臺北市立大安高級工業職業學校
綜合高中
專題報告
電子寵物
Electronic Pet

學生 組長：李侑璋
組員：孫御宸
組員：江侲緯
組員：蔡文浩
指導老師：王村益老師

中華民國 111 年 1 月

摘要

研究目的:隨著近些年物價飛速地上漲，為了降低養寵物所需要耗費的精力以及讓想養寵物的人不需花這麼多的開銷在寵物身上，而去研發電子寵物。

研究結果:我們為了實現電子寵物能夠像真實的寵物一樣能夠聽主人的呼喚並朝著主人的方向移動，所以我們使用了 Nano 33 BLE sense 和 LM386 語音辨識模組去判斷主人聲音及具體位置,在運用 PCA9685 操控馬達使電子寵物能依照聽取到的指令做不同動作。

理論探討:分成 4 個這專題主要的用到的理論部份，分別為語音辨識原理、機器學習、聲音方位辨識、四足步行。

專題準備:分成硬體架構和軟體架構，硬體架構主要是我們這專題所用到的一些物件，而軟體架構則是我們語音辨識、馬達控制的程式。

專題成果:電子寵物的各個指令解說和電子寵物在收到指令後會做出什麼相對應的動作。

結論與建議:寫未來展望還有結論。

目錄

圖目錄

第一章前言

1-1 研究背景與動機

隨著近幾年來物價一直上升，但是人們的工作所得卻緩慢成長甚至有時候沒漲，導致越來越多人覺得負擔不起養育小孩的費用，因此他們就去選擇養寵物因為養寵物所需的花費、精力都比培育一個小孩到他(她)成年來的還要少。自 2020 年以來，全世界已飽受新冠病毒的肆虐，造成人人只能待在家防疫，而且還有一些人失去了自己原本的工作，這也讓他們自己的生活已經陷入了困難，加上還要負擔寵物的花費，造成他們生活更加艱困。這時如果是電子寵物就不用去煩惱一些寵物的基本開銷，而且許多人養寵物的原因就是希望不要讓自己生活太孤單，而電子寵物也可以幫你解決掉這個問題，另一方面社會中常有獨居長者也能夠靠電子寵物去解決他們的一些問題，這些都是我們想做出電子寵物的動機和背景。

1-2 研究　目的

1. 讓研究出的電子寵物可以陪伴怕孤單的人和小朋友。
2. 減少養寵物的花費。
3. 希望能做出能夠講一句話就可以執行指令的電子寵物，當電子寵物接收到我們的指令後，可以根據聲音來源到達我們所在位置。
4. 如果有障礙物擋在其中，也能夠避開順利到達，未來希望電子寵物可以不只到達發聲人的位置，而是可以完成一些簡單的指令，例如拿取東西。

1-3 研究方向、步驟與進度

1-3-1 研究方向

如圖 1 所示為電子寵物研究方向，Arduino Mega 2560 及 Nano 33 ble sence 為主要控制板和聲音辨識，這兩塊就如同寵物大腦，Nano 33 ble sence 接收特定指令後，再交由 Mega 2560 發送訊號控制寵物活動。

身體將使用木板進行雷射切割，切出所需的部件後再以馬達、螺絲及強力膠進行組裝，藉由控制伺服馬達完成一系列指令。

在方位辨識上將使用四塊LM386判斷四方位聲音大小，比對完音量大小後，控制馬達將寵物正面轉向至音量最大那方。

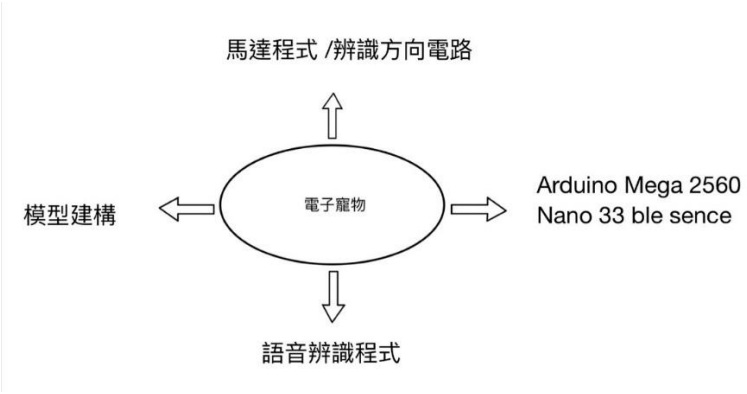

圖 1 研究方向圖

1-3-2 研究步驟(流程圖)

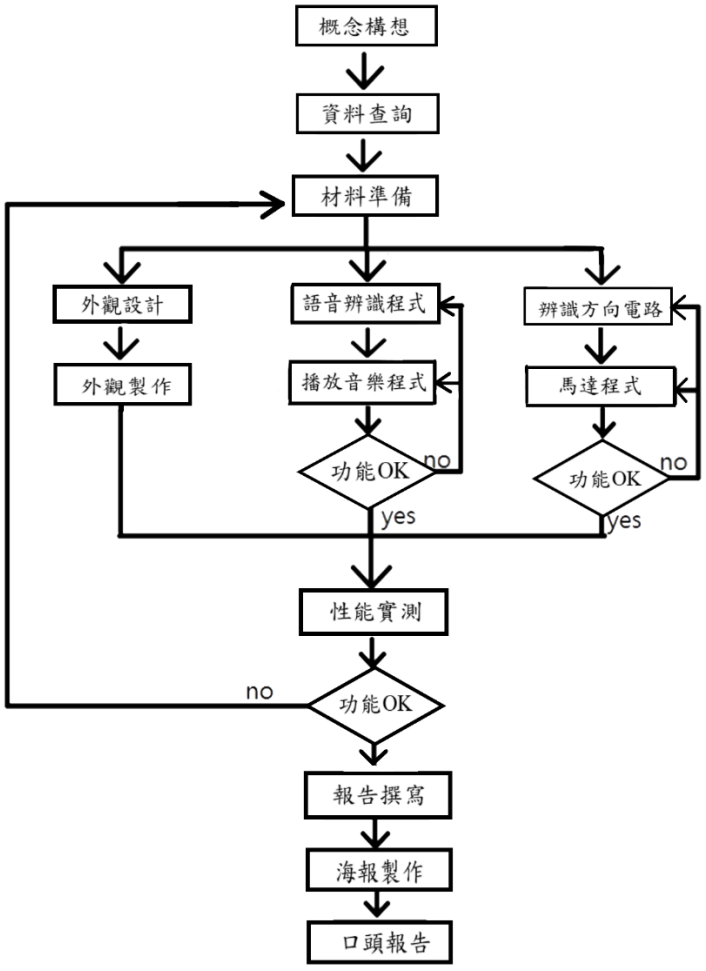

圖 2 流程圖

1-3-3 研究進度(甘特圖)

週次（日期） 工作項目	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	負責成員
概念構想	■	■																	李侑璋、孫御宸、江侃緯、蔡文浩
資料查詢		■	■	■	■														李侑璋、孫御宸、江侃緯、蔡文浩
材料準備			■	■	■	■	■												李侑璋、孫御宸、江侃緯、蔡文浩
外觀設計和製作							■	■	■	■									李侑璋、江侃緯、蔡文浩
語音辨識程式			■	■	■	■	■	■	■	■									江侃緯、蔡文浩
辨識方向電路								■	■	■	■	■	■	■	■				孫御宸、江侃緯
播放音樂程式									■	■	■	■	■	■					孫御宸、蔡文浩
馬達程式								■	■	■	■	■	■	■	■	■			江侃緯、蔡文浩
性能實測														■	■	■	■		李侑璋、孫御宸、江侃緯、蔡文浩
報告編寫														■	■	■	■	■	李侑璋、孫御宸、江侃緯、蔡文浩

