DISEÑO DE UN SISTEMA DE CONTROL DE SEGUIMIENTO DE TRAYECTORIAS PARA UN ROBOT MÓVIL

TESIS

Que para obtener el grado de

MAESTRÍA EN CIENCIAS EN INGENIERÍA ELÉCTRICA

presenta

Carlos A. Estrada Suárez

M.C. Juan José Rincón Pasaye

Director de Tesis

D.r. Leonardo Romero Muñoz

Co-Director de Tesis

Universidad Michoacana de San Nicolás de Hidalgo

Marzo 2006

A mis Padres y a mi novia Yanin

Agradecimientos

A Dios, por permitirme existir.

A mis padres y hermanos, por su constante e infinito cariño.

A Yanin, por su amor y comprensión.

A mis maestros, por transmitirme su conocimiento.

A mis asesor y Co asesor, J Juan Rincón Pasaye y Leonardo Romero Muñoz, por brindarme su tiempo y conocimiento para el desarrollo de este proyecto.

A la Universidad Michoacana de San Nicolás de Hidalgo y al Consejo Nacional de Ciencia y Tecnología, por el apoyo otorgado para realizar estos estudios.

Resumen

DISEÑO DE UN SISTEMA DE CONTROL DE SEGUIMIENTO DE TRAYECTORIAS PARA UN ROBOT MÓVIL

En este trabajo se presenta el desarrollo de una estrategia para el control del seguimiento de trayectorias para un robot móvil con estructura de locomoción diferencial para ambientes interiores. El seguimiento de la trayectoria se plantea desde la obtención de una curva suave basada en las curvas tipo spline interpoladoras de $Catmull - Rom$ a partir de puntos de control especificados por una etapa (no abordada) de la planeación de la trayectoria, posteriormente es discretizada y particionada en arcos circulares, dando como resultado una tabla de comandos de velocidad y tiempo para las ruedas de tracción izquierda y derecha. Finalmente esta información es implementada en el microcontrolador del robot móvil.

Se describe también el sistema de instrumentación electrónico, los sensores y actuadores, así como el software desarrollado para las distintas etapas del trabajo, detallando el sistema de medición, la obtención de las tablas de velocidad y tiempo, así como de la implementación y sintonización del controlador de velocidad proporcional integral (PI), la discretización de la curva tipo spline y la implementación del arranque y paro mediante rampas de velocidad que evitan oscilaciones del robot y mejoran el desempeño del controlador de velocidad.

Se presentan también los resultados de varias pruebas realizadas, las cuales van desde trayectorias muy simples y cortas, hasta trayectorias largas y complejas, sobre diferentes tipos de terreno.

Abstract

DESIGN OF A PATH FOLLOWING CONTROL SYSTEM FOR A MOBILE ROBOT

This thesis presents a path following control strategy for a differential drive mobile robot for indoor environments. The path following is obtained from a smooth curve based on the $Catmull - Rom$ interpolator Spline which uses control points that are suppose to be given in the path planning stage (not treated) as long as its discretization, circular segment partition, speed and time commands for left and right traction wheels and finally its implementation on the mobile robot microcontroller.

The thesis describes the electronic instrumentation system including actuators and sensors as well as the developed software for the different stages of the work, given a special attention to the measurement system, speed and time arrays, the antiwindup P.I. speed controller, the Spline discretization and the start-stop ramp implementation. Finaly are presented the results of several tests obtained in differents environments.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Símbolos

V_R, V_L y v_R	Velocidades lineales de las llantas de tracción y del centro de masa del robot (m/S o Vueltas/S).
r	Radio nominal de cada llanta (metros).
r_c	Radio de un cuerpo arbitrario en movimiento circular.
r/s	Unidad de medida de la velocidad angular (radian sobre segundo).
$Pulsos/ms$	Unidad de medida de la velocidad angular mediante encoder (Pulsos por mili segundo).
R	Radio instantáneo de curvatura (metros).
Icc	Centro de la trayectoria relativo al centro de masa del robot.
(x, y)	Posición instantanea del centro de masa con respecto a marco base.
θ	Orientación instantanea del robot referente al marco base (rad o grados).
ω_R	Velocidad angular instantanea de la estructura del robot (r/S).
L	Distancia entre llantas del robot (metros).
V_a	Voltaje de armadura (Volt).
i_a	Corriente de armadura (Ampere).
e_m	Voltaje debido a la fuerza contra-electromotriz (Volt).
ω_m, ω_L	Velocidades angulares de la flecha y de la carga del motor (rad/Seg).
K_E	Constante de la fuerza contraelectromotriz (V/rad/S).
K_T	Constante de torque (N-m/A).
K_g	Constante de reducción de la caja de engranes.
r1 y r2	Radio nominal de los engranajes (metros).
R_a	Resistencia de arnadura (Ohm).
L_a	Inductancia del circuito de armadura (Henrio).
J, J_m y J_L	Inercia total, del motor y de la carga (Kg-m^2).
D	Coeficiente de fricción viscosa (N-m-Seg).
T_m y T_L	Par ejercido por el motor y ejercido en la carga (N-m).
H	Altura del robot móvil (metros).
F	Fuerza actuante al centro de masa (Newton).
τ	Torque actuante al centro de masa (N-m).
F_r y F_l	Fuerzas ejercidas al robot por las llantas derecha e izquierda (Newton).
b	Distancia del centro de masa a las llantas (metros).
M	Masa del robot (Kg).
M_o y M_{sup}	Momento ejercido a la estructura del robot y

	momento ejercido por la superficie (N-m).
ω_{sp}	Velocidad de referencia o requerida para el control de los motores (r/S o Pulsos/mS).
e y $e(t_k)$	Señal de error continua y discreta (Pulsos/mS).
$u(t)$ y $u(t_k)$	Acción de control en tiempo continuo y discreto (Volt).
K	Constante proporcioal.
T_i	Constante denominada de tiempo integral.
K_I	Constante integral.
h	Periodo de muestreo o valor de la interrupción de tiempo real.
$P(t_k)$	Término proporcional.
$I(t_{k+1})$	Término integral.
$\Re^n$	Espacio de dimensión n.
G^n	Continuidad visual o geométrica.
C^n	Continuidad de la derivada n.
$B_k(x)$	Polinomios de Bernstein.
β	Parámetro de forma del Spline de Catmull-Rom.
Δu	Incremento del intervalo de definición de Spline de Catmull-Rom.
S	Sección de arco.
ND	Número de datos o longitud del vector de velocidades y tiempos.
PC	Número de puntos de control.
M	Múltiplo de la interrupción de tiempo real.
Subref	Subreferecia de velocidad (Pulsos/mS).
refn	Primer y último comando de velocidad (Pulsos/mS).
np	Número de pasos de la rampa de aceleración y desaceleración.
Submrti	Submúltiplo de la interrupción de tiempo real.
mrti	Primer y último dato del múltiplo de la interrupción.
nPT	Número de muestras para cada segmento de arco.
N_{md}	Número de muestras disponibles.
m_t	Periodo de muestreo.
k	Apuntador del vector de velocidades.

Capítulo 1

INTRODUCCIÓN

1.1. El Robot Móvil

Fue a principios de los años sesenta cuando se introdujeron en la industria dispositivos autónomos capaces de realizar tareas en sustitución de la mano del hombre. Estos dispositivos, también llamados robots, predecesores de las nuevas tecnologías, proliferaron de manera insospechada, fue entonces cuando se prestó especial interés por parte de los investigadores e ingenieros en desarrollar dispositivos robóticos con el mayor número de cualidades posibles; rápidos, precisos y fáciles de reparar, haciendo consecuentemente crecer la automatización industrial y a su vez la productividad de la misma.

Las tareas comúnmente desarrolladas por los robots, consistían en acciones que se repetían una y otra vez, como la alimentación de combustible y componentes necesarios para las tareas de otras máquinas. Algunas de las desventajas de estos esquemas, implicaban la dependencia de un entorno especial de trabajo, caracterizado principalmente por las limitantes de los mismos robots manipuladores y máquinas. Una alternativa para la solución a este problema consistió en la construcción de vehículos móviles. Uno de los primeros esquemas que se presentaron en la industria, fue un modelo guiado por rieles el cual ofrecía mayor flexibilidad a la hora de proveer las materias primas. De esta manera aparecieron los primeros vehículos móviles o *vehículos guiados automáticamente* (AGV's) [Muñoz95].

El entorno industrial, caracterizado por tener una estructura, permitió que este tipo de vehículos se desenvolvieran con cierta facilidad, dotados con una inteligencia mínima,

pero capaz de procesar la información de su entorno. Sin embargo, cualquier perturbación en el entorno de trabajo, propiciaba que el vehículo no lograra sus objetivos, con lo que de nueva cuenta se encontraron limitantes para este tipo de esquemas. Fué entonces cuando se pensó en un tipo de vehículo que fuera capaz de interactuar en entornos más generales e inciertos. Este tipo de entornos, exigía que los vehículos móviles estuvieran dotados de una inteligencia más compleja, capaces de procesar rápidamente la información, y así tomar una decisión sobre la acción a seguir.

La cantidad de aplicaciones para estos vehículos es variada, desde las tareas más complicadas y peligrosas para el ser humano, hasta las más sencillas. Entre las primeras está la minería, reconocimiento de areas bajo radiación en la industria nuclear, incluso la limpieza industrial, el otro tipo de actividades son la vigilancia o inclusive en hospitales, mediante la asistencia a personas discapacitadas. La amplia gama de aplicaciones de estos vehículos justifica su existencia, aún más alienta a los investigadores a diseñar nuevos esquemas y algoritmos que proporcionen mayor eficiencia en los robots a la hora de interactuar con el medio.

Existen dos características importantes para un robot móvil [Muñoz95] en las que se puede citar la diferencia con cualquier otro tipo de vehículo:

- Percepción: Determina la relación del robot con su entorno de trabajo, mediante el uso de sensores.
- Razonamiento: Determina las acciones que se han de realizar en cada instante, según el estado en el que se encuentra el robot y su entorno, para alcanzar las metas asignadas.

1.2. Contexto Actual

En la actualidad existen proyectos millonarios, en los cuales se involucra el uso de robots móviles, como por ejemplo el proyecto llamado *Mars pathfinder mission* supervisado por la *NASA* (*del inglés National Aeronautics and Space Administration*) [NASA04]. Este proyecto de aproximadamente 171 millones de USD incluidos los 25 millones para la construcción del robot móvil llamado *microrover* o *sojourner* tiene como objetivos realizar una serie de experimentos que contribuyan a la realización de futuras expediciones. Una breve descripción sobre este prototipo es dada a continuación, con la finalidad de visualizar la

tecnología de punta existente.

El *microrover* tiene un peso de 11 Kg en la tierra y un peso aproximado de 4.08 Kg sobre la superficie marciana, tiene el tamaño de un pequeño vagón de tren de un juego infantil. En la figura 1.1 se muestra una foto del microrover.

Figura 1.1: Microrover usado en la misión a marte.

El *microrover* tiene una configuración de seis ruedas, y puede moverse a una velocidad de 0.6 metros por minuto, esta velocidad es suficiente para la realización de las tareas programadas, además de que no se quiere que el robot se aleje más de 10 metros del lugar de aterrizaje. El chasis con seis ruedas ofrece una mayor estabilidad comparado con el de cuatro a la hora de sobrepasar obstáculos, razón por la cual se escogió esta configuración, las ruedas son de aluminio y tienen un diámetro de 13 cm. Tres sensores de movimiento a lo largo del *microrover* pueden detectar inclinaciones excesivas y así evitar posibles volcaduras del microrover.

El sistema de telecomunicaciones UHF (del inglés Ultra High Frequency) es bidireccional e inalámbrico entre el *rover* y el *pathfinder*. El enlace de radio es usado para mandar comandos provenientes de la tierra hacia el *rover* y recibir imágenes y datos cap-

turados por el *rover*. La señal de radio del *microrover* es una señal similar a la de un walkie-talkie, por lo que no es posible comunicarse con él directamente. Todo el trabajo de comunicación es hecho a través del *pathfinder* como se muestra en la figura 1.2.

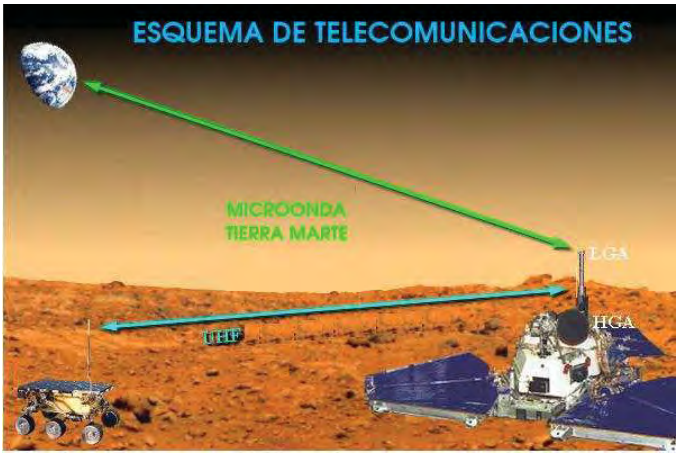


Figura 1.2: Esquema de comunicaciones.

Otro ejemplo en el desarrollo de robots móviles es el proyecto realizado en la Universidad de Luebeck, en Alemania [Luebeck04], y que lleva por nombre MAVERIC (*Mobile Autonomous Vehicle to Experiment upon Robotic Indoor Chores*). Este robot es para servicio experimental y realiza tareas en ambientes interiores. MAVERIC es un desarrollo flexible y útil, combina una variedad de potentes sensores, dos PC's y una conexión inalámbrica LAN. En la figura 1.3 se muestra la configuración del sistema descrito.

Existe una gran variedad de proyectos sobre robots móviles que podríamos mencionar, sin embargo no es el objetivo el profundizar en la variedad de robots existentes, más bien dar un panorama general del contexto actual.

1.2.1. Líneas de investigación en robótica móvil

En la actualidad existen diversas líneas de investigación dentro de la robótica móvil que a la fecha se encuentran en desarrollo [Jones99], entre éstas se encuentran la navegación, el reconocimiento, el aprendizaje, la cooperación entre sí, el razonamiento etc.

Figura 1.3: Maveric.

La navegación es uno de los retos de mayor complejidad dentro de la robótica móvil, ya que por lo general el mundo en el que nos desenvolvemos es un mundo en constante movimiento, dinámico, y de gran variabilidad geométrica.

Realizar una tarea de navegación para un robot móvil significa recorrer un camino que lo conduzca desde una posición inicial hasta otra final, pasando por ciertas posiciones intermedias o submetas. El problema de la navegación se divide en las siguientes cuatro etapas [Muñoz95].

- *Percepción del mundo*: Mediante el uso de sensores externos, creación de un mapa o modelo del entorno donde se desarrollará la tarea de navegación.
- *Planificación de la ruta*: Creación de una secuencia ordenada de objetivos o submetas que deben ser alcanzadas por el vehículo. Esta secuencia se calcula utilizando el modelo o mapa del entorno.
- *Generación del camino*: En primer lugar define una función continua que interpola la secuencia de objetivos construida por el planificador. Posteriormente procede a la discretización de la misma a fin de generar el camino.

- *Seguimiento del camino*: Efectúa el desplazamiento del vehículo, según el camino generado mediante el adecuado control de los actuadores del vehículo.

El trabajo desarrollado en esta tesis se enfoca a las dos ultimas etapas mencionadas.

1.2.2. Trabajos previos

Actualmente existe un proyecto de construcción de un robot móvil para ambientes interiores denominado "ROCA" (Romero Cardiel) desarrollado por el Dr. Leonardo Romero Muñoz [Romero01]. Dicho proyecto incluyendo hardware y software seran usados como base para el desarrollo del robot móvil usado en esta tesis. El tipo de locomoción, elementos mecánicos y eléctricos entre el robot ROCA y el robot desarrollado en este trabajo son idénticos. Con esto se pretende que los resultados de esta tesis se puedan incorporar al proyecto anteriormente citado. Aún cuando el robot móvil desarrollado para propósitos de experimentación, es una versión austera comparado con el anterior, éste es suficiente para aplicar los algoritmos desarrollados a lo largo de esta tesis. En la figura 1.4 se muestra una foto del proyecto ROCA.

Figura 1.4: Foto del robot móvil desarrollado por el D.r. Leonardo Romero M.

El diagrama de bloques de la figura 1.5 muestra la arquitectura de los principales componentes del proyecto ROCA [Arellano05].

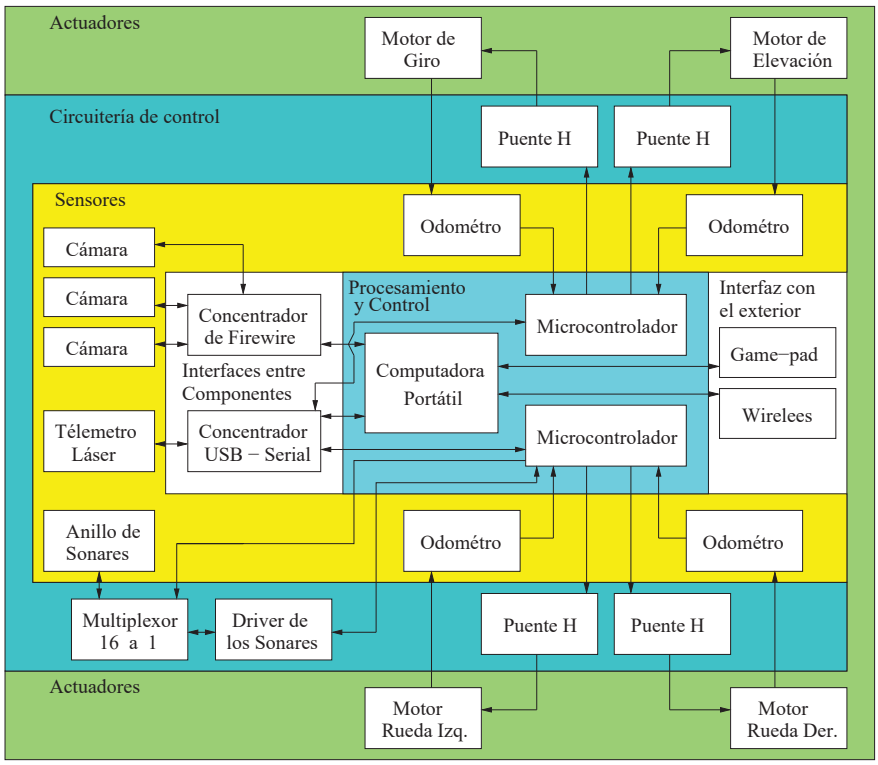


Figura 1.5: Arquitectura del proyecto ROCA.

1.3. Objetivo y Contribuciones

El objetivo general de este proyecto de tesis, es el diseño de un algoritmo para resolver el seguimiento de trayectorias de forma eficaz y aceptablemente precisa, es decir dada una trayectoria, el robot móvil debe ser lo suficientemente hábil para seguir dicha trayectoria, con un grado de error aceptable.

Este objetivo se enmarca en un proyecto más general, en el cual el seguimiento de trayectorias es una parte necesaria para lograr el movimiento autónomo del robot en un ambiente con obstáculos fijos o incluso móviles. Es importante resaltar que en esta tesis se propone una solución al problema del seguimiento de trayectorias, mas no la definición de éstas.

En esta tesis se describen los algoritmos desarrollados para lograr las metas im-

puestas, asi como la construcción del *hardware* (sensores y actuadores) y *software* necesarios para la realización de los algoritmos propuestos.

1.4. Plataforma de Desarrollo

El desarrollo del *software* y los algoritmos de control se realizaron bajo el sistema operativo Linux [Inc.04]. Asi por ejemplo, para la programación y compilación de los algoritmos para el microcontrolador [Mot02b] se usó el lenguaje "C"del proyecto 68HCS12 para Linux [Carrez03]. Por otra parte para la obtención del modelo de los motores de C.D. y simulación de los algoritmos de control, se hizo uso del software Octave [Eaton04] de licencia pública.

Las ventajas que ofrece esta plataforma son muchas, sin embargo, una de las más decisivas para su uso, fué sin duda la licencia de *software* libre y código abierto, como por ejemplo el compilador de lenguaje "C" recién citado para el microcontrolador, salvo algunas excepciones, como, el uso de Matlab o LabView en las cuales se implementaron la curva interpoladora que conforma la trayectoria y el sistema de control y adquisición de datos respectivamente para las cuales la universidad cuenta con licencia de uso.

1.5. Estructura de la Tesis

La tesis se encuentra dividida en cinco capítulos, tres apéndices y las referencias bibliográficas con lo que se pretende que futuros lectores interesados en estos temas encuentren en este trabajo una referencia cómoda y fácil de usar.

El capítulo 2, *"Arquitectura, modelado y control del robot móvil"*, habla en su mayor parte de los dispositivos que conforman al robot móvil, empezando por la arquitectura del robot, en donde se describe la disposición de las ruedas y la configuración geométrica del robot. También se habla de los modelos que gobiernan el comportamiento de todo el conjunto de dispositivos que conforman al robot móvil.

El capítulo 3, *"Generación de trayectorias basada en Splines"* aborda los conceptos teóricos sobre las *Splines*, sus orígenes y versiones existentes. Posteriormente se introduce la teoría y conceptos sobre la curva tipo *Spline* interpoladora de $Catmull - Rom$, la cual es utilizada en nuestro algoritmo de generación de trayectorias. Posteriormente se aborda el problema principal de esta tesis, que es el seguimiento y ejecución de trayectorias.

En el capítulo 4, *"Pruebas y resultados"* se reportan los resultados prácticos obtenidos de diversos experimentos realizados con el robot móvil.

En el último capítulo, *"Conclusiones"* se describen los resultados obtenidos a lo largo del desarrollo de esta tesis, así como las ideas que se pretenden incorporar a este proyecto en un futuro.

Finalmente en los *Apéndices* se reporta en su mayoria secciones de código pertenecientes a los programas que gobiernan los diferentes dispositivos del robot móvil. En el apéndice **A** se describen dos plataformas que fueron hechas para sintonizar el controlador digital del robot móvil y para el seguimiento de las trayectorias de prueba. En el apéndice **B** se reporta una sección de código, para la implementación de la curva tipo *Spline* interpoladora de $Catmull - Rom$. En el apéndice C se presentan los códigos para el microcontrolador. En el apéndice D se presenta el diagrama eléctrico del robot móvil. En el apéndice E se presentan las hojas de datos de los dispositivos que conforma el robot móvil. En el apéndice F se muestra el esquema de medición de corriente, utilizado en el cápitulo 2 sección 2.2. En el apéndice G se decribe otro experimento que sirvió para el modelado del robot móvil (ver cápitulo 2). Finalmente en el apéndice H se descibe el criterio de selección de la rotación del robot móvil durante el seguimiento de una trayectoria.

Capítulo 2

ARQUITECTURA, MODELADO Y CONTROL DEL ROBOT MÓVIL

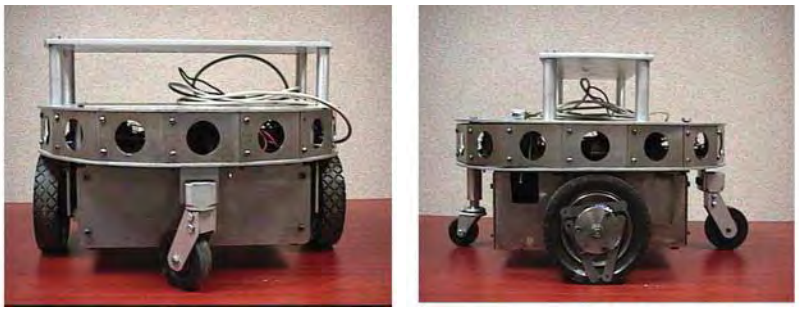
La robótica es una ciencia multidisciplinaria, ya que requiere de la integración de múltiples ciencias. La teoría de control y el modelado de sistemas son algunos de los temas implícitos dentro de las áreas involucradas por la robótica. El conocimiento de estos, es necesario para el desarrollo de una estrategia de control que será aplicada a los actuadores del robot móvil. El objetivo del presente capítulo es describir las características físicas y el comportamiento del robot móvil, haciendo uso de los modelos matemáticos que lo describen.

2.1. Arquitectura del Robot

La arquitectura del robot es la disposición de los elementos que lo integran, y por su arquitectura, nuestro robot móvil se clasifica como de tipo diferencial [Dudek00]. La locomoción diferencial mostrada en la figura 2.1 (a) es quizá una de las configuraciones más simples en robots móviles y es comúnmente usada para proyectos donde el robot móvil se desenvuelve en interiores como en nuestro caso. La configuración diferencial consta de dos ruedas de tracción montadas sobre un eje de rotación común, y en nuestro caso se agregaron dos ruedas de apoyo con giro libre montadas en la parte frontal y en la parte posterior del

robot, (ver figura 2.1(b)), formando asi un *diamante* entre las cuatro ruedas, (ver figura 2.3(a)).

Una característica importante de la configuración diferencial es la independencia de las ruedas de tracción, es decir, cada rueda puede tomar una velocidad distinta respecto a la otra, inclusive girar en sentidos opuestos, esta característica permite que el robot describa trayectorias con un grado de dificultad considerable con respecto a otras configuraciones. El siguiente ejemplo ilustra un caso donde la meta es lograr que el robot móvil logre llegar a la postura mostrada en la figura 2.2(a). En el caso de la configuración diferencial, esto es posible lograrlo siguiendo la trayectoria punteada, seguida de un giro sobre su centro. Para el caso de la configuración de tipo automóvil no es tan simple, ya que es necesario realizar una serie de movimientos más complejos y aun así es posible no lograr la postura deseada.



(a) ***Vista frontal*** **(b)** ***Vista lateral***

Figura 2.1: Locomoción diferencial.

Por otro lado, un inconveniente de la configuración diferencial, con arreglo de ruedas en forma de *diamante*, es el uso de las ruedas de giro libre sin ningún tipo de amortiguamiento, lo anterior puede provocar en un momento dado que las ruedas de tracción pierdan el contacto con el terreno (ver figura 2.3(b)).

Este inconveniente se logra superar agregando algun tipo de mecanismo que permita que las ruedas de apoyo tengan movimiento sobre su eje vertical.

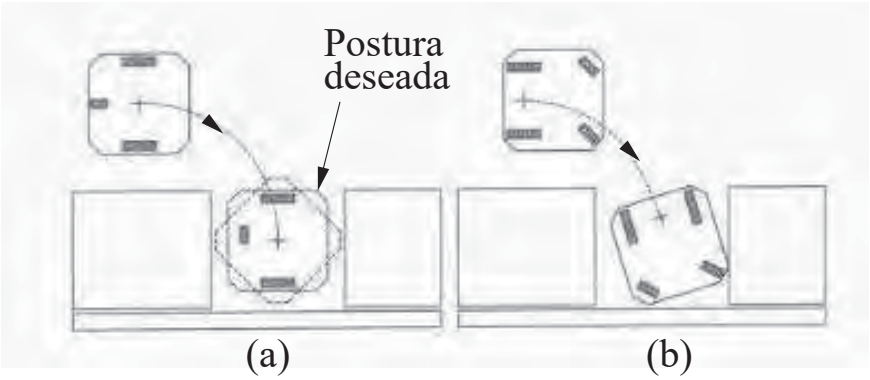
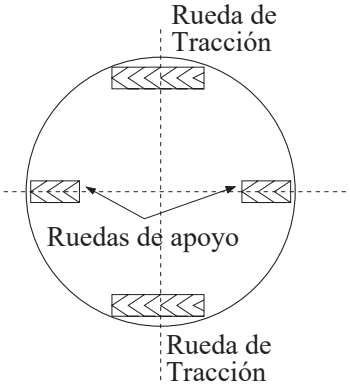
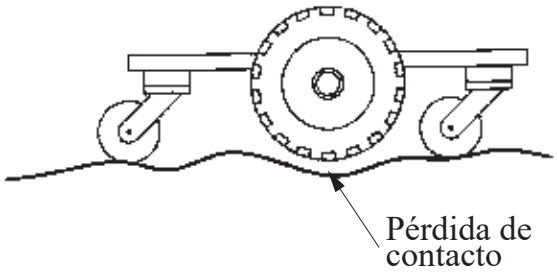


Figura 2.2: Configuración diferencial Vs la de tipo automóvil [Jones99].



(a) ***Configuración en diamante***



(b) ***Vista lateral del chasis del robot móvil, el cual ilustra el problema de las ondulaciones en el terreno [Jones99].***

Figura 2.3: Detalles.

2.1.1. Esquema eléctrico

En la figura 2.4 se muestra un esquema de conexiones muy general, es decir, con poco nivel de detalle. El diagrama detallado de conexiones se encuentra en el apéndice D.

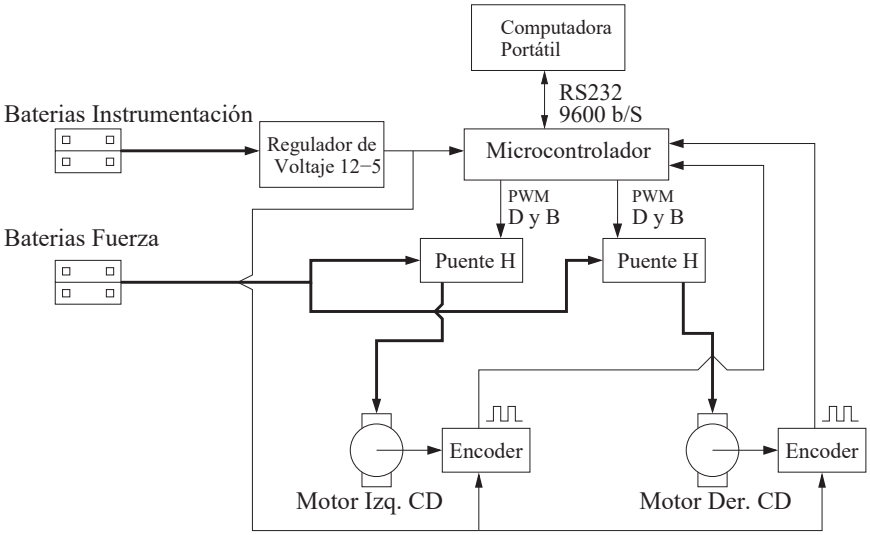


Figura 2.4: Esquema de conexiones.

El esquema de la figura 2.4 está dividido en dos etapas, la etapa de potencia y la etapa de instrumentación. Cada una cuenta con un par de baterías recargables de 12 volt, como fuente de alimentación. La etapa de potencia está conformada por un par de motores de C.D. y un par de puentes H que controlan su funcionamiento. La parte de instrumentación está conformada por un regulador de voltaje el cual alimenta tanto al microcontrolador como al par de encoders acoplados a las flechas de cada motor. La conexión entre la etapa de potencia y la etapa de control ocurre a través del puente H, ya que éste recibe señales provenientes del microcontrolador. Estas señales son: la dirección (Dir), freno (Brake) y PWM. Las señales *Dir* y *Brake* controlan la polaridad con la que son alimentadas las terminales de los motores, con la finalidad de definir el giro de rotación de la flecha de cada motor. La señal PWM (modulación de ancho de pulso) controla la velocidad de giro de las flechas de los motores. Para mayor información sobre el esquema PWM ver [Jones99]. El robot móvil también cuenta con una computadora portátil interconectada con el microcontrolador a través de un bus de transmisión de datos serial (RS-232), a una velocidad de 9600 b/S.

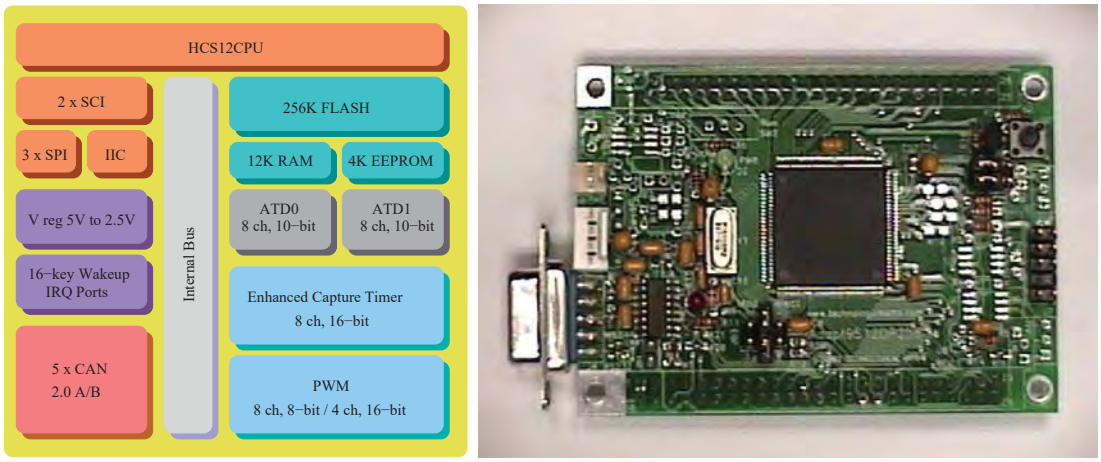
La siguiente sección describe con mayor detalle los componentes que conforman el esquema eléctrico de la figura 2.4.

2.1.2. Descripción del hardware

A continuación realizaremos una descripción detallada del *hardware* del robot móvil, ordenada de acuerdo a su grado de importancia dentro de la estructura del robot móvil:

El Microcontrolador:

El microcontrolador (MCU) MC9S12DP256 de *Motorola* es un dispositivo de 16 bits, compuesto por periféricos y una unidad central de procesamiento (HCS12 CPU) todo en un solo chip. Tiene 256 Kbytes de EEPROM flash, 12 Kbytes de RAM, 4 Kbytes de EEPROM, dos interfaces de comunicación serial asíncrona (SCI), tres interfaces periféricas seriales (SPI), un temporizador de captura mejorado de 8 canales, dos convertidores A/D de 10 bits con 8 canales, un modulo PWM con 8 canales, un controlador de enlace de datos digitales y 29 canales I/O digitales (Puerto A, Puerto B, Puerto K y Puerto E) [Mot02b].



(a) *Diagrama de bloques* **(b) *Módulo Adapt9S12DP256***

Figura 2.5: Diagrama de bloques y módulo Adapt9S12DP256 [Arts02].

El microcontrolador juega un papel muy importante en el funcionamiento del robot móvil, ya que realiza diversas operaciones como el control de la velocidad de los motores, la adquisición de datos y la comunicación con la PC. Debido a la complejidad de estas tareas, se justifica el uso del microcontrolador MC9S12DP256, ya que como se aprecia del párrafo anterior, es poderoso y versátil. Por otro lado en el proyecto ROCA se desarrolló *software* de comunicación y control para este microcontrolador, el cual se tomó como base para el desarrollo del presente trabajo.

En la figura 2.5(a) se muestra un diagrama de bloques del microcontrolador usado, así como el módulo de evaluación **Adapt9S12DP256** basado en el microcontrolador descrito anteriormente [Arts02].

Motores y encoders:

En muchas ocasiones los motores son llamados los músculos del robot [McComb01], debido a que éstos realizan el trabajo concerniente al movimiento del robot móvil. Los motores o servomecanismos de corriente directa (C.D) son muy comunes en la robótica por su relativa facilidad para ser controlados.

Para la tracción de las ruedas del robot móvil descrito en esta tesis, se usaron dos motores de la marca *Pittman* de C.D. a 24 V, de la serie GM9000 dotados con una caja de engranes para reducir la velocidad e incrementar el par (ver apéndice E), incluyen también un encoder óptico de la marca *Hewlett-Packard* (HP). El encoder cuenta con tres canales, dos de ellos con 500 ranuras por vuelta (CPR), y defasados 90^o entre sí. El canal restante llamado index tiene únicamente una ranura por vuelta. Este tipo de configuración permite, además, de medir la velocidad y posición de la flecha del motor, permite determinar el sentido de giro de la flecha (ver sec. 2.3).

En la figura 2.6 se muestra una foto del motor descrito en los parrafos anteriores. Es importante comentar que la selección de los motores de C.D. para el robot móvil se realizó en trabajos previos [Romero01], no obstante, cuando se va a realizar la selección de un motor para un nuevo proyecto es necesario tomar encuenta las condiciones de trabajo del motor de C.D. Lo anterior implica la selección de algunos parámetros como voltaje,

corriente, par entre otros [Jones99].

Figura 2.6: Motor 24V CD marca Pittman [Pittman01].

Puentes H:

En general, un puente H es un arreglo de cuatro transistores como se muestra en figura 2.7 y es quizá el arreglo más común para el control de motores de C.D. La idea básica que gobierna el funcionamiento de este arreglo es que sólo un par de transistores están encendidos a la vez. Por ejemplo, si Q1-Q4 están encendidos el nodo A se conecta con +V y el nodo B a tierra, de manera similar si Q2-Q3 están encendidos la conexión con +V y tierra ahora es de B a A. Este par de posibilidades permite que la flecha del motor de C.D. gire en ambas direcciones, digamos por ejemplo sentido horario y antihorario.

En nuestro caso se utilizó el circuito integrado LMD18200, el cual es un puente H de 3 A. desarrollado por la *National Semiconductor*. Este circuito fue diseñado para proyectos de control de velocidad y tiene diversas aplicaciones como por ejemplo: el diseño de *drivers* para motores de pasos y de CD, robots de automatización industrial, maquinaria de control numérico, impresoras y *plotters* entre otras. Este dispositivo es construido usando un proceso multi-tecnológico el cual combina circuitería de control bipolar y CMOS con circuitería de potencia DMOS, todo esto en una misma estructura monolítica. Para mayor detalle ver el apéndice E.

Baterías:

Las baterías utilizadas son baterías recargables de plomo-acído de 12 volts y 7 A-H. Las baterías de plomo-acído son muy comunes en automóviles o incluso en motocicletas [McComb01].

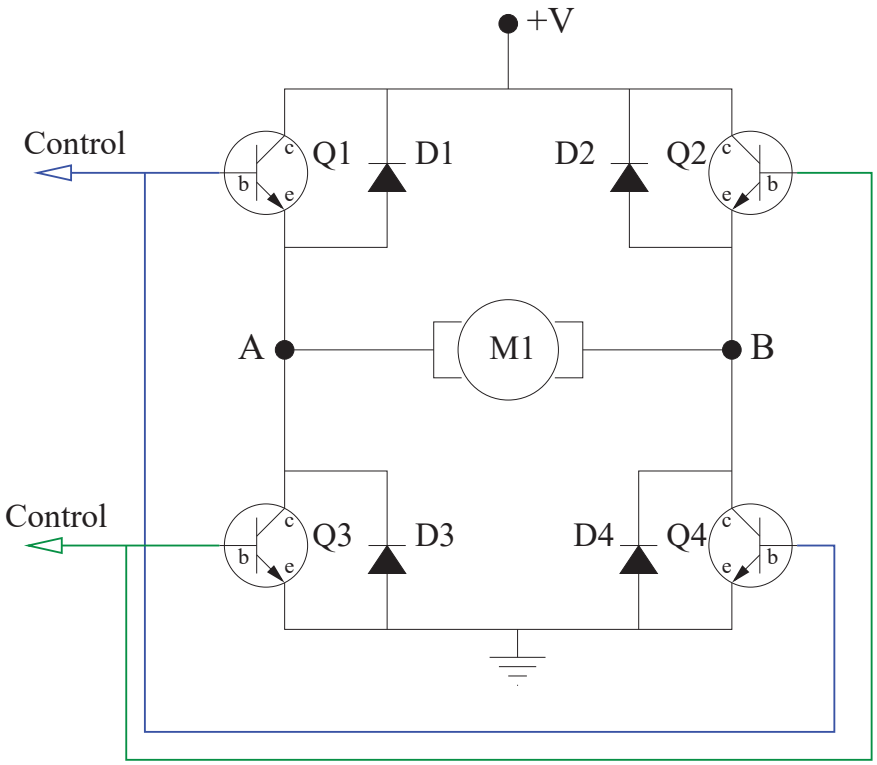


Figura 2.7: Conexión de cuatro transistores en arreglo H [McComb01].

Se componen de unas cuantas placas de plomo sumergidas en un contenedor con una base ácida electrolítica, y aunque tienen buena retención de carga, son algo pesadas.

2.2. Modelado

El modelado matemático es un aspecto muy importante para el logro de las metas propuestas en esta tesis, ya que este nos dará un mayor grado de conocimiento del sistema con el que estamos tratando. El grado de precisión del modelo depende en gran medida de la cantidad de información que incorporemos a él. Para la construcción de un modelo se hace uso de leyes físicas y relaciones constitutivas asociadas a los componentes del sistema. En las siguientes subsecciones describiremos diferentes tipos de modelos concernientes a la arquitectura de tipo diferencial.

2.2.1. Modelo cinemático

La *cinemática* es la parte de la mecánica que estudia el movimiento sin considerar las fuerzas que lo propician. Para su análisis y estudio, la cinemática además de la definición citada, tiene relación directa con dos aspectos que son de gran importancia para la construcción de un modelo cinemático para un robot móvil [Ribeiro02]:

- Las relaciones geómetricas que gobiernan el sistema.
- Las relaciones entre los parámetros de control y el comportamiento del sistema en estado estable.

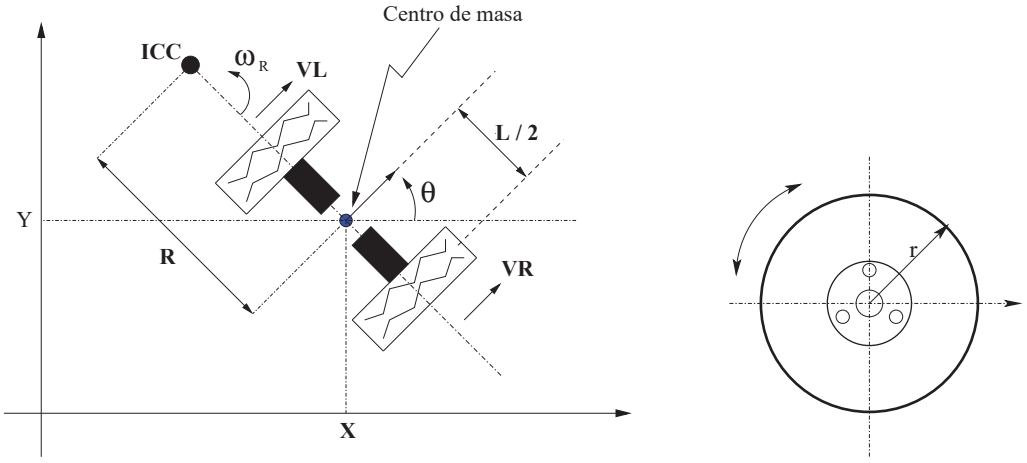
En la sección anterior se habló sobre la configuración del robol móvil, con lo cual podemos ahora establecer algunos parámetros físicos que nos llevaran a la construcción del modelo cinemático del robot móvil. En la figura 2.8 se muestra un esquema detallado de los aspectos geométricos de la arquitectura del robot. En la misma figura se ilustran las siguientes variables y parámetros cinemáticos del robot móvil:

- V_R: Velocidad lineal de la llanta de tracción derecha.
- V_L: Velocidad lineal de la llanta de tracción izquierda.
- r: Radio nominal de cada llanta.
- R: Radio instantáneo de curvatura de la trayectoria del robot, relativo al centro de masa.
- Icc: Centro de giro, relativo al centro del robot.
- (x,y): Posición instantánea del centro del robot con respecto a un marco base.
- θ: Orientación instantánea del robot móvil con respecto al eje X.
- ω_R: Velocidad angular instantánea de la estructura del robot móvil.
- L/2: Distancia entre una llanta, y el centro del robot.

Antes de comenzar a discutir las relaciones matemáticas que gobiernan el comportamiento de este sistema, es importante comentar la relación existente entre la velocidad lineal (V) y la velocidad angular (ω) de un cuerpo arbitrario en movimiento circular con un radio (r_c), la cual será utilizada en el transcurso del desarrollo del modelo cinemático. Esta relación está descrita por la siguiente ecuación [Sears67].

$$V = \omega r_c \tag{2.1}$$

La cinemática tiene relación directa con los parámetros de control y el comportamiento del sistema en estado estable. La velocidad de las llantas es un parámetro que influye directamente en la evolución del sistema (robot móvil), ya que la variación de ésta, determina la trayectoria del movimiento del robot móvil.



(a) ***Cinemática Diferencial (Vista de planta) [Dudek00].***

(b) ***Detalles de la rueda.***

Figura 2.8: Parámetros Geométricos.

De la relación descrita por (2.1) tenemos que [Dudek00]:

$$V_R(t) = \omega_R(t)(R + \frac{L}{2})$$
$$V_L(t) = \omega_R(t)(R - \frac{L}{2}) \tag{2.2}$$

Resolviendo (2.2) para R y ω_R tenemos el siguiente resultado:

$$R = \frac{L}{2}\frac{V_L(t) + V_R(t)}{V_R(t) - V_L(t)}, \tag{2.3}$$

$$\omega_R(t) = \frac{V_R(t) - V_L(t)}{L} \tag{2.4}$$

Sustituyendo (2.3) y (2.4) en (2.1), obtenemos una expresión para la velocidad instantanea del centro del robot.

$$V(t) = \omega_R(t)R = \frac{1}{2}(V_R(t) + V_L(t)) \tag{2.5}$$

Existen varios casos de interés que conviene comentar para la ecuación (2.3). Por ejemplo si V_L=V_R el radio R es infinito, lo cual quiere decir que el robot móvil se mueve en línea recta. Otro caso es cuando V_L=$-V_R$ dando como resultado un radio igual cero y, por lo tanto, el robot rotará sobre su propio eje. Para otros valores de V_L y V_R el robot describirá una trayectoria curva. Por otro lado, la estructura cinemática de la configuración de tipo diferencial no permite cierto tipo de movimientos, como por ejemplo, no existe ninguna combinación para V_L y V_R tal que el robot móvil se mueva sobre el eje de rotación de las llantas, es decir, perpendicular a las llantas de tracción.

