

# 修平科技大學

## 資訊網路技術系

### 額溫雲端量測系統

指導老師：張瑞淇

學生：陳倉斌 BN107062

張宸維 BN107001

中華民國一一一年一月五日

# 修平科技大學

## 資訊網路技術系

### 額溫雲端量測系統

學生：陳倉斌 BN107062

張宸維 BN107001

指導老師：\_\_\_\_\_

口試老師：\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

中華民國一一一年一月五日

## 摘要

自從 2019 年底新冠肺炎疫迄今，因台灣超前佈署，加上國人對防疫的警慎，得以有完美的防疫成績，不過難免有漏網之魚。這次的專題研究是針對個人沒有手機可以簡訊實聯制紀錄足跡時，我們將額溫雲端量測系統與物聯網技術結合，將體溫與足跡訊息送至雲端資料庫，降低防疫期間手寫紙本個人資料時，人與人的間接接觸所帶來的風險，也節省紙本資料管理的成本，使防疫的措施能持續進步。

本專題以 WEMOS D1 Mini (ESP8266) 來實現額溫雲端量測系統，採用 Arduino IDE 開發環境撰寫程式，實現物聯網技術，應用於防疫控管。採用高效率的 MQTT 數據傳輸協定，將非接觸式額溫感測器的溫度資訊上傳至雲端的伺服器，雲端的伺服器採用視覺化的程式設計工具 Node-Red，使管理者可以透過伺服器的 UI 介面監控溫度(額溫)，並且可以將量測資訊儲存於雲端資料庫。

未來的研究方向可擴大物聯網技術的應用，將額溫雲端量測系統與 RFID 系統整合，如此將可以更精確分析與控管防疫，對防疫與人身安全一定有很大的幫助。

關鍵字：物聯網、MQTT、Node-RED、Line-Notify

## 目錄

第 1 章	緒論	1
1.1	研究動機	1
1.2	研究目的	1
1.3	研究方法	2
第 2 章	硬體開發模組簡介	4
2.1	紅外線額溫感測器 MLX90614ESF	4
2.2	WEMOS D1 mini 硬體	6
2.2	OLED 液晶顯示器	9
2.3	鋰電池電源	10
2.4	按鈕開關與蜂鳴器	11
2.5	紅外線接近感測器	12
第 3 章	開發環境與伺服器軟體安裝	14
3.1	Arduino IDE 安裝	14
3.2	Arduino IDE 開發環境介面	19
3.3	Arduino IDE 的擴充	20
第 4 章	雲端伺服器的安裝	26
4.1	Node-Red 安裝與設定	26
4.2	MQTT Broker 安裝與設定	31
4.3	MySQL 安裝與設定	37
第 5 章	額溫雲端量測系統	41
5.1	額溫系統硬體結構	41
5.2	額溫感測器程式	42
5.3	雲端伺服器 Node-Red 程式 & LineNotify 超溫通知	47
5.4	系統整合測試結果	51
第 6 章	結論與心得	56

6.1 結論 .....	56
6.2 心得 .....	57
參考文獻 .....	58

# 第1章 緒論

## 1.1 研究動機

自從前年新冠肺炎流行以來，政府、企業、學校等單位都大張旗鼓的實施防疫措施，其中最主要的就是戴口罩、掃 QR code 實聯制和量體溫，不管是大型百貨公司或是小型超商都會要求顧客要掃 QR code 實聯制和量體溫，如果沒有帶手機的時候就會是用紙本的方式留下姓名和聯絡電話，時常看到沒有帶手機或是不會掃 QR code 實聯制的民眾擠在店家門口等著在筆記本上寫下個人資料，導致耗費不少時間，因此我們想到了這個額溫雲端量測系統，讓門口負責量額溫的人員來輸入沒辦法掃 QR code 實聯制的顧客資料，不但能省下民眾等待寫字的時間，又能解決民眾字跡潦草導致辨識不易的情形，將顧客資料保存在雲端資料庫裡，如果政府須要調查有來過此商家的民眾時，就能很容易的將資料整理好繳交出來，讓大家辦事都能更有效率。

## 1.2 研究目的

本專題透過物聯網技術實現額溫雲端量測系統，可以偵測額頭溫度，來監控每個人的額溫情況，也透過 WiFi 連接將資料上傳至雲端。當防疫期間有人沒辦法掃 QR code 實聯制時，我們能以物聯網技術來輸入他個人的資料和額溫，假如已達到發燒 38 度時，額溫雲端量測系統的

會發出紅色警告，並發送訊息到管理者手機的 Line，提醒管理者有顧客發燒的情形出現，不但能省下民眾排隊等寫字的時間，還能確保每個體都是安全的，給予每個人安心的環境，讓大眾更放心。

### 1.3 研究方法

本專題採用的微控板是內建 ESP8266 的 WEMOS D1 mini 來實現額溫雲端量測系統，如下圖 1-1。ESP8266 的內建 2.4GHz 頻段的 IEEE 802.11n 介面，可以透過 WIFI 將額溫資料功能回傳到雲端伺服器。

系統架構如圖 1-2 所示，遠端的主要元件由額溫感測器、ESP8266 WEMOS D1 mini 模組，透過 MQTT 通訊協定將資料送至 Node-Red 伺服器。經由 Node-Red 提供的網頁介面，可以在任何的瀏覽器上顯示額溫的資訊。Node-Red 伺服器上也安裝有 MySQL 資料庫伺服器，透過 NodeRed 的資料庫模組功能，可將額溫資料寫入資料庫，或查詢資料。



圖 1-1 WEMOS D1 mini

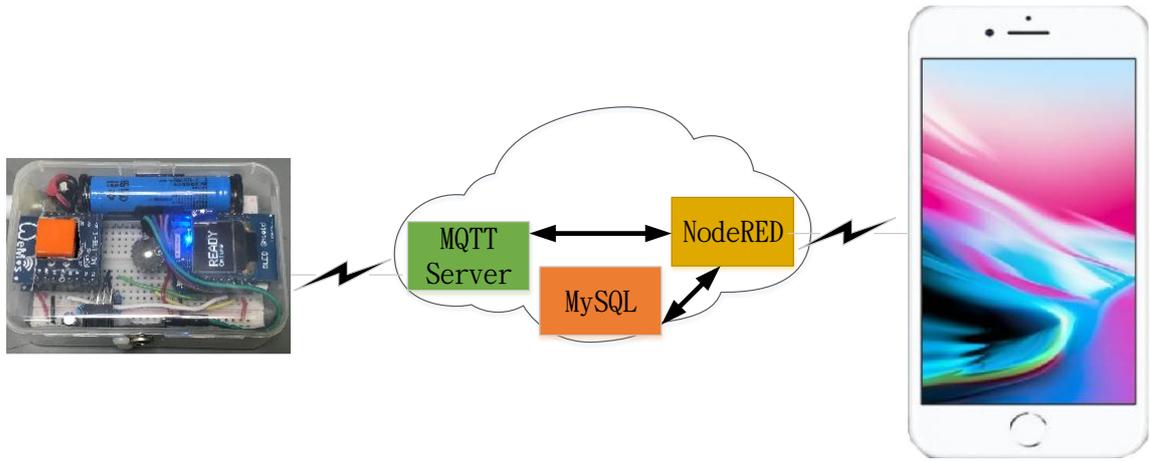


圖 1-2 系統架構圖

## 第2章 硬體開發模組簡介

### 2.1 紅外線額溫感測器 MLX90614ESF

本專題的重點就是紅外線額溫感測器，採用的是 Melexis 生產的 MLX90614ESF-BAA 是為非接觸式溫度感測而設計的紅外溫度計。內部 17 位 ADC 和強大的 DSP 有助於 MLX90614 的高精度和高分辨率。它具有大量應用，包括體溫測量和運動檢測。MLX90614 提供兩種輸出方法：PWM 和 I2C。10 位 PWM 輸出提供  $0.14^{\circ}\text{C}$  的分辨率，而 I2C 介面具有  $0.02^{\circ}\text{C}$  的分辨率。MLX90614 的出廠校準溫度範圍很廣：環境溫度為  $-40$  至  $85^{\circ}\text{C}$ ，物體溫度為  $-70$  至  $382.2^{\circ}\text{C}$ 。測量值是傳感器視場中所有物體的平均溫度。MLX90614 在室溫下的標準精度為  $0.5^{\circ}\text{C}$ 。由於新冠肺炎的影響，需要大量的額溫槍，MLX90614 已經全面缺貨，由國外進口的成本已經高於 29.95 美金。售價由新冠肺炎前的 250 元漲價至 900 元左右。

在內部，MLX90614 是兩個設備的配對：一個紅外熱電堆檢測器和一個信號調節應用處理器。根據 Boltzman 定律，任何不低於絕對零 ( $0^{\circ}\text{K}$ ) 的物體都會發出與其溫度成正比的紅外光譜中的光 (非人眼可見)。MLX90614 內部的特殊紅外熱感應材料在其視場中發射出多少紅外能量，並產生與之成比例的電信號。由紅外熱感應材

料產生的電壓由應用處理器的 17 位 ADC 接收，然後調節，然後傳遞給微控制器。

MLX90614 產生兩個溫度測量值：一個物體和一個環境讀數。所述對象溫度是你會從傳感器所期望的非接觸式測量，而環境溫度的措施對傳感器的模具的溫度。環境對於校準數據很有用，但是我們讀數的真實內容將來自目標溫度測量。物體溫度的測量範圍為-70 至 382.2°C (-94 至 719.96°F)，而環境溫度讀數的範圍為-40 至 125°C。環境溫度和物體溫度的分辨率均為 0.02°C。



圖 2-1 紅外線額溫感測器 MLX90614-BAA

本專題採用的 MLX90614 是藍板包裝的 MLX90614-BAA，如圖 2-1，市面上有許多不同包裝的模組，如圖 2-1 有離散零件、藍板、塑膠盒等包裝。它具有 3.6V-5V 的工作電壓，並帶有一個紅外傳感器和一個內部光學濾波器。MLX90614BAA 的光學視角是 90°。

## 2.2 WEMOS D1 mini 硬體

ESP8266 具有較寬的工作溫度範圍，能夠在工業環境中始終如一地運行。該晶片具有高度集成的片上功能和最少的外部分立元件數，可提供可靠性，緊湊性和堅固性。ESP8266 設備，可穿戴電子設備和物聯網應用而設計，結合多種專有技術實現了低功耗。省電架構具有三種操作模式：活動模式，睡眠模式和深度睡眠模式。這使得電池供電的設計可以運行更長時間。ESP8266 控制板處理器核心:32 位元 Tensilica Xtensa LX106，運作時派 80~160MHz。網路部分 Wi-Fi 網路:802.11b/g/n 協定與 WPA/WPA2 加密來做使用，ESP8266 工作電壓:3.3V，網路連線時最大消耗電流:215mA，GPIO(通用輸出/入埠):16 個，最大輸入電壓 3.6V，最大輸出電流 12mA，還有類比輸入腳:輸入準位 0~1V，10 位元解析度，支援的介面:UART,I<sup>2</sup>C,SPI。

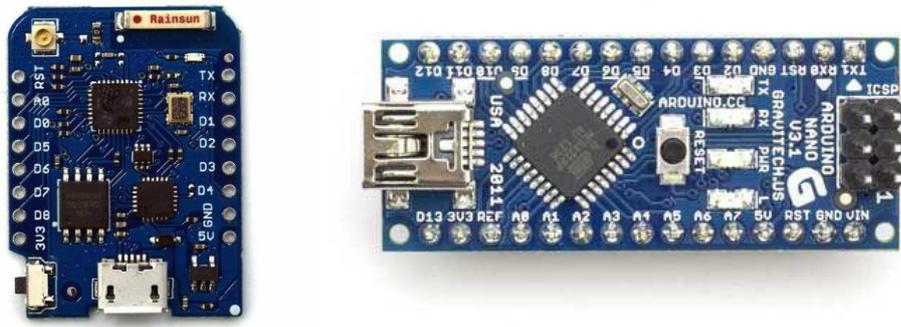


圖 2-2 WEMOS D1 mini 與 Arduino Nano 的外觀

本專題採用比較精簡版的 ESP8266 模組：WEMOS D1 mini，如圖 2-2 左側。它把 ESP-12F 大部分腳位引出了，可以當作一塊擁有 WiFi 功能的 Arduino 板，直接使用 Arduino IDE 來開發，使用 ESP8266 等相關函式庫快速方便，WEMOS D1 mini 最大的特色就是可以當積木使用，堆疊多種模組，可搭配模組有 OLED 液晶顯示組、鋰電池充放電模組、按鈕開關模組、BMP 氣壓模組、W2812 全彩 LED 模組、1 路 5V 繼電器模組，以上模組的搭配讓 WEMOS D1 mini 有更多不同的使用方式。也可以透過 WEMOS D1 mini 特色基於 ESP-8266 完全相容 Arduino，可使用 Arduino IDE 來程式設計，還有引腳介紹 11 x I/O 引腳、1 x ADC 引腳（輸入範圍 0-3.2V）。支援 OTA 無線上傳、安裝硬體後，可直接採用 Arduino IDE 開發，跟 UNO、Nano 一樣操作，對已經熟悉 Arduino IDE 的開發者是一大福音。

表 2-1 WEMOS D1 mini 規格表

微控制器	ESP-8266 12F
工作電壓	3.3V
數字 I/O 引腳	11
類比輸入引腳	1(最大輸入:3.2V)
長度	34.2 毫米

WEMOS D1 mini 的腳位與協定如圖 2-3 所示，除了電源、重置接腳外，共有 11 個通用輸入輸出接腳，這些接腳也分別支援 UART、I2C、SPI 與類比輸入，可以組成一個多功能的裝置。詳細腳位的說明如表 2-2 所示，

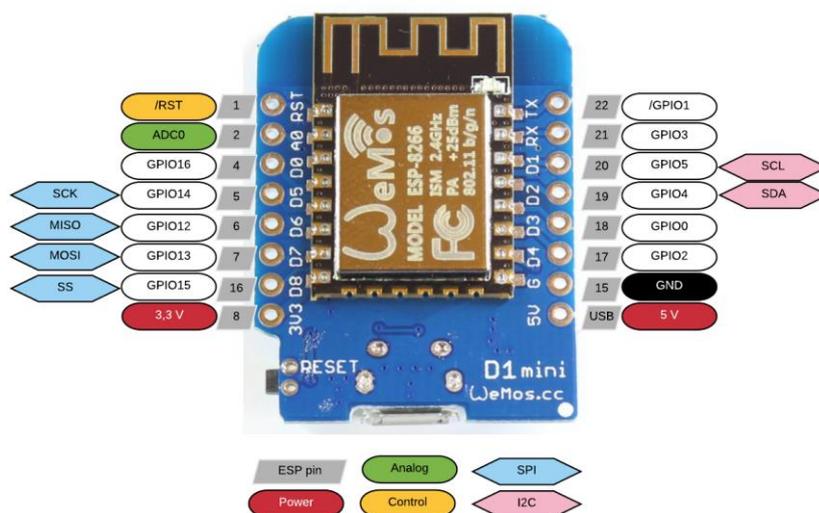


圖 2-3 WEMOS D1 mini 的腳位與協定

表 2-2 WEMOS D1 mini 腳位

板子腳位	ESP8266 腳位	功能
D0	GPIO16	IO
D1	GPIO5	IO, SCL
D2	GPIO4	IO, SDA
D3	GPIO0	IO, 內建 10K 上拉電阻
D4	GPIO2	IO, 內建 10K 上拉電阻與 LED
D5	GPIO14	IO, SCK
D6	GPIO12	IO, MISO
D7	GPIO13	IO, MOSI
D8	GPIO15	IO, SS, 內建 10K 上拉電阻
TX	TXD	UART 送端
RX	RXD	UART 收端
A0	ADC	類比輸入 (0~3.3V)

## 2.2 OLED 液晶顯示器

額溫量測器除了將額溫上傳至雲端，我們也在額溫量測器上採用 OLED 液晶顯示器即時顯示額溫，可以與雲端顯示的數據做時比對。我們採用 0.66 英吋的 OLED Shield 解析度為 64x48，與 ESP8266 的通信介面為 I2C。如圖 2-4，OLED Shield 可與 WEMOS D1 mini 堆疊以節省

空間。其規格如下：

螢幕尺寸： 64x48 像素（0.66 “橫向）

工作電壓： 3.3V

驅動 IC： SSD1306

介面： I2C（I2C）

I2C 地址： 0x3C 或 0x3D



圖 2-4 OLED 文字顯示器

### 2.3 鋰電池電源

由於額溫量測器需要手持操作，必須用電池才能方便使用。由於鋰電池的蓄電力強，沒有記憶效應與壽命長，因此本專題採用鋰電池充電升壓放電模組與作為額溫量測器的電源模組，如圖 2-5。鋰電池充電升壓

放電模組也可與 WEMOS D1 mini 堆疊，以節省空間。鋰電池的部分可以採用扁平或圓柱形的鋰電池模組，本專題因為額溫量測器空間的因素，採用編號為 14500 的圓柱形鋰電池。鋰電池充電升壓放電模組具備升壓功能，只需將鋰電池連接到該屏蔽層，可為整個 WEMOS D1 mini 系統提供電源。當電池電量耗盡時，您不需要拔掉電池，只需使用 USB 介面的 5V 電源進行充電。

