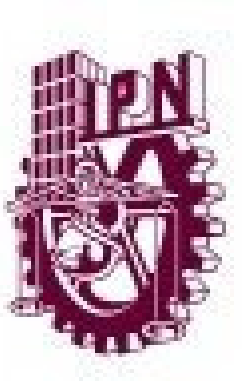


INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD PROFESIONAL AZCAPOTZALCO.

"INTELIGENCIA ARTIFICIAL EN ROBOTS MOVILES DE COMPETENCIA PARA SU APLICACIÓN EN EL DISEÑO DE UN AGENTE INTELIGENTE EN UN ROBOT SEGUIDOR DE TRAYECTORIAS"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN ROBÓTICA INDUSTRIAL

PRESENTA

MARIO ALBERTO GONZÁLEZ TORRES

ASESOR DE TESIS:

M. en C. RAMÓN VALDÉS MARTÍNEZ

MÉXICO D.F. **ENERO, 2009.**

La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica.

Aristóteles.

Agradecimientos

Al Instituto Politécnico Nacional y la ESIME Azcapotzalco, por ser mi segundo hogar y enseñarme los conocimientos necesarios para enfrentar los retos laborales de la vida.

A mi asesor M en C. Ramón Valdez Martínez, por compartir sus conocimientos no solo en este trabajo de tesis si no durante la carrera.

Para mi hija Azul, que tu sonrisa es el motor que mueve mi vida y me hace superar cada día.

Para Ibeth, el amor es lo que nos impulsa a seguir superándonos, gracias por estar a mi lado y crecer junto a mi.

A mi madre, por creer en mí, y no dejarme caer nunca.

A mi padre, gracias por tu comprensión y apoyo

A mis hermanos, Cesar, Maribel y Marisol, por enfrentar el reto de sacarme adelante y apoyarme en los momentos difíciles.

A mis Sobrinos, Jonathan, Erick y Dante, nunca olviden que con dedicación y persistencia se puede lograr cualquier objetivo pese a todos los obstáculos.

Para Alberto y Jaime, por brindarme la mano cuando, más lo necesitaba.

A mis amigos, porque disfrutamos cada momento de la vida y cresemos juntos, afrontando cada reto que la vida nos pone.

Al M. en C. Ezequiel Rojas, por el apoyo y consejos brindados estos últimos años.

…….GRACIAS TOTALES!!

Índice.

1.- INTRODUCCION.

En nuestro país, se realizan distintas competencias de robótica, en la última década el crecimiento de la tecnología ha ido desarrollando componentes con dimensiones reducida para la aplicación en la robótica.

En esta década las competencias se han dividido en los siguientes tipos por mencionar algunos:

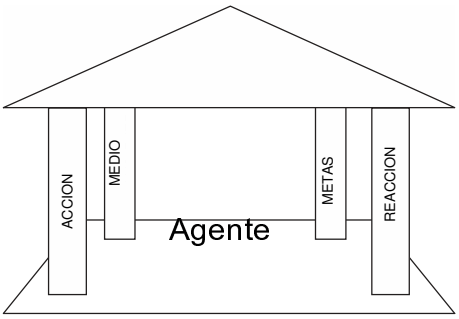
- Robots autónomos móviles.
- Micro robots autónomos móviles.
- Micro robots dirigidos móviles.
- Etc.

La robótica móvil en México surge en los 90, con robots analógicos. Una de las primeras competencias se dio en el año 1994, donde la competencia se basaba en dar un recorrido por una pista recta.

En este trabajo de tesis se describirán los agentes que integran la inteligencia artificial de un robot autónomo móvil para poder construir un modelo basado en estos agentes.

La robótica móvil siempre se plantea en torno a unas metas, las cuales se consiguen en un ambiente determinado con el cual el robot interactúa por medio de percepciones y acciones, a medida de lo deseado el robot móvil tendrá que "aprender" sobre las metas marcadas las estrategias necesarias o almacenadas en su memoria digital.

Podemos representar la autonomía como los cuatro pilares de un edificio. Estos deben tener sus dimensiones en la proporción justa para que el tejado se sustente correctamente. Cuanta más altura tengan, el grado de consecución de objetivos es mayor y proporcional a su autonomía



Representación de los pilares elementales de un agente

La aplicación de tales agentes inteligentes en nuestro país, esta empezado a desarrollarse, ya que países como España, Japón y Alemania por mencionar algunas, se ha implementado desde el año 2000. Por ello esta investigación trata de concientizar a los alumnos, profesores y toda la gente interesada en la robótica móvil, a diseñar los elementos del robot según el agente inteligente diseñado en base a las matrices PAMA.

Una matriz PAMA como lo fundamentaremos mas adelante, definirá la forma y construcción del agente inteligente de un robot con base a su percepción, ambiente, metas y acciones a tomar en cada punto de una competencia.

Este modelo PAMA es tomado de la inteligencia del razonamiento del ser humano y como sabemos, la robótica trata de imitar los movimientos y pensares del ser humano.

Tomando en cuenta que esta tesis va dirigida a la mejora de técnicas de construcción de dichos robots, se utilizaran materiales de bajo costo pero que no estarán limitados, se podrán sustituir siempre y cuando cumplan con los requerimientos del agente inteligente que lo integra.

II.- OBJETIVOS.

El objetivo principal de este trabajo de tesis es analizar los distintos tipos de agentes inteligentes que existen, para desarrollar la inteligencia artificial en robots móviles autónomos, en base a las matrices PAMA que lo integran.

Cabe mencionar que este análisis se llevará acabo mediante el desglose de funcionamiento, de los distintos tipos de robots de competencia, para así poder centrar la matriz PAMA en el diseño de un robot seguidor de trayectorias.

Un objetivo más especifico es el de desarrollar esta inteligencia artificial, para la construcción de un robot seguidor de línea con base a las matrices PAMA para así optimizar su acción en el medio en el que se desarrolla.

III.- JUSTIFICACION.

Debido a la evolución de la robótica móvil es necesario conocer y llevar acabo nuevas técnicas de Inteligencia Artificial y del diseño, construcción y programación, para acudir a competencias que, a nivel nacional e internacional, se llevan a cabo en la actualidad, donde se ha pretendido la participación en diferentes categorías. Estos concursos plantean diferentes desafíos en los que la autonomía de los robots es el elemento básico y se proclama vencedor aquel que proporciona un mayor nivel de atributos.

Fundamentado en esta idea, se plantean los principales enfoques desde los que se procede a la construcción de un robot: un diseño electromecánico que permita una mayor calidad y robustez de actuación o la incorporación de soluciones del campo de la Inteligencia Artificial que permitan optimizar las actuaciones de dicha construcción electromecánica. En este trabajo, sin dejar de lado el primer aspecto, se opta por diferentes razones por la realización de un mayor esfuerzo en el segundo aspecto intentando mostrar como la

utilización de soluciones de la IA permiten alcanzar unos buenos resultados que pueden llegar incluso a la superación de las limitaciones que pudiera producir diseños electromecánicos no optimizados. En este contexto se plantean el resto de los objetivos de este trabajo, que se enumeran a continuación:

- Utilizar la arquitectura de agente inteligente en la concepción y diseño de un robot móvil. En este sentido se pretende mostrar las posibilidades que este concepto ofrece a la hora de implementar las diferentes capacidades de percepción, razonamiento y actuación.

- Se analizarán las configuraciones que cumplan con las especificaciones en cada caso y permitan tener un buen grado de maniobrabilidad. Con ello se pretende facilitar su control lo que puede proporcionar una mayor fiabilidad de los procedimientos que se desarrollen en este sentido. Se realizará un estudio al mayor detalle posible de las especificaciones de cada uno de los elementos electrónicos y de programación que constituirán los robots, además se realizara una propuesta de la estructura para dichos funcionamiento

- Finalmente, se plantea la lógica borrosa como una solución adecuada justificándose por el hecho de que en otros ámbitos y en el de la robótica ha proporcionado prometedores resultados, centrándolo en la realización de un robot seguidor de trayectorias así como la realización del agente inteligente que lo integra de una forma electrónica y de programación, realizando una propuesta de la estructura del robot.

III.I.- ESTRUCTURA DE LA MEMORIA DE TESIS.

Esta memoria comienza con un pequeño repaso de los aspectos clave utilizado en las distintas etapas de este trabajo. A continuación se describe la arquitectura del sistema como agente, desde el punto de vista de los objetivos que se desean conseguir. En el siguiente capítulo se realizara la propuesta de diseño electrónico y de software en la creación de un robot seguidor de trayectorias, basado en la IA.

Al final se realizan las conclusiones correspondientes al manejo de las matrices PAMA, del sistema de locomoción, de la lógica difusa y de los métodos de programación utilizados para este trabajo de tesis.

1

ROBÓTICA MÓVIL

En este capitulo se referenciaran los datos mas importantes de forma conceptual sobre la robótica, se realizará una descripción de la Inteligencia Artificial (IA) como solución aplicada a los móviles robotizados así como el concepto de lógica difusa que se aplicara como uno de los principales métodos de inteligencia.. Así mismo veremos los agentes integradores de la inteligencia artificial y los distintos componentes y comportamientos de un robot autónomo.

1.1. Robótica móvil.

Para definir la robótica móvil, es necesario remontarse al término "robot", viene del vocablo checo *robota*, "servidumbre", "trabajo forzado" o "esclavitud", especialmente los llamados "trabajadores alquilados" que vivieron en el Imperio Austrohúngaro hasta 1848.

La robótica móvil comenzó en el año de 1953 con el primer robot de este tipo llamado ELSIE (Electro-Light-Sensitive Internal-External), construido en Inglaterra. Como consecuencia a este primer robot móvil en la década de los 70 se crea SHACKEY del SRI (standford Research Institute), que estaba provisto de una diversidad de sensores así como una cámara de visión y sensores táctiles y podía desplazarse por el suelo, creado como banco de pruebas para estudiar técnicas de Inteligencia Artificial. En los años 80 se tiene el despliegue definitivo debido al abaratamiento y mejores prestaciones de los sistemas computacionales.

La robótica móvil cubre muchos campos de aplicación, como son transporte de materiales, labores de limpieza, vigilancia y prospección, guiado de personas, así como aplicaciones militares. Los robots autónomos son sistemas mecánicos que pueden interactuar con un ambiente, y tienen un comportamiento individual aunque éste sea socializado. Las dificultades que se encuentran los robots autónomos son los círculos impredecibles y dinámicamente cambiantes, así como su limitada autonomía y sobre todo su posicionamiento en un determinado ambiente. Para evitar estos problemas se suelen utilizar robots guiados, por lo que se establecerá una comunicación con un ordenador central que dirigirá sus movimientos e incluso coordinará varios robots colaboradores.

El substancial problema de los robots guiados es que se restringen a caminos preestablecidos y siempre requieren algún tipo de adaptación en cada reestructuración del entorno.

Sin embargo los robots móviles autónomos no se limitan a caminos preestablecidos, ya que disponen de una capacidad de percepción. Son capaces de percibir el entorno e incluso de realizar una representación de él. También se les puede dotar de la capacidad de planificar sus movimientos evitando obstáculos, así como de detectar marcas o lugares característicos para conseguir una meta.

Los robots móviles deben tener características de maniobrabilidad, controlabilidad, capacidad de tracción, estabilidad, eficiencia y consideraciones de navegación.

Pueden tener multitud de sistemas de tracción según el medio en el que se vayan a mover se podrían adaptar partes como patas, orugas, ruedas, deslizadores, etc.

Una de sus principales dificultades es determinar su posición dentro de una zona determinada. Se pueden realizar estimaciones según una posición inicial bien utilizando sistemas de medidas absolutas respecto a una referencia externa, como balizas, satélites, etc.

El mecanismo de posicionamiento se basa en el sistema de percepción de que disponga. Existen multitud de sistemas sensores como sonar, laser, infrarrojos, radares, etc., aunque no es el único fin de estos, ya que la percepción del robot le ha de servir también para interaccionar con los elementos del ambiente por el que se mueve.

Todo robot móvil se debe plantear las dos preguntas básicas:

- ¿dónde estoy?
- ¿cómo llego al objetivo?

Sin duda para resolver estas cuestiones se debe recurrir a la planificación de movimientos por medio de los actuadores del sistema y a técnicas de planificación, manteniendo quizá una representación del ambiente

1.2. Cinemáticas en robótica móvil.

Existen múltiples sistemas para proporcionar movilidad a un robot, siendo las características de maniobrabilidad y controlabilidad las más importantes. Sin embargo, existen otros aspectos básicos como la eficiencia o estabilidad que no deben ser olvidados. En el caso de micro-robots como el que nos ocupa, se suelen utilizar sistemas basados en ruedas. Se van a revisar a continuación los distintos sistemas rodados de traslación en la robótica.

1.2.1. Configuración diferencial.

Este sistema es uno de los más sencillos, se caracteriza por tener dos motores con reductora acoplados a 2 ruedas diametralmente opuestas, en un eje perpendicular al robot para su desplazamiento.

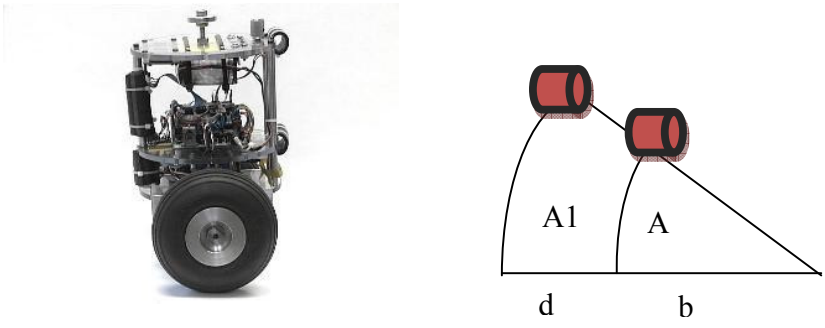


Figura 1.1. Principales parámetros de la configuración diferencial

Considerando que A_1 y A_2 son las distancias recorridas por la rueda izquierda y derecha (respectivamente) y d es la distancia entre los dos puntos de contacto de las ruedas en el suelo, entonces el desplazamiento

lineal *A* del punto central del robot se calcula mediante la siguiente expresión:

$$A = \frac{A1 + A2}{2}$$

El cambio *en el Angulo* del robot vendrá dado por:

$$\alpha = \frac{A1 - A2}{2}$$

Como principal inconveniente tiene la dificultad para trazar una línea recta. Cualquier pequeña desigualdad en las ruedas o en el terreno hace que pierda la línea que estaba trazando.

1.2.2. Configuración síncrona.

Se trata de un ensamblado donde todas las ruedas son direccionales y tienen tracción.

Este sistema tiene la ventaja de tener un control sencillo al poder separar la tracción de la rotación. Además el seguimiento de una línea recta lo asegura la propia configuración mecánica. Sin embargo su principal inconveniente es la complejidad de diseño e implementación.

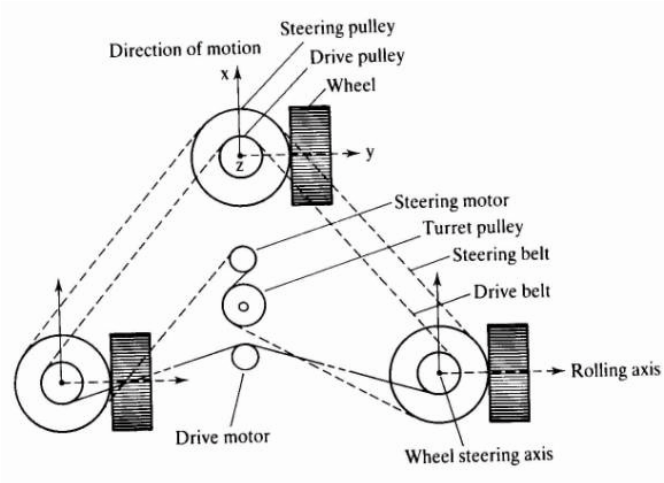


Figura 1.2. Sistema *Sincro drive*

1.2.3. Ruedas omnidireccionales.

Este sistema permite realizar movimientos complejos debido a la disposición de las ruedas. Sin embargo mecánicamente tiene dificultades para seguir líneas rectas y su implementación es muy complicada.

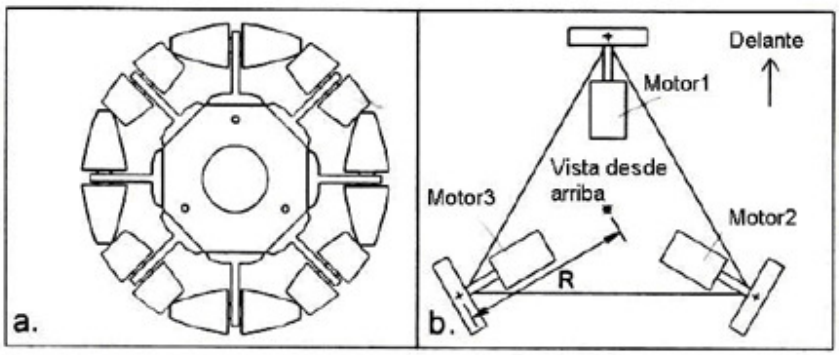


Figura 1.3: Sistema de tracción omnidireccional

1.2.4. Triciclo.

Este sistema se basa en tres ruedas, siendo una de ellas la directriz. La principal ventaja es que no hay deslizamientos de las ruedas como ocurría en la configuración de ruedas omnidireccionales. La desventaja es que tiene una cinemática complicada. Si se tiene la tracción en las ruedas traseras, plantea problemas al girar en curvas, ya que la rueda interior gira a la misma velocidad que la exterior.

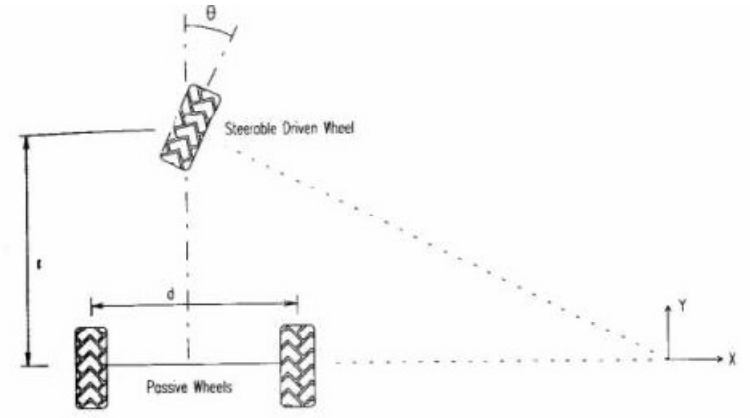


Figura 1.4: Configuración de un triciclo

1.2.5. Sistema Ackerman.

Es el sistema que normalmente utilizan los automóviles, se basa en cuatro ruedas, siendo dos de ellas directrices. Su ventaja es que es fácil de implementar, pero tiene una cinemática muy complicada.

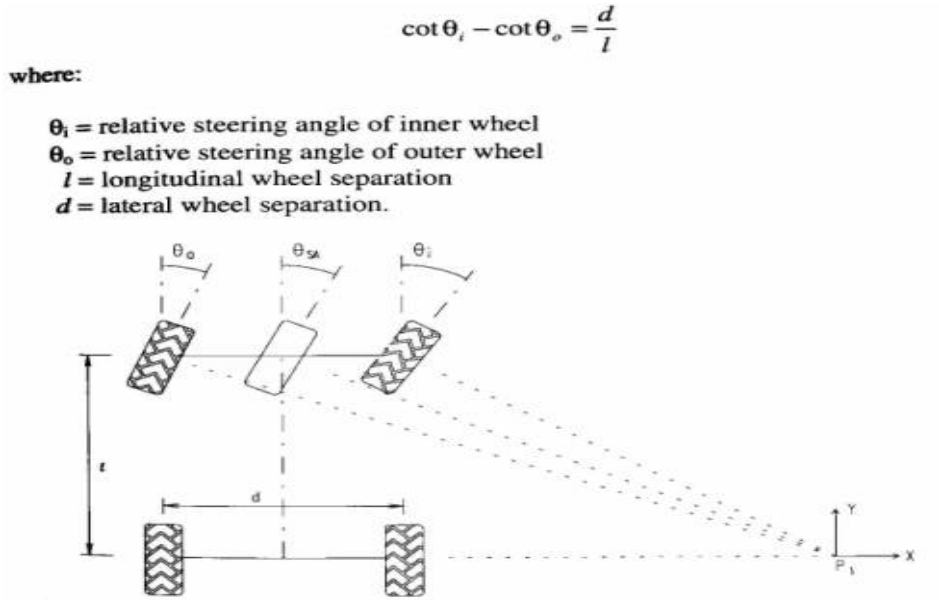


Figura 1.5: Configuración Ackerman

1.2.6. Sistema articulado.

En este sistema los dos ejes tienen capacidad directiva y de tracción. Es simple de implementar, pero es muy complicado de manejar

Figura 1.6. Sistema con dirección articulada

1.3. Aplicaciones

Existe multitud de documentación referente a robots móviles. Podemos realizar una revisión de acuerdo a los fines para los que han sido diseñados. Sin embargo es posible afirmar que los sistemas robóticos aplicados a algún campo se basan en una parte que proporciona la movilidad, junto con otra parte que proporciona la aplicación específica.

