

銘傳大學

電子工程學系
專題研究總審報告

智能導覽車
Intelligent guide car

指導教授：張世軍 老師

專研學生：陳彥邦 徐琦喻 劉懿葭

中華民國一〇九年十一月二十日

本校一〇九學年度 電子工程學系

中華民國 109 年 11 月 20 日

摘要

智慧型運輸系統可說是人們對於「行」的基本需求，而智能車是其中一個重要環節。當我們初次來到一個陌生環境，對於周遭的一切一無所知，可以透過智能導覽的設計在駕駛人行經某些特定建築時，用語音提供說明與介紹，讓駕駛人不需額外查詢資訊。

各式的智能安全系統與技術發展可有效協助駕駛者駕車；除了以被動防護來提供車輛的安全性外，我們更強調主動安全。防撞系統及倒車輔助系統中，車輛利用配置在車上的感測裝置所提供的資料運算，在當障礙物與車輛的距離低於警示範圍時，使用顯示器或蜂鳴器發出警示通知，提醒駕駛人注意；指紋辨識應用於智能導覽車的感應上，讓裝置辨別是否為已註冊的有效指紋，使駕駛人的指紋辨識通過時解鎖其他的功能（例如：溫溼度感測、倒車雷達等）；溫度濕度感測能讓駕駛人隨時注意周遭溫溼度的變化；影音導覽可輔助駕駛人到某景點時，可以透過選擇檔案撥放影片，影片就會詳細的去介紹景點內容。

關鍵詞：智能導覽車、Arduino、Arduino UNO 板、LabVIEW、防撞功能、倒車雷達輔助、指紋辨識、影音導覽、溫度濕度偵測。

Abstract

Smart cars based on the intelligent transportation system come to be the basic need for driving and travelling. When we first come to an unfamiliar city or a place, real-time information allows us to explore the city easily. Intelligent safety system protects drivers from injury and avoids vehicles collision. Sensing the distance between the obstacles and the vehicle warns drivers by a display or a buzzer to prevent accidents in various safety-critical situations. Fingerprint recognition is applied to identify whether it is a registered valid fingerprint, thereby improving the vehicle security. In addition, the temperature and humidity sensor allows the driver to pay attention to changes in the surrounding temperature and humidity at any time. That makes the driver comfort when driving. Finally, the audio guide function to assist the driver understand about local information when he arrives at a certain scenic spot. Driver can play the video by selecting the file, and the video will introduce the content of the scenic spot in detail.

Keywords: Intelligent guide car, Arduino, Arduino UNO , LabVIEW, Collision avoidance function, Reversing radar assist , Fingerprint identify, Audiovisual guided tour, Temperature & Humidity.

目錄

摘要	I
Abstract	II
第一章 簡介	1
1.1 研究動機與目的 -----	1
1.2 文獻探討 -----	2
1.3 專題架構 -----	3
1.4 章節簡介 -----	4
第二章 材料介紹	6
2.1 硬體元件介紹 -----	6
2.2 軟體介紹 -----	19
第三章 智能導覽車設計與操作	23
3.1 溫度濕度 (DHT11) -----	23
3.2 倒車雷達 -----	24
3.3 防撞功能 -----	26
3.4 指紋辨識解鎖 -----	27
3.5 連結 LabVIEW 的 Arduino 統合程式 -----	28
3.6 LabVIEW 設計 -----	30
第四章 成果與未來展望	46
4.1 成果外觀展示 -----	46
4.2 未來展望 -----	51
工作分配	53
專題進度	56

附錄	57
7.1 材料規格-----	57
7.2 Arduino 軟體下載-----	60
7.3 LabVIEW 軟體下載-----	64
7.4 Arduino 程式設計-----	67
7.5 LabVIEW 程式設計-----	94
參考文獻	100

表目錄

表 2-1	DHT11 與 DS18B20 的差異 -----	7
表 2-2	DHT11 偵測範圍 -----	7
表 2-3	DHT11 腳位說明 -----	8
表 2-4	HC-SR04 實測數據 -----	15
表 2-5	三者辨識的比較 -----	17
表 2-6	FPM10A 的腳位說明 -----	18
表 5-1	工作分配 -----	53
表 6-1	專題進度表 -----	56
表 7-1	Arduino Uno R3 規格 -----	57
表 7-2	HC-SR04 規格 -----	58
表 7-3	FPM10A 光學指紋辨識規格 -----	59

圖目錄

圖 1-1 專題流程圖	4
圖 2-1 DHT11	7
圖 2-2 Arduino Uno 開發版正面	9
圖 2-3 Arduino Uno 開發板背面	9
圖 2-4 聲波分佈	12
圖 2-5 HC-SR04	12
圖 2-6 HC-SR04 元件測距俯瞰圖	13
圖 2-7 硬體接線示意圖	13
圖 2-8 序列埠監控視窗	14
圖 2-9 距離與時間關係圖	16
圖 2-10 FPM10A 光學指紋辨識	17
圖 2-11 有源蜂鳴器	19
圖 2-12 Arduino 模型車	19
圖 2-13 程式方塊圖	22
圖 2-14 使用者面板(儀器面板)	22
圖 2-15 LabVIEW 簡介	22
圖 3-1 DHT11 電路設計	24

圖 3-2	硬體接線示意圖 -----	25
圖 3-3	硬體接線示意圖 -----	26
圖 3-4	指紋辨識接法 -----	27
圖 3-5	單項(溫溼度、防撞)程式區域分布 -----	29
圖 3-6	統合程式區域分布 -----	29
圖 3-7	LabVIEW 小技巧 1 -----	31
圖 3-8	LabVIEW 小技巧 2-----	31
圖 3-9	VISA configure serial port -----	32
圖 3-10	VISA 元件 -----	33
圖 3-11	架構-----	33
圖 3-12	時間延遲-----	34
圖 3-13	Match Pattern-----	35
圖 3-14	Fract/Exp String to number-----	35
圖 3-15	VISA Property Node-----	36
圖 3-16	VISA Property Node 四種模式 -----	36
圖 3-17	更改模式操作 1-----	37
圖 3-18	更改模式操作 2-----	37
圖 3-19	Read JPEG File.vi-----	38
圖 3-20	Draw Flattened Pixmap.vi -----	38

圖 3-21	Automation Open -----	39
圖 3-22	Path To String -----	39
圖 3-23	Greater Than 0? -----	40
圖 3-24	控制開關-----	41
圖 3-25	數據顯示-----	41
圖 3-26	VISA Resource -----	42
圖 3-27	Windows Media Player -----	42
圖 4-1	Arduino 統合程式成果部分截圖 -----	47
圖 4-2	LabVIEW 成果-----	47
圖 4-3	實體成果 -----	48
圖 4-4	Arduino IDE 序列埠監控執行成果-----	50
圖 4-5	LabVIEW 使用者介面成果-----	50
圖 7-1	Arduino 軟體下載-1-----	60
圖 7-2	Arduino 軟體下載-2-----	60
圖 7-3	Arduino 軟體下載-3-----	61
圖 7-4	Arduino 軟體下載-4-----	61
圖 7-5	Arduino 軟體下載-5-----	61
圖 7-6	Arduino 軟體下載-6-----	62
圖 7-7	Arduino 程式編輯視窗 -----	62

圖 7-8	Arduino 程式編輯視窗功能列	63
圖 7-9	Arduino 初步設定 1	63
圖 7-10	Arduino 初步設定 2	64
圖 7-11	LabVIEW 軟體下載-1	64
圖 7-12	LabVIEW 軟體下載-2	65
圖 7-13	LabVIEW 軟體下載-3	65
圖 7-14	LabVIEW 軟體下載-4	65
圖 7-15	LabVIEW 軟體下載-5.1	66
圖 7-16	LabVIEW 軟體下載-5.2	66
圖 7-17	LabVIEW 軟體下載-5.3	66
圖 7-18	LabVIEW 軟體下載-6	67
圖 7-19	DHT11 程式步驟 1	67
圖 7-20	DHT11 程式步驟 2	68
圖 7-21	DHT11 程式步驟 3-1	68
圖 7-22	DHT11 程式步驟 3-2	68
圖 7-23	DHT11 程式步驟 4	69
圖 7-24	DHT11 測試結果	69
圖 7-25	FPM10A 程式步驟 1	76
圖 7-26	FPM10A 程式步驟 2	77

圖 7-27	FPM10A 程式步驟 4 -----	77
圖 7-28	FPM10A 測試結果-----	78
圖 7-29	整體程式外觀 (Ture)-----	94
圖 7-30	整體程式外觀(False)-----	94
圖 7-31	溫溼度感測(Ture) -----	95
圖 7-32	溫溼度感測(False)-----	95
圖 7-33	倒車雷達(Ture) -----	96
圖 7-34	倒車雷達(False)-----	96
圖 7-35	指紋辨識(Ture) -----	97
圖 7-36	指紋辨識(False)-----	97
圖 7-37	影音導覽(Ture) -----	98
圖 7-38	影音導覽(False)-----	98
圖 7-39	防撞四角(Ture) -----	99
圖 7-40	防撞四角(False)-----	99

第一章 簡介

1.1 研究動機與目的

當一個人來到一個陌生的地方，附近又沒有任何人有空幫助的時候，這時候就只能靠自己很沒效率的去探索這個新地方。為了更加有效率去認識這個新地方，因此我們決定開發一個能夠靠自己就能簡單探索新地方的工具。

首先，我們聯想到的就是探索一個地方，一定要有個能夠快速移動的工具，因此我們決定使用車子來當主體。再來為了能加快了解這新地方，我們打算架設一些小工具在車子內部，當你想了解某一個地方資訊，使用這些工具就能得知當地的環境資訊、建築資訊、濕度、溫度。最後除了快速之外，還有一點也是非常重要的，也就是安全問題，畢竟車子就像水火一樣的無情，一個控制不當就會導致意外發生，所以我們需要在車體外做些安全的功能，功能有防撞功能，讓車子快撞到物體時，會發出警訊聲，可以避免發生交通事故時。

1.2 文獻探討

智慧汽車集合了環境感測、決策規劃、多等級輔助駕駛等多功能的結合系統，運用的技術包括電腦、感測、信息綜合、人工智慧與自動控制，讓我們希望能夠打造出一輛智能導覽車[1]。智慧汽車強調「智慧」，最大的差別，可分為兩大部份：第一是汽車車身和車載系統的電子化，第二是汽車引擎動力的電動化。主動安全、舒適便利以及節能，是智慧汽車技術架構的三大主軸，彼此功能相互整合，藉由車用網路連結。雷射掃描、影像辨識及影像感測器，成為智慧汽車主動安全系統的三大利器[2]。自駕車的感測器的使用分為兩種型態，以有無使用光達作為分界，光學雷達(LiDAR)採用單束窄帶激光二極體產生一波長為 905nm 的脈衝波，此波段屬於紅外線，對物體並沒有穿透性，絕大部分物體均可反射此脈衝波[3]。在車用光達常用的距離量測技術就是利用飛行時間 (Time Of Flight, TOF) 技術。即發出很短脈衝 (~10 ns, 10⁻⁸ 秒) 的雷射去照射目標，光達的角度解析度理論上是 77 GHz 毫米波的 4000 倍；我們使用了蜂鳴器量測距離，希望能夠達到防撞的目的[4]。車載系統在考量成本及穩定性等等面向上，多數製造商選擇在 Quick Unix (QNX) 平台上做開發，QNX 平台是嵌入式作業系統，具備體積小、運行速度快等優點，採用模組化設計，以便工程師可以根據車輛的硬體元件選擇載入的功能，而我們透過實驗室虛擬平台 Labview，與 Arduino 相互連結[5]。二維碼自

助影音導覽系統主要是把小景點的地理位置、區域特色、歷史文化、傳說故事等信息，以圖、文、特色音頻三者結合的形式收錄在二維碼標誌牌介紹中，使之成為景點的標誌「名片」。傳統的二維碼自助影音導覽系統已經不能滿足當今遊客的旅遊體驗。透過網際網路及物聯網等技術基礎之上的智能導覽系統，將景區導覽電子化；我們希望將此加入，達到影音導覽的功能[6]。指紋是生物辨識的方式之一，透過波峰與波谷以及隆起線的變化，就能抓到特定的特徵點；相較於鑰匙或密碼的安全性要高，可以用於智能車輛的門鎖[7]。指紋辨識器由軟體與硬體共同構成，硬體部分是指紋感應器，用於採集指紋，依照技術可分為電容式與光學式感應器，依照掃描方式可分成滑動式與按壓式。軟體則是指紋辨識演算法，當前端指紋感應器掃描完畢後，透過演算法比對資料庫內的指紋檔案[8]。

1.3 專題架構

一開始先是討論研究的方向，討論出要做出實體成果，後來就決定做智能導覽車。再來就是設計智能導覽車該有甚麼功能，功能包含了：「溫度濕度感測、環境導覽、指紋辨識解鎖、防撞警示、倒車雷達、影音導覽」，並逐一去測試單項的功能。單項功能都測試完後，就把所有功能都統合到模擬車上，並做最後的測試，以及探討優缺點和未來展望。

(架構如圖 1-1)

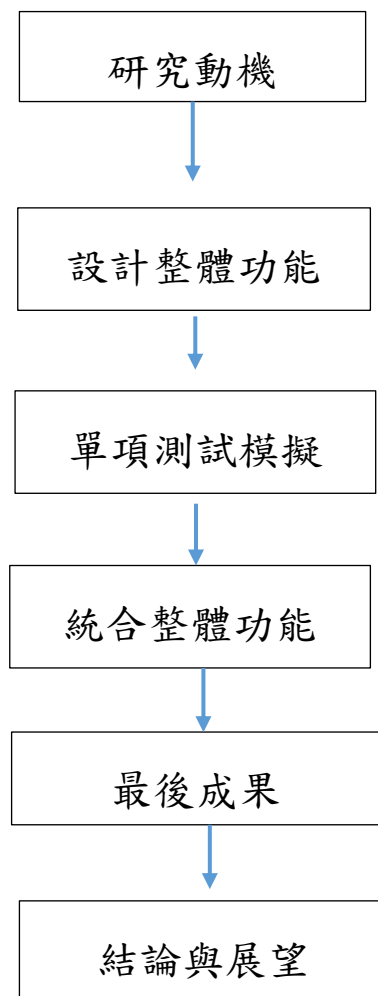


圖 1-1 專題流程圖

1.4 章節簡介

第一章 簡介：簡介就是概述整個專題的由來，包括研究動機和目的、文獻探討、專題流程圖。研究動機和目的是講述我們一開始為什麼會有想做這個研究的想法，以及探討最終的目的。文獻探討是講跟我們研究有關的相關文獻，並利用文獻裡面的知識來方便我們完成此研究。

專題流程圖是我們在做此研究時的依依循進的步驟。

第二章 材料介紹：材料介紹分為兩部分，分別是硬體的介紹和軟體的介紹。硬體介紹的部分包括：「DHT11 溫度濕度感測、Arduino UNO R3 開發板、HC-SR04 超音波感測器、KY-012 有源蜂鳴器、指紋辨識、Arduino 模型車」。軟體介紹的部分包括：「Arduino、LabVIEW」。

第三章 智能導覽車設計與操作：第三章為整個研究的核心所在，講述著各項功能的設計步驟以及如何統合的步驟，是能讓讀者在甚麼都不會的情況下，順利的完成整個智能導覽車。

第四章 成果與未來展望：將最終成果呈現在此章節，並探討還有哪裡需要改進的地方，或在新增那些功能會更好。以及探討此項研究將來能發展在哪一個領域。

第二章 材料介紹

2.1 硬體元件介紹

2.1.1 溫度濕度感測：

DHT11(實物如圖 2-1) 採用單匯流排協議與微控制器通訊，單片機發送一次復位訊號後，DHT11 從低功耗模式轉換到高速模式，等待主機復位結束後，DHT11 傳送響應訊號，並拉高匯流排準備傳輸資料。一次完整的資料為 40bit，按照高位在前，低位在後的順序傳輸。

DHT11 只有在接收到開始訊號後才觸發一次溫溼度採集，如果沒有接收到主機發送復位訊號，DHT11 不主動進行溫溼度採集。當資料採集完畢且無開始訊號後，DHT11 自動切換到低速模式。(DHT11 偵測範圍如表 2-2) (DHT11 元件腳位如表 2-3)

相比於 DS18B20 只能測量溫度，DHT11 能檢測溫度也能檢測溼度，不過 DHT11 量測出的精準度以及可量測範圍都低於 DS18B20，其溫度測量範圍為 0~50°C，誤差在 $\pm 2^{\circ}\text{C}$ ；溼度的測量範圍 20%~90%RH (Relative Humidity 相對溼度—指空氣中水汽壓與飽和水汽壓的百分比)，誤差在 $\pm 5\%\text{RH}$ 。(兩者比較如表 2-1)

表 2-1 DHT11 與 DS18B20 的差異

	DHT11	DS18B20
測溫範圍	0°C ~ 50°C	-55°C ~ +125°C
測濕範圍	20~90%	無
溫度測量誤差	±2°C	±1°C
濕度測量誤差	±5%	無

表 2-2 DHT11 偵測範圍

型號	測試範圍	測濕精度	測溫精度
DHT11	濕度 20-90% 溫度 0-50°C	±5%RH	±2°C

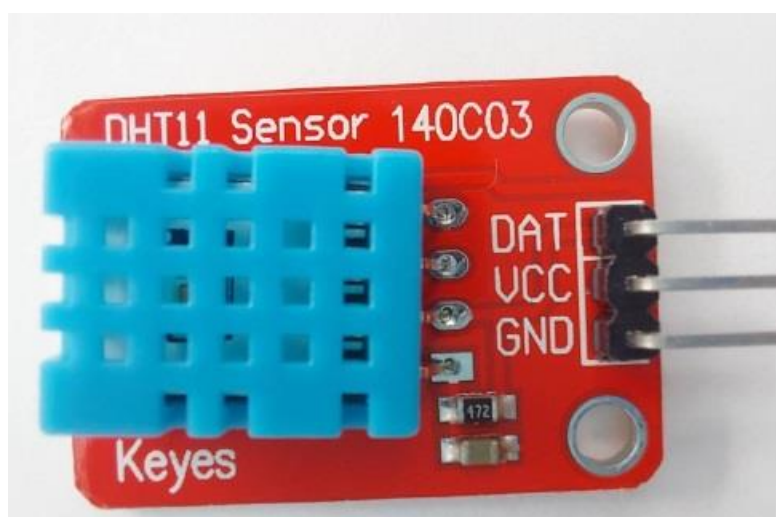


圖 2-1 DHT11

表 2-3 DHT11 腳位說明

Pin	名稱	說明
2	DAT	信號線
	VCC	3~5.5V
-	GND	接地線

2.1.2 Arduino UNO R3 開發板：

Arduino 電路板設計使用各種微處理器和控制器。Arduino Uno 是一塊以 ATmega328 為核心的微控制板，可以連接各種擴充板、麵包板和其他電子元件及感測器。它擁有 14 支數位 I/O 腳位(0、1、2、3、4、5、6、7、8、9、10、11、12、13)，而當中 6 個腳位(3、5、6、9、10、11)可以用 analogWrite() 函數當作 8-bit PWM 輸出；6 個類比輸入端(A0、A1、A2、A3、A4、A5)。各個腳位能提供或接收最大值 40mA 的電流。由於開發板具有燒錄在微處理器內部的 Bootloader(開機啟動程式)，因此能夠透過 USB(USB 線為 A 型轉 B 型)直接上傳程式至開發板，不需經過其他外部燒錄設備。供電部份可選擇由 USB 直接提供電源，或者使用 AC-to-DC Adapter 及電池作為外部供電。

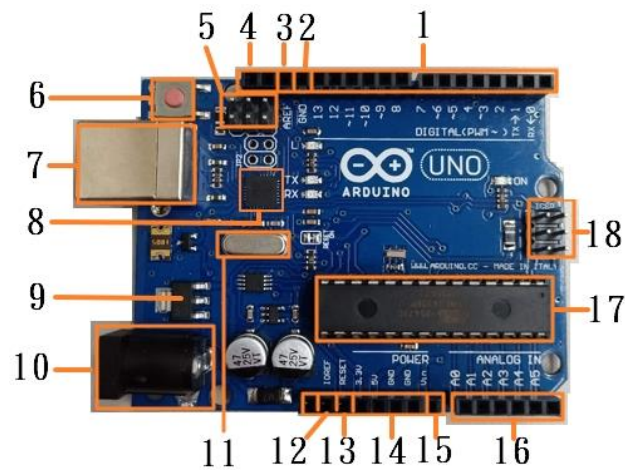


圖 2-2 Arduino Uno 開發版正面

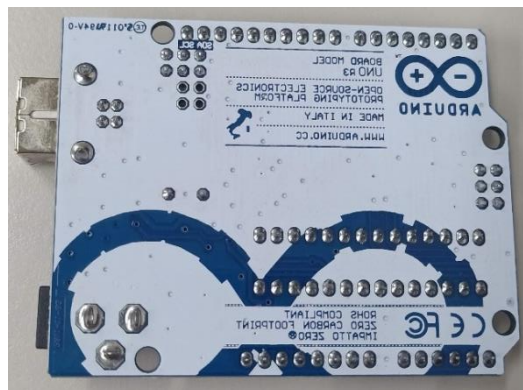


圖 2-3 Arduino Uno 開發板背面

以下 Arduino UNO R3 各部位介紹的編號請參照（如圖 2-2）（圖 2-3 為背面展示）。

1. 腳位 0~13 為數位(DIGITAL)輸入/輸出埠，數位埠上面標示有” ~”符號(PWM)的六個腳位(3、5、6、9、10、11)，其兼具模擬類比信號輸出功能。0、1 接腳也是序列埠的接收(RX)腳和傳送(TX)腳（對照編號 7。）。
2. GND 接地。

3. AREF(Analogue REference)類比輸入埠參考電壓。
4. (左邊)SCL 與 A5 相連，(右邊)SDA 與 A4 相連，在一些實驗中方便接線。
5. ICSP 用以燒錄 USB 轉序列埠韌體。
6. Reset 重置按鈕，重新啟動、重新將程式載入。
7. USB 連接埠：可以提供電力和傳送資料，與數位 0、1 腳位是相通的。
8. ATmega16U2，USB 轉序列埠晶片。
9. 穩壓器，令電壓的保持穩定。
10. 7V 至 12V 電源輸入插座，可以電池直接供電。
11. 16MHz 石英晶體。
12. IOREF 此接腳是與該板的 I/O 相對應的電壓(5V)，令 Arduino 擴充板知道運作電壓。
13. RESET 接腳，當輸入低電壓時，將重置 Arduino。功能相當於按下 Reset 重置按鈕。
14. 由左至右的接腳分別為 3.3V、5V 電壓輸出和兩個 GND 接地。

15. Vin 電壓輸入。
16. A0~A5 為類比輸入埠，也可以做為數位輸出/入埠使用，編號則為 14~19，但無法輸出類比訊號。
17. ATmega328P，8 位元微控制器。
18. ICSP(In-circuit serial programming)，連接 Bootloader(開機啟動程式)燒錄器的接腳。

2.1.3 超音波感測器(HC-SR04)

2.1.3.1 超音波(Ultrasound)

一般人耳能聽見聲音的最高頻率為 20kHz，而高於此頻率以上的聲波則定義為超音波，自然界許多生物即是透過這樣的聲波來傳達訊息、定位以及迴避障礙物。如(圖 2-4)所示，20Hz 以下為次聲波，20Hz~20kHz 之間為可聽聲波，20kHz 以上頻率就是本章要探討的超音波，運用範圍涵蓋生物、化學、醫療及工業上。

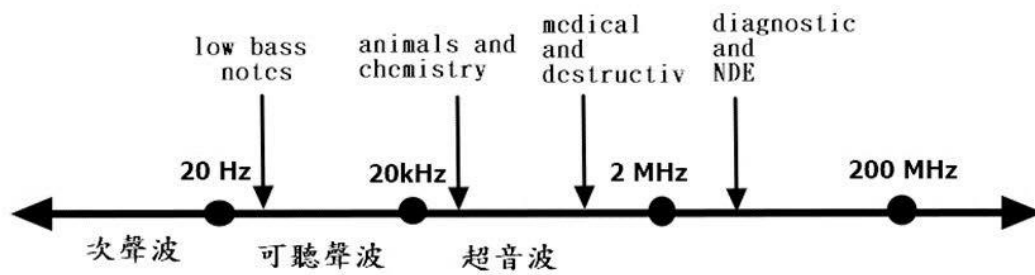


圖 2-4 聲波分佈

2.1.3.2 超音波元件

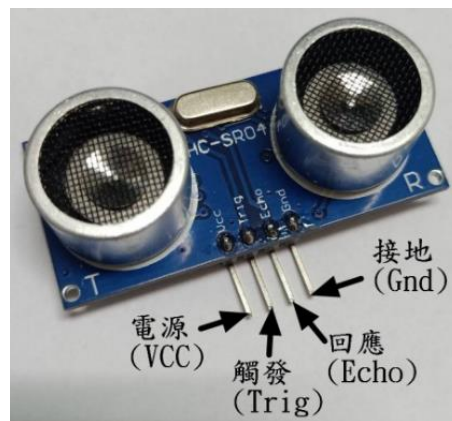


圖 2-5 HC-SR04

超音波感測模組(實物如圖 2-5)上面通常有兩個超音波元件，分別用於發射、接收超音波。當我們在觸發(Trig)腳位輸入 10 微秒以上的高電位時，會發射一連串 40 kHz 的超音波，發射之後與接受傳回來的超音波之前，回應(Echo)腳位會呈現高電位，程式便可由回應(Echo)腳位的高電位脈衝持續時間去推算距離。(圖 2-6)

聲速= $331.5(\text{m/s}) + 0.6 * \text{攝氏溫度}$ ，所以在室溫 20 度環境中的聲速為 $331.5 + 0.6 * 20 = 343.5(\text{m/s})$ ，大約是 344 (m/s)。超音波測距器從發

射至接收到反射波的時間，乘以聲速將得到超音波往返的距離，最後除以 2 即可求得與反射物體之間的距離，公式如下所示。

$$\text{距離} = 344 \text{ 公尺/秒} \times \frac{\text{傳播時間}}{2}$$

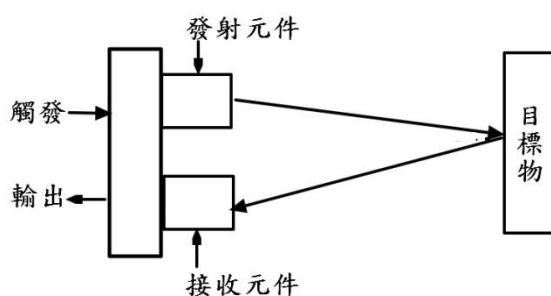


圖 2-6 HC-SR04 元件測距俯瞰圖

2.1.3.3 超音波元件實測

在模組的觸發腳位(TrigPin)輸入 10 微秒的高電位以發射超音波，接著用測量脈衝持續時間的 `pulseIn()` 函數，來讀取回應腳位(EchoPin)呈現高電位的時間，最後利用距離公式來求得與障礙物間的距離。(程式碼如 7.4.5 超音波元件測試程式)(硬體接線如圖 2-7)

