新北研遊匯
新北市111年度

資訊類

中和國小

叫你不要跑 — 校園碰撞預警機制
應用 NodeMCU 與 OpenCV 作為 AI 物件辨識之研究

研究者：呂顥天　指導老師：黃文聖、張凱茵

壹、研究動機

學校常常會發生走廊奔跑導致碰撞、跌倒等事件，後果總是導致劇烈無比的傷害，且學校老師、主任，甚至是校長，不斷宣導後仍沒有效果，後來學校又舉行了禮讓小手活動，讓各班同學輪流管理教室前走廊上跑步的學生，雖然許多學生感到厭煩，但是這個活動還是有效果的，然而禮讓小手活動結束了，卻提供了研究者一個想法：如果無法利用人力避免，就以科技的方式預防，或許成效會更好，因此研究者假設運用測速照相的模式，開發出校園碰撞的預警系統，希望能符合需求，進而提供學生一個更安全的校園生活環境。

貳、研究目的

一、製作出校園碰撞預警裝置
二、測試各種情境並改善問題
三、以實體與書面將成果呈現
四、應用本研究成果推廣至校園以避免學生碰撞受傷

參、 研究設計與流程

一、由速度偵測和物件辨識等相關研究文獻著手，藉此找尋出適用於本研究之架構、軟硬體與驗證方法，並評估其可行性。
二、規劃出裝置的整體架構並找尋適用本研究之各項硬體設備。
三、開發前端用來判斷速度的軟體程式與後端的影像中物件辨識程式，以及最後觸發聲音模組和擷取影片的程式。
四、佈置環境進行測試，評估成果的正確性與可行程度，並對系統進行改善修正。
五、進行實測，並以混淆矩陣來驗證開發系統之評估指標。
六、分析探討研究結果並整理與撰寫研究報告。

肆、結論

一、系統實用性：

本研究所提出的校園碰撞預警機制，是利用紅外線熱感應器偵測人體移動，透過物聯網元件 NodeMCU 來分析移動速度，藉由邊緣運算概念先由前端元件處理部份資料後，以無線網路傳至主機，並搭配 AI 人工智慧技術，應用 OpenCV 函式庫運算物件辨識，最後對在走廊上奔跑的學生發出告警，並擷取奔跑影像，以達到維護校園安全的目的。以實測之評估數據來看，確實可行。

二、成本因素：

本系統可拆開分成前端偵測和後端辨識與錄影，若僅裝設前端偵測告警，成本僅約數百元，可放置校園多處易發生碰撞的地方，但是缺乏後台 AI 人工智慧之物件辨識和錄影，且易發生誤判之情形。但是如果架設整套系統，還需要網路攝影機和電腦主機，費用較高，且需要考量電路和資訊線路。

三、 破壞問題：

系統在建置和實測時，就有不少學生因為好奇，經過時就碰一下，也有故意掰斷保麗龍夾版和扯出紅外線熱感應器的情形，在　動錄影功能且張貼告示，並告誡破壞的學生後，破壞情形已改善。

伍、建議

一、 技術層面

雖然實測數據顯示本系統性能指標都在可接受範圍，但紅外線熱感應器仍有延遲、溫度、日照、樹影干擾等影響判讀因素，日後研究方向可以用其他深度學習的演算法，如 YOLO、R-CNN 等技術，直接以影像分析物件移動，自動啟動並計算速度，就可捨棄前端設備，只留網路攝影機，即可運作相同功能，而且更準確。

二、 教育層面

本系統建置的目的是要提醒學生在走廊不要奔跑，來避免遭受傷害，但這是輔助的告警機制，校園安全還是要每一個學生自發性的不要在走廊上奔跑，就像校長說的，要回歸教育的本質，我們在學校除了學業的學習，也要聽從師長在生活上的教導，來保護自己和別人的安全。

叫你不要跑 — 校園碰撞預警機制

應用 NodeMCU 與 OpenCV 作為 AI 物件辨識之研究

研究者 呂顥天

壹、 緒論

一、 研究動機

學校常常會發生走廊奔跑導致碰撞、跌倒等事件，後果總是導致劇烈無比的傷害，且學校老師、主任，甚至是校長，不斷宣導後仍沒有效果，後來學校又舉行了禮讓小手活動，讓各班同學輪流管理教室前走廊上跑步的學生，雖然許多學生感到厭煩，但是這個活動還是有效果的，然而禮讓小手活動結束了，卻提供了研究者一個想法：如果無法利用人力避免，就以科技的方式預防，或許成效會更好，因此研究者假設運用測速照相的模式，開發出校園碰撞的預警系統，希望能符合需求，進而提供學生一個更安全的校園生活環境。

二、 研究目的

(一) 製作出校園碰撞預警裝置。

(二) 測試各種情境並改善問題。

(三) 以實體與書面將成果呈現。

(四) 應用本研究成果推廣至校園以避免學生碰撞受傷。

三、 研究設計與流程

(一) 由速度偵測和物件辨識等相關研究文獻著手，藉此找尋出適用於本研究之架構、軟硬體與驗證方法，並評估其可行性。

(二) 規劃出裝置的整體架構並找尋適用本研究之各項硬體設備。

(三) 開發前端用來判斷速度的軟體程式與後端的影像中物件辨識程式，以及最後觸發聲音模

組和擷取影片的程式。

(四) 佈置環境進行測試，評估成果的正確性與可行程度，並對系統進行改善修正。

(五) 進行實測，並以混淆矩陣來驗證開發系統之評估指標。

(六) 分析探討研究結果並整理與撰寫研究報告。

貳、 文獻探討

一、 整體概念與架構

(一) 人工智慧(Artificial Intelligence, AI)

工研院產業科技國際發展所所長蘇孟宗(2018)指出人工智慧發展歷程自 1950 年起至今已近 70 年，在全球科技發展趨勢上，AI 已經成為發展軟實力的關鍵技術。AI 科技最終的目標是為了帶給人類更便利的生活、促進人類知識的演進。目前 AI 主流為透過將五種感知功能嵌入智慧機器並結合 AI 運算，得以藉由視線、聲紋、肌力、觸感、氣體辨識，讓使用者能與具 AI 特性之智慧型系統進行人機互動，從而協同工作，最終開啟全新的應用情境與市場商機(蘇孟宗 & 陳右怡, 2018)。本研究即以上述五感中之視線與觸感功能嵌入設備並結合 AI 運算之智慧系統。

(二) 物聯網(The Internet of things, IoT)

物聯網 The Internet of things，從英文名稱看來，顧名思義，物聯網就是物物相連的網際網路，其中有兩層意思：第一，物聯網的核心和基礎仍然是網際網路，是在網際網路基礎上的延伸和擴展的網絡；第二，其用戶端延伸和擴展到了任何物品與物品之間，進行信息交換和通信。因此，物聯網的定義是通過紅外感應器、射頻識別、雷射掃描器、全球定位系統等訊息傳感設備，依通訊協定，把任何物品與網際網路相連接，進行信息交換與通信，以實現對物品的智慧化識別、定位、監控、跟蹤和管理的一種網路(Li, Xu, & Zhao, 2015)。本研究之前端各元件即為物聯網之應用。

(三) 邊緣運算(Edge Computing)

維基百科依據 Lopez 等人的文章，解釋邊緣運算是一種分散式運算的架構，將

應用程式、數據資料與服務的運算，由網路中心節點，移往網路邏輯上的邊緣節點來處理。邊緣運算將原本完全由中心節點處理大型服務加以分解，切割成更小與更容易管理的部份，分散到邊緣節點去處理。邊緣節點更接近於用戶終端裝置，可以加快資料的處理與傳送速度，減少延遲。本研究於偵測端已經先處理速度的計算，之後再將結果傳送之主機端，降低主機端的運算負擔，就是邊緣運算的應用。

二、 硬體裝置

(一) 速度偵測元件

物體直線速度可以許多方式偵測，其原理不外乎為無線電波、光波、聲波、氣流等， (梁依儒, 2016)之解釋為以感測器 Sensor 感應物件通過偵測區域，獲得時間、位置、流量、速度等資料。本研究為偵測學生行走或跑步速度，物件為活體熱源，故選擇紅外線熱感測器 Passive Infrared Sensor (PIR Sensor)最為適當。

(二) 紅外線熱感測器(Passive Infrared Sensor, PIR)

紅外線熱感測器(PIR)是眾所周知的探測器，由於它的低成本和低功耗，小巧的外形以及不顯眼且保護隱私的交互方式，已被廣泛用於人體跟踪系統。特別的是具有跟踪人類運動，識別行走的物體以及對進入或離開房間或建築物入口的人進行偵測與計數的功能，PIR 傳感器的模擬輸出信號所涉及的範圍不僅僅限於簡單的人，還包括人體與 PIR 間的距離，運動的模式(即方向和速度)，身體的形狀和步態，因此，我們可以利用 PIR 的輸出信號的判別特徵來開發各種用於人體跟踪和定位的應用程序(Yun & Lee, 2014)。

(三) 開發板(Evaluation Board)

又稱為 Demo Board，是用來進行嵌入式系統開發的電路板，包括中央處理器、存儲器、輸入設備、輸出設備、數據通路/匯流排和外部資源介面等一系列硬體組件(Liu, 2017)。目前開發板產品眾多，常見的有 Arduino、Raspberry Pi、Linkit One、Ameba 等，本研究衡量開發板之體積、成本、功能、編譯器，以及所需使用之感應元件後，選擇之開發板為 NodeMCU。

