

112 Annual Report
Institute of Space and Plasma Sciences
National Cheng Kung University

專題生：林松逸

指導教授：張博宇 博士

日期：2023/01

摘要

本次報告我主要完成了相機同步觸發的系統。由於實驗時產生的電磁脈衝雜訊，會使相機無法正常運作，所以為了降低電磁脈衝雜訊對相機的干擾，實驗室中的相機都分別放在法拉第籠中，並且使用 Raspberry Pi 透過網路來控制。為了可以更方便且快速的控制相機，我修改了控制相機的程式，並使用光訊號來觸發相機。我製作了許多的轉換器，將訊號產生器的電訊號轉為光訊號，藉由塑膠製的光纖傳輸，送到法拉第籠內部，再將光訊號轉為電訊號後傳送給 Raspberry Pi，再由 Raspberry Pi 控制相機做出拍攝的動作。

目錄

1. Arduino 練習
2. 同步相機
 - 2.1. 相機的控制程式
 - 2.2. 訊號產生器
 - 2.3. E/O converter
 - 2.4. O/E converter
 - 2.5. 光纖
 - 2.6. 杜邦線
3. 未來工作
4. 總結
5. Reference
6. 附錄

1. Arduino 練習

進入實驗室，我從練習 Arduino 開始，我是用老師推薦的書籍進行練習，書名為《超圖解 Arduino 互動設計入門》。這本書一開始是在介紹 Arduino 還有一些電子零件，介紹完後就開始編輯程式代碼的教學。我練習到第五章，在第三章到第五章的範例中，我學習了很多 Arduino 的基礎，例如消除開關彈跳訊號、製作跑馬燈等。以下三個是我做的範例。

1. 跑馬燈(1): 利用陣列的方式，從陣列第一個數值指定的腳位開始亮燈，等待 100ms 後熄燈，一直到陣列最後一個，再從最後到第一，周而復始，形成簡易跑馬燈。

```
sketch_sep28a  
  
const byte LEDs[] = {8,9,10,11,12};  
const byte total = sizeof(LEDs);  
  
void setup() {  
  // put your setup code here, to run once:  
  for (byte i=0; i<total; i++) {  
    pinMode(LEDs[i],OUTPUT);  
    digitalWrite(LEDs[i],LOW);  
  }  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  for (byte i=0; i<total-1; i++) {  
    digitalWrite(LEDs[i],HIGH);  
    delay(100);  
    digitalWrite(LEDs[i],LOW);  
  }  
  for (byte i=total-1; i>0; i-=1) {  
    digitalWrite(LEDs[i],HIGH);  
    delay(100);  
    digitalWrite(LEDs[i],LOW);  
  }  
}
```

宣告陣列
儲存陣列數量
i 小於 total 則加一
設定陣列所有腳位為輸出
依序將腳位輸出低電位
將腳位輸出高電位
等待 100 毫秒
將腳位輸出低電位

圖一、跑馬燈(1)

2. 跑馬燈(2): 直接設定每個腳位的 LED 燈的狀態，將每個燈泡依序亮起。

```
const byte LED1 = 8;
const byte LED2 = 9;
const byte LED3 = 10;

void setup() {
  // put your setup code here, to run once:
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(LED1, HIGH);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
  delay(200);
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, HIGH);
  digitalWrite(LED3, LOW);
  delay(200);
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, HIGH);
  delay(200);
}
```