1.3.1. Aplicaciones Industriales

La utilización de robots industriales está ampliamente extendida en todo tipo de fábricas y empresas industriales, obteniendo con ellos reducción de costes, aumento de la productividad, mejora de la calidad en la producción y eliminación de condiciones peligrosas de trabajo o mejora de las mismas. De este modo, la empresa industrial, a través de inversiones tecnológicas en el campo de la automatización industrial, podrá aumentar su competitividad en el mercado, corriendo el riesgo de quedarse rezagada en el mercado si descartase la utilización de la robótica en sus procesos de fabricación.

El principal papel de los robots es articular diferentes máquinas y funciones productivas; transporte, manejo de materiales, maquinado, carga y descarga, etc. mediante su capacidad para desempeñar diversas tareas u operaciones. El robot industrial ha sido descrito como el elemento más visible de la fabricación asistida por computador y como la base técnica para la mayor automatización de la producción. La inmensa mayoría de los robots industriales se componen de un brazo articulado, a través del cual desempeñan su tarea en una cadena de producción. Este tipo de robots se sale del marco de nuestra asignatura, por no ser ni 'micro robots', ni 'móviles', de modo que dejaremos de lado esta amplia

familia para indagar en robots más próximos a los tratados en clase, en busca de cierta inspiración para el posterior diseño de nuestros 'micro robots móviles'

1.3.1.1. Inspección y limpieza de ductos.

En este apartado dichos robots son los encargados de inspeccionar y/o limpiar los conductos de aireación, ya que debido a su pequeño tamaño y peso, se desenvuelven con soltura dentro de los conductos, asimismo son fáciles de transportar debido al control mediante ordenador portátil, con el que se controlan todas las opciones del robot limpiador (variación de luz y potencia de los motores) y sirve de almacenamiento de imágenes y vídeos. También son capaces de tomar muestras para su posterior análisis.

Además pueden mostrar sobre la pantalla de grabación variables físicas del interior de los conductos, como la temperatura y la humedad relativa, que quedan almacenados junto con la grabación de vídeo o fotos. También existen cepillos de limpieza para conductos de aireación, fabricados especialmente para los robots de limpieza.

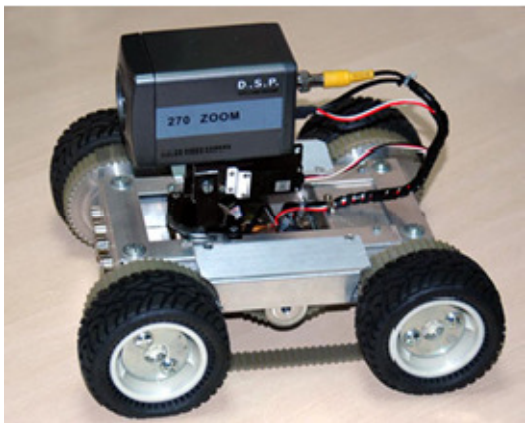


Figura 1.8.i-Bot (Robot limpia tubería

1.3.1.2. Domésticos y de oficina.

Las aplicaciones de robots móviles en el ámbito domestico y de oficina, es un largo camino hacia resultados perfectos, sin embargo, estamos empezando a ver la aparición de robots no excesivamente inteligentes, pero sí eficientes para ciertas tareas concretas. Ya podemos comprar robots que realizan tareas sencillas, como pueden ser la de pasar la aspiradora, cortar el césped, entretener a los niños o realizar pequeñas tareas de servicio como traer café o aperitivos. Esta perspectiva parece indicar que en un futuro no muy lejano puede aparecer lo que podríamos denominar el 'robot personal', un robot doméstico o de oficina de propósito general

1.3.1.4. Limpieza de pisos.

El robot de limpieza VC-RP30W de Samsung es un 'robot aspiradora' que utiliza el principio de mapeo similar al usado por los sistemas de misiles de alta tecnología, y 'dibuja' un mapa en tres dimensiones del ambiente en donde se encuentra para identificar su ubicación relativa, permitiéndole una limpieza más rápida y efectiva del área definida.

El VC-RP30W sabe cuál área debe ser limpiada, logrando un resultado mucho más exacto. Asimismo, con esta unidad, el usuario puede programar el tiempo de trabajo y opciones de limpieza avanzadas, de tal modo que el robot limpia el área automáticamente mientras el usuario está fuera de casa.

Figura 1.8. Robot de limpieza VC-RP30W

1.3.2. Robots de competencias.

Concursos y competiciones Al igual que ocurre con las competiciones de automovilismo o motociclismo, los concursos de robótica pretenden ser el punto de evaluación y comparación de las distintas tecnologías aplicadas a la construcción de robots autónomos.

Existen multitud de competiciones de este tipo. Quizá la más representativa a nivel internacional sea RoboCup (figura 1.9(a)). En esta competición se trata de jugar un partido de futbol con robots autónomos, los cuales son capaces de interactuar con otros robots y con el medio.

También existen pruebas específicas con otros objetivos, muchas veces auspiciadas por casas comerciales.

Sin duda la competición más importante a nivel internacional es HISPABOT (figura 1.9(b)). En este concurso existen las pruebas de rastreadores, sumo, laberinto y velocista, en las que se valoran muchos de los aspectos básicos en robótica, como la controlabilidad, interacción con el ambiente, técnicas de planificación, etc.

a) **b)**

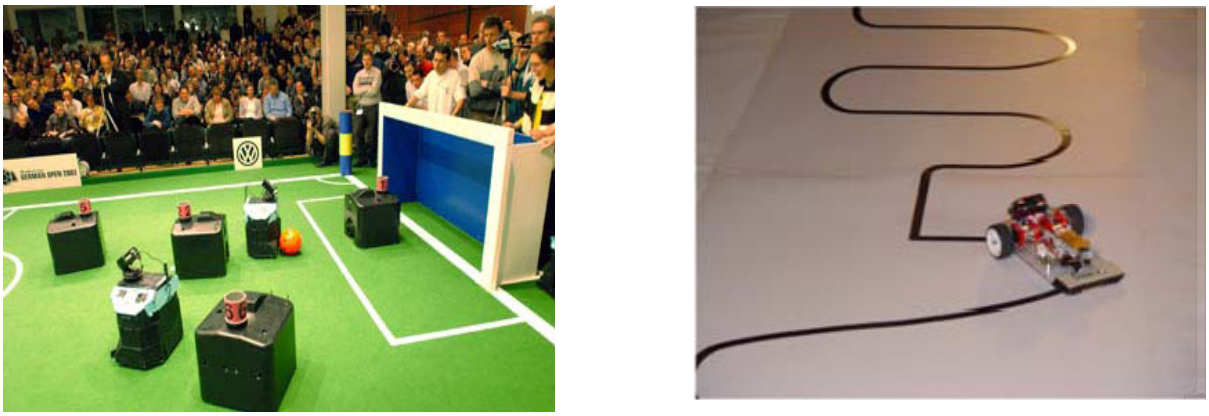


Figura 1.9. Robots de competencia

1.4. Inteligencia Artificial.

Él termino "inteligencia artificial" fue acuñado formalmente en 1956 durante la conferencia de Darthmounth, más para entonces ya se había estado trabajando en ello durante cinco años en los cuales se había propuesto muchas definiciones distintas que en ningún caso habían logrado ser aceptadas totalmente por la comunidad investigadora. La Al es una de las disciplinas más nuevas que junto con la genética moderna es el campo en que la mayoría de los científicos " más les gustaría trabajar".

Una de las grandes razones por la cuales se realiza el estudio de la IA es él poder aprender más acerca de nosotros mismos y a diferencia de la psicología y de la filosofía que también centran su estudio de la inteligencia, IA y sus esfuerzos por comprender este fenómeno están encaminados tanto a la construcción de entidades de inteligentes como su comprensión.

La Inteligencia Artificial (IA) es un campo científico en el que se trata de las que máquinas realicen tareas que podría realizar el hombre aplicando cualquier tipo de razonamiento. Para lograr este objetivo se necesitan programas informáticos.

En cierta medida cualquier programa informático puede ser considerado *inteligente*. El problema es diferenciar entre qué es lo que se considera un programa inteligente y qué no lo es.

Un programa inteligente es aquel que muestra un comportamiento similar al humano cuando se encuentra con un problema idéntico. No es necesario que el problema sea resuelto exactamente de la misma manera que lo haría una persona. Desde el punto de vista del comportamiento humano, la IA estudia la naturaleza de la inteligencia humana, como reproducirla e implementarla en un programa informático.

Sin embargo hay que diferenciar entre la inteligencia humana, y lo que se conoce como inteligencia ideal. En el primer caso se trata de la limitada

inteligencia de las personas, que no coincide con la inteligencia ideal o racionalidad. Se considera que un sistema es racional si siempre hace lo correcto. El enfoque centrado en el comportamiento humano constituye una ciencia empírica, que entraña el empleo de hipótesis y confirmación mediante experimentos. El enfoque racionalista combina matemáticas e ingeniería. Ambos enfoques son las bases de sendos grupos de investigación que muchas veces critican el trabajo del otro grupo. Sin embargo ambas orientaciones han hecho valiosas aportaciones.

1.5. Agentes inteligentes (Software y Hardware).

Un agente inteligente es un software capaz de realiza de forma autónoma tareas que requieren cierto grado de inteligencia y aprendizaje".

Los Agentes Inteligentes tienen una filosofía integradora de las distintas tecnologías utilizadas en Inteligencia Artificial. Se trata de una especificación formal que a una los distintos elementos hardware y software para el desarrollo de agentes inteligentes

Un Agente Inteligente es todo sistema que percibe su ambiente mediante sensores y que responde o actúa de forma inteligente para conseguir un determinado objetivo.

Algunos de los agente robóticos fundamentales son, sensores, cámaras, infrarrojos, etc. Los efectores son reemplazados mediante motores. En el caso de un agente software, sus percepciones y acciones vienen a ser las cadenas de bits codificados. Los agentes inteligentes se basan en la capacidad de percibir el ambiente que les rodea y tomar una decisión basándose no solamente en su sistema sensorial, sino también en su estado interno. De esta forma el agente será capaz de conseguir una meta para el que ha sido diseñado. Un agente inteligente es capaz de tener una memoria que representa el ambiente con el que interactúa.

El término medición de prestaciones trata de definir qué éxito ha tenido un agente al realizar una tarea. Desde luego no existe una medida fija que se pueda aplicar a todos los agentes. Sin embargo se deben disponer de sistemas eficientes de evaluación de prestaciones, de forma que permitan tomar uno u otro criterio posterior en función de este resultado.

Hay que dejar claro que existe una diferencia entre racionalidad y *omnisciencia*. Un agente omnisciente es aquel que sabe el resultado real que producirán sus acciones y su conducta es congruente con ello. Sin embargo en la realidad no existe la omnisciencia. No se puede culpar a un agente por no haber tomado en cuenta algo que no podía percibir, o por no emprender una acción de la que es incapaz. Sin embargo, la tolerancia en relación con la exigencia de perfección no es algo que tenga que ver con una actitud justa en favor de los agentes. Lo importante es que si se especifica que un agente inteligente siempre debe hacer lo que realmente es lo correcto, será imposible diseñar un agente para que satisfaga esta especificación.

La mayoría de los agentes poseen las siguientes tres características:

- Comunicación. El agente puede comunicarse con el usuario, con otros agentes y con otros programas. Con el usuario se comunica con un interfaz amigable, mediante el que personaliza sus preferencias. Algunos agentes permiten comunicarse en lenguaje natural, algo típico de los chatbots.

- El grado de inteligencia varía mucho de unos agentes a otros, que suelen incorporar módulos con tecnologías procedentes de la Inteligencia Artificial. Los más sencillos se limitan a recoger las preferencias del usuario, quien debe personalizarlos. Un ejemplo son los agentes inteligentes basados en tecnología de redes neuronales especializados en identificar mensajes de correo electrónico sospechosos de contener spam -mensajes no deseados- En una primera fase el usuario debe marcarlos como spam, el agente va

aprendiendo a identificar los rasgos que caracterizan a estos mensajes y posteriormente los filtra.

- Autonomía. Un agente no sólo debe ser capaz de hacer sugerencias al usuario sino de actuar. En el ejemplo anterior, el agente que filtra el spam no puede estar continuamente alertando al usuario en cada mensaje de correo que llega sobre la posibilidad de que sea un mensaje no deseado y su verdadera utilidad surge cuando elimina de forma autónoma dichos mensajes.

1.6. Lógica difusa.

La lógica difusa ha cobrado una fama grande por la variedad de sus aplicaciones, las cuales van desde el control de complejos procesos industriales, hasta el diseño de robots móviles, pasando por la construcción de artefactos electrónicos de uso doméstico y de entretenimiento, así como también de sistemas de diagnóstico. La expedición de patentes industriales de mecanismos basados en la lógica difusa tiene un crecimiento sumamente rápido en todas las naciones industrializadas del orbe.

Las lógicas difusas, pues de hecho hay que hablar de ellas en plural, son esencialmente lógicas multivaluadas que extienden a las lógicas clásicas las cuales deben su nombre a que imponen a sus enunciados únicamente valores *falso* o *verdadero*. Bien que las lógicas clásicas han modelado satisfactoriamente a una gran parte del razonamiento ``natural'', es cierto que el razonamiento humano utiliza valores de verdad que no necesariamente son ``deterministas''.

Las lógicas difusas tratan de crear aproximaciones matemáticas en la resolución de ciertos tipos de problemas. Pretenden producir resultados exactos a partir de datos imprecisos, por lo cual son particularmente útiles en aplicaciones electrónicas o computacionales.

El adjetivo ``difuso'' aplicado a estas lógicas se debe a que en ellas los valores de verdad no-deterministas utilizados tienen, por lo general, una connotación de incertidumbre. Un vaso medio lleno, independientemente de que también esté medio vacío, no está lleno completamente ni está vacío completamente. Qué tan lleno puede estar es un elemento de incertidumbre, es decir, de difusidad, entendida esta última como una propiedad de indeterminismo. Ahora bien, los valores de verdad asumidos por enunciados aunque no son deterministas, no necesariamente son desconocidos. Por otra parte, desde un punto de vista optimista, lo difuso puede entenderse como la posibilidad de asignar más valores de verdad a los enunciados que los clásicos ``falso'' o ``verdadero''. Consecuentemente, las lógicas difusas son tipos especiales de lógicas multivaluadas.

Se ha considerado de manera general que la lógica difusa se inició en 1965, en la Universidad de California en Berkeley por Lotfi A. Zadeh

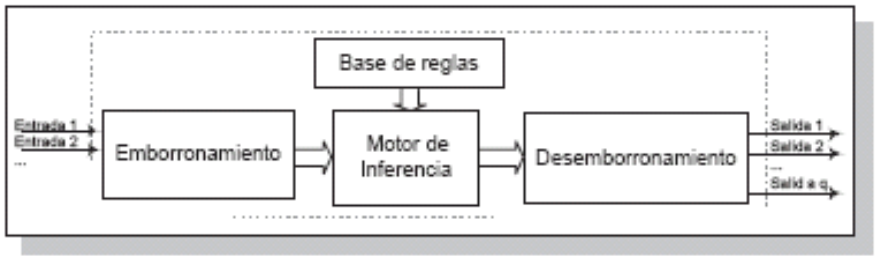


Figura 1.10. Etapas de la lógica difusa.

1.6.1. Localización topológica basada en la visión para robots móviles.

La localización de un robot en su entorno es uno de los problemas clásicos de la robótica móvil. Es el punto de partida de otros problemas como son la interacción con el entorno, y sobre todo, la navegación. Por localización se entienden las

técnicas que se encargan de solucionar el problema de determinar la posición de un robot en su entorno, utilizando sus propios sensores.

En entornos interiores existen una gran cantidad de tareas que un robot puede realizar: llevar correo de un despacho a otro, tele asistencia, tareas de seguridad, etc. Pero para realizarlas, un robot debe conocer su posición para trazar la ruta desde su posición a su destino. Lo que se pretende es detectar la posición de un robot móvil en un entorno de oficinas para que pueda navegar por los pasillos de un despacho a otro. Usa como principal sensor las imágenes de una cámara, y utiliza también un sensor de infrarrojos. La información que obtiene de la imagen de la cámara son elementos comunes del entorno, como puertas, luces del techo y elementos que están a ambos lados del robot.

A las posibles posiciones y orientaciones en las que el robot se puede encontrar se les llama "estados". Los estados son definidos sobre el entorno de la oficina formado por pasillos y habitaciones. No tienen ninguna relación métrica, pues esta navegación se basa en una visión del espacio puramente topológica, en la que las diferentes posiciones se determinan mediante una división del entorno en distintos nodos.

El proceso es el siguiente; primero se identifican los nodos como partes del mapa, una vez obtenidos, se obtienen los estados en los que el robot se puede encontrar. Las acciones que el robot puede tomar en cada posición son simples: avanzar hasta el siguiente nodo, girar hacia la izquierda o girar hacia la derecha.
Por ejemplo, si un robot está orientado hacia el fondo del pasillo y a ambos lados tiene pared, y decide realizar la acción de avanzar, el robot avanzará hacia delante hasta que en alguno de los lados deje de encontrar pared. En ese momento, mediante un proceso basado en el funcionamiento visto antes de los sistemas de control borroso, el robot decidirá entre las acciones que enumerábamos antes.

2

ARQUITECTURA DE LOS SISTEMAS CON INTELIGENCIA ARTIFICIAL

En este capitulo retomaremos los agentes inteligentes y realizaremos algunas aplicaciones de este en robots de competencia como solución deseada, basada en el agente de estructura PAMA.

Existen diferentes tipos de agentes, pero en términos generales todos actúan en un ambiente desde el cual reciben percepciones y sobre el que realizan acciones que en algunos casos pueden estar basadas en un cierto estado interno.

2.1. Agentes.

En el capitulo anterior conocimos la idea de un agente mediante la descripción de su conducta. Estas son las acciones que se producen después de una determinada secuencia.

Uno de los cometidos de la IA es el diseño de un programa de agente: una función que permita implantar la idea del agente para pasar de percepciones a acciones. Se supone que este programa se ejecutará en algún tipo de dispositivo de cómputo, al que se le denominará arquitectura. Desde luego, el programa elegido debe ser aquel que la arquitectura acepte y pueda ejecutar dentro de sus limitaciones de potencia. Esta arquitectura podría también incluir un software que ofrezca cierto grado de aislamiento entre la computadora y el programa de agente, lo que permitiría la programación a un nivel superior. En general la arquitectura pone al alcance del programa las percepciones obtenidas mediante los sensores, lo ejecuta y alimenta al efector con las acciones elegidas por el programa conforme éstas se van generando. La relación entre agentes, arquitectura y programas podría resumirse de la siguiente manera:

agente = arquitectura + programa

Antes de proceder al diseño de un programa de agente, es necesario contar con una idea bastante precisa de las posibles percepciones y acciones que inter- vendrán, qué metas o medidas de prestaciones se supone lleve a cabo el agente, así como el tipo de ambiente en que tal agente operará. Estos datos reunidos se denominan matriz PAMA (Percepciones, acciones, metas y ambiente)

Puede sorprender el hecho de que haya agentes que funcionen en un ambiente totalmente artificial, como por ejemplo un programa de traducción o

de busque- da. Lo que verdaderamente importa no es la diferencia entre ambientes reales y artificiales, sino la complejidad de la relación que existe entre la conducta del agente, la secuencia de percepciones que produce el ambiente y las metas que se espera alcance el agente. Algunos ambientes considerados como reales, de hecho son muy sencillos. Por ejemplo un robot diseñado para inspeccionar diversas piezas que pasan ante él en una cinta transportadora. Este puede suponer que la iluminación será siempre la misma, que lo único que está sobre la cinta son piezas conocidas y que solo se ejecutan dos acciones: aceptar y rechazar.

En contraste, en algunos muy ricos e ilimitados ámbitos, se utilizan agentes de software, también conocidos como robots software o *softbots*. Por ejemplo un agente diseñado para pilotar el simulador de vuelo de un 747. El simulador constituye un ambiente muy complejo y detallado. En él el agente de software debe elegir entre una amplia gama de acciones en tiempo real.

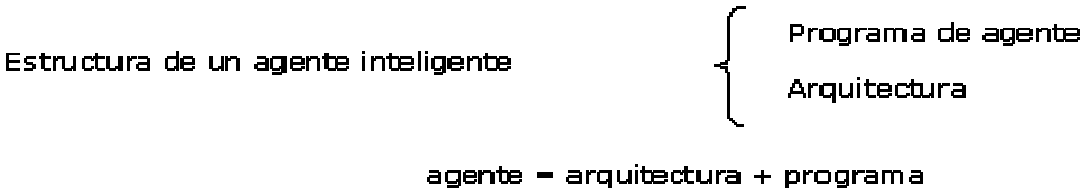
El más famoso de los ambientes artificiales es la prueba de Turing, caracterizado porque tanto agentes reales como artificiales se encuentran en igualdad de condiciones, pero el ambiente plantea tantos desafíos que resulta muy difícil para un agente de software desempeñarse tan bien como un humano. En la figura 2.1 se puede observar el esqueleto de un agente. Como se puede ver, cada vez que se solicite se actualiza la memoria para que refleje la nueva percepción, se escoge la mejor acción y también se refleja en la memoria la acción seleccionada. Obviamente la memoria es persistente de una solicitud a otra.

```
Función ESQUELETO-AGENTE(percepción) return acción.
        Static: memoria

memoria ← ACTUALIZACIÓN-MEMORIA(memoria,percepción)
acción ← ESCOGER LA MEJOR ACCIÓN(memoria)
memoria ← ACTUALIZACIÓN-MEMORIA(memoria,acción)

return acción
```

Figura 2.1. Forma básica de un agente.

