Embedded, Techkriti' 17

ELECTRONICS CLUB, IIT KANPUR

Problem Statement

- **Aim:** To design a miniaturized home automation system.
- **Explanation:** The project must have the following features
  - A centralized unit consisting of a display and a Wi-Fi module, connected to at least two home appliances (like Motor, LED, etc.).
  - A user interface in laptop or any other device (e.g., Python terminal can be directly used for the same)
  - A two way communication between the centralized unit and the user interface using Wi-Fi.
  - You should be able to command the central unit, using the user interface, to switch ON/OFF the devices instantly OR set an alarm for the same.
  - A display should show the Current Time, state of the devices and the last action taken by the user.
- **Constraints:**
  - Only 8-bit microcontroller platforms like Arduino can be used.
  - Use of any single board computer like Raspberry Pi is not allowed.
  - Any extra feature can also be implemented.

What you need to build?

Broker

MQTT

Server

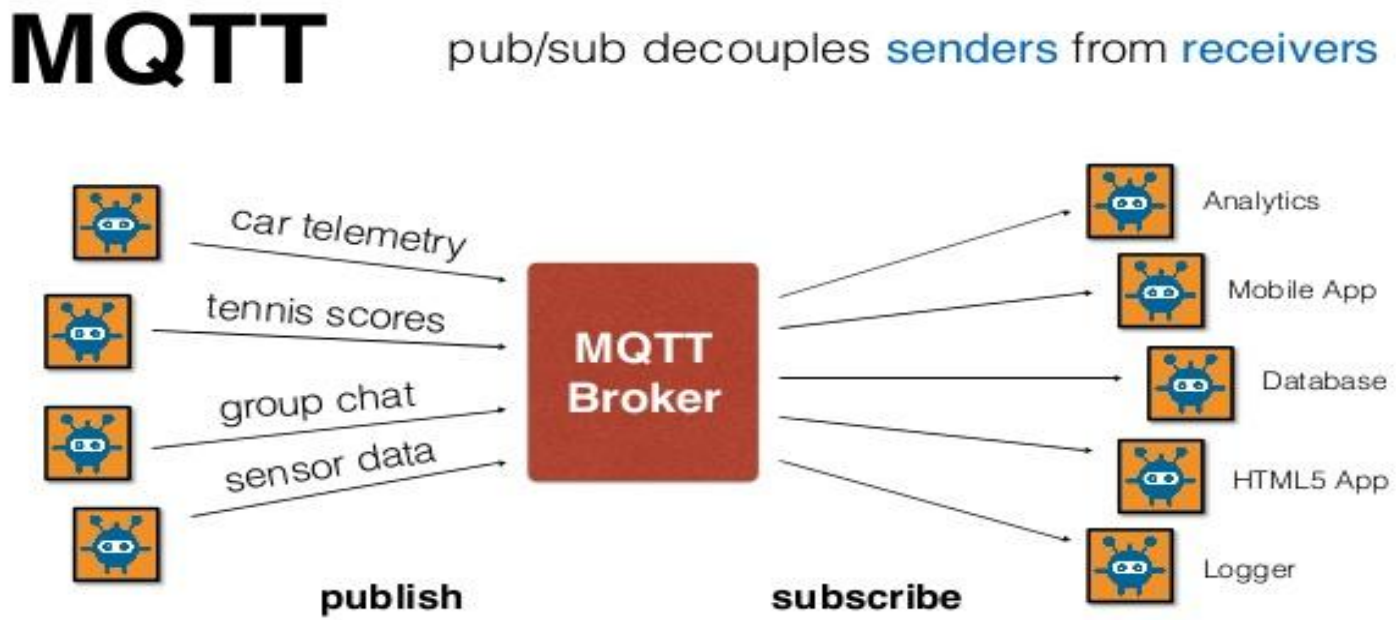
Home Appliances

1. Server-end

