# OminibotHV
# 底層通訊與小車運動學

By Bill

2023/08/25

CIRCUS Pi

# OminibotHV - 通訊封包格式(接收/回授)

# OminibotHV – 通訊封包格式定義

- 數據回授封包格式
  - 這裡還有一個地方需要注意，設定機器人硬體或是系統的原始資料，皆是浮點型的資料（float），因為浮點型資料使用串口傳輸不方便，所以這四個資料在發送之前先將浮點數**放大一千倍（保留小數點後三位）**，再將放大後的浮點數強制轉換成 short 型資料，最後在發送前將 short 型數據拆分成兩個 8 位元的數據。相應的，下位機端(STM32)在接收到資料後，需要將接收到的數據兩個 8 位元資料合併後轉換為 short 型，在**縮小一千倍**來進行單位的轉換。

# OminibotHV – 通訊封包格式定義(接收)

- STM32接收數據封包格式(14 char) – 系統配置

無單位量

| 數據 | Tx[0] | Tx[1] | Tx[2] | Tx[3] | Tx[4] | Tx[5] | Tx[6] | Tx[7] | Tx[8] | Tx[9] | Tx[10] | Tx[11] | Tx[12] | Tx[13] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 內容 | 幀頭 0x7B ( '{' ) | 命令模式 0x23 ( '#' ) | 馬達方向定義_高八位 | 編碼器方向定義_低八位 | 馬達PWM最大輸出值 | | 馬達PWM最小輸出值 | | 編碼器PPR (pulse per rotation) | | 預留 | | BBC 校驗位 | 幀尾 0x7D ( '}' ) |
| 字節大小 | char | char | Short | | Short (1 to 7199) | | Short (1 to 7199) | | Short | | Short (0) | | char | char |
| 陣列位置 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

舉例說明 :
馬達方向定義只用到低八位(M4/M3/M2/M1) ，0為正轉，1為反轉。
EX : 0x02 ，M2反轉，其餘正轉

| 0 | 0 | 1 | 0 |
|---|---|---|---|

編碼器方向定義，同馬達方向定義(E4/E3/E2/E1) 。
EX:



CIRCUS Pi

# OminibotHV – 通訊封包格式定義(接收)

- STM32接收數據封包格式(14 bytes) – **小車尺寸參數配置**

**單位:mm**

| 數據 | Tx[0] | Tx[1] | Tx[2] | Tx[3] | Tx[4] | Tx[5] | Tx[6] | Tx[7] | Tx[8] | Tx[9] | Tx[10] | Tx[11] | Tx[12] | Tx[13] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 內容 | 幀頭 0x7B ( '{' ) | 命令模式 0x24 ( '$' ) | 小車輪距 (Wheel Space) | | 小車軸距 (Axle Space) | | 馬達齒輪比 (Gear Ratio) | | 輪子直徑 (Wheel Diameter) | | 預留 | | BBC 校驗位 | 幀尾 0x7D ( '}' ) |
| 字節大小 | char | char | Short | | Short | | Short | | Short | | Short | | char | char |
| 陣列位置 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |



CIRCUS Pi

# OminibotHV – 通訊封包格式定義(接收)

- STM32接收數據封包格式(14 bytes) – **小車PID參數配置**

| 數據 | Tx[0] | Tx[1] | Tx[2] | Tx[3] | Tx[4] | Tx[5] | Tx[6] | Tx[7] | Tx[8] | Tx[9] | Tx[10] | Tx[11] | Tx[12] | Tx[13] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 內容 | 幀頭 0x7B ( '{' ) | 命令模式 0x40 ( '@' ) | POS_Kp (需要放大100倍) | | POS_Ki (需要放大100倍) | | POS_Kd (需要放大100倍) | | VEL_Kp (需要放大100倍) | | VEL_Ki (需要放大100倍) | | BBC 校驗位 | 幀尾 0x7D ( '}' ) |
| 字節大小 | char | char | Short | | Short | | Short | | Short | | Short | | char | char |
| 陣列位置 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

**單位:放大100倍** **單位:放大100倍** **單位:放大100倍**



CIRCUS Pi

# OminibotHV – 通訊封包格式定義(接收)

- 舉例KUBOT參數設定說明(最大速度只有 0.368 m/s) :
  - mobptr-> MotorDir=0x00;
  - mobptr-> EncoderDir =0x02;
  - mobptr->Divisor_Mode = 4;
  - mobptr->GearRatio = 139;
  - mobptr->WheelEncoderPrecision = Divisor_Mode * 973;
  - mobptr->WheelDiameter = 0.160f;
  - mobptr->TrackSpacing = 0.240f;
  - mobptr->AxleSpacing = 0.0f;
  - mobptr->PWM_DutyCy_Max = 7056; //set pwm max duty cycle to 98%
  - mobptr->PWM_DutyCy_Min = 1440; //set pwm min duty cycle to 20%

# OminibotHV – 通訊封包格式定義(接收)

- 舉例KUBOT參數設定說明(最大速度只有 0.368 m/s) :
  - mobptr->pid_pos[0] = 30.0f; //位置控制參數KP
  - mobptr->pid_pos[1] = 10.5f; //位置控制參數KI
  - mobptr->pid_pos[2] = 0.0f; //位置控制參數KD
  - mobptr->pid_vel[0] = 30.5f; //速度控制參數KP
  - mobptr->pid_vel[1] = 10.5f; //速度控制參數KI
  -

