教師專業社群

高中優質化輔助方案

108 學年度下學期-自編教材

專題製作

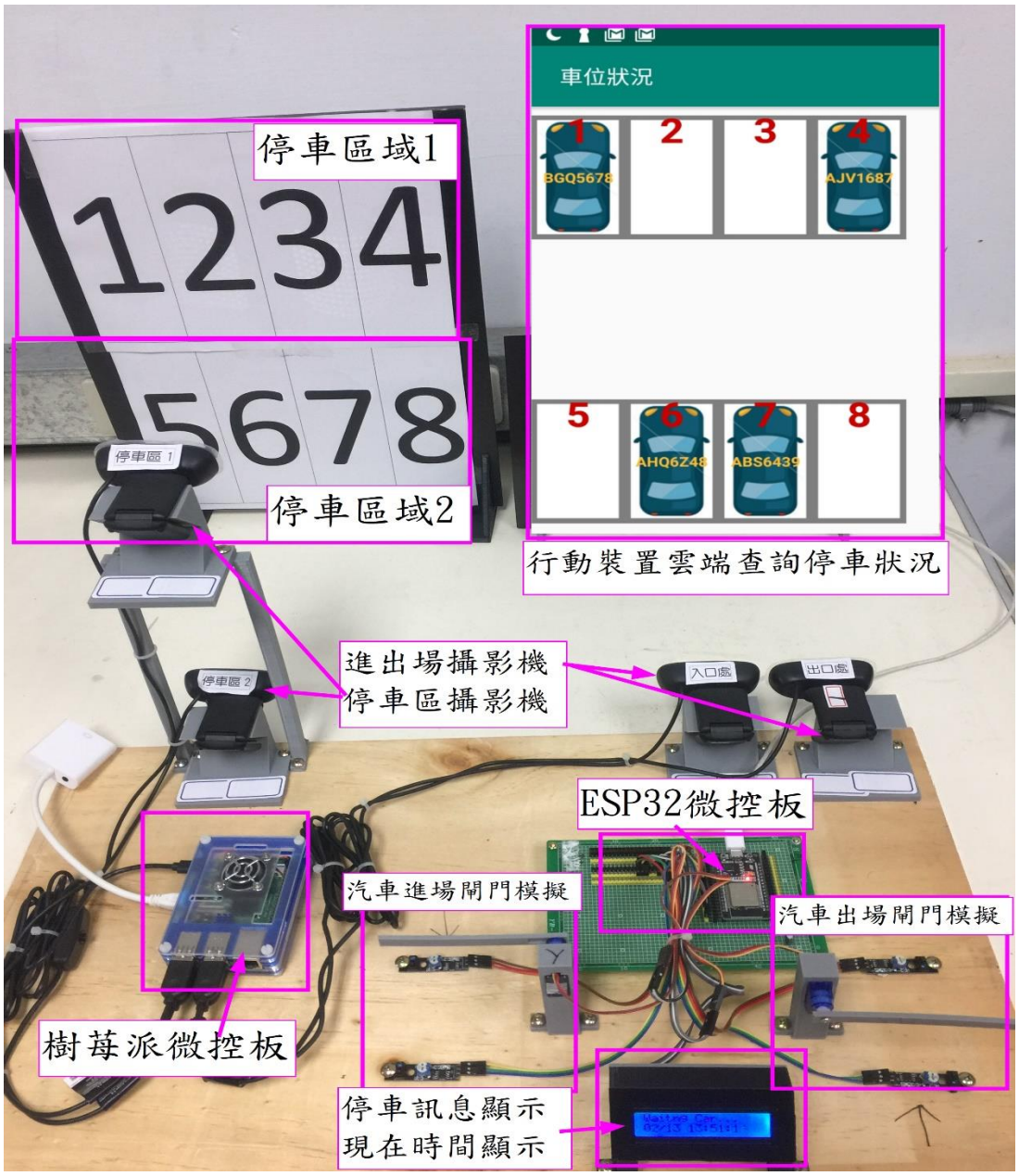
學校名稱：桃園市立楊梅高級中學

日期：中華民國 109 年 6 月

目錄

附件 10-1、全國高級中等學校專業群科 109 年專題及創意製作競賽

「專題組」作品說明書封面



群別：電機與電子群

作品名稱：自動化雲端智慧停車場尋車系統

關鍵詞：MQTT物聯網、手機APP尋車、影像辨識

目　　錄

【自動化雲端智慧停車場尋車系統】

壹、 摘要(300 字以內)

本作品是以物聯網為基礎，整合雲端服務、影像辨識、影像切割、手機 APP 等技術，研發一套自動化雲端智慧停車場尋車系統平台，讓停車場顧客及管理員可以輕鬆掌握停車概況與尋找車位等服務。隨時隨地透過用手機一點就能顯示入場時間、停車位置、停車時數、需繳金額等資訊，即使購物完忘了把您的愛車停在哪，透過手機 APP 就能立即找到您的愛車的位置。另外利用網路攝影機車牌辨識模組監測每個停車位的即時動態，要是遲遲找不到空車位，透過手機 APP 的訊息指引將減少尋找停車位的時間，達到快速停車的效果。另外在車牌辨識搭配影像切割技術，只要一台網路攝影機就能同時辨識定位四格車位，將可節省需架設的網路攝影機的數目，達到節省成本的效果。然而在 APP 的操作介面提供駕駛或是停車場管理者即時的訊息，不管者單一車位資訊或是目前停車場停車狀況一覽無遺，在辦公室裡就能輕鬆掌控停車場概況，進而減少人力成本，提升停車場收益等效果。

貳、 研究動機

在高中課程中有「微電腦應用實習」這門課，其中樹莓派（Raspberry Pi）有介紹關於雲端服務及影像辨識等內容，在對雲端技術與影像處理有了基本的概念後，我們對於其中的原理深感興趣，於是就以這個想法為基礎，來當作這次專題的主題。透過在「行動裝置應用實習」中所學的 Android Studio，整合雲端服務、影像辨識、影像切割、手機 APP 等技術，研發一套自動化雲端智慧停車場尋車系統平台，希望在學校不只能夠學習課本上的知識，而是把知識靈活運用，而且透過實作更能達到學習的目的。

現今市面上的智慧停車場，功能大多大同小異，都是利用單一攝影機照單一車位，有些智慧停車場也會配合室內設備，如定點平板、自動繳費機等....，但有時無法前往定點搜尋裝置，而且又擔心停車需繳金額會超過預算，我們從蒐集來的資料中一直思考，**如何能讓現有智慧停車場降低成本，結合到行動載具上**，是我們製作本作品的研究動機。

	
圖 1 現有智慧停車場內 LED 跑馬燈	圖 2 定點車位查詢裝置

隨著物聯網與影像辨識快速發展，越來越多便利的智慧設備問世，如高速公路 ETC 收費系統、智能交通等，有了這些系統，人們只要利用手機或網際網路透過物聯網就可以輕易取得資訊，我們對於其中的原理深感興趣，於是就以這個想法為基礎，來當作這次專題的主題。

- 以低成本材料達成**雲端尋車智慧**停車場功能
- 使用**影像切割技術減少需架設**之網路攝影機
- 除了現有停車場功能外，增加使用**手機 APP 雲端查詢車位**之功能

參、 主題與課程之相關性或教學單元之說明

一、 主題與課程相關性介紹

本作品搭配高二及高三課程中的「物聯網實習」、「微電腦應用實習」、「單晶片微處理機實習」、「行動裝置應用實習」、「程式設計實習」所學，擬整合所學過的核心專業科目設計模擬一「自動化雲端智慧停車場尋車系統」。

二、 主題與對應課程教學單元之說明如下表所示

表 1 主題與對應課程教學單元對照表

課程單元	作品內容對應
1. 微電腦應用技能領域： (1)微電腦應用實習 B.微電腦應用實習平台 D.應用軟體開發平台安裝 G.微電腦進階應用 (2)行動裝置應用實習 D.使用者介面設計 F.進階介面元件 K.綜合應用	(1) 本作品使用樹莓派（Raspberry Pi）開發板對應「微電腦應用實習」課程，使用樹莓派進行影像擷取、影像分割與辨識的處理及架設MQTT伺服器。 (2) 本作品特色為透過行動裝置APP與雲端，可即時查詢停車場的相關訊息及個人的停車資訊（停車位置、進場車子影像、停車所在位置影像、停車停留時間及目前需繳交的停車費），對應「行動裝置應用實習」的課程。
2. 晶片設計技能領域： (1)單晶片微處理機實習 C.單晶片微處理機開發流程 D.程式撰寫 F.進階應用控制 (2)程式設計實習 B.程式架構的認識與實作	(1) 為了讓整個作品生動有趣，使用ESP32微控板模擬停車系統出入口升降閘門、顯示系統時間及現有車位等相關訊息的顯示。對應「單晶片微處理機實習」的課程。 (2) 使用樹莓派（Raspberry Pi）開發板或是ESP32微控板，除了要瞭解開發板硬體的架構與GPIO的接腳功能外，如何在開發板上撰寫程式控制開發板為其首要之務，對應「程式設計實習」的課程。

肆、 研究方法

一、 設備及器材

下表 2 與表 3 是本專題研究所使用的研究材料與設備的彙整表

表 2 研究材料表

材料名稱	規格	數量	功能
樹莓派開發板	Raspberry Pi 4	1 片	車牌辨識、MQTT
單晶片開發板	ESP32	1 片	MQTT、模擬閘門動作、訊息顯示
萬用 PC 電路板	10cmx5cm	1 片	固定 EPS32 開發板用
焊錫	0.6mm	少許	焊接電路
OK 線	30AWG	少許	焊接電路
伺服馬達	SG90	2 個	模擬出入口閘門用
LCD 液晶顯示器	16x2	1 個	車輛入口處訊息顯示
網路攝影機	Logitech C270	4 個	出入口及停車區辨識用
出入口閘門模型	3D 自己建模設計列印	2 組	固定伺服馬達用
入口處訊息顯示	3D 自己建模設計列印	1 個	固定 LCD 液晶顯示器用
網路攝影機支架模型	3D 自己建模設計列印	4 個	固定網路攝影用
停車場位置支撐架模型	3D 自己建模設計列印	3 個	固定樣本車牌用
木製地板模型		1 片	固定上述相關組件及開發板用

表 3 研究設備表

研究設備	規格	數量
桌上型電腦	Intel Core i7	1
作業系統	樹莓派 linux　NOOBS 記憶卡	1
作業系統	Windows 10	1
Arduino 程式撰寫	ESP32 開發板	1
APP 開發軟體	android studio	1
遠端桌面軟體	Teamviewer 或是 anydesk	2
樹莓派 javascript	Node.js 系統	1

程式撰寫		
行動載具	手機或平板	1
3D 列印機	使用 TinkerCAD 系統平台開發	1
無線 AP	具 DHCP 之無線區域網路	1

二、 研究步驟

本小組經過充分討論後將研究步驟以模組方式予以區分，並依照下列步驟進行研究分析，如圖 3 之步驟進行方塊圖所示

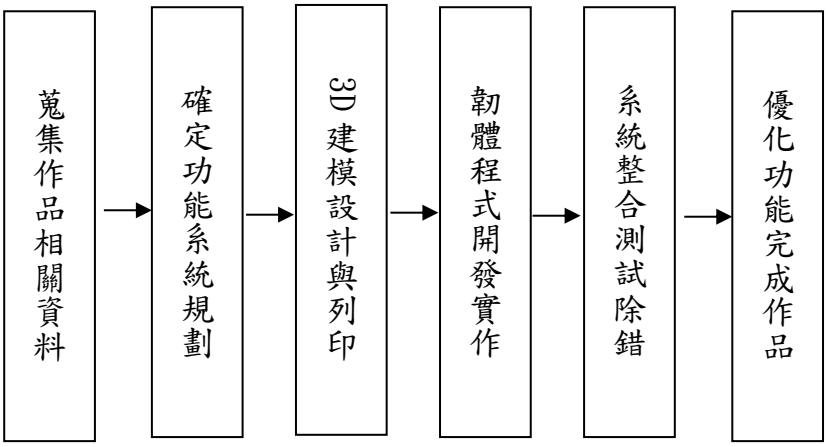


圖 3 研究步驟流程方塊圖

其中各步驟流程所要完成得事項如下：

（一）、收集研究主題文件資料並採購相關零組件。
（二）、規劃各開發板間的功能及硬體周邊控制電路。
（三）、樹莓派與 ESP32 微控板韌體程式的開發。
（四）、行動載具 APP 程式的開發。
（五）、停車場模型相關零組件支架 3D 建模設計與列印。
（六）、樹莓派與 ESP32 周邊硬體控制程式整合測試。
（七）、系統整合測試除錯，並優化相關系統功能。。

三、 系統架構

下圖 4 為本作品的系統架構圖，本系統使用兩種開發板分別是樹莓派（Raspberry Pi）與 ESP32 低功耗的單晶片微控制器，兩種開發板皆有連網都能力。其中樹莓派（Raspberry Pi）開發板主要負責車牌影像辨識、停車位置定位辨識及提供 MQTT Server 伺服服務。ESP32 開發板負責模擬停車系統升降閘門、系統時間、現有車位等相關訊息的顯示。透過行動裝置 APP 至 MQTT Server 查詢提供即時的停車訊息，讓停車取車更加便利。

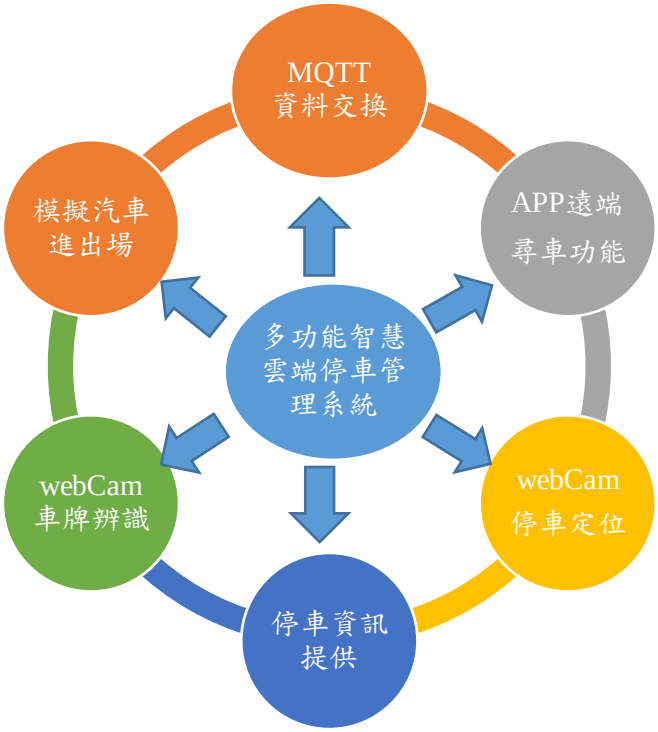


圖 4 自動化雲端智慧停車場尋車系統方塊圖

四、 研究進度甘特圖

表 4 研究進度甘特圖

時間 / 工作項目	10 月		11 月		12 月		1 月		2 月
主題確定									
蒐集資料									
電路製作 3D 列印									
韌體程式設計									
整合測試除錯									
撰寫報告與優化									

伍、 研究過程

由於本作品搭配 webcam 進行影像車牌辨識及停車定位，此部分功能需即時辨識，所以選擇樹莓派第四代（Raspberry Pi 4）開發板，其特點為運算速度快、影像處理能力強、使用 Linux 作業系統及開源程式庫資源完整。除此之外，樹莓派開發板也擔任 MQTT Server 服務，處理雲端資料的訂閱與發佈，雲端伺服器將記錄每一輛車的車號、停車位置、進場時間、停車時數、需繳金額等相關資訊。而 ESP32 開發板主要模擬停車系統升降閘門、系統時間、現有車位等相關訊息的顯示，因網路資源豐富，開發過程比較簡單且節省時間。

結合上述的各項技術模擬實際狀況使實驗過程生動有趣，進場處的 webcam 自動進行車牌辨識，將車號、進場時間透過樹莓派即時上傳到雲端（MQTT Server），有別於目前的停車場管理，透過 APP 在遠端就可查詢愛車的資訊，利用網路攝影機一對多個車位進行拍攝，切割的畫面定址達到可查詢車位位置的目的，用戶端使用手機 App 從 MQTT Server 訂閱主題（MQTT Topic），輸入車號可顯示此車的停車位置、進場時間、目前停車時數、需繳金額。

一、 目前停車場現況探討分析

探訪附近的室內停車場目前的停車功能接改成自動化的管理系統，車輛進場與出場採用手動磁扣感應或車牌自動辨識，繳費方式可透過悠遊卡或是繳費機繳費。仔細探究各停車場除少數比較新建置的停車場會在繳費處提供尋車服務如圖 5 及圖 6 所示所示，然大部分的停車場仍未提供此服務，在取車上比較不便。鑑於行動裝置的普及與網路的發達，發想把目前停車場內的相關**訊息透過雲端可在行動裝置隨時可進行查詢**，以增加使用者的便利性。

圖 5 尋車系統示例 1

圖 6 尋車系統示例 2

目前的尋車系統所提供的停車影像是透過現場停車位上方的攝影機所拍攝，仔細觀察每一停車位上方都安裝一個小型攝影機與車位是否被停放的指示燈，如圖 7 及圖 8 所示。考量每一停車位皆要安裝一部攝影機拍照使用者的停車位成本較高，**想要透過影像處理（分割）的方式降低攝影機的安裝個數以降低成本。**

圖 7 停車場攝影機安裝處示例 1

圖 8 停車場攝影機安裝處示例 2

二、 解決方案說明

（一） 利用影像辨識、影像切割的技術進行停車位置定位，目前實驗以一台攝影機拍攝四部車的範圍為例，如圖 9 所示。

（二） 可以在遠端透過手機或平板 APP 程式，可以查詢愛車停放位置、進場時間、所需繳交的停車費用、目前停車場的空位等訊息，如圖 10 所示。

圖 9 透過影像辨識分割定位

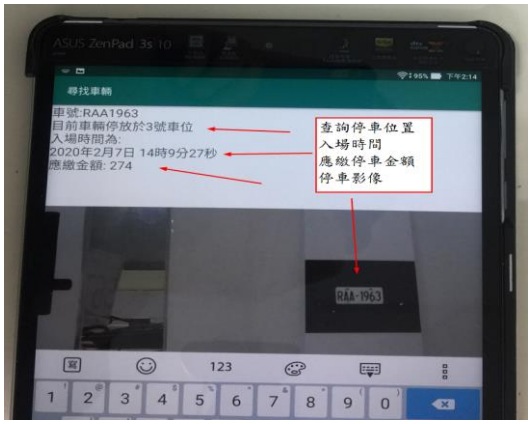


圖 10 APP 查詢愛車所的資訊

三、 軟硬體系統架構圖介紹

呼應研究方法的的系統架構，以現有的網路環境為平台為基礎，挑選所需的硬體與軟體進行規劃，其規劃出的硬體系統架構圖如下圖所示，接著將逐一介紹建置本系統的硬體、軟體、相關理論及設計理念想法。

