# **[Demo 14: How to use MQTT and Arduino ESP32 to build a simple Smart home system](http://www.iotsharing.com/2017/05/how-to-use-mqtt-to-build-smart-home-arduino-esp32.html)**

**1. Introduction**

- Currently, there are many IoT protocols such as: CoAP, MQTT, AMQP, … In this tutorial, I will introduce MQTT, one of the famous IoT protocols. This protocol is to control and transfer data between devices in an IoT network  
**Note**: for MQTTS please refer [Demo 30: How to use Arduino ESP32 MQTTS with MQTTS Mosquitto broker (TLS/SSL).](http://www.iotsharing.com/2017/08/how-to-use-esp32-mqtts-with-mqtts-mosquitto-broker-tls-ssl.html)

MQTT is stand for Message Queuing Telemetry Transport. It has some features:

+    Use Publish/Subscribe/Topic mechanism

+    Lightweight protocol

+    Small code footprint

+    Build on top of the TCP/IP protocol

+    Less network bandwidth.

- The principal of MQTT is traditional Client-Server model. In this model, there is one MQTT Server (also called Broker) and many MQTT Clients. The MQTT Clients always keep connection with MQTT Server. The role of MQTT Server (broker) is to filter and forward the messages to subscribed MQTT Clients. The communication between clients is based on Publish/Subscribe/Topic pattern in which:

+    **Topic**: the message that is transferred on network

+    **Publish**: sending the topic to network

+ **Subscribe**: listening topic that the client is interested in (ignore uninterested topic)

**- Topic is an utf-8 string** and has one - many levels which is separated by splash **"/"**. For example: **"floor1/room1/temp"**: this topic has 3 levels, human readable and easy to understand (we have floor 1 and in room 1 with temperature sensor). You can refer:

<http://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices>

- Beside that, there are other concept that you need to know:

\***QoS (Quality of Service)**: this indicator perform the guaranty of message exchange between sender and receiver. There are 3 levels:

+ QoS 0 - at most once (this level is the fastest, but not reliable)  
+ QoS 1 - at least once (this is the default mode)  
+ QoS 2 - exactly once (this level is the most reliable, but slowest)

You can refer at:

[https://www.ibm.com/support/knowledgecenter/en/SSFKSJ\_8.0.0/com.ibm.mq.dev.doc/q029090\_.htm](https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_8.0.0/com.ibm.mq.dev.doc/q029090_.htm" \t "http://www.iotsharing.com/2017/05/_blank)

\***Retained Messages**:  broker will keep the sent message so that when there is new subscriber that subscribe the topic that matches the retained message then that message will be sent to that subscriber.

- Most of MQTT libraries define some standard methods such as:

+    **Connect()**: connect to MQTT server.

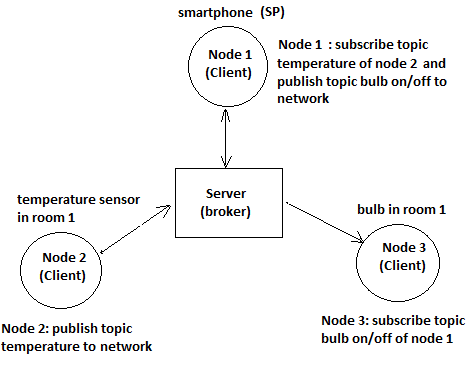
+    **Disconnect()**: disconnect from MQTT server.

+    **Subscribe()**: subscribe a topic with MQTT server.

+    **UnSubscribe()**: unsubscribe a topic with MQTT server

+    **Publish()**: client publish a topic to network.