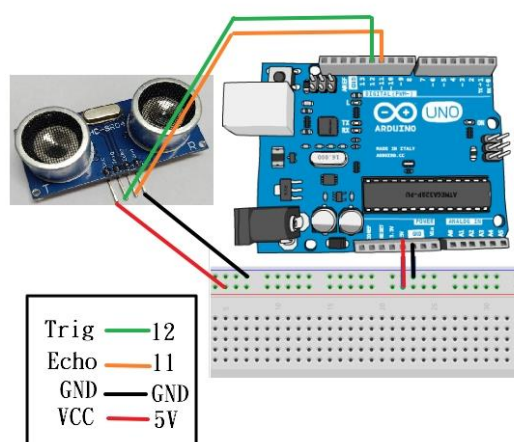


圖 2-7 硬體接線示意圖

2.1.3.4 超音波元件實測結果

(圖 2-8) 為序列埠監控視窗顯示實測結果，視窗中顯示的數值為 EchoPin 高電位脈衝持續的時間，以及元件與障礙物的距離。以下將(圖 2-8) 的數值擷取部分作為(表 2-4)。

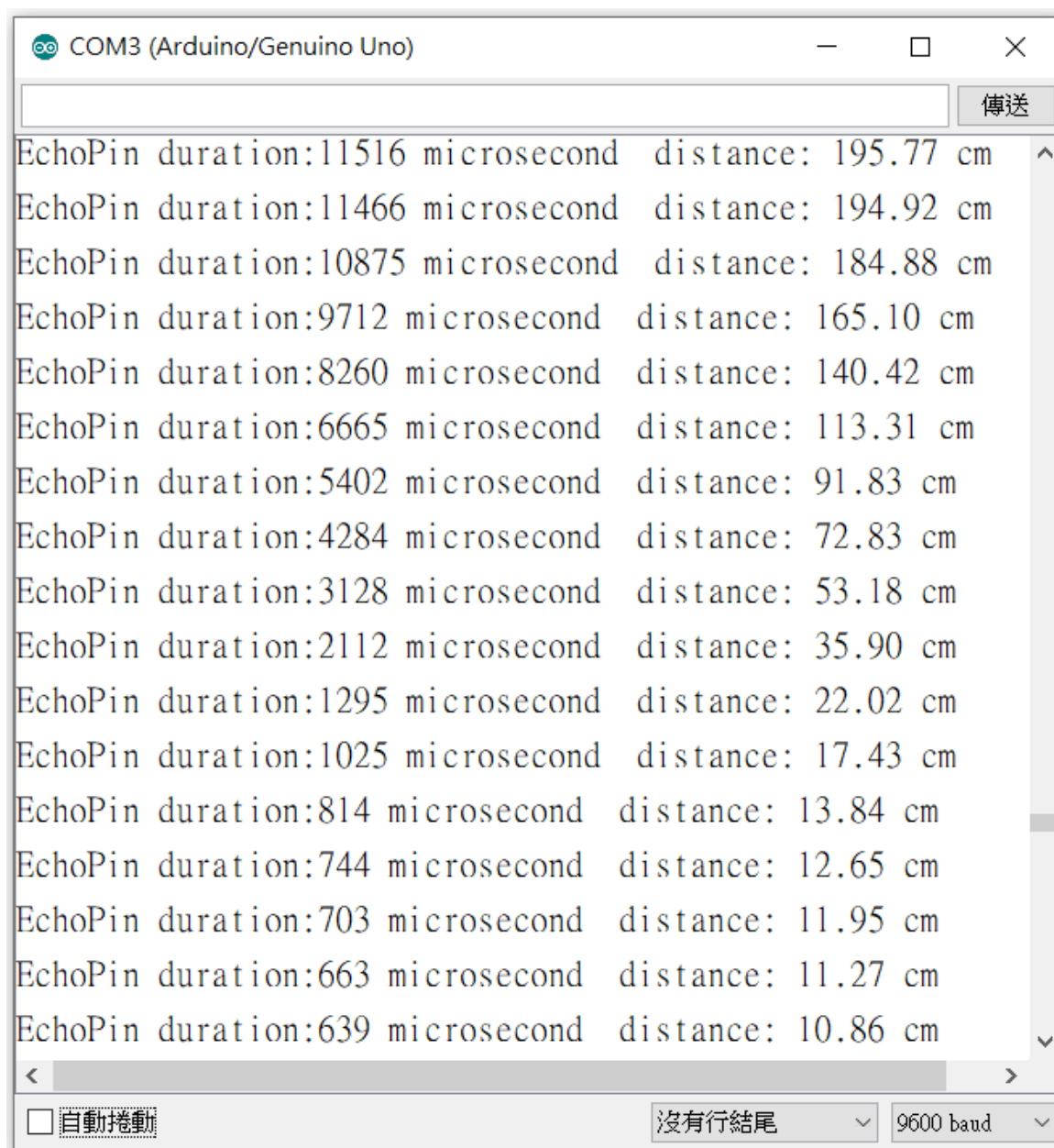


圖 2-8 序列埠監控視窗

表 2-4 HC-SR04 實測數據

EchoPin 高電位時間(μs)	距離實測值 (cm)	以公式計算聲音的速度(m/s)
11516	195.77	339.99
10875	184.88	340.01
9712	165.1	339.99
8260	140.42	340.00
6665	113.31	340.02
5402	91.83	339.99
4284	72.83	340.01
3128	53.18	340.03
2112	35.90	339.96
1025	17.43	340.10
639	10.86	339.91

最後根據(表 2-4)繪製成距離與時間的關係圖(圖 2-9)，從圖中摺線趨勢可以得知，當 EchoPin 高電位脈衝時間越長，元件與障礙物之間的距離也相對越長，距離與時間的關係約成正比，而二分之一乘以超音波傳遞的速度為兩者的斜率。

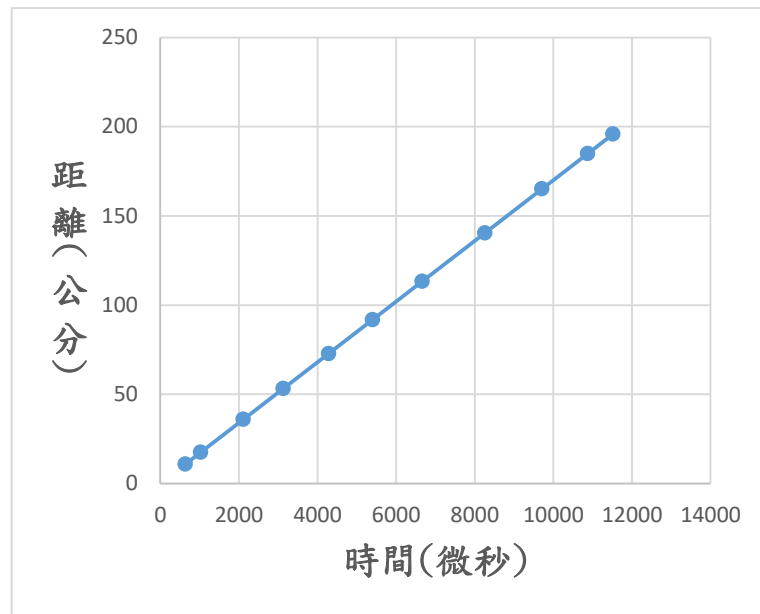


圖 2-9 距離與時間關係圖

2.1.4 FPM10A 指紋辨識

FPM10A(實物如圖 2-10)(腳位如表 2-6)是一個光學指紋辨識模組，光學式的設計較電容式要早，為透過光線形成陰影掃描，使用時手指按壓於三稜鏡上，靠著光源反射讓隆起線顯現出來，再透過感光元件擷取影像。指紋辨識器依照技術可以分為光學式、電容式以及超聲波式；電容式主要為透過電荷量溫差及壓力掃描，成本較為光學式高，感應器易受到汗水等外在因素影響，導致耐用度較差。超聲波指紋辨識主要是利用聲波反射原理，當手指觸碰到超聲波指紋感測器時，它會發出聲波，由於手指表面呈現凹凸不一的紋理，因此聲波反射回去時，再經過處理器運算則可形成指紋圖案。(三者比較如表 2-5)

表 2-5 三者辨識的比較

	光學式	電容式	超聲波式
指紋蒐集方式	蒐集光反射的 陰影	根據指紋凹凸不同 的電容量差異	以超音波掃描表皮 特徵
感測器位置	面板下方	面板上方	面板下方
優點	耐久性好	準確度高	耐久性好、準確度 高
缺點	準確度低	耐久性差	價格高

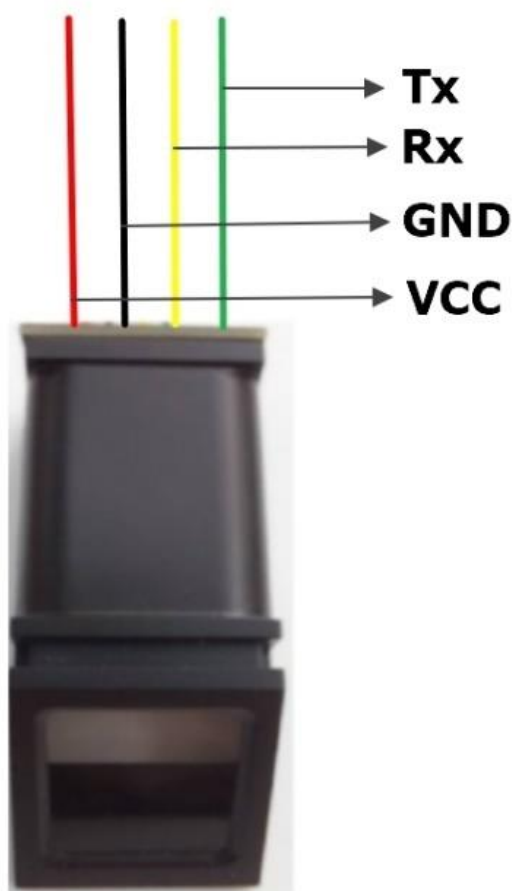


圖 2-10 FPM10A 光學指紋辨識

表 2-6 FPM10A 的腳位說明

PIN	名稱	說明
3	R _X	數字引腳 3，串行
2	T _X	數字引腳 2，串行
	VCC	3.3V~5V
	GND	接地線

2.1.5 有源蜂鳴器

蜂鳴器分為有源蜂鳴器和無源蜂鳴器兩種，上述提及的(源)是指震盪源。有源蜂鳴器本身帶有震盪源，其內建一組固定的頻率，接通電源後便會發出固定的音調，於程式控制上較為方便；無源蜂鳴器則必須透過程式寫入頻率才能得到我們需要的音調。

本實驗使用的蜂鳴器為有源蜂鳴器(KY-012)(圖 2-11)，工作電壓為 3.5V~5.5V，尺寸長 1.85 公分、寬 1.5 公分，最大電流 30mA / 5VDC，工作溫度為 -20 °C ~70 °C，當信號(S) Pin 輸入高電平時，蜂鳴器發出頻率大約 2.5 kHz 的聲音。

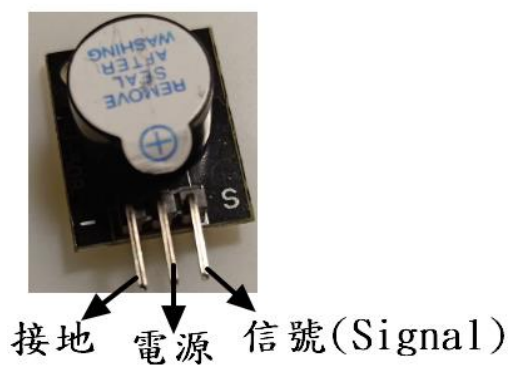


圖 2-11 有源蜂鳴器

2.1.6 Arduino 模型車

為一台簡單的四輪小車(實物如圖 2-12)，用於在做整體測時所需要的車模型。



圖 2-12 Arduino 模型車

2.2 軟體介紹

2.2.1 Arduino 平台

Arduino 是一個開放原始碼的單晶片微控制器，它使用了 Atmel AVR

單晶片，採用了開放原始碼的軟硬體平台，建構於簡易輸出/輸入介面板，並且具有使用類似 Java、C 語言的 Processing/Wiring 開發環境。Arduino 出現之前，若是想做個自動控制的設備，像是要控制一些感測器或是透過繼電器控制家電，大家想到的就是單晶片、8051，這時就必須深入了解單晶片的程設設計，除了程式難度較高，所需的設備成本也高出許多；但是 Arduino 不一樣，它把控制晶片和燒錄功能整合在一塊小板子上，並讓 Pin 腳更容易接線、配合麵包板，可以輕鬆的接上各類感測器或週邊設備，初學者只要會插杜邦線，就能開始進行開發工作，即使沒有相關背景的人也可以快速學習；還有一點很重要，Arduino 的硬體本身是不主張專利的，在公共許可下任何人都能生產印刷電路板的複製品，還能重新設計，甚至銷售原設計的複製品。

2.2.2 LabVIEW 平台

2.2.2.1 LabVIEW 基本介紹

LabVIEW(實驗室虛擬儀器工程平台如圖 2-15)是美國國家儀器公司開發的圖形化程式編譯平台，LabVIEW 程式語言，也被稱為 G 語言。LabVIEW 早期是為了儀器自動控制所設計，至今轉變成為一種逐漸成熟的高階程式語言。圖形化程式編譯平台是只要靠幾個內建的圖形工具戶

相連接，就能完成一個簡單的程式設計。並不需要像一般的程式設計還需要去學程式語法和一長串的程式長度才能完成。(目前可支援 Windows，UNIX，Linux，macOS 等作業系統)

2.2.2.2 圖形化編程

LabVIEW 的程式/子程式被稱為虛擬儀器 (VI)。每個 VI 都有三個組成部分：程式方塊圖(圖 2-13)、使用者介面(圖 2-14)、圖示/連接器(圖 2-14)。連接器是用來供其他的程式方塊圖呼叫本 VI 之用。程式設計師可以利用使用者介面上的控制項將資料輸入正在執行的 VI，或者用顯示控制項將運算結果輸出。使用者介面還可以作為程式的介面：每個虛擬儀器 (VI) 既可以當作使用者介面，作為一個程式來執行，也可以作為一個節點放到另一個 VI 程式方塊圖中，通過連接器面板連接起來，而使用者介面則定義 VI 的輸入和輸出。這意味著每個 VI，在作為子程式嵌入到一個大型的專案之前，都可以很方便地進行測試。

在 LabVIEW 編程環境下，藉助已經提供的大量常式和文件，可以很容易地建立小型應用程式。

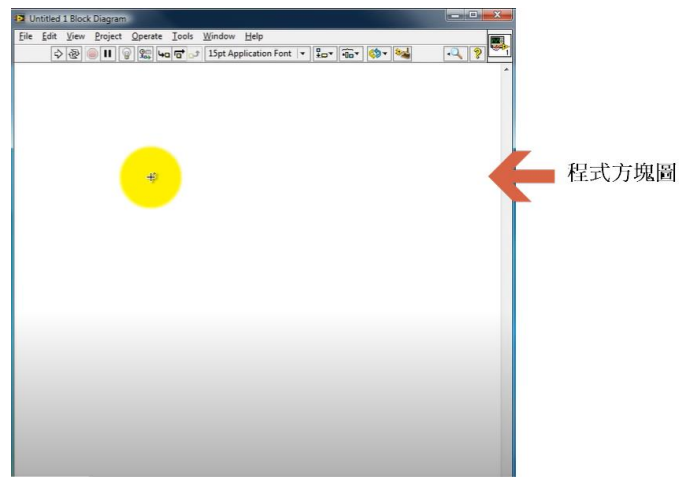


圖 2-13 程式方塊圖

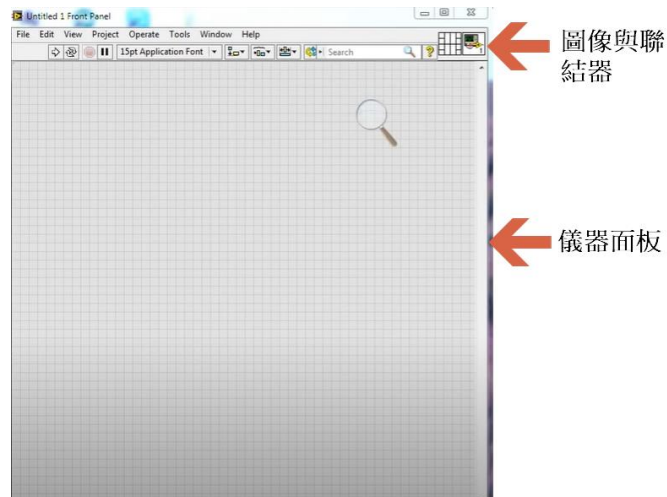


圖 2-14 使用者面板(儀器面板)

開發者	美國國家儀器公司
初始版本	1986年，34年前
穩定版本	2019（2019年5月，11個月前）
程式語言	視覺化程式設計語言
作業系統	Windows、macOS、Linux
系統平台	跨平台 ^[1]
語言	英語、簡體中文
許可協定	專有軟體
網站	http://www.ni.com/labview/

圖 2-15 LabVIEW 簡介

第三章 智能導覽車設計與操作

3.1 溫度濕度 (DHT11)

步驟一：新增一個 arduino 檔案，在草稿碼找到匯入程式庫，並點選管理程式庫。(如圖 7-19)

步驟二：在程式庫管理員視窗上搜尋「DHT11」，並安裝下方一個名為「SimpleDHT by Winlin」的程式庫。(如圖 7-20)

步驟三：點選檔案，找到「SimpleDHT」中的「DHT11Default」並點選，會出現另一個新視窗。(如圖 7-21 圖 7-22)

步驟四：打開新視窗後，注意到下圖程式碼中的 PIN 腳設定為 2，所以在元件與 UNO 板連接時，DAT 的那支接腳要接到 PIN2。將鮑率由 115200 改為 9600。(如圖 7-23)

步驟五：將 DHT11 與 UNO R3 板接上。(如圖 3-1)

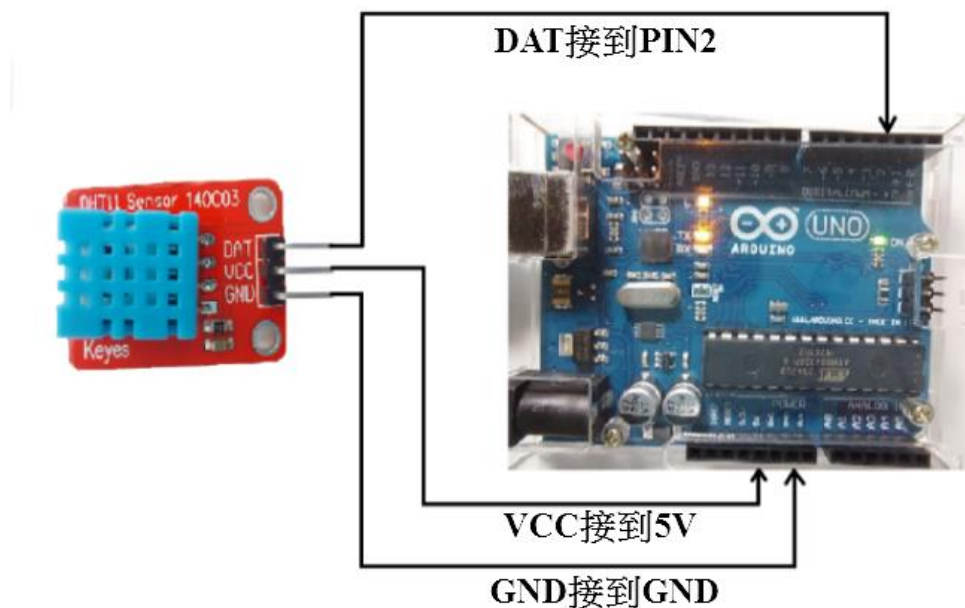


圖 3-1 DHT11 電路設計

步驟六：驗證並上傳程式，確認無誤後，打開右上方的「序列埠監控視窗」，即可顯示溫度及溼度的數值。（如圖 7-24）

3.2 倒車雷達

開啟 Arduino 軟體並新增一個檔案，接著開始編寫倒車雷達的程式碼。本裝置由兩個 HC-SR04 元件搭配一個蜂鳴器所組成。（7.4.3 倒車）

步驟一：首先決定兩種元件在開發板上使用的腳位，並對其進行宣告(HC-SR04 的 Trig Pin、Echo Pin 和蜂鳴器的信號 S(signal) Pin)。

步驟二：將 Trig Pin、蜂鳴器的 S Pin 設定為輸出腳位，Echo Pin 設定為輸入腳位。

步驟三：利用 digitalWrite() 函數分別給予 Trig Pin 高、低電位，接著以 pinMode() 函數讀取 Echo Pin 的電位，以及 pulseIn() 讀取收到高電位的時間。最後運用(距離 = 344 公尺/秒 $\times \frac{\text{傳播時間}}{2}$) 公式完成測量。

步驟四：用 if-else 判斷式，自行設定車體與障礙物之間每階段距離所要發出的警示聲響。

步驟五：硬體連接如(圖 3-2)，首先將兩顆 HC-SR04 元件個別的 Trig Pin 和 Echo Pin 分別連接於開發板的 12、11、10、9 腳位，蜂鳴器的 S Pin 連接於 13 腳位，最後將三個元件的 VCC 和 GND Pin 連接於麵包板，即完成硬體電路連接。

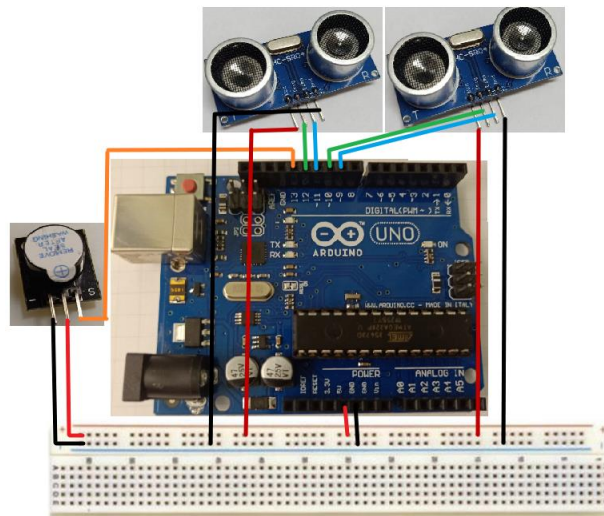


圖 3-2 硬體接線示意圖

步驟六：最後利用 USB 將程式碼傳入開發板，即完成操作。

3.3 防撞功能

開啟 Arduino 軟體並新增一個檔案，接著開始編寫防撞功能的程式碼。本裝置由四個 HC-SR04 元件組成。(防撞功能程式)

步驟一：首先決定四個 HC-SR04 元件個別的 Trig Pin 和 Echo Pin 在開發板上使用的腳位，並對其進行宣告。

步驟二：將四個 HC-SR04 元件的 Trig Pin 設定為輸出腳位，Echo Pin 設定為輸入腳位。

步驟三：利用 `digitalWrite()` 函數分別給予 Trig Pin 高、低電位，接著以 `pinMode()` 函數讀取 Echo Pin 的電位，以及用 `pulseIn()` 讀取收到高電位的時間。最後運用(距離 = 344 公尺/秒 $\times \frac{\text{傳播時間}}{2}$)公式來測量出距離。

步驟四：硬體連接如下(圖 3-3)所示，將四顆 HC-SR04 元件個別的 Trig Pin 和 Echo Pin 分別連接於開發板的 8、7、6、5、4、3、A0、A1 腳位，VCC 和 GND Pin 連接於麵包板，即完成硬體電路連接。

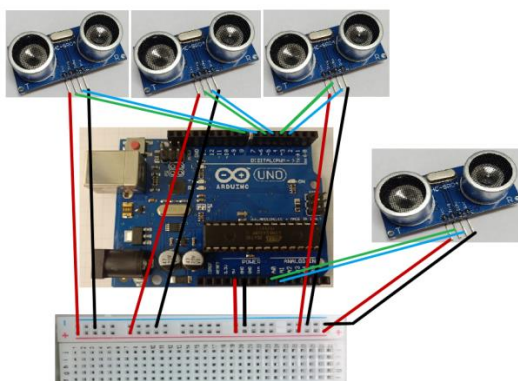


圖 3-3 硬體接線示意圖

3.4 指紋辨識解鎖

步驟一：新增一個 arduino 檔案，在草稿碼找到匯入程式庫，並點選管理程式庫。開啟程式庫後，搜尋「fingerprint」，找到「Adafruit Fingerprint Sensor Library」，點選安裝。(如 7.4.6)

步驟二：開啟範例後，找到第三方程式庫「Adafruit Fingerprint Sensor Library」，開啟「enroll」註冊指紋。(如 7.4.7)

步驟三：將指紋辨識模組與 UNO 板子結合。(如圖 3-4)

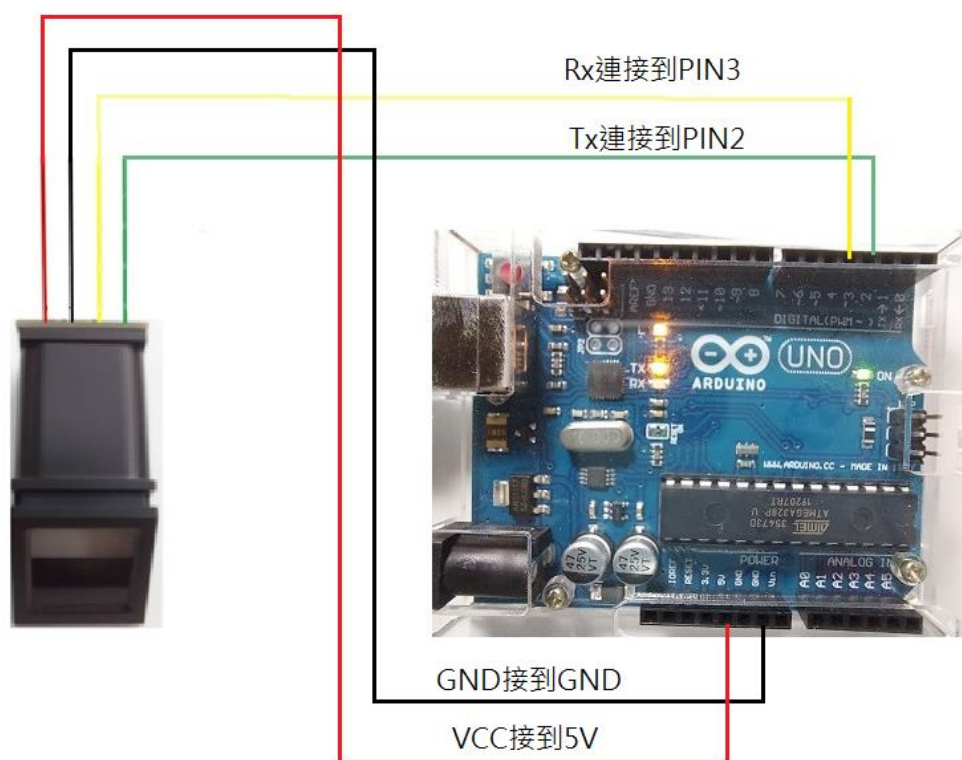


圖 3-4 指紋辨識接法

步驟四：編譯並上傳程式碼，打開右上角的序列埠監控視窗，輸入一個數值即可註冊指紋。(如 7.4.7)

步驟五：再次開啟範例，找到第三方程式庫「Adafruit Fingerprint Sensor Library」，開啟「Fingerprint」辨識已註冊的指紋。編譯並上傳程式碼，打開右上角的序列埠監控視窗，即可辨識剛才輸入的指紋。(如 7.4.8)

3.5 連結 LabVIEW 的 Arduino 統合程式

步驟一：先把以上有關 Arduino 的程式都開啟，此部分可以分為三大區域，分別是宣告(steup 以上)區域、初始設定(steup 內)區域、主程式(steup 以下所有)區域(如圖 3-5)。

