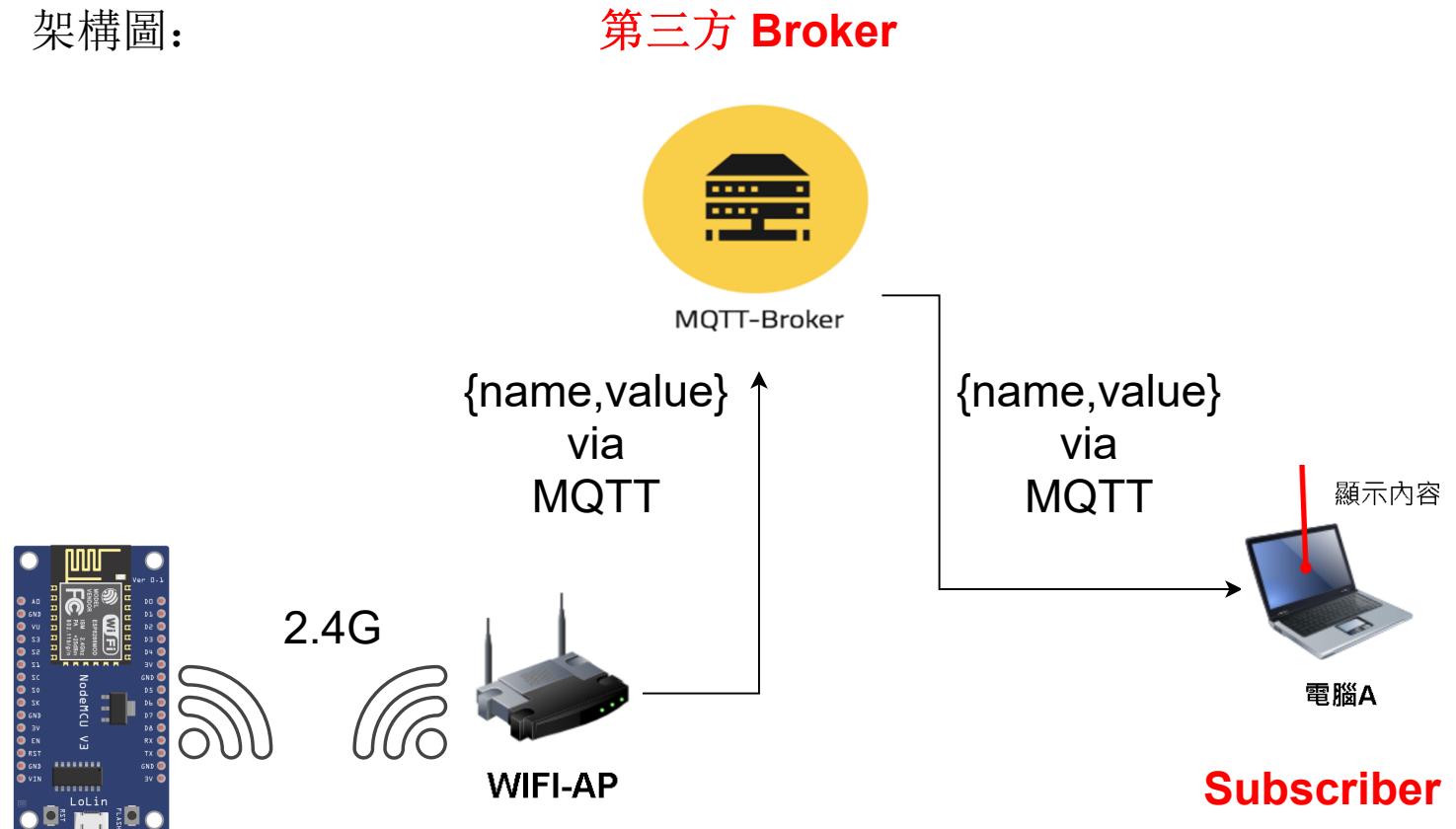


# MQTT

此章節解說如何透過 NodeMCU 與 MQTT 的應用。如果對於建置環境不了解，先參考「[NodeMCU\\_HelloWorld](#)」章節

架構圖：



## Publisher

MQTT 有定義三個角色， Broker、Publish、Subscriber。

上圖，ESP8266 作為 Publisher 將 MQTT Message(name,value) 送給 Broker，電腦A作為 Subscriber，訂閱 ESP8266的 Topic(cs/dutTest/Info)，從Broker收到Message(name,value)

