國立台灣大學

110 學年度物聯網導論
期末專題報告
智慧非吸菸區取締系統

學生 ： 何適楷 F06942018

蔡嘉倫 R10942015

指導老師：吳瑞北教授

賴怡吉教授

中華民國 111 年 1 月 13 日

目錄

一、 摘要

本專案提出一種無須人為監控針對非吸菸區取締吸菸行為的一種解決方案與系統架構。並依照所提出之架構，利用平價、易取得之物聯網技術實現。此專案使用 Google 所開發的 mobilenet_v2 物件辨識套件來偵測取締區域的人，使用 GP2Y1014AU 粉塵感測器作為 PM2.5 感測器，並使用 LoRa 傳送 PM2.5 的數據到真正的運算中心，本專案之解決方案能夠找到在取締區內的人之吸菸行為、拍下當事人吸菸行為、紀錄當時產生之 PM2.5，記錄犯行當下時間等。

關鍵字：吸菸取締、物件辨識、粉塵感測器、LoRa

二、介紹

為了維護國人學子的健康，目前各大專院校都實施全校禁菸的政策，高中職以下更是全面禁菸，依照「菸害防制法」第十三條第一項第九款、「學校衛生法」第二十四條規定：「高級中等以下學校，應全面禁菸」，兒童及少年福利法第二十六條亦規定：「不得吸菸、飲酒、嚼檳榔等行為」。為落實無菸校園並預防青少年吸菸，國民健康局結合縣市衛生局、教育局與專家學者等資源，共同擬訂適合各縣市的策略與模式，推動「無菸校園計畫」，透過學校自主營造無菸校園，並擴展到家庭與社區，齊力以具體行動支持無菸害的學習環境。[1]然而該政策已經實施十餘年，在台灣第一首府台大各處還是依然看到、或察覺到有人在某些角落、樓梯間、頂樓等違規抽菸，這些區域往往偏僻，少人經過，難以取締，除非派特殊人力特別定期取締，然而此種方式勞師動眾，在人力成本極高的情況下，這種方案並不實際，架設監視器也是一種方案，但是一個禮拜有 168 小時的影像的情況下，也是需要大量的人力去檢查，所以又回歸到人力成本的問題，故本專案致力於利用最低的成本來達到取締、偵測、辨識等功能。

本專案所提出的解決方案乃是用攝影機來偵測在禁菸取締區滯留的人，與 GP2Y1014AU 粉塵感測器做為該滯留人是否有抽菸的依據，本專案的系統架構會在第三章詳細敘述，第四章則介紹影像辨識與粉塵感測器之細節，第五章展示實作的成果，第六章總結上述章節並探討本專案未來之展望。

三、系統架構

本系統分為兩個部分，一部分架設在取締區，一部分架設在 server 端(辦公室)，架設在取締區的設備有：攝影機、粉塵感測器、LoRa(發射)，server 端的設備有：PC(筆電或其他運算單元)、LoRa(接收)、AP 基地台。

其中攝影機內建 wifi 模組，可以藉由智慧型手機設定並連結到 server 端的 AP 基地台，其中 PC 也會連到該 AP 基地台，如此一來 PC 跟該攝影機就在同一個區域網路(LAN)裡面，PC 就能夠藉由區域網路跟攝影機內建的 API 來存取即時的影像。

粉塵感測器則由 Arduino 使用類比 port 讀取讀值，再藉由 LoRa 模組傳回 PC 端的 LoRa，PC 一樣從 USB port 讀取 Ardino/LoRa 模組傳回來的資訊。