海報製作																		李侑璋、孫御宸
預定進度	02%	05%	08%	10%	13%	15%	18%	20%	25%	30%	35%	45%	50%	65%	75%	85%	95%	100%	累積百分比%

圖 3 研究進度圖

第二章 理論探討

2-1 語音辨識原理

電子寵物主要下達指令方式為語音，為了讓電子寵物可以精準接收傳達的指令，語音辨識與機器學習，可達成以上需求。

特徵提取方法介紹

如圖 4 上面有肉眼可見聲音特徵，可直接做特徵的提取。

而圖 5 這張圖，可以把它理解圍聲音錄下來最原始的樣子。

要從圖 5 處理圖 4 的特徵基本上經過一個動作，倒譜分析，分為兩個步驟，第一個步驟稱為包絡，在講包絡之前我們要先了解共振峰這個東西，它可以解讀為語音的主要頻率成分，而這些共振峰就是攜帶語音辨識的屬性，而包絡，就是取共振峰的平滑取線，第二個步驟是提取上面的頻譜細節，完成這兩個步驟後，在經過倒譜分析，結合成上面這張線性的聲譜圖，整個過成大致上就是這樣。

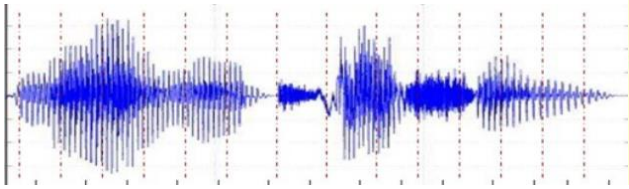

圖 4 聲譜圖

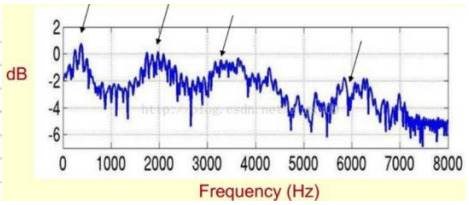

圖 5 頻譜圖

聲音特徵學習

聲音的特徵學習主要分為線性預測編碼（Linear Predictive Coding），和梅爾頻率倒譜系數（MeI-Freguency CeptraI Coefficients）兩種，其中的 LPC 是一 個能夠很好的反映語音的聲道特徵，但是卻對語音的其他特徵無能為力的語音分析技術，通常應用於音樂的製作，而非機器學習。

而 MFCC 擁有獨特的倒譜提取方式，其更加符合人類的聽覺原理，因而也是最為普遍、最有效的語音特徵提取演算法，被廣泛應用於機器的學習和訓練。

MFCC 的主要流程大致分為

1. 先對語音進行預加重、分幀和加窗。

預加重簡單來說增加訊號中高頻，使訊號頻譜變得平坦，移除頻譜傾斜，來補償語音訊號是到發音系統所抑制的高頻部分，簡單來說就是消除發生過程中聲帶和嘴唇的效應，使語音訊號變的平穩。

分幀是指在跟定的音訊樣本檔案中，按照某一個固定的時間長度分割，分割後的每一片樣本，稱之為一幀，這裡需要區分時域波形中的幀，分割後的一幀是分析提取 MFCC 的樣本，而時域波形中的幀是時域尺度上對音訊的取樣而取到的樣本。

加窗在對音訊進行分幀之後，需要對每一幀進行加窗，以增加幀左端和右端的連續性，減少頻譜洩漏。在提取 MFCC 的時候，比較常用的視窗函式為 Hamming 窗。

這三個動作可以理解成提取的動作。

2. 對每一個短時分析窗，通過 FFT 得到對應的頻譜。
3. 將上面的頻譜通過 Mel 濾波器組得到 Mel 頻譜。
4. 在 Mel 頻譜上面進行倒譜分析 （取對數，做逆變換，實際逆變換一般是通過 DCT 離散餘弦變換來實現），這樣就可獲得 Mel 頻率倒譜系數。

MFCC，這個 MFCC 就是這幀語音的特徵。

補充:

FFT(Fast Fourier transform)快速博立葉變換:

這個演算法，可以將一個訊號轉換成頻域(頻率圖)，因為有些訊號在一般在時域上很難看出或判斷出其中的特徵，所以如果利用這個演算法轉換成頻域之後，可輕易的判斷和提取特徵的部分，另外，這個演算法也擁有將一個訊號進行頻譜提取的功用，在頻譜分析方面十分的方便。

2-2 機器學習

機器學習(Machine Learning = ML) 是人工智慧 (AI) 的一種，它透過演算法將收集到的資料進行分類或預測模型訓練，在未來中，當得到新的資料時，可以透過訓練出的模型進行預測 ，如果這些效能評估可以透過利用過往資料來提升的話，就叫機器學習。機器學習包含不同類型的學習模式，並使用各種演算技術，根據資料的性質和期望結果，可以採用監督式、非監督式、半監督式或強化式共四種學習模式。

1. 監督式學習（Supervised Learning） 在訓練的過程中提供物件（向量）和預期輸出，可以是「有標籤」的分類資料或是一個連續的值（迴歸分析），例如輸入了大量已標示清楚標籤的腳踏車和機車給機器後，讓機器分辨尚無標簽的照片是機車還是腳踏車。類似於動物和人類的認知感知中的「概念學習」(concept learning)。

   註：迴歸分析是運用一個或一組變項來預測另一個變項的統計技術總稱，被預測的變項稱為效標變項或依變項，預測變項也可稱為自變項。只根據一個預測變項來預測效標變項的迴歸分析稱為「簡單迴歸」（simple

regression），若預測變項為兩個或兩個以上則稱為「多元迴歸」（multiple regression）。例如，單用學生國中在校成績來預測其國中基本學力測驗分數，即為簡單迴歸分析的例子，若同時用學生在校成績和智商兩個變項來預測其國中基本學力測驗分數，則為多元單迴歸分析的應用。

圖 6 監督式學習

2. 半監督式學習（Semi-supervised learning） 介於監督學習與非監督學習之間。這樣的學習方式會先將「有標籤」的資料和「無標籤」的資料切出一條分界線，再將「無標籤」資料依據整體分布，調整出兩大類別的新分界。不需要百分之百大量的「有標籤」資料，讓半監督學習同時能降低成本又具有非監督式學習高自動化的優點。

圖 7 半監督式學習

3. 非監督式學習（Unsupervised Learning） 這樣的機器學習方式不需要人力事前的輸入標籤，僅僅提供了輸入範例，便直接以沒有標準答案的資料來訓練機器，在學習時機器會自動找出潛在類別的規則，並且反覆以經過測試後的學習結果應用到新的案例上。

