

投稿類別：工程技術類

篇名：自走車跟車系統實作研究

作者：

林宗翰 國立竹北高中 二年級 08 班 06 號 普通科
王宥傑 國立竹北高中 二年級 01 班 02 號 普通科

指導老師：

陳顥元

壹、前言

一、研究動機

面對電動車近年來的蓬勃發展，各種智慧駕駛的功能如雨後春筍般出現，例如自動跟車、自動駕駛…等，由於我們對於電動車的喜好，加上高一參加中原大學程式語言的先修課程，以及高二 Python 程式語言學習，所以對軟體有一定的基礎能力與興趣。我們參考一些自走車相關的書籍資料，以及 Device Plus 網站(<https://micro.rohm.com/tw/deviceplus/>)，該網站是由全球知名半導體和電子零件製造商 ROHM 所創建，提供各種實用有趣的電子相關資訊。因此我們決定利用樹莓派(Raspberry Pi)搭配影像辨識來做自動跟車系統(賀雪晨等，2021)、(柯博文，2015)，這個實作研究是一個具有挑戰的題目。藉此讓我們透過所學的知識和網路學習，來製作一台自走車跟車系統，並從中學習如何從無到有，直到成品完成的過程。

二、研究目的

利用樹莓派結合鏡頭及超音波測距儀，然後搭配軟體撰寫，以控制自走車的伺服馬達實現自動跟車之實作模擬。

首先透過鏡頭捕捉周圍環境的影像資訊，並進行即時分析以識別目標車輛的位置和運動狀態。再來利用超音波測距儀準確測量自走車與目標車輛之間的距離和方向，以確保跟車過程中的安全距離和運動軌跡。最後通過撰寫軟體程式，運用樹莓派的控制能力，適時調整伺服馬達的轉向角度和速度，實現自走車對目標車輛的穩定跟隨與運動控制。

貳、文獻探討

一、器材挑選

此專題研究主要是以 Raspberry Pi 4B 開發版為硬體核心，然後結合樹莓派鏡頭(Raspberry Pi Camera)、MG90S 伺服馬達、HC-SR04 超音波測距儀、麵包版、鋰電池、車體架構...etc。

二、主要開發版(Raspberry Pi 4)

樹莓派(Raspberry Pi)是英國樹莓派基金會所開發的一款單板電腦，屬於類 Unix 系統，其初始主要的目的是以低價硬體及自由軟體促進學校的基本電腦科學教育。使用 Broadcom 製造的 ARM 架構處理器，其硬體製造主要是由 Premier Farnell 和 RS Components 進行生產，目前樹莓派記憶體介於 1GB 至 8GB，儲存方式是透過 Micro-SD 卡，具備 USB 介面和 HDMI 視訊輸出(支援聲音輸出)，內建 Ethernet/WLAN/Bluetooth 網路鏈結的方式(依據型號決定)，並且有多種作業系統可供使用。曾憲祐(2022)。

本論文所使用的樹莓派屬於 Raspberry Pi 4 model B 版本，其硬體規格如表 2-1 所示。樹莓派實際硬體配置如圖 2-1 所示。

表 2-1 Raspberry Pi 4 model B 硬體規格表

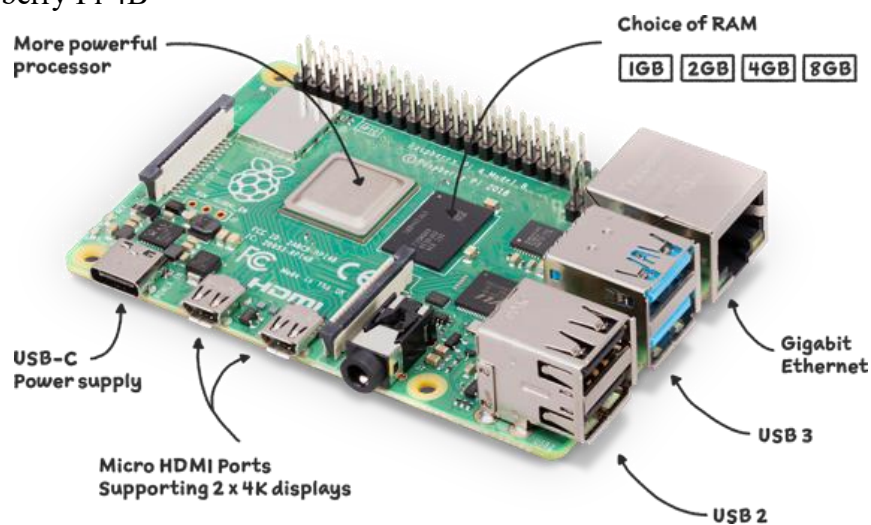
CPU	Broadcom BCM2711
CPU 速度	Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz(四核心)
GPU	Broadcom H.265 (4Kp60), H.264 (1080p60 / 1080p30)，OpenGL ES 3.1
記憶體	1GB, 2GB, 4GB or 8GB LPDDR4-3200 SDRAM