2.1.1. Estructura de los agentes

La base de la Inteligencia Artificial es el diseño de un programa de agente: Una función que permita implantar el mapeo del agente para pasar de percepciones a acciones. Este programa se ejecutará en algún tipo de dispositivo de cómputo al que se denominará arquitectura. La arquitectura puede ser una computadora sencilla o un hardware especial.



Elementos básicos que se consideran en la elección de los tipos de agente:

Tipo de agente	*Percepciones*	*Acciones*	*Metas*	*Ambiente*
Sistema diagnóstico médico	-Sistemas -Evidencias -Respuestas del paciente	-Preguntas -Pruebas -Tratamientos	Paciente saludable	Paciente hospital
Robot clasificador de partes	Pixels de intensidad variable	Recoger partes y clasificarlas	Poner las partes en el bote correspondiente	Banda transportadora

Tipos de Agente y sus descripciones PAMA

Figura 2.2. Agentes PAMA.

PAMA (Percepciones, Acciones, Metas y Ambiente)

Softbots (Agentes de software o robots de software)

2.1.1.1. Programas de Agente

function FORMA-AGENT (percept) **returns** Action

Función FORMA-AGENTE (percepción) **responde con una** acción

estática: memoria (la memoria del mundo del agente)

memoria ¬ ACTUALIZACION-MEMORIA (*memoria, percepción*)

acción ¬ ESCOGER-LA-MEJOR-ACCION (*memoria*)

memoria ¬ ACTUALIZACION-MEMORIA (*memoria, acción*)

responde con una acción

El esqueleto de un agente. Cada vez que así se solicite, se actualiza la memoria para que refleje la nueva percepción, se escoge la mejor acción y también se consigna en la memoria la acción emprendida. La memoria persiste de una solicitud a otra.

function TABLE-DRIVEN-AGENT (percept) **returns** action

función AGENTE-CONDUCIDO-MEDIANTE-TABLA (percepción) **responde** con una acción

estático: percepciones, una secuencia, originalmente está vacía

tabla: una tabla indizada mediante secuencia de percepciones originalmente especificada en su totalidad

añadir la percepción al final de todas las percepciones

acción ¬ CONSULTA (*percepciones, tabla*)

devolver acción

Un agente basado en una tabla de consulta previamente especificada. Se mantiene al tanto de la secuencia de percepciones y se limita a definir cuál es la mejor acción. Carecen de autonomía, pero son válidos.

Antes de proceder al diseño de un programa de agente es necesario contar con una idea bastante precisa de las posibles percepciones y acciones que intervendrán, qué metas o medidas de desempeño se supone lleve a cabo el agente, así como del tipo de ambiente en que tal agente operará, para tal efecto, denominamos a la matriz PAMA (percepciones, acciones, metas y ambientes).

2.1.2 Tipos de agentes

En este apartado se realizara el análisis de cada agente inteligente

2.1.2.1. Agentes de reflejo simple:

Este tipo de agente no contiene internamente estados y sus procesos o acciones que realiza son respuestas a la entrada de percepciones, a esta conexión entre percepciones y acciones se las denomina reglas de condición-acción. Ejemplo: Si el carro de adelante está frenando entonces empiece a frenar.

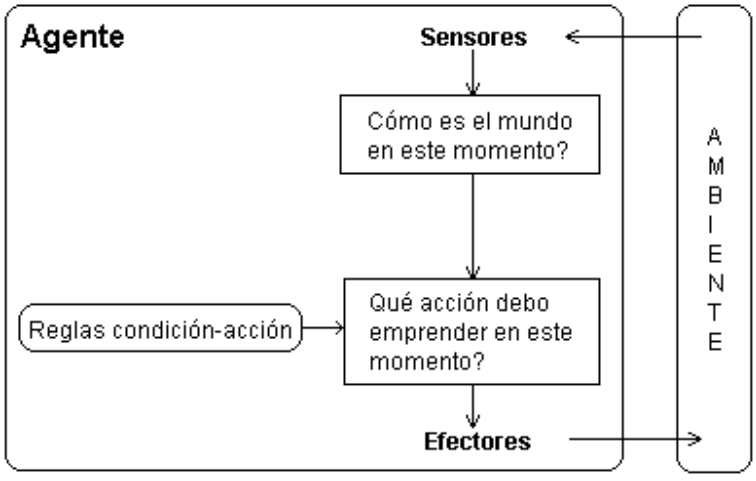


Figura 2.2. Diagrama esquematizado de un agente reflejo simple.

2.1.2.2. Agentes bien informados de todo lo que pasa:

Este tipo de agente guarda estados internos lo que nos sirve sin consideración para ejecutar una acción. Los sensores no nos pueden informar a la vez de todos los estados que maneja nuestro ambiente, es por este caso que el agente necesita actualizar algo de información en el estado interno. Esto le permite discernir que entre estados del ambiente que generan la misma entrada de percepciones pero, sin embargo; para cada uno de los estados se necesitan acciones distintas.

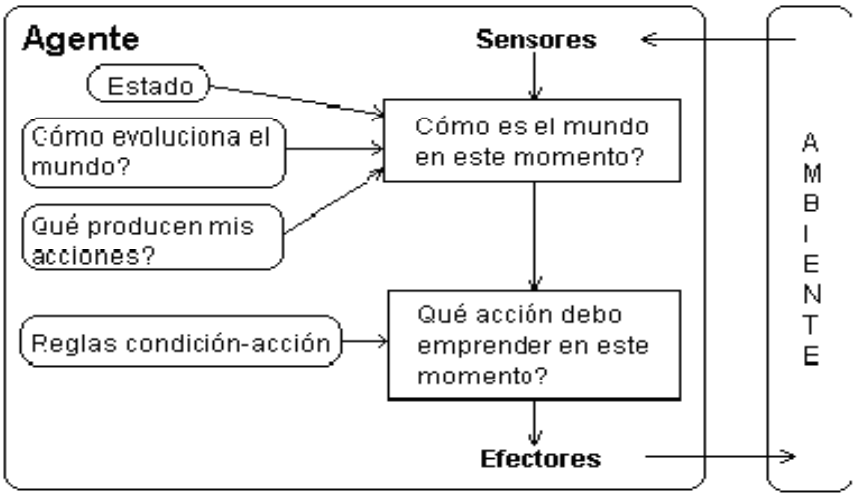


Figura 2.3. Diagrama esquematizado de un agente reflejo con estado interno

2.1.2.3. Agentes basados en metas:

Además de los estados, los agentes necesitan cierto tipo de información sobre sus metas Estas metas van a detallar las situaciones a las que se desea llegar de este modo, el programa de agente puede combinar las metas con la información de los resultados (acciones) que emprenda y de esta manera poder elegir aquellas acciones que permitan alcanzar la meta.

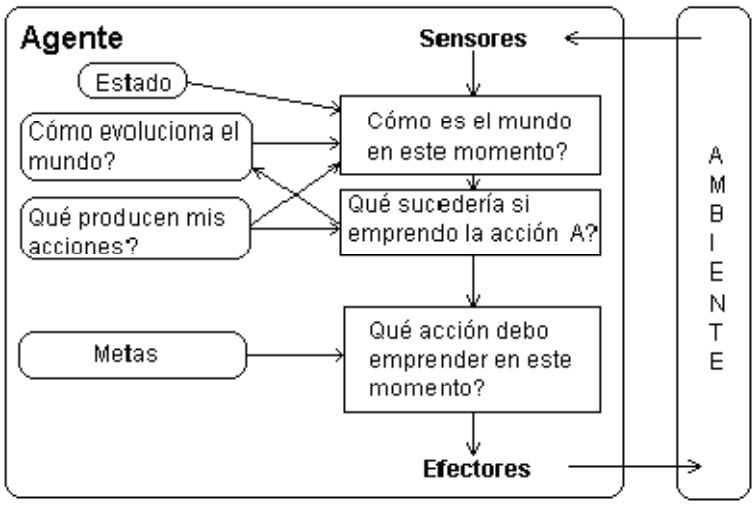


Figura 2.4. Diagrama esquematizado de un agente basado en metas especificas.

2.1.2.4. Agentes basados en utilidad:

Las metas por sí solas me garantizan la obtención de una conducta de alta calidad. En mi programa de agente se podría tener un conjunto de metas pero la obtención de éstas no me garantizan distinciones entre estados felices e infelices, mediante una medida de desempeño se podría establecer una comparación entre los diversos estados del mundo (ambientes) para poder encontrar el estado de felicidad para el agente. Este estado ofrecerá una mayor utilidad al agente.

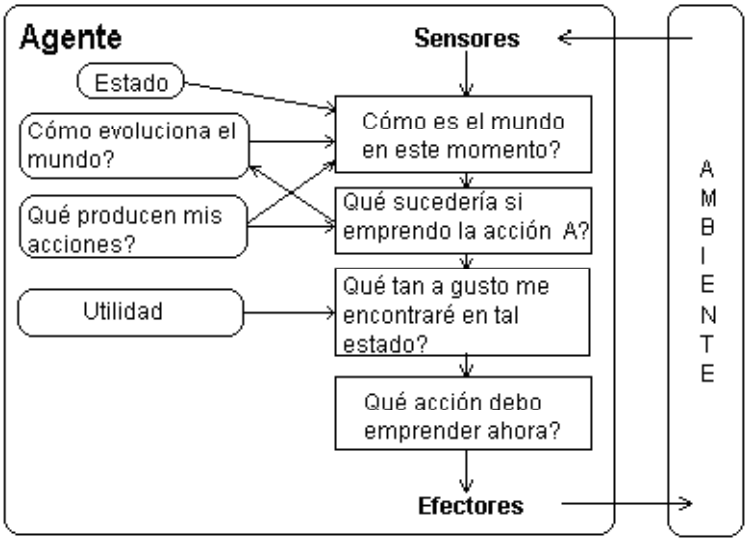


Figura 2.5.Un agente completo basado en la utilidad.

2.2. Características del medio donde interactúa un robot (Ambiente).

En este apartado se va a estudiar el modo de acoplar un agente a un ambiente. La relación que existe de agentes con ambientes es siempre la misma: es el agente quien ejerce acciones sobre el ambiente que a su vez, aporta percepciones al agente. A continuación se verán los diferentes tipos de ambientes y cómo condicionan el diseño de los agentes.

2.2.1. Accesibles y no accesibles.

Si el aparato sensorial de un agente le permite tener acceso al estado de un ambiente, se dice que éste es accesible a tal agente.

2.2.2. Deterministas y no deterministas.

Si el estado siguiente de un ambiente se determina completamente mediante el estado actual y las acciones escogidas por los agentes, se dice que el ambiente es determinista.

2.2.3. Episódicos y no episódicos.

En un ambiente episódico, la experiencia del agente se divide en *episodios*. Cada episodio consta de un agente que percibe y actúa. La calidad de su actuación dependerá del episodio mismo, dado que los episodios siguientes no dependerán de las acciones producidas en episodios anteriores.

2.2.4. Estáticos y dinámicos.

Si existe la posibilidad de que el ambiente sufra modificaciones mientras el agente se encuentra deliberando, se dice que tal ambiente se comporta en forma dinámica en relación con el agente, de lo contrario se dice que es estático.

2.2.5. Discretos y continuos.

Si existe una cantidad limitada de percepciones y acciones distintas y claramente discernibles, se dice que el ambiente es discreto

2.3. Descripción de un agente.

El objetivo de la IA es el diseño de un programa de agente. Para ello es necesario contar con una idea bastante precisa del ambiente, las percepciones y acciones, y por último de las metas. A este conjunto de datos se le conoce como matriz PAMA (Percepción, Acción, Meta y Ambiente).

Tipo de agente	Percepciones	Acciones	Metas	Ambiente
Conductor de taxi	• Cámaras, velocímetro, sistema GPS, sonar, micrófono	• Manejo de volante, acelerar, frenar, hablar con el pasajero	• Un viaje seguro, rápido, sin infracciones, cómodo, obtención de máxima ganancia	• Carretera, tráfico, peatones, cliente

Figura 2.6.Matriz PAMA para un conductor de taxi.

Sus percepciones se justifican por la necesidad de saber dónde se encuentra, quién más circula por su vía y a qué velocidad está circulando. Estos datos se obtienen de su sistema de percepción. También es interesante tener un acelerómetro para medir la dureza de las curvas, así como conocer el estado del vehículo, por lo que necesitaremos varios sensores más.

Las acciones que puede producir este conductor de taxi son más o menos las mismas que las de un humano: control del acelerador, freno y del volante. Además necesitará un sintetizador de voz que le permita contestar al pasajero o a otros conductores. Entre los objetivos a cumplir, principalmente deben ser: llegar de forma segura, con un consumo mínimo, tiempo reducido y respetar las normas de circulación. Desde luego, la consecución de alguno de estos objetivos está en conflicto con la consecución de otros, por lo que será necesario conceder ciertos márgenes. Por último sería necesario definir el ambiente de condición. ¿Irá por la ciudad o por el extrarradio? ¿Habrá problemas externos como nieve, lluvia, etc.? ¿Se conducirá por el lado derecha o por la izquierda? obviamente vemos que mientras menos restringido sea el ambiente, menos complicado será el problema de diseño. La cuestión ahora es, una vez reunidos los datos PAMA, seleccionar el modelo más adecuado de agente, según los tipos ya descritos.

2.4. Aplicación del modelo PAMA en la construcción de un robot móvil.

Para mostrar las posibilidades que el concepto de agente ofrece en la implementación de capacidades de percepción, actuación y consecución de metas en un determinado ambiente, en los siguientes apartados se va a aplicar el modelo PAMA para la construcción de *un* robot móvil. Es difícil establecer la normatividad del diseño del robot para competencias, ya que depende de la institución que este realizando el evento, en concreto para este trabajo se usara la norma del “Torneo Mexicano de Robótica”, organizado por la federación mexicana de robótica, las cuales serán descritas, y a continuación se realizará una revisión de cada uno de los elementos PAMA aplicado a cada una de esas pruebas, concluyendo

finalmente con el tipo de agente más adecuado que se puede utilizar en cada una de las categorías.

2.5. Seguidor de línea.

En esta prueba se valora la capacidad de un robot de realizar un recorrido por una pista con diversas rutas alternativas. Se trata de explotar la capacidad de seguimiento de caminos complejos, y de la capacidad de detección inequívoca de marcas para indicar rutas.

La pista será una línea negra sobre fondo blanco, y medirá entre 1.5 y 2.5 cm. Pueden tener tantas curvas y ángulos como la organización considere oportuno. Para tomar un camino correcto ante una bifurcación, habrá una marca paralela a la pista y separada entre 1 y 2 cm, midiendo entre 4 y 6 cm de largo. Estas marcas estarán a una distancia de entre 10 y 15 cm de una bifurcación. Por ultimo los robots, tendrán una dimensión de 20 x 30 cm. La valoración final se hace por puntos, y en caso de empate por tiempo. Los puntos se acumularán en caso de penalización por haber tomado una bifurcación incorrecta o por tardar más de un tiempo determinado por la organización.

2.5.1 Matriz PAMA del robot seguidor de línea.

Percepciones.

El agente rastreador (seguidor de líneas), tiene que disponer de una percepción suficientemente precisa como para poder realizar un seguimiento de la línea con una cierta precisión y sin oscilaciones. Debido a los requisitos que se plantean en esta prueba, debe ser capaz de detectar las marcas de bifurcación correctamente, para poder tomar el camino correcto. Por lo tanto su sistema de percepción deberá aceptar las tolerancias propias del ancho de la pista, luminosidad, etc.

Debido a la especial característica de la pista de ser negro sobre blanco, resultan adecuados los sensores infrarrojos reflectivos, de forma que si el haz infrarrojo rebota sobre negro, dará un voltaje alto y si rebota en blanco será pequeño.

Ambiente.

El ambiente, como ya se menciono, es una pista negra sobre un fondo blanco. Existe suficiente diferencia de color como para no confundirlos en condiciones normales. Sin embargo debemos ser capaces de filtrar las situaciones no ideales como pista sucia o pequeñas anomalías de los sensores, siendo la alternativa la variación de voltaje que identifica el sensor con cada color en la pista, además del rango de tiempo de ese voltaje para identificar imperfecciones.

Acciones.

El sistema efector de este agente debe ser capaz de realizar las maniobras indicadas en sus metas, de la forma más fiel posible. En esta prueba no prima la velocidad tanto como la capacidad de tomar el camino correcto, por lo que unos motores con suficiente par, que respondan inmediatamente serían adecuados. En este tipo de agente se suelen utilizar servos, ya que aunque no son demasiado veloces, disponen de un par alto y son soluciones cerradas con una fiabilidad grande. Las acciones serán la de moverse por la pista de acuerdo a las percepciones, de una forma lo más parecida posible al movimiento deseado.

Metas.

Dado que la pista es continua y según cada competencia, no se puede fijar como meta un tiempo definido, ya que no se sabe cuando va a tener que parar; así que

será necesario ajustar esta meta conforme a cada competencia y no influirá en su diseño.

Una de sus metas será el seguimiento de la pista de la forma más fiel posible. Otra de sus metas

Tipo de agente	Percepciones	Acciones	Metas	Ambiente
Rastreador	• Píxel blancos o negros	• Aplicar potencia a los motores	• Avanzar sobre la línea • Tomar bifurcaciones correctamente	• Pista negra sobre blanco

Figura 2.7.Matriz PAMA del robot seguidor de líneas.

2.5.2. Agente a utilizar en el robot seguidor de línea.

Para poder elegir que tipo de agente será el adecuado para interactuar con el robot seguidor de líneas, es necesario analizar los agentes que ya conocemos y establecer el trabajo que se va a realizar.

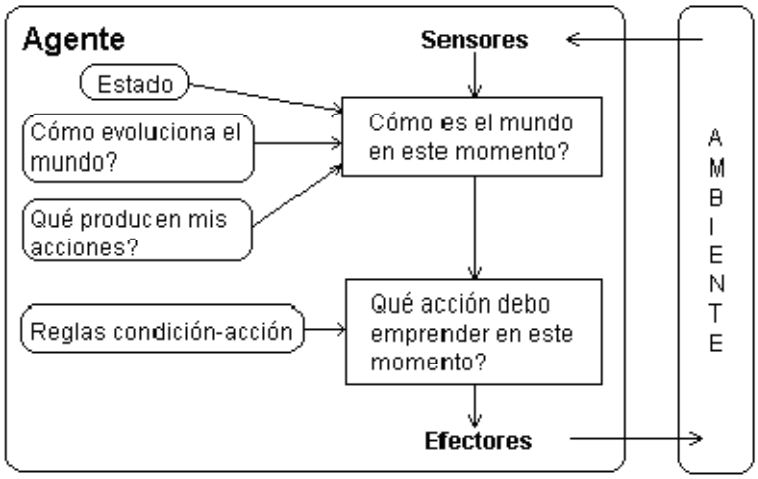


Figura 2.8. Representación de un agente reflejo de estado interno

De acuerdo a la matriz PAMA de la figura 2.7, en esta prueba parece bastante adecuado plantear el agente reflejo con estado interno. No sería posible utilizar un agente de reflejo simple debido a que se necesita memorizar temporalmente hacia qué lado se debe tomar una bifurcación. Tampoco se pueden utilizar los agentes basados en metas o en utilidad, ya que la meta no conduce a un estado final o de parada, sino a un estado intermedio continuo

2.6. Robot móvil de laberinto

El robot de laberinto es uno de los robots más populares ya que este tipo de competencias es de las pioneras en la robótica, al igual que otras competencias el robot esta estandarizado a 30 x 20 cm.

La pista (el laberinto) esta formado por dos líneas negras sobre un fondo blanco, las cuales tendrán una separación de 40cm. Existen distintos tipos de vueltas o bifurcaciones los cuales están definidos en un ángulo de 45º, 60º o 90º.

Estas medidas están restringidas por el tamaño del robot, ya que entre mas grande este saldría de la pista marcada y rompería una regla fundamental de este concurso. El robot ganador se define, por cuestión de tiempo ya que el robot que realice el menor posible será el ganador, en caso de empate ganara el que haya tenido menos errores en las bifurcaciones.

Tiene en cuenta que hay varias formas de llegar a la meta, es decir, que tienen costo distinto El agente basado en utilidad se encarga de encontrar la solución óptima según la utilidad

2.6.1. Matriz PAMA del robot móvil de laberinto.

Percepciones.

Un agente de laberinto debe ser capaz de localizarse dentro del laberinto y salir de él. Desde el punto de visa de percepción, debe ser capaz de percibir las distancias a las paredes que tiene enfrente, atrás y a los lados. De esta forma, y al cabo de realizar algunos movimientos debería conseguir la primera meta que es saber dónde está. La solución del laberinto debería estar pre calculada, por lo que la meta ahora sería tomar el camino correcto para salir. También resulta muy recomendable una capa intermedia en la percepción. Esta capa debería dar una estimación de la casilla en la que se encuentra el agente. Con estas dos consideraciones el agente tendrá una visión de alto nivel del laberinto, aislándolo de la complejidad para poder centrarse simplemente en las metas.

