東海大學電機工程學系

碩士論文

全向輪高階控制方法的開發

Development of a high level tool for programming mecanum wheel

學生：黃冠璋

指導教授：蔡坤霖、黃宇中 博士

中華民國 106 年 7 月

東海大學電機工程學系

碩士論文

全向輪高階控制方法的開發

Development of a high level tool for programming mecanum wheel

學生：黃冠璋

指導教授：蔡坤霖、黃宇中 博士

中華民國 106 年 7 月

論文口試委員審定書

東海大學電機工程學系碩士學位
考試委員審定書

電機工程學系研究所 黃冠璋 君所提之論文

全向輪高階控制語法的開發，經本考試

委員會審查，符合碩士

資格標準。

學位考試委員會 召集人：[illegible] (簽章)

委 員：蔡坤霖

[illegible]

[illegible]

[illegible]

中 華 民 國 106 年 06 月 30 日

誌謝

研究過程中遇到許多困難，感謝黃宇中老師給予很多的建議，不論是專業的知識或是研究的方向，許多地方都是老師教導後才能完成此論文，很感謝蔡坤霖老師擔任指導老師，提供一個良好的環境與指導角色給予我很大的幫助，實驗期間也常詢問老師的建議老師們也都很友善的回答我的問題，培養我跟他人溝通的能力，即使遇到困難也能透過溝通後解決。

實驗室裡也受到其他學長的幫忙，遇到一些邏輯上的問題互相討論後，更容易想出答案，營造一間環境融洽的實驗室，感謝他們的幫忙與支持。

感謝家人給予的支持與幫助，辛苦一路撫養我的媽媽提供我讀大學與研究所，弟弟也很孝順沒有亂闖禍，他們也是我努力的動力，很感謝師長與家人的幫忙，祝福大家都身體健康平安。

中文摘要

全向輪擁有極小的轉彎半徑並有便利的電子控制特點，故在自動化產業上的發展極其快速。但現有應用工具十分不便，因此本文探討建立一個以數學模型為基礎的全向輪高階控制指令集。

本實驗提供人性化的介面並設計指令控制全向輪的位移與旋轉，並對全向輪移動距離的誤差做了改善。藉由座標與路徑的規劃，使全向輪路徑控制可銜接 CAD 介面。實驗結果達成移動距離的誤差達到可接受的範圍內。此成果對產業有莫大的助益。

關鍵詞: 全向輪、麥克納姆輪、四輪運動模型。

Abstract

The omni wheel has a very small turning radius and convenient electronic control features. So the development of the automation industry is extremely fast. But the existing application tools are very inconvenient, so this paper explores related issues. Establishment of a mathematical model based on the omni wheel high level control instruction set. This result has great help to the industry.

This experiment provides a user-friendly interface and uses the instruction set to control the omni wheel movement. The error of the omni wheel moving distance is improved. Use the coordinates and path planning to round the wheel control is very convenient. The final experimental results make the error of the moving distance whitin 5%.

Key word: Omni-wheel robot,Mecanum-wheel robot,Four-wheel model movement.

目錄

圖目錄

表目錄

第一章　緒論

1.1　前言

全球自動化的演進越來越快速以及其需求不斷的增加如圖 1，如何在全球的市場上占領重要的地位是我們迫切需要解決的問題，全球發展工業自動化機器人的四大龍頭主要是瑞士 ABB、德國庫卡、日本 Fanuc、日本安川電機。工業型機器人的發展主要使用在汽車或是電子產業的地方，在工業自動化機器人的發展上台灣已經落後其他國家的腳步，我們可以繼續發展工業機器人或是往服務或是照護型機器人發展，但是不論在工業或是服務型機器人發展機器的移動是一定需要得，所以為了增加移動的效率發明了全向輪。

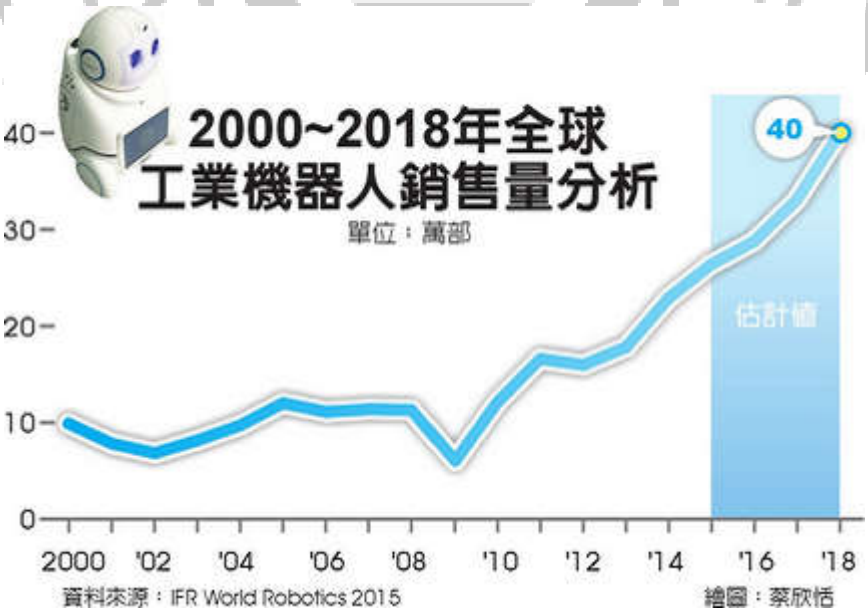


圖 1 全球機器人銷量

全向輪因為有零轉彎半徑的特性與電控裝置容易操作，能夠增加移動時的效率也增加了使用的方便性，為了達到全向移動的功能在發展上也有不同的類型的全向輪，目前較常見全向輪 Mecanum wheel 與 Omni wheel，Mecanum 車輪與轉軸相差 45 度角，由於四個輪子擺放角度不同利用分力相互抵消，可以讓平台達到各方向的移動，Omni 車輪則是在輪子上裝上與輪軸相差 90 度的多個小輪子，多數將兩個輪子合併，這樣可以減少誤差，使平台移動更加順暢。

這些全向輪有更好的移動機能，可以使用在探勘、救災、搬運、或是交通工具上……，現在全球的物流越來越興盛，能夠使用在工廠使物品有效的搬運，或是能夠改善我們目前的汽車，以經有不少工廠使用全向移動搬運車。

此實驗中使用的全向輪為麥克納姆輪，麥克納姆輪是 1973 年瑞士發明家 Bengt Ilon 在 Mecanum 公司所發明的，能夠達到任意自由方向移動的全向輪，這種輪子在車輪的外環加裝許多與軸心夾角 45 度的小輪子，車輪旋轉時這些小輪子會與地面接觸，會產生摩擦力此力可分為 X 分量與 Y 量，藉由改變輪子旋轉方向與轉速，可以控制 XY 分量，使全向輪達到各方向的移動。

1994 年 F. G. Pin 和 S. M. Killough 兩位學者提出全方位運動機器人，也就是三輪式的全向輪以 120 度的間距配置於移動裝置上，由於個全向輪的作用利之間有 120 度間距，使三輪式的全向輪可以利用合力往任一方向移動，其中三輪式全向輪特點可同時並且獨立控制旋轉和平移[1]，如何分配好三個輪子方向與轉速相當重要。

之後利用全向輪組裝成四輪的移動裝置，透過不同的轉速可以達到全向的移動，並且在麥克納姆輪上進行改良，原本從小輪的兩邊進行固定的麥克納姆輪較難在不平坦的地面移動，因此改良後的麥克納姆輪從小輪的中間做固定，經過改善更能適應於不平坦的地面。

1.2 相關研究

自從全向輪發明後，越來越多相關研究出現，有使用 Omni 做三輪式的移動平台、四輪的移動平台或是使用 Mecanum 輪作的四輪平台，而國內大多研究以 Omni 三輪式平台較多，三輪分別至於 120 角度擺放不同的角度，達到全向移動，只有三輪控制部分教四輪容易，但也容易產生偏斜的問題，所以加入定位系統來改善偏斜，研究上就有許多關於移動與定位的相關議題。

國內相關的全向輪平台的開發有參考[2]，雖然使用的是三輪式的全向輪平台，但是仍與我們系統類似，利用室內定定位的裝置透過

無線通訊設備使全向平台移動到特定位置，但其中還是有偏斜與不精確的問題存在，而其移動模式只有依照單一點進行移動，在此方面於我們的實驗中以多點的模式進行改善。

在三輪式全方位移動機器人之運動模型與控制系統中參考[1]，探討編碼器定位與感測器定位，其研究結果實測後顯示編碼器定位優於感測器定位，主因為感測器如三軸加速度計與電子羅盤均有誤差，隨著時間累加誤差也不斷放大，但是以上兩種方法都還是會有誤差，而我們的實驗中依據其成果採用增量式編碼器來完成定位的功能。

也有實驗使用相機鏡頭進行全向輪定位的方法參考[3]，其實驗結果也能控制全向輪平台到指定位置於五公分內的誤差，透過相機可以計算出速度與其位置，但其缺點於教正相機鏡頭參數繁雜，且受限於相機鏡頭攝影的範圍內，因此在這部分可以在進行改善。

全向輪型機器人之控制設計之 DSP 實踐參考[4]，利用 DSP 使用 C 語言達到全向輪的控制，並介紹其週邊電路，實驗結果達到全向輪機器人基本操作。

具無線監控能力之全向式輪型機器人設計與實作參考[5]，使用的式四輪式的全向輪其輪子為 Omni 型的輪子，設計出無線監控之機構、電路、韌體以及監控軟體。使用無線 IP 監視器傳送到個人 PC 軟體進行監看並使用 PS2 搖桿遠端遙控全向輪。

Development of Mobile Robot Drive System using Mecanum Wheels 參考[6]提到利用控制器與感測器如何完成整個全向輪的架構，並且利用搖桿完成對全向輪的控制。

A Robust Adaptive Control of Mecanum Wheel Mobile Robot: Simulation and Experimental Validation 參考[7]，利用強健控制法減少全向輪移動誤差，其實驗中使用 kinect 和 IMU 作為位置與方向的感測器，回傳其參數並利用閉迴路方式改善其輸入電壓達到減少移動誤差的目的。

Driving Control of Mobile Robot with Mecanum Wheel using Fuzzy Inference System參考[8]，提供動態模型的原理並利用模糊控制控制全向輪，因為全向輪常有滑動因此很難計算其移動的準確值，其論文先計算發生滑動時所造成的速度與與角速度關係，在全向輪實際發生滑動時使全向輪預期做較準，減少滑動時所造成的移動誤差。

Research and Development of Mecanum-wheeled Omnidirectional Mobile Robot Implemented by Multiple Control Methods參考[9]，此篇論文利用多種無線通訊的方法對全向輪進行控制，並講述其方法的優缺點。所使用的通訊有藍芽、紅外線遙控制、語音辨識系統，透過這些可以對全向輪進行無線操控，但是這些方法都有其優缺點，可以應用在不同的場合。

1.3 研究動機與目的

現代倉儲與搬運系統需求的效率越來越高，不僅需要自動化還要高效率，在各大電子商務廠商競爭下效率成為很重要的一環，而全向輪系統是增加效率的辦法之一，因為其擁有 0 轉彎半徑的特點以及其控制架構非常容易，只需要電控的方式就能達到全向的移動，在物流工廠中，有更好的效率達到指定位置，發展全向輪移動平台的開發 增加其便利性是本篇論文的動機。

全向輪應用的層面非常廣泛，目前多使用在搬運系統上，但是多使用於有引導式的軌道，其中有許多方法，磁帶導引、電磁導引、光帶導引都是使移動平台行走於特定路線，為了增加移動平台的適應性與方便性，研究如何不必局限於特定路線為此目的，並且可以適用在平坦的工廠中。

為了增加全向輪的方便性與更容易控制全向輪的移動，以下是本文實驗所要解決的問題：

- 提供人性化的介面並使用指令就能夠控制全向輪的移動。
- 修正全向輪移動的距離，將其移動的誤差減少。
- 能夠利用座標或是路徑規劃讓全向輪進行移動。

1.4 章節安排

本論文共分成五個章節:

第一章　緒論介紹了全向輪的歷史與應用,並說明本文的動機與目的。

第二章　推導全向輪的原理,討論輪子與整個全向輪移動的關係。

第三章　介紹本實驗所使用的系統架構,包含硬體與軟體部分,並且列出所使用的程式範例。

第四章　提出全向輪的數學矩陣。明確列出本文所設計的指令規範,包括使用以及範例。使用指令後所執行的動作,對其移動距離做改善並整理其方向控制。

第五章　對整體實驗的結果展現,並找出可以改善的地方於未來展望中進行討論。

第二章　全向輪原理

討論四輪平台移動時的運動狀況，因輪子上面都還有加入其他斜角 45 度的小輪子，所以除了輪子本來的速度之外還要再加入小輪子速，度整個運動模型如圖 2。若輪子轉動會產生 V 和 V_{ir} 兩個速度，由於四個輪子上的小輪子方向不同，再根據不同運動模式狀態下，左右的力可以互相抵銷，就可以達到全向的移動。

根據四顆輪子的方向及轉速，就可以改變整個平台的移動，整個平台的速度可以分為水平速度 V_y、垂直速度 V_x 與角速度 ω 還有所移動的角度 θ，推導出四顆輪子與整個平台的移動關係，此數學驗證參考[10]。

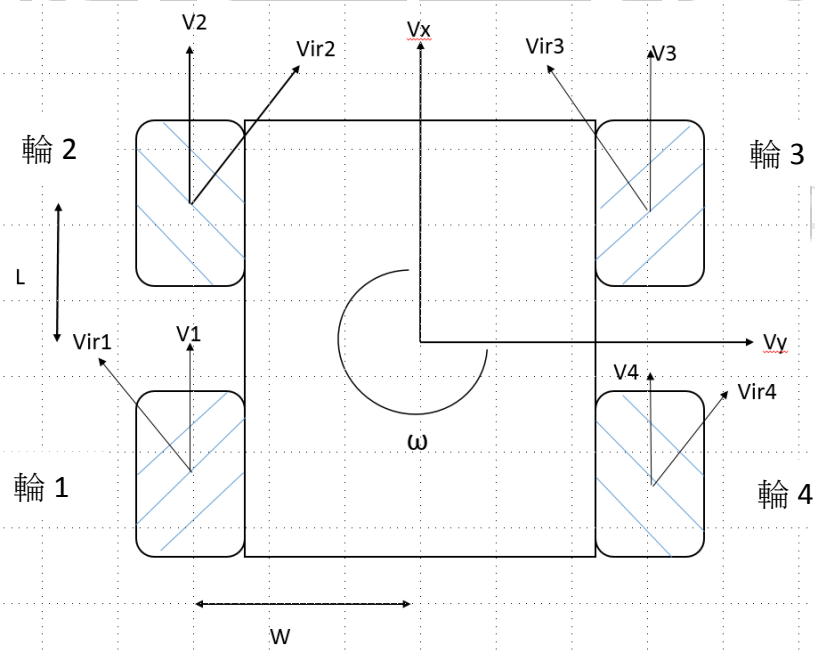


圖 2 全向輪運動定義

首先我們先討論單一輪子轉動時所產生的速度與輪子關係，在經由每個輪子的轉動所產生的 X、Y 方向的力對全向輪的影響進行討論，並且將運動模型以數學模型表示。

參考[7]結構在平面的運動上主要分為:

X 軸方向、Y 軸方向、ω軸自轉。

V_x　　表示 X 軸運動的速度，即左右方向，定義向右為正

V_y　　Vy 表示 Y 軸運動的速度，即前後方向，定義向前為正

ω　　表示　自轉的角速度，定義逆時針為正

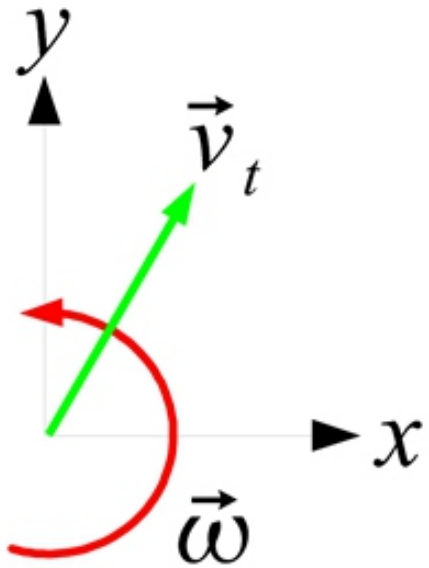


圖 3 單一輪子運動定義

定義：

$\vec{r}$　　為從幾何中心指向輪子軸心的值；

$\vec{v}$　　為輪子軸心的運動速度值；

$\vec{v}_r$　　為垂直於軸心延垂直於$\vec{r}$的方向的速度直;

那麼可以計算出：

$$\vec{v} = \vec{v}_t + \vec{\omega} \times \vec{r}$$

分別計算 X、Y 軸的值為：

$$\begin{cases} v_x = v_{t_x} - \omega \cdot r_y \\ v_y = v_{t_y} + \omega \cdot r_x \end{cases}$$

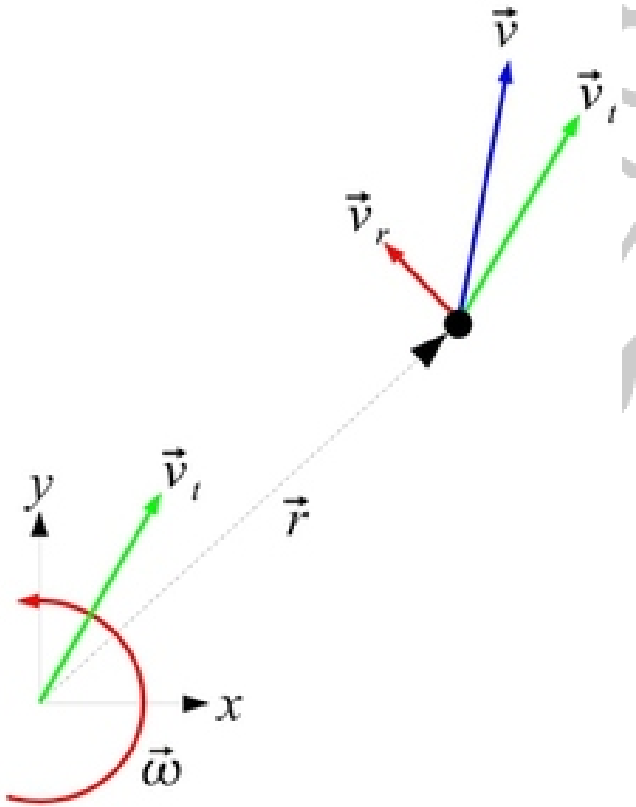


圖 4 單一輪子運動狀況

同理可以算出其他三個輪子軸心的速度。

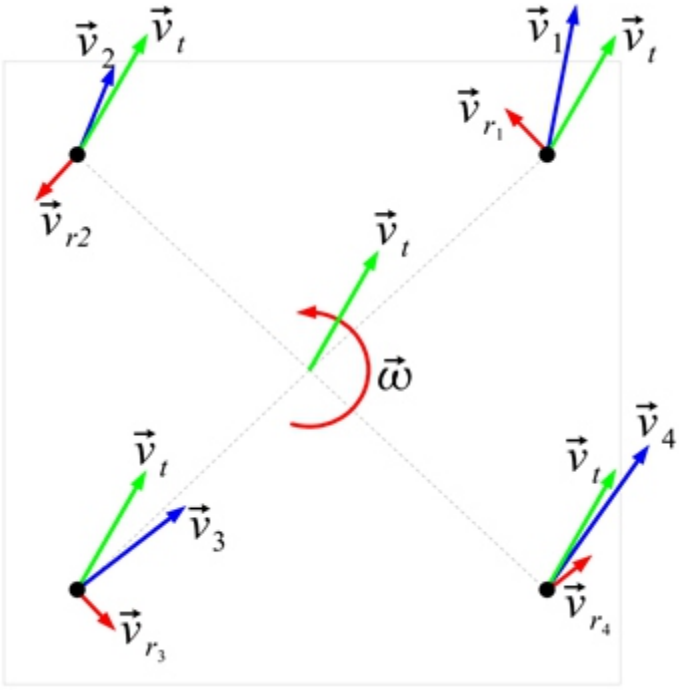


圖 5 四輪運動定義

這邊算出單一個輪子與軸中心的關係，並且利用將此關係變成公式，而全向輪有四顆馬達，所以根據式子也可以推算出總共四顆馬達的公式，但是實驗中的全向輪為長方型的架構，會根據長寬導致馬達對軸心有不同的作用力。

接著要將馬達的速度改為全向輪輪子，因全向輪的會有不同的方向，關於這部分將由下面一章做解釋。

單一輪子的轉動與速度關係如圖 6:

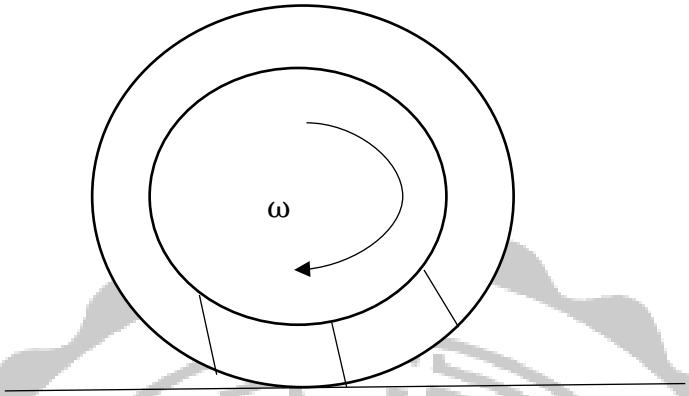


圖 6 單一全向輪運動狀況

當輪子轉動的角速度為ω，會造成小輪子的轉動與大輪的轉動，小輪所產生的速度為 V_{ir}，將速度分為 X 方向與 Y 方向，輪子轉動時與 V_{ir} 的關係:

$$V_{ir} = \omega * \text{小輪半徑 } r \tag{2.1}$$

$$V_{ir1x} = V_{ir1} * \cos 45° \tag{2.2}$$

$$V_{ir1y} = V_{ir1} * \sin 45° \tag{2.3}$$

大輪產生的速度為

$$V = \omega * R \tag{2.4}$$

將單一輪子轉動時小輪子與大輪子相加後 XY 方向的力

$$V_x = V + V_{irx} \tag{2.5}$$

$$V_y = V_{iry} \tag{2.6}$$

根據式子(2.5)、(2.6)推算出單一輪子轉動時造成的速度後，從圖可以得知四輪裝置在車體的位置，而全向輪的整體速度包括ＸＹ方向的速度與角速度，全向輪整體速度則由四顆輪子轉動所造成，以下討論四顆輪子對全向輪整體速度造成的影響。

根據上述式子，可以整理每個輪子單一旋轉狀況如下

$$V_{1x} = V_1 + V_{ir1x} \quad (2.7)$$

$$V_{1y} = -V_{ir1y} \quad (2.8)$$

$$V_{2x} = V_2 + V_{ir2x} \quad (2.9)$$

$$V_{2y} = V_{ir2y} \quad (2.10)$$

$$V_{3x} = V_3 + V_{ir3x} \quad (2.11)$$

$$V_{3y} = -V_{ir3y} \quad (2.12)$$

$$V_{4x} = V_4 + V_{ir4x} \quad (2.13)$$

$$V_{4y} = V_{ir4y} \quad (2.14)$$

因為四個輪子擺放在全向輪的位置不同以及車體的長寬不同，會造成不同方向的力與速度，輪子對於全向輪的整體速度的影響其結果如下。

$$V_{1x} = V_x + W * \omega \quad (2.15)$$

$$V_{1y} = V_y + L * \omega \quad (2.16)$$

$$V_{2x} = V_x + W * \omega \quad (2.17)$$

$$V_{2y} = V_y - L * \omega \quad (2.18)$$

$$V_{3x} = V_x - W * \omega \quad (2.19)$$

$$V_{3y} = V_y - L * \omega \quad (2.20)$$

$$V_{4x} = V_x - W * \omega \quad (2.21)$$

$$V_{4y} = V_y + L * \omega \quad (2.22)$$

經過式子(2.15)~(2.22)整理可以得到，單一輪子速度可由全向輪的ｘｙ方向與角速度所組成，其結果如下

$$V_1 = V_x + V_y + (L + W) * \omega \quad (2.23)$$

$$V_2 = V_x - V_y + (L + W) * \omega \quad (2.24)$$

$$V_3 = V_x + V_y - (L + W) * \omega \quad (2.25)$$

$$V_4 = V_x - V_y - (L + W) * \omega \quad (2.26)$$

可以得知當全向輪移動時，四顆輪子與全向輪整體速度的關係，輪子的轉動速度跟全向輪的ｘｙ方向與角速度和距離中新的距離有關，為了增加式子可讀性將式子轉換成矩陣:

$$\begin{bmatrix} V1 \\ V2 \\ V3 \\ V4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & (L+W) \\ 1 & -1 & (L+W) \\ 1 & 1 & -(L+W) \\ 1 & -1 & -(L+W) \end{bmatrix} \begin{bmatrix} Vx \\ Vy \\ \omega \end{bmatrix} \quad (2.27)$$

將式子(2.27)經過反矩陣轉換：

$$\begin{bmatrix} Vx \\ Vy \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 1 & (L+W) \\ 1 & -1 & (L+W) \\ 1 & 1 & -(L+W) \\ 1 & -1 & -(L+W) \end{bmatrix}^{-1} \begin{bmatrix} V1 \\ V2 \\ V3 \\ V4 \end{bmatrix} \quad (2.28)$$

將輪子的速度轉換為角速度ω，這邊假設輪子的半徑為 r，經過整理:

$$\begin{bmatrix} Vx \\ Vy \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 1 & (L+W) \\ 1 & -1 & (L+W) \\ 1 & 1 & -(L+W) \\ 1 & -1 & -(L+W) \end{bmatrix}^{-1} \begin{bmatrix} r\omega 1 \\ r\omega 2 \\ r\omega 3 \\ r\omega 4 \end{bmatrix} \quad (2.29)$$

$$V_x = \frac{r}{4} * (\omega 1 + \omega 2 + \omega 3 + \omega 4) \quad (2.30)$$

$$V_y = \frac{r}{4} * (-\omega 1 + \omega 2 - \omega 3 + \omega 4) \quad (2.31)$$

$$\omega = \frac{r}{4(W+L)} * (\omega 1 + \omega 2 + \omega 3 + \omega 4) \quad (2.32)$$

$$\mathrm{dx} = \int_{t0}^{t} vx\, dx \quad (2.33)$$

$$\mathrm{dy} = \int_{t0}^{t} vy\, dx \quad (2.34)$$

透過數學分析可以得知全向輪的 X、Y 方向與輪子的關係，全向輪的速度根據輪子擺放的方向有關，因為四輪麥克納姆全向輪有不同的擺放方法，透過公式可以了解擺放方向對全向輪的影響，實驗中我們的擺放方法，全向輪的 X、Y 方向速度與四輪的角速度和輪子半徑相關，而全向輪整體的角速度還需考慮全向輪車體的長寬，透過方程式更容易理解輪子與車體運動關係更容易達到全向輪的控制。

根據全向輪的動態模型的推導可以推算出全向輪移動時的狀態，底下列出全向輪主要移動的方向，透過改變輪子的方向與轉速達到不同的運度狀態如圖 7，要推算出運動時的狀況都可以利用上述的公式得知不同運動時的狀況。

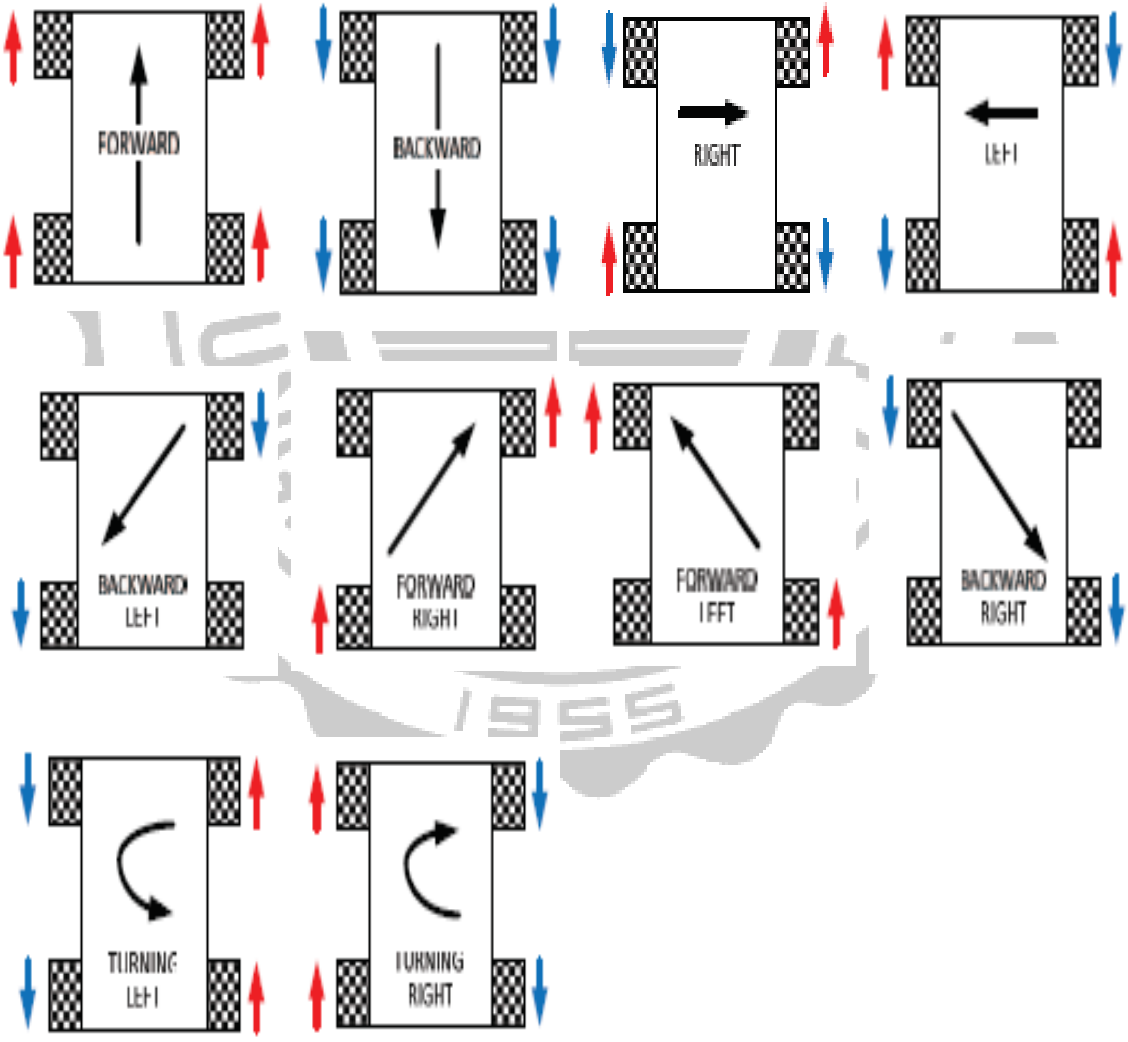


圖 7 全向輪運動方向[11]

