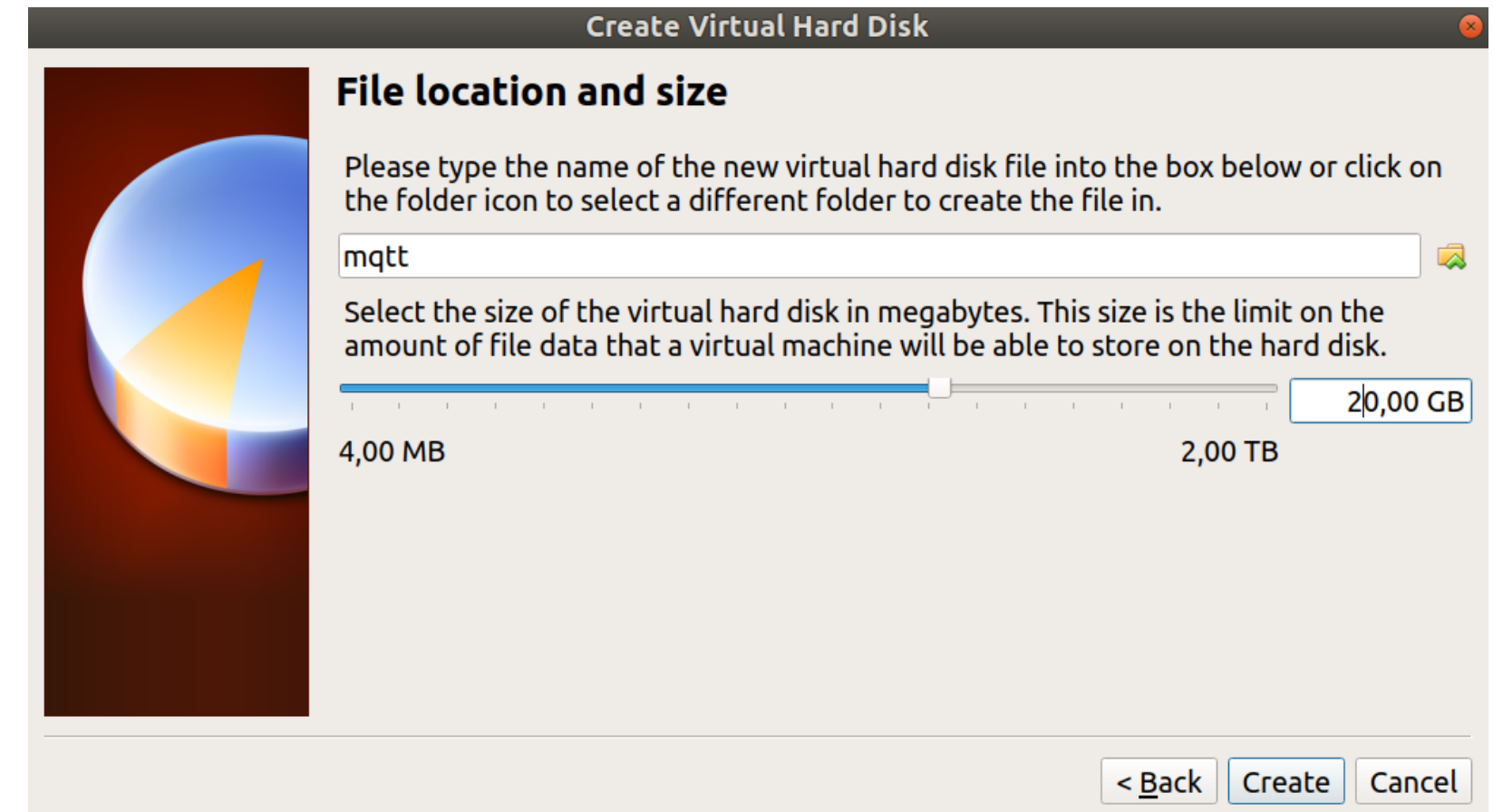
### 4) Create a new virtual disk with dynamic allocation



## Using VirtualBox (3/4)

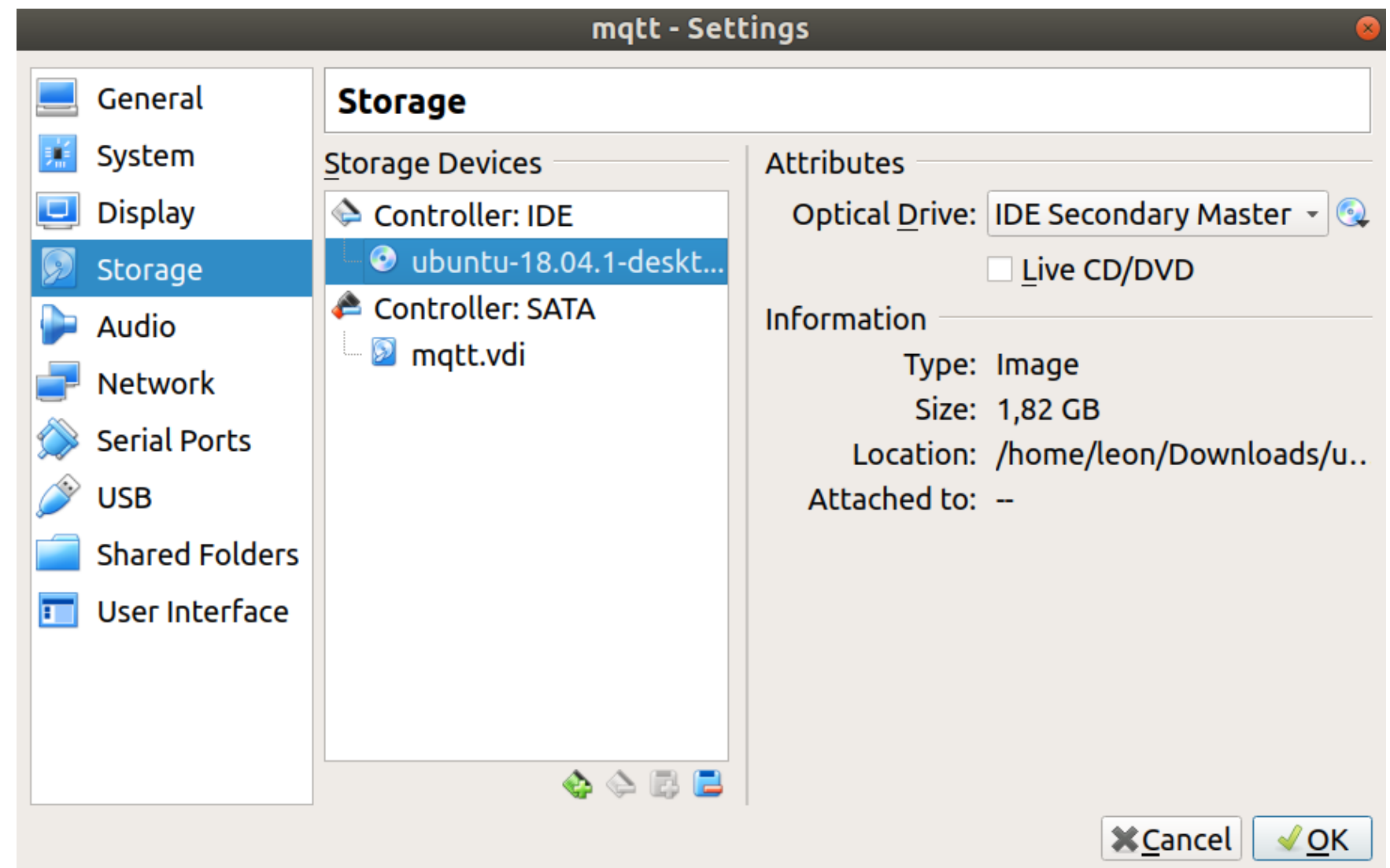
5) Select hard disk file type VDI

6) Set disk size



## Using VirtualBox (4/4)

- 7) Add the downloaded ISO for Ubuntu 18.04 LTS
- 8) Start the virtual machine in VirtualBox
- 9) Click **Install** and follow the onscreen instructions to finalize a minimal installation of Ubuntu on the virtual machine created in the previous steps
- 10) Restart the virtual machine after completing the installation