Acciones.

El sistema efector de este agente debe ser capaz de realizar las maniobras indicadas en sus metas, así como mantener una velocidad adecuada para su recorrido. A diferencia del robot seguidor de líneas, este robot tendrá que tener las aptitudes de tomar decisiones y controlar su velocidad ya que es demasiado relevante para el objetivo planeado

Metas.

Una de las metas mas importante es la de tener decisión en cada giro para realizarlo dentro de la pista con gran precisión, a demás es necesario indicarle la meta del tiempo, por su relevancia dentro de esta competencia. La meta final se dará por realizada cuando el robot salga del laberinto en un tiempo reducido, así como la precisión de ellos movimientos.

Ambiente.

El ambiente es un laberinto de 40cm de ancho delineado por un par de líneas negras paralelas entre si, que cuenta con distintas curvas y bifurcaciones.

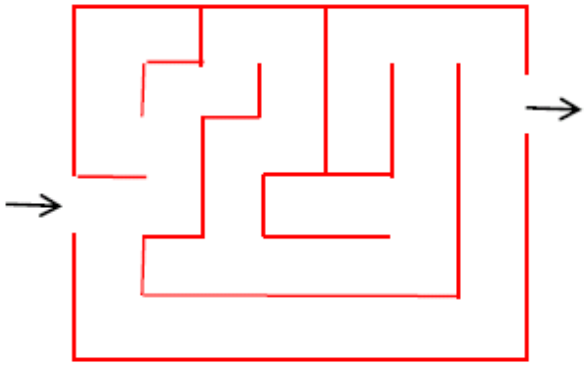


Figura 2.9. Laberinto de la prueba para concurso en el rango de 12 a 16 años en la UVM

Tipo de agente	Percepciones	Acciones	Metas	Ambiente
Laberinto	• Distancia a las cuatro paredes	• Aplicar potencia a los motores	• Salir del laberinto rápidamente	• Laberinto con cos salidas

Figura 2.10.Matriz PAMA del robot seguidor de líneas

2.6.2. Agente a utilizar en el robot de laberinto.

Para poder elegir que tipo de agente será el adecuado para interactuar con el robot de laberinto, es necesario analizar los agentes que ya conocemos y establecer el trabajo que se va a realizar.

El efecto de cada acción se calcula por la "distancia en L" de cada parte de la pista, para ello utilizaremos un agente basado en metas, ya que este identifica las acciones y las analiza hasta llegar a la meta y no para hasta llegar a ella.

Es decir, este agente por metas modela al efecto con forme a el avance realzado regresa al análisis para seguir buscando su meta

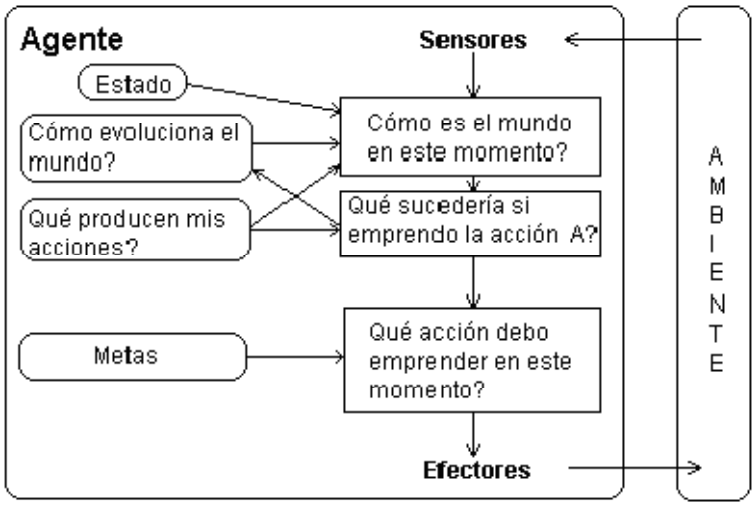


Figura 2.11. Diagrama esquematizado de un agente basado en metas especificas

2.7. Robot luchador (sumo)

En esta prueba se compite contra otro robot. El objetivo es expulsar al adversario del tatami. El tatami no es más que un tablero circular que mide 175 cm de diámetro de color negro. Existe una línea blanca de 5 cm de ancho para delimitar el borde del tatami. Los robots no podrán pesar más de 3 kg y medir 20 × 20 cm como máximo en su situación de reposo. Es posible que posteriormente al arranque desplieguen palas o planchas para facilitar la expulsión del adversario. La altura del robot será libre. Al igual que las competiciones de sumo humanas, no se podrá golpear ni provocar desperfectos deliberadamente, existiendo un sistema de penalización para ello.

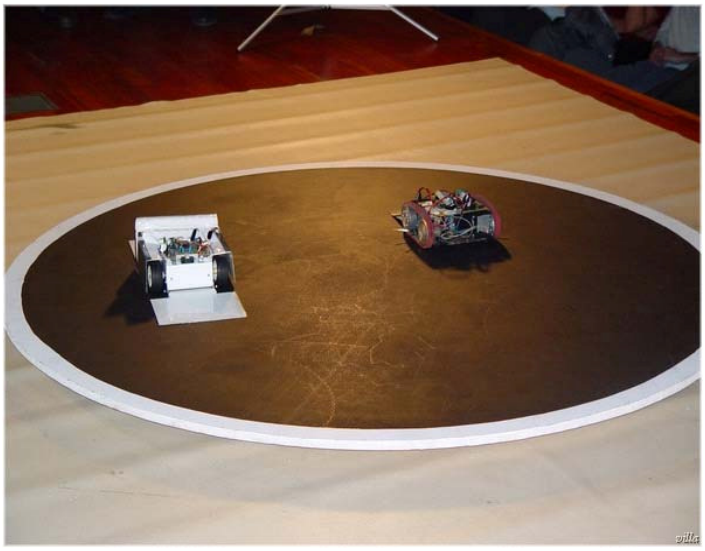


Figura 2.12. Robots luchadores de sumo en el tatami de competencia

2.7.1. Matriz PAMA del robot luchador de sumo.

Percepciones

El sistema de percepción de estos agentes se basa en la detección del final de la zona de juego, y en la detección del adversario. La zona de juego se marca con verde, y su límite en blanco, por lo que al igual que en los agentes rastreadores, resulta fácil utilizar sensores de reflexión de infrarrojos para detectar límites.

Resulta más complicada la percepción del adversario, ya que no existe un sensor idóneo que sea capaz de detectar en todas las situaciones al adversario. El intento de medir la distancia a la que está el oponente suele resultar infructuoso, ya que no conocemos su trayectoria o ángulo de ataque. Empíricamente si observamos a los mejores robots de este tipo, suele ser suficiente detectar solamente el contacto físico con el adversario. También resultaría sumamente interesante percibir si en nuestro intento de empujarlo, estamos consiguiéndolo o no. Por ello sería muy importante

detectar esta condición. Si decidimos percibir si las ruedas giran o si el consumo es normal, no sabremos si el robot está patinando, por lo que en este caso se aconsejan sistemas con un funcionamiento similar a un ratón óptico que nos indique si efectivamente el robot se está desplazando o está patinando, y sobre qué eje lo hace.

Acciones

Los actuadores de un agente de sumo deben ser motores acoplados a una reductora, de forma que el par que desarrollen sea muy grande, resultando muy importante transmitir esa potencia al suelo para conseguir empujar al adversa- rio. Las acciones directas pueden ser simplemente las de moverse por el tatami y no salirse de él.

En la práctica, algunos robots de sumo utilizan unas planchas metálicas que se abren en el arranque y tienen el objetivo de subir encima al adversario para expulsarlo del tatami.

Metas

Las metas o comportamientos serán las de ser atacante o defenderse. Si se ataca, debemos percibir que es el robot el que empuja, ya que en caso contrario debería tomar una actitud de defensa. Efectivamente, un error muy común en robots de este tipo consiste en que los dos estén empujándose, sin lograr moverse del sitio aunque las ruedas de ambos giren, consiguiendo agotar la energía y resentir las mecánicas.

Ambiente

En cuanto al ambiente para este tipo de agentes, no es del todo correcto decir que es solamente el tatami, ya que es posible que el adversario suba encima de sus planchas o que una parte de nuestro agente se encuentre fuera del tatami pero no haya caído totalmente. Se debe distinguir estos tipos

especiales del ambiente y tratar de percibir en cual de ellos está el robot para tratar de realizar el mejor comportamiento sin equivocaciones.

Tipo de agente	Percepciones	Acciones	Metas	Ambiente
Sumo	• Píxel blarcos o negros • Contacto con oponente	• Aplicar potencia a los motores	• Expulsar al adversario	• Tatami circular

Figura 2.13.Matriz PAMA del robot luchador (sumo)

2.7.2. Agente a utilizar en el robot luchador (sumo).

En este tipo de robots podemos usar alguno de estos dos tipos de agentes ya que los dos, cumplen las expectativas necesarias para conseguir el objetivo

Agente reflejo de estado interno.

Se guarda un estado interno sobre la situación actual, para tomar el comportamiento adecuado de continuar un ataque o de realizar una evasión.

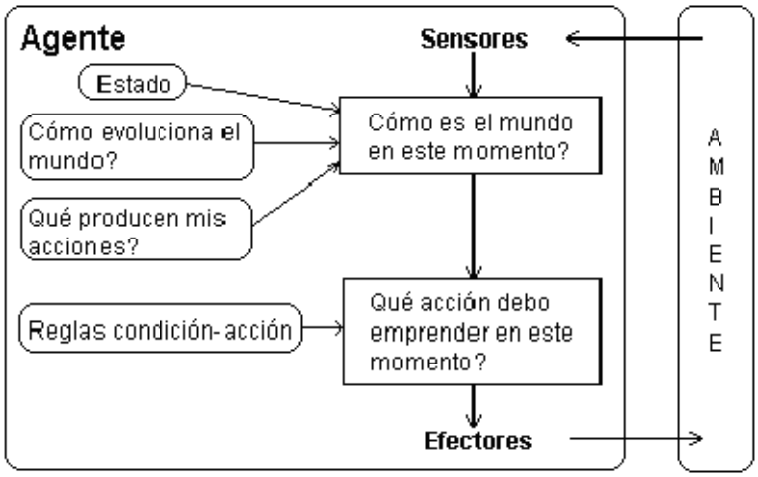


Figura 2.14. .Agente reflejo interno

Agentes basados en utilidad.

Este tipo de agente podría ser adaptativo al nivel de dificultad del enemigo, ya que puede evaluar el grado de acierto en los ataques, probando diversas técnicas de ataque hasta encontrar el que haga cumplir la meta de mejor forma. Sin embargo el inconveniente es que añade la necesidad de implementar una función que evalué el grado de acierto, no siendo siempre sencillo.

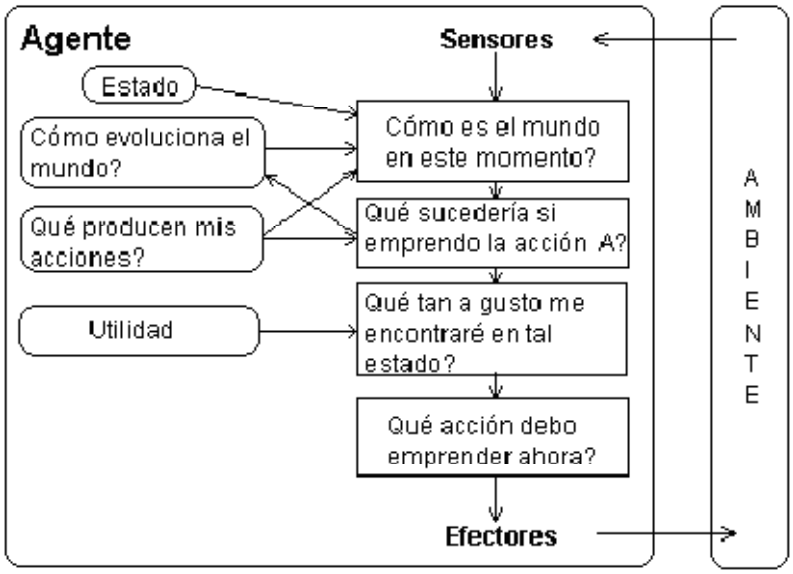


Figura 2.15. Agente basado en utilidad

2.8. Robot de competencias de velocidad.

Esta prueba valora la velocidad de un robot sobre una pista. Entra en juego sobre todo la velocidad máxima que un robot puede alcanzar siguiendo una pista, y cómo se resuelven los problemas de controlabilidad y autonomía. En este caso se trata de una pista con cuatro líneas negras paralelas durante todo el recorrido, con un ancho entre 1.5 y 2.5 cm. Estas calles tendrán un ancho entre 10 y 20 cm. El robot podrá guiarse con cualquiera de estas líneas, pero no podrá pisar nunca las exteriores ya que sería descalificado.

La organización medirá el tiempo que el robot tarda en dar un número determinado de vueltas. Hay una garantía de que las curvas serán de un radio de 40 cm como mínimo. En la pista puede haber rampas con una pendiente máxima del 20 %. Por último señalar que las dimensiones máximas del robot son de 20 × 30 cm, con una altura máxima de 15 cm.

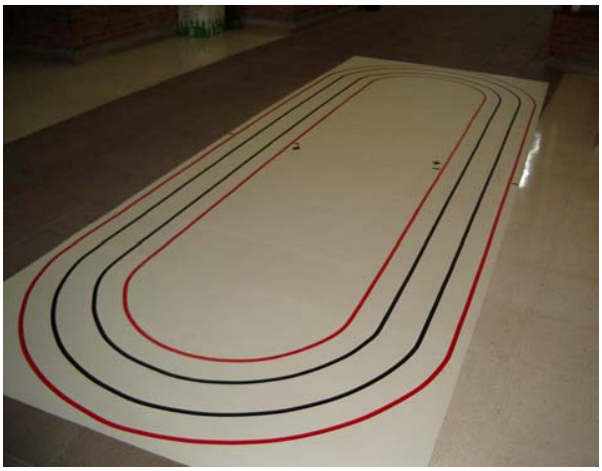


Figura 2.16. Pista de competencia de velocidad.

2.8. 1. Matriz PAMA del robot luchador de sumo.

Percepciones

Al igual que el agente rastreador, este agente debe disponer de una percepción similar. Sin embargo debido a que prima la velocidad, todo sistema sensorial que aporte precisión en la lectura, redundará en un beneficio importante para realizar un seguimiento fiel de la pista. Esto se conseguirá en función del número de sensores utilizados, de su disposición y su posición en el robot.

Acciones

Es deseable que su sistema de actuación sea mucho más rápido que en el agente rastreador, ya que si bien no se necesita un par tan grande por ser curvas mucho más abiertas, se requiere una velocidad mucho mayor para conseguir la meta. Las acciones que se deben tomar son las de realizar movimientos sobre la pista, de forma que sean fiel reflejo de los deseos de movimiento.

Metas

Su objetivo o meta simplemente es seguir la línea de la forma más fiel y rápida posible. Este agente al igual que el rastreador no es capaz de saber si ha conseguido su meta, ya que el tamaño de la pista es desconocido y no debe existir una meta de terminación.

Ambiente

El ambiente es similar a la pista de rastreadores: una pista negra sobre fondo blanco, con la suficiente diferencia de color como para no plantear problemas de detección. Sin embargo hay que filtrar pequeños errores de los sensores, que pueden hacer ver todo negro en un momento dado, así como manchas o suciedad de la pista.

Tipo de agente	Percepciones	Acciones	Metas	Ambiente
Velocista	• Píxel blancos o negros	• Aplicar potencia a los motores	• Seguir la pista de forma rápida	• Pista negra sobre blanco

Figura 2.17.Matriz PAMA del robot para competencias de velocidad

2.8.2. Agente a utilizar en el robot para competencias de velocidad.

Se puede deducir que el tipo de agente más adecuado es el de reflejo de estado interno, ya que ante una condición se ejecuta una acción pero en la

situación de perder la línea requiere actuar de otro modo, que podría ser torcer hacia el lado que perdió la pista. Por lo tanto resulta necesario un estado interno, pero no necesita evaluar la consecución de las metas, por lo que se justifica directamente la elección tomada.

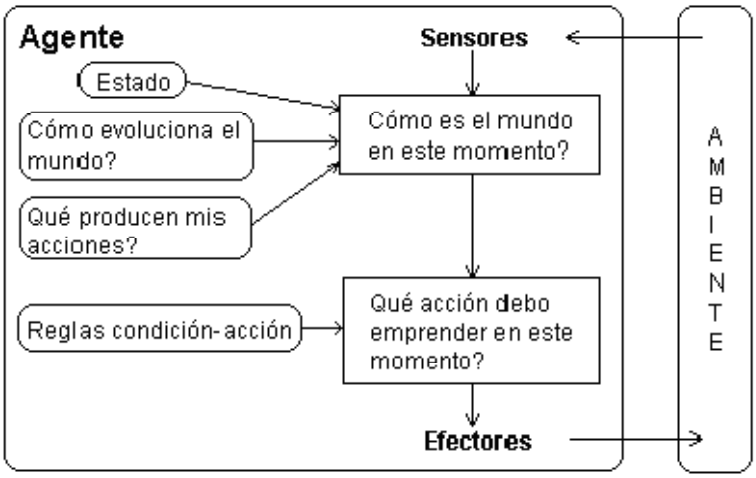


Figura 2.18. Agente de reflejo de estado interno.

3

ESTRUCTURA DE LA PROGRAMACIÓN DEL MICROCONTROLADOR

A continuación se van a describir los detalles técnicos más sobresalientes utilizados para la programación de los dos agentes estudiados, atendiendo al pre procesamiento de la entrada, máquina de estados y lógica difusa.

3.1. Restricciones de programación impuestas por el PIC.

En la programación en C para este PIC, hay que tener especial cuidado en cumplir una serie de reglas que a continuación se enumeran:

- Se disponen de 8 kbytes de memoria para código.

- Se dispone de 384 bytes de memoria RAM para uso de variables, aunque hay que tener en cuenta que el compilador asigna variables temporales, por lo que pueden ser menos.

- El tipo de variables utilizadas debe ser de tamaño byte solamente, con valores entre 0 y 255.

- Los parámetros de entrada y salida a funciones serán variables de tipo byte.

- Las matrices solamente pueden ser unidimensionales, con un tamaño máximo de 256 elementos.

- En todas las operaciones aritméticas se debe prestar especial atención en los desbordamientos. Se deben realizar funciones para realizar operaciones seguras.

- No se deben utilizar divisiones, ya que no están directamente soportadas por el PIC sino emulada por el compilador con múltiples instrucciones en ensamblador.

- Se deben probar partes pequeñas de código, debido a que no se puede depurar. Es importante ir creando bibliotecas de funciones utilizables ya probadas

3.2. Lecturas esporádicas.

El filtrado de lecturas esporádicas se hace necesario sobre todo en el caso de que los sensores del robot no se encuentren sobre la línea negra, y capte un valor esporádico de negro, producido por suciedad o brillos en la pista. En la figura 3.2 se muestra un ejemplo de como un ruido, en el momento de encontrarse fuera de la pista, podría hacer que el robot se saliese.

Para solucionar esto, antes de realizar otra acción distinta a la que estábamos haciendo nos aseguramos de que ya estamos viendo efectivamente la pista y no es una lectura esporádica. En la figura 3.1 se puede ver la representación temporal en la percepción de una lectura esporádica. El tiempo *te* para considerar que se trata de una lectura esporádica y no de la pista se ha establecido en 4 ms. En este tiempo, y según su velocidad, ambos recorrerán 1.6 mm. En la figura 3.2 aparece el algoritmo a seguir para evitar lecturas esporádicas.

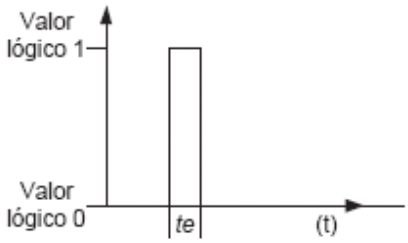


Figura 3.1. Representación temporal de una percepción

3.3. Ejemplo de estructura del software para los rastreadores.

El control se basa en un bucle principal en el que se realiza una lectura de la entrada, se pre procesa, se calcula el nuevo estado en función de la entrada y del estado actual, y por último se actúa.

La función *leeSensores* retorna la lectura del puerto que contiene los sensores de infrarrojos.

La función *procesaLectura* extrae el parámetro de la dirección. Para ello se han definido las siguientes funciones:

- char bitD(char n): Devuelve el primer bit puesto a uno por la derecha del parámetro de entrada n

- char bitI(char n): Devuelve el primer bit puesto a uno por la izquierda del parámetro de entrada n

- char ladoMarca(char D, char I, char antD,char antI): Retorna el lado por el que se ha localizado la marca. Esta información está en función de la lectura actual y la anterior.

La función *evento* retorna el código del evento que se ha producido en función de la entrada. En esta función se comprueban los eventos siguientes: sobre la línea, fuera de la línea, sobre un hueco y temporización terminada.

