UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

DIVISIÓN DE ESTUDIOS DE POSGRADO FACULTAD DE INGENIERÍA

CONTROL DE UN ROBOT MÓVIL BASADO EN UN SISTEMA EXPERTO Y LÓGICA DIFUSA

QUE PARA OBTENER EL GRADO DE
MAESTRO EN INGENIERÍA
(ELÉCTRICA)

P R E S E N T A

CÉSAR JIMÉNEZ CALVILLO

DIRECTOR
DR. JESÚS SAVAGE CARMONA

Ciudad Universitaria, México, D. F., Diciembre 2003

A

Dedicatoria

A mis padres: Cándida (q.e.p.d) y Tiburcio, por que este logro es parte del resultado de la educación que con muchos sacrificios me brindaron.

Agradecimientos

A la UNAM,
Con mi reconocimiento eterno, por ser fuente de pensamiento y razón.

A los profesores de la División de Estudios de Posgrado de la facultad de Ingeniería, que compartieron conmigo sus conocimientos y experiencias.

Al Dr. Jesús Savage Carmona por su amistad, por el apoyo y los conocimientos que me brindo en la realización de este trabajo, por ser un académico comprometido con las nuevas generaciones.

A mis hermanos:
Por ser siempre compañeros en la vida. Por el apoyo recibido en los momentos más difíciles. Gracias por sus alegrías y por todos los momentos que hemos pasado juntos. Especialmente a ti Elia por tu trabajo y entrega.
A ti Josué por ser mi testimonio de vida, espero que lo que hemos compartido juntos lo transmitas a la nueva generación familiar porque a ti te tocó la oportunidad de convivir más tiempo con nosotros, al mismo tiempo has adquirido un compromiso mayor, ahora supéranos es tu gran reto y confío en que lo lograrás.

A Brenda:
Por el apoyo incondicional en la elaboración de este trabajo, por ser parte de este proyecto que hemos formado juntos. Por aceptar que las metas se tiene que alcanzar a base de sacrificios y trabajo diario, por ser parte de esta etapa de superación profesional en mi vida.

A mis amigos:
Por que siempre han estado conmigo en los momentos difíciles y que nunca me han regateado su amistad, comprensión y ayuda.

A mis compañeros de trabajo:
Por que me han dado la oportunidad de colaborar, aprender y desarrollarme como profesionista.

A todas aquellas personas que se vieron involucradas en mi trabajo y que de uno u otra forma han contribuido para que este momento pueda ser posible.

Agradezco a Dios por todas las bendiciones y por permitirme despertarme cada mañana con la ilusión de seguir soñando.

A los integrantes del jurado:

Profesores de la División de Estudios de Posgrado de la Facultad de Ingeniería y del Instituto de Investigaciones Matemáticas Aplicadas y Sistemas, por aceptar formar parte de mi jurado.

Dr. Rogelio Alcántara Silva,

Dr. Jesús Savage Carmona,

Dr. Héctor Benítez Pérez,

Dr. Marco Arteaga Pérez,

Dr. José Cohen Sak.

Índice General

Lista de figuras y tablas

Capítulo 1

Introducción

1.1 Antecedentes

La robótica es una disciplina que esta relacionada con el diseño y construcción de máquinas que sean capaces de comportarse como el ser humano. Estas máquinas son conocidas como robots [1]. Un robot es un dispositivo mecánico programable que interactúa con el mundo físico para resolver problemas planeados por el hombre [2]. Aunque la industria se intereso en robots en los años 60's, los robots comerciales fueron producidos durante los años 70's. Los primeros robots, llamados robots industriales fueron diseñados para acciones repetitivas de una línea de producción típica. Los avances en tecnología tanto en la área de la computación y de inteligencia artificial, en los 80's estimularon gradualmente la ambición de la robótica. Las investigaciones en robótica fueron orientadas primordialmente en robots con alto grado de inteligencia [1].

El robot inteligente debería de ser equipado con al menos 3 tipos de funciones: *Apropiadas funciones perceptuales*, tales como sensores que permitan al robot percibir el ambiente; *funciones de procesamiento de la información* que permitan al robot procesar la información recibida; y *funciones mecánicas* con apropiado control para permitir al robot moverse y actuar como desee [1].

Tradicionalmente los robots fueron diseñados en un camino donde la acción de planeación y el controlador eran tratados como cuestiones separadas. Los diseñadores de sistemas robotizados se concentraron en el diseño del controlador, y la acción de la planeación y navegación, fue dejada como una tarea al usuario del robot. Esto es entendible, porque la acción de planeación dependía del medio ambiente en que el robot desarrollara dicha tarea [2], [4].

La investigación en el área de la Robótica se ha venido desarrollando en el laboratorio de interfases inteligentes (LII) del Departamento de Computación de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, como consecuencia de ello se han venido construyendo varios prototipos de

PAGINACION DISCONTINUA

robots móviles, así como la adquisición de robot B14 de la empresa Real World Interface (RWI).

Debido a que un robot móvil es un dispositivo que opera en el mundo físico y sus acciones necesarias para concluir una tarea está determinada por la naturaleza de la misma, se han implementado técnicas de planeación y movimiento. Una de las técnicas de control de planeación y navegación para los movimientos del robot, la cual ha tenido aplicaciones satisfactorias en muchas de las áreas de la ingeniería es la Lógica Difusa.

La Lógica Difusa es una técnica relacionada con la inteligencia artificial (AI), que ofrece grandes ventajas en el manejo de problemas en los cuales no es fácil definir un modelo matemático. La raíz de lo anterior se deriva de su habilidad para esquematizar soluciones y generar respuestas basadas en información vaga, ambigua, cualitativa, incompleta o imprecisa. La teoría difusa involucra cuantificadores utilizando lenguaje natural en las que muchas palabras tienen significado ambiguo, tales como "alto", "caliente", "un poco", "mucho" [5]. Así la Lógica Difusa utiliza conjuntos difusos para diseñar sistemas que son capaces de captar en forma de reglas, la habilidad que todo ser humano posee para modelar un sistema o proceso utilizando el lenguaje natural. El concepto y fundación teórica del control difuso fue desarrollado por Lotfi. A. Zadeh en 1965 [4], [5], [12]. Esta última ha sido usada de manera exitosa, prácticamente en todos los campos técnicos, incluyendo sistemas expertos, procesamiento de imagen y señales.

El mayor éxito se ha alcanzado en el área de control automático, aquí los denominados controladores lógicos difusos (CLD), o simplemente controladores difusos, han emergido como una de las áreas de investigación más activas en la aplicación de la teoría de los conjuntos difusos [6]. En el futuro, cabe esperar un notable aumento de aplicaciones industriales; fundamentalmente en los sectores de automóviles, procesos térmicos, industria química, control de medio ambiente y de robótica.

Los controladores basados en Lógica Difusa, pretenden modelar las experiencias de los seres humanos, así como su comportamiento para tomar decisiones; estas ultimas fundamentadas en las experiencias adquiridas por el operador humano del proceso [12].

La investigación más temprana y más formativa sobre CLD ha sido realizada por Mamdani, en Londres a mediados de los 70´s en una maquina de vapor; Más tarde los CLD fueron ampliados por Procyk y Mamdani en 1979 para aplicaciones vehículos autónomos dirigidos. Los pilotos automáticos de barcos han sido áreas populares para la aplicación de CLD. Zadeh en un principio concibieron la lógica difusa como un paradigma apropiado para la conducción

de vehículo. No es por lo tanto sorprendente que CLD ha encontrado un uso muy amplio en este tipo de aplicaciones.

Una aplicación de CLD fue desarrollada para la conducción de un vehículo automático el cual podía ser estacionado en una posición previamente determinada y en un espacio limitado [29]. La aplicación consistió en sistemas con dos de los pilotos, uno lateral y otro longitudinal. El piloto lateral calculaba la curvatura deseada de la trayectoria y la trayectoria consiguiente. Mientras que la tarea del piloto longitudinal era la de controlar la velocidad del vehículo. La trayectoria fue considerada un sistema de líneas rectas para formar una trayectoria general, en la cual se definió una mínima curvatura en la trayectoria a través de los pasillos libres de obstáculos para reducir al mínimo la aceleración lateral [29].

Otra aplicación de un controlador difuso para problema de estacionamiento de un carro, fue diseñado por Sugeno y Murakami. Las reglas difusas de control fueron derivadas de la manera de como un experto conduce un coche. En los procesos donde la habilidad de un operador es importante, es muy útil el control de reglas difusas por el modelado de acción de control de un operador. El coche modelo utilizado estaba equipado con un microprocesador Z80 como el cerebro del controlador difuso, además de unos convertidores analógicos-digital y digital-analógicos, para el control de sus motores; unos sensores eran utilizados para determinar la posición vehículo [13].

1.2 Planteamiento del problema

Implantar mecanismos de planeación y navegación con robots móviles, utilizando técnicas Lógica Difusa y un Sistema Experto para dotar de conocimiento al robot.

1.3 Objetivo

El objetivo de este trabajo es contar con robots móviles que ejecuten órdenes bajo condiciones iniciales, así como cambios de posición y orientación dentro de un área previamente definida.

En este trabajo se presentan dos aplicaciones; bajo diferentes plataformas de programación.

La primera aplicación se realiza en un robot móvil diseñado y construido en el Laboratorio de Interfases Inteligentes, donde el elemento principal de control

es un sistema mínimo basado en el microcontrolador MC68HC11E9 fabricado por Motorola, el algoritmo difuso fue programado en el lenguaje C.

La segunda aplicación fue implantada en el robot móvil B14, para el desarrollo del algoritmo difuso, se utilizo un lenguaje para sistemas expertos (CLIPS). El robot utilizará un controlador difuso para realizar sus movimientos, además un sistema experto que nos servirá para programar algoritmos de planeación y navegación.

Al final de esté trabajo se pretende comprobar y verificar dos aspectos importantes.

a) Que la teoría de lógica difusa facilite el desarrollo de un control eficiente para la manipulación de los movimientos de un robot móvil.

b) Que los sistemas expertos sean utilizados para resolver problemas de navegación.

Los hechos arriba mencionados han motivado a la realización de éste trabajo de tesis, que es la implementación de control basado en lógica difusa y sistemas expertos en robots móviles.

1.4 Desarrollo

El presente trabajo se estructura de la siguiente manera: en el capítulo 2 se describe un resumido panorama de la inteligencia artificial, así como las características y componentes más importantes de un sistema experto.

En el capítulo 3 se describen las bases teóricas de los principales fundamentos matemáticos sobre lógica difusa: definición de variables lingüísticas, el concepto de conjuntos difusos, relaciones difusas, además de la construcción a bloques y parámetros importantes de un controlador difuso. Se describen los modelos de control difuso utilizados en la actualidad en función de sus características estructurales.

La planeación y el modelo matemático de los movimientos utilizados en la navegación, así como la organización típica de un robot móvil y sus características principales, son presentados en el capítulo 4.

Por lo que respecta al capítulo 5, se describe la arquitectura de los robots utilizados; se explica paso a paso la construcción y diseño del robot utilizado

en la primera aplicación; así como las etapas más importantes. Para la implantación de nuestra aplicación utilizando el Robot B14, se explica la plataforma de desarrollo, la comunicación entre los diferentes módulos del B14; así como los requerimientos de comportamiento, necesario para llegar a nuestro objetivo, además se muestran fragmentos de programas codificados en CLIPS para su mejor descripción y entendimiento.

El capítulo 6 describe las pruebas realizadas para verificar el funcionamiento de los robots.

Dentro del capítulo 7 se presentan las conclusiones del trabajo y algunas recomendaciones para modificaciones futuras en los algoritmos.

Para finalizar, este trabajo contiene 6 apéndices organizados de la siguiente manera: El apéndice A, ilustra los elementos básicos de programación en CLIPS. El apéndice B, describe las características principales del puerto serial de comunicación del MCU 68HC11, así como de la etapa de potencia para el control de motores. El apéndice C, muestra la manera de programar y ejecutar un programa con el Compilador del HC11, además de una pequeña explicación del Pcbug11. El apéndice D, que describe los elementos principales del robot B14. El apéndice E, muestra la arquitectura diseñada, así como las características importantes de la tarjeta basada en el microcontrolador MC68HC11E9. Finalmente el apéndice F, lista el programa en Lenguaje C utilizado para la primera aplicación.

Capítulo 2

Aplicaciones de la Inteligencia Artificial

Los "robots inteligentes" son capaces de adaptarse a una gran variedad de condiciones cambiando sus acciones basados en la información recibida por sus sensores. La inteligencia depende de los algoritmos que permitan interpretar los datos y tomar decisiones, debido a esto el controlador de un robot debe de basarse en técnicas de inteligencia artificial (IA). Desde que nació la IA como disciplina ha existido la intención de realizar robots autónomos e inteligentes, capaces de realizar las tareas como el ser humano.

2.1 La Inteligencia Artificial

La IA se podría definir como la ciencia que trata de la comprensión de la inteligencia y del diseño de máquinas inteligentes, es decir, el estudio y la simulación de las actividades intelectuales del hombre. Dentro de la IA existen muchos campos como son [17]:

- Robótica.
- Sistemas Expertos.
- Lenguaje Natural.

Uno de los problemas que surgieron en aquel entonces fue la aparición de la explosión combinatoria en los cálculos exhaustivos que limitaban la profundidad en los mismos y el número de conocimientos que se podían procesar, es decir se calculaban todas las posibles vías de solución para luego elegir la óptima. Tanto el software y hardware estaban adaptados al cálculo numérico y no al campo simbólico [17].

La IA tal como hoy la concebimos, hace su aparición en la década de los cincuenta, cuando se empiezan a escribir programas de tipo simbólico para la solución automática de problemas [17].

En la década de los sesenta, coincidiendo con la segunda época de la IA aparecieron numerosos trabajos sobre el método general y universal de resolución desarrollado sobre ordenadores, de ellos el más famoso es el "General Program Solver" de Newell y Simón de la Universidad de Carnege [17], [19].

La primeras aplicaciones de robots Inteligentes fueron construidos a fines de los 60´s y principios de los 70´s, Shakey, es un robot desarrollado en el Instituto de Investigaciones de Stanford, contaba con un medio ambiente diseñado y preparado especialmente para él, una cámara de televisión, como sensor primario, una computadora externa para analizar las imágenes con base en descripciones hechas en la forma de cálculo de predicados de primer orden, que generaban una secuencia de acciones que eran comandos enviados a los actuadores que contaban con realimentación con base en otros sensores como el odómetro y sensores de contacto. En la mayoría de estos sistemas existía la tendencia a utilizar la visión por computadora para reconstruir un mundo tridimensional a partir de imágenes bidimensionales. La IA era utilizada para realizar descripciones del mundo y manipulaciones de los objetos reales [17].

La robótica sólo tenía la misión de la interacción física con el mundo, los problemas principales eran: la planeación de una ruta libre de colisiones; la cinemática y dinámica directa; y la cinemática y dinámica inversa.

2.2 Sistemas Expertos

Un experto humano "es una persona que es competente en una área determinada del conocimiento del saber". Un sistema experto "es una programa de computadora que reemplaza a un experto humano utilizando conocimiento y procedimientos de inferencia para poder resolver problemas" [17].

Un experto humano es capáz de resolver de una forma rápida y eficáz un problema completamente nuevo dentro de un campo que un no experto, esto es debido a que el experto posee además de un conocimiento, estrategias básicas de resolución y numerosas tácticas que le permiten evitar pruebas inútiles o poco útiles [17],[19].

2.2.1 Elementos de un Sistema Experto

Existen muchas maneras de representar el conocimiento, las formas más comunes son: el modelo basado en reglas, los marcos o tramas (frames), las redes semánticas y por objetos, pero la forma más usada para expresarlo es el conocimiento basado en reglas. El conocimiento expresado en reglas es el más utilizado debido a su gran sencilléz al ser la formulación más inmediata del principio de causalidad (causa-efecto) [17].

La representación del conocimiento fue propuesta por Newell y Simon al observar que el conocimiento necesario para resolver problemas, se puede expresar mediante reglas de producción de la forma [19]:

SI <Antecedentes> ENTONCES <Consecuentes>

Un sistema experto trabaja con hechos, que se almacenan en la memoria de trabajo, estos hechos son necesarios para definir las condiciones iniciales en nuestra aplicación. Los antecedentes de las reglas son generalmente cuestionamientos acerca de la existencia o inexistencia de hechos con algún patrón predefinido, mientras que los consecuentes son acciones, que suelen referirse a la afirmación o negación de un hecho. En la figura 2.1 se muestran los componentes de un Sistema Experto [19].

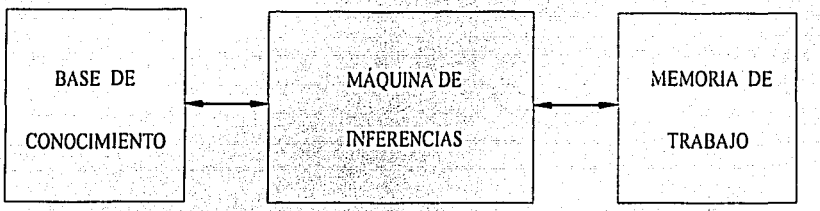


Figura 2.1 Esquema general de un sistema experto.

Además de las reglas y hechos un sistema experto cuenta con una máquina de inferencias, a través de ésta el sistema realiza inferencias tratando de emular la forma de razonar como lo hace el hombre. Cuando los antecedentes se cumplen se dice que la regla está activada.

El proceso de inferencia consiste en encontrar las reglas activadas y ejecutar el consecuente correspondiente a cada una de ellas. La función que realiza la

máquina de inferencia es la de buscar los antecedentes de las reglas y los coteja con los hechos que hay almacenados en la memoria de trabajo.

El sistema experto está formado de dos componentes principales. La base de conocimientos, esta contiene el conocimiento con el que la máquina de inferencia muestra las conclusiones. Estas conclusiones son la respuesta del sistema experto a las preguntas del usuario experto. Un sistema experto se define como "Un programa de computadora que utiliza conocimiento y procedimiento de inferencia para resolver problemas que son lo suficientemente difíciles, que para su solución requieren de un humano experto [19].

Aunque las maneras de construir un sistema experto pueden ser diversas, sus componentes típicos, se muestran en la figura 2.2.

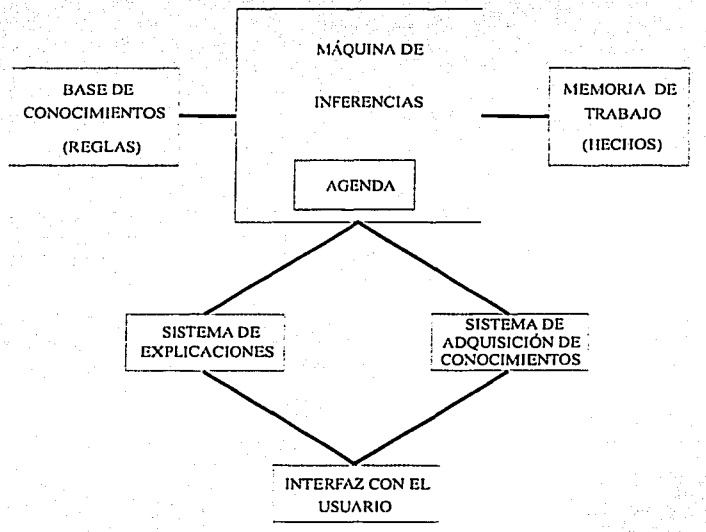


Figura 2.2 Esquema básico de un sistema experto basado en reglas.

Los conocimientos del sistema del experto humano se guardan en la base de conocimiento. También un mecanismo de inferencia hace lo que su nombre implica, inferencias: el proceso por el cual, se llega a una conclusión con base en una serie inicial de proposiciones. Las inferencias que hace el sistema

experto están basadas en su interpretación de la representación de los conocimientos del experto.

En un sistema experto la estrategia de resolución es realmente el control del sistema, que se denomina máquina de inferencia, que constituye una forma dinámica de soluciones. El proceso de inferencia selecciona, decide, interpreta y aplica el conocimiento de la base de conocimientos, sobre la base de hechos con el fin de obtener la solución buscada [19].

Los componentes típicos de un Sistema Experto son 19]:

Interfaz con el usuario. Módulo de comunicación entre el usuario y el sistema experto.

Sistema de explicaciones. Le explica al usuario el mecanismo de inferencia realizado para llegar a una conclusión.

Sistema de adquisición de conocimiento. Permite que el usuario introduzca conocimiento al sistema.

Base de conocimiento. Base de datos global que almacena las reglas.
Memoria de trabajo. Base de datos global que almacena los hechos usados para activar las reglas.

Máquina de inferencia. Realiza la inferencia para determinar que reglas son satisfechas por hechos y ejecuta la de mayor prioridad.

Agenda. Es una lista de reglas creada en el proceso de inferencias cuyos patrones son satisfechos por hechos u objetos en la memoria de trabajo y que por lo tanto son susceptibles de ejecutarse.

2.3 Lenguajes y herramientas para sistemas experto

Desde la década de los sesentas aparecieron numerosos trabajos en el área de sistemas expertos, así como herramientas enfocadas a su desarrollo, Lenguajes tales como LISP creados por John McCarthety en 1960, que fue en los 60' s el desarrollo mas significativo en su momento. PROLOG creado por Alan Colmeraver de la Universidad de Marsella en 1972, que permite alimentar desde instrucciones simples, hasta herramientas que incluso le permiten al usuario introducir el conocimiento a través de ejemplos almacenados en tablas u hojas de cálculo [17].

Pero ninguno de ellos proporcionaba una manera flexible para representar el conocimiento, esto hizo necesario contar con lenguajes de programación para sistemas expertos que permitieran dos niveles de abstracción tanto de datos como de conocimiento [17].

Una herramienta muy útil para el desarrollo de sistemas expertos es CLIPS (C language Integrated Production Systems) [19]. Esta herramienta fue desarrollada en el Centro Espacial Johnson de la NASA a partir de 1984 para profundizar en el conocimiento de la construcción de sistemas expertos. La primera versión de CLIPS que fue puesta a disposición a usuarios fuera de la NASA fue la versión 3.0 en 1986, las versiones que le siguieron fuerón 4.0, 4.1, 5.0 5.1 entre los años 1987 y 1991. Actualmente se usa la versión 6.0 que apareció en 1993. CLIPS es utilizado en Universidades y en numerosas Compañías [19].

Capítulo 3

Lógica difusa

3.1 Introducción

Cuando se desea modelar algún sistema (físico, biológico, etc.) es necesario disponer de un modelo matemático lo cual implica considerar ecuaciones Integro-diferenciales o ecuaciones en diferencias. Sin embargo existen sistemas en los cuales para obtener un modelo basado en tales ecuaciones se tienen que hacer muchas simplificaciones, lo cual implica pérdida de información en su comportamiento dinámico. Debido a que la lógica difusa crea sus propias fuentes de control al combinar conjuntos y reglas difusas, esto permite al diseñador construir un sistema de control aún cuando su comprensión del modelo matemático del sistema sea incompleto [5]. A pesar de que la lógica difusa fue creada hace como 29 años, se ha reconocido como una tecnología útil y rentable para el control industrial, tanto para sistemas no lineales ó donde no es fácil definir un modelo matemático [19].

Los controladores basados en esta teoría, intentan modelar las experiencias de los seres humanos, así como su comportamiento para tomar decisiones y están fundamentados en las experiencias adquiridas por el operador del proceso con los cuales puede controlar éste, aún sin el conocimiento de su dinámica fundamental [16].

Las bases teóricas fuerón desarrolladas por Lofti A. Zadeh, profesor de la Universidad de California, en Berkeley, a comienzos de los sesentas [7]. El primer control que utilizó lógica difusa fue realizado por el profesor E. H. Mamdani, en un sistema de control de vapor. Desde entonces, el número de aplicaciones industriales en lógica difusa se ha extendido, a casi todos los campos de la Ingeniería [14].

La lógica difusa aparece comercialmente por primera vez en un controlador desarrollado por *Hitachi* en el tren subterráneo en la ciudad de *Sendai*, Japón

El controlador gobernaba los aspectos de aceleración, frenado y paro proporcionando un aumento de precisión en la plataforma de parada y sobre todo consumo mínimo de energía eléctrica. Desde entonces, el número de aplicaciones industriales ha ido en aumento [14].

Durante la década pasada la lógica difusa se ha desarrollado en una gran variedad de direcciones, ha encontrado aplicaciones en diversos campos como la topología, lingüística, teoría de autómatas, teoría de control, reconocimiento de patrones y medicina [1].

Este interés creciente se plasmó con la creación de *LIFE (Laboratory for International Fuzzy Engineering Research)* en 1989 en Yokohama, Japón. Actualmente diversas empresas sobre todo japonesas, construyen una serie de productos de consumo difuso, dotados de un control más preciso que los convencionales [9].

En este capítulo se describe una breve revisión de teoría clásica de conjuntos, así como el concepto de función de pertenencia, enseguida se define el concepto de lógica difusa. Posteriormente se dan definiciones básicas en el estudio de la teoría difusa, como la variable lingüística así como algunas consideraciones para el diseño de un Control Difuso.

3.2 Teoría clásica de conjuntos

Un conjunto clásico es normalmente definido como una colección de elementos *u* objetos (discretos o continuos), $x \in U$, pueden ser numerable o no numerables, *U* es llamado el universo de discurso. Cada elemento puede pertenecer o no pertenecer a un conjunto A, $A \subset U$. La forma de representar cuando un objeto es miembro de un conjunto, es por medio de su función característica (función de pertenencia). Si un objeto es elemento de un conjunto la función característica es 1. Si un objeto no es un elemento de un conjunto, entonces decimos que su función característica es 0.

Tradicionalmente una expresión lógica puede caer solo en un de los extremos: completamente verdadero o completamente falso. Sin embargo en la lógica difusa toma valores entre 0 y 1 para representar diferentes valores para la función característica.

3.2.1 Función de pertenencia o membresía

Sea *U* un conjunto y *A* un subconjunto de *U*, esto es:

$$A \subset U$$

Sea ahora x un elemento de U. El hecho de que x pertenezca a A se simboliza como:

$$x \in A$$

Para indicar esta pertenencia se puede utilizar de la siguiente forma.

Definición 3.1
Función de pertenencia o función de membresía

$\mu_A : U \rightarrow [0, 1]$ es una función de pertenencia o función característica del conjunto A sí y solo si para toda x se cumple que:

$$\mu_A(x) = \begin{cases} 1 & Si\, x \in A \\ 0 & Si\, x \notin A \end{cases} \qquad (3.1)$$

Como puede observarse, la pertenencia o membresía del elemento x al conjunto A, es total o nula [0, 1]. Esta definición expresa el concepto clásico de que un objeto pertenece o no pertenece a un conjunto.

3.2.2 Conceptos de subconjuntos difusos

El problema con la lógica bivalente es que se vive en un mundo analógico y no digital. En el mundo real los procesos no permanecen en estos dos estados. El desarrollo de teorías analógicas de computación tales como la teoría difusa es representar con más precisión al mundo real. A continuación se dan los fundamentos básicos de la teoría de conjuntos difusos.

Definición 3.2
Subconjuntos Difusos

Si tomamos una colección de objetos denotados por $\{u\}$, el cual puede ser discreto o continuo. Donde U es llamado el universo de discurso y u representa el elemento genérico de U. En un conjunto difuso un elemento pertenece parcialmente a un conjunto. El grado de pertenencia de este elemento se define de la siguiente manera:

Un subconjunto difuso A de U es una colección de pares ordenados del elemento x y su grado de pertenencia $\mu_A(x)$:

$$A = \{(u, \mu_A(u)) \mid u \in U\} \qquad (3.2)$$

$$\mu_A(x): U \rightarrow [0, 1]$$

En seguida, se dan otras definiciones importantes sobre lógica difusa. Dichas operaciones se definirán en términos de funciones de pertenencia para facilitar así el manejo de los conceptos de la lógica difusa.

