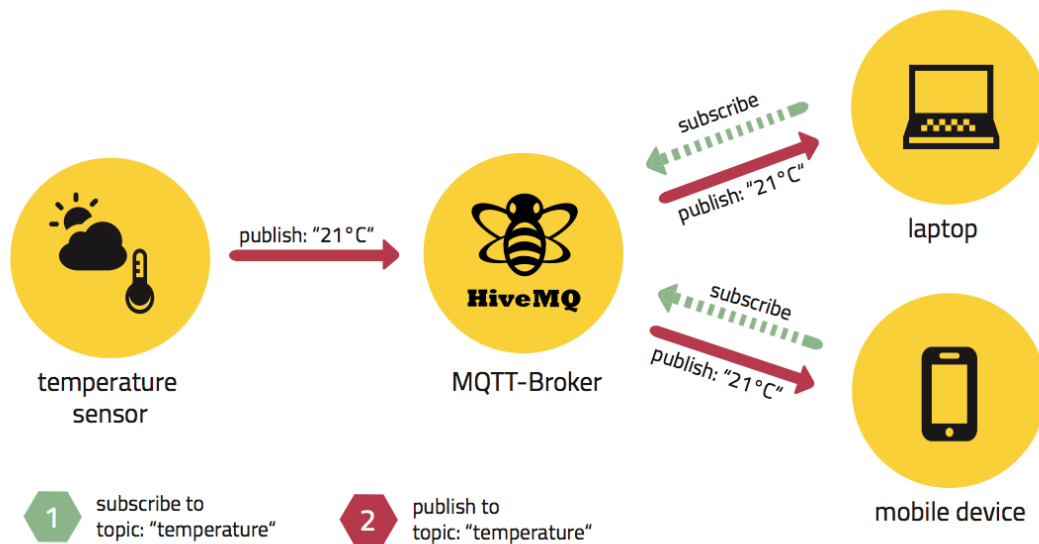


What is MQTT?

http://www.eclipse.org/community/eclipse_newsletter/2014/october/article2.php

Visit above web site to know more about MQTT.



The above figure shows the system architecture. The messages are handled by the MQTT broker. You need a program at the sensor to publish data, and another program as a client to process the data. The messages are forwarded according to the “TOPIC” field.

You can install mqttbox (win10) to subscribe the message. Let’s take Arduino for example.

Some free mqtt brokers.

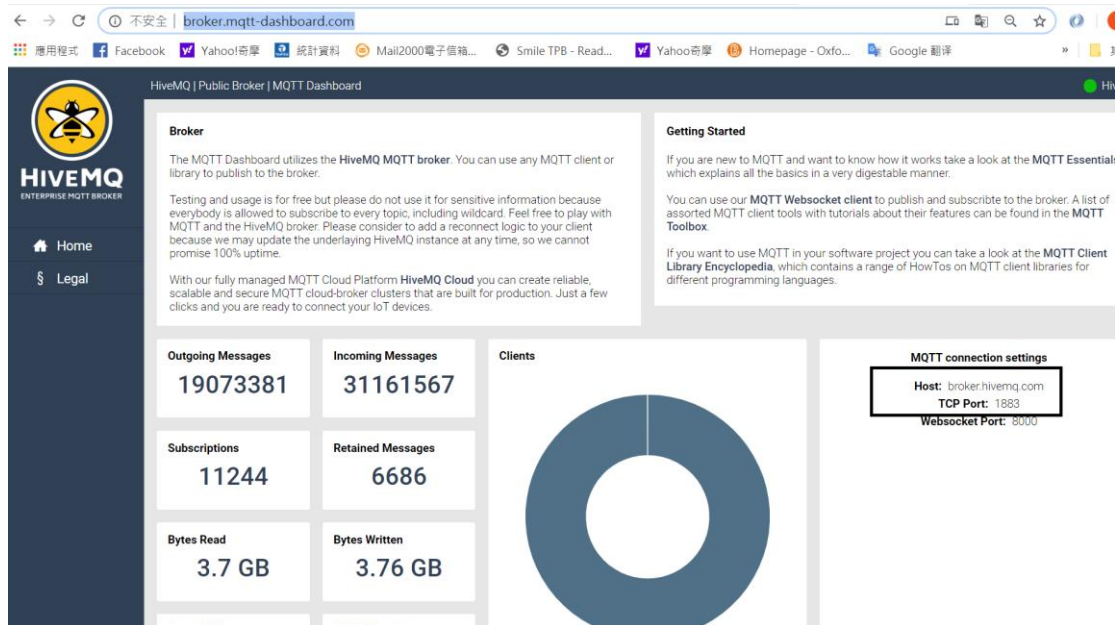
<https://diyprojects.io/8-online-mqtt-brokers-iot-connected-objects-cloud/#.XpZgg8gzbb0>

