大安高工電子科

寵物狗

Mecha Dog

指導老師：

張顯盛 老師

組長：

電子三甲　　徐偉翔

組員：

電子三甲　　文浩宇

電子三甲　　王靖越

電子三甲　　許譽騰

摘要

在這科技進步飛快的時代，飼養擁有一隻擁有人工智慧的寵物狗可以說是必要的，然而所需要的功能也相當多元化，而我們是利用 Raspberry pi 3 這塊板子，並且配戴有人臉辨識的攝像頭藉由安裝 open cv 來得到主人辨識的能力，而身為一隻動物，動作變得相當重要，而我們為了達到動作精準的控制，所以我們採用 MG90s 的伺服馬達做為我們的關節，但 Raspberry pi 推動伺服馬達時，電流太大和雜訊太高，導致無法推動多顆伺服馬達且精準，所以我們加上了可以推動 16 顆伺服馬達的 pwm 穩壓控制器，最後利用 3D 模組做成我們寵物狗的機構，加以組裝並變成了我們的寵物狗。

Abstract

In this era of rapid technological advancement, it is necessary to raise a dog with artificial intelligence, but the functions required are quite diverse, and we use the Raspberry pi 3 board equipped with camera that is capable for face recognition. The face recognize camera achieves the effect of master recognition by the installation Open CV, and as an animal, the action becomes quite important, and in order to achieve precise control of motion, we use the servo motor of MG90s as our joint. However, when the Raspberry pi pushes the servo motor, the current is too large and the noise is too high, which makes it impossible to push multiple servo motors and is accurate. Therefore, we added a pwm regulator controller that can push 16 servo motors, and finally use 3D modules to do The institution that became our pet dog was assembled and turned into our pet dog.

目錄

表目錄

圖目錄

第一章 緒論

1-1 背景及目的

在這科技進度飛快的時代，科技所帶來的方便性超越以往的想像，科技與日常生活結合，所帶來的便捷和實用性，超越從前粗糙又繁瑣的手工製作，以至於許多的傳統被新穎所取代。

俗話說：「狗是人類最要好的朋友」，然而現今科技日新月異，擁有一隻能陪伴你的機器狗不再是天方夜譚，然而為何不養一隻真正的狗呢？因為常常早早出門，卻很晚回家，早就已經精疲力盡，而無法給牠最好的照顧和陪伴，所以有如此的啟發。

1-2 預期成果

希望能夠在資金、設備和時間充足的情況下，做出一隻行動自如，甚至能夠跑的狗，在清楚辨識出主人時，能夠給予熱情的回應，像是向前撒嬌和搖尾巴等等，並且在遇到不認識的人時，能夠吠叫和豎起尾巴等等，去增加更多更相似於狗的行為模式。

第二章 理論探討

2-1 Raspberry Pi 3

Raspberry pi 3 是一款以 Linux 為架構的單晶片電腦。它由英國的樹莓派基金會所開發，目的是以低價的硬體和自由軟體來促進學校的基本電腦科學教育。

可以用這塊面積大小差不多一張一元美金的板子來進行網頁瀏覽、影音播放、文書處理……等而最重要的功能是：可以撰寫 arduino、python……等程式，並透過板子上的 GPIO 接腳進行 INPUT/OUTPUT 也是我們控制伺服馬達轉動的訊號源，其外觀與 GPIO 接腳如圖 1

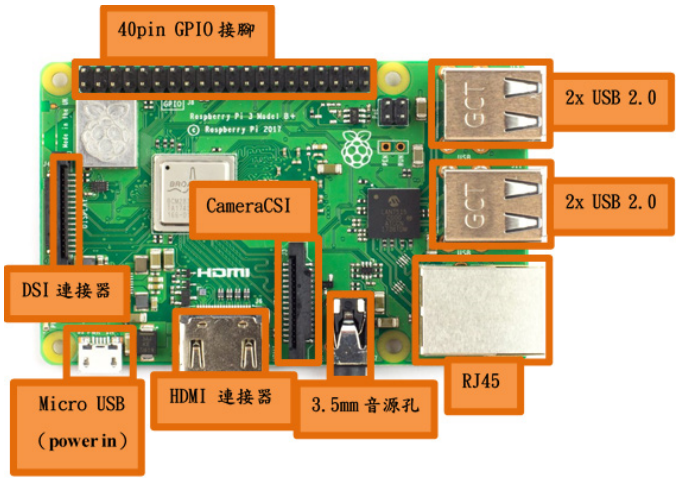


圖 1 Raspberry Pi 3

2-2 Python

Python 是一款被廣泛使用高階的程式語言，它擁有著以下的特點:

1. 物件導向:類似 C++同樣擁有物件導向。
2. 直譯式:執行時為一行一行執行，無須在執行前經過編譯等動作。
3. 跨平台:同樣的一份 code 可以在多種不同的平台執行。

圖 2 python

2-3 Open CV

Open CV 用 C++語言編寫，是一個影像處理函式庫。在影像處理方面應用廣泛，可以讀取儲存圖片、視訊、矩陣運算、統計、影像處理等，我們將會使用它來進行我們的人臉辨識。

圖 3 OpenCV

2-4 深度神經網路

深度神經網路

深度神經網路（Deep Neural Network ,DNN）是深度學習（Deep Learning）的一種，是我們採用的人臉辨識方法，它使用的是反向傳播演算法，是一種與最優化方法結合使用的，對網絡中所有權重計算損失函數的梯度。其中我們所使用的損失函數為三重損失函數（Triplet Loss Function）其示意圖如圖 4。