Definición 3.3
Soporte de un subconjunto difuso

El soporte ***S(A)*** de un subconjunto difuso ***A*** es el conjunto de todos los puntos *x* en U tales que, $\mu_A(x)$ es positiva. Es decir

$$S(A) = \{x \in U \mid \mu_A(x) > 0\}$$

Definición 3.4
Singularidad difusa

Subconjuntos difusos cuyo soporte es un punto único en U. Si A es una singularidad difusa cuyo soporte es x escribimos

$$A = \mu / x$$

Es decir, el conjunto ***A*** está formado por el elemento x, el cual tiene un grado de pertenencia μ en ***A***. En seguida se da un ejemplo para dejar claro definición de singularidad difusa.

Ejemplo: considere el universo de discurso dado por $U = \{0,1,2...,10\}$ un subconjunto difuso A_1, definido de la siguiente manera:

$$A_1 = 1.0/0 + 0.9/1 + 0{,}75/2 + 0.5/3 + 0.2/4 + 0.1/5$$

De igual manera el subconjunto difuso A_2 se puede expresar de la siguiente manera:

$$A_2 = 0.17/0 + 0.25/1 + 0.35/2 + 0.56/3 + 0.83/4 + 0.1/5 + 0.83/6 + 0.56/7 + 0.35/8 + 025/9 + 0.17/10$$

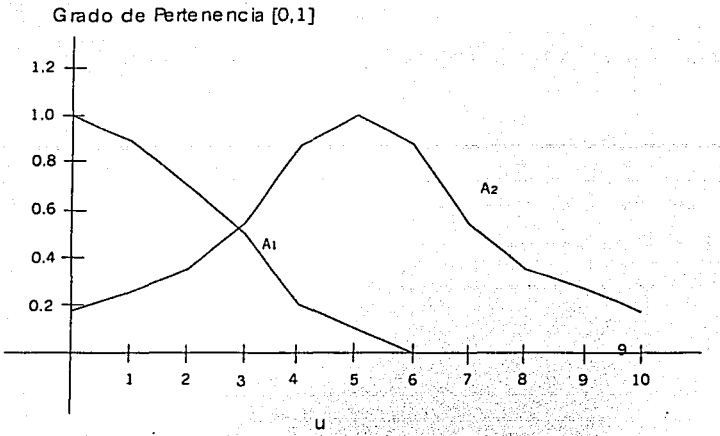


Figura 3.1 Definición de conjuntos difusos A_1 y A_2

Esta estructura matemática permite manejar problemas pobremente definidos, ya que le grado de pertenencia puede jerarquizarse [29].

De tal manera que el subconjunto difuso definido anteriormente puede verse como la unión de sus singularidades constituyentes. Por otro lado, si el conjunto universo es continuo podemos representar al subconjunto difuso como:

$$A = \int_U \mu_A(x)/x$$

De igual manera, si el conjunto universo es discreto y finito, tal como se muestra en la figura 3.1, se puede representar como:

$$A = \sum_{i=1}^{n} \mu_i / y_i$$

En la siguiente sección se definirán las operaciones principales entre subconjuntos difusos.

3.2.3 Operaciones básicas entre subconjuntos difuso

Sea ***A*** y ***B*** dos subconjuntos difusos definidos en el conjunto universo ***U***, con funciones de pertenencia μ_A y μ_B, como se muestra en la Figura 3.1.

Observemos el efecto en la función de pertenencia de la unión de los subconjuntos A y B.

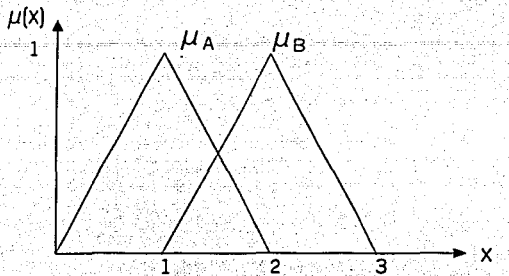


Figura 3.2 Función de pertenencia de dos subconjuntos A y B.

Entonces las operaciones de unión, intersección y complemento quedan definidas por medio de sus funciones de pertenencia como sigue:

Definición 3.5

La función de membresía $\mu_{A\cup B}$ de la unión de $A \cup B$ cuya función de pertenencia es:

$$\mu_{A\cup B} = \max\{\mu_A(u), \mu_B(u)\}$$

La Figura 3.2 muestra la función de pertenencia de la unión de los subconjuntos A y B.

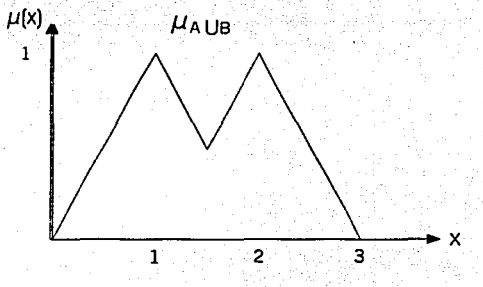


Figura 3.3 Función de pertenencia de la unión de los conjuntos A y B.

Definición 3.6

La intersección de *A* y *B* es un subconjunto A ∩ B cuya función de pertenencia se define como:

$$\mu_{A\cap B} = \min\{\mu_A(u), \mu_B(u)\}$$

Ahora adviértase la función de pertenencia de la intersección de dos conjuntos difusos.

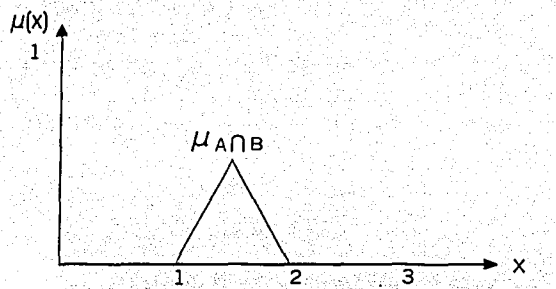


Figura 3.4 Función de pertenencia de la intersección de los subconjuntos A y B.

Definición 3.7

El complemento de *A* es un subconjunto difuso *–A* cuya función de pertenencia es:

$$\mu_{A-1}(u) = 1 - \mu_A(u)$$

Finalmente tenemos el complemento del subconjunto difuso A.

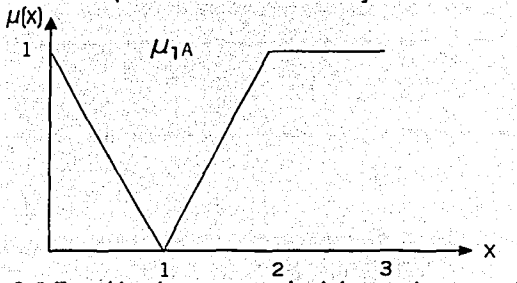


Figura 3.5 Función de pertenencia del complemento de A.

A continuación se dan tres definiciones adicionales que serán necesarias para comprender la sección siguiente.

Definición 3.8
Relación Difusa

Una relación difusa R de un subconjunto U a un subconjunto V es un subconjunto del producto cartesiano $U \times V$. R está caracterizada por una función de pertenencia bivariada $\mu_R(u,v)$ que es expresado como:

$$R = \int_{U \times V} \mu_R(u,v)/(u \cdot v) \qquad u \in U,\ v \in V \tag{3.3}$$

Es decir, el par (u,v) tiene un grado de pertenencia μ_R en la relación R o dicho de otra manera, el grado de la relación R del elemento u con el elemento es de μ_R[29].
Por ejemplo, sean X y Y dos conjuntos de personas:

$$X = \{Juan, Pedro\} \qquad Y = \{Luis, Paco\}$$

Y tenemos una relación de "amistad " entre los miembros de X y Y, lo cual puede expresarse como:

$$"amistad" = 0.8/(Juan, Luis) + 0.6/(Juan, Paco) + 0.2/(Pedro, Luis) + 0.9(Pedro, Paco)$$

Alternativamente, puede expresarse la relación anterior como una matriz de decisiones.

$$\begin{array}{c} \\ Juan \\ Pedro \end{array} \begin{array}{c} \begin{array}{cc} Pedro & Paco \end{array} \\ \begin{bmatrix} 0.8 & 0.6 \\ 0.2 & 0.9 \end{bmatrix} \end{array}$$

Definición 3.9
Composición supremo-estrella

Si R y S son relaciones difusas en $U \times V$ y en $V \times W$, respectivamente, la composición de R y S es una relación difusa denotada por $R \circ S$ y se define por :

$$R \circ S = \int_{U \times W} \sup(\mu_R(u,v) * \mu_s(v,w))/(u,w) \tag{3.4}$$

$$u \in U, v \in V, w \in W.$$

donde * denota cualquier clase de normas triangulares[1].

Definición 3.10
Regla composicional de inferencia supremo-estrella

Si R es una relación difusa en *UxV* y *x* es un conjunto difuso en *U*, entonces la regla composicional de inferencia establece que el conjunto difuso e inducido por x está dado por:

$$y = x \circ R \qquad (3.5)$$

donde $x \circ r$ es la composición supremo estrella de *x* y *R*.

3.3 Variables Lingüísticas

Un concepto básico que juega un papel importante en muchas de sus aplicaciones, en el área de control difuso, es la variable lingüística. Una variable lingüística, como su nombre lo indica, es una variable cuyos valores no son números sino palabras o sentencias en lenguaje natural.

Definición 3.11
Variables lingüística

Una variable lingüística está caracterizada por 5 elementos (*x, T(x), U, G, M*) en donde *x* es el nombre de la variable, *T(x)* es el conjunto de términos de *x*, es decir; el conjunto de nombres de los valores lingüísticos de *x*, siendo cada uno de estos valores un conjunto difuso definido en *U*, *G* es una regla sintáctica para generación de nombres de valor de *x*; y *M* es una regla semántica para asociar cada valor con su significado [7],[10]. Por ejemplo la variable *"EDAD"* es una variable lingüística y sus valores pueden ser *"MUY JOVEN"*,"*JOVEN*","*VIEJO*", Y "*MUY VIEJO*". Estas variables pueden ser representadas utilizando la variable base *U* [4], que representa la edad en años de vida (*U= [0,100]*) de la forma que se muestra en la figura 3.6. La presentación de la variable "*EDAD*" en la Figura 3.6 muestra dos aspectos muy importantes que hay que mencionar:

[1] Las normas triangulares se usan para definir conjunciones (ejemplos de normas triangulares son el mínimo –regla de Mamdani -). [7], [8].

1.- Las líneas punteadas entre las valores de *"EDAD "* indican que los rangos de esta variable con respecto a la variable base, están completamente abiertos para ser especificados según el criterio o razonamiento del diseñador.

2. - El porcentaje de probabilidad de que un valor *X* de la variable base pertenezca a un valor de edad, es representado por el número que se incluye en cada una de las líneas continuas. Por ejemplo, la probabilidad de que un valor de la variable base tal como 25 pertenezca al valor de edad *"MUY JOVEN"* es de 0.4, pero la probabilidad de que este mismo valor pertenezca al valor de edad "JOVEN" es de 0.9. Por lo tanto sea cual fuera el valor de la variable base; el controlador basándose en los rangos establecidos para cada valor de edad, es capaz de determinar a que valor de edad pertenece, para poder utilizarlo posteriormente.

Los valores de una variable lingüística pueden ser generadas de su termino primario (por ejemplo *"JOVEN"*) su antónimo (*"VIEJO"*), una colección de modificaciones (*"NO"*, *"MUY"*, *"MAS"* O *"MENOS"*, *"IGUAL"*, *"NO MUY"*, etc.) y los conectores *"Y"* y *"O"*. Por ejemplo un valor de edad puede ser *"NO MUY JOVEN"* y *"NO MUY VIEJO"*. Cada valor puede ser generado por un contexto-libre gramátical. Además, cada valor de una variable lingüística representa una posibilidad de distribución como se muestra en la Figura 3.7.

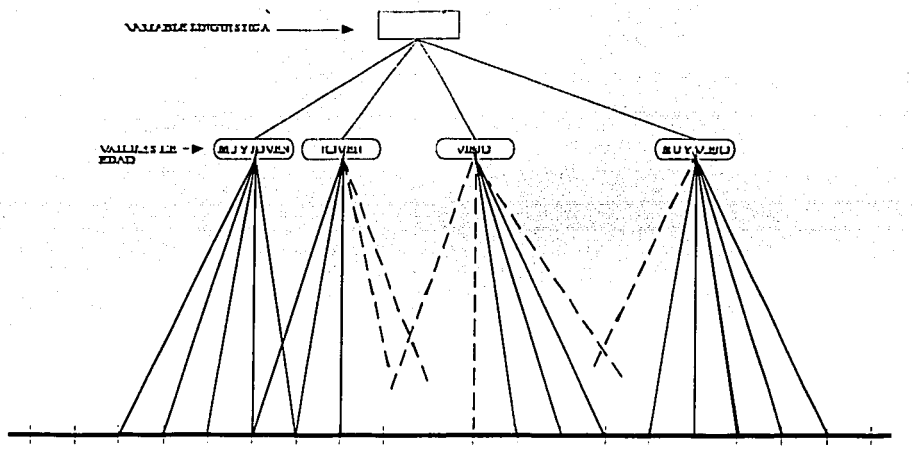


Figura 3.6 Representación de la Variable Lingüística EDAD.

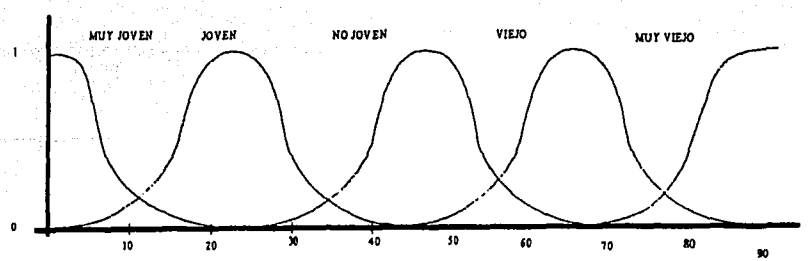


Figura 3.7 Valores Lingüísticos de Edad.

3.3.1 Relaciones simples entre variables difusas mediante enunciados condicionales

Dado un conjunto de variables numéricas, estas pueden relacionarse de diversas maneras, una es, por ejemplo, mediante una función. Tal como se muestra en la figura 3.8a).Sean *X* y *Y* dos universos de diferente discurso. Si un elemento de *x* esta contenido en *X* y corresponde a un elemento *y* contenido en *Y*. Otra manera es a través de una tabla, la cual puede ser descrita en palabras por medio de un conjunto de oraciones condicionales. Del mismo modo pueden relacionarse dos variables en lógica difusa pero en vez de ser variables numéricas serán variables lingüísticas [6].

Por ejemplo.

Si *x* es ***pequeña*** entonces *y* es ***muy grande***
Si *x* ***no es muy pequeña*** entonces *y* es ***muy grande***
Si *x* es ***no es pequeña*** *y* ***no es muy grande*** entonces *y* ***no es muy grande***

Se dice que la relación es simple por que puede describirse por medio de un conjunto de enunciados condicionales de la forma *"si x es A entonces y es B"*, donde *A* y *B* son las etiquetas de conjuntos difusos que representan los valores que toman las variables *x* y *y* respectivamente.

La figura 3.8 b) ilustra la geometría de la relación subconjuntos difusos (A_i,,B_i).

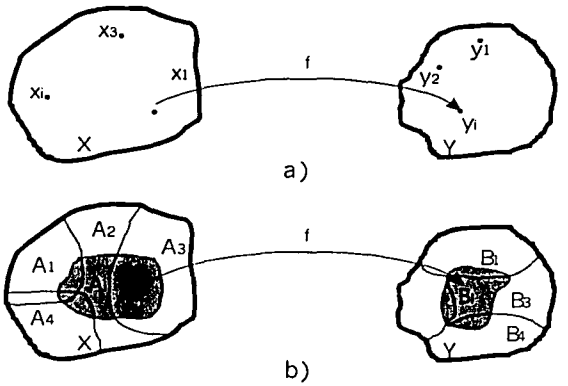


Figura 3.8 a) Relaciones numéricas b) Relaciones de conjuntos difusos.

3.4 Estructura básica de un controlador difuso [15][16].

La estructura básica de un controlador difuso está formada por 4 etapas fundamentales, las cuales se describen a continuación. Estas son:

1) Interfaz de fuzificación (fuzzification)[2].
2) Base de conocimientos.
3) Lógica de decisiones.
4) Interfaz de defuzificación (defuzzification).

La información del sistema a controlar se obtiene a través de la interfaz de fuzificación. Dicha información se procesa por medio de la base de conocimiento y de la lógica de decisiones, la cual es el corazón del controlador. Finalmente la salida de control pasa por la interfaz de defuzificación y es entonces cuando se alimenta al sistema bajo control. En la figura 3.9 se muestra el esquema de un controlador Difuso en diagrama de bloques:

A continuación se hace una breve descripción de cada una de las etapas que componen un control difuso.

[2] Las palabras fuzzification y defuzzification generalmente son traducidas como emborramiento y desborramiento en varios textos en español, pero en este trabajo se utilizará los términos fuzificación y defuzificación respectivamente.

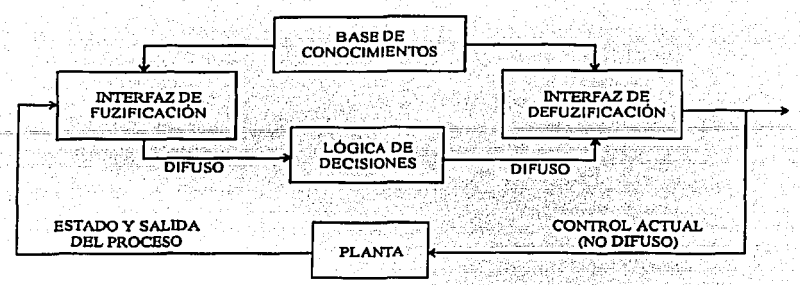


Figura 3.9 Estructura de un Controlador Difuso.

1) Interfaz de Fuzificación

La fuzificación es el proceso mediante el cual, se determinan ciertos valores para representar las entradas con grados de pertenencia, en cada conjunto difuso.

a) Medición de las variables de entrada.

b) Efectúa un mapeo a escala de los valores de entrada a sus correspondientes conjuntos universos (ver figura 3.6).

c) Realiza la fuzificación de los valores escalados, es decir, transforma las variables de entrada en variables lingüísticas, las cuales son valuadas en conjuntos difusos.

2) Base de conocimientos

La base de conocimientos que contiene la información de como debe controlarse el proceso, su estructura se deriva de la experiencia desarrollada por un operador humano *y/o* el conocimiento previo de ingeniería de control, por lo tanto; depende del proceso a controlar y del desempeño deseado. La base de conocimientos consiste de una base de datos y reglas que son expresadas en un lenguaje apropiado que permite describir el comportamiento del proceso usando términos lingüísticos (difusos).

a) La base de datos contiene las definiciones que se usan para especificar las reglas de control lingüístico y el tratamiento en términos de lógica difusa de los datos disponibles.

b) La base de reglas define los objetivos y las estrategias de control tomados de la experiencia, por medio de un conjunto de reglas de control lingüístico.

3) Lógica de decisiones.

Es la parte más importante del controlador difuso, ya que es la que simula el mecanismo de toma de decisiones utilizado por el cerebro humano, esto mediante inferencia difusa. Este mecanismo de inferencia está formado por reglas del siguiente tipo: Si un conjunto de condiciones son satisfechas, entonces un conjunto de consecuencias pueden ser inferidas. Estas reglas son expresadas en un lenguaje apropiado que permite describir el comportamiento del proceso usando términos lingüísticos. La forma genérica de expresar las reglas es como se muestra a continuación:

*Si **x** es **A** y **y** es **B** entonces **z** es **C***

4) Interfaz de defuzificación

El proceso de defuzificación es el proceso mediante el cual, el resultado obtenido en el proceso de inferencia, es transformado en un valor numérico. Realiza las siguientes funciones:

a) Un mapeo a escala que transfiere el rango de valores de las variables de salida a sus correspondientes conjuntos universos (universo de discurso).

b) Defuzificación, la cual transforma la salida de control difuso obtenida (variable lingüística) en un valor abrupto (no difuso) de control (variable numérica) aplicable al sistema.

3.5 Parámetros de diseño de control difuso [11],[12].

Los principales parámetros de diseño de un Controlador difuso son los siguientes:

1) Fuzificación

La operación básica consiste en convertir una cantidad precisa de entrada en un conjunto difuso. En la representación de los conjuntos se utilizan algunas funciones las cuales permiten describir las variables lingüísticas en términos de valores. El operador fuzificación (fuzificador) más simple, transforma un

dato numérico en uno singular difuso, es decir; un valor de entrada se convierte en un subconjunto difuso que tiene un grado de pertenencia entre uno y cero. Existen tres tipos de funciones que permiten describir las variables lingüísticas en términos de valores. Las funciones más comunes que puede generar un fuzificador son: triangular, trapezoidal, función S y la función $\prod$ [3]. La función de pertenencia más utilizada es la representada por un triángulo, donde su representación matemática en forma general es la siguiente:

$$A(x) = \begin{cases} \theta_1(x) & para \quad p_o \le x \le p_1 \\ \theta_2(x) & para \quad p_1 \le x \le p_2 \\ \cdot & \\ \cdot & \\ \cdot & \\ \theta n(x) & para \quad p_{n-1} \le x \le p_n \end{cases} \tag{3.6}$$

donde $\theta_i(x) \in [0,1]$ para $x \in [P_{i-1}, P_i]$, además $\theta_i(x) = \sum_{k=0}^{n} C_k^i x^k$ y los coeficientes C_k^i parámetros del polinomio $\theta_i(x)$.

Otra de las funciones más utilizadas es la función S, su representación matemática esta definida de la siguiente forma [6]:

$$S(x;\alpha,\beta,\gamma) = \begin{cases} 0 & para \quad x \le \alpha \\ 2\left(\dfrac{x-\alpha}{\gamma-\alpha}\right)^2 & para \quad \alpha \le x \le \beta \\ 1-2\left(\dfrac{x-\alpha}{\gamma-\alpha}\right)^2 & para \quad \beta \le x \le \gamma \\ 1 & para \quad x \ge \gamma \end{cases} \tag{3.7}$$

En esta definición α, β y γ son parámetros que pueden ser ajustados de acuerdo a la función de pertenencia requerida. El dibujo de la función S se muestra en al figura 3.10.

[3] Se muestra las funciones de pertenencia más comunes, aunque de hecho pueden definir un gran número de ellas.

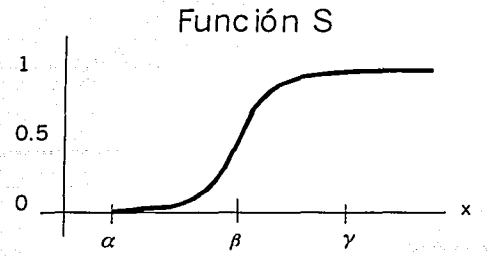


Figura 3.10 Función S

Otra función en la cual podemos obtener una curva similar, es llama la función $\prod$, definida de la siguiente manera [6].

$$\prod(x;\beta,\gamma) = \begin{cases} S(x,\gamma-\beta,\gamma-\frac{\beta}{2},\gamma) & para \quad x \le \gamma \\ 1-S(x;\gamma,\gamma+\frac{\beta}{2},\gamma+\beta) & para \quad x \ge \gamma \end{cases} \tag{3.8}$$

La función $\prod$ se muestra en la figura 3.11. La distancia β determina el ancho de banda, en los puntos $x = \gamma \pm \beta$ obtendremos una pertenencia igual a cero. Mientras que los puntos de x que cruza con el ancho de banda están dados en $x = \gamma \pm \frac{\beta}{2}$.

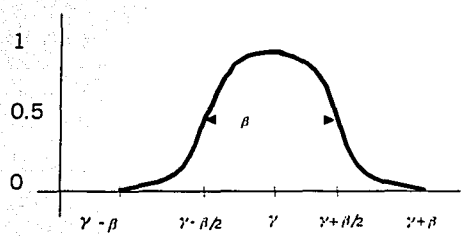


Figura 3.11 Función $\prod$

La forma de la función de pertenencia depende básicamente de la incertidumbre en las mediciones y de la capacidad de cálculo de control.

2) Base de Datos

Incluye la discretización y la normalización de los universos de entrada y salida, la definición de los subconjuntos difusos (que incluye la especificación de las funciones de pertenencia y la partición de los universos de entrada y salida) y el cumplimiento de la propiedad de completitud, la cual indica que el algoritmo debe ser capáz de inferir una acción de control correcta para todo estado del proceso.

a) Discretización. A la discretización de un universo de discurso se le llama cuantización. En efecto, la cuantización discretiza un universo en un cierto número de segmentos (niveles de cuantización). Cada segmento va etiquetado como un elemento genérico y por lo tanto forma un universo discreto. Un conjunto difuso se define entonces al asignar diversos valores (grados) de pertenencia a cada elemento cuantizado del nuevo universo discretizado. La elección de dichos niveles de cuantización influirá en qué tan fino será el controlador.

b) Partición de los universos. En general, una variable lingüística está asociada con un conjunto de términos. Una partición difusa determina cuantos términos existirán en el conjunto de términos. Lo anterior implica que la partición difusa de los universos está determinada, por el número de subconjuntos difusos en cada universo.

c) Funciones de Pertenencia. Pueden definirse numéricamente o utilizando una función. La definición numérica relaciona cada elemento del universo con un grado de pertenencia al subconjunto correspondiente. En contraste, también se puede definir el grado de pertenencia a un subconjunto difuso por medio de una función analítica [16].

2) Base de reglas

El objetivo es definir el comportamiento de las fuentes de control, esto implica escribir reglas de la forma CONDICIÓN-ACCIÓN, llamadas reglas difusas. Estas reglas son expresadas en un lenguaje apropiado que permite describir el comportamiento del proceso usando términos lingüísticos. Las reglas de control que forman el algoritmo difuso pueden definirse usando los siguientes criterios:

a) Selección de las variables. Las variables de entrada se seleccionan basandose en la experiencia y del conocimiento de ingeniería de control. Generalmente se usa como entrada la variable de error, el cambio (derivada) del error, etc.

b) Origen y obtención de la variable de control. Hay cuatro fuentes de obtención de las reglas de control:
   b.1) La experiencia y los conocimientos de Ingeniería de control.
   b.2) La observación de las acciones de control de operadores humanos.
   b.3) El modelo difuso del proceso.
   b.4) Dando al control la capacidad de construir sus propias reglas. En este caso se tendría un controlador auto-organizado.

c) Tipos de reglas de algoritmos difusos. Las reglas usadas en estos algoritmos, se expresan en forma de enunciados condicionales de la forma:

R_1 : *Si* $\boldsymbol{x}$ *es* $\boldsymbol{A_1}$,..., *y* $\boldsymbol{y}$ *es* $\boldsymbol{B_1}$ *entonces* $\boldsymbol{z}$ *es* $\boldsymbol{C_1}$
R_2 : *Si* $\boldsymbol{x}$ *es* $\boldsymbol{A_2}$,..., *y* $\boldsymbol{y}$ *es* $\boldsymbol{B_2}$ *entonces* $\boldsymbol{z}$ *es* $\boldsymbol{C_2}$
.....
.....
R_n : *Si* $\boldsymbol{x}$ *es* $\boldsymbol{A_l}$,..., *y* $\boldsymbol{y}$ *es* $\boldsymbol{B_l}$ *entonces* $\boldsymbol{z}$ *es* $\boldsymbol{C_l}$

Donde $\boldsymbol{x}$,...,$\boldsymbol{y}$ y $\boldsymbol{z}$ son variables lingüísticas que representan a las variables de entrada del proceso y las variables de control respectivamente ; $\boldsymbol{A_l}$,...,$\boldsymbol{B_l}$ y $\boldsymbol{C_l}$ son los valores de las variables lingüísticas $\boldsymbol{x}$,...,$\boldsymbol{y}$, y $\boldsymbol{z}$ en los universos de discurso $\boldsymbol{U}$, ..., $\boldsymbol{V}$ y $\boldsymbol{W}$.