# OminibotHV – 通訊封包格式定義(接收)

- 舉例小車系統配置：
  - 7B 23 00 02 1B 90 05 A0 03 CD 00 00 BA 7D( { # 0 2 7056 1440 973 0 } )
- 舉例小車尺寸參數配置：
  - 7B 24 00 F0 00 00 00 8B 00 A0 00 00 84 7D( { $ 240 0 139 160 0 B } )
- 舉例小車PID參數配置:
  - 7B 40 0B B8 04 1A 00 00 0B B8 04 1A 3B 7D( { @ 3000 1050 0 3000 1050 B } )
  - ( { @ 6000 1050 0 6000 1050 B } )



CIRCUS Pi

# OminibotHV – 通訊封包格式定義(接收)

- STM32接收數據封包格式- **運動控制配置**

單位:mm/s或是1000*rad/s

| 數據 | Tx[0] | Tx[1] | Tx[2] | Tx[3] | Tx[4] | Tx[5] | Tx[6] | Tx[7] | Tx[8] | Tx[9] | Tx[10] | Tx[11] | Tx[12] | Tx[13] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 內容 | 幀頭 0x7B ( '{' ) | 命令模式 ( '%' '&' ) | 控制模式(停止/位置/速度) | 小車型別 | X方向速度或A馬達速率_高八位 | X方向速度或A馬達速率_低八位 | Y方向速度或B馬達速率_高八位 | Y方向速度或B馬達速率_低八位 | Z方向速度或C馬達速率_高八位 | Z方向速度或C馬達速率_低八位 | W方向速度或D馬達速率_高八位 | W方向速度或D馬達速率_低八位 | BBC 校驗位 | 幀尾 0x7D ( '}' ) |
| 字節大小 | char | char | char | char | short | | short | | short | | short | | char | char |
| 陣列位置 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |



CIRCUS Pi

# OminibotHV – 通訊封包格式定義(接收)

- STM32接收數據封包格式 – 命令模式選擇(由上層的ROS決定)
  - 0x23(#) ：小車運動系統配置
  - 0x24 ($) ：小車運動學參數配置
  - 0x25(%) ：小車速度控制模式(Vx, Vy, Vz)
  - 0x26 (&) ：小車各馬達獨立控制 with 編碼器(M1, M2, M3, M4)
  - 0x40 (@) ：小車PID參數
- STM32接收數據封包格式 – 小車型別(由上層的ROS決定)
  - 0x02：TWODDMOBILE (ex: kubot)
  - 0x03：OMINIMOBILE (ex: 三輪智慧機器人)
  - 0x04：MECANUMMOBILE(ex: rosky2, racingrat)
  - 0x05：FOURWDMOBILE(rosky1)

STM32接收數據封包格式 - 控制模式

FORCESTOP ： 0 (強制停止)

POSITIONCTRL ： 1 (位置控制)

VELOCITYCTRL ： 2 (速度控制)



CIRCUS Pi

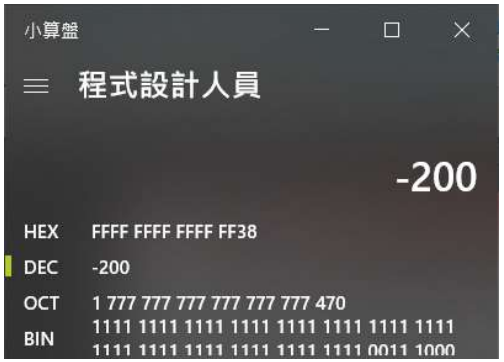
# OminibotHV – 通訊封包格式定義(接收)

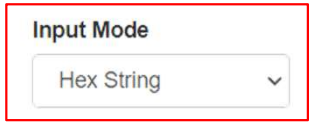
- 舉例說明：
  - X軸，+200mm/s ，7B 25 02 05 00 C8 00 00 00 00 00 00 91 7D
  - 停止，0mm/s ，7B 25 02 05 00 00 00 00 00 00 00 00 59 7D
  - X軸，-200mm/s ，7B 25 02 05 FF 38 00 00 00 00 00 00 9E 7D

車體ROSKY1做速度控制

  - 馬達M1，+200mm/s，7B 26 02 05 00 C8 00 00 00 00 00 00 92 7D
  - 馬達M4，-200mm/s ，7B 26 02 05 00 00 00 00 00 00 FF 38 9D 7D

車體ROSKY1做馬達速度控制


Block Check Character Calculator



CIRCUS Pi

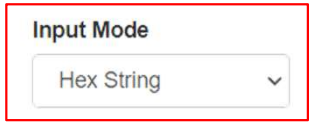
# OminibotHV – 通訊封包格式定義(接收)

- 舉例說明：
  - X軸，+200mm/s ，7B 25 02 02 00 C8 00 00 00 00 00 00 96 7D
  - 停止，0mm/s ，7B 25 02 02 00 00 00 00 00 00 00 00 5E 7D
  - X軸，-200mm/s ，7B 25 02 02 FF 38 00 00 00 00 00 00 99 7D

KUBOT車體做線性速度控制

  - 馬達M1/M2，+100mm/s，7B 26 02 02 00 64 00 64 00 00 00 00 5D 7D
  - 馬達M1/M2 ，-200mm/s ，7B 26 02 02 FF 38 FF 38 00 00 00 00 5D 7D

KUBOT馬達做旋轉速度控制


Block Check Character Calculator



CIRCUS Pi

# OminibotHV – 通訊封包格式定義(回授)

- STM32回授數據封包格式 – 命令模式選擇(由上層的ROS決定)
  - 0x33( ‘3’ ) : 回授小車運動系統配置
  - 0x34 ( ‘4’ ) : 回授小車運動學參數配置
  - 0x35( ‘5’ ) : 回授小車速度控制模式(Vx, Vy, Vz)
  - 0x36 ( ‘6’ ) : 回授小車各馬達獨立控制 with 編碼器(M1, M2, M3, M4)
  - 0x50 ( ‘P’ ) : 回授小車PID參數