 **BREAK**

10min



# Section 2: Comparison of MQTT brokers

## In this section you will learn...



Comparison of open source MQTT brokers



Overview of commercial MQTT brokers



Recommendations for selecting the most appropriate MQTT broker for your project



## What is Open Source?

Software or hardware in which the author (and/or copyright holder) releases source code under a license in which users are granted the rights to study, change, and distribute the software to anyone and for any purpose



# Open Source MQTT Brokers

- Mosquitto
- HiveMQ (commercial license, open source plugins)
- Mosca
- emqttd
- VerneMQ
- ActiveMQ
- RabbitMQ





# Mosquitto

- Free and open source MQTT broker written in the C programming language
- Supports MQTT protocol version 3.1 and 3.1.1
- Supports QoS 0, 1 and 2
- Supports web sockets
- Available for Windows, FreeBSD, Mac OS and GNU/Linux distributions
- Also provides simple command line MQTT clients called mosquitto\_pub and mosquitto\_sub
- <https://mosquitto.org/>



# Mosquitto Development

- Created by Roger Light in 2010
- Project of [iot.eclipse.com](https://iot.eclipse.com) and using git since 2014
- Development has been sponsored by Cedalo AG since 2018
- Available at GitHub under dual license (Eclipse Public License 1.0 and the Eclipse Distribution License 1.0): <https://github.com/eclipse/mosquitto>
- 63 contributors, 1165 commits (as of end of 2018)
- Most of the commits are by the author (969)
- 26 releases, current stable v1.5.5



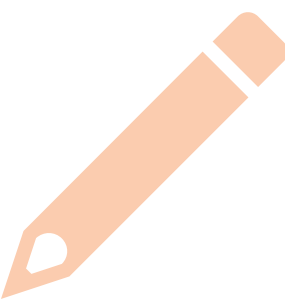
## Mosca

- Free and open source MQTT broker written in JavaScript
- Can be used standalone or embedded in another Node.js application
- Supports MQTT protocol version 3.1 and 3.1.1
- Supports web sockets
- Supports QoS 0 and 1
- Available for all platforms on which you can run Node.js: MS Windows, Mac OS and GNU/Linux distributions




## Mosca Development

- Available at GitHub under MIT license: <https://github.com/mcollina/mosca>
- Started by Matteo Collina in 2013
- 56 contributors, 973 commits (as of end of 2018)
- Most of the commits are by the author (551)
- 98 releases, current stable version 2.8.3 from 12 July 2018
- Above 2 thousands weekly downloads of the Node.js package from npmjs



# Mosca at npmjs.com



Search
[log in or sign up](#)

Be heard. Your voice will help us improve JavaScript. [Take the 2018 JavaScript Ecosystem Survey »](#)

mosca

2.8.3 • Public • Published 6 months ago

Readme

32 Dependencies

66 Dependents


121 Versions

Mosca

build

error


coverage 92%



install

weekly downloads
 

2,010



version

2.8.3

license

MIT



## EMQ (emqttd)

- Free and open source MQTT broker written in Erlang/OTP
- Supports MQTT protocol version 3.1, 3.1.1 and 5.0
- Supports QoS 0, 1 and 2
- Supports web sockets, MQTT-SN, CoAP, STOMP and SockJS
- Available for Windows, FreeBSD, Mac OS and GNU/Linux distributions



## EMQ Development

- Available at GitHub under Apache License 2.0:  
<https://github.com/emqx/emqx>
- Started by Feng Lee in 2012
- 38 contributors, 3571 commits (as of March 2019)
- Most of the commits are by the author
- 118 releases, current stable version 3.0.1 from 25 January 2019
- EMQ Enterprise provides commercial support and services for the open source EMQ project



## VerneMQ

- Free and open source MQTT broker written in Erlang/OTP
- Supports MQTT protocol version 3.1, 3.1.1 and 5.0
- Supports QoS 0, 1 and 2
- Supports websockets
- Available for GNU/Linux distributions and Mac OS
- Not working on Windows due to the LevelDB code





# VerneMQ Development

- Available at GitHub under Apache License 2.0:  
<https://github.com/erlio/vernemq>
- Started in 2016
- 21 contributors, 1850 commits (as of March 2019)
- Most of the commits are by Andre Graf and Lars Hesel Christensen
- 43 releases, current stable version 1.7.1 from 25 February 2019
- Octavo Labs AG (successor of Erlio GmbH) provides commercial support and services



# Apache ActiveMQ

- Free and open source message broker written in JAVA
- Supports large number of transport protocols: MQTT, OpenWire, STOMP, AMQP & others
- Supports MQTT protocol version 3.1 and 3.1.1
- Supports QoS 0, 1 and 2
- Supports websockets
- Available for GNU/Linux distributions, UNIX compatible systems and Windows



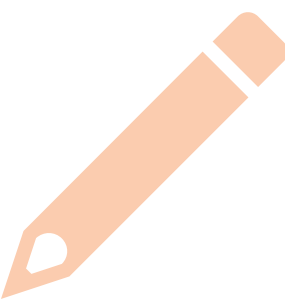
# Apache ActiveMQ Development

- Project of the Apache Software Foundation
- Available under Apache License 2.0:  
<https://git-wip-us.apache.org/repos/asf?p=activemq.git>
- Started in 2004 by LogicBlaze, donated to the Apache Software Foundation in 2007
- 104 contributors, 10070 commits (as of 9th October)
- 8 contributors with more than 500 commits
- 61 releases, current stable version 5.15.6 from 10 September 2018



# RabbitMQ

- Free and open source message broker written in Erlang
- Supports large number of transport protocols: MQTT, STOMP, AMQP, HTTP
- Supports MQTT protocol version 3.1 via a plugin
- Supports QoS 0 and 1
- Supports websockets
- Available for GNU/Linux distributions, BSD & UNIX compatible systems, Mac OS and MS Windows



# RabbitMQ Development

- Available at GitHub under Mozilla Public License 1.1:  
<https://github.com/rabbitmq/rabbitmq-server>
- Started as a joint venture between LShift and CohesiveFT in 2007, acquired SpringSource, a division of VMware in 2010, part of Pivotal Software since 2013
- 74 contributors, 17779 commits (as of March 2019)
- 3 contributors with more than 500 commits
- 244 releases, current stable version 3.7.14 from 29 March 2018



## More open source MQTT brokers...

- Apache ActiveMQ Artemis (written in JAVA, based on HornetQ)
- Moquette (written in JAVA, available in GitHub under Apache License 2.0)
- Vertx-mqtt-broker (written in JAVA with Vert.x, available in GitHub under Apache License 2.0)
- Wave (written in Erlang, available in GitHub under GNU Affero General Public License v3.0, latest release from June 2016)
- MQTTnet (written in C#, available in GitHub under MIT)



## (Some) Commercial MQTT brokers

- HiveMQ
- IBM IoT MessageSight
- Flespi
- JoramMQ
- PubSub+



# HiveMQ

- MQTT broker implement in the Java programming language
- Supports MQTT protocol version 3.1, 3.1.1 and 5.0
- Supports web sockets
- **Commercial license**, owned by dc-square GmbH
- Lead developer Dominik Obermaier
- Open source plugins available at GitHub under Apache-2.0:  
<https://github.com/HiveMQ>