其系統架構圖如圖 3.1。

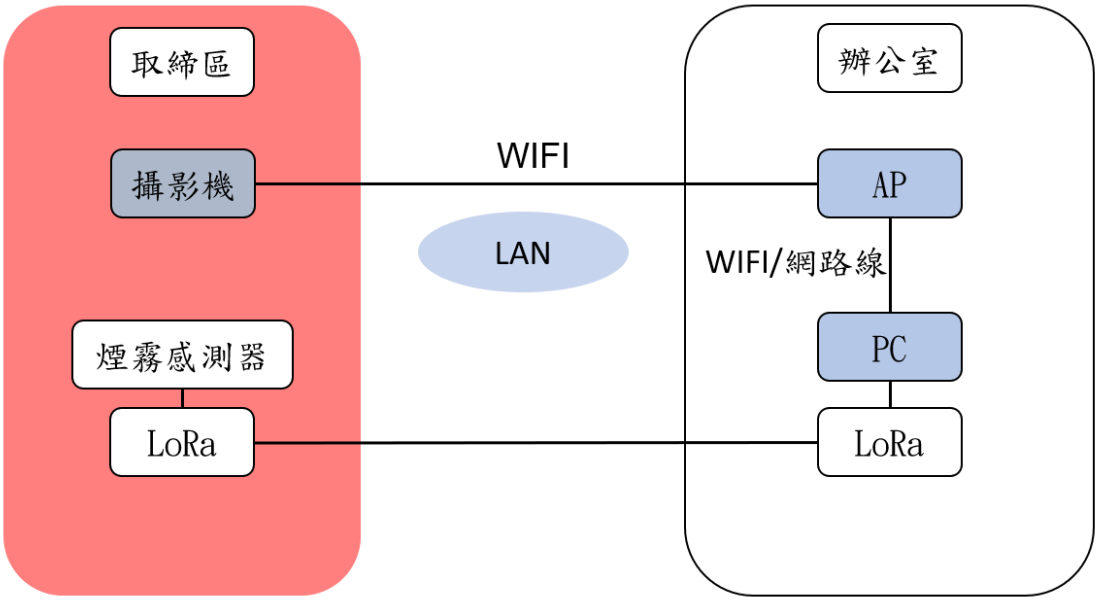

圖 3.1：本計畫之系統架構

在主程式中可藉由區域網路存取攝影機之影像後，對影像中的物件進行辨識，如果辨識到人的話，則可以進一步確認粉塵感測器讀值是否正常，最終判斷是否有違規行為，如有則會儲存照片跟當時的時間、PM2.5 讀值。

四、影像辨識與粉塵感測器

在本專案使用的影像辨識模型是採用 Google 開發的 mobilenet_v2[2]，此模型是基於開源資料庫 OpenImagesV4 所學習，該模型具備強大的性能，能夠辨識 600 種常見的物件，在本專案中，我們使用其中標記為「人」的分類來做為判斷取締區是否有人，該模型使用方式如圖 4.1：

```
# Apply image detector on a single image.
detector = hub.Module("https://tfhub.dev/google/openimages_v4/ssd/mobilenet_v2/1")
detector_output = detector(image_tensor, as_dict=True)
class_names = detector_output["detection_class_names"]
```

圖 4.1：mobilenet_v2 的使用方式

該模型輸入為一個 3 個 channel 的圖片，輸出有判斷物件的方框、判斷物件的名字、判斷物件的分數。

從區域網路讀取攝影機的圖檔後，我們轉成 3 個 channel 的圖片，將其輸入到 mobilenet_v2 中判斷圖中的物件，並數出圖片中人數，輸出到即時顯示的圖表；同時 PC 也從 LoRa 端讀到當時的 PM2.5 讀值，並也同時顯示到圖表上，如果讀值超乎所設定閥值，且攝影機的人數也大於一人，則程式則會把圖片、當時的時間與 PM2.5 讀值寫入硬碟中，做為日後取締的證據，圖 4.2 為程式的流程圖。管理員只要定期檢查該資料夾就能看到違規者違規當下的照片。圖 4.3 為進行取締後將圖片存入資料夾的截圖。

