UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO CONSTRUCCIÓN Y CONTROL DIFUSO DE UN ROBOT MÓVIL DE COMPETENCIA

TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO ELÉCTRICO-ELECTRÓNICO

MÓDULO SISTEMAS DIGITALES

P R E S E N T A :

AMARANTO DE JESÚS DÁVILA JÁUREGUI

Y PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

DIANA AURORA CRUZ HERNÁNDEZ

DIRECTOR DE TESIS:

Dr. Yu Tang Xu

CIUDAD UNIVERSITARIA, MÉXICO, D. F., 2004

Agradecimientos:

A mis padres Humberto Cruz Ramírez y Margarita Hernández Baltazar por su apoyo ejemplo y confianza brindados durante toda mi vida, este logro es tan suyo como mío.

A mis hermanas Luz Adriana y Sofía, a mi amiga Cecilia, quienes siempre han confiado en mí.

A mis tías Rosario y María por su cariño y apoyo.

A la persona clave para mí en la realización de este proyecto quien nunca permitió que me diera por vencida: a mi novio, amigo y compañero: Amaranto de Jesús Dávila Jáuregui.

Diana Aurora.

Autorizo a la Dirección General de Bibliotecas de la UNAM a difundir en formato electrónico e impreso el contenido de mi trabajo recepcional.
NOMBRE: Diana Aurora Cruz Hernández
FECHA: 23 - enero - 2004
FIRMA:

A mi padres H. Amaranto Dávila Ramírez y Esperanza Jáuregui Aguilar; a mi hermana Columba Dávila Jáuregui, y a mi abuela Lorenza Ramírez, por todo el apoyo, confianza y cariño que me brindan.

A Ignacio Jáuregui y Oscar Santana por su ejemplo.

A mi novia Diana Aurora Cruz por apoyarme, soportarme y quererme.

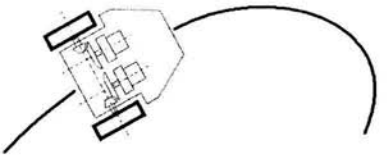
Amaranto.

Autorizo a la Dirección General de Bibliotecas de la UNAM a difundir en formato electrónico e impreso el contenido de mi trabajo recepcional.
NOMBRE: Amaranto de Jesús Dávila Jáuregui
FECHA: 23/01/2004
FIRMA:

Gracias a Dios por ver concluida una etapa más de nuestras vidas.

Gracias a la UNAM por brindarnos la oportunidad de ser parte de la mejor Universidad de México.

Índice



1. Introducción

La robótica es uno de los temas que más despiertan la imaginación asociada con ideas futuristas, aunque es una idea muy vieja, puesto que el hombre desde siempre ha tratado de construir máquinas que realicen el trabajo por él. La mayoría de las personas tiene la idea de que los robots deben tener apariencia fantástica y capacidades humanas. En el sentido más general podemos decir que un robot es un mecanismo electromecánico capaz de realizar una tarea de forma autónoma o casi autónoma, a través de la programación de un algoritmo en su sistema de control. Bajo esta definición hay actualmente una infinidad de mecanismos electrónicos que cabrían en ella, desde los brazos industriales que se encuentran en las más modernas líneas de producción hasta los electrodomésticos más comunes como el horno de microondas. Los robots industriales o brazos manipuladores fueron los primeros en desarrollarse debido a su aplicación en la industria, sin embargo la robótica es mucho más que esto y los robots móviles también juegan un papel importante en este campo.

En los últimos tiempos se ha registrado un creciente interés en la investigación de robots móviles, debido en parte al alto grado de desarrollo que se ha alcanzado en el área de sensores y mecanismos. Lo anterior ha permitido la implantación de sistemas con un alto grado de interacción con el medio logrando que el robot obtenga una descripción más fiel de su entorno, la cual puede emplear para realizar sus tareas con más exactitud.

La diferencia principal entre los robots industriales convencionales (manipuladores) y los robots móviles es que estos últimos no están anclados, sino por el contrario pueden desplazarse por el terreno, por el agua o incluso volar. Los aspectos más relevantes de la robótica móvil son los relacionados con tales desplazamientos autónomos o navegación del robot. Donde el papel de la robótica móvil es más importante es en las nuevas aplicaciones como son los robots de servicio entre los cuales se incluyen los robots domésticos, robots de ayuda a discapacitados, y robots asistentes en general.

A los robots móviles podemos clasificarlos en términos de la manera en la cual obtienen su propulsión, que pueden ser por ruedas, orugas u otros. Los llamados vehículos guiados automáticamente (AGV, por sus siglas en inglés), son

una aplicación muy difundida de este tipo de robots, sobre todo en el sector de manufactura, muchos de ellos se emplean en el traslado de material y/o productos a ciertas áreas de la planta, logrando una mayor sistematización y una mejor integración de la planta hacia esquemas de manufactura integrada por computadora (CIM).

La característica fundamental de este tipo de robots es su alto grado de autonomía; es decir, el robot tiene todos los subsistemas necesarios para desarrollar sus tareas, los cuales deberán interactuar adecuadamente a fin de que la tarea emprendida por el robot sea ejecutada satisfactoriamente.

Los concursos de robótica y mini robótica, sirven de semillero para incrementar el desarrollo tecnológico en esta área. México es un país que no se caracteriza por su desarrollo en este campo, por lo tanto este tipo de concursos fomentan el interés de los jóvenes y su incursión en él. Los vehículos rodantes son los más comunes en estos concursos por ser mecánicamente simples y fáciles de construir. Además se pueden encontrar varios dispositivos con ruedas, como juguetes, que pueden ser modificados para utilizarlos en la construcción del robot.

Este proyecto se inició con la idea de desarrollar un robot móvil seguidor de línea para participar en este tipo de concursos, específicamente en la categoría *"robot móvil de velocidad"* [27], el objetivo de esta competencia es que el robot siga una trayectoria definida por una línea blanca sobre un fondo negro lo más rápido posible con fidelidad a la trayectoria. El desarrollar un robot es pensar en un proyecto mecatrónico en el cual se conjugan cuatro áreas básicas de la ingeniería: mecánica, electrónica, control y computación.

La primera parte de este trabajo describe brevemente las configuraciones mecánicas más comunes en este tipo de robots; seleccionar adecuadamente el tipo de tracción a utilizar es determinante para un buen desempeño, ya que cada una cuenta con diferentes características que las hacen más o menos apropiadas dependiendo de la aplicación del robot. El sistema de transmisión es otro de los temas tratados, para concluir presentando el diseño mecánico final del robot.

En el capítulo 3 presenta el sistema electrónico del robot. Se explican dos alternativas utilizadas para sensar la línea que define la trayectoria así como la forma en que se leen las velocidades de las llantas; también se describen las características principales y ventajas del microcontrolador utilizado. Los motores utilizados tienen un alto consumo de corriente y bajo voltaje de operación, lo que requiere de una cuidadosa selección de la etapa de potencia. La mayoría de los circuitos ofrecidos soportan grandes corrientes pero funcionan con voltajes mayores a 12 [V], en este capítulo se plantea una solución a este problema. Otro aspecto incluido son las baterías del robot, por ser estas uno de los componentes que determinan el peso y el costo del robot, se hace un breve análisis de las características de los distintos tipos de baterías que se pueden encontrar en el mercado, a fin de hacer una selección adecuada.

En los últimos años se ha constatado un crecimiento significativo de aplicaciones basadas en lógica difusa. El éxito del control difuso se ha hecho patente, en buena medida, cuando sus aplicaciones dejaron de ser simples proyectos de laboratorio y han finalizado sus posibilidades de comercialización, desde control de procesos industriales hasta objetos de consumo próximos a nosotros como cámaras fotográficas, aspiradoras, lavadoras de ropa, sistemas de aire acondicionado, etc. En la robótica existen diversas posibilidades de aplicación del control difuso por tal motivo se eligió un esquema de control difuso para el robot.