USB 介面數	USB 2.0 (2 個), USB 3.0 (2 個)
視訊輸出	Micro-HDMI (2 個), 最大支援 4Kp60, MIPI DSI 影像介面 (1 個)
音訊輸出	Micro-HDMI (2 個)
儲存	MicroSD 卡插槽
Ethernet	Gigabit Ethernet
Wireless	支援 2.4 GHz/5.0 GHz 802.11ac 無線網路
Bluetooth	Bluetooth 5.0
Camera	2-lane MIPI CSI camera port
GPIO 引腳數	40
額定功率	15 瓦 (5V/3A)
電源輸入	5V 電壓 (透過 USB-C 或經 GPIO 輸入)

資料來源：曾憲祐 (2022)。基於開發版樹莓派 Pi 4 之遠端遙控視訊小車。國立勤益科技大學電子工程研究所：碩士論文

https://ndltd.ncl.edu.tw/cgi-bin/gs32/gsweb.cgi/ccd=1MO_MA/search#XXX

圖 2-1 Raspberry Pi 4B



圖片來源：樹莓派官網 <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>

三、MG90S 360 度伺服馬達

MG90S 伺服馬達構造相似於 SG90，裡面含有直流馬達、齒輪箱、軸柄、以及控制電路，差別僅是採用金屬齒輪而非塑膠，360 度伺服馬達（連續旋轉）實際相當於無極變速的減速電機，可以控制速度和正反方向，沒有 0-360 度角度控制的功能(不可控制角度)，控制方式和一般伺服馬達的控制信號相同。當 PWM 為 0.5ms => 正向最大速度；1.5ms => 速度為 0；2.5ms => 反向最大速度。Shine (2021)。

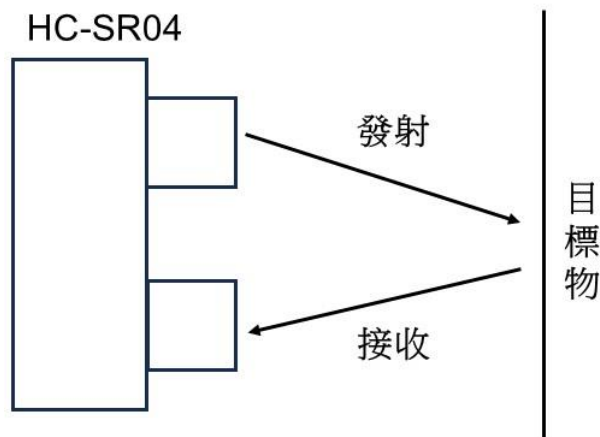
四、HC-SR04 超音波測距儀

超音波模組常用在無人機、智慧遙控車、自走車等專案，用來偵測障礙物的距離，作為

迴避障礙物的用途。典型的超音波感模組是由發射端、接收端以及 I/O 電路控制三大主要功能區塊共同組成。當 I/O 給予指示讓超音波模組觸發訊號時，發射端會馬上射出 40kHz 連續的聲波，此聲波會被距離模組最近的物件反射回模組的接收端，最後由控制電路計算出訊號發射出去至接收回來之訊號時間，如圖 2-2 所示。

由於聲音在空氣中傳遞的速度大約 340m/s，在傳遞易受大氣溫度所影響，溫度越高，傳遞速度越快。若聲音在空氣中傳遞的速度為 340 m/s，則聲音傳播 1 公分所傳遞的距離所需時間為： $1000000 / 340 * 100 = 29.4$ (ms) 透過四捨五入計算，得到聲音傳遞 1 公分所傳距離需要時間約為 29 ms (microseconds，百萬分之一秒)；因為模組發射端至返回接收端所行進的距離是待測物和超音波感模組距離的二倍，所以模組計算上需要將距離除以 2，以獲得物體本身實際距離。吳宗霖 (2022)。

圖 2-2 HC-SR04 超音波測距儀



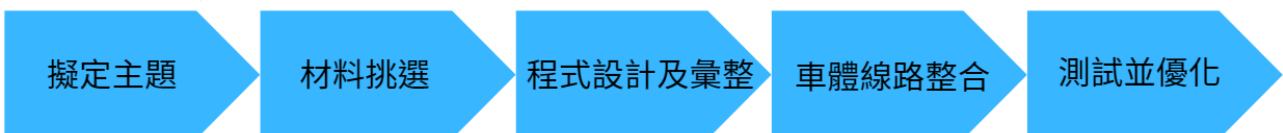
圖片來源: 研究者自行繪製

參、研究方法

一、研究流程

本研究流程如圖 3-1 所示。首先是擬定主題，思考主題的可行性，再來討論並挑選相對應的材料進行購買，接下來設計所有模組零件的程式碼且將所有程式碼彙整，然後將所有模組透過麵包板結合樹梅派和個人電腦進行車體線路整合，最後進行最終測試並不斷優化。