1. Free MQTT broker.

Visit <http://broker.mqtt-dashboard.com/>

Host: broker.hivemq.com

TCP Port: 1883



2. Install mqttbox

http://workswithweb.com/html/mqttbox/installing_apps.html#install_on_windows

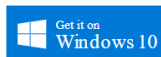
Installing on Windows

You can run MQTTBox apps on Windows in two ways.

1. From Windows App Store (Recommended)
2. Installing from .exe file

Installing from Windows Store (Recommended)

MQTTBox is available on Windows store for free. (Available only from windows 10 anniversary update and above)



Installing from .exe file


Installing MQTTBox on windows is straightforward. [Click here to download MQTTBox-win.exe file](#) and double click on .exe file to install.

Run mqttbox and create a new

MQTTBox

MQTTBox Edit Help


Menu MQTT CLIENT SETTINGS

MQTT Client Name	MQTT Client Id
free	d3edaf25-0c88-4839-b6cb-441a42 
Protocol	Host
mqtt / tcp	broker.mqtt-dashboard.com:1883
Username	Password
Username	Password
Reconnect Period (milliseconds)	Connect Timeout (milliseconds)
1000	30000
Will - Topic	Will - QoS
Will - Topic	0 - Almost Once

You can see that MQTT connected the broker.

MQTTBox

MQTTBox Edit Help

Menu **Connected** Add publisher Add subscriber 

free - mqtt://broker.mqtt-dashboard.com:1883

Topic to publish	Topic to subscribe
Topic to publish	Topic to subscribe
QoS	QoS
0 - Almost Once	0 - Almost Once
Retain <input type="checkbox"/>	
Payload Type	
Strings / JSON / XML / Characters	
e.g: {'hello':'world'}	
Payload	Subscribe

Try to deliver a message. You can try any topic name, but remember to use as special as possible. Click “subscribe” to subscribe the topic you just type.

MQTTBox

MQTTBox Edit Help

Menu

←

Connected

Add publisher

Add subscriber

free - mqtt://broker.mqtt-dashboard.com:1883

Topic to publish

mytest1234

QoS

0 - Almost Once

Retain

Payload Type

Strings / JSON / XML / Characters

e.g: {'hello':'world'}

Payload

Topic to subscribe

mytest1234

QoS

0 - Almost Once

Subscribe

click

Send a message.

