TECNOLÓGICO DE MONTERREY
Biblioteca
Campus Ciudad de México

TECNOLÓGICO DE MONTERREY®

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Ciudad de México

División de Ingeniería y Arquitectura

Ingeniería en Mecatrónica

Departamento de Mecatrónica
Proyectos de Ingeniería Mecatrónica II (MR00039)

Controlador Neuro-Difuso para Aplicaciones Colaborativas Simples en Robótica

AUTORES: Dejanira Araiza Illan
Hiram Eredín Ponce Espinosa
Ximena Rodríguez de la Vega Olivares
Fernando Alvarez Rivas

ASESOR: Dr. Pedro Ponce Cruz

PROFESOR: Dr. Raúl Crespo Saucedo

México D.F. noviembre de 2007

Índice

CAPÍTULO 1

INTRODUCCIÓN

En la actualidad, la tecnología se ha diversificado de forma sorprendente abarcando todas las áreas del conocimiento y la vida cotidiana. Una de las aplicaciones tecnológicas que ha tenido mayores avances en el último siglo es la robótica, junto con las teorías matemáticas que permiten la modelación y el control de los sistemas dentro de la misma. A pesar de que el término *robot* proviene de la palabra checoslovaca que significa trabajo o trabajador, no fue sino hasta los 50's que surgió la robótica como el estudio de máquinas programables involucrando el aspecto mecánico, electrónico, de control, computación e inteligencia artificial [1]. El proyecto presentado en este documento gira en torno a esta área, con aplicación específica a pequeña escala, ya que los micro-robots autónomos permiten realizar desarrollos mecánicos, eléctricos y electrónicos con fines educativos y de investigación.

1.1 Antecedentes

A lo largo de la carrera de Ingeniería en Mecatrónica, se adquieren numerosos conocimientos referentes a distintas áreas, entre las que se encuentran el diseño de mecanismos y estructuras, el modelado matemático de sistemas, el uso de diversos lenguajes de programación computacional, el uso de circuitos integrados y procesadores para el control y

monitoreo de procesos, y el uso de simuladores para verificar los desarrollos llevados acabo, por mencionar algunos ejemplos. El diseño y funcionamiento de un micro-robot autónomo implica la aplicación de todo el aprendizaje llevado a cabo en los semestres anteriores para resolver problemas específicos, por lo que se consideró la mejor alternativa de enfoque dentro de un área interdisciplinaria.

La idea de desarrollar e implementar la colaboración entre micro-robots móviles significa, además de la aplicación de las herramientas de la inteligencia artificial para el beneficio de la situación académica del país, una nueva forma de realizar tareas de la vida cotidiana del hombre para evitar situaciones que pongan en riesgo la integridad física, como el rescate y el manejo de sustancias o dispositivos peligrosos, además de algunas otras tareas que mejoren la calidad de vida de las personas [1], como la navegación autónoma de una silla de ruedas.

1.2 Definición del problema

La inteligencia artificial comprende herramientas poderosas para la solución de problemas que se derivan de distintas áreas de la vida cotidiana del hombre. Desde la determinación del dinero necesario en cada cajero automático, dependiendo de la demanda del mismo, hasta la determinación de las enfermedades más comunes de los niños mexicanos para poder conocer la demanda y cubrir la atención en el sector salud, pasando por la toma de decisiones en la micro, pequeña y mediana empresa, el desarrollo del software, hardware y demás interfaces para contribuir al diseño de alternativas de solución de problemas implica al final el impulso al desarrollo del país. En la actualidad, algunos proyectos en estos sentidos ya se llevan a cabo [2].

Por lo anterior, la generación de herramientas genéricas y globales para el desarrollo de algoritmos y programas es una gran área de oportunidad para que el trabajo académico realizado en la materia de Proyectos de Ingeniería Mecatrónica contribuya al desarrollo de la

comunidad del Campus Ciudad de México, respecto a la preparación de profesionales en el ramo tecnológico que ayuden a impulsar el desarrollo del país, como afirma la Misión de la institución [3].

Nuestro proyecto se enfoca al aprendizaje respecto al uso y desarrollo de herramientas sobre el control inteligente y la inteligencia artificial, como ya lo han hecho otros grupos de estudiantes en el Campus [4], [5]. Por otra parte, el proyecto está también enfocado a la difusión del uso y las aplicaciones de la micro-robótica combinada con la inteligencia artificial dentro de en nuestra comunidad estudiantil, para intentar disolver la falta de interés por la investigación tecnológica entre los compañeros, ya que somos parte de una sociedad que consume productos tecnológicos de punta, pero no es partícipe ni del diseño de los mismos, ni de su posible mejora.

1.3 Objetivos

El objetivo del proyecto comprende lograr implementar un controlador neuro-difuso para ser aplicado en tareas que involucren el movimiento colaborativo de un conjunto de micro-robots que intercambian información, mediante el uso y desarrollo de algoritmos. Para lograr esto, es necesario lo siguiente:

- Diseñar e implementar un controlador neuro – difuso basado en trabajos realizados anteriormente, para lograr la navegación autónoma de los micro-robots en un área, evadiendo los obstáculos fijos.
- Implementar intercambio de información entre robots.
- Diseñar e implementar una tarea simple que implique que ejemplifique el trabajo colaborativo entre robots.

Adicionalmente, se plantearon distintos objetivos específicos:

- Diseñar e implementar el controlador y los algoritmos necesarios para la tarea colaborativa en MATLAB y LabVIEW.

- Evaluar el uso de MATLAB y LabVIEW para el control y la inteligencia artificial.
- Diseñar una tarea colaborativa simple que puedan desempeñar los robots.
- Realizar las adaptaciones pertinentes a la estructura actual de los robots para adecuarlos a la tarea colaborativa elegida.
- Generar un nuevo diseño físico y de control para un robot de tres ruedas, dos diferenciales y una libre, que se adapte a las necesidades de presupuesto y desempeño de proyectos académicos.

1.4 Justificación

El proyecto se ha realizado con base en la inquietud tecnológica respecto a las aplicaciones del control inteligente y la inteligencia artificial a la micro-robótica, lo cual involucra el uso del conocimiento adquirido a lo largo de la carrera respecto a las áreas involucradas en la robótica, desde el diseño mecánico de la estructura de un robot hasta la programación de los algoritmos que controlarán su funcionamiento según los objetivos planteados. Los resultados del proyecto podrán ser extrapolados a otros trabajos que involucren el uso de controladores neuro-difusos o algoritmos para tareas colaborativas simples y que requieran de la navegación autónoma, para contribuir en una cadena de conocimiento que ayude al desarrollo tecnológico y científico de nuestra comunidad.

1.5 Estado del arte

Para lograr una visión global general del panorama actual de la inteligencia artificial, así como de la micro-robótica aplicada principalmente a la industria y de los métodos modernos de control que permiten lograr una mayor eficiencia y una mejor respuesta ante los requerimientos del medio para el funcionamiento de los robots, se realizó una búsqueda de información y distintas alternativas relacionadas. Asimismo, para la implementación física también se

buscaron distintas opciones de uso frecuente y conocido disponibles en el mercado y accesibles para trabajar con las mismas como base.

1.5.1 Inteligencia artificial

La inteligencia artificial ha sido definida como la parte de las ciencias computacionales que involucra procesos de manipulación de símbolos para producir acciones inteligentes. Una acción inteligente significa un acto o decisión orientado a una meta, a partir de una cadena entendible o un análisis simbólico y distintos pasos de razonamiento [6]. Las funciones que han sido investigadas y desarrolladas dentro de la inteligencia artificial [6] son las siguientes:

- Entendimiento del lenguaje natural
- Entendimiento y procesamiento de imágenes
- Sistemas expertos que codifican la experiencia humana para realizar acciones o responder preguntas
- Adquisición de conocimiento y representación
- Búsqueda heurística
- Razonamiento deductivo
- Planeación y monitoreo

1.5.2 Robótica industrial

En cuarenta años, la evolución de los robots se ha dado principalmente en la industria. Las aplicaciones en las áreas productivas se han diversificado, debido a la flexibilidad de adaptación de los robots mediante la reprogramación y los cambios en los equipos periféricos. Los robots industriales comprenden en su totalidad a los manipuladores multifunción reprogramables, con varios grados de libertad, capaces de manipular materias, piezas, herramientas o dispositivos especiales, siguiendo trayectorias programadas para ejecutar sus tareas [7].

Existen cuatro tipos de robots típicamente usados en la industria: secuencial, de trayectoria controlable, adaptable y telemanipulado, los cuales a su vez pueden ser clasificados según su evolución como se muestra en la Tabla 1.1 [7].

Generación	*Nombre*	*Tipo de control*	*Grado de movilidad*	*Usos frecuentes*
1ª – 1982	Pick and place	Fin de carrera, aprendizaje	Ninguno	Manipulación, servicio de máquinas
2ª – 1984	Servo	Servo control, trayectoria continua, programación condicional	Desplazamiento por vía	Soldadura, pintura
3ª – 1989	Ensamblado	Servos de precisión, visión, tacto, programación off-line	Guiado por vía	Ensamblado, rebabeo
4ª – 2000	Móvil	Sensores inteligentes	Patas, ruedas	Construcción, mantenimiento
5ª – 2010	Especiales	Inteligencia artificial	Andante, saltos	Militar, espacial

Tabla 1.1 Evolución de la robótica industrial [7]

En la actualidad los robots se utilizan en la industria de manera generalizada, siendo un componente indispensable en una gran parte de los diferentes procesos de manufactura. La industria que se ha beneficiado principalmente de los robots es la automotriz. Algunas de las aplicaciones definidas son las siguientes [7]: manipulación en fundición, moldeo de plásticos, tratamientos térmicos, forja y estampación; soldadura de arco, por puntos, por gas, por láser; aplicación de pintura, adhesivos y secantes; mecanización; carga y descarga de máquinas; corte mecánico, rectificado, rebabeo y pulido; corte láser, con chorro de agua; montaje mecánico, inserción, unión por adhesivos, unión por soldadura; medición, inspección, control de calidad; manipulación de materiales; formación, enseñanza e investigación.

1.5.3 Micro-robótica

A principios de los 90's la reducción del tamaño de los componentes electrónicos y el uso de los microchips permitieron el desarrollo de sistemas de manufactura microscópicos. Estos avances permitieron la disminución del peso, volumen, costo y consumo energético de los robots, para realizar la programación de tareas simples en entornos controlados [8].

El desarrollo actual de la fabricación a micro escala ha propiciado la posibilidad de creación de componentes pequeños y precisos, tal es el caso de los sensores, controladores, motores y baterías, los cuales permiten la creación de micro-robots que realizan tareas complejas en un microentorno. Estos robots son capaces de realizar tareas con la posibilidad de reprogramación, con grados de adaptabilidad y control remoto. Sin embargo, están diseñados para emular capacidades humanas en una escala menor, con rapidez y precisión [8].

La estructura básica de un micro-robot puede apreciarse en la Figura 1.1. En general, se habla de cuatro capas principales, que comprenden el diseño físico, la interacción con el medio o reacción, el sistema de control y la inteligencia del robot, respectivamente [8].

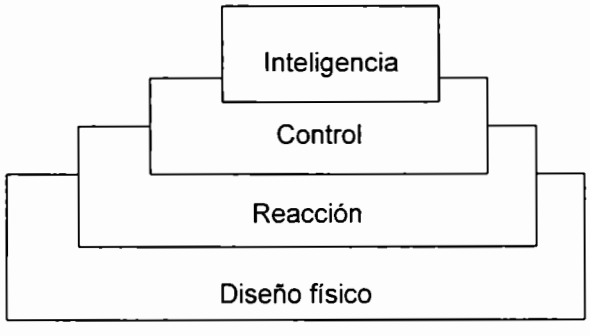


Figura 1.1 Esquema de la estructura de un micro-robot [8]

1.5.4 Colaboración entre robots

La colaboración entre robots significa la presencia de metas compartidas, comunicación sobre el estado del ambiente local interno y externo, así como la acción coordinada para lograr

alcanzar las metas planteadas. Cada agente robótico del sistema contribuye a fortalecer el equipo [9].

La colaboración de robots se ha implementado en las siguientes áreas:

- Robots móviles de vigilancia contra intrusos [10]
- Líneas de ensamblado, pintura, reparación y manufactura [11]
- Brazos robóticos [12]
- Entrega y manipulación de materiales [9]
- Teleconferencias [9]
- Entretenimiento [9]
- Minería [9]
- Limpieza [9]

1.5.5 Lógica difusa

Las bases teóricas de la lógica difusa fueron enunciadas por Lotfi Zadeh en 1965, mediante la definición de un conjunto difuso. La lógica difusa es una extensión de la teoría clásica de conjuntos. Un conjunto difuso es una clase de objetos con grados de pertenencia continuos, asignados entre 0 y 1 [13].

Después del surgimiento de la lógica difusa las aplicaciones al análisis de sistemas complejos y procesos se hicieron notables, ya que es una herramienta que permite el uso de variables lingüísticas en conjunto con las numéricas, y las relaciones entre los conjuntos difusos pueden lograrse mediante enunciados condicionales, e incluso si las relaciones son complejas es posible emplear algoritmos. En la actualidad algunas de las aplicaciones comprenden controladores difusos, una extensión de sistemas expertos, permitiendo la modelación y el control de procesos de diversos tipos, principalmente sistemas no lineales y cuyos parámetros cambian a cada instante [14].

1.5.6 Redes neuronales

El concepto de las redes neuronales surge a partir de la observación del funcionamiento de las neuronas en el cuerpo humano para la transmisión y el procesamiento de la información, siendo aplicadas como una herramienta para lograr emular el pensamiento y aprendizaje del cerebro dentro de la inteligencia artificial. Una red neuronal está compuesta por neuronas interconectadas entre sí y agrupadas en capas, con diversas entradas y salidas, las cuales son entrenadas para reaccionar de una manera deseada [15].

Algunos de los primeros modelos son el Perceptrón, y las redes ADALINE y MADALINE. Sin embargo, actualmente se manejan redes complejas que implican interconexiones de diversa índole con retroalimentación o relaciones más sofisticadas, y los modelos antiguos evolucionaron hasta convertirse en redes complejas como las de propagación hacia atrás (*Backpropagation*), de Hopfield, ART, de Kohonen, de contra-propagación (*Counterpropagation*), las máquinas de Boltzmann y de Cauchy, entre otras [15].

En la actualidad, las redes neuronales son ampliamente utilizadas para diversas aplicaciones que van desde el reconocimiento de patrones, las bases de datos, el control de robots, y el filtrado de señales, hasta la toma de decisiones. En general, las redes neuronales se emplean en conjunto con otras técnicas de control, brindándoles, entre otras ventajas, el aprendizaje adaptivo, la auto-organización, la tolerancia a fallos, la operación en tiempo real, y una fácil implementación tecnológica [15].

1.5.7 Combinaciones de métodos

Debido a las ventajas combinadas que es posible obtener a partir de la fusión de diversas técnicas computacionales y de análisis matemáticos, surgen diversas formas de aplicar el control moderno al control clásico. Asimismo, de una fusión entre la lógica difusa y las redes neuronales es posible obtener controladores difusos que cuentan con aprendizaje. Algunos ejemplos de técnicas que incorporan diversas técnicas son: redes *back-propagation*

con lógica difusa, algoritmos genéticos, mínimos cuadrados ortogonales, tablas de búsqueda [16].

1.5.8 Intercomunicación

Existen diversas alternativas para lograr la comunicación entre dos dispositivos, ya sea a través de cables o mediante el uso del espectro de radiofrecuencia. En el caso de los micro-robots móviles, resulta más conveniente el uso de dispositivos que permitan el libre movimiento del robot en el ambiente que lo rodea, por lo que el uso de radiofrecuencias es una alternativa con más ventajas que el uso de cables. No obstante, para ambos métodos de comunicación existen diversos protocolos que se han desarrollado para que la tecnología sea compatible.

En la actualidad, las comunicaciones vía inalámbrica se emplean diariamente para hacer conexiones a Internet y para el monitoreo de sistemas. La variedad de las radiofrecuencias que pueden manipularse para establecer comunicaciones ha generado protocolos para estandarizar el funcionamiento de los sistemas. Algunos de éstos, comercialmente famosos, pueden apreciarse en la Tabla 1.2 a continuación.

Protocolo	*Velocidad de transmisión de datos*	*Distancia máxima*	*Frecuencias empleadas*
Bluetooth IEEE 802.15.1	720 kb/s	10 a 100 m	2.4 a 2.48 GHz
Wi-Fi IEEE 802.11a	54 Mb/s	300 m	2.4 – 5 GHz
Wi-MAX IEEE 802.16	75 Mb/s	48 km	2-11 GHz

Tabla 1.2 Resumen de protocolos de comunicación inalámbrica [17]

1.6 Áreas de desarrollo

Para la realización del proyecto, ha sido fundamental la adquisición continua y paulatina de conocimientos y entrenamiento en diversas áreas del conocimiento, relacionadas

directamente con el área matemática y tecnológica de la inteligencia artificial aplicada a la robótica:

1. Lógica difusa
    a. Teoría matemática y justificación en el modelado de sistemas no lineales, variables en el tiempo
    b. Controladores difusos
    c. Métodos de sintonización de funciones de pertenencia y matriz de reglas difusas
2. Mejoramiento del control difuso
    a. Algoritmos que permiten sintonizar la base de reglas de un controlador difuso para mejorar el comportamiento del mismo
    b. Diseño y aplicación de algoritmos de búsqueda Tabú para encontrar las mejores reglas difusas
3. Lenguaje y estructura de la programación de distintos dispositivos electrónicos para controlar procesos
    a. Microcontroladores con sistema mínimo integrado y programación en lenguajes de alto nivel (Basic Stamp 2)
4. Uso de software especializado para el control de procesos
    a. Uso avanzado de MATLAB y LabVIEW
    b. Adquisición de datos de los dispositivos de control hacia los programas del procesamiento de los algoritmos
5. Uso de sensores y sistemas de adquisición de datos
    a. Sensores analógicos básicos
    b. Sensores y adquisición de señales avanzados
6. Aprendizaje y algoritmos colaborativos
    a. Redes neuronales aplicadas
    b. Revisión de algoritmos desarrollados y publicados

A futuro, se pueden presentar diversas áreas de oportunidad para aplicar los algoritmos desarrollados e implementados a lo largo del proyecto. Algunas de estas áreas son las siguientes:

- Industria de la manufactura, para aumentar la eficiencia de los sistemas de producción.
- Industria aeroespacial, con el fin de optimizar trayectorias tanto en vehículos espaciales como el trabajo colaborativo de robots para la exploración.
- Ingeniería biomédica, para la exploración óptima y remota del cuerpo humano.
- Rescate y salvamento, así como algunas tareas peligrosas para el ser humano en situaciones de riesgo de su integridad física.
- Mantenimiento de instalaciones en general, limpieza y construcción.

En general, las aplicaciones se resumen a un marco general de modelos adaptivos que permiten el empleo de controladores para solucionar tareas repetitivas y/o peligrosas para el ser humano, mediante robótica confiable en sus funciones programadas de manera autónoma.

CAPÍTULO 2

MARCO TEÓRICO

El marco teórico presentado a continuación ha sido dividido en dos secciones principales, la primera correspondiente al controlador neuro-difuso propuesto para el proyecto, y la segunda referente a los métodos de búsqueda heurísticos revisados para la selección de uno que permitiera encontrar mejores soluciones para la matriz de asociación difusa o FAM.

2.1 Controlador neuro-difuso

La primera fase del proyecto ha girado en torno a lograr el movimiento autónomo de los robots empleados a lo largo del mismo, para lo cual se recopilaron los trabajos realizados previamente en otros proyectos previos respecto a un sistema de navegación neuro-difuso basado en un controlador que contempla series trigonométricas y agrupamientos difusos, el cual permite un movimiento autónomo con evasión de obstáculos, mediante el entrenamiento que proporciona una red neuronal en combinación con las propiedades del uso de la lógica difusa para el control. La recopilación de información se empleó para programar posteriormente diversas estructuras genéricas base de un controlador neuro-difuso como el que se propone en este capítulo, después utilizadas para desarrollar el controlador neuro-difuso específico del proyecto.

2.1.1 Sistema de navegación

Para lograr el movimiento de los robots, se propuso la topología para el control de la navegación autónoma de los robots mostrado en la Figura 2.1, el cual contempla un controlador neuro-difuso basado en series trigonométricas [18] y el uso de una interfaz de comunicación entre el robot y el sistema de procesamiento de los algoritmos correspondientes, que permiten la evasión de obstáculos estáticos y la adaptación del control según la interacción entre el robot y el medio ambiente.

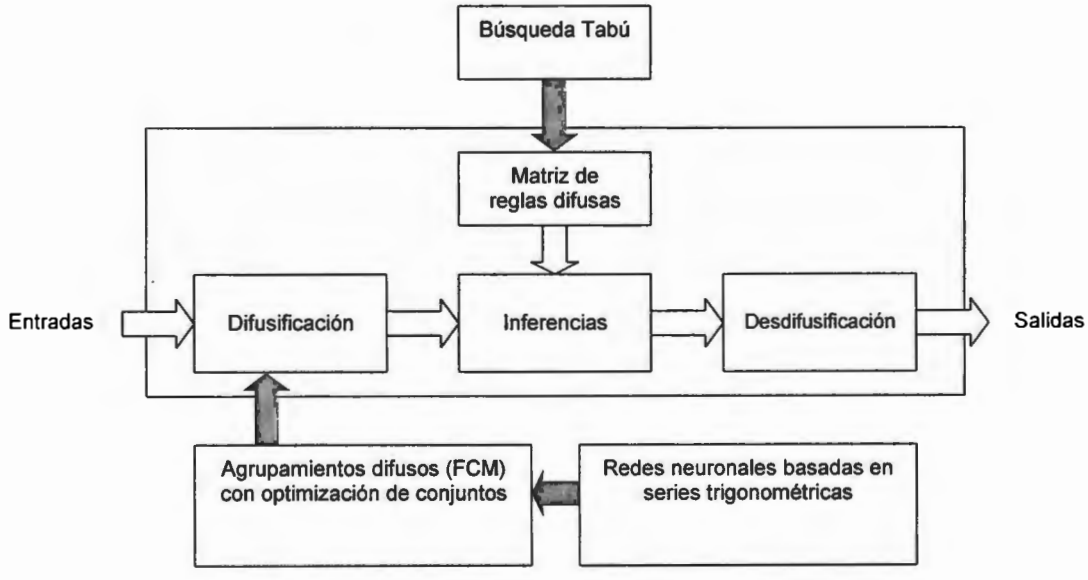


Figura 2.1 Topología propuesta para el controlador neuro-difuso de navegación autónoma [18]

Para el caso específico del proyecto, debido a que el robot es un sistema de entradas múltiples y salidas múltiples, es posible subdividir el controlador total en dos controladores neuro-difusos y así simplificar las matrices de reglas difusas: uno que se encarga de la evasión de obstáculos y el otro del control del movimiento de los servomotores del robot a partir de la posición inicial y la deseada. Esto puede apreciarse esquemáticamente en la Figura 2.2.

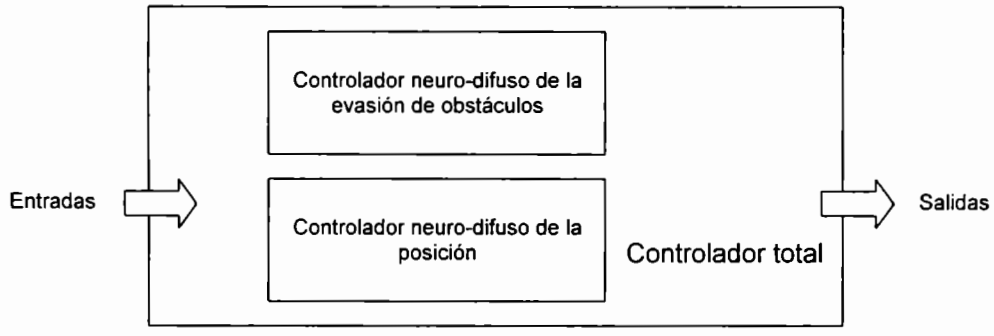


Figura 2.2 División del controlador total

2.1.2 Controladores difusos

A lo largo del desarrollo e implementación de la lógica difusa para el control moderno, se han desarrollado tres tipos de controladores difusos: directos sin optimización, directos con optimización e híbridos. Por otra parte, éstos también pueden agruparse en auto-organizados, con auto-aprendizaje y modeladores, dependiendo principalmente de la estructura matemática empleada y otras herramientas empleadas dentro de la inteligencia artificial [19].

Un controlador difuso estándar está conformado por las siguientes partes [19], según se muestra en la Figura 2.3:

- **Difusificador:** establece una relación entre la entrada x no difusa y diversos conjuntos difusos. Se emplean diversas estrategias para realizar las relaciones, entre las que se encuentran:
  - Singleton: uso directo de los valores no difusos de entrada
  - No singleton: las entradas se traducen a valores difusos a través de funciones exponenciales tipo campana de Gauss, triangulares, trapezoidales, entre otras.
- **Base de reglas difusas:** combina los conjuntos difusos de la entrada y asigna un conjunto difuso para la salida. Frecuentemente se emplean asociaciones lógicas verbales, empleando conjunciones de la forma *Si... entonces*. Las reglas se pueden

representar como una tabla o por medio de matrices (memorias asociativas difusas o FAM).

- **Dispositivos de inferencia:** interpretan las reglas de la base, por medio de operaciones realizadas con las funciones de pertenencia. Las más usadas son: regla del mínimo, regla del producto, regla aritmética, regla máximo-mínimo, regla booleana, regla de Goguen, entre otras.
- **Desdifusificador:** transforma un conjunto difuso en un valor no difuso de salida. Existen diversos métodos para realizar la transformación, siendo los más empleados el método del máximo, el centroide de un área y la media de centroides, por mencionar algunos. El método del centroide o centro de un área [19] arroja un valor final de la siguiente forma, continua o discreta:

$$z_o = \frac{\int_a^b zC(z)dz}{\int_a^b C(z)dz}, \quad z_o = \frac{\sum_{j=1}^{n} z_j C(z_j)}{\sum_{j=1}^{n} C(z_j)}. \tag{1}$$

Siendo C un conjunto difuso, y $[a,b]$ el intervalo que contiene al conjunto C.

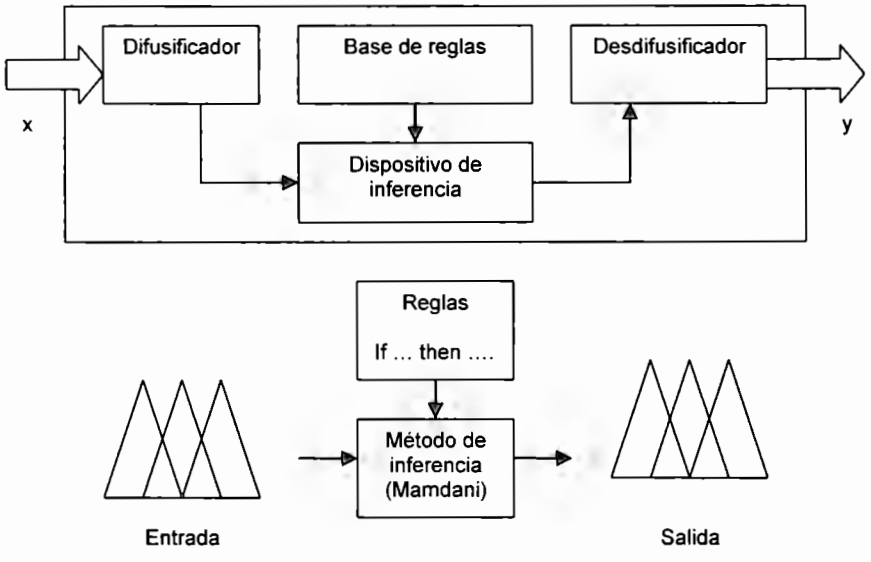


Figura 2.3 Controlador difuso estándar, en dos representaciones [19]

Para la realización de los procesos abarcados por el controlador difuso, existen diversos modelos que han sido desarrollados en base a los fundamentos matemáticos de la lógica difusa, y que comprenden todas las etapas del controlador. Los más empleados en la actualidad, debido a su funcionamiento y versatilidad en los procesos, comprenden el de Mamdani, el de Larsen, el Tagaki-Sugeno-Kang (TSK) y el Tsukamoto [19].

El modelo Mamdani está definido [19], dado un conjunto de reglas "Si x es A_i entonces y es B_i" $i = 1,...,n$ donde $x = (x_1, x_2, ..., x_k)$, por la siguiente combinación:

$$R(x, y) = \bigvee_{i=1}^{n} \left(A_i(x) \wedge B_i(y) \right). \tag{2}$$

Para cada k en $x = (x_1, x_2, ..., x_k)$ esto resulta en un conjunto difuso R_x definido por:

$$R_x(y) = \bigvee_{i=1}^{n} A_i(x) \wedge B_i(y). \tag{3}$$

Teniendo en consideración el conjunto de reglas expandido R_i : Si A_{i1} y A_{i2} y ... y A_{ik} entonces B_i, $i = 1,2,...,n$, R_x se define como:

$$R_x(y) = R(x_1, x_2, ..., x_k, y) = \bigvee_{i=1}^{n} \left(A_{i1}(x_1) \wedge A_{i2}(x_2) \wedge ... \wedge A_{ik}(x_k) \wedge B_i(y) \right). \tag{4}$$

2.1.3 Controladores neuro-difusos

Los controladores neuro-difusos aplican la combinación de los controladores difusos convencionales con las redes neuronales para la obtención de las entradas y/o las salidas. En el caso específico de la aplicación de las redes neuronales a la salida, se tiene lo siguiente [18]:

Las inferencias difusas de la forma *Si* x *es* B *entonces* y *es* C considerando que la salida será una red neuronal, se pueden expresar como:

$$Si\ x_1\ es\ B_1\ y\ x_2\ es\ B_2\ y ...\ x_n\ es\ B_n\ entonces\ y = g(*). \tag{5}$$

Donde $y = g(*)$ puede ser por ejemplo un conjunto de funciones trigonométricas en el caso de las redes neuronales basadas en éstas [18], o alguna otra función que describa a las redes neuronales.

2.1.4 Agrupamientos difusos para encontrar funciones de pertenencia

La búsqueda de estructuras o patrones en la información ha sido fundamental para lograr la automatización e inteligencia artificial de las máquinas, para emular el comportamiento humano de aprendizaje y desempeño de tareas. Para llevar a cabo el reconocimiento de patrones en datos provenientes del medio es necesaria la siguiente metodología básica propuesta en [20]:

- Recolectar los datos del medio (a través de sensores en el caso de un robot móvil)
- Proponer las estructuras que indican relaciones entre variables
- Establecimiento de reglas o algoritmos que formalicen las propuestas de estructuras realizadas anteriormente
- Analizar propiedades de las estructuras, como linealidad, continuidad, estabilidad, etc.
- Clasificar los resultados mediante un entrenamiento
- Hacer pruebas y comparar los resultados con otros modelos

Un resumen de la metodología que se sigue para el análisis de datos numéricos y la división de los mismos en clusters o agrupamientos se muestra en la Figura 2.4, contemplando la lógica difusa para realizar los procedimientos correspondientes detallados posteriormente. La finalidad del análisis de clusters es encontrar conjuntos o subgrupos con elementos lo más similarmente posible, lo cual depende del criterio empleado. La clasificación de los datos, posteriormente al análisis de clusters, implica encontrar particiones difusas para cada punto o dato en el espacio analizado, lo cual es conveniente llevar a cabo mediante redes neuronales [20].

