題目：智慧超音波臉部識別警示系統.

Ideas
Hatch

摘要

本專題旨在開發一個智慧型臉部識別系統，以實現對特定人員進行即時識別的功能。系統結合了視訊串流、臉部辨識、Wi-Fi 連接等技術，能夠在不同場景下實現準確的臉部識別，透過聲波感測器能讓辨識人員在適當辨識距離下得知最佳辨識距離，同時有燈光提醒讓辨識人員得知辨識完成。通過詳細的硬體配置和程式碼實現，我們實現了一個功能完善的系統，可以應用於安全監控、出入管理等領域。

Abstract

This project aims to develop an intelligent facial recognition system to achieve real-time identification of specific individuals. The system integrates technologies such as video streaming, facial recognition, and Wi-Fi connectivity to achieve accurate facial recognition in various scenarios. Through detailed hardware configuration and code implementation, we have realized a fully functional system that can be applied in areas such as security surveillance and access management.

目錄

第1章 緒論 1-1 前言

隨著科技的不斷發展，智慧型安全監控系統在現代社會中扮演著越來越重要的角色。在眾多的安全監控技術中，臉部識別技術因其高效、準確的特性而受到廣泛關注。本專題旨在利用現有的技術和資源，開發一個智慧型臉部識別系統，以滿足對於特定人員的快速準確識別需求。

1-2 研究目的

本專題的主要目的包括：

1. 開發一個能夠實時辨識特定人員的智慧型臉部識別系統。
2. 整合視訊串流、臉部辨識、Wi-Fi 連接等功能，實現系統的全面性能。
3. 測試系統在不同場景下的穩定性和準確性，並進行優化和改進。

2-1 HUB 8735 ultra

如圖 2-1 所示，這是 HUB 8735 ultra。

圖 2-1：HUB 8735 ultra

2-1-1 HUB 8735 ultra 介紹

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。

2-1-2 HUB 8735 規格

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

2-2 LCD1602

如圖 2-2 所示，這是 LCD1602。

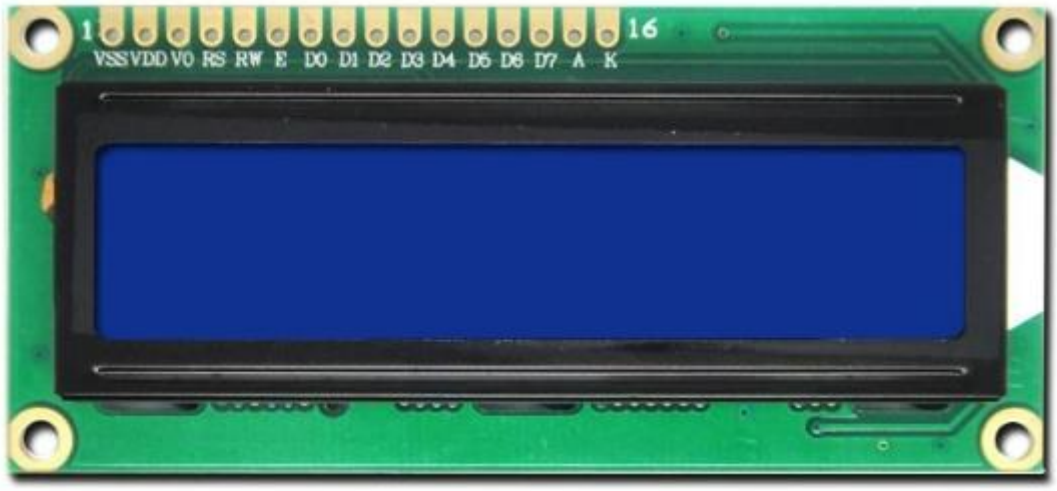


圖 2-2：LCD1602

2-2-1 LCD1602 介紹

LCD1602 是一款功能強大的液晶顯示器，提供可靠的文字顯示功能，可輕鬆與微控制器進行接口連接和控制。它具有優秀的可視性和易於使用的特性，適用於各種應用場景。

2-2-2 LCD1602 規格

2 行 16 字符的清晰顯示
背光功能，可通過單個命令控制開啟或關閉
支持 ASCII DEC 字符 32-127
具有內置的八個自定義字符的顯示功能
可以通過單個命令將游標移動到顯示器的任何位置
集成了調節對比度的功能
電源需求：+5Vdc，約 20mA（不帶背光時），約 150mA（帶背光時）
尺寸：約 80 x 35mm
工作溫度範圍：-20 至+70°C