4) Lógica de decisiones

Este punto se refiere, a los mecanismos de inferencia a utilizar, se trata de simular la toma de decisiones como los haría un ser humano de acuerdo a la definición de la implicación difusa.

De acuerdo a los diferentes procedimientos de inferencia se puede tener el método *max-min* de inferencia, *product-maximo* y *suma–producto*. Sus gráficos se muestran en las Figuras 3.9, 310 y 311. Dichas decisiones se fundamentan principalmente en dos conceptos:

- Relaciones Difusas.(ver definición 3.6)
- Regla composicional de inferencias. (ver definición 3.7 y 3.8)

Es decir las reglas utilizadas en el algoritmo de control se pueden representar por medio de relaciones difusas. Entonces, usando la regla composicional de inferencia se deduce la acción de control correcta para la situación presentada. Los gráficos para los procedimientos de inferencia se presentan a continuación.

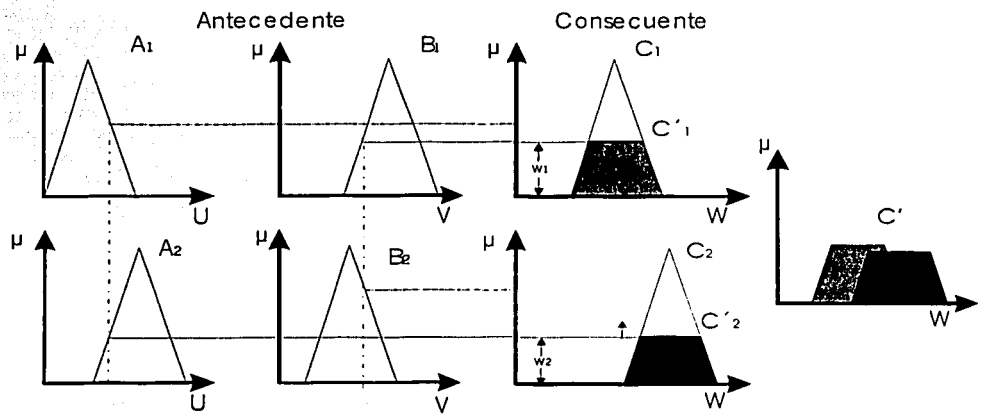


Figura 3.12 Método gráfico *max-min* de inferencia.

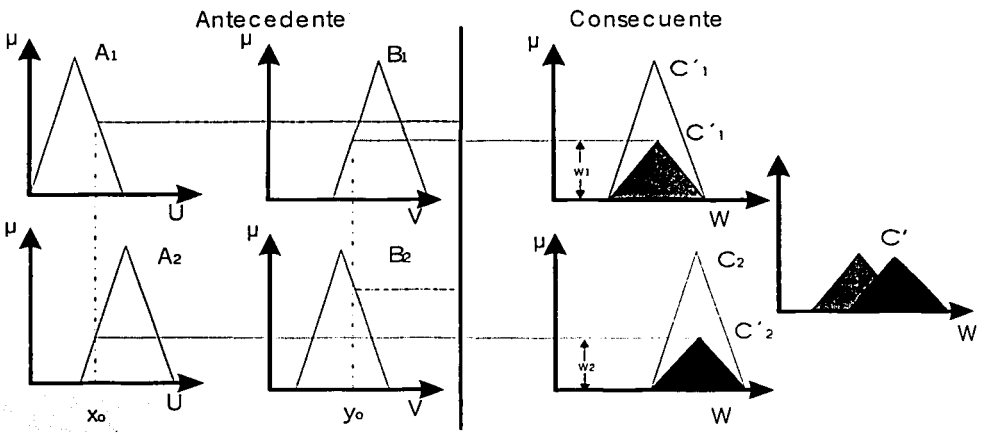


Figura 3.13 Método gráfico *max-product* de inferencia.

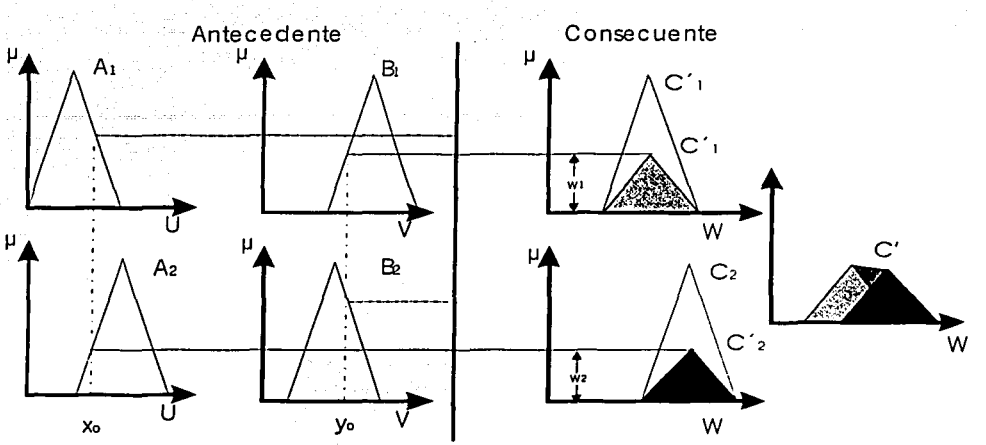


Figura 3.14 Método Gráfico *sum-product* de inferencia

En la figura 3.15 se muestra la arquitectura básica de un control difuso de una entrada y una salida (SISO). El sistema mapea un conjunto difuso de entrada A (antecedente) con su correspondiente conjunto difuso B (consecuente), las reglas activadas corresponden un grado de pertenencias determinado por los conjuntos B_1, B_2 ....Bm para cada valor de entrada A_1,A_2.....A_m. La inferencia difusa calcula los conjuntos de salida B'_j resultantes, donde w_j representa el grado de activación de la regla [6].

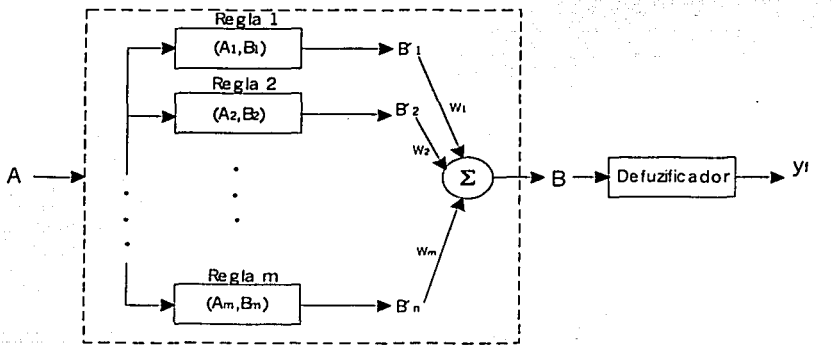


Figura 3.15 Arquitectura de un controlador difuso.

5) Estrategias de defuzificación

Existen varios métodos par efectuar el proceso de defuzificación, entre ellos, los más utilizados son[4]: a) Centro de gravedad o centroide, b)Criterio del máximo y c) Promedio de influencias.

a) Método del centro de gravedad o centroide

El mejor método disponible para realizar el proceso de defuzificación es el centro de gravedad o centroide, ya que éste considera la contribución de cada regla disparada durante el proceso de evaluación de reglas, matemáticamente el método del centroide se define por:

$$Z_O = \frac{\int_z \mu_c(z) z dz}{\int_z \mu_c(z) dz} \tag{3.9}$$

Donde $\int_z \mu_c(z)dz$ es el área de la función de pertenencia de la salida de control, Zo es el centro de gravedad de la función de membresía $\mu_c(z)$, en otra palabras el punto central sobre el eje x para cada función de pertenencia.

b) Método del máximo

Este método selecciona como salida la conclusión (conjunto difuso) de aquella regla que tenga el máximo grado de activación. De este conjunto, el punto en w que tiene el máximo valor de pertenencia al mismo, es la salida z_o, por ejemplo, si durante el proceso de evaluación de reglas se disparan 3 reglas r_1, r_2 y r_3, obteniéndose para cada uno los grados de influencia w_1= 0.75, w_2=0.40, y w_3=0.20 respectivamente; entonces el conjunto de salida es el dado por el consecuente de la regla r_1, ya que este posee el máximo grado de activación con respecto a las otras dos entradas. La Zo es, entonces el punto en w que tenga el máximo grado de pertenencia en dicho conjunto. Matemáticamente esto se expresa como :

$$Z_O = \max_w \mu_c(z) \tag{3.10}$$

[4] Se muestran los métodos más comunes, haciendo notar que existen algunos otros que se muestran en [13].

El método del máximo es el más simple, pero también es el menos utilizado, debido a que sólo da como resultado el punto en el cual la distribución de posibilidades de la acción de control alcanza el valor máximo.

c) Promedio de Influencias

Esta técnica genera una señal de control que representa el valor medio de todas las acciones de control locales dadas por aquellas reglas disparadas durante el proceso de evaluación de reglas. Su expresión matemática es la siguiente:

$$Z_o = \sum_{i=1}^{n} \frac{w_i}{n} \qquad (3.11)$$

donde w_i es el grado de activación asignada a cada regla , y n es el número de reglas activado ($w_i > 0$).

3.6 Modelos de sistema de control difuso

Tres son los modelos de control difuso utilizados en la actualidad, estos son:

1) Modelo Mamdani.
2) Modelo Sugeno-Takagi.
3) Modelo Tsukamoto.

A continuación se hace una breve descripción de cada uno de estos modelos.

3.6.1 Modelo Mamdani.

Este tipo de control se realiza mediante la combinación de un conjunto de reglas lingüísticas de la forma "si (antecedente), entonces (consecuente)" obtenidas de las experiencias de operadores humanos. La característica de este modelo es que tanto en la parte del antecedente como en la parte del consecuente se definen las funciones de membresía comúnmente utilizadas.
La forma en que se lleva a cabo el razonamiento difuso en este tipo de controladores, es aplicando la regla composicional *max-min*, tal como se muestra en la figura 3.9. La salida z_o se obtiene aplicando el método del centroide, pero cualquier otro método puede aplicarse.

A la propuesta original de Mamdani se le han hecho modificaciones, por ejemplo en lugar de utilizar la composición *max-min* (ver figura 3.9) se puede utilizar la composición *product-maximo* (ver figura 3.14). Otra modificación fue realizada por Bart Kosko, quien utiliza la regla composicional *suma-producto* (ver figura 3.14) y se le da el nombre de sistema difuso aditivo.

3.6.2 Modelo Sugeno,Takagi

Este modelo fué propuesto por Sugeno, Takagi y Kang en un esfuerzo por modelar un sistema capáz de generar, de manera sistemática, reglas difusas a partir de un conjunto dado de entradas-salidas. Una regla típica de este modelo es:

"si x es A y y es B, entonces z=f(x,y)"

donde *A* y *B* son conjuntos difusos en el antecedente; *z=f(x,y)* es una función en el consecuente. De manera general *f(x)* se define como un polinomio de las variables de entrada *x* y *y*, pero se puede emplear cualquier función que describa de manera apropiada la salida del sistema dentro de la región difusa especificada por el antecedente de cada regla. Cuando *f(x,y)* es un polinomio de primer orden el sistema de control se denomina modelo Sugeno de primer orden, el cual corresponde a la propuesta original descrita por Sugeno. Si *f(x,y)* se define como una constante, entonces el sistema difuso se conoce como modelo Sugeno de orden cero.

En la Figura 3.12 se muestra el razonamiento difuso para un control difuso de primer orden. Dado que cada regla tiene una salida parcial, la salida total del sistema se obtiene mediante un promedio ponderado de la conclusión aportada por cada regla disparada, lo cual implica un menor consumo de tiempo de cálculo.

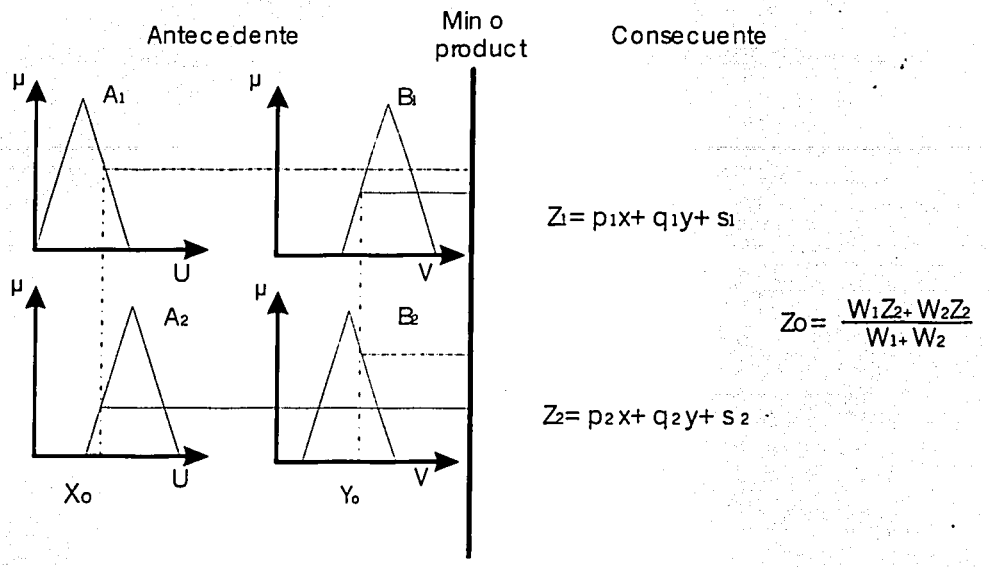


Figura 3.16 Razonamiento difuso para un control difuso Sugeno de primer orden.

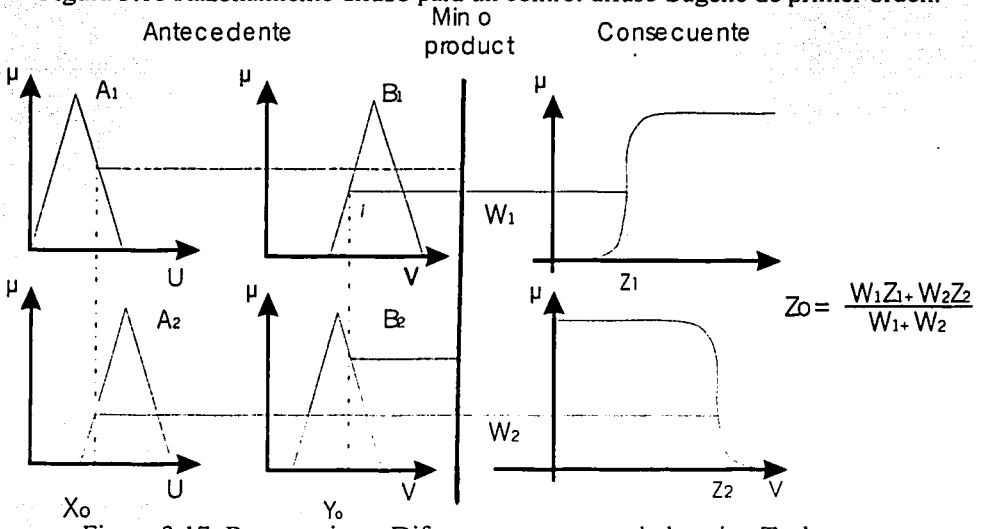


Figura 3.17 Razonamiento Difuso para un controlador tipo Tsukamoto.

3.6.3 Modelo Tsukamoto

En el modelo Tsukamoto el consecuente de cada a regla se representa con un conjunto difuso cuya función de membresía es una función autónoma, tal como se muestra en la figura 3.17. Como resultado de lo anterior, la salida inferida para cada regla está dada por un valor inducido por el grado de activación de dicha regla, La salida total del sistema es el promedio ponderado de las salidas parciales dadas por cada regla. La figura 3.17 muestra el razonamiento en este modelo, para el caso de dos entradas y dos reglas.

3.7 Conclusión

La habilidad del pensamiento humano es que puede resumir, a partir de grandes volúmenes de información, lo que considera más importante usando el mínimo necesario para realizar de manera adecuada las tareas que se propone; así esta habilidad de manejar la información bajo una etiqueta y la consecuente habilidad para resumir constituye las características más importantes de la mente humana.

El modo de pensar del humano es difuso, por ejemplo, para realizar alguna actividad cotidiana como la de tomar un objeto de un punto inicial y llevarla a un punto final, el cerebro piensa de la siguiente manera:

El llevar la mano ***cerca*** del objeto.
Tomar el objeto con ***suficiente*** fuerza.
Llevar el objeto ***cerca*** del punto final.

De la misma manera se lleva acabo el control en procesos complejos o pobremente definidos en donde el experto humano es de vital importancia. La manera de capturar la experiencia de una persona y traducirla en términos de lógica matemática es la lógica difusa, y la manera de manipular a nuestro antojo sistemas complejos o pobremente definidos es por medio del control difuso.

Capítulo 4

Planeación de movimientos

Un robot es un dispositivo que interactúa con el mundo físico. La secuencia de acciones necesarias para concluir una tarea es determinada por la naturaleza de la misma. Este problema es enfrentado con técnicas de planeación de movimientos y navegación. Una organización lógica típica para un robot móvil se muestra en la figura 4.1. La misión de cada módulo se lista a continuación.

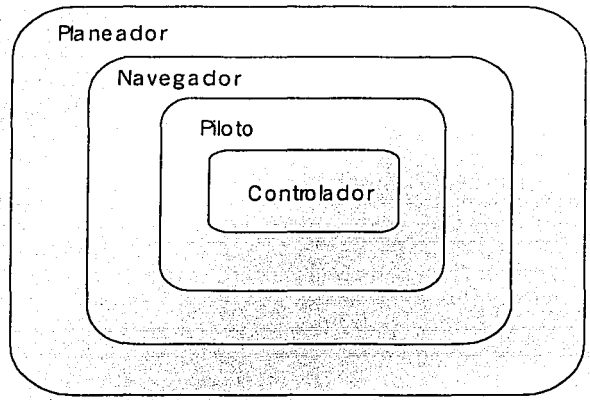


Figura 4.1 Organización de un robot móvil

Planeador. Determina una ruta óptima global, de acuerdo a algún criterio de optimización, utilizando conocimientos a priori del entorno del robot y de las restricciones a las que está sujeto.

Navegador. Divide la ruta seleccionada por el planeador en segmentos independientes y encuentra un movimiento continuo; proporciona al piloto las distancias, velocidades y ángulos de giro que tendrá que realizar.

Piloto. Sigue la ruta definida por el navegador y evita obstáculos detectados con los sensores del robot si lo requiere la aplicación.

Controlador: Ejecuta las ordenes enviadas por el piloto para que los motores efectúen los movimientos deseados.

El módulo del planeador, como se menciono anteriormente, su tarea es determina una ruta optima, previo conocimiento del entorno del robot, para nuestra aplicación se considera que es suficiente considerar las dimensiones del área donde el robot se va desplazar; además de que no se define un máximo tiempo para cada movimiento.

4.1 Planeación

Una vez que está definido el espacio donde el robot podrá desplazarse libremente para realizar sus movimientos, el siguiente paso es definir el problema de planeación.

Consideremos un sólido **S** aislado tal como se muestra en la figura 4.2, en un punto cualquiera se le coloca una referencia ortonormal R. Puede considerarse que el número de grados de libertad del sólido, es el número de desplazamientos independientes que se pueden hacer respecto a la referencia R. Una forma de contar estos desplazamientos consiste en decir que el sólido puede efectuar.

3 Translaciones T1, T2 ,T3 a lo largo de las ejes OX, OY y OZ.
3 Rotaciones R1, R2 y R3 alrededor de OX, OY y OZ.

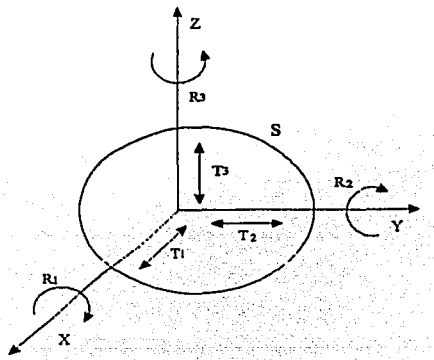


Figura 4.2 Grados de Libertar de un Sólido S

Un sólido por lo tanto tiene 6 grados de libertad .Cuando se establece una unión entre sólidos, cada uno de ellos pierde un grado de libertad respecto al otro. Si hablamos de un robot compuesto por eslabones articulados, estas articulaciones le impiden realizar con libertad algunos movimientos por lo tanto decimos que tiene restricciones. Para el caso de un robot móvil, auque se puede desplazar y girar, no se puede mover hacia los lados, y los giros que realiza están limitados por el rango de su volante.

Las restricciones que afectan a un robot articulado se pueden representar mediante ecuaciones que relacionan los parámetros de configuración y que se pueden usar para eliminar algunos parámetros y reducir la dimensión del espacio de configuraciones. A estas ecuaciones se les llaman *restricciones holonómicas*. Las restricciones que afectan a los movimientos de un robot tipo coche se pueden convertir en una ecuación y una desigualdad que involucran los parámetros de velocidad del mismo y son llamadas *restricciones no holonómicas*.

4.2 Navegación

Como se menciona anteriormente el navegador se encarga de realizar los movimientos de manera continua que lleve al robot de una posición a otra a través de una ruta geométrica localmente definida.

4.2.1 Modelo Matemático

En la parte de Navegación lo que nos interesa es definir la posición y orientación del robot en un punto dado en un espacio real.

La forma más simple de llevar al robot de la configuración q_i a la configuración q_{i+1} es dividir la ruta en una secuencia de líneas rectas, como se observa en la Figura 4.3.

El navegador recibe esa secuencia del planeador y genera configuraciones que el robot puede alcanzar con movimientos simples es decir, giros y desplazamientos. Cada vez que el robot realiza un giro y un desplazamiento, el robot pasa de la configuración q_i a la configuración q_{i+1}. La posición de q_{i+1} es el siguiente punto a alcanzar moviendose en línea recta, y su orientación es la orientación necesaria para que el robot apunte hacia el siguiente punto.

En resumen, el robot primero se orienta hacia el siguiente punto y después avanza para alcanzarlo.

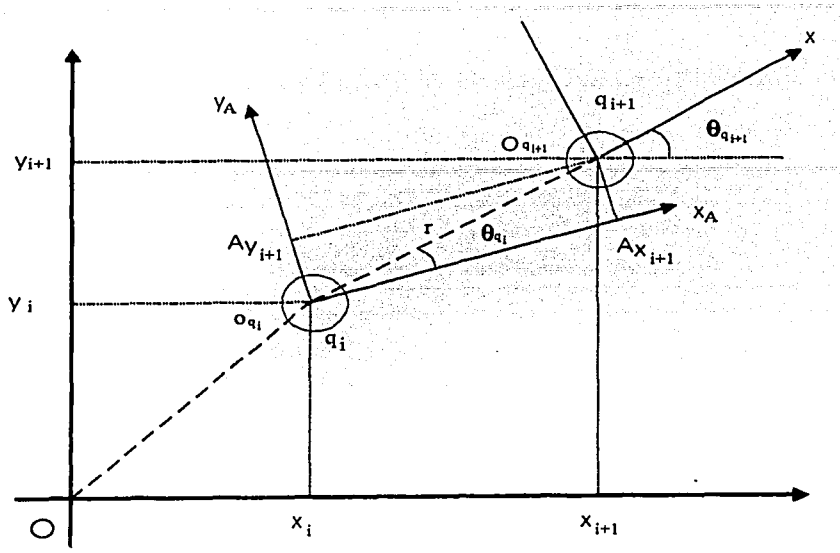


Figura 4.3 Sistema de referencia

Si suponemos que el robot solo se puede mover a una la dirección del eje x y que puede girar sobre su eje para orientarse en cualquier dirección, solo hace falta encontrar las ecuaciones que rigen sus giros y desplazamientos.

La Figura 4.3 muestra las configuraciones q_i y q_{i+1}. Se puede observar que el robot pase de la configuración q_i a q_{i+1}. Primero debe orientarse de manera que su eje x_A apunte hacia el origen de q_{i+1}, es decir, debe adquirir la orientación $\theta_{q_{i+1}}$ y después moverse sobre el segmento de recta $\overline{O_{q_i} O_{q_{i+1}}}$.

El concepto de transformación nos ayuda a encontrar los movimientos que debe realizar el robot [22], localizando la posición y orientación del robot en la configuración q_{i+1} con respecto a la configuración q_i. En la Figura 4.3 se observa que.

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \overline{O_{q_i}} + \overline{O_{q_{i+1}}} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} r\cos\left({}^{A}\theta_{i+1} + \theta_i\right) \\ r\,\mathrm{sen}\left({}^{A}\theta_{i+1} + \theta_i\right) \end{pmatrix}$$

en donde

$${}^{A}\theta_{i+1} = \theta_{i+1} - \theta_i \qquad (4.1)$$

Aplicando identidades trigonométricas queda:

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} x_i + r\cos^{A}\theta_{i+1}\cos\theta_i - rsen^{A}\theta_{i+1}sen\theta_i \\ y_i + r\cos^{A}\theta_{i+1}sen\theta_i - rsen^{A}\theta_{i+1}\cos\theta_i \end{pmatrix} \tag{4.2}$$

Por otro lado:

$$\begin{pmatrix} {}^{A}x_{i+1} \\ {}^{A}y_{i+1} \end{pmatrix} = \begin{pmatrix} r\cos^{A}\theta_{i+1i} \\ rsen^{A}\theta_{ii+1} \end{pmatrix} \tag{4.3}$$

$$\begin{pmatrix} {}^{A}x_{i+1} \\ {}^{A}y_{i+1} \end{pmatrix} = \begin{pmatrix} x_i + {}^{A}x_{i+1}\cos\theta_i - {}^{A}y_{i+1}sen\theta_i \\ y_i + {}^{A}x_{i+1}\sin\theta_i + {}^{A}y_{i+1}\cos\theta_i \end{pmatrix} \tag{4.4}$$

de lo cuál:

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} \cos\theta_i & -sen\theta_i \\ sen\theta_i & \cos\theta_i \end{pmatrix} \begin{pmatrix} {}^{A}x_{i+1} \\ {}^{A}y_{i+1} \end{pmatrix} + \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{4.5}$$

Agregando ésta ecuación de manera que incluya el tercer parámetro de la configuración que es la orientación y se obtiene de la ecuación (4.3):

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \\ \theta_{i+1} \end{pmatrix} = \begin{pmatrix} \cos\theta_i & -sen\theta_i & 0 \\ sen\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} {}^{A}x_{i+1} \\ {}^{A}y_{i+1} \\ {}^{A}\theta_{i+1} \end{pmatrix} + \begin{pmatrix} x_i \\ y_i \\ \theta_i \end{pmatrix} \tag{4.6}$$

que también se puede expresar como

$$q_{i+1} =_i R^{A}q_{i+1} + q_i \tag{4.7}$$

en donde ${}_iR$ es la matriz de senos y cosenos que representa la rotación de la configuración q_i . Esta matriz es ortonormal, por lo que

$$R^{-1} = R^T$$

despejando se obtiene

$${}^{A}q_{i+1} = {}_i R^{-1}(q_{i+1} - q_i) = {}_i R^T(q_{i+1} - q_i) \tag{4.8}$$

Conceptualmente

$${}^{A}q_{i+1} = \begin{pmatrix} {}^{A}x_{i+1} \\ {}^{A}y_{i+1} \\ {}^{A}\theta_{i+1} \end{pmatrix}$$

es el avance del robot sobre el nuevo eje de coordenadas en *x* e *y*.