# OminibotHV – 通訊封包格式定義(回授)

- STM32回授數據封包格式- **速度控制配置**

單位:mm/s或是1000*rad/s

| 數據 | Tx[0] | Tx[1] | Tx[2] | Tx[3] | Tx[4] | Tx[5] | Tx[6] | Tx[7] | Tx[8] | Tx[9] | Tx[10] | Tx[11] | Tx[12] | Tx[13] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 內容 | 幀頭 0x7B ( '**{**' ) | 命令模式 0x35/0x36 ( '**5**' '**6**') | 控制模式(位置/速度/**停止**) | 小車型別 | X方向速度或A馬達速率_高八位 | X方向速度或A馬達速率_低八位 | Y方向速度或B馬達速率_高八位 | Y方向速度或B馬達速率_低八位 | Z方向速度或C馬達速率_高八位 | Z方向速度或C馬達速率_低八位 | W方向速度或D馬達速率_高八位 | W方向速度或D馬達速率_低八位 | BBC 校驗位 | 幀尾 0x7D ( '**}**' ) |
| 字節大小 | char | char | char | char | short | | short | | short | | short | | char | char |
| 陣列位置 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |



CIRCUS Pi

# OminibotHV – 通訊封包格式定義(回授)

- STM32回授數據封包格式(14 char) – **系統配置**

**無單位量**

| 數據 | Tx[0] | Tx[1] | Tx[2] | Tx[3] | Tx[4] | Tx[5] | Tx[6] | Tx[7] | Tx[8] | Tx[9] | Tx[10] | Tx[11] | Tx[12] | Tx[13] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 內容 | 幀頭 0x7B ( '{' ) | 命令模式 0x33 ( '3' ) | 馬達方向定義_高八位 | 編碼器方向定義_低八位 | 馬達PWM最大輸出值 | | 馬達PWM最小輸出值 | | 編碼器PPR (pulse per rotation) | | 預留 | | BBC 校驗位 | 幀尾 0x7D ( '}' ) |
| 字節大小 | char | char | Short | | Short (max to 7199) | | Short (min to 720) | | Short | | Short (0) | | char | char |
| 陣列位置 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

| 0 | 0 | 1 | 0 |
|---|---|---|---|



CIRCUS Pi

# OminibotHV – 通訊封包格式定義(回授)

- STM32回授數據封包格式(14 bytes) – **小車尺寸參數配置**

**單位:mm**

| 數據 | Tx[0] | Tx[1] | Tx[2] | Tx[3] | Tx[4] | Tx[5] | Tx[6] | Tx[7] | Tx[8] | Tx[9] | Tx[10] | Tx[11] | Tx[12] | Tx[13] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 內容 | 幀頭 0x7B ( '{' ) | 命令模式 0x34 ( '4' ) | 小車輪距 (Wheel Space) | | 小車軸距 (Axle Space) | | 馬達齒輪比 (Gear Ratio) | | 輪子直徑 (Wheel Diameter) | | 預留 | | BBC 校驗位 | 幀尾 0x7D ( '}' ) |
| 字節大小 | char | char | Short | | Short | | Short | | Short | | Short | | char | char |
| 陣列位置 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |



CIRCUS Pi

# OminibotHV – 通訊封包格式定義(回授)

- STM32回授數據封包格式(14 bytes) – **小車PID參數配置**

| 數據 | Tx[0] | Tx[1] | Tx[2] | Tx[3] | Tx[4] | Tx[5] | Tx[6] | Tx[7] | Tx[8] | Tx[9] | Tx[10] | Tx[11] | Tx[12] | Tx[13] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 內容 | 幀頭 0x7B ( '{' ) | 命令模式 0x50 ( 'P' ) | POS_Kp (需要放大100倍) | | POS_Ki (需要放大100倍) | | POS_Kd (需要放大100倍) | | VEL_Kp (需要放大100倍) | | VEL_Ki (需要放大100倍) | | BBC 校驗位 | 幀尾 0x7D ( '}' ) |
| 字節大小 | char | char | Short | | Short | | Short | | Short | | Short | | char | char |
| 陣列位置 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

**單位:放大100倍** **單位:放大100倍** **單位:放大100倍**



CIRCUS Pi

# OminibotHV – 通訊封包格式定義(回授)

- 舉例讀回小車系統配置：
  - 7B 33 00 00 00 00 00 00 00 00 00 00 48 7D( { 3 0 0 0 0 0 B } )
- 舉例讀回小車尺寸參數配置：
  - 7B 34 00 00 00 00 00 00 00 00 00 00 4F 7D( { 4 0 0 0 0 0 B } )
- 舉例讀回小車PID參數配置:
  - 7B 50 00 00 00 00 00 00 00 00 00 00 2B 7D( { P 0 0 0 0 0 B } )

# OminibotHV – 通訊封包格式定義(回授)

- STM32回授數據封包格式(mm/s & 放大1000)

| 數據 | Tx[0] | Tx[1] | Tx[2] | Tx[3] | Tx[4] | Tx[5] | Tx[6] | Tx[7] | Tx[8] | Tx[9] | Tx[10] | Tx[11] | Tx[12] | Tx[13] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 內容 | 幀頭 0x7B | 預留 | X方向速度_高八位 | X方向速度_低八位 | Y方向速度_高八位 | Y方向速度_低八位 | Z方向速度_高八位 | Z方向速度_低八位 | X軸加速度_高八位 | X軸加速度_低八位 | Y軸加速度_高八位 | Y軸加速度_低八位 | Z軸加速度_高八位 | Z軸加速度_低八位 |
| 字節大小 | char | char | short | | short | | short | | short | | short | | short | |
| 陣列位置 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |



CIRCUS Pi

# OminibotHV – 通訊封包格式定義(回授)

- STM32回授數據封包格式(放大1000)

| 數據 | Tx[14] | Tx[15] | Tx[16] | Tx[17] | Tx[18] | Tx[19] | Tx[20] | Tx[21] | Tx[22] | Tx[23] | Tx[24] | Tx[25] | Tx[26] | Tx[27] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 內容 | X方向角速度高八位 | X方向角速度低八位 | Y方向角速度高八位 | Y方向角速度低八位 | Z方向角速度高八位 | Z方向角速度低八位 | 四元數W方向高八位 | 四元數W方向低八位 | 四元數X方向高八位 | 四元數X方向低八位 | 四元數Y方向高八位 | 四元數Y方向低八位 | 四元數Z方向高八位 | 四元數Z方向低八位 |
| 字節大小 | short | | short | | short | | short | | short | | short | | short | |
| 陣列位置 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |



CIRCUS Pi

# OminibotHV – 通訊封包格式定義(回授)

- STM32回授數據封包格式
  - 這裡還有一個地方需要注意，機器人 XYZ **三軸線性速度、加速度計、角速度計、四元數方位值以及電池電壓**的原始資料是浮點型的資料（float），因為浮點型資料使用串口傳輸不方便，所以這四個資料在發送之前先將浮點數**放大一千倍（保留小數點後三位）**，再將放大後的浮點數強制轉換成 short 型資料，最後在發送前將 short 型數據拆分成兩個 8 位元的數據。相應的，上位機端在接收到資料後，需要將接收到的數據兩個 8 位元資料合併後轉換為 short 型，在縮小一千倍來進行單位的轉換。

| 數據 | Tx[28] | Tx[29] | Tx[30] | Tx[31] |
|---|---|---|---|---|
| 內容 | 電壓_高八位 | 電壓_低八位 | BBC校驗位 | 幀尾0x7D |
| 字節大小 | short | | char | char |
| 陣列位置 | 28 | 29 | 30 | 31 |

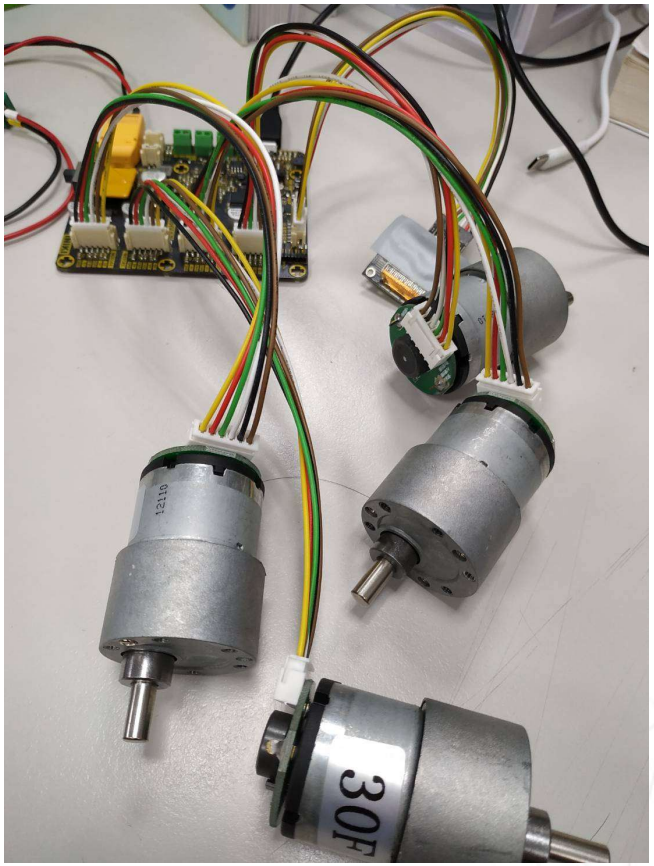


CIRCUS Pi

# OminibotHV板 – QC功能測試範例程式(4輪小車)

- 確認硬體接線和開關
  - PWR_ON(開啟電源)
  - MODE_ON(OLED才會顯示)
- 參數設定
  - 四輪小車系統配置
- 觀察OLED
  - 四輪小車運動控制


預設+12V



```
class ominibothv:
    def __init__(self,
                 port = '/dev/ominibot',
                 baud = 115200,
                 divisor_mode = 3,
                 motor_direct = 2,
                 encoder_direct = 0,
                 motor_pwm_max = 5200,
                 motor_pwm_min = 720,
                 encoder_ppr = 390,
                 wheel_space = 110,
                 axle_space = 0,
                 gear_ratio = 30,
                 wheel_diameter = 60,
                 pos_kp = 3000,
                 pos_ki = 1050,
                 pos_kd = 0,
                 vel_kp = 3000,
                 vel_ki = 1050):
```



CIRCUS Pi

# OminibotHV板 –QC功能測試範例程式(4輪小車)

- 第一步，依序執行四輪小車系統配置：
  1) 7B 23 **05 00** 10 00 02 D0 01 86 00 00 **18** 7D( { # 0 2 4096 720 390 0 } )
  2) 7B 24 00 F0 00 00 00 8B 00 A0 00 00 **84** 7D( { $ 240 0 139 160 0 B } )
  3) 7B 40 0B B8 04 1A 00 00 0B B8 04 1A **3B** 7D( { @ 3000 1050 0 3000 1050 B } )
- 第二步，四輪小車運動控制：
  1) X軸，+200mm/s ，7B 25 02 **05 00 C8** 00 00 00 00 00 00 **91** 7D
  2) 停止，0mm/s ，7B 25 02 **05** 00 00 00 00 00 00 00 00 **59** 7D
  3) X軸，-200mm/s ，7B 25 02 **05 FF 38** 00 00 00 00 00 00 **9E** 7D
  4) 馬達M1，+200mm/s，7B 26 02 **05 00 C8** 00 00 00 00 00 00 **92** 7D
  5) 馬達M2，+200mm/s，7B 26 02 **05** 00 00 **00 C8** 00 00 00 00 **92** 7D
  6) 馬達M3，+200mm/s，7B 26 02 **05** 00 00 00 00 **00 C8** 00 00 **92** 7D
  7) 馬達M4，+200mm/s，7B 26 02 **05** 00 00 00 00 00 00 **00 C8 92** 7D