# Public MQTT Brokers

. Server	. Broker	. Port	. Websocket
. <a href="https://iot.eclipse.org">iot.eclipse.org</a>	. Mosquitto	. 1883 / 8883	. 80 / 443
. <a href="https://test.mosquitto.org">test.mosquitto.org</a>	. Mosquitto	. 1883 / 8883 / 8884	. 8080 / 8081
. <a href="https://test.mosca.io">test.mosca.io</a>	. Mosca	. 1883	. 80
. <a href="https://broker.hivemq.com">broker.hivemq.com</a>	. HiveMQ	. 1883	. 8000
. <a href="https://broker.mqttdashboard.com">broker.mqttdashboard.com</a>	. HiveMQ	. 1883	. 8000
. <a href="https://cloudmqtt.com">cloudmqtt.com</a>	. Mosquitto	. 1xxxx	. 3xxxx



## Notes and conclusions

- There is a huge variety of high-quality free and open source MQTT brokers
- The business model for open source MQTT brokers is providing commercial support and services
- Open source MQTT brokers are highly dependent from their authors who remain leading developers up to date
- Most popular languages for implementing MQTT brokers are Erlang/OTP, JAVA and C
- All reviewed open source MQTT brokers run on GNU/Linux distributions



## In this section you learned...

- Which the popular open source MQTT brokers are
- Which the popular commercial MQTT brokers are
- How to select an appropriate MQTT broker for your project





# Lab 2: Configuring Mosquitto

## Scenario:

1. Enable web sockets in the open source MQTT broker Mosquitto.
2. Perform tests to verify that all key features of MQTT are working.

## Aim:

Enable web sockets and verify that Mosquitto is working fine.



## Required hardware

- Raspberry Pi 3 Model B or B+
- 5V/2.5A USB power supply
- Monitor with HDMI input
- HDMI cable
- USB keyboard
- USB mouse
- MicroSD card
- Personal computer (with Windows, Mac OS or a Linux distribution)



## Steps (1/2)

- 1) Create configurations to enable web sockets in Mosquitto in **/etc/mosquitto/conf.d/websocket.conf**
- 2) Add the following lines to the file:  
**listener 1883**  
**listener 1884**  
**protocol websockets**
- 3) Restart Mosquitto  
**sudo systemctl restart mosquitto**
- 4) Verify that Mosquitto is running  
**systemctl status -l mosquitto**
- 5) Subscribe for MQTT messages in a HTML5 web page, publish MQTT messages from the command line



## Steps (2/2)

5) Get the source code:

6) **git clone** <https://github.com/leon-anavi/mqtt-iot-examples.git>

7) Modify MQTT connection details in **js/demo.js**

8) Run the HTML5 page, for example with Node.js:

```
sudo apt-get update
```

```
sudo apt-get install -y nodejs npm
```

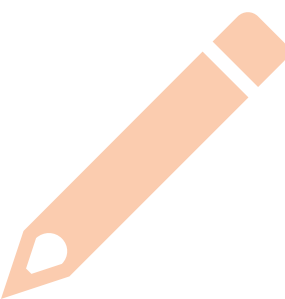
```
sudo npm install http-server -g
```

```
http-server
```

9) Open a web browser and load the HTML5 web page

10) Publish MQTT messages from the command line

```
mosquitto_pub -h 192.168.1.184 -d -p 1883 -t "home/room/temperature" -m  
'{ "temperature": 20 }'
```





 **BREAK**

10min

# Section 3: Integrating MQTT client library in your own project

## In this section you will learn...



Comparison of open source libraries for providing MQTT clients in various programming languages



Understanding how to use Node.js and MQTT for Internet of Things (IoT)

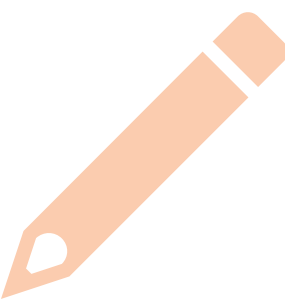


Benefits of using open source MQTT clients



## Popular Open Source libraries for MQTT Clients

- Paho offers MQTT client implementations for C/C++, Java, JavaScript, Python and C#  
<http://www.eclipse.org/paho/>
- Node.js library for implementing MQTT clients  
<https://www.npmjs.com/package/mqtt>
- Arduino client for MQTT:  
<http://pubsubclient.knolleary.net/>



## The Eclipse Paho project

- Free and open source client implementations of MQTT and MQTT-SN messaging protocols for popular programming languages
- Project of Eclipse foundation
- Source code available in various GitHub repositories:  
<https://github.com/eclipse>
- Supported programming languages: C, C++, Java, JavaScript, Python, Go, Rust, C# and an Android services
- Different subset of features is supported depending on the programming language



# Paho C Features

MQTT 3.1	✓
MQTT 3.1.1	✓
MQTT 5.0	✓
LWT	✓
SSL/TLS	✓
Message Persistence	✓
Automatic Reconnect	✓

Offline Buffering	✓
WebSocket Support	✓
Standard TCP Support	✓
Non-Blocking API	✓
Blocking API	✓
High Availability	✓



# Paho C++ Features

MQTT 3.1	✓
MQTT 3.1.1	✓
MQTT 5.0	✗
LWT	✓
SSL/TLS	✓
Message Persistence	✓
Automatic Reconnect	✓

Offline Buffering	✓
WebSocket Support	✓
Standard TCP Support	✓
Non-Blocking API	✓
Blocking API	✓
High Availability	✓



# Paho Java Features

MQTT 3.1	✓
MQTT 3.1.1	✓
MQTT 5.0	✗
LWT	✓
SSL/TLS	✓
Message Persistence	✓
Automatic Reconnect	✓

Offline Buffering	✓
WebSocket Support	✓
Standard TCP Support	✓
Non-Blocking API	✓
Blocking API	✓
High Availability	✓





# Paho JavaScript Features

MQTT 3.1	✓
MQTT 3.1.1	✓
MQTT 5.0	✗
LWT	✓
SSL/TLS	✓
Message Persistence	✓
Automatic Reconnect	✓

Offline Buffering	✗
WebSocket Support	✓
Standard TCP Support	✗
Non-Blocking API	✓
Blocking API	✗
High Availability	✓



# Paho Python Features

MQTT 3.1	✓
MQTT 3.1.1	✓
MQTT 5.0	✗
LWT	✓
SSL/TLS	✓
Message Persistence	✗
Automatic Reconnect	✓

Offline Buffering	✓
WebSocket Support	✓
Standard TCP Support	✓
Non-Blocking API	✓
Blocking API	✓
High Availability	✗



# Paho Rust Features

MQTT 3.1	✓
MQTT 3.1.1	✓
MQTT 5.0	✓
LWT	✓
SSL/TLS	✓
Message Persistence	✓
Automatic Reconnect	✓

Offline Buffering	✓
WebSocket Support	✓
Standard TCP Support	✓
Non-Blocking API	✓
Blocking API	✓
High Availability	✓



# Paho Go Features

MQTT 3.1	✓
MQTT 3.1.1	✓
MQTT 5.0	✗
LWT	✓
SSL/TLS	✓
Message Persistence	✓
Automatic Reconnect	✓

Offline Buffering	✓
WebSocket Support	✓
Standard TCP Support	✓
Non-Blocking API	✓
Blocking API	✗
High Availability	✓



