



第12章 MQTT通訊協定： 實作手機App遠端監控

- 12-1 認識MQTT通訊協定
- 12-2 MQTT代理人和客戶端
- 12-3 使用Adafruit.IO的MQTT代理人
- 12-4 使用MQTT遠端控制LED
- 12-5 整合應用：使用MQTT上傳資料至物聯網平台





12-1 認識MQTT通訊協定

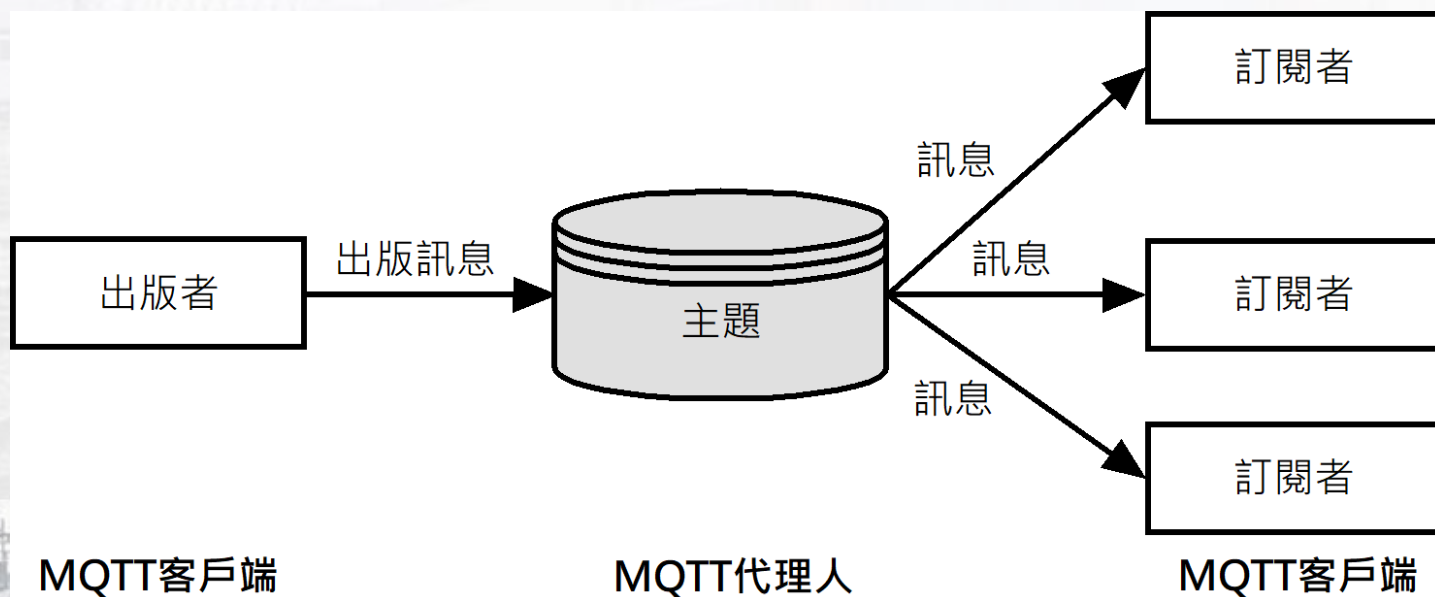
- MQTT（Message Queuing Telemetry Transport）是OASIS標準的一種訊息通訊協定（Message Protocol），這是架構在TCP/IP通訊協定，針對機器對機器（Machine-to-machine，M2M）的輕量級通訊協定。
- MQTT可以在低頻寬網路和高延遲IoT裝置來進行資料交換，特別適用在IoT物聯網這些記憶體不足且效能較差的微控制器開發板。





12-1 認識MQTT通訊協定

- 基本上，MQTT是使用「出版和訂閱模型」（Publish/Subscribe Model）來進行訊息的雙向資料交換，如下圖所示：





12-1 認識MQTT通訊協定

- 所有MQTT客戶端都需要連線MQTT代理人（MQTT **Broker**）才能出版指定主題（**Topic**）的訊息，其扮演的角色是出版者和訂閱者（也可以同時扮演出版者和訂閱者），如下所示：
 - 出版者（**Publisher**）：MQTT客戶端並不需要事先訂閱主題，就可以針對指定MQTT主題（**Topic**）來出版訊息，作為出版者。
 - 訂閱者（**Subscriber**）：每一個MQTT客戶端都可以訂閱指定主題作為訂閱者，當有出版者針對此主題出版訊息時，所有訂閱此主題的訂閱者都可以透過MQTT代理人來接收到訊息。如果出版者本身也有訂閱此主題，因為也是訂閱者，所以一樣可以收到訊息。



