屏東縣第 61 屆國中小學科學展覽會
作品說明書

科　　別：生活與應用科學科(一)（機電與資訊）

組　　別：國中組

作品名稱：複眼結構紅外線測距模型的實作與手勢操作介面的設計

關 鍵 詞：複眼、飛時測距(ToF)、使用者介面

編號：**B6015**

複眼結構紅外線測距模型的實作與手勢操作介面的設計

摘要：

本研究目的是藉由製作紅外線感測器模型，再進行手勢操作介面的設計，並探討相關的應用。研究以長距離小型紅外線測距感測器為主要設備，結合輻射狀的 8 組感測支架和可二維移動的立架，以及 9 顆感測器，構成一個如同昆蟲複眼結構的實驗性質模型。在硬體線路方面，以具有較多腳位的 Arduino Mega2560 為控制中心，搭配 I2C 多工器以簡化感測器配線，並使用 LED 可顯示即時偵測狀態，進一步設計以手勢操作的使用者介面。

研究結果顯示，本模型可以配合需要，如同昆蟲複眼一般，指向任意方向，偵測物體的距離，並進行多種實驗。而手勢操作介面，可以辨識出直覺的手勢動作，本研究也設計並測試了多達數十種以上實用的動作碼。本研究同時也展示了多個應用的範例，例如追距定位、遊戲和文件瀏覽的操作手勢，以及可以偵測八個方位的即時動態雷達圖功能。

壹、研究動機

在學校資訊課程時第一次接觸到使用紅外線測距模組以及能辨識出九種手勢的手勢辨識模組，我們被這些紅外線感測器的功能深深吸引，引發了我們將功能加以延伸和加強的想法。例如：手勢可否再增加一些？將操作距離延伸到較遠的位置，是否可行？偵測的角度可否再廣一點或是可以調整？更多顆的感測器要撰寫程式控制？可以產生哪些好玩的應用？

本研究的構想是結合更多顆長距離的紅外線測距模組，嘗試設計和製作可以任意彈性調整角度的模型，再試著透過程式的撰寫，賦予模型能具有非接觸式的手勢操作功能，並希望能找到有趣應用。因應目前新冠肺炎等公共衛生的挑戰，使用非接觸式的手勢操作方式，或許可以應用在學校或是公共場合的導覽平台和展示機器操作上，能夠降低操作時的接觸傳染風險。

貳、研究目的

一、紅外線感測模組的測試與選用

二、手勢操作的系統架構設計

三、複眼結構紅外線測距模型的實作的設計與製作

四、手勢操作系統的基本功能設計

五、手勢操作介面的設計

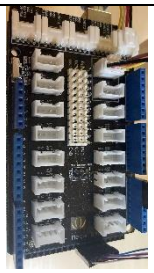
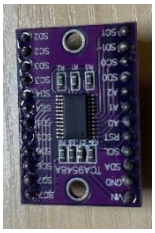
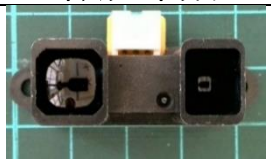
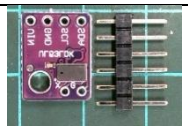
六、追蹤定位功能程式設計與性能測試

七、手勢操作介面與模型的應用展示

參、研究設備及器材

微控制器	Arduino mega2560
擴展板	Grove － Mega shield
紅外線測距模組	SHARP-GP2Y0A02、TOF10120、VL53L0X
手勢辨識模組	Seeed Grove gesture v1.0 (晶片 PAJ7620U2)
LED	3mm 單色 (紅色)
I2C 多工器	TCA9548A(8 通道)
電動鑽床	鑽頭尺寸 2 mm、3 mm、4mm、5 mm
模型用木材	圓木條 3 mm、方木條 5 mm*5 mm、6 mm*6 mm
其他電動工具	電動線鋸機、電動砂布機

		
鑽床	線鋸機	砂帶機

 微控制器	 擴展板	 I2C 多工器
 紅外線感測器 (SHARP)	 紅外線感測器 (ST VL53L0K)	 紅外線感測器 (TOF10120)

肆、研究過程

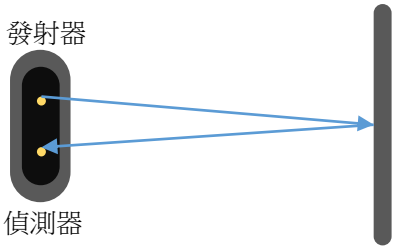
一、文獻探討

(一) 複眼

根據維基百科的內容得知，複眼構造常見於昆蟲等節肢動物身上，例如蜻蜓、蜜蜂和蜘蛛。動物複眼的優點是能夠為提供較寬廣的視野，可以有效的計算與物體的方位和距離，有利於動物作出更快速的判斷和反應。

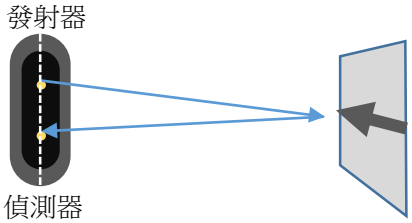
(二) 紅外線飛時測距(Time of Flight)原理

市面上常見的紅外線感測器模組上一般是一個紅外線發射器和一個偵測器組成。其原理是發射器會產生波長為 940nm 的紅外線光源，如果照射到物體後反射回偵測器，利用取得光的往返時間，使用自然課本中所提的公式：距離=(光速 x 往返時間)/2，就可以求得距離。根據產品規格書得知，感測器測距的準確性會受到室內外光源、反射物體的顏色、距離和電源的影響。例如：室內外其他光源太強會造成干擾。白色物體反射率較高於灰色物體，能有較遠的偵測距離。而距離越遠，其準確度會會下降。有模組則建議使用時要並聯電容器，降低電源的影響。而不同的模組在設計上也有不同的測量距離範圍。而建議的安裝角度則為物體的移動方向與發射器和偵測器軸線垂直。本研究希望能從市面的產品中，挑選出適合我們需求的模組。



距離=(光速 x 往返時間) / 2

圖 1 距離公式



發射器與偵測器軸線應與物體移動方向垂直

圖 2 建議安裝角度

(三) 圖形使用者介面(Graphic User Interface, GUI)

在電腦中常見兩種使用者介面，分別為命令列介面和圖形使用者介面，前者需使用鍵盤輸入指令，常常需要記憶不少的指令和參數；而後者是利用指標裝置(滑鼠)去點選特定的圖示，以執行特定的程式或功能，較適合一般使用者學習和操作。

要成為實用的使用者介面，在進行相關的設計時，要考量介面設計的一致性、可復原性、指引功能和多元性…等。一致性是指執行不同功能時，也有相同的操作流程；可復性是指讓使用者可以很快從錯誤操作中恢復成之前的狀態；多元性則是考量不同熟悉程度的使用者的需求，例如：初學者需要提供較多的操作指引，而進階者則可提供快速的操作捷徑功能。使用介面中常見的滑鼠操作如下表 1。

表 1 常見滑鼠操作動作

滑鼠的操作動作	說明
Mouse Down	滑鼠指標在圖示上方且按下滑鼠按鍵
Mouse Up	滑鼠指標在圖示上方且放開滑鼠按鍵
Mouse Click	按滑鼠按鍵一下
Mouse Hover	滑鼠指標停留在圖示上方
Mouse Move	滑鼠指標移動到圖示上方

本研究希望能利用紅外線感測器的測距功能來實作出類似滑鼠的部份操作功能功能，搭配 LED 來顯示執行的狀態，實現手勢操作的目標。

二、研究過程：

(一) 紅外線測距模組的比較與測試

1. **基本規格的比較**：本研究購買尺寸較小且可測量距離在 200cm 內的紅外線測距模組，

參考規格建議書，比較尺寸、測量距離、發射器角度/偵測器角度、偵測速度、連線介面及其他特性。

2. **距離測試：**以白色物體為反射板，使用各模組的範例程式在三種距離(40cm、80cm、120cm)下進行距離的測量，觀察比較數據的變動情形。以挑選符合本研究需求的模組，如圖 3。

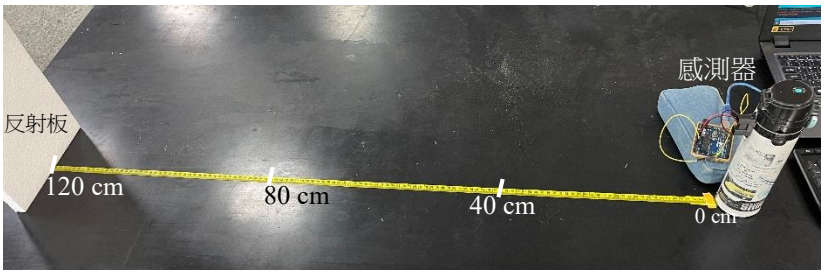


圖 3 模組測試方式

(二) 手勢操作系統架構需求

本研究所提出的手勢操作系統依功能需求分成四個部份：1. 具有系統狀態顯示、2. 具有使用者操作模式、3.具有追蹤定位功能和 4. 具有動作展示功能。依需求再擬定出可行的系統架構，和後續的模型設計。敘述如下：

1. **具有系統狀態顯示**：利用 LED 的明暗和閃爍以輔助顯示感測器的偵測情形。閃爍代表使用者的手部落在特定操作範圍內，而恒亮則代表使用者選定某感測器，成為焦點。
2. **具有使用者操作模式**：讓使用者可以透過手勢指揮其他機器或程式，執行特定動作或功能。相當於電腦的圖形使用者介面，提供無線滑鼠的部份功能。
3. **具有追蹤定位功能**：追蹤定位採用自動的方式，經由感測的結果，能驅動有伺服馬達的支架往垂直和水平方向轉動，讓多數的感測器可以更正確的指向使用者。當取得使用者的距離後，再進一步自動設定手部操作的距離範圍。相關的距離範圍，也可以採用固定模式加以設定。也可以進行物體的追蹤。
4. **提供動作展示功能**：可將辨識出的手勢動作以特定格式的方式輸出，可以驅動伺服馬達或是傳遞給其他程式使用，以展示特定動作或執行特定功能。

