題目：英雄辨識後直接顯示英雄電影介紹

目錄

摘要 1-1

本專題旨在設計一款人臉辨識組，它可以在掃瞄出不同電影角色後，跳轉出該英雄之電影及簡介，此裝置分兩裝置 HUB 8735 Ultra 以及執行程式之電腦，透過 HUB 8735 Ultra 鏡頭模組讓程式辨識人物後執行跳轉介紹之功能。

介紹程式 1-2

設計上我們使用較為簡單直接的方式，由 Arduino 進行辨識後，發送 HTTP 請求到先前架設好之 Python 伺服器，再由伺服器開啟特定網頁，來達成掃瞄出英雄角色後跳轉至該角色之見紹網站

介紹元件 2-1

2-1-1

HUB 8735 ultra，Smart AI CAM 是具備多功能影像處理的高度集成模組，內置 NPU AI 運算引擎，加速處理 AI 模型以及 802.11 a/b/g/n 雙頻 Wi-Fi 與 BLE 低耗電藍牙傳輸，可廣泛應用於各種結合影像識別或 AI 運算之物聯網場域，適用於智能家庭，工業物聯網，智慧零售，健康照護或是車用電子等場景；多款 Pre-trained AI models 已最佳化在模組直接運行，可做 為 AI 教學之體驗工具，亦可直接整合在產品設計中作為快速導入 Edge AI 應用的快製套件。

HUB 8735 規格

2-1-2

處理器：RTL8735 B AIOT 國產晶片
影像輸入：搭配國產 Full HD1080 P CMOS 感測
語音輸入：內建 MIC 語音輸入功能
高感度數位 MIC
儲存裝置：支援 SD 記憶卡
無線連通：Wi-Fi 2.4GHz/5GHz
Bluetooth BLE
無線影像串流
影像壓縮：H.264/265
AI 處理：提供多種 pre-trained AI models 供快速上手
自已客製 AI Models
UART 介面：提供多組 UART 外接週邊
提供與 ESP32 CAM 相同的 AT Command
USB 介面：USB 燒入、debug
USB 影像輸出
LED：補光 LED
I/O 擴充板：依照開發者需求擴充功能。如 IMU 感測器、超聲波感測器、喇叭語音輸出、溫度·震動、濕度等功能

執行步驟

1. 準備好 HUB 8735 Ultra 以及電腦裝置，並確認裝置能正常使用。

2. 使用 Type-C 傳輸線連接 HUB 8735 Ultra 及電腦。HUB 8735 Ultra 可以使用 Type-C 傳輸線來進行燒錄執行之功能，在使用上比 HUB 8735 更簡單方便。

3. 開啟 Arduino 執行程式。

程式碼:

```
#include "WiFi.h"
#include "StreamIO.h"
#include "VideoStream.h"
#include "RTSP.h"
#include "NNObjectDetection.h"
#include "VideoStreamOverlay.h"
#include "ObjectClassList.h"
#include "WiFiClient.h"

char ssid[] = " network SSID ";    // your network SSID (name)
char password[] = " network password ";   // your network
password
WiFiClient client;

#define CHANNEL   0
#define CHANNELNN 3
#define NNWIDTH  576
#define NNHEIGHT 320

VideoSetting config(VIDEO_FHD, 30, VIDEO_H264, 0);
VideoSetting configNN(NNWIDTH, NNHEIGHT, 10, VIDEO_RGB, 0);
NNObjectDetection ObjDet;
RTSP rtsp;
StreamIO videoStreamer(1, 1);
StreamIO videoStreamerNN(1, 1);

IPAddress ip;
int rtsp_portnum;

bool ironManRequestSent = false;
bool caRequestSent = false;
unsigned long lastRequestTime = 0;
const unsigned long requestInterval = 10000; // 10 seconds

void setup()
{
    Serial.begin(115200);
```

```
    // Attempt to connect to WiFi network:
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print("Attempting to connect to WPA SSID: ");
        Serial.println(ssid);
        WiFi.begin(ssid, password);
        delay(2000);
    }
    ip = WiFi.localIP();

    // Configure camera video channels with video format
information
    // Adjust the bitrate based on your WiFi network quality
    config.setBitrate(2 * 1024 * 1024);    // Recommend to use
2Mbps for RTSP streaming to prevent network congestion
    Camera.configVideoChannel(CHANNEL, config);
    Camera.configVideoChannel(CHANNELNN, configNN);
    Camera.videoInit();

    // Configure RTSP with corresponding video format
information
    rtsp.configVideo(config);
    rtsp.begin();
    rtsp_portnum = rtsp.getPort();

    // Configure object detection with corresponding video
format information
    // Select Neural Network(NN) task and models
    ObjDet.configVideo(configNN);
    ObjDet.setResultCallback(ODPostProcess);
    ObjDet.modelSelect(OBJECT_DETECTION, DEFAULT_YOLOV4TINY,
NA_MODEL, NA_MODEL);
    ObjDet.begin();

    // Configure StreamIO object to stream data from video
channel to RTSP
    videoStreamer.registerInput(Camera.getStream(CHANNEL));
    videoStreamer.registerOutput(rtsp);
```