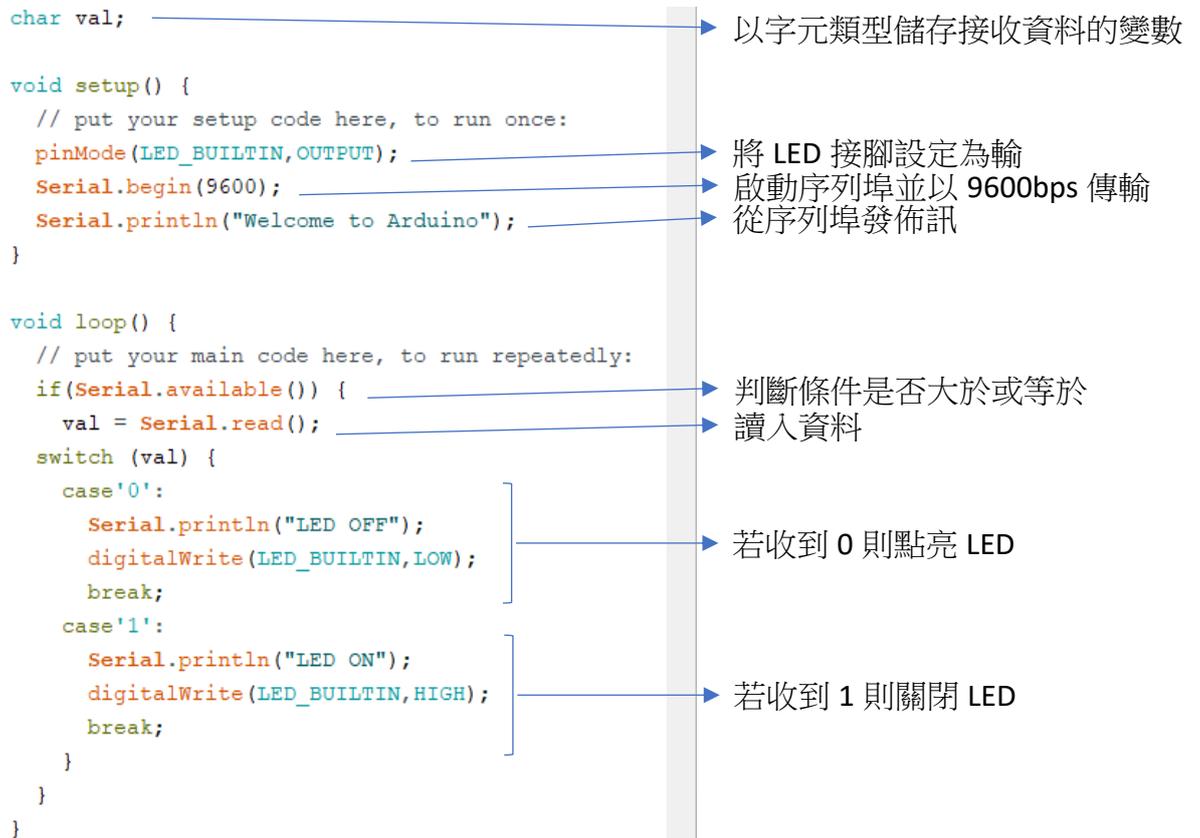
儲存腳位

設定腳位為輸出

輪流亮起，從 LED1 到 LED3

圖二、跑馬燈(2)

3. 透過序列埠控制 LED: 在序列埠中輸入特定字元來控制 LED，若收到 0 則點亮 LED 燈，若收到 1 則關閉 LED 燈。

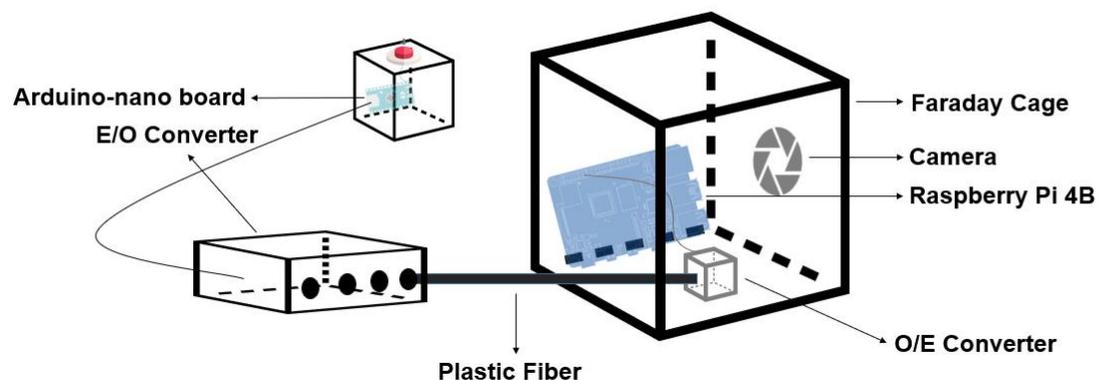


圖三、從序列埠控制 LED

2. 系統相機同步

在相機同步系統完成之前，實驗中進行拍攝時，需要手動一個一個的開啟每一台相機，第一台相機與最後一台相機開啟的時間會有很大的延遲(延遲時間約為五秒)。為了縮小相機啟動的延遲，所以我想透過觸發訊號將所有相機進行同步。

當脈衝功率系統放電時，會產生電磁脈衝雜訊，而電磁脈衝雜訊會干擾相機的運作，因此相機都被放置在鋁製的法拉第籠之中。相機同步之後，觸發信號會通過光纖穿過法拉第籠，傳送到相機。因此我利用電光轉換器(Electrical to optical converter)、光纖、光電轉換器(Optical to electrical converter) 和 Raspberry Pi 板控制相機。