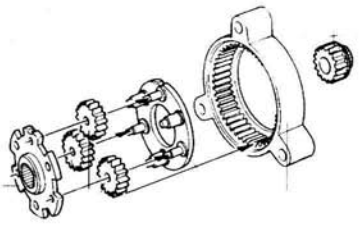
En el capítulo cuatro se describe el esquema de control implementado, revisando los conceptos básicos de la lógica difusa necesarios para el diseño del controlador. Partimos de la identificación del escenario en el cual se mueve el robot con extracción de las características del entorno y obtención de las variables de situación del robot. Las reglas de control difuso relacionan valores cualitativos de estas variables con las acciones de control, en este caso el ángulo de dirección del vehículo. Los consecuentes de las reglas emplean una combinación lineal de las variables de situación empleando coeficientes que se obtienen experimentalmente. Una vez obtenidos los valores inferidos por cada regla, la acción final de control resulta de una suma ponderada normalizada de los resultados de cada regla.

Se obtuvo un sencillo modelo matemático del robot móvil, con el fin de poder simular el controlador y hacer ajustes en la estrategia de control. El capítulo cinco describe el desarrollo del modelo, así como la forma en que se obtuvo el ángulo de desviación para efectos de simulación. Las simulaciones se realizaron en Matlab creando un esquema en simulink, esta simulación nos ayudó a tener una idea general del comportamiento del controlador.

El capítulo seis describe la implementación física del control en el microcontrolador seleccionado, explica brevemente el algoritmo programado mostrando los diagramas de flujo asociados al código. La programación intenta ser lo más eficiente posible, identificando cada una de las tareas a realizar para tener una estructura modular en el código. Se incluye una parte de simulación del código en ensamblador, para verificar el código a implementar; para esto se empleó una herramienta de desarrollo que proporciona el fabricante del microcontrolador llamada MPLAB. Este capítulo incluye todas las consideraciones que se tomaron en cuenta para implementar el controlador, así como el resultado de las pruebas físicas realizadas.

El capítulo final reporta por medio de gráficas el desempeño del robot así como las conclusiones derivadas del desarrollo de este proyecto.

Se anexa la bibliografía consultada y apéndices con información sobre el software utilizado, componentes electrónicos y códigos de programación.



2. Diseño Mecánico

2.1 Características mecánicas.

Las características establecidas por las reglas de los concursos [27], para el robot y la pista, son las siguientes:

1. *Diseñar y construir un robot móvil autónomo que sea capaz de seguir una trayectoria blanca dibujada sobre fondo negro.*
2. *El robot podrá ser controlado con microcontroladores, o algún otro tipo de controlador que gestione los movimientos del robot de manera autónoma.*
3. *Los robots deberán ser capaces de superar obstáculos en la pista de hasta 1 [mm] de alto o una cavidad con una profundidad de hasta 1 [mm].*
4. *Las dimensiones máximas del robot serán 250 [mm] de largo, por 200 [mm] de ancho, sin haber restricciones en cuanto a altura.*
5. *El ancho de la línea es de 15 [mm].*
6. *La pista podrá presentar ondulaciones máximas de 3 [mm] en 100 [mm] de desplazamiento.*
7. *Las curvas tendrán como mínimo 15 [cm] de radio de curvatura, la trayectoria del robot será una curva suave.*
8. *La pista no se bifurcará en ningún punto del recorrido y será un circuito cerrado.*

El objetivo del robot en la competencia es hacer el menor tiempo de recorrido, con la mayor fidelidad posible a la trayectoria.

Actualmente la velocidad promedio en los concursos de velocidad es 1 [m/s].

De acuerdo a las reglas anteriores y a las velocidades que se manejan en la competencia, las características mecánicas deseadas son:

- Un diseño ligero.
- No mayor a las dimensiones citadas anteriormente (250x200 [mm]).

- Capaz de girar curvaturas de al menos 150 [mm] de radio.
- Con un centro de gravedad bajo.
- Capaz de alcanzar velocidades cercanas o mayores a 1 [m/s].
- Agarre suficiente en las llantas.

2.2 Sistema de locomoción.

Los robots móviles pueden tener varias formas de locomoción, las más comunes son por llantas, extremidades u orugas.

La locomoción más usada en robots móviles de competencia es por medio de llantas, ya que en superficies planas tiene mayor eficiencia que las extremidades u orugas; aunque su principal desventaja es, que al moverse sobre terrenos desiguales tienen un bajo desempeño.

Como regla un vehículo rodante tiene problemas si la altura del obstáculo supera el radio de las llantas. Una solución simple es que las llantas sean tan grandes como los obstáculos que tiene que superar, sin embargo en muchos casos esto es bastante impráctico. En nuestro caso esto no representa ningún inconveniente dado que la superficie sobre la cual el robot hará su recorrido es totalmente plana. Por lo anterior para el robot se eligió la locomoción por llantas.

Para un robot móvil con llantas, se pueden escoger diferentes configuraciones de tracción. Estos arreglos (Fig. 2.1) son: diferencial, síncrono, triciclo y tipo carro.

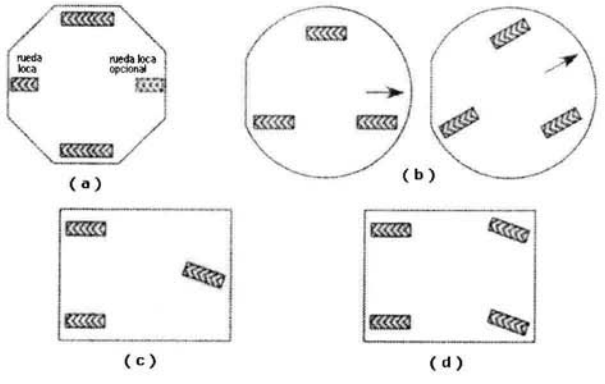


Figura 2.1: Algunos arreglos de llantas vistos desde abajo. (a) Diferencial: Usa una o posiblemente dos ruedas locas. (b) Síncrono: Todas las llantas giran al mismo tiempo. Las llantas de dirección y tracción están mostradas en dos diferentes direcciones. (c) Triciclo: Tiene el motor de dirección en una llanta y la de tracción en el par trasero. (d)Carro: Las dos llantas delanteras rotan para dar la dirección

2.2.1 Tracción Síncrona.

Un arreglo conocido como tracción síncrona se ilustra en la figura 2.1(b); en este tipo de arreglo, todas las llantas (usualmente tres) tienen tracción y dirección, las llantas están enlazadas de tal forma que todas apuntan en la misma dirección todo el tiempo; para poder cambiar la dirección del movimiento, el robot gira simultáneamente todas las llantas sobre un eje vertical como se muestra en la figura 2.1(b), por lo tanto la dirección del movimiento del robot cambia pero el chasis de éste continúa apuntando a la misma dirección. Si el robot va a tener un frente (presumiblemente en donde se colocan los sensores), se deben agregar mecanismos adicionales para mantener el cuerpo apuntando a la misma dirección que las llantas, el sistema síncrono soluciona muchos problemas asociados a la locomoción diferencial, triciclo y tipo carro, con un aumento en la complejidad mecánica.

2.2.2. Tracción tipo Triciclo y Carro.

La tracción tipo carro por sus cuatro puntos de apoyo proporciona buena estabilidad, la locomoción tipo triciclo tiene características similares con la ventaja de ser mecánicamente más simple debido a que la locomoción tipo carro requiere algún tipo de enlace entre las dos llantas de dirección. En general para ambas tracciones las dos llantas fijas estarán conectadas a un motor de tracción y las llantas o llanta de dirección girarán libremente.

Sin embargo en algunos robots las llantas de dirección también cuentan con tracción, en este tipo de locomoción no es necesario monitorear la velocidad de las llantas para lograr una trayectoria recta, basta con posicionar la llanta de dirección en un punto neutral. Una de las principales desventajas sería que a altas velocidades se tienen deslizamientos de la parte trasera del robot.

Estos tipos de locomoción se muestran en la figura 2.1(c) y 2.1(d).

2.2.3 Tracción Diferencial.

La configuración diferencial puede ser uno de los sistemas de tracción menos complicados en su construcción. El esquema diferencial consiste en dos llantas sobre un eje común (Fig. 2.1(a)), una llanta se controla independiente de la otra, acoplándola directamente o por medio de engranes a un motor. Tal arreglo da al robot la habilidad de ir derecho, girar en arco y dar vuelta sobre su propio eje.