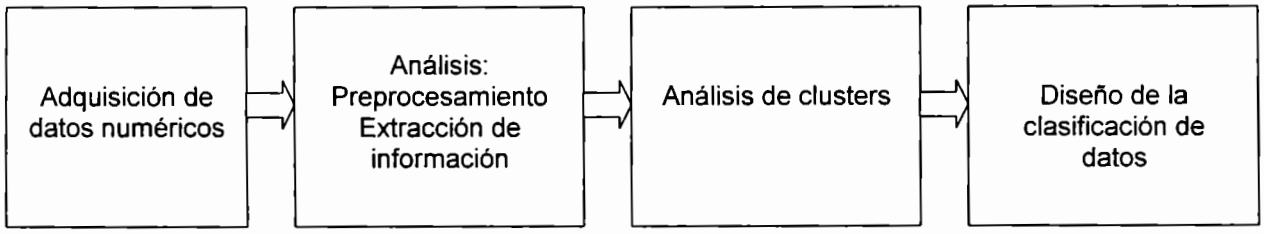


Figura 2.4 Análisis de datos numéricos para encontrar patrones [20]

En nuestro caso particular, es necesario que el robot pueda adaptar y sintonizar las funciones de pertenencia de la difusificación de acuerdo a su percepción del medio ambiente que lo rodea. El método de agrupamientos difusos (*Fuzzy Cluster Means*) permite agrupar los puntos en un número c de conjuntos, asignando un grado de pertenencia de cada elemento en un conjunto c respecto a todos los conjuntos definidos [18].

Teniendo un conjunto de n elementos $X = \{x_1, x_2, ...x_n\}$ que queremos dividir en c conjuntos representados por el mismo centro v de cada conjunto: $V = \{v_1, v_2, ...v_c\}$, se plantea como función objetivo para la división la mínima suma cuadrática del error de distancia de los puntos al centro, la cual se define matemáticamente como

$$J = \frac{1}{2}\sum_{x=1}^{N}\sum_{i=1}^{c}\mu_{x,i}^{m} d^2(z_x, v_i) \ . \qquad (6)$$

Donde $\mu_{x,i}$ es un valor de pertenencia del elemento $x = \{1,2,..., N\}$ en el conjunto difuso $i = \{1,2,..., c\}$, v_i es el centroide de cada conjunto difuso; z_x, con $x = \{1,2,..., N\}$, es el conjunto de datos; m es el valor de difusificación; $d^2(z_x, v_i)$ es la distancia euclidiana entre z_x y v_i; N es el número de elementos o datos y c el número de conjuntos difusos[20].

Para obtener los conjuntos difusos primero se plantean c centroides prototipo [20]. Posteriormente, el grado de pertenencia $\mu_{x,i}$ se calcula empleando la siguiente ecuación

$$\mu_{x,i} = \frac{\left(\frac{1}{d^2(z_x, v_i)}\right)^{1/(m-1)}}{\sum_{i=1}^{c}\left(\frac{1}{d^2(z_x, v_i)}\right)^{1/(m-1)}} . \qquad (7)$$

A continuación se calculan los nuevos centroides, a partir de los grados de pertenencia $\mu_{x,i}$ y el conjunto de elementos z_x, con la ecuación

$$v_i^{'} = \frac{\sum_{x=i}^{N} \mu_{x,i}^{m} z_x}{\sum_{x=i}^{N} \mu_{x,i}^{m}} \qquad (8)$$

Con los nuevos centroides $v_i^{'}$ se recalculan los grados de pertenencia $\mu_{x,i}^{'}$ empleando la ecuación (7). Finalmente se compara J calculada con $\mu_{x,i}$ y v_i -valores penúltimos calculados-, con $J^{'}$ calculada con $\mu_{x,i}^{'}$ y $v_i^{'}$ -valores últimos calculados-, para determinar si el error ε es menor a un criterio establecido entre 0 y 1:

$$J - J^{'} \langle \varepsilon . \qquad (9)$$

Si el error es mayor al criterio, $\mu_{x,i}^{'}$ y $v_i^{'}$ pasarán a ser los valores penúltimos, y será necesario repetir el cálculo de los grados de pertenencia $\mu_{x,i}^{''}$ con la ecuación (7) y de los centroides $v_i^{''}$ con la ecuación (8) y la comparación, cuantas veces sea necesario hasta alcanzar el criterio del error ε. El algoritmo puede apreciarse de forma gráfica en la Figura 2.5.

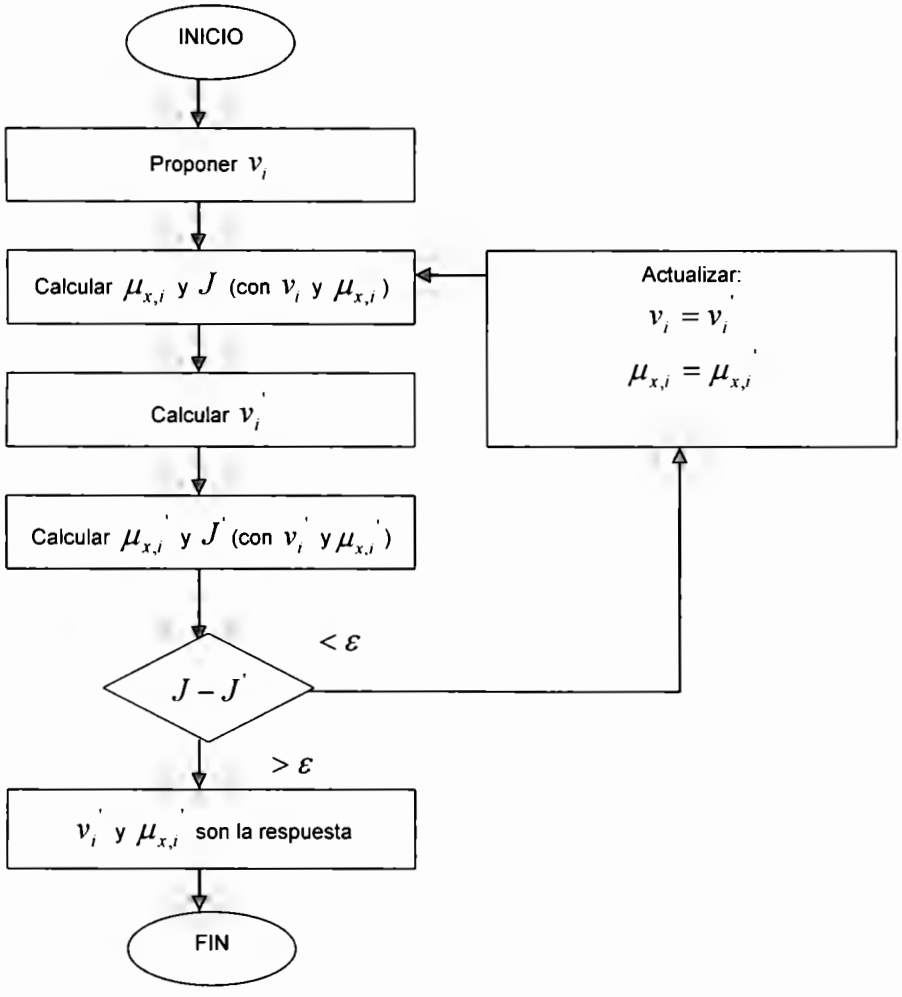


Figura 2.5 Algoritmo para la obtención de agrupamientos difusos

2.1.5 Redes neuronales

Una neurona artificial mimetiza las características principales de una biológica. Cada neurona se caracteriza por un valor o estado de activación $a_j(t)$, una señal de salida y_i, un peso ω_{ji} que modifica las entradas x_i provenientes de un medio externo o directamente de la salida de otras neuronas –red neuronal- . La forma y componentes de la neurona artificial pueden observarse en la Figura 2.6 [16].

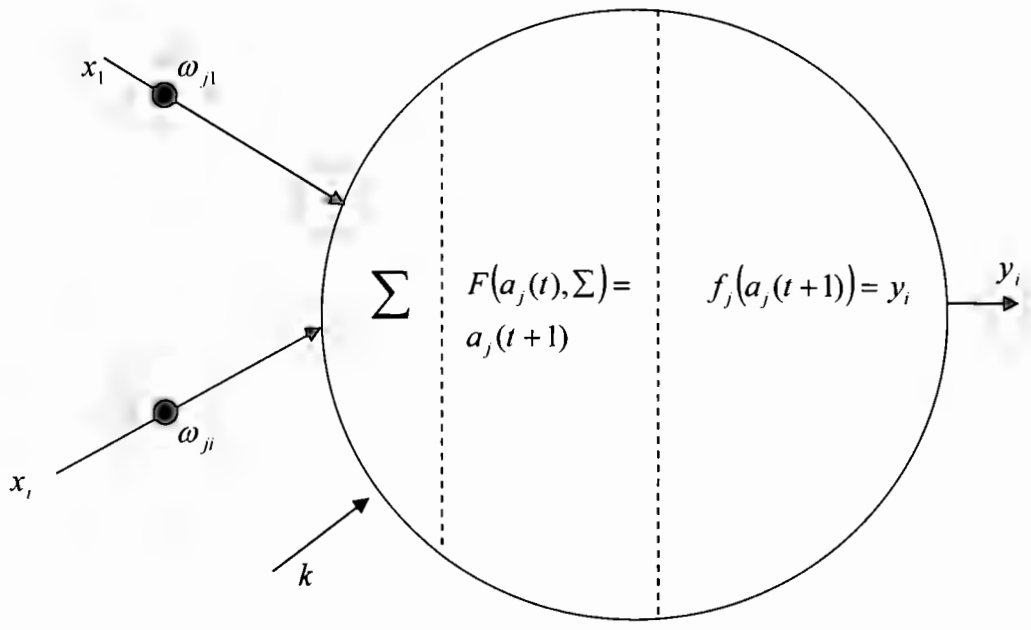


Figura 2.6 Representación esquemática de una neurona [16]

La función de activación F determina el nuevo estado de activación $a_j(t+1)$ de la neurona, considerando la entrada total calculada y el estado anterior de activación $a_j(t)$. La función de salida f_j determina el valor de salida a partir del nuevo estado de activación [16].

Una red neuronal, es decir un conjunto de neuronas, puede ser representada matemáticamente como se muestra a continuación [16]:

$$Y = f\left(\sum_{i=0}^{n} X_i \omega_i\right) \tag{10}$$

Esta ecuación describe que la salida Y será la suma de las funciones de entrada X_i provenientes de otras neuronas multiplicadas por un peso ω_i. Para obtener una salida deseada, es necesario entrenar a la red hasta que los pesos proporcionen las características deseadas en la salida [16]. En el caso del controlador neuro-difuso, se emplea una red para entrenar las funciones de pertenencia obtenidas mediante los agrupamientos difusos.

2.1.5.1 Redes neuronales con funciones trigonométricas

Las series de Fourier pueden ser empleadas para modelar la función de entrada de la red neuronal, así como para obtener los coeficientes de los pesos [21].

Siendo una serie de Fourier definida por la ecuación

$$f(x)=\frac{1}{2}a_0+\sum_{n=1}^{\infty}\left(a_n\cos(n\omega x)+b_n\sin(n\omega x)\right)\ ,\ \omega=\frac{\pi}{l}\ . \tag{11}$$

Donde:

$$a_0=\frac{1}{l}\int_0^{2l}f(x)dx \tag{12}$$

$$a_n=\frac{1}{l}\int_0^{2l}f(x)\cos(n\omega x)dx \tag{13}$$

$$b_n=\frac{1}{l}\int_0^{2l}f(x)\sin(n\omega x)dx\ . \tag{14}$$

Es posible observar el parecido entre la ecuación (10) y la ecuación (11), ambas compuestas por una suma de funciones multiplicadas por un coeficiente, más una constante.

La topología propuesta implica dos capas de neuronas [21]. La primera capa está compuesta por neuronas con una función de activación trigonométrica y un peso que depende directamente de la frecuencia. En la segunda capa se realiza la suma de las funciones trigonométricas multiplicadas por sus pesos más una constante, como se muestra en la Figura 2.7.

Los coeficientes o pesos de la red neuronal pueden ser calculados de manera sencilla si la señal es par o impar. Si se tiene cualquiera de los dos casos es posible realizar una división de las ecuaciones resultantes en la Tabla 2.1 [21].

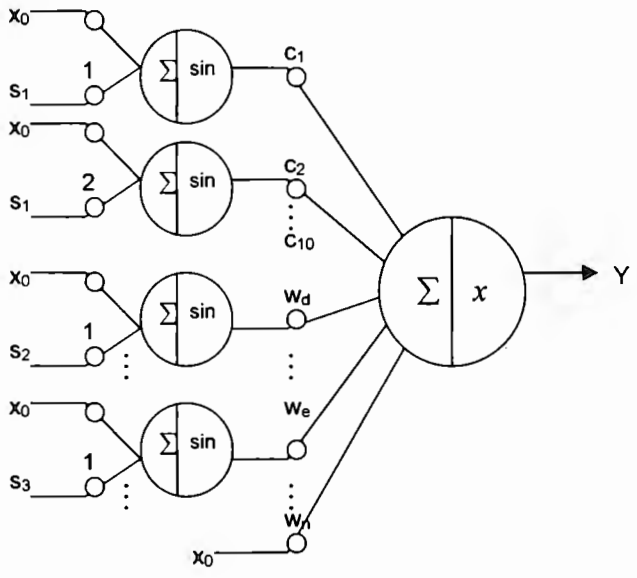


Figura 2.7 Topología de una red neuronal basada en funciones trigonométricas [21]

Impar	Par
Forma:	Forma:
$y(x)=\sum_{n=1}^{p} b_n \sin(n\omega x)$	$y(x)=\frac{1}{2}a_0+\sum_{n=1}^{p} a_n \cos(n\omega x)$
Coeficientes:	Coeficientes:
$b_n=\frac{2}{l}\int_0^{l} f(x)\sin(n\omega x)dx$	$a_n=\frac{2}{l}\int_0^{l} f(x)\cos(n\omega x)dx$
Después de calcular los coeficientes:	Después de calcular los coeficientes:
$y(x)=y_i+\sum_{n=1}^{p} b_n \sin(n\omega(x-x_i))$	$y(x)=\frac{1}{2}a_0+\sum_{n=1}^{p} a_n \cos(n\omega(x-x_i))$

Tabla 2.1 Cálculo de coeficientes [21]

Es posible encontrar un resultado empleando el método de los mínimos cuadrados [21] para encontrar los pesos, a partir de modelar la señal con la siguiente ecuación, buscando que b_n se acerque lo más posible a la función original

$$g(x) = b_0 + \sum_{n=1}^{p} b_n \sin(n\omega x)\,. \tag{15}$$

Con esto, se calculan los mínimos cuadrados expresados como

$$F(b_0, b_1, ..., b_n) = \sum_{i=1}^{m} \omega_i \left[g(b_0, b_1, ..., b_p; x_i) - y_i \right]^2 . \tag{16}$$

Por medio de la derivación e igualando a cero:

$$\frac{\partial F}{b_j} = \sum_{i=1}^{m} \omega_i \left[g(b_0, b_1, ..., b_p; x_i) - y_i \right] \frac{\partial g}{b_j} = 0\,. \tag{17}$$

De esta expresión se llega al siguiente resultado:

$$\sum_{i=1}^{m} \omega_i g(b_0, b_1, ..., b_p; x_i) \sin(j\omega x) = \sum_{i=1}^{m} \omega_i y_i \sin(j\omega x) \text{ para } j \neq 0\,. \tag{18}$$

Finalmente, lo anterior puede escribirse en matrices de la siguiente forma:

$$\begin{bmatrix} \sum_{i=1}^{m}\omega_i & \sum_{i=1}^{m}\omega_i sen(\omega x_i) & \sum_{i=1}^{m}\omega_i sen(2\omega x_i) & \sum_{i=1}^{m}\omega_i sen(p\omega x_i) \\ \sum_{i=1}^{m}\omega_i sen(\omega x_i) & \sum_{i=1}^{m}\omega_i sen^2(\omega x_i) & \sum_{i=1}^{m}\omega_i sen(\omega x_i) sen(2\omega x_i) & \sum_{i=1}^{m}\omega_i sen(\omega x_i) sen(p\omega x_i) \\ \sum_{i=1}^{m}\omega_i sen(2\omega x_i) & \sum_{i=1}^{m}\omega_i sen(2\omega x_i) sen(\omega x_i) & \sum_{i=1}^{m}\omega_i sen^2(2\omega x_i) & \sum_{i=1}^{m}\omega_i sen(2\omega x_i) sen(p\omega x_i) \\ \sum_{i=1}^{m}\omega_i sen(p\omega x_i) & \sum_{i=1}^{m}\omega_i sen(p\omega x_i) sen(\omega x_i) & \sum_{i=1}^{m}\omega_i sen(p\omega x_i) sen(2\omega x_i) & \cdots \quad \sum_{i=1}^{m}\omega_i sen^2(p\omega x_i) \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_p \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m}\omega_i y_i \\ \sum_{i=1}^{m}\omega_i y_i sen(\omega x_i) \\ \sum_{i=1}^{m}\omega_i y_i sen(2\omega x_i) \\ \vdots \\ \sum_{i=1}^{m}\omega_i y_i sen(p\omega x_i) \end{bmatrix} \tag{19}$$

2.2 Búsqueda Tabú

Para poder llevar a cabo una optimización del trabajo realizado en otros proyectos anteriores respecto al diseño de un controlador neuro-difuso, se buscaron distintas alternativas que llevan a cabo dicha optimización por medio de métodos de búsqueda, principalmente orientados en el proyecto a la parte de la matriz de reglas, puesto que los métodos de agrupamientos difusos y las redes neuronales con funciones trigonométricas implican de hecho una mejor consideración del medio ambiente de desempeño del robot en la entrada [22].

2.2.1 Clasificación de los métodos de búsqueda

Los métodos de búsqueda computacionales tratan de encontrar las soluciones que satisfacen un problema formulado unívocamente, esto es, si se conoce el espacio de búsqueda posible, a través de la obtención de un máximo o mínimo que es óptimo. Para realizar un método de búsqueda es necesario tener un modelo del problema que se ajuste lo más posible a la realidad. Si es necesario solucionar problemas dinámicos, los métodos de búsqueda con restricciones son ampliamente usados, asignando valores de importancia a las restricciones para hacerlas suaves o fuertes [23].

Los distintos métodos de búsqueda pueden dividirse en ciegos y heurísticos. Los métodos ciegos no requieren decisiones inteligentes y el camino recorrido es totalmente arbitrario, por ejemplo la búsqueda exhaustiva y otras técnicas clásicas. Los heurísticos utilizan funciones para minimizar el número de iteraciones para alcanzar la solución, dependiendo de la cantidad de información proporcionada, incluyendo desde los métodos de búsqueda global hasta los mapas de Kohonen que utilizan redes neuronales [24].

Las técnicas de búsqueda computacional en general se mencionan y describen a continuación, siendo agrupadas por sus características principales [23].

2.2.1.1 Técnicas clásicas

Las técnicas clásicas se dividen en globales y locales, donde las primeras buscan encontrar el máximo o el mínimo global y las segundas sólo de una vecindad y posteriormente se combinan con otros algoritmos para globalizarse. Pueden emplearse iteraciones para llegar al resultado, o una serie de acotamientos [23].

2.2.1.1.1 Búsqueda exhaustiva

Este método se emplea para espacios de búsqueda pequeños y numerables, examinando cada una de las soluciones posibles, lo cual presenta la desventaja de requerir

una capacidad de procesamiento proporcional a la cantidad de posibilidades, por lo que se puede usar combinada con otros métodos que limiten el espacio. La solución es la que tiene el valor máximo respecto a la función objetivo [23].

2.2.1.1.2 Búsqueda local

Los algoritmos de la búsqueda local implican una solución inicial que va siendo alterada mediante operadores, y posteriormente se compara la alteración respecto al original. Si la alteración implica una mejoría se acepta, y si no es así se descarta y se vuelve a la solución inicial. Este procedimiento se repite hasta no poder hacer mejorías. Una de las búsquedas locales más usadas es la de gradientes o *hillclimbings* cuando son ascendentes [23], [24], [25].

2.2.1.1.3 Métodos con soluciones parciales

Los métodos con soluciones parciales hacen construcciones de variable por variable del problema, obteniendo la respuesta hasta que todas las variables son asignadas. Son eficaces si el espacio de búsqueda puede numerarse y si el número de variables no es excesivo. Dentro de éstos, existen diversas variantes [23]:

- Algoritmos "voraces" o *greedy*: contienen criterios en cada paso realizado y asignan valores que maximizan funciones de utilidad.
- "Divide y vencerás": subdividen el problema para encontrar más rápidamente la solución.
- *Branch and bound*: descartan partes del espacio de búsqueda de acuerdo a una decisión

2.2.1.1.4 Búsqueda global

Los algoritmos de búsqueda global tratan de evitar los máximos o mínimos locales añadiendo aleatoriedad en su recorrido, saltando hacia otro lugar del espacio de búsqueda donde podría haber otro máximo o mínimo global. Los más conocidos son los siguientes [23]:

- "Cocimiento simulado" o *simulated annealing*: gradiente ascendiente que combina saltos aleatorios a otras partes del espacio.
- *Greedy Randomized Adaptive Search Procedure* o GRASP: contiene fases de exploración aleatorias para expandir el espacio de búsqueda recorrido.
- "Búsqueda Tabú", revisado con detalle posteriormente.

2.2.1.2 Algoritmos bioinspirados

A continuación se describen diversos tipos de algoritmos bioinspirados.

2.2.1.2.1 Algoritmos de colonias de hormigas

En la actualidad son los más populares, basados en el comportamiento de las hormigas. Se basan en tres principios: retroalimentación positiva y negativa, la amplificación de fluctuaciones provocada por la retroalimentación y la presencia de fluctuaciones múltiples debido a la presencia e interacción de los individuos del espacio de búsqueda [23].

2.2.1.2.2 Computación basada en el ADN

Para resolver los problemas con esta técnica es necesario poder representar un problema con la forma de una cadena de ADN, y se encuentran las soluciones más "aptas" mediante la optimización combinatoria [23].

2.2.1.2.3 Algoritmos de sistema inmune

Consisten básicamente en una diferenciación de lo que pertenece y no pertenece a la solución que se busca. Adicionalmente, contiene diversos principios inspirados en los biológicos para llevarse a cabo: la diversidad de lo que no pertenece mediante el uso de librerías, la posibilidad de responder ante lo que no pertenece, memoria para guardar lo que se ha realizado anteriormente, y tolerancia para no rechazar lo que pertenece [23].

2.2.1.2.4 Computación de membrana o sistemas P

Los algoritmos están basados en la idea de un procesamiento de la información empleando jerarquías y transformando cada uno de los nodos de la jerarquía en turno. Los estados o nodos del método reciben una entrada de información, cambian de estado y transfieren la información procesada a otro estado. Incorpora principios de las redes neuronales [23].

2.2.1.2.5 Optimización por enjambre de partículas

También llamada *particle swarm optimization* en inglés, implica que el método recorre distintos caminos a través de las soluciones tratando de dirigirse hacia la mejor alternativa del conjunto, considerando los resultados de soluciones óptimas encontrados por cada uno de los caminos en su vecindad. El movimiento de búsqueda se realiza con una aceleración que es parámetro del método, y puede adaptarse con facilidad a cambios en el entorno [23].

2.2.1.3 Algoritmos evolutivos

A continuación se describen algunos tipos de algoritmos evolutivos.

2.2.1.3.1 Algoritmos genéticos

Los algoritmos genéticos son métodos evolutivos, que aplican los métodos de la evolución biológica: selección basada en la población, reproducción sexual y mutación. Los algoritmos genéticos generan parámetros del problema en forma de variables codificadas posteriormente en cromosomas. Los operadores se aplican sobre los cromosomas individualmente o en conjunto [23].

Una vez codificadas las soluciones, éstas compiten para ver cuál es la mejor y que sobreviva o deje su material genético a las siguientes generaciones – las siguientes soluciones exploradas-. En los mecanismos de búsqueda se introduce la variación por medio de la

reproducción sexual y la mutación, las cuales generan cambios en los cromosomas [23]. El algoritmo puede verse resumido de la siguiente manera:

1. Se evalúan los genes codificados en cromosomas.
2. Se reproducen de acuerdo a la aptitud resultante.
3. Se emparejan los individuos de la nueva población para intercambiar el material genético y algunos de los genes son alterados en mutaciones espontáneas.
4. Continúa el proceso hasta lograr una selección

El tamaño de la población evaluada debe ser suficiente para garantizar la diversidad de soluciones, y la terminación del algoritmo se da con una condición que generalmente implica un número de generaciones predeterminado o la convergencia [23].

Este método es robusto ante la independencia respecto a problema, por lo que puede ser empleado para resolver prácticamente cualquier tipo donde se realice una búsqueda. A la vez, su falta de especialización lo hace débil [23].

2.2.1.3.2 Programación genética

Consiste en la evolución de programas computacionales para obtener algunos que realicen funciones determinadas. Sin embargo, sólo se obtienen funciones multivariadas con una sola salida. Son empleados para circuitos y regresiones simbólicas, entre otros [23].

2.2.1.3.3 Algoritmos evolutivos paralelos

En los algoritmos evolutivos normales la función de evaluación es iterada una gran cantidad de veces dependiendo de la cantidad de cromosomas y genes. Debido a que cada elemento del conjunto de soluciones puede ser evaluado normalmente de forma independiente, por la descentralización de los resultados, el procesamiento puede realizarse con estructuras paralelas que permitan la evaluación simultánea [23].

2.2.1.4 Optimización multiobjetivo

Cuando los problemas involucran la presencia de varios máximos o mínimos es posible recurrir a la optimización multiobjetivo. Esta optimización implica encontrar un vector de variables de decisión que satisfacen restricciones y mejoran una función vectorial objetivo. En este caso, optimizar significa encontrar una solución que proporcione valores "aceptables" a todas las funciones objetivo en conjunto, ya que es difícil encontrar un punto donde se satisfaga a todas simultáneamente [23].

Este concepto distinto de optimización se denomina óptimo Pareto, y normalmente existe un conjunto de ellos – también denominado conjunto no dominado –, y puede ser representado gráficamente mediante un diagrama llamado "frente Pareto", el cual permite visualizar características de los óptimos Pareto [23].

2.2.1.5 Mapas de Kohonen

Los mapas de Kohonen o SOM – *self-organizing* map – están clasificados dentro de los algoritmos de clasificación no supervisados, ya que auto-asocian las entradas a ciertas etiquetas para realizar su procesamiento. Analizan un conjunto de datos mediante un análisis exploratorio que permite visualizar las relaciones y la cantidad de los grupos naturales que hay en la entrada. Suelen agruparse dentro de las redes neuronales; ya que a partir de un proceso de entrenamiento agrupan los datos de un conjunto según características establecidas [23]. Estos mapas tienen distintas aplicaciones descritas a continuación:

- Agrupamientos de datos de entrada siguiendo distintos criterios.
- Visualización de agrupamientos ordenados.
- Clasificación de datos desconocidos.
- Interpolación de una función asignando valores a cada nodo de la red.

2.2.2 Búsqueda Tabú

El método tabú es una técnica heurística agrupada como clásica debido a que incorpora mecanismos de búsqueda global, añadiendo cierto grado de aleatoriedad programada.

2.2.2.1 Desarrollo histórico

Los algoritmos heurísticos se derivan del desarrollo conjunto entre la inteligencia artificial y la investigación de operaciones, con la meta de resolver problemas, en los 50's y 60's. Con la posterior divergencia entre estos dos campos debido al cambio en el enfoque de la investigación de operaciones, hacia la optimización enfocada en resultados matemáticos de convergencia, se introdujeron algunos métodos no tradicionales para la optimización, lejos de las ideas de convergencia y empleando memoria. Asimismo, se introdujo el uso de principios probabilísticos y conceptos de diseño integrado, y sin embargo el uso de los nuevos métodos fue prácticamente nulo [26].

En los 80's se retomaron cuatro desarrollos previos para integrar la búsqueda Tabú [26]:

- Estrategias que combinan reglas de decisión basadas en reestructuración lógica y búsqueda no monótona
- Violación y restauración de las posibilidades
- Memoria flexible basada en hechos recientes y en la frecuencia
- Procesos selectivos para combinar soluciones, aplicado a una población mantenida sistemáticamente

2.2.2.2 Fundamentos base

La búsqueda Tabú inicia como cualquier búsqueda local o de vecindario, procediendo con iteraciones de una solución a otra hasta que el criterio de terminación es satisfecho. El método parte de una función $f(x)\ \ x \in X$ que se desea optimizar, donde $f(x)$ puede ser lineal o no lineal, y X es un conjunto que resume al vector de x variables de decisión. Cada $x \in X$

tiene un vecindario asociado $N(x) \subset X$, y cada solución $x' \in N(x)$ es alcanzada desde x por un operador denominado "movimiento" [26].

En este punto, el método Tabú puede compararse con uno descendente, con el siguiente algoritmo:

1. Se elige $x \in X$ para iniciar el proceso
2. Encontrar $x' \in N(x)$ tal que $f(x') < f(x)$
3. Si x' no puede ser encontrada, x es el óptimo local y el método se detiene
4. Si no es así, x' es designada como la nueva x y se regresa al paso 2

No obstante, el método Tabú determina una lista de candidatos apropiados como elementos de $N(x)$, para garantizar un balance entre la calidad de x' encontrada y el tiempo del método, ya que los métodos de búsqueda descendente no consideran el tiempo necesario para recorrer todos los elementos del conjunto de posibles soluciones, lo cual depende directamente del número de elementos contenidos [26]. El método Tabú opera desviándose de un camino de búsqueda, pero dicha desviación no es elegida de forma aleatoria, sino que se exploran nuevas regiones del espacio de soluciones con la premisa de poder aceptar respuestas que no son localmente óptimas si conducen hacia una región no explorada previamente. Esto asegura que el espacio sea explorado evitando mínimos o máximos locales [27].

Para lograr lo descrito anteriormente, la búsqueda Tabú emplea memoria de largo y corto términos, acompañadas de sus estrategias especiales. Estas pueden verse como modificadoras de la vecindad $N(x)$ de la solución actual x, donde la nueva vecindad se denota como $N^*(x)$, característicamente un subconjunto de $N(x)$. Se realiza una clasificación de "tabú " para identificar los elementos excluidos en $N^*(x)$ [26].

