國立臺灣師範大學理學院

資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Science

National Taiwan Normal University

Master's Thesis

以 Flex 感測器為基礎的智慧手套設計

Smart Glove Design Based on Flex Sensors

邵蓁玓

SHAO, Zhen-Di

指導教授：黃文吉 博士

Advisor: Wen-Jyi Hwang, Ph.D.

中華民國 113 年 1 月

January 2024

致謝

在 MC2 實驗室學習的過程中，我獲益良多，在這裡我學到許多以前從未接觸過的知識。首先；我由衷感謝黃文吉老師，老師不辭辛勞的教導我許多有關硬體技術、神經網路、程式語言等等程式相關知識。當我在研究面臨到困難時，黃老師適時給予我許多鼓勵及幫助；時常與我討論研究上的問題，非常謝謝老師不僅花了許多時間的教導我許多在外界無法學習的知識，還給予我改進的機會，我很高興可以成為 MC2 實驗室的學生。

此外，要感謝我的家人，他們在我念碩士的這段期間，一直給予我無微不至的照顧和鼓勵，為我加油打氣，讓我在學業上可以更加專注。在實驗室的這段期間，也要謝謝實驗室的同學們，翔升、松恬、任慶、智翔及暄峒這幾位同學及實驗室學弟給我的幫助，由於我本身不是念資工系，因此在所了解的相關知識有限；但是，我很謝謝黃文吉老師、實驗室同學及學弟協助我解決提出的各種疑問，讓我可以從中學習，同時讓我學習到與他人共同合作的重要性。

最後，我要再次感謝黃文吉老師以及實驗室的同仁表達深深的謝意；如果沒有大家細心的指導與幫助，我無法取得今日的成就，感激大家給予我的祝福與支持，讓我可以順利完成碩士學業。

摘要

手勢辨識的技術目前逐漸廣泛應用於 AI 領域中，尤其穿戴式裝置的研究推動了這一領域的發展。透過穿戴式裝置的方式，來偵測手勢的動作並進行辨識；儘管目前有許多手勢辨識的研究，但鮮少有涉及到五隻手指感測器的研究，由於使用五隻手指來完成智慧手套會遇到各種不同的問題，例如：手套製作系統複雜或是選擇的感測器應用等等問題；因此，本論文將探討以精簡的方式製作具有五隻手指的智慧手套。

本論文使用了兩種主動式彎曲感測器為主體，分別以 Bend Lab 的 Digital Flex Sensor 和 Sparkfun 的 Qwiic Flex Glove Controller 兩種彎曲感測器來完成五隻手指的智慧手套；利用這兩種彎曲感測器不僅解決了手套感測器數量的問題，同時也降低了手套製作及系統的複雜度問題。最後，則透過神經網路來辨識出手勢的種類，以實現五隻手指的手勢辨識。

關鍵字：智慧手套、Bend Lab 的 Digital Flex Sensor、Sparkfun 的 Qwiic Flex Glove Controller

目錄

圖目錄

表目錄

第一章 緒論

本章節中會先探討研究背景、動機、困難及貢獻，分為四個部分；首先，在第一小節中，將介紹手勢辨識的相關研究及其應用領域上的運用，同時，我們也會探討手勢偵測過程中所使用的方法以及其兩種類型的智慧手套設計所使用的感測器。再來，第二小節會說明本研究的動機，受到哪些啟發而製作出本論文的智慧手套。最後，在第三和第四小節中，將會說明在使用感測器的過程中所遇到的困難，以及貢獻。

1-1 研究背景

隨著科技的進步，手勢辨識已廣泛應用在人工智慧的技術中，特別是在許多人機介面中，手勢成為一種直覺且自然的控制方式，例如：虛擬實境[1]、擴增實境[2]、物聯網[3]以及復健[4]等等相關領域的研究中廣泛的應用。

手勢辨識的方法主要分為影像處理、感測器偵測兩種方式來進行手勢辨識偵測。首先，在影像辨識方面，較常使用關鍵點進行辨識手勢辨識，如圖 1 所示；透過影像處理，可以準確捕捉手勢的動作，並準確地捕捉手勢細微的變化。雖然影像可以清楚看出手勢的動作及手勢的位置;但影像的資料較大而且拍攝的影像可能有涉及隱私權的問題。另一方面，圖 2 顯示了感測器的種類，利用感測器偵測手勢則是整合不同的感測器來偵測手勢動作，例如：三軸加速度器、陀螺儀[5]或是彎曲感測器來偵測手勢。

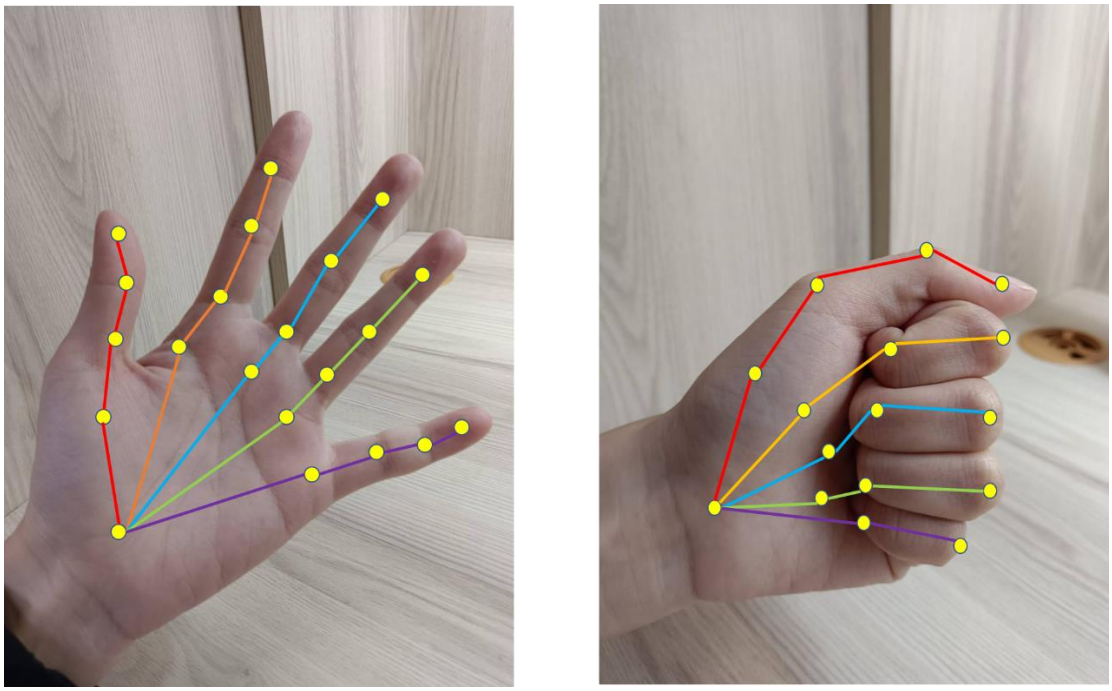
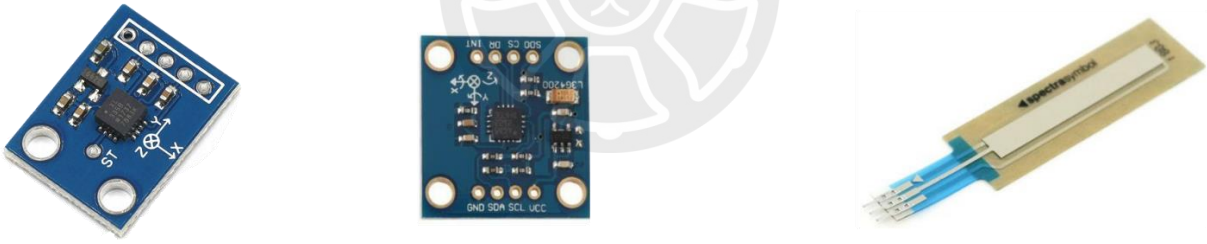


圖 1、透過手指骨架和關節的影像辨識，對於手指的彎曲程度和較細微的動作變化，影像也能準確捕捉。



(a) 三軸加速度器　(b) 陀螺儀　(c) 彎曲感測器 - SoftPot Membrane Potentiometer

圖 2、在感測器的實例中分別有使用 (a)三軸加速度器、(b)陀螺儀、(c)彎曲感測器，而在本實驗中，僅考慮彎曲感測器，沒有考慮三軸加速度器以及陀螺儀的原因是由於在使用智慧手套時，可以透過彎曲感測器就能夠來感測我們每隻手指的變化。

使用感測器來做手勢偵測，不會涉及隱私權的問題；但是，感測器來偵測手勢的缺點在於如果有相似的手勢動作，感測器可能會產生類似的訊號，使得辨識效果降低。相較於影像，由於感測器的偵測無法與影像一樣可以清楚看到手勢細微的變化，所以會導致不同的手勢可能會被辨識錯誤，進而影響手勢辨識的效果。

利用彎曲感測器做手勢偵測，與一般的三加速度器和陀螺儀相比，彎曲感測器更具有優勢，因為彎曲感測器可以獨立感測出每一隻手指的彎曲變化，每隻手指有各自的數據；進一步提升手勢辨識的準確性。同時，彎曲感測器可應用於製作智慧手套，可以有效感測出每隻手指彎曲的程度，可以實現靈活且準確的手勢辨識功能。

圖 3 以論文[6]、[7]所提出的智慧手套為例，這兩種智慧手套設計的方法截然不同。論文[6]提出的智慧手套設計，彎曲感測器是 Sparkfun 的 2.2 inch Flex Sensor 這是一款根據感測器彎曲的程度來改變電阻值的彎曲感測器，而使用的 Arduino 板是 LilyPad，LilyPad 是一種圓形的 Arduino 板，可以方便配置在像是衣物、配件等穿戴式物品上，且焊接口口徑較大比較容易焊接。藍芽部分則是使用 HC-08 的藍芽模組進行無線連線，藍芽的版本為 4.0，將這些模組用導電線全部縫製一起而成的智慧手套設計。

而論文[7]所設計的智慧手套，在彎曲感測器方面，則是使用 Bend Lab 的 Digital Flex Sensor (數位彎曲感測器)，這款感測器質地較軟，而且較耐用，感測方面也較為精準；Aruino 板則是使用 Bluno Beetle，此模組本身就內含藍芽功能，Bluno Beetle 的藍芽版本為 4.0；當藍芽程式碼上傳後，等待 Bluno Beetle 上的 TX 燈閃爍，表示可以傳送感測器的資料；因此只需要將藍芽接收器插在 PC 端上，LIHK 藍芽燈亮起表示感測器藍芽和 PC 端連線成功，不須額外再接藍芽模組，節省了硬體上的空間。因此可以看出論文[6]、[7]中在模組及應用上的差別。

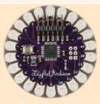
	論文[6]的智慧手套	論文[7]的智慧手套
		
Flex Sensor	Sparkfun 的 2.2 inch Flex Sensor 	Bend Lab 的 Digital Flex Sensor 
Arduino板	LilyPad Arduino 	Bluno Beetle (Bluetooth 4.0) 
Bluetooth	HC-08 Bluetooth (4.0) 	藍芽接收器(4.0) 

圖 3、介紹了論文[6]和[7]中智慧手套的元件。論文[6]使用 Sparkfun 2.2 inch Flex Sensor，感測器缺少數位處理器架構，相對複雜；而[7]使用 Bend Lab 的 Digital Flex Sensor 感測器則含有數位處理器架構，使用上更為簡便。

1-2 研究動機

由於使用彎曲感測器進行手勢偵測，不同手勢動作可能產生相似的訊號，導致手勢容易辨識錯誤。在[6]、[7]兩篇論文中所製作的智慧手套僅使用三隻手指，也就是三個彎曲感測器進行手勢辨識，這可能導致當手勢相似度較高時，辨識效果較差。因此，本研究的目標是製作一款智慧手套，希望至少搭配五隻手指以提升手勢辨識率。

在手勢辨識的研究領域中，較少有人做到五隻手指來進行偵測。由於實現五隻手指的感測器會引入複雜的管理系統和協調問題，例如一隻手套上可能需要搭載多個彎曲感測器，增加系統的複雜性等等。

1-3 研究困難

在製作智慧手套時，必須考慮彎曲感測器的配置和手指長短不一的問題，以避免配置感測器時出現手指長度與感測器長度不匹配的情況。舉例來說，拇指的

手指結構相對較短，如果使用和其他四隻手指相同長度的彎曲感測器放置在拇指上，可能因感測器過長而無法與拇指匹配，並且在長時間動作中容易導致感測器斷裂，影響感測器的使用壽命。

此外，感測器分為主動式彎曲感測器和被動式彎曲感測器兩種，因此在選擇感測器時，不僅需要考慮感測器配合手指的長短問題，還需思考所選擇的感測器是主動式還是被動式。被動式彎曲感測器在智慧手套製作上較為複雜，因此在本論文中，我們皆選擇使用主動式彎曲感測器；這些感測器整合了硬式和軟式的兩種型態，以實現更靈活的手套設計，同時進行手勢偵測。這種結合的設計主要希望可以提高智慧手套的適應性和精確性，使其更適合不同手指結構和動作需求。

1-4 研究貢獻

由於智慧手套的在論文[6]、[7]中，都只有使用三隻手指來進行偵測，因此為了希望能夠提升手勢的辨識效果，本論文製作了五隻手指的智慧手套，具有以下優點：

1. 智慧手套製作：

   選擇結合硬式 Sparkfun 的 Qwiic Flex Glove Controller 和軟式 Bend Lab 的 Digital Flex Sensor 兩種彎曲感測器，實現了五隻手指的智慧手套製作。透過這兩款彎曲感測器，成功解決了在五隻手指上感測器數量太多的問題，也解決了拇指感測器配置的難題，以及避免了感測器之間地址互相衝突等問題，這樣的智慧手套設計使得整個手套的感測器結構更加簡潔，同時確保了有效的手勢辨識。

2. 手勢資料收集：

   透過五隻手指感測，本論文中定義了五種手勢來進行手勢偵測；透過彎曲感測器能夠收到每隻手指的數據，在偵測方面可以更加精確感測到每

隻手指在動作時的變化；使我們可以收集到更完整的手勢資料，這樣有助於提升手勢的準確性。

3. 提升手勢辨識率：

利用五隻手指的感測器來進行偵測，和僅使用三隻手指的辨識效果相比，五隻手指在動作時感測到的資料更為明顯，以提高手勢辨識率。相較於僅使用部分手指進行感測，五隻手指的辨識率使得我們能夠更明確地分析手部動作的細微變化，使系統更能夠辨認和區分各種手勢。

第二章 感測器介紹

在本章節中會介紹不同彎曲感測器之間的差異，進一步了解感測器之間的特性；接著，將探討本研究中智慧手套所使用的彎曲感測器與論文[6]、[7]智慧手套所使用的彎曲感測器差別，並比較這幾種彎曲感測器的特徵，以深入了解感測器的相關內容。同時也會說明感測器之間是如何進行通訊；最後，將會說明是用何種標記方法來標記手勢。

2-1 主動式感測器與被動式感測器

主動式感測器和被動式感測器[8]差別在於主動式彎曲感測器具備主動提供資料或訊號，然後再透過 Bus(例如：I^2C、SPI 等等通訊協定)，將資料傳給 CPU；主動式彎曲感測器在本論文中以 Bend Lab 的 Digital Flex Sensor 和 Sparkfun 的 Qwiic Flex Glove Controller 為例。另一方面，被動式彎曲感測器不像主動式彎曲感測器一樣可以主動提供資料或訊號，被動式彎曲感測器是透過外部彎曲的壓力來產生電阻值，我們以電阻式彎曲感測器 Sparkfun 的 2.2 inch Flex Sensor 為例[9]，電阻式彎曲感測器是一個典型的例子，會依據感測器彎曲的程度而改變電阻值[10]，但是這種彎曲式的感測器可能會因為材料的差異和製造過程等因素可能導致感測器即使在相同的彎曲程度下，電阻值未必相同。

這兩種彎曲感測器通常用於虛擬實境、動作捕捉等等偵測用途；但相較之下，由於被動式彎曲感測器結構較簡單，其感測器功能有一些限制；因此可能需要添加其他額外的電路來輔助使用，使得在製作智慧手套方面相對會較複雜；因此，在本論文中，我們皆使用主動式彎曲感測器來完成智慧手套製作。