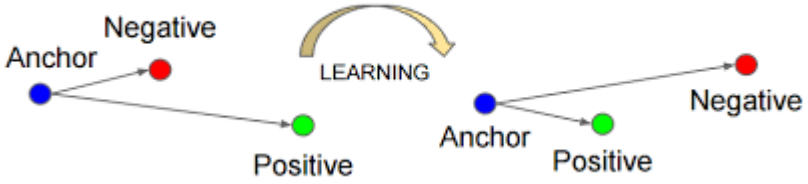


圖 4 三重損失函數概念

三重損失函數是指將資料分批輸入（Batch），一次其中包括一個母體（Anchor）、一個正子樣本（Positive）以及一個負子樣本（Negative）母體與正子樣本為同樣身分的資料，而負子樣本則為不同身分的資料，進行訓練的時候我們會希望將正子樣本盡可能的拉近母體，負子樣本則相反，實際上的圖如圖 5。

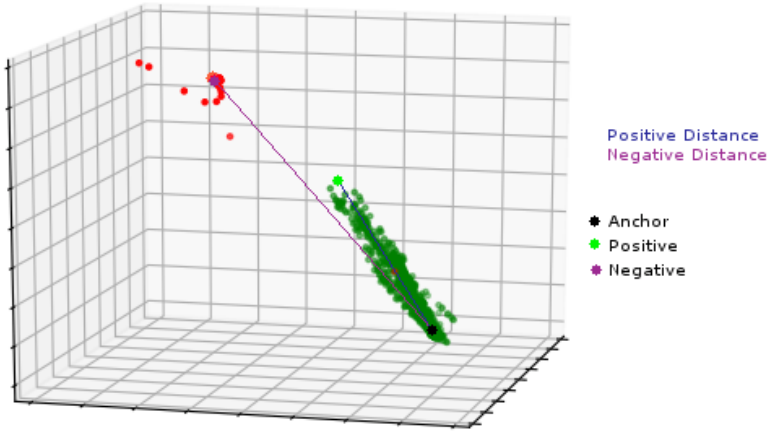


圖 5 三重損失實測結果

2-5 人臉辨識理論

所謂的人臉辨識是指透過人臉來解釋其身分，而實際上在進行人臉辨識之前，會需要先採集人臉資料，而辨識的過程則需要先定位臉部、處理圖像、身份確立等步驟才能辨識出其身份。

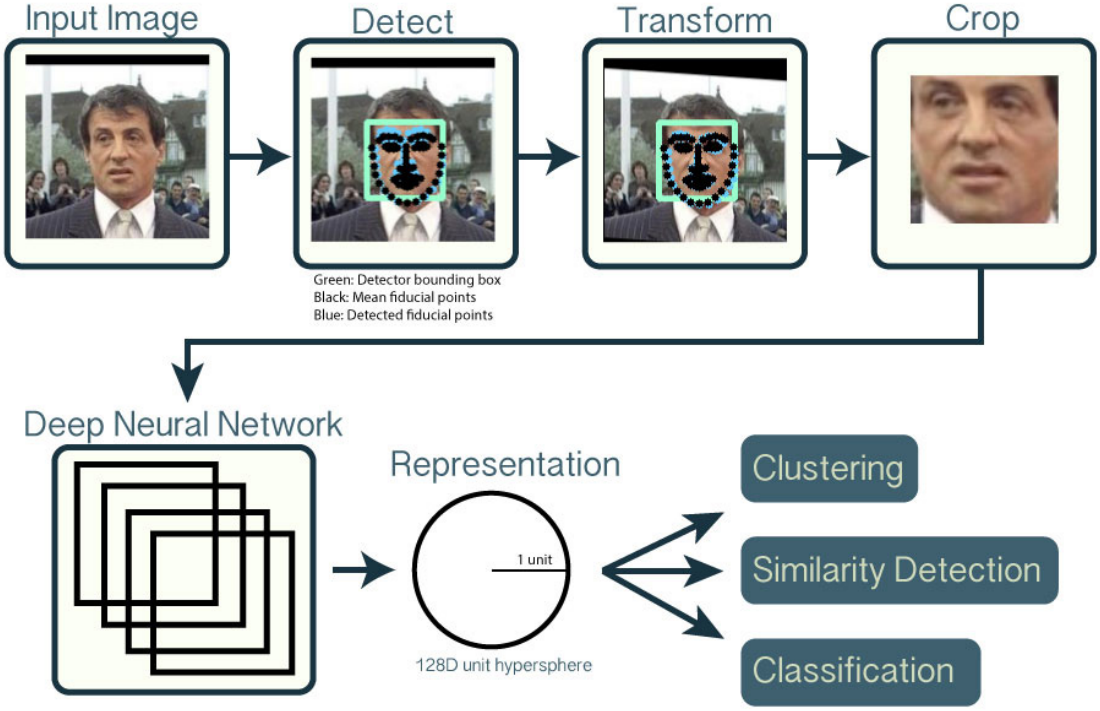


圖 6 人臉辨識流程

收拾（Crop）、深度神經網路（Deep Neural Network）、表示（Representation）、群集分析（Clustering）、相似偵測（Similarity Detection）、分類（Classification）流程如圖所示我們要先取得照片並尋找其中的人臉，並將其變形使其眉毛與下巴對準至固定位置以便與資料庫比較，再將其取下後交由深度神經網路將其轉換成 128 維單位向量圖，進行族群尋找以及相似偵測和分類並定位身份

2-6 伺服馬達及其控制原理

我們所選用的伺服馬達為 MG90S 其工作原理如下:

伺服馬達分為交流（AC）和直流（DC）兩種，直流伺服馬達機體較細長，因此轉子慣性較小，而且具有線性反應佳與簡單易於控制特性，因為直流伺服馬達因為操作容易，也就是旋轉方向由電流決定，並且旋轉速度由改變施加的電壓來控制，控制簡單所以廣泛使用因此現在直流伺服馬達是使用最多的馬達，MG90S 是使用直流馬達，我們控制馬達的方法為下面介紹的脈波寬度調變。

脈波寬度調變

是將類比訊號轉換為脈波的一種技術，一般轉換後脈波的週期固定，但脈波的工作週期會依類比訊號的大小而改變。

在類比電路中，類比訊號的值可以連續進行變化，在時間和值的幅度上都幾乎沒有限制，基本上可以取任何實數值，輸入與輸出也呈線性變化。所以在類比電路中，電壓和電流可直接用來進行控制對象。