步驟二：開啟新的 Arduino 程式，此部分可以分為四大區域，分別是宣告區域、setup 區域、主程式區域、函式區域(如圖 3-6)。

步驟三：把步驟一的宣告區域的變數和函式庫都複製到步驟二的宣告區域。再把步驟一的 setup 區域複製到步驟二的 setup 區域、最後把步驟一的主程式區域複製到函式區域，一個單項主程式區域對應一個統合程式的函式區域(例如:在統合程式內的溫溼度的函示名稱為 void one(){}，倒車雷達的為 void two(){}))，所以會用到多個函式。

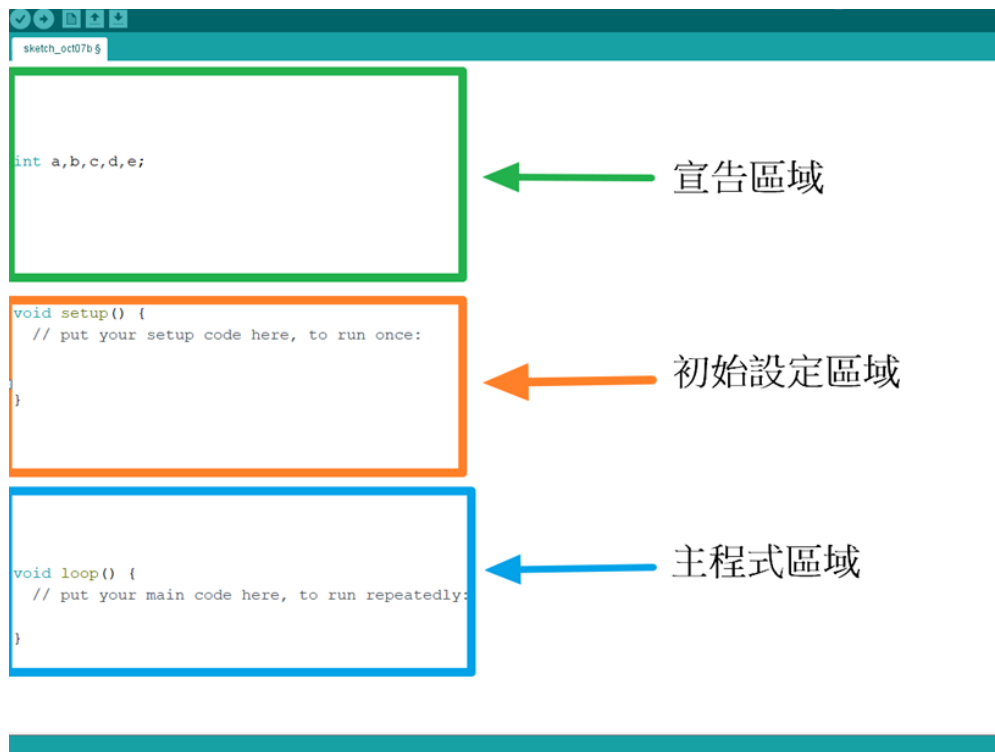


圖 3-5 單項(溫溼度、防撞)程式區域分布

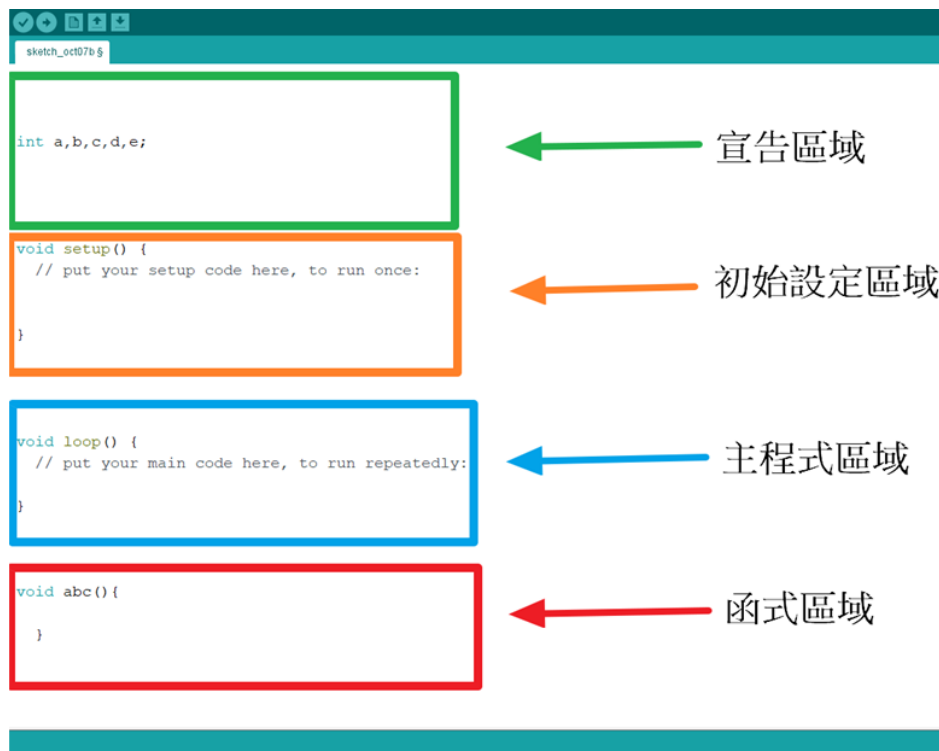


圖 3-6 統合程式區域分布

步驟四：更改統合程式的宣告變數區域部分，把各個電子元件所

需要使用的腳位在程式上規劃好(此部分自由發揮，視需求而定)，可參考附錄程式(如 7.4.9)。

步驟五：撰寫統合程式的主程式區域，並且調整函式的名稱去配合主程式的執行，程式參考附錄(如 7.4.9)。

步驟六：更改函式區域各單項程式，調整為適合 LabVIEW 去傳輸以及接收資料的程式，程式參考附錄(如 7.4.9)。

步驟七：嫌麻煩的話直接複製附錄程式(如 7.4.9)。

3.6 LabVIEW 設計

3.6.1 LabVIEW 小技巧

1. 使用者介面的大部分元件都可以連點 2 下，將會導向此元件在程式方塊圖的何處。
2. 在程式方塊圖部分，某些元件在腳位部分點擊右鍵可生成該元件的 Constant(固定訊號輸入/輸出)、Control(可在使用者介面操控的訊號輸入)、Indicator(可在使用者介面顯示的輸出資料)。(如圖 3-7)

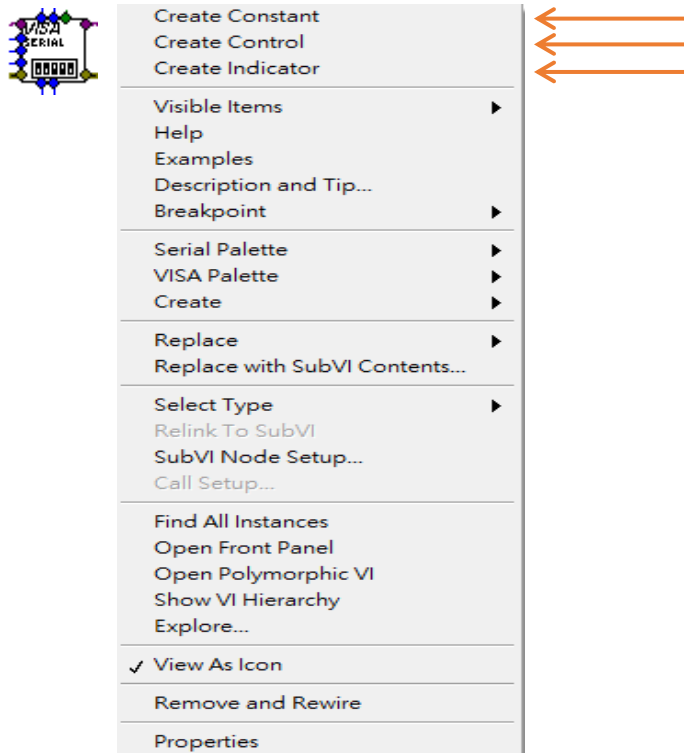


圖 3-7 LabVIEW 小技巧 1

3. 在使用者介面可點選 Show Block Diagram，即可跳到程式方塊圖畫面。如(圖 3-8)

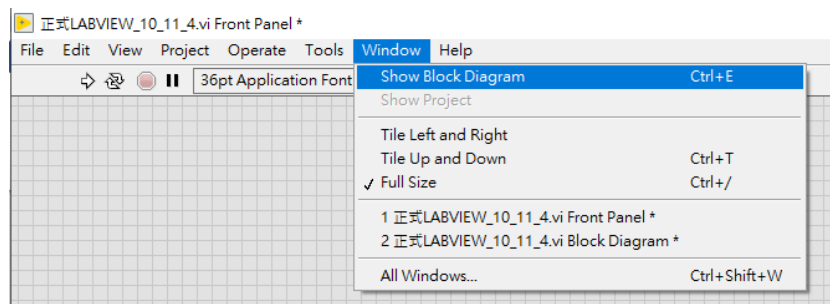


圖 3-8 LabVIEW 小技巧 2

3.6.2 程式方塊圖的元件&簡單介紹&呼叫方式

1. VISA configure serial port：去讀取連接電腦 USB 輸出入阜的資料。

呼叫方式如(圖 3-9)(紅色箭頭和數字為步驟)

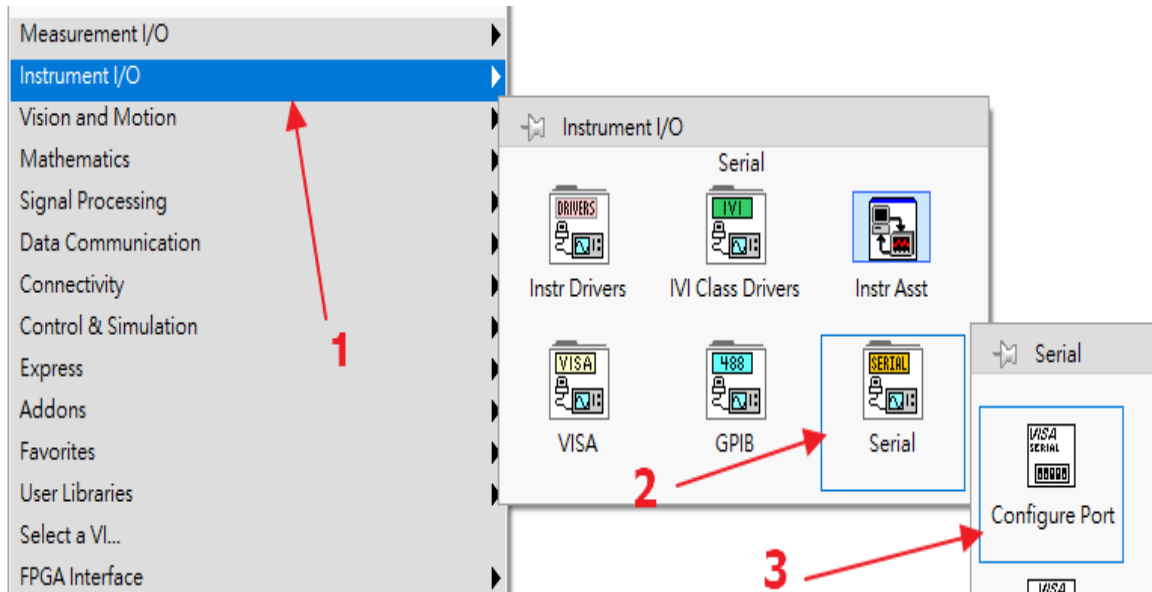


圖 3-9 VISA configure serial port

2. VISA Write：透過 USB 寫入資料到 Arduino UNO 裡面。呼叫方式如(圖 3-10) (紅色箭頭和數字為步驟)
3. VISA Read：透過 USB 讀取 Arduino UNO 回傳的資料。呼叫方式如(圖 3-10) (紅色箭頭和數字為步驟)
4. VISA Clear：清除所有的回傳資料或輸出訊號。呼叫方式如(圖 3-10) (紅色箭頭和數字為步驟)

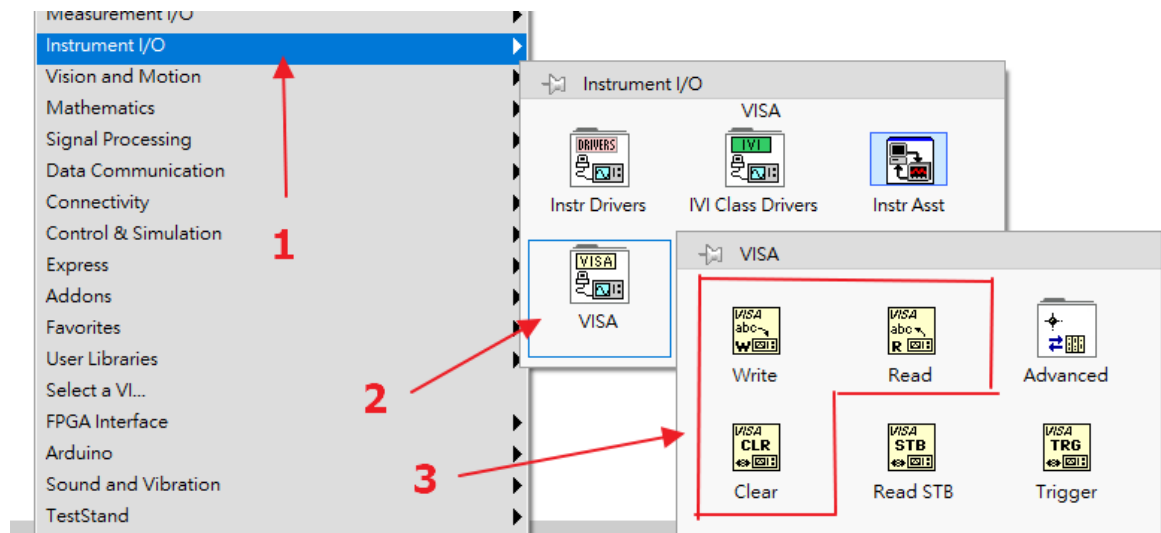


圖 3-10 VISA 元件

5. While Loop(條件式無窮迴圈): 只要沒控制停止, 就會一直執行。

呼叫方式如(圖 3-11) (紅色箭頭和數字為步驟)

6. Case Structure(條件架構): 要有符合條件才會執行。呼叫方式如

(圖 3-11) (紅色箭頭和數字為步驟)

7. Flat Sequence(平面順序): 會依序架構內的動作。呼叫方式如(圖

3-11) (紅色箭頭和數字為步驟)

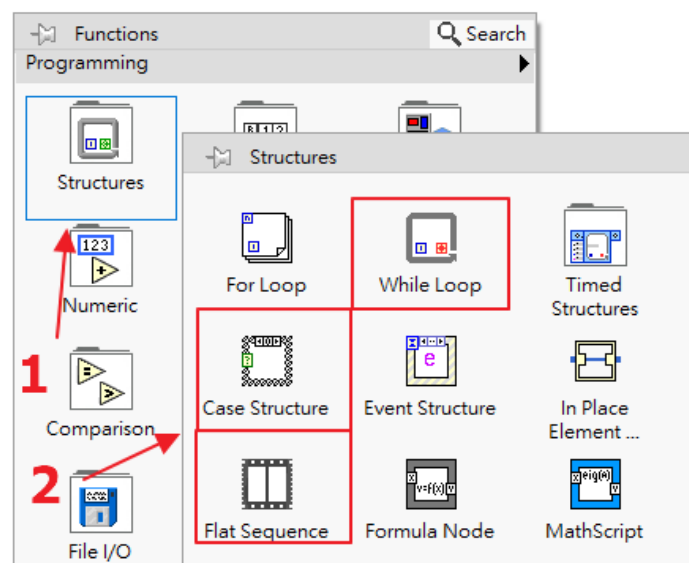


圖 3-11 架構

8. Wait(ms)(時間延遲)：延遲動作的執行。呼叫方式如(圖 3-12)

(紅色箭頭和數字為步驟)

9. Wait Until Next ms Multiple：等到下一個毫秒的倍數。呼叫方式

如(圖 3-12) (紅色箭頭和數字為步驟)

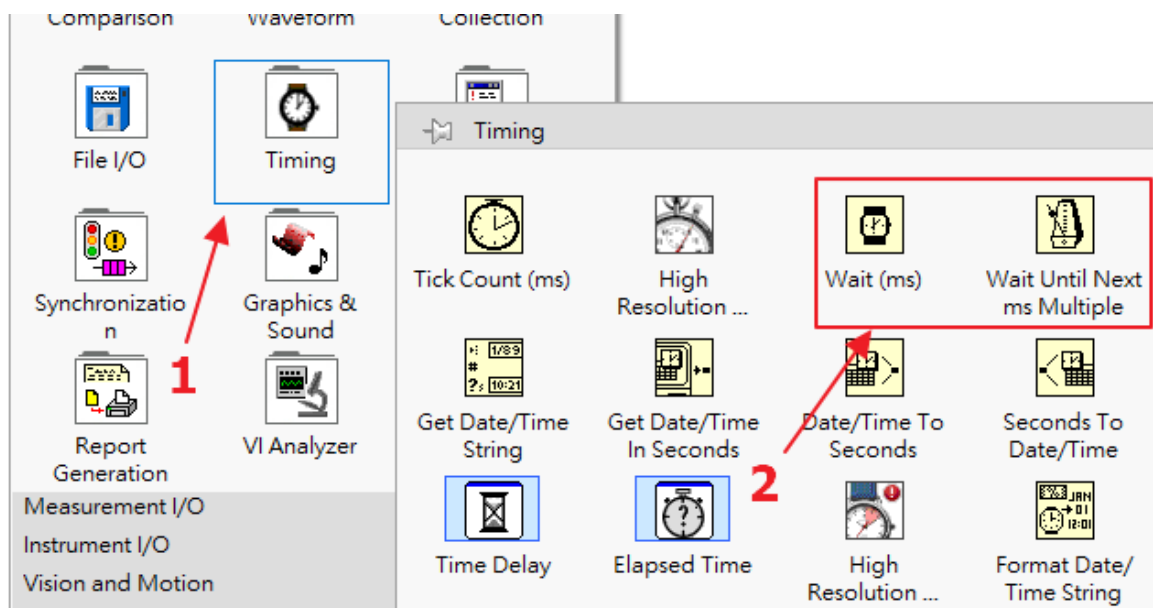


圖 3-12 時間延遲

10. Match Pattern (匹配模式)：可用來分割字串內容(例如:4+2，分

割成4、+、2，三部分)。呼叫方式如(圖 3-13) (紅色箭頭和數

字為步驟)

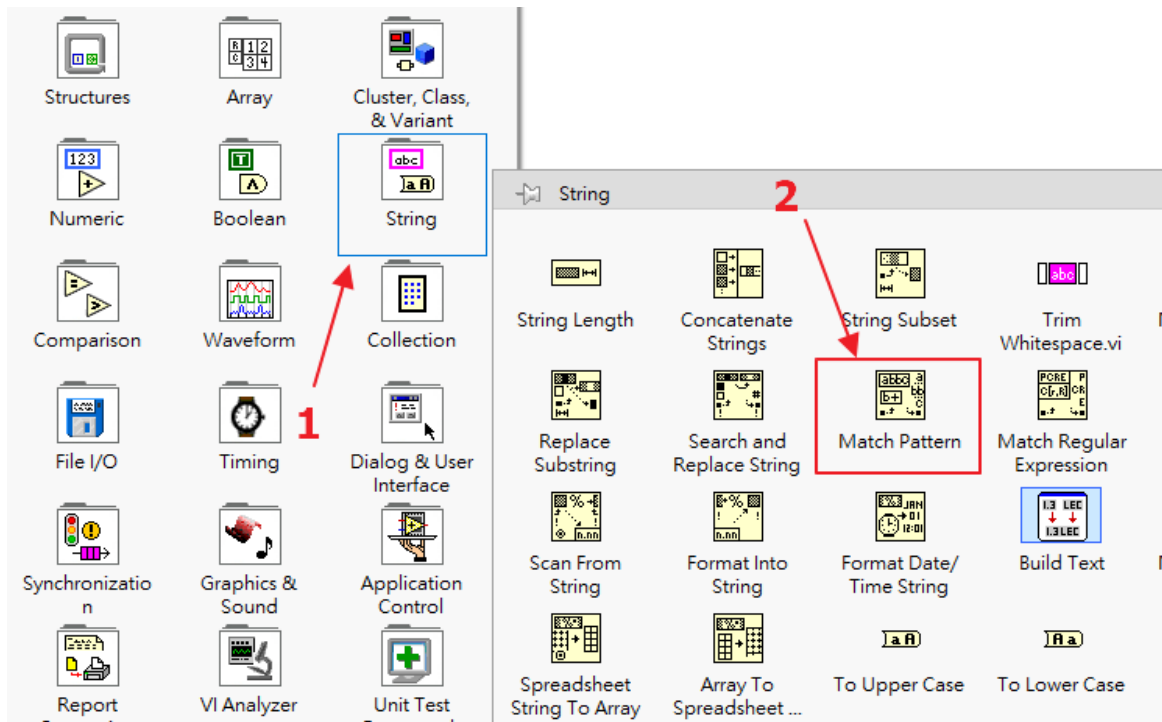


圖 3-13 Match Pattern

11. Fract/Exp String to number：用來把字串轉成數字的工具。呼叫方式如(圖 3-14) (紅色箭頭和數字為步驟)

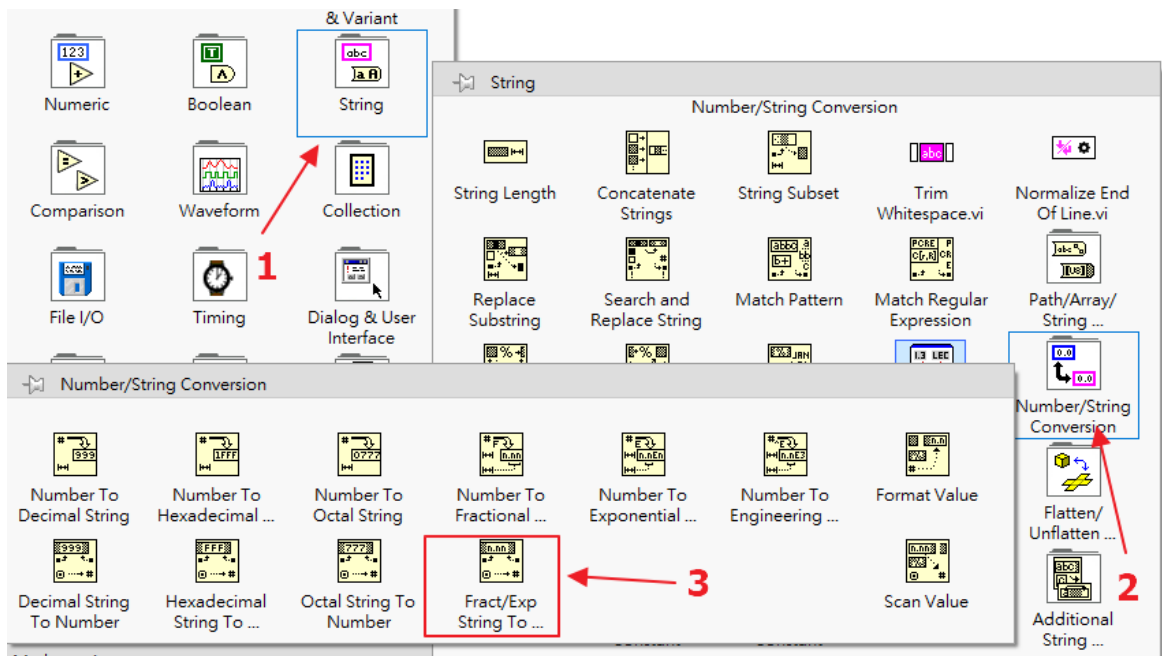


圖 3-14 Fract/Exp String to number

12. VISA Property Node：操控將要執行怎樣的動作。我們將會用到

四種模式(如圖 3-16)，更改模式方式為右鍵點選上方黃色部分

(如圖 3-17)和左鍵點選下放白色部分的右邊箭頭(如圖 3-18)。

呼叫方式如(圖 3-15) (紅色箭頭和數字為步驟)