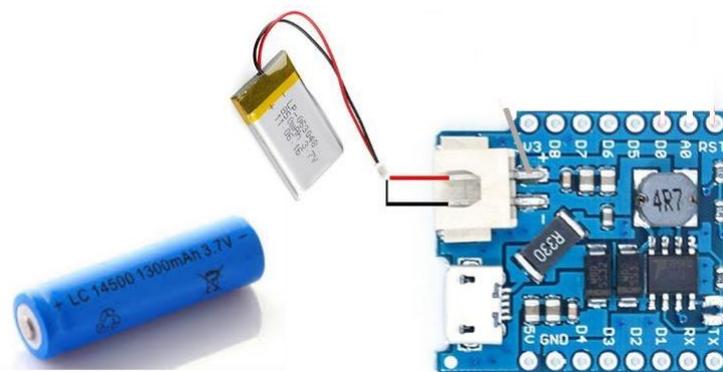


圖 2-5 鋰電池電源模組

## 2.4 按鈕開關與蜂鳴器

額溫量測器需要有介面方便使用者的操作，因此我們設計了兩個裝置讓使用者操作，分別是按鈕開關與蜂鳴器，如圖。當使用者按下按鈕開關時，額溫量測器就開始進行溫度量測動作，量測結束時，蜂鳴器會發出「嗶」一聲，代表量測已經完成，並將溫度顯示於液晶顯示器與傳送至雲端伺服器。如果量測的溫度超過 38 度 C，蜂鳴器會發出「嗶、嗶、嗶」三聲，代表量得異常溫度。



圖 2-6 按鈕開關與蜂鳴器

## 2.5 紅外線接近感測器

使用按鈕開關來控制額溫量測有一個很大的缺點就是不容易控制額溫量測器與額頭的距離，導致量測的結果會出現很地的誤差，因此本專題後來追加了紅外線接近感測器 ST188，如圖 2-7，以感測額頭的紅外線強度比背景的紅外線強度超過一定值後，自動進行量測。當量測完成後，蜂鳴器會發出「嗶」一聲，將溫度顯示於 OLED 液晶顯示器，並自動將額溫上傳至雲端伺服器。



圖 2-7 紅外線接近感測器

紅外線接近感測器 ST188 內建有紅外線發光二極體與光電晶體，其偏壓電路如圖 2-8。當發光二極體發射出的光照射在額頭上反射回光電晶體，若反射光線夠強，光電晶體將導通進入飽和模式 AD0 的電壓會下降。AD0 接至 WEMOS D1 mini 的類比輸入接腳 A0，可以透過 ESP 8266 內建的類比數位轉換器，量到紅外光的強度。當紅外光愈強，代表與額頭距離愈近。由於不同環境下，紅外光強度的背景值有很大差異，本專題採用相對紅外光強度來辨別距離的改變，以降低量測上的誤差。

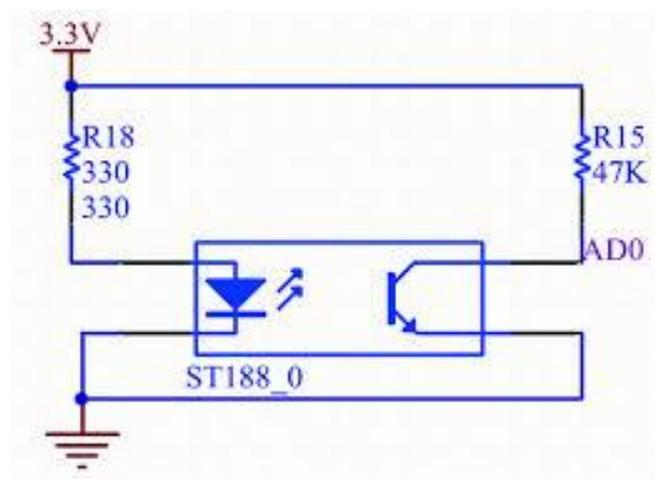


圖 2-8 ST188 偏壓電路

# 第3章 開發環境與伺服器軟體安裝

## 3.1 Arduino IDE 安裝

在瀏覽器上輸入 Arduino-Home 目前最新版本是 1.8.7，為了穩定與相容性考量，採用 1.8.5 版，如下圖 3-1 ~ 圖 3-10 安裝流程。

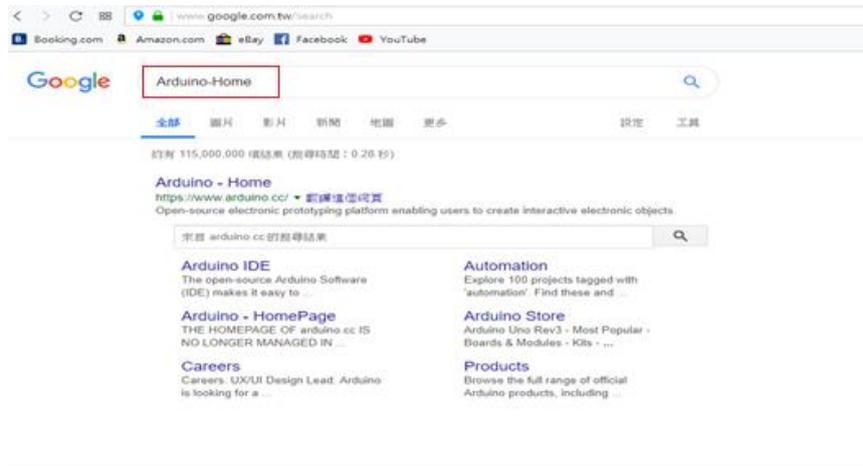


圖 3-1 搜尋輸入 Arduino

點入 Arduino IDE

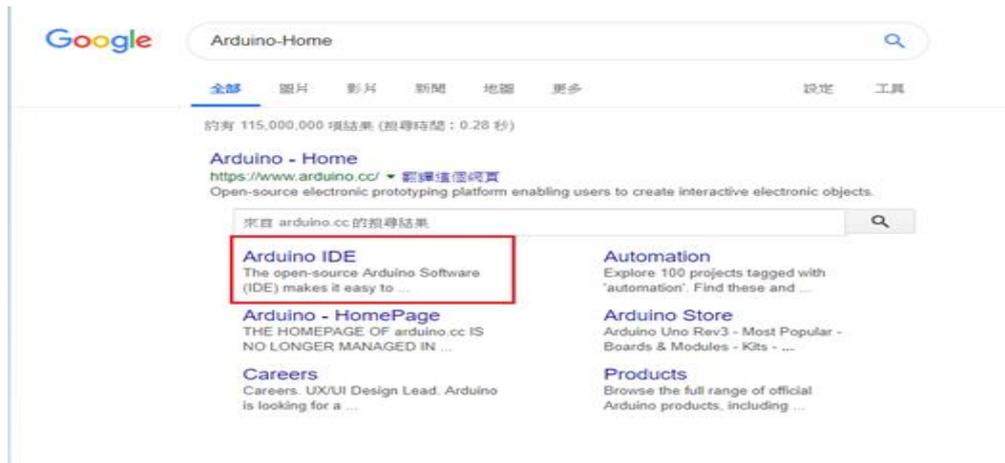


圖 3-2 點擊網頁

## 進入先前版本



圖 3-3 先前版本

## Windows 下載(選擇自己的電腦系統的軟體)



圖 3-4 選擇系統軟體

## Arduino - 捐贈 - 點選只是下載



圖 3-5 下載 Arduino

啟動安裝程式後，出現如圖 3-6 的畫面。按下同意 I Agree。接著出現如圖 3-7 的畫面，按下下一步 Next。

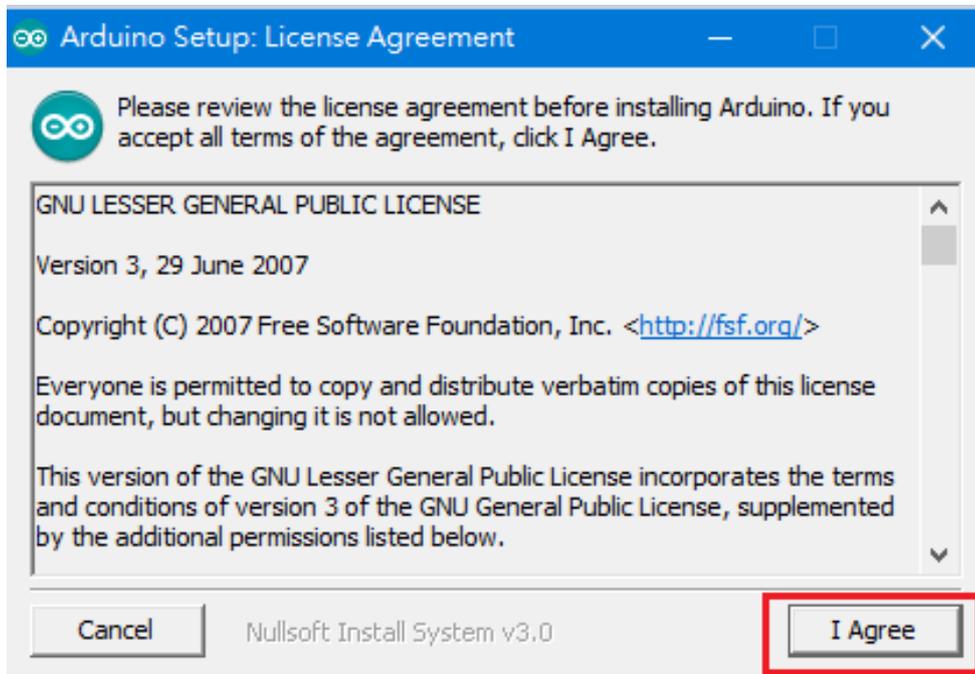


圖 3-6 安裝流程-1

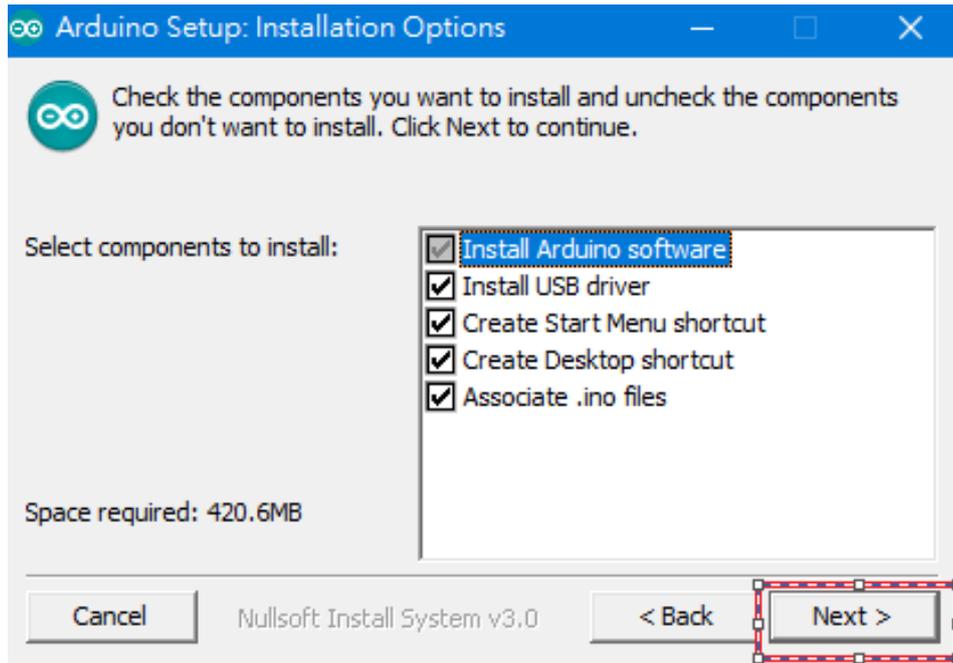
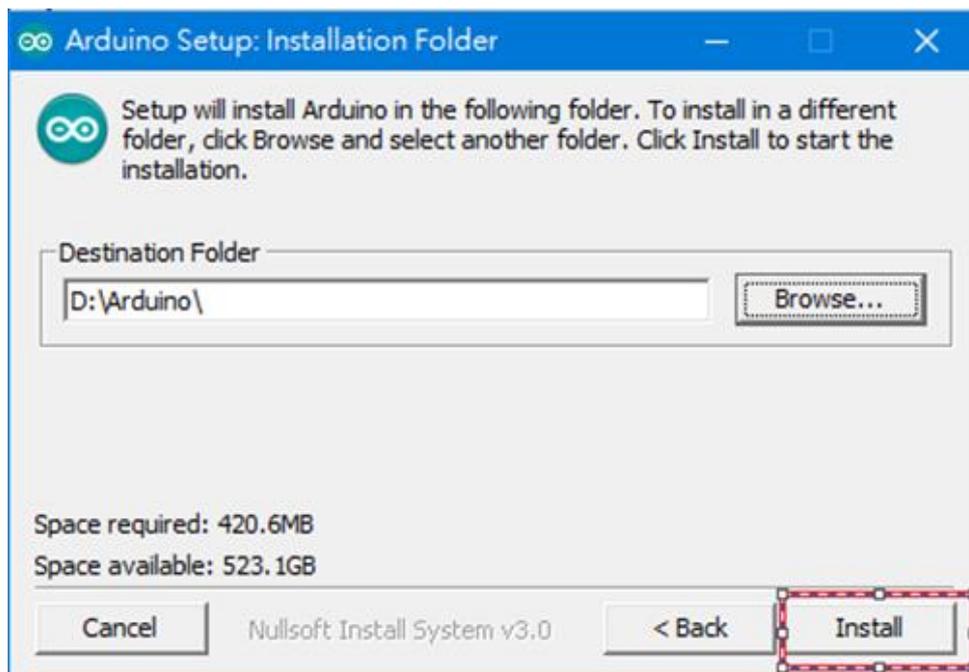


圖 3-7 安裝流程-2

接著出現如圖 3-8，設定於你想要安裝的位置。設定好後，點選安裝 Install，就開始進行安裝。安裝過程會如圖 3-9 呈現安裝進度，最後如圖 3-10 出現代表已安裝完成，按下關閉 Close，結束安裝。



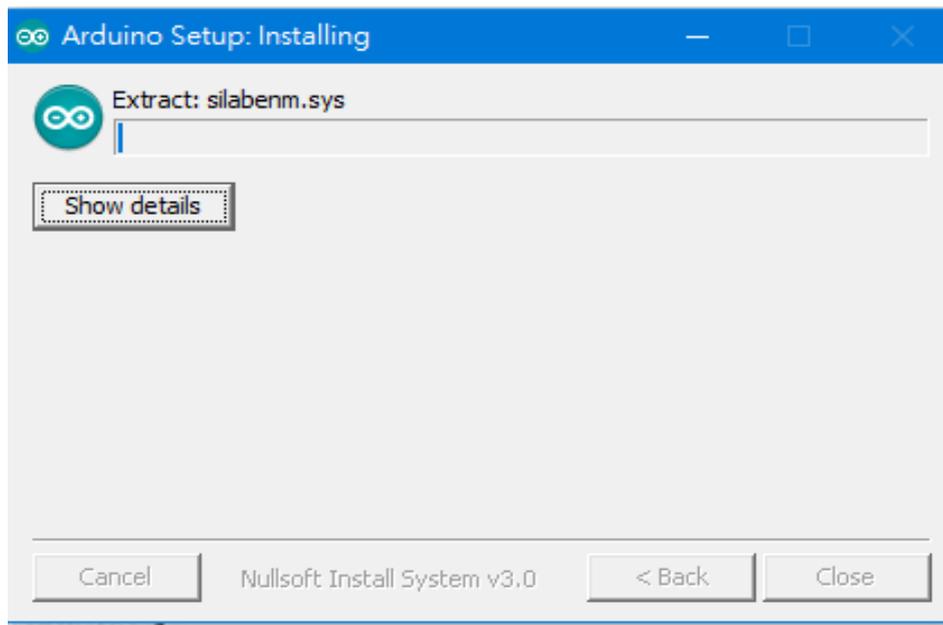


圖 3-8 安裝畫面

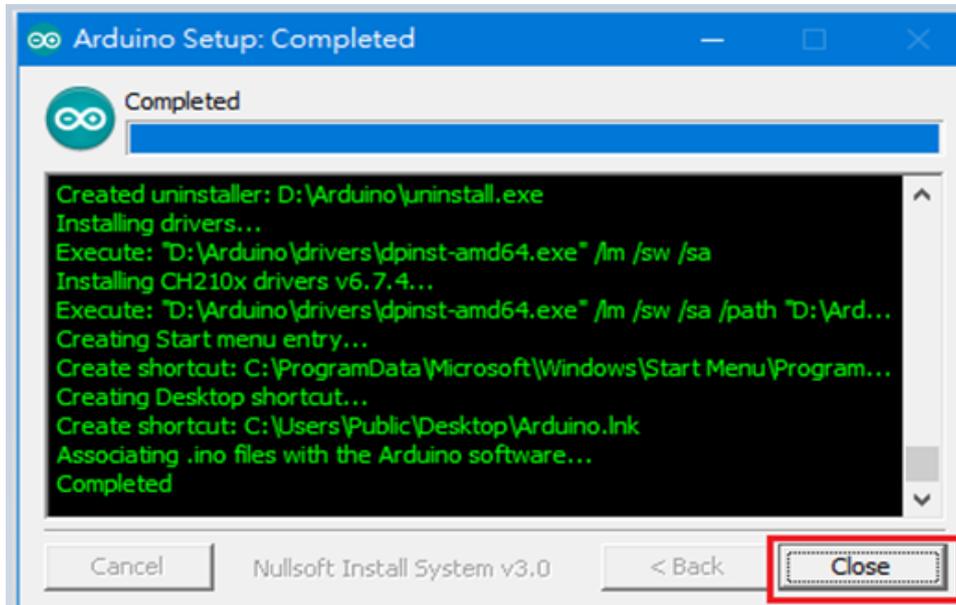


圖 3-9 安裝完成

### 3.2 Arduino IDE 開發環境介面

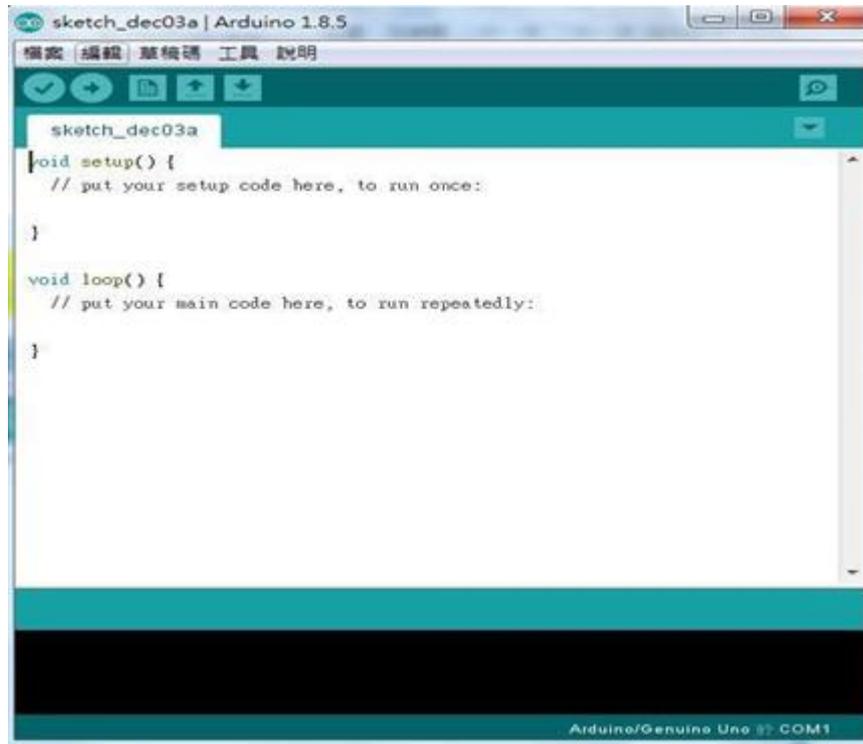


圖 3-10 Arduino 介面

表 3-1 Arduino IDE 常用功能

	Verity 檢查參數設定或引入程式是否產生錯誤。
	Upload 程式編譯，將程式碼透過 USB 介面燒錄至 Arduino 控制板。
	New 產生新的專案。
	開啟專案，顯示在同一頁面上。點擊不同專案便顯示不同專案。
	Save 腳本專案。
	Serial Monitor 開啟監視器頁面，監視 Arduino I/O 介面。