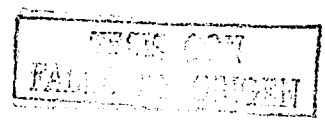
$${}^{A}q_{i+1} = \begin{pmatrix} {}^{A}x_{i+1} \\ {}^{A}y_{i+1} \\ {}^{A}\theta_{i+1} \end{pmatrix} = \begin{pmatrix} \cos\theta_i & sen\theta_i & 0 \\ -sen\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{i+1} - x_i \\ y_{i+1} - y_i \\ \theta_{i+1} - \theta_i \end{pmatrix} \tag{4.9}$$

De aquí se concluye que el robot primero tiene que girar ${}^{A}\theta_{i+1}$ unidades, y después desplazarse ${}^{A}x_{i+1}$ unidades (ya que solo se mueve en la dirección del eje X^+).

$${}^{A}\theta_{i+1} = \theta_{i+1} - \theta_i \tag{4.10}$$

$${}^{A}x_{i+1} = (x_{i+1} - x_i)\cos\theta_i + (y_{i+1} - y_i)sen\theta_i \tag{4.11}$$

Para ilustrar el uso de las expresiones (4.10) y (4.11) se plantea el problema de encontrar los movimientos necesarios para seguir la secuencia mostrada en la Figura 4.12.



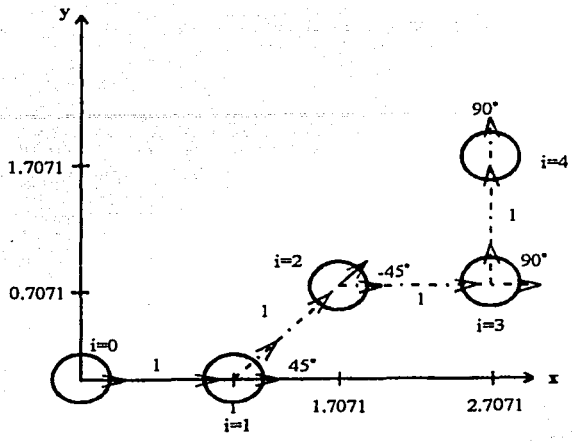


Figura 4.3 Ejemplo de Navegación

En *i= 0* el robot se encuentra en el origen del sistema (x_0 =0, y_o = 0) con un ángulo de 0^0 con respecto al eje *x* ($\theta_0 = 0$). Para llegar a la configuración en la que x_1 = 1, y_1 = 0 y $\theta_1 = 0$ se utilizan las ecuaciones (4.9) y (4.10) para encontrar el desplazamiento y giro necesario.

$$\Delta\theta_1 = 0^0 - 0^0 = 0$$

$$\Delta x_1 = (1-0)\cos 0^0 + (0-0)sen0^0 = 1$$

$$\Delta y_1 = (1-0)sen0^0 + (0-0)\cos 0^0 = 0$$

Ahora *i=1* y la meta es llegar a la configuración $x_2 = 1 + \frac{1}{\sqrt{2}}$, $y_2 = \frac{1}{\sqrt{2}}$ y $\theta_2 = 45$, se vuelven a aplicar las formulas:

$$\Delta\theta_2 = 45^0 - 0^0 = 45$$

$$\Delta x_2 = (1 + \frac{1}{\sqrt{2}} - 1)\cos 45^0 + (\frac{1}{\sqrt{2}} - 0)sen45^0 = 1$$

$$^{A}y_2 = -(1+\frac{1}{\sqrt{2}}-1)sen45^0 + (\frac{1}{\sqrt{2}}-0)\cos 45^0 = 0$$

Para $i=2$, $x_3 = 2+\frac{1}{\sqrt{2}}$, $y_3 = \frac{1}{\sqrt{2}}$ y $\theta_3 = 0^0$, los movimientos son:

$$^{A}\theta_3 = 0^0 - 45^0 = -45^0$$

$$^{A}x_3 = (2+\frac{1}{\sqrt{2}}-1-\frac{1}{\sqrt{2}})\cos 0^0 + (\frac{1}{\sqrt{2}}-\frac{1}{\sqrt{2}})sen0^0 = 1$$

$$^{A}y_3 = (2+\frac{1}{\sqrt{2}}-1-\frac{1}{\sqrt{2}})sen0^0 + (\frac{1}{\sqrt{2}}-\frac{1}{\sqrt{2}})\cos 45^0 = 0$$

Para $i=3$, $x_4 = 2+\frac{1}{\sqrt{2}}$, $y_4 = 1+\frac{1}{\sqrt{2}}$ y $\theta_4 = 90^0$, entonces

$$^{A}\theta_4 = 90^0 - 0^0 = 90$$

$$^{A}x_4 = (2+\frac{1}{\sqrt{2}}-2-\frac{1}{\sqrt{2}})\cos 90^0 + (1+\frac{1}{\sqrt{2}}-\frac{1}{\sqrt{2}})sin90^0 = 1$$

$$^{A}y_4 = -(2+\frac{1}{\sqrt{2}}-2-\frac{1}{\sqrt{2}})sen90^0 + (1+\frac{1}{\sqrt{2}}-\frac{1}{\sqrt{2}})\cos 90^0 = 0$$

Cuando $i=4$, $x_5 = 1+\frac{1}{\sqrt{2}}$, $y_5 = 1+\frac{1}{\sqrt{2}}$ y, $\theta_5 = 180^o$, se tiene que :

$$\theta_5 = 180^o - 90^o = 90^o$$

$$^{4}x_5 = (1+\frac{1}{\sqrt{2}}-2-\frac{1}{\sqrt{2}})\cos 180^0 + (1+\frac{1}{\sqrt{2}}-1-\frac{1}{\sqrt{2}})sen180^0 = 1$$

$$^{4}y_5 = -(1+\frac{1}{\sqrt{2}}-2-\frac{1}{\sqrt{2}})sen180^0 + (1+\frac{1}{\sqrt{2}}-1-\frac{1}{\sqrt{2}})sen180^0 = 0$$

La llegada a la meta establecida en la Figura 4.12. En la misma figura se observa junto a las líneas que sigue el robot la distancia recorrida, y el ángulo de giro se puede ver junto al robot, donde éste cambia de dirección.

i	x_i	y_i	θ_i	x_{i+1}	y_{i+1}	θ_{i+1}	$^{4}x_{i+1}$	$^{4}y_{i+1}$
0	0	0	0	1	0	0	1	0
1	1	0	0	1.707	0.707	45	1	0
2	1.707	0.707	45	2.707	0.707	0	1	0
3	2.707	0.707	0	2.707	1.707	90	1	0
4	2.707	1.707	90	0.707	1.707	180	0	0
5	1.707	1.707	180	*	*	*	*	*

Tabla 4.1 Movimientos generados por el navegador

En la Tabla 4.1 se muestran los datos utilizados y obtenidos en cada paso. Se pude ver que el valor de $^{4}y_{i+1}$ siempre es 0 y que la distancia entre la posición del robot y el punto al que requiere desplazarse en el tiempo *i+1* coincide con el valor de $^{4}x_{i+1}$, debido a que el robot siempre se mueve en la dirección de su eje *x*.

La ecuación (4.13) y (4.14) se puede aplicar en técnicas de navegación que cuenta con toda la información de entorno y también en las que no tiene, pero cuando no se cuenta con información completa suelen ser insuficientes, así que es necesario realizar la navegación utilizando sensores y auxiliando del piloto.

4.3 Pilotaje

El piloto es el encargado de seguir las trayectorias que le indica el navegador, hasta alcanzar la posición deseada, estos movimientos pueden estar ligados a la variable tiempo si así se requiere.

4.4 Consideraciones

Difícilmente un robot se desplaza en una superficie lisa perfectamente plana, por el contrario sus movimientos se realizan en la mayoría de las veces en superficies rugosas. Por lo tanto para tener información sobre la posición actual del robot después de cada desplazamiento, la mayoría de las veces de confía en encodificadores que informan a un sistema el número de veces que ha girado el eje del motor. Esta información junto con las dimensiones físicas de las ruedas del robot, son utilizadas para determinar la distancia que el robot se ha desplazado, y por consiguiente determinar el punto actual del mismo.

Capítulo 5

Implementación de algoritmos difusos para el control de robots móviles

En este capítulo se muestran las dos implementaciones realizadas; basadas en lógica difusa; la primera aplicación consiste en realizar un algoritmo programado en el lenguaje C basado en el microcontrolador de motorota 68HC11E9 y el segundo algoritmo desarrollado en el lenguaje para sistemas experto CLIPS[1].

5.1 Descripción del robot móvil utilizado

El robot móvil (prototipo) cuenta con una tarjeta de evaluación de Motorola basada en el Microcontrolador 68HC11E9, etapa de sensores fotoresistivos que servirán como entrada para nuestra primera aplicación y una etapa de control de motores. Una etapa de control de motores, que controlará 2 motores de corriente directa. En la Figura 5.1 se muestra el diagrama de bloques de la construcción general del robot móvil.

5.1.1 Etapa de sensores fotoresistivos

Los sensores utilizados en esta etapa son de tipo resistivo, este tipo de sensores varian su resistencia de acuerdo a la intensidad de luz que incide sobre ellos, el objetivo que se persigue con esta etapa es que el robot desarrolle determinadas acciones de acuerdo a la lectura presente en cada uno de ellos, el muestreo de estas lecturas se realiza a través del convertidor A/D del microcontrolador.

En la Figura 5.2 se muestra el circuito electrónico general del robot móvil utilizado para la primera aplicación, el cual se explicará más adelante.

[1] Para más información sobre este lenguaje de programación se puede consultar el apéndice A.

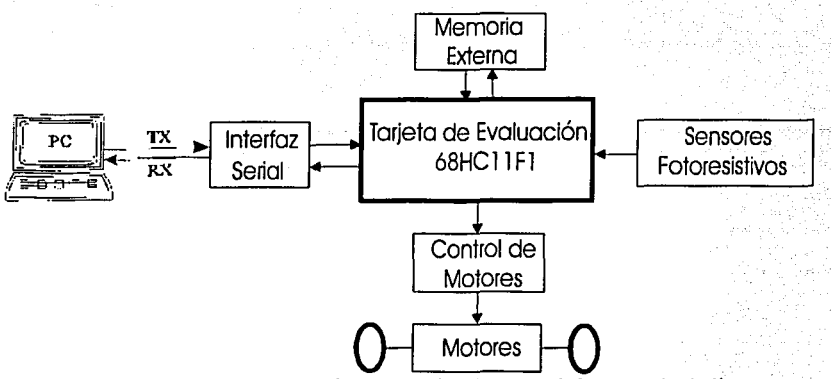


Figura 5.1 Diagrama de bloques del robot Móvil

5.1.2 Muestreo de señales de entrada

El microcontrolador 68HC11E9 como se ha mencionado anteriormente, contiene un convertidor A/D, dos de los canales del convertidos son conectados a un arreglo en serie de un resistor fijo y una fotoresistencia, la señal analógica obtenida está comprendida entre 0 y 5.0 volts. Estos canales se muestrean para adquirir las señales de los SENSOR_IZQUIERDO y SENSOR _DERECHO, como se muestra en la figura 5.2

5.1.3 Etapa de control de Motores

Para la etapa del control de motores se utilizó el Circuito Integrado (CI) LM293, tiene la característica principal de que acepta niveles estándares de lógica TTL y DTL, podemos controlar dos motores de CD de 5 volts de manera independiente.

En LM 293 se requieren cuando menos cuatro señales de control, las cuales serán enviadas por el microcontrolador. Para el control de un motor son necesarias dos señales, una que indicará la dirección de giro del motor (derecha o izquierda), y la otra señal que nos indicará la velocidad a la que se desea el giro.

Las características del LM293, así como sus parámetros principales se muestran en el apéndice B. El circuito de control de motores se muestra en la Figura 5.2.

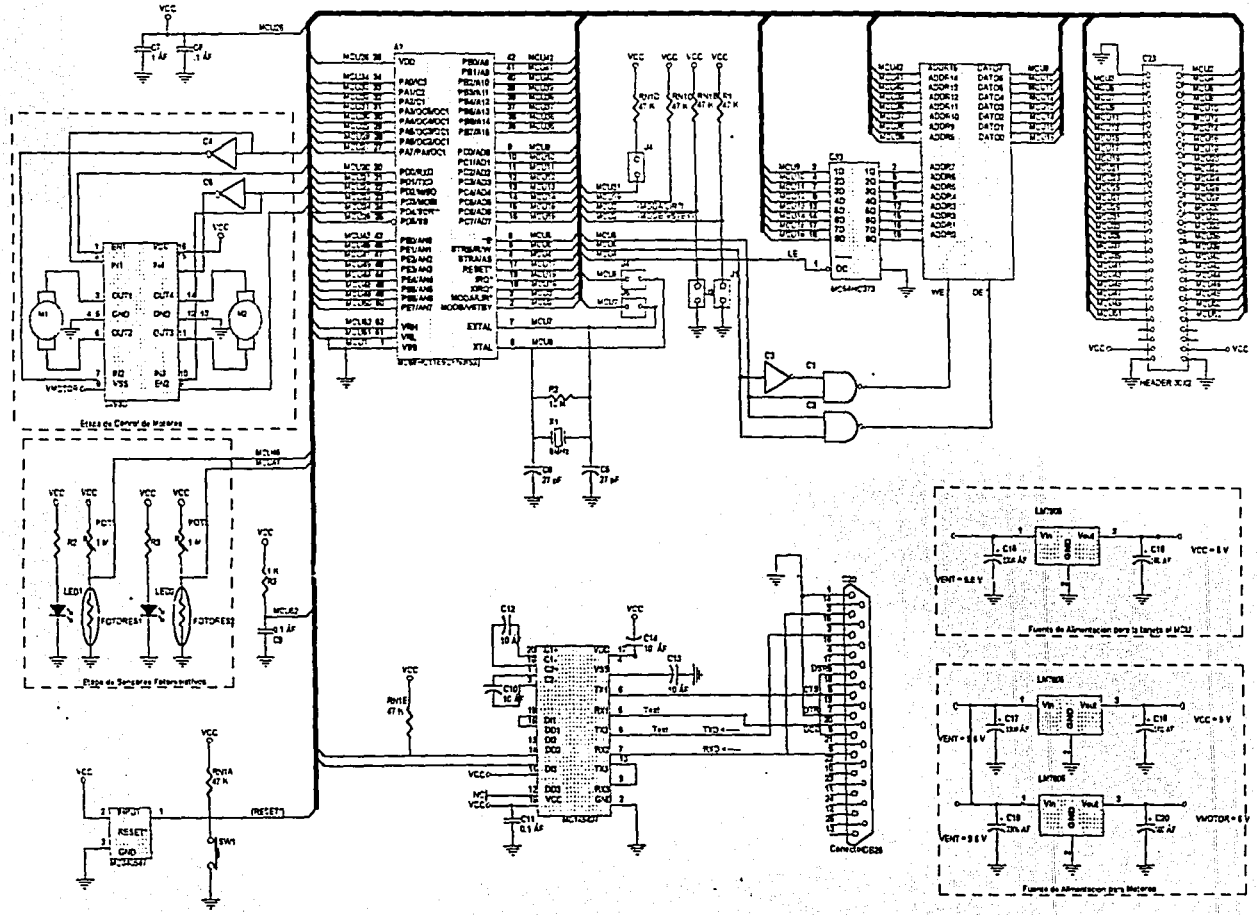


Figura 5.3 Diagrama Electrónico del Circuito Controlador del Robot Móvil

Como se observa en la Figura 5.3, se tienen 4 salidas por el puerto D del microcontrolador, las cuales sirven para controlar la dirección del movimiento de los motores, se decidió mantener una velocidad fija, determinada por anchos de pulso y solamente controlar la dirección del movimiento y el tiempo en el que se realiza el mismo. Para realizar el control por ancho de pulsos se utilizaron las funciones de comparación de salida OC3 y 0C4 y las salidas de datos D2 y D2 del microcontrolador. En la Tabla 5.1 se muestran las funciones para el control del robot.

D2	A5	D3	A4	FUNCIÓN
0	1	0	0	Atrás
1	1	0	0	Adelante
*	0	1	1	Izquierda
*	0	1	1	Derecha

Tabla 5.1 Señales de control para el robot móvil.

El robot móvil cuenta con una rueda de movimiento libre en la parte delantera; para realizar sus desplazamientos se utilizaron dos motores de corriente directa en la parte trasera, los cuales recibirán los comandos por la etapa de control de motores. El arreglo de motores se muestra en la Figura 5.4.

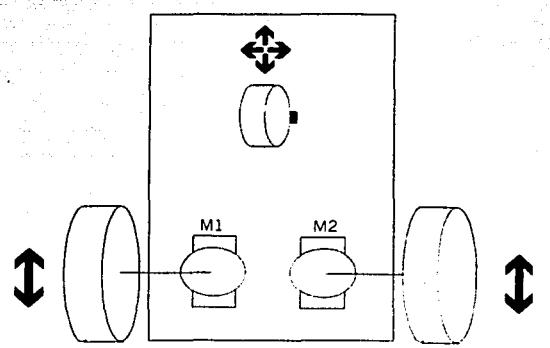
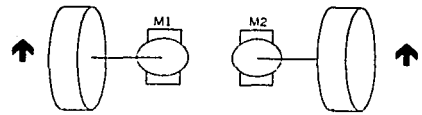


Figura 5.3 Estructura del robot móvil

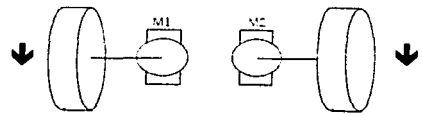
Para trasladar el robot tipo coche de un punto inicial a un punto final es necesario primero realizar un giro en la dirección correcta al punto final, y después avanzar un distancia para alcanzar el objetivo. Cabe mencionar que

debido a la estructura del robot, para realizar una vuelta los giros se realizarán haciendo funcionar los motores en sentido opuesto, donde el motor que gire en sentido contrario de la dirección, mandará en el movimiento del robot; las combinaciones de las direcciones de giro de los motores determinaran el movimiento del mismo; estos movimientos de giro se muestran en la Figura 5.4.

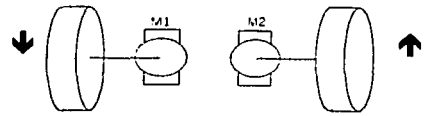
En la Figura 5.5 se muestra la construcción física del robot empleado para muestra primera implementación de nuestro algoritmo difuso (ver capítulo 6).



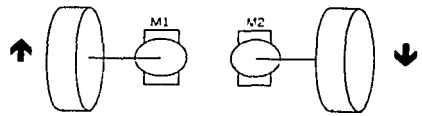
a)Desplazamiento hacia adelante



b)Desplazamiento hacia atrás

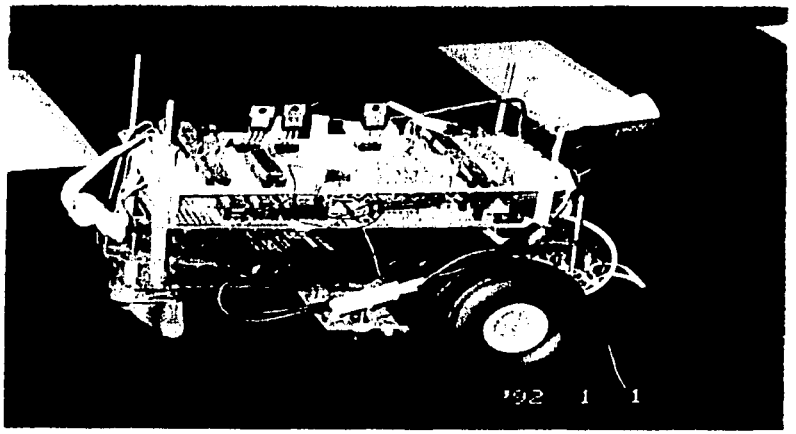


c) Giro a la Izquierda

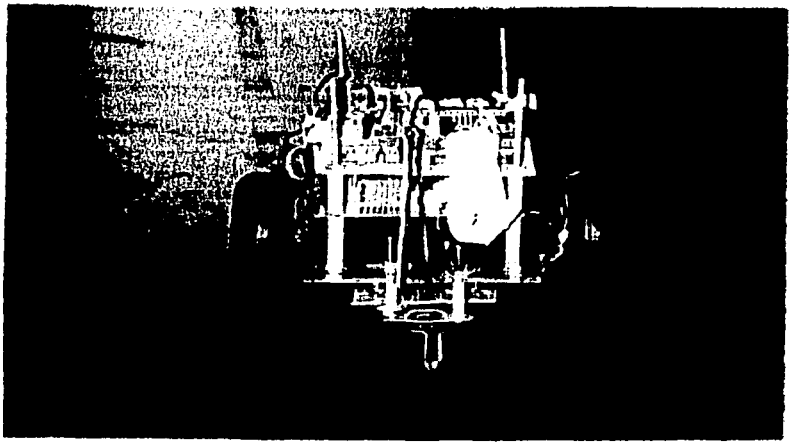


d) Giro a la derecha

Figura 5.4 Combinación para los movimientos del robot



b) Vista Lateral



c) Vista frontal

Figura 5.5 Apariencia físico del robot móvil.

5.2 Objetivo (primera aplicación)

La primera aplicación experimental se requiere que el robot móvil se desplace sobre el centro de una línea en un espacio libre obstáculos, con una velocidad constante. La ruta que seguirá estará compuesta de curvas de diferentes diámetros. Tal como se observa en la Figura 6.1

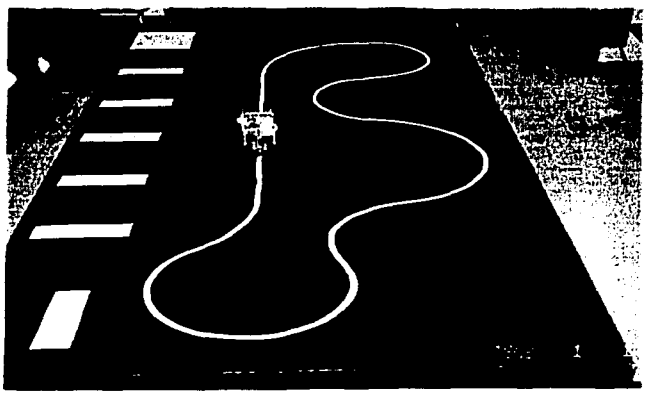


Figura 5.6 Pista diseñada para el recorrido del robot

El robot móvil será capaz de seguir el recorrido marcado con línea color blanco hasta llegar al otro extremo.

A continuación se explica algunos puntos importantes que fueron consideradas para el desarrollo del control difuso.

a) Selección de Variables

Las variables de entrada utilizadas para esta aplicación fueron las señales enviadas por los sensores fotoresistivos denominadas SENSOR_DERECHO y SENSOR_IZQUIERDO.

b) Partición de los Universos

En este punto es muy importante la capacidad de cálculo con que se cuente y la precisión deseada en el control. Cabe mencionar que no existe un método

definido para la obtención del valor óptimo del número de elementos del conjunto de términos, con base a esto se definió los siguientes conjuntos para las 2 variables de entrada, es decir la variable SENSOR_DERECHO y SENSOR_IZQUIERDO tomarón los siguientes valores: MO=MUY_OSCURO, O=OSCURO, INT=INTERMEDIO, C=CLARO y MC=MUY *CLARO*. De tal manera que cuando la lectura leída por el sensor cae dentro del subconjuto MO=MUY_OSCURO significará que el sensor no esta esta viendo la línea, cuando cae dentro del subconjunto difuso O=OSCURO significará que el sensor esta más próximo a la línea, y así sucesivamente hasta llegar al subconjunto difuso MUY=CLARO que significará que el sensor esta viendo la línea blanca.

c) Fuzificador

Este operador relaciona los datos medidos con sus correspondientes subconjuntos difusos. La Figura 6.3 muestra los conjuntos difusos definidos para las dos variables. La selección de la función de pertenencia está relacionada con la incertidumbre en las mediciones. Si por ejemplo, se eligen funciones de pertenencia trapezoidal, quiere decir que se tiene gran confianza en los valores que representan los subconjuntos difusos, por otro lado, si se elige una campana de Gauss o triángulo, significara que el nivel de confianza es menor que el caso anterior. Las funciones de pertenencia para las variables de entrada se muestran en la Figura 5.7

d) Reglas de Control

Se deduce de los conocimientos en ingeniería de control y de la experiencia previa.

En este caso tendremos reglas de tipo:

Si SENSOR_DERECHO es MUY_OSCURO y SENSOR_IZQUIERDO es MUY_OSCURO

Entonces DIRECCION ES ADELANTE y TIEMPO es MUY_LARGO.

Lo cual implica que se trata de un control difuso de tipo proporcional. Esto se nota de inmediato por que ante un cierto valor de las dos entradas, la corrección será proporcional a dicha entrada.