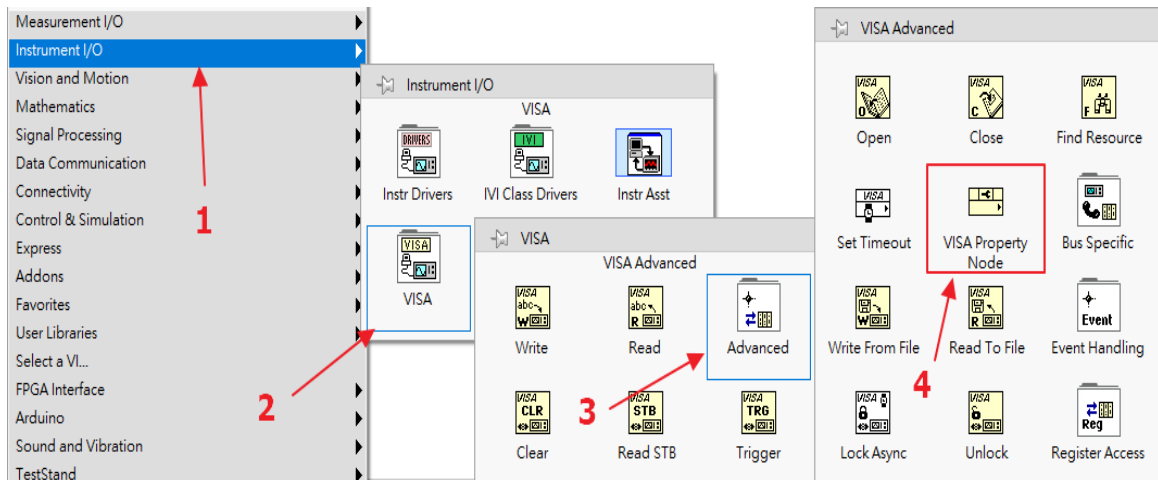


圖 3-15 VISA Property Node

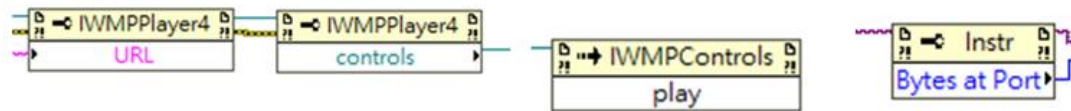


圖 3-16 VISA Property Node 四種模式

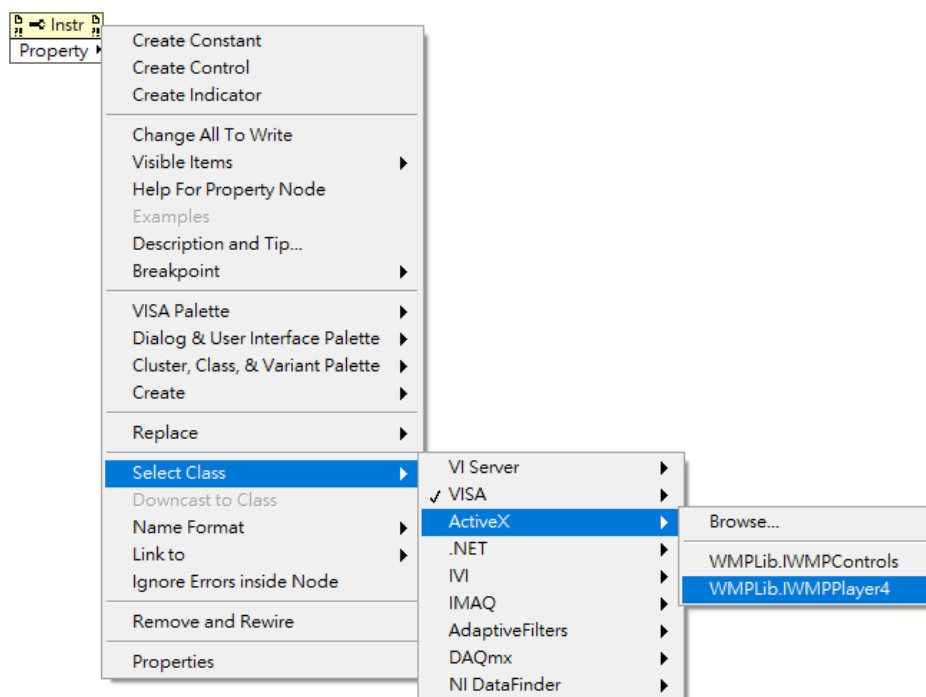


圖 3-17 更改模式操作 1

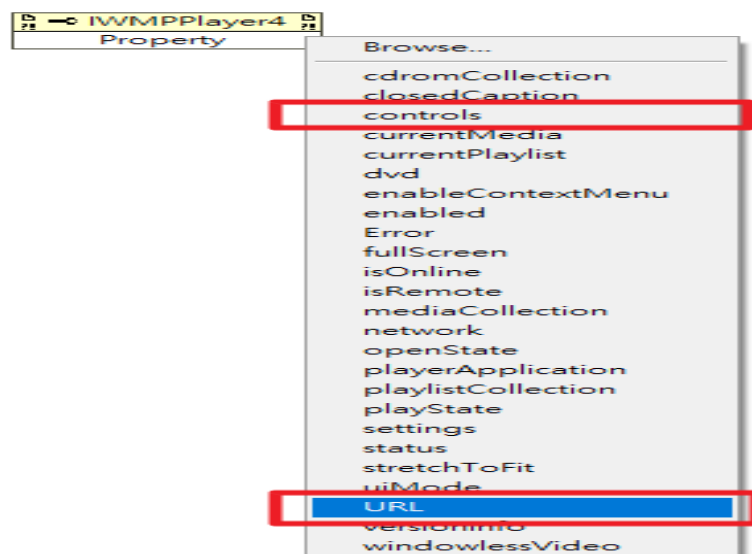


圖 3-18 更改模式操作 2

13. Read JPEG File.vi：讀取圖片檔案。呼叫方式如(圖 3-19) (紅色箭頭和數字為步驟)

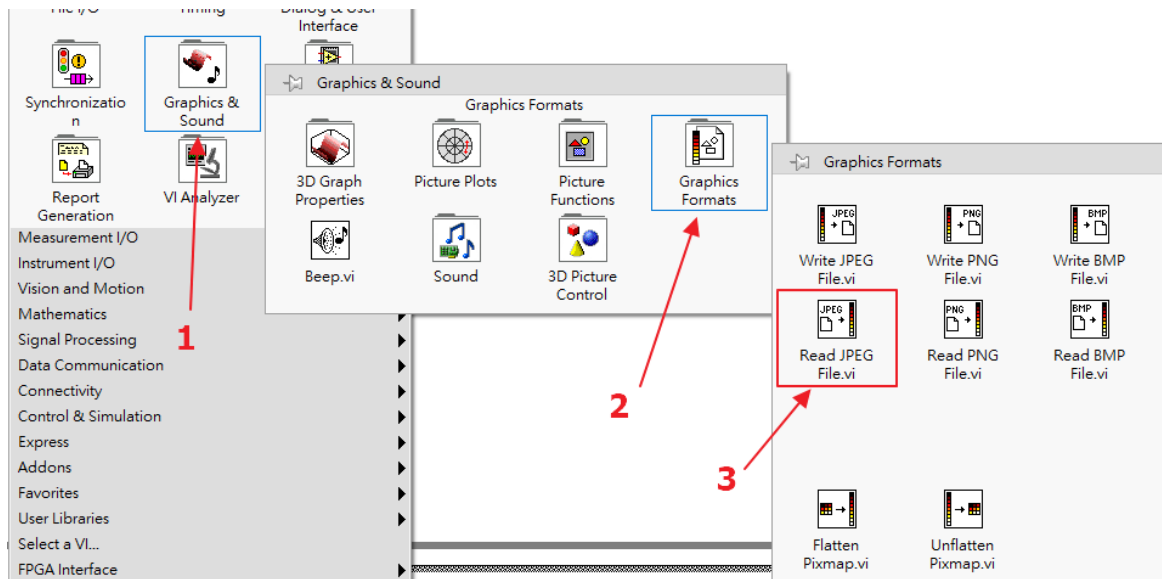


圖 3-19 Read JPEG File.vi

14. Draw Flattened Pixmap.vi：輸出圖片檔案資料。呼叫方式如(圖 3-20) (紅色箭頭和數字為步驟)

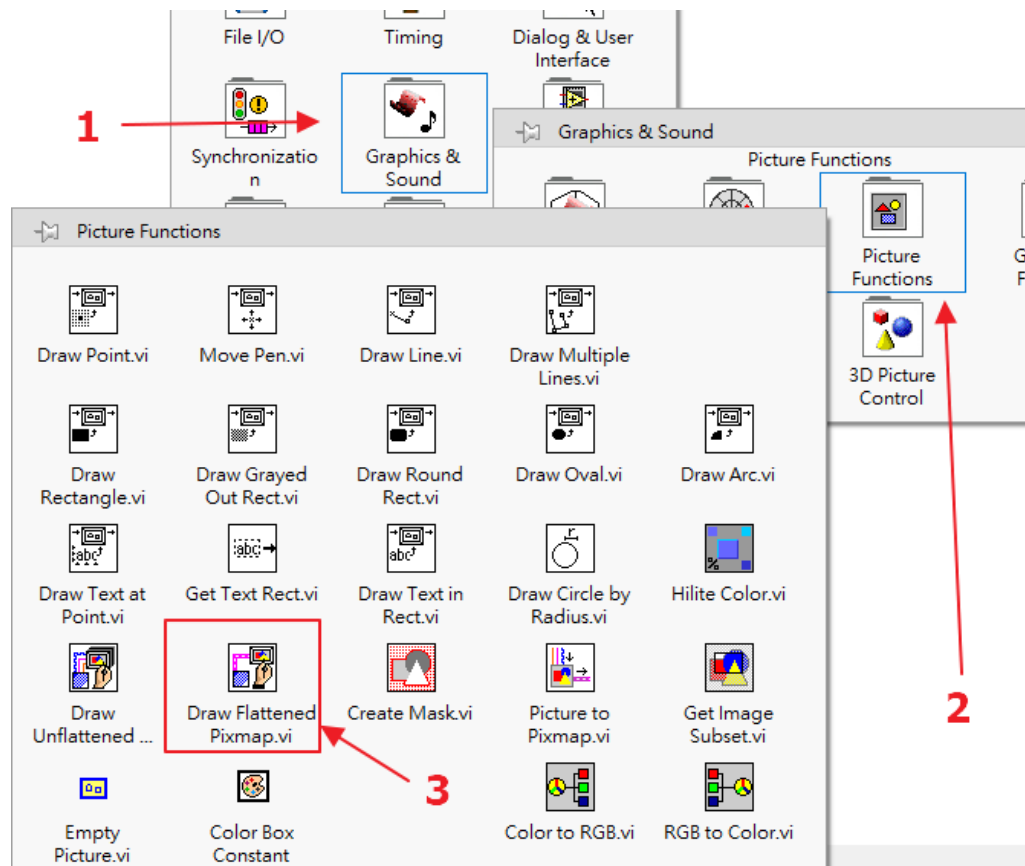


圖 3-20 Draw Flattened Pixmap.vi

15.Automation Open：自動開啟影片檔案。呼叫方式如(圖 3-21) (紅色箭頭和數字為步驟)

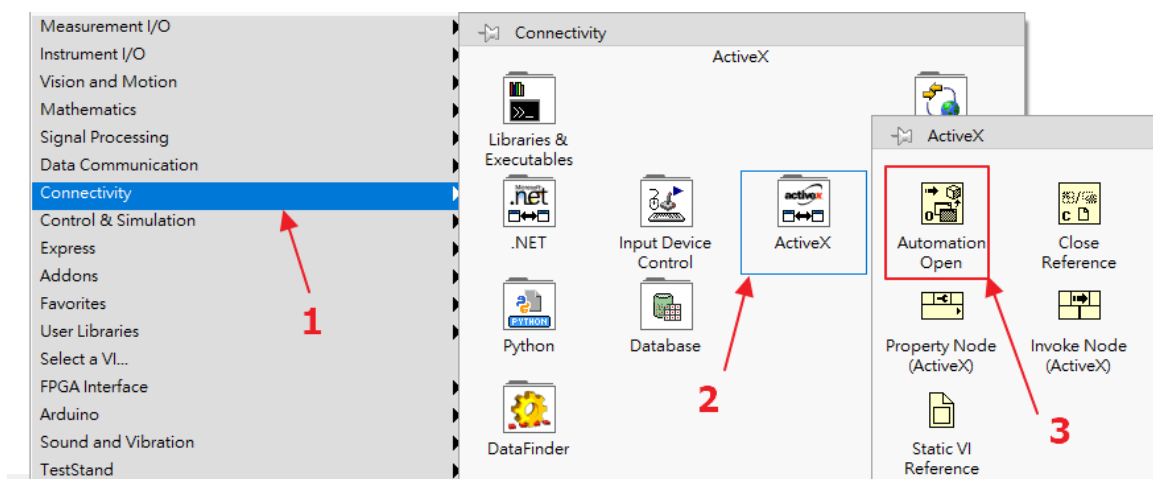


圖 3-21 Automation Open

16.Path To String：路徑轉成字串。呼叫方式如(圖 3-22) (紅色箭頭和數字為步驟)

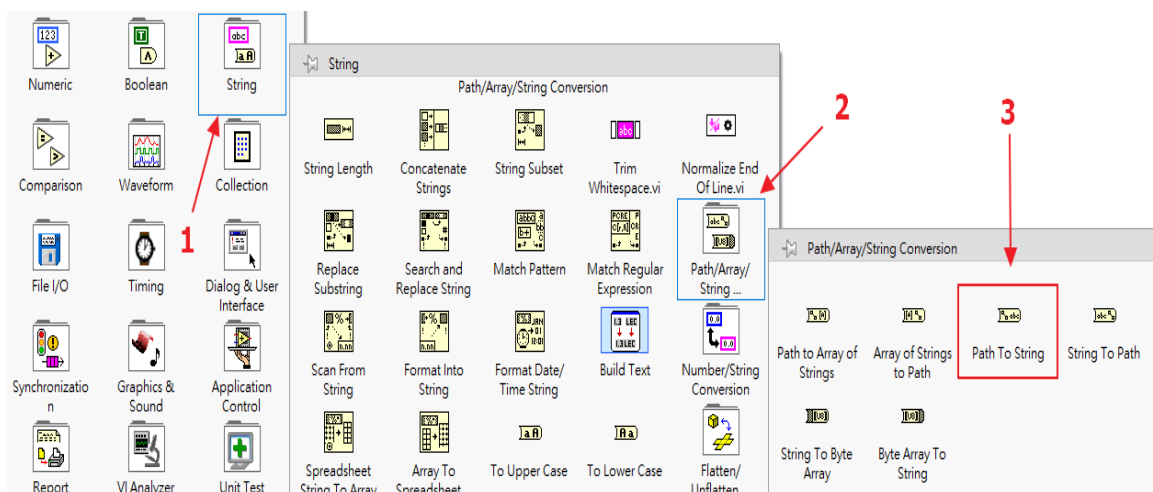


圖 3-22 Path To String

17.Greater Than 0?：判斷數值是否大於0。呼叫方式如(圖 3-23) (紅色箭頭和數字為步驟)

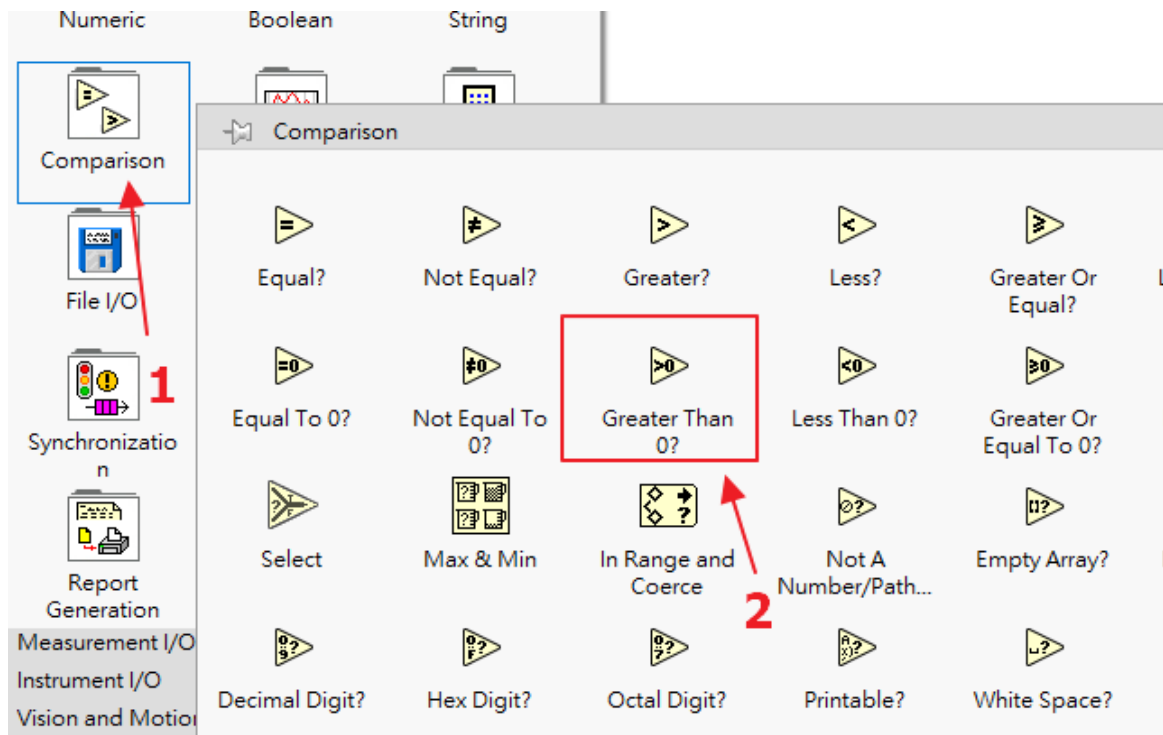


圖 3-23 Greater Than 0?

3.6.3 使用者面板的元件&簡單介紹&呼叫方式

1. Vertical Toggle Switch(垂直撥動開關)：能維持狀態的控制開關。

呼叫方式如(圖 3-24)

2. OK Button(按鈕)：不可維持狀態的控制開關。呼叫方式如(圖 3-24)

3. Stop Button(按鈕)：不可維持狀態的控制開關。呼叫方式如(圖 3-24)

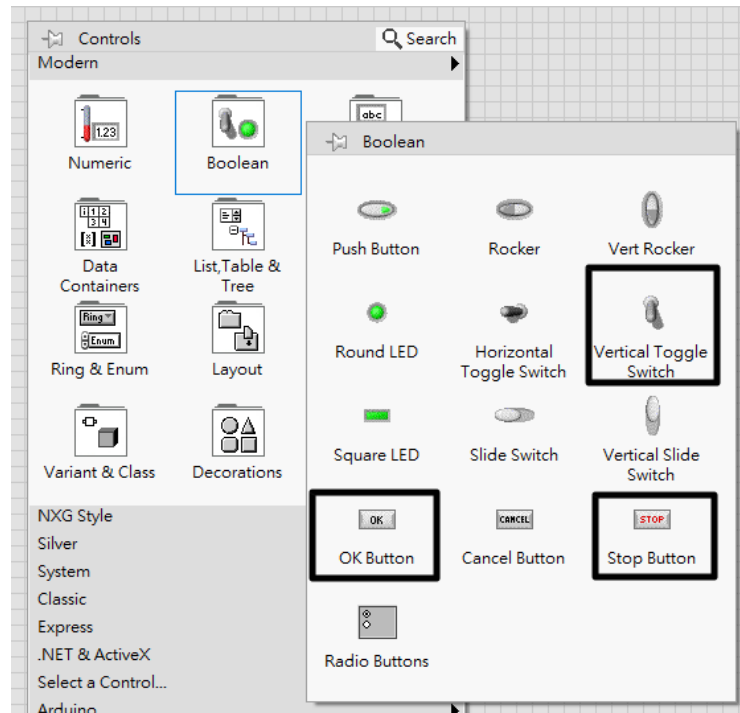


圖 3-24 控制開關

4. Numeric Indicator : 顯示數值。呼叫方式如(圖 3-25)
5. Vertical Fill Slide : 顯示數值條(藍色)。呼叫方式如(圖 3-25)
6. Thermometer : 顯示數值條(紅色)。呼叫方式如(圖 3-25)

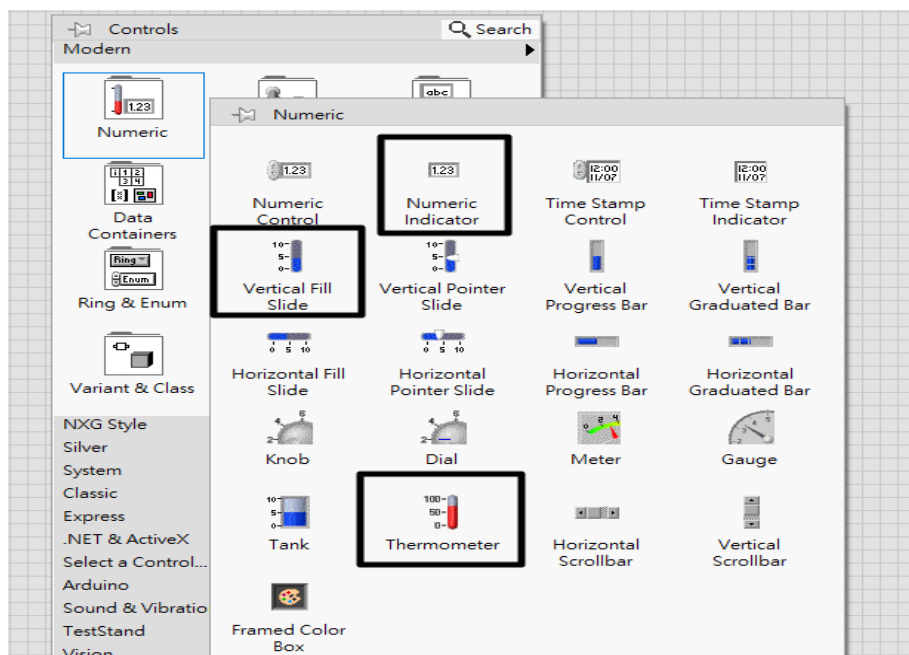


圖 3-25 數據顯示

7. VISA Resource：選擇將連接到的 USB 阜。呼叫方式如（圖 3-26）

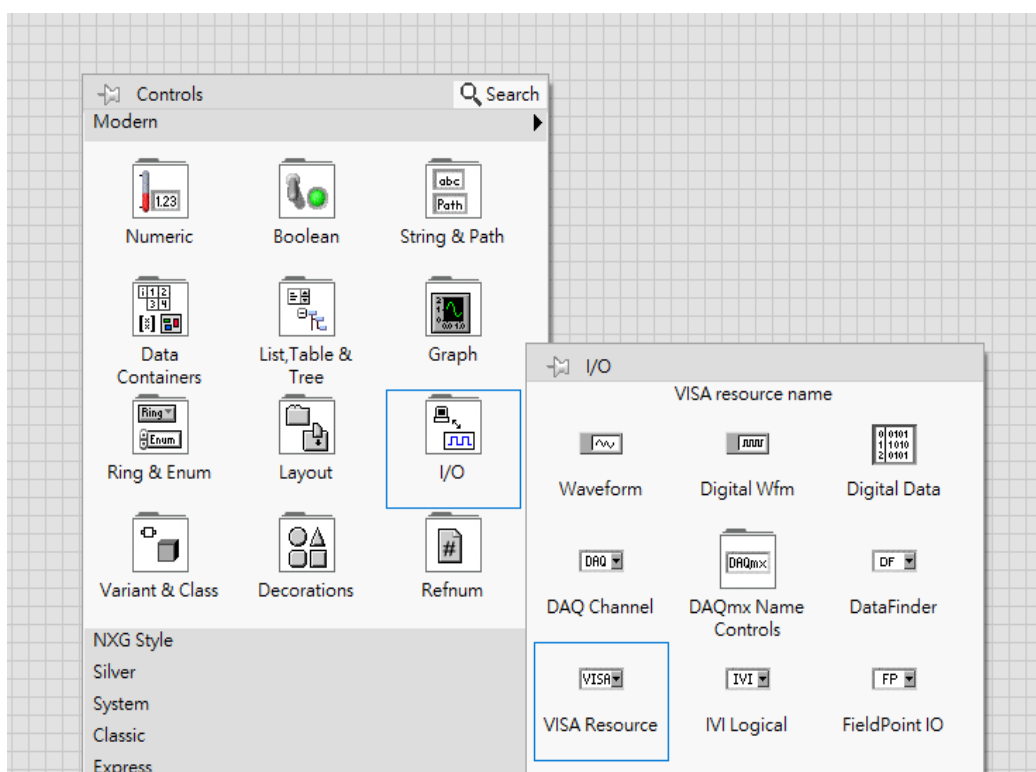


圖 3-26 VISA Resource

8. Windows Media Player：影片撥放器。呼叫方式如(圖 3-27)

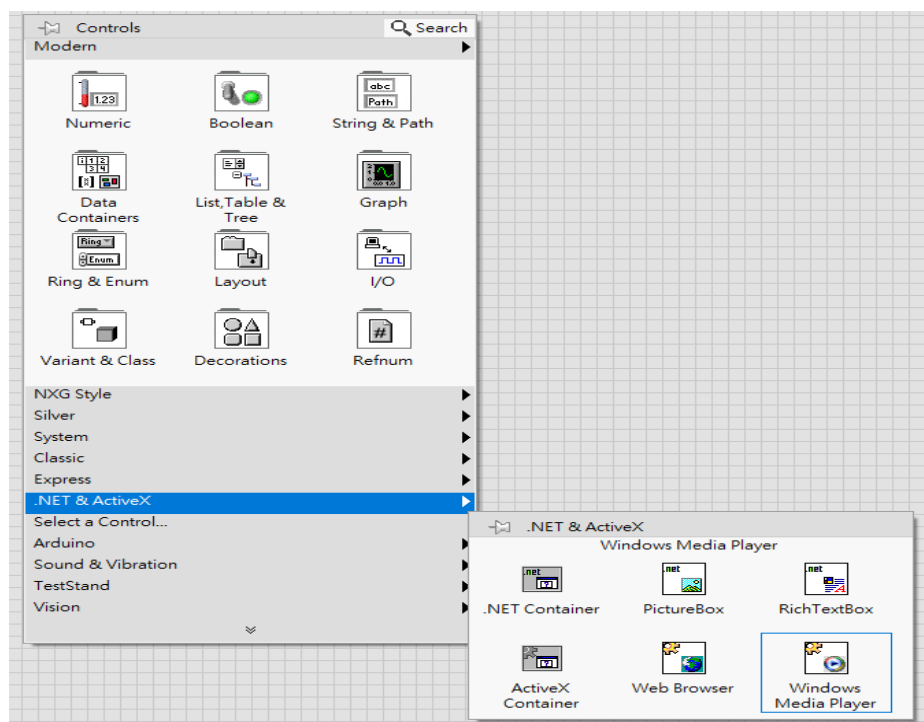


圖 3-27 Windows Media Player

3.6.4 外部程式

LabVIEW 是圖形介面的程式設計，要設計視覺外觀(使用者介面)及連結程式方塊圖。在程式方塊圖的部分，必要時要指定參數。

附錄(7.5.1)為程式方塊圖的設計，圖 7-29 為 Case Structure 是 True 狀態的圖形介面，圖 7-30 為 Case Structure 是 False 狀態的圖形介面。程式方塊圖的接法參考附錄(7.5.1)

3.6.5 溫溼度感測

由 LabVIEW 透過 USB 傳輸控制訊號(01)到 Arduino UNO 板內所儲存的程式，並執行 01 模式的程式，使 Arduino UNO 可去控制溫濕度感測元件(DHT11)的啟動，並且回傳 Arduino UNO 接收到元件的數據到 LabVIEW 使用者介面當中。

附錄(7.5.2)為程式方塊圖的設計，圖 7-31 為 Case Structure 是 True 狀態的圖形介面，圖 7-32 為 Case Structure 是 False 狀態的圖形介面。程式方塊圖的接法參考附錄(7.5.2)

3.6.6 倒車雷達

由 LabVIEW 透過 USB 傳輸控制訊號(02)到 Arduino UNO 板內所儲

存的程式，並執行 02 模式的程式，使 Arduino UNO 可去控制車體後方超音波感測器(HC-SR04)的啟動，此時元件會跑出數據傳到 Arduino UNO，之後 Arduino UNO 會根據得到的數值去控制有源蜂鳴器(KY-012)發出聲音，且同時回傳後方物體距離超音波感測器的距離到 LabVIEW 使用者介面上。

附錄(7.5.3)為程式方塊圖的設計，圖 7-33 為 Case Structure 是 True 狀態的圖形介面，圖 7-34 為 Case Structure 是 False 狀態的圖形介面。程式方塊圖的接法參考附錄(7.5.3)

3.6.7 指紋辨識

LabVIEW 一開始執行時，只有指紋辨識功能是可以用的，其他功能必須等到指紋辨識成功後才能執行。LabVIEW 會傳輸控制訊號(03)給儲存在 Arduino UNO 板的程式，並且執行 03 模式的程式，此時指紋辨識(FPM10A)會一直閃爍紅色燈，這是辨識指紋進行時的燈，亮一次掃描一次。當掃到曾經註冊過的指紋時，會輸出一個大於 0 的數值，這個數值是信任度數值，越高表示越符合，要是掃到沒註冊過的指紋，此時的值是無。所以程式方塊圖的設計是當指紋辨識讀到大於 0 的值時，就會解鎖其他功能和撥放解鎖影片。

附錄(7.5.4)為程式方塊圖的設計，圖 7-35 為 Case Structure 是 True

狀態的圖形介面，圖 7-36 為 Case Structure 是 False 狀態的圖形介面。程式方塊圖的接法參考附錄(7.5.4)

3.6.8 影音導覽

影音導覽是透過先前錄製好的導覽影片存取到顯示器中，再藉由 LabVIEW 設計程式方塊圖，使在 LabVIEW 的使用者介面可以去抓取影片檔案並撥放。

附錄(7.5.5)為程式方塊圖的設計，圖 7-37 為 Case Structure 是 True 狀態的圖形介面，圖 7-38 為 Case Structure 是 False 狀態的圖形介面。程式方塊圖的接法參考附錄(7.5.5)

3.6.9 防撞四角

LabVIEW 會傳輸控制訊號(04)給儲存在 Arduino UNO 內的程式，並且執行 04 模式的程式，此時 Arduino UNO 會送出訊號給左前、右前、左、右的超音波感測器，使超音波感測器開始運作，再去接收超音波感測器回傳的數據並傳到 LabVIEW 的使用者介面。

附錄(7.5.6)為程式方塊圖的設計，圖 7-39 為 Case Structure 是 True 狀態的圖形介面，圖 7-40 為 Case Structure 是 False 狀態的圖形介面。程式方塊圖的接法參考附錄(7.5.6)