MQTTBox

MQTTBox Edit Help

Menu

←

Connected

Add publisher

Add subscriber

free - mqtt://broker.mqtt-dashboard.com:1883

Topic to publish

mytest1234

QoS

0 - Almost Once

Retain

Payload Type

Strings / JSON / XML / Characters

e.g: {'hello':'world'}

Payload

{'hello':'world'}

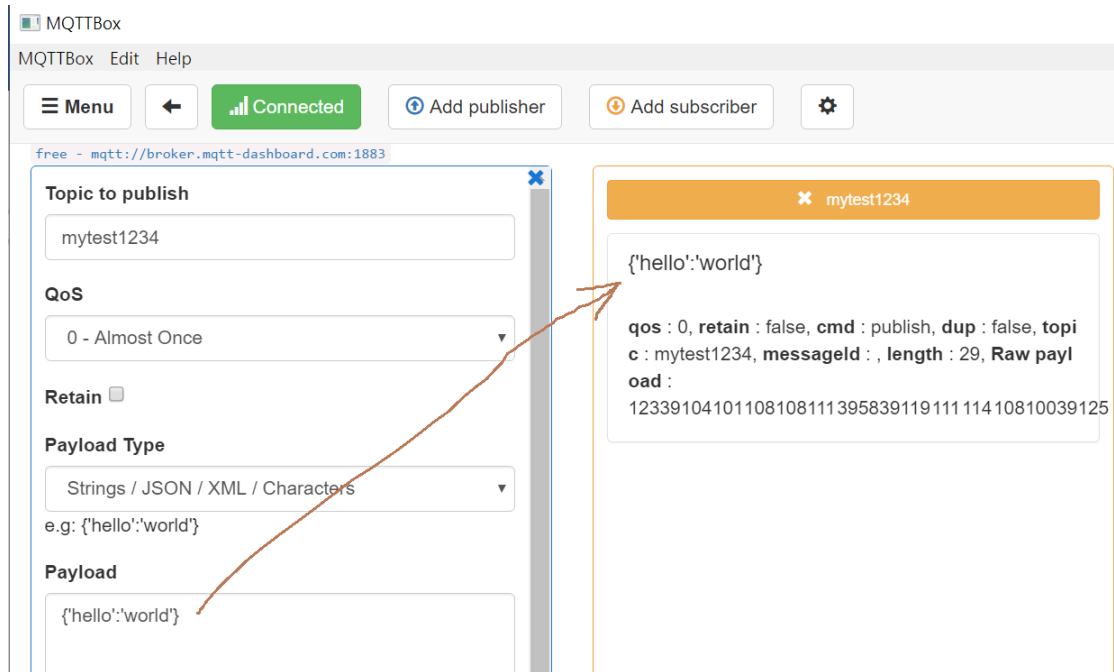
type a message here

Publish

click publish

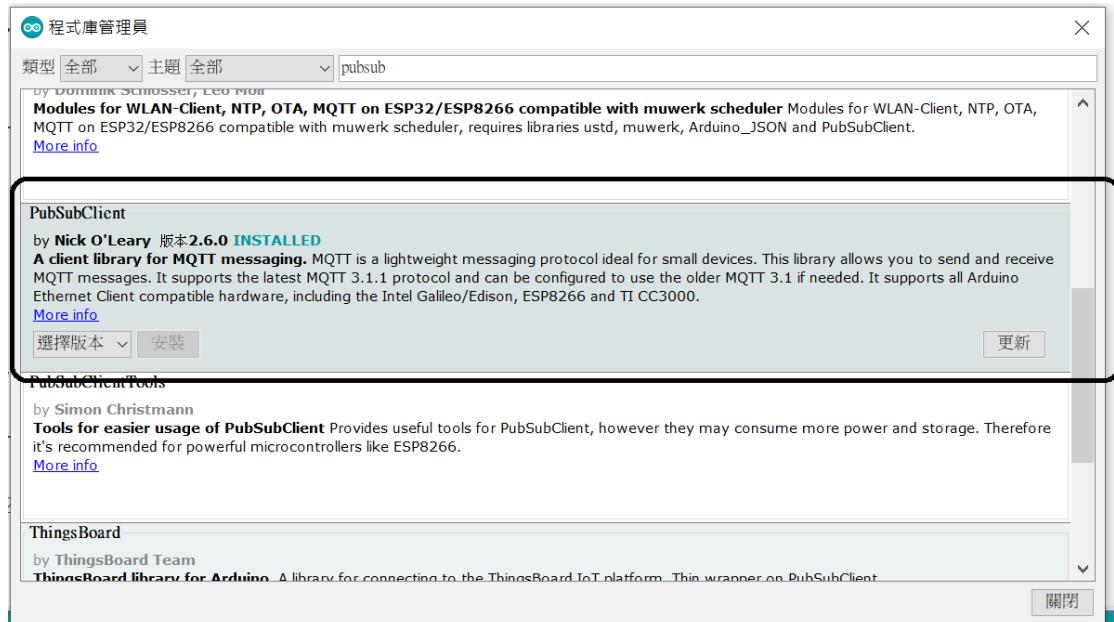
mytest1234

You will see that message returns at left.



In this experiment, we use mqttbox to publish a message with topic *mytest1234*. We can see the message at the right hand side because mqttbox also subscribe the same topic.