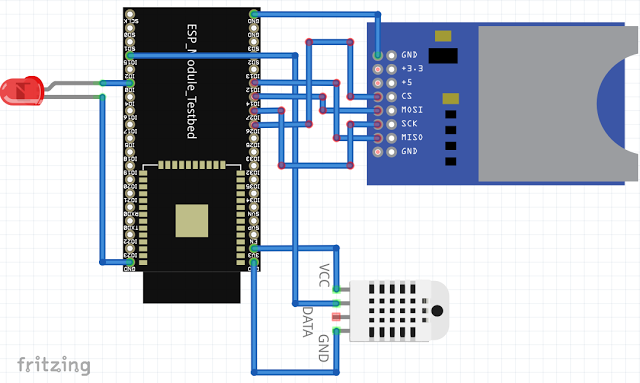
- For demo, we create a simple smart home network that have 3 client nodes (Smart phone, WiFi MCU with temperature sensor, WiFi MCU with LED/bulb controller) and 1 server node as a broker (PC or Raspberry Pi). In our application, we want to use smart phone to monitor the temperature and control the LED/bulb on or off.  So we design the MQTT model like below:

[](https://2.bp.blogspot.com/-5lhJwh6Wc6Q/WSPVEseIy2I/AAAAAAAAD68/6YQ7OJ4l84IvzSmU3Jns0FzpbVxbx-icACPcB/s1600/smart-home-mqtt-esp32.png)

**Figure: MQTT model for simple smart home application**

**2. Hardware**  
- To implement the model above, I will collect Node2 and Node3 into one node and this node is our ESP32 with DHT22 sensor and LED (bulb). Finally, we have 2 nodes: SM node and ESP32 node. We re-use the hardware schematic of [Demo 13: How to display temperature/humidity using Google Chart/Jquery and control LED through Arduino ESP32 Web Server](http://www.iotsharing.com/2017/05/how-to-display-dht11-google-chart-jquery-led-control-esp32-webserver.html" \t "http://www.iotsharing.com/2017/05/_blank). In this demo we do not use microSD so please ignore it.

- Connections:  
Connect VCC and GND of DHT22 (VCC=3.3V) to VCC (Vcc=3.3V) and GND of ESP32.  
[ESP32 IO15 - DHT22 DATA]  
[ESP32 IO2 - LED ANODE]  
[ESP32 GND - LED CATHODE]

[](https://2.bp.blogspot.com/-Tlv2_AA2g0o/WSMGl9clAUI/AAAAAAAAD6Q/mtiv90hz18UDV3XBFLDzGFMLPDHBQrv5wCPcB/s1600/esp32webser-chart-dht2-led.png)

**Figure: esp32 + dht22 + LED for MQTT smart home demo**

**3. Software  
3.1 MQTT Client side**  
**SM node:** I will use an Android Smartphone with a MQTT client application (**IoT MQTT Dashboard)**that is available on Google Play. You can download it here:  
<https://play.google.com/store/apps/details?id=com.thn.iotmqttdashboard>  
**ESP2 node:** I will use a MQTT client library (Pubsubclient). You can download it here:   
<https://github.com/knolleary/pubsubclient>  
Then unzip the downloaded file and copy it to Arduino/libraries folder.  
- This library supports some standard functions that are mentioned above. To use these function we create an instance **PubSubClient client(wifiClient)**. Because MQTT is built on top of the TCP/IP protocol so the input of this constructor is a TCP WiFiClient object.

**3.2 MQTT server side**  
- I will use a popular MQTT server called **Mosquito**. You can download and install it here:  
**- Windows user**:

<http://www.eclipse.org/downloads/download.php?file=/mosquitto/binary/win32/mosquitto-1.4.11-install-win32.exe>  
After finishing, from command line just run this command to start mosquito server: "mosquitto"  
**-  Ubuntu user:**   
From command line type the command below:

**sudo apt-get install mosquitto mosquitto-clients**  
This will install mosquito as a service. You can check whether the service is start or not by using command line:

**sudo service --status-all 2>&1 | grep mosquitto**  
-  Other OS, just follow:  <https://mosquitto.org/download/>  
**Note:** when we install mosquito, it also install 2 client programs called "**mosquitt\_sub**" and "**mosquito\_pub**" that we can use for debugging.