(四) NodeMCU

NodeMCU 是用於 IoT 的開發平台，運行在 ESP8266 Arduino 內核上，程式碼基於 C/C++語言編譯，具有完整支援 TCP/IP 的 2.4 GHz Wi-Fi 晶片 ，系統可以遠端自動控制，並且 NodeMCU 具有 128 KB 的暫存、4 MB 的存儲，11 個數字輸入/輸出，1 個模擬輸入/輸出通道。該板的尺寸小於 Arduino Uno，最重要的是，這個微控制器開發板是成熟的 ESP8266 技術，整合豐富的可用資源(Waleed, Sathish Kumar, Raed, & Chandrasekharan, 2018)。因此，該開發板非常適合本研究。

(五) DFPlayer Mini 語音模組

本研究之告警聲音，是學務主任叫學生不要跑的 mp3 檔案，所以使用了 DFPlayer Mini 語音播放模組，該模組外觀小巧又價格低廉，可直接連接揚聲器撥放簡單的音樂或語音，內建常見的音頻格式解碼，如 MP3、WAV 和 WMA 格式，同時支援 TF 卡驅動，支援 FAT16、FAT32 檔案系統，通過簡單的指令即可完成播放指定的音訊(Arunkumar, Kriti, Das, & Bhattacharyya, 2019)。

(六) 乙太網路供電技術(Power over Ethernet, PoE)

Ahmed, Hasaneen, and Orabi (2018)說明 PoE 是依附在乙太網路架構而在發展出來的一種技術，其優點在於利用相同的網路纜線，相同的通訊標準，但是可以同時傳送資料與電源。利用乙太網路供電技術即可方便的解決設備地點無電源的問題。也因為 PoE 是架構在乙太網路上，故有 100 公尺的傳輸距離，此優點也是其他種混合資料和電源傳送的介面所無法達到的。本研究之攝影機若以無線網路連接，就須另尋找電源插座去連接，基於上述考量，故選擇 PoE 之交換器及網路攝影機。

三、 軟體開發

(一) Integrated Development Environment (IDE)

撰寫程式碼需要一套編輯器（Editor）、編譯器（Compiler）、一個連結器（Linker）、除錯器（Debugger），而把這些集合起來成為一個方便的開發環境，就是 IDE。而 Visual Studio Code 是一個由 Microsoft 開發，同時支援 Windows、Linux 和 macOS 等作業系統，且開放原始碼的程式碼，支援多種程式語言(Amann, Proksch, Nadi, & Mezini, 2016)。

Visual Studio Code 可以在編輯器中執行指令碼、編譯軟體、除錯指令碼、設定斷點、做版本管理，所以 Visual Studio Code 是跨平台與跨程式語言的整合程式設計環境(Kaplan, 2018)。本研究前端是以 Arduino 程式語言撰寫 NodeMCU 開發板的控制程式，而後端是以 Python 程式語言來撰寫物件辨識與擷取影片的程式，兩種程式語言都可以在 Visual Studio Code 的開發環境下編寫。

(二) Open Source Computer Vision Library(OpenCV)

Open Source Computer Vision Library 是具有跨平台且支援多種程式語言的視覺函式庫，能以 Python、C/C++、Java 在 Windows、MacOS、Linux 的作業系統下使用，可以開發機械學習演算法，以及圖像處理、機器視覺系統等辨識功能，應用的範圍包括擴增實境(Augmented Reality)、人臉辨識(Face Recognition)、手勢識別(Gesture Recognition)、人機互動(Human-Machine Interaction)、運動追蹤(Motion Tracking)、物件辨識(Object Recognition)等各類領域，十分廣泛(Noble, 2016)。因 OpenCV 適用於物件辨識，所以本研究以 OpenCV 做為處理影像辨識的函式庫

四、 評估指標

Tharwat (2018)指出混淆矩陣是資料科學、資料分析和機器學習中總結分類模型預測結果的情形分析表，以矩陣形式將資料集中的記錄按照真實的類別與分類模型作出的分類判斷兩個標準進行匯總。基本的混淆矩陣是將結果分成四種類型，如表 2-1 混淆矩陣定義所示，TP 為目標正類且判斷為正類，FN 為目標正類但被判斷為負類，FP 為目標負類但被判斷為正類，TN 為目標負類且被判斷為負類。

表 2-1 四種數值評估的參數表示

	正類-有異常狀態 (應發出告警)	負類-無異常狀態 (不應發出告警)
正類 系統有發出告警	True Positive (TP)	False Positive (FP)
負類 系統沒有發出告警	False Negative (FN)	True Negative (TN)

透過混淆矩陣，可計算出正確率(Accuracy)、敏感度(Sensitivity，亦為召回率 Recall)、精確度（Precision）、特異性(Specifity)以及調和均值（F1 score）。

以本研究為例，Accuracy 對整個系統總體判斷的準確率，數值越高表示系統整體對樣本的識別能力越強；Recall 是指真實應發出告警的樣本中，系統發出告警的樣本數所佔的比例，數值越高表示模型對正類樣本的識別能力越強；Precision 是指系統發出告警的樣本中，真正應發出告警的樣本數所佔的比例，數值越高，表示系統也是對正類樣本的區別度越好；Specificity 是指真實不應發出告警的樣本中，系統沒有發出告警的樣本數所佔的比例，數值越高表示模型對負類樣本的識別能力越強。F1 Score 則是綜合了 Precision 和 Recall 的一個指標，用於判斷系統的穩定性，其分數介於 0 到 1 之間，越接近 1 則穩定性越高，算法如式：

正確率(Accuracy) $= \frac{TP+TN}{TP+FP+FN+TN}$ ←對整個系統總體判斷的準確率

敏感度(Sensitivity) = 召回率(Recall) $= \frac{TP}{TP + FN}$ ←真實應發出告警狀況的準確率

精確度(Precision) $= \frac{TP}{TP + FP}$ ←系統發出告警的準確率

特異度(Specificity) $= \frac{TN}{TN+FP}$ ←真實不應發出告警狀況的準確率

$$F1\ \ Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

參、 研究方法與過程

一、 系統規劃

(一) 系統運作流程

依循文獻探討後，為了提升校園安全而在物聯網和邊緣運算的人工智慧概念下，提出一套利用物件感測與物件辨識來進行學生碰擊預警系統，當學生行進過程通過偵測元件時，系統會進行判斷並做出相對應的動作，如圖 3-1 為本研究的整體流程架構，主要區分成三個不同的部分來說明：(1)偵測：紅外線熱感測器偵測到學生經過，由 NodeMCU 寫入之程式判斷是否達速度標準，未達速度標準則停止，達速度限制就把訊息傳至電腦主機；(2)分析：當電腦主機接受到 NodeMCU 所傳之訊息時，透過自編之 Python 程式在 OpenCV 函式庫分析網路攝影機所拍攝之影像；(3)執行：若分析出影像為多人以上則系統直接略過(Pass)，若為單人，則訊息回傳 NodeMCU 啟動告警音訊，並同步擷取 5 秒之影片，儲存至指定之網路硬碟，供學務處老師依 AD 權限存取。

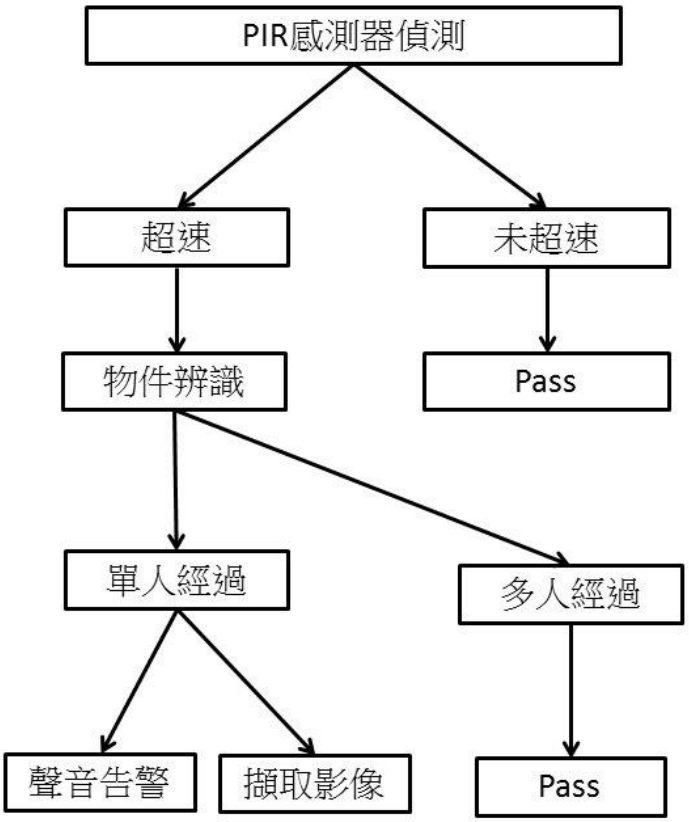

圖 3-1 系統運作流程圖

(二) 系統架構

學生碰撞通常有兩種情形，一種是走廊轉角處碰撞撞，一種是走廊直線奔跑和出教室的學生碰撞，或是直線反方向對撞，對於這兩種危險情形，設計出兩種設備配置(如圖 3-2、3-3)。