12-1 認識MQTT通訊協定

MQTT訊息

- MQTT訊息（MQTT Message）是在不同裝置之間交換的資料，傳送的資料可能是命令；也可能是資料。MQTT訊息是標頭、主題和內容所組成，如下圖所示：





12-1 認識MQTT通訊協定

MQTT訊息

- 在MQTT訊息的標頭可以指定是否保留（**Retained**）訊息和服務品質（**Quality of Service, QoS**），如下所示：
 - 保留（**Retained**）：如果選擇保留，MQTT代理人會保存此主題的訊息，如果之後有新的訂閱者，或之前斷線的訂閱者，當重新連線後，都能收到最新一則的保留訊息（請注意！並非全部訊息）。
 - 服務品質（**Quality of Service, QoS**）：可以指定MQTT出版者與代理人，或MQTT代理人與訂閱者之間的訊息傳輸品質。在MQTT定義三種等級的服務品質，如下表所示：

QoS值	說明
0	最多傳送一次（at most once）- 平信
1	至少傳送一次（at least once）- 掛號
2	確實傳送一次（exactly once）- 附回信



12-1 認識MQTT通訊協定

MQTT主題

- MQTT主題（MQTT Topic）是使用「/」主題等級分隔字元來分割字串，如同檔案的目錄結構，這是一種階層結構的名稱，如下圖所示：

sensors/livingroom/temp

主題等級 主題等級





12-2 MQTT代理人和客戶端

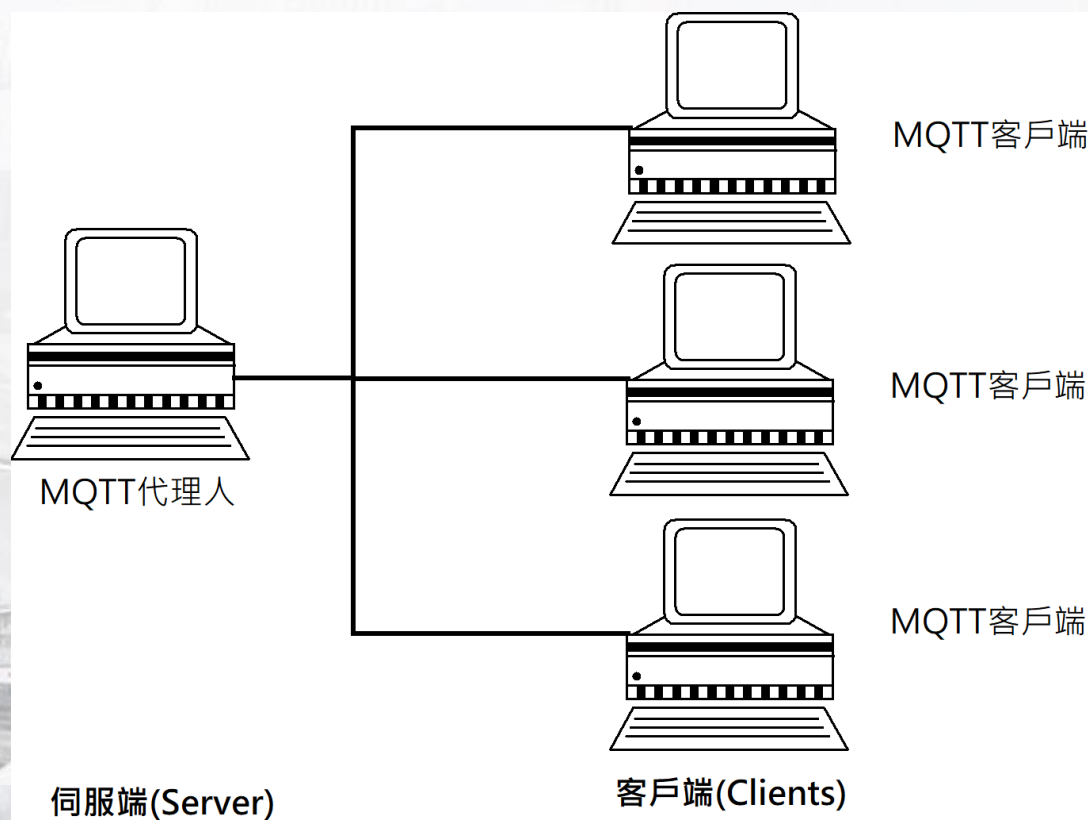
- 12-2-1 MQTT代理人
- 12-2-2 MQTT客戶端





12-2 MQTT代理人和客戶端

- MQTT通訊協定的硬體架構類似主從架構，只是將主從架構的伺服端改成MQTT代理人，而MQTT客戶端就是主從架構的客戶端，如下圖所示：





12-2-1 MQTT代理人

- MQTT代理人負責接收所有出版者的訊息、過濾訊息和決定有哪些訂閱者，並且負責將MQTT客戶端出版的訊息發送至所有訂閱者。MQTT代理人有多家廠商的軟體，和開放原始碼的Mosquitto專案。
- 在實務上，我們可以自行安裝MQTT代理人軟體，或直接使用公開的MQTT代理人。一些常用的公開MQTT代理人，如下所示：
- MQTT公開代理人：HiveMQ Public MQTT Broker
- HiveMQ GmbH公司的MQTT公開代理人，其官方網址如下所示：
- <https://www.hivemq.com/public-mqtt-broker/>