For example:   
- To monitor all topic on network using:

**mosquitto\_sub -v -h broker\_ip -p 1883 -t "#" (change broker\_ip to mqtt server ip)**  
- To publish a topic (publish topic room1/temp with value 30) using:

**mosquitto\_pub -t 'room1/temp' -m 30**

**3.3 Assign roles**  
We define topics:

-  Topic1: **smarthome/room1/bulb #value** : value can take 0 or 1 means on/off the LED (bulb).

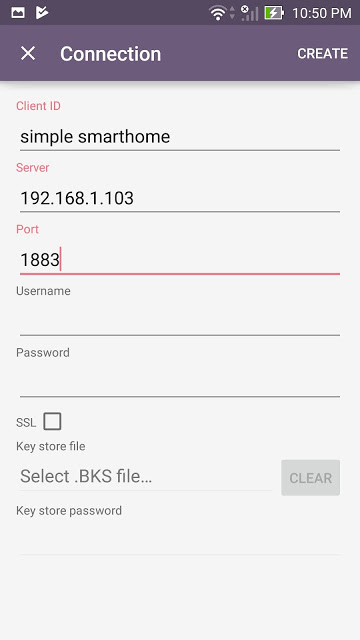
- Topic2: **smarthome/room1/temperature #value** : value can take float number to express temperature.  
Example: smarthome/room1/temperature 30

With these topics SM node can subscribe Topic2 and publish Topic1. ESP32 node can publish Topic2 and subscribe Topic1.

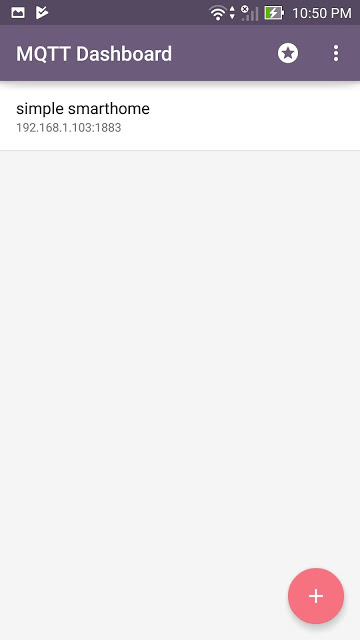
**3.4 Steps to run the system**  
- Start the MQTT server (on Wins invoke it manually, on Linux it is a service so just check the service is started)  
- From Terminal run this: **mosquitto\_sub -v -h broker\_ip -p 1883 -t '#'** for debugging. You will see all the messages on the network.  
- Create an Arduino project and Save as **esp32mqtt**with code:

|  |
| --- |
| /\* Here ESP32 will keep 2 roles:  1/ read data from DHT11/DHT22 sensor  2/ control led on-off  So it willpublish temperature topic and scribe topic bulb on/off  \*/  #include <WiFi.h>  #include <PubSubClient.h>  #include "DHT.h"  /\* change it with your ssid-password \*/  const char\* ssid = "dd-wrt";  const char\* password = "0000000000";  /\* this is the IP of PC/raspberry where you installed MQTT Server  on Wins use "ipconfig"  on Linux use "ifconfig" to get its IP address \*/  const char\* mqtt\_server = "192.168.1.103";  /\* define DHT pins \*/  #define DHTPIN 14  #define DHTTYPE DHT22  DHT dht(DHTPIN, DHTTYPE);  float temperature = 0;  /\* create an instance of PubSubClient client \*/  WiFiClient espClient;  PubSubClient client(espClient);  /\*LED GPIO pin\*/  const char led = 12;  /\* topics \*/  #define TEMP\_TOPIC "smarthome/room1/temp"  #define LED\_TOPIC "smarthome/room1/led" /\* 1=on, 0=off \*/  long lastMsg = 0;  char msg[20];  void receivedCallback(char\* topic, byte\* payload, unsigned int length) {  Serial.print("Message received: ");  Serial.println(topic);  Serial.print("payload: ");  for (int i = 0; i < length; i++) {  Serial.print((char)payload[i]);  }  Serial.println();  /\* we got '1' -> on \*/  if ((char)payload[0] == '1') {  digitalWrite(led, HIGH);  } else {  /\* we got '0' -> on \*/  digitalWrite(led, LOW);  }  }  void mqttconnect() {  /\* Loop until reconnected \*/  while (!client.connected()) {  Serial.print("MQTT connecting ...");  /\* client ID \*/  String clientId = "ESP32Client";  /\* connect now \*/  if (client.connect(clientId.c\_str())) {  Serial.println("connected");  /\* subscribe topic with default QoS 0\*/  client.subscribe(LED\_TOPIC);  } else {  Serial.print("failed, status code =");  Serial.print(client.state());  Serial.println("try again in 5 seconds");  /\* Wait 5 seconds before retrying \*/  delay(5000);  }  }  }  void setup() {  Serial.begin(115200);  // We start by connecting to a WiFi network  Serial.println();  Serial.print("Connecting to ");  Serial.println(ssid);  WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  /\* set led as output to control led on-off \*/  pinMode(led, OUTPUT);  Serial.println("");  Serial.println("WiFi connected");  Serial.println("IP address: ");  Serial.println(WiFi.localIP());  /\* configure the MQTT server with IPaddress and port \*/  client.setServer(mqtt\_server, 1883);  /\* this receivedCallback function will be invoked  when client received subscribed topic \*/  client.setCallback(receivedCallback);  /\*start DHT sensor \*/  dht.begin();  }  void loop() {  /\* if client was disconnected then try to reconnect again \*/  if (!client.connected()) {  mqttconnect();  }  /\* this function will listen for incomming  subscribed topic-process-invoke receivedCallback \*/  client.loop();  /\* we measure temperature every 3 secs  we count until 3 secs reached to avoid blocking program if using delay()\*/  long now = millis();  if (now - lastMsg > 3000) {  lastMsg = now;  /\* read DHT11/DHT22 sensor and convert to string \*/  temperature = dht.readTemperature();  if (!isnan(temperature)) {  snprintf (msg, 20, "%lf", temperature);  /\* publish the message \*/  client.publish(TEMP\_TOPIC, msg);  }  }  } |

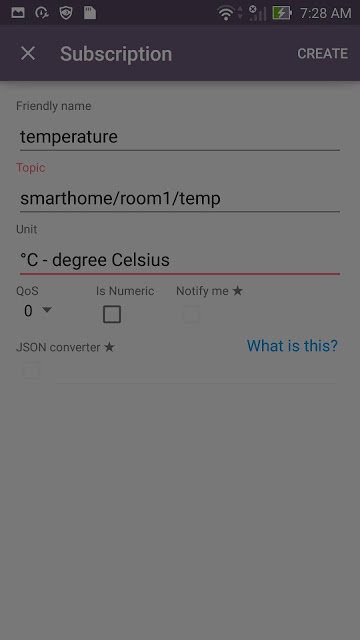
**-**From Android smart phone, open IoT MQTT Dashboard) and follow steps below to set up it a MQTT client:

[](https://4.bp.blogspot.com/-S98LwSuGD4g/WSRbkBTBnyI/AAAAAAAAD7Q/1_NgdXb_la04vLn-uxFySmeiz4ILET-oACEw/s1600/h1.jpg)

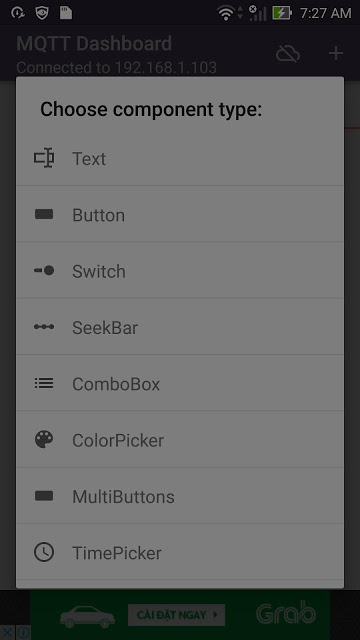
**Figure: Configure MQTT Server that it will connect to**