Una de las consideraciones al diseñar un vehículo con tracción diferencial es la forma de balancearlo, usualmente se resuelve este problema colocando una o dos ruedas locas, con la menor fricción posible, en un arreglo de triángulo o diamante. Se debe tomar en cuenta que aún cuando a sendos motores se les

aplique el mismo voltaje estos girarán a diferentes velocidades debido a varios factores como la fricción. Por lo tanto la velocidad de las llantas debe ser controlada dinámicamente, lo cual implica monitorear y controlar la velocidad del motor cuando el robot esta en marcha, ya que si por ejemplo se desea que el robot siga una trayectoria recta el aplicar el mismo voltaje a los motores no será suficiente para lograr este objetivo. La simplicidad del diseño diferencial implica un incremento en la complejidad del sistema de control requerido. Disminuir la complejidad mecánica a favor de incrementar la complejidad electrónica y de software es frecuentemente lo más rentable y confiable.

Considerando las descripciones anteriores y las características deseadas se optó por una configuración diferencial, dada su sencillez mecánica y su bajo costo de construcción, respecto a las otras configuraciones.

2.3 Selección de motores.

Un motor eléctrico convierte energía eléctrica en energía mecánica. Los motores de DC son comúnmente usados en robots móviles pequeños porque típicamente se energizan con baterías de DC; además se encuentran en una gran variedad de tipos y tamaños. Para las necesidades de un robot, un motor de DC usualmente maneja muy altas velocidades y muy bajos torques. Para intercambiar estas características, se usan sistemas de reducción por medio de engranes.

Un pequeño sistema de engranes puede ser agregado al eje del motor o puede comprarse un motor con el engranaje ya incluido junto con la cubierta del motor. Este tipo de motores son llamados motores con cabeza reductora y son muy usados para hacer robots pequeños. Los motores con cabeza reductora están normalmente basados en imanes permanentes sin rotor de hierro para ser lo más ligero posible. Estos motores pueden además comprarse con salidas de encoder integrado. Otro tipo de motores de DC muy usados son los servo motores de radio control, estos son más bien un ensamble que incorpora un motor de DC, una reducción, límites de giro, un potenciómetro para retro-alimentar velocidad y un circuito integrado para control de posición. Un servomotor no gira completamente, a menos que se modifiquen para ello.

Los fabricantes de los motores generalmente entregan una hoja de datos con las especificaciones de los motores y otra con las especificaciones de la cabeza reductora, si es que cuentan con ella, estas hojas muestran con gráficas las características de torque, corriente y potencia que entregan los motores.

Para la aplicación se seleccionaron los motores Mabuchi RE260, cuya hoja de especificaciones se muestran en el apéndice A. Estos motores tienen las características siguientes:

Voltaje máximo de operación: 4.5 [VDC].
Torque de la flecha: 70 [gr/cm].
Velocidad sin carga: 10500 [rpm].
Corriente máxima con carga: 2.7 [A].
Peso: 28 [gr].

Estas características cumplen con las especificaciones deseadas. El voltaje de operación permite utilizar baterías pequeñas sin que esto implique un costo excesivo. La velocidad es, como ya se mencionó, muy alta para este tipo de aplicaciones, lo que nos permite agregar un sistema de reducción para aumentar el torque y lograr al mismo tiempo una velocidad adecuada. Otra ventaja importante son las reducidas dimensiones y peso.

2.4 Caja de engranes.

El sistema de engranes utilizado consiste en dos discos con un arreglo de engranes tipo planetario marca Tamiya (Fig. 2.2), los cuales proporcionan una reducción 4:1 cada uno, logrando así una reducción total de 16:1.

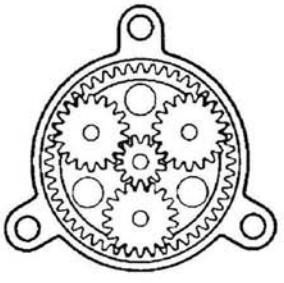


Figura 2.2: Arreglo de engranes tipo planetario

Los engranes planetarios constan de un engrane sol, tres engranes planeta, un brazo y un anillo cada uno (Fig. 2.3).

Las cajas de engranes planetarios tienen las siguientes características:

- Tienen radios pequeños o grandes en un diseño muy compacto.
- El engrane sol está localizado en el centro de todo el engranaje (sistema solar).
- El brazo conduce a los engranes planeta en orbita alrededor del engrane sol.
- Los engranes planeta orbitan en la parte interior del anillo.

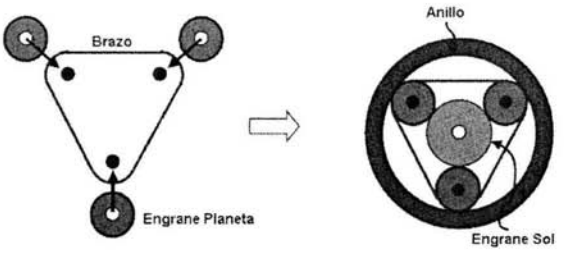


Figura 2.3 Partes de un arreglo de engranes planetarios

La figura 2.4 ilustra como se conectan varios arreglos planetarios para dar un mayor torque al motor y lograr mayor reducción en la velocidad.

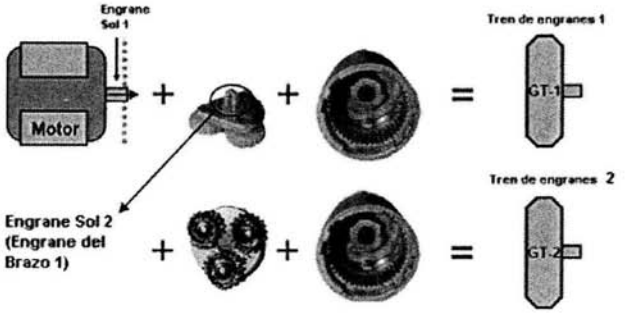


Figura 2.4 Conexión de dos discos de engranes planetarios

Podemos obtener la velocidad de la flecha de las llantas con la siguiente expresión:

$$\omega_{ai} = \frac{\omega_{ri} N_{ri} - \omega_{si} N_{si}}{N_{ri} + N_{si}} \quad \text{.....(2.1)}$$

Donde:

Ni = Número de dientes de engrane i.
ωi = Velocidad angular del engrane i.

Subíndices

1 = Componente del tren #1.
2 = Componente del tren #2.

s = Engrane Sol.
p = Engrane Planeta.
r = Engrane Anillo.
a = Brazo.

Ns1 = 11

Nr1 = Nr2 = 49

Ns2 =16

ω_{s1} = 10500 [rpm]

$\omega_{s2} = \omega_{a1}$

Sustituyendo los valores de los engranes utilizados para el primer disco en la ecuación 2.1 tenemos que:

$$\omega_{a1} = 1925 \text{ [rpm]}$$

Con este valor sustituido en la misma ecuación como ω_{s2} podemos obtener la velocidad de la flecha

$$\omega_{a2} = 474 \text{[rpm]}$$

Con esta información y con las características de las llantas que se detallan más adelante sabemos que la velocidad máxima es igual a 1.38 [m/s], lo cual cumple con las especificaciones deseadas.

Los engranes están diseñados para acoplar directamente a la flecha del motor, facilitando le montaje de los mismos. También cuentan con los accesorios necesarios para acoplar las llantas y fijarlos al chasis del coche (Fig. 2.5).

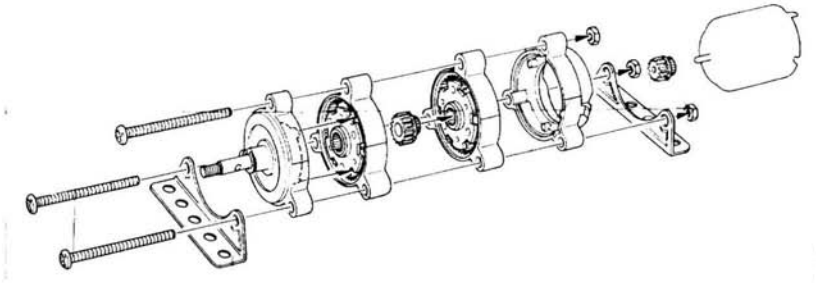


Figura 2.5 Caja de engranes planetarios Tamiya

2.5 Llantas.

Las llantas utilizadas para la tracción son de huele lo cual permite tener buena adherencia sobre la superficie en la que viaja el robot, además de tener canales para un mayor agarre. Tienen un diámetro de 56 [mm] y 25 [mm] de ancho. El rin tiene un adaptador para acoplar el motor con facilidad como muestra la figura 2.6.