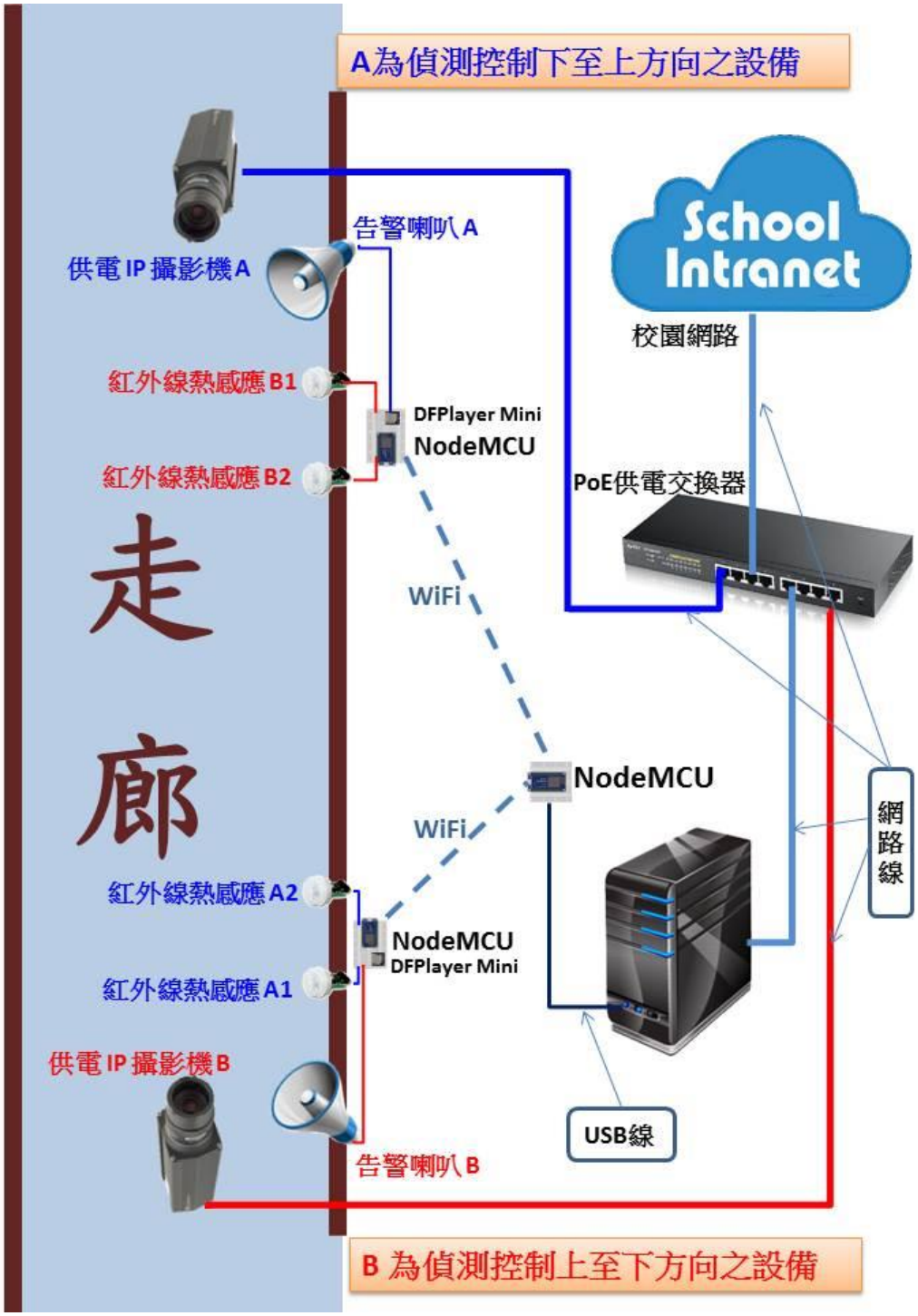

圖 3-2 系統架構圖(直行走廊)

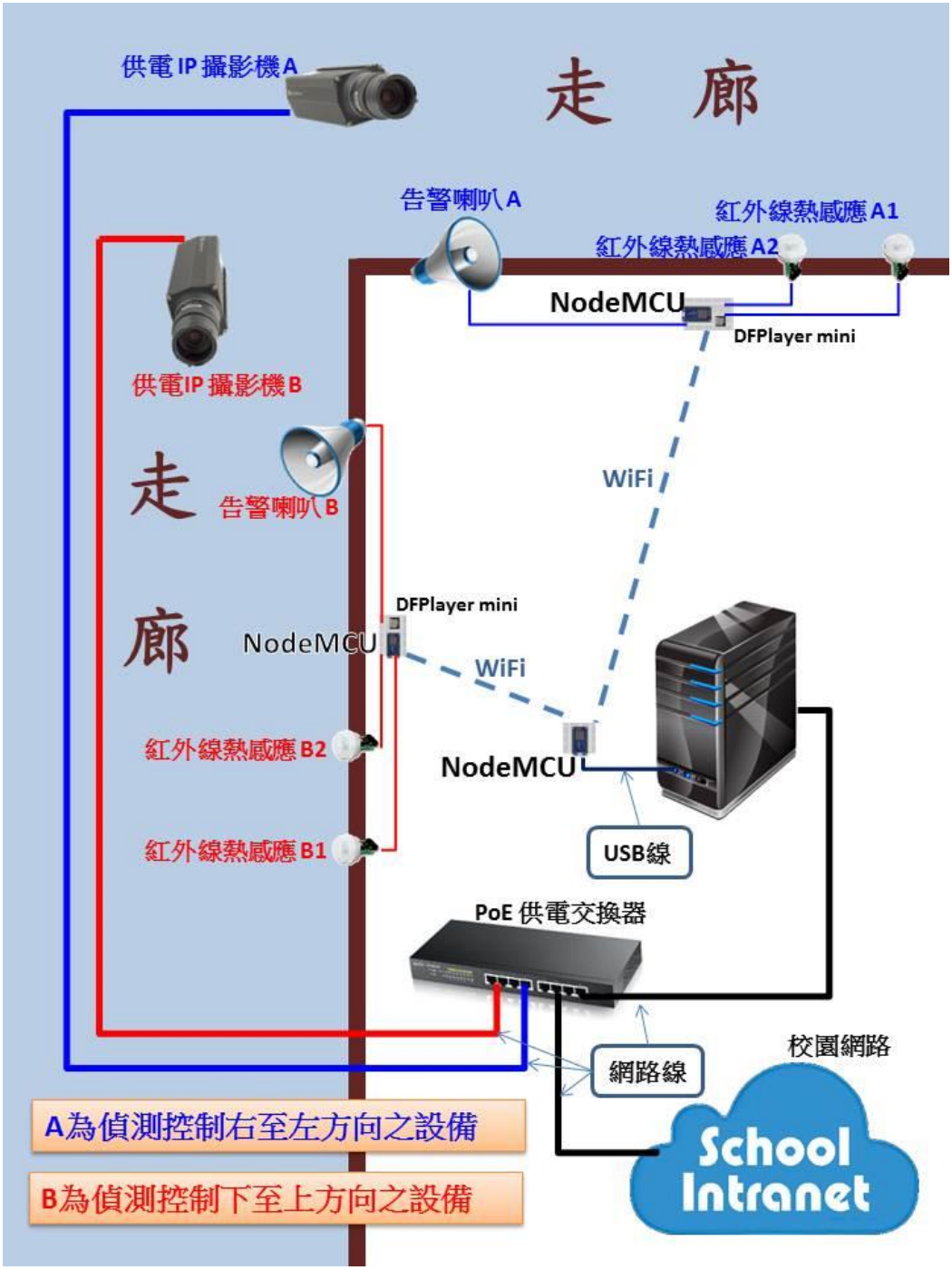

圖 3-3 系統架構圖(轉角)

(三) NodeMCU 電路架構

由兩個 NodeMCU 分別讀取 PIR，以時間差計算速度，並由其無線網路功能，將資料傳送至 Server 端 NodeMCU，Server 端的 NodeMCU 利用序列埠傳送至主機。

經主機 AI 人工智慧運算後，將訊號傳送至 Server 端 NodeMCU，再發送無線訊號傳回 Client 端的 NodeMCU，啟動 DFPlayer mini 音訊模組，最後由喇叭發出聲音告警。

