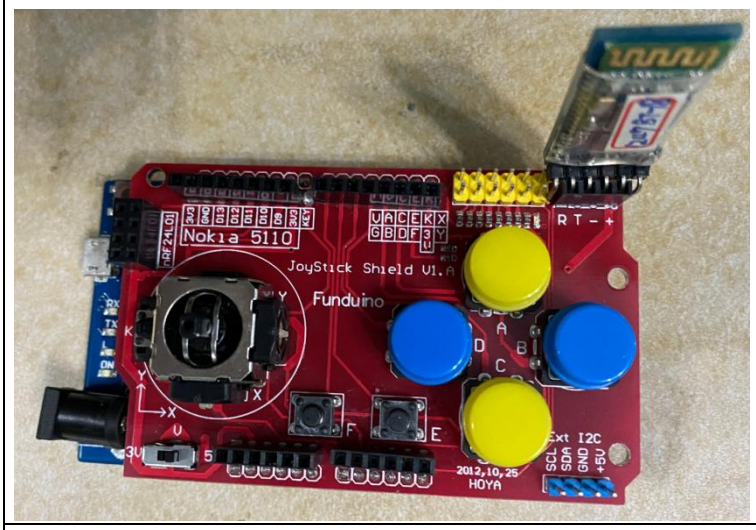

搖桿程式碼	
#include <SoftwareSerial.h>　// 引用程式庫	
int SW1 = 2;	伺服馬達 A 上：按鍵 A
int SW2 = 3;	伺服馬達 B 右：按鍵 B
int SW3 = 4;	伺服馬達 A 下：按鍵 C
int SW4 = 5;	伺服馬達 B 左：按鍵 D
int MotorA_DelayTime = 300, MotorB_DelayTime = 300, ServoA_DelayTime = 300, ServoB_DelayTime = 300; //延遲時間變數	設定馬達被按下的延遲時間(避免同時執行)
int ServoA_Up, ServoA_Down, ServoB_Right, ServoB_Left; //按鍵按壓讀取訊號變數	設定腳位讀取類比訊號
int Motor_Action_Joy_Value = 0; //搖桿方向變數	設定搖桿方向變數
int vrx, vry, x_axis, y_axis; //搖桿讀取訊號變數	設定搖桿讀取類比訊號
int MotorA_Forward_Flag = 0, MotorA_Backward_Flag = 0, MotorA_MoveStop_Flag = 0; //MotorA 前進、後退、停止旗標變數	設定直流馬達 A 正反轉及停止的執行變數
int MotorB_Forward_Flag = 0, MotorB_Backward_Flag = 0, MotorB_MoveStop_Flag = 0; //MotorB 前進、後退、停止旗標變數	設定直流馬達 B 正反轉及停止的執行變數
int ServoA_Up_Flag = 0, ServoA_Down_Flag = 0, ServoA_MoveStop_Flag = 0; //ServoA 上、下、停止旗標變數	設定伺服馬達 A 正反轉及停止的執行變數
int ServoB_Right_Flag = 0, ServoB_Left_Flag = 0, ServoB_MoveStop_Flag = 0; //ServoB 左、右、停止旗標變數	設定伺服馬達 B 正反轉及停止的執行變數
int JoyStop_Status = 0; //搖桿停止變數	搖桿沒被按壓時的執行變數
int ServoA_Up_Status = 0, ServoA_Down_Status = 0, ServoB_Right_Status = 0,	右側四顆按鈕被按壓狀態

ServoB_Left_Status = 0; //ServoA、ServoB 按鍵按壓狀態變數	變數
int ServoA_Up_Press_Flag = 0, ServoA_Down_Press_Flag = 0, ServoB_Right_Press_Flag = 0, ServoB_Left_Press_Flag = 0; //ServoA、ServoB 按鍵按壓旗標變數	按鈕按壓後，伺服馬達的執行變數，變數 A 上下；變數 B 左右
int MotorA_Status = 0, MotorB_Status = 0; //MotorA、MotorB 狀態變數	左側搖桿被推移的的狀態變數
unsigned long currentTime = 0; //時間變數	搖桿開啟時的時間點 0
unsigned long MotorA_Time = 0, MotorB_Time = 0; //MotorA、MotorB 時間變數	直流馬達 AB 的執行時間
unsigned long ServoA_Up_Press_Time = 0, ServoA_Up_Release_Time = 0, ServoA_Down_Press_Time = 0, ServoA_Down_Release_Time = 0; //ServoA 按壓按鍵、釋放按鍵時間變數	伺服馬達 A 上下的按壓時間點與釋放時間點
unsigned long ServoB_Right_Press_Time = 0, ServoB_Right_Release_Time = 0, ServoB_Left_Press_Time = 0, ServoB_Left_Release_Time = 0; //ServoB 按壓按鍵、釋放按鍵時間變數	伺服馬達 B 左右的按壓時間點與釋放時間點
void setup() {	
Serial.begin(9600);	螢幕編號設定
Serial1.begin(38400);	藍牙編號設定
pinMode(SW1, INPUT);	SW1 數位腳位狀態輸入
pinMode(SW2, INPUT);	SW2 數位腳位狀態輸入
pinMode(SW3, INPUT);	SW3 數位腳位狀態輸入
pinMode(SW4, INPUT);	SW4 數位腳位狀態輸入
pinMode(SW1, INPUT_PULLUP);	SW1 腳位起始狀態 High
pinMode(SW2, INPUT_PULLUP);	SW2 腳位起始狀態 High
pinMode(SW3, INPUT_PULLUP);	SW3 腳位起始狀態 High
pinMode(SW4, INPUT_PULLUP);	SW4 腳位起始狀態 High
pinMode(A0, INPUT);	A0 類比腳位狀態輸入
pinMode(A1, INPUT);	A1 類比腳位狀態輸入
}	
void Motor_SwitchSignal_Transmitter() //按鍵藍芽傳送 function	
{	
if(ServoA_Up_Flag == 1 && ServoA_Up_Status == 0 && ServoA_Down_Press_Flag == 0 && ServoB_Right_Press_Flag == 0 && ServoB_Left_Press_Flag == 0) //確認按鍵上旗標是否動作，且各方向按壓旗標皆沒有動作	如果只有按鍵 A 被按下
{	
for(int i ; i <= 10 ; i++)	