3. Install Arduino library, PubSubClient.



Let take mqtt_8266 for example. This an example file of pubsubclient.

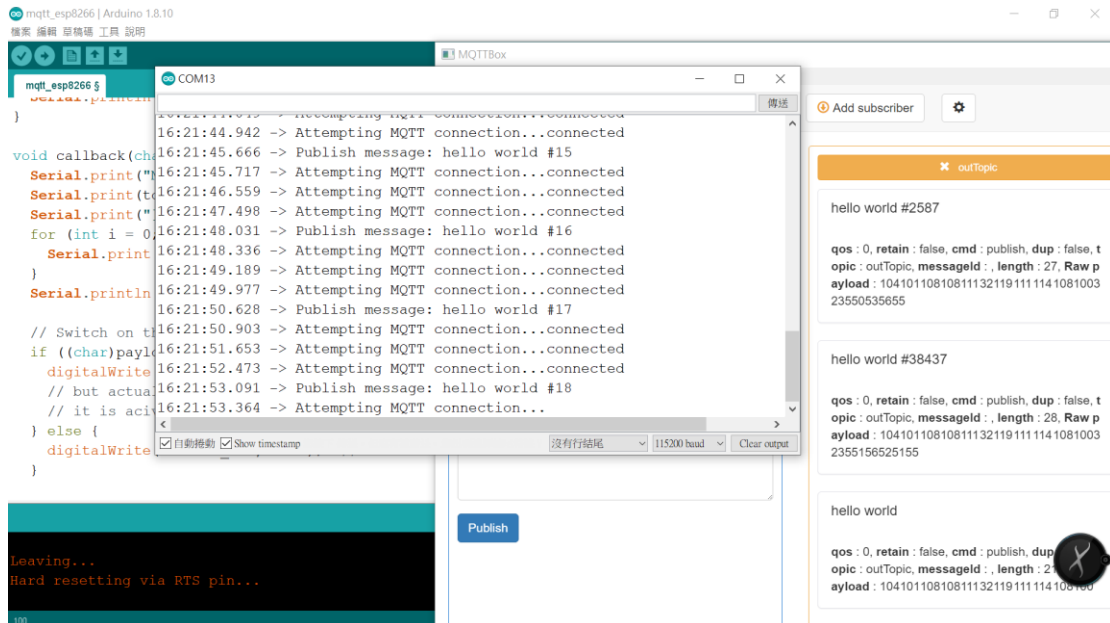
- Modify ssid, password, and mqtt_server.

The default topic to publish: outTopic

The default topic to be subscribed: inTopic

You can modify the them.

Open mqttbox and subscribe "outTopic". You will see many messages arrive.



This is because many other users run this example at the same time. If you modify the topic, you can observe your own messages only. I just append "ncnu" with the old topics.

***** Important remark *****

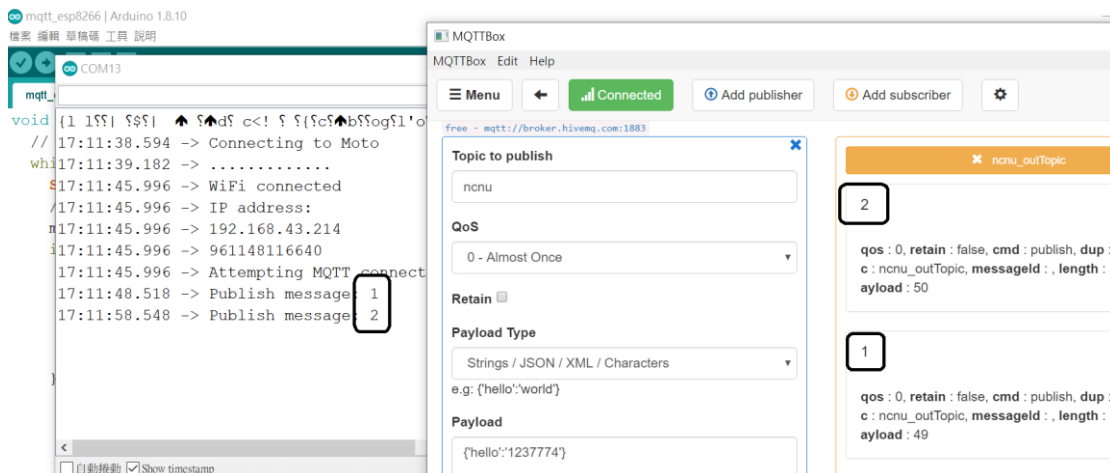
client.connect("ESP8266Client")

This example uses "ESP8266Client" as the session ID. That's why we see that 8266 always need to reconnect to the broker because many other users use the same ID.

