序 PREFACE

物聯網（Internet of Things，簡稱 IoT）是指在每個實體物品上裝設感測器，使物品變得「**有意識**」而能夠善解人意。微控制器再將感測器所擷取的數據資料，透過藍牙、Wi-Fi 等無線通訊技術，連接網際網路至雲端，進行計算、辨識、監控等服務。物聯網應用範圍廣泛，由個人穿戴裝置，到家庭、醫療、汽車、交通、工廠等多個領域，其中以**「智慧家庭」**領域的進入門檻較低，競爭也較激烈。國內外科技大廠積極投入研發照明、空調、門鎖、影音設備及智慧喇叭等家庭聯網專案開發。近年人工智慧（Artificial Intelligence，簡稱 AI）的快速發展，結合物聯網及 5G 技術的 AIoT（AI+IoT）智聯網應用，勢必會創造人類未來智慧生活的無限可能。

本書為誰而寫

本書專為對於現今當紅 **AI**、**IoT** **及智慧家庭**有興趣，卻又苦於沒有足夠知識、經驗與技術能力去開發設計的學習者編寫。全書淺顯易懂的圖文解說，按圖施工，保證成功。本書同時使用 **Arduino Uno** 及 **ESP32** 兩種最受歡迎的嵌入式開源開發板，完成相同物聯網專案設計。Arduino 板具有大量函式庫及模組支援，輕鬆上手、學習快速有效率，節省專案開發時間。ESP32 板內建藍牙、Wi-Fi 模組，節省開發成本，且 ESP32 晶片內含雙核心處理器，高效能的處理能力，極適合物聯網專案開發。另外，影像辨識使用瑞昱（Realtek）公司開發設計的 **AMB82-MINI** 開發板，內建人臉檢測及識別神經網路模型，可以輕鬆完成人臉辨識智聯網專案。

本書如何編排

全書內容以「**智慧家庭**」為主軸，從物聯網的基本概念，感知層的辨識、感測技術，網路層的藍牙、Wi-Fi 無線通訊技術，應用層的雲端運算、智慧插座、照明，以及人工智慧應用的手勢、指紋、語音及影像辨識等。全書**近百個生活化應用範例及練習**，絕對是一本最實用且 CP 值最高的物聯網入門與應用書籍。

各章所需軟、硬體相關知識及技術都有詳細圖文解說與實作，讀者可依自己興趣，適當安排閱讀順序。稍加修改本書範例，即可輕鬆完成實用的物聯網專案。

在物聯網中的網路層如同人體的**神經系統**，負責將神經末梢所感應的資訊傳送到大腦進行分析、判斷。在感知層與網路層之間的無線通訊，主要是使用藍牙（Bluetooth）、ZigBee 及 Wi-Fi 三種無線通訊技術。如表 4-1 所示 Bluetooth、ZigBee 及 Wi-Fi 的特性比較，Bluetooth 與 ZigBee 都能滿足**低成本**、**低功耗**、**快速連結**及**安全性高**等需求，常應用於無線個人區域網路（Wireless Personal Area Network，簡稱 WPAN）。相較於 ZigBee，Bluetooth 成本較低，常應用於物聯網設備中。Wi-Fi 通訊距離長，而且可以連上網際網路，常應用於無線區域網路（Wireless Local Area Network，簡稱 WLAN）。**WPAN 主要特點是低功耗，可使用電池供電**。

表 4-1　Bluetooth、ZigBee 及 Wi-Fi 的特性比較

特性	Bluetooth	ZigBee	Wi-Fi
協會 logo			
傳輸標準	IEEE 802.15.1	IEEE 802.15.4	IEEE 802.11
使用頻率	2.4GHz	868MHz、915MHz、4GHz	2.4GHz、5GHz
傳輸速度	1~3Mbps、24Mbps（HS 註 1）	10~250Kbps	11~54Mbps
傳輸距離	10 公尺（BLE 註 2）~300 公尺	10~100 公尺	50~100 公尺
消耗功率	4mA（BLE）、15~200mA	5mA	50mA
節點數目	8（BT）、32,000（BLE）	≤ 65,000	≤ 32
設備成本	中	低	高
安全性	高	中	低
網路拓撲	點對點、網狀（BLE）	星狀（star）、網狀	網狀（mesh）

註 1：HS 是高速藍牙（Bluetooth High Speed）的縮寫。

註 2：BLE 是低功耗藍牙（Bluetooth Low Energy）的縮寫。

4-1 藍牙技術

如圖 4-1 所示藍牙符號，藍牙技術是由 Ericsson、IBM、Intel、NOKIA、Toshiba 五家公司協議，使用 IEEE 802.15.1 傳輸標準，為一**低成本**、**低功率**、**涵蓋範圍小**的射頻（Radio frequency，簡稱 RF）系統。藍牙適用於連結電腦與電腦、電腦與周邊，以及電腦與其他行動數據裝置如手機、遊戲機、平板電腦、藍牙耳機、藍牙喇叭等。

圖 4-1　藍牙符號

藍牙所使用的載波頻帶不需要申請使用執照，大家都可以任意使用，所以可能造成通訊設備間的干擾問題。因此，藍牙使用跳頻展頻（Frequency Hopping Spread Spectrum，簡稱 FHSS）技術，來減少通訊設備間及電磁波的干擾。另外，使用加密技術來提高資料的保密性。所謂 **FHSS 技術是指載波在極短的時間內快速不停地切換頻率**。依 FHSS 技術規範，至少必須使用 75 個以上的頻率範圍，而且兩個不同頻寬之間跳頻的最大間隔時間為 0.04 秒（每秒至少跳頻 25 次以上）。**藍牙技術的傳輸規範使用 79 個頻率範圍，每秒跳頻 1600 次**。

每個藍牙連接裝置都是依據 IEEE 802 標準所制定的 48 位元位址，可以一對一或一對多連接。藍牙 2.0 傳輸率 1Mbps，藍牙 2.0+EDR（Enhanced Data Rate）傳輸率 3Mbps，藍牙 3.0+HS（High Speed）傳輸率 24Mbps。一般藍牙傳輸距離約 10 公尺，藍牙 4.0 提高傳輸距離至 60 公尺，並且提升了電源效率及網路節點數目。

自藍牙 4.0 開始，藍牙技術聯盟（Special Interest Group，簡稱 SIG）提出傳統藍牙（Bluetooth，簡稱 BT）、高速藍牙（Bluetooth High Speed，簡稱 Bluetooth HS）、低功耗藍牙（Bluetooth Low Energy，簡稱 BLE）三種模式。為了與低功耗藍牙區別，藍牙又稱為經典（classic）藍牙，可分傳統藍牙及高速藍牙兩種。

