中華民國第 64 屆中小學科學展覽會

作品說明書

國中組　生活與應用科學(一)科

佳作

032813

行人警示系統之開發與研究

學校名稱： 高雄市立文府國民中學

作者：	指導老師：
國二 高正龍 國二 邱俊瑋 國二 彭冠綸	郭懿尹

關鍵詞： Yolov8、OpenCV、Python

摘要

台灣近年來，行人與汽機車的交通事故頻傳，我們開始思考是否可以開發出一套行人警示系統，來輔助駕駛了解前方路況，即時提醒駕駛需禮讓行人，同時告知後方來車前方路況，減少追撞事故。我們使用 Yolov8 進行斑馬線與行人辨識模型的訓練，Python 程式執行模型進行物件偵測，Supervision 進行區域內物件數量的統計。經由條件設定，判斷車輛前方是否有行人或是斑馬線，若車輛前方有行人，將於影像畫面中顯示警示標語，發出警示語音提醒車內駕駛，降低汽車與行人的事故發生機率，同時送出藍芽訊號使警示燈顯示動態警示燈號，提醒後方駕駛前方路況，降低前車因禮讓行人而被後車追撞的機會。我們已經成功開發出『行人警示系統』，並可成功上路使用。

關鍵字：Yolov8、OpenCV、Python

壹、研究動機

一、研究動機與背景

這幾年以來，每次打開新聞台，常常看到行人與汽機車發生事故的新聞，且往往都是令人傷心的結局。而行人行經斑馬線上穿越馬路時，也常面臨著駕駛未停車禮讓的情況，使得行人行走於斑馬線上的過程險象環生，因此台灣獲得了”行人地獄”這樣不名譽的封號。而台灣大多的汽機車駕駛都是遵守交通規則的用路人，但因少部分的不良駕駛(酒駕、超速等)或是駕駛注意力不集中(恍神、看手機、撿物品等)而造成的疏忽，就會導致悲劇的發生。根據交通部路政及道安司資訊查詢網的資料顯示，全台灣在112年1月至12月於路口發生事故之案件數量為9422件，死傷人數為9212人，其中死亡人數為193人，如表1-1，這表示台灣幾乎每天都會發生行人與汽機車的事故，且幾乎都會造成傷亡。

表1-1

112 年全國行人路口事故件數各年齡層分布(依件數)

排序	運具別	件數	死亡人數	受傷人數	死傷人數
1	成年人(25-64 歲)	4,476	45	4,344	4,389
2	高齡者(65 歲以上)	3,262	143	3,024	3,167
3	年輕人(18-24 歲)	635	2	628	630
4	兒童(0-12 歲)	560	2	558	560
5	少年(13-17 歲)	441	1	432	433
6	不明	48	0	33	33
	總計	9422	193	9019	9212
備註：不明係指案件中當事者資料無身份證號及出生日期。					

註：道安資訊查詢網，交通部路政及道安司，2023(https://roadsafety.tw)

圖1-1 「什麼是停讓行人」6/30新制上路懶人包

註：內政部警政署網站，2023(https://www.npa.gov.tw)

而政府為降低上述事故的發生機率，進行相關道路交通安全規則條例的修改，如圖1-1，並於112年6月30日開始加強執法，其中停讓行人是指車輛須在完全停止的狀態下禮讓行人通過斑馬線，雖然保障了行人通過斑馬線的安全，但往往後方的駕駛並無法了解前車禮讓行人的情況，進而對前車按壓喇叭(圖1-2)或是加速超車，或因無法了解前車的動態，而無保持適當的行車距離，進而導致後車碰撞前車的事故(圖1-3)，所以禮讓行人而被追撞的事故也是經常發生，因此停讓行人的行為，雖然保障了行人過馬路的安全，但同時也增加了被追撞的風險。

圖1-2 禮讓行人被按喇叭催促

註：東森新聞，2023

圖1-3 禮讓行人遭追撞

註：TVBS新聞，2023

根據上述的狀況，我們開始思考是否可以透過電腦視覺的概念開發出一套行人警示系統，來輔助駕駛了解前方路口的行人狀況，並即時提醒駕駛，降低因駕駛注意力不集中而發生交通事故的機率，同時也能告知後方來車，目前前方路口是否有行人穿越馬路，減少前車被後車追撞的機會，讓整個禮讓行人的過程能夠更加的安全。因此進行相關研究並開發出含有上述概念的系統，並可運用於生活中，就是我們進行這次研究的最大動機。

貳、研究目的

為了開發出可以及時提醒駕駛前方路口行人狀況的輔助裝置，並可以同時將路口行人狀況告知後方來車的警示系統，我們將這樣的設計想法分成兩大部分進行討論，分別為：

1. 行人辨識系統設計　2. 後方警示系統設計

首先，我們針對行人辨識系統設計的部分與老師進行了討論，了解到可以使用物件偵測技術來協助我們辨識影像畫面中的物件，並由找到的資料中發現 YOLO 物件偵測的影像辨識

技術可以使電腦快速地辨識出一張影像畫面中的物體及物體的位置，由於能夠在很短的時間內完成對圖片中物體的識別和定位，並檢測到影像中的多個物體，且具有很高的準確性，所以在很多需要即時影像辨識的場域中具有很高的應用價值。YOLO 的這些優點與我們的構想相當吻合，因此我們將選用 YOLO 作為我們電腦視覺程式設計的開發系統。

接著我們針對後方警示系統的部分進行討論，我們觀察到公車在轉彎時除了會有方向燈之外，在公車後方也會有一片 LED 的跑馬燈顯示公車要轉彎的方向，因此我們將效仿這樣的顯示模式來提醒後方駕駛前方路口行人的情況，我們將使用 Arduino 與藍芽連線系統來進行後方警示系統的設計。

將上述兩個討論後的設計方案結合，就可以完成即時提醒駕駛與同時告知後方來車的警示裝置，而我們將此裝置稱為『行人警示系統』，要完成『行人警示系統』的開發，須完成以下研究目的：

(一)、YOLO 行人辨識系統設計

(二)、後方警示系統設計

(三)、行人警示系統整合

一、研究原理

為了開發出『行人警示系統』，我們將使用 YOLO 系統進行路口斑馬線與行人的影像辨識，而所選用的 YOLO 系統版本為較新 Yolov8(YOLO 第八版)。YOLO 行人辨識系統設計將使用 Python 做為程式的開發語言，再加上 OpenCV、Yolov8、Supervision 等相關套件來進行程式的撰寫。以下將針對 OpenCV、Yolov8 以及 Supervision 進行簡單說明與介紹：

(一)、OpenCV

OpenCV 的全名為 Open Source Computer Vision Library，是一個跨平台的電腦視覺庫。OpenCV 最初是由英特爾公司發起並參與開發，以 BSD 授權條款授權發行，可以在商業和研究領域中免費使用。OpenCV 可用於開發即時圖像處理、電腦視覺以及圖型識別程式等。本研究將使用 OpenCV 進行影像畫面處理，並將影像載入 Yolov8 系統進行物件偵測，來辨識影像中之物件，並於影像畫面中框出物件所在之範圍。

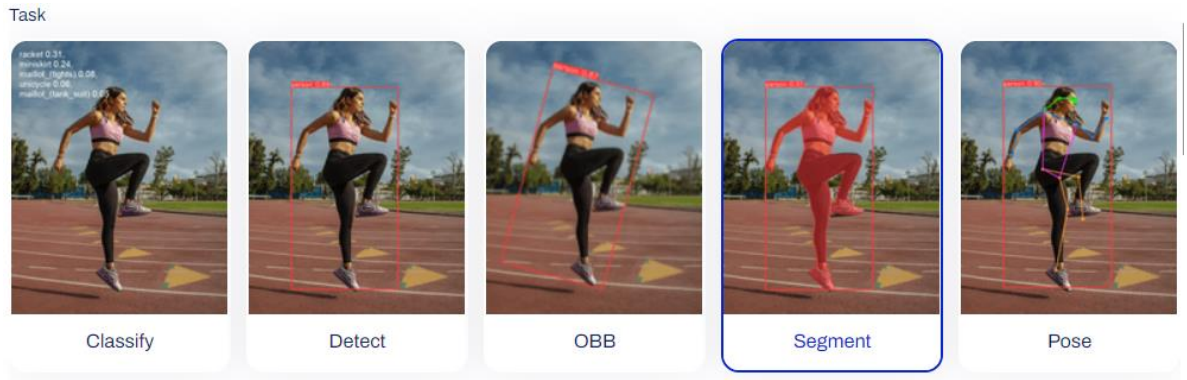
Python 安裝 OpenCV 套件指令為：pip install opencv-python

(二)、Yolov8

Yolo 為 You Only Look Once 之縮寫，表示電腦只須看過一次圖片，便可完成物件的辨識與定位，所以 Yolo 物件偵測是一項可讓電腦快速辨別影像畫面中物體種類與位置的技術，目前廣泛運用即時影像辨識環境中。而我們所設計的行人警示系統是在車子移動中不斷進行影像辨識，也是屬於即時影像辨識環境，因此我們將選用 Yolo 作為行人警示系統的影像辨識技術。Yolo 物件偵測是深度學習的一種，需要透過模型訓練學習，才能完成影像辨識。

Yolov8 為 Yolo 物件偵測系統的第八代版本，由 Ultralytics 公司所製作。Yolov8 目前擁有較佳影像辨識效能，同時有更快的辨識速度。而 Ultralytics 也提供線上模型訓練平台 Ultralytics HUB，搭配 Roboflow 線上物件標記網站，便可於線上完成 Yolov8 的模型訓練，訓練模型種類包含物件分類、物件偵測、實例分割、姿勢偵測等，如圖 1.4，本研究將採用實例分割的模式進行模型訓練。

圖 1.4 Yolov8 訓練模型種類



註：Ultralytics HUB 官方網站

Python 安裝 yolov8 套件指令為：pip install ultralytics

(三)、Supervision

Supervision 為 Roboflow 所開發之 python 套件，Roboflow 為線上影像標記平台，可下載標記後的資料集，並可供 Yolov8 進行訓練模型。而 Supervision 為 python 的電腦視覺化工具，可將 Yolov8 模型的辨識結果以視覺化的方式顯示於圖片或是影片上，或是計算影像中特定區域的辨識物件數量，本研究將使用 Supervision 的特定區域物件計算功能來進行”行人警示體統”的開發。

Python 安裝 yolov8 套件指令為：pip install supervision

二、文獻回顧

(一)、緊急煞車輔助系統相關報導

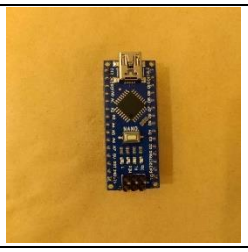
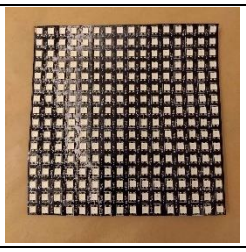
近年來，車輛與行人發生的交通事故頻傳，因此有立法委員希望修法將 AEB 系統設定為車輛的標準配備，而 AEB(Autonomous Emergency Braking)為緊急煞車輔助系統，緊急煞車輔助系統是透過攝影機、雷達或光達等感應器，偵測前方目標，並利用控制器計算危險程度，當與前方物體(車輛、行人、建築物、山壁等)距離過近時，系統會發出警示，若駕駛未能即時煞車，則系統會主動介入啟動緊急煞車程序。目前歐盟、日本、美國皆將 AEB 系統設定為新車的標準配備，但台灣政府尚未修法將 AEB 列為新車的標準配備，即使新車皆裝設有 AEB 系統，但馬路上大多數的舊款車輛並未裝設，而且舊型車款可能因硬體設備不足而無法裝設，或者因為 AEB 價格過高降低駕駛裝設意願。因此我們希望可以開發出一套行人警示系統，當車輛前方有行人時，可以即時警示駕駛與後方來車，降低交通事故的發生機率。