圖 3-1 研究流程



圖片來源: 研究者自行繪製

二、樹梅派程式設計

(一)系統運作架構

本次自動跟車系統電動車主要程式設計分成四大步驟: 鏡頭拍攝、影像辨識、超音波測距儀偵測、伺服馬達控制。而我們這次操作樹梅派的連線方式是採用 VNC 遠端操作軟體 (RealVNC Viewer) 來進行 (賀雪晨等, 2021)。本次系統運作架構主要是以多線成的方式同

時進行上述四大步驟，以鏡頭偵測並進行影像辨識，然後判斷所要追蹤物體的方向以控制伺服馬達進行車輛轉彎，同時用超音波測距儀在當距離低於 40 公分時進行煞車。

(二) 鏡頭拍攝

本次鏡頭偵測及傳送至 server 端的程式(client.py) 如圖 3-2 所示，首先主要先設定所拍攝照片的畫質，由於車輛方向控制是透過即時影像辨識來操控，為了讓偵測到辨識整個處理過程的傳送速度加快，我們將拍攝畫質降至最低以加快速度，再來是將拍攝好的畫面傳至 server 端進行影像辨識的程式解讀。

圖 3-2 鏡頭拍攝傳送及接收 server 端指令

```

30 while True:
31
32     ret,photo = cap.read()
33
34
35     # cv2.imshow('streaming',photo)
36     ret,buffer = cv2.imencode(".jpg",photo,[int(cv2.IMWRITE_JPEG_QUALITY),15])
37     x_as_bytes = pickle.dumps(buffer)
38     client_socket.send(x_as_bytes)
39     client_socket.send(b"stop")
40
41     messageFromServer = client_socket.recv(1024).decode()[0]
42     okay = True
43
44     """
45     Direction message:
46
47
48     1 stands for right
49     """
50
51     messageFromServer == "0" and messageFromServer != preMessage:
52         preMessage = "0"
53         print("Turn left")
54         direction = 0
55     elif messageFromServer == "1" and messageFromServer != preMessage:
56         preMessage = "1"
57         print("Turn right")
58         direction = 1
59     elif messageFromServer == "2" and messageFromServer != preMessage:
60         preMessage = "2"
61         print("Turn right")
62         direction = 2
63     elif messageFromServer == "start" and messageFromServer != preMessage:
64         preMessage = "start"
65         print("Start objtracking")
66
67     if cv2.waitKey(10)==ord('q'):
68         break
69

```

圖片來源: 研究者本人成果擷圖

(三) 影像辨識

物體追蹤(Object tracking)是指在一系列的圖片中追蹤特定物體。在這個過程中，首先選定或自動偵測要追蹤的物體，然後透過不同的演算法在每一幀中，依照物體移動的慣性來預測並更新物體的位置、大小和形狀。本研究旨在將這一技術應用於移動的車輛上，實現對前方物體的自動跟隨功能。

我們在程式教學網站 <https://learnopencv.com/>，查詢並學習如何使用 OpenCV 來追蹤物體的程式，如圖 3-3 是查詢到的重要程式碼。原作者(Satya Mallick, 2017)主要提供 7 種追蹤演算法，首先選擇輸入你要辨識的影像檔，在讀第一幀畫面後，並選擇追蹤物體。第 1 個參數以(bbox)作為該物體的邊界框(Bounding box)，第 2 個參數代表是否要顯示游標(Crossair)。接下來程式將會以第一幀的照片及選取的 Bounding box 初始化追蹤器，之後的程式都在 while 迴圈裡，持續讀取下一幀畫面，然後不斷地更新追蹤作為車體控制方向的計算。