如表 4-2 所示經典藍牙與低功耗藍牙的特性比較，經典藍牙持續保持連接，進行大量數據通訊，消耗功率較大。**低功耗藍牙以短脈衝形式，進行少量數據通訊**，以降低功率消耗，但傳輸距離較短，低功耗藍牙並不相容於經典藍牙。SIG 聯盟於 2016 年推出藍牙 5.0 版本，與藍牙 4.0 相較，藍牙 5.0 具有更低的功耗、更快的傳輸速率、更高的安全性、更遠的傳輸距離，並且支援物聯網設備。

表 4-2　經典藍牙與低功耗藍牙的特性比較

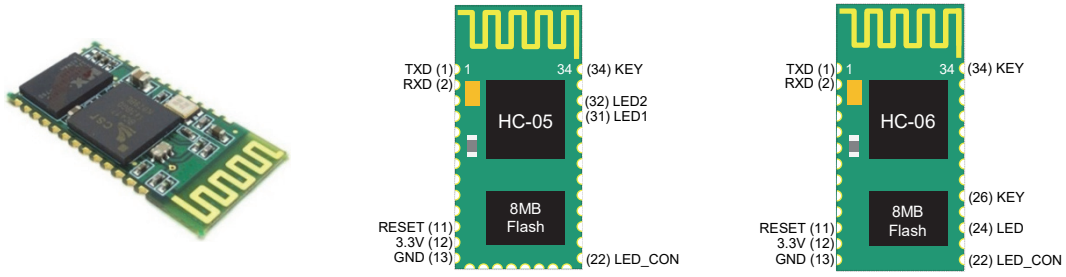
基本特性	經典藍牙 BT	低功耗藍牙 BLE
使用頻率	2.4GHz	2.4GHz
頻道使用	跳頻展頻（FHSS）	跳頻展頻（FHSS）
傳輸速率	1~3Mbps、24Mbps（HS）	1Mbps

基本特性	經典藍牙 BT	低功耗藍牙 BLE
傳輸距離	10~100 公尺	10~30 公尺
安全性	64/128 位元，可自訂	128 位元 AES 註1，可自訂
語音功能	有	無
聲音串流	有	無
免手持語音	有	無
消耗電流及功率	15~200mA / 1W	4~6mA / 0.01W~0.5W
網路拓撲	點對點	點對點、廣播、網狀網路

註 1：AES 是進階加密標準（Advanced Encryption Standard）的縮寫。

4-1-1 藍牙模組

如圖 4-2 所示廣州匯承信息科技所生產製造的 HC 系列藍牙模組，符合藍牙 V2.0+EDR 規格，並且支援 SPP（Serial Port Profile），係指使用者透過藍牙連線時可將其視為**序列埠裝置**。藍牙模組出廠預設參數為自動連線**從端（Slave）**角色（role），鮑率 9600bps、8 個資料位元、無同位元及 1 個停止位元的 8N1 格式，PIN 碼 1234。藍牙模組周邊如郵票的齒孔為其接腳，需自行焊接於萬孔板或專用底板上。



(a) 模組外觀　(b) HC-05 模組接腳　(c) HC-06 模組接腳

圖 4-2　HC 系列藍牙模組

如圖 4-2(b) 所示 HC-05 模組，同時具有**主控端（Master）**及**從端（Slave）**兩種工作模式，出廠前已經預設為從端模式，可使用 AT 命令更改工作模式。如圖 4-2(c) 所示 HC-06 模組只具有**主控端或從端**其中一種工作模式，出廠前已經設定為**從端**模式，無法再使用 AT 命令更改。

藍牙模組是一種能**將原有的全雙工串列埠 UART-TTL 介面轉換成無線傳輸**的裝置。藍牙模組不限作業系統、不需安裝驅動程式，就可以直接與各種微控制器連接，使用起來相當容易。HC-06 是較早期的版本，不能更改工作模式，AT 命令也相對較少，建議購買 HC-05 藍牙模組。如表 4-3 所示 HC-05 藍牙模組的主要接腳功能說明，只要注意電源規格及串列埠 RXD、TXD 的接腳，就能正確配對連線。

表 4-3　HC-05 藍牙模組的主要接腳功能說明

模組接腳	功能說明
1	TXD：藍牙串列埠傳送腳，連接至微控制器的 RXD 腳。
2	RXD：藍牙串列埠接收腳，連接至微控制器的 TXD 腳。
11	RESET：模組重置腳，低電位動作，不用時可以空接。
12	3.3V：電源接腳，電壓範圍 3.0V~4.2V，典型值為 3.3V，工作電流小於 50mA。
13	GND：模組接地腳。
31	LED1：工作狀態指示燈，有三種狀態說明如下： 1. 配對完成時，此腳輸出 2Hz 方波，也就是每秒快閃二下。 2. 模組通電同時令 KEY 腳為高電位，此腳輸出 1Hz 方波（慢閃），表示進入【AT 命令回應】模式，使用 38400 bps 的傳輸速率。 3. 模組通電同時令 KEY 腳為低電位或空接，此腳輸出 2Hz 方波（快閃），表示進入【自動連線】模式。
32	LED2：配對指示燈。配對連線成功後，輸出高電位且 LED2 恒亮。
34	KEY：模式選擇腳，有兩種模式。 1. 當 KEY 為低電位或空接時，模組通電後進入【自動連線】模式。 2. 當 KEY 為高電位時，模組通電後進入【AT 命令回應】模式。

4-1-2　含底板 HC-05 藍牙模組

為了減少使用者焊接的麻煩，元件製造商會將藍牙模組的 KEY、VCC、GND、TXD、RXD、RESET、LED1、LED2 等主要接腳，焊接組裝成如圖 4-3 所示含底板 HC-05 藍牙模組。不同製造廠商會有不同的引出接腳名稱，但接腳大同小異。

多數微控制器的工作電壓為 5V，而**藍牙模組的工作電壓為 3.3V**。因此，底板上內置 3.3V 直流電壓調整 IC（LD33V），可將輸入電壓 5V 穩壓輸出 3.3V，再供電給藍牙模組使用。

(a) 模組外觀　　(b) 接腳圖

圖 4-3　含底板 HC-05 藍牙模組

4-1-3　藍牙工作模式

藍牙模組有**自動連線**及 **AT 命令回應**兩種工作模式，當藍牙模組的 KEY 腳為低電位或空接時，藍牙模組工作在自動連線模式。在自動連線模式下又可分成「主端（Master）」、「從端（Slave）」及「回應測試（Slave-Loop）」三種工作角色。Master 角色為主動連接，Slave 角色為被動連接，而 Slave-Loop 角色為被動連接並接收遠端藍牙設備數據，並且將數據原樣傳回。

使用藍牙模組前必須**先進行配對**，配對完成再進行連線，連線成功後才能開始傳輸資料。藍牙模組連線前的電流約為 30mA，連線後不論通訊與否的電流約為 8mA，沒有休眠模式。當 KEY 腳為高電位時，工作在 **AT 命令回應**模式時，能執行所有 AT 命令對藍牙模組進行設定。