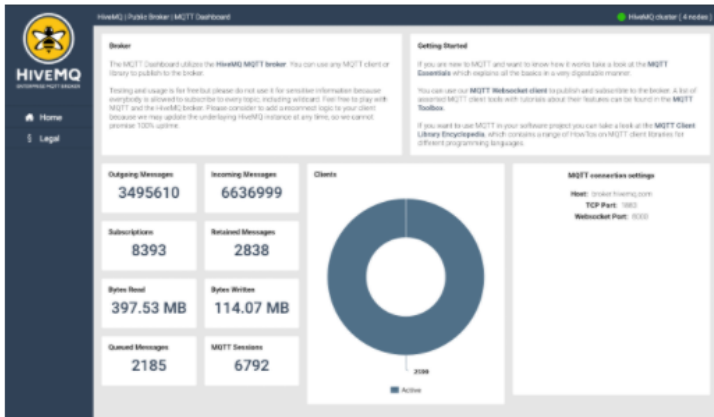
12-2-1 MQTT代理人

MQTT公開代理人：HiveMQ Public MQTT Broker

- HiveMQ GmbH公司的MQTT公開代理人，其官方網址如下所示：

<https://www.hivemq.com/public-mqtt-broker/>

Public MQTT Broker



Public MQTT Broker

Outgoing Messages: 3495610

Incoming Messages: 6636999

Subscriptions: 8393

Retained Messages: 2838

Bytes Read: 397.53 MB

Bytes Written: 114.07 MB

Queued Messages: 2185

MQTT Sessions: 6792

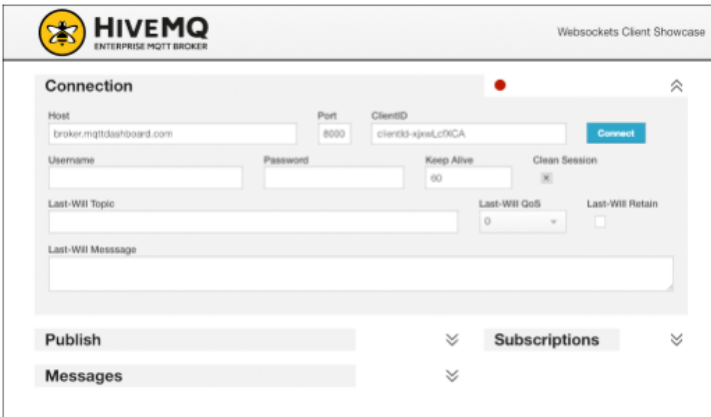
MQTT connection settings

Host: broker.hivemq.com

TCP Port: 1883

Websocket Port: 9000

MQTT Browser Client



MQTT Browser Client

Connection

Host: broker.mqttdashboard.com

Port: 8080

ClientID: client00-kjwLc7KCA

Connect

Username:

Password:

Keep Alive: 60

Clean Session: ☒

Last-Will Topic:

Last-Will QoS: 0

Last-Will Retain: ☐

Last-Will Message:

Publish

Subscriptions

Messages

Our **Public HiveMQ MQTT broker** is open for anyone to use. Feel free to write an MQTT client that connects with this broker. We have a **dashboard** so you can see the amount of traffic on this broker. We also keep a list of **MQTT client libraries** that can be used to connect to HiveMQ.

