國立臺北教育大學
112 年度資訊科學系學生專題成果展
《專題報告》

作品名稱：基於離線語音辨識之聲控電風扇

指導老師：許佳興 教授

組　　員：陳昶勳

：江英碩

：郭逸凡

中華民國　112　年　12　月　22　日

國立臺北教育大學資訊科學系－學生專題成果展資料表

編號 （由系辦填寫）		
作品名稱	基於離線語音辨識之聲控電風扇	
班級	資四甲	
指導老師	許佳興 教授	
組　　員		
姓名	手機	E-mail
陳昶勳	0909256981	smilele0515@gmail.com
江英碩	0968620832	kevin57545@gmail.com
郭逸凡	0979439433	evankuo2017@gmail.com
指導老師簽章		

此專題研究致力於結合科技與人性，以提升日常生活便利性。我們聚焦於改進家電，鎖定電風扇的控制方式。過去的遙控器並未解決操作上的便利性問題，因此我們想要實做「聲控電風扇」之概念，透過聲音指令操作電風扇，提供多樣化操作，以此省去遙控器的媒介。

專題研究包括利用 Arduino 控制電風扇主板，透過電路設計實現聲控功能。使用 Snowboy 喚醒詞偵測引擎訓練語音模組，以實現透過 Raspberry Pi 平台進行離線語音識別。若偵測到特定指令，將發送訊號至 Arduino，進而控制電風扇來達成使用著欲執行之操作。

這項研究的目標是創新日常家電操作方式，提升人們生活品質，實現未來家電發展的新趨勢。

中華民國　112　年　12　月　22　日

目　　錄

第一章　前言

第一節　研究背景

俗話說「科技來自於人性」，追求效率與便利是人類的天性，所以人們會絞盡腦汁想出不需自行動手的方式來解決生活中遇到的問題。於是我們決定從最貼近日常生活的家電出發，在討論後，發現從「電風扇」著手會有相當良好的效果。

以往使用者總是要走到電風扇前方才能控制開關與風量段數等，如同早期的電視機一樣，造成使用上的不便，後來電視機有了遙控器，而電風扇也得以使用遙控器操作，方便了許多。

然而，經實際體驗後發現，可以遙控的電風扇仍有諸多不便，與電視機相比，我們會拿著遙控器看電視，卻不會總是拿著遙控器吹電風扇，除了使用頻率較低外，甚至還會時常找不到遙控器！

於是我們有了聲控電風扇這個想法，相信利用聲控就可以解決上述電風扇不夠便利的現狀，以創新日常家電操作方式，提升人們生活品質，實現未來家電發展的新趨勢。

第二節　研究目的

達成電風扇基本的聲控開關功能，利用語音辨識來達成電風扇風量的段數操作，如加速指令、或直接指定特定的電風扇風量，並可控制自然風、一般風、睡眠風等模式切換，做出我們自己理想的智慧電風扇，讓使用者擁有更便利的智慧家居生活。

第三節　適用族群

適用於廣大族群，上班族、家庭、學生、長者等各種欲擁有更便於控制的居家電風扇之所有使用群眾。

第二章　實作方法

第一節　使用步驟

一、 將電風扇、Raspberry Pi 分別接上電源

二、 電風扇即開始接受指令

1. 使用者利用呼叫指令「小北小北」取得電風扇的注意
2. 啟用後，使用者可對電風扇下達預設的聲控指令：

(1) **開啟**：開啟電風扇。

(2) **關閉**：關閉電風扇。

(3) **設定風量**：

將發出提示音，此時可任意指定 1 至 12 段風量，超過 15 秒未下達指令則發出錯誤提示音，逾時退出。

(4) **加速**：將風速提升一段，最高為 12 段風量。

(5) **開始擺頭**：電風扇開始擺頭。

(6) **停止擺頭**：電風扇擺頭。

(7) **定時**：

將發出提示音，此時可任意指定 1 至 8 小時，超過 15 秒未下達指令則發出錯誤提示音，逾時退出。

(8) **不定時**：取消定時。

(9) **一般風**：電風扇模式設為一般風。

(10) **睡眠風**：電風扇模式設為睡眠風。

(11) **自然風**：電風扇模式設為自然風。

以上除**設定風量**、**定時**為不同的兩連音外，其餘收到指令後皆會發出一聲提示音，若超過 30 秒未下達指令則需要重新喚醒電風扇。

電風扇成品圖如下:

操作範例:

- 呼叫「小北小北」➔電風扇發出兩聲提示音
- 呼叫「開啟」➔ 電風扇開始運轉且風量預設為「三段風」

- 呼叫「加速」➔ 電風扇風量提升至「四段風」

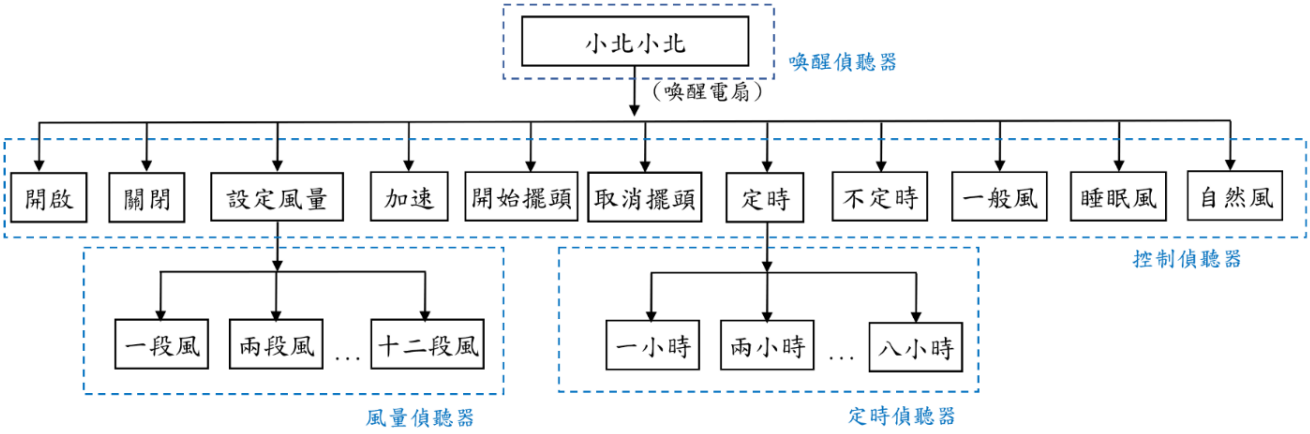
- 呼叫「定時」➔ 電風扇發出兩聲提示音 ➔ 說出「五小時」➔定時至五小時、面板閃爍「5H」並停留 3 秒鐘供使用者確認

- 呼叫「風量」➔ 電風扇發出兩聲提示音 ➔ 說出「七段風」➔電風扇加速至七段風

- 呼叫「自然風」➔ 電風扇進入自然風模式

- 呼叫「睡眠風」➔ 電風扇進入睡眠風模式

經由分析電風扇的功能及操作邏輯後，設計之喚醒詞指令可由以下樹狀圖來呈現。其中，長方形框為可供電風扇辨識之指令，箭頭代表使用者達成某指令後，需再接續下達命令以達成最終執行，藍色虛線框則為負責掌管該指令之偵聽器(不同的指令隸屬於不同偵聽器)。

