

iTemperature

此章節解說智能溫度計(iTemperature)的實作方法，應用於環境溫溼度感測並回報資訊給中控台。

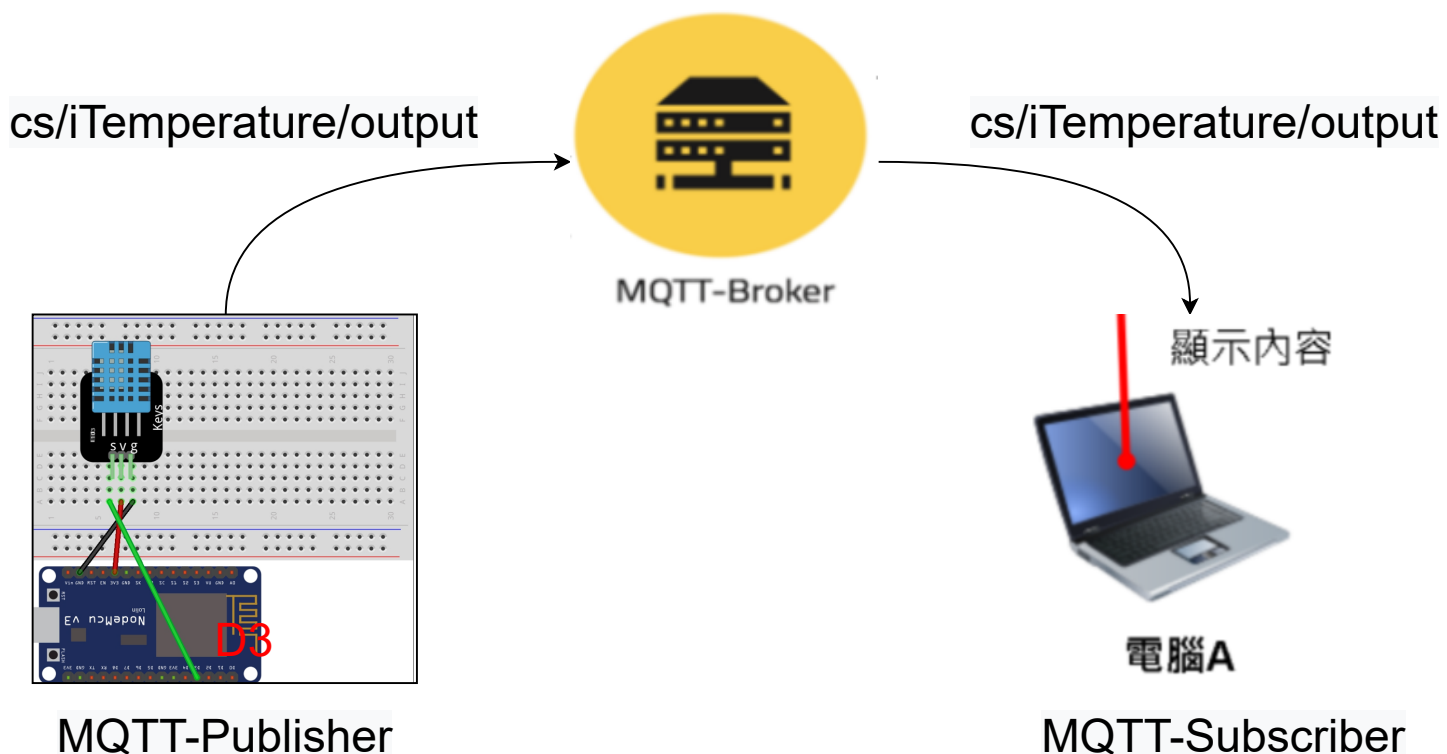
如果對於建置環境不了解，先參考「NodeMCU_HelloWorld」章節。

如果對於DHT11不了解，先參考「DHT11」章節。

如果對於MQTT不了解，先參考「NodeMCU_MQTT」章節。

架構圖：

(1) 將 VCC 與 GND接上，DATA訊號接在 D3 Pin 腳



上圖，NodeMCU為Publisher角色，電腦A為Subscriber角色。