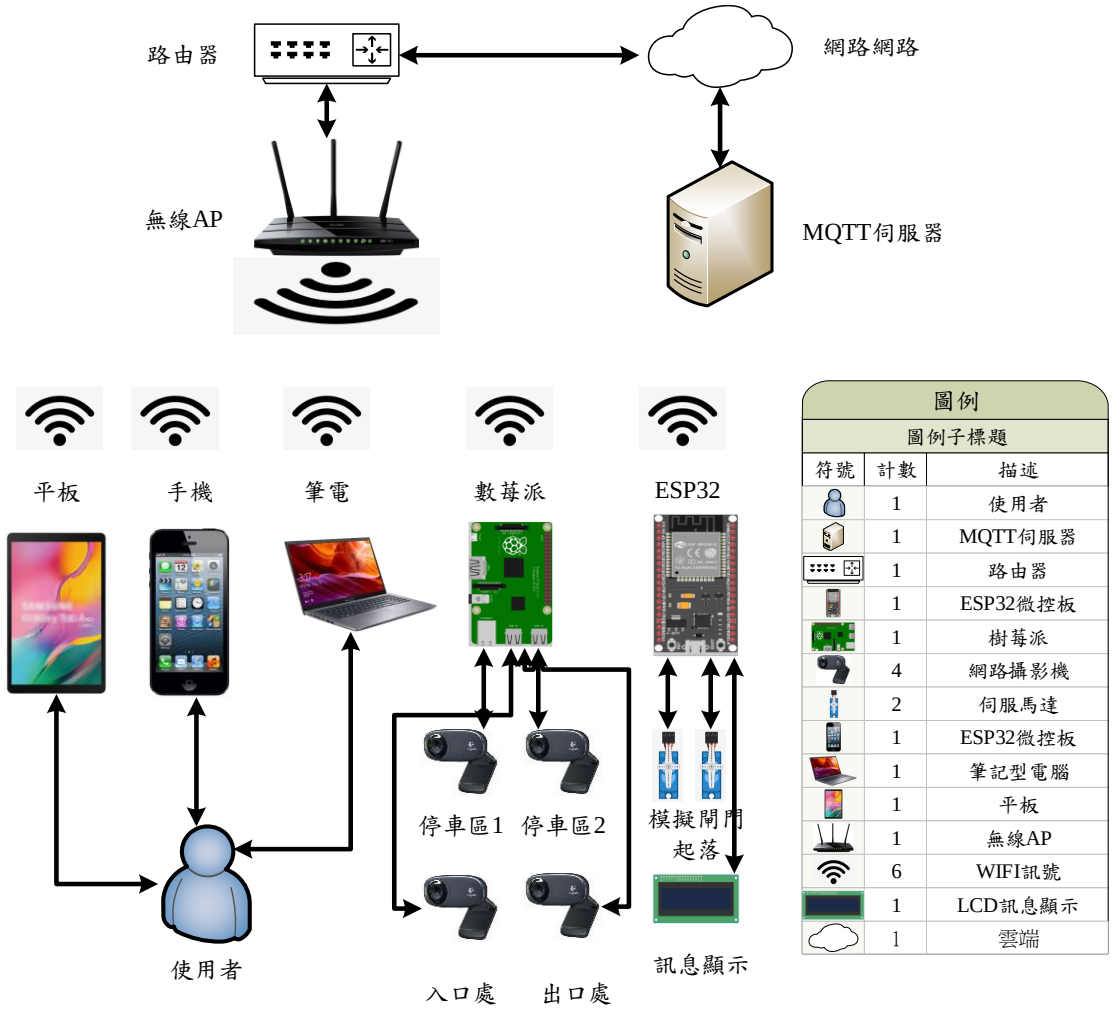


圖 11 自動尋車建置軟硬體系統架構圖

本系統依其資料傳輸量的大小使用不同的通訊協定傳送不同的資料，其中 TCP/IP、UDP、HTTP 等通信協定主要傳送進場及停車位位置的影片資料，而 MQTT 協定傳輸汽車進出場相關訊息，如車牌號碼、進場時間資料、停車位置及與 ESP32 微控板進出場模擬控制訊息交換等輕量資料。由於大家對於 HTTP 相關協定比較熟悉，這裡僅就 MQTT 協定稍加說明。

MQTT 消息隊列遙測傳輸（Message Queuing Telemetry Transport，縮寫 MQTT）是一個能在極差的網際網路聯結下而設計的一種通訊協定，它工作在 TCP/IP 協議族上，是物聯網其中一種通訊協定，必須安裝 MQTT 服務的軟體在伺服器上運作，通常這台主機也稱為 Broker Server。

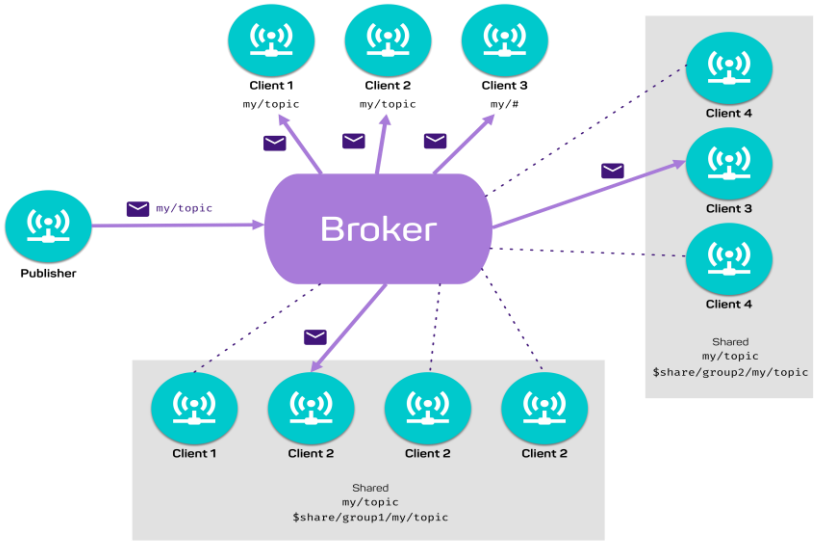


圖 12 MQTT 物聯網資料傳遞示意圖

由圖 11 自動尋車建置軟硬體系統架構中得知本系統需要 MQTT 的服務，考量使用免費 MQTT 服務也是可行，但考量網路品質與避免延遲原則，我們在數莓派中安裝 Mosquitto 免費軟體提供 MQTT 服務，下圖 13 為本系統安裝於數莓派 Mosquitto 版本免費軟體的安裝索引。

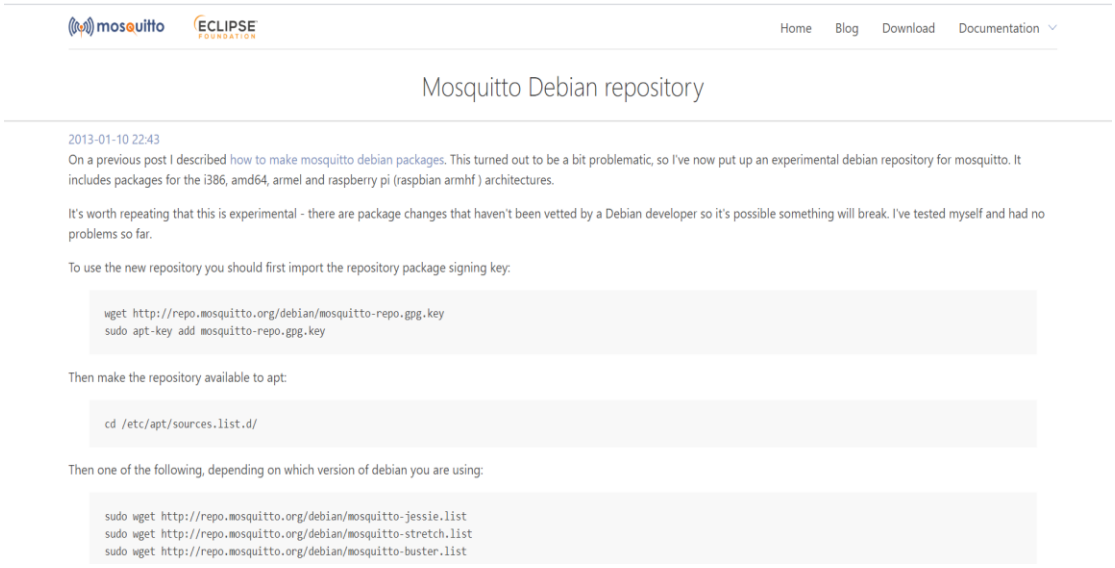


圖 13 Mosquitto 版本免費軟體的安裝索引

四、 系統各相關功能介紹

樹莓派（Raspberry Pi）是基於 Linux 的單晶片電腦，由英國樹莓派基金會開發，是一款超小體積的計算機其處理器運算能力更強，由於具有作業系統，適合做一些需要較多運算和功能的項目。開發上具有低價硬體及自由軟體開發的特點，對於學校推動電腦科學教育非常適合，如下圖所示。

圖 14　RASPBERRY PI 4　實體外觀圖

圖 15 網路攝影機

本系統利用樹莓派開發主要完成影像辨識、影像分割定位、MQTT 伺服器等功能，如下圖所示。本作品共使用了 4 個網路攝影機（如上圖 15 所示），進場出場各一台以及停車處兩台，停車處的網路攝影機將拍到的影像切割成四等分，再單一做車牌辨識，進出場的網路攝影機則不做切割處理。

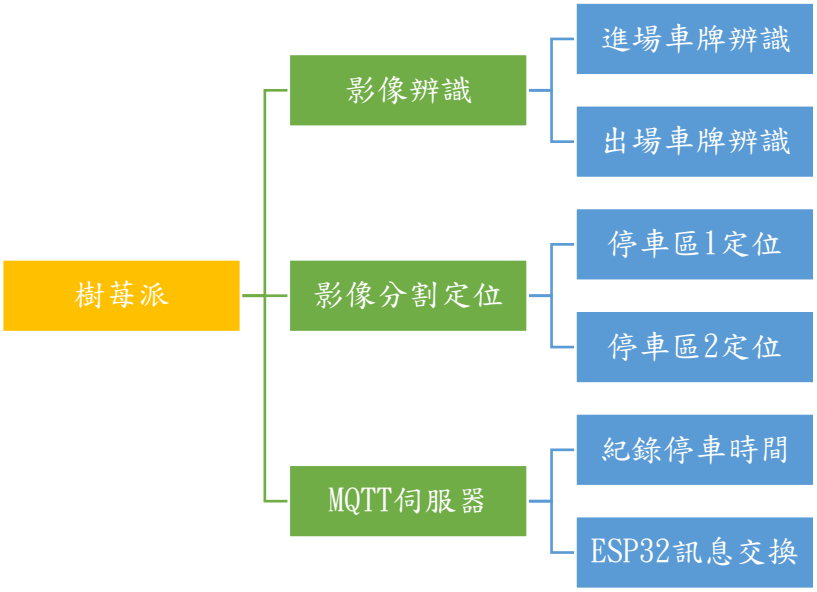


圖 16　樹莓派執行的相關功能

本系統搭配 ESP32 開發板模擬停車場流程使實驗過程生動有趣，ESP32 主要負責進出場閘門起落控制及停車場訊息控制，詳細的內容請參照下圖所示。ESP32 開發板是一低成本，低功耗的單晶片微控制器，整合了 Wi-Fi 和雙模藍牙。 ESP32 系列採用雙核心和單核變體，內建天線開關，RF 變換器，功率放大器，低雜訊接收放大器等模組。

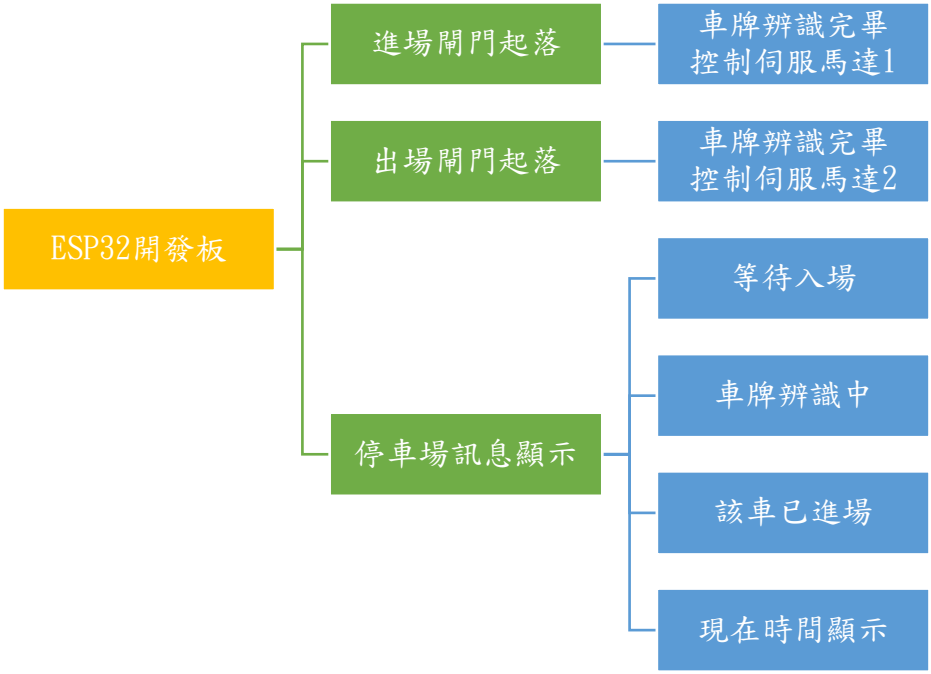


圖 17　樹莓派執行的相關功能

本系統在行動裝置 APP 程式的開發採用 Android Studio 整合式開發環境，透過手機或平板可以查詢愛車及停車場的空位等訊息，如圖 25 所示

（一）影像辨識功能介紹

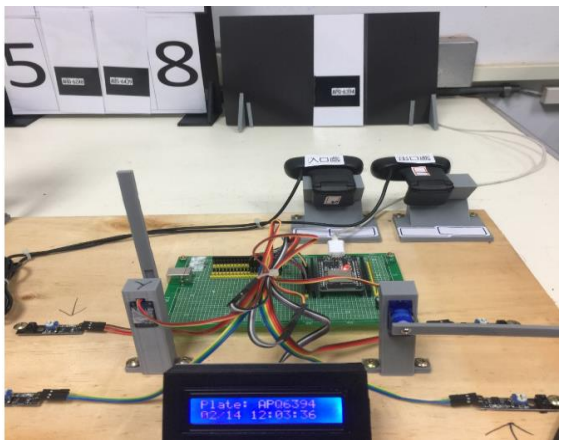


圖 18 完成車牌辨識入口閘門舉起

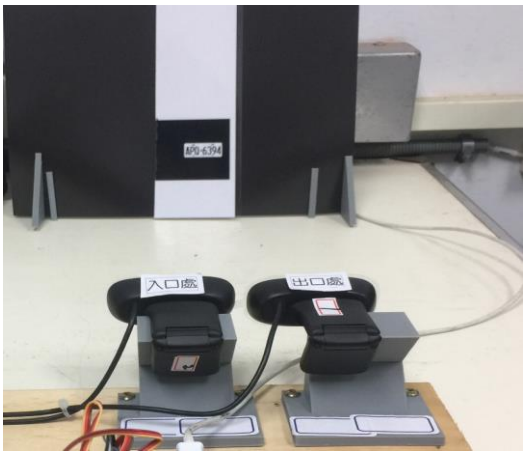


圖 19 APQ6394 汽車進場入口

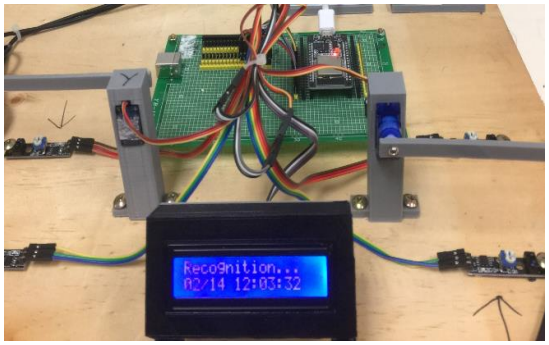
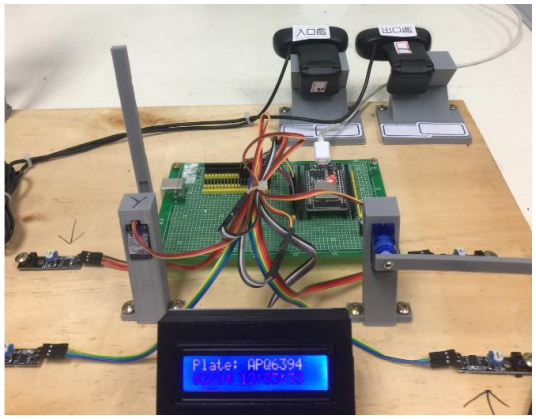
	

圖 20 車牌辨識中 Recognition..	圖 21 辨識完成顯示 APQ6394

（二）影像分割定位介紹

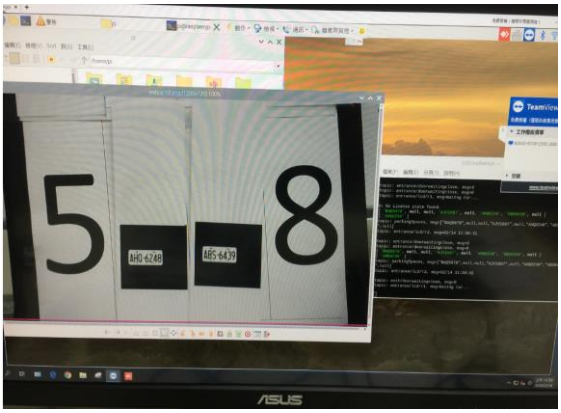
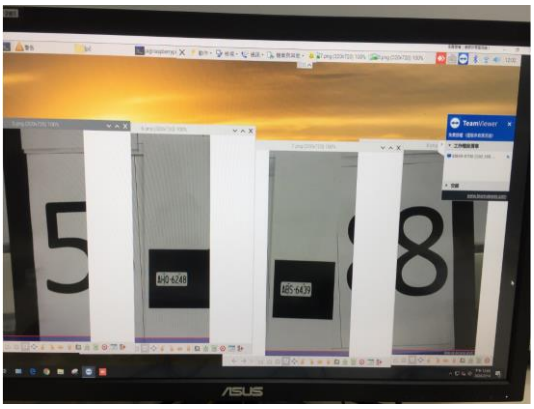
	

圖 22 停車區 2 於 6,7 停車位停車	圖 23 分割成四個影像辨識定位

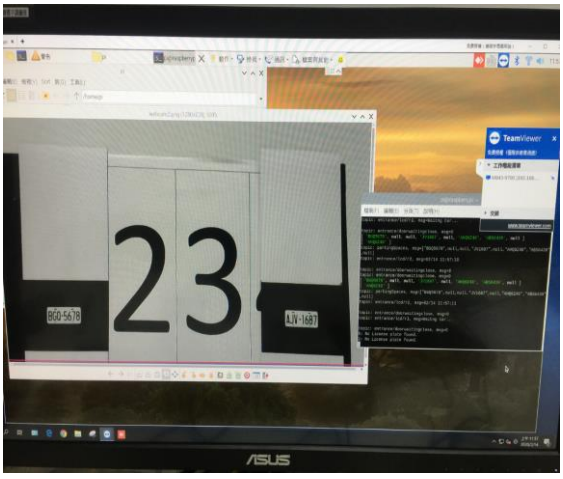
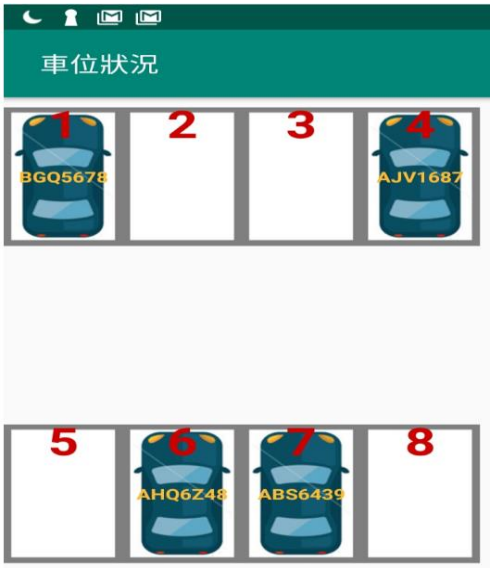

