

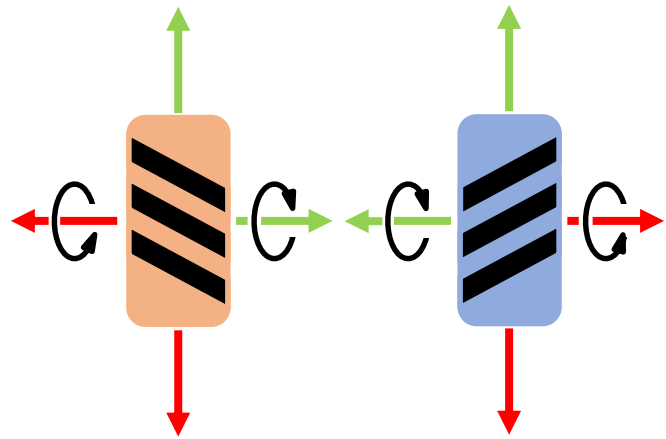
# WIFI 模組教學



# 車輪控制

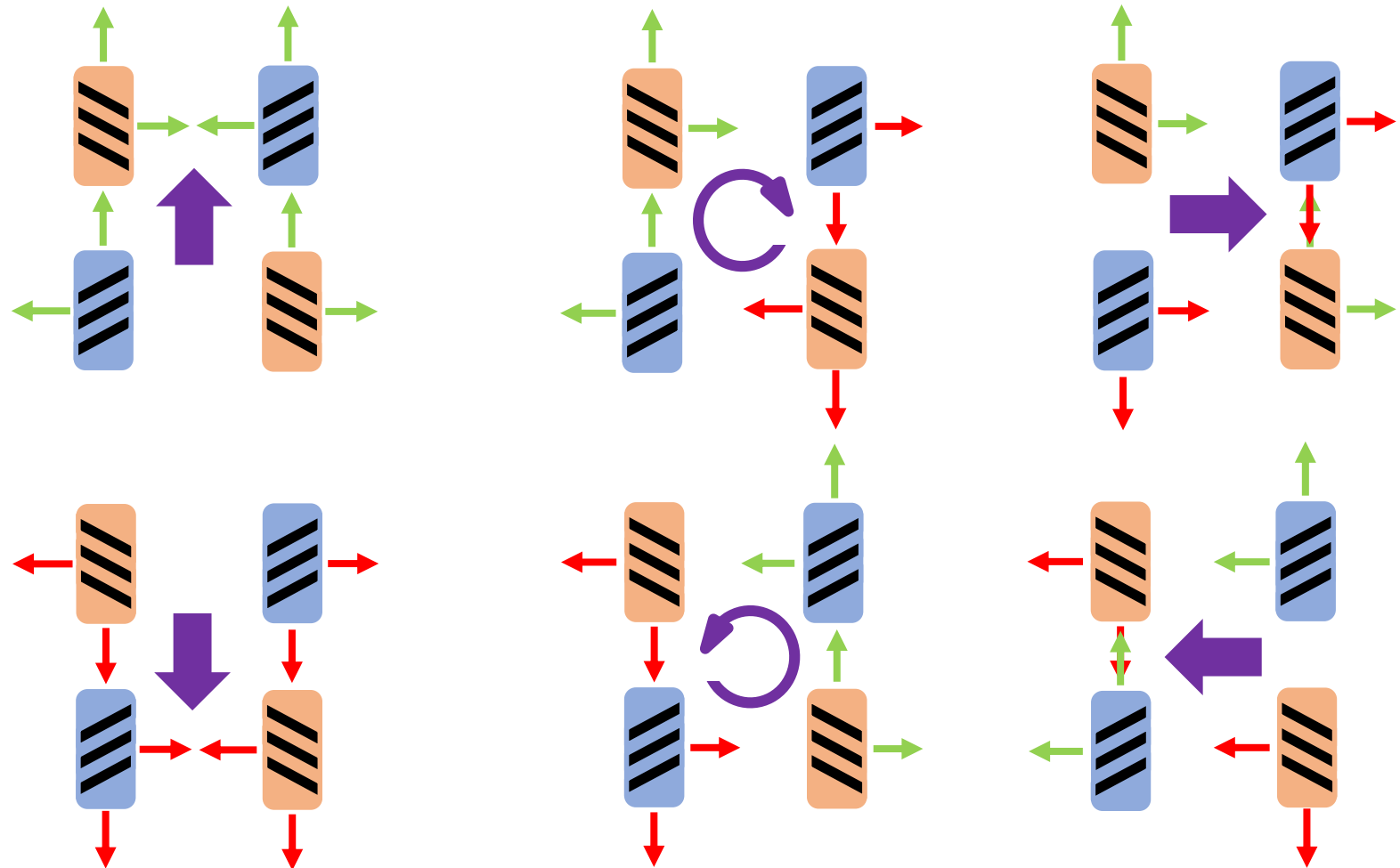


# 麥克納姆輪

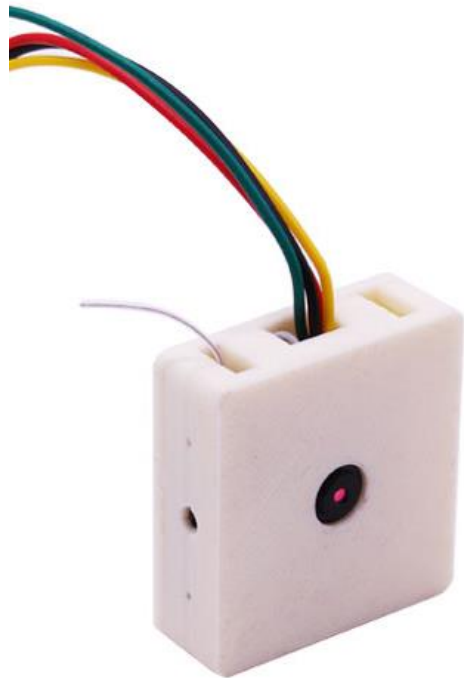


綠色 正轉造成的車輪作用力方向  
紅色 反轉造成的車輪作用力方向  
紫色 為合力方向

車輪必須遵照如右圖的配置否則會異常



# 硬體



Wi-Fi 攝影機模組

解析度：30 萬

傳輸方式 WIFI

傳輸距離 70M

控制輸出 APP 控制 + 序列埠

天線增益 2.17 Db

工作電流 220 ~ 320mA

工作電壓 3.5~5V



# APP 安裝說明



Android

[Yahboom Robot - Google Play 應用程式](#)