(三) 可調整式感測器支架模型的設計與製作

1. 第一代感測器支架

感測器的配置為 3*3 的型式。使用方型木條和螺絲為材料，在十字交叉處使用口徑 5mm*5mm 長度 4cm 的木條，其餘則使用 3cm 木條，組合成外形類似「王」字形的支架；將九個 IR 感測器裝在中心點和八個端點，上下左右感測器彼此的距離約略相等。垂直支架和水平支架可以分別前後彎曲約 360 度，如圖 4。

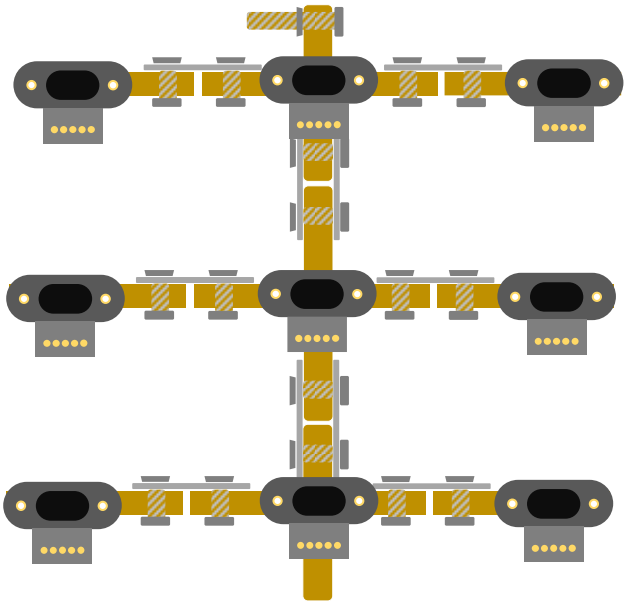


圖 4 第一代可調整式感測器支架的設計圖

2. 第二代感測器支架和立架

(1) 感測器支架外形設計

設計上參考 CD 光碟片尺寸(直徑約為 12 公分)的圓盤形構造，呈輻射對稱，外圍與中央感測器的距離約略相等。使用口徑為 5mm、6mm 方型木條和 3mm 圓形木條，製作 8 支感測器小支架，中央木板寬約 3cm 的八邊形木板，將 8 個感測器分別安裝在 8 個不同方位，再將 LED 以螺絲固定在感測器上方或下方的位置，如圖 5。

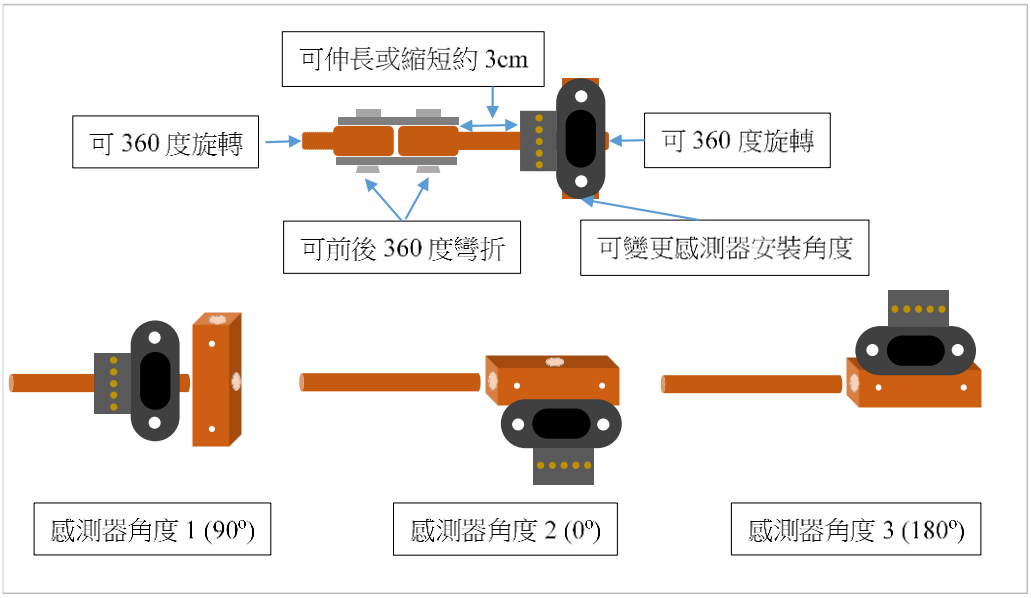
(2) 可轉動角度

8 支感測器小支架可以分別各自進行 360 度的旋轉和彎曲；感測器也可以同時進行 360 度旋轉，並可選擇三個不同的安裝角度(0°、90°、180°)。

(3) 可垂直和水平轉動立架

立架主要用支撐感測器支架，並可以適當調整前後或上下的長度。使用兩顆小伺服馬達並透過程式控制，使感測器朝向特定的方向轉動。

依序使用 3mm 螺絲和樹脂進行全部零件的組裝。主要的過程如圖 6 所示。



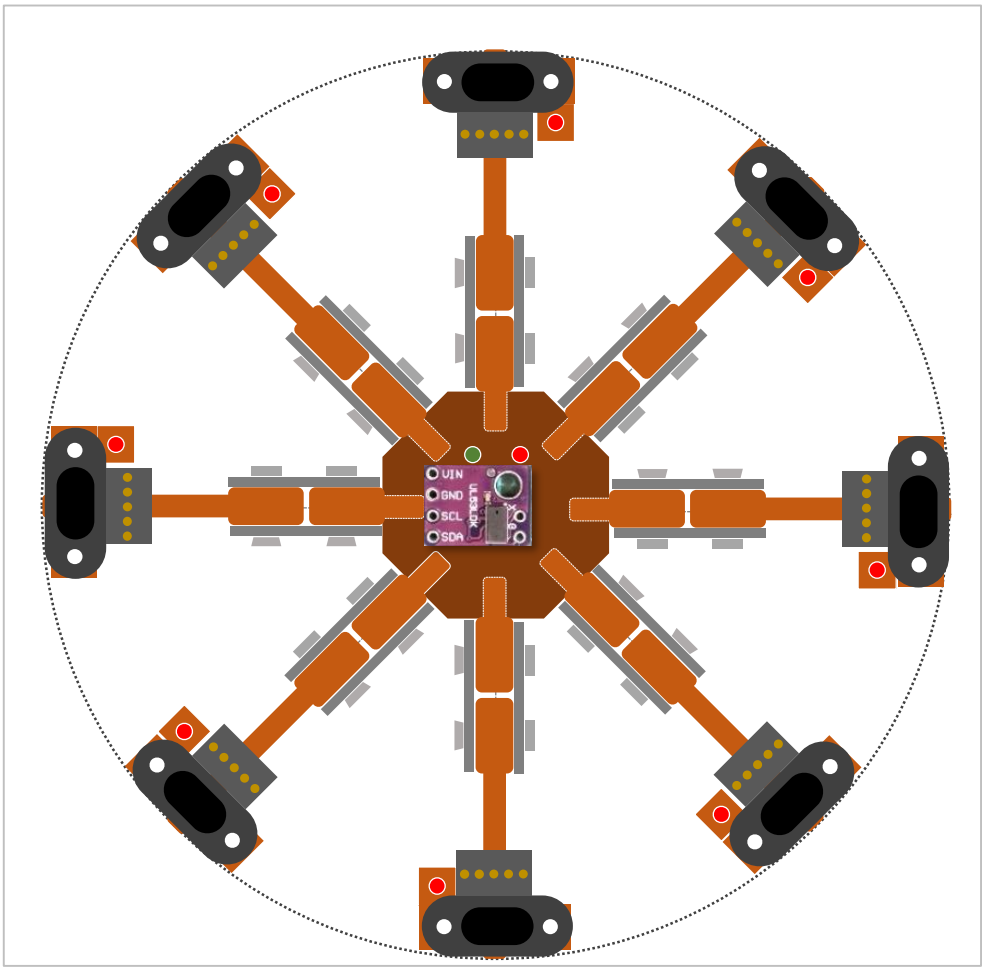
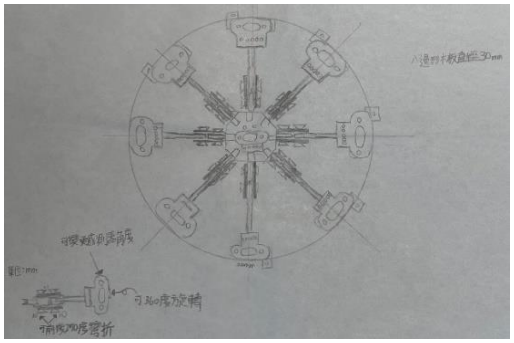
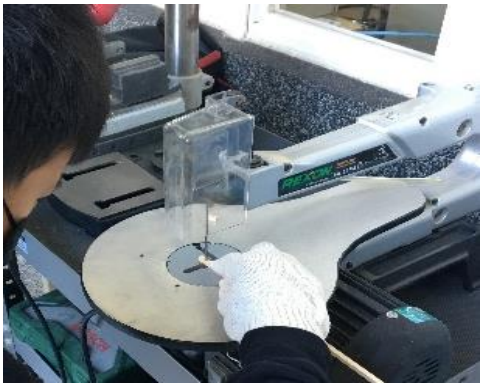


