IoT Lab3 report

一、 Objective:

使用RFID sensor判別該對象持有卡片是否授權，另外設置警報系統，其中以buzzer嚇阻非授權的人員，並使用line notify傳送偵測到非授權卡的通知。

使用Device	數量	備註
Raspberry pi	2	
LED	2	一紅一綠
Buzzer	1	
RFID sensor	1	
Switch	1	

二. Specification of sensors and actuators used

Table 1, Sensors and Actuators

pi 1	pi 2
RFID sensor 感測	Buzzer 控制
LED 指示燈	Switch
	Line notify

兩個raspberry pi各自負責範圍

Table 2, Distribution

三. System design

使用RFID sensor來讀取RFID卡的卡號判斷是否授權，並將卡號上傳至MCS。此外，若授權則亮起綠色LED燈即可，若非授權則亮起紅色LED燈、以及在另一台raspberry pi傳送line notify通知並響起buzzer，若要停止buzzer，則必須按下switch的開關。

狀態	LED	Buzzer		Line
授權	綠燈	X		X
非授權	紅燈	未按switch	持續發出聲響	傳送alert訊息
		按下switch	停止	

Table 3, 裝置情況

四. Flowcahrt

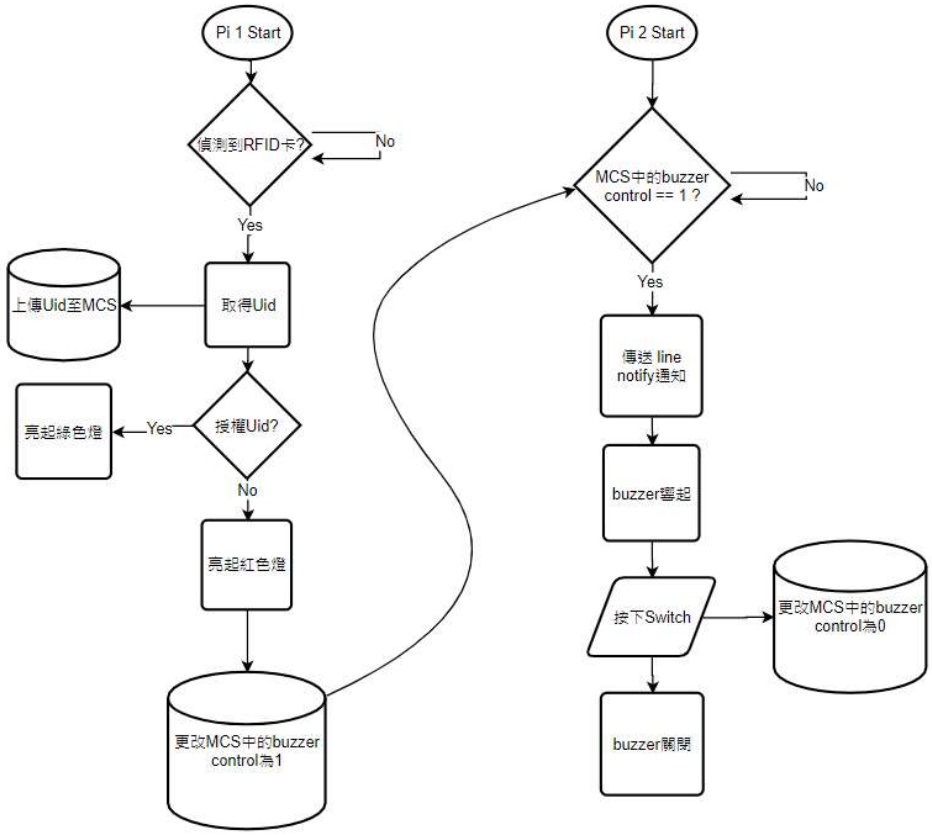
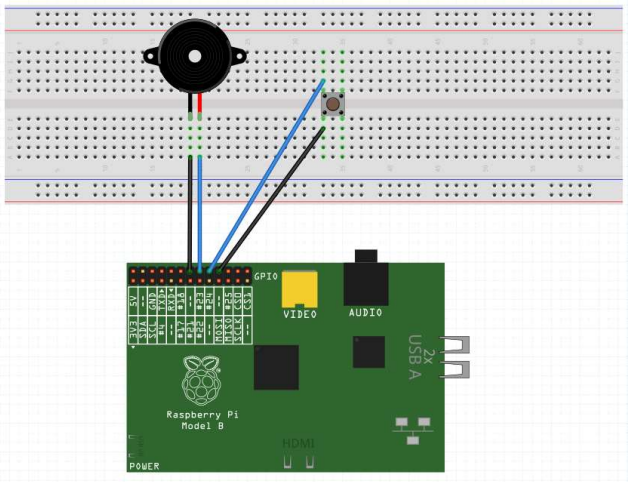