[Try MQTT Browser Client](#)



12-2-1 MQTT代理人

MQTT公開代理人：Eclipse IoT

- Eclipse IoT的MQTT公開代理人是使用開放原始碼的Mosquitto專案，其官方網址如下所示：

<https://iot.eclipse.org/projects/sandboxes/>

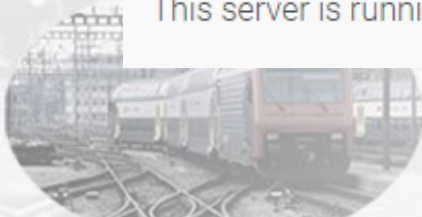
MQTT

You can make use of this MQTT server with client code from the Paho project, the Eclipse MQTT view from Paho, or from one of the other client APIs listed on the MQTT.org downloads page.

Access the server using the hostname `mqtt.eclipse.org` and port `1883`. You can also access the server using encrypted port `8883`.

The encrypted port support TLS v1.2, v1.1 or v1.0 with x509 certificates and requires client support to connect.

This server is running the open source Mosquitto broker in its most recently released version.





12-2-2 MQTT客戶端

- MQTT客戶端（MQTT Client）是訊息的出版者，也是接收者，我們可以使用MQTT客戶端出版指定主題的訊息至MQTT代理人，也可以從MQTT代理人接收訂閱主題的訊息。
- 基本上，任何IoT裝置或電腦上執行的工具程式或函式庫，可以透過網路使用MQTT通訊協定連接MQTT代理人來交換訊息，就是一個MQTT客戶端。例如：在第12-3-2節和第12-4-1節是使用MicroPython程式來建立MQTT客戶端。





12-2-2 MQTT客戶端

MQTT客戶端：HiveMQ Browser Client

- HiveMQ Browser Client瀏覽器是使用Websocket連線的MQTT客戶端工具，我們可以使用網頁介面工具來測試MQTT訊息的傳遞，其URL網址如下所示：

<http://www.hivemq.com/demos/websocket-client/>

The screenshot shows the HiveMQ Websockets Client Showcase interface. At the top left is the HiveMQ logo with the text "HIVEMQ ENTERPRISE MQTT BROKER". To the right is the title "Websockets Client Showcase". Below this is a "Connection" section with a red status indicator and a collapse icon. The connection fields include: Host (broker.mqttdashboard.com), Port (8000), ClientID (clientId-9btJQGuxfN), a "Connect" button, Username, Password, Keep Alive (60), Clean Session (checked), Last-Will Topic, Last-Will QoS (0), Last-Will Retain (unchecked), and Last-Will Message. At the bottom are three expandable sections: "Publish", "Subscriptions", and "Messages".

HIVEMQ
ENTERPRISE MQTT BROKER

Websockets Client Showcase

Connection

Host: broker.mqttdashboard.com Port: 8000 ClientID: clientId-9btJQGuxfN **Connect**

Username: Password: Keep Alive: 60 Clean Session: ☒

Last-Will Topic: Last-Will QoS: 0 Last-Will Retain: ☐

Last-Will Message:

Publish **Subscriptions** **Messages**



12-2-2 MQTT客戶端

MQTT客戶端：MQTTLens

- 在Chrome瀏覽器可以安裝MQTTLens客戶端工具來測試MQTT訊息的傳遞。請啟動Chrome瀏覽器輸入MQTTLens關鍵字來搜尋後，可以在應用程式商店看到MQTTLens，如下所示：

https://chrome.google.com/webstore/detail/mqttlens/hemojaaeigabkbcokmlgmdigohjobjm?utm_source=chrome-ntp-launcher





12-2-2 MQTT客戶端

MQTT客戶端：MQTTLens

The image shows two screenshots of the MQTTLens web interface. The top screenshot shows the 'Add a new Connection' dialog box. The bottom screenshot shows the main MQTTLens interface with a connection named 'HiveMQ' selected and the 'Subscribe' section active.