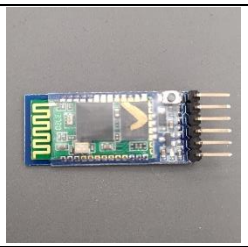
(二)、科學展覽中使用 Yolo 技術的相關作品

Yolo 運用深度學的方法，進行影像辨識的訓練，過程包含訓練、驗證、測試，過程較為繁瑣，我們想了解在那些情境下使用 Yolo 技術比較適合，因此透過科教館的科展作品檢索系統，使用關鍵字：Yolo 進行收尋，以下為國中階段使用 Yolo 技術於科展作品中的相關文獻，張宸珣等人(2019)使用 OpenCV 和 Yolo 開發一個即時影像辨識系統，用於監測火車月台和軌道附近的人員，並提供安全警示，可改善火車行駛的安全性，所使用的 Yolo 版本為 Yolov2。顏妤晅等人(2021) 利用 Yolo 系統來即時偵測大型車轉彎時駕駛視線死角區域內的行人，並即時發出音訊警告，使用的 Yolo 版本為 Yolov4。蔡奕章(2021)用運 Yolo 開發 AI 輔助視力量測系統，提升量測效率，減少護理人員的工作量，使用的 Yolo 版本為 Yolov3。楊雅盈等人(2022)運用 Yolo 進行於魚種辨識，並進行即時追蹤，根據活動軌跡判斷魚之活動力、成長狀況等，並傳遞簡訊通知飼主，所使用的 Yolo 版本為 Yolov5。由以上的文獻可以發現 Yolo 系統非常適合進行即時的影像辨識，因此本研究將使用 Yolo 做為行人警示系統的影像辨識工具，我們將選用較新的 Yolov8 作為我們的 Yolo 系統版本。

參、研究設備及器材

一、硬體設備與裝置

			
Arduino NANO	16x16 RGB 矩陣 LED	行動電源	筆記型電腦

			
Arduino 藍芽模組 (hc-05)	行車記錄器	視訊鏡頭	智慧型手機

二、設計軟體與套件

Arduino IDE	Python(3.9.7)	Thonny

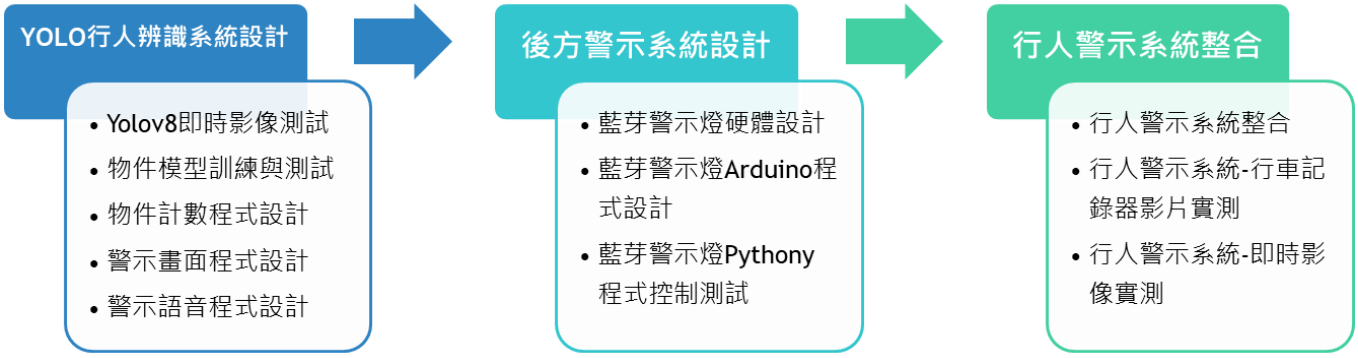
Ultralytics(Yolov8)	Roboflow(Supervision)	OpenCV

肆、研究過程與方法

本研究將使用 Python 語言進行『行人警示系統』的開發，使用 Yolov8 進行行人與斑馬線的物件辨識，透過 Supervision 獲取所辨識出物件的相關資料，進行區域物件計算，並藉由程式參數設定來判斷與分析前方路況。使用 OpenCV 將影像初步處理後導入，並於影像畫面中顯示警示標語，來提醒駕駛前方路況。將判斷結果透過藍芽通訊系統傳至 Arduino 並於顯示警示燈號，來告知後方駕駛前方行人之狀況。

研究架構流程圖如下：

圖 4.1『行人警示系統』研究架構流程圖



一、YOLO 行人辨識系統設計

實驗一、Yolov8 即時影像辨識測試

因『行人警示系統』需架設於汽車內部，並搭配視訊鏡頭或是行車記錄器的即時影像畫面進行使用。此實驗將透過 Python 運行 Yolov8 之模型，進行即時影像物件辨識測試與即時影像實例分割測試，使用的模型為 Ultralytics YOLOv8 預訓練之模型，物件辨識模型：yolov8n.pt，實例分割模型：yolov8n-seg.pt。

Python 程式碼設計如下：

```
import cv2
from ultralytics import YOLO
model = YOLO('yolov8n.pt')
#video_path = "path/to/your/video/file.mp4"
cap = cv2.VideoCapture(0)
while cap.isOpened():
    success, frame = cap.read()
```

```
    if success:
        results = model(frame)
        annotated_frame = results[0].plot()

        cv2.imshow("YOLOv8_realtime_test", annotated_frame)

        if cv2.waitKey(1) & 0xFF == ord("q"):# Break the loop if 'q' is
pressed
            break
    else:
        break

cap.release()
cv2.destroyAllWindows()
```

✓ **實驗測試結果**：此程式碼可順利於即時影像中運行 Yolov8 之模型，並即時進行物件偵測(圖 4.2)與實例分割(圖 4.3)之工作，並輸出辨識結果(圖 4.4)。

圖 4.2 即時影像物件偵測結果(yolov8n.pt)

圖 4.3 即時影像實例分割結果(yolov8n-seg.pt)

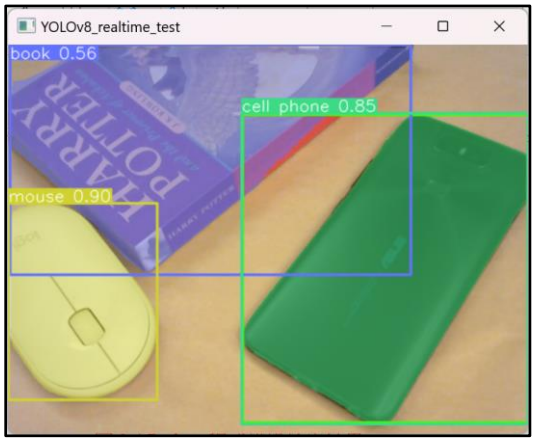


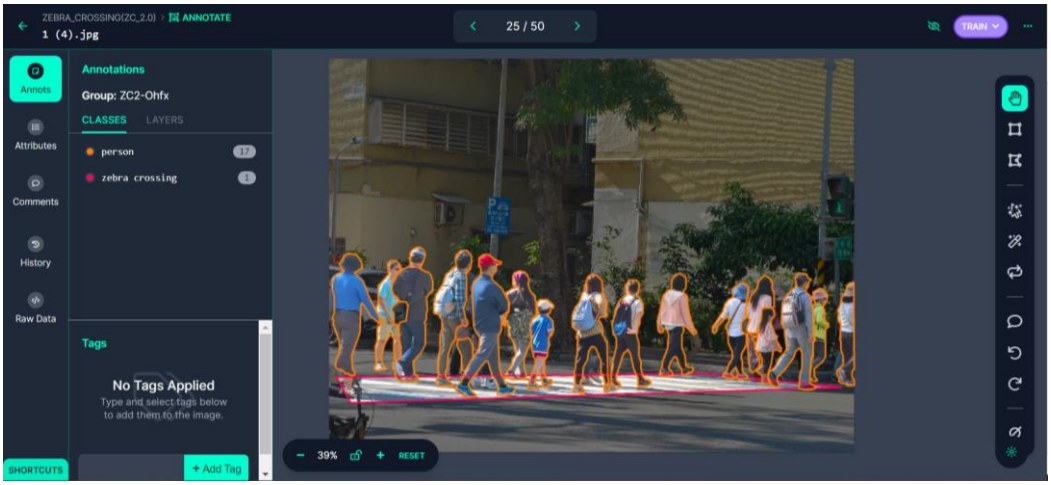
圖 4.4 Python 程式辨識輸出結果

```
互動環境 (Shell)
 0: 480x640 1 mouse, 1 cell phone, 1 book, 11.0ms
Speed: 1.0ms preprocess, 11.0ms inference, 1.6ms postprocess per image at shape (1, 3, 480, 640)

 0: 480x640 1 mouse, 1 cell phone, 1 book, 11.0ms
Speed: 1.0ms preprocess, 11.0ms inference, 2.0ms postprocess per image at shape (1, 3, 480, 640)

 0: 480x640 1 mouse, 1 cell phone, 1 book, 11.0ms
Speed: 1.0ms preprocess, 11.0ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)

 0: 480x640 1 mouse, 1 cell phone, 1 book, 10.9ms
Speed: 1.1ms preprocess, 10.9ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)
```

實驗二、Yolov8 斑馬線與行人實例分割模型訓練與測試

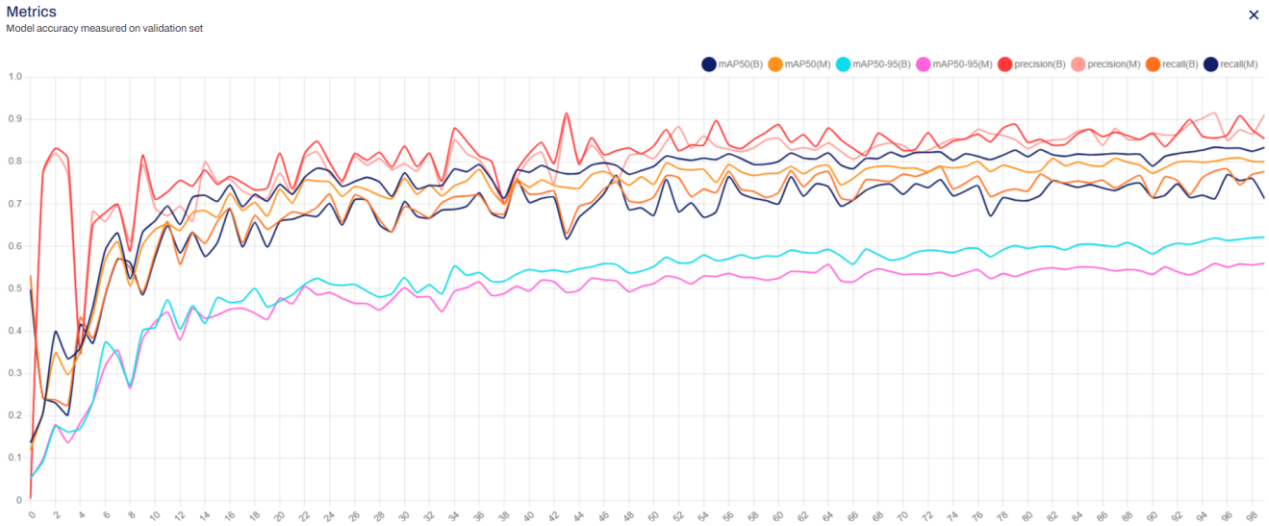
『行人警示系統』需透過分析路口行人數量，來判斷是否送出警示訊號。而路口幾乎都有斑馬線，因此只要能夠偵測斑馬線就能判斷路口位置，但 Yolov8 的預訓練模型中並未包含斑馬線的辨識資料集，所以需要自行訓練含斑馬線資料集的 Yolov8 模型。本實驗採用實例分割的模式進行模型訓練，透過拍照的方式收集斑馬線的影像資料，將影像資料上傳至 roboflow(roboflow)進行線上影像標記，所標記的物件種類有：行人(person)、斑馬線(zebra crossing)，如圖 4.5，並下載標記完成的斑馬線資料集(資料數量：857)。上傳斑馬線資料集至 Ultralytics HUB(ultralytics HUB BETA)進行線上模型訓練，最後下載訓練好的模型進行測試。

圖 4.5 roboflow 線上物件標記(行人-person、斑馬線-zebra crossing)



✓ **模型訓練結果：**

圖 4.6 訓練模型的 mAP 結果圖



由圖 4.6 可以看出，在訓練過程中，物件偵測的 mAP50(B)值與 mAP50-90(B)值皆持續上升，並於訓練後段趨於平緩，而 mAP50(B)值於訓練後段幾乎都高於 0.8，mAP50-90(B)值於訓練後段幾乎維持於 0.5 上下。在實例分割的部分，mAP50(M)值與 mAP50-90(M)值於訓練前段程上升趨勢，於訓練後段趨於平緩，mAP50(M)值於訓練後段維持於 0.8 上下，mAP50-90(M)於訓練後段維持於 0.45 上下。