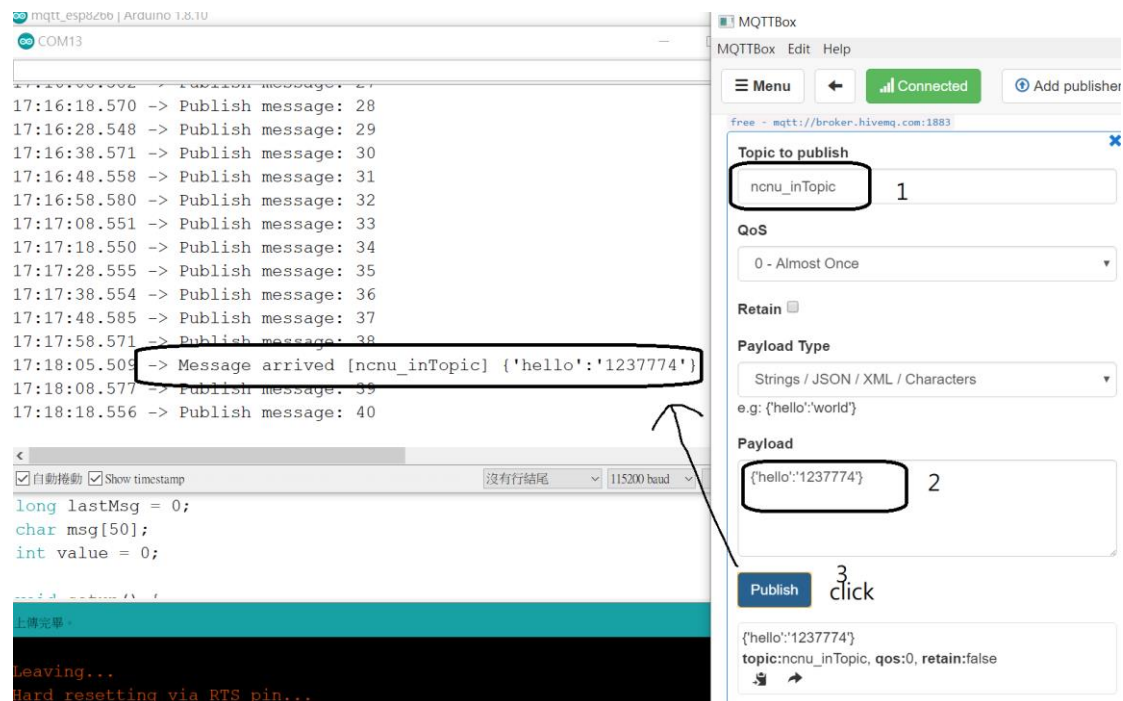
The best way is to use MAC address of 8266 as the ID. So we can change it to client.connect(msg)

where msg is a string containing MAC address.

Then we can have a clean result. You won't see the reconnecting message and mqttbox won't get any strange messages.



So far, we know how to send a message from a device to the broker. Next step is to know the way to receive a message from the broker. That message can be generated by mqttbox.



You can see the message displayed in the terminal. The message is handled by the *callback* function. You can implement your own task to process the message.