IOS

[在 App Store 上的「YahBoomRobot」\(apple.com\)](#)



# 連線說明

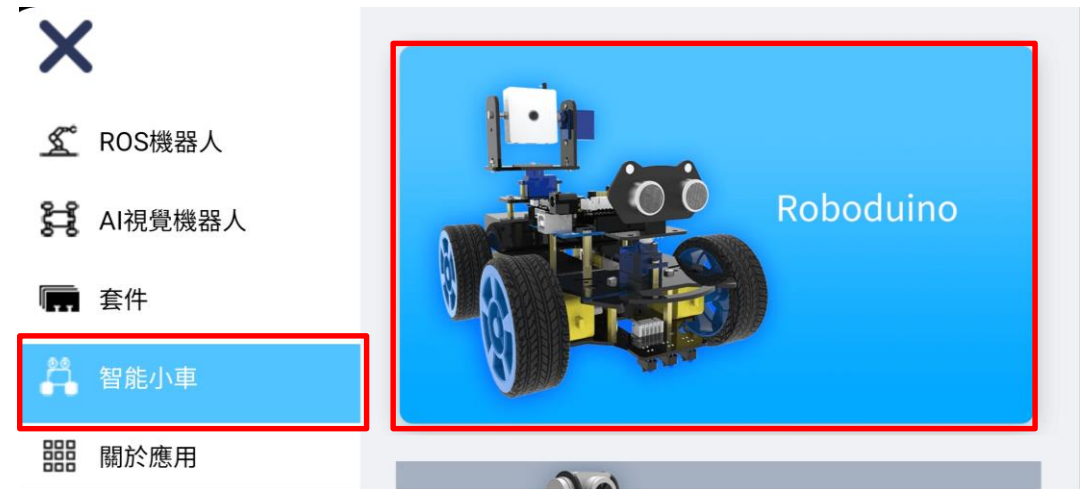
WI-FI 模組上電，透過手機可以看到 Yahnoom\_ 開頭的 WI-FI，Android 手機連線後可能會出現無法使用網際網路，需要點選保持 Wi-Fi 連線