NodeMCU發送的topic為cs/iTemperature/output，反之電腦A接收的topic為cs/iTemperature/output

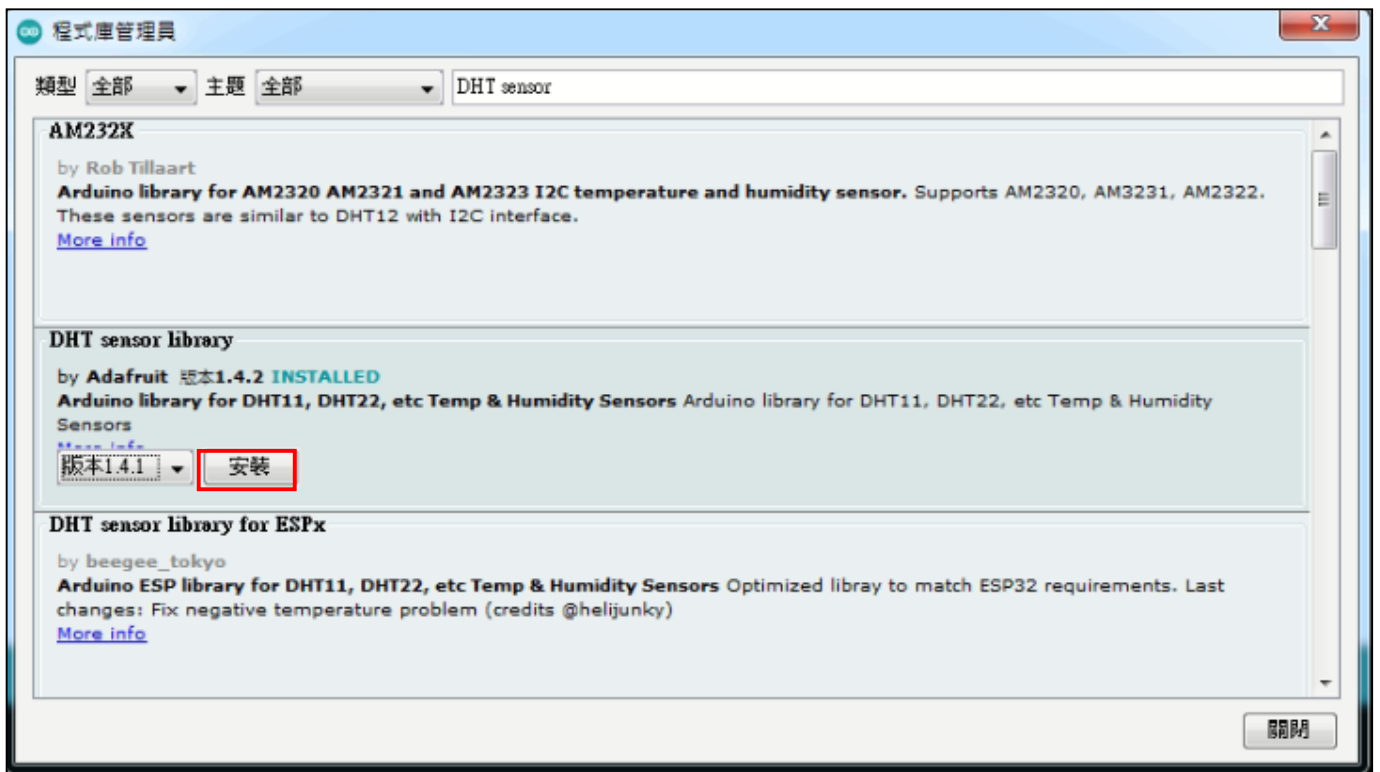
架構介紹：

1. NodeMCU連上WiFi AP(2.4G only)，也連上MQTT-Broker
2. NodeMCU每2秒發送訊息至topic(cs/iTemperature/output)
3. 電腦A連上WiFi AP，也連上MQTT-Broker
4. 電腦A接收topic(cs/iTemperature/output)，知道溫溼度資訊