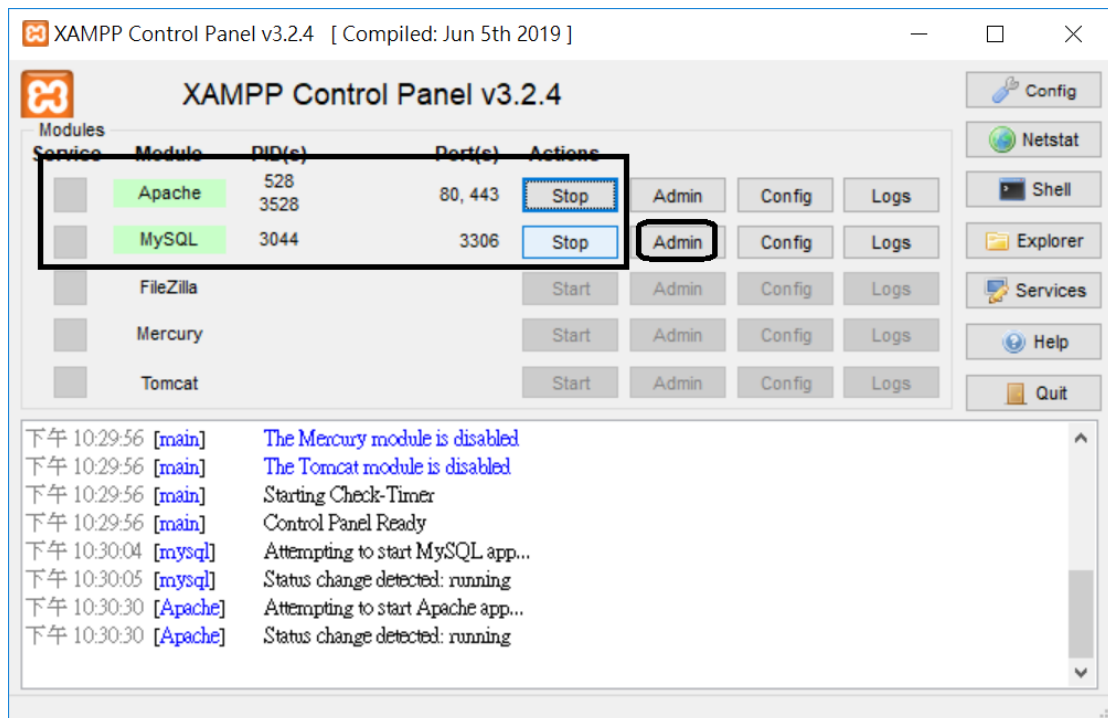
```
void callback(char* topic, byte* payload, unsigned int length)
```

**** important remark ****

The message is handled by "client.loop();" We need to put this inside the loop of Arduino. Remember don't put a LONG busy waiting in your loop. Otherwise, the response time of callback will be very long.

4. XAMPP: https://www.apachefriends.org/zh_tw/download.html

Install it for database. Just install xampp.



Start the two service and click mysql admin to open phpMyAdmin. Search web pages to learn how to create tables.

TEMP.sql: create a table called temp to store temperature values.

TEMP_CHECK.sql: create a table called temp_check to store query information which can be used as footprint tracking.

5. Install python. I use python 3.7.
<https://www.python.org/downloads/windows/>
6. Information flow of auto-thermometer
 Raspberry pi publish (submit.py) → broker → temp_submit.py (store data to “temp” table.
7. Information flow of card screener
 WeMos mini publish () → broker → temp_ask.py (store data to “temp_check” table and query “temp” table.
 Temp_ask.py publish → broker → WeMos mini sub (trigger callback function)
8. temp_submit.py
 Three other files are required.
 - (a) config.txt contains the system configuration.
 - (b) location.txt: we have to interact with the university web server which names

each station with a particular id. The file records the corresponding name.

```
{  
  "L00001":"build1",  
  "L00002":"build2"  
}
```

(c) mapping.txt is used to indicate the device (either auto-thermometer or card screener) is associated with a location id.

```
{  
  "5002914f7d50":"L00001",  
  "b827ebd658b0":"L00002"  
}
```

Let's publish a message showed below.

The image shows a web-based MQTT publish interface. It contains several input fields and a button. The 'Topic to publish' field is highlighted with a red border and contains the text 'NCNU_TEMP'. Below it is a 'QoS' dropdown menu set to '0 - Almost Once'. There is a 'Retain' checkbox which is unchecked. The 'Payload Type' dropdown is set to 'Strings / JSON / XML / Characters'. Below this is an example payload: 'e.g: {'hello':'world'}'. The 'Payload' field is also highlighted with a red border and contains a JSON object: '{"mac":"b827ebd658b0","id":"123451570", "temperature":"36"}'. At the bottom is a blue 'Publish' button.

