封面故事

Chapter 14

MQTT 遠端控制溫濕度及風扇　智慧化畜牧業管理

封面故事人物「黃科翔」，畢業於嘉義大學動物科學系畢業，因對畜牧業有興趣下而接手了家族企業，選擇傳承自家牧場，現正經營「黃科翔畜牧場」，盡力利用科技優勢，以翻轉農業過往費力勞苦的既定印象。學習物聯網課程，因此將原本牧場手動控制的設備，包括風扇、水簾、燈光、閘門等全部改為雲端 MQTT 控制，不僅提升豬隻的存活率及換肉率，良好的環境也更能養餵出健康的豬隻。

近期完成設計豬隻記號工具，可以作到一個人抓豬隻且施打疫苗，節省人力時間成本。「這些年育成率過程從 7 成提升到接近 100%，將所學的專業能運用在工作上，真得很開心。」科翔分享轉變過程。

科翔牧場負責人及其養殖的小豬

利用 APP 控制養殖環境

MQTT 互動

14-1　MQTT 協定

14-2　MQTT 遠端飼料機

前面幾章使用 ESP32 WiFi 的功能，不管是客戶端（Client）收發資料或是成為伺服器（Server）接收資料都相當方便，而本章則是介紹新的另外一種傳輸模式 MQTT。

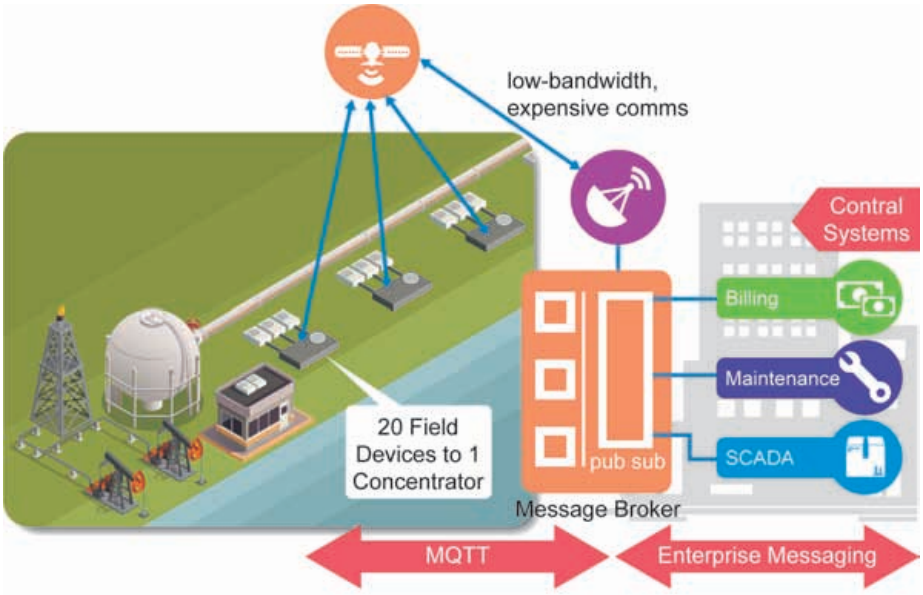
MQTT 雖然也是 TCP 架構，但是有別於 HTTP POST 及 GET 的傳輸模式，設計上具有輕量、低功耗的特性，非常適合給物聯網裝置使用，本章先簡介 MQTT 的歷史與架構，並結合 MQTTGO 的數據呈現工具來接收 ESP32 上的數據，第二節則利用伺服馬達做一個可以遠端操控飼料機，以免家中的寵物挨餓。

14-1 MQTT 協定

本節會先對 MQTT 的歷史與傳輸模式做簡單說明，由於 MQTT 的傳輸模式設計與以往有著很大差異，因此透過本節的說明讓讀者了解 MQTT 的重要性及傳輸的方式，接著用 DHT11 將溫濕度傳輸給電腦，並可以在電腦中下指令改變燈號。

一 MQTT 歷史與特點

所謂的 MQTT 全名是：Message Queueing Telemetry Transport，意思是訊息數列遙測傳輸，於 1999 年由 IBM（International Business Machines Corporation）公司為了觀測輸油管線狀態所研發，由於輸油管線相當長，感測器傳輸過程需要耗費大量頻寬及電力，因此傳統的傳輸模式較不合適，需要研發一款針對感測器的傳輸模式，最重要的就是要能作到低功耗、低流量、非同步的特點，為此 IBM 研發了 MQTT 傳輸模式來因應需求。

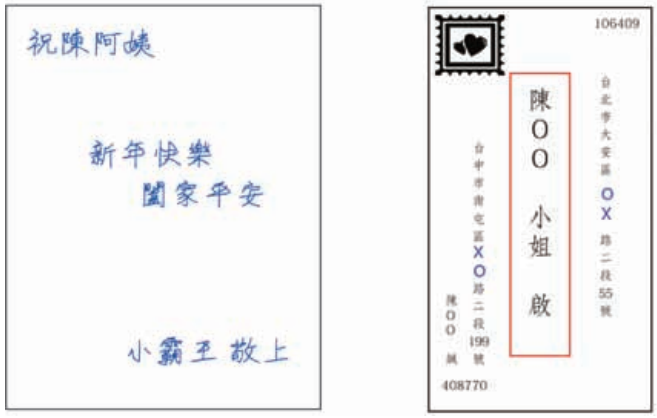


▲ IBM 研發 MQTT 架構

在前幾章我們已學習在 WiFi 傳輸資料協定像是 HTTP POST 或 GET，而這與 MQTT 有著幾個顯著的差異，本書主要以傳輸量及傳輸模式兩個部分來比較：

1. 輕量化傳輸

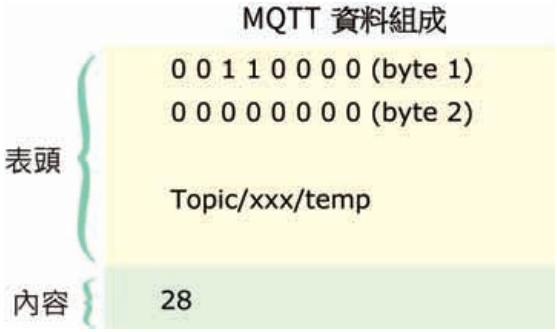
當我們在網路上傳遞資料時，並非只有資料本身，這裡用一個生活中的例子來說明：假設寫一張卡片寄給遠方的親友，寫完祝賀詞放入信封內，在信封寫上雙方的地址，並貼上郵票才能投入郵筒寄出，如果只有投入卡片，沒有信封、郵票、地址，對方是沒辦法收到的。



寄出的信件包含了「卡片」、「信封」、「地址」、「郵票」…等等資訊，但是實際上我們要對遠方親友所傳達的只有卡片內的祝賀詞，但為了讓信件能安全的寄達，必須要外面加上這些元素。相同的，網路上傳輸資料時也是類似的狀況，第十章的傳送 LINE 通知為例，雖然我們要傳送資料是「Temp=28」（溫度 =28 度），所佔的資料長度只有 7 個位元組大小，但是為了要能送達伺服器，還要加上網址、IP、認證碼、資料格式…等等的資料，這些則代表了信件外面的信封、地址、郵票等等的資訊，這些資料的長度則佔約 161 個位元組，卻遠遠超過資料本身。

當然對於一般的使用者來說這樣的資料長度不是什麼大問題，但是對於 IBM 在監測輸油管線時就產生大量的成本，因為輸油管線經過的地方都是廣無人煙的沙漠地帶，當然沒有任何網路訊號可以用來傳輸資料，因此只能借助天上的衛星，而衛星的傳輸價格非常貴，因此如何將資料減量就成為當時的課題。

