



國產IC開發套件 HUB 8735 手勢辨識居家智慧



案例規劃

➤ IC開發板：HUB-8735

➤ 案例/功能說明：

- 手勢辨識居家智慧系統，能提供特定族群解決生活上的不便
- 透過 HUB-8735 開發板連接到智能家居網路的開關或電器，再藉由鏡頭偵測手勢，即可控制開關燈或是電器用品

➤ 搭配模組：

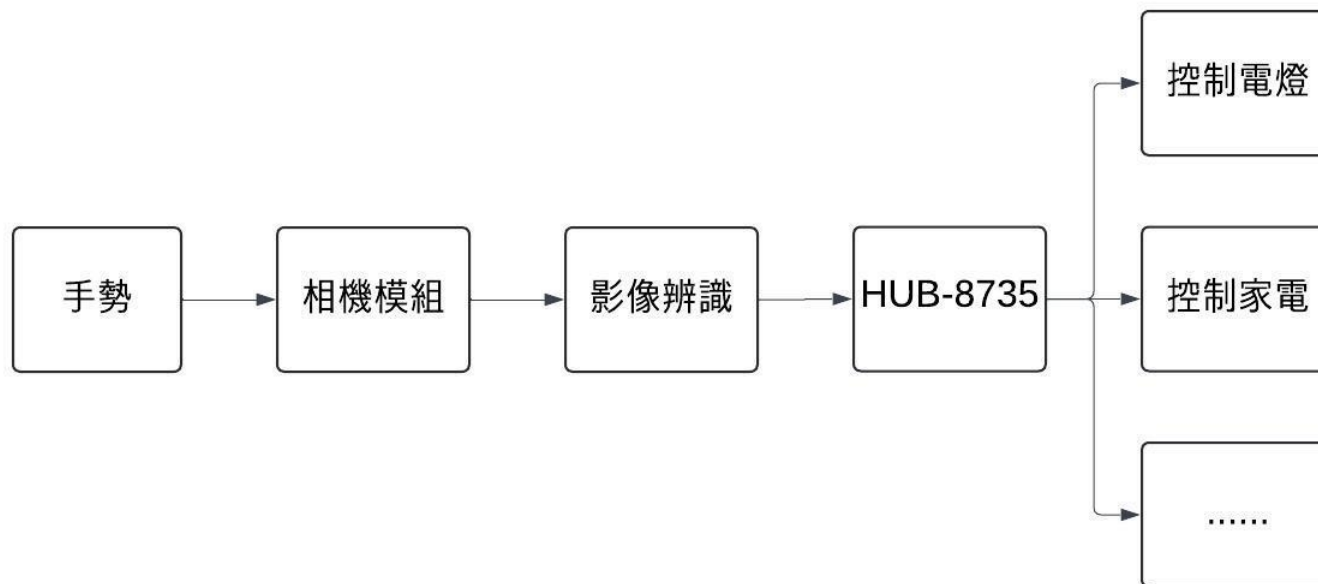
- RGB LED 燈模組

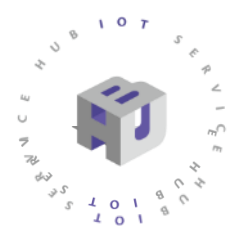
➤ 其他補充：

- 採用YOLOv4 進行AI訓練



案例架構圖





HUB 8735介紹



HUB 8735 特色

- 多功能影像處理
- 內建的NPU加速處理AI模型
- 支援Wi-Fi和藍芽傳輸
- 內置多款預訓練的AI模型
- 適用於多個領域
- 小尺寸設計可整合到產品設計中





HUB 8735 規格

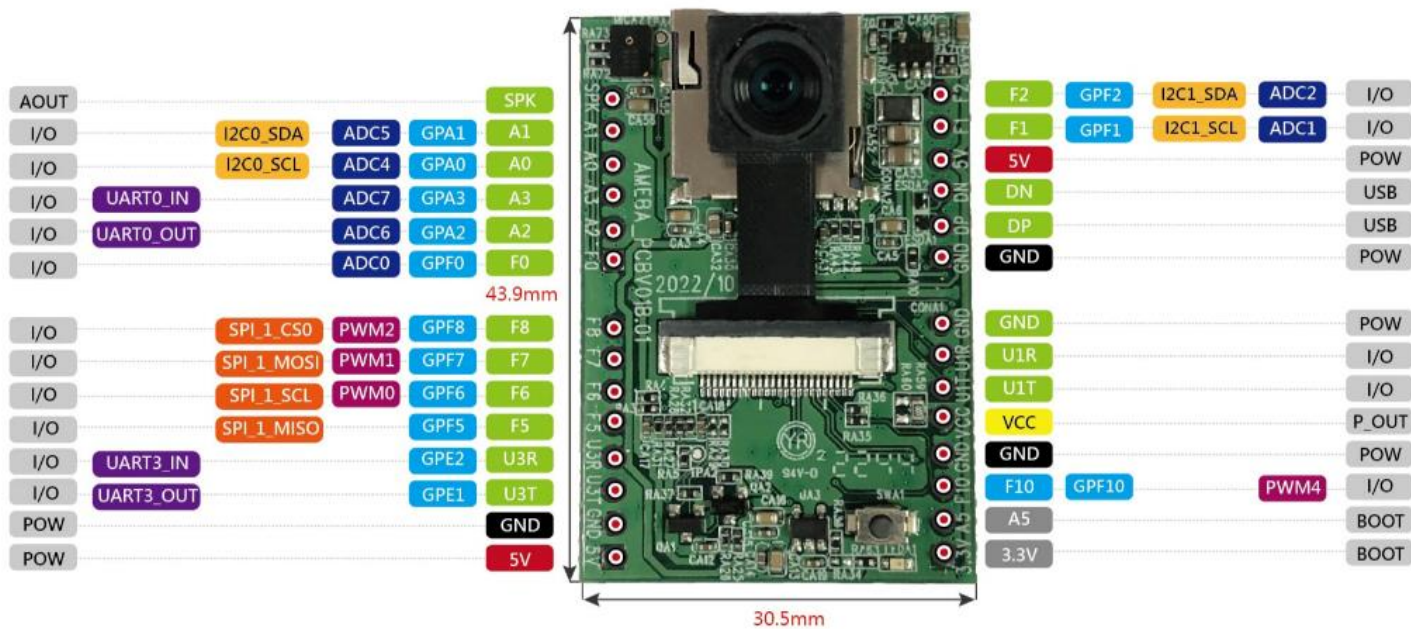
功能	描述
處理器	RTL8735B AIoT Chips
影像輸入	Full HD 1080P CMOS
語音輸入	內建數位麥克風
儲存裝置	支援SD記憶卡
無線支援	Wi-Fi 2.4 GHz / 5 GHz Bluetooth BLE 4.2 無線影像串流
影像壓縮	H.264/265
AI處理	提供多種pre-trainedAI models
I/O 界面	依照開發者需求擴充功能 1.Speaker 語音輸出 2.IMU Sensor 3.擴充溫度、溼度、震動 Sensor。



HUB 8735 接腳圖

DESIGN &
MADE IN TAIWAN

HUB 8735





前置作業

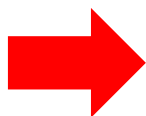
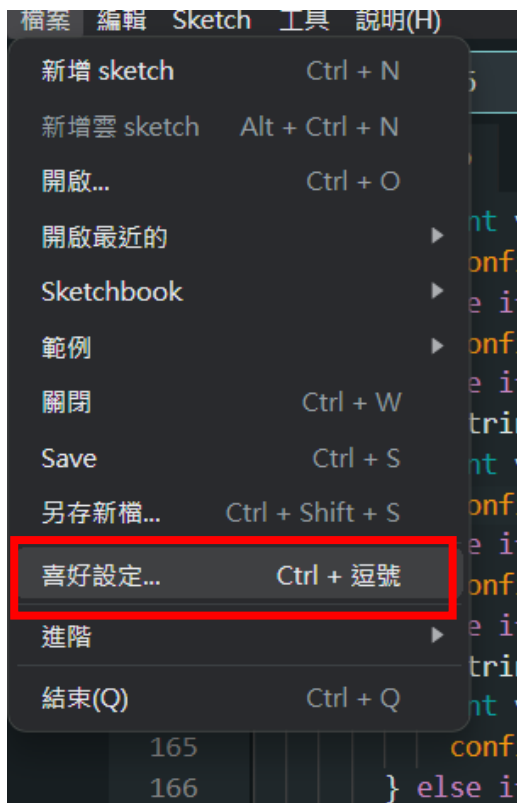


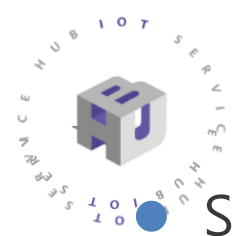
開發環境

- Step.1:至 Arduino 官網下載 Arduino IDE 1.8.19 之後的版本
<https://www.arduino.cc/en/software>
- Step.2:開啟Arduino IDE，選取檔案>偏好設定>在其他開發版版管理員網址貼上SDK地址

https://github.com/ideashatch/HUB-8735/raw/main/amebapro2_arduino/Arduino_package/ideasHatch.json

Step.2



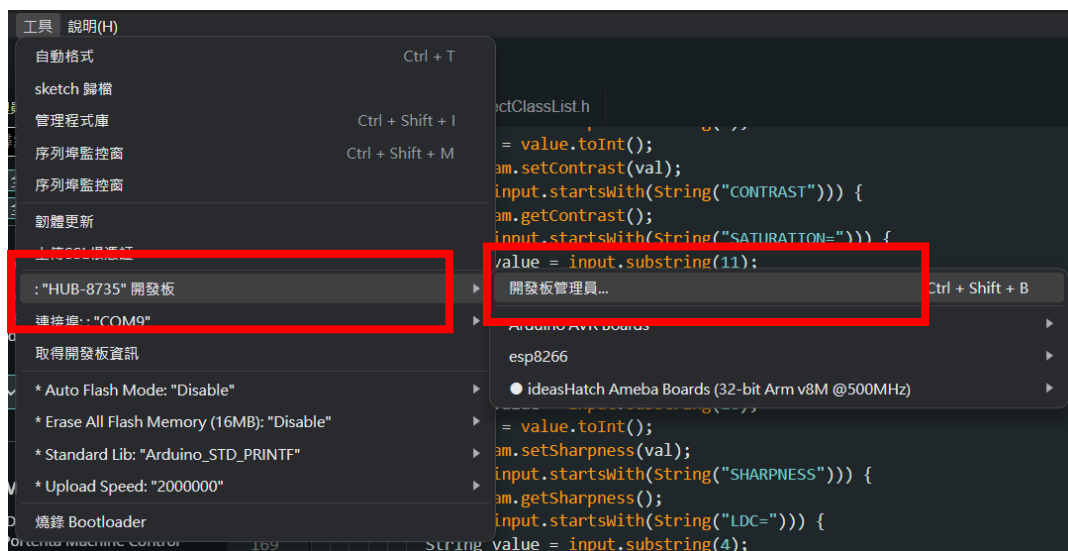


開發環境

- Step. 3:再從工具> HUB 8735的開發版> 開發板管理員中搜尋：HUB 8735，並安裝最新版的 HUB 8735 板子
- Step. 4:至以下網址下載 VLC Media Player