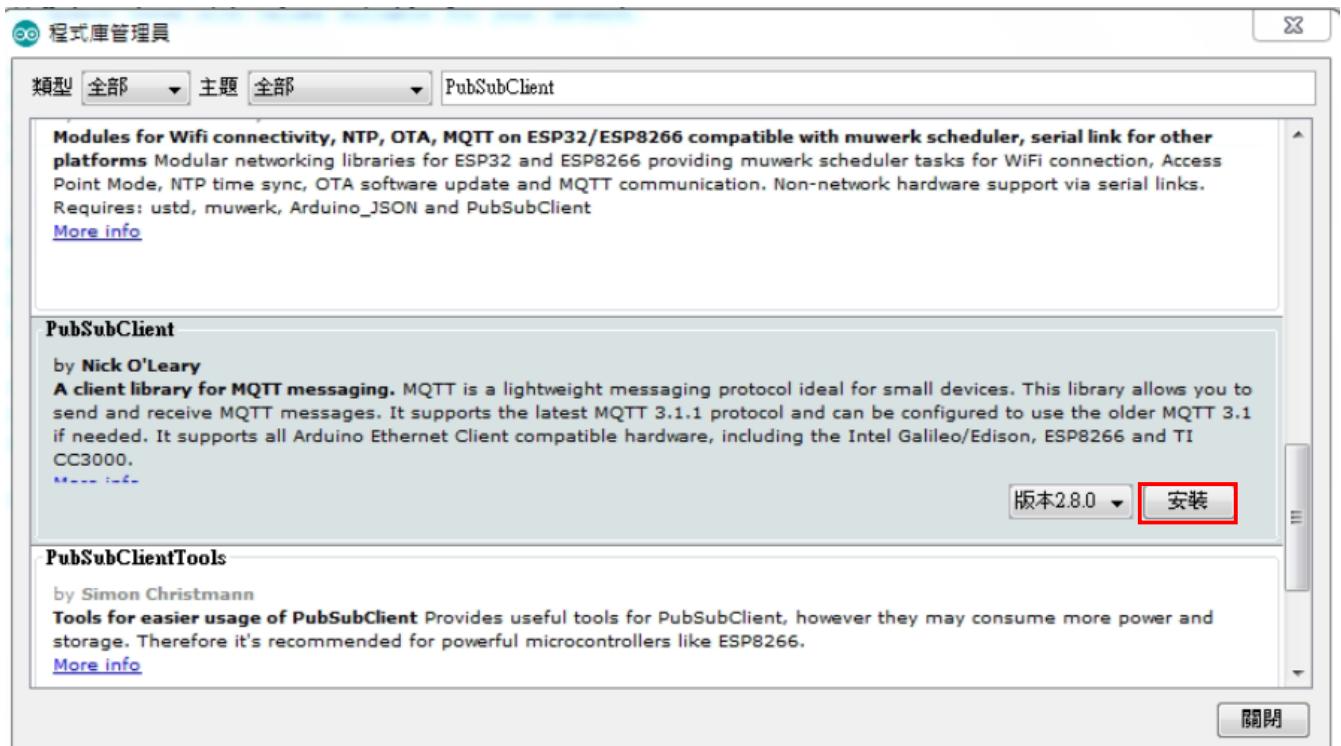
架構介紹：

- 1.ESP8266 連上 WIFI AP(2.4G only)
2. ESP8266 每2秒回報 Message 到 Broker
3. 電腦A 連上 WIFI AP
4. 電腦A開啟 Node-Red，新增Subscriber 並向 Broker 訂閱 Topic(cs/dutTest/Info)，解析 Message 並顯示在網頁上。

# 1. 安裝 PubSubClient 函式庫

在IDE上方，

工具 -> 管理程式庫... -> 搜尋欄位輸入「PubSubClient」 -> 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



The screenshot shows the Arduino IDE interface with the sketch\_may13a file open. The code is a MQTT example for an ESP8266. It includes comments explaining the steps to connect to WiFi and establish an MQTT connection. It defines constants for WiFi SSID and password, MQTT broker details, and MQTT topics. The setup function handles WiFi connection and MQTT reconnection logic, printing status messages to the serial port.

```
sketch_may13a | Arduino 1.8.12
檔案 編輯 草稿碼 工具 說明
sketch_may13a
/*
 * @desc: this is MQTT example, which publish/subscribe DUT's topics
 * @step: 1. connect to WIFI
 *        2. establish mqtt connection, it's publisher and subscriber
 */

#include <ESP8266WiFi.h>
#include <PubSubClient.h>

#define MSG_BUFFER_SIZE (50)

//ESP8266 名稱
#define DUTNAME "MyESP8266"

//WIFI網路的 SSID， 密碼
const char* ssid = "your_wifi_ssid";
const char* password = "your_wifi_password";

//Third-party MQTT Broker Domain Name & Port
const char* mqtt_server = "broker.emqx.io";
const int mqtt_port = 1883;

//MQTT topic & message
char send_topic[]="cs/dutTest/Info";
char msg[MSG_BUFFER_SIZE];

WiFiClient espClient;
PubSubClient client(espClient);

//count time
unsigned long lastMsg = 0;

void setup_wifi() {
    delay(10);

    //Start to connect to WIFI
    printf("Connecting to %s\n", ssid);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        printf(".");
    }

    printf("WiFi connected, IP:%s\n", WiFi.localIP().toString().c_str());
}

void mqttconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        printf("Attempting MQTT connection...\n");

        // Create a random client ID
        String clientId = "ESP8266Client-";
        clientId += String(random(0xffff), HEX);

        // Attempt to connect
        printf("client id:%s\n", clientId.c_str());
        if (client.connect(clientId.c_str())) {
            printf("MQTT connected\n");
        } else {
            printf("failed, rc=%d\n", client.state());

            //Wait 5 seconds before retrying
            printf(" try again in 5 seconds\n");
            delay(5000);
        }
    }
}
```

```

void setup() {
    //console baud rate
    Serial.begin(115200);

    //啟動WiFi連線
    setup_wifi();

    //設定 MQTT Broker & Port
    client.setServer(mqtt_server, mqtt_port);
}

int value = 0;
void loop() {

    if (!client.connected()) {
        //啟動 MQTT 連線
        mqttconnect();
    }
    client.loop();

    //每2秒回報1次
    unsigned long now = millis();
    if (now - lastMsg > 2000) {
        lastMsg = now;

        ++value;

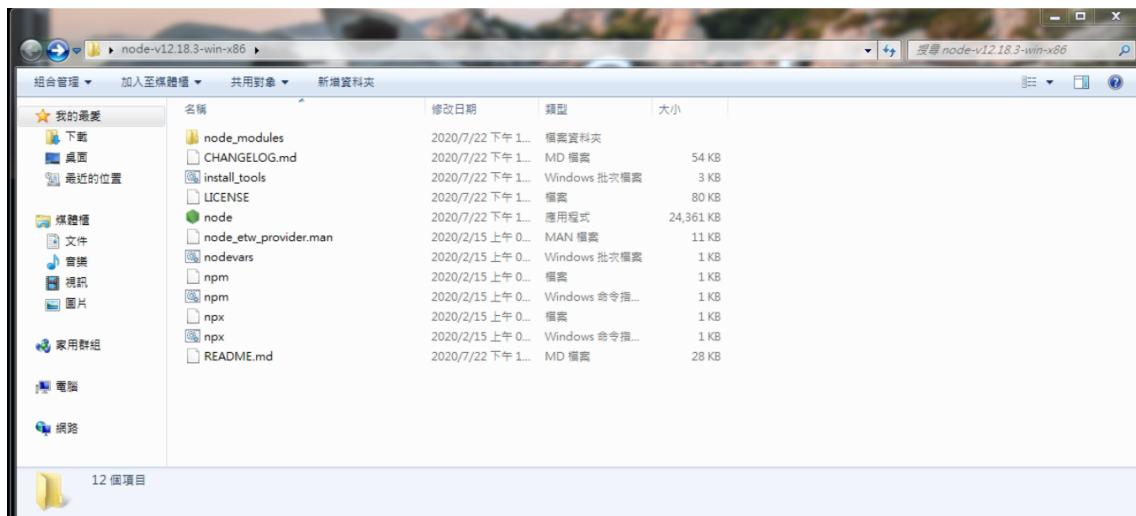
        sprintf (msg, MSG_BUFFER_SIZE, "{\"name\":\"%s\", \"value\":%d}", DUTNAME,value);
        printf("Publish message: %s\n",msg);
        client.publish(send_topic, msg, true);
    }
}

