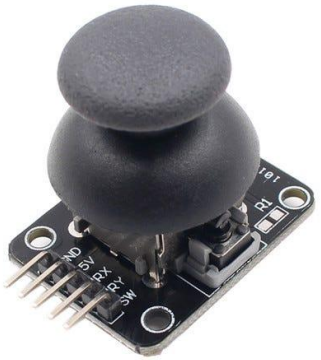
Joystick Mapping
(控制杆映射)

Joystick 操縱桿

- 操縱桿是一種常見的人機交互裝置
- 當使用者移動桿體時,感測器會檢測桿體在X和Y軸上的偏移量
- 這些偏移量訊號會被轉換成數位訊號

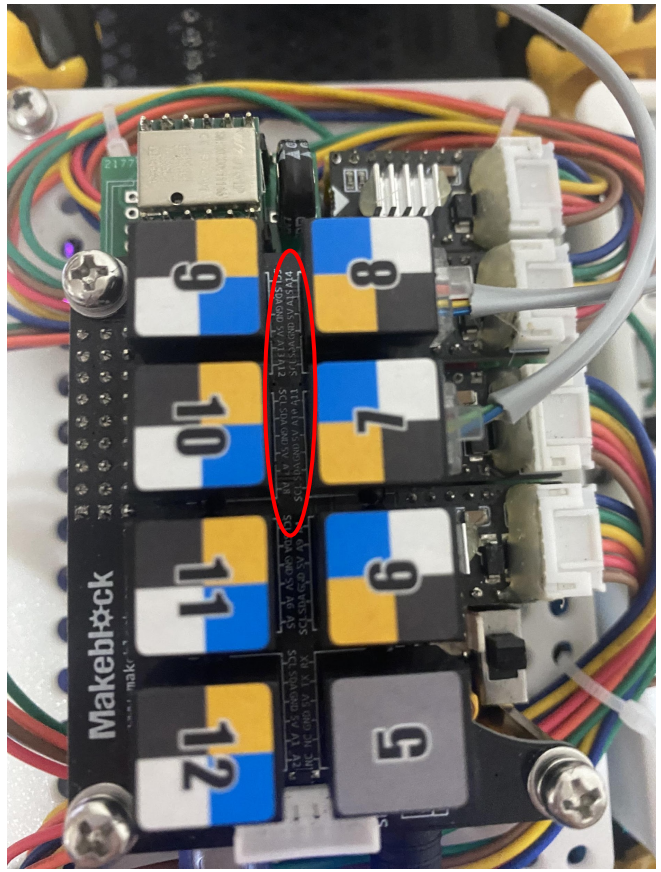


讀取Joystick資料

讀取資料

- 連接至7/8
- 找到屬於該連接口的Pinout
- 例如: SCL SDA GND 5V A10 A11

- A10等於該控制桿的X軸資料
- A11等於該控制桿的Y軸資料



原理

- Step 1: 使用analogRead讀取資料
- Step 2: 在Serial Monitor中展示資料

digitalRead → high / low

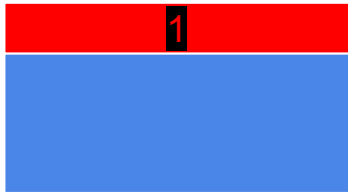
analogRead → range

讀取資料

Controller.ino

```
1   #define joy1X A15
2   #define joy1Y A14
3   #define joy2X A10
4   #define joy2Y A11
5
6   int x1Value = 0;
7   int y1Value = 0;
8   int x2Value = 0;
9   int y2Value = 0;
10
11
12  void setup() {
13    Serial.begin(9600);
14  }
```

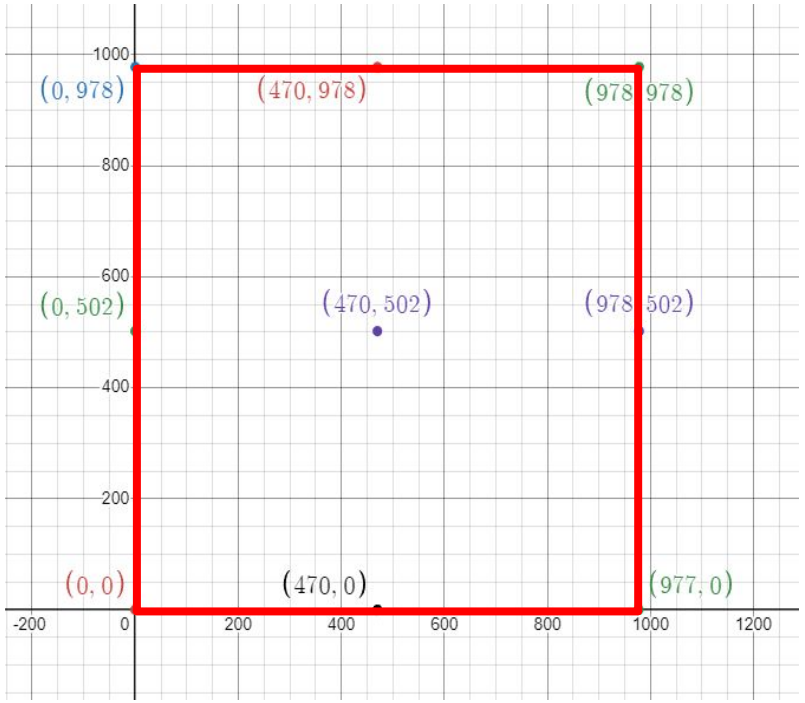
```
16  void loop() {
17
18    x1Value = analogRead(joy1X);
19    y1Value = analogRead(joy1Y);
20    x2Value = analogRead(joy2X);
21    y2Value = analogRead(joy2Y);
22
23    Serial.print(x1Value);
24    Serial.print("\t");
25    Serial.print(y1Value);
26    Serial.print("\t");
27    Serial.print(x2Value);
28    Serial.print("\t");
29    Serial.println(y2Value);
30
31    delay(100);
32  }
33
```




```
470     503     481     476
471     503     481     476
470     503     481     476
470     503     481     476
471     503     481     476
470     503     481     476
470     503     481     476
470     503     481     476
471     503     481     476
470     503     481     476
470     503     481     476
470     503     482     476
```

讀取資料

- 記錄你的控制杆的閒置/極限數值
- 方便於之後寫條件



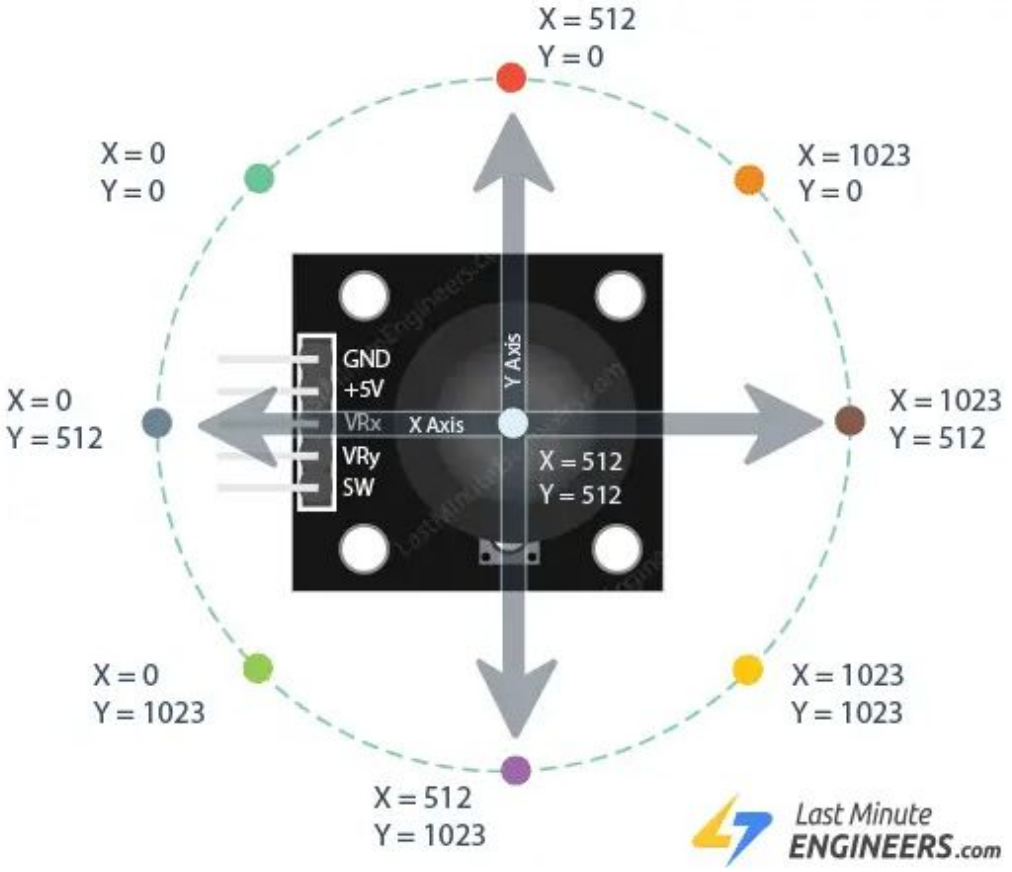
470	503	481	476
471	503	481	476
470	503	481	476
470	503	481	476
471	503	481	476
470	503	481	476
470	503	481	476
470	503	481	476
471	503	481	476
470	503	481	476
470	503	481	476
470	503	482	476

判斷方向

- 使用x , y坐標系統來判斷方向
- 例如以 X = 512 , Y = 512作為中心點

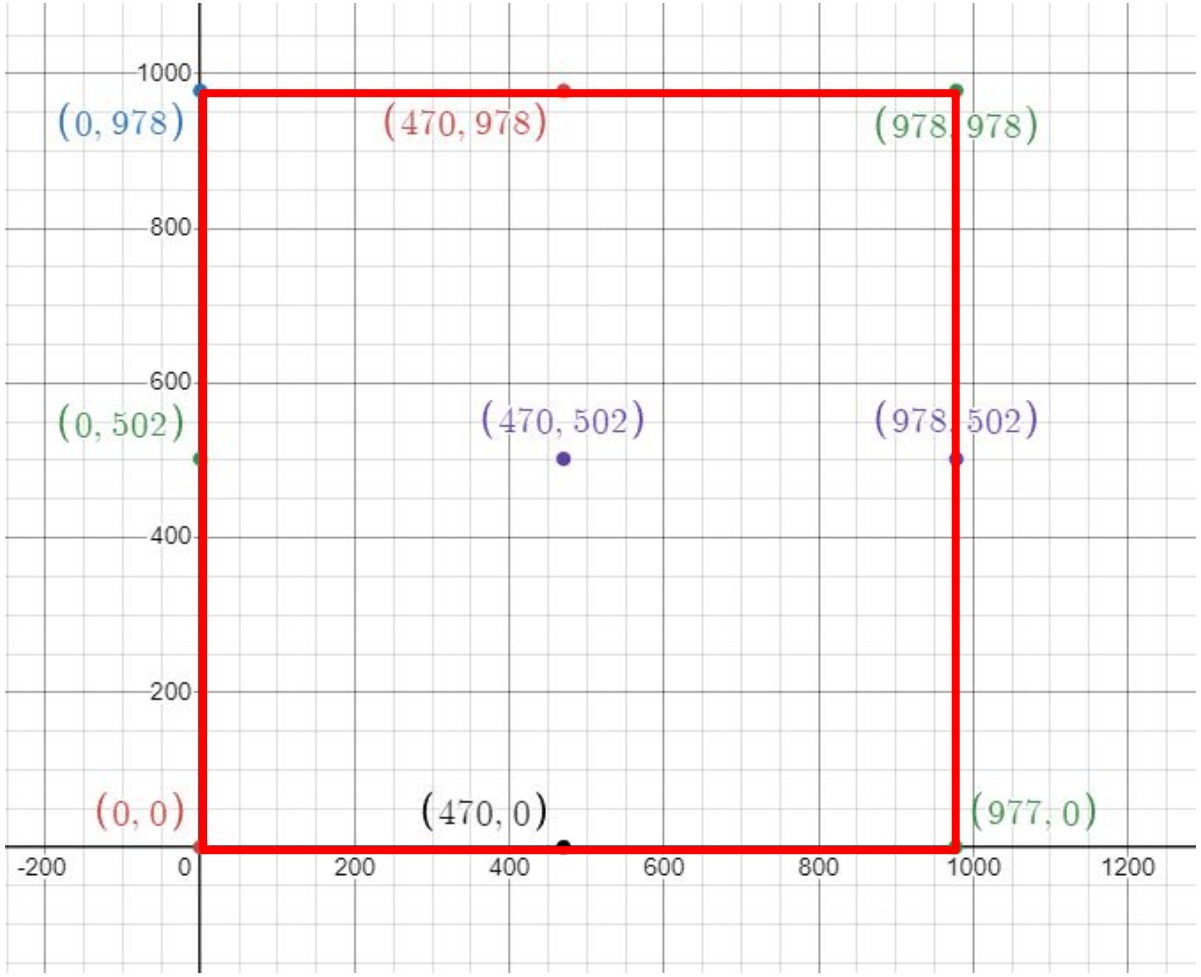
- 如果X > 512 就是向右移動
- 如果X < 512 就是向左移動
- 以此類推