Top Screenshot: Add a new Connection

Connection Details

- Connection name: HiveMQ
- Connection color scheme: (Green bar)
- Hostname: tcp:// broker.hivemq.com
- Port: 1883
- Client ID: (Empty)

Bottom Screenshot: MQTTLens Main Interface

Connections + ^ | <

- HiveMQ (Green bar)

Connection: HiveMQ

Subscribe

sensors/livingroom/temp

Publish

topic

Subscription Options:

- 0 - at most once
- 0 - at most once (Selected)
- 1 - at least once
- 2 - exactly once

Buttons: SUBSCRIBE, PUBLISH

Version: 0.0.14



12-3 使用Adafruit.IO的MQTT代理人

- 12-3-1 新增名為lights的FEED
- 12-3-2 使用Adafruit.IO的MQTT代理人





12-3-1 新增名為lights的FEED

- 請繼續第11-3節在Adafruit.IO新增名為lights的FEED，可以看到新增的FEED，lights是Feed Name和Key。

hueyanchen2014 > Feeds

[+ New Feed](#) [+ New Group](#)

Default [+ New Feed](#) [Group Settings](#)

Feed Name	Key	Last value	Recorded
<input type="checkbox"/> Humidity	humidity	87	about 21 hours ago
<input type="checkbox"/> lights	lights		less than a minut...
<input type="checkbox"/> Temperature	temperature	21	about 21 hours ago



12-3-2 使用Adafruit.IO的MQTT代理人

- 在Adafruit.IO新增FEED後，我們就可以建立MicroPython程式的MQTT客戶端來連線MQTT代理人，並且訂閱主題和發送訊息，然後在接收訊息後進行相關處理。
Adafruit.IO MQTT代理人的連線相關資訊，如下圖所示：

MQTT Connection Details

We *strongly* recommend connecting using SSL (Port 8883) if your client allows it. Port 443 is for MQTT-over-Websockets clients which generally run in browsers, like Eclipse Paho, HiveMQ Websockets, or MQTTJS.

Host	io.adafruit.com
Secure (SSL) Port	8883
Insecure Port	1883
**MQTT over Websocket	443
Username	Your Adafruit IO Username
Password	Your Adafruit IO Key



12-3-2 使用Adafruit.IO的MQTT代理人

- Adafruit.IO的MQTT主題有專屬格式，如下所示：
<username>/feeds/<feed key>
- 上述主題是Adafruit.IO使用者名稱開頭，中間是"/feeds/"，最後是**FEED**名稱。





12-3-2 使用Adafruit.IO的MQTT代理人

使用Adafruit.IO的MQTT代理人來控制LED：ch12-3-2.py

- 在MicroPython程式需要匯入MQTTClient類別來建立MQTT客戶端，以便讓MicroPython程式可以連線MQTT代理人，如下所示：

```
from machine import Pin
```

```
from umqtt.simple import MQTTClient
```

```
import utime, xtools
```

```
xtools.connect_wifi_led()
```

```
ledG = Pin(12, Pin.OUT)
```

```
ledG.value(0)
```





12-3-2 使用Adafruit.IO的MQTT代理人

使用Adafruit.IO的MQTT代理人來控制LED：ch12-3-2.py

```
ADAFRUIT_IO_USERNAME = "hueyanchen2014"  
ADAFRUIT_IO_KEY = "<AIO KEY>"  
FEED = "lights"
```

```
# MQTT 客戶端
```

```
client = MQTTClient (  
    client_id = xtools.get_id(),  
    server = "io.adafruit.com",  
    user = ADAFRUIT_IO_USERNAME,  
    password = ADAFRUIT_IO_KEY,  
    ssl = False,  
)
```



12-3-2 使用Adafruit.IO的MQTT代理人

使用Adafruit.IO的MQTT代理人來控制LED：ch12-3-2.py

```
def sub_cb(topic, msg):  
    global ledG  
    print("收到訊息: ", msg.decode())  
    if msg.decode() == "ON":  
        ledG.value(1)  
    if msg.decode() == "OFF":  
        ledG.value(0)  
  
client.set_callback(sub_cb) # 指定回撥函數來接收訊息  
client.connect()           # 連線  
topic = ADAFRUIT_IO_USERNAME + "/feeds/" + FEED  
print(topic)  
client.subscribe(topic)    # 訂閱主題
```