```

連上WIFI & Broker後，每 2 秒送指定Topic的Message給Broker

### 3. 安裝 Node.js

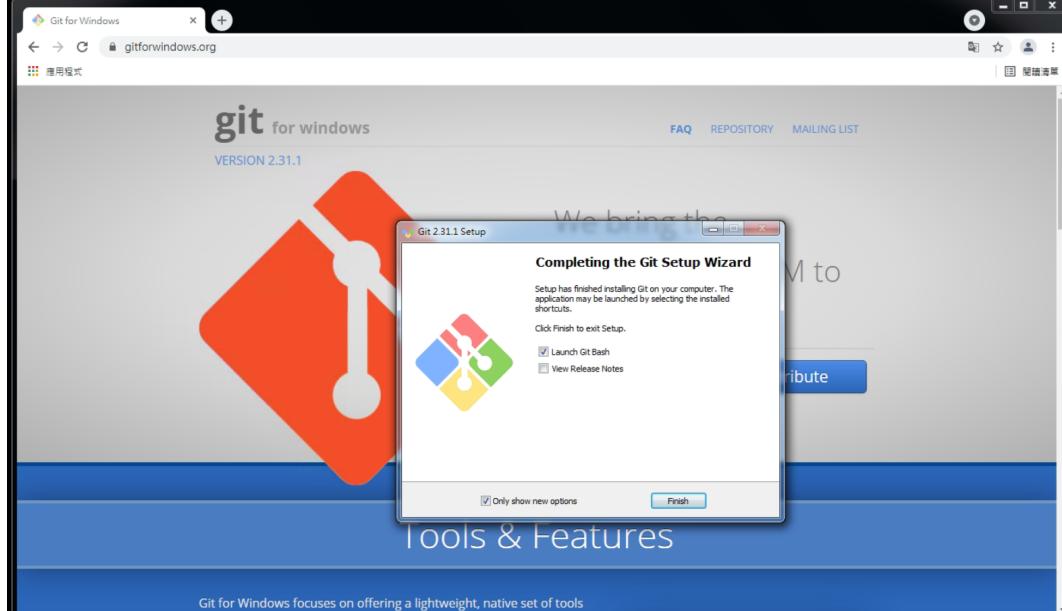
電腦A 作為MQTT Subscriber 是使用 node-red 及 node-red-dashboard，這些基於Node.js，首先安裝Node.js portable 32bit  
[官網下載點此](#)