主程式原始碼可以參考附錄。

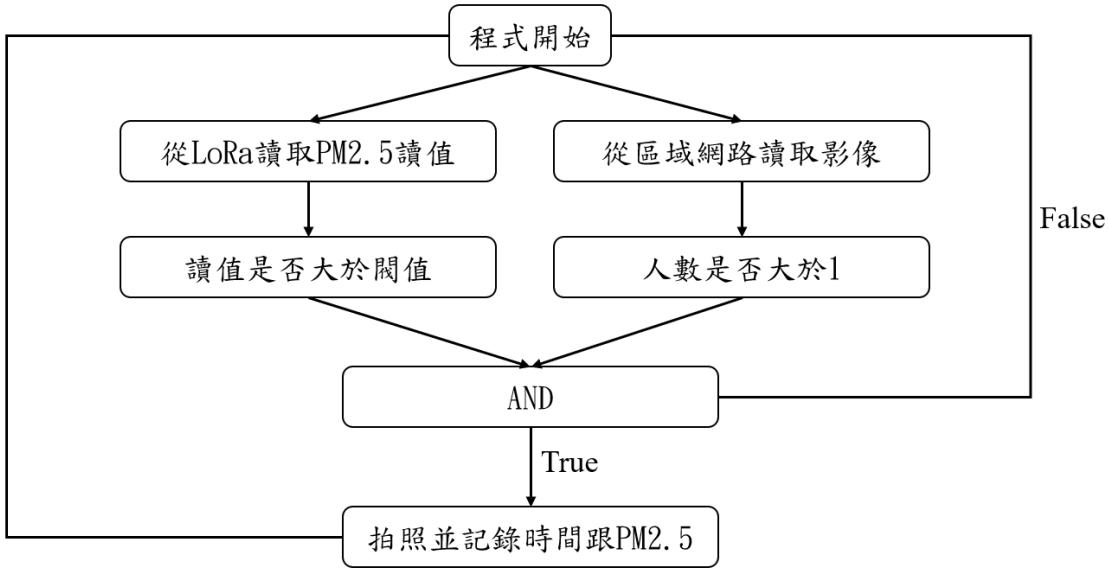

圖 4.2：主程式之系統流程圖

圖 4.3：取締後將違規者照片存入硬碟

五、實作成果

本專案實作場地為台大電機二館 502 室旁邊通往頂樓的樓梯間，這邊為電機二館有名的違規熱區，基於安全理由，我們將攝影機架設在底部角落，基本上不影響物件判斷，粉塵感測器則放在中央手把處，因為學校全面禁菸，我們用常見的酒精噴霧劑做為抽菸時產生的粉塵。圖 5.1 為實際架設圖：