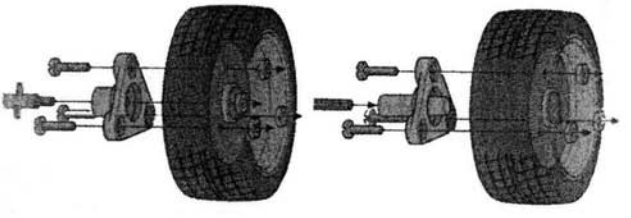


Figura 2.6 Llantas marca Tamiya 56 mm de diámetro

Para balancear el robot se utilizó una rueda loca muy pequeña con el fin de mantener el centro de gravedad bajo y no ocupar demasiado espacio. La figura 2.7 muestra la rueda loca utilizada.

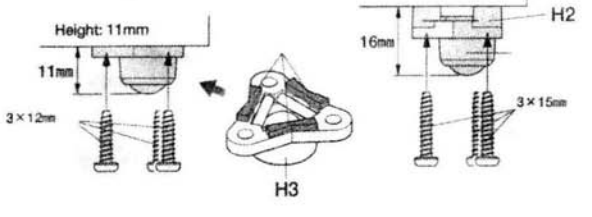


Figura 2.7 Rueda loca

2.6 Configuración mecánica final.

Una vez seleccionados todos los componentes mecánicos se definieron las dimensiones de la plataforma. El diseño final tiene una plataforma de fibra de vidrio de 12 x 11 [cm] de la cual sale un soporte de 12 [cm] de largo por 4 [cm] de ancho para colocar el sensor. Las dimensiones totales del robot son 22 x 17 [cm] incluyendo las llantas.

Las baterías se colocan en la plataforma principal delante de los motores, para que el peso quede uniformemente distribuido. Sobre los motores y baterías se colocó otra base para colocar los circuitos.

La figura 2.8 muestra un esquema del diseño mecánico final del robot.

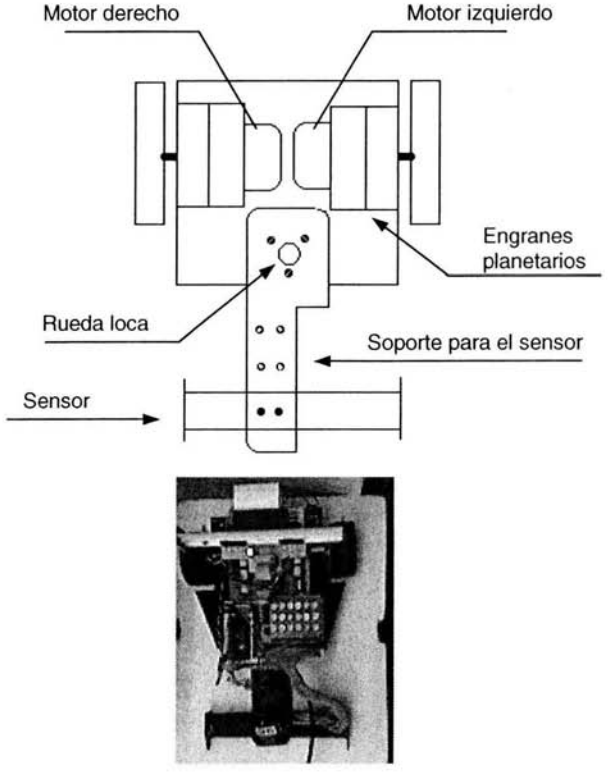
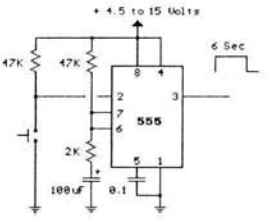


Figura 2.8: Diseño Mecánico Final; diseño diferencial con una rueda loca de apoyo y caja de engranes planetarios



3. Diseño Electrónico

3.1 Etapa de sensado.

La habilidad de un robot para sensar su mundo y cambiar su comportamiento en base a esto, es lo que hace a un robot un artefacto útil y con cierta inteligencia. Sin los sensores los robots no serían más que una automatización fija realizando una tarea repetitiva una y otra vez en un ambiente cuidadosamente controlado. Con ayuda de los sensores un robot puede operar en ambientes no controlados y adaptarse mientras el entorno cambia alrededor de él.

En un robot móvil de velocidad nos interesa detectar la trayectoria a seguir conforme el robot avanza sobre esta. Por ser un diseño diferencial, además de conocer el ángulo de desviación con respecto a la trayectoria es necesario conocer la velocidad de cada una de las llantas.

3.1.1 Sensor de línea.

Para sensar la trayectoria a seguir se diseñó un sensor capaz de entregar un valor proporcional al ángulo de desviación.

Las características deseadas del *sensor de línea* fueron las siguientes:

- Capacidad para identificar superficies de alto contraste (blanco-negro).
- Inmunidad a cambios de iluminación del ambiente.
- Estabilidad en valores de salida.
- Bajo consumo de potencia.
- Tiempo de respuesta mínimo.
- Capacidad de entregar valores analógicos o digitales.

El funcionamiento del sensor se basó en la cuantificación de la cantidad de energía reflejada por una superficie al hacer incidir un haz de luz sobre ella. Con estas características podemos elegir los componentes electrónicos que más se adecuen a los requerimientos del diseño; las opciones que se tienen para sensar la línea pueden ser las fotorresistencias y fototransistores debido a su bajo costo y fácil implementación.

La resistencia eléctrica de una fotorresistencia en la oscuridad es muy alta, alrededor de 1 [MΩ] o más, disminuyendo cuando la fotorresistencia es iluminada; al hacer una análisis de las características de las fotorresistencias, se encontró que tienen "efecto memoria" por lo que su tiempo de respuesta a cambios de intensidad de luz es muy alto, requiriendo de hasta un segundo para regresar a su estado de alta resistencia después de que la fuente de luz es retirada; a pesar de ser muy sensitivas y fáciles de usar por tratarse de una aplicación en donde se requieren tiempos mínimos de muestreo no representan una buena alternativa.

Por otro lado los fototransistores son dispositivos semiconductores que permiten un mayor o menor paso de corriente entre sus terminales emisor y colector (E y C) dependiendo de la cantidad de luz que llega a su parte superior o ventana. Esta última corresponde a la base (B) y posee un lente que le proporciona una mayor sensibilidad que la de la fotorresistencia. Esta corriente puede ser convertida fácilmente en un voltaje que maneje un determinado circuito o llevarse a un valor digital que pueda ser interpretado por un microprocesador o un microcontrolador. Los fototransistores pueden ser usados casi con cualquier tipo de luz visible o infrarroja.

Otras características del fototransistor son:

- Larga vida útil.
- Bajo consumo de potencia.
- Generación mínima de calor.
- Disponible en gran variedad de encapsulados.
- Bajo costo.
- Tiempo de respuesta mínimos (microsegundos).
- Mismas características generales de los transistores comunes.

Las características del fototransistor son las que más se acercan a las deseadas, especialmente el tiempo de respuesta por lo que se decidió utilizar varios pares emisor-receptor infrarrojos en el diseño del sensor. Dado que el fototransistor responde a varios tipos de radiación entre ellas la luz natural se debe hallar la manera de filtrar otro tipo de radiaciones que puedan llegar a afectar el funcionamiento del sensor.

Por esto se utilizó el fototransistor PT1302/C2 de alta velocidad (3 microsegundos de recuperación), cuyo rango de detección es de 700 – 1200 [nm] que además posee un filtro que solo permite el paso de luz infrarroja emitida por un diodo emisor IR, dando una respuesta estable en cualquier condición de

iluminación, además el sensor se protegió con una cubierta obscura que impide que los fototransistores y emisores se ensucien y desacomoden por algún golpe.

Los diodos emisores de luz (LED) infrarrojos son azul transparente, de 5 mm de diámetro, con longitud de onda de 940 [nm], la cual acopla con el rango de longitud de onda de detección del fototransistor.