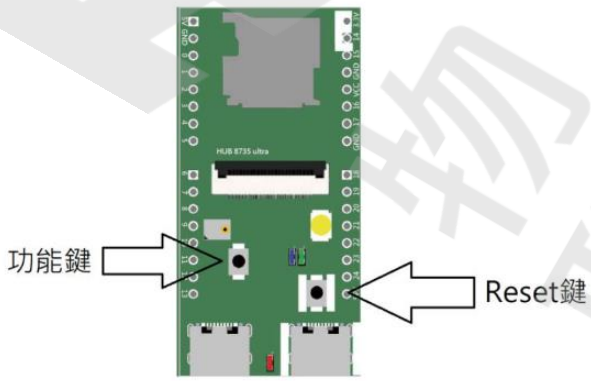
```
    if (videoStreamer.begin() != 0) {
        Serial.println("StreamIO link start failed");
    }

    // Start data stream from video channel
    Camera.channelBegin(CHANNEL);

    // Configure StreamIO object to stream data from RGB video
channel to object detection
    videoStreamerNN.registerInput(Camera.getStream(CHANNELNN));
    videoStreamerNN.setStackSize();
    videoStreamerNN.setTaskPriority();
    videoStreamerNN.registerOutput(ObjDet);
    if (videoStreamerNN.begin() != 0) {
        Serial.println("StreamIO link start failed");
    }

    // Start video channel for NN
    Camera.channelBegin(CHANNELNN);

    // Start OSD drawing on RTSP video channel
    OSD.configVideo(CHANNEL, config);
    OSD.begin();
}

void loop()
{
    if ((ironManRequestSent || caRequestSent) && (millis() -
lastRequestTime >= requestInterval)) {
        ironManRequestSent = false; // Reset flag after 10
seconds
        caRequestSent = false; // Reset flag after 10 seconds
    }
    // Do nothing else in loop
}

// User callback function for post processing of object
detection results
```

```
void ODPostProcess(std::vector<ObjectDetectionResult> results)
{
    uint16_t im_h = config.height();
    uint16_t im_w = config.width();

    Serial.print("Network URL for RTSP Streaming: ");
    Serial.print("rtsp://");
    Serial.print(ip);
    Serial.print(":");
    Serial.println(rtsp_portnum);
    Serial.println(" ");

    printf("Total number of objects detected = %d\r\n",
ObjDet.getResultCount());
    OSD.createBitmap(CHANNEL);

    if (ObjDet.getResultCount() > 0) {
        for (int i = 0; i < ObjDet.getResultCount(); i++) {
            int obj_type = results[i].type();
            if (itemList[obj_type].filter) {    // check if
item should be ignored

                ObjectDetectionResult item = results[i];
                // Result coordinates are floats ranging from
0.00 to 1.00
                // Multiply with RTSP resolution to get
coordinates in pixels
                int xmin = (int)(item.xMin() * im_w);
                int xmax = (int)(item.xMax() * im_w);
                int ymin = (int)(item.yMin() * im_h);
                int ymax = (int)(item.yMax() * im_h);

                // Draw boundary box
                printf("Item %d %s:\t%d %d %d %d\n\r", i,
itemList[obj_type].objectName, xmin, xmax, ymin, ymax);
                OSD.drawRect(CHANNEL, xmin, ymin, xmax, ymax,
3, OSD_COLOR_WHITE);
```

```
                // Print identification text
                char text_str[20];
                snprintf(text_str, sizeof(text_str), "%s %d",
itemList[obj_type].objectName, item.score());
                OSD.drawText(CHANNEL, xmin, ymin -
OSD.getTextHeight(CHANNEL), text_str, OSD_COLOR_CYAN);

                // Open browser and navigate to specific
webpage when "Iron man" or "CA" object is detected
                if (strcmp(itemList[obj_type].objectName, "Iron
man") == 0 && !ironManRequestSent) {
                    if (!client.connect("172.20.10.2", 8888)) {
                        Serial.println("Connection to Python
server failed");
                        return;
                    }
                    // Make a HTTP request to Python server to
open browser:
                    client.println("GET /open_browser1
HTTP/1.1");
                    client.print("Host: 172.20.10.2\r\n");
                    client.println("Connection: close\r\n");
                    client.println();

                    ironManRequestSent = true; // Set flag
after sending HTTP request
                    lastRequestTime = millis(); // Record the
time of the request
                }

                if (strcmp(itemList[obj_type].objectName, "CA")
== 0 && !caRequestSent) {
                    if (!client.connect("172.20.10.2", 8888)) {
                        Serial.println("Connection to Python
server failed");
                        return;
                    }
```

```
                    // Make a HTTP request to Python server to
open browser:
                    client.println("GET /open_browser2
HTTP/1.1");
                    client.print("Host: 172.20.10.2\r\n");
                    client.println("Connection: close\r\n");
                    client.println();

                    caRequestSent = true; // Set flag after
sending HTTP request
                    lastRequestTime = millis(); // Record the
time of the request
                }
            }
        }
    }
    OSD.update(CHANNEL);
```