✓ **模型訓練結果：**

圖 4.7 訓練模型的 Loss 結果圖(含 Box Loss、Class Loss、Object Loss)

圖 4.7 顯示，在經過多次訓練後，Box Loss、Class Loss、Object Loss 皆有降低的趨勢，Box Loss 的部分訓練值(train)下降至 1.0 以下，驗證值(val)下降至 1.0 左右；Class Loss 的訓練值(train)與驗證值(val)皆下降至 1.0 以下，其中訓練值(train)下降至 0.5 以下；在 Object Loss 的部分，訓練後段訓練值(train)下降至 1.0 以下，而驗證值(val)則維持在 1.1 附近。由圖 4.6 與圖 4.7 可以發現，此模型多次訓練後，mAP50 皆上升至 0.8 左右，而 Box Loss、Class Loss、Object Loss 皆下降至 1.0 左右，這顯示此模型針對於物件種類(行人-person、斑馬線-zebra crossing)應有良好的辨識表現。

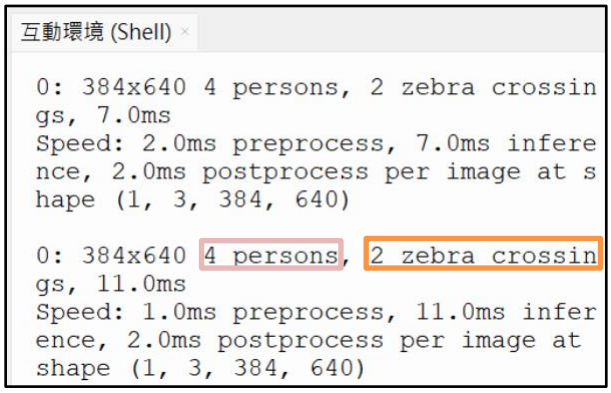
接著將上述訓練好的模型(模型名稱：zc-2.0.pt，Classes：person, zebra crossing)載入實驗一的程式碼，並導入行車記錄器影片進行測試。模型程式碼修正如下：

```
model = YOLO('yolov8n.pt')  →  model = YOLO('zc-2.0.pt')
```

✓ **實驗測試結果：**可成功辨識出斑馬線與行人，並進行影像實例分割，如圖 4.8、4.9。

圖 4.8 模型 zc-2.0.pt 實例分割結果

圖 4.9 模型 zc-2.0.pt Python 程式辨識輸出結果



實驗三、Yolov8 斑馬線與行人計數程式設計與測試

此實驗將透過 Supervision 套件進行特殊區域內的辨識物件數量計算，運用 PolygonZone 語法進行特殊區域設定，並使用 PolygonZoneAnnotator 語法計算設定區域內辨識物件，透過此方法可以進行路口區域的行人數量計算與斑馬線數量計算。

Python 程式碼設計如下：

```
import cv2
from ultralytics import YOLO
import supervision as sv
import numpy as np
ZONE_POLYGON0 = np.array([[0,0],[240,0],[240,360],[0,360],])
ZONE_POLYGON1 = np.array([[240,0],[480,0],[480,360],[240,360],])
cap = cv2.VideoCapture("20240315M.mp4")
model = YOLO("zc-2.0.pt")
model0 = YOLO("zc-2.0.pt")
model1 = YOLO("zc-2.0.pt")
wh=[480,360]
zone0 =
sv.PolygonZone(polygon=ZONE_POLYGON0,frame_resolution_wh=tuple(wh))
zone1 =
sv.PolygonZone(polygon=ZONE_POLYGON1,frame_resolution_wh=tuple(wh))
zone_annotator0 =
sv.PolygonZoneAnnotator(zone=zone0,color=sv.Color.green())
zone_annotator1 =
sv.PolygonZoneAnnotator(zone=zone1,color=sv.Color.blue())
while True:
```

```
    ret, frame = cap.read()
    result  = model(frame)[0]
    result0 = model0(frame,classes=0)[0]
    result1 = model1(frame,classes=0)[0]
    detections0 = sv.Detections.from_ultralytics(result0)
    detections1 = sv.Detections.from_ultralytics(result1)
    zone0.trigger(detections=detections0)
    zone1.trigger(detections=detections1)
    frame = zone_annotator0.annotate(scene=frame)
    frame = zone_annotator1.annotate(scene=frame)
    frame = result.plot()
    cv2.imshow("Yolov8_count_test_01", frame)
    if cv2.waitKey(1)&0xFF==27:
        break
cap.release()
cv2.destroyAllWindows()
```

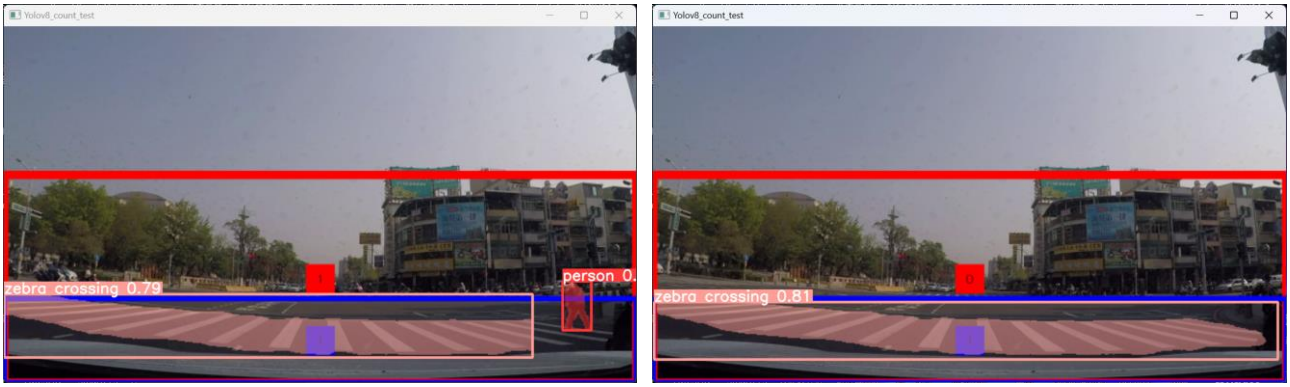
此程式於影像畫面中設定左右兩個等大的物件計數區域(綠區、黃區)，辨識物件為行人(`classes=0`)，當行人於黃區時，黃區記數顯示 1，綠區因無行人，記數顯示 0；當行人由黃區穿越至綠區時，綠區記數顯示 1，黃區此時無行人，記數顯示 0，如圖 4.10。

也可透過程式設定更改物件計數區域(紅區、藍區)與辨識物件種類(紅區：行人；藍區：斑馬線)並完成記數，如圖 4.11，可同時完成特地區域的斑馬線數量計算與行人數量計算。

✓ **實驗測試結果**：成功透過 Supervision 套件進行特殊區域之物件數量計算。

圖 4.10 區域行人記數顯示(黃區、綠區)

圖 4.11 區域物件記數顯示(藍區：斑馬線；紅區：行人)



實驗四、Yolov8 行人警示系統警示畫面程式設計與測試

此實驗將透過條件設定的方式，來觸發顯示警示標語，提醒駕駛前方行人之狀況，並使用 Pillow 套件設計警示標語於影像畫面中。首先透過 np.sum()語法取得區域物件之數量，程式碼如下：

```
c0=zone0.trigger(detections=detections0)
c1=zone1.trigger(detections=detections1)
d0=np.sum(c0==True)
d1=np.sum(c1==True)
```

```
互動環境 (Shell)
0: 384x640 4 persons, 15.0ms
Speed: 2.0ms preprocess, 15.0ms inference, 3.0ms postpro
cess per image at shape (1, 3, 384, 640)

0: 384x640 1 zebra_crossing, 16.0ms
Speed: 2.0ms preprocess, 16.0ms inference, 1.0ms postpro
cess per image at shape (1, 3, 384, 640)
[ True  True  True  True]
4
[ True]
1
```

將區域內斑馬線物件數量存入 d1，區域內行人物件數量存入 d0，當 d1>0 表示汽車行經過路口，當 d0>0 表示前方有行人，當 d1>0 與 d0>0 表示前方路口有行人。將根據上狀況進行示畫面警示標語的設計，當 d1>0 時，影像畫面上將顯示”斑馬線路口”，字體顏色為綠色；當 d0>0 時，影像畫面上將顯示”前方有行人”，字體顏色為黃底紅色字；當 d1>0 與 d0>0 時，影像畫面上將顯示”前方路口有行人”(字體顏色為黃底紅字)與”斑馬線路口”(字體顏色為綠色)，如圖 4.12，並於畫面右方顯示即時的行人數量與斑馬線數量，程式碼設計如下：

```
if d0 >0 and d1==0:
    frame = cv2ImgAddText(frame, "前方有行人", 280, 240, (255, 0, 0),
    150,(255,255,0),(255, 0, 0),3)
if d1 >0 and d0==0:
    frame = cv2ImgAddText(frame, "斑馬線路口", 480, 50, (0, 255, 0),70)
if d1 >0 and d0>0:
    frame = cv2ImgAddText(frame, "斑馬線路口", 480, 50, (0, 255, 0),70)
    frame = cv2ImgAddText(frame, "前方路口有行人", 280, 240, (255, 0,0),
```

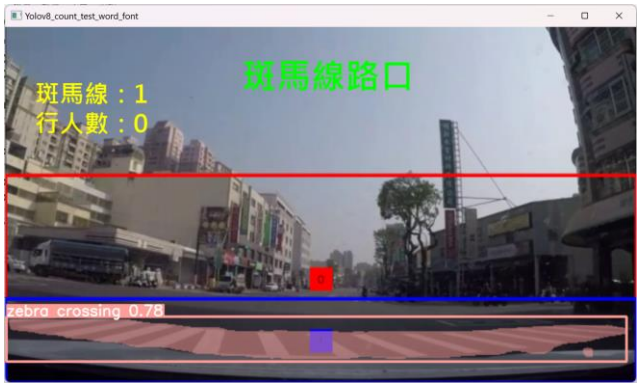
```
    150,(255,255,0),(255, 0, 0),3)
frame = cv2ImgAddText(frame, "斑馬線："+str(d1), 60, 100, (255, 255, 0),
50)
frame = cv2ImgAddText(frame, "行人數："+str(d0), 60, 160, (255, 255, 0),
50)
```

其中 cv2ImgAddText 語法為自行定義之副函式，cv2ImgAddText()副函式程式碼如下：

```
def cv2ImgAddText(frame, text, left, top, textColor=(0, 255, 0),
      textSize=20,rectanglecolor=None,outlinecolor=None,width=0):
  if (isinstance(frame, numpy.ndarray)):
    frame = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
    draw = ImageDraw.Draw(frame)
    fontText=ImageFont.truetype("msjhbd.ttc",textSize,encoding="utf-8")
    bbox = draw.textbbox((left, top), text, font=fontText)
    draw.rectangle(bbox,fill=rectanglecolor,outline=outlinecolor,
                   width=width)
    draw.text((left, top), text, textColor, font=fontText)
    return cv2.cvtColor(numpy.asarray(frame), cv2.COLOR_RGB2BGR)
```

✓ **實驗測試結果**：可成功觸發警示標語於影像畫面中。

圖 4.12 影像畫面顯示警示標語



(a) (d1>0 → 顯示"斑馬線路口")

(b) (d0>0 → 顯示"前方有行人")

(c) (d0>0 and d1>0 → 顯示"前方路口有行人"+"斑馬線路口")

實驗五、Yolov8 行人警示系統語音警示程式設計

本實驗將運用 pyttsx3 套件進行警示語音設計，當 d0>0 表示前方有行人，發出警示語音:”前方有行人”；當 d1>0 表示前方有斑馬線，汽車將行經路口，發出警示語音:”前方路口，請小心駕駛” ；當 d1>0 與 d0>0 表示前方路口有行人，發出警示語音:”前方路口有行人，請小心駕駛”。自行定義之副函式 sayword()與語音觸發程式設計如下：

sayword()副函式	語音觸發程式設計

import pyttsx3
def sayword(word):
txet = word
engine = pyttsx3.init()
engine.setProperty('rate',200)
engine.say(txet)
engine.runAndWait()
	if d0 >0 and d1==0:
	sayword('前方有行人')
	
	if d1 >0 and d0==0:
	sayword('前方路口 請小心駕駛')
	
	if d1 >0 and d0>0:
	sayword('前方路口有行人，請小心駕駛')