	

圖 24 停車區 1 於 1,4 停車位停車	圖 25 行動裝置顯示停車現況

陸、 研究結果

本系統在小組成員的通力合作下，歷經三個多月的不斷測試與修改，其研究測試結果大致上符合當初規劃的功能，相關功能展示如下各圖所示：

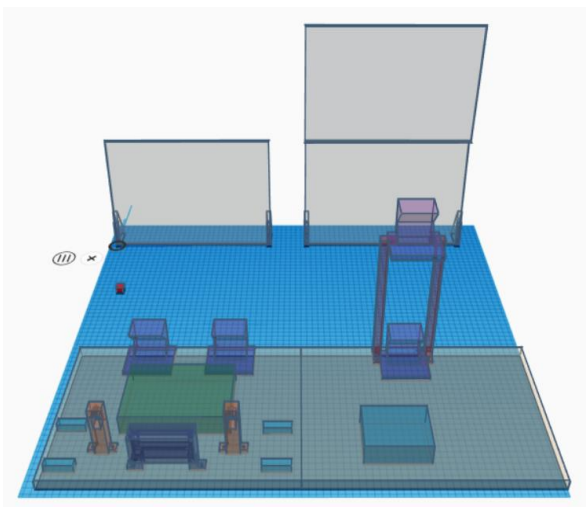


圖 26 本系統模型外觀 3D 設計圖

圖 27 本系統完成實體圖

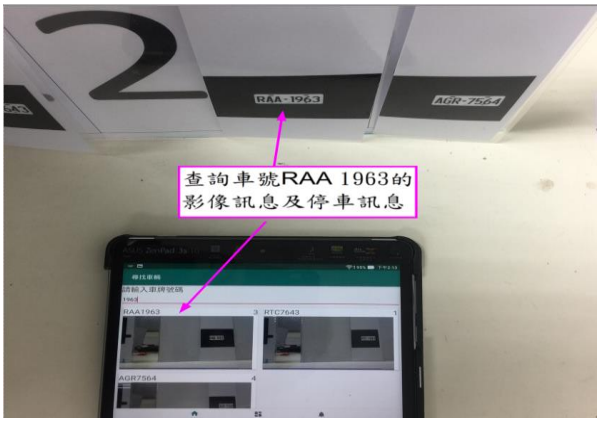


圖 28 行動裝置查詢停車相關訊息

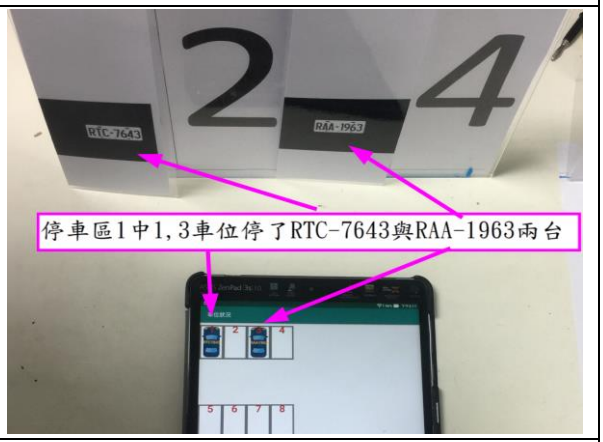


圖 29 行動裝置查詢停車現況

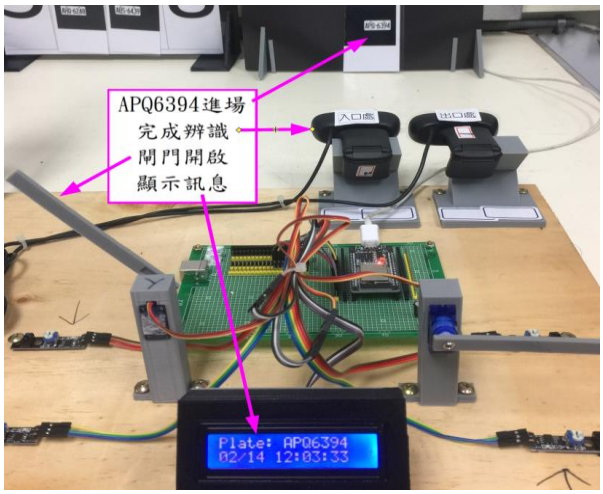


圖 30 行動裝置顯示停車現況

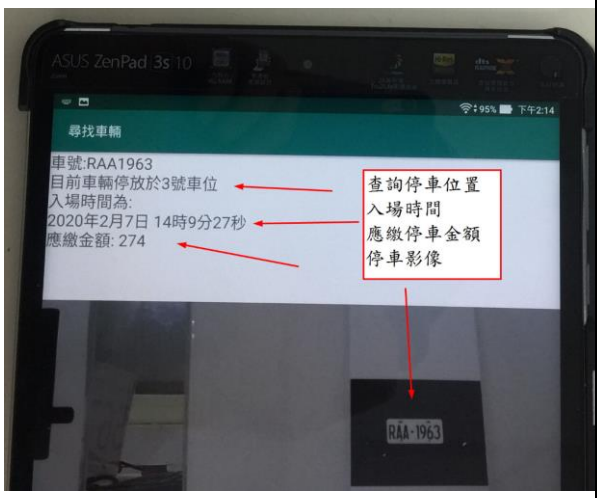
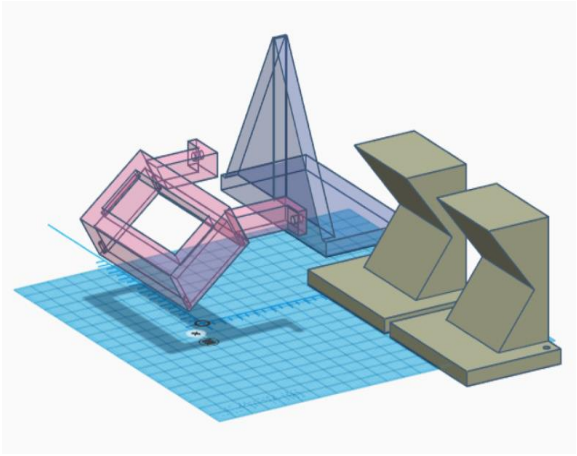
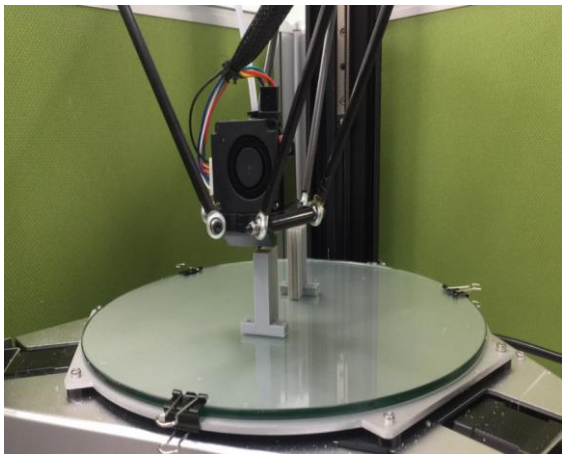
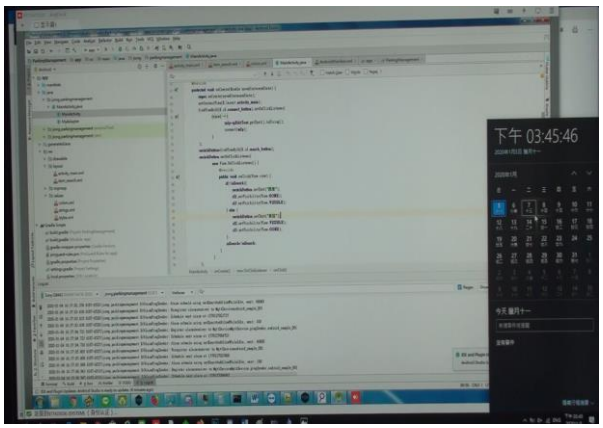
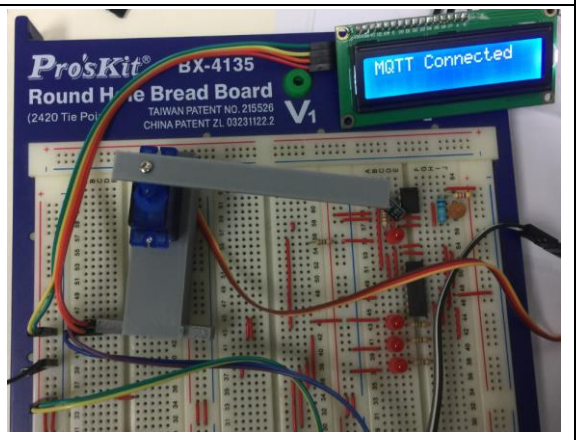


圖 31 行動裝置顯示停車現況

在此次的專題研究製作過程中，感謝科上的師長及同學的協助及提供問題解決的建議，茲將過程中我們實作的照片羅列如下:(圖 32 至圖 37)

<table>
<tr><td> 
</td><td>  </td></tr>
<tr><td>圖 32 樣本車牌製作測試</td><td>圖 33 影像分割辨識定位測試</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>圖 34 停車場 3D 模組元件設計</td><td>圖 35 停車場 3D 模型列印</td></tr>
<tr><td>  </td><td> 
</td></tr>
<tr><td>圖 36 本系統程式開發設計</td><td>圖 37 麵包板電路模擬測試</td></tr>
</table>

柒、 討論

Ｑ１：在製作過程中，利用樹莓派開源軟體進行車牌辨識遇到甚麼困難？解決方案為何？

Ａ１：由於本系統需要建構模擬一個停車場的情境，並且把車牌資料傳至 MQTT 雲端，以方便資料的查詢，所以也必須模擬實際進場車牌辨識的過程。由於是使用別人分享的開源軟體，車牌辨識的過程一直卡關無法正確辨識網路攝影機所拍攝的車牌。後經由師長的建議首先確認攝影機拍攝的效果及清晰度，再來就是取樣用的車牌號碼的製作是否符合該軟體當初的訓練樣本字體。後來發現一開始使用舊型的網路攝影機無法對焦，擷取的照片模糊不清無法辨識，換了一台新的網路攝影機後影像品質大大改善，但是其辨識度仍然很差，於是回去仔細閱讀說明，終於找到最終原因就是車牌的字型也要符合，才能正確的辨識車牌。

Ｑ２：在製作過程中，由於要建置停車場出入口閘門與車牌自動辨識的連動控制伺服馬達遇到甚麼困難？解決方案為何？

Ａ２：單純的控制伺服馬達只可以聽到馬達轉動的齒輪聲，必須配合橫桿舉起放下才會生動活潑，然而要安穩的固定伺服馬達卻是一大挑戰，起初都是用雙面膠黏貼在支架上，但是沒多久時間就脫落了有點沮喪。後來，經由師長建議可以上 TinkerCAD 平台，透過 3D 建模設計所需要的伺服馬達支架，把伺服馬達嵌入到支架中並可以直接用螺絲將支架固定在木板上，不僅美觀且非常穩定。後續相關模型的零組件固定也是採用 3D 列印，印出相關零組件的支撐架方便專題的製作。

Ｑ３：在製作過程中，由於暫時無網際網路實體 IP 架 MQTT 伺服器測試？解決方案為何？

Ａ３：在本模擬系統中透過行動裝置 APP 可以查詢愛車的停車位置與影像，所以傳送的資料有兩種分別是車牌相關訊息的小型資料及網路攝影機所拍攝的大型影像資料。所以，在網路的通訊協定上會採行 MQTT 與 HTTP 相關協定作為資料傳輸的通道。其解決方案為把樹莓派開發板、ESP32 開發板及手機同時連上同一台無線 WIFI AP 先由內部的區網互連進行功能測試。

捌、 結論

在物聯網雲端時代，生活周遭許多事物得以改善，然而科技始終來自於人性，並不是所有地方都需要物聯網雲端技術，有需求才會研發。本作品就創意性、實用性及未來展望層面分別說明如下：

（一） 創意性：

1. 利用網路攝影機（webCam）對多格車位進行停車定位，用影像處理分割技術將一個區域的停車位切割成多塊進行車牌定位掃描，可以減少需架設的網路攝影機或感測器以達到**降低成本**的需求。
2. 結合雲端服務技術，開發**遠端智慧尋車 APP**。

（二） 實用性：

1. 可以讓使用者取車時減少尋車時間、即時掌控停車場動態。同時具有防呆系統，忘記把車停在哪裡時，可利用手機 APP 查詢車號，可顯示愛車停放的位置編號、車停放處及入口進場的影像。
2. 管理人員可以快速了解停車場所有資訊。
3. 使用者可以透過 APP 隨地瞭解愛車狀態、是否遭竊、惡意損毀等停車糾紛。

（三） 未來展望：

本系統由於只有使用一台樹莓派微控板負責所有的區域的影像辨識（進出口及停車區域）與影像分割，影像處理速度上稍嫌不足，導致結果呈現會稍微延遲。未來若有繼續延續研究主題的話，將會使用多台樹莓派微控板負責不同區域的影像辨識與分割。

（四） 學習心得：

1. 此次的專題的完成充分落實務實致用的課綱精神，凸顯團隊解決問題及小組合作學習之重要。專題實作過程中，雖然碰到不少的難題，像是一開始的車牌辨識經過多次測試研究與改善，終於找出最佳的車牌大小與字體。
2. 由於要模擬出停車場進場、停車及出場的流程，需設計相關的出入口閘門、停車場崗位區及網路攝影機的固定架。為了節省製作成本上述的相關模擬支架，皆經由 Tinkercad 網站 3D 建模設計並列印輸出跨域整合學習，發揮自造精神。

3. 經由這次實作，讓我們更了解課本中闡述的理論，與跨領域的技能結合與應用的重要性。過程中因為程式碼的錯誤或是電路接錯的關係而失敗，但從失敗中發掘原因，進而改善、解決問題，並從中吸取新知。感謝師長從旁辛勤指導，提供寶貴的建議和方法，使我們在實作中在專業技能上更加精進，也提升了對於專題製作探究與實作的熱情。

玖、 參考資料及其他

1. 張元翔(2016)，Raspberry Pi 嵌入式系統入門與應用實作，碁峰出版社。
2. Patrick Mulder Kelsey Breseman(2017)，Node.js 物聯網裝置開發，歐萊禮出版社。
3. 施威銘主編(2017)，Android App 程式設計教本之無痛起步，旗標出版社。
4. 黃建庭(2018)，輕鬆玩 Arduino 程式設計與感測器入門，碁峰出版社。
5. 朱鐵斌(2018)，Raspberry Pi 專案實作大全，碁峰出版社。
6. 曹民和 賴勇軍(2017)，透視 3D 列印：Tinkercad 建模設計自己來，拓客出版社。
7. Dominique Guinard, Vlad Trifa(2017)，打造 Web 物聯網：使用 Node.js 與 Raspberry Pi，碁峰出版社。
8. 趙英傑(2016)，超圖解 Arduino 互動設計入門 第 3 版，旗標出版社。

全國高級中等學校專業群科109年專題及創意競賽

「專題組」作品說明書

群　　別：電機電子群

參賽作品名稱： 智慧居家安全系統

關 鍵 詞： 智慧居家系統、安全、防盜

目錄

壹、摘要

現代科技的進步，資訊研究的發展衍伸到可以做出各式各樣在生活上　具有方便性與實質性的發明，因此我們想要透過一整套智慧的系統來連結到生活上，做出一個智慧居家安全的防盜系統。利用 RFID 加上 App 達到解開門鎖的功能，內部還有一氧化碳警報器，只要偵測到一氧化碳，蜂鳴器就會發出警報，以及窗戶開啟的警報器，還有室內溫度的感測顯示跟智能插座以及即時監控系統。

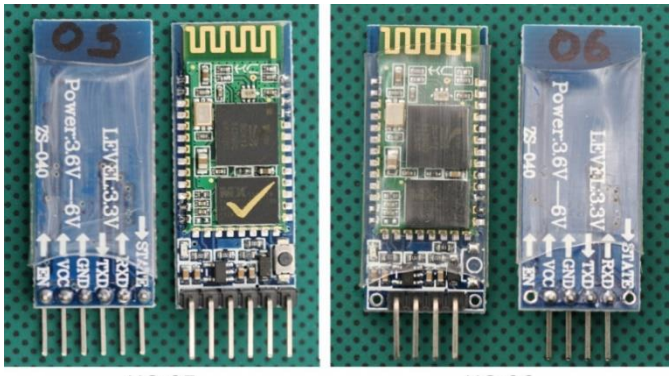
貳、研究動機

長期以來,有各式各樣的業者推出各種不同的防盜鎖或是一些門鎖的控制，但是價格不斐。現在的時代每個人最先聯想到的 3C 產品第一個就是手機,因此想應用人人都有的手機,來連結居家安全的防盜，透過手機藍芽連接到 uno 板來做到透過 RFID 以及需要答案輸入機制的防盜功能，另外為增加居家安全性，又再加裝了一些感測裝置。

參、主題與課程之相關性或教學單元之說明

一、藍芽

我們參考課本上的藍芽教學，學習手機透過藍芽控制 LCD 的亮滅，及數值顯示之後我們改成 Arduino 透過藍芽傳送資料給手機。我們在比對 HC-06 和 HC-05 藍芽時，發現 HC-05 比 HC-06 的指令多，但我們的也只是單純的接收藍芽的資訊，而且 HC05 的價格 HC-06 比 HC-06 高，以上的因素使我們選擇 HC-06 藍芽使用。



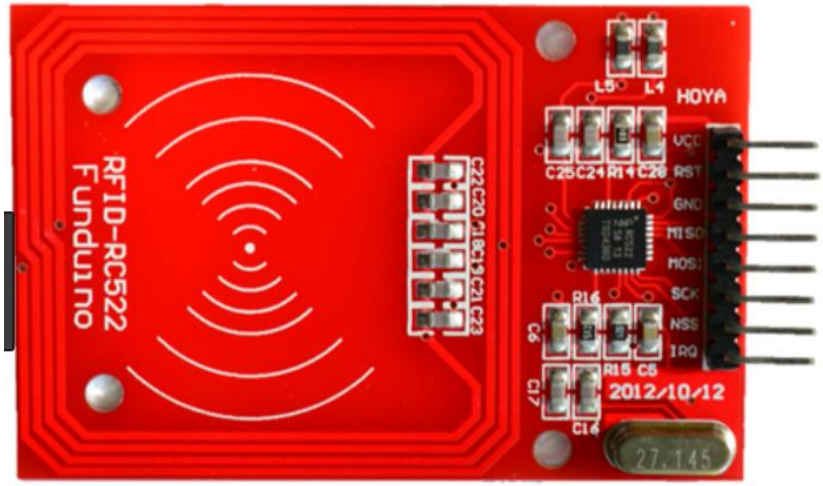
HC-05　　HC-06

圖(一)

二、RFID 掃描器(Mifare)

Mifare 卡內建 EEPROM，應用程式可對它寫入和讀取數據，例如：停車票卡可紀錄車主的停車時間。Mifare 還具備「防衝突處理」機制，也就是避免訊號干擾：若多張卡片同時出現在偵測範圍，Mifare 讀寫器將能逐一選擇卡片進行處理。例如，假設超級市場裡的每個商品都貼上 RFID 標籤，結帳櫃台的 RFID 讀取設備