圖 8 非監督式學習

4. 強化學習（reinforcement learning） 源自於心理中行為主義理論的學習方法，即如何在環境給予的獎懲刺激下，一步步形成對於這些刺激的預期，來產生能夠獲得最大利益的習慣性行為，強調的是透過環境而行動，並會隨時根據輸入的資料逐步修正。這個方法具有普適性，因此在其他許多領域，如博弈論、統計學及遺傳算法等都有研究。

圖 9 強化學習

機器學習的流程共有以下七個步驟：

1. 收集資料（Gathering data ）
2. 準備數據（Preparing that data）
3. 選擇模型（Choosing a model）
4. 訓練機器（Training）
5. 評估分析（Evaluation）
6. 調整參數（Hyperparameter tuning）
7. 預測推論（Prediction）

機器學習的優點如所示：

1. 發掘見解：
   機器學習可協助識別結構化和非結構化資料中的模式和架構，協助了解資料代表的意義。
2. 改善資料完整性 ：

機器學習相當適合用來進行資料採礦,且能進一步發展,隨著時間改善能力。

3. 增強使用者體驗 ：
   自適性介面、目標式內容、聊天機器人和語音虛擬助理，均是機器學習協助將客戶體驗最佳化的範例。
4. 降低風險 ：
   機器學習會隨著詐騙策略不斷進步，監視及識別新的模式，在嘗試成功前捕捉。
5. 預測客戶行為 ：
   機器學習可以挖掘客戶相關資料來協助識別模式與行為,能讓您將產品建議最佳化，並提供最好的客戶體驗。
6. 降低成本 ：
   機器學習應用程式為進流程自動化，可節省時間和資源，讓您的小組專注於最重要的事物上。

一提到機器學習，深度學習也與之脫離不了關係，機器學習之所以稱為「深度」,是因為包含許多層神經網路，以及大量複雜且分散的數據。為了實現深度學習，系統會與多層神經網路互動，萃取出更高層次的結果。神經網路等深度學習技術經常用於圖像分類,因為這類技術能在複雜的情況下有效地找到圖像中的相關特徵，藉由強大的圖像分類技術，機器學習能夠應用在零售、金融服務、電商、科學、醫療保健、建築和能源等。

適合機器學習的程式語言有 Python、C++、JavaScript、Java、C#、Julia、Shell、R、TypeScript 和 Scala （用來與大數據互動的語言）， 在這之中最常使用的是 Python，由於 Python 是一種直譯式 (interpreted)、開放原始碼、物件導向的語言，因此它的程式碼在由 Python 虛擬機器執行之前，會先轉換成位元碼 (bytecode)。

Python 對於機器學習優點有以下幾點：

1. 有大量現成可用的強大套件，其中有些是專為 ML 開發的套件。
2. 可以很容易地快速建立原型 (prototyping)。
3. 有各式各樣的協作工具可用。
4. 從特徵截取、建立模型，一路到更新 ML 解決方案，資料科學家都能使用 Python 來作業，不必在不同階段變換不同的語言。

2-3 聲音方位辨識

這次我們是用四塊 LM386 在四個不同方位去判斷聲音的大小,而大小判斷則

是用類比數值 0 到 1023 去表示聲音大小，然後再去判斷要執行哪一個迴圈比較合適，所以程式會分成四個迴圈，每個迴圈都代表其中一個方位收到的聲音比其他三個方位大，在讓電子寵物轉到相對應的方向。

2-4 四足步行

四足動物行走模式

1. 四足動物行走

我們的電子寵物，移動方式預計是以四足動物的行走方式，來進行移動。

如圖 10 所示四足動物的前肢運動就像人類一般的走路模式，重心偏向腳後跟的部分，行熟主要以腳發力。

如圖 11 所示四足動物的後肢運動就像人類用很長的腳踮著腳尖走路，而動物的大小也會影響到身體向下的移動幅度，動物越小越輕，身體向下運動的延遲和幅度就越小。像馬這樣的大型動物，向下運動的延遲和幅度相當明顯。