圖四、同步相機系統構造

將相機系統同步之後，不僅可以避免相機在實驗中被電磁脈衝破壞，還可以使相機分別啟動的延遲時間減少，並更加方便啟動相機。以下是我如何實驗相機系統的同步。

2.1. 相機的控制程式

如果想要利用訊號來控制相機的話，必須將原本的程式進行修改。原本的流程是手動執行程式後，就會開始設定相機，然後等待 60 秒。完成相機設定，最後開始曝光 15 秒，而這個步驟要執行六次，才能使六台相機開始曝光。我希望透過訊號來觸發相機，因此我利用 python 的 RPi.GPIO 組件，如圖五所示，將 RPi.GPIO 導入後，使用 setmode 函數對“編號系統”進行設定，使用 BOARD 模式編號，因為這種模式的編號方式與 Raspberry Pi 腳位的編號一樣。如此以來，未來在接線的時候，任何編號都可以直接對應到 Raspberry Pi 板上的腳位。另外，datetime 組件是用於抓執行程式的時間，例如我在 2022/12/31 23:59.59 執行的話，那程式就會記錄下這個時間點，用來設定儲存檔案的檔名。最後 PiCamera 則是用來控制相機動作的，例如旋轉畫面、啟動相機等等。

```
from datetime import datetime
from fractions import Fraction
import time as sleep
import RPi.GPIO as GPIO
from picamera import PiCamera
```

圖五、Raspberry Pi 相機的初始設定

設置完編號模式後，如圖六所示，我使用 setup 函數來分配引腳輸入或輸出。我設定第七腳位做為輸入，以第七腳位來接收從訊號產生器生的訊號。

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7,GPIO.IN)
```

圖六、引腳配置

接下來利用原本的程式去設定相機的參數，旋轉角度等等，如圖七所示。

```
t = 1500
tt = t+100
extime = t*1000
iso = 1

camera = PiCamera()
camera.resolution = (4056,3040)
camera.rotation = 180
camera.framerate = Fraction(1000,t)
camera.shutter_speed = extime
camera.iso = iso
```

圖七、設定相機

在設定好相機之後，進入到迴圈之內，目的是為了偵測訊號。當接收到訊號後，就會開啟相機曝光，並離開迴圈。為防止當實驗出狀況或是設備異常時，我在迴圈中加入了超過一定重複次數(10000 圈)之後，如果沒收到訊號那就會將程式 break，防止程式進入無限迴圈，導致電腦當機，如圖八所示。

```
for i in range(10002): #must > first if value
    result = GPIO.input(7)

    if i > 10000: #10000 approximate 22 sec
        print('too long')
        break
```

圖八、等待迴圈

如圖九，當收到來自訊號產生器的訊號後，程式就會控制相機開始曝光。完成拍攝之後則會將照片進行儲存。最後一步就是將相機關閉，並顯示拍攝時間，最後離開迴圈。

```
if result == 1:
    print('success')
    camera.exposure_mode = 'off'
    print("Camera ready")

    now = datetime.now()
    current_time = now.strftime("%y-%m-%d_%H-%M-%S_")
    filepath,filename,filetype,ext,ISO,cameraname = '/home/pi/Desktop/Soft/',current_time,'.bmp',str(t),str(ISO),'laser2_'
    file = filepath+filename+cameraname+ext+'_'+ISO+filetype
    print(1,current_time)

    camera.capture(file,format='bmp')
    print(camera.exposure_speed)
    print(camera.shutter_speed)

    camera.close()

    now = datetime.now()
    current_time = now.strftime("%y.%m.%d-%H:%M:%S")
    print(2,current_time)
    break
```

圖九、相機曝光與儲存

完整的程式碼請見附錄一。

2.2. 訊號產生器

我利用林彥呈學長製作的訊號產生器[1]，當作我用來遠程控制相機的訊號產生器。不過為了讓相機系統可以偵測到訊號，我將學長原本設定的 1.25 毫秒改為 0.5 秒，如下頁圖十所示，以確保不會發生按下按鈕後，Raspberry Pi 沒收到訊號的狀況。

```

int TLED = 8;      trigger LED 腳位
int SLED = 7;      stand by LED 腳位
int t0 = 0;        trigger LED 初始值
int s0 = 0;        stand by LED 初始值
int t1 = 0;        trigger 前一時間的值
int s1 = 1;        stand by 前一時間的值
int t2;           trigger 當前時間的值
int s2;           stand by 當前時間的值
int triggerpin = 2; trigger 腳位
int standbypin = 4; stand by 腳位
int st;           s2-s1
int tt;           t2-t1

void setup()      腳位等系統設定
{
  pinMode(TLED, OUTPUT);      trigger LED 腳位模式設定
  pinMode(SLED, OUTPUT);      stand by LED 腳位模式設定
  pinMode(triggerpin, INPUT); trigger 讀取腳位模式設定
  pinMode(standbypin, INPUT); stand by 讀取腳位模式設定
  Serial.begin(9600);         設定通訊速率
}