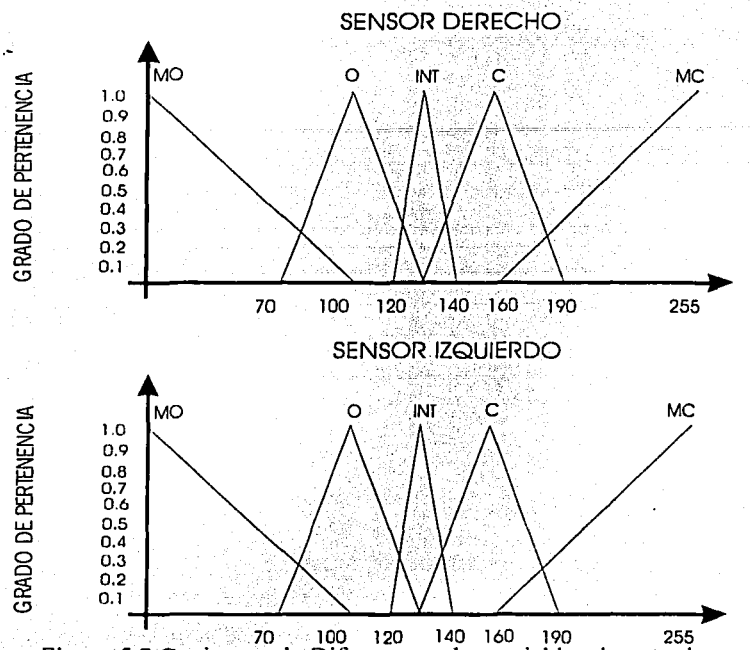


Figura 5.7 Conjuntos de Difusos para las variables de entrada

Dado que la cardinalidad de los conjuntos de términos de las variables de entrada es de 5, tendremos que el máximo número de reglas posibles es de 25, Por ejemplo, si hubiera dos variables de entrada y la primera tuviera una cardinalidad de tres y la segunda tuviese seis, entonces el número máximo de reglas posibles sería de 3(6)=18 En el presente ejemplo, estas reglas pueden acomodarse en un vector de decisiones (que en general sería una *matriz de decisión)*. Las reglas expresadas en forma matricial son la siguiente:

SENSOR_DER/SENSOR_IZQ	MUY_OSCURO	OSCURO	INTERMEDIO	CLARO	MUY_CLARO
MUY_OSCURO	ADELANTE	ADELANTE	ADELANTE	IZQUIERDA	IZQUIERDA
OSCURO	ADELANTE	ADELANTE	ADELANTE	IZQUIERDA	IZQUIERDA
INTERMEDIO	ADELANTE	ADELANTE	ADELANTE	ADELANTE	IZQUIERDA
CLARO	DERECHA	DERECHA	ADELANTE	ADELANTE	ADELANTE
MUY_CLARO	DERECHA	DERECHA	DERECHA	ADELANTE	ADELANTE

a) Reglas en forma matricial para la variable de DIRECCION

SENSOR_DER/SENSOR_IZQ	MUY_OSCURO	OSCURO	INTERMEDIO	CLARO	MUY_CLARO
MUY_OSCURO	MUY_LARGO	LARGO	MEDIO	MEDIO	MUY_LARGO
OSCURO	LARGO	MEDIO	POCO	MEDIO	MUY_LARGO
INTERMEDIO	MEDIO	POCO	MUY_POCO	MUY_POCO	MUY_POCO
CLARO	MEDIO	MEDIO	MUY_POCO	MUY_POCO	MUY_POCO
MUY_CLARO	MUY_LARGO	MUY_LARGO	MUY_POCO	MUY_POCO	MUY_POCO

b) Matriz de reglas para la variable de salida TIEMPO.

Tablas 5.2 Reglas en forma matricial

f) Lógica de Decisión.

Se utilizo la regla de Mamdani[1], para implantar la lógica de decisiones. Esta regla es un caso particular de la Definición 2.9. Para ilustrar dicha regla se da el siguiente ejemplo.

Supóngase que se tiene un grado de pertenencia para la variable de entrada SENSOR_IZQ correspondiente a los subconjuntos INTERMEDIO y OSCURO, con valor de 0.4 y 0.2 respectivamente, para el SENSOR_DER, corresponde a los subconjuntos difusos INTERMEDIO y CLARO, se obtiene los valores de pertenencia de 0.3 y 0.45. Enseguida se aplica la regla de Mamdani (Producto Lógico Mínimo) que consiste en tomar el valor mínimo de pertenencia de cada par de entrada como se observa en la Figura 5.10.

La correspondiente acción de salida se calcula por medio de reglas composicionales de inferencia (Definición 3.10) La cual esta en términos de una relación difusa.

g) Defuzificación

El método del centroide fue utilizado por las razones expuestas anteriormente, su ecuación es la siguiente.

$$Z_o = \frac{\int_z \mu_c(z) z dz}{\int_z \mu_c(z) dz}$$

[1] Se utilizó esta regla por ser la más popular.

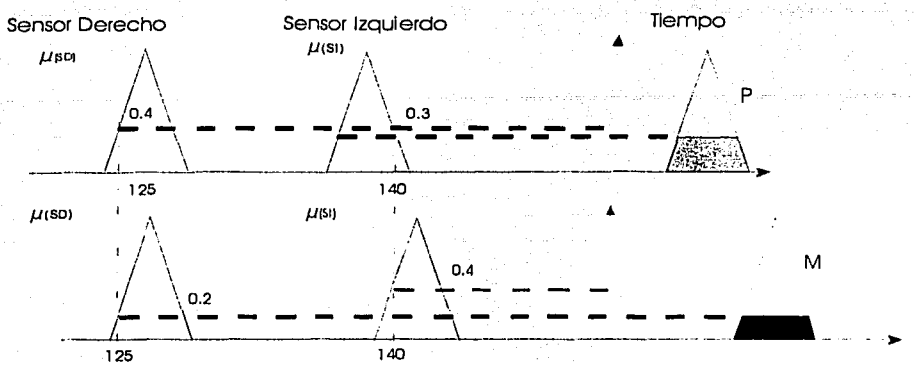


Figura 5.8 Ejemplo del proceso de inferencia

Las variables de salida difusa son TIEMPO de movimiento y DIRECCIÓN del mismo. A la variable de tiempo se le puede asignar los valores de MUY_POCO, POCO, MEDIO LARGO y MUY_LARGO, mientras a la variable de dirección se le asignan los valores de ADELANTE,DERECHA y IZQUIERDA. Si bien en nuestro caso por tener un control muy limitado de dirección, esta variable no cambiaría tan suave de un valor a otro como la variable de tiempo. La figura 5.10 muestra los conjuntos difusos para la salida TIEMPO.

Para la etapa de defuzificación para el caso de la salida TIEMPO tenemos el resultado, tal como se muestra en la figura 5.10 Para este caso se utilizo el método del centroide, el cual consiste de obtener el centro de gravedad de las áreas resultantes.

Ahora queda la selección del método de defuzificación. Como tenemos dos salidas de naturaleza distinta. Tendremos dos métodos de defuzificación: Por centroide de área para el TIEMPO y por conjunto de valor máximo para la DIRECCION.

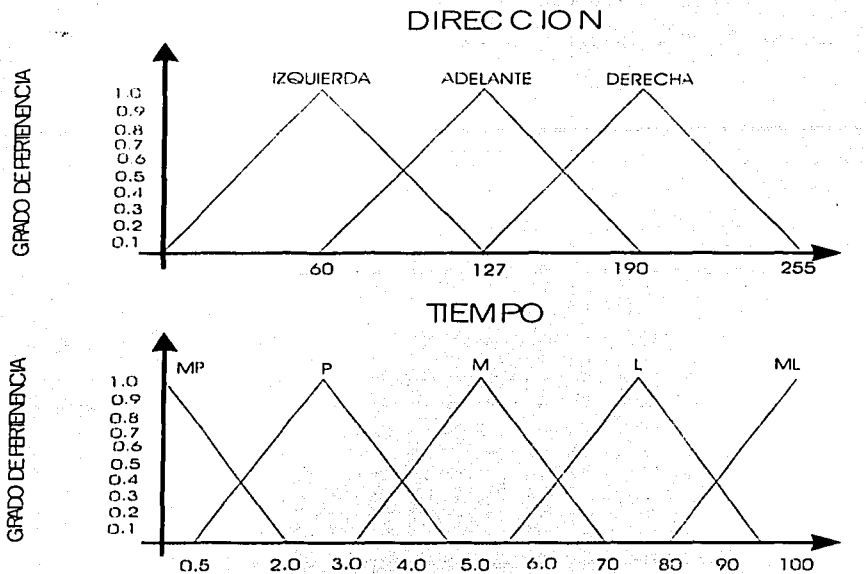


Figura 5.9 Conjuntos difusos para variables de Salida

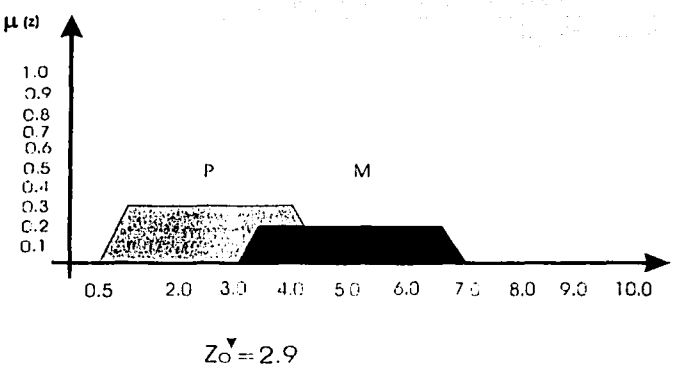


Figura 5.10 Proceso de Fuzificación para la variable de salida TIEMPO.

5.3 Objetivo (segunda aplicación)

En esta aplicación el robot partirá de un punto con cierta orientación, definida por el usuario y deberá alcanzar un punto y orientación deseada dentro de un área previamente definida.

El objetivo de utilizar algoritmo de control difuso, consiste en dotar al robot de la capacidad de obedecer ordenes un tanto ambiguas, por ejemplo,"ve hacia allá ", donde "allá" es un punto de una zona conocida. En esta aplicación con el robot *B14* se utilizo la organización típica para un robot móvil mencionada en el capítulo 4, debido a que el *B14* reúne las condiciones necesarias para la realización de nuestro objetivo, como se explicará más adelante.

5.4 Plataforma de desarrollo

Las características que presenta el robot *B14* define la plataforma de desarrollo de hardware y software para esta aplicación. El robot cuenta con una computadora Pentium con Sistema Operativo Linux, por lo tanto la plataforma de desarrollo de software y hardware de todos los módulos debe ser compatible. La Figura 5.11 muestra la estructura a bloques del controlador difuso.

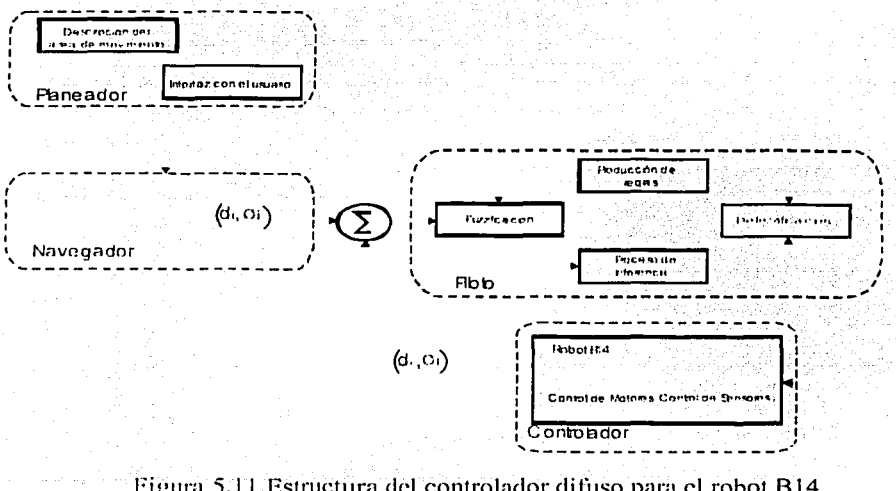


Figura 5.11 Estructura del controlador difuso para el robot B14.

Para implantar el algoritmo difuso se decidió utilizar CLIPS, la herramienta de desarrollo de Sistema Experto descrita en le Capítulo 2. La comunicación entre los módulos se realizan a través de la red, el Sistema Experto se puede correr en cualquier computadora que este conectada en red con el robot. En nuestra implantación se utilizó una Workstation Digital con procesador Alpha.

La función del *planeador* es básicamente la de recibir las instrucciones del usuario a través del teclado de la computadora. Una vez que recibe la orden de llegar a un punto meta, enseguida es determinar si este se encuentra dentro del área definida y esta información es enviada al navegador.

El *navegador* genera los movimientos continuos que lleva al robot de una posición a otra a través de una ruta localmente definida.

La implantación al *piloto* se utilizó el *API* (Applications Programming Interface) del *B14* [25], que permite instruir al controlador mediante la llamada a funciones incluidas. Estas funciones permiten hacer que el robot se desplace o gire, controlar su velocidad, aceleración, torque y obtener las lecturas de los sensores. También tiene un "*scheduler*" que es una tabla cuyos campos son funciones y el momento en que se debe ejecutar. Esto permite al programador definir la ejecución de funciones de manera periódica o como respuesta a señales de los sensores o eventos sin necesidad de definir ciclos estrictos de ejecución.

El módulo del *controlador* es el que esta definido por la estructura del robot y simplemente se utilizo.

5.5 Requerimientos del comportamiento

El usuario se comunica con el robot mediante una terminal o una computadora con conexión de red, a través del teclado y una interfaz gráfica. El robot será capaz de ejecutar la orden de alcanzar un punto dado, con coordenadas (x,y,ϕ).

El área donde se desplazará esta libre de obstáculos y previamente definida tal como se muestra en la Figura 5.12.

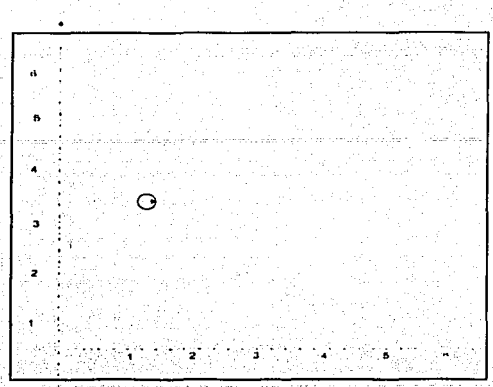


Figura 5.12 Mapa del Laboratorio de Interfases inteligentes.

Las dos señales generadas por el algoritmo y enviadas a la etapa controladora del robot son giros y desplazamientos. En cada giro y desplazamiento es utilizado el odómetro el cual nos indica la posición actual (x, y θ) dentro del área permitida.

5.6 Planeador

El planeador es el agente que tiene las siguientes tareas:

- Mantiene una representación del entorno del robot en la que se incluyen los límites del área donde se mueve el robot.
- Indica los puntos que debe seguir.

5.6.1 Definición de Área y hechos iniciales

Para representar el área en la cual el robot podrá moverse, es necesario declarar la misma antes de la ejecución de nuestro programa. Debido a que los movimientos del robot son sobre un plano, solo es necesario definirlos como un polígono asignándole sus coordenadas (x,y) de sus cuatro vértices, a través de hechos en la memoria de trabajo de CLIPS. En dicha memoria se asignan la definición de los subconjuntos difusos, así como los hechos iniciales requeridos para la ejecución correcta del programa.

Para definir el área de trabajo del robot se utilizó el siguiente formato en CLIPS:

(area_definida ?x1 ?y1 ?x2 ?y2 ?x3 ?y3 ?x4 ?y4)

Cada pareja *?xi ?yi* representa el vértice del polígono o área donde podrá el robot moverse libremente.

De esta manera podemos generar cualquier objeto dentro del área definida, si así lo requiriera la aplicación.

5.7 Navegador

La función del navegador es generar los movimientos individuales para llegar al punto deseado y después ordenar al piloto que ejecute estos movimientos.

- Envía al *piloto* las coordenadas y la orientación de los nuevos puntos hasta que el robot llegue al punto final deseado. Y este a su vez reporte la posición alcanzada.

- Mostrar la interfaz gráfico de secuencia de movimientos del robot.

La ruta que generada el *planeador* es a través de una lista de hechos, los cuales son tomados por el *navegador* elemento por elemento de esa lista y enviados al *piloto*, el navegador espera que este le reporte el punto alcanzado.

5.7.1 Ejecución de Movimientos

Una vez que la regla *promedio_angulo_distancia* se ejecuta genera un conjuntos de hechos *(centroide_distan?distdif))* y *(centroide_angulo?angulodif)*, los cuales servirán para determinar las diferentes posiciones que ocupara el robot antes de llegar al punto final. La regla de *envia_datos* recibe y envía las nuevas coordenadas del robot.

El *navegador* espera del piloto una cadena de la forma *(s2c ?indice ?nueva_posicion ?x ?y ?h)* cada que realice algún movimiento, la cual nos indicará la nueva posición del robot, esta cadena nos servirá para determinar la nueva distancia y ángulo de entrada con respecto a punto final para la etapa de fuzificación del *planeador.*

5.7.2 Interfaz gráfica

La interfaz gráfica muestra el área donde el robot se desplaza, así como los movimientos realizados para llegar al objetivo. Para desplegar algún objeto,

en este caso el robot en la pantalla, se utilizo un conjunto de funciones CONDOR [26] antes mencionadas.

Para desplegar en la pantalla tanto el área, como los movimientos del robot, se utilizo un conjunto de funciones que previamente se le introdujeron a CLIPS. Este conjunto de funciones son las funciones CONDOR [27], y nos sirven para abrir una ventana, trazar líneas u objetos, y requiere que se cuente con librerías de *Motif* para ser compiladas [26].

5.8 Piloto

El piloto se encarga de cumplir con las órdenes del navegador instruyendo al controlador los movimientos que debe realizar.

El piloto realiza las siguientes tareas:

- Reporta al navegador las coordenadas en cada movimiento, además de indicarle cuando llego al punto final.
- Recibe del Navegador los parámetros necesarios como son punto inicial y punto final, para el inicio de los cálculos del *piloto*
- Contiene también la base de reglas que definen la estrategia de control de movimientos del robot.
- Determina los subconjuntos relacionados de acuerdo a los valores medidos de las variables de entrada, en la etapa de *fuzificación*.
- El *proceso de inferencia*, simula la toma de decisiones para determinar las reglas activadas.
- La etapa de defuzificación calcúla el promedio de todas las reglas activadas y le indica al controlador cuándo y cuánto tiene que girar y desplazarse.

5.8.1 Base de reglas

Cuando se ejecuta un programa en CLIPS, se introduce en la memoria de trabajo el hecho *(initial-fac)*, si una regla se activa con este hecho, es posible garantizar que sea la primera en ejecutarse. Aprovechando esto se carga a la memoria de CLIPS en forma de hechos la base de reglas que define la estrategia de control; como ya se menciono en el capítulo 2; son reglas de la

forma *CONDICION-ACCION* (si-entonces) Estas reglas son expresadas en un lenguaje apropiado que permite describir el comportamiento del robot.

El conjunto de reglas almacenadas en la memoria de trabajos de CLIPS son expresadas con los siguientes hechos:

(rules_angulo ¿No_regla ¿antecedente1 ¿antecedente2 ¿consecuente)

Por ejemplo ; la regla numero 30 esta formada de la siguiente manera:

(rules_distancia 30 MN MUP MUN)

Donde *MN* el *¿antecedente1* para la distancia, *MUP* es el *¿antecedente2* del ángulo y el *¿consecuente* es el conjunto difuso *MUN* de salida. Los conjuntos de Entrada y Salida se definen en la secciones siguientes.

5.8.2 Fuzificación

Las variables de entrada para esta aplicación fueron *error_distancia* y *error_ángulo*, las funciones distancia y ángulo se ejecutan una vez que el robot realizó un desplazamiento y un giro, para obtener el error de distancia y ángulo que serán las siguientes entradas. La codificación en CLIPS del calculo de la distancia y el ángulo son las siguientes.

```
(deffunction distancia
    (?x1 ?y1 ?x2 ?y2)
    (bind ?dx (- ?x1 ?x2))
    (bind ?dy (- ?y1 ?y2))
    (bind ?regresa (+ (* ?dx ?dx) (* ?dy ?dy)))   ;Calcula la Distancia
    (bind ?dist (sqrt ?regresa))
    (return ?dist )
```

Cabe recordar que si el robot gira en el sentido opuesto a las manecillas del reloj se consideran giros positivos y negativos en el sentido de las manecillas del reloj, La función que realiza este cálculo se muestra a continuación.

```
(deffunction angulo1
    (?x0 ?y0 ?x1 ?y1)
    (bind ?dx (- ?x1 ?x0))
    (bind ?dy (- ?y1 ?y0))
    (if (and (= ?dx 0) (> ?dy 0)) then  ;Calcula el ángulo si el punto se encuentra en el
        (bind ?af 90)                   ;primer cuadrante.
    )
    (if (and (= ?dx 0) (< ?dy 0)) then
        (bind ?af -90)
```

```
    )
    (if (and (> ?dx 0) (>= ?dy 0)) then ;Calculo el ángulo si el punto se encuentra en
        (bind ?af1 (atan (/ ?dy ?dx))) ;el segundo cuadrante
        (bind ?af (/ (* 360 ?af1) 6.283185307))
    )
    (if (and (> ?dx 0) (< ?dy 0)) then;Calculo del ángulo si el punto se encuentra en el
        (bind ?af2 (atan (/ ?dy ?dx))); tercer cuadrante
        (bind ?af1 (/ (* 360 ?af2) 6.283185307))
        (bind ?af (+ 360 ?af1))
    )
    (if (and (< ?dx 0) (>= ?dy 0)) then
        (bind ?af1 (atan (/ ?dy ?dx)))
        (bind ?af (/ (* 360 ?af1) 6.283185307))
        (bind ?af (+ 180 ?af))
    )
    (if (and (< ?dx 0) (< ?dy 0)) then;Calculo del ángulo si el punto se encuentra en el
        (bind ?af2 (atan (/ ?dy ?dx)));cuarto cuadrante
        (bind ?af1 (/ (* 360 ?af2) 6.283185307))
        (bind ?af (+ 180 ?af1))
    )
    (return ?af)
)
```

Una vez que se determinaron los errores de distancia y ángulo del punto inicial al punto final, se determinan los subconjuntos involucrados, así como el grado de pertenencia correspondiente. Los subconjuntos de la variable de *entrada error_angulo* fueron los siguientes: *MUP=Muy_positivo, MP=Medio_positivo, PP=Poco_positivo, CE=Cero, PN=Poco_Negativo, MN=MedioNegativo, LB=Muy Positivo*, en estos conjuntos se definen giros izquierda en contra de las manecillas del reloj y giros a la derecha en sentido de las manecillas del reloj, además de su modificadores *"poco"*, *"mucho"*, "medio" y "muy", de tal manera que si la variable de error cae dentro del conjunto MUP que se encuentra en el intervalo [-180, -80]. En la tabla siguiente se muestran los elementos de los conjuntos definidos para la variable de entrada θ.

CONJUNTO DIFUSOS	CONJUNTO DE ELEMENTOS
Muy Negativo (MUN)	$-180 \leq MUN \leq -80$
Medio Negativo (MN)	$-100 \leq MN \leq -20$
Poco Negativo (PN)	$-30 \leq PN \leq 0$
Cero (CE)	$-5 \leq CE \leq 5$
Poco Positivo (PP)	$0 \leq PP \leq 30$
Medio Positivo (MP)	$20 \leq MP \leq 100$
Muy Positivo (MUP)	$80 \leq MUP \leq 180$

Tabla 5.3 Subconjuntos difusos para la variable de entrada *error_angulo*.

Los conjuntos para la variable entrada *error_distancia* están definidos de la siguiente manera: *Cero=CE*, *Muy Pequeña=MUP*, *Mediana=MED*, *Grande=GR* y *Muy grande=MUG.* En la tabla 6.4 se muestran los conjuntos para la variable distancia , se observa en la tabla que el conjunto difuso *MUG* tiene como limite máximo 5.5 mts.

CONJUNTO DIFUSOS	CONJUNTO DE ELEMENTOS
Cero(CE)	$0 \leq CE \leq 0.5$
Pequeña(MUP)	$0.5 \leq MUP \leq 1.0$
Mediana(ME)	$0.5 \leq MED \leq 2.0$
Grande(GR)	$1.5 \leq GR \leq 3.5$
Muy Grande(MUG)	$2.75 \leq MUG \leq 5.25$

Tabla 5.4 Subconjuntos difusos para la variable de entrada *error_distancia*.

Los conjuntos para las variables de entrada fueron codificados en CLIPS como hechos de la siguiente forma:

```
(fuzzy_sets_ent1 CE MUP MED GR MUG)
(fuzzy_sets_ent2 RB RU RV VE LV LU LB)
```

Para calcular los grados de pertenencia de las variables de entrada (ángulo y distancia) fue definida una regla por cada conjunto, las cuales se definieron de acuerdo a la siguiente ecuación:

$$f(x) = \begin{cases} \frac{2}{l}x & para \quad 0 < x < \frac{1}{2}l \\ \\ \frac{2}{l}(l-x) & para \quad 0 < x < l \end{cases} \tag{6.1}$$

Por ejemplo para la variable de entrada ángulo del conjunto difuso *MUP* la regla en CLIPS para formar este conjunto es la siguiente:

```
(defrule calculo-restax1-3
(declare (salience -8))
?angulo <- (angulo_entrada 2 ?variable2)
=>
(if (and (>= ?variable2 80) (<= ?variable2 130)) then ;Determina el conjunto difuso de
(printout t "fuzzy set MUP" crlf)                     ;entrada y su grado de pertenencia
(bind ?membresia3 (- (* (/ 2 100) ?variable) 1.6))
(printout t "El grado de membresia del Angulo (y3)= "?membresia3 crlf)
(assert (membresia ent2 LB ?membresia3)))
(if (and (> ?variable2 130)(<= ?varaible2 180)) then ;Determina el conjunto difuso de
```

```
(printout t "fuzzy set MUP" crlf)                    ;entrada y su grado de pertenencia
(bind ?membresia3 (+ (* (/ 2 100) (- 100 ?variable2)) 1.6))
(printout t "El grado de membresia del Angulo (y3)= "?membresia3 crlf)
(assert (membresia ent2 MUP ?membresia3))
```

Los grados de pertenencia de una variable a funciones de miembros complementarios, deben de sumar la unidad Esta es la única restricción y es debida a que esta lógica trabaja con los valores comprendidos en un intervalo de (0 a 1), por lo tanto no debe de rebasar este límite.

5.8.3 Proceso de Inferencia

En esta etapa se evaluarán las reglas activadas para determinar la salida de control correspondiente. La formación de decisiones lógicas usadas para el proceso de inferencia es muy importante, y puede ser el componente más flexible en el sistema difuso.

Cada regla activada nos indica en que conjunto difuso se encuentra la variable de entrada, así como el grado de pertenencia. Trabajando con este grado de pertenencia asignados por el proceso de fuzificación, el proceso de inferencia, realiza 2 operaciones
muy importantes para evaluar las reglas y poder producir una salida difusa que responda a las entradas, estas son: el producto lógico y la suma lógica.

El producto lógico de todos los grados de pertenencia es simplemente igual al valor mínimo de todos grados de miembros involucrados (regla de Mamdani). La figura 6.3 muestra este proceso.

La regla que lleva a cabo esta tarea es llamada *producto_logico_distancia*, junto con la función *logical-product*, la sintaxis de estas funciones se muestra a continuación:

```
(defrule Producto_logico_distancia

(membresia ?ent1 ?indice1 ?x3)     ;Función de pertenencia para la distancia
(membresia ?ent2 ?indice2 ?y3)  ;Función de pertenencia para el ángulo
(test (and (neq ent1 ?ent2)(neq ent2 ?ent1)(neq ?indice1 ?indice2)(neq ?indice2
indice1)))
=>
(bind ?desp (logical-product ?x3 ?y3))
(printout t " Producto Logico : " ?desp crlf)
(assert (membresia_salida_distancia ?indice1 ?indice2 ?desp)) ; valor mínimo.
)
```

En la regla *Producto_logico_distancia* realiza un llamado a la función *logical-producto* que se encarga de obtener el grado de pertenencia correspondiente para el conjunto de salida. Este mecanismo se realiza para cada una de reglas activadas. La sintaxis para esta función es la siguiente.

```
(deffunction logical-product    ;Calculo de producto lógico de los valores de Pertencia.
      ( ?x1 ?y1)
     (if (and (and (<= ?x1 1) (> ?x1 0))
              (and (<= ?y1 1) (> ?y1 0))) then
                 (if (< ?x1 ?y1) then
                     (bind ?xa ?x1)
                 else
                     (bind ?xa ?y1)
                 )
     else
      (bind ?xa 0) )
      (return ?xa)                  ;retorna el valor mínimo obtenido.
```

Una vez que se define el grado de pertenencia para cada regla "disparada" la etapa siguiente es determinar su consecuente con base en los antecedentes o conjuntos determinados en la etapa de Fuzificación.