## Arduino Client for MQTT

- MQTT client library for the Arduino and compatible devices
- Supports Arduino Ethernet, Arduino Ethernet Shield, Arduino YUN, Arduino WiFi Shield, Sparkfun WiFly Shield, TI CC3000 WiFi, Intel Galileo/Edison, ESP8266 and ESP32
- Available at GitHub under MIT license:  
<https://github.com/knolleary/pubsubclient>
- 162 commits, 18 releases, latest stable from 2 November 2018

<http://pubsubclient.knolleary.net/>



## MQTT.js

- Node.js client library for the MQTT protocol written in JavaScript
- Available at GitHub under MIT license: <https://github.com/mqttjs/MQTT.js>
- 129 releases, latest stable v2.18.8 from 30 August 2018
- 1283 commits by 118 contributors (as of the end of 2018)
- 3 maintainers: Adam Rudd, Matteo Collina and Maxime Agor



# MQTT.js at npmjs.com

mqtt

2.18.8 • Public • Published 4 months ago

Readme

14 Dependencies

981 Dependents

132 Versions



build passing

codecov 93%

npm

mqtt downloads (12 months)

Download stats from npm are currently unavailable

install

```
> npm i mqtt
```

↓ weekly downloads



version	license
2.18.8	MIT

open issues	pull requests
133	8

homepage	repository
github.com	github



## MQTT.js command line tools

- Installation:

```
npm install mqtt -g
```

- Connect and subscribe for a topic:

```
mqtt sub -t 'hello' -h 'test.mosquitto.org' -v
```

- Publish a message on the same topic:

```
mqtt pub -t 'hello' -h 'test.mosquitto.org' -m 'world'
```





## Using MQTT in Node.js (1/2)

- MQTT.js is a client library for the MQTT protocol available through **npm**:  
<https://www.npmjs.com/package/mqtt>
- Connect to MQTT broker:  
**var mqtt = require('mqtt');**  
**var client = mqtt.connect('mqtt://test.mosquitto.org');**
- Publish message:  
**client.publish('home/room', 'hello');**



## Using MQTT in Node.js (2/2)

- Subscribe for a topic:

```
client.on('connect', function () {  
  client.subscribe('home/#', function (err) {  
    //Do something...  
  })  
});
```

- Handle event **message** to process a received message:

```
client.on('message', function (topic, message) {  
  // Do something with the received message  
});
```



## Benefits of using open source libraries for MQTT clients

- Saves time and allows focus on the application development
- Flexibility to have ready to use libraries for different programming languages
- Relies on public resources, tutorials and examples by the community
- Bug fixes and improvements are welcomed in the upstream of the project



## In this section you learned...

- Which the popular open source MQTT client libraries are
- How to use MQTT and Node.js



# ? Questions?

# Lab 3:

# Implementing MQTT client using Node.js

## Scenario:

1. Install Node.js on Raspberry Pi.
2. Attach HTU21D temperature and humidity sensor to Raspberry Pi.
3. Write Node.js application to publish data from the sensor through MQTT.

## Aim:

Learn how to use MQTT in Node.js application



## Required hardware

- Raspberry Pi 3 Model B or B+
- 5V/2.5A USB power supply
- Monitor with HDMI input
- HDMI cable
- USB keyboard
- USB mouse
- MicroSD card
- Personal computer (with Windows, Mac OS or a Linux distribution)





## Steps

- 1) Install Node.js and npm:  
**sudo apt-get update**  
**sudo apt-get install -y nodejs npm**
- 2) Attach I2C HTU21D sensor module for measuring temperature and humidity (for Raspberry Pi users)
- 3) Create Node.js application to publish temperature via MQTT, **lab3-mqtt-nodejs** from <https://github.com/leon-anavi/mqtt-iot-examples>
- 4) Run the Node.js application
- 5) Verify that the application is running fine by subscribing for the same MQTT topic in command-line interface and heating the sensor

**NOTE:** VirtualBox users don't have the sensor and should use Node.js app that sends random data for testing purposes only.



# End of Day 1





## PRACTICAL MQTT FOR THE INTERNET OF THINGS (DAY 2)

# MQTT Course (Day 2)

1. 2 Days

2. 6 Sections

3. 6 Lab exercises

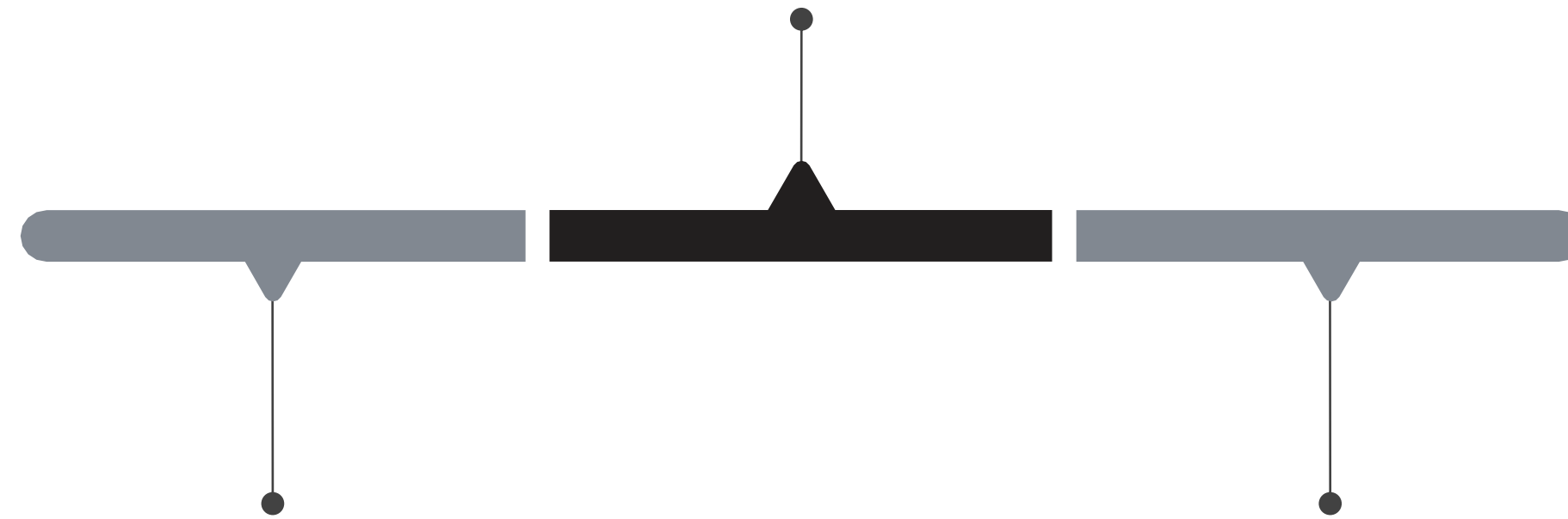
4. 180+ slides

5. Examples in  
GitHub



# Roadmap Day 2

Arduino & MQTT



Web sockets

Home Assistant





# Section 4: MQTT, HTML5 & Web Sockets

# In this section you will learn...



Introducing the Paho Project for JavaScript MQTT clients



Understanding how web sockets work in HTML5 and JavaScript



Publishing and receiving MQTT messages over web sockets



# What is a WebSocket?

- Allows front-end web applications to continuously transmit and receive data with the web back-end on a server
- Full-duplex communication channel over a single TCP connection
- Defined by a Web sockets specification in HTML5:  
<https://html.spec.whatwg.org/multipage/web-sockets.html>
- Supported by most web browsers: Google Chrome, Firefox, Safari, Opera, Microsoft Edge and Internet Explorer





# Why WebSocket is important for MQTT?

- Hypertext Transfer Protocol (HTTP) defines session as a sequence of network request-response transactions which is **not** convenient for real-time events
- MQTT provides (near) real-time communication, often based on real-life events such as change of temperature, humidity, detection of movement, power consumption, etc.
- WebSocket allows quick and convenient way to process quickly MQTT messages and to display them to in user-friendly way to the user through modern HTML5 technologies