2.2.2. Cinemática directa

La determinación de las posiciones y orientaciones (x,y,θ) alcanzables por el robot, dados los parámetros de control V_L y V_R, es conocido como el problema de la *cinemática*

directa (ver figura 2.9). Si el robot tiene una postura (x,y,θ) en el instante t y las velocidades V_L y V_R se mantienen constantes durante un periodo de tiempo, entonces las coordenadas del punto ICC están dadas por.

$$ICC = \left\{ \; x - Rsen(\theta), y + Rcos(\theta) \; \right\} \tag{2.6}$$

La siguiente ecuación describe el movimiento de rotación del robot móvil sobre el punto ICC con una velocidad angular dada por ω [Dudek00].

$$\begin{bmatrix} x'(t) \\ y'(t) \\ \theta'(t) \end{bmatrix} = \begin{bmatrix} cos\theta(t) & 0 \\ sen\theta(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ w(t) \end{bmatrix} \tag{2.7}$$

Si integramos la ecuación (2.7), es posible saber la posición del robot en el instante de tiempo t, en base a los parámetros de control V_L(t) y V_R(t):

$$\begin{aligned} x(t) &= \int_0^t v(t)cos(\theta(t))dt \\ y(t) &= \int_0^t v(t)sen(\theta(t))dt \\ \theta(t) &= \int_0^t w(t)dt \end{aligned} \tag{2.8}$$

Sustituyendo las ecuaciones (2.4) y (2.5) en (2.8) obtenemos el siguiente resultado:

$$\begin{aligned} x(t) &= \tfrac{1}{2}\int_0^t (V_R(t) + V_L(t))cos(\theta(t))dt \\ y(t) &= \tfrac{1}{2}\int_0^t (V_R(t) + V_L(t))sen(\theta(t))dt \\ \theta(t) &= \tfrac{1}{L}\int_0^t (V_R(t) - V_L(t))dt \end{aligned} \tag{2.9}$$

2.2.3. Cinemática inversa

Un problema más interesante para este trabajo, es el de la *cinemática inversa*, el cual consiste en encontrar los parámetros de control V_L y V_R, del tal manera que éstos nos lleven a una posición o al seguimiento de una trayectoria específicos.

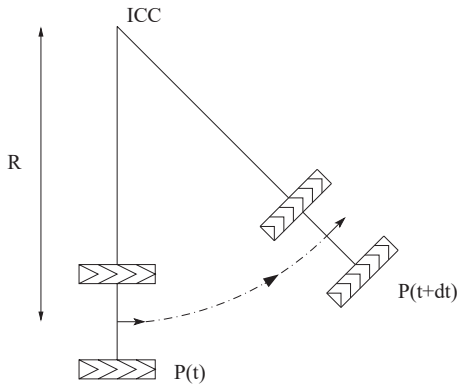


Figura 2.9: Geometría de la cinemática directa [Dudek00].

Debido a la restricción de la no holonomicidad ([Dudek00] y [Ribeiro02]) representada en las ecuaciones (2.8) y (2.9) no es posible encontrar una solución en general para este problema, así que para lograr que el robot logre una postura (x,y,θ) podemos usar los casos citados, es decir podemos hacer que el robot rote sobre su centro hasta lograr alinearse con (x,y), después podemos hacerlo avanzar hacia adelante hata alcanzar un objetivo y por último una rotación que logre la orientación θ. Por supuesto que esta forma de lograr el objetivo (x,y,θ) no es la única ni la más apropiada, ya que se requiere de un gran esfuerzo por parte de los actuadores. Una alternativa para la solución a este problema se basa en el uso de trayectorias suaves el cual se explica más adelante (ver capítulos 3 y 4).

2.2.4. Modelo de los motores de C.D.

El motor de C.D. es un sistema electromecánico ampliamente estudiado en la ingeniería [Rodríguez98], y el modelado de éste se incluye aquí, ya que el sistema (robot móvil) construido, cuenta con dos motores de este tipo.

El tipo de motor de C.D. utilizado para la construcción del robot móvil, es del tipo de imán permanente, el cual se puede modelar como un motor de C.D. con excitación independiente. Para esto consideremos el circuito equivalente mostrado en la figura 2.10.

Aplicando ley de voltajes al circuito de armadura tenemos que.

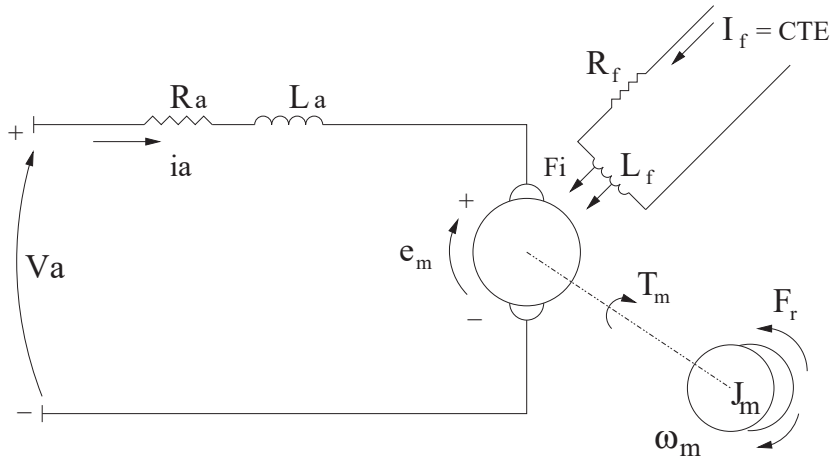


Figura 2.10: Circuito equivalente del motor de C.D [Rodríguez98].

$$V_a(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + e_m(t) \tag{2.10}$$

Donde $e_m(t)$ es el voltaje generado por la fuerza contraelectromotriz proporcional a la velocidad angular ω_m y al campo magnético ϕ, que es constante debido al imán permanente. La ecuación (2.11) describe esta relación.

$$e_m(t) = K_E \omega_m(t) \tag{2.11}$$

Donde: $K_E = K'\phi$, se le llama constante de fuerza contraelectromotriz.

Sustituyendo la ecuación (2.11) en (2.10) y reacomodando los términos tenemos el siguiente resultado.

$$\frac{di(t)}{dt} = -\frac{R_a}{L_a} i_a(t) - \frac{K_E}{L_a} \omega_m(t) + \frac{1}{L_a} V_a \tag{2.12}$$

Ahora para la parte mecánica, si aplicamos la segunda ley de Newton (movimiento angular) a la carga conectada a la flecha del motor tenemos que.

$$T_m(t) = J_m \frac{d\omega_m(t)}{dt} + D\omega_m(t) \tag{2.13}$$

Donde J_m es la inercia de la flecha del motor y D es el coeficiente de fricción viscosa (ver apéndice E).

El par T_m producido por el motor depende de la corriente de armadura y de la intensidad del campo magnético como sigue.

$$T_m(t) = K"\phi i_a(t) = K_T i_a(t) \tag{2.14}$$

Donde $K_T = K"\phi$ es la constante de par.

Si sustituimos (2.14) en (2.13) y reordenamos los términos de la misma manera que en (2.12) obtenemos el siguiente resultado.

$$\frac{d\omega_m(t)}{dt} = \frac{K_T}{J_m} i_a(t) - \frac{D}{J_m}\omega_m(t) \tag{2.15}$$

De (2.12) y (2.15) podemos escribir las ecuaciones de estado del motor de C.D. en forma matricial como sigue.

$$\begin{bmatrix} i_a'(t) \\ \omega_m'(t) \end{bmatrix} = \begin{bmatrix} -R_a/L_a & -K_E/L_a \\ K_T/J_m & -D/J_m \end{bmatrix} \begin{bmatrix} i_a(t) \\ \omega_m(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} \\ 0 \end{bmatrix} V_a \tag{2.16}$$

El modelo anterior describe el comportamiento tanto de la corriente del circuito de armadura, como de la velocidad angular de la flecha del motor. Esta última es nuestra variable de interés ya que con ella se pretende controlar el movimiento de nuestro robot móvil. Para esto resulta adecuado introducir una forma alterna al modelo en varibles de estado la cual exponga las variables de interés. Esta forma es la función de transferencia,

donde la variable de entrada es el voltaje de armadura y la salida es la velocidad angular.

Si transformamos las ecuaciones (2.10), (2.11), (2.13) y (2.14) al dominio de *Laplace* obtenemos:

$$\begin{aligned} V_a(S) &= R_a.I_a(S) + L_a[S.I_a(S) - i_a(0)] + E_m(S) && (2.17)\\ E_m(S) &= K_E.\Omega(S) && (2.18)\\ T_m(S) &= J_m[S.\Omega(S) - \omega(0)] + D.\Omega(S) && (2.19)\\ T_m(S) &= K_T.I_a(S) && (2.20) \end{aligned}$$

Ahora si consideramos que $i_a(0) = 0$ y sustituyendo (2.18) en (2.17) tenemos.

$$V_a(S) = I_a(S).(SL_a + R_a) + K_E.\Omega(S) \tag{2.21}$$

De la misma manera si $\omega(0) = 0$ y sustituyendo (2.20) en (2.19) tenemos.

$$K_T.I_a(S) = \Omega(S).(SJ_m + D) \tag{2.22}$$

Despejando la corriente de (2.22) y sustituyéndola en (2.21) obtenemos el resultado esperado.

$$P(S) = \frac{\Omega(S)}{V_a(S)} = \frac{K_T}{S^2 J_m L_a + S.(J_m R_a + D L_a) + D R_a + K_T K_E} \tag{2.23}$$

Ahora bien si sustituimos en (2.23) los valores de la hoja de datos del motor de C.D. (ver apéndice E) obtenemos el siguiente resultado.

$$P(S) = \frac{\Omega(S)}{V_a(S)} = \frac{4.58x10^{-2}}{18.67x10^{-9}S^2 + 17.68x10^{-6}S + 2.10x10^{-3}} \tag{2.24}$$

El cual usaremos para analisar y comparar con los resultados obtenidos en 2.2.4.

El siguiente desarrollo es un enfoque alterno del motor de C.D., el cual usaremos en la subsección 2.2.5 para el análisis dinámico. La figura 2.11 muestra un esquema similar al mostrado en la figura 2.10 solo que esta vez se incluye una caja de engranes conectada a la flecha del motor [Beard02].

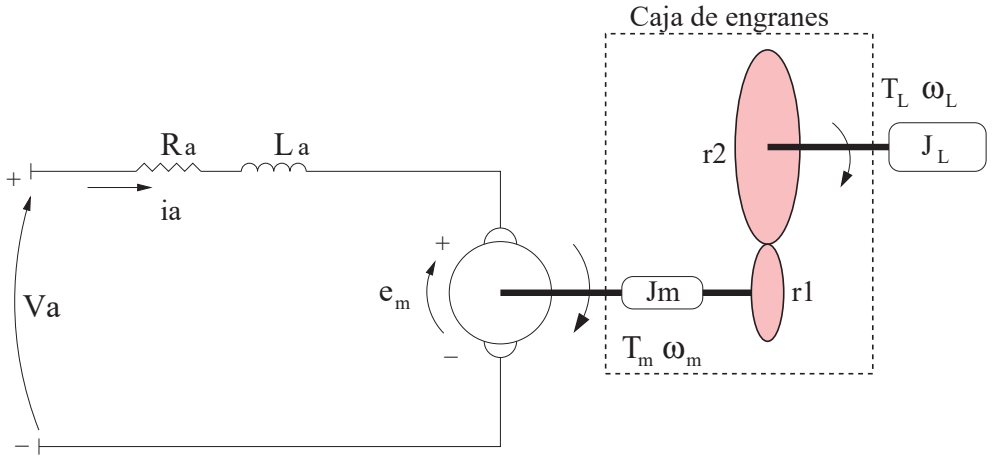


Figura 2.11: Circuito equivalente del motor de CD con carga [Beard02].

La relación (2.25) denota la constante de reducción, la cual es comúnmente encontrada en las hojas de datos de los motores (ver apéndice E).

$$Kg = \frac{r2}{r1} \tag{2.25}$$

Donde:

r1 y r2: Radios de los engranajes.

J_L: Inercia de la carga.

T_L: Par ejercido en la carga.

Con esto la relación entre T_m y T_L está dada por la siguiente ecuación.

$$T_L = K_g T_m \tag{2.26}$$

Mientras que para ω_m y ω_L está dada por.

$$\omega_L(t) = \frac{1}{K_g}\omega_m(t) \tag{2.27}$$

Un enfoque que refleja el hecho de conectar una carga a través de una caja de engranes es dado a continuación. Si despreciamos la inductancia en el circuito de armadura la ecuación 2.10 se reduce a la siguiente expresión.

$$V_a(t) = R_a * i_a(t) + e_m(t) \tag{2.28}$$

Usando (2.11) y (2.14) obtenemos.

$$V_a(t) = \frac{R_a}{K_T}T_m + K_E\omega_m(t) \tag{2.29}$$

Usando (2.26) y (2.27) para reflejar la carga tenemos.

$$V_a(t) = \frac{R_a}{K_g K_T}T_L + K_g K_E\omega_L(t) \tag{2.30}$$

La inercia total vista del lado de la carga es calculada de la siguiente manera [Beard02].

$$J_T = J_L + K_g^2 J_m \tag{2.31}$$

Usando la segunda ley de Newton.

$$T_L = J_T \frac{d\omega_L(t)}{dt} \tag{2.32}$$

Sustituyendo la ecuación (2.32) en (2.30) tenemos.

$$\frac{d\omega_L(t)}{dt} = -\frac{K_g^2 K_T K_E}{R_a(J_L + K_g^2 J_m)}\omega_L(t) + \frac{K_g K_T}{R_a(J_L + K_g^2 J_m)}V_a(t) \tag{2.33}$$

Definiendo k_1 y k_2 como.

$$K_1 = \frac{K_g K_T}{R_a(J_L + K_g^2 J_m)}$$
$$K_2 = \frac{K_g^2 K_T K_E}{R_a(J_L + K_g^2 J_m)}$$

Finalmente la dinámica del motor, incluyendo la reducción y la carga puede ser escrita como.

$$\frac{d\omega_L(t)}{dt} = -K_2\omega_L(t) + K_1 V_a(t) \tag{2.34}$$

Las constantes K_1 y K_2 pueden ser determinadas ya sea por identificación de sistemas o utilizando la hoja de datos que provee el fabricante para hacer una aproximación. En nuestro caso y aprovechando las mediciones de las múltiples pruebas realizadas con el robot móvil utilizaremos el primer método. Debido a que no es el objetivo primordial de este trabajo la identificación de sistemas no entraremos en detalle sobre él. En el apéndice G se muestran el método utilizado y los resultados obtenidos.

Validación del modelo de los motores y experimentación

La dinámica de un sistema puede ser determinada experimentalmente por la respuesta del mismo; a pulsos, escalones, rampas u otro tipo de señales que expongan el comportamiento del sistema. Para la realización de este experimento se requiere que el sistema se encuentre en reposo y que no existan errores de medición, sin embargo, en muchas ocasiones ésto es difícil de lograr, sobre todo en las mediciones, donde siempre existe un factor de ruido que debe tomarse en cuenta al momento de ser interpretadas.
El escalón es el tipo de señal que se usó para experimentar con nuestro sistema (Robot móvil) ya que es fácil de implementar.

Medición de la velocidad: El encoder como se describió en la sección 2.1.2 es un disco perforado (500 CPR cuentas por revolución) acoplado a la flecha del motor (ver figura 2.12). El tipo de información que entrega este sensor es un tren de pulsos, el cual debe ser interpretado de la siguiente manera. El microcontrolador cuenta con un módulo contador de pulsos [Mot02a] el cual puede ser activado para funcionar ya sea en los flancos de subida, flancos de bajada o en ambos (ver fig 2.13). En nuestro caso usaremos flancos de subida y bajada con la finalidad de tener mayor resolución. Así pues la velocidad se calcula como número de pulsos entre el periodo de muestreo. La ecuación (2.35) muestra esta relación [Ramirez Z98].

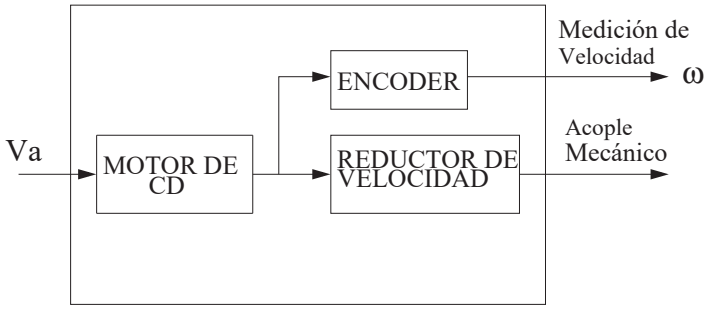


Figura 2.12: Diagrama esquemático del actuador del Robot móvil.

$$\omega = \frac{Num.\ pulsos}{Periodo\ de\ muestreo} \tag{2.35}$$

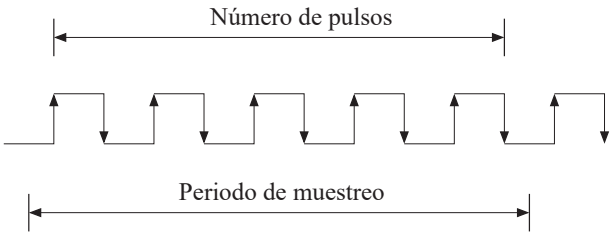


Figura 2.13: Medición de la velocidad.

Existen otras relaciones que transforman las unidades de medición de la velocidad que pueden ser de gran utilidad [kheir00].

$$RPM = (\frac{Num.\ pulsos}{Periodo\ de\ muestreo})(\frac{1\ Rev.}{Pulsos\ por\ rev.})(\frac{60}{1min}) \tag{2.36}$$

$$rad/seg = (\frac{Num.\ pulsos}{Periodo\ de\ muestreo})(\frac{2\pi}{Pulsos\ por\ rev.}) \tag{2.37}$$

El microcontrolador además de usarse para la realización de diversas tareas comentadas en 2.1.2, sirvió para la implementación del escalón aplicado a los actuadores del robot móvil, así como para el almacenamiento y envío de datos a la PC para su posterior análisis. El diagrama de flujo de la figura 2.14 muestra el inicio de la secuencia de actividades para la realización del experimento del escalón.

Primeramente la inicialización de módulos del microcontrolador, realiza operaciones como la fijación de la frecuencia del reloj, inicialización del PWM y activación del contador de pulsos entre otras. Posteriormente, se habilita la interrupción del puerto serie, y a través de ésta, se ejecutará la rutina del escalón, adquisición y envío de datos. El diagrama de flujo de la figura 2.15 muestra esta última.

En el diagrama de flujo de la figura 2.15 hay algunos aspectos que deben ser comentados. La primera parte, implementa el escalón mediante los comandos Dir, Brake y

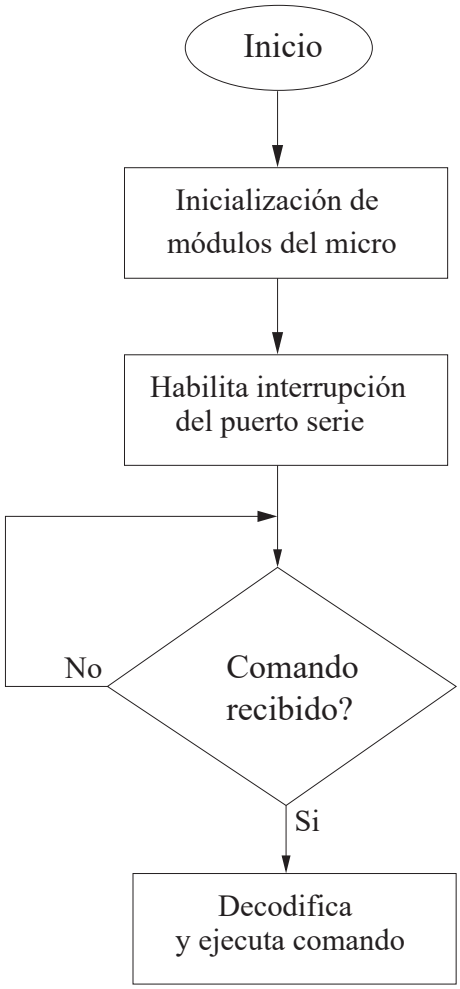


Figura 2.14: Diagrama de flujo principal.

PWM. Poniendo la salida del PWM al 100 se obliga de manera súbita a los puentes H a suministrar su máximo de voltaje, en este caso $24volt$, implementando así un escalón de esta magnitud. En la segunda parte se envían los datos vía puerto serie. La adquisición de datos se realiza de forma paralela una vez activada la interrupción de tiempo real (ver figura 2.16). Este proceso llega a su fin una vez completado el número de datos deseados. En este caso se requirieron 150 datos, muestreados cada 1.024ms.

El segmento de código 2.1 muestra la rutina de la interrupción de tiempo real, donde se realiza la adquisición y almacenamiento de las muestras de velocidad. El resto del

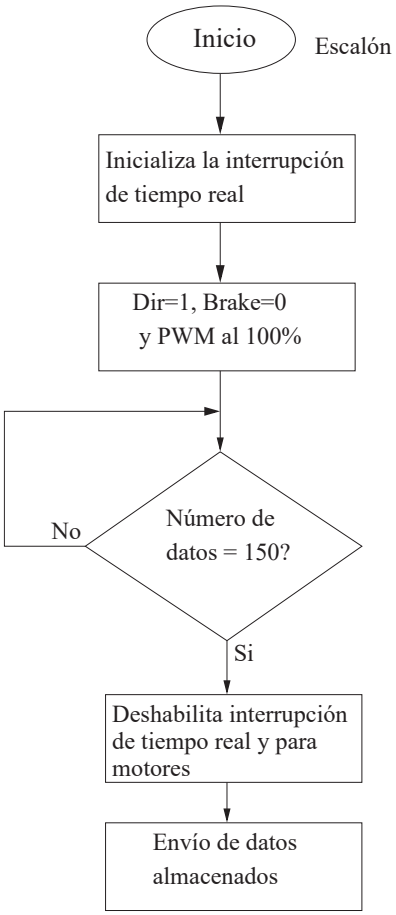


Figura 2.15: Rutina del escalón.

código se puede ver en el apéndice C.

```
//-------------------------------------------------------------------------------------
void  __attribute__ ((interrupt)) ISR_RealTimeInt(void)
{
   CRGFLG |= RTIF;      //clear flag
   if (im <  NPULSOS ){
        MCCTL |= ICLAT;         //Escribe el contenido de los acumuladores
        pulsos_izq[im] = PA2H;  //en sus registros asociados (PA2H Y PA3H en este caso)
        pulsos_der[im] = PA3H; //Almacena la lectura de pulsos (PT2) Canal A del encoder
        im++;
   }
}
//-------------------------------------------------------------------------------------
```

Código 2.1. Interrupción de tiempo real.

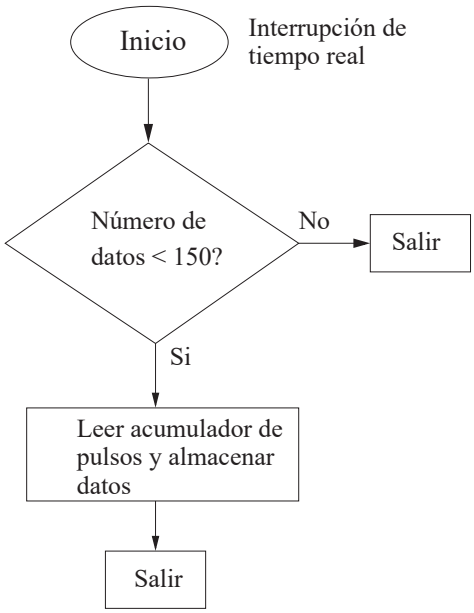


Figura 2.16: Interrupción de tiempo real.

Respuesta al escalón del motor de C.D. en vacío: La figura 2.17 muestra la curva de arranque del motor de CD en vacío obtenida a través del encoder, y la obtenida con la simulación del modelo matemático construido con la hoja de datos que provee el fabricante (ver ec. 2.24), ambas curvas se obtuvieron aplicando una entrada $V_a = 24\ v$. Como se puede observar existe una ligera diferencia entre ambas respuestas, la cual puede ser debida a que el modelo matemático construido con la hoja de datos del motor de C.D. (ver apéndice E) no considera la caja de engranes conectada a la flecha del motor, y ésta actúa como una ligera carga conectada a la flecha.

La comparación entre las dos curvas es útil, ya que expone el grado de error de la medición. Así por ejemplo, si la respuesta obtenida a partir del encoder difiere completamente de la obtenida por el modelo matemático, esto sería un indicativo de la existencia de un error, y por tanto sería recomendable modificar el sistema de medición con la finalidad de obtener mejores resultados. En nuestro caso la curva de velocidad obtenida muestra un patrón de comportamiento muy similar a la del modelo, lo cual nos alienta a confiar en dicha adquisición.

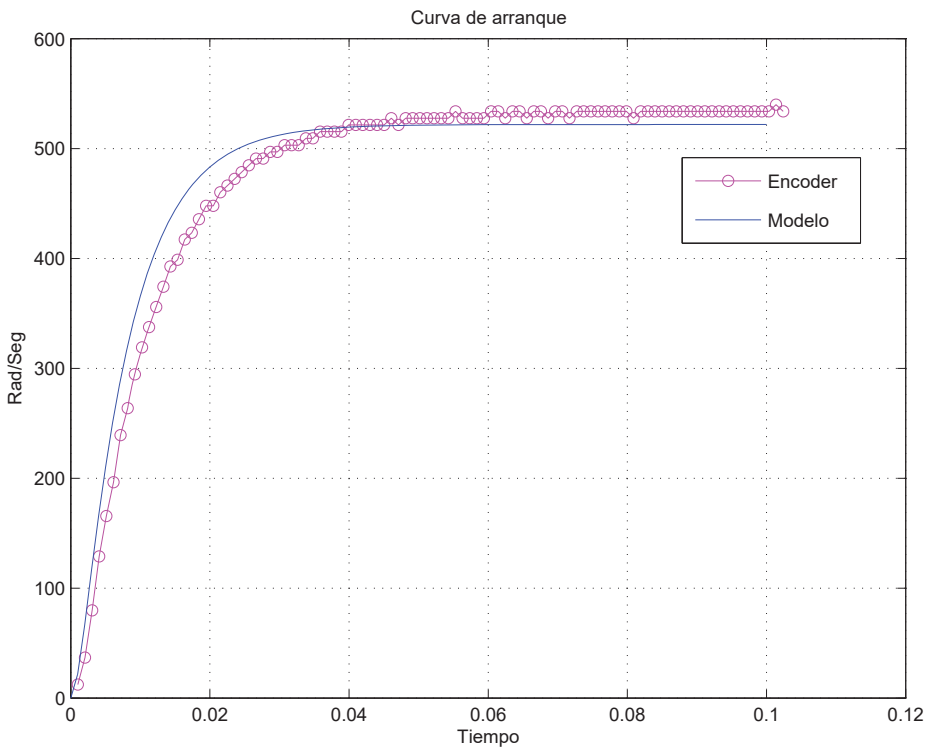


Figura 2.17: Respuesta al escalón del sistema en vacío.

Por otra parte, es posible obtener algunos parámetros de interés del sistema a partir de la curva de velocidad mostrada en la figura 2.17. Por ejemplo, la velocidad nominal del sistema, el tiempo que tarda en alcanzar el estado estable y la ganancia del sistema. En nuestro caso la velocidad máxima es aproximadamente $\omega_{nom} = 533\ r/s$ con un tiempo de establecimiento de t=0.06s aproximadamente, mientras que la ganancia se calcula como sigue.

$$\frac{\omega_{nom}}{V_a} = \frac{533}{24} = 22.2$$

El cálculo de estos parámetros puede ser de gran utilidad para el diseño de un controlador, sin embargo esto lo comentaremos oportunamente en la sección 2.3.

En la figura 2.18 se muestran las curvas de arranque de ambos motores en función de pulsos por milisegundo . Por comodidad de aquí en adelante usaremos como unidades de velocidad, los pulsos por milisegundo, ya que stas son las unidades medidas por el micro-

controlador. De cualquier manera y en caso de considerarse oportuno podemos transformar estas unidades usando las ecuaciones (2.36) y (2.37) descritas anteriormente.

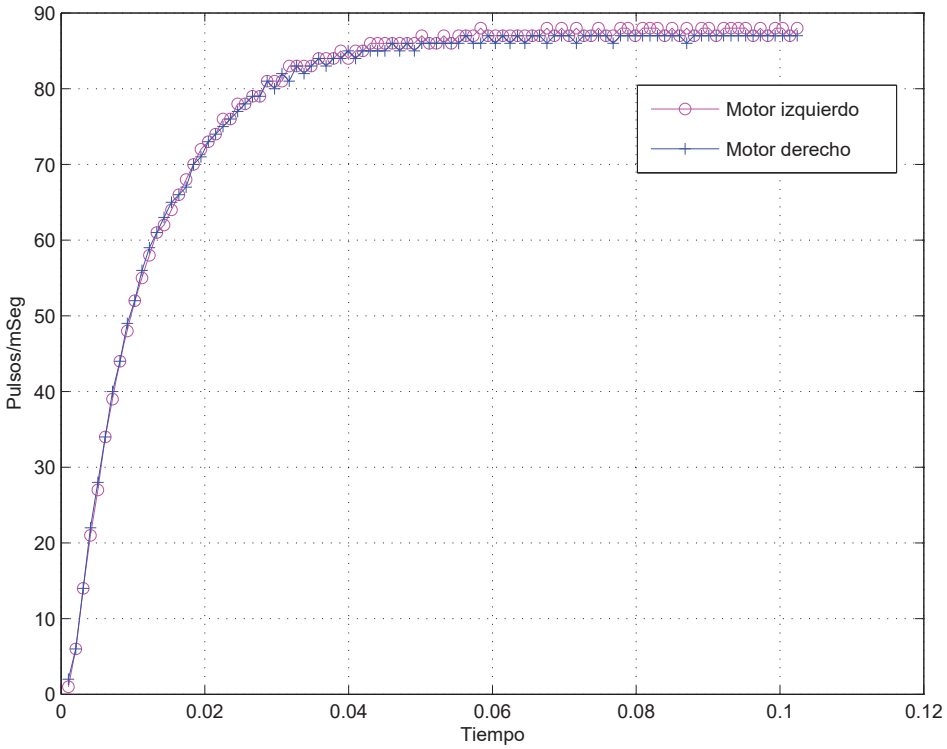


Figura 2.18: Respuesta al escalón del motor izquierdo y derecho sin carga.

Respuesta al escalón del sistema (Robot móvil): Las pruebas descritas en los párrafos anteriores mostraron el comportamiento de los motores de C.D. en vacío. Esto es, desconectados mecánicamente del robot móvil. A continuación se describen algunos experimentos realizados una vez que los motores han sido acoplados al chasis del robot móvil para impulsarlo sobre una superficie plana.

La curva mostrada en la figura 2.19 corresponde a uno de los primeros experimentos realizados al sistema (robot móvil), en donde como se puede observar, tiene una dinámica muy diferente con respecto a las mostradas en los experimentos previos. Por ejemplo en esta curva se observa un cambio brusco en la velocidad, aproximadamente en t=0.015s. Hasta aquí es difícil poder dar un diagnóstico de la causa de este comportamiento, se podría pen-

sar en una falla del motor o quizá en la fuente de alimentación (baterías). Para eliminar o asegurar que se trata de esta última posibilidad hay una prueba que podemos implementar relativamente fácil a través del convertidor analógico digital del microcontrolador [Mot02b]. Esta prueba es la medición de la corriente en el circuito de armadura de los motores de C.D. (Ver apéndice F). Con las mediciones de corriente y velocidad se pretende encontrar la naturaleza del comportamiento descrito.

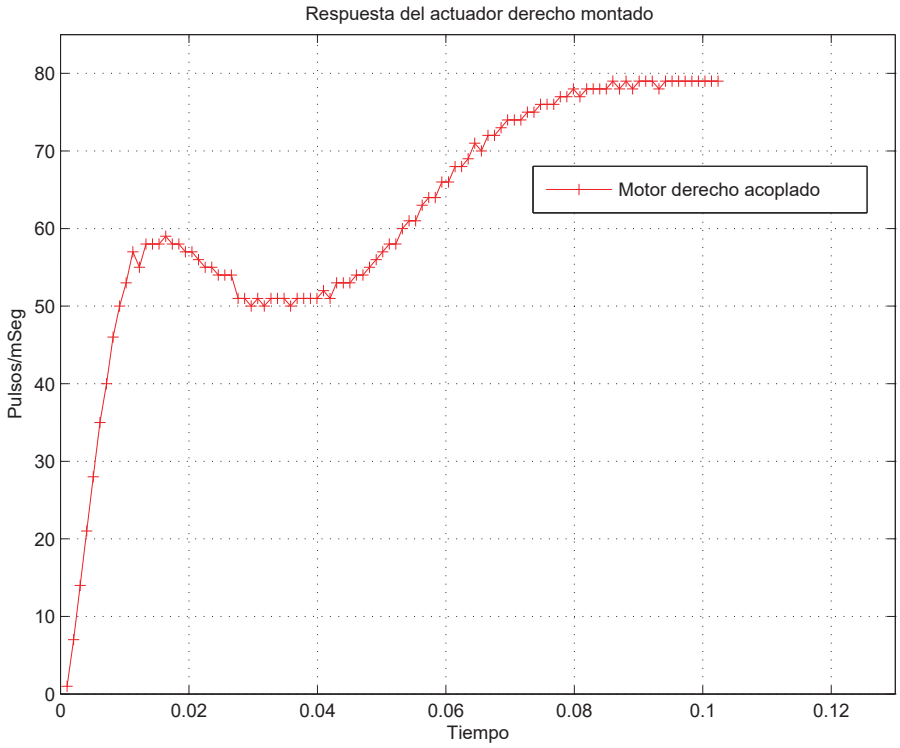
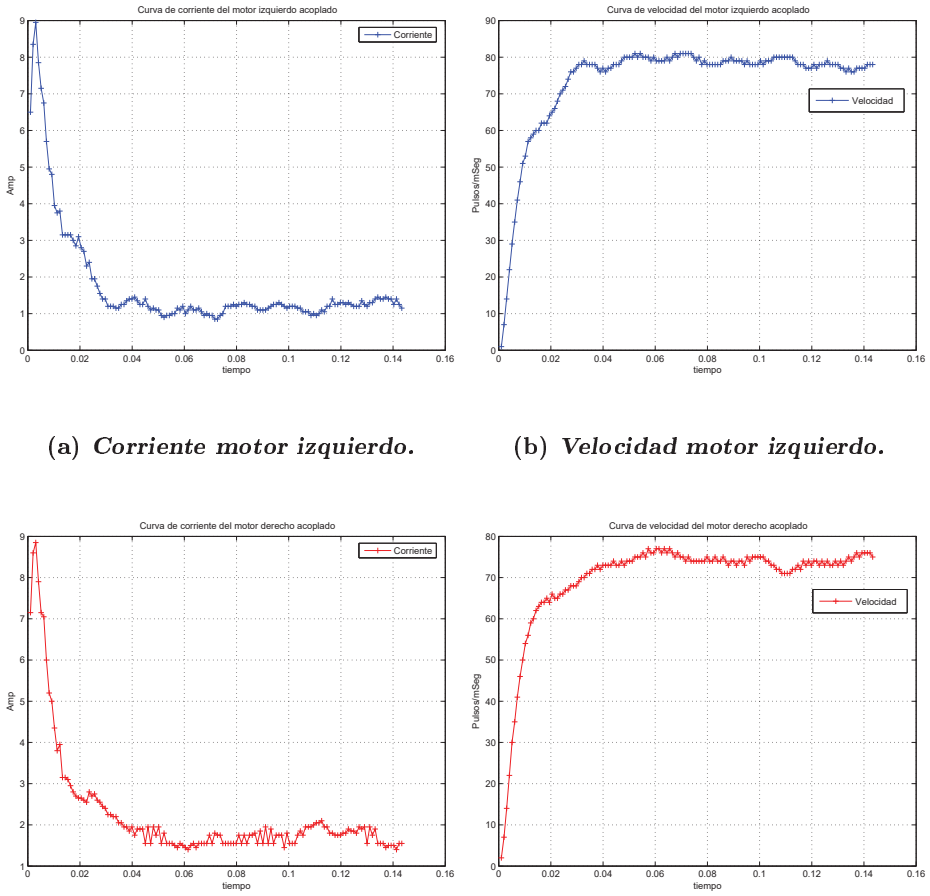


Figura 2.19: Respuesta al escalón del motor derecho acoplado al chasis del robot móvil.

En la figura 2.20 se muestran las mediciones de corriente y velocidad obtenidas al aplicar una entrada escalón a ambos motores. Como se puede observar siguen apareciendo comportamientos en forma de oscilaciones, las cuales más que ser acasionadas por una falla, nos inclinan a pensar que se tratan de comportamientos naturales del sistema.



(a) ***Corriente motor izquierdo.*** **(b)** ***Velocidad motor izquierdo.***

(c) ***Corriente motor derecho.*** **(d)** ***Velocidad motor derecho.***

Figura 2.20: Medición de la corriente y velocidad.

Es importante comentar que los experimentos mostrados en las figuras 2.19 y 2.20 son sólo algunos de los muchos realizados con nuestro sistema (robot móvil), y se considera que este par de experimentos son los más significativos ya que aportan información valiosa.

Al final de toda la experimentación se adquirió un grado de conocimiento del comportamiento del sistema, por lo que estamos en la posibilidad de concretar que las oscilaciones que se presentan en las curvas, son debidas a comportamientos propios del sistema. Por ejemplo, para el caso del comportamiento observado en la figura 2.19, se llegó a la conclusión que fue debido a un resorte colocado en la rueda de apoyo delantera (ver figura 2.21), el cual se incluyó con la finalidad de resolver el problema de la pérdida de contacto con la superficie. Una breve reseña de este comportamiento es comentada a continuación.

Al inicio, cuando el escalón es aplicado a los motores de C.D. estos responden de manera inmediata incrementando su velocidad y por consecuencia la velocidad de la estructura de robot móvil; sin embargo, cuando la fuerza F ejercida sobre la estructura debida a los motores supera a la ejercida por el resorte, éste comienza a comprimirse momentáneamente provocando con esto un movimiento tipo péndulo, el cual afecta directamente a la velocidad. En la figura 2.21 se muestra este comportamiento.

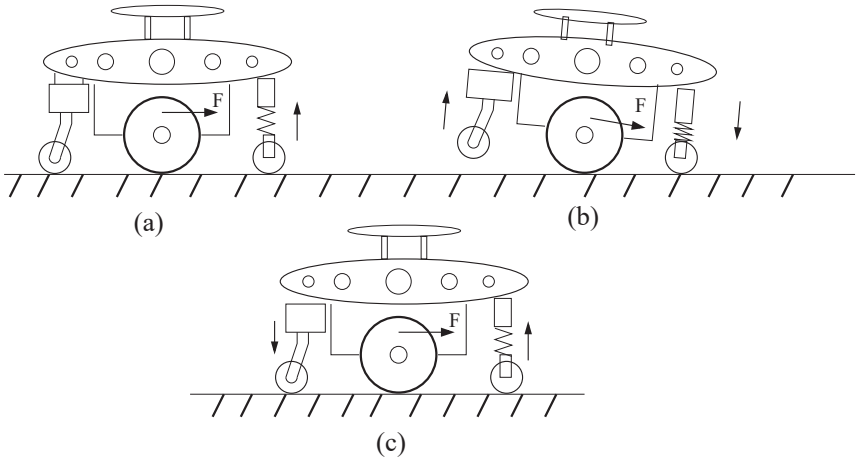


Figura 2.21: Movimiento pendular de la estructura del robot móvil.

Una alternativa la cual fue usada para reducir este efecto consistió en usar un resorte con una constante mayor; sin embargo, es importante no excederse en la rigidez, ya que de lo contrario podríamos tener el problema de la pérdida de contacto con la superficie.

Por otro lado, respecto al comportamiento oscilatorio mostrado en la figura 2.20

podemos comentar que éste fue debido a la fricción no homogénea existente en el acople de la flecha de los motores con sus respectivas llantas.

Los comportamientos descritos en los párrafos anteriores, pueden también ser tratados como perturbaciones propias del sistema. Estas perturbaciones son de origen mecánico y pueden o no, hacer que el sistema sea inestable, sin embargo, con la implementación de un algoritmo de control se espera resolver este problema.

2.2.5. Modelo dinámico

Se denomina *dinámica* a la parte de la mecánica que estudia conjuntamente el movimiento y las fuerzas que lo originan. El modelado *dinámico* de nuestro robot móvil representa un reto con un grado de complejidad relativamente alto debido a sus características físicas, sin embargo, esto no significa que no se puedan desarrollar algunas relaciones interesantes.

Un efecto de las fuerzas que actúan sobre un cuerpo, consiste en modificar su estado de movimiento [Sears67]. El movimiento de un cuerpo puede considerarse compuesto de su movimiento de *traslación*, y de cualquier movimiento de *rotación* que el cuerpo pueda tener. En nuestro caso en particular, este par de movimientos ocurren al momento de trasladar nuestro robot móvil de un punto a otro, sin embargo, existe la posibilidad de que estos mismos aparezcan de manera negativa provocando efectos indeseables, los cuales comentaremos a continuación.

Antes de comenzar con el primer caso de análisis, es necesario hacer una consideración sobre la estructura física de nuestro robot móvil. En la figura 2.22 se muestra una aproximación de la estructura de nuestro robot móvil en forma cilíndrica, aunque estrictamente hablando es obvio que la estructura real, es una estructura irregular y no uniforme con lo que al peso se refiere.

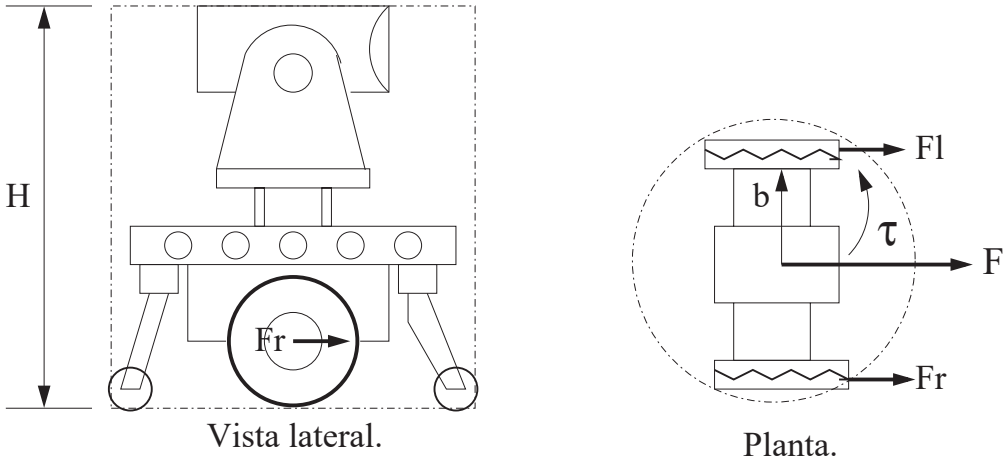


Figura 2.22: Aproximación cilíndrica de la estructura del robot móvil.

Donde:

H: Es la altura del robot móvil.
F: Es la fuerza actuante al centro de masa.
τ: Es el par actuante al centro de masa.
F_r y F_l: Son las fuerzas ejercidas al robot por las llantas derecha e izquierda, respectivamente.
b: Es la distancia del centro de masa a las llantas.

Si M y J_R son la masa y la inercia del robot respectivamente, la segunda ley de Newton implica lo siguiente.

$$F = M\frac{dv_R}{dt} \tag{2.38}$$

y,

$$\tau = J_R\frac{d\omega_R}{dt} \tag{2.39}$$

Donde:

v_R: Es la velocidad lineal del robot.
ω_R: Es la velocidad angular del robot.

Las fuerzas F_r y F_l pueden calcularse de la siguiente forma [Beard02].

$$F_r = \frac{\tau_r}{r} \tag{2.40}$$

Donde:

τ_r: Par actuante en la llanta del robot.
r: Es el radio de la llanta.

Sustituyendo la ecuación (2.39) en esta última ecuación tenemos.

$$F_r = \frac{1}{r} J_r \frac{d\omega_r(t)}{dt} \tag{2.41}$$

Donde:

J_r: Es la inercia vista por la flecha de la llanta derecha.