La *regla consecuente_distancia* evalúa los conjuntos correspondientes contra el total de conjuntos para la variable distancia y ángulo para determinar el conjunto correspondiente de salida; mapeando a las reglas definidas como hechos en la memoria de trabajo de CLIPS, esto se realiza para cada regla. La regla que se muestra a continuación realiza lo antes descrito.

```
(defrule consecuente_distancia
(fuzzy_sets_ent2 ?MUN ?MN ?PN ?CE ?PP ?MP ?MUP )
 (fuzzy_sets_ent1 ?CE ?MUP ?ME ?GR ?MUG)
 (fuzzy_sets_distancia_salida  ?out_CE  ?out_MUP  ?out_ME  ?out_GR  ?out_MUG)
(membresia_salida_distancia ?indice1 ?indice2 ?desp)
=>
(bind ?i 13)                          ;realiza todas las combinaciones posibles sobre los
conjuntos
(while (> ?i 0                        ; entrada para encontrar el conjunto de salida
(bind ?set (nth$ ?i (create$ CE MUP ME GR MUG )))
(bind ?j 7)
  (while (> ?j 0)
      (bind ?set2 (nth$ ?j (create$ MUN MN PN CE PP MP MUP )))
          (if (and (eq ?set ?indice1)(eq ?indice2 ?set2)) then
             (assert  (fuzzy_salida_distancia  ?indice1  ?indice2  ?desp))  ;Conjunto de
salida )
   (bind ?j (- ?j 1)))
(bind ?i (- ?i 1))))
```

5.6.5 Defuzificación

Los variables de salida para el control difusos fueron *salida-ϕ* ,y *Salida-distancia*, para la *salida-ϕ*, los subconjuntos se definieron tal como se muestra en la Tabla 6.5.

CONJUNTO DIFUSOS	CONJUNTO DE ELEMENTOS
Muy Negativo (MUN)	-45≤MUN≤-25
Medio Negativo (MN)	-40≤ MN≤-10
Poco Negativo (PN)	-20≤PN≤0
Cero (CE)	-7.5≤CE≤7.5
Poco Positivo (PP)	0≤PP≤20
Medio Positivo (MP)	10≤MP≤40
Muy Positivo (MUP)	25≤MUP≤40

Tabla 5.5 Subconjuntos difusos definidos para la variable de salida ϕ.

Y para los subconjuntos de *salida-distancia*. .

CONJUNTO DIFUSOS	CONJUNTO DE ELEMENTOS
Cero(CE)	0≤CE≤0.25
Pequeña(MUP)	0≤MUP≤0.5
Mediana(ME)	0.25≤MED≤1.75
Grande(GR)	1.0≤GR≤3.0

Tabla 5.6 subconjuntos difusos definidos para la variable de salida distancia

El números de subconjuntos definidos para las variables de salida fueron codificados mediante hecho en CLIPS.

(fuzzy_sets_ distancia _salida CE MUP ME GR)
(fuzzy_sets_ angulo _salida MUN MN PN CE PP MP MUP)

Para los subconjuntos de entrada y salida, se construyeron triángulos con diferentes pendientes, para el caso de los conjuntos de salida se definieron los parámetros necesarios para realizar el calculo de sus áreas, para la variable *salida-ϕ* se definieron los 7 subconjuntos ,mediante los siguientes de hechos.

(Fuzzy_set_out MUN 20 40 45)
(Fuzzy_set_out MN 15 30 25)
(Fuzzy_set_out PN 10 20 10)
(Fuzzy_set_out CE 7.5 15 0)
(Fuzzy_set_out PP 10 20 -10)
(Fuzzy_set_out MP 15 30 250)
(Fuzzy_set_out MUP 20 40 45)

De la misma forma para la variable distancia.

```
(Fuzzy_set_out  CE  0.25 0.5 0)
(Fuzzy_set_out  MUP  0.25 0.5 0.25)
(Fuzzy_set_out  ME  0.75 1.5 1)
(Fuzzy_set_out  GR  1.0 2.0 2)
```

De los hechos anteriores, se generan áreas correspondientes a los conjuntos de salida. Como se menciono anteriormente la defuzificación consiste en calcular el centro de masa o centroíde, del área obtenida en el proceso de inferencia. El método de inferencia utilizado fue el del centro de gravedad. Cada subconjunto de salida es identificado mediante un hecho que cumple con la siguiente estructura:

```
(fuzzy_set_out  ¿nombre ¿parámetro1 ¿parámetro2 ¿parámetro3)
```

Donde la etiqueta *¿nombre* que identifica al subconjunto de salida, *¿parámetro1* indica el punto medio o central del subconjunto , *¿parámetro2* indica la base del subconjunto y finalmente el *¿parámetro3* nos informa el punto central del conjunto respecto al eje ***U*** o universo de discurso.

Una vez disparadas las reglas *suma_areas_distancia* realiza la suma de áreas correspondientes al conjunto de salida (consecuente) de cada regla.

```
(defrule suma_areas_distancia
(declare (salience -39))
  ?fact_1 <- (area_distancia ?indice_1 ?area1); suma área de los conjuntos de salida
  ?fact_2 <- (area_distancia ?indice_2 ?area2)  ;en la variable distancia.
  (test (neq ?indice_1 ?indice_2))
=>
  (retract ?fact_1)
  (retract ?fact_2)
  (bind ?resultado (+ ?area1 ?area2))
  (if (> ?indice_2 ?indice_1) then
    (assert (area_distancia ?indice_2 ?resultado))
  else
    (assert (area_distancia ?indice_1 ?resultado))  ;Guarda la suma.))
```

La regla *suma_area_m_distancia* realiza la suma de la multiplicación del área con el punto central correspondiente al conjunto de salida, es decir *A1*Punto medio del conjunto1*, donde *A1* es el área de conjunto de salida, tal como se muestra en la ecuación 3.9 (ver capítulo 3).

```
(defrule suma_areas_m_distancia
(declare (salience -40))
  ?fact_1 <- (a*setm_distancia ?indice_1 ?area1) ;Multiplicación del Área con el Punto
```

```
?fact_2 <-(a*setm_distancia ?indice_2 ?area2) ;medio del conjunto correspondiente.
  (test (neq ?indice_1 ?indice_2))
=>
  (retract ?fact_1)
  (retract ?fact_2)
  (bind ?resultado (+ ?area1 ?area2))
  (if (> ?indice_2 ?indice_1) then
    (assert (a*setm_distancia ?indice_2 ?resultado))
  else
    (assert (a*setm_distancia ?indice_1 ?resultado)) ;Toatal de la suma.
  ))
```

Finalmente con la regla *promedio_area_distancia* obtenemos la señal de salida para el control del robot. Esta señal de salida es obtenida usando la ecuación (3.9).La codificación de la regla *promedio_area_distancia*, se lista a continuación.

```
(defrule promedia_areas_distancia
(declare (salience -42))
?fact1<- (area_distancia ?indice ?area1)
?fact2 <- (a*setm_distancia ?indice ?aream)
 =>
(retract ?fact1 ?fact2)
  (bind ?dist_dif (/ ?aream ?area1))
  (assert (centroide_distan ?dist_dif))
(printout t "Centroide (Distancia) = " ?dist_dif crlf) ;Salida distancia difusa.
(assert (membresia_distancia))
)
```

De igual manera existe una función *promedio_area_angulo* para el cálculo de la variable de salida ángulo ϕ, la cual se omite por tener la misma estructura.

5.8.1 Scheduler

El *scheduler* permite definir en un programa un conjunto de funciones que se ejecutan periódicamente o después de que se reciba algún mensaje de la base.

Para hacer uso de este mecanismo se debe primero arrancar el programa servidor de comunicaciones (*tcxServer*), que permite comunicar al todos los programas que participan en el control del B14, y el servidor de la base (*baseServer*).que comunica a los motores y a los sensores con la aplicación del usuario.

El programa del usuario debe localizar al servidor de la base, introducir los módulos que participan en el *scheduler* y darle el control al *scheduler*, esto se hace con las siguientes instrucciones dentro de la función *main* del piloto:

```
RegisterBaseClient ( );
InitClient("read_socket",commShutdown);
FindBaseServer( );   /*se engancha al baseServer está corriendo*/
RaiInit( );          /*inicializa,o arranca, el schenduler*/
catchInterrupts ( );
initClientModules( ); /* introduce comportamientos al scheduler*/
RaiStart ( ); /*arranca el scheduler*/
return,
```

En la función *CreateModules* se inicializa la posición y orientación del robot, se establecen las velocidades , aceleración y torques de giro y translación y se informa al *scheduler* qué módulos tienen que ejecutarse periódicamente y que módulos van a responder a un evento generado por registros de estatus o por la base (*callbacks*).A continuaciones muestran el código de ésta función:

```
void createModules (void)
{
RaiModule *communicate Module, *gotoModule;
RaiModule *evadeModule, *arbitrateModule;
Printf ("Setting up the modules\n")
loadPosition (0x8000800); /*posición inicial */
loadheading (0);          /*orientación inicial */
statusReportPeriod  (100*256/100), /*Tiempo entre reporte */
registerStatusCallback (statusCallback);
registerBaseCallback (baseCallback);
setTranslateAcceleration (800);
setTranslateVelocity (200);
setRotateAcceletion (300 );
setRotateVelocity(300);
setRotateTorque(255);

/*Establece módulo para poleo cada 100 milisegundos*/

communicateModuel = makeModule("comunicate",moduleShutdown);
addPoling (communicateModule,communicatePoll,100);
gotoModule= makeModule("goto", moduleShutdown);
addPolling (gotoModule,gotopoll,50);
arbitateModule=makeModule ("arbitrate",moduelShutdown);
addPolling (arbitrateModule,arbitratePoll,300);
printf ("done\n");}
```

El *callbacks* de status se activa cada que hay un cambio en el estado del robot, por ejemplo posición y orientación. Se le aprovecha para actualizar las

variables de posición y orientación que se utilizan los demás módulos cuando el odómetro reporta cambios. El *callback* de la base se utiliza para detectar el momento en que el robot termina sus movimientos.

5.9 Controlador

El controlador se encarga de realizar los movimientos enviados por el piloto, como son giros y desplazamientos. El controlador es un modulo que ya se encuentra implementado en el robot B14. A través de interfaz de programación de aplicaciones (API) se establecen la velocidad, aceleración, torque de giro y desplazamientos se envían a órdenes a la base [27]. La base del robot es controlada por el microcontrolador NEC 78310L que se encarga de mover los motores y controlarlos para realizar sus giros y desplazamientos requeridos. Además utiliza dos microcontroladores MC68HC11 que controlan a todos los sensores del robot y reportan, cuando así se requieren los datos que éstos recogen. Para nuestra aplicación se fijo una velocidad constante, es decir no fue de nuestro interés la aceleración y el tiempo, que toma en llegar al punto final.

5.10 Comunicación entre Módulos

Comunicación entre los módulos se realiza mediante sockets, que es el mecanismo definido por UNIX BSD para comunicar procesos [25]. Esto permite que los módulos puedan operar en computadoras distintas, con el único requisito de que estén interconectadas a través de una red. En el sistema UNIX existen varias API´s que permiten la comunicación entre procesos [22],los más importantes son los sockets de Berkeley y la Interfaz de la Capa de Transporte (TLI) de systems V. Ambas utilizan los servicios de TCP/IP para realizar la comunicación, por lo que ya se han traducido a otras plataformas para permitir comunicación entre computadoras sin importar el sistema operativo con el que se trabaje [25].

La comunicación entre el piloto y el controlador, se utilizó la arquitectura *Cliente_Servidor* de *Beesoft*, formada por un conjunto de programas que tienen incorporados el robot B14 y que tiene varias funciones. Para el problema de éste trabajo se utiliza el programa "*baseServer* " y "*tcxServer*".El programa "*baseServer*" envía órdenes a los controladores de los motores para que se puedan realizar que se puedan realizar desplazamientos y giros, además recibe las lecturas de los sensores del robot. El programa "*tcxServer*" coordina las comunicaciones entre todas las computadoras del robot y sus dispositivos,así como entre los servidores y los programas de usuario (cliente) [26]. En la Figura 6.6 se muestra esta arquitectura.

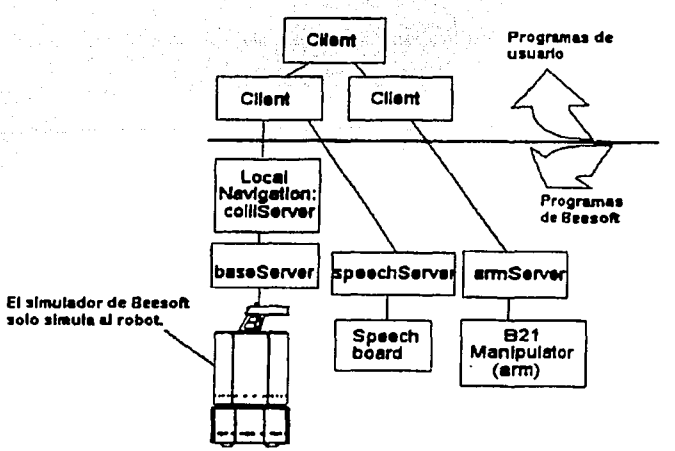


Figura 5.13 Arquitectura Cliente-Servidor de Beesoft.

5.11 Consideraciones

Cabe hacer mención que la construcción de un robot móvil fue el primer objetivo que se propuso al inicio de este trabajo, con el fin de tener un prototipo experimental, y contar con una infraestructura adecuada para realizar investigación en el área de la robótica, posteriormente se añadieron módulos adicionales necesarios (sensores, transductores, etc) para aplicaciones específicas, siendo una de ellas la que se muestra en este trabajo.

El algoritmo para aplicación fue desarrollado en lenguaje C para el Microcontrolador 68HC11, los fragmentos de este algoritmo se muestran en el apéndice F.

Para el caso del robot B14 un punto importante que se debe de cuidar al integrar los módulos es verificar que los programas se comuniquen a través del puerto correspondiente. El navegador envía ordenes al piloto por el puerto 2000, lee sus resultados y solicitudes por el puerto 2001.

El piloto está codificado en un único programa que hace uso del API del B14 y que deben correr en la máquina Pentium que éste tiene. El único programa que forzosamente debe de correr en el B14 es el piloto, todos los demás pueden correr en cualquier computadora que esté conectada a través de una red TCP/IP con el robot, aunque lo ideal es que todos los programas corran en el mismo.

Capítulo 6

Experimentos y Resultados

En éste capítulo se describen las pruebas realizadas para verificar el funcionamiento de los robots. Para la primera aplicación se construyó una pista con fondo negro y sobre esta, una línea blanca con anchura de 1.5 centímetros, que definiera la ruta que debería seguir el robot móvil. Para el robot B14 se realizó sobre una superficie libre de obstáculos, los dos experimentos se realizaron en el Laboratorio de Interfaces Inteligentes del Edificio Valdez Vallejo de la Facultad de Ingeniería.

6.1 Primera Aplicación

Las pruebas se realizaron en una pista formada de segmentos en línea recta y curvas de diferentes diámetros, como se muestra en el capítulo 6 (ver figura 5.6).

Una vez diseñada la pista para el robot móvil se procedió cargar nuestro algoritmo dentro de memoria RAM externa de la arquitectura diseñada (ver capítulo 4) y posteriormente fue ejecutado, para valorar su desempeño.

El primer paso fue realizar unas pruebas y ajustar nuestro algoritmo de tal manera que realizara el recorrido en el mínimo de tiempo, este ajuste como ya se comentó en capítulos posteriores se puede realizar de 2 formas:

a) Moviendo los subconjuntos de entrada para forzar que los datos medidos estén dentro de un subconjunto deseado.

b) Realizando ajuste en las reglas y determinar, así la acción de control correspondiente.

El robot se deja libre sobre la pista, una vez que cualesquiera de los dos sensores detecta blanco, inmediatamente toma la acción de rectificar para que los dos observen negro, de tal manera que realice la corrección necesaria

hasta tomar la posición adecuada y seguir el recorrido marcado por la ruta. En las pruebas se observó que el robot frecuentemente se salía de la pista, cuando "tomaba" una curva, por lo cual para eliminar este problema se realizaron ajustes para que la respuesta de control fuera más lenta, comparada con la respuesta para segmentos de línea recta. Se realizarón previamente algunas pruebas de tal manera que el robot hiciera el recorrido completo a la pista; tomando el tiempo efectuado en realizar dicho recorrido para cada prueba. Después de algunas pruebas y modificaciones en las reglas del algoritmo se encontró que el tiempo mínimo que tomó el robot en realizar una vuelta completa fué de 62 seg.

6.1.1 Análisis de Sensibilidad

Una vez realizadas las pruebas y ajuste en las reglas, así como el determinar el tiempo mínimo realizado en completar una vuelta. El análisis realizado a nuestro algoritmo es de sensibilidad, que consiste en cambiar algunas reglas aleatoriamente y tomar el tiempo que toma en realizar un recorrido completo a la pista, para esto se consideraron los siguientes parámetros.

Tiempo de recorrido (TR) Tiempo que el robot tardó en realizar una vuelta completa.

% Reglas alteradas (%RA) Porcentaje de Reglas alteradas en el algoritmo difuso.

La Tabla 6.1 muestra los tiempos de recorrido medidos para los diferentes porcentajes de reglas alterada.

PORCENTAJE DE REGLAS ALTERADAS (%RA)	TIEMPO DE RECORRIDO (SEGUNDOS)
0	62
10	63
20	66
30	81
40	83.3
50	85.9
60	89
70	79.2
80	85.9
90	80
100	71

Tabla 6.1 Datos obtenidos

La Figura 6.1 muestra la gráfica de los datos obtenidos, considerando el tiempo de recorrido **(TR)** contra Reglas alteradas **(%RA)** en el algoritmo difuso.

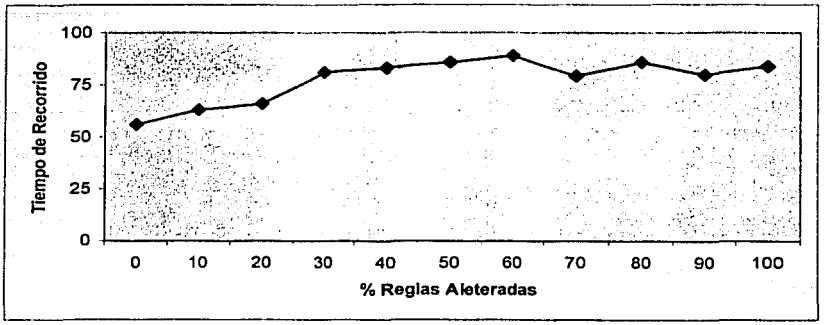


Figura 6.1 Comportamiento de la trayectoria del robot

6.2 Segunda aplicación

Para verificar el funcionamiento del algoritmo implementado en el Robot B14, la prueba consistió en indicarle al B14 el punto y ángulo inicial, así como punto y ángulo final dentro del área definida; algunos puntos medidos son los mostrados en la Tabla 6.2.

Por ejemplo se quería que el robot se trasladara del punto inicial de coordenadas (2,2,0°) al punto destino de coordenadas (5,5,45°), obteniéndose la posición final del robot como(5.28, 4.7, 23°), podemos observar que existe una desviación de la posición final del robot con respecto al punto de destino deseado. Para cuantificar esta desviación se considera la siguiente restricción:

Cuando la distancia de la posición inicial de robot al punto destino deseado, sea menor o igual a 2 veces el radio del Robot (el diámetro del robot de 34 cm. ver apéndice D), se considera que éste alcanzó su destino.

Partiendo de esta consideración se puede obtener un porcentaje de error con respecto a la posición del robot al punto destino, de tal manera que las variables utilizadas son las siguientes:

Diámetro del Robot (DR)= es el diámetros del robot

Distancia del Robot al Punto Destino (DFD)= es la distancia entre el punto final (punto alcanzado por el robot) al punto destino medida en centímetros.

%Error_desviación (%ED) = [100(DFD-DR)]/DR

En la tabla 6.1 podemos observar que una desviación entre el punto meta y el punto alcanzado, así como su ángulo de llegada.

El **%ED** es evaluado una vez que el robot realiza el último de sus movimientos.

PUNTOS	ORIGEN(X, θ)	META(X,Y, θ)	PUNTO ALCANZADO (X,Y, ϕ)	DFD	%ED
1	(2 ,2 , 0)	(5 , 5 45)	(5.28 , 4.7 30)	41.03	20.6
2	(2, 2 30)	(5, 5 , 30)	(5.44 ,5.4, -14)	59.46	74.88
3	(5, 2, 30)	(3 , 3 10)	(3.30 , 2.81, 30)	35.51	4.41
4	(5, 2, 0)	(3 , 3 20)	(3.20 , 3.28 30)	34.40	0.109
5	(1,3, 20)	(2,4,0)	(2.25,4.3,20)	62.64	84.11

Tabla 6.2 Pruebas realizadas con el Robot B14 para diferentes puntos

A continuación se presentan los resultados gráficos de los puntos mostrados en la tabla anterior, así como los desplazamientos individuales necesarios realizados por el robot para llegar a su objetivo.

En la figura 6.2 se muestra el punto No. 1 de acuerdo a la tabla 1, se observa que el punto alcanzado por el robot es (5.28, 4.7, 30°), de esta manera se observa que los desplazamientos individuales necesarios realizados por el robot para llegar para llegar al punto deseado son dos, de tal manera que se observa *% ED* tiene un de valor de 20.6%.

La tabla 6.3 muestra los desplazamientos individuales realizados por el robot para tratar de llegar al punto No. 1 de acuerdo a la tabla 6.2.

NO. DE MOVIMIENTOS	ORIENTACIÓN (RADIANES)	DISTANCIA (CM)	PUNTO ALCANZADO (X,Y)
1	0.833	75	(2.50,2.55)
2	-0.171	35.3	(5.28,4.7,30)

Tabla 6.3 Secuencia de movimientos realizados por el robot

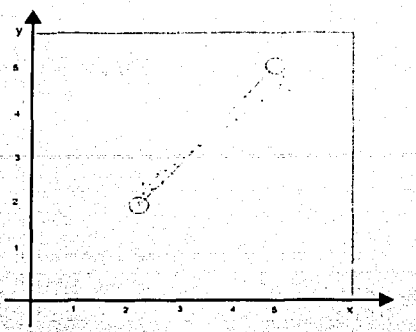


Figura 6.2 Desplazamientos realizados por el robot para alcanzar el punto 1 de acuerdo a la tabla 6.3

De la misma manera ahora se muestra el punto No. 2, se observa que el punto meta, es el mismo punto que el anterior, con diferencia es que tenemos una orientación inicial diferente; de tal forma que el punto alcanzado por el robot (5.44, 5.4, -14), así obtenemos una *%ED* de 74.7 % tal como se observa en la tabla 6.1, de esta manera se muestran en la tabla 6.4 los desplazamientos necesarios realizados por el robot para llegar al punto deseado No.2

NO. DE MOVIMIENTOS	ORIENTACIÓN (RADIANES)	DISTANCIA (CM)	PUNTO ALCANZADO (X,Y)
1	1.047	75	(2.0,2.75)
2	1.047	193	(3.67,3.72)
3	0.261	137.8	(4.65,4.69)
4	-0261	106.2	(5.44,5.4,-14)

Tabla 6.4 Secuencia de movimientos realizados por el robot

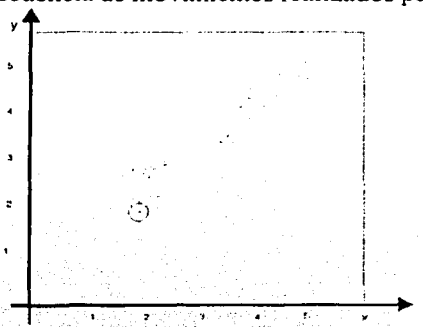


Figura 6.3 Desplazamientos realizados por el robot para alcanzar el punto 2 de acuerdo a la tabla 6.2

6.2.1 Análisis de Sensibilidad

De igual manera que en la primera aplicación se realizó un análisis de sensibilidad al algoritmo difuso. Una vez que se realizaron algunas pruebas para diferentes puntos iniciales, en seguida se escogieron los siguientes puntos, el punto inicial con coordenadas (2, 2,30) y el punto final (5,5, 45).

El siguiente paso fue alterar algunas de las reglas de control, partiendo de un ajuste previo en las mismas. Una variable a medir es el *error_en_la_trayectoria* que es la relación de la *longitud de la trayectoria* actual del robot dividido por la *distancia trayectoria en línea recta* del punto inicial al punto deseado, su definición es la siguiente:

$$Error\ en\ la\ trayectoria = \frac{Longitud\_de\_la\_trayectoria.}{Distancia(Posicion\_inicial, Posición\_final\_deseada)}$$

donde *longitud de la trayectoria* es la suma de distancias parciales realizadas por el robot para llegar al punto deseado.

Con base en esta variable y al porcentaje de reglas alteradas, los resultados se muestran en la figura 6.4.

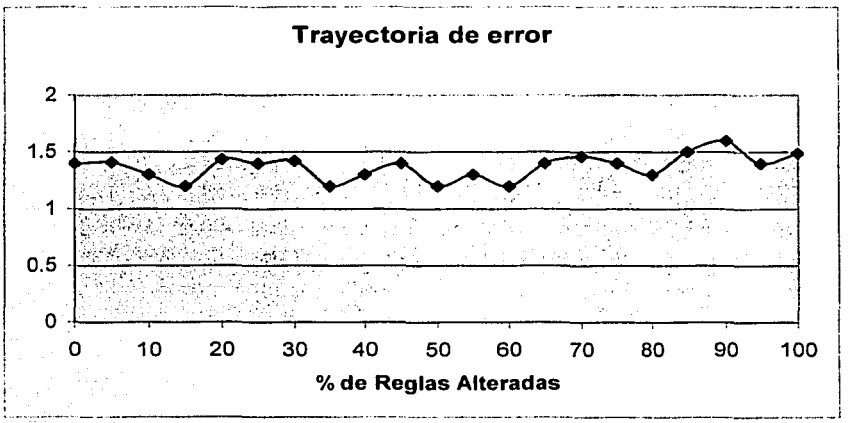


Figura 6.4 Gráfica de *Error_en_trayectoria* del controlador difuso

Capítulo 7

Conclusiones

Características principales de los controladores difusos

En el presente trabajo se han introducido los conceptos básicos de la lógica difusa, la cual compite con nuestra habilidad para razonar y hacer uso de datos aproximados para encontrar soluciones precisas, así como cuando no se tiene el modelo matemático de un proceso a controlar, es decir se cuenta con la experiencia para controlar un proceso aun cuanto no se conozca su interpretación matemática, ni sus variables estado, así como tampoco de su función de trasferencia.

La lógica difusa no esta limitada a problema de control, en realidad la lógica difusa originalmente fue pensada para poder ser más aplicable a problemas de "ciencias no exactas", tal como psicología, economía, biología etc.

Los fundamentos mencionados arriba hacen posible establecer las siguientes características que distinguen a la lógica difusa de los sistemas tradicionales.

1.- Los conocimientos son interpretados como una colección de límites elásticos en una colección de variables.

2.- Proporciona las herramientas necesarias para el tratamiento de la INEXACTITUD, un concepto asociado al conocimiento humano.

3.- Es una extensión de Multi-niveles donde todo es manejado como grados de verdad.

La lógica difusa intenta modelar las experiencias de los seres humanos así como su comportamiento en la toma de decisiones, estas experiencias están fundamentadas en la experiencia adquirida por el operador del proceso a controlar.