此時若我們改以 MQTT 的方式來傳輸一筆溫度，資料則可大幅度的簡化，參考下圖的 MQTT 資料格式可以簡化如下，最前方的表頭只用到兩個位元組，而 Topic 則是用來標明資料寫入的位置，以及最後的資料「28」，整體算下來不到 30 位元組，所以一樣是傳輸溫度是 28 度，MQTT 資料量僅有將近 POST 的 1/5，這也就是 MQTT 被稱為輕量化傳輸的重要原因，因此有些人會把 MQTT 稱為是「明信片」，資料量少，郵資當然便宜了。



2. 代理傳輸模式

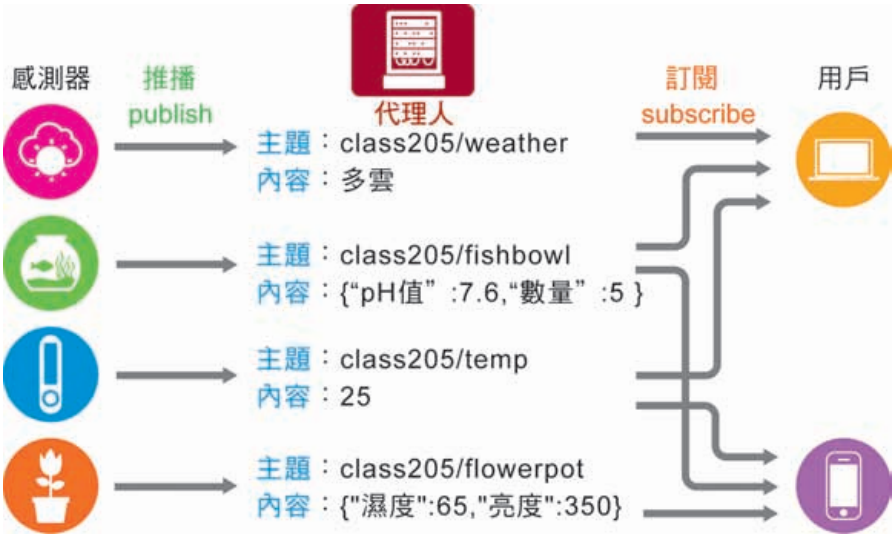
讀者在比較 POST 與 MQTT 資料格式時有沒有發現一個問題，MQTT 在傳輸資料時並不需要指定接收方的地址，而是改用主題「Topic」的方式，而這也是與 POST 傳輸過程最大的差異，在 MQTT 中資料並不是直接傳給用戶，而是透過一個代理人（Broker）來接收眾多分散在各地的感測器資訊，並儲存在特定的主題內，而使用者則會接收到代理人最新的主題更新資訊，關於 MQTT 主要幾個關鍵字說明如下：

代理人：接收或傳遞資料的伺服器。

主題：代理人伺服器上儲存資料的地方。

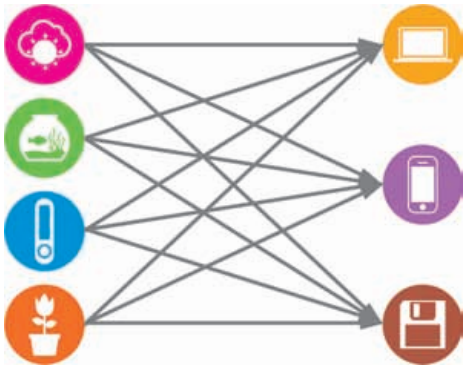
推播：感測器向代理人主動傳遞資訊，並會儲存在特定的主題。

訂閱：當資訊有更新時，代理人會主動將最新的資訊傳遞給有訂閱該主題的使用者。



代理機制可以將眾多的感測裝置的資料先收集到代理人身上，再將資料分送到需要的用戶手上，感測器不需要知道有哪些用戶需要資料？這些用戶在哪裡？使用什麼方式接收？只需要把資料送到代理人手上即可，而使用者就會透過代理機制收到最新的數值。

這樣的方式為什麼能降低流量及功耗呢？想像一下如果公司有 50 封帳單要寄給不同的客戶，一般來說每一封信都要單獨收郵資，而 MQTT 的代理機制則類似把 50 封帳單放在一個包裹內寄到郵局，郵局則放在每個客戶專屬的郵政信箱內，再通知客戶來拿即可，這樣全部只收一次運費了，其傳輸路徑則由原本的 NxM 變成 N+M，這樣算起來是不是很省運費呢。



▲ 傳統 NxM 條路徑

▲ MQTT N+M 條路徑

除了上面講述「輕量傳輸」、「代理機制」兩大特點之外，其他還有一些像是 QoS（Quality of Service）、非同步（Asynchronous）等特點，這些機制讓 MQTT 能達到低流量、低功耗的特性，非常適合分散式的物聯網裝置，也因此目前非常多的服務都慢慢接受改用 MQTT 機制，甚至連 Facebook 的即時通訊 message 也改用 MQTT 作為主要傳輸協定，足見 MQTT 在未來的重要性。

在了解 MQTT 的傳輸方式之後，接下來我們先做一個簡單的測試，將 DHT11 的數值透過 MQTT 傳輸到電腦裡。

1. 實驗名稱：

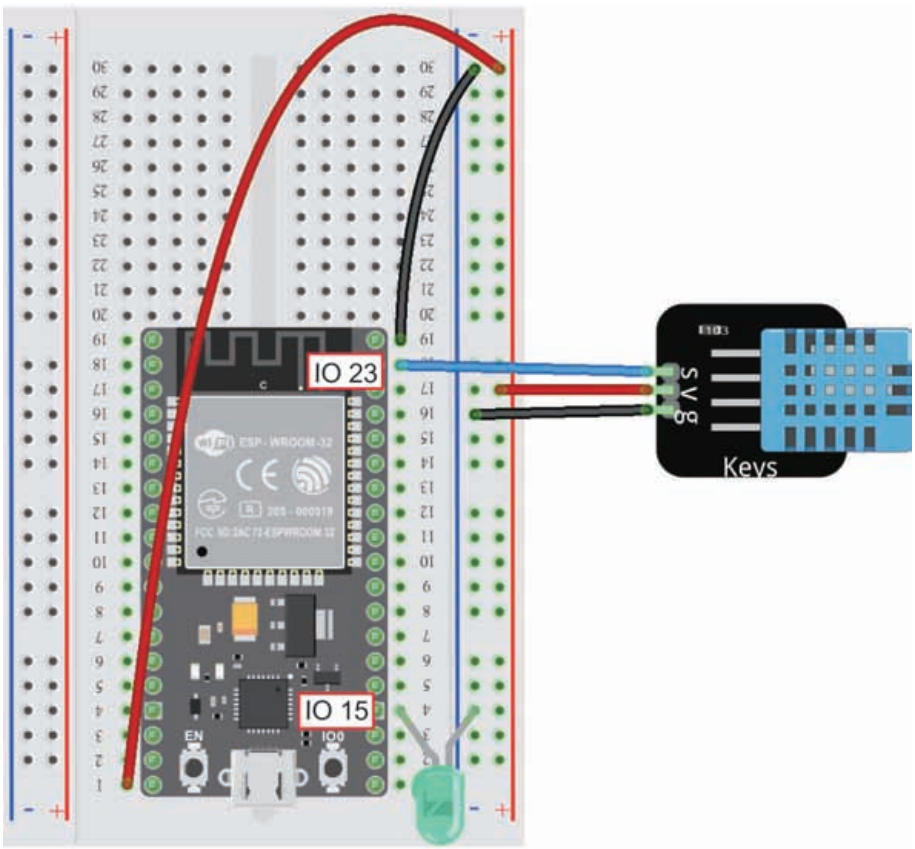
MQTT 的設定與傳輸方式。

2. 實驗目的：

以 MQTT 推播方式將教室 205 的溫濕度資料至特定主題，並透過訂閱方式改變燈號。

3. 準備材料：

- ESP32 × 1
- DHT11 × 1
- 麵包板 × 1
- 綠色 LED × 1
- 杜邦線若干



首先我們先將 LED 及 DHT11 依照上圖方式接好，接下來要在 Arduino 中安裝 MQTT 的程式庫。

點選左側程式庫圖示開啟程式庫管理員，在程式庫管理員上方的 ❶ 空白處輸入關鍵字：「PubSubClient」後，在 ❷ 下方找出我們要安裝作者為「Nick O'Leary」程式庫，點選右側的安裝按鈕，等候 1～2 分鐘即可完成安裝。