<https://github.com/portapps/vlc-portable/releases>

Step.3



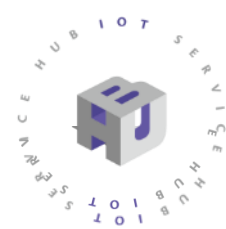


- ## USB to TTL



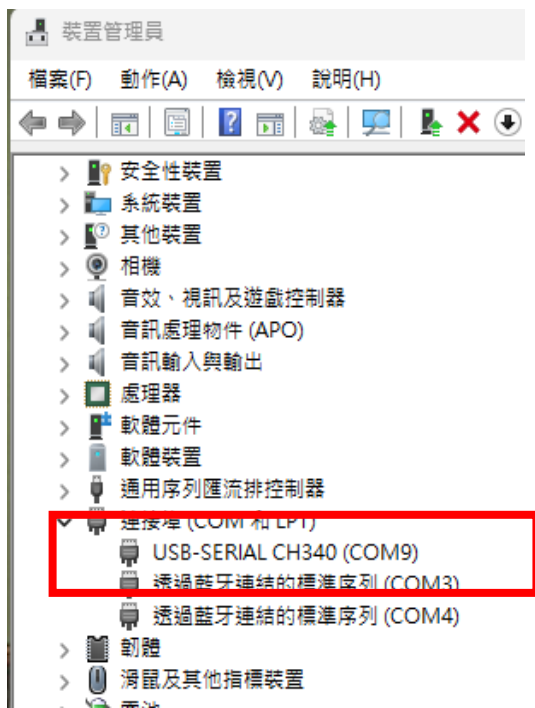


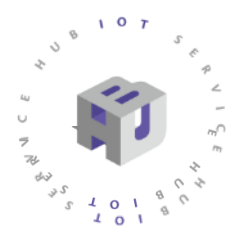
燒錄HUB 8735-以物件偵為測範例



燒錄 HUB 8735 - 以物件偵測為範例

- Step. 1: 開啟程式 範例 > AmebaNN > ObjectDetectionCallback
- Step. 2: 選擇開發板及序列埠





燒錄 HUB 8735 - 以物件偵測為範例

- Step. 3: 輸入 Wi-Fi SSID 及密碼
- Step. 4: 插入 BOOT_MODE 跳線並按下 Reset 鈕

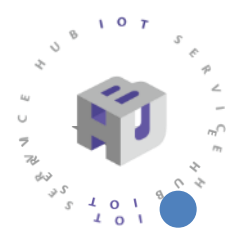
● Step. 5: 上傳程式

```
ObjectDetectionCallback | Arduino IDE 2.3.2
檔案 編輯 Sketch 工具 說明(H)
[Check] [Next] [Upload] [USB] HUB-8735
ObjectDetectionCallback.ino ObjectClassList.h
37 // Lower resolution for NN processing
38 #define NNWIDTH 576
39 #define NNHEIGHT 320
40
41 CameraSetting configCam;
42 VideoSetting config(VIDEO_FHD, 30, VIDEO_H264, 0);
43 VideoSetting configNN(NNWIDTH, NNHEIGHT, 10, VIDEO_RGB, 0);
44 NNObjectDetection ObjDet;
45 RTSP rtsp;
46 StreamIO videoStreamer(1, 1);
47 StreamIO videoStreamerNN(1, 1);
48
49 char ssid[] = " "; // your network SSID (name)
50 char pass[] = " "; // your network password
51 int status = WL_IDLE_STATUS;
52 uint8_t oldNum;
53
54 IPAddress ip;
```

Step.3

Step.5

```
ObjectDetectionCallback | Arduino IDE 2.3.2
檔案 編輯 Sketch 工具 說明(H)
[Check] [Next] [Upload] [USB] HUB-8735
ObjectDetectionCallback.ino ObjectClassList.h
115 OSD.configVideo(CHANNEL, config);
116 OSD.begin();
117
118 //configCam.setGrayMode(1);
119
120 pinMode(LED_BUILTIN, OUTPUT);
121 pinMode(9, OUTPUT);
122 }
123
124 void loop()
125 {
126
127 }
128
129 // User callback function for post pr
130 void ODPPostProcess(std::vector<Object
131 {
132     uint16_t im_h = config.height();
133     uint16_t im_w = config.width();
```



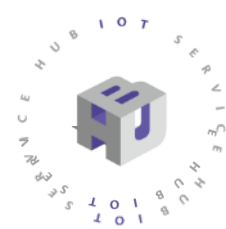
燒錄HUB 8735-以物件偵測為範例

- Step. 6: 燒入完成之後拔掉BOOT_MODE跳線
- Step. 7: 開啟序列埠監控視窗
- Step. 8: 設定鮑率為115200
- Step. 9: 按下Reset鈕
- Step. 10: 查看IP位址

The screenshot displays the Arduino IDE 2.3.2 interface. The top toolbar shows the upload button (a right-pointing arrow) highlighted with a red box and labeled "Step. 6". Below the toolbar, the "ObjectDetectionCallback.ino" file is open, showing code for camera settings and object detection. The "Serial Monitor" window is open, showing the upload progress: "Sketch uses 14995456 bytes (89%) of program storage space. Maximum is 16777216 bytes." and "Uploading.....upload success". The "Serial Monitor" window is also highlighted with a red box and labeled "Step. 6 燒入成功".

The "Serial Monitor" window is set to "COM9" and "115200 鮑率" (Baud Rate), which is highlighted with a red box and labeled "Step. 8". The "Serial Monitor" window is also highlighted with a red box and labeled "Step. 7".

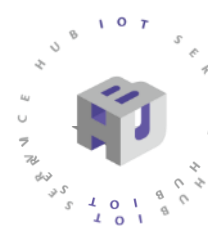
The "Serial Monitor" window shows the output of the program, including the IP address "192.168.227.179" and the RTSP URL "rtsp://192.168.227.179:554", which is highlighted with a red box and labeled "Step. 10". The "Serial Monitor" window is also highlighted with a red box and labeled "序列埠監控視窗".



燒錄HUB 8735-以物件偵測為範例

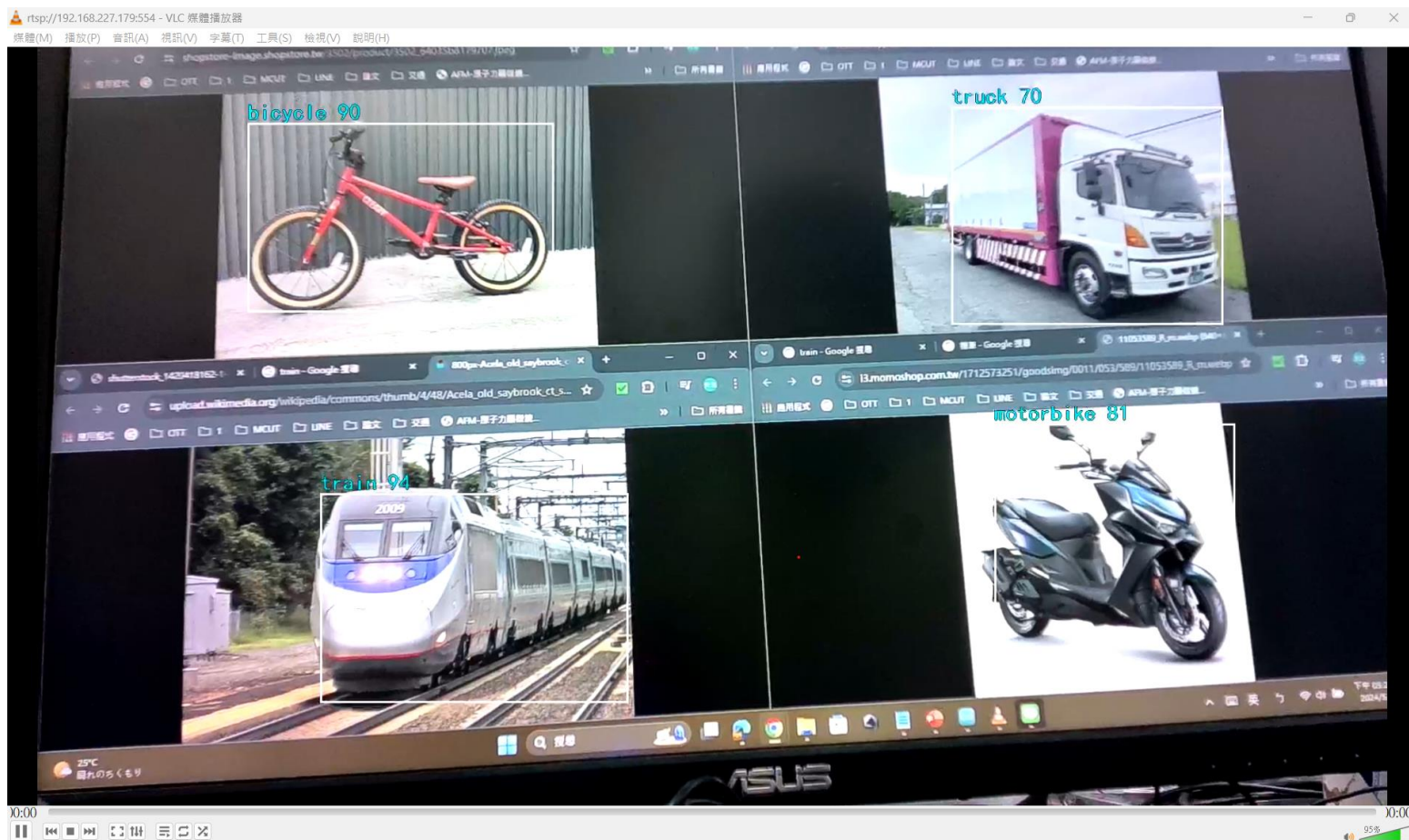
- Step. 11: 打開VLC Media Player > 媒體 > 串流
- Step. 12: 選取網路 > 並於網路通訊協定貼上IP 位址