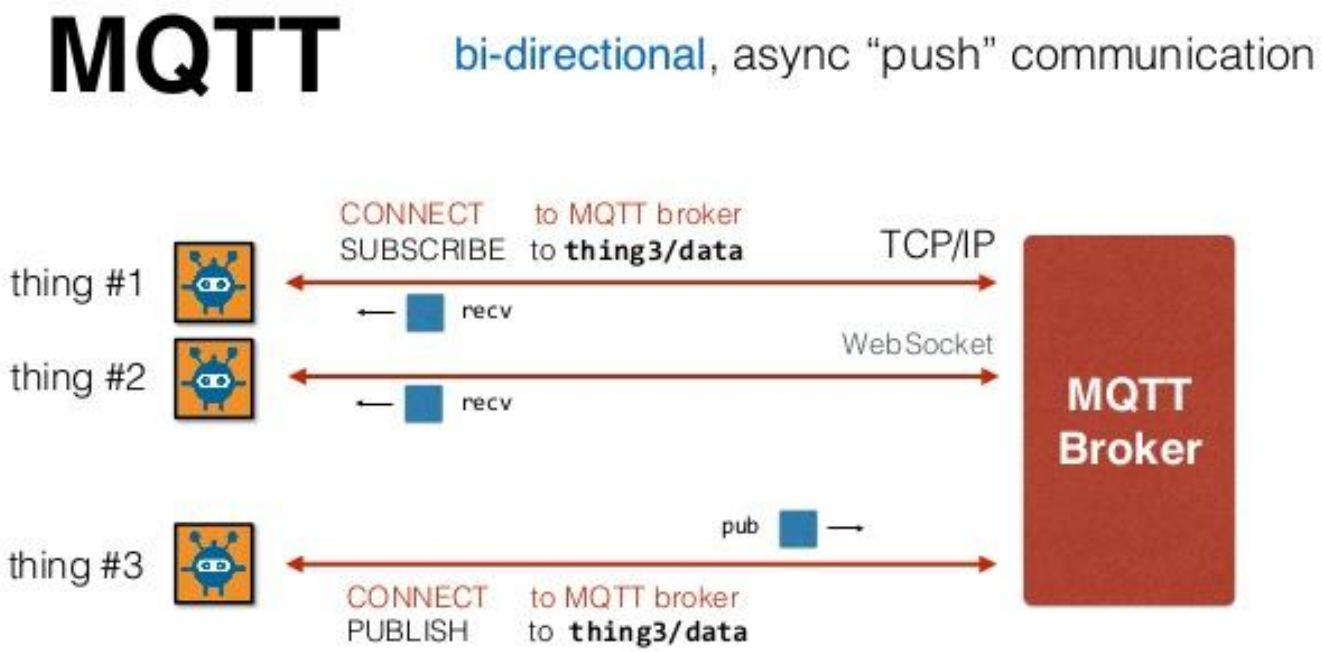
MQTT - Protocol

- Lightweight **message queuing** and **transport protocol** on top of **TCP/IP**.
  - The higher layer, Transmission Control Protocol, manages the assembling of a message or file into smaller packets that are transmitted over the Internet and received by a TCP layer that reassembles the packets into the original message. The lower layer, Internet Protocol, handles the address part of each packet so that it gets to the right destination. Each gateway computer on the network checks this address to see where to forward the message.
  - TCP/IP uses the client/server model of communication
  - TCP/IP and the higher-level applications that use it are collectively said to be "stateless" because each client request is considered a new request unrelated to any previous one
- **Broker – Client Relationship** through **Publish and Subscribe** feature.
- Extremely useful for lightweight message transfer and **IoT**.
- Clients can **Subscribe/Unsubscribe** to a **Topic**.
- Subscription to a Topic would allow clients to **get** and **push** updates for that Topic.