### 3.3 Arduino IDE 的擴充

為了使 Arduino IDE 也支援 ESP8266，必須安裝擴充套件。步驟如下。如圖 3-11 下載 Arduino IDE 18.5 免安裝版本以及按右鍵選擇解壓縮打開程式。3.1 節的標準安裝無法將 Arduino IDE 並拷貝至另一台電腦執行，為了使 Arduino IDE 18.5 為可攜式的程式，我們可以下載免安裝的 Arduino IDE 壓縮檔，在解壓後的目錄下建立一個資料夾「Portable」，擴充 ESP8266 套件將安裝於此目錄底下。

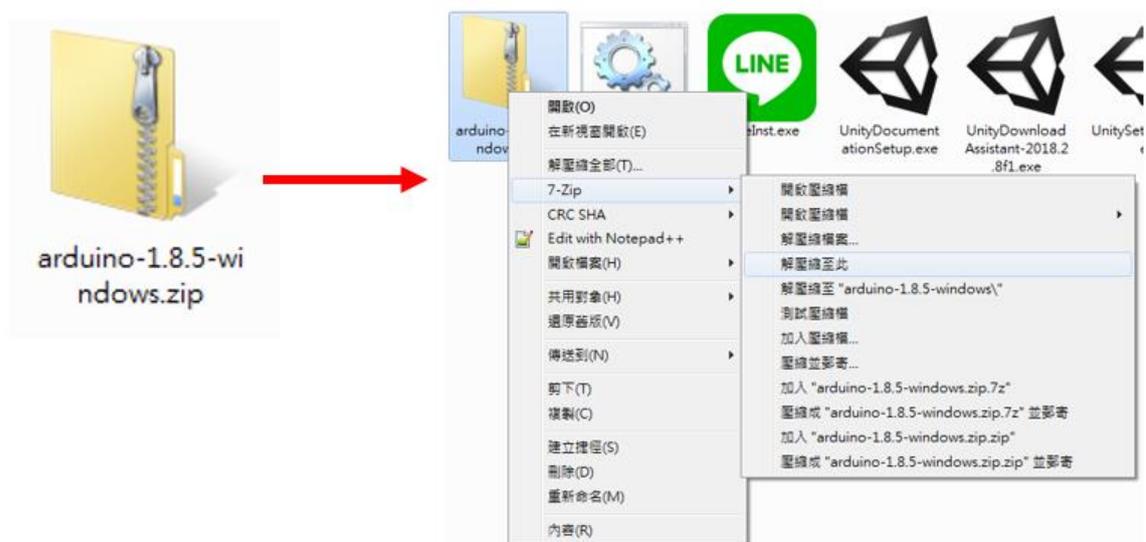


圖 3-11 Arduino IDE 解壓縮

設定 Arduino IDE 偏好如圖 3-12，開啟 Arduino 1.8.5 版本，點擊檔案>點選偏好設定>設定程式碼位子>額外開發板管理網址>，輸入下列超連結，設定偏好檔位置。

[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

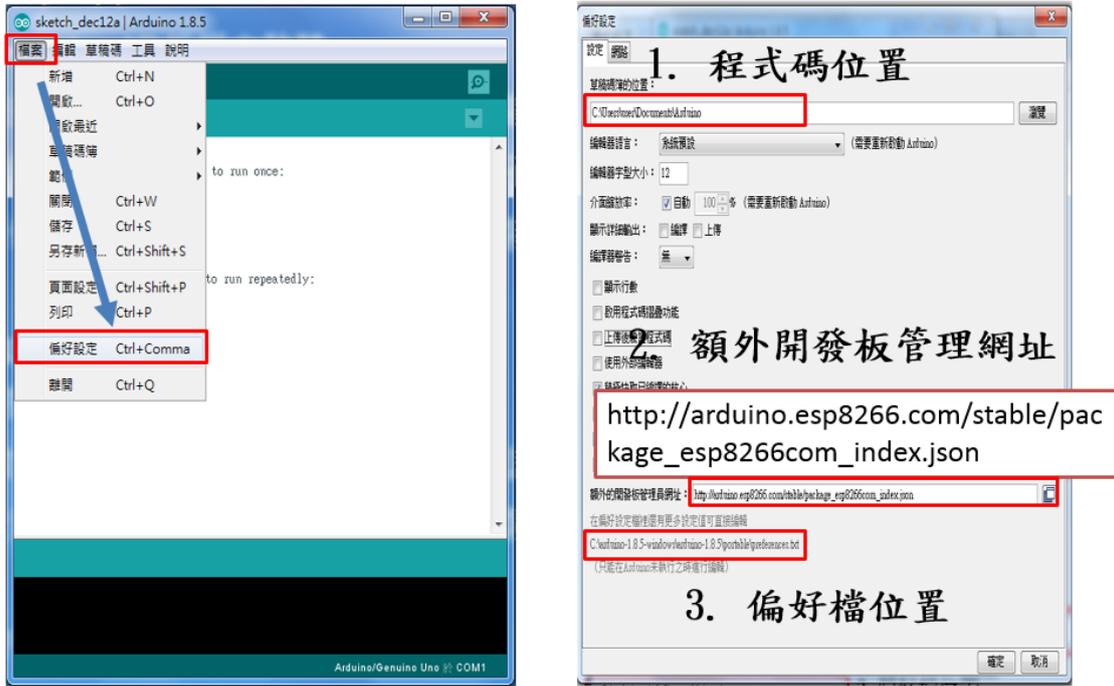


圖 3-12 Arduino IDE 偏好

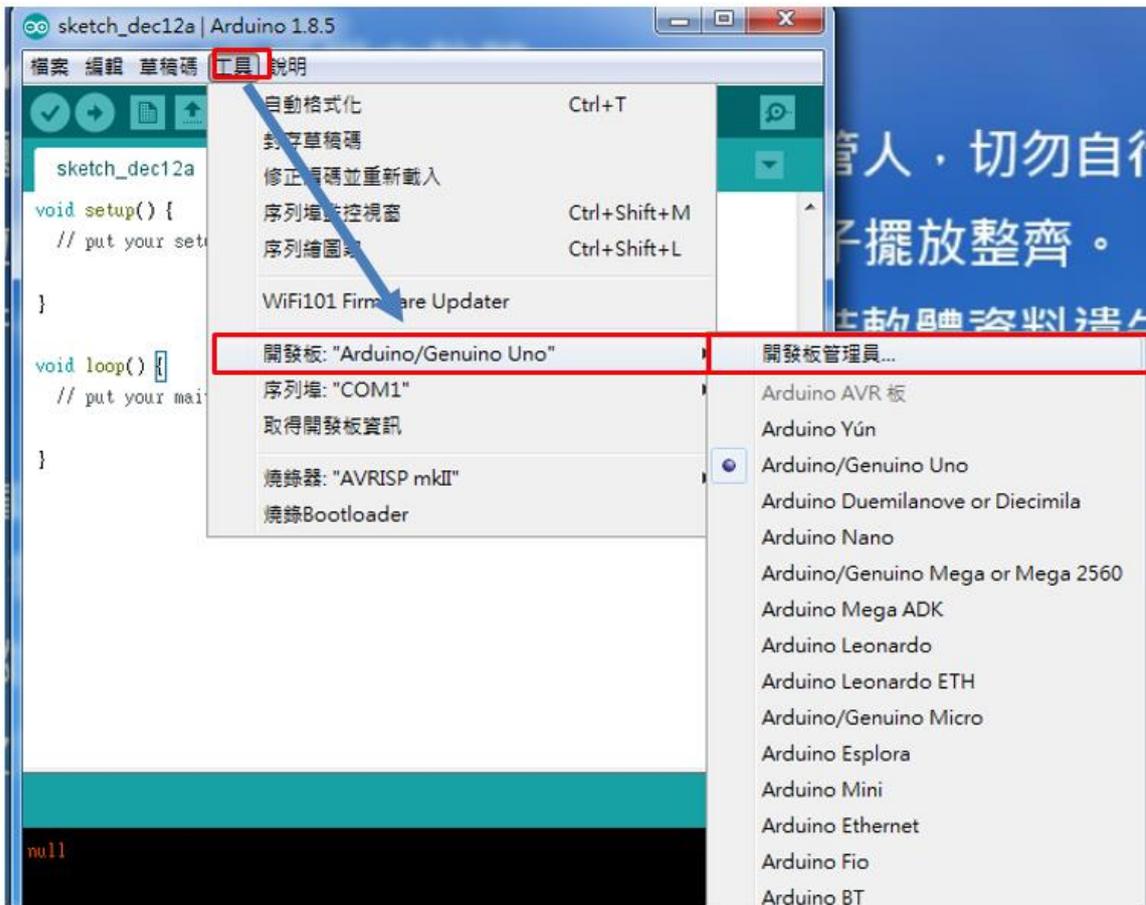


圖 3-13 Arduino IDE 開發環境設定

在 Arduino IDE 開發環境設定新增 ESP8266 開發版，如圖 3-13

Arduino IDE 選擇工具列>開發板”Arduino/Genuino Uno”>開發板管理員。

在開發板管理員中搜尋 ESP8266>選擇 24.2 版，執行安裝，如下

圖 3-14 安裝完成。



圖 3-14 安裝 ESP8266 板子

在如圖 3-15 Arduino IDE 選擇工具列>開發板”LOLIN(WEMOS) D1 mini ，WEMOS 改名為 LOLIN。如圖 3-16 工具就會有開發板硬體資訊顯示。當把 WEMOS D1 mini 用 USB 連接到電腦的 USB Port 時，如圖 3-17 控制台>裝置管理員(或電腦管理)>連接埠，就可以看到電腦與 WEMOS D1 mini 的連線是正常的，我們就可以開始用 Arduino IDE 開發 ESP8266 的專案。