確認程式無異狀後，先將 HUB 8735 Ultra 按 Reset 鍵，再按住功能鍵不放按 Reset 鍵來開啟燒錄模式，再將程式燒錄至 HUB 8735 Ultra 來執行辨識物件之工作。(注意要將 10、11 行修改成電腦連線之網路)

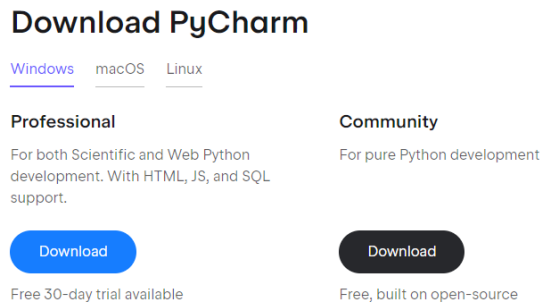


4. 安裝 Python

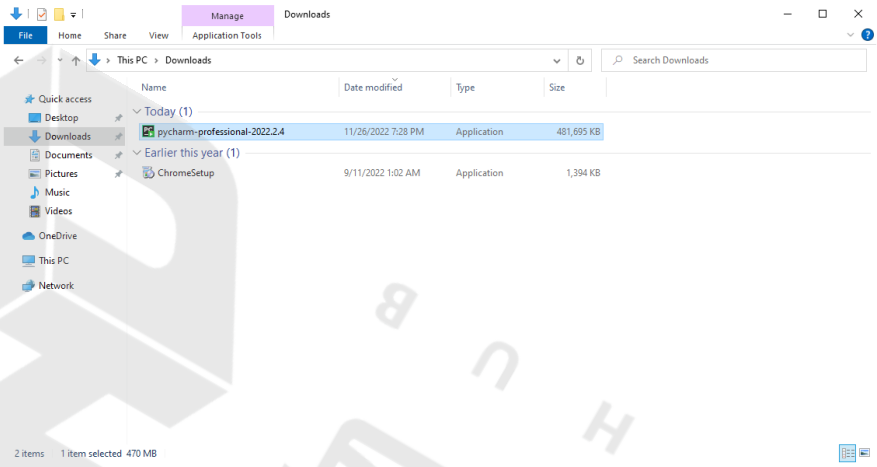