Ahora bien si sustituimos la ecuación (2.34) obtenida al final de la subsección 2.2.4 obtenemos el siguiente resultado.

$$F_r = \frac{1}{r} J_r(-K_{2r}\omega_r(t) + K_{1r}V_r(t)) \tag{2.42}$$

De manera similar para F_l tenemos.

$$F_l = \frac{1}{r} J_l(-K_{2l}\omega_l(t) + K_{1l}V_l(t)) \tag{2.43}$$

El cálculo de F_r y F_l se puede hacer bajo varias circunstancias, pero una en particular que resulta interesante, es la prueba del escalón con los motores acoplados a la estructura del robot móvil ya que en ella los motores de C.D. son exigidos a su máxima capacidad al inicio de la prueba. Un aspecto que es importante comentar es que la estructura de nuestro robot móvil está sujeta a cambios físicos, ya que al agregar nuevo equipo se alteran algunas de las características de éste, especialmente se espera que el crecimiento del robot sea en forma vertical. Para una aproximación númerica (mostrado a continuación) de algunos de los parámetros del robot móvil se utilizó el estado actual de éste.

Para poder hacer el cálculo de F_r y F_l es necesario a su vez hacer una estimación de las constantes K_{2l}, K_{1l}, K_{2r} y K_{1r}, lo cual se presenta en el apéndice G por no ser el objetivo principal de este trabajo.

En la figura 2.23 se muestra el comportamiento de F_l y F_r (ver ecuaciones 2.42 y 2.43), ante un escalón.

Al inicio cuando la masa del robot se encuentra en reposo y los motores son energizados repentinamente, estos desarrollan el máximo par permisible modificando el estado de reposo de la masa del robot móvil. Dicho de otra forma la masa del robot es acelerada a una tasa de cambio relativamente alta y es en este momento cuando se desarrolla la máxima fuerza aplicada a la estructura del robot móvil. Una vez roto el estado de reposo se necesita una fuerza menor a la inicial para mantenerlo en movimiento (ver fig. 2.23).

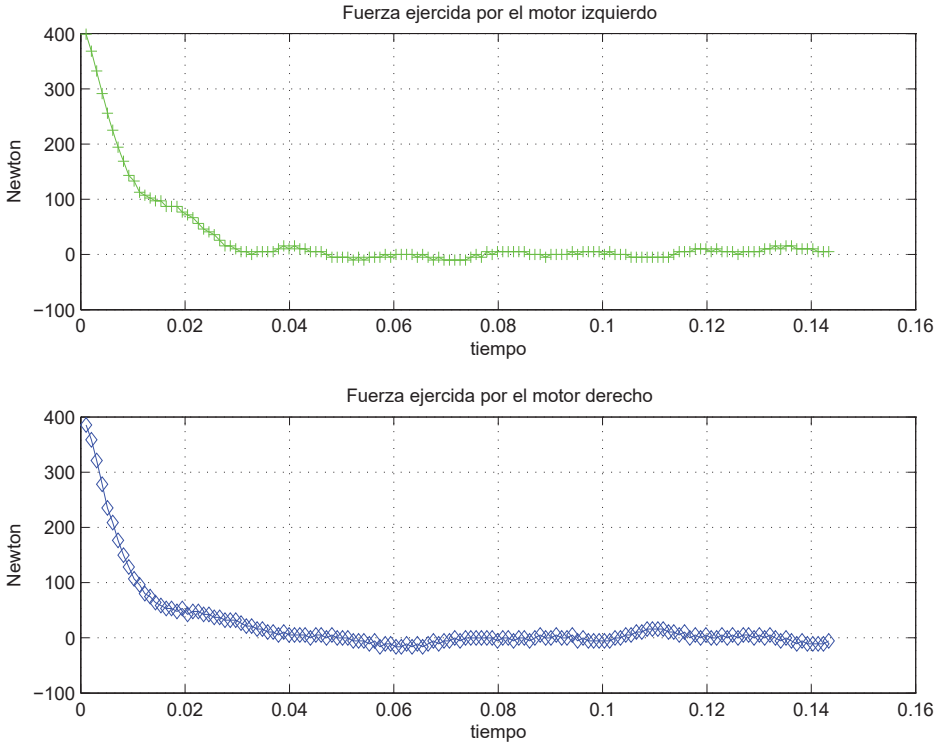


Figura 2.23: Fuerzas estimadas.

Obviamente cuando la estructura del robot móvil es expuesta a una fuerza, en este caso en t cercano a 0, que es el instante en que se tiene la mayor fuerza aplicada en la parte inferior del robot, ésta presenta además de un movimiento de translación un movimiento de rotación con respecto al punto O de la figura 2.24, el cual puede o no presentarse. Hasta el momento las pruebas del escalón realizadas con el robot móvil sólo aportaron un movimiento de traslación a la estructura o masa del robot, más no de rotación, sin embargo dado que la estructura del robot tiende a crecer como se mencionó anteriormente es importante tomar ciertas consideraciones ya que este último podría presentarse y acasionar problemas de estabilidad en el arranque.

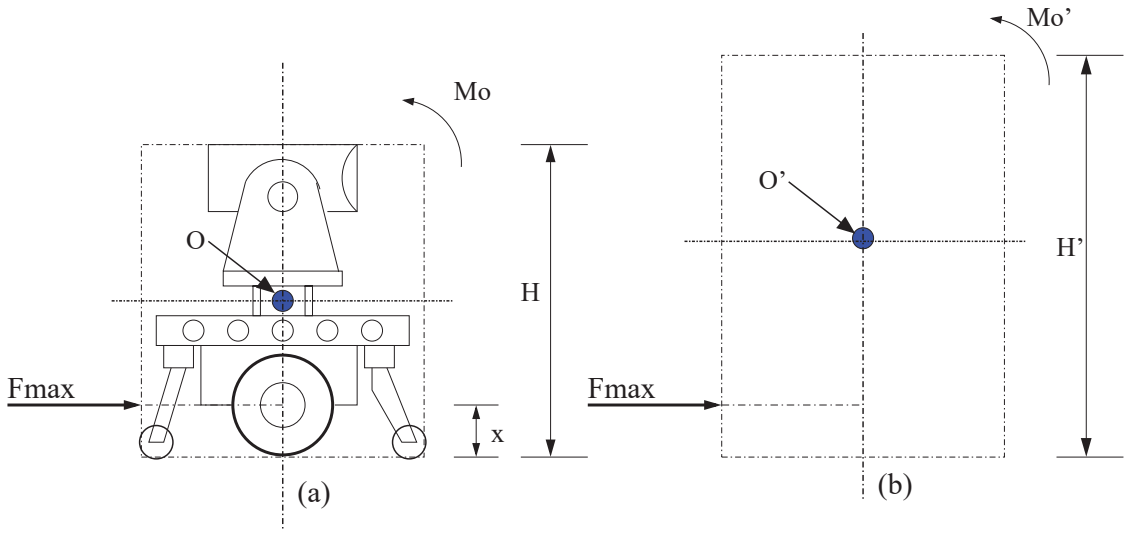


Figura 2.24: Comportamiento indeseable.

Recordando la definición del momento de una fuerza, descrita por la siguiente ecuación.

$$M_o = F * l \tag{2.44}$$

Donde:

M_o: momento.
l: es el brazo de momento.
F: es la fuerza aplicada.

Si consideramos el momento respecto al punto O como se muestra en la figura 2.24 el brazo de momento l de la fuerza F_{max} para este caso será $\frac{H}{2} - x$. Es notorio que a medida que l aumenta así mismo lo hace M, aumentando con esto la posibilidad de romper con la condición de equilibrio rotacional.

Siendo prácticos y basándonos más en la observación y sentido común, podemos decir que es muy improbable que se presente un giro de la estructura del robot móvil en el

sentido mostrado en la figura 2.25. Esto debido a la configuración física del robot móvil.

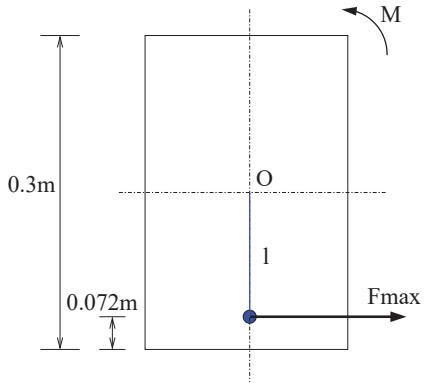


Figura 2.25: Medidas de la estructura del robot móvil.

Dado que el equilibrio desde el punto de vista giratorio se mantiene, es preciso decir que debe existir una fuerza capaz de producir un momento opuesto al aplicado. Esta fuerza proviene principalmente de la superficie sobre la que se encuentra el robot móvil como se observa en la figura 2.26.

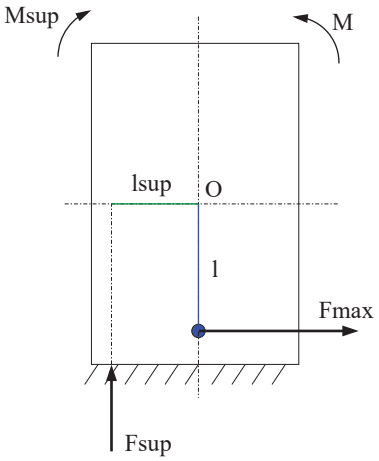


Figura 2.26: Suma de momentos igual a cero (Equilibrio).

De esta manera el momento resultante es nulo, es decir $\Sigma M_T = M_o + M_{sup} = 0$.

2.3. Control de Velocidad de los Motores de C.D.

2.3.1. Introducción.

El diagrama de bloques de la figura 2.27 muestra un lazo de control clásico, en donde el sistema tiene dos componentes principales, la planta en este caso el motor de C.D. y el controlador. El motor tiene una entrada V_a (voltaje de armadura) llamada variable de control y una salida ω (velocidad angular) llamada variable del proceso, esta última es medida por un sensor (encoder) y retroalimentada al punto de suma, donde es substraida del valor de referencia ω_{sp}. La variable ω_{sp} es el valor deseado de velocidad al cual se quiere que trabaje la planta. La diferencia entre estas variables es llamada error actuante (e) el cual es la entrada del controlador. El propósito de este esquema es mantener la variable del proceso ω cerca de la referencia deseada, incluso en presencia de perturbaciones.

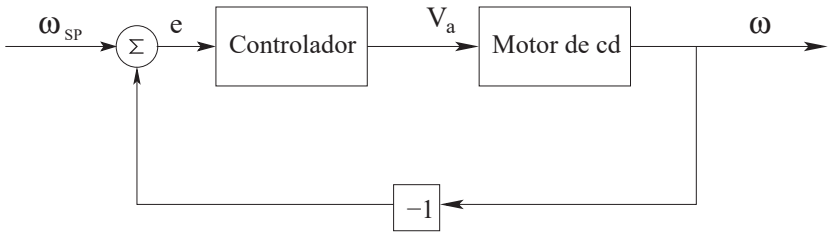


Figura 2.27: Diagrama de bloques del sistema de control en lazo cerrado.

2.3.2. El controlador utilizado.

Antes de iniciar con la descripción de tipo de controlador usado en el robot móvil es importante comentar que la decisión de utilizar un control discreto o digital en lugar de un control analógico, está basada en el tipo de dispositivos con los que cuenta el robot, ya que en su totalidad son digitales. El controlador PID es por mucho el algoritmo de control más común en los procesos de control [Astrom95]. Este es implementado en diferentes formas y versiones. Puede ser implementado de forma aislada o como parte de un proceso de control. En este trabajo se usó la versión corta por así llamarle, del controlador PID; es decir, únicamente la acción proporcional e integral, ya que con ella se obtuvieron buenos resultados, los cuales son detallados a continuación.

La ecuación (2.45) es la versión continua de un controlador PI. Cuando se usa una computadora digital en un proceso de control es necesario discretizar la ecuación (2.45). La parte proporcional no representa mayor esfuerzo a la hora de implementarse. Ésta se implementa simplemente usando los valores muestreados. La ecuación (2.46) muestra tal relación.

$$u(t) = K(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau) \tag{2.45}$$

Donde.

K es la ganancia proporcional.
T_i es la constante denominada de tiempo integral.

Por otro lado para la implementación de la parte integral existen varias formas de aproximar este término. En nuestro caso usamos la aproximación de la integral por rectángulos. La ecuación (2.47) muestra esta aproximación.

$$P(t_k) = K(y_{sp}(t_k) - y(t_k)) = Ke(t_k) \tag{2.46}$$

$$I(t_k) = I(t_{k-1}) + \frac{Kh}{T_i}e(t_k) \tag{2.47}$$

Donde.

h es el periodo de muestreo.

La función principal de la parte integral es asegurarse que la salida en estado estable del proceso coincida con la referencia. La suma de ambos términos forma la ecuación de diferencias del controlador PI. Esta versión del controlador es llamada por algunos autores como forma posicional [Astrom95]. La ecuación (2.48) describe tal relación.

$$u(t_k) = P(t_k) + I(t_k) = Ke(t_k) + I(t_{k-1}) + \frac{Kh}{T_i}e(t_k) \tag{2.48}$$

Donde por comodidad haremos la siguiente dedfinición:

$K_I = \frac{Kh}{Ti}$.

Mientras que K se mantiene sin cambios. Reescribiendo (2.48) tenemos.

$$u(t_k) = Ke(t_k) + I(t_{k-1}) + K_I e(t_k) \tag{2.49}$$

Un aspecto importante que debe ser tomado en cuenta es la saturación, ya que provoca un desempeño inadecuado de nuestro controlador. La saturación se presenta en los actuadores cuando la acción de control sobrepasa sus límites. La forma de eliminar este problema consiste en implementar una acción que evite que la parte integral continue sumando una vez presentada esta situación. Esta acción es comúnmente llamada *anti-windup* [Astrom90].

El rango de operación de los actuadores o específicamente los motores de C.D. es de 0 a 24 v, el cual es controlado a través del puente H y del microcontrolador. El microcontrolador $68HCS12$ cuenta con un módulo denominado PWM_8B8C [Mot02c], usado en el modo de 8 bits de resolución, por tanto tenemos que 0 Volt $\rightarrow$ 0x00 y 24 Volt $\rightarrow$ 0xFF.

En la siguiente subsección describiremos los detalles de la implementación digital, en la cual se toma en cuenta el efecto de la saturación.

2.3.3. Implementación digital del controlador utilizado.

Para la implementación digital de un controlador es necesario seguir los siguientes pasos:

1. Esperar la interrupción del reloj.
2. Leer la señal proveniente del sensor.
3. Calcular la acción de control.
4. Enviar señal de control.
5. Ir a 1.

A continuación se presenta una sección del código implementado en el microcontrolador, el cual tiene como objetivo calcular la acción de control para ambos motores. Para poder tomar encuenta la saturación y así omitir la acumulación en la parte integral, es necesario realizar un primer cálculo de la acción de control. Una vez hecho esto, ya es posible preguntar si la acción de control superó los limites de saturación. Esto se puede observar en las sentencias $if > 255$ y $if < 0$, donde *left_error* y *right_error* son las acciones de control para los motores izquierdo y derecho. Una vez realizado el cálculo de la parte integral y la parte proporcional son sumadas y asignadas a las variables antes descritas. Por último las funciones *alter_left_power* y *alter_right_power* son las encargadas de modificar la potencia aplicada a los motores a través del modulo PWM de microcontrolador y el puente H. En el apéndice C se muestra el resto del código implementado.

```
//-------------------------------------------------------------------------
//Controlador PI en su version clasica  U(tk)=P(tk)+I(tk) donde:
//P(tk)=k( Wref(tk) - W(tk) )
//I(tk+1)=I(tk) + kh/Ti(Wref(tk) - W(tk) )
//************************** Termino Integral **********************************
if(bandera==1){
     if(left_error > 255){      //Saturación limite superior
     }
     else if(left_error < 0){   //Saturación limite inferior
     }
     else{                      //Operacion normal del actuador
          integral_left +=  (k_integral*(des_left_vel_clicks - left_vel))/den;
     }
     if(right_error > 255){     //Saturación limite superior
     }
     else if(right_error < 0){  //Saturación limite inferior
     }
     else{                      //Operación normal del actuador
```

```
          integral_right += (k_integral*(des_right_vel_clicks - right_vel))/den;
     }
}
else{
     integral_left +=  (k_integral*(des_left_vel_clicks - left_vel))/den;
     integral_right += (k_integral*(des_right_vel_clicks - right_vel))/den;
     bandera=1;
}
//***********************Termino Proporcional *********************************
     prop_left =( (k_pro*((des_left_vel_clicks) - left_vel))/den );
     prop_right =( (k_pro*((des_right_vel_clicks) - right_vel))/den );

//*****************-Cálculo de la acción de control-****************************

     left_error  = (prop_left) + (integral_left);
     right_error = (prop_right) + (integral_right);

     alter_left_power (left_error );
     alter_right_power(right_error);
//------------------------------------------------------------------------------
```

Código 2.2 Implementación del controlador Proporcional Integral.

Sintonización del controlador: Cuando se trata de diseñar un controlador es necesario tener en cuenta cuales son los objetivos de control. Dos tipos de objetivos comunes son: el seguimiento de la referencia y el rechazo a las perturbaciones, los cuales fueron los principales a seguir en este trabajo.

El controlador PI propuesto como estrategia de control es descrito por dos parámetros, la ganancia proporcional K y la constante integral K_I, estos dos parámetros pueden ser sintonizados analíticamente para lograr el comportamiento deseado. El uso de un método análitico para el diseño del controlador, es en muchas ocasiones un punto de partida para la consecusión de los objetivos propuestos, sin embargo en la práctica es muy difícil encontrarse con el caso en que las constantes aproximadas por el método analítico logren el desempeño deseado, y en la mayoría de los casos se obtienen los últimos ajustes usando el método de prueba y error.

La sintonización de las constantes del controlador PI implementado en nuestro sistema se hizo por prueba y error, y la justificación de esto aparte de la mencionada en el

párrafo anterior, se basa en los resultados obtenidos en la subsección de experimentación 2.2.4, ya que en ellas se observaron dinámicas no incluidas en el modelo matemático construido.

Otro aspecto por el cual se tomó esta decisión es la aritmética de punto fijo del microcontrolador, la cual nos limita desde el punto de vista númerico.

De acuerdo con lo establecido en los parrafos anteriores, para la sintonización de las constantes del controlador PI se construyó una plataforma en *LabVIEW 7.0* [Bishop04] la cual nos permitió modificar las constantes y observar los resultados debidos a estos cambios. En el apéndice A se describen los detalles funcionales de esta plataforma.

La mejor respuesta tomando en cuenta que los objetivos primordiales son el rechazo a las perturbaciones y el seguimiento de la referencia, se obtuvieron con $K_P = 1.4$ y $K_I = 1.8$.
En nuestro caso ambos objetivos son importantes; sin embargo, el seguimiento de la referencia juega un papel muy importante en el seguimento de trayectorias (Ver cápitulo 4).

En la figura 2.28 se muestran las curvas de velocidad, el error y de la acción de control para ambos motores, las cuales describimos a continuación. El comportamiento de la velocidad para ambos casos comienza con oscilaciones pronunciadas (ver fig. 2.28 (e) y (f)), sin embargo termina fijándose a la referencia de velocidad con un grado de error aceptable. Como recién se comentó en los párrafos anteriores, el seguimiento de la referencia es un aspecto de vital importancia en el seguimiento de las trayectorias, ya que una trayectoria es en esencia un conjunto de referencias de velocidad. De esta manera, entre mejor sea la ejecución o apego a las referencias de velocidad, menor sera el error descrito por el robot móvil al seguir una trayectoria. En el capítulo 4 describiremos en detalle este aspecto. Cabe mencionar que las oscilaciones iniciales (aún considerables) se deben a que la referencia es un escalón, el cual tiene un cambio brusco inicial que acentúa el efecto oscilatorio de la dinámica no modelada. Este efecto es posible disminuirlo implementando una rampa de arranque la cual describiremos en el capítulo 3. El comportamiento de los errores, como se esperaba, comienzan con valores cercanos a sus respectivas referencias, éste se mantiene oscilando sobre el cero unos instantes y finalmente se estabilizan en un valor cercano a

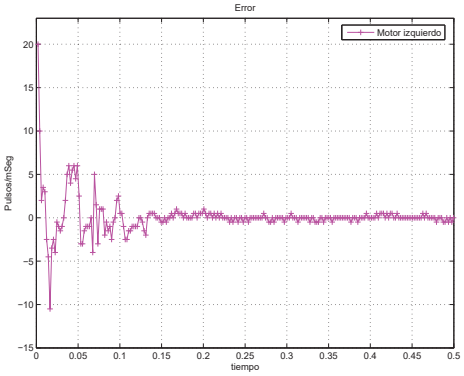
cero. En las pruebas mostradas, la referencia de velocidad para el motor izquierdo fue de 20 Pulsos/ms. mientras que para el motor derecho es fue de 15 Pulsos/ms.

2.4. Resumen

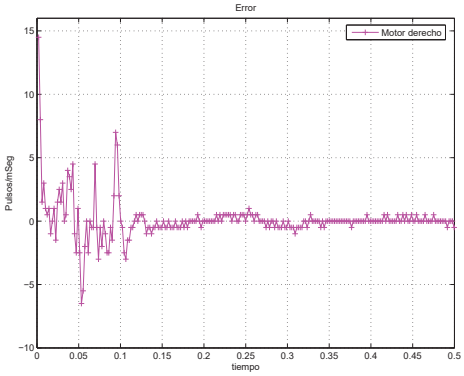
En esta sección realizaremos un resumen de la información generada a lo largo de este capítulo. Comenzando con la sección 2.1, en la cual se estableció el tipo de arquitectura y configuración del robot, asi como el *hardware* que lo comforma.

En la sección 2.2 referente al modelado del robot móvil, se obtuvieron diferentes tipos de modelos matemáticos referentes a un robot de locomoción diferencial dotado con servomotores de CD en las llantas de tración. El primer modelo desarrollado fue el modelo cinemático, con el cual fue posible conocer las variables de control que lo gobiernan asi como los tipos de movimientos y posiciones alcanzables por una estructura con estas caracteristicas. El siguiente modelo desarrollado fue el modelo del motor de CD. Este modelo es de vital importancia, ya que éste tiene relación directa con algunas de las varibles de control que intervienen en el modedlo cinemático. Estas variables son, la velocidad lineal y la velocidad angular de la estructura del robot. El modelo del motor de CD, además, sirvió para la realización de varios experimentos (ver *respuesta del motor de CD en vacío* y *respuesta al escalón del sistema*) los cuales sirvieron para la identificación de las dinámicas que gobiernan el robot móvil visto como sitema. Finalmente en esta misma sección se desarrolló un modelo dinámico, en el que se muestra de forma estimada las fuerza actuantes en la estructura del robot móvil al momento en que es expuesta a un arranque subito. Mismo que se obtuvo aprovechando la información generada en el experimento del escalón.

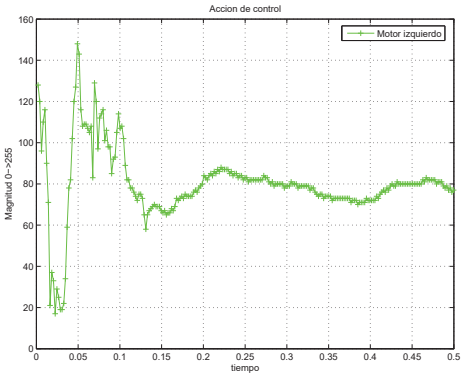
En la última sección 2.3 Control de velocidad de los motores de CD, se describió de manera detallada el algoritmo de control implementado. Mismo que se realizó con la finalidad de eliminar los comportamientos no deseables debidos a la dinámica propia del sistema asi como para tener un control preciso sobre la velocidad angular y lineal de la estructura del robot móvil.



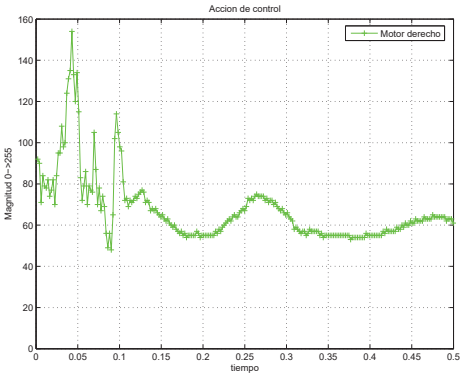
(a) ***Error Motor Izquierdo.***



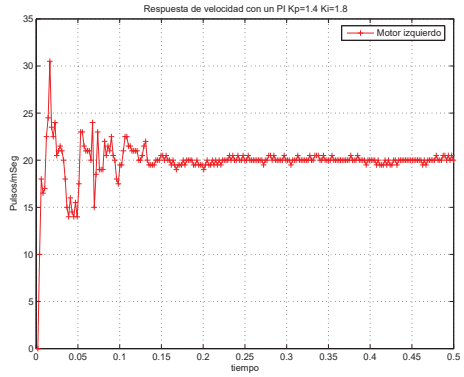
(b) ***Error Motor Derecho.***



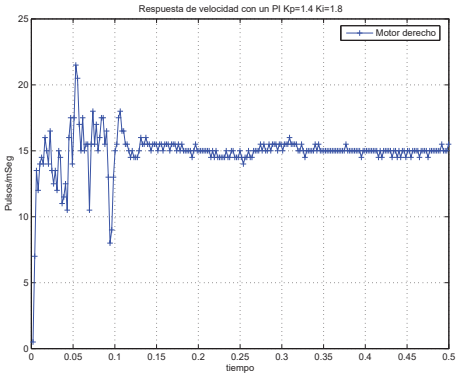
(c) ***Acción de control Motor Izquierdo (PWM).***



(d) ***Acción de control Motor Derecho (PWM).***



(e) ***Respuesta Motor Izquierdo.***



(f) ***Respuesta Motor Derecho.***

Figura 2.28: Comportamiento del Error, Acción de control y Velocidad de ambos motores.

Capítulo 3

GENERACIÓN DE TRAYECTORIAS BASADA EN SPLINES

En el primer capítulo se comentó que el objetivo de esta tesis tiene que ver con el seguimiento de caminos o trayectorias, por lo que la generación adecuada de las trayectorias a seguir es un paso fundamental para lograr este objetivo. La generación de una trayectoria comienza estableciendo una serie de puntos por donde se quiere que pase el vehículo.

El criterio de selección de estos puntos tiene relación directa con dos aspectos. El primero de ellos tiene que ver con las características físicas del entorno donde navegará el robot móvil, mientras que el segundo aspecto tine que ver con las limitaciones dinámicas y cinemáticas del robot.

En este capítulo se presenta un método de interpolación de los puntos de control basado en la teoría de los *Splines*, la cual se describe a continuación.

3.1. Conceptos Básicos

3.1.1. Curvas paramétricas

Generalmente la definición de una curva se hace a través de una función contínua de $\Re$ en $\Re^3$ o equivalentemente, mediante tres funciones escalares que expresan independi-

entemente cada una de las tres coordenadas de los puntos de la curva [Cordero02]:

$$x = f_1(t)$$
$$y = f_2(t)$$
$$z = f_3(t)$$

Donde t es el parámetro real tal que $t \in [a, b]$.

Aunque intuitiva, la representación paramétrica no está exenta de problemas. El más notable es la no unicidad de la representación. Esto significa que una misma curva puede ser definida por distintas funciones de $\Re$ en $\Re^3$ y el ejemplo más sencillo lo encontramos en la expresión paramétrica de una línea recta:

$$\begin{aligned} &F : (-\infty, \infty) \to \Re^3 \\ &t \to P_0 + \varphi(t)(P_1 - P_0) \end{aligned} \tag{3.1}$$

donde $P_0, P_1 \in \Re^3$ son puntos fijos y $\varphi(t)$ es cualquier función continua $\varphi : \Re \to \Re$, estrictamente creciente y verificando $lim_{t\to-\infty}\varphi(t) = -\infty$, $lim_{t\to\infty}\varphi(t) = \infty$. Generalmente, se supone que $\varphi(0) = 0$ y $\varphi(1) = 1$, de modo que $F(0) = P_0$ y $F(1) = P_1$ de esta manera para $t \in [0, 1]$ se obtienen los puntos del segmento comprendido entre P_0 y P_1.

Por ejemplo, tomando $\varphi(t) = t$ y $\varphi(t) = t^3$ tenemos dos representaciones esencialmente distintas para la misma recta:

$$F_1 \ : \ t \to P_0 + t(P_1 - P_0) \tag{3.2}$$

$$F_2 \ : \ t \to P_0 + t^3(P_1 - P_0) \tag{3.3}$$

$$t \in (-\infty, \infty)$$

En la figura 3.1 se muestra se muestra el segmento de recta para $t \in [0, 1]$ para ambas ecuaciones.

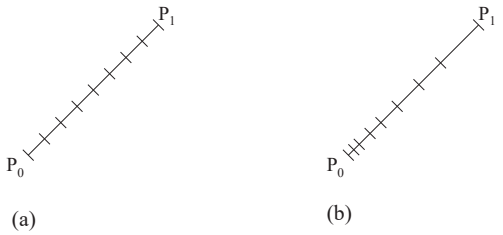


Figura 3.1: Representación paramétrica.

Mientras que las curvas son geométricamente idénticas, no ocurre así con la distribución de sus puntos en función de t. Para un incremento constante de t ($\triangle t = 0,1$), en (a) los puntos de la recta están uniformemente distribuidos, lo que no ocurre en (b).

Por tanto, podemos pensar que cuando dos funciones de $\Re$ en $\Re^3$ describen la misma curva, lo que se está indicando son dos formas diferentes de recorrer una misma entidad geométrica.

3.1.2. Vector tangente (Velocidad)

Si la función $F : [a, b] \to \Re^3$ es diferenciable para cada $t \in [a, b]$ entonces se define el vector tangente:

$$\begin{aligned} &T : [a, b] \to \Re^3 \\ &t \to [f_1'(t), f_2'(t), f_3'(t)] \end{aligned} \tag{3.4}$$

Observemos que, aunque T puede interpretarse a su vez como una curva paramétrica, en la práctica es más conveniente considerarla como una función vectorial de variable real.

Si diferentes funciones F definen una misma curva paramétrica , el vector tangente correspondiente al mismo punto de $\Re^3$ tiene siempre idéntica dirección, siempre y cuando las funciones F sean diferenciables y el vector tangente no sea nulo. Aquí es importante observar que la afirmación se refiere no a iguales valores del parámetro t en dos representaciones distintas, sino a valores distintos t_0 y t_1 en cada una de las respectivas ecuaciones, correspondientes al mismo punto de $\Re^3$.

Para una recta esto es bastante evidente, pues cualquiera que sea la función (diferenciable) $\varphi(t)$ en 3.1, se tiene:

$$T : t \to \varphi'(t)(P_1 - P_0)$$

y la dirección del vector T(t) es siempre la del vector $P_1 - P_0$.

Puesto que la derivada de una función representa la variación infinitesimal de la misma, cuando el parámetro t representa el tiempo, el vector tangente suele denominarse vector *velocidad*, indicando entonces el cambio de un punto que se mueve a lo largo de la curva.

3.1.3. Continuidad

Consideremos la curva definida por las ecuaciones paramétricas (ver figura 3.2).

$$F(t) = \left\{ \begin{array}{lll} (t,0,0) & si & t \in [-1,0] \\ (2t,4t^2,0) & si & t \in (0,\frac{1}{2}] \end{array} \right\} \tag{3.5}$$

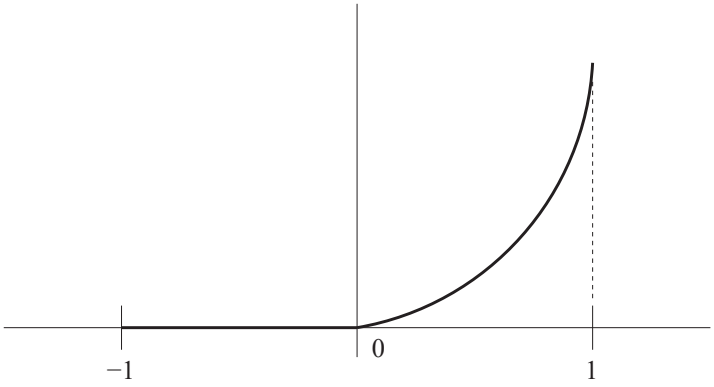


Figura 3.2: Continuidad geométrica.

F(t) es continua en todo su dominio y derivable en cada uno de sus dos tramos, pero con una discontinuidad de la primera derivada en t=0.

Veámoslo con el vector tangente correspondiente:

$$V(t) = \left\{ \begin{array}{lll} (1,0,0) & si & t \in [-1,0] \\ (2,8t,0) & si & t \in (0,\frac{1}{2}] \end{array} \right\} \tag{3.6}$$

y comprobamos que efectivamente, $V(t)$ es discontinuo para t=0:

$$V(0^-) = [1,0,0]$$
$$V(0^+) = [2,0,0]$$

Por lo que la curva es de clase C^0 (continua) en [-1,1]; pero no es de clase C^1 (derivada continua).

Observemos, sin embargo, que la dirección del vector tangente sí es una función continua. Para ser más precisos, podemos definir la dirección de un vector no nulo como el vector dividido por su norma, que en este caso sería:

$$T(t) = \frac{V(t)}{\|V(t)\|} = \left\{ \begin{array}{lll} (1,0,0) & si & t \in [-1,0] \\ \frac{1}{\sqrt{1+16t^2}}(1,4t,0) & si & t \in (0,\frac{1}{2}] \end{array} \right\} \tag{3.7}$$

Y ahora tenemos:

$$T(0^-) = [1,0,0]$$
$$T(0^+) = [1,0,0]$$

lo que confirma que T(t) es continua.

Las funciones como 3.7, para las cuales la dirección del vector tangente, es decir T(t), es continua se dice que poseen *continuidad visual de clase* G^1.

Si no permitimos curvas con vector tangente nulo, entonces toda curva de clase C^1 (continuidad de la derivada primera o equivalente, del vector tangente), es asimismo de

clase G^1. Evidentemente, lo contrario no es cierto, como ha puesto de manifiesto el ejemplo anterior.

Observemos que si en 3.7 efectuamos un cambio de parámetro:

$$S = \varphi(t) = \left\{ \begin{array}{cccc} t & si & t \in [-1,0] \\ 2t & si & t \in (0,\frac{1}{2}] \end{array} \right\} \tag{3.8}$$

obtenemos la misma curva, aunque parametrizada de otra forma:

$$G(s) = G(\varphi(t)) = F(t) = \left\{ \begin{array}{cccc} (s,0,0) & si & s \in [-1,0] \\ (s,s^2,0) & si & s \in (0,1] \end{array} \right\} \tag{3.9}$$

El vector tangente para esta parametrización es:

$$W(s) = \left\{ \begin{array}{cccc} (1,0,0) & si & s \in [-1,0] \\ (1,2s,0) & si & s \in (0,1] \end{array} \right\} \tag{3.10}$$

y ahora tenemos:

$$W(0^-) = W(0^+) = [1,0,0]$$

con lo cual G(s) tiene derivada primera continua y por consecuencia es de clase C^1 (y también, desde luego, de clase G^1).

Vemos entonces que el concepto de continuidad de la derivada está fuertemente ligado a la parametrización de la curva, por lo que no se trata de una propiedad intrínseca de la geometría de la curva. En general, muchos textos adoptan como definición de continuidad de clase G^k el que exista una reparametrización de clase C^k.

3.1.4. Curvas de Bézier

La idea principal en una curva de Bézier está en considerar una base especial del espacio del polinomios que permite una excelente interpretación geométrica de los coeficientes de un polinomio cuando son expresados en función de dicha base. Ésta es la denominada *Base de Bernstein* [Cordero02], que fue descrita en 1912 por S. Bernstein, quien la empleó en el contexto de la aproximación uniforme a funciones continuas.

Los polinomios de Bernstein son descritos por la siguiente expresión.

$$B_k(x) = \begin{pmatrix} n \\ k \end{pmatrix} x^k (1-x)^{n-k} \; con \; 0 \leq k \leq n \tag{3.11}$$

donde $(\frac{n}{k}) = \frac{n!}{k!(n-k)!}$ son los coeficientes binomiales.

La expresión 3.11 define una familia de polinomios denominada polinomios de Bernstein y generalmente sólo se define para $x \in [0,1]$, que es donde manifiesta sus propiedades.

Algunos ejemplos son:

- Para $n = 1$:
  $B_0(x) = 1 - x$
  $B_1(x) = x$

- Para $n = 2$:
  $B_0(x) = (1-x)^2$
  $B_1(x) = 2x(1-x)$
  $B_2(x) = x^2$

- Para $n = 3$:

  $B_0(x) = (1-x)^3$

  $B_1(x) = 3x(1-x)^2$

  $B_2(x) = 3x^2(1-x)$

  $B_3(x) = x^3$

Cuando un polinomio arbitrario de grado menor o igual que n se expresa como combinación lineal de los polinomios de Bernstein, suele denominarse *Curva de Bézier* simple o de un tramo.

$$C(t) = \sum_{k=0}^{n} P_k B_k(t) \;\; 0 \leq t \leq 1 \tag{3.12}$$

donde $P_k \equiv (x_k, y_k)$ si la curva es de $\Re^2$ y $P_k \equiv (x_k, y_k, z_k)$ si la curva es de $\Re^3$.

Estos coeficientes P_k respecto a la base de Bernstein son denominados *puntos de control* de la curva de Bézier.

Asi por ejemplo:

Para $n = 1$

$$C(t) = \sum_{k=0}^{n} P_k B_k(t) = P_0 B_0(t) + P_1 B_1(t) = (1-t)P_0 + tP_1 \tag{3.13}$$

Se tiene que:

$$C(0) = P_0$$
$$C(1) = P_1$$

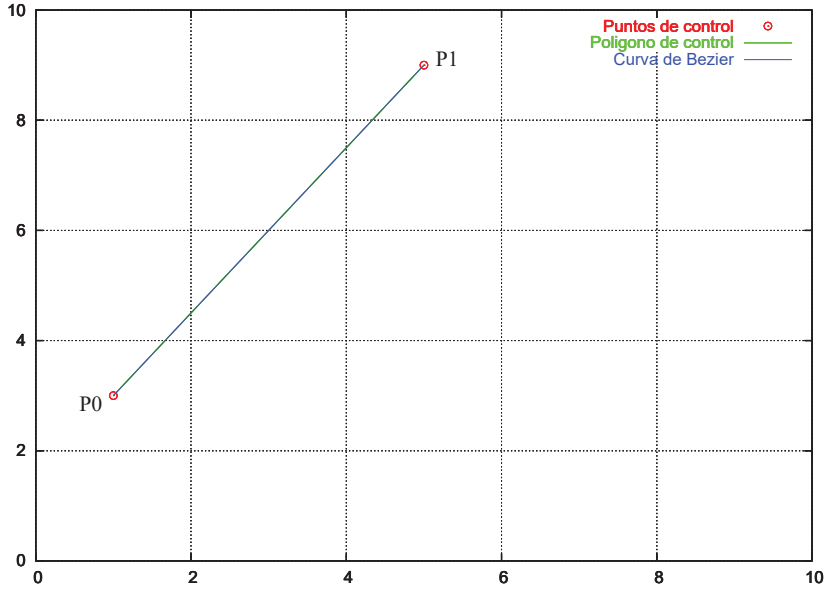


Figura 3.3: Curva de Bézier con n=1 .

Por tanto C(t) es la recta que une los puntos del plano representados por los coeficientes P_0 y P_1 y que para este caso coincide con el poligono de control haciendo el efecto de sobreposición como se observa en la figura 3.3:

Para $n = 2$

$$C(t) = P_0 B_0(t) + P_1 B_1(t) + P_2 B_2(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2 \tag{3.14}$$

Análogamente, al caso $n = 1$ tenemos:

$$C(0) = P_0$$
$$C(1) = P_2$$

con lo cual interpola al primer y tercer puntos de control (ver figura 3.4).

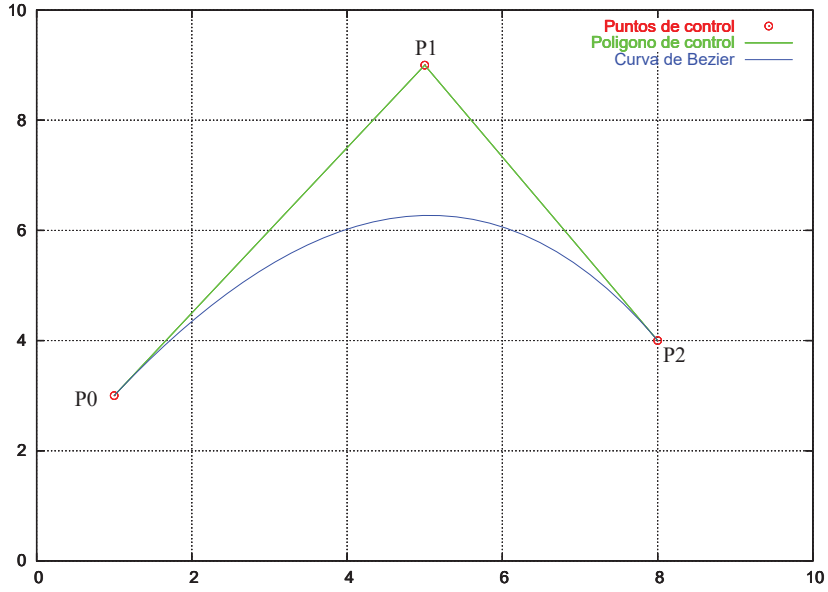


Figura 3.4: Curva de Bézier con n=2 .

Es importante comentar que C(t) posee otra propiedad geométrica. Derivando 3.14 tenemos lo siguiente:

$$C'(t) = -2(1-t)P_0 + 2(1-2t)P_1 + 2tP_2 \tag{3.15}$$

donde:

$$C'(0) = 2(P_1 - P_0)$$
$$C'(1) = 2(P_2 - P_1)$$

$C'(0), C'(1)$ son respectivamente vectores tangentes a la curva C(t) en 0 y en 1 y poseen la dirección de los vectores $P_1 - P_0$ y $P_2 - P_1$ y por tanto, la curva C(t) es tangente al poligono que une P_0, P_1 y P_2 llamada polígono de control (Ver figura 3.4).

A continuación citaremos algunas de la propiedades más importantes de las curvas de Bézier simples:

1. Interpolación de los puntos de control extremos.

$$C(0) = P_0$$
$$C(1) = P_n$$

2. Tangencia al polígono de control en sus extremos.

$$C'(0) = n(P_1 - P_0)$$
$$C'(1) = n(P_n - P_{n-1})$$

3. Control seudolocal.
   Si dos curvas de Bézier C(t), $\overline{C}(t)$ tienen los mismos puntos de control excepto uno, digamos el k-ésimo, entonces:

$$C(t) - \overline{C}(t) = B_k(t)(P_k - \overline{P}_k)$$

La teoría hasta aquí descrita, representó una base de conocimiento necesario para un buen entendimiento de lo que presentaremos en la siguiente sección. Es importante comentar que la teoría descrita en las secciones anteriores, presentan los aspectos más relevantes de estos temas, sin embargo estos pueden encontrarse con mayor detalle en [Cordero02].

3.2. La Curva Tipo Spline de Catmull Rom

La curva de Bézier descrita en la sección anterior interpola unicamente el primero y último de los vértices de control, tal curva es llamada aproximadora [Barsky90], ya que ésta pasa cerca de sus puntos o vértices de control. La curva tipo *Spline* de $Catmull - Rom$ [Rom74] tiene la ventaja de que puede aproximar o interpolar los vértices o puntos de control, tiene control local y es fácil de calcular. La curva tipo *Spline* de $Catmull - Rom$ posee

parámetros de forma (β), los cuales pueden ser usados para modificar la suavidad y forma de la curva. Los parametros de forma pueden aplicarse globalmente para afectar la curva entera o pueden ser modificados localmente afectando solo una porción, cerca del punto de unión correspondiente.

La curva tipo *Spline* cúbico de $Catmull-Rom$ es una curva interpoladora de clase C^1 donde el i-ésimo segmento de la curva se escribe de la forma:

$$q_i(u) = \sum_{j=-1}^{2} V_{i+j}\varphi_j(u) \; u \in [0,1] \tag{3.16}$$

Las funciones peso $\varphi_j(u)$, con $j = -1, 0, 1, 2$ son construidas de manera que q_i y q_{i+1} se juntan en el punto de unión con continuidad de tipo G^1. Las funciones peso son descritas como sigue:

$$\varphi_{-1}(u) = -\beta_1^2 \frac{u^3-2u^2+u}{\beta_1+1} \tag{3.17}$$

$$\varphi_0(u) = \frac{(\beta_1^2+\beta_1+1)u^3-(2\beta_1^2+2\beta_1+1)u^2+(\beta_1^2-1)u+\beta_1+1}{\beta_1+1} \tag{3.18}$$

$$\varphi_1(u) = -\frac{(\beta_1^2+\beta_1+1)u^3-(2\beta_1^2+\beta-1+1)u^2-\beta_1 u}{\beta_1(\beta_1+1)} \tag{3.19}$$

$$\varphi_2(u) = \frac{u^3-u^2}{\beta_1(\beta_1+1)} \tag{3.20}$$

El efecto del parametro de forma β_1 se muestra en la figura 3.5 y 3.6. Para la generación de nuestras trayectorias de prueba se usó el parámetro de forma $\beta_1 = 1$, ya que este valor

genera trayectorias sin muchas oscilaciones, lo cual es conveniente para la navegación del robot móvil. Por otro lado con $\beta_1 = 0.5$ se generan trayectorias con oscilaciones pronunciadas (ver figura 3.6) lo cual no es muy conveniente desde el punto de vista de la navegación. En el apéndice B se encuentra implementado en MATLAB dicha curva.