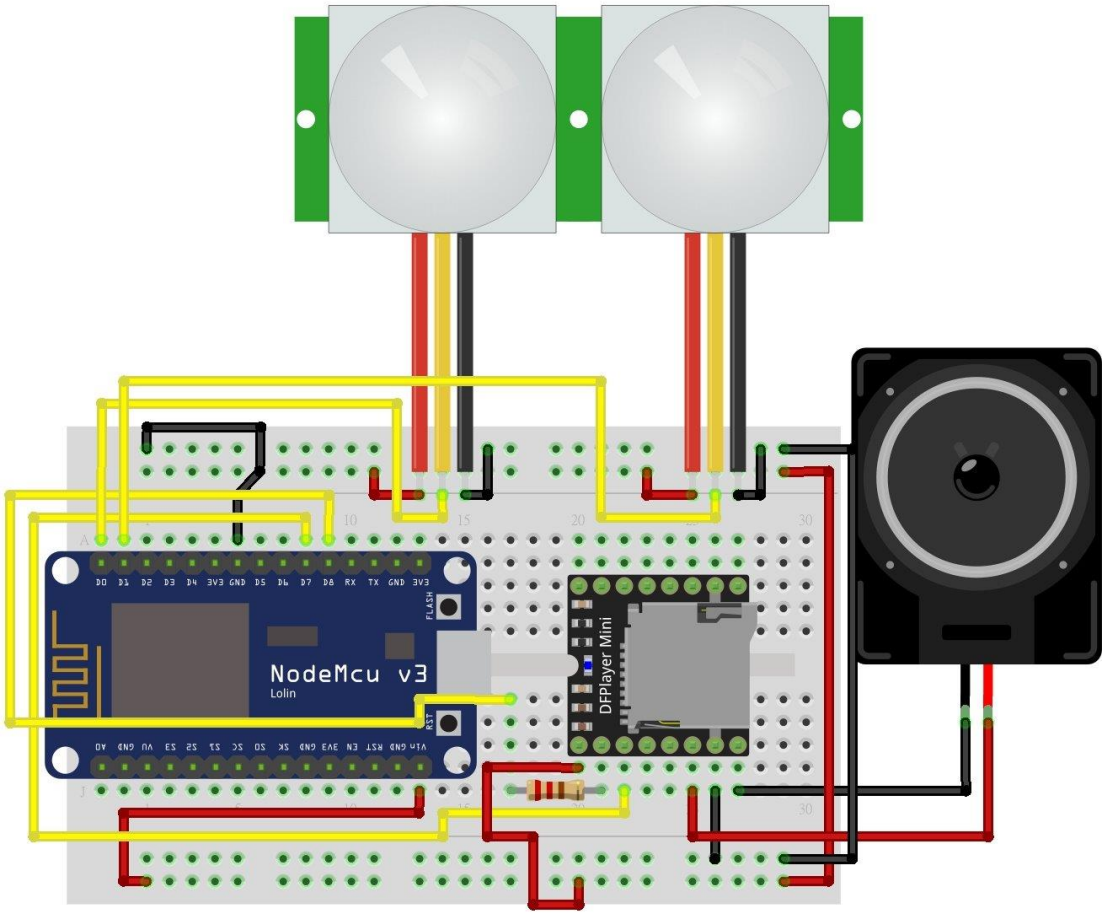

圖 3-4 NodeMCU 電路架構

二、 程式設計

(一) Arduino 部份

1. NodeMCU 接收 PIR 訊號並檢測速度

```
int PIR_A1_val = digitalRead(PIR_A1);
int PIR_A2_val = digitalRead(PIR_A2);
float time_original = 0;
if(PIR_A1_val == HIGH){
      A_used = true;
      while(PIR_A2_val != HIGH){
            time_original += 1;
            delay(1);
            PIR_A2_val = digitalRead(PIR_A2);
      }
time = time_original / 1000;
}
```

2. NodeMCU Server 端查看是否有接收到 Request

```
String request = client.readStringUntil('\r');
```

```
if (request != ""){
      Serial.println(request);
      while (1){
            String reply = Serial.readStringUntil('\n');
            if (reply != ""){
                  client.println(reply);
                  break;
            }
      }
     } client.flush();
```

3. NodeMCU Client 端傳送 Request

```
if(time <= 0.001){
      client.println("LR");
       client.flush();
}
```

4. NodeMCU Server 端啟動

```
WiFiServer server(80);
IPAddress ip(10, 241, 241, 27);
IPAddress gateway(10, 241, 241, 254);
IPAddress subnet(255, 255, 255, 0);
WiFi.config(ip, gateway, subnet);
WiFi.begin(ssid, pass);
while(WiFi.status() != WL_CONNECTED){
      delay(500);
} server.begin();
```

(二) Python 部份

1. 網路攝影機錄影

```
cap_1 = cv2.VideoCapture("rtsp://admin:Aa@123456@10.241.241.23")
cap_2 = cv2.VideoCapture("rtsp://admin:Aa@123456@10.241.241.24")
w = cap_1.get(cv2.CAP_PROP_FRAME_WIDTH)
h = cap_1.get(cv2.CAP_PROP_FRAME_HEIGHT)
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out_1 = cv2.VideoWriter(str("./Original_Video/A_CAM-" + time.strftime("%Y%m%d-
%H%M%S-", time.localtime()) + str(vid_count_I) + str(".avi")), fourcc, 20.0, (int(w),
int(h)))
vid_count_I += 1
out_2 = cv2.VideoWriter(str("./Original_Video/B_CAM-" + time.strftime("%Y%m%d-
%H%M%S-", time.localtime()) + str(vid_count_I) + str(".avi")), fourcc, 20.0, (640, 360))
vid_count_II += 1
```

2. 若偵測速度超過限制(向左)，執行之函式

```
def left_run():
      # grab the variables and make them global
      global detect_run
      global vid_count_I
      global vid_count_II
```

```
        global fourcc
        global out_1
        global out_2
        detect_run = True   # edit this boolean variable to let the threads pause
        print("Detect Running! (left)")
        # remove the files started with "B"
        dir_name = "./Original Video/"
        files = os.listdir(dir_name)
        for item in files:
            if item.startswith("B"):
                os.remove(os.path.join(dir_name, item))
        # get the video file of A CAM
        list_of_files = glob.glob("./Original Video/*")
        latest_file = max(list_of_files, key = os.path.getctime)
        # put the video file into the moviepy so it can trim
        capture_vid = VideoFileClip(latest_file)
        # declare the trim range seconds
        vid_len = capture_vid.duration
        cut_start = vid_len - 5
        # trim the video
        edited_vid = capture_vid.subclip(cut_start, vid_len)
        # save the file
        file_to_move = "D:/資優班/科展/程式碼/Back_End/video_capture/" +
latest_file[17:len(latest_file)]
        edited_vid.write_videofile(file_to_move, codec = "libx264")
        del capture_vid             # remove the variable
        shutil.rmtree("D:/資優班/科展/程式碼/Back_End/Original Video/")        # clean
the Original Video folder
        os.mkdir("D:/資優班/科展/程式碼/Back_End/Original Video/")        # create the
Original Video folder
        # create VideoWriters
        out_1 = cv2.VideoWriter(str("./Original Video/A_CAM-" + time.strftime("%Y%m%d-
%H%M%S-", time.localtime()) + str(vid_count_I) + str(".avi")), fourcc, 20.0, (320, 180))
        vid_count_I += 1
        out_2 = cv2.VideoWriter(str("./Original Video/B_CAM-" + time.strftime("%Y%m%d-
%H%M%S-", time.localtime()) + str(vid_count_I) + str(".avi")), fourcc, 20.0, (320, 180))
        vid_count_II += 1
        detect_run = False  # let the threads continue
```

3. 若偵測速度超過限制(向右)，執行之函式

```
def right_run():
        # grab the variables and make them global
        global detect_run
        global vid_count_I
        global vid_count_II
        global fourcc
        global out_1
```

```
        global out_2
        detect_run = True   # edit this boolean variable to let the threads pause
        print("Detect Running! (right)")
        # remove the files started with "B"
        dir_name = "./Original Video/"
        files = os.listdir(dir_name)
        for item in files:
            if item.startswith("A"):
                os.remove(os.path.join(dir_name, item))
        # get the video file of A CAM
        list_of_files = glob.glob("./Original Video/*")
        latest_file = max(list_of_files, key = os.path.getctime)
        # put the video file into the moviepy so it can trim
        capture_vid = VideoFileClip(latest_file)
        # declare the trim range seconds
        vid_len = capture_vid.duration
        cut_start = vid_len - 5
        edited_vid = capture_vid.subclip(cut_start, vid_len)
        # save the file
        file_to_move = "D:/資優班/科展/程式碼/Back_End/video_capture/" +
    latest_file[17:len(latest_file)]
        edited_vid.write_videofile(file_to_move, codec = "libx264")
        del capture_vid             # remove variable
        shutil.rmtree("D:/資優班/科展/程式碼/Back_End/Original Video/")         # clean
    the Original Video folder
        os.mkdir("D:/資優班/科展/程式碼/Back_End/Original Video/")         # create the
    Original Video folder
        # create VideoWriters
        out_1 = cv2.VideoWriter(str("./Original Video/A_CAM-" + time.strftime("%Y%m%d-
    %H%M%S-", time.localtime()) + str(vid_count_I) + str(".avi")), fourcc, 20.0, (320, 180))
        vid_count_I += 1
        out_2 = cv2.VideoWriter(str("./Original Video/B_CAM-" + time.strftime("%Y%m%d-
    %H%M%S-", time.localtime()) + str(vid_count_I) + str(".avi")), fourcc, 20.0, (320, 180))
        vid_count_II += 1
        detect_run = False          # let the threads continue
```

4. 讀取影像並將其寫入至影片檔案(雙 CPU 平行運算)

```
def read_1():   # declare the function to read frames of IP CAM I
        # let the variables become global
        global ret_1
        global frames_1
        global out_1
        while True:
            # if get signal of running, let the code pause
            if detect_run == True:
                time.sleep(2)
            else:# or else
                ret_1, frame_1 = cap_1.read()       # read ret, frame
```

```
                frame_1 = cv2.resize(frame_1, dsize = (320, 180))# resize
                frames_1.put(frame_1)  # put frames into the queue
                out_1.write(frame_1)     # write the frame into the VideoWriter
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

    def read_2():   # declare the function to read frames of IP CAM I
        # let the variables become global
        global ret_2
        global frames_2
        global out_2
        while True:
            # if get signal of running, let the code pause
            if detect_run == True:
                time.sleep(2)
            else:# or else
                ret_2, frame_2 = cap_2.read()       # read ret, frame
                frame_2 = cv2.resize(frame_2, dsize = (320, 180))# resize
                frames_2.put(frame_2)  # put frames into the queue
                out_2.write(frame_2)     # write the frame into the VideoWriter
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
```