第四章 成果與未來展望

4.1 成果外觀展示

圖 4-1 和圖 4-2 和圖 4-3 分別是我們專題研究的最後成果的展示。

圖 4-1 為 Arduino 統合程式的部分截圖，模式 01 為溫溼度感測、模式 02 為倒車雷達、模式 03 為指紋辨識、模式 04 防撞四角。完整程式碼參考附錄 7.4.9。

圖 4-2 為顯示介面，其功能包含了溫溼度感測、防撞四角、倒車雷達、指紋辨識解鎖、影音導覽。

圖 4-3 為實體物，其功能原件包含了 1 個 Arduino UNO 板、6 個超音波感測器(HC-SR04)、1 個有源蜂鳴器(KY-012)、1 個溫溼度感測器(DHT11)、1 個指紋辨識(FPM10A)、1 台模擬車。

```

Arduino 1.8.12
檔案 編輯 草稿碼 工具 說明
 Arduino_10_12_1

////////////////////////////////////
///LabVIEW with arduino IDE 程式設計
char command;
String string;
////////////////////////////////////
///指紋辨識
int getFingerprintIDez();
#include <Adafruit_Fingerprint.h>
SoftwareSerial mySerial(A1, A0);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
// 腳位 A1 對應指紋辨識的綠線
// 腳位 A0 對應指紋辨識的黃線
////////////////////////////////////
//DHT11 溫度濕度感測
#include <SimpleDHT.h>
int pinDHT11 = A2;
SimpleDHT11 dht11(pinDHT11);
//DAT: A2 //溫書度感測資料線街腳為A2
////////////////////////////////////
///倒車雷達
//完美蜂鳴器為數位接腳A0

```

圖 4-1 Arduino 統合程式成果部分截圖

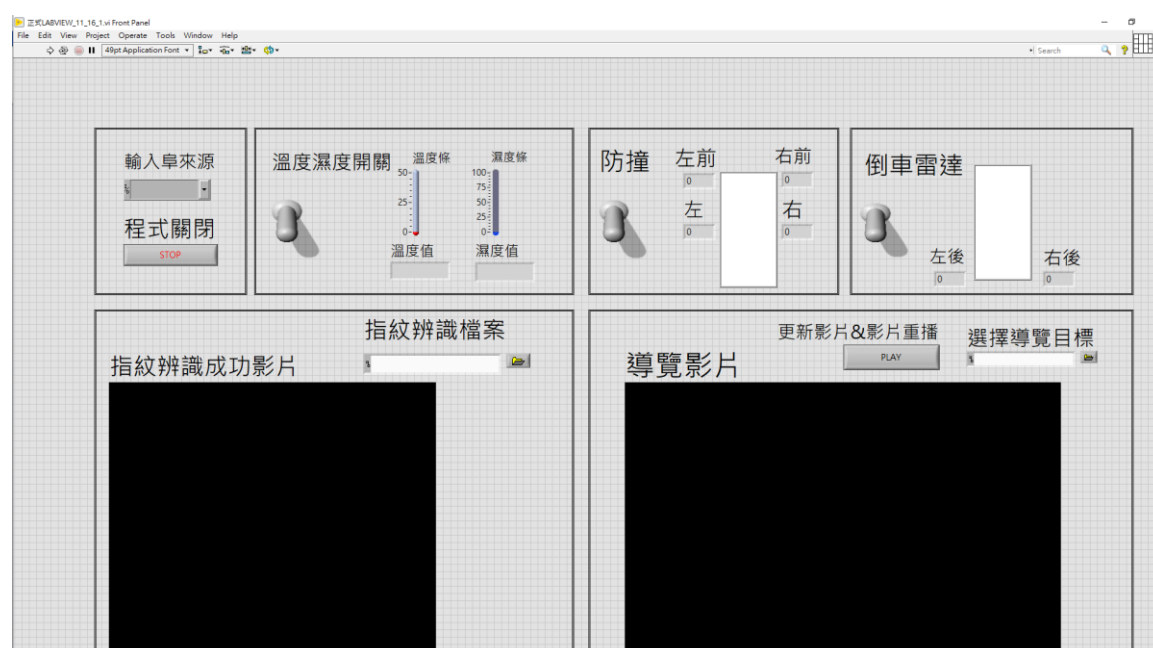


圖 4-2 LabVIEW 成果

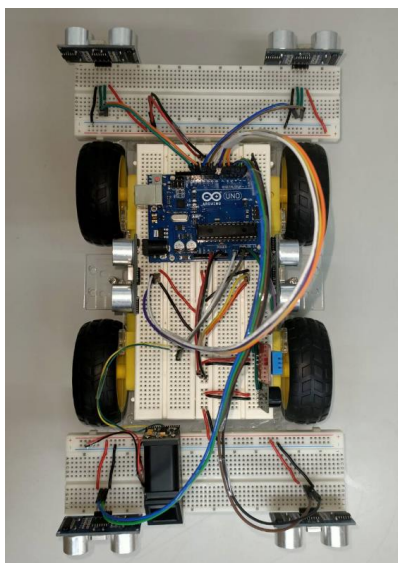


圖 4-3 實體成果

4.1.1 智能導覽車的操作

首先要先把 Arduino IDE 的統合程式(如 7.4.9)燒入到 Arduino Uno 板子裡面，然後再開啓 LabVIEW 設計好的檔案，開啟後就會有如圖 4-2 顯示器上的一樣的畫面。顯示器上內容包含防撞數值顯示、導覽功能、溫度濕度顯示、指紋辨識、倒車雷達的距離和警示聲。

4.1.2 顯示器功能介紹

1.防撞數值顯示：顯示器上會顯示出車體左側、右側、左前和右前與障礙物的距離值，駕駛人可以觀察此數據，在過彎、路邊停車、兩輛車會車時，防止擦撞意外的發生。

2.導覽功能：點擊後，顯示器會顯示一個畫面，以及包含有各個比

較重要的景點或建築物的導覽影音檔，例如：銘傳大學—科技大樓、資訊大樓、圖書館……。

3.溫度濕度：點擊後，顯示器會顯示出當前導覽車周圍環境的溼度以及溫度。

4.指紋辨識：駕駛將手指伸去指紋辨識器(FPM10A)，辨識成功後便會解鎖顯示器上的其他功能。

5.倒車雷達距離：點擊後，將會顯示模擬車體後方距離最近物體的實際距離數值，並且在各個距離會發出不同種音效。

4.1.3 模擬車構造介紹

模擬車在車頭左前角、右前角、車身左邊和右邊、車尾左後角、右後角都裝上了超音波感測器，車身中間裝置了 Arduino UNO 版、溫度濕度感測器、有源蜂鳴器和指紋辨識器。硬體接線如圖 4-3。

4.1.4 成果執行展示

圖 4-4 中的 01、02、03、04，分別是 01 溫溼度感測數據、02 倒車雷達數據、03 指紋辨識數據、04 防撞四角。其中的英文字母是為了方便 LabVIEW 去分割數據字串(例如：圖 4-4 中 a24i84，LabVIEW 可以分成 24 和 84 分別顯示)。

圖 4-5 為 LabVIEW 執行中的樣子，全部功能都能同時進行。

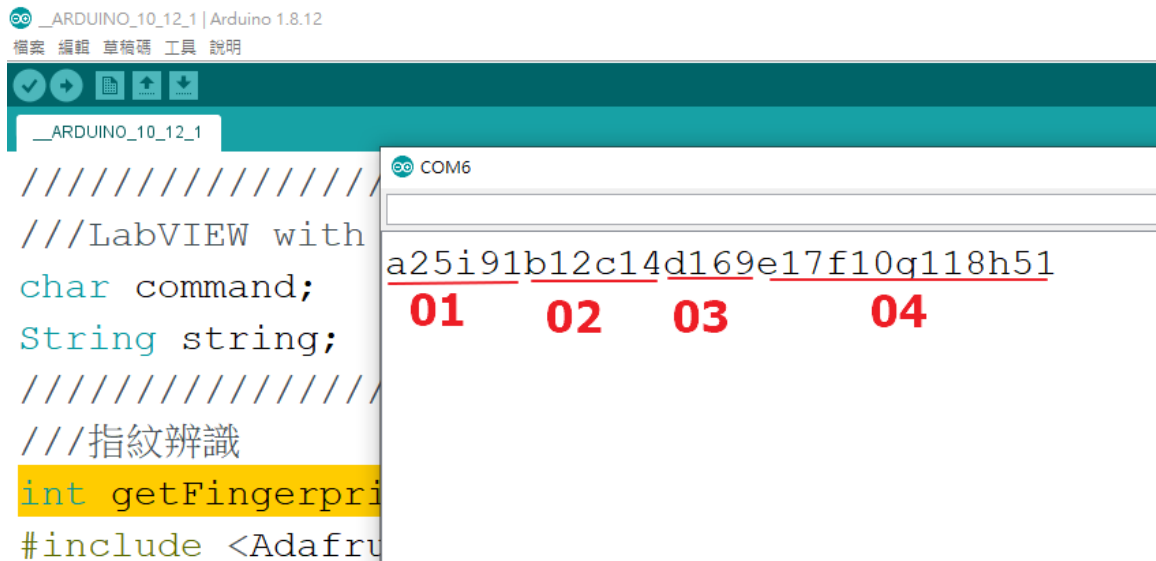


圖 4-4 Arduino IDE 序列埠監控執行成果



圖 4-5 LabVIEW 使用者介面成果

4.2 未來展望

4.2.1 運用範圍

此設計可以運用在觀光解說的部分，運用的時機像是：「觀光景點、觀光工廠、公司向客戶介紹自己的公司的時候、校園介紹……」。

4.2.2 優勢

在少子化的這世代，人力成本只會越來越高。此設計可以讓原本需要 10 幾個人力工作縮減成 2 人左右就可以了，只需要負責租借導覽車的使用權即可，以此來減少人力成本，以及機器可以完整清楚地介紹當地環境，可以避免有些時候解說人員的口誤或著是忘詞之類的，雖然少了人員介紹的人情味，但觀賞者可以比較自由自在的自己去了解，不會有麻煩到別人的感覺。

4.2.3 未來發展展望

在未來，希望此設計可以朝著以下幾 5 點發展：

1. 任何人都能更輕鬆使用的客製化設計。
2. 能夠以更簡略的方式生產智能導覽車。

3. 使智能導覽車的導覽功能設備更佳的完備。
4. 降低成本，使一般小型的觀光景點都能使用。
5. 網路化發展，使得人力成本為 0，只需靠網路就可申請使用權。

工作分配

表 5-1 工作分配

章節	標題	負責人	進度
摘要	X	劉懿葭	6 月完成
英文摘要	X	共同	6 月完成
第一章 簡介	1.1 研究動機與目的	陳彥邦	4 月完成
	1.2 文獻探討	劉懿葭	5 月完成
	1.3 專題架構	陳彥邦	4 月完成
	1.4 章節簡介	陳彥邦	5 月完成
第二章 材料介紹	2.1.1 DHT11 溫度濕度感測	劉懿葭	4 月完成
	2.1.2 Arduino UNO R3 開發版	徐琦喻	4 月完成
	2.1.3 HC-SR04	徐琦喻	4 月完成

	超音波感測器		
	2.1.4 FPM10A 指紋辨識	劉懿葭	5 月完成
	2.1.5 有源蜂鳴器	徐琦喻	5 月完成
	2.1.6 Arduino 模型車	陳彥邦	5 月完成
	2.2.1 Arduino 平台	徐琦喻	5 月完成
	2.2.2 LabVIEW 平台	陳彥邦	5 月完成
第三章 Intelligent guide car 設計與操作	3.1 溫度濕度 (DHT11)	劉懿葭	4 月完成
	3.2 倒車雷達 (HC-SR04)	徐琦喻	5 月完成

	3.3 防撞功能		
	3.4 指紋辨識解鎖	劉懿葭	7 月完成
	3.5 連結 LabVIEW 的 Arduino IDE 統合程式	陳彥邦	9 月完工
	3.6 LabVIEW 軟體設 計	陳彥邦	9 月完成
第四章 成果與 未來展望	4.1 成果	陳彥邦	10 月完成
	4.2 未來展望	陳彥邦	6 月完成
工作分配	X	陳彥邦	5 月完成
專題進度	X	陳彥邦	10 月完成
附錄	X	共同	5 月完成
參考文獻	X	共同	10 月完成

專題進度

表 6-1 專題進度表

年份	月份	工作進度
2020 年	1 月	討論整理研究整體架構&功能
	2、3 月	購買材料&一些簡單功能(DHT11 溫度濕度、HC-SR04 超音波感測、蜂鳴器)的測試。
	4 月	完成基本的專題初審 word 檔、了解 webcam 網路攝影機如何運作、模型車的組裝。
	5 月	完成整體的專題初審 word 檔。
	6 月	初審 word 檔給老師最後的 check 和簽名、交出初審報告、熟悉 LabVIEW & Arduino 語法。
	7、8 月	完成所有單項功能、更改初審 word 檔
	9 、 10 月	統合所有功能到模擬車上、做出總審 word 檔、做出總審報告 ppt 檔、最後結果的檢查

附錄

7.1 材料規格

表 7-1 Arduino Uno R3 規格

項目	內容
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14(of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40mA
DC Current for 3.3V Pin	50mA
Flash Memory	32 KB(ATmega328) of which 0.5 KB used by bootloader
SRAM	2KB(ATmega328)
EEPROM	1 KB(ATmega328)

Clock Speed	16MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

表 7-2 HC-SR04 規格

元件名稱	HC-SR04
Working Voltage	DC 5V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
Measuring Angle	15degree
Trigger Input Signal	10usTTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45mm x 20mm x 15mm

表 7-3 FPM10A 光學指紋辨識規格

項目	內容
供電電壓	DC 3.6~6.0V
工作電流	<120mA
峰值電流	<140mA
指紋圖像錄入時間	<1.0 秒
窗口面積	14×18 mm
特徵檔	256 位元組
範本檔	512 位元組
存儲容量	1000 枚
認假率(FAR)	<0.001% (安全等級為 3 時)
拒真率(FRR)	<1.0% (安全等級為 3 時)
搜索時間	<1.0 秒 (1:500 時，均值)
上位機介面	UART (TTL 邏輯電平)
通訊串列傳輸速率(UART)	(9600×N)bps 其中 N=1~12 (預設值 N=6，即 57600bps)
工作環境(溫度)	-20℃ 至+50℃
相對濕度	40%RH 至 85%RH(無凝露)
儲存環境(溫度)	-40℃ 至+85℃

外形尺寸(L×W×H)	56×20×21.5mm
-------------	--------------

7.2 Arduino 軟體下載

第一步:到 <https://www.arduino.cc/en/Main/Software> 下載最新版 Arduino IDE，依照電腦的作業系統選擇安裝程式。(如圖 7-1)

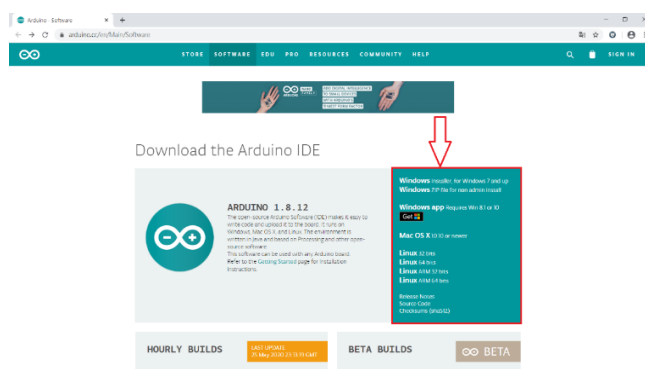


圖 7-1 Arduino 軟體下載-1

第二步:進入頁面後，點選「JUST DOWNLOAD」。(如圖 7-2)

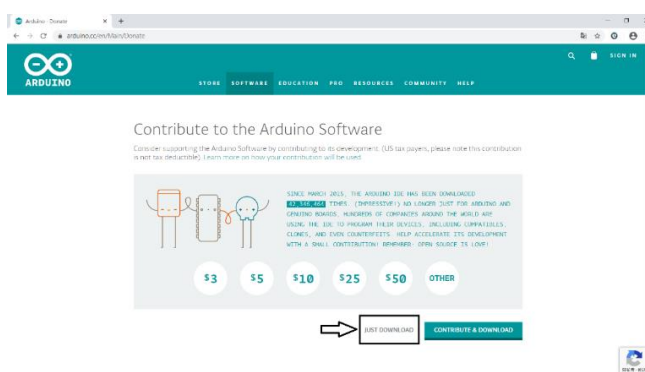


圖 7-2 Arduino 軟體下載-2

第三步:點選「I Agree」，同意版權宣告。(如圖 7-3)

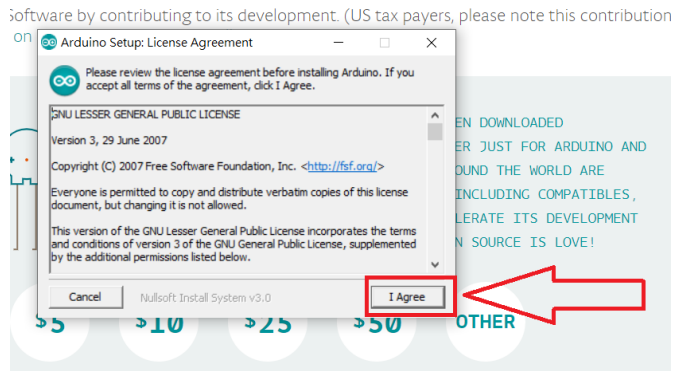


圖 7-3 Arduino 軟體下載-3

第四步:點選要安裝的元件,可以依照預設值全部安裝,點選「Next」。

(如圖 7-4)

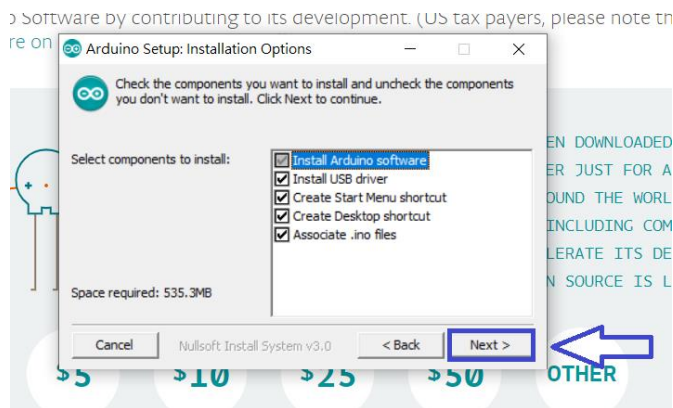


圖 7-4 Arduino 軟體下載-4

第五步:選擇安裝的資料夾,點選「Browse...」可以重新選擇安裝資

料夾,點選「Install」安裝 Arduino IDE。(如圖 7-5)

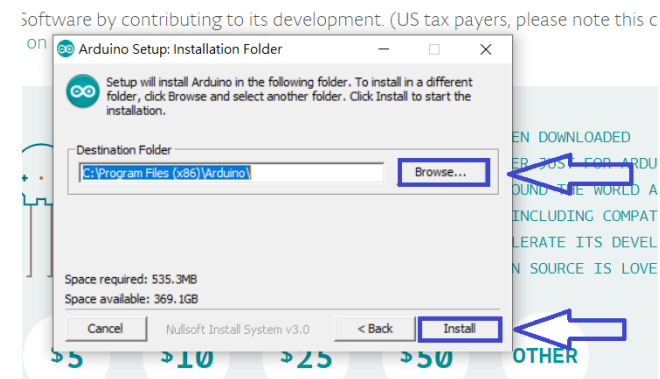


圖 7-5 Arduino 軟體下載-5

第六步:出現畫面表示安裝完成，點選「Close」。(如圖 7-6)

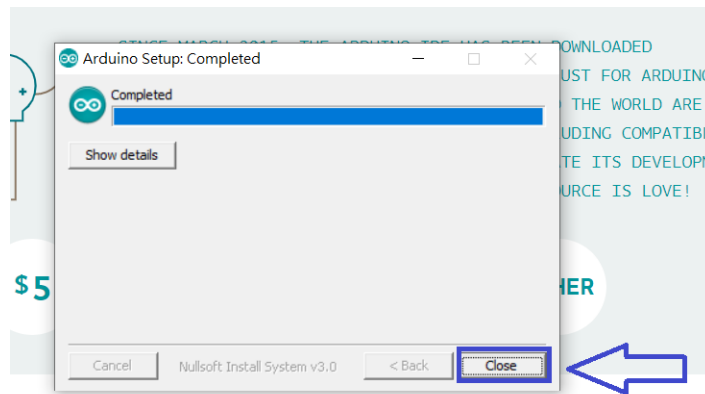


圖 7-6 Arduino 軟體下載-6

7.2.1 Arduino 軟體介面

並能在 Windows、Linux、Mac OS 上運行。開發程式通常採用免費、開放原始碼，是具有類似 java、C 語言的開發環境，軟體介面介紹如(如圖 7-7 圖 7-8)

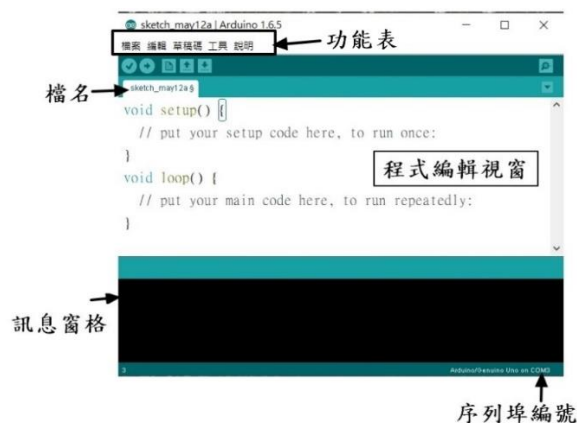


圖 7-7 Arduino 程式編輯視窗

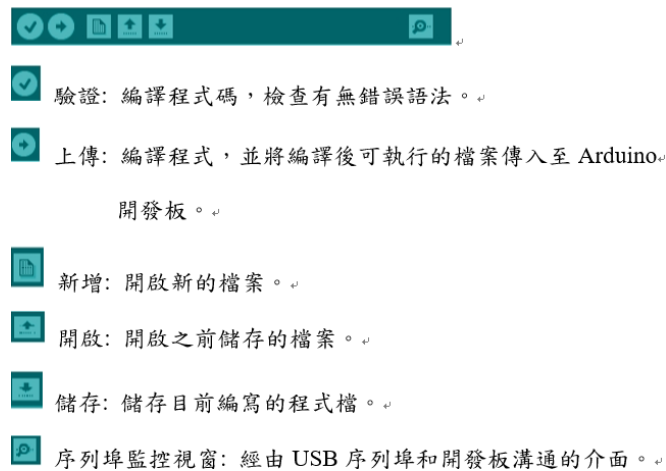


圖 7-8 Arduino 程式編輯視窗功能列

初次使用 Arduino 程式開發工具，首先將開發板與電腦USB連接，接著在功能表工具設定 Arduino 板子類型(圖 7-9)和序列埠編號(圖 7-10)。1.先點選工具、2.選擇板子:” Arduino/Genuino Uno”、3.點選 Arduino/Genuino Uno、4.再次點選工具、5.選擇序列埠:” COM3(Arduino/Genuino Uno)”、6.點選 COM3(Arduino/Genuino Uno)。

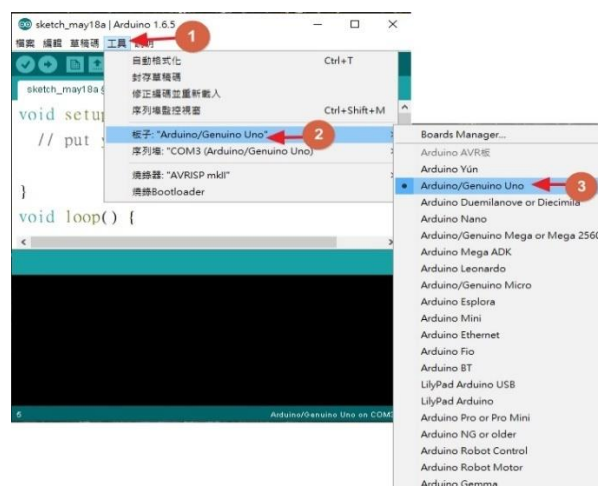


圖 7-9 Arduino 初步設定 1

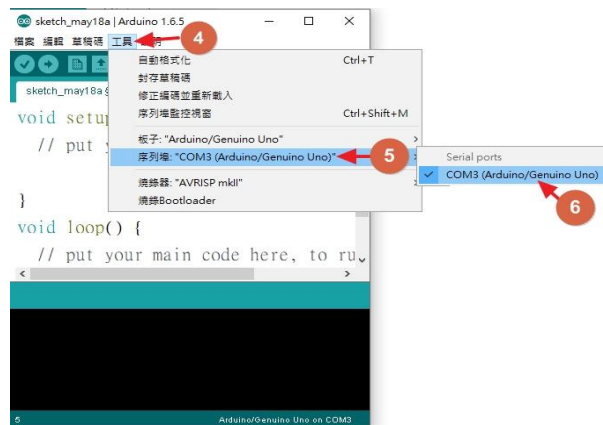


圖 7-10 Arduino 初步設定 2

設定完成之後，將編寫好的程式利用 USB 上傳至開發板，即可操作、控制各式與板子連接的電子元件與感測器（例如：超音波模組、溫溼度感測器、紅外線感測、LED、光敏電阻等等）。

7.3 LabVIEW 軟體下載

Step 1：

Google 搜”下載 LabVIEW” 找到(圖 7-11)。



圖 7-11 LabVIEW 軟體下載-1

Step 2：

選擇下載免費體驗版。

All LabVIEW editions are available in English, French, German, Korean, Japanese, and Simplified Chinese.

	LabVIEW NXG Base	LabVIEW NXG Full	LabVIEW NXG Professional
Starting from	TWD 15,600.00/year	TWD 117,700.00	TWD 196,200.00
Select edition	SELECT	SELECT	SELECT
免費體驗版	—	—	免費體驗版
主要差異	<ul style="list-style-type: none"> 建議用於桌上型量測應用 包含適用於 NI 硬體與第三方儀器的裝置驅動程式 包含基本的數學與訊號處理功能 	<ul style="list-style-type: none"> 建議用於執行電腦的行內數學與訊號處理 包含 LabVIEW NXG 基本版功能 	<ul style="list-style-type: none"> 建議用於部署應用 包含 LabVIEW NXG 完整版功能
LabVIEW 2019 基本版	LabVIEW 2019 基本版	LabVIEW 2019 完整版	LabVIEW 2019 專業版


圖 7-12 LabVIEW 軟體下載-2

Step 3：

登入 OR 創建帳號。

NI User Account

Log In



Email

Password [Forgot Password?](#)

☐ Stay logged in

[LOG IN](#)

[Create Account >](#)

圖 7-13 LabVIEW 軟體下載-3

Step 4：

登入帳號後就會跑出下載的畫面。

NATIONAL INSTRUMENTS

INNOVATIONS PRODUCTS SUPPORT COMMUNITY

Downloading LabVIEW 2019 and Drivers

Next Steps

Your downloaded software is delivered using NI's Package Manager. After the download is complete, NI Package Manager will automatically install the software. If the download doesn't start automatically, restart the download now.