- 但坐標 +/- 會因控制杆而異



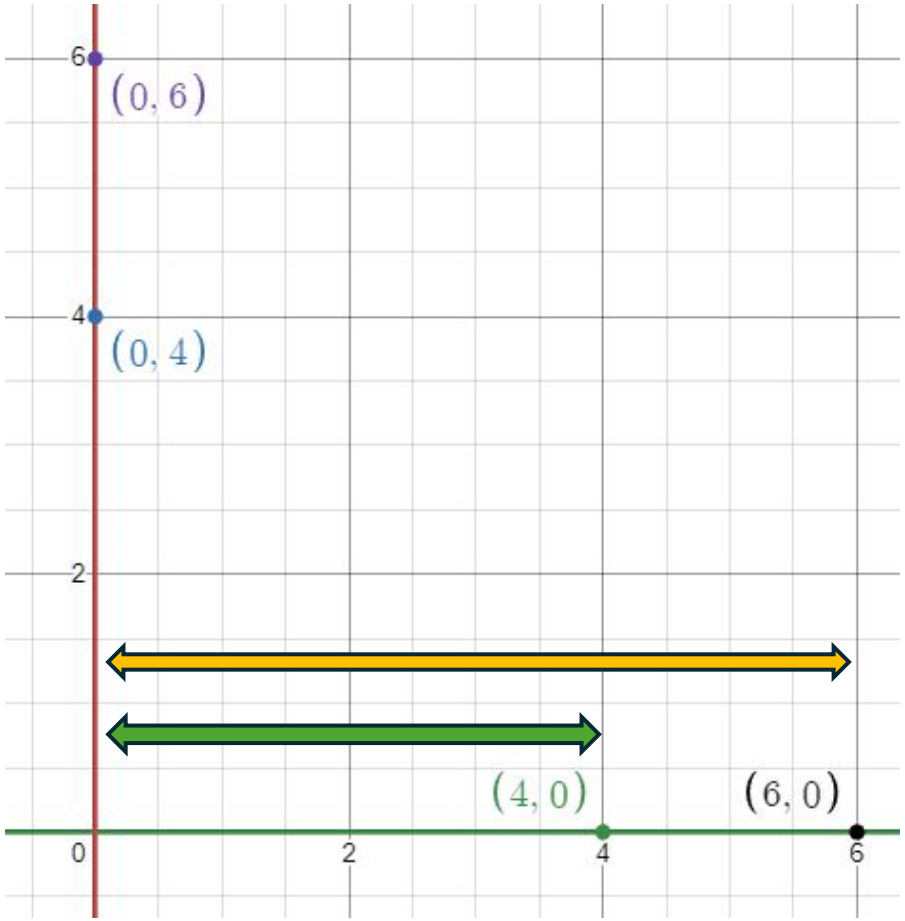
判斷方向

- 每個控制桿的範圍都可能有些許的誤差
- 極限數值/範圍 + 大概 30 / 或將極限範圍設為(1024, 1024)
- 應付控制杆產生超出範圍的數值



判斷速度

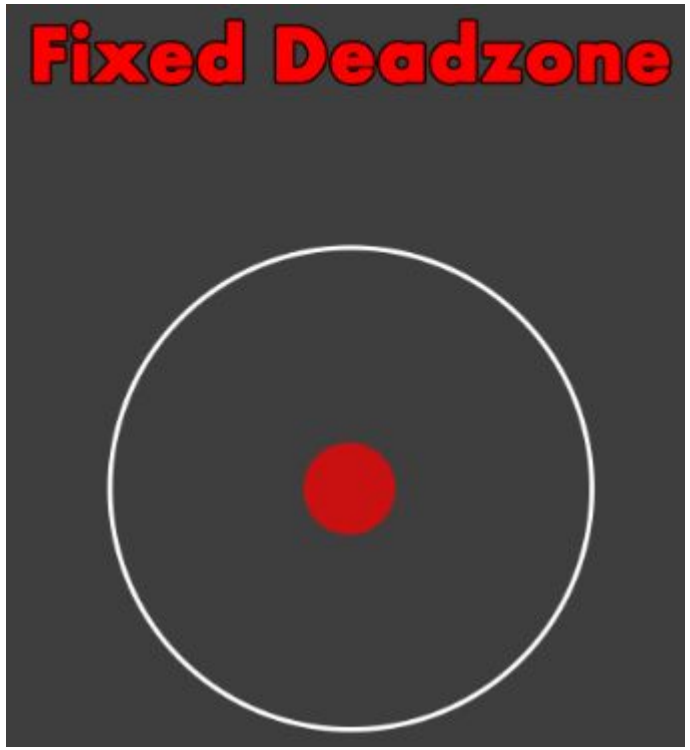
- 垂直 / 水平綫斜率為 無限 / 0
- 利用垂直 / 水平上的一點 和 最大範圍來計算
- 配合map能夠分配一個範圍至速度範圍