圖 5 第二代複眼結構可調整式支架設計圖



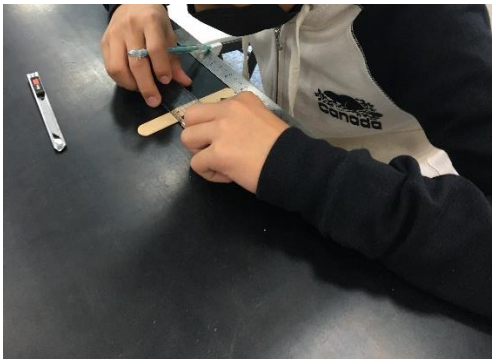
1.繪製設計草圖

2.使用電鑽及虎鉗在木條上鑽孔

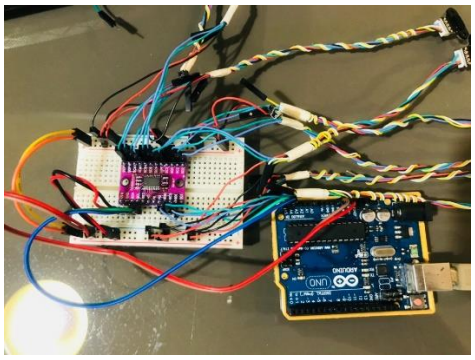


3.使用線鋸機進行木條切割

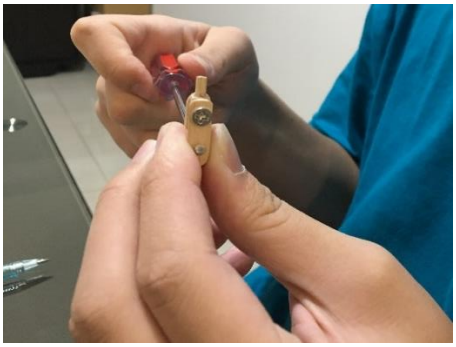
4.完成的零件



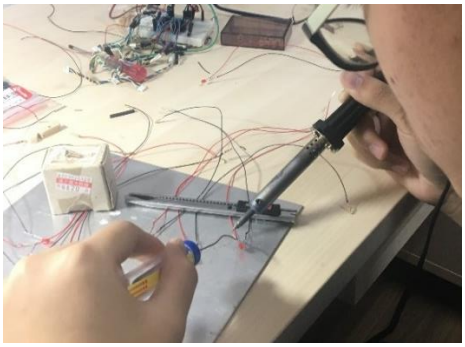
5.立架模型的劃線及美工刀切割



6.感測器與多工器連接測試



7.支架模型組裝



8.LED 接線的焊接

圖 6 模型製作和組裝過程

(四) 手勢操作系統的基本功能設計

1. 狀態顯示功能

每一個感測器配置一顆紅色 3mm LED，分別串聯 220 歐姆的電阻，並連接到 Arduino 微控制器的數位腳位，用以顯示偵測到的狀態，如下所述：

(1) 系統啟動測試：順時鐘方向輪流閃爍時間約 100ms，確認 LED 是否正常。

(2) 偵測範圍內：在不同距離範圍有不同閃爍時間約 10ms~20ms。

(3) 選定成為焦點：持續點亮。

透過實際測試，調整不同程式中各 LED 閃爍的時間，找出較佳的設定值，以加速測距功能和清楚顯示目前狀能。

2. 測距功能

共有 9 顆感測器。其中 8 個為外圍感測器，接在 8 通道 I2C 多工器上；1 個中央感測器直接以 I2C 方式連接微控制器擴展板。依序讀取感測器資料後，存入距離陣列中，提供其他程式的讀取。程式主要流程如下：

(1) **引入不同感測器的程式庫的標頭檔**。TOF10120 模組使用內建的 Wire 程式庫，VL53L0K 模組使用原廠提供的程式庫。

(2) **驅動多工器連接 TOF10120 模組**。透過程式切換到多工器不同的 bus。bus0~bus7 則分別連接到感測器 0~感測器 7。感測器 8(VL53L0K 模組)則獨立初始化。

(3) **取得距離資料**。依序(由感測器 0~8)分別讀取距離資料。TOF10120 模組會傳回 2 個位元組，資料需進行轉換。接著將距離單位由毫米(mm)調整成公分(cm)，依序存入距離陣列中。

(4) **簡化距離數值**。依設定距離範圍分為六個區域，主要作用區為身體、**手部 A 區和手部 B 區**，分別以整數 1~3 代替，非作用區以 0 代替。身體距離 D 可採用自動偵測或是固定式兩種。

(五) 手勢操作介面的設計

利用手部在**外圍感測器的 A、B 區**被兩次偵測結果的組合，產生手勢動作編碼來對應到手勢動作。系統提供如同智慧型手機 HOME 鍵的快速回復功能，使用**中央感測器**隨時都可以回復上一層選單或是最初狀態，如表 2。主要程序如下：

1. 如果已偵測到一個焦點，則第二次偵測加速。
2. 由距離陣列中依序讀取，分別累積計算手部 A 區和手部 B 區的偵測次數。分別存入相對分區的計數陣列中。
3. 當計數陣列中有項目達到最大偵測次數(MAX_COUNT)，代表該感測器(位置)為選定(焦點)，輸出字串；若有第二個字串，則疊加後輸出。
4. 輸出結果字串，代表動作編碼。每個編碼長度為 4，由兩組指令組成。每組指令格式：區域名稱 + 感測器編號。

表 2 手勢動作的程式設計說明

手勢動作	對應滑鼠動作	LED 顯示	程式設計說明
選定(焦點)	Mouse Hover	恒亮	手部**停留**在某**外圍感測器**上 A 或 B 區中連續偵測到一定的次數
選定後向前 (A 區→B 區)	Mouse Hover + Mouse Down	恒亮後閃爍	在某**外圍感測器**上 A 區中選定狀態後，接著移動到 B 區中被連續偵測到一定的次數。**或距離值由 2→3。**
選定後向後 (B 區→A 區)	Mouse Hover + Mouse Up	恒亮後閃爍	在某**外圍感測器**上測量 B 區中選定狀態後，接著移動到 A 區中被連續偵測到一定的次數。**或距離值由 3→2。**
方向性移動 感測器 1→ 2	Mouse Hover + Mouse Move	恒亮後閃爍	在某外圍感測器上 A 或 B 區中為選定狀態後，接著移動到另一外圍感測器 A 或 B 區中被連續偵測到一定的次數。 **可辨識出下列 8 種基本手勢動作：** **(1)向上 (2)向下 (3)向右 (4)向左** **(5)向右上 (6)向左上 (7)向右下** **(8)向左上。**
回復上一層	Mouse Hover	恒亮後恒暗	在**中央感測器**上 A 區為連續出現兩次選定狀態。
回復至最初狀態	Mouse Hover + Mouse Down	恒亮後恒暗 綠色 LED 亮	在**中央感測器**上 A 區中選定狀態後，接著移動到中央感測器 B 區中被連續偵測到一定的次數。 或距離值由 2→3。

(六) 追蹤定位功能程式設計與性能測試

此功能主要將多數的感測器儘量自動地指向使用者的手部位置，以方便進行後續手勢辨識。我們的構想是，若**中央感測器**未偵測到手部而**外圍感測器**有偵測到物體落在手部操作範圍，控制伺服馬達往此外圍感測器方位移動。使用者將手移開偵測範圍(距離)、達一次數或是中央感測器已順利偵測到，則伺服馬達停止移動。主要步驟如下：

1. **設定角度座標系統：**參考數學的直角座標表示法，以左下角為原點，格式為(水平方向角度, 垂直方向角度)，左下角為(0^{o}, 0^{o})，左上角為(0^{o}, 180^{o})，右上角(180^{o}, 180^{o})，右下角(180^{o}, 0^{o})。如圖 7。
2. **設定移動方向：**採用中心點為原點(0, 0)。-1 代表左移或下移，0 代表方向不變，1 代表右移或上移。如圖 7。
3. **執行測距程式：**更新全部感測器的距離資料。使用預設距離或取得使用者的身體距離(D cm)後，再進一步調整手部偵測範圍。如圖 8 和表 3。
4. **邊界判斷**：先依序判斷距離數據是否位於偵測範圍內，如果是，則再進一步判斷接下來的移動是否會超過設定的感測邊界(角度)。如果都符合條件，則此方位上的 LED 將持續閃爍。
5. **移動伺服馬達**：控制伺服馬達往此方位移動特定角度。
6. **終止條件**：當使用者將手往後移或中央感測器偵測到，離開偵距距離，則停止定位。