GETTING STARTED

Register Your Software

Activate a License

Automate Software Installation on Multiple Computers

圖 7-14 LabVIEW 軟體下載-4

Step 5：

完成 Step 4 之後會跑出一個畫面，並且把(圖 7-15 圖 7-16 圖 7-17)

所有有打勾的地方都打勾。

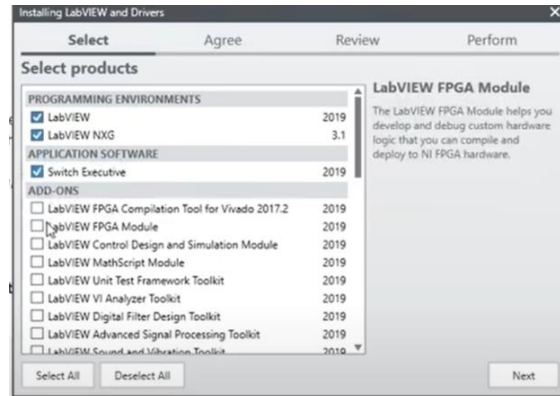


圖 7-15 LabVIEW 軟體下載-5.1

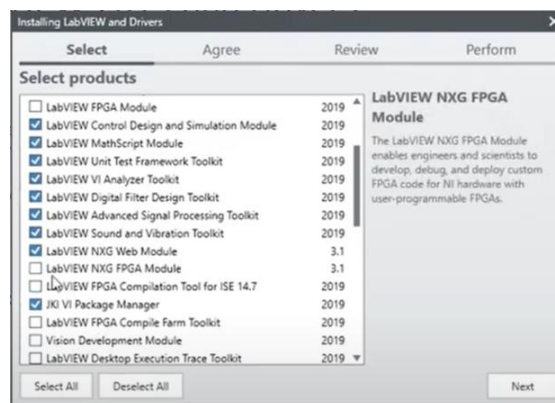


圖 7-16 LabVIEW 軟體下載-5.2

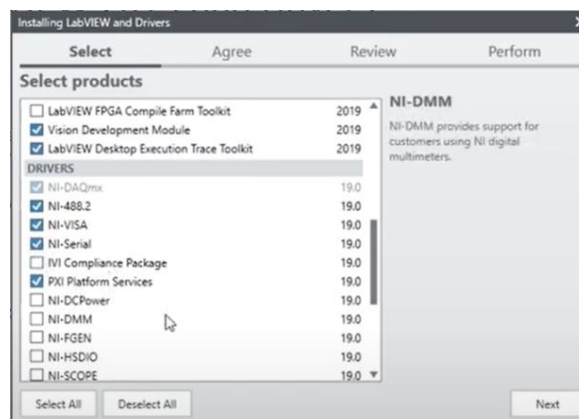


圖 7-17 LabVIEW 軟體下載-5.3

Step 6 :

之後就一直按 Next 和 Agree，直到跑出下載中的(圖 7-18)畫面。下載完後就可以從打開 LabVIEW 的檔案了。

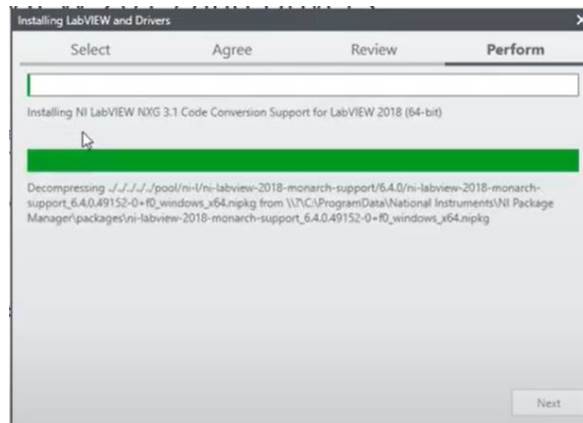


圖 7-18 LabVIEW 軟體下載-6

7.4 Arduino 程式設計

7.4.1 DHT11 溫度濕度步驟

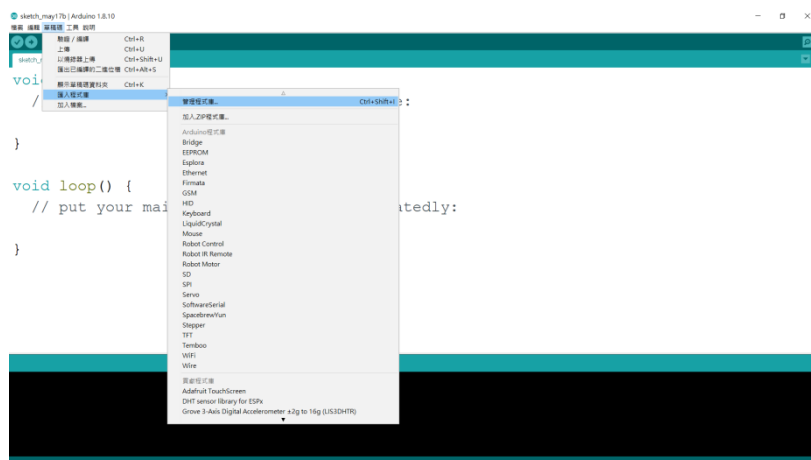


圖 7-19 DHT11 程式步驟 1

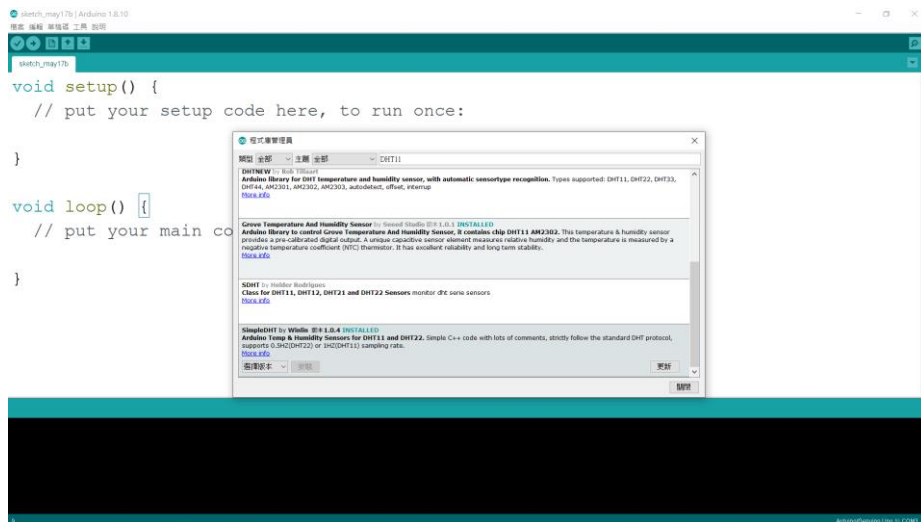


圖 7-20 DHT11 程式步驟 2

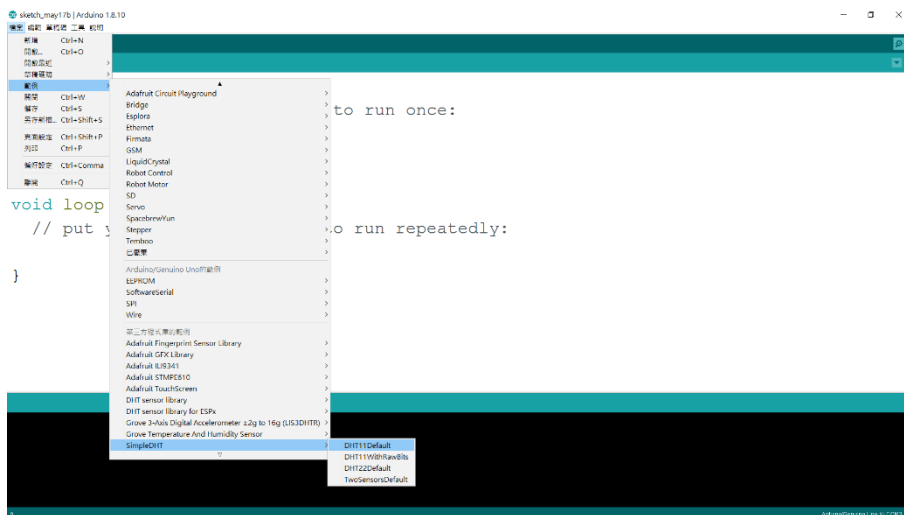


圖 7-21 DHT11 程式步驟 3-1

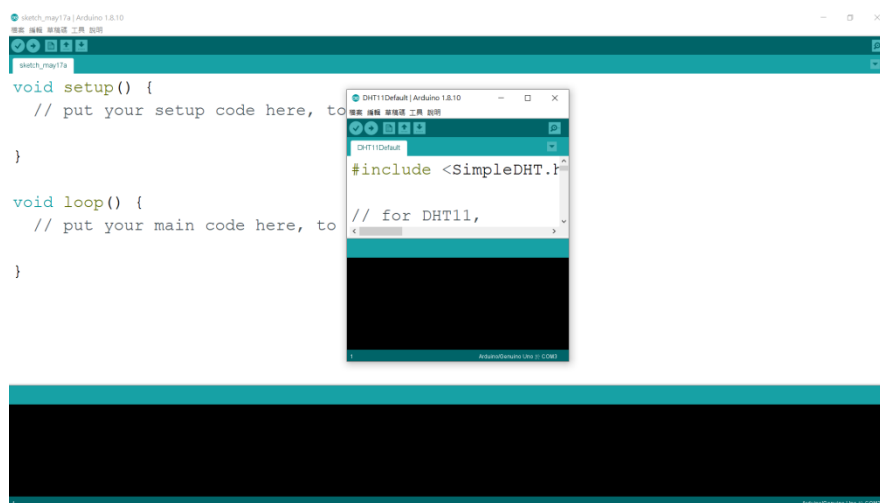


圖 7-22 DHT11 程式步驟 3-2

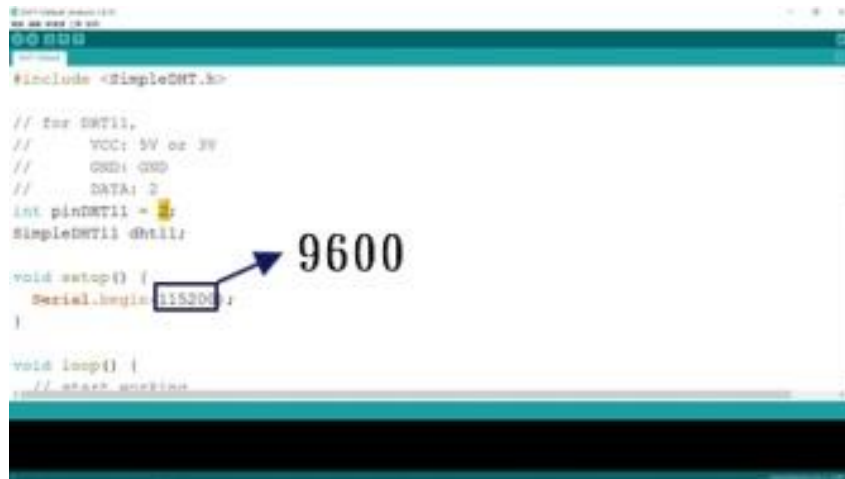


圖 7-23 DHT11 程式步驟 4

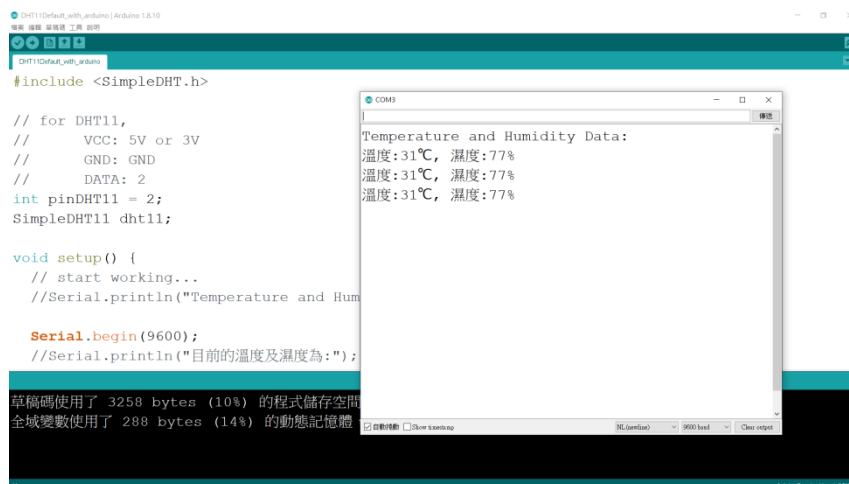


圖 7-24 DHT11 測試結果

7.4.2 DHT11 溫度濕度感測程式

```

#include <SimpleDHT.h>

// for DHT11,
//   VCC: 5V or 3V
//   GND: GND
//   DATA: 2
int pinDHT11 = 2;
SimpleDHT11 dht11;

void setup() {
  // 開始作業...

```

```

//印出("目前的溫濕度為")
Serial.begin(9600);
//印出("目前的溫度及濕度為:");
Serial.println("溫濕度數據:");
}
void loop() {
  delay(250);
  // read without samples.
  byte temperature = 0;
  byte humidity = 0;
  int err = SimpleDHTErrSuccess;
  if ((err = dht11.read(pinDHT11, &temperature, &humidity, NULL)) != SimpleDHTErrSuccess) {
    Serial.print("讀取 DHT11 失敗，錯誤=="); Serial.print(err);
    return;
  }
  Serial.print("溫度:"); Serial.print((int)temperature); Serial.print("度 C, ");
  if (temperature>33){
    Serial.print("溫度過高! ");
  }
  Serial.print("濕度:");Serial.print((int)humidity); Serial.println("% ");
  // DHT11 採樣率為 1HZ
  delay(3000);
}

```

7.4.3 倒車雷達程式

```

int spk=13;           //定義蜂鳴器為數位接腳 13
int trigPinRL =12;    //定義(左後 HC-SR04)Trig Pin(超音波模組觸發腳) -接腳 12
int echoPinRL =11;    //定義(左後 HC-SR04)Echo Pin(超音波模組接收腳) -接腳 11
int trigPinRR =10;    //定義(右後 HC-SR04)Trig Pin(超音波模組觸發腳) -接腳 10
int echoPinRR =9;     //定義(右後 HC-SR04)Echo Pin(超音波模組接收腳) -接腳 9
long durationRL;      //宣告 durationRL
long durationRR;      //宣告 durationRR
float distanceRL;      //宣告 distanceRL
float distanceRR;      //宣告 distanceRR
int s=100;            //基準時間
void lndash();         //宣告長劃訊號

```

```

void dash();           //宣告劃訊號
void dot();            //宣告點訊號
void sdot();           //宣告短點訊號
void setup() {
    Serial.begin (9600);           //Serial Port begin 初始化序列埠
    pinMode(trigPinRL, OUTPUT);    //trigPinRL，輸出
    pinMode(echoPinRL, INPUT);     //echoPinRL，輸入
    pinMode(trigPinRR, OUTPUT);    //trigPinRR，輸出
    pinMode(echoPinRR, INPUT);     //echoPinRR，輸入
    pinMode(spkr, OUTPUT);         //spkr，輸出
}
void loop()
{
    digitalWrite(trigPinRL, LOW);   //給 Trig 低電位
    delayMicroseconds(2);           //持續 2 微秒
    digitalWrite(trigPinRL, HIGH);  //給 Trig 高電位
    delayMicroseconds(10);          //持續 10 微秒
    digitalWrite(trigPinRL, LOW);   //給 Trig 低電位
    pinMode(echoPinRL, INPUT);       //讀取 echo 的電位
    durationRL = pulseIn(echoPinRL, HIGH); //收到高電位時的時間
    distanceRL = durationRL*0.034/2; //將時間換算成距離 cm
                                     //cm = (duration/2) / 29.1

    Serial.print("distanceRL: ");   //Serial.print(distanceRL:)
    Serial.print(distanceRL );      //Serial.print(實測距離)
    Serial.println(" cm ");         //Serial.print(cm);
    delay(500);                     //等待 0.5 秒鐘(0.5 秒測一次)
    ///////////////////////////////////////////////////////////////////

    digitalWrite(trigPinRR, LOW);   //給 Trig 低電位
    delayMicroseconds(2);           //持續 2 微秒
    digitalWrite(trigPinRR, HIGH);  //給 Trig 高電位
    delayMicroseconds(10);          //持續 10 微秒
    digitalWrite(trigPinRR, LOW);   //給 Trig 低電位
    pinMode(echoPinRR, INPUT);       //讀取 echo 的電位
    durationRR = pulseIn(echoPinRR, HIGH); //收到高電位時的時間
    distanceRR = durationRR*0.034/2; //將時間換算成距離 cm
                                     //cm = (duration/2) / 29.1

    Serial.print("distanceRR: ");   //Serial.print(distanceRR:)
    Serial.print(distanceRR );      //Serial.print(實測距離)
}

```

```

Serial.println(" cm "); //Serial.print(cm);
delay(500); //等待 0.5 秒鐘(0.5 秒測一次)
if(distanceRL>distanceRR){
    distanceRL = distanceRR ;
}
else if(distanceRL<distanceRR){
    distanceRR = distanceRL ;
}
if( distanceRL >= 80 && distanceRL <=100 || distanceRR >= 80 && distanceRR <=100){
    lndash();
} //假設距離介於 80~100 之間蜂鳴器聲音為長劃訊號
else if (distanceRL >=60 && distanceRL <80 || distanceRR >=60 && distanceRR <80){
    dash();
} //假設距離介於 60~79 之間蜂鳴器聲音為劃訊號
else if (distanceRL >= 40 && distanceRL <60 || distanceRR >= 40 && distanceRR <60){
    dot();
} //假設距離介於 40~59 之間蜂鳴器聲音為點訊號
else if (distanceRL >= 30 && distanceRL <40 || distanceRR >= 30 && distanceRR <40){
    sdot(); sdot();
} //假設距離介於 30~39 之間蜂鳴器聲音為兩個短點訊號
else if (distanceRL >= 20 && distanceRL <30 || distanceRR >= 20 && distanceRR <30){
    sdot(); sdot(); sdot(); sdot();
} //假設距離介於 20~29 之間蜂鳴器聲音為四個短點訊號
else if (distanceRL <20 || distanceRR <20){
    digitalWrite(spkr, HIGH);
    delay(10);
} //假設距離小於 19 公分，蜂鳴器持續發出聲響
else
    digitalWrite(spkr, LOW);
} //超過 100 公分以上蜂鳴器不發出聲響
void lndash(){ //設定長劃訊號
    digitalWrite(spkr,HIGH); //蜂鳴器響 4 秒
    delay(s*4);
    digitalWrite(spkr,LOW);
}
void dash(){ //設定劃訊號
    digitalWrite(spkr,HIGH); //蜂鳴器響 2.5 秒
    delay(s*2.5);

```

```

    digitalWrite(spk,LOW);
}
void dot(){
    //設定點訊號
    digitalWrite(spk,HIGH); //蜂鳴器響 1.5 秒
    delay(s*1.5);
    digitalWrite(spk,LOW);
}
void sdot(){
    //設定短點訊號
    digitalWrite(spk,HIGH); //蜂鳴器響 1 秒，停頓 0.3 秒
    delay(s);
    digitalWrite(spk,LOW);
    delay(s*0.3);
}

```

7.4.4 防撞功能程式

```

int trigPinFL =8;           //定義(左前 HC-SR04)Trig Pin(超音波模組觸發腳) -接腳 8
int echoPinFL =7;          //定義(左前 HC-SR04)Echo Pin(超音波模組接收腳) -接腳 7
int trigPinFR =6;          //定義(右前 HC-SR04)Trig Pin(超音波模組觸發腳) -接腳 6
int echoPinFR =5;          //定義(右前 HC-SR04)Echo Pin(超音波模組接收腳) -接腳 5
int trigPinL =4;           //定義(左 HC-SR04)Trig Pin(超音波模組觸發腳) -接腳 4
int echoPinL =3;           //定義(左 HC-SR04)Echo Pin(超音波模組接收腳) -接腳 3
int trigPinR =A0;          //定義(右 HC-SR04)Trig Pin(超音波模組觸發腳) -接腳 A0
int echoPinR =A1;          //定義(右 HC-SR04)Echo Pin(超音波模組接收腳) -接腳 A1
long durationFL;           //宣告 durationFL
long durationFR;           //宣告 durationFR
float distanceFL;          //宣告 distanceFL
float distanceFR;          //宣告 distanceFR
long durationL;            //宣告 durationL
long durationR;            //宣告 durationR
float distanceL;           //宣告 distanceL
float distanceR;           //宣告 distanceR
void setup() {
    Serial.begin (9600);    //Serial Port begin 初始化序列埠
    pinMode(trigPinFL, OUTPUT); //trigPinFL，輸出
    pinMode(echoPinFL, INPUT);  //echoPinFL，輸入
    pinMode(trigPinFR, OUTPUT); //trigPinFR，輸出
}

```

```

pinMode(echoPinFR, INPUT);           //echoPinFR，輸入
pinMode(trigPinL, OUTPUT);           //trigPinL，輸出
pinMode(echoPinL, INPUT);            //echoPinL，輸入
pinMode(trigPinR, OUTPUT);           //trigPinR，輸出
pinMode(echoPinR, INPUT);            //echoPinR，輸入
}
void loop()
{
    digitalWrite(trigPinFL, LOW);      //給 Trig 低電位
    delayMicroseconds(2);              //持續 2 微秒
    digitalWrite(trigPinFL, HIGH);     //給 Trig 高電位
    delayMicroseconds(10);             //持續 10 微秒
    digitalWrite(trigPinFL, LOW);      //給 Trig 低電位
    pinMode(echoPinFL, INPUT);         //讀取 echo 的電位
    durationFL = pulseIn(echoPinFL, HIGH); //收到高電位時的時間
    distanceFL = durationFL*0.034/2;    //將時間換算成距離 cm
                                         //cm = (duration/2) / 29.1
    Serial.print("distanceFL: ");      //Serial.print(distanceFL:)
    Serial.print(distanceFL );         //Serial.print(實測距離)
    Serial.println(" cm ");           //Serial.print(cm);
    delay(500);                       //等待 0.5 秒鐘(0.5 秒測一次)
    //////////////////////////////////////
    digitalWrite(trigPinFR, LOW);      //給 Trig 低電位
    delayMicroseconds(2);              //持續 2 微秒
    digitalWrite(trigPinFR, HIGH);     //給 Trig 高電位
    delayMicroseconds(10);             //持續 10 微秒
    digitalWrite(trigPinFR, LOW);      //給 Trig 低電位
    pinMode(echoPinFR, INPUT);         //讀取 echo 的電位
    durationFR = pulseIn(echoPinFR, HIGH); //收到高電位時的時間
    distanceFR = durationFR*0.034/2;    //將時間換算成距離 cm
                                         //cm = (duration/2) / 29.1
    Serial.print("distanceFR: ");      //Serial.print(distanceFR:)
    Serial.print(distanceFR );         //Serial.print(實測距離)
    Serial.println(" cm ");           //Serial.print(cm);
    delay(500);                       //等待 0.5 秒鐘(0.5 秒測一次)
    //////////////////////////////////////
    digitalWrite(trigPinL, LOW);      //給 Trig 低電位
    delayMicroseconds(2);              //持續 2 微秒

```

```

digitalWrite(trigPinL, HIGH);           //給 Trig 高電位
delayMicroseconds(10);                 //持續 10 微秒
digitalWrite(trigPinL, LOW);           //給 Trig 低電位
pinMode(echoPinL, INPUT);              //讀取 echo 的電位
durationL = pulseIn(echoPinL, HIGH);   //收到高電位時的時間
distanceL = durationL*0.034/2;         //將時間換算成距離 cm
                                        //cm = (duration/2) / 29.1

Serial.print("distanceL: ");           //Serial.print(distanceFL:)
Serial.print(distanceL );              //Serial.print(實測距離)
Serial.println(" cm ");               //Serial.print(cm);
delay(500);                           //等待 0.5 秒鐘(0.5 秒測一次)
////////////////////////////////////
digitalWrite(trigPinR, LOW);           //給 Trig 低電位
delayMicroseconds(2);                 //持續 2 微秒
digitalWrite(trigPinR, HIGH);         //給 Trig 高電位
delayMicroseconds(10);               //持續 10 微秒
digitalWrite(trigPinR, LOW);          //給 Trig 低電位
pinMode(echoPinR, INPUT);             //讀取 echo 的電位
durationR = pulseIn(echoPinR, HIGH);  //收到高電位時的時間
distanceR = durationR*0.034/2;        //將時間換算成距離 cm
                                        //cm = (duration/2) / 29.1

Serial.print("distanceR: ");           //Serial.print(distanceFR:)
Serial.print(distanceR );              //Serial.print(實測距離)
Serial.println(" cm ");               //Serial.print(cm);
delay(500);                           //等待 0.5 秒鐘(0.5 秒測一次)
}

```

7.4.5 超音波元件測試程式

```

int trigPin =12;                      //Trig Pin(超音波模組觸發腳)-接腳 12
int echoPin = 11;                     //Echo Pin(超音波模組接收腳)-接腳 11
long duration;                        //宣告 duration
float distanceCM;                     //宣告 distanceCM
int s=100;                            //基準時間
void setup() {
    Serial.begin (9600);               // Serial Port begin 初始化序列埠
    pinMode(trigPin, OUTPUT);          // trigPin, 輸出

```

```

    pinMode(echoPin, INPUT);          //echoPin, 輸入
}
void loop()
{
    digitalWrite(trigPin, LOW);        //給 Trig 低電位
    delayMicroseconds(2);              //持續 2 微秒
    digitalWrite(trigPin, HIGH);       //給 Trig 高電位
    delayMicroseconds(10);             //持續 10 微秒
    digitalWrite(trigPin, LOW);        //給 Trig 低電位
    pinMode(echoPin, INPUT);           //讀取 echo 的電位
    duration = pulseIn(echoPin, HIGH); //收到高電位時的時間
    Serial.print("EchoPin duration:");
    Serial.print(duration);            //Serial.print(實測 Echo 腳位呈現高電位的時間)
    Serial.print(" microsecond ");
    distanceCM =duration*0.034/2;      //將時間換算成距離 cm
    Serial.print("distance: ");        //Serial.print(distance:)
    Serial.print(distanceCM );         //Serial.print(實測距離)
    Serial.println(" cm ");            //Serial.print(cm);
    delay(1000);                       //等待一秒鐘(一秒測一次)
}