如圖 12 所示一般的四足動物行走，腳的運動是兩隻兩隻來看的，左前腳和右後腳是一組，右前和左後是一組，互相配合來達到行走。

如圖 13 所示從頭懂得角度來看，四足動物在行走時，腳接觸的三個點所為出的三角形，涵蓋了整體的重心，使動物在行走時，能夠頻穩地向前。

如圖 14 所示從以上這些特點也可觀察到，動物前後腿著地的接觸點會靠得非常近。

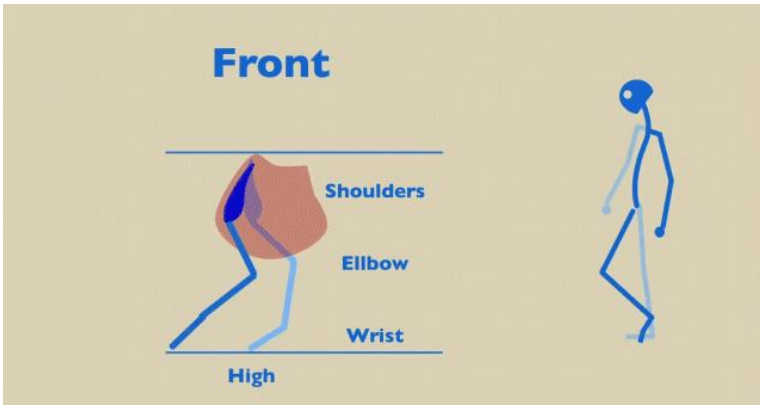

圖 10 前腳示意圖

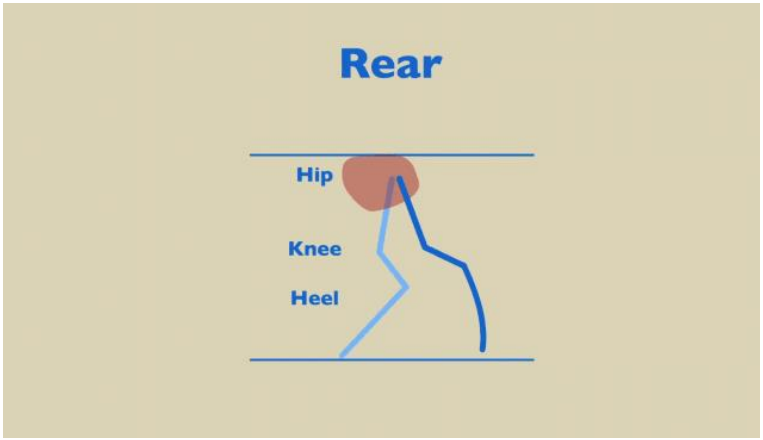

圖 11 後腳示意圖

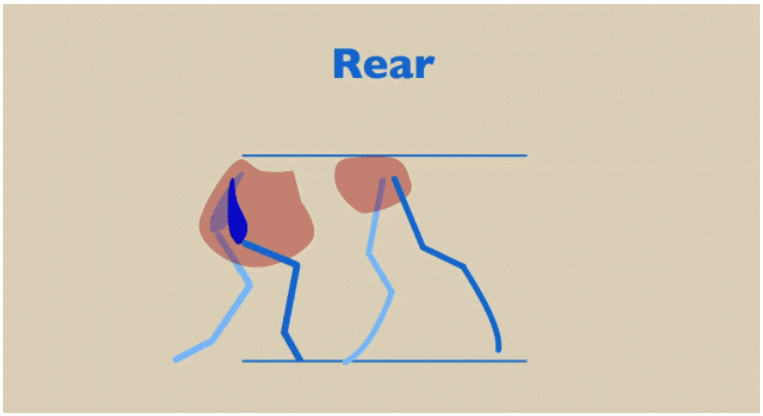

圖 12 前後腳結合圖

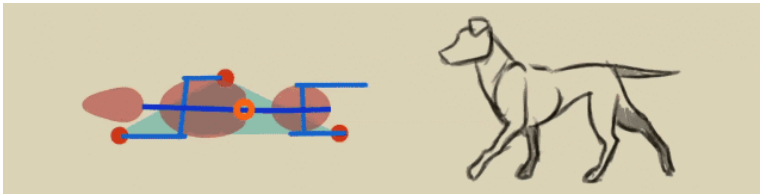

圖 13 頂視圖

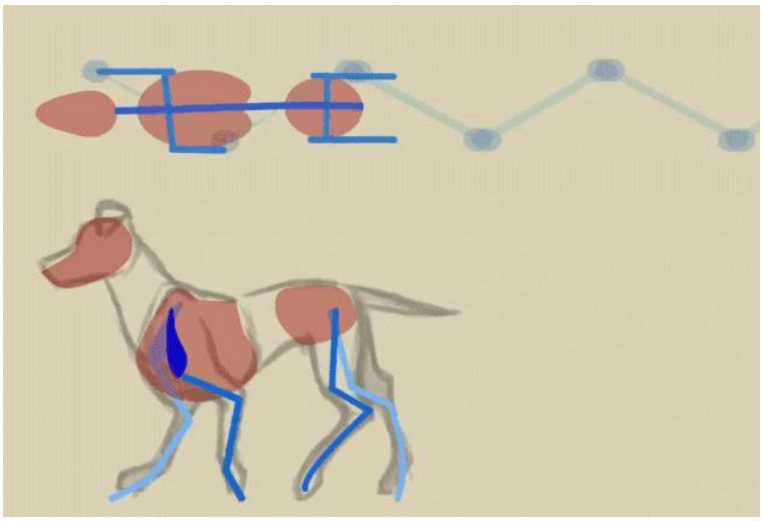

圖 14 整體行走示意圖

第三章 專題準備

3-1 硬體架構

3-1-1 Nano 33 BLE sense

外觀如圖 15、圖 16 所示，Arduino Nano 33 BLE Senseg 是用 3.3V 工作電壓，64MHz 時脈，1MB Flash，256KB SRAM 14 個 DIO(數位輸出)腳[全部具備 PWM(類比輸出)]，8 個 ADC(類比數位轉換器)腳，無 DAC(數位類比轉換器)腳，USB 晶片內建於 MPU(微處理器)中，內建麥克風、溫溼度感測器、氣壓計、亮度與色彩感測器，9 軸感測器， 以及藍芽安全 IoT 連線功能等等，適合各種物聯網專案用。這塊板子在我們專題中是用來做語音辨識。

圖 15 Arduino Nano 33 BLE Sense 正面

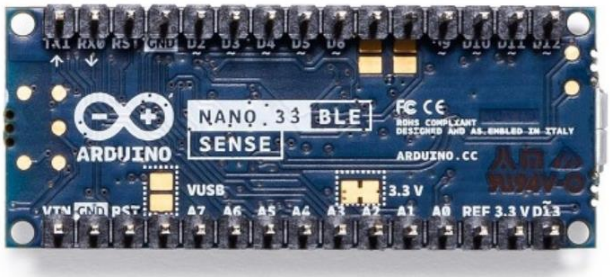

圖 16 Arduino Nano 33 BLE Sense 反面

3-1-2 LM386 聲音感測模組

外觀如圖 17 所示，LM386 聲音感測模組是採用音頻處理晶片 LM386，它可以對音頻信號進行 200 倍放大、類比信號的電壓輸出幅值調整、信號輸出指示，主要用途是用來檢測環境聲音的有無和判斷聲音強度的大小等。我們使用這塊板子來做判斷聲音來源。

圖 17　LM386 聲音感測模組

原理

LM386 主要應用於低電壓的消費產品。原始電壓增益為 20，但在 1 腳和 8 腳之間增加一隻外接電阻和電容，便可將電壓增益調為任意值，直至 200。輸入端以接地為參考，同時輸出端被自動偏壓到電源電壓的一半，在 6V 電源電壓下，它的靜態功率消耗僅為 24mW，使得 LM386 特別適用於電池供電的場合。

內部圖

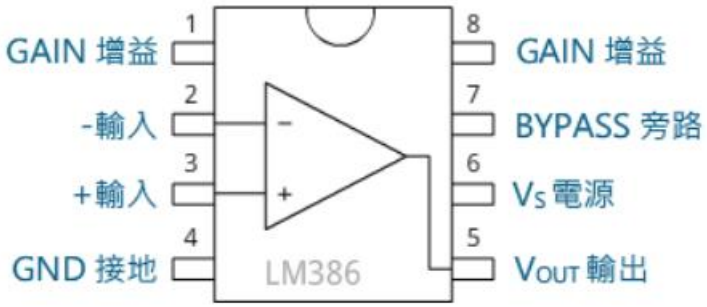

圖 18　LM386 接腳圖

3-1-3　PCA9685

外觀如圖 19、圖 20 所示，PCA9685 使用 I2C 介面，不佔用 GPIO 介面（預設位址 0x40），同時支持樹莓派（3.3v 電壓）和 arduino（5v 電壓），但兩種主控不能同時控制，支援 16 路 PWM 通道輸出，可以控制 16 路馬達或者 LED 燈，12 位元解析度，可調 PWM 頻率高達 1.6KHz，所有 PWM 輸出線上都放一個 220 歐姆系列電阻器來保護他們，並能輕易的驅動 LED。可應用於 RGB 或 RGBA LED 驅動器、LED 狀態信息、LED 顯示器、LCD 背光、手機或手持設備的鍵盤背光、伺服馬達控制器。

此專題用來控制寵物四肢共 8 顆馬達。

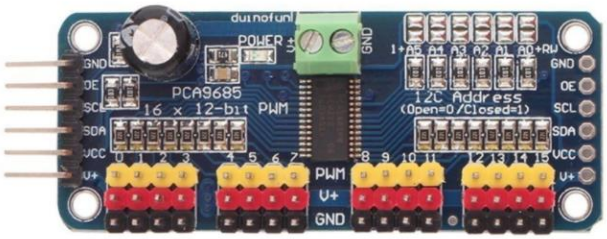

圖 19 PCA9685 正面

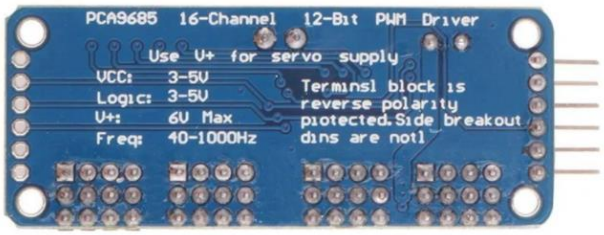

圖 20 PCA9685 反面

3-1-4　MG996R

外觀如圖 21 所示，MG996R 為 90 度馬達， 扭力最大為 13KG，脈寬信號 500-1500-2500，對應的角度是 0 度~±90 度，在 4.8V 時反應速度為 0.17sec， 6V 時為 0.14sec，工作電壓在 4.8V~7.2V，工作死區為 5us，紅色是 5V 電源正極，棕色是共地線，橙色是控制訊號線。我們使用伺服馬達來實現電子寵物的肢體活動。