Un proceso de búsqueda Tabú basado estrictamente en estrategias de corto plazo permite que una solución x sea visitada más de una vez, pero el vecindario $N^*(x)$ será

distinto cada vez. Si se incluye la memoria de largo plazo, la probabilidad de duplicar un vecindario anterior revisando una solución es inexistente [26].

Las estructuras de memoria comprenden calidad, influencia, frecuencia y tiempo. La calidad se refiere a la habilidad de diferenciar si las soluciones visitadas en la búsqueda son buenas elecciones o no. La influencia considera el impacto de las elecciones durante la búsqueda tanto en calidad como en estructura. Asimismo, el tiempo implica un almacenamiento de qué tan recientemente se ha explorado una solución a lo largo de la búsqueda para poder establecer criterios que acoten el tamaño de la nueva región de posibles movimientos [28].

Por otra parte, la memoria es explícita y atributiva, almacenando soluciones completas que son seleccionadas durante la búsqueda. Una extensión de esta memoria almacena vecindarios de búsqueda atractivos que son vecinos de la selección elite, pero inexplorados. La combinación de la memoria y su extensión son empleadas para expandir la búsqueda local a otras regiones [28].

2.2.2.2.1 Memoria a corto plazo

El aspecto crucial de la búsqueda Tabú implica encontrar una definición apropiada de $N^*(x)$, que depende de la trayectoria seguida en el movimiento de una solución a otra. Para examinar la memoria a corto plazo, se considera una lista tabú T que contiene t diferentes soluciones $T = \{x_1, x_2, ..., x_t\}$, donde $N^*(x) = N(x) \setminus T$. Esta memoria está basada en las soluciones visitadas recientemente o sus atributos, eligiendo algunas de ellas o determinados atributos que son etiquetados "activos en tabú". Esto previene que sean parte de $N^*(x)$ y por lo tanto que sean visitados nuevamente [26].

Las clasificaciones tabú pueden no ser simétricas ni estáticas, mediante la clasificación a partir del número de iteraciones que una solución o atributo es mantenido activo en tabú -*tabu*

tenure-, siendo dinámico y aleatorio o dinámico sistemático, con un rango definido por t_{min} y $t_{máx}$, por ejemplo $\alpha\ t_{min}$ iteraciones [26].

Adicionalmente, se introduce un criterio de aspiración para determinar cuándo puede prescindirse de la activación del tabú. Algunos de los criterios son: emplear un grado de influencia normalmente relacionado con las distancias en los movimientos; seleccionar el movimiento "menos tabú" si todos los movimientos disponibles han sido clasificados como tabú; *eliminar* o agregar atributos de una solución si la dirección de búsqueda no ha cambiado [26].

Para hacer la lista de candidatos a formar parte de $N*(x)$ se utilizan distintas estrategias descritas a continuación [26]:

- Criterio de "aspiración plus", midiendo la calidad de un movimiento a través de la historia del patrón de búsqueda, y analizando distintos movimientos hasta encontrar uno que satisfaga un umbral de calidad.
- Lisa de candidatos elite, examinando todos o un gran número de movimientos y determinando un número k de los mejores encontrados, siendo k un parámetro del proceso.
- Estrategia de filtrado sucesivo, a través de la evaluación de elementos.
- Lista de candidatos secuencial, la cual genera p movimientos alternativos en una etapa dada.
- Lista de candidatos enlazada, restringiendo el dominio de alternativas.

2.2.2.2.2 Memoria a largo plazo

En muchas aplicaciones la memoria de corto plazo es suficiente para producir soluciones óptimas, sin embargo el uso de la memoria a largo plazo fortalece el método de búsqueda. La memoria basada en frecuencia complementa la memoria en corto plazo creando

una estructura compuesta para crear penalizaciones e inducciones que modifican la evaluación de los movimientos [26].

El enfoque de frecuencia emplea una medida de transición para el número de iteraciones donde un atributo cambia la solución dentro de una trayectoria particular visitada, y también una medida de residencia para el número de iteraciones en las cuales un atributo pertenece a las soluciones. La medida de residencia no considera las características particulares o atributos de una solución [26].

La memoria basada en frecuencia es aplicada usualmente introduciendo estados de tabú graduales, empleando estrategias de intensificación para modificar reglas de elección de movimientos, e impulsar combinaciones y características de soluciones encontradas históricamente buenas. También se emplean estrategias de diversificación para incrementar la efectividad en la exploración del espacio total de soluciones, introduciendo atributos no usados o usados poco en los criterios de selección.

Emplear estrategias de intensificación ayuda a modificar las reglas de elección de soluciones para impulsar combinaciones encontradas como buenas, e incluso para regresar hacia regiones de búsqueda atractivas para recorrerlas a fondo. Las estrategias de diversificación ayudan a visitar regiones nuevas y generar soluciones que varíen significativamente de las vistas previamente [28].

Por otra parte, es posible recurrir a una oscilación estratégica que conjunta la intensificación con la diversificación descritas anteriormente. En un nivel crítico donde normalmente el método se detendría, las reglas para seleccionar movimientos son modificadas para que la región definida sea cruzada en otra dirección. Para realizar los cruces de regiones es necesario elegir un patrón de oscilación, que puede ser uniforme o intensificado [26].

2.2.2.3 Algoritmo general

En la Figura 2.8 se muestra un esquema básico del algoritmo, el cual inicia con una solución inicial posible, posteriormente se genera una nueva solución de prueba en el espacio alrededor de la primera solución, y se evalúa usando un criterio que permita decidir si es el mejor movimiento siguiente a realizar [29].

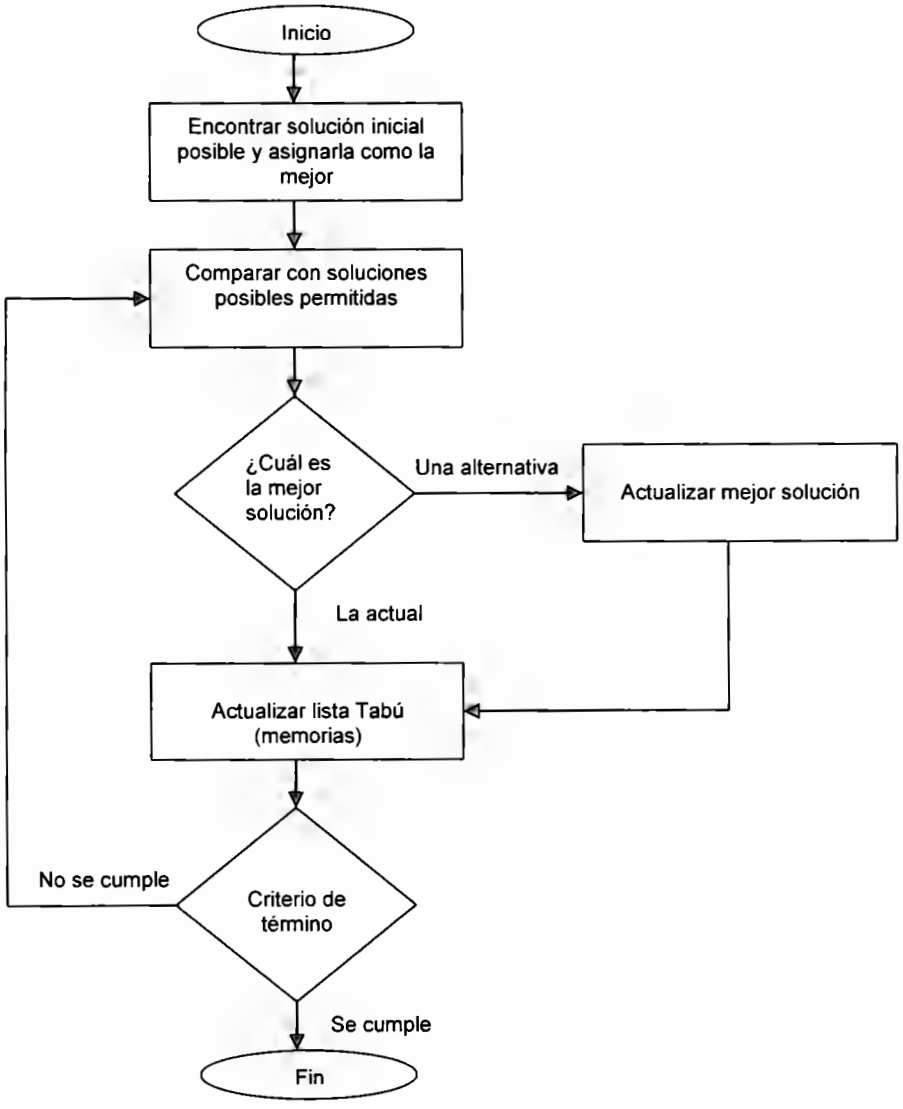


Figura 2.8 Diagrama esquemático representando el método de búsqueda Tabú [29]

A continuación se muestra un ejemplo [26] en el cual se empleó la búsqueda Tabú descrita en el diagrama anterior para obtener la mejor solución posible a un problema de reemplazar la arista de un árbol partiendo de la solución inicial propuesta en la Figura 2.9. La Tabla 2.2 muestra los pesos obtenidos en la evaluación de cada solución posible, así como las aristas agregadas a la lista Tabú en un ciclo y en dos ciclos, actualizando la columna correspondiente a la lista. Esto se muestra también gráficamente en la Figura 2.10.

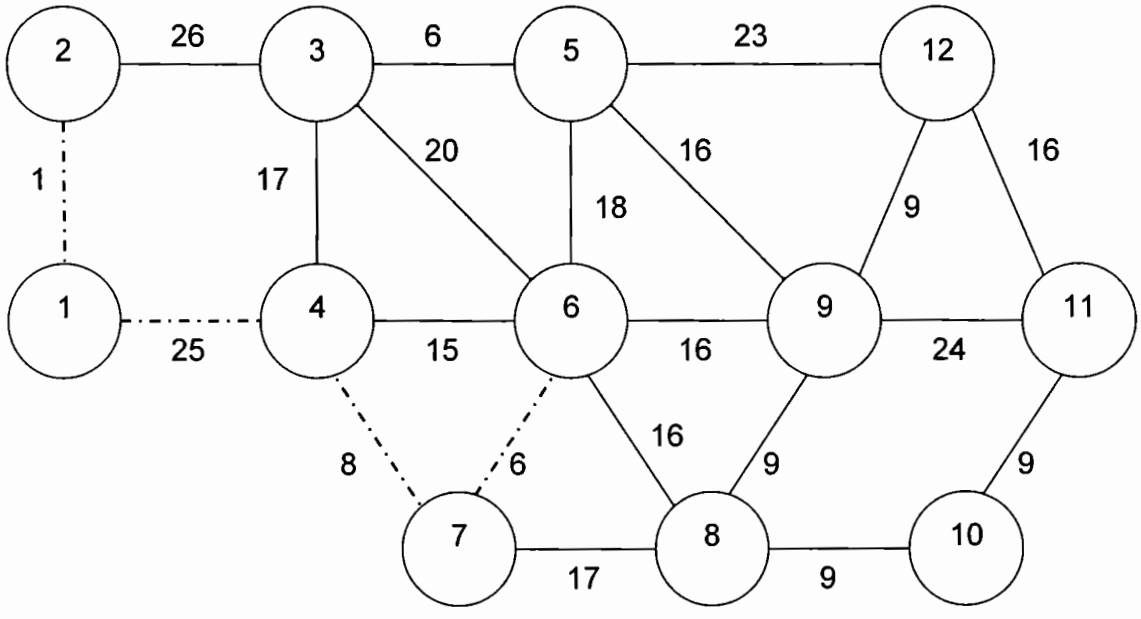


Figura 2.9 Solución inicial planteada para el ejemplo del método Tabú [26]

Iteración	*Lista Tabú*	*Clasificado Tabú durante 2 iteraciones*	*Clasificado Tabú durante 1 iteración*	*Peso obtenido como mejor solución*
1	Vacía	(4,7)	(4,6)	47
2	(4-6), (4-7)	(6,7)	(6,8)	57
3	(6-8), (4-7), (6,7)	(1,2)	(8,9)	65

Tabla2.2 Lista Tabú, agregados y mejor solución obtenida para cada iteración [26]

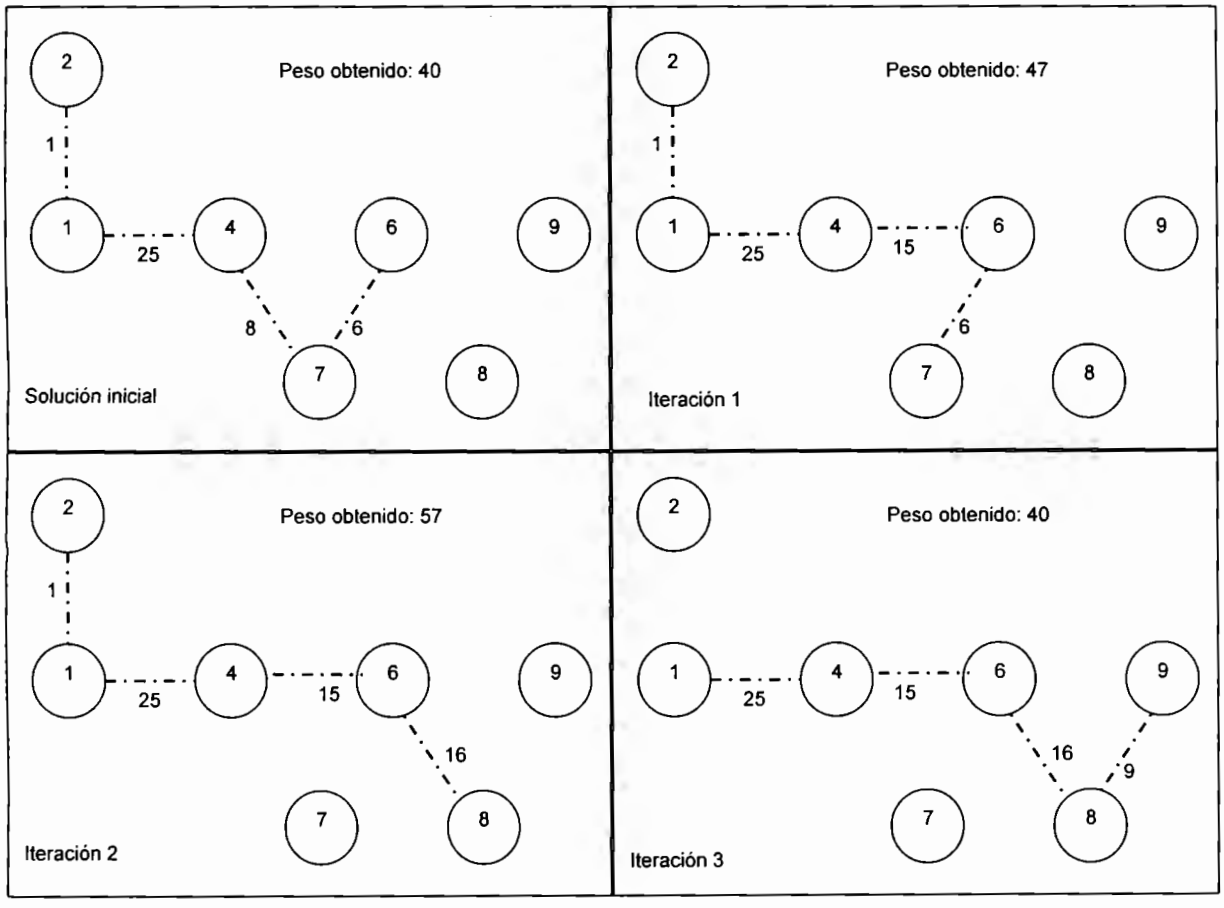


Figura 2.10 Iteraciones para el ejemplo de búsqueda Tabú [26]

2.2.2.4 Aplicaciones

Algunas aplicaciones ilustrativas de la búsqueda Tabú en diversas áreas pueden enlistarse a continuación [26]:

- Calendarización y planeación: calendarización de sistemas de manufactura, requerimientos generalizados de capacidad en la planeación de la producción, planeación de la producción con aprendizaje de la fuerza de trabajo, optimización del plan de proceso para centros CNC con varias torretas y bases para piezas, calendarización para la reforestación sustentable.
- Telecomunicaciones: redes de comunicación Hub-and-Spoke, optimización de redes B-ISDN, diseño de redes ópticas.

- Computación paralela: mapeo de tareas para minimizar el tiempo de comunicación de procesadores en un sistema multiprocesador, calendarización de tareas en programas paralelos de multiprocesadores, problema de asignación cuadrática en una conexión de máquina, diseño de circuitos integrados.
- Transporte, diseño de rutas y diseño de redes: problemas de transporte de carga fija, rutas para vehículos, distribuciones, diversificación e intensificación probabilísticas para diseñar rutas, aplicaciones de masa de tránsito, patrones para rutas de vehículos guiados autónomamente.
- Estructuras de optimización: modelo de conformación de proteínas, optimización de estructuras electromagnéticas, reforzamiento de polímeros, emplazamiento de estructuras espaciales, control acústico estructural activo.
- Optimización en gráficos: particiones de gráficos, dibujos.
- Redes neuronales y aprendizaje: aprendizaje sub-simbólico, optimización global de las redes neuronales artificiales, optimización combinatoria, sistemas VLSI con capacidades de aprendizaje.
- Optimización continua y estocástica: administración de portafolios.
- Manufactura: asignación de tareas para el balanceo de líneas de ensamble, selección de mapas de instalaciones, cortes irregulares en dos dimensiones.
- Análisis financiero: diseño por computadora de productos financieros, problemas de inversiones dinámicas.

2.2.3 Algoritmo para la búsqueda Tabú

Después de revisar diversos tipos de sistemas de búsqueda, se optó por emplear el método de la búsqueda Tabú para sintonizar la matriz de reglas difusas o FAM (Matriz de Asociación Difusa) [30] de manera óptima, debido a que permite la obtención de un óptimo local en un tiempo más reducido que a través de métodos de búsqueda exhaustiva. La

búsqueda Tabú es flexible en la obtención del conjunto de reglas para el controlador neuro-difuso, y permite un funcionamiento paralelo al controlador, para la obtención de reglas en tiempo real en lo que se conoce como búsqueda Tabú dinámica o reactiva; sin embargo, la sintonización de las reglas puede realizarse de forma estática, antes de emplear el controlador.

En las Figuras 2.11, 2.12, 2.13, 2.14 y 2.15 se aprecia el algoritmo propuesto [30], tanto en forma lineal como en diagrama de flujo, para la búsqueda Tabú reactiva, en la cual el conjunto de movimientos M serán los conjuntos difusos de salida para cada relación entre los conjuntos de entrada, donde: $L(\mu)$ es la última iteración en la que el movimiento μ fue aplicado, siendo $-\infty$ si μ nunca ha sido empleado; $\Pi(f)$ es la última iteración en la que la configuración f ha sido visitada durante la búsqueda, siendo $-\infty$ si f nunca ha sido visitada o si ya no se encuentra en memoria; $\Phi(f)$ es el número de veces que la configuración f ha sido visitada durante la búsqueda, teniendo $\Phi(f) = 0$ en un inicio para las configuraciones.

A cada iteración t, el conjunto $A^{(t)}$ contiene los movimientos que no revierten a aquellos realizados recientemente, es decir, los movimientos permitidos; el conjunto $T^{(t)}$ contiene los movimientos no permitidos o tabú, con un periodo de prohibición T que representa el número de los últimos pasos de la búsqueda. Al inicio de la búsqueda, $A^{(0)} = M$ y $T^{(0)} = \phi$, pero es necesario inicializar $T^{(t)}$ con un valor pequeño para iniciar el periodo de tabú.

Para cada configuración visitada, el valor más bajo de E, la función del error a minimizar, y la configuración f que generó dicho valor de E son guardadas en la memoria. Si f es encontrada nuevamente durante la búsqueda $\Phi(f)$ y $\Pi(f)$ son actualizados. En caso de que la cantidad de repeticiones sea mayor que un umbral denominado REP, f es agregada al conjunto C, y si C es mayor que el umbral llamado $CHAOS$ es necesaria una fase de diversificación de la búsqueda.

```
BusquedaTabuReactiva( )
% Rutina principal
        %Inicialización
        t = 0                          %Contador de iteraciones
        f^(0) = random                 %Configuración inicial
        A^(t) = Ω                      %Movimientos admisibles, todos inicialmente
        T^(t) = φ
        f_b = f^(0)                    %Mejor configuración encontrada
        E_b = E(f^(0))                 %Mejor valor para la función del error a minimizar
        %Ciclo de iteraciones
        Repetir
                %Verificar si la configuración actual es una repetición
                escape = ReaccionBasadaEnMemoria( f^(t) )
                if escape = FALSO then
                        μ = MejorMovimiento( )
                        f^(t+1) = μ(f^(t))
                        Actualizar A^(t) y T^(t)
                        t++
                        if E(f^(t)) < E_b then          %Actualizar mejor configuración
                                f_b = f^(t)
                                E_b = E(f^(t))
                        endif
                else
                        DiversificarBusqueda( )
                endif
        Hasta que E_b sea un valor aceptable o se alcance el máximo de iteraciones posibles
```

Figura 2.11 Algoritmo del método de búsqueda Tabú reactivo, rutina principal [30]

```
MejorMovimiento( )
% Movimiento a aplicar en la configuración actual
        if |A^(t)| < 2 then
                T^(t) = L - 2          %Cuando menos dos movimientos permitidos
                t_T = t
        endif
        δ = max(SAMPLE, |A^(t)|)       %Movimientos al azar elegidos del conjunto A^(t)
        μ = arg min_{v∈δ} E(v(f^(t)))
        return μ
```

Figura 2.12 Algoritmo del método de búsqueda Tabú reactivo, rutina para la obtención del mejor movimiento [30]

```
ReaccionBasadaEnMemoria( f^(t) )
%Verificar si se requiere una fase de diversificación
    %Buscar la configuración f en memoria
    if Π(f) ≠ −∞ then                    %La configuración ya fue visitada
        R = t − Π(f)                     %Longitud del ciclo
        Π(f) = t
        Φ(f)++
        if Φ(f) > REP then
            C = C ∪ f                    %Agregar f al conjunto de configuraciones frecuentes
            if |C| > CHAOS then
                C = φ
                return VERDADERO
            endif
        endif
        if R < 2(L − 1) then
            T^(t+1) = T^(t) · INCREASE
            t_T = t
            R_ave = 0.1R + 0.9R_ave
        Endif
    else                                 %La configuración no está en memoria
        Π(f) = t
        Φ(f) = 1
    endif
    if (t − t_T) > R_ave then
        T^(t+1) = max(T^(t) · DECREASE, 1)
        t_T = t
    endif
    return FALSO
```

Figura 2.13 Algoritmo del método de búsqueda Tabú reactivo, rutina de reacción basada en memoria [30]

Para eliminar el ciclado del método, se aumenta el valor de $T^{(t)}$ por un factor denominado $INCREASE$. No obstante, es necesario disminuir el valor de $T^{(t)}$ para las fases de búsqueda siguientes mediante un factor $DECREASE$.

DiversificarBusqueda( )
% Ejecuta un paseo al azar

Inicializar Π y Φ

$\delta = \min(|M|, 1 + R_{ave}/2)$ %Movimientos al azar escogidos del conjunto M

$T^{(t)} = L - 2$ %Cuando menos dos movimientos permitidos

$t_T = t$

for $\sigma \in \delta$

$f^{(t+1)} = \sigma(f^{(t)})$

$\Lambda(\sigma) = t$ %Actualizar $A^{(t)}$ y $T^{(t)}$

$t--$

if $E(f^{(t)}) < E_b$ then %Actualizar mejor configuración encontrada

$f_b = f^{(t)}$

$E_b = E(f^{(t)})$

endif

endfor

Figura 2.14 Algoritmo del método de búsqueda Tabú reactivo, diversificación de la búsqueda [30]

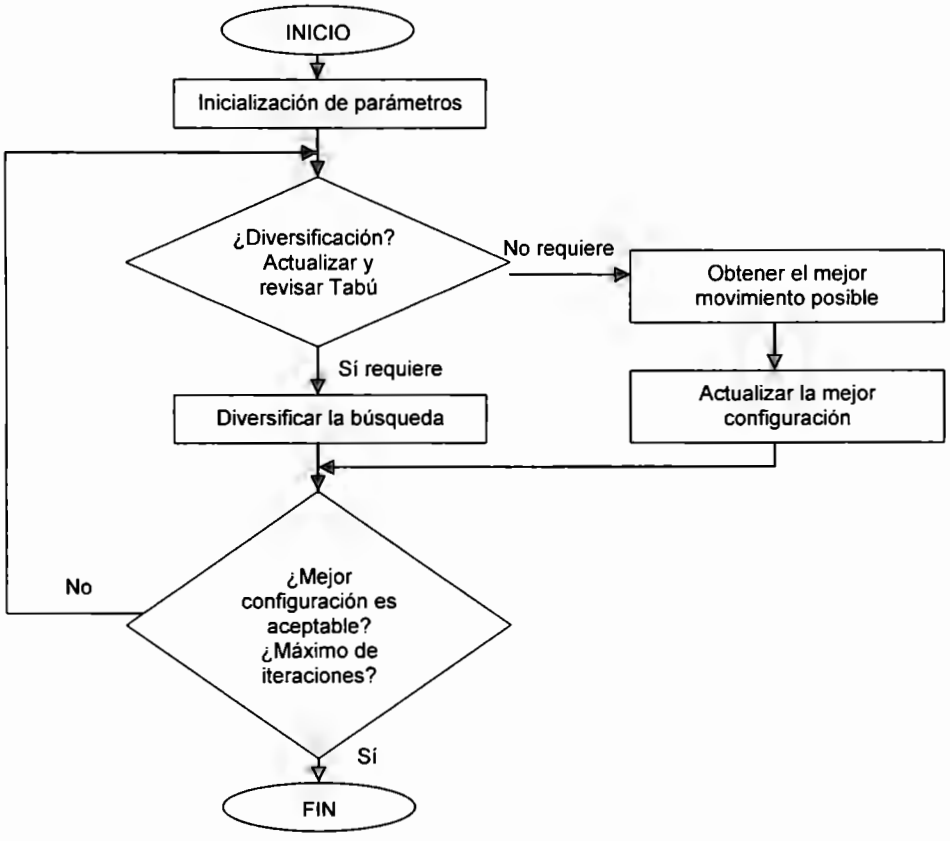


Figura 2.15 Algoritmo del método de búsqueda Tabú reactivo, diagrama esquemático [30]

Tecnológico de Monterrey, Campus Ciudad de México
Biblioteca

2.2.3.1 Programa desarrollado

El programa específico que se propuso para el proyecto se aprecia en la Figura 2.16, el cual comprende una variante estática que busca la mejor solución fuera de la línea de ejecución del controlador neuro-difuso.

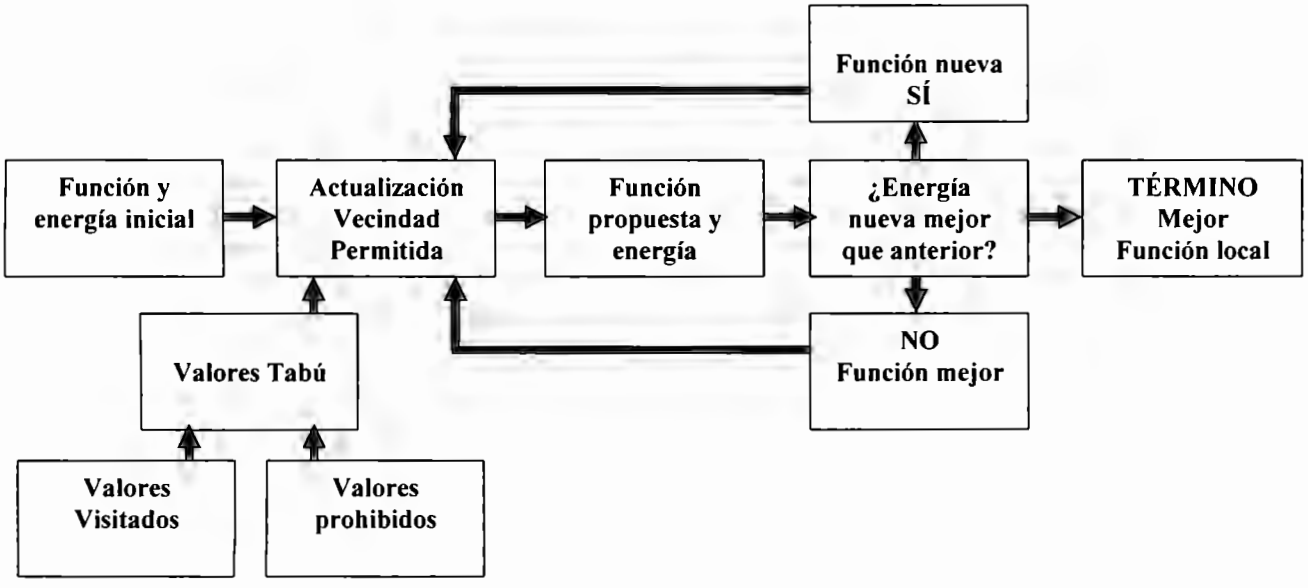


Figura 2.16 Algoritmo Tabú propuesto para el proyecto

CAPÍTULO 3

DISEÑO DEL SISTEMA

La etapa de diseño del sistema se subdividió en cuatro partes correspondientes al controlador, el prototipo físico, la comunicación y la tarea colaborativa simple, las cuales se describen detalladamente a continuación.

3.1 Diseño del controlador

A continuación se presenta el diseño obtenido para el controlador neuro-difuso de navegación autónoma con búsqueda Tabú, que comprende el controlador para la posición y el controlador para la evasión de obstáculos, el primero siendo verificado en simulación con la obtención de un modelo matemático para el comportamiento de la planta a partir del movimiento de los servomotores del robot.

3.1.1 Diseño del controlador neuro-difuso con búsqueda Tabú

Un controlador neuro-difuso combina diversas ventajas sobre las técnicas de control clásico empleadas constantemente en la industria, entre las que se encuentran la gran flexibilidad de la estructura de control para adaptarla a sistemas lineales y no lineales, que trabajan en distintos puntos de operación [18]. Debido a estas características, el uso de

controladores neuro-difusos resulta una alternativa viable para lograr el control de los robots del proyecto.

El diseño del controlador neuro-difuso total con búsqueda Tabú fue dividido en dos etapas, una correspondiente al controlador de posición, y otra referente al controlador de navegación para la evasión de obstáculos, siguiendo los procedimientos bosquejados en las Figuras 3.1 y 3.2, para cada controlador. El diseño de ambos controladores es descrito a continuación.