void loop()       主迴圈
{
  t2 = digitalRead(triggerpin); 讀取開關狀態(斷路為 0，通路為 1)
  s2 = digitalRead(standbypin);
  st = s2-s1;                判斷是否有扳動開關或按下按鈕
  tt = t2-t1;

  if (s0==0)                s0=1 才能送訊號
  {
    if(st==1)                此判斷式表示開關撥桿從 off 到 on
    {
      s0 = 1;                這裡是唯一有機會將 s0 變為 1
      s1 = s2;                將 stand by 前一時間值=現在值
      digitalWrite(SLED, s0); stand by LED 亮
    }
    else
    {
      s1 = s2;                將 stand by 前一時間值=現在值
    }
  }

  else
  {
    if (st==1)                此判斷式表示開關撥桿從 on 到 off
    {
      s0=0;
      digitalWrite(SLED, s0); stand by LED 暗
      s1 = s2;                將 stand by 前一時間值=現在值
    }
    elseif (tt==1)            此判斷式表示觸發按鈕按下
    {
      t0=1;
      digitalWrite(TLED, t0); trigger LED 亮
      delayMicroseconds(1250); 亮 1.25ms → 將 1.25ms 改為 0.5s
      digitalWrite(TLED, 0);   trigger LED 暗
      t1 = t2;                將 trigger 前一時間值=現在值