5. 執行 OpenCV 模組之函式，臉部偵測及計算數量(AI 運算)

```
    def detect_1():             # declare the function of displaying video and doing face
    detection
        # grab the variable and make it global
        global people_1
        while True:
            # if get signal of running, let the code pause
            if detect_run == True:
                time.sleep(2)
            else:# or else
                if not frames_1.empty():# if the queue to put frames isn't empty
                    frame_1 = frames_1.get()       # get the frames from the queue
                    grey_1 = cv2.cvtColor(frame_1, cv2.COLOR_BGR2GRAY)      # convert
    the frame into gray
                    # declare the detectMultiScale for detecting faces
                    faces_1 = faceCascade_1.detectMultiScale(
                        grey_1,
                        scaleFactor = 1.1,
                        minNeighbors = 5,
                        minSize = (30, 30))
                    # draw rectangle around faces
                    for (x_1, y_1, w_1, h_1) in faces_1:
                        cv2.rectangle(frame_1, (x_1, w_1), (x_1 + w_1, y_1 + h_1), (0,
    255, 0), 2)
                    # the people count is the length of the detectMultiScale
```

```
                people_1 = len(faces_1)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

def detect_2():
    # grab the variable and make it global
    global people_2
    while True:
        # if get signal of running, let the code pause
        if detect_run == True:
            time.sleep(2)
        else:# or else
            if not frames_2.empty():# if the queue to put frames isn't empty
                frame_2 = frames_2.get()      # get the frames from the queue
                grey_2 = cv2.cvtColor(frame_2, cv2.COLOR_BGR2GRAY)      # convert
the frame into gray
                # declare the detectMultiScale for detecting faces
                faces_2 = faceCascade_2.detectMultiScale(
                    grey_2,
                    scaleFactor = 1.1,
                    minNeighbors = 5,
                    minSize = (30, 30))
                # draw rectangle around faces
                for (x_2, y_2, w_2, h_2) in faces_2:
                    cv2.rectangle(frame_2, (x_2, w_2), (x_2 + w_2, y_2 + h_2), (0,
255, 0), 2)
                # the people count is the length of the detectMultiScale
                people_2 = len(faces_2)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
```

6. 執行平行運算之函式

```
thread_1 = threading.Thread(target = read_1)
thread_2 = threading.Thread(target = read_2)
thread_3 = threading.Thread(target = detect_1)
thread_4 = threading.Thread(target = detect_2)
thread_1.start()
thread_2.start()
thread_3.start()
thread_4.start()
```

7. 讀取 NodeMCU Server 端傳送之資料並做記錄檔

```
dr = ser.readline()          # read what the NodeMCU says
dt = dr.decode()               # decode it
if dt == "LR\r\n":           # if the data is LR
all_times += 1          # all get signal times + 1
ser.write(people_1)  # write to the Serial port the people count
if people_1 == 1:      # if only one people in front of device
```

```
        triggered_1 += 1 # plus one to the variable of the triggered times (left)
        logging.info("Time: " + time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))
        # logging - time
        logging.info("Event: Device Triggered")
        # logging - event
        logging.info(str(people_1) + " People In Front Of Device")
        # logging - people
        logging.info("All Times: " + all_times)
        # logging - all times
        logging.info("Left Triggered Times: " + triggered_1)
        # logging - left triggered times
        logging.info("Right Triggered Times: " + triggered_2)
        # logging - right triggered times
        logging.info("Pause Times: " + pause_times)
        # logging - pause times
        logging.info("Ignore Times: " + ignore_times + "\n")
        # logging - ignore times
        # release the VideoWriters
        out_1.release()
        out_2.release()
        left_run()# run the function
        continue  # let the loop restart
    else:# if the people in front of the device is not only one
        pause_times += 1    # plus one to the variable of the pausing times
        logging.info("Time: " + time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))
        # logging - time
        logging.info("Event: Device Paused Triggering")
        # logging - event
        logging.info(str(people_1) + " People In Front Of Device")
        # logging - people
        logging.info("All Times: " + all_times)
        # logging - all times
        logging.info("Left Triggered Times: " + triggered_1)
        # logging - left triggered times
        logging.info("Right Triggered Times: " + triggered_2)
        # logging - right triggered times
        logging.info("Pause Times: " + pause_times)
        # logging - pause times
        logging.info("Ignore Times: " + ignore_times + "\n")
        # logging - ignore times
    # if the signal that recieved is right run
    elif dt == "RR\r\n":
    all_times += 1  # all get signal times + 1
    ser.write(people_2)# write to the Serial port the people count
    if people_2 == 1:     # if only one people in front of device
        triggered_2 += 1      # plus one to the variable of the running times (right)
        logging.info("Time: " + time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))
        # logging - time
```

```
            logging.info("Event: Device Triggered")
            # logging - event
            logging.info(str(people_1) + " People In Front Of Device")
            # logging - people
            logging.info("All Times: " + all_times)
            # logging - all times
            logging.info("Left Triggered Times: " + triggered_1)
            # logging - left triggered times
            logging.info("Right Triggered Times: " + triggered_2)
            # logging - right triggered times
            logging.info("Pause Times: " + pause_times)
            # logging - pause times
            logging.info("Ignore Times: " + ignore_times + "\n")
            # logging - ignore times
            # release the VideoWriters
            out_1.release()
            out_2.release()
            right_run()             # run the function
            continue        # restart the loop
        else:# if the people in front of the device is not only one
            pause_times += 1    # plus one to the variable of pausing times
            logging.info("Time: " + time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))
            # logging - time
            logging.info("Event: Device Paused Triggering")
            # logging - event
            logging.info(str(people_1) + " People In Front Of Device")
            # logging - people
            logging.info("All Times: " + all_times)
            # logging - all times
            logging.info("Left Triggered Times: " + triggered_1)
            # logging - left triggered times
            logging.info("Right Triggered Times: " + triggered_2)
            # logging - right triggered times
            logging.info("Pause Times: " + pause_times)
            # logging - pause times
            logging.info("Ignore Times: " + ignore_times + "\n")
```