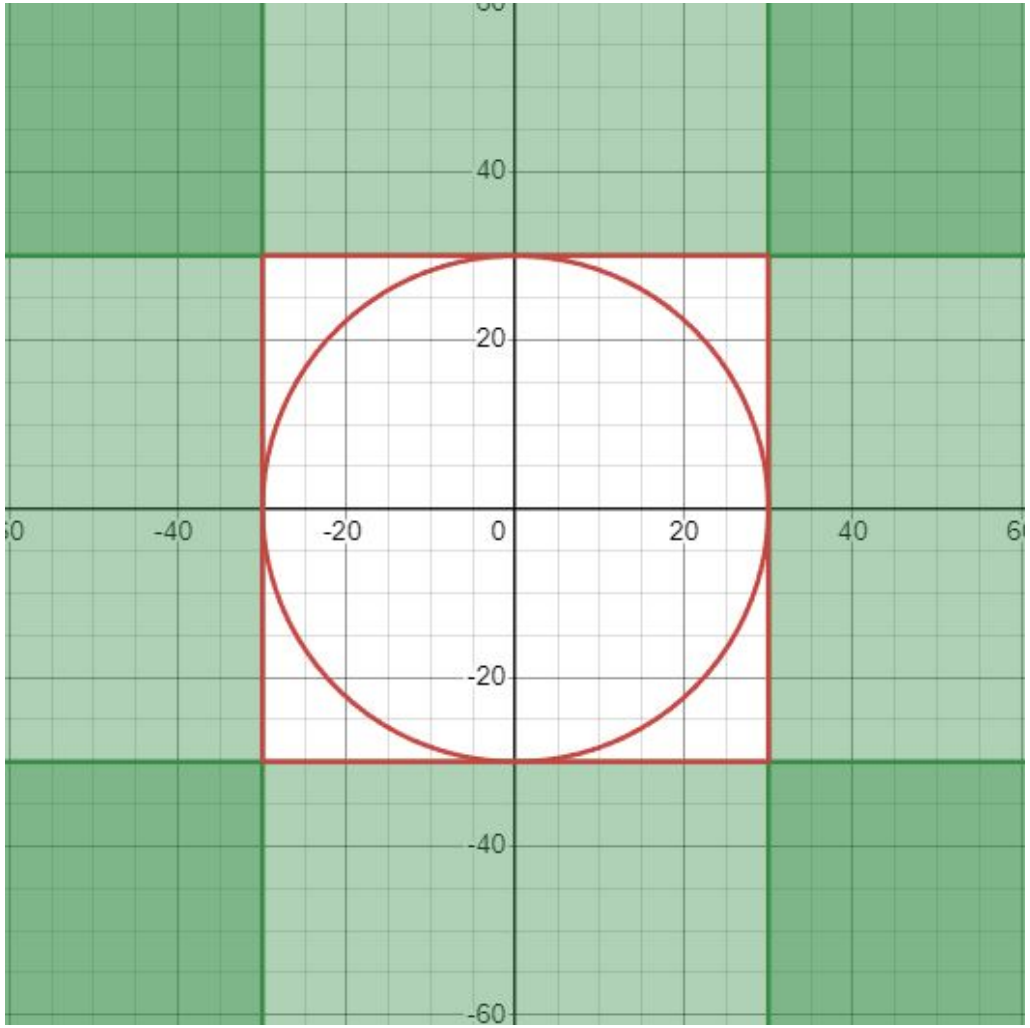
控制杆死區

- 微小的誤差或噪聲會產生一些不必要的訊號變化
- 導致不穩定的控制

- 死區範圍內的控制杆變化將被忽略，不會產生任何輸出。

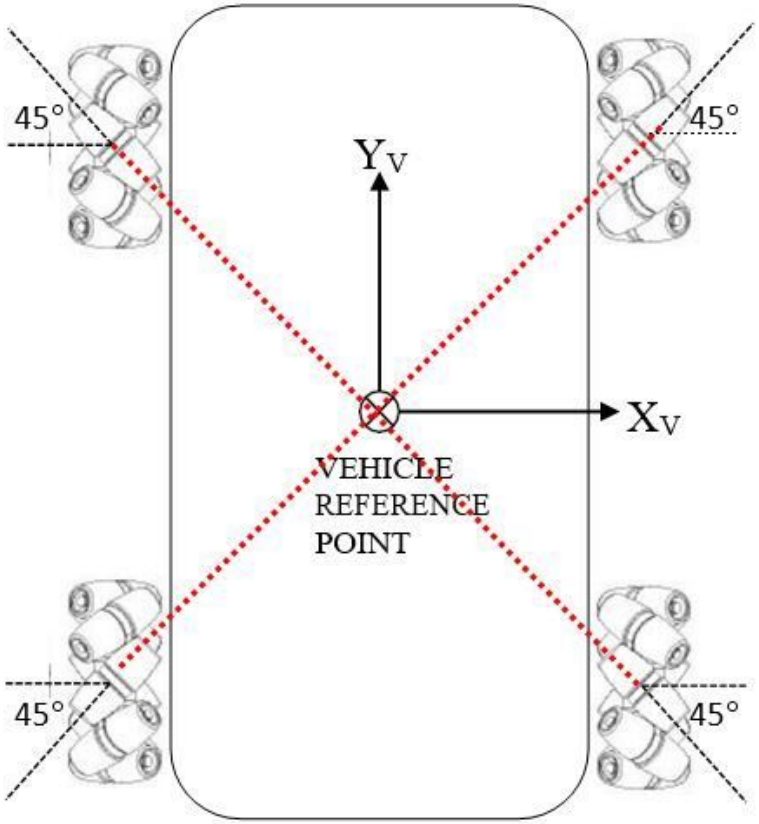


控制杆死區



全向輪的分別

- 車輪的外環安裝了與軸心成45度角排列的橡膠輥子
- 轉動時與地面接觸的摩擦力會產生與輪軸呈45度的反推力。

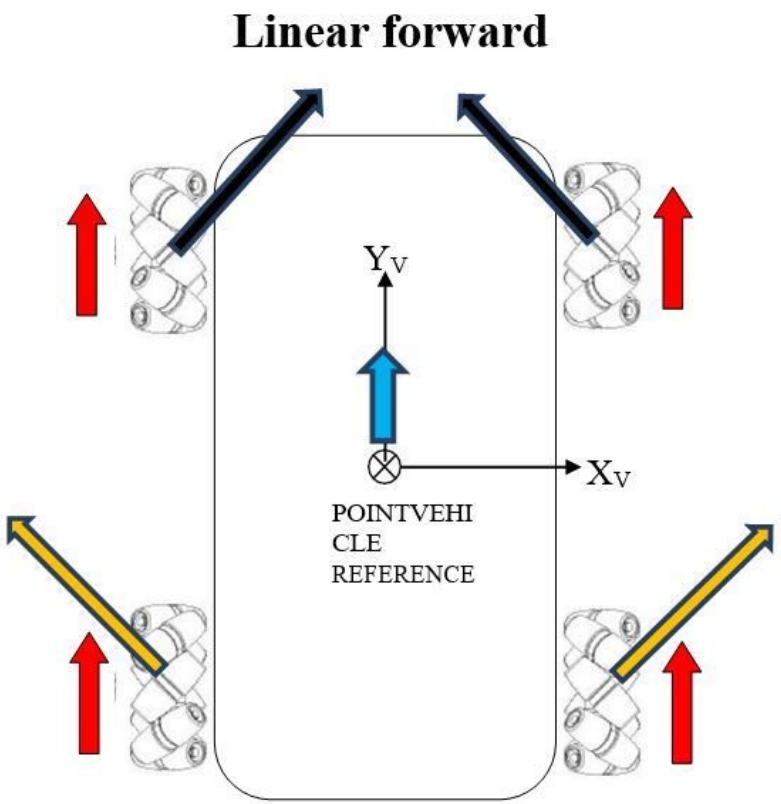


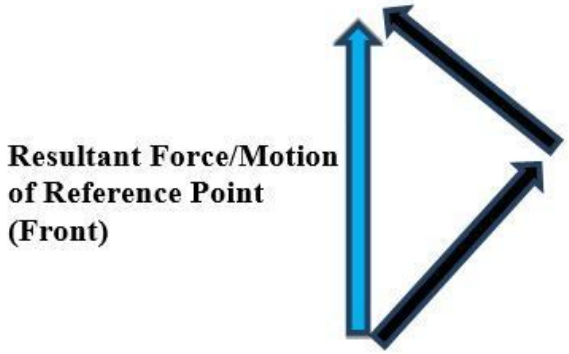
全向輪

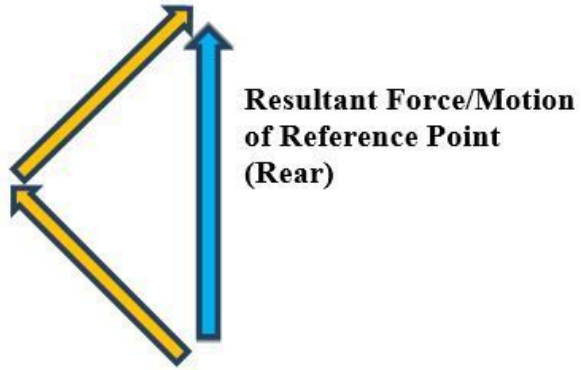
紅色: 輪子向量的大小和方向

黑色/黃色: 輪子移動時橡膠輥子產生的摩擦力/反作用力向量

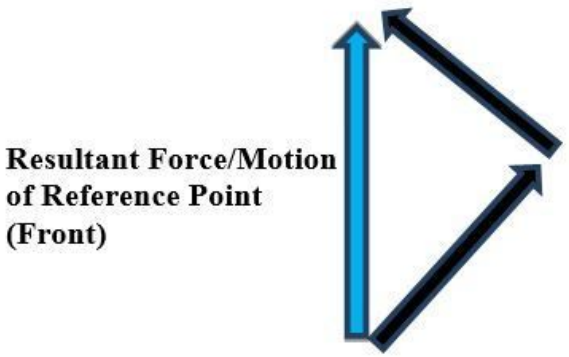
藍色: 根據參考點而得出的動力向量

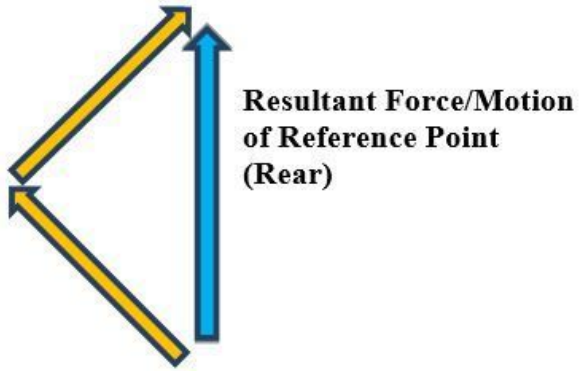


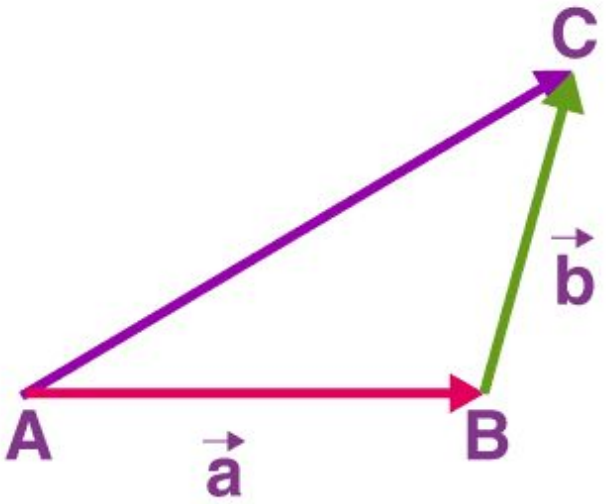




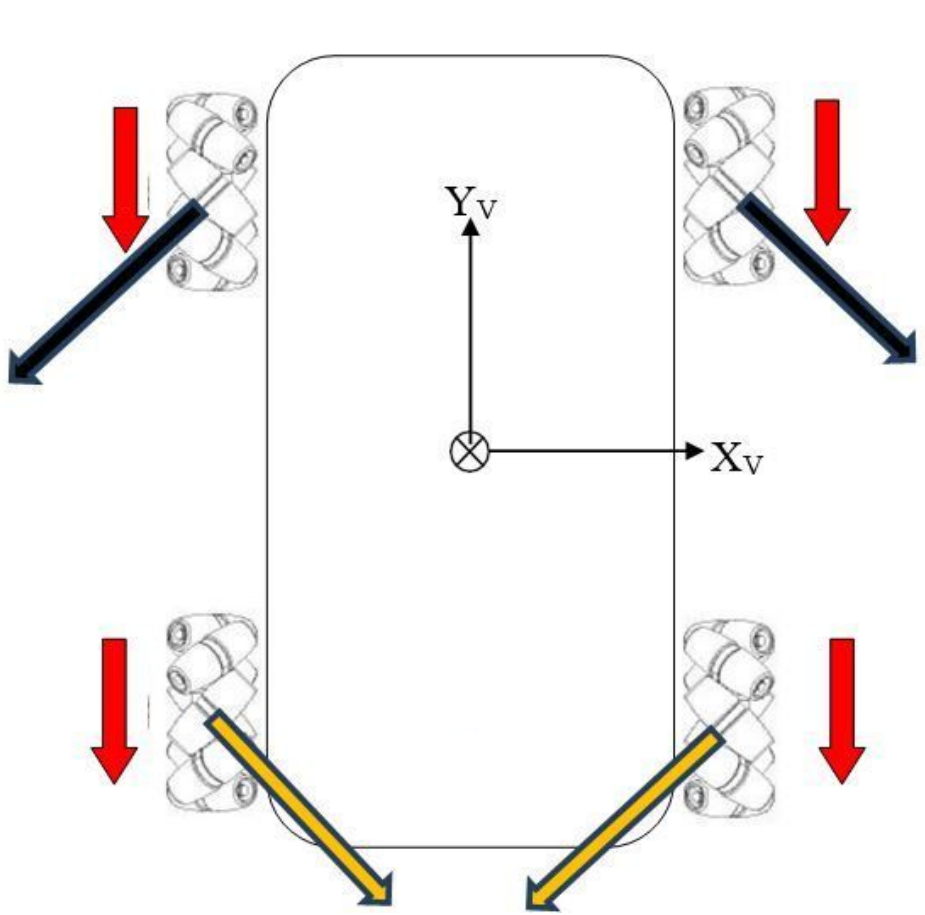
Vector(向量)運算

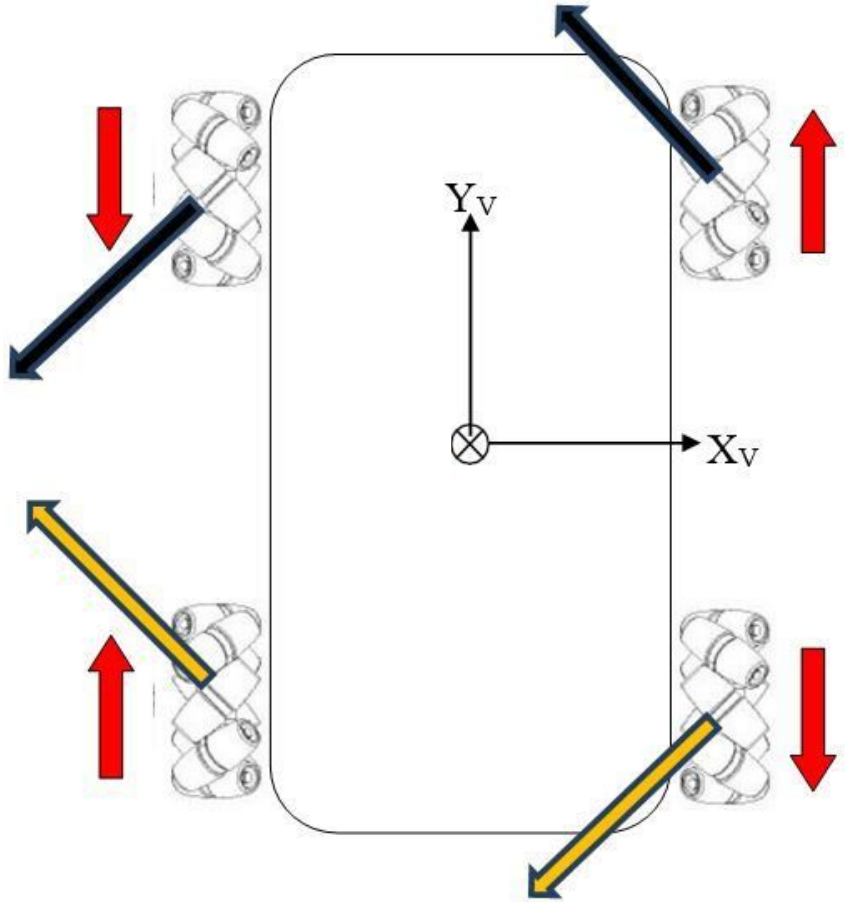

Resultant Force/Motion
of Reference Point
(Front)


Resultant Force/Motion
of Reference Point
(Rear)