//https://www.arduino.cc/reference/en/language/structure/control-structure/for/ - 上述網站有詳細函示解說	
{	
Serial1.write(22); //發送 ServoA 上訊號	發送訊號啟動伺服馬達 A，動作向上
Serial.println("ServoA_Up"); //傳送到監控視窗以利觀察是否進入此迴圈	
}	
ServoA_Up_Press_Time = currentTime; //記錄當下按壓"上"的時間	紀錄按下按鍵的時間點
ServoA_Up_Status = 1; //改變 ServoA 按壓"上"的狀態，避免一直發送訊號	伺服馬達 A 上執行狀態改為 1
ServoA_Up_Press_Flag = 1; //ServoA 上旗標 - 1:動作，0:不動作	
}	
if(currentTime - ServoA_Up_Press_Time >= ServoA_DelayTime) //確認按下按鍵時間是否超過延遲時間	馬達開始執行後有 0.3 秒的延遲時間
{	
ServoA_Up_Press_Flag = 0; //確認超過時間後，ServoA 上旗標 - 1:動作，0:不動作，其作用是確保兩顆馬達同時啟動時可以錯開執行	超過延遲時間才會執行其他馬達的動作
}	
if(ServoA_Up == 1 && ServoA_Up_Status == 1) //確認按鍵"上"是否被放開且按鍵"上"狀態為 1(代表有被按過)	如果按鍵 A 被放開且伺服馬達 A 上還在執行
{	
for(int i ; i <= 10 ; i++)	訊號發送 10 次
{	
Serial1.write(26); //發送 ServoA 停止訊號	發送訊號伺服馬達 A 停止
Serial.println("ServoA_Up_Stop"); //傳送到監控視窗以利觀察是否進入此迴圈	
}	
ServoA_Up_Release_Time = currentTime; //記錄當下按壓"上"被釋放的時間	紀錄按鍵被放開的時間點
ServoA_Up_Status = 2; //改變 ServoA 按壓"上"的狀態，避免一直發送訊號	伺服馬達 A 上執行狀態改為 2
}	
if(ServoA_Up_Status == 2)	如果伺服馬達 A 上執行狀態為 2
{	
if(currentTime - ServoA_Up_Release_Time >= ServoA_DelayTime) //確認按壓"上"被釋放的時間是否超過延遲時間	馬達停止後有 0.3 秒的延遲時間
{	
ServoA_Up_Status = 0; //ServoA 按壓"上"的狀態清 0，確定已經超過延遲時間，其作用主要是確保馬達轉態時會有足夠的停止時間	超過延遲時間才會執行其他馬達的動作
}	
}	
if(ServoA_Down_Flag == 1 && ServoA_Down_Status == 0 && ServoA_Up_Press_Flag == 0 && ServoB_Right_Press_Flag == 0 && ServoB_Left_Press_Flag == 0) //確認按鍵下旗標是否動作，且各方向按壓旗標皆沒有動作	如果只有按鍵 C 被按下
{	
for(int i ; i <= 10 ; i++)	

{
Serial1.write(23); //發送 ServoA 下訊號
Serial.println("ServoA_Down"); //傳送到監控視窗以利觀察是否進入此迴圈
}
ServoA_Down_Press_Time = currentTime; //記錄當下按壓"下"的時間
ServoA_Down_Status = 1; //改變 ServoA 按壓"下"的狀態，避免一直發送訊號
ServoA_Down_Press_Flag = 1; //ServoA 下旗標 - 1:動作，0:不動作
}
	發送訊號啟動伺服馬達 A，動作向下
	紀錄按下按鍵的時間點
	伺服馬達 A 下執行狀態改為 1

if(currentTime - ServoA_Down_Press_Time >= ServoA_DelayTime) //確認按下按鍵時間是否超過延遲時間
{
ServoA_Down_Press_Flag = 0; //確認超過時間後，ServoA 下旗標 - 1:動作，0:不動作，其作用是確保兩顆馬達同時啟動時可以錯開執行
}
	馬達開始執行後有 0.3 秒的延遲時間
	超過延遲時間才會執行其他馬達的動作