這裡我採用 PyCharm 應用程式，因為組員較熟悉此程式之介面，所以採用。

1. 官方網站下載安裝包

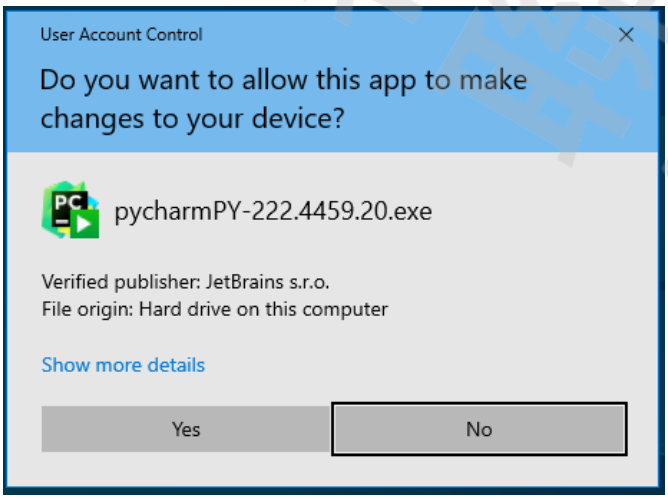
2. 選擇 Professional 版本(Community 也可以



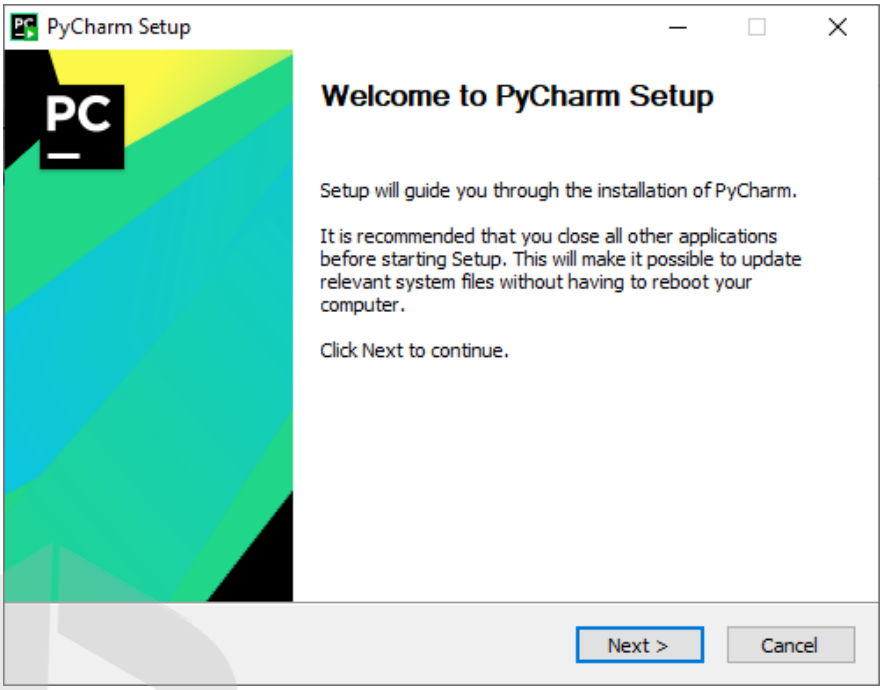
3. 打開安裝檔



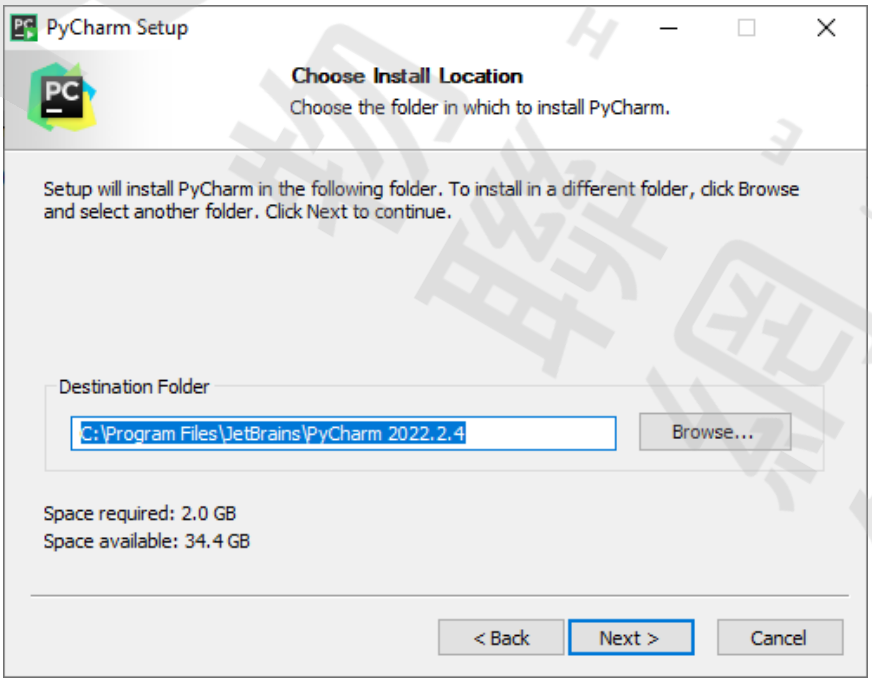
4. 出現 UAC 警告，點擊"Yes"