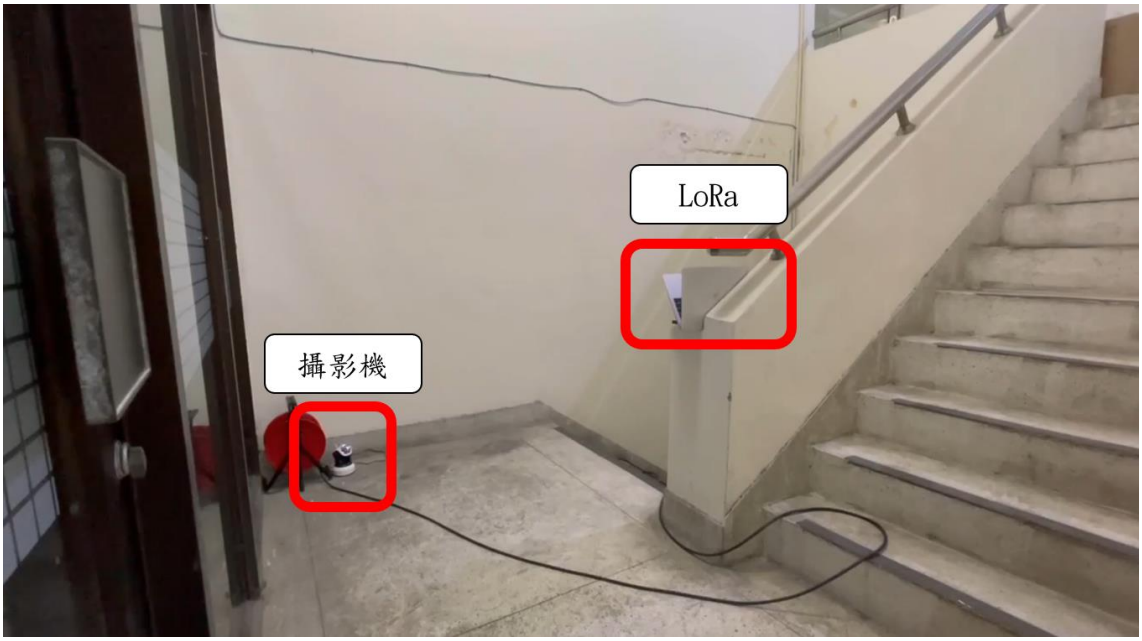

圖 5.1：本專案實作實際架設圖

從前章討論本專案主要著重在節省大量人力成本來取締、看顧，故本實驗希望對於抽菸誤報率降到最低，我們設想三種基礎情境，一、無人且無煙霧，二、有人無煙霧，三、有人有煙霧，理想情況下，情境一與二不希望有誤報的情況，情況三不希望有沒報的情況。以上整理成表 5.1：

