01132
50

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

ALGORITMOS DE MOVIMIENTO PARA LA OPERACIÓN DE UN ROBOT TIPO INSECTO.

T E S I S

Que para obtener el título de

INGENIERO EN COMPUTACIÓN

Presenta

Hernández López Miguel Angel

Director de tesis Dr. Jesús Savage Carmona

MÉXICO, D.F. **2003**

Agradecimientos

A los profesores de la facultad de ingeniería que tuve durante la carrera, por los todos conocimientos que me transmitieron, así como por la confianza y amistad brindada.

Al Dr. Jesús Savage Carmona por el apoyo recibido en el tiempo que tengo colaborando con él.

A la Universidad Nacional Autónoma de México y a la Facultada de Ingeniería y a todos lo miembros que la conforman.

A mis padres y a mi hermana por el apoyo y ayuda que de ellos he recibido.

B

Dedicatoria

Dedico la tesis a mi padre Jesús Hernández Villa Nueva, a mi madre María Guadalupe López y a mi hermana Beatriz Virginia Hernández López, por el apoyo que de ellos he recibido, por las palabras de ánimo que me han dado y sobre todo por el cariño que me han entregado.

ÍNDICE

CAPÍTULO 1

INTRODUCCIÓN

1.1 Descripción de la tesis

Uno de los objetivos principales de la robótica ha sido el poder reproducir la forma en como los seres vivos realizan sus movimientos, es así, que se han construido robots bípedos con la idea de imitar la manera de caminar de los seres humanos, o bien robots tipo insecto, con la misma idea de reproducir la manera en la cual se desplazan. Siendo más específicos nos referimos a los robots con patas; que entre sus grandes ventajas se encuentra la de poder librar obstáculos de una forma más sencilla, en comparación a un robot móvil con ruedas, sobre todo cuando el terreno en el que se desenvolverán no es uniforme. Por lo antes expuesto es que se ha optado por incluir un robot tipo insecto de cuatro patas, como parte de un proyecto en el que se busca la creación de un lenguaje de programación, que permita la realización de programas con los cuales poder operar cualquier tipo de robot, sin la necesidad de modificar sus códigos, salvo, cambiar algunas librerías correspondientes al robot que se utilice.

De esta manera la presente tesis tiene como objetivo la creación de los algoritmos necesarios para poder operar un robot con cuatro patas con el mismo programa con el cual se manipularía un robot móvil (con ruedas). Así como también desarrollar el hardware necesario para facilitar lo anterior. Cabe mencionar que el lenguaje de programación ya se encuentra diseñado, de tal forma que únicamente se hace una descripción de las instrucciones y de su sintaxis con que se cuenta.

El contenido en los capítulos subsecuentes tiene como propósito explicar los algoritmos diseñados, el hardware utilizado y por otro lado se busca

que quien lea el documento pueda reproducir lo aquí expuesto. De esta forma la estructura que se presenta es la siguiente, en lo que resta del capítulo se presentan algunas cuestiones referentes a la robótica y su historia, en el capítulo 2, se exponen algunos conceptos teóricos correspondientes a las áreas de compiladores, regresión lineal y al control de motores. En el capítulo 3, se presenta una descripción de las partes que componen el robot, como son: las dimensiones y forma de las piezas del robot, así como el hardware utilizado en el proyecto. En el capítulo 4 se describen los algoritmos creados, en el capítulo 5, se presentan los resultados de experimentos realizados para probar los algoritmos creados, en el capítulo 6 y de acuerdo a lo observado a lo largo del trabajo, se presentan algunas propuestas para trabajos posteriores que se pueden llevar a cabo y que mejorarían el desempeño del robot, en el capítulo 7 se presentan las conclusiones obtenidas. Por último se presentan dos anexos, en el primero se presentan las piezas de la estructura del robot en sus dimensiones reales, también el circuito impreso utilizado para multiplexar los puertos serie de la computadora y el mini SSC II, en el segundo anexo se presenta una descripción de las instrucciones del lenguaje desarrollado, así como su sintaxis.

1.2 Robótica

1.2.1 Definición

En la actualidad la palabra robot es muy común para la mayoría de las personas, pero ¿qué es en realidad un robot? La palabra ***robot*** proviene del checo "*robota*" (labor, trabajo) y "*robotnik*" (trabajador). Fue introducida por el escritor Karel Kapek en la obra "R.U.R.". Pero esto no es mas que su raíz léxica. En cuanto a lo que se refiere propiamente a una definición existen diversas al respecto entre las que se encuentran las siguientes:

- "agente activo artificial cuyo ambiente es el mundo físico" [Russell y Norvig]

- "conexión inteligente de percepción de acción" [Jones y Flynn]

- "una máquina programable capaz de percibir y actuar en el mundo con cierta autonomía" [Sucar]
- "manipulador programable y multifuncional diseñado para mover materiales, partes, herramientas o dispositivos mediante movimientos programados para realizar diferente tareas" `[Instituto de Robótica de América]

1.2.2 Antecedentes históricos

La imagen del robot como una máquina a semejanza con el ser humano, subyace en el hombre desde hace muchos siglos, es así que los primeros robots aparecen en la mitología griega y en las obras de ficción:

- *Talos*, gigante de bronce que vigilaba Creta
- *Golem*, protector de los judíos en Praga

Pero cómo poder olvidar la legendaria obra de Frankestein en la cual se presenta a un ser al cual se le vuelve a la vida por medio del uso de la electricidad, sin embargo ya en años más recientes tenemos al ya mencionado Karel Kapek quien en 1917 introduce por primera vez la palabra robot o a Isaac Asímov quien a partir de 1940 volvió a referirse a los robots en sus libros, como: "I Robot (leyes de la robótica)".

Pero a lo largo de la historia han existiendo diversos trabajos que le han dado paso a la robótica; desde los pájaros mecánicos de Hero de Alejandría, en el siglo 1 A. C. hasta nuestra era en la que incluso se han construido robots con la idea de explorar el espacio exterior, como fue el

complejo Karl Sagan. Ya en lo referente a la elaboración de robots los primeros construidos en tiempos recientes fueron:

- 1890; Tesla: vehículos radio controlados
- 1940s; Wiener: dispositivo antiaéreo
- 1950; *tortuga* electrónica
- 1966; S.R.I: *Shakey* (primer robot móvil con IA)
- 1960s; G.E., *Quadruped* (primeros robots de patas)
- 1973; Stanford: *Cart*
- 1975; Francia: *Hilare* I
- 1980; C.M.U.: *Rover*

En la actualidad los robots ya son más sofisticados, aunque todos sus elementos se pueden clasificar en alguno de los siguientes grupos:

- Actuadores: Dispositivos que permiten al robot modificar el medio ambiente, presentándose básicamente dos tipos principales:
  - Locomoción, que permiten cambiar la posición del robot respecto al medio ambiente. Existen dos formas básicas de locomoción: Ruedas (que son más eficientes y fáciles de controlar, aunque limitadas a terrenos planos) y patas (complejas, inestables y difíciles de controlar pero con mayor flexibilidad para todo tipo de terreno.

  - Manipulación, para mover otros objetos en el medio ambiente. Normalmente son construidos basándose en una serie de eslabones con articulaciones entre ellos, presentándose tres tipos básicos: rotación, cilíndricas y prismáticas; según el número de articulaciones será la cantidad de movimientos independientes que el manipulador pueda llevar acabo (a lo que se le denomina grados de libertad). Los manipuladores tienen en el extremo un efector final que interactúa directamente con los objetos, existiendo

diferentes tipos dependiendo de la tarea, pinzas, herramientas (desarmador,...), pistola de pintura, cautín, "manos", etc.

- Sistema de inteligencia, programas que permiten que el robot realice sus tareas, dependiendo del tipo de robot y de la complejidad así como la variedad de las tareas a realizar, se tienen diferentes tipos de programas.
- Sistema de control, provee la interfaz entre el sistema de procesamiento del robot y sus sensores y actuadores. Normalmente se realiza mediante una combinación de hardware y software.
- Sistema de comunicación, el cual permite a los usuarios comunicarse con el robot y modificar su operación.
- Sensores, dispositivos que permiten al robot percibir el medio ambiente y su estado interno. Los principales tipos son: posición y movimiento, codificadores en uniones de manipuladores, fuerza, táctiles, ultrasonido (sonares)
- Sistema de alimentación, proporciona la energía eléctrica para la operación de las diferentes partes: electrónica, motores, sensores, etc.

1.2.3 Clasificación, características principales y aplicaciones de los Robots

En este sentido existen de igual forma diversas clasificaciones, aquí presentamos unas de las más comunes:

- Robots manipuladores (brazos)
- Robots móviles
- Robots "híbridos" (móviles con manipulación)
- Vehículos autónomos
- Robots caminantes
  - 2 patas (*humano*)

- 4/6 patas (*insectos*)

Algunas características principales que se deben tomar en cuenta al construir un robot son las siguientes:

- Grados de Libertad
- Zonas de trabajo y dimensiones del manipulador
- Capacidad de carga
- Precisión de la repetición
- Velocidad
- Coordenadas de movimientos
- Tipo de actuadores
- La facilidad de su programación

Es así que de acuerdo a sus características físicas algunas de las aplicaciones de los robots son las siguientes:

- Manufactura y manejo de materiales
- Exploración de ambientes hostiles
- Exploración espacial e interplanetaria
- Robots de servicio
- Militares y operaciones de rescate
- Ambientes submarinos
- Estudios fisiológicos y cognitivos
- Entretenimiento y juego

En fin una gran gama de aplicaciones que sería complicado mencionar todas. Y que solo se encuentran restringidas por la imaginación que tengan las personas que los crean y adquieren.

CAPÍTULO 2

ANTECEDENTES TEÓRICOS

A lo largo de este capítulo se presentarán algunos antecedentes teóricos asociados al desarrollo de la tesis, como son cuestiones sobre compiladores, lenguajes de programación, regresión lineal, técnica de control de motores PWM.

2.1 Análisis regresión lineal

Como se verá posteriormente, la cinemática de la araña presenta básicamente un comportamiento lineal, de tal forma que para obtener las relaciones entre distancia, velocidad y desplazamiento angular en los motores, se hizo uso de rectas, es por ello que a continuación se presenta un explicación de los que es la regresión lineal y en especial el método de mínimos cuadrados.

En el análisis de regresión lineal tratamos con experimentos aleatorios que incluyen dos variables, de las cuales al menos una es aleatoria. En general se suele manejar el par (X, Y), la variable X, puede considerarse como variable ordinaria, esto es; que se puede medir sin error apreciable. La otra variable, Y, es una variable aleatoria. A X se le denomina variable independiente o controlada, nuestro interés es la dependencia que tiene Y respecto a X.

Así en los experimentos se seleccionan en primer lugar n valores x1, x2,..., xn de X, luego se observan los valores de Y que corresponden a estos valores de X, de tal manera que se obtiene una muestra de la forma (x1, y1), (x2, y2),..., (xn, yn). En el análisis de regresión, se considera que la media u de Y depende de X, es decir; que es una función $\upsilon=\upsilon(x)$ en el sentido ordinario. La curva de $\upsilon(x)$ se llama curva de regresión de Y

con base en X. En el caso más sencillo, ésta puede ser una recta representada por:

$$\upsilon(x) = mX + b$$

A ésta se le llama recta de regresión de Y con base en X y la pendiente m se llama coeficiente de regresión. Este caso lineal es muy importante, ya que cualquier función $\upsilon(x)$ puede ser aproximada con suficiente exactitud mediante una función lineal si X varía en un intervalo corto. Entonces, la muestra puede usarse para estimar los parámetros m y b. Las estimaciones puntuales de m y b se pueden obtener por el método de mínimos cuadrados. Si suponemos que:

a) para cada X fija la variable Y es normal con media mX + b y variancia α^2
b) las n ejecuciones del experimento que proporcionan la muestra son independientes

Entonces la estimación de m y b que se obtiene por medio de método de mínimos cuadrados con las estimaciones de máxima similitud.

2.1.1 Línea de regresión.

Método de mínimos cuadrados

Supóngase que en cierto experimento aleatorio tratamos de manera simultánea con dos cantidades, una variable ordinaria X y una variable ordinaria Y, cuya media depende de X, y efectuamos el experimento de tal manera que seleccionamos primero n valores X1, X2,...,Xn de X, y luego, para cada Xj seleccionada, obtenemos un valor observado Yj de Y. Entonces tenemos una muestra de n parejas de valores:

(x1, y1), (x2, y2),..., (xn, yn).

Con el fin de tener una primera impresión, podemos graficar las n parejas como puntos en el plano xy en la forma usual. En algunos casos se pueden ver que los n puntos se encuentran muy cercanos a la recta, e inclusive realizar un ajuste de la recta por medio gráficos. Como se muestra en la figura 2.1.

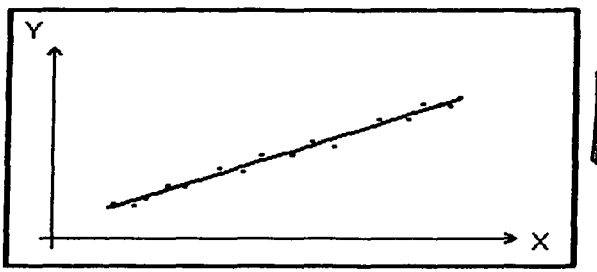


Figura 2.1 Ajuste de la recta por medios gráficos cuando los puntos cercanos.

Si los puntos están dispersos entonces ajustamos la recta (como se ve en la figura 2.2), aunque el resultando es irreal, además de que el ajuste puede variar de acuerdo a la persona, es por ello que se hace necesario utilizar un método matemático para llevar a cabo la regresión lineal. Un procedimiento de este tipo que es muy utilizado es el método de mínimos cuadrados desarrollado por Gauss. En nuestra situación presente puede formularse de la siguiente manera.

La recta debe ajustarse a los puntos dados de manera que la suma de los cuadrados de las distancias de estos puntos hasta la recta sea mínima en donde la distancia se mide en dirección vertical (en dirección de Y), según se muestra en la figura 2.3

Figura 2.2 Ajuste de la recta por medios gráficos, con puntos dispersos

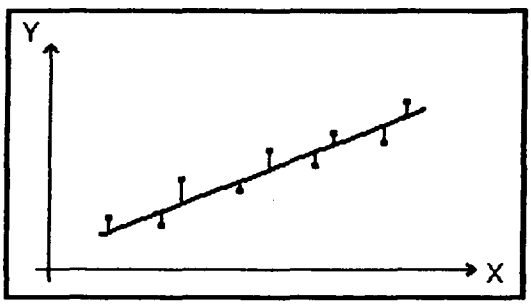


Figura 2.3 Principio de ajuste de rectas por mínimos cuadrados

Es evidente que las distancias intervienen en el método. El usar cuadrados de distancias, en lugar de valores absolutos, resulta práctico desde el punto de vista del cálculo y de la teoría. Esto hace preferible el método, en comparación con otros métodos que pueden usarse para el

mismo fin. El uso de la distancia vertical se justifica por el hecho de que X es una variable independiente, cuyos valores se seleccionan al principio del experimento, y que deseamos estimar valores de Y que corresponden a valores dados de X.

Par poder explicar en términos de fórmula. Es necesario en primer lugar, hacer la siguiente suposición:

Suposición general. Los valores de X, x1, x2,..., xn. De nuestra muestra (x1, y1), (x2, y2),..., (xn, yn) no son iguales.

La distancia vertical de un punto (xi, yi) hasta la recta

$$y = a + bx \qquad (1)$$

es [yi – a - bxi] (consultar figura 2.4), Por lo tanto, los cuadrados de las distancias verticales de los n puntos de la muestra a la recta (1) tiene la suma

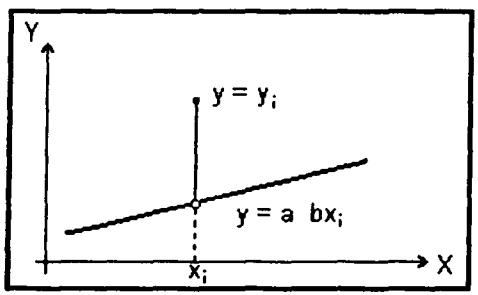


Figura 2.4 Distancia de un punto a la recta

$$q = \sum_{j=1}^{n} (y_j - a x_j)^2 \qquad (2)$$

Es decir, que es función de a y b. Para que esto sea un mínimo, es necesario que:

$$\frac{\partial q}{\partial a} = 0 \qquad \frac{\partial q}{\partial b} = 0$$

A partir de estas dos condiciones obtenemos la recta

$$y - y_p = b\,(x - x_p) \qquad (3)$$

en donde Xp y Yp son las media de los valores X y Y de la muestra, esto es:

$$X_p = \frac{1}{n}\,(X_1 + \ldots + X_n) \qquad Y_p = \frac{1}{n}\,(Y_1 + \ldots + Y_n) \qquad (4)$$

y la pendiente de la recta esta dada por la formula:

$$b = \frac{S_{x^2}}{S_{xy}} \qquad (5)$$

con el denominador :

$$S_x^2 = \frac{1}{n+1} \sum_{j=1}^{n} (X_j - X_p)^2 \qquad (6)$$

y el numerador :

$$S_{xy} = \frac{1}{n+1} \sum_{j=1}^{n} (Y - y_p)\ (x - x_p) \qquad (7)$$

La recta (3) se llama recta de regresión de los valores y respecto de la muestra con base en los valores de X de la misma. A b se le llama el coeficiente de regresión y a S_x^2 variancia de los valores x en la muestra. S_{xy} se llama covariancia de la muestra.

$$S_{xy} = \frac{1}{n+1} \left(\sum_{j=1}^{n} x_j y_j - n x_p y_p \right) \qquad (8)$$

Vemos que la recta de regresión (3) pasa por el punto (Xp, Yp)

S_{xy} puede ser positivo, cero, o negativo por lo tanto lo mismo es cierto para b. Si b es positivo (negativo), la regresión se llama regresión positiva (negativa).

2.2 Compiladores

Una parte del proyecto es la adaptación del los algoritmos diseñados al lenguaje Robel, por lo mismo se hace una descripción de lo que es un compilador y las fases que lo componen.

La escritura de compiladores comprende los lenguajes de programación, la arquitectura de computadoras, la teoría de lenguajes, los algoritmos y la ingeniería de software.

A grandes rasgos, un compilador es un programa que lee un programa escrito en un lenguaje, lenguaje de fuente; y lo traduce a un lenguaje equivalente en otro lenguaje, el lenguaje objeto. Los compiladores a menudo se clasifican de acuerdo ha como hayan sido construidos o de que función se supone que realizan en:

- De una pasada
- De múltiples pasadas
- De carga y ejecución
- De depuración o de optimización

En la compilación hay dos partes: análisis y síntesis. La parte de análisis divide al programa fuente en sus elementos componentes y crea una representación intermedia del programa fuente. La parte de síntesis construye el programa objeto deseado a partir de la representación intermedia.

Muchas herramientas de software que manipulan programas en código fuente realizan primero algún tipo de análisis, algunos ejemplos de tales herramientas son:

- Impresoras Estéticas. Una impresora estética analiza un programa y lo imprime de forma que la estructura del programa resulte claramente visible.
- Verificadores estáticos. Un verificador estático lee un programa, lo analiza e intenta descubrir errores potenciales sin ejecutar el programa.
- Intérpretes. En lugar de producir un código objeto como resultado de una traducción, un intérprete realiza las operaciones que implica el programa fuente.

Muchas veces los intérpretes se usan para ejecutar lenguajes de órdenes, pues cada operador que se ejecuta en un lenguaje de órdenes suele ser una invocación de una rutina compleja, como un editor o un compilador.

2.2.1 Análisis del programa fuente

En la compilación, el análisis consta de tres fases:

1. Análisis lineal, en el que la cadena de caracteres que constituye el programa fuente se lee de izquierda a derecha y se agrupa en componentes léxicos, que son secuencias de caracteres que tienen un significado colectivo.
2. Análisis jerárquico, los componentes léxicos se agrupan jerárquicamente en colecciones anidadas con un significado colectivo.
3. Análisis semántico, se realizan ciertas revisiones para asegurar que los componentes de un programa se ajustan de un modo significativo.

Análisis léxico. En un compilador, el análisis lineal se llama análisis léxico o exploración.

Análisis jerárquico. El análisis jerárquico se denomina análisis sintáctico. Este implica agrupar los componentes léxicos del programa fuente en frases gramaticales que el compilador utiliza para sintetizar la salida. La estructura jerárquica de un programa normalmente se expresa utilizando reglas recursivas.

El análisis semántico. Revisa el programa fuente para tratar de encontrar errores semánticos y reúne la información sobre los tipos para la fase posterior de generación de código. En ella se utiliza la estructura jerárquica determinada por la fase de análisis sintáctico para identificar los operadores y operandos de expresiones y proposiciones.

Un componente importante del análisis semántico es: la verificación de tipos; aquí el compilador verifica si cada operador tiene operandos permitidos por la especificación del lenguaje.

2.2.2 Las fases de un compilador

Conceptualmente, un compilador opera en fases, cada una de las cuales transforma al programa fuente de una representación a otra. En la figura 2.5 se muestra una descomposición típica de un compilador.

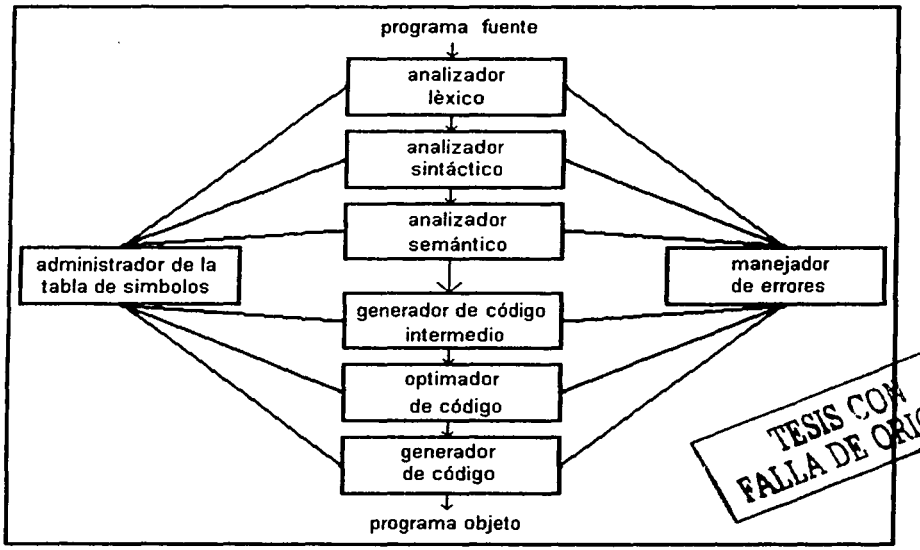


Figura 2.5 Diagrama de bloques de las fases de un compilador

Las tres primeras fases, que forman la mayor parte de la porción de análisis de un compilador. Otras dos actividades, la administración de la tabla de símbolos y el manejo de errores, se muestran en interacción con las seis fases de análisis léxico, análisis sintáctico, análisis semántico generación de código intermedio, optimización de código y generación de código.