# APP 操作說明



1. 點選右上角選單

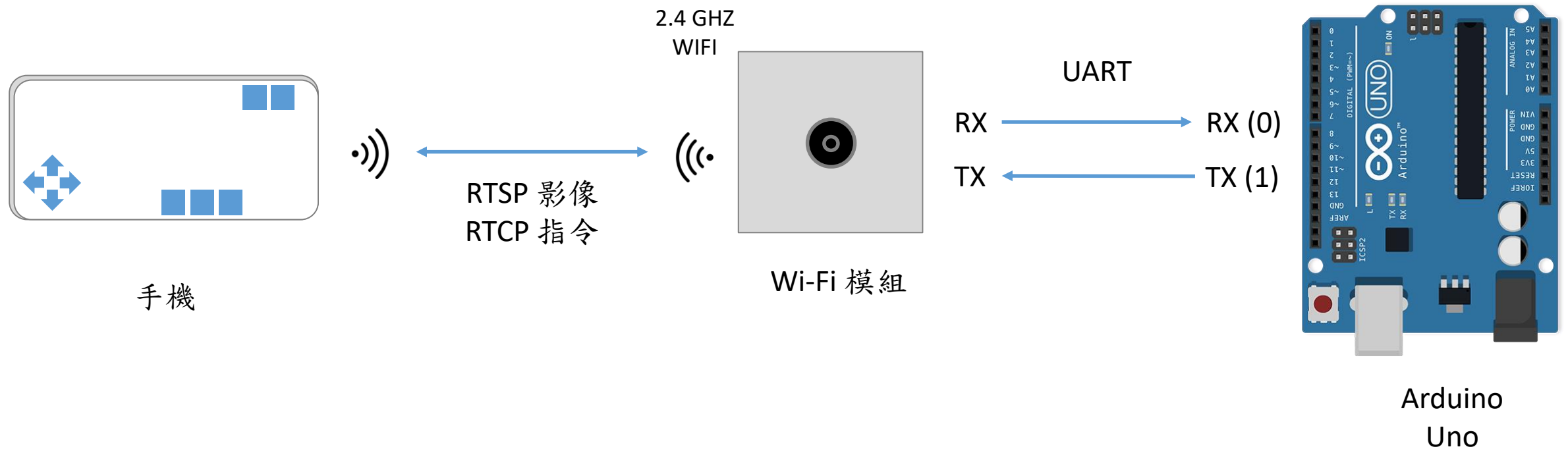


2. 點選智能小車

3. 選擇 Roboduino

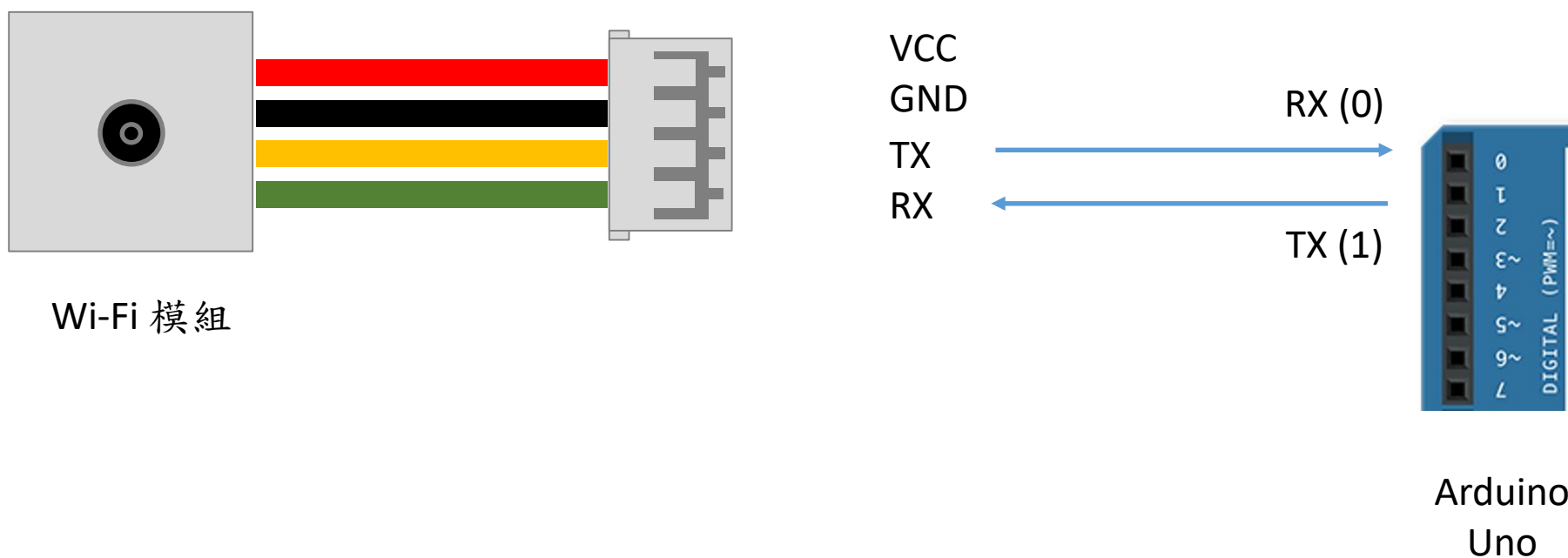


# 原理





# 傳輸協定解析



Note : 如果 WIFI 模組 RX 收到未知的指令會從 TX 發送亂碼，所以 Arduino TX 不需要連線到 Wi-Fi 模組 RX 上，且 RX 需加一顆電阻做上拉

# Serial



# Serial

- 由於 WIFI 模組回傳的訊息是整串的文字，所以我們需要一個程式把 Serial 回傳的訊息合併成一個字串

當按下左旋轉時，Serial 會收到 \$5,0,0,0#

索引	0	1	2	3	4	5	6	7	8
	開頭	命令位1	分隔號	命令位2	分隔號	命令位3	分隔號	命令位4	結尾
左旋轉	\$	5	,	0	,	0	,	0	#

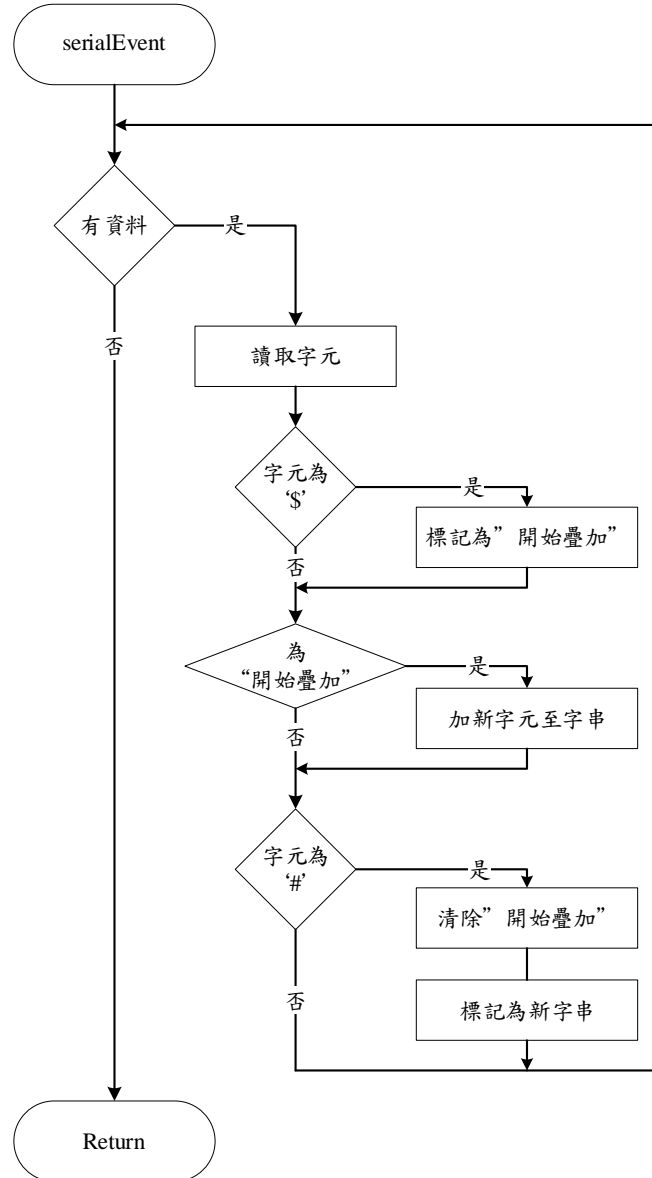
我們會發現他發送的每一條命令皆有開頭及結尾

可以利用 `Serial.read()`；一次讀取一個字元，  
 當字元為 “\$”，開始組合接下來的字元成一個字串  
 當字元為 “#”，將剛剛組合的字串標記為完成