4-1-4　藍牙參數設定

多數的藍牙模組都能讓使用者自行調整參數，在出廠時預設為**自動連線**模式，模組使用設定好的參數來傳送或接收資料，但不會解讀資料內容。如果要設定藍牙模組的參數，KEY 腳必須為高電位，才能進入 **AT 命令回應**模式執行 AT 命令。**AT 命令不能透過藍牙無線傳輸來設定**，必須使用如圖 4-4 所示 USB 對 TTL 轉換器，將藍牙模組連接至電腦，再以序列埠監控軟體（如 AccessPort 通訊軟體）來設定。

(a) 連接線外觀　　(b) 接腳

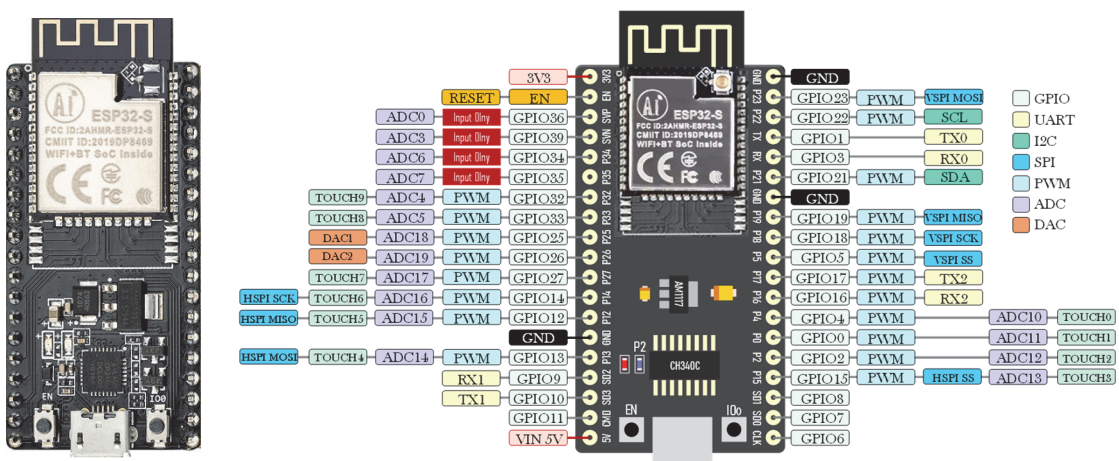
圖 4-4　USB 對 TTL 轉換器

4-3 認識 ESP32 開發板

ESP32 開發板使用雙核心處理器 ESP32 晶片，工作頻率 160MHz~240MHz。ESP32 晶片整合**經典藍牙**、**低功耗藍牙**及 **Wi-Fi** 功能，是專為行動裝置、穿戴式電子產品和物聯網應用而設計。ESP32 開發板種類繁多，常見產品有樂鑫（Espressif）官方原廠生產的 ESP32-DevKit / ES P32-WROOM-32 開發板、安信可（Ai-Thinker）官方原廠生產的 NodeMCU ESP32-S 開發板，以及副廠相容產品。**本書使用 NodeMCU ESP32-S 開發板**。

4-3-1 NodeMCU ESP32-S 開發板

如圖 4-23 所示 NodeMCU ESP32-S 開發板，有 20 支可用的通用輸入 / 輸出接腳（General-Purpose Input / Output 簡稱 GPIO），編號依序為 0、2、4、5、12~19、21~23、25~27、32、33，每支 GPIO 接腳**最大輸出電流 40mA**。GPIO34、35、36 及 39 只能當輸入腳，且沒有內建提升電阻。ESP32-S 開發板提供 UART、I2C、SPI 等串列介面，20 組 PWM，10 組內建的電容觸控（touch）感測器 TOUCH0~TOUCH9（GPIO4、0、2、15、13、12、14、27、33、32）及 16 組 12 位元 ADC 轉換器。



(a) 外觀　　(b) 接腳圖

圖 4-23　NodeMCU ESP32-S 開發板

NodeMCU ESP32-S 開發板內建兩個 LED，一個連接在底板上的 GPIO2，作為測試之用，另一個為電源指示燈。另內建兩個按鍵，EN 按鍵為重置（reset）鍵，**IO0 按鍵連接 GPIO0 接腳**，ESP32-S 開發板內建一顆 USB 晶片，用來負責 USB 與 UART

之間的信號轉換，並且使用 GPIO1、GPIO3 當做 UART 介面。依 ESP32 開發板的版本不同，有 CH340C 及 CP2102 兩種 USB 晶片，第一次使用 ESP32 開發板，須先安裝 USB 晶片驅動程式。

4-3-2 Arduino Uno 與 ESP32 特性比較

如表 4-10 所示 Arduino Uno 與 ESP32 特性比較，ESP32 開發板內建藍牙及 Wi-Fi 模組，可以直接用來開發物聯網產品。Arduino Uno 開發板必須再外接藍牙模組（如 HC-05 模組）或 Wi-Fi 模組（如 ESP-01 模組），才能開發物聯網產品。相較於 Arduino Uno 板，**ESP32 板具有容量大、速度快、體積小、開發成本低等優點**。

表 4-10 Arduino Uno 與 ESP32 特性比較

特性	Arduino Uno	ESP32
工作電壓	5V	3.3V
MCU	AVR ATmega328P	Tensilica Xtensa LX6
核心	單核 20MHz	雙核 160 / 240MHz
資料寬度	8 位元	32 位元
Flash ROM	32KB	4~32MB
SRAM	16KB	520KB
GPIO	14 支	20 支
UART	1 組	1 組
I2C	1 組	2 組
SPI	1 組	2 組
PWM	6 支	20 支
ADC	6 組（10 位元）	16 組（12 位元）
DAC	無	2 組
Wi-Fi	無	802.11b/g/n
藍牙	無	BLE 4.2
內建電容觸摸感測器	無	10 組
內建溫度感測器	無	1 組
內建霍爾感測器	無	1 組

· CHAPTER ·

06

雲端運算

6-1 認識雲端運算

在物聯網中需要更快的處理器與更多的儲存容量，來處理大量感知元件所產生的大量數據資料。這些數據資料必須經由雲端（cloud）伺服器來進行分析、運算及管理，才能成為有用的共享資訊。什麼是雲端呢？在資訊技術中的**「雲」泛指網路，「端」泛指任何可以連上網路的通訊設備，例如物品、手機、平板、筆電及電腦等**。在「端」的使用者（Client）只要知道如何透過網際網路來得到相應的服務，不需要了解位於「雲」上的基礎設施細節及相關專業知識。如同家中的電信網路，當我們需要用電時，只要將設備的電源插頭插進插座，插座即成為電信網路的「端」設備。