# The Paho Project

- MQTT publish/subscribe client libraries for popular programming languages such as C, C++, Java, Python, Go, Rust, C# and JavaScript
- Free and open source available under Eclipse Distribution License 1.0 (BSD) and Eclipse Public License 1.0
- Actively developed by IBM and lots of individual contributors
- Project of [iot.eclipse.com](https://iot.eclipse.com)
- Part of other popular Eclipse projects such as Photon, Oxygen, Neon, and Luna
- <https://www.eclipse.org/paho/>



# Eclipse Paho JavaScript Client

- MQTT client library for easy integration in HTML5 web pages
- Written in JavaScript
- Uses Websockets to connect to an MQTT Broker
- Documentation: <http://www.eclipse.org/paho/files/jsdoc/index.html>



# Eclipse Paho JavaScript Client Features

MQTT 3.1	✓
MQTT 3.1.1	✓
LWT	✓
SSL / TLS	✓
Message Persistence	✓
Automatic Reconnect	✓

Offline Buffering	✗
WebSocket Support	✓
Standard TCP Support	✗
Non-Blocking API	✓
Blocking API	✗
High Availability	✓



# Eclipse Paho JavaScript Client CDNs

- Include the following line to use the plain library:  
**`<script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.js" type="text/javascript"></script>`**
- Include the following line to use the minified library:  
**`<script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.min.js" type="text/javascript"></script>`**



# Class Message Properties

<b>payloadString</b>	(string)	The payload as text with valid UTF-8 characters
<b>payloadBytes</b>	(ArrayBuffer)	The payload as an ArrayBuffer
<b>destinationName</b>	(string)	Mandatory property with the topic of the MQTT message
<b>qos</b>	(number)	Represents MQTT QoS, must be 0, 1 or 2, the default value is 0
<b>retained</b>	(Boolean)	Specified if the message has to be retained by the MQTT broker, if true it will be delivered to any new subscriber for this topic
<b>duplicate</b>	(Boolean)	True if the message might have been already received, only set on messages received from the server



# Using WebSocket for MQTT in Web Page (1/3)

- Create MQTT client:  
**client = new Paho.MQTT.Client("iot.eclipse.org", 443, "myMqttClient");**
- Assign callback functions for processing received MQTT messages:  
**client.onMessageArrived = onMessageArrived;**
- Connect to MQTT broker:  
**client.connect({onSuccess:onConnect});**



## Using WebSocket for MQTT in Web Page (2/3)

- Subscribe to MQTT topic after successfully establishing a connection with the MQTT broker:

```
function onConnect() {  
    client.subscribe("home/#");  
}
```

- Process received messages in the callback function assigned previously:

```
function onMessageArrived(message){  
    //Do something with the message  
}
```





## Using WebSocket for MQTT in Web Page (3/3)

- Publish MQTT messages using the JavaScript library from the Paho project:  
**message = new Paho.MQTT.Message("Hello");**  
**message.destinationName = "World";**  
**client.send(message);**




# In this section you learned...

- What the Paho projects is?
- How to use the Paho project for MQTT communication directly in HTML5 web pages with web sockets



# 🔍 Questions?



# Lab 4: Creating HTML5 web page with the Paho Project

## Scenario:

Creating HTML5 web page for showing temperature received through MQTT via web socket.

## Aim:

Learn how to use MQTT in HTML5 web page through web sockets with JavaScript and the open source Paho project.



# Required hardware

- Raspberry Pi 3 Model B or B+
- 5V/2.5A USB power supply
- Monitor with HDMI input
- HDMI cable
- USB keyboard
- USB mouse
- MicroSD card
- Personal computer (with Windows, Mac OS or a Linux distribution)



# Steps

- 1) Create simple HTML5 web page
- 2) Include the JavaScript version of the Paho project
- 3) Write JavaScript source code to connect to MQTT broker, subscribe to a certain topic and display the received messages in the HTML5 web page
- 4) Test using the Node.js application from the previous example



 **BREAK**

10min





# Section 5: Arduino & MQTT

## In this section you will learn...



Introducing the open source hardware movement



Exploring Arduino



Getting started with Arduino IDE



Writing Arduino sketches



Exploring pubsubclient, the Arduino library that provides support for MQTT



# Open Source Hardware

- OSHWA defines Open Source Hardware as follows:

“Open Source Hardware (OSHW) is a term for tangible artifacts — machines, devices, or other physical things — whose design has been released to the public in such a way that anyone can make, modify, distribute, and use those things.”

Taken from: <https://www.oshwa.org/definition/> [April 2019]

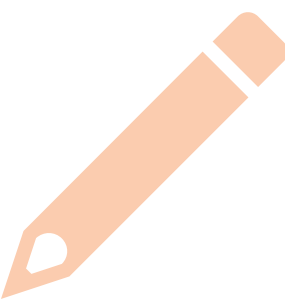
- The Open Source Hardware Association (OSHWA) maintains the Open Source Hardware certification

<https://www.oshwa.org/>



# Arduino

- “Arduino is an open-source electronics platform based on easy-to-use hardware and software”  
(<https://www.arduino.cc/en/Guide/Introduction>)
- Available commercially in preassembled form or as do-it-yourself (DIY) dev boards and kits
- Support for add-on boards called **Arduino Shields**
- Arduino IDE and sketch (name that Arduino uses for a program)



# Arduino Brief History

- Created at the Ivrea Interaction Design Institute in Italy
- Named after a bar in Ivrea, Italy which was named after Arduin of Ivrea
- Registered as a company in 2008 by 5 co-founders: Massimo Banzi, David Cuartielles, Tom Igoe, David Mellis and Gianluca Martino
- After trademark dispute from 2015 until 2017, BCMI, founded by Massimo Banzi, David Cuartielles, David Mellis and Tom Igoe, acquired Arduino AG and all the Arduino trademarks





# Arduino UNO rev3

- Microcontroller: ATmega328P
- Analog Input Pins: 6
- Digital I/O Pins: 14
- PWM Channels: 6
- Operating voltage: 5V
- Input voltage: 6-20V (7-12V recommended)
- <https://store.arduino.cc/usa/arduino-uno-rev3>





# Arduino UNO WiFi

- Microcontroller: ATMEGA4809
- Analog Input Pins: 6
- Digital I/O Pins: 14
- PWM Channels: 5
- Operating voltage: 5V
- Input voltage: 7-12V recommended
- <https://store.arduino.cc/usa/arduino-uno-wifi-rev2>





# Arduino Leonardo

- Microcontroller: ATmega32u4
- Analog Input Pins: 12
- Digital I/O Pins: 20
- PWM Channels: 7
- Operating voltage: 5V
- Input voltage: 6-20V (7-12V recommended)
- <https://store.arduino.cc/usa/arduino-leonardo-with-headers>





# Arduino Compatible Products

- Hardware designed and manufactured by other vendors that is compatible with Arduino
- Hardware derived under the requirements of the open source hardware license of the original Arduino
- Compatible products manufactured by other companies, cannot contain the name Arduino because it is a trademark:

<https://www.arduino.cc/en/Trademark/HomePage>



# Beware of Fake Arduino Clones

- Over the years there have been several frauds with crowd funding campaign promising identical Arduino clones at dirt cheap prices that never delivered the orders of the backers
- Beware of cheap counterfeit parts, for more information:  
<https://www.sparkfun.com/news/350>