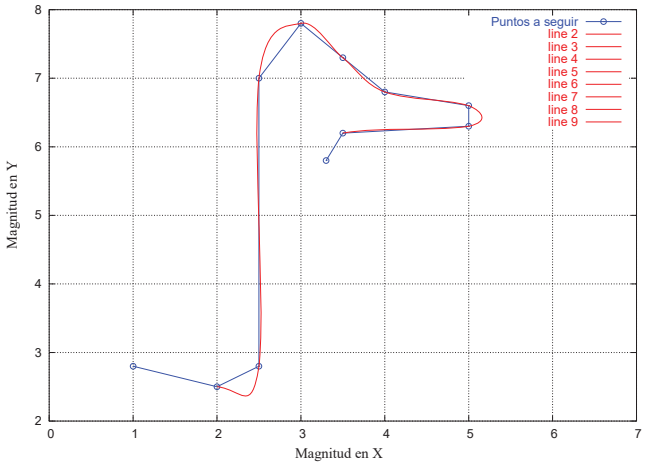


Figura 3.5: Con $\beta 1$=1 .

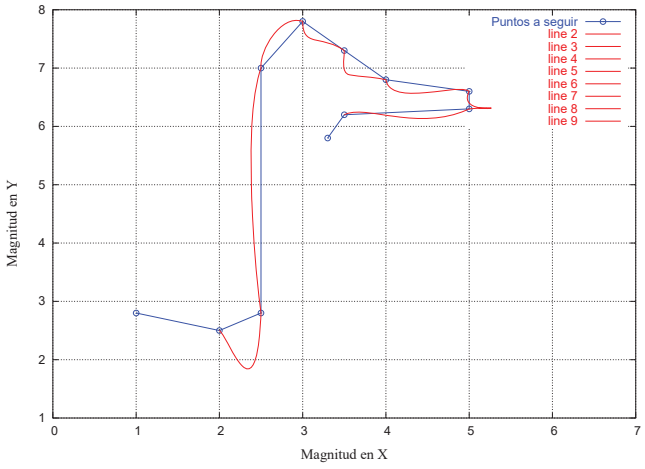


Figura 3.6: Con $\beta 1$=0.5 .

3.3. Trayectorias con la Curva Tipo Spline de Catmull Rom

La generación de las trayectorias de prueba para el robot móvil se basa en la siguiente idea. Dados unos puntos de control por donde se requiere que pase el robot móvil, encontrar una trayectoria con buenas características entre punto y punto que logren llevar al robot móvil desde un punto de inicio hasta un objetivo final. Esta trayectoria se obtiene utilizando la interpolación mediante la curva tipo *Spline* de $Catmull - Rom$.

Antes de comenzar a utilizar la curva tipo *Spline* de $Catmull - Rom$ es importante comentar algunos detalles. El primero de ellos es que éste interpola a partir del segundo punto de control hasta el punto $n-1$ como se muestra en la figura 3.5, y dado que a nosotros nos interesa interpolar todos los puntos de control este detalle debe tomarse en cuenta. La solución a este problema consta en repetir el primero y último de los puntos de control (ver fig. 3.7 y 3.8).

Otro detalle es el siguiente. En la figura 3.5 se muestra como la curva tipo *Spline* de $Catmull - Rom$ une los puntos de control como si fuera de manera continua, esto es debido a la forma de recorrer el intervalo de definición del Spline $u \in [0, 1]$, lo cual se hace en incrementos relativamente pequeños. En realidad la graficación del *Spline* se hace de manera discreta, determinada por el incremento ($\triangle u$) seleccionado. Este aspecto es determinante a la hora de la realización de cálculos, es decir, el uso de incrementos pequeños implica un número mayor de cálculos pero mayor resulución en la graficación y el uso de incrementos grandes implica menor número de cálculos pero menor resolución en la graficación. En nuestro caso para la generación de las trayectorias de prueba es necesario seleccionar cuidadosamenete este incremento, ya que existe un compromiso entre el número de cálculos y la resolución en la graficación. Por ejemplo un menor número de cálculos es un aspecto bueno si se requiere optimizar tiempo, sin embargo esto implica la pérdida de información o características de la trayectoria lo cual es importante también.

En la figura 3.7 se muestra como el uso de $\triangle u = 0.1$ implica el cálculo de diez puntos entre vértices adyacentes, mientras que en 3.8 el uso de $\triangle u = 0.2$ reduce el número de cálculos a la mitad. Este último caso ofrece un buen resultado desde el punto de vista del número de cálculos y sin pérdida de resolución en la trayectoria, por lo que este resultado será usado para la generación de las trayectorias de prueba que usaremos en la siguiente sección.

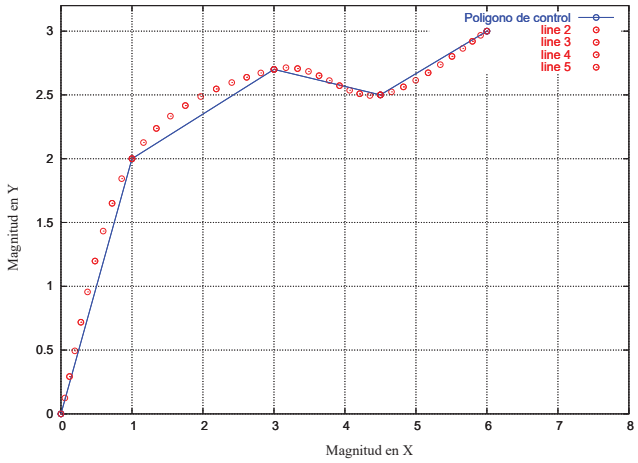


Figura 3.7: Con $\triangle$u=0.1 .

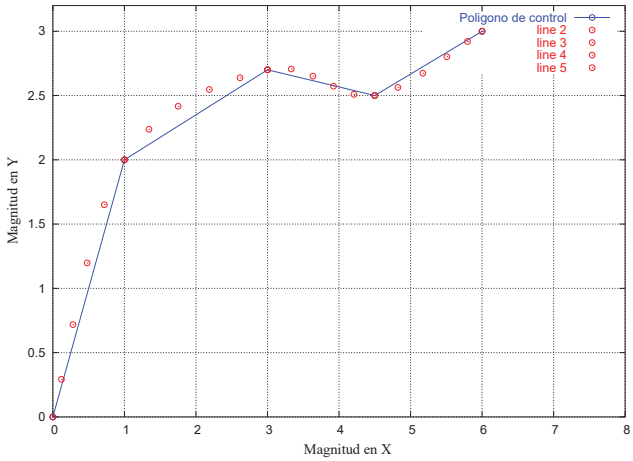


Figura 3.8: Con $\triangle$u=0.2 .

3.4. Seguimiento de Trayectorias

El planteamiento del problema de seguimiento de trayectorias puede hacerse de la siguiente manera: dada una trayectoria o camino con ciertas características, se requiere encontrar una serie de comandos comprensibles por los actuadores del robot móvil que logren que describa dicha trayectoria o camino. Para lograr esto, es necesario sustraer cierta información de la trayectoria, posteriormente se usa dicha información junto con el modelo cinemático del robot móvil descrito en el capítulo 2 para obtener los comandos ejecutables por los actuadores. El diagrama de bloques de la figura 3.9 muestra de manera general lo antes descrito.

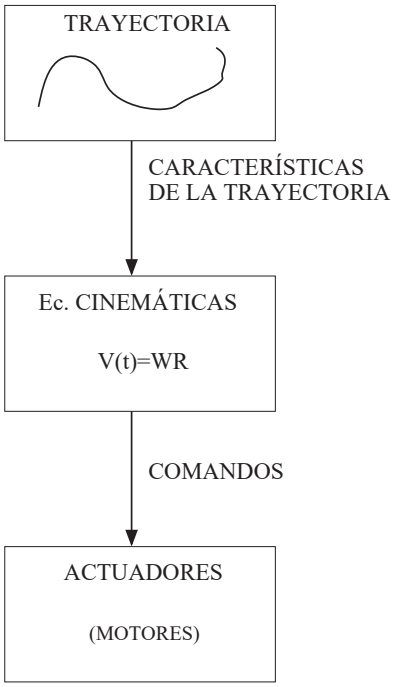


Figura 3.9: Planteamiento general del problema.

3.5. Generación del Vector de Velocidades

Recordando la teoría sobre la cinemática del robot móvil descrita en el capítulo 2, la ecuación (2.5) implica un giro de la estructura del robot móvil con respecto a un punto de rotación (ver fig. 2.9). Este tipo de movimiento corresponde al de una trayectoria en forma de arco perteneciente a un círculo de radio **R**, también llamado radio de curvatura. En el capítulo anterior se describió un método de generación de trayectorias basado en la curva tipo *Spline* de $Catmull - Rom$, en el cual es muy común encontrar trayectorias como las recién comentadas, por tanto la clave para solucionar el problema del seguimiento de trayectorias en forma de arco (cinemática inversa), yace en encontrar el radio de curvatura **R** a partir de la información de la trayectoria que se desea seguir. El cálculo del radio de curvatura a partir de una trayectoria curva se hace de la siguiente manera. Dentro de la geometría analítica [Fuller02] existe una forma de determinar un círculo y por consiguiente el radio de curvatura **R** dadas ciertas condiciones geométricas. Estas condiciones geométricas para nuestro caso son: encontrar la ecuación del círculo que pasa por tres puntos dados (ver fig. 3.10). La ecuación 3.21 es la forma general de la ecuación del círculo, la cual usaremos para abordar este problema.

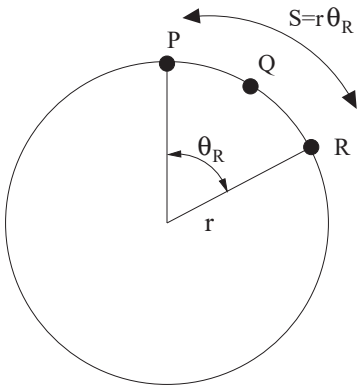


Figura 3.10: Arco de círculo que pasa por tres puntos dados.

$$x^2 + y^2 + Dx + Ey + F = 0 \tag{3.21}$$

Dados los puntos $P(x_p, y_p)$, $Q(x_q, y_q)$ y $R(x_r, y_r)$, tales que no estén sobre una misma recta; se desea encontrar los coeficientes para D, E y F de tal manera que la ecuación (3.21) se satisfaga para las coordenadas de cada uno de los puntos dados. Esto es:

$$\begin{aligned} x_p^2 + y_p^2 + Dx_p + Ey_p + F = 0 \\ x_q^2 + y_q^2 + Dx_q + Ey_q + F = 0 \\ x_r^2 + y_r^2 + Dx_r + Ey_r + F = 0 \end{aligned} \tag{3.22}$$

Reescribiendo y acomodando términos tenemos:

$$\begin{bmatrix} x_p & y_p & 1 \\ x_q & y_q & 1 \\ x_r & y_r & 1 \end{bmatrix} \begin{bmatrix} D \\ E \\ F \end{bmatrix} = \begin{bmatrix} -(x_p^2 + y_p^2) \\ -(x_q^2 + y_q^2) \\ -(x_r^2 + y_r^2) \end{bmatrix} \tag{3.23}$$

Donde la solución a este sistema matricial de 3x3 es como sigue:

$$\begin{bmatrix} D \\ E \\ F \end{bmatrix} = \begin{bmatrix} x_p & y_p & 1 \\ x_q & y_q & 1 \\ x_r & y_r & 1 \end{bmatrix}^{-1} \begin{bmatrix} -(x_p^2 + y_p^2) \\ -(x_q^2 + y_q^2) \\ -(x_r^2 + y_r^2) \end{bmatrix} \tag{3.24}$$

En la ecuación (3.21) se encuentra de manera implícita el radio del círculo, y dado que esta es la variable de interés, es necesario transformar la ecuación general del círculo a su forma centro radio (ver ecuación. 3.25 centro en o(h,k) y radio r), lo cual se hace mediante la ecuación (3.26).

$$(x - h)^2 + (y - k)^2 = r^2 \tag{3.25}$$

$$(x + \tfrac{D}{2})^2 + (y + \tfrac{E}{2})^2 = \tfrac{D^2}{4} + \tfrac{E^2}{4} - F \tag{3.26}$$

La ecuación (3.26) tendrá como gráfica un circulo, si.

$$\frac{D^2}{4} + \frac{E^2}{4} - F > 0$$

Para el caso en que esto no se cumpla, estariamos hablando precisamente del caso en el que los puntos P, Q y R están alineados o que pertencen a una sección de una parábola. Aún que estos casos tienen una muy baja probabilidad de que sucedan sobre todo en trayectorias suaves, se implementó una protección en los códigos que realizan estos cálculos.

Si aplicamos este proceso para cada tres puntos por ejemplo de la trayectoria mostrada en la figura 3.11 , obtenemos un vector con los radios de curvatura $\mathbf{R}$ tales que describen los segmentos de arco que pasan por cada tres puntos de la trayectoria.

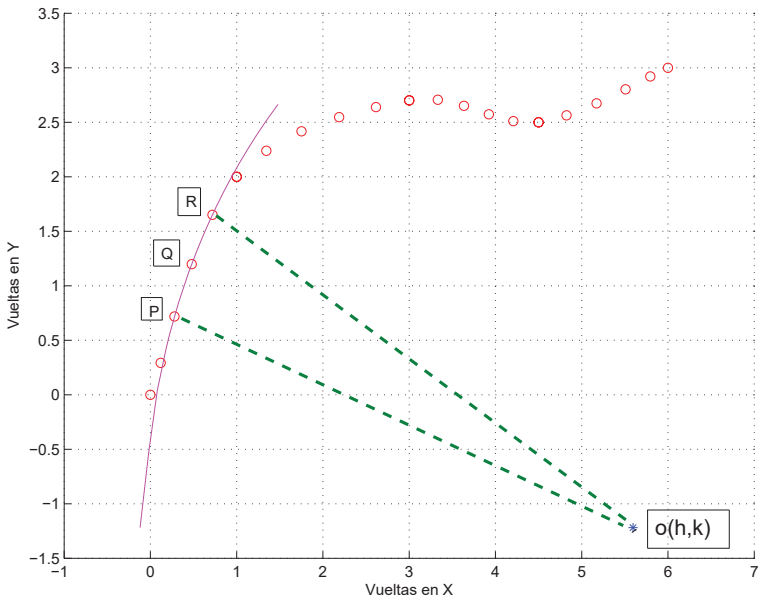


Figura 3.11: Círculo determinado por condiciones geométricas.

Considerando que la velocidad lineal del centro del robot móvil es una variable elegible, lo único que nos queda es despejar la velocidad de rotación ω_R de la ecuación 2.5, la cual corresponde a la velocidad angular de rotación de la estructura del robot móvil.

La ecuación (3.27) describe tal relación.

$$\omega_R = \frac{V}{R} \tag{3.27}$$

Si sustituimos este resultado en (3.28) descrita en el capítulo 2 obtenemos el vector de velocidades lineales para el actuador izquierdo y derecho del robot móvil.

$$\begin{aligned} V_R(t) &= \omega_R(t)(R + \tfrac{L}{2}) \\ V_L(t) &= \omega_R(t)(R - \tfrac{L}{2}) \end{aligned} \tag{3.28}$$

Este último par de ecuaciones requiere de un análisis más detallado, ya que éstas representan pieza vital en la rotación o giro del robot móvil (ver apéndice H).

3.5.1. Discretización de los segmentos círculares

Hasta este momento se ha descrito la mayor parte de los cálculos necesarios para la obtención del vector de velocidades, no obstante pensemos en la siguientes dos cuestiones: ¿Por cuanto tiempo es necesario mantener cada una de las velocidades? y ¿Como pasar de una velocidad a otra?, esta última la responderemos en las siguientes secciones, pero la primera se resuelve de la siguiente manera.

Dado que la velocidad angular ω_R se mantiene constante en cada segmento circular tenemos.

$$\omega_R = \frac{\theta_R}{t_s} \tag{3.29}$$

donde:

θ_R: Ángulo recorrido.
t_s: Tiempo de sostenimiento (segundos.)

Resolviendo para t_s tenemos:

$$t_s = \frac{\theta_R}{\omega_R} \tag{3.30}$$

Para encontrar el tiempo de sostenimiento de las velocidades, se requiere encontrar el ángulo θ_R el cual está determinado por los puntos P, Q y R (ver fig. 3.10 y 3.12).

El cálculo del ángulo de apertura θ_R se realiza aplicando la ley de los cosenos (ecuación. 3.31) al triángulo OPR mostrado en la figura 3.12.

$$C^2 = R^2 + R^2 - 2R^2 cos(\theta_R) \tag{3.31}$$

Donde:

$$C = \sqrt{(x_r - x_p)^2 + (y_r - y_p)^2} \tag{3.32}$$

Resolviendo para θ_R tenemos.

$$\theta_R = cos^{-1}(1 - \frac{C^2}{2R^2}) \tag{3.33}$$

Finalmente el número de datos (velocidades y tiempo) obtenidos a partir de una trayectoria discretizada con $\triangle u$ (Incremento del intervalo de definición de la curva tipo Spline de $Catmull - Rom$) se cálcula de la siguiente manera.

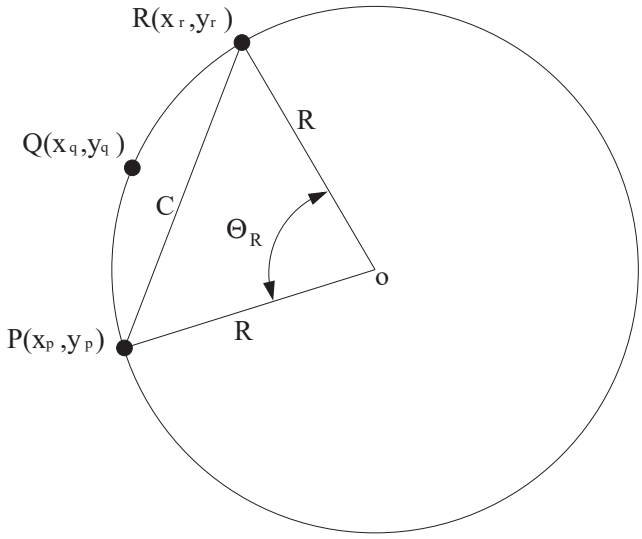


Figura 3.12: Triangulo asociado al segmento circular.

$$ND = (PC - 1)(\frac{\frac{1}{\triangle u}}{2}) \tag{3.34}$$

Donde:

ND: Número de datos (Longitud del vector de velocidades y tiempo).
PC: Número de puntos de control.

La relación (3.34) tiene una doble interpretación, ya que también denota el número de segmentos de arco interpolados por la ecuación del círulo. La selección de $\triangle u = 0.2$ impone una restricción en la selección del número de puntos de control. Esta restricción es que el número de puntos de control sea impar. De esta manera se asegura que el número de arcos y por consiguinete el número de datos sea un número entero.

Tomando en cuenta este último aspecto el conjunto de datos resultante tiene el formato mostrado en la tabla 3.1.

Este resultado, aún es incomprensible por los actuadores del robot móvil, por lo que es necesario dar un tratamiento final.

Tabla 3.1: Arreglo de velocidades y tiempo

Dato	Actuador Izq.	Actuador Der.	tiempo
1	$Vel\ Izq_1$	$Vel\ Der_1$	t_1
2	$Vel\ Izq_2$	$Vel\ Der_2$	t_2
.	.	.	.
.	.	.	.
.	.	.	.
ND	$Vel\ Izq_{ND}$	$Vel\ Der_{ND}$	t_{ND}

Primeramente es necesario transformar estas velocidades lineales en velocidades angulares, ya que estas últimas son las unidades de medida del movimento giratorio de las flechas de los motores de CD. Esta transformación se logra usando la ecuación (3.35).

$$\omega = \frac{V}{r} \tag{3.35}$$

Donde:

V: Velocidad lineal.
r: Radio nominal de la llanta del robot móvil.

En el capítulo 2 se comentó que los motores de CD cuentan con un reductor o caja de engranes conectada en la flecha del motor, la cual sirve para incrementar el par a costa de reducir la velocidad. Este aspecto debe de ser tomado en cuenta, ya que para la medición y control de la velocidad se toma como referencia la parte interior o dicho de otra manera el punto de medición está antes de la caja de engranes. La ecuación (3.36) muestra la relación entre las velocidades antes y despues de la caja de engranes.

$$\omega_m = K_g \omega_L \tag{3.36}$$

Donde:

ω_m: Velocidad angular antes de la caja de engranes.
ω_L: Velocidad angular después de la caja de engranes.

K_g: Constante de reducción (ver apéndice E).

Ya que la unidad de medida tomada como referencia son pulsos por milisegundo (Pulsos/ms), la transformación de la velocidad angular (r/s) a pulsos por milisegundo se hace usando la ecuación (2.37) desarrollada en el capítulo 2. Resolviendo para los Pulsos/s tenemos.

$$Pulsos/s = (r/s) * (Pulsos\ por\ rev/2\pi) \tag{3.37}$$

Transformando a milisegundos tenemos.

$$Pulsos/ms = (Pulsos/s) * (1s/1000ms) \tag{3.38}$$

Dado que el periodo de muestreo h para nuestro caso se seleccionó de $2ms$, los resultados obtenidos a partir de la ecuación (3.38) deben ser duplicados para transformarlos a $Pulsos/ms$.

El tiempo de sostenimiento es fácilmente convertible a unidades comprensibles por el microcontrolador, expresándolo en términos de un múltiplo (M) del periodo de muestreo (h).

$$M = \frac{t_s}{h} \tag{3.39}$$

Donde:

M: Múltiplo de la interrupción de tiempo real.
h: Valor de la interrupción de tiempo real.

Debido a que el periodo de muestreo o valor de la interrupción de tiempo real es un valor pequeño $2ms$, el error debido al redondeo, obtenido al realizar el cociente descrito en la ecuación (3.39) es del orden de las milésimas de segundo.

De esta manera finalmente obtenemos una serie de comandos ejecutables por los actuadores del robot móvil (μC y motores de CD), los cuales tienen el formato mostrado en la tabla 3.2.

Tabla 3.2: Arreglo final de comandos

Dato	Actuador Izq.	Actuador Der.	tiempo
1	$Pulsos/mS\ Izq_1$	$Pulsos/mS\ Der_1$	M_1
2	$Pulsos/mS\ Izq_2$	$Pulsos/mS\ Der_2$	M_2
.	.	.	.
.	.	.	.
ND	$Pulsos/mS\ Izq_{ND}$	$Pulsos/mS\ Der_{ND}$	M_{ND}

3.6. Ejecución del Vector de Velocidades

Una vez obtenido el vector de velocidades y tiempos, la ejecución se realiza basándonos en la siguiente idea. La PC de abordo procesa y calcula el vector de comandos mostrados en la tabla 3.2, posteriormente esta información es enviada via puerto serie (RS-232) y se almacena en memoria del microcontrolador en forma de arreglo. Una vez realizado esto, el siguiente paso es hacer un barrido de las velocidades y múltiplos de la interrupción, lo cual detallaremos en la siguiente sección. El diagrama de flujo de la figura 3.13 muestra de manera general las actividades realizadas durante la interrupción de tiempo real, incluyendo la ejecución del vector de velocidades.

Una de las primeras actividades que se realiza es la medición de la velocidad, la cual como ya se comentó en los capítulos anteriores se hace a través de los encoders correspondientes a cada motor. Posteriormente se pregunta a través de una bandera, sobre el estado de la ejecución de la rampa de arranque (ver subsección 3.6.1). En caso de resultar positivo el resultado, pasamos directamente a la parte donde se ejecutan los comandos de velocidad.

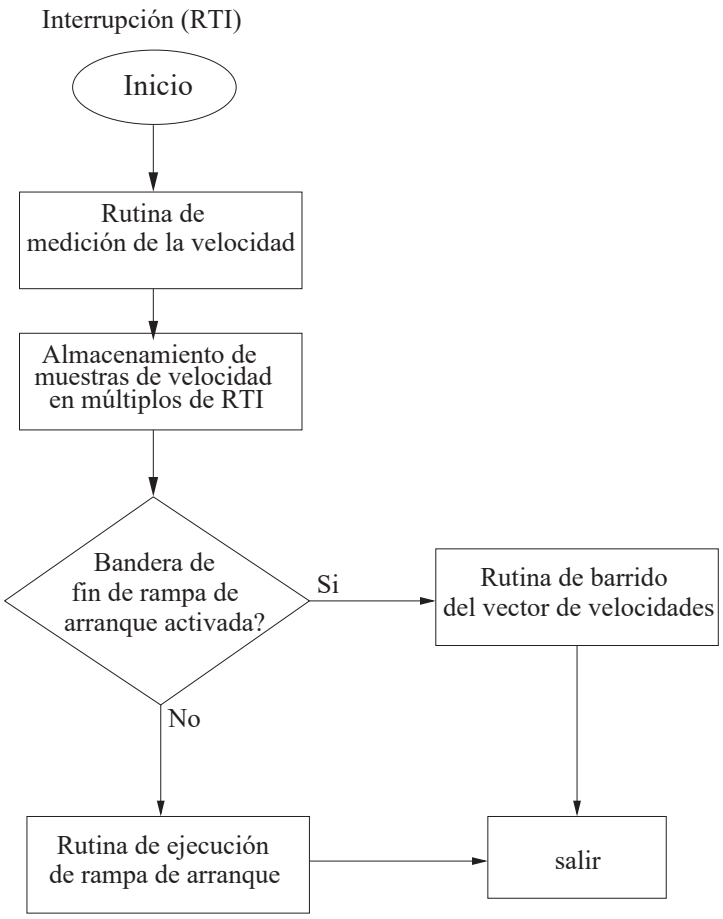


Figura 3.13: Diagrama de flujo de la interrupción de tiempo real.

El diagrama de flujo de la figura 3.14 muestra de manera general la idea implementada para ejecutar los comandos de velocidad calculados. La ejecución de esta idea se basa principalmente en dos preguntas, las cuales deciden el flujo del programa. Dado que los comandos de velocidad son una tabla de datos, es importante saber en cual renglón de la tabla nos encontramos, lo cual se decide en la primera parte.

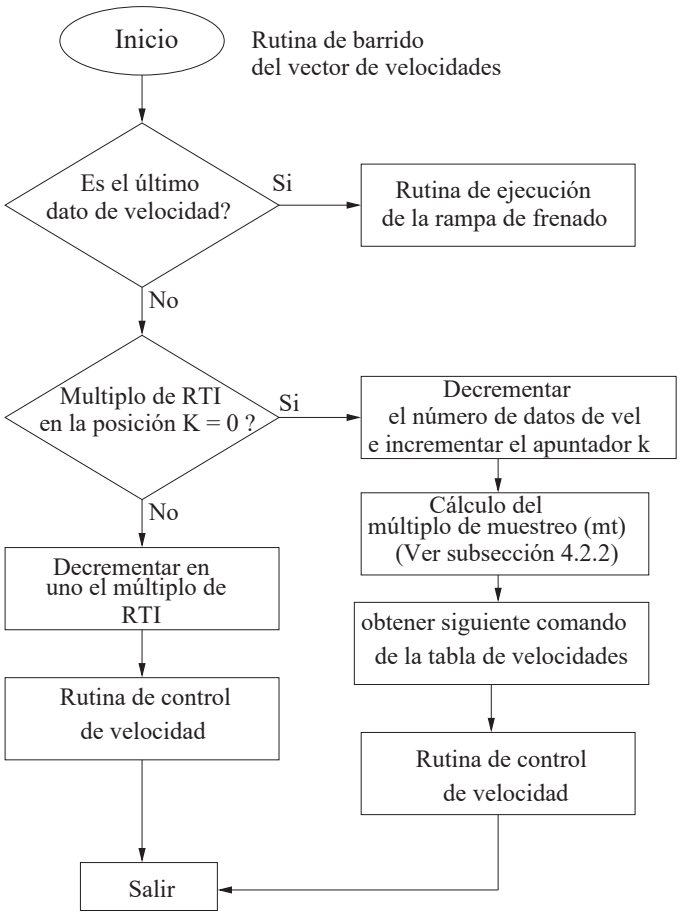


Figura 3.14: Diagrama de flujo de la ejecución del vector de velocidades.

Las siguientes actividades que se realizan son: decrementar el múltiplo de la interrupción de tiempo real y posterior a eso se ejecuta la rutina de control de velocidad (ver cap 2). Esta rama del diagrama de flujo se mantiene así hasta que el múltiplo de la interrupción llega a su fin, implicando esto dos aspectos: El primero asegura el hecho de que ya se ejecutó un comando o dato de velocidad, lo cual afecta directamente el estado del número de datos, el segundo aspecto es pasar al siguiente dato o comando de velocidad, para posteriormente pasarlo como referencia a la rutina de control de velocidad. Este proceso se ejecuta de manera iterativa hasta llegar al último dato de velocidad donde ahora se ejecuta la rutina para el frenado suave, la cual se describe en la siguiente subsección.

3.6.1. Arranque y frenado suave

El arranque y frenado suave es una buena técnica para la eliminación de efectos indeseables en el robot móvil, tales efectos pueden ser por ejemplo, las oscilaciones de la estructura debidas a un arranque o frenado repentinos. Daños en la instrumentación o incluso en la estructura pueden ser causa de la omisión de este aspecto. Afortunadamente la flexibilidad que ofrece el microcontrolador, permite una fácil implementación de dicha técnica.

La implementación de las rampas de arranque y frenado se hizo basándonos en la siguiente idea. Se dividió el primer y último comandos o datos de velocidad entre un múltiplo o un número de pasos (np) deseados. Para el caso de la rampa de arranque se inicializó o se utilizó como primer valor de referencia de velocidad el valor obtenido en el cociente descrito en (3.40), posteriormente este valor se incrementó en magnitud igual al mismo cociente hasta alcanzar el primer valor de velocidad calculado. Por otra parte la ejecución de la rampa de frenado se hizó de manera similar, solo que esta vez se utilizó el último valor o comando de velocidad como referencia de velocidad, posteriormente este valor se decrementó en magnitud similar al caso anterior hasta alcanzar una velocidad nula. El resultado de esta técnica se asemeja a una escalera con un número de escalones igual a np (ver figura 3.15).

$$Subref = \frac{refn}{np} \tag{3.40}$$

Donde:

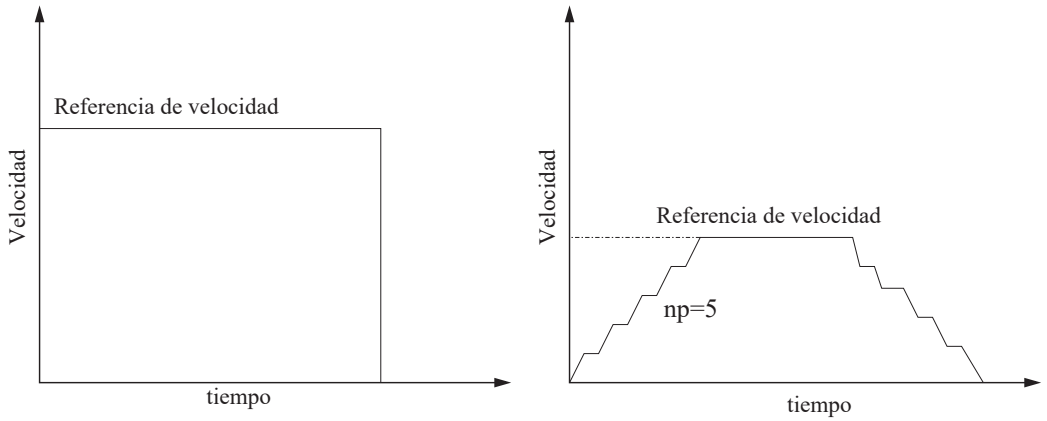
Subref: Subreferencia de velocidad.
refn: Primer y último comando de velocidad.
np: Número de pasos o múltiplo de las referencias de velocidad.

De manera similar para el primer y último dato del múltiplo de la interrupción tenemos.

$$Submrti = \frac{mrtin}{np} \tag{3.41}$$

Donde:

Submrti: Submúltiplo de la interrupcción de tiempo real.
mrtin: Primer y último dato del múltiplo de la interrupción.



(a) Arranque y frenado brusco. (b) Arranque y frenado suave.

Figura 3.15: Diferencia entre arranque brusco y suave.

A continuación describiremos usando para esto, los diagramas de flujo, las implementaciones tanto de la rampa de arranque como la de frenado. El diagrama de flujo de la figura 3.16 muestra de manera general la idea usada para la implementación tanto de la rampa de arranque como de la rampa de frenado.

En la primera parte del diagrama de flujo se prueba el estado del submúltiplo de la interrupción, el cual, al inicio obviamente tiene un valor diferente a cero, ya que en la rutina de inicialización de variables se realizaron las operaciones concernientes a esta variable (ver apéndice C). Debido a esto, el resultado a la primera pregunta es negativo y continua de esta manera hasta que el submúltiplo de la interrupción llega a su fin, implicando con esto la aseveración de que ya se ejecutó un paso de la rampa.

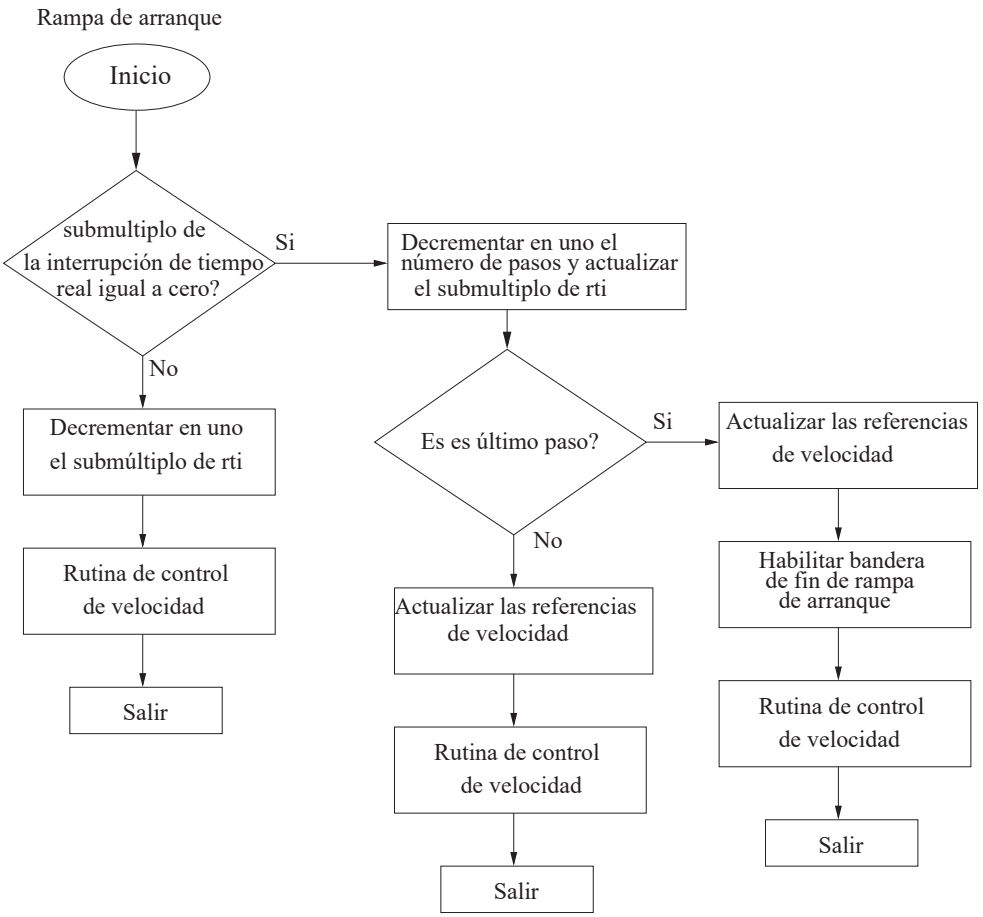


Figura 3.16: Diagrama de flujo para la rampa de arranque.

El siguiente paso es decrementar el número de pasos y actualizar la variable del submúltiplo de la interrupción. Posteriormente se prueba el estado del número de pasos a través de la segunda pregunta. Este proceso se repite hasta llegar al último paso de la rampa, en donde esta vez se habilita una bandera de fin de rampa, recordando que el diagrama de flujo de la figura 3.13 hace alusión a esta variable, con la finalidad de decidir el rumbo de las operaciones que se realizan dentro de la interrupción.

Como se comentó en los párrafos anteriores esta idea se aplicó para implementar la rampa de arranque y frenado, sin embargo existe una ligera variante para la rampa de frenado ya que en lugar de habilitar un bandera que indique el fin de la ejecución de la rampa, esta vez solo se restablecen algunas variables y se frenan ambos motores, indicando con esto el fin de la trayectoria.

3.6.2. Sensado de velocidades

Como se observó al inicio de este capítulo una trayectoria resulta en una serie de comandos de velocidad que deben ser ejecutados por los motores del robot móvil, es por esta razón que es importante comentar el mecanismo que se usó para el sensado de las velocidades. El sensado de las velocidades no difiere de la forma en la que se realizó en las pruebas del escalón, en donde el microcontrolador almacenaba las muestras de velocidad a un determinado periodo de muestreo para posteriormente enviarlas a la PC vía puerto serie.

Existe una diferencia con respecto al esquema del escalón. Dicha diferencia está en el periodo de muestreo, es decir, dado que el microcontrolador se usó como almacenador de muestras para ambos motores, existe una limitación en este sentido, caracterizada por la capacidad de memoria del microcontrolador.

En este caso se utilizaron para el almacenamiento de las muestras de velocidad, dos arreglos de longitud igual a 500, los cuales fueron suficientes para cubrir las trayectorias de prueba.

Tomando en cuenta lo descrito en los párrafos anteriores, el cálculo del periodo de muestreo m_t se realizó basado en la siguiente idea. Primeramente, es necesario repartir el número de muestras disponibles N_{md} entre el número de datos de velocidad ND, para así asegurarnos de cubrir toda la trayectoria. En secciones anteriores se comentó la forma de calcular la longitud del vector de comandos de velocidad (ver ec. 3.34). Debido a que los segmentos de arco interpolados por la ecuación del círculo, tienen diferentes longitudes, resulta conveniente calcular el periodo de muestreo en función de esta información. El múltiplo de la interrupción de tiempo real $mrti$ descrito por la ecuación (3.39), refleja claramente el hecho de la longitud de cada segmento de arco, por lo que nos es útil para hacer el cálculo del periodo de muestreo. Como ya se comentó en secciones anteriores, el número de datos de velocidad ND es igual al número de segmentos interpolados, es por esto que algunas ocasiones se ha dicho que una trayectoria equivale a un vector de comandos de velocidad. Las ecuaciones (3.42) y (3.43) describen lo antes comentado.

$$nPT = \frac{N_{md}}{ND} \tag{3.42}$$

Donde:

nPT: Es el número de muestras para cada segmento de arco.
N_{md}: Es el número de muestras disponibles.

y

$$m_t = \frac{mrti[k]}{nPT} \tag{3.43}$$

Donde:

m_t: Es el periodo de muestreo para la medición de velocidad
(múltiplo del valor de la interrupción de tiempo real).
$mrti[k]$: Es el múltiplo de la interrupción de tiempo real en la posición k.
k: Apuntador de la tabla de datos de velocidad y múltiplo de la interrupción.

El diagrama de flujo 3.14, descrito en la sección 3.6 muestra el momento en el que estos cálculos son realizados.

Por otra parte en el diagrama de flujo 3.13 se muestra el momento en el que las muestras de velocidad son almacenadas, situación que es detallada en el diagrama de flujo de la figura 3.17.

Un aspecto que es importante comentar es que el almacenamiento de las muestras de velocidad se realiza en múltiplos del periodo de muestreo seleccionado para fines de control. En nuestro caso el cálculo del error y de la acción de control se hizo cada dos milisegundos, por lo que este valor es el valor mínimo con el que se puede almacenar las muestras de velocidad.

El inicio del muestreo para el primer segmento, se realiza en el instante en que inicia la primera interrupción de tiempo real. Para lograr que el muestreo se lleve a cabo

en la primera interrupción fue necesario inicializar el periodo de muestro en cero. Una vez hecho esto, el múltiplo del muestreo m_t es refrescado con el valor calculado con la ecuación (3.43). Este proceso se repite una y otra vez hasta el fin de la trayectoria. El diagrama de flujo de la figura 3.17 muestra el esquema de almacenamiento de las muestras de velocidad.

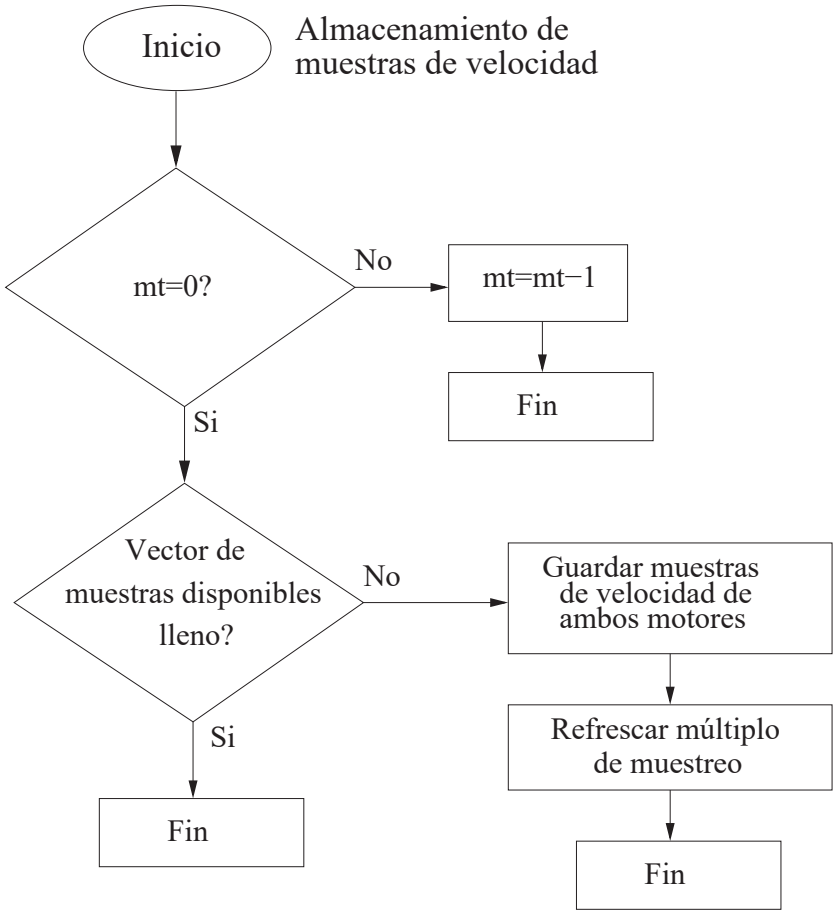


Figura 3.17: Diagrama de flujo para el almacenamiento de las muestras de velocidad.

3.6.3. Criterio de arranque

Antes de comenzar a describir los resultados obtenidos al someter a nuestro robot móvil a una serie de experimentos, es importante comentar un criterio que se adoptó para una correcta ejecución o seguimiento de una trayectoria.

Haciendo referencia a la figura 2.8 (a) expuesta en el capítulo 2, es notorio que el robot móvil tiene una posición (x, y) y una orientación θ referente a un marco base sobre el cual se moverá el robot móvil. En nuestro caso el cálculo de la trayectoria siempre supone que en el instante en que es inicializado el robot móvil, este se encuentra en el origen y apuntando hacia el eje X positivo, es decir $(x = 0, y = 0, \theta = 0)$ como lo muestra la figura 3.18.

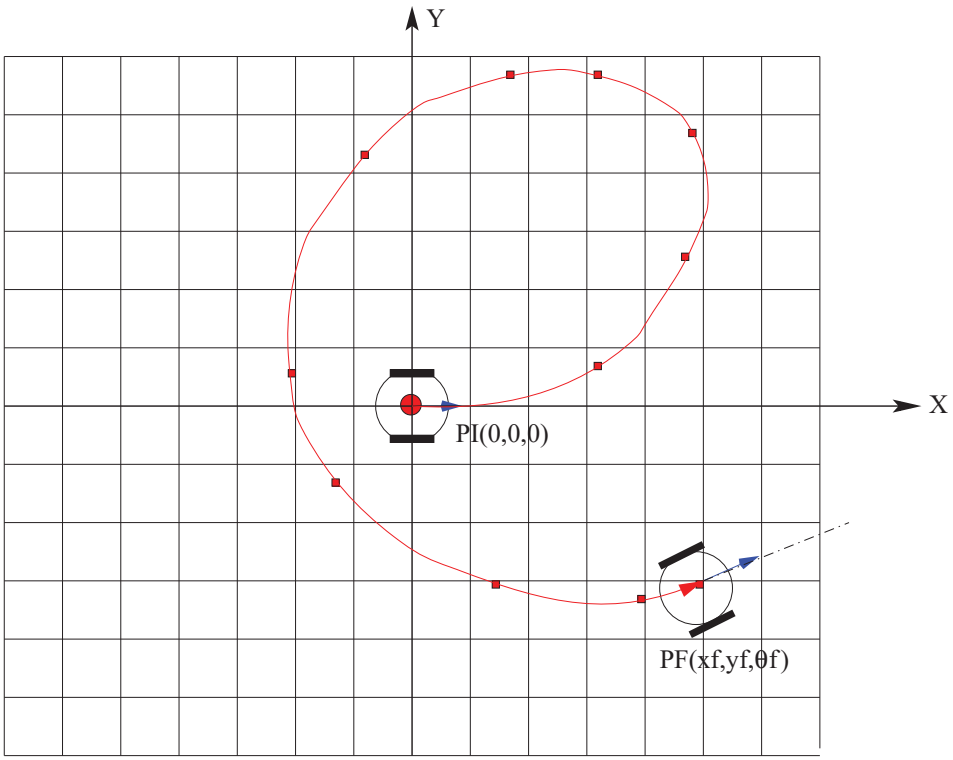


Figura 3.18: Posición inicial al momento de inicializar el robot móvil.