圖 3-3 用 OpenCV 來追蹤物體的程式

```
# source: https://learnopencv.com/object-tracking-using-opencv-cpp-python/

import cv2
import sys

(major_ver, minor_ver, subminor_ver) = (cv2.__version__).split('.')

if __name__ == '__main__':

    # 作者提供7種追蹤演算法
    tracker_types = ['BOOSTING', 'MIL', 'KCF', 'TLD', 'MEDIANFLOW', 'GOTURN', 'MOSSE', 'CSRT']
    tracker_type = tracker_types[2]

    # 影像輸入來源，範例中設為影片。
    video = cv2.VideoCapture("videos/chaplin.mp4")

    # 讀取一幀畫面，並選擇追蹤物體，以bbox作為該物體的邊界框(Bounding box)，第2個參數代表是否要顯示游標(Crossair)。
    ok, frame = video.read()
    bbox = cv2.selectROI(frame, False)

    # 以第一幀的照片及選取的Bounding box初始化追蹤器。
    ok = tracker.init(frame, bbox)

    while True:
        # 讀取下一幀畫面，如果未讀取成功，結束迴圈。
        ok, frame = video.read()
        if not ok:
            break

        # 更新追蹤後的物體其邊界框的位置。
        ok, bbox = tracker.update(frame)

        # 如果追蹤成功，在使用者畫面上畫出該物體。
        if ok:
            # 成功
            p1 = (int(bbox[0]), int(bbox[1]))
            p2 = (int(bbox[0] + bbox[2]), int(bbox[1] + bbox[3]))
            cv2.rectangle(frame, p1, p2, (255,0,0), 2, 1)
        else:
            # 失敗
            cv2.putText(frame, "Tracking failure detected", (100,80), cv2.FONT_HERSHEY_SIMPLEX, 0.75,(0,0,255),2)

        cv2.putText(frame, tracker_type + " Tracker", (100,20), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (50,170,50),2);

        cv2.putText(frame, "FPS : " + str(int(fps)), (100,50), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (50,170,50), 2);

        # 顯示畫面供使用者檢視。
        cv2.imshow("Tracking", frame)

        # 按ESC鍵離開
        k = cv2.waitKey(1) & 0xFF
        if k == 27: break
```

資料來源: (Satya Mallick, 2017)

<https://learnopencv.com/object-tracking-using-opencv-cpp-python/>

為了提升運算效能，我們透過 Socket 模組將影像傳送至電腦，以筆電為 server，樹梅派為 client 端。程式碼我們參考(Abhishek kesare, 2021) <https://github.com/abhikesare9/live-streaming-with-opencv>，最後依照我們設計的模組需求，自行修改後的server.py完整程式碼如圖 3-4，自行修改後的 obtrack.py 程式碼如圖 3-5。

圖 3-4 server.py

```
import cv2, socket, numpy, pickle
import time

class Server:

    def __init__(self, clientIP):
        print("start building socket...")
        self.s=socket.socket()
        self.host = socket.gethostname()
        print(self.host)
        self.ip=clientIP
        self.port=6666
        self.s.bind((self.host,self.port))
        self.s.listen(2)
        self.conn, address = self.s.accept()
        print("Connection from: " + str(address))
        print("connect successfully")

    def recvImageFromClient(self):
        x=self.conn.recvfrom(1000000)
        self.data=x[0]
        self.data=pickle.loads(self.data)
        self.data = cv2.imdecode(self.data, cv2.IMREAD_COLOR)

        return self.data

    def sendMessageToClient(self, message):
        """
        Direction message:

            0 stands for left
            1 stands for right
        """
        self.conn.send(message.encode())
```

資料來源：研究者本人成果擷圖

圖 3-5 obtrack.py 判斷車輛方向

```
client = "0.0.0.0"

if __name__ == '__main__':
    myServer = Server(clientIP = client)
    myServer.sendMessageToClient("Hello from server")
    frame = myServer.recvImageFromClient()
    ...
    ...
    bbox = cv2.selectROI(frame, False)

    while True:
        myServer.sendMessageToClient("")
        # 從樹梅派接收照片
        frame = myServer.recvImageFromClient()
        ok, bbox = tracker.update(frame)

        # 傳給樹梅派追蹤結果
        directionMessage = ""
        if centerPoint[0] - currentPoint[0] > 0:
            directionMessage = "Turn Left"
            myServer.sendMessageToClient("0")
        else:
            directionMessage = "Turn Right"
            myServer.sendMessageToClient("1")
```

資料來源：研究者本人成果擷圖

(四) 超音波測距儀控制

超音波測距儀在本研究中，主要是負責當自走車過於靠近前方車輛或物體時，進行煞車的功能。而此次超音波測距儀的程式碼如圖 3-6 所示，首先 Trigger 端發射出超音波訊號持續 10 微秒後，再來開始進行第 36 行的時間計算。也就是當超音波打出，開始計算當下時間，直到接收到後，再計算一次當下時間，然後兩個時間相減得到時間差值後，乘以聲速轉換為距離，讓樹梅派接著判斷持續進行馬達轉動或應該停止。