✓ **實驗測試結果**：影像辨識過程中，可成功觸發警示語音。

二、後方警示系統設計

實驗六、藍芽警示燈硬體設計

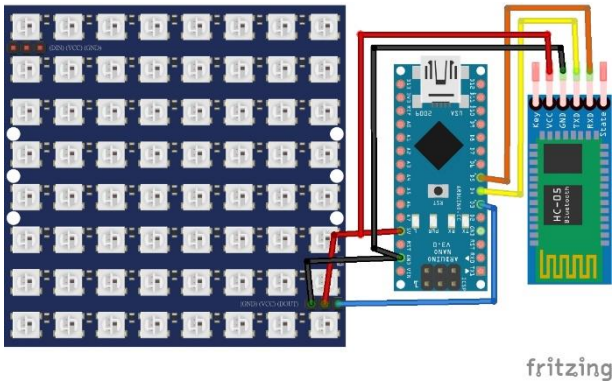
前面的實驗已經可以透過 Python 程式來執行 Yolov8 進行人與斑馬線的辨識，並即時透過畫面警示標語與警示語音提醒車內駕駛前方的行人之狀況，接著將設計後方的警示裝置，來告知後方來車前方行人之情況，本裝置效仿公車後方之 LED 顯示面板，如圖 4.13。

圖 4.13 公車後方 LED 顯示面板

本警示裝置採用 Arduino 開發板作為控制中心，因需將此警示裝置安裝於後車窗上，所

以使用藍芽無線系統接收警示訊號，並採用 16x16 RGB 矩陣 LED 元件做為顯示屏幕。為了縮小體積，此裝置使用 Arduino nano 連接藍芽模組 HC-05 與 16x16 RGB 矩陣 LED 元件，並稱為藍芽警示燈，藍芽警示燈的元件電路連接圖，如圖 4.14，Arduino nano 只需 5V 電壓就可運作，因此只須與行動電源連接就可啟動藍芽警示燈。

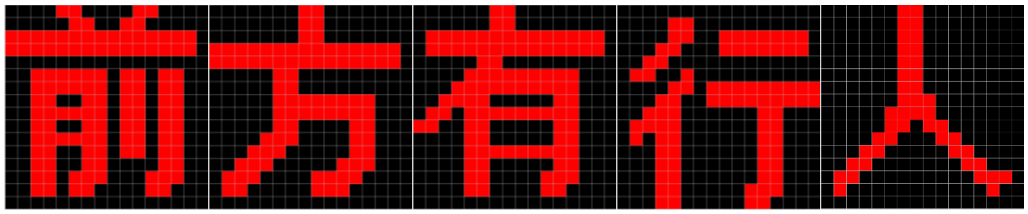
圖 4.14 藍芽警示燈元件電路連接圖



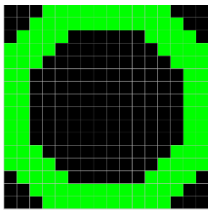
實驗七、藍芽警示燈 Arduino 程式設計

本實驗將進行藍芽警示燈的 Arduino 程式設計，當前方有行人時，16x16 RGB 矩陣 LED 元件將以逐字顯示的方式，來告知後方駕駛前方行人之狀況，顯示文字為” 前方有行人”，顯示色光為紅光，如圖 4.15(a)，每字顯示的時間為 0.5 秒；而當前方無行人時，將顯示綠色圓圈圖案，表示前方路口順暢，如圖 4.15(b)。

圖 3.15 藍芽警示燈設計圖示



(a) ”前方有行人”圖示



(b)”綠圈”圖示

Arduino 程式碼設計如下：

```
#include <avr/pgmspace.h>
#include "FastLED.h"
#define NUM_LEDS 256
#define DATA_PIN 3
CRGB leds[NUM_LEDS];
```

```
const long p01[] PROGMEM ={顯示文字:前，色光:紅光};
const long p02[] PROGMEM ={顯示文字:方，色光:紅光};
const long p03[] PROGMEM ={顯示文字:有，色光:紅光};
const long p04[] PROGMEM ={顯示文字:行，色光:紅光};
const long p05[] PROGMEM ={顯示文字:人，色光:紅光};
const long p06[] PROGMEM ={顯示:綠光圈，色光:紅光};
void setup() {
FastLED.addLeds<NEOPIXEL,DATA_PIN>(leds, NUM_LEDS);
FastLED.setBrightness(10);
}
void loop() {
for(int passtime = 0; passtime < 1; passtime++) {
FastLED.clear();
for(int i = 0; i < NUM_LEDS; i++) {leds[i]= pgm_read_dword(&(p01[i]));}
FastLED.show();
delay(500);
FastLED.clear();
for(int i = 0; i < NUM_LEDS; i++) {leds[i]= pgm_read_dword(&(p02[i]));}
FastLED.show();
delay(500);
FastLED.clear();
for(int i = 0; i < NUM_LEDS; i++) {leds[i]= pgm_read_dword(&(p03[i]));}
FastLED.show();
delay(500);
FastLED.clear();
for(int i = 0; i < NUM_LEDS; i++) {leds[i]= pgm_read_dword(&(p04[i]));}
FastLED.show();
delay(500);
FastLED.clear();
for(int i = 0; i < NUM_LEDS; i++) {leds[i]= pgm_read_dword(&(p05[i]));}
FastLED.show();
delay(500);
FastLED.clear();
```

```
for(int i = 0; i < NUM_LEDS; i++) {leds[i] =
pgm_read_dword(&(p06[i]));}
FastLED.show();
delay(500);  }
}
```

✓ **實驗測試結果**：成功使用 Arduino 程式完成警示文字設計，並於 16×16 RGB 矩陣 LED 元件上顯示，圖 4.16(a)為"前方有行人"點亮圖，圖 3.16(b)"為綠色圓圈"點亮圖

圖 4.16 藍芽警示燈實際點亮圖

(a) "前方有行人"點亮圖

(b)"綠圈"點亮圖

實驗八、藍芽警示燈 Python 程式控制測試

由於實驗七已經完成 16×16 RGB 矩陣 LED 警示標語的顯示程式設定，本實驗將透過 Python 程式進行藍芽警示燈顯示控制，透過 pynput 套件偵測鍵盤訊號，由 serial 套件將控制訊號透過電腦藍芽傳送給藍芽警示燈，並進行警示標語的顯示。藍芽通訊設定如下：

向上方向鍵(↑) → python serial 透過藍芽傳送訊號：R → Arduino 接收訊號執行 redword()

向下方向鍵(↓) → python serial 透過藍芽傳送訊號：G → Arduino 接收訊號執行 greenword()

Python 程式碼設計如下：

```
import serial
import time
from pynput import keyboard
evenement=""
print("Start")
port="COM13"
bluetooth=serial.Serial(port, 9600)
print("Connected")
bluetooth.flushInput()
result = str(0)
def on_key_release(key):
    global result
    if result !='stop':
```

```
        result = 'stop'
        print(result)
        result_bytes = result.encode('utf_8')
        bluetooth.write(result_bytes)
def on_key_pressed(key):
    global result
    #print('pressed %s' % key)
    result1 ='%s' % key
    if result == 'stop':
        if result1=='Key.up' :
           result = 'R'
           print(result)
        if result1=='Key.down' :
           result = 'G'
           print(result)
        result_bytes = result.encode('utf_8')
        bluetooth.write(result_bytes)
with keyboard.Listener(on_release = on_key_release,
                    on_press=on_key_pressed) as listener:listener.join()
```

Arduino 主程式設計如下：

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10,9);
int val;
#include <avr/pgmspace.h>
#include "FastLED.h"
#define NUM_LEDS 256
#define DATA_PIN 3
CRGB leds[NUM_LEDS];
void setup() {
 Serial.begin(9600);
 mySerial.begin(9600);
FastLED.addLeds<NEOPIXEL,DATA_PIN>
  (leds, NUM_LEDS);
  FastLED.setBrightness(10);  }
void loop() {
if(mySerial.available() > 0)
{
    val = mySerial.read();
```

```
redword()副函式
void redword()
{Serial.println("R");
for(int passtime=0;passtime<1;
passtime++) {
FastLED.clear();
for(int i=0;i<NUM_LEDS;i++){
leds[i]=pgm_read_dword(&(p01[i]));}
FastLED.show();
delay(500);
FastLED.clear();
for(int  i=0;i<NUM_LEDS;i++){
leds[i]=pgm_read_dword(&(p02[i]));}
FastLED.show();
delay(500);
FastLED.clear();
for(int i=0;i<NUM_LEDS;i++) {
leds[i]=pgm_read_dword(&(p03[i]));}
```

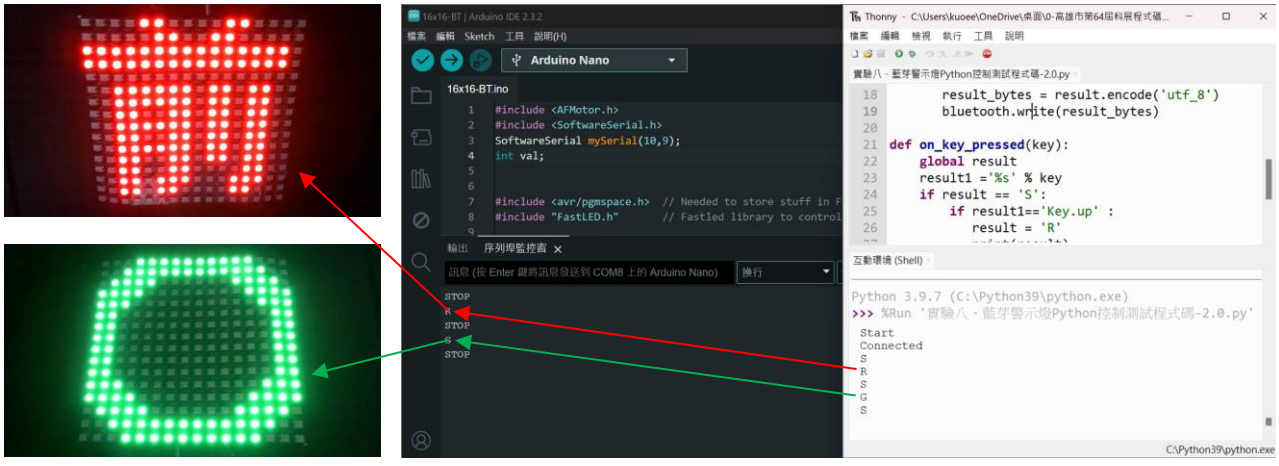
```
    if (val == 'R'){redword();}
    if (val == 'G'){greenword();}
    if (val == 'S'){Stop();}
  }
}
greenword()副函式
void greenword()
{Serial.println("G");
for(int passtime=0;passtime<1;
passtime++) {
FastLED.clear();
for(int i=0;i<NUM_LEDS;i++) {
leds[i]=pgm_read_dword(&(g01[i]));}
FastLED.show();
delay(1000);
FastLED.clear();
for(int i=0;i<NUM_LEDS;i++) {
leds[i]=pgm_read_dword(&(p06[i]));}
FastLED.show();
delay(500);}}
```

```
FastLED.show();
delay(500);
FastLED.clear();
for(int i=0;i< NUM_LEDS;i++) {
leds[i]=pgm_read_dword(&(p04[i]));}
FastLED.show();
delay(500);
FastLED.clear();
for(int i=0;i<NUM_LEDS;i++) {
leds[i]=pgm_read_dword(&(p05[i]));}
FastLED.show();
delay(500);
FastLED.clear();
for(int i=0;i<NUM_LEDS;i++) {
leds[i]=pgm_read_dword(&(p06[i]));}
FastLED.show();
delay(500);}}
```