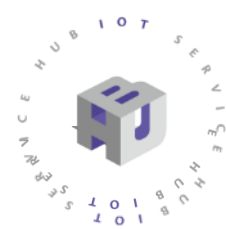




燒錄HUB 8735-以物件偵測為範例

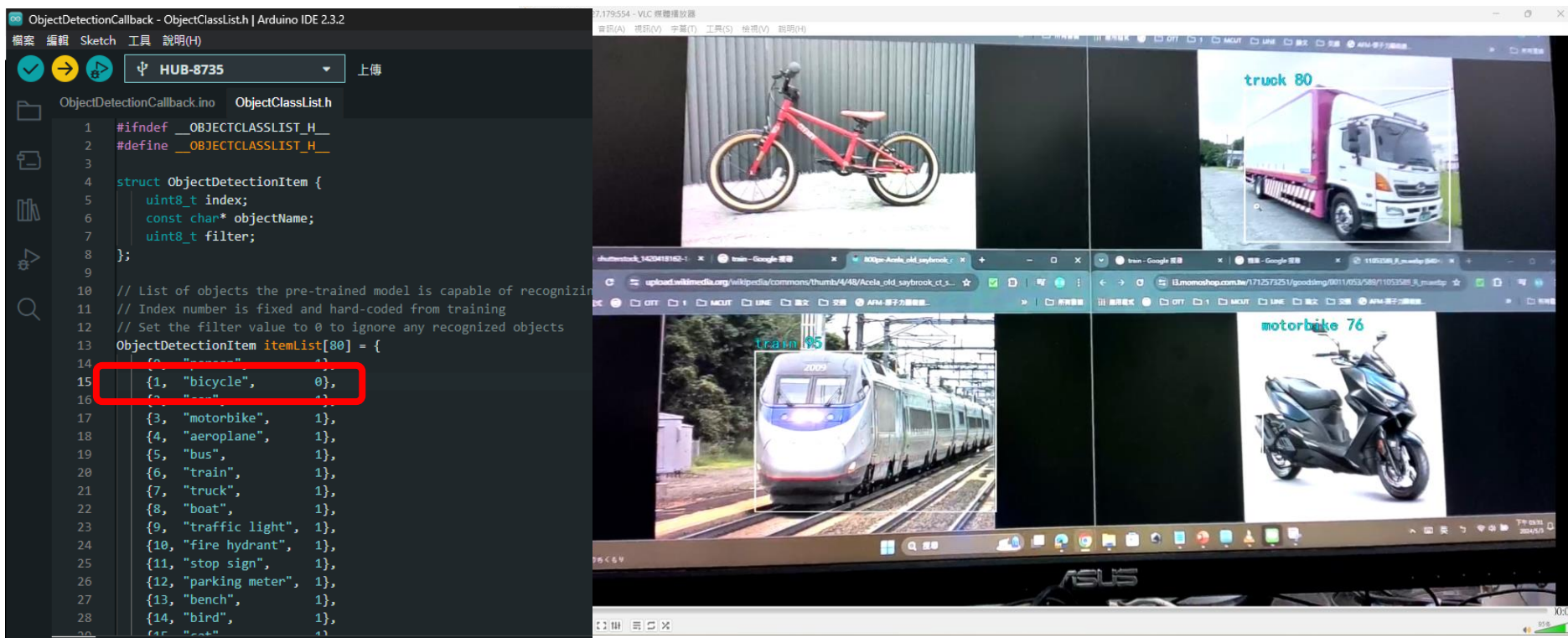
- 播放後就會開始撥放串流，您可以在播放器看到偵測到的物件決策框





燒錄 HUB 8735 - 以物件偵測為範例

- 若要關閉特定物件，可到 _OBJECTCLASSLIST_H_ 把該物件的1改為0





訓練自己的模型

我們的程式碼都在：<https://github.com/lalalin412/hub8735.git>

照片訓練及資料：https://drive.google.com/drive/folders/1MMyN57JvJl1ECvJT1zq97zCGAZI7YD5A?usp=drive_link



AMB82 AI 模型轉換

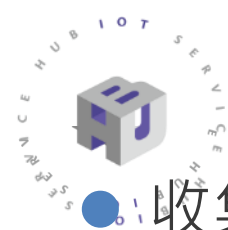
● HUB-8735支援以下模型轉換

注意

Please Log in to access the page.

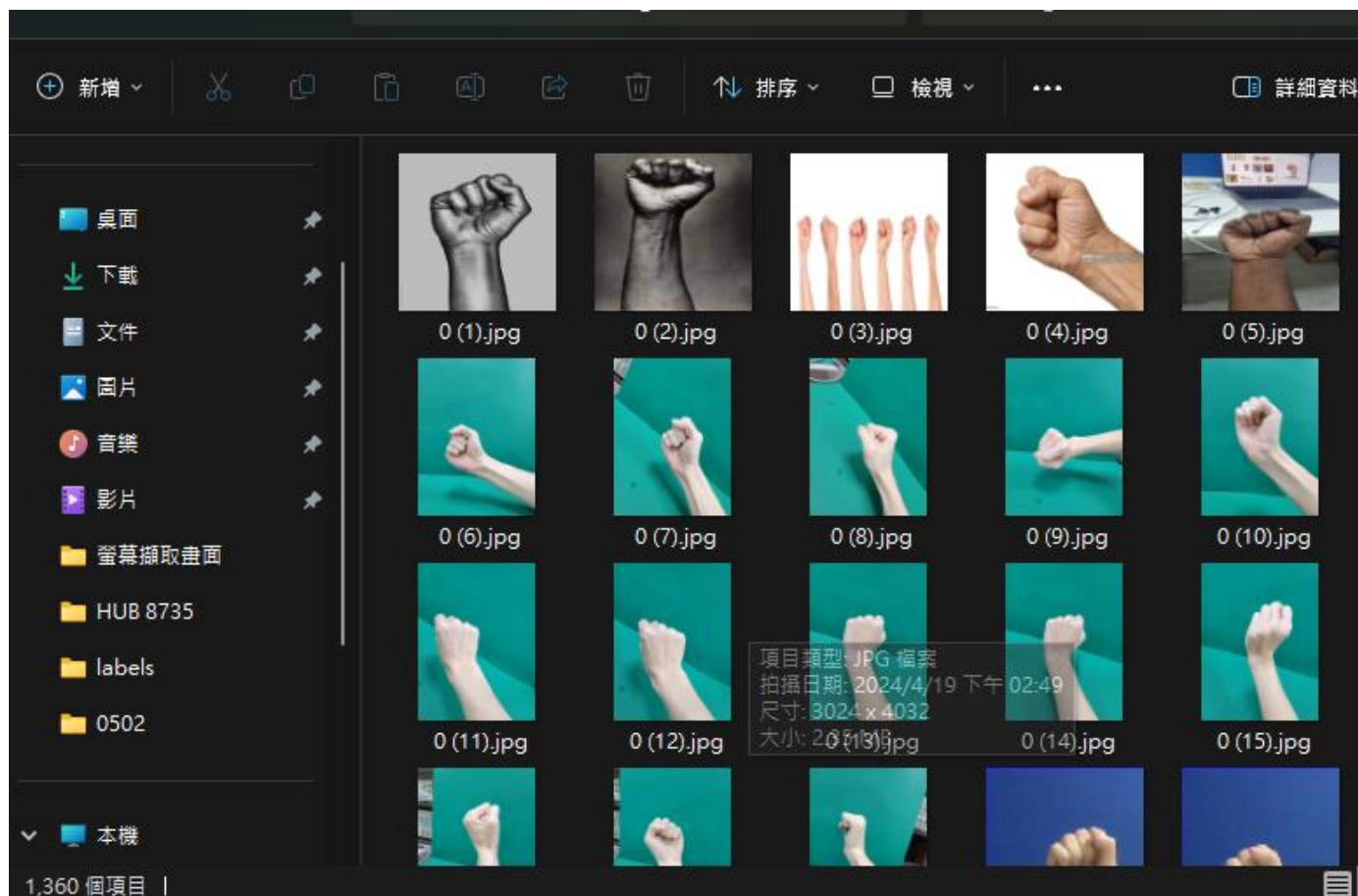
目前支援線上轉換的型號請參考下表。

Models	Basic functions	Requires files
yolov3-tiny, darknet	Object Detection	".cfg", ".weights"
yolov4-tiny, darknet	Object Detection	".cfg", ".weights"
yolov7-tiny, darknet	Object Detection	".cfg", ".weights"
yolov7-tiny, pytorch	Object Detection	".pt"
scrfd/mobilefacenet	Face Detection & Recognition	".pt" or ".onnx"
yamnet	Sound Classification	".h5"
CNN Gray/RGB	Image Classification	".h5"