	有煙霧	無煙霧
有人	三、需警報	二、無須警報
無人	X	一、無須警報

表 5.1：三種基礎情境

經過實驗以上三種情境都經過測試，圖 5.2 為情境一測試圖，右上角為 server 端同步錄影畫面。可以看到 PM2.5 處在良好的狀態，也未判斷出有人經過。

圖 5.3 為情境二測試圖，可以看到 PM2.5 處在良好的狀態，但是有判斷出人的存在。

圖 5.4 為情境三測試圖，可以看到 PM2.5 狀態非常差，而且有判斷出人員的存在，故該情況會被 server 紀錄。

圖 5.5 為情境三發生時 server 所記錄下來的照片。

圖 5.6 為情境三發生時 server 所記錄下來的時間與 PM2.5。

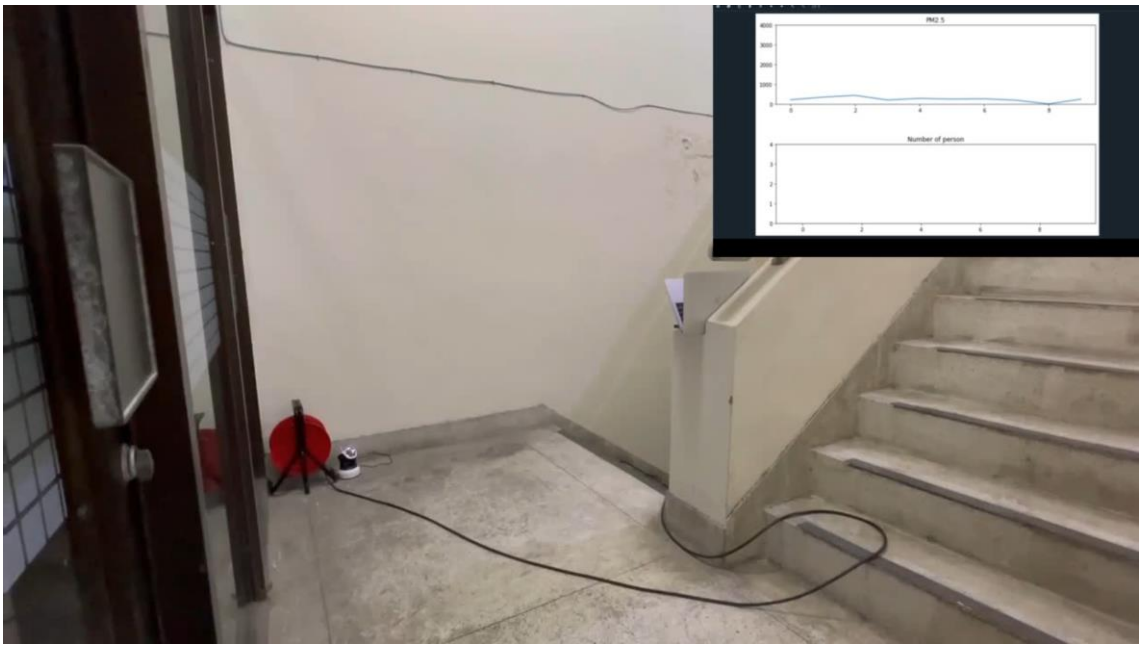

圖 5.2：情境一實驗情形

圖 5.3：情境二實驗情形

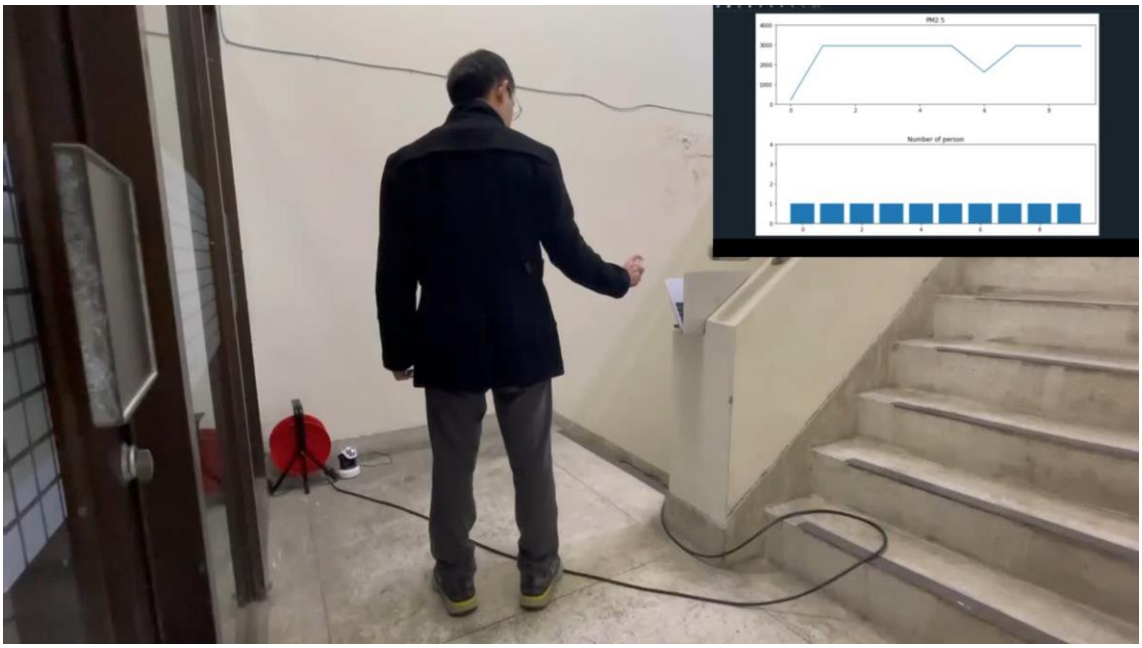

圖 5.4：情境三實驗情形

圖 5.5：情境三下捕捉到違規者的照片

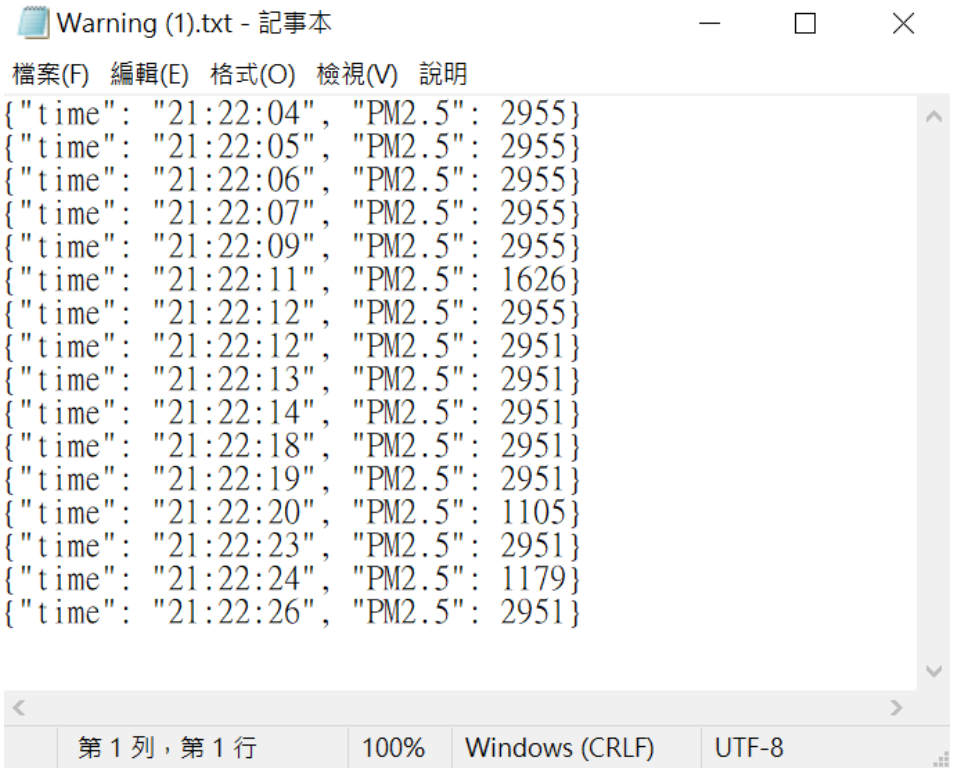

圖 5.6：情境三下紀錄的違規詳細資訊

六、結論

本專案成功整合影像辨識與粉塵感測器，設計出一個無人監督之非吸菸區取締系統，並能正確分辨三種不同情境，並正確儲存違規者的照片與其詳細資訊。

未來商業化的話，可根據客戶環境做應變，例如：

1. 擷取客戶現有的攝影機，而不需自備攝影機
2. 根據客戶 server 端的效能，使用複雜度不同的 model
   e.g. inception_resnet_v2 這個 model 可以再運算效能更低的裝置運行
3. 使用 MQTT broadcast 到網路中，有需求者則可以訂閱以取得即時資訊。

成本分析	
煙霧偵測器	335 元
攝影機	1500 元
Arduino UNO	188 元
LoRa Shield	750 元
總計	2773 元

表 6.1：成本分析

商業價值分析	
全台高中職	787 所
全台大學	126 所
總計	913
假設採用率 10%，每所學校平均設置 4 個菸害取締區，則銷售約 365 台，則至少產生 1,012,700 銷售額。	