三、 系統建置與環境測試

(一) 前端架設

1. 紅外線感測器

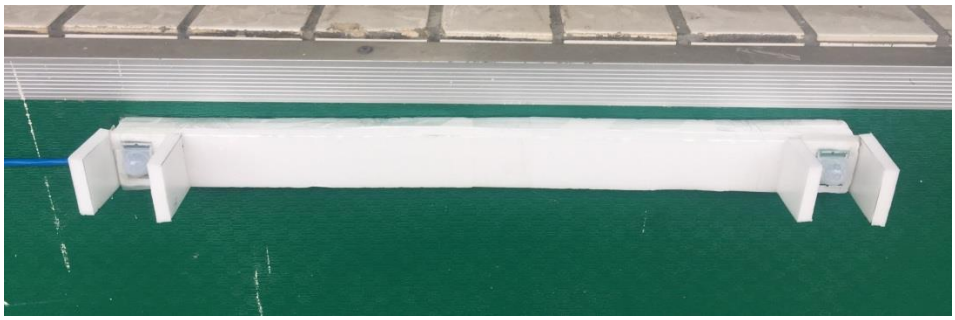

圖 3-5 紅外線感測器部份

2. NodeMCU、DFPlayer Mini、降壓模組、電源供應、喇叭

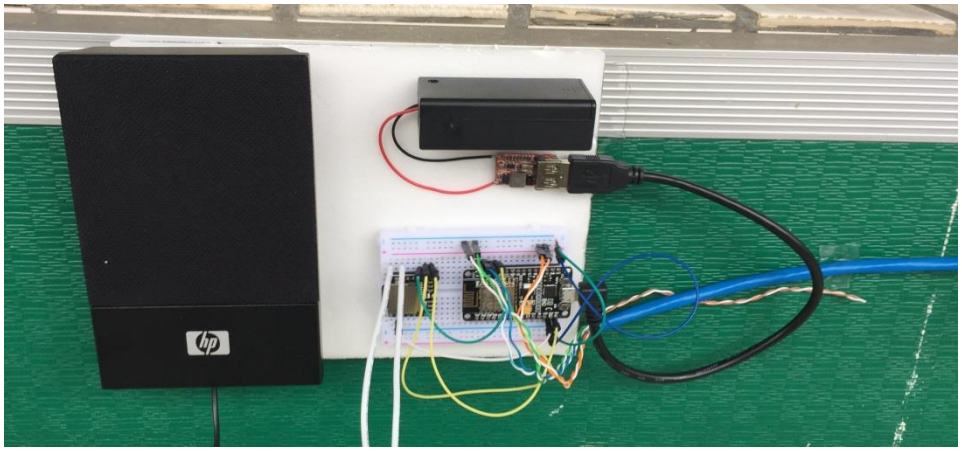

圖 3-6 前端運算部份

3. 前端整體

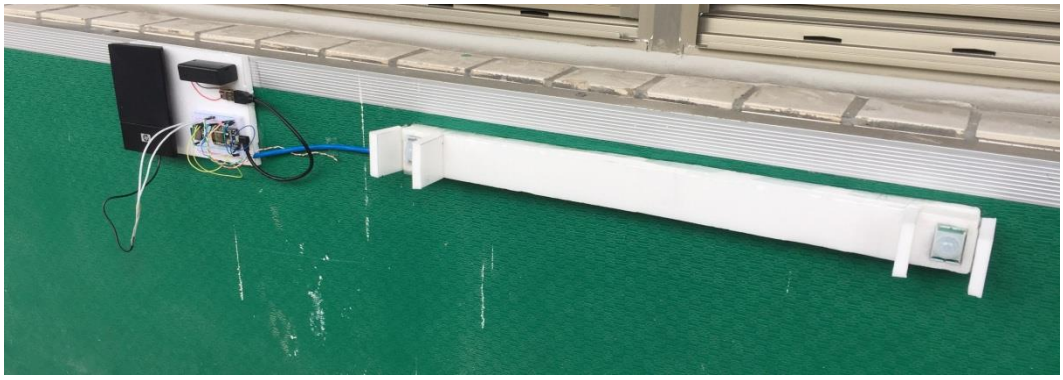

圖 3-7 前端運算模組

(二) PoE 攝影機架設

圖 3-8 攝影機位置

(三) 完成走廊環境架設

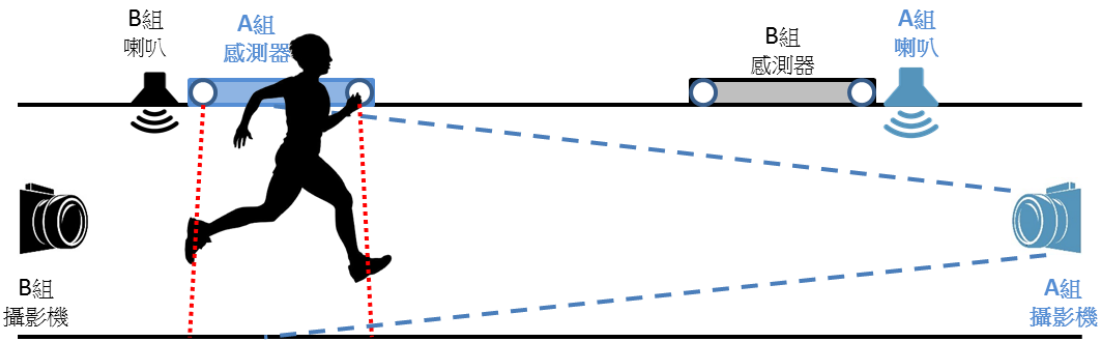

圖 3-9 從左往右方向偵測示意圖

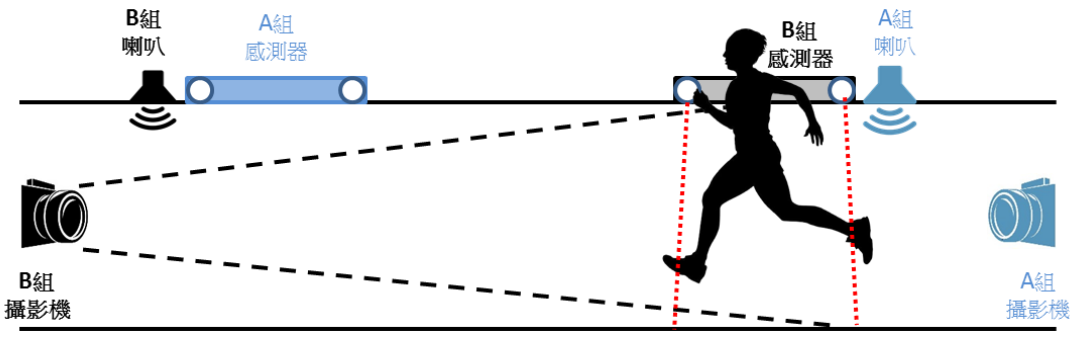

圖 3-10 從右往左方向偵測示意圖

圖 3-11 實際走廊環境圖

(四) 測試與偵錯

圖 3-12 實際測試圖

肆、 研究結果與分析

在系統實測中，對本研究的整體運行從下列這幾點來進行評估：(1)當單人經過且超過速度限制時，能否正確被偵測出；(2)系統能否正確發出告警聲音；(3)當多人經過時系統能否正確判斷忽略；最後(4)告警同時系統是否擷取 5 秒影片存放於檔案伺服器。

一、 混淆矩陣評估

在系統實測的結果部分，在資優班教室走廊共蒐集了 103 筆數據來做系統評估，如表 4-1 受到紅外線熱感測器延遲的影響，還是有連續跑步通過的情形不能成功偵測並發出警報。而在受干擾的部分，雖然理論上紅外線熱感測器只對人體體溫而產生的 10 微米左右的紅外線通過菲泥爾濾光片增強後聚集到紅外感應源上，但在本研究實測中，因為溫度、日照、樹影搖動對環境的影響劇烈而有產生誤判的可能。最後，從表 4-2 中，本研究所發展的校園碰撞預警機制整體性能指標都在 0.72 以上。

表 4-1 系統告警測試結果

	正類-有異常狀態 (應發出告警)	負類-無異常狀態 (不應發出告警)
正類 系統有發出告警	13	2
負類 系統沒有發出告警	5	83

表 4-2 系統整體性能指標

	正確率	敏感度	精確度	特異度	F1 Score
指標數值	0.93	0.72	0.87	0.98	0.79

二、 影像擷取紀錄

擷取之影像儲存於網路磁碟，供學務處老師依 AD 權限存取(如圖 4-1、4-2)。

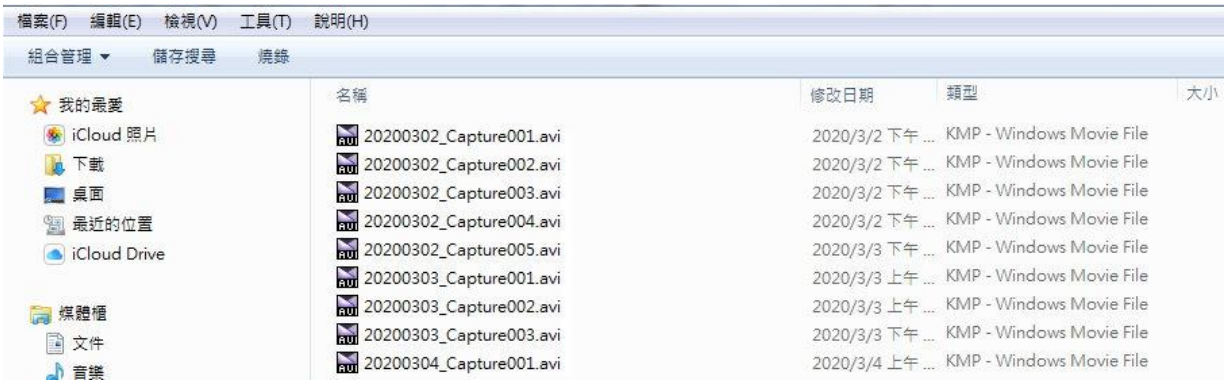

圖 4-1 影像資料夾

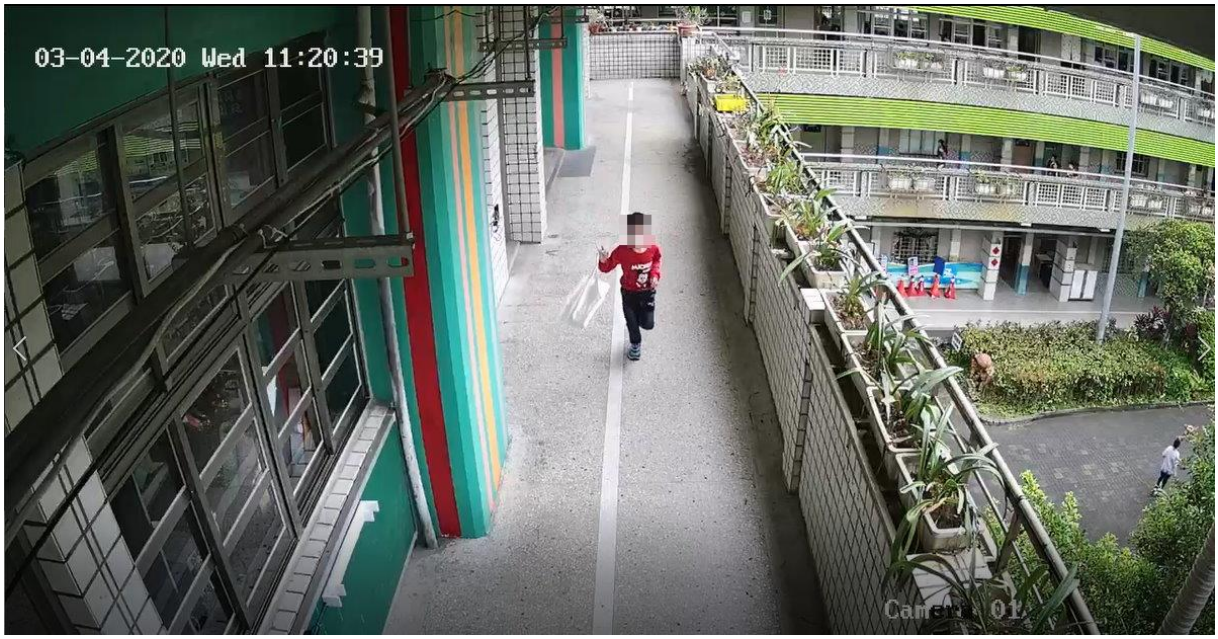

圖 4-2 播放影像擷圖

伍、 結論與建議

本研究依序從文獻探討來收集符合研究動機與目的之技術理論與概念、硬體設備以及軟體開發環境，再根據第三章所規劃的方法來進行系統建置，並透過第四章所進行的實測結果對系統作分析評估，所得結論與建議如下：

一、 結論

(一) 系統實用性：

本研究所提出的校園碰撞預警機制，是利用紅外線熱感應器偵測人體移動，透過物聯網元件 NodeMCU 來分析移動速度，藉由邊緣運算概念先由前端元件處理部份資料後，以無線網路傳至主機，並搭配 AI 人工智慧技術，應用 OpenCV 函式庫運算物件辨識，最後對在走廊上奔跑的學生發出告警，並擷取奔跑影像，以達到維護校園安全的目的。以實測之評估數據來看，確實可行。

(二) 成本因素：

本系統可拆開分成前端偵測和後端辨識與錄影，若僅裝設前端偵測告警，成本僅約數百元，可放置校園多處易發生碰撞的地方，但是缺乏後台 AI 人工智慧之物件辨識和錄影，且易發生誤判之情形。但是如果架設整套系統，還需要網路攝影機和電腦主機，費用較高，且需要考量電路和資訊線路。

(三) 破壞問題：

系統在建置和實測時，就有不少學生因為好奇，經過時就碰一下，也有故意掰斷保麗龍夾版和扯出紅外線熱感應器的情形，在啓動錄影功能且張貼告示，並告誡破壞的學生後，破壞情形已改善。

二、 建議

(一) 技術層面

雖然實測數據顯示本系統性能指標都在可接受範圍，但紅外線熱感應器仍有延遲、溫度、日照、樹影干擾等影響判讀因素，日後研究方向可以用其他深度學習的演算法，如 YOLO、 R-CNN 等技術，直接以影像分析物件移動，自動啟動並計算

速度，就可捨棄前端設備，只留網路攝影機，即可運作相同功能，而且更準確。

(二) 教育層面

本系統建置的目的是要提醒學生在走廊不要奔跑，來避免遭受傷害，但這是輔助的告警機制，校園安全還是要每一個學生自發性的不要在走廊上奔跑，就像校長說的，要回歸教育的本質，我們在學校除了學業的學習，也要聽從師長在生活上的教導，來保護自己和別人的安全。

陸、 參考文獻

Ahmed, K., Hasaneen, E.-S., & Orabi, M. (2018). Power management system for Ethernet-based IoT devices. *Ain Shams Engineering Journal, 9*(4), 3033-3043. doi:10.1016/j.asej.2017.11.007

Amann, S., Proksch, S., Nadi, S., & Mezini, M. (2016, 14-18 March 2016). *A Study of Visual Studio Usage in Practice.* Paper presented at the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER).

