Anti_SPY

第九組

范姜揚 E94081107
黃鼎元 P46061411
廖品瑜 C24061143

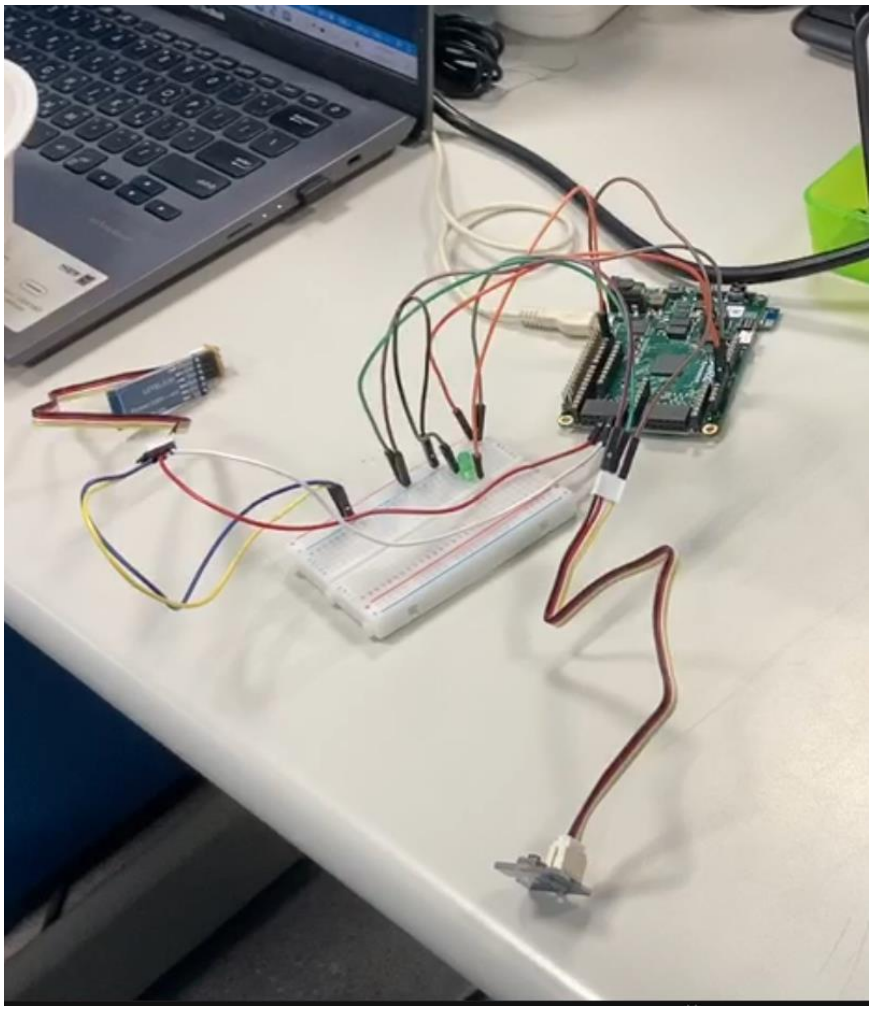
目錄

動機

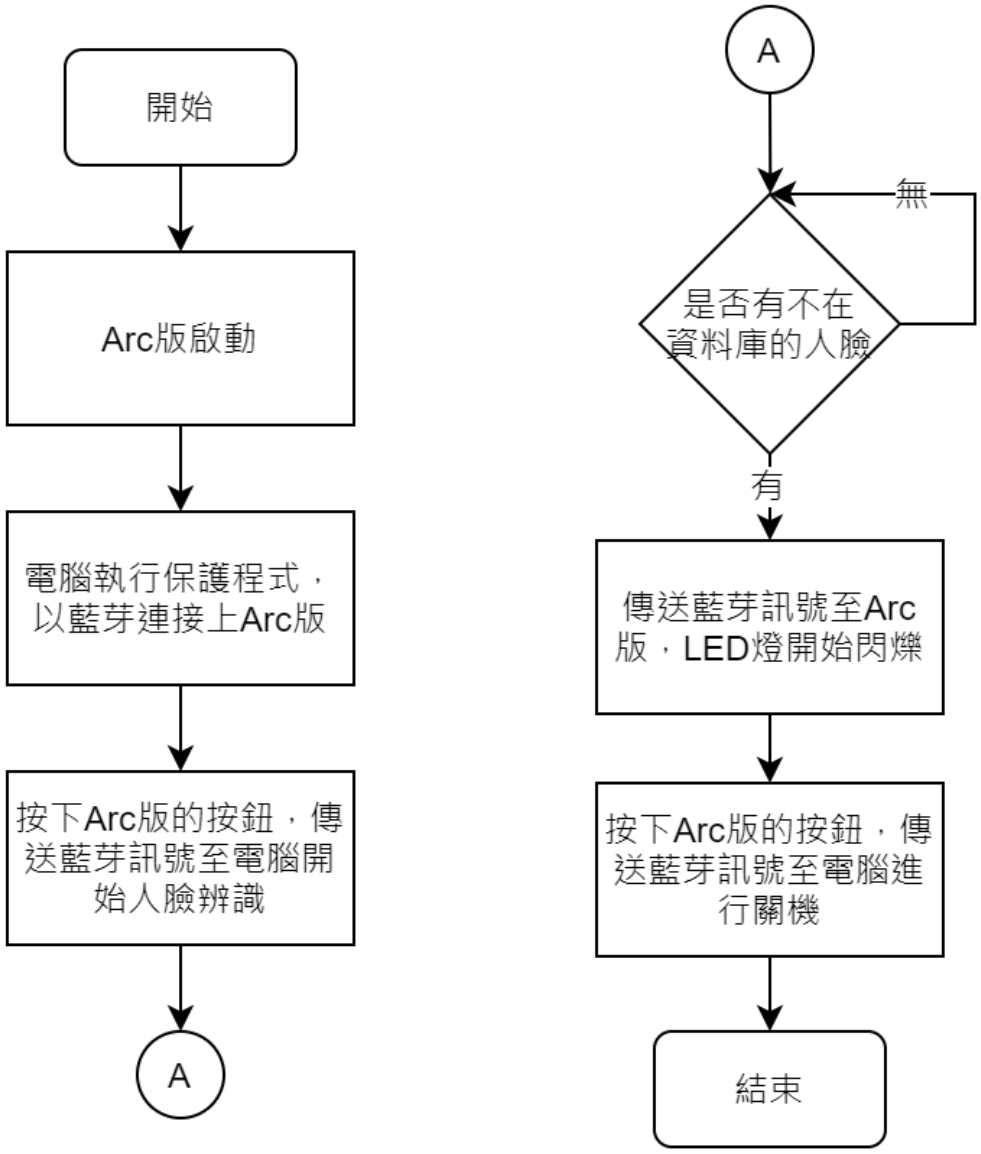
有時候因一些事情暫時離開電腦，又不想要讓電腦進螢幕保護程式，剛好手邊有嵌入式版以及一些人臉辨識的相關資料，就有了這次的 demo，設計一個保有彈性的防盜系統。

使用材料

1. 有攝像頭、藍芽的電腦
2. ARC IOTdk
3. 藍芽模組 HC-05
4. 按鈕
5. LED
6. 蜂鳴器故障(故沒用)



流程圖


開始
Arc版啟動
電腦執行保護程式，以藍芽連接上Arc版
按下Arc版的按鈕，傳送藍芽訊號至電腦開始人臉辨識
A
A
是否有不在資料庫的人臉
無
有
傳送藍芽訊號至Arc版，LED燈開始閃爍
按下Arc版的按鈕，傳送藍芽訊號至電腦進行關機
結束