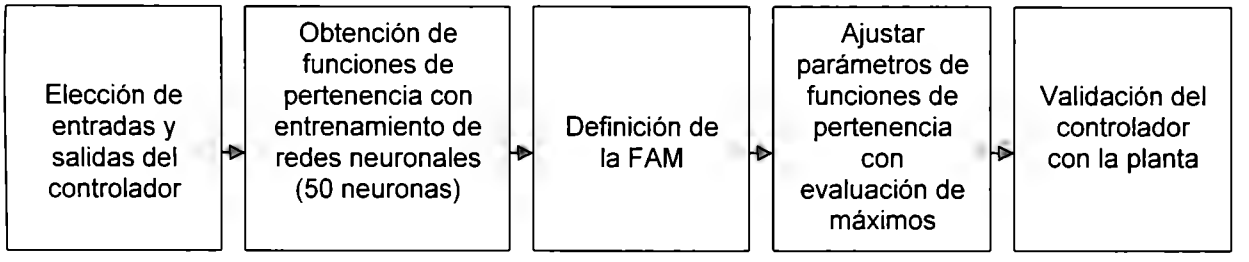


Figura 3.1 Procedimiento de diseño para el control de posición

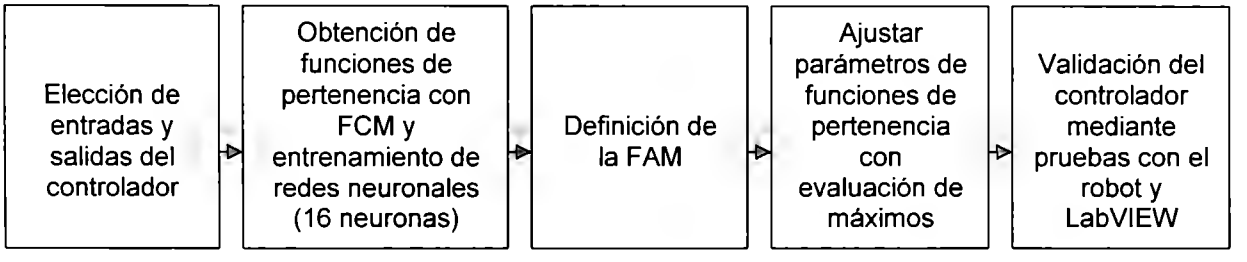


Figura 3.2 Procedimiento de diseño para el control de evasión de obstáculos

3.1.1.1 Diseño del controlador de posición

Después de una cuidadosa revisión de los trabajos realizados con anterioridad [18], [22] referentes al diseño e implementación de un controlador neuro-difuso para la navegación autónoma de un robot hexápodo y posteriormente un robot cuadrúpedo, se diseñaron y programaron los siguientes algoritmos genéricos, que se encuentran adjuntos como archivos

tipo matemática textual (m-files) en el CD anexo y en el Anexo 2 (FCM), al igual que la explicación del código para el controlador neuro-difuso genérico en el Anexo 3.

3.1.1.1.1 Algoritmo FCM con redes neuronales

1. Se propone un número de agrupamientos difusos o clusters c, así como el coeficiente difuso m.

2. Se introduce el conjunto de datos que serán agrupados, con centroides elegidos aleatoriamente.

3. Se calculan las funciones de pertenencia de todos los datos en cada uno de los agrupamientos difusos, y posteriormente se recalculan los centroides para cada agrupamiento.

4. Se evalúa la función objetivo con las funciones de pertenencia y los centroides calculados en el paso anterior.

5. Se realiza el procedimiento de los puntos 3 y 4 hasta que la función objetivo sea mínima y estable.

6. Se introducen las funciones de pertenencia resultantes en las redes neuronales para su entrenamiento.

7. Se encuentran los coeficientes de los pesos correspondientes a las entradas de la red neuronal.

3.1.1.1.2 Algoritmo de Búsqueda Tabú

1. Se encuentra una solución inicial para la matriz de asociación difusa y se asigna como la mejor.

2. Se compara la solución catalogada como mejor con otras soluciones posibles o distribuciones de reglas permitidas.

3. Se determina cuál es la mejor solución, y se actualiza.

4. Se actualiza la lista Tabú a través de las memorias de corto y largo plazo, donde se colocan las soluciones que no se desean visitar en las siguientes etapas de la búsqueda.

5. Se verifica el criterio de término de la búsqueda, y si no ha concluido se repiten los pasos 2 a 4.

3.1.1.1.3 Controlador neuro-difuso

1. El algoritmo FCM sintoniza las funciones de membresía del controlador, que son aprendidas a través de redes neuronales.

2. Los datos introducidos en la entrada son difusificados con las funciones de membresía obtenidas y entrenadas en el paso 1.

3. Se realiza el procedimiento de inferenciación tipo Mamdani (mínimo-máximo), a partir de una matriz de asociación difusa introducida al sistema, proveniente o no del método de búsqueda Tabú.

4. Se desdifusifican las salidas del sistema a través de tablas de búsqueda establecidas.

Una vez programados todos los algoritmos necesarios, se procedió a seleccionar las entradas y salidas del controlador específico de nuestro proyecto, correspondientes al error de la magnitud y ángulo medidos entre la posición real y la deseada y referido al plano del robot para las entradas, y los pulsos para el movimiento de los servomotores como salidas, como se muestra en la Figura 3.3.

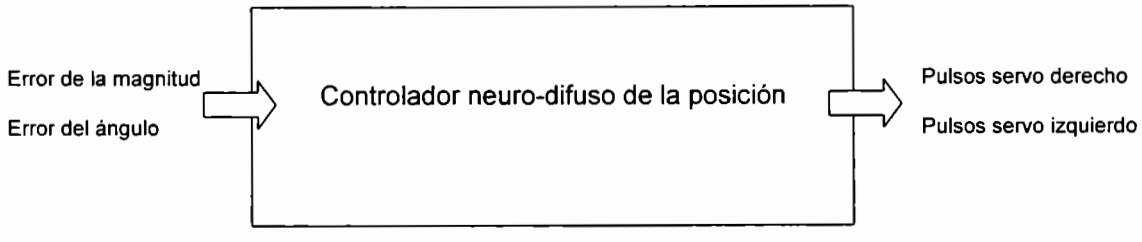


Figura 3.3 Entradas y salidas del controlador de posición

Se diseñaron en MATLAB las funciones de pertenencia partiendo de las bases teóricas de un controlador difuso, considerando cinco para cada una de las dos entradas. Las Figuras 3.4 y 3.5 muestran los parámetros empleados, para el error de la magnitud y para el error del ángulo, respectivamente. Estas funciones han sido denominadas como NG (negativos grandes), NP (negativos pequeños), Z (cero), PP (positivos pequeños) y PG (positivos grandes).

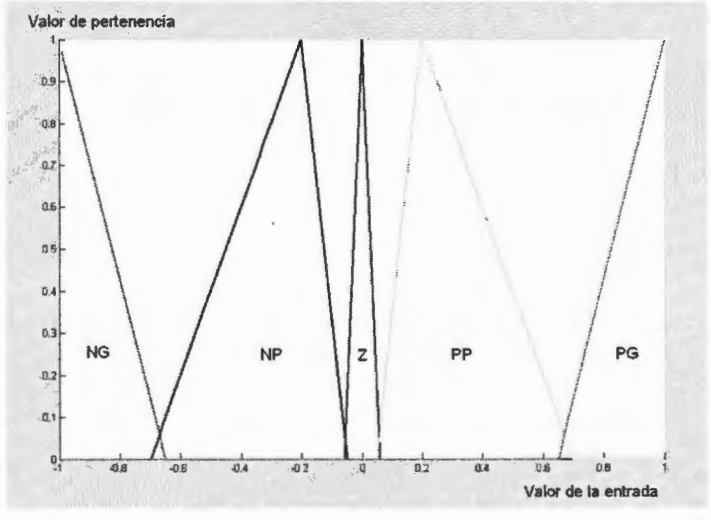


Figura 3.4 Funciones de pertenencia para el error de la magnitud

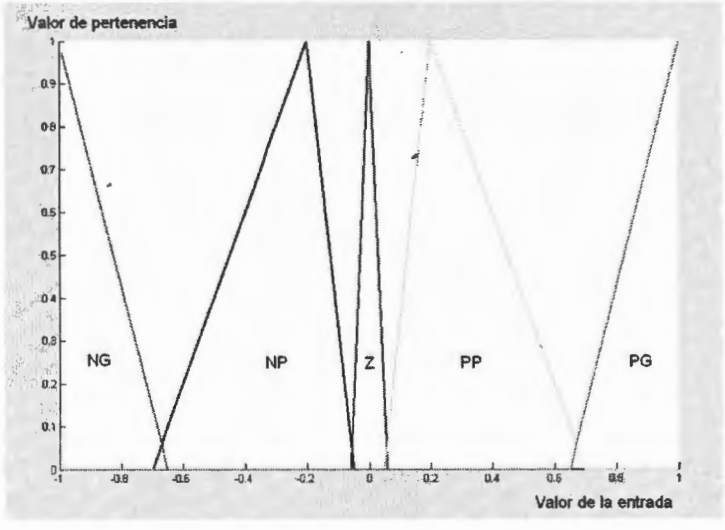


Figura 3.5 Funciones de pertenencia para el error del ángulo

De las funciones de pertenencia se extraen un conjunto de puntos que se introducen a dos redes neuronales basadas en series trigonométricas, una por cada variable de entrada y con cincuenta neuronas por función de pertenencia, para efectuar un entrenamiento y obtener los coeficientes de los pesos de las entradas de cada red. Después del entrenamiento, el controlador ya se encuentra listo para difusificar las entradas que pasan a través de la red neuronal y de ahí al proceso de inferencia. Cabe mencionar que fue necesario verificar los parámetros de amplitud empleados en las funciones de pertenencia triangulares propuestas en primer lugar para el entrenamiento de las redes, a través del envío de los valores máximos de cada función en la entrada por separado, y evaluando las salidas respectivas del controlador, como se ilustra en la Figura 3.6.

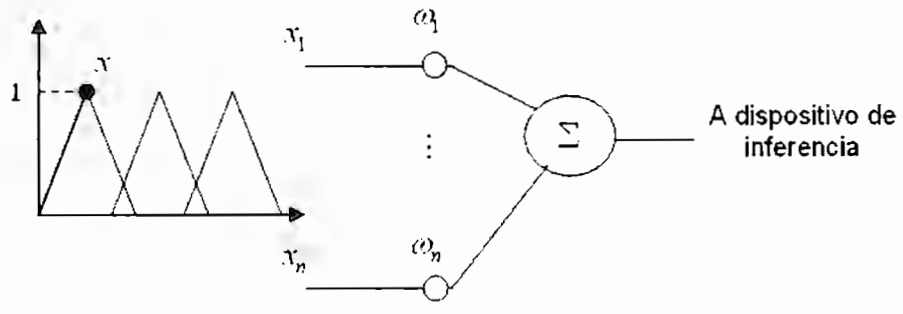


Figura 3.6 Proceso de verificación de los parámetros de las funciones de pertenencia empleadas para el entrenamiento de la red neuronal

Para la matriz de asociación difusa o FAM se definieron las reglas que se muestran en la Tabla 3.1, para cada una de las funciones de pertenencia de ambas entradas.

Las funciones de pertenencia correspondientes al pulso de la rueda derecha y al de la rueda izquierda pueden apreciarse en la Figura 3.7 y 3.8, respectivamente, para un rango de pulsos escalado entre -10 y 10 en lugar de los valores convencionales entre 0 y 1. Estas funciones fueron obtenidas empleando la ecuación:

$$pulso_j = \frac{\sum_{i=1}^{n}[\min\{\mu_i\} \bullet FAM(i,j)]}{\sum_{i=1}^{n}\min\{\mu_i\}}. \qquad (20)$$

Para la etapa de desdifusificación se consideró el uso de redes neuronales basadas en series trigonométricas en un principio, y de esta forma obtener la salida del controlador. Sin embargo, la longitud y forma de las funciones de pertenencia obtenidas a partir de las redes disparaba varias reglas de la FAM simultáneamente, por lo que el controlador hacía que el robot llegara a una posición completamente errónea. Por ello, las redes neuronales se reemplazaron con tablas de búsqueda basadas en valores discretos, y así finalmente se logró la aproximación del robot a la posición deseada.

Magnitud del error	*Ángulo del error*	*Pulso derecha*	*Pulso izquierda*	*Rutina*
NG	NG	0	30	I++
NG	NP	0	15	I+
NG	Z	30	30	C++
NG	PP	15	0	D+
NG	PG	30	0	D++
NP	NG	0	30	I++
NP	NP	0	15	I+
NP	Z	15	15	C+
NP	PP	15	0	D+
NP	PG	30	0	D++
Z	NG	0	0	Z
Z	NP	0	0	Z
Z	Z	0	0	Z
Z	PP	0	0	Z
Z	PG	0	0	Z
PP	NG	0	30	I++
PP	NP	0	15	I+
PP	Z	-15	-15	C-
PP	PP	15	0	D+
PP	PG	30	0	D++
PG	NG	0	30	I++
PG	NP	0	15	I+
PG	Z	-30	-30	C–
PG	PP	15	0	D+
PG	PG	30	0	D++

Tabla 3.1 FAM para el controlador de posición

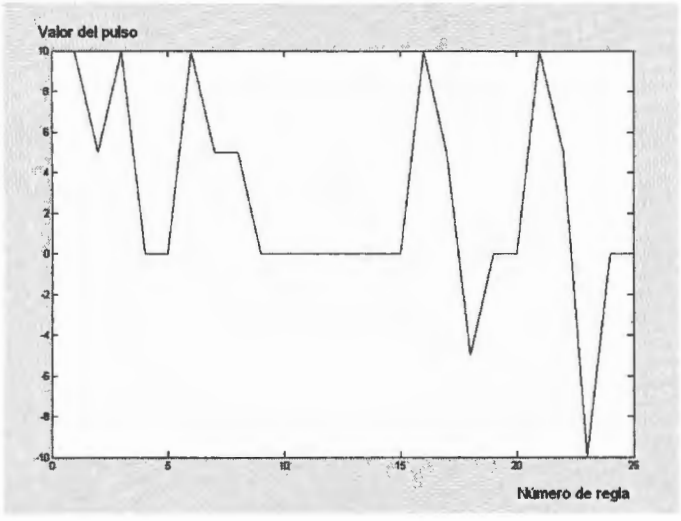


Figura 3.7 Funciones de pertenencia para el pulso de la rueda derecha

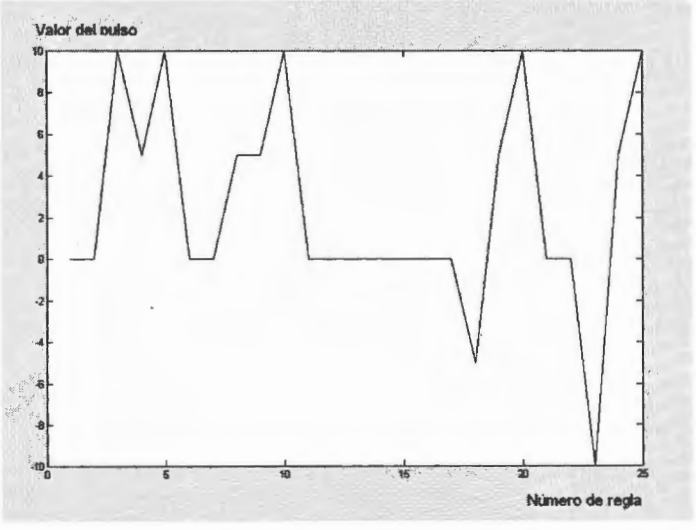


Figura 3.8 Funciones de pertenencia para el pulso de la rueda izquierda

Después del diseño realizado en MATLAB, se procedió a la implementación del controlador de posición en LabVIEW, considerando las ventajas de la intercomunicación inalámbrica que ofrece a través de las herramientas de conexión Bluetooth, y el procesamiento en paralelo de los algoritmos. La interfaz gráfica diseñada para el controlador en LabVIEW puede apreciarse en la Figura 3.9, tanto en su forma gráfica real como en un esquema a bloques que describe cada una de sus partes.

El último paso del diseño consistió en la validación del controlador a través de la conexión del mismo con el modelo matemático de la planta presentado en los apartados a continuación, el cual también fue programado con MATLAB y posteriormente implementado en LabVIEW. El modelo de la planta permite también la obtención de la retroalimentación del movimiento del robot, ya que actúa como un observador del sistema.

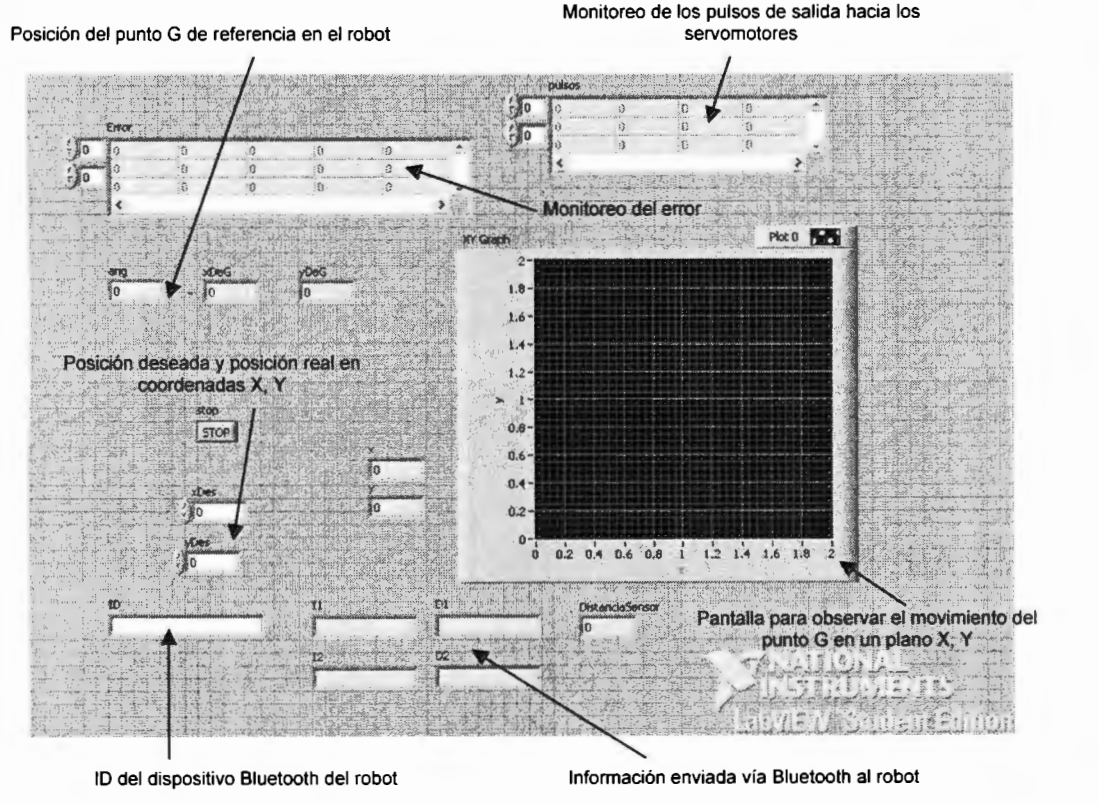


Figura 3.9 Controlador de posición implementado en LabVIEW

En la Figura 3.10 se muestra un ejemplo de las simulaciones realizadas para comprobar el funcionamiento del controlador de la posición, graficando un punto correspondiente al movimiento del robot en un plano coordenado *X, Y* desde el origen hacia una referencia de 0.5 m en *X* y 0.5 m en *Y*.

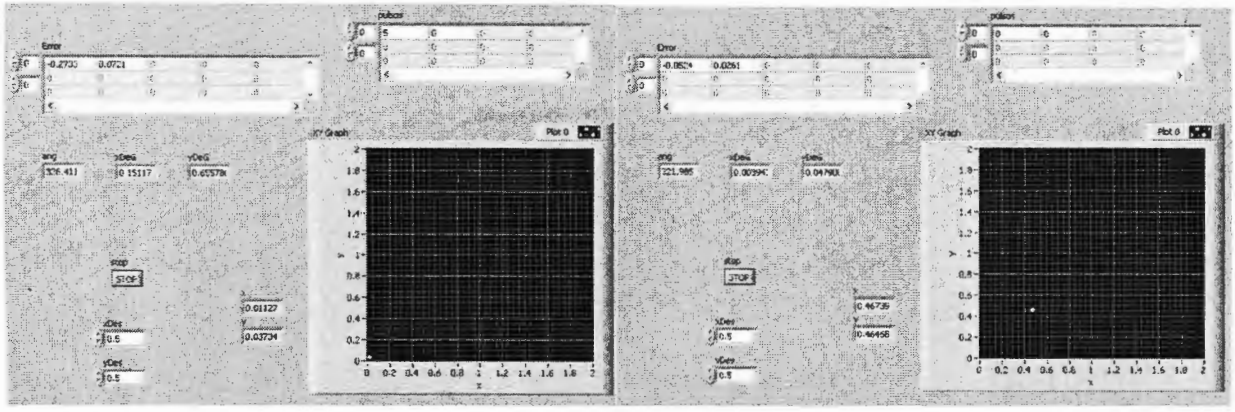


Figura 3.10 Simulación del funcionamiento del controlador de la posición del robot

El programa completo del controlador de posición usado para la implementación se encuentra en el CD anexo a este texto.

3.1.1.2 Diseño del controlador de navegación para la evasión de obstáculos

Para este controlador nuevamente se procedió a seleccionar las entradas y salidas, correspondientes a la distancia medida en cada uno de los tres sensores ultrasónicos del robot, localizados a la derecha, al frente y a la izquierda, así como los pulsos de los servomotores derecho e izquierdo para el movimiento de esquivación del robot, según se muestra en la Figura 3.11.

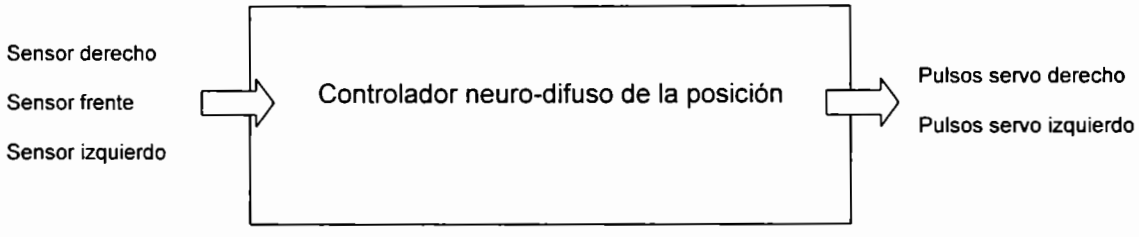


Figura 3.11 Entradas y salidas del controlador de evasión de obstáculos

Posteriormente se emplearon tres conjuntos difusos correspondientes a las distancias denominadas Muy cerca, Cerca y Lejos, para determinar las funciones de pertenencia en la

parte de difusificación en la entrada. Estas funciones se ajustan en línea con el algoritmo de agrupamientos difusos FCM, y se entrenan en una red neuronal de 16 neuronas.

Una vez realizado el proceso de la incorporación del FCM y la red neuronal artificial, se procedió a diseñar la matriz de asociación difusa o FAM, empleando la misma Ecuación (22) para determinar los pulsos a partir de las reglas activadas. Las funciones de pertenencia correspondientes al pulso de la rueda izquierda y al de la rueda derecha pueden apreciarse en la Figura 3.12 y 3.13, respectivamente.

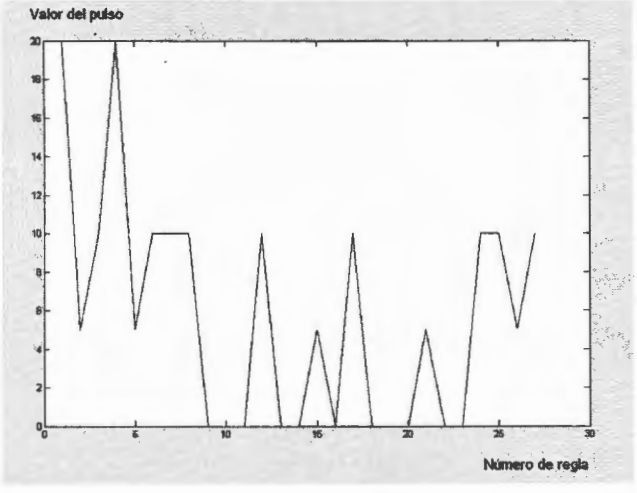


Figura 3.12 Funciones de pertenencia para el pulso de la rueda izquierda

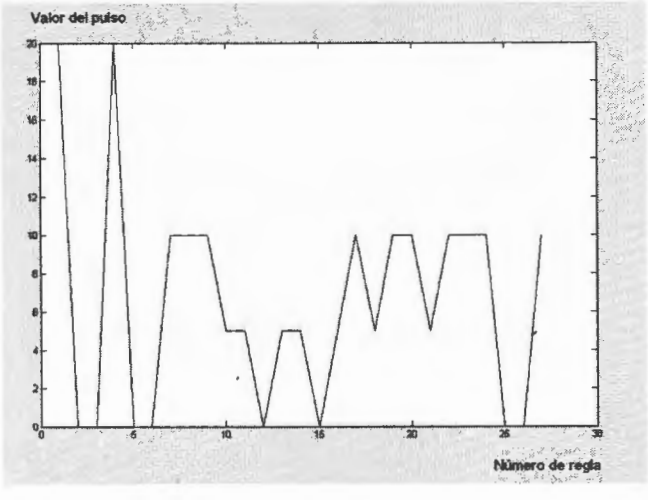


Figura 3.13 Funciones de pertenencia para el pulso de la rueda derecha

Finalmente, el controlador fue implementado en LabVIEW para ser validado en pruebas reales con el robot. El programa del controlador de evasión de obstáculos se encuentra en el CD anexo a este texto.

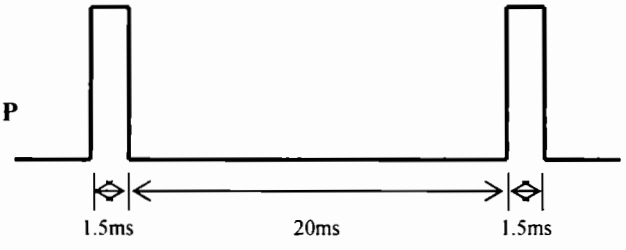
3.1.2 Modelación matemática de la planta

Para poder ajustar los parámetros del controlador, fue necesario encontrar un modelo matemático que describiera el comportamiento de navegación del robot. Para ello, se llevaron a cabo pruebas para caracterizar el comportamiento del servomotor, y posteriormente se diseñó un modelo que indicara la posición final a partir del movimiento del servomotor.

3.1.2.1 Función de aproximación del servomotor

El modelado de la planta se basa en el sentido de encontrar una función que sea capaz de describir la relación entre dos pulsos de entrada que activan el movimiento de las llantas y el desplazamiento del robot tomando en cuenta la dirección de giro.

Antes de comenzar a adquirir datos que permitan observar el comportamiento de la planta, es necesario calibrar los dos servomotores por los cuales se provee el desplazamiento del robot [31]. Para llevarlo a cabo, se programa una rutina de pulsos que serán enviados a los servomotores; donde lo que se busca es detener el movimiento del eje al cambiar el valor de la resistencia variable del interior del motor. En la Figura 3.14 se muestra el pulso continuo de referencia y el programa de calibración.




```
'{$STAMP BS2}
'{$PBASIC 2.5}

DO
   PULSOUT 12, 750
   PAUSE 20
LOOP
```

Figura 3.14 (A) Pulso de prueba para calibrar los servomotores. (B) Programa del pulso de prueba para el Basic Stamp 2

Una vez que los servomotores son calibrados, es posible definir la entrada como el tiempo de activación p del ciclo programado mostrado en la Figura 3.15 y la salida como el ángulo de giro del eje θ. Con base en estas variables, se observaron 29 tiempos de activación con 3 muestras cada una. En la Tabla 3.2 se muestran los datos obtenidos.

```
'{$STAMP BS2}
'{$PBASIC 2.5}

contador    VAR    Byte

FOR contador = 1 TO p
   PULSOUT 12, 650
NEXT
```

Figura 3.15 Programa utilizado para determinar la relación entre p y θ

A partir del promedio de cada observación, se obtuvo la curva aproximada del comportamiento del servomotor; mediante la cual se realizó una aproximación lineal por el método de mínimos cuadrados. La ecuación (21) es resultado de la función lineal observada en la Figura 4.16:

$$\theta = 11.927p + 9.220 \qquad (21)$$

De los datos adquiridos (Figura 3.16), se determina que la desviación estándar promedio de las muestras es de 0.97° (0.0169 rad). Este dato es importante para conocer cuánta diferencia existirá entre las mediciones de los ángulos para un tiempo de activación p dado.

3.1.2.2 Modelo de la planta

El robot se puede modelar de forma matemática mediante parámetros obtenidos por parte de la planta real. En la siguiente sección se explica el procedimiento y resultados de la modelación.

3.1.2.2.1 Funcionamiento a bloques del sistema

Para los propósitos que se requieren en la manipulación y control del sistema, se define a la variable de entrada $u = [u_1 \; u_2]^T$ como el vector de tiempos máquina que se requieren para mover cada una de las llantas del robot y la variable de salida $y = [y_1 \; y_2 \; y_3]^T$ como el vector de posición del robot en el escenario de navegación (plano de dos dimensiones) con extensión del ángulo de dirección del robot con respecto a un plano coordenado *M* en $\Re^2$ definido en los estados iniciales.