Aunque unas de las desventajas que se tendría de conocer el comportamiento del proceso es realizar pruebas modificando las reglas de inferencia hasta que

el desarrollo del control difuso sea el adecuado, es decir por lo general el ajuste de un algoritmo difuso se realiza mediante un método sistemático de ensayo, el cual carece de formalidad; sin embargo produce buenos resultados. Además que cuando se trata de describir el comportamiento de un proceso el controlador difuso puede tener un número muy grande de reglas y por consecuencia el tiempo de procesamiento es mayor.

Actualmente, están disponibles una gran cantidad de herramientas para desarrollar y modelar sistemas de control difuso, algunos ellos son *Fuzzy TECH* , *FIDE (Fuzzy Inference Developnent Environment)* etc, incluyen lenguajes de inferencias difuso, un editor gráfico(con el se pude dibujar cualquier función de pertenencia) y generador de código en tiempo real, facilitando la simulación de nuestro proceso.

Primera aplicación

Motivado por los aspectos arriba mencionados; en este trabajo se presentan el desarrollo de dos algoritmos difusos tipo Sugeno de primer orden.

En la primera aplicación el desarrollo de algoritmo se realizó en el compilador de Lenguaje C para el microcontrolador 68HC11,que resulta una herramienta de gran ayuda, ya que se evita estar programando en ensamblador, aunque esto implicará conectar a nuestro sistema mínimo memoria externa.

En cuanto al desarrollo del robot se observó que siempre concluyó la tarea de seguir el camino marcado , aunque se observó que el tiempo de ejecución es mayor en algunos casos, la cual no fue importante para nuestro objetivo.

Cabe hacer mención que el diseño y construcción de un robot móvil era otro de los objetivos al principio de este trabajo, ya que era necesario contar con un prototipo experimental, y así poder verificar el funcionamiento de este algoritmo.

Segunda aplicación

Con la adquisición de un robot B14 fabricado por la empresa Real World Interface nos dio la pauta para profundizar más en esta área de la Lógica Difusa.

Ahora el objetivo era desarrollar un algoritmo difuso para controlar el B14, pero bajo la estructura de un sistema experto utilizando el lenguaje CLIPS.

El sistema experto basado en la teoría difusa y diseñado bajo la organización mostrada en el Capítulo 4, presenta una característica importante que es la separación explícita entre datos y hechos o conocimientos que permiten hacer crecer el sistema en forma modular. Por ejemplo si se desea modificar o añadir las funciones de pertenencia, en la etapa de fuzificación; solo es necesario definir los parámetros para la nueva función ó si por ejemplo, se requiere modificar o añadir el conjunto de reglas de inferencia, sólo es necesario definirlas como hechos.

Por otro lado, los datos obtenidos en la tabla 6.2 se observa que el error máximo obtenido es de 84.1% aunque esto puede parecer demasiado, no lo es tanto si lo comparamos con los demás datos, en donde el error no fue mayor a 25 %. Cabe hace mención que solo se estudio algunos puntos en especifico, de lo contrario se tendría un sin fin de pruebas que tendrían que realizarse.

En cuanto análisis de sensibilidad los puntos elegidos para realizar este análisis fueron punto inicial con coordenadas (2,2,30°) y el punto deseado con coordenadas (5,5,45°), los resultados obtenidos en la gráfica 6.3. muestran que al modificar en un porcentaje el banco de reglas, el robot siempre estuvo cercano al punto deseado.

Trabajos futuros

El robot B14 permite implanta programas rápidamente utilizando sus sensores y actuadores, con sólo hacer llamadas a funciones en Lenguaje C. Para lograr esto es necesario entender su interfaz de programación. Además existen otros programas de propósito especifico, como servidor de voz y otro para usar un brazo manipulado, los cuales pueden ser utilizados para otras aplicaciones.

Por ejemplo una aplicación interesante consistiría en dotar al robot de la capacidad de obedecer órdenes un tanto ambiguas, por ejemplo "ve hacia allá" en donde "allá" puede ser cualquier punto dentro de zona previamente definida.

Apéndice A

Clips

Clips es un shell para realizar sistemas expertos que se empezó a desarrollar en el centro espacial Johnson de NASA desde 1985 y que ya está en versión 6.0. Aquí se presenta una breve introducción; para mayor información es necesario consultar el manual de referencia.

A.1 Interacción con CLIPS

CLIPS es un intérprete, así que los programas escritos para él se pueden alimentar mediante teclado o se pueden introducir en un archivo y hacer que los ejecute clips. En UNIX el comando par ejecutar el intérprete es :

\%clips [-f <archivo>]

Si la opción *–f* no se alimenta, el programa entra en modo interactivo y admite instrucciones del usuario. Si por otro lado la opción *–f* se introduce, el programa busca el archivo y lo ejecuta.

A.2 Elemento básicos de programación

CLIPS admite ocho tipos de datos: float, integer, symbol, string, external-addres, fact-addres, instante-name e instante-addres. Ninguna variable tiene que declararse, ya que CLIPS le asigna el tipo de dato correspondiente a la primera asignación que haya recibido.

A.2.1 Representación de datos

El mecanismo que más se utiliza en CLIPS para representar la información son los hechos. Los hechos son almacenados en una lista que también se

conoce como base de hechos y forman la unida fundamental de datos utilizada en las reglas. Las operaciones que se pueden realizar con los hechos son : agregar (con el comando *assert*), retirar (con el comando *retract*), modificar (con el comando *modify*) y duplicar (con el comando *duplicate*). Estas operaciones se realizan dentro de las reglas o a través de interacción directa con el usuario.

Un hecho puede tener dos formas: ordenado o no ordenado. Un hecho ordenado es un símbolo seguido de cero más campos separados por blancos y delimitado por paréntesis como en el caso de:

(robot uno 1 1 1 10)

(Objeto caja 1 1 1 2 2 2 2 1)

Un hecho no ordenado se forma mediante la sentencia *deftemplate* que permite asociar a un dato un conjunto de slots o campos que pueden tener algún valor, cada campo es otro hecho. Por ejemplo:

(robot posicion 1 1) (orientacion 10))

A.2.2 Representación de conocimiento

Hay dos formas de representar el conocimiento. Los conocimientos heurísticos se representan mediante reglas de producción y los conocimientos procedurales mediante funciones.

Las reglas de producción se definen mediante la construcción *defrule* que tiene las siguientes sintaxis:

(defrule <nombre_de_regla> [<comentario>]
[<declaración>] ; propiedeade de las regla
<elemento_condicional> ;Lado izquierdo de la regla
=>
<acción>) ; Lado derecho de la regla*

Los campos rodeados por corchetes son opcionales. Para incluir un comentario se introduce un ";" y CLIPS toma el resto de la línea como comentario. Los elementos condicionales y la acción pueden ser cero o más. Las acciones son una secuencia de llamada a funciones, afirmaciones o eliminación de hechos. Cuando se cumplen todos los elementos condicionales se ejecutan en orden todas las acciones.

Las funciones se definen con la construcción *deffuncion* cuya sintáxis se muestra a continuación:

(deffuntion <nombre_de_función> [<comentario>]
(<parámetro regular_> [<parámetro comodín>]*
<acción>)*

Un parámetro regular es una variable de un sólo campo que se reconoce por estar antecedida por un signo de interrogación?, mientras el parámetro comodín es una variable multicampo que se reconoce por estar antecedida de un signo de pesos seguido de la interrogación *$?*.

Apéndice B

LM18293/I293 MANEJADOR DE POTENCIA

B.1 Descripción General

El LM18293 es un circuito integrado diseñado para manejar motores hasta de 1 Amp. Entre las aplicaciones típicas incluye manejo de cargas inductivas como solenoide, relevadores, motores de corriente directa y motores de paso, emplea internamente los transistores de potencia y utiliza un buffer para señales de nivel bajo.

En la figura B.1, se presenta el patigrama de este dispositivo, contiene cuatro entradas para ingresar señales de control de motores de paso, acepta niveles estándares de lógica TTL y DTL, para realizar su interfaz; dos señales de habilitación para controlar la velocidad, que también aceptan la misma lógica. Cada habilitador controla dos canales; cuando el pin de habilitación está desactivado (cero lógico), las salidas correspondientes se encontrarán con lógica de tres estados, si el pin no está conectado (flotando), el circuito funcionará como si estuviera habilitado.

Se cuenta con dos pines para proporcionar el voltaje; el pin 8 proporciona la potencia del motor y el pin 16 suministra un voltaje independiente del anterior, que polariza los circuitos internos. El chip está incluido en un diseño DIP de 16 pines, el dispositivo es capáz de operar con voltajes máximos de 36 volts.

La figura B-2 muestra la forma de conectar dos motores y controlar al mismo tiempo el sentido de giro, en este caso podrán girar ambos en sentido horario y antihorario.

Características

- Salida por canal de 1 Amp.
- Reemplazo directo por el circuito integrado L293B y L293 D
- Empaquetado DIP de 16 pines
- Protección térmica contra sobrecarga

- Cero Lógico hasta 1.5 volts.
- Alta inmunidad al ruido.

Máximos rangos de volts

- Voltaje para las cargas (Vs) 36 volts
- Voltaje de la fuente Lógica (Vss) 36 volts
- Voltaje de entrada (Vi) 7 volts
- Habilitación de voltaje (Ve) 7 volts
- Corriente de salida 2 amperes.

Características Eléctricas

Vs=24V, Vss=5V, T=25 oC, L=0.4V, H=3.5V.

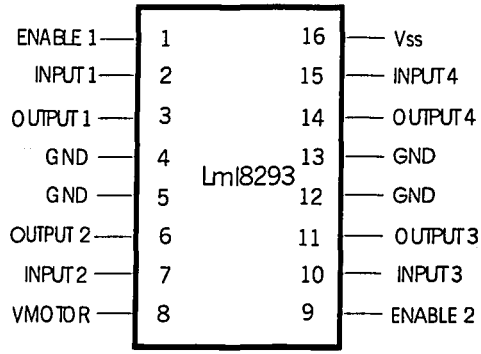


Figura B-1 Asignación de pines

B.2 Control de Motores de Corriente Directa

Para controlar los motores se utiliza el circuito integrado L293D, que como se describió anteriormente, nos presenta la capacidad de controlar los dos motores.
Como se muestra en la figura D-1 y de acuerdo a las características de este circuito. Se requiere de cuando menos cuatro señales de control, las cuales serán otorgadas por el microcontrolador. Para controlar un motor se requiere

de una señal que entregará el comando de dirección del motor, es decir; hacia donde deseamos que gire (derecha o izquierda), y otra señal que nos proporcionará la velocidad de giro del motor. Para el control del otro motor, se necesita de la misma cantidad de señales.

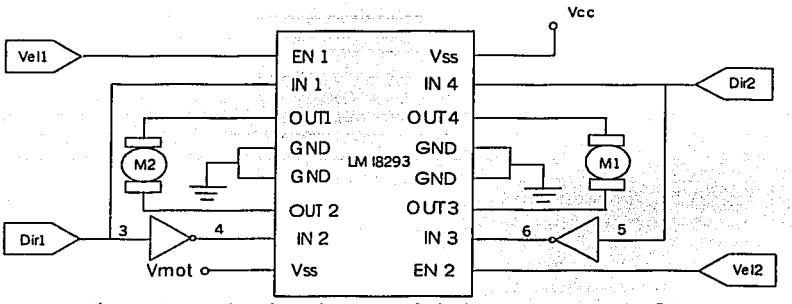


Figura B-2 Circuito de control de loas motores de CD.

B.3 Interfaze de comunicación (SCI)

El sistema transmisor receptor asíncrono universal (UART) de tipo interfase de comunicación serie (SCI), el cual es uno de los dos tipos de subsistemas independientes seriales de E/S en el MC68HC11, permite al MCU ser eficientemente una interfase con los dispositivos periféricos que permiten un formato de datos seriales en forma asíncrona. El SCI utiliza un formato estándar de no retorno a cero (NRZ) con una gran variedad de Baudio derivados del circuito de reloj de cristal. La interfase es realizada usando las terminales del puerto D, el pines PDO es usado para la recepción de datos(RxD) y PD1, para la trasmisión de datos (TxD); el circuito de generación de velocidad en bauds contienen un reloj programable, pre-escalador y divisor.

Formato de datos.

El formato de los datos seriales requiere lo siguiente:

- Una línea desocupada en un estado de prioridad alta para la transmisión ó recepción de un mensaje.
- Un bit de comienzo (cero lógico) que es transmitido-recibido, indicando el comienzo de cada carácter.
- Datos que son transmitidos y recibidos primero con el bit menos significativos.
- Un bit deparada (uno lógico) usando para indicar el final de un frame. Un frame consiste de un bit de inicio, carácter de 8 o 9 bits de datos y un bit de parada.

- Un break, el cual está definido como la transmisión o recepción de un cero lógico para algunos múltiples números de frames.

La selección de la longuitud de palabra es controlada por el bit M del registro de control SCCR1.

B.3.1 Operación de transmisión

El registro SCI incluye un registro de datos de transmisión paralela, llamado SCDR, y un registro de corrimiento serial que coloca el dato del SCDR en forma serial. El contenido del registro de corrimiento serial solo puede ser escrito mediante el SCDR. Este sistema de doble tope permite sacar un carácter serialmente usando corrimientos mientras otro caractér está esperando en el SCDR para ser transferido al registro de corrimiento serial. La salida del registro de corrimiento serial estará conectada al PD1 mientras la transmisión esté en progreso o mientras el bit "habilita al transmisor" (TE) del registro de control de comunicaciones serie (SCCR2) esté en uno.

B.3.2 Operación recepción

En operaciones de recepción, la secuencia de transmisión es invertida. El dato es recibido en el registro de corrimiento serial y es transferido a un registro de datos de recepción paralela (SCDR), como una palabra completa. Este sistema de doble separación permite que un caractér sea introducido serialmente por medio de un corrimiento mientras otro caractér está ya en el registro SDCR. Un avanzado mecanismo de recuperación de datos es usado para distinguir un dato válido del ruido en la cadena de datos serial. La entrada de datos es muestrada selectivamente para detectar el dato recibido, y un circuito de elección mayoritaria determina el valor y la integridad de cada bit.

B.3.3 Los Registros de control SCI

Principalmente, el sistema SCI esta configurado y controlado por medio de cinco registros, (BAUD, SCCR1, SCCR2, SCSR Y SCDR). Además del registro del puerto D, el registro de dirección de datos del puerto D (DDRD), y el bit de modo OR alambrada para el puerto D en el registro de control SPI (SPCR) están relacionados de manera secundaria al sistema SCI.

Apéndice C

C.1 Compilador para el HC11 en C y Pcbug11

El compilador de lenguaje C para el microcontrolador 68HC11 permite traducir el programa de aplicación de Lenguaje C, en código de ensamblador para los microcontroladores MC68HC11. Este compilador consiste de los siguientes archivos: un programa manejador (ICC11.EXE), un preprocesador de C (ICPP.exe) y un compilador generador de código (ICCOM.EXE

El compilador es un sistema tradicional que acepta y se ajusta al lenguaje ANSI. El compilador invoca al programa manejador (ICC11.EXE), a las funciones necesarias y al archivo que contiene las rutinas básicas del ensamblador (BASICS.S) para poder procesar el archivo en C (archivo.c)

El archivo BASICS.S contiene también el inicio del programa en donde se inicializa el apuntador de pila y se hace un llamado del *main( )*, que es la rutina que se ejecuta al principio de un programa en C. Si el compilador cruzado no detecta ningún error, éste produce un archivo objeto, que contiene el código en ensamblador (ARCHIVO.S). Si existió un error, se indica dónde se encuentra y de que tipo se trata.

El programa manejador (ICC11.EXE) direcciona al preprocesador (ICPP.EXE) para que busque en los directorios especificados los encabezados de los archivos que contienen las funciones en C y que son necesarios para generar el archivo con el código en ensamblador.

El preprocesador (ICPP.EXE) agrega al archivo en C las funciones necesarias (Headers files), generando un archivo en lenguaje de alto nivel C (ARCHIVO.I). Finalmente el compilador (ICCOM.EXE) traduce el código del archivo en lenguaje de alto nivel en un código fuente para el ensamblador.

C.1.1 Ejecución de un programa

La manera de programar, generar y correr una aplicación en el microcontrolador 68HC11 es la siguiente:

1.- Se crea el programa fuente en C, utilizando el editor de texto de su preferencia.

2.- Se guarda el programa utilizando la extensión C.

3.- Empleando el programa CC6811.bat que realiza la liga entre el compilador, ensamblador, programas con rutinas y declaraciones en lenguaje ensamblador, se generan los programas con formato s, i, S19 y LST.

Se ejecuta de la siguiente manera:

CC6811 Nombre_programa

El archivo CC6811.bat contiene lo siguiente.

```
Icpp.exe –DHC11 –D__LCC_-I/icc11/include %1.c51.i
Iccom11.exe –v %1.i %1.s
Ias11 –o%1 –1 basics.s %1.s
```

4.- Una vez teniendo el programa con el formato S19, está listo para ejecutarse.

5.- Con el PCBUG11, que es un programa que permite tener control de nuestra unidad, se puede cargar, correr, monitorear el estado de los registros, manipulación de datos en memoria est.

6.- Se cuenta con otro archivo, EPROM.bat que realiza la tarea de configuración y selección del puerto de comunicación serial donde se conectó el microcontrolador, así mismo de cargar.un macro que configura el modo de operación, para trabajar en expandido; el contenido del archivo EPROM es el siguiente:

PCBUG11 –E PORT = (1/2) MACRO=EPROM

7.- Una vez en el sistema de PCBUG11, se carga el programa con la siguiente instrucción:

LOADS Nombre_programa

8.- Se manda a ejecutar el programa:

G Dirección_inicio

9.- El sistema con la aplicación seleccionada comenzará a ejecutarse. En el apéndice

C.2 Programa Pcbug11

El Pcbug11 es un paquete de software que permite una sencilla operación y un fácil acceso a los microcontroladores de la familia M68HC11. El PCBUG11 permite programar cualquier miembro de la Familia M68HC11 y examinar el comportamiento de los periféricos internos bajo condiciones especificas. Además permite cargar programas en la RAM, ejecutarlos, correrlos paso a paso o correrlos programado puntos de ruptura.

Una vez conectada la tarjeta al puerto serial de la computadora se presiona el botón de *reset* en la tarjeta y en la computadora se ejecuta el programa PCBUG11 de la siguiente manera:

PCBUG11 –E MACRO =MACRO

Por omisión el programa selecciona el puerto 1 para establecer la comunicación con la tarjeta En caso de usar otro puerto se debe anexar la directiva que le indica al programa cual puerto se utilizará. En el siguiente ejemplo se selecciona el puerto 2.

PCBUG11 –E MACRO= PORT=2

C.3 Ventanas de la pantalla

C.3.1 Ventana principal

Esta ventana ocupa la mitad de la pantalla y se encuentra en la parte superior de la misma. Esta ventana despliega la mayor parte de información sobre la operación de l PCBUG11: resultado de comandos, contenidos de memoria, códigos de operación, mnemónicos, macros, etc.

C.3.2 Ventana de registros

Esta ventana se encuentra en el centro de la pantalla : Esta ventana muestra el contenido de los registros del procesador. Note que los valores de la ventana de registros se actualizan sólo al arrancar a petición del usuario.

C.3.3 Ventana de Estados

Esta ventana se encuentra en la parte central derecha de la pantalla. Esta ventana muestra el MCU (corriendo, deteniendo, paso a paso); el estado de la línea RS232 RTS, y los vectores de interrupción actuales fijados por el usuario.

C.3.4 Ventana de Comandos

Esta ventana se encuentra en la parte inferior izquierda de la pantalla. Use esta ventana para introducir y leer comandos del PCBUG11. El cursor de comandos (el carácter >>) se encuentra en la línea de la parte inferior de esta ventana; los comandos que introduzca aparecen después del cursor de comandos. Los comandos previos y el último mensaje de errór también aparecen en la ventana de comandos.

C.3.5 Ventana de Errores

Esta ventana indica cualquier error en la comunicación con el microcontrolador. Para borrar la ventana de error se presiona la tecla ESC.

C.3.6 Ventana de Ayuda

Esta ventana despliega la información de ayuda a través del comando HELP. Para borrar la ventana de ayuda, basta presionar la tecla ESC.

Apéndice D

Robot B14

El robot cuenta con sensores tales como sonares, infrarrojos, además cuenta con un odómetro para determinar sus posición en cualquier momento. El B14 es un robot móvil diseñado para investigación en el área de la robótica y la inteligencia artificial.

D.1 Descripción del Robot

EL robot B14 está equipado con una CPU Pentium Pro, además exísten 16 sensores infarrojos,16 sonares y un odómetro. La comunicaciones entre el CPU y los sensores se realiza via puerto serial RS232.

La base del B14 es la parte de le robot móvil donde se localizan los motores, ruedas, baterías y los controles electrónicos. La base mide 17.2 cm de altura y 34 cm de diámetro. En la figura D.1, se puede observar la estructura física y algunos de los componentes del robot B14.

D.1.1 Bus de Comunicación

EL B14 utiliza diferentes tipo de bus de comunicación. Una lista de ellos es la siguiente.

- Comunicación asincrióna RS-232. Cada CPU contiene 2 puerto RS-232.
- Red Ethernet. El robot puede ser equipado con un adaptador de Ethernet el cual permitirá una comunicación de alta velocidad (10Mbit/seg).
- Puerto paralelo. El CPU instalado en el B14 tiene un puerto paralelo el cuál es utilizado para control en varios caminos.
- Bus para los sonares.
- Bus de sensores Infrarrojos.

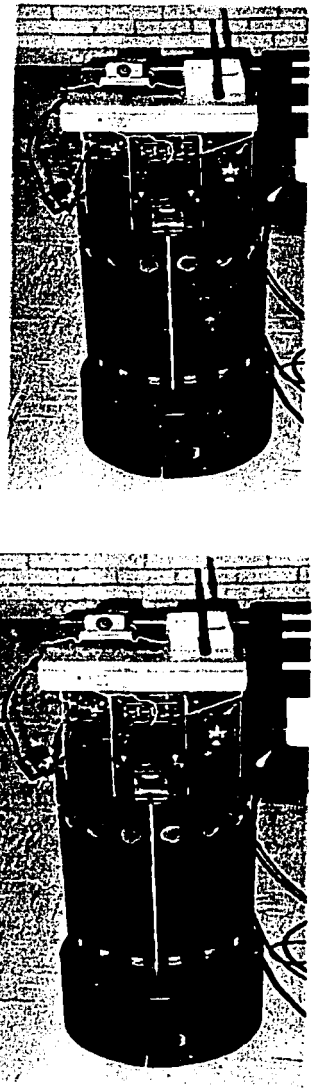


Figura D.1 Estructura física del robot B14 de RWI

D.2 Sensores y Actuadores

El robot cuenta con sonares, sensores infrarrojos, sensores de contacto y sensores de posición. El odómetro se utiliza para determinar la posición y orientación del robot con respecto a una referencia inicial, después de cada desplazamiento. Su principal problema es que basa sus lecturas encodificadores, que miden el número de veces que giran las ruedas. Este puede llevar a lecturas imprecisas o erróneas por los desniveles en la superficie que se desplace.

Los actuadores que tiene el robot forma un arreglo de tipo "synchro-drive".En la figura D.2 se muestra la base del B14. En la D.1. a) el robot esta orientado a la derecha y en D.1. b) ha girado 45° sobre su eje en sentido opuestos a las manecillas del reloj. Se observa el cambio de orientación, las tres ruedas cambian su orientación de manera sincrónica.

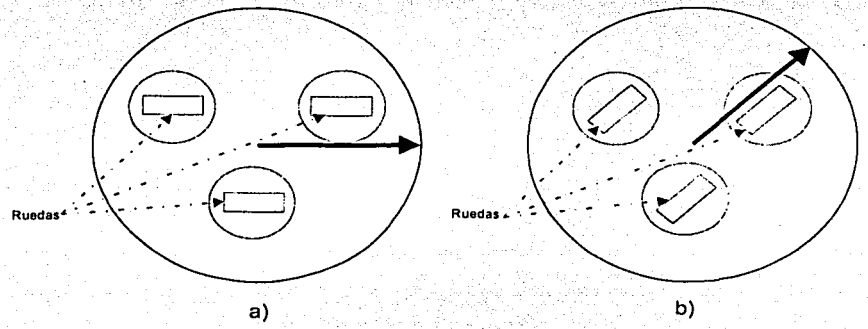


Figura D.2.Ruedas en arreglo Synchro Drive

En la tabla D.1 se lista las características y requerimientos necesarios para el funcionamiento del robot B14.

Computador	*Comentario*
NEC783OL	Microcontrolador para control de la base.
MC68HC11	Dos microcontrolador para controlar sensores.
Pentium	Computador con sistema operativo Linux, tarjeta de Red ethernet 10baseT y 10base2 y puerto serie.

Sensores	
Características Físicas	
Forma de huella	Circular.
Diámetro	35 cm.
Altura 60 cm.	
Peso 27.2 kg.	
Distancia base-peso	1.27 cm.

Computadoras	
Sensores	
Sonares	Anillo horizontales de 16 sonares a 50 cm.
Infrarrojos	Anillo horizontal de 16 emisor-sensor a 24 cm.
Tacto	22 sensores de tacto en toda su superficie.
Odómetro	Encodificadores, para conocer posición y orientación del robot.

Actuadores	
Motores	2 servo motores de 12 V de DC de gran torque.
Giros	3 ruedas de giro.
Conducción	3 ruedas en arreglo sincróno.
Diámetro de ruedas	8.36 cm.
Radios de giro.	0
Velocidad translación	90 cm/s.
Resolución translación	155° / s.
Resolución rotación	0.350 °.
Capacidad de carga	20 kg.

Energía	
Energía requerida para su funcionamiento.	
Convertidor DC-DC	Suminstra 10"@ +5V,2A@+12V y 2A@ -12V.
Baterías	2 intercambiadores 192 W-hr.
Tiempo de recarga	4 horas apagado; 10 horas encendido.
Tiempo de autonomía	1 a 2 horas con movimientos, 3.5 horas inmóvil.

Tabla D.1 Requerimientos y características del robot B14

Apéndice E

Arquitectura del robot móvil

E.1 Introducción

Actualmente las aplicaciones basadas en microcontroladores se han desarrollado enormemente. En la vida diaria las encontramos en juguetes, lavadoras y automóviles, etc. Particularmente, el área de la robótica también es auxiliada más de los microcontroladores como una herramienta.

La familia de los microcontroladores de Motorola MC68HC11 es bastante popular en aplicaciones de robótica, por su tecnología CMOS con bajo consumo de potencia y diversos subsistemas integrados en un solo chip; los cuales son adecuados para desarrollos autónomos en potencia, permite además manejar un número importante de dispositivos con interacciones hacia el mundo real.

En este capítulo se describe el diseño y construcción de un robot móvil, cabe hacer mención que en este desarrollo se utilizaron recursos existentes en el Laboratorio de Interfases Inteligentes ubicado en el Edificio Váldez Vallejo como la tarjeta de evaluación (MCU) basada en el microcontrolador 68HC11E9, así como la electrónica necesaria para llegar a nuestro objetivo de nuestra primera aplicación en la implementación de algoritmos difusos.

E.2 Características del MC68HC11

La tarjeta de evaluación consiste en un microcontrolador MC68HC11E9 de Motorola (MCU), el cual por su arquitectura interna de 16 bits, sus módulos de integrados (8 canales de conversión A/D, puerto serie, temporizador, 512 bytes de memoria RAM, 512 bytes de memoria EEPROM y 4 puertos de 8 bits, además de su precio accesible y su facilidad de programación constituye una de las opciones actuales más eficientes y baratas para automatizar procesos [23].