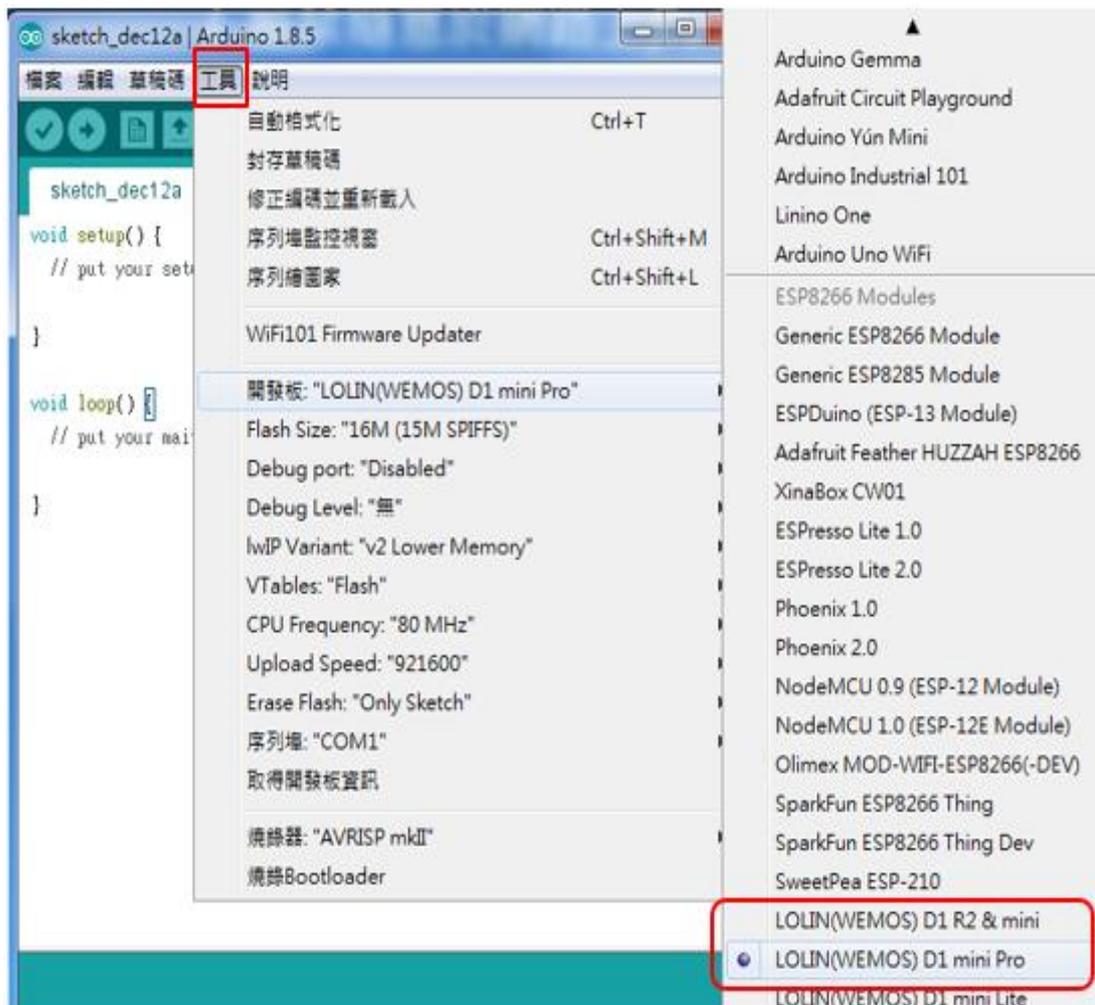


圖 3-15 選擇適當的 ESP8266 板子

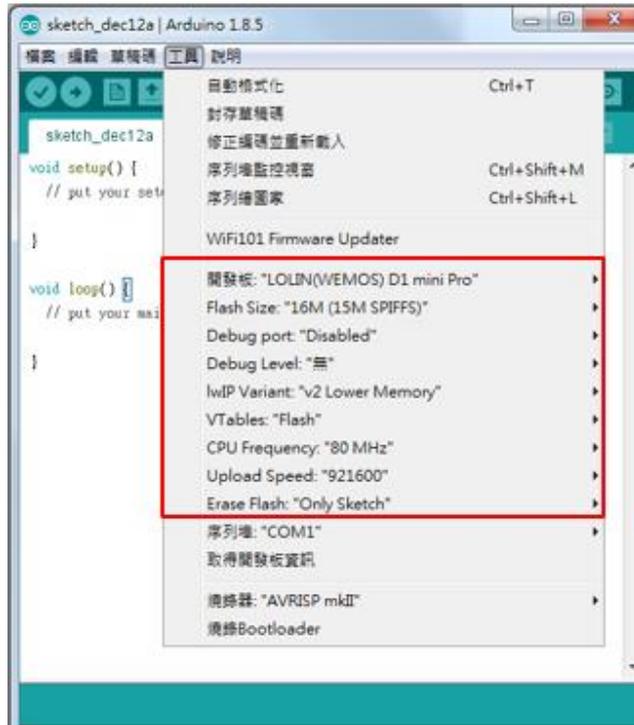


圖 3-16 開發板的硬體資訊

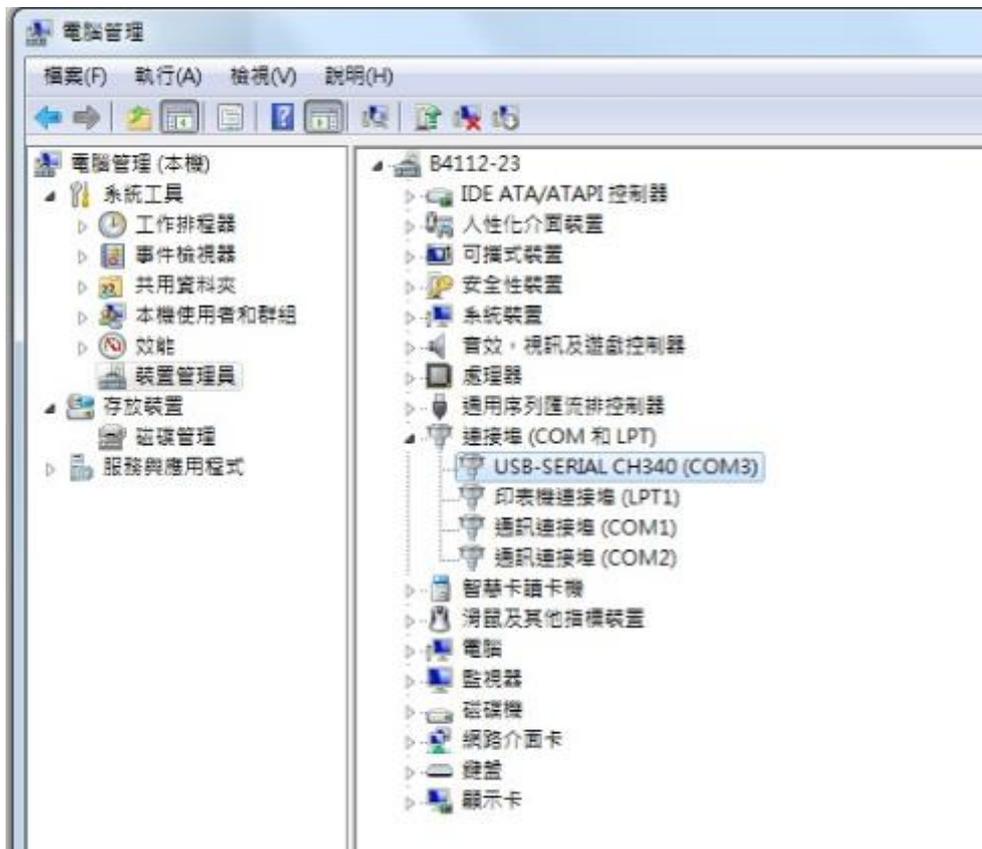


圖 3-17 檢查 WEMOS D1 mini 和電腦連接

如圖 3-18 存在 C 槽 Arduino IDE 1.8.5，原本空的資料夾「Portable」，目前已裝安裝了 ESP8266 的相關開發檔案了。

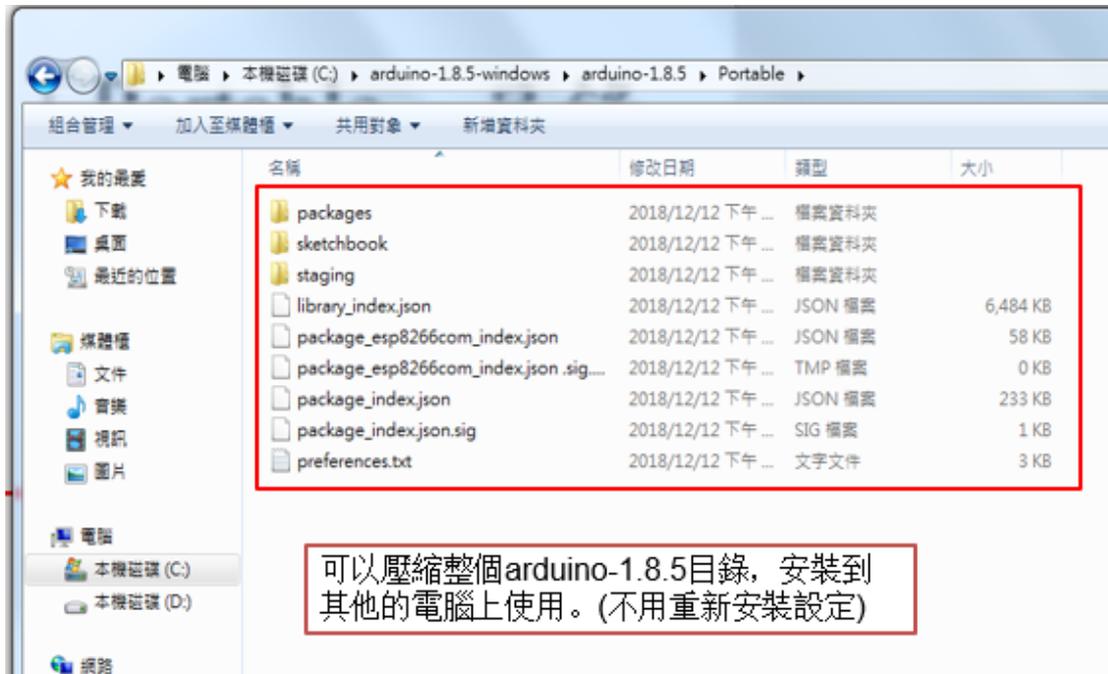


圖 3-18 擴充套件安裝於目錄

## 第4章 雲端伺服器的安裝

### 4.1 Node-Red 安裝與設定

Node-RED 是 IBM 以 Node.js 為基礎，開發出來的視覺化 IOT 開發工具，因為純粹透過流程圖的方式工作，所以不需要會 Node.js 也可以透過 Node-RED 完成許多後端才能做的事情。因為是以 Node.js 為基礎，所以要使用前必須要安裝 Node.js，直接前往 Node.js 官方網站下載安裝。

- 步驟 1：安裝 Node.js，在 Node.js 的網站 <https://nodejs.org/en/>，下載並安裝 Node.js，目前版本為 16.13.1，過程如圖 4-1。

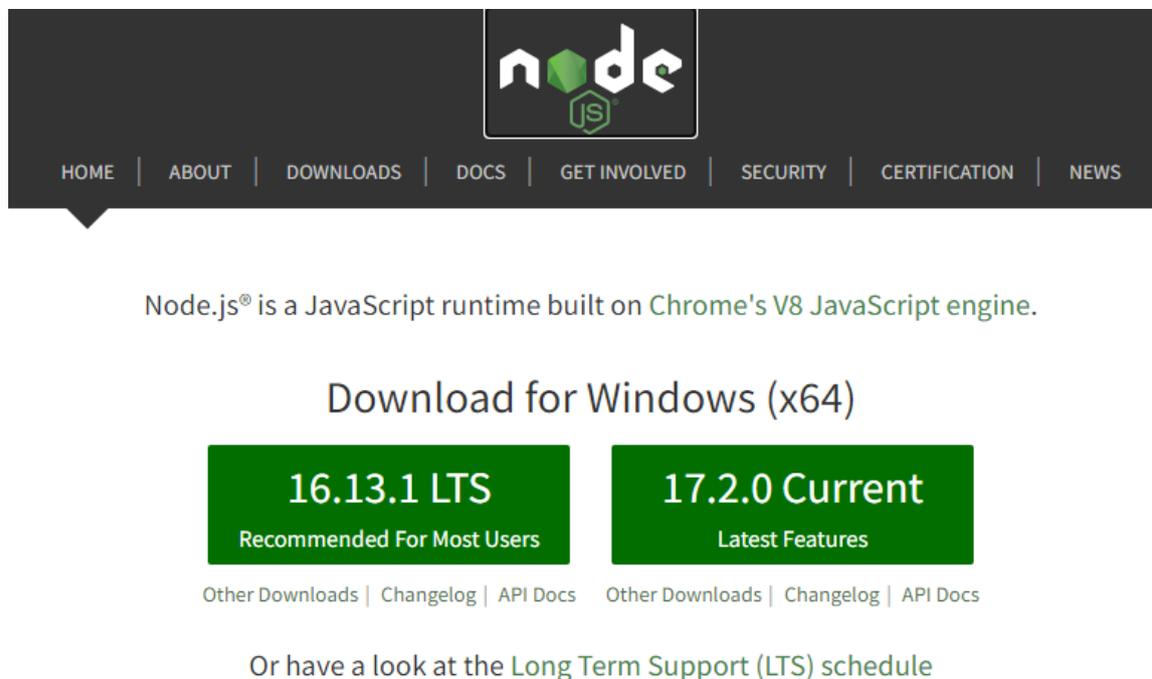
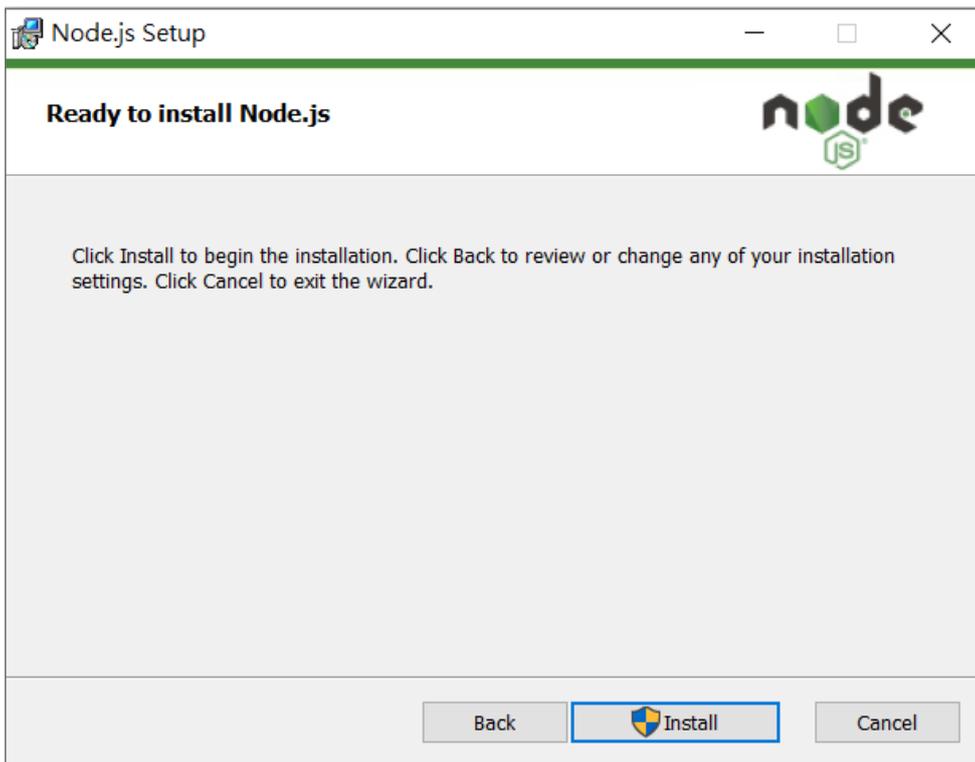
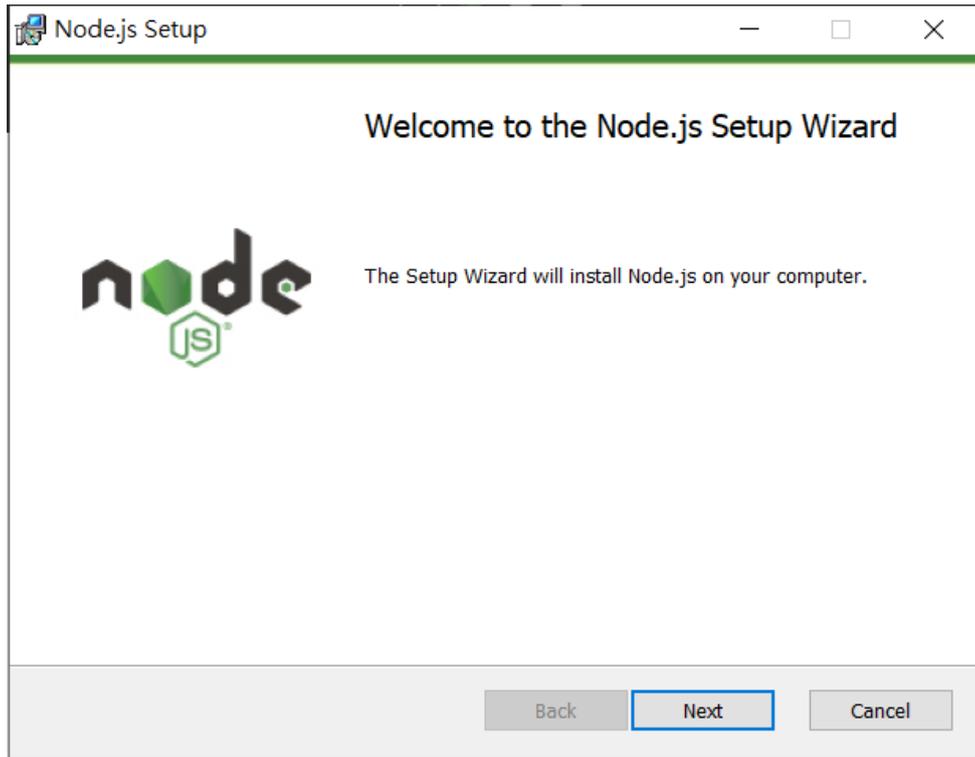
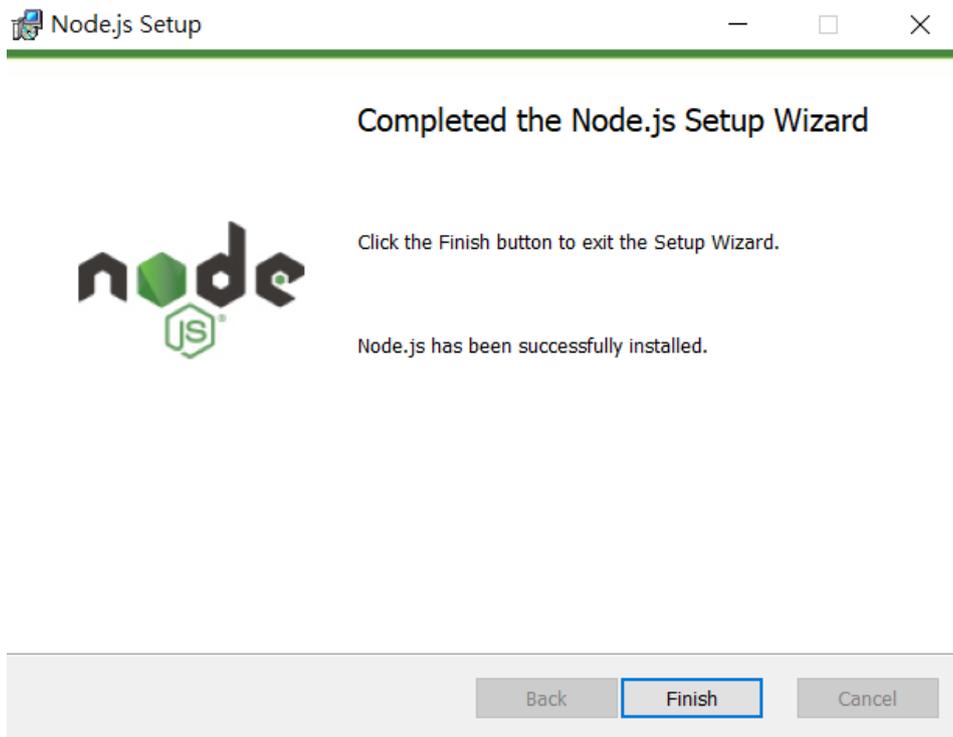


圖 4-1 Node.js 的網站下載並安裝&過程





- 步驟2：安裝Node-Red，以系統管理員進入命令視窗，輸入`npm install -g --unsafe-perm node-red`，以執行Node-Red的安裝，如圖4-2。