```

7.4.6 指紋辨識步驟

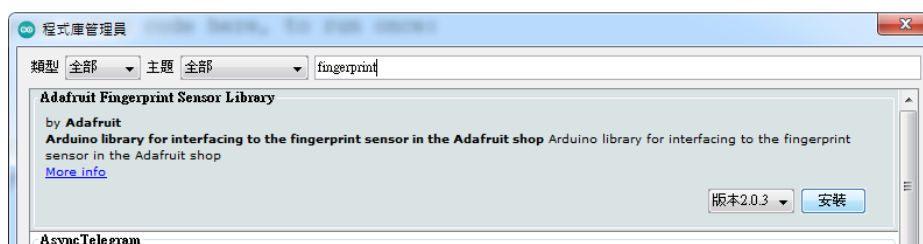


圖 7-25 FPM10A 程式步驟 1

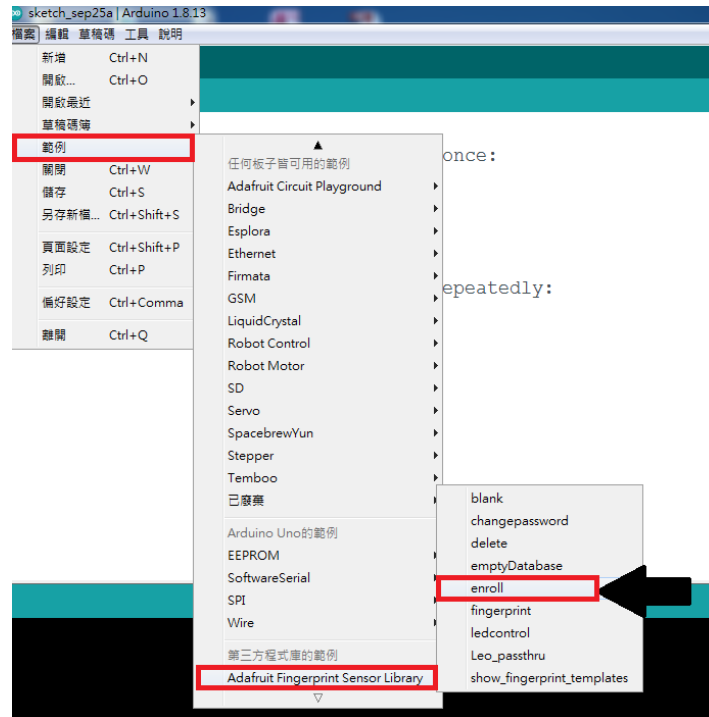


圖 7-26 FPM10A 程式步驟 2

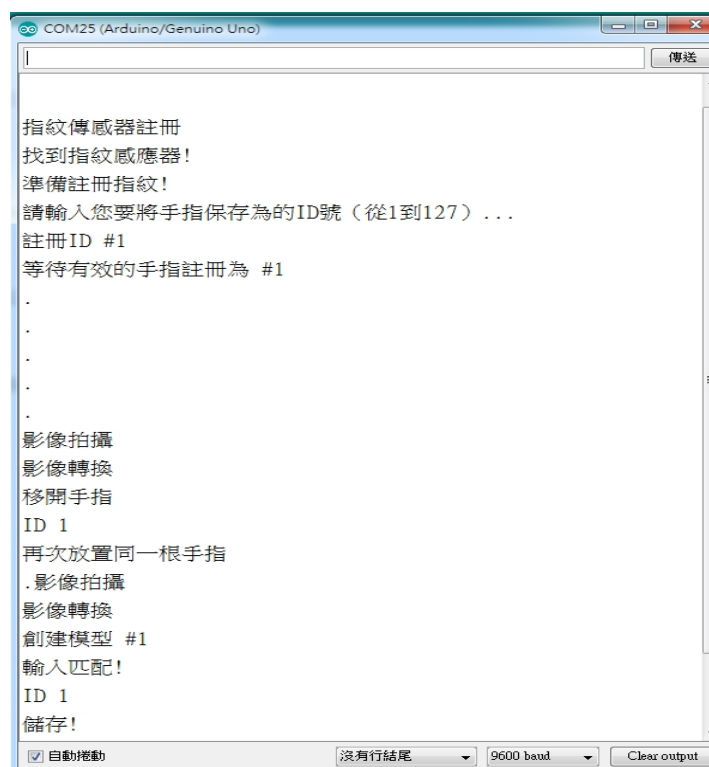


圖 7-27 FPM10A 程式步驟 4



圖 7-28 FPM10A 測試結果

7.4.7 指紋註冊程式

```
#include <Adafruit_Fingerprint.h>

// 對於 UNO 和其他沒有硬件序列的用戶，我們必須使用軟件序列...
// 接腳 2 從傳感器（綠線）進入
// 接腳 3 從 arduino（黃線）輸出
// 設置串行端口以使用軟件串行。
SoftwareSerial mySerial(2, 3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
uint8_t id;
void setup()
{
  Serial.begin(9600);
  while (!Serial); // For Yun/Leo/Micro/Zero/...
  delay(100);
  Serial.println("\n\n 指紋傳感器註冊");
  //設置傳感器串行端口的數據速率
  finger.begin(57600);

  if (finger.verifyPassword()) {
```

```

        Serial.println("找到指紋感應器!");
    } else {
        Serial.println("沒有找到指紋感應器 :(");
        while (1) { delay(1); }
    }
}

uint8_t readnumber(void) {
    uint8_t num = 0;
    while (num == 0) {
        while (! Serial.available());
        num = Serial.parseInt();
    }
    return num;
}

void loop()                                //一遍又一遍
{
    Serial.println("準備註冊指紋!");
    Serial.println("請輸入您要將手指保存為的 ID 號 (從 1 到 127) ...");
    id = readnumber();
    if (id == 0) { // ID #0 不允許，再試一次!
        return;
    }
    Serial.print("註冊 ID #");
    Serial.println(id);
    while (! getFingerprintEnroll() );
}

uint8_t getFingerprintEnroll() {
    int p = -1;
    Serial.print("等待有效的手指註冊為 #"); Serial.println(id);
    while (p != FINGERPRINT_OK) {
        p = finger.getImage();
        switch (p) {
            case FINGERPRINT_OK:
                Serial.println("影像拍攝");
                break;
            case FINGERPRINT_NOFINGER:
                Serial.println(".");
                break;

```

```

case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("通信故障");
    break;
case FINGERPRINT_IMAGEFAIL:
    Serial.println("影像錯誤");
    break;
default:
    Serial.println("未知錯誤");
    break;
}
} // 確認成功!
p = finger.image2Tz(1);
switch (p) {
case FINGERPRINT_OK:
    Serial.println("影像轉換");
    break;
case FINGERPRINT_IMAGEMESS:
    Serial.println("影像太亂");
    return p;
case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("通信故障");
    return p;
case FINGERPRINT_FEATUREFAIL:
    Serial.println("找不到指紋特徵");
    return p;
case FINGERPRINT_INVALIDIMAGE:
    Serial.println("找不到指紋特徵");
    return p;
default:
    Serial.println("未知錯誤");
    return p;
}
Serial.println("移開手指");
delay(2000);
p = 0;
while (p != FINGERPRINT_NOFINGER) {
    p = finger.getImage();
}

```

```

Serial.print("ID "); Serial.println(id);
p = -1;
Serial.println("再次放置同一根手指");
while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
        Serial.println("影像拍攝");
        break;
    case FINGERPRINT_NOFINGER:
        Serial.print(".");
        break;
    case FINGERPRINT_PACKETRECEIVEERR:
        Serial.println("通信故障");
        break;
    case FINGERPRINT_IMAGEFAIL:
        Serial.println("影像錯誤");
        break;
    default:
        Serial.println("未知錯誤");
        break;
    }
} // 確認成功!
p = finger.image2Tz(2);
switch (p) {
    case FINGERPRINT_OK:
        Serial.println("影像轉換");
        break;
    case FINGERPRINT_IMAGEMESS:
        Serial.println("影像太亂");
        return p;
    case FINGERPRINT_PACKETRECEIVEERR:
        Serial.println("通信故障");
        return p;
    case FINGERPRINT_FEATUREFAIL:
        Serial.println("找不到指紋特徵");
        return p;
    case FINGERPRINT_INVALIDIMAGE:

```

```

        Serial.println("找不到指紋特徵");
        return p;
    default:
        Serial.println("未知錯誤");
        return p;
    }
}
//確認轉換!
Serial.print("創建模型 #"); Serial.println(id);
p = finger.createModel();
if (p == FINGERPRINT_OK) {
    Serial.println("輸入匹配!");
} else if (p == FINGERPRINT_PACKETRECEIVEERR) {
    Serial.println("通信故障");
    return p;
} else if (p == FINGERPRINT_ENROLLMISMATCH) {
    Serial.println("指紋辨識不匹配");
    return p;
} else {
    Serial.println("未知錯誤");
    return p;
}
Serial.print("ID "); Serial.println(id);
p = finger.storeModel(id);
if (p == FINGERPRINT_OK) {
    Serial.println("儲存!");
} else if (p == FINGERPRINT_PACKETRECEIVEERR) {
    Serial.println("通信故障");
    return p;
} else if (p == FINGERPRINT_BADLOCATION) {
    Serial.println("無法存儲在該位置");
    return p;
} else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("寫入 flash 時錯誤");
    return p;
} else {
    Serial.println("未知錯誤");
    return p;
}
}

```

```
}
```

7.4.8 指紋辨識程式

```
#include <Adafruit_Fingerprint.h>
// 對於 UNO 和其他沒有硬件序列的用戶，我們必須使用軟件序列...
// 接腳 2 從傳感器（綠線）進入
// 接腳 3 從 arduino（黃線）輸出
// 設置串行端口以使用軟件串行。
SoftwareSerial mySerial(2, 3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
void setup()
{
  Serial.begin(9600);
  while (!Serial); // For Yun/Leo/Micro/Zero/...
  delay(100);
  Serial.println("\n\n 指紋傳感器註冊");
  // 設置傳感器串行端口的數據速率
  finger.begin(57600);
  delay(5);
  if (finger.verifyPassword()) {
    Serial.println("找到指紋感應器!");
  } else {
    Serial.println("沒有找到指紋感應器 :(");
    while (1) { delay(1); }
  }
  finger.getTemplateCount();
  Serial.print("傳感器包含 "); Serial.print(finger.templateCount); Serial.println(" 模板");
  Serial.println("等待有效的手指...");
}

void loop() // 一遍又一遍
{
  getFingerprintIDez();
  delay(50); //不需讓它全速運行
}

uint8_t getFingerprintID() {
  uint8_t p = finger.getImage();
```

```

switch (p) {
    case FINGERPRINT_OK:
        Serial.println("影像拍攝");
        break;
    case FINGERPRINT_NOFINGER:
        Serial.println("沒有檢測到手指");
        return p;
    case FINGERPRINT_PACKETRECIEVEERR:
        Serial.println("通信故障");
        return p;
    case FINGERPRINT_IMAGEFAIL:
        Serial.println("影像錯誤");
        return p;
    default:
        Serial.println("未知錯誤");
        return p;
} // 確認成功!
p = finger.image2Tz();
switch (p) {
    case FINGERPRINT_OK:
        Serial.println("影像轉換");
        break;
    case FINGERPRINT_IMAGEMESS:
        Serial.println("影像太亂");
        return p;
    case FINGERPRINT_PACKETRECIEVEERR:
        Serial.println("通信故障");
        return p;
    case FINGERPRINT_FEATUREFAIL:
        Serial.println("找不到指紋特徵");
        return p;
    case FINGERPRINT_INVALIDIMAGE:
        Serial.println("找不到指紋特徵");
        return p;
    default:
        Serial.println("未知錯誤");
        return p;
}

```



```

// OK converted!
p = finger.fingerFastSearch();
if (p == FINGERPRINT_OK) {
    Serial.println("輸入匹配!");
} else if (p == FINGERPRINT_PACKETRECEIVEERR) {
    Serial.println("通信故障");
    return p;
} else if (p == FINGERPRINT_NOTFOUND) {
    Serial.println("指紋辨識不匹配");
    return p;
} else {
    Serial.println("未知錯誤");
    return p;
} // found a match!
Serial.print("找到 ID #"); Serial.print(finger.fingerID);
Serial.print(" 可信度為 "); Serial.println(finger.confidence);
return finger.fingerID;
}

// returns -1 如果失敗，則返回 ID #
int getFingerprintIDez() {
    uint8_t p = finger.getImage();
    if (p != FINGERPRINT_OK) return -1;
    p = finger.image2Tz();
    if (p != FINGERPRINT_OK) return -1;
    p = finger.fingerFastSearch();
    if (p != FINGERPRINT_OK) return -1
    // 找到匹配!
    Serial.print("找到 ID #"); Serial.print(finger.fingerID);
    Serial.print(" 可信度為 "); Serial.println(finger.confidence);
    return finger.fingerID;
}

```

7.4.9 Arduino IDE 統合所有功能且連接 LabVIEW 程式

```

////////////////////
///LabVIEW with arduino IDE 程式設計
char command;

```

```

String string;
////////////////////////////////////
///指紋辨識
int getFingerprintIDez();
#include <Adafruit_Fingerprint.h>
SoftwareSerial mySerial(A1, A0);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
// 腳位 A1 對應指紋辨識的綠線
// 腳位 A0 對應指紋辨識的黃線
////////////////////////////////////
//DHT11 溫度濕度感測
#include <SimpleDHT.h>
int pinDHT11 = A2;
SimpleDHT11 dht11(pinDHT11);
//DAT: A2 //溫書度感測資料線街腳為 A2
////////////////////////////////////
///倒車雷達
int spk=A3; //定義蜂鳴器為數位接腳 A3
int trigPinRL=5; //定義(左後 HC-SR04)Trig Pin(超音波模組觸發腳)-接腳 5
int echoPinRL=4; //定義(左後 HC-SR04)Echo Pin(超音波模組接收腳)-接腳 4
int trigPinRR=3; //定義(右後 HC-SR04)Trig Pin(超音波模組觸發腳)-接腳 3
int echoPinRR=2; //定義(右後 HC-SR04)Echo Pin(超音波模組接收腳)-接腳 2
long durationRL; //宣告 durationRL
long durationRR; //宣告 durationRR
float distanceRL; //宣告 distanceRL
float distanceRR; //宣告 distanceRR
int s=100; //基準時間
void lndash(); //宣告長劃訊號
void dash(); //宣告劃訊號
void dot(); //宣告點訊號
void sdot(); //宣告短點訊號
////////////////////////////////////
///防撞
int trigPinFL=12; //定義(左前 HC-SR04)Trig Pin(超音波模組觸發腳)-接腳 12
int echoPinFL=13; //定義(左前 HC-SR04)Echo Pin(超音波模組接收腳)-接腳 13
int trigPinFR=10; //定義(右前 HC-SR04)Trig Pin(超音波模組觸發腳)-接腳 10
int echoPinFR=11; //定義(右前 HC-SR04)Echo Pin(超音波模組接收腳)-接腳 11
int trigPinL=9; //定義(左 HC-SR04)Trig Pin(超音波模組觸發腳)-接腳 9

```

```

int echoPinL =8;           //定義(左 HC-SR04)Echo Pin(超音波模組接收腳) -接腳 8
int trigPinR =6;           //定義(右 HC-SR04)Trig Pin(超音波模組觸發腳) -接腳 6
int echoPinR =7;           //定義(右 HC-SR04)Echo Pin(超音波模組接收腳) -接腳 7
long durationFL;           //宣告 durationFL
long durationFR;           //宣告 durationFR
float distanceFL;          //宣告 distanceFL
float distanceFR;          //宣告 distanceFR
long durationL;            //宣告 durationL
long durationR;            //宣告 durationR
float distanceL;           //宣告 distanceL
float distanceR;           //宣告 distanceR

void setup() {
    Serial.begin(9600);
    while (!Serial);
    finger.begin(57600);
    finger.getTemplateCount();
    pinMode(A3, OUTPUT);
    pinMode(trigPinRL, OUTPUT);    //trigPinRL , OUTPUT
    pinMode(echoPinRL, INPUT);     //echoPinRL , INPUT
    pinMode(trigPinRR, OUTPUT);    //trigPinRR , OUTPUT
    pinMode(echoPinRR, INPUT);     //echoPinRR , INPUT
    pinMode(spkr, OUTPUT);         //spkr , OUTPUT
    pinMode(trigPinFL, OUTPUT);    //trigPinFL , OUTPUT
    pinMode(echoPinFL, INPUT);     //echoPinFL , INPUT
    pinMode(trigPinFR, OUTPUT);    //trigPinFR , OUTPUT
    pinMode(echoPinFR, INPUT);     //echoPinFR , INPUT
    pinMode(trigPinL, OUTPUT);     //trigPinL , OUTPUT
    pinMode(echoPinL, INPUT);      //echoPinL , INPUT
    pinMode(trigPinR, OUTPUT);     //trigPinR , OUTPUT
    pinMode(echoPinR, INPUT);      //echoPinR , INPUT
}

void loop(){
    string="";
    for(int i=0;i<=1;i++){
        command = ((byte)Serial.read());
        string += command;
    }
    if(string == "01")

```

```

    {
        TemperatureHumidity();
    }
else    if(string == "02")
    {
        BackDoDoDo();
    }
else    if(string == "03"){
        Finger();
    }
else    if(string == "04"){
        AvoidCollision();
    }
else{
    }
    delay(50);
}

////////////////////
////////////////////溫度濕度
void TemperatureHumidity(){
    byte temperature  ;
    byte humidity  ;
    int err = SimpleDHTErrSuccess;
    if ((err = dht11.read(&temperature, &humidity, NULL)) != SimpleDHTErrSuccess) {
        Serial.print("Read DHT11 failed, err=");
        Serial.println(err);
        return;
    }
    Serial.print('a');
    Serial.print((int)temperature);
    Serial.print('i');
    Serial.print((int)humidity);
}

////////////////////
////////////////////倒車雷達
void BackDoDoDo(){
    //////////////////////////////////////
    ////左後超音波感測

```

```

int twoa,twob;
digitalWrite(trigPinRL, LOW);           //給 Trig 低電位
delayMicroseconds(2);                   //持續 2 微秒
digitalWrite(trigPinRL, HIGH);          //給 Trig 高電位
delayMicroseconds(10);                  //持續 10 微秒
digitalWrite(trigPinRL, LOW);           //給 Trig 低電位
pinMode(echoPinRL, INPUT);              //讀取 echo 的電位
durationRL = pulseIn(echoPinRL, HIGH);  //收到高電位時的時間
distanceRL =durationRL*0.034/2;          //將時間換算成距離 cm //cm = (duration/2) / 29.1
if(distanceRL>1000){
    distanceRL=1;
}
twoa=distanceRL;
Serial.print('b');                      //方便 LabVIEW 分割數據
Serial.print(twoa);                     //Serial.print(實測距離)
////////////////////////////////////
////////右後超音波感測
digitalWrite(trigPinRR, LOW);           //給 Trig 低電位
delayMicroseconds(2);                   //持續 2 微秒
digitalWrite(trigPinRR, HIGH);          //給 Trig 高電位
delayMicroseconds(10);                  //持續 10 微秒
digitalWrite(trigPinRR, LOW);           //給 Trig 低電位
pinMode(echoPinRR, INPUT);              //讀取 echo 的電位
durationRR = pulseIn(echoPinRR, HIGH);  //收到高電位時的時間
distanceRR =durationRR*0.034/2;          //將時間換算成距離 cm //cm = (duration/2) / 29.1
if(distanceRR>1000){
    distanceRR=1;
}
twob=distanceRR;
Serial.print('c');                      //方便 LabVIEW 分割數據
Serial.print(twob);                     //Serial.print(實測距離)
////////////////////////////////////
////////雷達部分
if(distanceRL>distanceRR){
    distanceRL = distanceRR ;
}
else if(distanceRL<distanceRR){
    distanceRR = distanceRL ;
}

```

```

    }
    if( distanceRL >= 80 && distanceRL <=100 || distanceRR >= 80 && distanceRR <=100){ //假設距離
介於 80~100 之間
        lndash();
//spk 聲音為長劃訊號
    }
    else if (distanceRL >=60 && distanceRL <80 || distanceRR >=60 && distanceRR <80){ //假設距離
介於 60~79 之間
        dash();
//spk 聲音為劃訊號
    }
    else if (distanceRL >= 40 && distanceRL <60 || distanceRR >= 40 && distanceRR <60){ //假設距離介
於 40~59 之間
        dot();
//spk 聲音為點訊號
    }
    else if (distanceRL >= 30 && distanceRL <40 || distanceRR >= 30 && distanceRR <40){ //假設距離介
於 30~39 之間
        sdot(); sdot();
//spk 聲音為兩個短點訊號
    }
    else if (distanceRL >= 20 && distanceRL <30 || distanceRR >= 20 && distanceRR <30){ //假設距離介
於 20~29 之間
        sdot(); sdot(); sdot(); sdot(); //spk 聲
音為四個短點訊號
    }
    else if (distanceRL <20 || distanceRR <20){ //假設
距離小於 19cm
        digitalWrite(spkr, HIGH);
//spk 持續發出聲響
        delay(10);
    }
    else
        digitalWrite(spkr, LOW);
//超過 100cm 以上 spk 不發出聲響
}

void lndash(){ //設定長劃訊號
    digitalWrite(spkr,HIGH); //spk 響 4 秒

```

```

    delay(s*4);
    digitalWrite(spkr,LOW);
}

void dash(){
    //設定劃訊號
    digitalWrite(spkr,HIGH); //spkr 響 2.5 秒
    delay(s*2.5);
    digitalWrite(spkr,LOW);
}

void dot(){
    //設定點訊號
    digitalWrite(spkr,HIGH); //spkr 響 1.5 秒
    delay(s*1.5);
    digitalWrite(spkr,LOW);
}

void sdot(){
    //設定短點訊號
    digitalWrite(spkr,HIGH); //spkr 響 1 秒，停頓 0.3 秒
    delay(s);
    digitalWrite(spkr,LOW);
    delay(s*0.3);
}

////////////////////////////////////
////////////////////////////////////指紋辨識部分
int Finger(){
    uint8_t p = finger.getImage();
    if (p != FINGERPRINT_OK) return -1;
    p = finger.image2Tz();
    if (p != FINGERPRINT_OK) return -1;
    p = finger.fingerFastSearch();
    if (p != FINGERPRINT_OK) return -1;
    Serial.print('d'); //方便 LabVIEW 分割數據
    Serial.print(finger.confidence); //輸出手指信任度數值
    return finger.fingerID;
}

////////////////////////////////////
////////////////////////////////////防撞四腳
void AvoidCollision(){
    //////////////////////////////////////左前
    Serial.print('e'); //方便 LabVIEW 分割數據
    int foura,fourb,fourc,fourd; //使輸出數值能整數且更新

```

```

digitalWrite(trigPinFL, LOW);          //給 Trig 低電位
delayMicroseconds(2);                  //持續 2 微秒
digitalWrite(trigPinFL, HIGH);         //給 Trig 高電位
delayMicroseconds(10);                 //持續 10 微秒
digitalWrite(trigPinFL, LOW);          //給 Trig 低電位
pinMode(echoPinFL, INPUT);             //讀取 echo 的電位
durationFL = pulseIn(echoPinFL, HIGH); //收到高電位時的時間
distanceFL = durationFL*0.034/2;        //將時間換算成距離 cm //cm = (duration/2) / 29.1
    if(distanceFL>1000){
        distanceFL=1;
    }
foura = distanceFL;
Serial.print(foura);                  //Serial.print(實測距離)
////////////////////////////////////右前
Serial.print('f');                    //方便 LabVIEW 分割數據
digitalWrite(trigPinFR, LOW);         //給 Trig 低電位
delayMicroseconds(2);                 //持續 2 微秒
digitalWrite(trigPinFR, HIGH);        //給 Trig 高電位
delayMicroseconds(10);                //持續 10 微秒
digitalWrite(trigPinFR, LOW);         //給 Trig 低電位
pinMode(echoPinFR, INPUT);            //讀取 echo 的電位
durationFR = pulseIn(echoPinFR, HIGH); //收到高電位時的時間
distanceFR = durationFR*0.034/2;       //將時間換算成距離 cm //cm = (duration/2) / 29.1
if(distanceFR>1000){
    distanceFR=1;
}
fourb = distanceFR;
Serial.print(fourb);                  //Serial.print(實測距離)
////////////////////////////////////左
Serial.print('g');                    //方便 LabVIEW 分割數據
digitalWrite(trigPinL, LOW);          //給 Trig 低電位
delayMicroseconds(2);                 //持續 2 微秒
digitalWrite(trigPinL, HIGH);         //給 Trig 高電位
delayMicroseconds(10);                //持續 10 微秒
digitalWrite(trigPinL, LOW);          //給 Trig 低電位
pinMode(echoPinL, INPUT);             //讀取 echo 的電位
durationL = pulseIn(echoPinL, HIGH);  //收到高電位時的時間
distanceL = durationL*0.034/2;        //將時間換算成距離 cm //cm = (duration/2) / 29.1

```



```

if(distanceL>1000){
    distanceL=1;
}
fourc = distanceL ;
Serial.print(fourc);                //Serial.print(實測距離)
//////////////////////////////////////右
Serial.print('h');                  //方便 LabVIEW 分割數據
digitalWrite(trigPinR, LOW);        //給 Trig 低電位
delayMicroseconds(2);               //持續 2 微秒
digitalWrite(trigPinR, HIGH);       //給 Trig 高電位
delayMicroseconds(10);              //持續 10 微秒
digitalWrite(trigPinR, LOW);        //給 Trig 低電位
pinMode(echoPinR, INPUT);           //讀取 echo 的電位
durationR = pulseIn(echoPinR, HIGH); //收到高電位時的時間
distanceR =durationR*0.034/2;        //將時間換算成距離 cm //cm = (duration/2) / 29.1
if(distanceR>1000){
    distanceR=1;
}
fourd = distanceR ;
Serial.print(fourd);                //Serial.print(實測距離)
}

```

7.5 LabVIEW 程式設計

7.5.1 整體程式外觀

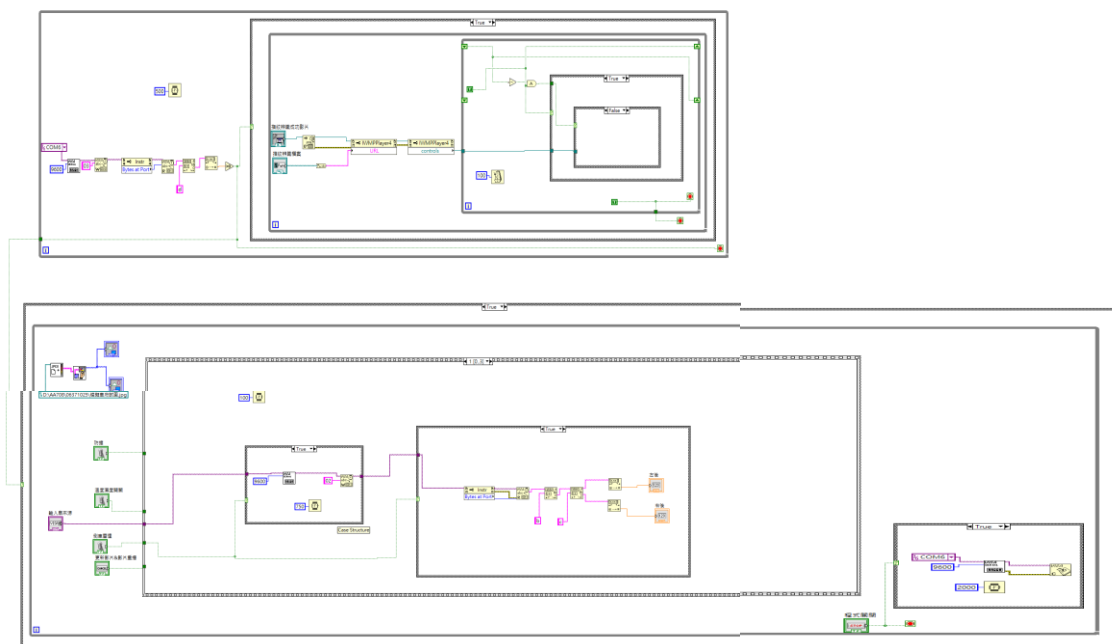


圖 7-29 整體程式外觀 (Ture)