雲端運算（Cloud Computing）一詞是近年來相當熱門的科技新知識，它不是一種全新的資訊技術，而是一種概念。美國國家標準與技術研究院（National Institute of Standards and Technology，簡稱 NIST）定義雲端運算的運作模式，是透過**連上網際網路，以隨處、隨時、隨選所需的方式來存取共享運算資源**（如網路、伺服器、儲存、應用程式及服務等）。只需要最少的管理作業與供應商涉入，就能快速配置與發布運算資源。

6-1-1 雲端運算服務模式

如圖 6-1 所示雲端運算服務模式的基礎架構，NIST 定義雲端運算服務模式有**軟體即服務**（Software as a Service，簡稱 SaaS）、**平台即服務**（Platform as a Service，簡稱 PaaS）及**架構即服務**（Infrastructure as a Service，簡稱 IaaS）三種。

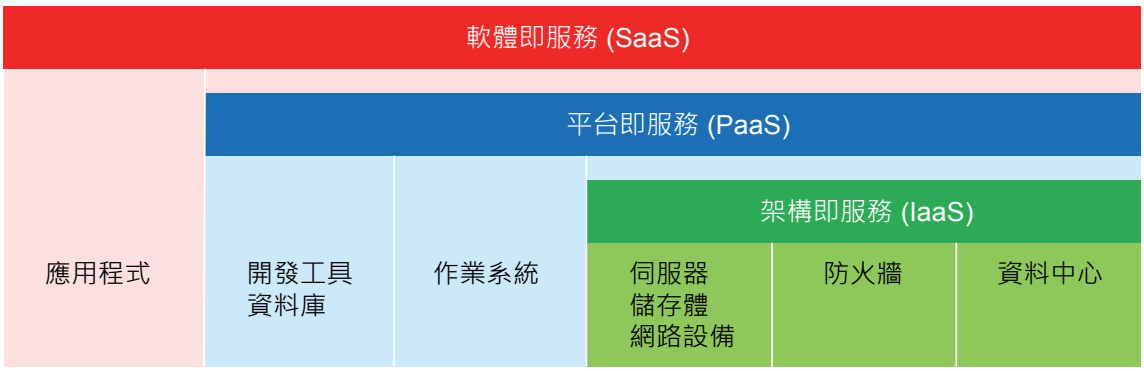


圖 6-1　雲端運算服務模式的基礎架構

動手做：利用手機 App 查詢雲端氣象資訊

一 功能說明

下載並安裝 App 程式 APP/ch6/WiFiWeather.aia，開啟如圖 6-9 所示 App 程式查詢雲端氣象站資料頁面。頁面顯示 ThingSpeak 平台上已建立的 dht11 及 weather 兩個通道。按下 查詢 鈕將 ThingSpeak 平台上 dht11 及 weather 兩個通道的最新（最後新增）氣象資料顯示在手機畫面。

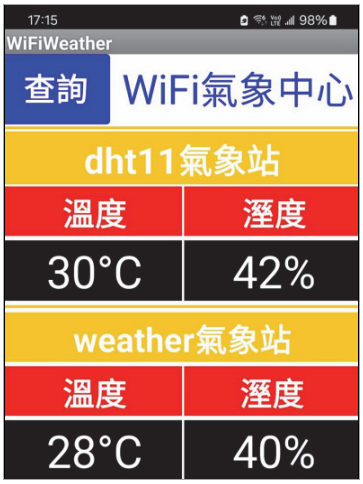


圖 6-9　查詢雲端氣象站資訊 App 程式 WiFiWeather.aia

用戶端使用 App 來查詢 ThingSpeak 平台上的通道資料，必須要有通道的 Channel ID 及 Read API 金鑰，再使用 GET 方法來取得通道上的 JSON 資料。一個 JSON 資料 **{"created_at":"2024-10-08T08:44:48Z","entry_id":3,"field1":"31","field2":"45"}**，會被 App 程式解碼（created_at 2024-10-08T08:44:48Z）、（entry_id 3）、（field1 31）及（field2 45）等四個元素的清單。第 1 個元素為建立日期、第 2 個元素為建立順序、第 3 個元素為溫度值、第 4 個元素為溼度值。如果 App 程式再進一步解碼（field1 31）會產生 field1 及 31 兩個元素，31 即為溫度。解碼（field2 45）會產生 field2 及 45 兩個元素，45 即為相對溼度。