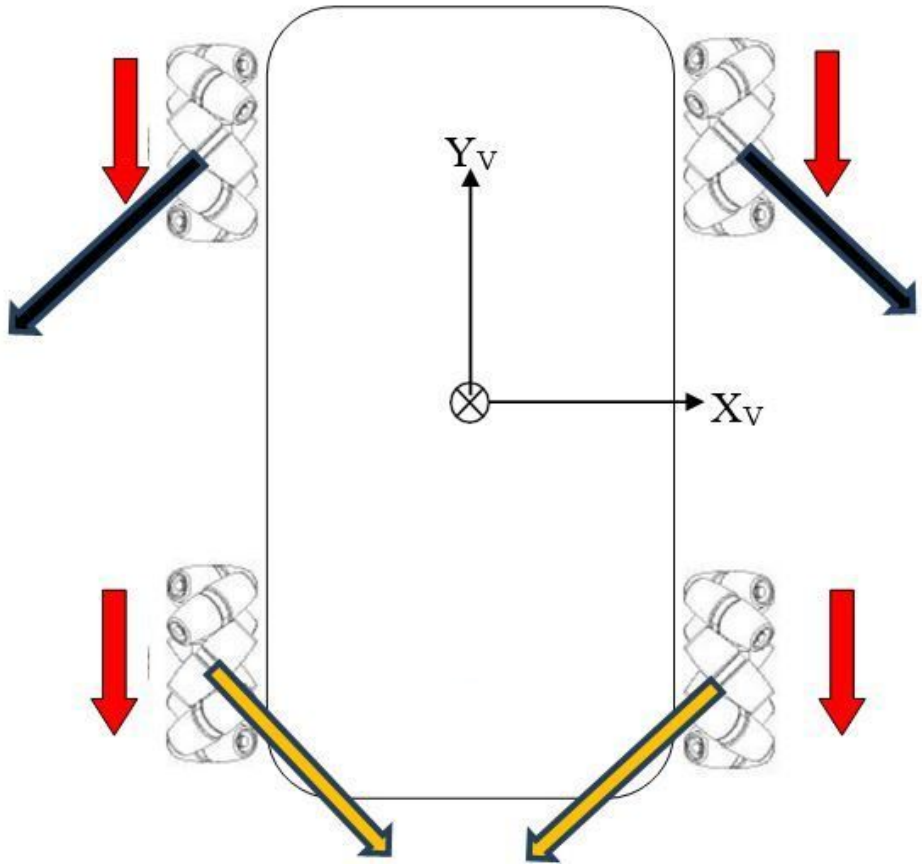

C
b
A
a
B

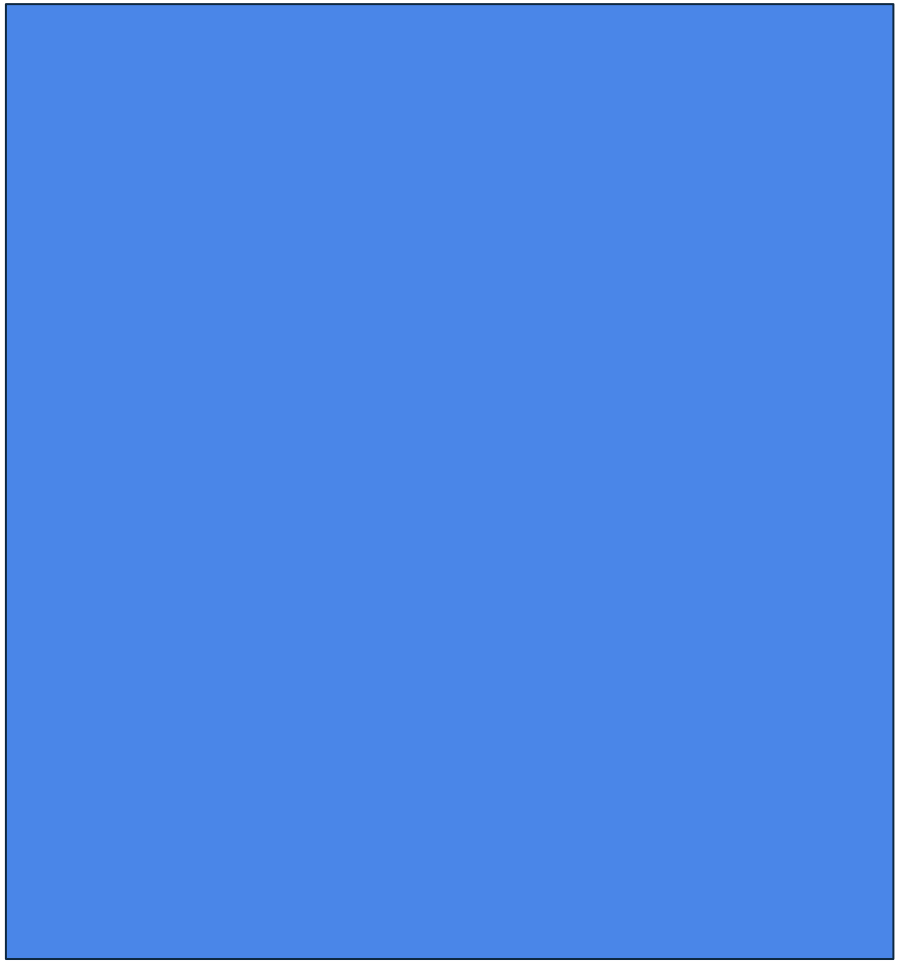
Vector(向量)運算



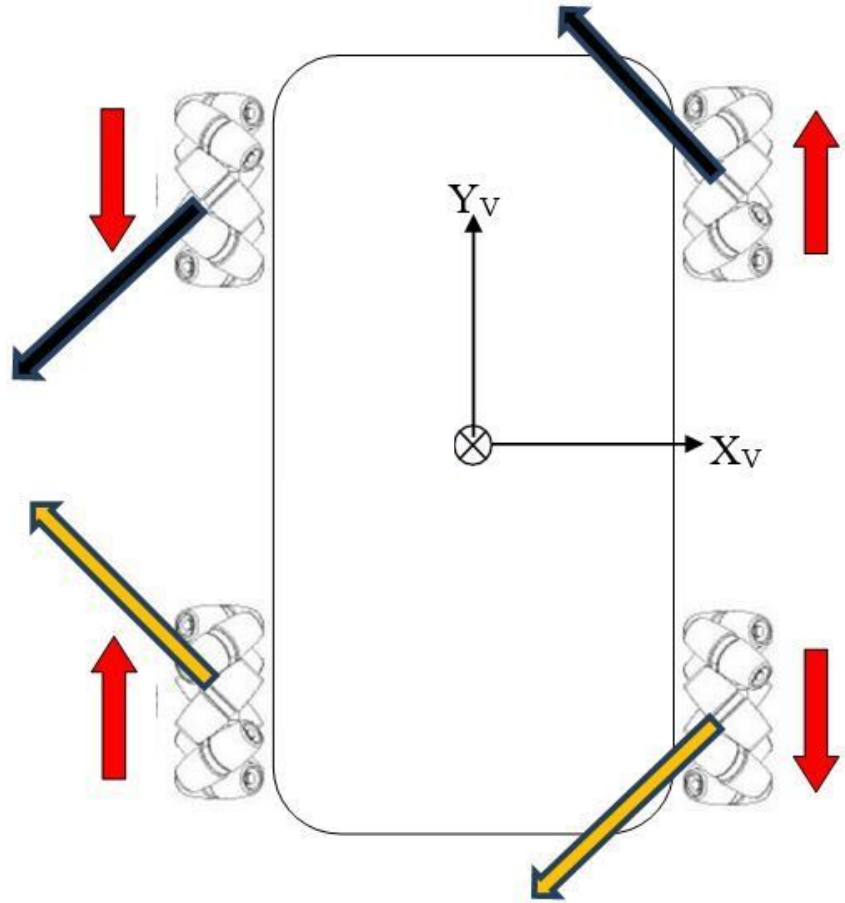


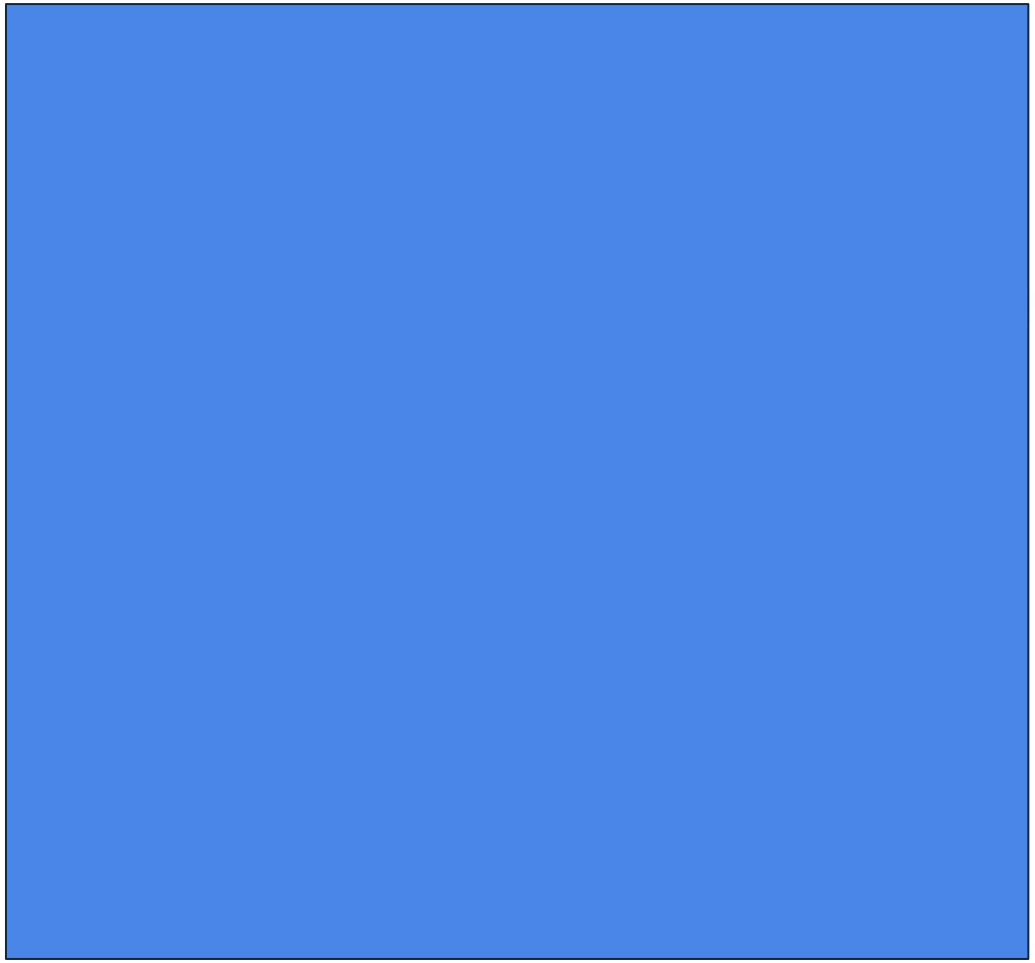
Vector(向量)運算



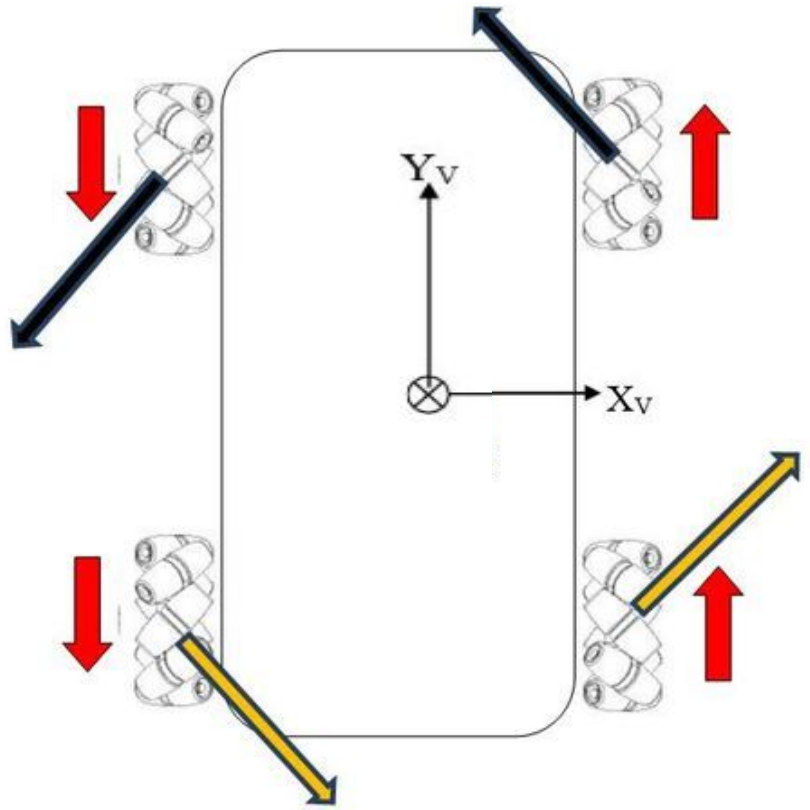


Vector(向量)運算

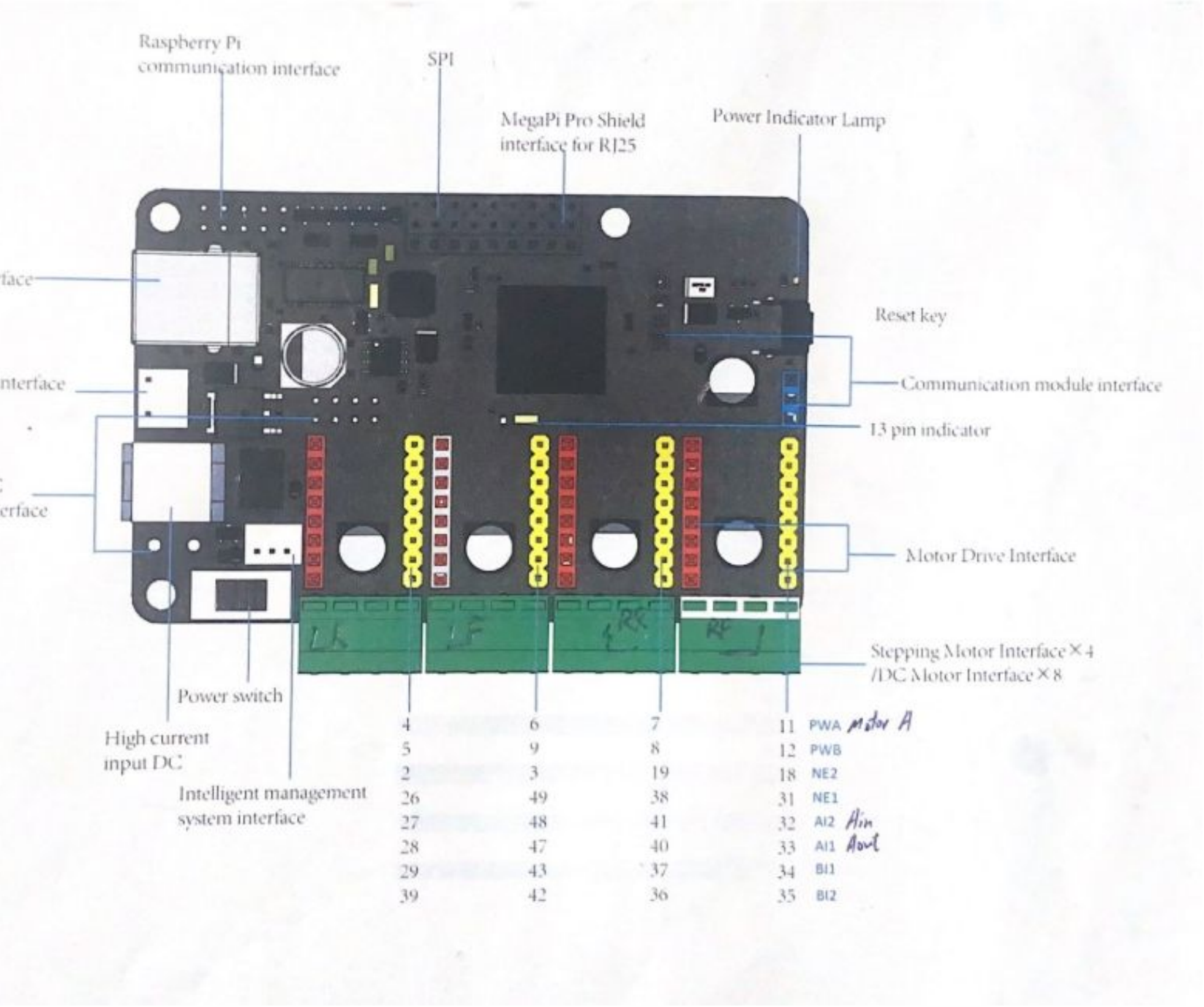




Vector(向量)運算



控制Mecanum wheel(全向輪)



- PWB --> 控制**速度**
  |(-255, 255)| → (0, 255)

(analog)

- BI 1 + BI 2 --> 控制**方向**
  (HIGH(1)/LOW(0))

(digital)

```
#define LR_BI2 39

pinMode(LR_BI2, OUTPUT);

digitalWrite(RF_BI2, LOW);
```

目標: 確認每個車輪都能正常轉動。

控制編程

RR = Right Rear (右 + 後)