在上傳程式之前，我們先來了解一下 MQTT 主題名稱，本範例將會推播溫度及濕度到 MQTT 上，還會有一個主題用來改變燈號，所以一共需要三個主題，這三個主題本範例將會這樣命名：

主題 1：yourTopic/class205/temp，推播教室 205 的溫度到此主題
主題 2：yourTopic/class205/humi，推播教室 205 的濕度到此主題
訂閱 1：yourTopic/class205/led，訂閱此主題用來改變 led 燈的狀態

讀者會有疑問的是主題名稱是如何決定的？其實 MQTT 主題名稱使用者可以自行定義即可，但是必須遵守以下規定：

1. 主題為階層式，以「/」區分不同階層，我們可以直接看成電腦的檔案資料夾結構，例如「class205/temp」可以想像成 C 槽底下有一個名稱為 class205 的資料夾，temp 是裡面的檔案，MQTT 接收推播時將會存到這個檔案。
2. 階層名稱可以為空值，例如「class205/temp」不等於「class205//temp」，「class205//temp」代表 class205 下有一個沒有名稱的目錄，而這個沒有名稱的目錄下有一個 temp 檔案。
3. 主題可用英文數字中文皆可，但區分大小寫，所以「class205/temp」不等於「class205/Temp」，class205/濕度不等於 class205/溼度，因為「溼」與「濕」是不同的。

4. 不可使用以下字元：$、#、+、–、*、空白…等。

5. 階層數沒有限制，但總字數長度不可超過 65536 個字元。

除了上述的硬性規定之外，筆者還有自己的經驗分享給讀者，第一個是命名請用有意義的文字，且依照階層命名，舉例來說「Eric1030Home/2F/temp」，很容易理解就是 Eric1030 家的 2 樓的溫度，而「Eric1030Home/1F/humi」則代表 1 樓的濕度，主要就是容易辨識就可以了。

第二原則是名稱不能過於簡單，例如說我們將名稱訂為「Eric/test」，這樣會有什麼問題呢？主要是我們使用的是公用的伺服器，主題名稱不需要事先申請，所以全世界的人都可以自行命名，如果你的主題名稱過於簡單就很容易與他人重複，導致收到一堆其他人的資料，因此建議您在名稱前加上自己的英譯名稱，這樣就很難與他人重複，但若讀者有自己的 MQTT 伺服器就可以不需要考慮這個原則。

至於接收與傳遞資料的 MQTT 伺服器需要自己架設嗎？其實不用，因為網路上有很多公用的 MQTT 伺服器，除了免費之外也免註冊，省去申請帳號的麻煩，不過也因為是公用的關係，可能會有資料外洩的問題，但是因為練習用的資料不算機密，使用這類的公用 MQTT 較為方便，未來如果有需求，可再學習如何架設一台 MQTT 伺服器。

在眾多的 MQTT 伺服器中，筆者選擇使用 MQTTGO 伺服器，優點是伺服器架設在台灣，因此傳輸速率是最快的，更重要的是可以使用中文的主題名稱，以及內建數據視覺平台，非常推薦大家使用。以下為其之連線資訊：

- 名稱：mqttgo
- 網址：mqttgo.io
- 帳號密碼：不須帳號密碼
- mqtt 通訊埠：1883（預設）
- mqtts 通訊埠：8883

了解主題的命名方式後，讀者可以自行修改下面程式的主題名稱，接下來我們將程式燒錄進 ESP32 中。

```
#include <WiFi.h>
#include <PubSubClient.h> // 請先安裝 PubSubClient 程式庫
#include <SimpleDHT.h>
// ------ 以下修改成你自己的 WiFi 帳號密碼 ------
char ssid[] = "你的 WiFi SSID";
char password[] = "你的 WiFi 密碼";

//------ 以下修改成你腳位 ------
int pinDHT11 = 23;          // DHT11
```

```
SimpleDHT11 dht11(pinDHT11);
int pinGLED = 15;                          // 綠色 LED

// ------ 以下修改成你 MQTT 設定 ------
char* MQTTServer = "mqttgo.io";            // 免註冊 MQTT 伺服器
int MQTTPort = 1883;                       // MQTT Port
char* MQTTUser = "";                       // 不須帳密
char* MQTTPassword = "";                   // 不須帳密
// 推播主題 1: 推播溫度（記得改 Topic）
char* MQTTPubTopic1 = "YourTopic/class205/temp";
// 推播主題 2: 推播濕度（記得改 Topic）
char* MQTTPubTopic2 = "YourTopic/class205/humi";
// 訂閱主題 1: 改變 LED 燈號（記得改 Topic）
char* MQTTSubTopic1 = "YourTopic/class205/led";

long MQTTLastPublishTime;                  // 此變數用來記錄推播時間
long MQTTPublishInterval = 10000;          // 每 10 秒推撥一次
WiFiClient WifiClient;
PubSubClient MQTTClient(WifiClient);

void setup() {
  Serial.begin(115200);
  pinMode(pinGLED, OUTPUT);                // 綠色 LED 燈

  // 開始 WiFi 連線
  WifiConnecte();

  // 開始 MQTT 連線
  MQTTConnecte();
}

void loop() {
  // 如果 WiFi 連線中斷，則重啟 WiFi 連線
  if (WiFi.status() != WL_CONNECTED) { WifiConnecte(); }

  // 如果 MQTT 連線中斷，則重啟 MQTT 連線
  if (!MQTTClient.connected()) {  MQTTConnecte(); }

  // 如果距離上次傳輸已經超過 10 秒，則 Publish 溫溼度
  if ((millis() - MQTTLastPublishTime) >= MQTTPublishInterval ) {
```

```
    // 讀取溫濕度
    byte temperature = 0;
    byte humidity = 0;
    ReadDHT(&temperature, &humidity);
    // ------ 將 DHT11 溫度送到 MQTT 主題 ------
    MQTTClient.publish(MQTTPubTopic1, String(temperature).c_str());
    MQTTClient.publish(MQTTPubTopic2, String(humidity).c_str());
    Serial.println("溫溼度已推播到 MQTT Broker");
    MQTTLastPublishTime = millis();  // 更新最後傳輸時間
  }
  MQTTClient.loop();                 // 更新訂閱狀態
  delay(50);
}

// 讀取 DHT11 溫濕度
void ReadDHT(byte *temperature, byte *humidity) {
  int err = SimpleDHTErrSuccess;
  if ((err = dht11.read(temperature, humidity, NULL)) !=
      SimpleDHTErrSuccess) {
    Serial.print("讀取失敗，錯誤訊息 =");
    Serial.print(SimpleDHTErrCode(err));
    Serial.print(",");
    Serial.println(SimpleDHTErrDuration(err));
    delay(1000);
    return;
  }
  Serial.print("DHT 讀取成功：");
  Serial.print((int)*temperature);
  Serial.print(" *C, ");
  Serial.print((int)*humidity);
  Serial.println(" H");
}

// 開始 WiFi 連線
void WifiConnecte() {
  // 開始 WiFi 連線
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
90    }
91    Serial.println("WiFi 連線成功");
92    Serial.print("IP Address:");
93    Serial.println(WiFi.localIP());
94  }
95
96  // 開始 MQTT 連線
97  void MQTTConnecte() {
98    MQTTClient.setServer(MQTTServer, MQTTPort);
99    MQTTClient.setCallback(MQTTCallback);
100   while (!MQTTClient.connected()) {
101     // 以亂數為 ClietID
102     String  MQTTClientid = "esp32-" + String(random(1000000, 9999999));
103     if (MQTTClient.connect(MQTTClientid.c_str(), MQTTUser, MQTTPassword)) {
104       // 連結成功，顯示「已連線」。
105       Serial.println("MQTT 已連線");
106       // 訂閱 SubTopic1 主題
107       MQTTClient.subscribe(MQTTSubTopic1);
108     } else {
109       // 若連線不成功，則顯示錯誤訊息，並重新連線
110       Serial.print("MQTT 連線失敗，狀態碼 =");
111       Serial.println(MQTTClient.state());
112       Serial.println("五秒後重新連線");
113       delay(5000);
114     }
115   }
116 }
117
118 // 接收到訂閱時
119 void MQTTCallback(char* topic, byte* payload, unsigned int length) {
120   Serial.print(topic); Serial.print("訂閱通知 :");
121   String payloadString;         // 將接收的 payload 轉成字串
122   // 顯示訂閱內容
123   for (int i = 0; i < length; i++) {
124     payloadString = payloadString + (char)payload[i];
125   }
126   Serial.println(payloadString);
127   // 比對主題是否為訂閱主題 1
128   if (strcmp(topic, MQTTSubTopic1) == 0) {
129     Serial.println("改變燈號 :" + payloadString);
```

```
130     if (payloadString == "1") digitalWrite(pinGLED, HIGH);
131     if (payloadString == "0") digitalWrite(pinGLED, LOW);
132   }
133 }
```