```
const long p01[] PROGMEM ={ 前 }; const long p02[] PROGMEM ={ 方 };
const long p03[] PROGMEM ={ 有 }; const long p04[] PROGMEM ={ 行 };
const long p05[] PROGMEM ={ 人 }; const long p06[] PROGMEM ={ ■ };
const long g01[] PROGMEM ={ ○ };
```

✓ **實驗測試結果**：成功透過 Python 程式傳送藍芽訊號控制藍芽警示燈，並顯示警示標語，實測結果如圖 4.17。

圖 4.17 藍芽警示燈 Python 程式控制測試結果



三、行人警示系統整合

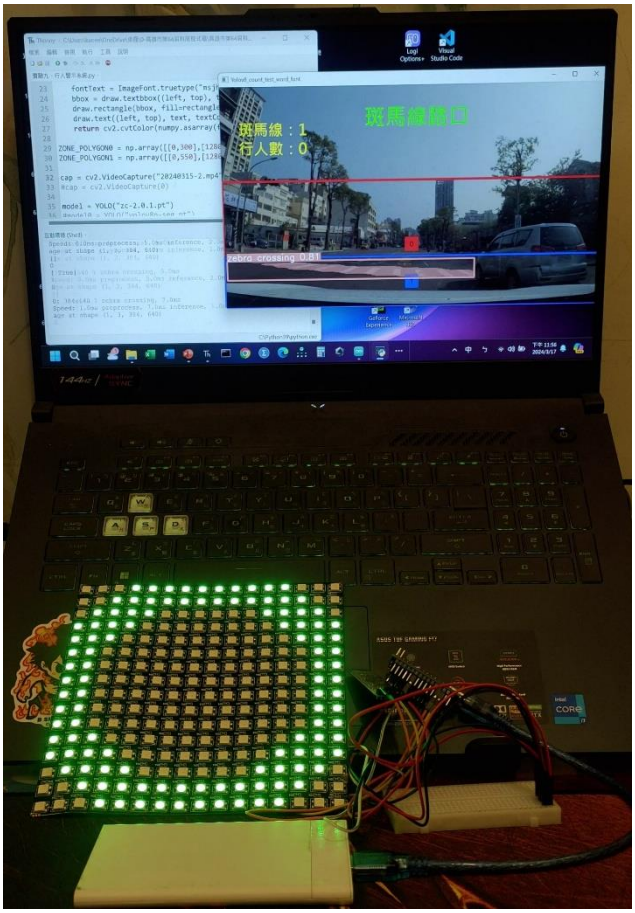
實驗九、行人警示系統整合

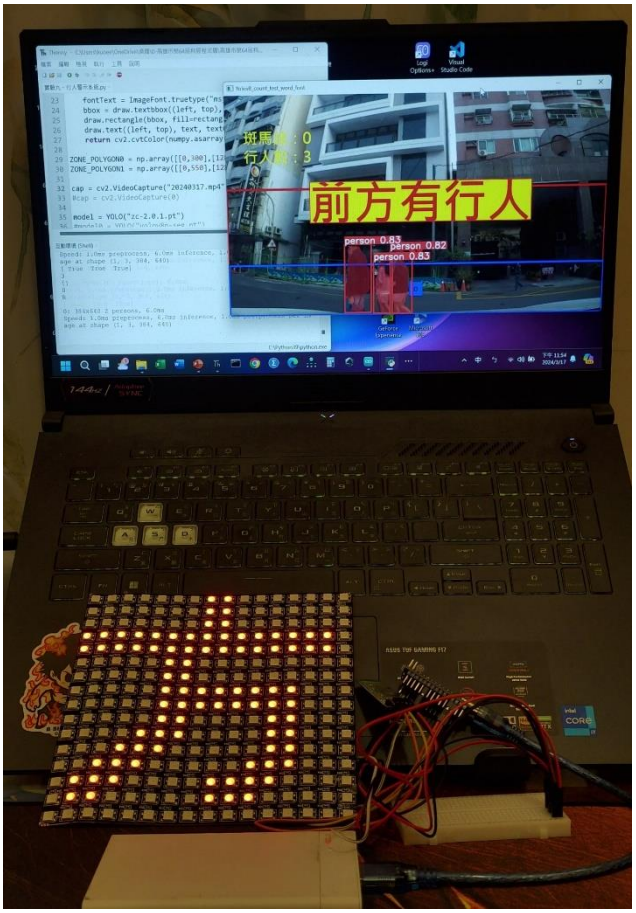
我們所設計的行人警示系統包含即時提醒車內駕駛與告知後方駕駛目前前方路口行人之情況，因此需要將”YOLO 行人辨識系統”與”後方警示系統”整合在一起，才能完成我們所設計的行人警示系統。此實驗需使用實驗二所訓練出的 Yolov8 模型：zc-2.0.pt，並將實驗五的 Yolov8 影像辨識程式碼與實驗八的 Python 藍芽控制程式碼結合，此程式碼將可透過藍芽訊號控制 Arduino 開發版，搭配實驗八的 Arduino 就可啟動後方警示系統，並可於 16x16 RGB 矩陣 LED 上顯示警示標語，完成告知車後方車輛駕駛任務。

Python 完整程式碼設計如附件一。

- ✓ **實驗測試結果：**成功完成行人警示系統 python 程式碼的整合，並可透過藍芽傳送訊號控制藍芽警示燈顯示警示標語，實驗結果如圖 4.18、圖 4.19。

圖 4.18 行人警示系統(斑馬線路口→亮綠圈)　圖 4.19 行人警示系統(前方有行人→亮紅字)





實驗十、行人警示系統－行車記錄器影片實測

本實驗將行車記錄器影片載入行人警示系統進行測試，發現辨識模型(zc-2.0.pt)會有錯誤偵測的現象，如圖 4.20，因此將設定 conf 值來提減少錯誤偵測的機會，我們發現將辨識模型(zc-2.0.pt)的 conf 值設定在 0.7 可減少錯誤偵測的機會，提升辨識品質，如圖 4.21，conf 值設定語法如下：

行人辨識：
`result0=model0(frame,classes=0)→result0=model0(frame,classes=0,conf=0.7)`
斑馬線辨識：
`result1=model1(frame,classes=1)→result1=model1(frame,classes=1,conf=0.7)`

✓ **實驗測試結果：**成功透過設定 conf 值來提升行人警示系統的辨識品質，降低錯偵測的機會。

圖 4.20 錯誤偵測

圖 4.21 無錯誤偵測(conf = 0.7)

實驗十一、行人警示系統－即時影像實測

本實驗將行人警示系統安裝於汽車內進行上路實測，行人警示系統在行車過程中可不斷的進行影像辨識，並於畫面中顯示警示標語，提醒車內駕駛前方路況，同時點亮後方的藍芽警示燈，告知後車駕駛前方行人的情況，如圖 4.22 所示。當行人警示系統只偵測到斑馬線時，於畫面中顯示”斑馬線路口”，於後方藍芽警示燈上點亮綠色圓圈，如圖 4.22(a)所示；當行人警示系統只偵測到行人時，於畫面中顯示”前方有行人”，於後方藍芽警示燈上點亮”前方有行人”的動態警示標語跑馬燈，如圖 4.22(b)所示；當行人警示系統同時偵測到斑馬線與行人時，於畫面中顯示”斑馬線路口”與”前方路口有行人”，於後方藍芽警示燈上點亮”前方有行人”的動態警示標語跑馬燈，如圖 4.22(c)所示。

✓ **實驗測試結果：**行人警示系統成功完成上路實測。

圖 4.22 行人警示系統路上實測結果

(a) (行人警示系統只偵測到斑馬線→畫面顯示”斑馬線路口”→後方警示燈點亮”綠色圓圈”)

(b) (行人警示系統只偵測到行人→畫面顯示”前方有行人”→後方警示燈點亮”前方有行人”)

(c) (同時偵測到行人與斑馬線→畫面顯示”前方路口有行人”→後方警示燈點亮”前方有行人”)

實驗十二、行人警示系統車輛速率實測

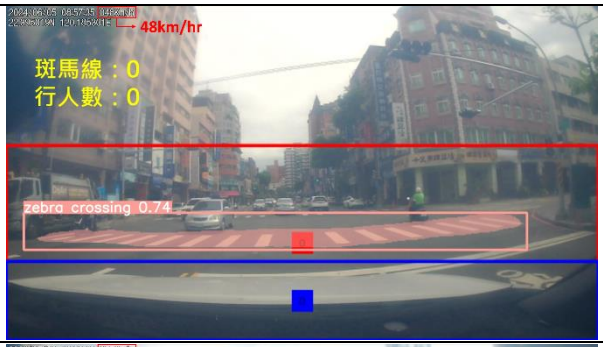
本實驗將於不同行車速率下，進行行人警示系統的實測，來了解在較高行車速率下(大於50km/hr)行人警示系統能否順利運作，行車速率控制在一般市區道路限速(60km/hr)及省道限

速(70km/hr)以下進行實測，實測結果如表 4-1。

✓ **實驗測試結果：**『行人警示系統』可於行車速率 70km/hr 以下成功完成影像辨識，可運用於一般市區道路與省道上。

表 4-1

不同行車速率下『行人警示系統』辨識實測結果

行車時速(km/hr)	辨識畫面	辨識結果
≦50 km/hr		成功
50~60 km/hr		成功
60~70 km/hr		成功

實驗十三、行人警示系統辨識模型優化

由實驗十二之結果，行人警示系統於較高行車速率下可順利運作，但因 zc-2.0.pt 模型影像照片資料皆為路口之影像，因此無法提早提醒駕駛前方路況，本實驗將加入新的影像數據，使行人警示系統可於較高行車速率下提早提醒駕駛前方路況。

根據中華民國交通部公路局的資料顯示，行車安全距離約為車速的 1/2 距離，因此行車時速 60~70km/hr 的安全距離為 30~35m。將使用行車紀錄之影像畫面，進行遠距離的斑馬線

物件影像標記，如圖 4.23，並進行模型訓練(模型名稱 zc-5.0.pt)進行實測，實測結果如表 4-2。

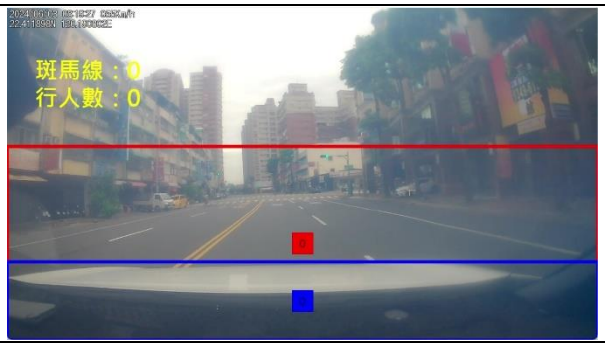
圖 4.23 路口影像與遠距離影像標記

(a)路口影像標記　　(b)遠距離的斑馬線物件影像標記

✓ **實驗測試結果：**『行人警示系統』搭載 zc-5.0.pt 模型，可以進行遠距離影像辨識，可提早提醒駕駛前方路口之路況。

表 4-2

模型影像辨識比較表

zc-2.0.pt	zc-5.0.pt
	

使用 Ultralytics HUB 時可選擇五個預訓練模型模式(YOLOv8n、YOLOv8s、YOLOv8m、YOLOv8l、YOLOv8x)進行模型訓練，我們將使用 zc-5.0.pt 之資料集(含遠距離斑馬線物件影像標記)進行上述五個模式的模型訓練，訓練完成的模型名稱分別為：zc-5.0n.pt、zc-5.0s.pt、zc-5.0m.pt、zc-5.0l.pt、zc-5.0x.pt。將訓練好的模型與 Ultralytics HUB 官方訓練之分割模型進行比較(YOLOv8n-seg、YOLOv8s-seg、YOLOv8m-seg、YOLOv8l-seg、YOLOv8x-seg)，如表 4-3。影像辨識速率則採用幀率(Frames per second, FPS)進行討論，實測結果如表 4-3。

表 4-3

Ultralytics HUB 官方模型及本研究訓練之模型比較表

官方 Model	mAP^{box}_{50-95}	mAP^{mask}_{50-95}	訓練 Model	mAP^{box}_{50-95}	mAP^{mask}_{50-95}	幀率(FPS)

YOLOv8n-seg	0.367	0.305	zc-5.0n.pt	0.622	0.561	72.02
YOLOv8s-seg	0.446	0.368	zc-5.0s.pt	0.631	0.560	65.76
YOLOv8m-seg	0.499	0.408	zc-5.0m.pt	0.638	0.591	41.10
YOLOv8l-seg	0.523	0.426	zc-5.0l.pt	0.638	0.574	29.32
YOLOv8x-seg	0.534	0.434	zc-5.0x.pt	0.644	0.582	20.35

由表 4-3 可以發現本研究所訓練之模型(zc-5.0n.pt、zc-5.0s.pt、zc-5.0m.pt、zc-5.0l.pt、zc-5.0x.pt)在 mAP(box50-95)值與 mAP(mask50-95)值皆優於 Ultralytics HUB 官方之模型，表示所訓練之模型皆有良好的辨識效果。

在影像辨識速率方面，目前一般市售行車記錄器常以幀率 30FPS ~60FPS 進行拍攝，因此選用 zc-5.0n.pt、zc-5.0s.pt 及 zc-5.0m.pt 作為影像辨識模型，可保有順暢的辨識影像畫面，其中使用由 YOLOv8m 模式訓練出來的 zc-5.0m.pt 模型，可獲的次佳的 mAP(box50-95)值 0.638 與最佳的 mAP(mask50-95)值 0.591，並保有順暢的影像品質(幀率為 41.10)。

伍、研究結果

一、行人警示系統

由前面的實驗過程，我們已經完成行人與斑馬線的 Yolov8 實例分割模型訓練(模型名稱:zc-2.0.pt 及 zc-5.0.pt)，經 python 程式測試後，zc-2.0.pt 及 zc-5.0.pt 皆有良好的辨識效果，將使用 zc-2.0.pt 及 zc-5.0.pt 作為我們行人警示系統的辨識模型。透過 Supervision 進行斑馬線物件辨識區設定與人行物件辨識區設定，並進行影像中物件數量的即時統計。透過辨識區內的物件統計數量進行條件設定，判斷車輛前方是否有行人或是斑馬線，若車輛前方有行人，將於影像畫面中顯示”前方有行人”的警示標語，並發出”前方有行人”的警示語音，來提醒車內駕駛；同時送出藍芽訊號控制 Arduino NANO 開發版，使藍芽警示燈顯示”前方有行人”的動態警示標語跑馬燈，提醒後方駕駛前方路況；若車輛前方為無行人之斑馬線，影像畫面中將顯示”斑馬線路口”，並發出”前方路口 請小心駕駛”的提醒語音，提醒車內駕駛目前正行經斑馬線路口，需小心駕駛，而後方警示燈將亮起綠色圓圈燈號，告知後車駕駛目前前方路口無行人可小心通過。若車輛前方同時出現斑馬線與行人時，影像畫面中將顯示”斑馬線路口”與”前方路口有行人”，並發出”前方路口有行人 請小心駕駛”的警示語音，來提醒車內駕駛，並於後方警示燈顯示”前方有行人”之警示標語跑馬燈，提醒後方駕駛須保持適當的安全行車

距離，避免追撞事故的發生。此系統可接透過影像辨識來判斷前方路口行人之狀況，並同時提醒車內駕駛與後車駕駛，來降低汽車與行人的事故發生機率，同時也降低前車因禮讓行人而被後車追撞的機會，因此我們已經成功開發出『行人警示系統』。