if(ServoA_Down == 1 && ServoA_Down_Status == 1) //確認按鍵"下"是否被放開且按鍵"下"狀態為 1(代表有被按過)
{
for(int i ; i <= 10 ; i++)
{
Serial1.write(26); //發送 ServoA 停止訊號
Serial.println("ServoA_Down_Stop"); //傳送到監控視窗以利觀察是否進入此迴圈
}
ServoA_Down_Release_Time = currentTime; //記錄當下按壓"下"被釋放的時間
ServoA_Down_Status = 2; //改變 ServoA 按壓"下"的狀態，避免一直發送訊號
}
	如果按鍵 C 被放開且伺服馬達 A 下還在執行
	訊號發送 10 次
	發送訊號伺服馬達 A 停止
	紀錄按鍵被放開的時間點
	伺服馬達 A 下執行狀態改為 2

if(ServoA_Down_Status == 2)
{
if(currentTime - ServoA_Down_Release_Time >= ServoA_DelayTime) //確認按壓"下"被釋放的時間是否超過延遲時間
{
ServoA_Down_Status = 0; //ServoA 按壓"下"的狀態清 0，確定已經超過延遲時間，其作用主要是確保馬達轉態時會有足夠的停止時間
}
}
	如果伺服馬達 A 下執行狀態為 2
	馬達停止後有 0.3 秒的延遲時間
	超過延遲時間才會執行其他馬達的動作

if(ServoB_Right_Flag == 1 && ServoB_Right_Status == 0 && ServoA_Down_Press_Flag == 0 && ServoA_Up_Press_Flag == 0 && ServoB_Left_Press_Flag == 0) //確認按鍵右旗標是否動作，且各方向按壓旗標皆沒有動作
{
for(int i ; i <= 10 ; i++)
{
Serial1.write(24); //發送 ServoB 右訊號
	如果只有按鍵 B 被按下
	訊號發送 10 次
	發送訊號啟動伺服馬達

Serial.println("ServoB_Right"); //傳送到監控視窗以利觀察是否進入此迴圈	B，動作向右
}	
ServoB_Right_Press_Time = currentTime; //記錄當下按壓"右"的時間	紀錄按下按鍵的時間點
ServoB_Right_Status = 1; //改變 ServoB 按壓"右"的狀態，避免一直發送訊號	伺服馬達 B 右執行狀態改為 1
ServoB_Right_Press_Flag = 1; //ServoB 右旗標 - 1:動作，0:不動作	
}	
if(currentTime - ServoB_Right_Press_Time >= ServoB_DelayTime) //確認按下按鍵時間是否超過延遲時間	馬達開始執行後有 0.3 秒的延遲時間
{	
ServoB_Right_Press_Flag = 0; //確認超過時間後，ServoB 右旗標 - 1:動作，0:不動作，其作用是確保兩顆馬達同時啟動時可以錯開執行	超過延遲時間才會執行其他馬達的動作
}	
if(ServoB_Right == 1 && ServoB_Right_Status == 1) //確認按鍵"右"是否被放開且按鍵"右"狀態為 1(代表有被按過)	如果按鍵 B 被放開且伺服馬達 B 右還在執行
{	
for(int i ; i <= 10 ; i++)	訊號發送 10 次
{	
Serial1.write(27); //發送 ServoB 停止訊號	發送訊號伺服馬達 B 停止
Serial.println("ServoB_Right_Stop"); //傳送到監控視窗以利觀察是否進入此迴圈	
}	
ServoB_Right_Release_Time = currentTime; //記錄當下按壓"右"被釋放的時間	紀錄按鍵被放開的時間點
ServoB_Right_Status = 2; //改變 ServoB 按壓"右"的狀態，避免一直發送訊號	伺服馬達 B 右執行狀態改為 2
}	
if(ServoB_Right_Status == 2)	如果伺服馬達 B 右執行狀態為 2
{	
if(currentTime - ServoB_Right_Release_Time >= ServoB_DelayTime) //確認按壓"右"被釋放的時間是否超過延遲時間	馬達停止後有 0.3 秒的延遲時間
{	
ServoB_Right_Status = 0; //ServoB 按壓"右"的狀態清 0，確定已經超過延遲時間，其作用主要是確保馬達轉態時會有足夠的停止時間	超過延遲時間才會執行其他馬達的動作
}	
}	
if(ServoB_Left_Flag == 1 && ServoB_Left_Status == 0 && ServoA_Down_Press_Flag == 0 && ServoA_Up_Press_Flag == 0 && ServoB_Right_Press_Flag == 0) //確認按鍵左旗標是否動作，且各方向按壓旗標皆沒有動作	如果只有按鍵 D 被按下
{	
for(int i ; i <= 10 ; i++)	訊號發送 10 次
{	
Serial1.write(25); //發送 ServoB 左訊號	發送訊號啟動伺服馬達 B，動作向左
Serial.println("ServoB_Left"); //傳送到監控視窗以利觀察是否進入此迴圈	
}	