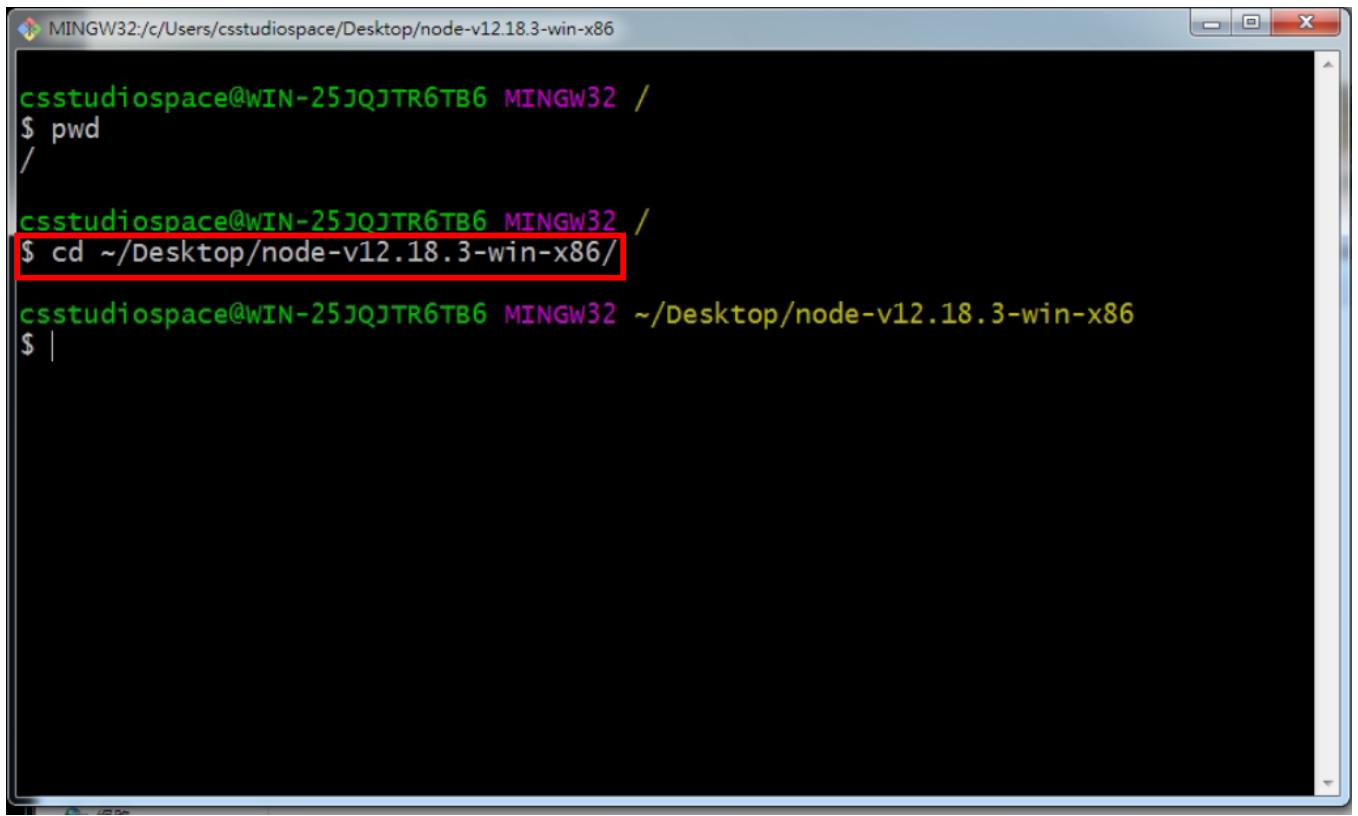
此為 Portable 版，無安裝過程，直接解壓縮即可

## 4. 安裝 Git for windows

Git for windows 是在 Windows 系統下，執行 Command的工具，裡面也包含了大部份的 Linux Command可用，Command的執行，會直接轉換成 Windows 的Command  
[官網下載點此](#)



過程也是一直下一步，直到Finish



```
MINGW32:/c/Users/csstudiospace/Desktop/node-v12.18.3-win-x86
$ pwd
/
csstudiospace@WIN-25JQJTR6TB6 MINGW32 /
$ cd ~/Desktop/node-v12.18.3-win-x86/
csstudiospace@WIN-25JQJTR6TB6 MINGW32 ~/Desktop/node-v12.18.3-win-x86
$ |
```

啟動時，預設不會進入Windows的任何路徑下，本範例皆會在Node.js目錄下操作（即是「C:/Users/[你的名稱]/Desktop/node-v12.18.3-win-x86」）。因此使用 cd 指令移動到該路徑下。

## 5. 安裝 node-red 及 node-red-dashboard

```
csstudiospace@WTN-25101TR6TR6:~ /Desktop/node-v12.18.3-win-x86
$ ./npm install node-red
npm WARN deprecated bcrypt@3.0.6: versions < v5.0.0 do not handle NUL in passwords properly
npm WARN deprecated node-pre-gyp@0.12.0: Please upgrade to @mapbox/node-pre-gyp: the non-scoped node-pre-gyp package is deprecated and only the @mapbox scoped package will receive updates in the future
npm WARN deprecated request@2.88.0: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated bcrypt@3.0.8: versions < v5.0.0 do not handle NUL in passwords properly
npm WARN deprecated node-pre-gyp@0.14.0: Please upgrade to @mapbox/node-pre-gyp: the non-scoped node-pre-gyp package is deprecated and only the @mapbox scoped package will receive updates in the future
C:\Users\csstudiospace\Desktop\node-v12.18.3-win-x86\node-red -> C:\Users\csstudiospace\Desktop\node-v12.18.3-win-x86\node_modules\node-red\red.js
C:\Users\csstudiospace\Desktop\node-v12.18.3-win-x86\node-red-pi -> C:\Users\csstudiospace\Desktop\node-v12.18.3-win-x86\node_modules\node-red\bin\node-red-pi

> bcrypt@3.0.6 install C:\Users\csstudiospace\Desktop\node-v12.18.3-win-x86\node_modules\node-red\node_modules\bcrypt
> node-pre-gyp install --fallback-to-build

'"node"' is not recognized as an internal or external command,
operable program or batch file.

> bcrypt@3.0.8 install C:\Users\csstudiospace\Desktop\node-v12.18.3-win-x86\node_modules\node-red\node_modules\node-red-admin\node_modules\bcrypt
> node-pre-gyp install --fallback-to-build

'"node"' is not recognized as an internal or external command,
operable program or batch file.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: bcrypt@3.0.6 (node_modules\node-red\node_modules\bcrypt):
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: bcrypt@3.0.6 install: `node-pre-gyp install --fallback-to-build`
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: Exit status 9009
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: bcrypt@3.0.8 (node_modules\node-red\node_modules\node-red-admin\node_modules\bcrypt):
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: bcrypt@3.0.8 install: `node-pre-gyp install --fallback-to-build`
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: Exit status 9009

+ node-red@1.3.4
added 302 packages from 279 contributors in 124.326s
```

透過npm 安裝指令(上圖紅框處), node-red 已安裝完成, 並花費124秒左右。npm是Node.js的套件管理器, 類似於 APP Store。

```
MINGW32:/c/Users/csstudiospace/Desktop/node-v12.18.3-win-x86
$ cd ~/Desktop/node-v12.18.3-win-x86/
$ ./npm install node-red-dashboard
npm WARN deprecated debug@4.1.1: Debug versions >=3.2.0 <3.2.7 || >=4 <4.3.1 have a low-severity ReDos regression when used in a Node.js environment. It is recommended you upgrade to 3.2.7 or 4.3.1. (https://github.com/visionmedia/debug/issues/797)
+ node-red-dashboard@2.29.0
added 71 packages from 59 contributors in 24.079s
$
```