MQTT - Protocol



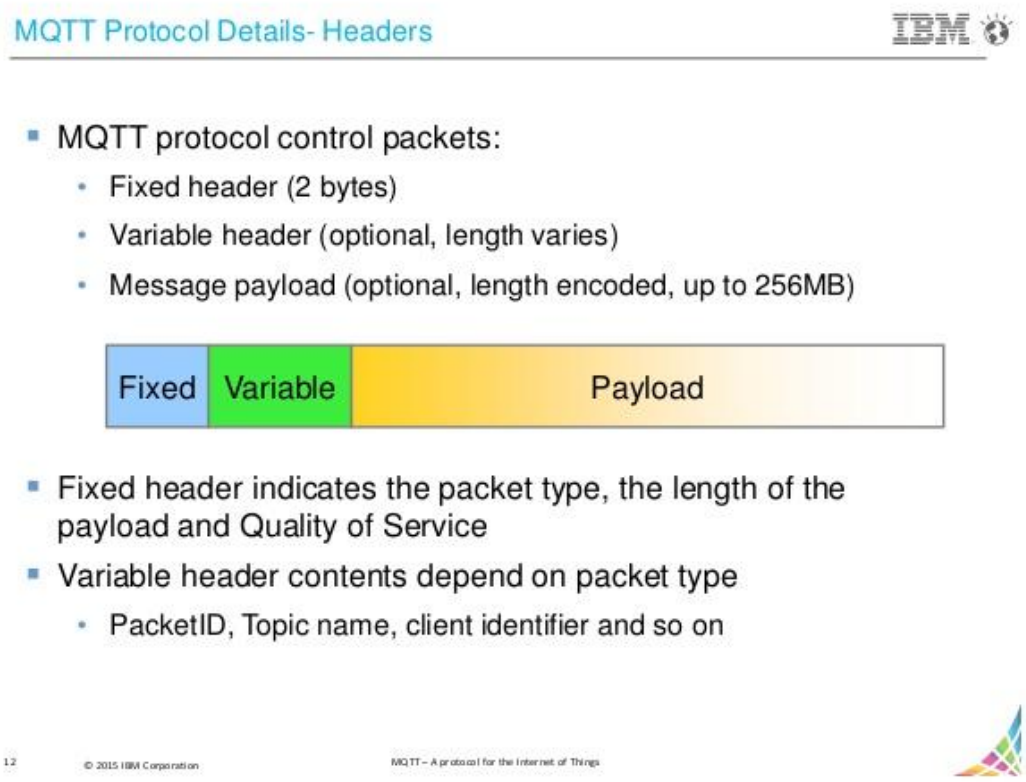
MQTT - Protocol



MQTT Methods

- **Connect** : Waits for a connection to be established with the server.
- **Disconnect** : Waits for the MQTT client to finish any work it must do, and for the TCP/IP session to disconnect.
- **Subscribe** : Waits for completion of the Subscribe or UnSubscribe method.
- **Unsubscribe** : Requests the server unsubscribe the client from one or more topics.
- **Publish** : Returns immediately to the application thread after passing the request to the MQTT client.