數位電路其輸出只可能為 ON 和 OFF 兩種狀態，所以電壓或電流會通/斷方式的重複脈波序列加載到類比負載。PWM 技術是一種對類比訊號電位的數位編碼方法，通過使用高解析度計數器調變方波的占空比，從而實現對一個類比訊號的電位進行編碼。

2-7 3D 印表機

3D 列印，屬於快速成形技術的一種，它是一種數位模型檔案為基礎，透過逐層堆疊累積的方式來構造物體的技術簡單說就是物品透過材料被一層層的累積列印出來，該技術也被稱為累積製造(Additive manufacturing)。

3D 列印的成品擁有以下特點:無限制、幾何結構、裸空、高精密度、滿足客製化需求、材質多樣化、製作速度快、材質多樣化以及少量生產較便宜。

3D 列印漸漸的被使用在航空、建築、汽車、國防、牙科、醫療等領域，但同時也帶來管理上的問題，像是印製槍枝、子彈合不合法等等，都需要政府加以

規範。而我們所使用的 3D 印表機為 SKY-MAKER R2，使用時我們會先將檔案放入 SD 卡，開機後便可以從螢幕上選取我們要印製的檔案，開始印製。

圖 7 SKY-MAKER R2

第三章 專題準備

3-1 系統方塊

本專題會先由 camera 拍攝畫面後，再交給 raspberry pi 3 進行人臉辨識，如果辨識結果正確，執行馬達控制的程式，讓狗狗動作。

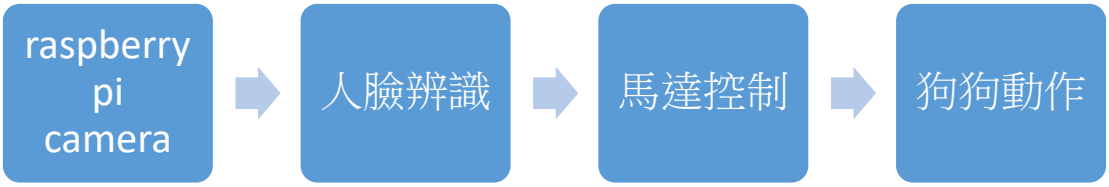
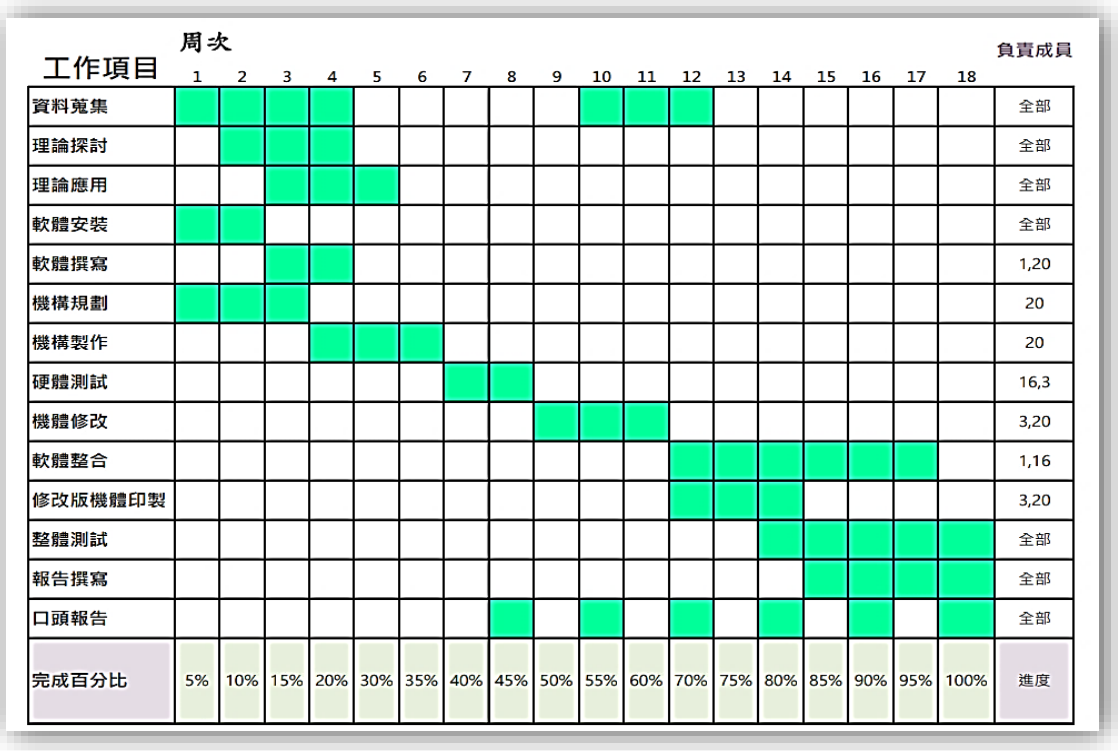


圖 8 系統架構圖

3-2 甘特圖

本專題在分工的部分，前期與後期大部分都是一起討論、完成，只有中間硬體與軟體有些微的分工。



工作項目	周次 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	負責成員
資料蒐集																			全部
理論探討																			全部
理論應用																			全部
軟體安裝																			全部
軟體撰寫																			1,20
機構規劃																			20
機構製作																			20
硬體測試																			16,3
機體修改																			3,20
軟體整合																			1,16
修改版機體印製																			3,20
整體測試																			全部
報告撰寫																			全部
口頭報告																			全部
完成百分比	5%	10%	15%	20%	30%	35%	40%	45%	50%	55%	60%	70%	75%	80%	85%	90%	95%	100%	進度