圖 21 MG996R

註: 死區（dead band）有時也稱為中性區（neutral zone）或不作用區，是指控制系統的傳遞函數中，對應輸出為零的輸入信號範圍。齒輪中的背隙就是一種死區。當齒輪咬合恰好在背隙時，不論輸入軸正轉或是反轉，輸出軸都不會動作。等咬合不在背隙時，輸出軸才會隨著輸入軸而動作。原理如圖 14

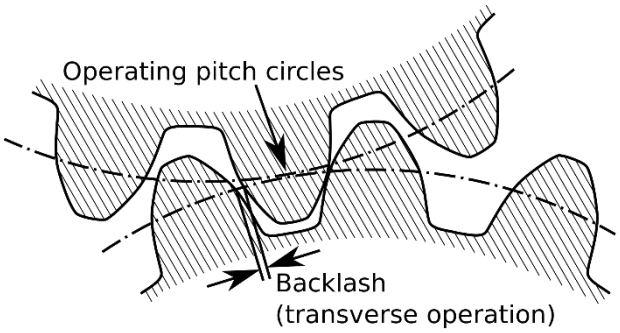

圖 22 死區

3-1-5 DFPlayer mini

接腳如圖 15，外觀如圖 16 所示，DFPlayer mini 是一款小巧且價格低廉的 MP3 模組可以直接接擴音器， 24 位元 DAC 輸出，動態範圍支援 90dB，訊號雜訊比支持 85dB，有多種控制模式可選，例如：IO 控制模式、單工模式、AD 按鍵控制模式。音訊資料按資料夾排序，最多支援 100 個資料夾，每個資料夾可以分配 255 首曲目，擁有 30 個階段的音量大小調整，6 種 EQ 可以調 (EQ 就是調節頻率的放大或衰減)。DFPlayer mini 是用來重現狗叫聲。

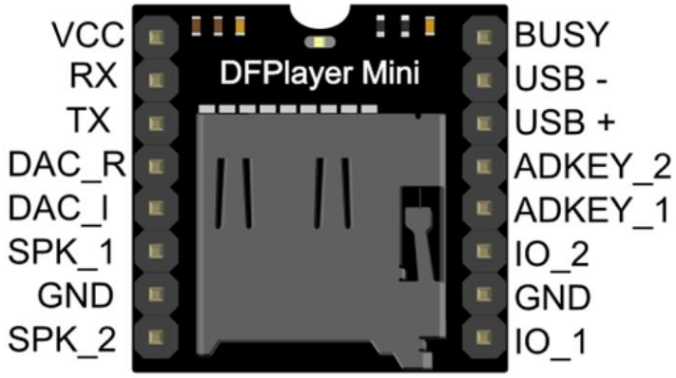

圖 23 DFR0299 mini 接腳圖

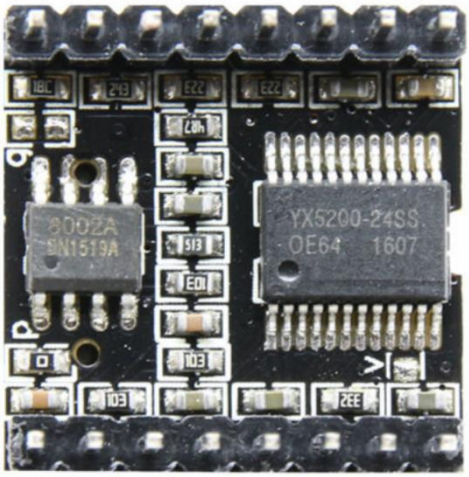

圖 24 DFR0299 mini 反面

3-2 軟體架構

3-2-1　軟體程式

```
if(laina == 1) {      //讀取萊納撥放mp3
   pwm.setPWM(9, 0, angleToPulse(90));          //右後大腿
   pwm.setPWM(11, 0, angleToPulse(90));         //左前大腿
   pwm.setPWM(13, 0, angleToPulse(80));         //右前大腿
   pwm.setPWM(15, 0, angleToPulse(90));         //左後大腿

   pwm.setPWM(1, 0, angleToPulse(85));         //左前小腿
   pwm.setPWM(3, 0, angleToPulse(90));          //右前小腿
   pwm.setPWM(5, 0, angleToPulse(90));          //右後小腿
   pwm.setPWM(7, 0, angleToPulse(90));          //左後小腿
                    softwareSerial.begin(9600);
                    if(player.begin(softwareSerial)){
                      player.volume(30);                //控制音量0~30
                      player.play(2);                   //選擇播放曲目

                    }
                    }

//--------------------------------------------------------------------------------
```

圖 25 萊納程式

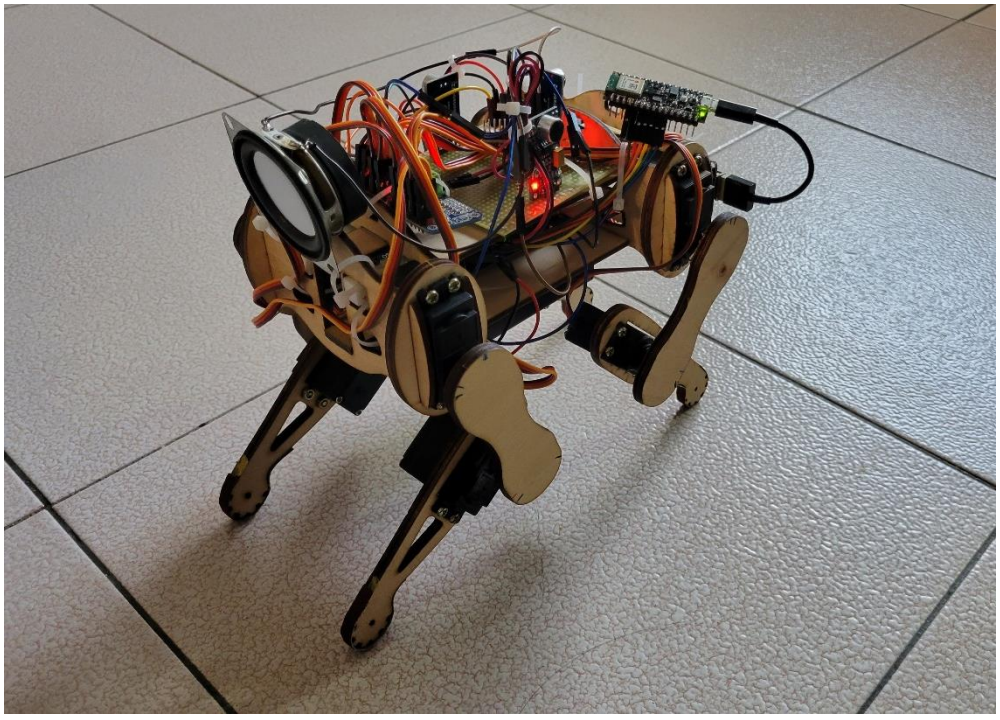

圖 26 實際姿勢(萊納)

```
    if(kneel == 1){    //趴下

                    movementA();
                    delay(100);

                    movementB();
                    delay(100);

                    movementC();
                    delay(100);
    }
```