所以當字串被標記為完成後，我們就可以讀出整個完整的指令。



# Serial



```

void serialEvent()
{
    while (Serial.available() && !Serial_newLine)
    {
        //一個字節一個字節地讀,下一句是讀到的放入字符串數組中組成一個完成的數據包
        char IncomingByte = Serial.read();
        if (IncomingByte == '\0')
            continue;
        if (IncomingByte == '$') // 當接收到 開頭字元
        {
            Serial_startFlag = true; // 開始組合資料
        }
        if (Serial_startFlag == true)
            Serial_inputString += (char)IncomingByte; // 將字元一個個串接
        if (IncomingByte == '#') // 當接收到結尾字元
        {
            Serial_startFlag = false; // 結束接收串接程序
            Serial_newLine = true; // 標記為資料接收完成 (完整字串)
        }
    }
}

```



# Serial

序列埠資料接收完後，可以將透過 `Serial_newline` 判斷是否有新資料。

這裡將資料接收額外包在另一個函式，這個函示可以放在 `loop()` 內或 `serialEvent();` 內的最後一行。

注意，資料解析完後，要清除 `Serial_newLine` 和 `Serial_inputString` 兩個，否則下一筆資料會異常。

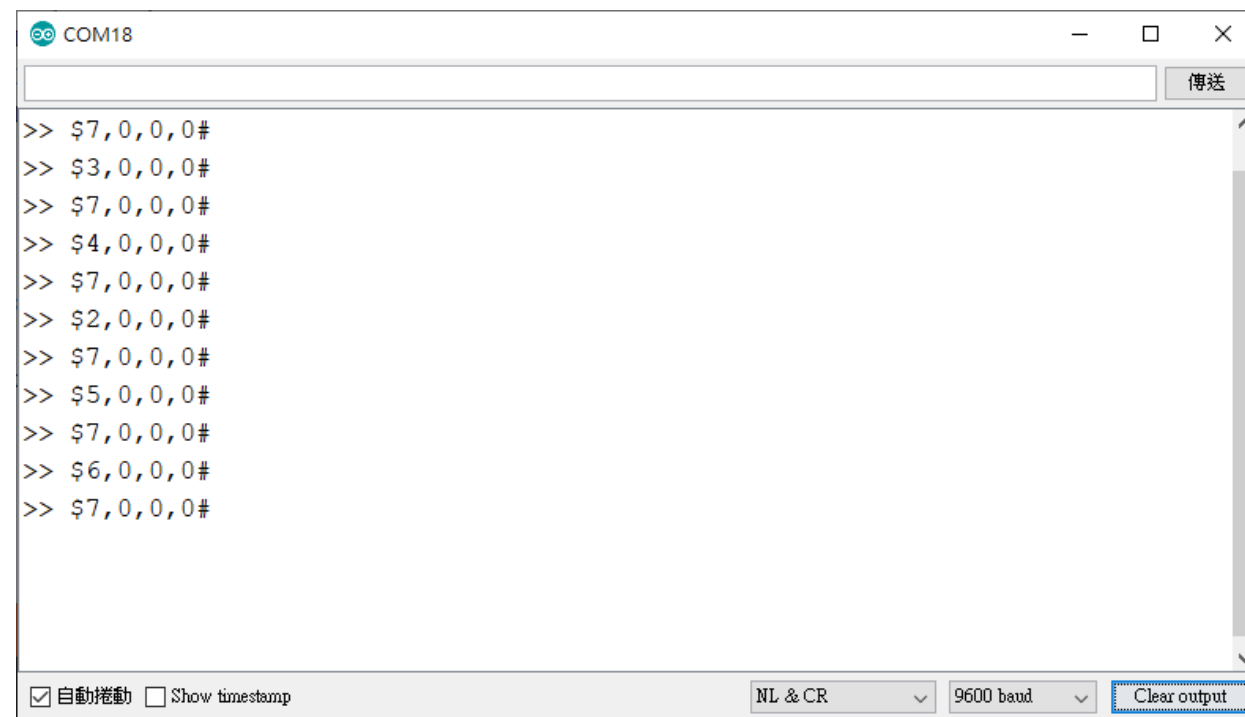
```
// 資料解析
//
void Serial_dataParse()
{
    if (!Serial_newLine) // 如果沒有新資料
        return;         // 結束此資料解析
    Serial.print(">> ");
    Serial.println(Serial_inputString);

    // 解析完成，清除
    Serial_newLine = false;
    Serial_inputString = "";
}
```



# 實驗 1

- 將 WI-FI 視訊模組連接至 Arduino
- 使用 APP 連接 WI-FI 視訊模組
- 使用 序列埠監控視窗 觀察在 APP 中按下按鈕時的輸出



# 傳輸協定解析 - 1

通用協議		移動方向		加減速		鳴笛		點燈	
	0	1	2	3	4	5	6	7	8
	開頭	命令位1	分隔號	命令位2	分隔號	命令位3	分隔號	命令位4	結尾
無	\$	0	,	0	,	0	,	0	#
前進	\$	1	,	0	,	0	,	0	#
後退	\$	2	,	0	,	0	,	0	#
左轉	\$	3	,	0	,	0	,	0	#
右轉	\$	4	,	0	,	0	,	0	#
左旋轉	\$	5	,	0	,	0	,	0	#
右旋轉	\$	6	,	0	,	0	,	0	#
停止	\$	7	,	0	,	0	,	0	#
加速	\$	0	,	1	,	0	,	0	#
減速	\$	0	,	2	,	0	,	0	#
鳴笛	\$	0	,	0	,	1	,	0	#
點燈	\$	0	,	0	,	0	,	1	#