Tiempo de Activación	Número de Muestras (Ángulo)			Promedio De las Muestras
	1	2	3	
1	16	15	14	15.00
2	33	31	33	32.33
3	49	50	50	49.67
4	64	62	62	62.67
5	71	77	72	73.33
6	90	91	90	90.33
7	105	105	105	105.00
8	116	115	117	116.00
9	126	128	127	127.00
10	137	137	137	137.00
11	151	151	151	151.00
12	162	160	163	161.67
13	173	172	172	172.33
14	184	186	186	185.33
15	198	198	197	197.67
16	207	209	210	208.67
17	220	222	221	221.00
18	231	231	230	230.67
19	242	240	242	241.33
20	254	253	254	253.67
21	264	265	264	264.33
22	275	277	277	276.33
23	284	285	288	285.67
24	297	297	295	296.33
25	305	307	308	306.67
26	317	317	318	317.33
27	331	330	332	331.00
28	344	343	344	343.67
29	355	356	354	355.00

Tabla 3.2 Datos adquiridos para la modelación del servomotor

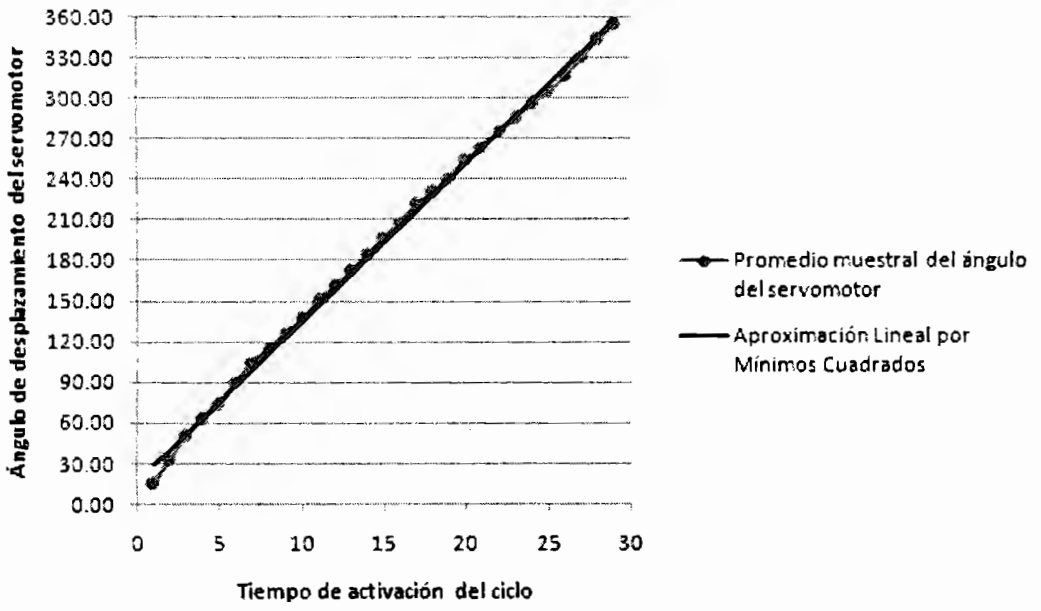


Figura 3.16 Gráfica de los ángulos con respecto al tiempo de activación y curva de la aproximación lineal

En general, se puede dividir la acción de la planta en dos grandes bloques: (a) el modelo de desplazamiento angular y (b) la ubicación en el plano *M*. La Figura 3.17 muestra el diagrama a bloques del sistema.

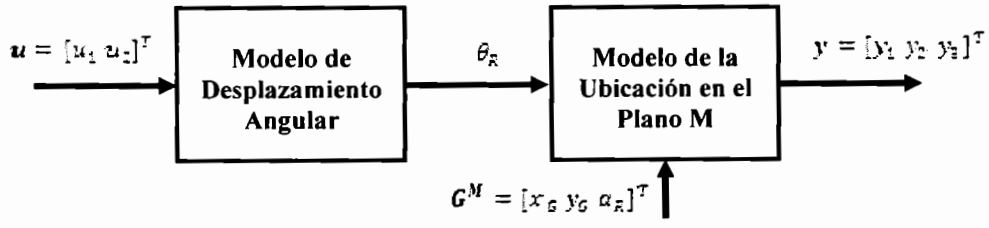


Figura 3.17 Diagrama a bloques del modelo de la planta

3.1.2.2.2 Modelo de desplazamiento angular

La variable de entrada $u = [u_1\ u_2]^T$ contiene la información sobre el tiempo en que estará activa la rotación de cada llanta. Con respecto a la Figura 3.18, se define a u_1 como el

tiempo de activación de la rotación de la llanta izquierda; mientras que u_2 representa el tiempo de activación de la rotación de la llanta derecha. Por otra parte, la variable de salida de este bloque $y_d = \theta_R$ es el desplazamiento angular relativo del plano *R* en $\Re^2$ con centro en el origen G^M, mapeo que se hace con respecto al robot mismo. En otras palabras, sencillamente este bloque determina el desplazamiento angular relativo que tuvo el robot con respecto a la ubicación anterior al movimiento y el valor de la posición después del movimiento (Figura 3.18).

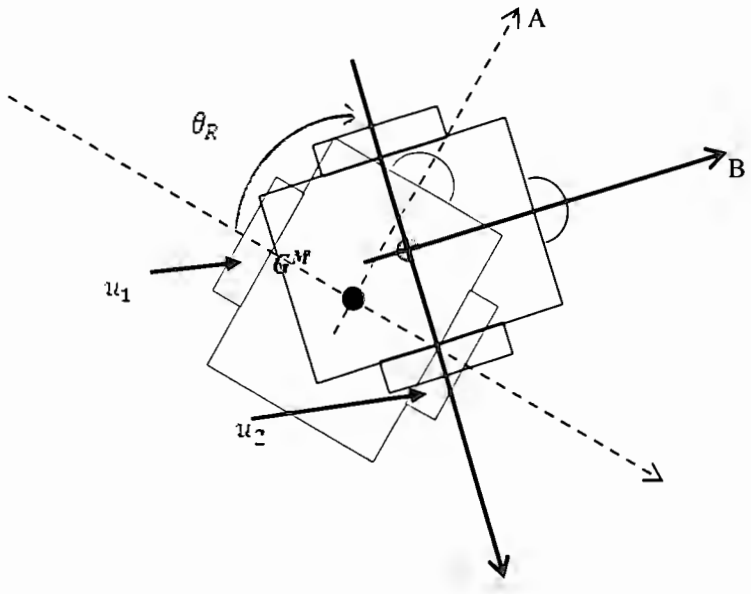


Figura 3.18 Modelo de desplazamiento angular del robot después de aplicarse un conjunto de pulsos u a las llantas. El punto G^M representa el punto de interés del robot; a partir de él, se define el plano R. (A) Plano R antes de la entrada de pulsos, (B) Plano R después del movimiento producido por los pulsos.

Para determinar el modelo de desplazamiento angular es importante restringir el tipo de movimiento. Para ello, se definen los siguientes criterios:

i) El robot se desplaza hacia adelante o hacia atrás. Es decir, el vector u contiene a sus elementos con el mismo valor.

ii) El robot gira hacia la izquierda; entonces, $u_1 = 0$ y $u_2 \neq 0$.

iii) El robot gira hacia la derecha; entonces, $u_1 \neq 0$ y $u_2 = 0$.

iv) El robot se puede quedar estático. Es decir, el vector u contiene a sus elementos con el valor de cero.

v) El desplazamiento lineal o el giro hacia adelante, se cumple cuando el signo del vector es positivo.

vi) El desplazamiento lineal o el giro hacia atrás, se cumple cuando el signo del vector es negativo.

Bajo estas condiciones, se puede modelar la correspondencia entre el vector de pulsos y el desplazamiento angular. Con respecto al análisis de la sección 3.1.2.1 sobre el modelo lineal del servomotor, la función que responde al desplazamiento angular de una llanta con respecto a un pulso p se define en (22):

$$\theta_{LLANTA} = \begin{cases} 11.927[abs(p_{LLANTA})] + 9.220 & p_{LLANTA} \neq 0 \\ 0 & p_{LLANTA} = 0 \end{cases} \tag{22}$$

Con respecto a ello, se pueden contemplar los criterios de desplazamiento del robot. La ecuación (23) muestra la función del ángulo de la llanta en términos del vector, extensión de la ecuación (22); en (24) se comprende la relación entre el arco de desplazamiento de las llantas y el ángulo de desplazamiento de ellas con un radio de llanta $R_{LL} = 0.03m$; finalmente, se muestra el ángulo de desplazamiento relativo del robot (25) con la longitud del eje (dimensión entre llanta y llanta) de $R_{eje} = 0.0105m$:

$$\theta_{LL} = 11.95|u| + 30.94;\ \theta_{LL} = \begin{bmatrix} \theta_{IZQ} & \theta_{DER} \end{bmatrix}^T \tag{23}$$

$$S_{LL} = R_{LL}\theta_{LL};\ S_{LL} = \begin{bmatrix} S_{IZQ} & S_{DER} \end{bmatrix}^T \tag{24}$$

$$\theta_R = \begin{cases} \text{sgn}(u)\dfrac{S_{LL}}{R_{eje}} & si\ \ u_1 \neq u_2 \\ 0 & si\ \ u_1 = u_2 \end{cases}. \tag{25}$$

3.1.2.2.3 Ubicación en el plano M3.1

El desplazamiento absoluto del robot (es decir, conocer la nueva posición del robot) depende de la posición actual del robot en términos del plano *M*. Por lo que se define un punto de interés G^M que será el valor de origen del plano euclidiano *R* (Figura 3.19). Asimismo, el punto G^M lleva consigo el valor de la dirección con respecto al plano *M*, donde este ángulo es la dirección del robot (Figura 3.19). Matemáticamente G^M es un vector de posición con el valor del ángulo de dirección (26) visto desde un plano coordenado *M*:

$$G^M = [x_G \;\; y_G \;\; \alpha_R]^T \tag{26}$$

Es decir, G^M es un vector que muestra la relación entre el plano *M* y el plano *R*; en la Figura 4.19 se muestran los planos antes mencionados.

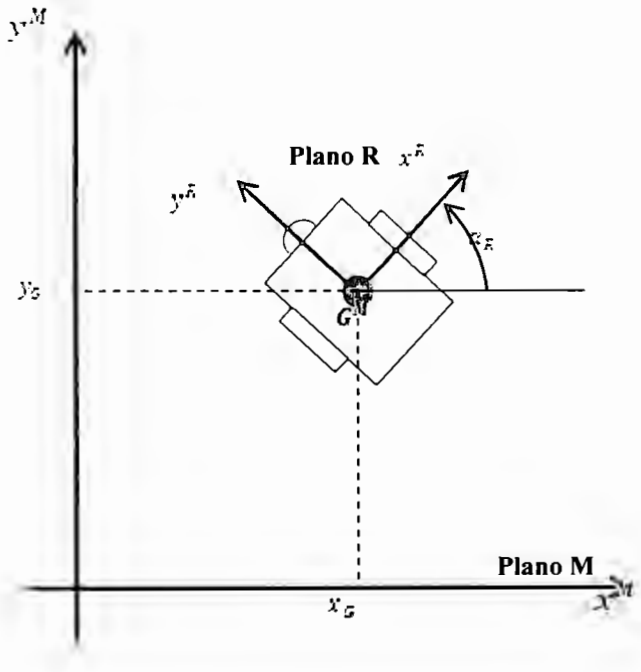


Figura 3.19 Relación entre el plano M y el plano R, ambos en $\Re^2$

Sea el plano *A*, el plano *R* antes del movimiento del robot y el plano *B*, la representación del plano *R* después del movimiento (definidos en la Figura 3.19); es posible determinar la posición del punto de interés después del movimiento, es decir, conocer a G^M de *B* en *A*. A este mapeo de G^M se le define como $\beta : B \rightarrow A$. Entonces, se obtienen tres casos de mapeo, según el criterio de desplazamiento.

En el primer caso, cuando el desplazamiento es lineal hacia adelante o hacia atrás, la ecuación (27) define el mapeo. En el segundo caso, cuando el robot gira hacia la izquierda el mapeo se observa en (28) y en el caso cuando el robot gira hacia la derecha el mapeo cumple con (29).

$$\beta = \begin{bmatrix} 0 \\ S_{LL} \end{bmatrix} \tag{27}$$

$$\beta = \begin{bmatrix} \left(\frac{R_{eje}}{2}\right)(\cos\theta_R - 1) \\ \left(\frac{R_{eje}}{2}\right)\sin\theta_R \end{bmatrix} \tag{28}$$

$$\beta = \begin{bmatrix} \left(\frac{R_{eje}}{2}\right)(-\cos\theta_R + 1) \\ \left(\frac{R_{eje}}{2}\right)\sin\theta_R \end{bmatrix} \tag{29}$$

La transformación de coordenadas entre el plano *R* y el plano *M*, se puede resolver mediante matrices homogéneas de transformación [32], [33]. El mapeo $\varphi : R \rightarrow M$, se puede realizar matemáticamente con (30) donde la matriz de transformación es (31) y el vector aumentado (32); obteniendo de (30) el mapeo (33):

$$\varphi' = T\beta' \tag{30}$$

$$T = \begin{bmatrix} \cos\alpha_R & -\sin\alpha_R & x_G \\ \sin\alpha_R & \cos\alpha_R & y_G \\ 0 & 0 & 1 \end{bmatrix} \tag{31}$$

$$\beta' = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{32}$$

$$\varphi = \begin{bmatrix} \varphi'_1 & \varphi'_2 \end{bmatrix}^T . \tag{33}$$

Las coordenadas de φ corresponden a la nueva posición de la planta en el plano *M*. Finalmente, para obtener el nuevo ángulo de dirección del robot se sigue la función:

$$\alpha_R^{k+1} = \begin{cases} \alpha_R^k & u_1 = u_2 \\ \alpha_R^k + \theta_R & u_1 = 0, u_2 \neq 0 \\ \alpha_R^k - \theta_R & u_1 \neq 0, u_2 = 0 \end{cases} . \tag{34}$$

El vector de salida de este bloque (y del modelo de la planta) se define como $y = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}^T$; donde los valores de sus elementos cumplen con:

$$y = \begin{bmatrix} \varphi \\ \alpha_R^{k+1} \end{bmatrix} . \tag{35}$$

3.1.2.3 Validación del modelo de la planta

Para concluir con la modelación de la planta, es necesario probar que las estimaciones son lo suficientemente similares a los valores reales de la posición del robot en un ambiente en dos dimensiones.

3.1.2.3.1 Metodología empleada

1. Realizar una serie de mediciones en un escenario sin relieves ni protuberancias que no tenga ningún grado de inclinación.
2. Trazar un plano coordenado *M* en el escenario y proponer el valor de origen del mismo.

3. Para las mediciones, determinar una serie de puntos en dos dimensiones (x_0, y_0) dentro del plano coordenado.
4. Encontrar el punto de interés G en el robot de prueba.
5. Obtener los valores del radio de la llanta y la longitud del eje en el robot.
6. Para cada punto de prueba, posicionar el punto de interés del robot y determinar la dirección del mismo.
7. Registrar el punto actual de la posición del robot y el ángulo de dirección en términos del desfasamiento del plano coordenado *R* con respecto al plano *M*.
8. Determinar la entrada de pulsos aleatorio que cumpla con los criterios de desplazamiento propuestos.
9. Medir la nueva posición del punto de interés del robot en el plano *M*.

Se realizaron 20 observaciones con distintas posiciones y ángulos de desfasamiento de los planos con el fin de conocer el comportamiento si:

a) El robot se desplaza hacia adelante
b) El robot gira hacia la izquierda
c) El robot gira hacia la derecha

El desplazamiento hacia atrás se prueba bajo el supuesto de que es igual que el punto a) pero en sentido contrario y que la calibración de los servomotores se realizó con éxito de tal forma que tienen una respuesta dual (es decir, tienen el mismo comportamiento con un giro en sentido a las manecillas del reloj y un giro en sentido contrario).

En el Anexo 1 se concentra la tabla de valores actuales, la entrada de pulsos propuesta, los valores después del desplazamiento. Al mismo tiempo, se muestran los valores estimados bajo la simulación del algoritmo del modelo de la planta. En el mismo anexo se encuentran los errores absolutos y relativos. De ellos, se obtuvieron el valor máximo, mínimo y el promedio del error en la estimación. Bajo estos datos, se muestra claramente que en promedio, la

confiabilidad de que la estimación de la posición del robot en el escenario sea correcta es **mayor al 95%.** Con esto, queda validada la modelación lineal de la planta. En la prueba se ocupó un radio de 0.033m y de longitud de eje de 0.0105m.

3.2 Diseño del prototipo físico

Se propuso un nuevo diseño para la carcasa, mostrado en la Figura 3.20, considerando emplear componentes electrónicos disponibles en el mercado, como la tarjeta con el microprocesador Basic Stamp 2 y los servomotores, para lograr un diseño más funcional, barato y flexible que podrá ser utilizado en otros proyectos a futuro.

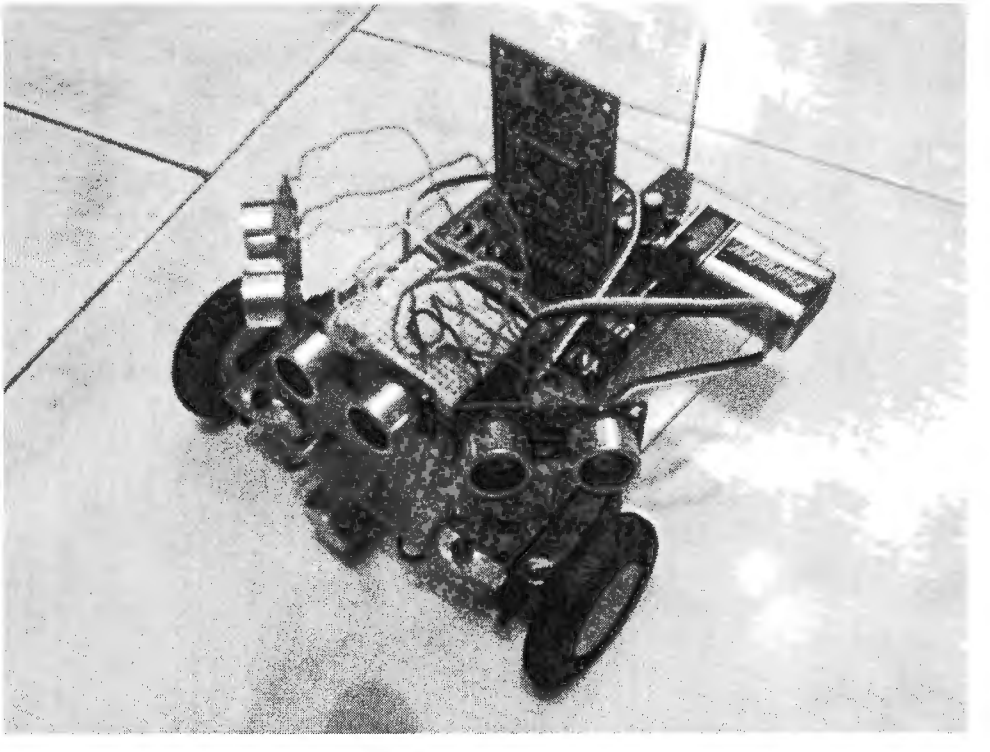


Figura 3.20 Diseño físico implementado

3.2.1 Características físicas de la nueva propuesta

A continuación se muestran diversas facetas del nuevo diseño propuesto con sus respectivas medidas, en las Figuras 3.21, 3.22 y 3.23.

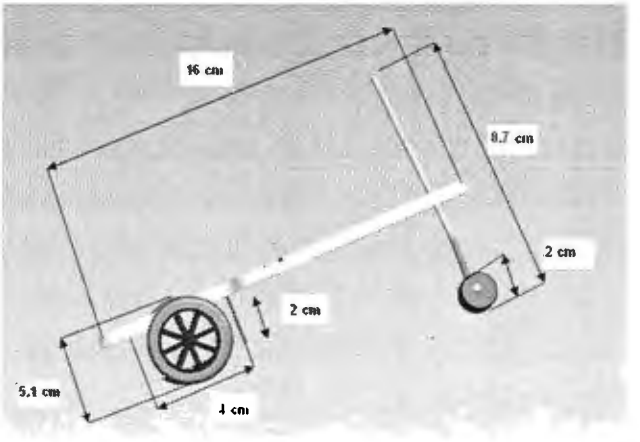


Figura 3.21 Medidas para el nuevo diseño del robot, vista lateral

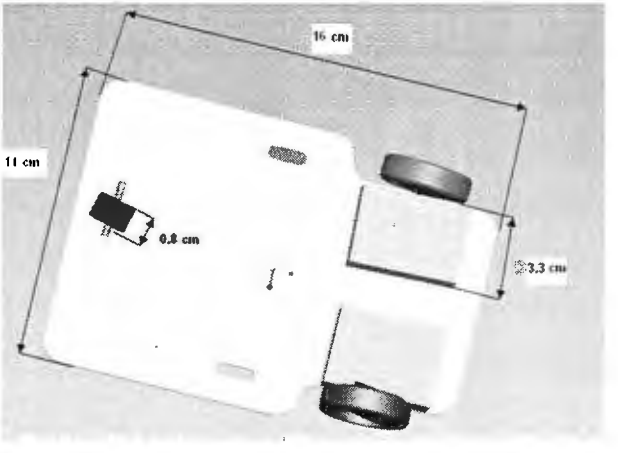


Figura 3.22 Medidas para el nuevo diseño del robot, vista inferior

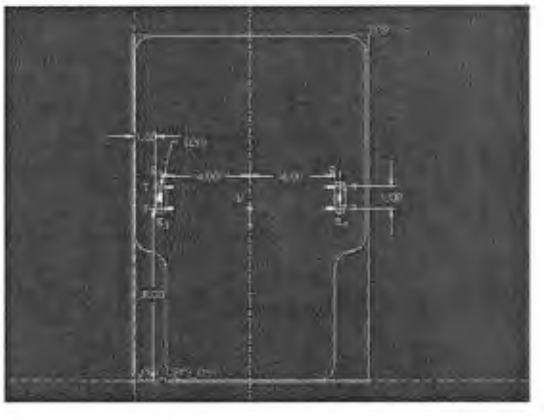


Figura 3.23 Medidas para el nuevo diseño del robot, vista superior

Para el chasis se propusieron como elementos mecánicos los siguientes:

- Dos servomotores
- Dos llantas de polímero
- Llanta trasera con soporte
- Placa de acrílico como chasis
- Dos placas de aluminio para sostener los servomotores
- Tornillos
- Imán de 4 kg en la parte delantera

Se propusieron los materiales enlistados en la Tabla 3.3, al igual que las dimensiones de cada uno de éstos.

	Materiales	***Dimensiones (cm)***
Par de Llantas (*Grand Wing Servo-Tech Co, ltd Replacement Parts*)	**Llanta**: Hule espuma **Centro de la llanta**: Polipropileno	**Llanta**: 5.1 Diámetro interior X 2.8 Diámetro exterior X 1.2 Altura **Centro de la llanta**: 2.8 Diámetro X 1.2 Altura
Soporte Trasero (*Tailwheel Bracket*)	**Soporte**: Nylon **Varilla**: Acero **Collar**: Acero **Arandelas**: Acero	**Soporte**: 3 Largo X 1.9 Ancho X 1.7 Altura **Varilla**: 0.2 Diámetro X 8.7 Largo **Collar**: 0.7 Diámetro X 0.5 Altura **Arandelas**: 0.6 Diámetro X 0.1 Altura
Llanta Trasera (*3/4" Dia. Tailwheel For .40 Size Airplanes*)	**Llanta**: Goma **Centro de la llanta**: Aluminio	**Llanta**: 2 Diámetro exterior X 1.6 Diámetro interior X 0.8 Altura **Centro de la llanta**: 1.6 Diámetro interior X 0.8 Altura
2 Pares de Tornillos (*6-32 X 1/2" Socket Head Cap Screws*)	Acero cubierto de óxido negro	0.5 Diámetro (cabeza) X 2 Altura
2 Pares de Tuercas (*6-32 Steel Hex Nuts*)	Zinc	1 Diámetro X 0.3 Altura
1 Juego de Tuercas con Ajuste (*Bolt Sets with Lock Nuts*)	Acero	0.8 Diámetro X 0.5 Altura
1 Placa de Acrílico	Acrílico	16 Largo X 11 Ancho X 0.3 Altura
1 Placa de Aluminio (*#255 Aluminum 0.16X4X10*)	Aluminio	10 Largo X 2.1 Ancho X 0.1 Altura
2 Tornillos Sujetadores de Llantas	Acero	0.5 Diámetro (cabeza) X 1.5 Altura
2 Tornillos Sujetadores de Equipo Posterior	Acero	0.6 Diámetro (cabeza) X 1.3 Altura

Tabla 3.3 Materiales y dimensiones del chasis

A su vez, se contemplaron los siguientes elementos electrónicos, siendo los dos primeros necesarios para el control de la navegación autónoma, y el tercero parte de la comunicación inalámbrica descrita posteriormente.

- Board of Education con Basic Stamp 2 de Parallax
- Tres sensores ultrasónicos Ping)))
- Tarjeta Embedded Bluetooth de Parallax para Basic Stamp 2

3.2.2 Comparativo de costos

La Tabla 3.4 presenta el costo de un robot Boe-Bot fabricado por la empresa Parallax [34], utilizado en un principio para la etapa de diseño del controlador, y además el precio de un robot con el diseño físico propuesto desglosado en la Tabla 3.5.

Cotizaciones	Precio Unitario (dlls)	Precio Unitario (pesos)	Unidades	Total (pesos)
BOE-Bot Robot Kit Serial and USB compatible	$ 149.95	$ 1,634.46	1	$ 1,634.46
Envío a México *Ups Worl Wide express*	$ 99.75	$ 1,087.28	1	$ 1,087.28
				$ 2,721.73

Tabla 3.4 Cotización de un Boe-Bot de Parallax

Cotizaciones	Precio Unitario (dlls)	Precio Unitario (pesos)	Unidades	Total (pesos)
Board of Education (Serial) - Full Kit	$ 99.95	$ 1,089.46	1	$ 1,089.46
Parallax (Futaba) Standard Servo	$ 12.95	$ 141.16	2	$ 282.31
Envío a México *Ups Worl Wide Express*	$ 68.76	$ 749.48	1	$ 749.48
Par de Llantas (*Grand Wing Servo-Tech Co, ltd Replacement Parts*)		$ 27.00	1	$ 27.00
Soporte Trasero (*Tailwheel Bracket*)		$ 26.00	1	$ 26.00
Llanta Trasera (*3/4" Dia. Tailwheel For .40 Size Airplanes*)		$ 24.00	1	$ 24.00
2 Pares de Tornillos (*6-32 X 1/2" Socket Head Cap Screws*)		$ 12.00	1	$ 12.00
2 Pares de Tuercas (*6-32 Steel Hex Nuts*)		$ 8.00	1	$ 8.00
1 Juego de Tuercas con Ajuste (*Bolt Sets with Lock Nuts*)		$ 12.00	1	$ 12.00
1 Placa de Acrílico		$ 30.00	1	$ 30.00
1 Placa de Aluminio (*#255 Aluminum 0.16X4X10*)		$ 23.00	1	$ 23.00
2 Tornillos Sujetadores de Llantas		$ 0.30	2	$ 0.60
2 Tornillos Sujetadores de Equipo Posterior		$ 0.30	2	$ 0.60
				$ 2,284.45

Tabla3.5 Cotización de un robot con el nuevo diseño

En un principio se planteó trabajar con tres robots para el diseño de la tarea colaborativa por lo que se realizó un comparativo al mayoreo para verificar el ahorro total, lo cual se muestra a continuación en las Tablas 3.6 y 3.7, donde se presentan los costos de la adquisición de tres robots Boe-Bot [34] contra adquirir y ensamblar tres del nuevo diseño.

Cotizaciones	Precio Unitario (dlls)	Precio Unitario (pesos)	Unidades	Total (pesos)
BOE-Bot Robot Kit Serial and USB compatible	$ 149.95	$ 1,634.46	3	$4,903.37
Envío a México *Ups Worl Wide express*	$ 161.74	$ 1,762.97	1	$1,762.97
				$6,666.33

Tabla3.6 Cotización de tres robots Boe-Bot de Parallax

Cotizaciones	Precio Unitario (dlls)	Precio Unitario (pesos)	Unidades	Total (pesos)
Board of Education (Serial) - Full Kit	$ 99.95	$ 1,089.46	3	$ 3,268.37
Parallax (Futaba) Standard Servo	$ 12.30	$ 134.07	6	$ 804.42
Envío a México *Ups Worl Wide Express*	$ 91.20	$ 994.08	1	$ 994.08
Par de Llantas (*Grand Wing Servo-Tech Co, ltd Replacement Parts*)		$ 27.00	3	$ 81.00
Soporte Trasero (*Tailwheel Bracket*)		$ 26.00	3	$ 78.00
Llanta Trasera (*3/4" Dia. Tailwheel For .40 Size Airplanes*)		$ 24.00	3	$ 72.00
2 Pares de Tornillos (*6-32 X 1/2" Socket Head Cap Screws*)		$ 12.00	3	$ 36.00
2 Pares de Tuercas (*6-32 Steel Hex Nuts*)		$ 8.00	3	$ 24.00
1 Juego de Tuercas con Ajuste (*Bolt Sets with Lock Nuts*)		$ 12.00	1	$ 12.00
1 Placa de Acrílico		$ 30.00	1	$ 30.00
1 Placa de Aluminio (*#255 Aluminum 0.16X4X10*)		$ 23.00	1	$ 23.00
2 Tornillos Sujetadores de Llantas		$ 0.30	6	$ 1.80
2 Tornillos Sujetadores de Equipo Posterior		$ 0.30	6	$ 1.80
				$ 5,426.47

Tabla 3.7 Cotización de tres robots con el nuevo diseño

Es posible apreciar, según las tablas anteriores, que existe un ahorro significativo al incorporar un nuevo diseño. Si se compara un solo robot, el ahorro es del 16%, pero cuando la comparación se refiere a los tres robots planteados para el proyecto, el ahorro asciende al 19% del costo total.

3.2.3 Comparativo de pesos

Adicionalmente a los costos, se realizó una comparación respecto al peso del producto que ofrece Parallax [34] y el del diseño propuesto, el cual se puede apreciar en la Tabla 3.8 y la Tabla 3.9.

	Aluminio	Acrílico
Densidad (g/cm^3)	2.7	1.8
Volúmen (11 X 16 X 0.5 cm^3)	88	88
Masa (g)	237.6	158.4

Tabla 3.8 Comparación de pesos de la placa que conforma el chasis del robot

	Kit completo Parallax	Kit armado con Tableta de Acrílico
Peso de equipo sin llantas ni tableta (g)	1144.48	1144.48
Peso de llantas (g)	173.768	103.28
Masa de la base Al vs Acrílico (g)	237.6	158.4
Peso Total (g)	1555.848	1406.16
Ahorro en Peso	10%	

Tabla 3.9 Comparación del peso total de un robot

Es evidente el cambio en el peso del robot al utilizar un material acrílico, ayudado además por el tamaño y material esponjoso de las llantas, un polímero más liviano. Este cambio facilita y a la vez optimiza el movimiento de los servomotores, así como la maniobrabilidad de los robots. Cabe destacar que el diseño realizado en acrílico resulta más agradable a la vista del usuario final, siendo esta muchas veces una característica importante en la presentación de los proyectos.

3.3 Diseño de la comunicación

La tecnología Bluetooth es un estándar global de comunicación por ondas de radio de corto alcance. El protocolo Bluetooth corresponde a la especificación oficial IEEE 802.15.1, surgida con el propósito de simplificar la comunicación entre dispositivos [35].

Las especificaciones técnicas del hardware son las siguientes [35]:

- Canal de comunicación de hasta 720 kb/s
- Alcance de 10 metros a 1mW de potencia de transmisión, o hasta 100 con repetidores y potencia de 100 mW a 1 W
- Frecuencia de 2.4 a 2.48 GHz
- Posibilidad de transmisión Full Duplex
- Circuitos CMOS de 9x9 mm de tamaño
- Bajo consumo de energía

El hardware se compone de un dispositivo de radio, que modula y transmite la señal, y un controlador digital que se encarga del proceso de la señal y de la interfaz con el dispositivo al cual está conectado el sistema Bluetooth [35].

