



Feb 18<sup>th</sup> 2019

# Mentors

Assem Hasna

asem.hasna@gmail.com

Coding is my life

Since 2013

Former Cisco Intern

2016 - 2017

Cloud Solutions Developer

Since 2017

# Specs

A 1.2GHz 64-bit quad-core ARMv8 CPU

802.11n Wireless LAN

Bluetooth 4.1

1GB RAM

4 USB ports

40 GPIO pins

3.5mm audio jack & composite video

Full HDMI port

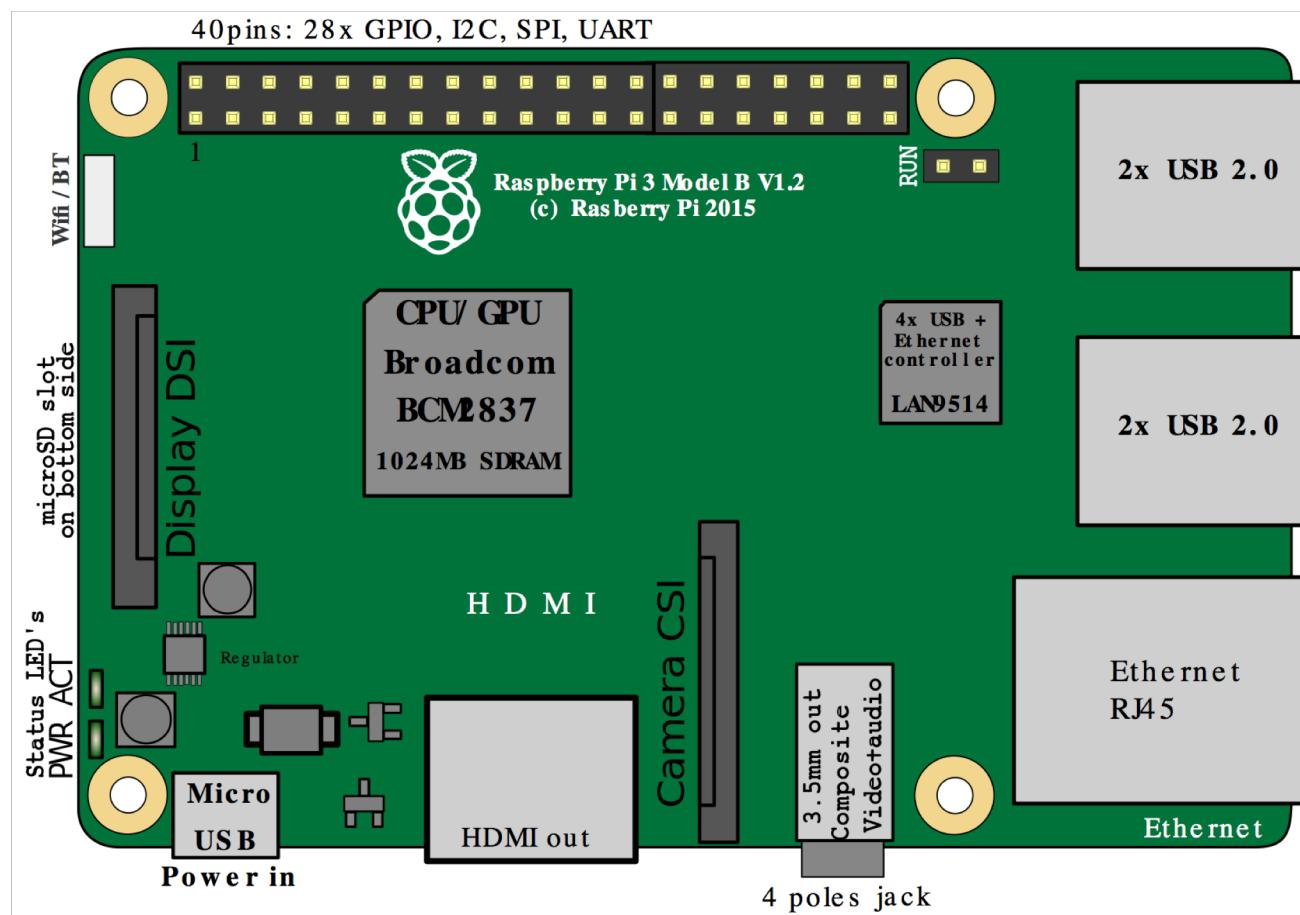
Ethernet port

Camera interface (CSI)