二 電路接線圖

無。

三 程式

無。

四 App 介面配置及說明：APP/ch6/WiFiWeather.aia

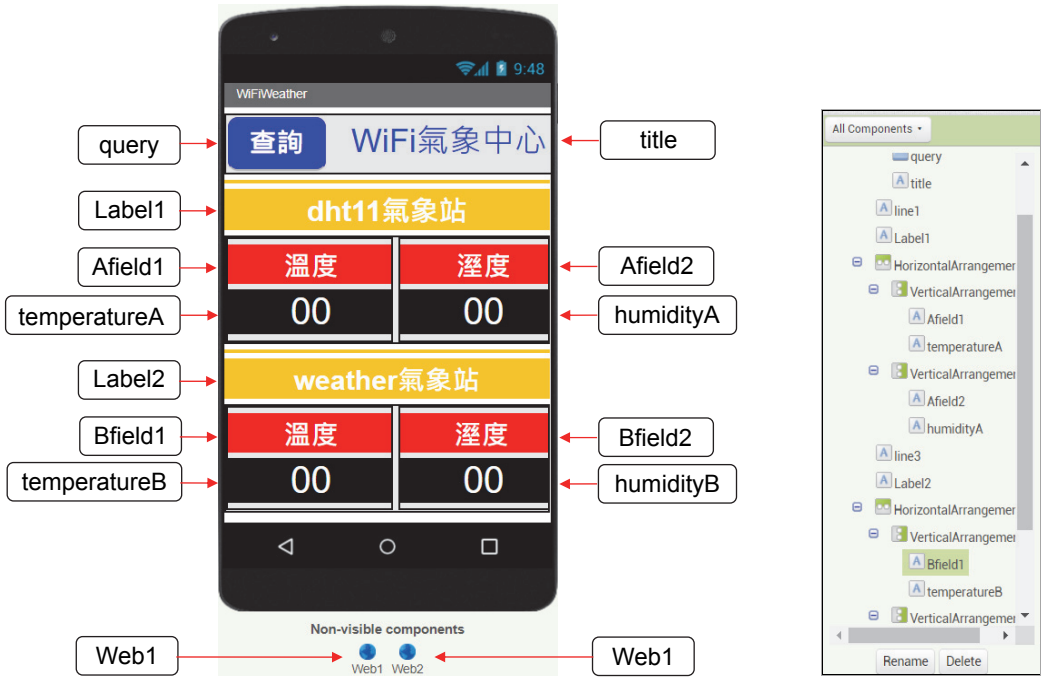
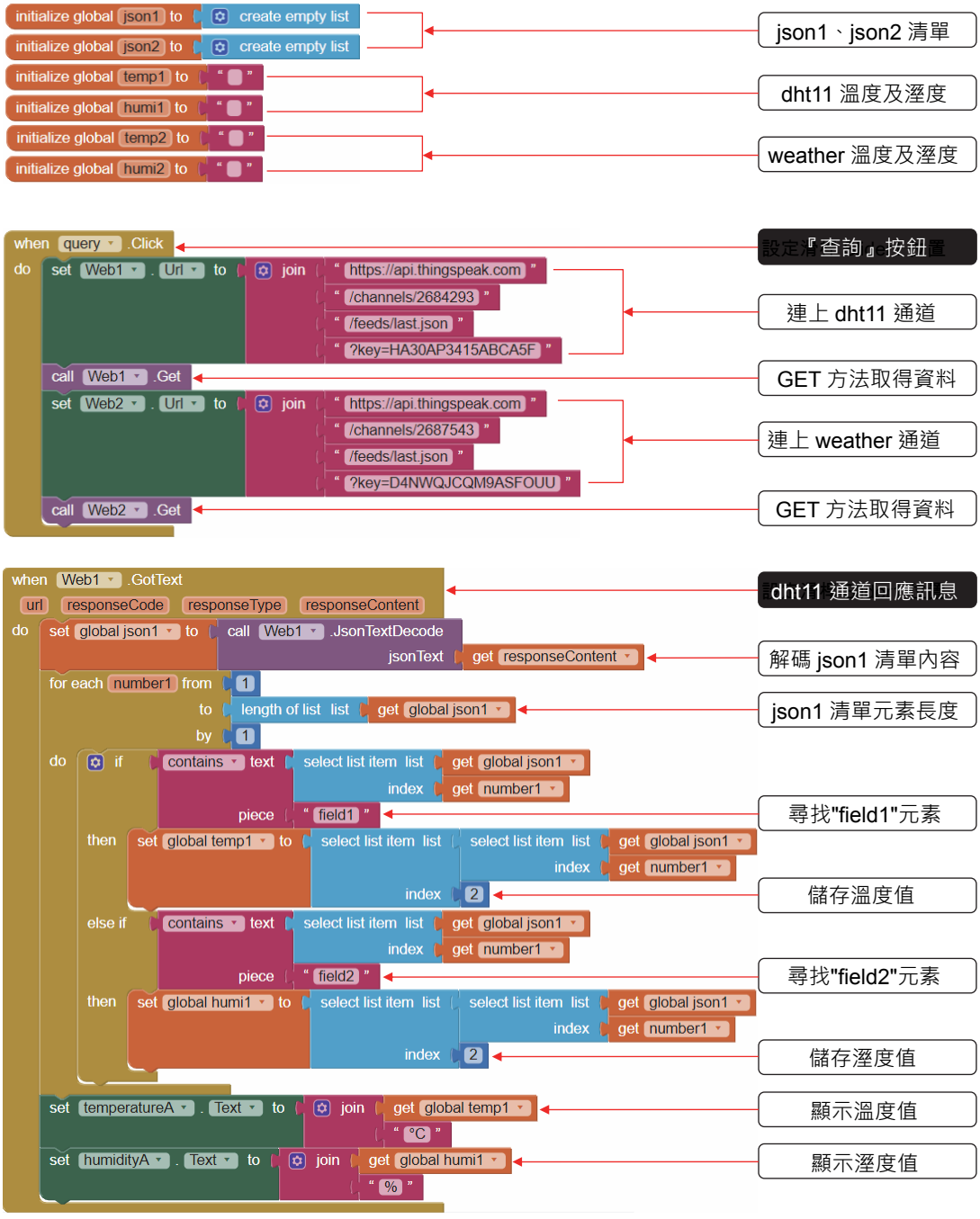


圖 6-10　App 程式 WiFiweather 介面配置

表 6-1　App 程式 WiFiweather 元件屬性說明

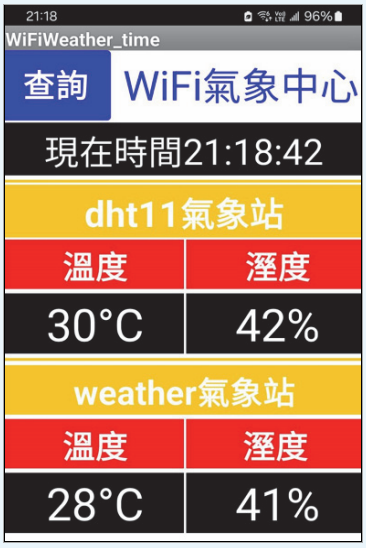
名稱	元件	主要屬性說明
query	Button	Height=Automatic,Width=30 percent,FontSize=30
Title	Label	Height=Automatic,Width=70 percent,FontSize=36
Label1	Label	Height=Automatic,Width=Fill parent,FontSize=32
Afield1、Bfield2	Label	Height=Automatic,Width=50 percent,FontSize=30
Afield2、Bfield2	Label	Height=Automatic,Width=50 percent,FontSize=30
temperatureA、B	Label	Height=Automatic,Width=50 percent,FontSize=40
HumidityA、B	Label	Height=Automatic,Width=50 percent,FontSize=40
Label2	Label	Height=Automatic,Width=Fill parent,FontSize=32

五 App 方塊功能說明：APP/ch6/WiFiWeather.aia

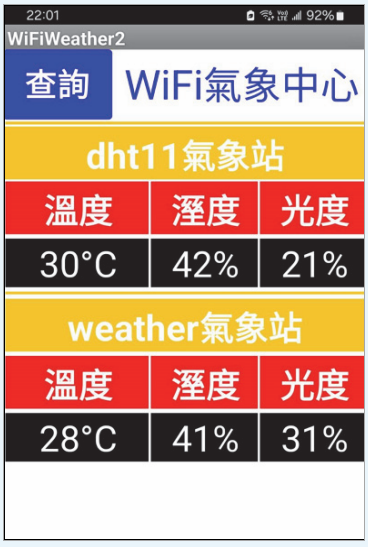


練習

1. 接續範例，使用如圖 6-11(a) 所示手機 App 程式 WiFiWeather_time 介面配置，新增顯示「現在時間」。



(a) WiFiWeather_time 介面配置



(b) WiFiWeather2 介面配置

圖 6-11　App 程式介面配置

2. 接續範例，使用如圖 6-11(b) 所示手機 App 程式 WiFiWeather2 介面配置，新增顯示「光度」數據。

動手做：利用 Arduino 查詢雲端氣象資訊

一　功能說明

如圖 6-12 所示 ThingSpeak 平台使用 JSON 格式傳回的氣象資訊。JSON 是以階層方式來表示資料內容，使用「**索引鍵-值**」組來儲存資訊，每個「索引鍵-值」組再以逗號分隔，並且以大括號"{ }"將所有「索引鍵-值」組括起來。