La configuración a utilizar es la siguiente (Fig. 3.1):

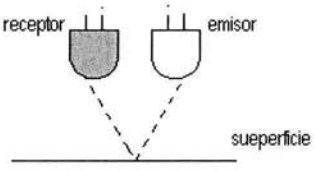


Figura 3.1: Arreglo utilizado en el sensor infrarrojo

En este arreglo el haz de luz infrarroja sale del emisor hacia la pista, parte de la energía emitida es reflejada por la superficie hacia el receptor, la cantidad de luz infrarroja reflejada al receptor depende del color de la pista, en una superficie blanca habrá más reflexión que en la superficie negra.

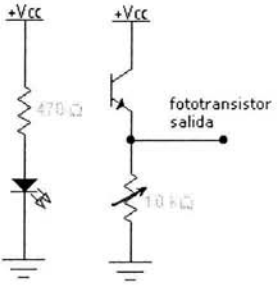


Figura 3.2 Configuración del par emisor - receptor

En la figura 3.2 el diodo infrarrojo tiene una resistencia de 470[Ω] para limitar la corriente. La parte receptora está formada por un circuito amplificador en configuración colector común. El fototransistor estará funcionando en la región activa, esto es, que generará una respuesta proporcional a la cantidad de luz recibida en la base del fototransistor hasta cierto nivel, cuando la cantidad de luz sobrepasa este nivel el fototransistor se satura y la salida no se incrementa

aunque la cantidad de luz siga aumentando. Un valor correcto para las resistencias puede determinarse por la siguiente ecuación.

$$Vcc > R \times Icc \quad \ldots\ldots(3.1)$$

Se colocó un potenciómetro de 10 [kΩ] para poder calibrar el sensor dependiendo de las características de reflexión de la pista.

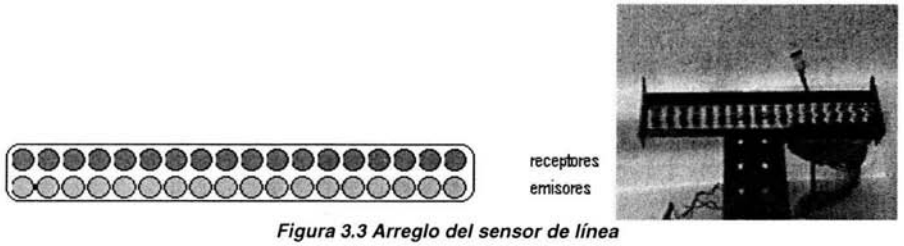


Figura 3.3 Arreglo del sensor de línea

Se utilizaron dieciocho pares *emisor – receptor* como los mostrados en la figura 3.2 colocados a una distancia de un centímetro uno del otro en línea recta como se muestra en la figura 3.3. Según las características del sensor dadas arriba la salida del sensor deberá ser analógica y digital por lo que se deberán hacer dos arreglos distintos para lograr lo dos tipos de salida.

3.1.1.1 Sensado digital.

Para obtener la salida digital el voltaje obtenido en cada uno de los receptores es enviada a un comparador analógico basado en el circuito LM339 el cual es un comparador de voltaje de precisión diseñado para trabajar con una sola fuente de alimentación para así obtener una señal digital para cada fototransistor ("1" sobre blanco, "0" sobre negro) Fig. 3.4.

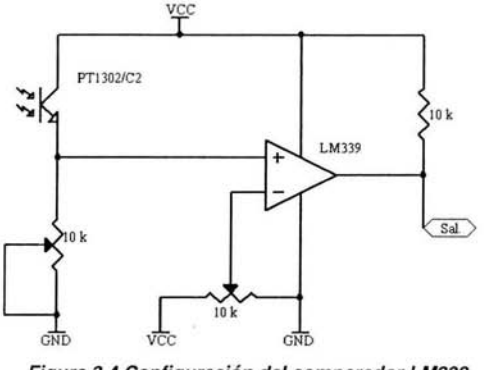


Figura 3.4 Configuración del comparador LM339

El C.I. LM339 tiene cuatro comparadores de voltaje por lo que se utilizaron cinco *chips* de este tipo para cubrir todas las salidas de los fototransistores y así tener una palabra digital de dieciocho bits, por ser una palabra grande no fue práctico enviarla directamente a las entradas del microcontrolador, para solucionar este inconveniente se utilizó una memoria 27C4001 4 Mbit (512Kb x 8) UV EPROM la cual permitirá además de reducir la palabra implementar códigos para distintas combinaciones de los fototransistores dando gran flexibilidad al sensor y agrega la posibilidad al robot móvil de no solo ser usado en competencias de velocidad sino también en otros tipos de competencia en donde la trayectoria a seguir es discontinua y tiene bifurcaciones y/o en donde la trayectoria tiene códigos escritos dentro de la misma línea que el robot deberá seguir.

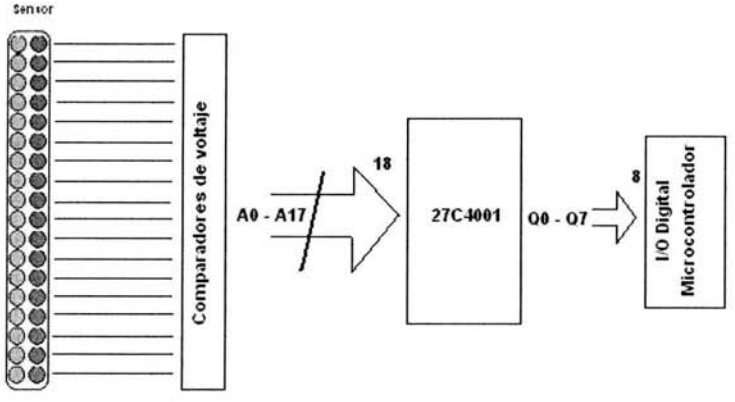


Figura 3.5 Diagrama del funcionamiento en modo digital del sensor

La memoria tiene diecinueve pines de dirección, dieciocho de las cuales se conectan a las salidas de los comparadores (Fig. 3.5), la salida de la memoria dependerá de lo programado en la localidad direccionada por el sensor, el tiempo de acceso a la localidad de memoria direccionada es de 120 [ns] el cual es suficientemente rápido y no afecta el tiempo de respuesta del sensor, así tenemos la siguiente tabla para programar la memoria y configurar el sensor para el robot móvil de velocidad.

Sensores																		Dirección	Salida	Salida
S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	De memoria	Dec.	Hex.
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	**0 0 0 0 1**	1	**1**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	**0 0 0 0 3**	2	**2**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	**0 0 0 0 7**	3	**3**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	**0 0 0 0 6**	4	**4**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	**0 0 0 0 E**	5	**5**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	**0 0 0 0 C**	6	**6**

0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	**0**	**0**	**0**	**1**	**C**	7	**7**
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	**0**	**0**	**0**	**1**	**8**	8	**8**
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	**0**	**0**	**0**	**3**	**8**	9	**9**
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	**0**	**0**	**0**	**3**	**0**	10	**A**
0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	**0**	**0**	**0**	**7**	**0**	11	**B**
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	**0**	**0**	**0**	**6**	**0**	12	**C**
0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	**0**	**0**	**0**	**E**	**0**	13	**D**
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	**0**	**0**	**0**	**C**	**0**	14	**E**
0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	**0**	**0**	**1**	**C**	**0**	15	**F**
0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	**0**	**0**	**1**	**8**	**0**	16	**10**
0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	**0**	**0**	**3**	**8**	**0**	17	**11**
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	**0**	**0**	**3**	**0**	**0**	18	**12**
0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	**0**	**0**	**7**	**0**	**0**	19	**13**
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	**0**	**0**	**6**	**0**	**0**	20	**14**
0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	**0**	**0**	**E**	**0**	**0**	21	**15**
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	**0**	**0**	**C**	**0**	**0**	22	**16**
0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	**0**	**1**	**C**	**0**	**0**	23	**17**
0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	**0**	**1**	**8**	**0**	**0**	24	**18**
0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	**0**	**3**	**8**	**0**	**0**	25	**19**
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	**0**	**3**	**0**	**0**	**0**	26	**1A**
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	**0**	**7**	**0**	**0**	**0**	27	**1B**
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	**6**	**0**	**0**	**0**	28	**1C**
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	**E**	**0**	**0**	**0**	29	**1D**
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**	**C**	**0**	**0**	**0**	30	**1E**
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**C**	**0**	**0**	**0**	31	**1F**
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**8**	**0**	**0**	**0**	32	**20**
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**3**	**8**	**0**	**0**	**0**	33	**21**
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**3**	**0**	**0**	**0**	**0**	34	**22**
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**	**0**	**0**	**0**	**0**	35	**23**

En la tabla los "*unos*" representan a los fototransistores que se encuentran sobre la línea blanca a seguir y los *"ceros"* representan a los que se encuentran sobre la superficie negra, se presentan todos los valores posibles que tendrá como entrada la memoria. El número de fototransistores que *vean* la línea en algún instante de tiempo dependerá del grosor de la misma. La línea en las competencias de velocidad oscila entre 1.5 [cm] y 2.2 [cm] por lo que como se muestra en la tabla en algunas ocasiones dos fototransistores detectarán la línea y en otros serán tres.