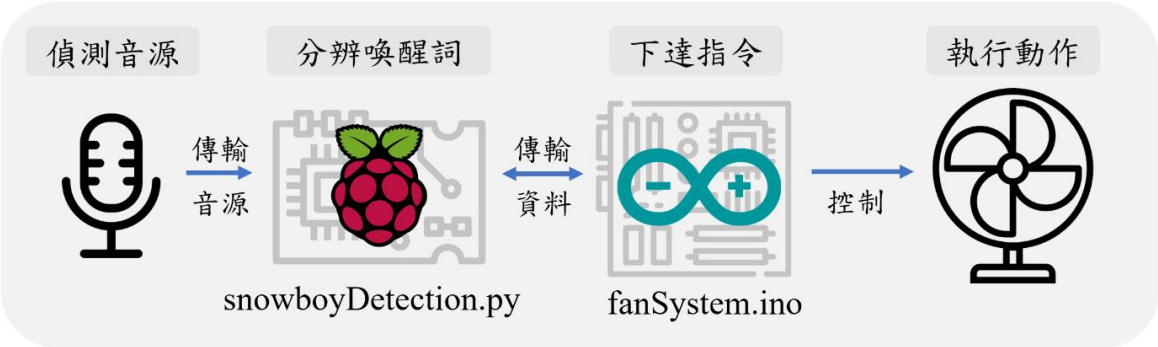


在下達命令前，以喚醒偵聽器監聽，喊出「小北小北」來喚醒電扇。接著切換至控制偵聽器，此時電風扇進行監聽眾多功能之指令(開啟、擺頭、定時等...)，若隔一段時間使用者未發出指令，則會再度回到閒置狀態。

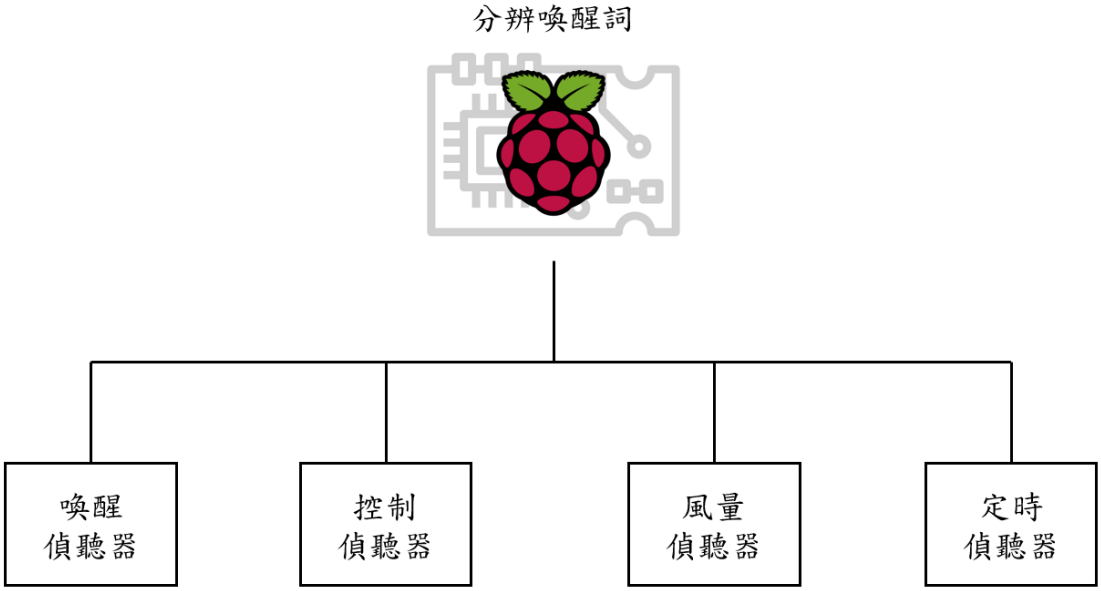
喚醒電扇後，說出「設定風量」可直接指定段數，切至風量偵聽器，「定時」可直接指定時數，切至定時偵聽器，完成風量或定時指令後，切回控制偵聽器，在電扇回到閒置狀態前，可以再次下達其他指令。

第二節　系統概述

一、 系統配置



系統由電風扇、麥克風、**Arduino Uno R3**、**Raspberry Pi** 四大元件組成



作為聲控模組的 Raspberry Pi 切換並開啟不同的偵聽器以監聽聲控指令，每個偵聽器將根據偵測到的關鍵字呼叫對應副程式，執行功能，下圖為每個偵聽器呼叫副程式一覽:

喚醒偵聽器	控制偵聽器	風量偵聽器	定時偵聽器
hello_callback()	close_callback()	one_callback()	onehr_callback()
	open_callback()	two_callback()	twohr_callback()
	accelerate_callback()	three_callback()	threehr_callback()
	startOSC_callback()	four_callback()	fourhr_callback()
	stopOSC_callback()	five_callback()	fivehr_callback()

	sleepwind_callback() naturalwind_callback() normalwind_callback() setspeed_callback() settime_callback() cancel_settime_callback()	six_callback() seven_callback() eight_callback() nine_callback() ten_callback() eleven_callback() twelve_callback()	sixhr_callback() sevenhr_callback() eighthr_callback()

- **電風扇**：

  電風扇採用市面直流 DC 電風扇，其硬體具備良好的擴充與修改性。

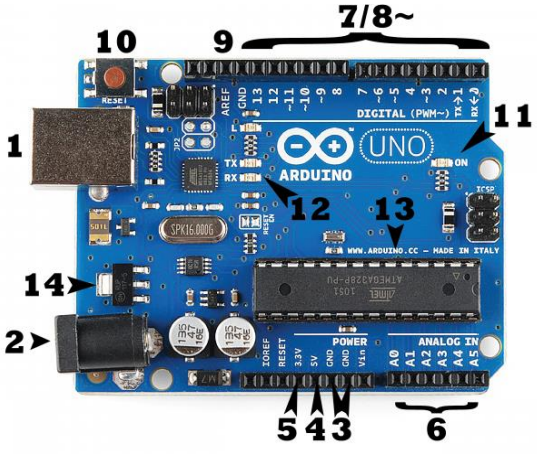
- **麥克風**：

  外接麥克風，負責外界聲音接收且採全指向型收音，增加聲音辨識範圍、角度，並連結至 Raspberry Pi 傳輸音源。

- **Arduino Uno R3**：

  Arduino 程式負責管理、執行整個電風扇的控制流程，包括：觸發功能、狀態訊息輸出與傳遞、狀態計時等。Arduino 板在整個電風扇系統中扮演控制電風扇之主控版的角色，並與聲控模組(Raspberry Pi)進行溝通。

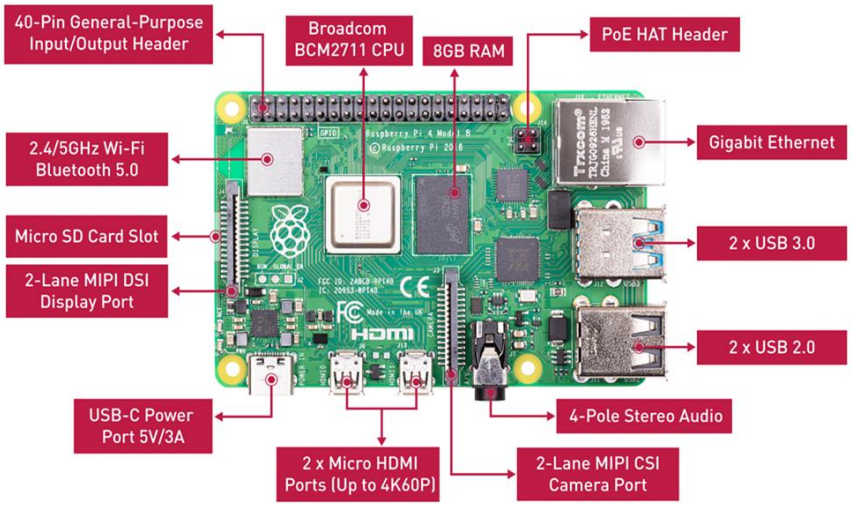
  Arduino Uno R3 具備良好的擴充性及易使用性，具有 14 個數位輸入/輸出引腳（其中 6 個可用作 PWM 輸出），6 個模擬輸入、USB 連接、電源插孔，復位按鈕等。它包含支持微控制器所需的一切；只需使用 USB 線即可將其連接到計算機電腦，或使用 AC-DC 適配器或電池為其供電即可開始使用。



- **Raspberry Pi**：

此次研究使用的 Raspberry Pi 4 computer Model B 是基於 Linux 的單晶片電腦，由英國樹莓派基金會開發，搭載作業系統的小型電腦可提供便攜性以及可運行多種軟體的高自由度以及豐富的周邊支持。