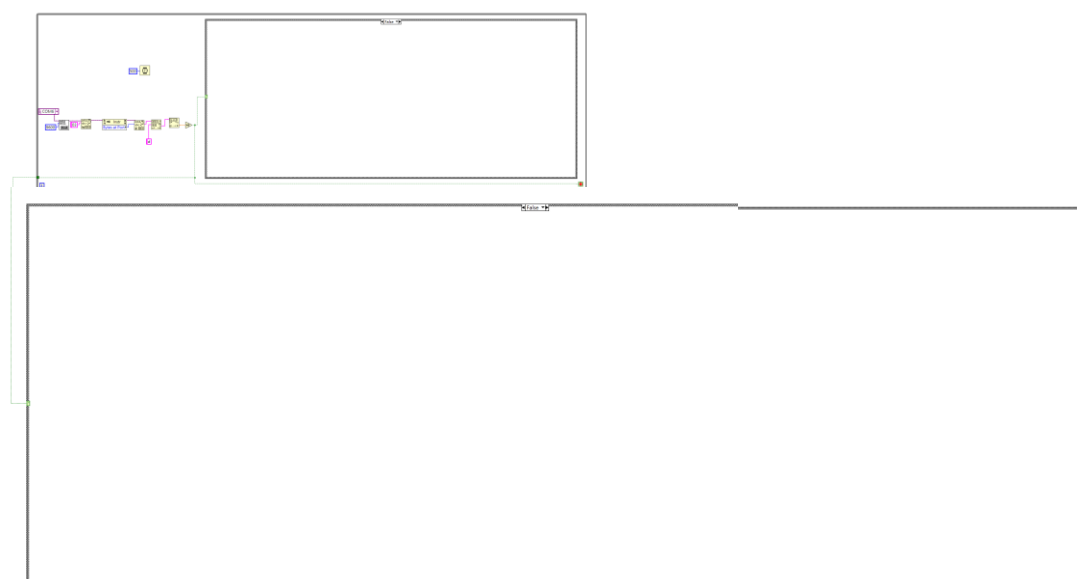


圖 7-30 整體程式外觀(False)

7.5.2 溫溼度感測

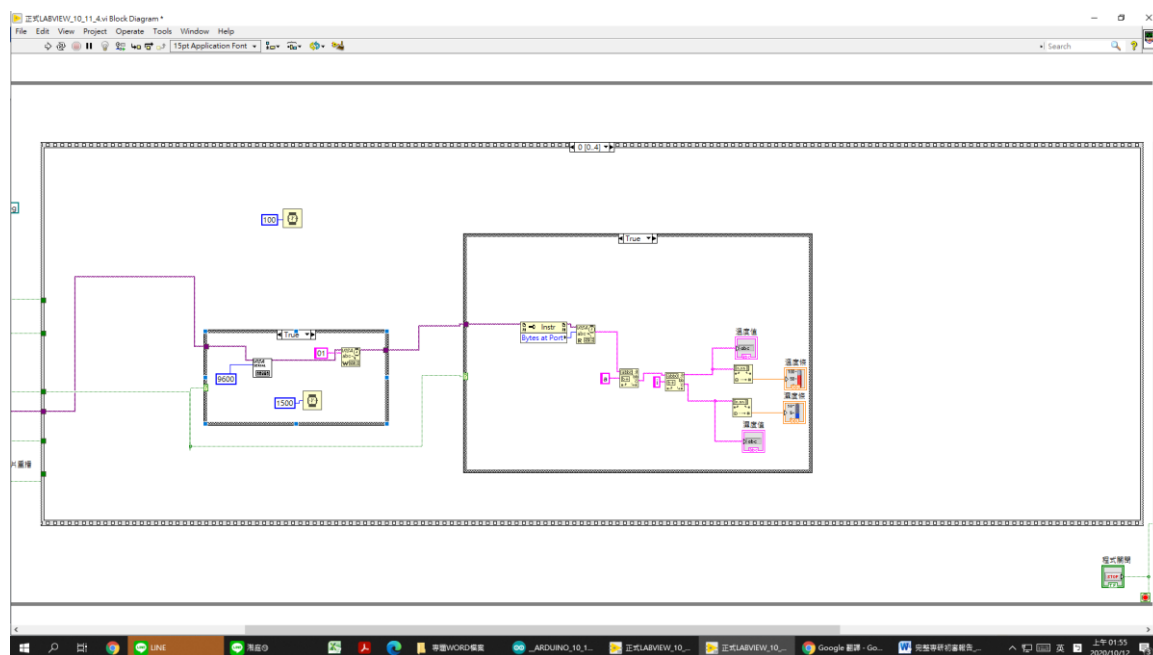


圖 7-31 溫溼度感測(Ture)

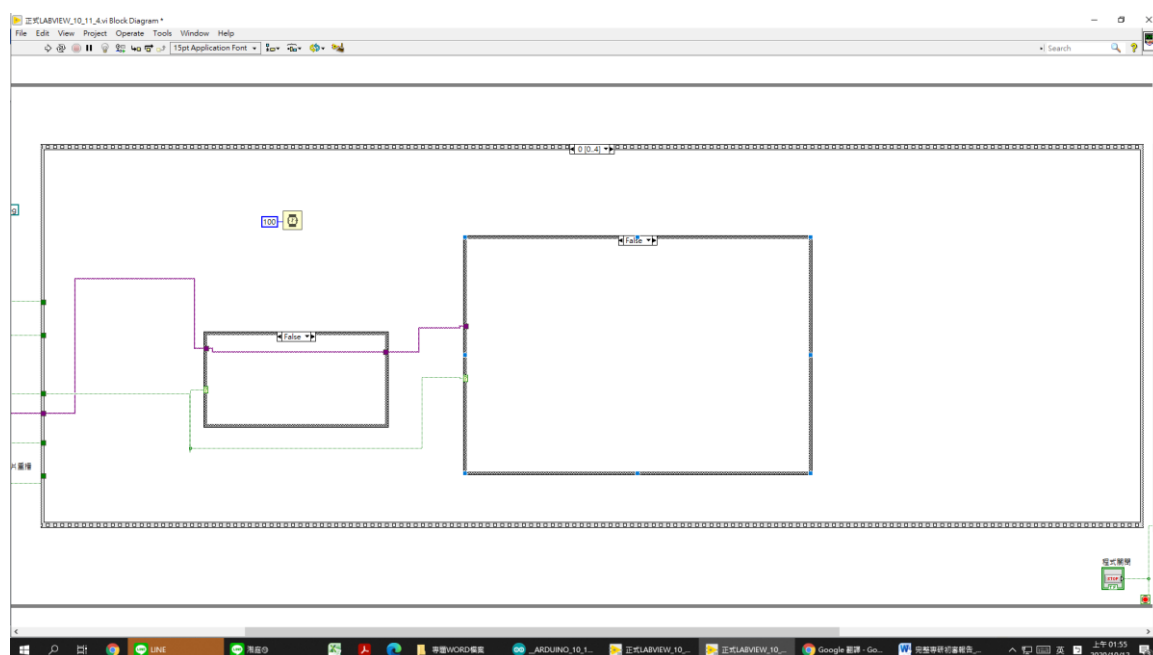


圖 7-32 溫溼度感測(False)

7.5.3 倒車雷達

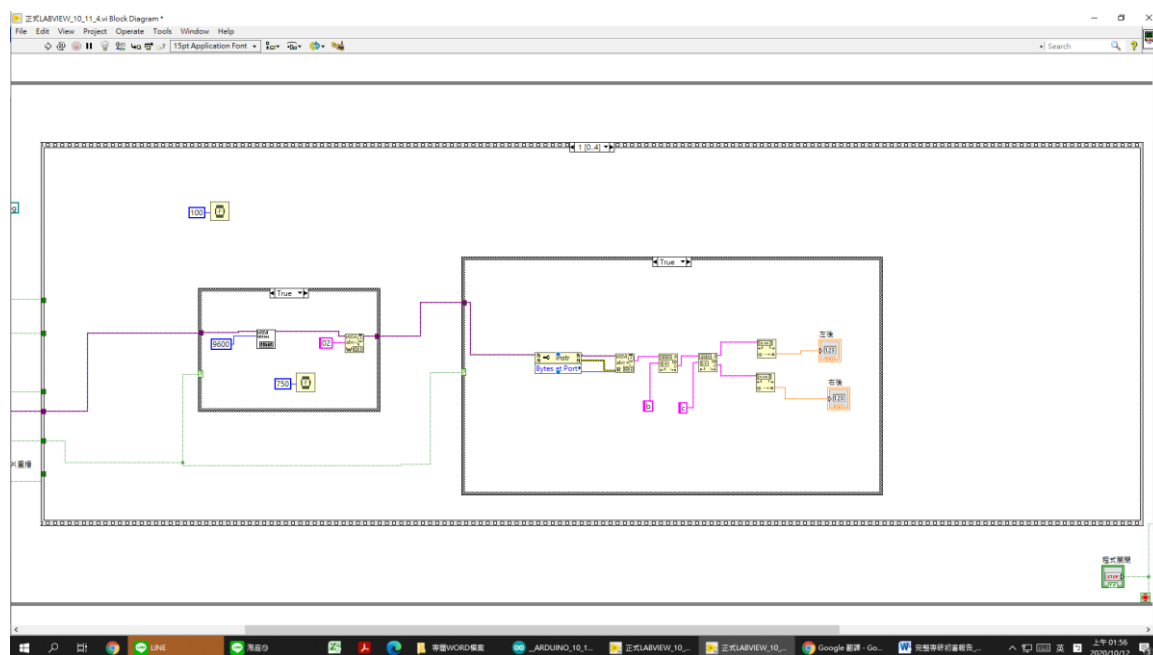


圖 7-33 倒車雷達(Ture)

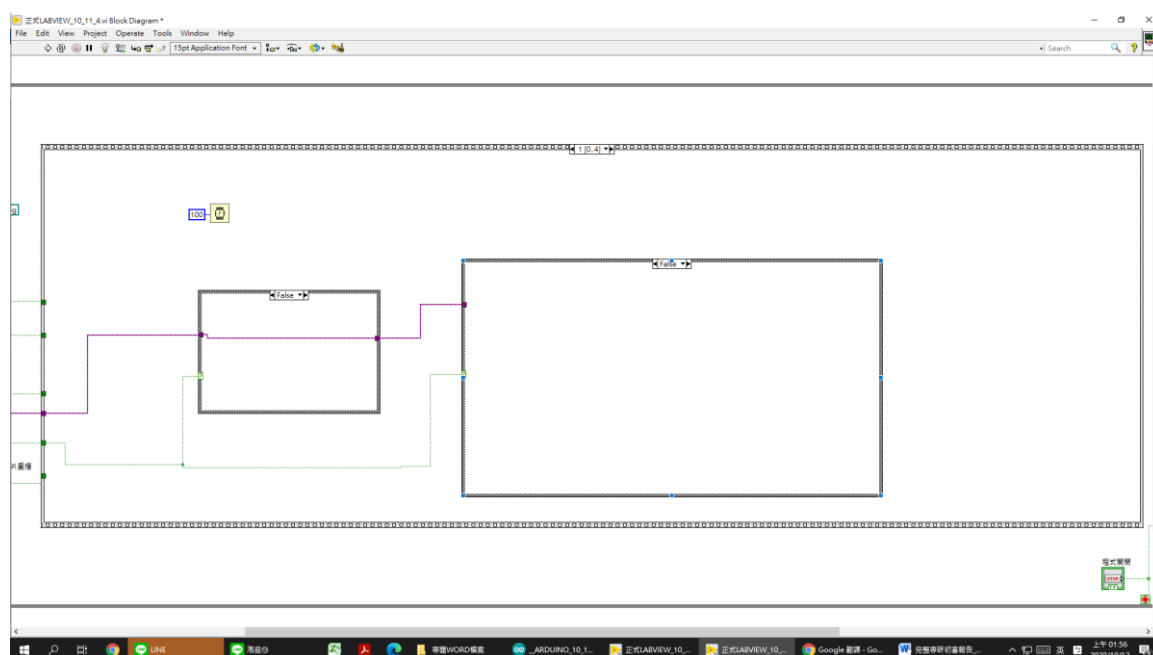


圖 7-34 倒車雷達(False)

7.5.4 指紋辨識

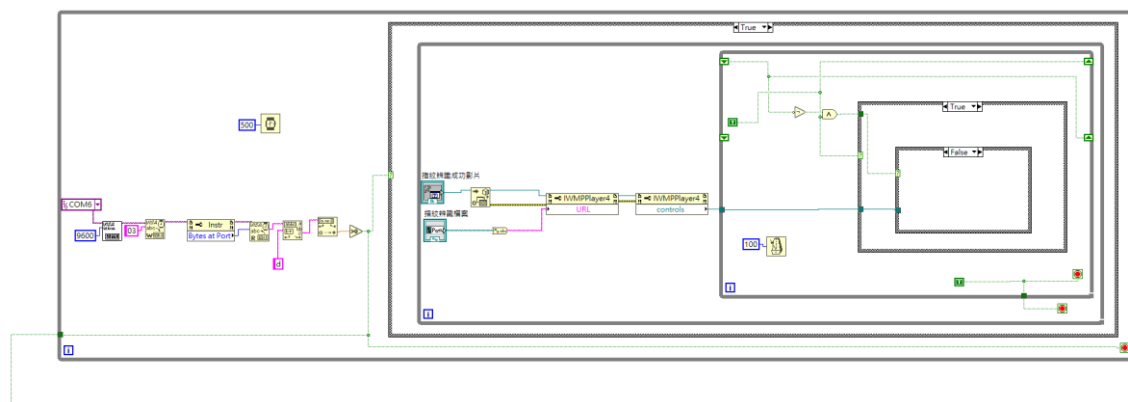


圖 7-35 指紋辨識(Ture)

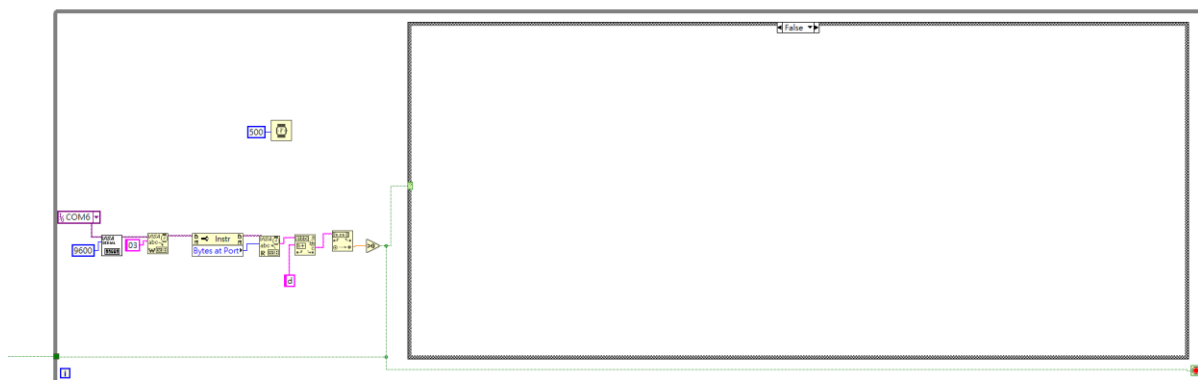


圖 7-36 指紋辨識(False)

7.5.5 影音導覽

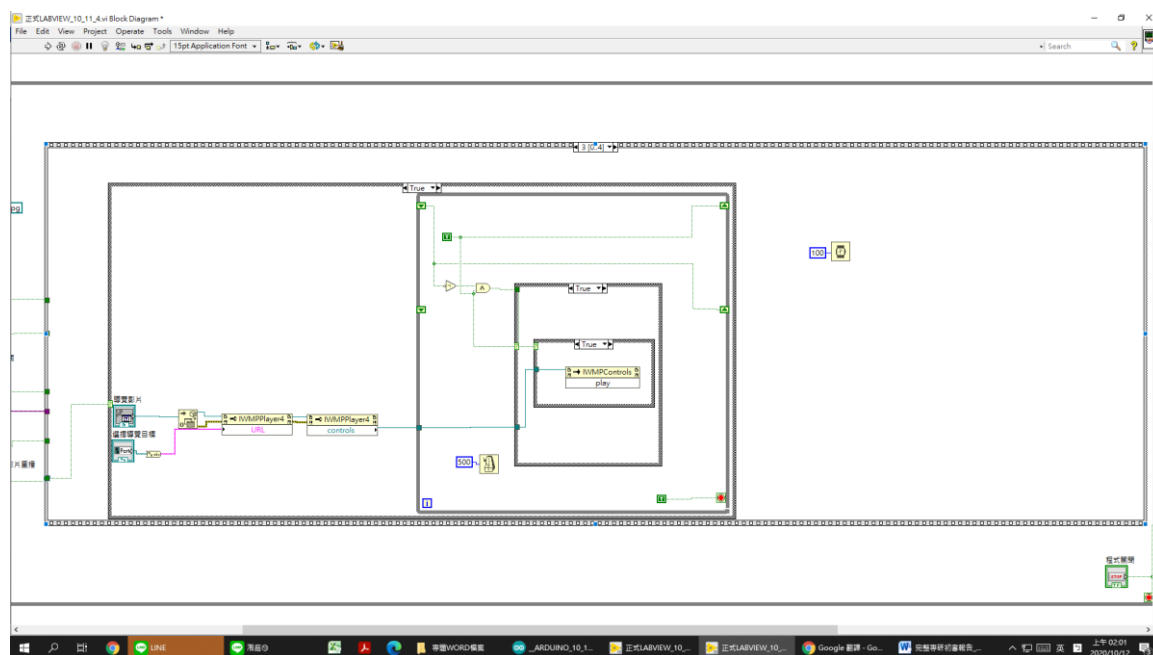


圖 7-37 影音導覽(Ture)

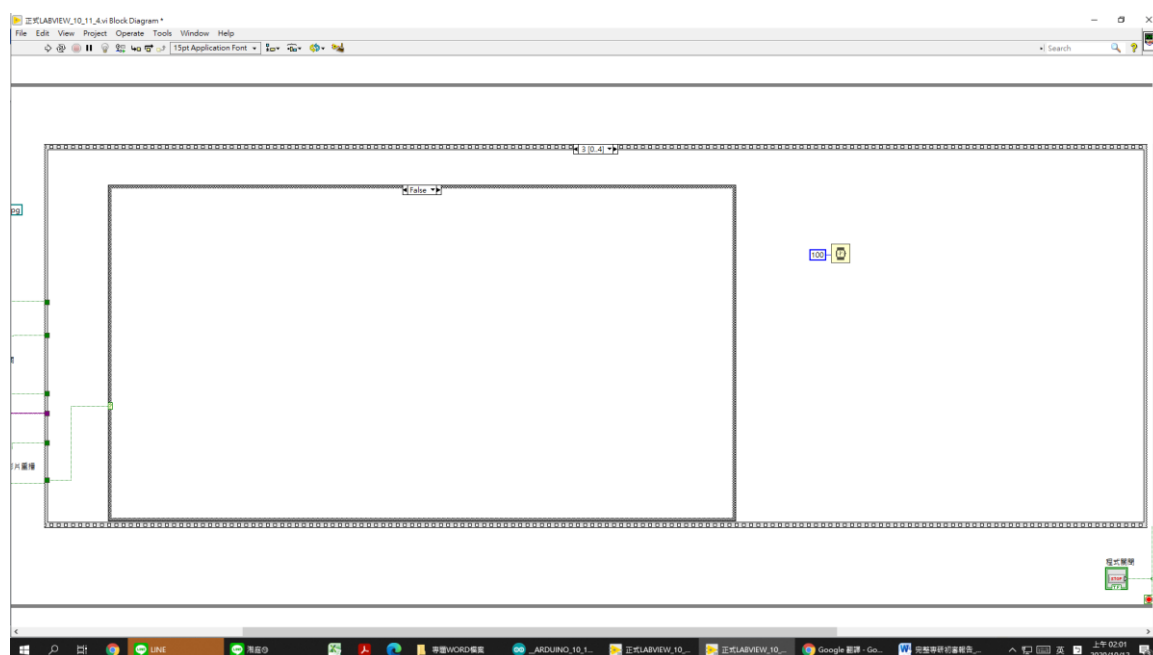


圖 7-38 影音導覽(False)

7.5.6 防撞四角

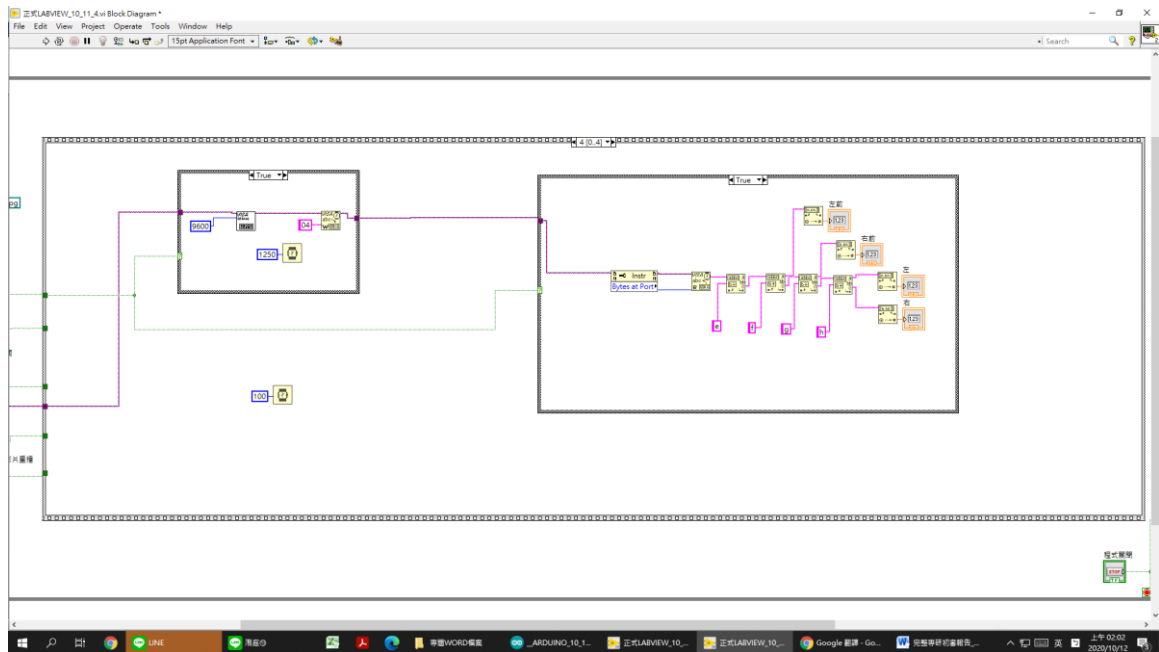


圖 7-39 防撞四角(Ture)

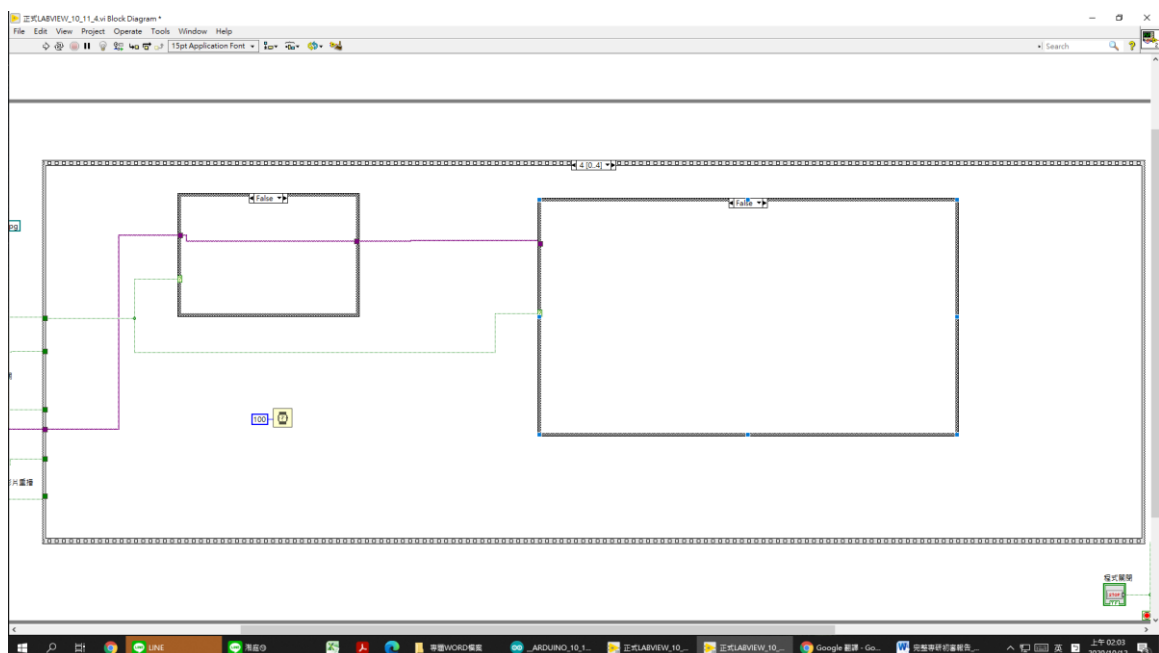


圖 7-40 防撞四角(False)

參考文獻

內文參考文獻：

[1]: http://amebse.nchu.edu.tw/new_page_401.htm

[2]: <https://www.ctimes.com.tw/DispArt/tw/%E6%99%BA%E6%85%A7%E6%B1%BD%E8%BB%8A/1101061144M5.shtml>

[3]: https://www.artc.org.tw/upfiles/ADUpload/knowledge/tw_knowledge_501525402.pdf

[4]: <https://scitechreports.blogspot.com/2017/11/lidar.html>

[5]: https://www.digitimes.com.tw/iot/article.asp?cat=158&cat1=20&cat2=80&id=0000566963_4tflrywn51rs644r6zwxh

[6]: <https://kknews.cc/zh-tw/travel/oap6g6o.html>

[7]: <https://www.cool3c.com/article/84035>

[8]: <https://www.cool3c.com/article/84035>

溫度濕度感測器：

<https://www.taiwansensor.com.tw/product/dht11-%E6%BA%AB%E6%BA%BC%E5%BA%A6%E6%84%9F%E6%B8%AC%E5%99%A8%E6%A8%A1%E7%B5%84-%E6%BA%AB%E5%BA%A6%E6%BF%95%E5%BA%A6-%E9%9B%99%E9%87%8D%E6%84%9F%E6%B8%AC/>

<https://www.itread01.com/content/1541965345.html>

http://www.elecfans.com/yuanqijian/sensor/20180126623808_a.html

Arduino UNO 板：

<http://epaper.gotop.com.tw/PDFSample/AEH004000.pdf>

HC-SR04 超音波感測器：

<https://zh.wikipedia.org/wiki/%E8%B6%85%E8%81%B2%E6%B3%A2>

參考文獻：《超圖解 Arduino 互動設計入門》第三版

指紋辨識介紹：

https://www.cool3c.com/article/84035#comment-area-main-post_8403

5-0

<https://www.yiboard.com/thread-820-1-1.html>

<https://goods.ruten.com.tw/item/show?21820031266674>

<https://news.cnyes.com/news/id/4375167>

<https://read01.com/zh-mo/Q34yemG.html#.Xsop9mgzaUk>

有源蜂鳴器：

<https://arduinomodels.info/ky-012-active-buzzer-module/>

<https://blog.jmaker.com.tw/arduino-buzzer/>

Arduino 平台介紹：

[https://sites.google.com/site/wenyumaker3/03-arduino-guo-xiao-ke-cheng/01ren-](https://sites.google.com/site/wenyumaker3/03-arduino-guo-xiao-ke-cheng/01ren-shiarduino?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1)

[shiarduino?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDia](https://sites.google.com/site/wenyumaker3/03-arduino-guo-xiao-ke-cheng/01ren-shiarduino?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1)

[log=1](https://sites.google.com/site/wenyumaker3/03-arduino-guo-xiao-ke-cheng/01ren-shiarduino?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1)

<https://blog.jmaker.com.tw/arduino-tutorials-1/>

LabVIEW 平台介紹：

<https://zh.wikipedia.org/zh-tw/LabVIEW>

Arduino 軟體介面：

<https://www.arduino.cc/en/Guide/Environment>

LabVIEW 設計：

參考課本→LabVIEW201X 虛擬儀控程式設計