Arunkumar, K. A., Kriti, K., Das, A., & Bhattacharyya, B. (2019, 30-31 March 2019). *Wireless Speakers using WiFi and IoT.* Paper presented at the 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN).

Kaplan, V. (2018). Writing Your Own Debugger and Language Extensions with Visual Studio Code. *CODE Magazine*, 42-56.

Li, S., Xu, L. D., & Zhao, S. (2015). The internet of things: a survey. *Information Systems Frontiers, 17*(2), 243-259. doi:10.1007/s10796-014-9492-7

Liu, X. (2017). *Open-source low-cost internet of things platform for buildings.* (M.S.), The Florida State University, Ann Arbor.

Lopez, P. G., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., . . . Riviere, E. (2015). *Edge-centric Computing: Vision and Challenges* (Vol. 45): Association for Computing Machinery.

Noble, F. K. (2016, 28-30 Nov. 2016). *Comparison of OpenCV's feature detectors and feature matchers.* Paper presented at the 2016 23rd International Conference on Mechatronics and Machine Vision in Practice (M2VIP).

Tharwat, A. (2018). Classification assessment methods. *Applied Computing and Informatics*. doi:10.1016/j.aci.2018.08.003

Waleed, A.-G., Sathish Kumar, S., Raed, A., & Chandrasekharan, N. (2018). Smart Tree Care System with Internet of Things. *Research Journal of Applied Sciences, Engineering and Technology, 15*(9), 328-336. doi:10.19026/rjaset.15.5923

Yun, J., & Lee, S.-S. (2014). Human Movement Detection and Identification Using Pyroelectric Infrared Sensors. *Sensors, 14*(5). doi:10.3390/s140508057

梁依儒. (2016). *利用交通速度偵測數據於高速公路塞車行為之研究.* (碩士), 國立中興大學, 台中市.

蘇孟宗, & 陳右怡. (2018). 人工智慧驅策臺灣產業跨域創新. *國土及公共治理季刊, 6*(4), 40-49.

柒、 附錄

一、 Arduino 部分

```
#include <SPI.h>
#include <ESP8266WiFi.h>
#define led LED_BUILTIN
char ssid[] = "AP";
char pass[] = "Science_Fair";
WiFiServer server(80);
WiFiClient client;
IPAddress ip(10, 241, 241, 27);
IPAddress gateway(10, 241, 241, 254);
IPAddress subnet(255, 255, 255, 0);
int i;
char reply_char[8];

void setup()
{
  Serial.begin(9600);
  WiFi.mode(WIFI_STA);
  WiFi.config(ip, gateway, subnet);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(100);
  }
  Serial.println("Connected to WiFi");
  server.begin();
  pinMode(led, OUTPUT);
}

void loop()
{
  client = server.available();
  client.setTimeout(100);
  Serial.setTimeout(100);
  if (client)
  {

    Serial.println("Available");
    if (client.connected())
    {
      Serial.println("Connected");
    }

    while (client.connected())
    {
      // Serial.println(".");

      String request =
client.readStringUntil('e');
      if (request == "LR")
      {
        digitalWrite(led, LOW);
        Serial.println("LR");
        for (i = 0; i < 3; i++)
        {
          String reply =
Serial.readStringUntil('e');
          if (reply == "WARN B")
          {
            reply.toCharArray(reply_char,
8);
            client.printf("%s%s",
reply_char, "e");
            break;
          }
          yield();
        }
      }
      else if (request == "RR")
      {
        Serial.println("RR");
        for (i = 0; i < 3; i++)
        {
          String reply =
Serial.readStringUntil('e');
          if (reply == "WARN A")
          {
            reply.toCharArray(reply_char,
8);
            client.printf("%s%s",
reply_char, "e");
            break;
          }
          yield();
        }
        yield();
      }
      yield();
    }
  }
  Serial.println("");
  client.stop();
}
```

```
#include <SPI.h>
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>
#include <DFPlayer_Mini_Mp3.h>
#define PIR_A D3
#define PIR_B D4
#define led LED_BUILTIN
char ssid[] = "AP";
char pass[] = "Science_Fair";
SoftwareSerial DFPlayer_Mini(D1, D2);
WiFiClient client;
IPAddress server(10, 241, 241, 27);
IPAddress ip(10, 241, 241, 28);
IPAddress gateway(10, 241, 241, 254);
IPAddress subnet(255, 255, 255, 0);

void setup()
{
    pinMode(PIR_A, INPUT);
    pinMode(PIR_B, INPUT);
    pinMode(led, OUTPUT);
    DFPlayer_Mini.begin(9600);
    mp3_set_serial(DFPlayer_Mini);
    mp3_set_volume(30);
    WiFi.mode(WIFI_STA);
    WiFi.config(ip, gateway, subnet);
    WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(100);
    }
    Serial.begin(9600);
}

void loop()
{
    int i;
    unsigned long time = 0;
    int PIR_A_VAL = digitalRead(PIR_A);
    int PIR_B_VAL = digitalRead(PIR_B);
    client.connect(server, 80);
    client.setTimeout(100);
    String warnStatus =
client.readStringUntil('e');
    if (PIR_A_VAL == HIGH)
    {
        while (1)
        {
            time += 1;
            delay(1);
            if (time >= 1000)
            {
                Serial.println("OFD");
                for (i = 0; i < 35; i++)
                {
                    delay(100);
                    yield();
                }
                break;
            }
            else if (PIR_B_VAL == HIGH)
            {
                if (time == 1)
                {
                    client.print("LRe");
                    for (i = 0; i < 35; i++)
                    {
                        delay(100);
                        yield();
                    }
                    break;
                }
                break;
            }
            PIR_B_VAL = digitalRead(PIR_B);
            yield();
        }
    }
    else if (PIR_B_VAL == HIGH)
    {
        Serial.println("B Triggered");
        for (i = 0; i < 35; i++)
        {
            delay(100);
            yield();
        }
    }
    else if (warnStatus == "WARN A")
    {
        mp3_play();
    }
    Serial.println("OFL");
    //  client.stop();
}
```