12-3-2 使用Adafruit.IO的MQTT代理人

使用Adafruit.IO的MQTT代理人來控制LED：ch12-3-2.py

while True:

```
    print("送出訊息: ON")
```

```
    client.publish(topic, "ON")
```

```
    utime.sleep(2)
```

```
    client.check_msg()
```

```
    print("送出訊息: OFF")
```

```
    client.publish(topic, "OFF")
```

```
    utime.sleep(2)
```

```
    client.check_msg()
```

- 其執行結果可以看到間隔2秒鐘來閃爍綠色LED。





12-4 使用MQTT遠端控制LED

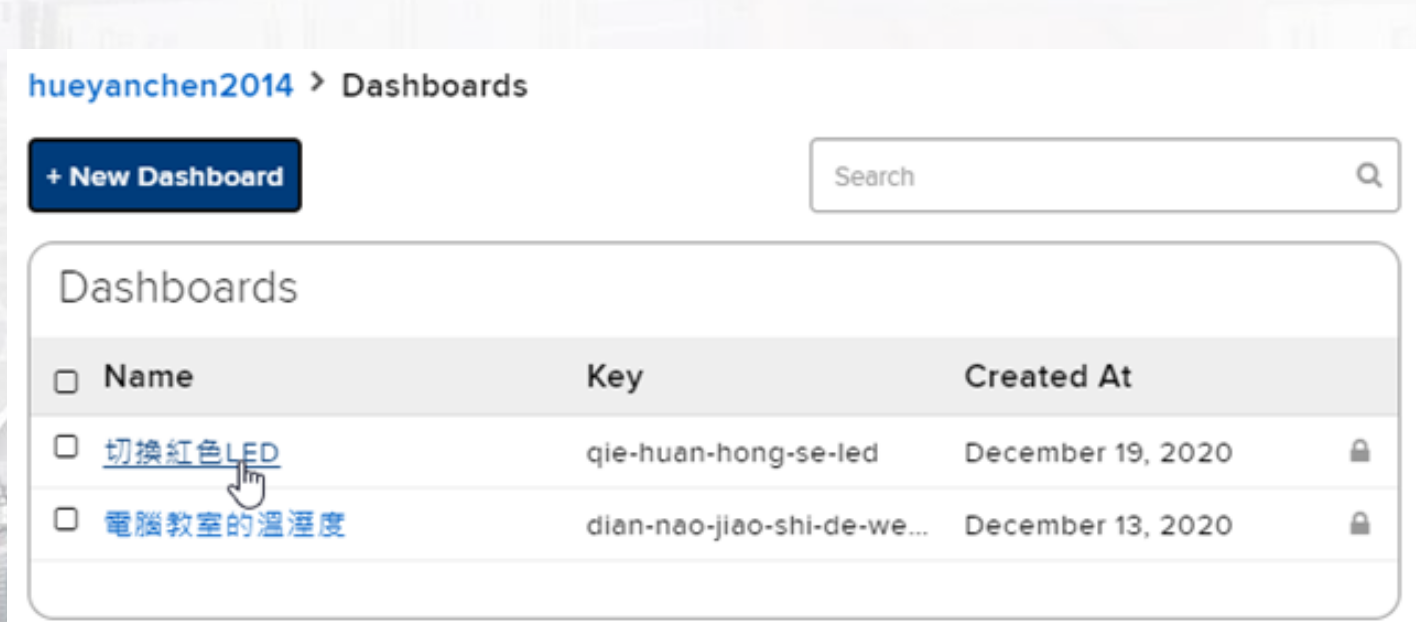
- 12-4-1 使用Adafruit.IO儀表板遠端控制LED
- 12-4-2 使用Android手機App遠端控制LED





12-4-1 使用Adafruit.IO儀表板遠端控制LED 在Adafruit.IO建立儀表板和新增切換按鈕

- 在Adafruit.IO新增名為lights的FEED後，就可以新增儀表板來新增切換按鈕（Toggle Button）區塊，建立遠端控制所需的使用介面。
- 新增儀表板: 切換紅色LED





12-4-1 使用Adafruit.IO儀表板遠端控制LED

在Adafruit.IO建立儀表板和新增切換按鈕

- 新增切換按鈕（Toggle Button）區塊：

Block settings

In this final step, you can give your block a title and see a preview of how it will look. Customize the look and feel of your block with the remaining settings. When you are ready, click the "Create Block" button to send it to your dashboard.