以此 Raspberry Pi 做為系統的聲控模組，主要執行偵聽器管理、聲音的辨識、生成控制電風扇的訊號。它會接收 Arduino 的訊息以及外界的聲音，對這些資料經過計算後，將正確的控制訊息傳遞給 Arduino，使 Arduino 得知該對電風扇進行何種控制。



三、 運行軟體與介面

- **Python**：

Raspberry Pi 之功能均由其中運行之 python 程式完成，此程式引入了 Snowboy 模組以達成聲音辨識，並且利用多條執行緒來同時處理多個工作，如：Arduino 的資料交換、持續開啟偵聽器等。

其中，持續開啟的偵聽器只能有一個，但系統中管理了四種偵聽器，與 Arduino 資料交換的主要目的即為告知 Raspberry Pi 何時該切換為哪種偵聽器進行監聽。

- **C ：**

Arduino IDE 為專為於 Arduino 設計的集成開發環境，可讓開發者編寫程式碼，並將程式碼上傳到 Arduino 板上運行。其支援 C 語言和 C++語言，並且具有內建的函式庫，使開發者可以輕鬆地使用各種感應器和模組。

此系統在 Arduino IDE 中撰寫一 C 語言程式，其完成了上述 Arduino 控制至風扇主控版、與 Raspberry Pi 交換資料等功能，透過接收 Raspberry Pi 傳遞訊號，並依程式流程進而控制電風扇，以達成使用者想要的動作。

- **Snowboy：**

Snowboy 是一個可高度自定義的喚醒詞(hotword)偵測引擎，可以自由建立想要的喚醒詞，並透過一連串的樣本訓練後，就可在裝置上進行離線識別。其特點有：

- 高度自定義：可以在官網上或使用 API 訓練樣本集，並產生模型檔(.pmdl 檔)。
- 使用深度神經網路訓練。
- 保護隱私：Snowboy 不需連結到網路就可以進行辨識，且不會將語音傳輸到雲端。
- 硬體需求低：可在任何嵌入式系統上執行，如在 Raspberry Pi 上執行時，只佔用不到 10%之 CPU 使用率。
- 高相容性：目前兼容眾多作業版本(如 Raspberry Pi OS、Mac OS、 Ubuntu 14、 ARM64 等...) ，並且支援現今普遍的程式語言環境(如 C/C++、Java、Python 等...) 。

第三節　實作步驟

一、 實現 Arduino 控制電風扇主板

首先使用數條杜邦線，將 Arduino 之 output 與電風扇電路板的各接點進行焊接，如此便可使用 Arduino 輸出負電，使用負緣觸發以達成電風扇按鈕操作(包括開啟、關閉、加速、定時、擺頭等)。

二、 撰寫 Arduino 端程式碼

我們使用 Arduino IDE 撰寫 C 語言程式運行於 Arduino Uno 板，使其從 Arduino Uno 板輸出高低電壓變化，以達到控制電風扇之各種行為。檔名為「fanSystem.ino」，主要功能包括：

(一) 風扇控制

電風扇的控制板按鍵接口皆焊上杜邦線並連接到 Arduino Uno 板，故 Arduino Uno 板可以經由輸出高低電壓變化直接控制電風扇的原有功能，並且可以短時間觸發多次控制以達到快速調整風量、定時時間等。

(二) 計時系統

如喚醒電風扇後五秒鐘內可以接收指令，超過五秒後電風扇將會離開喚醒狀態，此過程整個系統的狀態轉換及計時本身都由 Arduino 負責。

(三) 與 Raspberry Pi 的序列埠通訊

Arduino 需要 Raspberry Pi 提供聲控指令才可得知如何控制電風扇實際的行為，因此 Arduino 要與 Raspberry Pi 進行數據傳輸。而我們選用序列埠通訊來實現，當 Raspberry Pi 偵測到聲控關鍵詞後，Arduino 會經由序列埠接收來自 Raspberry Pi 的指令，以執行相關動作。