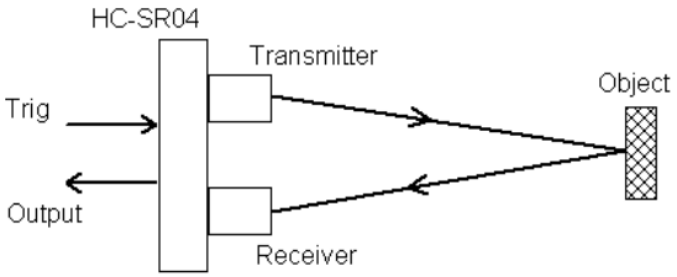
若具備防干擾機制，將能自動掃描堆放在購物推車裡的所有商品，結帳人員不必再手動逐一掃描商品條碼，也不會重複計算。而我們使用的悠遊卡屬於被動式、無內建電源，卡片所需的電力來自讀寫器的電磁場。



圖(二)

三、超音波感測器

超音波感測器是由超音波發射器、接收器和控制電路所組成。當它被觸發的時候，會發射一連串 40kHz 的聲波並且從離它最近的物體接收回音。超音波是人類耳朵無法聽見的聲音，因為它的頻率很高。如圖(三)所示，超音波測量距離的方法，是測量聲音在感測器與物體之間往返經過的時間：

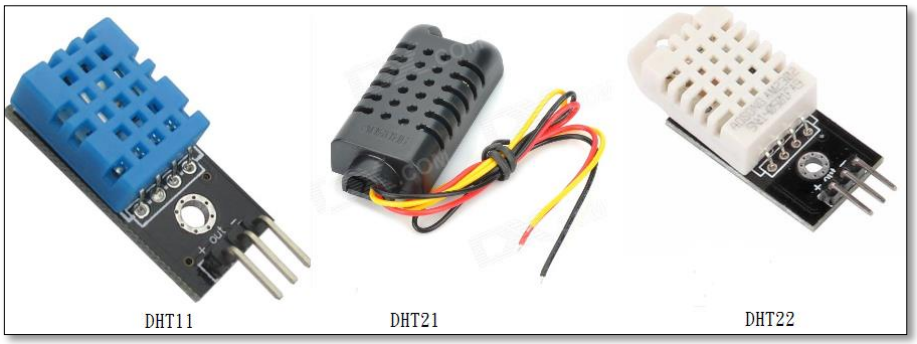


圖(三)

四、溫溼度控制

我們學習課本上的溫溼度感測器，將 DHT11 溫溼度感測器量測溫溼度傳給 LCD 螢幕顯示器。我們選擇 DHT 家族如圖(四)時，比對 DHT11、DHT21 和 DHT22 溫溼度感測器，發現差異是測量溫度的範圍和精準度，因為這些差異對我們機器並沒有很大的影響，之後我們比對他們的價格，發現 DHT11 的價格是最為

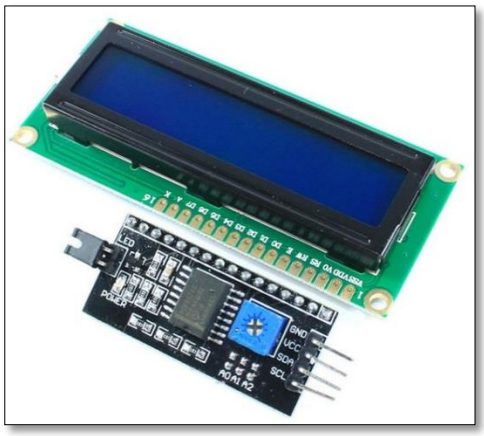
親民的，因此最後選擇 DHT11 溫溼度感測器。



圖(四)

五、串列式 LCD 螢幕顯示器

我們學習如何透過 Arduino 把文字顯示在串列式 LCD 螢幕顯示器上如圖(五)。我們採用 IIC 介面 LCD，設定串列式 LCD 螢幕顯示器顯示 Arduino 接收到的資料，因為只有螢幕只有兩行，我們把室內的溫度跟濕度顯示在上下行。



圖(五)

六、一氧化碳感測器(MQ9)

MQ-9 氣體傳感器所使用的氣敏材料是在清潔空氣中電導率較低的二氧化錫(SnO2)。採用高低溫循環檢測方式低溫（1.5V 加熱）檢測一氧化碳，傳感器的電導率隨空氣中一氧化碳氣體濃度增加而增大，高溫（5.0V 加熱）檢測可燃氣體甲烷、丙烷並清洗低溫時吸附的雜散氣體。使用簡單的電路即可將電導率的變化，轉換為與該氣體濃度相對應的輸出信號。MQ-9 氣體傳感器對一氧化碳、甲烷、液化氣的靈敏度高，這種傳感器可檢測多種含一氧化碳及可燃性的氣體。

圖(六)

七、LinkIt smart 7688 Duo

該板特別設計用於智能家居或辦公室 RichIoT 應用程序的原型設計。由於它與 Arduino 兼容，因此可以使用 ArduinoYún 和 LinkItSmart7688Duo 的不同功能，該板特別設計用於智能家居或辦公室 RichIoT 應用程序的原型設計。由於它與 Arduino 兼容，因此可以使用 ArduinoYún 和 LinkItSmart7688Duo 的不同功能不過我們這一塊板子只單純用來驅動 Webcam 而已！

圖(七)

八、Webcam

閉路電視的一種，一般具有視訊攝影/傳播和靜態圖像捕捉等基本功能，它是藉由鏡頭採集圖像後，由網路攝影機內的感光元件電路及控制元件對圖像進行處理並轉換成電腦所能識別的數位訊號，然後藉由並列埠、通用序列匯流排連接，輸入到電腦後由軟體再進行圖像還原。有些則支援乙太網路或 WiFi，內建有處理器及網頁伺服器，接上網路後可連線檢視畫面。

圖(八)

九、AppInventor

我們利用我們三年級課程所學的 appinventor，寫出屬於我們自己的 APP。我們參考課本的內容學習如何運用藍芽 Client 端接收藍芽 HC-06 傳送的資料，並顯示在手機螢幕上。

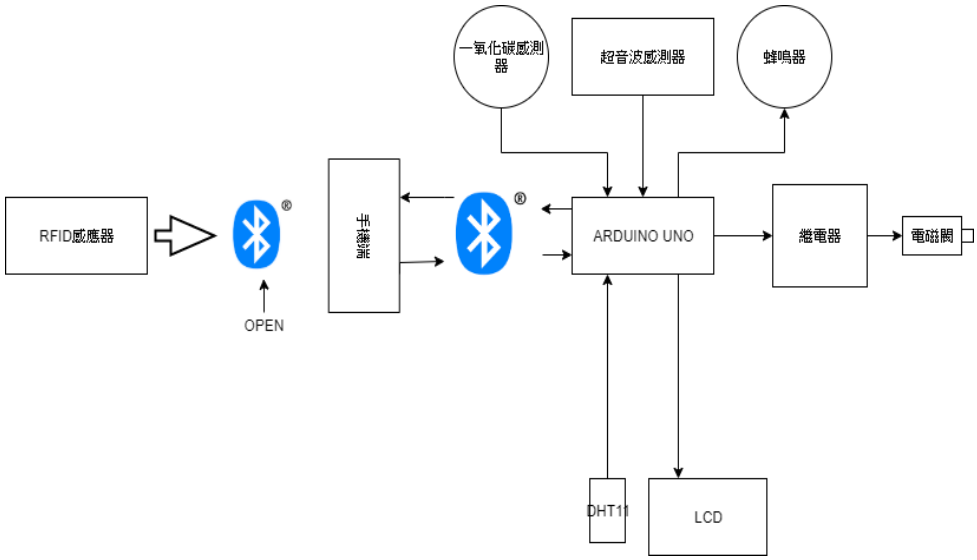
肆、研究方法(過程)

一、流程圖

圖(九)

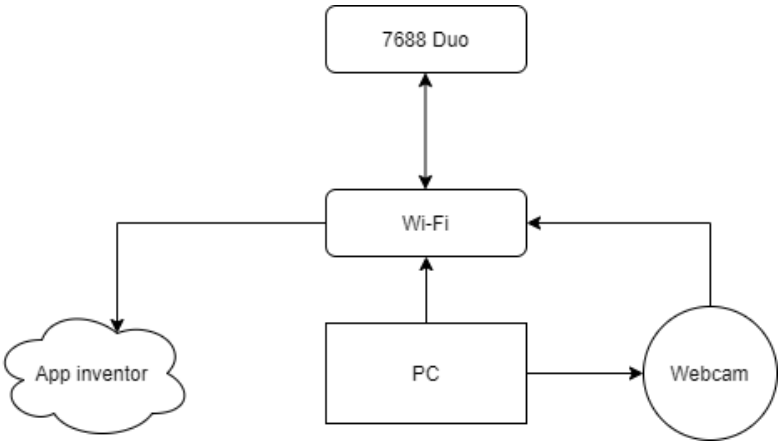
二、系統規劃

如圖(十)所示，分成兩大部分，第一部分是智慧門鎖，當 RFID 感應到密碼卡時，會傳給 Arduino，進而開啟藍芽通訊，而藍芽會一直接收手機傳過來的資訊，當答案正確時就會控制繼電器進而讓鎖打開，輸入答案的方式可用鍵盤或語音輸入。第二部分是屋內安全監控裝置部分，溫溼度顯示、窗戶開啟偵測、一氧化碳警報器等，統一由 Arduino 來控制，居家即時影像監控則透過 Linkit7688 控制，可經由 Wi-Fi 傳至手機。



圖(十)

如圖(十一)所示，當電腦連上 7688Duo 的 Wi-Fi 後，經由電腦開啟網路攝像頭，此時網路攝像頭會連線上 7688Duo 的內網，並把畫面顯示在 192.168.100.1:8080 的 IP 位置中，再將此 IP 位置放置於 APPinventor 中網路瀏覽器的功能中，這樣就可以在手機上看到網路攝像頭投射出來的畫面了。



圖(十一)

三、板子功能規劃

(一)、智慧門鎖

智慧門鎖是我們的主要功能，只要少了密碼卡就不行開門，也同時卡要先掃過，認證完畢，藍芽才會啟動，才可以輸入第二道鎖的答案！

(二)、一氧化碳警報器

我們考慮到，家中有可能會發生火災，瓦斯外洩的情況，所以我們多加裝了一個 MQ-9 來偵測瓦斯，不過因為實驗的時候無法達到完全密閉的空間，所以我們把程式修改為只要偵測到瓦斯，蜂鳴器就會發出警報！

(三)、溫溼度顯示器

接著我們考慮到在家中的氣溫以及濕度，所以我們也加裝了 DHT11，來偵測家中的溫度以及濕度並讓它顯示在 LCD 上面，如圖（十二）。

圖(十二)

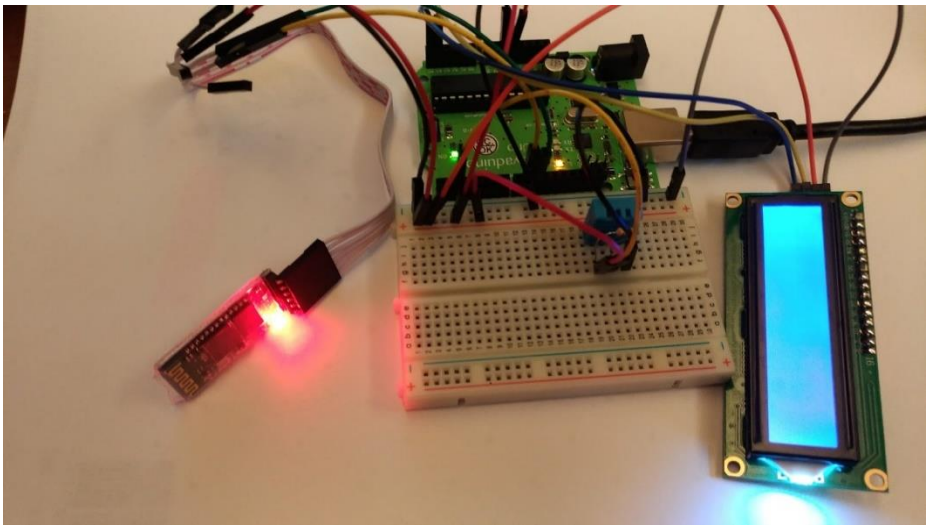
(四)、窗戶開啟偵測器

如果家裡在沒有人注意的情況下,有入侵者從家中窗戶進入,只要一開啟窗戶,窗戶被打開後,就會被偵測器感應到發出警報聲響! 如圖（十三）

圖（十三）

(五)、藍芽傳送模組

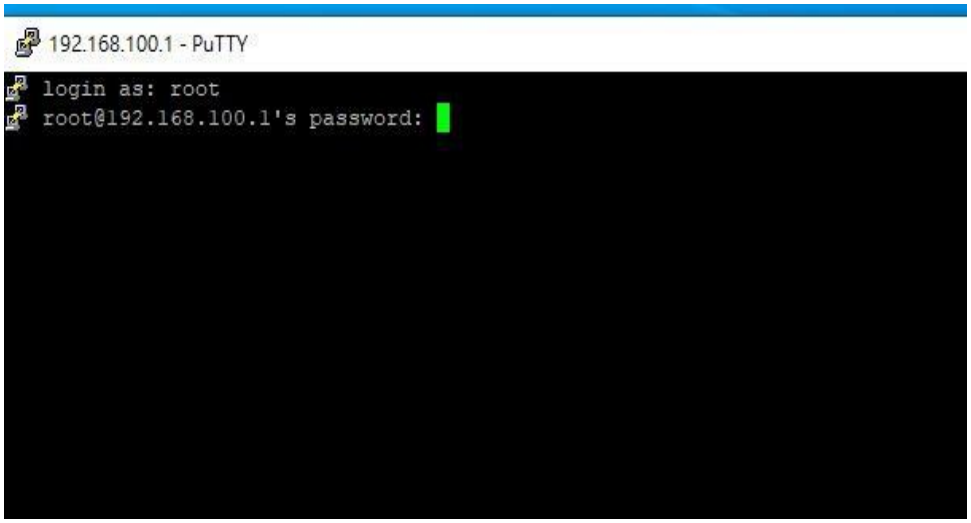
我們把藍芽模組 HC-06 加上去，如圖（十四），讓機器讀取到的資料可以透過藍芽模組 HC-06 傳送到手機裡。只要手機與機器不超過 10 公尺以上，手機就可以接收到資料，為了解鎖，我們利用藍芽，進行門鎖的第二道認證。



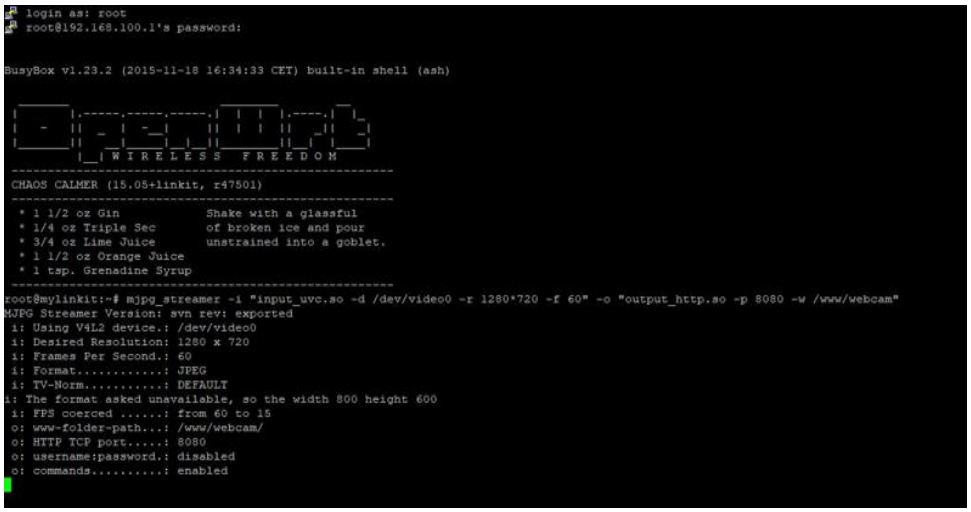
圖(十四)

(六)、居家即時監控

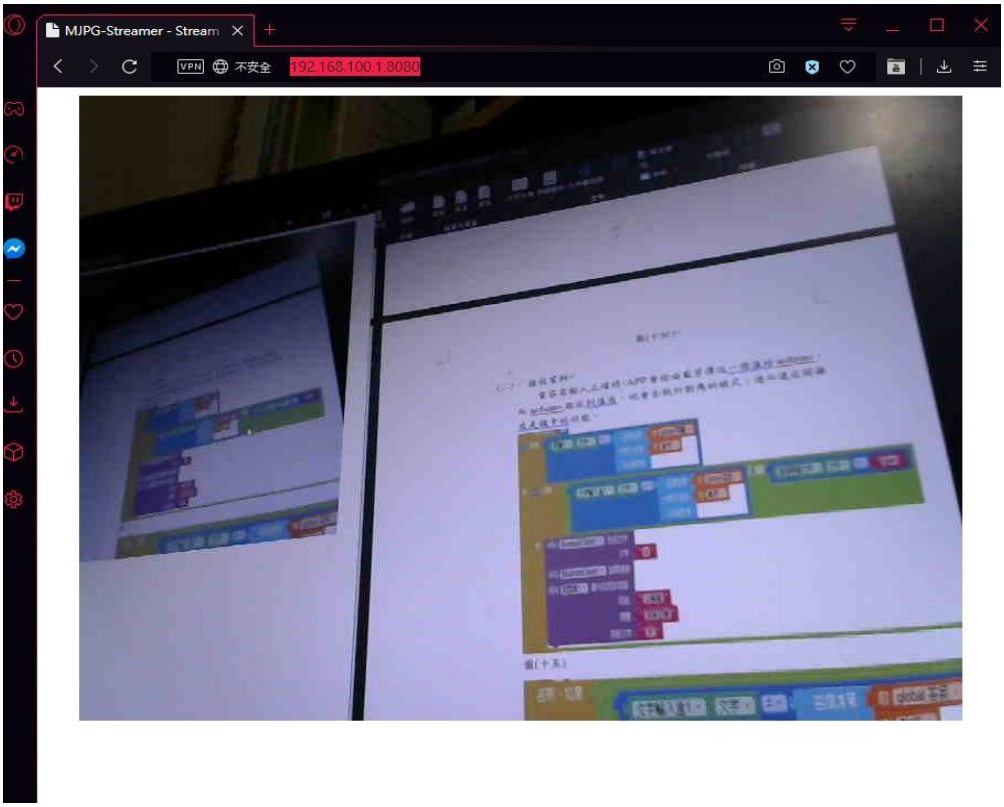
我們除了使用 ArduinoUNO 版之外，還有使 LinkIt Smart 7688 Duo，用來開啟攝像機，並且提供攝像機一個內網讓它可以將畫面顯示到網路上。



圖(十五)



圖(十六)



圖(十七)

四、製作 APP

(一)、APP 介面

這一個 APP 是用來輸入我們第二道鎖的通關密碼，輸入密碼的方式可選擇用文字輸入或語音輸入的方式來進行解鎖，當輸入超時或錯誤達三次時，會被鎖卡。

同時在 APP 的介面中，若有連接到 7688 的 Wi-Fi 的話，可以看到圖(十八)中下方的畫面。

1.開啟 APP 的介面，連接藍芽的介面

圖(十八)