AI 模型訓練

● 收集資料集-本範例的資料集共有 6 個手勢(數字0~5)，全部有 1360 張圖片



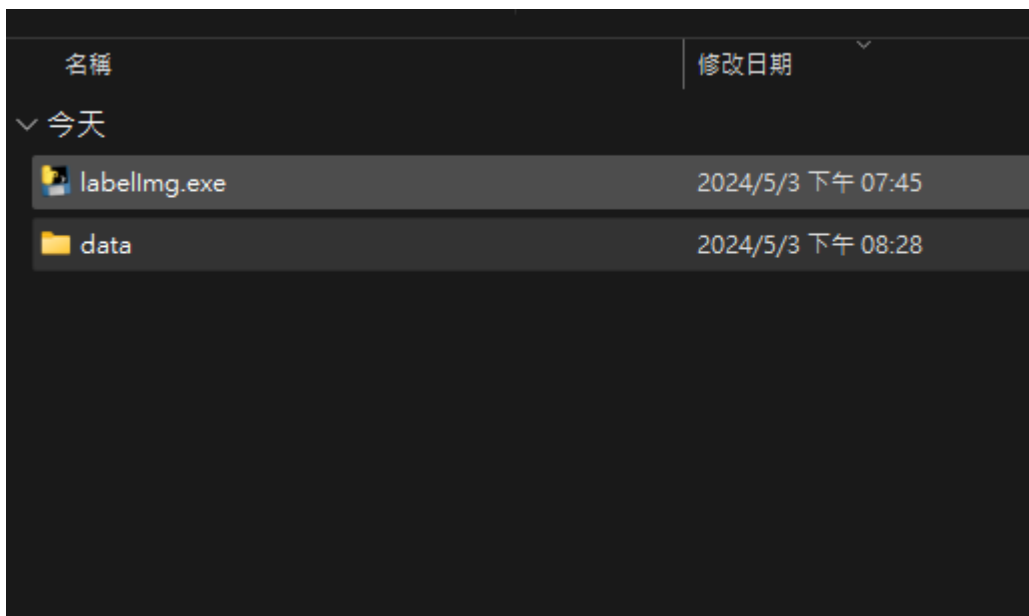


AI 模型訓練

- 於以下網址下載labellmg

<https://github.com/HumanSignal/labellmg/releases>

- 解壓縮後進入data資料編輯類別



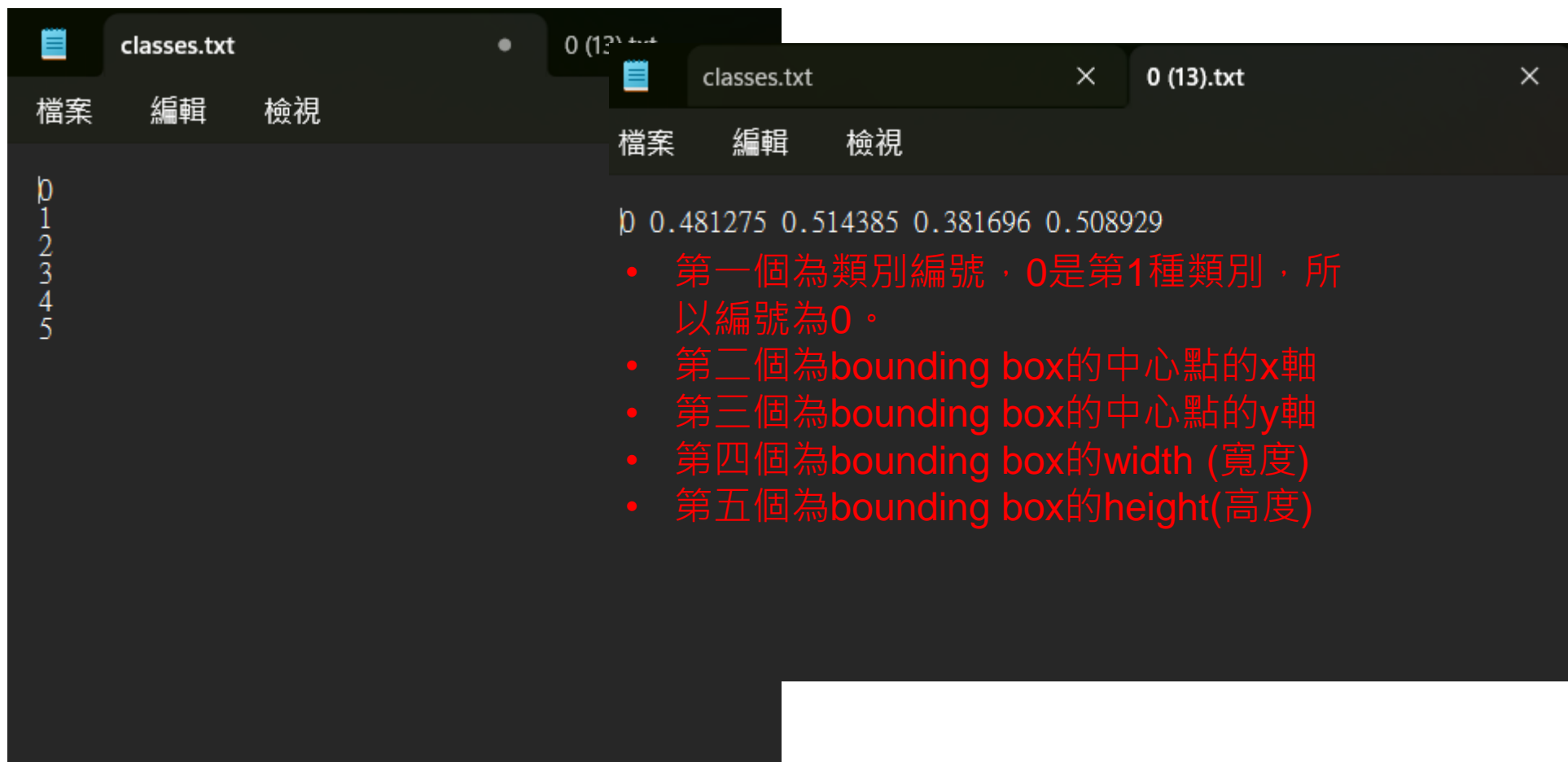


-
- The screenshot shows the 'labellingm' application window. The main area displays an image of a hand with a green bounding box. The left sidebar contains several buttons: 'Open', 'Open Dir', 'Change Save Dir', 'Next Image', 'Prev Image', 'Verify Image', 'Save', 'yolo', 'YOLO', 'Create RectBox', and 'Duplicate RectBox'. The right sidebar has a 'Box Labels' section with 'Edit Label', 'difficult', and 'Use default label' options, and a 'File List' section showing a directory tree. Red annotations are present: 'Step2. 選取圖片資料夾' points to 'Open Dir'; 'Step3. 選取txt存放目錄' points to 'Change Save Dir'; 'Step4. 需顯示成YOLO' points to 'YOLO'; 'Step5. 畫框' points to 'Create RectBox' with a red arrow; and 'Step6. 儲存資料' points to 'Save'.



AI 模型訓練

- 回到txt的存放目錄，就可以發現classes.txt 及剛剛儲存的txt檔，會與圖片檔同名，txt裡存著bounding box的資訊。



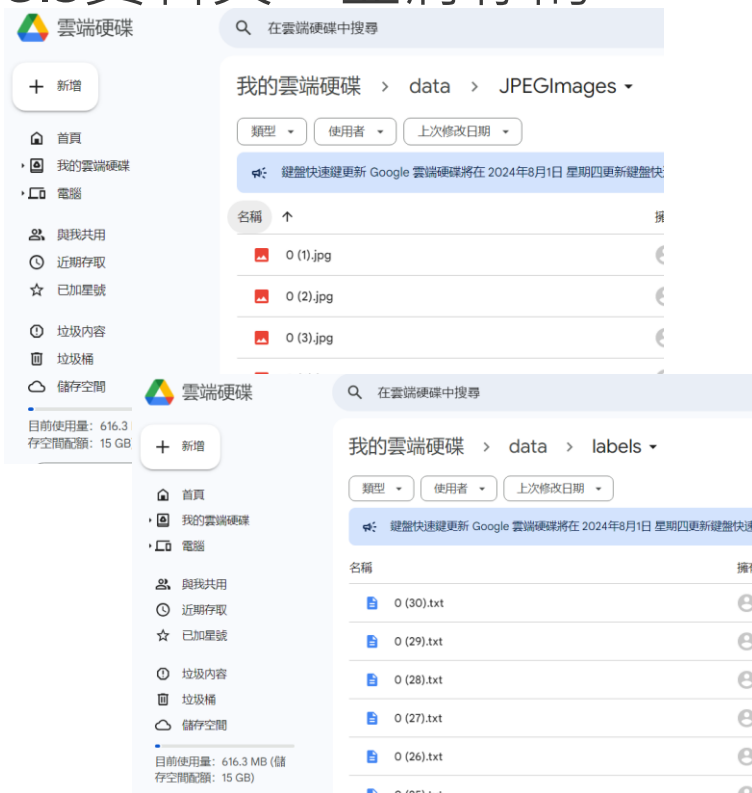


AI 模型訓練

- 至google drive命名一個名為data資料夾
- 在data資料夾內命名一個為JPEGImages資料夾，並將你的圖片存入資料夾內
- 在data資料夾內命名一個為labels資料夾，並將你的txt檔存入資料夾內



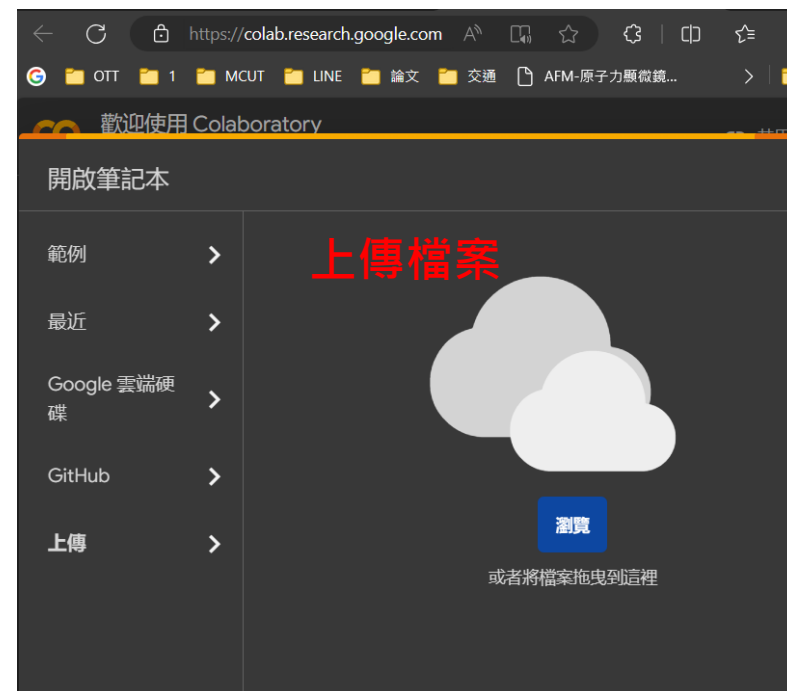
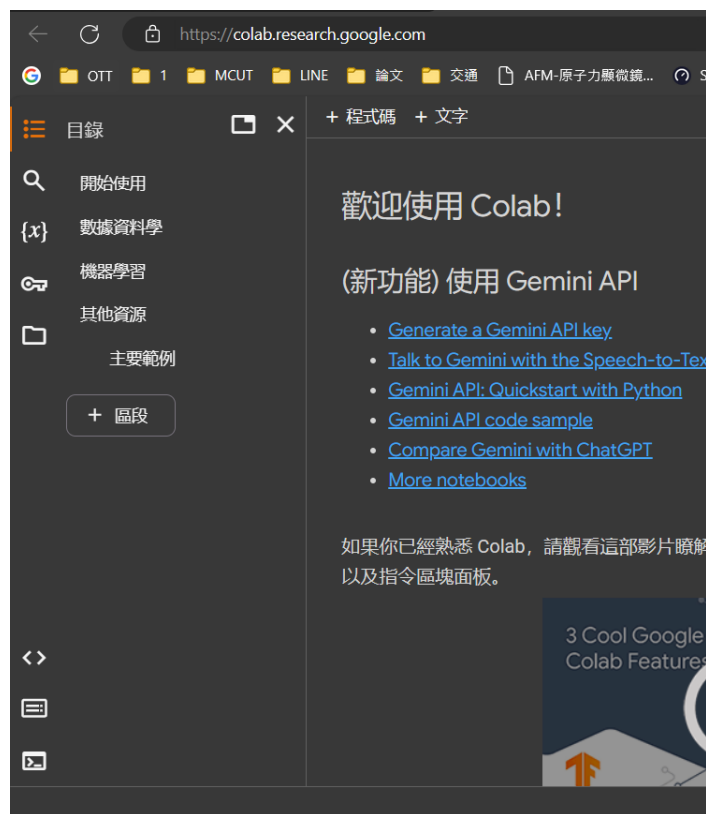
***注意**
一定要按照資料集的格式存放，因為
darknet 原始碼中的介面定義就是需要
這樣放置和命名。