MQTT Control Packets



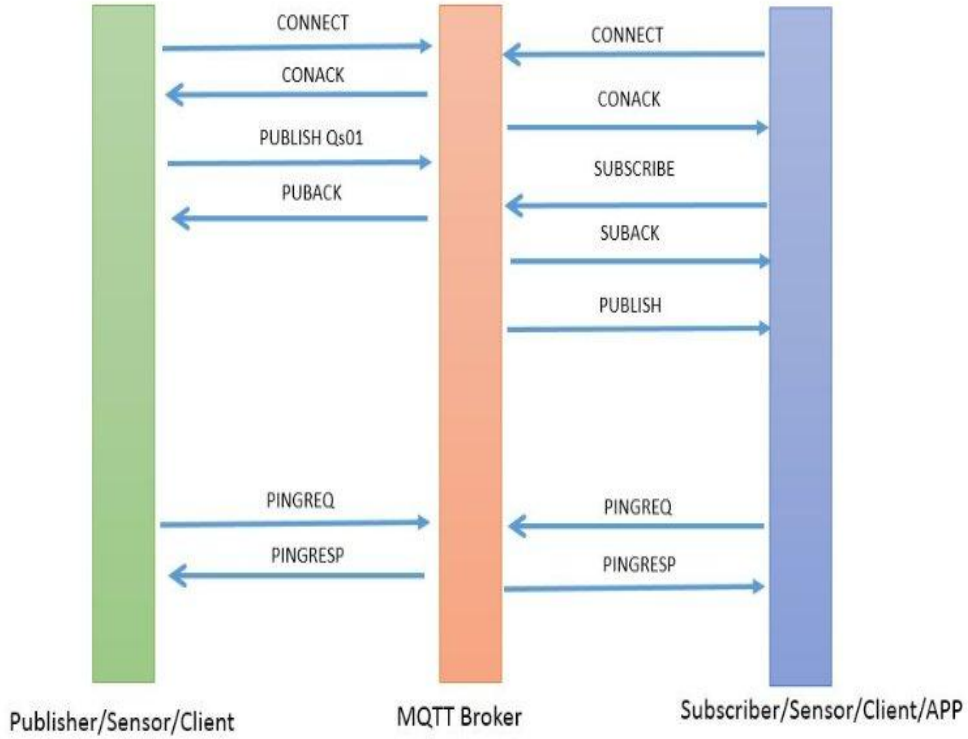
Reference link for details on Control packets:

http://docs.solace.com/MQTT-311-Prtl-Conformance-Spec/MQTT%20Control%20Packets.htm

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

MQTT Control Packets

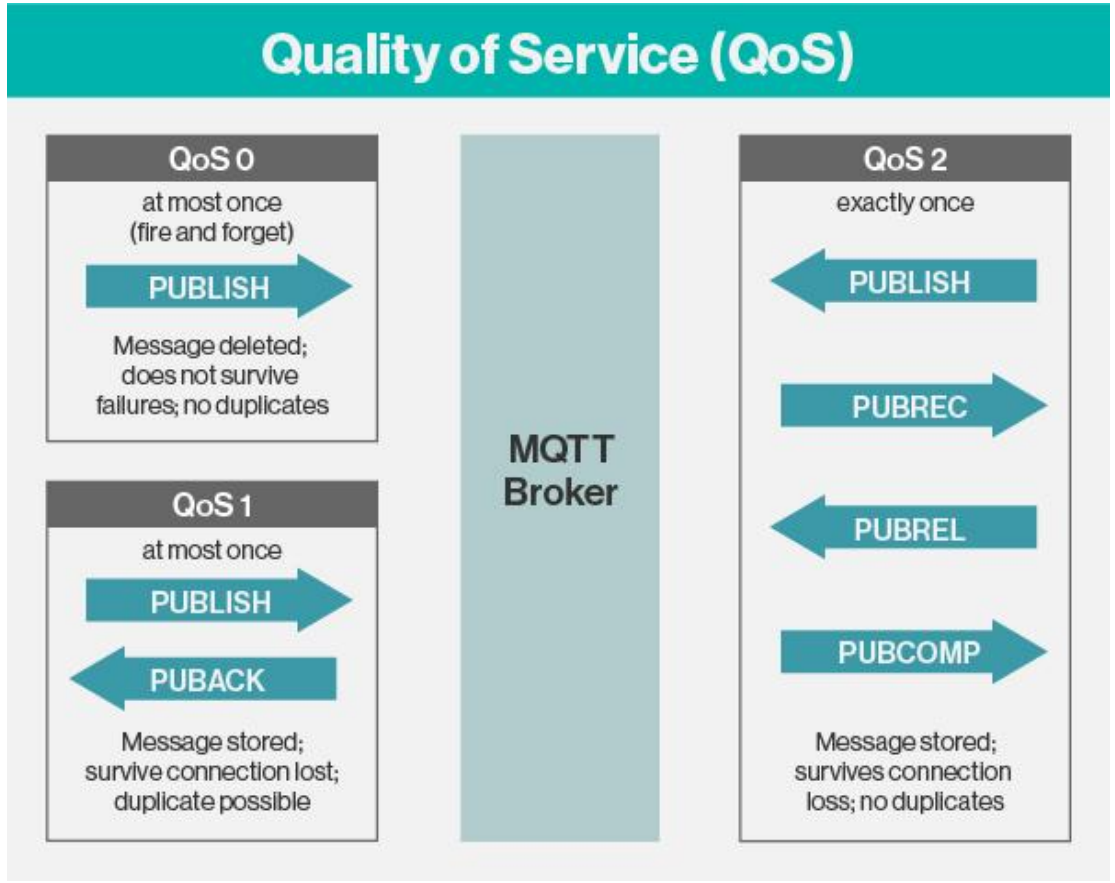
Control packet	Direction of flow	Description
CONNECT	Client to Server	Client request to connect to Server
CONNACK	Server to Client	Connect acknowledgment
PUBLISH	Client to Server or Server to Client	Publish message
PUBACK	Client to Server or Server to Client	Publish acknowledgment
PUBREC	Client to Server or Server to Client	Publish received (assured delivery part 1)
PUBREL	Client to Server or Server to Client	Publish release (assured delivery part 2)
PUBCOMP	Client to Server or Server to Client	Publish complete (assured delivery part 3)
SUBSCRIBE	Client to Server	Client subscribe request
SUBACK	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	Client to Server	Unsubscribe request
UNSUBACK	Server to Client	Unsubscribe acknowledgment
PINGREQ	Client to Server	PING request
PINGRESP	Server to Client	PING response
DISCONNECT	Client to Server	Client is disconnecting