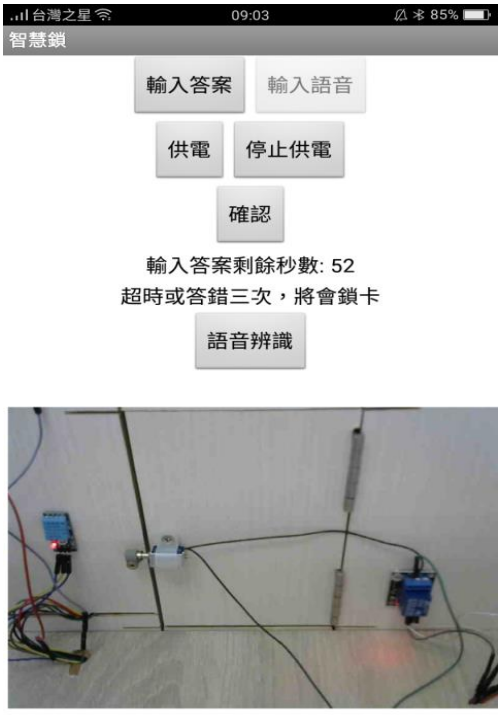
2.連接藍芽後，所顯示的畫面，將會出現控制智慧插座的狀態，以及輸入解鎖密碼的按鈕。如圖(十九)

圖(十九)

3.選擇使用文字輸入答案所顯示的畫面。如圖(二十)

圖(二十)

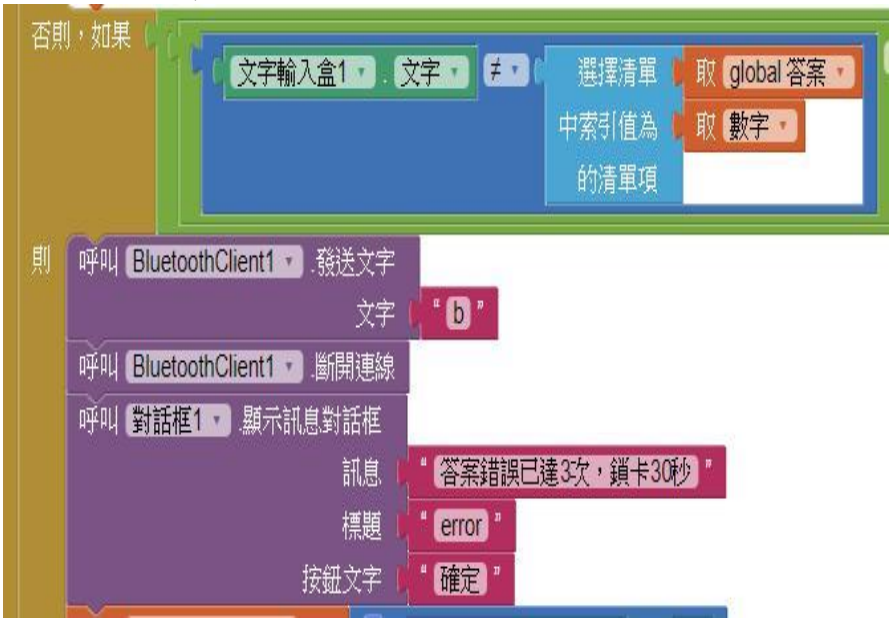
4. 選擇使用語音輸入答案所顯示的畫面。如圖(二十一)



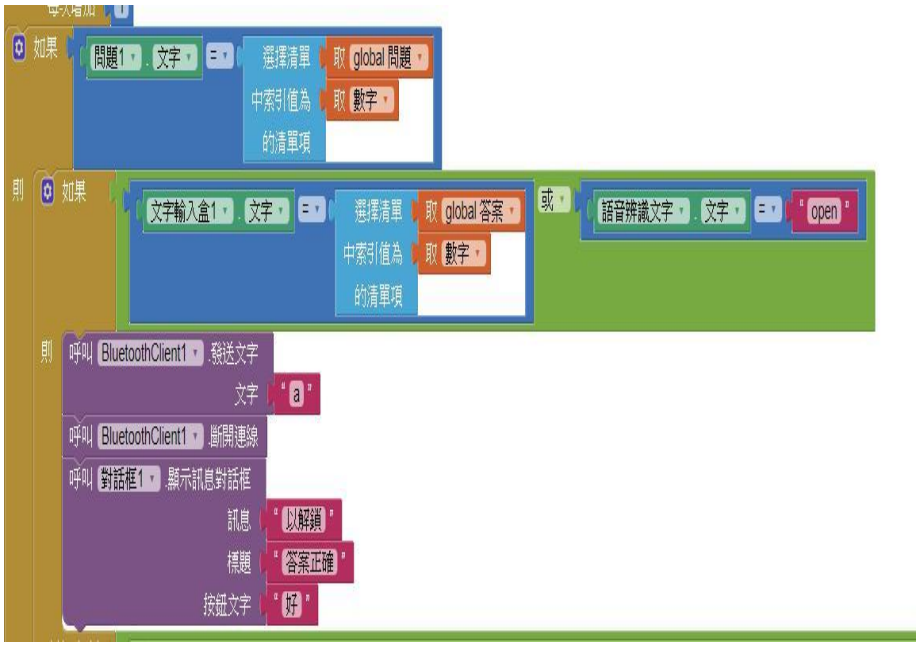
圖(二十一)

(二)、接收資料

當答案輸入正確時，APP 會經由藍芽傳送一個值給 arduino，而 arduino 接收到值後，他會去執行對應的程式，進而達成開鎖或是鎖卡的功能。

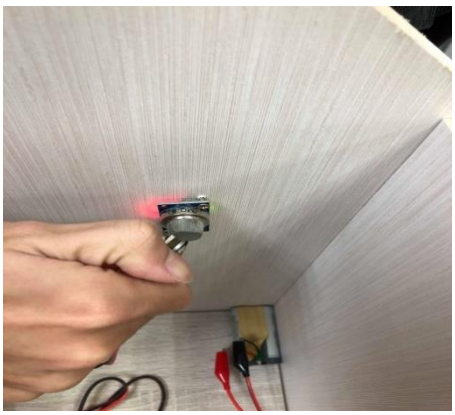
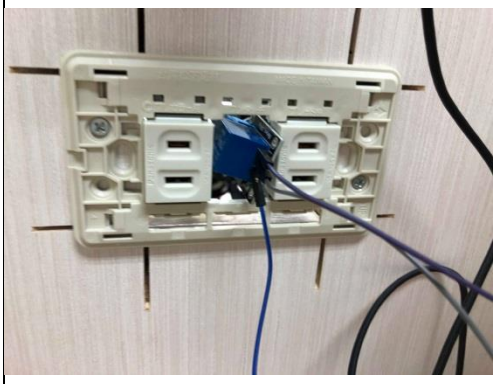


圖(二十二)



圖(二十三)

伍、實體組裝

<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>一、　窗戶開啟偵測器</td><td>二、　一氧化碳警報器</td></tr>
<tr><td>  </td><td> 
</td></tr>
<tr><td>三、　智慧插座</td><td>四、　溫濕度顯示器</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>五、　居家即時監控</td><td>六、　智慧門鎖</td></tr>
</table>

陸、研究結果

透過 Arduino 與 Appinventor 的連動，以及 Webcam 的網路攝像機，可以預防闖空門、火災、監控溫溼度，只要一台手機，就可以即時監控家中的狀況。

(一)、利用 7688duo 的功能開啟 Webcam 並使用 Appinventor 的功能，讓網路攝像機可以利用 7688 的內建 IP 畫面顯示在手機上。

(二)、利用 Arduino 撰寫整個系統所需要的功能，包含了瓦斯感測器、溫度顯示器、RFID 解鎖、智慧插座等等。

(三)、利用 Appinventor 製作手機解鎖的 UI 介面和當手機到連接 7688duo 的內網已達到顯示網路攝像機的畫面，以及回傳各功能所需的數值到藍芽，來驅動 Arduino 的程式。

圖(二十四)

柒、討論

我們目前系統使用的連接方式是透過藍芽，因為藍芽可以一對一，不怕他人也可以同時控制，在同一時間，只能有一人進行操作。

未來我們會將程式修改為使用 WIFI 達到更方便的使用，只要人接近會自動連接到 WIFI，不用再先連接藍芽才能解鎖，以達到未來社會所必備的物聯網功能，在網路上也能監視家中目前的狀況，並加以控制，未來也會增加更多功能並簡化程式碼，讓程式變得更便利，更好使用!

捌、結論

在製作專題之前，我們在思考如何應用之前的所學去想出專題的內容，我們一開始是用 Arduino 和 APPinventor，一開始我們只會一些基本的語言，但後來自己去找課本和網路上學習和找資料，就寫出比一開始進階很多的內容了，我覺得程式語言是我們在這次專題上最大的突破，原本我的學習的都只是一些基礎，但現在我們已經可以靠自己的能力寫出很不一樣的內容了！製作完智慧居家安全系統後，我們發現如果可以去紀錄平常我們使用的時間，例如哪個時段可能會比較頻繁的解鎖，就可以在比頻繁的時段減少這次到下一次解鎖的時間，如果這個時段每達到幾次的解鎖，就相對地去縮短在那之間的時間，盡而達到具備安全還有方便性的效果。

在製作專題之前，我們在思考如何應用之前的所學去想出專題的內容，我們一開始是 Arduino 和 APP inventor，一開始我們只會一些基本的語言，但後來自己去找課本和網路上學習和找資料，就寫出比一開始進階很多的內容了，我覺得程式語言是我們在這次專題上最大的突破，原本我的學習的都只是一些基礎，但現在我們已經可以靠自己的能力寫出很不一樣的內容了！

玖、參考資料及其他

[1]實戰物聯網 Linkit Smart 7688 Duo CAVEDU 教育團隊，曾吉弘，徐豐智，薛皓云，謝宗翰，袁佑緣，蔡雨錡編著–WIFI 設定&Webcam 設定

[2]超圖解 Arduino 互動設計入門趙英傑著-藍芽設定&DHT11+LCD 顯示器[3]https://swf.com.tw/?p=1027RFID 門禁

[4]http://maker.tn.edu.tw/modules/tad_book3/page.php?tbdsn=70&fbclid=IwAR1w_NhDD4gUcBFUrmAAZun116Ar8eP4hqhkur-dfjjElJWOIyHhO0zY9qcMQ-9 氣體感測器

[5]http://appinventor.mit.edu/explore/Appinventor

全國高級中等學校專業群科 109 年專題及創意製作競賽

「專題組」作品說明書封面

群別：電機與電子群

作品名稱：**讓電費帳單數字變小的工具**

關鍵詞：**ESP8266、Node-Red、ACS712**

目錄

壹、 摘要

處在科技化時代之下，人們日常生活幾乎離不開電器用品，而電器與插座密不可分，當**插在插座上之家電未使用時，亦會在無形中讓電流有些微消耗**，因此本專題之目標是以**減少那些不必要之耗能**來設計作品，本專題所製作的插座，可以讓使用者直接利用智慧手機之 LINE 通訊軟體來查詢，藉以了解插座上電器的使用狀況，**可透過表格及圖表立即觀察到在哪個時段哪項電器消耗了多少功率**，便於統計與分析家中的用電情況。此外本專題亦設計**可以透過 LINE 輸入指令來關閉插座電源**，再配合上述查詢表格及圖表功能，即可省下不必要之能源消耗。

貳、 研究動機

現今人們環保觀念與永續發展意識抬頭，節能減碳人人皆有責，而電器用品與人們之日常生活息息相關，是不可或缺之一部分，但在無形之中電器用品亦會造成些許浪費，而現代大部分家庭中仍沒有將未使用電器時即把其電源關閉之觀念，因電器雖未使用，但電源若未關閉亦會造成能源流失，且電器產品長久接電有可能會電線走火之疑慮產生。因此，本專題製作之插座可用來測量使用電流，同時間接算出功率來檢視家中消耗較大之電器用品，透過此即能對症下藥來節省家庭開支，另外，亦可透過通訊軟體來操控插座，藉以控制電器產品的開關，就不用依靠人力巡視來一一檢視各個開關，且為了安全考量，在**電流超過插座的額定值時，電源會直接關閉**並向手機發送**警訊**。

研究目的

- 學習如何透過 linebot 來達成指令
- 透過實驗了解 ACS712、ESP8266、Relay、Attiny85、ATmega32u4 特性
- 學習使用 Node-Red 來連結各種元件跟設計 linebot
- 本專題拆成三個部分以達到模組化
  - 第一部份:手機端、Node-Red、ESP8266 之連動
  - 第二部份:Attiny85、ACS712、繼電器、電器用品、按鈕之連動
  - 第三部份:ATmega32u4、Google 試算表之連動

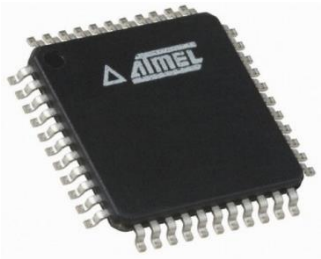
參、 主題與課程之相關性與教學單元之說明

在製作本專題時，有運用到學校所學的基本電學第一章:P = I * V、電子學第一章:$Vrms = \sqrt{\frac{(\text{週期內有效值})^2 * \text{時間}}{\text{週期}}}$、專題製作實習中所學的 Arduino 與 WIFI-ESP8266 單晶片控制，及學校電子學第八章教過 MOSFET 的原理，我們用 MOSFET 做 5V 與 3.3V 準位轉換。

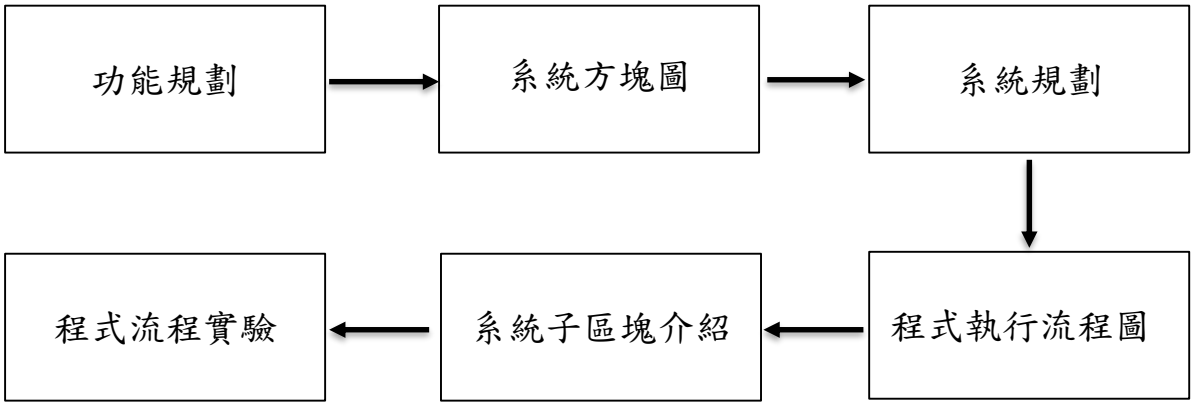
肆、 研究方法(過程)

一、 研究設備及材料

		
ESP8266	**繼電器**	**ACS712**

	
Attiny85	**ATmega32u4**

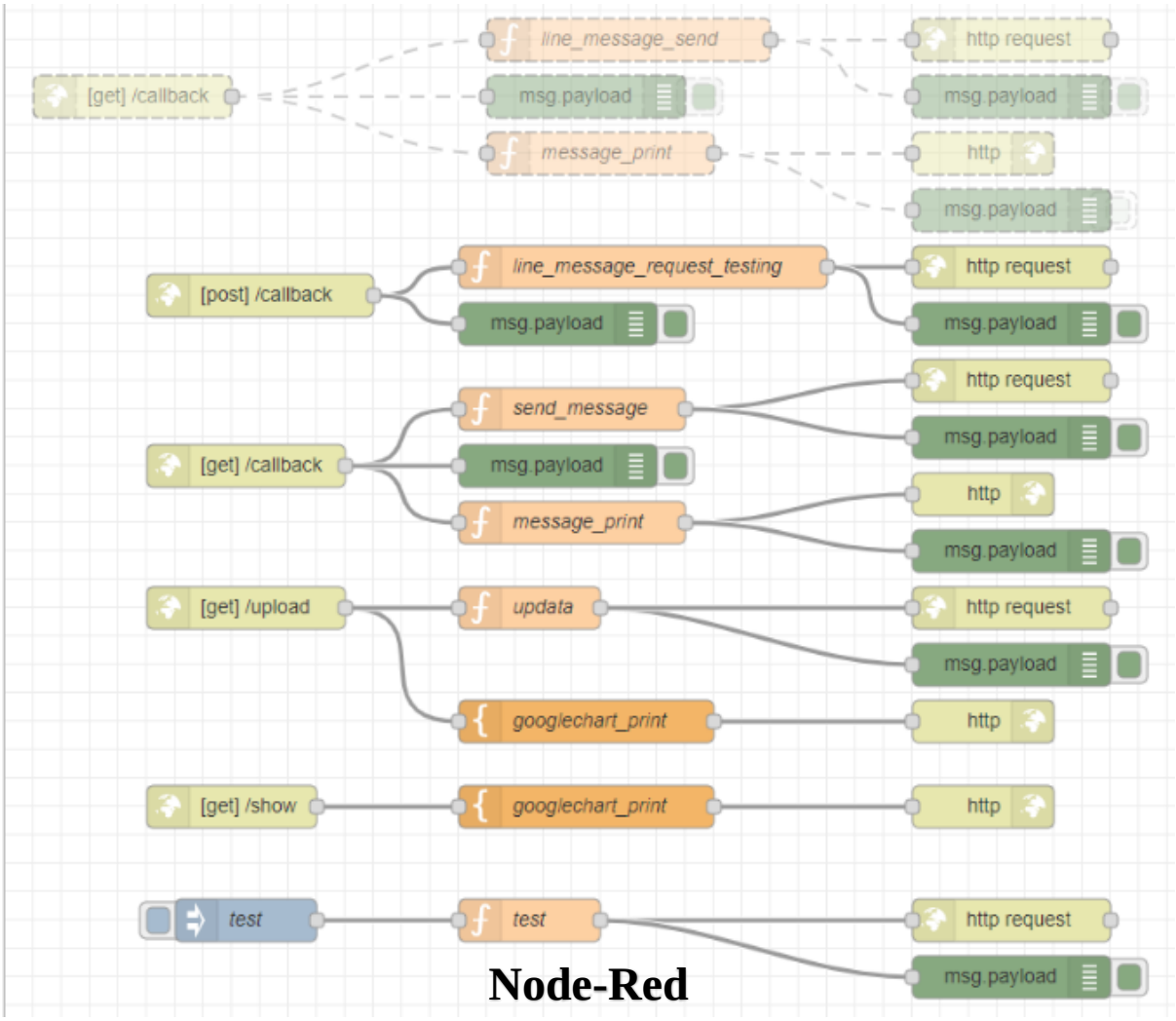
二、 流程圖



三、 功能規劃

1. Node-Red

本專題為何選用 Node-Red，主要是因為方便開發，它是積木式的程式軟體，可讀性較高，對於新手來說較簡單，也容易達到實驗的需求，本專題把它拿來寫 linebot、提供資料給 ESP8266、上傳 ACS712 所測之電流值。



2. ESP8266

ESP8266 是功耗很低的 UART-Wi-Fi 透傳模組，本專題把它作為網卡。

3. Level Shifter

電壓轉換電路，由 N 通道的 MOSFET 和兩顆電阻組成，把 ESP8266 之信號由 3.3V 轉換成 5V。

4. ATmega32u4

本專題選用它主因為其成本低而且跟 ATmega328 相較下少了一個 CH340，用以溝通 Attiny85 跟 ESP8266 並負責偵錯。

I2C 連結

本專題選用 I2C 而非 UART 之原因為， I2C 有兩條資料線，且可並聯 256 個裝置以達到模組化，故本專題使用 I2C 連接 ATmega32u4 跟 Attiny85。