表 6.2：商業價值分析

七、參考資料

1. https://www.hpa.gov.tw/Pages/Detail.aspx?nodeid=1124&pid=1604

2. https://tfhub.dev/google/openimages_v4/ssd/mobilenet_v2/1

八、附錄

本章節為實作程式之原始碼。

```
# Runs with stable version tensorflow 2.1.0.
# For running inference on the TF-Hub module.
import tensorflow as tf
import tensorflow_hub as hub
from tkinter import messagebox

gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)
# tf.config.experimental.set_memory_growth(gpus[0], True)    #tf 2.1
# compatibility issue:
https://github.com/tensorflow/tensorflow/issues/37233

# For downloading the image.
import matplotlib.pyplot as plt

# For drawing onto the image.
import numpy as np
from PIL import Image, ImageColor, ImageDraw, ImageFont, ImageOps

# For measuring the inference time.
import time
import cv2
import imageio
import threading

import serial  # 引用 pySerial 模組
import re
import sys, json
from datetime import datetime

import tkinter as tk
```

```
import matplotlib.pyplot as plt
import ctypes  # An included library with Python install.
from easygui import msgbox
from datetime import datetime
fi_num = datetime.now().strftime("%H_%M_%S")
# Print Tensorflow version
print(tf.__version__)

# Check available GPU devices.
print("The following GPU devices are available: %s" %
tf.test.gpu_device_name() or 'n.a.')

# url = 'http://192.168.1.49:8001/?action=stream'
#url = 'rtsp://192.168.2.3:554/ipcam_mjpeg.sdp'
url
='http://192.168.50.205:31697/snapshot.cgi?user=admin&pwd=12345678&164
16150949550.7179290013837389&fbclid=IwAR3dN4ie7Td2XPL639nEdzux6s1aPYqV
mcITXOsmbLMEcUchSTt5AYHURLc'
#url = '
rtsp://admin:12345678@192.168.1.214:10554/chID=1&streamtype=main&linkT
ype=tcp'
#url = ' rtsp://admin:12345678@192.168.1.214:10554/ipcam_mjpeg.sdp'

def display_image(image):
    fig = plt.figure(figsize=(20, 15))
    plt.grid(False)
    plt.imshow(image)


def draw_bounding_box_on_image(image, ymin, xmin, ymax, xmax, color,
                               font, thickness=4, display_str_list=()):
    """Adds a bounding box to an image."""
    draw = ImageDraw.Draw(image)
    im_width, im_height = image.size
    (left, right, top, bottom) = (xmin * im_width, xmax * im_width,
```

```
                                ymin * im_height, ymax * im_height)
    draw.line([(left, top), (left, bottom), (right, bottom), (right,
top),
               (left, top)], width=thickness, fill=color)

    # If the total height of the display strings added to the top of
the bounding
    # box exceeds the top of the image, stack the strings below the
bounding box
    # instead of above.
    display_str_heights = [font.getsize(ds)[1] for ds in
display_str_list]
    # Each display_str has a top and bottom margin of 0.05x.
    total_display_str_height = (1 + 2 * 0.05) *
sum(display_str_heights)

    if top > total_display_str_height:
        text_bottom = top
    else:
        text_bottom = bottom + total_display_str_height
    # Reverse list and print from bottom to top.
    for display_str in display_str_list[::-1]:
        text_width, text_height = font.getsize(display_str)
        margin = np.ceil(0.05 * text_height)
        draw.rectangle([(left, text_bottom - text_height - 2 * margin),
                        (left + text_width, text_bottom)], fill=color)
        draw.text((left + margin, text_bottom - text_height - margin),
                  display_str, fill="black", font=font)
        text_bottom -= text_height - 2 * margin


def draw_boxes(image, boxes, class_names, scores, max_boxes=10,
min_score=0.25):
    """Overlay labeled boxes on an image with formatted scores and
label names."""
```

```
    colors = list(ImageColor.colormap.values())
    font = ImageFont.load_default()

    for i in range(min(boxes.shape[0], max_boxes)):
        if scores[i] >= min_score:
            ymin, xmin, ymax, xmax = tuple(boxes[i])
            display_str = "{}:
{}%".format(class_names[i].decode("ascii"), int(100 * scores[i]))
            color = colors[hash(class_names[i]) % len(colors)]
            image_pil = Image.fromarray(np.uint8(image)).convert("RGB")
            draw_bounding_box_on_image(image_pil, ymin, xmin, ymax,
xmax, color, font, display_str_list=[display_str])
            np.copyto(image, np.array(image_pil))