此外，由於 Arduino 配有計時系統(如喚醒電風扇數秒後離開喚醒狀態) ，這才是直接決定 Raspberry Pi 應該如何切換偵聽器的不可或缺條件。因此，Arduino 和 Raspberry Pi 的「雙向」通訊是必要的，

所以我們也同樣經由序列埠通訊實現了 Raspberry Pi 接收 Arduino 訊號。

(四) 提示音輸出

Arduino 亦有接線對電風扇的蜂鳴器進行控制，根據不同的聲控指令，使電風扇本身的蜂鳴器發出與原廠預設相異的聲音，以達到提示使用者的效果。

三、 撰寫 Raspberry Pi 端程式碼

透過 python 撰寫承載系統邏輯運作之程式碼，此程式藉由偵聽器來偵測外界音訊，並執行相對應之動作。檔名為「snowboyDetection.py」主要功能包括：

(一) 聆聽及識別功能

Python 程式主要功能為利用 Snowboy 關鍵字模型建立偵聽器，由於需要監聽的關鍵字眾多，我們建立了四個偵聽器，將各關鍵字辨識模型根據其功能分別分派給不同的偵聽器，如此，每個偵聽器配有多個模型，可以辨識多個關鍵字，但每個偵聽器可偵測的關鍵字數量又不會太多，減少了太多關鍵字聲音模型會有互相干擾的問題。

因此我們利用偵聽器實現外部聲音的偵測和辨識，根據偵測到的特定關鍵詞觸發不同的回應，如開啟、關閉、加速、設定風量、模式切換等。

(二) 與 Arduino 的雙向序列埠通訊

一切的聲控指令需要經由 Arduino 才得以轉換成電風扇實際的行為，因此 Raspberry Pi 要與 Arduino 進行數據傳輸。而我們選擇利用序列埠通訊來實現，由偵測到的關鍵詞，經由序列埠發送指令至 Arduino，以執行相關之動作。

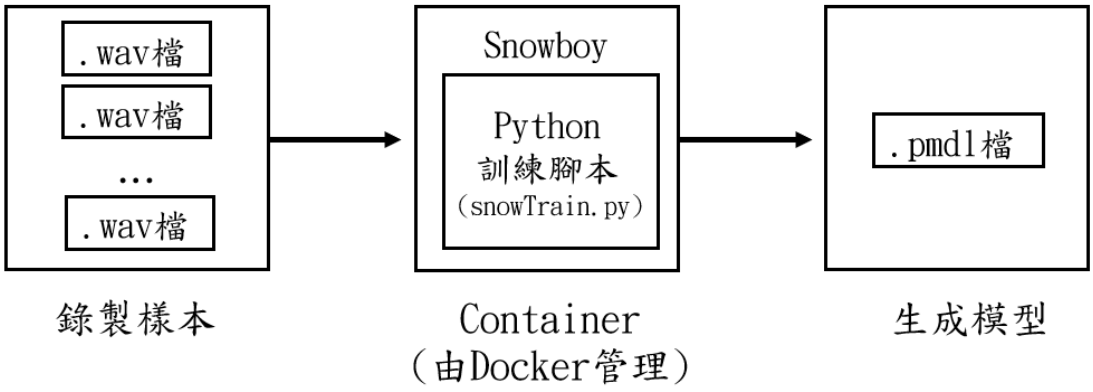
Raspberry Pi 負責偵聽器管理，但是系統的許多狀態資訊，如時間狀態等涉及偵聽器切換的訊號由 Arduino 處理，所以需要建立雙向通訊。

(三) 多執行緒運作及互斥控制

在 Python 程式碼中我們建立兩條執行緒，一條負責執行與 Arduino 的序列埠通訊，不間斷的監聽並接收資料，另外還有一條主執行緒會開啟某個偵聽器，並持續監聽。

