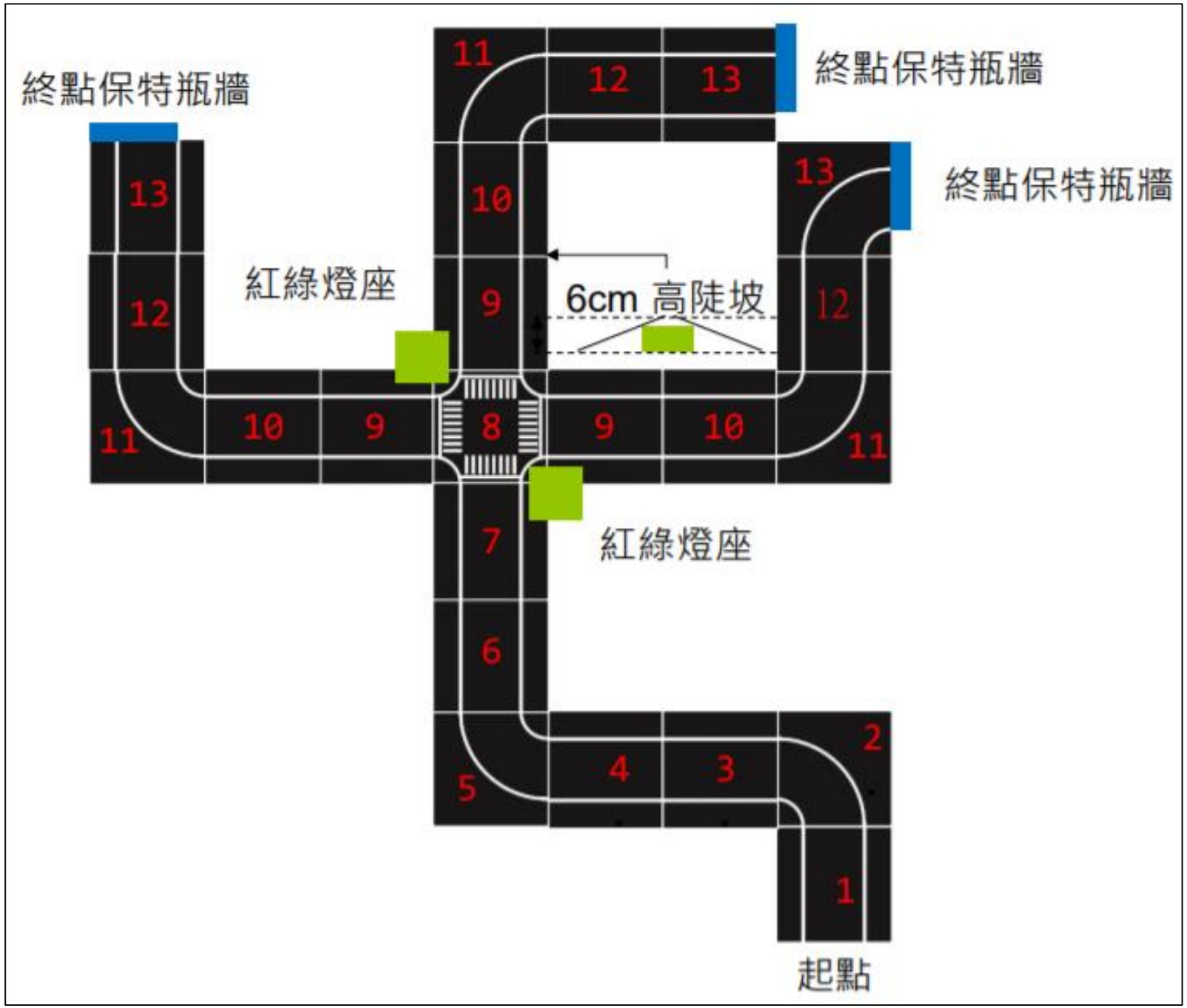
比賽場地


終點保特瓶牆
終點保特瓶牆
終點保特瓶牆
紅綠燈座
6cm 高陡坡
紅綠燈座
起點
11
12
13
13
10
9
12
11
10
9
8
9
10
11
7
6
5
4
3
2
1