Display interface (DSI)

Micro SD card slot

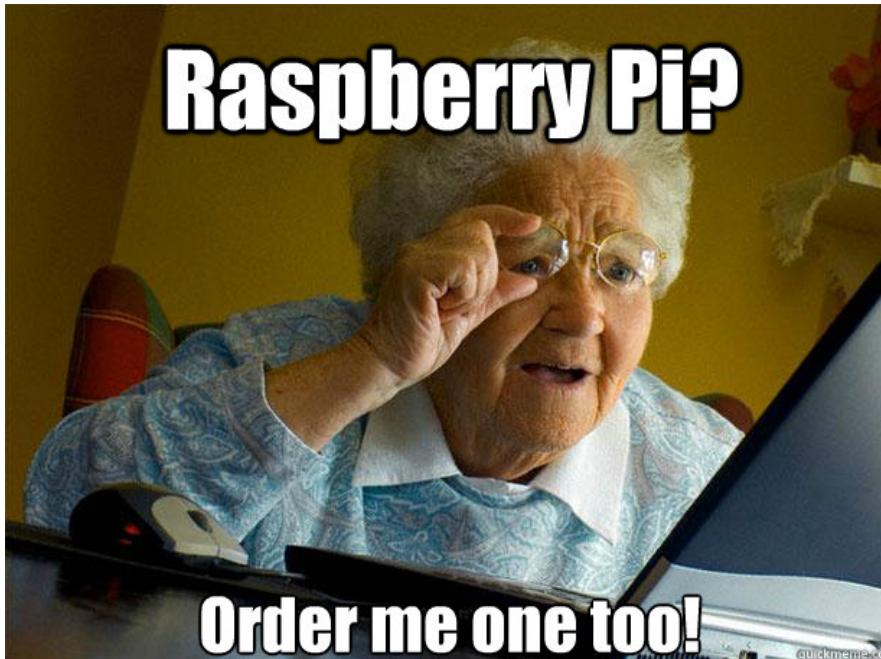
VideoCore IV 3D graphics core



source: wikipedia.org

# Price point

30-35 €



# Applications

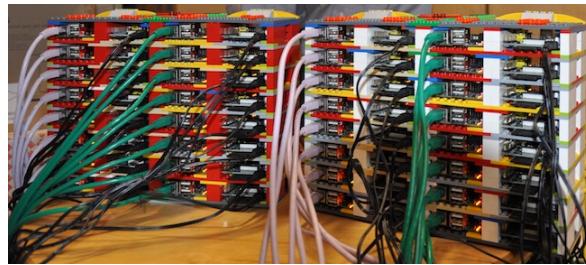
- Physical computing (home automation, sensor)

- Web server



source: blog.parts-people.com

- Mini-computer



source: wired.com

- Supercomputer



- Robotics



source: cisco.com

# Set-up

- A laptop running Linux/Windows/Mac
- SD card of min 8GB: NOOBS:Raspbian OS (or others)
- 2A power supply
- Ethernet cable

<https://www.raspberrypi.org/downloads/>

# LibreElec: KODI media center

*Kodi® (formerly known as XBMC™) is an award-winning **free** and **open source** (GPL) software media center for playing videos, music, pictures, games, and more*

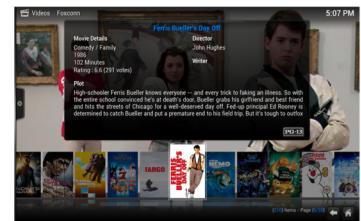
<https://libreelec.tv/download-temp/>



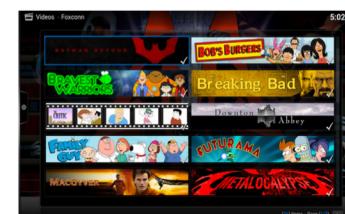
Kodi can play all your music including AAC, MP3, FLAC, OGG, WAV and WMA formats. It has cue sheet, tagging support, MusicBrainz integration, and smart playlists for ultimate control of your music collection.