Ex. 當按下左旋轉時，Serial 會收到 \$5,0,0,0#



# 傳輸協定解析 - 2

## 舵機

索引	0	6	7	9	10
左搖	\$Servo	,	LR	L	#
右搖	\$Servo	,	LR	R	#
暫停	\$Servo	,	LR	S	#
索引	0	6	7	9	12
角度	\$Servo	,	UD	角度(三位)	#

## 音樂

索引	0	6	7	9
小星星	\$Music	,	11	#
賓果	\$Music	,	21	#
聖誕快樂	\$Music	,	31	#
歡樂頌	\$Music	,	41	#
生日快樂	\$Music	,	51	#
關閉音樂	\$Music	,	00	#

## 模式

索引	0	5	6	8
七彩	\$Mode	,	11	#
避障	\$Mode	,	21	#
循線	\$Mode	,	31	#
關閉	\$Mode	,	00	#





# 指令解析

通用協議		移動方向		加減速		鳴笛		點燈	
	0	1	2	3	4	5	6	7	8
	開頭	命令位1	分隔號	命令位2	分隔號	命令位3	分隔號	命令位4	結尾
無	\$	0	,	0	,	0	,	0	#
前進	\$	1	,	0	,	0	,	0	#
後退	\$	2	,	0	,	0	,	0	#
左轉	\$	3	,	0	,	0	,	0	#

- 我們可以觀察到每個欄位皆有功能，且每個功能都是用“,”隔開來
- 假設我們今天要讀取移動方向，就只要讀取索引為 1 的欄位內的值，依此類推

```
int direction = Serial_inputString[1] - '0';
int speedCtrl = Serial_inputString[3] - '0';
int beep = Serial_inputString[5] - '0';
int light = Serial_inputString[7] - '0';
```



# 指令解析

```
//  
// 資料解析  
//  
Void Serial_dataParse()  
{  
    if (!Serial_newLine) // 如果沒有新資料  
        return; // 結束此資料解析  
    Serial.print(">> ");  
    Serial.println(Serial_inputString);  
  
    generalCommand_dataParse();  
  
    // 解析完成，清除  
    Serial_newLine = false;  
    Serial_inputString = "";  
}
```

- 將通用協定的解析放入副函式內，因為後期會有多個協定。
- 在通用協定資料解析的副函式放入 `Serial_dataParse()`

```
//  
// WIFI 模組解析  
//  
void generalCommand_dataParse()  
{  
    //小車原地左旋右旋判斷  
    int direction = Serial_inputString[1] - '0';  
    int speedCtrl = Serial_inputString[3] - '0';  
    int beep = Serial_inputString[5] - '0';  
    int light = Serial_inputString[7] - '0';  
  
    setDirection(direction);  
  
    //速度控制  
    switch (speedCtrl)  
    {  
        case 0:  
            break;  
        case 1:  
            PWM_SPEED += 10;  
            if (PWM_SPEED > 255)  
                PWM_SPEED = 255;  
            break;  
        case 2:  
            PWM_SPEED -= 10;  
            if (PWM_SPEED < 10)  
                PWM_SPEED = 10;  
            break;  
    }  
    Serial.println(PWM_SPEED);  
}
```



