IoT Final 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。

此外，本次加入嫌疑人員拍照的功能，在非授權的人員刷卡後，將會拍攝照片並透過 line notify 傳送。

並且本次加入卡片寫入的功能，與 arduino 系統合作，以便提供更多授權卡片並達到客製化的自我防盜裝置。

二、 Specification of sensors and actuators used

使用Device	數量	備註
Raspberry pi	2	
LED	2	一紅一綠
Buzzer	1	
RFID sensor	1	有寫入功能
Switch	1	
Camera	1	本次加入

Table 1, Sensors and Actuators

兩個 raspberry pi 各自負責範圍如下

pi 1	pi 2
RFID sensor 感測	Buzzer 控制
LED 指示燈	Switch
RFID 卡片寫入	Line notify
	Camera

Table 2, Distribution

三、 System design

1. 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。
   - 若授權: 亮起綠色 LED 燈即可。
   - 若非授權: 亮起紅色 LED 燈，並在另一台 raspberry pi 啟動 camera 拍照，同時透過 line notify 傳送照片 ，以及響起 buzzer。若要停止 buzzer，則必須按下 switch 的開關。
2. 獨立系統: Arduino 控制寫入號碼，可以自行加到讀取機制中授權號碼

list 中，讀取到空卡片後將自己所想要的 UID 碼寫到 arduino 程式，即可完成寫入。

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

Table 3, 可能狀況顯示

四、 Flowcahrt

主要透過不同系統分成三個大類做區分: Pi 1, Pi 2, Arduino，如 Fig. 1。

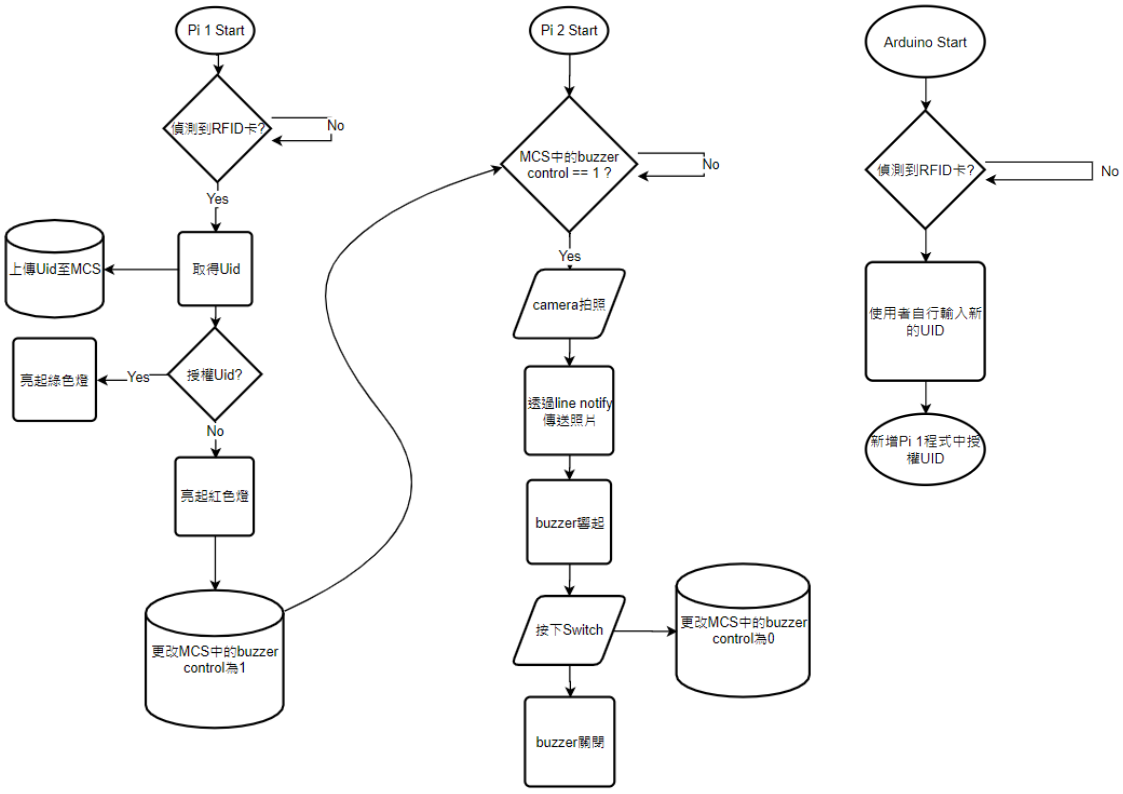

Fig. 1, Flowchart

五.線路圖

因 RFID sensor 的線路圖在 Lab1 的 report 已有在此即不加入，且本次加入的 camera 直接與 raspberry pi 連結，不需額外的 wire 連結，實際連結情況如 Fig. 2，buzzer 部分則為 lab3 也不多做描述，而 Arduino 的部分連接如 Fig. 3。