2-1-1 類比被動式彎曲感測器

類比被動式彎曲感測器，在本論文中以圖 4 所示的類比被動式彎曲感測器，Sparkfun 的 2.2 inch Flex Sensor 為例；這款感測器本身是一個可變電阻，電阻值和感測器彎曲程度呈正比關係，會根據感測器彎曲的程度而改變電阻值[11][12][13]，這種彎曲感測器通常成本較低，通常較適用於需求相對簡單的 IoT 應用。同時，在感測方面也有良好的靈敏度，但是為了保持感測器使用的穩定性，防止感測器鬆動影響感測，需要將感測器底部的接腳固定；確保感測器可以維持在一個穩定的位置，保持感測器彎曲的數值在感測器範圍內。

類比被動式彎曲感測器在使用方面較複雜而且其電阻值未必準確，如圖 5 所示；在感測器接線方面，要將被動式彎曲感測器應用在感測的訊號當中，需要利用額外的電阻及類比數位轉換器(Analog-to-Digital Converter, ADC)才能感知訊號。同時，被動式感測器在電壓供應方面也非常敏感；當電壓不足的情況下，電壓會急遽下降，即使感測器在相同的彎曲程度下，感測器的電阻值也會因電壓變化而不同，使得辨識過程較不穩定。

Sparkfun 的 2.2 inch Flex Sensor 其結構是由 Segmented Conductor(分段導電)、Conductive Link(導電連結)兩個元素來組成[14]，如圖 6 所示；Segmented Conductor(分段導電)主要功用是當感測器彎曲時，彎曲部分的分段導電區域會被伸展或壓縮，而改變電阻值，用於量測感測器的彎曲程度。而 Conductive Link(導電連結)是連接分段導電的元素，形成整體的感測器結構。Flex Sensor 的基板通常使用酚醛樹脂(人造樹脂)製成，由於感測器的基板是使用人造塑膠製成，因此在長時間使用的狀態下，可能導致感測器基板有容易有斷裂的問題。

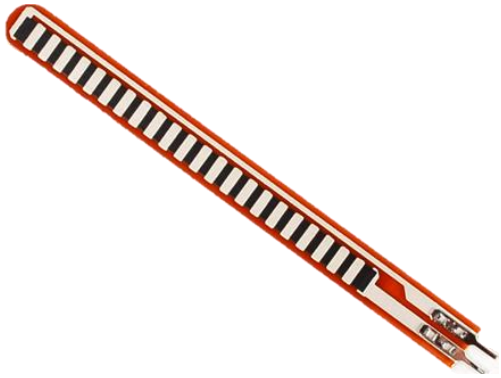


圖 4、Sparkfun 的 2.2 inch Flex Sensor 是一種被動式彎曲感測器，不具備數位處理器架構。感測器本身為可變電阻，彎曲程度與電阻值成正比；其成本較低，適合簡單的 IoT 應用。

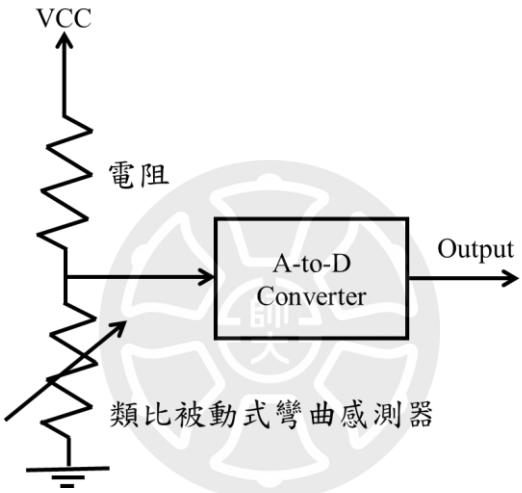


圖 5、Sparkfun 的 2.2 inch Flex Sensor 使用原理電路圖，由於是類比被動式彎曲感測器，所以要應用在感測的訊號當中，還需要額外添加電阻與類比數位轉換器才可以測到訊號。

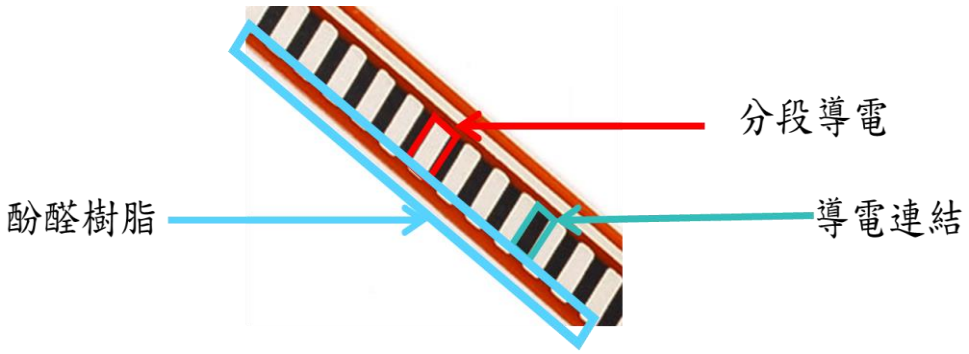


圖 6、Sparkfun 的 2.2 inch Flex Sensor 由紅色區的分段導電、綠色區的導電連結以及藍色區的酚醛樹脂構成。手指彎曲時，分段導電伸展或壓縮，改變電阻值。

Sparkfun 的 2.2 inch Flex Sensor 中的 Segmented Conductor(分段導電)是由石墨電阻組成，石墨電阻的表面含有導電粒子，使其能透過感測器彎曲的程度來改變電阻值；以圖 7、圖 8 為例，當感測器是在伸直狀態下，電阻值約為 30 kΩ。而當感測器彎曲大約 90 度時，感測器的導電粒子分開會使得電阻值改變，電阻值則改變為 50 ~ 70 kΩ[15]。

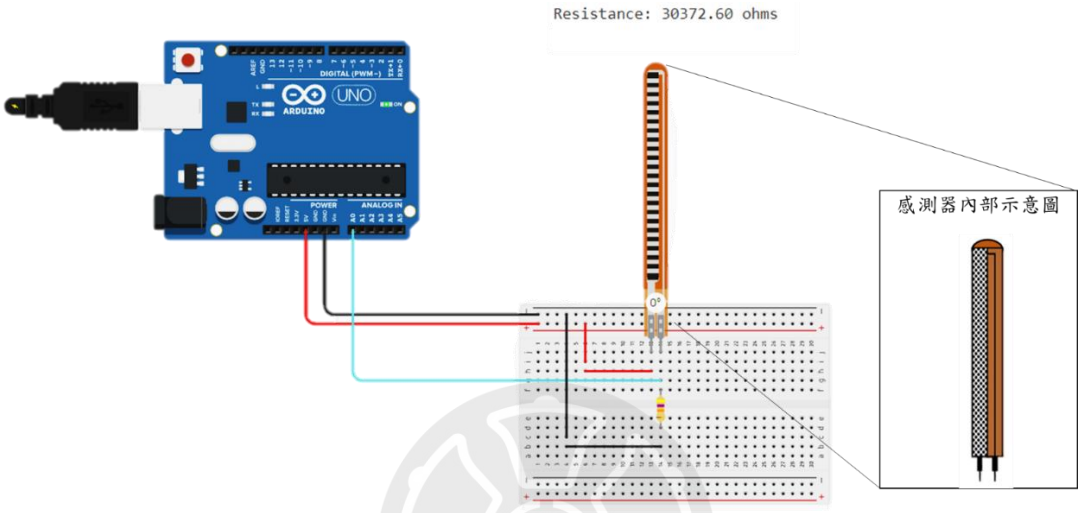


圖 7、透過 Arduino 測試 Sparkfun 的 2.2 inch Flex Sensor，感測器伸直時電阻約 30 kΩ。石墨電阻表面的導電粒子緊密相連，電阻值保持穩定。

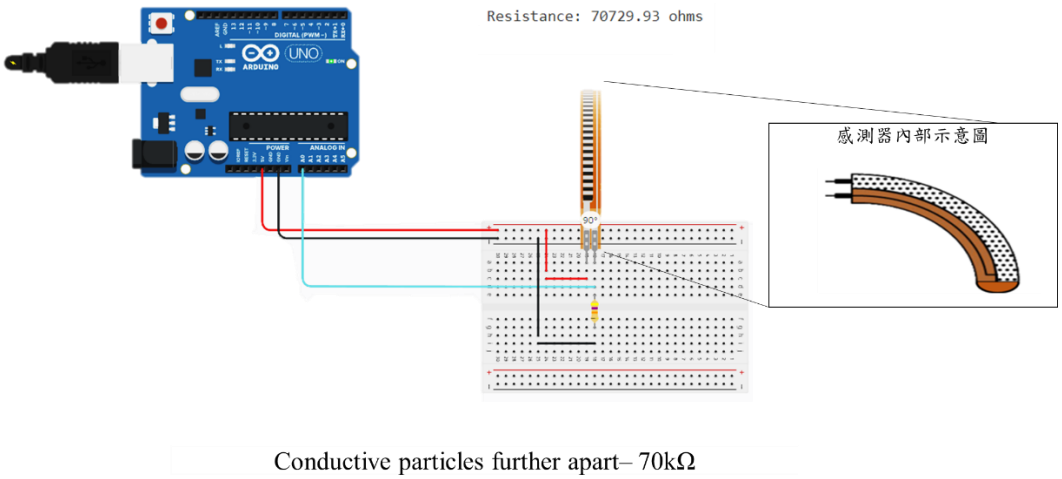


圖 8、當感測器呈現 90 度彎曲時，石墨電阻表面的導電粒子分開，導電粒子之間距離較遠會改變其感測器的電阻值，感測器彎曲 90 度時電阻值約 70 kΩ。

2-1-2 數位主動彎曲感測器

在本論文中，分別以 Bend Lab 的 Digital Flex Sensor 和 Sparkfun 的 Qwiic Flex Glove Controller 為例，這兩種彎曲感測器皆屬於數位主動式彎曲感測器。

2-1-2-1 Bend Lab 的 Digital Flex Sensor

Bend Lab 的 Digital Flex Sensor，屬於主動式彎曲感測器，如圖 9 所示，這種感測器內含數位處理器會主動提供資料，且此彎曲感測器具有高精確度量測，也同時保持低功耗；有較高的耐用性，感測器質地較柔軟[16]，製作在智慧手套上適用於每一隻手指。但是 Bend Lab 的 Digital Flex Sensor 的價格較為昂貴，假設以 Bend Lab 的 Digital Flex Sensor 來做五隻手指的彎曲感測器可能會造成感測器數量太多、手套系統較複雜、成本太高等問題。

Bend Lab 的 Digital Flex Sensor 是由兩個從中心軸偏移並延伸到感測器整個長度的柔性電容製成，是根據感測器的彎曲來改變電容值；因此只對彎曲的動作有偵測反應，像是拉扯或是扭轉等動作感測器不會偵測到訊號變化[17]。

圖 9、此彎曲感測器為 Bend Lab 的 Digital Flex Sensor，感測器內含有數位處理器架構，感測器可以主動提供訊號，且感測器質地柔軟、在感測方面較為精準，但成本較高且已停產。

2-1-2-2 Sparkfun 的 Qwiic Flex Glove Controller

Sparkfun 的 Qwiic Flex Glove Controller，也是數位主動式彎曲感測器，將兩隻柔性彎曲感測器和 Qwiic 板合併而成[18]，如圖 10；在一個主板上連接了兩支彎曲感測器[19]，這款彎曲感測器不管是在接線方式還是製作在五隻手指的智慧手套，都精簡了許多五隻手指的彎曲感測器實作上的限制；雖然和 Bend Lab 的 Digital Flex Sensor 感測器相比，感測器質地稍微較硬，但此感測器也具有良好的耐用性。

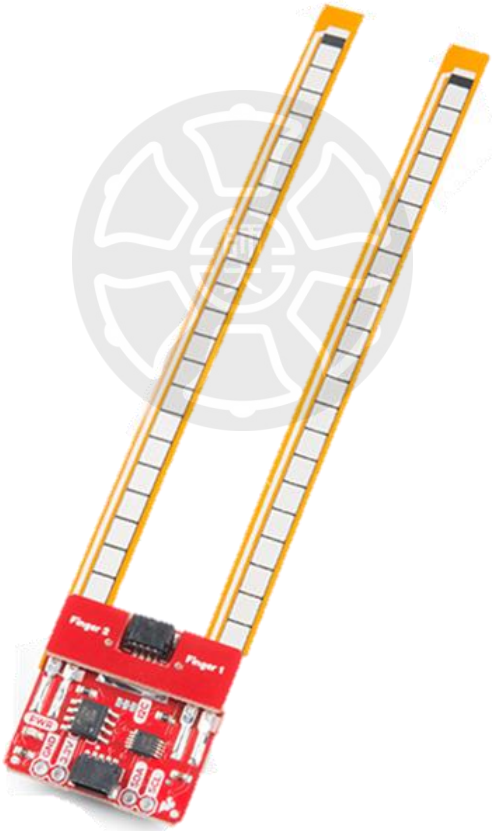


圖 10、Sparkfun 的 Qwiic Flex Glove Controller 為數位彎曲感測器，可主動提供訊號。雖然感測器質地較硬，但優點在於一個主板上即搭載兩隻彎曲感測器。

2-2 感測器與 CPU 整合：I^2C通訊協定

本研究在製作智慧手套方面，採用了主動式彎曲感測器來進行；由於主動式彎曲感測器不僅能主動提供感測器資料，而且感測器本身具有數位處理器架構，並支援I^2C通訊協定，因此主動式彎曲感測器主動提供的資料會透過I^2C將資料傳送給 CPU。圖 11 所示為利用I^2C所架設的通訊協定簡單實例，由於 Master(代表 CPU)與 Slave(代表主動式彎曲測器)之間必須要使用 SDA(訊號線)和 SCL(時脈線)連接，感測器之間才能夠互相通訊。其中 SDA 為雙向資料線、SCL 為時脈線[20]。透過 SDA 雙向的資料線傳輸，Master 和 Slave 之間的資料可以互相傳輸。通常傳輸資料的狀況下，是由 Slave 將量測的資料傳輸給 Master，並且利用 SDA 序列傳送的特性，將資料依序以一個個 bit 的形式來傳輸，而 SCL 為單向傳輸，是由 Master 傳輸給 Slave，所以當 SCL 訊號為 0 時，可執行資料寫入，訊號為 1 時則 Master 可執行資料讀取，負責同步數據的傳輸；也就是協調時間來傳輸資料。因此 SDA 和 SCL 兩者必須是同步進行，以實現正常的資料傳輸，僅須透過 SDA 和 SCL 兩條線便能夠有效降低硬體部署的複雜度。