ServoB_Left_Press_Time = currentTime; //記錄當下按壓"左"的時間	紀錄按下按鍵的時間點
ServoB_Left_Status = 1; //改變 ServoB 按壓"左"的狀態，避免一直發送訊號	伺服馬達 B 左執行狀態改為 1
ServoB_Left_Press_Flag = 1; //ServoB 左旗標 - 1:動作，0:不動作	
}	
if(currentTime - ServoB_Left_Press_Time >= ServoB_DelayTime) //確認按下按鍵時間是否超過延遲時間	馬達開始執行後有 0.3 秒的延遲時間
{	
ServoB_Left_Press_Flag = 0; //確認超過時間後，ServoB 左旗標 - 1:動作，0:不動作，其作用是確保兩顆馬達同時啟動時可以錯開執行	超過延遲時間才會執行其他馬達的動作
}	
if(ServoB_Left == 1 && ServoB_Left_Status == 1) //確認按鍵"左"是否被放開且按鍵"左"狀態為 1(代表有被按過)	如果按鍵 D 被放開且伺服馬達 B 左還在執行
{	
for(int i ; i <= 10 ; i++)	訊號發送 10 次
{	
Serial1.write(27); //發送 ServoB 停止訊號	發送訊號伺服馬達 B 停止
Serial.println("ServoB_Left_Stop"); //傳送到監控視窗以利觀察是否進入此迴圈	
}	
ServoB_Left_Release_Time = currentTime; //記錄當下按壓"左"被釋放的時間	紀錄按鍵被放開的時間點
ServoB_Left_Status = 2; //改變 ServoB 按壓"左"的狀態，避免一直發送訊號	伺服馬達 B 左執行狀態改為 2
}	
if(ServoB_Left_Status == 2)	如果伺服馬達 B 左執行狀態為 2
{	
if(currentTime - ServoB_Left_Release_Time >= ServoB_DelayTime) //確認按壓"左"被釋放的時間是否超過延遲時間	馬達停止後有 0.3 秒的延遲時間
{	
ServoB_Left_Status = 0; //ServoB 按壓"左"的狀態清 0，確定已經超過延遲時間，其作用主要是確保馬達轉態時會有足夠的停止時間	超過延遲時間才會執行其他馬達的動作
}	
}	
}	
void Motor_JoySignal_Transmitter() //搖桿藍芽傳送 function	
{	
if(MotorA_MoveStop_Flag == 1 && MotorB_MoveStop_Flag == 1) //確認 MotorA 和 MotorB 停止旗標是否動作 - 停止	如果直流馬達 AB 執行狀態皆為停止
{	
if(JoyStop_Status == 0) //確認搖桿狀態是否為 0	如果搖桿執行狀態為 0
{	
MotorA_Time = currentTime; //記錄當下的時間，此為 MotorA 準備停下的當下時間	紀錄直流馬達 AB 執行狀態皆為停止的時間點
MotorB_Time = currentTime; //記錄當下的時間，此為 MotorB 準備停下的當下時間	
JoyStop_Status = 1; //紀錄後搖桿改變其狀態，代表已經紀錄	搖桿執行狀態改為 1

```
    }
    MotorA_Forward_Flag = 0; //MotorA 前進旗標 - 1:動作，0:不動作                    直流馬達 A 前進後退的執
    MotorA_Backward_Flag = 0; //MotorA 後退旗標 - 1:動作，0:不動作                   行狀態
    MotorB_Forward_Flag = 0; //MotorB 前進旗標 - 1:動作，0:不動作                    直流馬達 B 前進後退的執
    MotorB_Backward_Flag = 0; //MotorB 後退旗標 - 1:動作，0:不動作                   行狀態
    MotorA_Status = 0; //MotorA 狀態清 0                                          直流馬達 A 停止的狀態
    MotorB_Status = 0; //MotorB 狀態清 0                                          直流馬達 B 停止的狀態
    if(JoyStop_Status == 1)                                                      搖桿的執行狀態為 1
    {
      JoyStop_Status = 2; //準備開始發送藍芽訊號後改變搖桿狀態，代表準備開始發送訊號   搖桿的執行狀態改為 2
      for(int i ; i <= 10 ; i++)                                                 訊號發送 10 次
      {
        Serial1.write(15); //發送 MotorA 停止訊號                                發送訊號直流馬達 A 停止
        Serial1.write(16); //發送 MotorB 停止訊號                                發送訊號直流馬達 B 停止
        Serial.println("Stop"); //傳送到監控視窗以利觀察是否進入此迴圈
      }
    }
  }
  if(MotorA_Forward_Flag == 1 && MotorA_Backward_Flag == 0) //確認 MotorA 前進旗標是   如果馬達 A 前進執行狀態
否動作                                                                          為 1
  {
    if((currentTime - MotorA_Time) >= MotorA_DelayTime) //確認馬達停下時間是否超過延   如果直流馬達 A 停止超過
遲時間                                                                          0.3 秒
    {
      if(MotorA_Status == 0) //確認 MotorA 狀態是否為 0                           如果直流馬達 A 為停止狀
      {                                                                         態
        for(int i ; i <= 10 ; i++)                                              訊號發送 10 次
        {
          Serial1.write(11); //發送 MotorA 前進訊號                             發送訊號直流馬達 A 前進
          Serial.println("MotorA_Forward"); //傳送到監控視窗以利觀察是否進入此迴圈
        }
        MotorA_Status = 1; //訊號發送完畢後改變 MotorA 狀態，避免一直發送訊號
      }
    }
  }

  if(MotorA_Backward_Flag == 1 && MotorA_Forward_Flag == 1) //確認 MotorA 前進旗標和   如果直流馬達 A 的前進與
後退旗標是否都有動作，代表搖桿有改變狀態(前進變為後退或後退變為前進)，因此要讓   後退執行狀態皆為 1
MotorA 先停止
  {
    MotorA_Status = 0; //MotorA 狀態清 0                                        直流馬達 A 停止
    if(Motor_Action_Joy_Value == 4 || Motor_Action_Joy_Value == 2) //確認是哪個方向的數   如果搖桿方向為下(2)或
```