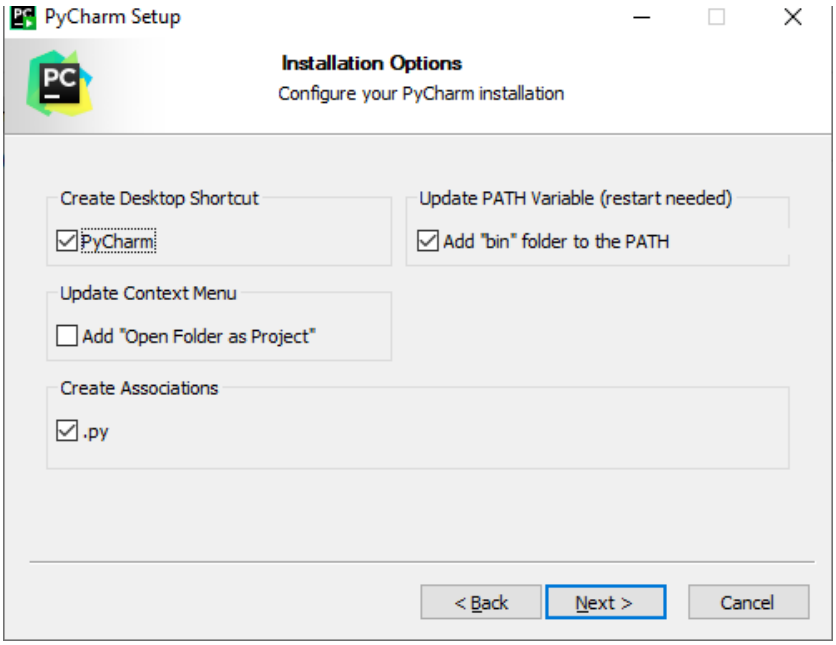


5. 點擊下一步

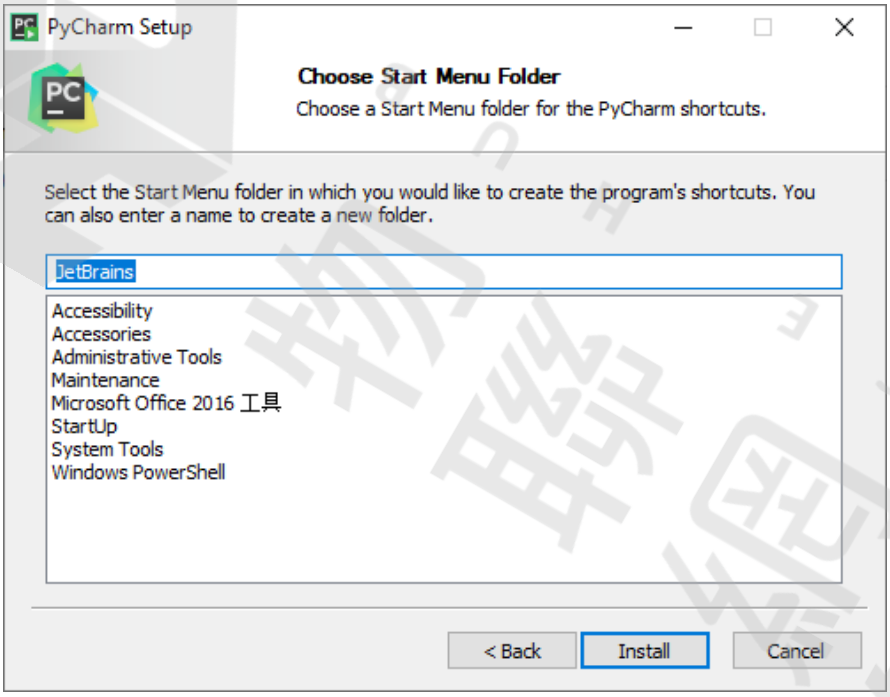


6. 安裝路徑使用預設路徑即可

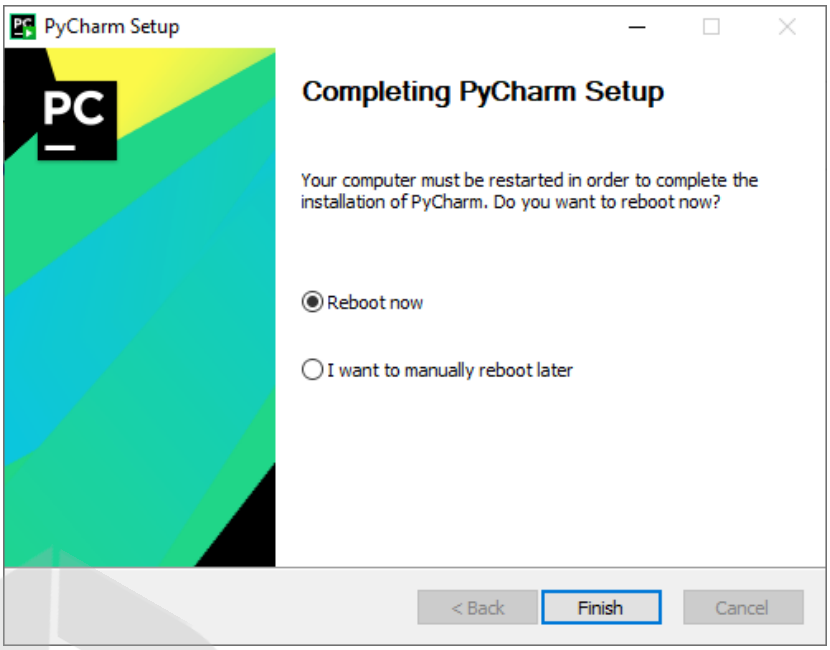


7. 