MQTT Protocol flows

- Most control packets have a corresponding acknowledgment
- Connect
  - Can restart a previous session
  - Can specify a "Last Will and Testament" message and topic
- Subscribe can specify multiple topics
- Publish flows
  - Flow depends on QoS level
  - Sent from client → server to publish a message, or
  - Server → client to send messages

Quality of Service



1. Three levels of Quality of Service (QoS)
2. How hard the broker/client will try to ensure that a message is received
3. Messages may be sent at any QoS level, and clients may attempt to subscribe to topics at any QoS level
4. if a message is published at QoS 2 and a client is subscribed with QoS 0, the message will be delivered to that client with QoS 0.
5. If a second client is also subscribed to the same topic, but with QoS 2, then it will receive the same message but with QoS 2.
6. If a client is subscribed with QoS 2 and a message is published on QoS 0, the client will receive it on QoS 0

Wild-Cards

- MQTT – Extensions (For security etc.)
  http://www.hivemq.com/extensions/ Go to the link and show!!!

- Single level wild card : "+"
  - ex. ***myhome/groundfloor/+/temperature***
  - **a/b/c/d** OR **+/b/c/d** OR **a/+/c/d** OR **a/+/+/d** OR **+/+/+/+**

- Multi level Wild card : "#"
  - ex. ***myhome/groundfloor/#***
  - **a/b/c/d** OR **#** OR **a/#** OR **a/b/#** OR **a/b/c/#** OR **+/b/c/#**

  **Note: '#' can be used only at the end

Clean Session

1. On connection, a client sets the "clean session" flag, also known as the "clean start" flag.
2. If clean session is set to false, then the connection is treated as durable and any subscriptions it has will remain and any subsequent QoS 1 or 2 messages will be stored until it connects again in the future.
3. If clean session is true, then all subscriptions will be removed for the client when it disconnects.