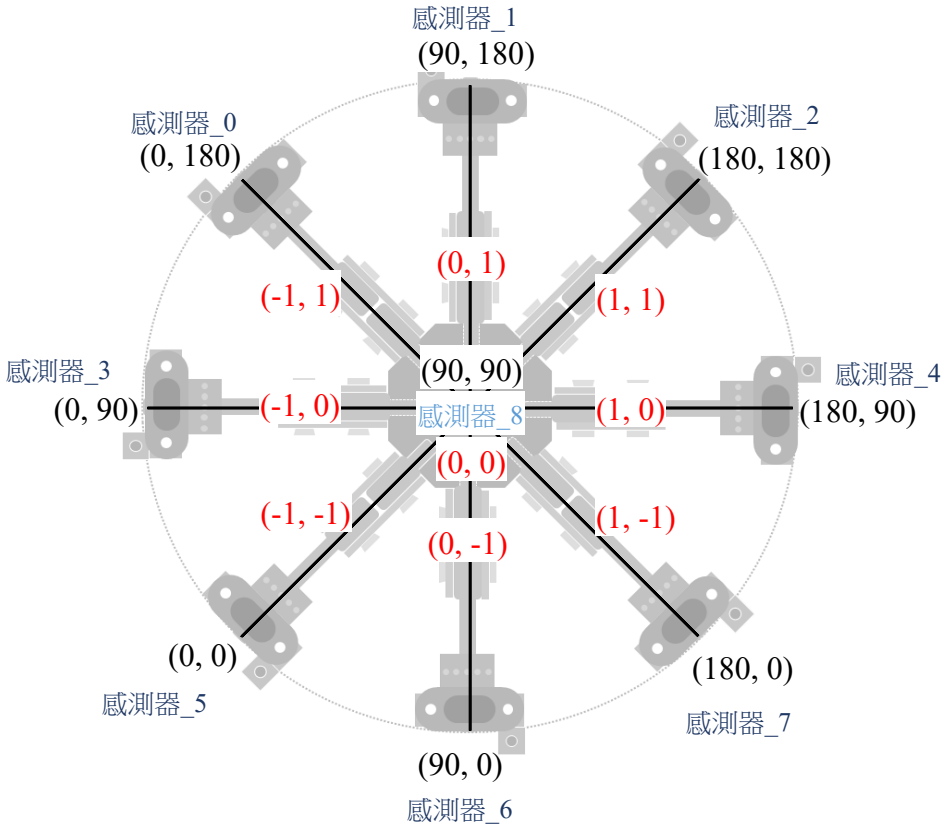


圖 7 感測器編號、角度座標系統和方向的設定

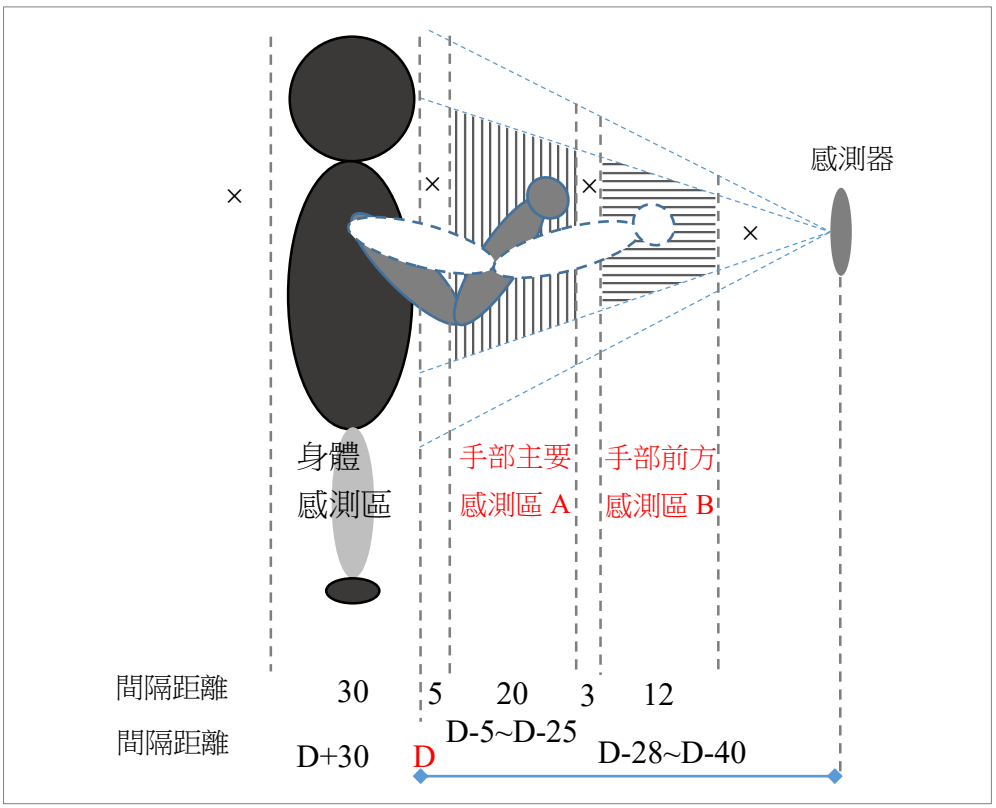


圖 8 感測區域示意圖。共分六個區，主要為手部 A 區和手部 B 區。

表 3 測量範圍設定值

區域	距離範圍 (D：身體前緣)	處理方式	距離值 調整
超出範圍(無作用區)	> D+30 大於 120cm	無作用距離值調整為 0。	0
身體區域	介於 D+30~D 之間或 介於 80~50cm 之間	將距離值調整為 1。	1
間隙(無作用區)	介於 D~D-5 之間或 介於 50~45cm 之間	無作用，將距離值調整為 0。	0
手部區域 hand_a **(A 區)**	介於 D-5~D-25 之間或 介於 45~25cm 之間	將距離值調整為 2	**2**
間隙(無作用區)	介於 D-25~D-28 之間或 介於 25~22cm 之間	無作用，將距離值調整為 0。	0
手部區域 hand_b **(B 區)**	介於 D-28~D-40 之間或 介於 22~10cm 之間	將距離值調整為 3	**3**
間隙(無作用區)	< D-40 小資 10cm	無作用，將距離值調整為 0。	0

(七) 手勢操作介面與模型的應用展示

1. **手勢操作介面：**本研究嘗試以常見的手勢操作方向、遊戲或網頁導覽的情境，設計出對應的感測器偵測方式，如表 4。藉撰寫程式進行測試，並進行動作碼編寫。

表 4 各種情境的手勢操作需求

進入基本方向模式	進入方向控制	進入瀏覽模式
向上	右轉	下一頁
向右上	左轉	上一頁
向右	加速前進	向左翻頁
向右下	正常速度前進	向右翻頁
向下	離開方向控制	回首頁
向左下		離開瀏覽模式
向左		
向左上		
離開基本方向模式		

2. 模型測距的應用：繪製即時動態雷達圖

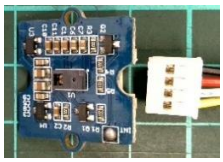
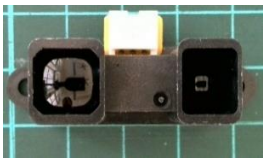
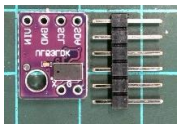
模型具有 8 顆外圍感測器，配合支架和立架，將可以即時測量不同位置的距離，可以進一步應用在防盜或是智能車上。利用試算表 Excel 的外掛程式 PLX-DAQ，透過序列埠，將測到的距離傳送到試算表中的特定欄位，並即時繪出動態雷達圖，具有視覺化的效果。

伍、研究結果

一、紅外線感測模組的測試與選用

(一) 基本規格比較

表 5 不同模組的基本規格比較表

模組名稱	Seeed Gorve-gesture (PAJ7620U2)	GP2Y0A02	VL53L0X	TOF10120
外觀				
尺寸(mm)	20 × 20	29.5×13×21.6	10.5 x13.3	20.0×13.2×2.0
紅外線雷射	940 nm VCSEL	940 nm VCSEL	940 nm VCSEL	940 nm VCSEL
測量距離(cm)	5-10	20-150	30~200	10~180
發射器角度/偵測器角度	60°	無數據	35 / 25	24 / 25
最高偵測速度	1200°/s	無數據	20ms	30ms
連線介面	I2C	類比輸入	I2C	UART / I2C
程式測試與設計考量	另行安裝程式庫 (Gesture_PAJ7620)	無需額外程式庫。撰寫程式將電壓值轉換成距離	另行安裝程式庫 (Adafruit_VL53L0X)	使用內建 Wire 程式庫
其他特性	可測 9 種手勢 (提供接線)	在接近模組位置需並聯電容器以提高供電穩定性	官方提供豐富測試數據文件	前方有特殊玻璃提供濾波測量(提供接線)

(二) 程式測試結果

模組代號：G：GP2Y0A02　V：VL53L0X　T：TOF10120

表 6 不同模組的基本規格比較表

測量距離	40cm			80cm			120cm		
模組代號	G	V	T	G	V	T	G	V	T
1	40	44	42.1	103	82.9	84	130	117.7	126.5
2	40	43.3	42	89	83	82.3	130	126.5*	126.5
3	41	43.9	41.8	102	82.8	80	130	123.7	126.5
4	40	43.1	41.8	102	83.3	83	130	126.5*	124.9
5	42	43.4	42	102	83.7	80.7	130	124.7	122.8
6	40	43.6	41.9	102	82.1	84.9	130	126.5*	131.1
7	40	43.8	42.2	102	83.3	82.7	130	123.8	128.4
8	40	43.5	42.2	103	83.5	81.6	130	117	123.5
9	41	43.5	42.2	102	84.5	82.3	131	122.9	127.1
10	41	42.6	42	105	81.7	82.8	134	126	120.5
11	40	43.4	42.1	101	84.3	83.3	130	118.1	127
12	37	43.7	42.1	102	83.6	82.6	130	114.7	128.3
13	40	43.4	42.2	103	84.3	83.9	130	126.5*	121.4
14	40	42.7	42.2	101	81.2	82.3	131	121	121.4
15	40	42.7	42.5	103	82	82.2	131	123.6	124.6
16	39	43.6	42.5	103	84.1	82.6	130	120.8	123
17	40	42.3	42.5	101	82.7	82	130	125.5	123
18	38	42.9	42.5	102	82.9	82	130	122.3	124.4
19	41	43.2	42.5	104	84.6	81.5	130	120.9	122.1
20	38	43	41.5	104	84	81.2	131	126.5*	120.2
平均值	39.9	43.3	42.1	101.8	83.2	82.4	130.4	122.8	124.7
標準差	1.2	0.5	0.3	3.2	1.0	1.1	0.9	3.6	3.0