      s0 = 0;
      digitalWrite(SLED, s0); stand by LED 暗
    }
    else
    {
      t1 = t2;                將 trigger 前一時間值=現在值
    }
  }
}

```

圖十、訊號產生器的程式碼[1]

下圖左邊包含了 Arduino-nano board、SN75451BN、transmitter 還有三個 LED 燈，而右邊則是啟動按鈕，設計圖請參見參考資料[1]。

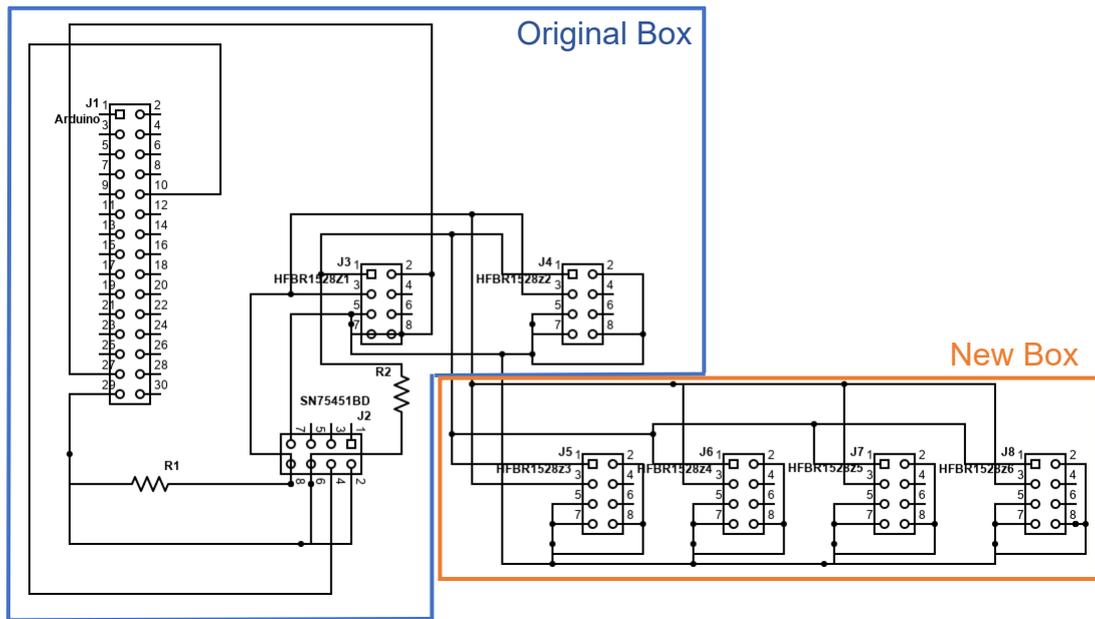


圖十一、訊號產生器外型

2.3. E/O converter

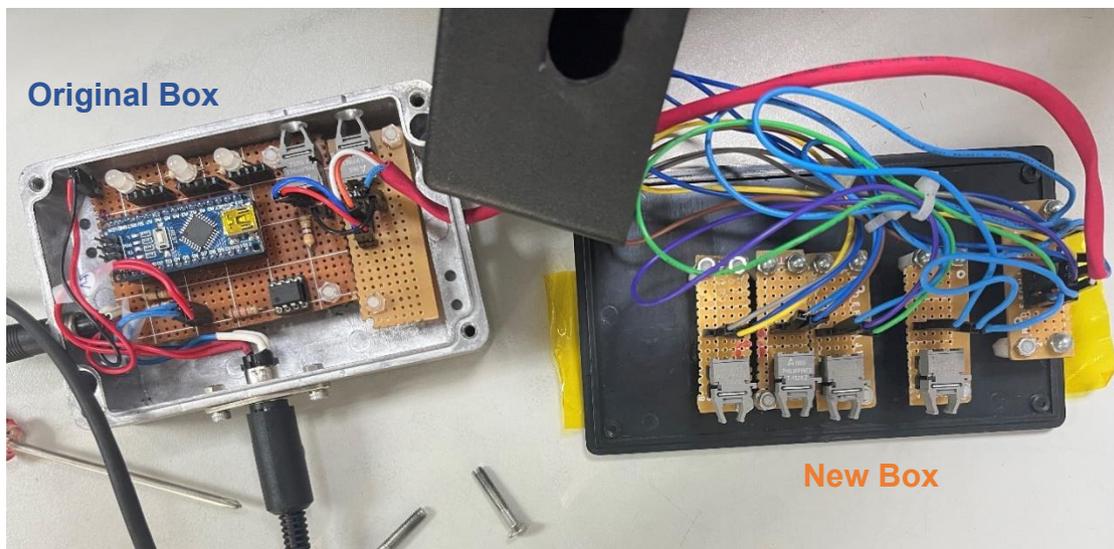
製作 E/O converter 的目的是為了讓訊號可以經由塑膠製的光纖傳輸，利用塑膠製的光纖傳輸訊號的好處是電磁脈衝雜訊不會跟著光纖進入到相機內，造成相機無法正常運作。

因為實驗室有六台相機，所以我除了將發出訊號的時間改變以外，我在 transmitter 的部分又新增了四個(圖十二中 New Box 框框)，加上原本輸出的兩個訊號(圖十二中 Original Box 框框)，可以對六台相機同時發出訊號。原本以為林彥呈學長製作的訊號產生器無法同時推動六台 transmitter，但是完成以後發現，東西都能正常運作，所以就沒有再增加 SN75451BN 來放大輸出的訊號。E/O converter 的電路圖如圖十二。



圖十二、E/O converter 電路圖

最後將新增的四個 transmitter 固定在新的盒子裡面，並將盒子挖孔，就完成了 E/O converter 的製作。

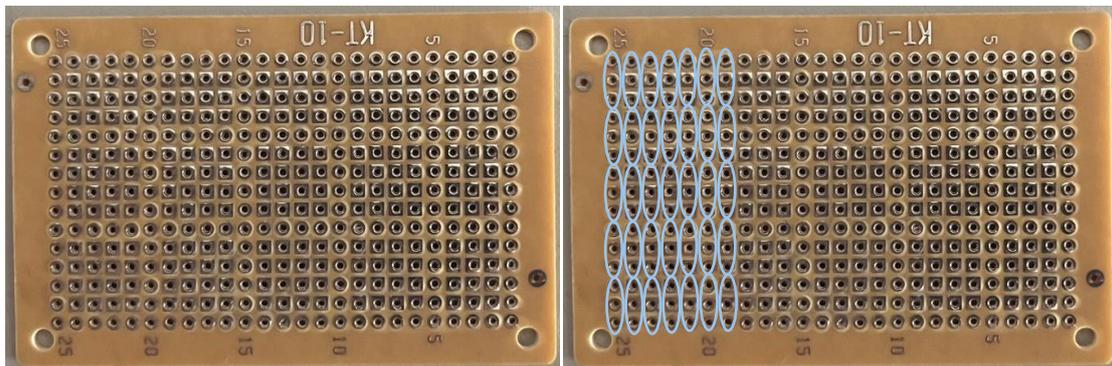


圖十三、E/O converter

2.4. O/E converter

製作 O/E converter 的目的是將光訊號轉為電訊號，才能將法拉第籠外部藉由光纖傳入的訊號再傳給 Raspberry Pi。