2-3 HC-SR04

如圖 2-3 所示，這是 HC-SR04。

圖 2-3：HC-SR04。

2-3-1 HC-SR04 介紹

Arduino HC-SR04 是一款基於超聲波技術的距離測量模組，廣泛應用於機器人導航、障礙物檢測等領域。這款模組能夠準確測量目標物體與模組之間的距離，並將其轉換為數字信號，方便與 Arduino 等微控制器進行連接和控制。

2-3-2 HC-SR04 規格

工作電壓： 5V，與常用的 Arduino 系統兼容。
工作原理： 透過發射超聲波脈衝並接收回波，通過計算發射到接收的時間差，以確定目標物體與模組之間的距離。
測量範圍： 通常在 2 厘米到 400 厘米之間，適用於不同範圍的應用場景。
精度： 一般可達到±3 毫米的高精度。
工作頻率： 超聲波發射器和接收器的工作頻率通常在 40kHz 左右。
接口： 包含 VCC（正電源）、Trig（觸發）、Echo（回波）和 GND（地）等接口，方便與外部設備連接。

```
#include "WiFi.h"
#include "StreamIO.h"
#include "VideoStream.h"
#include "AudioStream.h"
#include "AudioEncoder.h"
#include "RTSP.h"
#include "NNFaceDetectionRecognition.h"
#include "VideoStreamOverlay.h"
#include "LiquidCrystal_PCF8574.h"

// Default audio preset configurations:
// 0 :  8kHz Mono Analog Mic
// 1 : 16kHz Mono Analog Mic
// 2 :  8kHz Mono Digital PDM Mic
// 3 : 16kHz Mono Digital PDM Mic

//set LCD
LiquidCrystal_PCF8574 lcd(0x27);

//腳位設定
//HC-SR04
const int trigPin = 18;
const int echoPin = 19;
//LED
const int R = 8;
const int G = 7;
const int B = 6;
// Choose between using AAC or G711 audio encoder
AAC encoder;
//G711E encoder;

AudioSetting configA(3);
Audio audio;
RTSP rtsp;
StreamIO audioStreamer1(1, 1);   // 1 Input Audio -> 1 Output encoder
StreamIO audioStreamer2(1, 1);   // 1 Input encoder -> 1 Output RTSP


#define CHANNEL     0
#define CHANNELNN   3

// Customised resolution for NN
#define NNWIDTH     576
#define NNHEIGHT    320
```

```
VideoSetting config(VIDEO_FHD, 30, VIDEO_H264, 0);
VideoSetting configNN(NNWIDTH, NNHEIGHT, 10, VIDEO_RGB, 0);
NNFaceDetectionRecognition facerecog;

StreamIO videoStreamer(1, 1);
StreamIO videoStreamerFDFR(1, 1);
StreamIO videoStreamerRGBFD(1, 1);

char ssid[] = "vivo Y55s 5G";    // your network SSID (name)
char pass[] = ".00000000";   // your network password
int status = WL_IDLE_STATUS;

IPAddress ip;
int rtsp_portnum;

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

  // 設定超音波模組的引腳
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  // 設定 LED 燈的引腳
  pinMode(R, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(B, OUTPUT);
  //LCD 顯示

  lcd.begin(16, 2);
  lcd.setBacklight(200);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Hello");
  delay(1000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("waitting for");
  lcd.setCursor(0,1);
  lcd.print("connect...");

   // Attempt to connect to Wifi network:
   while (status != WL_CONNECTED) {
      Serial.print("Attempting to connect to WPA SSID: ");
      Serial.println(ssid);
      status = WiFi.begin(ssid, pass);

      // wait 2 seconds for connection:
      delay(2000);
   }

   ip = WiFi.localIP();
```

```
    // Configure audio peripheral for audio data output
    audio.configAudio(configA);
    audio.begin();

    // Configure audio encoder
    // For G711 audio encoder, choose between u-law and a-law algorithm
    encoder.configAudio(configA);
//    encoder.configCodec(CODEC_G711_PCMU);
//    encoder.configCodec(CODEC_G711_PCMA);
    encoder.begin();

    // Configure RTSP with identical audio format information and codec
    rtsp.configAudio(configA, CODEC_AAC);
//    rtsp.configAudio(configA, CODEC_G711_PCMU);
//    rtsp.configAudio(configA, CODEC_G711_PCMA);
    audioStreamer1.registerInput(audio);
    audioStreamer1.registerOutput1(encoder);
    audioStreamer1.begin();

    audioStreamer2.registerInput(encoder);
    audioStreamer2.registerOutput(rtsp);
    audioStreamer2.begin();

    // 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 Face Recognition model
    // Select Neural Network(NN) task and models
    facerecog.configVideo(configNN);
    facerecog.modelSelect(FACE_RECOGNITION, NA_MODEL,
DEFAULT_SCRFD, DEFAULT_MOBILEFACENET);
    facerecog.begin();
    facerecog.setResultCallback(FRPostProcess);

    // 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 face
detection
    videoStreamerRGBFD.registerInput(Camera.getStream(CHANNELNN));
    videoStreamerRGBFD.setStackSize();
    videoStreamerRGBFD.setTaskPriority();
    videoStreamerRGBFD.registerOutput(facerecog);
    if (videoStreamerRGBFD.begin() != 0) {
        Serial.println("StreamIO link start failed");
    }

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

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

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // 讀取超音波返回的脈衝時間
  long duration = pulseIn(echoPin, HIGH);

  // 將脈衝時間轉換為距離（公分）
  int D = duration * 0.034 /2;
   if(D<5){
  digitalWrite(B,0);
  digitalWrite(R,0);
  digitalWrite(G,1);



    if (Serial.available() > 0) {
        String input = Serial.readString();
        input.trim();

        if (input.startsWith(String("REG="))){
            String name = input.substring(4);
            facerecog.registerFace(name);
            lcd.clear();
            lcd.setCursor(0,0);
```