值，如果是 2 或是 4 則代表 MotorA 前進的旗標不動作	右(4)馬達 A 執行後退
{	
MotorA_Forward_Flag = 0; //MotorA 前進旗標 - 1:動作，0:不動作	前進執行狀態設為 0
MotorA_Time = currentTime; //記錄當下的時間，此為 MotorA 準備停下的當下時間	紀錄馬達 A 停下的時間點
for(int i = 0; i <= 10; i++)	訊號發送 10 次
{	
Serial1.write(15); //發送 MotorA 停止訊號	發送訊號直流馬達 A 停止
Serial.println("MotorA_Stop"); //傳送到監控視窗以利觀察是否進入此迴圈	
}	
}	
if(Motor_Action_Joy_Value == 1 || Motor_Action_Joy_Value == 3) //確認是哪個方向的數值，如果是 1 或是 3 則代表 MotorA 後退的旗標不動作	如果搖桿方向為上(1)或左(3)馬達 A 執行前進
{	
MotorA_Backward_Flag = 0; //MotorA 後退旗標 - 1:動作，0:不動作	後退執行狀態設為 0
MotorA_Time = currentTime; //記錄當下的時間，此為 MotorA 準備停下的當下時間	紀錄馬達 A 停下的時間點
for(int i = 0; i <= 10; i++)	訊號發送 10 次
{	
Serial1.write(15); //發送 MotorA 停止訊號	發送訊號直流馬達 A 停止
Serial.println("MotorA_Stop"); //傳送到監控視窗以利觀察是否進入此迴圈	
}	
}	
}	
if(MotorA_Backward_Flag == 1 && MotorA_Forward_Flag == 0) //確認 MotorA 後退旗標是否動作	如果馬達 A 後退執行狀態為 1
{	
if((currentTime - MotorA_Time) >= MotorA_DelayTime) //確認馬達停下時間是否超過延遲時間	如果直流馬達 A 停止超過 0.3 秒
{	
if(MotorA_Status == 0) //確認 MotorA 狀態是否為 0	如果直流馬達 A 為停止狀態
{	
for(int i ; i <= 10 ; i++)	訊號發送 10 次
{	
Serial1.write(12); //發送 MotorA 後退訊號	發送訊號直流馬達 A 後退
Serial.println("MotorA_Backward"); //傳送到監控視窗以利觀察是否進入此迴圈	
}	
MotorA_Status = 1; //訊號發送完畢後改變 MotorA 狀態，避免一直發送訊號	
}	
}	
}	
if(MotorB_Forward_Flag == 1 && MotorB_Backward_Flag == 0) //確認 MotorB 前進旗標是否	如果馬達 B 前進執行狀態