圖 9 專題甘特圖

3-3 流程圖

本專題步驟會預先蒐集可能會用到的資料，最後我們分為三個大步驟進行，三個大方向都完成後再統整在一起，完成此專題。

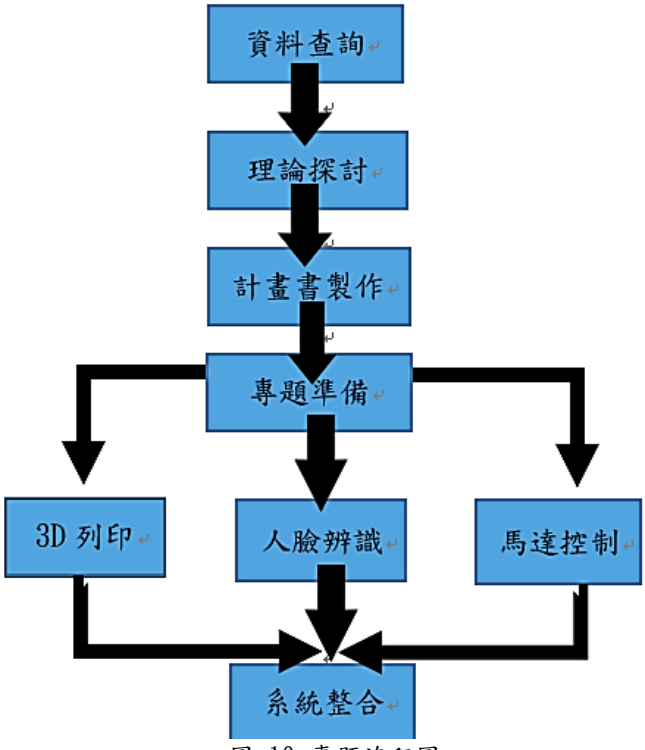


圖 10 專題流程圖

3-4 硬體設計

1. 利用 Sketchup 設計狗狗模型(設計完後把狗狗分解成模組並設計卡榫)