圖 27 趴下程式

```
//-----------------------------------------------------------------趴下一
void movementA(){
      pwm.setPWM(9, 0, angleToPulse(90));          //右後大腿
      pwm.setPWM(11, 0, angleToPulse(90));         //左前大腿
      pwm.setPWM(13, 0, angleToPulse(80));         //右前大腿
      pwm.setPWM(15, 0, angleToPulse(90));         //左後大腿

      pwm.setPWM(1, 0, angleToPulse(85));         //左前小腿
      pwm.setPWM(3, 0, angleToPulse(90));          //右前小腿
      pwm.setPWM(5, 0, angleToPulse(90));          //右後小腿
      pwm.setPWM(7, 0, angleToPulse(90));          //左後小腿
      delay(650);
}
//-----------------------------------------------------------------趴下二
void movementB(){
      pwm.setPWM(9, 0, angleToPulse(90));          //右後大腿
      pwm.setPWM(11, 0, angleToPulse(90));         //左前大腿
      pwm.setPWM(13, 0, angleToPulse(90));         //右前大腿
      pwm.setPWM(15, 0, angleToPulse(90));         //左後大腿

      pwm.setPWM(1, 0, angleToPulse(105));         //左前小腿
      pwm.setPWM(3, 0, angleToPulse(70));          //右前小腿
      pwm.setPWM(5, 0, angleToPulse(110));         //右後小腿
      pwm.setPWM(7, 0, angleToPulse(70));          //左後小腿
      delay(650);
}
//-----------------------------------------------------------------趴下三
void movementC(){
      pwm.setPWM(9, 0, angleToPulse(90));          //右後大腿
      pwm.setPWM(11, 0, angleToPulse(90));         //左前大腿
      pwm.setPWM(13, 0, angleToPulse(90));         //右前大腿
      pwm.setPWM(15, 0, angleToPulse(90));         //左後大腿

      pwm.setPWM(1, 0, angleToPulse(140));         //左前小腿
      pwm.setPWM(3, 0, angleToPulse(35));          //右前小腿
      pwm.setPWM(5, 0, angleToPulse(100));         //右後小腿
      pwm.setPWM(7, 0, angleToPulse(80));          //左後小腿
}
```

圖 28 趴下副程式

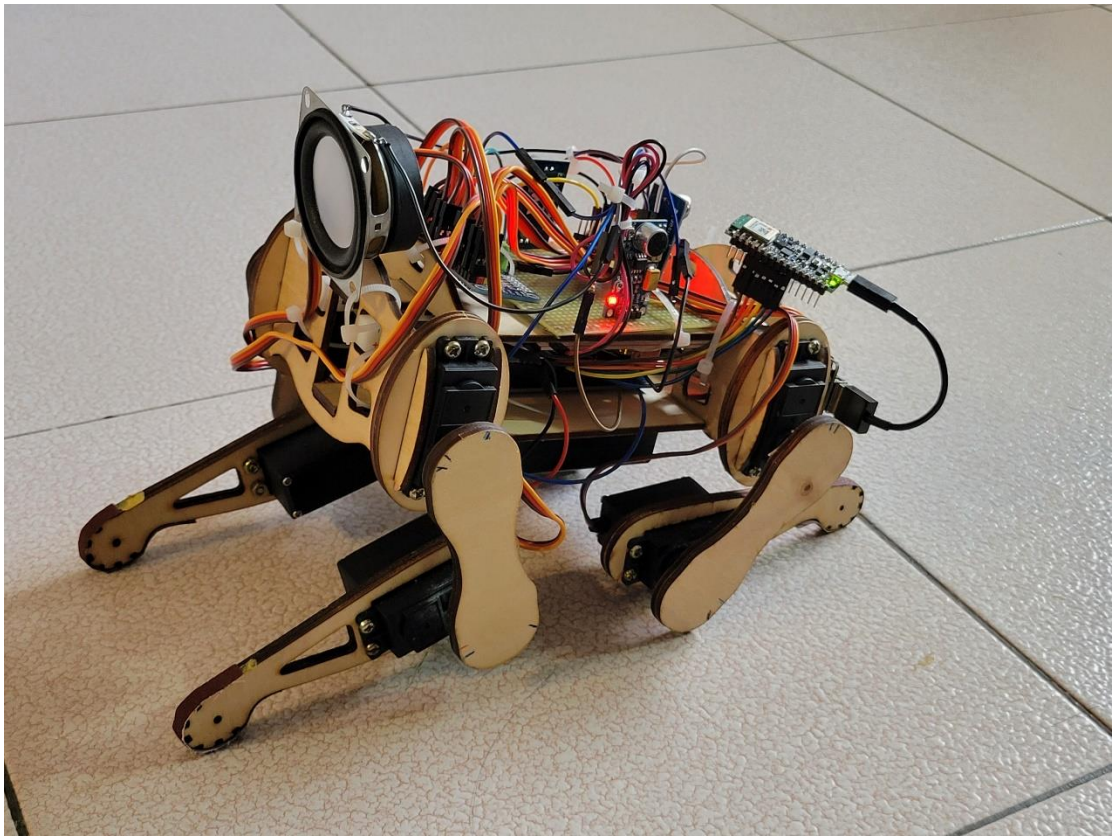

圖 29 趴下實際姿勢

```
if(here == 1){
  if(e==1){
     righting();
  }
  else if(s==1){
    sourth();
  }
  else if(w==1){
    lefting();
  }
}
```

圖 30 這裡程式

```
void  left1(){                                          //左轉1
    pwm.setPWM(9, 0, angleToPulse(angle));              //右後大腿
    pwm.setPWM(11, 0, angleToPulse(angle));             //左前大腿

    pwm.setPWM(15, 0, angleToPulse(angle));             //左後大腿
    pwm.setPWM(13, 0, angleToPulse(80));              //右前大腿

    delay(10);

    pwm.setPWM(1, 0, angleToPulse(85));               //左前小腿
    pwm.setPWM(3, 0, angleToPulse(90));             //右前小腿

    pwm.setPWM(5, 0, angleToPulse(angle));              //右後小腿
    pwm.setPWM(7, 0, angleToPulse(angle));          //左後小腿
    delay(1000);
}
//-------------------------------------------------------------------------------
void  left2(){                                        //左轉2
    pwm.setPWM(9, 0, angleToPulse(50));               //右後大腿
    pwm.setPWM(11, 0, angleToPulse(55));            //左前大腿

    pwm.setPWM(15, 0, angleToPulse(55));              //左後大腿
    pwm.setPWM(13, 0, angleToPulse(80));            //右前大腿

    delay(10);

    pwm.setPWM(1, 0, angleToPulse(45));               //左前小腿
    pwm.setPWM(3, 0, angleToPulse(100));            //右前小腿

    pwm.setPWM(5, 0, angleToPulse(55));             //右後小腿
    pwm.setPWM(7, 0, angleToPulse(82));
    delay(1000);
}
```