La solución que en este trabajo se propone, para lograr llevar al robot móvil desde una posición inicial hasta una posición final, es por supuesto a través de una trayectoria

dada. Esta trayectoria debe de iniciar o coincidir con la posición y orientación inicial del robot móvil (ver figura 3.18).

En la figura 3.19 se pueden observar un par de trayectorias $T1$ y $T2$, esta última coincide con la posición inicial del robot móvil, más no con su orientación. Si pretendieramos lograr $T2$, realmente lo que obtendriamos seria $T1$ debido a la orientación del robot móvil. Una alternativa para lograr $T2$ seria rotar el robot móvil sobre su eje hasta lograr la alineación con $T2$ y así comenzar con el seguimiento de la trayectoria, sin embargo esta situación nos obligaría a realizar este movimiento cada vez que se requiera comenzar una trayectoria.

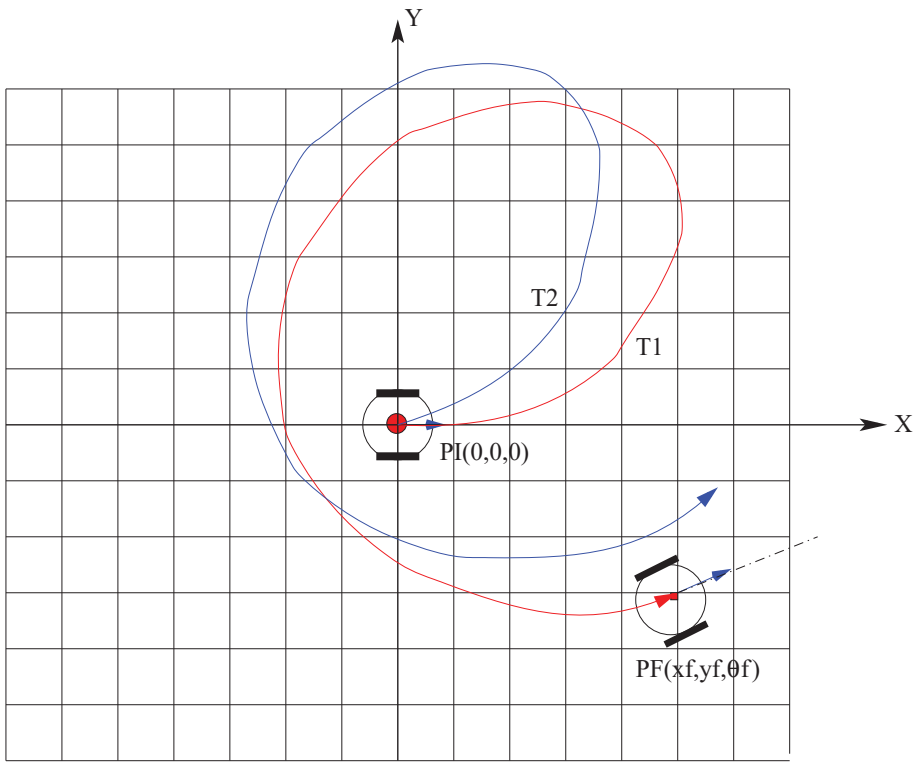


Figura 3.19: Inicio y orientación de la trayectoria.

En lugar de aplicar este último criterio, pensamos en diseñar una trayectoria que coincida con la orientación inicial del robot $\theta = 0$, y así lograr llevar a nuestro robot móvil desde su posición inicial, en este caso $(x = 0, y = 0, \theta = 0)$, hasta un objetivo final

$(xf, yf, \theta f)$. La trayectoria que cumple con lo recién establecido es $T1$ (ver figura 3.19).

3.7. Resumen

En esta última sección realizaremos un compendio de las ecuaciones más relevantes descritas a lo largo de este capítulo.

Comenzaremos en la sección 3.2, con la ecuación que define la curva tipo *Spline* de $Catmull - Rom$, la cual es usada para la generación de las trayectorias de prueba.

$$q_i(u) = \sum_{j=-1}^{2} V_{i+j}\varphi_j(u) \; u \in [0,1] \tag{3.44}$$

$$\varphi_{-1}(u) = -\beta_1^2 \frac{u^3-2u^2+u}{\beta_1+1} \tag{3.45}$$

$$\varphi_0(u) = \frac{(\beta_1^2+\beta_1+1)u^3-(2\beta_1^2+2\beta_1+1)u^2+(\beta_1^2-1)u+\beta_1+1}{\beta_1+1} \tag{3.46}$$

$$\varphi_1(u) = -\frac{(\beta_1^2+\beta_1+1)u^3-(2\beta_1^2+\beta-1+1)u^2-\beta_1 u}{\beta_1(\beta_1+1)} \tag{3.47}$$

$$\varphi_2(u) = \frac{u^3-u^2}{\beta_1(\beta_1+1)} \tag{3.48}$$

Con $\triangle u = 0,2$ y $\beta = 1$

De la sección 3.5 Generación del vector de velocidades tenemos las siguientes ecuaciones.

$$\begin{bmatrix} x_p & y_p & 1 \\ x_q & y_q & 1 \\ x_r & y_r & 1 \end{bmatrix} \begin{bmatrix} D \\ E \\ F \end{bmatrix} = \begin{bmatrix} -(x_p^2+y_p^2) \\ -(x_q^2+y_q^2) \\ -(x_r^2+y_r^2) \end{bmatrix} \tag{3.49}$$

$$(x + \frac{D}{2})^2 + (y + \frac{E}{2})^2 = \frac{D^2}{4} + \frac{E^2}{4} - F \tag{3.50}$$

Mismas que sirven para realizar el cálculo del radio de curvatura **R**.

$$\omega_R = \frac{V}{R} \tag{3.51}$$

$$\begin{aligned} V_R(t) &= \omega_R(t)(R \pm \tfrac{L}{2}) \\ V_L(t) &= \omega_R(t)(R \pm \tfrac{L}{2}) \end{aligned} \tag{3.52}$$

Con este par de ecuaciones se encuentra la velocidad angular del centro del robot y las velocidades lineales para ambas llantas respectivamente (ver apéndice H).

De 3.5.1 tenemos las siguientes ecuaciones.

$$t_s = \frac{\theta_R}{\omega_R} \tag{3.53}$$

$$C = \sqrt{(x_r - x_p)^2 + (y_r - y_p)^2} \tag{3.54}$$

$$\theta_R = cos^{-1}(1 - \frac{C^2}{2R^2}) \tag{3.55}$$

Mismas que sirven para encontrar el tiempo de sostenimiento de las velocidades de ambas llantas. En esta misma subsección tenemos las siguientes ecuaciones.

$$\omega = \frac{V}{r} \tag{3.56}$$

$$\omega_m = K_g \omega_L \tag{3.57}$$

$$Pulsos/s = (r/s) * (Pulsos\ por\ rev/2\pi) \tag{3.58}$$

$$Pulsos/ms = (Pulsos/s) * (1s/1000ms) \tag{3.59}$$

Con estas cuatro últimas ecuaciones se realiza el cálculo de la velocidad angular para ambos motores en unidades de pulsos por milisegundo. Es importante comentar que las unidades de las todas las ecuaciones descritas en este resumen serán descritas en el capítulo de experimentación.

Finalmente en esta misma subsección tenemos la siguiente ecuación, la cual sirve para el cálculo del múltiplo de la interrupción de tiempo real.

$$M = \frac{t_s}{h} \tag{3.60}$$

Hasta aqui hemos presentado de forma ordenada el conjunto de ecuaciones que sirven para realizar los cálculos de los comandos de velocidad y tiempo de ejecución de estos comandos. En el siguiente capítulo se realizaran los cálculos numéricos para la primer trayectoria de prueba. Esto con la finalidad de mostrar el uso de este resumen de ecuaciones y las unidades que intervienen en cada una de ellas.

Capítulo 4

PRUEBAS Y RESULTADOS

4.1. Experimentación

4.1.1. Descripción del terreno de prueba

El tipo de terreno bajo el cual se realizó la experimentación es de dos tipos. Una superficie lisa (alfombra) y una superficie rugosa (concreto). Este par de superficies se usaron de forma aislada (ver 4.1.2 y 4.2) y de forma combinada (ver 4.1.3 y 4.1.4) durante la experimentación con nuestro robot móvil. Es notorio que una superficie rugosa representa mayor grado de dificultad al navegar o trasladarse sobre ella, con respecto a una superficie lisa. En las siguientes subsecciones describiremos los resultados obtenidos al utilizar nuestro robot móvil sobre este par de superficies.

4.1.2. Trayectoria sencilla

La primera trayectoria de prueba descrita en la figura 4.1, es una trayectoria corta en cuanto a distancia se refiere y con un par de curvas relativamente suaves. Los puntos de control referentes a esta trayectoria se presentan en la tabla 4.1.

Para propósitos de demostración a continuación calcularemos el primer par de comandos de velocidad que describen los dos primeros segmentos de la trayectoria. Un aspecto que vale la pena comentar es que para la experimentación, todos los cálculos de distancia realizados se hicieron en términos de vueltas de las ruedas de nuestro robot móvil. Esto

Tabla 4.1: Puntos de control

Punto de control	Coordenada en X	Coordenada en Y
1	0	0
2	2	0.2
3	3.5	2
4	4.5	3
5	6	2.7

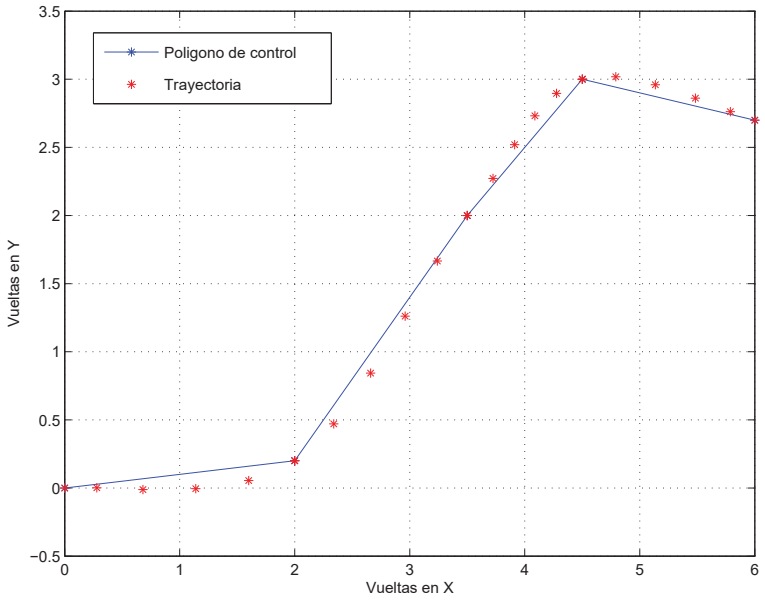


Figura 4.1: Trayectoria de prueba.

con la finalidad de tener una escala pequeña, la cual es fácil de implementar en espacios reducidos, lo cual es nuestro caso, además de que nos permite observar con mayor detalle el funcionamiento del robot móvil.

El primer paso a realizar es usar la curva tipo *Spline* de $Catmull - Rom$ para interpolar con curvas suaves los puntos de control (ver ecuación 3.16). De esta manera las dos primeras tercias de puntos que usaremos para cálcular los dos primeros comandos de velocidad se muestran en la tabla 4.2.

Tabla 4.2: Puntos intermedios

Punto	Coordenada en X	Coordenada en Y
1	0	0
2	0.28	0.0016
3	0.68	-0.0112
4	1.14	-0.0048
5	1.6	0.0544
6	2	0.2

El siguiente paso es encontrar los radios de curvatura $\mathbf{R}$ usando las ecuaciones (3.24) y (3.26) descritas anteriormente.

$$\begin{bmatrix} D \\ E \\ F \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0.28 & 0.0016 & 1 \\ 0.68 & -0.0112 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ -0.07840 \\ -0.4625 \end{bmatrix} = \begin{bmatrix} -0.38306 \\ 18.0369 \\ 0.00 \end{bmatrix} \tag{4.1}$$

Ahora si sustituimos los valores de D E y F en la ecuación (3.26) tenemos.

$$R = \sqrt{\frac{D^2}{4} + \frac{E^2}{4} - F} = \sqrt{\frac{-0.38306^2}{4} + \frac{18.036^2}{4} - 0} = 9.0211 \tag{4.2}$$

Éste corresponde al primer radio (R1) mostrado en la figura 4.2. De forma similar si tomamos a partir del tercer punto hasta el quinto tenemos que el segundo radio (R2) de curvatura es $R = 4.0513$, el cual podemos observar en la misma figura.

Para transformar algunos de los parámetros del robot como el radio de las llantas r, la distancia entre ellas L, entre otras, en unidades de vueltas, es necesario realizar los siguientes cálculos.

En este caso el radio de las llantas de nuestro robot móvil es aproximadamente de $r = 7cm$, por lo que tenemos un perímetro de $P = 44cm$ el cual equivale a una vuelta de la rueda del robot móvil. Por conveniencia para la experimentación tomaremos la velocidad

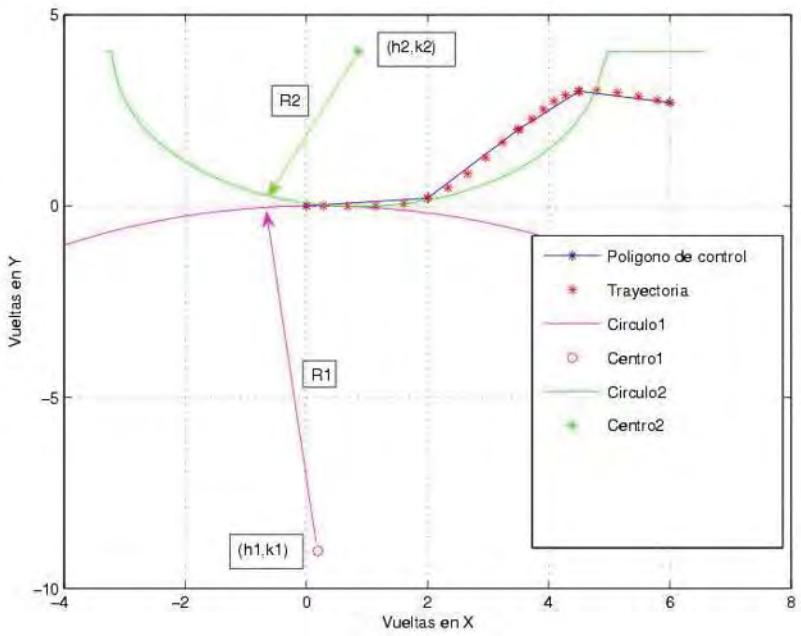


Figura 4.2: Arcos de círculo de los dos primeros segmentos de la trayectoria.

lineal de translación del robot móvil, en vueltas por segundo $Vueltas/s$. Aun y cuando estas unidades de medida no son muy comunes, son fácilmente transladables a m/s. u otras que se desee.

Si elegimos como velocidad lineal del centro de masa $V = 0.3068\ Vueltas/s.$, la cual es equivalente a $20 Pulsos/ms.$ y usando la ecuación (3.27) tenemos.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} = \begin{bmatrix} \frac{0.3068}{9.0211} \\ \frac{0.3068}{4.0513} \end{bmatrix} = \begin{bmatrix} 0.034 \\ 0.07572 \end{bmatrix} \tag{4.3}$$

Este par de velocidades angulares de rotación en r/s de la estructura del robot móvil, corresponden a las velocidades necesarias para describir los primeros dos tramos o segmentos de la trayectoria, como se observa en la figura 4.3, la cual es una ampliación de la figura 4.2. En ella se puede observar como el primer arco de círculo implica un giro en sentido horario, mientras que en el segundo el giro es en antihorario. Esta situación puede

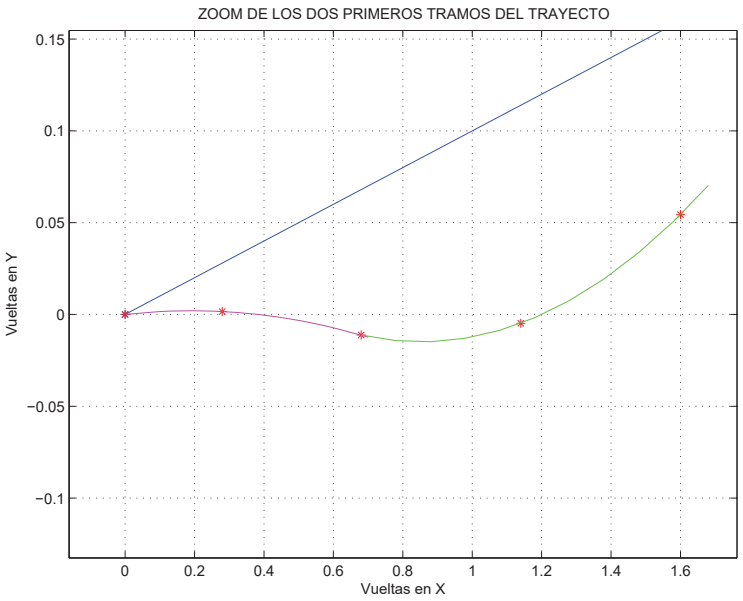


Figura 4.3: Primeros dos tramos de la trayectoria.

representar un problema a la hora de utilizar el par de ecuaciones enumeradas en (2.2) las cuales sirven para calcular las velocidades lineales de ambos motores que describirán dichos giros. El problema está en que este par de ecuaciones describen un giro antihorario como el del segundo tramo, obviamente considerando que el robot móvil recorrerá la trayectoria de izquierda a derecha. Este par de ecuaciones sin embargo pueden ser modificadas fácilmente para que describan un giro en sentido horario cambiando únicamente el signo del término $R \pm L/2$ en ambas ecuaciones como se expone en (4.4) (ver apéndice H).

$$V_R(t) = \omega(t)(R - \tfrac{L}{2})$$
$$V_L(t) = \omega(t)(R + \tfrac{L}{2}) \tag{4.4}$$

De acuerdo con esto se hace el cálculo de las velocidades lineales necesarias para describir el primer segmento de la trayectoria. Primeramente es necesario transformar el parámetro L (Distancia entre llantas, ver figura 2.8) en unidades de vuelta. Esto se realiza de la siguiente manera. De acuerdo con las características físicas del robot móvil tenemos

que $L = 42cm$, y para tener este valor en función de vueltas solo es necesario dividir entre el perimetro calculado previamente, esto es. $L = 42/44 = 0.9545$ de vuelta, de manera similar y ya que posteriormente usaremos este parámetro, el radio nominal de las llantas queda $r = 7/44 = 0.159$ de vuelta. Sustituyendo valores en (4.4) tenemos.

$$\begin{aligned} V_{R1} &= 0.034(9.0211 - \tfrac{0.9545}{2}) = 0.2906\ Vueltas/s. \\ V_{L1} &= 0.034(9.0211 + \tfrac{0.9545}{2}) = 0.3229\ Vueltas/s. \end{aligned} \tag{4.5}$$

Similarmente pero esta vez usando 2.2 para el siguiente tramo tenemos.

$$\begin{aligned} V_{R2} &= 0.07572(4.0513 + \tfrac{0.9545}{2}) = 0.3427\ Vueltas/s. \\ V_{L2} &= 0.07572(4.0513 - \tfrac{0.9545}{2}) = 0.2708\ Vueltas/s. \end{aligned} \tag{4.6}$$

Las cuales para transformarlas a velocidades angulares es necesario aplicar la relación entre la velocidad lineal y angular descrita por (3.35).

$$\begin{aligned} \omega_{R1} &= \tfrac{0.2906}{0.159} = 1.8276\ r/s. \\ \omega_{L1} &= \tfrac{0.3229}{0.159} = 2.030\ r/s. \end{aligned} \tag{4.7}$$

y

$$\begin{aligned} \omega_{R2} &= \tfrac{0.3427}{0.159} = 2.1553\ r/s. \\ \omega_{L2} &= \tfrac{0.2708}{0.159} = 1.7031\ r/s. \end{aligned} \tag{4.8}$$

Multiplicando estos valores por la constante de la caja de engranes para referirlas a la parte anterior de ésta, obtenemos lo siguiente.

$$\begin{aligned}
\omega_{A_R1} &= (1.8276)(65.5) = 119.7\ r/s. \\
\omega_{A_L1} &= (2.030)(65.5) = 132,96\ r/s. \\
\omega_{A_R2} &= (2.1553)(65.5) = 141.17\ r/s. \\
\omega_{A_L2} &= (1.7031)(65.5) = 111.55\ r/s.
\end{aligned} \tag{4.9}$$

Convirtiendo estos valores a pulsos por milisegundo mediante (3.38) tenemos.

$$\begin{aligned}
(Pulsos/ms)_{A_R1} &= \tfrac{119.7*1000}{2*\pi} * 0.001 = 19.05 \\
(Pulsos/ms)_{A_L1} &= \tfrac{132,96*1000}{2*\pi} * 0.001 = 21.16 \\
(Pulsos/ms)_{A_R2} &= \tfrac{141.17*1000}{2*\pi} * 0.001 = 22.46 \\
(Pulsos/ms)_{A_L2} &= \tfrac{111.55*1000}{2*\pi} * 0.001 = 17.75
\end{aligned} \tag{4.10}$$

Como el control funciona sólo con valores enteros, es necesario redondear al valor más cercano. En la tabla 4.3 se muestran los comandos de velocidad obtenidos al realizar los cálculos para cada tercia de puntos de la trayectoria completa. Para no saturarnos con operaciones engorrosas, el cálculo del múltiplo de la interrupción de tiempo real, no se muestra, de cualquier manera en secciones anteriores se explicó la forma de realizarlo.

Tabla 4.3: Comandos de velocidad y múltiplo de RTI

P/ms Mizq	P/ms Mder	Múltiplo de RTI	Velocidad Promedio (P/ms.)
21	19	1109	$\frac{21+19}{2} = 20$
18	22	1506	$\frac{18+22}{2} = 20$
13	27	1408	$\frac{13+27}{2} = 20$
18	22	1639	$\frac{18+22}{2} = 20$
21	19	1493	$\frac{21+19}{2} = 20$
19	21	1082	$\frac{19+21}{2} = 20$
26	14	857	$\frac{26+14}{2} = 20$
33	7	885	$\frac{33+7}{2} = 20$
23	17	1159	$\frac{23+17}{2} = 20$
19	21	881	$\frac{19+21}{2} = 20$

En la figura 4.4 se puede observar el perfil de velocidades obtenido al realizar el seguimiento de la trayectoria de prueba. Como podemos observar al inicio existe una rampa compuesta por cinco pasos de arranque, la cual acelera de forma gradual la estructura del robot móvil, terminando en el primer comando de velocidad. Así mismo se observa como el control implementado, sigue los comandos de velocidad con un grado de precisión aceptable. De hecho en estado estable y bajo ningún tipo de perturbación externa el error en el seguimiento de la velocidad es de ± 1 Pulso/ms (ver figura 4.5).

Observe como a lo largo del perfil de velocidades mostrado en la figura 4.4 siempre el par de comandos de velocidad correspondiente a cada segmento circular mantiene la velocidad promedio deseada, en este caso de $20 Pulsos/ms$. Las características de una trayectoria influyen en los cambios de velocidad de cada motor, como se observa en los últimos tres valores antes de la rampa de desaceleración. En ella se puede observar como la velocidad pasa de $26 Pulsos/ms$ a $33 Pulsos/ms$ en el motor izquierdo, mientras que para el derecho, el cambio es de $14 Pulsos/ms$ a $7 Pulsos/ms$. La escala de variación del perfil de velocidades de la figura 4.4 no pasa de $10 Pulsos/ms$, siendo precisamente este valor el máximo, el cual se presenta en el cambio del penúltimo al último comando de velocidad para ambos motores. Esta situación se presenta generalmente cuando la trayectoria contiene un cambio de un tramo con una curvatura suave o casi recto, a uno con curvatura pronunciada, situación que se presenta casi al final de nuestra trayectoria de prueba. Puede haber trayectorias con cambios de curvatura tales que, incluso requieran que una llanta se frene mientras que la otra lo impulsa provocando un giro en el robot.

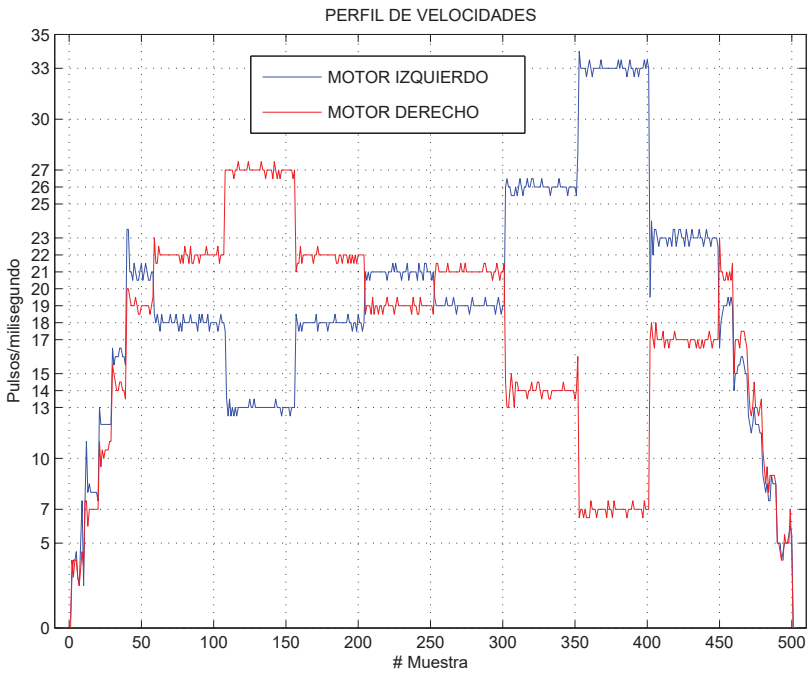


Figura 4.4: Perfil de velocidades correspondiente a la trayectoria.

El razonamiento al que nos lleva esta situación es al del diseño de una trayectoria menos costosa desde este punto de vista, lo cual por el momento queda fuera de los objetivos de este trabajo.

Por otro lado como el muestreo para el sensado de velocidades se realiza en múltiplos del periodo seleccionado para el control (2ms.) los pulsos almacenados en el acumulador deben ser divididos a la mitad para convertirlos a pulsos por milisegundo, es por esta razón que en la gráfica de la figura 4.5 se observa una resolución de $\pm\frac{1}{2}$ Pulso/ms.

Finalmente la rampa de desaceleración también con cinco pasos reductores de la velocidad, logra que la estructura del robot llege a su objetivo de manera suave. La selección de los cinco pasos de aceleración y de la desaceleración representó una buena opción para nuestro caso; sin embargo este valor se puede ajustar con la finalidad de obtener un arranque aún más suave.

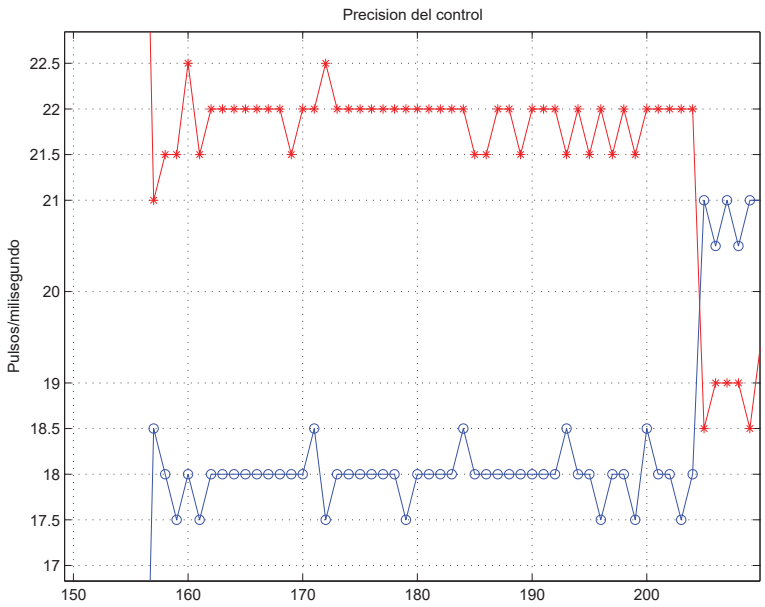


Figura 4.5: Variación de la velocidad.

Velocidad promedio

La figura 4.6 muestra el comportamiento de la velocidad promedio referente al centro de masa del robot móvil, la cual se obtiene a partir de los valores mostrados en la figura 4.4. En ella se pueden observar algunas variaciones en la velocidad que podrían en caso extremo lograr que el robot móvil se desvíe de la trayectoria planeada, sin embargo éstas son reguladas por el control de forma rápida logrando mantener la velocidad deseada, que en este caso se trata de $V = 0.3068\ Vueltas/s.$ equivalente a 20 $Pulsos/ms.$

Error de posición

El error en la posición es en realidad el error del robot móvil al recorrer una trayectoria de prueba. Físicamente resulta interesante construir un ambiente de prueba donde se puedan trazar las trayectorias de prueba, así como la trayectoria que describe el robot móvil, para posteriormente compararlas y establecer el grado de precisión con el que nuestro robot

móvil describe dichas trayectorias. Para esto se construyó un marco o eje coordenado como el mostrado en la figura 3.18, situado sobre el suelo (ver fig. 4.7).

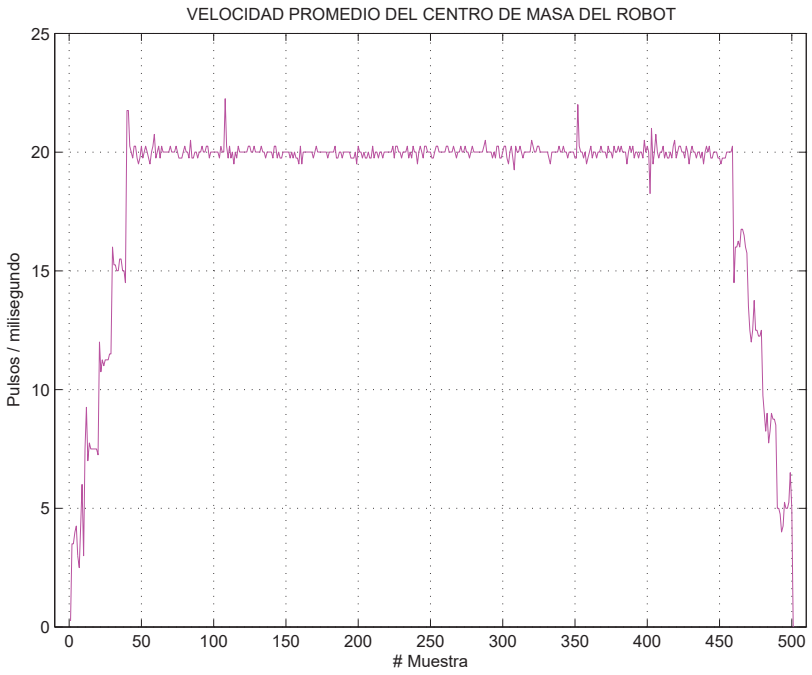


Figura 4.6: Velocidad promedio referente al centro de masa del robot móvil.

El mecanismo de medición del error de la posición propuesto fué el siguiente. Colocar o marcar los diferentes puntos de control sobre el marco o eje coordenado relacionados con la trayectoria de prueba. Posteriormente ejecutar el experimento de seguimiento de la trayectoria con el robot móvil y observar y medir la diferencia entre ambas.

De acuerdo con lo comentado en el párrafo anterior el error en la posición referente a la primer trayectoria de prueba se puede observar en la figura 4.8. Los puntos en verde representan posición alcanzada por el centro del robot durante el seguimiento de la trayectoria de prueba, mientras que la orientación es la de la línea recta formada por el punto referente a la posición actual y el punto referente a la posición anterior.

El seguimiento de la trayectoria siempre se realiza comenzando en el origen del

marco de prueba y con una orientación hacia el eje de las x positivas, como se comentó oportunamente en 3.6.3. En lo consecutivo es importante mantener en mente lo recién comentado, para asi dar una correcta interpretación a las gráficas de los errores de las siguientes trayectorias de experimentación.

Figura 4.7: Marco de prueba.

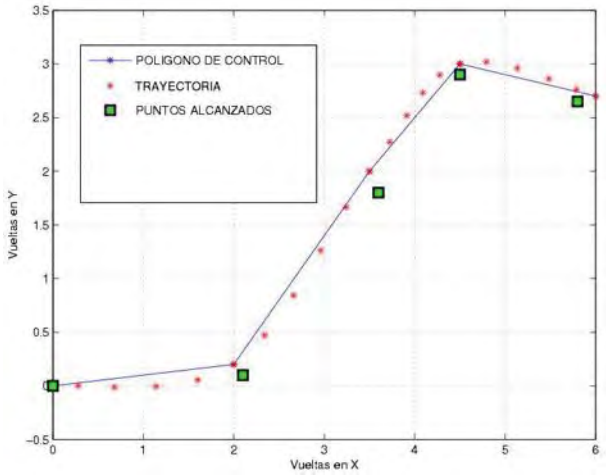


Figura 4.8: Puntos alcanzados por el robot móvil.

4.1.3. Trayectoria larga

En este experimento se expuso al robot móvil a una trayectoria sencilla pero relativamente larga. Con este tipo de trayectoria se pretende mostrar la precisión del robot móvil para llegar a su objetivo final. En la figura 4.9 se muestra dicha trayectoria.

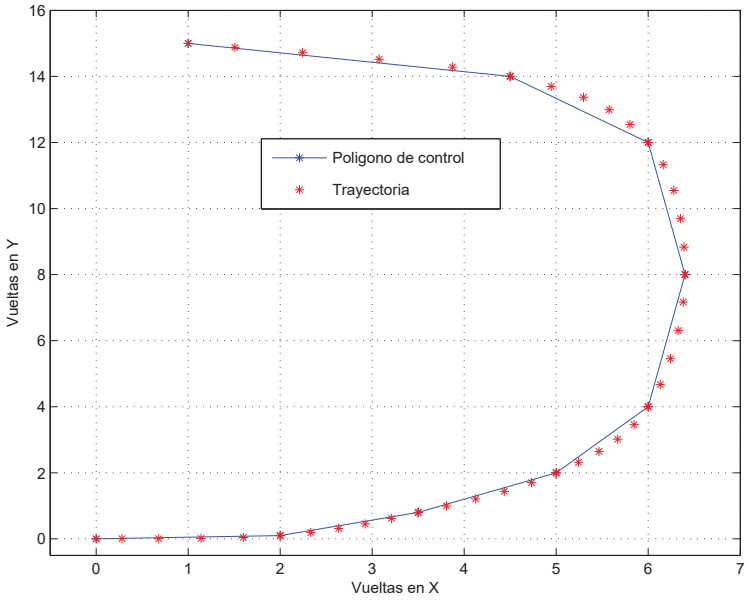


Figura 4.9: Trayectoria de prueba.

Cuyos puntos de control se muestran en la tabla 4.4.

Tabla 4.4: Puntos de control para la segunda trayectoria.

Punto de control	Coordenada en X	Coordenada en Y
1	0	0
2	2	0.1
3	3.5	0.8
4	5	2
5	6	4
6	6.4	8
7	6	12
8	4.5	14
9	1	15

En esta ocasión la velocidad con la que se realizó este experimento fue de 0.6136 $Vueltas/s$; la cual es equivalente a 40 $Pulsos/ms$. Recordando que esta velocidad es relativa al centro de masa del robot móvil. En la figura 4.10 se puede observar el perfil de velocidades muestreado durante el seguimiento de esta trayectoria.

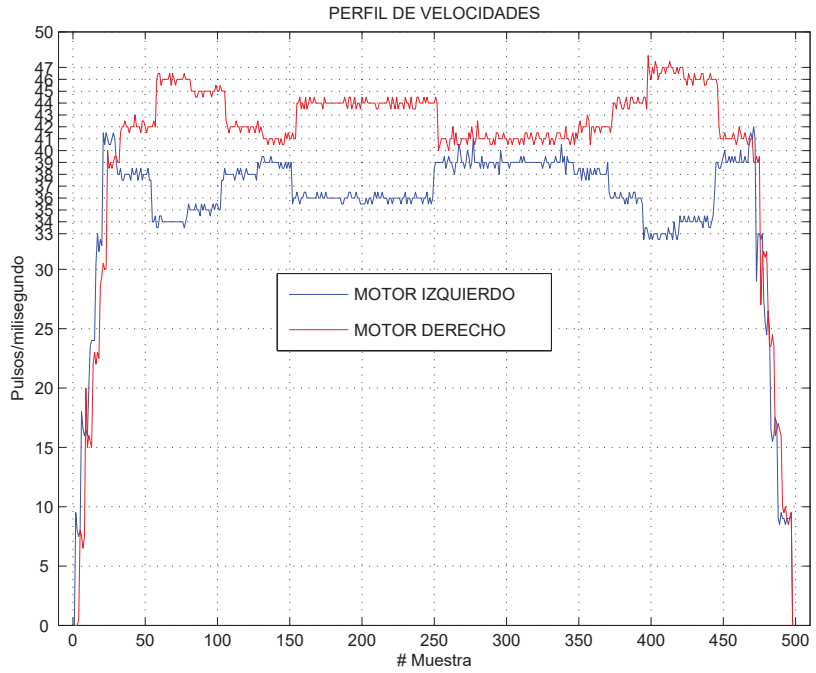


Figura 4.10: Perfil de velocidades correspondiente a la segunda trayectoria.

Una característica de este perfil de velocidades es que no presenta cambios bruscos entre las referencias de velocidad. Esto es comprensible ya que la trayectoria de prueba no presenta curvaturas pronunciadas. En realidad el mayor cambio existente entre las referencias de velocidad es de $5Pulsos/ms$, el cual se presenta casi al final del perfil de velocidades.

Velocidad promedio

En la figura 4.11 se puede observar el comportamiento de la velocidad referente al centro de masa del robot móvil y aún que esta fué mayor con respecto al experimento anterior y al que comentaremos al final, no se trata del valor máximo alcanzable por los motores. Las

pruebas realizadas a los motores de CD bajo condiciones de carga (ver cap 2) muestran claramente como el motor izquierdo alcanza una velocidad promedio máxima de 78 $Pulsos/ms$, mientras que para el motor derecho el promedio máximo es de 75 $Pulsos/ms$. El incremento de la velocidad con respecto al experimento anterior nos permitió obtener mejores resultados en cuanto al error de posición, sin embargo el incrementar la velocidad del centro de masa requiere de un compromiso con los valores máximos de las velocidades recién comentados.

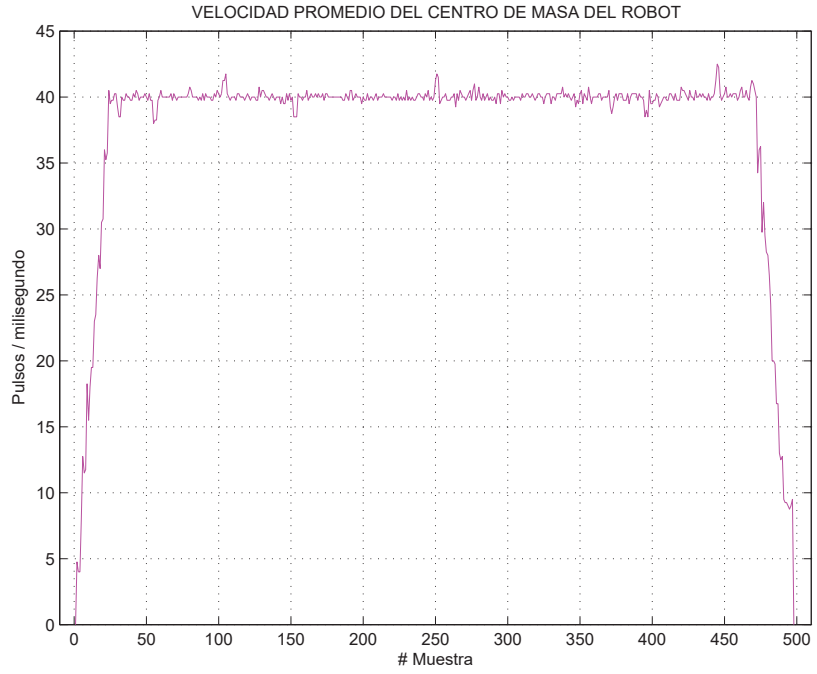


Figura 4.11: Velocidad promedio para la segunda trayectoria.

Error de posición

La suavidad de la trayectoria de prueba, permitió obtener anticipándonos al siguiente experimento, el mejor resultado respecto a la posición. En la figura 4.12 se puede observar como prácticamente los primeros cinco puntos son alcanzados con un error mínimo, mientras que en los consecutivos se observó una ligera desviación. La fricción no uniforme de la superficie o suelo, es entre muchas una de las causas de la acumulación de errores, no obstante los resultados obtenidos en este caso fueron altamente aceptables, ya que el robot

móvil alcanzó satisfactoriamente a su objetivo final.

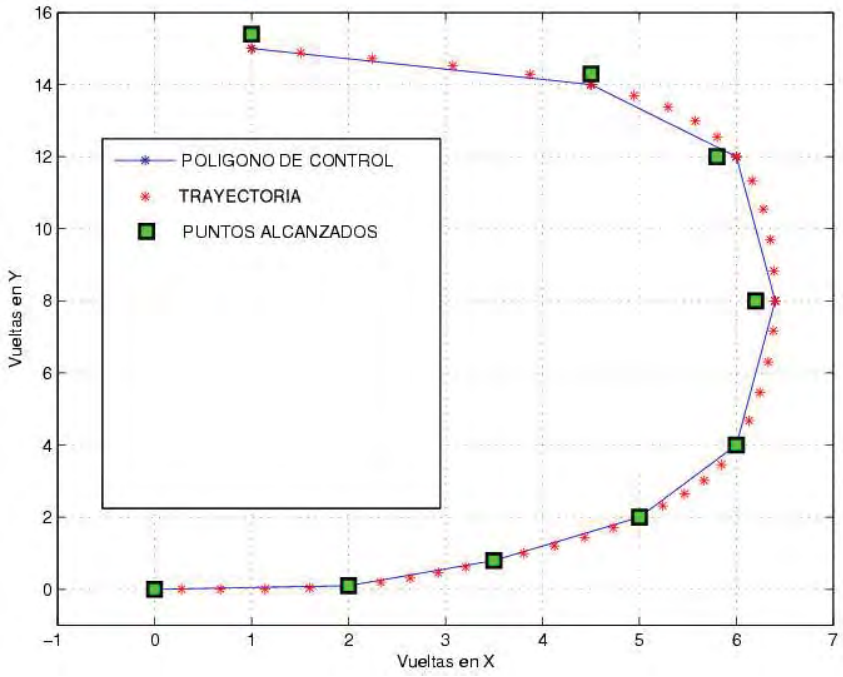


Figura 4.12: Puntos alcanzados para la segunda trayectoria.

4.1.4. Trayectoria larga con curvas

Esta última trayectoria de prueba se diseño con la finalidad de exponer al robot móvil a una trayectoria con curvas pronunciadas. En la figura 4.13 se puede observar las características de esta trayectoria.

Una trayectoria con curvas pronunciadas representa una prueba con un grado de dificultad mayor comparanda con las anteriores, ya que requiere de mayor habilidad por parte del robot móvil (actuadores) para lograr que éste pase por los puntos requeridos y llegue a su objetivo final.

En la tabla 4.5 se muestan los puntos de control relativos a esta trayectoria.

Tabla 4.5: Puntos de control para la tercer trayectoria.