第三章　系統架構

系統上可以分為軟體與硬體，軟體部分有輸入的簡意介面與程式編輯和燒錄軟體，硬體則為整個全向輪的架構包含控制器、感測器與全向輪，藉由軟體部分控制全向輪硬體的系統在本章節描述。

3.1 硬體配置

本文使用四顆麥克納姆輪作為全向輪的輪子，利用直流馬達控制輪子轉向與轉速，根據不同轉向與轉速達到全向的移動，全向輪的硬體配置如圖 8，

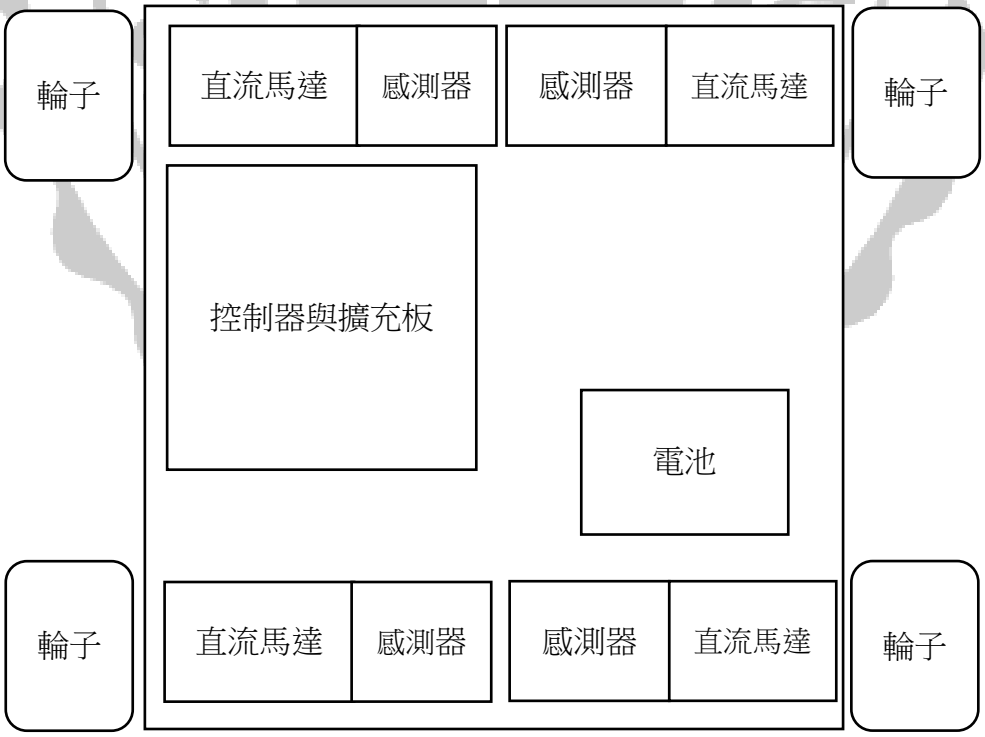


圖 8 全向輪硬體配置

利用控制器經過馬達驅動器對直流馬達進行方向與轉速控制，馬達後端裝有光學式增量型感測器，光學式增量型感測器主要有兩個輸出腳位，透過兩輸出可以得知馬達目前的方向與速度，控制的架構圖如圖 9。

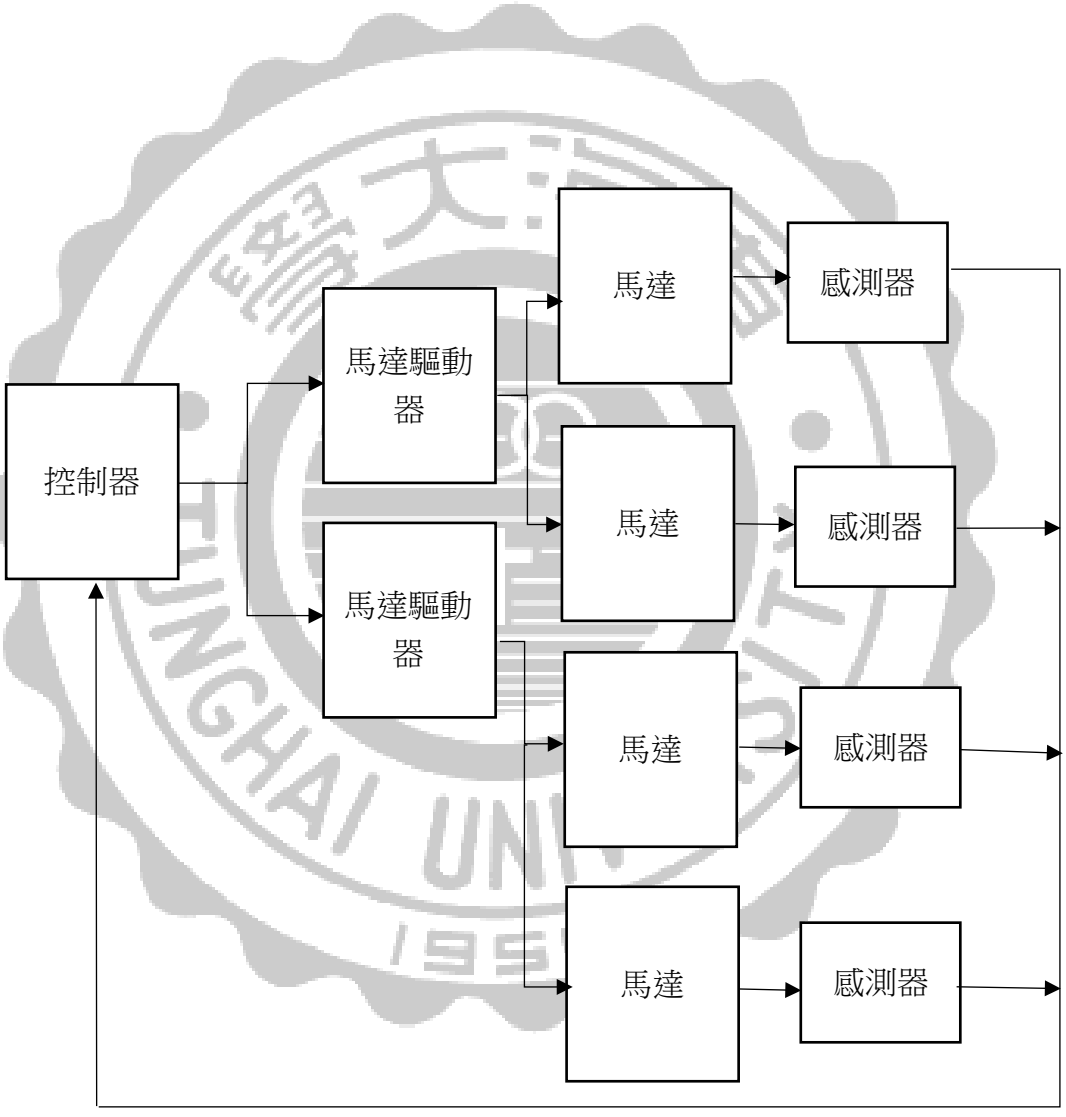


圖 9 硬體控制方塊圖

直流馬達與感測器

馬達使用 12 伏特直流馬達並搭配馬達驅動器 L298 作為馬達驅動 IC，馬達在沒有附載的狀況下其轉速高達 9600 RPM/minute，位了增加馬達扭力以驅動全向輪在馬達部分增加減速齒輪，齒輪比為 80:1 加入齒輪後扭力增加但其速度減少為 120 RPM/minute。

感測器加裝在四顆直流馬達的後端功能為感測馬達的轉速，使用的種類為光學式增量型感測器其輸出向為訊號為 A、B 解析度 2CPR。

馬達的規格如下表 1:

表 1 直流馬達規格

Type	12 V
No load speed	9600 RPM/minute
Reduced speed	120 RPM/minute
Gearbox ratio	80:1
Continuous torque	0.5 Nm
Stall torque	1.6 Nm

3.2 軟體架構

軟體部分使用 Arduino 進行編輯與燒錄本文提供了一個更簡易的全向輪操作介面，利用 txt 檔進行編輯且利用精簡的指令集就可以輸入給 Arduino 達到控制的目的，其控制的方塊圖如下。

圖 10 軟體架構

全向輪的軟體部分 NEXUS ROBOT 公司[12]也提供一些基礎範例程式以利於操作全向輪的移動，基礎範例程式的指令如下。

單一馬達的轉動方向與速度控制:

unsigned int **setSpeedRPM**(int speedRPM,bool dir)

Set the speed and direction for the motor

Parameters:	int speedRPM The speed set for Motor Bool dir The direction set for Motor
See:	**Motor:: PIDSetSpeedRPMDesired**() **Motor::setDesiredDir**() **Motor::getSpeedRPM**()

unsigned int **runPWM**(unsigned int PWM,bool dir,bool saveDir=true);

Set the PWM and direction for Motors .then return the PWM.

Parameters:	unsigned int PWM The PWM set for the Motor Bool dir The direction set for the Motor Bool saveDir A flag to confirm if the direction will be reset
Return:	PWM

unsigned int **advancePWM**(unsigned int PWM)

Set the pwm when the motor run advance

Parameters:	unsigned int PWM The PWM set for Motor
Return:	runPWM(PWM,DIR_ADVANCE)

unsigned int **backoffPWM**(unsigned int PWM);

Set the pwm when the motor backoff

Parameters:	unsigned int PWM The PWM set for Motor
See:	**Motor:: runPWM**()

unsigned int **getSpeedRPM**() const

Get the speed of the motor (round per minute)

See:	**SPEEDPPS2SPEEDRPM**().

unsigned int **setMotorAll**(unsigned int speedMMPS=0,bool dir=DIR_ADVANCE)

Set all motors' speed and direction

Parameters:	unsigned int speedMMPS=0 The speed for the motor run,initialize it. bool dir=DIR_ADVANCE The direction the motor run
See:	**Omni4WD:: wheelULSetSpeedMMPS**() **Omni4WD::wheelLLSetSpeedMMPS**() **Omni4WD::wheelLRSetSpeedMMPS**() **Omni4WD::wheelURSetSpeedMMPS**()

unsigned int **setMotorAllStop**()

Set all Motors stop

This will lie within the range specified at **Omni4WD::setMotorAll**()

See:	**Omni4WD::setMotorAll**()

unsigned int **setMotorAllAdvance**(unsigned int speedMMPS=0)

Set all motors run forward

Parameters:	unsigned int speedMMPS=0 The speed for the motor run,initialize it.
See:	**Omni4WD::setMotorAll**()

unsigned int **setMotorAllBackoff**(unsigned int speedMMPS=0)

Set all motors run back off

Parameters:	unsigned int speedMMPS=0 The speed for the motor run,initialize it.
See:	**Omni4WD::setMotorAll**()

unsigned int **setCarRight**(unsigned int speedMMPS=0)

Set the car turn right

Parameters:	unsigned int speedMMPS=0 The speed for the motor run,initialize it.
See:	**Omni4WD::wheelULSetSpeedMMPS**() **Omni4WD::wheelLLSetSpeedMMPS**() **Omni4WD::wheelLRSetSpeedMMPS**() **Omni4WD::wheelURSetSpeedMMPS**() **Omni4WD::setCarstat**()

unsigned int **setCarRotateLeft**(unsigned int speedMMPS=0)

Set the car rotate left

Parameters:	unsigned int speedMMPS=0 The speed for the car moves,initialize it.
See:	**Omni4WD::setCarstat**() **Omni4WD::setMotorAllBackoff**()

unsigned int **setCarRotateRight**(unsigned int speedMMPS=0)

Set the car for rotate right

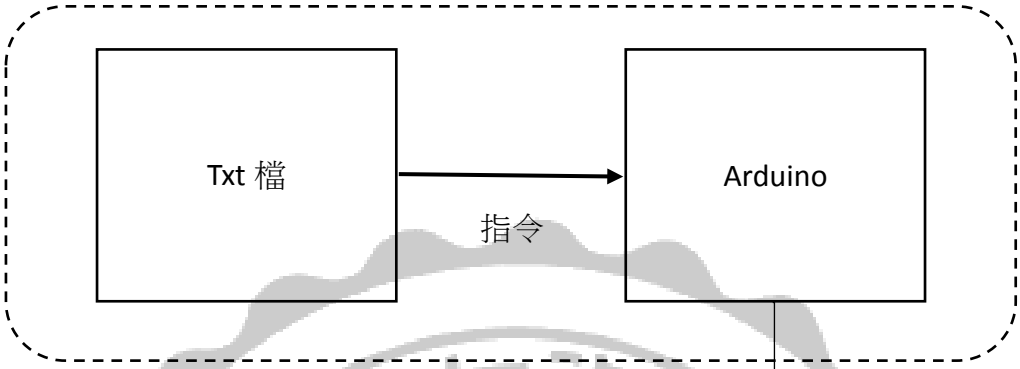
Parameters:	unsigned int speedMMPS=0 The speed for the car moves,initialize it.
See:	**Omni4WD::setCarstat**() **Omni4WD::setMotorAllAdvance**()

3.3 系統架構

整個系統架構圖如圖 11，利用上述的元件所組成，其中在馬達到輪子上加裝了減速齒輪為了增加力矩使的容易推動輪子，雖然馬達的轉速變慢了，但是這樣容易推動輪子，避免只有馬達轉動但是輪子不轉動的問題。

整個系統的包含軟體與軟體，從 TXT 檔中進行編輯利用本文所提供的指令集以具其規範使用，傳輸到 Arduino 進行分析後可以達到使用者所輸入的指令動作，經過 Arduino 後轉為硬體部分，主要是控制馬達與感測器的資料傳輸。透過馬達驅動 IC 驅動直流馬達並使用光感測器檢測馬達移動距離，檢測是否到達使用者所輸入距離，在繼續執行下一個所輸入的全向輪動作。