RF = Right Front (右 + 前)

LR = Left Rear (左 + 後)

LF = Left Front (左 + 前)

```
#define RF_BI1 34
#define RF_BI2 35
#define RF_PWM 12

#define RR_BI1 37
#define RR_BI2 36
#define RR_PWM 8

#define LF_BI2 42
#define LF_BI1 43
#define LF_PWM 9

#define LR_BI2 39
#define LR_BI1 29
#define LR_PWM 5

int speed_slow = 100;
int speed_fast = 250;
```

```
void setup() {

  pinMode(RR_PWM,  );
  pinMode(LF_PWM,  );
  pinMode(LR_PWM,  );
  pinMode(RF_PWM,  );

  pinMode(RF_BI1,  );
  pinMode(RF_BI2,  );
  pinMode(RR_BI1,  );
  pinMode(RR_BI2,  );
  pinMode(LF_BI1,  );
  pinMode(LF_BI2,  );
  pinMode(LR_BI1,  );
  pinMode(LR_BI2,  );

}
```

1

1

```
void loop() {
  //RF
  digitalWrite(  2  ,  3  );
  digitalWrite(  4  ,  5  );
  analogWrite(  6  , speed_slow);
  delay(2000);
  analogWrite(  6  , 0);

  //RR
  digitalWrite(RR_BI1, HIGH);
  digitalWrite(RR_BI2, LOW);
  analogWrite(RR_PWM, speed_slow);
  delay(2000);
  analogWrite(RR_PWM, 0);

  //LF
  //LR
}
```

利用控制杆控制全向輪(前進/後退) | 方法1

分辨條件

```
if((x1Value >= (middle_x - deadzone) && x1Value <= (middle_x + deadzone) && (y1Value >= middle_y + deadzone))){
```

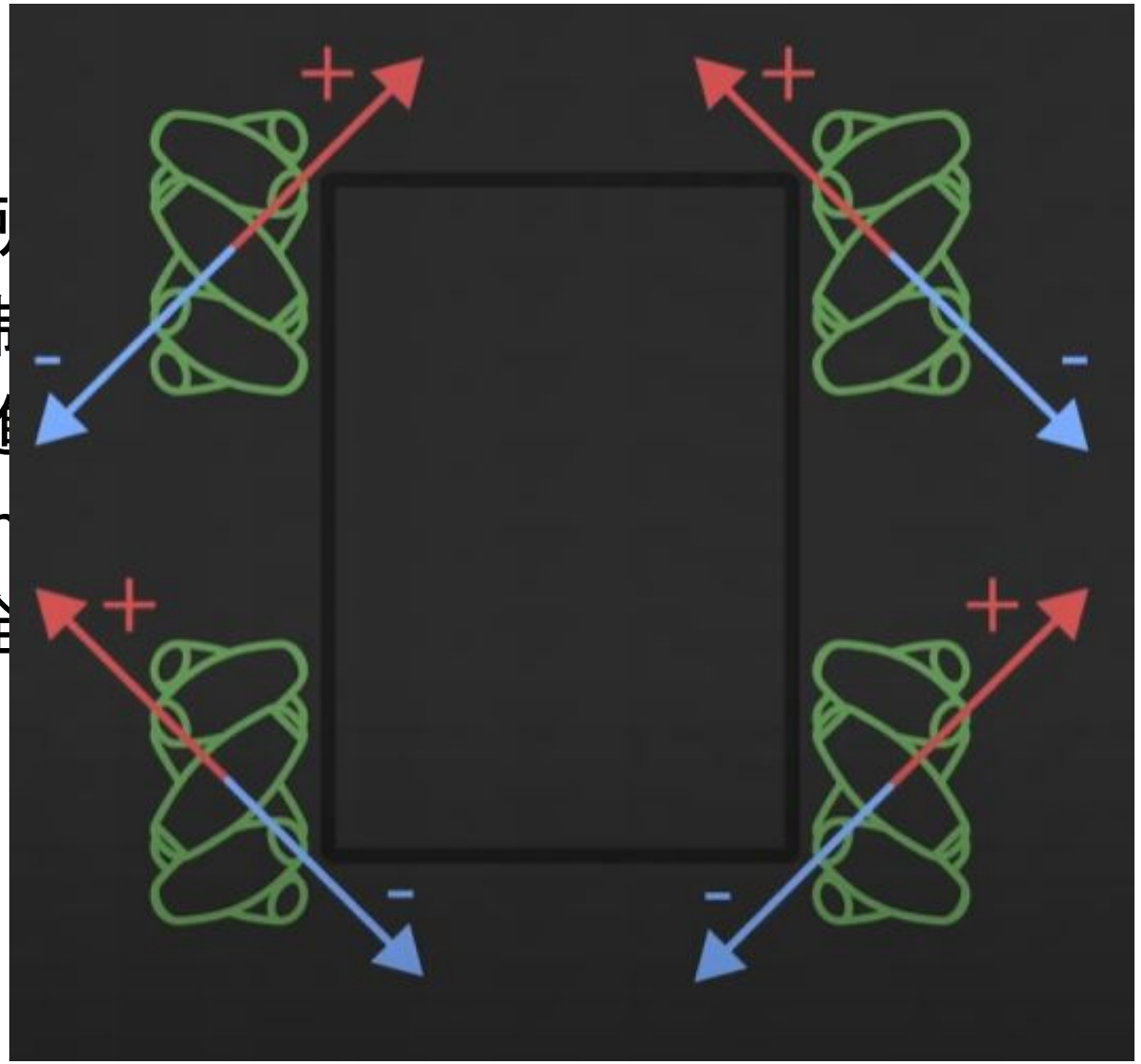
前進條件

- X1/Y1為控制杆輸出, middle_x/middle_y 為控制杆中心點
- (若x1 >= 中心點 - 死區) + (x1 <= 中心點 + 死區) ==> x1在死區內
- (若y1 >= 中心點 + 死區) ==> y1在死區外

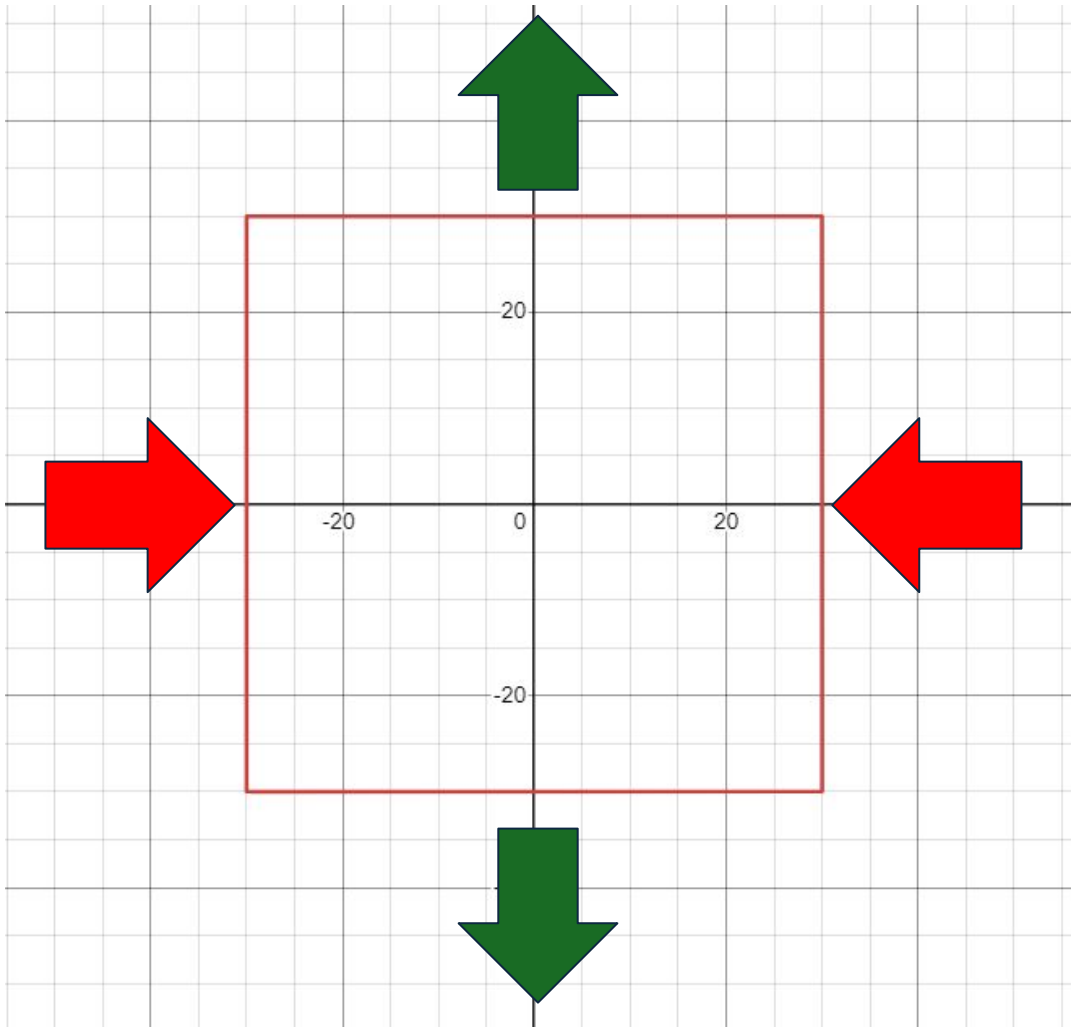
➡ 僅有y軸輸出(前進)

原理

- Step 0: 先定義
- Step 1: 收取控
- Step 2: 判斷前
- Step 2.1: 利用m
- Step 2.2: 控制全



原理



前進/後退編程

```
#define RF_BI1 34
#define RF_BI2 35
#define RF_PWM 12

#define RR_BI1 37
#define RR_BI2 36
#define RR_PWM 8

#define LF_BI1 43
#define LF_BI2 42
#define LF_PWM 9

#define LR_BI1 29
#define LR_BI2 39
#define LR_PWM 5

#define joy1X A9
#define joy1Y A4
#define joy2X A10
#define joy2Y A11
```

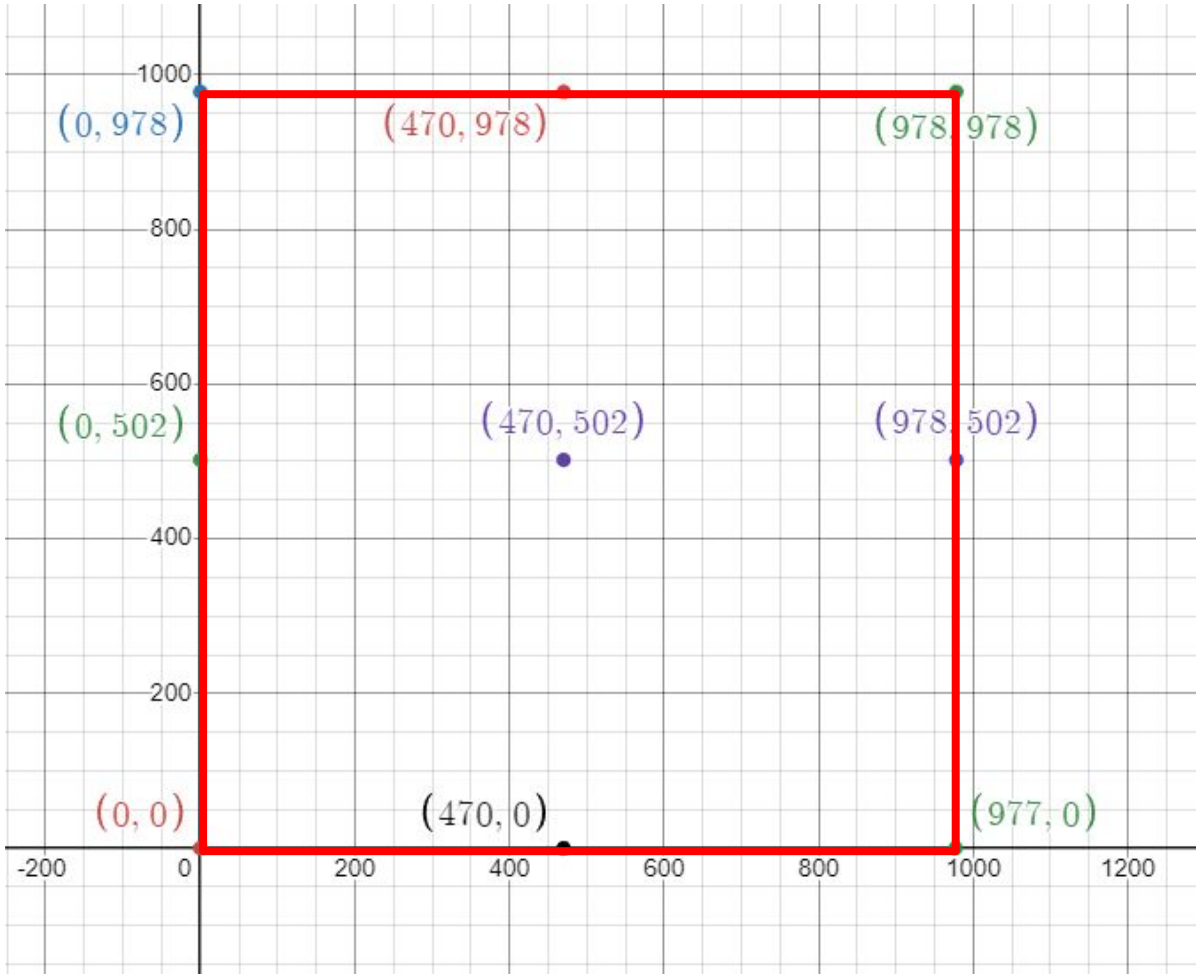
```
int x1Value = 0;
int y1Value = 0;
int x2Value = 0;
int y2Value = 0;
int deadzone =
int middle_x =
int middle_y =
```

```
void setup() {

  pinMode(RF_BI1, OUTPUT);
  pinMode(RF_BI2, OUTPUT);
  pinMode(RF_PWM, OUTPUT);

  pinMode(RR_BI1, OUTPUT);
  pinMode(RR_BI2, OUTPUT);
  pinMode(RR_PWM, OUTPUT);

  pinMode(LF_BI1, OUTPUT);
  pinMode(LF_BI2, OUTPUT);
  pinMode(LF_PWM, OUTPUT);

  pinMode(LR_BI1, OUTPUT);
  pinMode(LR_BI2, OUTPUT);
  pinMode(LR_PWM, OUTPUT);
  Serial.begin(9600);

}
```

```
void controlRF(int BI1, int BI2, int speed){
  摩打控制
}

void controlRR(int BI1, int BI2, int speed){

}

void controlLF(int BI1, int BI2, int speed){

}

void controlLR(int BI1, int BI2, int speed){

}
```