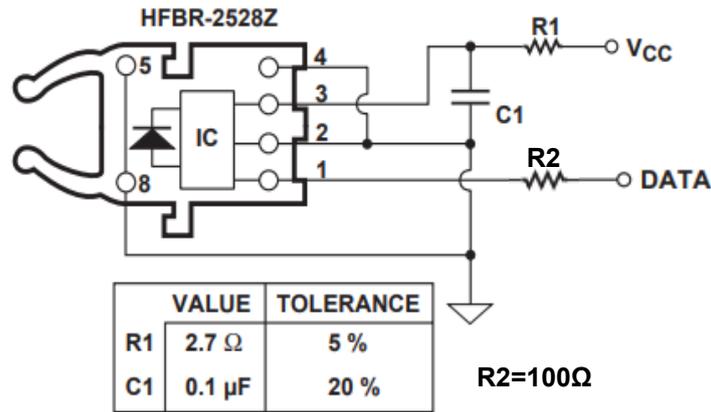
O/E converter 是我在實驗室做的第一個物件，總共製作了六個。在製作的過程中我遇到了許多問題，從一開始因為是第一次焊接電子元件，所以要請教學長外，遇到最大的問題就是剛做完時，電阻會一直燒掉。老師教了我除錯的方法，使用三用電表來確認是否有焊接好電路，學長也幫我檢查電路有無問題，最後在老師的幫助下才發現原來是使用錯電路板的問題。應該使用一般的板子(圖十四)，但是我用成有串聯起來的(圖十五為示意圖)，由於圖十五電路板部分地方是串聯，會造成有些不該導通的地方導通，使電路失效。



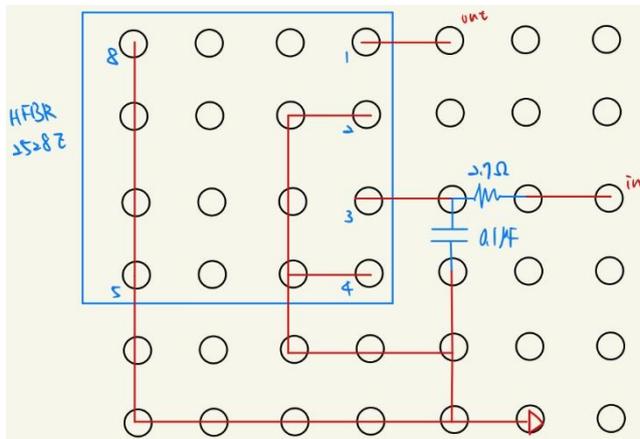
圖十四、一般洞洞板

圖十五、有串聯之洞洞板

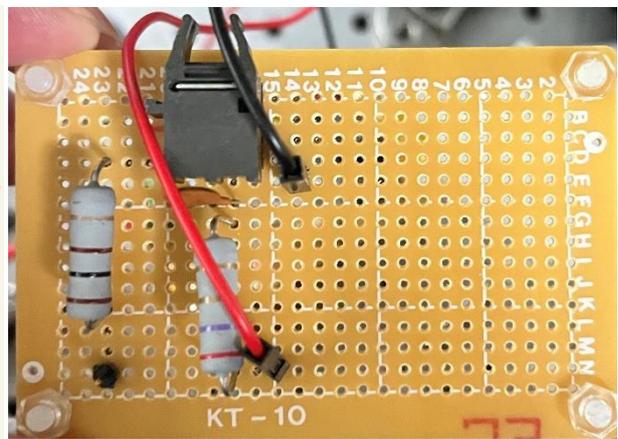
圖十六到十八分別是我的電路、layout 圖與成品。總共製作了六個，每個 O/E converter 都必須分別放在每個相機的法拉第籠中。



圖十六、電路圖[2]



圖十七、Layout 圖



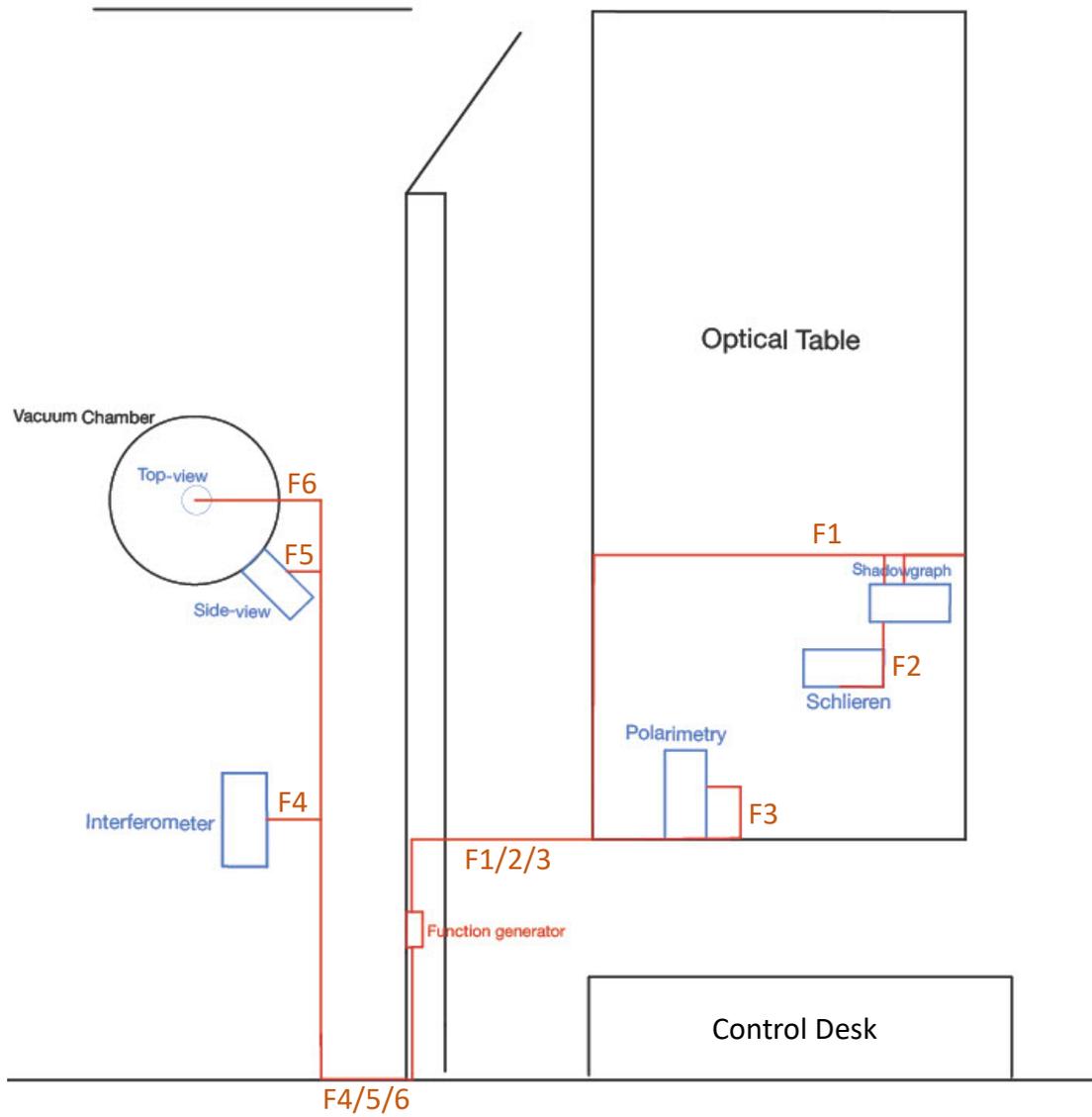
圖十八、成品

2.5. 光纖

E/O converter 與 O/E converter 之間是用塑膠製的光纖連接，在完成 E/O converter 與 O/E converter 之後，下一項任務就是用光纖將其連接起來，為了避免光纖擋到量測儀器中雷射的光路、妨礙到人員的行動，所以我將光纖靠著牆壁走，並且利用束帶固定，然後用標籤將每一條光纖命名。以下是實驗室中光纖的路徑。

| | |
|----|----------------|
| F1 | Shadowgraph |
| F2 | Schlieren |
| F3 | Polarimetry |
| F4 | Interferometer |
| F5 | Side-view |
| F6 | Top-view |

表一



圖十八、光纖路徑

2.6. 杜邦線