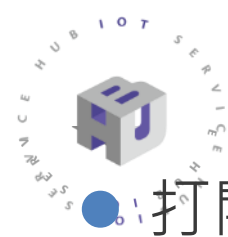




AI 模型訓練

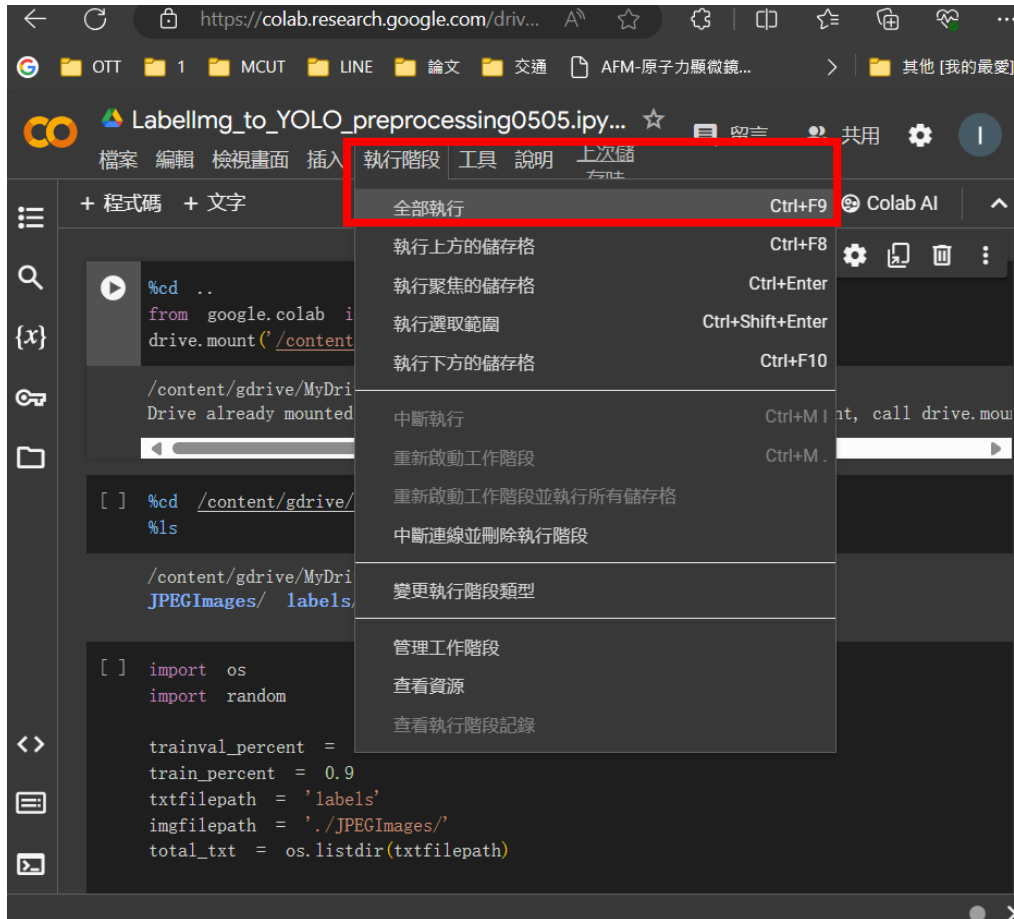
- 至Google Colab製作AI模型
- <https://colab.research.google.com/?hl=zh-tw>





AI 模型訓練

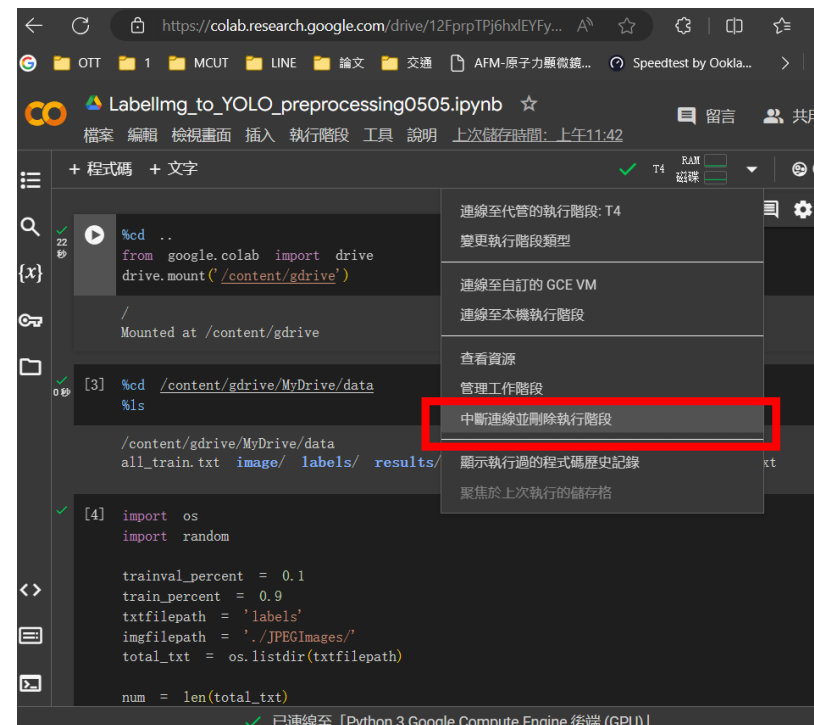
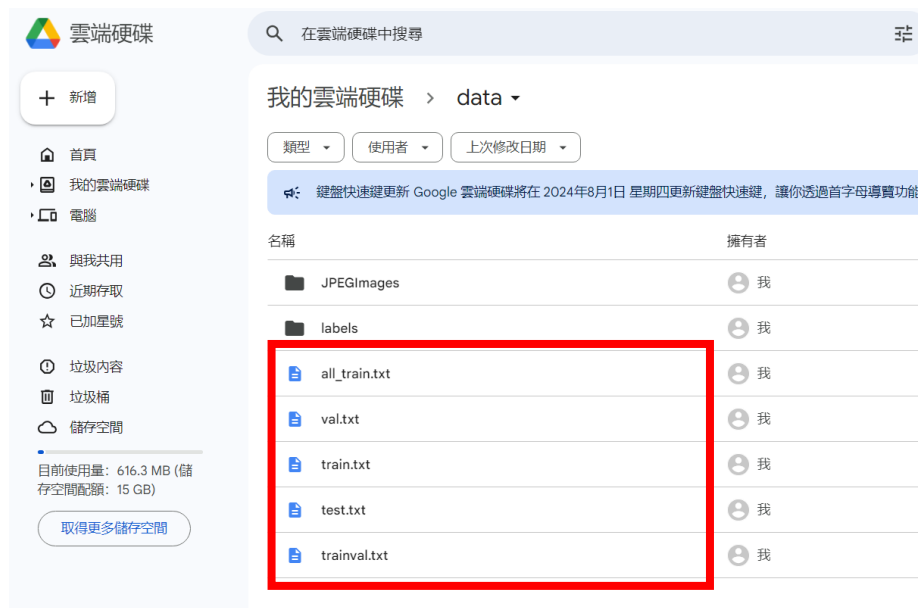
打開Labellmg_to_YOLO_preprocessing，按下執行階段>全部執行





AI 模型訓練

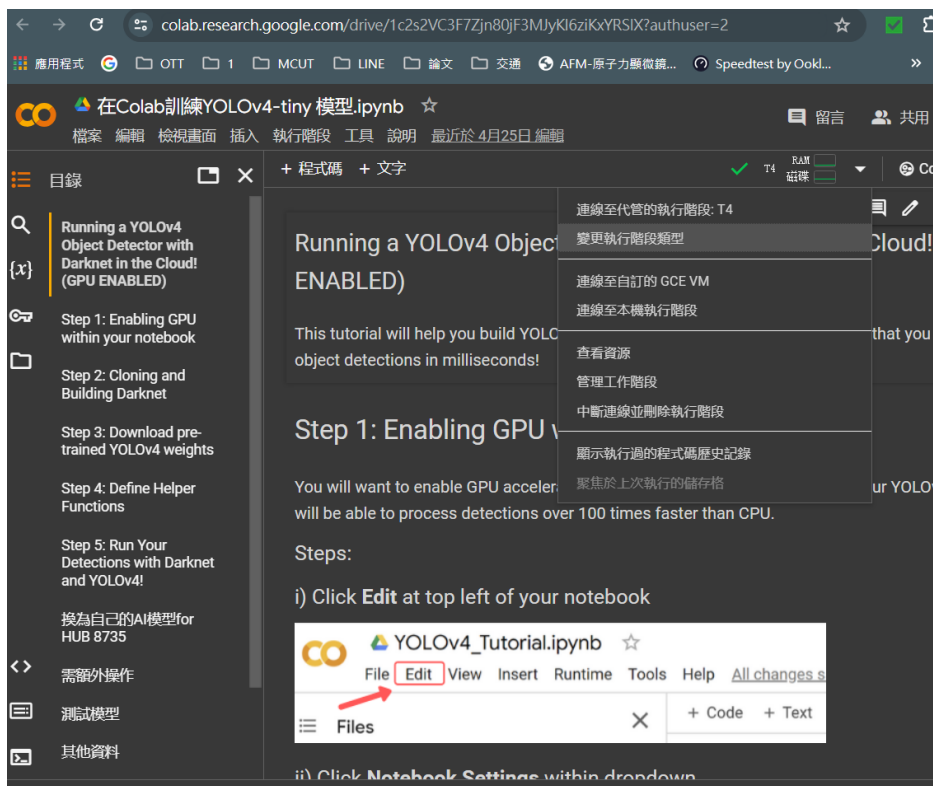
- 執行完成後就可以在你google drive的data 目錄下看到以下檔案
- 中斷執行階段後就可以執行下一步驟





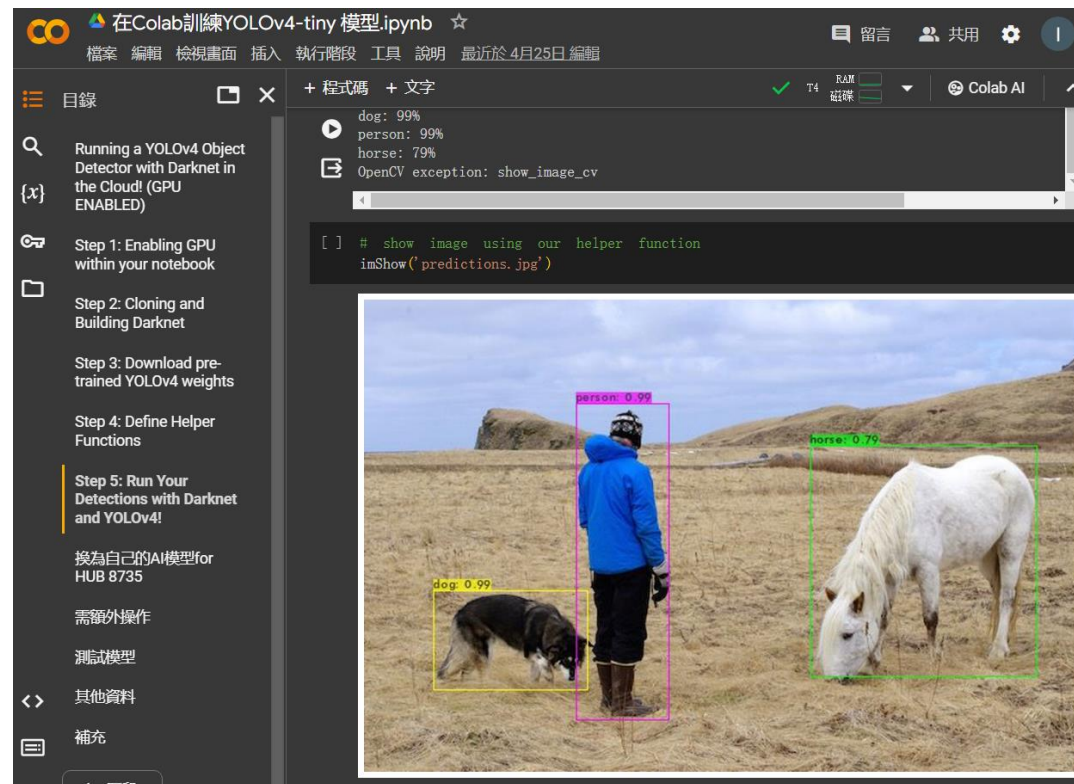
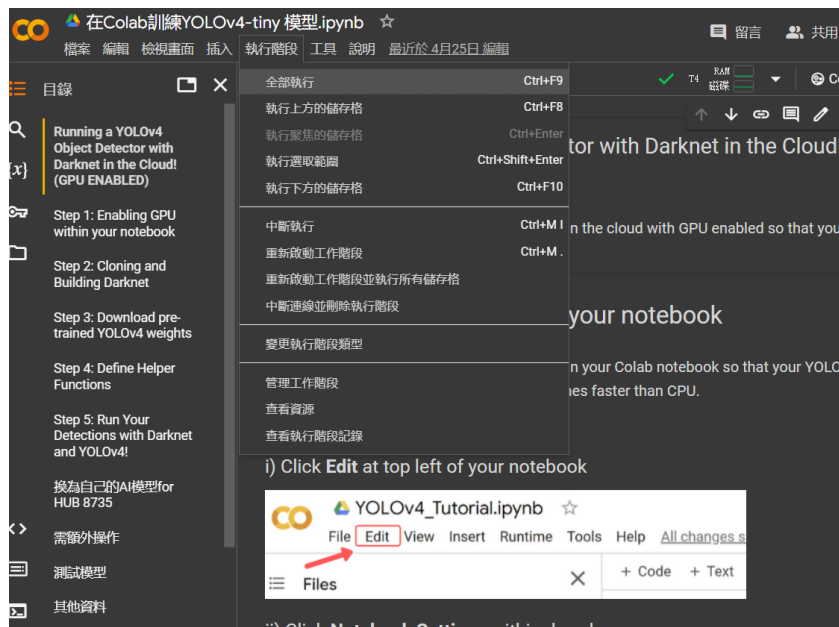
AI 模型訓練