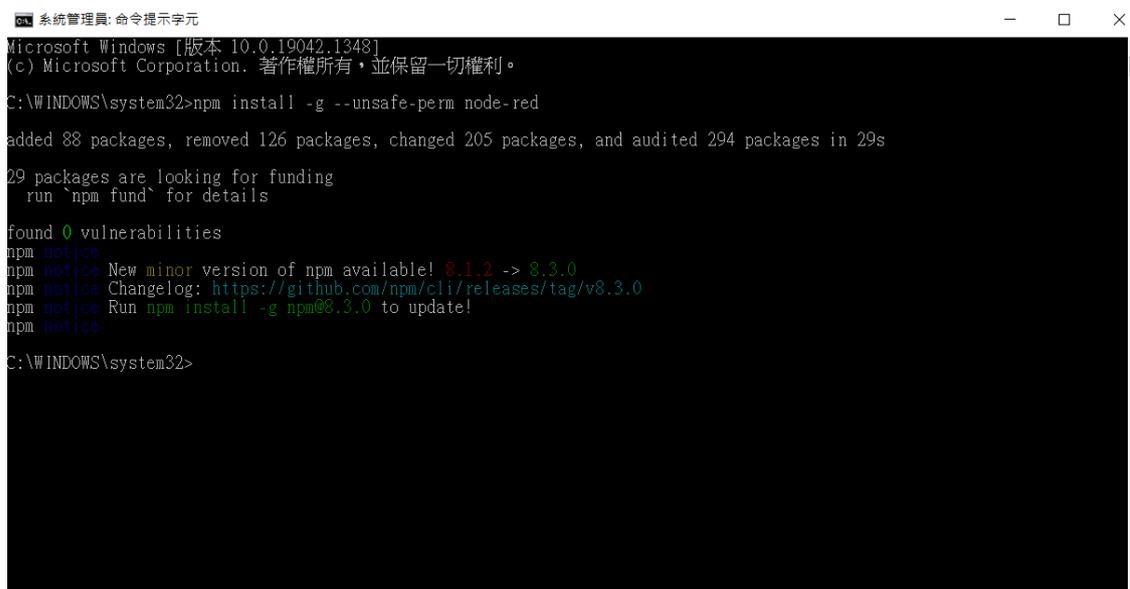
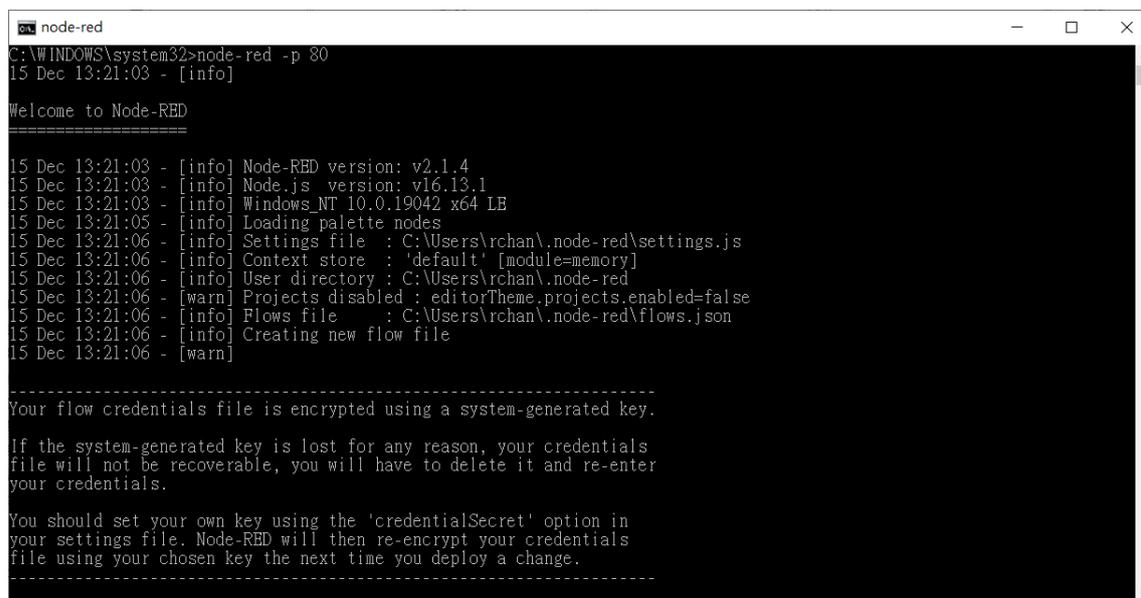


圖 4-2 Node-Red 的安裝

- 步驟 3：啟動 Node-Red，由於學校會阻擋 TCP port 80 以外的通訊

埠，為了在校外也可以存取 Node-Red，將 Node-Red 的通訊埠設為 80，才不會被防火牆阻擋。系統管理員身分進入命令視窗，輸入 `node-red -p 80` 以啟動 Node-Red。啟動 Node-Red 的過程如圖 4-3。



```
node-red
C:\WINDOWS\system32>node-red -p 80
15 Dec 13:21:03 - [info]
Welcome to Node-RED
=====
15 Dec 13:21:03 - [info] Node-RED version: v2.1.4
15 Dec 13:21:03 - [info] Node.js version: v16.13.1
15 Dec 13:21:03 - [info] Windows_NT 10.0.19042 x64 LE
15 Dec 13:21:05 - [info] Loading palette nodes
15 Dec 13:21:06 - [info] Settings file : C:\Users\rchan\.node-red\settings.js
15 Dec 13:21:06 - [info] Context store : 'default' [module=memory]
15 Dec 13:21:06 - [info] User directory : C:\Users\rchan\.node-red
15 Dec 13:21:06 - [warn] Projects disabled : editorTheme.projects.enabled=false
15 Dec 13:21:06 - [info] Flows file : C:\Users\rchan\.node-red\flows.json
15 Dec 13:21:06 - [info] Creating new flow file
15 Dec 13:21:06 - [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.
-----
```

圖 4-3 啟動 Node-Red 的過程

- 步驟 4：進入 NodeRed，啟動瀏覽器(如 Chrome)，網址輸入：  
<http://127.0.0.1:80/>，就可進入 Node-Red 的操作畫面，如圖 4-4。

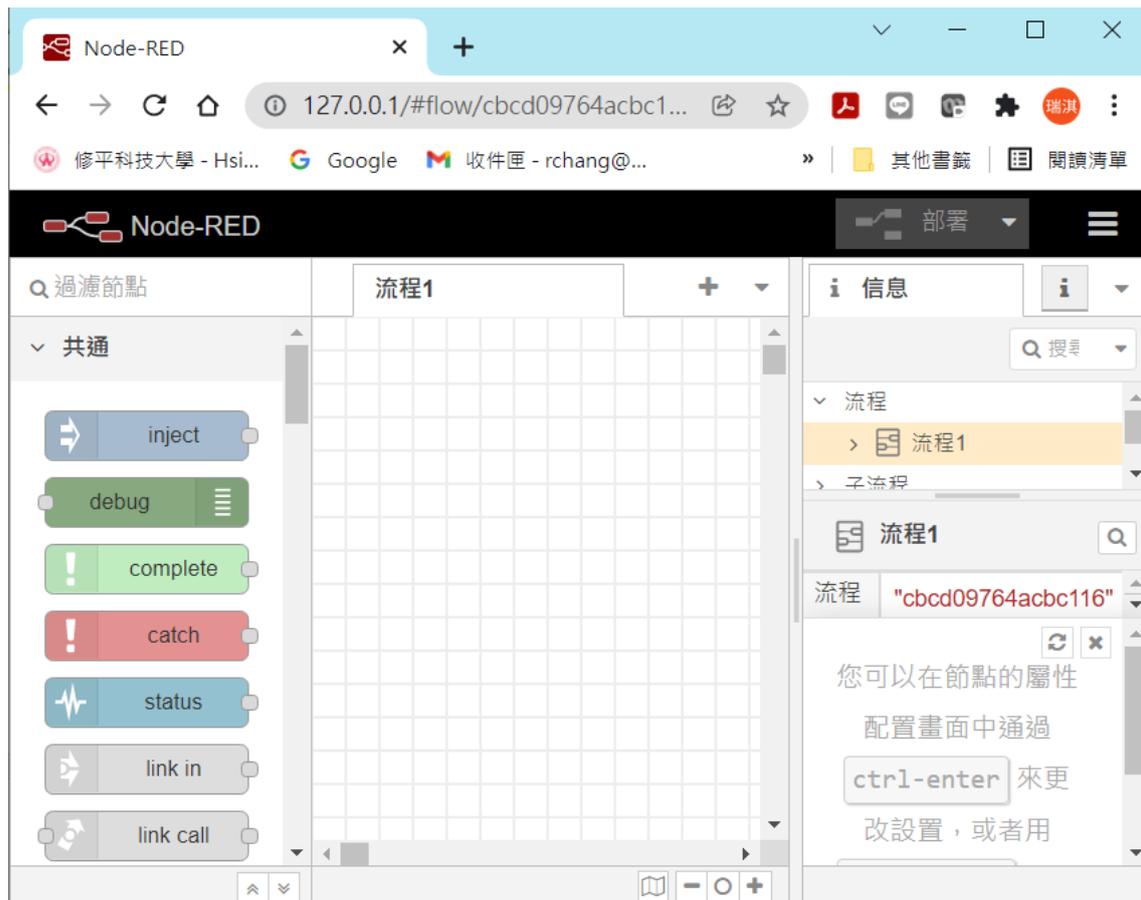


圖 4-4 Node-Red 的操作畫面

- 步驟 5：為了要使 NodeRed 連結至 Mysql 資料庫、MQTT broker 與 LineNotify 服務，要進入 NodeRed 節點管理，新增節點來安裝 node-red-dashboard、node-red-node-mysql、node-red-contrib-mqtt-broker、node-red-contrib-line-notify，已安裝完成的節點如圖 4-5 所示。



圖 4-5 安裝完成的 Node-Red 節點

## 4.2 MQTT Broker 安裝與設定

- 步驟 1：安裝 MQTT Broker (mosquitto)，MQTT 協定是由 IBM 與 Arcom 兩家公司所發展的傳輸協定 Message Queueing Telemetry Transport (訊息佇列遙測傳輸)。MQTT 協定的訊息內容很精簡，非常適合用於處理器資源及網路頻寬有限的物聯網裝置。根據參考文獻[2]下載與安裝 MQTT Broker。

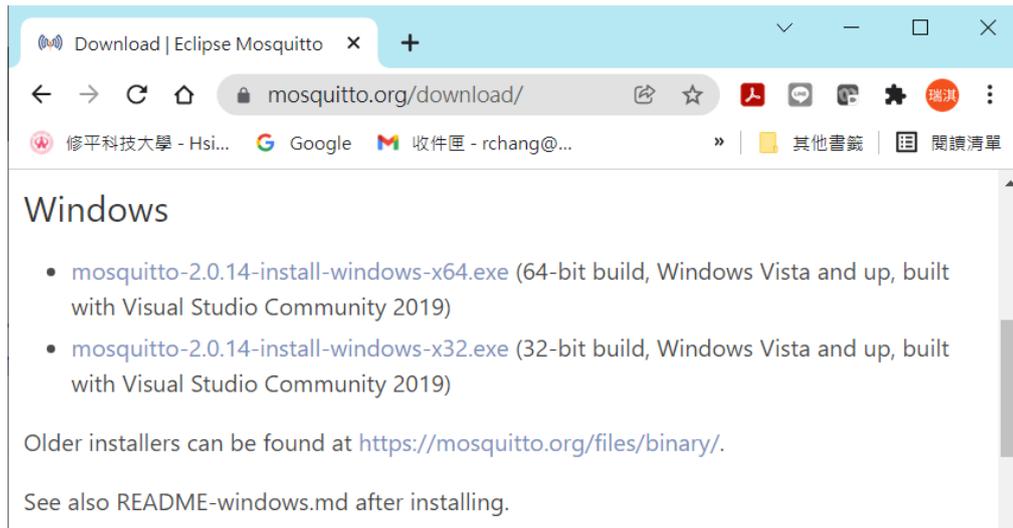


圖 4-6 Mosquito MQTT 下載畫面

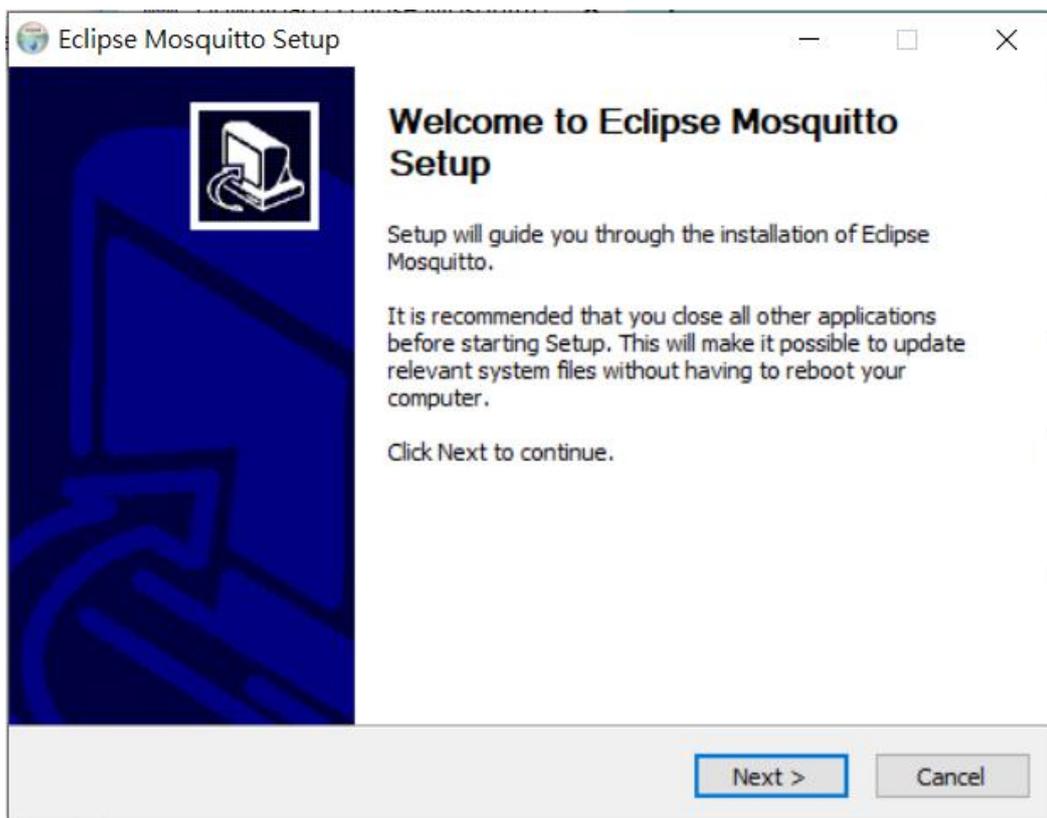


圖 4-7 Mosquito MQTT 開始安裝畫面

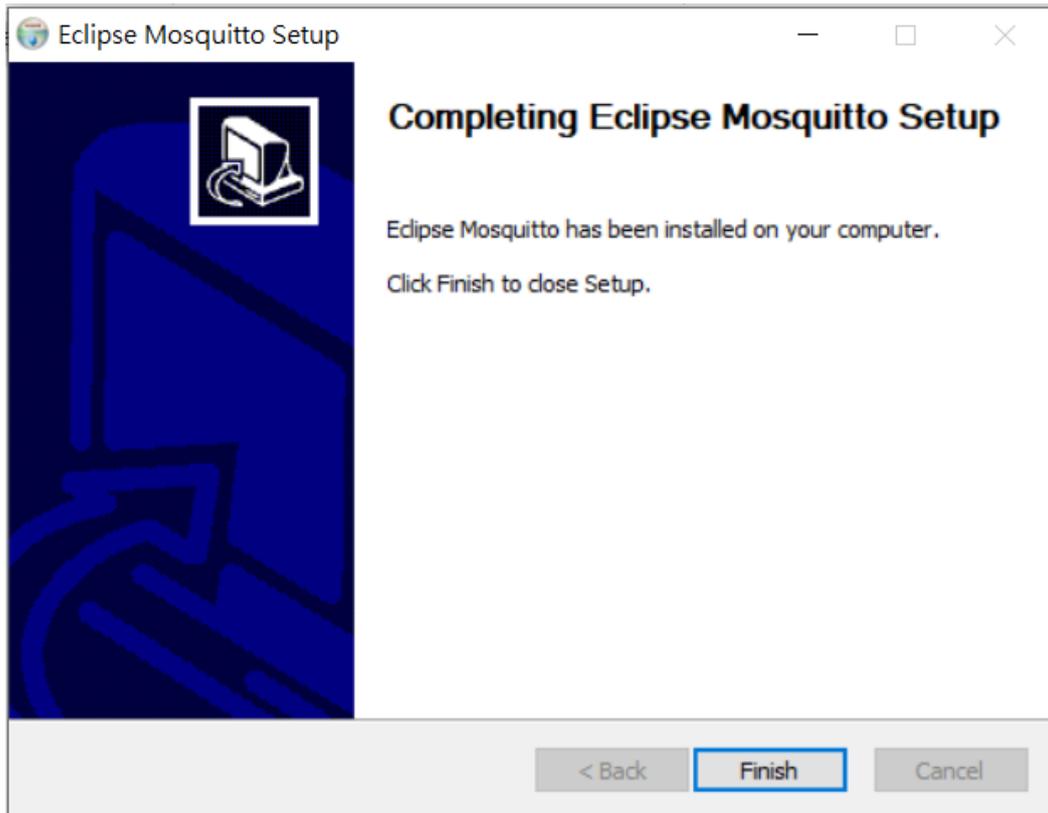


圖 4-8 Mosquito MQTT 完成安裝畫面

- 步驟 2: 由於額溫感測器可能不在學校網路內, 根據[2]的說明步驟, 如圖 4-9, 將 mosquitto MQTT Server 的 Port 改為 8080 必須修改 mosquitto.conf 中的 listener 設定為 listener 8080, 才不會被防火牆阻擋。另外, 為了不要用帳號密碼就可以連線至 mosquitto MQTT Server, mosquitto.conf 中的 allow\_anonymous 要設定為 true, 如圖 4-10。

```

209 #=====
210 #Listeners
211 #=====
212
213 #Listen on a port/ip address combination. By using this variable
214 #multiple times, mosquitto can listen on more than one port. If
215 #this variable is used and neither bind_address nor port given,
216 #then the default listener will not be started.
217 #The port number to listen on must be given. Optionally, an ip
218 #address or host name may be supplied as a second argument. In
219 #this case, mosquitto will attempt to bind the listener to that
220 #address and so restrict access to the associated network and
221 #interface. By default, mosquitto will listen on all interfaces.
222 #Note that for a websockets listener it is not possible to bind to a host
223 #name.
224 #
225 #On systems that support Unix Domain Sockets, it is also possible
226 #to create a # Unix socket rather than opening a TCP socket. In
227 #this case, the port number should be set to 0 and a unix socket
228 #path must be provided, e.g.
229 #listener 0 /tmp/mosquitto.sock
230 #
231 #listener port-number [ip address/host name/unix socket path]
232 listener 8080
233