本例要顯示溫度及溼度，可先搜尋第一個字元 f 所在位置（position，簡稱 pos），相對位置 pos+9、pos+10 為溫度值，相對位置 pos+23、pos+24 為溼度值，相對位置 pos+37、pos+38 為光度值。

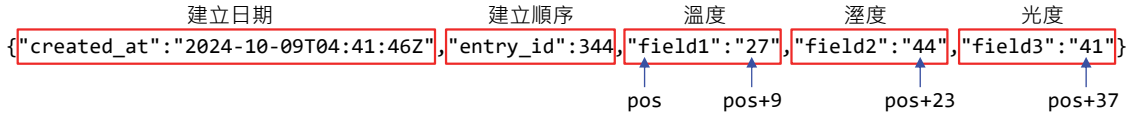


圖 6-12　ThingSpeak 平台使用 JSON 格式傳回的氣象資訊

如圖 6-14 所示 Arduino 查詢雲端氣象資訊電路接線圖，當電源重置時，Wi-Fi 指示燈 L1 閃爍三次。ESP8266 模組初始化並且設定為 STA 模式，開始進行如圖 6-13 所示使用 Arduino 查詢雲端氣象資訊的 Wi-Fi 連線動作。連線成功加入 AP 後，Wi-Fi 指示燈 L1 恆亮。按下圖 6-14 所示 SW1 鍵，建立與 ThingSpeak 平台連線，查詢雲端氣象資訊並顯示在「序列埠監控視窗」中。

圖 6-13　使用 Arduino 查詢雲端氣象資訊的 Wi-Fi 連線動作

二 電路接線圖

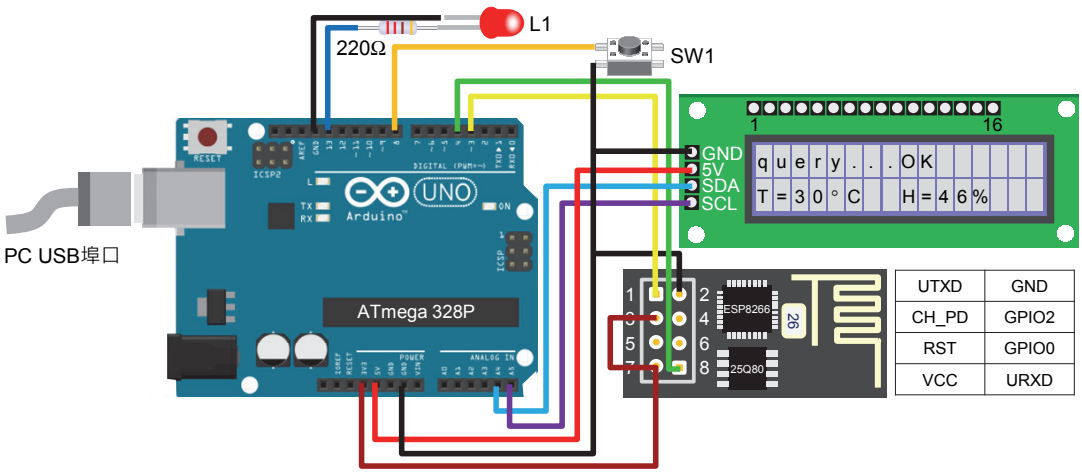


圖 6-14　Arduino 查詢雲端氣象數據電路接線圖

三 程式：ch6-2.ino

```
#include <SoftwareSerial.h>                    //載入 SoftwareSerial 函式庫。
#define SSID "輸入您的 AP 帳號"                 //AP 帳號。
#define PASSWD "輸入您的 AP 密碼"               //AP 密碼。
#define IP "184.106.153.149"                   //thingspeak.com 的 IP 位址。
SoftwareSerial ESP8266(3,4);                   //設定 D3 為 RXD，D4 為 TXD。
const int sw=8;                                //D8 連接按鍵開關 SW1。
const int WiFiled=13;                          //D13 連接 Wi-Fi 指示燈 L1。
boolean FAIL_8266 = false;                     //true:連線失敗，false:連線成功。
int i;                                         //迴圈次數。
char c;                                        //ESP8266 的接收字元。
unsigned long timeout;                         //逾時計時器。
String cmd;                                    //AT 指令。
String message;                                //ESP8266 的接收數據。
String temp,humi;                              //溫度值及溼度值。
char buf[3];                                   //緩衝區。
//初值設定
void setup()
{
    pinMode(WiFiled,OUTPUT);                   //設定 D13 為輸出埠。
    digitalWrite(WiFiled,LOW);                 //關閉 Wi-Fi 指示燈 L1。
    ESP8266.begin(9600);                       //設定 ESP8266 傳輸率為 9600bps。
```

6-2-5 認識 ESP32 I2C

積體匯流排電路（Inter-Interated Circuit，簡稱 I²C），唸法是 I 平方 C （I squared C）。I²C 由飛利浦公司在 1980 年代所開發，是一種短距離、快速二線同步傳輸協定，包含串列時脈腳（Serial Clock，簡稱 SCL）及串列資料腳（Serial Data，簡稱 SDA）二線。I²C 主要用於主機板、嵌入式系統等主控端與低速周邊設備之間的通信。**ESP32 的 I²C 介面使用 GPIO21 爲 SDA 接腳，GPIO22 爲 SCL 接腳**。

I²C 主從結構是由一個主控設備（如 ESP32）及多個從端設備組成，所有 I²C 設備的時脈腳 SCL 及資料腳 SDA，必須分別連接在一起。I²C 主控設備發出一個 **7 位元長度的位址**，產生 128（=2^7）個不同的位址編號，用來識別每一個具有唯一位址編號的從端設備。除了部分位址編號保留給特殊用途使用之外，剩餘 112 個位址編號皆可以使用。本例所使用的 I²C 串列 LCD 顯示器位址編號為 0x27。

動手做：ESP32 控制 I2C 串列 LCD 顯示字元

一 功能說明

如圖 6-15 所示 ESP32 控制 I2C 串列 LCD 顯示字元電路接線圖。GPIO21 連接 LCD 的 SDA 接腳，GPIO22 連接 LCD 的 SCL 接腳，並且使用 ESP32 的 5V 電源供電給 LCD 模組。電源重啟後，在 LCD 第 0 列顯示「Hello, world!」。