1. 安裝 DHT 函式庫

在IDE 上方，

工具 -> 管理程式庫... -> 搜尋欄位輸入「DHT sensor」 -> 按下**安裝**



安裝完畢後，直接關閉視窗，安裝時下方有安裝進度條。

2. 編寫草稿碼 -> 上傳至 NodeMCU 開發板

iTemperature | Arduino 1.8.14

檔案 編輯 草稿碼 工具 說明

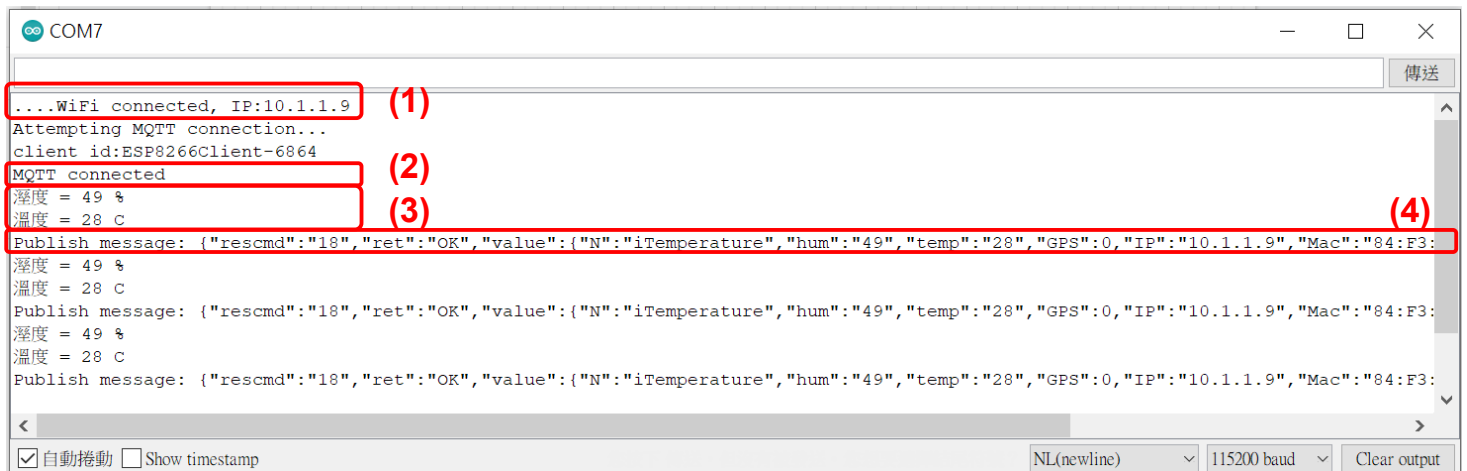
```

7 #include <ESP8266WiFi.h>
8 #include <PubSubClient.h>
9 #include <SimpleDHT.h>
10
11 #define MSG_BUFFER_SIZE (1024)
12
13 //ESP8266 名稱
14 #define DUTNAME "iTemperature"
15
16 //DHT11 (1)
17 int pinDHT11 = D3; //D3 Pin腳 讀取DATA
18 SimpleDHT11 dht11; //dht11 class
19
20 //回報間隔 (2)
21 const int report_interval = 2000;
22
23 //mac string
24 char dut_mac_str[17+1] = {0};
25
26 //WIFI網路的 SSID, 密碼 (3)
27 const char* ssid = "your_wifi_ssid";
28 const char* password = "your_wifi_key";
29
30 //Third-party MQTT Broker Domain Name & Port (4)
31 const char* mqtt_server = "broker.emqx.io";
32 const int mqtt_port = 1883;
33
34 //MQTT topic & message
35 char send_topic[]="cs/"DUTNAME"/output";
36 char msg[MSG_BUFFER_SIZE];
37
38
39 WiFiClient espClient;
40 PubSubClient client(espClient);
```

- (1) 定義DHT11 data pin腳為D3
- (2) 設定回報間隔時間，單位毫秒(ms)
- (3) 設定WiFi SSID與密碼
- (4) 設定MQTT-Broker、Port、及發送的topic