Las fases de análisis

La fase de análisis lee los caracteres del programa fuente y los agrupa en una cadena de componentes léxicos, en los que cada componente representa una secuencia lógicamente coherente de caracteres como: un identificador, una palabra clave, un carácter de puntuación, o un operador de varios caracteres. La cadena de caracteres que forma el componente léxico se denomina lexema del componente.

Generación de código intermedio

Después del análisis sintáctico y semántico, algunos compiladores generan una representación intermedia explícita del programa fuente. Se puede considerar esta representación intermedia como un programa para una máquina abstracta. Esta representación intermedia debe tener dos propiedades importantes: debe ser fácil de producir y fácil de traducir al programa objeto.

Optimización de código

La fase de optimización de código trata de mejorar el código intermedio, de modo que resulte un código de maquina más rápido de ejecutar. Una parte significativa del tiempo de compilación se ocupa en esta fase.

Ensambladores

Algunos compiladores producen código ensamblador, que se pasa a un ensamblador para su procesamiento. Otros compiladores realizan el trabajo del ensamblador, produciendo código de máquina relocalizable que se puede pasar directamente al código de carga y enlace.

2.2.3 Análisis léxico

El analizador léxico es la primera fase de un compilador. Su principal función consiste en leer caracteres de entrada y elaborar como salida una secuencia de componentes léxicos que utiliza el analizador sintáctico.
La forma de trabajar del analizador léxico es leer los caracteres de entrada hasta que pueda identificar el siguiente componente léxico.

Componentes léxicos, patrones y lexemas

Cuando se menciona el análisis sintáctico, los términos "componente léxico"(token), "patrón" y "lexema" se emplean con significados específicos, en la tabla 1 se presentan algunos ejemplos de dichos usos.

Componente léxico	Lexemas	Patrón
id	Pi, cuenta, x, y	Letra seguida de letras
num	3.1416, 6, 5	Cualquier constante numérica

Tabla 1

En general, hay un conjunto de cadenas en la entrada para la cual se produce como salida el mismo componente léxico. Este conjunto de cadenas se describe mediante una regla llamada patrón asociada al componente léxico. Se dice que el patrón concuerda con cada regla del conjunto. Un lexema es un secuencia de caracteres en el programa fuente con la que concuerda el patrón para un componente léxico.

En la mayoría de los lenguajes de programación, se considera componente léxico las siguientes construcciones:

- Palabras clave
- Operadores
- identificadores
- Cantantes
- Cadenas literales
- Signos de puntuación

Un patrón es una regla que describe el conjunto de lexemas que pueden representar a un determinado componente léxico en los programas fuente. El analizador léxico recoge información sobre los componentes léxicos en sus atributos asociados. Los componentes léxicos influyen en las decisiones del análisis sintáctico y los atributos en la traducción de los componentes léxicos. En la práctica, los componentes léxicos suelen tener un solo atributo.

Diagramas de transición

Como paso intermedio en la construcción de un analizador léxico, primero se produce un diagrama de flujo estilizado llamado: diagrama de transiciones. Los diagramas de transición representan: las acciones que tienen lugar cuando el analizador léxico es llamado por el analizador sintáctico para obtener el siguiente componente léxico.

Las posiciones en un diagrama de transiciones se representan con un círculo y se llaman estados. Los estados se conectan mediante flechas, llamadas aristas. Las aristas que salen del estado s tienen etiquetas que indican los caracteres de entrada que pueden aparecer después de haber llegado al estado s. Teniendo como características que ningún símbolo puede concordar con las etiquetas de dos aristas que salgan del mismo estado. Un estado se etiqueta como estado de inicio; es el estado inicial del diagrama de transición donde reside el control cuando se empieza a reconocer un componente léxico. Ciertos estados pueden acciones que se

ejecutan cuando el flujo del control alcanza dicho estado. Al entrar en un estado se lee el siguiente carácter de entrada. Si hay una arista del estado en curso de ejecución cuya etiqueta concuerde con ese carácter de entrada, entonces se va al estado apuntando por la arista. De otro modo, se indica un fallo.

En la figura 2.6 se muestra un diagrama de transiciones para los patrones >= y >. El diagrama de transiciones funciona de la siguiente forma. Su estado de inicio es el estado 0. En el estado 0 se lee el siguiente carácter de entrada. La arista etiquetada con > del estado 0 se debe seguir hasta el estado 6 si el carácter de entrada es >. De otro modo, significa que no habrá reconocido ni > ni >=.

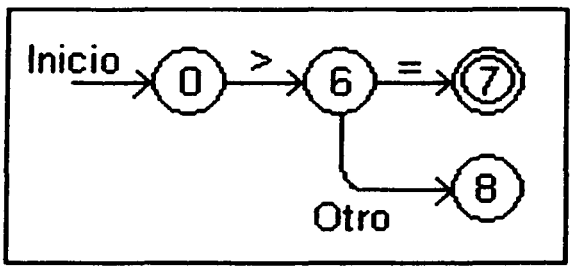


Figura 2.6 Diagrama de transición de los patrones > y >=

Al llegar al estado 6 se lee el siguiente carácter de entrada. La arista etiquetada con = que sale del estado 6 deberá seguir hasta el estado 7 si este carácter de entrada es un =. De otro modo, la arista etiquetada con **otro** indica que deberá ir al estado 8. El círculo doble del estado 7 indica éste es un estado de aceptación, un estado en el cual se ha encontrado el componente léxico >=.

2.2.4 Análisis sintáctico

Todo lenguaje de programación tiene reglas que describen la estructura sintáctica de programas bien formados. En Pascal, por ejemplo; un programa se compone de bloques, un bloque de proposiciones, una proposición de expresiones, una expresión de componentes léxicos, y así sucesivamente. Se puede describir la sintaxis de las construcciones de los lenguajes de programación por medio de gramáticas de contexto libre o notación BNF (Backus-Naur Form).

Las gramáticas ofrecen ventajas significativas a los diseñadores de lenguajes y a los desarrolladores de compiladores.

- Las gramáticas son especificaciones sintácticas y precisas de lenguajes de programación.
- A partir de una gramática se puede generar automáticamente un analizador sintáctico.
- El proceso de construcción puede llevar a descubrir ambigüedades.
- Una gramática proporciona una estructura a un lenguaje de programación, siendo más fácil generar código y detectar errores.
- Es más fácil ampliar/modificar el lenguaje si está descrito con una gramática.

Aquí surge la pregunta ¿qué es el analizador sintáctico? Es la fase del compilador que se encarga de revisar el texto de entrada en base a una gramática dada. Y en caso de que el programa de entrada sea válido, suministra el árbol sintáctico que lo reconoce. En teoría, se supone que la salida del analizador sintáctico es alguna representación de árbol sintáctico que reconoce la secuencia de tokens suministrada por el analizador léxico. En la práctica, el analizador sintáctico también suele hacer las siguientes tareas:

- Acceder a la tabla de símbolos (para hacer parte del trabajo del analizador semántico).
- Análisis de tipos (del analizador semántico).
- Generar código intermedio.
- Generar errores cuando se producen.

En definitiva, realiza casi todas las operaciones de la compilación. Este método de trabajo da lugar a los métodos de compilación dirigidos por sintaxis.

Manejo de errores sintácticos

Si un compilador tuviera que procesar sólo programas correctos, su diseño e implantación se simplificarían mucho. Pero los programadores a menudo escriben programas incorrectos, y un buen compilador debería ayudar al programador a identificar y localizar errores. Es más, considerar desde el principio el manejo de errores puede simplificar la estructura de un compilador y mejorar su respuesta a los errores. Los errores en la programación pueden ser de los siguientes tipos:

- Léxicos, producidos al escribir mal un identificador, una palabra clave o un operador.
- Sintácticos, por una expresión aritmética o paréntesis no equilibrados.
- Semánticos, como un operador aplicado a un operando incompatible.
- Lógicos, puede ser una llamada infinitamente recursiva.

El manejador de errores de un analizador sintáctico debe tener como objetivos:

- Indicar los errores de forma clara y precisa. Aclarar el tipo de error y su localización.
- Recuperarse del error, para poder seguir examinando la entrada.
- No ralentizar significativamente la compilación.

Tipo de gramática que acepta un analizador sintáctico

Nos centraremos en el análisis sintáctico para lenguajes basados en gramáticas formales, ya que de otra forma se hace muy difícil la comprensión del compilador, y se pueden corregir, quizás más fácilmente, errores de muy difícil localización, como es la ambigüedad en el reconocimiento de ciertas sentencias. La gramática que acepta el analizador sintáctico es una gramática de contexto libre, definidas como:

Gramática: G (N, T, P, S)
N = Conjunto de No terminales.
T = Conjunto Terminales.
P = Reglas de Producción.
S = Axioma Inicial.

Ejemplo: Se considera la gramática que reconoce las operaciones aritméticas.

1. E => E + T
2. | T
3. T => T * F
4. | F
5. F => ID
6. | NUM
7. | (E)

En el que:

N = {E, T, F} están a la izquierda de la regla.
T = {ID, NUM, (,), +,*}
P = Son las siete reglas de producción.
S = Axioma inicial. Podría ser cualquiera, en este caso es E.

Tipos de derivaciones

- Derivaciones: La idea central es que se considera una producción como una regla de reescritura, donde el no terminal de la izquierda es sustituido por la cadena del lado derecho de la producción.
- Derivación por la izquierda: Derivación donde solo el no terminal de más a la izquierda de cualquier forma de frase se sustituye en cada paso.
- Derivación por la derecha o Derivación canónica: Derivación donde el no terminal más a la derecha se sustituye en cada paso.

Árbol sintáctico:

Es una representación que se utiliza para describir el proceso de una sentencia. En los nodos internos del árbol, se sitúan los elementos no terminales de las reglas de producción que vayamos aplicando, y en los hijos, los símbolos que existan en la parte derecha de dichas reglas.

Veamos un ejemplo: Sea la gramática anterior.

E -> E + T | T
T -> T * F | F
F -> (E) | a | b

Supongamos que hay que reconocer: (a + b) * a + b. Si el árbol puede construirse, es que la sentencia es correcta (figura 2.7):

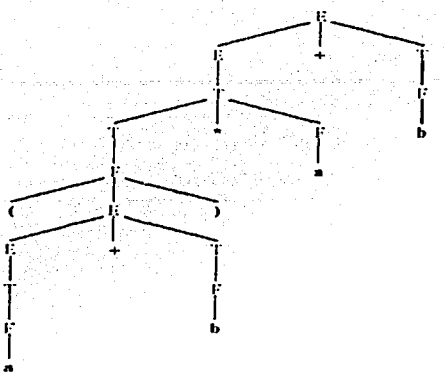


Figura 2.7 Árbol de análisis de sintáctico

Tipos de Análisis

De la forma de construir el árbol sintáctico se desprenden dos tipos o clases de analizadores sintácticos. Pueden ser descendentes o ascendentes.

- Descendentes: Parten del axioma inicial, y van efectuando derivaciones a izquierda hasta obtener la secuencia de derivaciones que reconoce a la sentencia. Pueden ser:
  - Con retroceso.
  - Con recursión.
  - LL (1)
- Ascendentes: Parten de la sentencia de entrada, y van aplicando reglas de producción hacia atrás (desde el consecuente hasta el antecedente), hasta llegar al axioma inicial. Pueden ser:
  - Con retroceso.
  - LR (1)

2.3 Control de motores de DC

Una parte fundamental del robot son sus actuadores, que no son más que 8 servomotores los cuales se encargan de levantar, bajar y girar cada una de las patas, internamente los servomotores cuentan con un motor de corriente directa y una etapa de control y potencia, motivo por lo cual se a continuación una descripción de los motores de corriente directa y de la técnica de control PWM.

2.3.1 Características generales de un motor de DC

Los elementos que permiten generar el movimiento a partir de la red eléctrica son los motores eléctricos, que pueden clasificarse en cuatro grandes familias atendiendo a su principio de funcionamiento. Estas familias son:

1) Motores síncronos
2) Motores asíncronos de inducción
3) Motores de corriente continua (o DC)
4) Motores de colector

De todos estos tipos, el que tradicionalmente ha sido más utilizado para funcionar en régimen de velocidad variable es: el motor de DC (figura 2.9), ya que sus características permiten variar su velocidad de una forma relativamente sencilla manteniendo unas prestaciones satisfactorias.

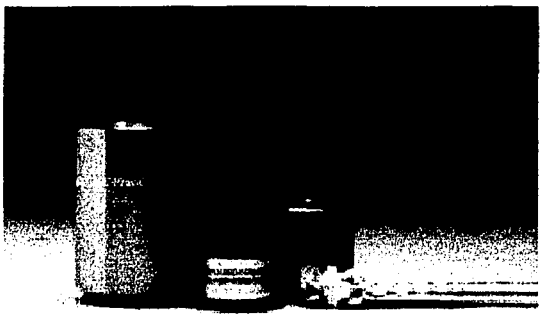


Figura 2.9 Motores de corriente directa

En el motor DC, los bobinados del estator están recorridos por corriente continua y son los encargados de generar el campo magnético, por lo que se les denomina bobinas del inductor. Por su parte, el motor esta constituido por un cilindro de material ferromagnético sobre el que se colocan los bobinados del inductor, los cuales se conectan eléctricamente con el colector de la máquina. Unas escobillas fijas en el espacio permiten el contacto eléctrico entre el colector y el circuito exterior. Estas piezas se muestran en las figuras 2.10 y 2.11.

Como en cualquier motor eléctrico, el par de giro que aparece en el eje se debe a la interacción de los campos magnéticos creados por el inductor y el inducido, por lo que el motor no funcionará si uno de los dos devanados no está correctamente alineado.

Tradicionalmente, la máquina de corriente continua ha sido la más utilizada en aquellas aplicaciones que requieren variación de velocidad; ya que resultan muy sencillas de controlar incluso con métodos clásicos.

Figura 2.10 Imán permanente de un motor

Figura 2.11 Esquema del robot de un motor

Si no se consideran los fenómenos de saturación, se demuestra que las ecuaciones que rigen su funcionamiento son las siguientes:

$$E = K_v \, i_{exc} \, N \qquad (9)$$

$$T = K_t\, i_{exc} \quad (10)$$
$$U = E + RI + L\,(dI/dt) \quad (11)$$

Donde

- E: Fuerza Electromotriz
- T: Par de motor
- N: Velocidad
- U: Tensión en los bordes del inducido
- I: Corriente en el inducido
- K_v: Constante de fuerza electromotriz
- K_t: Constante de par
- i_{exc}: Corriente en el inductor

En régimen permanente de funcionamiento y considerando que la alimentación es perfectamente continua, el término diferencial de la ecuación (Ldt/dt) es nulo, por lo que la velocidad del motor puede expresarse:

$$N = \frac{U - RI}{K i_{EXC}} = \frac{U}{K i_{EXC}} \quad (12)$$

Ya que el término R-I suele ser muy pequeño frente a la tensión de alimentación. De esta expresión se deduce que tenemos dos mecanismos de variación de velocidad en un motor de corriente continua, que actúan sobre la tensión de alimentación (U) y/o sobre la corriente del inductor ($i_{exc.}$).

La evolución de las variables más destacables en la máquina, según el rango de velocidades en que trabaje es la que se muestra en la figura 2.12

El par y la corriente en la máquina dependen de la carga. En las curvas de la figura 2.12 se han considerado el par y la corriente máximos sin que se sobre cargue el motor.

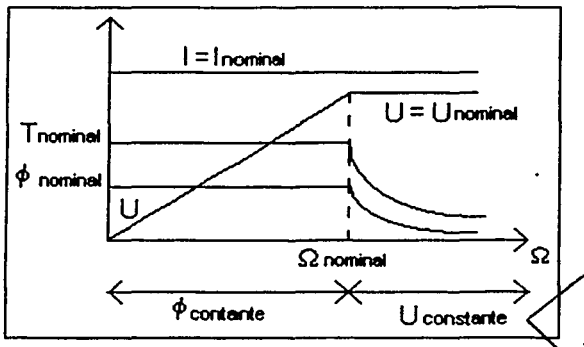


Figura 2.12 Curvas características de un motor

2.3.2 Convertidores eléctricos para el control de motores DC

La función de un convertidor electrónico es adaptar la naturaleza de la fuente de energía disponible a las necesidades del motor, de acuerdo con las presentaciones que se desean obtener. Las estructuras básicas que se utilizan son dos:

- Rectificadores. Se suelen utilizar en aplicaciones de gran potencia, donde la fuente de suministro es la red eléctrica de distribución.
- *Choppers.* Son conversores conmutados de DC-DC, se emplean en aplicaciones de potencias medias o bajas

En éste caso y debido al tipo de aplicación se mostrara únicamente el análisis de los *Choppers*.

Choppers

Para controlar la velocidad de un motor de DC mediante un conversor conmutado, se recurre a una configuración en puente completo (o puente H). En este caso, la velocidad del motor se controla variando el valor medio de la tensión aplicada al motor, para ello los interruptores del motor son conmutados a on y a off de acuerdo con un cierto ciclo de trabajo. Se denomina secuencia activa aquella en el que el estado de los interruptores es tal que el motor está conectado con la tensión de alimentación V_R. Cuando el motor se aísla de la alimentación y permanece en corto circuito, estamos en lo que se denomina secuencia de libre circulación.

La tensión de salida puede expresarse, a la vista de la figura 2.13 como

$$V_o(t) = V_{12}(t) = V_{1N}(t) - V_{2N}(t)$$

De acuerdo a lo anterior, se pueden generar dos estrategias de control de los interruptores que conducen a dos formas de onda de la tensión de salida distintas: PWM bipolar y PWM unipolar. Las principales características de ambos tipos de control se exponen a continuación.

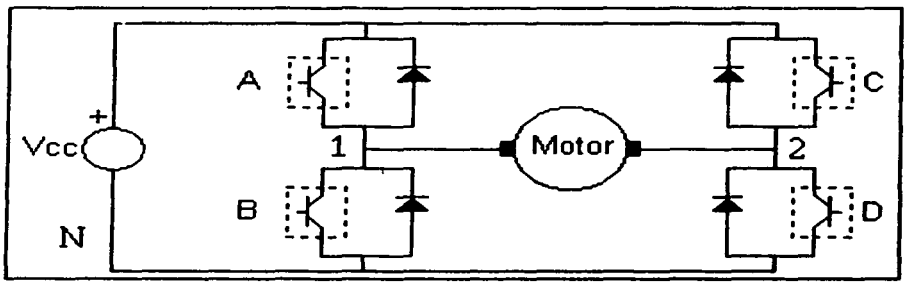


Figura 2.13 Esquema de control de un motor de CD

PWM bipolar

La modulación bipolar permite obtener tensiones de salida tanto negativas como positivas En este tipo de modulación nos encontramos con dos posibles secuencias de funcionamiento que se muestran la figura 2.14

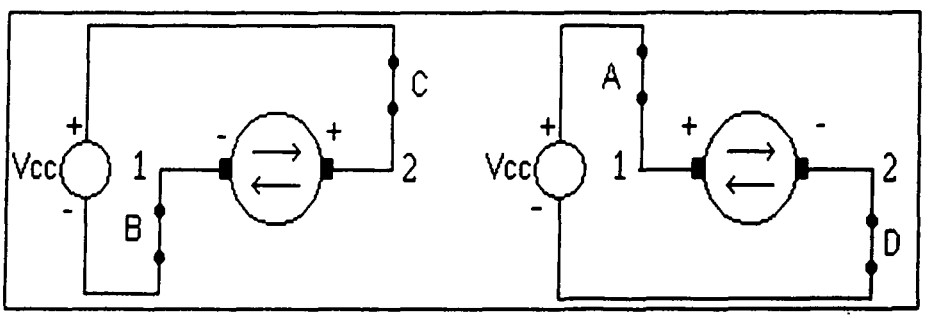


Figura 2.14 Secuencias de modulación bipolar

PWM unipolar

Con este tipo de modulación la tensión que se aplica al motor varía entre cero y la tensión de alimentación, tal como se indica en la figura 2.15.

Las diferentes secuencias de funcionamiento, en función de los sentidos de los semiconductores que conduzcan son las siguientes:

En los estados 1 y 3, la alimentación y el motor están conectados mediante los interruptores correspondientes. En los estados 2 y 4 el motor permanece en circuito corto a través de los diodos en antiparalelo. Estas

secuencias se denominan de libre circulación y son pasivas; no existiendo transferencia de energía entre la entrada y la salida.

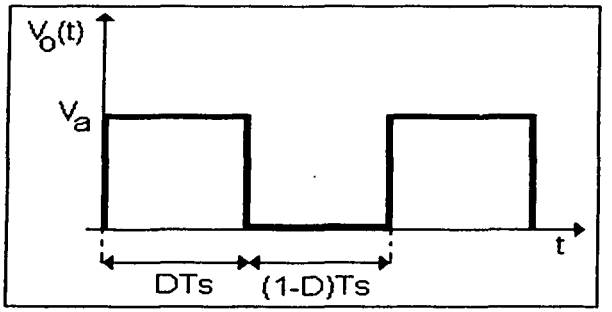


Figura 2.15 Tensión de alimentación de un motor

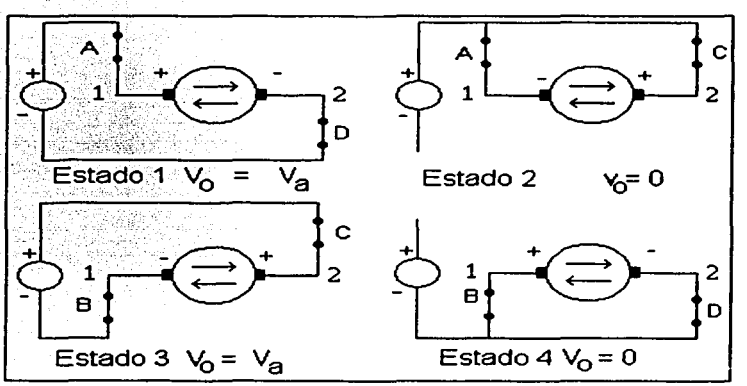


Figura 2.16 Secuencias de modulación unipolar

CAPÍTULO 3

DESCRIPCIÓN DE COMPONENTES

En este capítulo se presenta una descripción de los elementos que se utilizaron en el proyecto. En la figura 3.1 se puede apreciar un diagrama de bloques del robot, en este se ven los elementos que tiene el robot, como son los servomotores, tarjeta *Mini SSC II Serial Servo Controller*, circuito para el realizar el multiplexado de dos puertos serie, que son los que se explican en el siguiente capitulo, además de las partes que componen las patas del robot. El único componente del diagrama de bloques que no se explica es la tarjeta del microprocesador 68HC11.