二 電路接線圖

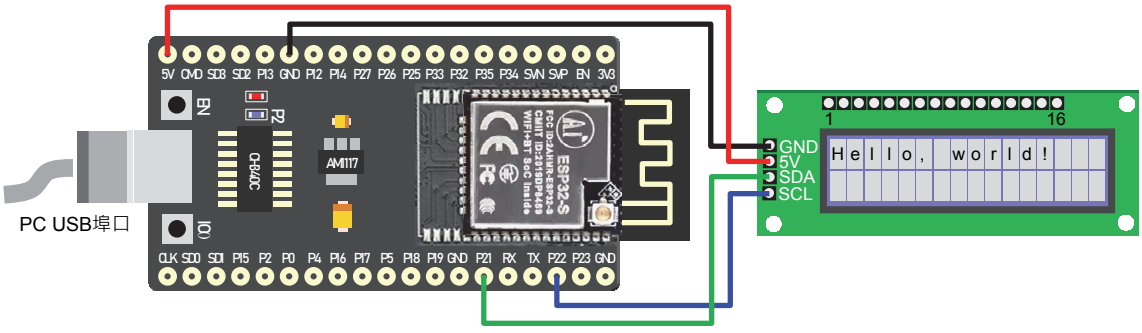


圖 6-15 ESP32 控制 I2C 串列 LCD 顯示字元電路接線圖

三 程式：ch6-3.ino

```
#include <LiquidCrystal_I2C.h>                  //載入 LiquidCrystal_I2C 函式庫。
LiquidCrystal_I2C lcd(0x27,16,2);               //建立 LCD 物件，使用 1602 串列 LCD。
//初值設定
void setup()
{
    lcd.init();                                 //初始化 LCD。
    lcd.backlight();                            //開啟 LCD 背光。
    lcd.setCursor(0,0);                         //設定 LCD 座標在第 0 行、第 0 列。
    lcd.print("Hello, world!");                 //顯示訊息：Hello,world!。
}
//主迴圈
void loop(){
}
```

練習

1. 接續範例，在第 0 列中間顯示「1234567890」，第 1 列中間顯示「Hello, world!」。
2. 接續範例，在第 0 列中間顯示「1234567890」，第 1 列顯示「Hello, world!」且每 200 毫秒由右向左移動一個字元。

6-2-6 URI 與 URL

統一資源識別器（Uniform Resource Identifier，簡稱 URI）是用來識別如 HTML 檔案、程式碼、影片、圖片等資源，是一種**通用概念**。而統一資源定位器（Uniform Resource Locator，簡稱 URL）指定資源所在位址，是 URI 的**具體實現方式**。URL 又稱為網址，主要是由**協定**（protocol）、**網域**（domain）及**文件路徑**三個部分組成。常用協定如超文字傳輸協定（Hypertext Transfer Protocol，簡稱 HTTP）、超文字傳輸安全協定（Hypertext Transfer Protocol Secure，簡稱 HTTPS）、檔案傳輸協定（File Transfer Protocol，簡稱 FTP）及簡單郵件傳輸協定（Simple Mail Transfer Protocol，簡稱 SMTP）等。

如圖 6-16 所示 URL 網址，協定是 https://、網域是 thingspeak.mathworks.com、而文件路徑是/channels/2684293/api_keys。

圖 6-16　URL 網址

6-2-7　HTTPClient 類別

HTTPClient 類別讓 ESP32 可以使用 HTTP 協定存取外網伺服器，將 URI 所識別的資源**傳送 HTTP 請求**和**接收 HTTP 回應**。如表 6-2 所示 HTTPClient 類別的常用方法，首先使用 HTTPClient 建立物件，再使用 begin()方法啟用連接。接著使用 GET()方法發起 GET 請求，再接收伺服器所傳回的狀態碼，確認是否請求成功。如果請求成功，再使用 getString()方法讀取伺服器所傳回的數據。

表 6-2　HTTPClient 類別的常用方法說明

方法	功能	參數說明
begin(String URL)	啓用連接。	URL：網址。
GET()	發起 GET 請求。	無。
getString()	讀取伺服器的回應數據。	無。
getSize()	伺服器的回應數據長度。	無。
end()	結束目前的連接。	無。

▸動手做：ESP32 Wi-Fi 雲端氣象站

一　功能說明

如圖 6-17 所示 ESP32 Wi-Fi 雲端氣象站電路接線圖，ESP32 開發板建立 Wi-Fi 連線，並且設定為 STA 模式。開始進行 Wi-Fi 連線時，指示燈 L1 閃爍三次，同時 LCD 顯示連線狀態。Wi-F 連線成功後，指示燈 L1 恆亮，同時 LCD 顯示 AP 所配置的私用 IP 位址。

Wi-Fi 連線成功後，再利用 HTTPClinet 函式庫連線 ThingSpeak 雲端運算平台 dht11 通道，每分鐘上傳一次 DHT11 感測器所感測到最新的溫度及溼度值。成功上傳 ThingSpeak 雲端運算平台後，LCD 顯示建立順序、溫度及溼度等數據資料。