在這裡解說上面的程式，首先在程式最上方宣告 WiFi 及 MQTT 伺服器的設定，WiFi 的部分可以參考前面的第八章 WiFi 連線的部分，而在 MQTT 部分，由於我們沒有自己建立 MQTT Broker 伺服器，所以選用「mqttgo.io」作為我們的 Broker，通訊 Port 號則是通用的 MQTT Port：1883。

主題名稱則依據前述的定義方式命名，而變數 MQTTLastPublishTime 及 MQTTPublishInterval 則是用來處理推播的時間，MQTTLastPublishTime 是最後一次推播時間，MQTTPublishInterval 則是定義推播間隔（本例為 10 秒），程式 Loop 中我們會用 (millis() - MQTTLastPublishTime) >= MQTTPublishInterval 來比對上次推播與現在時間是否已經超過 10 秒，如果已經超過就再推播一次，並把本次推播時間紀錄到 MQTTLastPublishTime 裡，這樣就可以依設定的時間完成每次推播。

讀者會有疑問的是，為什麼不用常見的 delay(10000) 方式來延遲 10 秒，主要是因為本例有「訂閱」的關係，當有訂閱時就必須呼叫 MQTTClient.loop( ) 來檢查主題是否有更新，若使用 delay 來做延遲且超過 15 秒時就會收不到即時的訂閱更新訊息，因 15 秒為 PubSubClient 的 keepalive 預設值，因此對於 MQTT 推播與訂閱時，就不能使用長時間的 delay 了，而要使用較小的 delay 讓 MQTTClient.loop( ) 會經常被執行到，以避免沒收到訂閱訊息。

此外本次程式採用的是副程式的寫法，將大部分重複的流程都拉到外面副程式中，這樣的好處是讓程式的架構更清楚，容易了解程式的執行步驟，也更容易除錯，建議往後讀者可以多多利用，本例的副程式包括如下：

WifiConnecte：檢查 WiFi 連線狀態，若斷線則重新連線

ReadDHT：讀取 DHT11 溫濕度資料

MQTTConnecte：檢查 MQTT Broker 連線狀態，若斷線則重新連線

MQTTCallback：當訂閱主題資料有更新時，讀取並處理訂閱資料

由於 WifiConnecte 及 ReadDHT 的內容在前面的章節都有說明過了，本章則針對 MQTTConnecte 及 MQTTCallback 進行講解。

- MQTTConnecte

```
MQTTClient.setServer(MQTTServer, MQTTPort);
MQTTClient.setCallback(MQTTCallback);
```

MQTTClient.setServer 是設定要連線的 MQTT 伺服器的位址及連接埠，而 MQTTClient.setCallback 則是設定 Callback 時要執行的副程式，所謂的 Callback 是一個事件觸發，當訂閱的主題有更新時，會執行 "MQTTCallback" 這個副程式，完成兩個設定之後，就是用 While 等候連線成功。

```
String MQTTClientid = "esp32-" + String(random(1000000, 9999999));
```

首先以亂數建立一個 ClientID，由於 ClientID 是 MQTT 伺服器用來辨別使用者身分的編號，不能與其他人重複即可，因此在這裡直接用亂數生成一個 ClientID，然後再開始連線。

```
MQTTClient.connect(MQTTClientid.c_str(), MQTTUser, MQTTPassword);
```

這句則是開始連線到 MQTT 伺服器，除了 ClientID 之外，一般還需要帳號密碼，但由於我們選用的是免帳號密碼的伺服器，因此 MQTTUser 及 MQTTPassword 都設定為空字串（程式第 16～17 行），本程式會返回一個 Boolean 值代表是否連線成功。

```
MQTTClient.subscribe(MQTTSubTopic1);
```

一旦連線成功則告知伺服器，我們要訂閱的主題名稱，反之若訂閱不成功，則等候 5 秒後返回 While 迴圈，重新開始連線 MQTT 伺服器。

- **MQTTCallback**

MQTTCallback 副程式是當訂閱主題更新時會觸發本副程式，並帶入三個參數：主題名稱 topic、主題內容 payload、內容長度 length。

```
  for (int i = 0; i < length; i++) {
    payloadString = payloadString + (char)payload[i];
  }
```

這部分是將主題內容 payload 由原本的 byte 陣列轉成較為常用的 String 變數 payloadString，並列印出來。

```
strcmp(topic, MQTTSubTopic1) == 0
```

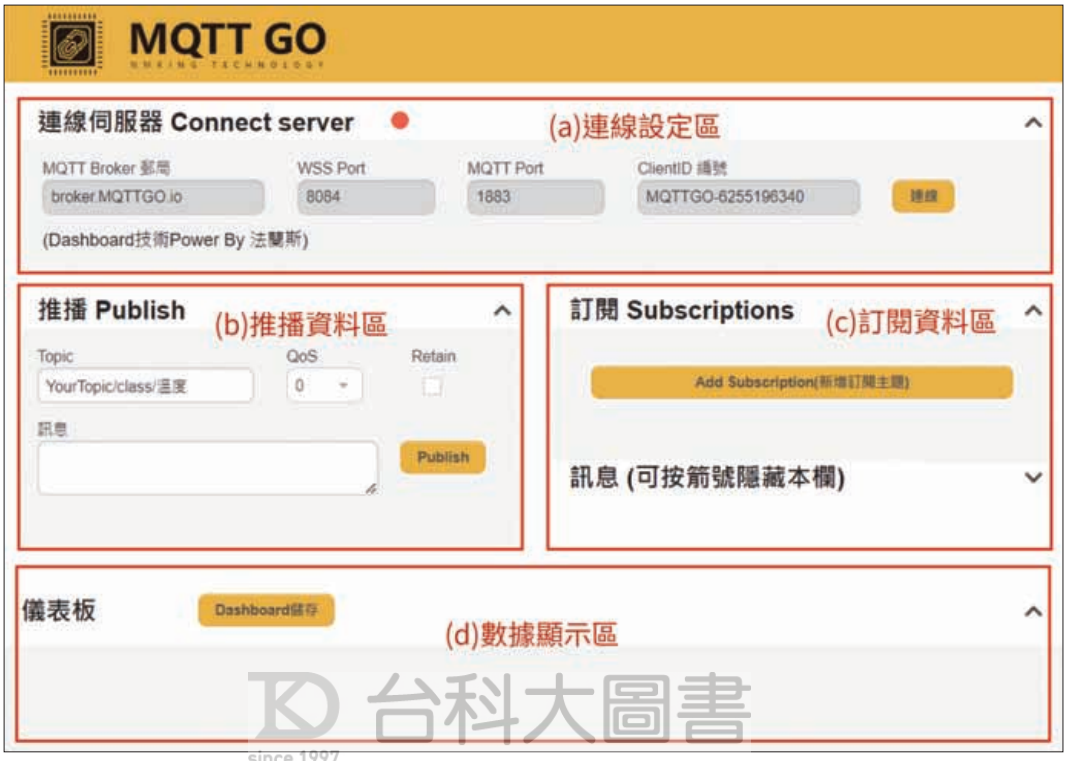
這句則是比對主題名稱是否為訂閱主題 1，本例中我們會用電腦的 MQTT Client 推播更新訂閱主題 1，本主題是讓 ESP32 開燈或關燈。因此若收到的 payloadString 等於 "1" 則用 digitalWrite(3, HIGH) 開啟 LED 燈，若為 "0" 則 digitalWrite(3, LOW) 來關閉 LED 燈。