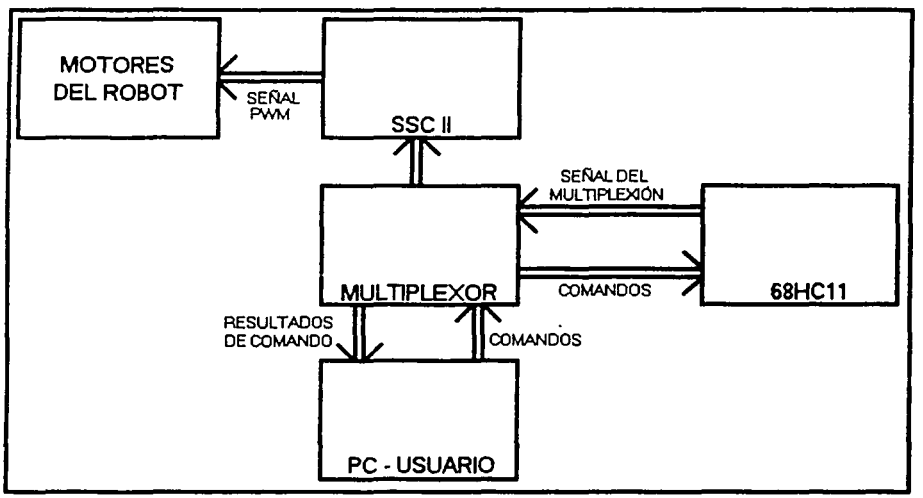


Figura 3.1 Diagrama de bloques del robot

3.1 Servomotores

Un servomotor (figura 3.2) es un dispositivo en forma de caja negra al que llegan tres cables. Contiene un pequeño motor, una caja de engranajes, un potenciómetro de un valor aproximado de 5Kg y un pequeño circuito integrado.

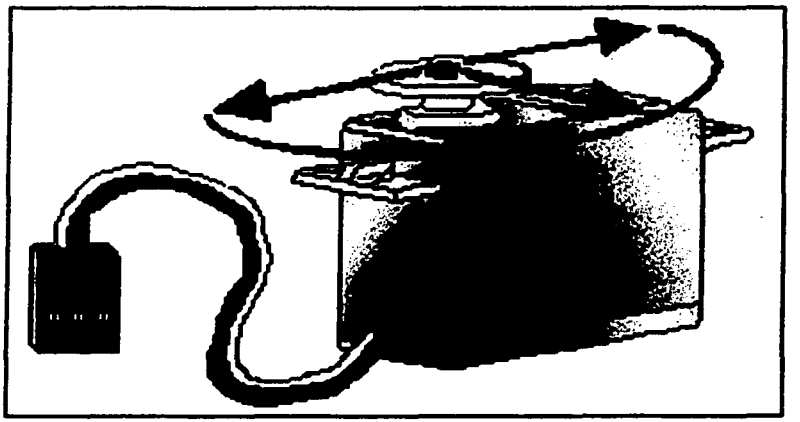


Figura 3.2 Servomotor

Este motor eléctrico en miniatura ataca a la magnitud que se ha de controlar: el giro y posicionamiento del eje del motor. A su vez, el movimiento de rotación angular del motor modifica la posición del potenciómetro interno, que controla un monoestable, también es integrado en el servomotor.

El eje del motor puede ser girado hasta una posición angular específica mediante una señal de control. Mientras se mantenga esta señal de

control, el servomotor mantendrá la posición angular del eje. Si la señal de control cambia, también cambia la posición de eje.

Conexiones

Los servomotores tienen tres cables: el de tierra, el de alimentación y el de la señal de control (figura 3.3). El positivo se conecta a + 5 y el de señal de control a una fuente de pulsos variables entre 1 y 2 milisegundos de duración que se repiten con una frecuencia de unos 12-20 ms. Los cables de los «servos» siguen casi siempre el mismo código de colores; por ejemplo en los Futaba el color rojo (V+), negro (tierra) y blanco (señal de control). Los fabricantes JR y Graupner colocan el cable de la señal de control de color naranja, mientras que algunos «servos» Sanwa tienen el cable de masa de color azul.

Figura 3.3 Estructura de interna de un servomotor

Funcionamiento

La velocidad del motor, así como la dirección del movimiento de los servos se controla mediante pulsos modulados en amplitud. En la Figura 3.4 se muestra la forma de estos pulsos. El servomotor convierte los pulsos en un movimiento mecánico. La magnitud del giro del eje del servo es proporcional a la anchura del pulso que llega por la línea de control. Este tipo de pulsos está formado por una señal digital que se genera aproximadamente cada 20 milisegundos, la anchura de estos pulsos va de un mínimo de 1 ms a un máximo de 2 ms.

Aunque la relación anchura del pulso y la posición del eje no está estandarizada, lo normal es que trenes de pulsos de 1,5 ms lleven el eje del servo al centro de su rango, anchura neutra. Si la anchura del pulso es de 1 ms, el servomotor se posiciona en el extremo izquierdo, mientras que si el pulso tiene una anchura de 2 ms la posición del servo es el extremo opuesto. Esta técnica se conoce como modulación por anchura de pulso, en ingles PWM (Pulse Width Modulation).

El servomotor (figura 3.3) trabaja comparando la anchura del pulso de entrada con la anchura del pulso producido por el *timer* interno. A su vez, el período del *timer* interno es controlado por el potenciómetro acoplado al eje del servo. La diferencia entre la anchura del pulso de entrada y la anchura del pulso interno, se utiliza como señal de error. La lógica del servo se encarga de determinar la dirección en la que ha de girar el motor para minimizar dicho error. Para ello activa los *drivers* de salida apropiados. El motor girará modificando la posición del potenciómetro de retroalimentación. Cuando llega el siguiente pulso se vuelve a realizar la comparación, comprobando de forma continua la posición del eje y realizando también constantemente las correcciones necesarias en la posición del mismo. Como se ha podido apreciar, se trata de un bucle de

retroalimentación negativa. Si la posición del potenciómetro no se iguala con la posición deseada del eje, el motor se moverá hacia adelante o hacia atrás, hasta que la posición del potenciómetro sea equivalente a la posición deseada del eje. En este momento la corriente del motor se apaga. La precisión al posicionarse depende tanto de la precisión del potenciómetro como de la precisión de la anchura de los pulsos que llegan al motor. La mayoría de los modelos

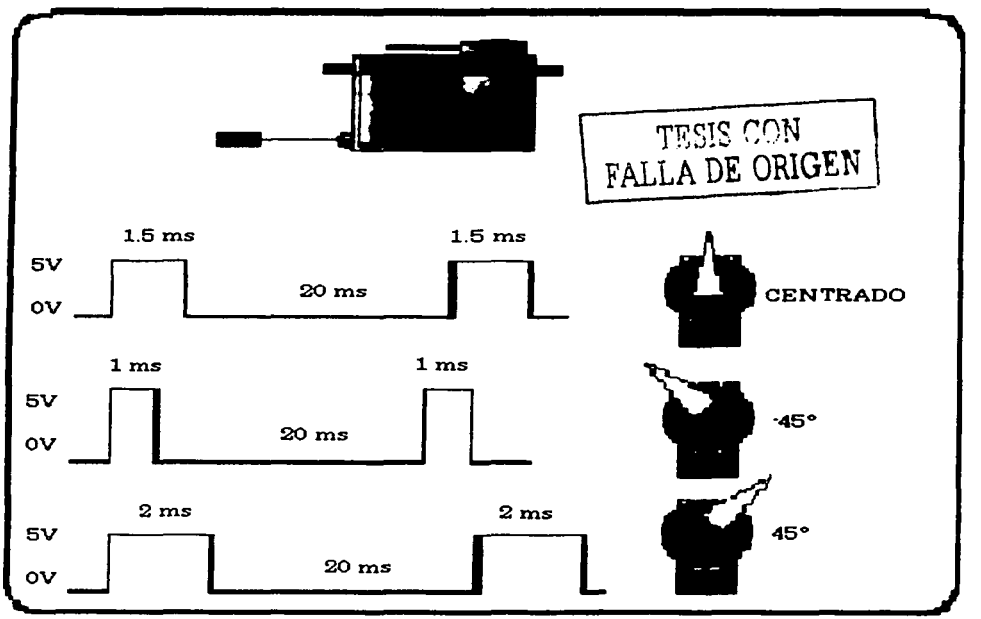


Figura 3.4 Control de la posición de un servomotor por modulación de pulsos

de servomotores consiguen una resolución de 0,5 grados. Cuando se reduce la señal de error a un nivel aceptable, el eje del servo se encuentra en la posición correcta. En ese momento la señal de error suele ser de unos 5μs, diferencia entre el ancho del pulso de la señal de entrada y el ancho del pulso de la señal interna. Esto se corresponde con una fracción de grado del recorrido del servomotor. Al ser el cero demasiado crítico, cuando el error está en este rango, conocido como zona muerta, el servo apaga los *drivers* del motor.

Si la señal de error no está por debajo de estos 5μs, la electrónica interna continuará intentando cancelar el minúsculo error, haciendo girar el motor atrás o adelante en un movimiento conocido como ***hunting***. La electrónica interna tiene como misión: mantener la anchura de los pulsos del monostable interno igual a la anchura de los pulsos de entrada.

Debido a que hay una relación fija entre el ángulo de rotación del potenciómetro y la anchura del pulso interno, la magnitud de rotación del servo se puede controlar directamente con la anchura de los pulsos aplicados. En conclusión, el circuito electrónico integrado en el motor convierte la anchura del pulso de entrada en una posición determinada del eje de salida.

Esquema de control

Hay dos formas de contemplar este tipo de esquemas de control. Desde el punto de vista del controlador, es un sistema de lazo abierto. No existe retroalimentación entre el servomotor y el sistema que genera los pulsos. Desde el punto de vista del nivel local (interior del servo) es un sistema de bucle cerrado. La electrónica del servomotor está constantemente tratando de eliminar la diferencia entre los comandos y la posición actual.

Esta doble personalidad es una característica muy importante, ya que el «servo» necesita una atención mínima por parte del controlador, pero a su

vez de forma constante resiste activamente corrigiendo las influencias externas que pueden llevar el eje lejos de la posición ordenada.

Aunque los servos son los posicionadotes casi ideales, son también fáciles de modificar para aplicaciones especiales. Por ejemplo, se puede alterar el circuito de retroalimentación para modificar el rango de giro. La mayoría de los servomotores se han diseñado para un viaje de unos 90°, pero en muchos casos esta limitación puede superarse.

Cuando se necesite mayor cantidad de giro de la que el fabricante ha dotado al servo, la mejor solución es actuar modificando el potenciómetro del circuito de retroalimentación. Para ello se añade una resistencia de un valor comprendido entre 1Kohm a 2Kohm en serie con cada extremo del potenciómetro y luego se vuelve a montar. De esta forma los pulsos de la señal de control aumentarán el rango de giro.

También se puede modificar el servo para que se comporte como un pequeño motor controlado mediante pulsos. Si se quita el potenciómetro interno y se sustituye por dos resistencias de 2Kohm, el circuito interno creerá que el eje del motor se encuentra siempre en posición centrada, así pues, si se envía señal de control para que se posicione a la derecha, el «servo» tratará de corregir continuamente la posición y girará en ese sentido. Se tiene de este modo un motor con engranajes cuya dirección de rotación puede ser controlada por un tren de pulsos mediante la técnica PWM.

Por otro lado, si se aplican señales analógicas al circuito de retroalimentación se pueden mezclar los efectos del control digital remoto con el ajuste analógico local. Estas modificaciones se detallarán extensamente en posteriores artículos, ya que para los hexápodos los «servos» se emplean tal y como los diseñó el fabricante.

3.2 Mini SSC II

El Mini SSC II es un módulo electrónico que puede controlar ocho servomotores por medio de la generación pulsos proporcionales a la posición deseada de acuerdo a instrucciones recibidas por medio de comunicación serial, siendo posible conectar dos o mas tarjetas para poder realizar el control de una mayor cantidad de motores por medio de la misma línea de comunicación

3.2.1 Conexión y configuración de *jumpers*

En la figure 3.4 se muestras el esquema del Mini SSC II la tarjeta del circuito con las localización de conectores y la configuración de *jumpers*.

Configuración del Mini SSC II

El Mini SSC II tiene una configuración predeterminada (la cual se presenta sin un solo jumper de configuración) que es:

- Una comunicación de 2400 baudio
- Manipulación de 8 servomotores
- Desplazamiento angula de movimiento = 90°

Para cambiar alguno de los parámetros mencionados, solo es necesario colocar un jumpers entre los pines correspondientes, como se muestra en la figura 3.6. Las opciones de configuración son:

Rango: Sin jumper en los pines "R", el Mini SSC II puede controlar los servomotores en un rango de 0° a 90°, cada una de las posiciones se expresan por medio de un valor código de 0 a 254, existiendo 0.36° entre

las posiciones. Con jumper los pines R, el rango es de 0° a 180°, correspondiendo a 0.72° entre cada cambio en posición.

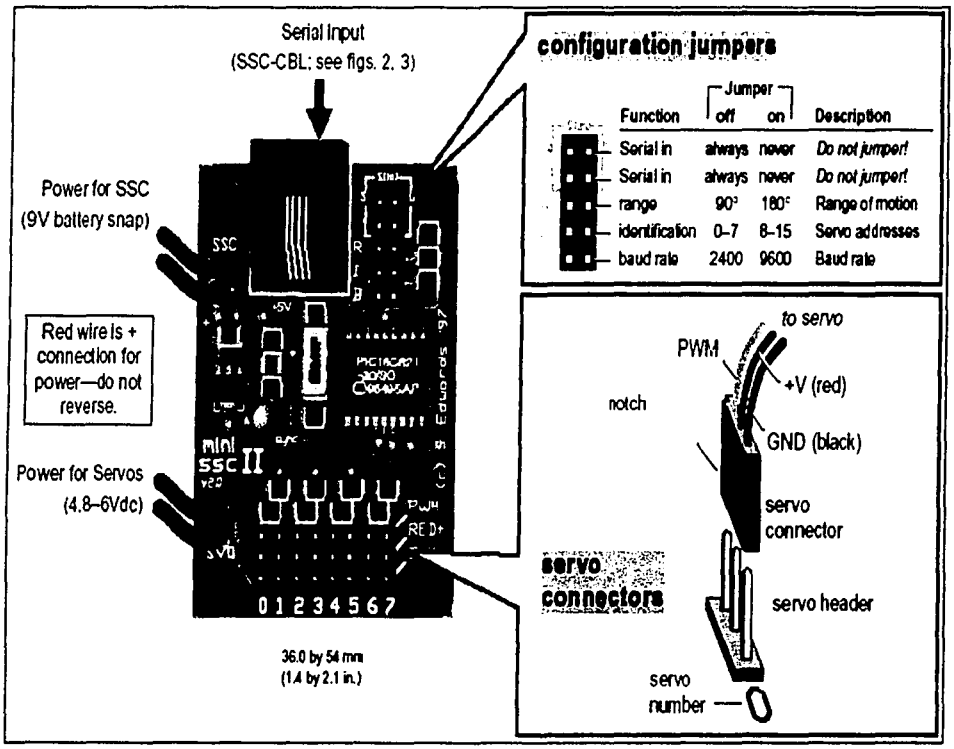


Figura 3.5 Tarjeta mini SSC II

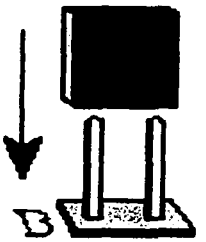


Figura 3.6 Jumper de configuración

Cabe hacer notar con respecto al rango, que algunos servomotores no se pueden mover a través de un 180° rango.

Identificación: Sin el jumper en los pines "I", la tarjeta pude controlar 8 motores que estarán identificados de 0 a 7. Con un jumper, el Mini SSC agrega 8 a estas direcciones, para que el servo conectado en la posición de 0 sea el octavo y así sucesivamente hasta estar colocado en la posición 7 al que le corresponderá el valor de. Esto le permite conectar dos Mini SSC al mismo puerto de serie y dirige cada uno de los servomotores de manera individual.

Baud: Sin el jumper en los pines B, el Mini SSC II reciben datos del puerto serie a 2400 baudio; con jumper la comunicación es a 9600 baudios. En cualquiera de los casos, los datos deben ser enviados en grupos de 8 bits, sin paridad, 1 (o más) bit(s) stop; abrevió N81.

Programación del Mini SSC II

Para mover un servomotor a una nueva posición, se requiere enviar a la tarjeta los siguientes tres bytes de forma serial y en el siguiente orden:

1. De sincronización, el primer byte a transmitir debe tener un valor de 255, y establecer la sincronización.
2. Número de motor, el segundo dato a trasmitir es el correspondiente al servomotor que se desea mover.
3. De Posición, en éste byte se envía la posición en la que se desea colocar el servomotor, el dato a trasmitir deberá estar en el rango de 0 a 254 de a acuerdo a la posición angular deseada de 0 a 90 o de o a 180 grados, de acuerdo a la configuración que se tenga.

Teoría de Funcionamiento

La tarjeta se diseño para poder controlar servomotores que puedan ser operados por medio un control PWM, en donde el ancho de pulso va de 1 a 2 milisegundos, con una frecuencia de 60 ciclos por segundo, de tal forma que la posición del servomotor es proporcional al ancho del pulso, como se muestra en la figura 3.7

Cuando se envían pulsos dentro del rango de 1 a 2 ms, se consigue un movimiento de 0° a 90° (o de 180°).

En la figura 3.8 se presenta un diagrama de conexiones de la tarjeta de control Mini SSC II con los servomotores, comunicación y alimentaciones (9 y 5 volts).

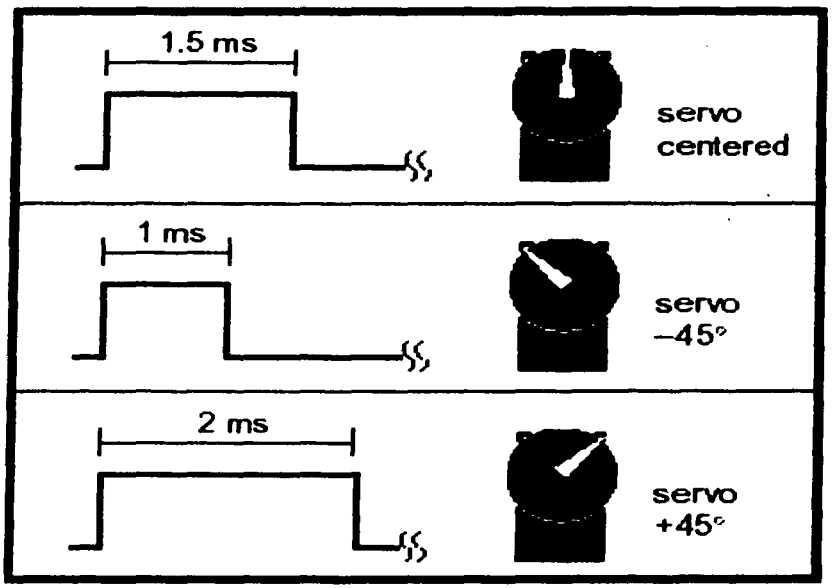


Figura 3.7 Cambio de posición en un servomotor
de acuerdo al ancho de pulso de la modulación

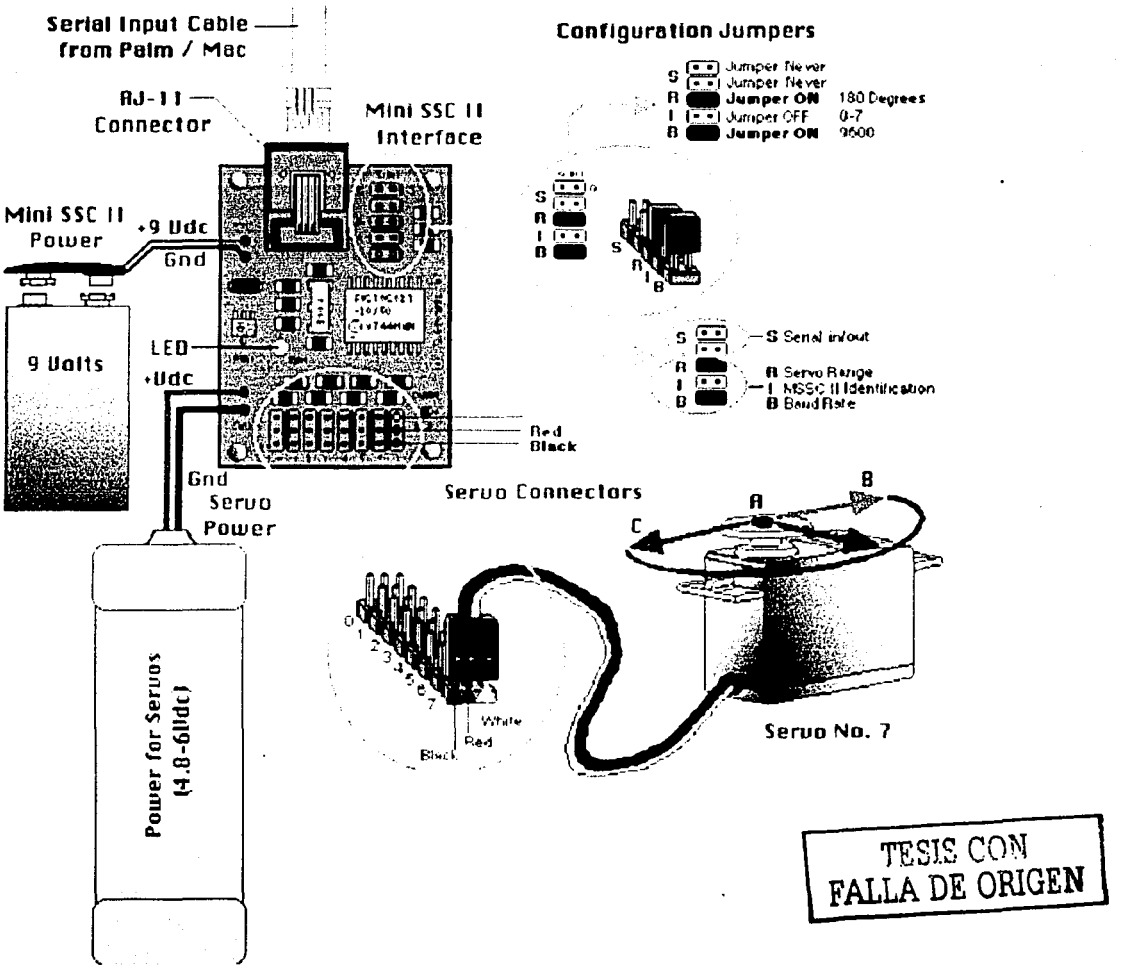


Figura 3.8 Esquema de conexión del la tarjeta Mini SSC II

3.3 Eslabones

A continuación se presenta de manera detenida las partes que componen la estructura del robot (la figura 3.9), en particular los que se refiere a la forma que tiene las piezas que componen las patas (figura 3.10 y 3.11)

Como se puede ver en las figuras 3.10 y 3.11 cada una de las patas esta compuesta de por 6 eslabones, aunque de estos, tres son iguales. En la figura 3.12, se pueden ver los 4 diferentes tipos con que se cuentan. En el apéndice A se pueden ver todos los eslabones en su tamaño natural y con las cotas respectivas.