判斷方向

- 每個控制桿的範圍都可能有些許的誤差
- 以你手上的控制杆的極限數值/範圍為準



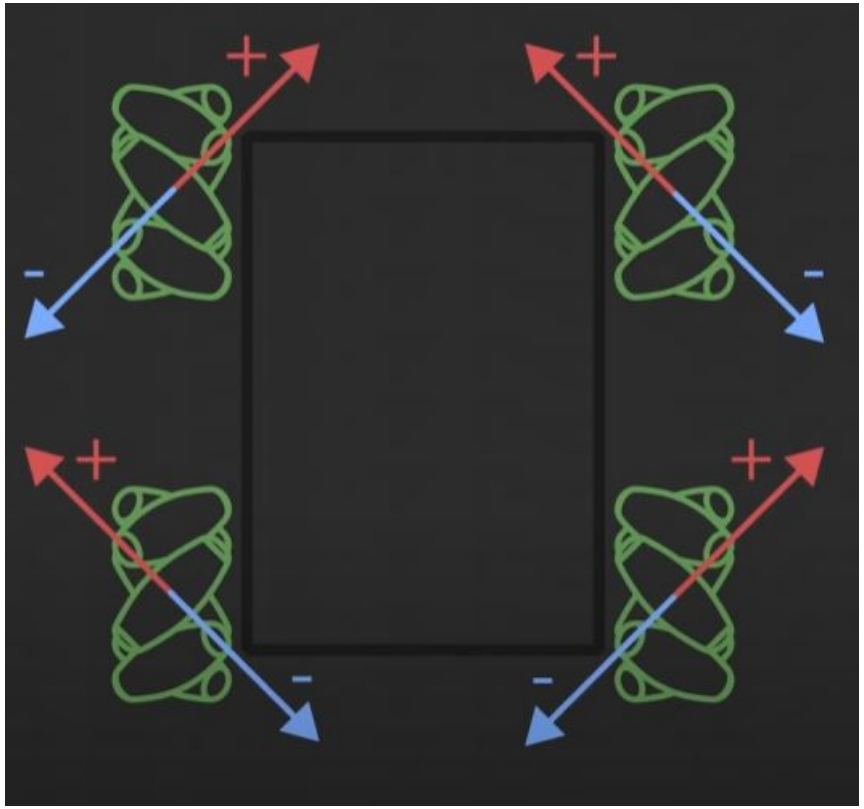
前進/後退編程

```
void loop() {

  x1Value = analogRead(joy1X);
  y1Value = analogRead(joy1Y);
  x2Value = analogRead(joy2X);
  y2Value = analogRead(joy2Y);

  if((x1Value >= (middle_x - deadzone) && x1Value <= (middle_x + deadzone) && (y1Value >= middle_y + deadzone))){
    //straight forward slope = inf
    int yaxis = abs(map(y1Value, 0, 978, -255, 255));
    controlRF(1, 0, yaxis);
    controlRR(1, 0, yaxis);
    controlLF(0, 1, yaxis);
    controlLR(0, 1, yaxis);
    Serial.println("forward");
  }
  else if((x1Value >= (middle_x - deadzone) && x1Value <= (middle_x + deadzone) && (y1Value <= middle_y - deadzone))){
    //backwards slope = inf
    int yaxis = abs(map(y1Value, 0, 978, -255, 255));
    controlRF(0, 1, yaxis);
    controlRR(0, 1, yaxis);
    controlLF(1, 0, yaxis);
    controlLR(1, 0, yaxis);
    Serial.println("backward");
  }
```

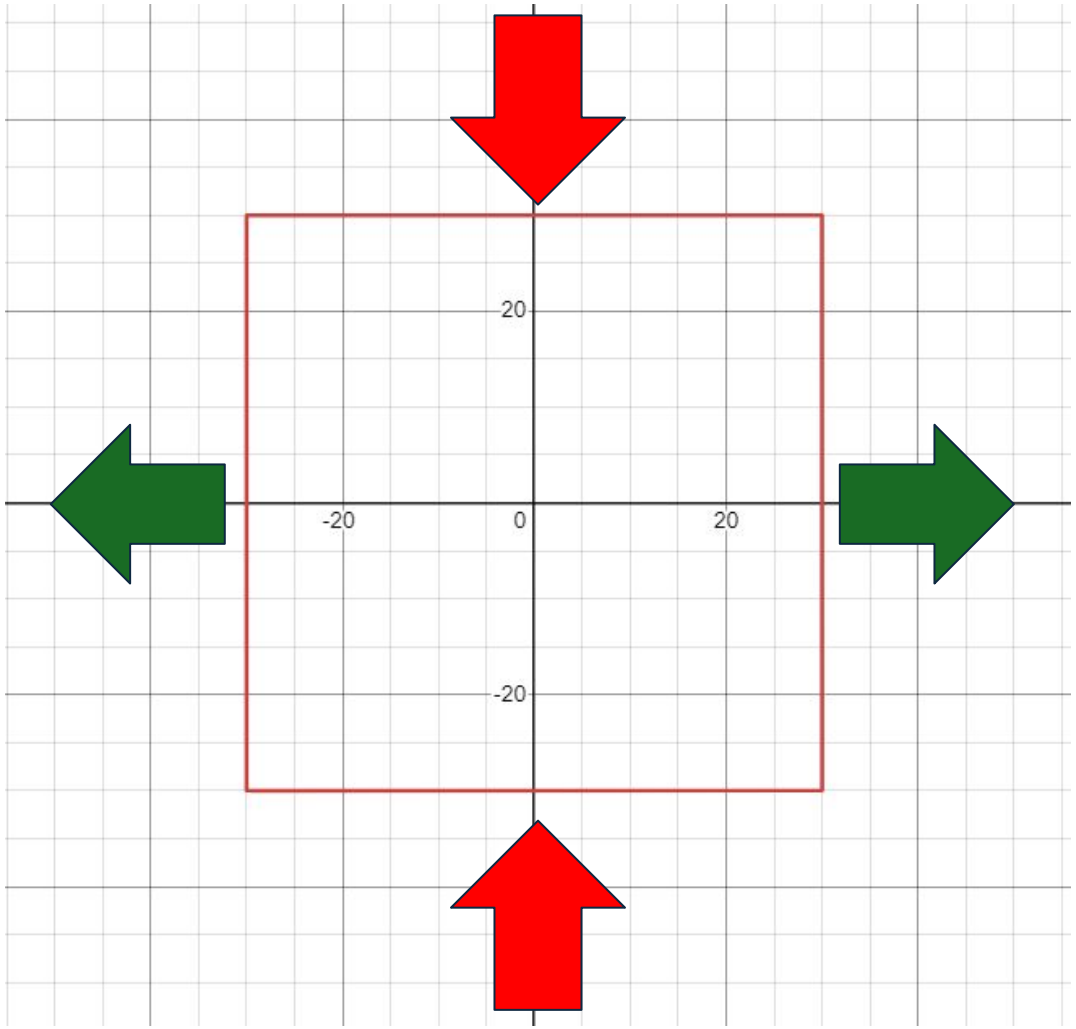
利用控制杆控制全向輪
(向左/向右)

原理

- Step 0: 先定義所有Pin位置和名稱，再宣告常用變量
- Step 1: 收取控制杆的資料
- Step 2: 判斷左/右的條件(分辨方向)
- Step 2.1: 利用map將長度換算速度 (PWB)
- Step 2.2: 控制全向輪(BI 1 + BI 2)



原理

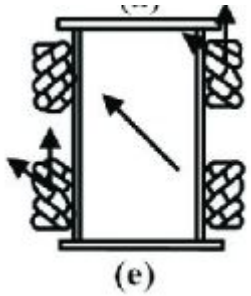


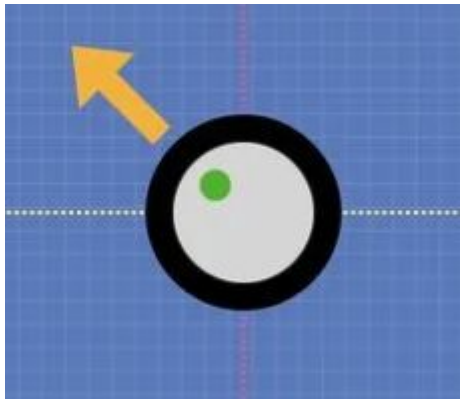
左/右編程

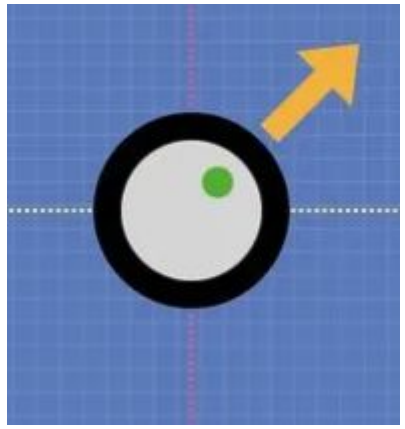
```
else if((x1Value >= middle_x + deadzone) && (                                                                    ){
  //sideways RIGHT slope = 0
  int xaxis =    (map(                                  );
  controlRF(                 );
  controlRR(                 );
  controlLF(                 );
  controlLR(                 );
  Serial.println("sidewaysRIGHT");
}
else if(                                                                                                         ){
  //sideways LEFT slope = 0
  int xaxis =    (map(                                  );
  controlRF(                 );
  controlRR(                 );
  controlLF(                 );
  controlLR(                 );
  Serial.println("sidewaysLEFT");
}
```

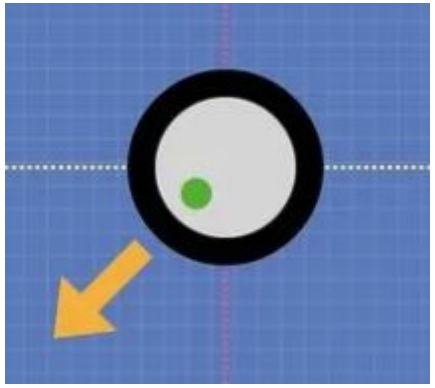
利用控制杆控制全向輪
(對角/Diagonal)