Las tecnologías inalámbricas Bluetooth y Wi-Fi son tecnologías complementarias. La tecnología Bluetooth fue diseñada como un sustituto de los cables de conexión entre computadoras, teléfonos móviles, agendas electrónicas, audífonos y otros periféricos. Los transmisores Wi-Fi pueden tener un alcance superior al de Bluetooth, de 45m en interiores y hasta 90m en exteriores, lo que sumado a su ancho de banda implica el uso del protocolo Wi-Fi para hacer redes inalámbricas [35].

3.3.1 Selección y diseño de la comunicación

Diversas frecuencias y protocolos de comunicación fueron considerados para lograr comunicar la computadora con los robots y realizar el control, así como para la

intercomunicación entre los robots. Entre estas se encuentran el protocolo Wi-Fi, radiofrecuencias de uso común, y el uso de cables seriales.

La alternativa seleccionada fue la comunicación inalámbrica vía Bluetooth, debido al costo, hardware necesario y facilidad de uso de la misma. Las ventajas y desventajas evaluadas en el proceso de selección pueden apreciarse en la Tabla 3.10.

	Ventajas	*Desventajas*
Hardware	Compatibilidad con Basic Stamp 2 y LabVIEW	Costos
Software	LabVIEW tiene bloques predeterminados para la comunicación Bluetooth	Aprender a usar los bloques de LabVIEW
Características físicas del protocolo	Sirve para hacer redes locales	No permite objetos entre las antenas, el rango de alcance es corto
Transmisión de la información	Los datos llegan enteros, no hay ruido o interferencias presentes durante el intercambio de información	El tiempo de intercambio es muy lento (hay un retardo considerable), es necesario un protocolo creado para que el software sepa cuando enviar y recibir datos

Tabla3.10 Ventajas y desventajas del uso del protocolo Bluetooth para la comunicación inalámbrica

Para la comunicación inalámbrica vía Bluetooth se emplearon los siguientes componentes, tanto de software como de hardware:

- LabVIEW Bluetooth Toolkit
- Adaptador USB-Bluetooth Belkin
- Tarjeta Embedded Bluetooth de Parallax para Basic Stamp 2

La Figura 3.24 muestra el diagrama de conexiones empleado para la implementación de la comunicación inalámbrica.

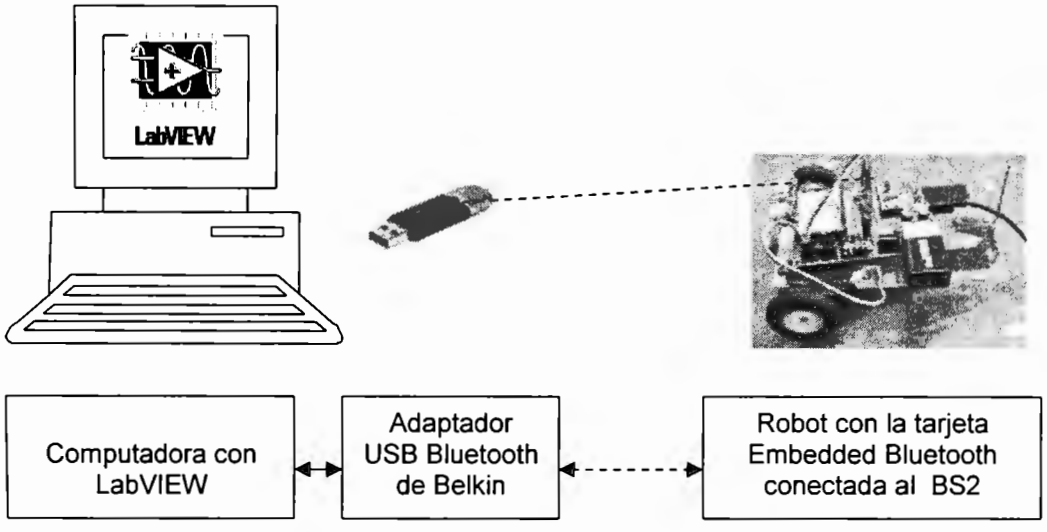


Figura 3.24 Diagrama de la conexión para la comunicación inalámbrica

3.3.2 Protocolo de envío y recepción de información

Para lograr un intercambio efectivo de información entre LabVIEW y el Basic Stamp 2 se diseñó un protocolo que implica el uso de palabras clave y el tamaño de los datos a enviar, el cual se muestra esquemáticamente en la Figura 3.25.

Una vez realizado el establecimiento de la conexión entre LabVIEW y el Basic Stamp 2, se procede al envío de las tres distancias detectadas por los sensores del robot, y posteriormente se recibe la señal de corrección reflejada en el movimiento de los servomotores, proveniente del controlador total. Cabe mencionar que para evitar la pérdida de información fue necesario el envío de la longitud de las distancias antes del dato, para que LabVIEW supiera cuántos bytes de información debía recibir. Por otra parte, el Basic Stamp 2 envía una señal correspondiente a la letra *g* para indicarle a LabVIEW que recibió el dato del pulso, y que ya puede mandar el siguiente dato.

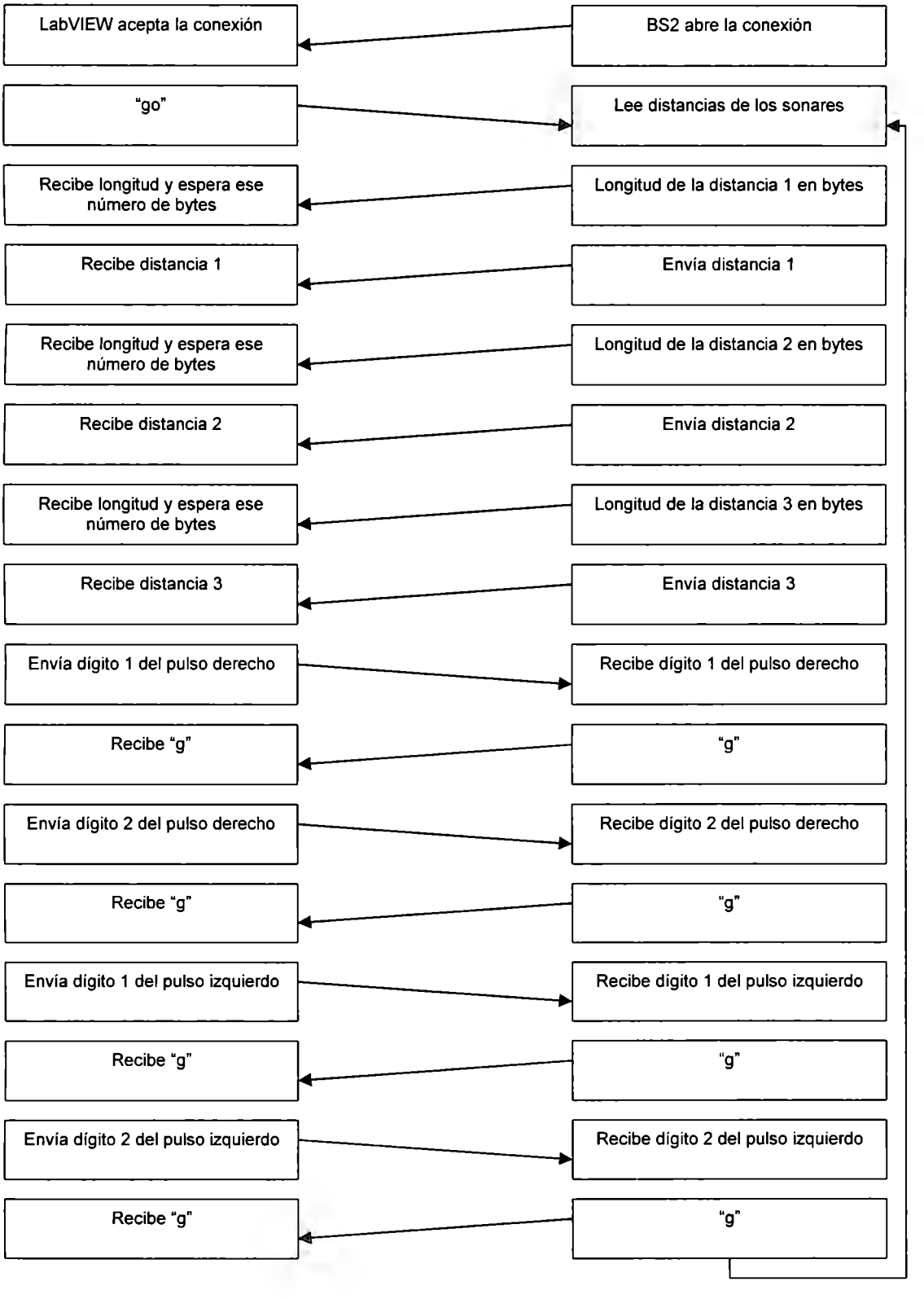


Figura 3.25 Esquema del protocolo para el intercambio de información entre LabVIEW y el Basic Stamp 2

3.4 Diseño de la tarea colaborativa simple

El uso de varios micro-robots para desempeñar tareas simples se ha investigado en varios institutos alrededor del mundo. Las tareas más comunes incluyen la búsqueda de diversos objetos en el ambiente para ser llevados a otra ubicación, recorrer un ambiente para desarrollar una tarea sobre el mismo, y buscar objetos para hacer otra tarea con los mismos. Estos robots pueden ser empleados para tareas como limpieza, patrullaje, ensamble de piezas, entre otros. Cabe mencionar que la aplicación seleccionada se enfoca a la parte de búsqueda y recolección de objetos, puesto que la versatilidad de la misma permite salvaguardar la integridad física de personas y facilitar tareas como el manejo de deshechos tóxicos, remoción de escombros, por mencionar algunas [36].

3.4.1 Descripción general

La idea general de la tarea colaborativa simple seleccionada para ejemplificar el uso del controlador neuro-difuso diseñado en el proyecto implica la detección y recolección, a través del uso de sensores ultrasónicos, de pequeños objetos que se encuentran en cierta ubicación, empleando intercomunicación entre los controladores para la repartición de los objetos a recolectar. A continuación se describen las características seleccionadas para el ambiente, el cual puede observarse gráficamente en la Figura 3.26.

3.4.1.1 Características físicas de la prueba

De acuerdo con el tamaño de los robots empleados para el proyecto, se plantearon las características físicas de la tarea colaborativa simple a desempeñar de la siguiente forma:

- Superficie de 2x2 m, perfectamente lisa
- Paredes perimetrales de 15 cm de altura
- Cinco objetos esféricos de unicel con incrustaciones metálicas
- Espacio no muy iluminado

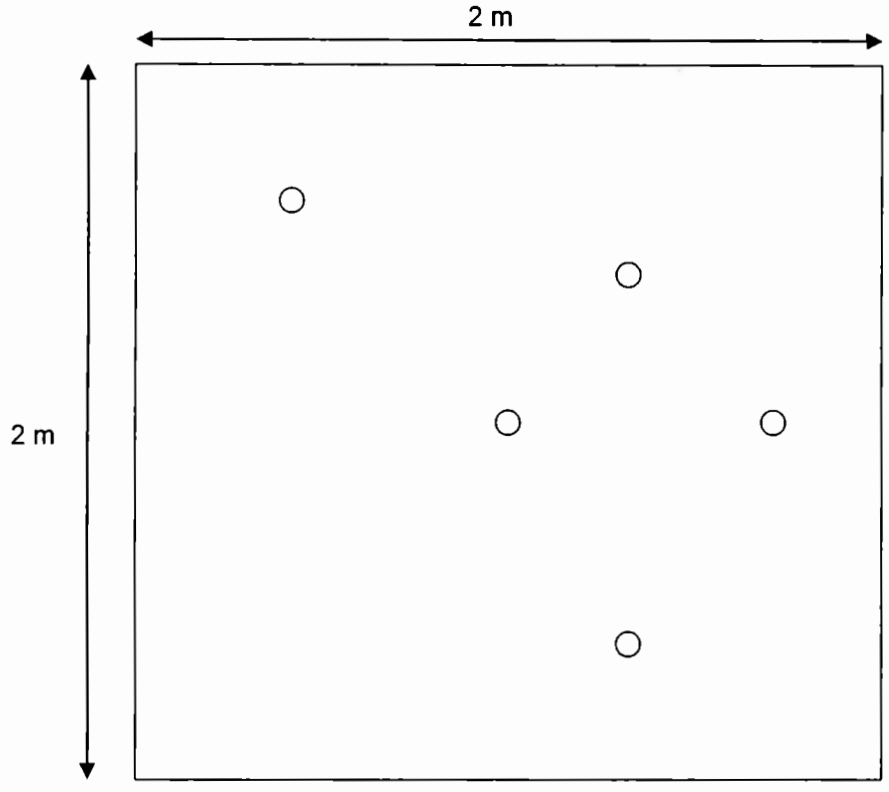


Figura 3.26 Escenario para las pruebas de la tarea colaborativa

En la distribución y uso de los objetos de unicel se consideraron las siguientes restricciones:

- Separación mínima entre objeto y objeto de 10 cm.
- Para la tarea colaborativa se emplean 5 objetos con una distribución predeterminada, mostrada en la Figura 3.27.

3.4.2 Consideraciones preliminares al diseño del algoritmo

Para determinar el uso de los sensores a emplear en la tarea colaborativa se evaluaron distintas alternativas como los infrarrojos, ultrasónicos e incluso una cámara. Por cuestiones de costos y limitaciones físicas de cada tipo de sensor, se seleccionaron los sensores ultrasónicos como mejor alternativa.

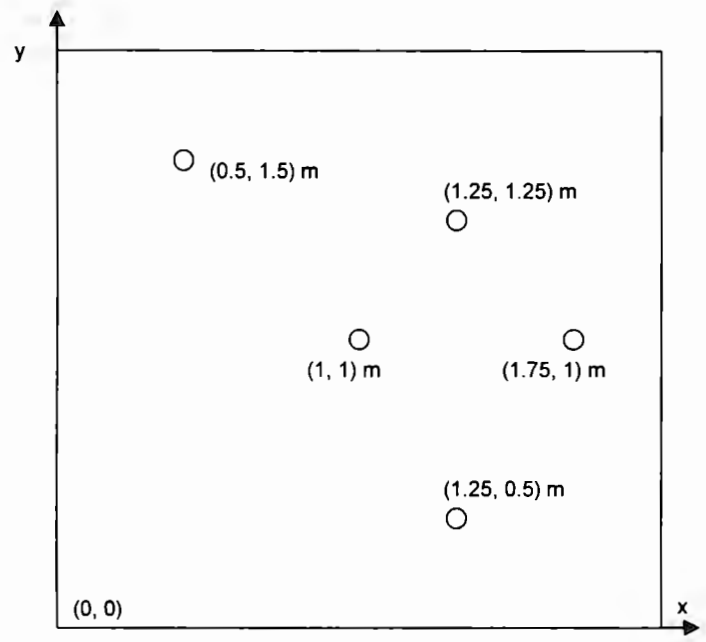


Figura 3.27 Distribución fija de objetos en la pista

3.4.3 Diseño del algoritmo

A continuación se describe el algoritmo general diseñado para la tarea colaborativa simple.

1. Cada robot se asume con una posición inicial determinada en un inicio como su *home* (Figura 3.28) y ninguno de ellos conoce la ubicación de los otros robots.
2. Cada robot determina la distancia que existe entre su posición inicial y cada una de las posiciones de los objetos a recolectar.
3. El robot maestro toma la primera decisión sobre qué objeto ir a recoger (el criterio es comenzar las pelotas más cercanas y al final las pelotas más lejanas).
4. Se comunican los robots para conocer la decisión tomada y elegir un objeto bajo el mismo criterio siempre que no haya sido elegido por un robot anterior o un ciclo de recolección anterior.
5. El siguiente robot decide el objeto a recolectar y comunica a los demás.

6. Dado que durante el proceso no se conoce qué robot terminará primero, se considera:
    a. El primer robot que termine un ciclo de recolección, decide por ir a recolectar otra pelota y comunica a los demás su decisión.
    b. El proceso se repite para cada robot.
    c. El proceso se repite hasta que no haya más pelotas qué recolectar.

    El algoritmo puede ser apreciado de forma gráfica en la Figura 3.29.

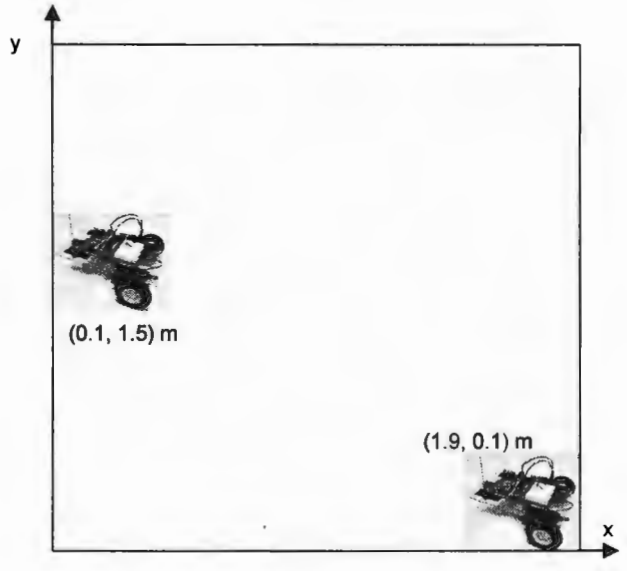


Figura 3.28 Posiciones iniciales predeterminadas de los robots en la pista

3.4.4 Evaluación de los algoritmos involucrados

Para evaluar el desempeño del controlador en la tarea colaborativa simple, así como de la comunicación inalámbrica, se seleccionaron los siguientes parámetros:

- Tiempo total de recolección y depósito de los objetos
- Tiempo de retardo en la comunicación inalámbrica
- Error real en la posición de los robots

Estos parámetros permiten medir el comportamiento del sistema al realizar la tarea colaborativa simple.

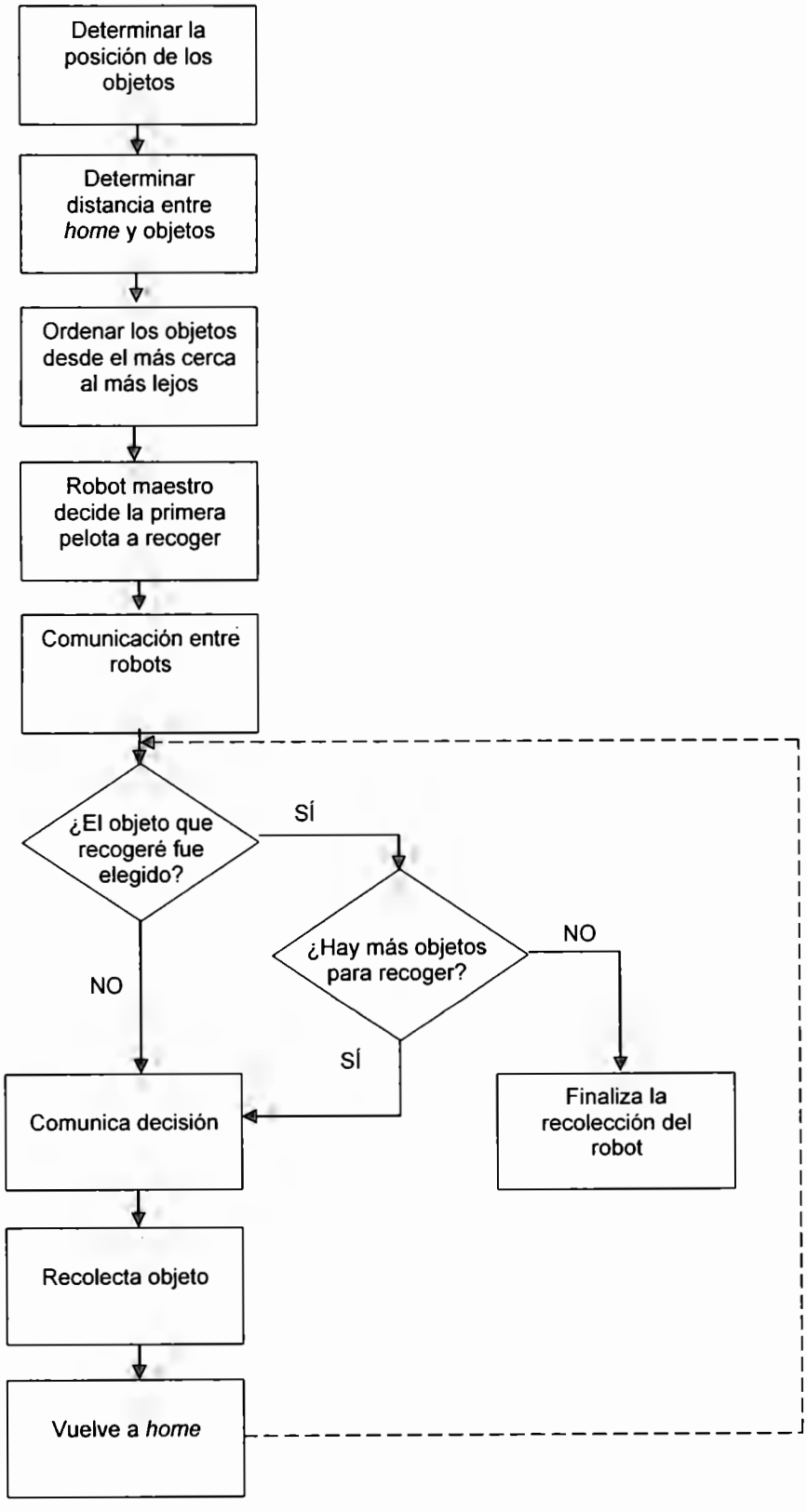


Figura 3.29 Diagrama del algoritmo propuesto para la tarea colaborativa

CAPÍTULO 4

PRUEBAS DE EVALUACIÓN Y RESULTADOS

Se realizaron diversas pruebas para verificar el comportamiento de los algoritmos diseñados en su implementación física, de acuerdo con los criterios de evaluación establecidos en la sección 3.4.4, cuyos resultados se describen en este capítulo. Adicionalmente, se mencionan los problemas principales encontrados a partir de los resultados, así como algunos criterios de implementación a partir del diseño mostrado en los apartados anteriores.

4.1 Controlador total

Después del diseño de los dos controladores neuro-difusos mencionada en el capítulo 4, así como la validación del controlador de posición a partir del modelo matemático de la planta, se procedió a la implementación en LabVIEW del controlador total, uniendo diversos bloques mostrados en la Figura 4.1. El resultado de la implementación puede apreciarse en la Figura 4.2.

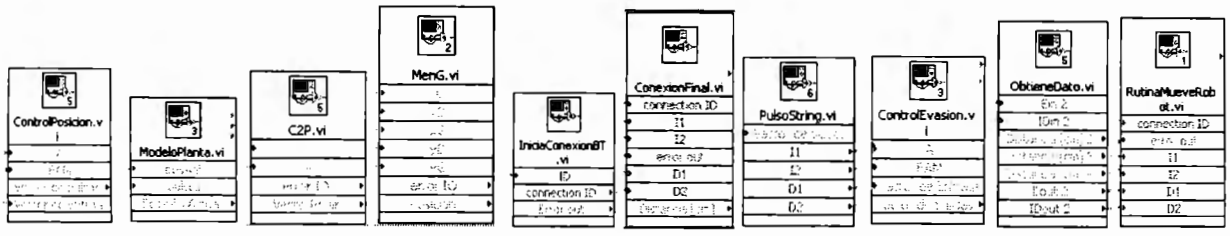


Figura 4.1 Bloques para la implementación del controlador total en LabVIEW

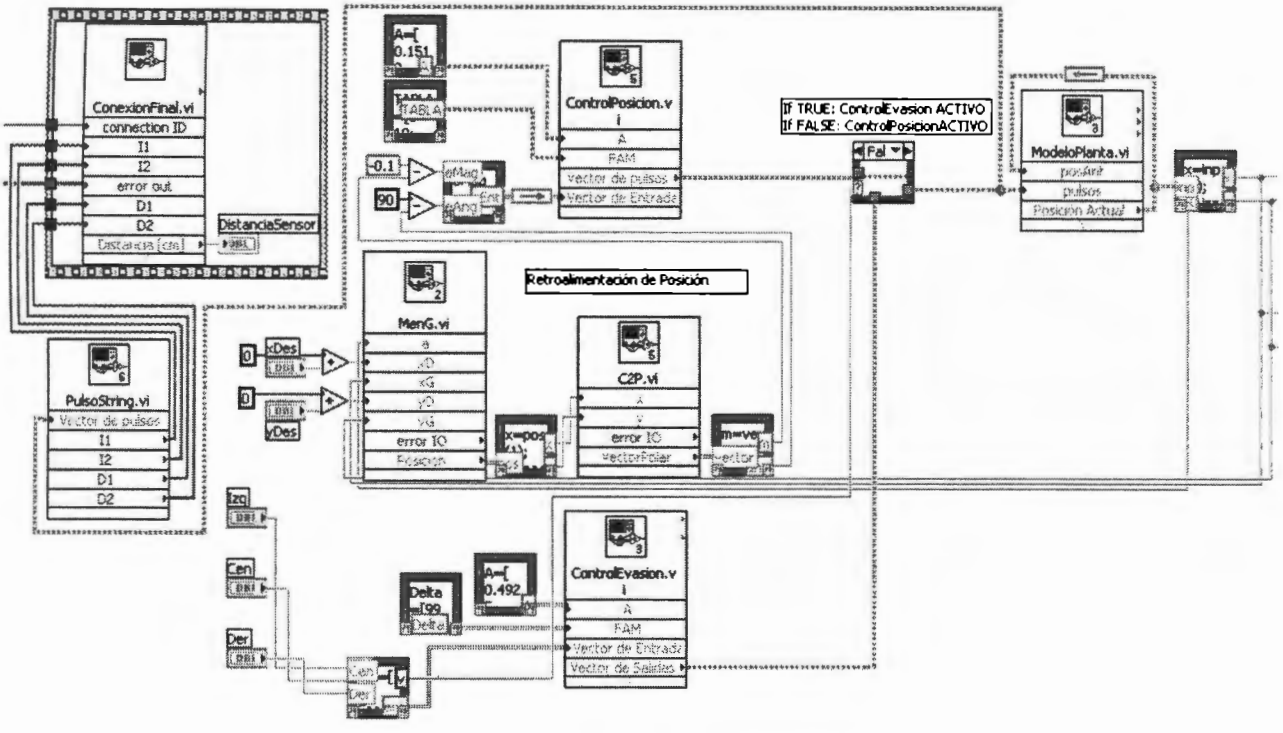


Figura 4.2 Implementación del controlador total en LabVIEW

4.1.1 Controlador de posición

Una vez implementado el controlador total, se realizaron diversas pruebas con el control de la posición del robot, para verificar el error real de desviación respecto a la referencia deseada. Para esto, se realizaron 20 pruebas considerando dos trayectorias:

- Trayectoria recta paralela respecto al eje *y*
- Trayectoria diagonal a 45° respecto al origen del sistema *x, y*

Se registraron los movimientos que hacen los robots para alcanzar las coordenadas de referencia, resultando un pequeño movimiento oscilatorio en trayectorias rectas respecto al eje *y* (Figura 4.3), y un movimiento de corrección del ángulo en primer lugar y posteriormente de acercamiento en trayectorias diagonales (Figura 4.4).

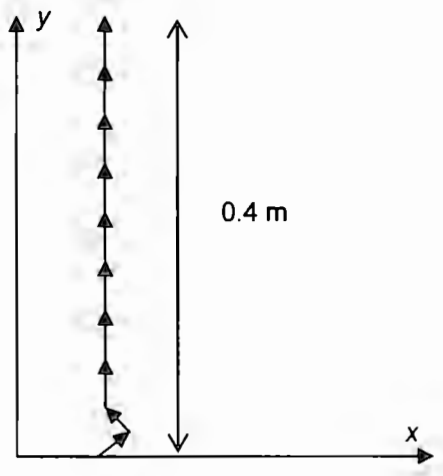


Figura 4.3 Trayectoria seguida por el robot durante un movimiento recto paralelo

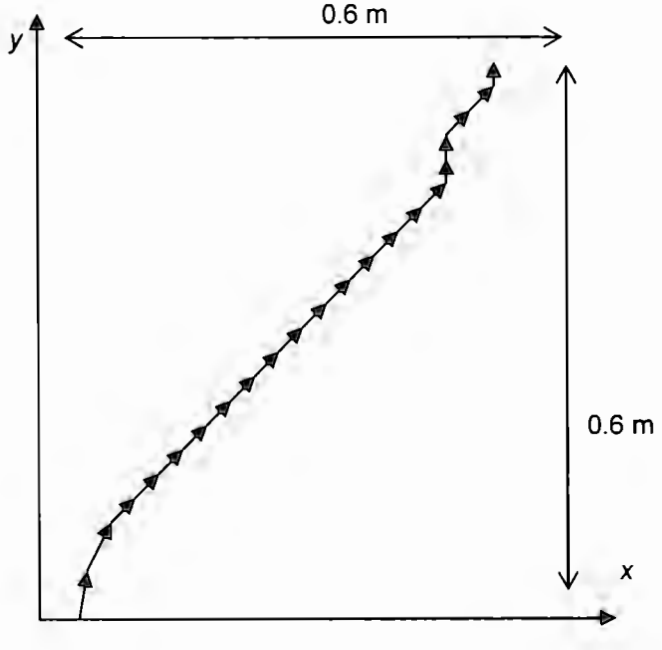


Figura 4.4 Trayectoria seguida por el robot durante un movimiento en dirección diagonal a 45°

Se observó un error promedio de 5.46% para las distintas muestras del movimiento recto paralelo, respecto a la posición deseada y la conseguida después de aplicar el controlador. Por otra parte, se observó un error del 6.2% en el movimiento diagonal tanto en el eje *x* como en el eje *y*. Las mediciones realizadas pueden observarse en las Tablas 4.1 y 4.2.

Durante la prueba, se observó que diversos factores influyen en el error de la posición real, respecto a la posición de referencia introducida en el controlador, el cual resultó distinto al error presentado en la pantalla de la interfaz gráfica. Estos factores son los siguientes:

- Nivel de las baterías: conforme el robot va moviéndose, las baterías se descargan y el error medido en metros crece.
- Superficie (rugosidad, uniones del piso): todas las superficies a nuestro alcance presentan cierto grado de rugosidad y uniones, por lo que el robot tiende a atorarse en instantes pequeños, pero que afectarán la posición final.
- Llanta trasera: la llanta trasera presenta cierta oscilación que provoca una ligera desviación angulada.
- Características físicas del robot: debido a la falta de una alineación perfecta, y debido a que no existen dos motores que sean completamente idénticos en sus parámetros, existe un error presente entre la posición reportada por el controlador y el modelo matemático del sistema respecto a la real.

Los porcentajes obtenidos para el movimiento real resultan de magnitud pequeña, pero es necesario realizar calibraciones en el programa de los robots conforme las condiciones de la batería se van modificando, y antes de hacer una corrida general para la tarea, ya que de esta forma es posible lograr una mayor precisión del robot al llegar a los objetos a recolectarlos y depositarlos.