系統介紹

IoTdk：

此開發版以藍芽連接電腦，透過按鈕傳送指令。藍芽使用 HC-05 模組，以 UART 口連接，按鈕與 LED 接在 GPIO 處。

程式碼：

```
#include "embARC.h"
#include "embARC_debug.h"

#define GPIO4B2_1_OFFSET    1
#define GPIO4B2_3_OFFSET    3

void t1_isr(void *ptr)  //use Timer to make the LED blink
{
    // Clear IP first
    timer_int_clear(TIMER_1);

    uint32_t read_value;
    DEV_GPIO_PTR gpio_4b2 = gpio_get_dev(DFSS_GPIO_4B2_ID);
    gpio_4b2->gpio_read(&read_value, 1<<GPIO4B2_3_OFFSET);
    gpio_4b2->gpio_write(~read_value, 1<<GPIO4B2_3_OFFSET);
}

int main(void)
{
    // get 4b2 gpio object
    DEV_GPIO_PTR gpio_4b2 = gpio_get_dev(DFSS_GPIO_4B2_ID);
    /*
      Open GPIO
      input   : 0, 1, 2
      output  : 3
    */
    gpio_4b2->gpio_open(1<<GPIO4B2_3_OFFSET);
    uint32_t read_value = 0;
```

```
    DEV_UART_PTR uart_1 = uart_get_dev(DFSS_UART_1_ID);
    uart_1->uart_open(9600);
    uint8_t read_value_blue = 0;
    //偵測按鈕是否被按
    while(1)
    {
        gpio_4b2->gpio_read(&read_value, 1<<GPIO4B2_1_OFFSET);
        //偵測到按鈕被按
        if(read_value>>1 == 1)
        {
            uart_1->uart_write("b", 1);  //透過按鈕發送"b"到處理
人臉辨識的電腦以啟動裝置
            board_delay_ms(1000, 0);
            break;
        }
    }
    read_value = 0;  //reset read_value 值以進行下一輪按鈕偵測

    while(1)
    {
        //裝置還沒收到來自另一台電腦的訊號'a'
        while(read_value_blue != 'a')
        {
            uart_1->uart_read(&read_value_blue, 1);  //透過藍芽接
收訊號
            EMBARC_PRINTF("a:%c\n", read_value_blue); //test
        }

        //裝置透過藍芽接收到來自另一台電腦的訊號'a'並且讓LED燈開
始閃以通知使用者

        // Reset TIMER1
        // Check whether TIMER1 is available before use it
        if(timer_present(TIMER_1))
        {
            // Stop TIMER1 & Disable its interrupt first
            timer_stop(TIMER_1);
```

```
            int_disable(INTNO_TIMER1);

            // Connect a ISR to TIMER1's interrupt
            int_handler_install(INTNO_TIMER1, t1_isr);

            // Enable TIMER1's interrupt
            int_enable(INTNO_TIMER1);

            // Start counting, 1 second request an interrupt
            timer_start(TIMER_1, TIMER_CTRL_IE, BOARD_CPU_CLOCK);
        }

        //偵測使用者是否透過按鈕對於閃燈作出回應
        while(1)
        {
            gpio_4b2->gpio_read(&read_value,
                                1<<GPIO4B2_1_OFFSET);
            //使用者按下按鈕，關閉裝置並且傳送"c"到另一台電腦促使
其關機
            if(read_value>>1 == 1)
            {
                uart_1->uart_write("c", 1);
                board_delay_ms(1000,0);  //若沒有 delay 會導致訊號
無法送達
                gpio_4b2->gpio_write(0<<GPIO4B2_3_OFFSET,
                                1<<GPIO4B2_3_OFFSET);  //關燈
                break;
            }
        }
        break;
    }

    gpio_4b2->gpio_close();
    uart_1->uart_close();

    return E_SYS;
}
```