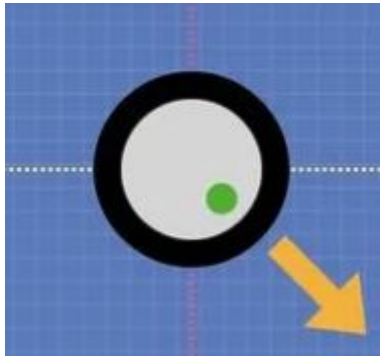
對角



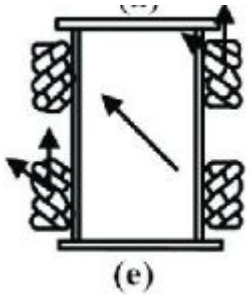


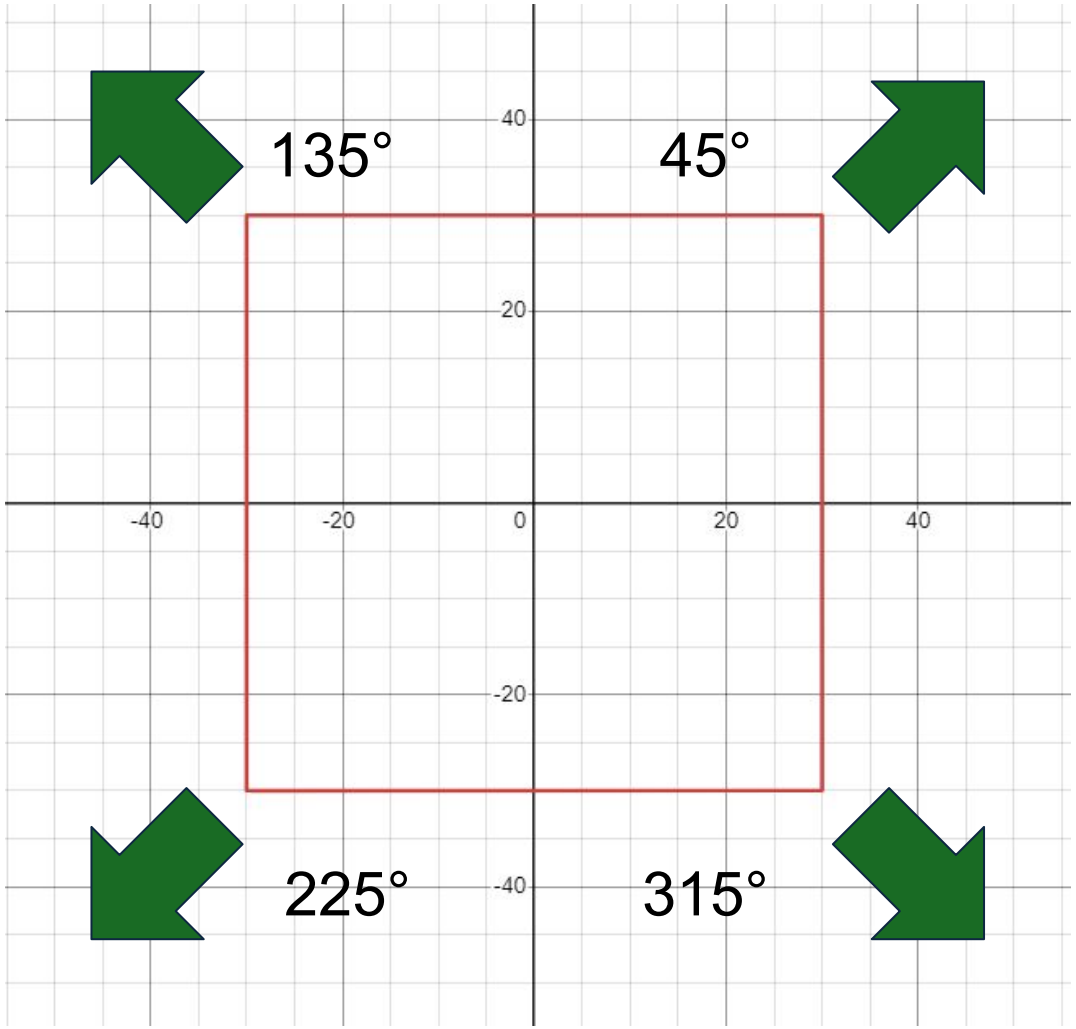




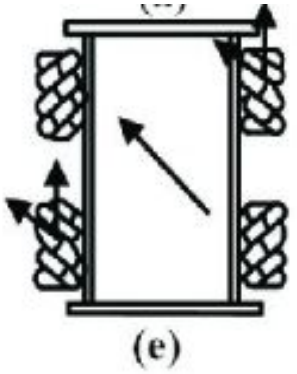


原理

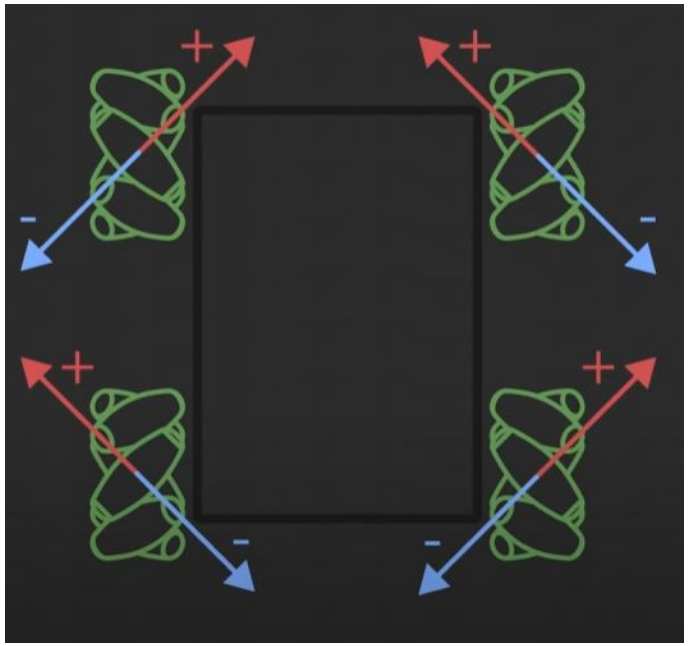




原理



- Step 0: 先定義所有Pin位置和名稱，再宣告常用變量
- Step 1: 收取控制杆的資料
- Step 2: 判斷4個對角的條件(分辨方向)
- Step 2.1: 利用map將長度換算速度 (PWB)
- Step 2.2: 控制全向輪(BI 1 + BI 2)



對角編程

```
else if((x1Value >= (middle_x + deadzone)) && (y1Value >= (middle_y + deadzone))){
  //diagonal 45 degrees
  double current_length = sqrt(
  int speed = abs(map(
  Serial.println("Diagon 45");
}
else if(
  //diagonal 135 degrees
  double current_length = sqrt(
  int speed = abs(map
  Serial.println("Diagon 135");
}
```

```
else if(
  //diagonal 225 degress
  double current_length = sqrt(
  int speed = abs(map(
  Serial.println("Diagon 225");
}
else if(
  //diagonal 315 degress
  double current_length = sqrt
  int speed = abs(map
  Serial.println("Diagon 315");
}
```

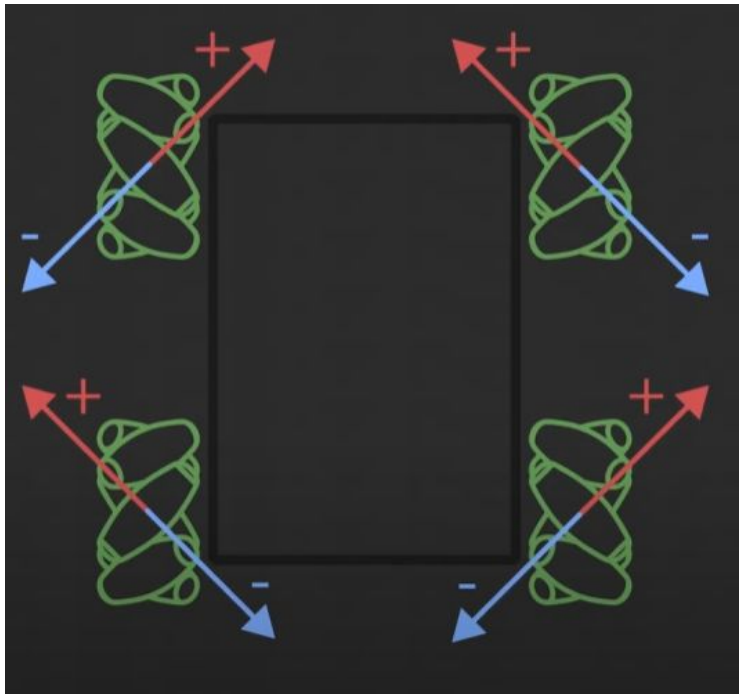
額外else{}

```
else{
    controlLF(0, 0, 0);
    controlRR(0, 0, 0);
    controlRF(0, 0, 0);
    controlLR(0, 0, 0);
    Serial.println("Idle");
}
```

利用控制杆(2)控制全向輪
(旋轉(順時針/逆時針))

原理

- Step 0: 先定義所有Pin位置和名稱，再宣告常用變量
- Step 1: 收取控制杆2的資料
- Step 2: 判斷順時針/逆時針(可以只用一個軸做判斷條件)
- Step 2.1: 利用map由範圍換算速度 (PWB)
- Step 2.2: 控制全向輪(BI 1 + BI 2)



旋轉編程

```
  if((y2Value [redacted]){
    //CW
    int yaxis = [redacted](map([redacted]);
    controlLF([redacted];
    controlRR([redacted];
    controlRF([redacted];
    controlLR([redacted];
    Serial.println("Rotate CW");
  }
  else if((y2Value [redacted]){
    //CCW
    int yaxis = [redacted](map([redacted]);
    controlLF([redacted];
    controlRR([redacted];
    controlRF([redacted];
    controlLR([redacted];
    Serial.println("Rotate CCW");
  }
}
```

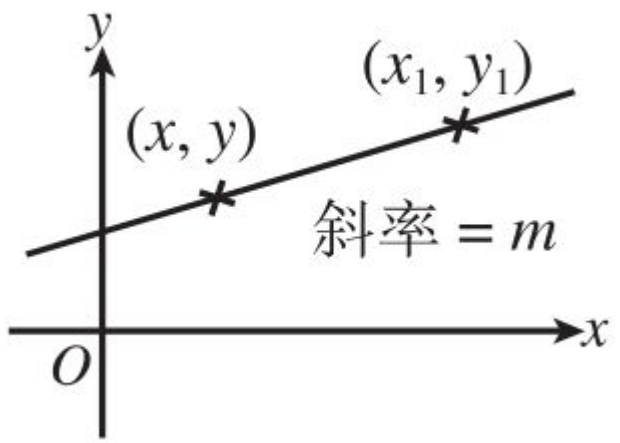
利用控制杆控制全向輪
方法2

判斷方向(點斜式/Point-slope)