Punto de control	Coordenada en X	Coordenada en Y
1	0	0
2	2	-0.1
3	4.2	-0.3
4	4.5	2
5	3.5	3.5
6	1	3.5
7	-0.5	2
8	0.2	-2
9	0	-6
10	0.3	-9
11	1.5	-10.5

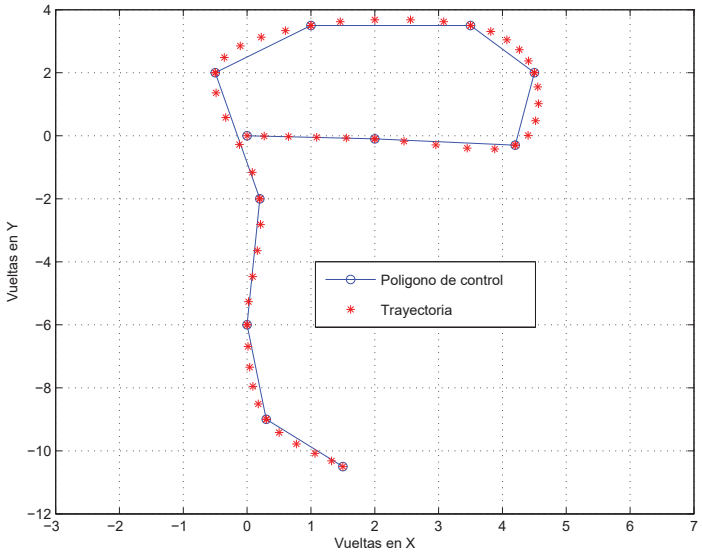


Figura 4.13: Tercer trayectoria de prueba.

La velocidad seleccionada para realizar este experimento fué de 0.46 $Vueltas/s$. la cual equivale a 30 $Pulsos/ms$. Este valor de velocidad es un valor intermedio con respecto a los experimentos anteriores y además puede ser considerada como una velocidad baja.

En la figura 4.14 se muestra el perfil de velocidades obtenido al realizar el seguimiento de la trayectoria mostrada en la figura 4.13. En este caso el mayor cambio entre las referencias de velocidad fué de 14 $Pulsos/ms$. el cual se presentó en el transcurso del tercer al cuarto comando de velocidad después de la rampa de aceleración.

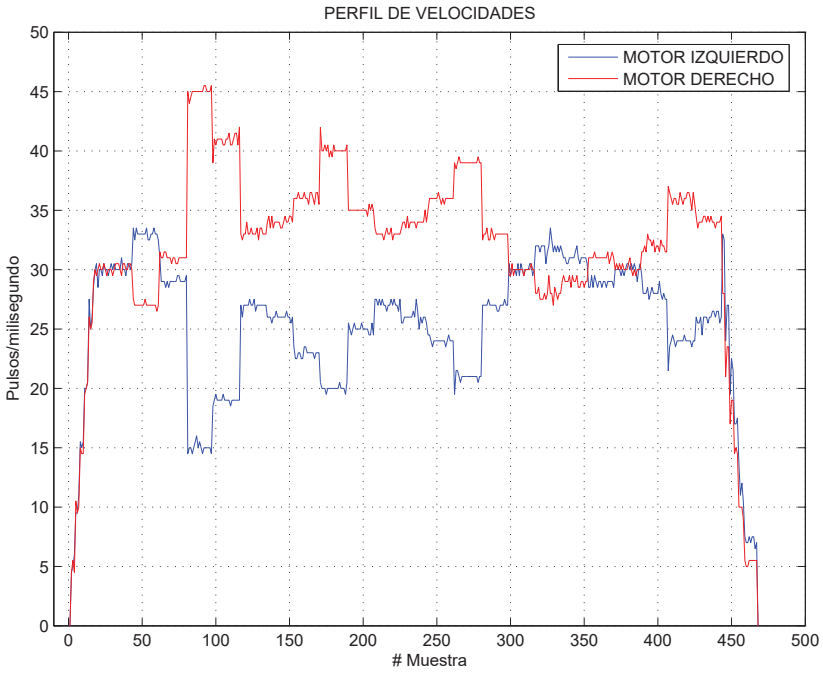


Figura 4.14: Perfil de velocidades para la tercera trayectoria.

Velocidad promedio

El comportamiento de la velocidad promedio referente al centro de masa del robot móvil se muestra en la figura 4.15, en ella se puede ver como el control logra mantener o fijar de manera aceptable la velocidad promedio deseada, en este caso de 30 $Pulsos/ms$.

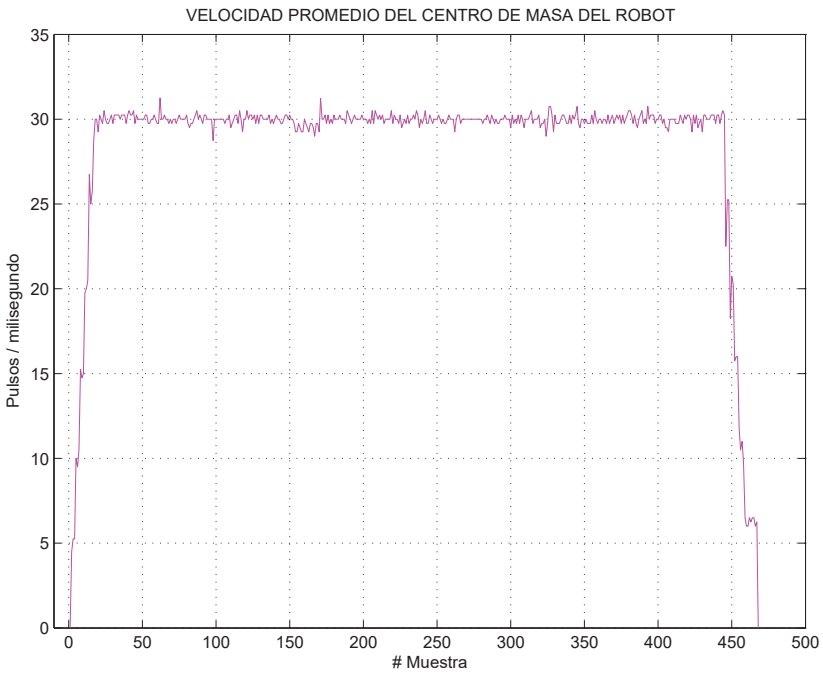


Figura 4.15: Velocidad promedio para la tercer trayectoria.

Error de posición

En este caso, debido a la exigencia de la trayectoria, se obtuvo el mayor error respecto a los experimentos anteriores. En la figura 4.16 se puede observar los puntos alcanzados por el robot móvil. La máxima desviación obtenida con respecto a la trayectoria, se presentó en los últimos dos puntos, sin embargo durante la ejecución de la primera parte de la trayectoria (óvalo) también se observaron desviaciones, solo que en este caso de menor magnitud. A pesar de estos inconvenientes podemos comentar que los resultados obtenidos fueron aceptables, considerando que el robot móvil no logró perderse durante la ejecución del seguimiento de la trayectoria de prueba.

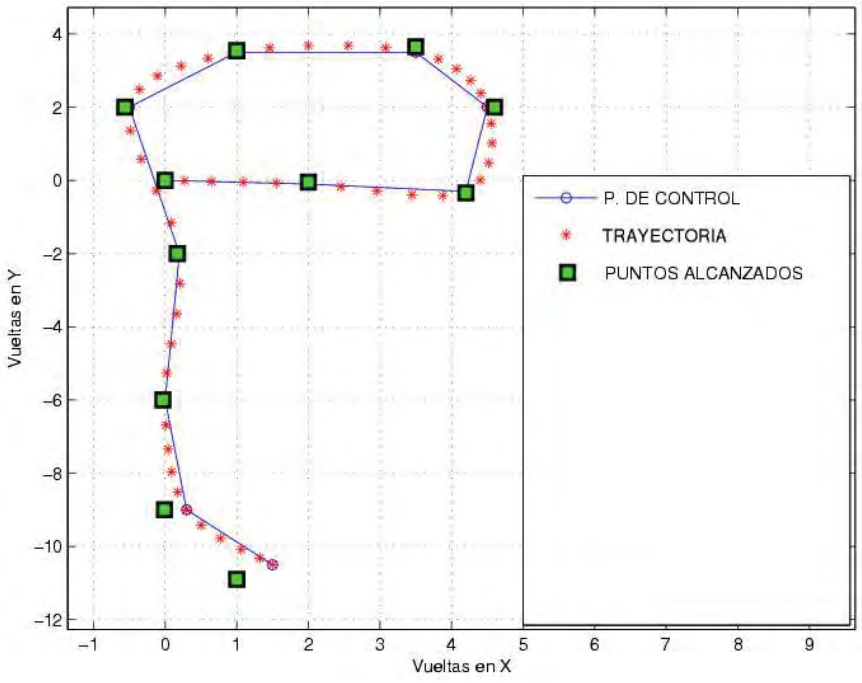


Figura 4.16: Puntos alcanzados para la tercer trayectoria.

4.2. Movimientos Complementarios

En esta sección presentaremos un par de movimientos fáciles de implementar en una configuración de tipo diferencial. Este par de movimientos complementarios pueden ser incorporados al planteamiento del *Spline*, incrementando con esto la habilidad del robot para navegar.

4.2.1. Línea recta

Una línea recta representa una de las trayectorias más sencillas de implementar en una configuración de tipo diferencial. Tan sólo es necesario hacer que las velocidades de ambos motores coincidan, es decir $V_R = V_L$. Para implementar una trayectoria recta, obviamente es necesario conocer la magnitud de la distancia y la velocidad con la que se quiere recorrer esta trayectoria recta. De nueva cuenta así como sucedió con el planteamiento expuesto en la sección anterior, el encoder es útil para lograr este objetivo.

Para encontrar el tiempo de ejecución, es necesario hacer uso de la ecuación 3.30 descrita en la sección 3.5. Psteriormente para encontrar el múltiplo de la interrupción de tiepo real (RTI), es necesario dividir el tiempo calculado entre el valor de la misma interrupción.

Para la ejecución de este experimento se seleccionó una velocidad de 0.6136 $Vueltas/s$ y una recta de magnitud igual 6, recordando que la escala se encuentra en vueltas. De esta manera el cálculo del tiempo es como sigue.

$$t = \frac{6}{0.6136} = 9.7783 \ s. \tag{4.11}$$

El cual al ser dividido entre el valor de la interrupción de tiempo real $2ms$ da como resultado.

$$mrti = \frac{9.7783}{0.002} = 4889 \tag{4.12}$$

En la figura 4.17 se muestra la trayectoria de prueba, la cual corresponde a la linea continua, mientras que la linea discontinua representa la descrita por el robot móvil al realizar el seguimiento de ésta. En ella se puede observar una ligera desviación con respecto al punto final.

El perfil de velocidades mostrado en la figura 4.18, obviamente coincide para este caso, con la velocidad promedio del centro de masa del robot móvil. Esto por su puesto se debe a que ambas velocidades son idénticas.

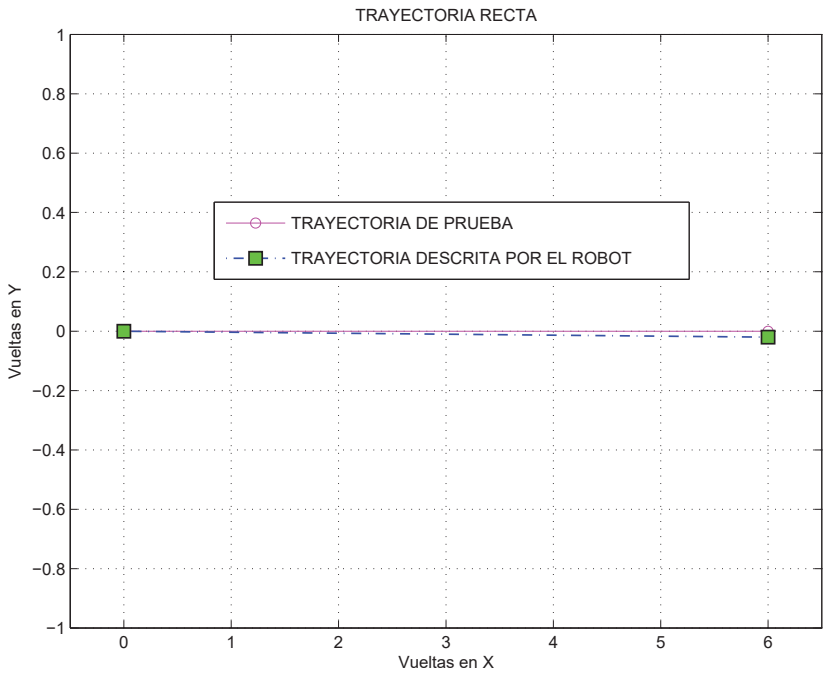


Figura 4.17: Trayectoria recta de prueba.

4.2.2. Giro sobre su centro de masa

En este caso para deducir una expresión que nos permita tener un control sobre el giro o rotación del robot móvil sobre su centro de masa, usaremos la ecuación (2.9), descrita en el capítulo 2 que se rescribe a continuación.

$$\theta = \frac{t}{L}(V_r - V_L) \tag{4.13}$$

Ahora bien, si $V_R = -V_L = V$ lo cual implica un giro sobre su eje, (4.13) se transforma como sigue.

$$\theta = \frac{t}{L}(2V) \tag{4.14}$$

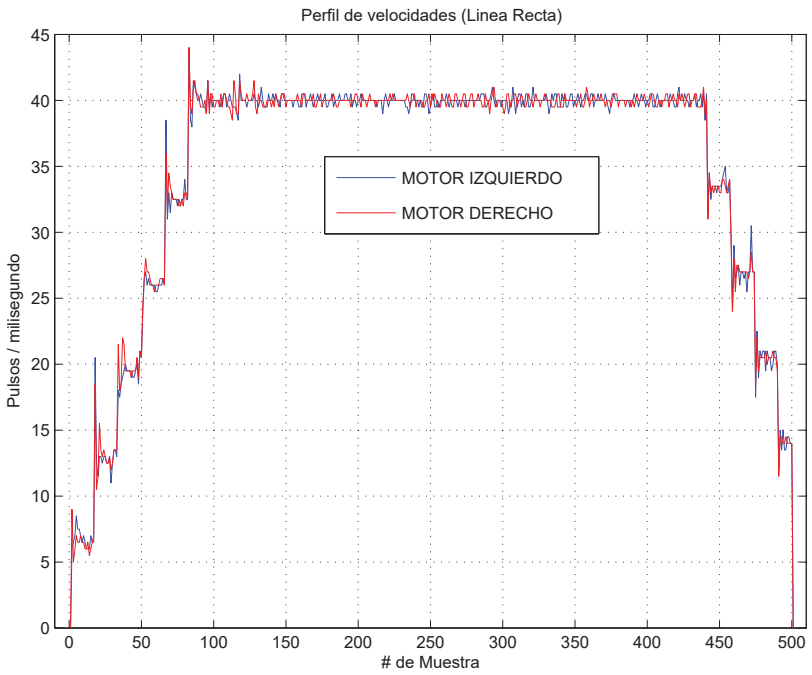


Figura 4.18: Perfil de velocidades (Trayectoria recta)

Despejando t de (4.14) encontramos una expresión que nos permite encontrar el tiempo de ejecución de las velocidades de ambos motores, tales que la estructura del robot móvil gire un ángulo θ con respecto al eje X sobre un marco base.

$$t = \frac{\theta L}{2V} \tag{4.15}$$

Si suponemos que se requiere que el robot gire $\pi/2$ o 90^o con una velocidad $V_R = -V_L = 0.1534\ Vueltas/s$ equivalente a $10\ Pulsos/ms$. De (4.15) tenemos que.

$$t = \frac{\frac{\pi}{2} * 0.95}{2 * 0.1534} = 4863.9\ s. \tag{4.16}$$

Ahora para implementar este tiempo en el microcontrolador es necesario dividir entre el valor de la interrupción de tiempo real (2ms).

$$mrti = \frac{4863.9}{2ms} = 2431.9 \simeq 2432 \tag{4.17}$$

En la figura 4.19 se muestra el perfil de velocidades obtenido al ejecutar el giro de 90^o, mientras que en la figura 4.20 se puede ver la orientación final alcanzada por el robot móvil. Es notorio que en este experimento el único objetivo es lograr una orientación del robot con el eje Y positivo, por lo que el movimiento esperado es una rotación sobre su centro. Para este caso como se puede observar en la figura 4.19 el error obtenido fue de 1°.

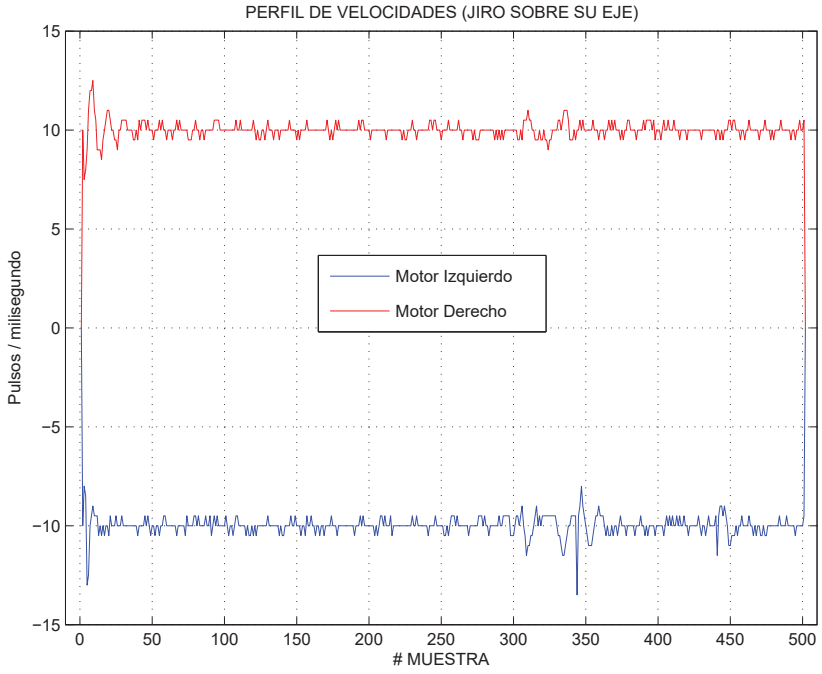


Figura 4.19: Perfil de velocidades (Giro)

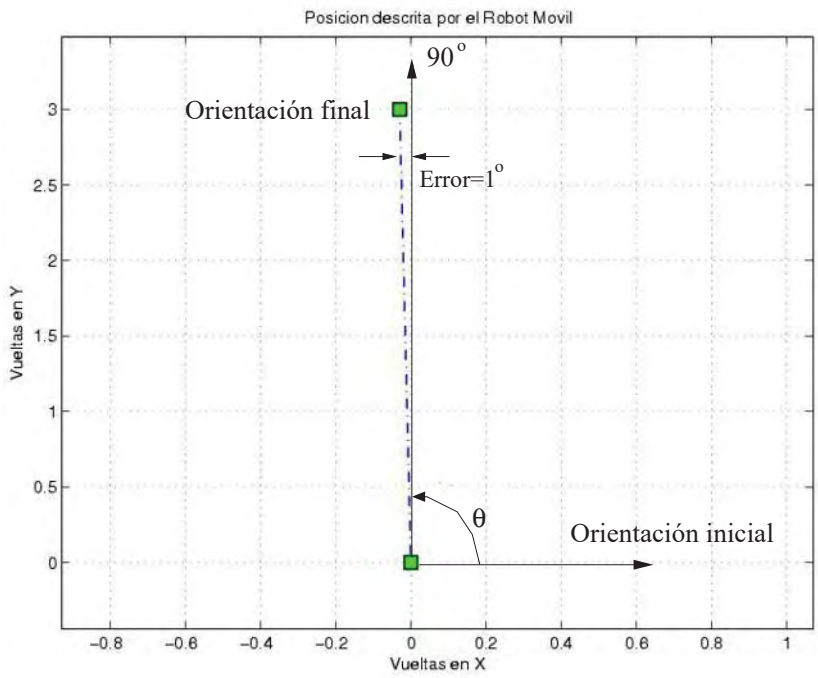


Figura 4.20: Orientación final alcanzada por el robot

Capítulo 5

CONCLUSIONES Y DESARROLLOS A FUTURO

5.1. Conclusiones

El seguimiento de trayectorias es sólo una pieza del gran rompecabezas que representa el diseñar y construir un robot móvil autónomo. La solución al problema de la cinemática inversa propuesta en este trabajo, no es en esencia la solución al problema planteado en la sección 2.2.3. La solución propuesta en esta tesis, hace uso de las ecuaciones básicas de movimiento de una configuración o locomoción diferencial (Ec. 2.1 a 2.4), complementadas con el algoritmo correspondiente a la curva tipo *Spline* de $Catmull - Rom$ expuesta en el capítulo 3. Al final el resultado obtenido consistió en una serie de arcos de círculo, descritos por las ecuaciones recién comentadas y que además son parte de las trayectorias de prueba. Los resultados obtenidos y expuestos en el capítulo anterior, dan una idea clara del alcance cinemático del robot móvil asi como la posibilidad de usar estos resultados en conjunción con otros algoritmos relacionados con la navegación con la finalidad de incrementar las habilidades de navegación del robot móvil.

Las limitaciones mecánicas del prototipo usado para la realización de este trabajo, representaron más que un obstáculo, una buena forma de reconocer algunos detalles que afectan el desempeño del robot móvil. Debido a esta situación se pudo obtener información valiosa para la construcción de versiones futuras (ya existentes). Por ejemplo la

selección y ubicación del resorte de la rueda de apoyo frontal fueron modificadas, con el afán de mejorar la respuesta que se obtuvo en el experimento del escalón. Otro detalle fue el tipo de acoplamiento de los motores con las llantas de tracción del robot móvil, ya que éste provocó en gran medida las variaciones de la velocidad comentadas y observadas en el capítulo anterior así como en las pruebas del escalón.

En resumen, podemos enumerar las aportaciones de esta tesis como sigue:

- Obtención de un *modelo cinemático* referente a una estructura de locomoción diferencial. En el capítulo 2 se describe la construcción de dicho modelo asi como las variables de control referentes a una estructura con estas caracteristicas.
- *Detección de las dinámicas oscilatorias de la estructura del robot móvil.* Esto fue posible mediante la implementación de la prueba del escalón a la estructura del robot móvil, asi como mediante la implementación del sistema de medición de las variables de interés (ver capítulo 2).
- *Implementación del controlador de velocidad PI.* Se diseño, implementó y sintonizó un controlador de velocidad apropiado para el seguimiento de trayectorias del robot móvil (ver capítulo 2).
- *Cálculo e implementación de la curva tipo Spline de Catmull−Rom.* Se Implementó la curva tipo Spline para la interpolación de los puntos de control o puntos por donde se requiere que pase el robot móvil (ver capítulo 3).
- *Estrategia de arranque y frenado.* Se implementaron las rampas de arranque y frenado, que eliminan efectos indeseables que se presentan justamente en estas etapas del seguimiento de trayectorias (ver capítulo 3).

Finalmente todo el trabajo e ideas desarrolladas a lo largo de este trabajo fueron hechas con la finalidad de fusionarlas con el actual proyecto **ROCA** desarrollado en la *Universidad Michoacana de San Nicolás de Hidalgo*, el cual es supervisado por el Dr. Leonardo Romero Muñoz.

5.2. Sugerencias Para Trabajos Futuros

1. La primera sugerencia, tiene que ver con la parte mecánica, la cual consiste en modificar el acoplamiento de los motores con las llantas de tracción, es decir agregar una banda dentada entre los motores y su respectiva llanta. De esta manera la banda puede absorber la fricción no homogénea debida a las imperfecciones mecánicas de las piezas. En la versión del proyecto **ROCA**, ya se encuentra implementada esta idea, sin embargo, esta recomendación puede ser de gran importancia para aquellos que estén interesados en realizar un proyecto similar.

2. El encoder o disco con perforaciones es en gran medida el artífice de la consecución del objetivo planteado en este trabajo, ya que con el se realizó el cálculo y monitoreo de los comandos de velocidad que describen las trayectorias en cuestión. Aunque la manera sencilla en la que se usó en este proyecto arrojó buenos resultados, no está por demás recomendar lo siguiente: diseñar y conectar a los encoder de cada motor un filtro eliminador de ruido o rebotes, producidos en gran medida por las vibraciones naturales del movimiento del robot. Esto con la finalidad de mejorar la información que posteriormente procesará el microcontrolador.

3. El seguimiento de las trayectorias de prueba realizadas en este trabajo, se hicieron en lazo abierto, es decir, no se implementó ningún tipo de retroalimentación (posición y orientación) que nos permitiera reafirmar o corregir la localización del robot. El tema de la localización [Siegwart04] es una buena opción para la implementación de esta sugerencia.

4. El diseño de la trayectoria es otro aspecto que debe de estudiarse más a fondo. Las trayectorias de prueba, usadas para la experimentación con el robot móvil fueron diseñadas únicamente bajo el criterio de su servidor, sin embargo como se comentó oportunamente en el capítulo 3, éstas deben de estar sujetas a las limitaciones dinámicas y cinemáticas del robot móvil, aunque existen otros aspectos que deben de tomarse en cuenta. Un ejemplo de esto es, la trayectoria más segura y que no necesariamente puede ser las más corta [Muñoz95].

5. La discretización de la curva tipo Spline realizada en esta tesis fue hecha de manera empírica y aunque dio buenos resultados, en la literatura [Muñoz95] existen métodos de discretización más elaborados que podrían ser incorporados a esta tesis.

Antes de dar por terminado este capítulo es importante comentar que algunas de las sugerencias mencionadas ya han sido tratadas en el proyecto **ROCA**, como por ejemplo el tema de la localización [Arellano05].

Apéndice A

LABVIEW

A.1. Descripción del Panel de Sintonización del Controlador Utilizado.

El panel de sintonización está dividido en cinco secciones. Éstas se encuentran delimitadas en recuadros con linea discontinua como se puede observar en la figura anterior. En la sección I se puede seleccionar el puerto de comunicación, la velocidad de transmisión y recepción (B/Seg), así como un recuadro de mensajes que recibe información proveniente del microcontrolador. Esta sección también cuenta con un indicador de comunicación establecida (Com OK).

En la sección II el usuario tiene la posibilidad de seleccionar las referencias de velocidad para ambos motores (*Slide* horizontal) así como el número de vueltas también para ambos motores (*Slide* vertical). Esta misma sección cuenta con cuatro botones, que son los encargados de ejecutar los comandos de velocidad, posición, paro repentino del robot y de la adquisición de las muestras de velocidad. Más adelante comentaremos la secuencia de ejecución de estos comandos, para un buen desenvolvimiento de la interfaz.

La sección III cuenta con un botón de encendido que habilita y deshabilita el panel de sintonización. También en esta misma sección se encuentra un *Combo Box* con cuatro modos de operación del panel, los cuales describiremos a continuación.

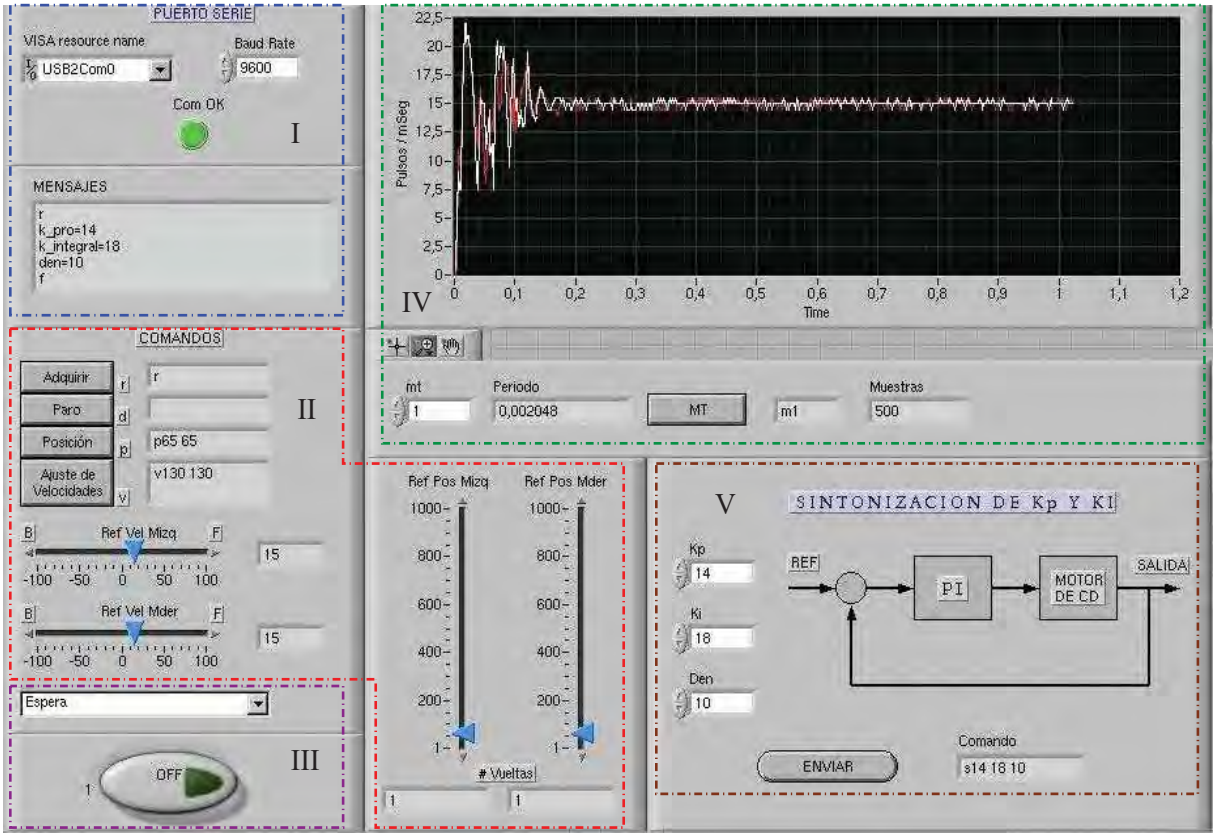


Figura A.1: Panel de control y sintonización.

Nueva adquisición: Este modo de operación permite al usuario iniciar una adquisición, la cual incluye la selección de las velocidades, el número de vueltas, la ejecución de estos comandos así como la graficación de las muetras de velocidad.

Cargar y graficar datos ya adquiridos: En este modo de operación como su nombre bien lo describe, el usuario puede cargar y graficar datos obtenidos en experimentaciones anteriores.

Sintonización: En esta opción el usuario tiene la posibilidad de seleccionar las constantes del controlador (Kp/Den, Ki/Den), de modo que obtenga los resultados que mejor se apeguen a sus objetivos. Es importante comentar que este modo de operación funciona una vez realizado un primer experimento. Este modo habilita la sección en café, la cual

comentaremos más adelante.

Espera: Esta opción mantiene funcionando el panel mientras es seleccionado cualquiera de los modos anteriores.

La sección IV consta de un *Waveform Graph* donde se grafican las velocidades adquiridas una vez terminado un experimento, así como dos controladores donde se selecciona y envía el periodo de muestreo. Este último se selecciona en múltiplos de la interrupción de tiempo real, que para este caso se seleccionó de $1mSeg.$, por lo que este valor es el mínimo con el que se puede muestrear.

Finalmente la sección V permite al usuario como ya comentamos anteriormente seleccionar y enviar al microcontrolador a través del puerto serie, las constantes deseadas con las que se quiere realizar una nueva adquisición.

A.2. Descripción del Panel de Seguimiento de Trayectorias.

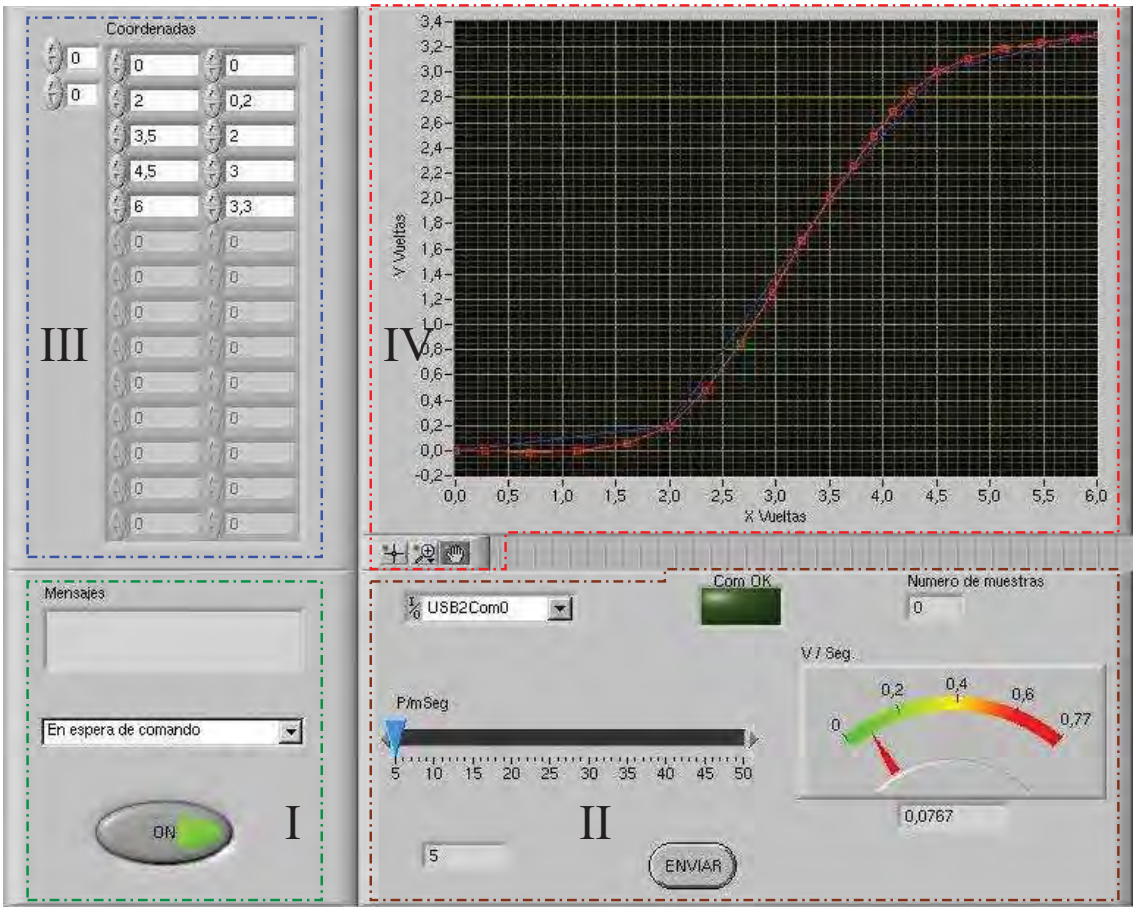


Figura A.2: Panel de control de seguimiento de trayectorias.

El panel de control de seguimiento de trayectorias mostrado en la figura A.2 está dividido en cuatro secciones delimitadas por recuadros con linea discontinua.

La seccción I cuenta con un boton que habilita todas las funciones del panel. También cuenta con un *Combo Box*, el cual contiene cuatro modos de operación del panel, los cuales describiremos a continuación.

En espera de comando: Este modo de operación mantiene activo el panel, hasta el momento en que se selecciona cualquiera de los modos que citaremos a continuación.

Iniciar comunicación: En esta opción, el usuario puede cerciorarse que la comunicación con el microcontrolador mediante el puerto serie es correcta. Esto se realiza haciendo un reset en el microcontrolador, el cual envia un mensaje de inicialización.

Construir Spline y Arranque: Este modo de operación construye el Spline correspondiente a los puntos ingresados en los controles, que se encuentran en la sección III. Posteriormente habilita los controles correspondientes a ls sección II, los cuales permiten que el usuario seleccione la velocidad referente al centro de masa del robot móvil, con la que a su vez se recorrera la trayectoria.

Graficar y guardar datos: En este modo de operación se gráfica y almacena los datos correspondientes al experimento realizado en el modo anterior. Para esto se hace uso de un *Wavefom Graph*, el cual se muestra en la sección IV.

Apéndice B

IMPLEMENTACIÓN DE LA CURVA TIPO SPLINE DE CATMULL-ROM

El codigo que se presenta a continuación fué escrito en *MATLAB*.

```
% crspline.m
% dibuja la curva de interpolacion basada en Splines
% de Catmull-Rom de clase G1 (Continuidad geometrica) para los puntos dados por la
% matriz de coordenadas de n x 2 (n debe ser mayor de 3)
% usando beta como parametro de forma.
function crspline(V,beta)
plot(V(:,1),V(:,2),'b-*')
hold on
aux=size(V);
n=aux(1,1)-3;    %numero de tramos
b=beta;
u=0:0.2:1;       %Delta_u=Incremento
m=length(u);
for i=1:n
    fi1=-b*(u.^3-2*u.^2+u)/(b+1);
    fi2=((b^2+b+1)*u.^3-(2*b^2+2*b+1)*u.^2+(b^2-1)*u+b+1)/(b+1);
    fi3=-((b^2+b+1)*u.^3-(2*b^2+b+1)*u.^2-b*u)/(b*(b+1));
    fi4=(u.^3-u.^2)/(b*(b+1));
    qx=V(i,1)*fi1+V(i+1,1)*fi2+V(i+2,1)*fi3+V(i+3,1)*fi4;
    qy=V(i,2)*fi1+V(i+1,2)*fi2+V(i+2,2)*fi3+V(i+3,2)*fi4;
    plot(qx,qy,'r*')
end
grid on
```

Apéndice C

PROGRAMAS PARA EL MICROCONTROLADOR

C.1. Rutinas Para el Escalón.

main.c

```
#include  <stdio.h>
#include  <string.h>
#include  "ioRegsHcs12DP256.h"
#include  "serial.h"
#include  "command.h"
#include  "control.h"

int                main(void);

extern char             _start_of_data;
extern char             _end_of_data;
extern const char       _start_of_init;

unsigned short int        start = 0x0000;
unsigned short int        end   = 0xffff;

#define BAUD57600  26      //(baud) 19200*3 baud with MCLK=24Mhz
#define BAUD28800  52      //(baud) 9600*3 baud with MCLK=24Mhz
#define BAUD14400 104      //(baud) 4800*3 baud with MCLK=24Mhz
#define BAUD9600 156  //(baud) 156 = 208 * 7200/9600 with MCLK=24Mhz
#define BAUD7200 208      //(baud) 2400*3 baud with MCLK=24Mhz

static void inline disableCOP(void)
```

```
{
   COPCTL = 0x00; //Disable COP
   COPCTL = 0x00;
}

#define OscClk 16000000 // Crystal or oscillator frequency
#define Eclock   24000000 // final E-clock frequency (PLL)
#define RefClock  8000000 // frequency used by the PLL to generate E-clock
#define REFDVVal (OscClk/RefClock)-1  //value for REFDV Register
#define SYNRVal (Eclock/RefClock)-1  //value for SYNR Register
static void inline rise24Mhz(void)
{
   REFDV = REFDVVal; //PLLCLK = 24MHz
   SYNR  = SYNRVal;
   asm("nop\n" "nop\n" "nop\n" "nop\n"); //nops required for bug
#define LOCK 0x08
   while(!(CRGFLG & LOCK)) //wait here till the PLL is locked
      ;
#define PLLSEL 0x80
   CLKSEL |= PLLSEL;  //System clocks are derived from PLLCLK

}

#define rtimask         b(0,1,0,0,0,0,0,1) //(1/16Mhz)2*2^13=1.024 msec interrupt
#define RTIF            b(1,0,0,0,0,0,0,0)
#define RTIE            b(1,0,0,0,0,0,0,0)

static void inline set_rti()
{
   RTICTL = rtimask;  //Initialize Real Time Interrupt rate
   CRGFLG |= RTIF; //clear flag
   CRGINT |= RTIE;  //Enable RTI
}

volatile extern short g_command_pending;
volatile extern short g_command_len;
volatile extern char g_command_line[];
volatile extern unsigned short g_timer;

static char command;
static short num_params;
static unsigned short param[20];


static decode_command()
{
```

```
   static int inicio_num, i;
   static char c;

   command=g_command_line[0];
   num_params=0;
   param[0]=0;
   inicio_num=0;
   for(i=1; (c=g_command_line[i]); i++) {
      if(c=='\r')
         continue;
      if(c==' ' || c == '\n') {
         param[++num_params]=0;
      } else {
         inicio_num=1;
         param[num_params] *= 10;
 param[num_params] += c - '0';
      }
   }
   if(!inicio_num)
      num_params--;
   g_command_len=0;
}


int  main()
{
   static int i;
   asm("sei"); //Disable Any interrupts

   disableCOP(); //Disable COP
   rise24Mhz(); //PLL Clock to 24Mhz
   memcpy(&_start_of_data, &_start_of_init, &_end_of_data - &_start_of_data);
   ser0_init(BAUD9600); //Baudrate is 9600
   //set_rti(); //Initialize Real Time Interrupt rate
   init_control();


   asm("cli");      //unmask interrupts habilita interrupciones

   ser0_puts("Please type I to star sensing\n"); //Manda mensaje de inicio
    //por el P_serie a B9600

   while(1){
     if(g_command_pending){
       g_command_pending=0;
       decode_command(); //Decodifica el comando tecleado ver->main.c
```

```
      do_command(command,num_params,param); //Ejecuta el comando tecleado ver->command.c
      g_command_pending=0;
    }
    check_end_commands(); //Ver->command.c
  }

  return  0;
}
```

command.c

```
#include  <stdio.h>
#include  <string.h>
#include  "ioRegsHcs12DP256.h"
#include  "serial.h"
#include  "control.h"

volatile extern unsigned short g_timer;

/* *** WAIT operation **** */
static short edo_wait=0;

static void inline wait(short num_params, unsigned short param[])
{
  switch (num_params) {
     case 1: g_timer = param[0]; edo_wait=1; break;
  }
}

static void inline check_end_wait()
{
  if(edo_wait && !g_timer){
     ser0_puts("w\n");
     edo_wait=0;
  }
}
/* *** End WAIT operation **** */

/* *** End QUERY operation **** */
void inline query(void)
{
   static unsigned short p0,p1;

#define ICLAT b(0,0,0,1,0,0,0,0)

   MCCTL |= ICLAT;  //Force the contents of 8-bit accumulators
    //to be latched into their
//associated holding registers
   p0=PA2H; //Canal A del motor Izquierdo
   p1=PA3H;     //Canal A del motor Derecho
   ser0_puts("p0=");
   ser0_putushort(p0);
   ser0_puts(" p1=");
   ser0_putushort(p1);
   ser0_putnl();
```

```
}

/* *** End WAIT operation **** */


void check_end_commands(void)
{
   check_end_wait();
}

void do_command(char command, int num_params, unsigned short param[])
{
   switch (command) {
      case 'w': wait(num_params, param);  break;
      case 'q': query();  break; //Proporciona datos adquiridos ver->comand.c
      case 'f': forward(num_params,param);  break;
      case 'b': backward(num_params,param);  break;
      case 'l': left(num_params,param);  break;
      case 'r': right(num_params,param);  break;
      case 'v': set_vel(num_params,param);  break;
      case 'd': stop();  break;
      case 'I': inicio(); break; // Comando de arranque motor y sensado ver->control.c //
   }
   check_end_commands();
}
```