# 指令解析

- 再依照 direction 去做指定的動作。

通用協定		移動方向		加減速		鳴笛		點燈	
	0	1	2	3	4	5	6	7	8
	開頭	命令位1	分隔號	命令位2	分隔號	命令位3	分隔號	命令位4	結尾
無	\$	0	,	0	,	0	,	0	#
前進	\$	1	,	0	,	0	,	0	#
後退	\$	2	,	0	,	0	,	0	#
左轉	\$	3	,	0	,	0	,	0	#
右轉	\$	4	,	0	,	0	,	0	#
左旋轉	\$	5	,	0	,	0	,	0	#
右旋轉	\$	6	,	0	,	0	,	0	#
停止	\$	7	,	0	,	0	,	0	#

```

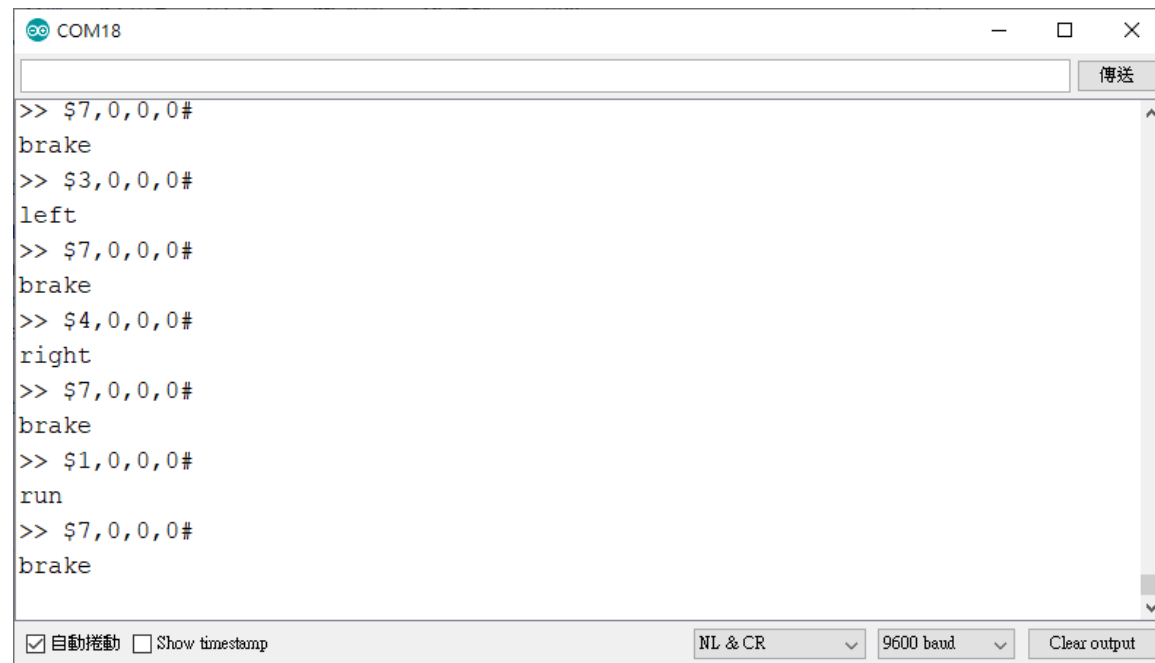
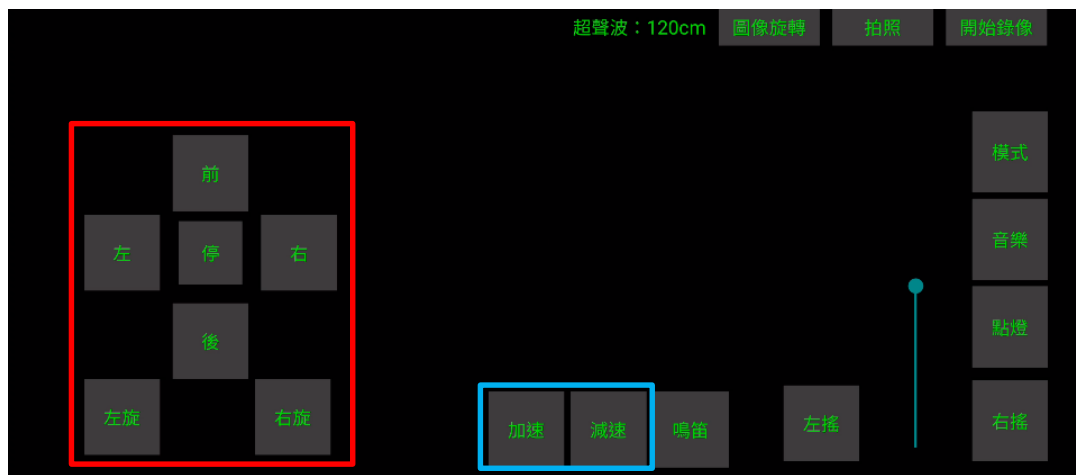
void setDirection(int direction)
{
    //方向控制
    switch (direction)
    {
        case 0:
            Serial.println("brake");
            brake();
            break;
        case 1:
            Serial.println("run");
            run();
            break;
        case 2:
            Serial.println("back");
            back();
            break;
        case 3:
            Serial.println("left");
            left();
            break;
        case 4:
            Serial.println("right");
            right();
            break;
        case 5:
            Serial.println("turn left");
            spin_left();
            break;
        case 6:
            Serial.println("turn right");
            spin_right();
            break;
        case 7:
            Serial.println("brake");
            brake();
            break;
    }
}

```



## 實驗 2

- 使用 APP 發送指令，並能分析指令所對應的動作並印出  
例如收到 \$3,0,0,0# 會印出 left，依此類推
- 控制車子移動
- 加速減速按鈕控制車子速度



```
>> $7,0,0,0#
brake
>> $3,0,0,0#
left
>> $7,0,0,0#
brake
>> $4,0,0,0#
right
>> $7,0,0,0#
brake
>> $1,0,0,0#
run
>> $7,0,0,0#
brake
```



# 舵機控制

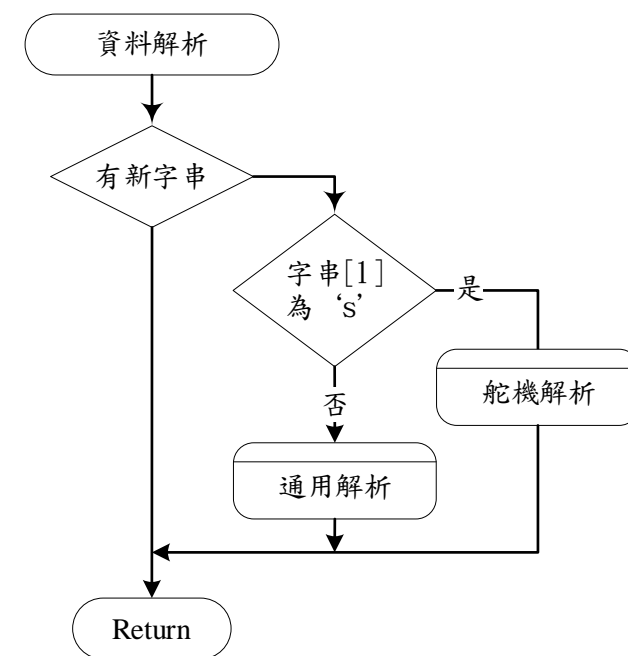


# 舵機協定解析

索引	0	6	7	9	10
左搖	\$Servo	,	LR	L	#
右搖	\$Servo	,	LR	R	#
暫停	\$Servo	,	LR	S	#
索引	0	6	7	9	12
角度	\$Servo	,	UD	角度(三位)	#