競賽分析

高中組 & 大專組

1. 從第1塊場地出發，依序走至十字路口
2. 紅燈時車子須停止，綠燈時需要選擇『直走』or『左轉』or『右轉』
3. 過十字路口後，依序走到終點，終點會有寶特瓶來讓車子偵測並停止

大專組

4. 於終點停止3秒後，須將車子掉頭並依原路線行駛回起點

前3步驟 LAB26程式碼即可，步驟4需增加程式碼

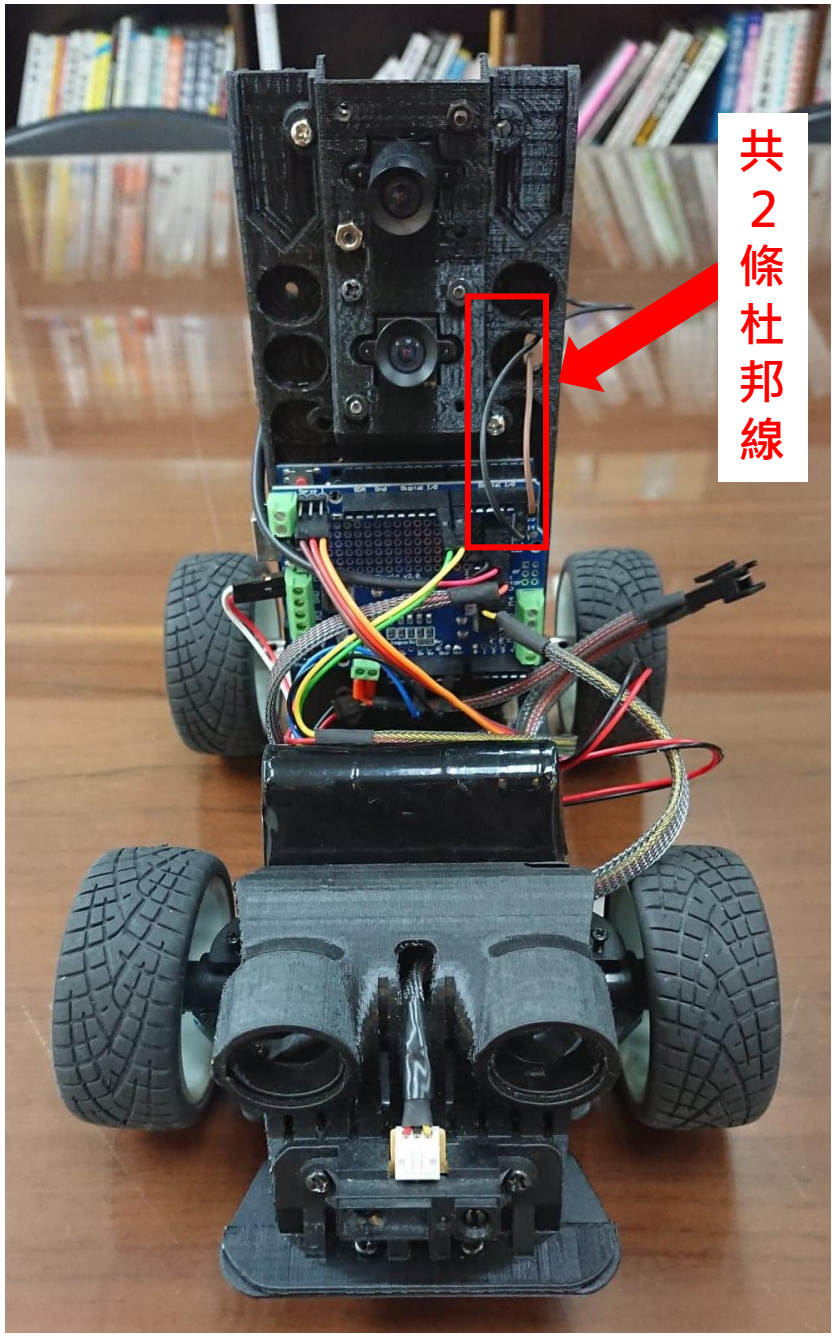
資料雙向傳輸 (步驟4需要)