- 使用點斜式配搭兩組坐標來計算速度

- X, Y, 永遠是中心點
- X1, Y1 是控制杆的輸入資料

- 適用於非僅垂直 / 水平移動



point-slope form

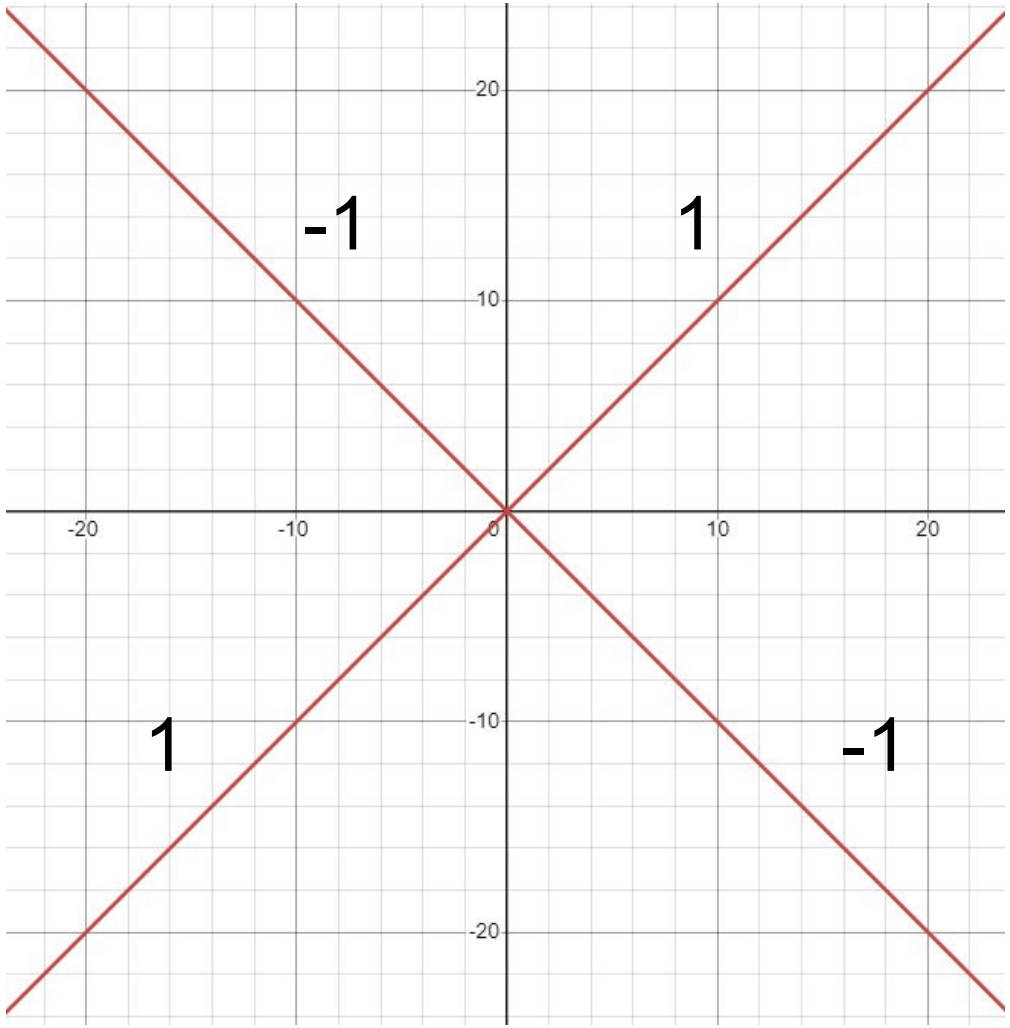
$$y - y_1 = m(x - x_1)$$

$$y - y_1 = m(x - x_1)$$

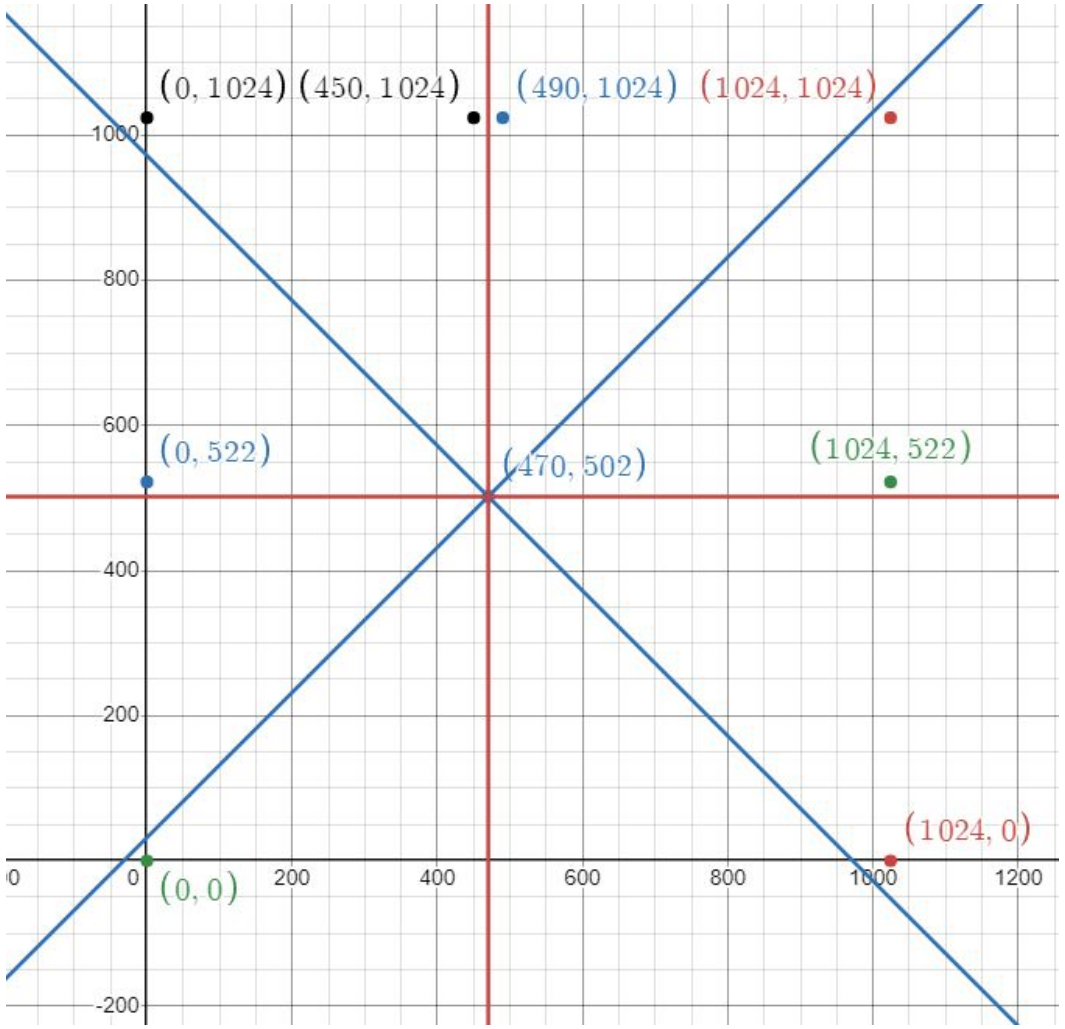
m: slope

(x_1, y_1) : coordinates of a point on the line

判斷方向(點斜式/Point-slope)



判斷方向(點斜式/Point-slope)



編程

```
#define RF_BI1 34
#define RF_BI2 35
#define RF_PWM 12

#define RR_BI1 37
#define RR_BI2 36
#define RR_PWM 8

#define LF_BI1 43
#define LF_BI2 42
#define LF_PWM 9

#define LR_BI1 29
#define LR_BI2 39
#define LR_PWM 5

#define joy1X A9
#define joy1Y A4
#define joy2X A10
#define joy2Y A11

int x1Value = 0;
int y1Value = 0;
int x2Value = 0;
int y2Value = 0;
int deadzone = 5;
int middle_x = 470;
int middle_y = 502;
int middle_y2 = 482;
int upper_bound = 1024;
```

```
double total_length = sqrt(sq(1024.0-470) + sq(1024.0-502));
double ref_slope = (float(upper_bound) - middle_y)/(upper_bound - middle_x);
```

Total_length: 點和點之間最遠的距離

ref_slope: 45角度下, 隨機一點和中心點連成直線的斜率

```
void setup() {
  pinMode(RF_BI1, OUTPUT);
  pinMode(RF_BI2, OUTPUT);
  pinMode(RF_PWM, OUTPUT);

  pinMode(RR_BI1, OUTPUT);
  pinMode(RR_BI2, OUTPUT);
  pinMode(RR_PWM, OUTPUT);

  pinMode(LF_BI1, OUTPUT);
  pinMode(LF_BI2, OUTPUT);
  pinMode(LF_PWM, OUTPUT);

  pinMode(LR_BI1, OUTPUT);
  pinMode(LR_BI2, OUTPUT);
  pinMode(LR_PWM, OUTPUT);
  Serial.begin(9600);

}
```

編程

```
55  void controlRF(int BI1, int BI2, int speed){
56    digitalWrite(RF_BI1, BI1);
57    digitalWrite(RF_BI2, BI2);
58    analogWrite(RF_PWM, speed);
59  }
60
61  void controlRR(int BI1, int BI2, int speed){
62    digitalWrite(RR_BI1, BI1);
63    digitalWrite(RR_BI2, BI2);
64    analogWrite(RR_PWM, speed);
65  }
66
67  void controlLF(int BI1, int BI2, int speed){
68    digitalWrite(LF_BI2, BI2);
69    digitalWrite(LF_BI1, BI1);
70    analogWrite(LF_PWM, speed);
71  }
72
73  void controlLR(int BI1, int BI2, int speed){
74    digitalWrite(LR_BI2, BI2);
75    digitalWrite(LR_BI1, BI1);
76    analogWrite(LR_PWM, speed);
77  }
```

```
79  int lengthtospeed(){
80    double current_length = sqrt(sq(float(x1Value) - middle_x) + sq(float(y1Value) - middle_y));
81    int dir_speed = abs(map(current_length, 0, total_length, 0, 255)); //Length ratio to speed
82    return dir_speed;
83  }
```

lengthtospeed(): 將長度換算成速度

部分編程

```
85  void loop() {
86    // put your main code here, to run repeatedly:
87    x1Value = analogRead(joy1X);
88    y1Value = analogRead(joy1Y);
89    x2Value = analogRead(joy2X);
90    y2Value = analogRead(joy2Y);
91
92    double slope = (y1Value - 502.0)/(x1Value - 470.0);
93    if(slope > 0 && y1Value >= (middle_y + deadzone) && x1Value >= (middle_x + deadzone)){
94      //First Quadrant
95      if(slope > (ref_slope + 0.11)){ //towards North
96        int dir_speed = lengthtospeed();
97        int angle_speed = map(slope*100, ref_slope*100, ((upper_bound-middle_y)/(middle_x+deadzone-middle_x))*100, 5000, dir_speed*100)/100;
98        Serial.println(slope);
99        //Serial.println(dir_speed);
100       Serial.println(angle_speed);
101       controlRF(1, 0, angle_speed);
102       controlLR(0, 1, angle_speed);
103       controlLF(0, 1, dir_speed);
104       controlRR(1, 0, dir_speed);
105       Serial.println("Diagonal 45 w/North");
106     }
```

部分編程

```
107   else if(slope <                    ){ //towards East
108     int dir_speed = lengthtospeed();
109     int angle_speed =                                                    ;
110     controlRF(                    );
111     controlLR(                    );
112     controlLF(                    );
113     controlRR(                    );
114     Serial.println("Diagonal 45 w/East");
115   }
116   else{
117     int dir_speed = lengthtospeed();
118     controlRF(                    );
119     controlLR(                    );
120     controlLF(                    );
121     controlRR(                    );
122     Serial.println("Diagonal 45");
123   }
124 }
```

部分編程

```
125   else if(slope [redacted] && [redacted]){
126     //Third Quadrant
127     if(slope [redacted]){
128       int dir_speed = lengthtospeed();
129       int angle_speed = map([redacted];
130       controlRF[redacted];
131       controlLR[redacted];
132       controlLF[redacted]
133       controlRR[redacted]
134       Serial.println("Diagonal 225 w/South");
135     }
```

部分編程

```
136     else if(slope [redacted]){
137       int dir_speed = lengthtospeed();
138       int angle_speed = map([redacted];
139       controlRF([redacted];
140       controlLR([redacted];
141       controlLF([redacted]
142       controlRR([redacted]
143       Serial.println("Diagonal 225 w/West");
144     }
145     else{
146       int dir_speed = lengthtospeed();
147       controlRF([redacted]
148       controlLR([redacted]
149       controlLF([redacted]
150       controlRR([redacted]
151       Serial.println("Diagonal 225");
152     }
```

部分編程

```
154   else if(slope [redacted] && [redacted]){
155     //Second quadrant
156     if(slope > ([redacted])){
157       int dir_speed = lengthtospeed();
158       int angle_speed = map([redacted];
159       controlLF([redacted];
160       controlRR([redacted];
161       controlRF([redacted]
162       controlLR([redacted]
163       Serial.println("Diagonal 135 w/West");
164     }
165     else if(slope < ([redacted])){
166       int dir_speed = lengthtospeed();
167       int angle_speed = map([redacted];
168       controlLF([redacted];
169       controlRR([redacted];
170       controlRF([redacted]
171       controlLR([redacted]
172       Serial.println("Diagonal 135 w/North");
173     }
174     else{
175       int dir_speed = lengthtospeed();
176       controlLF([redacted]
177       controlRR([redacted]
178       controlRF([redacted];
179       controlLR([redacted];
180       Serial.println("Diagonal 135");
181     }
182   }
```

部分編程

```
183   else if(slope [redacted] && [redacted]){
184     //Fourth quadrant
185     if(slope > [redacted]){
186       int dir_speed = lengthtospeed();
187       int angle_speed = map([redacted];
188       controlLF([redacted];
189       controlRR([redacted];
190       controlRF([redacted]
191       controlLR([redacted]
192       Serial.println("Diagonal 315 w/East");
193     }
194     else if(slope < [redacted])){
195       int dir_speed = lengthtospeed();
196       int angle_speed = map([redacted];
197       controlLF([redacted];
198       controlRR([redacted];
199       controlRF([redacted]
200       controlLR([redacted]
201       Serial.println("Diagonal 315 w/South");
202     }
203     else{
204       int dir_speed = lengthtospeed();
205       controlLF([redacted]
206       controlRR([redacted]
207       controlRF([redacted]
208       controlLR([redacted]
209       Serial.println("Diagonal 315");
210     }
211   }
```

部分編程 [redacted]

```
212   else if([redacted] >= (middle_x + deadzone)){
213     //右邊
214     int xaxis = abs(map([redacted]);
215     controlRF([redacted];
216     controlRR([redacted];
217     controlLF([redacted];
218     controlLR([redacted];
219     Serial.println("sidewaysRIGHT");
220   }
221   else if([redacted] <= (middle_x - deadzone)){ //左邊
222     int xaxis = abs(map([redacted]);
223     controlRF([redacted];
224     controlRR([redacted];
225     controlLF([redacted];
226     controlLR([redacted];
227     Serial.println("sidewaysLEFT");
228   }
```

```
229   else if(y1Value >= [redacted]){ //前
230     int yaxis = abs(map([redacted]);
231     controlRF([redacted]);
232     controlRR([redacted]);
233     controlLF([redacted]);
234     controlLR([redacted]);
235     Serial.println("forward");
236   }
237   else if(y1Value <= [redacted]){ //後
238     int yaxis = abs(map([redacted]);
239     controlRF([redacted]);
240     controlRR([redacted]);
241     controlLF([redacted]);
242     controlLR([redacted]);
243     Serial.println("backward");
244   }
245   else{
246     controlLF(0, 0, 0);
247     controlRR(0, 0, 0);
248     controlRF(0, 0, 0);
249     controlLR(0, 0, 0);
250     //Serial.println("Idle");
251   }
```

部分編程

```
252
253     if((y2Value >= (middle_y2 + deadzone))){
254       //CW
255       int yaxis = abs(map(y2Value, 0, 1024, -255, 255));
256       controlLF(0, 1, yaxis);
257       controlRR(0, 1, yaxis);
258       controlRF(0, 1, yaxis);
259       controlLR(0, 1, yaxis);
260       Serial.println("Rotate CW");
261     }
262     else if((y2Value <= (middle_y2 - deadzone))){
263       //CCW
264       int yaxis = abs(map(y2Value, 0, 1024, -255, 255));
265       controlLF(1, 0, yaxis);
266       controlRR(1, 0, yaxis);
267       controlRF(1, 0, yaxis);
268       controlLR(1, 0, yaxis);
269       Serial.println("Rotate CCW");
270     }
271   }
```

Appendix