動作	為 1
{	
if((currentTime - MotorB_Time) >= MotorB_DelayTime) //確認馬達停下時間是否超過延遲時間	如果直流馬達 B 停止超過 0.3 秒
{	
if(MotorB_Status == 0) //確認 MotorB 狀態是否為 0	如果直流馬達 B 為停止狀態
{	
for(int i ; i <= 10 ; i++)	訊號發送 10 次
{	
Serial1.write(13); //發送 MotorB 前進訊號	發送訊號直流馬達 B 前進
Serial.println("MotorB_Forward"); //傳送到監控視窗以利觀察是否進入此迴圈	
}	
MotorB_Status = 1; //訊號發送完畢後改變 MotorB 狀態，避免一直發送訊號	
}	
}	
}	
if(MotorB_Backward_Flag == 1 && MotorB_Forward_Flag == 1) //確認MotorB前進旗標和後退旗標是否都有動作,代表搖桿有改變狀態(前進變為後退或後退變為前進),因此要讓 MotorB 先停止	如果直流馬達 B 的前進與後退執行狀態皆為 1
{	
MotorB_Status = 0; //MotorB 狀態清 0	直流馬達 B 停止
if(Motor_Action_Joy_Value == 3 || Motor_Action_Joy_Value == 2) //確認是哪個方向的數值，如果是 2 或是 3 則代表 MotorB 前進的旗標不動作	如果搖桿方向為下(2)或左(3)馬達 B 執行後退
{	
MotorB_Forward_Flag = 0; //MotorB 前進旗標 - 1:動作，0:不動作	前進執行狀態設為 0
MotorB_Time = currentTime; //記錄當下的時間，此為 MotorB 準備停下的當下時間	紀錄馬達 B 停下的時間點
for(int i = 0; i <= 10; i++)	訊號發送 10 次
{	
Serial1.write(16); //發送 MotorB 停止訊號	發送訊號直流馬達 B 停止
Serial.println("MotorB_Stop"); //傳送到監控視窗以利觀察是否進入此迴圈	
}	
}	
if(Motor_Action_Joy_Value == 1 || Motor_Action_Joy_Value == 4) //確認是哪個方向的數值，如果是 1 或是 4 則代表 MotorB 後退的旗標不動作	如果搖桿方向為上(1)或右(4)馬達 B 執行前進
{	
MotorB_Backward_Flag = 0; //MotorB 後退旗標 - 1:動作，0:不動作	後退執行狀態設為 0
MotorB_Time = currentTime; //記錄當下的時間，此為 MotorB 準備停下的當下時間	紀錄馬達 B 停下的時間點
for(int i = 0; i <= 10; i++)	訊號發送 10 次
{	
Serial1.write(16); //發送 MotorB 停止訊號	發送訊號直流馬達 B 停止
Serial.println("MotorB_Stop"); //傳送到監控視窗以利觀察是否進入此迴圈	

```
      }
    }
  }

  if(MotorB_Backward_Flag == 1 && MotorB_Forward_Flag == 0) //確認MotorB後退旗標是否動作
  {
    if((currentTime - MotorB_Time) >= MotorB_DelayTime) //確認馬達停下時間是否超過延遲時間
    {
      if(MotorB_Status == 0) //確認 MotorB 狀態是否為 0
      {
        for(int i ; i <= 10 ; i++)
        {
          Serial1.write(14); //發送 MotorB 後退訊號
          Serial.println("MotorB_Backward"); //傳送到監控視窗以利觀察是否進入此迴圈
        }
        MotorB_Status = 1; //訊號發送完畢後改變 MotorB 狀態，避免一直發送訊號
      }
    }
  }
}

void SW_TX() //按壓按鍵狀態 Function
{
  ServoA_Up = digitalRead(SW1); //11
  ServoB_Right = digitalRead(SW2); //22
  ServoA_Down = digitalRead(SW3); //33
  ServoB_Left = digitalRead(SW4); //44
  if(ServoA_Up == 0 && ServoB_Right == 1 && ServoA_Down == 1 && ServoB_Left == 1) //判斷上按鍵狀態
  {
      ServoA_Up_Flag = 1; //上旗標 - 1:動作，0:不動作
      ServoA_Down_Flag = 0; //下旗標 - 1:動作，0:不動作
      ServoB_Left_Flag = 0; //左旗標 - 1:動作，0:不動作
      ServoB_Right_Flag = 0; //右旗標 - 1:動作，0:不動作
      ServoA_MoveStop_Flag = 0; //ServoA 馬達停止旗標 - 1:動作(停止)，0:不動作
      ServoB_MoveStop_Flag = 1; //ServoB 馬達停止旗標 - 1:動作(停止)，0:不動作
  }
  else if(ServoA_Up == 1 && ServoB_Right == 0 && ServoA_Down == 1 && ServoB_Left == 1) //判斷右按鍵狀態
  {
```

```
如果馬達 B 後退執行狀態為 1
如果直流馬達 B 停止超過 0.3 秒
如果直流馬達 B 為停止狀態
訊號發送 10 次
發送訊號直流馬達 B 後退
判斷 A 鍵是否被按下
判斷 B 鍵是否被按下
判斷 C 鍵是否被按下
判斷 D 鍵是否被按下
如果 A 鍵被按下
伺服馬達
A 上執行狀態 1
A 下執行狀態 0
B 左執行狀態 0
B 右執行狀態 0
A 停執行狀態 0
B 停執行狀態 1
否則如果 B 鍵被按下
伺服馬達
```

ServoB_Right_Flag = 1; //右旗標 - 1:動作，0:不動作	B 右執行狀態 1
ServoA_Up_Flag = 0; //上旗標 - 1:動作，0:不動作	A 上執行狀態 0
ServoA_Down_Flag = 0; //下旗標 - 1:動作，0:不動作	A 下執行狀態 0
ServoB_Left_Flag = 0; //左旗標 - 1:動作，0:不動作	B 左執行狀態 0
ServoA_MoveStop_Flag = 1; //ServoA 馬達停止旗標 - 1:動作(停止)，0:不動作	A 停執行狀態 1
ServoB_MoveStop_Flag = 0; //ServoB 馬達停止旗標 - 1:動作(停止)，0:不動作	B 停執行狀態 0
}	
else if(ServoA_Up == 1 && ServoB_Right == 1 && ServoA_Down == 0 && ServoB_Left == 1) //判斷上按鍵狀態	否則如果 C 鍵被按下
{	伺服馬達
ServoA_Down_Flag = 1; //下旗標 - 1:動作，0:不動作	A 下執行狀態 1
ServoA_Up_Flag = 0; //上旗標 - 1:動作，0:不動作	A 上執行狀態 0
ServoB_Left_Flag = 0; //左旗標 - 1:動作，0:不動作	B 左執行狀態 0
ServoB_Right_Flag = 0; //右旗標 - 1:動作，0:不動作	B 右執行狀態 0
ServoA_MoveStop_Flag = 0; //ServoA 馬達停止旗標 - 1:動作(停止)，0:不動作	A 停執行狀態 0
ServoB_MoveStop_Flag = 1; //ServoB 馬達停止旗標 - 1:動作(停止)，0:不動作	B 停執行狀態 1
}	
else if(ServoA_Up == 1 && ServoB_Right == 1 && ServoA_Down == 1 && ServoB_Left == 0) //判斷左按鍵狀態	否則如果 D 鍵被按下
{	伺服馬達
ServoB_Left_Flag = 1; //左旗標 - 1:動作，0:不動作	B 左執行狀態 1
ServoA_Up_Flag = 0; //上旗標 - 1:動作，0:不動作	A 上執行狀態 0
ServoA_Down_Flag = 0; //下旗標 - 1:動作，0:不動作	A 下執行狀態 0
ServoB_Right_Flag = 0; //右旗標 - 1:動作，0:不動作	B 右執行狀態 0
ServoA_MoveStop_Flag = 1; //ServoA 馬達停止旗標 - 1:動作(停止)，0:不動作	A 停執行狀態 1
ServoB_MoveStop_Flag = 0; //ServoB 馬達停止旗標 - 1:動作(停止)，0:不動作	B 停執行狀態 0
}	
else if(ServoA_Up == 0 && ServoB_Right == 0 && ServoA_Down == 1 && ServoB_Left == 1) //判斷上&右按鍵狀態	否則如果 A,B 鍵被按下
{	伺服馬達
ServoA_Up_Flag = 1; //上旗標 - 1:動作，0:不動作	A 上執行狀態 1
ServoB_Right_Flag = 1; //右旗標 - 1:動作，0:不動作	B 右執行狀態 1
ServoA_Down_Flag = 0; //下旗標 - 1:動作，0:不動作	A 下執行狀態 0
ServoB_Left_Flag = 0; //左旗標 - 1:動作，0:不動作	B 左執行狀態 0
ServoA_MoveStop_Flag = 0; //ServoA 馬達停止旗標 - 1:動作(停止)，0:不動作	A 停執行狀態 0
ServoB_MoveStop_Flag = 0; //ServoB 馬達停止旗標 - 1:動作(停止)，0:不動作	B 停執行狀態 0
}	
else if(ServoA_Up == 0 && ServoB_Right == 1 && ServoA_Down == 1 && ServoB_Left == 0) //判斷上&左按鍵狀態	否則如果 A,D 鍵被按下
{	伺服馬達
ServoA_Up_Flag = 1; //上旗標 - 1:動作，0:不動作	A 上執行狀態 1
ServoB_Left_Flag = 1; //左旗標 - 1:動作，0:不動作	B 左執行狀態 1