```

圖 4-9 mosquitto.conf 中的 listener 設為 8080

```

510 #=====
511 #Security
512 #=====
513
514 #If set, only clients that have a matching prefix on their
515 #clientid will be allowed to connect to the broker. By default,
516 #all clients may connect.
517 #For example, setting "secure-" here would mean a client "secure-
518 #client" could connect but another with clientid "mqtt" couldn't.
519 #clientid_prefixes
520
521 # Boolean value that determines whether clients that connect
522 # without providing a username are allowed to connect. If set to
523 # false then a password file should be created (see the
524 # password_file option) to control authenticated client access.
525 #
526 # Defaults to false, unless there are no listeners defined in the configuration
527 # file, in which case it is set to true, but connections are only allowed from
528 # the local machine.
529 allow_anonymous true
530

```

圖 4-10 Mosquito.conf 中的 allow\_anonymous 設為 True

- 步驟 3：如圖 4-11 電腦主機的服務管理員，其中顯示 MQTT Broker 運作正常。

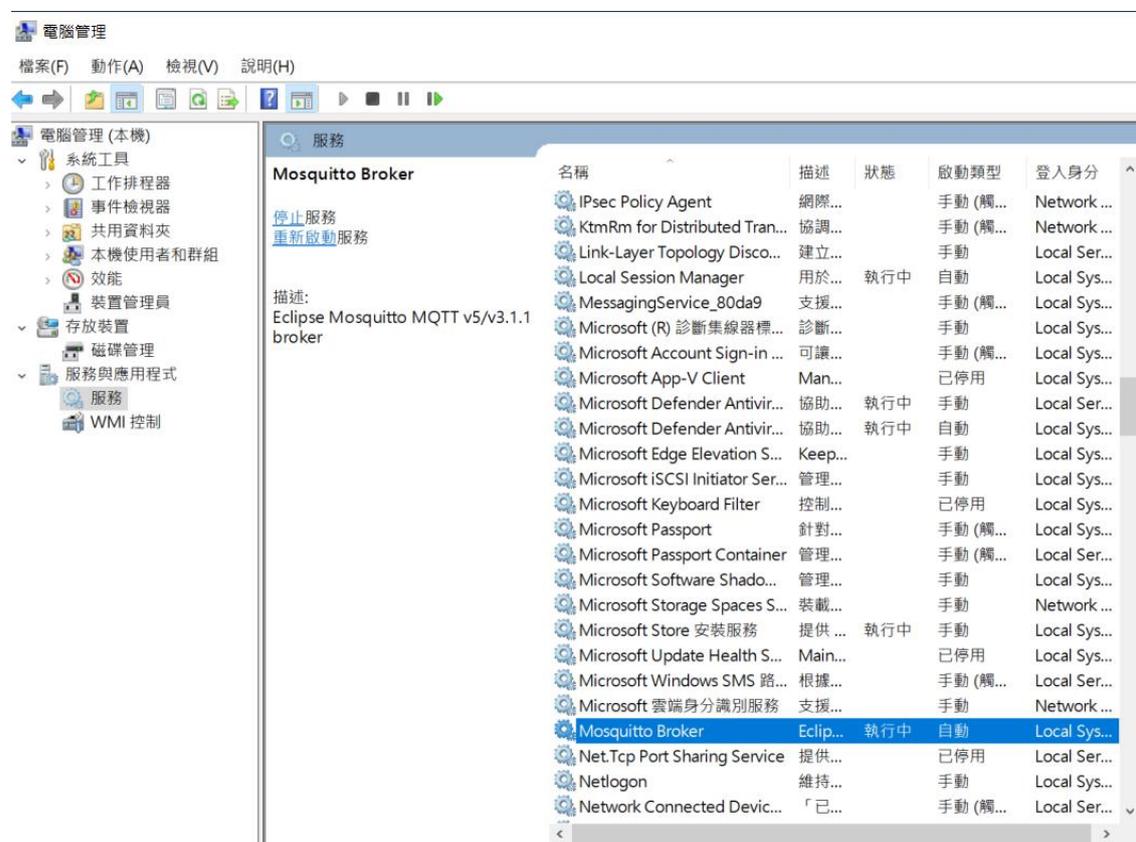


圖 4-11 mosquitto MQTT Broker 的運作狀態

- 步驟 4：將 mosquitto MQTT 設定可以通過 Windows 的防火牆，如圖 4-12。

#### 允許應用程式透過 Windows Defender 防火牆通訊

若要新增、變更或移除允許的應用程式與連接埠，請按一下 [變更設定]。

允許應用程式通訊的風險為何？

變更設定(N)

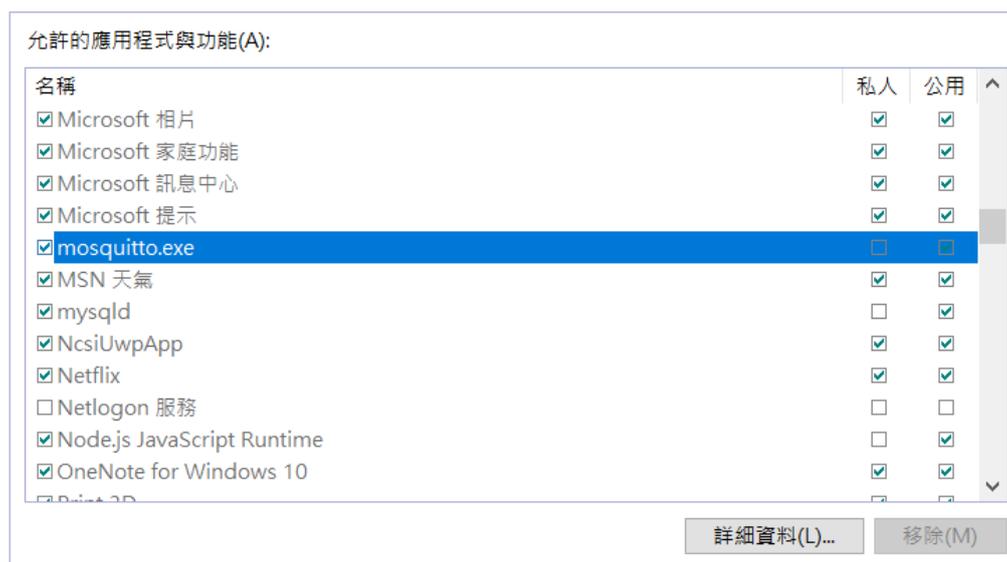


圖 4-12 將 mosquitto MQTT 設定可以通過 Windows 的防火牆

### 4.3 MySQL 安裝與設定

- 步驟 1：MySQL 資料庫安裝，本專題採用的是 XAMPP 套件，利用其中的 Apache 與 MS SQL 伺服器建置 MySQL 資料庫。我們根據參考文獻[1]的說明完成 MySQL 資料庫安裝與設定，XAMPP 下載畫面如 4-13。



圖 4-13 XAMPP 下載畫面

- 步驟 2：MySQL 資料庫安裝，本專題採用的是 XAMPP 套件，利用其中的 Apache 與 MS SQL 伺服器建置 MySQL 資料庫。我們根據參考文獻[1]的說明完成 MySQL 資料庫安裝與設定，開始安裝畫面與結束畫面如圖 4-14、4-15。

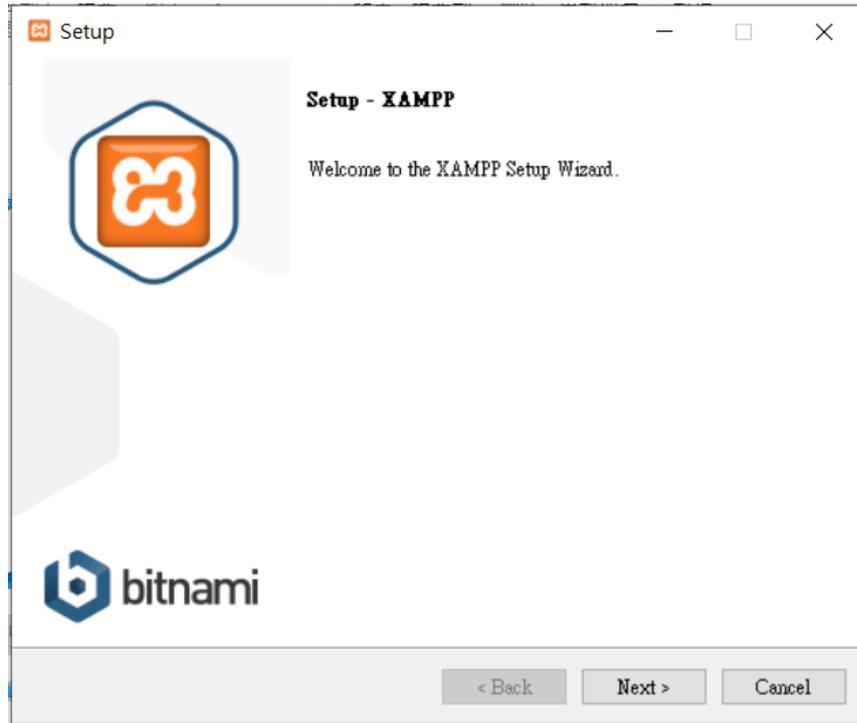


圖 4-14 XAMPP 開始安裝畫面

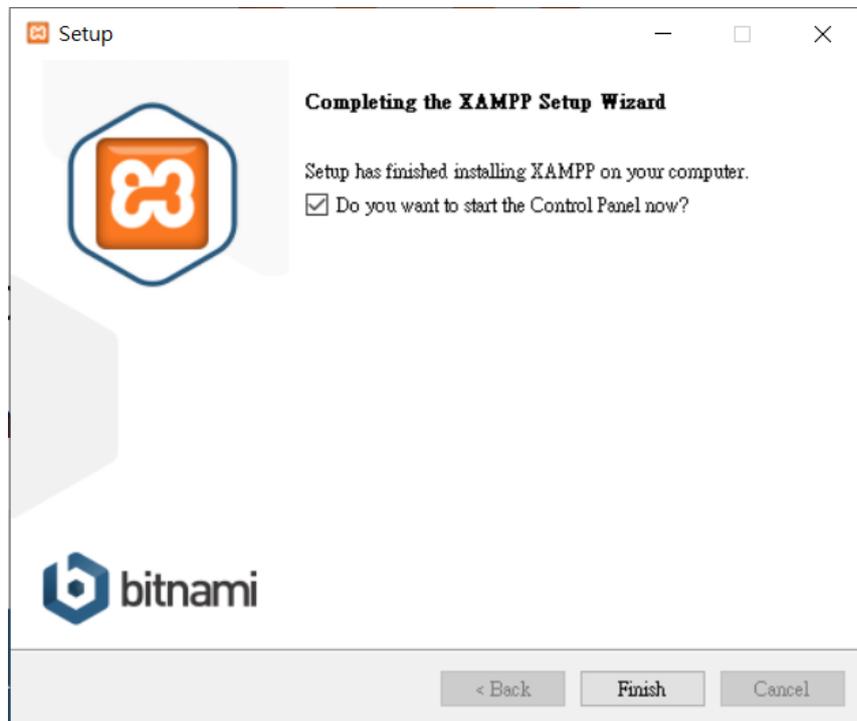


圖 4-15 XAMPP 結束安裝畫面

- 步驟 2: XAMPP 的控制台畫面如圖 4-16, 只啟動我們需要的 Apache 與 MySQL 伺服器。因為 port 80 已經被 NodeRed 使用了, 所以要修改 httpd.conf 中的 Listen 的值为 8081, 如圖 4-17。

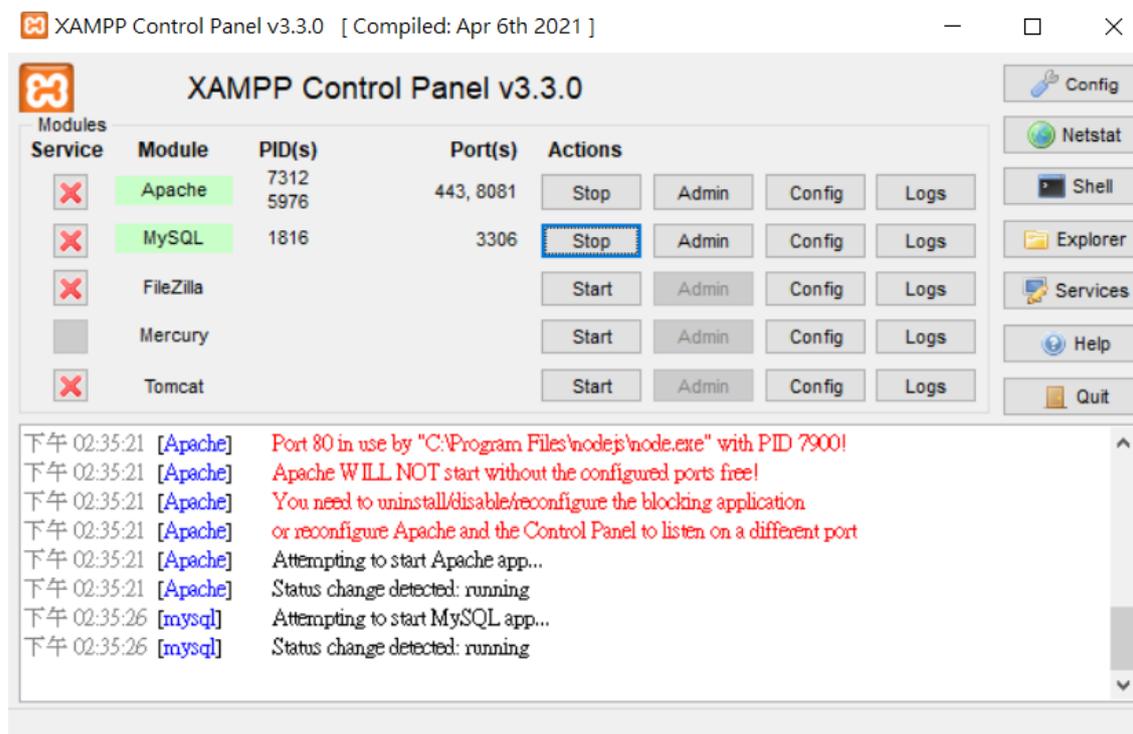


圖4-16 XAMPP的控制台畫面

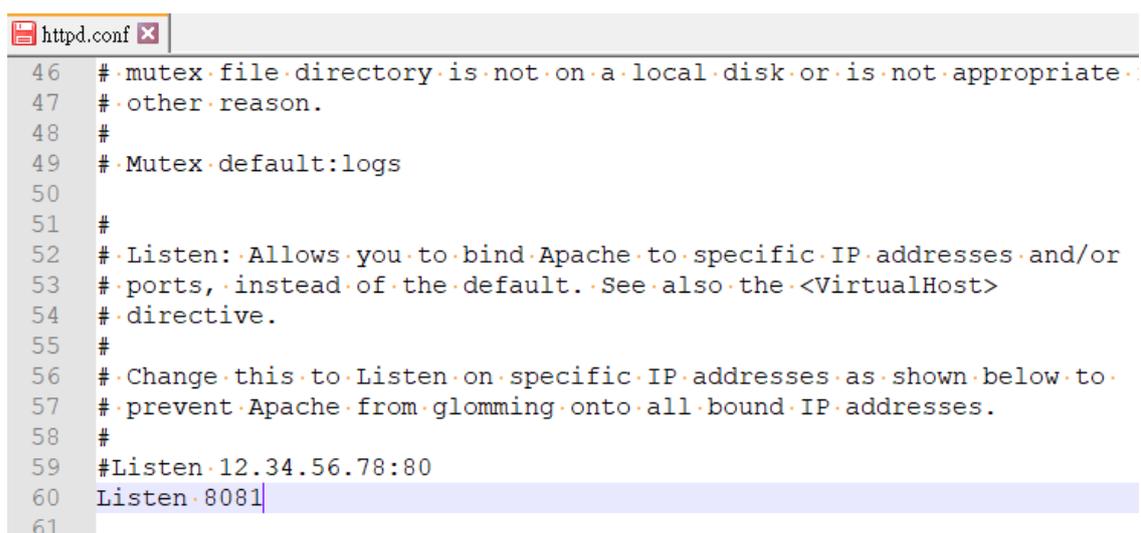


圖 4-17 Httpd.conf 中 Listen 的設定值为 8081

- 步驟 3: 並透過網頁介面，在 MySQL 伺服器中建置資存放額溫資資料的資料庫 stu\_temp 與表格 temp\_data，如圖 4-18。

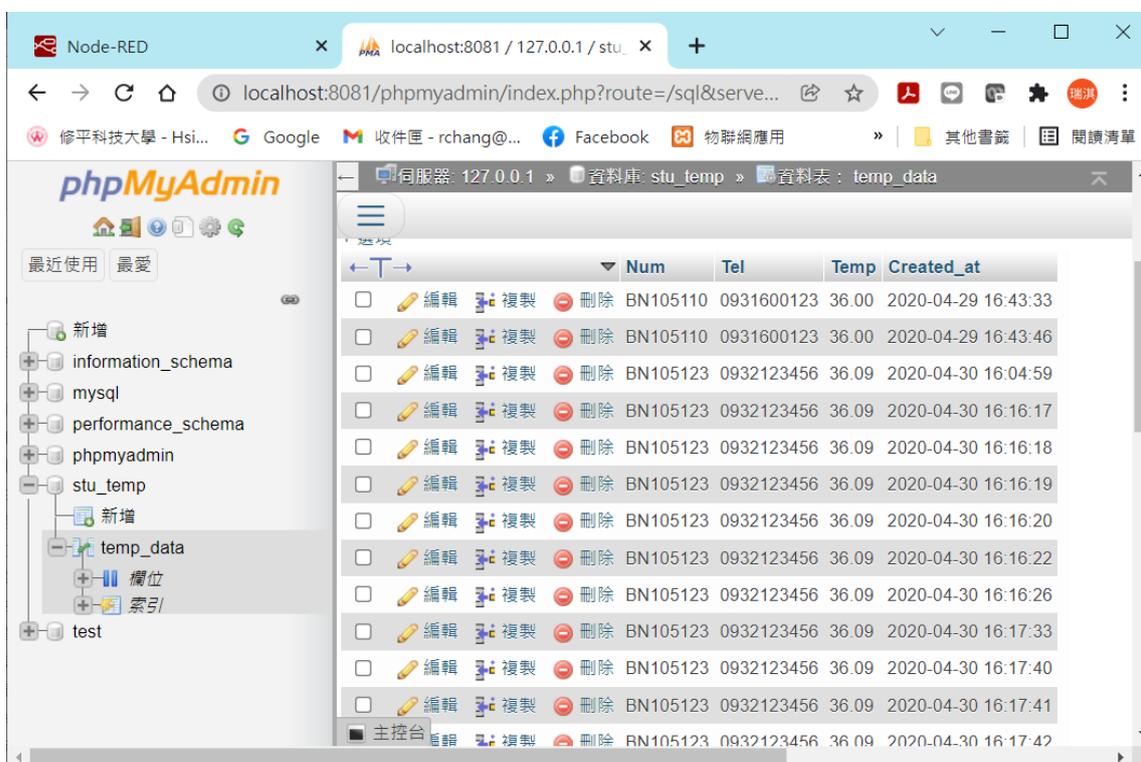


圖 4-18 MySQL 伺服器中建置了溫度的資料庫與表格

## 第5章 額溫雲端量測系統

### 5.1 額溫系統硬體結構

額溫系統的結構如下圖 5-1，所有零件安裝於麵包板上。除了紅外線額溫感測器 MLX90614ESF 與 WEMOS D1 mini 控制板外，並裝有鋰電池與充電、升壓與放電控制器與橘色按鈕開關。透過右手邊的小螢幕顯示量測的額溫，紅外線接近感測器與蜂鳴器也裝在盒內。