Fig. 1, Flowchart

五.線路圖



因RFID sensor的線路圖在Lab1的report已有在此即不加入，新增的部分是buzzer，連接狀況如Fig.2。

Fig. 2, buzzer與rpi連結示意圖

六. Source Code

0. Preface

本次使用兩台raspberry pi，其中一台負責讀取RFID卡號並上傳至MCS，因為在Lab2的report已敘述，在此即不再贅述。

```
5  import requests
6  import http.client, urllib
7  import socket
8  import json
9
10
11 import time
12 import RPi.GPIO as GPIO
```

1. Import Lib:

Fig. 3, library involved

a. request, http.client, urllib, socket, json: post_to_mcs()與get_to_mcs()使用

b. time: 為sleep()所使用的，主要在控制buzzer的頻率

c. os: 為操作line_notify_msg()時所使用

```
21 ### buzzer setting ###
22 GPIO.setwarnings(False)
23 GPIO.setmode(GPIO.BCM)
24
25 BUZZER_PIN = 23
26 BUTTON_PIN = 24
27 GPIO.setup(BUZZER_PIN,GPIO.OUT)
28 GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
29 ### buzzer setting ###
```

2. GPIO設定

Fig. 4, GPIO for buzzer

如電路圖，Buzzer的控制接腳為23，Switch的接收接腳為24，在此須注意的是，因為Switch的另一端接的是Ground，故按下開關，形成閉路後將會形成低電位，因此透過上拉電阻(line 28)讓開路時維持在高電位，讓按鈕的開與關形成不同的電位。

3. function:

a. MCS相關: 有get_to_mcs(): 使用的是助教的Code，用於檢查MCS中的Buzzer_Control是否為1，若為1則啟動警報系統; 還有post_to_mcs(): 也是使用的是助教的Code，用於在警報系統解除後將Buzzer_Control更改回0，以便接受下一次的警報訊號。

```
94 def main():
95     while(True):
96         if (get_to_mcs() == "1"):
97             alert()
98         time.sleep(0.5)
```

b. main()與alert()

Fig. 7, alert function, main

```
62 def alert():
63     line_notify_msg()
64     buzz()
```

i. main(): 主要function，每0.5秒檢查一次Buzzer_control是否為1，若為1則呼叫alert()，啟動警報系統。

Fig. 8, alert function, alert

ii. alert(): 警報function，本次使用的功能有:

- line的訊息通知，line_notify_msg()
- Buzzer的警報聲，buzz()。

c. 警報系統內容

```
70 def line_notify_msg():
71     os.system("""curl -H "Authorization: Bearer ebNm5twYljYeNQKqHOkBgnCDjIvM
72     -X POST https:/notify-api.line.me/api/notify  -F "message=alert!!!"
73     """)
```

Fig. 9, line_notify_msg

i. line_notify_msg(): 要使用line notify做通知，必須前往line的官方網站申請token，該token為一段代碼(即bearer後方的一長串)，而該token可以與自己的帳號或是自己有加入的群組連動，在本次Lab3是設定與自己的帳號連動，所以在line_notify_msg()使用該token將會傳送訊息給自己的帳號。

- ”-H”為Extra Header之意，為傳送token用途。
- ”-X”為Request之意，目的為與HTTP Server進行溝通，可操作的選項有GET、PUT…，在此使用的是POST的功能。
- ”-F”為Form之意，即傳送的訊息內容，message為訊息變數名稱，alert!!!為訊息內容。

```
80  def buzz():
81      for i in range(200):
82          GPIO.output(BUZZER_PIN,True)
83          time.sleep(0.1)
84          GPIO.output(BUZZER_PIN,False)
85          time.sleep(0.1)
86          if GPIO.input(BUTTON_PIN) == GPIO.LOW:
87              payload = {"datapoints":[{"dataChnId":"buzzer","values":{"va
88              post_to_mcs(payload)
89              break
```

Fig. 10, buzz

ii. buzz(): 產生警報聲用function，Buzzer在切換電位時會產生聲音，因此可藉由圖中的time.sleep()控制其頻率。

在響聲的途中，若按下Switch的按鈕，將會使BUTTON_PIN變成低電位，並做出兩個動作，一是將MCS中的Buzzer_Control更改回0，二是跳出迴圈，意即停止聲響。

七、實作影片: https://youtu.be/Lo2z1HvXr-8