Comparison

	DDS	AMQP	CoAP	MQTT	REST / HTTP	XMPP
TRANSPORT	UDP/IP (unicast + multicast) TCP/IP	TCP/IP	UDP/IP	TCP/IP	TCP/IP	TCP/IP
INTERACTION MODEL	Publish-and-Subscribe, Request-Reply	Point-to-Point Message Exchange	Request-Reply (REST)	Publish-and-Subscribe	Request-Reply	Point-to-Point Message Exchange
SCOPE	Device-to-Device Device-to-Cloud Cloud-to-Cloud	Device-to-Device Device-to-Cloud Cloud-to-Cloud	Device-to-Device	Device-to-Cloud Cloud-to-Cloud	Device-to-Cloud Cloud-to-Cloud	Device-to-Cloud Cloud-to-Cloud
AUTOMATIC DISCOVERY	✓	-	✓	-	-	-
CONTENT AWARENESS	Content-based Routing Queries	-	-	-	-	-
QoS	Extensive (20+)	Limited	Limited	Limited	-	-
INTEROPERABILITY LEVEL	Semantic	Structural	Semantic	Foundational	Semantic	Structural
SECURITY	TLS, DTLS, DDS Security	TLS + SASL	DTLS	TLS	HTTPS	TLS + SASL
DATA PRIORITIZATION	Transport Priorities	-	-	-	-	-
FAULT TOLERANCE	Decentralized	Implementation-Specific	Decentralized	Broker is SPoF	Server is SPoF	Server is SPoF