透過npm 安裝指令(上圖紅框處), node-red-dashboard 已安裝完成, 並花費 24秒左右。

## 6. 啟動 Node-RED Server

```
MINGW32/c/Users/csstudiospace/Desktop/node-v12.18.3-win-x86
$ ./node-red
14 May 03:00:14 - [info] Welcome to Node-RED
=====
14 May 03:00:14 - [info] Node-RED version: v1.3.4
14 May 03:00:14 - [info] Node.js version: v12.18.3
14 May 03:00:14 - [info] Windows_NT 6.1.7601 ia32 LE
14 May 03:00:20 - [info] Loading palette nodes
14 May 03:00:29 - [info] Dashboard version 2.29.0 started at /ui
14 May 03:00:30 - [info] Settings file : C:\Users\csstudiospace\.node-red\settings.js
14 May 03:00:30 - [info] Context store : 'default' [module=memory]
14 May 03:00:30 - [info] User directory : C:\Users\csstudiospace\.node-red
14 May 03:00:30 - [warn] Projects disabled : editorTheme.projects.enabled=false
14 May 03:00:30 - [info] Flows file : C:\Users\csstudiospace\.node-red\flows_WIN-25JQJTR6TB6.json
14 May 03:00:30 - [info] Creating new flow file
14 May 03:00:30 - [warn]

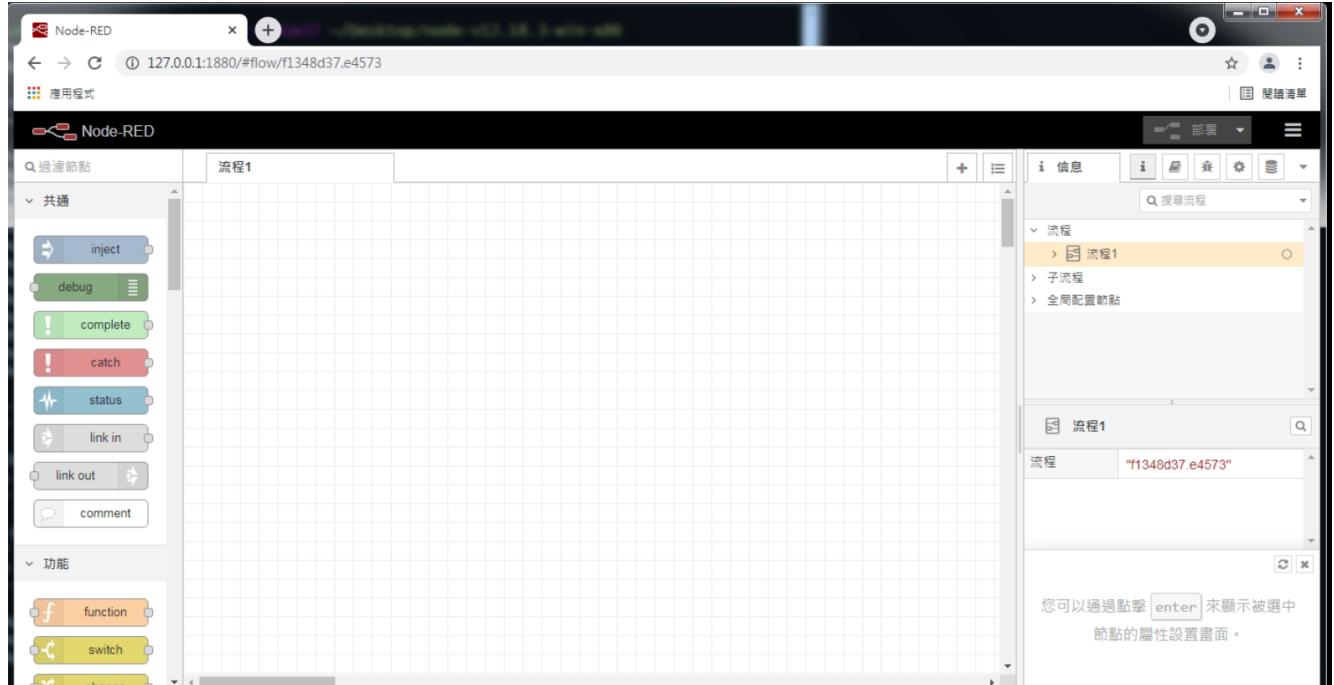
-----
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

-----
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.

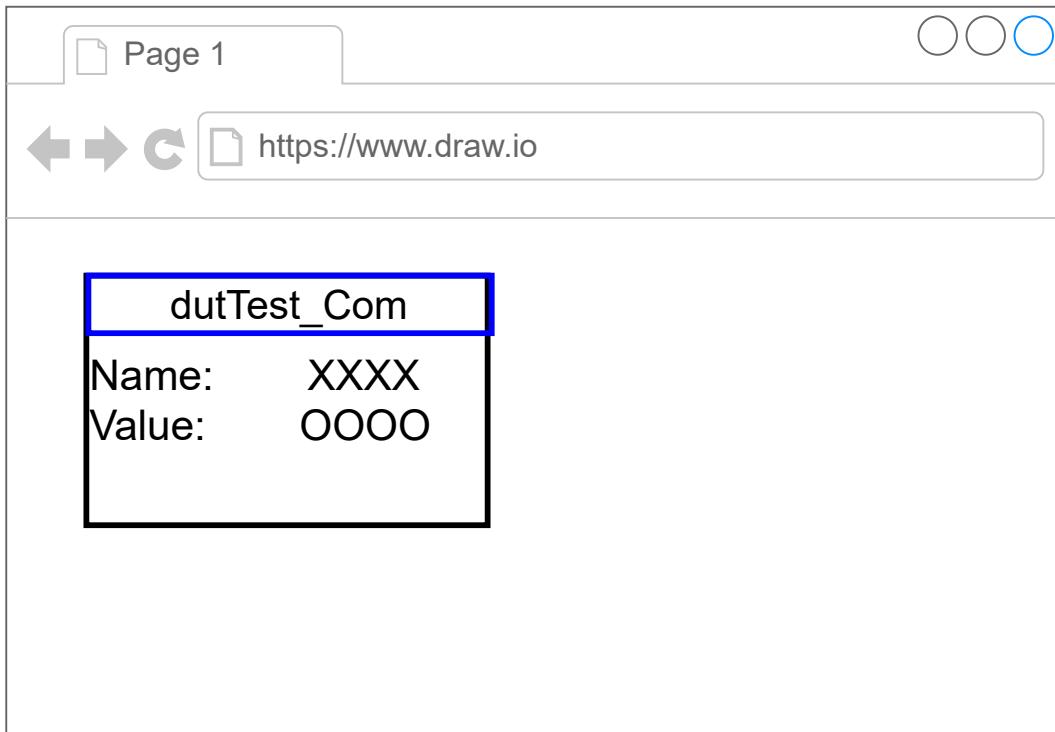
-----
14 May 03:00:31 - [info] Server now running at http://127.0.0.1:1880/
14 May 03:00:31 - [info] Starting flows
14 May 03:00:31 - [info] Started flows
```

透過 `./node-red` 指令(上圖紅框處), node-red-dashboard 已啟動, 且 Server 網址為 <http://127.0.0.1:1880/> (上圖藍框處)