El motor que realiza la función de levantar la pata, actúa directamente sobre el eslabón numero dos, como se puede ver en la figura 3.10, aunque por la arquitectura de esta el efecto se traslada a todos los eslabones. En lo que se refiere al giro en sentido horizontal, el motor su une directamente a las base que forman el cuerpo del robot, permitiendo así el desplazamiento hacia todas direcciones.

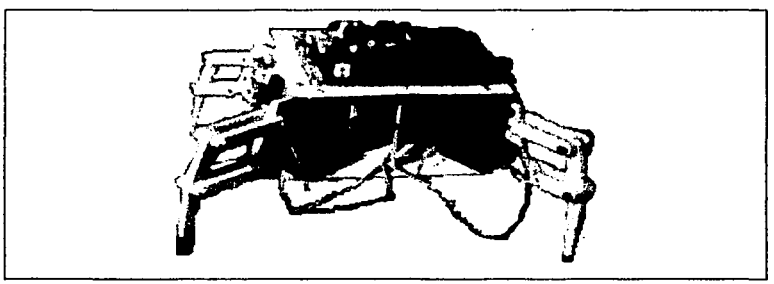


Figura 3.9 Vista diagonal del robot

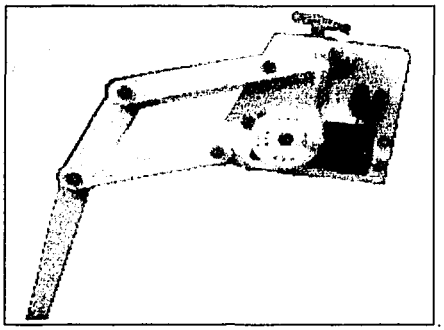


Figura 3.10 Vista frontal de una de las patas del robot

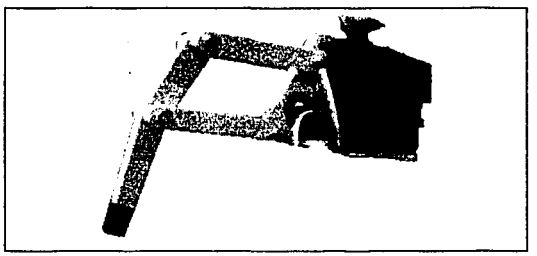


Figura 3.11 Vista posterior de una de las patas del robot

3.4 Circuito multiplexor para el puerto serie

Debido a que el microprocesador 68hc11 F1 solo cuenta con un puerto serie, aunado a la necesidad de tener que realizar la comunicación con el PIC de la tarjeta mini SSC II y con una computadora, es que fue preciso diseñar de un circuito que hiciera posible ésta tarea.

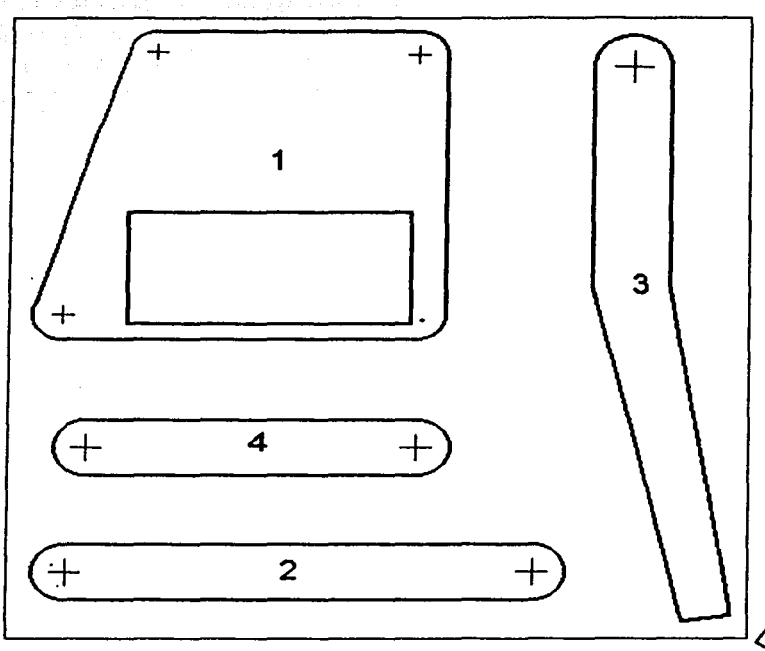


Figura 3.12 Eslabones de las patas del robot

3.4.1 Formato de comunicación serie

Las comunicaciones serie se realizan en dos formatos básicos: asíncrono o síncrono. Ambos formatos son similares, en cuanto que requieren una estructuración de la información a transmitir, es decir; ambos formatos requieren una información adicional de protocolo. La mayor diferencia entre ambos es que en el formato asíncrono la estructuración de la información se realiza por cada uno de los caracteres que intervienen en la comunicación, mientras que en el formato síncrono la estructuración

de la información se realiza por cada bloque de datos o mensajes. Por lo tanto, el sistema de formato síncrono es más eficiente que el asíncrono, aunque requiere una codificación más compleja. El formato síncrono es utilizado para enlaces de comunicación de alta velocidad, mientras que el asíncrono se emplea más para sistemas de baja velocidad de comunicación.

Por otro lado, dentro de las comunicaciones serie, se hace una distinción entre comunicaciones half duplex y full duplex. Un sistema half duplex es aquel en el que existe una única línea compartida para transmisión y recepción. Unas veces irá la información por esta línea en un sentido, y otras veces en el sentido contrario, con lo que no se puede transmitir y recibir al mismo tiempo en half duplex. Por contra, en sistemas full duplex existe una línea dedicada a transmisión y otra a recepción, lo cual posibilita la transmisión y la recepción simultáneas.

El formato asíncrono comienza con los bits de datos básicos que serán transmitidos, pero añadiéndoles delante un bit de start, o inicio, y uno o más bits de stop, o parada, al final de los mismos (figura 3.13). El bit de start es un '0' lógico o ESPACIO, mientras que el bit de stop corresponde con un '1' lógico, o MARCA. El bit de start indica al receptor el comienzo de un carácter y permite al receptor sincronizarse con el transmisor. Después de esta sincronización, la terminación depende de la duración del carácter (el siguiente carácter contendrá un nuevo bit de start). Por otro lado, es necesario añadir uno o más bits de stop para asegurar que el bit de start del siguiente carácter causará una transición en la línea de comunicación En comunicaciones asíncrononas no es necesario transmitir la señal de reloj junto con los datos, basta con que transmisor y receptor trabajen a la misma frecuencia. Tampoco es necesario que los relojes del transmisor y receptor tengan exactamente la misma frecuencia (dentro de unos ciertos márgenes de tolerancia) para una satisfactoria comunicación asíncrona.

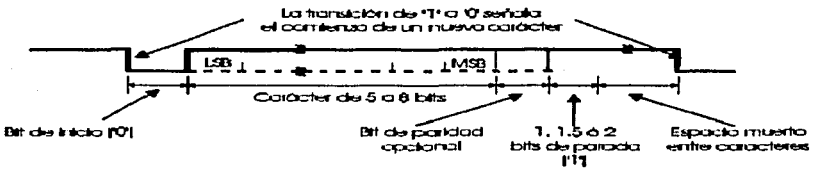


Figura 3.13 Formato de comunicación serie

El formato síncrono, en lugar de añadir bits a cada carácter, lo hace a un grupo de caracteres añadiendo una serie de caracteres adicionales al bloque de datos. Este conjunto de caracteres se conocen generalmente como caracteres de sincronismo, y son utilizados por el receptor para determinar los límites de los datos en una serie de bits. Cuando el transmisor no tiene datos que transmitir, inserta caracteres de reposo. En modo síncrono es muy común transmitir la señal de reloj con el fin de conseguir la máxima sincronización entre transmisor y receptor. Si este es el caso, el transmisor es el que envía la señal de reloj por una línea dedicada.

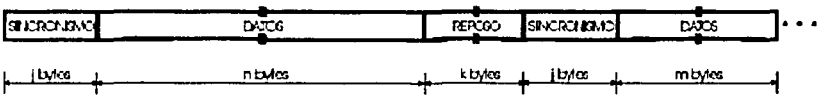


Figura 3.14 Formato de comunicación serie

En las figuras 3.15 y 3.16 se muestran los formatos correspondientes a ejemplos de transmisión asíncrona y síncrona respectivamente.

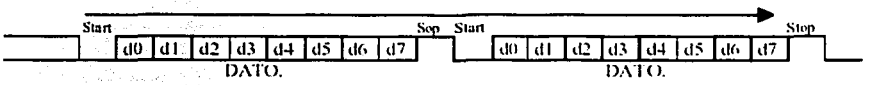


Figura 3.15 Formato de comunicación síncrona

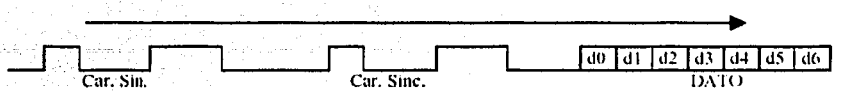


Figura 3.16 Formato de comunicación asíncrono

El formato síncrono de la figura 3.16 representa un típico ejemplo en el que son necesarios dos caracteres de sincronismo al comienzo del mensaje. El formato asíncrono de la figura 3.14 también representa un ejemplo típico en el que es necesario un bit de start seguido del correspondiente carácter de dato y finalmente un bit de stop para cada carácter que se transmite, que en este caso es de 8 bits. En este ejemplo, no se ha incluido el bit de paridad, que estaría, cuando sea empleado, entre el último bit de dato y el primer bit de stop.

En el caso del ejemplo, el modo asíncrono necesita 8+2+8+2=20 bits para transmitir solamente dos caracteres de información, frente a los 16+8+8=32 bits que se necesitan para transmitir también dos caracteres de información en modo síncrono. Desde este punto de vista, la transmisión asíncrona sería más ventajosa. Sin embargo, si se desea transmitir 1.000 caracteres de información, el modo asíncrono necesitaría (8+2) * 1000 = 10.000 bits totales, frente a los 8*1000 + 8 + 8 = 8.016 bits que serían necesarios en modo síncrono para transmitir los mismos caracteres de información que en el caso anterior. Con lo que se puede apreciar como la transmisión síncrona es más favorable en el caso de un volumen de información grande.

3.4.2 Circuito multiplexor

En la comunicación serie el protocolo que generalmente se suele utilizar es el EIA-RS-232-C, en donde se pueden utilizar voltajes bipolares de 9, 12, 15 [v], es por ello que se utiliza el circuito integrado max232, con el

propósito de que la señal proveniente del los diferentes puertos serie, se convierta a niveles de voltaje TTL, y una así con un multiplexor común, poder determinar el origen y destino de los datos, para posteriormente proceder a de nuevo convertir los voltajes TTL a voltajes RS-232.

En la figura 3.17 se presenta el circuito lógico del multiplexor y en la figura 3.18 el circuito eléctrico. Como se puede ver primero la señal RS-232 es convertido a TTL, pasa por un multiplexor (en ése caso se utilizan compuertas tres estados para poder realizar el multiplexor) y posteriormente se vuelve a pasar los datos al formato RS-232. En la figura 3.18 se presenta el circuito impreso. En el apéndice B se presenta en circuito impreso en sus dimisiones reales.

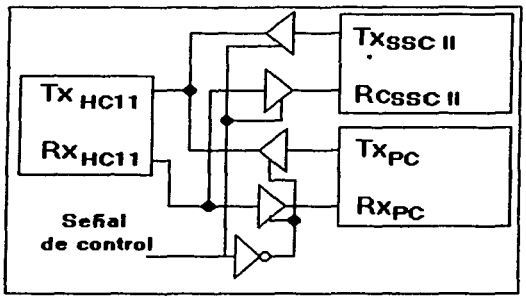


Figura 3.17 Circuito lógico del multiplexor

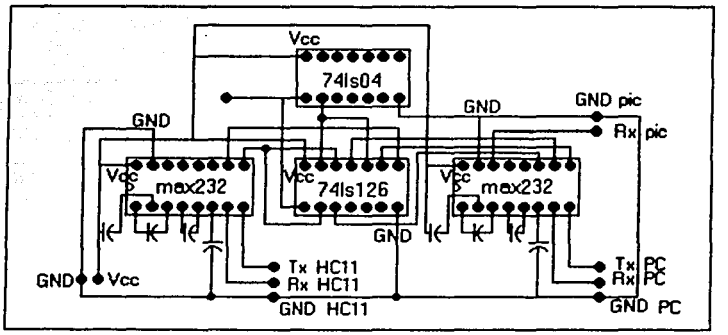


Figura 3.18 Circuito eléctrico del multiplexor

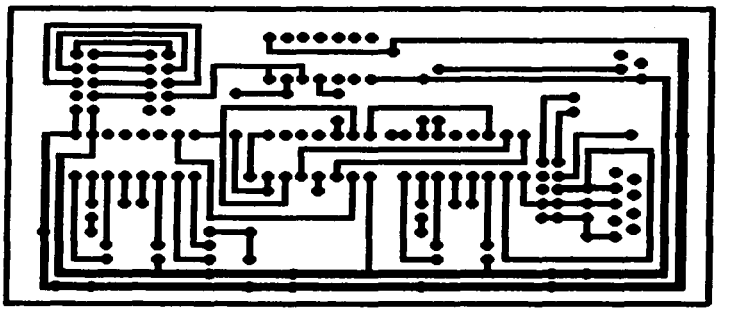


Figura 3.19 Circuito impreso del multiplexor

CAPÍTULO 4

ALGORITMOS DE MOVIMIENTO

Las máquinas caminantes ofrecen varias ventajas sobre los robots móviles, mientras que las ruedas deben concretarse a superficies planas, los robots con patas pueden trasladarse sobre superficies irregulares. Son capaces de maniobrar dentro de espacios confinados así como de transportar diferentes cargas. Es por ello que en la actualidad la investigación en el área de robots caminantes, una de sus tareas es la concerniente a la generación de los algoritmos que permitan realizar a los robots sus desplazamientos con un mínimo de inestabilidad.

En este capítulo se exponen en primer lugar las instrucciones del lenguaje Robel y de la sintaxis de las mismas, posteriormente los algoritmos desarrollados en el proyecto, así como también algunas ideas básicas en la generación de secuencias de desplazamientos de las patas.

4.1 Lenguaje Robel: instrucciones y sintaxis

A continuación se presenta un a descripción de las instrucciones con que cuentan el lenguaje Robel así como la sintaxis que estas tienen.

- Comando: left
  Sintaxis: left
  Ejemplo: left
  Descripción: El robot gira 10° a la izquierda.

- Comando: right
  Sintaxis: right
  Ejemplo: right
  Descripción: El robot gira 10° a la derecha.

- Comando: forward
  Sintaxis: forward
  Ejemplo: forward
  Descripción: El robot avanza 10 decímetros.

- Comando: backward
  Sintaxis: backward
  Ejemplo: backward
  Descripción: El robot retrocede 10 decímetros.

- Comando: mv
  Sintaxis: mv distance angle [speed]
  Ejemplo: mv 10 3.4
  Descripción: El robot primero gira y entonces avanza,
  La velocidad no es necesaria.

- Comando: mvto
  Sintaxis: mvto posX posY [speed]
  Ejemplo: mvto -5.3 2.7
  Descripción: El robot se dirige al punto (posX, posY),
  La velocidad no es necesaria.

- Comando: loads
  Sintaxis: loads script_name
  Ejemplo: loads test.txt
  Descripción: La instrucción carga un script.

- Comando: script
  Sintaxis: script script_name
  Ejemplo: script follow_line
  Descripción: Ejecuta un script que fue previamente cargado.

- Comando: ic
  Sintaxis: ic posX posY angle
  Ejemplo: ic -5 5 3.1
  Descripción: Introduce la posición y orientación del robot.

- Comando: sc
  Sintaxis: sc
  Ejemplos: sc
  Descripción: Muestra la posición y orientación del robot.

- Comando: shs
  Sintaxis: shs sensor_name id
  Ejemplo: shs reflective 1
  Descripción: Lee un sensores usando su nombre identificador.

- Comando: user
  Sintaxis: user número_de_función id
  Ejemplo: user 1
  Descripción: Ejecuta alguna de las funciones de usuario ya definidas.

4.2 Secuencias de movimiento

A continuación se presenta la explicación de los algoritmos diseñados durante el proyecto.

Con fines de facilitar el entendimiento y la explicación de los algoritmos es que se tomara la siguiente notación (cabe hacer mención de que no existe una notación estándar y la que aquí se plantea es propuesta por D. McCloy), en primer lugar se enumeran todas las patas de manera

secuencial, como 1, 2,..., n donde n es el número de patas del robot. Podemos describir las secuencias de dos formas:

- (a, b), Con esta notación se indica que la pata "a" se moverá en primer lugar y a continuación la pata "b".
- (a - b), en éste caso el guión indica el movimiento simultaneo de las patas "a" y "b".

A continuación se presenta un ejemplo de la notación, utilizando las secuencias para un robot trípedo.

- (1, 2, 3). Aquí se indica que la pata 1 se moverá en primer lugar a continuación la pata 2 y por ultimo la tercera.
- (1, 2 – 3) en éste caso la pata numero 1 se desplazará en primer lugar y a continuación las patas 2 y 3 lo harán de forma simultanea.

Para los cuadrúpedos existen 26 diferentes secuencias posibles para el funcionamiento de las patas. La primera (1-2-3-4) es el movimiento de las 4 patas de forma simultánea, las restantes 25 secuencias se enumeran en la tabla 4.1.

Funcionamiento como bípedo	Funcionamiento Como trípedo		Funcionamiento Como cuadrúpedo
1, 2-3-4	1, 2, 3-4	1, 3-4, 2	1,2,3,4
2, 1-3-4	1, 3, 2-4	1, 2-4, 3	1,2,4,3
3, 2-1-4	1, 4, 2-3	1, 2-3, 4	1,3,2,4
4, 1-2-3	2, 3, 1-4	2, 1-4, 3	1,3,4,2
1-2, 3-4	2, 4, 1-3	2, 4-3, 4	1,4,2,3
1-3, 2-4	3, 4, 1-2	3, 1-2, 4	1,4,3,2
1-4, 2-3			

Tabla 4.1

4.3 Estabilidad

En el caso de robots de 4 patas existen 6 secuencias potenciales y estáticamente estables. Un paso potencialmente estable, es aquél en el que el centro de gravedad se encuentra siempre contenido dentro del área encerrada por las patas de soporte. Para lograr una estabilidad estática es necesario que por lo menos estén tres patas siempre en contacto con el piso. A su vez se requiere que una máquina andante estáticamente estable tenga al menos cuatro patas, es decir, tres patas que le sirvan de apoyo mientras la cuarta se levanta.

4.4 Marcha

La marcha se puede definir como; la trayectoria característica de las pisadas, aunque también debe de considerarse la duración de cada fase de la secuencia. Tanto la secuencia como su duración pueden considerarse si la marcha se define en forma matricial. Si el estado elevado de una pata se designa por un 1 y el estado de asentamiento por un 0, entonces es posible construir una matriz en que se muestre el estado de todas las patas en todas las fases a través de todo el ciclo de una marcha.

Para el caso de los cuadrúpedos, como se vio anteriormente existen 26 secuencias diferentes para el movimiento de las patas y, de ellas 20 implican el movimiento simultaneo de 2 o más patas, por lo tanto, las seis secuencias restantes forman la base de las marchas cuadrúpedas no singulares. Existen 7!=5040 marchas cuadrúpedas no singulares cada una de las cuales involucran ocho fases; cuatro levantamientos y cuatro asentamientos.

EL gran número de marchas no singulares se reduce drásticamente si se exige, con el fin de mantener la estabilidad estática, que tres patas estén

en el piso en cualquier momento dado. Esta limitación reduce el número de marchas cuadrúpedas a seis (conocidas como marchas reptantes) con las secuencias:

1. 1,2,4,3
2. 1,3,4,2
3. 1,4,2,3
4. 1,2,3,4
5. 1,3,2,4
6. 1,4,3,2

Por ejemplo, la matriz de marcha para el andar de un cuadrúpedo es:

0 0 0 0
1 0 0 0
0 0 0 0
0 0 1 0
0 0 0 0
0 1 0 0
0 0 0 0
0 0 0 1

Lo anterior demuestra que el andar del cuadrúpedo alterna entre posturas en tres y cuatro patas.

4.5 Algoritmos desarrollados

A continuación se presentarán los diferentes algoritmos diseñados para operar los movimientos del robot, los algoritmos que se muestran son para desplazarse hacia atrás, o adelante, a la izquierda o derecha, y girar en sentido positivo o negativo. Aunque como se verá más adelante los algoritmos para el desplazamiento a la izquierda y a la derecha no son

utilizados para el lenguaje Robel, aunque pueden ser utilizados por medio de funciones de usuario por medio de la instrucción user.

4.5.1 Movimientos básicos

A continuación se presenta una explicación de los movimientos y posiciones que pueden tener cada una de las patas del robot de forma que más adelante se entiendan de forma más clara los algoritmos que se diseñaron.

Los movimientos y posiciones verticales. Las posiciones que se tienen son tres y son denominadas como extensión, flexión e inicial y los movimientos son solo subir y bajar la pata. La Posición vertical inicial es en la que el robot se encuentra al iniciar cada una de las secuencias de movimiento y se pude apreciar en la figura 4.1.

La posición de extensión (figura 4.2) se da antes de que alguna de las patas gire para desplazar el robot, y es con el propósito de que al girar la pata el efecto del movimiento sea aprovechado de una mejor forma evitando en lo posible deslizamientos al tener que soportar el efecto del peso.

La posición de flexión (figura 4.3) es con el propósito de que al girar la pata, el movimiento no afecte la ubicación ni orientación del robot.

En lo que se refiere a los movimientos horizontales son únicamente dos, giro en sentido horario o positivo (en la figura 4.4 se puede observar que la pata 2 deberá girar en sentido positivo) y antihorario o negativo (en la figura 4.5 se indica que deberá gira la pata 2 en sentido horario).