La función *calculaEstado* devuelve el nuevo estado a partir del estado actual y del evento producido realizando una consulta a la tabla de transiciones y estados. Esta tabla es bidimensional, aunque para adaptarla al PIC se hizo unidimensional

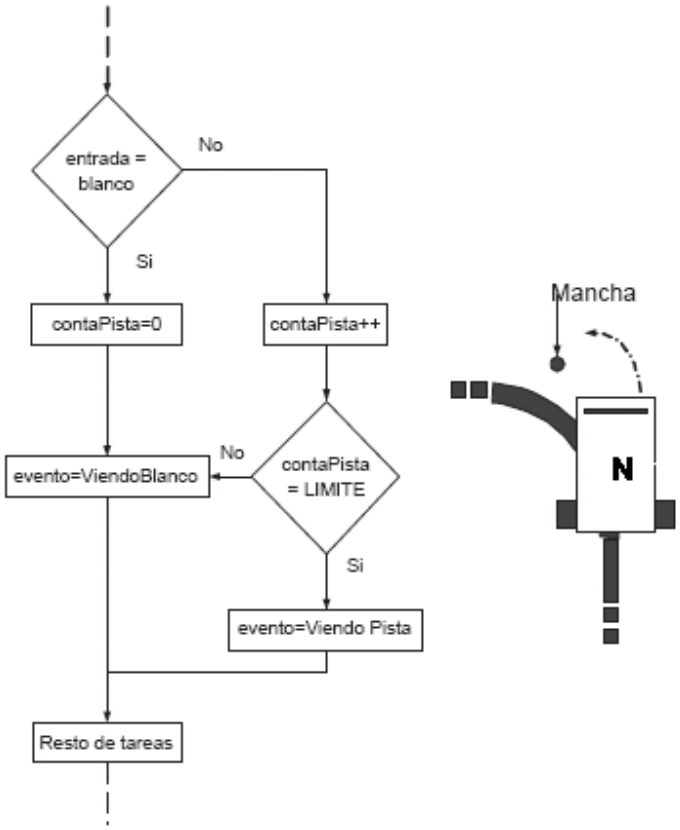


Figura 3.2. Filtro de lecturas espontaneas

La función *calculaEstado* devuelve el nuevo estado a partir del estado actual y del evento producido realizando una consulta a la tabla de transiciones y estados. Esta tabla es bidimensional, aunque para adaptarla al PIC se hizo unidimensional

La función *actuar*, es la que implementa el comportamiento del robot en función del estado actual y de la entrada. Como ya se dijo, las actuaciones posibles son control normal y control lateral. En el caso del robot, esta función incluye el sistema de control de velocidad.

```
MIENTRAS (1==1)
{
lectura=leeSensores();
lecturaTratada=procesaLectura(lectura);
estado=calculaEstado(estado,evento(lecturaTratada));
actuar(estado,lecturaTratada);
}
```

Figura 3.3. Protocolo del seguidor de líneas.

Debido a las limitaciones del PIC, solo es posible trabajar con variables tipo byte, por lo que las velocidades tendrán valores entre 0 y 255. El 255 representa a 1.5 m/s, el 128 equivale a parado, y el 0 equivale a -1.5 m/s.

Este controlador de velocidad se aplica en cada ciclo de control para los dos motores, por lo que su resultado final es bastante bueno a velocidades medias y altas.

3.4. Herramientas de programación.

El código se ha realizado en C debido a la íntima relación entre éste y el código ensamblador. Con la compilación se obtiene un código que resulta bastante optimizado, por lo que no se requiere su modificación antes de su ensamblado. La potencia proporcionada por el PIC a 4MHz, ha sido más que suficiente y no se ha necesitado afinar el código modificando el código ensamblador. Sin embargo se ha prestado atención en la optimización del código C de forma que las funciones repetitivas fuesen lo más rápidas posibles.

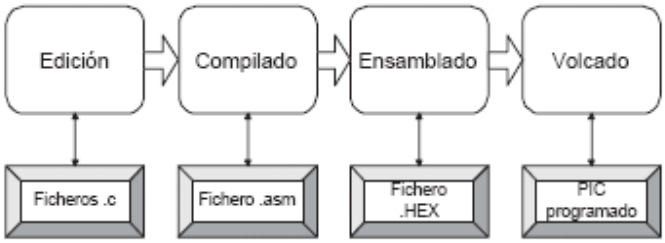


Figura 3.4. Etapas de programación del PIC

En la figura 3.4. se muestran las etapas necesarias en la programación del PIC. En la etapa de edición, se obtendrá el fichero en código C. En la compilación se obtiene un fichero con extensión .asm, el cual contiene los mnemónicos en ensamblador para el PIC. Este fichero se podría editar en este punto si fuese necesario. A continuación se ensambla el fichero .asm y se obtendrá un fichero con extensión .HEX. Este fichero contiene el código máquina que se va a volcar en la última etapa al PIC.

Se ha utilizado el entorno integrado C2C++ [C2C], el cual permite manejar fácilmente todas las herramientas correspondientes a las etapas del desarrollo, aunque el ensamblador y el programa de volcado no pertenecen a este entorno.

El ensamblador es el programa MPASM. Este programa es gratuito y lo proporciona directamente MICROCHIP el fabricante del PIC [MIC. La conexión del PIC al PC para el volcado se realiza con el programador SMT2.

3.5. Programa del robot.

```
/*****************************************************
Project : Robot Rastreador
 Author  : Mario Gonzalez
Chip type           : 16F84A
Clock frequency     : 40MHz
*****************************************************/

#include <16F84A.h>

#define Motor PORTB.2                 // Define un Motor
#define motor_max  180                // Valor para Controlar el Motor a 200 Hz
```

```
#define Servo PORTB.0
#define Servo PORTB.1
/#define Servo_max  510                  // Valor para controlar el Servo a 76 HZ


// Se definen los sensores

#define SenA PINB.3  // Izquierdo
#define SenB PINB.4  // Centro
#define SenC PINB.5  // Derecha



#define Uno 1
#define Cero 0
#define None 0
#define Si 1

#define Color  1 // 0= Blanco 1 = negro


int tiempo=0;
int pwmmotor=0;
int PWM=0;
int pwmservo=0;
int POS=0;

bit Sen_Izq=Cero;
bit Sen_Der=Cero;
bit Sen_Cen=Cero;

bit HuboCambio=None;

// Pin change interrupt service routine
interrupt [PCINT0] void pin_change_isr(void)
{
// Place your code here

   if (SenA==Color)
        Sen_Izq=Uno;
   else
     Sen_Izq=Cero;


        if (SenB==Color)
        Sen_Cen=Uno;
   else
```

```
        Sen_Cen=Cero;



  if (SenC==Color)
      Sen_Der=Uno;
   else
          Sen_Der=Cero;

  HuboCambio=Si;
}

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{

/* En esta rutina se controla la velocidad o el PWM de los motores de manera automatica.
*/



// Place your code here
tiempo++;                       // Variable para controlar Temporizacion
                               //Cuando Tiempo iguale a 37647 habra trasncurrido 1 Segundo
pwmmotor++;
pwmservo++;


// =====================================================
// Rutina de Servo
//=====================================================
if (pwmservo<POS>Servo_max)
   {
   pwmservo=0;
     Servo=1;
   }

  }



}

// Declare your global variables here

void main(void)
{
```

```
// Declare your local variables here

// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=0x80;
CLKPR=0x00;
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif

// Input/Output Ports initialization
// Port B initialization
// Func5=In Func4=In Func3=In Func2=Out Func1=Out Func0=Out
// State5=P State4=P State3=P State2=0 State1=0 State0=0
PORTB=0x38;
DDRB=0x07;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 9600.000 kHz
// Mode: Normal top=FFh
// OC0A output: Disconnected
// OC0B output: Disconnected
TCCR0A=0x00;
TCCR0B=0x01;
TCNT0=0x00;
OCR0A=0x00;
OCR0B=0x00;

// External Interrupt(s) initialization
// INT0: Off
// Interrupt on any change on pins PCINT0-5: On
GIMSK=0x20;
MCUCR=0x00;
PCMSK=0x38;
GIFR=0x20;

// Timer/Counter 0 Interrupt(s) initialization
TIMSK0=0x02;

// Analog Comparator initialization
// Analog Comparator: Off
ACSR=0x80;
ADCSRB=0x00;

// Global enable interrupts
#asm("sei")
```

```
tiempo=0;

// Apaga mabos motores para empezar
Motor=0;
Servo=0;

// Velocidad Minima para empezar
PWM=0;   //0 = Minima  180= Maxima
POS=0;


HuboCambio=None;
while (1)
     {
     // Place your code here


     while(HuboCambio==None);
     HuboCambio=None;

/*=================================================================
==================================
Debajo una estrategia simple de Seguidor de Linea. Este Sistema asume que esta utlizando 3
Sensores. La estrategia es de apagar, encender al máximo o a la mitad de velocidad los
motores.

Si esta activado solo el sensor den medio ambos motores se ponen a máxima capacidad.
Si esta activado el Sensor del medio y uno de los laterales entonces habra uno a máxima
capacidad y el otro  a mitad. Si se activa solo uno de los laterales, apagara un motor y el otro
esta a máxima capacidad.
El valor máximo se alcanza para 180 y el mínimo (apagado) es cero. Esto es a una
frecuencia de Reloj Interna de  40 MHZ
=================================================================
==================================*/



 if ((Sen_Cen==Uno)&&(Sen_Izq==Uno) )
         {
          POS=180;
          PWM=90;
         }
         else
```

```
        {
   if ((Sen_Cen==Uno)&&(Sen_Der==Uno) )
   {
   POS=90;
   PWM=180;
    }
          else

        {

    if (Sen_Cen==Uno)
    {
    POS=180;
    PWM=180;
    }
          else
          {
     if (Sen_Izq==Uno)
     {
     POS=180;
     PWM=0;
     }
     else
     {
                    if (Sen_Der==Uno)
                    {
                     POS=0;
                      PWM=180;
                    }

      }


          }
        }
      }
    };
}
```

3.5.1. Programa principal compilado (asm y hex)

```
*#cpu 16F84A
*
      ORG    $0000        $0800  CODE Starts here (Normally in ROM)
      LJMP   START


      ORG    $0003
      LJMP   SERVICE_EX0


      ORG    $000B
      LJMP   SERVICE_TIMER0_INTERRUPT.
?R0     EQU    0
?R1     EQU    ?R0+1
?R2     EQU    ?R0+2
?R3     EQU    ?R0+3
?R4     EQU    ?R0+4
?R5     EQU    ?R0+5
?R6     EQU    ?R0+6
?R7     EQU    ?R0+7
*
START   EQU     *
         MOV   SP,#?stk-1       Set up initial stack
      ORL  TMOD,#%00000001     set timer 0 to be counter 16 bit
```

```
    SETB    IE.7          $AF  EA
    SETB    IE.1          $A9  ET0 Enable timer 0 interrupt
    SETB    TCON.4        start timer 0

      LCALL         main            Execute program
    SJMP    *           JUMP HERE
SERVICE_TIMER0_INTERRUPT   EQU *
  PUSH ACC
  PUSH PSW
  MOV  TH0,#$FF     reload timer 0 for ms
  MOV  TL0,#$00
  INC  tick

  MOV  A,tick
  CJNE A,#100,RIGHT
  MOV  tick,#0

RIGHT
  CLR  C
  SUBB A,speedright
  JC   ON_RIGHT
  CLR  P1.0
  SJMP LEFT
```

```
ON_RIGHT
   SETB P1.0


LEFT
   MOV  A,tick
   CLR  C
   SUBB A,speedleft
   JC   ON_LEFT
   CLR  P1.1
   SJMP EXIT_I
ON_LEFT
   SETB P1.1
EXIT_I
   POP  PSW
   POP  ACC
   RETI
SERVICE_EX0 EQU *
   INC  cputick
   RETI
$SE:1
*#map1 Segment 1, initialized variables
$SE:2
```

```
*#map2 Segment 2, internal "register" variables
        ORG   $0008             Internal ram ALWAYS starts here

tick        DS  1
speedright  DS  1
speedleft   DS  1
cputick     DS  1
```

El archivo generado en hexadecimal es el siguiente:

```
:0300000002000EED
:03000300020055A3
:20000B00020024758 10F438901D2AFD2A9D28C12005880FE12001B80FBC0E0
C0D0758CFFC3
:20002B00758A000508E508B46403750800C395094004C2908002D290E508C39
50A4004C2F4
:20004B00918002D291D0D0D0E032050B327440F59074FFF5B07450F50C741E
F50D7400F548
:20006B000E7432F50F120080E5B0440FF5B01200BC020073220581058174017
8FD12022609
:20008B00F678FD120226E670030200B7E59008F6E65440120251 7B001202481
2025A45F0D2
```

:2000AB0070030200B4740018F602008C1581158122058 1E5B0540FF5B078FE120226F6E6FF

:2000CB0054011202517B00120248120 25A45F070030200E91201A97401F50E02018478FE52

:2000EB00120226E654081202517B0012024812025A45F0700302010D1201C87402F50E02C1

:20010B00018478FE120226E61202517B0912024812025A45F0700302013CE50C75F000C009

:20012B00E0C0F012018715811581740 0F50E02018478FE120226E61202517B0B1202481271

:20014B00025A45F0700CE61202517B0D120248120 25A45F0600CE50E75F0007B007C0012E8

:20016B00025A45F07003020184E50D75F000C0E0C0F012018715811581158127464F59067

:20018B0078FB120226E6240AF509E6F50AE50F75F000C0E0C0F012020915811581227468C0

:2001AB00F590E50D2405F509E50DF50AE50F75F000C0E0C0F0120209158115 81227454F5D4

:2001CB0090E50D2405F509E50DF50AE50F75F000C0E0C0F012020915811581227458F59015

:2001EB0078FB120226E6F509E62405F50AE50F75F000C0E0C0F012020915811581278FBCE

:20020B00120226860308860 4BB0004BC00012279E5A3D9FD1BBBFFF01C80F4C82581C8225C

:20022B00C92581C97A0022D083D082CF2581F581CFC082C08322CF2581F581CF227C00CBB6

:20024B0030E7011CCB2275F00030E70215F02212026D6009E4F5F02212026D60F7E4F5F057

:20026B000422C5F0C39C7003E5F09B22FBE493CB22FCE493FB740193CC22FAE493F9740192

:03028B0093CA22F1

:00000001FF

4

AGENTES INTELIGENTES APLICADOS EN EL DISEÑO DEL ROBOT

En este capitulo se analizara la inteligencia adoptada en el robot desde su comienzo, hasta una solución adecuada a cada uno de ellos. Fundamentada en los análisis y las experiencias adquiridas basado en los agentes que integran la inteligencia artificial.

4.1. Descripción del software del agente.

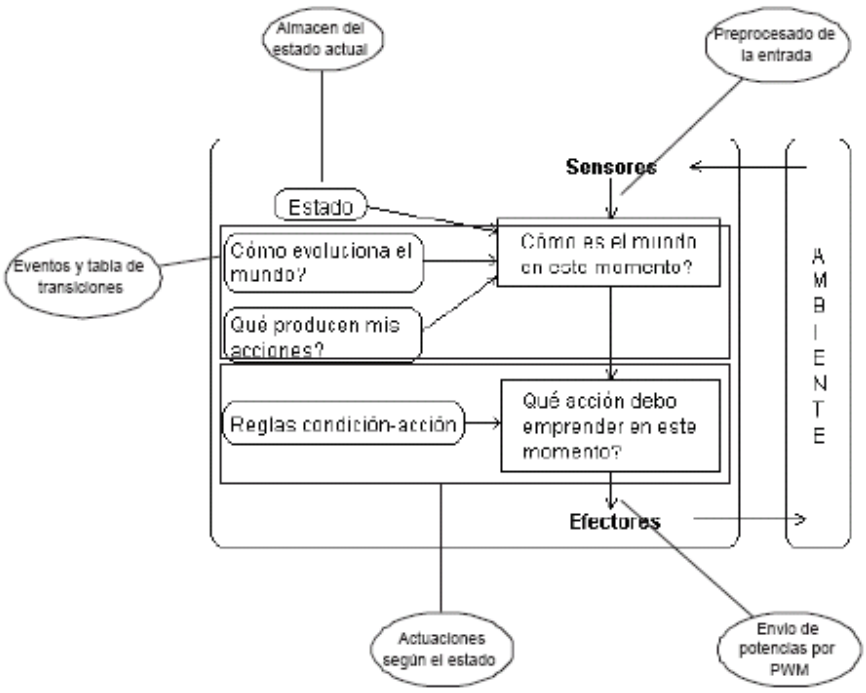


Figura 4.1. Software del agente con estado interno.

El agente propuesto para los robots rastreadores, es el de estado interno, siguiendo una línea de ejecución dada por el ambiente, donde el agente se pregunta ¿Cómo es el mundo en este momento?, para responder a esta pregunta es necesario que se conozca el ambiente, pero la entrada realmente no lo conoce tan solo con el diseño electromecánico, para ello el software tiene que conocer esta entrada del ambiente para poder procesarlo y decidir si es un hueco, una línea o una curva. Las preguntas ¿cómo evoluciona el mundo? y ¿qué producen mis acciones?, se pueden reflejar como una serie de estados en los que el robot toma uno u otro comportamiento. También se deben reflejar las transiciones entre los distintos estados, es decir, las condiciones que llevan a pasar de un estado a otro. De forma implícita deducimos que también debe existir un almacén del estado actual de funcionamiento.

De todo esto se puede recalcar que el agente software es un gran reto ya que tendrá que identificar las distintas formas de evolución de los movimientos del robot, dado que los estados que adopte serán muy importantes para el resultado que se desea, para ello tendrá que asignarle decisiones que se deberán tomar conforme a la entrada de los sensores.

4.2. Maquina de estados.

Se denomina ***máquina de estados*** a un modelo de comportamiento de un sistema con entradas y salidas, en donde las salidas dependen no sólo de las señales de entradas actuales sino también de las anteriores. En este modelo se define un conjunto de estados que sirve de intermediario en esta relación de entradas y salidas, haciendo que el historial de señales de entrada determine, para cada instante, un estado para la máquina, de forma tal que la salida depende únicamente del estado y las entradas actuales.

Las máquinas de estados se pueden clasificar en *aceptoras* o *transductoras*:

- **Aceptoras** (también llamadas **reconocedoras**): Son aquellas en donde la salida es binaria (si/no), depende únicamente del estado y existe un estado inicial. Puede decirse, entonces, que cuando la máquina produce una salida "positiva" (es decir, un "si"), es porque ha "reconocido" o "aceptado" la secuencia de entrada. En las máquinas de estados aceptoras, los estados con salida "positiva" se denominan *estados finales*.

- **Transductoras**: convierten una secuencia de señales de entrada en una secuencia de salida, pudiendo ésta ser binaria o más compleja, depender de la entrada actual (no sólo del estado) y pudiendo también prescindirse de un estado inicial.

En nuestro caso se utilizara una maquina de estados transductora, representa los distintos tipos de comportamientos que tiene el robot, según los eventos ocurridos en su mundo (la pista) para seguir un camino de forma correcta. En general una máquina de estados se compone de estados, eventos y acciones.

- Un estado representa una situación
- Un evento es un suceso que plantea el cambio de un estado a otro
- Una acción es un tipo de comportamiento

Para ser una máquina correcta debe cumplir las condiciones siguientes:

- Ser Finito. Debe de tener un número de estados representable
- Determinista. Debe cumplir las características de no tener estados ambiguos, y ser completo, es decir, que represente todos los estados o situaciones posibles

Una máquina de estados tiene bien definidos los eventos que reconoce y los que descarta, de forma que se construye basándose en un alfabeto. También debe incluir estado inicial o de arranque y algún estado final. Estos pueden coincidir

. Si cumple todas estas características se conoce como Autómata Finito Determinista (AFD). Si no se definen estados finales se conocen como semiautómatas finitos. Formalmente un AFD es una estructura de la forma $AFD = (Q, Ent, TR, qi, F)$, donde:

- Q Conjunto de estados

- *Ent* Alfabeto de entrada
- *TR* Función de transición del tipo $Q \times Ent \rightarrow Q$
- $Qi \in Q$ Estado inicial
- $F \subset Q$ Estados finales

En el caso que nos ocupa, se han implementado varias máquinas de estados según las pruebas y el robot al que van dirigidas.

En base a las dificultades de las pistas diseñadas en las competencias de robots seguidores de línea o rastreadores, es necesario aplicar cuatro estados para el buen funcionamiento del robot.

1. Seguir línea
2. Detectar marca
3. Tomar bifurcación
4. Fuera de pista

El estado inicial o de arranque será el de seguir línea (1). La percepción que tiene del mundo es a través de 3 sensores de infrarrojos que finalmente se traduce en 3 valores binarios. Por lo tanto su mundo solamente tiene 24 valores posibles. Lógicamente no podemos trabajar a este nivel tan bajo, sino que debemos conceptualizar lo que está viendo. De esta forma definimos funciones que con la lectura de esos 3 bits indiquen si se encuentra sobre una línea o estamos sobre una marca

En el momento que se ha decidido el lado de la bifurcación a seguir, se inicia un temporizador con un tiempo suficiente para asegurar que ya ha pasado Según la velocidad del robot se ha calculado este tiempo en 1 s, suficiente para sobrepasar la bifurcación.