Block Title (optional)

Button On Text


Button Off Text

Block Preview

RED LED

hueyanchen2014 > Dashboards > 切換紅色LED

RED LED



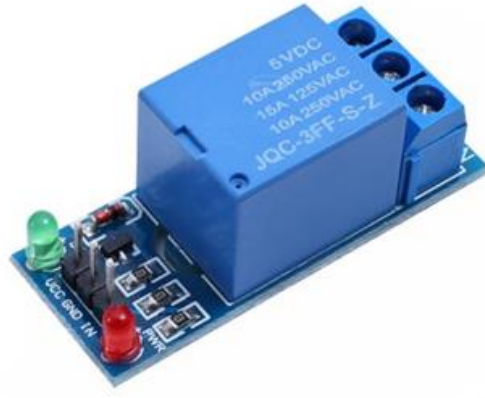




12-4-1 使用Adafruit.IO儀表板遠端控制LED

使用MicroPython程式遠端控制LED：ch12-4-1.py

- 繼電器（**Relay**）是使用數位輸出來進行控制，這是使用較小電流來控制較大電流的「自動開關」，如下圖所示：



- 上述圖例是繼電器，其數位輸出：**0**是打開；**1**是關閉。我們準備使用**LED**（數位輸出）來模擬繼電器的使用，假設**LED**就是使用繼電器連接控制的燈泡，我們可以在Adafruit.IO儀表板，使用切換開關來遠端點亮或熄滅紅色LED。



12-4-1 使用Adafruit.IO儀表板遠端控制LED

使用MicroPython程式遠端控制LED：ch12-4-1.py

```
from machine import Pin  
from umqtt.simple import MQTTClient  
import utime, xtools
```

```
xtools.connect_wifi_led()  
ledR = Pin(15, Pin.OUT)  
ledR.value(0)
```

```
ADAFRUIT_IO_USERNAME = "hueyanchen2014"  
ADAFRUIT_IO_KEY = "<AIO KEY>"  
FEED = "lights"
```



12-4-1 使用Adafruit.IO儀表板遠端控制LED

使用MicroPython程式遠端控制LED：ch12-4-1.py

MQTT 客戶端

```
client = MQTTClient (  
    client_id = xtools.get_id(),  
    server = "io.adafruit.com",  
    user = ADAFRUIT_IO_USERNAME,  
    password = ADAFRUIT_IO_KEY,  
    ssl = False,  
)
```





12-4-1 使用Adafruit.IO儀表板遠端控制LED

使用MicroPython程式遠端控制LED：ch12-4-1.py

```
def sub_cb(topic, msg):  
    if msg.decode() == "ON":  
        ledR.value(1)  
    elif msg.decode() == "OFF":  
        ledR.value(0)  
    print("收到訊息: ", msg.decode())  
client.set_callback(sub_cb) # 指定回撥函數來接收訊息  
client.connect()           # 連線
```





12-4-1 使用Adafruit.IO儀表板遠端控制LED

使用MicroPython程式遠端控制LED：ch12-4-1.py

```
topic = ADAFRUIT_IO_USERNAME + "/feeds/" + FEED  
print(topic)
```

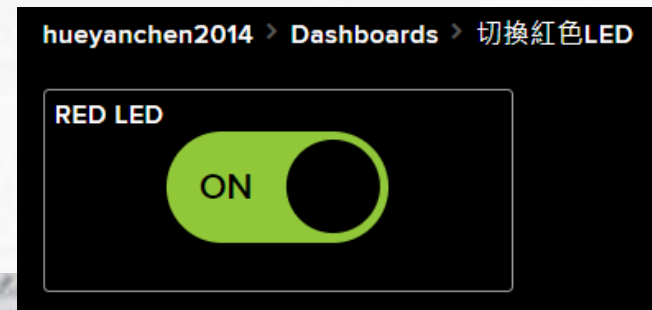
```
client.subscribe(topic)    # 訂閱主題
```

```
while True:
```

```
    client.check_msg()
```

```
    utime.sleep(3)
```