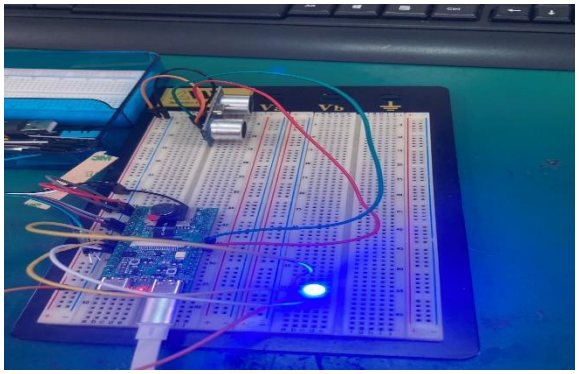
```
            lcd.print("Hello");
            lcd.setCursor(0,1);
            lcd.print(name);
        } else if (input.startsWith(String("DEL="))) {
            String name = input.substring(4);
            facerecog.removeFace(name);
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("BEY BEY");
            lcd.setCursor(0,1);
            lcd.print(name);
        } else if (input.startsWith(String("RESET"))) {
            facerecog.resetRegisteredFace();
        } else if (input.startsWith(String("BACKUP"))) {
            facerecog.backupRegisteredFace();
        } else if (input.startsWith(String("RESTORE"))) {
            facerecog.restoreRegisteredFace();
        }
    OSD.createBitmap(CHANNEL);
    OSD.update(CHANNEL);
   }
   }
    else if(D>5 && D<20){
   digitalWrite(R,0);
   digitalWrite(G,0);
   //LCD 訊息:請靠近一點
  lcd.clear();
  lcd.setCursor(2,0);
  lcd.print("Come Closer");
  digitalWrite(B,1);
  delay(200);
  digitalWrite(B,0);
  delay(200);
  digitalWrite(B,1);
  delay(200);
  digitalWrite(B,0);
  delay(1000);
 }
 else if(D>20){
   digitalWrite(R,1);
   digitalWrite(G,0);
   digitalWrite(B,0);
 }



   delay(1000);

}
```

```
// User callback function for post processing of face recognition results
void FRPostProcess(std::vector<FaceRecognitionResult> 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 faces detected = %d\r\n", facerecog.getResultCount());
    OSD.createBitmap(CHANNEL);

    if (facerecog.getResultCount() > 0) {
        for (uint32_t i = 0; i < facerecog.getResultCount(); i++) {
            FaceRecognitionResult 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);

            uint32_t osd_color;
            if (String(item.name()) == String("unknown")) {
                osd_color = OSD_COLOR_RED;
            } else {
                osd_color = OSD_COLOR_GREEN;
            }

            // Draw boundary box
            printf("Face %d name %s:\t%d %d %d %d\n\r", i, item.name(), xmin, xmax,
ymin, ymax);
            OSD.drawRect(CHANNEL, xmin, ymin, xmax, ymax, 3, osd_color);

            // Print identification text above boundary box
            char text_str[40];
            snprintf(text_str, sizeof(text_str), "Face:%s", item.name());
            OSD.drawText(CHANNEL, xmin, ymin - OSD.getTextHeight(CHANNEL),
text_str, osd_color);
        }
    }
    OSD.update(CHANNEL);
}
void printInfo_LCD(void){
  ip = WiFi.localIP();
  lcd.clear();
  lcd.setCursor(4,0);
```