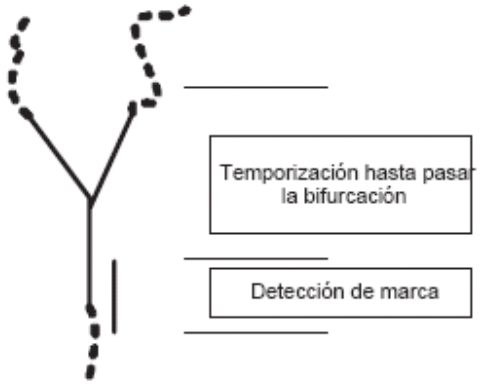


Figura 4.2. Recorrido discontinua del rastreador.

Si resumimos los eventos comentados tendremos:

A Línea normal

B Hueco esporádico

C Hueco seguro

D Fin de temporizador de bifurcación

E Viendo blanco (salida de pista)

Por último debemos obtener las acciones a realizar en cada estado.

En el estado 1: Control normal

En el estado 2: Control normal

En el estado 3: Control lateral

En el estado 4: Girar hacia el lado por el que salió

Finalmente queda pendiente definir las transiciones de estados en función de los eventos.

Evento\Estado	1	2	3	4
A	1	1	3	1
B	2	2	3	1
C	-	3	-	1
D	-	-	1	-
E	4	4	4	4

Figura 4.3. Tabla de transiciones

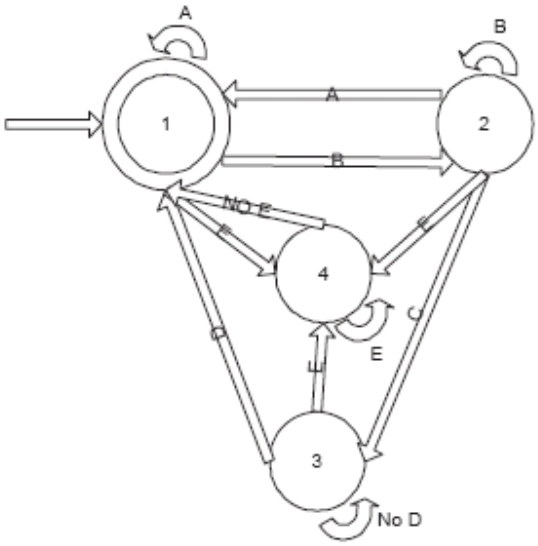


Figura 4.4. Maquina de estados

El estado 4 sucede cuando pierde la línea. En este caso el robot seguirá torciendo hacia el lado por el que salió hasta que la vuelva a detectar la pista. A este estado se entra desde cualquiera de los otros tres

4.3. Ajustando la lógica borrosa.

La lógica borrosa es una solución muy adecuada para este tipo de sistemas, ya que se consigue una suavidad de funcionamiento. Además se han realizado pruebas previas y simulaciones, de forma que se concluyó que resultaba la solución más interesante.

La forma de tratamiento de la entrada reduce considerablemente el número de casos a tratar. Así se plantea, como una solución idónea, tener pre calculados todos los valores de salida en función de la entrada, sin tener la necesidad de trabajar con lógica borrosa a nivel de PIC. De esta forma evitamos realizar para cada ciclo de control las etapas de *fuzzificacion* y *defuzzificacion*, que por otro lado serían bastante costosas para el procesador utilizado.

Figura 4.5. Estado de la lógica borrosa

Las reglas borrosas en el robot seguidor de líneas (o rastreador) solo se aplican cuando el camino es demasiado difícil para su propio aprendizaje.

Como es lógico, las reglas son simétricas para ambos motores, ya que se trata de dos motores en disposición opuesta y hay que realizar inversión de signo para que el robot avance. También se puede observar como siempre hay un motor que está girando a toda velocidad, con un valor *ph* (positivo alto). Esto significa que el robot va a su máxima velocidad mecánica en todo momento. Es una situación límite que ha probado que incluso en estos casos la lógica borrosa proporciona buenos

resultados. Podemos afirmar que las reglas aplicadas explotan prácticamente todo el rendimiento mecánico del robot.

motor izquierdo.

Entorno	Dirección		
	Negativa	Cero	Positiva
Fácil	ph	ph	z
Regular	ph	ph	nm
Difícil	ph	ph	nh

motor derecho

Entorno	Dirección		
	Negativa	Cero	Positiva
Fácil	z	ph	ph
Regular	nm	ph	ph
Difícil	nh	ph	ph

ph=positivo alto, pm=positivo medio, z=cero, nm=negativo medio, nh=negativo alto

Figura 4.6. Reglas para el robot

5

DISEÑO DE UN ROBOT SEGUIDOR DE LÍNEA BASADO EN LOS AGENTES INTELIGENTES

Anteriormente hemos visto como desarrollar la matriz PAMA del agente correspondiente a cada caso, en este capítulo se va a hacer una descripción del diseño electrónico de un robot seguidor de trayectorias, basado en dichos agentes inteligentes.

No pretende ser un manual de cómo construir un robot, pero sí una base de conocimiento de la que partir para realizar prototipos basados en las nuevas técnicas de inteligencia artificial para robots móviles.

5.1. Sistemas físicos del agente.

Como ya se justificó en el capítulo anterior, se ha elegido el agente reflejo con estado interno para las pruebas de rastreadores. En este apartado se va a analizar cómo partiendo del concepto de este agente, podemos llegar a concluir el hardware utilizado en la construcción de robots seguidores.

En la figura 5.1. Se puede ver el agente y los elementos hardware que lo componen. Podemos ver como los sensores, proporcionan la percepción del ambiente al robot. La percepción es el mecanismo de introducir de forma electrónica las lecturas de la línea dentro del sistema.

Estos sensores deben ser capaces de distinguir de forma inequívoca los dos colores posibles de la pista para las pruebas de rastreadores: blanco y negro. Sin embargo su tolerancia debe ser adecuada para no confundir los casos imperfectos. Los sensores más comunes son los de infrarrojos, ya que permiten modificar las tolerancias en la forma que necesitemos.

Los actuadores proporcionan una interacción con el ambiente. En este caso se encargan de realizar el movimiento siguiendo la pista. Los motores deben intentar reproducir fielmente los movimientos que resulten de los cálculos. Normalmente estos motores tendrán cajas reductoras para proporcionar la velocidad y/o potencia adecuadas. También debemos considerar las ruedas como una parte muy importante del robot. Deben tener unas dimensiones adecuadas para mantener una relación de velocidad/potencias adecuada, así como un agarre óptimo.

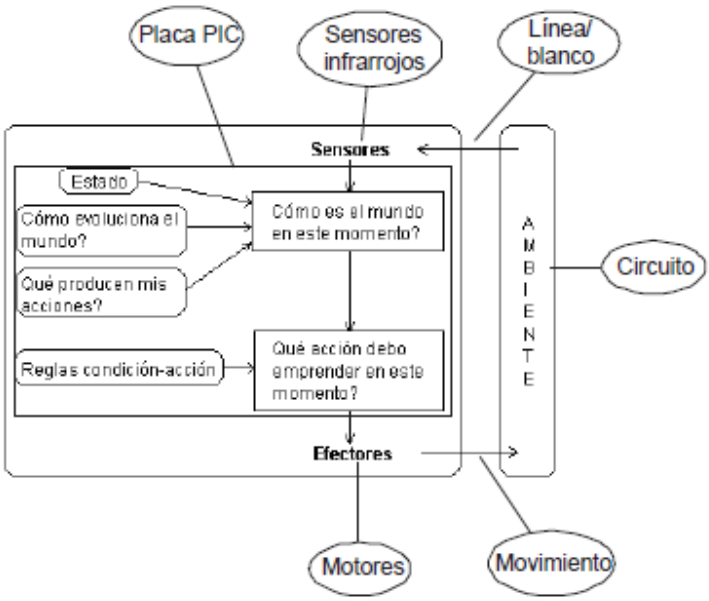


Figura 5.1. Agente con estado interno desde el punto de vista del hardware

Dentro del sistema de actuación, y en los sistemas que se estime oportuno, se incluirán los mecanismos hardware para el control de velocidad que garanticen la velocidad correcta.

Los sistemas de actuación representan el 95% del consumo total de energía de los robots. Este consumo será desconocido, dado que su entorno también lo es. No se conoce de antemano el esfuerzo que tendrán que realizar sus motores.

El resto de elementos del agente son partes del software que se ejecutará normalmente sobre una arquitectura sencilla, robusta y económica. Típicamente son sistemas microcontroladores.

Bajo el concepto de agente, una vez distinguidos los distintos elementos que se necesitan para su realización, se detallarán las soluciones hardware adoptadas en el robot rastreador a construir.

5.2. Robot seguidor de trayectorias.

El robot que se ha construido con materiales muy sencillos y de bajo costo. Sin embargo su simpleza no ha sido sencilla de obtener pues se han utilizado distintas configuraciones físicas hasta llegar al resultado final, que es la aplicación de las matrices PAMA.

5.2.1. Base del robot.

Tomando en cuenta que utilizaremos una configuración de de triciclo pará la base usaremos un MDF (Medium Density Fibreboard) son las siglas en inglés de "tableros de fibra de madera de media densidad". el cual tiene las siguientes características:

- Color uniforme.
- Tamaño de fibra homogéneo en todo el espesor.
- Perfil de densidad equilibrado.
- Superficie muy suave.
- Baja abrasividad (menor consumo de herramientas).
- Baja absorción (menor consumo de pintura).
- Excelente calibración de espesores.
- Grandes dimensiones (mejor aprovechamiento del material).
- Superiores propiedades físico-mecánicas.
- Perfil de densidad equilibrado.

El MDF es un compuesto de fibra y resinas sintéticas de calidad alta, el material es de bajo costo.

La estructura tiene una dimensión de 20 x 24 cm con un espesor de 5 mm ya que no tendrá un gran peso sobre de ella, en la siguiente figura se muestra el diseño apropiado para nuestro robot.

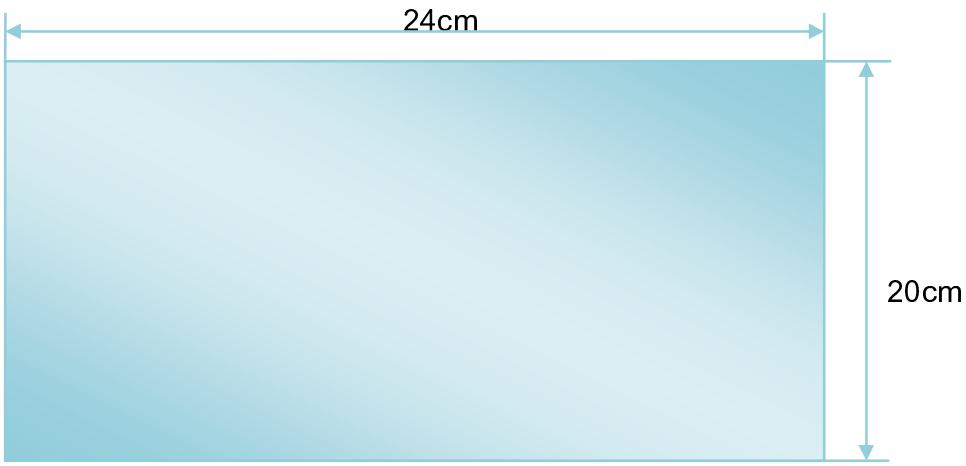


Figura 5.2. Dimensiones de la estructura.

El robot tendrá una base para los sensores de 6 x 4 cm, sujetada a la base de la rueda loca (**CASTER** WHEEL), dicha base tendrá un tamaño de 4.5 cm (de altura con un diámetro de 1.5 cm de la rueda loca), donde la base de los sensores se fijara a una distancia de 4 cm con respecto al cuerpo del robot para mantenerlos a una distancia de 5mm de la pista con una separación de 1.5 cm entre ellos ya que es el espesor de la línea marcada.

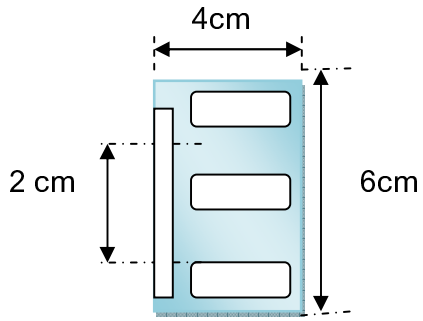


Figura 5.3. Dimensiones del soporte del sensor.

Teniendo las dos estructuras, es necesario ensamblarlas ya que el circuito este definido, para poder calcular la distancia a la que hay que colocar los sensores, y la dimensión de los mismos ya que aunque se ocupen de un solo tipo, el sensor no esta estandarizado y tienen una variación de 2 a 3 cm de largo; es necesario fijar la base de los sensores aproximadamente un centímetro fuera de la base principal.

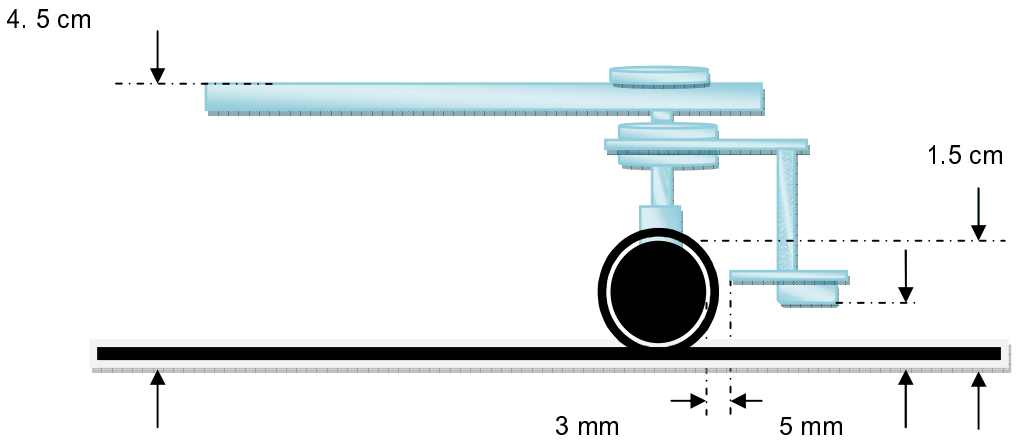
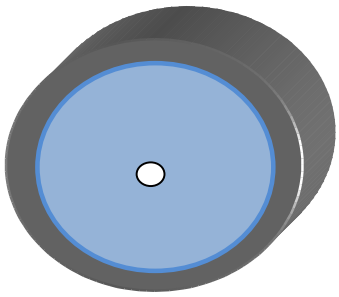


Figura 5.4. Vista lateral de la estructura de MDF con la caster whell .

Tomando en cuenta estas medidas, es necesario tener 2 ruedas de MDF de 8 cm de diámetro con 1 cm de espesor, para las ruedas principales del rastreador. Estas ruedas serán de de dicho material, y revestidas en su exterior, por una tira plástica de 5 mm de espesor, para un mejor desplazamiento sobre superficies planas.



.

Figura 5.5. Diseño de la rueda de MDF.

Al centro de las ruedas, se hará una perforación para fijarlas al motor por medio de un tornillo, (Los motores ya cuentan con el acoplamiento incluido).

Los motores se tendrán que fijar por debajo de la base principal ya que así, el peso sera repartido en la estructura de forma que el centro de gravedad sea mas bajo. Las ruedas se fijaran al acoplamiento plástico de los motores, por medio de un tornillo.

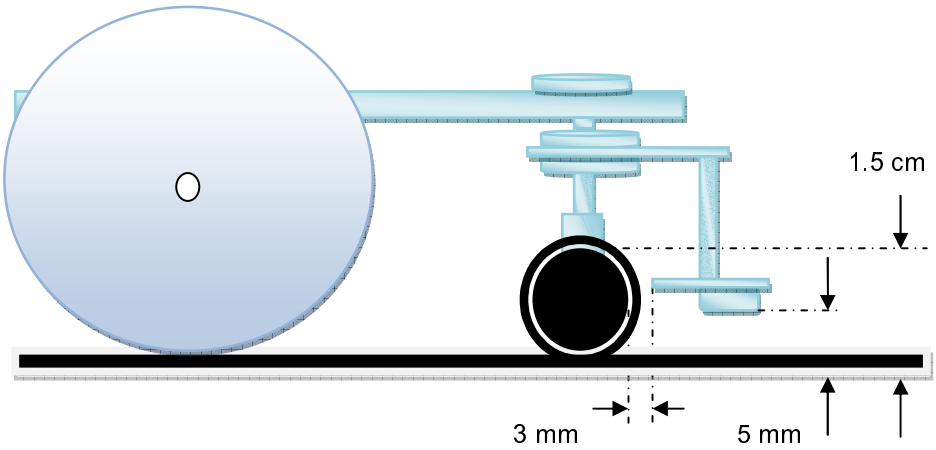


Figura 5.6. Estructura del robot.

.

5.2.2. Motores.

El sistema de motores se compone de dos servomotores modificados para giro libre, los servomotores disponen de unos topes físicos que hacen imposible que puedan girar más de unos 180º grados También originalmente tiene un potenciómetro que gira a la vez que el eje final y sirve como realimentación de la electrónica integrada en el servo para saber cual es la posición en la que se encuentra, al modificados se tienen que eliminar estos elementos, es decir se elimina el tope físico y se deja constante el potenciómetro.

Al hacer esto los servomotores giran 360º grados, los cuales son utilizados de esta manera debido al torque y la poca inercia que presentan facilitando así que el robot seguidor de línea arranque o pare rápidamente.

Estos están acoplados a unas ruedas de plástico de 14cm de diámetro. Un servomotor es una solución cerrada que se compone de un motor, una reductora y un driver de potencia que permite controlar la velocidad de giro del motor con entradas TTL, típicamente la salida del microcontrolador.

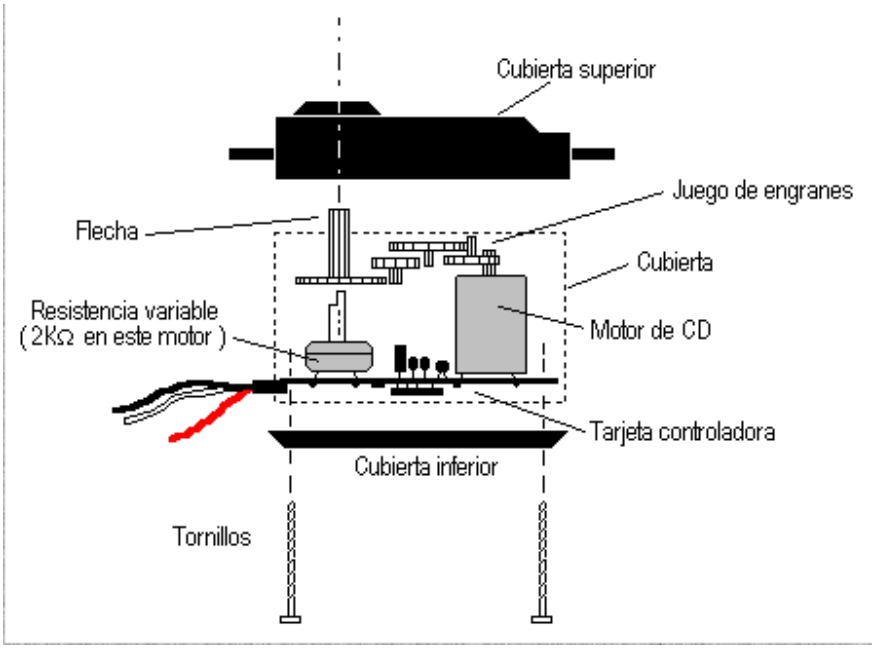


Figura 5.7. Partes de un servomotor

Los motores están definidos por motores RE-280 de la marca Mabuchi Motor. Su eficiencia es máxima alrededor del punto de par 20 g.cm, en el que el RE-140 se paraba. Aquí, el motor obtiene unas 7500 rpm a 6 V.

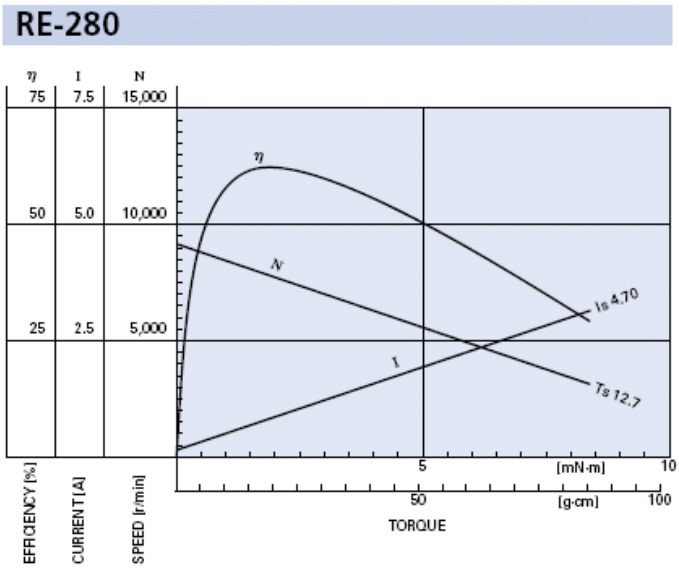


Figura 5.8. Características del motor RE-280

Sin embargo debemos estar prevenidos, ya que tendrá más peso y deberá soportar un poco más de esfuerzo.