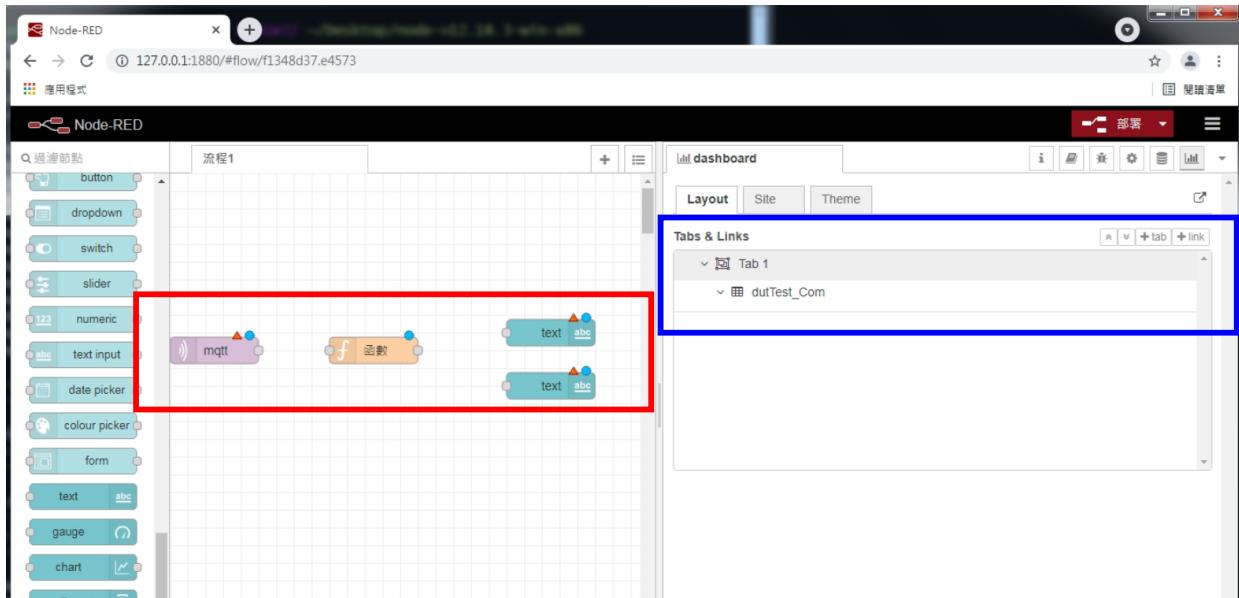


進入首頁後，可看到node-red 的元件操作頁面，使用者自行編輯想執行的元件，下一步會針對 MQTT Subscriber操作說明。

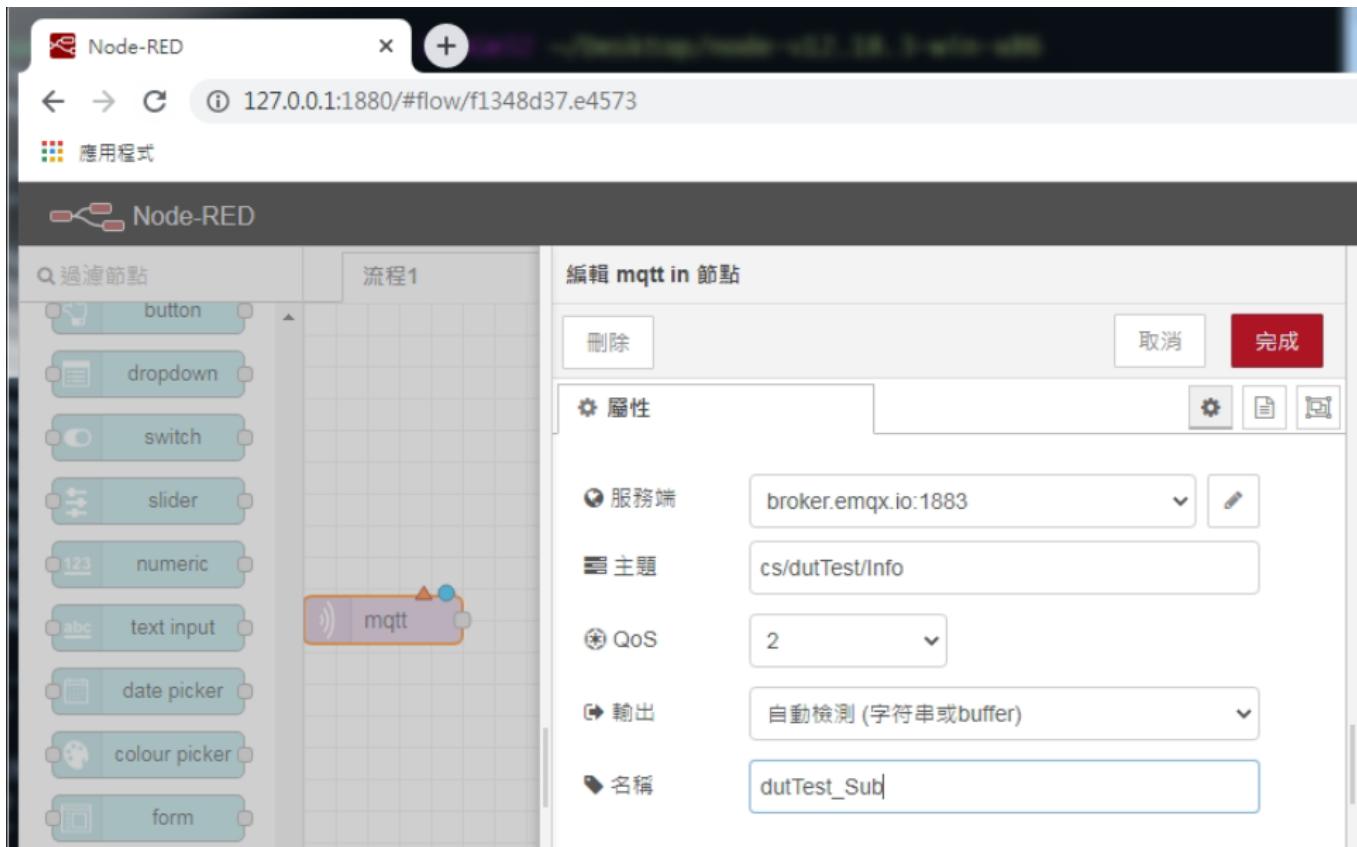
## 7. Node-Red 新增 MQTT Subscriber 向 Broker 訂閱 ESP8266 的 Topic