Por ejemplo si a la entrada del sensor tenemos lo siguiente:

|0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0|

Se direccionará la localidad 00300h de la memoria por lo que el microcontrolador leerá un valor de 12h por el puerto de entrada del sensor. Esto significa que el robot móvil se encuentra centrado sobre la línea blanca.

3.1.1.2 Sensado analógico.

Para obtener una salida analógica del sensor se utilizaron sólo cuatro fototransistores de los dieciocho que tiene el sensor, la idea básica se basó en el artículo "*A lateral Position Sensing System for Automated Vehicle Following" by Alleyne, Williams and DePoorter* [7]. Este artículo describe cómo las distribuciones de la respuesta característica de las fotorresistencias pueden ser ordenados secuencialmente para obtener una salida lineal, las fotorresistencias y los fototransistores tienen una distribución de salida muy parecidas por lo que se puede aplicar el análisis descrito a ambos dispositivos. El procedimiento envuelve una gran cantidad de cálculos analíticos para determinar el espacio que debe haber entre las fotorresistencias y la ganancia apropiada de cada una, sin embargo nosotros no seguimos estrictamente esa aproximación y más bien lo realizamos empíricamente. A continuación se describe el procedimiento seguido.

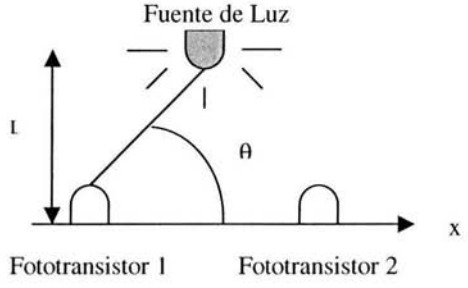


Figura 3.6 Angulo de incidencia de la luz sobre el fototransistor

Conforme una fuente de luz es movida lateralmente frente a un fototransistor, el ángulo de incidencia cambia desde 0 hasta 180 grados como se muestra en la figura 3.6, causando una variación en la corriente de colector del fototransistor. Cuando se tienen ángulos de incidencia pequeños, o cuando la fuente de luz se aleja a la derecha del fototransistor 1 la corriente disminuye. Esta corriente aumenta conforme la fuente de luz se mueve hacia la izquierda y es máxima cuando la fuente de luz está alineada con el fototransistor ($\theta=\pi/2$), si la fuente se mueve más a la izquierda la corriente empieza a disminuir nuevamente. Este mismo principio se puede aplicar a nuestro sensor según se muestra en la figura 3.7.

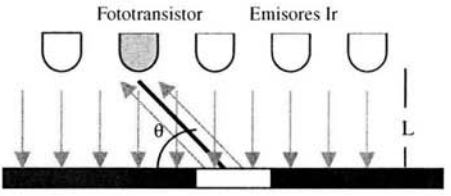


Figura 3.7 Angulo de incidencia de la luz sobre el fototransistor en el sensor de línea

Cuando el fototransistor está alineado con la línea blanca que determina la trayectoria la corriente de colector será máxima, conforme el fototransistor se aleja a la izquierda o a la derecha de la línea la corriente disminuirá. Si medimos la salida de un fototransistor en configuración colector común (misma configuración que utilizamos para los fototransistores del sensor) cuando se mueve de izquierda a derecha sobre la línea blanca que determina la trayectoria del robot móvil, obtenemos una distribución como la de la figura 3.8, el voltaje de salida depende de la distancia longitudinal, L, en este caso L = 2 [cm], así como de la distancia lateral x.

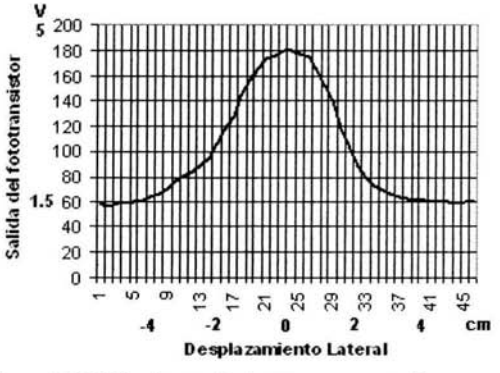


Figura 3.8 Salida de uno de los sensores cuando se mueve de izquierda a derecha sobre la línea blanca

Para obtener una salida lineal del sensor, los fototransistores necesitan tener una salida individual parecida a las curvas en la figura 3.9; *S1, S2, S3 y S4* representan las distribuciones de cada uno de los cuatro fototransistores. Para obtener una respuesta como la de la figura 3.9, cada uno necesita ser multiplicado por un peso, *S1* y *S2* por un peso positivo, w y x, *S3* y *S4* por un peso negativo, y y z.

Escogiendo un peso apropiado la suma de las distribuciones producirán teóricamente una relación lineal entre la salida del sensor y la ubicación sobre la línea blanca. La figura 3.10 muestra la suma de las distribuciones cada una con su peso. La ecuación de salida del sensor quedaría de la siguiente forma: [7].

$$Salida = w * S1 + x * S2 - y * S3 - z * S4 \quad \text{.....(3.2)}$$

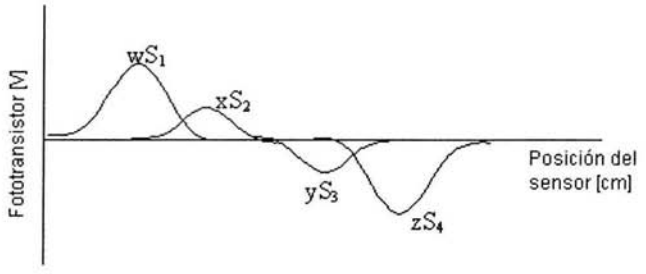


Figura 3.9 Salida de los los fototransistores con sus respectivos pesos.

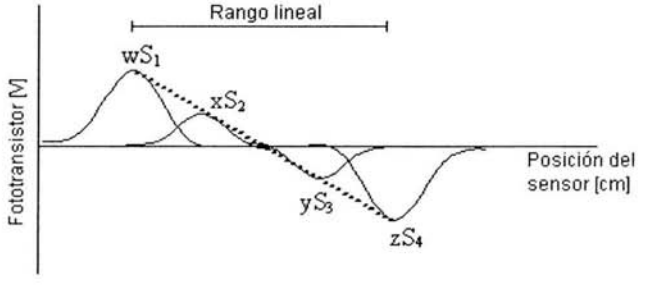


Figura 3.10 Suma de las distribuciones de los fototransistores

Tomando en cuenta la gráfica de la figura 3.8 se decidió que la separación más apropiada para los cuatro sensores era de cuatro centímetros. Para lograr obtener los *pesos* adecuados se aseguró que cada fototransistor tuviera la misma salida cuando estaba colocado en el centro de la línea blanca que determina a la trayectoria, los fototransistores se calibraron utilizando el potenciómetro del emisor, hasta obtener 4.5 [V] para cada uno.

Para iniciar el proceso de determinación de los *pesos* de cada sensor, primero se multiplicó cada uno por otro conjunto de pesos a, b, c y d, así la ecuación de salida del sensor quedaría de la siguiente forma:

$$Salida = w(aS1) + x(bS2) - y(cS3) - z(dS4) \quad \text{.....(3.3)}$$

Los pesos a, b, c y d fueron encontrados de la siguiente forma. Usando el microcontrolador se escogió un valor de 80h para ser el valor que cada sensor debería entregar cuando se encontraba sobre la línea. Con esta información cada salida de los fototransistores fue multiplicada por un peso tal que cada uno diera el valor 80h cuando estaba sobre la línea.