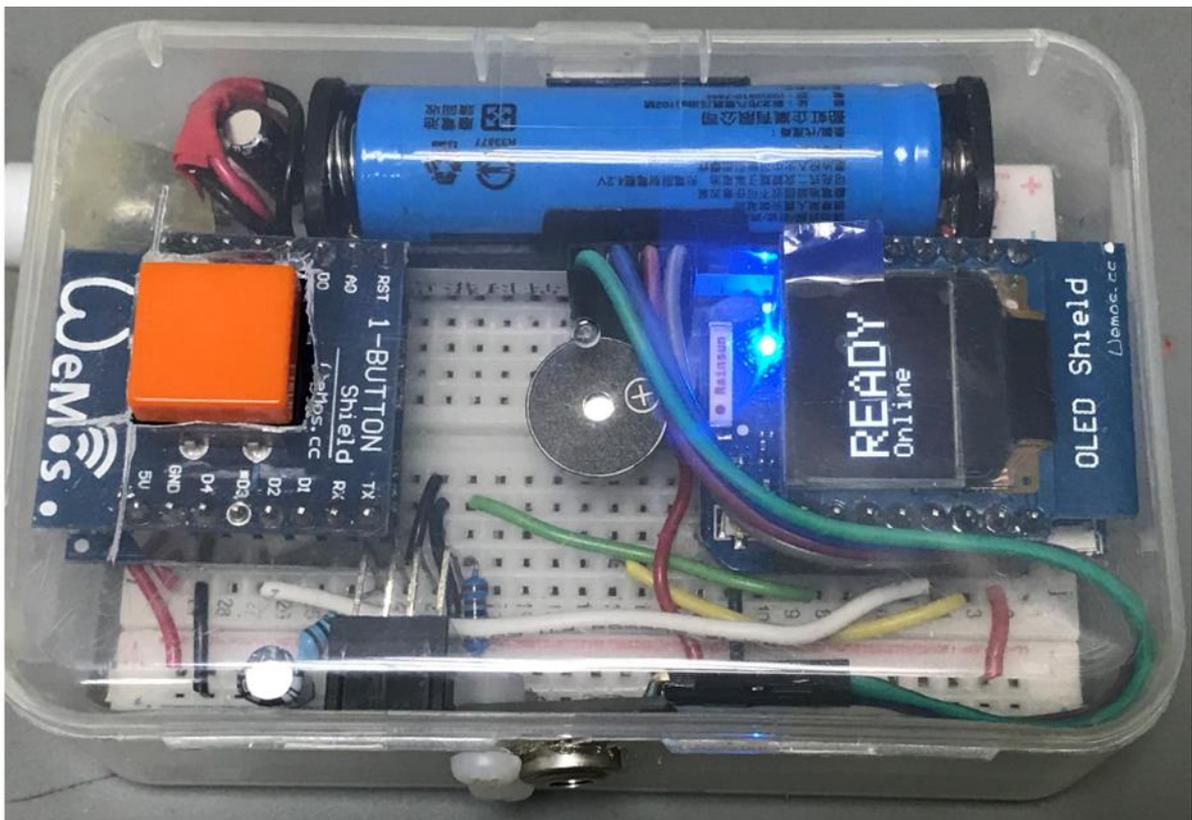


圖 5-1 額溫雲端量測系統

## 5.2 額溫感測器程式

圖 5-2 至圖 5-7 為採用 WEMOS D1 mini 製作額溫感測器的程式 mqtt\_esp8266\_MLX90614.ino 的程式碼。圖 5-2 為函式庫的引用與 WiFi 與 MQTT Broker 參數的宣告，也包括變數、紅外線接近感測器、蜂鳴器與按鈕開關等輸入/輸出接腳宣告。當然也有 OLED 顯示器、額溫感測器等物件的宣告。圖 5-3 為輸入輸出接腳、OLED 顯示器、額溫感測器與 MQTT 物件的初始化副程式，並將 MQTT 物件的回呼副程式定義為 callback()。圖 5-4 為 WiFi 連線的初始化副程式。

### mqtt\_esp8266\_MLX90614.ino

```
1 #include <Wire.h>
2 #include <Adafruit_GFX.h>
3 #include <Adafruit_SSD1306.h>
4 #include <Adafruit_MLX90614.h>
5 #include <ESP8266WiFi.h>
6 #include <PubSubClient.h>
7
8 // Update these with values suitable for your network.
9
10 const char* ssid = "rich";
11 const char* password = "19710507";
12 const char* mqtt_server = "120.109.164.201";
13
14 #define OLED_RESET 0 // GPIO0
15 const int buttonPin = D7;
16 const int buzzerPin = D8;
17 const int irSensorPin = A0;
18 Adafruit_SSD1306 display(OLED_RESET);
19 Adafruit_MLX90614 mlx = Adafruit_MLX90614();
20 const float eps = 0.8;
21 WiFiClient espClient;
22 PubSubClient client(espClient);
23 long lastMsg = 0;
24 char msg[50];
25 int opmode = 0;
```

圖 5-2 額溫感測器程式-1

## mqtt\_esp8266\_MLX90614.ino

```
27 void setup() {
28   ..pinMode(BUILTIN_LED, OUTPUT);.....// Initialize the BUILTIN_LED
29   ..pinMode(buttonPin, INPUT_PULLUP);
30   ..pinMode(buzzerPin, OUTPUT);
31   ..Serial.begin(9600);
32   ..// by default, we'll generate the high voltage from the 3.3v lir
33   ..display.begin(SSD1306_SWITCHCAPVCC, 0x3C);...// initialize with t
34   ..// init done
35   ..display.display();
36   ..delay(2000);
37   ..display.clearDisplay();
38   ..setup_wifi();
39   ..display.clearDisplay();
40   ..display.setTextColor(WHITE);
41   ..display.setTextSize(2);
42   ..display.setCursor(0,0);
43   ..display.println();...display.println("READY");
44   ..display.setTextSize(1);
45   ..if (opmode == 1) {
46     ....client.setServer(mqtt_server, 8080);
47     ....client.setCallback(callback);
48     ....display.println("Online");
49   ..}
50   ..else {
51     ....display.println("Offline");
52   ..}
53   ..display.display();
54   ..mlx.begin();...
55 }
56
```

圖 5-3 額溫感測器程式-2

## mqtt\_esp8266\_MLX90614.ino

```
57 void setup_wifi() {
58   ..delay(10);
59   ..// We start by connecting to a WiFi network
60   ..Serial.println();
61   ..Serial.print("Connecting:");
62   ..Serial.println(ssid);
63   ..display.setTextColor(WHITE);
64   ..display.setTextSize(1);
65   ..display.setCursor(0,0);
66   ..display.println("Connecting"); ..display.println(ssid);
67   ..display.display();
68   ..WiFi.begin(ssid, password);
69
70   ..while (WiFi.status() != WL_CONNECTED) {
71     ....delay(500);
72     ....Serial.print(".");
73     ....display.print(".");
74     ....display.display();
75     ....if (not digitalRead(buttonPin)) {
76       .....break;
77     ....}
78   ..}
79   ..if (WiFi.status() == WL_CONNECTED) {
80     ....opmode = 1;
81     ....Serial.println("");
82     ....Serial.println("WiFi connected");
83     ....Serial.println("IP address:");
84     ....Serial.println(WiFi.localIP());
85   ..}
```

圖 5-4 額溫感測器程式-3

圖 5-5 透過紅外線額溫感測器量測額溫並將訊息發佈至 MQTT Broker 副程式 `measure_publish()`。圖 5-6 的副程式 `callback()` 為接收到 MQTT Broker 請求時的回呼處理副程式，其中會呼叫副程式 `measure_publish()`，將訊息送回 MQTT Broker。由於 MQTT 的連線並不可靠，圖 5-6 的副程式 `reconnect()` 會隨時檢查 MQTT 連線是否中斷，如有中斷，將恢復連線。

## mqtt\_esp8266\_MLX90614.ino

```
88 void measure_publish() {
89     ....digitalWrite(BUILTIN_LED, LOW); ....// Turn the LED on.
90     ....display.clearDisplay(); ....display.setTextSize(1);
91     ....display.setCursor(0,0);
92     ....display.println("Measure");display.println("in");
93     ....display.println("Progress");
94     ....display.display();
95     ....delay(300);
96     ....float at = mlx.readAmbientTempC();
97     ....float ot = mlx.readObjectTempC();
98     ....float Toc = ot + 2.0;
99     ....display.clearDisplay();...
100    ....Serial.print("Ambient = "); Serial.print(at);
101    ....Serial.print("");
102    ....Serial.print("°C\tObject = "); Serial.print(ot);
103    ....Serial.println("°C");
104    ....Serial.print("°C\tCorrected = "); Serial.print(Toc);
105    ....Serial.println("°C");
106    ....Serial.println();
107    ....display.setTextSize(2);
108    ....display.setCursor(0,0);
109    ....display.println(Toc); display.println();
110    ....display.print((char)247); // degree symbol
111    ....display.println("C");
112    ....display.display();
113    ....if (opmode == 1) {
114    ....    ....snprintf(msg, 75, "%2.2f", Toc);
115    ....    ....Serial.print("Publish message: [outTopic] ");
116    ....    ....Serial.println(msg);
117    ....    ....client.publish("outTopic", msg);
118    ....}
119    ....tone(buzzerPin, 300, 1000);
120    ....delay(1000);
121    ....digitalWrite(BUILTIN_LED, HIGH); // Turn the LED off
122    ....}
```

圖 5-5 額溫感測器程式-4

## mqtt\_esp8266\_MLX90614.ino

```
124 void callback(char* topic, byte* payload, unsigned int length) {
125   ..Serial.print("Message arrived [");
126   ..Serial.print(topic);
127   ..Serial.print("]");
128   ..for (int i=0; i<length; i++) {
129     ....Serial.print((char)payload[i]);
130   ..}
131   ..Serial.println();
132
133   ..// Switch on the LED if an 1 was received as first character
134   ..if ((char)payload[0] == '1') {
135     ....measure_publish();
136   ..}
137 }
138
139 void reconnect() {
140   ..// Loop until we're reconnected
141   ..if (opmode == 1) {
142     ..while (!client.connected()) {
143       ....Serial.print("Attempting MQTT connection...");
144       ....// Attempt to connect
145       ....if (client.connect("ESP8266Client")) {
146         .....Serial.println("connected");
147         .....// Once connected, publish an announcement...
148         .....client.publish("outTopic", "0");
149         .....// ...and resubscribe
150         .....client.subscribe("inTopic");
151       .....} else {
152         .....Serial.print("failed, rc=");
153         .....Serial.print(client.state());
154         .....Serial.println(" try again in 5 seconds");
155         .....// Wait 5 seconds before retrying
156         .....delay(5000);
157       .....}
158     ..}
```

圖 5-6 額溫感測器程式-5

圖 5-7 的 loop()副程式是本專案的主程式迴圈，在此迴圈中，不斷檢查 MQTT 物件的連線是否正常，如有中斷會嘗試連線。也會檢查使用者是否按下按鈕開關或接近溫度感測器，如有發生，將會呼叫副程式 measure\_publish()，將量到溫度訊息送回 MQTT Broker。

```

161 void loop() {
162
163   ..if (!client.connected()) {
164     ....reconnect();
165     ....}
166   ..client.loop();
167   ..unsigned long now = millis();
168   ..if (now - lastMsg > 50) {
169     ....lastMsg = now;
170     ....if (not digitalRead(buttonPin) .or. (analogRead(irSensorPin) < 600)) {
171     .....measure_publish();
172     ....}
173   ..}
174 }