● 打開Train_YOLO_Object_Detection_With_Colab





AI 模型訓練



執行完成後就可以看到這張圖片



AI 模型訓練

- 現在，將可以開始訓練自己的AI模型了
- 首先執行下列3行程式

在Colab訓練YOLOv4-tiny 模型.ipynb

檔案 編輯 檢視畫面 插入 執行階段 工具 說明

檔案

{x}

darknet

3rdparty

backup

build

cfg

cmake

data

include

obj

results

scripts

src

CMakeLists...

程式碼 + 文字

AssertionError

Next steps: [Explain error](#)

換為自己的AI模型for HUB 8735

my_yolov4-tiny

```
from google.colab import drive
drive.mount('/content/gdrive')

!cp -r /content/gdrive/MyDrive/data/* /content/darknet

%mkdir /content/gdrive/MyDrive/data/results
```

執行儲存格 (Ctrl+Enter)
上次變更後執行過的儲存格
在 下午1:51 (0 分鐘前) 時排入佇列

於此手動上傳，編輯過之: my_ai.data, my_ai.names, my_yolov4-tiny.cfg

AI 模型訓練

修改my_yolov4-tiny.cfg檔案

```
my_yolov4-tiny.cfg*
10 channels=3
11 momentum=0.9
12 decay=0.0005
13 angle=0
14 saturation = 1.5
15 exposure = 1.5
16 hue=.1
17
18 learning_rate=0.00261
19 burn_in=1000
20
21 max_batches = 12000
22 policy=steps
23 steps=9600,10800
24 scales=.1,.1
25
26 #weights_reject_freq=1001
27 #ema_alpha=0.9998
28 #equidistant_point=1000
29 #num_sigmas_reject_badlabels=3
30 #badlabels_rejection_percentage=0.2
31
288 truth_thresh = 1
289 random=0
290 resize=1.5
```

- **max_batches = 種類數 * 2000, 但不少於 4000**
- **本專案有 6 個類別 所以是 12000**
- **steps =**
- **max_batches * 0.8, max_batches * 0.9**
- **e.g., max_batches = 4000 steps = 3200, 3600**

```
my_yolov4-tiny.cfg*
213 stride=1
214 pad=1
215 activation=leaky
216
217 [convolutional]
218 size=1
219 stride=1
220 pad=1
221 filters=33
222 activation=linear
223
224
225
226 [yolo]
227 mask = 3,4,5
228 anchors = 10,14, 23,27, 37,58, 81,82, 135,169, 344,319
229 classes=6
230 num=6
231 jitter=.3
232 scale_x_y = 1.05
233 cls_normalizer=1.0
234
```

- **修改 Filters = 3*(5+len(classes))**
- **修改[yolo]區塊的 classes**
本例子有 6 個 classes
所以 Filters=3*(5+6)=33
- **總共有兩個[yolo]區塊需要修**



AI 模型訓練

```
.names my_ai.data +
檔案 編輯 檢視
classes = 6
train = ./all train.txt
valid = ./test.txt
names = ./my_ai.names
backup = /content/gdrive/MyDrive/data/results
results = /content/gdrive/MyDrive/data/results
eval = coco
```

- 修改 **my_ai.data** 的 **classes** 數量

第 2 行, 第 1; 12, 192 個字元數 100% Unix (LF) UTF-8

```
my_ai. +
檔案 編輯 檢視
0
1
2
3
4
5
6
```

- 修改 **my_ai.name** 的變成你的類別名稱



AI 模型訓練

- 將3個檔案上傳自darknet

在Colab訓練YOLOv4-tiny 模型.ipynb

檔案 編輯 檢視畫面 插入 執行階段 工具 說明 已儲存所有變更

檔案

darknet

my_ai.data

my_ai.names

my_yolov4-tiny.cfg

名稱 修改日期 類型

名稱	修改日期	類型
my_ai.data	2024/5/5 下午 09:44	DATA
my_ai.names	2024/5/5 下午 09:23	NAME
my_yolov4-tiny.cfg	2024/5/5 下午 09:28	CONFIG

3 個項目 | 已選取 3 個項目 3.33 KB |

測試模型

0 秒 完成時間: 下午1:54



AI 模型訓練

- 這樣就可以開始訓練你的模型了

```
[15] %cd /content/darknet
      %ls

/content/darknet
3rdparty/      darknet_video.py      Makefile        src/
all_train.txt  data/                 my_ai.data      test.txt
backup/        docker-compose.yml    my_ai.names     train.txt
build/         Dockerfile.cpu        my_yolov4-tiny.cfg trainval.txt
build.ps1*     Dockerfile.gpu        net_cam_v3.sh*  val.txt
cfg/           image/               net_cam_v4.sh*  vcpkg.json
cmake/         image_yolov3.sh*      obj/            vcpkg.json.opencv23
CMakeLists.txt image_yolov4.sh*      package.xml     video_yolov3.sh*
darknet*       include/             predictions.jpg video_yolov4.sh*
DarknetConfig.cmake.in json_mjpeg_streams.sh* README.md       yolov4-tiny.conv.29
darknet_images.py labels/               results/        yolov4-tiny.weights
darknet.py     LICENSE              scripts/

!./darknet detector train my_ai.data my_yolov4-tiny.cfg yolov4-tiny.conv.29 -dont_show

...  CUDA-version: 12020 (12020), cuDNN: 8.9.6, CUDNN_HALF=1, GPU count: 1
      CUDNN_HALF=1
      OpenCV version: 4.5.4
my_yolov4-tiny
0 : compute_capability = 750, cudnn_half = 1, GPU: Tesla T4
net.optimized_memory = 0
mini_batch = 64, batch = 64, time_steps = 1, train = 1
  layer  filters  size/strd(dil)  input              output
0 Create CUDA-stream - 0
Create cudnn-handle 0
conv    32       3 x 3/ 2       416 x 416 x 3 -> 208 x 208 x 32 0.075 BF
1 conv  64       3 x 3/ 2       208 x 208 x 32 -> 104 x 104 x 64 0.399 BF
2 conv  64       3 x 3/ 1       104 x 104 x 64 -> 104 x 104 x 64 0.797 BF
3 route 2                          1/2 -> 104 x 104 x 32
```



AI模型訓練 補充

- 若想使用Yolov7模型可下載以下檔案
- Yolov7-tiny
 - ◆ cfg: <https://raw.githubusercontent.com/AlexeyAB/darknet/master/cfg/yolov7-tiny.cfg>
 - ◆ weights for fine-tuning:
<https://github.com/AlexeyAB/darknet/releases/download/yolov4/yolov7-tiny.conv.87>
- 修改my_yolov7-tiny.cfg後將2個檔案上傳自darknet 目錄下
- 修改

```
!./darknet detector train my_ai.data my_yolov4-tiny.cfg yolov4-tiny.conv.29 -dont_show
```

為

```
!./darknet detector train my_ai.data yolov7-tiny.cfg yolov7-tiny.conv.87 -dont_show
```