軟體架構



硬體架構

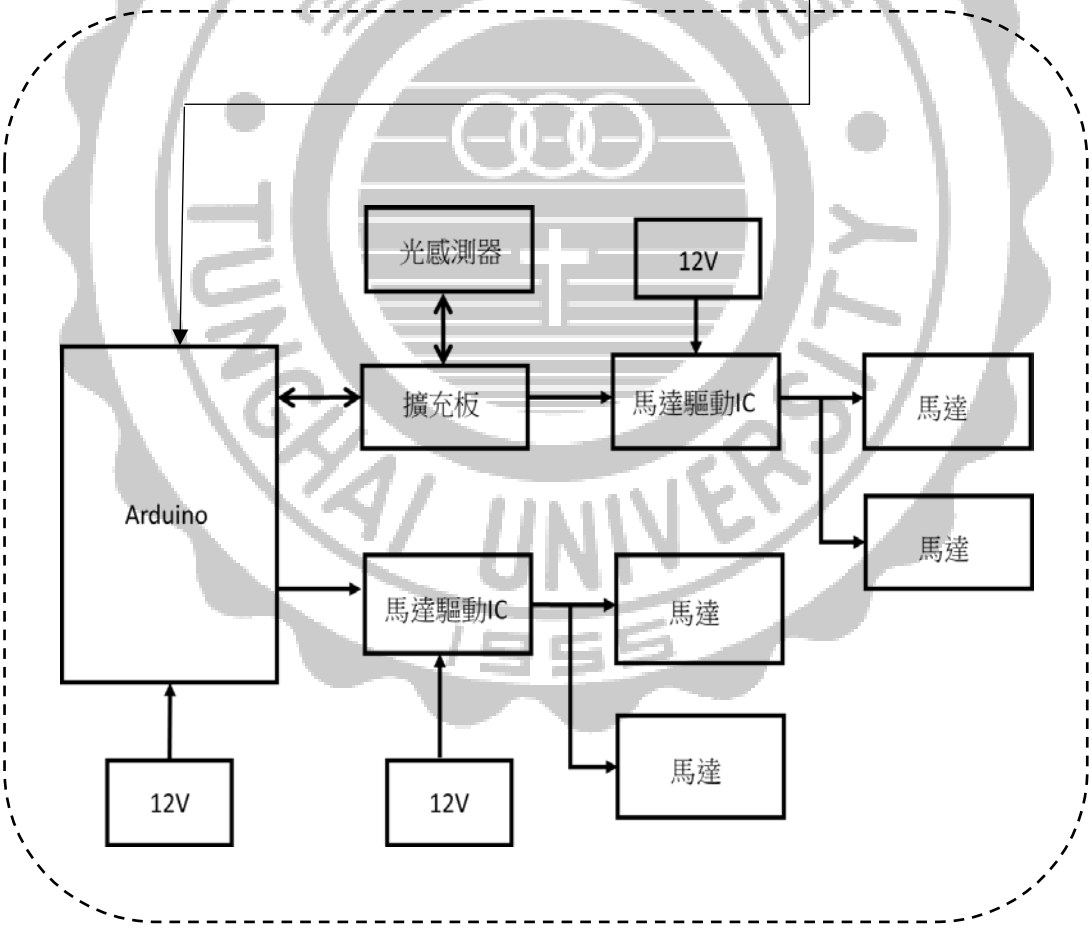


圖 11　整體系統方塊圖

第四章　設計與實驗

本章節將指令集的功能與說明如何使用，透過此指令控制全向輪並且修正全向輪的移動距離達到輸入的較少的誤差，全向輪經過實際量測後誤差可以到達 5%，結果將由此章說明。

4.1 指令集

透過這個指令可以控制全向輪，僅需要在 TXT 檔中按照指令集規範經過燒錄後就能達到指令功能，提供了一個更容易編寫的高階語言，讓使用者輕易的了解指令功能並快速的能夠自由操作全向輪，此指令由英文代碼和一組數字組成，英文代碼為全向輪運動的方式數字部分為全向輪根據運動方式所移動的距離，以下提供精簡指令集的規範，包含指令格式、說明與範例。

A(distance mm): 全向輪直線前進距離(mm)

指令格式

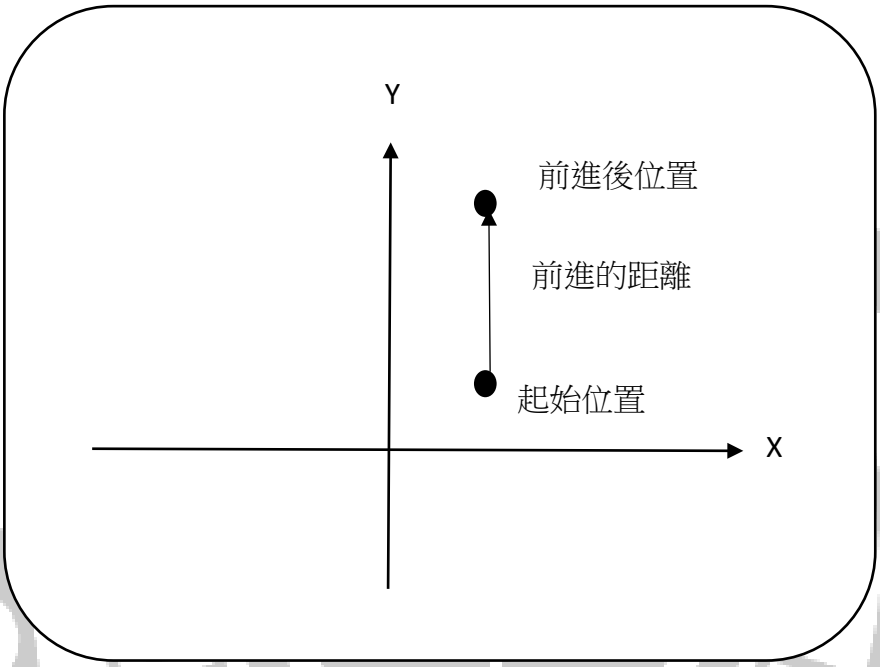
A (distance mm);

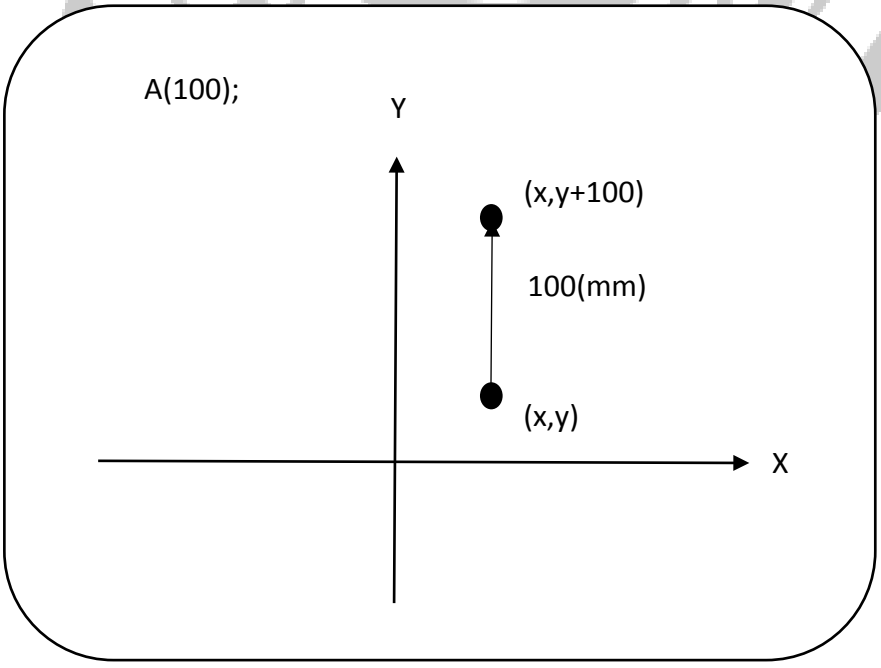
distance mm: 依據當前指令做絕對移動距離。

說明

全向輪直線前進所輸入的距離值，單位為毫米。

範例





B(distance mm): 全向輪直線退後距離(mm)

指令格式

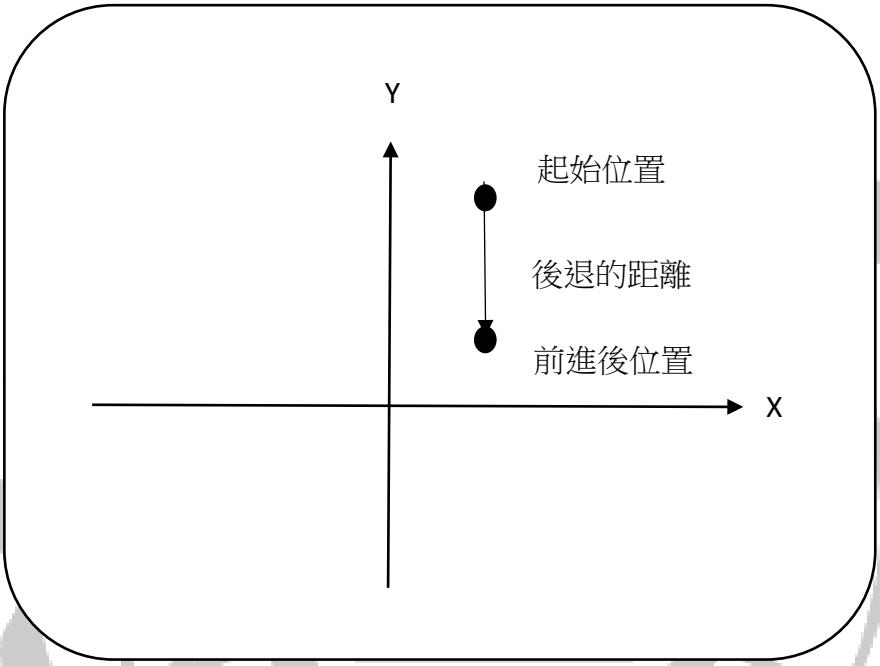
B (distance mm);

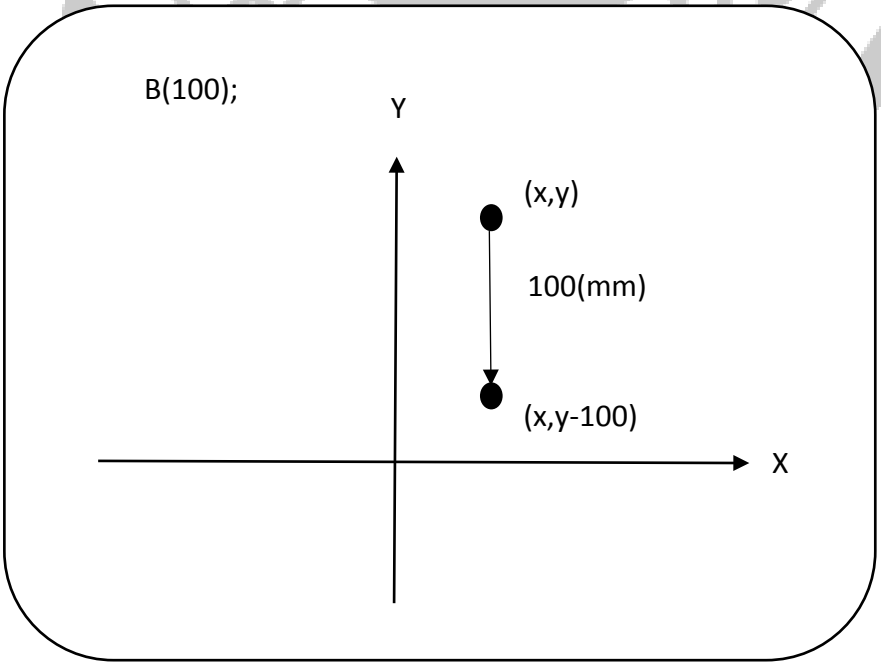
distance mm: 依據當前指令做絕對移動距離。

說明

全向輪直線退後所輸入的距離值，單位為毫米。

範例





L(distance mm): 全向輪水平左移距離(mm)

指令格式

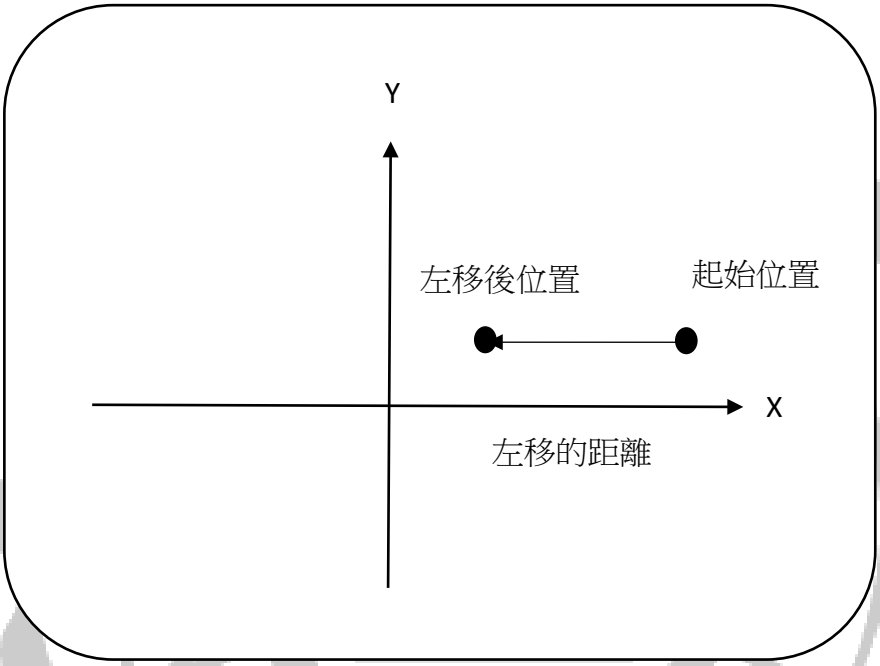
L (distance mm);

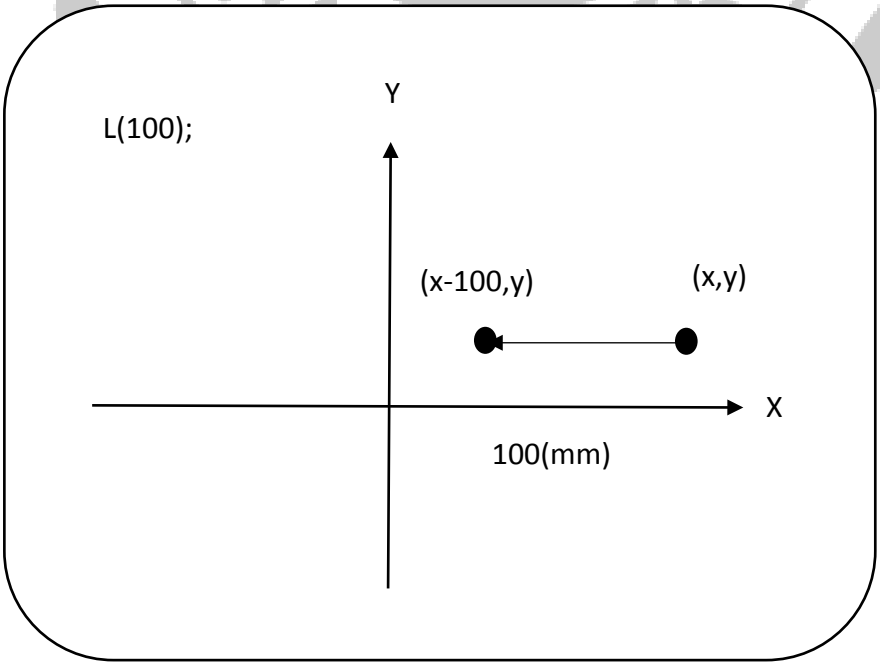
distance mm: 依據當前指令做絕對移動距離。

說明

全向輪水平左移所輸入的距離值，單位為毫米。

範例





R(distance mm): 全向輪水平右移距離(mm)