圖 31 這裡左轉副程式

```
void  right1(){                                          //右轉1
    pwm.setPWM(9, 0, angleToPulse(angle));              //右後大腿
    pwm.setPWM(11, 0, angleToPulse(angle));            //左前大腿

    pwm.setPWM(15, 0, angleToPulse(angle));             //左後大腿
    pwm.setPWM(13, 0, angleToPulse(80));              //右前大腿

    delay(10);

    pwm.setPWM(1, 0, angleToPulse(85));               //左前小腿
    pwm.setPWM(3, 0, angleToPulse(90));              //右前小腿

    pwm.setPWM(5, 0, angleToPulse(angle));              //右後小腿
    pwm.setPWM(7, 0, angleToPulse(angle));           //左後小腿
    delay(1000);
}
//---------------------------------------------------------------------
void  right2(){                                          //右轉2

    pwm.setPWM(9, 0, angleToPulse(125));              //右後大腿
    pwm.setPWM(11, 0, angleToPulse(angle));            //左前大腿
    pwm.setPWM(13, 0, angleToPulse(115));             //右前大腿
    pwm.setPWM(3, 0, angleToPulse(112));              //右前小腿
    delay(100);
    pwm.setPWM(1, 0, angleToPulse(85));               //左前小腿
    pwm.setPWM(5, 0, angleToPulse(110));              //右後小腿
    pwm.setPWM(7, 0, angleToPulse(110));
    pwm.setPWM(15, 0, angleToPulse(120));              //左後大腿

    delay(1000);
}
```

圖 32 這裡右轉副程式

```
if(sit == 1){    //坐下

                pwm.setPWM(9, 0, angleToPulse(90));        //右後大腿
                pwm.setPWM(11, 0, angleToPulse(90));       //左前大腿
                pwm.setPWM(13, 0, angleToPulse(90));       //右前大腿
                pwm.setPWM(15, 0, angleToPulse(90));       //左後大腿
                delay(10);
                pwm.setPWM(1, 0, angleToPulse(160));       //左前小腿
                pwm.setPWM(3, 0, angleToPulse(30));        //右前小腿
                pwm.setPWM(5, 0, angleToPulse(60));        //右後小腿
                pwm.setPWM(7, 0, angleToPulse(120));       //右前小腿
    }
```

圖 33 坐下程式

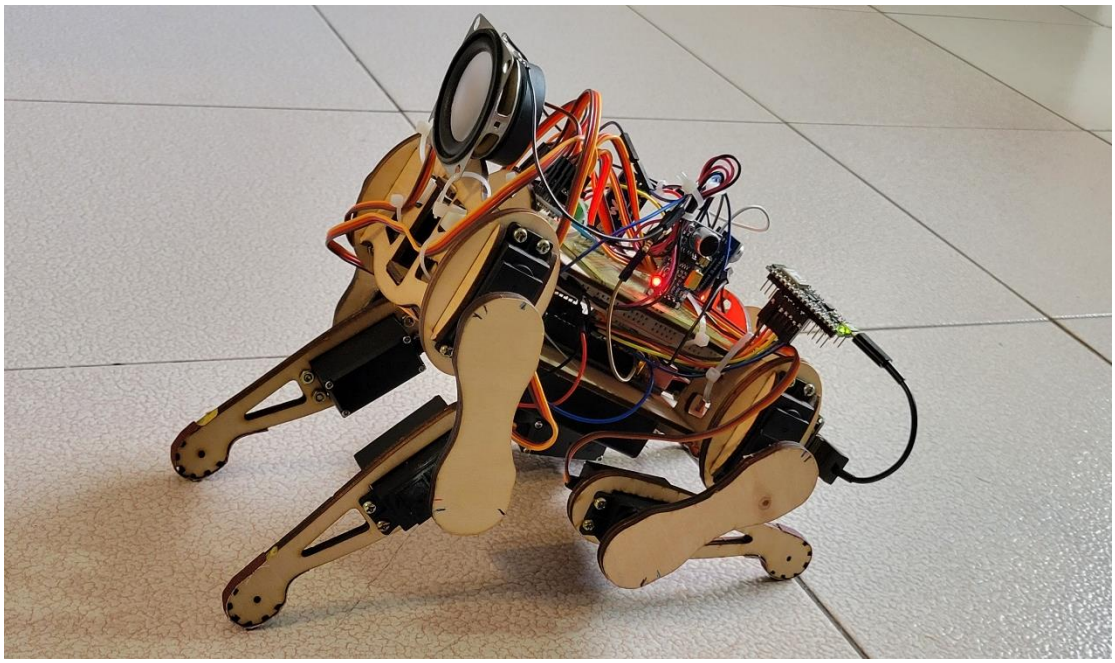

圖 34 坐下實際姿勢

```
if(come == 1 || x == 1){  //當come為高態時進行前進動作

        action0();
        delay(100);

        action1();
        delay(100);

        action2();
        delay(100);

        action0();
        delay(100);

        action3();
        delay(100);

        action4();
        delay(100);

        action5();
        delay(100);

        action6();
        delay(100);

        if(cut == 1){
          pwm.setPWM(9, 0, angleToPulse(90));          //右後大腿
          pwm.setPWM(11, 0, angleToPulse(90));         //左前大腿
          pwm.setPWM(13, 0, angleToPulse(80));         //右前大腿
          pwm.setPWM(15, 0, angleToPulse(90));         //左後大腿

          pwm.setPWM(1, 0, angleToPulse(85));         //左前小腿
          pwm.setPWM(3, 0, angleToPulse(90));          //右前小腿
          pwm.setPWM(5, 0, angleToPulse(90));          //右後小腿
          pwm.setPWM(7, 0, angleToPulse(90));          //左後小腿

          come = 0;
          x = 0;
        }

         else{
          x = 1;
         }
}
```

圖 35 過來的走路程式(1)

```
//------------------------------------------------------------走路零
void action0(){


      pwm.setPWM(9, 0, angleToPulse(90));          //右後大腿
      pwm.setPWM(11, 0, angleToPulse(90));         //左前大腿
      pwm.setPWM(13, 0, angleToPulse(80));         //右前大腿
      pwm.setPWM(15, 0, angleToPulse(90));         //左後大腿

      pwm.setPWM(1, 0, angleToPulse(85));         //左前小腿
      pwm.setPWM(3, 0, angleToPulse(90));          //右前小腿
      pwm.setPWM(5, 0, angleToPulse(90));          //右後小腿
      pwm.setPWM(7, 0, angleToPulse(90));          //左後小腿
      delay(750);
}
//------------------------------------------------------------走路一
void action1(){


      pwm.setPWM(9, 0, angleToPulse(70));          //右後大腿
  pwm.setPWM(11, 0, angleToPulse(90));         //左前大腿
  pwm.setPWM(13, 0, angleToPulse(80));         //右前大腿
  pwm.setPWM(15, 0, angleToPulse(90));         //左後大腿

  pwm.setPWM(1, 0, angleToPulse(85));         //左前小腿
  pwm.setPWM(3, 0, angleToPulse(90));          //右前小腿
  pwm.setPWM(5, 0, angleToPulse(75));          //右後小腿
  pwm.setPWM(7, 0, angleToPulse(90));          //左後小腿
    delay(650);
}
//------------------------------------------------------------走路二
void action2(){


     pwm.setPWM(9, 0, angleToPulse(70));          //右後大腿  後轉-20
  pwm.setPWM(11, 0, angleToPulse(90));         //左前大腿  後轉+30
  pwm.setPWM(13, 0, angleToPulse(100));         //右前大腿  前轉+40
  pwm.setPWM(15, 0, angleToPulse(90));         //左後大腿  前轉-30

  pwm.setPWM(1, 0, angleToPulse(95));         //左前小腿 上轉+20
  pwm.setPWM(3, 0, angleToPulse(92));         //右前小腿 上轉+15
  pwm.setPWM(5, 0, angleToPulse(75));          //右後小腿 下轉-15
  pwm.setPWM(7, 0, angleToPulse(90));          //左後小腿 下轉-15
    delay(650);
}
```