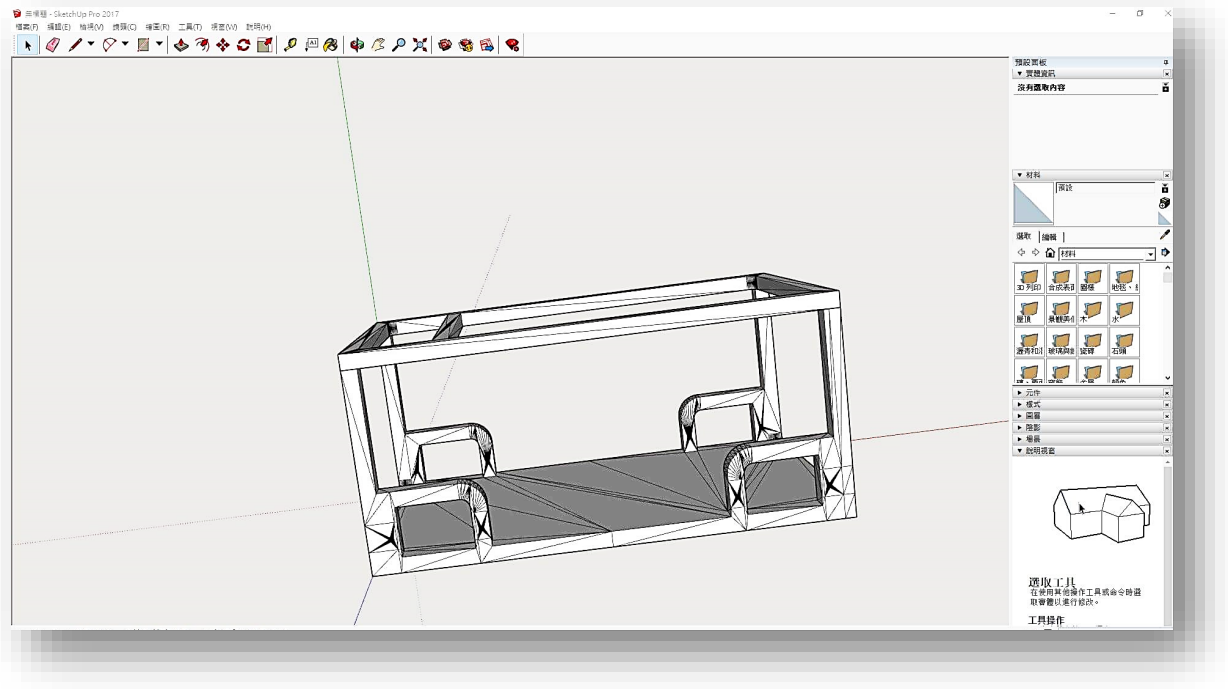


圖 11 Sketchup

2. 設計完成後，使用 MakerBot 設定列印密度和底板等設定，並輸出檔案

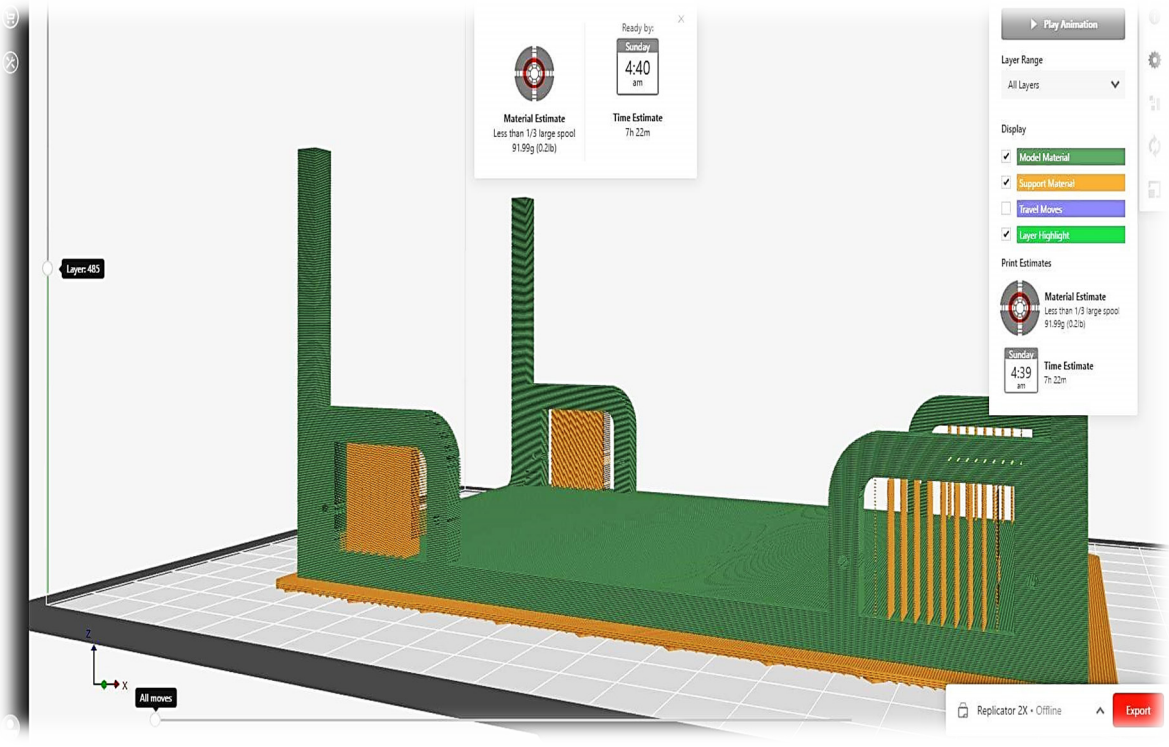


圖 12 MakerBot 設定

3. 使用 3D 印表機進行列印(一次列印一個模組)

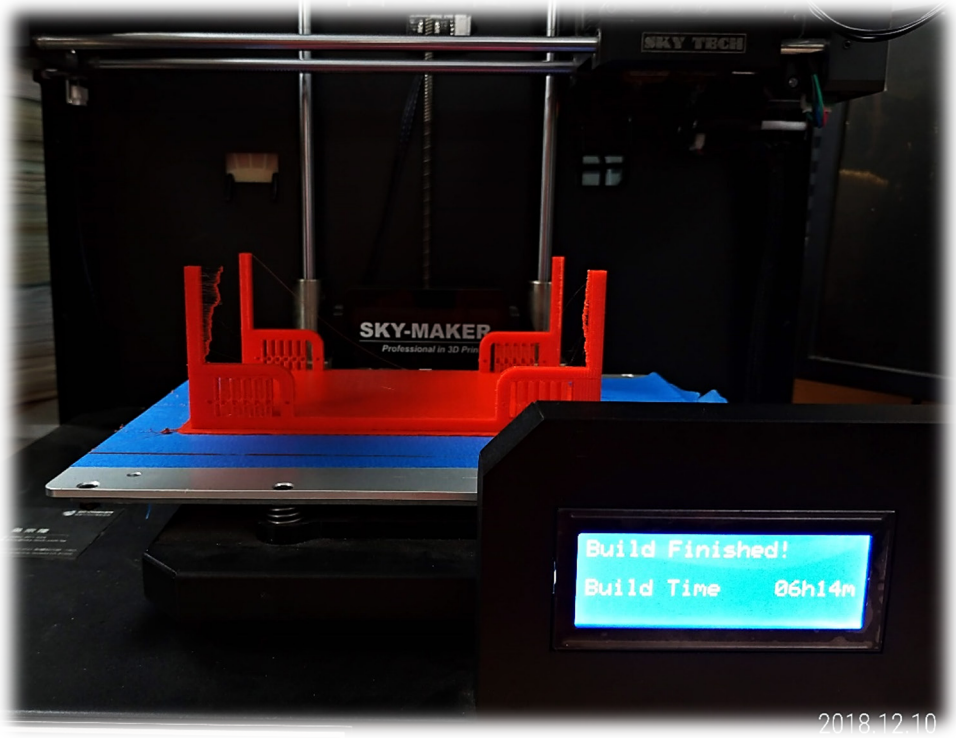


圖 13 3D 印表機列印完成

4. 將列印完的模組進行組裝

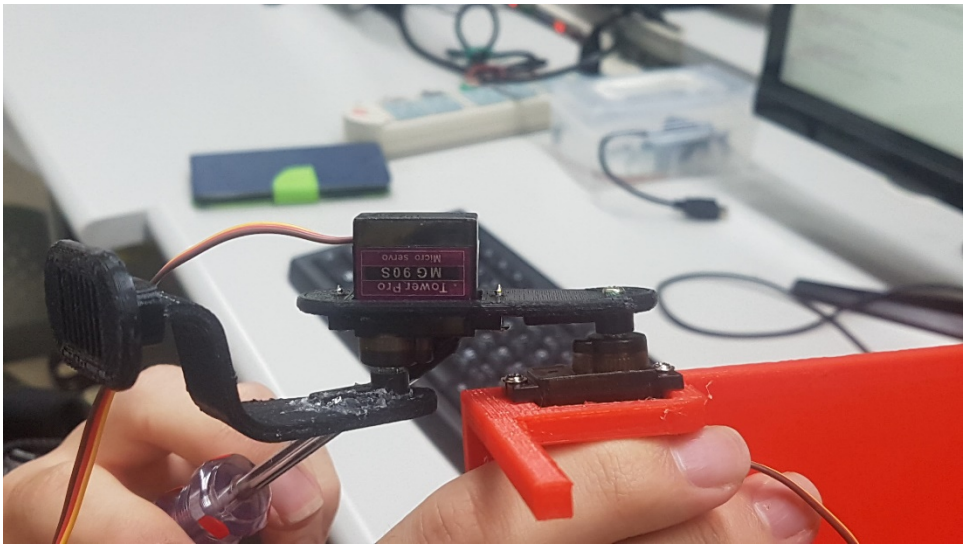


圖 14 組裝 3D 模型

5. 使用強力膠加強固定卡榫或黏接模型

圖 15 強力黏著劑

6. 將電路板和電源放置於機體上，並使用泡棉膠固定(成品)