指令格式

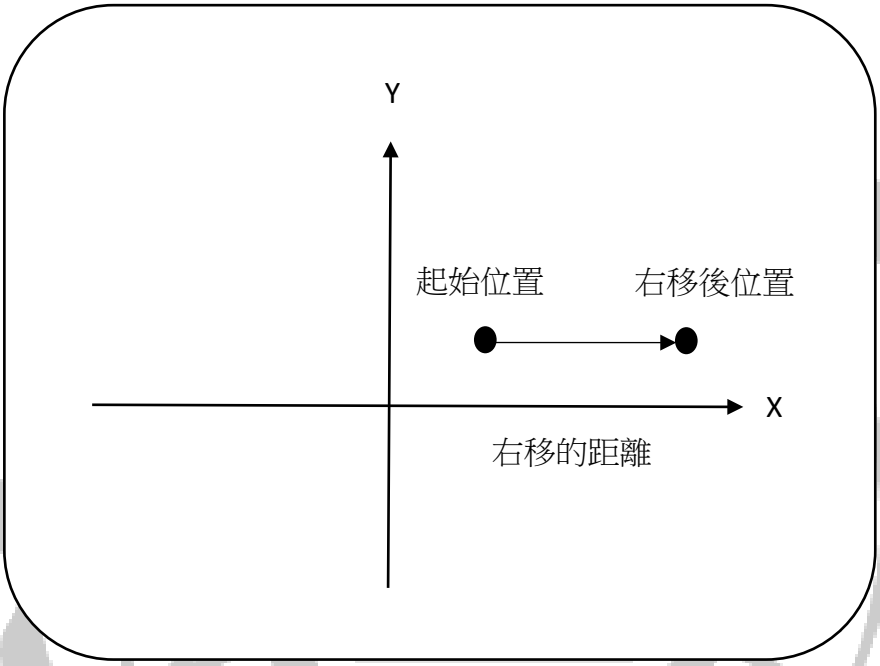
R (distance mm);

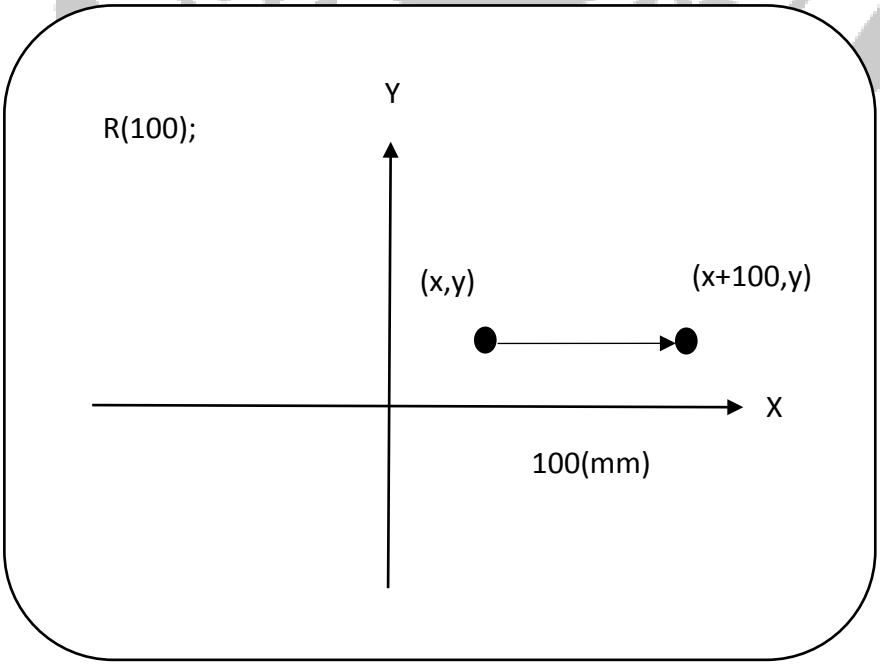
distance mm: 依據當前指令做絕對移動距離。

說明

全向輪水平右移所輸入的距離值，單位為毫米。

範例





RA(distance mm): 全向輪往第一象限 45 度移動距離(mm)

指令格式

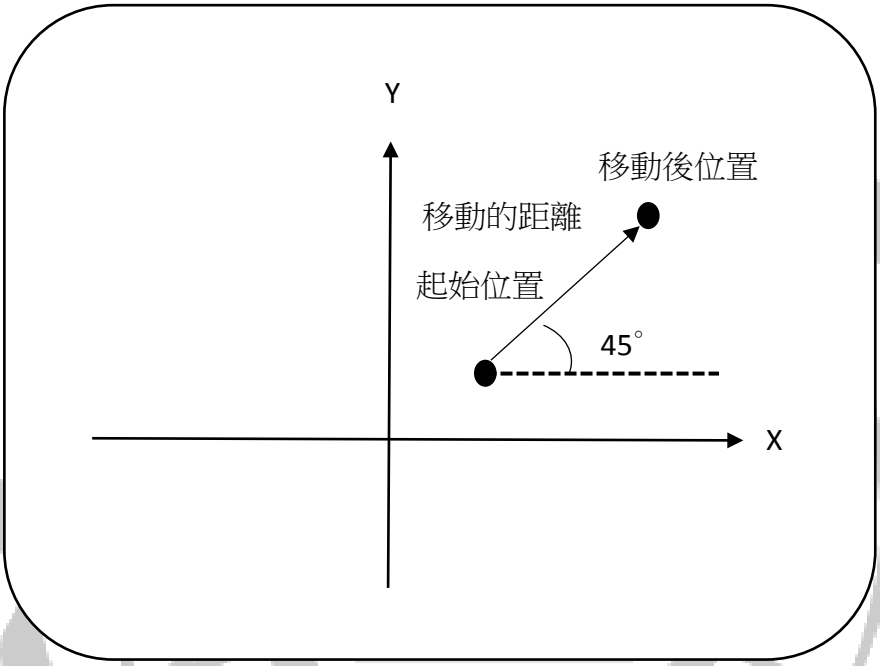
RA (distance mm);

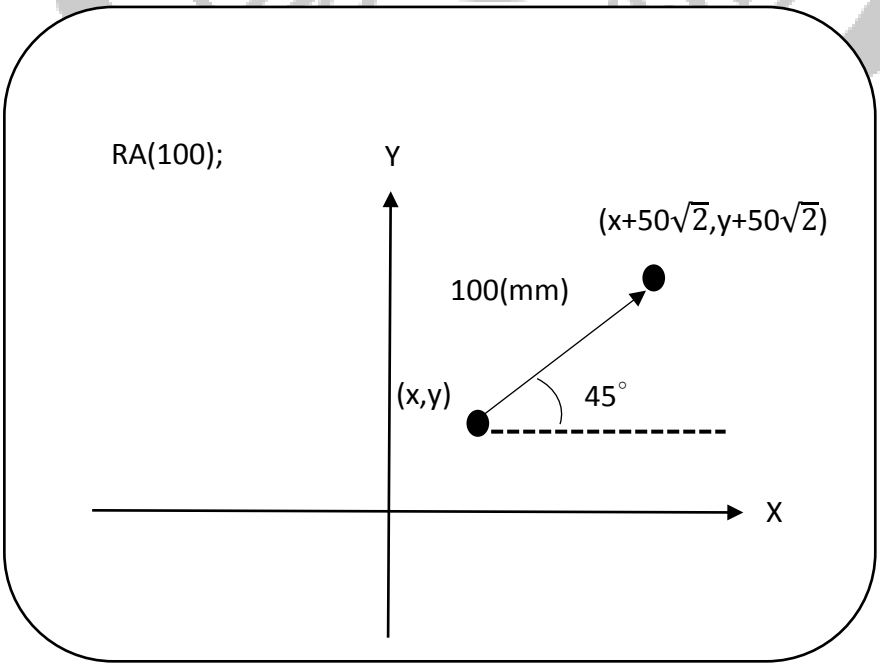
distance mm: 依據當前指令做絕對移動距離。

說明

全向輪往第一象限 45 度所輸入的距離值，單位為毫米。

範例





RB(distance mm): 全向輪往第四象限 45 度移動距離(mm)

指令格式

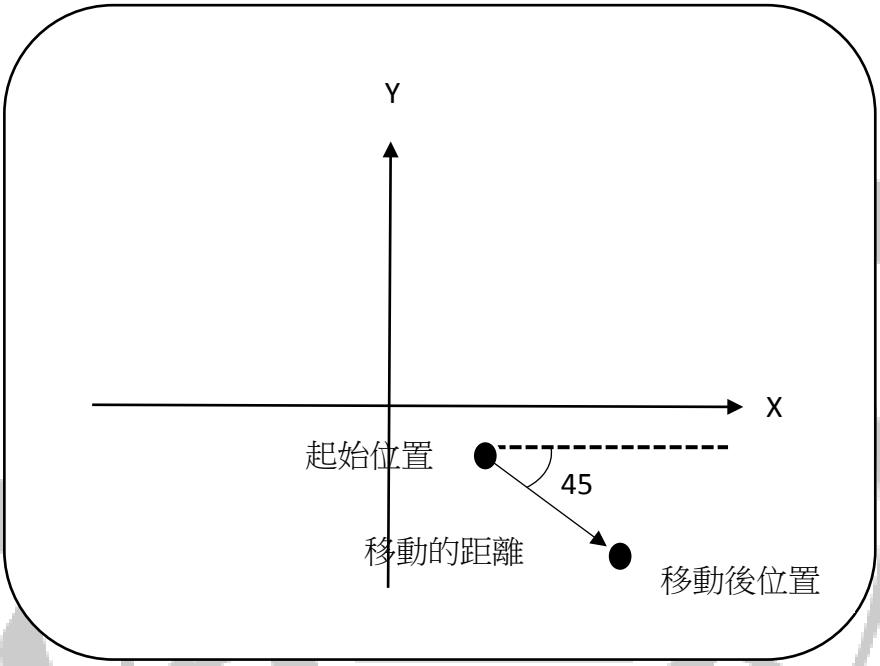
RB (distance mm);

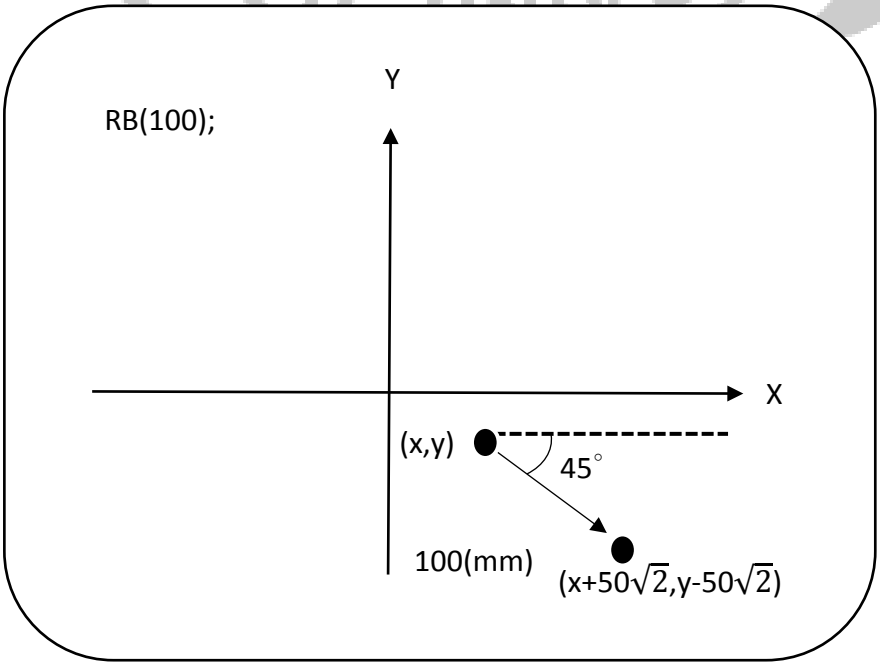
distance mm: 依據當前指令做絕對移動距離。

說明

全向輪往第四象限 45 度所輸入的距離值，單位為毫米。

範例





LB(distance mm): 全向輪往第三象限 45 度移動距離(mm)

指令格式

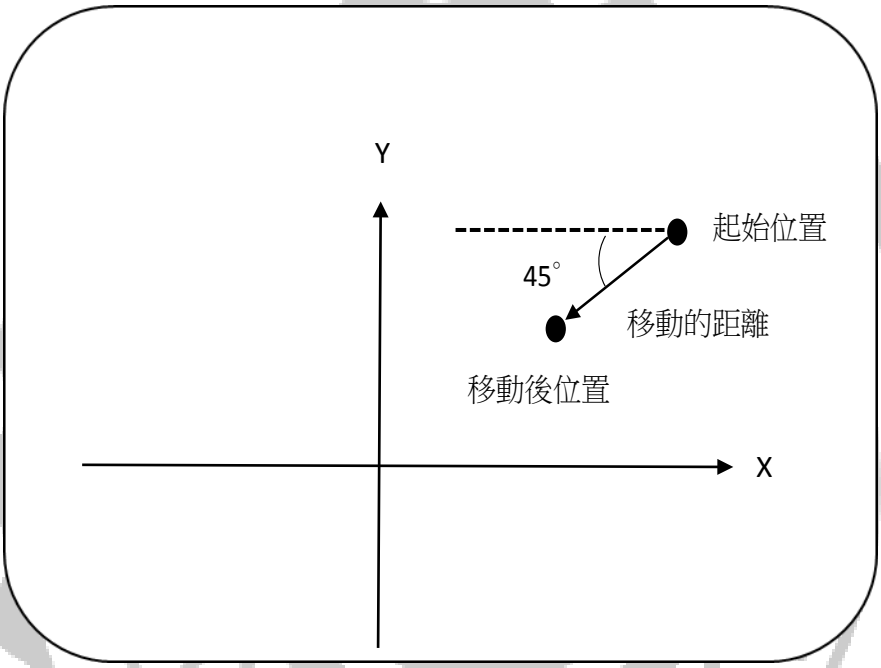
LB (distance mm);

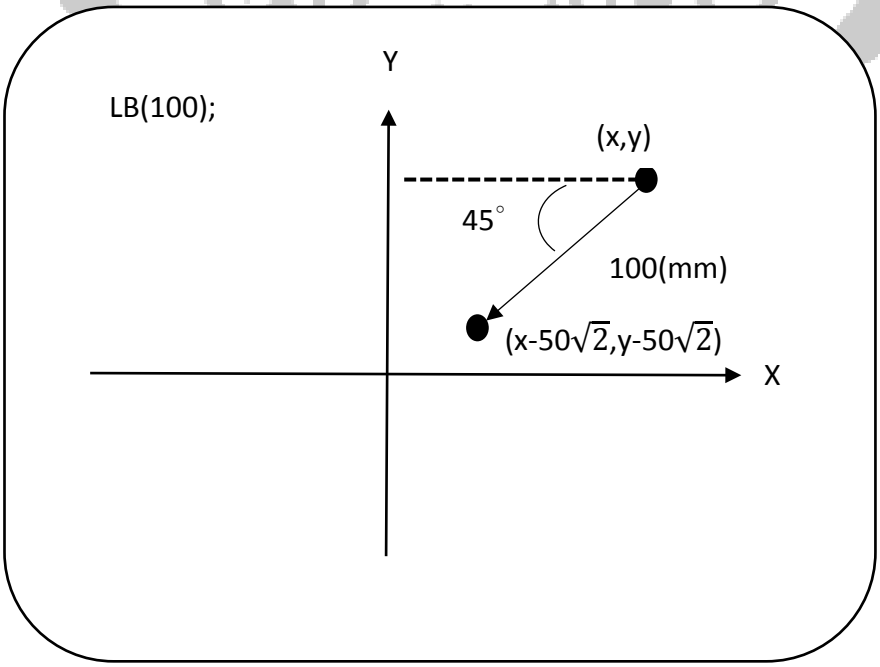
distance mm: 依據當前指令做絕對移動距離。

說明

全向輪往第三象限 45 度所輸入的距離值，單位為毫米。

範例





實際編輯時如圖，按照指令集的規範於記事本進行編輯，如範例中全向輪第一步會前進1500mm在來後退1000mm在向右500mm在往左200mm完成了全向輪的操作如圖12。