Movimiento recto paralelo			
Referencia en y (m)	**Valor final (m)**	**Error (m)**	**Error (%)**
0.1	0.095	0.005	5.00
0.2	0.189	0.011	5.50
0.3	0.284	0.016	5.33
0.4	0.379	0.021	5.25
0.5	0.472	0.028	5.60
0.6	0.569	0.031	5.17
0.7	0.658	0.042	6.00
0.8	0.759	0.041	5.13
0.9	0.847	0.053	5.89
1.0	0.947	0.053	5.30
1.1	1.041	0.059	5.36
1.2	1.135	0.065	5.42
1.3	1.225	0.075	5.77
1.4	1.317	0.083	5.93
1.5	1.418	0.082	5.47
1.6	1.519	0.081	5.06
1.7	1.599	0.101	5.94
1.8	1.697	0.103	5.72
1.9	1.805	0.095	5.00
		Promedio	5.47

Tabla 4.1 Mediciones realizadas para el movimiento recto paralelo

Movimiento diagonal a 45°							
Referencia en y (m)	**Referencia en x (m)**	**Valor final en y (m)**	**Valor final en x (m)**	**Error en y (m)**	**Error en y (%)**	**Error en x (m)**	**Error en x (%)**
0.1	0.1	0.094	0.095	0.006	6.00	0.005	5.00
0.2	0.2	0.188	0.186	0.012	6.00	0.014	7.00
0.3	0.3	0.282	0.280	0.018	6.00	0.020	6.67
0.4	0.4	0.375	0.376	0.025	6.25	0.024	6.00
0.5	0.5	0.471	0.472	0.029	5.80	0.028	5.60
0.6	0.6	0.567	0.562	0.033	5.50	0.038	6.33
0.7	0.7	0.651	0.657	0.049	7.00	0.043	6.14
0.8	0.8	0.752	0.748	0.048	6.00	0.052	6.50
0.9	0.9	0.843	0.841	0.057	6.33	0.059	6.56
1.0	1.0	0.940	0.938	0.060	6.00	0.062	6.20
1.1	1.1	1.032	1.030	0.068	6.18	0.070	6.36
1.2	1.2	1.130	1.120	0.070	5.83	0.080	6.67
1.3	1.3	1.218	1.219	0.082	6.31	0.081	6.23
1.4	1.4	1.305	1.312	0.095	6.79	0.088	6.29
1.5	1.5	1.410	1.408	0.090	6.00	0.092	6.13
1.6	1.6	1.502	1.499	0.098	6.13	0.101	6.31
1.7	1.7	1.587	1.586	0.113	6.65	0.114	6.71
1.8	1.8	1.687	1.683	0.113	6.28	0.117	6.50
1.9	1.9	1.785	1.779	0.115	6.05	0.121	6.37
				Promedio	6.16		6.29

Tabla 4.2 Mediciones realizadas para el movimiento diagonal

4.1.2 FAM obtenida con la búsqueda Tabú para el control de posición

Para calcular la matriz de asociación difusa a partir del método de Búsqueda Tabú se propusieron los valores posibles mostrados en la Tabla 4.3.

Pulso izquierdo	*Pulso derecho*
0	0
0	5
0	10
5	0
5	5
10	0
10	10
99	99

Tabla 4.3 Valores posibles para el método Tabú

El periodo de prohibición se propuso como 5. Además, por cada vez que se vuelve a seleccionar un valor que ya ha sido prohibido, se propuso seleccionar un nuevo par de valores posibles de la tabla anterior.

Para el control de posición, se propuso la FAM inicial de las primeras dos columnas de la Tabla 4.4, mencionada en el diseño del controlador, y el resultado del método Tabú se aprecia en las últimas dos columnas de dicha tabla.

Los resultados del movimiento de los robots respecto al uso de la FAM predeterminada y la resultante del método Tabú fueron los mismos.

Por otra parte, se probaron combinaciones al azar como soluciones iniciales, mostradas en las primeras columnas de la Tabla 4.5, para probar el método Tabú, y el resultado puede observarse en las últimas columnas de la misma tabla. Esta solución fue encontrada en diversas ocasiones, por lo que se presume es una opción muy buena partiendo de la idea de que no se tiene una FAM inicial obtenida a partir de la lógica, sino inicializada con valores aleatorios.

INICIAL		*RESULTADO TABÚ*	
Pulso izquierdo	***Pulso derecho***	***Pulso izquierdo***	***Pulso derecho***
0	10	0	10
0	5	0	5
10	10	10	10
5	0	5	0
10	0	10	0
0	10	0	10
0	5	0	5
5	5	5	5
5	0	5	0
10	0	10	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	10	0	10
0	5	0	5
-5	-5	-5	-5
5	0	0	10
10	0	10	0
0	10	0	10
0	5	0	5
-10	-10	-10	-10
5	0	5	0
10	0	10	0

Tabla 4.4 Resultado del método Tabú para el controlador de posición

El método Tabú evalúa la energía o error total del sistema en un intervalo de tiempo determinado y con ciertas características de operación. Lo anterior advierte que el método solo es capaz de dar soluciones locales; para vecindades grandes no es posible obtener una respuesta que minimice el error a distintos puntos de interés del espacio muestral.

Otro de los factores importantes a considerar es que la asignación del subconjunto de movimientos prohibidos no fue realizada de manera *reactiva*, lo cual implica que el periodo de prohibición fue constante para todos los casos. Esto implica que la solución propuesta por el algoritmo posiblemente no sea la mejor o la que optimice la energía del sistema analizado.

INICIAL		*RESULTADO TABÚ*	
Pulso izquierdo	***Pulso derecho***	***Pulso izquierdo***	***Pulso derecho***
0	0	0	0
0	0	0	0
0	0	0	0
0	5	0	5
0	5	0	5
10	0	10	0
10	10	10	10
5	0	5	0
5	0	5	0
10	10	10	10
0	10	5	0
0	5	0	5
5	0	5	0
10	0	0	0
10	0	0	0
5	5	5	5
5	0	5	0
0	5	0	5
10	0	10	0
10	0	10	0
10	10	10	10
0	10	0	10
0	10	0	10
5	0	5	0
5	0	5	0

Tabla 4.5 Resultado del método Tabú para el controlador de posición, combinaciones aleatorias

4.1.3 Controlador de evasión de obstáculos

Para la evaluación del controlador de evasión de obstáculos se establecieron cuatro casos hipotéticos posibles, y se registraron las trayectorias seguidas por el robot, según se muestra en las Figuras 4.5, 4.6, 4.7 y 4.8, para la presencia de dos paredes en una esquina de la pista, una pared frontal, un obstáculo no registrado y otro robot, respectivamente.

Según las pruebas realizadas, el controlador de evasión de obstáculos funciona según lo deseado, ya que en todos los casos se logra evadir a los obstáculos mientras el robot realiza una tarea de recolección o depósito de objetos.

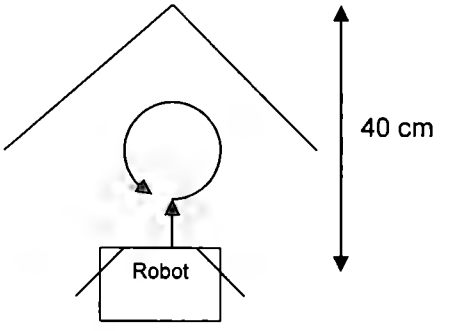


Figura 4.5 Trayectoria seguida por el robot ante la presencia de una esquina de paredes

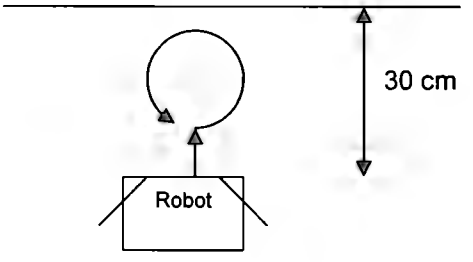


Figura 4.6 Trayectoria seguida por el robot ante la presencia de una pared frontal

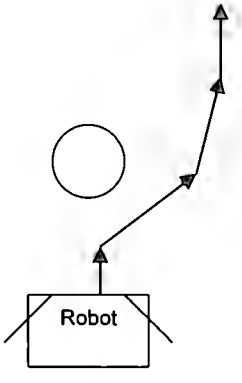


Figura 4.7 Trayectoria seguida por el robot ante la presencia de un obstáculo no registrado

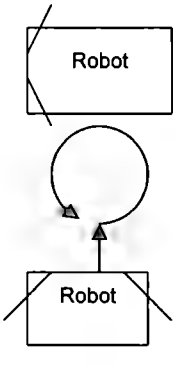


Figura 4.8 Trayectoria seguida por el robot ante la presencia de otro robot

4.2 Algoritmos genéricos del controlador

Durante el proyecto, se requirió probar una serie de controladores neuro-difusos para el diseño y desarrollo de los controladores de posición y evasión de obstáculos. Por esto, se decidió a modelar y programar un controlador genérico capaz de recibir un número de entradas finito, una serie de coeficientes para las funciones de pertenencia de cada una de las entradas (producto de de los pesos relacionados con el entrenamiento de las redes neuronales) y la tabla de asociación difusa; al mismo tiempo, el controlador es capaz de proponer un número de salidas o respuestas de la excitación.

Las ventajas que ofrece este controlador neuro-difuso desarrollado es:

1) La flexibilidad de simulación de resultados para el desarrollo de controladores.
2) Capacidad de controlar distintos procesos no solamente del área en robótica.
3) Facilidad de tratar la información.

Para comprobar el funcionamiento de los algoritmos del controlador se buscó otro sistema que fuera conocido o modelable matemáticamente, por lo que se recurrió a un motor de corriente directa de dimensiones pequeñas, empleado en el primer semestre del proyecto para llevar a cabo diversas pruebas con controladores difusos y neuro-difusos. Los parámetros del motor DC comprenden su función de transferencia, obtenida de la respuesta a impulso:

$$\frac{0.72}{0.12s+1} \tag{36}$$

Considerando como entradas voltajes correspondientes al error y a la derivada del mismo, midiendo el voltaje producido por un generador, se establecieron las funciones de pertenencia de la difusificación como se muestra en la Figura 4.9, conformadas por tres definidas como

$$Z(a,f,d)(x)=\begin{cases}0 & x<a\\ \dfrac{x-a}{f-a} & a\le x\le f\\ \dfrac{x-d}{f-d} & f<x\le d\\ 0 & d<x\end{cases}, \tag{37}$$

$$N(h,e,b)(x)=\begin{cases}1 & 0\le x<e\\ \dfrac{x-b}{e-b} & e\le x\le b\\ 0 & b<x\end{cases} \text{ y} \tag{38}$$

$$P(c,g,i)(x)=\begin{cases}0 & x<c\\ \dfrac{x-c}{g-c} & c\le x\le g\\ 1 & g<x\le i\end{cases} \tag{39}$$

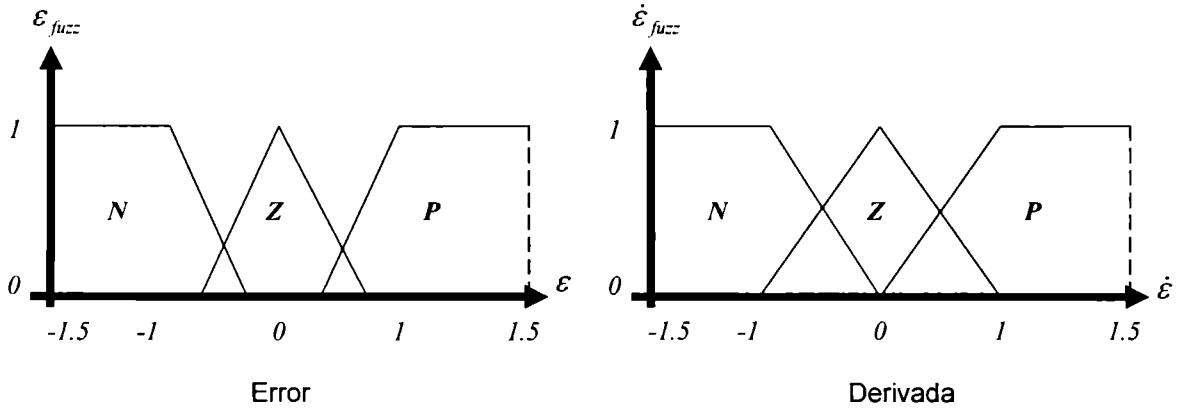


Figura 4.9 Funciones de pertenencia para la difusificación en controlador neuro-difuso del motor de DC

Las funciones de pertenencia mostradas anteriormente tienen los valores siguientes:

- Negativo: (-1,-1,-0.1)
- Cero: (-0.2,0,0.2)
- Positivo: (0.1,1,1)

Al introducir las funciones al entrenamiento de la red neuronal basada en series trigonométricas se obtuvieron los siguientes coeficientes:

```
A = [ 0.4052  0.1021  0.4923  0.4052  0.1021  0.4923;
      0.3484  0.0074 -0.4082  0.3484  0.0074 -0.4082;
      0.2141 -0.0971  0.2215  0.2141 -0.0971  0.2215;
      0.0831 -0.0205 -0.0618  0.0831 -0.0205 -0.0618;
      0.0125  0.0949  0.0008  0.0125  0.0949  0.0008;
      0.0034  0.0333 -0.0095  0.0034  0.0333 -0.0095;
      0.0164 -0.0835  0.0224  0.0164 -0.0835  0.0224;
      0.0200 -0.0429 -0.0135  0.0200 -0.0429 -0.0135;
      0.0102  0.0758  0.0008  0.0102  0.0758  0.0008;
      0.0023  0.0507 -0.0008  0.0023  0.0507 -0.0008;
      0.0029 -0.0604  0.0066  0.0029 -0.0604  0.0066;
      0.0075 -0.0544 -0.0059  0.0075 -0.0544 -0.0059;
      0.0073  0.0504  0.0008  0.0073  0.0504  0.0008;
      0.0037  0.0560  0.0006  0.0037  0.0560  0.0006;
      0.0012 -0.0348  0.0023  0.0012 -0.0348  0.0023;
      0.0028 -0.0537 -0.0031  0.0028 -0.0537 -0.0031];
```

Donde las primeras tres columnas representan las funciones de pertenencia N,Z,P para la primera entrada (error) y las otras tres columnas son igual para la segunda entrada (derivada del error).

Posteriormente, se definió una FAM según se muestra en la Tabla 4.6. Las funciones de pertenencia definidas para la salida se determinaron de la forma siguiente:

- Negativo: una barra vertical en el valor x= -1
- Cero: una barra vertical en el valor x= 0
- Positivo: una barra vertical en el valor x= 1

ε / $\dot{\varepsilon}$	**N**	**Z**	**P**
N	N	N	P
Z	N	Z	P
P	N	P	P

Tabla 4.6 FAM para el controlador del motor de DC

Consecuentemente se simuló la respuesta a escalón unitario en LabVIEW, según se muestra en la Figura 4.10, donde se aprecia que el máximo del sistema es de 0.72, y se presenta un ligero sobreimpulso.

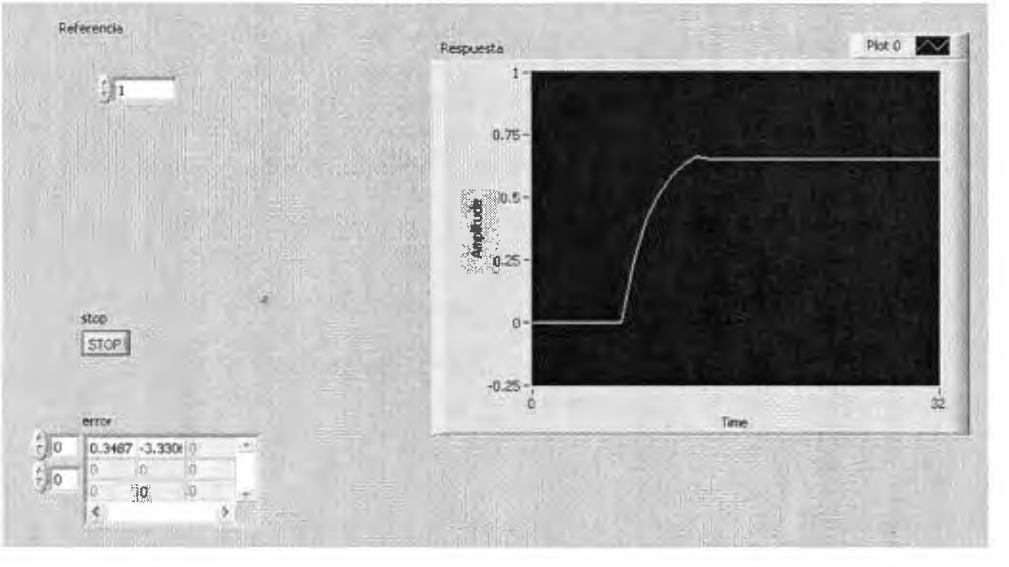


Figura 4.10 Respuesta a escalón del motor de DC con el controlador neuro-difuso genérico

Para comprobar el método Tabú en este caso se utilizó una sencilla tabla de asociación difusa propuesta para el control neuro-difuso del motor DC (Tabla 4.6). A continuación, se aprecia la tabla inicial que se propuso junto con el resultado del método después de 20 minutos de procesamiento en una computadora Pentium 4, 1.80GHz (Tabla 4.7).

INICIAL	***TABÚ***	***DESEADO***
Voltaje	***Voltaje***	***Voltaje***
0	-1	-1
1	-1	-1
-1	-1	-1
0	-1	-1
0	0	0
0	0	1
1	0	1
0	1	1
0	0	1

Tabla 4.7 Método Tabú aplicado a la FAM del motor DC

El método Tabú se corrió en este caso únicamente para obtener una respuesta a escalón unitario; por lo cual, la tabla de asociación difusa obtenida no corresponde fielmente a la tabla deseada u obtenida mediante criterios adquiridos por el análisis de la respuesta. En la Figura 4.11 se muestra la respuesta del sistema del motor de DC con la FAM obtenida por el método Tabú.

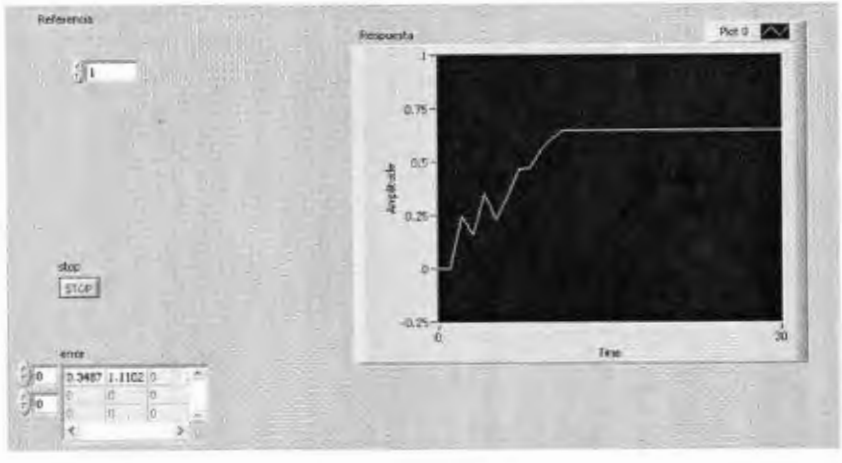


Figura 4.11 Respuesta a escalón del motor de DC con el controlador neuro-difuso genérico con la FAM obtenida de la búsqueda Tabú

Comparando las respuestas obtenidas con y sin método Tabú es posible apreciar que en el uso de la FAM obtenida por Tabú se presenta una oscilación en el arranque del sistema, aunque se alcanza a establecer en un tiempo menor al caso de la respuesta obtenida con la FAM propuesta. Por otra parte, se observa que hay un pequeño sobreimpulso en el primer caso evaluado, mientras que en el segundo caso el sobreimpulso no existe.

4.3 Comunicación inalámbrica

Para reducir los costos en el hardware de la implementación física de la comunicación inalámbrica se contempló el uso de un adaptador USB-Bluetooth genérico, fácil de adquirir en cualquier tienda de artículos electrónicos. Sin embargo, no fue posible lograr que LabVIEW trabajara con este tipo de dispositivos debido a que es necesario usar hardware que sea plenamente compatible con los programas de asociación e instalación de dispositivos de Windows. El adaptador Belkin cuenta con la característica indispensable de ser *plug and play* – es decir, con controladores preprogramados en Windows-, el cual es perfectamente compatible con el software LabVIEW, por lo que fue necesario recurrir al uso del mismo. En la Figura 4.12 se muestra una fotografía del sistema final empleado para la comunicación inalámbrica.

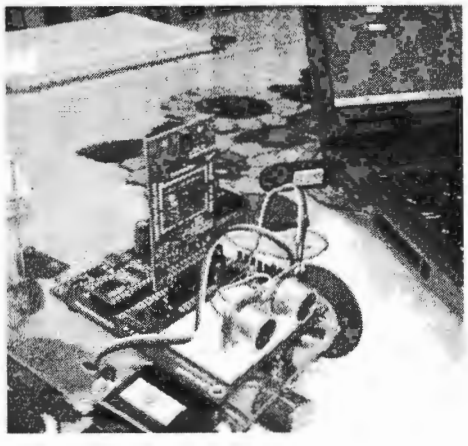


Figura 4.12 Hardware para implementar la intercomunicación inalámbrica, compuesto por un transmisor-receptor Embedded Bluetooth y un adaptador USB-Bluetooth marca Belkin

Realizando un gran número de pruebas se determinó que el protocolo de comunicación entre LabVIEW y el robot tiene un retardo de 2 segundos, para el envío de las tres distancias medidas por el robot y la recepción de la señal de corrección correspondiente. Este retardo tiene una magnitud considerable si se desea un control en tiempo real, sin posibilidades de ser reducida desde el software, por lo que resultaría más conveniente el uso de otro sistema distinto al inalámbrico para la comunicación entre el controlador y la planta, como un microprocesador embebido.

4.4 Implementación física de los prototipos

Para comprobar las ventajas funcionales del robot propuesto con uno ya realizado se analizaron diversos criterios con los resultados mostrados en la Tabla 4.8 a continuación, comparando el Boe-bot de Parallax con el diseño nuevo. Se determinaron criterios cinemáticos como la velocidad o la aceleración, además de otras características que determinan la flexibilidad o el comportamiento de los materiales ante condiciones normales de operación.

En términos generales, el diseño propuesto es una alternativa más barata y funcional que comprar un robot ya hecho, debido a que no hay una gran diferencia en las características dinámicas del comportamiento, y otras características que depende de los materiales como la resistencia al impacto, o el aislamiento eléctrico y magnético, son mejores. Sin embargo, existen algunos aspectos que podrían mejorarse en el diseño, entre estos el material de la llanta trasera, o los grados de libertad de la misma, ya que tiende a generar demasiada fricción respecto al suelo, generando error en la posición real del robot respecto a la deseada.

Con las carcasas de los robots completamente armadas según el nuevo diseño propuesto, y habiendo adquirido y conseguido los aditamentos electrónicos necesarios para la navegación autónoma de los robots, se procedió al ensamble de los sensores ultrasónicos según diversas pruebas que se realizaron para determinar la amplitud de detección real sobre objetos esféricos.

Criterio	*Robot Boe-Bot de Parallax*	*Robot con el nuevo diseño*
Velocidad	10.8 cm/s	10.8 cm/s
Tiempo de aceleración	Imperceptible	Imperceptible
Resistencia al impacto	La carcasa se talla en los impactos	La carcasa se arquea en los impactos
Fricción en la rueda de apoyo	La fricción es imperceptible, no se desvía al dar la vuelta por la rueda trasera	La rueda produce fricción y se produce un error en la posición real respecto a la deseada (desviación de la trayectoria)
Interferencia de señales eléctricas	Es necesario aislar la tarjeta con el Basic Stamp para que no haga contacto con la superficie metálica de la carcasa	El material de la carcasa es aislante. Ni se producen cortos circuitos en la tarjeta ni interferencia por el campo magnético de los imanes
Adaptabilidad a los cambios	Para hacer aditamentos del mismo material de la carcasa se requiere de equipo especial o conseguir piezas especiales de aluminio	El plástico es barato y fácil de manipular para adaptar nuevas piezas a la carcasa. Los pegamentos tienen mayor adherencia

Tabla 4.8 Comparativo funcional del diseño propuesto con un robot Boe-Bot de Parallax

De fábrica, el lóbulo de detección de los sensores para objetos redondos puede apreciarse en la Figura 4.13 con una amplitud de 40° totales. Se decidió colocar los tres sensores de cada robot a 30° de separación, para tener una visión que pueda diferenciar entre objetos, robots y paredes (Figura 4.14).

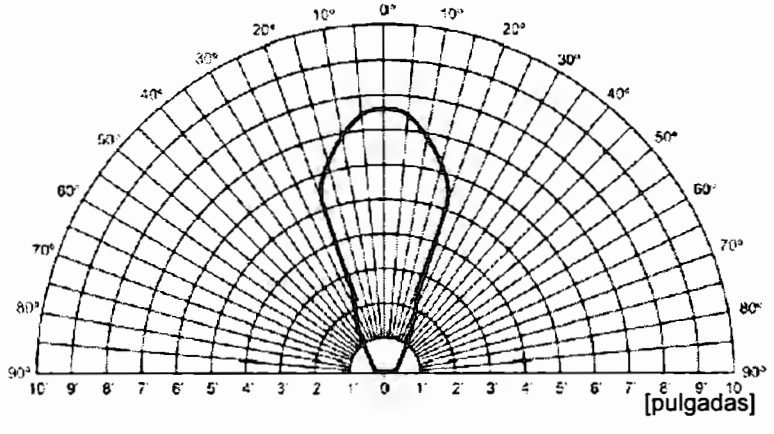


Figura 4.13 Lóbulo de detección de objetos redondos de los sensores ultrasónicos [37]

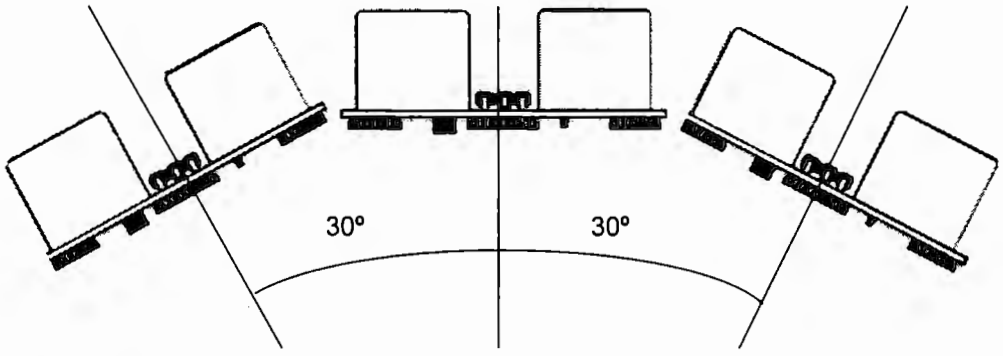


Figura 4.14 Arreglo de sensores para la detección de objetos y obstáculos

4.5 Desempeño en la tarea colaborativa simple

Para el trabajo colaborativo se diseñó una interfaz gráfica en LabVIEW que permite el monitoreo de la posición actual de los robots, considerando los dos controladores neuro-difusos implementados para la posición y la evasión de obstáculos, la cual puede observarse en la Figura 4.15. Se aprecia que el robot 01 tiene su *home* en (0,0) y el robot 02 lo tiene en (2,2). En la parte superior se encuentra el control de las posiciones de las pelotas.

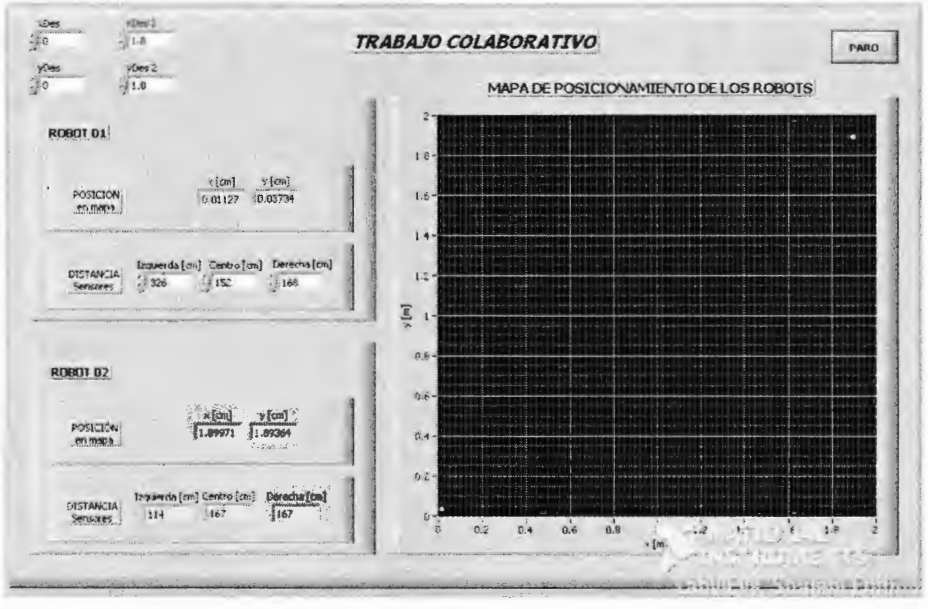


Figura 4.15 Interfaz en LabVIEW para el monitoreo de la tarea colaborativa

En la Figura 4.16 se aprecia la vista parcial de las conexiones realizadas para la tarea colaborativa. Cabe destacar que el programa en LabVIEW y su interfaz ha resultado de suma utilidad para realizar la evaluación de los parámetros delimitados para la tarea colaborativa, como la repartición de los objetos a recolectar, y el tiempo total de recolección.