AI 模型訓練

- 訓練完之後就可以在data/result看到你的權重檔

我的雲端硬碟 > data > results ▾



類型 ▾

使用者 ▾

上次修改日期 ▾

名稱 ↑

擁有者

上次修改時間 ▾

檔案大小



my_yolov4-tiny_last.weights

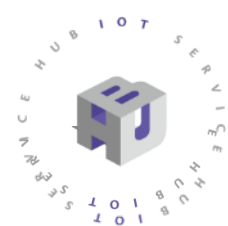


我

下午5:35

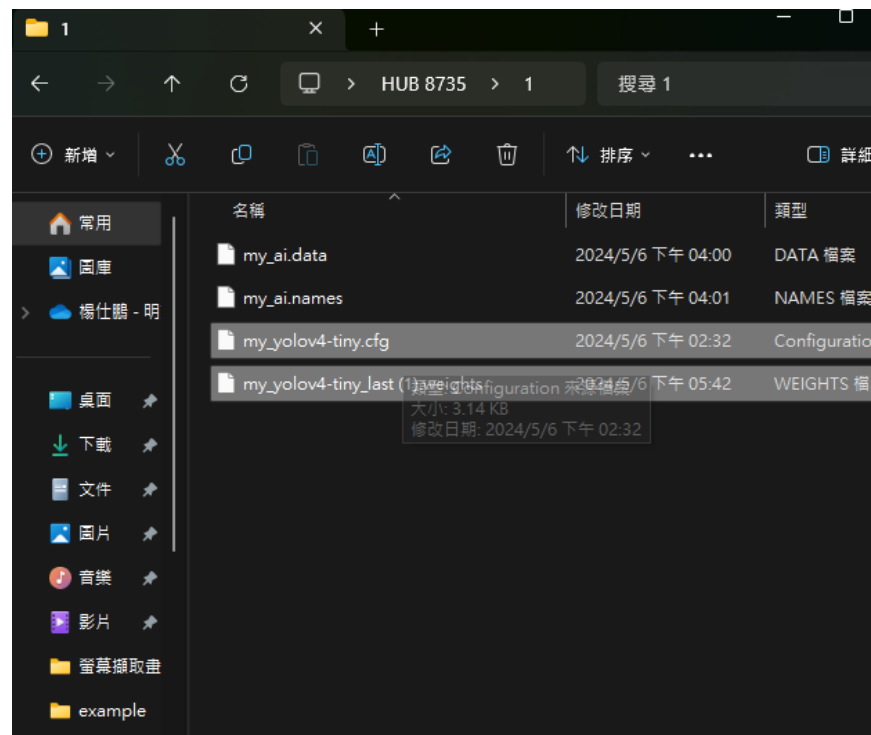
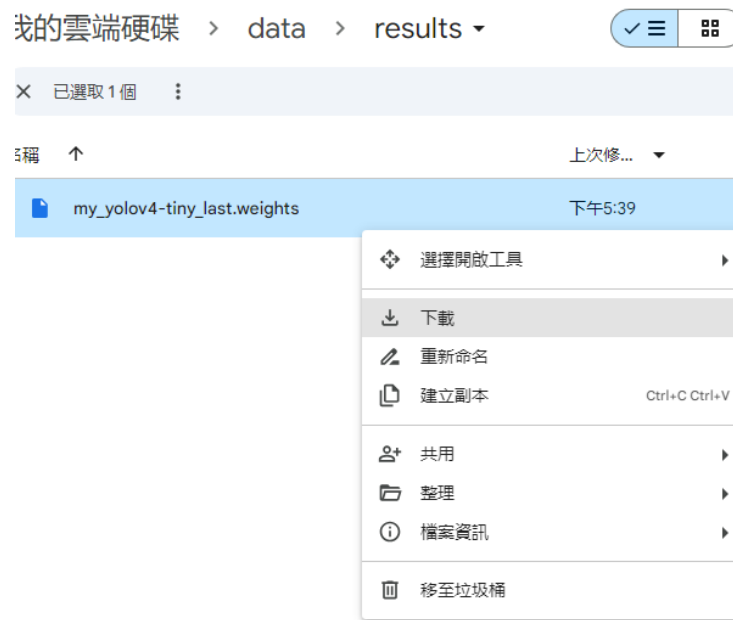
22.5 MB

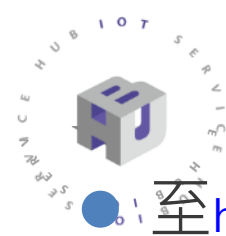




AI 模型轉換

- 把你的檔案下載下來並和my_yolov4-tiny.cfg壓縮成zip





AI 模型轉換

至 <https://www.amebaiot.com/zh/amebapro2-ai-convert-model/> 進行線上AI 模型轉換

CNN-RGB or CNN-GRAY)

YOLO-TINY

選擇YOLO-TINY



Quantize Type (required, UINT8 or INT16)

INT16

選擇一種量化類型，UINT8或INT16都可以



Upload zip file including a cfg file and a weights file(required, please upload the folder or compressed file contained the ".cfg" and ".weights" files, all named in English, limit:35MB)

選擇檔案

my_yolov4-tiny.zip

選擇zip檔上傳至此，檔案大小限制35MB

Upload one jpg file (required, limit:1MB)

選擇檔案

0 (21).jpg

選擇任何一張圖片來驗證

Upload jpg files (option, limit:1MB)

選擇檔案

沒有選擇檔案

選擇檔案

沒有選擇檔案



以Arduino IDE將模型燒入晶片

- 1.等待自動回覆至登記註冊e-mail 信箱裡，並將nb檔案下載。
- 2.開啟資料夾至:

C:\Users\<使用者名稱>\AppData\Local\Arduino15\packages\ideasHatch\hardware\AmebaPro2\4.0.10-Release\variants\common_nn_models

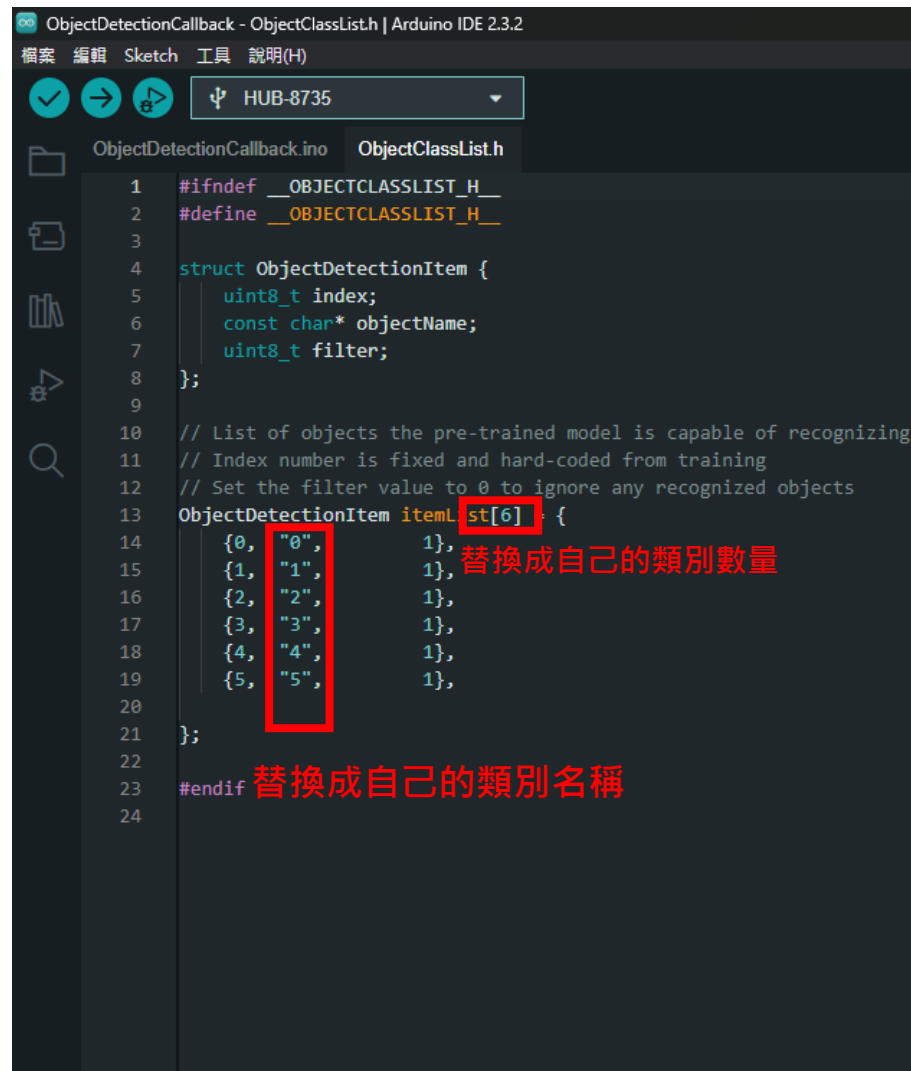
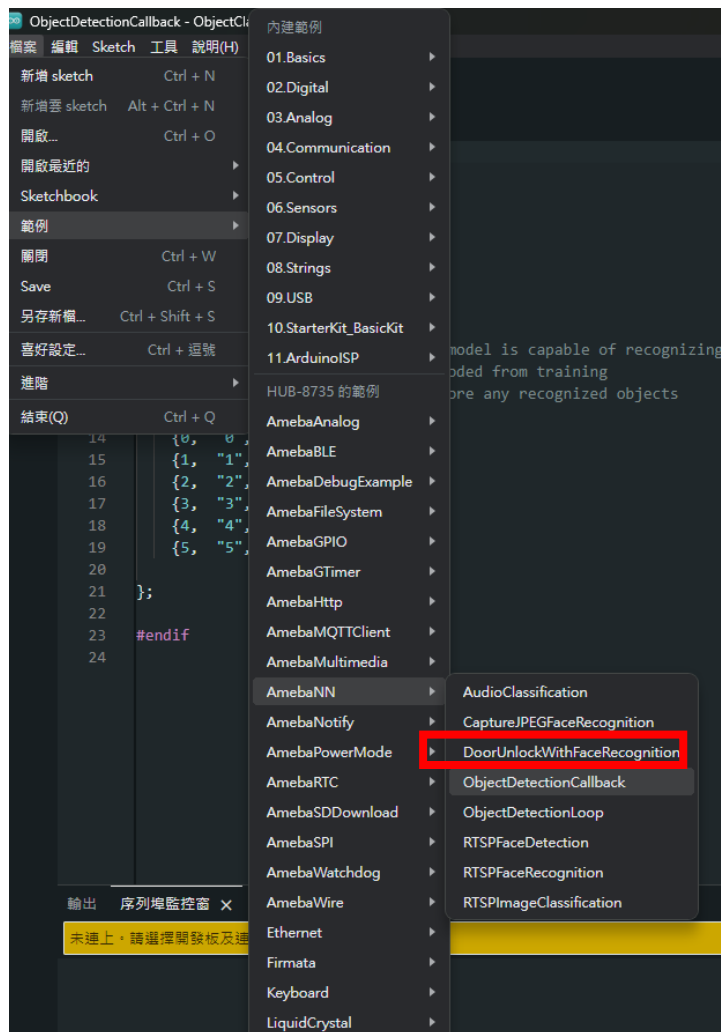
- 替換yolov4_tiny.nb 檔案。