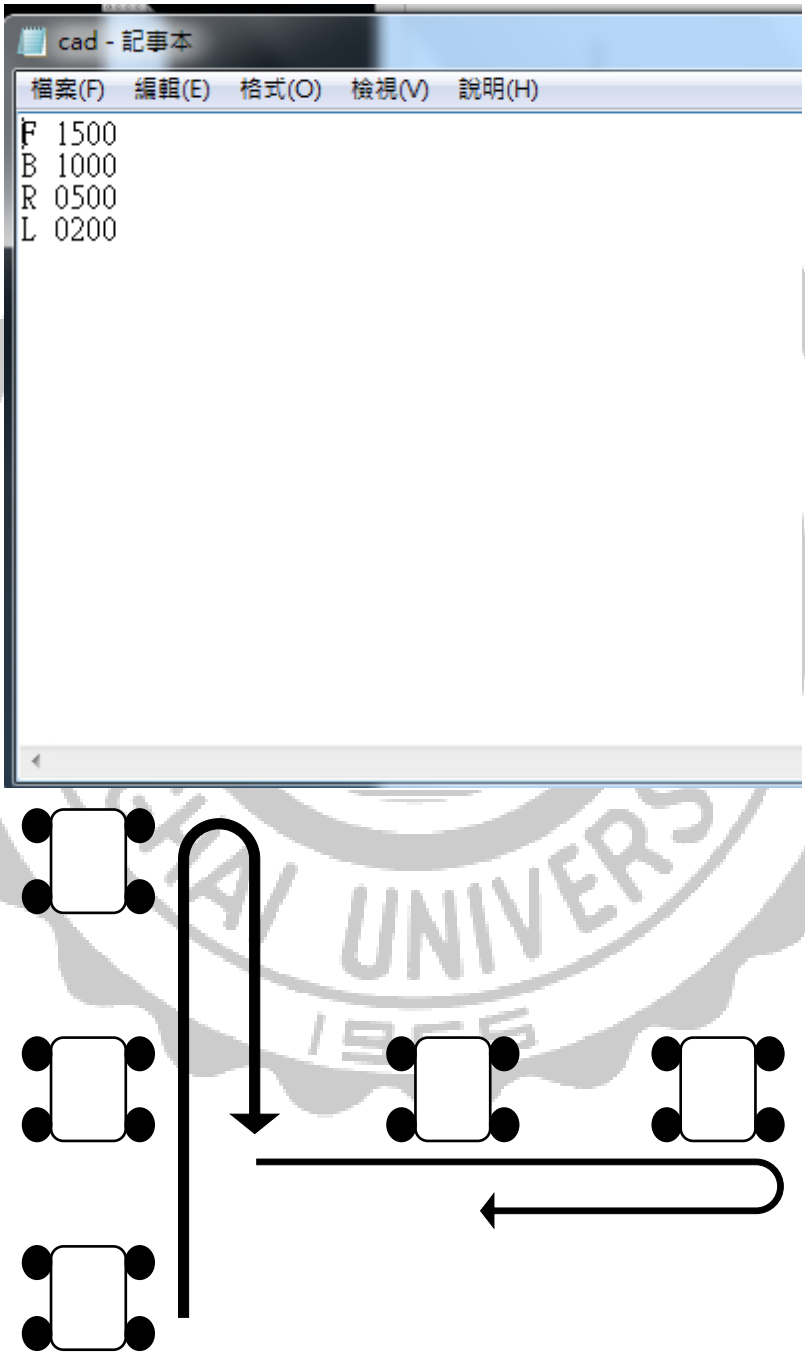


圖 12 指令集使用與運動狀況

由圖可知實驗成功透過精簡指令成功控制全向輪的移動，所以使用者可以透過這些指令更輕易的控制全向輪，只要在記事本上做編輯即可以完成並輸入指令動作給全向輪，本文提供了一些基礎的全向輪的運動模式，之後可以在增加更多的運動模式增加其功能。

根據上述的實驗可以成功透過指令進行移動，但是經過實際量測後，會發現在輸入端輸入的距離與實際行走的距離會有所誤差，為了讓全向輪移動順利達到我們所輸入的值，需修正其移動的距離。

其中導致全向輪的移動有誤差可能的原因有:

- 機構上的不平衡
- 馬達的空轉
- 馬達停止後輪子會繼續滑行
- 全向輪整體移動時產生的偏斜
- 感測器本身的誤差

可能造成全向輪移動的誤差如上，有些問題可以透過硬體上修改做修正有些可以從軟體上面做修正，但是實驗中主要使用軟體的修正方法，透過實際量測的方法考慮其誤差加入軟體進修正，完整的修正會在下面章節做說明。

4.2 方向控制

整理出全向輪平台行走不同的方向時四個輪子的狀況，這樣在實驗過程中，可以更有效率的進行方向的控制，如表 2。

表 2 方向控制

	輪 1	輪 2	輪 3	輪 4
前進	F	F	R	R
後退	R	R	F	F
左移	F	R	R	F
右移	R	F	F	R
右斜	0	F	0	R
左斜	F	0	R	0
左旋	R	R	R	R
右旋	F	F	F	F

在此實驗中我們主要使用前面六種方向控制與其他座標點的方向移動，都是利用直線移動方式，而左旋跟右旋在此文章並沒有利用。

實驗中以輸入 50 為基礎輸入值，其輸入範圍為 0~255，輸入值經過電路會轉換成實際的電壓值，為了避免慣性與滑動的造成太大影響，實驗選擇以 50 為輸入，利用上表的方向控制，建立輪子方向與輸入值的矩陣模型。

$$F = \text{直線前進} = \begin{vmatrix} 1 & -1 \\ 1 & -1 \end{vmatrix} \begin{vmatrix} 50 & 50 \\ 50 & 50 \end{vmatrix}$$

$$B = \text{後退} = \begin{vmatrix} -1 & 1 \\ -1 & 1 \end{vmatrix} \begin{vmatrix} 50 & 50 \\ 50 & 50 \end{vmatrix}$$

$$L = \text{左移} = \begin{vmatrix} -1 & -1 \\ 1 & 1 \end{vmatrix} \begin{vmatrix} 50 & 50 \\ 50 & 50 \end{vmatrix}$$

$$R = \text{右移} = \begin{vmatrix} 1 & 1 \\ -1 & -1 \end{vmatrix} \begin{vmatrix} 50 & 50 \\ 50 & 50 \end{vmatrix}$$

$$RA = \text{右前方斜線 45} = \begin{vmatrix} 1 & 0 \\ 0 & -1 \end{vmatrix} \begin{vmatrix} 50 & 0 \\ 0 & 50 \end{vmatrix}$$

$$LA = \text{左前方斜線 45 度} = \begin{vmatrix} 0 & -1 \\ 1 & 0 \end{vmatrix} \begin{vmatrix} 0 & 50 \\ 50 & 0 \end{vmatrix}$$

$$RB = \text{右後方斜線 45 度} = \begin{vmatrix} 0 & 1 \\ -1 & 0 \end{vmatrix} \begin{vmatrix} 0 & 50 \\ 50 & 0 \end{vmatrix}$$

$$RL = \text{左後方斜線 45 度} = \begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix} \begin{vmatrix} 50 & 0 \\ 0 & 50 \end{vmatrix}$$

以上列出移動平台的主要方向，為了讓移動平台能往其他角度移動，我們將其運動利用圖形方法表示，可以從圖形中了解在其他角度時應該如何調整四顆馬達的方向與速度。

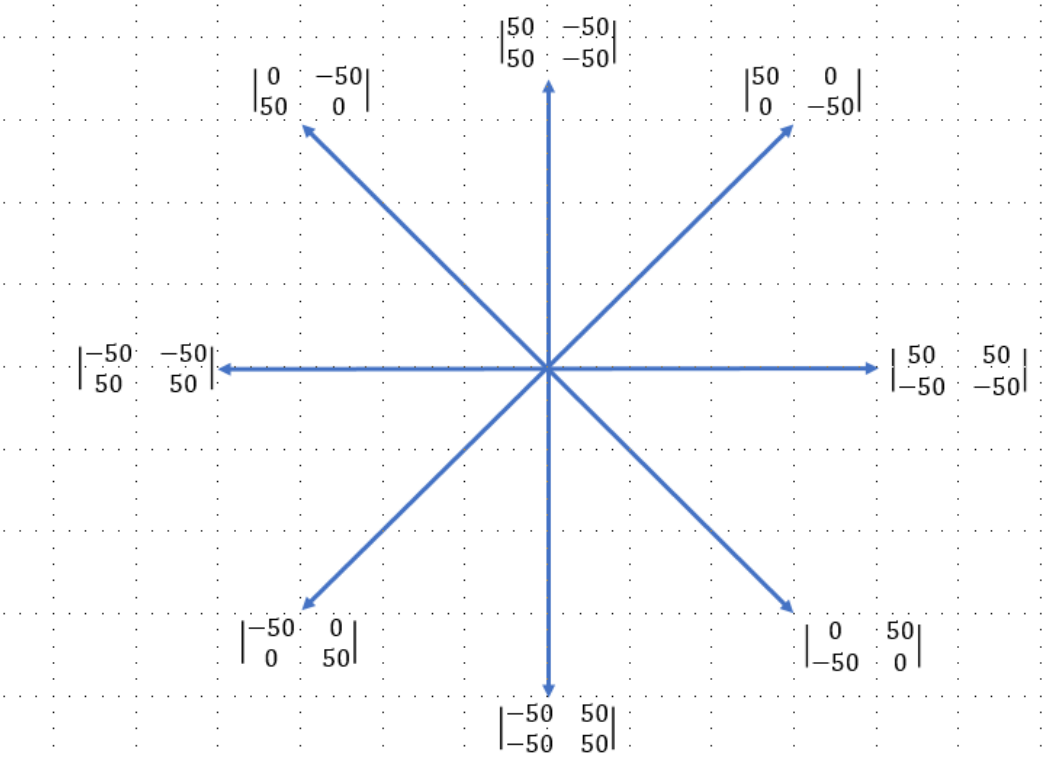


圖 13 全向輪移動時馬達狀態

由圖 13 可知在水平線與垂直線上四輪都必須轉動，而且四輪的轉速相同時，達到水平或垂直的力抵銷，而在 45° 時只需要兩顆相同方向的馬達轉動，由此可以得知若要往其他角度移動，調整 45° 時為 0 得馬達轉速，根據其方向與速度做調整就可以使移動平台往水平或垂直線移動，借由此方法就可以達到每個角度的移動。

以右前方 45° $\begin{vmatrix} 50 & 0 \\ 0 & -50 \end{vmatrix}$ 與右移 $\begin{vmatrix} 50 & 50 \\ -50 & -50 \end{vmatrix}$ 為例，其 2 輪與 4 輪相同，此時調整 1、3 輪的方向與右移相同，1 輪為反向、3 輪為正向，且持續增加 1、3 輪的轉速，則移動平台原本往右前方 45° 移動，

經過調整後會越來越偏向右移的方向，所以只要經過轉速與方向的調整就可以達到全向的控制，全向輪的流程圖如下。

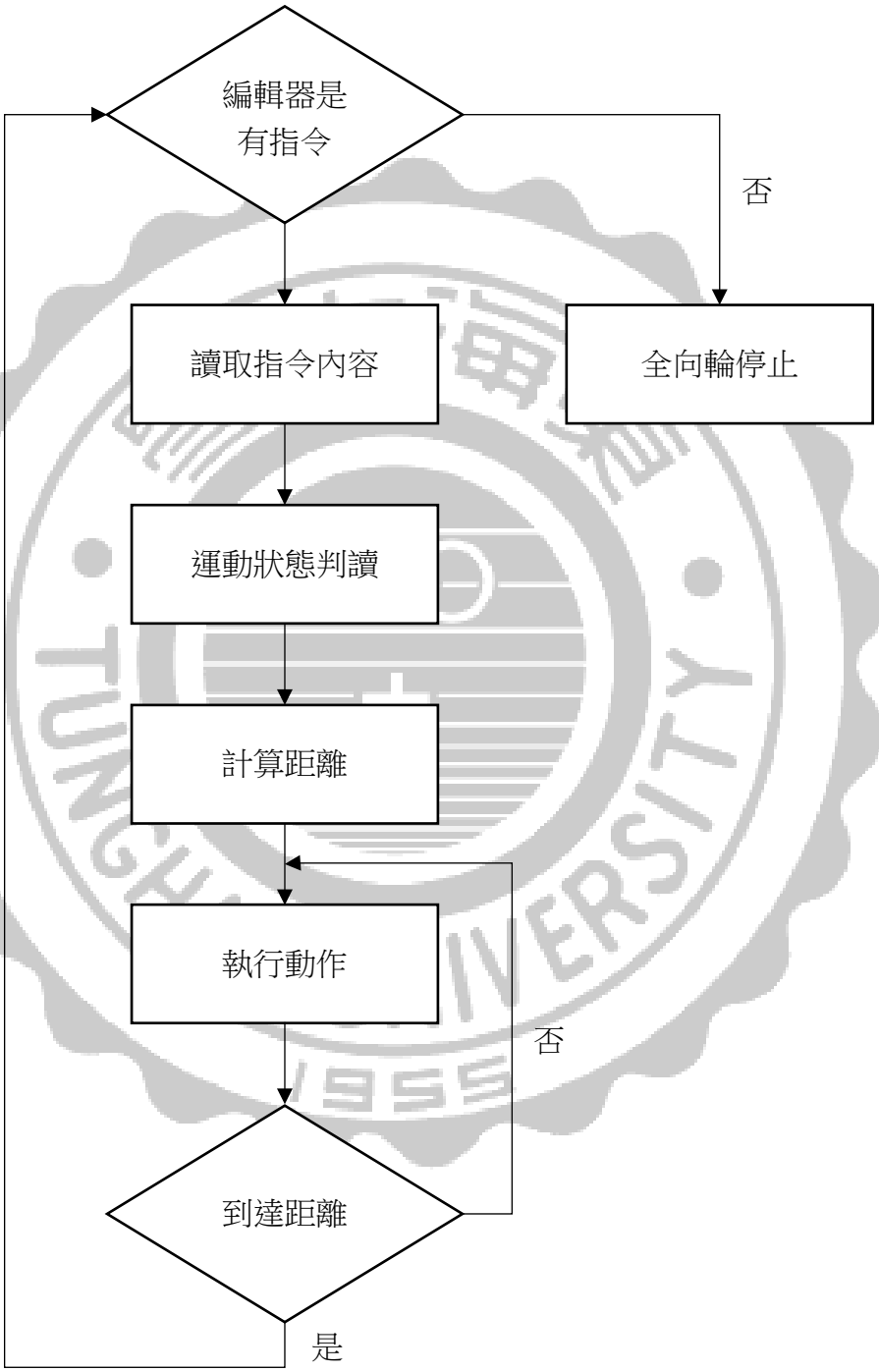


圖 14 全向輪流程圖

4.3 誤差改善

利用光感測器量測馬達所轉的圈數，經過齒輪比例轉換，感測器感測 160 格則全向輪為旋轉一圈，由於我們接收的數值為感測值之值，需要將感測器與實際的輪子行走距離作轉換，實際值如下:

輪子半徑為 3cm 其圓周為 6π

輪子一圈=感測器 160 格

感測器 1 格=輪子圓周/160=0.11781(cm)

經過此換算可以得知從感測上的值在實際行走時的距離為何，只要重複檢測感測器所轉得圈數即可得知目前輪子行走的距離，但是經過實測後偵測的距離與實際的距離會有所誤差，其原因可能機構上的不平衡，導致輪子有空轉的現象，或是馬達停止後仍繼續滑行產生誤差，所以在輸入距離與輸出距離產生問題，根據實測後對距離作修正，首先量測輸入為 20~100 公分間格 10 公分實際距離，記錄其移動值如圖

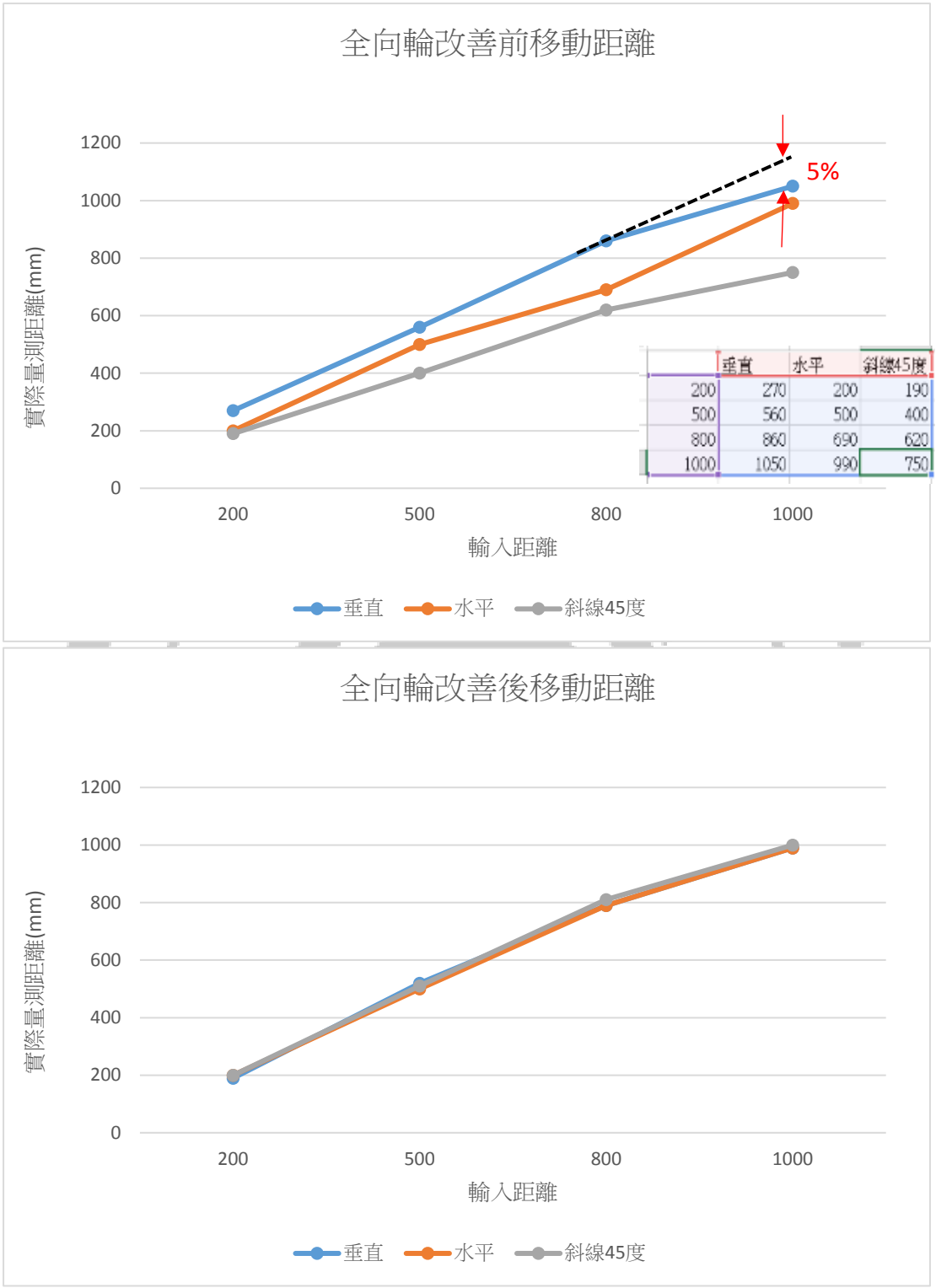


圖 15 改善前後移動實際量測

由實測圖 15 可以得知每個方向的距離誤差會有所不同，因此需要不同的參數修正，但可以發現輸入與實際值成正比，只要加入誤差

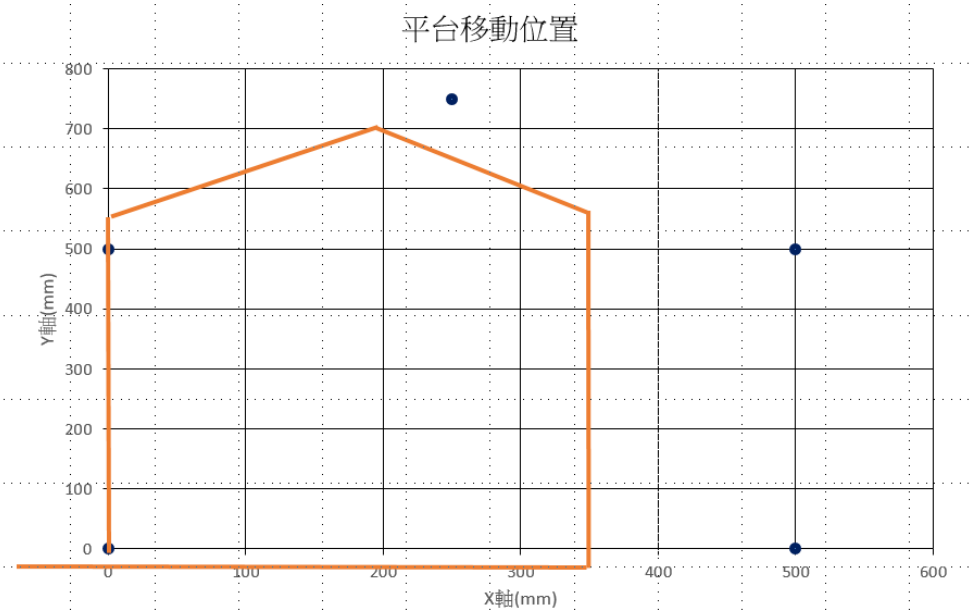
值就可以調整距離的準確度，經過修正後距離的誤差變為±1 公分，距離修正後就可以利用座標點的位置給予平台到指定的位置，並且距離指定位置有更好的精準度。

實驗過程中仍會發現一些問題，即使每個馬達給予相同的輸入值，但實際測量後每顆馬達的轉速會不相同，並且在移動過程中不穩定，經過誤差累積後整個全向輪移動平台會產生偏移的問題，關於這個問題在下面得章節在做討論。

4.4 量測結果

將實際量測平台移動的狀況，主要觀察平台是否照輸入的座標點作移動、移動的距離的誤差和平台的偏移問題，我們輸入座標(0,0)、(0,500)、(250,750)、(500,500)、(500,0)、(0,0)，並觀察平台移動，結果如圖 16、17。

可以看出在未修正的情況下，全向輪移動平台能夠正確判斷座標點的位置並且依照其座標點移動，但是可以發現其移動的距離會有較大的誤差，其中可能是慣性或是打滑等其他因素，因此經過上述的距離修正的方法，全向輪平台以較少的誤差移動到指定的座標位置，並且按照我們所輸入的路徑最後回到原點，並離原點誤差五公分之內，大大減少原本尚未修正的移動方法，由此可知實驗成功建立全向輪平台移動座標系，並且輸入座標點給予全向輪平台移動的方法。



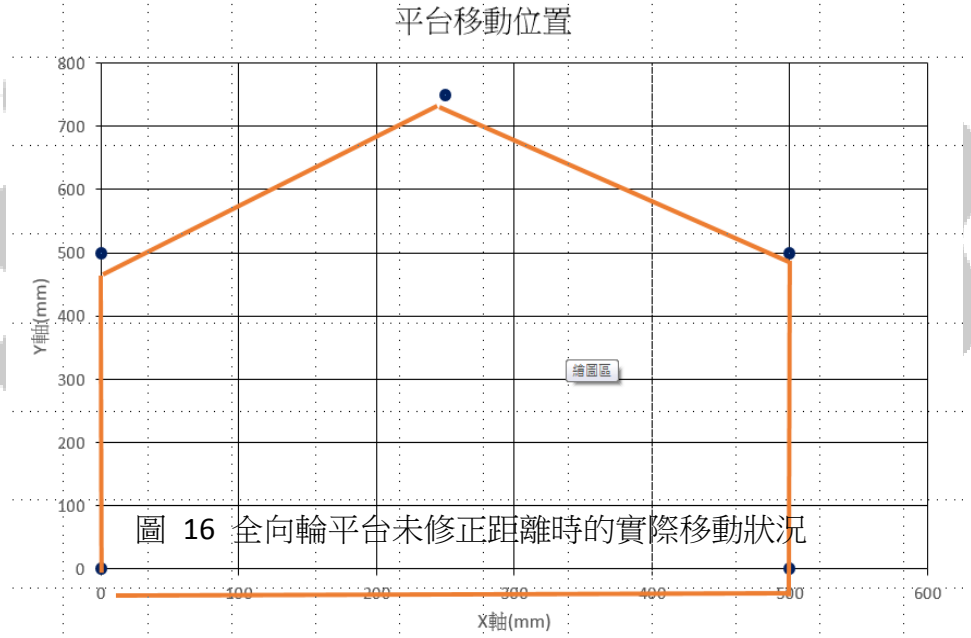


圖 16 全向輪平台未修正距離時的實際移動狀況

圖 17 全向輪平台修正距離後的實際移動狀況

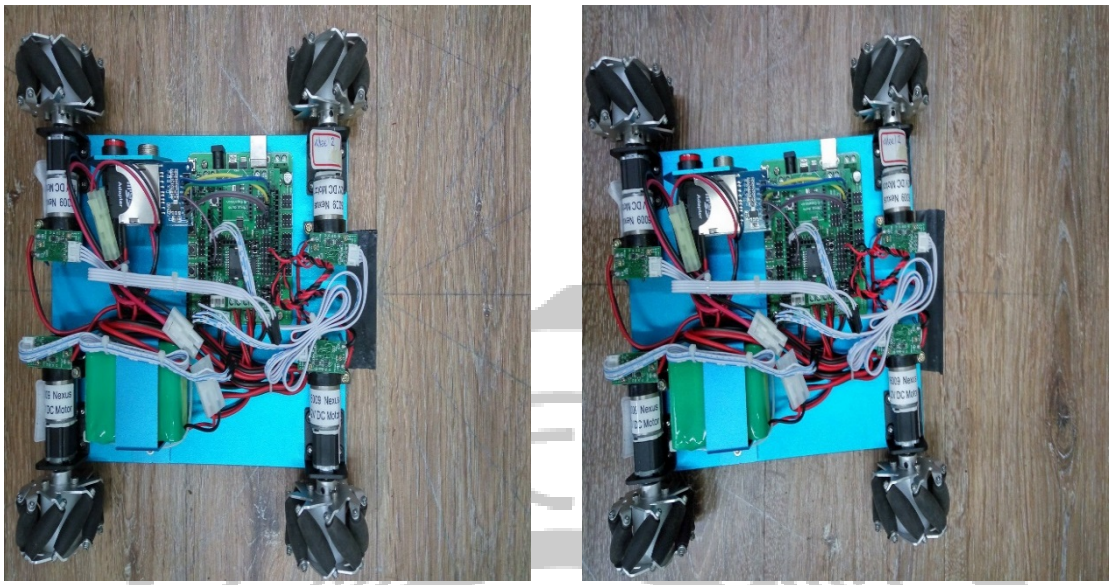


圖 18 全向輪偏斜問題

從實驗中會發現一些問題，全向輪移動平台在移動的過程中由於四顆馬達會有誤差，即使給予相同的電壓還是會有誤差並不會達到完全相同的轉速，所以會造成馬達有偏斜的問題，這樣會使的移動平台移動的角度會有所誤差，經過時間的累加容易累積誤差，使移動平台偏斜問題更加嚴重，平台的偏斜問題如圖 18。

第五章　結論與未來展望

5.1 結論

提供更人性化的介面利用指令達到控制全向輪的移動，我們可以使用 Autocad 產生出的檔案或是其他的座標輸入軟體，對此全向輪平台產生一個路經規劃的檔案，不必在額外加入其他的室內定位或是一些循跡軌道，依據輸入的座標就可以控制全向輪平台，擁有更好的適應性，不用照著既定的路線移動，提供了一個路經規劃與全向輪平台直接連接的方法。

此論文提供全向輪中麥克納姆輪數學推導，介紹我們實驗中所使用的系統，並且建立出系統座標，以平台初始位置為原點，可以使平台根據輸入的座標做判斷，根據座標對馬達產生不同電壓，可以使平台做任何角度的直線移動，並且加入距離修正方法，使平台在移動到座標時誤差更小。

原本每個方向的誤差為 1 公分，但是平台在作不同移動後，誤差累積後會越來越大，還有平台在移動過程中會有偏移的狀況，這些問題將在下章節作討論，總結實驗還是成功建立座標系統使平台移動，並且經過距離修正，使誤差達到 5%之內。

以下是本實驗所解決的問題:

- 提供人性的介面並透過指令控制全向輪。
- 修正全向輪的移動距離使誤差達到 5%之內。
- 能夠透過座標或路徑規劃的移動方法控制全向輪。

5.2 未來展望

在這個指令集的部分可以提供更多精簡指令控制全向輪的移動，像是旋轉或是行走圓弧路徑等等，達到更完善的控制方法，還能將整個編輯編譯燒錄功能做為一個完整的軟體。

硬體控制部分可以在加入其它感測器部分，來修正平台移動移動時造成的偏斜，以及考慮環境的影響，可以加入閉迴路的控制方法，減少誤差的累積。在啟動部分也可使用能夠四輪同時驅動的裝置，可以避免四顆馬達啟動時會有時間差導致偏斜。而目前的全向輪平台，僅能使用在平坦的環境下。未來能加入懸吊系統或是立於崎嶇的地型，全向輪就能有更好的發展空間，必受限使用於平坦地型，可以利用全向輪有更好的移動效率能替代目前的工具，使用在物流或是移動的裝置上。利用精確的距離移動、事先規劃好路徑，讓平台準確的達到我們所指定的位置。