```

圖 5-7 額溫感測器程式-6

### 5.3 雲端伺服器 Node-Red 程式 & LineNotify 超溫通知

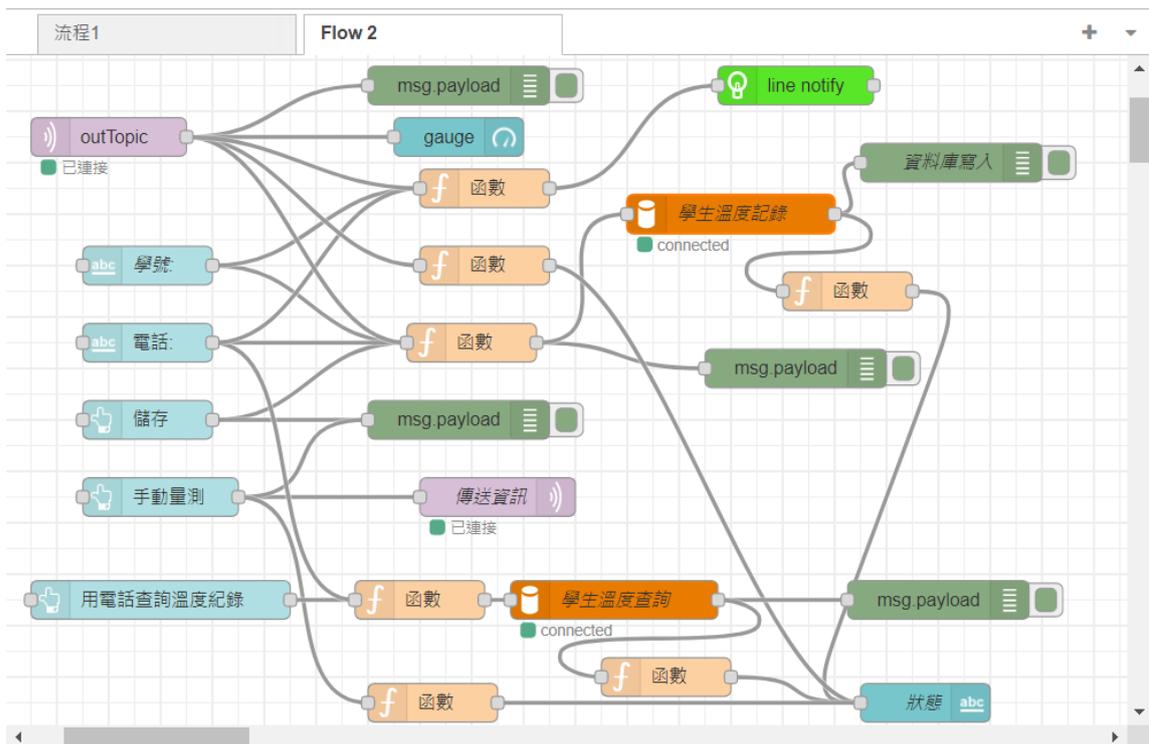


圖 5-8 Node-Red 流程化程式

圖 5-8 是 Node-Red 的流程化程式，此程式提供使用者介面(UI)，讓使用者輸入用戶基本資料如姓名電話，當額頭靠近額溫感測器或按下

按鈕時會，額溫感測器自動量測額溫，並將額溫顯示於使用者介面(UI)。

操作人員可以按下儲存資料按鈕，將資料儲存與 MySQL 資料庫。也可以

按下用電話查詢溫度記錄，將調出此電話的用戶最後一筆溫度資料。

我們在此流程中加入 LineNotify 的功能，當溫度超過 38 度會通知設定的 Line 連絡人，LineNotify 的設定步驟如下。

- 步驟 1：進入 LineNotify 官方網站，如圖 5-9。

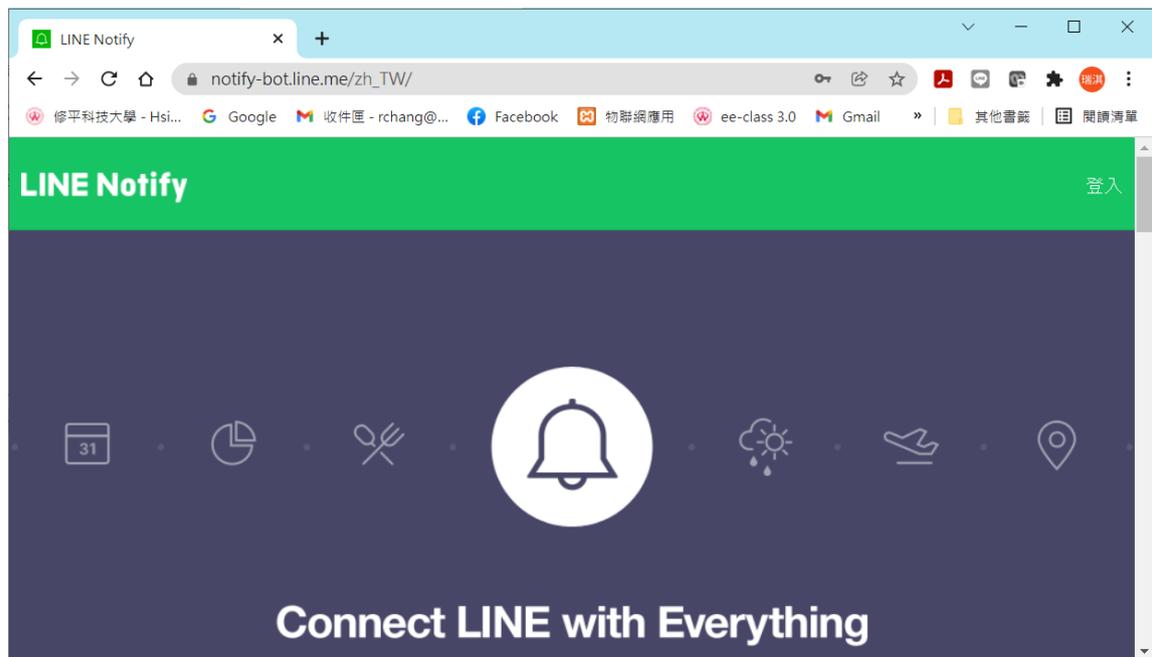


圖 5-9 LineNotify 官方網站

- 步驟 2：連動自己的 Line 帳號，如圖 5-10。

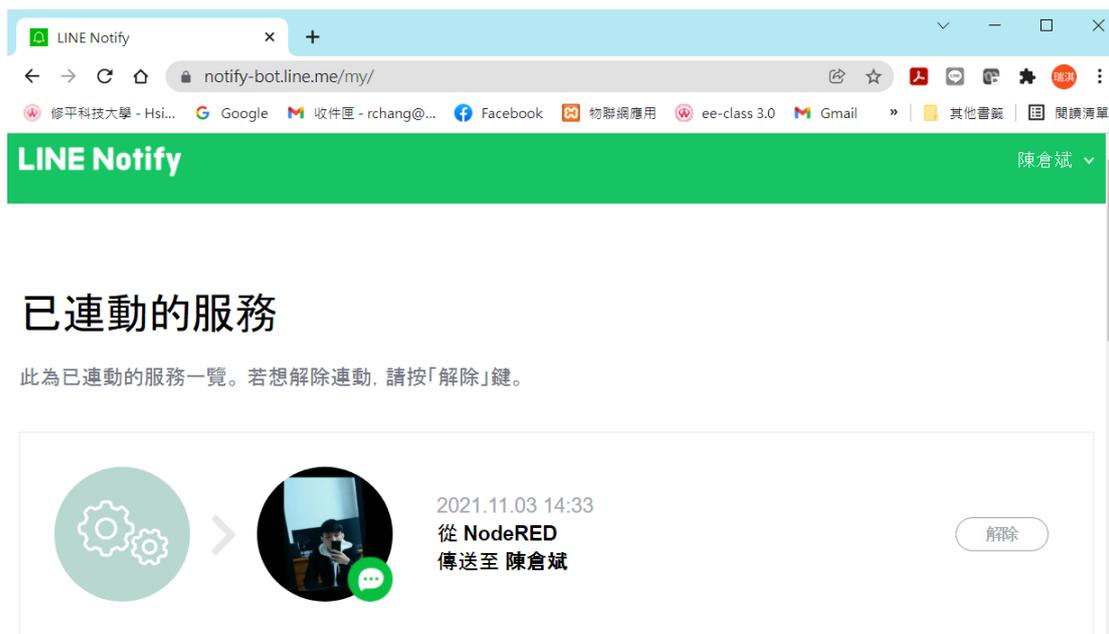


圖 5-10 將 LineNotify 連動至自己的 Line 帳號

- 步驟 3：選取發行權杖，如圖 5-11。



圖 5-11 選取發行權杖

- 步驟 4：選取接收 1 對 1 的 LineNotify 通知，如圖 5-12。

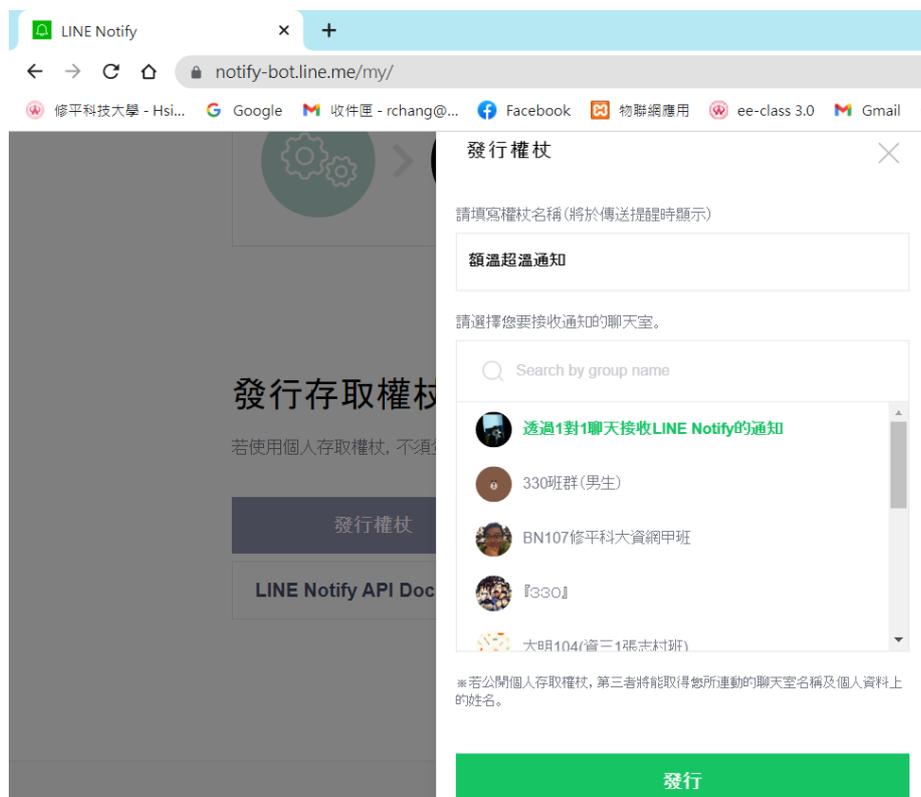


圖 5-12 選取接收 1 對 1 的 LineNotify 通知

- 步驟 5：將得到的權杖碼進行複製，如圖 5-13。

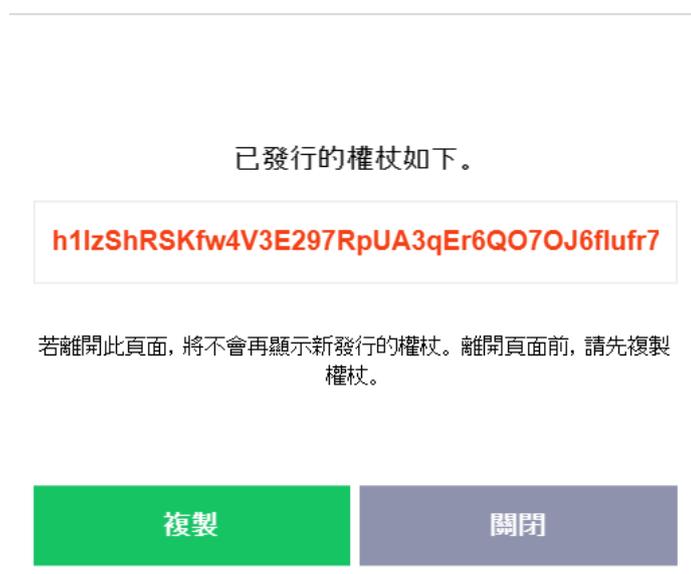


圖 5-13 將得到的權杖碼進行複製

- 步驟 6：在 Node-RED 的 linetoken 節點裡貼上權杖碼，如圖 5-14。

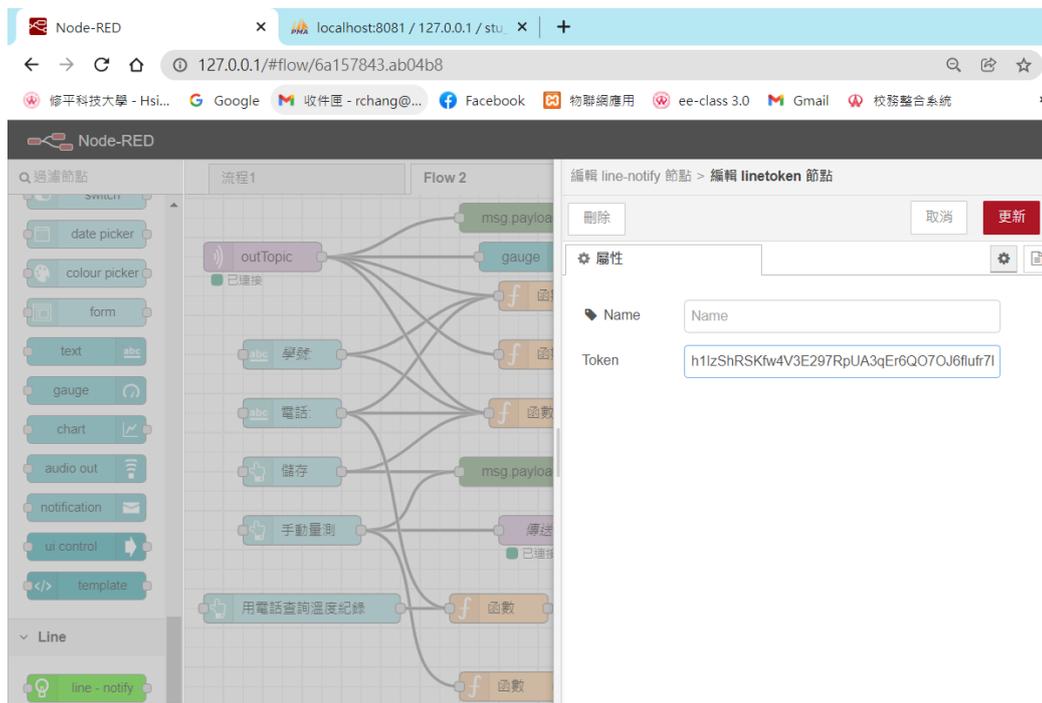


圖 5-14 在 Node-RED 裡 linetoken 節點貼上權杖碼

## 5.4 系統整合測試結果

圖 5-15 至圖 5-18 為系統整合測試結果，圖 5-15(a)為額溫量測前的畫面，溫度為 0 度。圖 5-15(b)為按下輻測按鈕，正在量測額溫的畫面。圖 5-16(a)為額溫量測失敗的畫面，其原因為溫量測器未開機或未連線。圖 5-16(b)為額溫量測成功且溫度正常的畫面。5-17(a)為額溫量測成功，但是溫度在警戒值範圍(37 度至 38 度)。5-18(a)為額溫量測成功，但是溫度超過警戒值範圍(高於 38 度)。5-18(a)與(b)均為成功查詢額紀錄的畫面。

圖 5-19 為收到量測溫度超過一般標準時發出的超溫通知，能夠即時通知管理者前往現場關心同學身體是否有不適的情形，以及後續的處理，來確保在校的每一位同學體溫都是在正常的狀態。



圖 5-15 (a) 額溫量測前；(b)額溫量測中



圖 5-16 (a) 額溫量測失敗；(b)額溫量測成功且正常



圖 5-17(a) 額溫量測在警戒範圍內；(b) 額溫量測出現重大事件



圖 5-18(a) 查詢異常溫度記錄；(b) 查詢正常溫度記錄

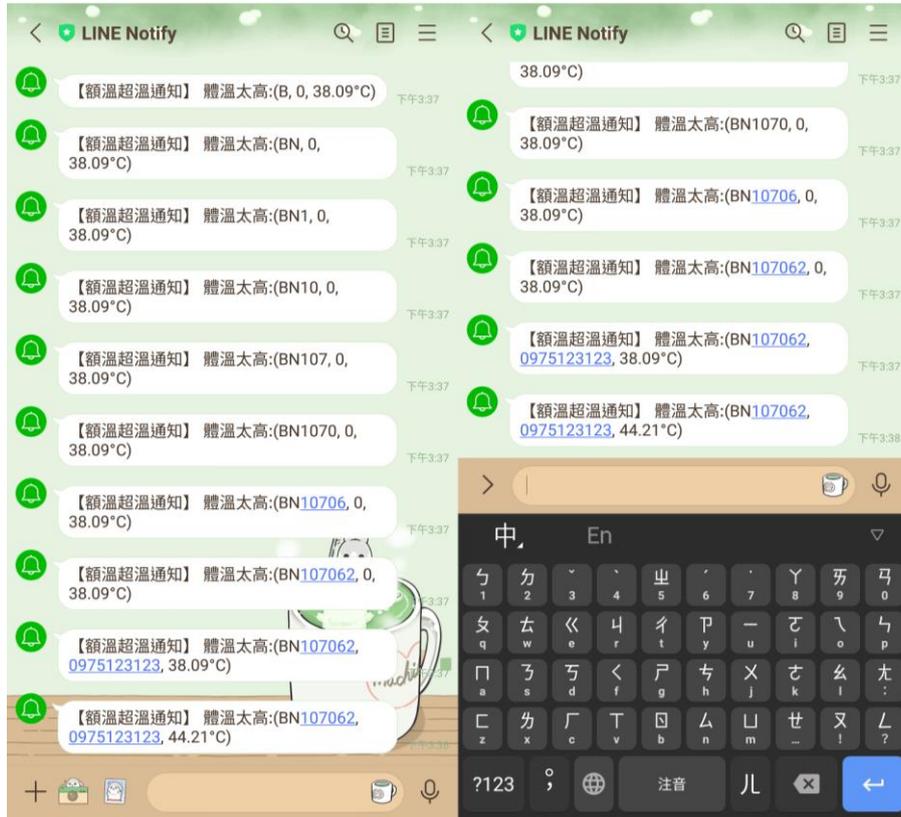


圖 5-19 LineNotify 超溫通知

## 第6章 結論與心得

### 6.1 結論

這次的專題中，一開始要從製作額溫器的外觀著手，需要考慮額溫器的大小與重量，同時空間也是一個問題，最後我們決定 DIY 的方式盡量讓重量降到最低。

接線也遇到很多問題，我們都準備小塊麵包版，製作過程中腳位不大了解，也詢問老師如何接線，以及搜尋 Google，來幫助接線完成。

經由老師指導和網路的摸索，最後我們還是成功幫我們的額溫雲端量測系統完成，再來就是使用 WEMOS D1 miniESP8266 作測試，在測試過程中，有遇到困難的或者手機應用程式不能執行控制，會去詢問老師尋求幫忙，老師會給予幫忙，也讓我們多嘗試很多次，反反覆覆的測試和實驗，也討論了整整一個學期，直到最後終於測試成功後，最後我們也花很多時間整理資料來準備我們的專題報告，覺得辛苦還是有代價的。

## 6.2 心得

- 陳倉斌心得

這次的額溫雲端量測系統，我們在修改程式裡遇到了許多小問題，像是要加入 LineNotify 的通知時，Node-RED 伺服器 linetoken 節點會無法辨識權杖碼，導致訊息無法傳送到手機，或是在修改程式時會遇到的一些小錯誤，經過老師的講解後都能夠順利的解決，也讓我在製作專題的過程中，學習到了許多處理錯誤的方法，加深了解程式和硬體之間的運作方式。

- 張宸維心得

這學期專題是額溫量測系統伺服器安裝，疫情還是有稍微減緩，但是在額溫量測器加上LineNotify可以通知聯絡人有沒有體溫超標，如果超標LineNotify會通知溫度超標訊息通知，利用伺服器來管理讓伺服器程式來呈現在手機LINE的訊息，通知可以讓知道自己身體狀況。

## 參考文獻

- [1] XAMPP 網頁伺服器架站工具設定與使用教學，2020/12/10 存取，  
網址：<https://www.kjnotes.com/devtools/54>
- [2] Mosquitto MQTT Broker，2020/12/10 存取，網址：  
<http://www.steves-internet-guide.com/mosquitto-broker/>
- [3] WeMos D1 WiFi Arduino UNO 開發板 ESP8266 直接用 Arduino  
IDE，2020/12/10 存取，網址：  
<https://sites.google.com/site/wenyumaker/10-esp8266/02memos-d1>
- [4] 楊明豐，Arduino 物聯網最佳入門與應用：打造智慧家庭輕鬆學，  
碁峰資訊股份有限公司，台北，2018 年 3 月
- [5] 趙英傑，超圖解 Arduino 互動設計入門(第三版)，旗標出版股份有  
限公司，台北，2016 年 12 月
- [6] 柯博文，Arduino 互動設計專題與實戰，碁峰資訊股份有限公司，  
台北，2014 年 8 月