La unidad microcontroladora (Microcrontroler Unit) MC68HC11E9, es un sistema de alta velocidad de procesamiento y bajo consumo de energía, cuenta además con las siguientes características:

- Sistema de control libre de 16 bits con 4 niveles de preescalamiento.
- Modos de espera para el ahorro de energía.
- Interfaz de comunicación serie.
- 5 puertos de 8 bits.
- 2 registros acumuladores de 8 bits y uno de doble de 16 bits.
- 2 registros indexados de 16 bits.
- 8 kbytes de memoria ROM.
- 512 bytes de memoria RAM.
- 512 bytes de memoria EEPROM.
- Convertidor analógico/digital de 8 canales de 8 bits.
- 8 fuentes de interrupción.

E.2.1 Modo de operación

Existen únicamente dos modos fundamentales de operación para el Microcontrolador 68HC11: *Expandido* y *Simple*. Cada modo tiene una variación normal y una variación especial. Estas variaciones son seleccionadas por medio de los niveles de las terminales modo A (MODA) y modo B (MODB) durante el restablecimiento (*Reset)*. La variación especial del modo simple se denomina modo *Bootstrap*; la variación especial del modo expandido se llama modo especial de *Prueba*. El modo especial de *Bootstrap* permite que los programas sean descargados a través de la interfase de comunicación serial (SCI) hacia la memoria de acceso aleatorio (RAM) para que sea ejecutado. El modo especial de prueba, está enfocado principalmente para prueba de fabricación, y es utilizado por parte del usuario para emulación y desarrollo [23].

El *MCU* utiliza dos pines (MOD A y MOD B) para seleccionar uno de los cuatro modos de operación, dos de ellos llamados básicos *(simple y expandido)* y dos especiales *(Bootstrap y prueba)*.

Simple : En este modo de operación, el MCU hace uso de sus recursos internos y no tiene buses externos, por lo que se aprovechan al máximo sus puertos de entrada-salida sin embargo se tiene la restricción de la memoria interna de 512 bytes.

Multiplexado–Expandio : Para este modo el microcontrolador puede tener un acceso de hasta 64K bytes de memoria física. La parte baja del bus se encuentra multiplexado con el bus de datos y son tomados del puerto C, la

línea "AS" se utiliza como línea de control para el demultiplexado de datos y direcciones; en este modo se reduce el número de puertos de entrada-salida, se cuenta además con líneas de control para el acceso a memoria o periféricos externos (AS, $R/\overline{W}$ y E).

Debido a la programación con el compilador de C, es necesario disponer de memoria RAM para ejecutar nuestra aplicación tanto en memoria interna como externa, el bus de direcciones está formado por el puerto B y F, mientras que el puerto C es utilizado para Datos.

Modo "bootstrap": Este modo de operación combina los modos anteriores, aunque cambia la posición de los vectores de interrupción, pueden ser usado para programar la EEPROM interna, para probar el programa residente en memoria en ROM y puede ser cambiado a otro modo bajo el control del programa. Para nuestra aplicación se utilizaron dos modos de operación, el modo *Bootstrap* y *expandido.* El modo de operación *Bootstrap* es considerado como un modo de operación especial distinto al modo de operación normal single-chip, este es un modo de operación versátil puesto que no hay esencialmente limitaciones sobre programas de propósito especial que pueden ser cargados en RAM interna. El programa de arranque (*Boot Loader Program*) está contenido en los 192 bytes de ROM *Bootstrap*. Esta ROM solo está habilitada cuando el microcontrolador es restablecido en el modo de operación especial *Bootstrap* apareciendo un espacio de memoria interna en las localidades $BF40-$BFFF. El programa cargador de arranque usa el SCI (Serial Comunications Interface) para leer programas de 256 bytes dentro de la RAM en las localidades $0000 a $00FF. Después de que el carácter para la dirección $00FF es reservado, el control pasa automáticamente al programa en la localidad $0000 [22].

Modo prueba. Este modo se utiliza para calibrar los sistemas que componen el MCU y solamente es usado por el fabricante.

E.2.2 Puertos de Entrada y Salida [22]

La mayoría de estos puertos tienen más de un propósito, dependiendo del modo de operación o funciones periféricas seleccionadas, tales como el puerto A, D, y E utilizados en esta aplicación. El puerto A puede ser configurado para funciones de contador de captura de entrada (IC), contador de captura de salida(OC) o la función de acumulador.

Puerto A: Está relacionado con el temporizador, puede ser configurado en todos los modos de operación con tres entradas para captura de datos, cuatro

salidas para funciones de comparación y una entrada (PA1) para acumulador de Pulsos.

Puerto B: En modo *single chip*, todas las terminales son usadas como salidas de propósito general; en modo multiplexado expandido se utiliza para direccionamiento.

Puerto C: En modo *simple Chip* se utiliza como puerto de propósito general de Entrada/ Salida, en modo expandido está multiplexado con el bus de datos y la parte baja de las direcciones, sus entradas pueden tener un registro *latch* controlado por la línea de control "STRA".

Puerto D: Es un puerto bidireccional de propósito general y puede ser configurado para obtener dos subsistemas de comunicación serie, sincrono (PD2-PD5) y asíncrono (PD0-PD1) de baudaje programable (Ver apéndice B).

Puerto E: Puede ser usado como puerto de entrada de propósito general o como entradas para el convertidor analógico/dígital hasta 8 canales en los modelos PLCC (*Plastic Leades Chip Carrier*).

E.3 Compilador para el HC11 [21].

La programación en lenguaje de alto nivel se ha convertido en una herramienta indispensable en el desarrollo de sistemas. Para los diseñadores en sistemas digitales es una herramienta muy popular y de gran ayuda, ya que permite la rápida programación de los mismos. Por esta razón, como parte de la extensa gama de herramientas de desarrollo con microprocesadores en general, cotidianamente se lanzan al mercado, ensambladores en lenguaje C, que permiten una programación en lenguaje de nivel intermedio. El objetivo es orientar los diversos lenguajes ensamblador hacia un estandar, la programación en C. Para el MC68HC11, existen ensambladores especiales que convierten el código en C a lenguaje ensamblador e inclusive al formato S19 necesario para las herramientas de evaluación o emulación.

Teniendo en cuenta una sintaxis orientada hacia la arquitectura y organización del microcontrolador, esta clase de programación puede ayudar al programador conocedor del lenguaje C en el desarrollo de sistemas basados prácticamente en cualquier microprocesador que soporte tal característica.

Algunas ventajas que ofrecen los fabricantes de ensambladores cruzados en C son: ambientes integrados de trabajo (editor y ensamblador), múltiples ventajas de edición, macros, programas de demostración, ambiente de depuración y comunicación con las tarjetas de evaluación, etc.

El compilador en cruzado ICC11 (ImageCraft68HC11C)[1] permite traducir el programa de aplicación de lenguaje C, en código de ensamblador para los microcontroladores MC68HC11, además se ajusta al lenguaje ANSI, y contiene los archivos y rutinas básicas necesarias para poder procesar el Archivo Generado en lenguaje C. El compilador cruzado en lenguaje C y desarrollos de medio ambiente para la familia de microcontroladores trabaja sobre PC-DOS/Windows y plataforma Linux [21].

De esta manera se muestra el desarrollo utilizando las herramientas disponibles como son el lenguaje ensamblador, el compilador en C, la tarjeta de evaluación para el mismo, además del PCBUG11 el cual se explica en la siguiente sección.

E.3.1 Programa PCBUG11

El pcbug11 es un programa que permite un fácil acceso a los microcontroladores de la familia M68HC11, examina el comportamiento de los periféricos internos bajo condiciones especificas, monitorea el estado de los registros, manipula datos en memoria, etc. Además nos permite bajar programas en la memoria RAM con extensión S19 generados por el compilador en C y ejecutarlos [21].

E.4 Arquitectura de Desarrollo

Debido a que el desarrollo de nuestras aplicaciones, se realiza en un compilador para el HC11 en C, es necesario el uso de memoria externa RAM, ya que la RAM interna de MCU no es lo suficientemente grande para almacenar el compilador en C. Para el uso de la memoria externa es necesario utilizar al MCU en modo multiplexado expandido, con el cual se tiene la posibilidad de direccionar más memoria o puertos de Entrada-Salida.

Como se muestra en el diagrama de bloques de la figura 5.1, el sistema cuenta con memoria RAM (para el almacenamiento del programa de aplicación), un decodificador, un *latch*, y el MCU.

En el modo multiplexado como ya se mencionó el bus de datos se encuentra multiplexado con la parte baja de las direcciones; para realizar el demultiplexaje se utiliza un circuito tipo *latch*, cuya habilitación proviene de la línea de control del MCU "AS"(address strobe)como se muestra figura E.1. La decodificación utiliza medio ciclo junto con la señal de reloj "E" para el bus de datos y el otro medio ciclo junto con las señales "*AS*" y *R/W* para el *bus* de

[1] Para más información sobre el compilador en C para el HC11 se puede consultar el apéndice C.

direcciones. Teniendo demultiplexado ambos *buses* se puede plantear el mapa de memoria para la arquitectura.

Las características principales de la arquitectura, se enumeran de la siguiente forma:

- Intervalo de conversión de señal de 0 a 5 V
- Tiempo de conversión 16 microsegundos.
- Memoria RAM de 4 Kbytes.
- Velocidad de transmisión de 9600 bauds.

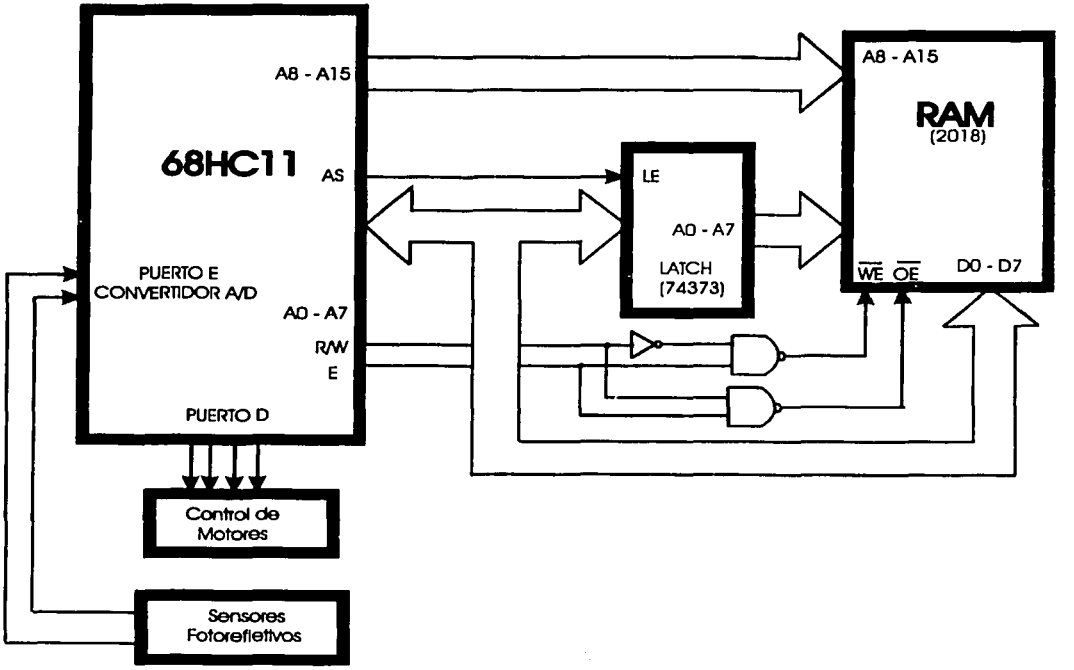


Figura E.1 Diagrama de bloques de la arquitectura diseñada

E.5 Convertidor Analógico/Digital.

El convertidor analógico/Digital esta asignado al puerto E.Las conversiones A/D de 8 bits son exactas dentro de 1 LSB, usando el sistema de reloj de 2 Mhz. Cada conversión es realizada en 32 ciclos de reloj E (2Mhz) del MCU. El bit CSEL del registro OPTION permite la selección de un oscilador interno RC que permite al convertidor A/D ser usado con velocidades de reloj del MCU muy bajas. Un ciclo de conversión típico requiere 16 μs a una frecuencia de 2 Mhz para completarse. Para la utilización del convertidor es necesario configurar los siguientes registros. OPTION y ADCTL (registros de control del convertidor A/D) [22].

Registro OPTIONS

DPU	CSEL	IRQE	DLY	CME		CR1	CRO

ADPU : A/D powerup , proporciona la energía para activar el convertidor.

0 = desactivado
1 = activado.

CSEL: clockselect, selecciona el oscilador de referencia para la conversión.

0 = 2 Mhz
1 = 750 khz

Los bits restantes no son utilizados en el control del convertidor A/D.

Para trabajar a su máxima capacidad se selecciona el oscilador de 2Mhz y se activa le energía para su funcionamiento, por lo cual se configura este registro con un dato $80.

Registro ADCTL

CCF	0	SCAN	MULT	CD	CC	CB	CA

CCF : Bandera de conversión completa, indica la terminación de una conversión.

0 = realizando conversión.
1 = conversión completa.

SCAN : Control de conversión continua, selecciona el tipo de conversión a realizar.

0 = conversión continua.
1 = una sola conversión.

MULT: Control de canal múltiple, selecciona la conversión de uno a cuatro canales.

0 = 1 canal
1 = 4 canales.

CD,CC,CB,CA : seleccionar el canal a convertir.

Cada vez que se requiera los datos de una conversión se debe escribir este registro, después de 16 microsegundos los resultados de la conversión de cuatro canales se tendrán disponibles en los registros ADR1-ADR4 (La dirección de los registros son $1031-$1034 respectivamente).

Apéndice F

Algoritmo en lenguaje C para el MC68HC11

F.1 Implantación del Sistema

El algoritmo desarrollado en lenguaje C para el microcontrolador 68HC11, utilizado para la primera aplicación ,es mostrado a continuación.

```
*/Definición de Constantes para los Conjuntos difusos./*

#define vel_rectas 0X6000 /* 0x08ff*/
#define vel_curvas 0X6000 /*0x57ff */
#define veces  1000
#define punto_muy_oscuro_izq    255
#define pendiente_muy_oscuro_izq  1000/103
#define punto_oscuro_izq      152
#define pendiente_oscuro_izq   1000/30
#define punto_intermedio_izq      121
#define pendiente_intermedio_izq   1000/30
#define punto_claro_izq      75
#define pendiente_claro_izq   1000/46
#define punto_muy_claro_izq    0
#define pendiente_muy_claro_izq  1000/75
#define punto_muy_oscuro_der    255
#define pendiente_muy_oscuro_der  1000/185
#define punto_oscuro_der      170
#define pendiente_oscuro_der   1000/27
#define punto_intermedio_der      143
#define pendiente_intermedio_der   1000/24
#define punto_claro_der      119
#define pendiente_claro_der   1000/24
#define punto_muy_claro_der    0
#define pendiente_muy_claro_der  1000/119
#define punto_muy_poco    0
#define pendiente_muy_poco  1000/2
#define punto_poco      2
#define pendiente_poco   1000/2
#define punto_medio      5
```

```
#define pendiente_medio   1000/2
#define punto_largo       7
#define pendiente_largo   1000/2
#define punto_muy_largo   10
#define pendiente_muy_largo  1000/2
#define t_maximo      10

/*Programa Principal*/

main(void)
{
  int sens_izq,sens_der,mayor;
  long tiempo;
  int izq_muy_oscuro, izq_oscuro, izq_intermedio, izq_claro, izq_muy_claro;
  int der_muy_oscuro, der_oscuro, der_intermedio, der_claro, der_muy_claro;
  int adelante, derecha, izquierda, muy_largo, largo, medio, poco, muy_poco;
  char direccion;
  inicializa();
  init_communications();
  do
  { lee_ad(&sens_izq,&sens_der);/* Lectura de Canales de Convertidor*/


/*Para representar los conjuntos difusos de la señal de entrada fuerón definidos con
la siguiente estructura:/*

 izq_muy_oscuro=pertenece(punto_muy_oscuro_izq,pendiente_muy_oscuro_izq,
                          sens_izq);

 izq_oscuro = pertenece (punto_oscuro_izq,pendiente_oscuro_izq,sens_izq);
izq_intermedio=pertenece(punto_intermedio_izq,pendiente_intermedio_izq.sens_izq);
izq_claro = pertenece (punto_claro_izq,pendiente_claro_izq,sens_izq);
izq_muy_claro =pertenece(punto_muy_claro_izq,pendiente_muy_claro_izq,sens_izq);

der_muy_oscuro=pertenece(punto_muy_oscuro_der,pendiente_muy_oscuro_der,
                         sens_der);

der_oscuro = pertenece (punto_oscuro_der,pendiente_oscuro_der,sens_der);

der_intermedio=pertenece(punto_intermedio_der,pendiente_intermedio_der,sens_de)
;der_claro = pertenece (punto_claro_der,pendiente_claro_der,sens_der);
der_muy_claro=pertenece(punto_muy_claro_der,pendiente_muy_claro_der,sens_de);


/*Calculo de conjunto de Salida.*/
     adelante = 0;
    derecha = 0;
    izquierda = 0;
```

```
    muy_largo = 0;
    largo = 0;
    medio = 0;
    poco = 0;
    muy_poco = 0;

/* Reglas de control definidas*/

 regla (izq_muy_oscuro,der_muy_oscuro, &adelante,&muy_largo);
    regla (izq_oscuro,der_muy_oscuro, &adelante,&largo);
    regla (izq_intermedio,der_muy_oscuro, &adelante,&medio);
    regla (izq_claro,der_muy_oscuro, &izquierda,&medio);
    regla (izq_muy_claro,der_muy_oscuro, &izquierda,&muy_largo);

    regla (izq_muy_oscuro,der_oscuro, &adelante,&largo);
    regla (izq_oscuro,der_oscuro, &adelante,&medio);
    regla (izq_intermedio,der_oscuro, &adelante,&poco);
    regla (izq_claro,der_oscuro, &izquierda,&medio);
    regla (izq_muy_claro,der_oscuro, &izquierda,&muy_largo);

    regla (izq_muy_oscuro,der_intermedio, &adelante,&medio);
     regla (izq_oscuro,der_intermedio, &adelante,&poco);
    regla (izq_intermedio,der_intermedio, &adelante,&muy_poco);
    regla (izq_claro,der_intermedio, &adelante,&muy_poco);
    regla (izq_muy_claro,der_intermedio, &izquierda,&muy_poco);

    regla (izq_muy_oscuro,der_claro, &derecha,&medio);
    regla (izq_oscuro,der_claro, &derecha,&medio);
    regla (izq_intermedio,der_claro, &adelante,&muy_poco);
    regla (izq_claro,der_claro, &adelante,&muy_poco);
    regla (izq_muy_claro,der_claro, &adelante,&muy_poco);

    regla (izq_muy_oscuro,der_muy_claro, &derecha,&muy_largo);
    regla (izq_oscuro,der_muy_claro, &derecha,&muy_largo);
    regla (izq_intermedio,der_muy_claro, &derecha,&muy_poco);
    regla (izq_claro,der_muy_claro, &adelante,&muy_poco);
    regla (izq_muy_claro,der_muy_claro, &adelante,&muy_poco);

;/*Esta etapa se tiene el la señal de defusificación tanto para las variables de
DIRECCION Y TIEMPO DE MOVIMIENTO, donde el método utilizado fué el del
centroide.*/

 /* Defuzificación de la variable de salida Dirección por el método del Centoide */
    mayor = adelante
    direccion = 'a';
    if (derecha > mayor)
    {  mayor = derecha;
        direccion = 'd';
```

```
    }
    if (izquierda > mayor)
    { direccion = 'i';
    }
    /* defuzificacion de la variable de salida Tiempo por el método del máximo  */
    tiempo = muy_poco*punto_muy_poco + poco*punto_poco + medio*punto_medio
+ largo*punto_largo + muy_largo*punto_muy_largo;
    tiempo = tiempo / (muy_poco + poco + medio + largo + muy_largo);
    mueve(direccion,tiempo);
  }while (1);
}

/*Habilitacion de los timer para las señales de control */

inicializa()
{     poke(0x1020, 0x28); /*timers 3 y 4 se limpian en comparación exitosa*/
        poke(0x100C, 0x30); /*oc1 afectar'a al 3 y al 4 */
        poke(0x100D, 0) ; /* timers 3 y 4 adquirirán el valor de cero en cuenta exitosa*/
        pokeword(0x101C);  /* (vel_curvas)*/
        pokeword(0x101A);  /*(vel_rectas)*/
        poke(0x1039, 0x90);  /* enciende el A/D con retardo*/
        bit_clr(0x1028, 0x20) ;  /* desactiva el open-drain del puerto d*/
        poke(0x1009, 0x3C);  /* activa como salidas D2 y D3*/
}

/*Función para la Lectura de Canales del convertidor A/D*/
lee_ad(int *izq,int *der)
{     int status;
        poke(0x1030, 0x10);
        while (((status=peek(0x1030))& 0x80)==0);
        izq=peek(0x1032);  /* Lectura del canal 2*/
        der=peek(0x1033); /* Lectura del canal 3*/
}

;/*Rutina para movimiento de Motores*/;

mueve(dir, tiem)
char dir;
long tiem;
{ long i,j;
  if (dir=='a')
  { poke(0x100D, 0x30);    ; /*Envio de comando movimiento hacia */
    poke(0x1008, 0x18);    ;/*adelante.*/
    putchar('a');
  }
  else if (dir=='d')
  { poke(0x100D, 0x30);    ;/*Envio de comando para giro hacia a la
    poke(0X1008, 0x14);    ;/*Derecha*/
```

```
      putchar('d');
    }
    else if (dir=='i')
    { poke(0x100D, 0x30);     /*Envio de comando para giro hacia a la
      poke(0x1008, 0x28);     ;/* Izquierda*/
      putchar('i');
    }
    putchar('-');
    puthex(tiem);

    putchar('*');
    putchar('*');
    for (j=0;j<veces;j++)
   for (i=tiem;i>=0; i--);
}

/*Función para el cálculo de la función de pertenencia*/

int pertenece(int punto, int pendiente, int sensor)
{ int tmp, derecho, izquierdo;
  derecho = -pendiente*(sensor-punto) + 1000;
  izquierdo = pendiente*(sensor-punto) + 1000;
  if ((izquierdo > 0) && (izquierdo <= 1000))
    tmp = izquierdo;
  else
  { if ((derecho > 0) && (derecho <=1000));
        tmp = derecho;
    else
        tmp = 0;
  }
  return (tmp);
}

/* Función para el cálculo de Inferencia*/

regla(int premisa1, int premisa2, int *accion1, int *accion2)
{ int and;
  and = premisa2;
  if (premisa1 <= premisa2);
    and = premisa1;
  if (and >= *accion1)
    *accion1 = and;
  if (and >= *accion2)
    *accion2 = and;
}
```

Bibliografía

[1] George J. Klir and Bo Yuan. "*Fuzzy set and fuzzy logic: Theory and applications*", Prentice Hall, USA, 1995.

[2] Joseph L. Jones Anita M. Flynn. "*Mobile robots, inspiración to implementation*". A K Peters, Massachusetts, USA, 1993.

[3] Bijoy K. Ghosh, Ning Xi and T. J. Tarn, "*Sensor-Based Planning and control for robotic Sytems*". Academic Press, 1999.

[4] Hao Ying, "*Fuzzy control and modelling*", IEEE PRESS, New York, USA, 2000.

[5] Earl Cox, *"Fuzzy Fundamental"*, IEEE Spectrum, Octubre 1992, pp.58-61.

[6] Bart Kosko, *"Neural Networks and Fuzzy Systems"*, Prentice Hall, USA, 1992.

[7] Lotfi A. Zade, *"Knowledge Representation in Fuzzy Logic", IEEE transactions on knowledge and data Engineering*, Vol. 1 No. 1 Marzo 1989.

[8] Stephen R. Watson,Jonathan J. Weiss and M. Donnell, *"Fuzzy Decision Analysis", IEEE Transactions on systems, man, and Cibernetics*. Vol. SMC-9, No. 1, Enero 1979.

[9] Yasunobu S and Miyamoto S, *"Automatic Train Operation System by Predictive Fuzzy Control"*, Industrial Applications of Fuzzy Control, North-Holland. Aamsterdam, Holland, 1985.

[10] H.J. Zimmerman, *"Fuzzy Set theory and its applications"*, 2nd revised edition, Kluwer- Publishing,Hingham.MA, USA,1991.

[11] E. H Mamdani,S. Assilian, *"An expriment in linguistic synthesis with a fuzzy logic controller", International Journal Man-Machine Studies*, Vol. 7, No. 1 , 1975, pp. 116-132.

[12] Viot, Greg, *"Fuzzy Logic in C", Dr Dobb's Journal*, Febrero 1993, pp.40-50.

[13] M. Sugeno, K. Murakami, "*An experimetal study on fuzzy parking control using a model car*", *Industrial Applications of Fuzzy Control*, North-Holland, Amsterdam,Holland,1985.

[14] Daniel McNeil, Paul Freiberger. "*Fuzzy Logic*",1ª edition. pp. 14-44.

[15] Lotfi A. Zadeh ,"*Fuzzy Logic*", *IEEE Spectrum* ,Abril 1988 pp. 83-94.

[16] Lee,Chuen Chien,"*Fuzzy Logic in Control Systems: Fuzzy Logic Cotroller, Part 1 and II*" ,*IEEE Transactions on systems, man and cybernetics*., Vol. 20 No2, 1990, pp.404-435.

[17] P. Sánchez y Beltrán. Sistemas Expertos, "*Una metodología de programación*".Macrobit Corporation, Miami Florida, USA 1990.

[18] Nick, Infelise; "*A clear Vision of Fuzzy Logic*", *Control Engineering*, Julio 1992. pp 28-30.

[19] Josep Giarratano Gary Riley, "*Expert Systems, Principles and programming*". PWS Publishing Company, Massachussets, USA, second edition, 1994.

[20] Rodney A. Brook,."*New approaches to robotics*", Science, Septiembre 1991.

[21] ImagenCraft 68HC11 C Compiler and Development Enviroment, User´s Manual,1997.

[22] M68HC11 E series; Technical Data Motorola, USA, 1991.

[23] M68HC11EVBU, Universal Evaluation Board User´s Manual Motorola, 1997.

[24] Referencia Manual, M68HC11 Motorola, USA, 1991.

[25] Real World Interface, New Hampshire, USA. *BeeSoft User´s Guide and Software Reference*,1997.

[26] Marco Antonio Morales Aguirre "*Control de un robot móvil usando Redes Neuronales y Sistemas Expertos*", Septiembre 1998.

[27] Jesús Savage-Carmona Marco Morales-Aguirre. Herramientas gráficas en Clips para simulación de robots. Ponencia presentada en el coloquio:

La Investigación en la Facultad de Ingeniería, la versión de CLIPS modificada está disponible en ftp://odin.fi-b.unam.mx,1997.

[28] Maiers, J. and Sherif, Y. S., *"Aplications of Fuzzy set Theory"*. IEEE trans. Syst.,Man,Cybern., Vol. SMC-15, No. 1, pp. 175-189, 1985.

[28] C.J. Harris, C. G. Moore and M, Brown. Intelligent Control *"Aspects of fuzzy Logic and Neural Nets"*, World Scientific,1992.