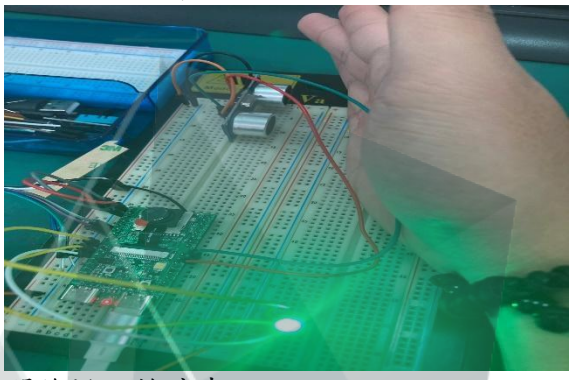
```
  lcd.print("complete");
  delay(1000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("rstp://");
  lcd.setCursor(0,1);
  lcd.print(ip);
  delay(1000);

}
```

Ideas Hatch

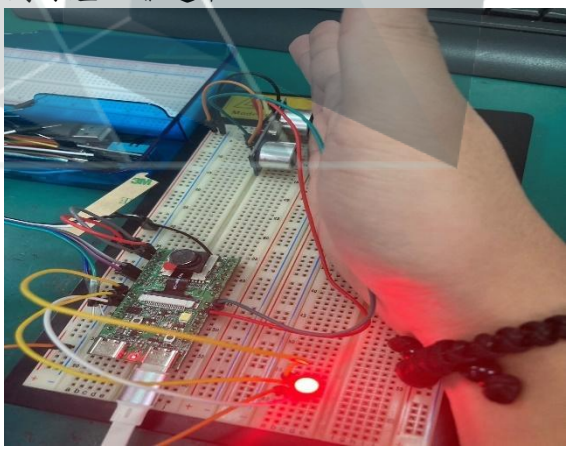
第4章 實作成品



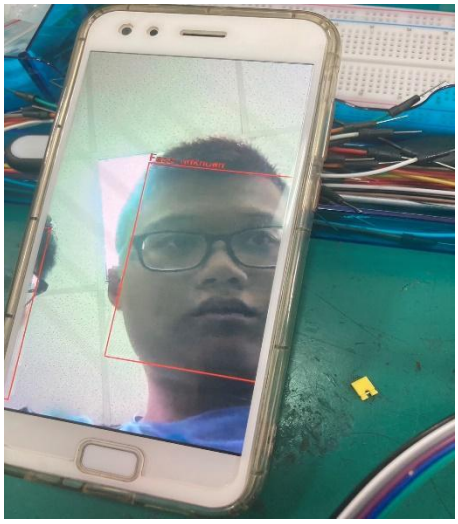
閃藍燈提醒靠近



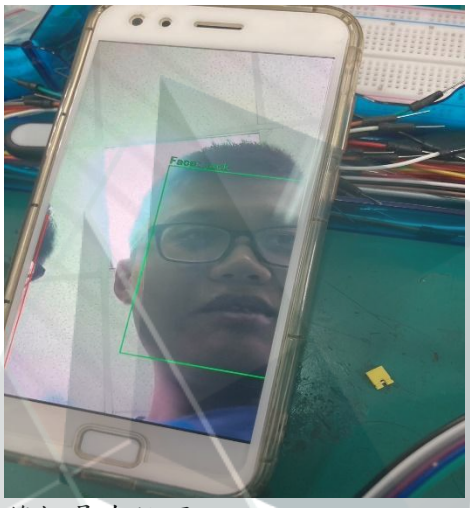
閃綠燈距離適中



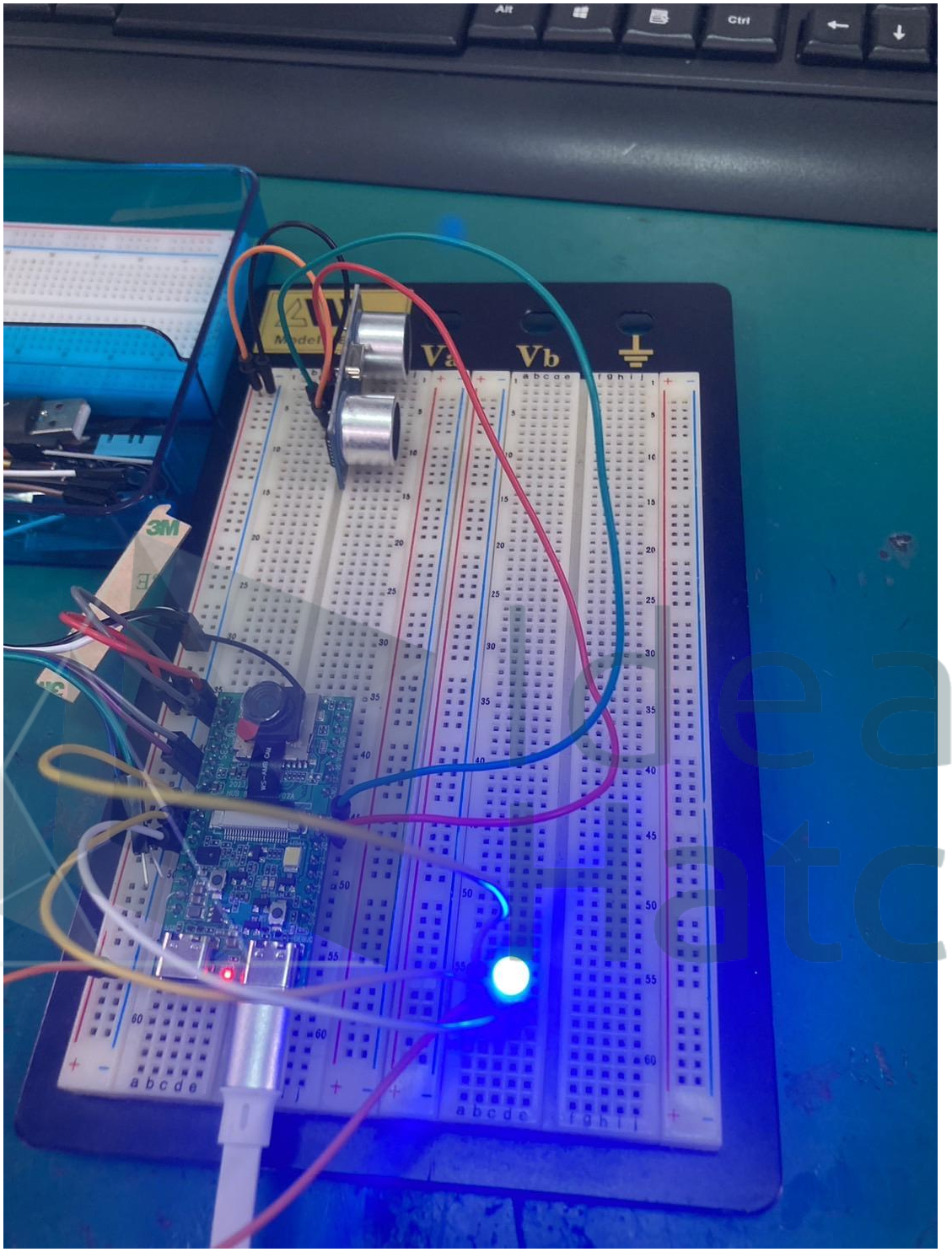
閃紅燈距離太近

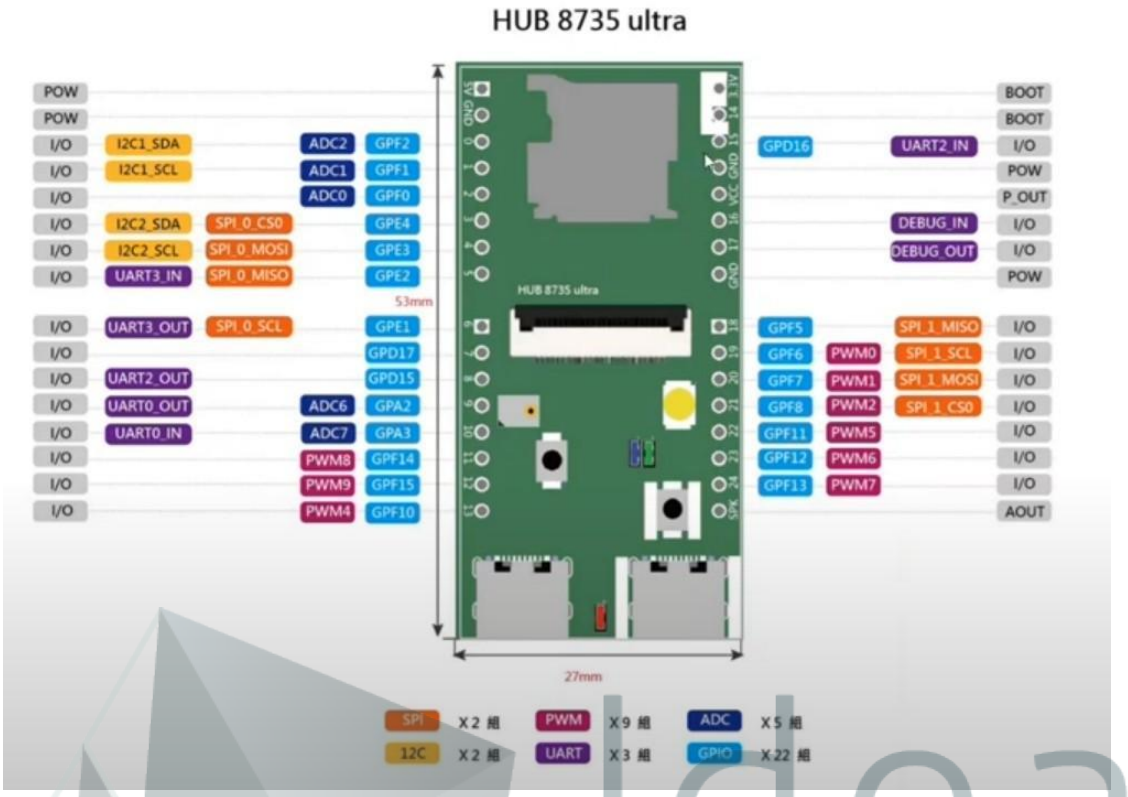


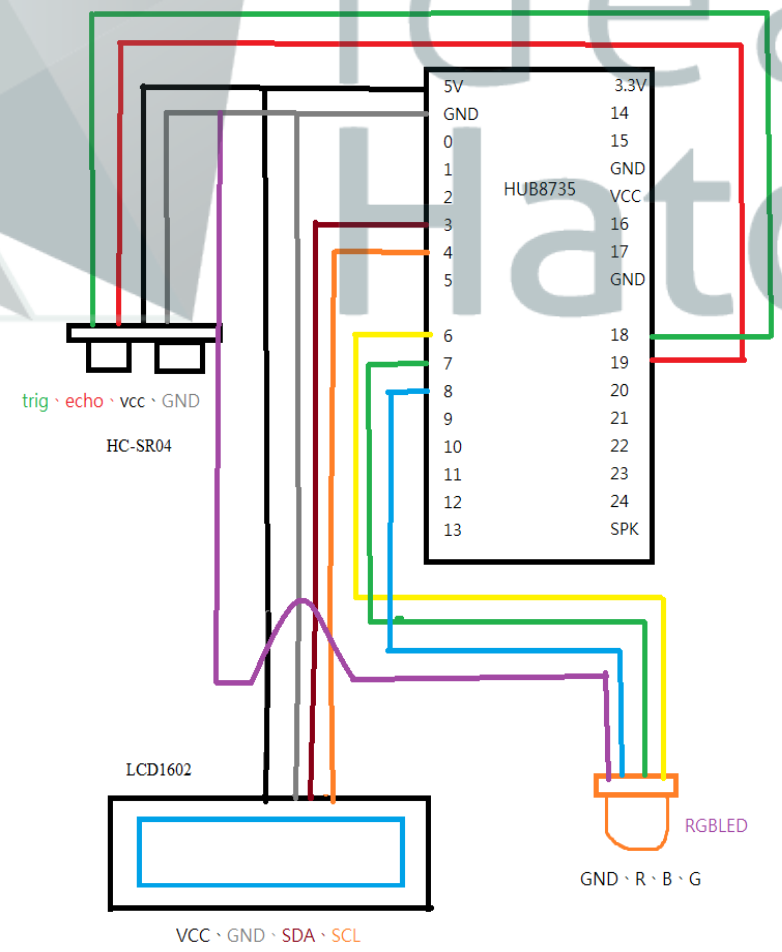
