MQTT(MQ Telemetry Transport or Message Queuing Telemetry Transport) Protocol is an ISO standard pub-sub messaging protocol.It is a lightweight messaging protocol that provides network clients a simple way to distribute telemetry information.It is an ideal protocol for machine to machine(M2M) communication.

History of MQTT:

MQTT was first developed by Dr.Andy Stanford Clark of IBM and Arien Nipper of Arcom (presently Eurotech).It was created in 1999 as a cost effective, small bandwidth and reliable way to connect monitoring devices used in the industries.

Why Use MQTT?

MQTT was designed to conserve both power and memory.It is an messaging protocol where the message size is small upto 2 Byte and in the form of binary but in case of other protocols (Eg.http) the message size is large and the size may go upto 40 bytes and it is in the ASCII format. The data distribution in MQTT is 1 to 0/1/N whereas in an http protocol its one to one.

Mosquitto:

Eclipse Mosquitto is an open source message broker that implements the MQTT protocol . Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers.

MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model for low bandwidth devices.

For Eg let’s take the scenario of newspaper distribution in our own locality.The first thing we have to do is to subscribe for a particular newspaper from the newspaper vendor. Whenever a new edition of the subscribed newspaper is available ie. daily he delivers it to us in case of a public holiday when there is no newspaper ie. no data available we don’t find any papers.It is delivered to the people who have subscribed to that particular topic. This is how Publish/Subscribe method works.

Deploying MQTT:

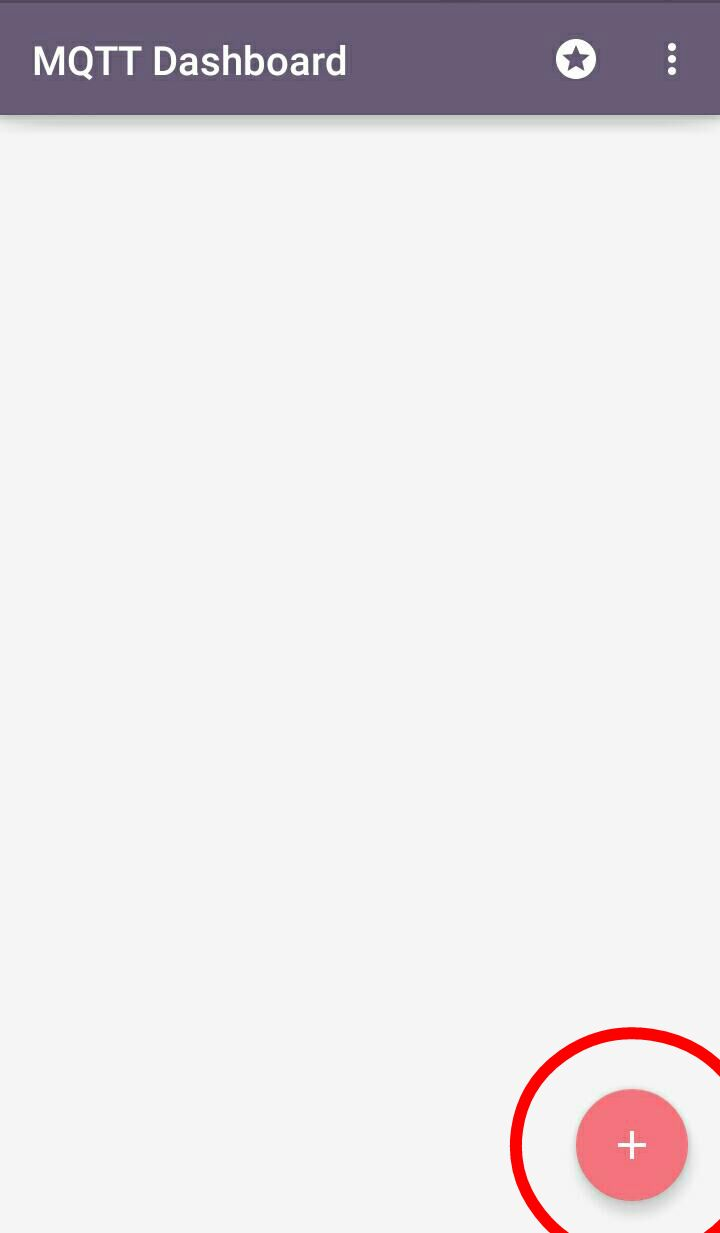
While Creating your instance select the custom TCP protocol with the port range of 1883.