參考文獻

[1] 黃久紘， 三輪式全方位移動機器人之運動模型與控制系統， 逢甲大學自動控制工程學系碩士班碩士論文，104

[2] 王以祥，室內無線定位系統於全向輪自走車之研究，國立臺北教育大學理學院資訊科學系，2010

[3] 鄭婷涓，以二維影像定位系統進行全向輪自走車之運動控制，逢甲大學自動控制工程學系，103

[4] 張志豪，全向輪型機器人之控制設計之DSP實踐，中國文化大學工學院數位機電科技研究所，99

[5] 張邑瑋，具無線監控能力之全向式輪型機器人設計與實作，龍華科技大學電子工程系，103

[6] Taha Bin Mohamed, Norsehah Abd Karim,Dr Norazlin Ibrahim, Dr Raja Fazliza Raja Suleiman,Murniwati Anwar, Muhammad Fayyad Bin Aminul Rashid , Mohamad Idzrul bin idris, ”Development of Mobile Robot Drive System using Mecanum Wheels”. Industrial Automation and Robotics Department,UNIKL Malaysia France Institute Bangi, Selangor. 105

[7] Veer Alakshendra, Shital S. Chiddarwar, A Robust Adaptive Control of Mecanum Wheel Mobile Robot:Simulation and Experimental Validation,2016

[8] Jungje Park, Sudae Kim, Jungmin Kim and Sungshin Kim, Driving Control of Mobile Robot with Mecanum Wheel using Fuzzy Inference System, Department of Electrical Engineering, Pusan National University, Busan, Korea,2010

[9] Qian Jia, Mulan Wang, Shuqing Liu, Jianjing Ge, Chen Gu, Research and Development of Mecanum-wheeled Omnidirectional Mobile Robot

Implemented by Multiple Control Methods, Industrial Center Nanjing Institute Technology,2016
[10] Hamid Taheri, Bing Qiao, Nurallah Ghaeminezhad,” Kinematic Model of a Four Mecanum Wheeled Mobile Robot”, College of Electronic and Information, College of Astronautics NUAA, College of Automation, 2015

[11] 麥克納姆輪淺談，http://www.gmii.tw/makeblock/4，2015

[12] 麥克納姆輪公司， http://www.nexusrobot.com/

附錄

```
#include <PinChangeInt.h>                                   /*加入所使用到的標頭檔*/
#include <PinChangeIntConfig.h>
#include <PID_Beta6.h>
#include <MotorWheel.h>
#include <SPI.h>
#include <SD.h>                                             /**********************/
#ifndef MICROS_PER_SEC
#define MICROS_PER_SEC 1000000
#endif
int olddistance=0,totaldistance=0,distance = 0;             /*        設定參數        */
int error=100;
unsigned long currenttime=0,pretime=0;
long X=0,Y=0,X1=0,X0=0,Y1=0,Y0=0;
float cp;
float cdis;
long a,b,c,d,e;
int power1=50,power2=50,power3=50,power4=50;
int xrad,yrad;
bool w1d,w2d,w3d,w4d;
float w1p,w2p,w3p,w4p;
const int chipSelect = 18;
char val;
int sum,i,j;
int data[50];
irqISR(irq1,isr1);
MotorWheel wheel1(3,2,16,17,&irq1);
irqISR(irq2,isr2);
MotorWheel wheel2(3,4,14,15,&irq2);
irqISR(irq3,isr3);
MotorWheel wheel3(9,8,16,17,&irq3);
irqISR(irq4,isr4);
MotorWheel wheel4(10,7,16,17,&irq4);                        /**************************/
void setup() {
TCCR1B=TCCR1B&0xf8|0x01; // Pin9,Pin10 PWM 31250Hz, Silent PWM       /*頻率調整與傳輸設定*/
TCCR2B=TCCR2B&0xf8|0x01; // Pin3,Pin11 PWM 31250Hz
wheel1.PIDEnable(KC,TAUI,TAUD,10);
```

```
wheel2.PIDEnable(KC,TAUI,TAUD,10);
wheel3.PIDEnable(KC,TAUI,TAUD,10);
wheel4.PIDEnable(KC,TAUI,TAUD,10);
 Serial.begin(9600);                                          /********************/
    while (!Serial) {}                                        /*讀取 txt 檔中資料*/
      if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    return;
  }
   Serial.println("card initialized.");
    File dataFile = SD.open("cad.txt");
    if(dataFile)
    {
      while(dataFile.available())
      {
       data[i]=dataFile.read();
       i=i+1;
      }
      dataFile.close();
    }                                                         /********************/
}
void loop()
{                                                             /*判斷 txt 中資料設定狀態*/
  for(j=0;j<i;j++)
  {
  if(data[j]=='F')
  {
    sum = (data[j+2]-48)*1000+(data[j+3]-48)*100+(data[j+4]-48)*10+(data[j+5]-48)*1;
    val='F';
  }
  if(data[j]=='B')
   {
    sum = (data[j+2]-48)*1000+(data[j+3]-48)*100+(data[j+4]-48)*10+(data[j+5]-48)*1;
    val='B';
   }
    if(data[j]=='R')
   {
```

```
    if(data[j+1]=='F')
    {
      sum = (data[j+3]-48)*1000+(data[j+4]-48)*100+(data[j+5]-48)*10+(data[j+6]-48)*1;
        val='W' ;
    }
    if(data[j+1]=='B')
    {
      sum = (data[j+3]-48)*1000+(data[j+4]-48)*100+(data[j+5]-48)*10+(data[j+6]-48)*1;
       val='Z';
    }
  sum = (data[j+2]-48)*1000+(data[j+3]-48)*100+(data[j+4]-48)*10+(data[j+5]-48)*1;
  val='R';
 }
   if(data[j]=='L')
 {

    if(data[j+1]=='F')
    {
       val='X'  ;
       sum = (data[j+3]-48)*1000+(data[j+4]-48)*100+(data[j+5]-48)*10+(data[j+6]-48)*1;
    }
    if(data[j+1]=='B')
    {
      sum = (data[j+3]-48)*1000+(data[j+4]-48)*100+(data[j+5]-48)*10+(data[j+6]-48)*1;
       val='Y';
    }
  sum = (data[j+2]-48)*1000+(data[j+3]-48)*100+(data[j+4]-48)*10+(data[j+5]-48)*1;
  val='L';
 }                                                    /******************/
switch (val)                                          /*根據狀態做出全向輪動作*/
{
  case  'F':
      Serial.print("F=");
      Serial.println(sum);
      A(sum);
      break;
  case  'B':
      Serial.print("B=");
```

```
        Serial.println(sum);
        B(sum);
        break;
    case  'R':
        Serial.print("R=");
        Serial.println(sum);
        R(sum);
        break;
    case  'L':
        Serial.print("L=");
        Serial.println(sum);
        L(sum);
        break;
    case  'W':
        Serial.print("W=");
        Serial.println(sum);
        RA(sum);
        break;
    case  'X':
        Serial.print("X=");
        Serial.println(sum);
        LA(sum);
        break;
     case  'Y':
        Serial.print("Y=");
        Serial.println(sum);
        LB(sum);
        break;
    case  'Z':
        Serial.print("Z=");
        Serial.println(sum);
        RB(sum);
        break;
    default :
        break;
}                                                    /**********************/
val='0';
sum=0;
```

```
  }
  S();
  while(1){}
}

int randpoint(float gogo)                                    /*隨機座標點函式*/
{
  while(1)
  {
    currenttime = millis();
    if((unsigned long)(currenttime - pretime) >= 50)
    {
       pretime = currenttime;
       distance = abs(wheel2.getCurrPulse()*1.1781);
       if((distance)*1.0 >= gogo )
       {
         wheel2.setCurrPulse(0);
         wheel1.setCurrPulse(0);
         break;
       }
    }
  }
}                                                            /*****************/

int A(float gogo)                                            /*全向輪前進函式*/
{
      wheel1.runPWM(50,1);
      wheel2.runPWM(50,1);
      wheel3.runPWM(50,0);
      wheel4.runPWM(50,0);
  while(1)
  {
    currenttime = millis();
    if((unsigned long)(currenttime - pretime) >= 50)
    {
      pretime = currenttime;
      distance = abs(wheel2.getCurrPulse()*1.1781);
      if((distance+60 )*1.0 >= gogo )
```

```
      {
          wheel2.setCurrPulse(0);
          wheel1.setCurrPulse(0);
          break;
      }
    }
  }
}                                                    /******************/

int B(float gogo)                                    /*全向輪後退函式*/
{
      wheel1.runPWM(50,0);
      wheel2.runPWM(50,0);
      wheel3.runPWM(50,1);
      wheel4.runPWM(50,1);
  while(1)
  {
    currenttime = millis();
    if((unsigned long)(currenttime - pretime) >= 50)
    {
      pretime = currenttime;
      distance = abs(wheel2.getCurrPulse()*1.1781);
      if((distance+60 )*1.0 >= gogo )
      {
          wheel2.setCurrPulse(0);
          wheel1.setCurrPulse(0);
          break;
      }
    }
  }
}                                                    /*******************/
int R(float gogo)                                    /*全向輪右移函式*/
{
      wheel1.runPWM(50,0);
      wheel2.runPWM(50,1);
      wheel3.runPWM(50,1);
      wheel4.runPWM(50,0);
  while(1)
```

```
  {
    currenttime = millis();
    if((unsigned long)(currenttime - pretime) >= 50)
    {
      pretime = currenttime;
      distance = abs(wheel2.getCurrPulse()*1.1781);
      if(distance  >= gogo )
      {
          wheel2.setCurrPulse(0);
          wheel1.setCurrPulse(0);
          break;
      }
    }
  }
}                                                    /*******************/
int L(float gogo)                                    /*全向輪左移函式*/
{
      wheel1.runPWM(50,1);
      wheel2.runPWM(50,0);
      wheel3.runPWM(50,0);
      wheel4.runPWM(50,1);
  while(1)
  {
    currenttime = millis();
    if((unsigned long)(currenttime - pretime) >= 50)
    {
      pretime = currenttime;
      distance = abs(wheel2.getCurrPulse()*1.1781);
      if(distance  >= gogo )
      {
          wheel2.setCurrPulse(0);
          wheel1.setCurrPulse(0);
          break;
      }
    }
  }
}                                                    /*******************/
```

```
int LA(float gogo)                                          /*全向輪左前函式*/
{
      wheel1.runPWM(50,1);
      wheel2.runPWM(0,1);
      wheel3.runPWM(50,0);
      wheel4.runPWM(0,0);
  while(1)
  {
    currenttime = millis();
    if((unsigned long)(currenttime - pretime) >= 50)
    {
      pretime = currenttime;
      distance = abs(wheel1.getCurrPulse()*1.1781);
      if( (distance*0.7) >= gogo )
      {
          wheel1.setCurrPulse(0);
          wheel2.setCurrPulse(0);
          break;
      }
    }
  }
}                                                           /******************/


int RA(float gogo)                                          /*全向輪右前函式*/
{
      wheel1.runPWM(0,1);
      wheel2.runPWM(50,1);
      wheel3.runPWM(0,0);
      wheel4.runPWM(50,0);
  while(1)
  {
    currenttime = millis();
    if((unsigned long)(currenttime - pretime) >= 50)
    {
      pretime = currenttime;
```

```
      distance = abs(wheel2.getCurrPulse()*1.1781);
      if( (distance*0.7) >= gogo )
      {
          wheel1.setCurrPulse(0);
          wheel2.setCurrPulse(0);
          break;
      }
    }
  }
}                                                    /***************/

int LB(float gogo)                                   /*全向輪左後函式*/
{
      wheel1.runPWM(0,1);
      wheel2.runPWM(50,0);
      wheel3.runPWM(0,0);
      wheel4.runPWM(50,1);
  while(1)
  {
    currenttime = millis();
    if((unsigned long)(currenttime - pretime) >= 50)
    {
      pretime = currenttime;
      distance = abs(wheel2.getCurrPulse()*1.1781);
      if( (distance*0.7) >= gogo )
      {
          wheel1.setCurrPulse(0);
          wheel2.setCurrPulse(0);
          break;
      }
    }
  }
}                                                    /*****************/

int RB(float gogo)                                   /*全向輪右後函式*/
{
      wheel1.runPWM(50,0);
      wheel2.runPWM(0,1);
```

```
      wheel3.runPWM(50,1);
      wheel4.runPWM(0,0);
  while(1)
  {
    currenttime = millis();
    if((unsigned long)(currenttime - pretime) >= 50)
    {
      pretime = currenttime;
      distance = abs(wheel1.getCurrPulse()*1.1781);
      if( (distance*0.7) >= gogo )
      {
          wheel1.setCurrPulse(0);
          wheel2.setCurrPulse(0);
          break;
      }
    }
  }
}                                                    /*****************/
int rad(float gogo)                                  /*全向輪隨機點函式*/
{
  while(1)
  {
    currenttime = millis();
    if((unsigned long)(currenttime - pretime) >= 50)
    {
      pretime = currenttime;
      power1=int(power2*xrad/yrad);
      power3=int(power2*xrad/yrad);

      aregual();
      rregual();
      wheel1.runPWM(power1,1);
      wheel2.runPWM(power2,1);
      wheel3.runPWM(power3,0);
      wheel4.runPWM(power4,0);

      distance = abs(wheel2.getCurrPulse()*1.1781);
      if( (50+distance*0.7) >= gogo )
```

```
      {
          wheel1.setCurrPulse(0);
          wheel2.setCurrPulse(0);
          break;
      }
    }
  }
}                                                          /**************/
void S(void)                                               /*全向輪停止函式*/
{
    while(1)
          {
            wheel2.backoffPWM(0);
            wheel1.advancePWM(0);
            wheel3.backoffPWM(0);
            wheel4.advancePWM(0);
          }
  }                                                        /******************/

void regual()                                              /*全向輪相位判斷函式*/
{
    a=abs(wheel1.getSpeedRPM());
    b=abs(wheel2.getSpeedRPM());
    c=abs(wheel3.getSpeedRPM());
    d=abs(wheel4.getSpeedRPM());
    e=(a+b+c+d)/4;
    if((abs(e-a))>10)
    {
      if((e-a)<0) power1 = power1-1;
       else if((e-a)>0) power1 = power1+1;
    }
    if((abs(e-b))>10)
    {
      if((e-b)<0) power2 = power2-1;
       else if((e-b)>0) power2 = power2+1;
    }
    if((abs(e-c))>10)
```

```
    {
      if ((e-c)<0) power3 = power3-1;
     else  if((e-c)>0) power3 = power3+1;
    }
    if((abs(e-d))>10)
    {
      if ((e-d)<0) power4 = power4-1;
      else  if((e-d)>0) power4 = power4+1;
    }
}                                                    /*******************/

void aregual()                                       /*全向輪1、3輪轉動函式*/
{
    a=abs(wheel1.getSpeedRPM());
    c=abs(wheel3.getSpeedRPM());
    e=(a+c)/2;
    if((abs(e-a))>10)
    {
      if((e-a)<0) power1 = power1-1;
       else if((e-a)>0) power1 = power1+1;
    }
    if((abs(e-c))>10)
    {
      if ((e-c)<0) power3 = power3-1;
     else  if((e-c)>0) power3 = power3+1;
    }
}                                                    /************************/
void rregual()                                       /*全向輪2、4輪轉動函式*/
{
    b=abs(wheel2.getSpeedRPM());
    d=abs(wheel4.getSpeedRPM());
    e=(b+d)/2;
    if((abs(e-b))>10)
    {
      if((e-b)<0) power2 = power2-1;
       else if((e-b)>0) power2 = power2+1;
    }
    if((abs(e-d))>10)
```

```
    {
      if ((e-d)<0) power4 = power4-1;
      else  if((e-d)>0) power4 = power4+1;
    }
}                                                    /************************/
```