杜邦線除了可以用於 E/O converter 與 O/E converter 內部連接外，還可以用於將 O/E converter 的電訊號傳入 Raspberry Pi。以下為製作過程。

1 準備工具

- 杜邦端子
- 杜邦接頭
- 尖嘴鉗
- 壓線鉗
- 剝線鉗

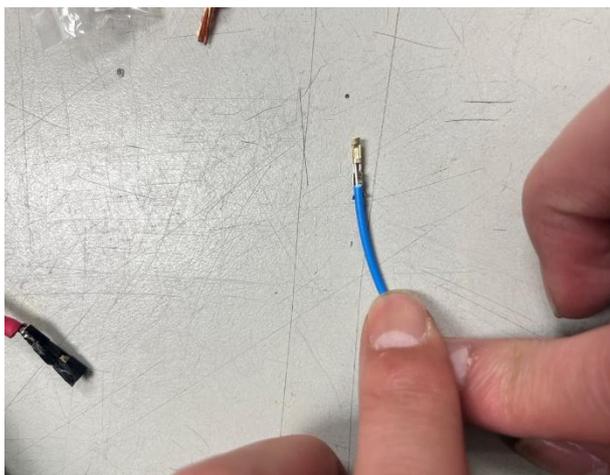
2 步驟

(1) 用剝線鉗將線剝除約 5mm 外皮。



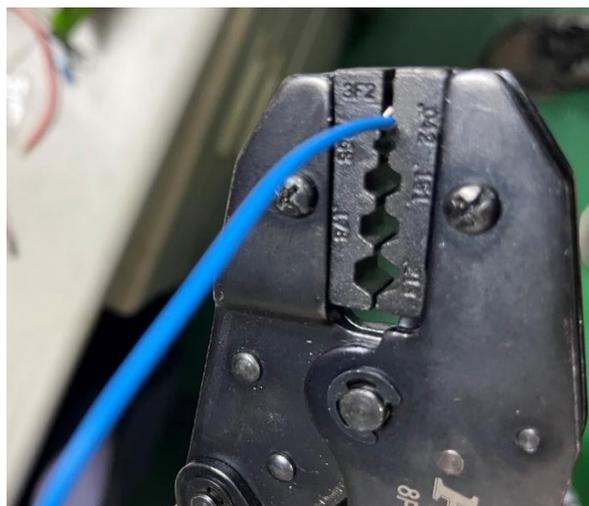
圖十九、步驟 2(1)

(2) 套上杜邦端子。



圖二十、步驟 2(2)

(3) 用壓線鉗將杜邦端子夾緊。

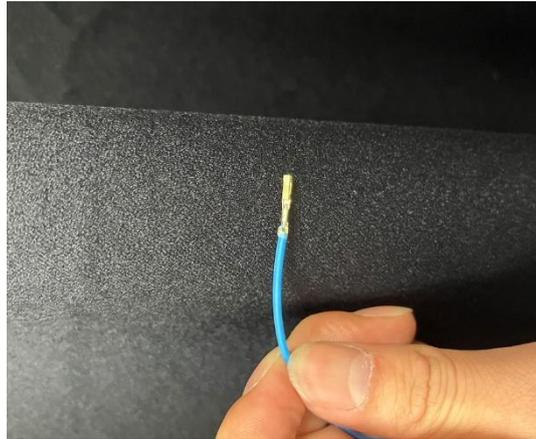


圖二十一、步驟 2(3)



圖二十二、步驟 2(3)結果

(4) 用尖嘴鉗將杜邦端子再次壓緊。



圖二十三、步驟 2(4)

(5) 最後套上杜邦接頭。



圖二十四、步驟 2(5)

(6) 另一端重複步驟 2.1~2.5，就完成一條杜邦線了。



圖二十五、杜邦線完成圖

3. 未來工作

- A. 設計一個架子，用於架高 E/O converter 與 Function generator。
- B. 學習推進的相關知識。
- C. 開始研究推進系統。

4. 總結

同步相機已經完成，之後實驗時拍攝就可以透過單個按鈕啟動所有相機，不需要再用手動一台一台啟動。

5. Reference

[1]https://capst.ncku.edu.tw/PGS/Student_Annual_Report/20200520_Annual_ProgressReport_%E6%9E%97%E5%BD%A5%E5%91%88_Final.pdf

[2]https://www.digikey.com/en/products/detail/HFBR-2528Z/516-2066-ND/1990464?utm_source=findchips&utm_medium=aggregator&WT.z_cid=ref_findchips_standard&utm_campaign=buynow

6. 附錄

附錄一、