\*See that you also have SSH protocol along with the Custom TCP\*

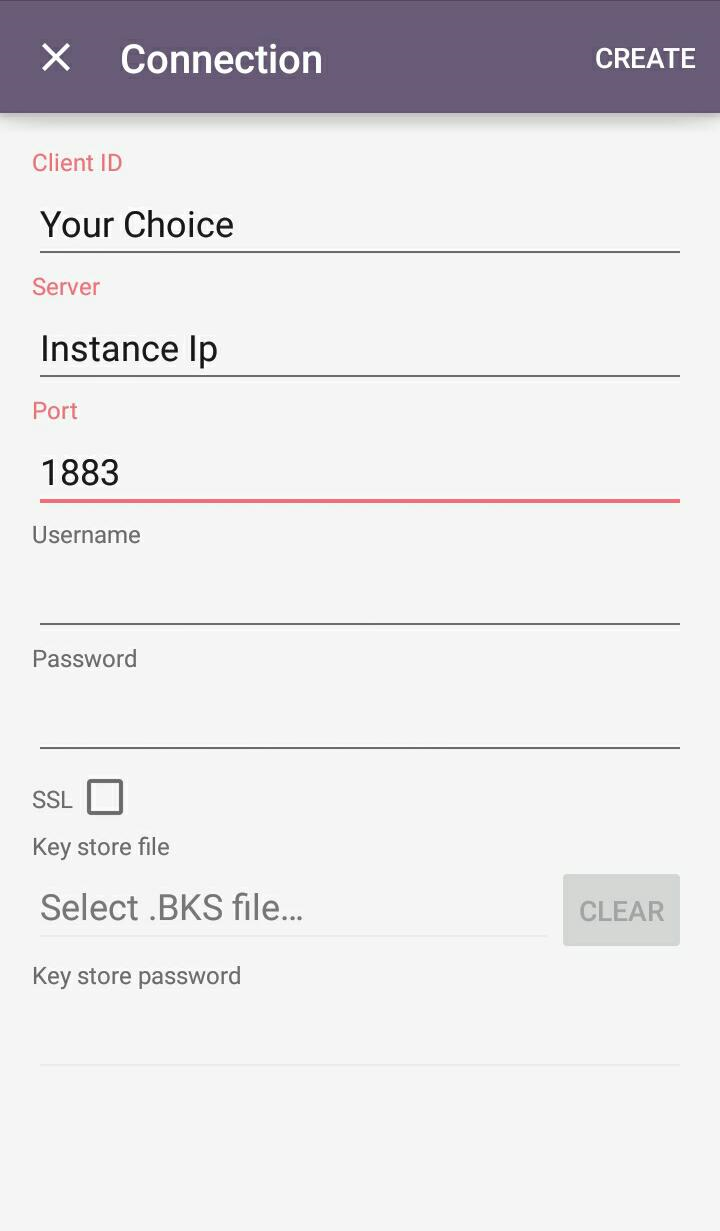
Download IoT MQTT Dashboard Application from the Google Play Store.