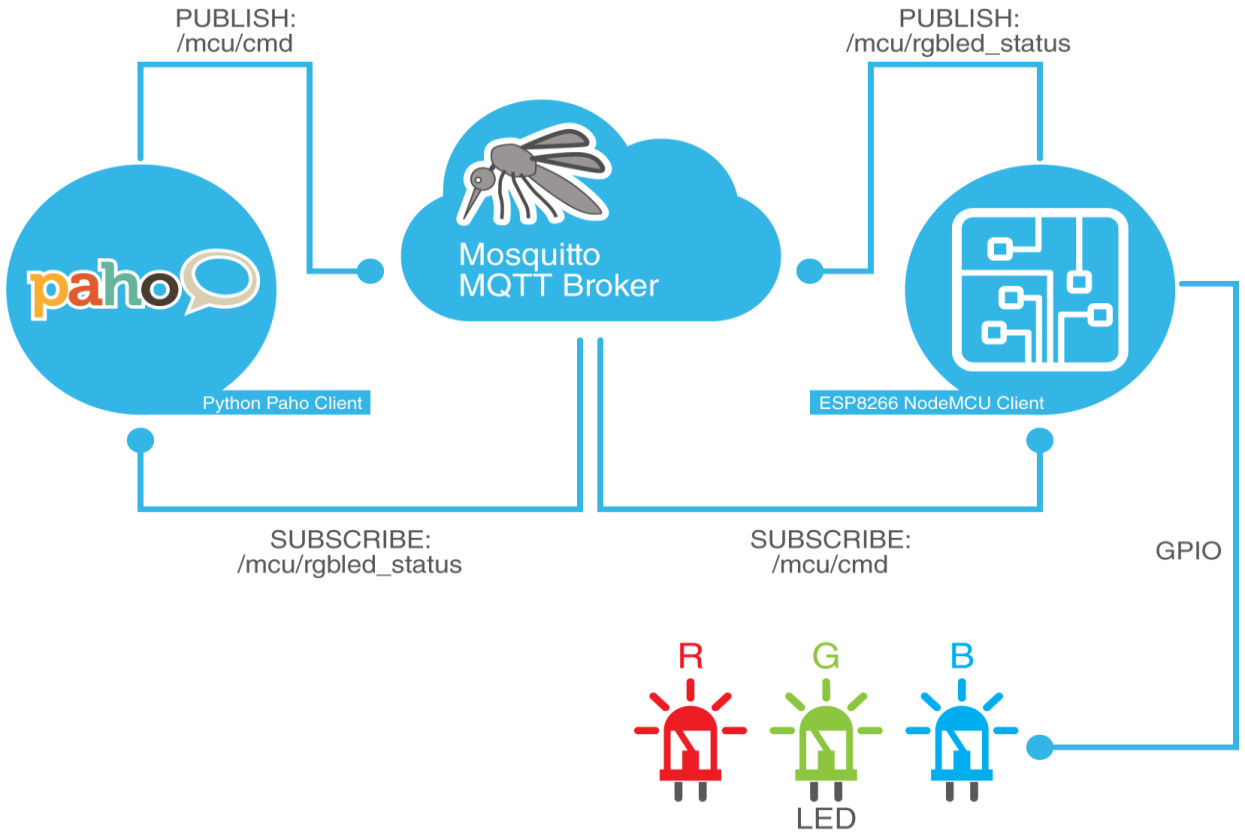
Things to Publish

- Current Time: You need to publish the current time from your laptop (server - time)
- On/Off Commands individually for each device connected to the central unit.
- Set an alarm individually for each device.
- Other things can also be included:
  - Notification to the user, of any activity related to any of the devices
  - Duration of operation of each device etc.

Running MQTT Broker

- You should have JAVA-jre preinstalled
- You can choose any of the MQTT Brokers as MQTT-Hive, MOSQUITO etc.
- Install MQTT-Hive → bin folder → run the file named “run.bat”

Schematic


PUBLISH:
/mcu/cmd
PUBLISH:
/mcu/rgbled_status
paho
Python Paho Client
Mosquitto
MQTT Broker
ESP8266 NodeMCU Client
SUBSCRIBE:
/mcu/rgbled_status
SUBSCRIBE:
/mcu/cmd
GPIO
R
G
B
LED

Python Interface

- You must have Python installed in your computer
- https://pip.pypa.io/en/stable/installing/

  (visit this site) → go to 'Installation'
- Under 'Install pip' section → click on 'get-pip.py' → save the page that occurs
- Go to Command Prompt → command: 'pip install paho-mqtt'
- And you are ready for interfacing python with your broker.

Python Interface

- You can get the basic commands at the following links

  https://pypi.python.org/pypi/paho-mqtt/1.1

  http://www.hivemq.com/blog/mqtt-client-library-paho-python

Let's have a demo!!

- We will be using :
  - HiveMQ as MQTT server (Broker)
  - Paho-MQTT as Client module.

  **Note: You are free to use any other broker-client module.

  **You also need to install JAVA-jre beforehand.

2. Centralized Unit

Integrating Components

- ESP-8266 WiFi module
- Arduino micro-controller
- GLCD display
- At-least ‘2’ devices (wired connections) connected to the central unit, which are to be controlled through the server

Arduino and WiFi module

- Install MQTT library for arduino
- You need to subscribe to certain ‘Topics’ with the broker
- Whenever a message is received from the broker, take the corresponding action

Final Model

Python interface
Client-1

Central Unit
Client-2

Commands

Publish and Subscribe

MQTT Broker

Judging Criterion

- **All the teams are required to make a PowerPoint presentation which should strictly follow the below format:**
  - A detailed explanation of the system with the help of a neat system-level block diagram and an operational flowchart.
  - Cost effectiveness
  - The effectiveness of the hardware and software used in solving the problem statement
  - User interface of the device. The overall design of the gadget.
  - Exact cost analysis of all the components used to build the system.
  - Innovation in design of the device - Example: power consumption, compactness, robustness in terms of usage, etc.
  - All basic features should be implemented and only after their evaluation the extra features would be considered and assessed.

Other Options

- You are free to use any other protocol HTTP, MQTT etc.
- For the user interface, you can design an android app and use it as one of the clients. This would make things more user friendly.
- Any other way of implementation is acceptable, unless it satisfies the bare minimum criterion

Thank You