首先，先預想在網頁上看到的樣子，網頁上有一個Group [dutTest\_Com]能顯示裝置的 {Name,Value}



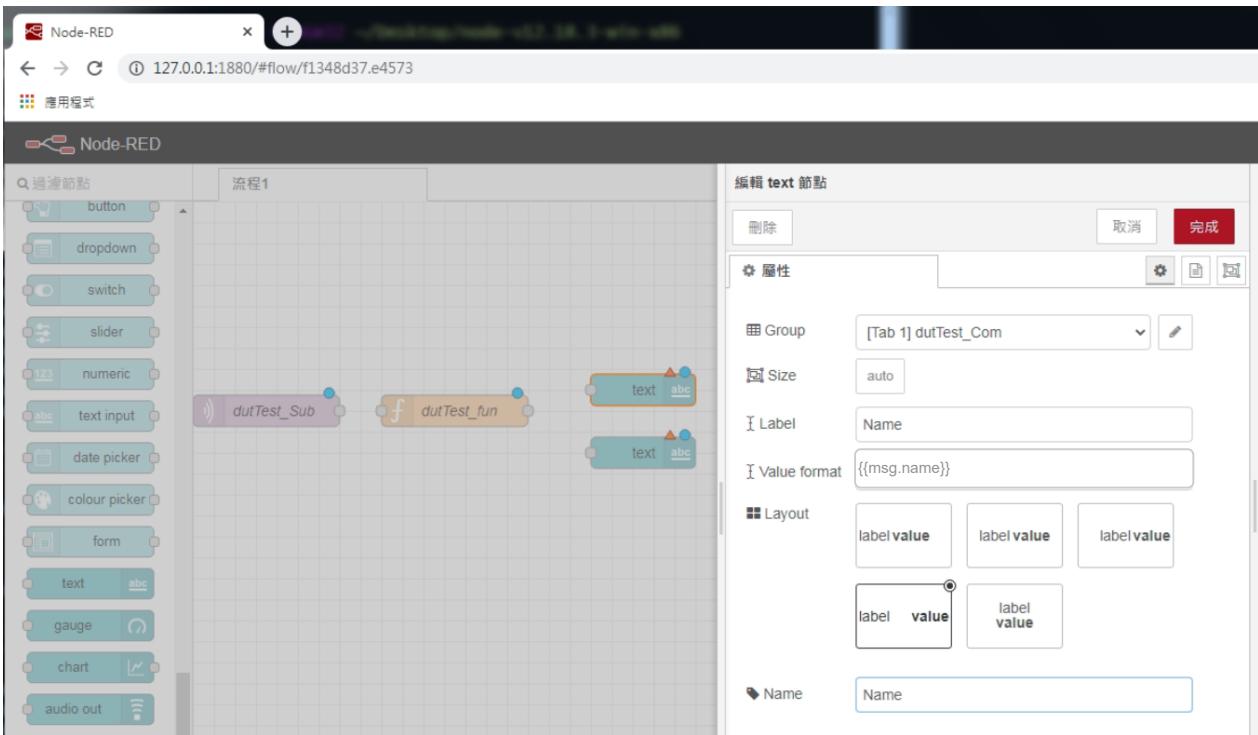
紅框處：把必要元件先建立出來，此時尚未修改。  
藍框處：建立一個Group 為 dutTest\_com。



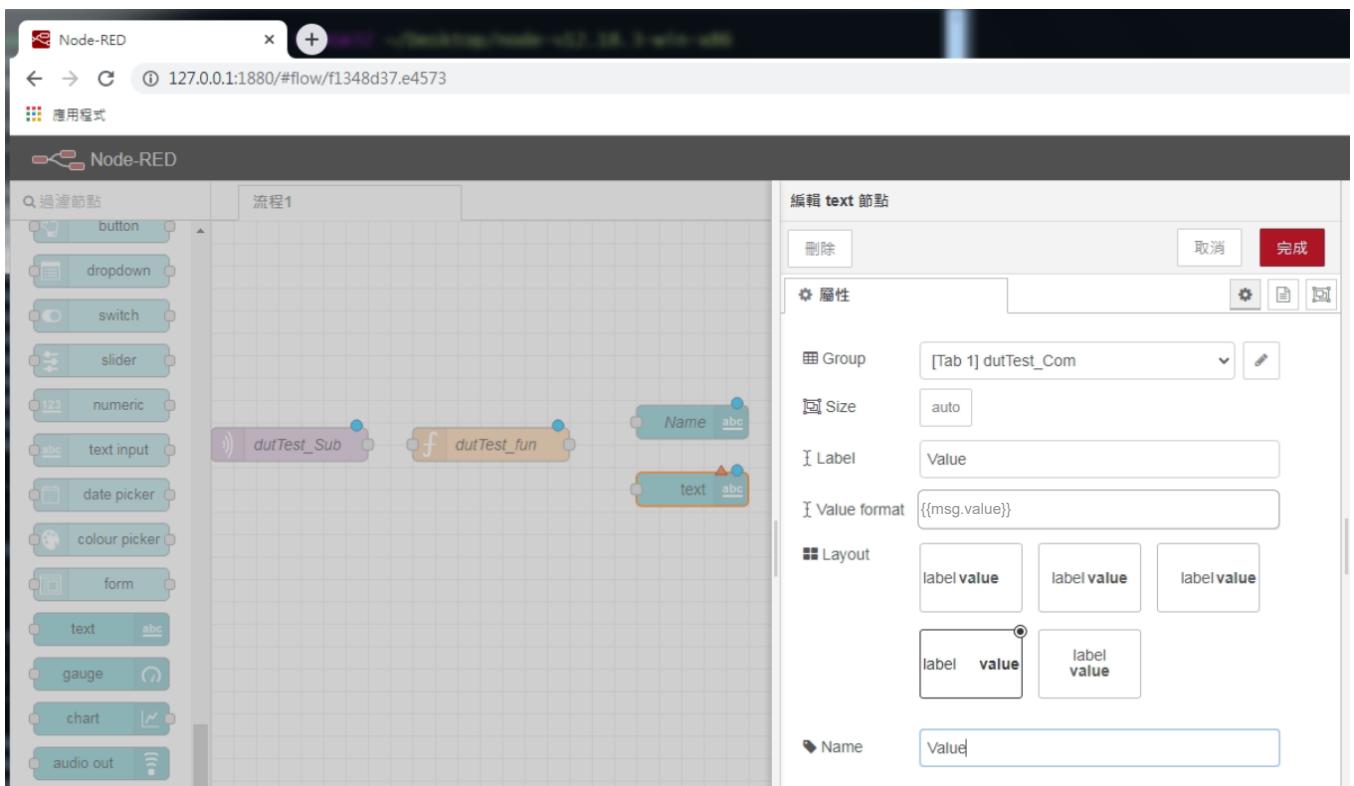
修改 mqtt 元件，服務端為第三方 broker 網址(點擊筆圖示新增，其他設定為預設值)，主題即為Topic「cs/dutTest/Info」，名稱為 dutTest\_Sub，接著按「完成」。