5. Attiny85

本專題選用 Attiny85 主因是為了要模組化，方便擴充，故把 Attiny85、ACS712、繼電器、按鈕裝在一張子版上，並將 ESP8266 當作母板透過 I2C 協定傳輸，且 Attiny85 體積小、腳位剛好，非常適於本實驗之需求，故本專題把它拿來控制繼電器。

6. 繼電器

本專題把繼電器拿來控制開關，透過 linebot 輸入指令即可遠端操控電器用品的電源。

7. ACS712

ACS712 是霍爾元件，易受電磁干擾，在本專題用來測量電器用品的電流。

8. 按鈕

另外本專題使用按鈕來手控電器用品的開關。

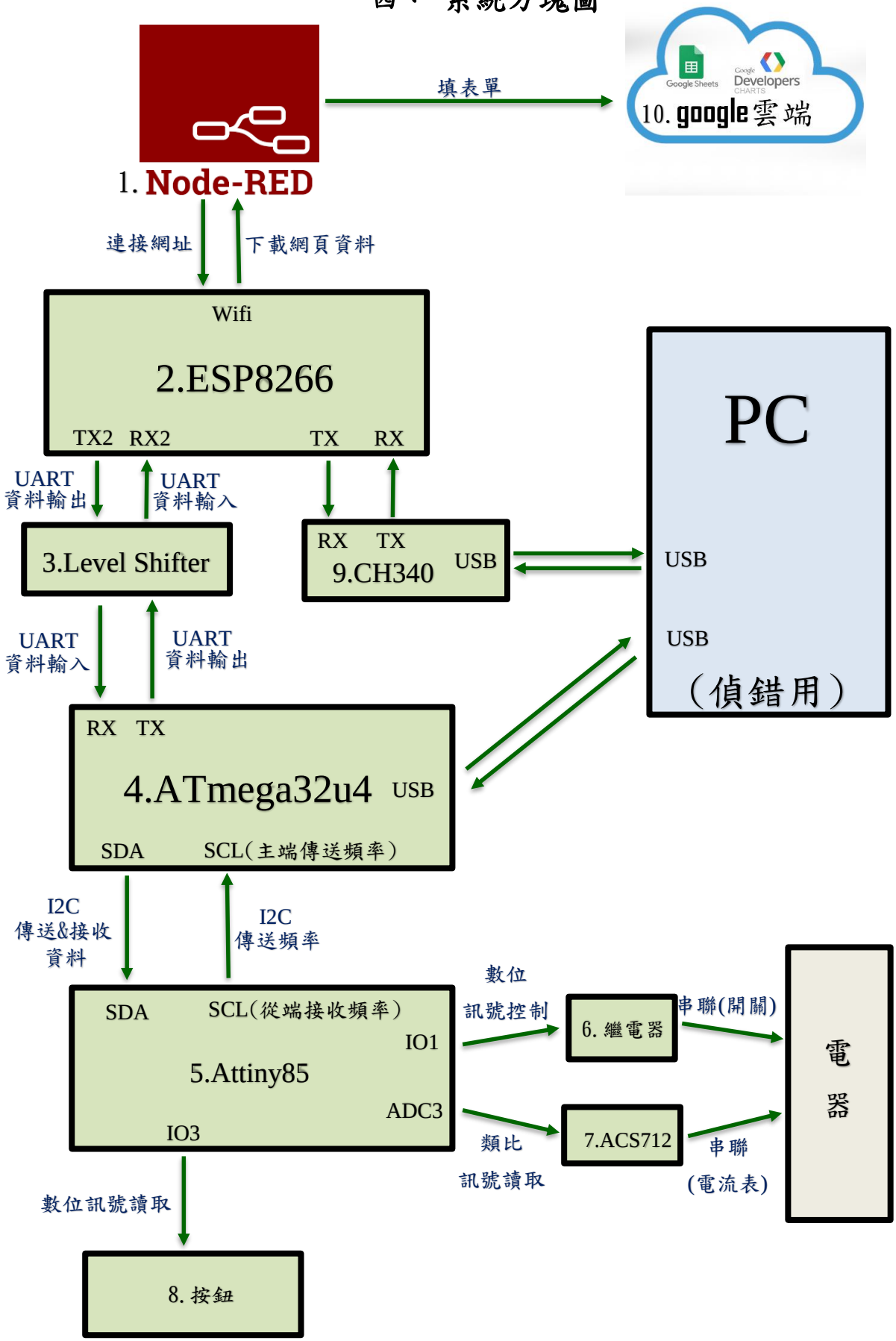
9. CH340

USB to ttl IC

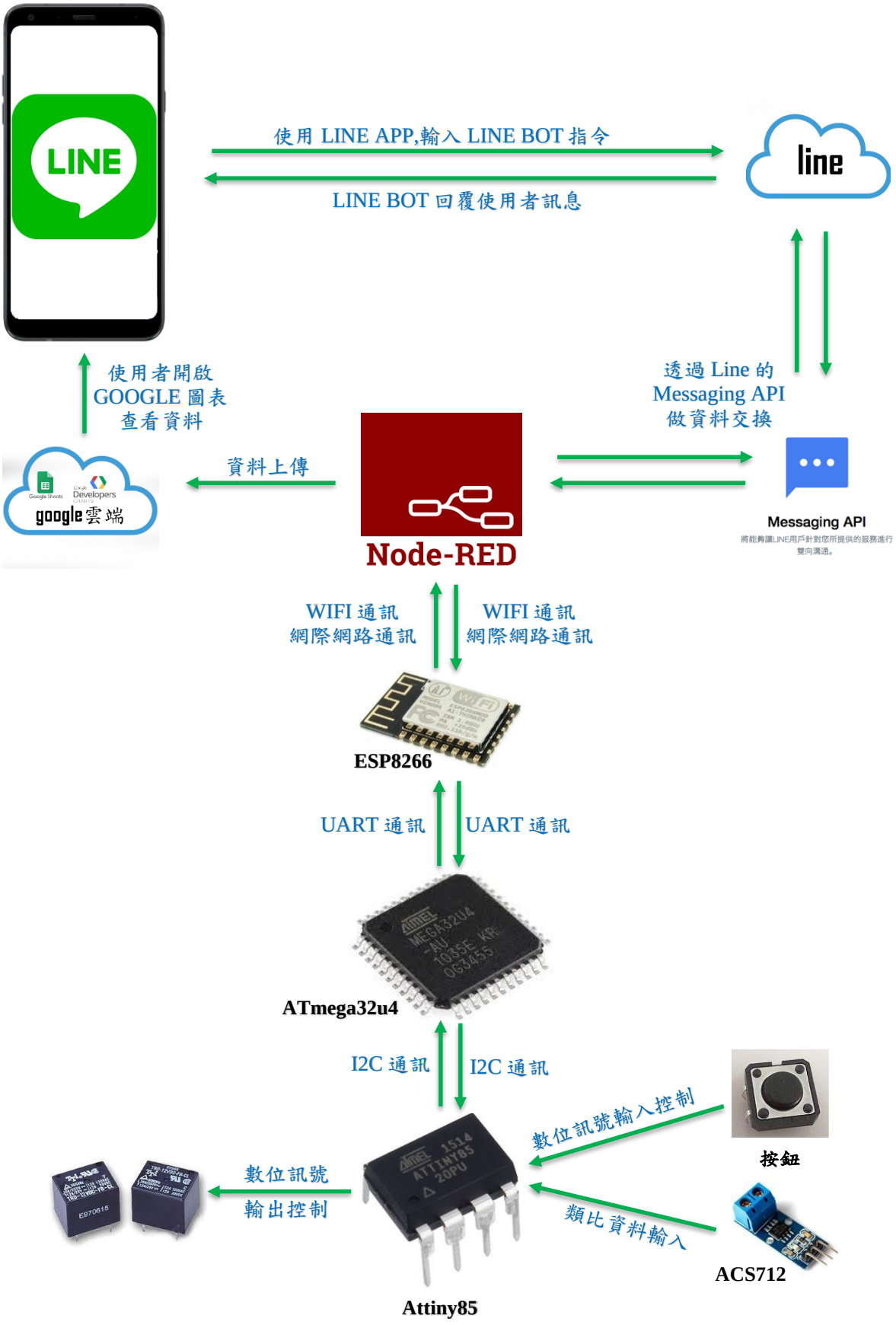
10. Google 試算表

本專題把 Google 試算表當成資料庫，用來存取所需資料，而且 Google 試算表便於與 Node-Red 連結，不需要再特別寫其他的程式語言(ex.MySQL)，而此插座每隔一段時間會將 ACS712 所測量到的電流上傳至 Google 試算表並算出功率，當使用者在 linebot 輸入指令，即可查看電器所消耗之功率。

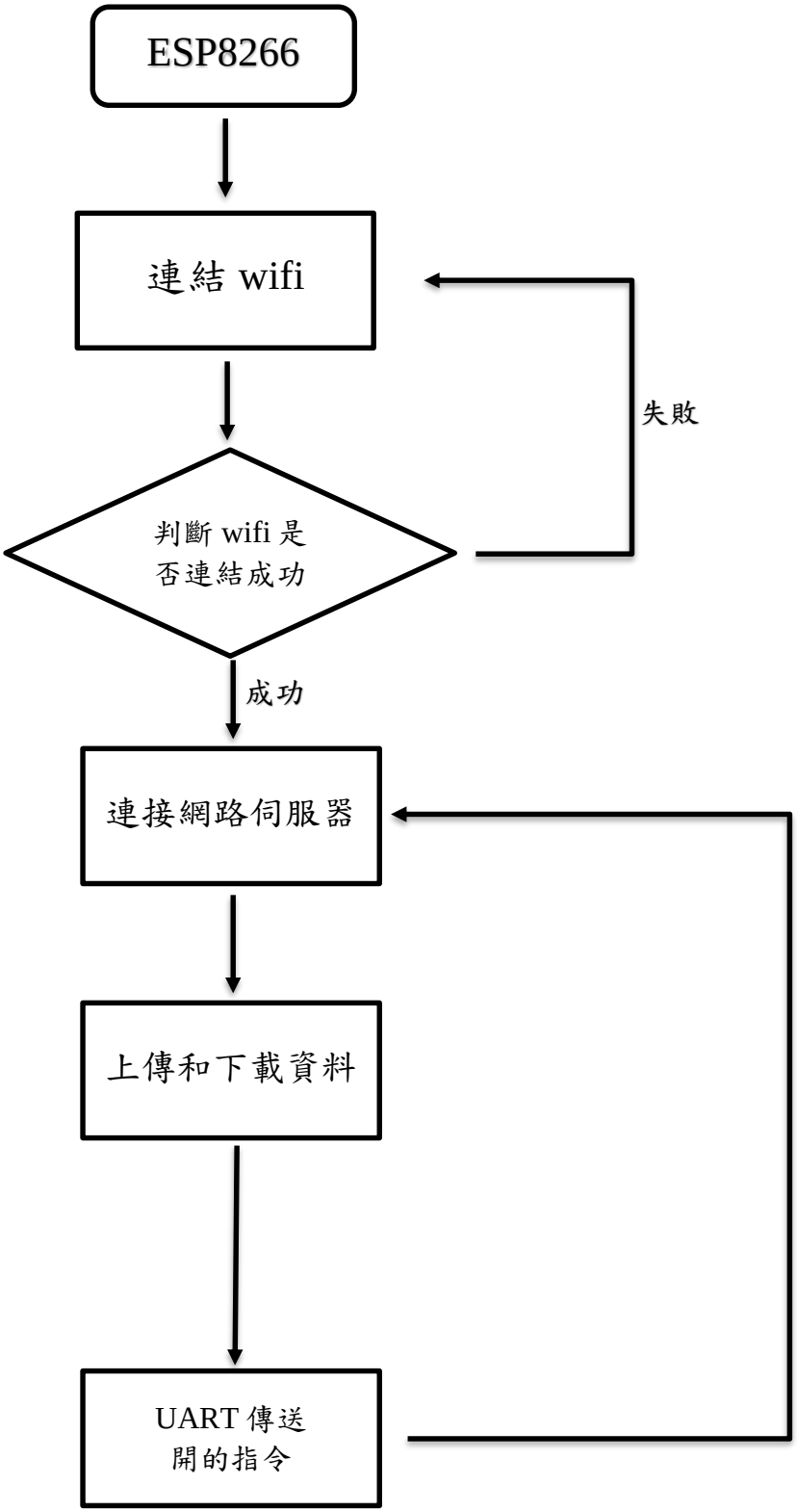
四、 系統方塊圖

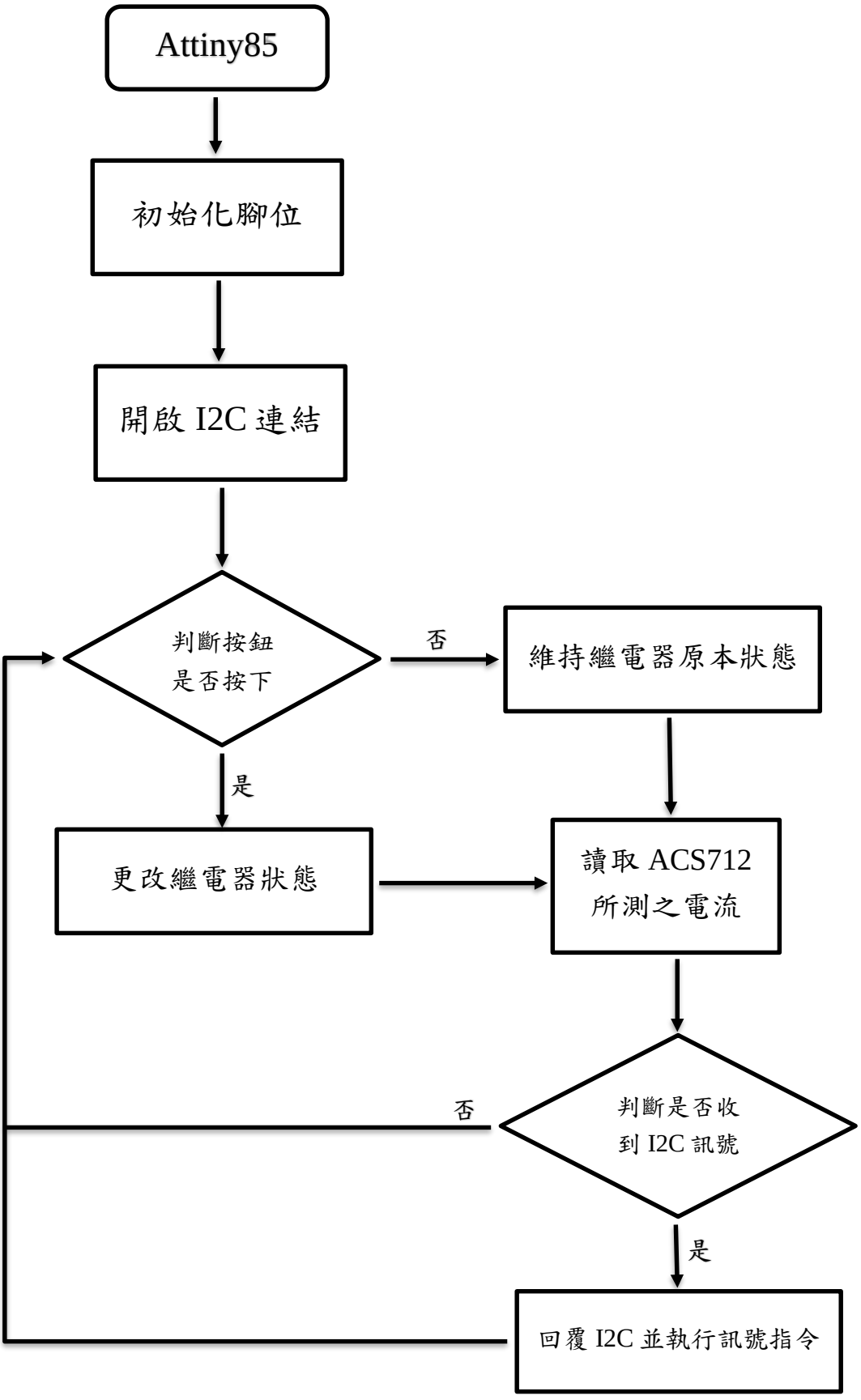

1. Node-RED
填表單
10. google雲端
連接網址
下載網頁資料
Wifi
2.ESP8266
TX2 RX2
TX RX
UART
資料輸出
UART
資料輸入
3.Level Shifter
RX TX
9.CH340
USB
PC
USB
USB
（偵錯用）
UART
資料輸入
UART
資料輸出
RX TX
4.ATmega32u4
USB
SDA
SCL(主端傳送頻率)
I2C
傳送&接收
資料
I2C
傳送頻率
SDA
SCL(從端接收頻率)
IO1
5.Attiny85
ADC3
IO3
數位
訊號控制
6. 繼電器
串聯(開關)
電
器
類比
訊號讀取
7.ACS712
串聯
(電流表)
數位訊號讀取
8. 按鈕

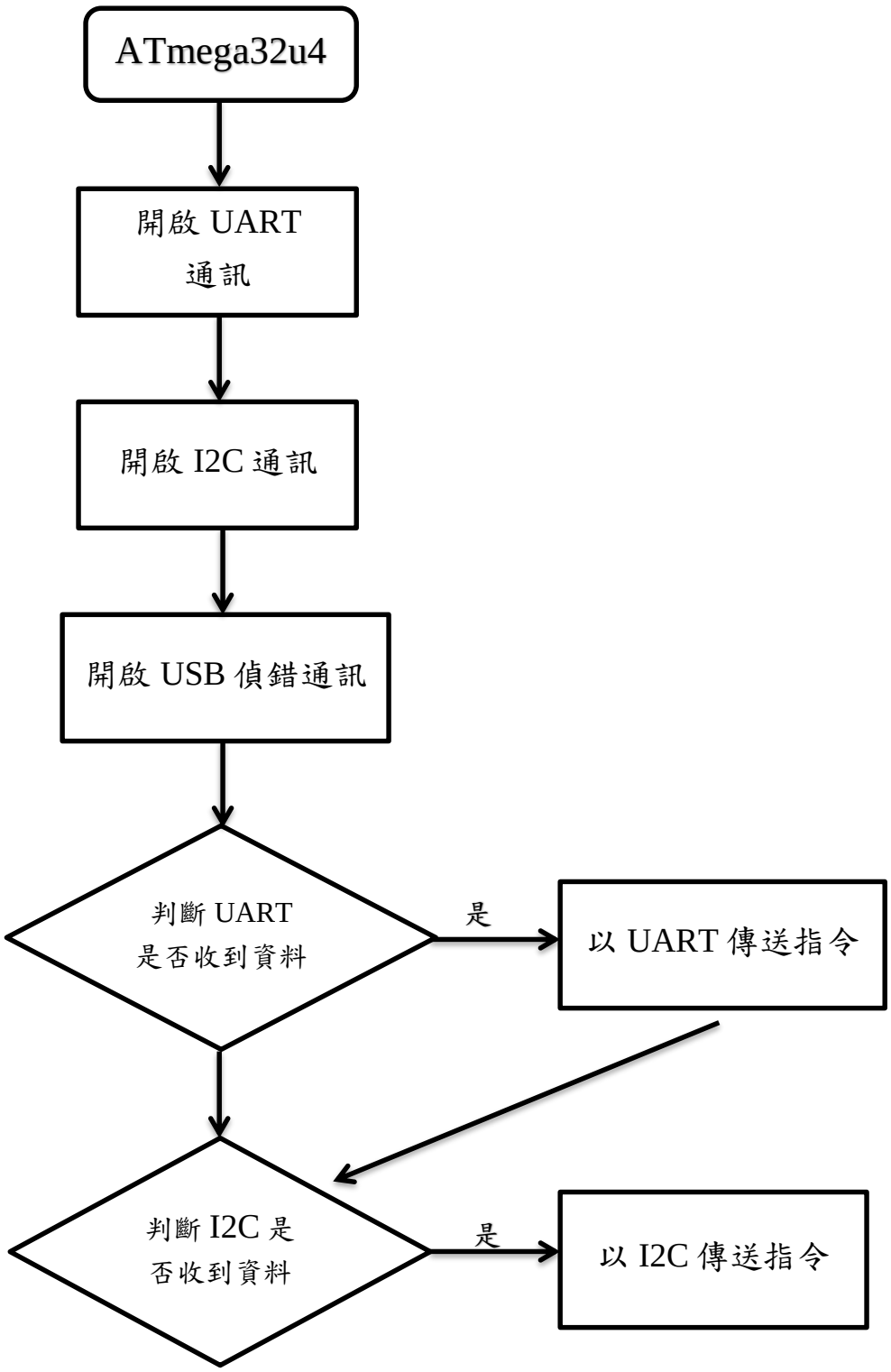
五、 系統規劃


使用 LINE APP,輸入 LINE BOT 指令
LINE BOT 回覆使用者訊息
line
使用者開啟
GOOGLE 圖表
查看資料
透過 Line 的
Messaging API
做資料交換
資料上傳
google 雲端
Node-RED
Messaging API
WIFI 通訊
網際網路通訊
WIFI 通訊
網際網路通訊
ESP8266
UART 通訊
UART 通訊
ATmega32u4
I2C 通訊
I2C 通訊
數位訊號輸入控制
按鈕
數位訊號
輸出控制
類比資料輸入
ACS712
Attiny85