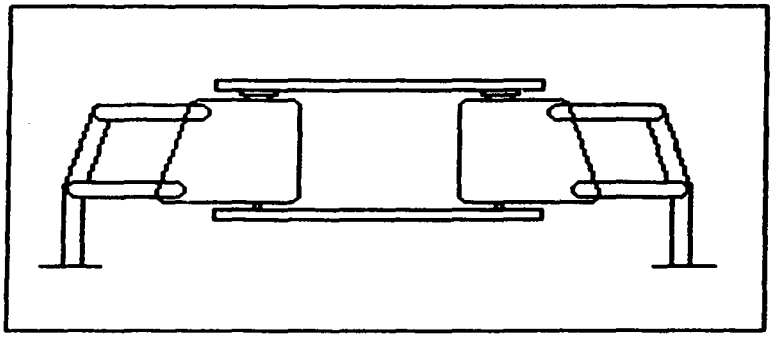


Figura 4.1 Posición inicial (vertical) de cada una de las patas antes de cada secuencia de movimiento.

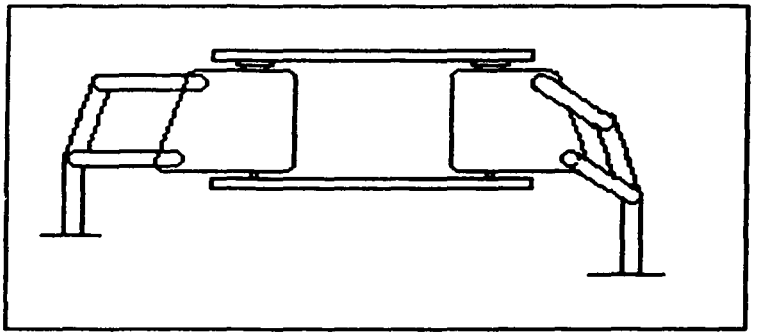


Figura 4.2 La pata de la derecha se encuentra en extensión, mientras que la de la izquierda en su posición inicial.

Las posiciones horizontales en las que se encuentra el robot solo son dos inicial o de equilibrio y girada. En la figura 4.4 se presentan las posiciones iniciales del robot antes de iniciar cada una de las secuencias de movimiento, mientras que en la figura 4.5 se presenta a la pata 2 en una posición de rotación y a partir de estas se indica el movimiento que realizara cada una de las patas.

4.5.2 Notación grafica

En cuanto a las secuencias de movimiento, se explicarán de dos formas, una de acuerdo a la notación mostrada en las secciones anteriores y otra gráfica. La primera es con el propósito de mostrar la forma en la que se desarrollará la secuencia, y la segunda para indicar los movimientos de las patas y la estabilidad de las mismas.

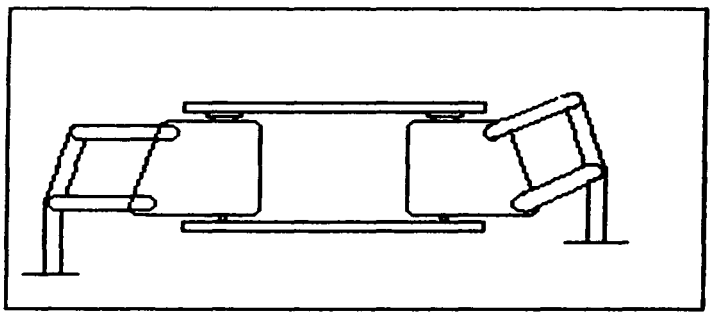


Figura 4.3 La pata de la derecha se encuentra en flexión, mientras que la de la izquierda en su posición inicial.

La notación gráfica que se propone, en primer lugar en la figura 4.6 se muestra la numeración que tiene cada una de las patas. El círculo dentro del cuerpo del robot se utiliza para señalar el frente y así definir las

direcciones atrás, adelante, izquierda y derecha. El punto que se muestra dentro del cuerpo del robot indica el centro de gravedad.

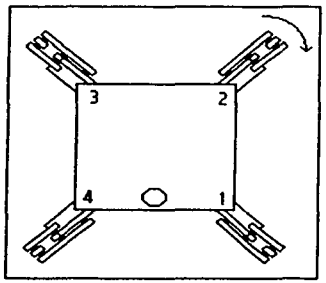


Figura 4.4 Indicación de giro en sentido horario a partir de su posición inicial.

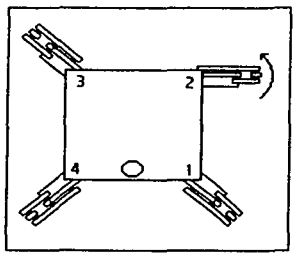


Figura 4.5 Indicación de giro en sentido antihorario a partir de que la pata esta fuera de su posición inicial.

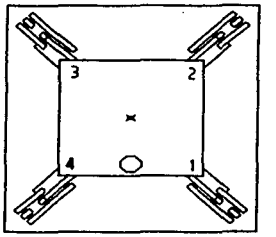


Figura 4.6 Numeración de las patas del robot.

En la representación gráfica, se utilizarán los movimientos antes mencionados y cómo se combinan para llevar a cabo los diferentes desplazamientos. Por otra parte junto a la figura del robot se presentará una flecha junto a una pata para indicar la que realizará un movimiento y cuál es el que se llevará a cabo, como se puede apreciar en la figura 4.5 donde se puede ver que la pata 2 deberá girar en sentido antihorario. También, si el extremo exterior de la pata está en blanco indicará que la pata está haciendo contacto con el piso, mientras que si está en negro, implicará que la pata no hace contacto con el piso. En la figura 4.7 se muestra un ejemplo en el que las patas 2, 3 y 4 se encuentran haciendo contacto con el piso, mientras que la pata 1 se encuentra elevada. Lo anterior para ser coherentes con la notación matricial, donde un 0 indica que la pata está haciendo contacto con el suelo y un uno que está en elevación.

En la figura 4.8 se puede apreciar la forma en la cual se explicará el estado vertical de las patas, en la parte inferior de la figura aparecerán las patas uno y tres, en la parte superior las 2 y 3. Así como en las figuras 4.7 en la 4.8 se puede ver que la pata 1 se encuentra elevada.

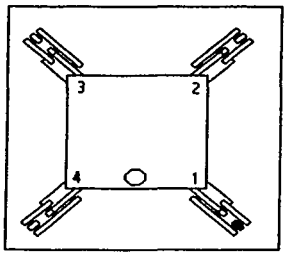


Figura 4.7 Indicación de que la pata 1
no hace contacto con el piso.

El acomodo de las columnas en la representación matricial es: la primera columna corresponderá a la situación de la pata 1 la segunda a la pata 2 y así respectivamente.

1 2 3 4
0 0 0 0

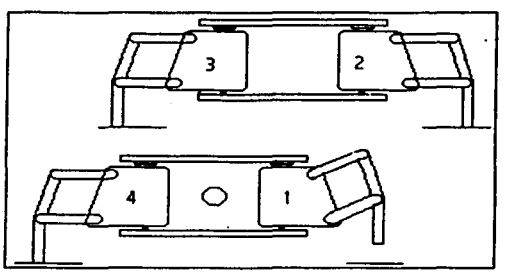


Figura 4.8 Vista vertical de las patas del robot.

En las siguientes subsecciones se presentan los algoritmos en forma tabular, en estas se presenta de forma grafica el movimiento y debajo de

la figura se describe el movimiento de la siguiente forma. Primero el movimiento (Giro [G], Extensión [E] o Flexión [F]), segundo, el sentido del giro horario (SH) o antihorario (SA) (únicamente en el caso de giros), Tercero la pata a mover (P1, P2, P3, P4). A continuación se presentan algunos ejemplos

1) G SA P1 girar en sentido positivo la pata 1
2) F P2 flexionar pata 1
3) E P4 extender para 4

4.5.3 Algoritmo básico

Lo seis algoritmos diseñados tiene una característica común, y es la de que se pueden realizar a partir de un algoritmo único y la diferencia radica en el orden en el cual se mueven cada una de las patas y en el caso de los algoritmos para desplazarse a la derecha e izquierda en donde cambia también el sentido de los giros.

Lo que realizan los seis algoritmos, es tomar dos de sus patas ejecutar el algoritmo básico y posteriormente tomar las patas restantes y ejecutar de nuevo el algoritmo.

El algoritmo se explica en dos formas, primero se platea el movimiento en seudocódigo y enseguida de manera grafica algunos de los pasos.

1.- Extender pata a

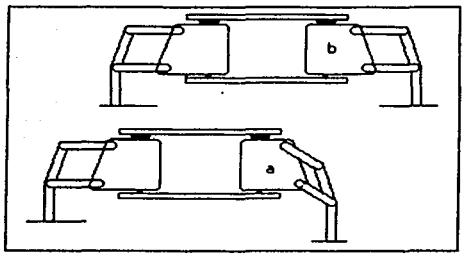


Figura 4.9 Paso 1.

2.- Girar pata a

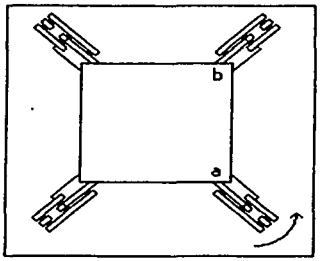


Figura 4.10 Paso 2

3.- Bajar pata a
4.- Levantar pata b

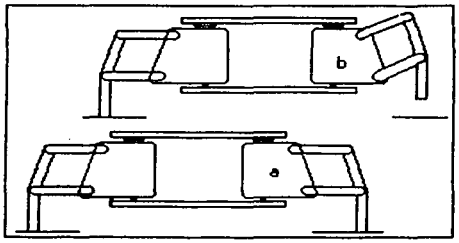


Figura 4.11 Paso 4.

5.- Girar pata b

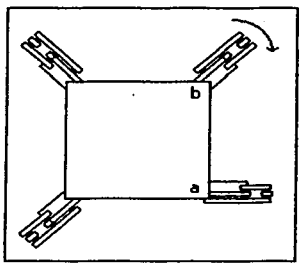


Figura 4.12 Paso 5.

6.- Bajar pata b
7.- Levantar pata a

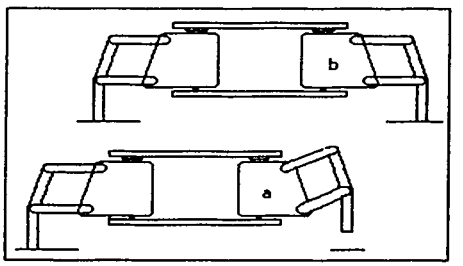


Figura 4.13 Paso 7.

8.- Girar pata a

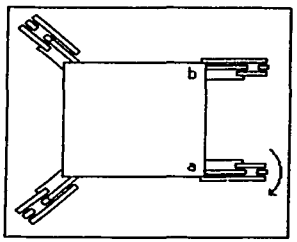


Figura 4.14 Paso 8

9.- Bajar pata a
10.- Extender pata b

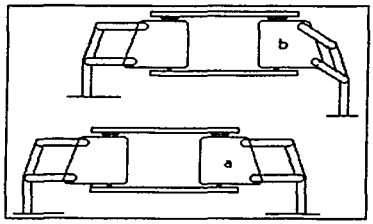


Figura 4.15 Paso 10

11.- Girar pata b

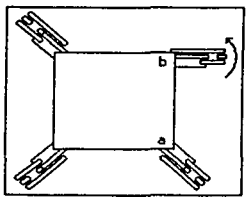


Figura 4.16 Paso 11

El estado después del algoritmo debe ser el que se muestra en la figura 4.14, debiendo terminar el robot en el estado de equilibrio o inicial.

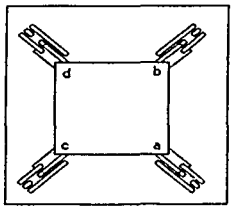


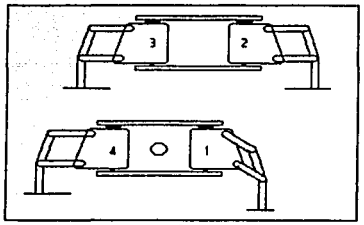
Figura 4.17 Estado al final del algoritmo

Para completar el ciclo las dos patas restantes "c" y "d" (ver figura 4.14) deben ejecutar el mismo algoritmo. En donde los movimientos equivalentes a la pata "a" son los que realizará la pata "c" y los que se mostraron para la pata "b" los realizará la pata "d"

Para los diferentes movimientos lo que va a cambiar es el orden en que se moverán las pata como ya se dijo, como a continuación se mostrará.

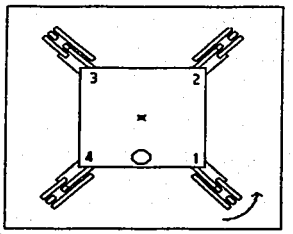
4.5.4 Desplazamiento hacia delante

La secuencia que se utiliza para poder realizar el desplazamiento hacia delante es moviendo las patas en el siguiente orden (1, 2, 4, 3), lo cual se verá en la figura 4.18.



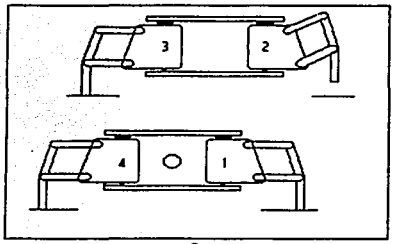
1

E - P3



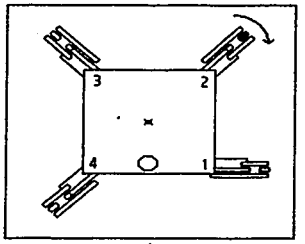
2

G - SA - P1



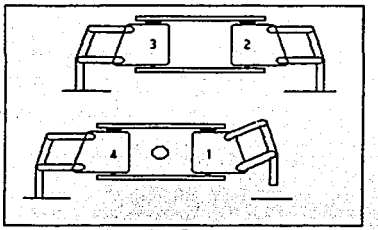
3

F - P1 Y F - P2



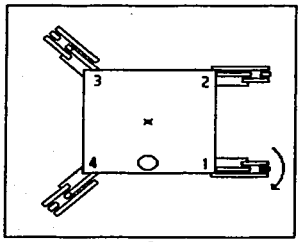
4

G - SH - P2



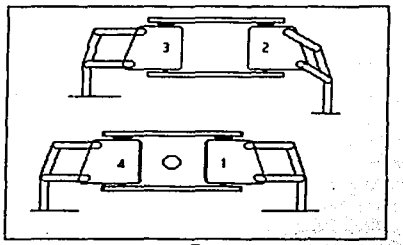
5

E - P2 Y F - P1



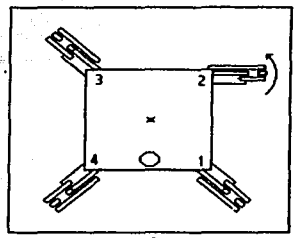
6

G SA P1



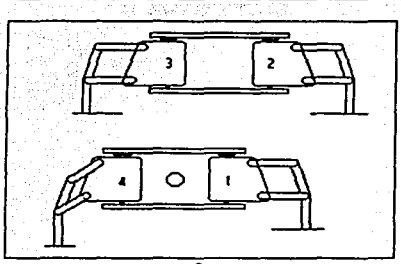
7

E - P1Y E – P2



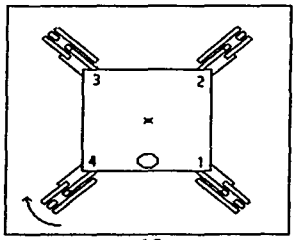
8

G SH P2 Y F - P2



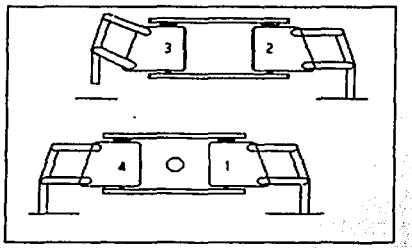
9

E – P4



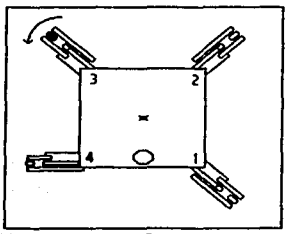
10

G - SH - P4



11

F – P4 Y F - P3



12

G – SA – P3

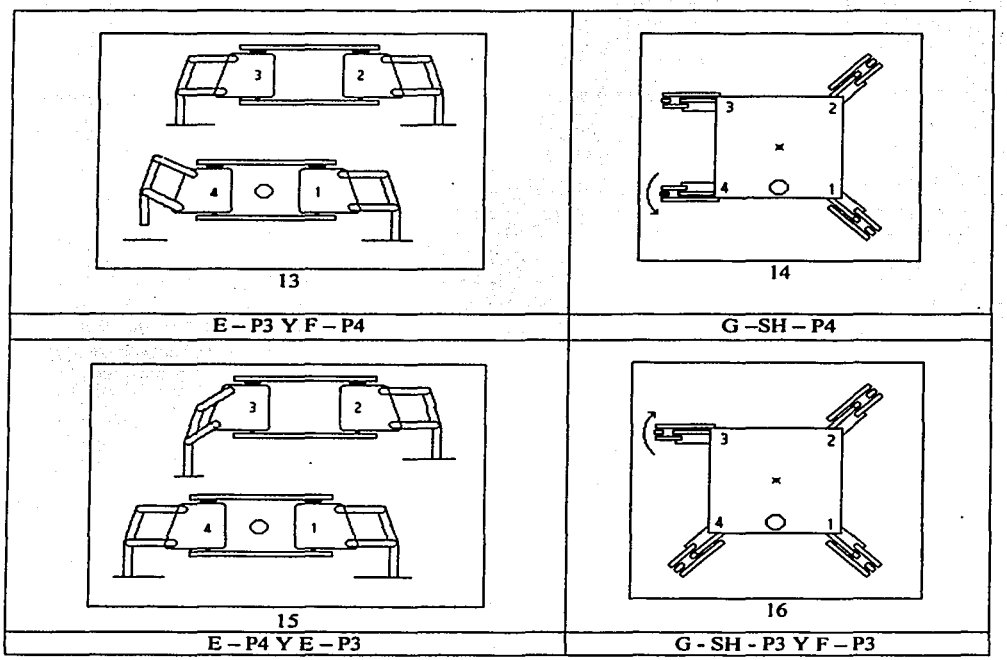


Figura 4.18 Algoritmo para el desplazamiento hacia delante

La representación matricial de la secuencia es:

0	0	0	0
0	1	0	0
0	0	0	0
1	0	0	0
0	0	0	0
0	0	0	1
0	0	0	0
0	0	1	0

En lo que se refiere a la estabilidad, en los casos en los que alguna de las patas está en elevación, el centro de gravedad está dentro del área formada por las patas que se encuentran en el piso (figura 4.19).

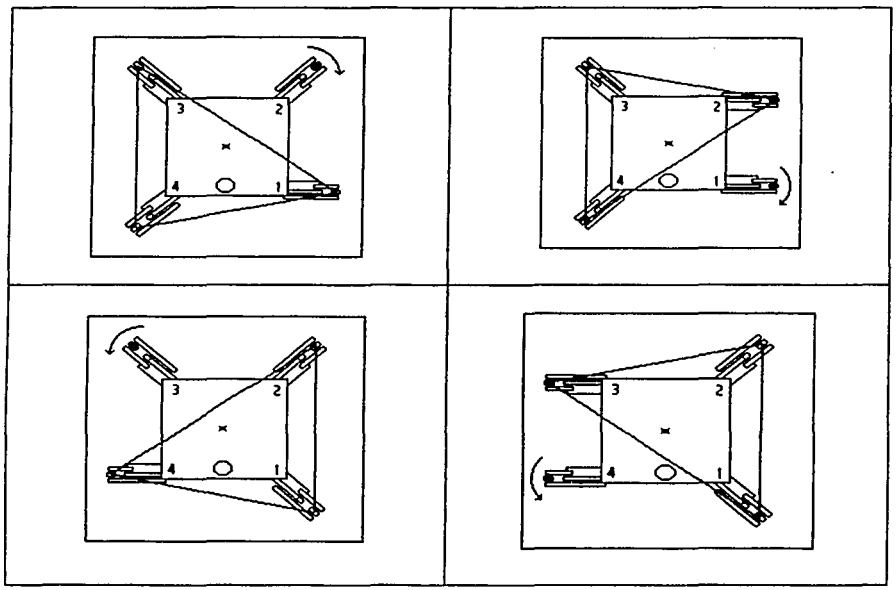


Figura 4.19 Ubicación del centro de gravedad en los casos en los que solo tres patas hacen contacto con el piso

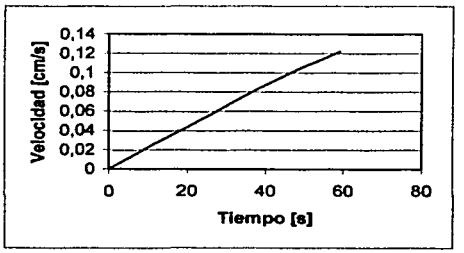
El algoritmo para el desplazamiento hacia delante se puede utilizar por medio de las instrucciones mv y forward, como se mostrará en dos ejemplos, en el ejemplo 1 el robot se desplazará 30 cm hacia delante, mientras en el segundo avanzará hacia delante 10 cm.

1. mv 0.3,0.0
2. forward

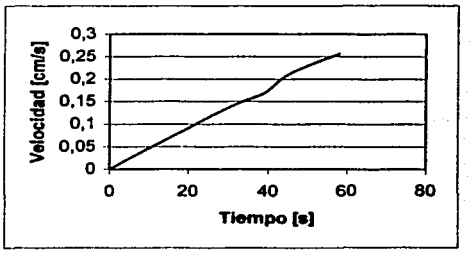
Cinemática: Un aspecto importante a considerar es la velocidad a que el robot se desplazará al aplicar el algoritmo, para poder determinar la función que gobierna este punto se realizaron pruebas experimentales en donde se media la distancia recorrida en un cierto tiempo, de tal forma que se obtuvieran las curvas representativas de las velocidad del robot. Obteniéndose en general comportamientos lineales como se muestra en las gráficas 1, 2, 3 y 4. A partir de estos resultados se realizó la obtención de la ecuación de la velocidad.

4.5.5 Desplazamiento hacia atrás

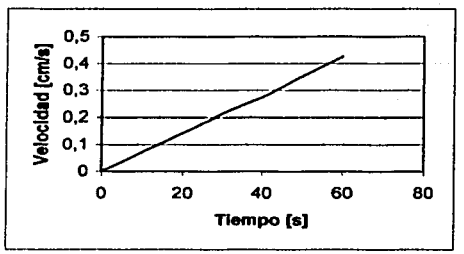
En el caso del desplazamiento hacia atrás, el algoritmo lleva a cabo la siguiente secuencia (2, 1, 4, 3).