圖 3-6 超音波測距儀的程式碼

```

22 def getDistance(self):
23
24     GPIO.output(self.pinTrigger, True)
25
26     # set Trigger after 0.01ms to LOW
27     time.sleep(0.00001)
28     GPIO.output(self.pinTrigger, False)
29
30     StartTime = time.time()
31     StopTime = time.time()
32
33     # save StartTime
34     while GPIO.input(self.pinEcho) == 0:
35         StartTime = time.time()
36
37     # save time of arrival
38     while GPIO.input(self.pinEcho) == 1:
39         StopTime = time.time()
40
41
42     # time difference between start and arrival
43     TimeElapsed = StopTime - StartTime
44     # multiply with the sonic speed (34300 cm/s)
45     # v = 331+0.6t
46     # and divide by 2, because there and
47     soundSpeed = (331 + self.temperature)*100
48     distance = (TimeElapsed * soundSpeed) / 2
49
50     print("current distance: %.4fcm" % distance)
51
52     return distance
53
54 def mainLoop():
55     currentDistance = myDisSensor.getDistance()
56     print("current distance: %.4fcm" % currentDistance)
57

```

圖片來源：研究者本人成果擷圖

(五) 伺服馬達轉彎控制

伺服馬達作為本論文自動跟車系統自走車的動力來源，主要功能在於接收影像辨識傳來的方向指令，以及超音波測距儀的煞停指令，主要部分程式碼如圖 3-7 所示。

由於我們的伺服馬達脈衝寬度約在 0.5ms 到 2.5ms，因此在操控的時候 DutyCycle 在 7.5 時不轉動，而越靠近 2 是正轉速度越快，越靠近 12 時則是逆轉速度越快。因為伺服馬達作為輪胎驅動動力，一顆在左邊另一顆在右邊，因此左右輪轉動方向須不同。然而每顆馬達轉速無法完全一樣，所以我們必須考量左右兩顆方向問題，同時也需測試並調整不同的轉速，來達到自走車行進時不同速度的需求。

圖 3-7 伺服馬達驅動控制程式

```

70 def GoStraight(self):
71
72     print("Car starts to go straight.")
73     self.left_pwm.ChangeDutyCycle(9.5)
74     self.right_pwm.ChangeDutyCycle(5)
75     '''self.left_pwm.ChangeDutyCycle(8.5)
76     self.right_pwm.ChangeDutyCycle(6)'''
77     self.status = STRAIGHT
78
79 def TurnRight(self):
80
81     print("Car starts to turn right.")
82     self.left_pwm.ChangeDutyCycle(8.5)
83     self.right_pwm.ChangeDutyCycle(7)
84     '''self.left_pwm.ChangeDutyCycle(8.5)
85     self.right_pwm.ChangeDutyCycle(6.5)'''
86     self.status = RIGHT
87
88 def TurnLeft(self):
89
90     print("Car starts to turn left.")
91     self.left_pwm.ChangeDutyCycle(7.5)
92     self.right_pwm.ChangeDutyCycle(6)
93     '''self.left_pwm.ChangeDutyCycle(8)
94     self.right_pwm.ChangeDutyCycle(6)'''
95     self.status = LEFT
96
97 def Stop(self):
98
99     print("Stop the car.")
100     self.left_pwm.ChangeDutyCycle(0)
101     self.right_pwm.ChangeDutyCycle(0)
102
103     self.status = STOP

```

圖片來源：研究者本人成果擷圖

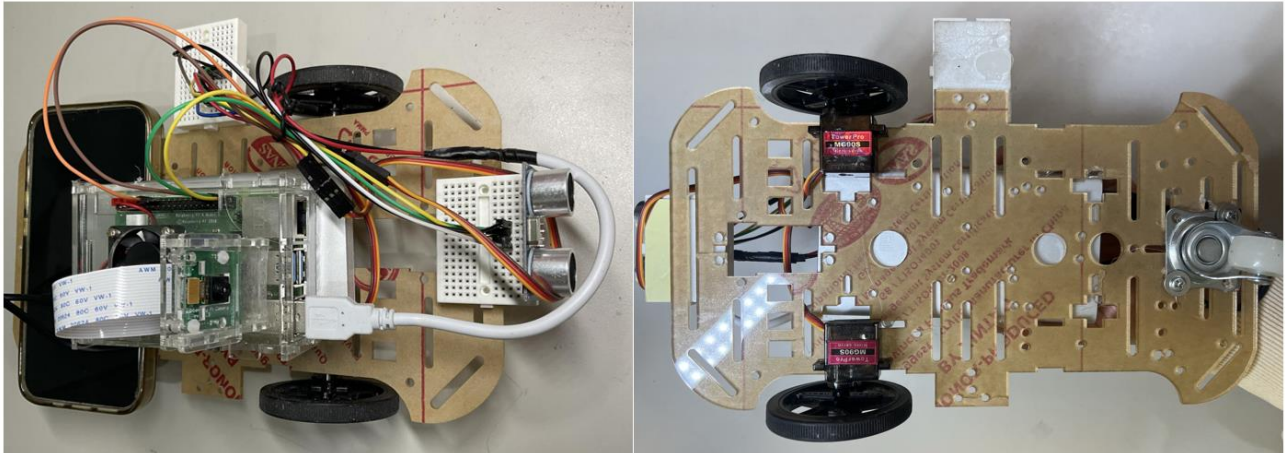
肆、研究分析與結果

一、 成果展示

本次論文自走車自動跟車系統是以壓克力板作為車身結構，以行動電源作為電力提供來源，正面是將行動電源安裝在中心，再將鏡頭和樹梅派放置於行動電源上方，超音波測距儀放於車輛最前方，伺服馬達則是放於車輛下方，並由兩塊小型麵包板分別將超音波測距儀和