control.c

```
#include "ioRegsHcs12DP256.h"
#include "serial.h"

//       PORT ASSIGNMENTS for this program for adapt9S12Dp256

//                                              pata    Conector
//       PT0     Echo Sonar      ;I             13              (H1)    4.7kohms
//       PT1     MIzqI           ;I             12
//       PT2     MIzqA           ;I             11                      4.7kohms
//       PT3     MDerA           ;I             10                      4.7kohms
//       PT4     MDerI           ;I              9                      4.7kohms
//       PT5     MIzqB           ;I              8
//       PT6     MderB           ;I              7
//               (red led on the board)
//       PT7     OC7             ;O              6

//       PP0     MIzqDir         ;O             21              (H1)
//       PP1     MIzqBrake       ;O             20
//       PP2     MDerDir         ;O             19
```

```
//       PP3     MDerBrake       ;O              18
//       PP4     InitSO          ;O              17
//       PP5     BinhSO          ;O              16
//       PP6     MIzqPWM         ;O              15
//       PP7     MDerPWM         ;O              14

//       PH0     Line A Sonar    ;O              42              (H1)
//       PH1     Line B Sonar    ;O              41
//       PH2     Line C Sonar    ;O              40
//       PH3     Line D Sonar    ;O              39
//       PH4     Spare           ;I              38
//       PH5     Spare           ;I              37
//       PH6     Spare           ;I              36
//       PH7     Spare           ;I              35

// Masks
#define mT_EchoSonar    b(0,0,0,0,0,0,0,1) // PORTT
#define mT_MIzqI        b(0,0,0,0,0,0,1,0)
#define mT_MIzqA        b(0,0,0,0,0,1,0,0)
#define mT_MDerA        b(0,0,0,0,1,0,0,0)
#define mT_MDerI        b(0,0,0,1,0,0,0,0)
#define mT_MIzqB        b(0,0,1,0,0,0,0,0)
#define mT_MDerB b(0,1,0,0,0,0,0,0)

#define mP_MIzqDir      b(0,0,0,0,0,0,0,1) // PORTP
#define mP_MIzqBrake    b(0,0,0,0,0,0,1,0)
#define mP_MDerDir      b(0,0,0,0,0,1,0,0)
#define mP_MDerBrake    b(0,0,0,0,1,0,0,0)
#define mP_InitSO       b(0,0,0,1,0,0,0,0)
#define mP_BinhSO       b(0,0,1,0,0,0,0,0)
#define mP_MIzqPWM b(0,1,0,0,0,0,0,0)
#define mP_MDerPWM b(1,0,0,0,0,0,0,0)

#define mH_SonarDirA    b(0,0,0,0,0,0,0,1)       // PORTH
#define mH_SonarDirB    b(0,0,0,0,0,0,1,0)
#define mH_SonarDirC    b(0,0,0,0,0,1,0,0)
#define mH_SonarDirD    b(0,0,0,0,1,0,0,0)

static unsigned char vel_izq=50, vel_der=50;

void set_vel(int num_param, unsigned short param[])
{
   vel_izq=param[0];
   vel_der=param[1];
}
```

```
static void inline set_vel_izq()
{
   PWMDTY6 = vel_izq;
}

static void inline set_vel_der()
{
   PWMDTY7 = vel_der;
}

static void m_izq_forward()
{
   PORTP |= mP_MIzqDir;
   PORTP &= ~mP_MIzqBrake;

   set_vel_izq();
}

static void m_izq_backward()
{
   PORTP &= ~mP_MIzqDir;
   PORTP &= ~mP_MIzqBrake;

   set_vel_izq();
}

static void m_izq_stop()
{
   PORTP |= mP_MIzqDir;
   PORTP |= mP_MIzqBrake;

   set_vel_izq(255U);

}

static void izq_forward()
{
   m_izq_forward();
}

static void izq_backward()
{
   m_izq_backward();
}

static void m_der_forward()
```

```
{
   PORTP &= ~mP_MDerDir;
   PORTP &= ~mP_MDerBrake;
   set_vel_der();
}

static void m_der_backward()
{
   PORTP |=  mP_MDerDir;
   PORTP &= ~mP_MDerBrake;
   set_vel_der();
}

static void m_der_stop()
{
   PORTP |= mP_MDerDir;
   PORTP |= mP_MDerBrake;

   set_vel_der(255U);

}

static void der_forward()
{
   m_der_forward();
}

static void der_backward()
{
   m_der_backward();
}


void forward(int num_param, unsigned short param[])
{
  izq_forward();
  der_forward();
}

void backward(int num_param, unsigned short param[])
{
  izq_backward();
  der_backward();
}

void left(int num_param, unsigned short param[])
```

```
{
  izq_backward();
  der_forward();
}

void right(int num_param, unsigned short param[])
{
  izq_forward();
  der_backward();
}

void stop()
{
   m_izq_stop();
   m_der_stop();
}

void init_control()
{
   DDRP = 0xff;         //all are output
   PORTP = 0;

   PWME |= mP_MIzqPWM | mP_MDerPWM;     // Pulse Width Channel 6 & 7 Enable (use Clock B)
   PWMPRCLK = b(0,0,1,0,0,0,0,0);       //Clock B=Bus clock/4
                                        //1/(1/24e6 * 4 *256))=23437Hz

//Initialize Analog To Digital
   ATD0CTL2 = 0x80; //enable ATD (1,0,0,0,0,0,0,0) ADPU=1 ->Normal ATD functionality
   ATD0CTL3 = 0x40; //(0,1,0,0,0,0,0,0) S8C=1,S4C=0,S2C=0,S1C=0 Number of Conversions
        //per Sequence=8
   ATD0CTL4 = 0xE0; //Select Sample rate (1,1,1,0,0,0,0,0) 8 bit resolution 16A/D conversion
        //and divide by 2
   ATD0CTL5 = 0xB0; //Select 8 channel mode, Continuous scan (1,0,1,1,0,0,0,0)
        //justificacion izq, Unsigned,


   ICPAR |= b(0,0,0,0,1,1,0,0); //Pulse Acumulator 3 and 2 are enabled
   DLYCT |= b(0,0,0,0,0,0,0,1); //Delay Counter = 256 bus clock cycles
   TIOS  |= b(0,0,0,0,0,0,0,0); //All channels are input capture
   TCTL4 |= b(1,1,1,1,0,0,0,0); //Input Capture Edge Control
//1,1=Capture on any edge, rising or falling (encoders)
   ICSYS |= b(0,0,0,0,0,1,1,1); //PACMX=1 When the 8-bit pulse accumulator has reached the value
//$FF, it will no be incremented further.
//BUFEN=1 Input Capture and pulse accumulators registers are enabled
//LATQ=1 Latch Mode of Input Capture is enabled
```

```
   PWMPOL = 0xff; // PWM outputs are high at the beginning of the period, then goes
// low when the duty count is reached
   m_izq_stop();
   set_vel_izq(255U);
   m_der_stop();
   set_vel_der(255U);
}

//*********************Función agregada*****************************//
//DESCRIPCIÓN: Esta función tiene como finalidad iniciar el proceso de sensado mediante
//un comando 'I' tecleado en la PC y enviado al microcontrolador por el puerto serie.
//Inicializa la interrupción de tiempo real (cada 1mSeg.) e inicia el arranque del motor
//en el esquema PWM a un 100%, con la finalidad de aplicar el escalon al motor.
#define NPULSOS 150
volatile extern unsigned short im;
volatile extern unsigned short pulsos_der[NPULSOS];
volatile extern unsigned short pulsos_izq[NPULSOS];
volatile extern unsigned short I_arranque[NPULSOS];

#define MAXVEL 255 //24 Volt
#define rtimask         b(0,1,0,0,0,0,0,1) //(1/16Mhz)2*2^13=1.024 msec interrupt
//#define rtimask b(0,1,0,1,0,0,0,1) //(1/16Mhz)2*2^14=2.048 msec interrupt
#define RTIF            b(1,0,0,0,0,0,0,0)
#define RTIE            b(1,0,0,0,0,0,0,0)

void inicio()
{
unsigned short index;
   im=0;
   RTICTL = rtimask;  //Initialize Real Time Interrupt rate (Ver: CRG Block User Guide Pag. 24)
   CRGFLG |= RTIF; //clear flag
   CRGINT |= RTIE;  //Enable RTI

   PWMDTY6 = MAXVEL; //PWM al 100% Mizq
   PWMDTY7 = MAXVEL; //PWM al 100% Mder

   PORTP &= ~mP_MDerDir;
   PORTP &= ~mP_MDerBrake;

   PORTP |= mP_MIzqDir; //Dir=1
   PORTP &= ~mP_MIzqBrake; //Brake=0


   while(im<NPULSOS); //Espera hasta que se almacenan los datos sensados (150 datos)

   CRGINT &=  b(0,1,1,1,1,1,1,1);  //Deshabilita la interrupcion de tiempo real (RTI)
```

```
   m_izq_stop();    //Detiene motor
   m_der_stop();

//ser0_puts("Pulsos Mder\n"); //Manda mensaje de inicio por el P_serie a B9600
   for (index=0; index<NPULSOS; index++){
    ser0_putushort(pulsos_der[index]);
ser0_puts("\n");
}

    //ser0_puts("Pulsos Mizq\n"); //Manda mensaje de inicio por el P_serie a B9600
   for (index=0; index<NPULSOS; index++){
    ser0_putushort(pulsos_izq[index]);
ser0_puts("\n");
}

}
```

isr.c

```
#include "ioRegsHcs12DP256.h"

#include  "isr.h"

void  __attribute__ ((interrupt)) ISR_Empty(void)
{
}

#define NPULSOS 150
unsigned short g_timer=0;
unsigned short im=NPULSOS;
unsigned short int pulsos_der[NPULSOS];
unsigned short int pulsos_izq[NPULSOS];
unsigned short int I_arranque[NPULSOS];

unsigned short g_ad2=0;
//unsigned short g_ad3=0;
//unsigned short g_ad4=0;
//unsigned short g_ad5=0;

#define RTIF b(1,0,0,0,0,0,0,0)
#define ICLAT b(0,0,0,1,0,0,0,0)
#define SCFflag         b(1,0,0,0,0,0,0,0)       //SCF - Sequence Complete flag
void  __attribute__ ((interrupt)) ISR_RealTimeInt(void)
{
   CRGFLG |= RTIF;      //clear flag
   asm("pshc\n pshd\n; pshx\n pshy");
   if(g_timer)
```

```
      g_timer--;
   asm("pulc\n puld\n; pulx\n puly");

   if (im <  NPULSOS ){
    MCCTL |= ICLAT; //Force the contents of 8-bit accumulators to be latched into their
//associated holding registers (PA2H Y PA3H en este caso)
pulsos_izq[im] = PA2H; //Almacena la lectura de pulsos (PT2) Canal A del encoder
pulsos_der[im] = PA3H;

im++;
   }

}
```

serial.c

```
#include "ioRegsHcs12DP256.h"

/* This assumes a 8 MHz clock. If not, you'll need to change the
   Baud rate enum.
 */

#define scimask  b(0,0,1,0,1,1,0,0) //RIE - SCI Interrupt enable ;RE - Receiver Enable


void ser0_init (unsigned short baud)
{
   SCI0BD = baud;
   /* enable transmitter and receiver and receiver interrupt */
   SCI0CR1 = 0;
   SCI0CR2 = scimask;
   SCI0SR1; //read register to clear flag RDRF
   SCI0DRL; //dummy read to flush receive buffer
}

#define TDRE 0x80
void ser0_putchar(char c)
{
   while ((SCI0SR1 & TDRE) == 0)
      ;
   SCI0DRL = c;
}

void ser0_puts(char s[])
{
   while(*s){
      ser0_putchar(*s);
      s++;
   }
}

void ser0_putnl()
{
  ser0_putchar('\n');
}

#define NDIGITS 5
void ser0_putushort(unsigned short x)
{
   static byte d[NDIGITS];
```

```
   static int i;

   i = NDIGITS -1;
   do {
      d[i--] = x % 10;
      x /= 10;
   } while (x);
   for(i++; i < NDIGITS; i++)
      ser0_putchar(d[i] + '0');
}

volatile short g_command_pending;
volatile char g_command_line[256];
volatile int  g_command_len=0;

void  __attribute__ ((interrupt)) ISR_SerialInput0(void)
{
   asm("pshc\n pshd\n pshx\n pshy");
   static char c, new_command=0, i=0;

   SCI0SR1;      //read register to clear flag RDRF
   c = SCI0DRL;       //read receive buffer

   g_command_line[g_command_len++]=c;
   if(c=='\n'){
      g_command_line[g_command_len++]=0;
//      ser0_puts(g_command_line);
      g_command_pending=1;
   }
//   ser0_putchar(c);

   asm("pulc\n puld\n pulx\n puly");
}
```

C.2. Rutinas Para la Sintonización del Controlador Utilizado.

```
/* command.c */

#include  <stdio.h>
#include  <string.h>
#include  "ioRegsHcs12DP256.h"
#include  "serial.h"
#include  "control.h"

volatile extern unsigned short g_timer;
volatile extern unsigned short g_ad2;
volatile extern unsigned short g_ad3;
volatile extern unsigned short g_ad4;
volatile extern unsigned short g_ad5;
volatile extern unsigned char  g_left_vel;
volatile extern unsigned char  g_right_vel;

volatile extern unsigned long  k_pro;
volatile extern unsigned long  k_integral;

/* *** WAIT operation **** */
static short edo_wait=0;

static void inline wait(short num_params, unsigned short param[])
{
  switch (num_params) {
     case 1: g_timer = param[0]; edo_wait=1; break;
  }
}

static void inline check_end_wait()
{
  if(edo_wait && !g_timer){
     ser0_puts("w\n");
     edo_wait=0;
  }
}
/* *** End WAIT operation **** */

/* *** QUERY operation **** */
void query(void)
{
   ser0_puts("vel_izq=");
   ser0_putushort(g_left_vel);
```

```
   ser0_puts(" vel_der=");
   ser0_putushort(g_right_vel);
   ser0_putnl();
}
/* *** End QUERY operation **** */

/* *** bateries operation **** */
void bateries(void)
{
   static unsigned short bat12,bat24,bat12M,bat24M;

   bat12  = (g_ad2 * (127200L/639)) / 10;
   bat24  = (g_ad3 * (259000L/691)) / 10;
   bat12M = (g_ad4 * (130100L/691)) / 10;
   bat24M = (g_ad5 * (261000L/696)) / 10;
   ser0_putchar('b');
   ser0_putushort(bat12);
   ser0_putchar(' ');
   ser0_putushort(bat24);
   ser0_putchar(' ');
   ser0_putushort(bat12M);
   ser0_putchar(' ');
   ser0_putushort(bat24M);
   ser0_putnl();
/*
   ser0_putchar('c');
   ser0_putushort(g_ad2);
   ser0_putchar(' ');
   ser0_putushort(g_ad3);
   ser0_putchar(' ');
   ser0_putushort(g_ad4);
   ser0_putchar(' ');
   ser0_putushort(g_ad5);
   ser0_putnl();
*/
}
/* *** End QUERY operation **** */

void check_end_commands(void)
{
   check_end_wait();
   check_stop_motors();
}

void do_command(char command, int num_params, unsigned short param[])
{
```

```
   switch (command) {
      case 'w': wait(num_params, param);  break;
      case 'q': query();  break;
      case 'v': set_vel(num_params,param);  break;
      case 'd': stop();  break;
      case 't': bateries(); break;
      case 'k': set_ac(num_params,param); break;
      case 'a': forward(num_params,param); break;
      case 'z': beep(num_params,param); break;
      case 'r': inicio(); break; // Envio de datos
      case 's': sintoniza(num_params,param); break; // Sintonizacion de las ganancias del PI//
      case 'p': posicion(num_params,param); break; //Numero de vueltas o fraccion de vuelta
      case 'm': rti_mul(num_params,param); break; //Multiplo de rti para muestreo
   }
   check_end_commands();
}
```

```
/// control.c ///

#include "ioRegsHcs12DP256.h"
#include "serial.h"

//       PORT ASSIGNMENTS for this program for adapt9S12Dp256

//                                              pata            Conector
//       PT0     Echo Sonar      ;I             13              (H1)    4.7kohms
//       PT1     MIzqI           ;I             12
//       PT2     MIzqA           ;I             11                      4.7kohms
//       PT3     MDerA           ;I             10                      4.7kohms
//       PT4     MDerI           ;I              9                      4.7kohms
//       PT5     MIzqB           ;I              8
//       PT6     MderB           ;I              7
//               (red led on the board)
//       PT7     OC7             ;O              6

//       PP0     MIzqDir         ;O             21              (H1)
//       PP1     MIzqBrake       ;O             20
//       PP2     MDerDir         ;O             19
//       PP3     MDerBrake       ;O             18
//       PP4     InitSO          ;O             17
//       PP5     BinhSO          ;O             16
//       PP6     MIzqPWM         ;O             15
//       PP7     MDerPWM         ;O             14

//       PH0     Line A Sonar    ;O             42              (H1)
//       PH1     Line B Sonar    ;O             41
//       PH2     Line C Sonar    ;O             40
//       PH3     Line D Sonar    ;O             39
//       PH4     Spare           ;I             38
//       PH5     Spare           ;I             37
//       PH6     Spare           ;I             36
//       PH7     Spare           ;I             35

// Masks
#define mT_EchoSonar    b(0,0,0,0,0,0,0,1) // PORTT
#define mT_MIzqI        b(0,0,0,0,0,0,1,0)
#define mT_MIzqA        b(0,0,0,0,0,1,0,0)
#define mT_MDerA        b(0,0,0,0,1,0,0,0)
#define mT_MDerI        b(0,0,0,1,0,0,0,0)
#define mT_MIzqB        b(0,0,1,0,0,0,0,0)
#define mT_MDerB b(0,1,0,0,0,0,0,0)

#define mP_MIzqDir      b(0,0,0,0,0,0,0,1) // PORTP
```

```
#define mP_MIzqBrake    b(0,0,0,0,0,0,1,0)
#define mP_MDerDir      b(0,0,0,0,0,1,0,0)
#define mP_MDerBrake    b(0,0,0,0,1,0,0,0)
#define mP_InitSO       b(0,0,0,1,0,0,0,0)
#define mP_BinhSO       b(0,0,1,0,0,0,0,0)
#define mP_MIzqPWM b(0,1,0,0,0,0,0,0)
#define mP_MDerPWM b(1,0,0,0,0,0,0,0)

#define mH_SonarDirA    b(0,0,0,0,0,0,0,1)       // PORTH
#define mH_SonarDirB    b(0,0,0,0,0,0,1,0)
#define mH_SonarDirC    b(0,0,0,0,0,1,0,0)
#define mH_SonarDirD    b(0,0,0,0,1,0,0,0)
#define NPULSOS 500

volatile extern unsigned short g_left_vel, g_right_vel;
volatile extern unsigned int im; //Numero de muestras adquiridas
volatile unsigned short mt=1; //Multiplo de rti para muestreo (mt*2mSeg.)
volatile static short des_left_vel_clicks;  // Desired vel, clicks/ms
volatile static short des_right_vel_clicks;  // Desired vel, clicks/ms
volatile static short des_bias_clicks;  // Desired bias, clicks/ms
volatile static short max_left_vel_clicks;  // Initial desired bias, clicks/ms
volatile static short max_right_vel_clicks;  // Initial desired bias, clicks/ms
volatile static short max_bias_clicks; // Initial desired bias, clicks/ms
volatile static unsigned char left_pwm, right_pwm;  //Power to motors
volatile static short  integral_left=0; // Integral of velocity difference
volatile static short  integral_right=0; // Integral of velocity difference
volatile static short prop_left=0;
volatile static short prop_right=0;
volatile static short E_left=0;
volatile static short E_right=0;
//***************-Constantes iniciales para el control-*********************//
volatile static long k_integral = 18; // Integral gain
volatile static long k_pro = 14; // Proportional gain
volatile static long den=10;  // Denominador
//************************************************************************//
//******************-Numero de vueltas iniciales-************************//
volatile static int des_posI = 65; // Vueltas iniciales ("antes de la caja de engranes")
volatile static int des_posD = 65; // Ambos motores  65V_E = 1V_OUT
//************************************************************************//
//*******************-Variables para el control ded posicion-**************//
unsigned short flagI=0;
unsigned short flagD=0;
volatile extern unsigned int VI;
volatile extern unsigned int VD;
unsigned int Total_posI=0;
unsigned int Total_posD=0;
```

```
volatile extern unsigned long Index_MI; //Index_MI e Index_MD cuentan hasta 2exp32
volatile extern unsigned long Index_MD;
unsigned short FI=1;
unsigned short FD=1;
//*********************************************************************//

volatile short edo_left_motor=0;
volatile short edo_right_motor=0;
volatile short k_vel_ini = 5;
volatile short k_vel_end = 30;
volatile short k_ms_inc  = 50;

volatile unsigned long clicks_stop, clicks_break, total_left_clicks, total_right_clicks;
volatile static unsigned short acelerando, frenando;
volatile static short timer_ac;

static void inline set_left_pwm()
{
   PWMDTY6 = left_pwm;
}

static void inline set_right_pwm()
{
   PWMDTY7 = right_pwm;
}

static void m_izq_forward()
{
   PORTP |= mP_MIzqDir;
   PORTP &= ~mP_MIzqBrake;
}

static void m_izq_backward()
{
   PORTP &= ~mP_MIzqDir;
   PORTP &= ~mP_MIzqBrake;
}

static void m_izq_stop()
{
   PORTP |= mP_MIzqDir;
   PORTP |= mP_MIzqBrake;
   left_pwm=255U;
   set_left_pwm();
}
```

```
static void m_der_forward()
{
   PORTP &= ~mP_MDerDir;
   PORTP &= ~mP_MDerBrake;
}

static void m_der_backward()
{
   PORTP |=  mP_MDerDir;
   PORTP &= ~mP_MDerBrake;
}

static void m_der_stop()
{
   PORTP |= mP_MDerDir;
   PORTP |= mP_MDerBrake;
   right_pwm=255U;
   set_right_pwm();
}

void set_velocity()
{
   if(!max_left_vel_clicks && !max_right_vel_clicks && !max_bias_clicks){
      m_izq_stop();
      m_der_stop();
      edo_left_motor=edo_right_motor=0;
      acelerando = 0;
      frenando   = 0;
      return;
   }
   left_pwm=right_pwm=0U;          // Initial power to motors
   edo_left_motor=edo_right_motor=1;
   integral_left=0;
   integral_right=0;
   set_left_pwm();
   set_right_pwm();
   des_left_vel_clicks = abs(max_left_vel_clicks);
   des_right_vel_clicks = abs(max_right_vel_clicks);
   des_bias_clicks = max_bias_clicks;
   if(max_left_vel_clicks > 0){
      m_izq_forward();
   } else {
      m_izq_backward();
   }
   if(max_right_vel_clicks > 0) {
      m_der_forward();
```

```
   } else {
      m_der_backward();
   }
}


void m_stop()
{
   edo_left_motor = edo_right_motor=0;
   max_left_vel_clicks = max_right_vel_clicks = max_bias_clicks = 0;
   set_velocity();
}

void stop()
{
   ser0_puts("d\n");
   m_stop();
}



/* *** SET_VEL COMMAND *** */
//****** Diferentes valores para la interrupcion de tiempo real ************
//#define rtimask       b(0,1,1,0,0,0,0,1) //(1/16Mhz)2*2^12=512 usec interrupt
#define rtimask         b(0,1,0,0,0,0,0,1) //(1/16Mhz)2*2^13=1.024 msec interrupt
//#define rtimask b(0,1,0,1,0,0,0,1) //(1/16Mhz)2*2^14=2.048 msec interrupt
//#define rtimask       b(0,1,1,0,0,0,0,1) //(1/16Mhz)2*2^15=4.096 msec interrupt
//*************************************************
#define RTIF            b(1,0,0,0,0,0,0,0)
#define RTIE            b(1,0,0,0,0,0,0,0)
//volatile extern unsigned int im;

void set_vel(int num_param, unsigned short param[])
{
   ser0_puts("v\n");
   if(num_param >=2) {
      max_left_vel_clicks  = param[0] - 100;
      max_right_vel_clicks  = param[1] - 100;
   }
   if(num_param == 3)
      max_bias_clicks = param[2] - 100;
   else
      max_bias_clicks = 0;
   set_velocity();
//*********************************************
   ICPAR |= b(0,0,0,0,1,1,1,1); //Pulse Acumulator 0->3 are enabled
```

```
   im=0;
   RTICTL = rtimask;  //Initialize Real Time Interrupt rate
   CRGFLG |= RTIF; //clear flag
   CRGINT |= RTIE;  //Enable RTI
   flagI=0; //Libera banderas
   flagD=0;
   Index_MI=0;
   Index_MD=0;
   VI=0;
   VD=0;
   FI=1;
   FD=1;
//*********************************************
   clicks_stop = 0;
   acelerando = 0;
   frenando  = 0;
}
/* *** END SET_VEL *** */


void set_ac(short num_params, unsigned short param[])
{
   ser0_puts("k\n");
   k_vel_ini = param[0];
   k_vel_end = param[1];
   k_ms_inc  = param[2];
}

void set_params(short num_params, unsigned short param[])
{
   if(num_params == 0)
      clicks_stop = 0;
   else
      clicks_stop = (long) param[0] * (long) 400;
   total_left_clicks = total_right_clicks =0;
   clicks_break =  clicks_stop/2;
   timer_ac = k_ms_inc;
   max_bias_clicks = 0;
   acelerando = 1;
   frenando = 0;
}


void forward(short num_params, unsigned short param[])
{
   ser0_puts("a\n");
```

```
   set_params(num_params, param);
   max_left_vel_clicks = max_right_vel_clicks = k_vel_ini;
   set_velocity();
}


/* SPEED_CONTROL */

//static unsigned char limit_range(unsigned char pwm, long error)
static unsigned char limit_range(long error)
{
   static long new_pwm;

   //new_pwm = pwm + error;
   //if(error==0){
    //new_pwm = pwm + error;
   //}
   //else{
    new_pwm = error;
   //}

   //if(new_pwm > 255)
   if(new_pwm > 255)
      //new_pwm = 255;
      new_pwm = 255;
   else if(new_pwm < 0)
      new_pwm = 0;
   return (unsigned char)new_pwm;
}

void alter_left_power(long error )
{
   //left_pwm = limit_range(left_pwm,error);
   left_pwm = limit_range(error);
   set_left_pwm();
}

void alter_right_power(long error )
{
   //right_pwm = limit_range(right_pwm,error);
   right_pwm = limit_range(error);
   set_right_pwm();
}

unsigned int im1=0;
unsigned short bandera=0;
```

```
unsigned short err_l[NPULSOS];
unsigned short err_r[NPULSOS];
unsigned short U_l[NPULSOS];
unsigned short U_r[NPULSOS];
//unsigned short Prop_l[NPULSOS];
//unsigned short err_r[NPULSOS];
//volatile extern unsigned int Index_MI;
//volatile extern unsigned int Index_MD;


void speed_control()
{
   static long int left_vel, right_vel, integral_error, left_error, right_error,
               vel_clicks, bias_clicks;

   if(!edo_left_motor && !edo_right_motor)
      return;
   left_vel  = g_left_vel;
   right_vel = g_right_vel;

   total_left_clicks += left_vel; //Update total clicks
   total_right_clicks += right_vel;

if(VI >= des_posI && FI ){ //Condicion de para por alcance de distancia Mizq
FI=0;
m_izq_stop();
flagI=1;
prop_left=0;
//des_posI=65;
//ser0_putushort(VI);
//ser0_puts("\n");
Total_posI += VI;
//ser0_putushort(Total_posI);
//ser0_puts("\n");
//asm("bclr 0x0068,0x0e"); //Pulse Acumulator 3,2 y 1 deshabilitados

ICPAR = b(0,0,0,0,0,0,0,0); //Pulse Acumulator 0->3 are disabled   1,2->MizqA,MizqI
//ICPAR |= b(0,0,0,0,1,1,1,1); //Pulse Acumulator 0->3 are enabled    0,4->MderI,MderA
}

if(VD >= des_posD && FD ){ //Condicion de para por alcance de distancia Mder
FD=0;
m_der_stop();
flagD=1;
prop_right=0;
//des_posD=65;
```

```
//ser0_putushort(VD);
//ser0_puts("\n");
Total_posD += VD;
//ser0_putushort(Total_posD);
//ser0_puts("\n");
//asm("bclr 0x0068,0x0d"); //Pulse Acumulator 3,2 y 0 deshabilitados

ICPAR = b(0,0,0,0,0,0,0,0); //Pulse Acumulator 0->3 are disabled
//ICPAR |= b(0,0,0,0,1,1,1,1); //Pulse Acumulator 0->3 are enabled
}

if( flagI && flagD ){
CRGINT &=  b(0,1,1,1,1,1,1,1); //Deshabilita la interrupcion de tiempo real (RTI)
//ICPAR = b(0,0,0,0,0,0,0,0); //Pulse Acumulator 0->3 are disabled
//MCCTL = b(0,0,0,1,0,0,0,0); //Force the contents of 8-bit accumulators to be latched into their
//associated holding registers (PA2H Y PA3H en este caso)
//TIE   =  b(0,0,0,0,0,0,0,0);  //Timer Interrupt Enable Register
//ser0_puts("Si\n");

}
//Controlador PI en su version clasica  U(tk)=P(tk)+I(tk) donde:
//P(tk)=k( Wref(tk) - W(tk) )
//I(tk+1)=I(tk) + kh/Ti(Wref(tk) - W(tk) )
//***************** Termino Integral ***************************
if(bandera==1){
if(left_error > 255){ //Saturación limite superior

}
else if(left_error < 0){ //Saturacion limite inferior

}
else{ //Operacion normal del actuador
integral_left +=  (k_integral*(des_left_vel_clicks - left_vel))/den;
}

if(right_error > 255){ //Saturación limite superior

}
else if(right_error < 0){ //Saturacion limite inferior

}
else{ //Operacion normal del actuador
integral_right += (k_integral*(des_right_vel_clicks - right_vel))/den;
}
}
else{
```

```
integral_left +=  (k_integral*(des_left_vel_clicks - left_vel))/den;
integral_right += (k_integral*(des_right_vel_clicks - right_vel))/den;
bandera=1;
}
//if (im1 <  NPULSOS ){
// err_l[im1] = (des_left_vel_clicks - left_vel);
// Prop_l[im1] = prop_left;
// im1++;
//}
//************Termino Proporcional *************************
prop_left =( (k_pro*((des_left_vel_clicks) - left_vel))/den );
prop_right =( (k_pro*((des_right_vel_clicks) - right_vel))/den );

//**********-Cálculo de la accion de contriol-*****************

   left_error  = (prop_left) + (integral_left);
   right_error = (prop_right) + (integral_right);


//******************-Mediciones-**********************
   if (im1 <  NPULSOS ){
err_l[im1] = (des_left_vel_clicks - left_vel); //Almacena lecturas
err_r[im1] = (des_right_vel_clicks - right_vel);
U_l[im1] = left_error;
U_r[im1] = right_error;
im1++;
}

   alter_left_power (left_error );
   alter_right_power(right_error);
}
//*****************************************************

void ac_control()
{
   if(timer_ac){
      timer_ac--;
      return;
   }
   if(acelerando){
      if(g_left_vel + 3 >= k_vel_end){ // reach the max velocity
          clicks_break = clicks_stop - total_left_clicks;
          acelerando = 0;
      }
      if(des_left_vel_clicks < k_vel_end) {
         des_left_vel_clicks++;
```

```
 des_right_vel_clicks++;
      }
   }
   if (frenando && des_left_vel_clicks) {
         des_left_vel_clicks--;
         des_right_vel_clicks--;
   }
   timer_ac = k_ms_inc;
}


void check_stop_motors(void)
{
   if(clicks_stop && total_left_clicks >= clicks_stop && total_right_clicks >= clicks_stop)
      m_stop();
   if(edo_left_motor && !frenando && total_left_clicks >= clicks_break) {
      acelerando = 0;
      frenando   = 1;
   }
}


void beep_on()
{
   PORTT |= b(1,0,0,0,0,0,0,0);
}

void beep_off()
{
   PORTT &= ~b(1,0,0,0,0,0,0,0);
}

void beep(int num_parmams, short param[])
{
   if(param[0])
      beep_on();
   else
      beep_off();
}


void init_control()
{
   DDRP = 0xff;         //all are output
   PORTP = 0;
```

```
   DDRT = b(1,0,0,0,0,0,0,0);
   PORTT = 0;

   //PWM

   PWME |= mP_MIzqPWM | mP_MDerPWM;      // Pulse Width Channel 6 & 7 Enable (use Clock B)
   PWMPRCLK = b(0,0,1,0,0,0,0,0);       //Clock B=Bus clock/4
                                        //1/(1/24e6 * 4 *256))=23437Hz

   //Initialize Analog To Digital
   ATD0CTL2 = 0x80; //enable ATD
   ATD0CTL3 = 0x40;
   ATD0CTL4 = 0x60; //Select Sample rate
   ATD0CTL5 = 0xb0;   //Select 8 channel mode, Continuous scan

   //ECT

   ICPAR |= b(0,0,0,0,1,1,1,1); //Pulse Acumulator 0->3 are enabled
   //ICPAR |= b(0,0,0,0,1,1,0,0); //Pulse Acumulator 0->3 are enabled
   DLYCT |= b(0,0,0,0,0,0,0,1); //Delay Counter = 256 bus clock cycles
   //DLYCT |= b(0,0,0,0,0,0,0,0); //Delay Counter = 256 bus clock cycles
   TIOS  |= b(1,0,0,0,0,0,0,0); //All channels are input capture but 7
   TCTL4 |= b(1,1,1,1,0,1,0,1); //Input Capture Edge Control
//1,1=Capture on any edge, rising or falling (encoders)
   //TCTL3 |= b(0,0,0,0,0,0,0,1); //Input Capture Edge Control
                                //0,1=Capture on rising only (mT_MIzqB, mT_MDerB, mT_MDerI)
   ICSYS |= b(0,0,0,0,0,1,1,1); //PACMX=1 When the 8-bit pulse accumulator has reached the value
//$FF, it will no be incremented further.
//BUFEN=1 Input Capture and pulse accumulators registers are enabled
//LATQ=1 Latch Mode of Input Capture is enabled

   //TIE   =  b(0,0,0,1,0,0,1,0); //Timer Interrupt Enable Register
   //TSCR2 =  b(0,0,0,0,0,1,1,1); //Timer Prescaler = Bus clock /128 =5.33 microseg
   //TSCR1 =  b(1,0,0,0,0,0,0,0); //TEN=1 => TimerENable = 1
   //TFLG1 |=  b(0,0,0,1,0,0,1,0); //Clear Sonar Interrupt Flag
   //MCCNT = 0;

   PWMPOL = 0xff; // PWM outputs are high at the beginning of the period, then goes
// low when the duty count is reached
   max_left_vel_clicks = max_right_vel_clicks = max_bias_clicks=0;
   set_velocity();
}

//******************Función agregada*****************************//
//DESCRIPCIÓN: Esta función tiene como finalidad iniciar el proceso de sensado mediante
//un comando 'I' tecleado en la PC y enviado al microcontrolador por el puerto serie.
```

```
//Inicializa la interrupción de tiempo real (cada 1mSeg.) e inicia el arranque del motor
//en el esquema PWM a un 100%, con la finalidad de aplicar el escalon al motor.


//volatile extern unsigned int im;
volatile extern unsigned short pulsos_der[NPULSOS];
volatile extern unsigned short pulsos_izq[NPULSOS];
volatile extern unsigned short I_arranque[NPULSOS];

void inicio()
{
unsigned int index;
ser0_puts("r\n"); //MANDA ECO
//*******-MANDA VALORES DE LAS CONSTANTES QUE SE USARON EN EL CONTROL-***********
    ser0_puts("k_pro=");
ser0_putushort(k_pro);
ser0_puts("\n");

ser0_puts("k_integral=");
ser0_putushort(k_integral);
ser0_puts("\n");

ser0_puts("den=");
ser0_putushort(den);
ser0_puts("\n"); //32

ser0_puts("f"); //Marcador de fin de variables utilizadas
ser0_puts("\n");
//ser0_puts("Muestras=");
ser0_putushort(im); //Manda el numero de muetras
ser0_puts("\n");

   for (index=0; index<NPULSOS; index++){
    ser0_putushort(pulsos_izq[index]);
ser0_puts("\n");
}

   for (index=0; index<NPULSOS; index++){
    ser0_putushort(pulsos_der[index]);
ser0_puts("\n");
}

}
//*************************************************//
void sintoniza(int num_param, unsigned short param[])
{
```

```
   ser0_puts(" Recivido\n");
    if (num_param == 0){
    ser0_puts("Sin Cambio\n");
}
else{
if(num_param == 1){
       k_pro  = param[0];

}
    else if (num_param == 2){
    k_pro  = param[0]; //Ajuste de la ganancia proporcional (k_pro*Error/den)
       k_integral  = param[1]; //Ajuste de la ganancia integral (k_integral*Error/den)

}
    else{
k_pro  = param[0]; //Ajuste de la ganancia proporcional (k_pro*Error/den)
       k_integral  = param[1]; //Ajuste de la ganancia integral (k_integral*Error/den)
    den = param[2]; //Denominador

}
    }


}
//************************************************//
void posicion(int num_param, unsigned short param[])
{

    if (num_param == 0){
    ser0_puts("Sin Cambio\n");
}
else{
if(num_param == 1){
       des_posI = param[0];
ser0_puts(" Recivido\n");

}
else if (num_param == 2){
    des_posI  = param[0]; //
       des_posD  = param[1]; //
ser0_puts("Esc: Frac Vueltas\n");
}

    else{
des_posI  = (param[0])*65; //
       des_posD  = (param[1])*65; //
```

```
ser0_puts("Esc: Vueltas\n");

}
    }


}
//*****************************************************//
void rti_mul(int num_param, unsigned short param[])
{

    if (num_param == 0){
    ser0_puts("Sin Cambio\n");
}
else{
ser0_puts("m\n");
       mt = param[0];
}


}
```

```
#include "ioRegsHcs12DP256.h"
/*
 *  isr.c
 */

#include  "isr.h"
#include  "control.h"
#include  "serial.h"
#include  "util.h"

void  __attribute__ ((interrupt)) ISR_Empty(void)
{
}

#define NPULSOS 500
unsigned short g_timer=0;
unsigned int im=0;
unsigned short int pulsos_der[NPULSOS];
unsigned short int pulsos_izq[NPULSOS];
//unsigned short int I_arranque[NPULSOS];

//volatile unsigned short n=1;
//unsigned short n1;
volatile unsigned short g_ad2=0;
unsigned short g_ad3=0;
unsigned short g_ad4=0;
unsigned short g_ad5=0;
volatile unsigned short g_left_vel;
volatile unsigned short g_right_vel;
unsigned long Index_MI=0; //Index_MI e Index_MD cuentan hasta 2exp32
unsigned long Index_MD=0;
unsigned int VI=0; //VI y VD cuentan hasta 65536
unsigned int VD=0;

unsigned short cc=2; //multilo de la RTI en este caso 2*1e-3
unsigned short mtAux=1;
volatile extern unsigned short mt;
//unsigned short mm=10;
//unsigned short key=1;

extern void speed_control();

//******Lectura de encoders para control de velocidad****************
static void inline get_vel()
{
#define ICLAT b(0,0,0,1,0,0,0,0)
```

```
MCCTL |= ICLAT; //Force the contents of 8-bit accumulators to be latched into their
//associated holding registers (PA2H Y PA3H en este caso)

Index_MD += PA0H;
Index_MI += PA1H;
g_left_vel = PA2H;
g_right_vel= PA3H;
}
//*****************************************************
#define RTIF b(1,0,0,0,0,0,0,0)
#define SCFflag         b(1,0,0,0,0,0,0,0)       //SCF - Sequence Complete flag
void  __attribute__ ((interrupt)) ISR_RealTimeInt(void)
{
   CRGFLG |= RTIF;      //clear flag
   asm volatile ("pshc\n pshd\n; pshx\n pshy");
   if(g_timer)
      g_timer--;
   //if (im <  NPULSOS ){
    //MCCTL |= ICLAT; //Force the contents of 8-bit accumulators to be latched into their
//associated holding registers (PA2H Y PA3H en este caso)
//pulsos_izq[im] = PA2H; //Almacena la lectura de pulsos (PT2) Canal A del encoder
//pulsos_der[im] = PA3H;


//while(!(ATD0STATH & SCFflag));       //Loop here until SCF of ATD is set  ATD0STATH->Status register
//I_arranque[im] = ATD0DR2L; //save ATD, Valor del convertidor-->Guardado en I_arranque
    //g_ad3=ATD0DR3H;
    //g_ad4=ATD0DR4H;
    //g_ad5=ATD0DR5H;
//im++;

if (cc == 1){ //Multilo de la interrupcion cc=2mSeg
get_vel();

//if(key == 1){
//get_vel2(); //Cada 2mSeg le el acumulador para el control de posicion
VI = Index_MI/500; //Durante 10 veces
VD = Index_MD/500;

//********-Almacenamiento de datos en multiplos de RTI-***************
if ( mtAux == 1 ){
//mtAux--;
if (im <  NPULSOS ){
pulsos_izq[im] = g_left_vel; //Almacena la lectura de pulsos (PT2) Canal A del encoder
pulsos_der[im] = g_right_vel;
//while(!(ATD0STATH & SCFflag));       //Loop here until SCF of ATD is set  ATD0STATH->Status register
```

```
//I_arranque[im] = ATD0DR2L; //save ATD, Valor del convertidor-->Guardado en I_arranque
im++;
//n1=n;
//speed_control();
//ac_control();
}
mtAux=mt;
}
else{
mtAux--;
}
//*************************************************
cc = 2; //Retorna valor original
speed_control();
}
else{

cc--;
}

  asm volatile ("pulc\n puld\n; pulx\n puly");
}

//unsigned int Index_MI;
void  __attribute__ ((interrupt)) ISR_MIzqI(void)
{
   //static unsigned int Index_MI;

   asm("sei");                  //Disable Any interrupts
   //m_izq_stop();
   Index_MI++;
   TFLG1 |= mT_MIzqI;
   asm("cli");                  //Enable Any interrupts
}

//unsigned int Index_MD;
void  __attribute__ ((interrupt)) ISR_MDerI(void)
{
   //static unsigned int Index_MD;

   asm("sei");                  //Disable Any interrupts
   //m_der_stop();
   Index_MD++;
   TFLG1 |= mT_MDerI;
   asm("cli");                  //Enable Any interrupts
}
```

C.3. Rutinas Para la Ejecución del Vector de Velocidades.

La sección de codigo expuesto a continuación, representa las rutinas para la formación del vector de velocidades así como para la ejecución del mismo. El resto del codigo no presenta cambio alguno al expuesto en la sección anterior.

```
//*************************************************************//
//Esta funcion sirve para almacenar las velocidades de ambos motores y el multiplo ded rti
//necesario para describir la seccion de curva a recorrer
void data_set(int num_param, unsigned short param[])
{
//ser0_puts("c\n");
ref_izq[zz]  = param[0]; //ej: c11 9 3721  (ref_izq=11P/mSEG, ref_der=9P/mSeg
ref_der[zz]  = param[1]; //& m_rti=3271 cuentas de 2mSeg.
m_rti[zz] = param[2]; //tiempo en multiplo de rti para mantener las velocidades de Mizq & Mder
    zz++;
   //ser0_putushort(zz);
//ser0_puts("\n");

}
//*************************************************************//
//Esta funcion sirve para inicializar la referencia de velocidad asi
//como la interrupcion de tiempo real RTI
volatile unsigned short ok=0; //Candado para la inicializacion de Spline
void star_spline() //Modo de ejecusion ej:f
{
np=6;
inc_sub_ref_izq = ref_izq[0] / np; //Inicializa la velocidad
sub_ref_izq = inc_sub_ref_izq;
inc_sub_ref_der = ref_der[0] / np;
sub_ref_der = inc_sub_ref_der;
des_left_vel_clicks = inc_sub_ref_izq; //Inicializa la velocidad
des_right_vel_clicks = inc_sub_ref_der;
inc_sub_m_rti = m_rti[0] / np;
sub_m_rti = inc_sub_m_rti;

npf=6;
dec_sub_ref_izq = ref_izq[zz-1] / npf;
sub_ref_izq_fin = ref_izq[zz-1];
dec_sub_ref_der = ref_der[zz-1] / npf;
sub_ref_der_fin = ref_der[zz-1];
dec_sub_m_rti = m_rti[zz-1] / npf;
sub_m_rti_fin = dec_sub_m_rti;
```

```
nPT = (NPULSOS-30) / zz; //Numero de puntos por tramo
//ser0_putushort(nPT);
//ser0_puts("\n");
mt = m_rti[0] / nPT; //Multilo de muestreo inicial
//ser0_putushort(mt);
//ser0_puts("\n");
im=0;
ok = 1;

ICPAR |= b(0,0,0,0,1,1,1,1); //Pulse Acumulator 0->3 are enabled
    RTICTL = rtimask;  //Initialize Real Time Interrupt rate
    CRGFLG |= RTIF; //clear flag
    CRGINT |= RTIE;  //Enable RTI
}
//*************************************************************//
//Esta funcion sirve para alinear el robot hacia el primer punto generado por el Spline
volatile unsigned short ok2=0; //Candado para la alineacion del Robot
volatile unsigned int m_alinea; //Multiplo de RTI para la alineacion
void alinea_robot(int num_param, unsigned short param[]) //Modo de ejecusion ej: b3122
{
m_alinea = param[0];
ok2=1;
}
//*************************************************************//
void retorna(int num_param, unsigned short param[])
//Modo de ejecusion ej: j0 & j1 (atras & adelante)
{
left_pwm=right_pwm=0U;          // Initial power to motors
    edo_left_motor=edo_right_motor=1;
    integral_left=0;
    integral_right=0;
    set_left_pwm();
    set_right_pwm();

if(param[0] == 1){
//ser0_puts("j1_Ad\n");
m_izq_forward();
m_der_forward();
}
    else{
//ser0_puts("j0_At\n");
m_izq_backward();
       m_der_backward();
}
}
```

Apéndice D

DIAGRAMA ELÉCTRICO

El diagrama eléctrico mostrado en la figura D.1 consta escencialmente de dos etapas: la etapa de potencia y la de instrumentación y control. En la etapa de potencia se tiene un par de baterias de 12 Volts conectadas en serie de tal forma que la suma de voltajes es igual al voltaje nominal de los motores ded C.D. Este voltaje es llevado a través de un fusible de protección de 10 A. hasta la terminal V_s correspondiente a cada puente H.

La etapa de instrumentación y control consta de un regulador de voltaje ajustable tipo corcholata de 5 A. de capacidad de corriente y configurado para un voltaje de 5 Volts [National98]. Dicho regulador es alimentado con una de las dos bateria disponibles para la etapa de instrumentación y control y éste alimenta tanto los encoders de ambos motores como el microcontrolador.