3. 觀看結果

3.1 命令列

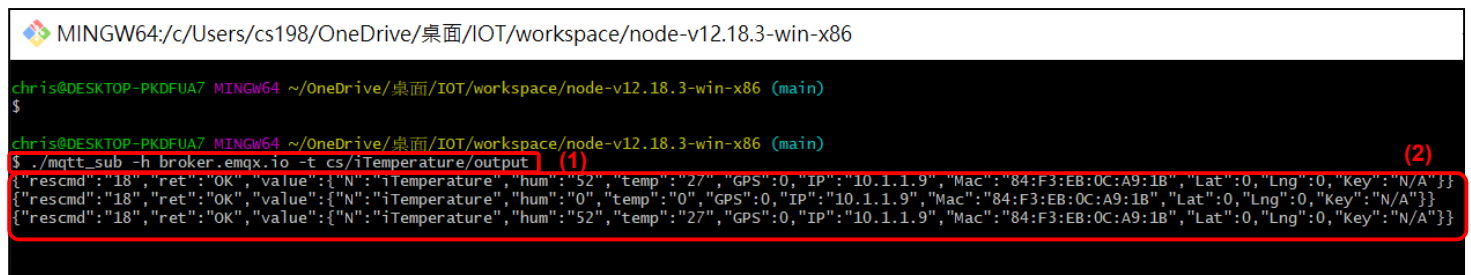


```
COM7
...WiFi connected, IP:10.1.1.9 (1)
Attempting MQTT connection...
client id:ESP8266Client-6864
MQTT connected (2)
溼度 = 49 % (3)
溫度 = 28 C (3)
Publish message: {"rescmd":"18","ret":"OK","value":{"N":"iTemperature","hum":"49","temp":"28","GPS":0,"IP":"10.1.1.9","Mac":"84:F3:EB:0C:A9:1B","Lat":0,"Lng":0,"Key":"N/A"}} (4)
溼度 = 49 %
溫度 = 28 C
Publish message: {"rescmd":"18","ret":"OK","value":{"N":"iTemperature","hum":"49","temp":"28","GPS":0,"IP":"10.1.1.9","Mac":"84:F3:EB:0C:A9:1B","Lat":0,"Lng":0,"Key":"N/A"}}
溼度 = 49 %
溫度 = 28 C
Publish message: {"rescmd":"18","ret":"OK","value":{"N":"iTemperature","hum":"49","temp":"28","GPS":0,"IP":"10.1.1.9","Mac":"84:F3:EB:0C:A9:1B","Lat":0,"Lng":0,"Key":"N/A"}}

☒ 自動捲動 ☐ Show timestamp
NL(newline) 115200 baud Clear output
```

- (1) NodeMCU連上WiFi，且拿到IP
- (2) NodeMCU連上MQTT-Broker
- (3) 每2秒列印溫溼度
- (4) 每2秒發送訊息至topic(cs/iTemperature/output)

3.2 電腦A - Git Bash介面



```
MINGW64:/c/Users/cs198/OneDrive/桌面/IOT/workspace/node-v12.18.3-win-x86
chris@DESKTOP-PKDFUA7 MINGW64 ~/OneDrive/桌面/IOT/workspace/node-v12.18.3-win-x86 (main)
$
chris@DESKTOP-PKDFUA7 MINGW64 ~/OneDrive/桌面/IOT/workspace/node-v12.18.3-win-x86 (main)
$ ./mqtt_sub -h broker.emqx.io -t cs/iTemperature/output (1)
{"rescmd":"18","ret":"OK","value":{"N":"iTemperature","hum":"52","temp":"27","GPS":0,"IP":"10.1.1.9","Mac":"84:F3:EB:0C:A9:1B","Lat":0,"Lng":0,"Key":"N/A"}} (2)
{"rescmd":"18","ret":"OK","value":{"N":"iTemperature","hum":"0","temp":"0","GPS":0,"IP":"10.1.1.9","Mac":"84:F3:EB:0C:A9:1B","Lat":0,"Lng":0,"Key":"N/A"}}
{"rescmd":"18","ret":"OK","value":{"N":"iTemperature","hum":"52","temp":"27","GPS":0,"IP":"10.1.1.9","Mac":"84:F3:EB:0C:A9:1B","Lat":0,"Lng":0,"Key":"N/A"}}
```

- (1) 用Node.js - mqtt 工具接收 topic(cs/iTemperature/output)訊息
- (2) 接收到的訊息內容：
 - 帶有hum欄位，代表溼度
 - 帶有temp欄位，代表溫度