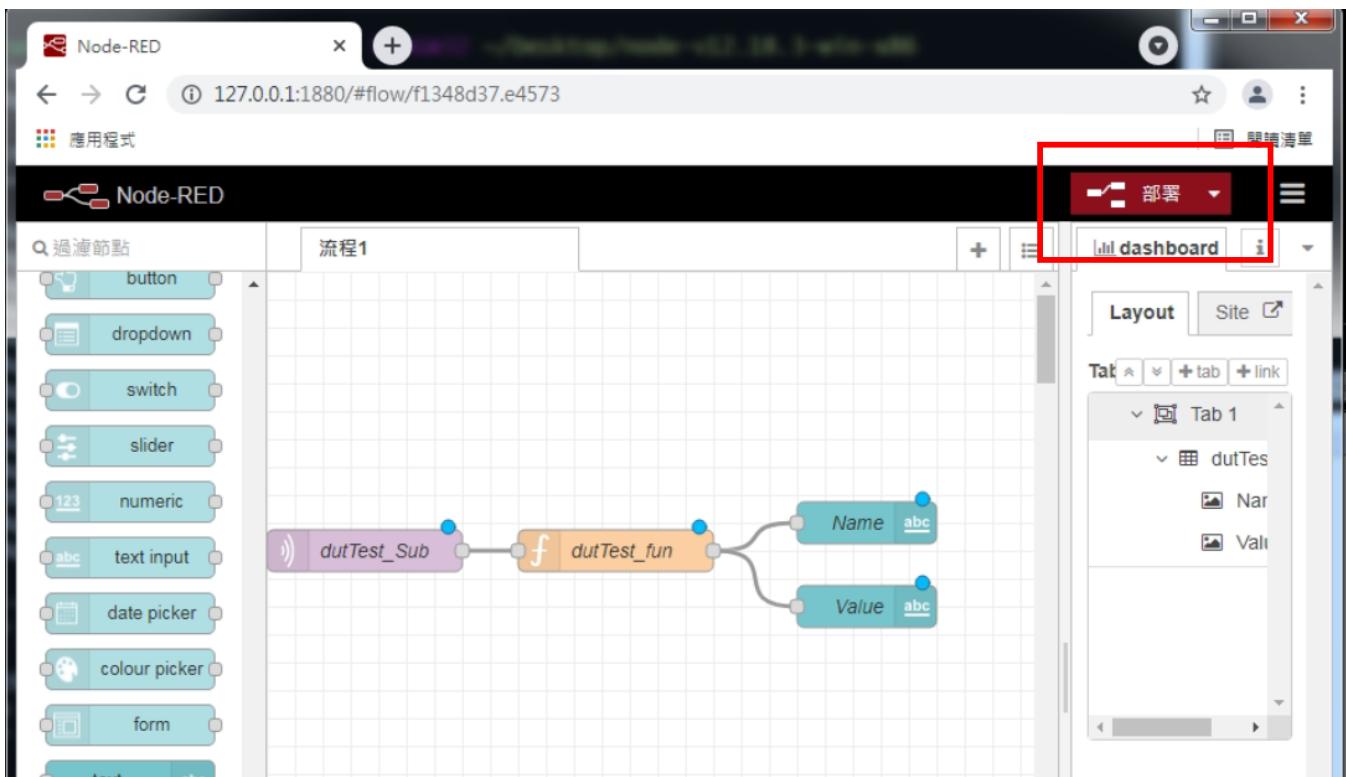
修改函數元件，此元件處理 dutTest\_Sub收到的Message，Node-Red會把Message放在結構體 msg.payload裡，在這裡用JSON解析msg.payload，並分別放在msg.name, msg.value, 往下一階傳送，名稱為dutTest\_fun，接著按「完成」。



修改 **text** 元件(綠方塊上)，此元件是顯示元件，就是在網頁上會看到的元件，因此需要指定顯示在哪個**Group**裡，選擇先前步驟已新增的**dutTest\_Com**。網頁上顯示的**Label**為 **Name**。Value Format 指從**dutTest\_fun**要接的變數為**msg.name**，選擇**Layout**，接著按「完成」。



修改 **text** 元件(綠方塊下)，此元件同 **Name**，也選擇先前步驟已新增的**dutTest\_Com**。不過要顯示**Value**，因此網頁上顯示的**Label**為 **Value**。Value Format指從**dutTest\_fun**要接的變數為**msg.value**，選擇**Layout**，接著按「完成」。



接著，把元件之間的線連起來(灰色線)，溝通就算完成了。  
最後的動作，把元件部署到網頁上，按下「部署」(紅框處)。

## 8. 觀看結果

Name	Value
Name	MyESP8266
Value	4705

觀看部署結果，在 <http://127.0.0.1:1880/ui>

如上圖所示，可看到電腦A從 Broker 收到 ESP8266的 Message，且 Value 會每2秒更新1次。

如果想將 Name, Value上下交換，可在Tab 1右方點擊Layout即可修改。