LAB-14提過Serial能夠做出**雙向**的通訊，但前面的LAB只需要Orange Pi傳送資料給 Arduino，所以1條杜邦線即可。

大專組的比賽項目中，車子在遇到終點障礙物後需要返回起點，所以這時會讓Arduino告訴Orange pi已經遇到終點，該返回了。

!!!因此需要再接1條杜邦線來讓Arduino傳送訊息給Orange Pi !!!

※ 杜邦線可以使用**組裝紅綠燈**時留下的杜邦線

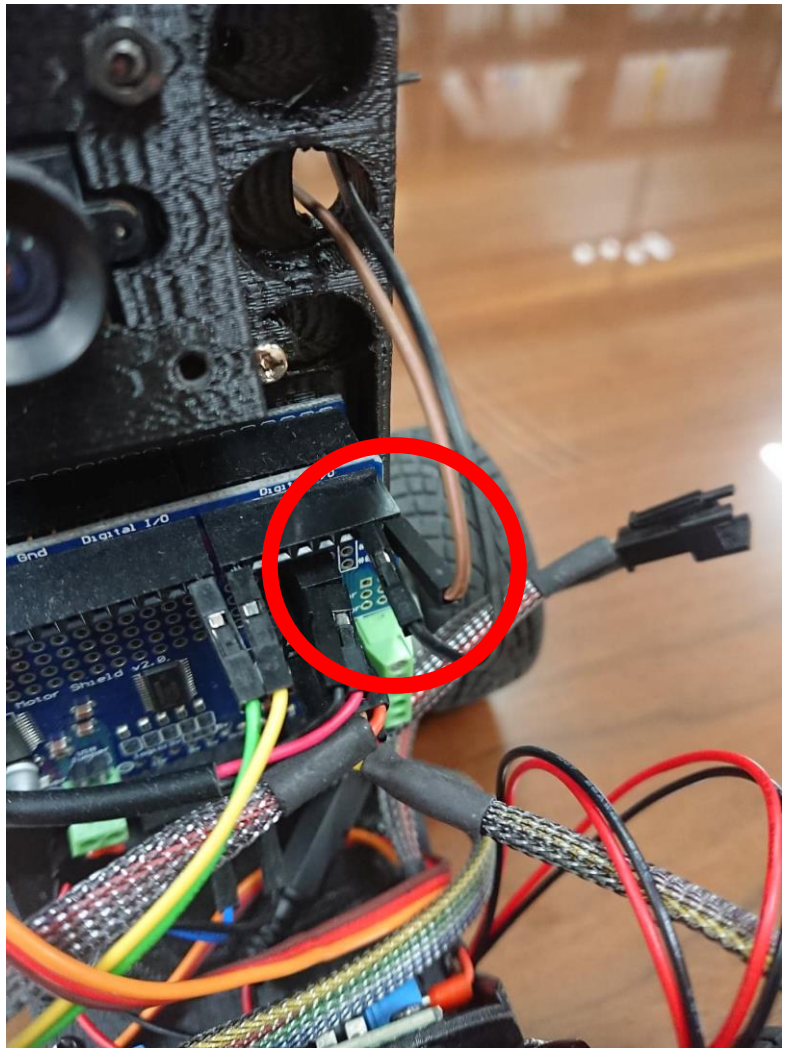


接腳位置

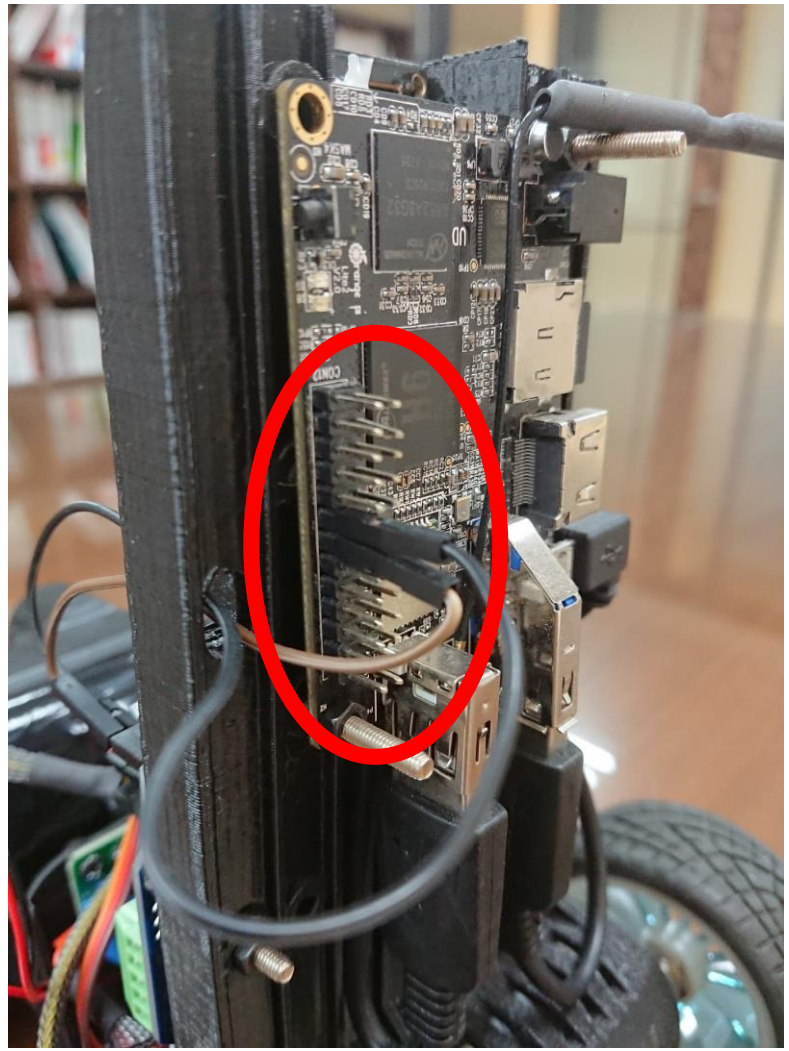
發射腳位(TX)、接收腳位(RX)

Orange Pi – 左上數來第六根針(RX)

Arduino – 1號腳位(TX)