伺服馬達連接到樹梅派的 GPIO 插槽上。由於我們是透過手機作為 WIFI 分享器，讓樹梅派和筆電處在共同網域下，為了讓影像資料傳送速度加快，我們將共同網域的手機放置於車輛後端，車輛正反面如圖 4-1 所示。。

圖 4-1 自走車正反面圖



圖片來源：研究者本人拍攝

二、研究過程困境與優化

(一) 超音波測距儀挑選

原先我們選擇用 TOF400F 紅外線測距儀作為距離偵測的偵測器，然而我們嘗試要去測試測量距離時，卻無法正常使用，而且因為是國外的產品，網路上查詢的資料很少，也可能是電子零件用海運的方式有所影響，因此我們後來改用超音波測距儀 HC-SR04 就可以正常運作了。圖 4-2 為兩者測距儀當時測試實驗結果。

圖 4-2 測距儀測試實驗

	測試方法	實驗結果	註解
TOF400F	使用網上教學的程式碼並修改	產生亂碼	1. 網路提供的程式碼有誤 2. 透過海運使產品潮解
HC-SR04	使用網上教學的程式碼並修改	經程式運算可轉換成距離	正常運作

圖片來源：研究者本人製作

(二) 馬達挑選及供電問題

我們原本預設是要用 TT 直流減速馬達，由於我們的車輛左右轉時是透過輪轉速度不同，而要控制 TT 直流減速馬達達到左右輪速度不同，則需要用到其他驅動板如 L298N，為了簡化控制及接線，因此我們改用伺服馬達 MG90S。

雖然伺服馬達的速度控制不算困難，但要驅動一顆 MG90S 最大電流約 300mA，因此無法藉由樹梅派 GPIO 孔的 5V 電源直接供應。於是我們經由網路查詢及 FB 台灣樹莓派社群討論，得到的各種資訊來進行測試，如圖 4-3 所示。最後實驗的結果決定經由樹梅派 USB Type-A 孔接線拉出 5V 電源來提供給兩顆伺服馬達使用。

圖 4-3 伺服馬達不同的供電實驗

伺服馬達電源驅動實驗					
樹梅派外接電源方式	如何供應馬達電源	電壓	電流	結果	原因
USB Type-C	透過樹梅派GPIO正負極	5.1	46mA	馬達只動一下，CPU 非常燙	樹梅派GPIO 供電電流受限
樹梅派GPIO 正負極	從樹梅派GPIO另一組正負極	5.2	NA	樹梅派燒毀，無法運作	雖然文獻資料稱樹梅派GPIO 孔 5V也能當輸入端，但實測過程樹梅派出現當機後綠色 LED 一直閃爍異狀，無法重新開機，經FB社群詢問後應該是樹梅派燒毀
USB Type-C	從樹梅派USB Type-A	5.08	245mA	馬達可正常運行	從樹梅派上USB孔輸出的電流夠大
USB Type-C	從行動電源	5.15	248mA	馬達可正常運行但接線複雜較長	從行動電源上USB孔輸出的電流夠大

圖片來源：研究者本人製作

(三) 左右輪馬達轉速不一致

本自走車的伺服馬達擺設採兩邊對放的方式，因此要使車輛能直走必須右側輪子正時鐘轉左側輪子逆時鐘轉，但可能因為兩個馬達之間有些許誤差，所以當程式設定在左右輪正反轉時，常常因為左右輪 Duty Cycle 值的匹配不同，導致自走車經常偏左行進，無法讓車輛直線前進。因此我們經過幾次的實驗測試後如圖 4-4 所示，終於得到讓車輛直行的最佳穩定三組 Duty Cycle 數據。

圖 4-4 馬達轉速不一的實驗

馬達速度不一實驗結果						
	DutyCycle 參數		預期結果	實驗結果	自走車速度	原因
	左輪參數	右輪參數				
1	8	6	直走	偏左		此參數下馬達轉速，右側轉速比左側快
2	8	6.5	偏右	直走	慢速	左側馬達不變，但右側馬達轉速變慢之下，達到兩側平衡
3	10	4	直走	偏左		左右兩側同時調教變快，但右側轉速依舊比左側快
4	10	4.5	偏右	直走	中速	雖然左側馬達不變，但右側馬達轉速變慢之下，達到兩側平衡
5	11	3	直走	偏左		當左右兩側調教再更快，右側轉速還是比左側快
6	12	2	直走	直走	快速	左右兩側馬達調教再變更快，兩側馬達可達到平衡