Topic to publish
NCNU_TEMP
QoS
0 - Almost Once
Retain <input type="checkbox"/>
Payload Type
Strings / JSON / XML / Characters
e.g: {'hello':'world'}
Payload
{"mac":"b827ebd658b0","id":"123451570", "temperature":"36"}
Publish

A mqtt message arrives and a sql insertion command is issued.

```
C:\Users\kuo>python temp_submit.py
Connected with result code 0
2020-04-17 21:48:30:NCNU_TEMP b'{"mac":"b827ebd658b0","id":"123451570", "temperature":"36"}'
1
INSERT INTO TEMP (`mac`,`station`,`id`,`temperature`) VALUES ('b827ebd658b0','build1','123451570', '36')
```

Then, we can see a new record shows in database.

顯示第 0 - 1 列 (總計 2 筆, 查詢用了 0.0007 秒。)

SELECT * FROM `temp`

效能分析 [行內編輯] [編輯] [SQL 語句分析] [建立 PHP]

全部顯示 | 資料列數: 25 | 篩選資料列: 搜尋此資料表 | 依主鍵排序: 無

	sequence	mac	station	id	name	number	temperature	updatetime
	PK	mac address	位置	CARD ID	名字	學號工號	體溫	上傳時間
<input type="checkbox"/> 編輯 <input type="checkbox"/> 複製 <input type="checkbox"/> 刪除	8444	b827ebd658b0	build1	123451570	NULL	NULL	36	2020-04-17 21:48:31
<input type="checkbox"/> 編輯 <input type="checkbox"/> 複製 <input type="checkbox"/> 刪除	8445	b827ebd658b0	build2	123451570	NULL	NULL	36	2020-04-17 22:14:46

9. temp_ask.py

A card screener (5002914f7d50) sends a request by MQTT.

The screenshot shows the MQTTBox interface. On the left, the 'Topic to publish' is set to 'NCNU_TEMP_CHECK'. The 'Payload' field contains the JSON object: {"mac":"5002914f7d50","id":"123451570"}. On the right, a message is received on the topic 'NCNU_TEMP_CHECK/5002914f7d50'. The message is titled 'message returned by temp_ask.py' and contains the same JSON object in the 'Raw payload' field.

temp_ask.py handles the mqtt message and query the database ("temp" table).

Then it stores this event in "temp_check" table.

```
C:\Users\kuo>python temp_ask.py
Connected with result code 0
2020-04-17T22:24:29.582693:NCNU_TEMP_CHECK b'{"mac":"5002914f7d50","id":"123451570"}'
build1
SELECT * FROM `TEMP` WHERE id =123451570 and updatetime >'2020-04-17 00:00:00'
INSERT INTO TEMP_CHECK (`mac`,`station`,`id`,`result`) VALUES ('5002914f7d50', 'build1','123451570','2')
2020-04-17T22:25:06.754880:NCNU_TEMP_CHECK b'{"mac":"5002914f7d50","id":"123451570"}'
build1
SELECT * FROM `TEMP` WHERE id =123451570 and updatetime >'2020-04-17 00:00:00'
INSERT INTO TEMP_CHECK (`mac`,`station`,`id`,`result`) VALUES ('5002914f7d50', 'build1','123451570','2')
```

In the temp_check table, a new entry is created. We can see that this query is from the station named "build1".

✓ 顯示第 0 - 1 列 (總計 2 筆, 查詢用了 0.0005 秒。)

`SELECT * FROM `temp_check``

☐ 效能分析 [\[行內編輯\]](#) [\[編輯\]](#) [\[SQL 語法\]](#)

☐ 全部顯示 | 資料列數: 篩選資料列: 依主鍵排序:

選項

		sequence PK	mac mac address	station station number	id CARD ID	result 查詢結果	updatetime 上傳時間
<input type="checkbox"/>	編輯 複製 刪除	447	5002914f7d50	build1	123451570	2	2020-04-17 22:24:30
<input type="checkbox"/>	編輯 複製 刪除	448	5002914f7d50	build1	123451570	2	2020-04-17 22:25:07

10.