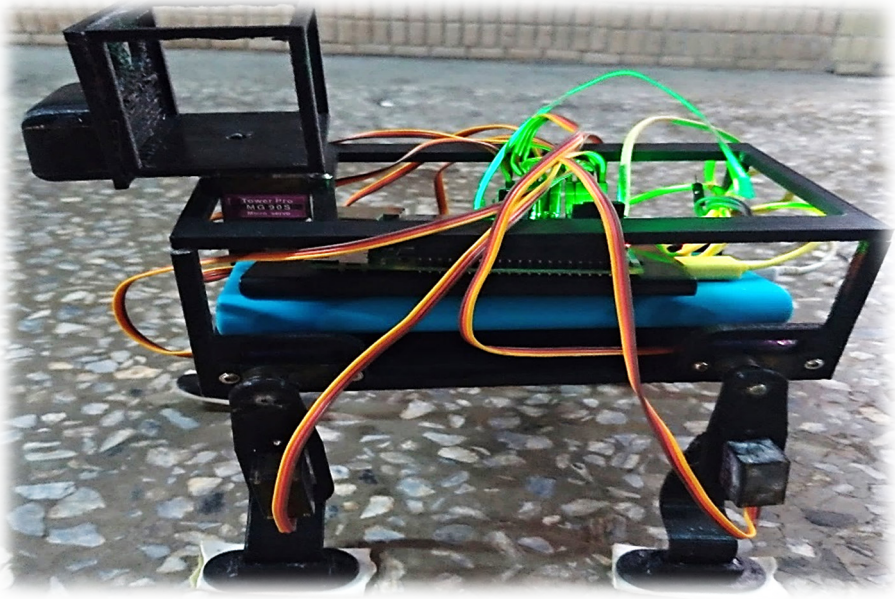


圖 16 狗狗成品

3-5 軟體設計

程式撰寫的部分，是使用 python 進行編輯的，以下介紹人臉辨識的程式。

人臉辨識

```
# USAGE
# python recognize_video.py --detector
face_detection_model \
#   --embedding-model openface_nn4.small2.v1.t7 \
#   --recognizer output/recognizer.pickle \
#   --le output/le.pickle

# import the necessary packages
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import pickle
import time
import cv2
import os

# construct the argument parser and parse the
arguments
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--detector",
required=True,
    help="path to OpenCV's deep learning face
detector")
ap.add_argument("-m", "--embedding-model",
required=True,
    help="path to OpenCV's deep learning face
embedding model")
ap.add_argument("-r", "--recognizer",
required=True,
    help="path to model trained to recognize
faces")
ap.add_argument("-l", "--le", required=True,
    help="path to label encoder")
ap.add_argument("-c", "--confidence", type=float,
default=0.5,
    help="minimum probability to filter weak
detections")
args = vars(ap.parse_args())

# load our serialized face detector from disk
```

使用方法（打在端機中）

呼叫區

外部變數連結

```
print("[INFO] loading face detector...")
protoPath = os.path.sep.join([args["detector"],
"deploy.prototxt"])
modelPath = os.path.sep.join([args["detector"],
    "res10_300x300_ssd_iter_140000.caffemodel"])
detector = cv2.dnn.readNetFromCaffe(protoPath,
modelPath)

# load our serialized face embedding model from
disk
print("[INFO] loading face recognizer...")
embedder =
cv2.dnn.readNetFromTorch(args["embedding_model"])

# load the actual face recognition model along
with the label encoder
recognizer =
pickle.loads(open(args["recognizer"],
"rb").read())
le = pickle.loads(open(args["le"], "rb").read())

# initialize the video stream, then allow the
camera sensor to warm up
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)

# start the FPS throughput estimator
fps = FPS().start()

# loop over frames from the video file stream
while True:
    # grab the frame from the threaded video
stream
    frame = vs.read()

    # resize the frame to have a width of 600
pixels (while
    # maintaining the aspect ratio), and then
grab the image
    # dimensions
    frame = imutils.resize(frame, width=600)
    (h, w) = frame.shape[:2]
```

啟用深度神經網路

啟用鏡頭

讀取鏡頭

調整大小

```
    # construct a blob from the image
    imageBlob = cv2.dnn.blobFromImage(
        cv2.resize(frame, (300, 300)), 1.0, (300,
300),
        (104.0, 177.0, 123.0), swapRB=False,
crop=False)

    # apply OpenCV's deep learning-based face
detector to localize
    # faces in the input image
    detector.setInput(imageBlob)
    detections = detector.forward()

    # loop over the detections
    for i in range(0, detections.shape[2]):
        # extract the confidence (i.e.,
probability) associated with
        # the prediction
        confidence = detections[0, 0, i, 2]

        # filter out weak detections
        if confidence > args["confidence"]:
            # compute the (x, y)-coordinates of
the bounding box for
            # the face
            box = detections[0, 0, i, 3:7] *
np.array([w, h, w, h])
            (startX, startY, endX, endY) =
box.astype("int")

            # extract the face ROI
            face = frame[startY:endY,
startX:endX]
            (fH, fW) = face.shape[:2]

            # ensure the face width and height
are sufficiently large
            if fW < 20 or fH < 20:
                continue

            # construct a blob for the face ROI,
then pass the blob
            # through our face embedding model to
obtain the 128-d
```

尋找人臉

鎖定人臉

進行辨識

確認資料夠不夠大

```
            # quantification of the face
            faceBlob =
cv2.dnn.blobFromImage(face, 1.0 / 255,
                (96, 96), (0, 0, 0), swapRB=True,
crop=False)
            embedder.setInput(faceBlob)
            vec = embedder.forward()

            # perform classification to recognize
the face
            preds =
recognizer.predict_proba(vec)[0]
            j = np.argmax(preds)
            proba = preds[j]
            name = le.classes_[j]

            # draw the bounding box of the face
along with the
            # associated probability
            text = "{}: {:.2f}%".format(name,
proba * 100)
            y = startY - 10 if startY - 10 > 10
else startY + 10
            cv2.rectangle(frame, (startX,
startY), (endX, endY),
                (0, 0, 255), 2)
            cv2.putText(frame, text, (startX, y),
                cv2.FONT_HERSHEY_SIMPLEX, 0.45,
(0, 0, 255), 2)

    # update the FPS counter
    fps.update()

    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the ‘q‘ key was pressed, break from the
loop
    if key == ord("q"):
        break

# stop the timer and display FPS information
fps.stop()
```