# OminibotHV - FLASH讀寫控制

# Stm32範例程式(九) – FLASH讀寫控制

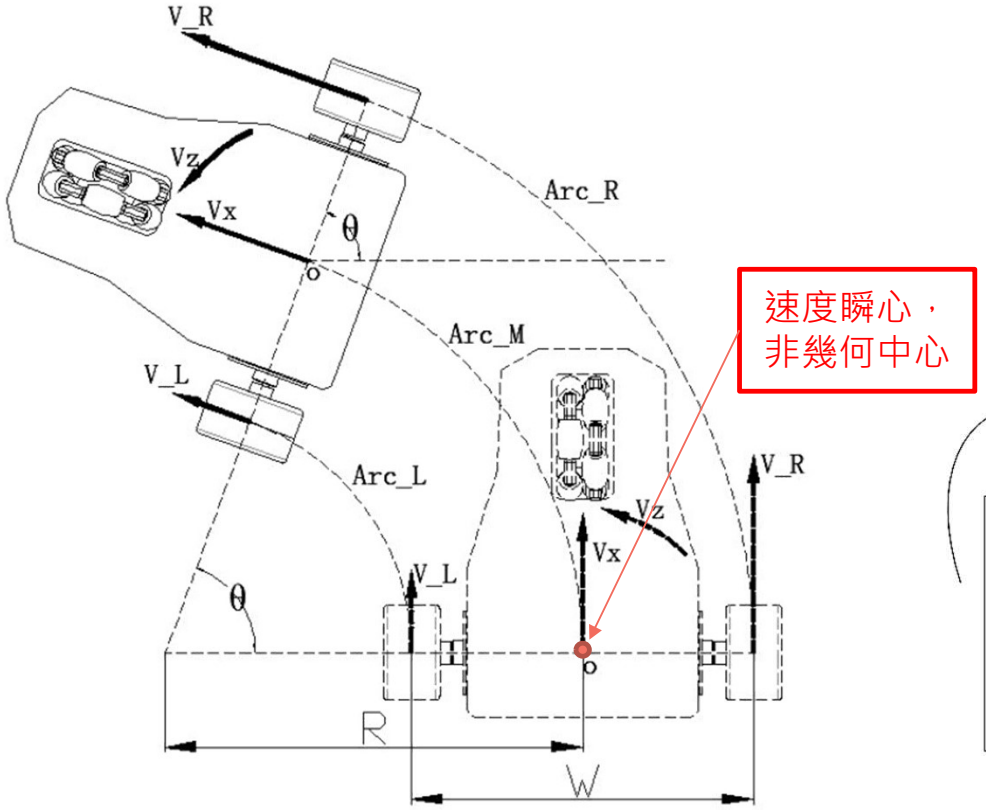
- FLASH存儲起始位址 : 0x0801FC00

| 數據 | [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] | [10] | [11] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 內容 | 小車型別(1~4) | 控制模式(1~2) | 位置PID_P值 | 位置PID_I值 | 位置PID_D值 | 速度PID_P值 | 速度PID_I值 | 速度PID_D值 | PWM_最大值 | PWM_最小值 | 預留 | 預留 |
| 字節大小 | word | word | word | word | word | word | word | word | word | word | word | word |
| 陣列位置 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

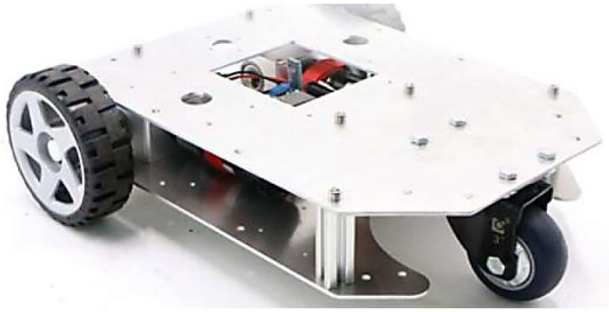
# OminibotHV - 小車正/逆運動學

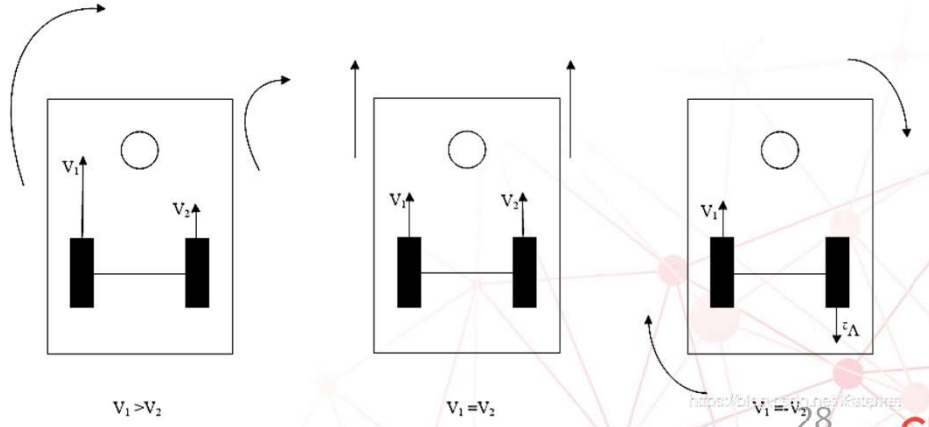
# OminibotHV - 小車正/逆運動學(兩輪差速)