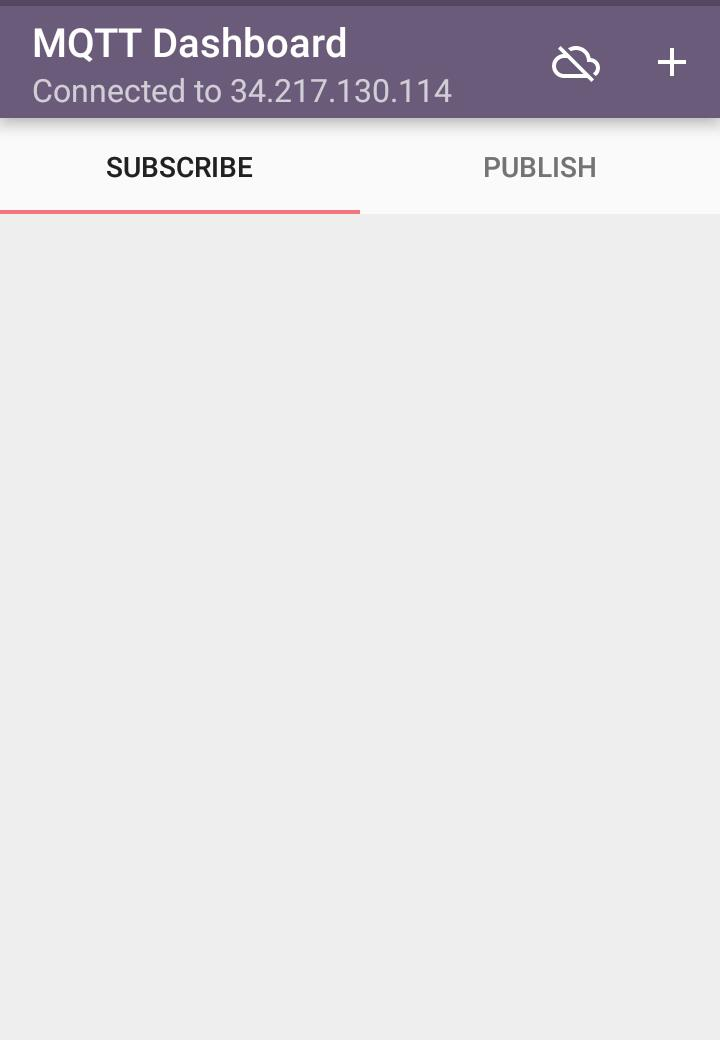
Open MQTT Dashboard App

Click on the Plus Mark available on Right Bottom End of the screen



Type in your Client ID(Your Choice), Server Ip(your instance Ip), Port Range as 1883 and Create your Connection.



If Everything is done correctly then the status should be “Connected to ---(IP Address)” 

The connection is successfully created and now you can subscribe or publish information.

To publish

Click on publish and type in friendly name(of your choice),topic (distinct one of your choice) and then click on create.

It’s time now for you to install mosquitto on to the Pi

For installing mosquitto on to Pi run the following commands

* Type the command to download Python client and install Pip

pip install paho-mqtt

* Create a directory and access it

mkdir mqtt

cd mqtt

* Create a file in the directory and paste the program in the file

sudo nano pahomqtt.py

import paho.mqtt.client as mqtt  
# The callback for when the client receives a CONNACK response from the server.  
def on\_connect(client, userdata, rc):  
 print("Connected with result code "+str(rc))  
 # Subscribing in on\_connect() means that if we lose the connection and  
 # reconnect then subscriptions will be renewed.  
 client.subscribe("topic")  
  
# The callback for when a PUBLISH message is received from the server.  
def on\_message(client, userdata, msg):  
 print(msg.topic+" "+str(msg.payload))  
  
client = mqtt.Client()  
client.on\_connect = on\_connect  
client.on\_message = on\_message  
  
client.connect("instance ip address", 1883, 60)  
  
  
client.loop\_forever()

* Compile and run the code

python mqtt.py

**Turning led on and off using Pi and MQTT**

* Create a file in the directory and paste the program in the file

sudo nano pahomqttled.py

Import time

Import RPi.GPIO as GPIO

import paho.mqtt.client as mqtt

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)  
  
# The callback for when the client receives a CONNACK response from the server.  
def on\_connect(client, userdata, flags, rc):  
 print("Connected with result code "+str(rc))  
  
 # Subscribing in on\_connect() means that if we lose the connection and  
 # reconnect then subscriptions will be renewed.  
 client.subscribe("TOPIC")

#client.publish(“topic”, “message” , 0)  
  
# The callback for when a PUBLISH message is received from the server.  
def on\_message(client, userdata, msg):  
 print(msg.topic+" "+str(msg.payload))

if(str(msg.payload)==”1”):

print(“LED ON”):

GPIO.OUTPUT(11, True)

else:

print(“LED OFF”):

GPIO.OUTPUT(11, False)

client = mqtt.Client()  
client.on\_connect = on\_connect  
client.on\_message = on\_message  
  
client.connect("Your instance", 1883, 60)  
  
# Blocking call that processes network traffic, dispatches callbacks and  
# handles reconnecting.  
# Other loop\*() functions are available that give a threaded interface and a  
# manual interface.  
client.loop\_forever()