以Arduino IDE將模型燒入晶片

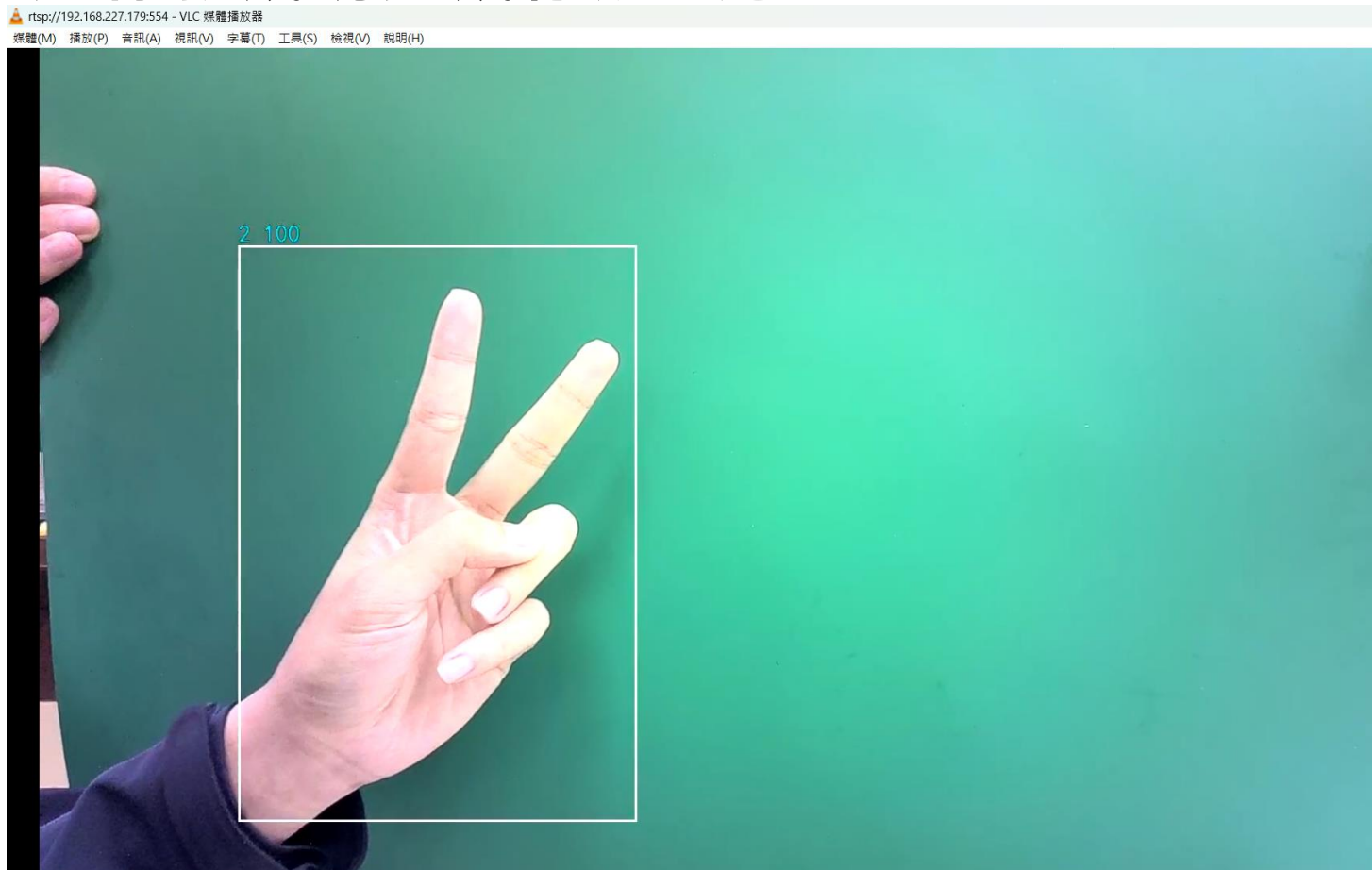
- 開啟程式 範例>AmebaNN>ObjectDetectionCallback





以Arduino IDE將模型燒入晶片

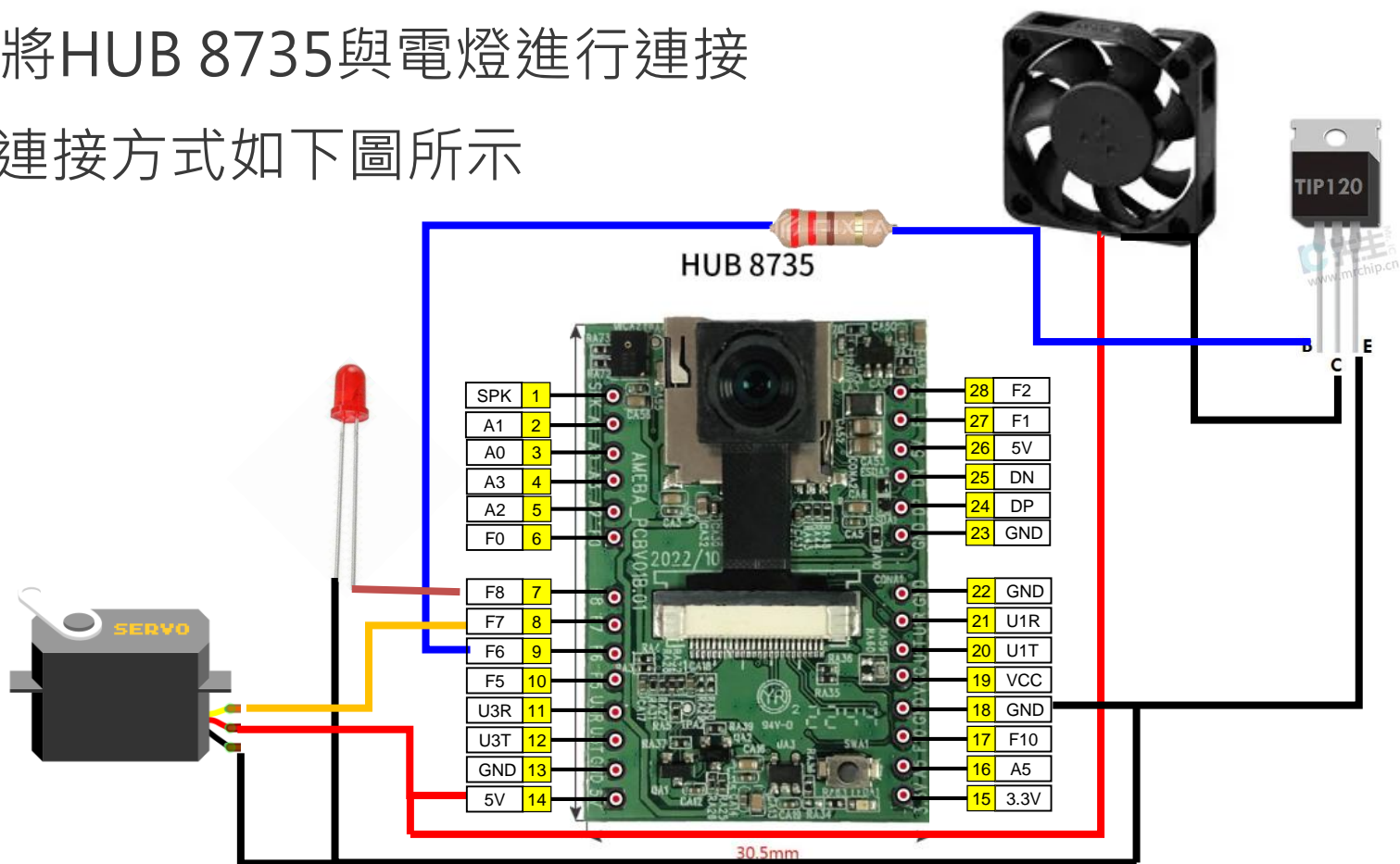
- 這樣就成功完成你燒入了！！





硬體準備(電燈)

- 將HUB 8735與電燈進行連接
連接方式如下圖所示





程式說明

#加入 servo函式庫

```
33 #include <AmebaServo.h>
34 AmebaServo myservo;
```

#在setup裡定義腳位

```
124 pinMode(7, OUTPUT);
125 pinMode(9, OUTPUT);
126 myservo.attach(8);
```

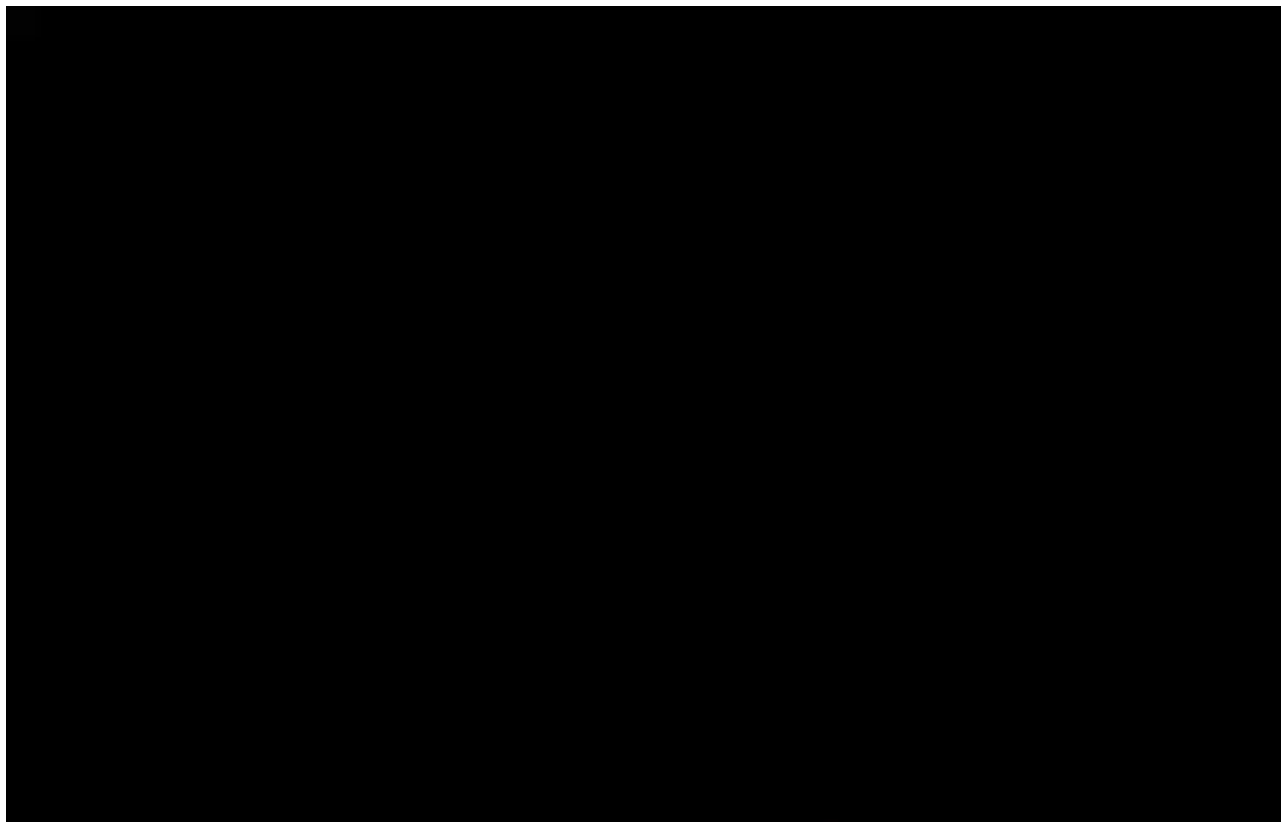
#設定功能

```
173 if (itemList[obj_type].index == 0) {
174     digitalWrite(7, LOW);
175 }
176 if (itemList[obj_type].index == 1) {
177     digitalWrite(7, HIGH);
178 }
179 if (itemList[obj_type].index == 2) {
180     for (pos = 0; pos <= 100; pos += 1) { // tell servo to go to position in variable 'pos'
181         myservo.write(pos);
182         delay(15);
183     }
184     delay(1000);
185 }
186 if (itemList[obj_type].index == 3) {
187     for (pos = 100; pos >= 0; pos -= 1) { // tell servo to go to position in variable 'pos'
188         myservo.write(pos);
189         delay(15);
190     }
191     delay(1000);
192 }
193 if (itemList[obj_type].index == 4) {
194     digitalWrite(9, HIGH);
195 }
196 if (itemList[obj_type].index == 5) {
197     digitalWrite(9, LOW);
198 }
```



成果展示

- 成果如影片所示
- 手勢0:關閉電燈
- 手勢1:開啟電燈
- 手勢2:開啟窗戶
- 手勢3:關閉窗戶
- 手勢4:開啟電扇
- 手勢5:關閉電扇





Thanks.

Every failure is a step to success.