程式上傳後，若設定正確，則可以從序列監控視窗中觀察到推播成功的訊息。接下來就是要安裝 MQTTLens 來接收 ESP32 上傳到 MQTT 的資料。

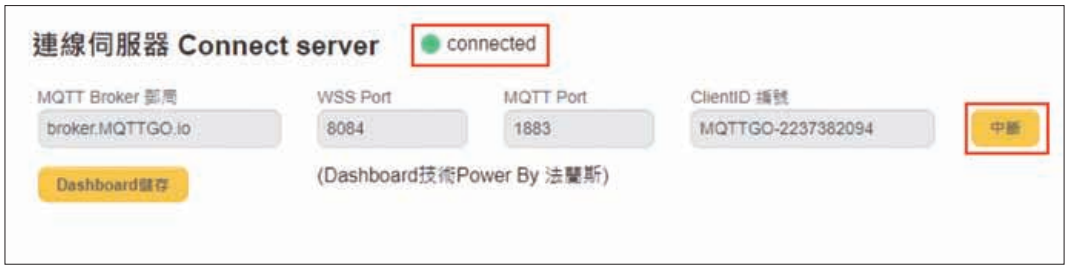
```
04:04:43.729 -> 溫溼度已推播到MQTT Broker
04:04:53.742 -> 讀取成功28 *C, 56 H
04:04:53.742 -> 溫溼度已推播到MQTT Broker
04:05:03.774 -> 讀取成功29 *C, 57 H
04:05:03.774 -> 溫溼度已推播到MQTT Broker
04:05:13.819 -> 讀取成功28 *C, 56 H
04:05:13.819 -> 溫溼度已推播到MQTT Broker
04:05:23.837 -> 讀取成功28 *C, 56 H
04:05:23.837 -> 溫溼度已推播到MQTT Broker
04:05:33.864 -> 讀取成功28 *C, 56 H
04:05:33.864 -> 溫溼度已推播到MQTT Broker
04:05:43.877 -> 讀取成功28 *C, 56 H
04:05:43.877 -> 溫溼度已推播到MQTT Broker
04:05:53.923 -> 讀取成功28 *C, 56 H
04:05:53.923 -> 溫溼度已推播到MQTT Broker
```

當看到序列視窗收到 ESP32 使用 MQTT 傳輸訊息之後，我們就可以用手機、電腦來訂閱 ESP32 的資訊，接收 MQTT 的資訊軟體及 APP 有很多種，以下介紹 MQTTGO 內建的 Web 界面接收資訊，並使用它提供的視覺化界面來呈現資料。

01 開啟一個瀏覽器，輸入網址：mqttgo.io，就可以開啟使用者界面，MQTTGO 界面一共分成四個區域：最上面是 (a) 連線設定區，中間左邊是 (b) 推播資料區，右側是 (c) 訂閱資料區，下面的儀表板則是 (d) 數據顯示區。

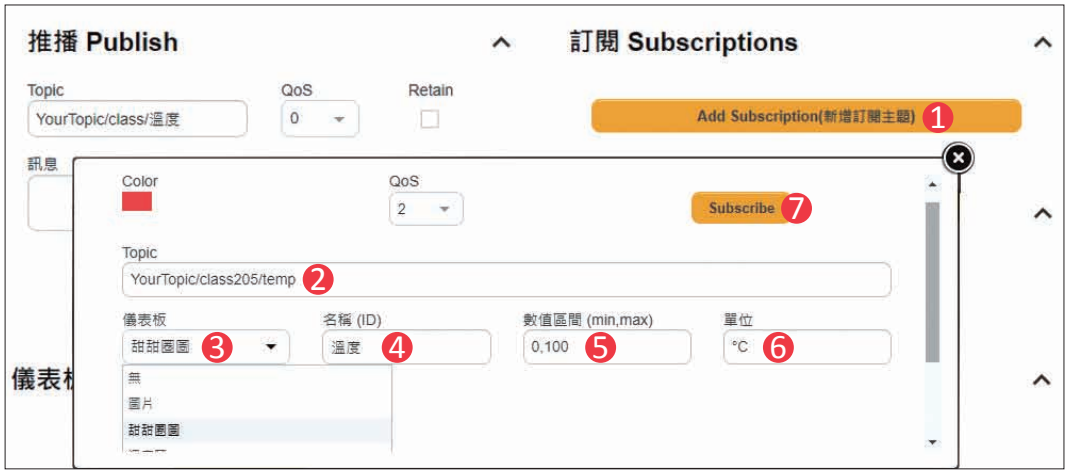


02 點選 (a) 連線設定區的連線按鈕，將網頁連線到 MQTT 伺服器，如果連線成功，中央的紅點會變成綠色的，且會出現 connected 的文字。



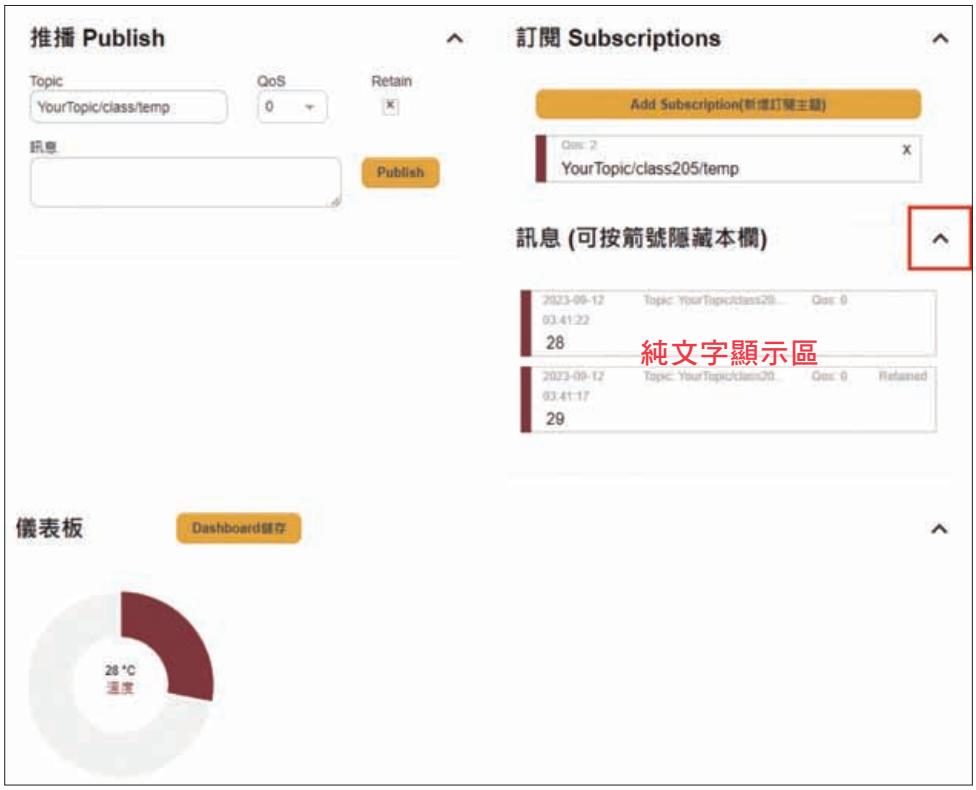
03 完成連線後，我們來訂閱 ESP32 推播的溫濕度資訊，首先：

❶ 點選右側的 (c) 訂閱資料區的 Add Subscription（新增訂閱主題）按鈕，即可跳出訂閱設定視窗，❷ 將 ESP32 推播的溫度主題名稱寫在下方的空白處，❸ 選擇要呈現的儀表板，這裡選用甜甜圈圖，❹ 名稱（ID）則輸入溫度，ID 就是這個圖表的名稱，依照規定每個圖表必須有不同的 ID 喔，❺ 輸入數值區間，例如我們選用溫度的範圍為 0～100 之間，則輸入「0,100」以逗號區隔開數值，❻ 最後再輸入單位，本例為溫度，因此單位為 °C。❼ 全部都設定完畢後按 Subscribe 按鈕完成訂閱。

