**COMPONENTS USED – IOT PLATFORM ARCHITECTURE**

Client Machine

(USER)

Breadboard with

LED LIGHTS

Access

(Sends request)

Control Interface

(Webpage)

CLOUD INTERFACE (AWS)

IOT PLATFORM

RASPBERRY PI

Subscriber

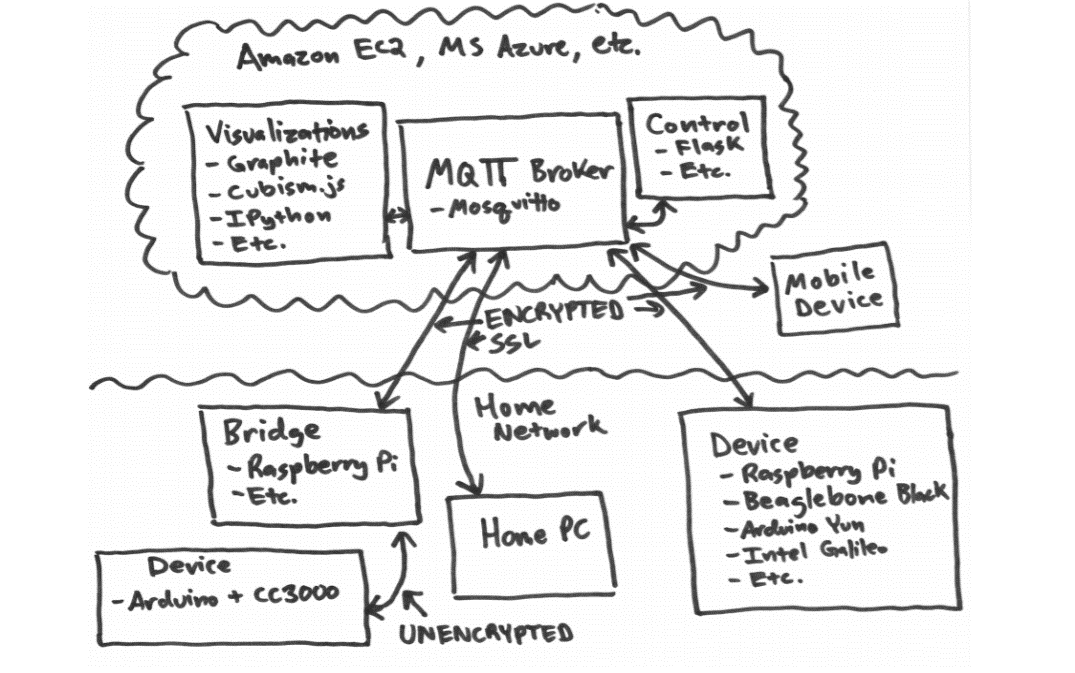
Register device

MQTT Message Broker (Publisher)

LED Toggle status

Connect to Raspberry Pi

Raspberry PI status



Source - <https://media.readthedocs.org/pdf/diy-device-cloud/latest/diy-device-cloud.pdf>

The user of the IOT device communicates with the device through the cloud platform. The user accesses the web page hosted on the cloud, AWS in this case. The message broker is also hosted on the same server.

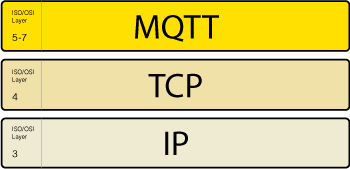
**Message Broker – MQTT**

We are using **Mosquitto** MQTT server, which is an open source message broker service that uses MQTT protocol to send and receive messages.

MQTT (Message Queueing Telemetry Transport) is a messaging protocol for Internet of thing devices. It is typically a publish/subscribe protocol.  It allows clients to connect as a publisher, subscriber, or both. The broker allows devices to communicate to other devices or applications without having to be tightly coupled or directly connected to each other.

**MQTT connection**

The MQTT protocol is based on top of TCP/IP and both client and broker need to have a TCP/IP stack.



The MQTT connection itself is always between one client and the broker, no client is connected to another client directly. **The connection is initiated through a client sending a CONNECT message to the broker. The broker response with a CONNACK** and a status code. Once the connection is established, the broker will keep it open as long as the client doesn’t send a disconnect command or it loses the connection.