陸、討論

一、行人警示系統優化

本研究所使用的開發工具與執行工具皆為筆記型電腦，但經過上路實測後發現安裝略為笨重，希望未來能將此系統移植至手機或是平板上，並可直接透過拍照鏡頭進行辨識，增加使用的方便性。而 Yolov8 還有物件追蹤的功能，期待之後能將此功能加入『行人警示系統』，並透過程式計算出行人行走速度，以及顯示行人通過斑馬線所需之時間。而於本研究的實驗期間，Yolov9 也問世了，有比 Yolov8 更佳的辨識效能，因此也可以透過 Yolov9 進行行人與斑馬線辨識的模型訓練，應可獲得比 Yolov8 模型更準確、更快速的辨識效果。以上皆為可以優化『行人警示系統』之方法。

二、研究應用推廣價值

配有 AEB(緊急煞車輔助系統)之車輛約於 2018 年開始進入台灣市場，根據台灣交通部公路局統計，車齡未滿 6 年的車輛數為 2,557,163 輛，而全台車輛總數為 8,616,202 輛，全台約有 71%的車輛無法配有 AEB 系統，且 AEB 為高階車款的配備，因此全台未裝有 AEB 之車輛數超過 71%。本研究開發出的『行人警示系統』雖無法直接做動車輛煞車，但具即時提醒車內駕駛與後車駕駛之功能，此功能與 AEB 的警示功能類似，且安裝成本較 AEB 低，可運用於未裝有 AEB 之車輛(約占全台車輛總數的 71%)，具有高度的應用推廣價值與市場價值。

三、未來工作

為了讓『行人警示系統』有更強大的功能，未來研究的工作目標如下：

1. 使用 Yolov9 進行行人與斑馬線的模型訓練。
2. 加入行人物件追蹤功能，計算出行人行走速度，推估行人過馬路的時間。
3. 開發『行人警示系統』App，將此系統移植至手機與平板平台上。

柒、結論

1. 成功透過 Python 執行 Yolov8 進行即時影像辨識。
2. 成功使用 roboflow 與 Ultralytics HUB 訓練出行人與斑馬線的 Yolov8 模型，並成功於影像中完成行人與斑馬線的實例分割。
3. 成功使用 Supervision 進行區域設定，並可計算出區域內 Yolov8 所辨識出物件的數量。
4. 成功完成透過 OpenCV 完成警示標語的設計，透過條件設定，可判斷前方路況顯示”前方有行人”、”前方路口有行人”及”斑馬線路口”等標語。
5. 成功透過完成語音警示的程式設計，可根據前方路況判斷，發出” 前方有行人”、” 前方路口 請小心駕駛”及”前方路口有行人 請小心駕駛”等警示語音。
6. 成功使用 Arduino NANO 與 16*16 RGB 矩陣 LED 開方出後方藍芽警示燈。
7. 成功使用 Arduino 程式碼設計出後方警示燈號，包含”前方有行人”之跑馬燈與綠色圓圈燈號。
8. 成功透過 Python 程式傳送藍芽訊號控制後方警示燈，並使後方警示燈點亮警示燈號。
9. 成功整合”YOLO 行人辨識系統”與”後方警示系統”的 Python 程式，完成『行人警示系統』的程式開發。
10. 成功透過行車紀錄器影片進行『行人警示系統』的測試，並透過設定 conf = 0.7 來降低錯誤偵測。
11. 成功將『行人警示系統』安裝於車輛上並進行上路測試，可即時偵測行人與斑馬線，並於前方有行人或斑馬線時，顯示警示標語並發出警示語音提醒車內駕駛，同於透過後方警示燈兩起警示燈號，告知後方駕駛前方路況。
12. 已成功開發行人警示系統，安裝於車輛上就可上路使用，希望此系統能降低行人與車輛發生交通事故的機率。

捌、參考文獻及其他

張宸珣、王閔溱、吳尚原(2019)。**火車行駛安全辨識系統**。全國中小學科展作品。

顏妤晅、陳以華、曹家瑜(2021)。**智慧影像，「One」視平安~大型車轉彎安全偵測警示系統**。

全國中小學科展作品。

蔡奕章(2021)。**AI 影像辨識輔助視力量測系統**。全國中小學科展作品。

楊雅盈、簡楷真(2022)。**「深」不可「測」——紅蓮燈魚 3D 座標重建與智慧管理系統**。全國中小學科展作品。

STEAM 教育學習網。https://steam.oxxostudio.tw/

Ultralytics YOLOv8 Docs。https://docs.ultralytics.com/

roboflow。https://app.roboflow.com/

Supervision。https://supervision.roboflow.com/latest/

OpenCV(2020 年 5 月 18 日)。載於**維基百科**。

https://zh.wikipedia.org/w/index.php?title=OpenCV&oldid=68163631

施威銘研究室(2020)。**用 Python 學 AIoT 智慧連網**。旗標科技股份有限公司。

郭英勝、鄭志宏、龔志銘、謝哲光 (2018) 。**實用 Pyrhon 程式設計**。碁峯資訊。

楊明豐(2018)。**Arduino 物聯網最佳入門與應用：打造智慧家庭輕鬆學**。碁峯資訊。

Pratik Desai(2016)。**Python x Arduino 物聯網整合開發實戰**(CAVEDU 團隊,曾吉弘 譯)．碁峯資訊。

朱克剛(2021)。**AIOT 與 OpenCV 實戰應用**。台北市．碁峯資訊。

李立宗(2020)。**科班出身的 AI 人必修課 OpenCV 影像處理 使用 Python**。台北市．深智數位股份有限公司。

謝東明(2023 年 5 月 9 日)。破除行人地獄污名　時代力量籲努力提升車輛行車安全配備。CNEWS 匯流新聞網。https://cnews.com.tw/204230509a05/

緊急煞車輔助之快速道路系統。**TNCAP**。https://www.tncap.org.tw/

姚　衍（總編輯）(2023)。國中自然科學 2 上課本。翰林出版社。

AI4kids(2023 年 3 月 1 日)。YOLO 是什麼？３分鐘了解 YOLO 的演進，可以應用在生活中哪些地方！。https://ai4kids.ai/blogs/blog/what-is-yolo

本研究之照片圖片皆為作者自行拍攝製作

【評語】032813

1. 本作品利用 YOLO 技術識別斑馬線上的行人，並通過 LED 顯示警語，與台灣當前對行人安全日益重視的趨勢高度契合，符合社會需求，值得讚許。

2. 本作品能實際裝設於汽車裡面，進行真實的即時拍攝與辨識，實用性頗高。

3. 若能進一步探討辨識所需的時間以及駕駛人反應時間，將能進一步提升其實用價值。

作品簡報

行人警示系統之開發與研究

摘要

台灣近年來，行人與汽機車的交通事故頻傳，我們開始思考是否可以開發出一套行人警示系統，來輔助駕駛了解前方路況，即時提醒駕駛需禮讓行人，同時告知後方來車前方路況，減少追撞事故。我們使用Yolov8進行斑馬線與行人辨識模型的訓練，Python程式執行模型進行物件偵測，Supervision進行區域內物件數量的統計。經由條件設定，判斷車輛前方是否有行人或是斑馬線，如果車輛前方有行人，將於影像畫面中顯示警示標語，發出警示語音提醒車內駕駛，降低汽車與行人的事故發生機率，同時送出藍芽訊號，使警示燈顯示動態警示燈號，提醒後方駕駛了解前方路況，降低前車因禮讓行人而被後車追撞的機會。我們已經成功開發出**『行人警示系統』**，並可成功上路使用。

關鍵字：Yolov8、OpenCV、Python

前言

一、研究動機與背景

台灣近年來，行人與汽機車發生交通事故頻傳，行人行經斑馬線上穿越馬路時，常面臨著駕駛未停車禮讓的情況，使得過程險象環生。因少部分的不良駕駛或是駕駛注意力不集中而造成疏忽，導致悲劇發生。政府為降低上述事故的發生機率，進行相關道路交通安全規則條例修改，其中**停讓行人**是指車輛須在完全停止的狀態下禮讓行人通過斑馬線，雖然保障了行人通過斑馬線的安全，但往往後方的駕駛並無法了解前車禮讓行人的情況，進而對前車按壓喇叭或因無法了解前車動態，而無保持適當的行車距離，導致後車碰撞前車的事故。

根據上述狀況，我們開始思考是否可以透過電腦視覺的概念，開發出一套行人警示系統，來輔助駕駛了解前方路口的行人狀況，並即時提醒駕駛，降低因駕駛注意力不集中而發生交通事故的機率，同時也能告知後方來車，目前前方路口是否有行人穿越馬路，減少前車被後車追撞的機會，讓整個禮讓行人的過程更加安全。因此進行相關研究，開發出包含上述概念的系統，並可運用於生活中，就是我們進行這次研究的最大動機。

資料來源：內政部警政署網站，2023

資料來源：東森新聞，2023

資料來源：TVBS新聞，2023

二、研究目的

為了開發出可即時提醒駕駛前方路口行人狀況的輔助裝置，並可以同時將路口行人狀況告知後方來車的警示系統，我們將這樣的設計想法分成兩大部分進行討論，分別為：**1. 行人辨識系統設計 2. 後方警示系統設計**

將這兩個概念結合，使用Yolov8進行斑馬線與行人的影像辨識，透過藍芽訊號將辨識結果傳送至後方警示系統，就開發出一個可以即時提醒駕駛並同時告知後方來車的警示裝置，我們稱此裝置為**『行人警示系統』**。開發出**『行人警示系統』**，是我們做這個研究最大目的。

研究過程與方法

本研究將使用Python語言進行『行人警示系統』的開發，使用Yolov8進行行人與斑馬線的物件辨識，透過Supervision獲取所辨識出物件的相關資料，進行區域物件計算，並藉由程式參數設定來判斷與分析前方路況。使用OpenCV將影像初步處理後導入，並於影像畫面中顯示警示標語，來提醒駕駛前方路況。將判斷結果透過藍芽通訊系統傳至Arduino並於顯示警示燈號，來告知後方駕駛前方行人之狀況。研究架構流程圖如下：

YOLO行人辨識系統設計

- Yolov8即時影像測試
- 物件模型訓練與測試
- 物件計數程式設計
- 警示畫面程式設計
- 警示語音程式設計

後方警示系統設計

- 藍芽警示燈硬體設計
- 藍芽警示燈Arduino程式設計
- 藍芽警示燈Python程式控制測試

行人警示系統整合

- 行人警示系統整合
- 行人警示系統-行車記錄器影片實測
- 行人警示系統-即時影像實測

一、YOLO行人辨識系統設計 以下研究之照片圖片皆為研究者自行拍攝製作

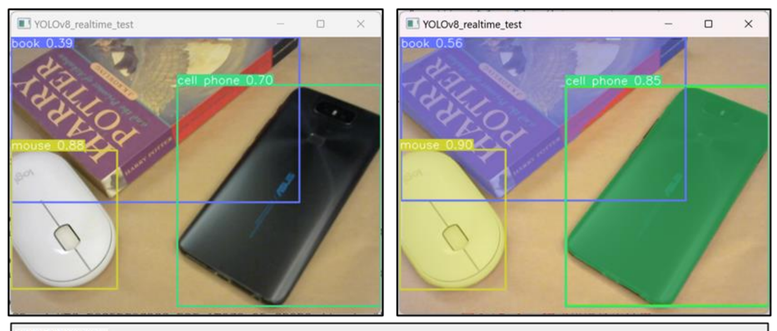
實驗一、Yolov8即時影像辨識測試

此實驗將透過Python運行Yolov8之模型，進行即時影像物件辨識測試與即時影像實例分割測試。
物件辨識模型：yolov8n.pt；實例分割模型：yolov8n-seg.pt

✓ **實驗測試結果：**

Python程式碼