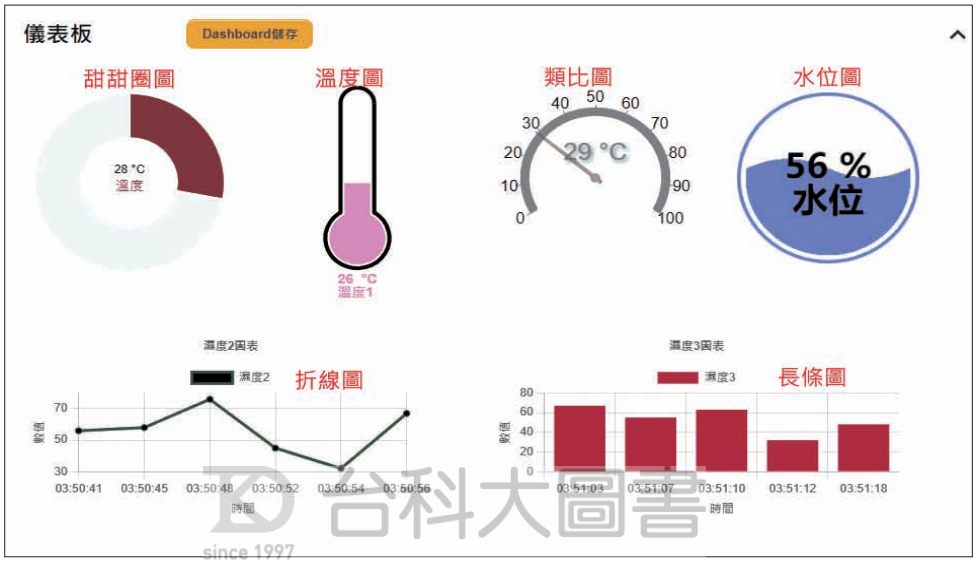


註：QoS 為接收時的 Quilty of Service，在此使用預設值 2 即可。

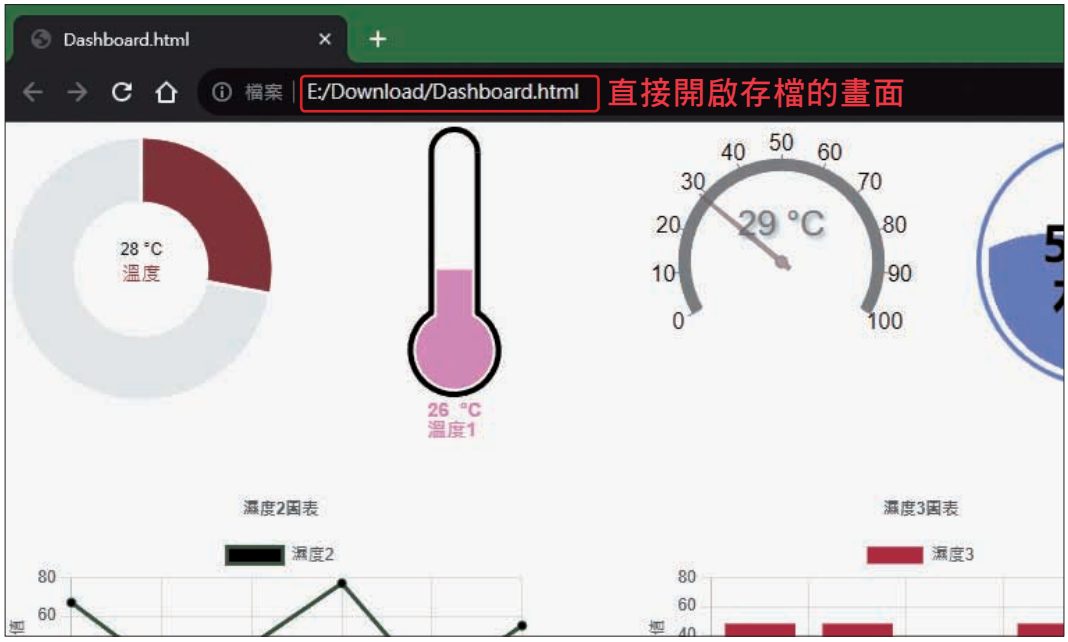
04 完成訂閱後，如果 ESP32 將資料傳到 MQTT 的主題時，這裡就會收到資料，除了純文字顯示外，下方圖表區也會產生一個甜甜圈圖，可以隨著數值變化改變甜甜圈的狀態。當收到的資料越來越多時，文字區可以會影響到圖表區的呈現，可以透過箭號「∧」將文字區隱藏起來。



05 讀者可以繼續完成濕度的圖表製作，本網站目前提供的圖表類型有七種，除了「圖片」這類是要給 ESP32CAM 傳送照片（第 17 章）外，讀者可以選擇一個你認為合適的數值呈現方式。



06 圖表儲存，當讀者設定好想要的圖表之後，可以將圖表儲存成一個 html 網頁檔案，以後只要用瀏覽器開啟這個存檔頁面，就可以直接收到資料，不需要連線到 MQTTGO 首頁再次設定，是不是很方便。



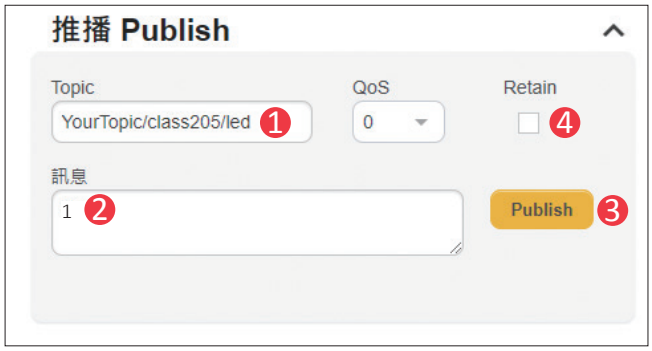
推播練習：

學會透過訂閱來收取 ESP32 的推播訊息後，接下來就是反過來讓我們把訊息推播給 ESP32 來訂閱，由於程式內我們已經設定 ESP32 訂閱主題「YourTopic/class205/led」，因此只需要將訊息 1 或 0 推播到這個主題即可讓 ESP32 收到，並依照程式開啟或關閉電燈。

07 推播訊息開啟電燈：❶ 在左側推播區的 Topic 輸入 ESP32 訂閱的主題名稱，例如筆者使用的是「YourTopic/class205/led」，❷ 在訊息處輸入開啟電燈的指令「1」，❸ 點選 Publish 按鈕，即可讓 ESP32 收到訊息，並開啟電燈。

這裡要注意的是，我們在程式內指定的是 1 或 0，有些同學會不小心加入一個換行或空白鍵，這樣都不認列，因為「1□」是不等於「1」的。

另外推播區 ❹ 右上角的「Retain」是保留訊息的功能，也就是說可以要求 MQTT 伺服器將我們推播的訊息永久保留在該主題內，這樣新的訂閱者或 WiFi 斷線重新連接後能於第一時間接收到最新的訊息，這個選項讀者可以選擇是否開啟。