註：*星號處數據超出其測量距離(out of range)，以略大的數值 126.5 代替。在實際使用上，可以透過程式去除掉上述的數據。

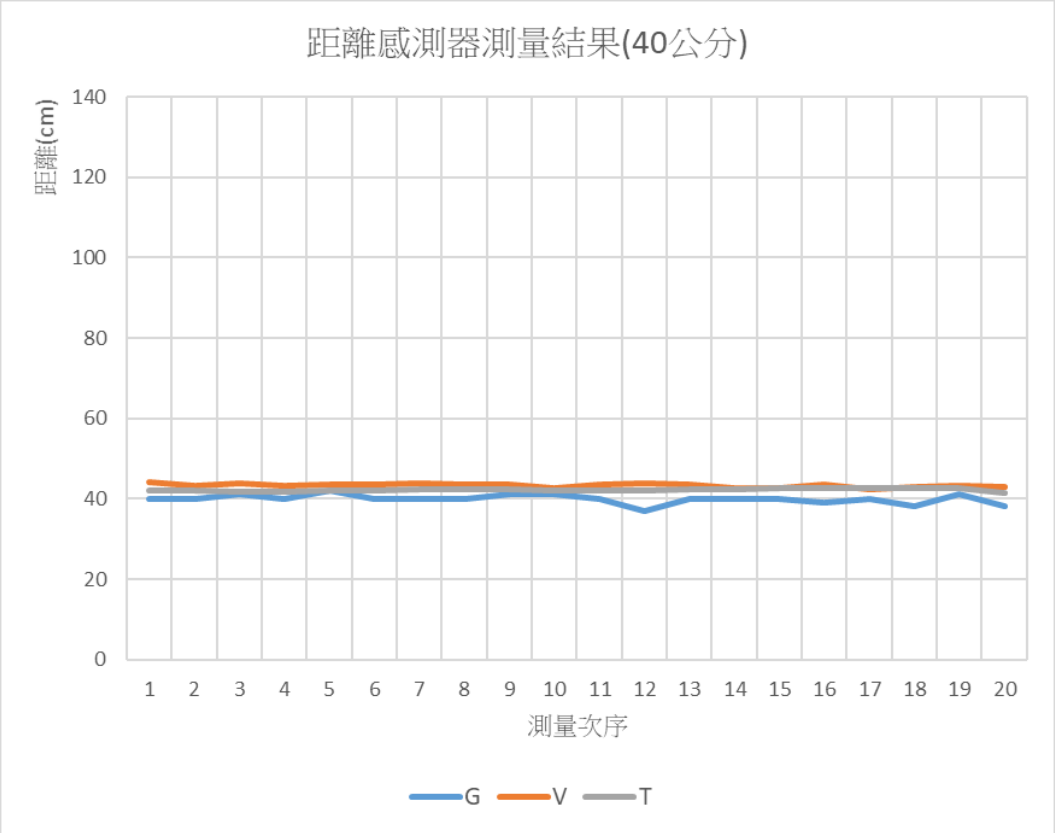


圖 9 三個模組在 40cm 下測量結果關係圖

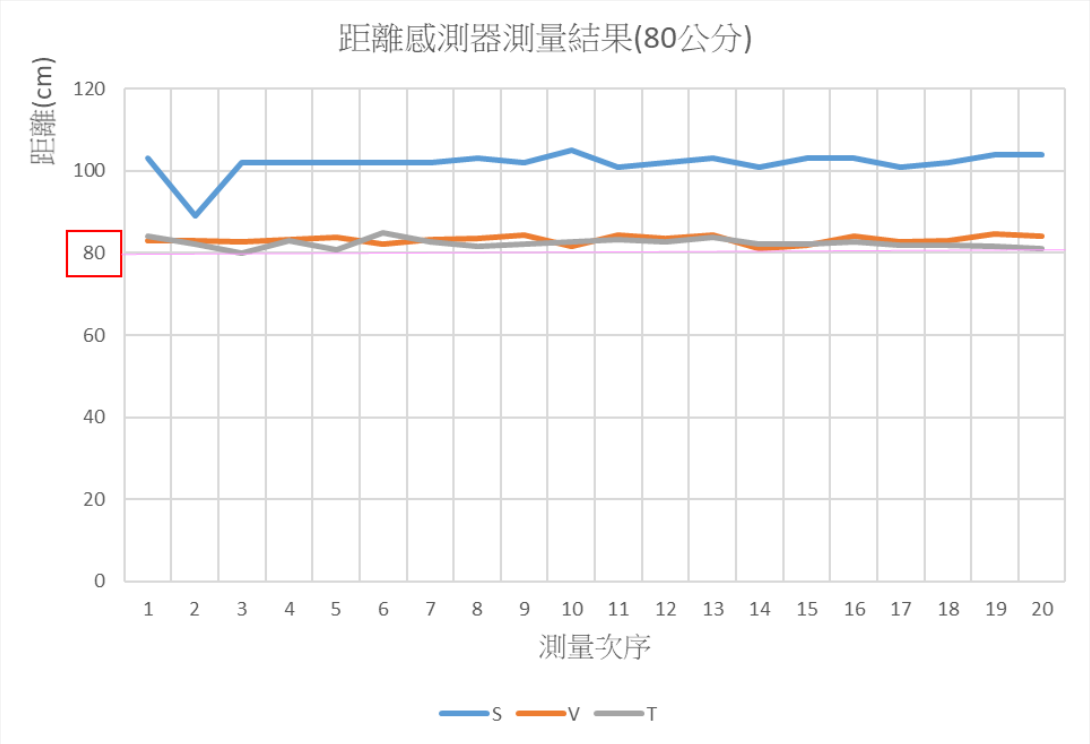


圖 10 三個模組在 80cm 下測量結果關係圖

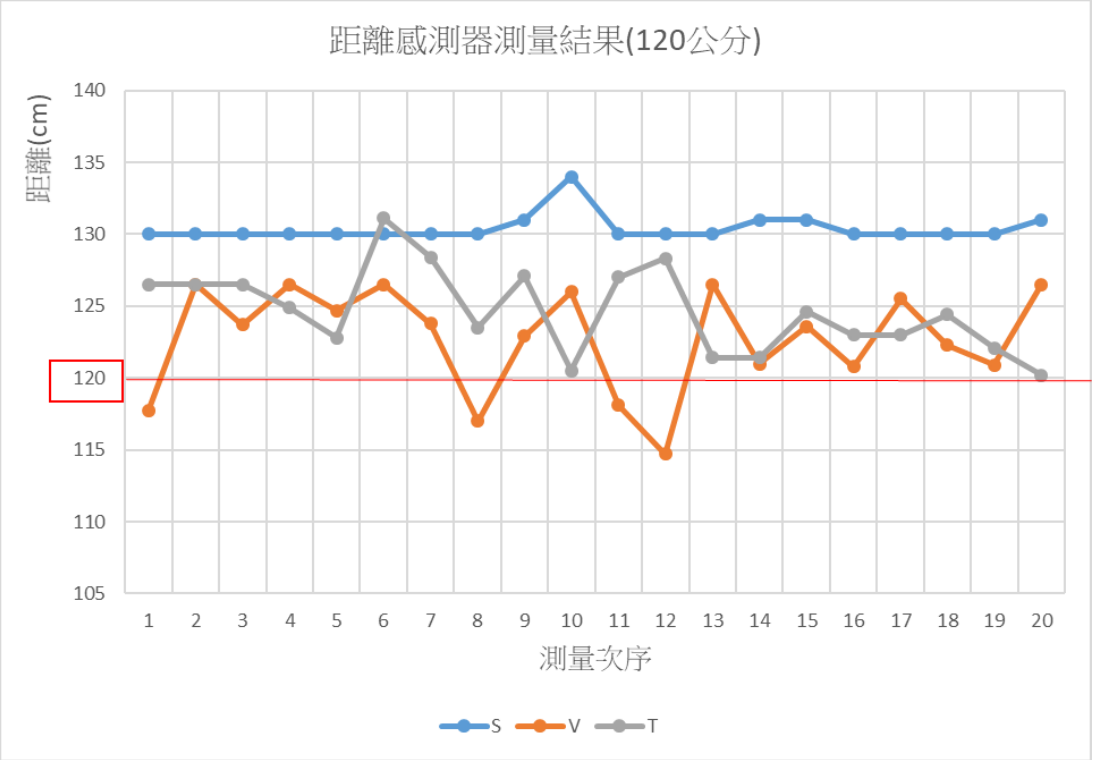


圖 10 三個模組在 120cm 下測量結果關係圖

結果分析：

1. 由表 7 的結果分析，隨著測量距離的增加，距離平均值的偏離與數值的分散程度(標準差)均有增加的趨勢，此結果與模組規格書提供的資料吻合。
2. 本研究使用網路取得基本範例程式進行初步測試，未進行調整和考量模組其他特性，結果並無法代表該模組的優劣，僅挑選符合本研究需求如下：連接的便利性(附線材)、尺寸較小、測量結果在可接受範圍的模組，即平均值接近且標準差較小者。
3. 因此，本研究選用 TOF10120 和 VL53L0K。手勢操作的範圍以 120cm 以內為主。

二、手勢操作的系統架構

本研究設計系統架構圖如圖 11。以 Arduino Mega2560 微控制器為核心，擁有較大的記憶體容量、更多的數位腳位和類比腳位(16 個)、支援 15 路 PWM。搭配擴展板，方便支援不同的連接端子的感測器或是裝置，相關的硬體功能說明如下。

1. **I2C 多工器**：使用 I2C 連線的裝置必需具有一個獨立的 I2C 位址，而市售的相同感測器預設的位址都一樣，會造成位址的衝突。而選用 8 通道的 I2C 多工器，可以最多同時連接 8 個 IR 感測器，在微控器與多工器之間只需透過一組 I2C 的線路連接，再

經由程式切換到不同的感測器，建立一對一的連線。使用多工器可以降低接線的複雜度，同時也不用更改每一個感測器的 I2C 位址。

2. **LED 顯示**：每一顆 LED 串連一顆 220 歐姆的電組，避免 LED 燒毀；使用數位腳位連接 Arduino 微控制器，以控制 LED 的顯示。將多顆電阻器和連接端子焊接在同一麵包板上，使用排線方式連接，以降低接線的複雜度。如圖 12。
3. **伺服馬達控制**：使用擴展板提供的 PWM 插座，連接具有 PWM(脈衝寬度調變)腳位，例如腳位 9 和 10，以控制多顆的伺服馬達(MG 90S)。
4. **動作展示**：使用手勢操作可以直接驅動伺服馬達或是 LED，以展示特定的動作。也可以進一步透過序列傳輸方式將使用者操作模式和 IR 感測器的數據或結果，傳送給其他機器或程式。