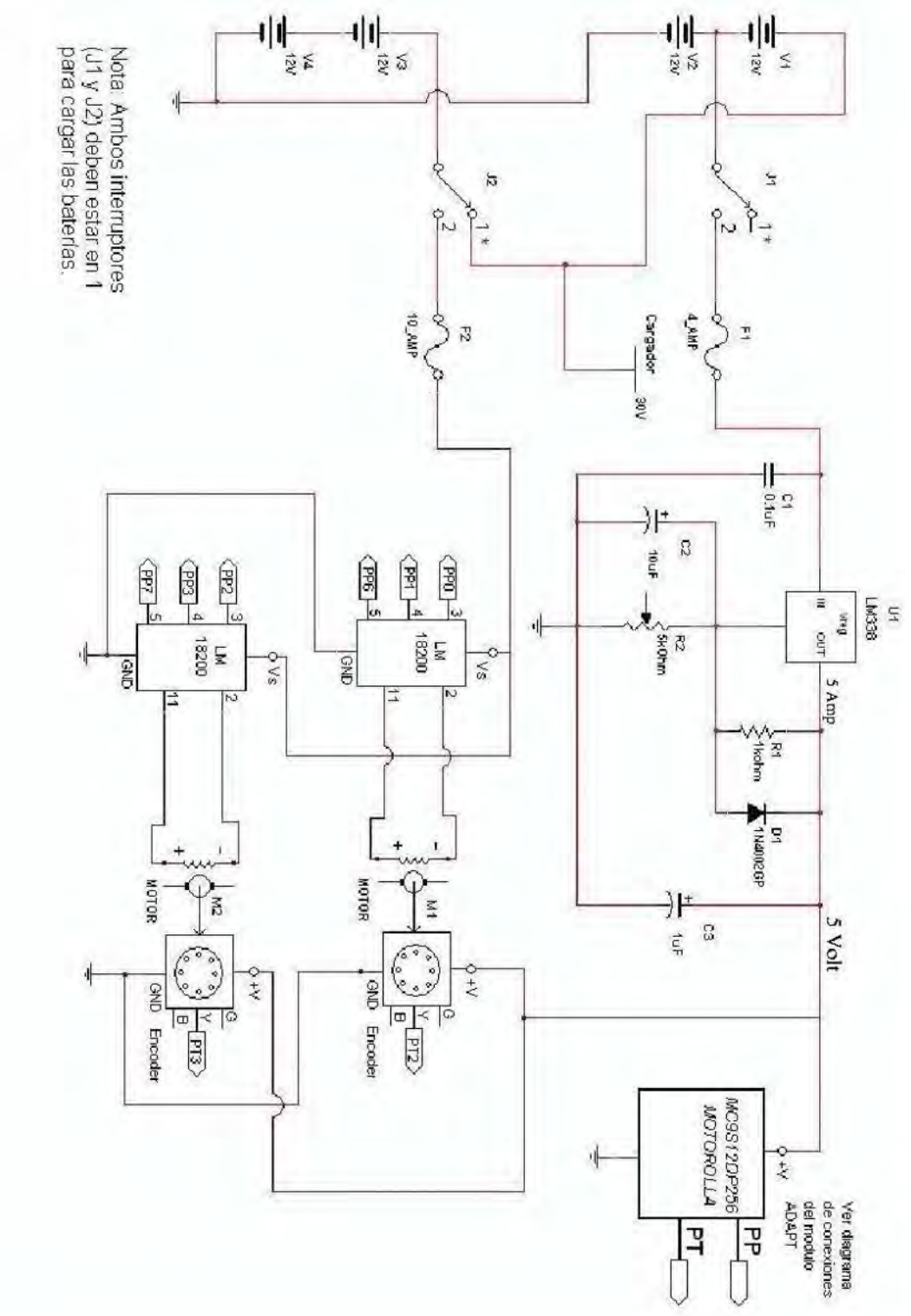


Figura D.1: Diagrama Eléctrico del robot.

Apéndice E

HOJAS DE DATOS

GM9236S027

Assembly Data	Symbol	Units	Value	
Reference Voltage	E	V	24	
No-Load Speed	S_{NL}	rpm (rad/s)	71	(7.4)
Continuous Torque (Max.)[1]	T_C	oz-in (N-m)	480	(3.4E+00)
Peak Torque (Stall)[2]	T_{PK}	oz-in (N-m)	2585	(1.8E+01)
Weight	W_M	oz (g)	23.7	(671)
Motor Data				
Torque Constant	K_T	oz-in/A (N-m/A)	6.49	(4.58E-02)
Back-EMF Constant	K_E	V/krpm (V/rad/s)	4.80	(4.58E-02)
Resistance	R_T	Ω	2.49	
Inductance	L	mH	2.63	
No-Load Current	I_{NL}	A	0.16	
Peak Current (Stall)[2]	I_P	A	9.64	
Motor Constant	K_M	oz-in/√W (N-m/√W)	4.11	(2.90E-02)
Friction Torque	T_F	oz-in (N-m)	0.80	(5.6E-03)
Rotor Inertia	J_M	oz-in-s^2 (kg-m^2)	1.0E-03	(7.1E-06)
Electrical Time Constant	τ_E	ms	1.06	
Mechanical Time Constant	τ_M	ms	8.5	
Viscous Damping	D	oz-in/krpm (N-m-s)	0.053	(3.5E-06)
Damping Constant	K_D	oz-in/krpm (N-m-s)	12.5	(8.5E-04)
Maximum Winding Temperature	θ_{MAX}	°F (°C)	311	(155)
Thermal Impedance	R_{TH}	°F/watt (°C/watt)	56.3	(13.5)
Thermal Time Constant	τ_{TH}	min	13.5	
Gearbox Data				
Reduction Ratio			65.5	
Efficiency[4]			0.80	
Maximum Allowable Torque		oz-in (N-m)	500	(3.53)
Encoder Data				
Channels			3	
Resolution		CPR	500	

Web Site: www.pittmannet.com

Figura E.1: Hoja de datos para los motores de CD [Pittman01].

National Semiconductor

September 1996

LMD18200 3A, 55V H-Bridge

General Description

The LMD18200 is a 3A H-Bridge designed for motion control applications. The device is built using a multi-technology process which combines bipolar and CMOS control circuitry with DMOS power devices on the same monolithic structure. Ideal for driving DC and stepper motors; the LMD18200 accommodates peak output currents up to 6A. An innovative circuit which facilitates low-loss sensing of the output current has been implemented.

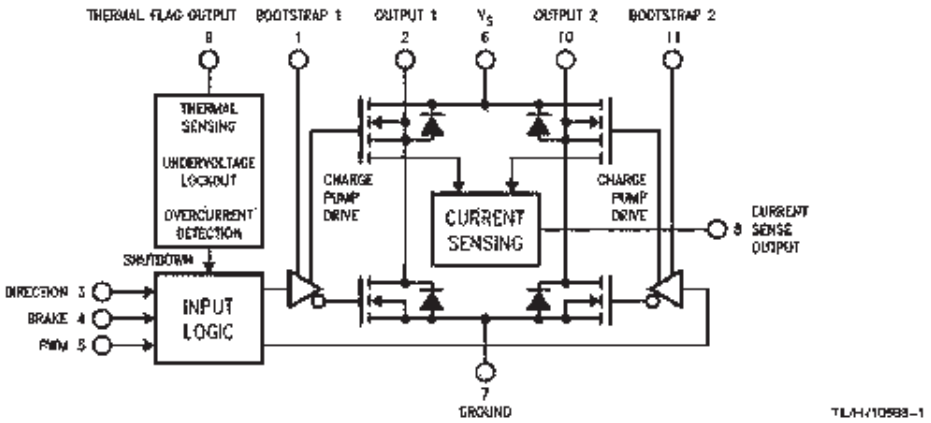
Features

- Delivers up to 3A continuous output
- Operates at supply voltages up to 55V
- Low $R_{DS}(ON)$ typically 0.3Ω per switch
- TTL and CMOS compatible inputs
- No "shoot-through" current
- Thermal warning flag output at 145°C
- Thermal shutdown (outputs off) at 170°C
- Internal clamp diodes
- Shorted load protection
- Internal charge pump with external bootstrap capability

Applications

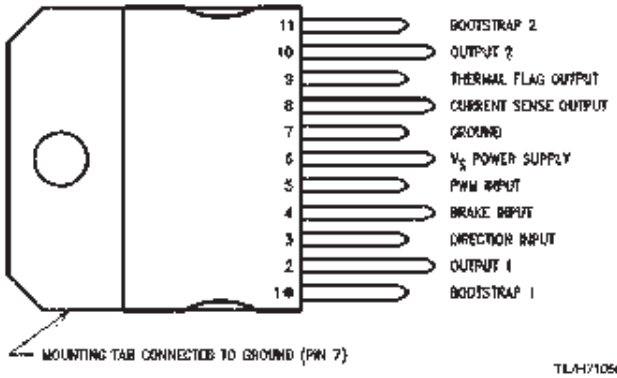
- DC and stepper motor drives
- Position and velocity servomechanisms
- Factory automation robots
- Numerically controlled machinery
- Computer printers and plotters

Functional Diagram



TL/H/10568–1

Connection Diagram and Ordering Information



TL/H/10568–2

Top View

Order Number LMD18200T
See NS Package TA11B

RRD-B30M96/Printed in U.S.A.

http://www.national.com

Figura E.2: Hoja de datos del puente H [National96].

Model IHA-25/IHA-100 Open Loop Hall Effect

Current Sensors

Description

The IHA Series Hall effect current sensors accurately measure DC and AC currents and provide electrical isolation between the output of the sensor and the current carrying conductor.

Features

- High accuracy
- Wide frequency range
- Excellent linearity
- Safety isolation
- Rack and bulkhead
- Light duty plastic housing

Applications

- Motor controllers and drives
- Battery supplied equipment
- Switch mode and uninterruptable power supplies
- Welding equipment

	Units	IHA-25	IHA-100
Measuring Circuit			
Full Scale (FS) DC or AC peak	± A	25	100
Full Scale output	± V	1	5
AC bandwidth (±1% of reading) (1)	kHz	50	
Response time (2)	µs	<1	
Slew rate	A/µs	>150	
Excitation Circuit			
Supply voltage	± Vdc	12 to 17	
Maximum supply current, positive supply (at 15V)	mA	10	
Maximum supply current, negative supply (at 15V)	mA	5	
Output			
Sensitivity	mV/A	40	50
Linearity	± %FS	<1	
Calibration point (3)	± %RDG	0.5	
Typical zero current offset	± mV	10	
Maximum zero current offset	± mV	20	
Maximum hysteresis of offset (4)	± mV	7	35
Minimum load resistance	kohms	>10	
Influences On Accuracy			
Typical offset drift with temperature	±mV/°C	1	
Maximum offset drift with temperature	±mV/°C	2	
Excitation change of ±1% - Max. sensitivity change	± %	0.005	
Typical sensitivity drift with temperature	±%/°C	0.010	
Maximum sensitivity drift with temperature	±%/°C	0.015	
Withstand Capabilities			
Dielectric test (5)	kV	6	
Output short or open		No Damage	
General Information			
Operating temperature range	°C	0 to +75	
Storage temperature range	°C	-25 to +85	
Package		flame retardant plastic case	
Aperture opening	inches (mm)	0.38 (9.65)	
Weight	grams	25.9	
Mounting		Mounting tabs accept No. 6 screws. Can be mounted on PCB or panel via use of appropriate connector.	
Output reference		To obtain a positive output on pin marked "Vo", positive conventional current must flow as per the direction of arrow marked on sensor.	

SYPRIS test & assessment
6120 Hanging Moss Road • Orlando, Florida 32807
Phone (407) 678-9748 • Fax (407) 677-5765 • www.fwbell.com

F.W. BEL

Rev. data 0

Figura E.3: Hoja de datos del sensor de corriente [BELL04].

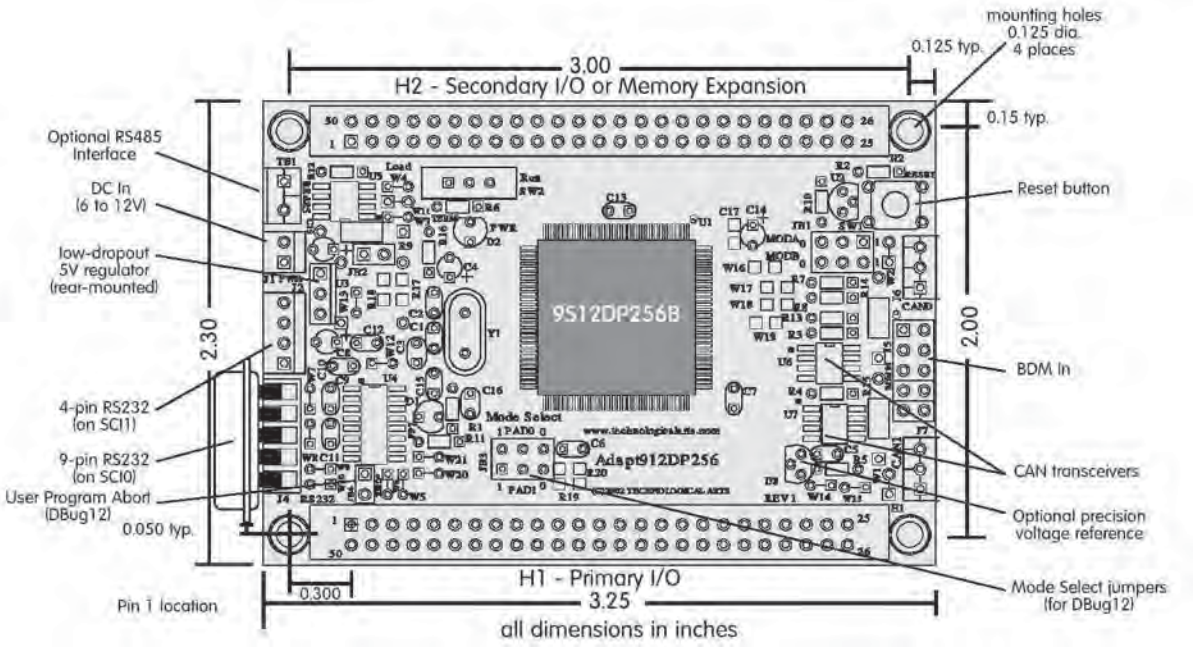


Figura E.4: Modulo Adapt9S12DP256 de Technological Arts [Arts02].

H1				H2			
PIN	SIGNAL NAME	PIN	SIGNAL NAME	PIN	SIGNAL NAME	PIN	SIGNAL NAME
1	PS4/MISO	50	GROUND	1	PA7/ADDR15/DATA15	50	VCC (+5VDC)
2	PS5/MOSI	49	GROUND	2	PA6/ADDR14/DATA14	49	GROUND
3	PS6/SCK	48	PS0/RXD0	3	PA5/ADDR13/DATA13	48	PE7/NOACC/XCLKS*
4	PS7/SS*	47	+5VDC	4	PA4/ADDR12/DATA12	47	PK7/ECS*
5	PS1/TXD0	46	PE1/IRQ*	5	PA3/ADDR11/DATA11	46	PK5/XADDR19
6	PT7/IOC7	45	PE0/XIRQ*	6	PA2/ADDR10/DATA10	45	PK4/XADDR18
7	PT6/IOC6	44	RESET*	7	PA1/ADDR9/DATA9	44	PK3/XADDR17
8	PT5/IOC5	43	PE7/NOACC/XCLKS*	8	PA0/ADDR8/DATA8	43	PK2/XADDR16
9	PT4/IOC4	42	PH0/KWH0	9	PB7/ADDR7/DATA7	42	PK1/XADDR15
10	PT3/IOC3	41	PH1/KWH1	10	PB6/ADDR6/DATA6	41	PK0/XADDR14
11	PT2/IOC2	40	PH2/KWH2	11	PB5/ADDR5/DATA5	40	PJ0/KWJ0
12	PT1/IOC1	39	PH3/KWH3	12	PB4/ADDR4/DATA4	39	PJ7/SCL
13	PT0/IOC0	38	PH4/KWH4	13	PB3/ADDR3/DATA3	38	PJ6/SDA
14	PP7/KWP7/PWM7	37	PH5/KWH5	14	PB2/ADDR2/DATA2	37	TxCAN3/PM7
15	PP6/KWP6/PWM6	36	PH6/KWH6	15	PB1/ADDR1/DATA1	36	RxCAN3/PM6
16	PP5/KWP5/PWM5	35	PH7/KWH7	16	PB0/ADDR0/DATA0	35	TxCAN2/PM5
17	PP4/KWP4/PWM4	34	PS2/RXD1	17	R/W* PE2	34	RxCAN2/PM4
18	PP3/KWP3/PWM3	33	PE4/ECLK	18	ECLK/PE4	33	TxCAN1/PM3
19	PP2/KWP2/PWM2	32	PS3/TXD1	19	LSTRB*/PE3	32	RxCAN1/PM2
20	PP1/KWP1/PWM1	31	VRL	20	IRQ*/PE1	31	TxCAN0/PM1
21	PP0/KWP0/PWM0	30	VRH	21	PJ1/KWJ1	30	RxCAN0/PM0
22	PAD00/AN00	29	PAD04/AN04	22	PAD08/AN08	29	PAD12/AN12
23	PAD01/AN01	28	PAD05/AN05	23	PAD09/AN09	28	PAD13/AN13
24	PAD02/AN02	27	PAD06/AN06	24	PAD10/AN10	27	PAD14/AN14
25	PAD03/AN03	26	PAD07/AN07	25	PAD11/AN11	26	PAD15/AN15

NOTES: * indicates active low signal

Figura E.5: Conexiones [Arts02].

Apéndice F

ESQUEMA DE MEDICIÓN DE CORRIENTE

Para las pruebas de medición de corriente en el circuito de armadura de los motores de CD, se utilizó un sensor de efecto *Hall* . La hoja de datos del sensor mencionado se muestra en el apéndice G.

Como primer paso es necesario hacer un análisis de la escala de medición, que en este caso se trata de la corriente del circuito de armadura de los motores de CD. De acuerdo con la hoja de datos de los motores, se espera una corriente pico de aproximadamente $I_{max} = 9.64Amp.$. Redondeando tenemos una escala de medición de corriente de 0 a 10 Amp y dado que el sensor de corriente tiene una salida de 40 mili Volt por Amper ($40mV/A$), en realidad tendremos una escala de 0 a 400mV, los cuales son enviados al modulo de conversión analógico digital (ATD) del microcontrolador no sin antes ser multiplicados por una ganancia A_v. En la figura F.1 se muestra el esquema de ganancia.

La relación existente entre el voltaje de entrada V_e y el voltaje de salida V_s está dada por la siguiente ecuación.

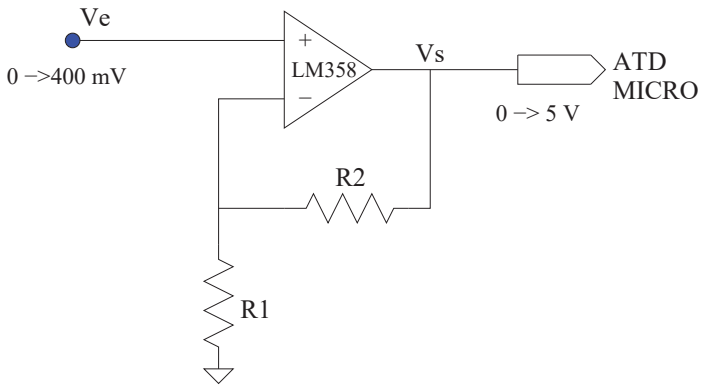


Figura F.1: Adaptación de la señal analógica.

$$\begin{aligned} V_s &= \frac{R1 + R2}{R1} V_e \\ o & \\ V_s &= A_v V_e \end{aligned} \tag{F.1}$$

Donde para tener una salida de 0 a 5 Volt es necesario tener una ganancia de $A_v = 5/255 = 12.5$ la cual se ajusta por medio de $R1$ y $R2$.

El módulo de conversión analógico digital se configuró para ser usado en el modo de resolución de 8 bit, con justificación izquierda y como voltaje de referencia el voltaje de alimentación (5 Volt). De esta manera tenemos la siguiente relación.

$\frac{5Volt}{255} = 19.6mV$ es decir que $19.6mV$ significa $d'1'$, sin embargo pruebas realizadas con la finalidad de mejorar las mediciones mostraron que la unidad se obtiene en $25mV$.

El esquema de muestreo es exactamente igual al de medición de la velocidad, el cual si recordamos un poco se hacia a través del conteo de pulsos cada periodo de muestreo, posteriormente se almacenaba un número determinado de muestras y una vez terminado el experimento se enviaban por el puerto serie. En este caso en lugar de pulsos se almacena el valor del convertidor. Dado que el valor de las muestras de corriente son enteros que oscilan entre 0 y 255 es necesario convertirlas para así tener las unidades reales de medición (Amp),

lo cual se hace de la siguiente manera.

Sabemos que la unidad $d'1'$ se tiene con un voltaje de entrada de $25mV$ aproximadamente, el cual es proporcional a un voltaje en la entrada del amplificador igual a $V_e = 25mV/12.5 = 2mV$. Dado que el sensor de corriente entrega $40mV/A$, la corriente necesaria para producir la unidad en el modulo de conversión será $2mV/40mV = 50e-3$. Si multiplicamos las muestras de corriente por este resultado obtenemos las unidades reales de la medición. Para evitar realizar operaciones en el microcontrolador, se decidio enviar las muestras de corriente tal y como se obtuvieron. Una vez realizada esta operación se aprovecho la flexibilidad que ofrece una PC para la realización de operaciones matemáticas con la finalidad de transformar las muestras a las unidades requeridas. Obviamente para poder realizar la adquisición de las muestras de velocidad y corriente es necesario contar con un programa residente en la PC que permita manejar esta información. En nuestro caso se utilizo una interfaz construida por el Dr. Leonardo Romero M. perteneciente al proyecto ROCA [Romero01].

Apéndice G

IDENTIFICACIÓN

Para la realización del experimento de identificación se utilizó el enfoque de las matrices operacionales [Lázaro03], así como el *Tool Box de Matlab* para su comprobación.

Los resultados de la experimentación realizada se muestran a continuación.

```
************-MODELO ESTIMADO USANDO TOOL BOX DE IDENTIFICACION DE MATLAB-****************
Motor Izquierdo                                  Motor Derecho
K1_i            K2_i           JT_i              K1_d            K2_d           JT_d
-----------------------------------------        -----------------------------------------
34.4882         -112.163       0.0349331         36.7292         -127.128       0.0328017

***************-MODELO ESTIMADO USANDO MATRICES OPERACIONALES-****************************
Motor Izquierdo                                  Motor Derecho
K1_iop           K2_iop          JT_iop          K1_dop          K2_dop          JT_dop
-----------------------------------------        -----------------------------------------
35.1779          -114.043        0.0342482       37.0094         -128.224        0.0325534
```

De esta manera las ecuaciones 2.34 para ambos motores queda de la siguiente manera.

$$\begin{aligned} \frac{d\omega_i(t)}{dt} &= K_{2_i op}\omega_i(t) + K_{1_i op}V_{ai}(t) \\ \frac{d\omega_i(t)}{dt} &= -114.043\omega_i(t) + 35.1779V_{ai}(t) \end{aligned} \tag{G.1}$$

y

$$\begin{aligned} \frac{d\omega_d(t)}{dt} &= K_{2_{d\,op}}\omega_i(t) + K_{1_{d\,op}}V_{ad}(t) \\ \frac{d\omega_d(t)}{dt} &= -128.224\omega_i(t) + 37.0094V_{ad}(t) \end{aligned} \tag{G.2}$$

Una buena práctica cuando se hace una identificación es, simular el modelo estimado o identificado y compararlo con las mediciones hechas. De esta manera es posible observar entre otras cosas el grado de precisión de la identificación. En las figuras G.1 y G.2 se muestran estas comparaciones.

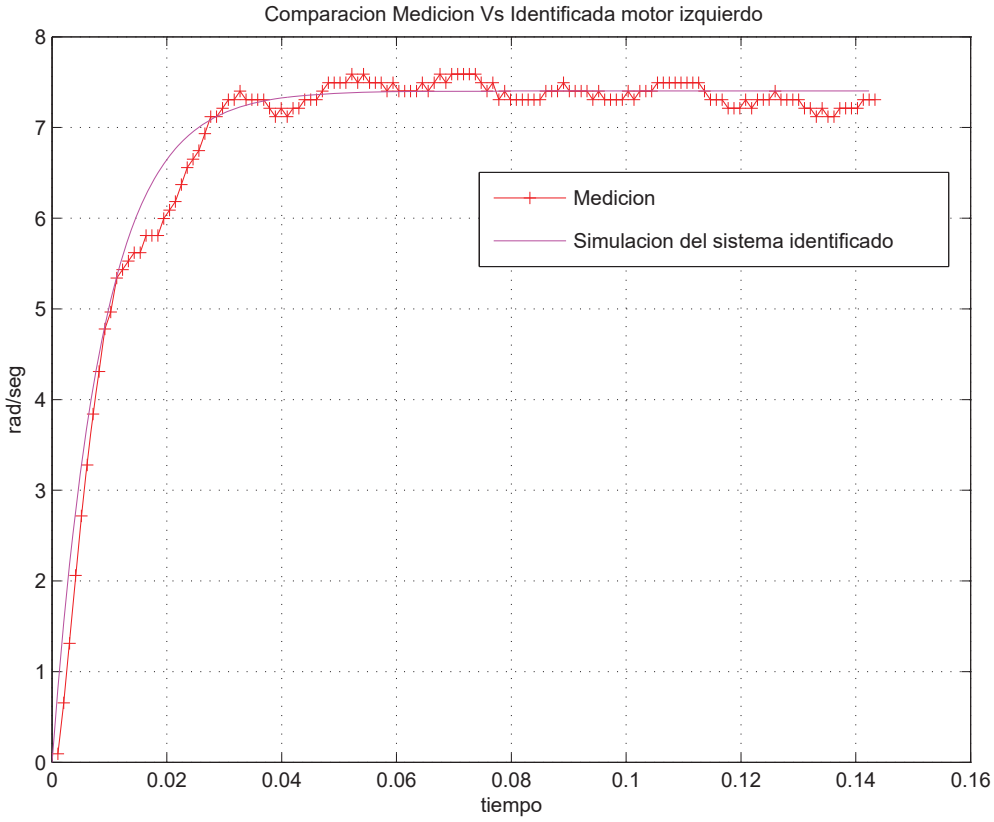


Figura G.1: Comparación para el motor izquierdo.

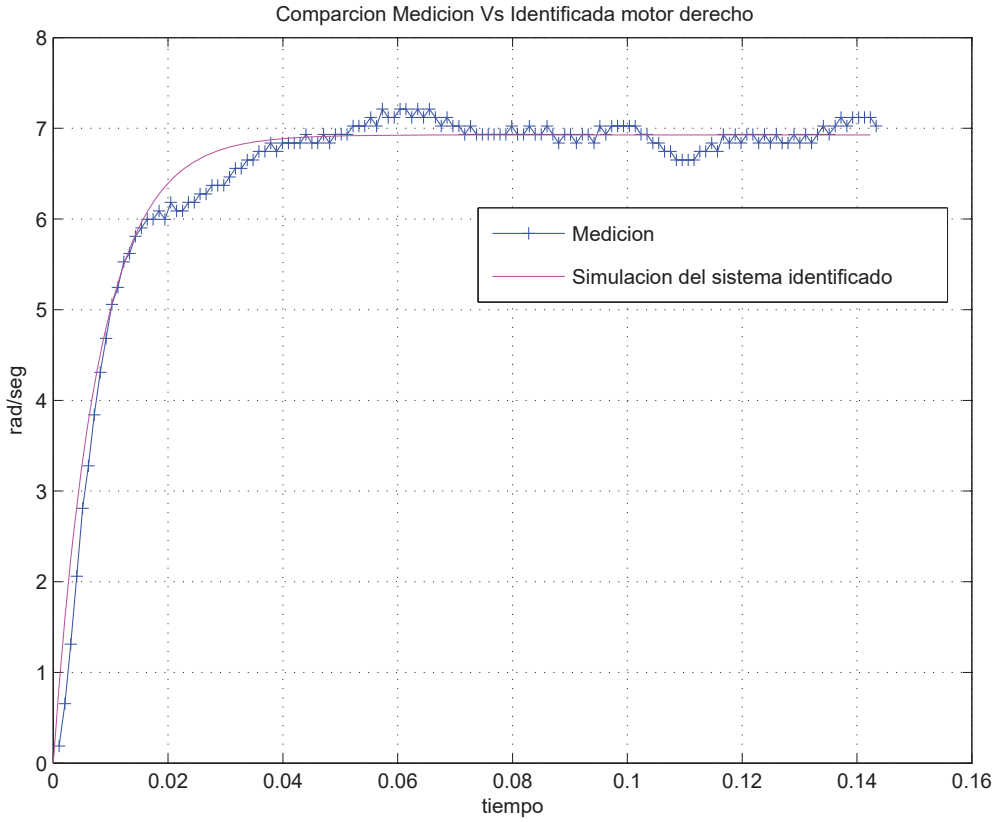


Figura G.2: Comparación para el motor derecho.

Como se puede observar en ambos resultados el modelo estimado elimina las oscilaciones debidas a la fricción no homogenea. Esto es debido a que el modelo usado para la simulación no icluye estos efectos.

Apéndice H

CRITERIO DE SELECCIÓN DE LA ROTACIÓN DEL ROBOT MÓVIL

En este apartado se describe una serie de criterios, que tienen como objetivo solucionar el problema concerniente al la rotación de la estructura del robot móvil durante el seguimiento de una trayectoria. Primeramente vamos a establecer una convención que usaremos a lo largo de esta sección y que además nos servira para su programación. Este criterio es el siguiente:

Proposición 1

Toda translación y rotación del robot móvil en sentido horario será equivalente a $+1$.

Proposición 2

Toda translación y rotación del robot móvil en sentido antihorario será equivalente a -1.

Proposición 3

La medición de angulos será en sentido antihorario.

Este último criterio, representa una base fundamental para la solución al problema del giro, es decir, dado que los arcos que conforman las trayectorias, están formados por tercias de puntos, es posible realizar una medición de ángulos tomando como referencia el primero de ellos, tales que la razón de cambio entre ángulos, nos den un indicativo de la rotación o giro requerida para navegar a travéz de esta sección de arco. Obviamente la posición y orientación inicial del robot móvil deben de ser tomadas en cuenta para una correcta interpretación. En la figura H.1 se muestra un bosquejo de lo antes comentado.

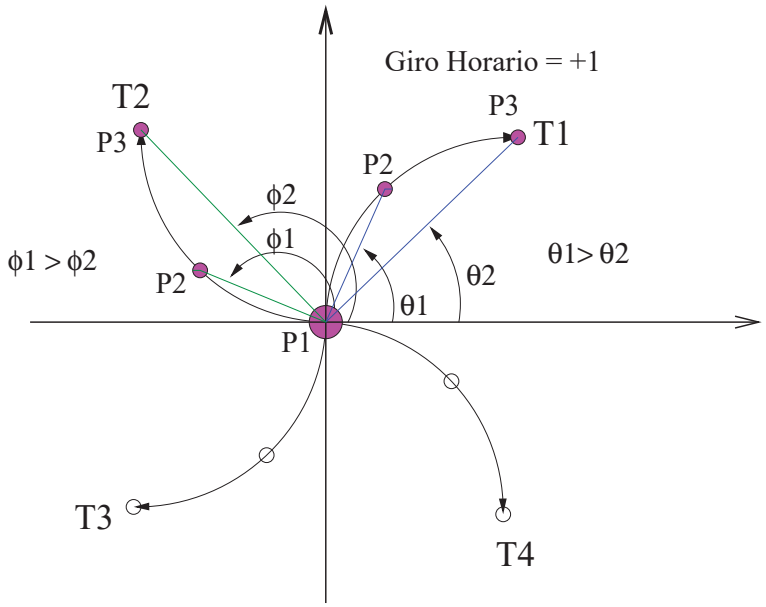


Figura H.1: Medición de ángulos en una sección de arco en sentido horario.

El sentido de la flecha de la trayectoria $T1$ (ver figura H.1) denota el sentido en que ésta es recorrida, lo cual implica un giro en sentido horario. Un patrón que se puede observar en la medición de los ángulos, es que, siempre el ángulo formado entre el eje de las ordenadas, tomando como origen el primer punto $P1$, y el punto $P2$ ($\theta 1$), es mayor que el formado con respecto al segundo punto $P2$, es decir $\theta 1 > \theta 2$. De igual forma resulta para la medición de los ángulos de la trayectoria $T2$.

En la figura H.2 se muestra el caso en que la rotación ahora es en sentido antihorario, lo cual implica un cambio en la magnitud de los ángulos ($\phi 1 < \phi 2$). De nueva

cuenta el sentido de la flecha referente esta vez a la trayectoria $T2$, indica el sentido en que esta sección de arco es recorrida. Como ya comentamos en los párrafos anteriores, la medición sistematizada de estos ángulos nos da como resultado un indicativo del sentido de rotación que es requerido para navegar a través de estas secciones de arco, no obstante existen algunas excepciones a este planteamiento, las cuales trataremos en lo consecutivo.

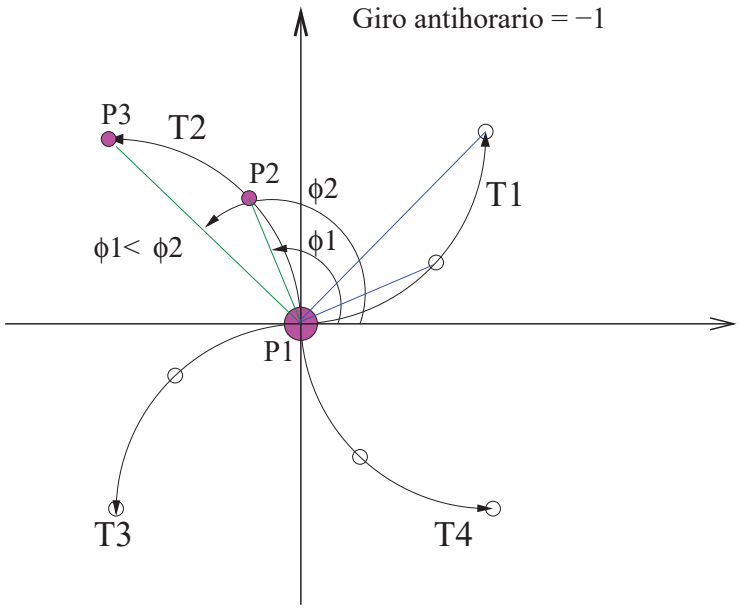


Figura H.2: Medición de ángulos en una sección de arco en sentido antihorario.

Dados tres puntos $P1$ con coordenadas (X_{p1}, Y_{p1}), $P2$ con (X_{p2}, Y_{p2}) y $P3$ con (X_{p3}, Y_{p3}), el primer paso es, como se comento anteriormente, tomar como referencia el primer punto, esto es, $P2' = (X_{p2} - X_{p1}, Y_{p2} - Y_{p1})$, de manera similar para $P3$, tenemos $P3' = (X_{p3} - X_{p1}, Y_{p3} - Y_{p1})$. De esta forma tenemos un nuevo sistema referido al punto número uno. Una vez realiado estos cálculos lo siguiente es, analizar en cual de los cuatro cuadrantes quedaron tanto $P2'$ como $P3'$, esto con la finalidad de realizar una correcta complementación de los ángulos calculados (ver ecuación H.1), ya que únicamente en el primer cuadrante obtenemos una medición correcta.

$$\theta_{p2'} = \frac{Y_{p2} - Y_{p1}}{X_{p2} - X_{p1}} \tag{H.1}$$

Dependiendo de lo anterior, podemos citar los siguientes cuatro casos de complementación de los ángulos calculados:

CASO 1:

Si P'_n se encuentra en el cuadrante I. El ángulo no sufre modificación.

CASO 2:

Si P'_n se encuentra en el cuadrante II. El ángulo más 180.

CASO 3:

Si P'_n se encuentra en el cuadrante III. El ángulo más 180.

CASO 4:

Si P'_n se encuentra en el cuadrante IV. El ángulo más 360.

A continuación, mostraremos los casos en que el planteamiento anterior tiene sus excepciones, como lo muestra la figura H.3. donde se puede observar el caso en que el punto $P2$ se encuentra en el cuadrante I, mientras que el punto $P3$ se encuentra en el cuadrante IV. Este caso representa una contradicción, en el sentido en que la medición de ángulos, según el planteamiento recién descrito implica un giro del robot móvil en sentido antihorario, no obstante es obvio que el segmento de arco requiere de un giro en sentido horario.

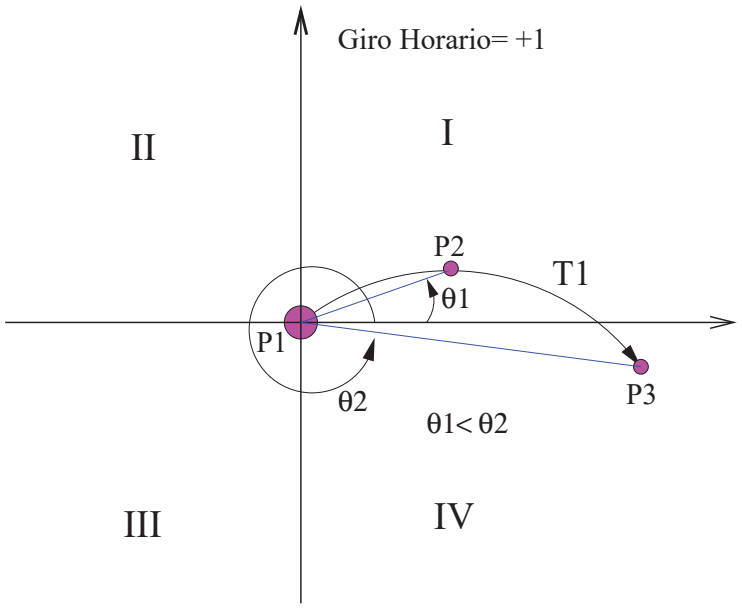


Figura H.3: Caso en que P2 yace en I y P3 en IV.

Otra excepción a este procedimiento, se presenta en el caso en que los puntos $P2$ y $P3$ se encuentran el los cuadrantes IV y I respectivamente. En este caso, como se observa en la figura H.4, según nuestro planteamiento el sentido de giro seria horario, no obstante la figura demuestra lo contrario.

La solución propuesta consiste precisamente en identificar este par de excepciones y realizar una operación negada y asi obtener una correcta interpretación de giro requerido para estas secciones de arco. Lo anterior se puede lograr de la siguiente manera. Si tomamos como bits de estado las coordenadas de los puntos $P2$ y $P3$, es decir (X_{P2}, Y_{P2}) Y (X_{P3}, Y_{P4}), tendriamos cuatro bits de estado, los cuales son interpretados como un estado alto o 1 al tener una coordenada con signo positivo, y un estado bajo o 0 para una coordenada con signo negativo. De esta forma es fácil observar que tenemos 16 posibles combinaciones, dentro de las cuales $b0111$ o $d7$ y $b1101$ o $d13$, corresponden a los casos anteriores respectivamente.

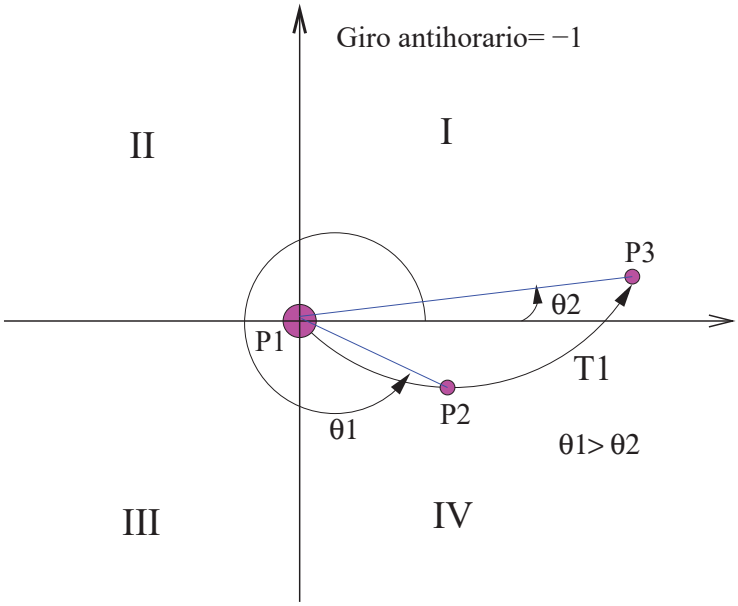


Figura H.4: Con P2 en IV y P3 en I.

Referencias

[Arellano05] Arellano, J. J. *Reconstrucción tridimensional de ambientes interiores utilizando un Robot Móvil equipado con un telémetro láserser* . Proyecto Fin de Carrera, Facultad de Ingeniería Eléctrica, Universidad Michoacana de San Nicolás de Hidalgo, 2005.

[Arts02] Arts, T., octubre 2002. Http://www.technologicalarts.com.

[Astrom90] Astrom, K. y Wittenmark, B. *Computer Controlled Systems Theory and Design.* Prentice-Hall, 2ª edón., 1990.

[Astrom95] Astrom, K. y Hagglund, T. *PID Controllers.* Instrument Society of America, 1995.

[Barsky90] Barsky. Geometric continuity of parametric curves: Constructions of geometrically continuous splines. *Parametric Curves Tutorial part two*, págs. 60–68, 1990.

[Beard02] Beard, R. W. Mathematical models for mobile robots. *Department of Electrical and Computer Engineering*, págs. 1–5, 2002.

[BELL04] BELL, F. Iha-25 hall effect, 2004. Http://www.fwbell.com.

[Bishop04] Bishop, R. H. *Learning with LabVIEW 7.0 Express.* Prentice Hall, 2004.

[Carrez03] Carrez, S. GNU for 68HC11/68HC12, 2003. Http://stephane.carrez.free.fr/. Updated 09/23/2003.

[Cordero02] Cordero, J. M. *Curvas y Superficies para Modelado Geométrico.* Alfaomega, 2002.

[Dudek00] Dudek, G. *Computational Principles of Mobile Robotics.* Cambridge University Press, 2000.

[Eaton04] Eaton, J. W. Octave, 2004. Http://www.octave.org.

[Fuller02] Fuller, G. *Geometría Analítica.* CECSA, 2002.

[Inc.04] Inc., R. H. redhat, 2004. Http://www.redhat.com/.

[Jones99] Jones, J. L. y Flynn, A. M. *Mobile Robots Inspiration to Implementations.* A K Peters, Ltd, 2ª edón., 1999.

[kheir00] kheir, M. *The M68HC11 Microcontroller.* Prentice Hall, 2000.

[Lázaro03] Lázaro, I. I. y Rico, J. J. Identificación de sistemas lineales usando cálculo operacional. *Memoria del XXV congreso internacional de Ingeniería Electrónicanica* , págs. 1021–1030, 2003.

[Luebeck04] Luebeck, octubre 2004. Http://www.maveric.

[McComb01] McComb, G. *The Robot Builders Bonanza.* McGraw-Hill, 2ª edón., 2001.

[Mot02a] Motorola Inc. *ECT-16B8C Block User Guide V01.02*, 2002. Http://www.technologicalarts.com.

[Mot02b] Motorola Inc. *MC9S12DP256B Device User V02.11*, 2002. Http://e-www.motorola.com/file/soft_dev_tools/data_sheet/9S12DP256B-ZIP.zip.

[Mot02c] Motorola Inc. *PWM-8B8C Block User Guide V01.16*, 2002. Http://www.technologicalarts.com.

[Muñoz95] Muñoz, V. F. *Planificación de Trayectorias para Robots Móviles.* Tesis Doctoral, Universidad de Malaga España, 1995.

[NASA04] NASA, octubre 2004. Http://mars.jpl.nasa.gov/MPF/rovercom/rovintro.html.

[National96] National, S. H bridge lm18200, 1996. Http://www.national.com.

[National98] National, S. Adjustable regulator, 1998. Http://www.national.com.

[Pittman01] Pittman. Dc servo gearmotor, 2001. Http://www.pittmannet.com.

[Ramirez Z98] Ramirez Z, S. *Diseño y Construcción de un Manipulador con dos Grados de Libertad y Orientación del Efector Final.* Proyecto Fin de Carrera, Facultad de Ingeniería Eléctrica, Universidad Michoacana de San Nicolás de Hidalgo, 1998.

[Ribeiro02] Ribeiro, M. I. Kinematics models of mobile robots. *Mobile Robotics course*, pág. 3, 2002.

[Rodríguez98] Rodríguez, J. E. *Introducción a la Ingeniería del Control Automático.* McGraw-Hill, 1998.

[Rom74] Rom, C. A class of local interpolating splines. *Computer Aided Geometric Design*, págs. 317–326, 1974.

[Romero01] Romero, L. *Construcción de mapas y localización de robots móviles: un enfoque probabilista.* Tesis Doctoral, Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Cuernavaca, 2001.

[Sears67] Sears, F. y Zemansky, M. *Física General.* Aguilar S.A, 4ª edón., 1967.

[Siegwart04] Siegwart. *Introduction to Autonomous Mobile Robots.* Massachusetts Institute of Technology, 2004.

Glosario

A/D Conversión analógica a digital.

AGV Vehículo guiado automáticamente.

Anti-Windup Control. Acción que evita un mal funcionamiento del controlador ante la presencia de saturación en los actuadores.

Brake Señal de control del puente H.

Dir Señal de control de dirección de los motores a través del puente H.

encoder Disco perforado que sirve como sensor de velocidad.

LabVIEW Herramienta computacional para el análisis.

MATLAB Herramienta computacional para el análisis.

Microcontrolador Circuito integrado programable.

microrover Robot móvil diseñado para la misión a Marte.

NASA del Inglés *Aeronautica nacional y administración espacial.*

pathfinder Nave espacial para la misión a Marte.

PID Control. Proporcional Integral Derrivativo.

PWM del Inglés *Modulación de ancho de pulso.*

ROCA Denominación de proyecto para robot móvil (Romero Cardiel).

RS-232 Protocolo de comunicación serial.

SCI Comunicación serial asíncrona.

SPI Interface periférica serial.

UHF Frecuencia ultra alta.

Índice