**Installing MQTT on raspberry pi**

1. import the repository package

* wget <http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key>
* sudo apt-key add mosquitto-repo.gpg.key

1. Then make the repository available to apt:

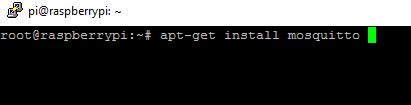
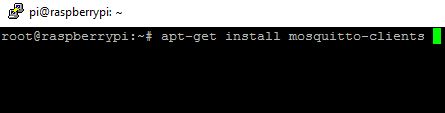
* cd /etc/apt/sources.list.d/

1. Enter the following

* sudo wget<http://repo.mosquitto.org/debian/mosquitto-wheezy.list>

1. Install mosquito

* apt-get install mosquito



1. Install mosquito clients

* apt-get install mosquitto-clients

**Install Paho MQTT**

* pip install paho-mqtt

**Build from source**

* git clone https://github.com/eclipse/paho.mqtt.python.git
* cd org.eclipse.paho.mqtt.python.git
* python setup.py install

**mqtt\_subscribe.py**

# Import package

import paho.mqtt.client as mqtt

# Define Variables

MQTT\_HOST = "iot.eclipse.org"

MQTT\_PORT = 1883

MQTT\_KEEPALIVE\_INTERVAL = 45

MQTT\_TOPIC = "helloTopic"

MQTT\_MSG = "hello MQTT"

# Define on connect event function

# We shall subscribe to our Topic in this function

def on\_connect(mosq, obj, rc):

mqttc.subscribe(MQTT\_TOPIC, 0)

# Define on\_message event function.

# This function will be invoked every time,

# a new message arrives for the subscribed topic

def on\_message(mosq, obj, msg):

print "Topic: " + str(msg.topic)

print "QoS: " + str(msg.qos)

print "Payload: " + str(msg.payload)

def on\_subscribe(mosq, obj, mid, granted\_qos):

print("Subscribed to Topic: " +

MQTT\_MSG + " with QoS: " + str(granted\_qos))

# Initiate MQTT Client

mqttc = mqtt.Client()

# Assign event callbacks

mqttc.on\_message = on\_message

mqttc.on\_connect = on\_connect

mqttc.on\_subscribe = on\_subscribe

# Connect with MQTT Broker

mqttc.connect(MQTT\_HOST, MQTT\_PORT, MQTT\_KEEPALIVE\_INTERVAL)

mqttc.loop\_forever()

# Continue monitoring the incoming messages for subscribed topic

**mqtt\_publish.py**

# Import package

import paho.mqtt.client as mqtt

# Define Variables

MQTT\_HOST = "iot.eclipse.org"

MQTT\_PORT = 1883

MQTT\_KEEPALIVE\_INTERVAL = 45

MQTT\_TOPIC = "helloTopic"

MQTT\_MSG = "hello MQTT"

# Define on\_publish event function

def on\_publish(client, userdata, mid):

print "Message Published..."

# Initiate MQTT Client

mqttc = mqtt.Client()

# Register publish callback function

mqttc.on\_publish = on\_publish

# Connect with MQTT Broker

mqttc.connect(MQTT\_HOST, MQTT\_PORT, MQTT\_KEEPALIVE\_INTERVAL)

# Publish message to MQTT Broker

mqttc.publish(MQTT\_TOPIC,MQTT\_MSG)

# Disconnect from MQTT\_Broker

mqttc.disconnect()

**Webpage**

A webpage is developed in Python which serves as the control interface. A user can sign up/login and register his device. The LED can be toggled on/off. A log is maintained for managing the users information. In addition, logs are maintained for raspberry Pi connections and for LED usage.

**Why do we need cloud, instead of accessing the smart devices directly?**

1. Discovering and connecting to your devices over the internet is hard. Dynamic IP addresses, firewalls, and more prevent you from easily accessing your home network and devices. Hosting your own device cloud on the internet allows devices to connect and communicate without the trouble of opening connectivity to your home network.
2. A device cloud allows for asynchronous communication between devices. As the device is connected at all times, it reduces power consumption greatly.
3. Total control over the data and services available to devices. You aren’t limited by what a 3rd party provides or concerned about what they might do with your data. You own the entire infrastructure and can mold it to your needs.
4. User security and privacy is increased as devices utilizing common APIs and back-end infrastructure, vital security updates are instantaneous.