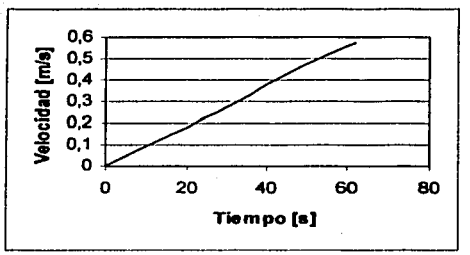
Gráfica 1. Velocidad del robot con desplazamientos angulares de 7.05°



Gráfica 2. Velocidad del robot con desplazamientos angulares de 14.1°



Gráfica 3. Velocidad del motor con desplazamientos angulares de 21.15°

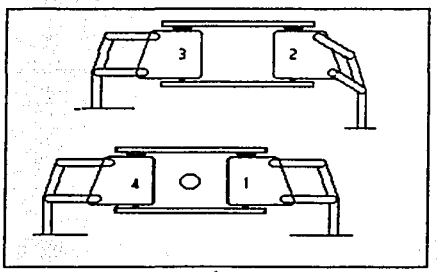


Gráfica 4. Velocidad del robot con desplazamiento angulares de 28.2°

La representación matricial de la marcha tendrá la siguiente forma.

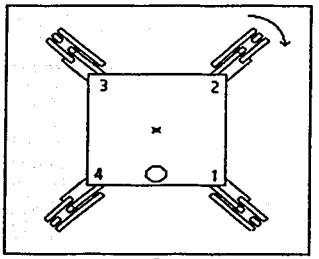
0	0	0	0
0	1	0	0
0	0	0	0
1	0	0	0
0	0	0	0
0	0	1	0
0	0	0	0
0	0	0	1

La representación gráfica para el algoritmo hacia atrás es la que se presenta en la figura 4.20.



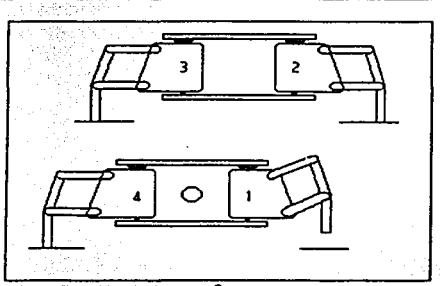
1

E- P2



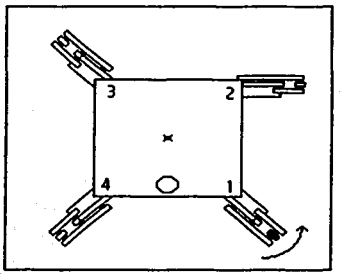
2

G – SH – P2



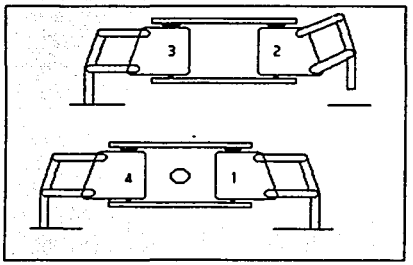
3

F – P2 Y F – P1



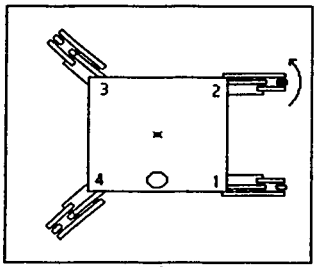
4

G – SA - P1



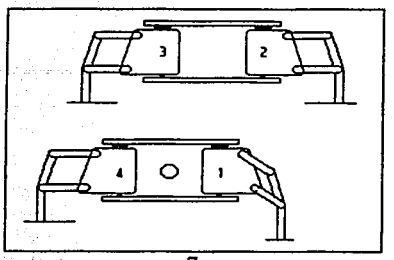
5

E – P1 Y F - P2



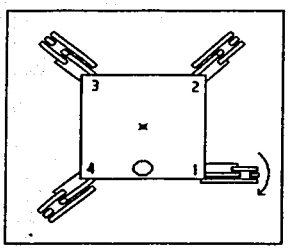
6

G – SA - P2



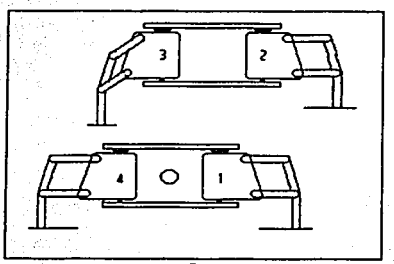
7

E – P2 Y E - P1



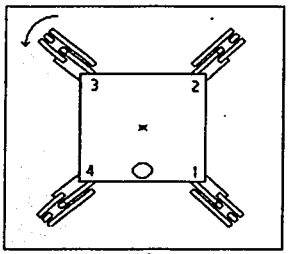
8

G – SH - P1 Y F - P1



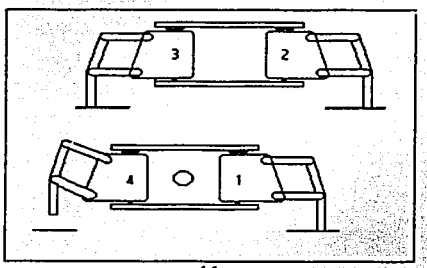
9

E – P3



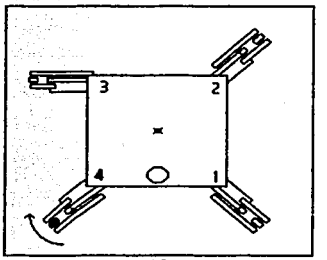
10

G - SA – P3



11

F - P3 Y F - P4



12

G – SH – P4

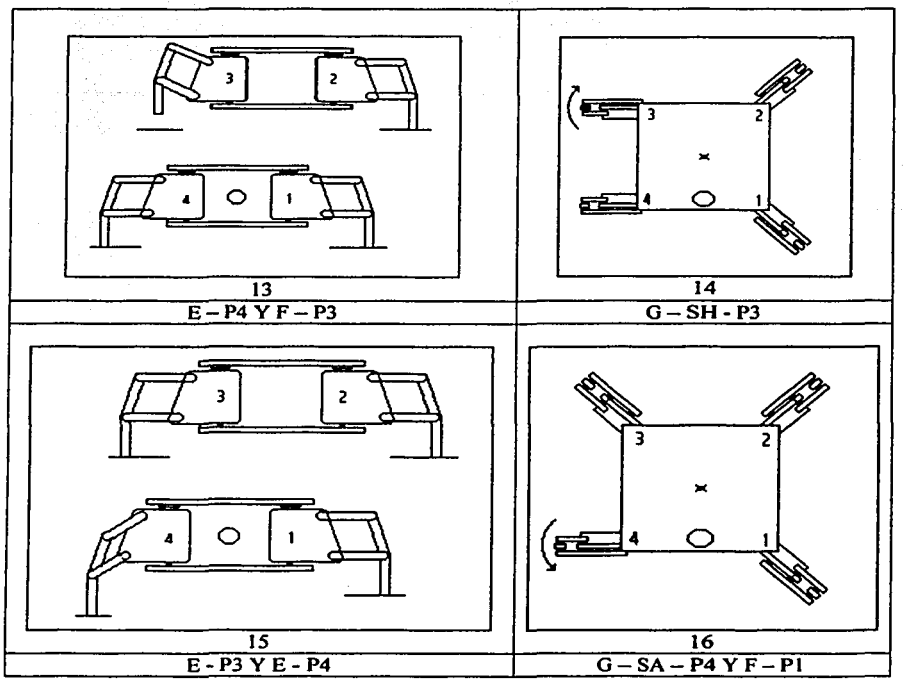


Figura 4.20 Algoritmo de desplazamiento hacia atrás

Para la estabilidad se presenta la misma situación del desplazamiento hacia delante, en donde solo 4 estados alguna de las patas no hace contacto con el piso, en los cuatro casos el centro de gravedad se encuentra dentro del triángulo formado por las patas que sí hacen contacto con piso. Esto se pude ver en la tabla 4.21.

Cinemática: Al igual que en el caso del desplazamiento hacia delante, se realizaron las pruebas de distancia contra tiempo con el propósito de

obtener las curvas de velocidad del robot para el algoritmo del movimiento hacia atrás. En este caso también se obtuvo un comportamiento lineal como se podrá ver en las gráficas 5, 6, 7 y 8, dicha situación facilitará los cálculos al momento de que se desee que el robot se desplace a una cierta velocidad y una determinada distancia, así como el análisis de los desplazamientos realizados.

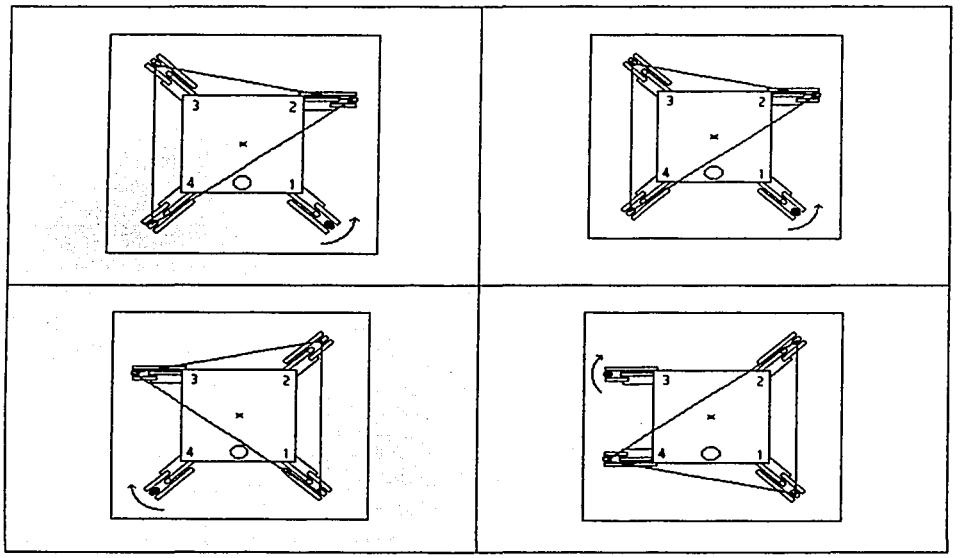
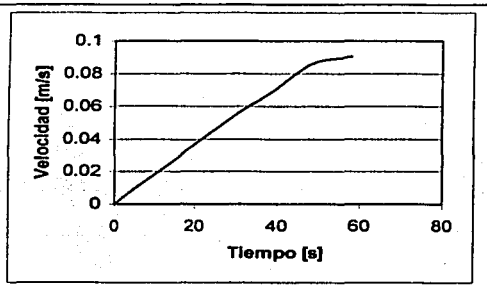


Figura 4.21 Ubicación del centro de gravedad en los casos en los que solo tres patas hacen contacto con el piso

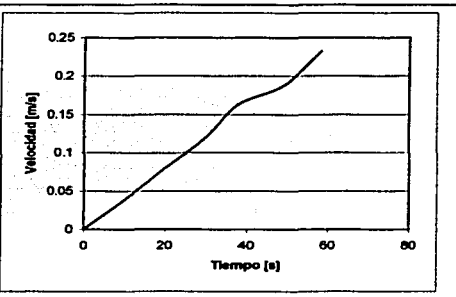
El algoritmo es utilizado cuando la distancia en la instrucción mv es negativa, o bien con la instrucción back.

4.5.6 Giro en sentido positivo

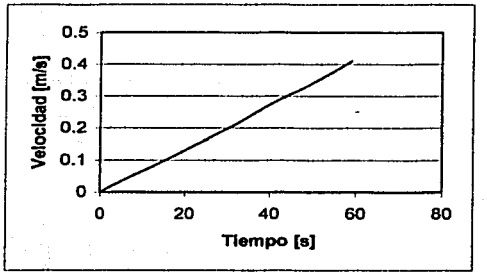
Para llevar a cabo el giro del robot en sentido positivo u horario, la secuencia que se sigue una combinación de las secuencias para desplazarse hacia delante y atrás, más explícitamente, los 8 primeros esquemas del giro positivo son los que se pueden observar en las figuras 18-1 a 18-8 (correspondientes al desplazamiento hacia delante), que son los que corresponden al movimiento de las patas 1 y 2, mientras que para los movimientos de las patas 3 y 4 son los que corresponden a las figuras 20-9 a 20-16 (que corresponden al movimiento hacia atrás), de tal forma que se completan los 16 esquemas del algoritmo.



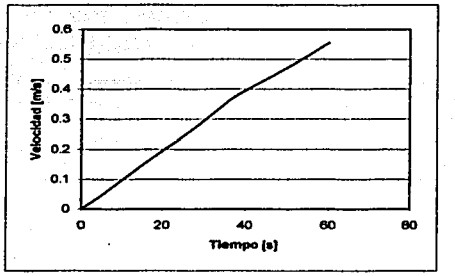
Gráfica 5. Velocidad del robot con desplazamientos angulares de 7.05°



Gráfica 6. Velocidad del robot con desplazamientos angulares de 14.1°



Gráfica 7. Velocidad del robot con desplazamientos angulares de 21.15°



Gráfica 8. Velocidad del robot con desplazamientos angulares de 28.2°

La representación matricial del algoritmo es la siguiente:

0	0	0	0
0	1	0	0
0	0	0	0
1	0	0	0
0	0	0	0
0	0	1	0
0	0	0	0
0	0	0	1

De esta forma la secuencia que se tiene es (1, 2, 3, 4).

4.5.7 Giro en sentido negativo

Al igual que el giro positivo, el que se da en sentido negativo también es una combinación las secuencias de movimiento hacia delante y atrás, en este caso los 8 primeros estados y movimientos del algoritmo son los que se pueden apreciar en las figuras 20-1 a 20-8 (del desplazamiento hacia

atrás) y los restantes 8 estados son los que se presentan en las figuras 19-9 a 19-16 (correspondientes al movimiento hacia delante).

En este caso como se pude ver la secuencia es (4, 3, 2, 1), y su matriz de marcha es:

0	0	0	0
0	0	1	0
0	0	0	0
0	0	0	1
0	0	0	0
1	0	0	0
0	0	0	0
0	1	0	0

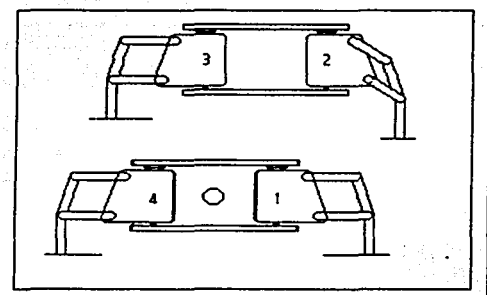
4.5.8 Desplazamiento hacia la derecha

Para lograr llevar a cabo los movimientos tanto hacia la derecha como a la izquierda existen dos posibilidades, una es girar el robot en sentido positivo o negativo 90° grados y avance, o bien, aprovechar las ventajas que este tipo de robot proporciona.

La secuencia que se utilizará para este movimiento es (2, 3, 1, 4), lo anterior se pude apreciar en las figuras 4.22, donde se apreciará el algoritmo. La forma matricial del algoritmo es:

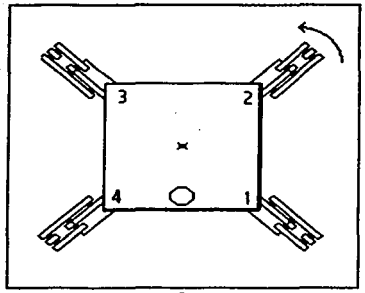
0	0	0	0
0	0	1	0
0	0	0	0
0	1	0	0
0	0	0	0

0	0	0	1
0	0	0	0
1	0	0	0



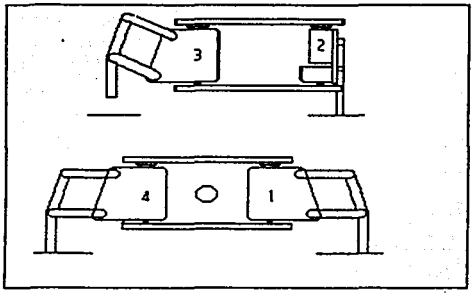
1

E – P2



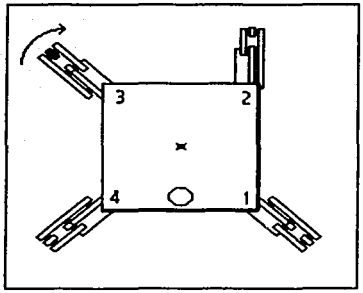
2

G - SA – P2



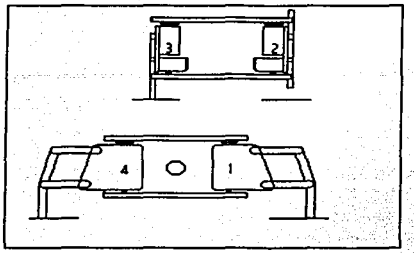
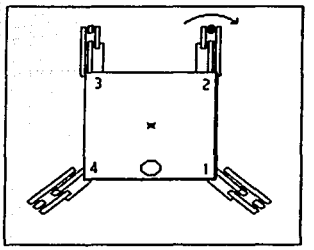
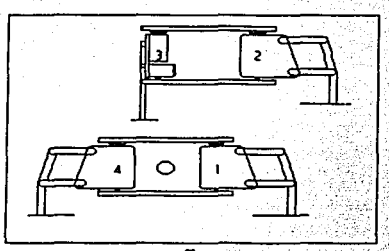
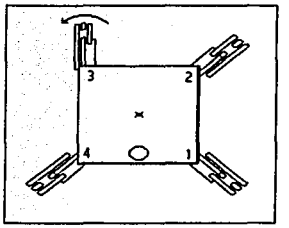
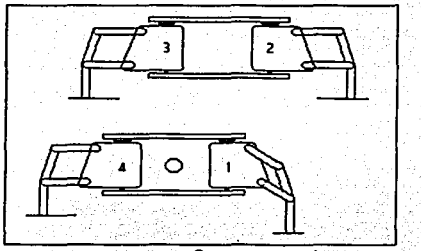
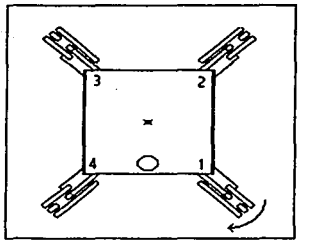
3

F – P2 Y F – P3



4

G - SH – P3

 5	 6
E – P3 Y F – P2	G – SH – P2
 7	 8
E – P3 Y E - P2	G - SA – P3
 9	 10
E - P1	G – SH – P1

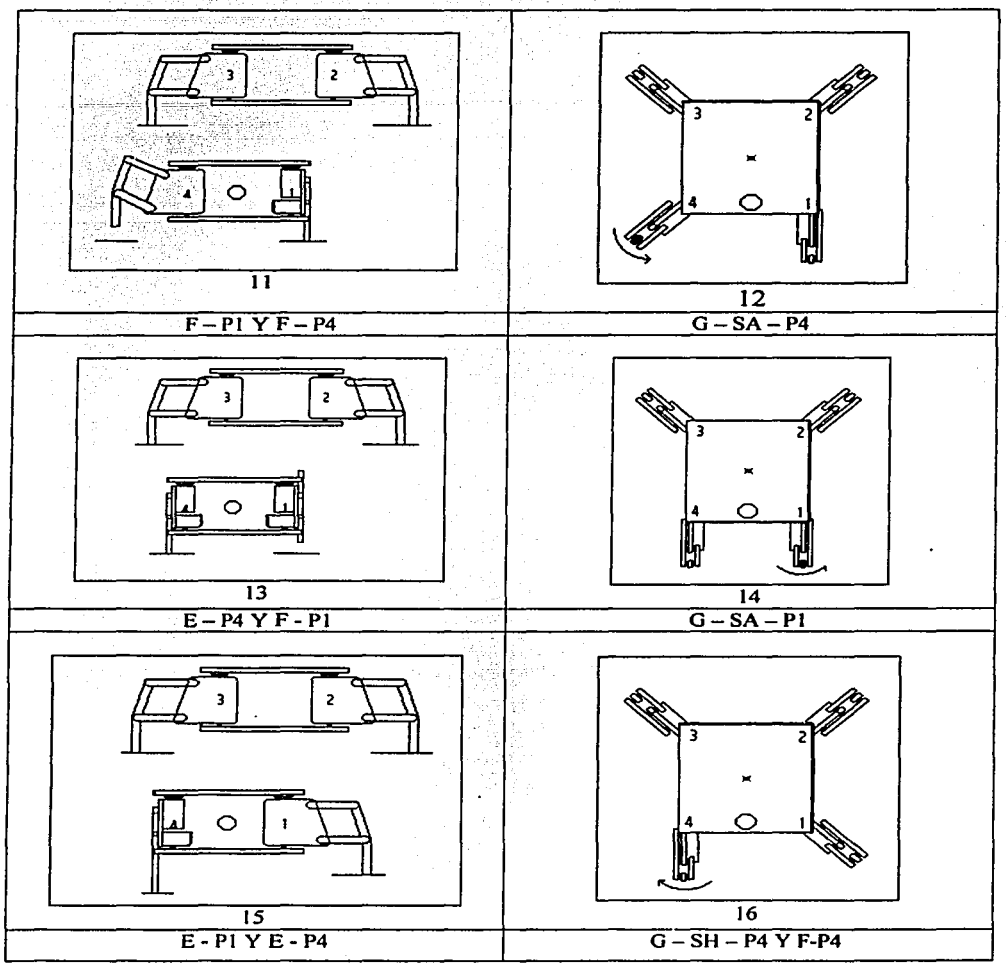
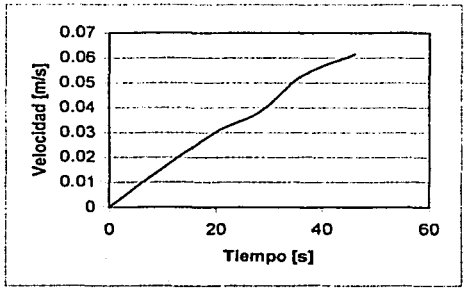


Figura 4.22 Algoritmo para los desplazamientos hacia la derecha

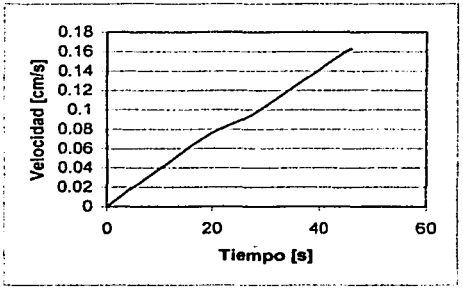
Cinemática: En las pruebas de velocidad se obtuvieron graficas muy similares a los casos del desplazamiento hacia delante y atrás solo con variaciones en la velocidad de los desplazamientos, como se verá en las gráficas 9, 10, 11 y 12, dichas situaciones pueden ser consideradas en un futuro, al diseñar un esquema de control en lazo cerrado ya que solo será necesario un solo controlador para los cuatro movimientos (desplazamientos a la izquierda, derecha atrás y adelante), y no un controlador para cada movimiento.

4.5.8 Desplazamiento hacia la izquierda

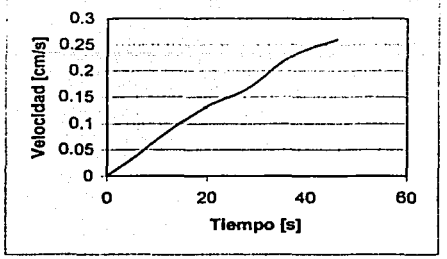
Este algoritmo es muy similar al que se presenta en el caso anterior, la diferencia consiste en el orden en el cual se desarrolla la secuencia, que es (3, 2, 4, 1), esta situación de igual forma se puede ver con la matriz de marcha.