六、 程式執行流程圖


ESP8266
連結 wifi
失敗
判斷 wifi 是
否連結成功
成功
連接網路伺服器
上傳和下載資料
UART 傳送
開的指令


Attiny85
初始化腳位
開啟 I2C 連結
判斷按鈕
是否按下
否
維持繼電器原本狀態
是
更改繼電器狀態
讀取 ACS712
所測之電流
判斷是否收
到 I2C 訊號
否
是
回覆 I2C 並執行訊號指令


ATmega32u4
開啟 UART
通訊
開啟 I2C 通訊
開啟 USB 偵錯通訊
判斷 UART
是否收到資料
是
以 UART 傳送指令
判斷 I2C 是
否收到資料
是
以 I2C 傳送指令

七、 系統子區塊介紹

（一） ESP8266

ESP8266 是一款功耗很低的 UART-Wi-Fi 透傳模組，是專門為移動裝置跟物聯網應用設計，可以把用戶的物理裝置連線至 Wi-Fi 無線網路上，進行互聯網或區域網路通訊，實現聯網的功能。可以廣泛應用於智能電網、智能交通、智能家具、手持裝置和工業控制等網域。如圖 1

圖 1

ESP8266 單晶片寫入韌體程式後，可透過 Client 傳入指令，根據不同指令做不同的對應處理(分別對應到 **下載 Node-Red 資料、解碼後把資料傳給 Attiny85**)

```
for (int i = 0; i < EspDataLength; i++)//下載 Node-Red 資料
  {
    message[i] = 0xFF;
  }
  if (web.httpRequest((String)"host:" + host, (String)"GET " +
(String)"/callback?command=RQ" + (String)" HTTP/1.1"))
  {
    Serial.printf("[HTTPS] connect http success\n");
    Serial.flush();
  }
  else
  {
    Serial.printf("[HTTPS] connect http fail");
    Serial.flush();
  }
```

```
    int strcom2int(String s)//將關鍵字轉換成關鍵字索引值
    {
      int returnmun = 32768;
      for (int i = 0; i <= index_max_now; i++)
      {
        if (s.indexOf(kwyword[i]) != -1)
        {
          returnmun = return_number[i];
          break;
        }
        else
        {
          returnmun = -1;
        }
      }
      return returnmun;
    }
```

（二） Attiny85

Attiny85 為一款微型控制器，它的體積小適合做小尺寸專案，一般 Arduino 板用的是 Atmel(ATMEGA328P)，而這個 Attiny85 是縮小版的，有 8 隻腳，具 8K flash 對於不需要太多輸出(5 個 IO 以內)與邏輯的小專案，不需要用 Arduino UNO 或 Nano，改用 Attiny85 就可以了。如圖 2

Attiny85

圖 2

Attiny85 透過 I2C 接收 ESP8266 的指令或按鈕控制繼電器(分別對應到 **按鈕、繼電器、ACS712**)

```
bool ReadButten()
  {
    if(butten_down)
    {
      if(digitalRead(buttenpin) == ON)//按鈕按著不改變狀態
      {
        butten_down = true;
        return false;
      }
      else
      {
        butten_down = false;
        delay(5);
        return false;
      }
    }
    else if(digitalRead(buttenpin) == ON)//按鈕按下就改變按鈕狀態並回傳
    {
      butten_down = true;
      ctrl_relay = !ctrl_relay;
      delay(5);
      return true;
    }
    else//未按下按鈕
    {
      return false;
    }
  }
```

```
void ctrlr(bool _ctrl)//設定繼電器應該要的狀態
  {if(_ctrl)
    {ctrl_relay = ON; }
    else
    {ctrl_relay = OFF;}
    relay( _ctrl);
  }
  void relay(bool ctrl)//控制繼電器
  {
    if(ctrl)
    {
      digitalWrite(relaypin,ON);
    }
    else
    {   digitalWrite(relaypin,OFF);    }
  }
  bool Now()//讀出繼電器應該要的狀態
  {
    if(ctrl_relay == ON)
    {
      return true;
    }
    else
    {
      return false;
    }
  }
```

```
int calibrate()//重置、校準
       {
              uint16_t acc = 0;
              for (int i = 0; i < 10; i++) {
                     acc += analogRead(pin);
              }
              zero = acc / 10;
              return zero;
       }
float getCurrentAC(uint16_t frequency) //讀出交流訊號
       {
              uint32_t period = 1000000 / frequency;
              uint32_t t_start = micros();
              uint32_t Isum = 0, measurements_count = 0;
              int32_t Inow;
              while (micros() - t_start < period)
              {
                     Inow = analogRead(pin) - zero;
                     Isum += Inow*Inow;
                     measurements_count++;
              }
              float Irms = sqrt(Isum / measurements_count) /
ADC_SCALE * VREF / sensitivity;
              return Irms;
      }
```

（三） ATmega32u4

ATmega32u4(如圖 3)是 Arduino Leonardo 開發板上的微控制器，跟 Arduino UNO 相比因為其有自己獨立的 USB 腳位(D+、D-)，故少了 ATmega16u2 或 Ch340，因而節省了成本和增加佈線空間(如圖 4)。

ATmega32u4

圖 3

ATmega32u4(分別對應到 **Attiny85、ESP8266**)

Arduino UNO

Arduino Leonardo

圖 4-比較圖

```
if (Serial1.available())//ESP8266 to attiny85
  {
    byte EspData[EspDataLength];
    byte Atmega32U4Data[Atmega32U4DataLength] = {0xFF, 0xFF, 0x00,
0x00, 0x00, 0x00};
    for (int i = 0; i < EspDataLength; i++)//讀 ESP8266 訊息
    {
      EspData[i] = Serial1.read();
      Serial.print("[ESP DATA "); Serial.print(i);
Serial.print("] :");
      Serial.println(EspData[i], HEX);
    }

    if ((EspData[0] != 0xFF) & (EspData[1] != 0xFF))//確認有指令
    {
      Atmega32U4Data[0] = 0x02;
      Serial.println("command correct");
      for (byte i = 0; i < Maxdrive; i++)//搜尋正確的地址
      {
        Serial.print("scan...");
        Serial.println(i);
        if (i == EspData[0]) //找到正確的地址
        {
```

```
          Serial.print("[ESP msg]");
          if (EspData[1] == 0x00) //執行指令
          {
            if (drive[i].Sstatus)
            {
              Serial.print(drive[i].addr); Serial.println(" :
off");
              drive[i].Wstatus = false;
            }
            Atmega32U4Data[0] = 0x00;//回覆指令
          }
          else if(EspData[1] == 0x01)//執行指令
          {
            if (!drive[i].Sstatus)
            {
              Serial.print(drive[i].addr); Serial.print(" : on");
              drive[i].Wstatus = true;
            }
            Atmega32U4Data[0] = 0x01;//回覆指令
          }
          else if(EspData[1] == 0x03)//執行指令
          {
            //Atmega32U4Data = {0x03, 0xFF, 0x00, 0x00, 0x00,
0x00};//回復指令
            //                回覆代號 索引值 由單精度浮點數換成 4 個 byte
          }
          Serial.println("");
          Atmega32U4Data[1] = (byte)i;//回覆索引值
          break;
        }
      }
      Serial1.write(Atmega32U4Data, Atmega32U4DataLength);//送出回覆
資料
    }
  }
```

八、 程式流程實驗

實驗 1: 手機端、Node-Red、ESP8266 之連結

實驗方法:在手機端輸入指令(編號+開/關),檢查 Node-Red 是否回覆正確與檢查 ESP8266 是否讀取 Node-Red 內部程式並把指令拆成編號+狀態傳送給 ATmega32u4。

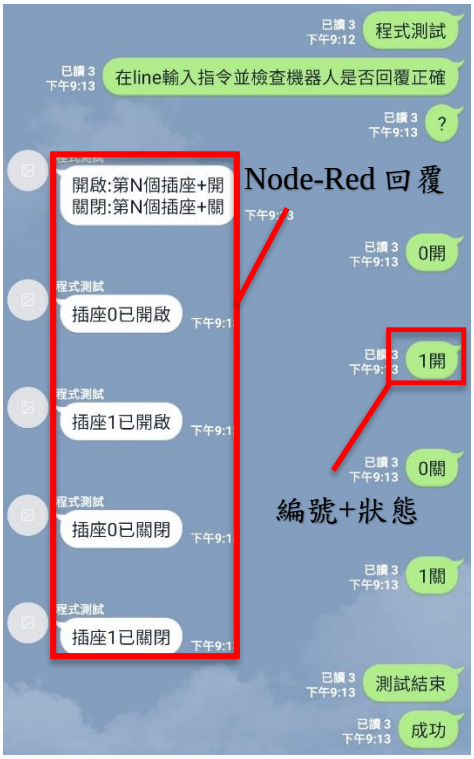


圖 1 手機端

```
var command = msg.req.query.command;

var text = context.global.text;
var re_text = "";
var ivaule = "";
if(command == "RQ")
{
    if (text.indexOf("開")> -1)
    {
        re_text = "on";
    }
    else if(text.indexOf("關")> -1)
    {
        re_text = "off";
    }

    for(var i = 0; i<text.length ; i++)
    {
        while((text.charAt(i) == "0")||
        (text.charAt(i) == "1")||
        (text.charAt(i) == "2")||
        (text.charAt(i) == "3")||
        (text.charAt(i) == "4")||
        (text.charAt(i) == "5")||
        (text.charAt(i) == "6")||
        (text.charAt(i) == "7")||
        (text.charAt(i) == "8")||
        (text.charAt(i) == "9"))
        {
        ivaule = String(ivaule) + String(text.charAt(i));
        i++;
        }
    }
    msg.payload = String("{") + String(re_text)
    + String(ivaule) + String("}") + String("<br>");
}
else if(command == "RB" && text=="")
{
    msg.payload = String("{null}<br>");
}
else
{
    msg.payload = String("{null}") + String("<br>");
}
return msg;
```

圖 2 Node-Red 程式碼

圖 3 取出 Node-Red 部份程式

```
msg.payload = String("{") + String(re_text)
+ String(ivaule) + String("}") + String("<br>");
```

圖 4 ESP8266 讀取 Node-Red 部份程式

讀取

```
String GetMessage()//讀取HTTP伺服器下載的資料
{
  String liner;
  liner = client.readStringUntil('<');
  String line;
  for(int i = liner.indexOf("{"); i<= liner.indexOf("}"); i++)
  {
    line += liner.charAt(i);
  }
  return line;
}
```

拆成編號

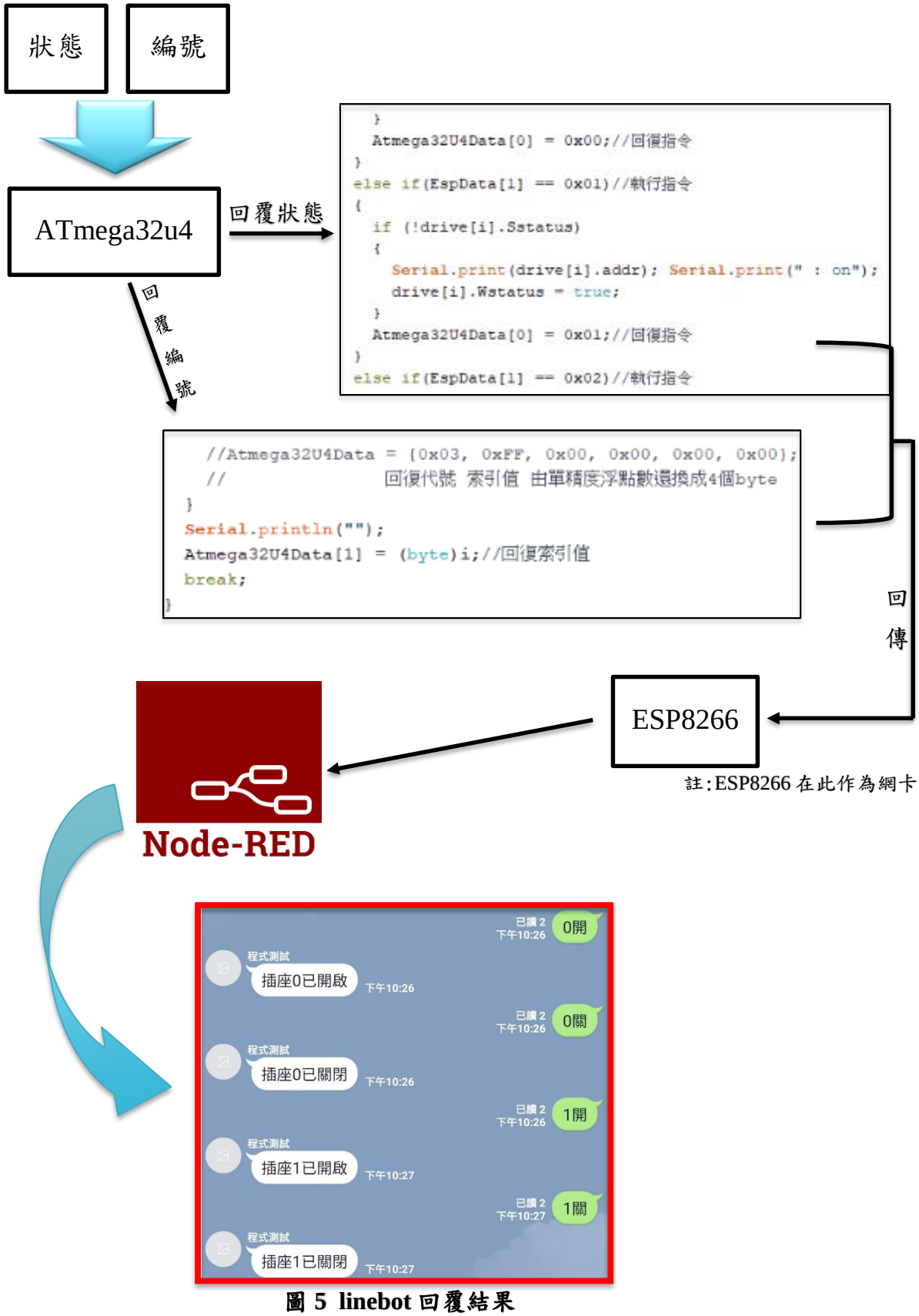
```
byte read_int_form_string(String s, String begin_char, String end_char)
{
  byte rtdata = 0xFF;
  bool star = false;
  bool ending = false;
  for (byte i = s.indexOf(begin_char); i < s.length() & (i != s.indexOf(end_char)); i++)
  {
    if (star & ending)
    { break;  }

    if (!ending & star )
    { ending = true;  }

    for (int i2 = 0; i2 < (sizeof(integer) / sizeof(char)); i2++)
    {
      if (!star)
      {
        if (s.charAt(i) == integer[i2])
        {
          star = true;
          rtdata = 0;
        }
      }

      if ((s.charAt(i) == integer[i2]) & star)
      {
        rtdata *= 10;
        rtdata += i2;
        break;
      }
      else if (star)
      {
        if (s.charAt(i) == integer[i2])
        { ending = false; }
      }
    }
    if (ending)
    { return rtdata;  }
  }
```

拆成狀態

```
int strcom2int(String s)//將關鍵字轉換成關鍵字索引值
{
  int returnmun = 32768;
  for (int i = 0; i <= index_max_now; i++)
  {
    if (s.indexOf(kwyword[i]) != -1)
    {
      returnmun = return_number[i];
      break;
    }
    else
    {
      returnmun = -1;
    }
```

說明	狀態	插座編號
關	00(H)	00(H)~ 09(H)
開	01(H)	00(H)~ 09(H)
查詢	02(H)	00(H)~ 09(H)

ESP8266 把從 Node-Red 讀取之程式碼拆解成狀態+編號後，再把指令傳送給 ATmega32u4，其會執行指令，再回傳狀態跟編號給 ESP8266，其再傳送給 Node-Red，最終 Node-Red 會把執行結果透過 line 傳給使用者。