電腦：

以藍芽連接開發板，當偵測到未在資料庫的人臉時，發出訊號。

程式碼：

```
import asyncio
from bleak import BleakClient
import face_recognition
import cv2
import numpy as np
global process_this_frame
process_this_frame = True
video_capture = cv2.VideoCapture(0)

obama_image = face_recognition.load_image_file("obama.jpg")
obama_face_encoding = face_recognition.face_encodings(obama_image)[0]

biden_image = face_recognition.load_image_file("biden.jpg")
biden_face_encoding = face_recognition.face_encodings(biden_image)[0]

max_image = face_recognition.load_image_file("max.jpg")
max_face_encoding = face_recognition.face_encodings(max_image)[0]


known_face_encodings = [
    obama_face_encoding,
    biden_face_encoding,
    max_face_encoding
]
known_face_names = [
    "Barack Obama",
    "Joe Biden",
    "Max"
]

# Initialize some variables
```

```
face_locations = []
face_encodings = []
face_names = []



address = '88:25:83:F0:2A:8D'
CHARACTERISTIC_UUID = '0000FFE1-0000-1000-8000-00805F9B34FB'
global read
global video_controller
video_controller = 1
read = 'a'

def notification_handler( sender, data):
    global read
    data = data.decode('utf-8')
    print(data)
    if(data == 'b'):
        read = 'b'
    if(data == 'c'):
        read = 'c'




async def run(address):
    global read
    global process_this_frame
    client = BleakClient(address)
    try:
        await client.connect()

        while(read == 'a'):
            await client.start_notify(CHARACTERISTIC_UUID, notification
_handler)
            await asyncio.sleep(1)
            await client.stop_notify(CHARACTERISTIC_UUID)

        print('starting video')
```

```
        global video_controller
        while (video_controller != 0):
            ret, frame = video_capture.read()
            if (ret):
                print('starting video2')
                small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.2
5)
                rgb_small_frame = small_frame[:, :, ::-1]

                if process_this_frame:
                # Find all the faces and face encodings in the current
frame of video
                    face_locations = face_recognition.face_locations(rg
b_small_frame)
                    face_encodings = face_recognition.face_encodings(rg
b_small_frame, face_locations)

                    face_names = []
                    for face_encoding in face_encodings:
                        # See if the face is a match for the known face
(s)
                        matches = face_recognition.compare_faces(known_
face_encodings, face_encoding)
                        name = "Unknown"

                        # # If a match was found in known_face_encoding
s, just use the first one.
                        # if True in matches:
                        #     first_match_index = matches.index(True)
                        #     name = known_face_names[first_match_index
]

                        # Or instead, use the known face with the small
est distance to the new face
                        face_distances = face_recognition.face_distance
(known_face_encodings, face_encoding)
                        best_match_index = np.argmin(face_distances)
                        if matches[best_match_index]:
```

```
                            name = known_face_names[best_match_index]

                        face_names.append(name)

                process_this_frame = not process_this_frame


                # Display the results
                for (top, right, bottom, left), name in zip(face_locati
ons, face_names):
                    # Scale back up face locations since the frame we d
etected in was scaled to 1/4 size
                    top *= 4
                    right *= 4
                    bottom *= 4
                    left *= 4

                    # Draw a box around the face
                    cv2.rectangle(frame, (left, top), (right, bottom),
(0, 0, 255), 2)

                    # Draw a label with a name below the face
                    cv2.rectangle(frame, (left, bottom - 35), (right, b
ottom), (0, 0, 255), cv2.FILLED)
                    font = cv2.FONT_HERSHEY_DUPLEX
                    cv2.putText(frame, name, (left + 6, bottom - 6), fo
nt, 1.0, (255, 255, 255), 1)

                # Display the resulting image
                cv2.imshow('Video', frame)
                for (top, right, bottom, left), name in zip(face_locati
ons, face_names):
                    if name == "Unknown":
                        print('find unknown!!')
                        cv2.waitKey(8000)
                        video_controller = 0;


                # Hit 'q' on the keyboard to quit!
```

```
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break
        #face_recog()
        #capture unknown then send signal
    # Release handle to the webcam
        video_capture.release()
        cv2.destroyAllWindows()
        await client.write_gatt_char(CHARACTERISTIC_UUID, b'a')

        while(read == 'b'):
            await client.start_notify(CHARACTERISTIC_UUID, notification
_handler)
            await asyncio.sleep(1)
            await client.stop_notify(CHARACTERISTIC_UUID)

        if(read == 'c'):
            print('we are going to reboot this computer')
            import os
            os.system('shutdown /p /f')



    except Exception as e:
        print(e)
    finally:
        await client.disconnect()

loop = asyncio.get_event_loop()
loop.run_until_complete(run(address))
```

心得

盧棻和:

我們在這次的 Final Project 花了不少的心力，最後一次的討論甚至在資工的實驗室做到凌晨 3 點，這是從來沒有過的體驗，覺得非常的特別。我在

這次 Project 中共同負責 ARC 板端的程式碼，包含透過按鈕傳送訊號到處理人臉辨識的電腦以啟動裝置、讓 LED 燈在接到異常訊號後閃爍以及透過按鈕發送讓處理人臉辨識的電腦關機的訊號。之前實驗課做 ARC 板的時候做的不太上手，但是這次情況就好很多了，最後也要感謝助教在藍芽方面的協助，如果沒有這方面的幫助我們自己大概也做不出來。

廖品瑜:

原本的構想是可以隨身攜帶的防 spy 裝置，但因為不知道如何燒錄到板子內，只能連接到另一台電腦操作，覺得有點可惜。實際看到整個流程的成功，覺得非常有成就感。

范姜揚:

這次我是負責處理板子的藍芽連接部分，原本是要用板子上的藍芽套件，但是資源實在是太少了，加上學長告知很難用，馬上換成外接的藍芽模組去操作。AT09 原本以為會很麻煩，後來發現幾乎跟接線一樣，只要把 uart open 設定對 baud rate 就可以 read write 了，這次實驗串聯了版子跟電腦上的程式，如果有更多的時間相信我們能設計的更好