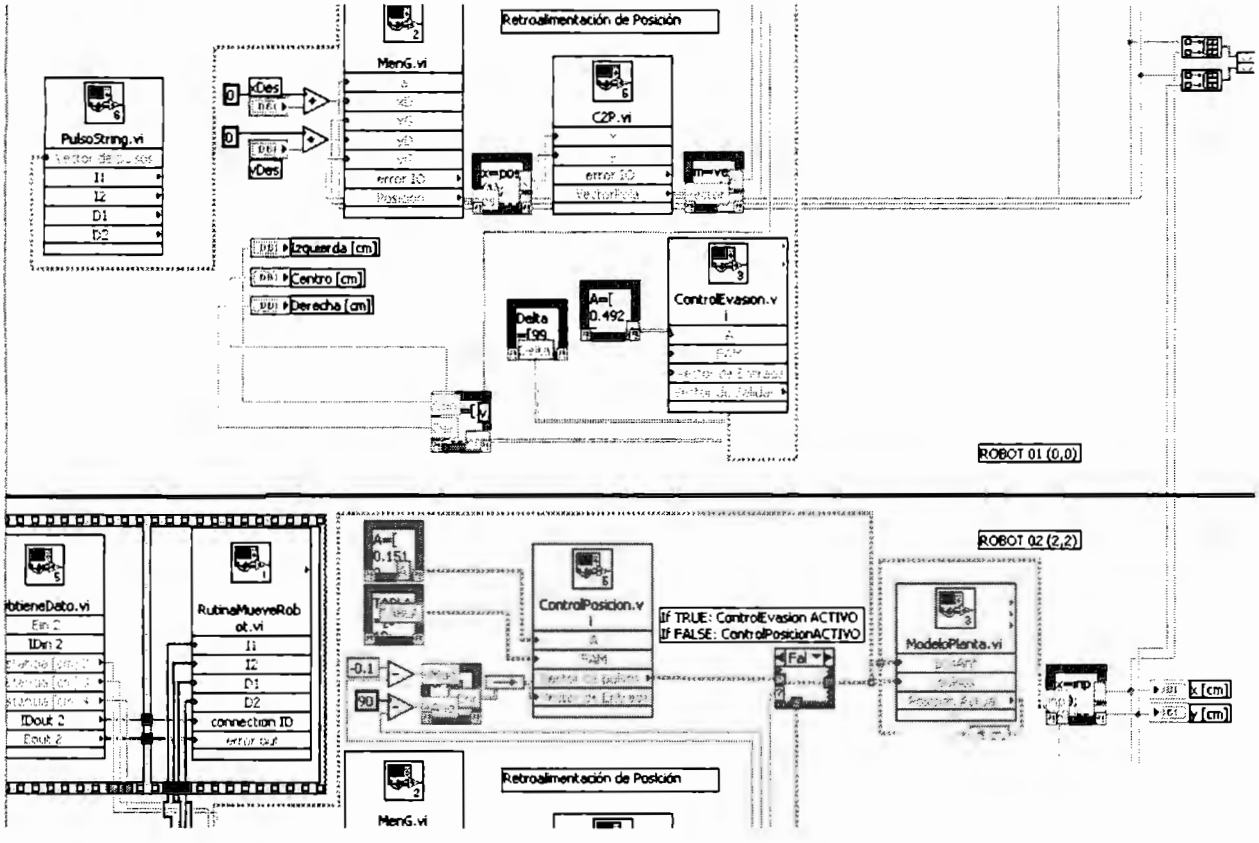


Figura 4.16 Vista parcial de las conexiones del controlador del trabajo colaborativo

Las pruebas finales de la tarea colaborativa se realizaron en la pista con los objetos y las posiciones delimitadas en el capítulo 7, como se muestra en la Figura 4.17. Se realizaron tomas de video, anexas en el CD, como registro del movimiento verídico de los robots según el algoritmo de la tarea colaborativa.

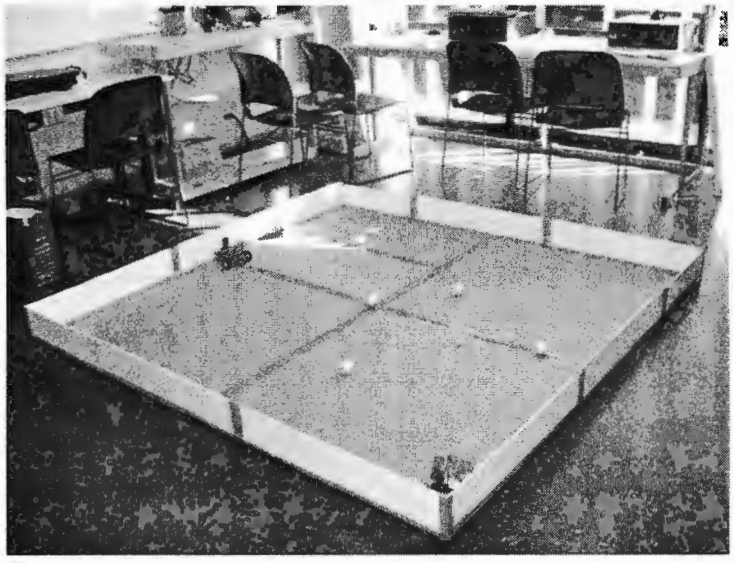


Figura 4.17 Fotografía del inicio de las pruebas finales para la tarea colaborativa

Las trayectorias seguidas por cada robot durante la recolección de objetos fueron las mostradas en la Figura 4.18, con un tiempo total de 17 minutos con 20 segundos. Se aprecia que el robot ubicado en la esquina inferior derecha recogió tres objetos, mientras que el otro sólo dos. Los números indican el orden de recolección de cada objeto.

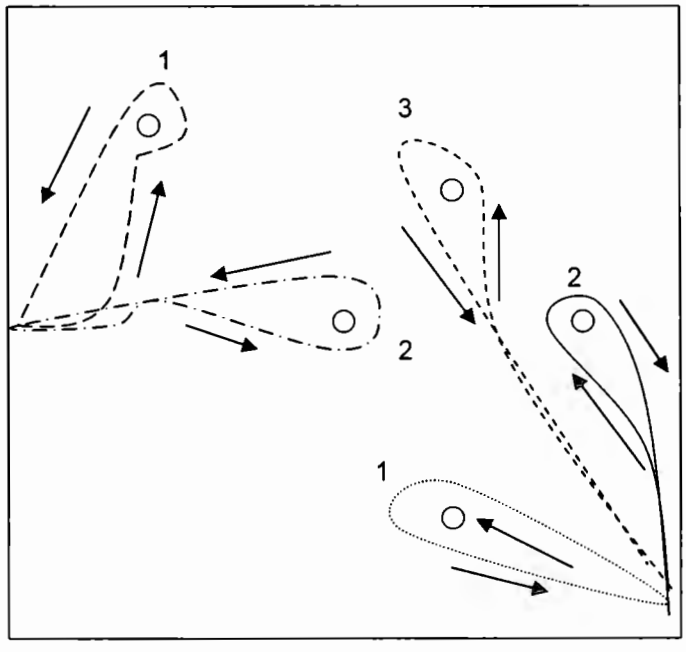


Figura 4.18 Trayectorias seguidas por los robots durante las pruebas de la tarea colaborativa

En términos generales, el tiempo total pareciera de gran magnitud, y sin embargo es necesario considerar el retardo causado por la comunicación cada vez que el robot envía la lectura de los sensores y recibe la señal de corrección del controlador vía Bluetooth.

CAPÍTULO 5

CONCLUSIONES Y TRABAJO A FUTURO

5.1 Conclusiones

De acuerdo con los objetivos planteados es posible establecer las siguientes conclusiones:

- Los controladores neuro-difusos son una alternativa viable para la navegación autónoma de micro-robots móviles, ya que el procesamiento se realiza en un tiempo corto y pueden ser adaptados a una gran variedad de sistemas no lineales debido a su flexibilidad.
- Los programas específicos realizados de forma genérica para los controladores neuro-difusos desarrollados en el proyecto se ajustan a diversos sistemas, como fue posible observar en las pruebas al motor de corriente directa, por lo que podrán ser empleados en otros proyectos que requieran el uso de este tipo de control.
- El control de posición resultó de una gran precisión, con un error entre 5 y 6%, y en conjunto con el modelo matemático de la planta permitió el movimiento deseado de los robots en el desempeño de la tarea simple.
- El control de evasión de obstáculos logra evitar las colisiones con la pared y entre los mismos robots, diferenciando claramente entre los objetos a recolectar y otros cuerpos presentes en la pista.

- El método de búsqueda Tabú propuesto permite encontrar alternativas a la FAM propuesta en un controlador neuro-difuso, mediante la evaluación de criterios y la selección de la alternativa que cumple con el resultado más óptimo.
- Se logró una comunicación efectiva entre LabVIEW y el Basic Stamp 2, para compartir la información referente a la medición hecha por los sensores ultrasónicos y la respuesta del controlador, y sin embargo el retardo presentado es bastante considerable, por lo que sería recomendable buscar una alternativa adicional como los sistemas embebidos, o los procesadores digitales de señales para la implementación física del controlador total.
- Los sistemas de control inteligente pueden ser empleados como alternativas a los sistemas de control tradicional, y el ejemplo realizado de la tarea colaborativa simple puede extrapolarse en complejidad y cantidad hacia otros sectores de la robótica.
- El nuevo diseño del robot cumple con las características deseadas: es más barato que los robots comerciales, es flexible para la realización de modificaciones en el mismo, es más ligero que otros modelos y sirve completamente en la tarea colaborativa propuesta sin mayor problema.

5.1.1 Evaluación de MATLAB y LabVIEW

Los programas MATLAB y LabVIEW fueron empleados en la fase de diseño, y el segundo específicamente en la fase de implementación física. Las diferencias funcionales entre los mismos los hacen ideales para distintas fases a lo largo del desarrollo de un proyecto como el mostrado.

MATLAB es ideal para programar algoritmos genéricos de forma lineal, si se cuenta con conocimientos suficientes de lenguajes de alto nivel en programación. De esta forma, los algoritmos quedan plasmados de forma resumida, y es posible agregar comentarios y crear funciones al gusto del usuario. También permite visualizar gráficas de forma rápida y es posible manipular datos en operaciones matemáticas con rapidez. Por otra parte, presenta la

desventaja de realizar la lectura del código y su procesamiento de forma lineal, ya que no permite el procesamiento paralelo de los algoritmos, y no es efectivo para la implementación física directa en sistemas porque no cuenta con funciones o módulos que faciliten la comunicación con otras aplicaciones o hardware.

Por otra parte, LabVIEW tiene dos opciones para realizar la programación de algoritmos: de forma gráfica a través de iconos o de forma textual pegando el código m en bloques. De esta forma, fue posible utilizar directamente los algoritmos genéricos desarrollados en MATLAB. Asimismo, cuenta con bloques predeterminados para la comunicación con dispositivos de hardware a través de diversos protocolos, desde comunicación inalámbrica vía Bluetooth hasta páginas web, pasando por dispositivos de adquisición de señales e instrumentos, lo que resultó fundamental para lograr aplicar el controlador al sistema de manera remota. Adicionalmente, permite la presencia de hilos múltiples en los programas, por lo que puede estar procesando varios bloques simultáneamente (aunque no sea técnicamente de esta forma). No obstante, para la simulación de los resultados y el manejo de distintos tipos de datos hay que realizar conversiones y manipulaciones que se traducen en pérdida de información, y en una constante revisión de las conexiones de los bloques en los programas.

En la parte de diseño, MATLAB es una herramienta irremplazable, mientras que en la implementación física LabVIEW fue una solución de bajo costo y resultados bastante aceptables.

5.2 Trabajo a futuro

A lo largo del proyecto fue posible encontrar diversas situaciones que podrían ser mejoradas en el futuro, de contar con distinto hardware y software, que además de procesar los algoritmos utilizados en el controlador a gran velocidad, permitieran reducir el tiempo total de acción y reacción del sistema en su totalidad. A continuación se describen las partes fundamentales a mejorar o trabajar en un futuro.

5.2.1 Procesamiento del controlador

- Implementar el controlador en un DSP o sistema embebido, para eliminar la comunicación inalámbrica entre el controlador en LabVIEW y el Basic Stamp 2.
- Cambiar el Basic Stamp 2 por un procesador más potente.

5.2.2 Sensores y percepción

- Uso de sistemas de visión para identificación de objetos y obstáculos con mayor precisión.
- Mapeo de los objetos previo a la tarea colaborativa.

5.2.3 Comunicación inalámbrica

- Eliminar la comunicación inalámbrica entre los robots y LabVIEW.
- Implementar comunicación inalámbrica directa entre los controladores de cada robot para la tarea colaborativa.

5.2.4 Nuevo diseño del robot

- Uso de una cubierta para proteger el robot de condiciones ambientales
- Base especial para los sensores o sistema de percepción a usar.
- Soporte para el imán de recolección de objetos.

5.2.5 Tarea colaborativa

- Alternativas de algoritmos para hacer comparaciones
- Cambio en el mecanismo para el depósito de los objetos una vez recolectados

REFERENCIAS BIBLIOGRÁFICAS

[1] Selig, J. M. (1992). *Introductory robotics.* (Pp. 1 - 6). Reino Unido, Prentice Hall Internacional.

[2] Loera, Marta Eva (2002). *Laboratorio de Inteligencia Artificial.* Revisado el 5 de octubre de 2007, en línea: http://comsoc.udg.mx/gaceta/paginas/260/260-8.pdf

[3] Tecnológico de Monterrey (2007). *Visión y Misión 2015.* Revisado el 22 de noviembre de 2007, en línea: http://www.itesm.edu/wps/portal?WCM_GLOBAL_CONTEXT=/wps/wcm/connect/ITESMv2/Tecnol%C3%B3gico+de+Monterrey/Con%C3%B3cenos/Principios%2C+visi%C3%B3n+y+misi%C3%B3n/Visi%C3%B3n+y+Misi%C3%B3n+2015/

[4] Rivero, Fernando Alberto (2005). *Control visual de trayectorias para un vehículo autónomo utilizando una cámara móvil.* Tesis elaborada en el Tecnológico de Monterrey, Campus Ciudad de México.

[5] Huesca, Gilberto (2006). *Laboratorio virtual de robótica móvil en esquemas de coordinación concurrente.* Tesis elaborada en el Tecnológico de Monterrey, Campus Ciudad de México.

[6] National Research Council (1983). *Applications of robotics and artificial intelligence to reduce risk and improve effectiveness.* Estados Unidos de América, National Academy Press.

[7] *Anteproyecto de Norma Oficial Mexicana para Seguridad en Instalaciones Robotizadas.* Revisado el 10 de marzo de 2007, en línea: http://biblioteca.itesm.mx/cgi-bin/doctec/opendoc?cual=2616&archivo=55386&pagina=1&paginas=1,2&query=(robotica,AND,historia),AND,tipo%3Da

[8] Álvarez Ruiz, José Ignacio & Rodríguez Lopera, Eduardo Javier (s.f.). *Microrobótica.* Revisado el 19 de junio de 2007, en línea: http://www.uco.es/%7Ei02alruj/Microrobotica.htm

[9] Little, James (s.f.). *Robot Partners: Collaborative Perceptual Robotic Systems.* Revisado el 5 de octubre de 2007, en línea: http://www.cs.ubc.ca/spider/little/links/papers/cdv.pdf

[10] Guo, Yi, et al. (2003). *Towards Collaborative Robots for Infrastructure Security Applications.* Revisado el 5 de octubre de 2007, en línea: http://citeseer.ist.psu.edu/690572.html

[11] Arabe, Katrina (2000). *Works Well with Others: Collaborative Robots Interacting on the Assembly Line.* Revisado el 5 de octubre de 2007, en línea: http://news.thomasnet.com/IMT/archives/2000/11/works_well_with.html?t=archive

[12] Ranky, Paul (2003). *Collaborative, synchronous robots serving machines and cells.* Revisado el 5 de octubre de 2007, en línea: http://www.emeraldinsight.com/Insight/viewContentItem.do?contentType=Article&hdAction=lnkhtml&contentId=1454242

[13] Zadeh, Lofti A. (1965). "Fuzzy sets". En: *Fuzzy Models for Pattern Recognition.* Estados Unidos de América, Institute of Electrical and Electronics Engineers.

[14] Zadeh, Lofti A. (1973). "Outline of a New Approach to the Analysis of Complex Systems and Decision Process". En: *Fuzzy Models for Pattern Recognition.* Estados Unidos de América, Institute of Electrical and Electronics Engineers.

[15] Brío, B. & Sanz, A. (2002). *Redes Neuronales y Sistemas Difusos.* 2ª edición. México, Alfaomega Grupo Editor.

[16] Hilera, J. & Martínez, V. (1995). *Redes neuronales artificiales. Fundamentos, modelos y aplicaciones.* 1ª edición. Estados Unidos de América, Addison-Wesley Iberoamericana.

[17] Departamento de Electrónica Universidad Autónoma Metropolitana Azcapotzalco (2007). *Estándares Inalámbricos.* Revisado el 5 de octubre de 2007, en línea: http://zeus.uam.mx/index.php?option=com_content&task=view&id=87&Itemid=172

[18] Ponce Cruz, Pedro et al. (2006). *A Novel Neuro-Fuzzy Controller Based on Both Trigonometric Series and Fuzzy Clusters.* Reporte interno del Tecnológico de Monterrey, México.

[19] Nguyen, H. et al (2002). *A First Course in Fuzzy and Neural Control.* 1a edición. Estados Unidos de América, Chapman & Hall.

[20] Bezdek, James C. & Pal, Sankar K. (1992). *Fuzzy Models for Pattern Recognition.* Estados Unidos de América, Institute of Electrical and Electronics Engineers.

[21] Ponce Cruz, Pedro & Saint Martin Suárez, Roberto (2005). *Neural Networks Based on Fourier Series.* Reporte interno. Tecnológico de Monterrey, México.

[22] Méndez, David & Ramírez, Fernando David (2007). *Sistema de Navegación Neuro-Difuso para Robots Móviles.* México, Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Ciudad de México.

[23] Merelo, J.J. (s.f.). *Técnicas heurísticas de resolución de problemas: computación evolutiva y redes neuronales.* Revisado el 31 de agosto de 2007, en línea: http://geneura.ugr.es/~jmerelo/tutoriales/heuristics101/

[24] Lo K.L. & Nashid, I. (1993). *Expert systems and their application to power systems. Part 2 Search methods and languages.* Revisado el 31 de agosto de 2007, en línea: http://0-ieeexplore.ieee.org.millenium.itesm.mx/iel1/2224/5856/00224085.pdf?tp=&arnumber=224085&isnumber=5856

[25] Hui Tan, Pen & Rasmussen, Lars K. (2004). *Multiuser Detection in CDMA – A Comparison of Relaxations, Exact, and Heuristic Search Methods.* Revisado el 31 de agosto de 2007, en línea: http://0-ieeexplore.ieee.org.millenium.itesm.mx/iel5/7693/29589/01343915.pdf?tp=&arnumber=1343915&isnumber=29589

[26] Glover, Fred & Laguna, Manuel (1997). *Tabu Search.* Estados Unidos de América, Kluwer Academia Publishers.

[27] Sandia National Laboratories (1997). *Tabu Search.* Revisado el 31 de agosto de 2007, en línea: http://www.cs.sandia.gov/opt/survey/ts.html

[28] Glover, Fred & Laguna, Manuel (s.f.). *Tabu Search.* Revisado el 31 de agosto de 2007, en línea: http://www.dei.unipd.it/~fisch/ricop/tabu_search_glover_laguna.pdf

[29] Rajan, C.C.A et al (2003). *Neural-based tabu search method for solving unit commitment problem.* Revisado el 31 de agosto de 2007, en línea: http://0-ieeexplore.ieee.org.millenium.itesm.mx/iel5/2195/27308/01214565.pdf?tp=&arnumber=1214565&isnumber=27308

[30] Denna, Maurizio, Mauri, Giancarlo & Zanaboni, Anna Maria (1999). *Learning Fuzzy Rules with Tabu Search – An Application to Control.* IEEE Transactions on Fuzzy Systems, Vol. 7, No. 2. Pp. 295-318.

[31] Lindsay, A. (2004). *Robotics with the Boe-Bot.* Estados Unidos de América: Parallax.

[32] Efimov, N. (1970) *Formas cuadráticas y matrices.* Moscú: Editorial Mir.

[33] Departamento de Arquitectura y Tecnología de Computadoras. (s.f.) *Matrices homogéneas.* Presentación en Power Point realizado por la docencia. Universidad de Sevilla, España. Revisado el 14 de septiembre de 2007.

[34] Parallax (2007). *Parallax products.* Revisado el 11 de octubre de 2007, en línea: http://www.parallax.com

[35] Blauden Electronics (2007). *¿Qué es Bluetooth?.* Revisado el 28 de septiembre de 2007, en línea: http://www.bluezona.com/index.php?option=com_content&task=view&id=25&Itemid=50/

[36] Instituto Tecnológico Autónomo de México [ITAM] (2007). *CANNES: Agentes Robóticos Autónomos.* Revisado el 12 de septiembre de 2007, en línea: http://cannes.itam.mx/Espaniol/investigacion/areas/robotica.html

[37] Parallax (2006). *PING))) Ultrasonic Distance Sensor (#28015).* Revisado el 7 de octubre de 2007, en línea: http://www.parallax.com/dl/docs/prod/acc/28015-PING-v1.3.pdf

ANEXOS

Anexo 1 Tabla de comparación entre la posición real nueva y la posición estimada

PUNTO	PULSOS		POSICIÓN ACTUAL			POSICIÓN NUEVA			POSICIÓN ESTIMADA			ERROR ABSOLUTO			ERROR RELATIVO		
	Pizq	Pder	xg (m)	yg (m)	ar (rad)	xg (m)	yg (m)	ar (rad)	xg (m)	yg (m)	ar (rad)	EAxg	EAyg	EAar	ERxg	ERyg	ERar
1	3	3	0.000	0.000	0.000	0.000	0.027	0.000	0.000	0.026	0.000	0.000	0.001	0.000	0.000	0.037	0.000
2	3	3	0.000	0.027	0.000	0.000	0.054	0.000	0.000	0.053	0.000	0.000	0.001	0.000	0.000	0.019	0.000
3	3	3	0.000	0.054	0.000	0.000	0.080	0.000	0.000	0.080	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	3	3	0.019	0.094	-0.785	0.038	0.113	-0.785	0.037	0.112	-0.785	0.001	0.001	0.000	0.026	0.009	0.000
5	3	3	0.038	0.113	-0.785	0.057	0.132	-0.785	0.056	0.131	-0.785	0.001	0.001	0.000	0.018	0.008	0.000
6	15	15	0.000	0.000	0.000	0.000	0.112	0.000	0.000	0.109	0.000	0.000	0.003	0.000	0.000	0.027	0.000
7	15	15	0.000	0.112	0.000	0.000	0.224	0.000	0.000	0.220	0.000	0.000	0.004	0.000	0.000	0.018	0.000
8	15	15	0.053	0.053	0.785	-0.026	0.132	0.785	-0.024	0.130	0.785	0.002	0.002	0.000	0.077	0.015	0.000
9	25	25	0.000	0.000	-1.571	0.183	0.000	-1.571	0.177	0.000	-1.571	0.006	0.000	0.000	0.033	0.000	0.000
10	25	25	0.000	0.000	0.000	0.000	0.183	0.000	0.000	0.177	0.000	0.000	0.006	0.000	0.000	0.033	0.000
11	0	4	0.050	0.050	-0.785	0.072	0.074	-0.454	0.060	0.063	-0.471	0.012	0.011	0.017	0.167	0.149	0.038
12	0	4	0.000	0.080	-1.571	0.017	0.083	-1.239	0.016	0.083	-1.257	0.001	0.000	0.017	0.059	0.000	0.014
13	0	15	0.000	0.080	-1.571	0.048	0.103	-0.628	0.045	0.105	-0.524	0.003	0.002	0.105	0.063	0.019	0.167
14	0	27	0.000	0.080	-1.571	0.053	0.146	0.227	0.051	0.145	0.244	0.002	0.001	0.017	0.038	0.007	0.077
15	0	9	0.000	0.080	-1.571	0.035	0.090	-0.960	0.031	0.090	-0.942	0.004	0.000	0.017	0.114	0.000	0.018
16	2	0	0.150	0.080	1.571	0.139	0.079	1.379	0.140	0.081	1.379	0.001	0.002	0.000	0.007	0.025	0.000
17	7	0	0.150	0.080	1.571	0.121	0.087	1.065	0.124	0.087	1.065	0.003	0.000	0.000	0.025	0.000	0.000
18	13	0	0.150	0.080	1.571	0.106	0.099	0.716	0.109	0.099	0.663	0.003	0.000	0.052	0.028	0.000	0.073
19	22	0	0.150	0.080	1.571	0.094	0.127	0.105	0.097	0.128	0.070	0.003	0.001	0.035	0.032	0.008	0.333
20	26	0	0.150	0.080	1.571	0.095	0.142	-0.140	0.098	0.142	-0.175	0.003	0.000	0.035	0.032	0.000	0.250

max	0.167	0.149	0.333
min	0.000	0.000	0.000
prom	0.036	0.019	0.049

prom	3.6%	1.9%	4.9%

***Anexo 2** Programa FCM con redes neuronales basadas en series trigonométricas hecho en MATLAB*

```
function[U,v]=fcmH(c,m,x)       %U,v

                                 %c<n - renglon
                                %col es esP feature
                                % i iteracion c cluster
                                % k iteracion n renglon
C=c;
[N,P]=size(x);

%Inicializa vector V
for(iniR=1:C)
    for(iniC=1:P)
        v(iniR,iniC)=rand(1,1);
    end
end

Jact=0;
ee=999;
%Inicia iteraciones p=1
p=1;
while(p<=100)
    Jant=Jact;
    %Matriz funciones difusas de membresía U
    for(reng=1:C)
        for(colg=1:N)
            sumdeu=0;
            for(sumc=1:C)
                corch= dista(x(colg,:),v(reng,:))/dista(x(colg,:),v(sumc,:));
                sumdeu=sumdeu+(corch^(2/(m-1)));
            end
            U(reng,colg)=1/sumdeu;
        end
    end
    %Matriz de vectores V de los centroides de C
    for(renv=1:C)
        abajo=0;
        for(itK=1:N)
            if(itK==1)
                arriba(itK,:)=(U(renv,itK)^m)*x(itK,:);
                abajo=(U(renv,itK)^m)+abajo;
            else
                arriba(itK,:)=(U(renv,itK)^m)*x(itK,:)+arriba(itK-1,:);
                abajo=(U(renv,itK)^m)+abajo;
            end
        end
        recip=1/abajo;
        v(renv,:)=recip*arriba(N,:);
    end
    Jact=objectiveF(U,v,x,c,m);
    ee=abs(Jant-Jact);
    if(ee<=(10^(-4)))
        p=101;
    else
```

```
        p=p+1;
    end
end


% Funcion Distancia
function[d]=dista(a,b)

[NN,DD]=size(a);
suma=0;
for(mm=1:DD)
    suma=suma+((a(mm)-b(mm))^2);
end
d=suma^(0.5);

%Funcion Jm
function[Jm]=objectiveF(Um,Vm,Xm,Cm,mm)
[NF,DF]=size(Xm);
sumaf=0;
for(kt=1:NF)
    sumad=0;
    for(it=1:Cm)
        dd=dista(Xm(kt,:),Vm(it,:));
        d2=dd^2;
        uu=Um(it,kt)^mm;
        sumad=sumad+uu*d2;
    end
    sumaf=sumaf+sumad;
end
Jm=sumaf;
```

tamEnt	número de entradas
dimCof	dimensión de la matriz de coeficientes
numCD	calcular número de conjuntos difusos
numReg	calcular número de reglas difusas

```
% Calcula los valores de pertenencia para cada entrada
for i=1 to tamEnt
        Ifuzzy(i,:)=funcionMembresia
end

%Minimización de los valores de pertenencia
For i=1 to numReg
        minRegla(i,:)      mínimo de los valores de pertenencia para cada regla
end

%Desdifusificación
sumaA=0
sumaB=0
for i=1 to numReg
        sumaA=sumaA+(minRegla(i,:))*(RedNeuronal(FAM,i));
        sumaB=sumaB+minRegla(i,:);
end
%Valor de la salida
U=sumaA/sumaB;
```

funcionMembresia
Se requiere utilizar las Series de Fourier para obtener estos valores.

RedNeuronal
Se requiere utilizar la tabla de asociación difusa para obtener estos valores.

Anexo 4 *Poster de divulgación científica*

TECNOLÓGICO DE MONTERREY.

Campus Ciudad de México
División de Ingeniería y Arquitectura
Departamento de Mecatrónica

Noviembre, 2007

Controlador Neuro-Difuso para Aplicaciones Colaborativas Simples en Robótica

Dejanira Araiza Illan 955232, Hiram Eredín Ponce Espinosa 968730,
Ximena Rodríguez de la Vega Olivares 993653, Fernando Alvarez Rivas 1105475

Asesor: Dr. Pedro Ponce Cruz
Coordinador: Dr. Raúl Crespo Saucedo

PROBLEMÁTICA

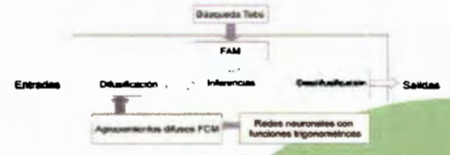
- Necesidad de impulsar el control inteligente en el país
- Contar con controladores genéricos para proyectos
- Lograr cooperación entre robots para tareas simples
- Necesidad de robots flexibles y con menores costos

OBJETIVOS

- Desarrollar e implementar un controlador neuro-difuso para la navegación autónoma
- Implementar el intercambio de información entre robots
- Diseñar e implementar una tarea simple que involucre el movimiento colaborativo de robots
- Evaluar el uso de MATLAB y LabVIEW
- Generar un nuevo diseño físico para los robots

DESARROLLO

Controlador neuro-difuso propuesto y programado en MATLAB y LabVIEW



1. Diseño del controlador neuro-difuso de posición

Modelo matemático de la planta

2. Diseño del controlador neuro-difuso de evasión de obstáculos

3. Diseño de algoritmo para la búsqueda Tabú

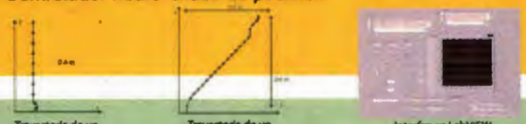
4. Nuevo diseño del robot

5. Diseño de la comunicación inalámbrica vía Bluetooth

6. Diseño del algoritmo y características de la tarea colaborativa simple

RESULTADOS

1. Controlador neuro-difuso de posición



Trayectoria de un movimiento recto
Error: 5.46%

Trayectoria de un movimiento diagonal
Error: 6.2%

Interfaz en LabVIEW

2. Controlador neuro-difuso de evasión de obstáculos

Comportamiento del robot ante trayectorias hipotéticas

3. Nuevo diseño del robot

Ahorro en costo: 19%
Ahorro en peso: 10%

4. Comunicación inalámbrica

Retardo: 2 segundos

5. Tarea colaborativa

Trayectorias seguidas por los robots durante las pruebas, con orden de recolección

Tiempo total: 17min 20 seg

Interfaz en LabVIEW

6. Controlador genérico probado en otro sistema

Motor de DC: $\frac{0.72}{0.12s+1}$

FAM propuesta y resultado de la búsqueda Tabú

Con FAM propuesta | Con FAM por Tabú

CONCLUSIONES

- Los sistemas de control inteligente pueden usarse como alternativas al control tradicional
- El controlador neuro-difuso funciona adecuadamente en los robots empleados
- Los algoritmos genéricos desarrollados para el controlador pueden emplearse en otros sistemas
- La comunicación inalámbrica fue efectiva pero con un retraso significativo
- El nuevo diseño físico es flexible y barato, y se adaptó a los requerimientos del proyecto

TRABAJO A FUTURO

- Implementar el controlador en un DSP o sistema embebido
- Usar sistemas de visión para mapeo de área y objetos
- Eliminar comunicación PC-robot
- Implementar comunicación directa robot-robot
- Diseñar una cubierta para la carcasa del robot
- Cambiar el mecanismo de recolección y depósito de objetos