Fig. 2, buzzer 與 rpi 連結示意圖

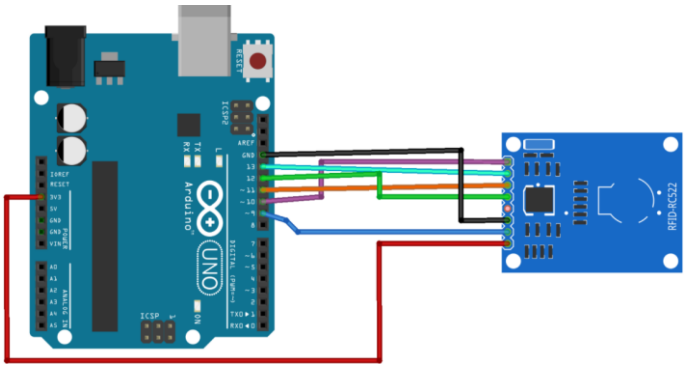

Fig. 3, Arduino 與 RC522 版連接圖

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。

1. 警報系統新增內容

```
77  def camera():
78      camera = PiCamera()
79      camera.rotation = 180
80      camera.capture('/home/pi/Desktop/image.jpg')
```

Fig. 4, camera 功能實現

a. camera(): 使用 PiCamera()可以產生 camera 物件，並使用該物件進行拍攝的操作，camera.rotation 可以旋轉拍攝的照片 ，camera.capture 為拍攝的 function，參數為存檔路徑。

```
66  def line_notify_image():
67      os.system("""curl -H "Authorization: Bearer ebNm5twYljYeNQKqHOkBgnCDjIvMcLwDwzonqKMA9FL"
68      -X POST https:/notify-api.line.me/api/notify  -F "message=alert!!!" -F "imageFile=@image.jpg"
69      """)
```

Fig. 5, line 實現上傳照片

b. line_notify_image(): 連動方式跟上次 lab3 相仿，因為要傳送照片訊息，所以加入 prefix “@” 用來描述要傳送的檔案的位置，照片會先在本地端儲存，之後透過 line api 的呼叫把本地的資料傳上去。

2. Arduino 寫入內容:

在這次的實驗中，為了可以自己客製化所授權的卡片 UID，所以再加上了寫入的功能，可以讓使用者寫空卡到自己的 UID 系統，成功打造專屬的智慧防盜系統。

裝置部分需要使用 Arduino 來完成，作法與 raspberry pi 在以往實驗中讀取跟寫入的部分相距不大，都是透過 mfrc522 的函式庫去引入功能，以下是研究函式庫功能後做出的讀取跟寫入程式介紹。

a. 讀取:

i. 因為需要從主裝置接收資料，所以需要引入 SPI 函式庫；除此之外整個程式都是透過 mfrc522 的函式庫來做讀取，也把它加上。

腳位的部分只需要特別說明 RST 跟 SDA 即可，其他的有自己確定的位置無法更動。

```
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN      9
#define SS_PIN       10

MFRC522 mfrc522(SS_PIN, RST_PIN);
```

Fig. 6, 引入函式庫與設定腳位

ii. 初始設定: 9600 是配合電腦 clock 的設定，並且開始接收資料，並將 mfrc522 開啟。

```
void setup() {
  Serial.begin(9600);
  Serial.println("RFID reader is ready!");

  SPI.begin();
  mfrc522.PCD_Init();
}
```

Fig. 7, 初始設定

iii. 主程式: 在 loop 中持續觀察是否有新卡片進到讀取處，如果有的話就會進行幾個步驟:

- 讀取 UID 與長度
- 顯示卡片類型
- 顯示卡片資料長度
- 用十六進位方式顯示資料(最後會得到四組兩位的數字)

```
RFID reader is ready!
PICC type: MIFARE 1KB
UID Size: 4
id[0]: B5
id[1]: 2C
id[2]: 8F
id[3]: 42
```

Fig. 8, 輸出範例

```
void loop() {
    // make sure whether there is a new card
    if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
      byte *id = mfrc522.uid.uidByte;   // read the UID
      byte idSize = mfrc522.uid.size;   // get UID length

      Serial.print("PICC type: ");      // display card type
      MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
      Serial.println(mfrc522.PICC_GetTypeName(piccType));

      Serial.print("UID Size: ");       // display UID length
      Serial.println(idSize);

      for (byte i = 0; i < idSize; i++) {  // display UID with hex
        Serial.print("id[");
        Serial.print(i);
        Serial.print("]: ");
        Serial.println(id[i], HEX);
      }
      Serial.println();
      mfrc522.PICC_HaltA();  // halt mode
    }
}
```