紅框是未註冊



綠框是有註冊


Alt
Ctrl
Va
Vb
3M


HUB 8735 ultra
POW
POW
I/O
I/O
I/O
I/O
I/O
I/O
I/O
I/O
I/O
I/O
I/O
I/O
I/O
I/O
I/O
I/O
I2C1_SDA
I2C1_SCL
I2C2_SDA
I2C2_SCL
UART3_IN
UART3_OUT
UART2_OUT
UART0_OUT
UART0_IN
SPI_0_CS0
SPI_0_MOSI
SPI_0_MISO
SPI_0_SCL
ADC2
ADC1
ADC0
ADC6
ADC7
PWM8
PWM9
PWM4
GPF2
GPF1
GPF0
GPE4
GPE3
GPE2
GPE1
GPD17
GPD15
GPA2
GPA3
GPF14
GPF15
GPF10
53mm
27mm
HUB 8735 ultra
GPD16
UART2_IN
DEBUG_IN
DEBUG_OUT
GPF5
GPF6
GPF7
GPF8
GPF11
GPF12
GPF13
PWM0
PWM1
PWM2
PWM5
PWM6
PWM7
SPI_1_MISO
SPI_1_SCL
SPI_1_MOSI
SPI_1_CS0
BOOT
BOOT
I/O
POW
P_OUT
I/O
I/O
POW
I/O
I/O
I/O
I/O
I/O
I/O
I/O
AOUT
SPI X2 組
12C X2 組
PWM X9 組
UART X3 組
ADC X5 組
GPIO X22 組


5V
GND
0
1
2
3
4
5
6
7
8
9
10
11
12
13
3.3V
14
15
GND
VCC
16
17
GND
18
19
20
21
22
23
24
SPK
HUB8735
trig、echo、vcc、GND
HC-SR04
LCD1602
VCC、GND、SDA、SCL
RGBLED
GND、R、B、G

接線解說:

5V 與 GND 接到 LCD1602 和 HC-SR04 的 VCC 跟 GND 相接，GPE4、GPE3 分別與 LCD1602 SDA 跟 SCL 相接，GPE1、GPD17、GPD15 分別接到 RGBLED 的三隻接腳，最長的接腳接地，GPF5、GPF6 與 HC-SR04 的 trig、echo 相接

功用

距離感應控制燈號：當乘客站在月台邊緣時，若距離太近，系統會自動亮起紅色燈號，提醒乘客保持安全距離；若距離適中，則亮起綠色燈號，表示可以進入車廂；若距離太遠，則系統會閃爍藍色燈號，提醒乘客前進以便讓其他乘客進入車廂。

人臉辨識：系統可以辨識月台上的人臉，這可以用於多種用途

個人安全監控：系統可以偵測是否有人站在禁止區域或危險區域，以防止意外事件發生。

C:\Users\lhu_F201\Desktop\739945546.592614.mp4

第5章 結論

5-1 結論

通過本專題的研究和開發，我們成功地實現了一個智慧型臉部識別系統，具有高效、準確的識別能力。該系統集成了多項先進技術，包括視訊串流、臉部辨識等，可以應用於各種場景，如安全監控、出入管理等。未來，我們將繼續改進系統的性能和功能，以滿足不斷變化的需求。

5-2 建議

1. 進一步優化系統的算法和效能，提高識別的準確性和速度。
2. 加強系統的安全性和穩定性，防止未授權訪問和故障。
3. 擴展系統的應用範圍，如結合人流統計、行為分析等功能，提高系統的智能化水平。