圖片來源：研究者本人製作

(四) 網傳速度過慢

我們發現如果單獨在樹梅派上進行影像辨識運算，效能甚差，不但無法追蹤即時影像，還可能導致過熱當機的情況。有鑑於此，所以我們把樹梅派得到的即時影像先傳到筆電，讓筆電處理影像辨識運算後，再把結果傳回樹梅派進行自走車方向控制。然而速度雖然有加快，但實際測試時發現車輛在第一次轉彎時容易轉彎過頭，後來才發現是因為 WIFI 網路傳輸速度過慢，導致新的圖像分析尚未回傳至樹梅派，自走車因跟車跟丟了，而轉彎過頭。經過我們實驗發現手機當 WIFI 分享器，放置於自走車平台上跟車的效果最佳，如圖 4-5 所示。

圖 4-5 WIFI 分享器放置位置之實驗

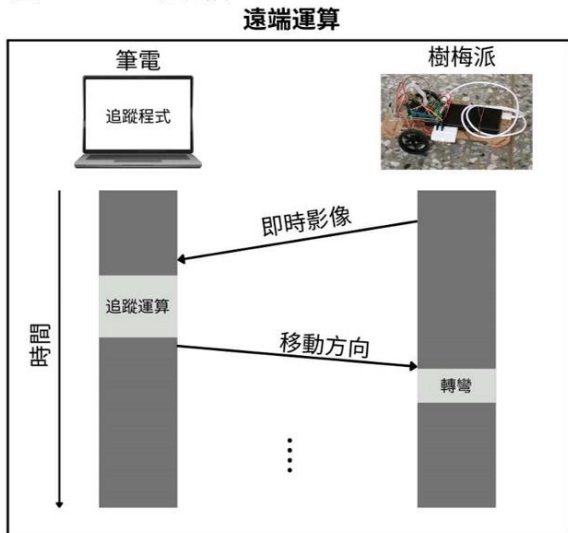
WIFI 網路連線實驗結果			
	WIFI 分享器位置	測試結果	當下懷疑原因
1	手機當WIFI 分享器放置於筆電與自走車中繼	跟車反應速度經常延遲	可能手機當WIFI 分享器，傳輸速度比較慢
2	家用WIFI 分享器放置於筆電與自走車中繼	跟車反應速度依舊會延遲	WIFI 發射器同時經過一段距離連線樹梅派及筆電，傳輸速度受限於兩側距離
3	手機當WIFI 分享器放置筆電旁	基本上無法進行跟蹤	可能是樹梅派WIFI 傳輸距離受限，導致傳輸速度變慢
4	手機當WIFI 分享器放置自走車平台上	跟車反應速度較快	樹梅派網傳速度相較於筆電來的慢，分享器靠近樹梅派，傳輸速度得以改善，跟蹤物體能力明顯變好

圖片來源：研究者本人製作

(五) 執行程式時，不斷報錯 bug

當在執行鏡頭辨識時，執行程式不斷報錯顯示 bug，後來在網路上搜尋後，發現這是因為 server 端在接收時無法一次就接收如此大的影像。為了要完整接收影像，就必須分次接收好幾次直到影像接收完成。我們基於 TCP 的架構如圖 4-6，修改 server 端及 client 端的程式碼。修改後的程式碼如圖 4-7，4-8。

圖4-6 TCP的架構



圖片來源：研究者本人製作

圖 4-7 server.py

```
def recvImageFromClient(self):
    image = b""
    while True:
        x=self.conn.recv(1000000)
        image += x
        if x[-4:] == b'stop': # 停止接收
            image = image[:-4]
            break
    self.data = image
    self.data=pickle.loads(image)
    self.data = cv2.imdecode(self.data, cv2.IMREAD_COLOR)
    return self.data
```

圖片來源：研究者本人成果擷圖

←

圖 4-8 client.py

```
client_socket.send(x_as_bytes)
client_socket.send(b'stop') # 告訴server停止接收
messageFromServer = client_socket.recv(1024).decode()
```

圖片來源：研究者本人成果擷圖

(六) 車輛軌跡程 S 型

當實測自走車在直線跟蹤物體行進時，不是往左就是往右軌跡呈現 S 型，即使被跟蹤物體沒有明顯左右移動亦是如此。原本以為是車輪不正的關係，經研究發現原來是因為在追蹤後，我們跟蹤的程式把方向只設定為「左」或「右」，沒有「直走」的情況考量，導致即使物體相較於自走車正中心的偏差很小時，仍會判定要轉彎。藉由實驗測試後如圖 4-9，我們得到一組最佳的 Diff 值，來做為追蹤物體後自走車是否轉彎或直行的依據。因此我們將 objtrack.py 方向判定的程式修改為如圖 4-10。當追蹤物體與自走車正中心相差不到 30 個像素時，程式解讀為直行的可接受誤差，樹梅派就會就讓自走車繼續執行直走的動作。