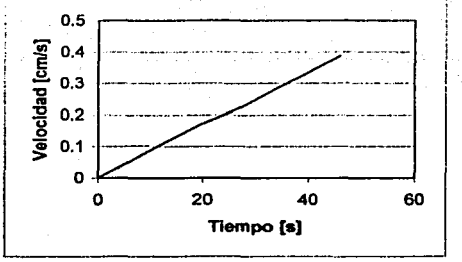
Gráfica 9. Velocidad del robot con desplazamientos angulares de 7.05°



Gráfica 10. Velocidad del robot con desplazamientos angulares de 14.1°



Gráfica 11. Velocidad del robot con desplazamientos angulares de 21.15°

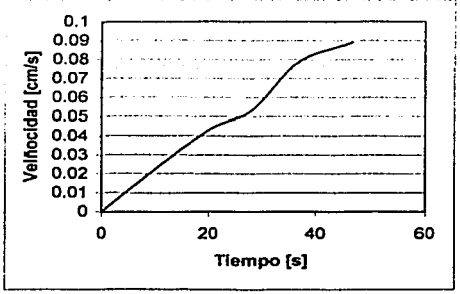


Gráfica 12. Velocidad del robot con desplazamientos angulares de 28.2°

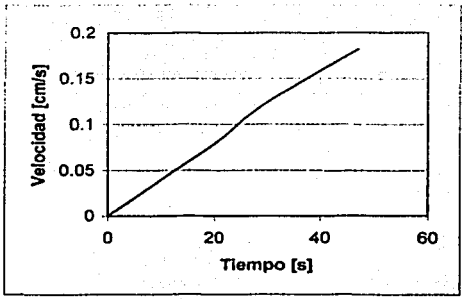
0	0	0	0
0	1	0	0
0	0	0	0
0	0	1	0
0	0	0	0
1	0	0	0
0	0	0	0
0	0	0	1

Como se puede apreciar, lo único que cambia es, cuál de las patas es la que se desplaza primero hacia fuera y cuál la que se mueve al centro y después hacia fuera.

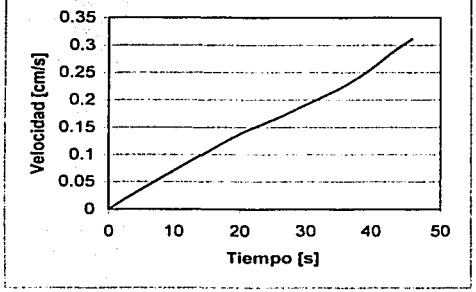
Cinemática: En las graficas anteriores (graficas 13, 14, 15 y 16) se podrá apreciar el comportamiento que el robot presenta al desplazarse a la izquierda. Cabría hacer mención que se pudo ver la cinemática del robot, la cual corresponde a un comportamiento prácticamente lineal (las variaciones se deben principalmente a deslizamientos que se presentaron, así como al juego que se presenta en las articulaciones), por



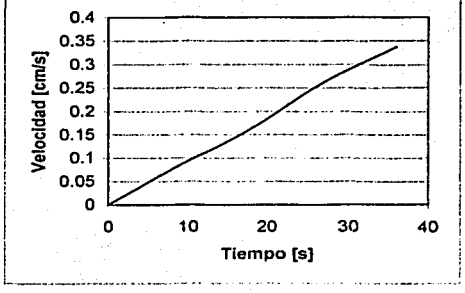
Gráfica 13. Velocidad del robot con desplazamientos angulares de 7.05°



Gráfica 14. Velocidad del robot con desplazamientos angulares de 14.1°



Gráfica 11. Velocidad del robot con desplazamientos angulares de 21.15°



Gráfica 11. Velocidad del robot con desplazamientos angulares de 28.2°

lo que a la hora de programar los movimientos tanto de velocidad como de distancia se consideraron movimientos lineales y para dichas rectas se hizo una aproximación por medio de regresión lineal.

CAPÍTULO 5

RESULTADOS EXPERIMENTALES

Una vez que se diseñaron los algoritmos y se obtuvieron las ecuaciones cinemáticas de movimiento, se realizaron diferentes pruebas para verificar tanto las ecuaciones y las secuencias de movimiento, pero ya con el lenguaje Robel. Los resultados de estas pruebas son presentados a continuación.

Las primeras pruebas que se realizaron, fueron combinaciones de movimientos de giro en sentido positivo y negativo más avanzar o retroceder una distancia dada. En el primer caso se hizo girar al robot un ángulo θ de 10, 20, 30 y 40 [°] una vez que giraba el robot se le hizo avanzar o retroceder 30 cm (ver figura 5.1), para ello se utilizó la instrucción mv en donde se podrá apreciar por un lado el error angular así como el error en la distancia recorrida (los resultado que se presentan son datos promedio). En las gráficas 5.1 y 5.2 se pueden ver los errores obtenidos al avanzar 30 cm mediante la instrucción:

mv 0.3, θ,25 ; Gira θ° y después avanza o retrocede 30 cm

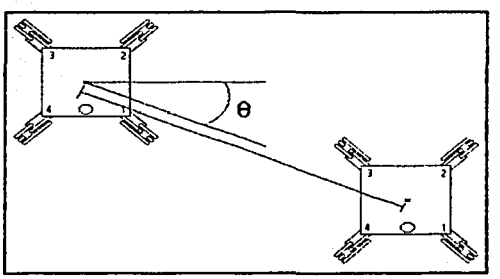
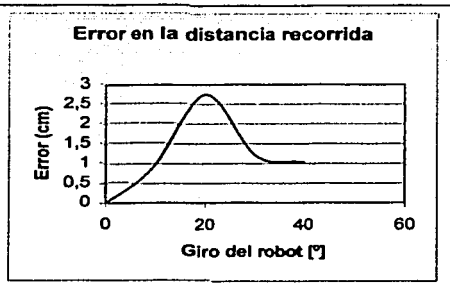
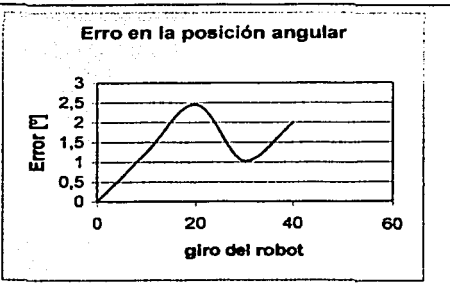


Figura 1 Experimento 1.

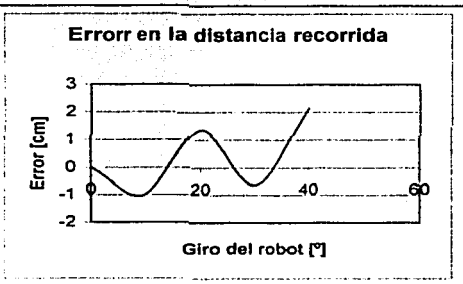


Gráfica 5.1 Error en la distancia recorrida con la instrucciónmv 0.3, θ,25

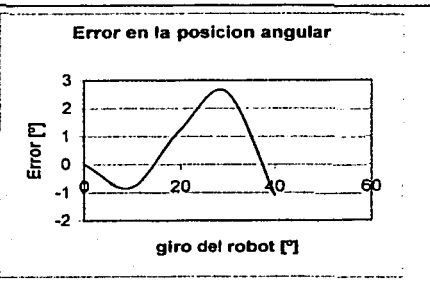


Gráfica 5.2 Error en el ángulo girado con la instrucción mv 0.3, θ,25

Las gráficas 5.3 y 5.4 son con la misma instrucción básicamente pero recorriendo una distancia de 20 cm
mv 0.2, θ,10 ; Gira θ° y después avanza o retrocede 20 cm



Gráfica 5.3 Error en la distancia recorrida con la instrucción mv 0.2, θ,10



Gráfica 5.4 Error en el ángulo girado con la instrucción mv 0.2, θ,10

El segundo experimento que se realizó, fue el hacer girar al robot 180 [°] que avanzara una distancia dada (20, 30 y 40 cm), y después volviera a girar 180 [°] y recorra la misma distancia (ver figura 5.2), de forma que regrese a su posición inicial. El experimento se realizó 10 veces con cada una de las distancias antes mencionadas. En las gráficas 5.4 y 5.5, se presentan los errores que se obtuvieron una vez que el robot había regresado a su posición inicial tanto en orientación como en distancia.

Para llevar a cabo el segundo experimento, se utilizó la siguiente instrucción en dos ocasiones.

mv d,3.14,t ;gira 180° y avanza una distancia d en un tiempo t=10d

El tercer experimento fue, el hacer que al desplazarse el robot hiciera un cuadrado. Este experimento se realizó en 20 ocasiones, recorriendo una distancia de 15 cm, 10 pruebas se hicieron girando en sentido positivo y 10 girando en sentido negativo, para ello se utilizó la instrucción:

mv 0.15,1.57,7 ;gira 90 grados y avanza 15 cm en 7 segundos

En el caso del giro en sentido positivo se presentó un error angular promedio de 3.75° y un error en la distancia recorrida de 1.36 cm, para el giro en sentido negativo los resultados fueron, en el error angular 2.16° y en la distancia recorrida 2.98 cm, los datos anteriores comparando la posición inicial y la final, aquí cabe hacer mención que también se presentó un segundo error esto fue un desplazamiento a la derecha de 5.63 cm en el caso en el cual el giro se dio en sentido negativo y un error de 7.81 cm a la izquierda en caso del giro en sentido positivo.

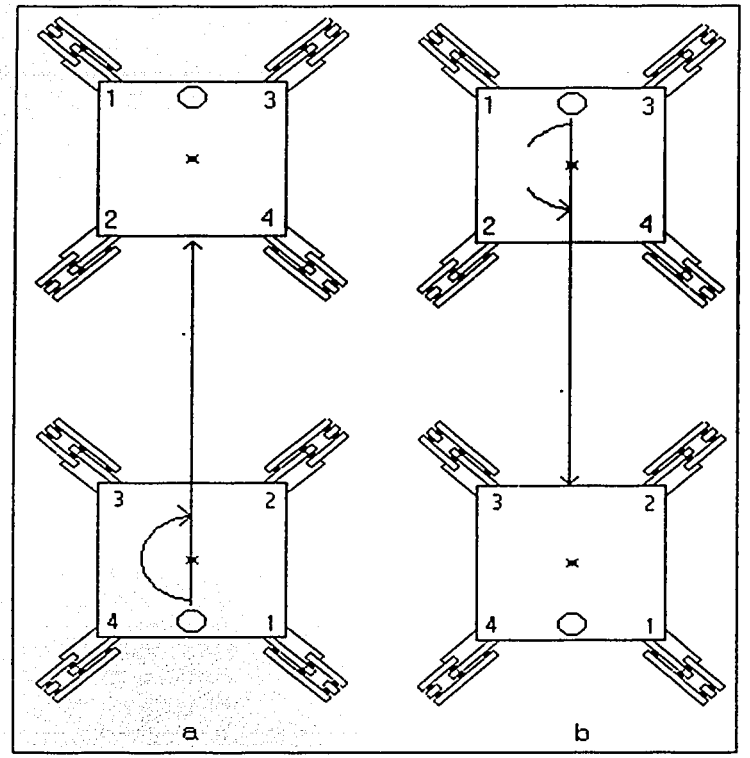
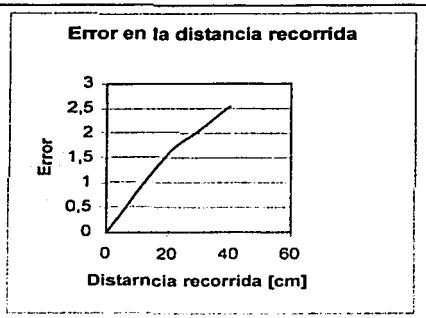


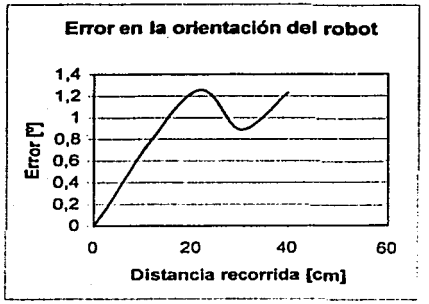
Figura 5.2 Experimento 2.

En los experimentos realizados se pudo apreciar la presencia de errores tanto en la posición y en la orientación, entre los recorridos deseados y los obtenidos, estos errores se existieron principalmente a deslizamientos que se presentaron en las patas al llevar a cabo la tracción del robot, además del juego que existe en las articulaciones del mismo. Dichos

errores pueden ser eliminados si se utiliza un sistema de monitoreo del desplazamiento real del robot al momento de que cada una de las patas lleva a cabo la tracción, de tal forma que se pueda conocer de manera precisa el efecto de dicho movimiento en la posición del robot. Aquí cabe mencionar que todos los desplazamientos se realizaron, fueron hechos en lazo abierto, no existió forma alguna de saber si el las patas realmente realizaron la tracción de forma adecuada.



Gráfica 5.5 Error en la distancia recorrida para el segundo experimento



Gráfica 5.6 Error en el ángulo para el segundo experimento

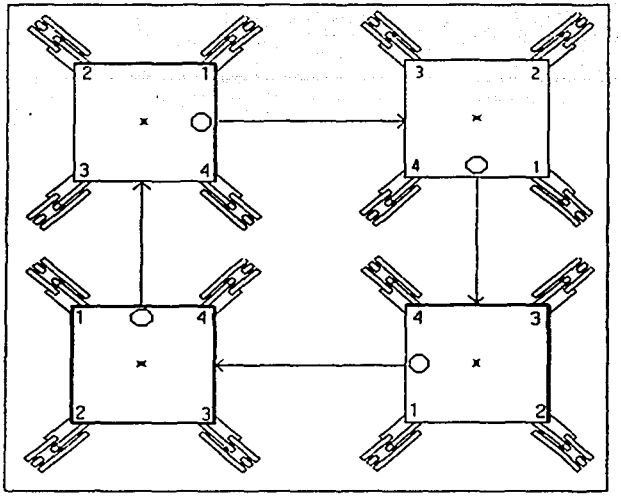


Figura 5.3 Experimento 3

CAPÍTULO 6

PERSPECTIVAS FUTURAS

Un aspecto importante a considerar para futuros trabajos, es el que se refiere al modelado y control de robot. Comenzando por su modelado, para posteriormente continuar con el diseño de un sistema de control, y por último un esquema que permita la detección e identificación de fallas. Para poder llevar a cabo lo anterior existen tres posibilidades: una es la de emplear control clásico (como son control robusto, control adaptable, etc.), el segundo esquema es el de hacer uso de control inteligente (como son redes neuronales y lógica difusa, así como programación evolutiva), y la tercera es la combinación de ambos casos. Por el tipo de robot que se tiene, sería conveniente hacer uso de un sistema de control distribuido, en el cual se considere al sistema como un conjunto de robots que están trabajando de manera conjunta.

En el capítulo, nos enfocamos al modelado y diseño del controlador por medio de control inteligente, presentando una alternativa que se podría seguir.

Como ya se mencionó el primer paso a seguir es la obtención de un modelo matemático que represente al robot. Para poder realizar el modelado, tanto por redes neuronales como por lógica difusa, será necesario entrenar el esquema propuesto, para ello se requerirá de conocer la respuesta que tiene el sistema ante una serie de diferentes entradas. Como entradas, se podría utilizar el giro que cada uno de los motores aplica al robot, y como salida las variaciones que en su posición sufra el robot con cada uno de los torques aplicados.

En las siguientes secciones se hace en primer lugar una descripción de las redes neuronales aplicadas al control, en especial al modelado y sistemas

de control que se podrían aplicar en el robot, en segundo lugar se presenta una propuesta empleando lógica difusa aplicada específicamente al robot.

6.1 Aplicación de redes neuronales en control

Quizás el desarrollo más excitante e innovador durante los últimos diez años en el campo del control, ha sido la introducción de métodos de redes neuronales artificiales y de lógica difusa para el modelado, identificación y control de sistemas físicos. El propósito de esta sección es el de dar una introducción a esta área.

6.1.1 Identificación de sistemas dinámicos

Es bien conocido que una función continua puede ser representada arbitrariamente bien por polinomios, de ahí que surja la idea de aproximar de forma similar un modelo no lineal por medio de redes neuronales. Un sistema de identificación por medio de redes neuronales estándar es el que muestra en la figura 6.1, donde el entrenamiento de una red neuronal está basado en el uso de los datos de entrada y salida de planta. El objetivo global es obtener una red neuronal que se pueda reproducir de una manera idéntica el comportamiento de la propia planta, en donde la red actúa como una aproximación no lineal.

Modelado no lineal ARMA

Bajo ciertas condiciones cualquier función no lineal puede ser modelada por una serie de Wiener o Volterra. Para iniciar, se considera una representación MA (Moving Average), definida como:

$$y(k+1) = \sum_{i=0}^{\infty} a_i u(k-i) \qquad 6.1$$

Donde y es la salida del sistema, u la entrada, K es el tiempo (es un sistema discreto), a_i son los parámetros a ajustar e i es el numero de muestra anterior a ser considerada. Una representación generalizada de las series de Wiener y Volterra da como resultado un sistema dinámico invariante en el tiempo operado por una transformación no lineal. La serie de Volterra es mostrada en la ecuación 6.2, incluyen una combinación lineal de entradas previas con respecto al tiempo:

$$y(k+1) = a_0 + \sum_{i=0}^{\infty} a_i\, u(k-i) + \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_{ij}\, u(k+i)\, u(k-j) + \ldots \qquad 6.2$$

En la cual a_0, a_i y a_j son coeficientes desconocidos.

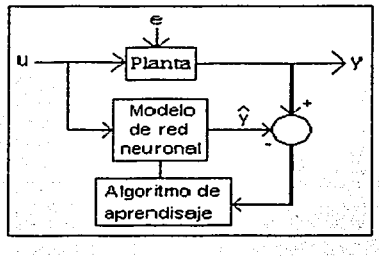


Figura 6.1 Sistema de identificación por medio de redes neuronales

Las redes neuronales incluyen una capa oculta que puede aproximar arbitrariamente cualquier mapeo no lineal, así que la red neuronal mostrada en la figura 6.2 puede representar cualquier función no lineal, que también puede ser modelada por una serie de Volterra (ecuación 6.2), una condición para lo anterior es que todos los valores de la entrada pasados están disponibles en la entrada de la red.

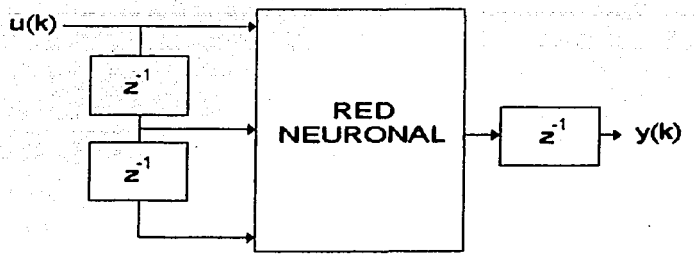


Figura 6.2 Diagrama de bloques de una red neuronal

En el esquema mostrado en la figura 6.2, se podrá ver que la red neuronal no contiene un número infinito de señales de entradas. La situación es normal en el caso de identificación, en donde solo un número finito de términos son usados, tal que se puede alcanzar un razonable nivel de exactitud. Otra manera de ver lo anterior es, si n señales de entrada previas se aplican a la red, entonces se asume que la entrada aplicada inmediatamente antes a n períodos de prueba tiene efectos despreciables en la presente señal de salida.

Modelos de autoregresión (AR) son también, en algunos casos, una forma útil de representación del comportamiento de un sistema, y en este caso la salida de la planta es una relación lineal de las salidas previas de la planta. Por la restricción de tener un modelo con n señales previas, la salida estará dada por:

$$y(k+1) = \sum_{i=0}^{n-1} a_i\, u(k-i) \qquad 6.3$$

Esta forma de modelo es útil para una descripción de series en el tiempo, involucrando un número finito de parámetros, en los cuales ninguna señal es directamente aplicada.

Finalmente, por la combinación de los modelos AR y MA, se puede conocer cualquier sistema dinámico de orden n. En tiempo discreto el modelo puede ser representado por la ecuación en diferencias

$$y(k+1) = \sum_{i=0}^{n-1} a_i u(k-i) + \sum_{i=0}^{n-1} b_i u(k-i) \qquad 6.4$$

la cual es usualmente conocida como modelo ARMA.

6.1.2 Diseño de un controlador

En la actualidad se han incrementado la cantidad de trabajos encaminados a la construcción y control de robots caminantes. Los cuales se han enfrentado a dos obstáculos, que son: el control y la coordinación de sus patas; es por ello que se ha buscado realizar esquemas de control, basados en control neurobiológico de la locomoción de insectos, éstos están caracterizados por su naturaleza distribuida, además de buscar la posibilidad de que éstos sean robustos a perturbaciones, con el propósito de continuar trabajando pese a la variación de las condiciones de su ambiente, cargas externas e incluso ante la posibilidad de sufrir daños en su estructura.

El que el controlador sea robusto, cobra gran importancia, si se tiene como propósito que el robot pueda trabajar de forma adecuada en ambientes complejos y potencialmente hostiles. En este sentido, se cuenta con dos aspectos a considerar. El primero, se enfoca al hecho de que el robot pueda trabajar pese a cambios inesperados en su ambiente, tales como: la fricción y la irregularidad del terreno. Segundo, que el

sistema pueda seguir trabajando de forma autónoma, pese a sufrir algún daño en su estructura.

Un esquema, que se puede utilizar para poder realizar el diseño de un controlador por medio de redes neuronales, es el que presentan K. G. Pearson y sus colaboradores en 1973. En este caso el controlador consta de tres neuronas para los movimientos horizontales (uno para el motor, uno para el movimiento hacia delante y otra para el movimiento hacia atrás) dos para los sensores (uno para el giro hacia atrás y otro para el giro hacia delante). Una neurona de marcapasos por cada una de las patas, además una neurona central que controla al robot. Con lo que se tiene una red de 25 neuronas. El estado de cada una de las neuronas, está gobernado por la ecuación

$$C_i \frac{dV_i}{dt} = \frac{-V_i}{R_i} + \sum w_{ij} f_j (V_j) + INT_i + EXt_i$$

Donde v_i, r_i, c_i representan al voltaje, resistencia y capacitancia que tiene la membrana de la neurona i - ésima, es la fuerza de la conexión entre la j- ésima e i- ésima neurona, f es una función de activación que se incrementa linealmente hasta que alcanza una saturación. EXT_i es la corriente externa inyectada a la neurona y INT es un tiempo y un voltaje dependiente de una corriente intrínseca originada por la oscilación de las neuronas de las patas restantes. El período de oscilación de una neurona, dependerá del nivel de sinapsis en la entrada de la neurona.