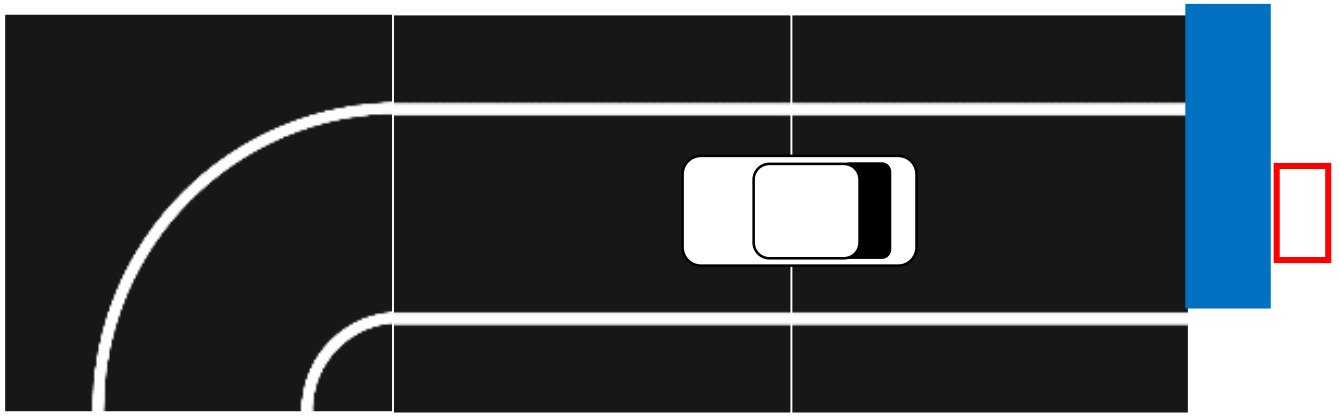
【Arduino】



【 Orange Pi 】

步驟 4 解析

1. 遇到終點進入**倒車模式**
2. 左轉或右轉倒退
3. 看到**直線**時輪胎打直繼續後退
4. 當直線離開視野後，開始右轉或左轉前進
5. 開始尋找車道線
6. 回歸正常尋軌模式



Arduino程式碼

Arduino範例程式：University

```
209     if(intersection==1 && car_stop==0 && millis()-stoptime>=3000){    //已經經過路口 且 測距而停止的時間大於3秒
210       Serial.println("z");                  //告訴單板電腦暫時不用看車道
211       car_stop=1;                           //變回開車狀態
212       if(last_turn=='f' || last_turn=='g'){
213         Forward(Speed_pwm);                 //先前進一小段
214         delay(300);                         //前進時間
215         servo1.write(right);
216         delay(100);
217         }
218       if(last_turn=='e'){
219         servo1.write(left);
220         delay(100);
221         }
222       go=0;                       //倒車模式
223       Rotating_speed_min=Slow_speed_min;          //更改標準:放慢
224       Rotating_speed_max=Slow_speed_max;          //更改標準:放慢
225       stop_start=0;  //停完後開始移動了
226       stoptime=millis();    //更改紀錄時間(這次是紀錄後退時間)
227       }
```

告知Orange Pi已經到達終點

Arduino範例程式：University

Python程式碼 – 接收訊息

```
242         if ser.in_waiting:                          #如果序列埠收到值
243             data_raw=ser.readline()                 #讀取序列埠值
244             data=data_raw.decode()                  #解碼
245             print('original_data:',data_raw)
246             print('new_data',data)
247             if(data=='z\r\n'):                      #如果傳過來的資料為 z  (後面都是 arduino傳送時會多加的換行字元)
248                 road_condition=road_condition_d['back']     #路況改為倒退
```

如果接收到 z ，設為倒車狀態

Python程式碼 – 倒車狀態

```
472     #當倒退中第一次看到車道線，要將輪胎打直----------------------------------------------------------------------------
473         if(road_condition==road_condition_d['back'] and len(crop_histogram)>=no_zeros_axis0>=len(crop_histogram)-15):
474             road_condition=road_condition_d['back_go_1']      #路況更改為後退並將輪胎打直
475             ser.write(b'u')    #告知將輪胎打直
```

看到直線，輪胎打直繼續後退

```
476     #當倒退中第二次看到車道線，要停下來並前進----------------------------------------------------------------------------
477         if(road_condition==road_condition_d['back_go_1'] and no_zeros_axis0<=10):
478             road_condition=road_condition_d['back_go_2']      #路況更改為後退完並前進
479             ser.write(b't')    #告知不繼續後退並開始前進
```

直線消失，開始往前轉彎

Python程式碼 – 尋找車道

```
#當 路況為後退完並前進時 & 兩車道線的頂點皆正確 & 最多點那條需要有 100~120個白點
if(road_condition==road_condition_d['back_go_2']  and log_lineLeft['x'][-1]==0 and log_lineRight['x'][-1]==0 and
  log_lineLeft['x'][0]==(original_img.shape[0])*(sliding_window_specs['n_steps']-1)/sliding_window_specs['n_steps'] and
  log_lineRight['x'][0]==(original_img.shape[0])*(sliding_window_specs['n_steps']-1)/sliding_window_specs['n_steps'] and
    100<=max_high<=120):

    road_condition=road_condition_d['normal']      #路況更改為正常
    status=status_d['normal']                      #狀態更改為正常
```

重新找到兩條車道線，找尋車道的方式與離開十字路口(LAB21)相似

變回正常尋軌模式

此範例執行結果

車子走到終點後，掉頭回到起點