[](https://2.bp.blogspot.com/-CuzE4bu8dlM/WSRbkHl9IzI/AAAAAAAAD7k/uVw-oZvdFOQiIcgOm0Gb1kJgnaczmvBCwCEw/s1600/h2.jpg)

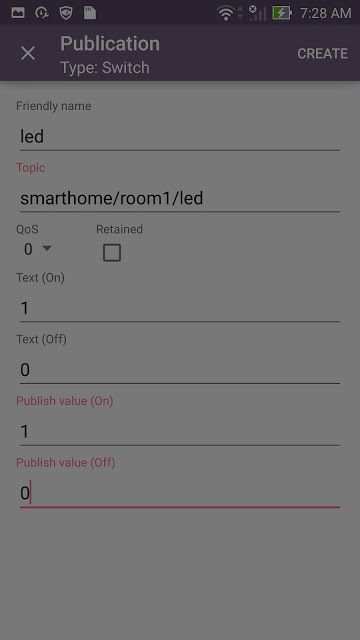
**Figure: After configuring server, choose it**

[](https://3.bp.blogspot.com/-IVTNgoS4vtk/WSRbj2j_WxI/AAAAAAAAD7k/_nyUqs98bgA_21B4Q97c4uO5sdQuXuVVACEw/s1600/h3.jpg)

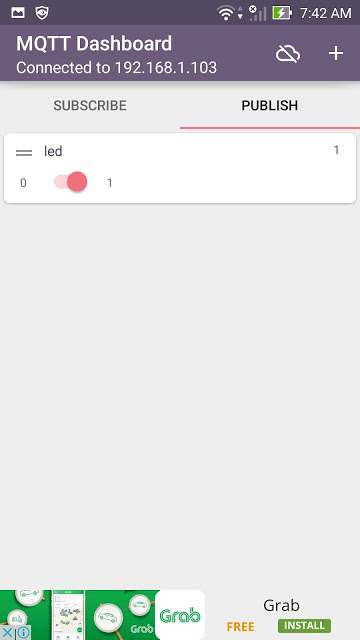
**Figure: Choose Subscribe tab and create topic temp**

[](https://3.bp.blogspot.com/-XCCOy_ugNyM/WSRbktIxniI/AAAAAAAAD7k/swNxuF7VI_QHroz0swQRYPUoTZxnpvwRgCEw/s1600/h4.jpg)

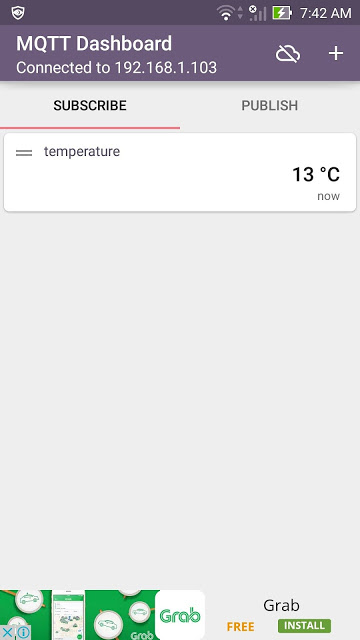
**Figure: Choose Publish tab to create a Switch for toogling LED**

[](https://4.bp.blogspot.com/-QO8ldO91-L0/WSRbk9UcnrI/AAAAAAAAD7k/8XAv2rw9FtMQltn24gWt1iuYfpVzpLwMwCEw/s1600/h5.jpg)

**Figure: Fill topic led for Switch**

[](https://3.bp.blogspot.com/-oLd0P6J6mcM/WSRblH7w64I/AAAAAAAAD7k/HWuqLHim5i8id1hjp-DSrNqTjQDdMO1cwCEw/s1600/h6.jpg)

**Figure: After finishing, here is the GUI for Publish tab, one Switch**

[](https://1.bp.blogspot.com/-tgZG4-JKuAY/WSRbluxj_pI/AAAAAAAAD7k/BzrFwzu2issWekNX8wZ5ePS4hgsM3w3KgCEw/s1600/h7.jpg)

**Figure: You can see smart phone received temp topic**