Fig. 9, 主要迴圈內容

b. 寫入:

i. 填入上階段讀到的卡片數值: 透過 define 的方式做宣告

```
/* Set your new UID here! */
#define NEW_UID {0xB5, 0x2C, 0x8F, 0x42}
```

Fig. 10, 即將寫入新卡的數值

ii. 初始設定: 跟讀取的部分相去不大

```
void setup() {
  Serial.begin(9600);
  while (!Serial);
  SPI.begin();         // Init SPI bus
  mfrc522.PCD_Init();  // Init MFRC522 card
  Serial.println(F("Warning: this example overwrites the UID of your UID changeable card, use with care!"));
}
```

Fig. 11, 初始設定

iii. loop 範圍內容:

- 如果一直有同張卡放置的話就會進到這邊進行 delay，以免程式不斷寫入

```
// Reset the loop if no new card present on the sensor/reader.
// if present, select one.
if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
  delay(50);
  return;
}
```

Fig. 12, 暫停寫入 delay

- 把原卡的 UID 印出，以免之後後悔想要寫回來的時候沒有得循回

```
// Now a card is selected. The UID and SAK is in mfrc522.uid.
// Dump UID
Serial.print(F("Card UID:"));
for (byte i = 0; i < mfrc522.uid.size; i++) {
  Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
  Serial.print(mfrc522.uid.uidByte[i], HEX);
}
Serial.println();
```

Fig. 13, 先印出原本的卡片 UID

- 接下來才真正是寫入的部分，在 SetUid 的函式內部其實會驗證這張卡片的授權度，如果是不可寫入的卡片使用會返回 false，所以在 true 的情況才算真正寫入成功

```
// Set new UID
byte newUid[] = NEW_UID;
if ( mfrc522.MIFARE_SetUid(newUid, (byte)4, true) ) {
  Serial.println(F("Wrote new UID to card."));
}
```

Fig. 14, 寫入新的數值

- 寫完尚未結束，還需要先停止動作，使用 HaltA 函數做停止，並且重新讀取卡片數值，如果有錯誤發生在這邊就會 return 回去了

```
// Halt PICC and re-select it so DumpToSerial doesn't get confused
mfrc522.PICC_HaltA();
if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
  return;
}
```

Fig. 15, 資料重整

- 最後會把新的卡片資訊透過 DumpToSerial 函示顯示出來，這個函式庫會把全部有經過授權的資料印出來，因為真正 rfid 晶片資料 UID 只是其中一小部分所以透過這個函式可以看到整個資料呈現

```
// Dump the new memory contents
Serial.println(F("New UID and contents:"));
mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
```

Fig. 16, 最後整個 sector 呈現

```
Warning: this example overwrites the UID of your UID changeable card, use with care!
Card UID: B5 2C 8F 42
Wrote new UID to card.
New UID and contents:
Card UID: D5 F0 58 26
Card SAK: 08
PICC type: MIFARE 1KB
```

Sector	Block	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	AccessBits
15	63	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[0 0 1]
	62	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	61	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
14	59	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[0 0 1]
	58	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	57	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	56	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
13	55	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[0 0 1]
	54	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	53	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	52	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
12	51	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[0 0 1]
	50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	49	MIFARE_Read() failed: Timeout in communication.																
	48	MIFARE_Read() failed: Timeout in communication.																
11	47	PCD_Authenticate() failed: Timeout in communication.																
10	43	PCD_Authenticate() failed: Timeout in communication.																
9	39	PCD_Authenticate() failed: Timeout in communication.																
8	35	PCD_Authenticate() failed: Timeout in communication.																
7	31	PCD_Authenticate() failed: Timeout in communication.																
6	27	PCD_Authenticate() failed: Timeout in communication.																
5	23	PCD_Authenticate() failed: Timeout in communication.																
4	19	PCD_Authenticate() failed: Timeout in communication.																
3	15	PCD_Authenticate() failed: Timeout in communication.																
2	11	PCD_Authenticate() failed: Timeout in communication.																
1	7	PCD_Authenticate() failed: Timeout in communication.																
0	3	PCD_Authenticate() failed: Timeout in communication.																

Fig. 17, 視聽器最後 output

七、實作影片:

八、參考資料: 1. Arduino: https://github.com/miguelbalboa/rfid

2. PiCamera: https://picamera.readthedocs.io/en/release-1.13/