辨識身分

標記人臉及其身份

重新抓取照片

輸出照片

外部中斷

```
print("[INFO] elasped time:
{:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS:
{:.2f}".format(fps.fps()))

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

清理和統計

▲表 1 人臉辨識程式碼

第四章 專題成果

4-1 成果

4-1-1 人臉辨識成果

執行完上述的程式後我們便可以得到辨識後的結果，可以辨識出畫面中的人為「wen」並且回傳一個「yes」讓我們知道他發現主人了。

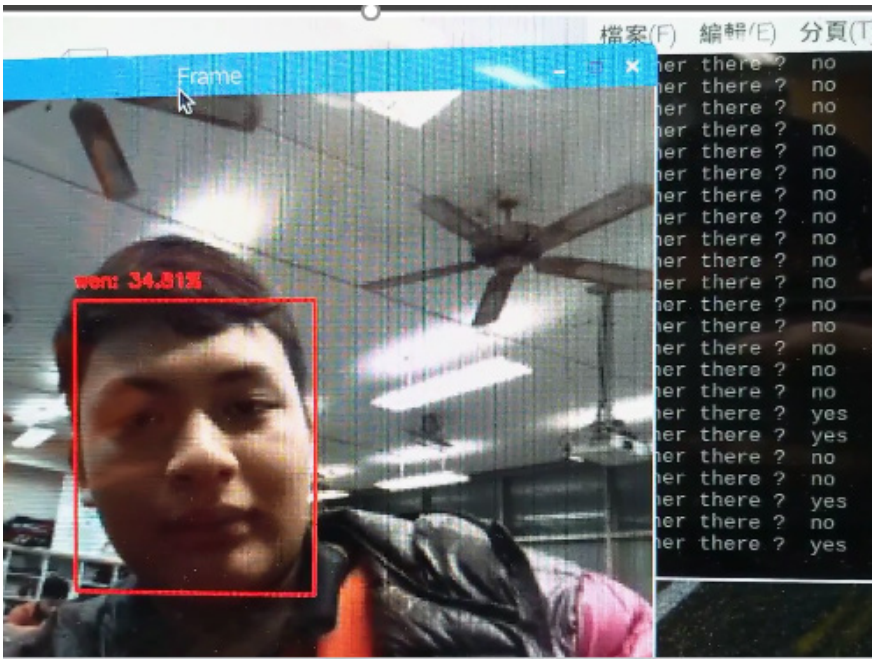


圖 17 人臉辨識成果

4-1-2 馬達控制成果

下方的三張圖片是從狗狗正在移動的影片所截下來的圖。狗狗正一步一步往前走著。

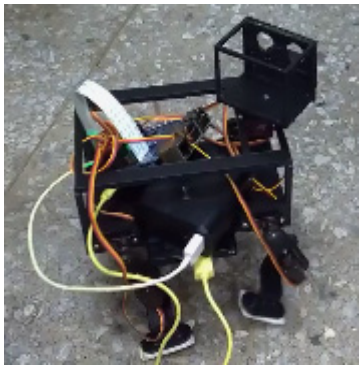


圖 18 移動前

圖 19 移動中

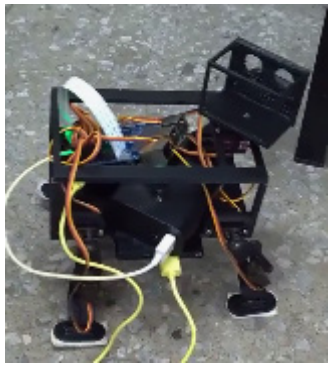


圖 20 移動後

4-2 問題與解決

一、 PWM 不穩

一開始測試多顆伺服馬達時，我們發現樹梅派所提供的 PWM 有不穩定的情形，所以我們購買了由 Adafruit 公司開發的馬達控制板來解決此問題

二、 人臉辨識

進行主人的辨識時，會因為樹梅派本身的效能不足而十分卡頓，也影響到後續整合時無法執行後續程式，再加上我們的資料庫檔案並不完整也影響到了辨識時的準確度。

三、 整合問題

誠如上述所言，樹梅派的效能不足不只影響了人臉辨識，也影響到了最後整合時，因效能不足導致無法同時執行人臉辨識與馬達控制，也讓我們無法將這兩大部份統整完成。

第五章 結論與建議

5-1 結論

這個專題與未來生活息息相關，進入人工智慧的時代，機器狗的發展是突飛猛進的，而我們理想目標是想要製作出走路順暢且可以幫忙主人的寵物狗。

由於技術的限制與經費的不足，我們目前只能做到可以行走、辨識主人和利用互聯網進行操控，但是這三項功能因為微控制器的效能不足，再結合上碰到了困難，因此必須更換效能更好的微控制器，但礙於經費的不足，即使我們讓處理器超頻，仍然無法達到三項功能的結合。.

5-2 建議

當你接觸的此專題，在專題規劃時一定不要想得太簡單，因為理論和硬體都相當複雜，光是寵物狗的機體設計，就需要花費大量的時間，因為當你一個關節設計不好，就會導致你無法組裝，再來你必須更換更好的微電腦，因為人臉辨識套件是可怕的怪獸，會讓你微電腦直接當機或燒毀，因為我們在實驗人臉辨識套件運作，就因工作溫度過高，導致微電腦某顆 IC 燒毀，讓攝相頭無法與微電腦連接，因此我們強烈建議你必須使用更好的微電腦並且增加散熱系統，例如:風扇、水冷系統。

除了硬體與機體要改善外，還可以加入語音辨識套件，讓寵物狗不必再利用其他設備控制，只需要說 HEY DOG 就可利用聲音控制，並可以結合人工智慧，讓寵物狗會自己學習，這樣的寵物狗將不再是寵物狗，而是會懂你會幫你

的智慧寵物狗，最後一定要在機體上裝設許多感測模組，例如在頭上安裝超音波感測器，讓寵物狗走路不再撞牆，最後可以在身體上安裝感測器，讓主人摸寵物狗的身體時，會有相對的反應，讓寵物狗更像真實的狗。

硬體與軟體結合完畢後，最後就是外觀的設計，首先是要把電路用機殼包來包覆，因為沒有包住很容易因為外在環境，導致元件損毀，再來頭型的設計必須要是能好看又能放進鏡頭和超音波感測器，而我們專題試設計方形的，這樣雖然印製方便，但組裝起來真的很不像狗，且只能放進去超音波模組，鏡頭完全沒有地方可以放置，所以我們建議你，製作寵物狗專題，必須要設想周全，如果沒有設想周全，除了會硬體跟軟體無法順利結合和外型醜陋外，還會讓你在製作過程中發生許多你想不到的問題，並很難去解決甚至無法解決，例如:CPU 效能不足。