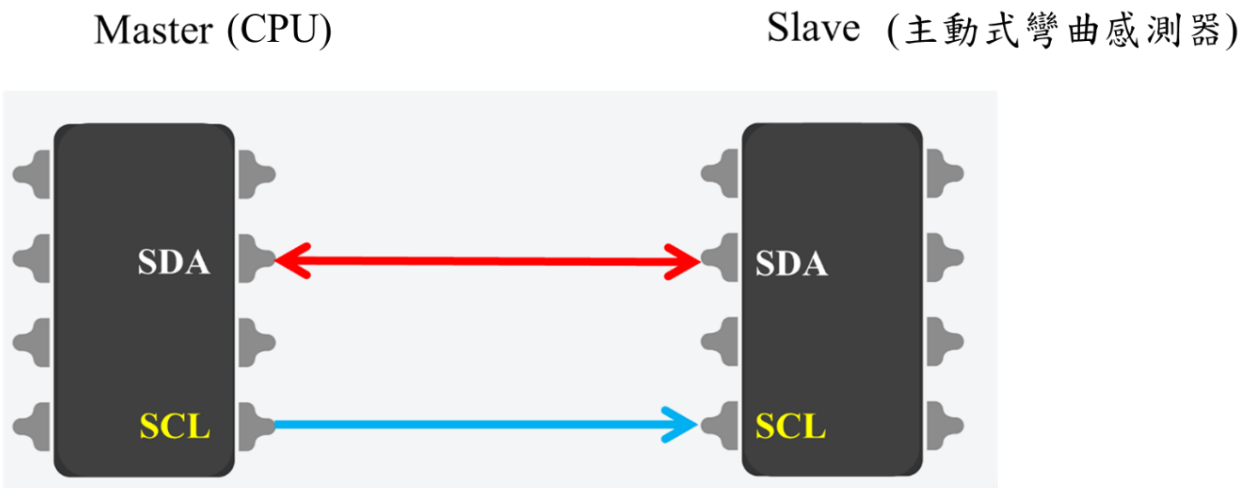


圖 11、利用I^2C所架設的簡單實例，其中 Master 為智慧手套的 CPU，而 Slave 為手套中的彎曲感測器。

I^2C協定只要感測器地址在不會互相衝突的狀況下，可允許 Bus 上連接多個 Master 和 Slave[21]，在本實驗中使用了一個 Master 和三個 Slave 來完成手套設計，如圖 12 所示。然而，在I^2C中使用了兩個 pull up 電阻，這是使I^2C可以正常運作的部分；由於I^2C是 open-drain(開漏架構)的方式來驅動 SCL 或 SDA[23]，表示每條訊號線的輸出都只能被拉到低電平，而無法把輸出拉到高電平，需要利用上拉電阻的方式，將低電平拉回至高電平。因此，在沒有訊號傳輸時，SCL 和 SDA 會被拉回高電平狀態，以保持I^2C協定的穩定性，防止通訊錯誤等等問題。

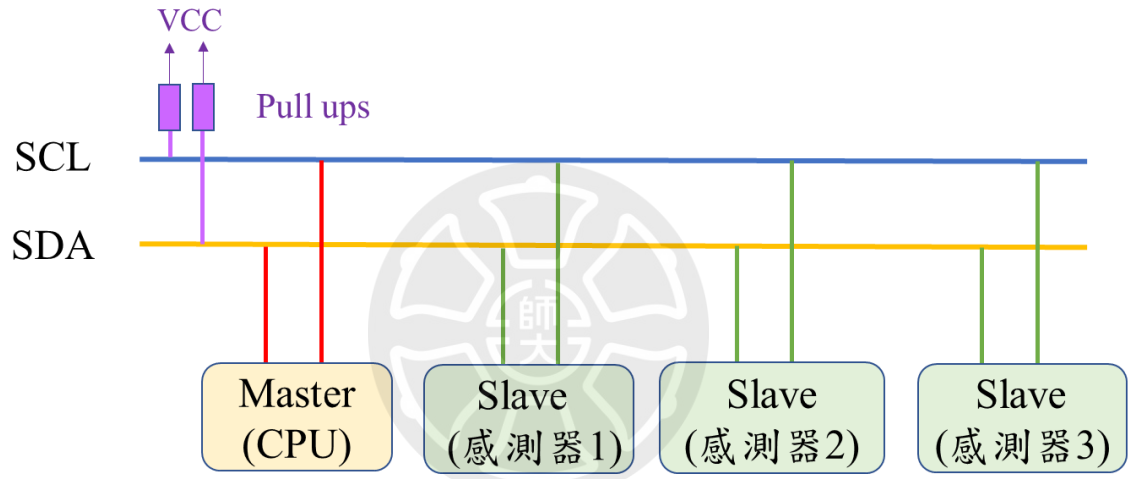


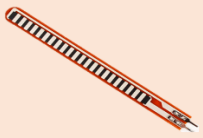
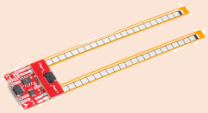
圖 12、利用I^2C所架構的實例應用在智慧手套中，CPU 擔任 Master 的角色，而三個彎曲感測器則作為 Slave 的部分。透過I^2C通訊協定，Master 可與每個 Slave 感測器之間進行通訊。

2-3 感測器比較

透過上述的三種彎曲感測器，摘錄於表格 1，我們討論了 Sparkfun 的 2.2 inch Flex Sensor、Qwiic Flex Glove Controller 和 Bend Lab 的 Digital Flex Sensor 這三款彎曲感測器支援不同的功能、成本比較以及各個彎曲感測器的相關結果比較。雖然 Sparkfun 的 2.2 inch Flex Sensor 在價格上較為便宜，但是；此類比被動式彎曲感測器沒有支援I^2C功能，而且這款感測器接線困難、抓取的訊號也較不穩定，因此使用類比被動式彎曲感測器製作在智慧手套上相對複雜。

而數位主動式彎曲感測器分別是 Bend Lab 的 Digital Flex Sensor 和 Sparkfun 的 Qwiic Flex Glove Controller 感測器兩者皆有支援I^2C功能，使多個彎曲感測器之間可以方便互相通訊，降低智慧手套製作的複雜度；其中，Bend Lab 的 Digital Flex Sensor 質地較軟，適用於每隻手指，但是如果要使用 Bend Lab 的 Digital Flex Sensor 做五隻手指的話，會造成手套上的感測器數量太多，I^2C系統複雜等問題。而 Sparkfun 的 Qwiic Flex Glove Controller 適用於後面四隻手指，因為 Sparkfun 的 Qwiic Flex Glove Controller 一個小板上有兩隻彎曲感測器，較不適合放在拇指，如果放在拇指上會使得感測器壽命縮短、容易斷掉。

表格 1、彎曲感測器的介紹，此部分說明了感測器所支援的功能、各個感測器的成本以及感測器的優缺點比較。

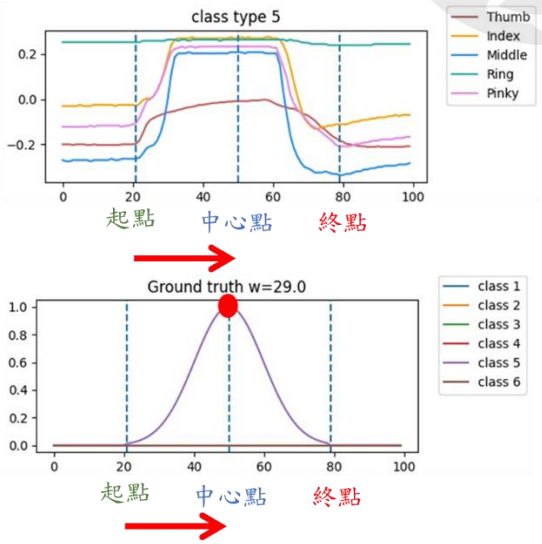
主動式或被動式	類比被動式彎曲感測器	數位主動式彎曲感測器	
Sensor 性質比較	Sparkfun的 2.2 inch Flex Sensor	Bend Lab的 Digital Flex Sensor	Sparkfun的 Qwiic Flex Glove Controller
			
支援I^2C功能	無	有	有
感測器成本	較便宜	較昂貴	價格中等
優點	低功耗、 價格便宜	低功耗、耐用性久、感測器質地較軟、適用於每隻手指	低功耗、一個主板上有兩隻感測器、降低手套製作複雜度，適用於後面四隻手指
缺點	較不耐用、感測器接線困難、感測器容易斷掉、抓取的訊號較不穩定。	價格較昂貴且已停產、多隻手指偵測會造成手套上感測器數量太多、系統複雜。	感測器質地較硬、較容易斷掉，且抓取訊號較不穩定。

2-4 Soft Labeling

製作智慧手套時，必須要做 Labeling，才可以讓神經網路學習手勢類別；在本研究中，標記的方式我們採用高斯核來進行標記，由於手勢較難判斷起點和終點，因此利用高斯核來找出接近手勢中心點的位置，然後再找出手勢的起點和終點。因此，在本研究中採用高斯核來做軟標籤(Soft Labeling)，並使用了密度型高斯核和累積分布高斯核兩種型態來做說明。

1. 型態(一) 密度型態高斯核：

   透過高斯核抓取接近手勢的中心點位置，並取手勢的中心點位置為最高分數，而起點和終點的分數較低，如圖 13 所示。所以即使手勢的起點或終點有標記錯誤，但因為起點和終點的分數較低，因此對 Labeling 的結果影響也相對較小；不過由於使用機率密度高斯核是取手勢的中心點位置為最高點，因此當神經網路在學習時或是推論的時候只會看到手勢的中心位置就會產生最高點進行判斷。



優點

手勢中心點的位置分數最高，而起點和終點的位置分數較低，所以即使手勢的起點和終點標記錯誤，對Labeling的結果影響也較小。

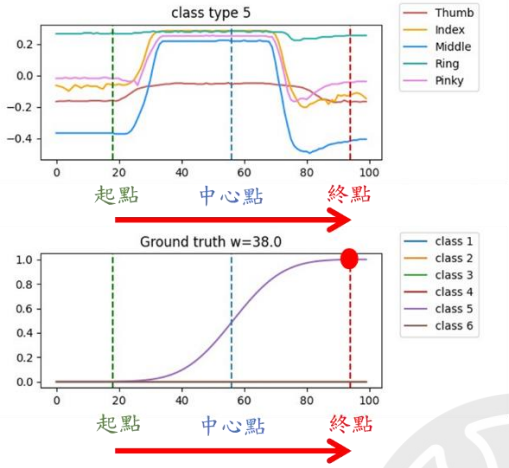
缺點

使用高斯核，神經網路在學習或推論時，只會看到手勢的中心點的位置就會產生出最高點進行判斷。

圖 13、密度型高斯核是以接近手勢中心點的位置為最高點，而起點和終點位置分數較低；但是當神經網路或是在推論時，只會看到手勢中心點的位置就會進行辨識。

2. 型態(二) 累積分布高斯核：

累積型態的高斯核一樣可以找出接近手勢中心點的位置，不過累積型高斯核是密度型高斯核的累加，如圖 14 所示；因此，在本論文中採第三個 Sample 點，所以手勢的終點接近 1，以看完整段手勢產生分數再進行辨識。



優點

- 一樣可以找出手勢中心點的位置。
- 是密度型高斯核的累加。
- 手勢終點位置分數接近1。
- 可看完整段手勢再產生分數。

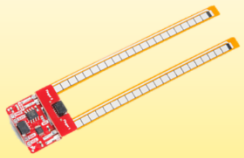
圖 14、利用累積型高斯核一樣可以找到手勢的中心點位置，但在這裡我們利用了累積型高斯核的特性，可以採取接近手勢終點的位置為分數最高點，以看完整段手勢在進行辨識。

第三章 研究方法

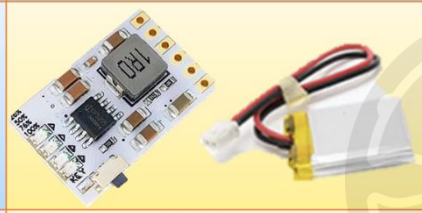
本章節將介紹智慧手套的設計和組成的要素；首先，第一小節介紹智慧手套所使用的各個元件，了解這些感測器的功能及整合方式，並用哪些元件來調整電池提供的電壓、充電等等問題。第二小節會介紹主控板電路架構的部分，說明架構應如何設計才能夠解決供應電壓不同的問題，第三小節會說明不同種類的彎曲感測器要如何修改感測器地址，在製作智慧手套上才可以避免地址衝突的問題。以及定義了哪些手勢來進行手勢辨識；最後，第四小節將會介紹本研究中應用於手勢辨識使用的模型架構。

3-1 手套元件介紹

在本研究中，我們使用了 Bend Lab 的 Digital Flex Sensor 和 Sparkfun 的 Qwiic Flex Glove Controller 兩種彎曲感測器來做為手指彎曲感測器的部分，用以實現五隻手指的智慧手套；為了提升整體效能，詳見圖 15 中的圖(a)，本研究中在 Arduino 板方面則選擇有整合藍芽功能及I^2C功能的 Bluno Beetle 板，使用這塊 Arduino 板不僅整合了藍芽和I^2C的功能，在硬體方面也節省了許多硬體空間。而 Bend Lab 的 Digital Flex Sensor 和 Sparkfun 的 Qwiic flex Glove Controller 兩種彎曲感測器分別配置於拇指和其他四隻手指上，降低了智慧手套系統的複雜度，也解決了拇指感測器的配置問題。此外，在充電方面，在圖 15 中的圖(b)，添加了鋰電池充電板、鋰電池和 USB Micro-B 充電口，讓智慧手套可以使用充電的方式來進行供電；這些元件的應用提升了系統的便利性，也有效降低了製作成本。

Arduino / Sensor	Arduino Bluno Beetle	Bend Lab的 Digital Flex Sensor	Sparkfun的 Qwiic Flex Glove Controller
			
功能	• 使用Bluno Beetle作為Arduino板。 • 本身具有藍芽，也支援I²C功能，可以減少硬體空間。	• 將Bend Lab的 Digital Flex Sensor 放置在拇指。	• Sparkfun的Qwiic Flex Glove Controller放置在其他四隻手指上。

(a) 本研究中使用 Bluno Beetle 作為 Arduino 主板，擁有藍芽及I²C功能。Digital Flex Sensor 裝於拇指，Qwiic Flex Glove Controller 裝在其他四指。

充電模組	鋰電池充電板、鋰電池	USB Micro – B充電口	降壓板
			
功能	• 使用鋰電池、鋰電池充電板，就不須像使用電池一樣要替換電池。	• 使用USB充電口，可以直接使用USB線即可充電。	• 由於彎曲感測器使用的電壓為3.3V，而電池提供的電壓為5V；因此需要降壓板降壓，否則電池提供的電壓對感測器來說電壓太大，會導致感測器燒壞。

(b) 充電模組包含鋰電池充電板、鋰電池和 USB Micro-B 充電口。避免定期更換電池，並透過降壓板調整電壓，有效避免感測器因電壓過高而損壞的風險。

圖 15、應用於智慧手套的元件說明，(a)為 Arduino 板及使用的彎曲感測器元件介紹、(b)為充電模組與降壓板應用說明。

3-2 電路架構

電路架構的部分則是參考論文[7]的電路架構，如圖 16 所示，由於本研究所使用的彎曲感測器電壓皆為 3.3V，而 Bluno Beetle 及鋰電池充電板供電的電壓均為 5V，如果直接將 5V 的電壓供應給彎曲感測器，會導致感測器有燒壞的風險。但同時也需要配合 Bluno Beetle 的電壓需求，否則在應用上會造成電壓不足無法正常使用的問題。因此，為了解決這個問題，則使用了降壓板；將 5V 電壓降低為 3.3V，以配合感測器使用的電壓範圍，確保感測器正常運作。在這裡也使用 USB Micro-B 充電口，使智慧手套的供電方式更加靈活；透過 USB Micro-B 充電口可以直接為手套充電，不需定期替換電池，也增加了手套使用的實用性和便利性。