Según la gráfica como su consumo en un momento dado puede aumentar hasta 4.7 A. No se trata de situaciones normales, pues su esfuerzo estará entorno a los valores antes comentados y su consumo normal será sobre 1 A, pero debemos tener en cuenta que la pista no es perfecta y que habrá picos de esfuerzo y consumo muy altos.

Lla velocidad óptima del motor RE-280 en carga son 7500 rpm, equivalentes a 125 rps (revoluciones por segundo). Eso equivale que la relación entre la rueda y el motor debe ser de 125 / 7,345 = 17,1. Sin embargo este motor se compra con un kit de engranajes de doble corona con 30 y 12 dientes. La relación más parecida a la calculada anteriormente la conseguimos con tres engranajes: $(30 / 12)_3$ = 15,625

La zona nominal de trabajo estimada, estará entre 5 y 30 g.cmy con esta carga, girará entre 9000 rpm y 7000 rpm aproximadamente.

Para la relación que disponemos, esto equivale aproximadamente a velocidades entre 2 m/s y 1.55 m/s. Para mantener la velocidad máxima en 1.5 m/s, y dada la relación disponible, el motor irá a menos revoluciones de las inicialmente calculadas, aproximadamente a unas 6900 rpm (115 rps). Aún así, si observamos su característica vemos como está en una zona con un rendimiento muy alto (60% aproximadamente) que hace perfectamente posible la utilización de este motor.

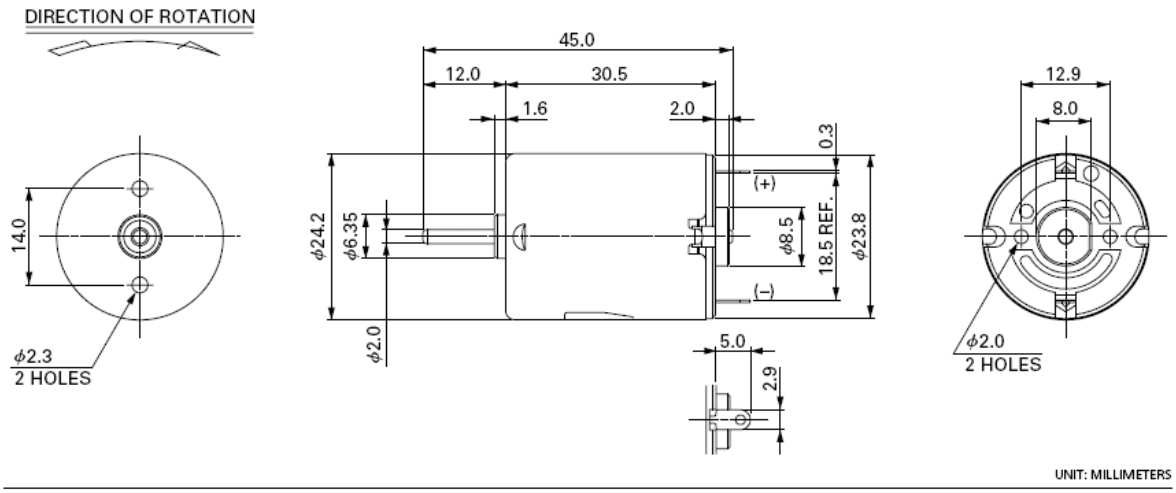


Figura 5.9. Dimensiones del motor RE-280

5.3. Composición de las partes del sistema electrónico.

A continuación se detallara las sub-faces que integran el circuito del robot para al final integrar el diagrama correspondiente.

5.3.1. Sistema de traslación.

Para la elaboración de esta parte se eligió el puente H, LMD18200, de National Semiconductors; este circuito integrado consta de once pines y ofrece las siguientes ventajas:

- tamaño reducido
- control de dirección
- velocidad del motos
- freno del motor

Tiene tres terminales de entrada TTL. La alimentación de este circuito es de 12 a 55 volts y proporciona una corriente mínima de 0.5 A y una máxima de 3 A.

5.3.2.- Los sensores del robot.

El sistema sensorial del robot se compone de 3 sensores HOA6389 en disposición lineal al sentido de la marcha.

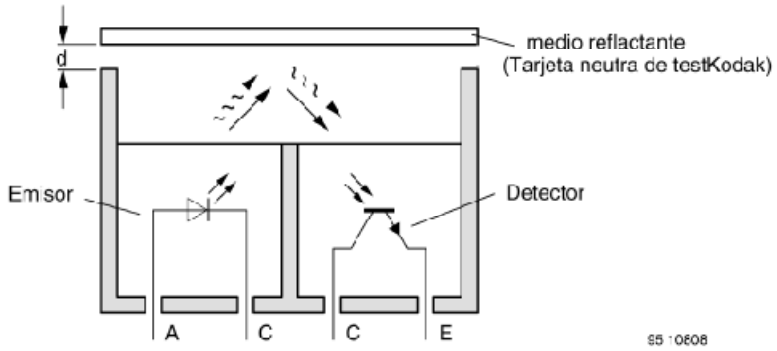


Figura 5.10. Tren de pulsos para el control de servomotores

El HOA6389 es un sensor de infrarrojos de corto alcance basado en un emisor de luz y un receptor, ambos apuntando en la misma dirección, y cuyo funcionamiento se basa en la capacidad de reflexión del objeto, y la detección del rayo reflectado

por el receptor. El HOA6389 tiene cuatro pines de conexión. Dos de ellos se corresponden con el ánodo y cátodo del emisor, y las otras dos se corresponden con el colector y el emisor del receptor. Los valores de las resistencias son típicamente 10K ohms para el receptor y 220 ohms para el emisor.

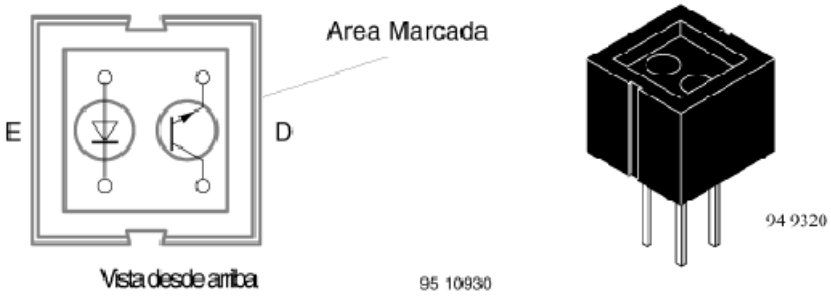


Figura 5.11. Sensor HOA6389

Los sensores HOA6389 tienen una salida analógica. Dan valores entre 0.2 V y 4 V si reflejan el blanco o negro respectivamente. Sin embargo también proporcionan valores intermedios de voltaje en otros niveles de grises.

El montaje completo de cada uno de estos sensores, se compone de un HOA6389, y dos resistencias, ambas utilizadas para limitar la intensidad máxima que circula por el LED y por el fototransistor. El esquema final de cada sensor se muestra en la figura 5.11. Cada una de las salidas de este esquema se conecta a una entrada del PIC (pines 28 a 21). Comúnmente este sensor tiene una resistencia de 1k en el emisor y 10k en el receptor, y trabajan a 5 v para su óptimo funcionamiento, las resistencias 6 y 7 protegen al emisor y receptor de una tención demasiado elevada.

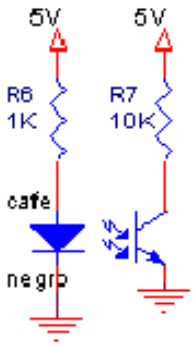


Figura 5.12. Diagrama del sensor

Es necesario conectar la salida del receptor a un amplificador operacional, este se encarga de enviar una señal útil de cinco volts al microcontrolador, cuando el sensor esté activado. Este amplificador operacional se encuentra dentro de un circuito integrado denominado LM324, el cual contiene cuatro amplificadores operacionales.

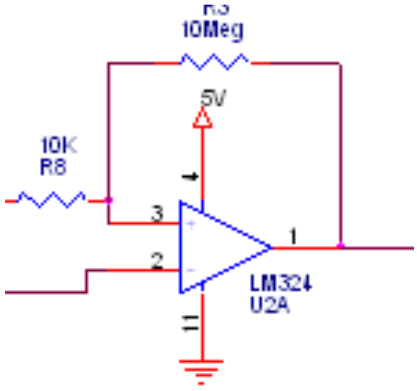


Figura 5.13. Conexión del LM324

Por otra parte el receptor, que es un fototransistor, se conectó a tierra el emisor y el colector a una resistencia de 10KW que va a 5V, para conseguir que el transistor funcione como inversor.

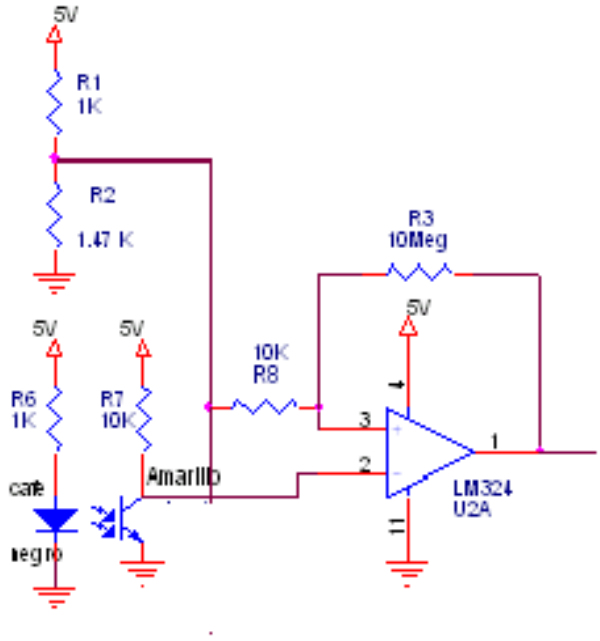


Figura 5.14. Conexión del sensor y el amplificador operacional LM324

5.3.3. ALIMENTACION.

Otra parte muy importante de un robot móvil es la alimentación, dos voltajes de alimentación, se emplean: el primero de 12 volts que alimenta la etapa de potencia y el segundo de 5 voltios el sistema mínimo y los sensores. Los 12 Voltios se consiguen con ocho pilas y los cinco voltios se obtienen de un 7805 conectado al voltaje anterior como entrada

Se propone utilizar baterías de NiMh de tamaño AA, con una capacidad de 1800 mA. Si medimos el consumo en vacío del robot da valores en torno a 2 A. En carga crece de forma muy importante, llegando a veces a medirse casi 5 A, límite máximo de descarga para este tipo de baterías. Sin duda alguna, unas baterías con mayor límite de descarga, potenciarían bastante a al robot, ya que actualmente el cuello de botella se encuentra en este punto.

Posiblemente las más adecuadas sean las de modelismo tipo Sub-C, con una capacidad de descarga muchísimo mayor que las de tamaño AA que nosotros utilizamos. Con este cambio las baterías podrían dar toda la potencia demandada por los motores (hasta 4,7 + 4,7 = 9,4 A), y no estar limitado a los 5 A de las actuales.

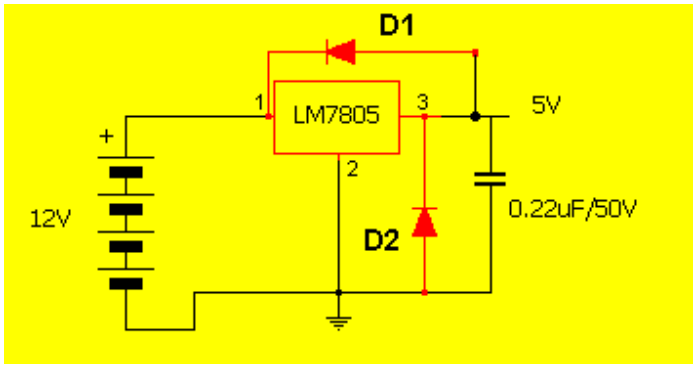


Figura 5.15. Fuente de alimentación.

5.3.4. Etapa de potencia.

El control de los motores se realiza con los módulos de PWM de los microcontroladores, los cuales tienen la característica de poder programar su ancho de pulso y frecuencia. Estas señales de PWM son reforzadas y aisladas para poder ser empleadas por los drivers de los motores en la tarjeta de potencia.

Para la excitación de los motores del robot se dispone del diseño de un driver de potencia. Para la elaboración de esta parte se eligió el puente H, LMD18200, de National Semiconductors; este circuito integrado consta de once pines y ofrece las siguientes ventajas: tamaño reducido, control de dirección, velocidad y freno del motor con tres terminales de entrada TTL. La alimentación de este circuito es de 12 a 55 voltios y proporciona una corriente mínima de 0.5 A y una máxima de 3 A.

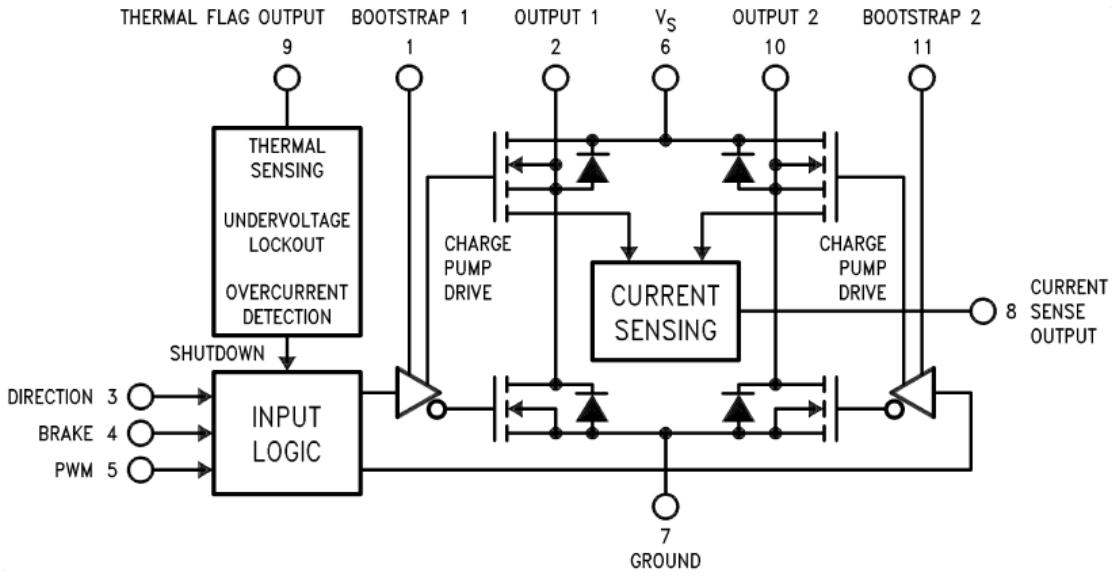


Figura 5.17. Diagrama de bloque del C.I. LMD18200

Este circuito integrado cuenta con las características de controlar motores mediante señales de onda de hasta una frecuencia de 1 MHz, determinar el sentido de rotación mediante una señal lógica aplicada al circuito, de contar con un freno controlado por un voltaje lógico, y de contar con un sistema de sensado corriente y auto apagado por sobrecalentamiento para evitar cortocircuitos y sobrecargas en los motores.

El movimiento principal de los motores, es desplazarse hacia adelante, esto es teniendo una entrada un valor alto, haciendo que los 2 motores giren con el mismo sentido. Los giros a la izquierda y a la derecha se logran haciendo que las ruedas se muevan en direcciones opuestas con lo cual el Robot vira sobre su centro de giro. Para la corrección de trayectoria hacia derecha o izquierda se apaga uno de los motores y se vuelve a prender cuando se ha corregido la trayectoria.

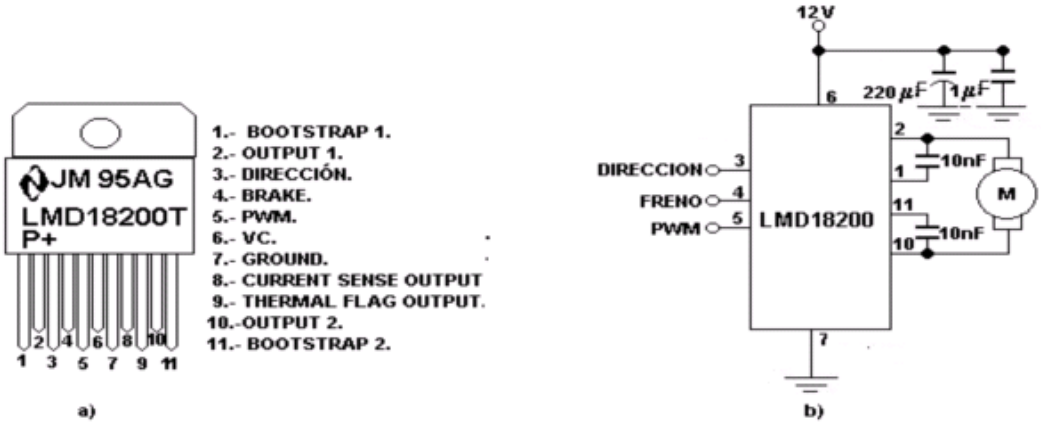


Figura 5.18. Estructura del puente h.

5.3.5. Sistema de control.

El elemento principal dentro del robot es el microcontrolador PIC16F84 de la compañía MICROCHIP, se emplean los sensores, un amplificador operacional LM324 como comparador y los puentes H, LMD18200.

Tiene dos puertos programables para entrada y salida que son el RAx y RBx donde RA puede ser desde RA0 hasta RA4 y RB0 hasta RB7.

Vss es alimentación a masa y VDD alimentación a positivo entre +2 voltios y 6 voltios, normalmente se utiliza 5 voltios.

MCLR es una patilla para habilitar a nivel bajo 0 voltios un reset al microcontrolador.

OSC1 YOSC2 se utilizan para configurar internamente o por hardware exterior la frecuencia de funcionamiento del PIC desde un mínimo de 1KHZ hasta los 10 MHZ de máxima que posee el pic. Normalmente se utiliza 4MHZ aprovechando que 4 ciclos de reloj producen 1 ciclo de ejecución de instrucción maquina, con lo

que 1 instrucción se ejecutaría en 1 MHZ y así poderlo tener en cuenta para calcular tiempos de retraso o espera en la ejecución del programa.

Este microcontrolador tiene un juego reducido de instrucciones de 32 instrucciones RISC de 14 bits cada instrucción, además posee una estructura Harvard de manera que diferencia en sitios diferentes los datos de las instrucciones, de modo que no pierde tiempo en diferenciar si es dato o instrucción como ocurría en la estructura Von Neuman.

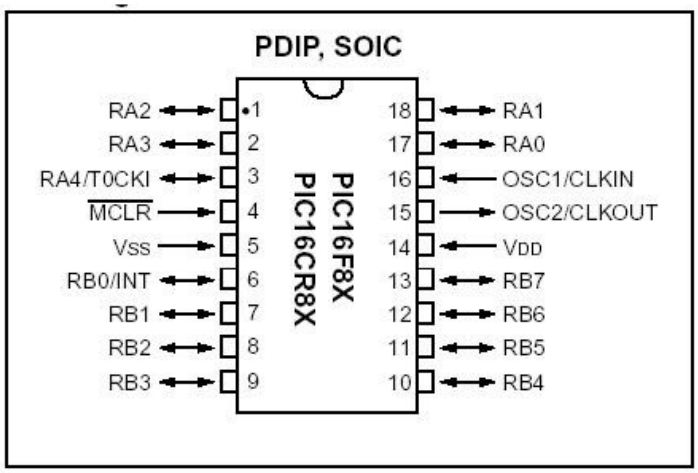


Figura 5.19. Distribución del PIC18F8

5.3.5.1. Conexión del PIC.

En este apartado veremos la forma de conectar el PIC16F84, ya que tiene que cumplir algunas especificaciones y características que necesitaremos para integrar el agente inteligente de nuestro robot.

5.3.5.2. Oscilador.

Todo microprocesador o microcontrolador requiere de una señal de reloj que sincronice su funcionamiento. Esta señal se obtiene mediante un oscilador de frecuencia.

Existen microcontroladores que tienen un oscilador interno y no requieren de componentes externos. El microcontrolador PIC16F84 requiere de un circuito externo de oscilación o generador de pulsos de reloj. La frecuencia de reloj máxima es de 4 MHz el cual utilizaremos en nuestro diseño.

El PIC16F84 puede utilizar cuatro tipos diferentes configuraciones de reloj. La elección dependerá de la precisión y velocidad que requiramos; por otro lado, el coste también es un aspecto a tener en cuenta a la hora de elegir uno u otro.

5.3.5.3. Oscilador TTL

Para obtener la frecuencia necesaria para nuestro PIC se utiliza un oscilador externo que está basado en un cristal que contiene toda la circuitería para generar una onda cuadrada. Este ha de ser conectado como si de un generador de señal externa se tratase. Al incluir toda la circuitería esto incrementa el costo del proyecto, que ya analizaremos mas adelante; pero resulta una forma interesante por la precisión en la señal de reloj emitida.

Estos tipos de cristales están diseñados especialmente para tecnologías TTL. La frecuencias disponibles para esta versión de cristal son muy amplias y las mas usuales son 1 - 1.8432 - 2 - 4 - 8 - 10 - 11.059 - 12 - 14.31818 - 16 - 20 - 25 - 32 - 33 - 40 - 50 - 80 y 100 Mhz.