參考文獻

一、楊仁元、張顯盛、林家德（2014）。專題製作理論與呈現技巧。台北市：台科大圖書股份有限公司

二、Adrian Rosebrock。OpenCV Face Recognition。2018 年 9 月 24 日，取自 http://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/

三、 matsevensen。Ardudog 3D module。2014 年 10 月 19 日，取自 https://www.thingiverse.com/thing:507162

四、Florian Schroff、Dmitry Kalenichenko、James Philbin(2015) FaceNet: A Unified Embedding for Face Recognition and Clustering 取自 https://www.cv- foundation.org/openaccess/content_cvpr_2015/app/1A_089.pdf

附錄

附錄一　設備及材料清單

材料清單

類別名稱	材料名稱	單位	數量	應用說明	備註
硬體	NGEN(Co-Polyester)	捆	1	3D 印表機耗材	無
硬體	2.48mm 螺絲	顆	10	固定用	無
硬體	模型膠	瓶	1	固定底板之用途	無
硬體	發光二極體	顆	10	指示燈	無

▲表 2 材料清單

設備清單

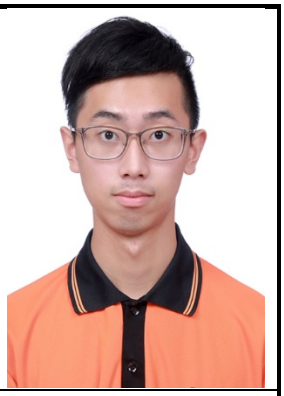
類別	設備、軟體名稱	應用說明
硬體	Raspberry Pi 3	微控制主機板
硬體	Raspberry Pi Camera Module	進行人臉動態辨識
硬體	3D 印表機	印製模型
硬體	HDMI 轉接頭	輸出螢幕
硬體	伺服馬達	驅動模型走動
硬體	外殼裝飾	優化外觀
硬體	PWM 專接器	矯正馬達轉動角度
硬體	行動電源	供應 Raspberry Pi 電源
軟體	OPEN CV	人臉辨識系統
軟體	Python	程式設計
軟體	WORD	編輯報告
軟體	POWERPOINT	製作簡報

▲表 3 設備清單

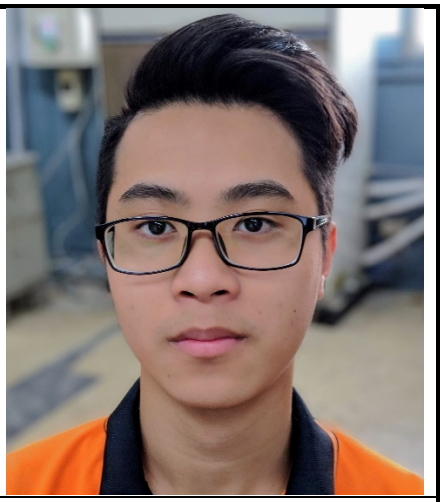
附錄二　成員簡歷

姓名	徐偉翔	班級	子三甲	
曾修習專業科目	基本電學、基本電學實習 數位邏輯、數位邏輯實習 電子學、電子學實習 微處理機、微處理機實習 電腦輔助設計實習 程式設計實習 專題製作實習			
參與專題工作項目	3D 列印 樹莓派人臉辨識程式設計 樹莓派硬體組裝			
經歷簡介	工業電子丙級技術士證照 英文小老師			

▲表 4 成員簡歷-徐偉翔

姓名	許譽騰	班級	子三甲	
曾修習專業科目	基本電學、基本電學實習 數位邏輯、數位邏輯實習 電子學、電子學實習 微處理機、微處理機實習 電腦輔助設計實習 程式設計實習 專題製作實習			
參與專題工作項目	3D 列印 樹莓派程式人臉辨識設計 樹莓派程式馬達驅動設計			
經歷簡介	工業電子丙級技術士證照 數位邏輯小老師			

▲表 5 成員簡歷-許譽騰

姓名	王靖越	班級	子三甲	
曾修習專業科目	基本電學、基本電學實習 數位邏輯、數位邏輯實習 電子學、電子學實習 微處理機、微處理機實習 電腦輔助設計實習 程式設計實習 專題製作實習			
參與專題工作項目	3D 列印 樹莓派硬體組裝 樹莓派人臉辨識程式設計			
經歷簡介	工業電子丙級技術士證照 衛生股長 體育股長			

▲表 6 成員簡歷-王靖越

姓名	文浩宇	班級	子三甲	
曾修習專業科目	基本電學、基本電學實習 數位邏輯、數位邏輯實習 電子學、電子學實習 微處理機、微處理機實習 電腦輔助設計實習 程式設計實習 專題製作實習			
參與專題工作項目	3D 列印 樹莓派程式人臉辨識設計 樹莓派程式馬達驅動設計			
經歷簡介	工業電子丙級技術士證照			

▲表 7 成員簡歷-文浩宇