Los pesos w, x, y, y z fueron escogidos por el método de prueba y error. Para iniciar el proceso todos fueron escogidos arbitrariamente tal que cayeran sobre una línea recta, así se les asignaron los valores de $S1=2$, $S2=1$, $S3=1$, y $S4=2$. La salida total del sensor fue registrada cuando cada uno de los fototransistores se colocaba sobre la línea. Estos valores eran graficados para verificar que los puntos cayeran sobre una recta, si no era así los pesos eran movidos dependiendo de la forma de la gráfica hasta lograr que estuvieran sobre la recta (Fig. 3.11 a y b). Una vez logrado esto el sensor se recorrió de izquierda a derecha sobre la línea registrando los valores entregados cada 3[mm], se obtuvo una salida como la mostrada en la figura 3.12, en esta gráfica se observa que para los dos fototransistores centrales se obtuvo una salida casi lineal pero para los de los extremos no fue así, por lo que se decidió recorrer dos centímetros hacia el centro los sensores S1 y S4, repitiendo el procedimiento descrito se volvieron a obtener los nuevos pesos (S1=2.5, S2=1.3, S3=1.15, S4=2.3) y la gráfica obtenida fue la mostrada en la figura 3.13 la cual muestra una salida casi lineal lo cual es suficiente para nuestros propósitos.

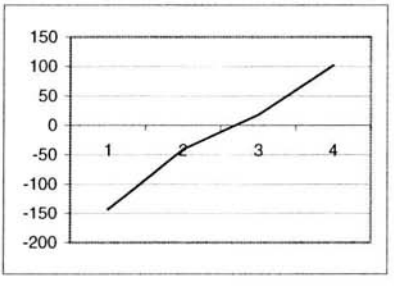


Figura 3.11a Salida de los fototransistores S1=2, S2=1, S3= 1, y S4=2

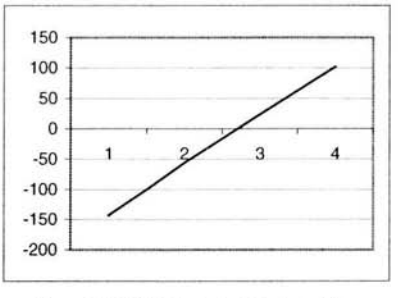


Figura 3.11b Salida de los fototransistores S1=2, S2=1.3, S3= 1.15, y S4=2.

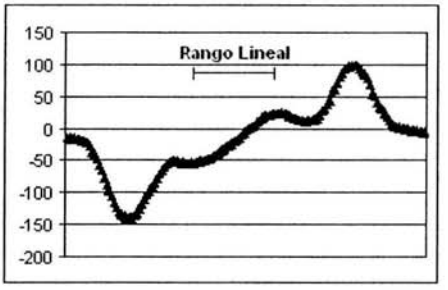


Figura 3.12 Salida del sensor **S1=2, S2=1.3, S3= 1.15,** *y* **S4=2.**

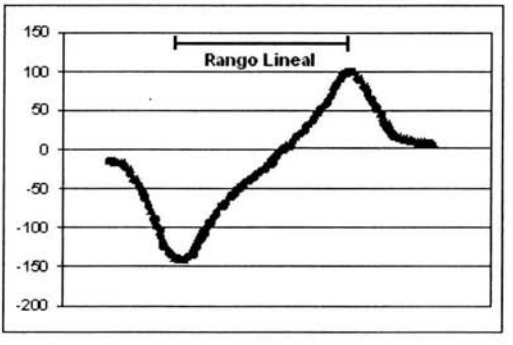


Figura 3.13 Salida del sensor S1=2.5, S2=1.3, S3=1.15, S4=2.3

Así la configuración final del sensor en modo analógico (Fig. 3.14) fue la siguiente:

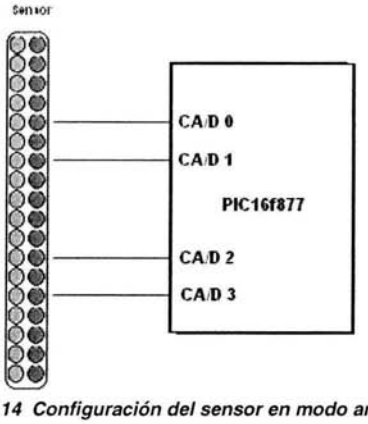


Figura 3.14 Configuración del sensor en modo analógico

3.1.2 Sensor de velocidad.

Para sensar la velocidad de las llantas se implementaron encoders ópticos incrementales en cada una de ellas los cuales miden las velocidades rotacionales de las llantas. El encoder utiliza el mismo principio de funcionamiento del sensor de línea; se diseñó un disco dividido en 64 secciones blancas y negras iguales como se muestra en la figura 3.15.

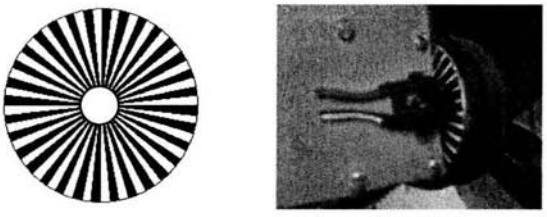


Fig. 3.15 Encoder incremental

Al hacer incidir un haz infrarrojo sobre el disco la energía reflejada sobre un fototransistor funcionando en saturación y corte da a la salida un tren de pulsos a una frecuencia proporcional a la velocidad de rotación de la llanta. Para esta etapa se utilizaron dos sensores optoreflectivos OPB703 los cuales contienen en un solo encapsulado al emisor y receptor infrarrojos, también se utilizó un comparador LM339 para convertir la señal entregada por el OPB703 a un valor digital.

Para acondicionar la señal de entrada al microcontrolador se utiliza un circuito convertidor frecuencia a voltaje LM2917 (Fig. 3.16), el cual entrega un voltaje proporcional a la frecuencia de entrada dado por la siguiente ecuación:

$$V_{out} = f_{in} * V_{CC} * R_1 * C_1 \quad \text{.....(3.4)}$$

donde:

V_{out} = Voltaje de salida.
f_{in} = Frecuencia de entrada.
R_1 = Resistencia de carga.
V_{CC} = Voltaje de alimentación.
C_1 = Capacitor de temporizado.

La salida del convertidor de frecuencia a voltaje es leída por el convertidor analógico digital del microcontrolador.

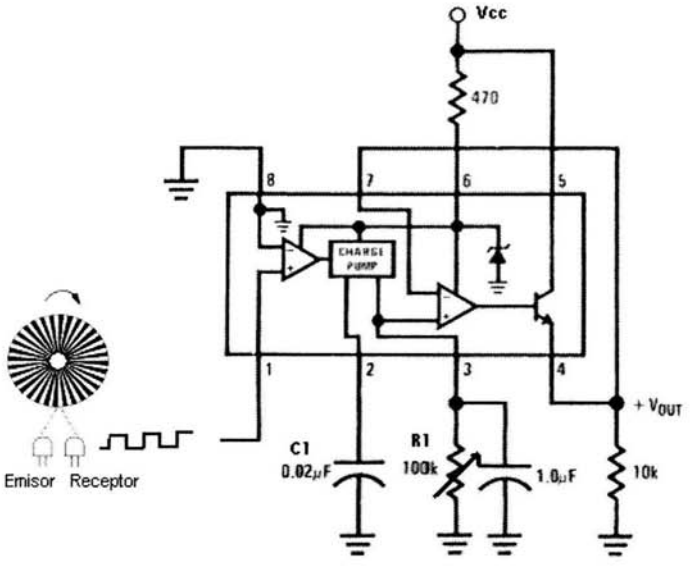


Figura 3.16 Diagrama del convertidor frecuencia-voltaje

3.2 Etapa de Potencia.

Un microcontrolador no puede manejar directamente un motor, ya que no puede suministrar suficiente corriente. Se deben tener circuitos, tal que, el poder del motor sea suministrado desde otra fuente y únicamente el control de las señales derive desde un microcontrolador. Dichos circuitos pueden ser implementados con una variedad de tecnologías, tales como relevadores, transistores bipolares, MOSFETS, y circuitos integrados dedicados específicamente para el control de motores como los LM18200, L293 y L298.