二 電路接線圖

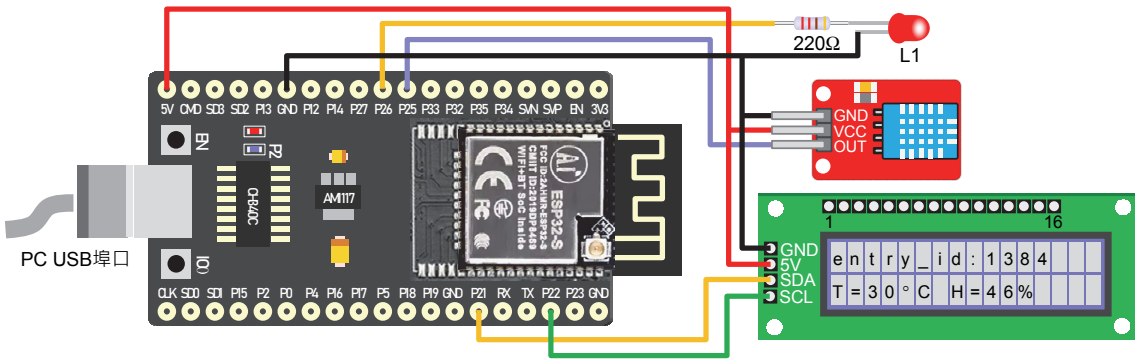


圖 6-17　ESP32 Wi-Fi 雲端氣象站電路接線圖

三 程式：ch6-4.ino

```
#include<WiFi.h>                                    //載入 Wi-Fi 函式庫。
#include <HTTPClient.h>                             //載入 HTTPClient 函式庫。
#include <LiquidCrystal_I2C.h>                      //載入 LiquidCrystal_I2C 函式庫。
#include <Adafruit_Sensor.h>                        //載入 Adafruit_Sensor 函式庫。
#include <DHT.h>                                    //載入 DHT 函式庫。
#include <DHT_U.h>                                  //載入 DHT_U 函式庫。
#define DHTPIN 25                                   //GPIO25 連接 DHT11 輸出。
#define WIFIled 26                                  //GPIO26 連接 WiFi 指示燈。
#define DHTTYPE DHT11                               //使用 DHT11。
DHT dht(DHTPIN, DHTTYPE);                           //建立 DHT11 物件。
LiquidCrystal_I2C lcd(0x27,16,2);                   //建立 LCD 物件。
const char ssid[]="輸入您的 AP 名稱";                 //AP 名稱
const char pwd[]="輸入您的 AP 密碼";                  //AP 密碼
unsigned long timeout=0;                                    //逾時計時器。
String url="https://thingspeak.mathworks.com/       //URL 網址。
            update?key=283OKW1GSMN0LFUD";           //寫入 API 金鑰。
//初值設定
void setup()
{
    dht.begin();                                    //初始化 DHT11。
    lcd.init();                                     //初始化 LCD。
    lcd.backlight();                                //開啟 LCD 背光。
```

```
    lcd.clear();                                  //清除 LCD 顯示內容。
    pinMode(WIFIled,OUTPUT);                      //設定 GPIO26 為輸出埠。
    digitalWrite(WIFIled,LOW);                    //關閉 Wi-Fi 指示燈。
    for(int i=0;i<3;i++)                          //Wi-Fi 指示燈閃爍三次。
    {
        digitalWrite(WIFIled,HIGH);               //Wi-Fi 指示燈亮。
        delay(200);                               //延遲 0.2 秒。
        digitalWrite(WIFIled,LOW);                //Wi-Fi 指示燈暗。
        delay(200);                               //延遲 0.2 秒。
    }
    WiFi.mode(WIFI_STA);                          //設定為 STA 模式
    WiFi.begin(ssid,pwd);                         //建立與 AP 的 Wi-Fi 連線。
    clearROW(0);                                  //清除 LCD 第 0 列內容。
    lcd.print("WiFi connecting.");                //LCD 顯示訊息：連線中。
    clearROW(1);                                  //清除 LCD 第 1 列內容。
    while(WiFi.status()!=WL_CONNECTED)            //尚未連線?
    {
        lcd.print(".");                           //等待連線。
        delay(500);                               //延遲 0.5 秒。
  }
  clearROW(0);                                    //清除 LCD 第 0 列內容。
  lcd.print("WiFi connected. ");                  //顯示訊息：已連線。
  clearROW(1);                                    //清除 LCD 第 1 列內容。
  lcd.print(WiFi.localIP());                      //顯示配置的私用 IP 位址
  digitalWrite(WIFIled,HIGH);                     //Wi-Fi 指示燈恆亮。
}
//主迴圈
void loop()
{
    if((millis()-timeout)>=60000)                 //已超過 1 分鐘?
    {
        timeout=millis();                         //儲存系統時間。
        float h = dht.readHumidity();             //讀取環境溫度。
        float t = dht.readTemperature();          //讀取相對溼度。
        if (isnan(t) || isnan(h))                 //溫度或溼度數據不正確?
        {
            clearROW(1);                          //清除 LCD 第 1 列內容。
```

```
            lcd.print("DHT11 error!");          //顯示訊息：DHT11 讀取失敗。
        }
        else                                    //讀取溫度及溼度數據正確。
        {
            clearROW(0);                        //清除 LCD 第 0 列內容。
            lcd.print("ThingSpeak....");        //顯示訊息:連線 ThingSpeak 平台。
            String url1=url+"&field1="+(int)t+"&field2="+(int)h;
            HTTPClient http;                    //建立 HTTP 物件。
            http.begin(url1);                   //開始進行 HTTP 連接。
            int httpCode = http.GET();          //GET 請求。
            if (httpCode == HTTP_CODE_OK)       //請求成功，伺服器回應代碼 200。
            {
                clearROW(0);                            //清除 LCD 第 0 列內容。
                String payload = http.getString();      //讀取伺服器回應訊息。
                lcd.print("entry_id:");
                lcd.print(payload);                     //顯示伺服器回應訊息。
            }
            else                                //GET 請求失敗。
            {
                clearROW(0);                    //清除 LCD 第 0 列內容。
                lcd.print("GET failed.");       //顯示訊息：GET 請求失敗。
            }
            http.end();                         //結束 HTTP 連接。
            clearROW(1);                        //清除 LCD 第 1 列內容。
            lcd.print("T=");                    //顯示訊息：T=。
            lcd.print((int)t);                  //顯示溫度值。
            char degree=0xdf;                   //溫度單位符。
            lcd.print(degree);                  //顯示溫度單位。
            lcd.print("C");                     //溫度單位符。
            lcd.setCursor(8,1);                 //設定 LCD 座在第 8 行、第 1 列。
            lcd.print("H=");                    //顯示訊息：H=。
            lcd.print((int)h);                  //顯示溼度值。
            lcd.print('%');                     //顯示溼度單位。
        }
    }
}
```

```
//LCD 顯示內容清除函式
void clearROW(int row)
{
    lcd.setCursor(0,row);                    //設定 LCD 座標在第 0 行、第 row 列。
    for(int i=0;i<16;i++)                    //清除第 row 列內容。
        lcd.print(' ');                      //清除一個字元。
    lcd.setCursor(0,row);                    //設定 LCD 座標在第 0 行、第 row 列。
}
```

練習

1. 接續範例，新增光敏電阻模組測量光線亮度，光敏電阻模組的輸出 AO 連接於 ESP32 開發板的類比輸入 ADC1_CH5（GPIO33）。
2. 接續範例，在 ThingSpeak 平台上建立第二個通道 weather，同樣新增溫度（Field1）、溼度（Field2）及光度（Field3）三個欄位。

動手做：利用 ESP32 查詢雲端氣象資訊

一 功能說明

如圖 6-12 所示 ThingSpeak 平台使用 JSON 格式傳回的氣象資訊，JSON 是以階層方式來表示資料內容，使用「**索引鍵-值**」組來儲存資訊。每個「索引鍵-值」組再以逗號分隔，並且以大括號"{ }"將所有「索引鍵-值」組括起來。本例要顯示溫度及溼度，可先搜尋第一個 f 所在位置 pos，相對位置 pos+9、pos+10 為溫度值，相對位置 pos+23、pos+24 為溼度值，相對位置 pos+37、pos+38 為光度值。

如圖 6-18 所示 ESP32 查詢雲端氣象資訊電路接線圖，當電源重置時，Wi-Fi 指示燈 L1 閃爍三次。ESP32 模組初始化，設定為 STA 模式，連線成功加入 AP 後，Wi-Fi 指示燈 L1 恆亮。

手指觸摸如圖 6-18 所示 TOUCH7（GPIO27）觸控板一下，ESP32 開始建立與 ThingSpeak 平台連線，查詢雲端氣象平台 dht11 通道最新（最後一筆）上傳的氣象資訊，並且顯示在 LCD 顯示器中。