Kodi can do movies too! Supporting all the main video formats and sources, including streamable online media, ISOs, 3D, H.264, HEVC, WEBM. Kodi can import these movies with full posters, fanart, disc-art, actor information, trailers, video extras, and more.



The TV shows library supports episode and season views with posters or banners, watched tags, show descriptions and actors. Video nodes/tags and smart playlists can further organize your library for special interests, making specific screens for sci-fi, anime, etc.



Import pictures into a library and browse the different views, start a slideshow, sort or filter them all using your remote control.



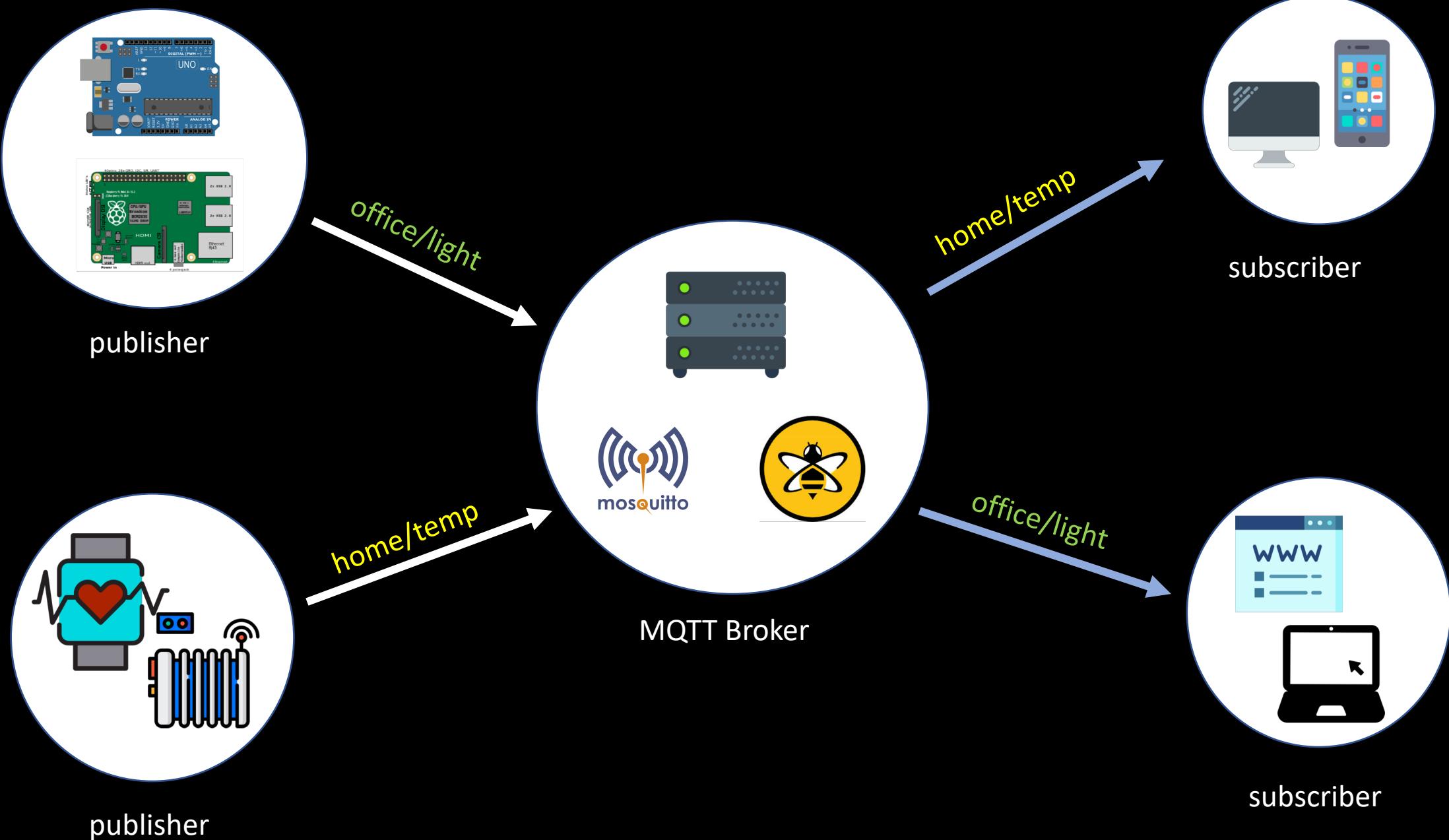


Questions?