Para un mejor manejo de señal he decidió tomar el cristal de 40 MHz. Para tener un rango amplio de manejo. En la imagen siguiente se muestra como debe conectarse al microcontrolador y las características del cristal.

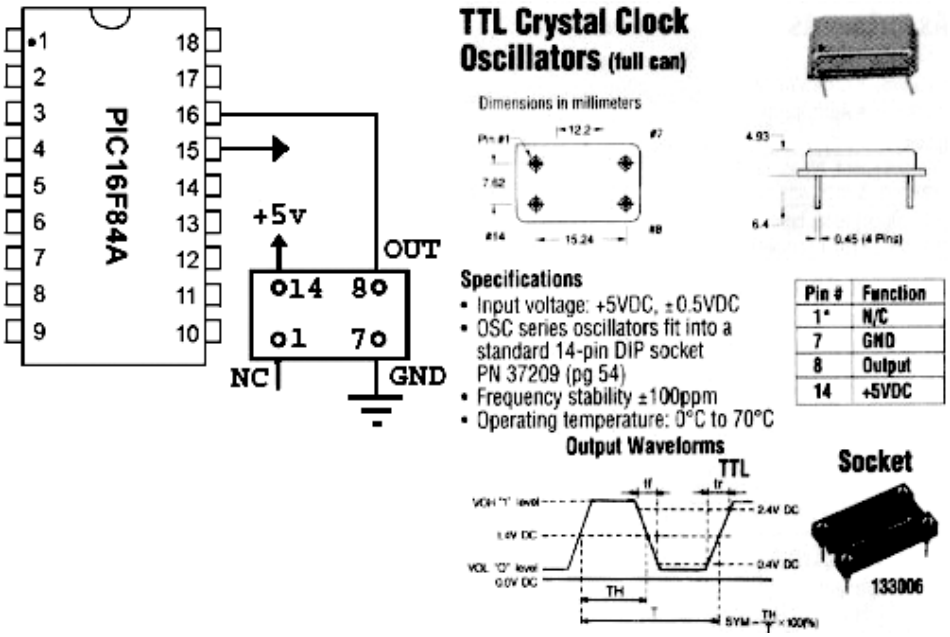


Figura 5.20. Conexión del cristal y el microcontrolador.

5.3.5.4. Reset del circuito.

A nuestro sistema de control le hemos agregado un reset manual a la patilla **MCLR** mediante un pulsador y una resistencia.

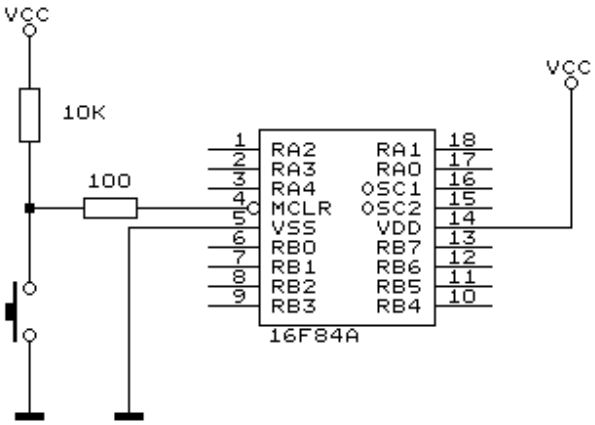


Figura 5.21. Conexión del reset.

Algunas reglas básicas para realizar este circuito son:

- Es recomendable que R1 sea menor de 40 K para asegurarse que su tensión no supere los 0,2 voltios cuando **MCLR** esté a nivel alto (la corriente máxima en la patilla MCLR es menor de 5 µA). Un voltaje mayor puede degradar el nivle V_{IH} de la patilla **MCLR**.
- Se recomienda que R2 esté entre 50 y 100 ohmios.
- El diodo D1 ayuda a provocar una descarga rápida del condensador cuando se elimina la alimentación.

Nota: Si en MCLR se aplica una tensión por debajo de VSS se pueden inducir corrientes mayores de 80 mA, que pueden causar problemas. Para solucionar esto se coloca una resistencia de 50 a 100 ohmios para aplicar un nivel bajo a la patilla MCLR en lugar de llevar esta patilla directamente a VSS. En nuestro caso esta resistencia es R2.

Para terminar habría que decir que, en la mayoría de los casos será más que suficiente con conectar **MCLR** a la alimentación positiva (VCC) y usar los dispositivos de reset del PIC.

La resistencia de 100 ohmios puede eliminarse del circuito al no existir peligro de que en MCLR aparezcan tensiones menores de VSS. Pero debería mantenerse si se coloca un condensador en paralelo al pulsador.

5.4. Integración de los distintos componentes del robot.

Al ya tener todas las partes que integran nuestro robot, es necesario unirlas para crear el diagrama electrónico del robot móvil.

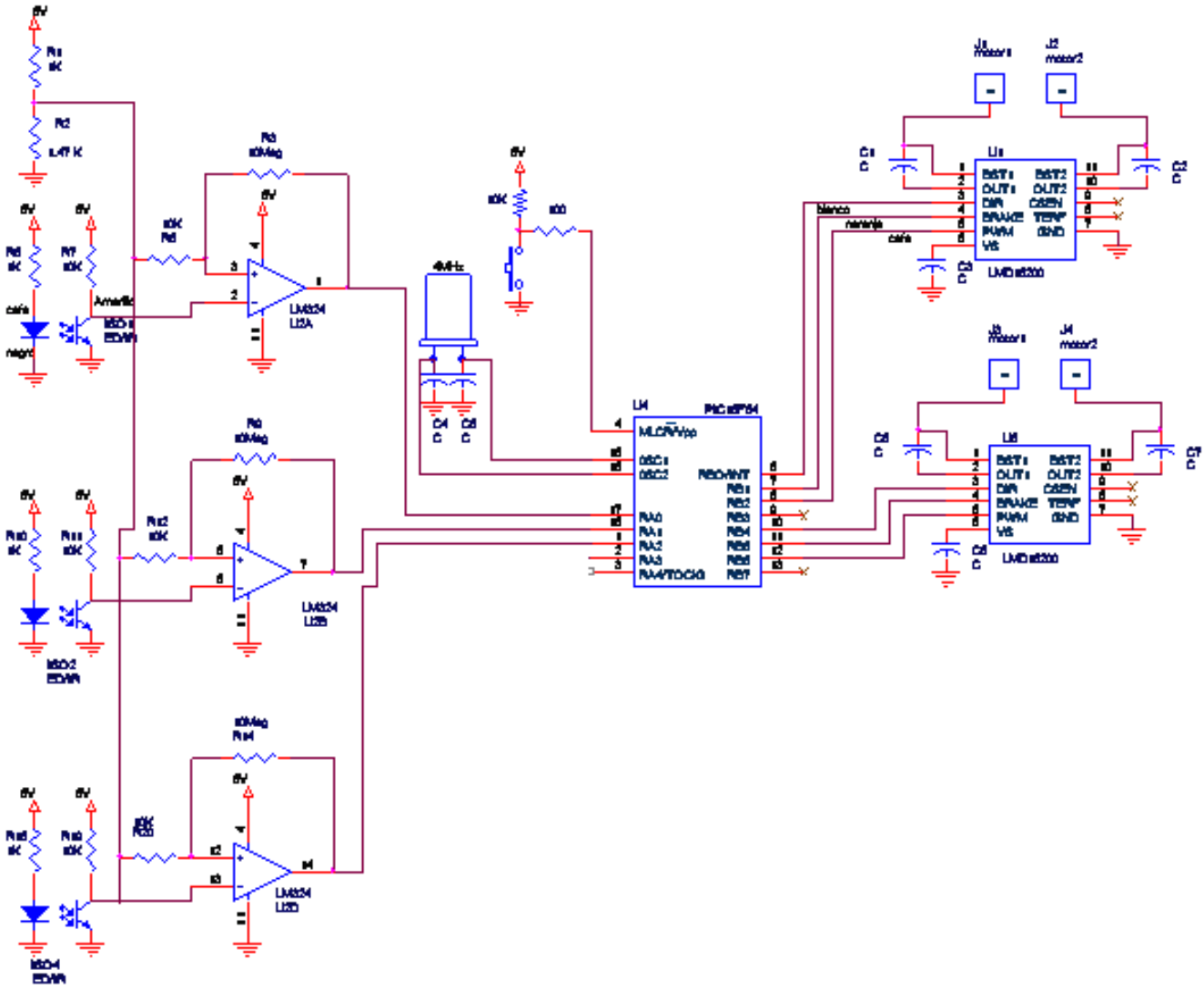


Figura 5.22. Diagrama electrónico del robot.

5.5. Evaluación económica del proyecto.

En este proyecto como se ha planteado desde el comienzo, la utilización de materiales de bajo costo.

El agente electrónico y de programación, se ha realizado de tal forma que no implique realizar gastos excesivos.

A continuación se detallaran los costos para la realización de este proyecto.

5.5.1 Costos de programación y de diseño.

En este apartado es donde encontramos una reducción de costos, ya que al realizar el programa en código c, y compilarlo en el software C++ compiler que es un software libre, no genera ningún costo de de adquisición.

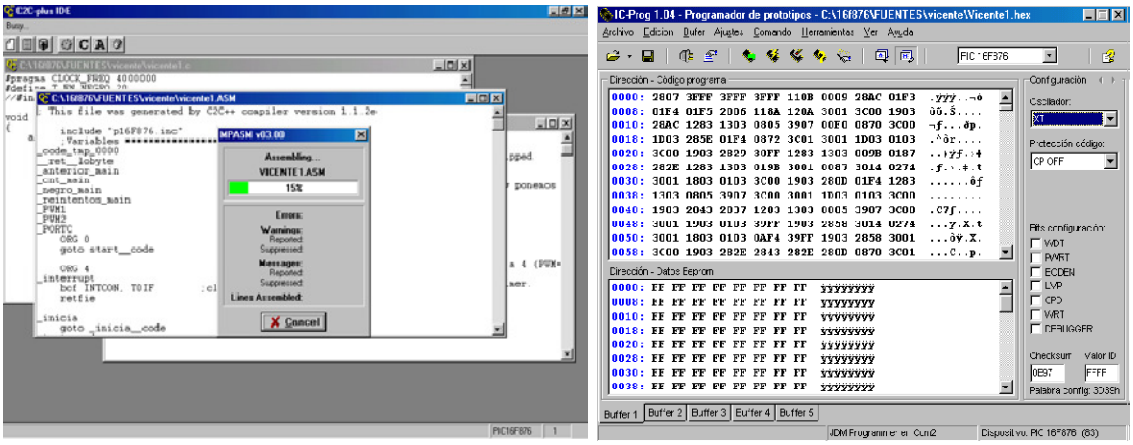


Figura 5.23. Software.

Para compilar el programa al PIC, se utilizo el programa MPLAB 9 de microchips, que al igual que los otros programas utilizados, son de código libre y descargable desde su página web.

La conexión del PIC al PC para el volcado se realiza con el programador SMT2.

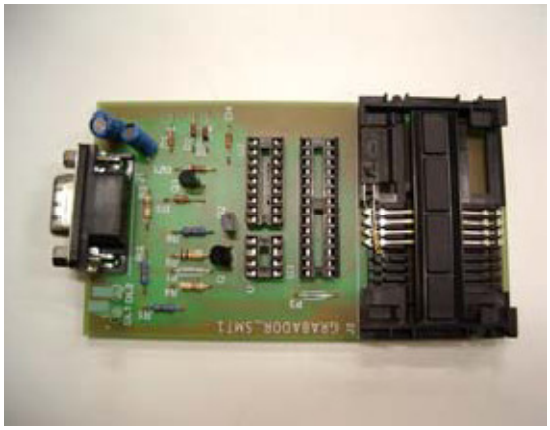


Figura 5.24. Grabador del PIC.

Los costos del diseño o también llamados costos de ingeniería se han establecido en base a la referencia de costos en el mercado de la robótica.

A continuación se detalla en la siguiente tabla los costos de software y de diseño:

Descripción	Costo en pesos
Software C++ Compiler	$0.00
Software MPLAB9	$0.00
Programador SMT2	$195.50
Costos de ingeniería	$10,500.00
TOTAL	$10,695.50

5.5.2. Costos del agente electrónico y de la estructura propuesta.

La realización de la parte electrónica, que integra el agente inteligente del robot, se ha seleccionado de tal forma que cumpla con las necesidades

planteadas al funcionamiento del robot.

Por su parte, la parte propuesta para la estructura consta de una tabla de MDF, que en el mercado la que se comercia que cubre nuestras dimensiones es la tabla de 60 x 50 cm. Para cortar la tabla de MDF a nuestras dimensiones solo ocuparemos una navaja y un apoyo de dirección como lo es una regla plástica.

Para la locomoción de el robot, solo hace falta una tira de goma para recubrir las llantas de MDF así, como dos servomotores ya descritos anteriormente y para su sujeción estos ya cuentan con el acoplamiento necesario para fijarlos con tornillos de ¼ de pulgada.

Por ultimo se utilizara una rueda Caster Wheel o también llamada rueda loca con base.

A continuación se detallan los costos del agente electrónico y de estructura:

Material	Precio por unidad	Cantidad	Precio total
Oscilador 40MHZ	$81.00	1	$81.00
PIC16F84	$140.00	1	$140.00
LM324	$53.00	2	$106.00
LMD18200	$62.00	2	$124.00
LM7805	$32.00	1	$32.00
HOA6389	$43.00	3	$129.00
Motores RE-280	$75.00	2	$150.00
Baterías nim	$13.00	6	$78.00
Diodo D1620	$3.00	1	$3.00
Resistencia 1.47k	$2.00	1	$2.00
Resistencia 10m	$2.00	4	$8.00
Resistencia 1k	$2.00	4	$8.00
Resistencia 10k	$2.00	7	$14.00

Material	Precio por unidad	Cantidad	Precio total
Capacitor 0.22uf	$5.00	1	$5.00
Capacitor 1uf	$5.00	4	$20.00
Capacitor 220uf	$5.00	4	$20.00
Switch	$10.00	1	$10.00
Tira de goma	$12.50	2	$25.00
MDF 60 X 50 cm	$80.00	1	$80.00
Rueda Caster wheel	$25.00	1	$25.00
		TOTAL	$979.00

Con esto se concluye el costo total de nuestro diseño lo cual es de $11,674.00.

Conclusiones.

En este trabajo, se ha planteado la concepción de robots móviles destinados a la competición en diversas categorías. En este sentido, en las diferentes fases de su diseño y construcción se ha llegado a diversos resultados que, a continuación, se detallan:

- Se ha utilizado el concepto de agente inteligente en la definición de las especificaciones y componentes que constituyen el robot. Se ha constatado que, para aquellos que están destinados a la participación de concursos de robótica, la autonomía se ha convertido en el problema central de diseño. En este sentido, se ha realizado la descripción PAMA de cada uno de los robots, lo que ha conducido a la propuesta de la arquitectura de agente más adecuada en cada caso o competición. Finalmente, se han implementado en el robot seguidor de línea diseñado.

- A partir de la arquitectura propuesta, se ha realizado un análisis de los sistemas de tracción más adecuados. En este sentido la configuración diferencial ha sido considerada como la más adecuada atendiendo fundamentalmente a un balance de los costos y la maniobrabilidad. Asimismo, se han considerado diferentes posibilidades motrices como es la utilización de servos o de motores de corriente continua con puente de potencia.

- Respecto a los sistemas sensores, la opción elegida ha sido la utilización de fotodetectores. Se trata de elementos de bajo costo cuyo procesamiento además requiere escasos recursos computacionales. Se ha desarrollado un procedimiento que permite disponer de una representación de la posición del robot lineal que es inmune al grosor de la línea.

- La arquitectura elegida ha sido la de agente reflejo con modelo interno. Ha sido la inaccesibilidad del ambiente la que ha justificado esta decisión. Ha sido necesario, en cada caso, desarrollar un autómata de estados finitos lo que ha requerido de un análisis detallado para determine los estados que describen al robot en cada instante, las transiciones así como las acciones en cada estado.

- El núcleo central del comportamiento inteligente del agente, se ha desarrollado utilizando la lógica borrosa. Esta elección se justifica por las ventajas que proporciona con suaves reacciones que permiten circular a velocidades altas sin oscilaciones que saquen al robot de la pista. Además, su aplicación se puede llevar a cabo mediante superficies o curvas de acción por lo que se reducen los requerimientos hardware y software. Así, el microcontrolador PIC 16F84 dispone de la potencia necesaria para la implementación final del control inteligente de los robots desarrollados.

Además de estas conclusiones donde se han mostrado los principales avances técnico-científicos, me ha parecido necesario incluir una serie de comentarios de índole práctico que pueden ser incluso más importantes que las anteriores. Éstos se sustentan en la experiencia personal de un constructor y desarrollador de robots y serán útiles para aquellos que afronten esta aventura.

- En este momento, con los conocimientos adquiridos, resulta relativamente sencillo llegar a una implementación funcional partiendo de los objetivos iníciales sin detenerse solamente en la simulación sin darnos cuenta que sobre el papel funciona todo.

- La realización de un robot se convierte en una tarea por etapas que permite ir realizando pruebas durante el desarrollo. Lo más recomendable es

primero realizar la placa principal del sistema, comprobando que funciona correctamente. De esta manera es posible probar los sensores y actuadores por separado con pequeños programas. En la parte de software es recomendable hacer lo mismo, ir probando pequeñas partes de código. En efecto, en sistemas de este tipo hay funciones que se llaman continuamente, por lo que se debe prestar mucha atención a la optimización del código.

- Los compiladores suelen generar un código ensamblador bastante bueno, ya que el lenguaje C se presta a ello. El uso de ensamblador directamente, solo se ve justificado cuando haya partes críticas en el tiempo.

- Otro aspecto importante ha sido el costo del desarrollo. Económicamente no ha sido un coste muy grande pero si lo ha sido en tiempo utilizado. Resulta evidente que se debe mantener la proporcionalidad del costo del desarrollo, y tratar de evitar gastar grandes sumas de dinero. Efectivamente existen en el mercado sensores láser, motores sin escobillas, y una multitud de sutilezas que podrían haber sido una solución magnífica para el desarrollo de un robot, pero que hubiesen multiplicado el costo por diez y deslucirían el resultado. La dedicación en horas se ve justificada por la ausencia de material de base, que poco a poco se ha ido solucionando.

Experimentalmente se ha concluido que cuando se crea un robot, una parte importante del tiempo es dedicada a solucionar problemas con los que no se contaba. Esto se puede solucionar en gran medida reutilizando diseños suficientemente probados. En mi caso, de forma evolutiva se ha llegado a la conclusión que la disposición de los distintos elementos del robot se asemeja bastante a los robots campeones de concursos. Como conclusión cabe pensar que en adelante debemos aprovecharnos de diseños ganadores y no tratar de reinventar la rueda.

Por último, como revisión de las experiencias vividas entorno al robot rastreador, se destacan las más importantes:

- El software es una de las partes más importantes del robot
- Hay que tratar de tener un costo razonable
- Es necesario conocer el tipo de IA que se va a utilizar.

Bibliografía.

Max Black. Vagueness, an exercise in logical analysis. *Philosophy of Science*, 4(4):427–455, 1937.

J.W. Burdick, J. Radford, y G.S. Chirikjian. Sidewinding locomotion gait for hyper-redundant robots. *Advanced Robotics*, 9(3):195–21, 1995.

J. Bares y D. Wettergreen. Dante II: Technical description, resultsand lessons learned. En *International Journal of Robotics Research*, tomo 18, páginas 621–649. Julio 1999.

C2c++ c++ compiler. http://www.picant.com/c2cpp/cpp.html.

J. M. Corchado y J. M. Molina. *Introducción a la Teoría de Agentes y Sistemas Multiagente*. Publicaciones Científicas, 2002.

Flexitrack. http://www.flexitrack.com.

Hispabot. http://www.hispabot.org/.

Ic-prog prototype programmer. http://www.ic-prog.com/.

Mabuchi motor. http://www.mabuchi-motor.co.jp/english/.

Karl Menger. Statistical metrics. En *National Academy of Science, U.S.A*, tomo 28, páginas 535–537. 1942.

Microchip. http://www.microchip.com.

M.Ñilsson. Snake robot free climbing. En *International Conference on Robotics and Automation*, páginas 3415–3420. 1997.

A. Ollero. *Robótica. Manipuladores y robots móviles*. Marcombo Boixareu Editores, 2001. ISBN 84 267 1313 0.

Stuart Russell y Peter Norvig. *Inteligencia Artificial. Un enfoque moderno*. Prentice Hall, 1996.

Sdr-4x. http://www.tokyodv.com/news/SonySDR-4XRobot.html.

Stmicroelectronics. http://www.st.com.

A.M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

Jose M. Angulo Usategui, Susana Romero Yesa, e Ignacio Angulo Martínez. *Macrobótica*. Paraninfo, segunda edición, 2001.

Vishay. http://www.vishay.com.

D. Wettergreen y C. Thorpe. Developing planning and reactive control For a hexapod robot. En *International Conference on Robotics and Automation*, tomo 3, páginas 2718 – 2723. Abril 1996.

Mark Yim. Climbing with snake-like robots. En *IFAC Workshop on Mobile Robot Technology*. Mayo 2001.