ServoA_Down_Flag = 0; //下旗標 - 1:動作，0:不動作	A 下執行狀態 0
ServoB_Right_Flag = 0; //右旗標 - 1:動作，0:不動作	B 右執行狀態 0
ServoA_MoveStop_Flag = 0; //ServoA 馬達停止旗標 - 1:動作(停止)，0:不動作	A 停執行狀態 0
ServoB_MoveStop_Flag = 0; //ServoB 馬達停止旗標 - 1:動作(停止)，0:不動作	B 停執行狀態 0
}	
else if(ServoA_Up == 1 && ServoB_Right == 0 && ServoA_Down == 0 && ServoB_Left == 1) //判斷下&右按鍵狀態	否則如果 B,C 鍵被按下
{	伺服馬達
ServoA_Down_Flag = 1; //下旗標 - 1:動作，0:不動作	A 下執行狀態 1
ServoB_Right_Flag = 1; //右旗標 - 1:動作，0:不動作	B 右執行狀態 1
ServoA_Up_Flag = 0; //上旗標 - 1:動作，0:不動作	A 上執行狀態 0
ServoB_Left_Flag = 0; //左旗標 - 1:動作，0:不動作	B 左執行狀態 0
ServoA_MoveStop_Flag = 0; //ServoA 馬達停止旗標 - 1:動作(停止)，0:不動作	A 停執行狀態 0
ServoB_MoveStop_Flag = 0; //ServoB 馬達停止旗標 - 1:動作(停止)，0:不動作	B 停執行狀態 0
}	
else if(ServoA_Up == 1 && ServoB_Right == 1 && ServoA_Down == 0 && ServoB_Left == 0) //判斷下&左按鍵狀態	否則如果 C,D 鍵被按下
{	伺服馬達
ServoA_Down_Flag = 1; //下旗標 - 1:動作，0:不動作	A 下執行狀態 1
ServoB_Left_Flag = 1; //左旗標 - 1:動作，0:不動作	B 左執行狀態 1
ServoA_Up_Flag = 0; //上旗標 - 1:動作，0:不動作	A 上執行狀態 0
ServoB_Right_Flag = 0; //右旗標 - 1:動作，0:不動作	B 右執行狀態 0
ServoA_MoveStop_Flag = 0; //ServoA 馬達停止旗標 - 1:動作(停止)，0:不動作	A 停執行狀態 0
ServoB_MoveStop_Flag = 0; //ServoB 馬達停止旗標 - 1:動作(停止)，0:不動作	B 停執行狀態 0
}	
else	否則
{	伺服馬達
ServoA_Down_Flag = 0; //下旗標 - 1:動作，0:不動作	A 下執行狀態 0
ServoB_Right_Flag = 0; //右旗標 - 1:動作，0:不動作	B 右執行狀態 0
ServoA_Up_Flag = 0; //上旗標 - 1:動作，0:不動作	A 上執行狀態 0
ServoB_Left_Flag = 0; //左旗標 - 1:動作，0:不動作	B 左執行狀態 0
ServoA_MoveStop_Flag = 1; //ServoA 馬達停止旗標 - 1:動作(停止)，0:不動作	A 停執行狀態 1
ServoB_MoveStop_Flag = 1; //ServoB 馬達停止旗標 - 1:動作(停止)，0:不動作	B 停執行狀態 1
}	
}	
void JoyStick_TX() //搖桿狀態 Function	
{	
vrx = analogRead(A0);	判斷搖桿被按下的類比讀
vry = analogRead(A1);	值
x_axis = map(vrx, 512, 0, 0, 255); // 向右為正，向左為負	x 軸(左右)換算至 0-255
//https://www.arduino.cc/reference/en/language/functions/math/map/ - 上述網站有詳細函示解說	