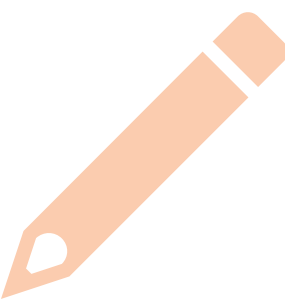
# ESP8266 and ESP32

- ESP8266 is a low-power microcontroller with WiFi
- ESP32 is a low-power microcontroller with WiFi and Bluetooth
- Compatible with Arduino IDE and sketches
- Tensilica Xtensa CPU
- Designed and produced by the Chinese company Espressif Systems



# Arduino Cores for ESP32 & ESP8266

- Arduino core is the APIs to build and install Arduino sketches
- Arduino cores for ESP32 and ESP8266 provide compatibility with Arduino IDE, libraries and sketches
- For ESP8266: <https://github.com/esp8266/Arduino>
- For ESP32: <https://github.com/espressif/arduino-esp32>



# Arduino IDE

- Integrated development environment (IDE)
- Available for MS Windows, macOS and GNU/Linux distributions
- Also available as Arduino Web Editor:  
<https://create.arduino.cc/editor>



# Installing Arduino IDE

- Download latest stable version of Arduino:  
<https://www.arduino.cc/en/Main/Software>



## ARDUINO 1.8.8

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

**Windows** Installer, for Windows XP and up  
**Windows** ZIP file for non admin install

**Windows app** Requires Win 8.1 or 10



**Mac OS X** 10.8 Mountain Lion or newer

**Linux** 32 bits

**Linux** 64 bits

**Linux** ARM

[Release Notes](#)  
[Source Code](#)  
[Checksums \(sha512\)](#)



# Arduino Sketch

- Arduino programs written in C/C++ programming languages using Arduino IDE
- Extension **.ino**



# Minimal Arduino sketch

- Each Arduino sketch has at least two functions:
- **setup()**
- **loop()**





# Arduino Libraries

- Extend the environment and provide extra functionality for the sketch
- A set of libraries are included in Arduino IDE by default
- Developers can make and publish new libraries
- Existing libraries can be retrieved and installed through the **Library Manager** in Arduino IDE
- Library can be added as a ZIP from **Sketch > Import Library**



# Arduino Client for MQTT

- **pubsubclient** - Arduino library that provides MQTT publish and subscribe functionality
- Created by Nick O'Leary
- Open source, available at GitHub under the MIT license
- <https://github.com/knolleary/pubsubclient>



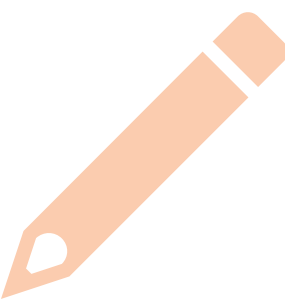
# Pubsubclient Compatible Hardware

- Arduino Ethernet and Arduino Ethernet Shield
- Arduino YUN
- Arduino WiFi Shield
- Sparkfun WiFly Shield – library
- TI CC3000 WiFi - library
- Intel Galileo/Edison
- ESP8266
- ESP32



# PubSubClient Constructors

- PubSubClient ()
- PubSubClient (client)
- PubSubClient(server, port, [callback], client, [stream])



# PubSubClient Functions

- boolean **connect** (clientId)
- boolean **connect** (clientId, willTopic, willQoS, willRetain, willMessage)
- boolean **connect** (clientId, username, password)
- boolean **connect** (clientId, username, password, willTopic, willQoS, willRetain, willMessage)
- boolean **connect** (clientId, username, password, willTopic, willQoS, WillRetain, willMessage, cleanSession)
- void **disconnect** ()
- int **publish** (topic, payload)
- int **publish** (topic, payload, retained)
- int **publish** (topic, payload, length)
- int **publish** (topic, payload, length, retained)
- int **publish\_P** (topic, payload, length, retained)



# PubSubClient Functions

- boolean **beginPublish** (topic, payloadLength, retained)
- size\_t **write** (uint8\_t)
- size\_t **write** (payload, length)
- boolean **endPublish** ()
- boolean **subscribe** (topic, [qos])
- boolean **unsubscribe** (topic)
- boolean **loop** ()
- int **connected** ()
- int **state** ()
- PubSubClient **setServer** (server, port)
- PubSubClient **setCallback** (callback)
- PubSubClient **setClient** (client)
- PubSubClient **setStream** (stream)



# Using PubSubClient (1/2)

- Include header files and initialize the MQTT client

```
#include <WiFi.h>
```

```
#include <PubSubClient.h>
```

```
WiFiClient espClient;
```

```
PubSubClient mqttClient(espClient);
```

- Callback for received messages

```
void mqttCallback(char* topic, byte* payload, unsigned int length) { ... }
```

- Register the callback

```
mqttClient.setServer(mqtt_server, mqttPort); mqttClient.setCallback(mqttCallback);
```



## Using PubSubClient (2/2)

- Establish a connection with the MQTT broker  
`const String clientId = "my-mqtt-device";`  
`mqttClient.connect(clientId.c_str(), username, password)`
- Subscribe for MQTT topic  
`mqttClient.subscribe(cmnd_power_topic);`
- Publish MQTT message  
`mqttClient.publish(stat_color_topic, payload, true);`





# ANAVI Light Controller

- Certified open source hardware WiFi device for controlling a 12 V RGB LED strip
- Supports sensors for light, temperature, humidity, and gesture recognition
- With ESP8266 microcontroller
- Arduino sketches:  
<https://github.com/AnaviTechnology/anavi-light-controller-sw>



# ANAVI Light Controller



Available at Crowd Supply:

<https://www.crowdsupply.com/anavi-technology/light-controller>



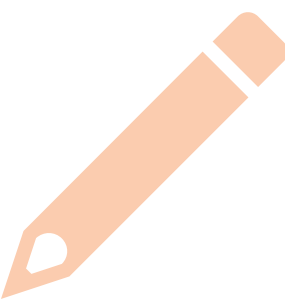
# ANAVI Light Controller (Hardware)

- Certified open source hardware **BG000005**:  
<https://certification.oshwa.org/bg000005.html>
- Designed with free and open source software only: **KiCad** (for the PCB) and **OpenSCAD** (for the acrylic enclosure)
- Schematics and project files:  
<https://github.com/AnaviTechnology/anavi-light-controller/>
- KiCad: <http://kicad-pcb.org/>
- OpenSCAD: <http://www.openscad.org/>



# In this section you learned...


- What open source hardware is and why is it important
- What Arduino is
- How to use Arduino IDE
- How to use the pubsubclient Arduino library for MQTT



# ? Questions?

# Lab 5:

## Creating Arduino sketch

 for publishing  
temperature and  
humidity through MQTT

## Scenario:

1. Setup ANAVI Light Controller and connect it to a computer via USB to UART debug cable.
2. Install and configure Arduino IDE.
3. Write a simple Arduino sketch and flash in on the device.