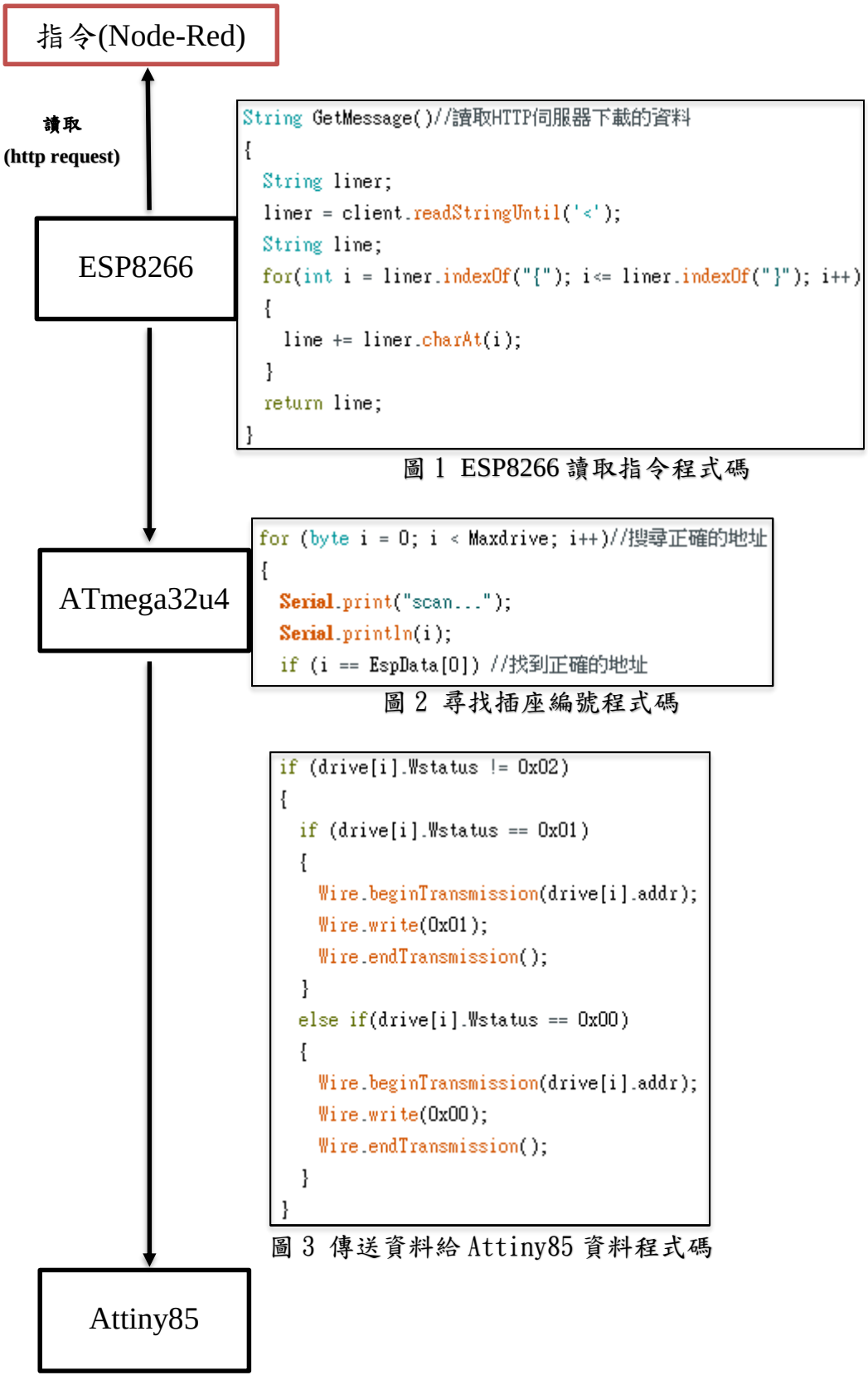


註:ESP8266 在此作為網卡

圖 5 linebot 回覆結果

實驗 2:電器用品、繼電器、按鈕、ACS712、Attiny85、ATmega32u4 之連結

實驗方法:ESP8266 收到使用者在 line 輸入的指令後會傳送給 ATmega32u4，其再找到對應之插座編號並傳送給 Attiny85。




```
String GetMessage()//讀取HTTP伺服器下載的資料
{
  String liner;
  liner = client.readStringUntil('<');
  String line;
  for(int i = liner.indexOf("{"); i<= liner.indexOf("}"); i++)
  {
    line += liner.charAt(i);
  }
  return line;
}
```

圖 1 ESP8266 讀取指令程式碼

```
for (byte i = 0; i < Maxdrive; i++)//搜尋正確的地址
{
  Serial.print("scan...");
  Serial.println(i);
  if (i == EspData[0]) //找到正確的地址
```

圖 2 尋找插座編號程式碼

```
if (drive[i].Wstatus != 0x02)
{
  if (drive[i].Wstatus == 0x01)
  {
    Wire.beginTransmission(drive[i].addr);
    Wire.write(0x01);
    Wire.endTransmission();
  }
  else if(drive[i].Wstatus == 0x00)
  {
    Wire.beginTransmission(drive[i].addr);
    Wire.write(0x00);
    Wire.endTransmission();
  }
}
```

圖 3 傳送資料給 Attiny85 資料程式碼

Attiny85 就會用使用者要插座執行的狀態(開/關)來控制繼電器以達到控制電器用品的電源。

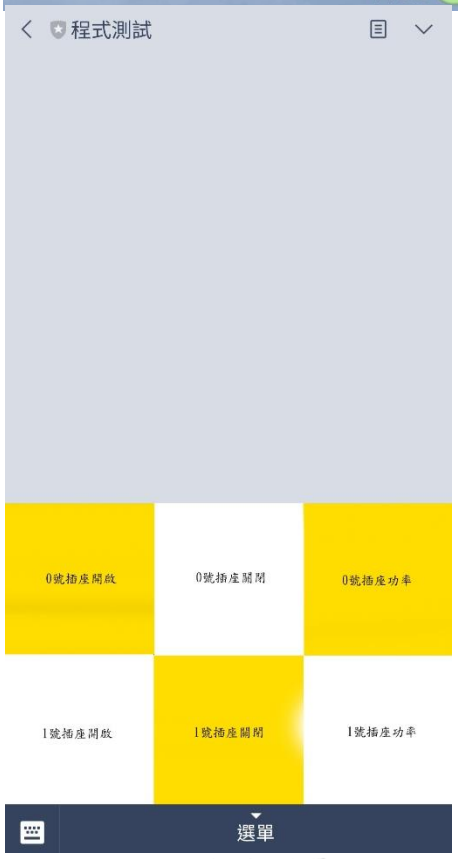
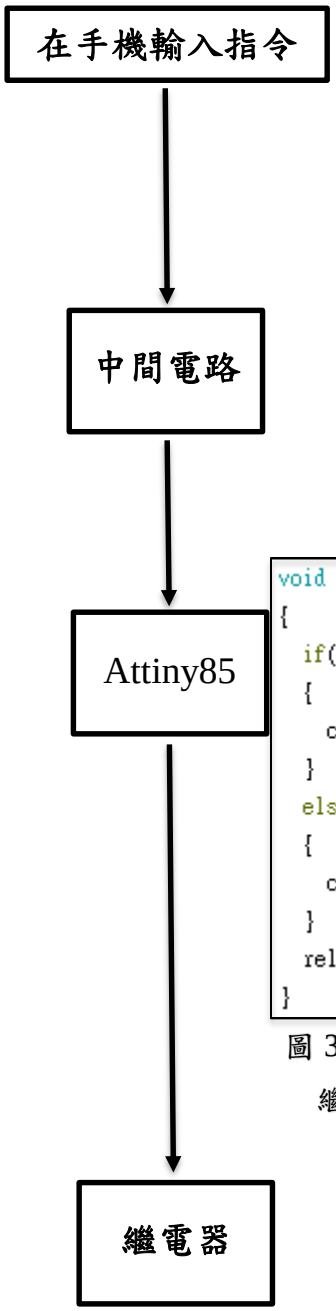


圖 2 指令選單




```
void ctrlr(bool _ctrl)
{
  if(_ctrl)
  {
    ctrl_relay = ON;
  }
  else
  {
    ctrl_relay = OFF;
  }
  relay( _ctrl);
}
```

圖 3 Attiny85 控制繼電器程式碼

在 LINE 輸入完指令後，所產生的結果。

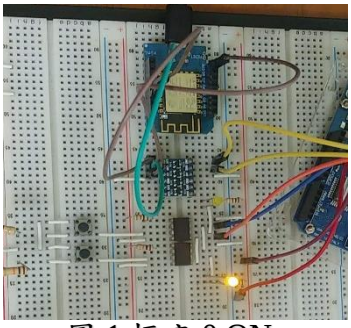


圖 1 插座 0 ON

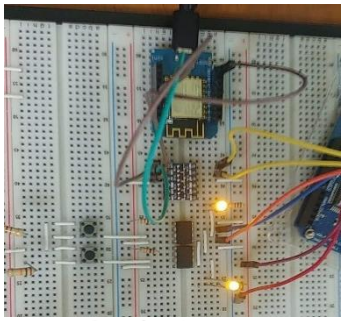


圖 2 插座 1 ON

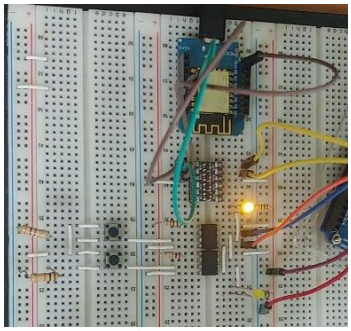


圖 3 插座 0 OFF

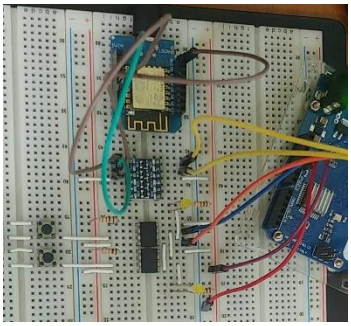


圖 4 插座 1 OFF

註:此 4 張為插座開關工作信號

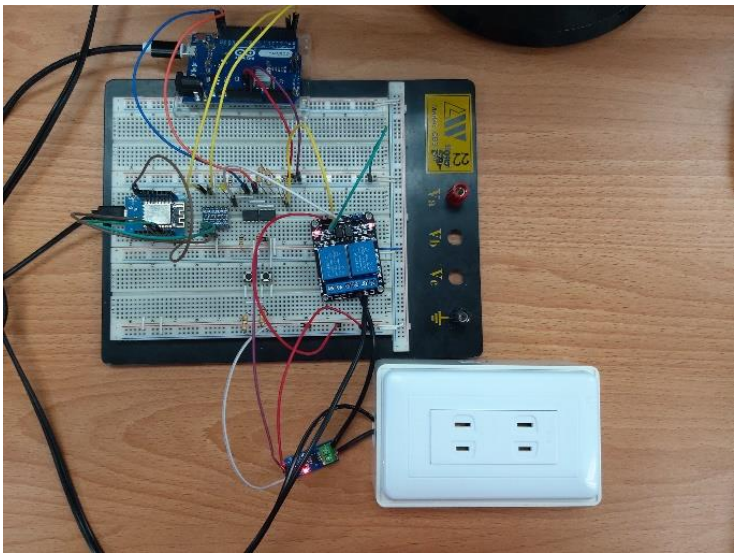
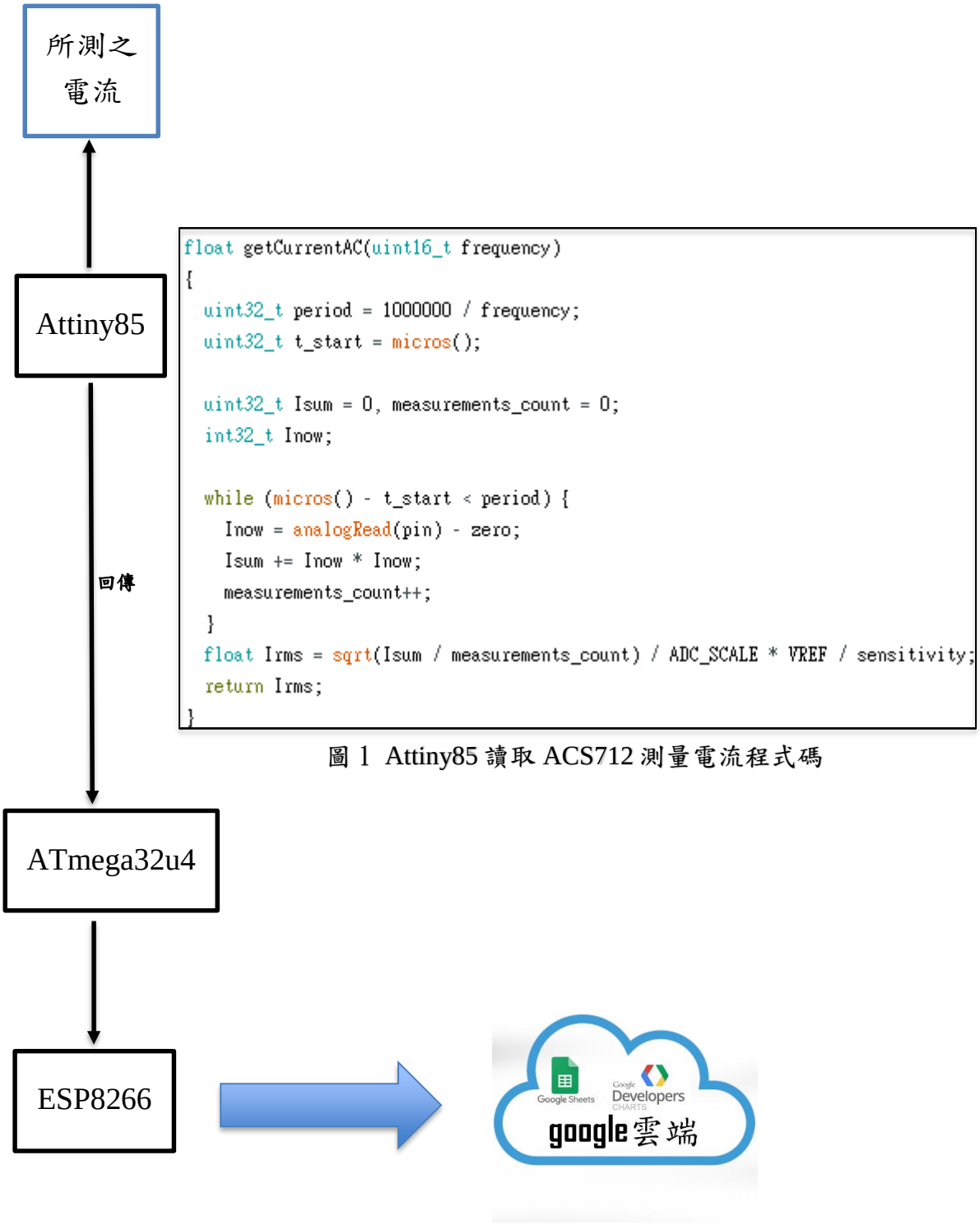


圖 5 完整電路及插座

ACS712 測量電器用品所使用之電流，Attiny85 會去讀取 ACS712 所測量的電流值並回傳給 ATmega32u4 再透過 ESP8266 傳送至雲端。




```
float getCurrentAC(uint16_t frequency)
{
  uint32_t period = 1000000 / frequency;
  uint32_t t_start = micros();

  uint32_t Isum = 0, measurements_count = 0;
  int32_t Inow;

  while (micros() - t_start < period) {
    Inow = analogRead(pin) - zero;
    Isum += Inow * Inow;
    measurements_count++;
  }
  float Irms = sqrt(Isum / measurements_count) / ADC_SCALE * VREF / sensitivity;
  return Irms;
}
```

圖 1 Attiny85 讀取 ACS712 測量電流程式碼

實驗 3:手機端、Google 試算表、圖表之連動

實驗方法:在手機端輸入指令查看 linebot 是否給出圖表及功率表連結。

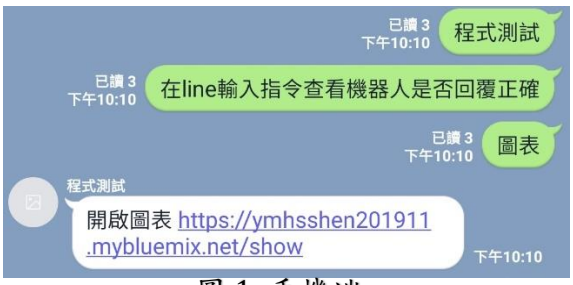


圖 1 手機端

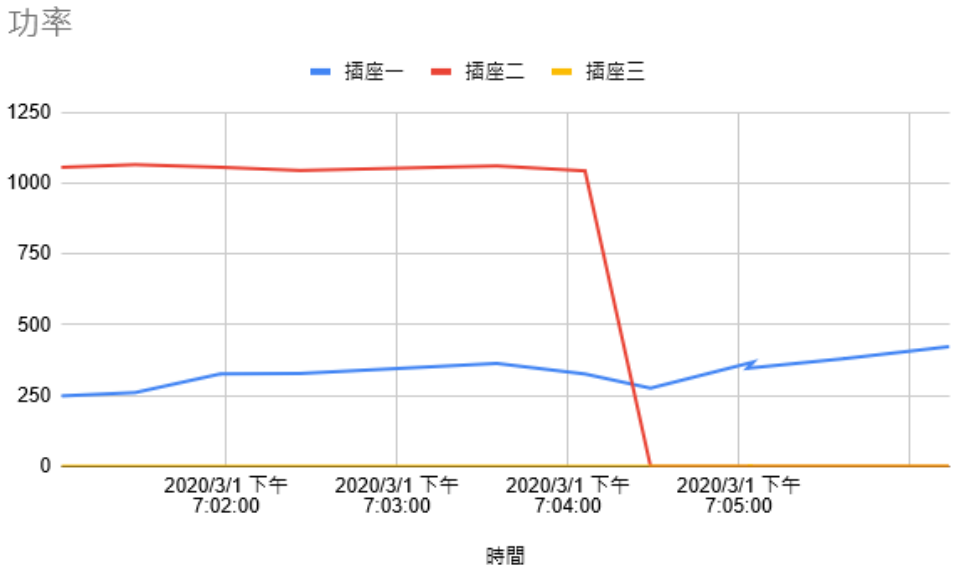


圖 2 圖表

時間戳記	插座一	插座二	插座三
	電腦	吹風機	沒有
2020/3/1 下午 7:01:02	248.9	1056.2	0
2020/3/1 下午 7:01:28	260.8	1065.7	0
2020/3/1 下午 7:01:58	326.9	1056.4	0
2020/3/1 下午 7:02:26	328.3	1045.2	0
2020/3/1 下午 7:03:35	363.8	1061.2	0
2020/3/1 下午 7:04:06	326.5	1043.8	0
2020/3/1 下午 7:04:29	276.5	0	0
2020/3/1 下午 7:05:06	368.4	0	0
2020/3/1 下午 7:05:03	346.7	0	0
2020/3/1 下午 7:05:37	380.7	0	0
2020/3/1 下午 7:06:15	423.2	0	0

圖 3 表格

伍、 研究結果

本專題成功利用了 linebot 來遠端控制電器，亦即達成物聯網之目標，此外亦挑選符合成本考量與機能適切之材料。使用模組化的概念來完成本專題，並可直接透過使用者手機查看圖表及功率。

陸、 討論

一、 在寫程式時有沒有遇到困難，是何種類型之困難，提出什麼解決方案?

答:困難點在於 Attiny85 與 ESP8266 直接用 I2C 通訊會發生問題(ESP8266 當 I2C 的主端訊號會打不出去)，解決方法為使用 ATmega32u4 取代 ESP8266 管理 Attiny85 的功能，也取代了暫存電流、繼電器狀態等資訊。

二、 在實驗中遇到何種類型之問題，因應之解決辦法為何?

答:連結 Node-Red 跟 Google 試算表時遇到困難，發現 Node-Red 無法使用 Google API 直接寫入試算表，經過討論與試驗後，本專題決定以填表單之方式把電器的電流值填上試算表，再使用函式算出功率。

三、 為什麼要建立 LineBot 圖文選單?

答:因為每次執行指令前都要輸入文字覺得很繁瑣，最後選擇製作動態 LINE 圖文選單來提升控制之便捷性。

四、 為何使用 I2C 連結?

答:因為本專題考慮到未來的擴充性，故使用 I2C 的連接方式。

柒、 結論

本專題研究利用 Node-Red 為主軸連結各電器與 LINE 之間的信號傳遞，包括 LINE、ACS712、ESP8266、雲端，使用者除了可以直接在 LINE 通訊軟體上輸入指令遠端操控電器電源，更能利用 LINE 來查詢各項電器在不同時間的消耗功率，藉此達到統計與分析家中各項電器的用電情況，以利日後的使用調整。本專題開始就朝明確方向努力並逐一完成各項目標，最後再將各部分相互統整連結出一完整成品，期望在現今這種科技化時代下，使用者能使用智慧型手機傳達開關電器之指令，並在電流超過插座額定值時，會直接自動關閉電源並透過通訊軟體向使用者發送警訊，藉以達到省電環保、避免浪費與安全考量的目標。

捌、 參考資料及其他

Attiny85

https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json

Attiny85-I2C

https://github.com/lucullusTheOnly/TinyWire

wire

https://blog.xuite.net/m0923678421/development/575407915

https://blog.xuite.net/m0923678421/development/571335811

https://blog.hoyo.idv.tw/?p=4591

ACS712 資料庫

https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf

ESP8266

https://swf.com.tw/?p=1260

float 2 byte

https://forum.arduino.cc/index.php?topic=43222.0