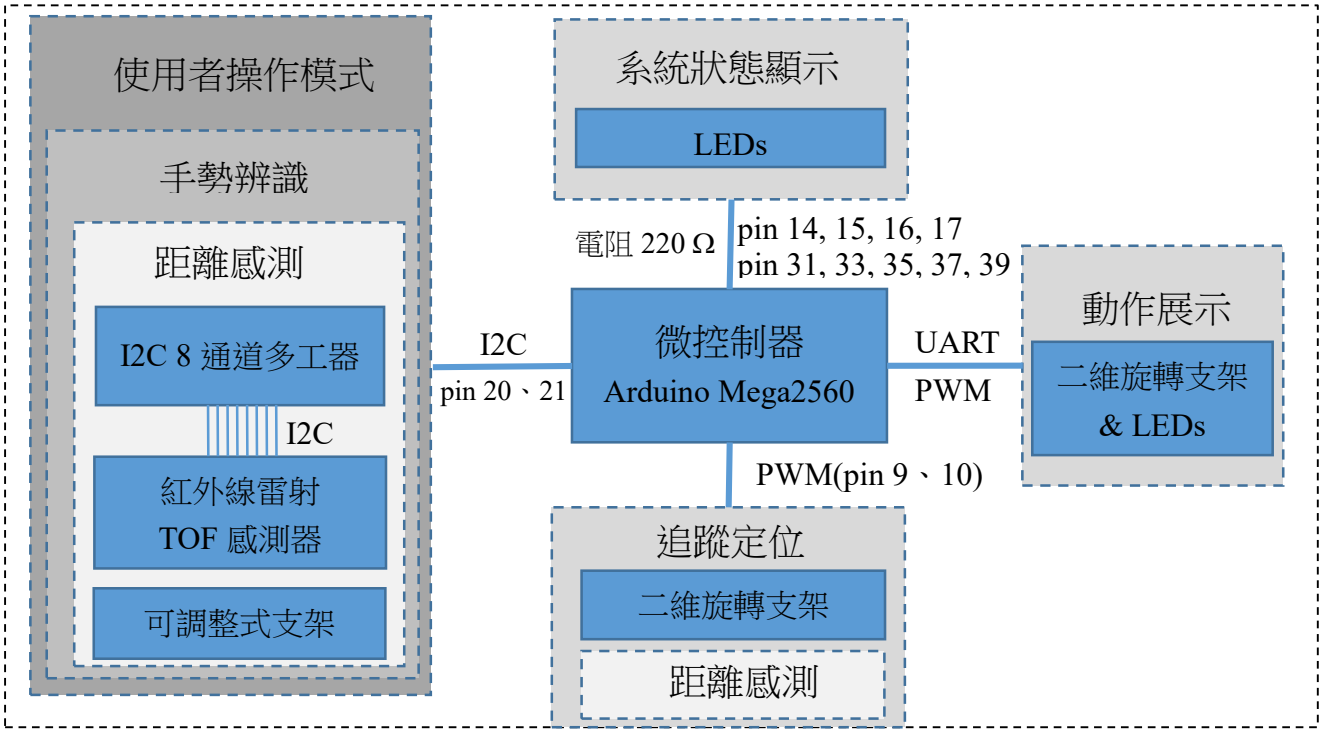


圖 11　系統架構圖

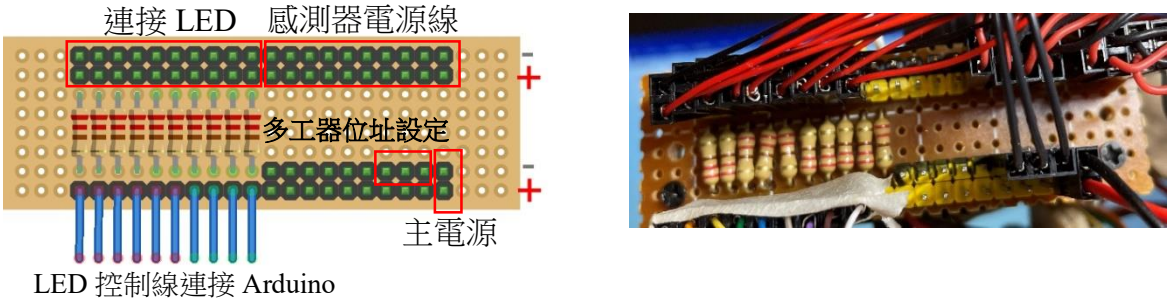
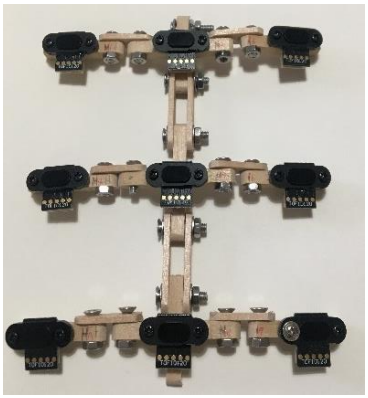


圖 12　LED 及感測器麵包板接線圖(左)、完成圖(右)

三、複眼結構可調整式感測器支架模型的設計與製作

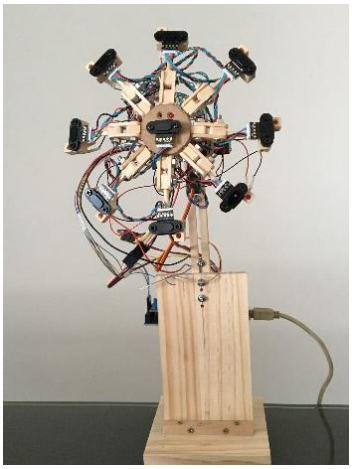


未彎曲

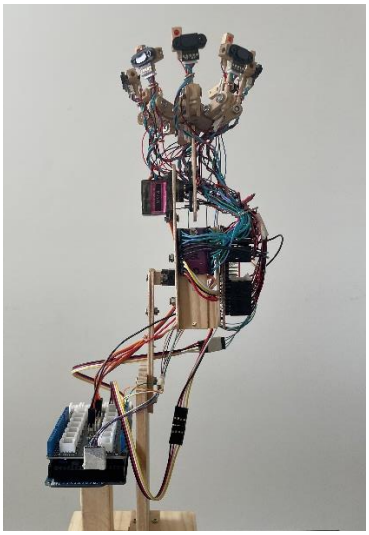
彎曲(改變偵測方向)

圖 13 第一代模型

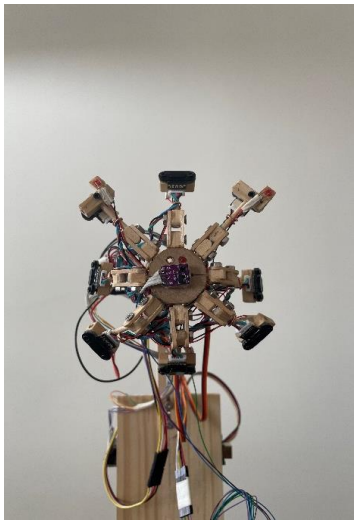
1.二維伺服馬達立架(側面)



2.感測器支架(正面)



3.垂直狀態(側面)，感測器可全部旋轉朝外可應用於即時動態雷達圖)



4.感測器分別指向不同方位(可偵測前方、上方、地面、兩側及後方。)

圖 14 第二代模型及支架的變化

結果說明：

1. 第一代模型感測器的缺點：不易改變安裝角度；四個邊角感測器彎折後偵測方向不易調整，感測器之間的距離也無法調整，如圖 13。因此修改為中心輻射狀的形狀類似昆蟲複眼結構的第二代模型，如圖 14。
2. 第二代模型的優點：
   (1) 具有 LED 顯示，可以顯示偵測的狀態或是展示手勢操作的結果。
   (2) 增加立架和伺服馬達，且有移動功能，可以水平和垂直方向各自移動 90 度。立架可調整高度和角度。
   (3) 感測器支架可以多方向彎折約 360 度，也可以旋轉 360 度。可完全配合研究的需要，調整偵測方向。
   (4) 感測器安裝角度也有三種彈性選擇，0 度、90 度和 180 度。感測器本身也可以 360 度旋轉。可配合研究的需要，調整偵測方向。

四、手勢操作系統的基本功能設計

(一)狀態顯示功能

```
int LED_NO[9]={14,15,16,17,33,35,37,39,31}; //LED 連接的 Arduino Mega 2560 腳位
//使與感測器相同編號(led_no)的 LED 恒亮(0)、恒暗(-1)或閃爍指定時間。

void led(int led_no, int delaytime){
  //0: 恒亮，-1:恒暗，  選定(焦點)：恒亮，其他：  閃爍
  if (delaytime == 0){
    digitalWrite(LED_NO[led_no], HIGH);
  }
  else if(delaytime == -1){
    digitalWrite(LED_NO[led_no], LOW);
  }
  else if(focus_a[led_no]==1 or focus_b[led_no]==1){   //選定(焦點)，恒亮。
    digitalWrite(LED_NO[led_no], HIGH);
  }
  else{
    digitalWrite(LED_NO[led_no], HIGH);
    delay(delaytime);
    digitalWrite(LED_NO[led_no], LOW);
    delay(delaytime);
  }
 }
```

圖 15 狀態顯示功能程式碼片段

結果說明：

1. LED 的閃爍預設時間設定值以 10ms 較佳。太短不易分辨，太長則可能會影響偵測的動作。選定成為焦點時，LED 恒亮，可以提醒使用者開始進行手勢的下一動作。
2. 程式可以彈性的在程式的不同位置被呼叫，再依不同的狀況使特定 LED 有不同的閃爍長短。