8. 點選"Install"等候安裝完畢



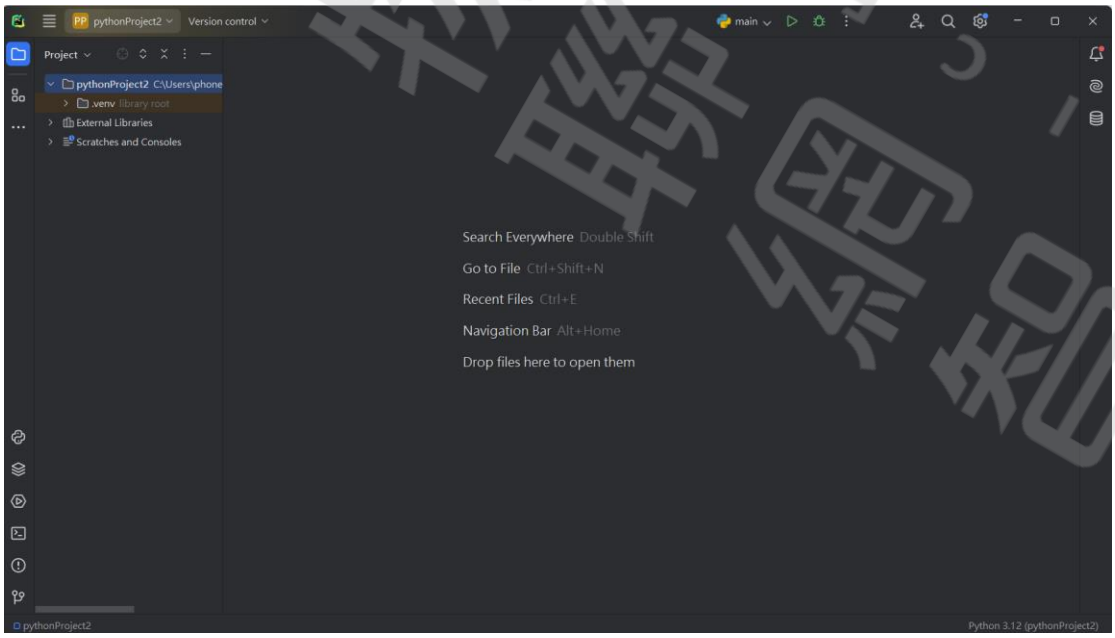
9. 選擇"Rebot Now"，重新開機完成安裝



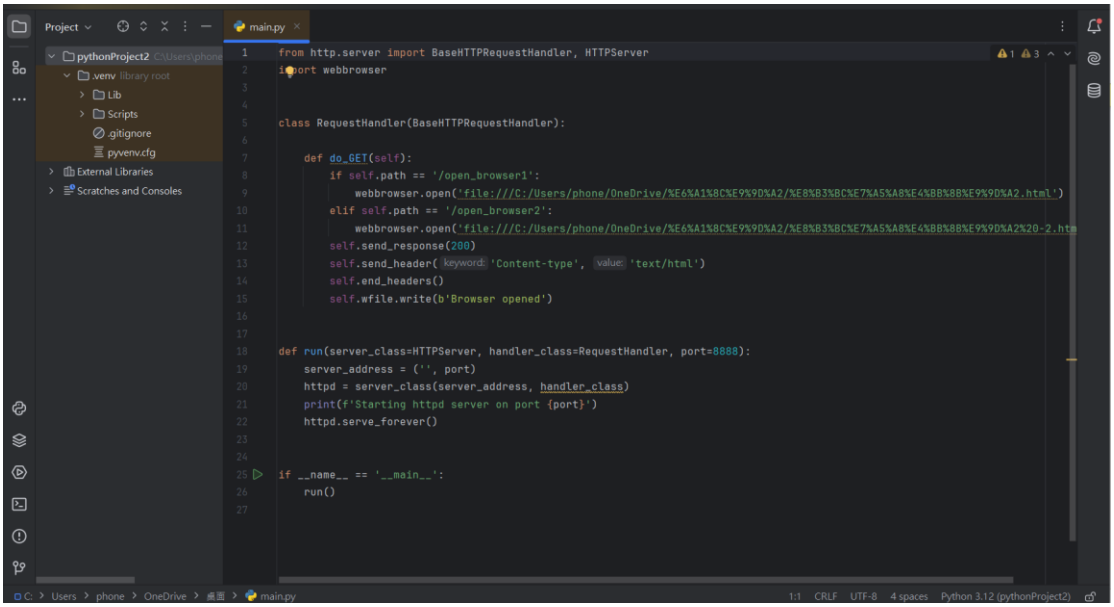
10.到官網註冊帳號後即可登入使用。