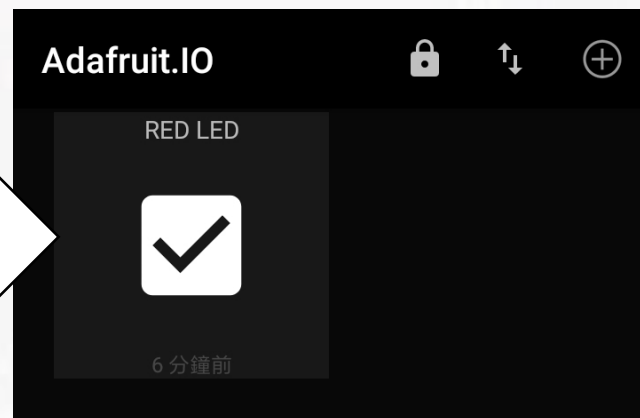
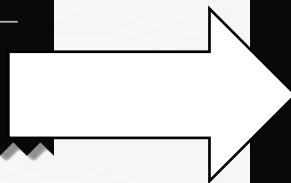
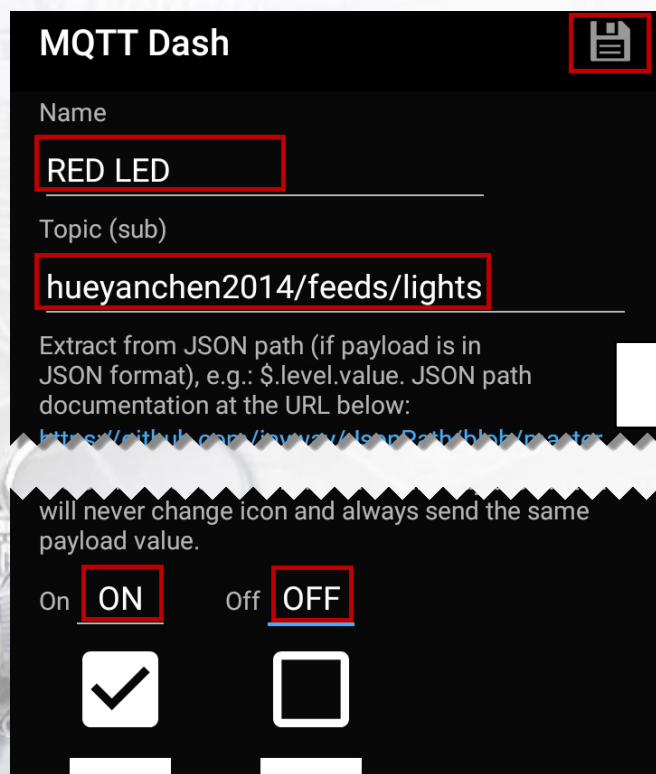
- 我們就可以在儀表板點選切換開關，遠端來控制紅色LED是點亮或熄滅，如下圖所示：





12-4-2 使用Android手機App遠端控制LED

- MQTT Dash是支援MQTT通訊協定的Android App，我們可以在App新增MQTT代理人後，再新增Switch/button元件來遠端控制LED，如下圖所示：





12-5 整合應用：使用MQTT上傳資料至物聯網平台-上傳資料至ThingSpeak：ch12-5.py

- ThingSpeak上傳資料的MQTT主題格式，如下所示：
channels/<CHANNEL_ID>/publish/<API_KEY>
- 上述<CHANNEL_ID>是頻道編號；<API_KEY>是WRITE API金鑰字串。上傳資料就是出版MQTT訊息，訊息格式如下所示：
field1=<temperature>&field2=<humidity>
- 上述<temperature>是Field 1欄位值，<humidity>是Field 2欄位值。





12-5 整合應用：使用MQTT上傳資料至物聯網平台-上傳資料至Adafruit.IO：ch12-5a.py

- Adafruit.IO上傳資料至FEED的MQTT主題格式，如下所示：

<ADAFRUIT_IO_USERNAME>/feeds/<FEED1>

<ADAFRUIT_IO_USERNAME>/feeds/<FEED2>