y_axis = map(vry, 512, 0, 0, 255); //向下為正，向上為負	y 軸(上下)換算至 0-255
if(y_axis <= -220 && (x_axis <= 70 && x_axis >= -70)) //根據不同讀值來判斷搖桿的方向 - 前進	如果方向鍵 上被按下
{	
Motor_Action_Joy_Value = 1; //前進方向代表的數值	車體運作方式 1(前進)
MotorA_Forward_Flag = 1; //MotorA 前進旗標 - 1:動作，0:不動作 //MotorA 右邊、MotorB 左邊	馬達 A 前進執行狀態 1
MotorB_Forward_Flag = 1; //MotorB 前進旗標 - 1:動作，0:不動作	馬達 B 前進執行狀態 1
MotorA_MoveStop_Flag = 0; //MotorA 馬達停止旗標 - 1:動作(停止)，0:不動作	馬達 A 停止執行狀態 0
MotorB_MoveStop_Flag = 0; //MotorB 馬達停止旗標 - 1:動作(停止)，0:不動作	馬達 B 停止執行狀態 0
JoyStop_Status = 0; //每次搖桿改變方向後狀態清 0	搖桿執行狀態設為 0
}	
else if(y_axis >= 220 && (x_axis <= 70 && x_axis >= -70)) //根據不同讀值來判斷搖桿的方向 - 後退	否則如果方向鍵 下被按下
{	
Motor_Action_Joy_Value = 2; //後退方向代表的數值	車體運作方式 2(後退)
MotorA_Backward_Flag = 1; //MotorA 後退旗標 - 1:動作，0:不動作	馬達 A 後退執行狀態 1
MotorB_Backward_Flag = 1; //MotorB 後退旗標 - 1:動作，0:不動作	馬達 B 後退執行狀態 1
MotorA_MoveStop_Flag = 0; //MotorA 馬達停止旗標 - 1:動作(停止)，0:不動作	馬達 A 停止執行狀態 0
MotorB_MoveStop_Flag = 0; //MotorB 馬達停止旗標 - 1:動作(停止)，0:不動作	馬達 B 停止執行狀態 0
JoyStop_Status = 0; //每次搖桿改變方向後狀態清 0	搖桿執行狀態設為 0
}	
else if(x_axis >= 220 && (y_axis <= 70 && y_axis >= -70)) //根據不同讀值來判斷搖桿的方向 - 左轉	否則如果方向鍵 左被按下
{	
Motor_Action_Joy_Value = 3; //左轉方向代表的數值	車體運作方式 3(左轉)
MotorA_Forward_Flag = 1; //MotorA 前進旗標 - 1:動作，0:不動作	馬達 A 前進執行狀態 1
MotorB_Backward_Flag = 1; //MotorB 後退旗標 - 1:動作，0:不動作	馬達 B 後退執行狀態 1
MotorA_MoveStop_Flag = 0; //MotorA 馬達停止旗標 - 1:動作(停止)，0:不動作	馬達 A 停止執行狀態 0
MotorB_MoveStop_Flag = 0; //MotorB 馬達停止旗標 - 1:動作(停止)，0:不動作	馬達 B 停止執行狀態 0
JoyStop_Status = 0; //每次搖桿改變方向後狀態清 0	搖桿執行狀態設為 0
}	
else if(x_axis <= -220 && (y_axis <= 70 && y_axis >= -70)) //根據不同讀值來判斷搖桿的方向 - 右轉	否則如果方向鍵 右被按下
{	
Motor_Action_Joy_Value = 4; //右轉方向代表的數值	車體運作方式 4(右轉)
MotorA_Backward_Flag = 1; //MotorA 後退旗標 - 1:動作，0:不動作	馬達 A 後退執行狀態 1
MotorB_Forward_Flag = 1; //MotorB 前進旗標 - 1:動作，0:不動作	馬達 B 前進執行狀態 1
MotorA_MoveStop_Flag = 0; //MotorA 馬達停止旗標 - 1:動作(停止)，0:不動作	馬達 A 停止執行狀態 0
MotorB_MoveStop_Flag = 0; //MotorB 馬達停止旗標 - 1:動作(停止)，0:不動作	馬達 B 停止執行狀態 0
JoyStop_Status = 0; //每次搖桿改變方向後狀態清 0	搖桿執行狀態設為 0
}	

else if((y_axis >= -20 && y_axis <= 20) && (x_axis >= -25 && x_axis <= 25)) //根據不同讀值來判斷搖桿的方向 - 停止	否則如果方向鍵沒被按下
{	
MotorA_MoveStop_Flag = 1; //MotorA 馬達停止旗標 - 1:動作(停止)，0:不動作	馬達 A 停止執行狀態 1
MotorB_MoveStop_Flag = 1; //MotorB 馬達停止旗標 - 1:動作(停止)，0:不動作	馬達 B 停止執行狀態 1
}	
}	
void loop() {	
currentTime = millis(); //https://www.arduino.cc/reference/en/language/functions/time/millis/ - 上述網站有詳細函示解說	
SW_TX();	
Motor_SwitchSignal_Transmitter();	
JoyStick_TX();	
Motor_JoySignal_Transmitter();	
}	