圖 36 過來的走路程式(2)

```
//------------------------------------------------------------走路三
void action3(){


     pwm.setPWM(9, 0, angleToPulse(90));          //右後大腿
  pwm.setPWM(11, 0, angleToPulse(110));
  pwm.setPWM(13, 0, angleToPulse(80));         //右前大腿
  pwm.setPWM(15, 0, angleToPulse(70));         //左後大腿  前轉-30

  pwm.setPWM(1, 0, angleToPulse(85));         //左前小腿
  pwm.setPWM(3, 0, angleToPulse(107));          //右前小腿
  pwm.setPWM(5, 0, angleToPulse(90));          //右後小腿
  pwm.setPWM(7, 0, angleToPulse(75));          //左後小腿
  delay(650);
}
//------------------------------------------------------------走路四
void action4(){

    pwm.setPWM(9, 0, angleToPulse(90));          //右後大腿
  pwm.setPWM(11, 0, angleToPulse(90));         //左前大腿
  pwm.setPWM(13, 0, angleToPulse(80));         //右前大腿
  pwm.setPWM(15, 0, angleToPulse(120));         //左後大腿

  pwm.setPWM(1, 0, angleToPulse(85));         //左前小腿
  pwm.setPWM(3, 0, angleToPulse(90));          //右前小腿
  pwm.setPWM(5, 0, angleToPulse(90));          //右後小腿
  pwm.setPWM(7, 0, angleToPulse(115));          //左後小腿
    delay(650);
}
//------------------------------------------------------------走路五
void action5(){
  pwm.setPWM(9, 0, angleToPulse(90));          //右後大腿
  pwm.setPWM(11, 0, angleToPulse(70));         //左前大腿
  pwm.setPWM(13, 0, angleToPulse(80));         //右前大腿
  pwm.setPWM(15, 0, angleToPulse(120));         //左後大腿

  pwm.setPWM(1, 0, angleToPulse(85));         //左前小腿
  pwm.setPWM(3, 0, angleToPulse(77));          //右前小腿
  pwm.setPWM(5, 0, angleToPulse(90));          //右後小腿
  pwm.setPWM(7, 0, angleToPulse(115));          //左後小腿
  delay(650);
}
```

圖 37 過來的走路程式(3)

```
//------------------------------------------------------------------走路六
void action6(){
  pwm.setPWM(9, 0, angleToPulse(110));          //右後大腿
  pwm.setPWM(11, 0, angleToPulse(90));         //左前大腿
  pwm.setPWM(13, 0, angleToPulse(60));         //右前大腿
  pwm.setPWM(15, 0, angleToPulse(90));         //左後大腿

  pwm.setPWM(1, 0, angleToPulse(70));         //左前小腿
  pwm.setPWM(3, 0, angleToPulse(90));          //右前小腿
  pwm.setPWM(5, 0, angleToPulse(100));          //右後小腿
  pwm.setPWM(7, 0, angleToPulse(90));          //左後小腿
    delay(650);
}
```

圖 38 過來的走路程式(4)

第四章 專題成果

4-1 聲音辨識

總共錄製六個指令分別為萊納、過來、這裡、坐下、趴下、停下。

1. 萊納：調整為站立姿勢，並讓喇叭放出狗叫聲。
2. 過來：寵物進行向前行走動作，並在聽取到”停下”指令後回復至站姿。
3. 這裡：配合方位辨識板，讀取左、右、後數值。調整寵物面向數值較大的那一方。
4. 坐下：讓寵做出坐下姿勢。
5. 趴下：讓寵做出趴下姿勢。
6. 停下:讀取到停下時,可中斷”過來”指令,使寵物在行走中回到站立姿勢。

4-2 方位辨識

在寵物的左邊、右邊、及後方擺上三塊聲音感測模組，在三方位分別喊出”這裡”的指令，藉由讀取到的音量大小及收取到”這裡”的指令後，讓寵物進行向左轉或是向右轉。

第五章 結論與建議

5-1 結論

經過一個學期的努力，從最一開始的語音辨識，一直到最後的馬達控制與整合，一路上經歷了許多困難與波折，不過我們都一一解決了，最後成品也順利地做出來，雖然呈現出來的外觀不是很符合當初想做出來的樣子，但當初想做出來的功能，都有成功的做出來，這也讓我們非常開心，也希望之後有機會可以把電子寵物的外型做得更好看。

5-2 未來展望、建議

在我們對它進行性能實測後，由於電子寵物的腿太過纖細、馬達的扭力過大以及關節處無法黏得很牢固，再加上我們將語音辨識、方向辨識和馬達的控制板等零零總總的東西全部安裝在它的機身上，但機身的空間有限所以我們將控制板往上疊，杜邦線只能圍繞在機身的外圍，導致它在執行動作的時候，腿與腿的關節處很容易就分離了，因此我們希望能藉由減輕電子寵物的機身重量，設計更堅固、空間更大的機身，讓腳和關節處的負擔不必這麼大，也讓電子寵物能夠更佳美觀。為了讓電子寵物能夠達成聽得懂主人的指令也能藉由聽得知道主人的所在位置，使其與一般的寵物沒什麼差別，我們運用語音辨識、方位辨識去實現它，但語音辨識的準確度非常的低，以至於有時要對它喊很多次，它才會理會我並執行指令，方位辨識我們只用了四個邊而且每個都非常的相鄰，造成我們對其中一邊發出聲音，另外的幾個邊也會一同偵測到，故此我們期望能增強語音辨識的程度和增加方位辨識的個數，讓它能夠判斷的比較清楚主人要它做的指令，也能夠比較準確的分析主人聲音的方向來源。

參考文獻

1. 楊仁元、張顯盛、林家德（2021). 專題實作理論與呈現技巧 office2016 版. 台科圖書

2.不知（2019 年 09 月 28 日).基於 MFCC 的語音資料特徵提取概述.擷自 Itread01:

https://www.itread01.com/content/1569654127.html

3.不知（2019 年 02 月 16 日).語音識別-特徵提取.擷取自 Itread01 ：

https://www.itread01.com/content/1550332657.htm

4.Shawn Hymel(2021 年 5 月 27 日).Keyword Spotting with Edge Impulse.擷取自 GitHub:

GitHub - ShawnHymel/ei-keyword-spotting

5. cubie（2017 年 09 月 09 日).模組介紹與自製 LM386 麥克風聲音放大器

.擷自 swf:

聲音檢測／聲音放大器（一）：模組介紹與自製 LM386 麥克風聲音放大器 - 超圖解系列圖書（swf.com.tw）

6.不知（2019 年 02 月 13 日). 6 路 PWM 舵機驅動板(PCA 9685) + Arduino

.擷取自 Itread01 ：

【學習筆記】——16 路 PWM 舵機驅動板(PCA 9685) + Arduino - IT 閱讀(itread01.com)

7.陳明燦（2020 年 11 月).Arduino 實作入門與專題應用.博碩出版社