需要注意的是，負責接收 Arduino 資料執行緒的目的，就在於根據接收到的資料更改偵聽器，而主執行緒負責的就是執行偵聽器，因此兩條執行緒會需要利用許多共享變數來進行溝通，這些溝通的過程會產生 Synchronization 問題需要處理。而我們採用自己撰寫的 Binary Semaphore 以解決問題，由於只有兩條執行緒，且執行緒需要暫停的時間很短，因此 Semaphore 採用 Busy waiting 來製作，其中用 Peterson's solution 來滿足 Semaphore 本身的互斥存取。

四、 進行 Snowboy 語音模型訓練



為了實現 Snowboy 的語音辨識功能，我們需要自行訓練需要的喚醒詞，Snowboy 的喚醒詞需要多個 wav 音檔，所以我們各自錄製了大量的資料集以備訓練模型。

從官方文檔得知有語音訓練環境相關的映像檔存在，因此我們利用 docker 建立可以執行訓練模型程式的容器，並且自行撰寫了可以將資料集上傳、完成訓練的 python 腳本。

這個腳本執行完畢後即輸出 pmdl 模型檔，可以直接套用至 Raspberry Pi 的 python 主程式中，以做為偵聽器辨識的聲控模型使用。

第三章　結果與討論

第一節　結論

透過此次專題研究，我們團隊完成了軟硬體之間的協調運作，從 Raspberry Pi 到 Arduino 的運作，再到對於電風扇的控制，所對應的程序和資料傳輸、編排，每個步驟都是環環相扣的，且透過分工和合作，可以促成團隊間的協調，以強化團隊的溝通的能力。

在這個專題也結合了人工智慧與語音模型方面，訓練過程我們應用了 docker、virtual box 等工具建置環境，多方嘗試訓練出更好的語音模型，最終在調節每個語音模型的靈敏度後，呈現出了好的成果。

當有軟、硬體上的錯誤產生時，利用大學所學的分析與除錯理論，不斷地反覆試驗(trial and error)，透過除錯經驗來找出問題之所在，並應用了 Semaphore、Multithreading 等理論課學習到的技術來解決問題。

最後，此次專題讓我們結合了大學四年所學知識，並呈現在作品之上，完成一臺實用的「聲控電風扇」。

第二節　未來展望

目前此聲控電風扇已可達成基本功能的控制，我們未來可以往持續加強聲控的靈敏度與精準度的方向前進，並且增加更強大、方便且有趣的的功能控制：

(1) 持續調整收音麥克風的靈敏度，以改善偵測的精確度。

(2) 加大麥克風的可收音的距離，以增加使用者操控的方便性。

(3) 增加聲控模型的樣本數與樣本的多樣性，以加強聲控的準確度。

(4) 加入聽聲辨位功能，讓電風扇可以擺頭至發聲者的方位。

(5) 開發使用者 APP，將電風扇連網，讓使用者可自定義聲控關詞。

參考資料

期刊文獻

1. A Practical, Real-Time Speech-Driven Home Automation Front-end Theodoros Giannakopoulos, Nicolas – Alexander Tatlas, Todor Ganchev and Ilyas Potamitis https://researchr.org/publication/GiannakopoulosTGP05

2. Training Wake Word Detection with Synthesized Speech Data on Confusion Words – Yan Jia, Zexin Cai, Murong Ma, Zeqing Zhao, Xuyang Wang, Junjie Wang, Ming Li https://arxiv.org/abs/2011.01460

網路資料

1. Snowboy Personal Wake Word Recorder- https://github.com/rhasspy/snowboy-seasalt

2. Snowboy 本地模型訓練基於樹莓派 4b - https://lneverl.gitee.io/posts/6c144ece.html

3. python 之使用 snowboy 離線語音喚醒- https://blog.csdn.net/weixin_45729594/article/details/119478227

圖片來源

1. https：//www.semanticscholar.org/paper/PID-CURRENT-CONTROL-TECHNIQUE-FOR-THREE-PHASE-MOTOR-Yusoff/0bde280c60c793e8c68c310de626bc84ecf26b3f/figure/1

2. https：//www.amazon.com/Raspberry-Pi-Computer-Suitable-Workstation/dp/B0899VXM8F

3. https://www.flaticon.com/