(二)測距功能

測距功能主要有四個函式：

- update_distance( )：**測距的主要程式**。呼叫 ReadDistance()、多工器和 ir_center()，依序取得每一個感測器的距離，依區域調整成 0、1、2、3 數值存入距離陣列。
- get_distance( )：讀取一個指定感測器，傳回值為距離，單位為公分。
- SensorRead( )：連接感測器，傳回 2 位元組的距離資料。(範例程式)
- ReadDistance( )：呼叫 SensorRead( )，將 2 位元組資料運算處理後以公分單位輸出。(範例程式)
- ir_center( )：請取中央感測器的距離，依區域調整成 0、1、2、 3 數值存入距離陣列。

```
//更新全部的感測器距離，依區域調整數值，存入 distance[ ]陣列中
void update_distance( ){
   for(int i=0; i<8; i++){
      Wire.beginTransmission(0x70); //控制多工器
      Wire.write(1 << i);                  //傳送 1 位元組，選擇 bus 0~7(連接感測器 0~7)
      Wire.endTransmission();
//將距離轉換成索引值 0 1 2 3
      dist = ReadDistance();
      if(dist < 80 and dist > 50){          //身體區  body
         distance[i]=1;
      }
      else if(dist < 45 and dist > 25){   //手部區  hand_a
         led(i, LED_DETECTED *2);
         distance[i]=2;
      }
```

```
      else if(dist < 22 and dist > 10 ){   //手部區  hand_b
         led(i, LED_DETECTED    );
         distance[i]=3;
      }
      else{                                //無作用區
         distance[i]=0;
      }
   }
   ir_center(); //更新中央感測器 sensor_8 (型號 VL53L0x)
}
```

圖 16 測距功能主要程式碼

測試結果說明：

1. 採用不同區域的閃爍長短，身體區因為會持續被偵測到，所以 LED 不閃爍；手部(a 區)為 20ms，較長。手部(b 區)為 10ms，較短。上述設定值可清楚分辨所在區域 a 或 b。
2. 中央感測器採用不同型號，使用不同函式庫，主要擔任 HOME 鍵的角色，所以寫成獨立的函式 ir_center()，可以單獨被呼叫執行。

五、手勢操作介面的設計

```
//焦點的判斷：由 distance[]計算每一個感測器累積的偵測到的次數，達一定值，則判
//斷為選定(焦點)。指令格式：區域代號+感測器編號。例如 A4：A 區+感測器 4。
//由兩個指令字串合併(長度為 4)，產生一個動作碼。
String update_count(){
   int count;
   //偵測完第一個焦點後，最大偵測到次數減半(加快)。
   if(focus_current.length()>=2){count = MAX_COUNT / 2;}
   else{count = MAX_COUNT;}

   for(int i=0; i<9;i++){
      switch(distance[i]){
         case 2:
            count_a[i]++;
            if(count_a[i] >= count){
               memset (distance, 0, sizeof(distance)); //清除陣列，重設為 0
               memset (count_a, 0, sizeof(count_a));
               memset (focus_a, 0, sizeof(focus_a));
               focus_a[i]=1;
               if(focus_current.length()>=4){focus_current="";} //限制長度
               focus_current += "A";
               focus_current += i;
               return focus_current;
```

```
            }
            break;
        case 3:
            count_b[i]++;
            if(count_b[i] >= count){
                led(i, 0);
                memset (distance, 0, sizeof(distance));
                memset (count_b, 0, sizeof(count_b));
                memset (focus_b, 0, sizeof(focus_a));
                focus_b[i]=1;
                if(focus_current.length()>=4){focus_current="";}//長度大於 4 則清空字串
                focus_current += "B";
                focus_current += i;
                return focus_current;
            }
            break;
    }
  }
}
```

圖 17 操作介面主要程式碼

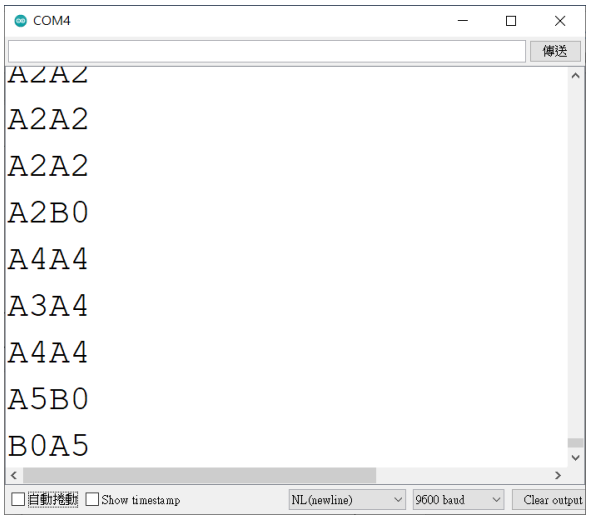


圖 18 操作介面程式動作編碼輸出畫面

測試結果說明：

1. 動作碼格式由連續兩次的偵測結果組合而成(例如：A2B0)，可以有效避免誤判的情況。如果只使用單次偵測結果，將可能因為偵測的速度過快，執行錯誤的命令。
2. 最大偵測次數(MAX_COUNT)的策略：第一次數值大(偵測慢)，像是**預備動作**；第二次數值小(偵測快)，像是**動態動作**，使用上，比較接近實際的手勢動作。測試結果，手勢較為流暢，也能被順利偵測出來。

六、追蹤定位功能程式設計與性能測試

(一) 二維支架伺服馬達的控制

伺服馬達的相關函式：

- void move_home( )：移動全部的伺服馬達到初始位置(90, 90)
- int current_pos(Servo &myServo)：可傳回某一顆伺服馬達目前位置(角度)
- void move_s(Servo &myServo, int s1, int s2)：移動指定伺服馬達(有變速功能)
- void move_xy(int x, int y) ：移動水平和垂直方向的伺服馬達至特定角度(無降速)
- **void move_servo(int x1, int y1, int x2, int y2, int velocity)**：主要函式。移動水平和垂直方向的伺服馬達至特定位置(角度)，採**逐漸降速**方式。

```
int INIT_X = 90;//初始座標
int INIT_Y = 90;//初始座標
//參考數學二元一次直線方程式。採逐漸降速移動，輪流移動 x 軸(水平)和移動 y 軸(垂直)各一步到終點，可走斜線。起點座標(x1, y1)，終點座標(x2, y2)
void move_servo(int x1, int y1, int x2, int y2, int velocity){
  //delta 的資料型態設為 float 避免除法運算時可能出現零，造成 servo 不移動。
  float delta_x = x2-x1;
  float delta_y = y2-y1;
  float delta_max = max(abs(delta_x),abs(delta_y));
  //下一個新座標
  int x;
  int y;
  //降速用的迴圈
  for(int i=0; i< delta_max; i++){
    x = x1 + i * (delta_x / delta_max);
    myservo_x.write(x);
    y = y1 + i * (delta_y / delta_max);
    myservo_y.write(y);
    delay(velocity);   //移動快慢控制
  }
  //記錄(更新)伺服馬達最後的座標；也可採用呼叫 current_pos( )取得。
  current_x = x2;
  current_y = y2;
}
```

圖 19 主要程式碼片段

測試結果：

1. 伺服馬達的移動快慢，呼叫時可以利用 velocity 值的調整。
2. 經測試結果 velocity 值為 0~2 之間，移動時較為流暢。

(二) 追蹤定位功能

追蹤定位的相關函式：

- bool isDetectArea()：//判斷 current_x 和 current_y 是否仍在偵測區(角度)內
- bool positioning()：**主要函式**。取得感測器資料，移動馬達，持續追蹤定位。

```
//追蹤定位功能：取得距離資料，移動馬達，持續追蹤定位。
const int DIR_X[8]= {-1, 0, 1, -1, 1, -1, 0, 1}; //代表不同方向
const int DIR_Y[8]= {1, 1, 1, 0, 0, -1, -1, -1}; //代表不同方向
const int BORDER_L = 30;       //X 軸左移邊界(角度)
const int BORDER_R = 150;      //X 軸右移邊界(角度)
const int BORDER_U = 150;      //Y 軸上移邊界(角度)
const int BORDER_D= 60;        //Y 軸下移邊界(角度)
int positioning_count = 0;
int MAX_COUNT = 50;
void setup(){
   myservo_x.attach(9);   //水平馬達初始化，接 arduino 腳位 9(具有 PWM 功能)
   myservo_y.attach(10);   //垂直馬達初始化，接 arduino 腳位 10(具有 PWM 功能)
}
bool positioning(){
   int rate= 0;   //移動速度控制(延遲時間)
   int steps =2; //移動速度控制(每次移動角度)
   update_distance( ); //更新距離。也可放在函式外執行。
   for(int i=0; i<8; i++){
      //手的位置某個感測器的(A)區或是//(B)區，則伺服馬達往此方向移動
      if(distance[i])==2 or distance[i])==3){
         if(isDetectArea(current_x + steps * DIR_X[i], current_y + steps* DIR_Y[i])){
            led(i, LED_DETECTED);
            move_servo( current_x, current_y, current_x + steps* DIR_X[i], current_y +
                        steps * DIR_Y[i], rate);
            positioning_count ++;
            //停止條件：超過設定偵測次數上限，則停止。
            if(positioning_count == MAX_COUNT){
               positioning_count = 0 ;
               return true;
            else{
               return false;
            }
         }
```

```
    }
  }
  home_key_check(); //檢查 HOME 鍵
}
```