El control para una pata sencilla es esquematizado en la figura 6.3.

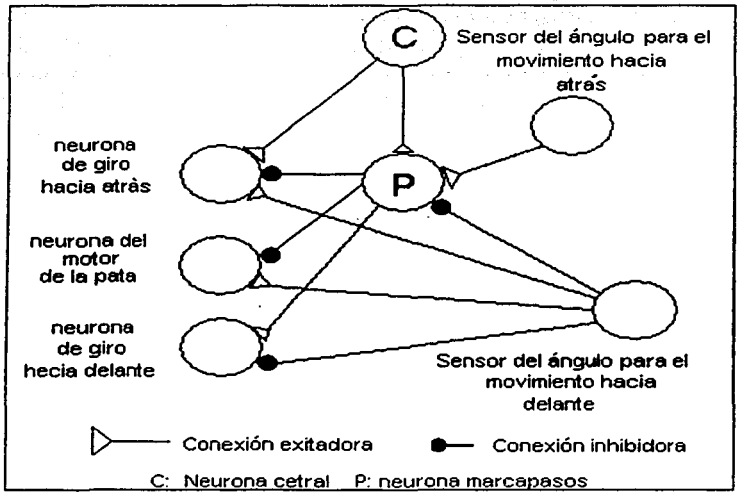


Figura 6.3 Esquema de un red neuronal para una pata del robot.

6.2 Modelado y control utilizando lógica difusa

6.2.1 Modelado

Una de las formas de llevar a cabo el modelado y control de sistemas dinámicos es por medio del uso de lógica difusa. Independientemente del sistema difuso que se utilice (tipo Mandani o Sugeno), es necesario determinar los parámetros de las funciones membresía del sistema difuso, tanto de las funciones consecuentes como las antecedentes (tanto en el caso de los sistemas difusos de tipo Mandani, como Sugeno, los parámetros a determinar en las funciones membresía de los antecedentes son los mismos, las funciones membresía y los parámetros necesarios se muestran en la figura 6.4 mientras que los parámetros de los

consecuentes en el caso de los sistemas difusos de tipo Mandani son los mismos que de los antecedentes, el caso de los sistemas tipo Sugeno los parámetros a utilizar son la pendiente y la ordenada de rectas y los intervalos en los que se aplicarán como se puede ver en la figura 6.5), para lograr hacer esto, existen dos formas, una de ellas es hacerlo de acuerdo a la experiencia que se tenga, o bien haciendo uso de algoritmos genéticos, por medio de los cuales, seráa posible determinar todos los valores del sistema difuso.

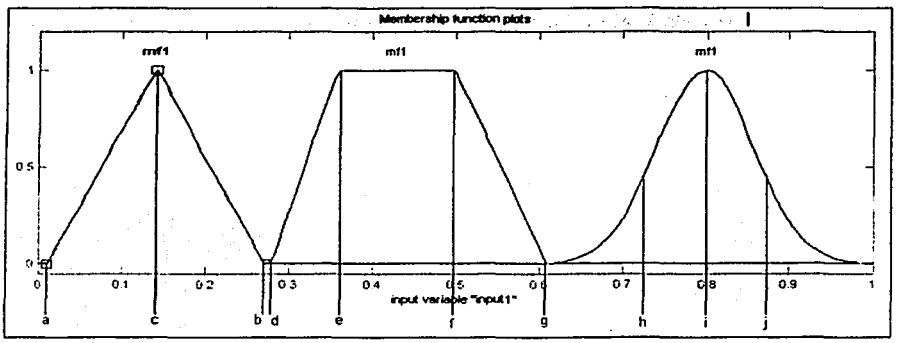


Figura 6.4 Posibles funciones membresía a utiliza para un sistema difuso

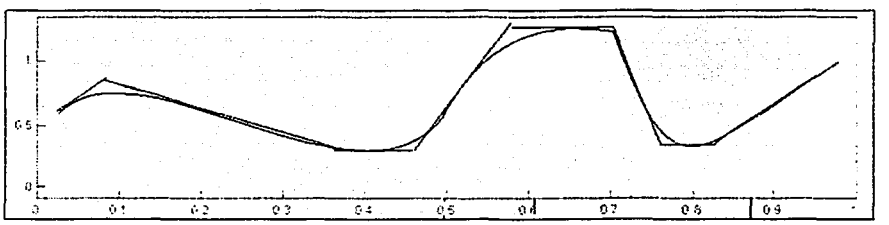


Figura 6.5 aproximación de funciones por rectas

Para poder realizar la determinación de los parámetros de un sistema de lógica difusa por medio del uso de algoritmos genéticos, es necesario conocer dos tipos de señales: las entradas aplicadas y la respuesta obtenida a dichas entradas. En este caso, las entradas aplicadas al sistema son: los torques que los motores ejercen sobre la estructura del robot. Como salida del sistema se puede considerar la posición, respecto a un sistema de referencia, del centro de gravedad del robot. Como se muestra en la figura 6.6.

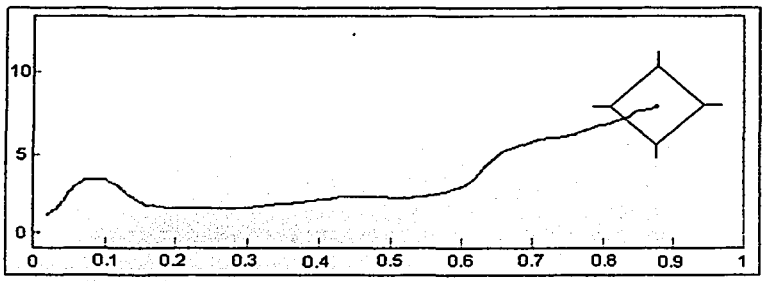


Figura 6.6 Trayectoria seguida por el robot

En la figura 6.7 se muestra la salida del voltaje de un potenciómetro que se conecta a la flecha de un motor, con el propósito de poder determinar la posición angular que guarda. Las líneas crecientes representan el giro en un sentido (por ejemplo horario), y las decrecientes el giro en sentido contrario (antihorario). Un aspecto a resaltar, es el de que la señal de los motores de la patas del robot permanecerá por algunos instantes en valores constantes. Como se puede ver en la figura 6.8. El conjunto de señales obtenidas de los potenciómetros correspondientes a cada motor y la señal correspondiente a la posición del centro de gravedad, son las que se pueden utilizar con el propósito de que haciendo uso de algoritmos genéticos se puedan determinar los parámetros del sistema de lógica difusa que se emplee.

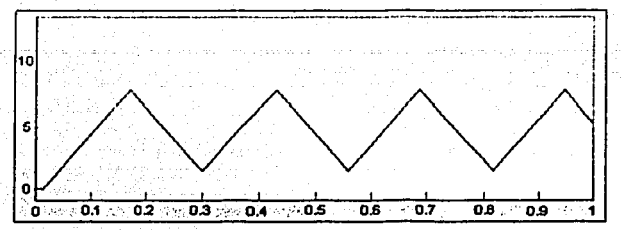


Figura 6.7 voltaje en el potenciómetro para determinar el control de posición de una pata del robot

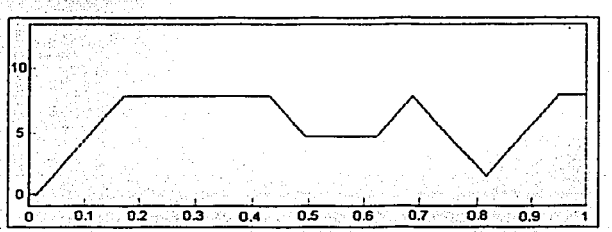


Figura 6.8 Comportamiento del potenciómetro de una pata cuando se tiene movimiento secuencial

6.2.2 Diseño del controlador

Un aspecto que se puede ver en la figura 6.6, es que el conocer la posición del centro de gravedad, nos permitiría únicamente controlar la posición del centro de gravedad, mas no así la orientación en que se encuentre, es por ello necesario poder conocer la posición que tienen por lo menos dos de los puntos del cuerpo del robot (a y g), para con ellos poder determinar la orientación que guarda el robot respecto a un eje, como se muestra en la figura 6.9. Con la ubicación de los dos puntos se podrá contar con dos señales de error, por un lado, la diferencia entre la

posición del robot y la trayectoria deseada, además, se pude obtener el error angular entre la línea que forman los dos puntos en el cuerpo del robot que se conocen y una línea que se elija; estas líneas podrían ser la recta tangente al punto donde se encuentre el centro de gravedad del robot (figura 6.9), o bien una recta paralela a alguno de los ejes del sistema de referencia (figura 6.10) y a partir de ellas determinar los parámetros del controlador (el diseño del controlador presenta una desventaja, que es la necesidad de tener experiencia en el diseño de controladores difuso), de lo contrario será necesario llevar a cabo un proceso de prueba y error para poder determinar los parámetros de las funciones membresía de las funciones antecedentes y consecuentes del controlador.

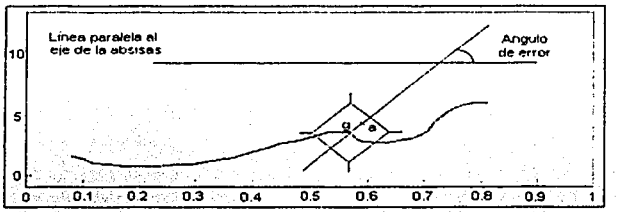


Figura 6.9 Medición del error de orientación de un robot por medio en comparación a una recta paralela a los ejes de coordenados

En la actualidad se cuenta con varias herramientas que se pueden utilizar para el diseño del modelo matemático y el controlador del sistema, un ejemplo de ellos son las funciones fuzzy y anfis de MATLAB.

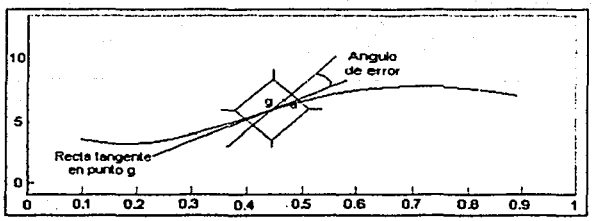


Figura 6.10 Medición del error de orientación de un robot por medio de la recta tangente a la trayectoria que se desea siga.

CAPÍTULO 7

CONCLUSIONES

A continuación se presentan las conclusiones a las que se pudo llegar de acuerdo a lo observado en el desarrollo del proyecto, éstas se dividen básicamente en dos. Las primeras están relacionadas a sugerencias respecto a la estructura del robot, así como a cuestiones en las que se podría trabajar con el propósito de mejorar el desempeño del robot. El segundo tipo es con respecto a los algoritmos diseñados, así como a posteriores diseños que se puede hacer de éstos, sobre todo si se maneja la posibilidad de incrementar el número de patas en la estructura del robot.

En lo que se refiere a la estructura del robot, considero que si bien se puede conservar la forma del cuerpo del robot, sería necesario intentar adaptar otro tipo de motores, en los que controlan el movimiento de subir y bajar cada una de las patas, por unos que proporcionen un mayor torque, para poder dar un mejor soporte a toda la estructura, además de permitir transportar un mayor peso que el actual. En los motores que manejan el desplazamiento horizontal del robot, presentan un juego importante en los engranes junto con los eslabones de las patas, lo cual produce que al encontrarse frente a obstáculos grandes gire la pata; pero sin presentar en realidad un desplazamiento proporcional en el robot, y sí que el motor realice un esfuerzo mayor del cual soporta la estructura interna (en especial los engranes).

Una situación que también se deberá de buscar cambiar es la de que el robot únicamente se puede mover una pata a la vez lo cual por ser un robot de 4 patas se presenta tanto por estabilidad como por el mini SSC II que mueve un motor a la vez, pero para robots de más de 4 patas esta

situación sería contraproducente, debido a que no se aprovecharía de manera eficiente al robot, por lo que se deberá de buscar el movimiento simultáneo de las patas, con el fin de obtener una mayor velocidad en los desplazamientos. Los tres aspectos anteriores también se podrían omitir si se diseñara un modelo matemático del robot, en el cual se tomen en cuenta las variaciones de la fricción de las superficies sobre las cuales el robot se desenvolverá, y el juego que se tiene en los engranes del motor, situaciones que se podrían considerar como perturbaciones o bien como fallas en el sistema y de acuerdo a estas consideraciones diseñar un esquema de control robusto.

En lo que se refiere a los algoritmos, si bien existen varias secuencias que pueden dar un mejor desempeño del robot de acuerdo a ciertas aplicaciones y a un desplazamiento angulares mínimos, esto en propósitos generales presentan inestabilidad, esto se debe principalmente a que el centro de gravedad del robot, se encuentra fuera del área que forman las patas que permanecen en contacto con el suelo o bien sobre las aristas de estas áreas, lo cual si bien no siempre resulta en inestabilidad sí produce un balanceo de la estructura, por lo cual cualquier algoritmo que mantenga al centro de gravedad dentro estas áreas, puede ser considerado para realizar los desplazamientos.

Por la combinación de los movimientos de las patas, los algoritmos que se encargan de realizar los desplazamientos hacia atrás, adelante, izquierda y derecha presentan trayectorias en las cuales el centro de gravedad describe una señal en forma de diente de sierra (figura 6.1), de tal forma que la velocidad del robot dependerá de la frecuencia que la señal tenga, y la distancia que recorrerá por ciclo será proporcional al coseno del ángulo de giro de los motores del robot.

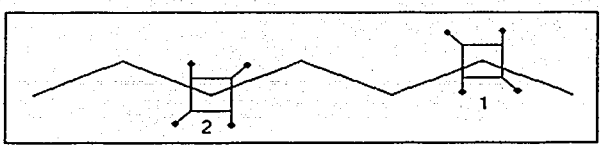


Figura 6.1 Trayectorias que sigue el robot al desplazarse.

APÉNDICE A

ESQUEMAS DE LA ESTRUCTURA DEL ROBOT Y CIRCUITO MULTIPLEXOR

A continuación en la figura A.1 se presentan los esquemas de las piezas que componen cada una de las patas del robot en dimensiones reales.

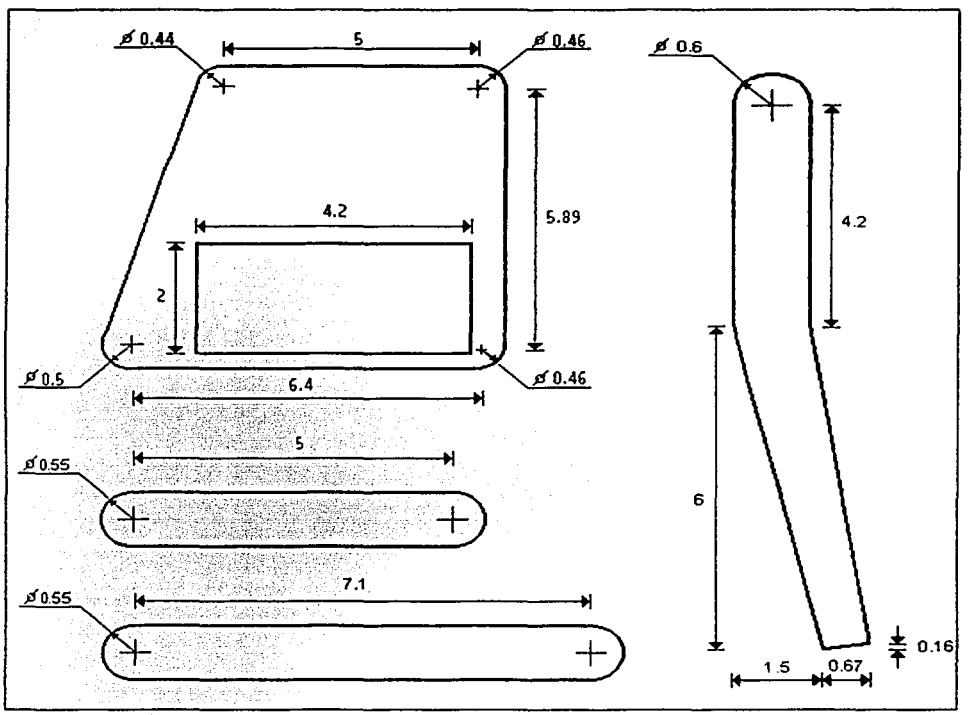


Figura A.1. Eslabones de las patas del robot.

En la figura A.2 se presenta el circuito impreso para el multiplexor los puertos serie del mini SSC II, el 64HC11F1 y la computadora

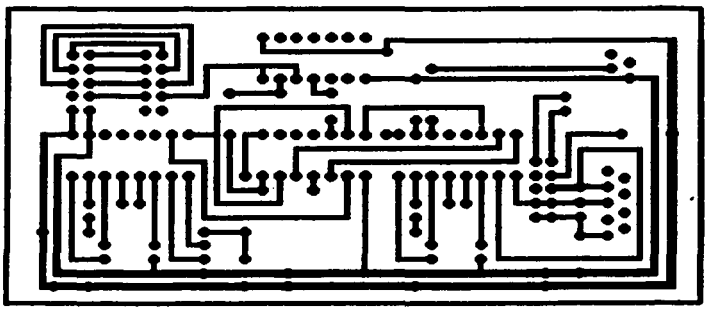


Figura A2 Circuito impreso del multiplexor de comunicación vía puerto serie

APÉNDICE B

FUNCIONES PARA LA PROGRAMACIÓN DE LOS ALGORITMOS

A continuación de presenta las funciones que se utilizaron para la programación de los algoritmos del robot, esta funciones fueron programadas en lenguaje C

```
void move_m(int motor,int angle){
        transmit(PORT_CONT,255);
        transmit(PORT_CONT,motor);
        switch(motor){
                case 4: case 0: case 7: case 3:
                        transmit(PORT_CONT,angle);
                        break;
                case 6: case 2: case 5: case 1:
                        transmit(PORT_CONT,250 - angle);
                        break;
        }

}


void displace_foot(int a,int b,int c,int d,int disp){
        move_m(d,30);
        move_m(a,MOTOR[a]+disp);
        move_m(d,MOTOR[d]);
        move_m(b,250);
        move_m(c,MOTOR[c]+disp);
        move_m(b,30);
        move_m(d,250);
        move_m(a,MOTOR[a]);
        move_m(d,MOTOR[d]);
```

```
        move_m(c,MOTOR[c]);
        move_m(b,MOTOR[b]);
}

void back(float distance,float time){
        float speed,dpc,dr;
        int disp1,disp2,ciclos,i;

        speed = distance / time;
        disp1 = 3820.79 * speed - 3.4920;

        if(disp1 > 50){             disp1=50;   }

        dpc = 0.001746 * disp1 - 0.00634;
        ciclos=distance / dpc;
        dr=distance-dpc*ciclos;
        disp2=(dr+0.00634)/0.001746;

        if(distance != 0){
                for(i=0;i<ciclos;i++){
                        displace_foot(0,3,2,1,disp1);
                        displace_foot(6,5,4,7,disp1);
                }
                displace_foot(0,3,2,1,disp2);
                displace_foot(6,5,4,7,disp2);
        }
}

void forward(float distance,float time){
        float speed,dpc,dr;
        int disp1,disp2,ciclos,i;

        speed = distance / time;
```

```
        disp1 = 4193.71 * speed + 1.483;

        if(disp1 > 50){
                disp1=50;
        }

        dpc = 0.00199 * disp1 - 0.00719;
        ciclos=distance/dpc;
        dr=distance - dpc*ciclos;
        disp2=(dr+0.00719)/0.00199;

        if(distance != 0){

                for(i=0;i<ciclos;i++){
                        displace_foot(4,7,6,5,disp1);
                        displace_foot(2,1,0,3,disp1);
                }

                displace_foot(4,7,6,5,disp2);
                displace_foot(2,1,0,3,disp2);
        }
}

void to_turn_pos(float angle){
        int disp,cic,i;
        float ang;

        if(angle <= 0.52){
                disp = (int) 81.96 * angle + 3.521;
        }
        else{
                cic = (int) angle / 0.52;
                ang = angle - cic * 0.52;
```

```
            disp = (int) 81.96 * ang + 3.521;
            for(i=0;i<cic;i++){
                displace_foot(0,3,2,1,0.52);
                displace_foot(4,7,6,5,0.52);
            }
        }
        if(disp >=1){
            displace_foot(0,3,2,1,disp);
            displace_foot(4,7,6,5,disp);
        }
}

void to_turn_neg(float angle){
        int disp,cic,i;
        float ang;

        if(angle <= 0.698){
            disp = (int) 61.069 * angle + 4.054;
        }
        else{
            cic = (int) angle / 0.698;
            ang = angle - cic * 0.698;
            disp = (int) 61.069 * ang + 4.054;
            for(i=0;i<cic;i++){
                displace_foot(6,5,4,7,0.698);
                displace_foot(0,1,0,3,0.698);
            }
        }
        if(disp >= 1){
            displace_foot(6,5,4,7,disp);
            displace_foot(2,1,0,3,disp);
        }
}
```

```
void left(float distance,float time){
        float speed,dpc,dr;
        int disp1,disp2,ciclos,i;

        speed = distance / time;
        disp1 = 4022.26 * speed + 2.569;

        if(disp1 > 50){
                disp1=50;
        }

        dpc = 0.00199 * disp1 - 0.006809;
        ciclos=distance/dpc;
        dr=distance - dpc*ciclos;
        disp2=(dr+0.006809)/0.00199;

        if(distance != 0){

                for(i=0;i<ciclos;i++){
                        displace_foot(2,5,4,3,-disp1);
                        displace_foot(0,7,6,1,-disp1);
                }

                displace_foot(2,5,4,3,-disp2);
                displace_foot(0,7,6,1,-disp2);
        }
}

void rigth(float distance,float time){
        float speed,dpc,dr;
        int disp1,disp2,ciclos,i;
```

```
        speed = distance / time;
        disp1 = 41019.9 * speed + 4.31;

        if(disp1 > 50){
                disp1=50;
        }

        dpc = 0.00224 * disp1 - 0.002309;
        ciclos=distance/dpc;
        dr=distance - dpc*ciclos;
        disp2=(dr+0.002309)/0.00224;

        if(distance != 0){

                for(i=0;i<ciclos;i++){
                        displace_foot(4,3,2,5,disp1);
                        displace_foot(6,1,0,7,disp1);
                }

                displace_foot(4,3,2,5,-disp2);
                displace_foot(6,1,0,7,-disp2);
        }
}
```

BIBLIOGRAFÍA

1.- "Introducción a la estadística matemática, principios y métodos"
Edwin Kreyszig. Ed Limusa 1991

2- "Robótica práctica, Tecnología y aplicaciones"
José Ma. Angulo Usastegui

3.- "Robótica, control, detección e inteligencia"
K. S. Fu, A. C. Gonzáles. Ed McGraw Hill 1978

4.- "Robótica, una introducción"
D. McCloy. Ed Limusa

5.- "Elementos de robótica."
P. Corffet, M. Chirouze