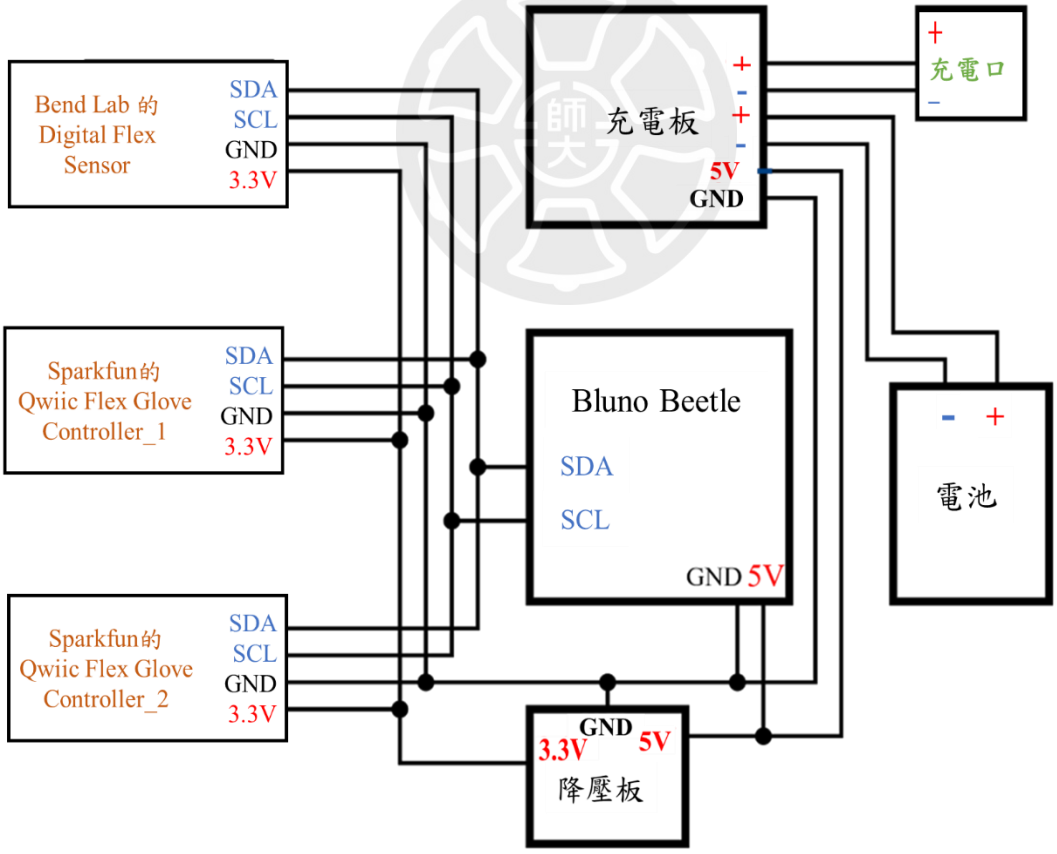


圖 16、電路架構圖中，我們使用了三個彎曲感測器來實現五隻手指的偵測，分別由一個 Bend Lab 的 Digital Flex Sensor 以及兩個 Sparkfun 的 Qwiic Flex Glove Controller 組成；其中，藍芽功能則整合在 Bluno Beetle 中，所以不須額外添加藍芽模組。

3-3 Arduino：感測器地址設定原因

本論文使用了兩種彎曲感測器，此兩種彎曲感測器分別來自不同廠牌，而且這兩種彎曲感測器的通訊協定都是採用I^2C協定，感測器之間共享相同的 Bus；所以感測器的預設地址可能會有相互衝突的問題。因此，將感測器配置在手套之前，需要先確保感測器是否有相同地址；若發現感測器有相同的地址，必須將感測器地址進行調整，確保每個感測器在不一樣的地址狀況下可以在同一系統中正常運作。

這兩種彎曲感測器修改地址的方式，是依據I^2C提供的地址修改方式進行設定，每個感測器都各自具有獨立的數位處理器架構，當數位處理器抓到感測器的訊號時；會將訊號存在地址暫存器中，所以每個暫存器都會有一個獨立的地址。因此，在修改地址的過程中，為了避免感測器之間互相干擾，我們必須採用逐一修改的方式來修改感測器地址，不可與其他感測器同時接線；否則會導致地址無法修改，因此，修改完之後才繼續修改下一個感測器。在圖 17 中以 Bend Lab 的 Digital Flex Sensor 為例，修改 Bend Lab 的 Digital Flex Sensor 感測器地址時，只能接一個彎曲感測器，因為需要與 Arduino 上的 CPU 做地址驗證，修改完一個彎曲感測器的地址之後才換下一個感測器進行地址修改。不可同時接多個彎曲感測器；若同時接多個感測器，會導致地址有無法修改的狀況。

而圖 18 為 Sparkfun 的 Qwiic Flex Glove Controller 地址修改順序，我們修改完 Bend Lab 的 Digital Flex Sensor 後，才去修改 Sparkfun 的 Qwiic Flex Glove Controller 感測器，此感測器修改地址方式是透過焊接的方式來進行修改，所以不必與 Arduino 上的 CPU 去做地址驗證。

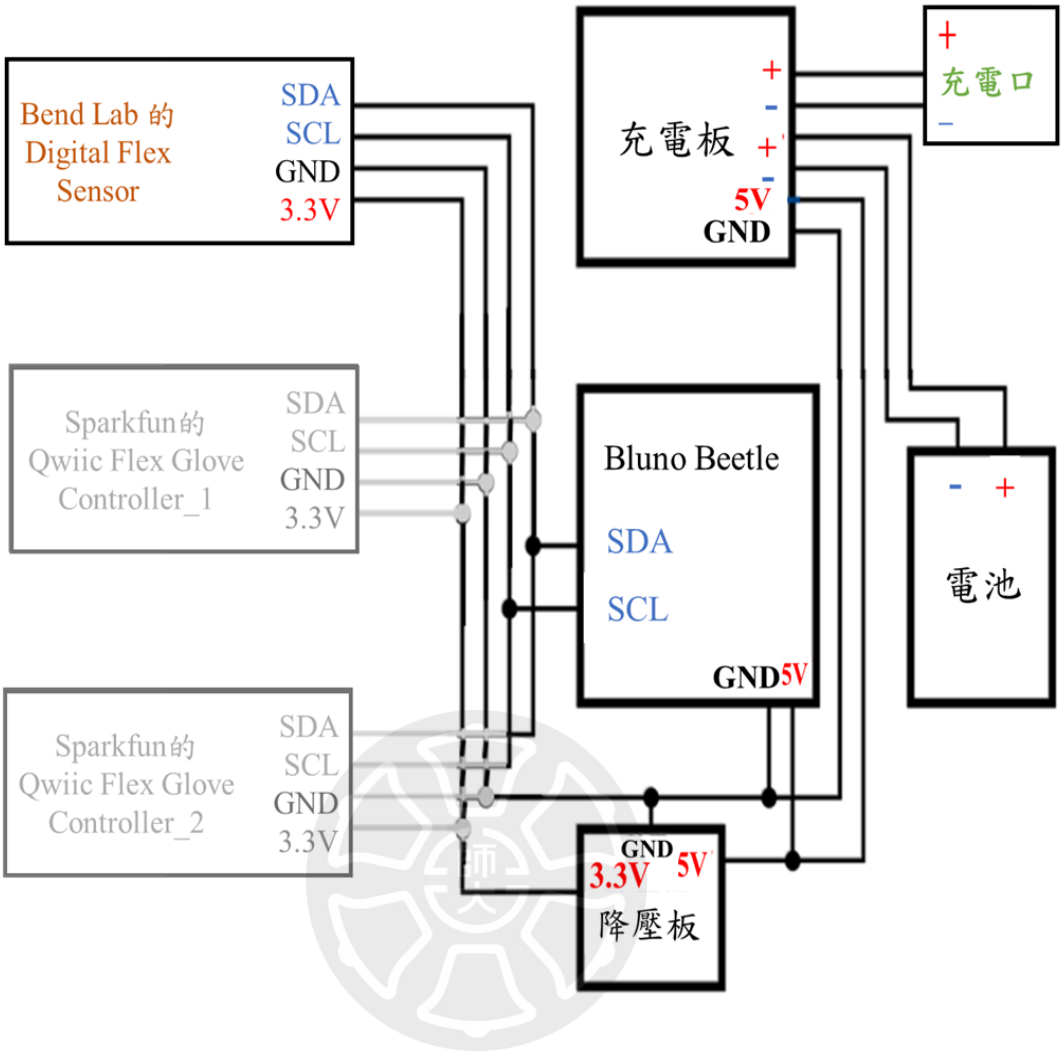


圖 17、Bend Lab 的 Digital Flex Sensor 感測器修改地址方法，由於感測器地址要逐一修改，如果同時接上多個彎曲感測器會導致地址無法修改成功，需要與 Arduino 上的 CPU 做地址驗證。

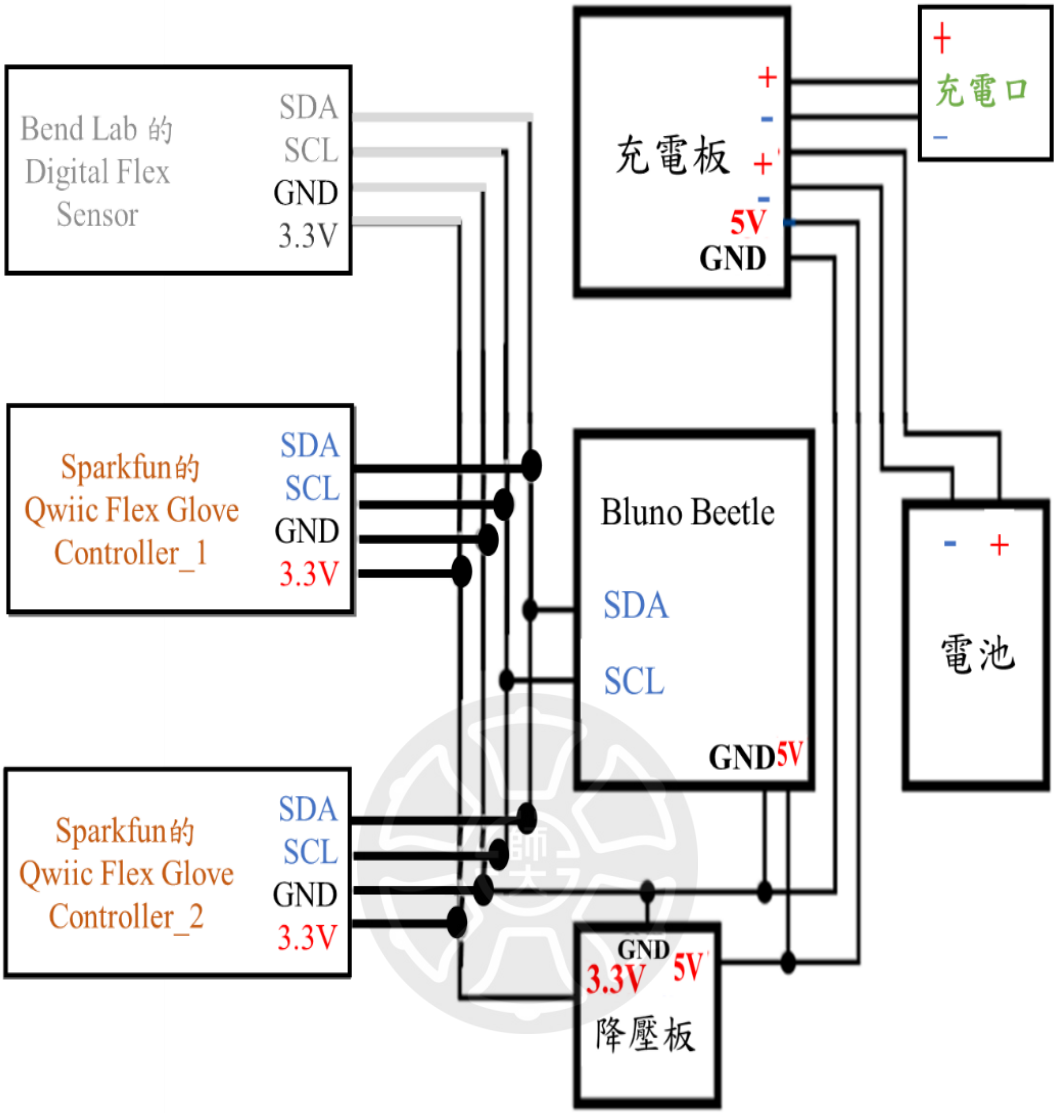


圖 18、當我們修改完 Bend Lab 的 Digital Flex Sensor 感測器地址時，才逐一修去修改 Sparkfun 的 Qwiic Flex Glove Controller_1 及 Qwiic Flex Glove Controller_2 的感測器地址，此 Qwiic Flex Glove Controller 感測器修改方式為焊接方式，因此不必與 Arduino 上的 CPU 去做地址驗證。

3-4 Arduino：感測器地址修改

手套製作使用了三個彎曲感測器，為了避免感測器之間有地址衝突問題，需要將這三個彎曲感測器的地址進行修改；以下為兩種不同的彎曲感測器修改方式：

1. Bend Lab 的 Digital Flex Sensor 地址修改：

將感測器要修改地址的程式碼上傳至感測器模組後，此部分地址修改方式可直接在 Arduino 的 Serial Monitor 執行地址設定，而 Bend Lab 的 Digital Flex Sensor 我們配置在拇指，如圖 19 所示拇指的位置；輸入十進制地址格式即可修改感測器地址。

i. 在圖 20 中可以從 Arduino 的 Serial Monitor 確定感測器目前的預設地址，而目前的預設地址為 0x28 / 40。

圖 19、這張圖顯示了 Bend Lab 的 Digital Flex Sensor，它被放置在拇指的位置，可以配合我們拇指長度的彎曲感測器。

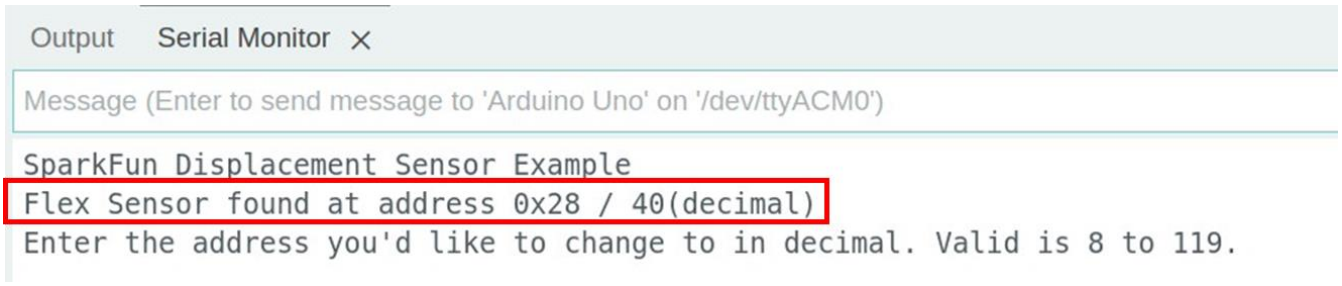


圖 20、Bend Lab 的 Digital Flex Sensor 設定地址方式在 Arduino 的 Serial Monitor 進行即可；先確定感測器的預設地址，目前此預設地址為 0x28 / 40(decimal)。

ii. 確定好感測器的預設地址後，可以在圖 21 中描述的 Arduino Serial Monitor 的 Message 處，輸入要修改的十進制地址格式，有效地址範圍為 8 ~ 119，例如：十進制地址 18。

iii. 地址設定完按 Enter 送出要設定的地址後，即可修改感測器地址，於圖 22 可以看到成功將感測器原本的十進制地址 40 更改為十進制地址 18。

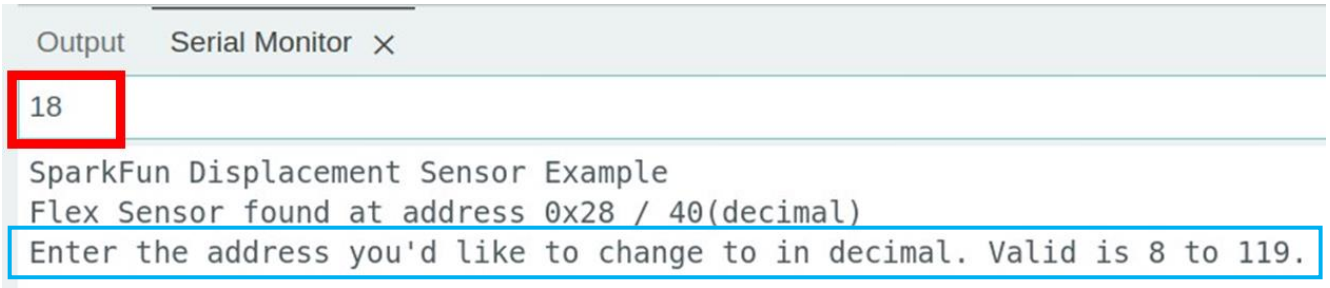


圖 21、輸入要修改的地址，這裡輸入修改的地址 18，地址有效範圍為 8~119。

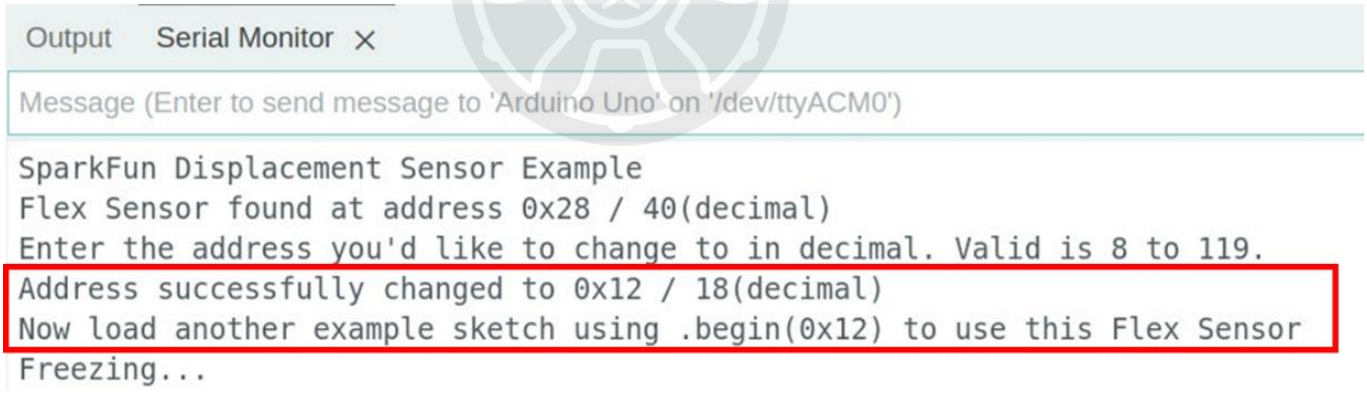


圖 22、設定完地址後，傳送指令修改 Digital Flex Sensor 地址，成功後顯示「Successfully」，將原本地址 40 已成功改為地址 18。