5. 架設 Python 伺服器

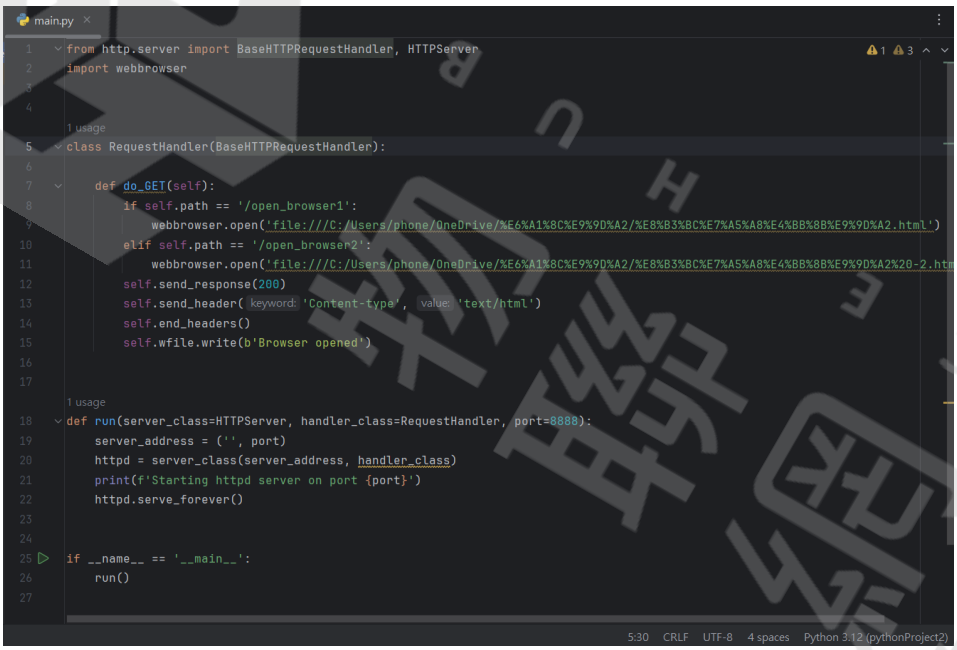
1. 開啟 Pycharm 應用程式



2.



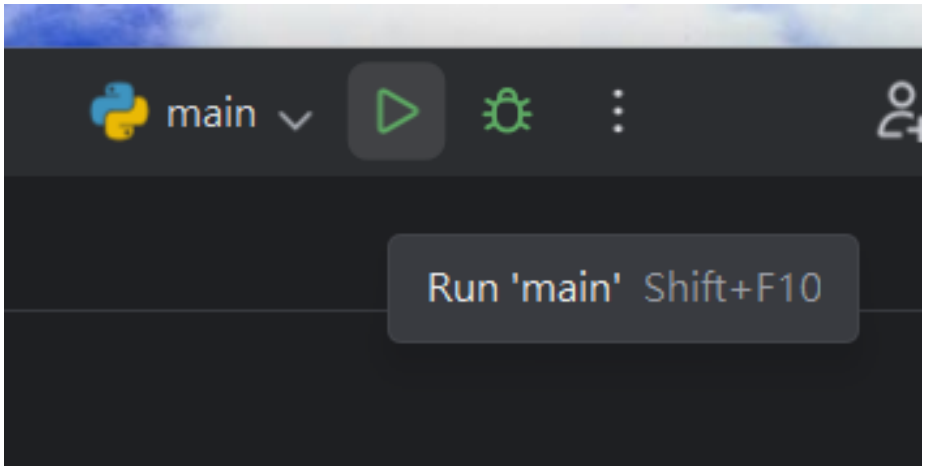
3. 確認程式碼無誤，如下



程式碼:

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import webbrowser

class RequestHandler(BaseHTTPRequestHandler):

    def do_GET(self):
        if self.path == '/open_browser1':
            webbrowser.open('愈開啟網站之網址)')
        elif self.path == '/open_browser2':
            webbrowser.open(愈開啟網站之網址)
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()
        self.wfile.write(b'Browser opened')


def run(server_class=HTTPServer, handler_class=RequestHandler,
port=8888):
    server_address = ('', port)
    httpd = server_class(server_address, handler_class)
    print(f'Starting httpd server on port {port}')
    httpd.serve_forever()


if __name__ == '__main__':
    run()
```

4. 確認程式沒問題後，點擊右上角”執行”或是 Shift+F10 執行程式來開啟伺服器(注意此伺服器必須和 Arduino 端為同一網域才能進行指令互通)

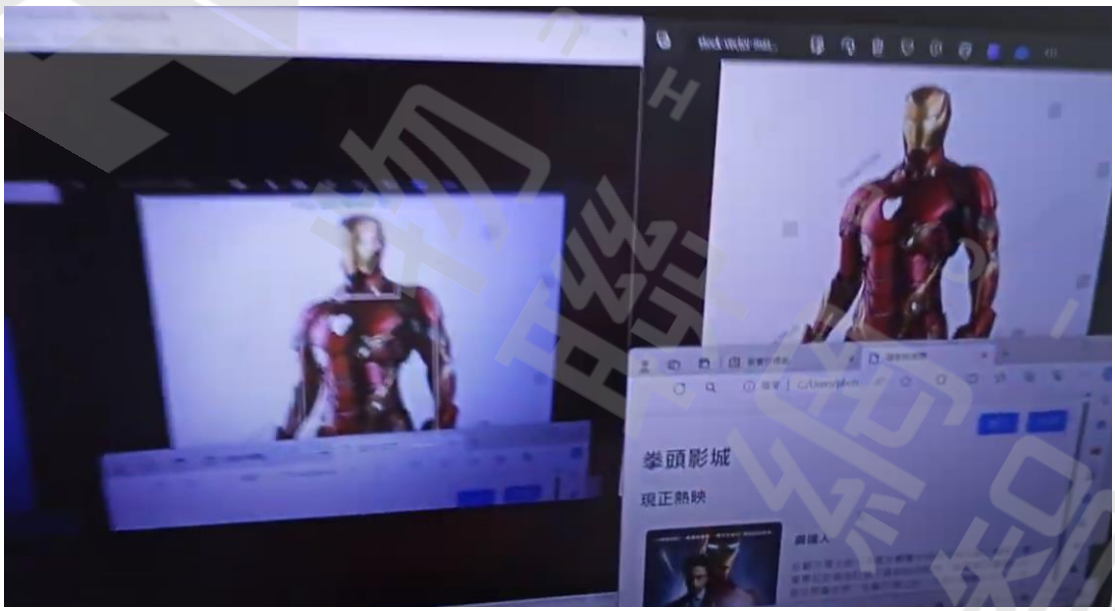


6. 使用 VLC 撥放器來查看 HUB 8735 Ultra 鏡頭之畫面以及辨識情形
使用 Arduino 上顯示的 RTSP IP 在 VLC 撥放器進行串流

```
14:32:10.696 ->
14:32:10.696 -> Total number of objects detected = 0
14:32:10.868 -> YOLOv4t tick[81]
14:32:10.868 -> Network URL for RTSP Streaming: rtsp://172.20.10.6:554
14:32:10.868 ->
14:32:10.868 -> Total number of objects detected = 0
14:32:11.052 -> YOLOv4t tick[74]
14:32:11.052 -> Network URL for RTSP Streaming: rtsp://172.20.10.6:554
14:32:11.052 ->
14:32:11.052 -> Total number of objects detected = 0
14:32:11.221 -> YOLOv4t tick[76]
14:32:11.221 -> Network URL for RTSP Streaming: rtsp://172.20.10.6:554
14:32:11.221 ->
14:32:11.221 -> Total number of objects detected = 0
14:32:11.405 -> YOLOv4t tick[75]
14:32:11.405 -> Network URL for RTSP Streaming: rtsp://172.20.10.6:554
14:32:11.437 ->
14:32:11.437 -> Total number of objects detected = 0
14:32:11.605 -> YOLOv4t tick[75]
14:32:11.605 -> Network URL for RTSP Streaming: rtsp://172.20.10.6:554
14:32:11.638 ->
14:32:11.638 -> Total number of objects detected = 0
14:32:11.807 -> YOLOv4t tick[75]
14:32:11.807 -> Network URL for RTSP Streaming: rtsp://172.20.10.6:554
```

圖中系統已辨識到角色美國隊長，在辨識到人物後，HUB 8735 Ultra 會發送 HTTP 請求至 Python 伺服器，伺服器接收請求後會執行開啟網頁之指令，來完成辨識人物後開啟介紹網頁的功能。(再次提醒，伺服器以及 Arduino 端必須同一網域，才能接收到 HTTP 請求。)

7. 確認跳轉出已設置好之網站



如以上兩圖，跳轉出的網頁皆為辨識之角色的介紹。