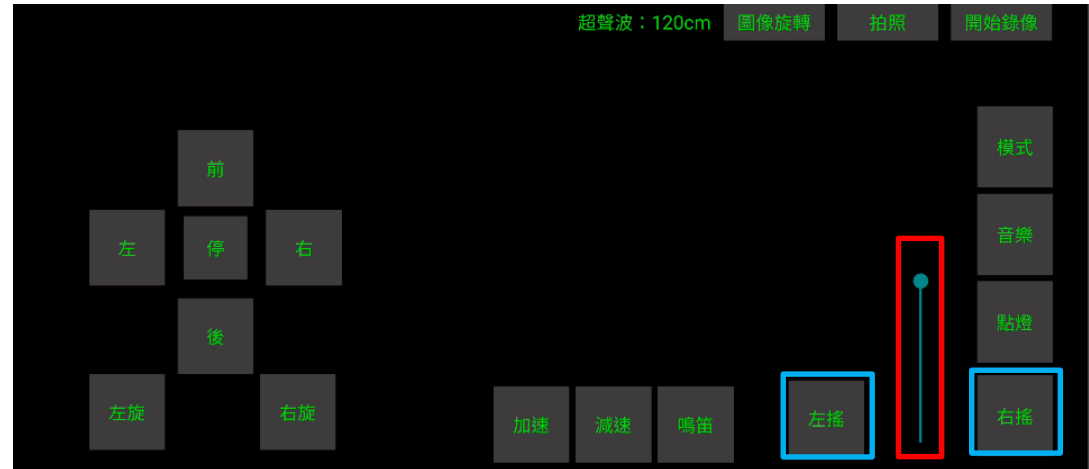
通用協定		移動方向		加減速		鳴笛		點燈	
索引	0	1	2	3	4	5	6	7	8
	開頭	命令位1	分隔號	命令位2	分隔號	命令位3	分隔號	命令位4	結尾
無	\$	0	,	0	,	0	,	0	#
前進	\$	1	,	0	,	0	,	0	#
後退	\$	2	,	0	,	0	,	0	#
左轉	\$	3	,	0	,	0	,	0	#

- 舵機協定跟通用協定開頭不同，我們可以透過檢查舵機開頭的索引 1，判斷是不是 's'，當為 's' 進去 舵機資料解析  
如果不是 's'，就當作通用協定去解析



# 舵機協定解析

索引	0	6	7	9	10
左搖	\$Servo	,	LR	L	#
右搖	\$Servo	,	LR	R	#
暫停	\$Servo	,	LR	S	#
索引	0	6	7	9	12
角度	\$Servo	,	UD	角度(三位)	#



- 舵機協定解析又有分兩種，一個是單純控制方向，一個是可以透過 滑動條 直接控制對應角度
- 兩者差異在 LR 及 UD，可以用前頁方法判斷是 L 還是 U
- 如果是 L 在對索引為 9 的進行判斷
- 如果是 U 就解析角度



# 字串轉換成數字

索引	0	6	7	9	10
左搖	\$Servo	,	LR	L	#
右搖	\$Servo	,	LR	R	#
暫停	\$Servo	,	LR	S	#
索引	0	6	7	9	12
角度	\$Servo	,	UD	角度 (三位)	#

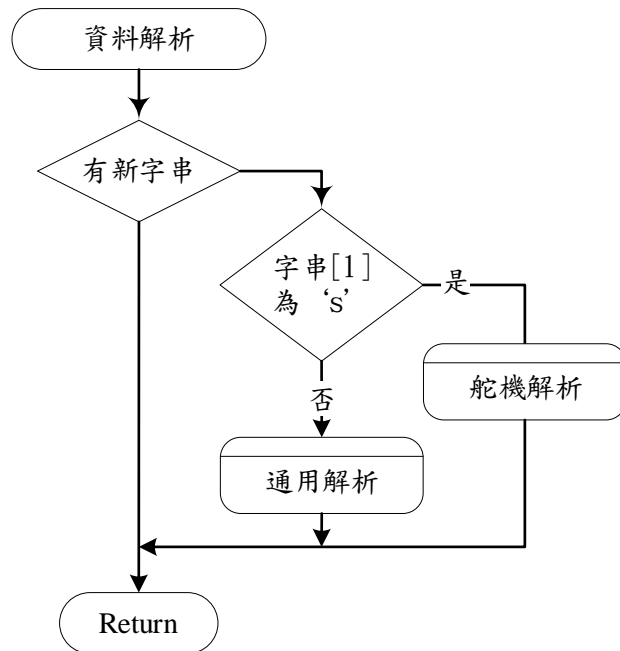
- 角度那位有三位，控制 App 的 滑動條可以輸出 000 ~ 180
- 對 [9] [10] [11] 位 - 'a' 轉成數字
- 然後數值為 索引9的值 \* 100 + 索引10的值 \* 10 + 索引11的值 \* 1

```
int hundreds = (Serial_inputString[9] - '0') * 100;
int tens = (Serial_inputString[10] - '0') * 10;
int units = (Serial_inputString[11] - '0' );
servoValue = hundreds + tens + units;
```





# 範例



```
//  
// 資料解析  
//  
void Serial_dataParse()  
{  
    if (!Serial_newLine) // 如果沒有新資料  
        return; // 結束此資料解析  
    Serial.print(">> ");  
    Serial.println(Serial_inputString);  
  
    if (Serial_inputString[1] == 'S')  
        servoCommand_dataParse();  
    else  
        generalCommand_dataParse();  
  
    // 解析完成，清除  
    Serial_newLine = false;  
    Serial_inputString = "";  
}
```



## 範例

```
void servoCommand_dataParse()
{
    char mode = Serial_inputString[7]; // 判斷是方向還是角度指令

    if (mode == 'U')                // UD 設定舵機角度
    {
        // 將字元轉成數字
        int hundreds = (Serial_inputString[9] - '0') * 100;
        int tens = (Serial_inputString[10] - '0') * 10;
        int units = (Serial_inputString[11] - '0');
        wifiServoValue = hundreds + tens + units;
    }
}
```