二、 Python 部分

```
import serial
import time
from moviepy.editor import VideoFileClip
import cv2
import os
import glob
import threading
import shutil
import queue
import logging

port = "COM8"
baud = 9600
ser = serial.Serial(port, baud)
vid_count_I = 1
vid_count_II = 1
cascPath = "haarcascade_frontalface_default.xml"
faceCascade_1 = cv2.CascadeClassifier(cascPath)
faceCascade_2 = cv2.CascadeClassifier(cascPath)
detect_run = False
logging.basicConfig(level = logging.INFO,
                            format = "%(message)s",
                            datefmt = "%Y-%m-%d
%H:%M",
                            handlers =
[logging.FileHandler("logging.log", "w", "utf-8")])
cap_1 =
cv2.VideoCapture("rtsp://admin:Aa@123456@10.241.241.
23")
cap_2 =
cv2.VideoCapture("rtsp://admin:Aa@123456@10.241.241.
24")
w = cap_1.get(cv2.CAP_PROP_FRAME_WIDTH)
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out_1 = cv2.VideoWriter(str("./Original Video/A_CAM-
" + time.strftime("%Y%m%d-%H%M%S-",
time.localtime()) + str(vid_count_I) + ".avi"),
fourcc, 20.0, (320, 180))
vid_count_I += 1
out_2 = cv2.VideoWriter(str("./Original Video/B_CAM-
" + time.strftime("%Y%m%d-%H%M%S-",
time.localtime()) + str(vid_count_I) + ".avi"),
fourcc, 20.0, (320, 180))
vid_count_II += 1
ret_1, frame_1 = cap_1.read()
ret_2, frame_2 = cap_2.read()
frames_1 = queue.Queue()
frames_2 = queue.Queue()
people_1 = 0
people_2 = 0
all_times = 0
triggered_1 = 0
triggered_2 = 0
pause_times = 0
ignore_times = 0

def read_1():
      global ret_1
      global frames_1
      global out_1
      while True:
            if detect_run == True:
                  time.sleep(2)
            else:
                  ret_1, frame_1 = cap_1.read()
                  frame_1 = cv2.resize(frame_1, dsize
= (320, 180))
                  frames_1.put(frame_1)
                  out_1.write(frame_1)
            if cv2.waitKey(1) & 0xFF ==
ord('q'):
                  break

def read_2():
      global ret_2
      global frames_2
      global out_2
      while True:
            if detect_run == True:
                  time.sleep(2)
            else:
                  ret_2, frame_2 = cap_2.read()

                  frame_2 = cv2.resize(frame_2,
dsize = (320, 180))
                  frames_2.put(frame_2)
                  out_2.write(frame_2)
            if cv2.waitKey(1) & 0xFF ==
ord('q'):
                  break

def detect_1():
      global people_1
      while True:
            if detect_run == True:
                  time.sleep(2)
            else:
                  if not frames_1.empty():
                        frame_1 = frames_1.get()
                        grey_1 =
cv2.cvtColor(frame_1, cv2.COLOR_BGR2GRAY)
                        faces_1 =
faceCascade_1.detectMultiScale(
                              grey_1,
                              scaleFactor = 1.1,
                              minNeighbors = 5,
                              minSize = (30, 30))
                        for (x_1, y_1, w_1, h_1)
in faces_1:

      cv2.rectangle(frame_1, (x_1, w_1), (x_1 +
w_1, y_1 + h_1), (0, 255, 0), 2)
                        people_1 = len(faces_1)
            if cv2.waitKey(1) & 0xFF ==
ord('q'):
                  break

def detect_2():
      global people_2
      while True:
            if detect_run == True:
                  time.sleep(2)
            else:
                  if not frames_2.empty():
                        frame_2 = frames_2.get()
                        grey_2 =
cv2.cvtColor(frame_2, cv2.COLOR_BGR2GRAY)
                        faces_2 =
faceCascade_2.detectMultiScale(
                              grey_2,
                              scaleFactor = 1.1,
                              minNeighbors = 5,
                              minSize = (30, 30))
                        for (x_2, y_2, w_2, h_2)
in faces_2:

      cv2.rectangle(frame_2, (x_2, w_2), (x_2 +
w_2, y_2 + h_2), (0, 255, 0), 2)
                        people_2 = len(faces_2)
```

```
                break
thread_1 = threading.Thread(target = read_1)
thread_2 = threading.Thread(target = read_2)
thread_3 = threading.Thread(target = detect_1)
thread_4 = threading.Thread(target = detect_2)
thread_1.start()
thread_2.start()
thread_3.start()
thread_4.start()
print("Started threads")
def left_run():
      global detect_run
      global vid_count_I
      global vid_count_II
      global fourcc
      global out_1
      global out_2
      detect_run = True
      print("Detect Running! (left)")
      dir_name = "./Original Video/"
      files = os.listdir(dir_name)
      for item in files:
            if item.startswith("B"):
                  os.remove(os.path.join(dir_name,
item))
      list_of_files = glob.glob("./Original
Video/*")
      latest_file = max(list_of_files, key =
os.path.getctime)
      capture_vid = VideoFileClip(latest_file)
      vid_len = capture_vid.duration
      cut_start = vid_len - 5
      edited_vid = capture_vid.subclip(cut_start,
vid_len)
      file_to_move = "D:/資優班/科展/程式碼
/Back_End/video_capture/" +
latest_file[17:len(latest_file)]
      edited_vid.write_videofile(file_to_move, codec
= "libx264")
      del capture_vid
      shutil.rmtree("D:/資優班/科展/程式碼
/Back_End/Original Video/")
      os.mkdir("D:/資優班/科展/程式碼
/Back_End/Original Video/")
      out_1 = cv2.VideoWriter(str("./Original
Video/A_CAM-" + time.strftime("%Y%m%d-%H%M%S-",
time.localtime()) + str(vid_count_I) + str(".avi")),
fourcc, 20.0, (320, 180))
      vid_count_I += 1
      out_2 = cv2.VideoWriter(str("./Original
Video/B_CAM-" + time.strftime("%Y%m%d-%H%M%S-",
time.localtime()) + str(vid_count_I) + str(".avi")),
fourcc, 20.0, (320, 180))
      vid_count_II += 1
      detect_run = False
def right_run():
      global detect_run
      global vid_count_I
      global vid_count_II
      global fourcc
      global out_1
      global out_2
      detect_run = True
      print("Detect Running! (right)")
      dir_name = "./Original Video/"
      files = os.listdir(dir_name)
      for item in files:
            if item.startswith("A"):
                  os.remove(os.path.join(dir_name,
item))
      list_of_files = glob.glob("./Original
Video/*")
      latest_file = max(list_of_files, key =
os.path.getctime)
      capture_vid = VideoFileClip(latest_file)
      vid_len = capture_vid.duration
      cut_start = vid_len - 5
      edited_vid =
capture_vid.subclip(cut_start, vid_len)
      file_to_move = "D:/資優班/科展/程式碼
/Back_End/video_capture/" +
latest_file[17:len(latest_file)]
      edited_vid.write_videofile(file_to_move,
codec = "libx264")
      del capture_vid
      shutil.rmtree("D:/資優班/科展/程式碼
/Back_End/Original Video/")
      os.mkdir("D:/資優班/科展/程式碼
/Back_End/Original Video/")
      out_1 = cv2.VideoWriter(str("./Original
Video/A_CAM-" + time.strftime("%Y%m%d-%H%M%S-",
time.localtime()) + str(vid_count_I) +
str(".avi")), fourcc, 20.0, (320, 180))
      vid_count_I += 1
      out_2 = cv2.VideoWriter(str("./Original
Video/B_CAM-" + time.strftime("%Y%m%d-%H%M%S-",
time.localtime()) + str(vid_count_I) +
str(".avi")), fourcc, 20.0, (320, 180))
      vid_count_II += 1
      detect_run = False

try:
      latest_file = max(list_of_files, key =
os.path.getctime)
      capture_vid = VideoFileClip(latest_file)
      vid_len = capture_vid.duration
      cut_start = vid_len - 5
      edited_vid =
capture_vid.subclip(cut_start, vid_len)
      file_to_move = "D:/資優班/科展/程式碼
/Back_End/video_capture/" +
latest_file[17:len(latest_file)]
      edited_vid.write_videofile(file_to_move,
codec = "libx264")
      del capture_vid
      shutil.rmtree("D:/資優班/科展/程式碼
/Back_End/Original Video/")
      os.mkdir("D:/資優班/科展/程式碼
/Back_End/Original Video/")
      out_1 = cv2.VideoWriter(str("./Original
Video/A_CAM-" + time.strftime("%Y%m%d-%H%M%S-",
time.localtime()) + str(vid_count_I) +
str(".avi")), fourcc, 20.0, (320, 180))
      vid_count_I += 1
      out_2 = cv2.VideoWriter(str("./Original
Video/B_CAM-" + time.strftime("%Y%m%d-%H%M%S-",
time.localtime()) + str(vid_count_I) +
str(".avi")), fourcc, 20.0, (320, 180))
      vid_count_II += 1
      detect_run = False

try:
      while True:
            while ser.in_waiting and
cap_1.isOpened() and cap_2.isOpened():
                  if ret_1 == True and ret_2 ==
True:
                        if cv2.waitKey(1) & 0xFF
== ord('q'):
                              break
                        dr = ser.readline()
                        dt = dr.decode()
                        if dt == "LR\r\n":
                              all_times += 1
                              if people_1 == 1:
```

```
                                    triggered_1 += 1
                                    ser.write("WARN
Be")
                                    logging.info("Time:
" + time.strftime("%Y-%m-%d %H:%M:%S",
time.localtime()))

     logging.info("Event: Device Triggered")

     logging.info(str(people_1) + " People In Front
Of Device")
                                    logging.info("All
Times: " + all_times)
                                    logging.info("Left
Triggered Times: " + triggered_1)
                                    logging.info("Right
Triggered Times: " + triggered_2)
                                    logging.info("Pause
Times: " + pause_times)
                                    out_1.release()
                                    out_2.release()
                                    left_run()
                                    continue
                               else:
                                    pause_times += 1
                                    logging.info("Time:
" + time.strftime("%Y-%m-%d %H:%M:%S",
time.localtime()))

     logging.info("Event: Device Paused
Triggering")

     logging.info(str(people_1) + " People In Front
Of Device")
                                    logging.info("All
Times: " + all_times)
                                    logging.info("Left
Triggered Times: " + triggered_1)
                    logging.info("Right Triggered
Times: " + triggered_2)
                                    logging.info("Pause
Times: " + pause_times)
                          elif dt == "RR\r\n":
                                    all_times += 1
                                    if people_2 == 1:
                                    triggered_2 += 1
                                    ser.write("WARN
Ae")
                                    logging.info("Time:
" + time.strftime("%Y-%m-%d %H:%M:%S",
time.localtime()))

     logging.info("Event: Device Triggered")

     logging.info(str(people_1) + " People In Front
Of Device")
                                    logging.info("All
Times: " + all_times)
                                    logging.info("Left
Triggered Times: " + triggered_1)
                                    logging.info("Right
Triggered Times: " + triggered_2)
                                    logging.info("Pause
Times: " + pause_times)
                                    out_1.release()
                                    out_2.release()
                                    right_run()
                                    continue
                    else:
                                    pause_times += 1
                                    logging.info("Time:
" + time.strftime("%Y-%m-%d %H:%M:%S",
time.localtime()))

     logging.info("Event: Device Paused
Triggering")

     logging.info(str(people_1) + " People In
Front Of Device")

     logging.info("All Times: " + all_times)

     logging.info("Left Triggered Times: " +
triggered_1)

     logging.info("Right Triggered Times: " +
triggered_2)

     logging.info("Pause Times: " +
pause_times)
                          else:
                               print("Cound not
recognise data, please make sure if you've made
a typo")
except KeyboardInterrupt:
    print("KeyboardInterrupt")
ser.close()
cap_1.release()
cap_2.release()
out_1.release()
out_2.release()
cv2.destroyAllWindows()
```