```
from datetime import datetime
from fractions import Fraction
import time as sleep
import RPi.GPIO as GPIO
from picamera import PiCamera

GPIO.setmode(GPIO.BOARD)
GPIO.setup(7,GPIO.IN)

t = 1500
tt = t+100
extime = t*1000
iso = 1

camera = PiCamera()
camera.resolution = (4056,3040)
camera.rotation = 180
camera.framerate = Fraction(1000,t)
camera.shutter_speed = extime
camera.iso = iso
result = GPIO.input(7)
print(result)
```

(接下頁)

```

for i in range(10002): #must > first if value
    result = GPIO.input(7)

    if i > 10000: #10000 approximate 22 sec
        print('too long')
        break
    if result == 1:
        print('success')
        camera.exposure_mode = 'off'
        print("Camera ready")

        now = datetime.now()
        current_time = now.strftime("%y-%m-%d_%H-%M-%S_")
        filepath,filename,filetype,ext,ISO,cameraname =
'/home/pi/Desktop/Soft/',current_time,'.bmp',str(t),str(ISO),'laser2_'
        file = filepath+filename+cameraname+ext+'_'+ISO+filetype
        print(1,current_time)

        camera.capture(file,format='bmp')
        print(camera.exposure_speed)
        print(camera.shutter_speed)

        camera.close()

        now = datetime.now()
        current_time = now.strftime("%y.%m.%d-%H:%M:%S")
        print(2,current_time)
        break
print(i)

```