```
import cv2
from ultralytics import YOLO
model=YOLO('yolov8n.pt')
model=YOLO('yolov8n-seg.pt')
cap=cv2.VideoCapture(0)
while cap.isOpened():
 success, frame=cap.read()
 if success:
  results=model(frame)
  annotated_frame=results[0].plot()
  cv2.imshow("YOLOv8_realtime_test",
annotated_frame)
```




```
互動環境 (Shell)
0: 480x640 1 mouse, 1 cell phone, 1 book, 11.0ms
Speed: 1.0ms preprocess, 11.0ms inference, 1.6ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 mouse, 1 cell phone, 1 book, 11.0ms
Speed: 1.0ms preprocess, 11.0ms inference, 2.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 mouse, 1 cell phone, 1 book, 11.0ms
Speed: 1.0ms preprocess, 11.0ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 mouse, 1 cell phone, 1 book, 10.9ms
Speed: 1.1ms preprocess, 10.9ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)
```

實驗二、Yolov8斑馬線與行人實例分割模型訓練與測試

本實驗採用實例分割的模式進行模型訓練，透過拍照收集斑馬線的影像資料，將影像資料上傳至 roboflow 進行線上影像標記，標記物件種類為：行人(person)、斑馬線(zebra crossing)，下載標記完成的資料集(資料數量：857)。上傳資料集至 ultralytics HUB 進行線上模型訓練，下載訓練好的模型並載入實驗一的程式碼進行測試。

✓ **實驗測試結果：**成功辨識出斑馬線與行人，並進行影像實例分割。


```
互動環境 (Shell)
0: 384x640 4 persons, 2 zebra crossings, 7.0ms
Speed: 2.0ms preprocess, 7.0ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 persons, 2 zebra crossings, 11.0ms
Speed: 1.0ms preprocess, 11.0ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)
```

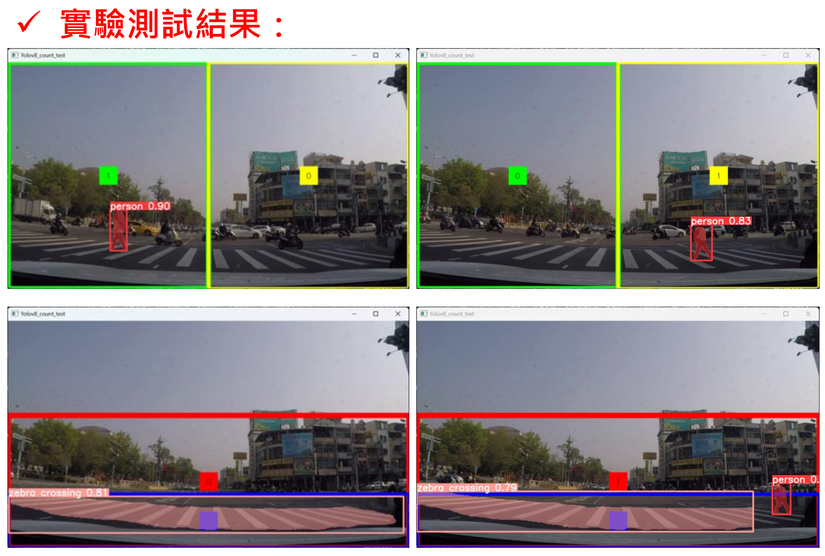
實驗三、Yolov8斑馬線與行人計數程式設計與測試

此實驗將透過Supervision套件進行特殊區域內的辨識物件數量計算，運用PolygonZone語法進行特殊區域設定，並使用PolygonZoneAnnotator語法計算設定區域內辨識物件，透過此方法可以進行路口區域的行人數量計算與斑馬線數量計算。

Python程式碼

```
import supervision as sv
import numpy as np
ZONE_POLYGON0 =
np.array([[0,0],[240,0],[240,360],
[0,360]]) 設定區域
ZONE_POLYGON1 =
np.array([[240,0],[480,0],[480,360
],[240,360]]) 設定區域
wh=[480,360]
zone0 =
sv.PolygonZone(polygon=ZONE_POLYGO
N0,frame_resolution_wh=tuple(wh))
zone1 =
sv.PolygonZone(polygon=ZONE_POLYGO
N1,frame_resolution_wh=tuple(wh))
zone_annotator0 =
sv.PolygonZoneAnnotator(zone=zone0
,color=sv.Color.green()) 計算數量
zone_annotator1 =
sv.PolygonZoneAnnotator(zone=zone1
,color=sv.Color.blue()) 計算數量
```

✓ **實驗測試結果：**



實驗四、Yolov8行人警示系統警示畫面程式設計與測試

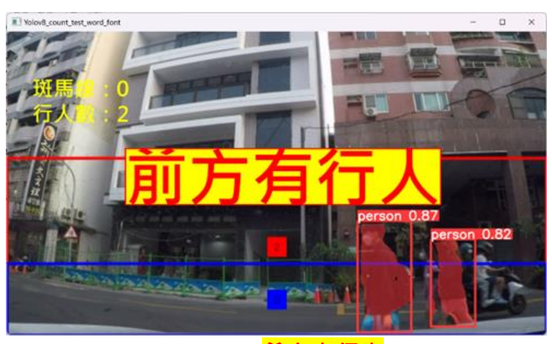
此實驗將透過條件設定的方式，來觸發顯示警示標語，透過`np.sum()`語法取得區域物件之數量，將區域內斑馬線物件數量存入`d1`，區域內行人物件數量存入`d0`，將根據`d1`及`d0`的條件狀態進行示畫面警示標語顯示。

實驗測試結果：

Python程式碼

```
c0=zone0.trigger(detections=detections0)
c1=zone1.trigger(detections=detections1)
d0=np.sum(c0==True)
d1=np.sum(c1==True)
```

```
互動環境 (Shell)
0: 384x640 4 persons, 15.0ms
Speed: 2.0ms preprocess, 15.0ms inference, 3.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 zebra crossing, 16.0ms
Speed: 2.0ms preprocess, 16.0ms inference, 1.0ms postprocess per image at shape (1, 3, 384, 640)
[ True  True  True  True]
4
[ True]
1
```



(d0>0 → 前方有行人)

(d1>0 → 斑馬線路口)

(d0>0 d1>0 →斑馬線路口+前方路口有行人)

實驗五、Yolov8行人警示系統語音警示程式設計

本實驗將運用pyttsx3套件進行警示語音設計，並自行定義副函式`sayword()`來發出警示語音，語音觸發程式設計如下：

sayword()副函式

```
import pyttsx3
def sayword(word):
    txet = word
    engine = pyttsx3.init()
    engine.setProperty('rate',200)
    engine.say(txet)
    engine.runAndWait()
```

Python語音觸發程式

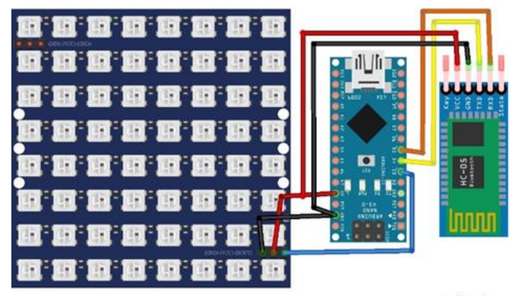
```
if d0 >0 and d1==0:
    sayword('前方有行人')

if d1 >0 and d0==0:
    sayword('前方路口 請小心駕駛')

if d1 >0 and d0>0:
    sayword('前方路口有行人，請小心駕駛')
```

✓ **實驗測試結果：**影像辨識過程中，可成功觸發警示語音。

二、後方警示系統設計

實驗六、藍芽警示燈硬體設計

本實驗採用Arduino開發板作為控制中心，使用HC-05藍芽無線系統接收警示訊號，採用16×16 RGB 矩陣LED元件做為顯示屏幕，電源使用5V行動電源，稱此裝置為藍芽警示燈。

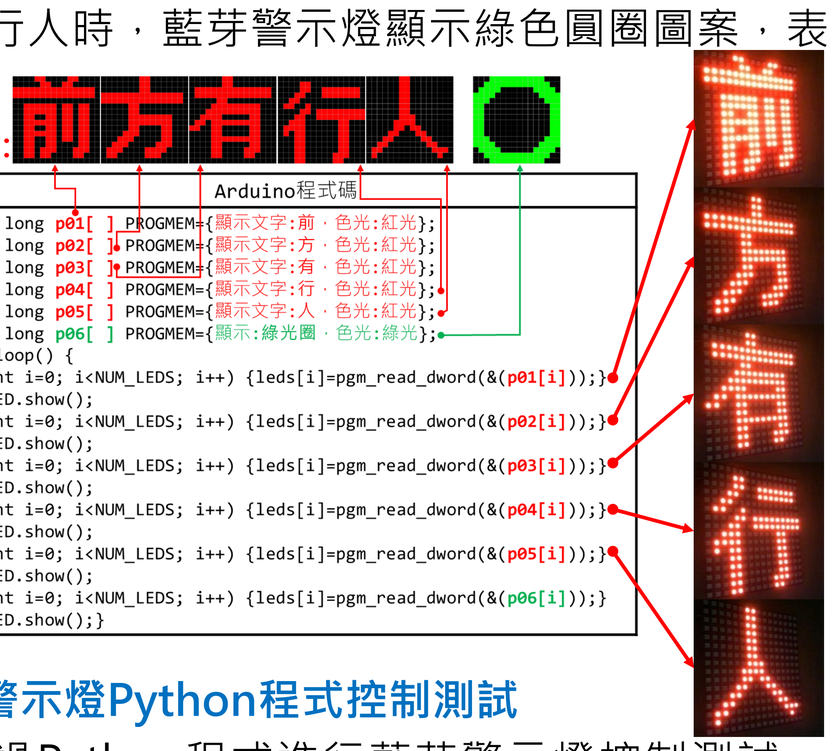


藍芽警示燈元件電路連接圖

實驗七、藍芽警示燈Arduino程式設計

本實驗將進行藍芽警示燈的Arduino程式設計，當前方有行人時，藍芽警示燈顯示文字為”前方有行人”，顯示色光為紅光；當前方無行人時，藍芽警示燈顯示綠色圓圈圖案，表示前方路口順暢。

✓ **實驗測試結果：**



Arduino程式碼