2. Sparkfun 的 Qwiic Flex Glove Controller 地址修改：

圖 23 顯示了 Sparkfun 的 Qwiic Flex Glove Controller 的配置位置，我們則應用於其他四隻手指；在圖 24 中可見，該感測器預設地址皆為 0x48。然而，除了預設地址 0x48 以外，這個感測器提供了其他三個可用的地址，分別有 0x49、0x4A、0x4B。

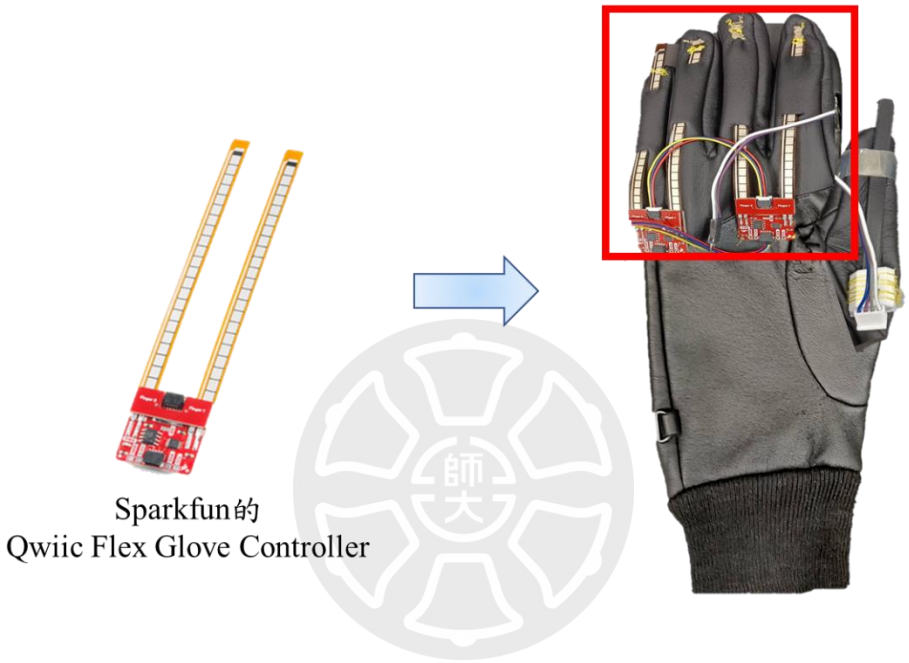
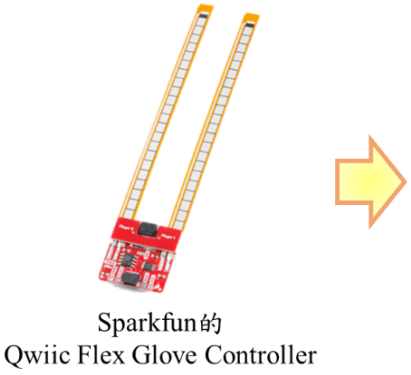


圖 23、Sparkfun 的 Qwiic Flex Glove Controller 一個主板上有兩隻彎曲感測器，則配置在智慧手套中其他四隻手指的位置。



Pin	Address
GND	0x48 (Default)
VCC	0x49
SDA	0x4A
SCL	0x4B

圖 24、Sparkfun 的 Qwiic Flex Glove Controller 可選擇其他的地址，除了預設地址之外，這款彎曲感測器還提供了三個可用地址，為 0x49、0x4A、0x4B。

i. 將小板背後原本焊接盤的連接處要先切割開，不先切割開連接處的話是感測器預設地址 (0x48)，會無法更改為其他地址，因此需要先切割連接處再焊接其他焊接盤來更改感測器地址，如下圖 25 所述的感測器地址修改處。

ii. 圖 26 中呈現了I^2C地址設定規格；由於I^2C 的 BUS 架構是以 1 Byte 傳輸，即 8 bits。在這 8 bits 中，最後 1 bit 是用來指示 Master 要讀取 Slave 的資料或是 Master 要寫入 Slave 的資料，因此 Slave_Address 只有 7 bits 可以用來指定地址[22]，其中，地址 0x48 的 Write_Address 為 0x90、Read_Address 為 0x91。

圖 25、Sparkfun 的 Qwiic Flex Glove Controller 地址修改處，這六個焊接盤的部分為修改感測器地址的地方。

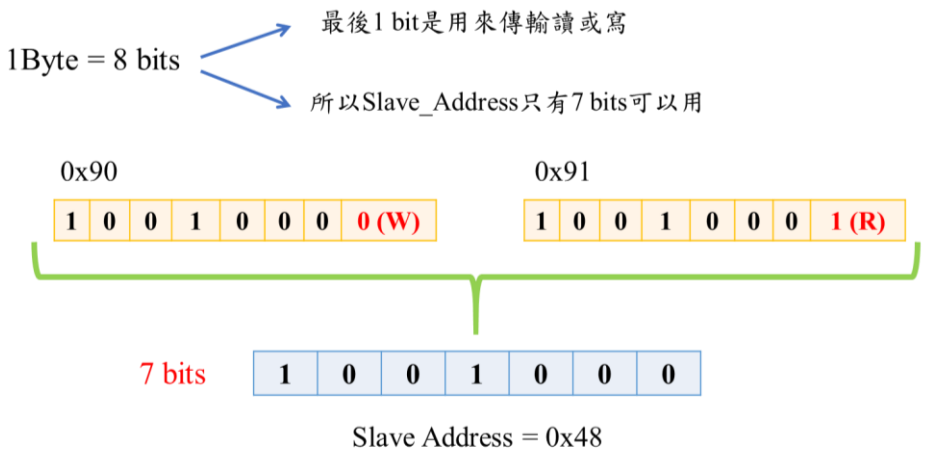


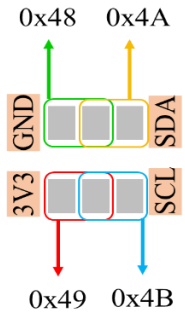
圖 26、Slave_Address 的地址設定；由於I^2C以 1Byte 來傳輸，而最後 1bit 是用來表示 Master 傳輸讀或寫，其中 Write_Address 為 0x90、Read_Address 為 0x91，所以 Slave_Address 只有 7 bits 可用來指定地址，地址為 0x48。

iii. Sparkfun 的 Qwiic Flex Glove Controller 感測器的地址修改處有設定跳線裝置(中間焊接盤的位置)，允許修改跳線狀態，改變I^2C地址[23]，於圖 27 可見I^2C地址格式以及每個地址的規格。

iv. 在本實驗中焊接的地方，詳見圖 28 中所示的 3.3V 與中間焊接盤的位置；焊接完成之後，透過跳線裝置更改了I^2C地址設定，使感測器地址變為 0x49。對應到I^2C的地址設定，於圖 29 所見，0x49 的 Slave_Address 就會變為 1001001(0x49)，其中對應的 Write_Address 為 0x92、Read_Address 為 0x93。

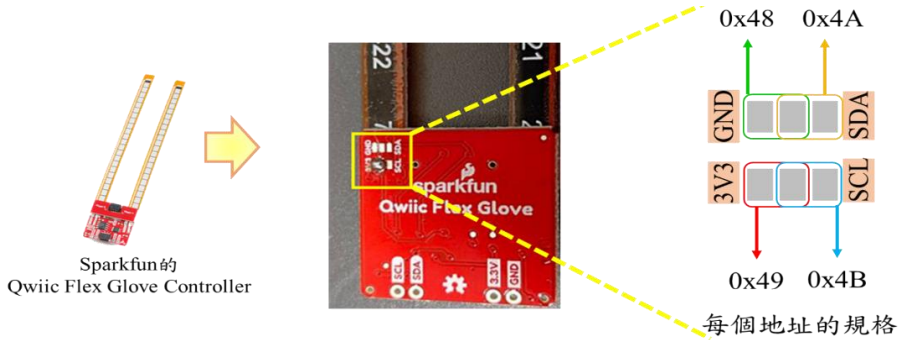
v. 圖 30 呈現了三個感測器的地址；當所有的感測器都修改完地址後，要偵測所有的彎曲感測器是否都完成地址修改；而在智慧手套中使用了三個彎曲感測器，所以同時偵測三個彎曲感測器會偵測到三個感測器地址，分別為 18、72、73。

ADDR PIN CONNECTION	SLAVE ADDRESS
GND	1001000
VCC	1001001
SDA	1001010
SCL	1001011



每個地址的規格

圖 27、Sparkfun 的 Qwiic flex Glove Controller 的 Slave_Address 設定，這六個焊接盤位置為感測器的跳線裝置，與中間的焊接盤焊接完成後，允許修改感測器地址的跳線狀態。



每個地址的規格

圖 28、本研究在感測器地址設定焊接處，焊接了 3.3V 與中間焊接盤的位置， 所以成功將感測器從預設地址 0x48 變更為 0x49 這個位置。

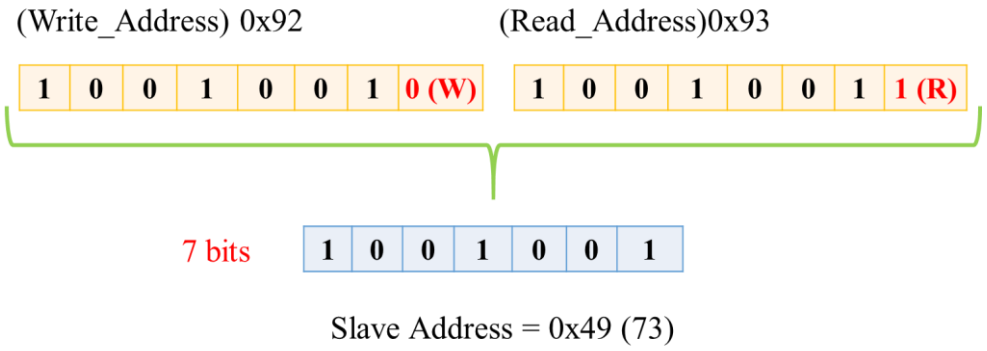


圖 29、由於地址更新為 0x49，對應到圖 26 中I²C的地址設定原理，其中 Write_Address 為 0x92、Read_Address 為 0x93，所以 7 bits 的 Slave_Address 為 0x49。

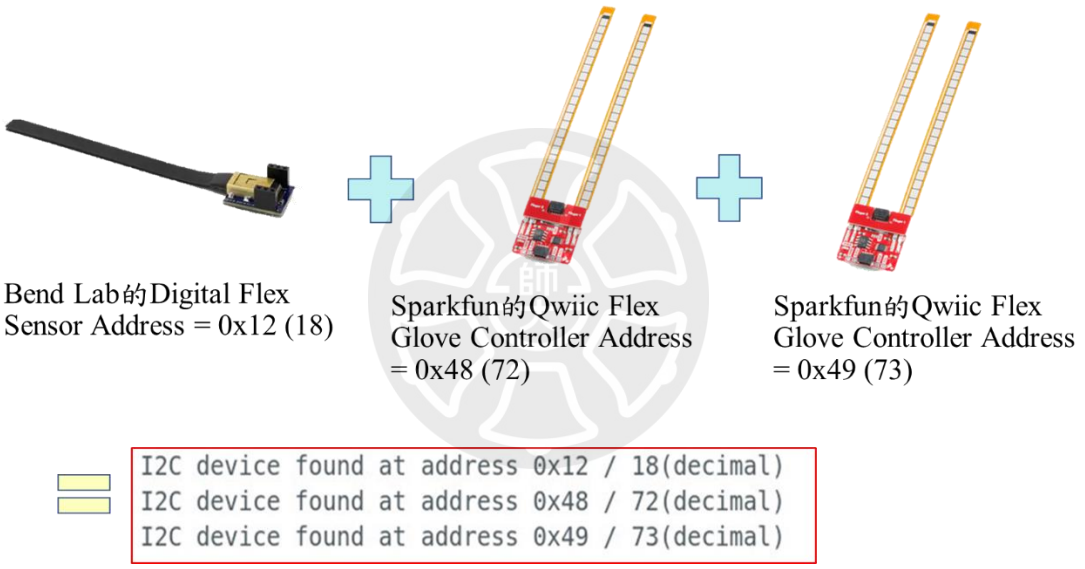


圖 30、設定完這三個彎曲感測器地址後，在檢查感測器地址偵測時，會偵測到所有的彎曲感測器地址，會顯示出三個地址，分別是 18、72、73。

3-5 智慧手套設計

當所有的彎曲感測器地址都設定好後，將彎曲感測器縫製到手套上固定，而其他 Arduino 等主控板則焊接在電路板上並配置在手背上固定，而且在主控板上添加一個綁帶固定在手腕，以便固定主控板位置。這樣配置和固定的方式確保在做手勢時主控板不會亂動，否則執行動作時可能會造成主控板晃動，造成感測器

接線有接觸不良，增加感測器接線可能會有接觸不良以及系統不穩定的風險，如圖 31。

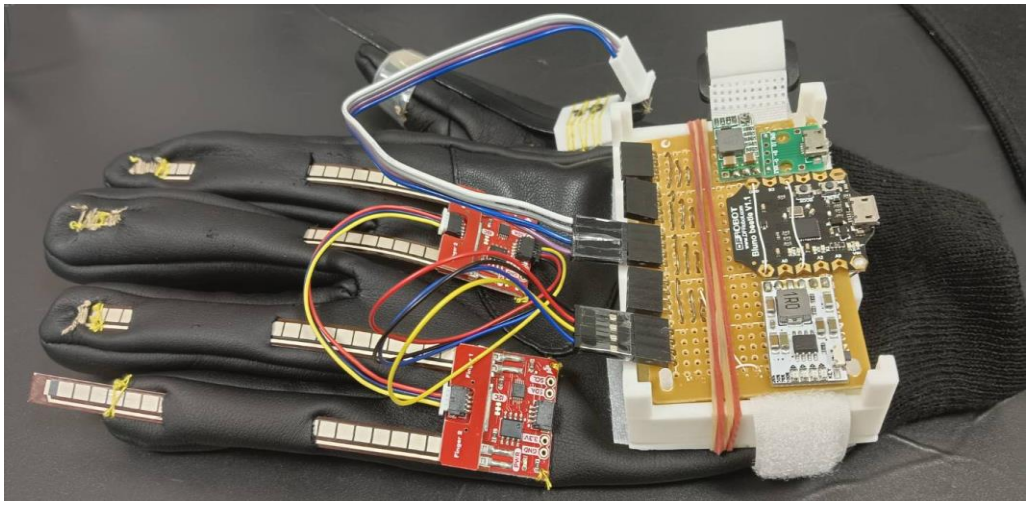


圖 31、呈現完成的智慧手套，依據圖 16 電路架構，將感測器縫製在手套上，實現五隻手指智慧手套，主控板則透過綁帶固定，避免在做手勢動作時，主控板鬆動造成感測器有接觸不良的問題。