圖 20 主要程式碼片段

測試結果：

1. 追蹤定位功能能幫忙不同的使用者，找到適合自已的手部操作位置，有助於後續的操作。實際使用發現，也可以帶來有趣的體驗。

七、手勢操作介面的展示與模型的應用

(一) 手勢操作介面的展示

表 7 方向控制模式的功能和手勢操作方式說明

功能	手勢操作	動作指令編碼
進入基本方向模式	一或兩顆外圍感測器 A 區為選定，則進入方向控制模式	A0B0
向上	編號 6 的 A 區選定後，編號 1 為 A 區為選定	A6A1
向右上	編號 5 的 A 區選定後，編號 2 為 A 區為選定	A5A2
向右	編號 3 的 A 區選定後，編號 4 為 A 區為選定	A3A4
向右下	編號 0 的 A 區選定後，編號 7 為 A 區為選定	A0A7
向下	編號 1 的 A 區選定後，編號 6 為 A 區為選定	A1A6
向左下	編號 2 的 A 區選定後，編號 5 為 A 區為選定	A2A5
向左	編號 4 的 A 區選定後，編號 3 為 A 區為選定	A4A3
向左上	編號 7 的 A 區選定後，編號 0 為 A 區為選定	A7A0
離開基本方向模式	中央感測器為選定，則離開方向控制模式	A8A8、A8B8

表 8 方向控制模式的功能和手勢操作說明

功能	手勢操作	動作指令編碼
進入方向控制	一或兩顆外圍感測器(例如：編號 1 的 AB 區為選定，則進入方向控制模式	A2B2
右轉	順時鐘方向感測器(編號 0 或編號 7) A 區有偵測到手部或選定	A3A0、A0A0 A4A7、A7A7
左轉	逆時鐘方向感測器(編號 5 或編號 2) A 區有偵測到手部或選定	A3A5、A5A5 A4A2、A2A2
加速前進	編號 3 或編號 4 為 B 區為有偵測到手部或選定	A3B3、B3B3 A4B4、B4B4
正常速度前進	編號 3 或編號 4 為 A 區為有偵測到手部或選定	B3A3、B4A4
離開方向控制	中央感測器為選定，則離開方向控制模式	A8A8、A8B8

表 9 網頁瀏覽模式的功能和手勢操作說明

功能	手勢操作	動作指令編碼
進入瀏覽模式	一或兩顆外圍感測器(例如：編號 2 的 AB 區為選定，則進入瀏覽模式	A3B3
下一頁	右方感測器編號 4 的 A 區有偵測到手部或選定	A4A4
上一頁	左方感測器編號 3 的 A 區有偵測到手部或選定	A3A3
向左翻頁	右方感測器編號 4 的 A 區為選定後，左方感測器 B 區編號 3 有偵測到手	A4B3
向右翻頁	左方感測器編號 3 的 A 區為選定後，品方感測器 B 區編號 4 有偵測到手	A3B4
回首頁	左上方感測器(編號 0)有偵測到手部	A0A0
離開瀏覽模式	中央感測器 A 區為選定，則離開方向控制狀態	A8A8、A8B8

結果說明：

1. 手勢動作可以被辨識出來，依測試發現，同一個功能，可以由有多個動作編碼，例如表 8 的右轉功能，當手部移動後新的位置後，正常會持續停留較長時間，會進一步產生另一個動作碼，因此，將這類後續的手勢動作，視為同一功能。

(二) 即時動態雷達圖

```
char cell_name[8] ={‘A’, ’B’, ’C’, ’D’, ’E’, ’F’, ’G’, ’H’} //試算表欄位
int ir[8] = {1, 2, 4, 7, 6, 5, 3, 0 } //感測器順時針編號
void setup(){
      Serial.begin(9600);
      Serial.println("CLEARDATA");
      Serial.println("LABEL, IR_1, IR_2, IR_4, IR_7, IR_6, IR_5, IR_3, IR_0");//欄位名稱
}

void loop(){
   int t = 50;
   for(int i=0; i<8;i++){
      Serial.println((String)"CELL,SET," + cell_name[i] +"," + get_distance(ir[i]));
   delay(t);
}
```

圖 21 主要程式碼

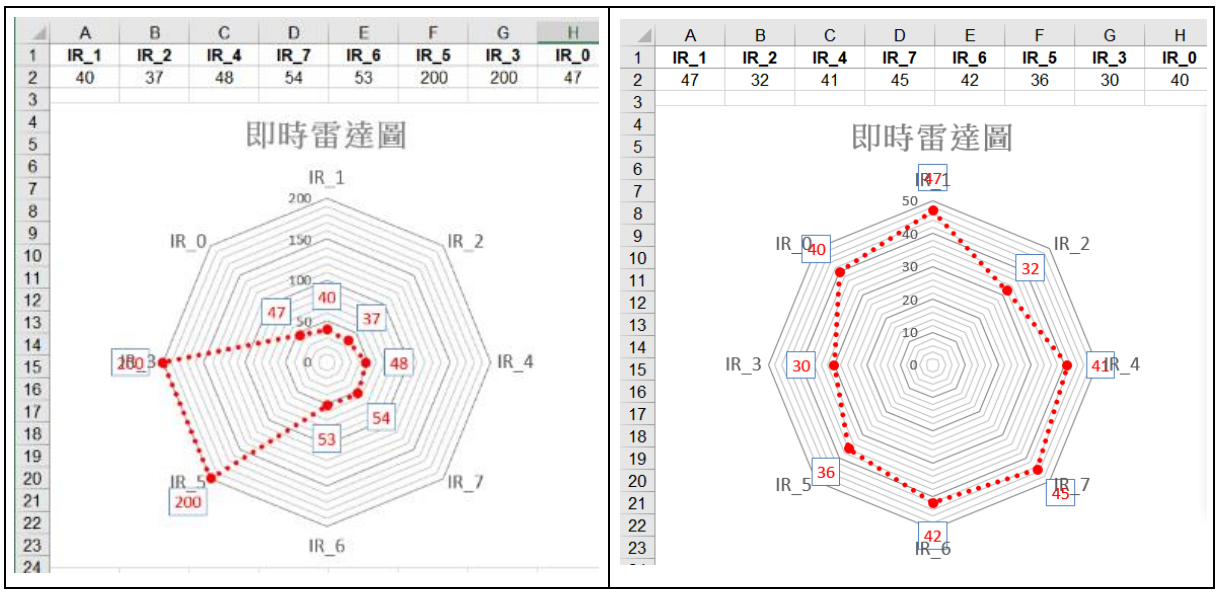


圖 22 在 Excel 中使用 PLX-DAQ 自動繪製即時動態雷達圖

結果說明：藉由多顆感測器和現成的軟體就可以呈現週遭的物體距離畫面

陸、討論

一、紅外線感測模組的測試與選用的問題

每個感測器均有不同的設計和準確度，例如 GP2Y0A02 建議並聯電容器或是程式碼微調；TOF10120 有濾波模式。而光源環境、物品表面顏色，都會影響測量結果。所以，基本上不容易明確的判斷優劣，只能依需求選擇。

二、感測器和 LED 數量對手勢動作辨識的影響。

根據模組的規格書，可準確測量的距離會受到測量時間的影響，多數模組均建議每一顆的測量所需時間大約在 30ms 左右，較長的距離和準確性，則需要較長設定較長的等待時間。本研究的感測器加上 LED 多達 18 顆，明顯會佔用辨識動作的時間。因此，本研究的手勢操作採用”停留”在某感測器上的方式，手勢動作偏慢但是較容易準確判讀。

三、感測器讀取順序、編號順序與資料儲存的問題。

因為最初模型為「王」字形，因此感測器編號、LED 編號、距離陣列、感測器讀取和 LED 顯示順序，都採用由左而右由上而下的方式。但在後續的程式撰寫中，卻也經常出現必須採用順時鐘順序來讀取、顯示或傳送，造成困擾。這部份有待後續的修改。

柒、結論

本研究共有以下幾點結論：

一、不同的紅外線感測模組有不同的特性，需依使用的距離和設計的需求來選用。

二、本研究提出的系統架構設計可以被實作出來，也助於程式的撰寫和硬體線路的製作。

三、本研究提出的複眼結構紅外線測距模型，具有充份的彈性，如同昆蟲複眼一般，可以指向任意想測量方位，提供更寬廣的視野。

四、測距功能的相關設定能符合實驗的需求，能將測量距離切割成不同的虛擬區域(有作用、無作用、身體、手部 A 區和手部 B 區)，能聚焦在手勢操作的兩個主要區域；而搭配 LED 的使用，使用者能清楚了解各個感測器的偵測狀態。

五、本研究提出的手勢操作介面設計，採用雙指令組合成一個動作編碼的方式，可以提供數十種以上的手勢操作組合。而操作介面設計具有一致性，配合使用情境選用直覺的操作指令，並提供了一鍵復原(返回)的 HOME 鍵設計，降低使用的複雜度。

六、追蹤定位功能除了能夠主動地偵測使用者的手部位置，能提高使用的便利性。

七、手勢操作系統和模型具有實用性，能支援多種的使用情境。

我們的模型製作和程式功能仍有很多改進的地方，例如為二維模型增加一個轉動方向；在材料方面，改用較厚較粗木材以提高耐用程度。狀態顯示上，使用雙色、全彩 LED 或是小尺寸 OLED 顯示器，可以顯示出更多的狀態；程式設計上，更快速或更複雜的手勢的辨識仍需要克服，這些是我們未來仍需進一步努力的方向。

捌、參考資料

1. 複眼。維基百科。https://zh.wikipedia.org/wiki/複眼。
2. 梅克・施密特(民 101)。Arduino 快速上手指南。(曾吉弘譯)。臺北市：泰電電業。
3. Arduino Cookbook 錦囊妙計第二版(徐德發譯)(民 101)。臺北市：碁峰資訊。
4. 資訊科技(一上)。翰林出版社。2020。
5. 圖形使用者介面。維基百科。https://zh.wikipedia.org/zh-tw/圖形使用者介面。
6. 使用者介面設計。維基百科。https://zh.wikipedia.org/wiki/使用者介面設計。