    return image

def count_person(class_names, scores):
    """Overlay labeled boxes on an image with formatted scores and
label names."""
    image_filter = class_names[np.argwhere(scores>0.25)]
    counter = 0
    for i in range(len(image_filter)):
        if image_filter[i,0] == b'Clothing':
            counter = counter + 1
    return counter

module_handles =
["https://tfhub.dev/google/openimages_v4/ssd/mobilenet_v2/1",
"https://tfhub.dev/google/faster_rcnn/openimages_v4/inception_resnet_v
2/1", "https://tfhub.dev/tensorflow/tfjs-model/blazeface/1"]
module_handle = module_handles[0]

detector = hub.load(module_handle).signatures['default']


COM_PORT = 'COM13'    # 指定通訊埠名稱
```

```
BAUD_RATES = 9600    # 設定傳輸速率 #使用 USB 連線序列口
ser = serial.Serial(COM_PORT, BAUD_RATES)
counter=0
history_data_pm25 = []
history_data_person = []
person_index_counter = 0
def read_data():
    while(1):
        data_raw = ser.readline().decode().rstrip()
        data_raw = data_raw.split(" ")
        if len(data_raw)>=5:
            break
    flag=0
    print("PM2.5:",data_raw[4])
    data_raw =int(data_raw[4])
    if(data_raw>1000):
        flag=1
        #print('warning')
    return flag,data_raw


def pop_up():
    root = tk.Tk()

    text = tk.Text(root, width=20, height=5)
    text.pack()

        # 設定 tag
    text.tag_config("tag_1", backgroun=None, foreground="red")

        # "insert" 索引表示插入游標當前的位置
    text.insert("end", "Someone is smoking", "tag_1")

    root.mainloop()
```

```
try:
    t1 = time.time()
    while True:
        cam = imageio.imread(url)
        t1_last = t1
        t1 = time.time()
        #ret, img = cam.read()
        buffer_time = time.time() - t1
        fps = 1. / max((t1 - t1_last), 0.001)

        #if not ret:
            #break
        flag , data_raw =read_data()
        color_image = np.asanyarray(cam)
        color_image=color_image[:,:,::-1]
        #print(color_image.shape)
        converted_img  = tf.image.convert_image_dtype(color_image,
tf.float32)[tf.newaxis, ...]
        t2 = time.time()
        result = detector(converted_img)
        infer_time = time.time() - t2

        result = {key:value.numpy() for key, value in result.items()}

        print("Found %d objects." % len(result["detection_scores"]))
        print('Inference time:', infer_time, 'Buffer delay:',
buffer_time, 'FPS:', fps)

        image_with_boxes = draw_boxes(color_image,
result["detection_boxes"],
                result["detection_class_entities"],
result["detection_scores"])
        num_person = count_person(result["detection_class_entities"],
result["detection_scores"])
        print("num_person:",num_person)
```

```
        if(num_person and flag):
            counter=counter+1
            counter=counter%10
            ti = datetime.now().strftime("%H:%M:%S")
            #print('---Time---: ', ti)
            with open('Warning.txt', 'a') as fout:
                json.dump({'time': ti, 'PM2.5':data_raw }, fout)
                fout.write('\n')

imageio.imwrite('smoking_person{}_PM25_{}.jpg'.format(str(counter),str
(data_raw)),cam)
            print("someone is smoking")


        fig, (ax1, ax2, ax3, ax4) = plt.subplots(nrows=1,
ncols=4,figsize=(11, 7))

        grid = plt.GridSpec(2, 1, wspace=0.2, hspace=0.5)
        ax1 = plt.subplot(grid[0, 0])
        plt.ylim(0,4000)

        ax1.title.set_text('PM2.5')
        history_data_pm25.append(data_raw)

        if len(history_data_pm25)>10:
            history_data_pm25.pop(0)
        plt.plot(history_data_pm25)

        ax3 = plt.subplot(grid[1, :1])
        ax3.title.set_text('Number of person')
        history_data_person.append(num_person)
        plt.ylim(0,4)
        if len(history_data_person)>10:
            history_data_person.pop(0)
```

```
        plt.bar(range(len(history_data_person)),history_data_person,
width=0.8, bottom=None)
        plt.show()
        cv2.namedWindow('IP_cam + Tensorflow Object Detection',
cv2.WINDOW_AUTOSIZE)
        cv2.imshow('IP_cam + Tensorflow Object Detection',
image_with_boxes)
        key = cv2.waitKey(1)
        # Press esc or 'q' to close the image window
        if key & 0xFF == ord('q') or key == 27:
            break
finally:
    cv2.destroyAllWindows()
```