3-6 手勢定義

在本研究中，我們定義了五種手勢，並利用彎曲感測器來蒐集這些手勢資料；定義的五個手勢中，每個手勢都由三階段組成，我們會採取三階段的手勢主要是為了確保使用者在做手勢動作時能夠輕鬆記得這些手勢動作；所以定義越簡單且自然的手勢動作為佳，讓使用者可以更容易記住這些手勢動作。手勢動作從起始動作開始，然後進行中間手勢，最後再回到起始動作以完成整個手勢；這些手勢也可用作於連續的手勢動作。

定義的這五種手勢基本上為三個階段，但也可以組合成多階段手勢，成為另一種新的手勢種類，以如圖 32 為例；Gesture 2、Gesture 3 和 Gesture 4 手勢可以組成更多不同的手勢；而 Gesture 1 和 Gesture 5 也可以組成另外一種連續動作的新手勢；因此這五個手勢動作也可以有多階段的連續手勢應用。為了讓模型可以訓練多樣性資料，我們分別請八位同學幫忙蒐集定義的五種手勢資料。

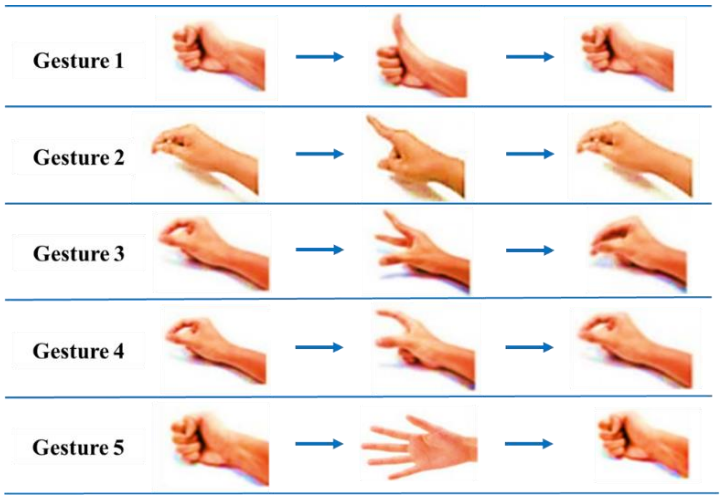


圖 32、本研究定義五個手勢，每個手勢都是三個動作組成，可以組合成多種不同的手勢，但本研究利用定義的這五個手勢來觀察我們五隻手指偵測的辨識效果。

3-7 模型架構

本論文在手勢辨識中，使用 Sliding_Window 只是用來驗證本實驗法則的有效性，本實驗中的手勢長度大部分都接近 100，且手勢的起點、中心點及終點位置我們假設皆為已知，所以不太需要考慮 Sliding_Window 的部分；而真正需要應用 Sliding_Window 的部分則由後續的實驗繼續進行。以圖 33 為例，由於手勢長度各不相同，所以我們使用一個固定長度的 Sliding_Window 來固定手勢的長度；此 Sliding_Window 所涵蓋的範圍則會當作類神經網路的輸入。由於採樣頻率為每秒 50 個 Sample 點，我們設定每個 Sliding_Window 大小為 100，也就是 100 個 Sample 點，與神經網路的輸入長度相符，並且再由神經網路產生輸出分數。

參照圖 34，透過利用 Sliding_Window 的方式來滑過手勢；因此本論文將 Window_Size 的大小設為 100 來固定手勢資料的大小，讓類神經網路可以針對 Window 固定長度的範圍產生分數，而且設定 Window 之間間隔為 20 個 Sample 點，代表 Window 與 Window 之間的間隔有 20 個 Sample 點；再滑動至下一個

Sliding_Window 並輸出每個 Sliding_Window 的分數，所以每兩秒會有一次手勢辨識。

由於手勢資料是一維資料，因此神經網路是使用一維卷積神經網路來實現，在神經網路中，使用了五層的 Convolution (卷積層)，最後再透過 Softmax 做手勢分類，分類出前景手勢和背景手勢；因此最後輸出會得到一個 Window_Size 為 100，共有 6 (五種前景手勢 + 背景手勢)種手勢，為[100][6]大小的輸出；神經網路架構如圖 35 所示。

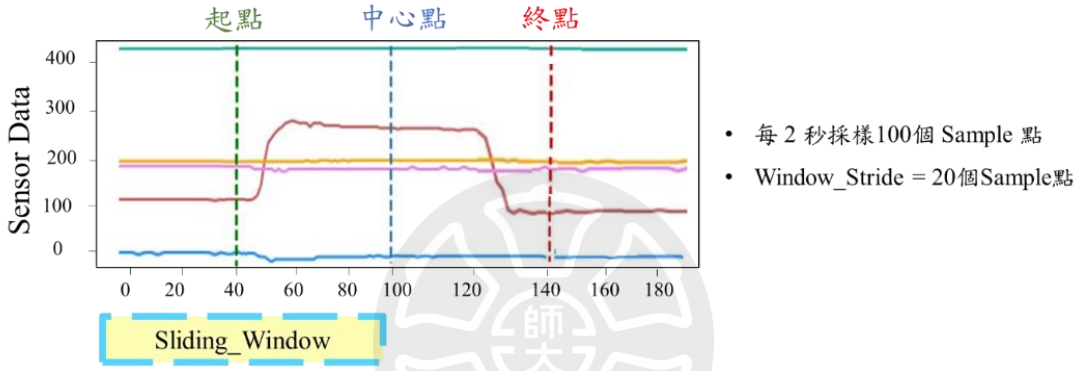


圖 33、手勢的固定 Window 說明，每秒採樣 50 個 Sample 點，本論文是採取 100 個 Sample 點，而 Window_Stride 設為 20 個 Sample 點，所以每兩秒會有一次手勢辨識。

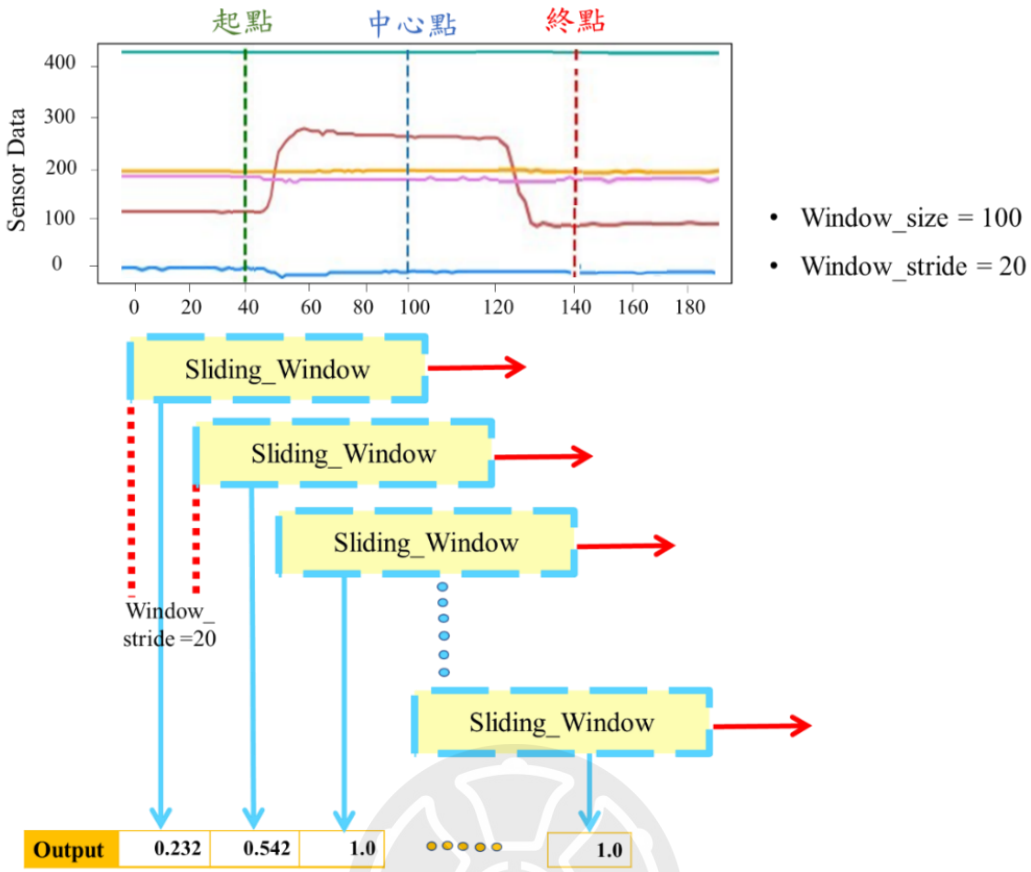


圖 34、利用 Sliding_Window 滑過手勢，讓神經網路可以針對固定的手勢長度各別輸出每個 Sliding_Window 的分數。

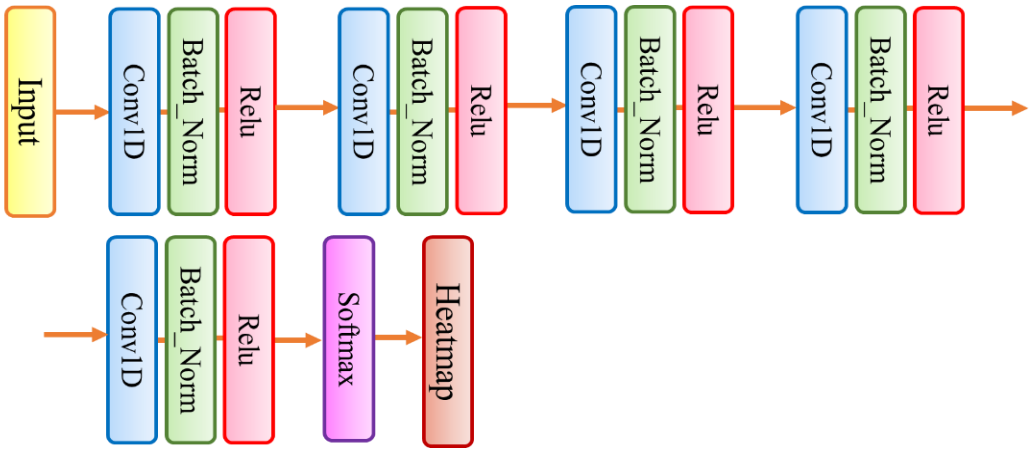


圖 35、模型架構，手勢為一維資料，因此使用一維的神經網路來實現，透過五層卷積層，最後再經過 Softmax 來進行手勢分類。

第四章 實驗數據與效能分析

本章節的第一節會先介紹本研究的實驗環境，第二節的部分會詳細說明手勢資料是如何收集，包括確認手套 Arduino 板上的藍芽是否與 PC 端藍芽是如何進行連線。同時，說明手勢資料的格式，以呈現手指數據對應的順序。而第三節中，則闡述如何使用累積型高斯核來繪製手勢的 Ground Truth；最後，在第四節中，將分析密度型高斯核和累積型高斯核在使用後所達到的準確率。

4-1 實驗環境

在實驗環境方面，選擇了 Ubuntu 20.04 版本的作業系統；而程式語言使用 python3.6 版本，此版本有支援本研究需要使用的機器學習套件版本。機器學習方面，使用的套件是 TensorFlow 2.3.0 來進行本研究，CPU 為 i7–7700，GPU 則是 RTX 2070；另外，由於有大量的手勢數據要進行分析，因此要確保實驗的效果和效率，因此記憶體至少需要 32GB。

4-2 三種類型智慧手套差異性

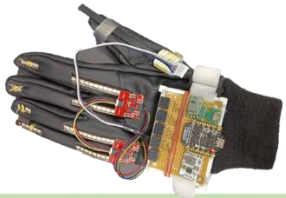
在論文[6]、[7]及本論文的這三種智慧手套中，分別都使用了不一樣的彎曲感測器來進行手勢辨識，摘錄於表格 2，本論文比較了這三種智慧手套的差異性；首先，在論文[6]的智慧手套，使用了 Sparkfun 的 2.2 inch Flex Sensor 來製作智慧手套，但是使用 Sparkfun 的 2.2 inch Flex Sensor 彎曲感測器時，需要使用導電線縫製而成；由於導電線要絕緣處理，因此使用導電線縫製還需要減少跨線，否則手套整個系統會短路。

而在論文[7]的智慧手套中使用 Bend Lab 的 Digital Flex Sensor，不僅提高感測的準確度，而且在使用上感測器耐用度較高，也不需使用類比被動式彎曲感測器透過導電線縫製等等方式來製作。論文[7]所設計的智慧手套降低了許多硬體複雜度的部分；但由於上述的兩種彎曲感測器都只使用三隻手指來偵測，因此容易造成有相似的動作使得手勢在辨識時有混淆的狀況。

本研究所製作的智慧手套中，我們進一步降低了手套的系統複雜性，而且在感測器方面也避免使用五個彎曲感測器來製作五隻手指的智慧手套；透過精簡的方式採用了 Bend Lab 的 Digital Flex Sensor 和 Sparkfun 的 Qwiic Flex Glove Controller 兩種軟硬兼具的彎曲感測器製作了五隻手指的智慧手套；不僅提升了手勢辨識的效果，還解決了五隻手指在感測器上使用的問題。

與一般被動式的彎曲感測器相比，本研究所使用的主動式彎曲感測器還支援了I^2C通訊協定，減少了感測器之間的接腳使用；降低手套製作及系統的困難度。

表格 2、三種類型智慧手套差異性，與論文[6]、[7]兩篇所提出的智慧手套不同的地方在於這兩篇所製作的智慧手套都只使用三隻手指來進行辨識，而本研究所製作的智慧手套為五隻手指來進行辨識，且有效提升手勢辨識率。

	 論文[6]的智慧手套	 論文[7]的智慧手套	 本論文設計的智慧手套
Sensor	Sparkfun 的 2.2 inch Flex Sensor 	Bend Lab 的 Digital Flex Sensor 	Bend Lab 的 Digital Flex Sensor + Sparkfun 的 Qwiic Flex Glove Controller  
Sensor數目	3	3	5
辨識效果	良好	良好	良好
差異性	製作時要減少跨線，導電線要絕緣處理，否則會手套系統會短路。	只使用三隻手指，有些動作太相似會造混淆，但五個感測器數量又過多。	利用三個感測器完成五隻手指隻手套，降低手套製作複雜度。

4-3 收集手勢資料方式

4-3-1 手套啟動連線

在開始進行手勢偵測之前，首先；需要先測試每個彎曲感測器的地址是否都能被偵測到，為了確保感測器接線是否正確、是否有正常運作，例如：在檢查感測器地址時，假設只有偵測到一個或兩個地址，則表示感測器在接線或是程式運作上有出了錯誤。當所有的感測器地址都有被偵測到才可以透過藍芽連線將資料傳送到電腦，偵測時會進行以下步驟：

1. 手套啟動：

   將手套電源開啟後，先確定每個使用的彎曲感測器是否都有偵測到地址，由於是使用三個彎曲感測器，因此在偵測感測器地址時，在做所有感測器偵測時，在圖 36 中顯示了三個感測器地址。

2. 藍芽連線：

   在 PC 端接上藍芽接收器後，等待感測器模組上的藍芽燈亮起；詳見圖 37，當 Bluno Beetle 上的藍芽燈亮起，表示 Bluno Beetle 上的藍芽和 PC 端上的藍芽已連線。