## Aim:

Learn how to use MQTT for Arduino-compatible devices and flash custom firmware through Arduino IDE



# Required hardware

- Personal computer (with Windows, Mac OS or a Linux distribution)
- ANAVI Light Controller
- 12V RGB LED strip
- 12V power supply
- USB to UART debug cable





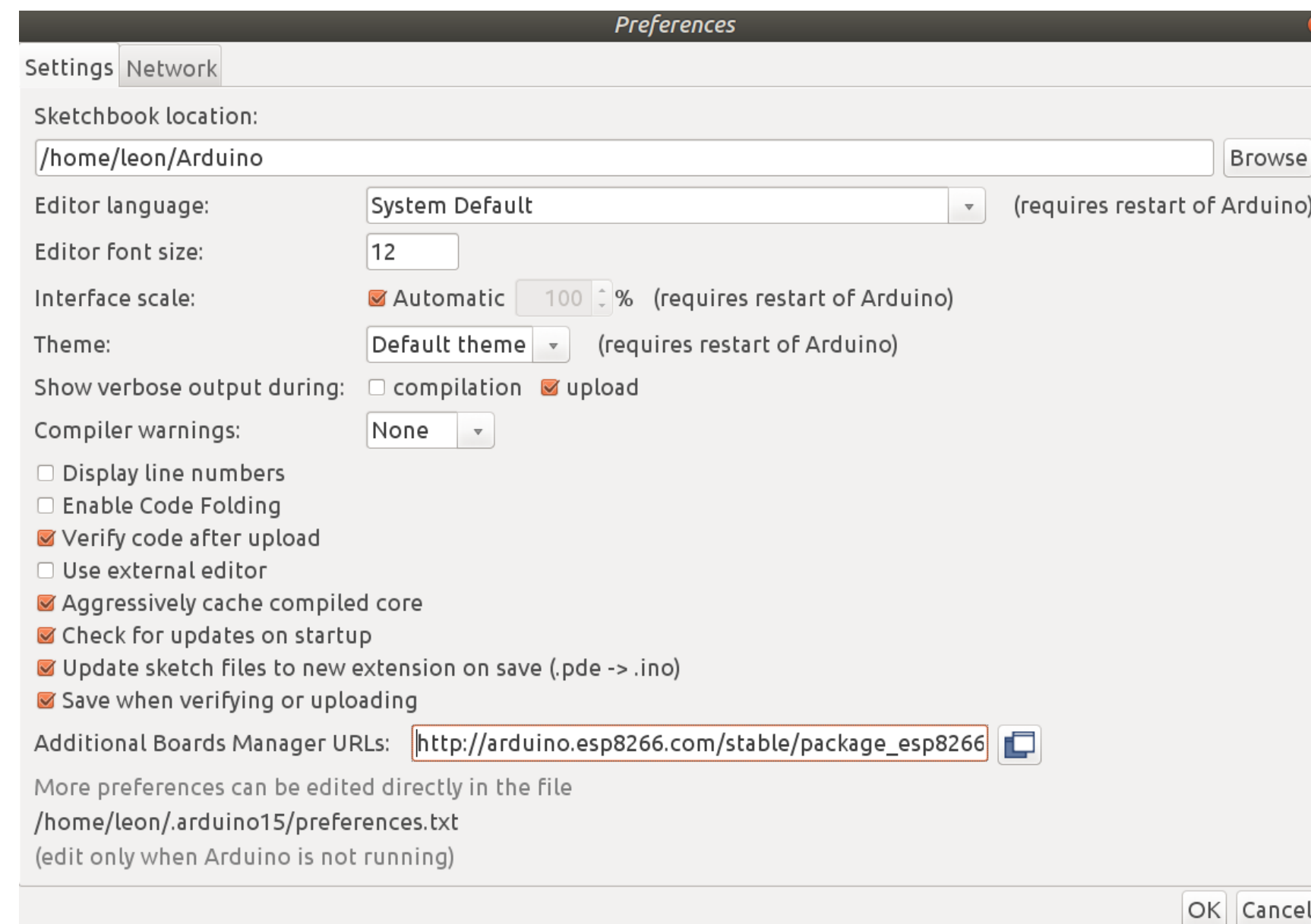
# Steps

- 1) Install Arduino IDE
- 2) Configure Arduino IDE for ESP8266 and install libraries
- 3) Attach 12V RGB LED strip to ANAVI Light Controller
- 4) Connect ANAVI Light Controller to PC with USB to UART debug cable
- 5) Plug appropriate 12V power supply in ANAVI Light Controller to turn it on
- 6) Write simple Arduino sketch for a blinking LED and upload it to ANAVI Light Controller
- 7) Compile and upload again the default open source firmware on ANAVI Light Controller:  
<https://github.com/AnaviTechnology/anavi-light-controller-sw>



# Add ESP8266 Board Manager

- Open Arduino IDE and go to **File > Preferences**
- Configure ESP8266 at Additional Board Managers URLs:  
[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)



# Configure Arduino IDE for ESP8266

- Go to **Tools > Board**
- Select **Generic ESP8266 Module**
- Select **Upload speed: 115200**
- Select **Flash size: 4M (1M SPIFFS)**
- Select an appropriate port



 **BREAK**

10min



# Section 6: Home Assistant

## In this section you will learn...



Overview of popular open source platforms for home automation



Introducing Home Assistant



Understanding how Home Assistant works



Integrating custom devices in Home Assistant using various components



# Popular Open Source Platforms for Home Automation

- Home Assistant
- OpenHAB
- Domoticz
- Calaos
- MisterHouse
- Many more...



# OpenHAB

- Developed in JAVA using the open source framework Eclipse SmartHome and Eclipse Jetty for a web server
- Available for MS Windows, Mac OS, Docker and GNU/Linux distributions, including images for Raspberry Pi and PINE A64
- Two major version with guidelines how to migrate from OpenHAB 1.x to 2: <https://www.openhab.org/docs/configuration/migration/>
- <https://www.openhab.org/>





# OpenHAB Development

- Available at GitHub: <https://github.com/openhab>
- Started by Kai Kreuzer in 2010
- Core team of 18 maintainers and more than 50 contributors
- Openhab-distro has 22 releases and it is available under Eclipse Public License 1.0



# Domoticz

- Developed in C++ with HTML5 scalable front-end
- Available for MS Windows, Mac OS and GNU/Linux distributions
- 32-bit and 64-bit packages for ARM devices such as Raspberry Pi, Odroid, Cubieboard, etc.
- <http://www.domoticz.com>



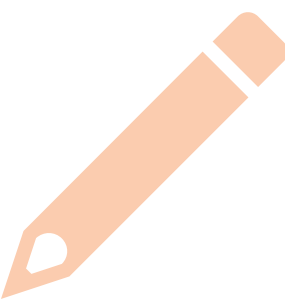
# Domoticz Development

- Available at GitHub: <https://github.com/domoticz/>
- Started by “Gizmocuz” in 2012
- More than 230 contributors
- Has 7 releases and it is available under GNU General Public License v3.0 (GPLv3)



# Home Assistant

- Open-source home automation platform running on Python 3
- Perfect to run on a Raspberry Pi
- More than 1200 components for integration with popular Internet of Things such as IKEA Trådfri, Philips Hue, Google Assistant, Alexa / Amazon Echo, Nest, KODI, etc.
- <https://home-assistant.io/>



# Brief history of Home Assistant

- Source code available at GitHub under Apache 2.0 license
- Started in 2013 by Paulus Schoutsen
- Huge community, more than 830 contributors