08 推播完成後，觀察序列視窗是否有收到我們推播的訊息，再查看燈號有沒有發生改變。

```
04:19:36.331 -> 溫溼度已推播到MQTT Broker
04:19:46.352 -> 讀取成功29 *C, 57 H
04:19:46.352 -> 溫溼度已推播到MQTT Broker
04:19:56.401 -> 讀取成功29 *C, 57 H
04:19:56.401 -> 溫溼度已推播到MQTT Broker
04:20:06.416 -> 讀取成功29 *C, 57 H
04:20:06.416 -> 溫溼度已推播到MQTT Broker
04:20:16.457 -> 讀取成功28 *C, 56 H
04:20:16.457 -> 溫溼度已推播到MQTT Broker
04:20:26.474 -> 讀取成功29 *C, 57 H
04:20:26.474 -> 溫溼度已推播到MQTT Broker
04:20:36.515 -> 讀取成功29 *C, 57 H
04:20:36.515 -> 溫溼度已推播到MQTT Broker
04:20:44.327 -> YourTopic/class205/led訂閱通知:1
04:20:44.327 -> 改變燈號：1
```

ESP32 接收到我們推播的訊息

14-2 MQTT 遠端飼料機

如果我們出遠門去旅行時，家中的寵物或魚缸中的魚餓肚子該怎麼辦呢？這時上一節所學的 MQTT 可以派上用場，我們可以製作一個遠端飼料機，讓 ESP32 搭配伺服馬達 SG90 來控制飼料機的開關，並透過訂閱一個 MQTT 主題，讀者就可以用 MQTT 遠端餵食自己心愛的寵物或魚。

1. 實驗名稱：

MQTT 遠端控制飼料機。

2. 實驗目的：

透過 MQTT 遠端控制 SG90 伺服馬達，製作飼料機來灑放飼料。

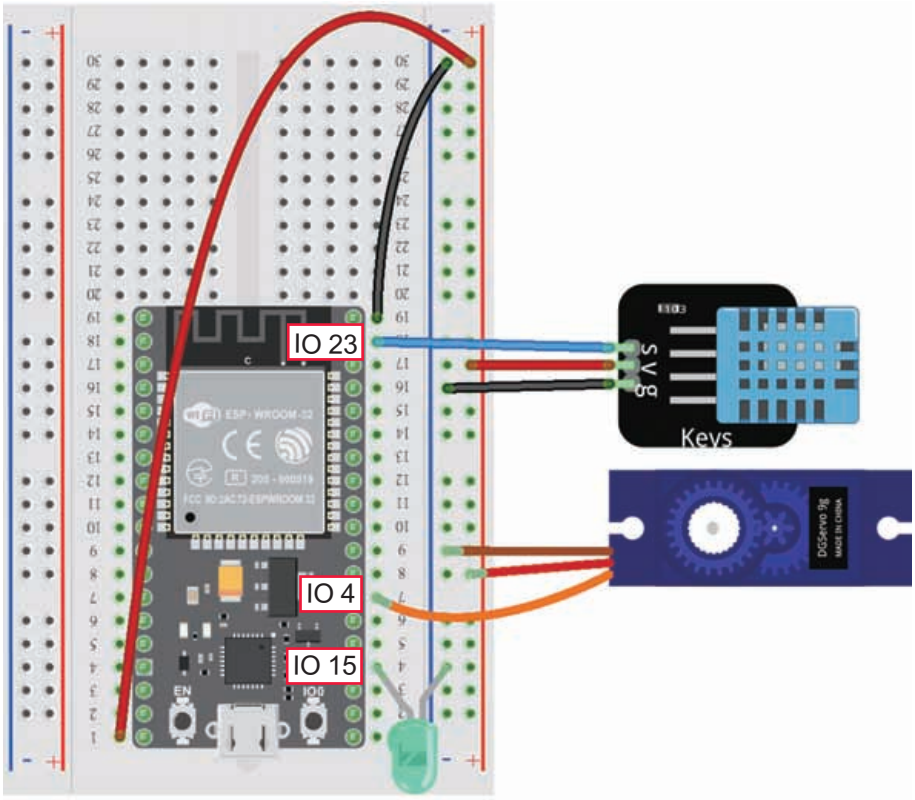
3. 準備材料：

- ESP32 × 1
- 麵包板 × 1
- 伺服馬達 SG90 × 1
- 透明寶特瓶 × 1
- 瓦楞紙 × 1
- 綁線數條
- 飼料若干（實驗時可用紙團代替）
- 杜邦線若干

伺服馬達（servo）因常用於遙控模型飛機，所以又常稱為 RC 伺服機（Radio Control Servo）、伺服馬達舵機等，目前市面上常見的有兩種規格：一款式藍色小型的 SG90，另外一款則是黑色外型較大的 MG995，兩者的差異在於扭力不同，SG90 所能產生的扭力大約 1.8KG/cm，而 MG995 則高達 13KG/cm，讀者可以依據要控制的物品重量來採購合適的伺服馬達，本例所使用的 SG90 外型如下圖，SG90 的接頭為三個母腳，顏色依序為棕、紅、橘，依序代表 GND、VCC 及控制腳位。

伺服馬達與一般的 DC 馬達有著很大的不同，伺服馬達的轉動角度為 0～180 度，而 DC 馬達則像風扇可以一直旋轉不停，最重要的是伺服馬達內部有一個控制電路，能精準的轉動到我們所要求的角度，而控制角度的方式與第五章所談到的蜂鳴器類似，都是採用 PWM 來控制，因此使用上必須安裝「ESP32Servo」程式庫，其安裝方式請參考第五章說明。

接線部分則請參考下圖，我們將 SG90 的橘色線（控制腳）接在 ESP32 的 GPIO4（左側 7），黑色接在右側的 GND，紅色線則接在右側 VCC 處。



在開始製作飼料機之前，我們先用程式測試我們的 SG90 是否工作正常，以下程式為 SG90 的測試程式。

`myServo.attach(4, 500, 2400);` 則是代表我們的伺服馬達接在 GPIO 4，且脈衝寬度是 500～2400 的 SG90，完成 SG90 的設定之後，我們就可以利用 `myServo.write（角度）` 來控制轉動的角度，角度值須介於 0～180 之間，若超出這個值會運作不正常，甚至可能損害內部機構。

本程式的 loop 內一共有四個 myServo.write（角度），分別是 90、180、90、0，代表我們將會讓伺服馬達從原點 0 度轉到 90 度，然後轉到 180 度，接著 90 度，最後回到原點 0 度的位置，上面的動作每 10 秒重複一次，若轉動正常，我們就可以接著來製作飼料機。

```
#include <ESP32Servo.h>
Servo myServo;                  // 建立一個伺服馬達物件
void setup(){
  // SG90 的脈衝寬度 500~2400
  myServo.attach(4, 500, 2400);
}
void loop(){
  myServo.write(90);            // 轉到 90 度
  delay(1000);
  myServo.write(180);           // 轉到 180 度
  delay(1000);
  myServo.write(90);            // 轉到 90 度
  delay(1000);
  myServo.write(0);             // 轉到 0 度（原點）
  delay(5000);
}
```

上面測試沒問題的話，就可以開始製作飼料機，我們可以利用 bb 彈或將報紙揉成許多紙團當作飼料裝進空白寶特瓶中，接著剪下一片大小合適的瓦愣紙，並利用 SG90 附帶的白色十字片及螺絲固定瓦愣紙，再將該瓦愣紙放在瓶口，最後用熱溶膠或綁線將 SG90 固定在寶特瓶瓶口處，關於本機構讀者可以參考右圖。

▲ 寶特瓶飼料機

程式碼部分則是修改上一節的內容，只是將 LED 換成伺服馬達，不過開啟方式則稍有修改，LED 的開啟及關閉由我們輸入「ON」及「OFF」來直接控制，但是飼料機部分則改成輸入要開啟多少時間然後就直接關閉，例如讀者在推播時輸入 1000，代表伺服馬達會開啟 1000ms，然後會「直接」將伺服馬達轉到關閉的位置，而不用手動下達關閉指令，這樣的好處是避免網路問題導致關閉指令未送到，結果飼料全部掉進魚缸內，魚兒就死光光了。