3. 電腦畫面顯示：

   圖 38 說明了藍芽連線成功之後，執行收集手勢的程式碼，即可開始收集手勢數據，並存取為 txt 檔案。

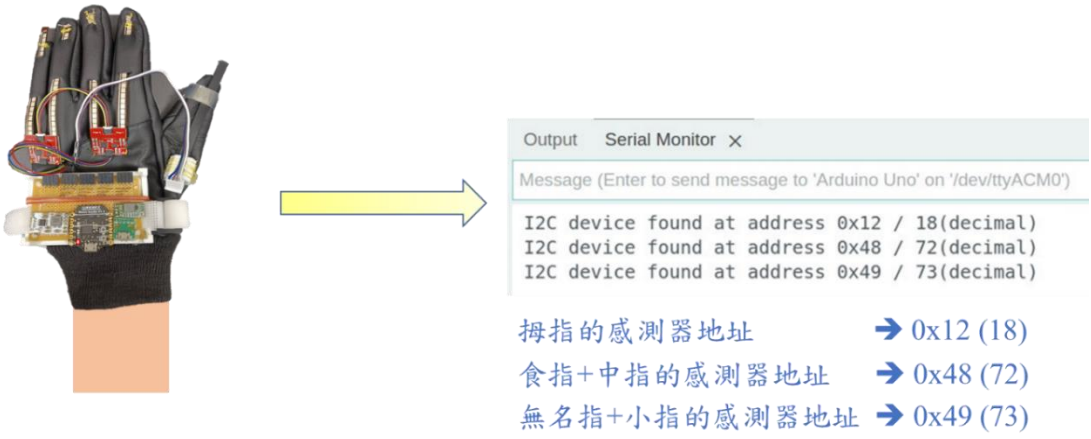


圖 36、手套電源啟動後，先檢查感測器地址是否都有偵測到，有偵測到三個地址表示三個彎曲感測器皆有正常運作。

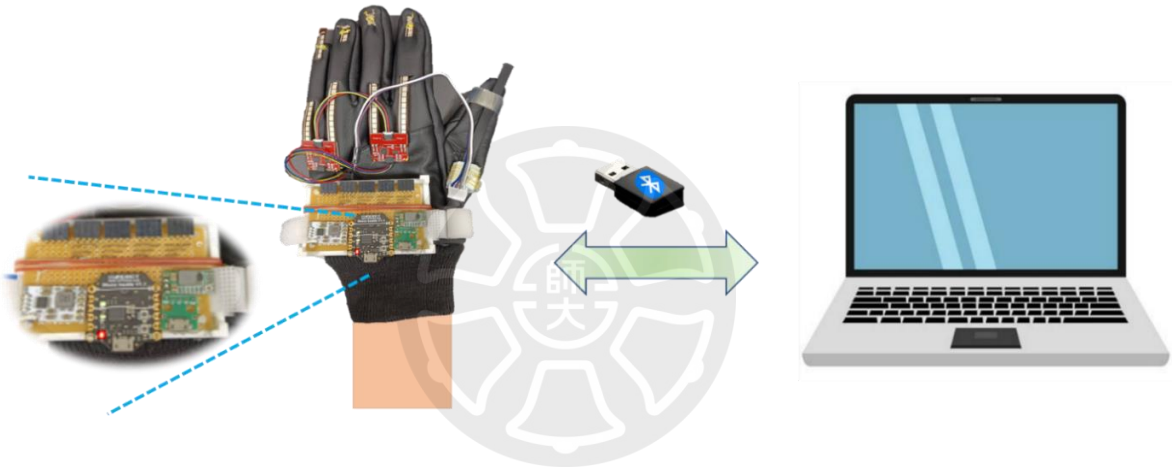


圖 37、藍芽連線部分，由於 Bluno Beetle 上已經有整合藍芽，所以只須將藍芽接收器插在 PC 端上，等待模組上的藍芽燈亮起，表示連線完成。

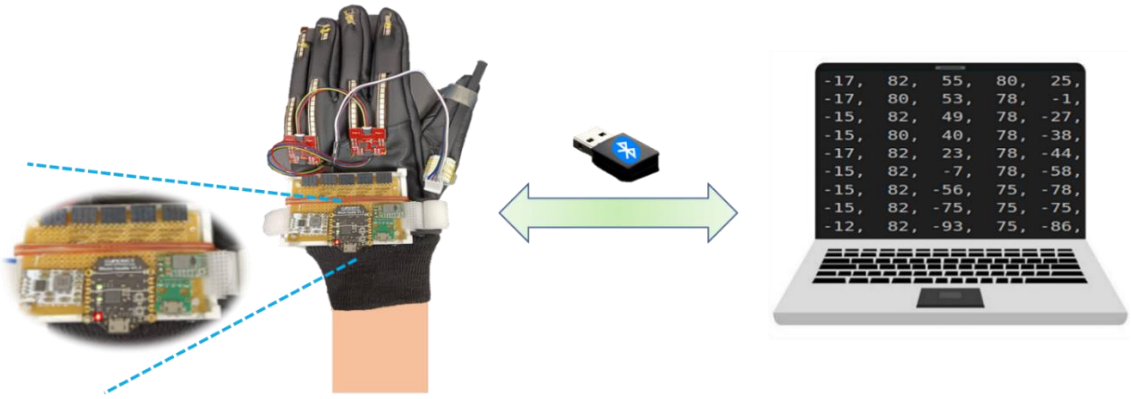


圖 38、藍芽連接後，感測器就可將感測的數據傳輸到 PC 端上，此接收數據部分可在 PC 端上看到。

4-3-2 手套藍芽連線

在進行收集手勢前，必須先確定模組上的藍芽是否與電腦的藍芽有正常連線；在圖 39 描述的部分，將藍芽的程式碼上傳之後，Bluno Beetle 模組上的 TX 會開始閃爍，此時執行 Serial Monitor 會出現亂碼，表示可以傳輸彎曲感測器感測的資料。接著，將藍芽接收器插在電腦上，等待模組的 LIHK(藍芽燈)亮起，表示藍芽連線成功。

在收集完手勢資料，每一筆手勢資料會個別存為 txt 檔，圖 40 說明了手勢資料格式的排序，我們將資料的格式設置與手指相同的順序，拇指、食指、中指、無名指、小指。

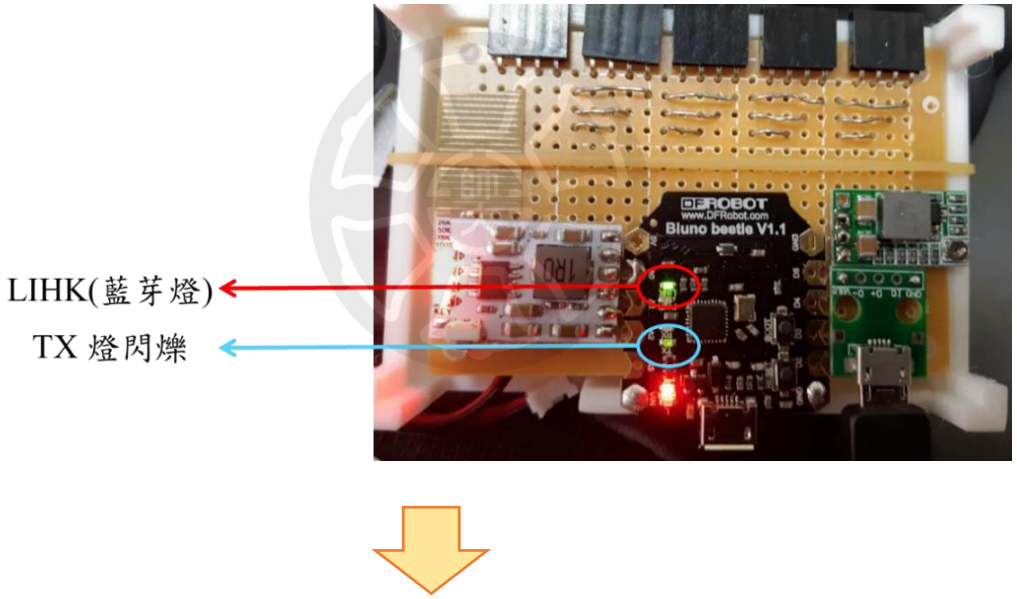


圖 39、當藍芽的程式碼上傳至模組後，模組上的 TX 燈開始閃爍，表示可以傳輸感測器資料，在 Arduino Serial Monitor 上面出現亂碼，表示感測器可以傳送資料。由於是以 Byte 格式顯示，所以會呈現亂碼的樣子；這時只需將藍芽接收器插上 PC 端即可。

Thumb	Index	Middle	Ring	Pinky
30,	-23,	-93,	82,	-51,
33,	-25,	-93,	80,	-49,
37,	-25,	-93,	80,	-49,
35,	-25,	-93,	80,	-49,
36,	-27,	-95,	80,	-49,
35,	-27,	-95,	80,	-49,
29,	-27,	-95,	82,	-49,
28,	-27,	-95,	80,	-49,
32,	-27,	-95,	80,	-47,
37,	-27,	-95,	82,	-49,
36,	-27,	-93,	80,	-49,
34,	-27,	-93,	80,	-49,
32,	-27,	-93,	82,	-47,
32,	-27,	-93,	80,	-49,
29,	-25,	-93,		
31,	-27,	-93,		

圖 40、手勢資料定義的格式，此排序由手指的順序來進行排列，所以為拇指、食指、中指、無名指及小指。

4-3-3 手勢資料收集

為了增進手勢辨識的多樣性及準確性，本論文定義了五種手勢來進行辨識；收集多位同學的手勢資料，這樣可以蒐集到每個人不同的手勢訊號，讓神經網路可以學到多樣資料。因此訓練集和測試集分別蒐集了八位同學的手勢資料，每位同學各收 75 筆手勢資料；訓練集共 2100 筆、測試集共 1500 筆，如表格 3 所示，確保有足夠的資料量用於模型訓練，且有充足的資料進行測試及評估。

表格 3、訓練集和測試集手勢筆數，class 1 ~ class 5 為五個定義的前景手勢，class 6 為背景手勢。

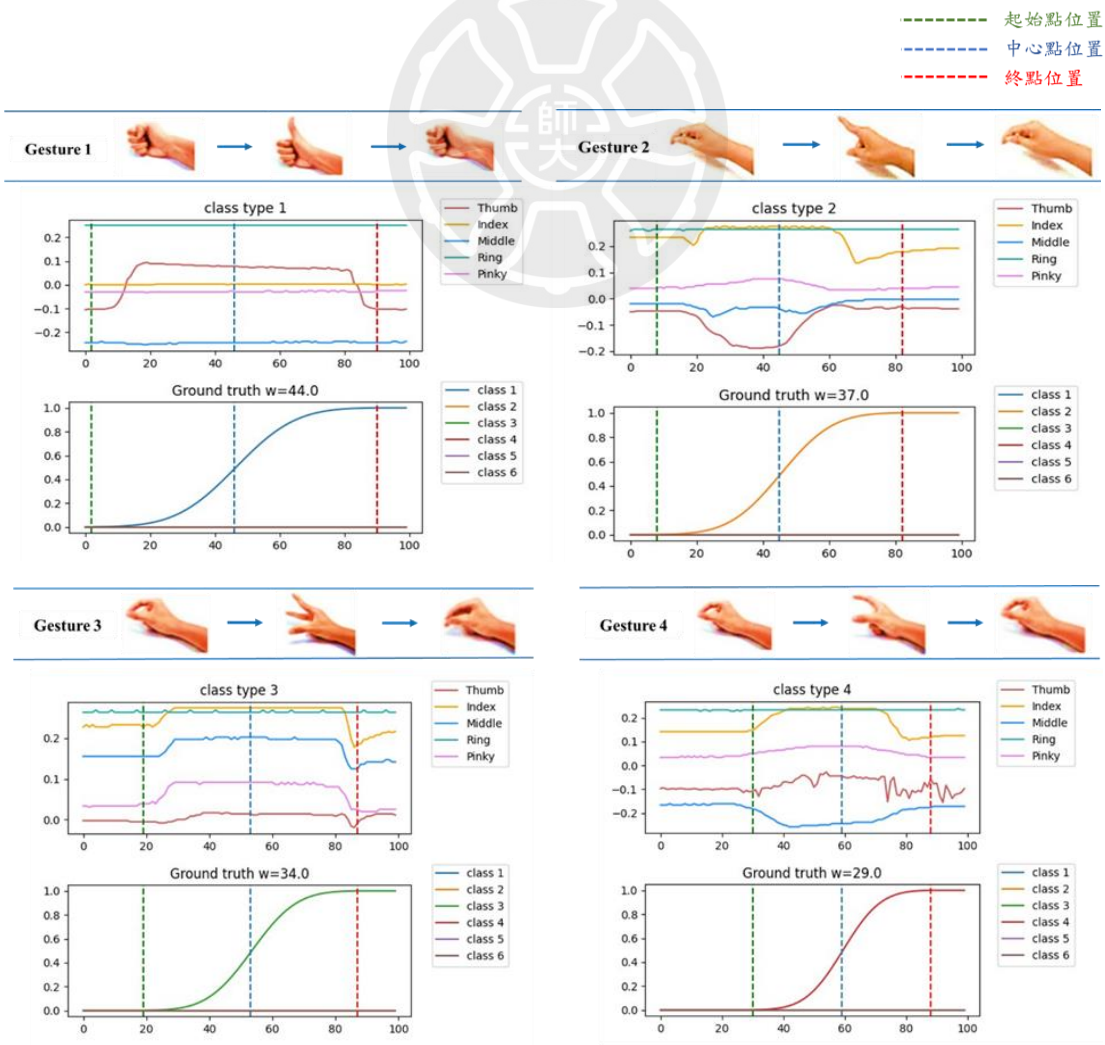
訓練資料：2100 筆

測試資料：1500 筆

Gesture class	class 1	class 2	class 3	class 4	class 5	class 6 (背景手勢)
訓練集	350	350	350	350	350	350
測試集	250	250	250	250	250	250

4-4 Gesture Ground_Truth

在 Ground Truth 方面，本實驗中我們假設這些手勢的起點、中心點及終點位置皆為已知，且手勢的長度都接近 100；並取接近手勢終點的位置來觀察整段手勢。圖 41 中顯示了每個手勢的 Ground Truth，而本論文使用了累積型高斯核畫 Ground Truth，這裡會使用累積型高斯核的原因主要是因為在本論文中的神經網路分類是使用 Softmax 做手勢分類；由於高斯是連續分布，但如果使用密度型高斯核做積分的話不會為 1，且會大於 1；所以這裡使用累積型高斯核來畫 Ground Truth，累積型高斯核是密度型高斯核的積分，且累積型高斯核的積分為 1；因此在本論文中我們採用累積型高斯核來繪製 Ground Truth，並取第三個 Sample 點最高的分數，以看完整段手勢再來進行辨識手勢。



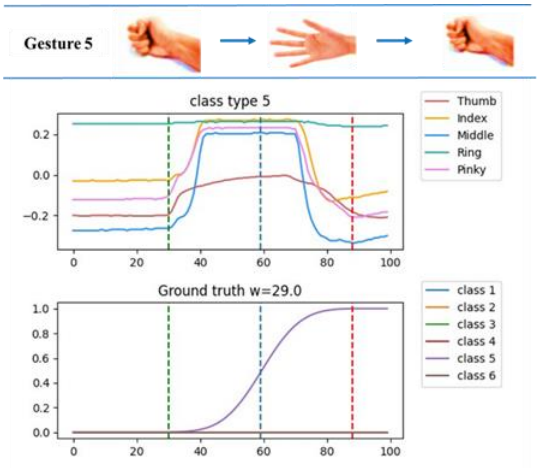


圖 41、五個手勢利用累積型高斯核畫出的 Ground Truth，透過累積型高斯核，我們皆取接近手勢終點的位置為最高點。

在圖 42 中展示了密度型高斯核和累積型高斯核辨識手勢最高點的位置；在密度型高斯核中，由於最高點為手勢中心點的位置，因此神經網路在學習或是推論只會取到中間第二個 Sample 點，也就是手勢的一半就會進行辨識；而累積型高斯核，可以採取最高點為手勢終點的位置，表示取第三個 Sample 點，以看完整段手勢後再進行辨識。

由於在收集手勢時，每個人的伸展、彎曲程度不同，因此在做動作時，手套也會產生摩擦、或是有些同學在做某些手勢動作時會有手抖等問題，如圖 43 所示，而感測器也都偵測到這些手勢訊號；但因為使用了五隻手指來進行偵測的關係，進而提高了手勢辨識的效果，因此即使有手套摩擦的訊號等手勢，一樣可以辨識出該手勢為哪一類手勢。