圖4-9 兩者中心距離 (Diff) 實驗結果

被追蹤的物體與自走車中心點相差距離 (Diff) 實驗結果		
	距離 (Diff 值)	自走車跑動結果
1	0	不斷地S行扭動
2	10	容易造成S行扭動
3	30	可近乎直線
4	50	當物體轉彎時，自走車反應過慢

圖片來源：研究者本人製作

圖 4-10 objtrack.py

```
directionMessage = ""

diff = centerPoint[0] - currentPoint[0] # 偏差

if ( abs(diff) > 30 ):
    if diff > 0:
        directionMessage = "Turn Left"
        myServer.sendMessageToClient("0")
    else:
        directionMessage = "Turn Right"
        myServer.sendMessageToClient("1")
else: # 若偏差過小，直走即可
    directionMessage = "Go Straight"
    myServer.sendMessageToClient("2")
```

圖片來源：研究者本人成果擷圖

伍、研究結論與建議

一、結論

在這次自動跟車系統實作研究，我們成功地開發出自動跟車系統的能力，讓自走車透過影像辨識及超音波測距，判別前方物體相對於本車位置，執行了自動跟車直行、轉彎或煞停的動作。讓我們學習了如何從無到有實作開發一個系統專題，其中包括題目制定、材料挑選購買、樹莓派及相關零件程式設計、車體線路整合、最後結合電腦測試並優化。當然中間遇到了很多的瓶頸與困難，不過經由我們彼此的討論，藉由書籍論文與網路的查詢，以及老師的指導之下，終於成功完成此論文自動跟車系統開發與研究。

二、可優化部分

本次車體的系統控制主要以樹莓派來進行，然而經由我們實驗發現樹莓派對影像辨識的處理能力不足，因此我們將影像先傳送至個人電腦進行辨識處理後，再以指令的方式回傳至樹莓派端，這樣的結果發現其速度有明顯變快，但由於是透過兩次的傳送，仍然會有些許的網路延遲造成自走車反應稍慢，以至於我們的車輛無法加快速度。因此期待將來有更高效能的單板電腦(SBC)來取代現有的樹莓派，讓自走車的反應及跟車能力更佳更為流暢。

三、未來展望

電動車將會成為未來汽車的主流之一，而自動跟車及自動駕駛將在未來能輔助人類進行車輛操控，藉本次自動跟車系統實作研究，期待未來大學有機會將此經驗繼續累積擴展，在加大車體大小和馬力的情況下，結合 5G 和 GPS 定位再融合人工智慧的程式設計，達到自動駕駛自動跟車及路線規劃控制能力...等等。吳宗霖（2022）。

陸、參考文獻

1. 賀雪晨、孫錦中、劉丹丹、謝凱年、楊佳慶、仝明磊（2021）。**樹莓派智能項目設計**。清華出版社。
2. 柯博文（2015）。**Raspberry Pi 最佳入門與實戰應用(第二版)**。碁峰出版社。
3. Device Plus。河島 晋（2023 年 11 月 15）從歷史到使用方法的全面瞭解！電子作品創作不可或缺的“Raspberry Pi（樹莓派）”究竟是何方神聖？
<https://micro.rohm.com/tw/deviceplus/how-to/raspberrypi-guide/basic-of-raspberrypi/>
4. 曾憲祐（2022）。基於開發版樹莓派 Pi 4 之遠端遙控視訊小車。國立勤益科技大學電子工程研究所：碩士論文
https://ndltd.ncl.edu.tw/cgi-bin/gs32/gswweb.cgi/ccd=1MO_MA/search#XXX
5. Shine（2021 年 1 月 28 日）。STM32 控制 SG90 舵機教程（180 度和 360 度）。
<https://blog.csdn.net/SHRtuji/article/details/113354315>
6. 吳宗霖（2022）。**智能自走車之開發**。正修科技大學／電子工程研究所：碩士論文
[臺灣博碩士論文知識加值系統：自由的博碩士學位論文全文資料庫 \(ncl.edu.tw\)](https://www.ncl.edu.tw/academic/graduate/thesis/thesis_list.htm)
7. Raspberry Pi 官網 <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>
8. Satya Mallick (2017)。 <https://learnopencv.com/object-tracking-using-opencv-cpp-python/>
9. Abhishek kesare(2021) <https://github.com/abhikesare9/live-streaming-with-opencv>