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <SimpleDHT.h>

// ------ 以下修改成你自己的 WiFi 帳號密碼 ------
char ssid[] = "你的 WiFi SSID";
char password[] = "你的 WiFi 密碼";

//------ 以下修改成你腳位 ------
int pinDHT11 = 23;                        // DHT11
SimpleDHT11 dht11(pinDHT11);

// -------- 伺服馬達控制 ---------
#include <ESP32Servo.h>
Servo myServo;                            // 建立一個伺服馬達物件
int pinServo = 4;                         // 伺服馬達腳位

// ------ 以下修改成你 MQTT 設定 ------
char* MQTTServer = "mqttgo.io";           // 免註冊 MQTT 伺服器
int MQTTPort = 1883;                      // MQTT Port
char* MQTTUser = "";                      // 不須帳密
char* MQTTPassword = "";                  // 不須帳密
// 推播主題 1: 推播溫度（記得改 Topic）
char* MQTTPubTopic1 = "YourTopic/class205/temp";
// 推播主題 2: 推播濕度（記得改 Topic）
char* MQTTPubTopic2 = "YourTopic/class205/humi";
// 訂閱主題 1: 改變 LED 燈號（記得改 Topic）
char* MQTTSubTopic1 = "YourTopic/class205/servo";

long MQTTLastPublishTime;                 // 此變數用來記錄推播時間
long MQTTPublishInterval = 10000;         // 每 10 秒推撥一次
WiFiClient WifiClient;
PubSubClient MQTTClient(WifiClient);

void setup() {
  Serial.begin(115200);
  myServo.attach(pinServo, 500, 2400);  // 伺服馬達

  // 開始 WiFi 連線
  WifiConnecte();

```

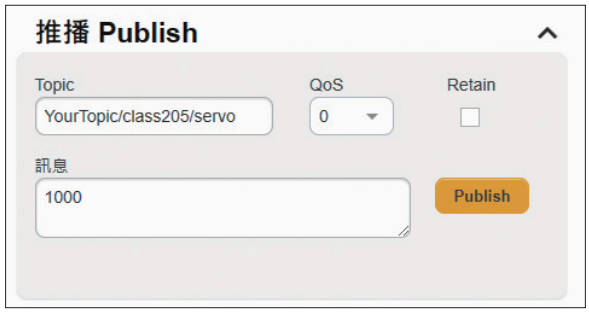
```
42    // 開始 MQTT 連線
43    MQTTConnecte();
44  }
45
46  void loop() {
47    // 如果 WiFi 連線中斷，則重啟 WiFi 連線
48    if (WiFi.status() != WL_CONNECTED) WifiConnecte();
49
50    // 如果 MQTT 連線中斷，則重啟 MQTT 連線
51    if (!MQTTClient.connected()) MQTTConnecte();
52
53    // 如果距離上次傳輸已經超過 10 秒，則 Publish 溫溼度
54    if ((millis() - MQTTLastPublishTime) >= MQTTPublishInterval ) {
55      // 讀取溫濕度
56      byte temperature = 0;
57      byte humidity = 0;
58      ReadDHT(&temperature, &humidity);
59      // ------ 將 DHT11 溫度送到 MQTT 主題 ------
60      MQTTClient.publish(MQTTPubTopic1, String(temperature).c_str());
61      MQTTClient.publish(MQTTPubTopic2, String(humidity).c_str());
62      Serial.println("溫溼度已推播到 MQTT Broker");
63      MQTTLastPublishTime = millis();     // 更新最後傳輸時間
64    }
65    MQTTClient.loop();                    // 更新訂閱狀態
66    delay(50);
67  }
68
69  // 讀取 DHT11 溫濕度
70  void ReadDHT(byte *temperature, byte *humidity) {
71    int err = SimpleDHTErrSuccess;
72    if ((err = dht11.read(temperature, humidity, NULL)) !=
73        SimpleDHTErrSuccess) {
74      Serial.print("讀取失敗，錯誤訊息 =");
75      Serial.print(SimpleDHTErrCode(err));
76      Serial.print(","); Serial.println(SimpleDHTErrDuration(err));
77      delay(1000);
78      return;
79    }
80    Serial.print("DHT 讀取成功：");
81    Serial.print((int)*temperature); Serial.print(" *C, ");
82    Serial.print((int)*humidity); Serial.println(" H");
```

```
}

// 開始 WiFi 連線
void WifiConnecte() {
  // 開始 WiFi 連線
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi 連線成功");
  Serial.print("IP Address:");
  Serial.println(WiFi.localIP());
}

// 開始 MQTT 連線
void MQTTConnecte() {
  MQTTClient.setServer(MQTTServer, MQTTPort);
  MQTTClient.setCallback(MQTTCallback);
  while (!MQTTClient.connected()) {
    // 以亂數為 ClietID
    String  MQTTClientid = "esp32-" + String(random(1000000, 9999999));
    if (MQTTClient.connect(MQTTClientid.c_str(), MQTTUser,
                           MQTTPassword)) {
      // 連結成功，顯示「已連線」。
      Serial.println("MQTT 已連線");
      // 訂閱 SubTopic1 主題
      MQTTClient.subscribe(MQTTSubTopic1);
    } else {
      // 若連線不成功，則顯示錯誤訊息，並重新連線
      Serial.print("MQTT 連線失敗，狀態碼 =");
      Serial.println(MQTTClient.state());
      Serial.println("五秒後重新連線");
      delay(5000);
    }
  }
}

// 接收到訂閱時
void MQTTCallback(char* topic, byte* payload, unsigned int length) {
```

```
123   Serial.print(topic);
124   Serial.print("訂閱通知 :");
125   String payloadString;        // 將接收的 payload 轉成字串
126   // 顯示訂閱內容
127   for (int i = 0; i < length; i++) {
128     payloadString = payloadString + (char)payload[i];
129   }
130   Serial.println(payloadString);
131   // 比對主題是否為訂閱主題 1
132   if (strcmp(topic, MQTTSubTopic1) == 0) {
133     Serial.println("開啟飼料機：" + payloadString + "ms");
134     myServo.write(90);         // 轉到 90 度（開啟飼料機）
135     // 將資料轉成整數後，設定為開啟時間
136     int opentime = payloadString.toInt();
137     delay(opentime);           // 開啟時間
138     myServo.write(0);          // 轉到 0 度（關閉飼料機）
139   }
140 }
```

控制飼料機的程式在最後一個副程式 MQTTCallback 中，與前一節控制 LED 時一樣要先比對訂閱主題是否正確，若主題沒問題的話，就是將伺服馬達轉到 90 度的位置，再暫停指定的時間，然後轉回 0 度也就是關閉。

在 MQTTGO 上的操作就是在推播的地方輸入 ESP32 所訂閱的主題名稱，以本例而言是「YourTopic/class205/servo」，然後在下方的訊息處輸入要開啟的時間，時間不能太短（例如小於 100），否則伺服馬達來不及轉動到 90 度，也不建議太長，筆者建議 500～1000（0.5～1 秒）之間，讀者可以自行調整時間來找到最合適的飼料掉落量，讀者也可以依照飼料的顆粒大小，自行修改伺服馬達開啟的角度喔。



後續讀者也透過第 16、17 章的 ESP32CAM 串流，搭配 MQTT 影像直播觀察剩餘的飼料量，再決定要開啟多久，關於 MQTT 的部分，本章就簡單的介紹到這裡，讀者可以進一步思考可以將 MQTT 用在哪些有趣的領域。

CH14 延伸練習

利用土壤感測器及水泵製作一個遠端澆花器，將土壤濕度數值以 MQTT 推撥到主題，當土壤太乾時，就利用 MQTT 開啟水泵來遠端澆花，避免植物渴死。