- 如果是 L 在對索引為 9 的值進行判斷
- 如果是 U 就對索引 9-11 的解析角度

```
else // LR 左右旋轉
{
    char action = Serial_inputString[9];
    switch (action)
    {
        case 'L':
            wifiServoValue -= 10;
            break;

        case 'R':
            wifiServoValue += 10;

            break;

        default:
            break;
    }
}

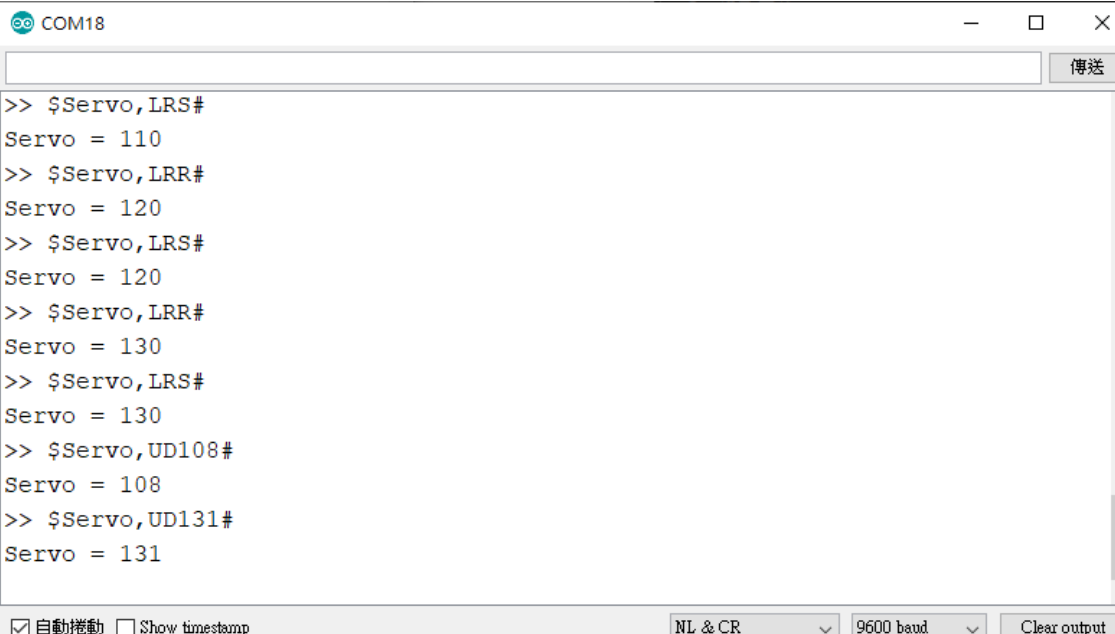
// 限制角度
if (wifiServoValue < 0)
    wifiServoValue = 0;
else if (wifiServoValue > 180)
    wifiServoValue = 180;

wifiServo.write(wifiServoValue);
Serial.print("Servo = ");
Serial.println(wifiServoValue);
}
```



## 實驗 3

- 使用 App 控制 左搖 / 右搖，並控制舵機
- 使用 App 拖動滑動條，並從 序列埠監控視窗查看 解析出來的角度
- 拖動滑動條控制舵機角度



```
>> $Servo,LRS#  
Servo = 110  
>> $Servo,LRR#  
Servo = 120  
>> $Servo,LRS#  
Servo = 120  
>> $Servo,LRR#  
Servo = 130  
>> $Servo,LRS#  
Servo = 130  
>> $Servo,UD108#  
Servo = 108  
>> $Servo,UD131#  
Servo = 131
```



# 模式控制



# 模式控制

## 模式

索引	0	5	6	8
七彩	\$Mode	,	11	#
避障	\$Mode	,	21	#
循線	\$Mode	,	31	#
關閉	\$Mode	,	00	#

- 使用前面的方法判斷索引 1 是否為 'M'
- 在依照索引 6 去個別設定模式。
- 假設循跡的程式進入點在 loop(); 內，只要前面加個判斷式，當模式為 3 時就執行循跡副程式
- 利用先前的方式去製作解析的函式



# Appendix



# 傳輸協定解析 (for 6WD)

通用協定		運動狀態		旋轉		鳴笛		加減速		左右搖		唱歌		點燈		滅火		舵機復位	
位	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
停止	\$	0	,	0	,	0	,	0	,	0	,	0	,	0	,	0	,	0	#
前進	\$	1	,	0	,	0	,	0	,	0	,	0	,	0	,	0	,	0	#
後退	\$	2	,	0	,	0	,	0	,	0	,	0	,	0	,	0	,	0	#
左轉	\$	3	,	0	,	0	,	0	,	0	,	0	,	0	,	0	,	0	#
右轉	\$	4	,	0	,	0	,	0	,	0	,	0	,	0	,	0	,	0	#
左旋轉	\$	0	,	1	,	0	,	0	,	0	,	0	,	0	,	0	,	0	#
右旋轉	\$	0	,	2	,	0	,	0	,	0	,	0	,	0	,	0	,	0	#
鳴笛	\$	0	,	0	,	1	,	0	,	0	,	0	,	0	,	0	,	0	#
加速	\$	0	,	0	,	0	,	1	,	0	,	0	,	0	,	0	,	0	#
減速	\$	0	,	0	,	0	,	2	,	0	,	0	,	0	,	0	,	0	#
舵機復位	\$	0	,	0	,	0	,	0	,	0	,	0	,	0	,	0	,	1	#