# MQTT?

- Open Source Communication Protocol invented in 1999
- M2M / "Internet of Things"
- Simple and Lightweight
- Publish/Subscribe based
- Security?



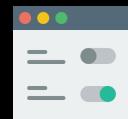


*MQTT Simple Architecture*

# How to use MQTT



Interacting with an MQTT Broker



Using UIs & Public Brokers



Or Using Code & Private Brokers



The screenshot shows a desktop application window titled "HiveMQ" with a yellow bee logo. The title bar also includes "Websockets Client Showcase". The main area is titled "Connection" and contains fields for Host (localhost), Port (8000), ClientID (clientid-mDh03dOEog), Username, Password, Keep Alive (60), SSL (unchecked), Clean Session (unchecked), Last-Will Topic, Last-Will QoS (0), Last-Will Retain (unchecked), and Last-Will Message (empty). A green circular button indicates "connected". Below this is a Python code editor window titled "run.py" showing a script for controlling a GPIO pin via MQTT. The code uses the paho library and interacts with a Raspberry Pi's GPIO pins. To the right of the code editor is a large purple "iot" logo with "eclipse.org" underneath, and below that is a blue "mosquitto" logo with a yellow antenna icon. At the bottom is a purple circle containing the text "MQTT fx".

```
run.py
1  # This is a simple example of how to use MQTT to control a
2  # GPIO pin on a Raspberry Pi. We first stop the warnings which is a feature in this Python GPIO library.
3  import paho.mqtt.client as mqtt
4  import RPi.GPIO as GPIO
5  import time
6  import sys
7  import json
8  import requests
9  import os
10 import signal
11 import atexit
12 import Adafruit_BBIO.GPIO as GPIO
13
14 # Enter your MQTT Broker details here.
15 mqtt_broker = "Enter your MQTT Broker address as a string here"
16 mqtt_broker_port = "Enter the port as a number here"
17
18 # initialize GPIO. First we stop the warnings which is a feature in this Python GPIO library.
19 GPIO.setwarnings(False)
20 # We also set the GPIO mode to BCM which is kind of standard RaspberryPi GPIO mapping scheme.
21 GPIO.setmode(GPIO.BCM)
22 # Finally do a cleanup() this means that we set all GPIO pins to its default state.
23 GPIO.cleanup()
24
25 # Set up the pin 14 as an output.
26 GPIO.setup(14, GPIO.OUT)
27 # the low state is off which is equivalent to "OFF"
28 GPIO.output(14, 0)
29
30 client = mqtt.Client()
31 # defining on_message function that controls the GPIO pin to turn ON/OFF the light and prints received messages.
32 def on_message(client, userdata, msg):
33     if (msg.payload == "on"):
34         GPIO.output(14,1)
35     if (msg.payload == "off"):
36         GPIO.output(14,0)
37     print(msg.payload)
38
39 # creating the MQTT client and establishing connection
40 # paho Python Client - documentation
41 client.on_message = on_message
42 client.connect(mqtt_broker, mqtt_broker_port)
43 client.subscribe("cisco/light")
44 client.loop_start()
```

Still Awake?  
If yes, any questions?

# Secure Shell (SSH)

Access Shell (Command Line) Remotely

On Mac/Linux:  
`ssh pi@<ip_address>`

On Windows: Install an SSH Client  
Termius or PUTTY



# Basic CLI Commands on RaspberryPi

Change Directory: `cd`

List the contents of a directory: `ls`

Run a python script: `python <filename>`

Print work directory: `pwd`

Update Packages: `sudo apt-get update`

Update Packages: `sudo apt-get upgrade`

More about Linux Commands:

<https://www.codecademy.com/learn/learn-the-command-line>

LET'S DO IT ...

# Assignment

- Install an SSH client
- Login to your RaspberryPi (ssh pi@<IP\_address>)
- Follow the instructions at  
<https://ahasna.github.io/mqtt-iot-workshop/>