- 小車運動速度與各馬達速率輸出的轉換關係




速度瞬心，
非幾何中心








CIRCUS Pi

# OminibotHV - 小車正/逆運動學(兩輪差速)

- 小車運動速度與各馬達速率輸出的轉換關係


$$\begin{bmatrix} V_x \\ \omega_z \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{-1}{d} & \frac{1}{d} \end{bmatrix} \begin{bmatrix} V_L \\ V_R \end{bmatrix}$$

$$\begin{bmatrix} V_L \\ V_R \end{bmatrix} = \begin{bmatrix} 1 & \frac{-d}{2} \\ 1 & \frac{d}{2} \end{bmatrix} \begin{bmatrix} V_x \\ \omega_z \end{bmatrix}$$

```
void Drive_Motor(float vx,float vz)
{
    Target_Left  = vx - vz * WIDTH_OF_ROBOT / 2.0f; //计算出左轮的目标速度
    Target_Right = vx + vz * WIDTH_OF_ROBOT / 2.0f; //计算出右轮的目标速度
}
```

以上语句是通过机器人的 X 和 Z 轴方向的速度求两个电机的目标速度大小（运动学逆解），其中 WIDTH_OF_ROBOT 是两个车轮直线距离的宏定义。



CIRCUS Pi

# OminibotHV - 小車正/逆運動學(四輪差速)

- **前輪和後輪的速度是同步的**
- 點COG為車體幾何中心點
- 點ICR為車體作圓周運動的中心點，大小與角速度相關
- **底盤速度瞬心在COM點**，但COM與COG不一定重合

$d_i$：四个轮子到 ICR 的距离↵

轮子的实际速度$v_i$：侧向滑动速度$v_{iy}$和预设目标速度$v_{ix}$的合成速度 i=1, 2, 3, 4↵

在模型中底盘的速度瞬心在质心 COM 处，而 COM 和 COG 往往是不重合的。↵

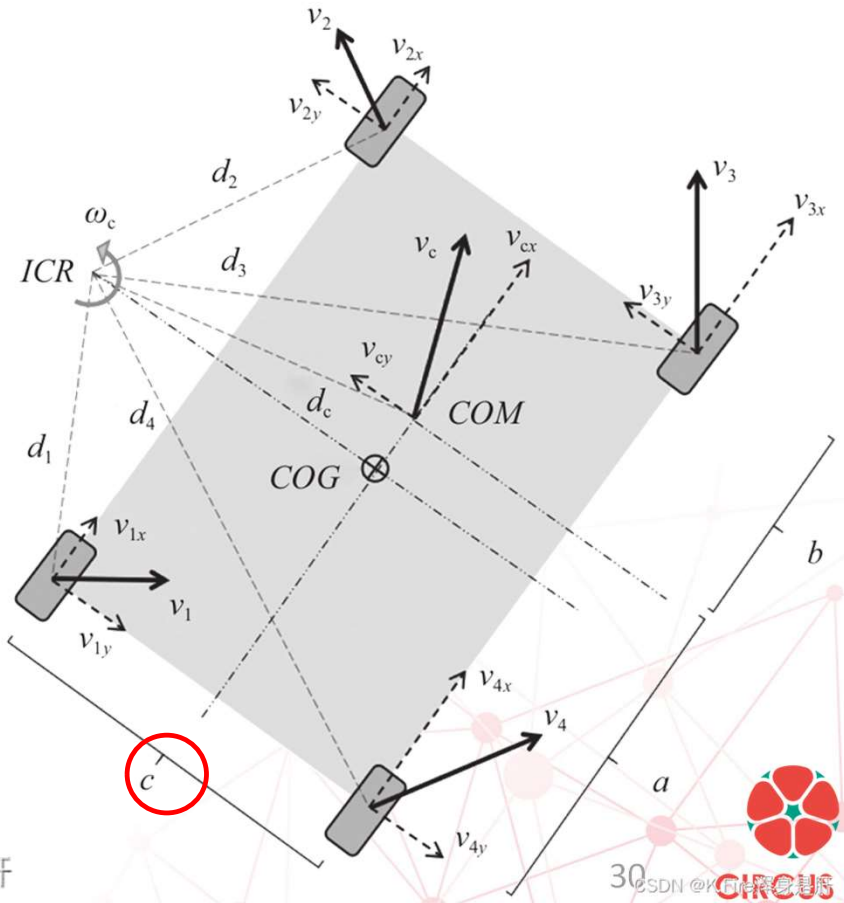
整个底盘的运动速度：用 COM 位置处的线速度$v_c$和角速度$\omega_c$表示↵

$d_c$：COM 到 ICR 的距离↵

c：底盘中左轮、右轮轴距↵

a：点 COM 与底盘后端之间距离↵

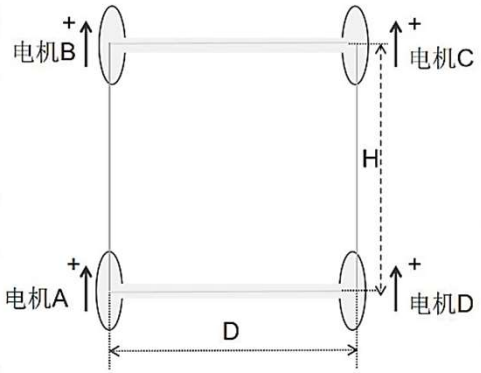
b：点 COM 与前端之间的距离↵



# OminibotHV - 小車正/逆運動學(四輪差速)

$$\begin{bmatrix} V_{cx} \\ \omega_c \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{-1}{c} & \frac{1}{c} \end{bmatrix} \begin{bmatrix} V_L \\ V_R \end{bmatrix}$$

$$\begin{bmatrix} V_L \\ V_R \end{bmatrix} = \begin{bmatrix} 1 & \frac{-c}{2} \\ 1 & \frac{c}{2} \end{bmatrix} \begin{bmatrix} V_{cx} \\ \omega_c \end{bmatrix}$$