```
const long p01[ ] PROGMEM={顯示文字:前，色光:紅光};
const long p02[ ] PROGMEM={顯示文字:方，色光:紅光};
const long p03[ ] PROGMEM={顯示文字:有，色光:紅光};
const long p04[ ] PROGMEM={顯示文字:行，色光:紅光};
const long p05[ ] PROGMEM={顯示文字:人，色光:紅光};
const long p06[ ] PROGMEM={顯示:綠光圈，色光:綠光};
void loop() {
for(int i=0; i<NUM_LEDS; i++) {leds[i]=pgm_read_dword(&(p01[i]));}
FastLED.show();
for(int i=0; i<NUM_LEDS; i++) {leds[i]=pgm_read_dword(&(p02[i]));}
FastLED.show();
for(int i=0; i<NUM_LEDS; i++) {leds[i]=pgm_read_dword(&(p03[i]));}
FastLED.show();
for(int i=0; i<NUM_LEDS; i++) {leds[i]=pgm_read_dword(&(p04[i]));}
FastLED.show();
for(int i=0; i<NUM_LEDS; i++) {leds[i]=pgm_read_dword(&(p05[i]));}
FastLED.show();
for(int i=0; i<NUM_LEDS; i++) {leds[i]=pgm_read_dword(&(p06[i]));}
FastLED.show();}
```

實驗八、藍芽警示燈Python程式控制測試

本實驗透過Python程式進行藍芽警示燈控制測試，透過pynput套件偵測鍵盤訊號，由serial套件將控制訊號透過電腦藍芽傳送給藍芽警示燈，並進行警示標語的顯示。

✓ **實驗測試結果：**

三、行人警示系統整合

行人警示系統包含即時提醒車內駕駛與告知後方駕駛目前前方路口行人之情況，需要將『YOLO行人辨識系統』與『後方警示系統』整合在一起，才能完成『行人警示系統』。

實驗九、行人警示系統整合

此實驗使用實驗二訓練出的Yolov8模型(zc-2.0.pt)，並將實驗五的Yolov8影像辨識程式碼與實驗八的Python藍芽控制程式碼結合，此程式碼將可透過藍芽訊號控制Arduino開發版，搭配實驗八的Arduino程式碼就可啟動後方警示系統，並可於16×16 RGB 矩陣LED上顯示警示標語，完成告知車後方車輛駕駛任務。✓ **實驗測試結果：**

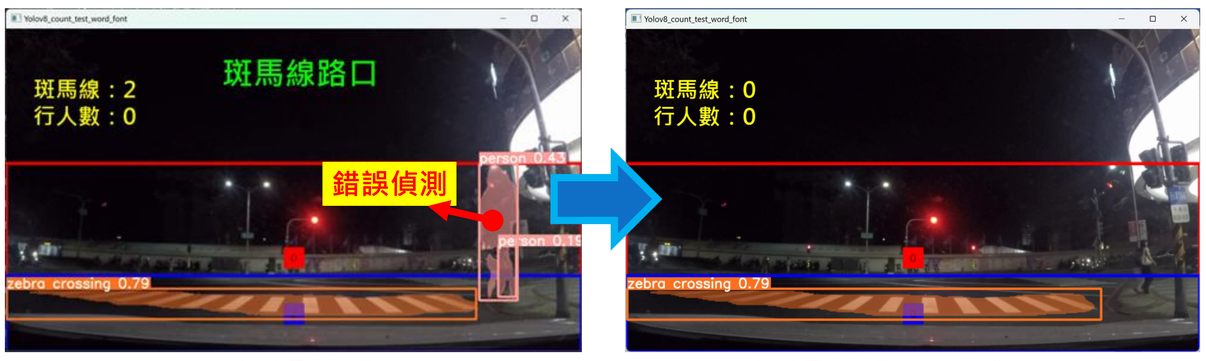
(斑馬線路口→亮綠圈)

(前方有行人→亮紅字)

實驗十、行人警示系統 - 行車記錄器影片實測

本實驗將行車記錄器影片載入行人警示系統進行測試，發現辨識模型(zc-2.0.pt)會有錯誤偵測的現象，設定conf值來提減少錯誤偵測的機會，發現將conf值設定在0.7可減少錯誤偵測的機會，提升辨識品質。

✓ **實驗測試結果：**



錯誤偵測　　無錯誤偵測(conf = 0.7)

實驗十一、行人警示系統 - 即時影像實測

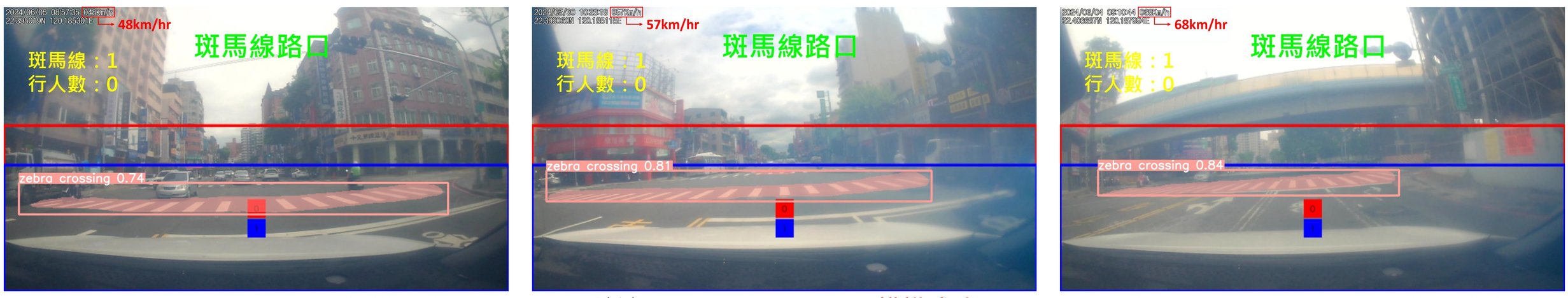
本實驗將行人警示系統安裝於汽車內進行上路實測，行人警示系統在行車過程中可不斷進行影像辨識，並於畫面中顯示警示標語，提醒車內駕駛前方路況，同時點亮後方藍芽警示燈，告知後車駕駛前方行人的情況。

✓ **實驗測試結果：**行人警示系統成功完成上路實測。

實驗十二、行人警示系統車輛速率實測

本實驗將於不同行車速率下，進行『行人警示系統』的實測，來了解在較高行車速率下(大於50km/hr)行人警示系統能否順利運作，行車速率控制在一般市區道路限速(60km/hr)及省道限速(70km/hr)以下進行實測。

✓ **實驗測試結果：**『行人警示系統』可於行車速率70km/hr以下成功完成影像辨識，可運用於一般市區道路與省道上。



時速：≦ 50 km/hr → **辨識成功**　　時速：50~60 km/hr → **辨識成功**　　時速：60~70 km/hr → **辨識成功**

實驗十三、行人警示系統辨識模型優化

本實驗加入行車記錄器畫面，進行遠距離物件影像標記，並進行模型訓練(zc-5.0.pt)與實測，使『行人警示系統』可於較高行車速率下提早提醒駕駛前方路況。

優化模型比較：

使用zc-5.0.pt之資料集進行Ultralytics HUB五個訓練模式的模型訓練，並與官方訓練之分割模型進行比較。我們所訓練模型在mAP(box50-95)值與mAP(mask50-95)值皆優於官方之模型，表示所訓練之模型皆有良好的辨識效果。

使用zc-5.0m.pt模型，可獲得次佳的mAP(box50-95) 值 0.638 與 最 佳 的 mAP(mask50-95)值0.591，並保有順暢的影像品質(幀率為41.10)。

✓ **實驗測試結果：**搭載zc-5.0.pt模型，可進行遠距離影像辨識，提早提醒駕駛前方路況。

訓練模型：zc-2.0.pt　　訓練模型：zc-5.0.pt → **可遠距離影像辨識**

Ultralytics HUB官方模型與本研究訓練之模型比較表(資料來源：研究者整理)

官方Model	mAP^{box}_{50-95}	mAP^{mask}_{50-95}	訓練Model	mAP^{box}_{50-95}	mAP^{mask}_{50-95}	幀率(FPS)
YOLOv8n-seg	0.367	0.305	zc-5.0n.pt	0.622	0.561	72.02
YOLOv8s-seg	0.446	0.368	zc-5.0s.pt	0.631	0.560	65.76
YOLOv8m-seg	0.499	0.408	zc-5.0m.pt	0.638	0.591	41.10
YOLOv8l-seg	0.523	0.426	zc-5.0l.pt	0.638	0.574	29.32
YOLOv8x-seg	0.534	0.434	zc-5.0x.pt	0.644	0.582	20.35

研究結果與討論

一、行人警示系統

本研究完成Yolov8模型(物件：行人、斑馬線)訓練，將所訓練之模型作為辨識模型。透過辨識區域內物件數量來判斷前方路況。若車輛前方有行人或斑馬線，於影像畫面中顯示警示標語，並發出警示語音，提醒車內駕駛；同時藍芽警示燈顯示警示標語，提醒後方駕駛前方路況。此系統透過影像辨識來判斷前方路口行人之狀況，同時提醒車內駕駛與後車駕駛，降低汽車與行人事故發生機率，降低前車因禮讓行人而被後車追撞的機會，因此我們成功開發出**『行人警示系統』**。

前方路況	影像畫面中顯示	警示語音	藍芽警示燈
行人	前方有行人	前方有行人	前方有行人
斑馬線	斑馬線路口	前方路口 請小心駕駛	綠色圓圈燈號
斑馬線 + 行人	斑馬線路口 前方路口有行人	前方路口有行人 請小心駕駛	前方有行人

二、研究應用推廣價值

配有AEB(緊急煞車輔助系統)之車輛約於2018年開始進入台灣市場，根據台灣交通部公路局統計，未裝有AEB之車輛數超過全台車輛總數的71%。本研究開發出的『行人警示系統』雖無法直接操作車輛煞車，但具即時提醒車內駕駛與後車駕駛之功能，此功能與AEB的警示功能類似，且安裝成本較AEB低，可運用於未裝有AEB之車輛(約占全台車輛總數的71%)，具有高度的應用推廣價值與市場價值。

三、行人警示系統優化與未來工作

1. 使用Yolov9、Yolov10進行行人與斑馬線的模型訓練。
2. 加入行人物件追蹤功能，計算出行人行走速度，推估行人過馬路的時間。
3. 開發『行人警示系統』App，將系統移植至手機及平板上，直接透過拍照鏡頭進行辨識，增加使用的方便性。

結論

1. 成功訓練出行人與斑馬線的Yolov8模型，並於影像中完成行人與斑馬線的實例分割。
2. 成功使用Supervision進行區域設定，計算出區域內辨識物件的數量。
3. 成功完成警示標語與警示語音的程式設計，透過條件設定，判斷前方路況後，顯示警示標語，發出警示語音。
4. 使用Arduino與RGB矩陣LED開發出藍芽警示燈，透過Arduino程式碼設計警示燈號。
5. 成功透過Python程式傳送藍芽訊號控制警示燈，並使後方警示燈點亮警示燈號。
6. 整合『YOLO行人辨識系統』與『後方警示系統』，完成『行人警示系統』開發。
7. 成功透過行車紀錄器影片進行『行人警示系統』測試，可設定conf 值降低錯誤偵測。
8. 成功將『行人警示系統』安裝於車輛上並進行上路測試，可即時偵測行人與斑馬線，於前方有行人或斑馬線時，顯示警示標語並發出警示語音提醒車內駕駛，透過後方警示燈亮起警示燈號，告知後方駕駛前方路況。
9. 已成功開發行人警示系統，安裝於車輛上可直接上路使用，希望此系統能降低行人與車輛發生交通事故的機率。

參考文獻

1. 張宸珣、王閔溱、吳尚原(2019)。**火車行駛安全辨識系統**。全國中小學科展作品。
2. 顏妤晅、陳以華、曹家瑜(2021)。**智慧影像，「One」視平安~大型車轉彎安全偵測警示系統**。全國中小學科展作品。
3. 蔡奕章(2021)。**AI影像辨識輔助視力量測系統**。全國中小學科展作品。
4. 楊雅盈、簡楷真(2022)。**「深」不可「測」——紅蓮燈魚3D座標重建與智慧管理系統**。全國中小學科展作品。