# Home Assistant Components

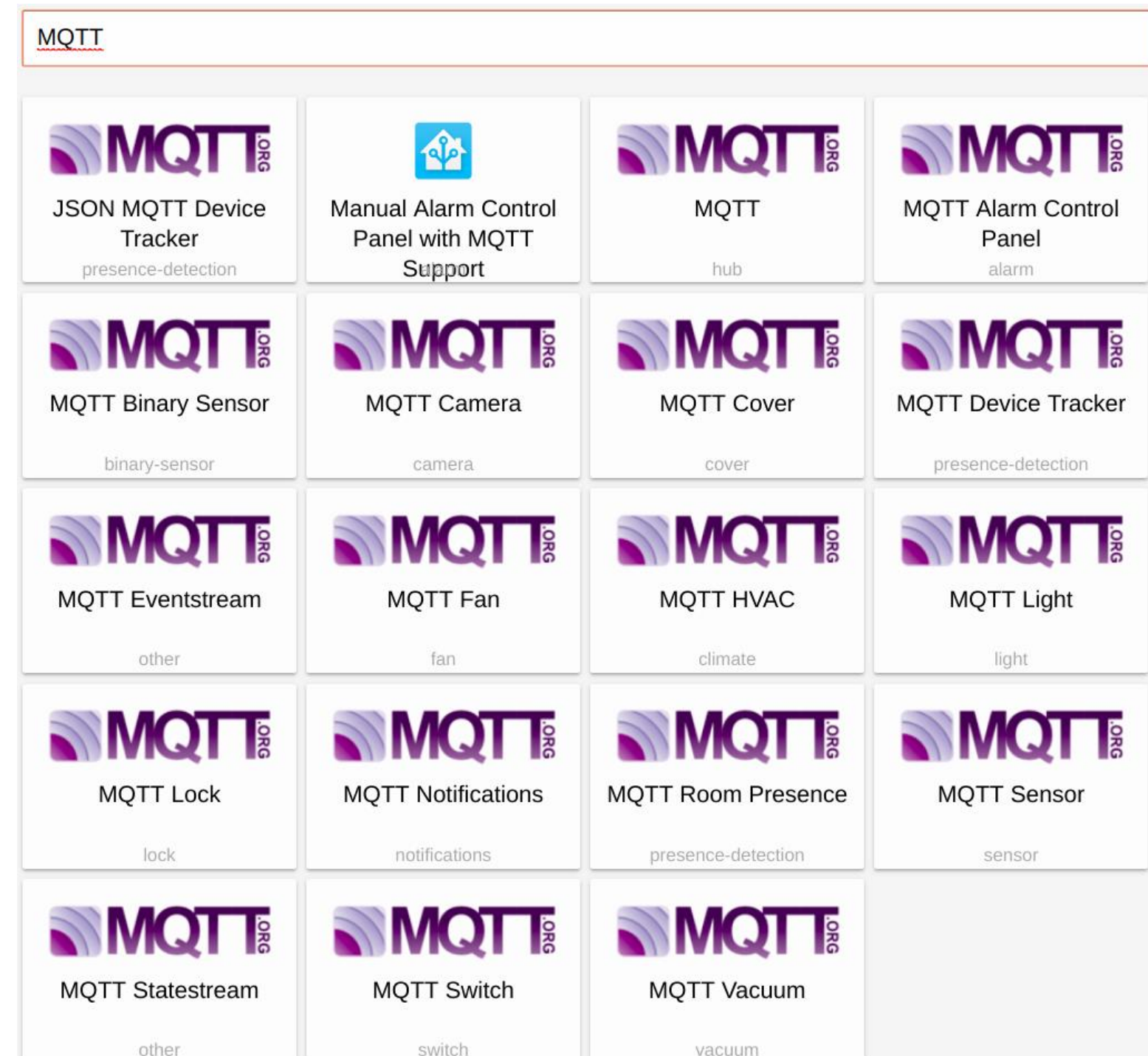
- More than 1200 components!

Alexa / Amazon Echo	Arduino	Belkin WeMo	Dark Sky
ecobee	Google Assistant	Google Cast	IFTTT
IKEA Tradfri (Tradfri)	Kodi	Minut Point	MQTT
MySensors	Nest	Philips Hue	Plex



# Home Assistant MQTT Components

- MQTT
- MQTT Light
- MQTT Sensor
- MQTT Switch
- More...



# MQTT Light Component

- Home Assistant component for controlling a MQTT-enabled light that can receive JSON messages
- [https://home-assistant.io/components/light.mqtt\\_json/](https://home-assistant.io/components/light.mqtt_json/)





# MQTT Sensor

- Home Assistant component for receiving data from MQTT-enabled sensor and saving a history graph
- Supports JSON in the payload of the MQTT messages
- Optionally, more flexible templating for the payload of the MQTT message is available
- <https://www.home-assistant.io/components/sensor.mqtt/>



# Home Assistant on Raspberry Pi

A couple of popular options for getting started:

## Hass.io

- An operating system based on BalenaOS (previously known as ResinOS) and Docker for running Home Assistant. Started by Pascal Vizeli in 2017. Compatible with Raspberry Pi, Intel NUC, OrangePi Prime, Odroid C2 or generic Linux servers.

## Hassbian

- GNU/Linux distribution for Raspberry Pi with Home Assistant based on Raspbian that uses the same repositories



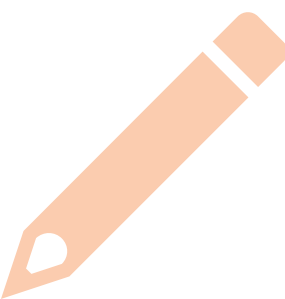
# Home Assistant Installation Notes

## Hass.io

- <https://www.home-assistant.io/hassio/installation/>

## Hassbian

- <https://www.homeassistant.io/docs/installation/hassbian/installation/>



# Home Assistant URL

After successful installation Home Assistant will be available at the following URL in your local area network:

## Hass.io

- <http://hassio.local:8123>

## Hassbian

- <http://hassbian.local:8123>

default username: **pi**

default password: **raspberry**



# Home Assistant Configurations

- Configuration file **configuration.yaml**
- **YAML** (YAML Ain't Markup Language) is a human readable descriptive programming language for data serialization
- The path to configuration.yaml may vary depending on the installation of Home Assistant, it should be visible at the about page in the web interface
- Alternatively, you can edit **configuration.yaml** through HASS Configurator add-on
- 



# In this section you learned...

- Which the popular open source software platforms for home automation are
- What Home Assistant is
- How to integrate IoT in Home Assistant using MQTT components



# ? Questions?

# Lab 6:

# Installing Home Assistant on Raspberry Pi



## Scenario:

1. Install Home Assistant and MQTT on Raspberry Pi
2. Configure the Arduino-compatible device from the previous lab exercise in Home Assistant

## Aim:

Getting started with the popular open source software platform Home Assistant on Raspberry Pi and integrating custom devices using MQTT



# Required hardware

- Raspberry Pi 3 Model B or B+
- 5V/2.5A USB power supply
- Monitor with HDMI input
- HDMI cable
- USB keyboard
- USB mouse
- MicroSD card
- Personal computer (with Windows, Mac OS or a Linux distribution)



# Steps

- 1) Download and install Home Assistant on Raspberry Pi
- 2) Install Mosquitto and enable web sockets (as in lab exercises 2 and 3)
- 3) Add MQTT JSON Light component to configuration.yaml
- 4) Configure ANAVI Light Controller
- 5) Change the lights of ANAVI Light Controller through the web interface of Home Assistant



# End of Day 2



# In this session you learned...

- Working of MQTT along with Installing Raspbian and Mosquitto on Raspberry Pi 3
- Open Source and Commercial MQTT brokers
- Understanding how to use Node.js and MQTT for Internet of Things (IoT)
- Introduction to Paho project and publishing and receiving MQTT messages over web sockets
- Arduino IDE and library that provides support for MQTT
- Home assistant for home automation



# Further reading

- **Raspberry Pi Zero W Wireless Projects**  
<https://www.packtpub.com/hardware-and-creative/raspberry-pi-zero-w-wireless-projects>
- **Internet of Things with Arduino Cookbook, Marco Schwartz**  
<https://www.packtpub.com/hardware-and-creative/internet-things-arduino-cookbook>
- **Node.js By Example, Krasimir Tsonev**  
<https://www.packtpub.com/application-development/nodejs-example>
- **ESP8266 Home Automation Projects**  
<https://www.packtpub.com/hardware-and-creative/esp8266-home-automation-projects>
- **Raspberry Pi 3 Home Automation Projects**  
• <https://www.packtpub.com/hardware-and-creative/raspberry-pi-3-home-automation-projects>
- 



# THANK YOU!