② C 语言实现

```
void Drive_Motor(float vx,float vy,float vz)
{
    MotorTarget.A   = (vx-vz*(Wheel_spacing+Wheel_axlespacing));
    MotorTarget.B   = (vx-vz*(Wheel_spacing+Wheel_axlespacing));
    MotorTarget.C   = (vx+vz*(Wheel_spacing+Wheel_axlespacing));
    MotorTarget.D   = (vx+vz*(Wheel_spacing+Wheel_axlespacing));
}
```

平衡小車這邊有問題

$V_L$

$V_R$

Wheel_axlespacing 为小车(前后)轴距参数，Wheel_spacing 为小车(左右)轮距参数。



CIRCUS Pi

# OminibotHV - 小車正/逆運動學(麥克拉姆輪)

*A. Mobile Platform with 4 Mecanum Wheels*

A typical configuration of a Mecanum wheeled platform consists of 4 wheels (see Fig. 4). The positions of the wheels

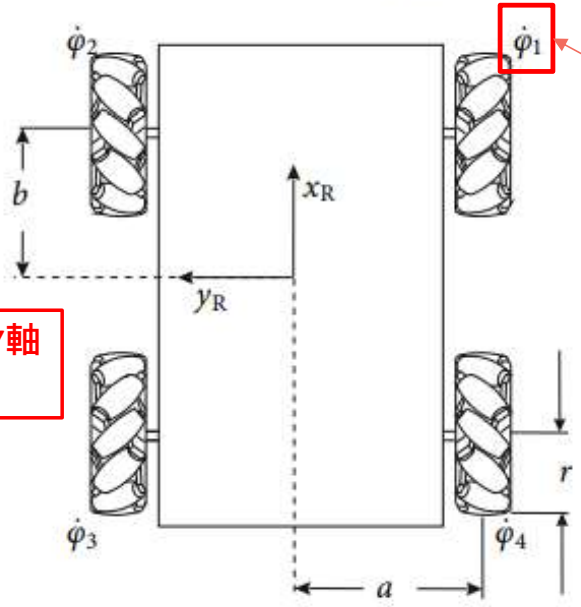
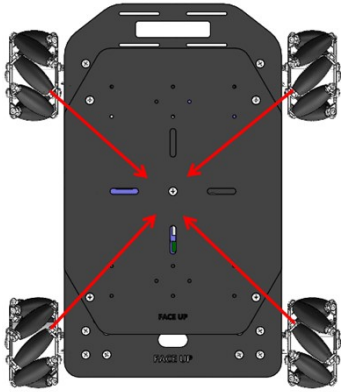



Fig. 4. Omnidirectional platform with Mecanum wheels (top view)

注意輪子的位置



Installation of the Wheels

注意車子的XY軸方向定義

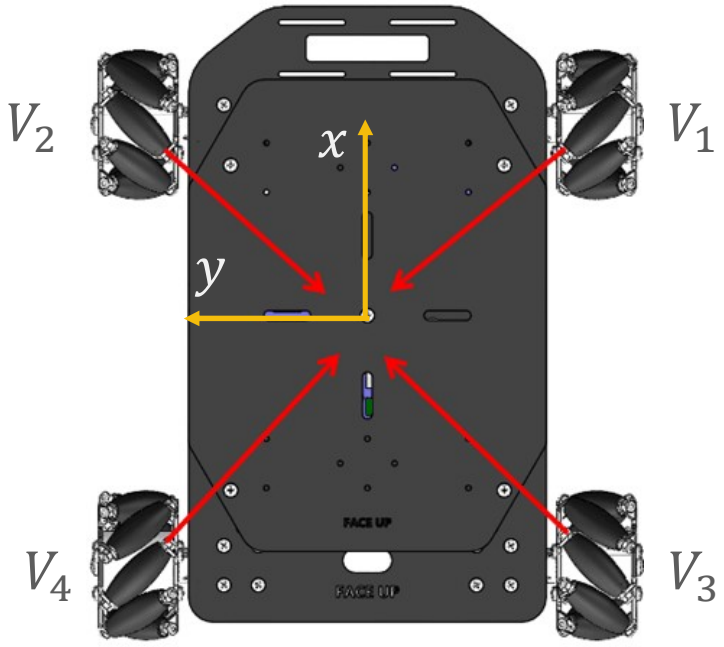
with respect to the robot frame can be defined as $\delta_i = 0$ and $l_i = \sqrt{a^2 + b^2}$. This leads to the configuration parameters shown in table I. These parameters in in conjunction with (8) yield to the inverse kinematics

$$\begin{pmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \\ \dot{\varphi}_4 \end{pmatrix} = \boldsymbol{J} \begin{pmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta} \end{pmatrix}, \quad \text{with } \boldsymbol{J} = \frac{1}{r} \begin{pmatrix} 1 & 1 & (a+b) \\ 1 & -1 & -(a+b) \\ 1 & 1 & -(a+b) \\ 1 & -1 & (a+b) \end{pmatrix} \tag{15}$$