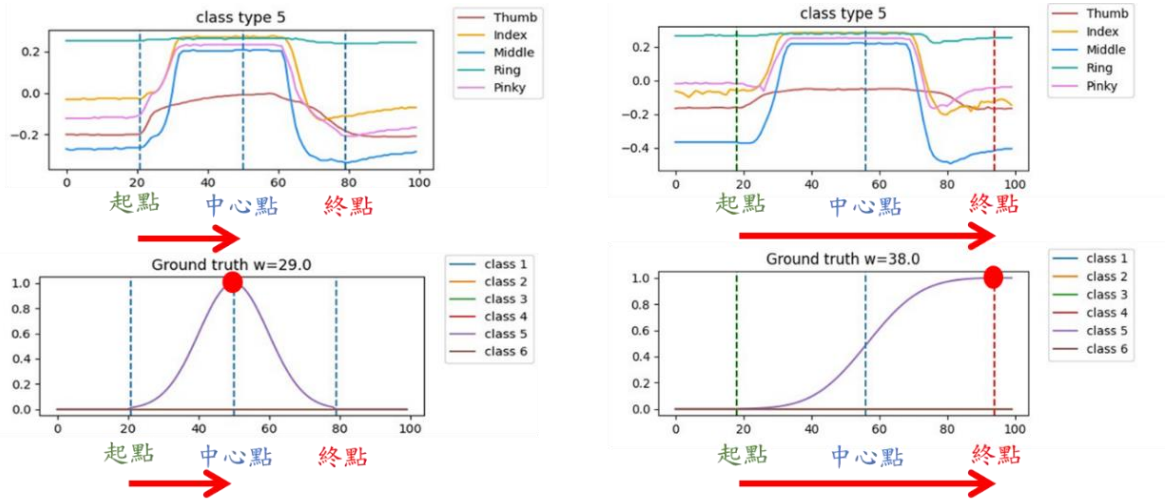


圖 42、密度型高斯核是取接近手勢中心點的位置，也就是第二個 Sample 點；而累積型高斯核則是取接近手勢終點的位置，也就是第三個 Sample 點。

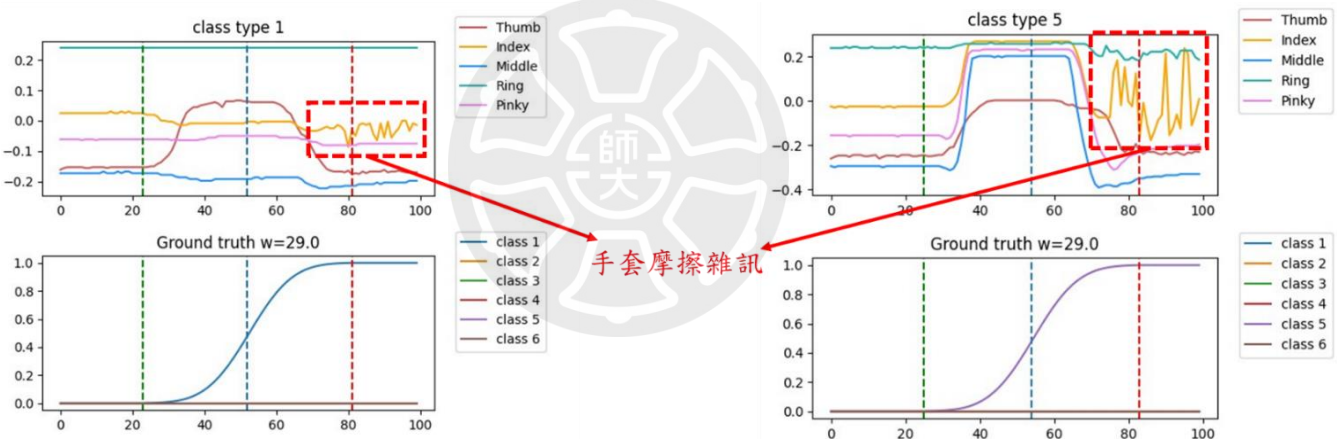


圖 43、使用智慧手套在收集資料時，在做手勢動作會使手套產生摩擦的訊號，紅色框線部分上下晃動較為劇烈的部分為手套摩擦的雜訊。

4-5 Confusion Matrix

在本論文中，採用密度型高斯核和累積型高斯核進行手勢辨識效果比較，並透過 Confusion Matrix 畫出了這兩種方法對手勢辨識的效果，分別是考慮只看到手勢中心點的位置的手勢區段以及看完整段手勢區段的部分；型態一，透過密度型高斯核，只看到手勢中心點的位置。型態二，使用累積型高斯核，看完整段手

勢特徵兩個方法來比較手勢辨識的效果：

1. 圖 44 為密度型高斯核的 Confusion Matrix；可在表格 4 中看到密度型高斯核準確率為 0.91 左右。
2. 圖 45 為累積型高斯核的 Confusion Matrix，而在表格 5 中，累積型高斯核的準確率高達 0.96 左右。

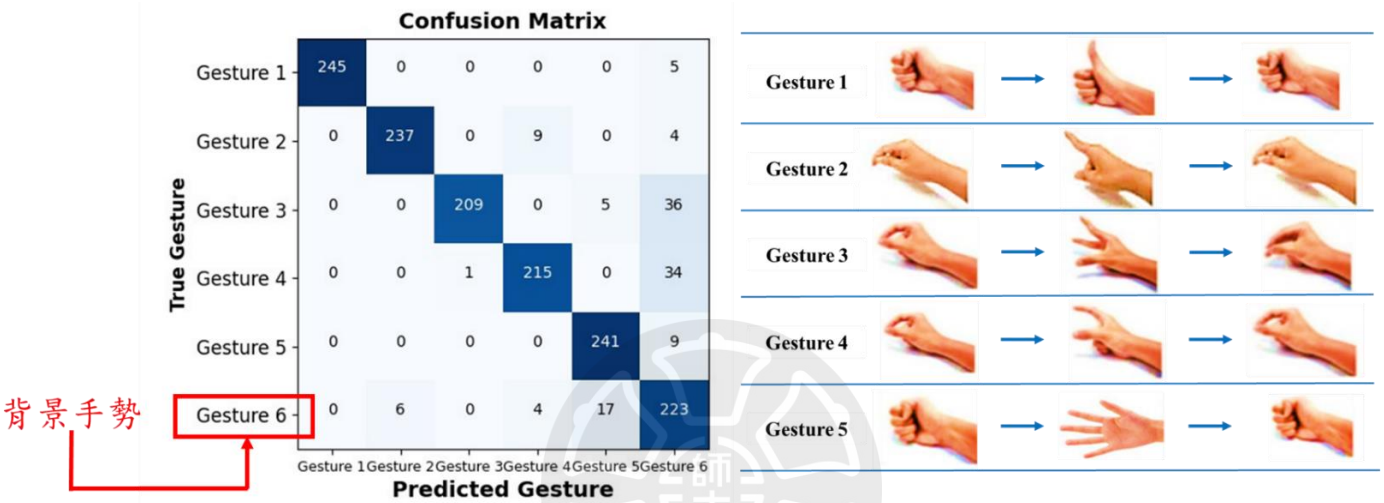


圖 44、密度型高斯核 Confusion Matrix，以取接近手勢的中心點位置來進行辨識。

表格 4、透過密度型高斯核只看到接近手勢的中心點位置，可以發現有些手勢可能有誤判的狀況，所以手勢的平均準確率為 0.91 左右。

Class Gesture	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6 (背景手勢)
每個手勢平均	0.98	0.948	0.836	0.84	0.964	0.892
Hit Rate	0.913					

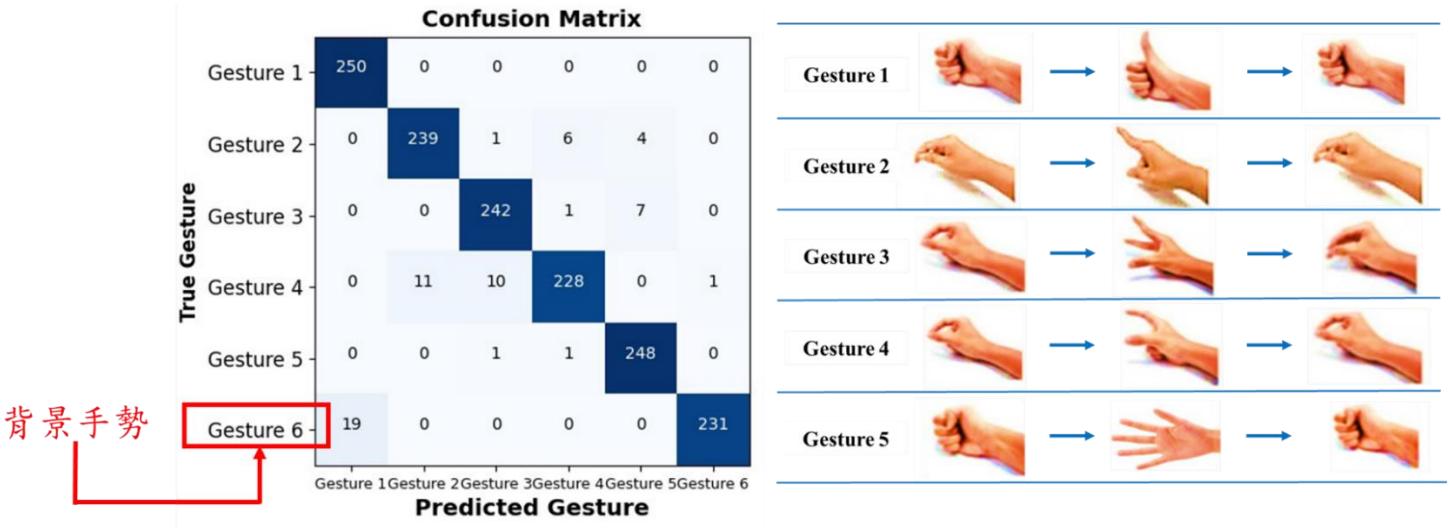


圖 45、累積型高斯核 Confusion Matrix，則是以取接近手勢的終點位置來進行辨識，以分析完整段手勢。

表格 5、利用累積型高斯核取接近手勢的終點位置，以看完整段手勢在進行辨識；而看完整段手勢辨識效果有明顯提升，手勢的平均準確率約 0.96 左右。

Class Gesture	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6 (背景手勢)
每個手勢平均	1.0	0.956	0.968	0.912	0.992	0.924
Hit Rate	0.958					

這兩種方法相比，由於密度型高斯核是以手勢的中心點位置為最高點，因此當手勢進行到一半時，就會產生手勢的最高點；無法完整看完整段手勢再進行辨識。因此如果有較相似的手勢，可能容易發生誤判的狀況。而本論文使用的方法是累積型高斯核，並採取終點的位置為手勢的最高點再進行辨識，以分析整段手勢；所以從實驗結果可以發現，使用累積型高斯核看完整段手勢再進行辨識的方法，比密度型高斯核僅看到手勢一半的辨識效果好。

第五章 結論

本論文整合了軟式的 Bend Lab Digital Flex Sensor 以及硬式的 Sparkfun Qwiic Flex Glove Controller 兩種彎曲感測器，利用三個彎曲感測器實現五隻手指的智慧手套設計；在本實驗中，我們使用的這兩款彎曲感測器均有支援I^2C架構，利用I^2C的連接方式，方便感測器之間可以互相進行通訊；而且I^2C同時提供了較容易修改彎曲感測器地址的方式，使感測器的地址設定更加靈活。此外，本研究採用藍芽來完成無線傳輸的部分，將感測器所感測的數值透過藍芽來進行資料傳輸，表示我們可以使用感測器的訊號來錄製這些訊號的波型。

透過藍芽傳輸感測器的資料後，我們使用一維卷積神經網路模型來做檢測，在假設已知手勢的起點、中心點及終點位置的條件下，分析手勢分類的準確率；因此本論文使用兩種型態的高斯核，分別利用密度型高斯核以及累積型高斯核來比較手勢辨識的效果。最後透過實驗結果可以發現，密度型高斯核只取到手勢的中心點位置準確率約 0.91 左右，而累積型高斯核取到手勢的終點位置準確率約 0.96；所以可以發現，在手勢辨識中使用累積型高斯核取最後一個 Sample 點，即觀察完整段手勢後進行辨識，相對於密度型高斯核僅取中心點位置的方法，具有更好的辨識效果。

參考文獻

[1] C. Y. Cheng, " VR and Gesture Recognition based Interactive Instrument Showroom, " National Ilan University, Jul, 2020, doi:10.6820/niu202000207.

[2] H. I. Fang, " The Study of Augmented Reality Operation Gesture Preference, " Tatung University, Jul , 2018.

[3] J. M. LIU, " Using Raspberry Pi to Realize Gesture Recognition Based on Image and Gesture Control Panel, " Chaoyang University of Technology, Jun, 2018.

[4] T. F. Chien et al. " A Remote Hand Rehabilitation System Enhanced with Virtual Reality and Games," 南台學報 38(3), pp. 33-44. Sep 2013.

[5] C. C. HSIAO, "A Design and Implementation of A Home Rehabilitation Wearable System, " National Yunlin University of Science and Technology, Jul, 2017, doi: https://doi.org/10.1145/2753509.2753521.

[6] W. C. Chuang, " Continous Finger Gesture Recognition Based on Flex Sensors, " National Taiwan Normal University, Jun, 2019, doi:10.6345/NTNU201900309.

[7] H. K. Chang , " Real-time gesture recognition system based on CenterNet algorithm and digital flex sensor, " National Taiwan Normal University, Aug. 2021, doi:10.6345/NTNU202101300.

[8] W. J. Mai , " Damage Assessment of rainfall-induced Sediment Disaster, "Chang Jung Christian University, Jan, 2013, doi: https://doi.org/10.6833/CJCU.2013.00203

[9] How to Make Bi-Directional Flex Sensors, accesased on Dec 5, 2023 https://www.instructables.com/How-to-Make-Bi-Directional-Flex-Sensors/

[10] G. Saggio et al., " Resistive flex sensors: a survey, " Smart Materials and Structures , Vol 25(1): 013001, Dec , 2015, doi:10.1088/0964-1726/25/1/013001.

[11] G. Saggio, " Mechanical Model of Flex Sensors Used to Sense Finger Movements, " Sensors and Actuators A: Physical, Vol. 185, pp.53-58, Jul, 2012. doi: https://doi.org/10.1016/j.sna.2012.07.023.

[12] F. Salman et al. ," A Wireless-controlled 3D printed Robotic Hand Motion System with Flex Force Sensors, " Sensors and Actuators A: Physical, Vol. 309, 2020. doi:10.1016/j.sna.2020.112004.

[13] B. Junseung et al.,"A Prototype of Flex Sensor Based Data Gloves to Track the Movements of Fingers, " KoreaScience, Vol. 8(4), pp.53-57, 2019. doi:10.30693/SMJ.2019.8.4.53.

[14] Interfacing Flex Sensor with Arduino, accessed on Nov 12, 2023 https://lastminuteengineers.com/flex-sensor-arduino-tutorial/

[15] Flex Sensor Hookup Guide, accessed on Nov 15, 2023 https://learn.sparkfun.com/tutorials/flex-sensor-hookup-guide

[16] One Axis Bending Sensor, accessed on Nov 20, 2023 https://vimeo.com/240203983

[17] Soft Angular Displacement Sensor Theory Manual, accessed on Nov 20, 2023 https://zhuanlan.zhihu.com/p/141967442

[18] Qwiic Flex Glove Controller Hookup Guide, accessed on Nov 20, 2023 https://youtu.be/8PYa828t3MQ

[19] Qwiic Flex Glove Controller Hookup Guide, accessed on Nov 20, 2023 https://learn.sparkfun.com/tutorials/qwiic-flex-glove-controller-hookup-guide

[20] K. S. Lin. "Implementation of I2C Communication Protocol with Microprocessor-based, " National Taipei University of Technology, May,2016.

[21] Y. H. Chiu. "The Development of Real-Time Balance Control System in Biped Robots Using Fuzzy Logic Algorithm," National Yang Ming Chiao Tung

University, Jul, 2006.

[22] ADS101x Ultra-Small, Low-Power, I²C-Compatible, 3.3-kSPS, 12-Bit ADCs With Internal Reference, Oscillator, and Programmable Comparator, accessed on Nov 20, 2023
https://www.ti.com/lit/ds/symlink/ads1015.pdf

[23] C. W. CHOU, " The Study of Communication Interface for Embedded System, " Chung Yuan Christian University, May, 2019. doi：10.6840/cycu201900156.