CIRCUS Pi

# OminibotHV - 小車正/逆運動學(麥克拉姆輪)

- 根據ROSKY2的電路設計定義




Installation of the Wheels

$$\begin{bmatrix} V_x \\ V_y \\ W \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{a+b} & \frac{-1}{a+b} & \frac{-1}{a+b} & \frac{-1}{a+b} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix}$$

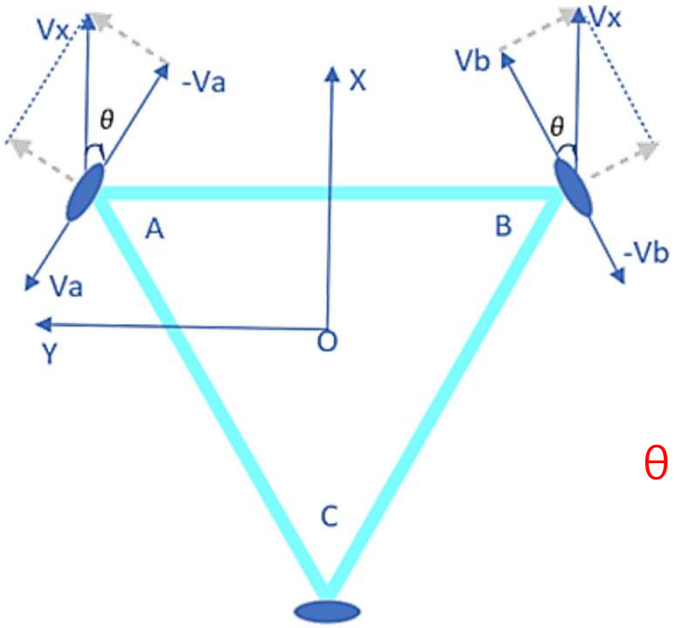
$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & (a+b) \\ 1 & -1 & -(a+b) \\ 1 & -1 & (a+b) \\ 1 & 1 & -(a+b) \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ W \end{bmatrix}$$



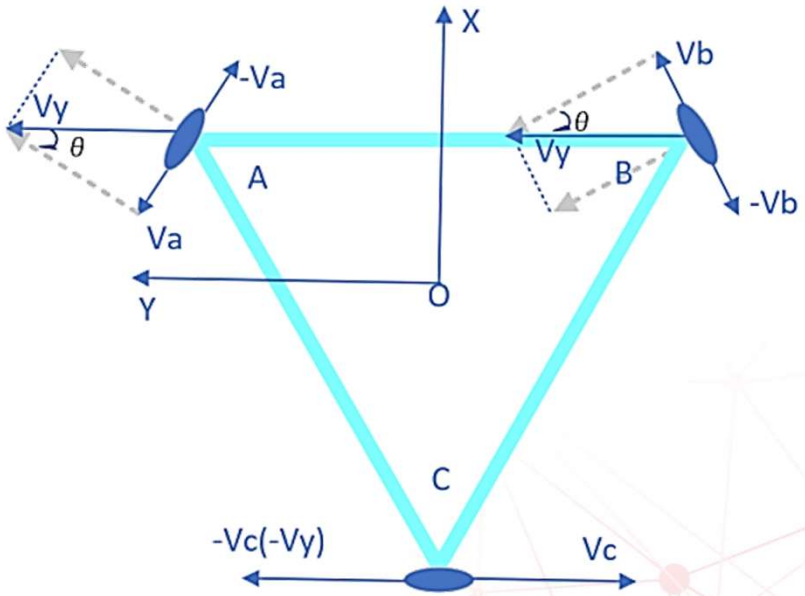
CIRCUS Pi

# OminibotHV - 小車正/逆運動學(全向輪)

- 小車運動速度與各馬達速率輸出的轉換關係




θ=30°




X軸方向移動

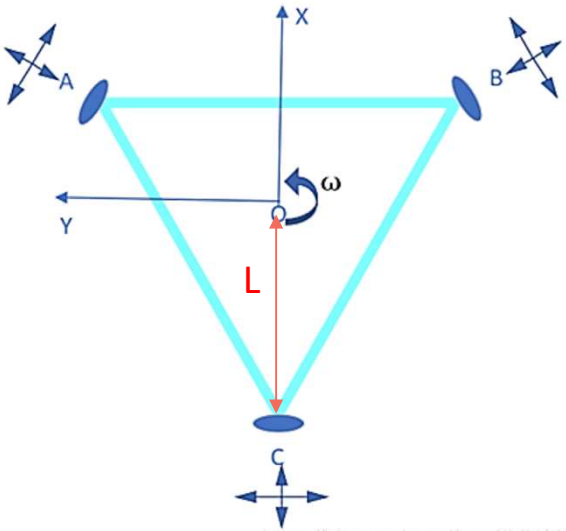
Y軸方向移動



CIRCUS Pi

# OminibotHV - 小車正/逆運動學(全向輪)

- 小車運動速度與各馬達速率輸出的轉換關係
  - 全向輪不與地面打滑，同時地面有足夠摩擦力;
  - 電機**軸線中心**正是底盤重心;
  - 各輪之間是絕對的互成**120°安裝**。




$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \begin{bmatrix} \frac{-\sqrt{3}}{2} & \frac{1}{2} & L \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & L \\ 0 & -1 & L \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ W_z \end{bmatrix}$$

$$\begin{bmatrix} V_X \\ V_Y \\ W_Z \end{bmatrix} = \begin{bmatrix} \frac{-\sqrt{3}}{3} & \frac{\sqrt{3}}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{-2}{3} \\ \frac{1}{3L} & \frac{1}{3L} & \frac{1}{3L} \end{bmatrix} \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix}$$





CIRCUS Pi