Para diseñar la etapa de potencia de los motores se buscaron circuitos capaces de manejar bajos voltajes en la carga y corrientes de hasta 2 [A], en primera instancia se pensó en el circuito LM18200 ya que soporta una corriente de 3 [A], pero al verificar el voltaje mínimo de la carga se encontró que son 12 [V], al hacer pruebas con este circuito verificamos que el voltaje mínimo que necesita es de 10.5 [V] lo cual sobrepasa por mucho el voltaje máximo de los motores que es 5[V]. Por esto se decidió usar el circuito L298 el cual contiene cuatro switches tipo *push – pull* que pueden ser usados independientemente como dos *Puentes H.* Cada canal es controlado por niveles de voltaje TTL y cada par de drivers está equipado con una entrada *enable* el cual controla un *Puente H* completo. El circuito tiene entradas de voltaje separadas para la etapa de control y para la etapa de potencia. El voltaje mínimo para la etapa de potencia es de Vcc+2.5 [V].

El circuito soporta cargas en total de hasta 2 [A] por canal, debido a que los motores usados en ocasiones demandan 2.2 [A], el circuito genera demasiado calor, por lo que se aumentó la capacidad de corriente del driver conectando en paralelo los dos *Puentes H* del circuito. Fig. (3.14 a). Al conectar en paralelo los puentes se logra obtener un driver con capacidad de soportar hasta 3.5 Amp. [8]. La fotografía (Fig. 3.14b) muestra el arreglo de los dos *Puentes H* con su disipador de calor.

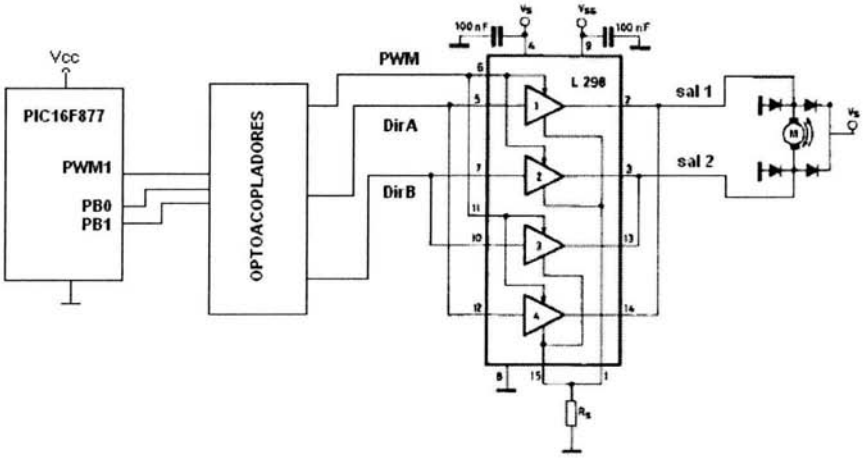


Figura 3.14 a) Diagrama eléctrico de la etapa de potencia de uno de los motores.

Figura 3.14 b). Fotografía de los circuitos L298 con su disipador de calor.

La siguiente tabla muestra las distintas combinaciones que se presentan en el funcionamiento del puente H.

DirA	DirB	Sentido de la llanta
0	0	Libre
0	1	Adelante
1	0	Atrás
1	1	Freno

Una de las desventajas que se observaron en los microcontroladores PIC es su muy baja inmunidad al ruido electromagnético generado principalmente por los motores, para evitar mal funcionamiento del robot por esta causa y por seguridad se separaron las fuentes de alimentación de la etapa de control y la etapa de potencia, para esto se utilizaron circuitos opto - acopladores H11L1 los cuales garantizan un tiempo de switcheo menor a 4 [us] parámetro importante ya que este circuito se utilizará en la salida del PWM del microcontrolador y en los bits de dirección del motor. La configuración utilizada se muestra en la figura 3.15.

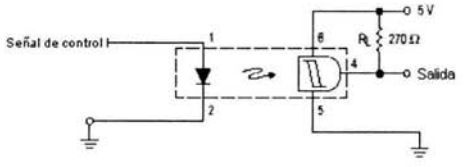


Figura 3.15 Configuración usada para el opto acoplador H11L1

Con esta configuración se logra que las tierras de los circuitos de control y sensado queden totalmente separadas.

3.3 Electrónica del microcontrolador.

El microcontrolador a utilizar debe tener las características necesarias para la implementación del algoritmo de control, además de contar con la capacidad de recibir y procesar las señales de control y generar las salidas. Existe una gran variedad de microcontroladores en el mercado que se utilizan en este tipo de aplicaciones, la familia de microcontroladores PIC de Microchip son microcontroladores fáciles de conseguir y relativamente económicos, además de tener una gran cantidad de opciones en cuanto a memoria y periféricos.

MODELO		PIC16F877
MEMORIA PROG.	Bytes	14336
	Palabras	8192X14
MEMORIA DATOS	Bytes EEPROM	256
	Bytes RAM	368
C A/D		8(10 bits)
BOD (Detección de baja tensión)		si
LINEAS E/S		33
COMUNICACIÓN SERIE		USART/ MSSP
CCP		2
TEMPORIZADORES		1-16 bit, 2-8 bit, 1-WDT
FREC. MAX. MHz		20
ICSP (Programación Serie en Circuito)		si
ENCAPSULADOS		40P,44L,44PQ,44PT
FUENTES DE INTERRUPCIÓN		14
COMUNICACIÓN PARALELO		si

Figura 3.16 Características del PIC16F877

Los microcontroladores PIC requieren un sistema mínimo para funcionar. Microchip proporciona un software gratuito para desarrollar proyectos, el cual incluye desde un compilador hasta un ambiente de simulación y emulación (Apéndice B). Se seleccionó el microcontrolador PIC16F877, el cual es un microcontrolador de gama media, cuyas características se muestran en la figura 3.16, como podemos ver, cuenta con una serie de periféricos que nos facilitan el recuperar las señales de entrada y generar las salidas.

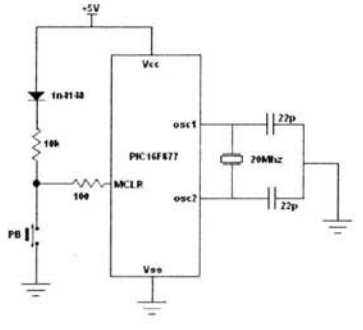


Figura 3.17 Sistema mínimo del PIC16F877

El diseño de la tarjeta electrónica para este microcontrolador es muy sencillo y no requiere de una gran cantidad de componentes, la figura 3.17 muestra el diseño de la tarjeta electrónica realizada para el microcontrolador, adicionalmente se realizó una interfaz de comunicación serie con un circuito convertidor de niveles de voltaje TTL a niveles RS232 (max232), para poder comunicar el PIC con la computadora. Fig. 3.18.

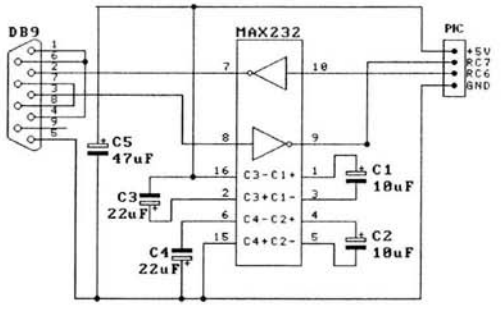


Figura 3.18 Módulo de comunicación RS232

El microcontrolador funciona con un cristal oscilador externo de 20 [Mhz]. lo cual da un ciclo de instrucción de 5 [Mhz] ya que el microcontrolador divide internamente la frecuencia del oscilador entre cuatro.

En la tarjeta diseñada para el microcontrolador se colocaron disponibles todos los puertos del microcontrolador con el fin de poder anexar sensores si se desea o algún otro tipo de periféricos al robot móvil; el diagrama completo se muestra en el apéndice E. La programación del microcontrolador se lleva a cabo mediante un bootloader cargado en la memoria del microcontrolador, este es un pequeño programa que permite al microcontrolador auto programar su memoria flash con ayuda de un sencillo programa (Pic downloader) en la PC, cuyas características se explican en el apéndice B

3.4 Baterías y fuente de alimentación.

Una vez terminados e interconectados los circuitos del robot móvil se midió el consumo de corriente de la etapa de control y de potencia dando los siguientes resultados.

Etapa de control	200 [mA]
Etapa de potencia	max 2.2 [A]

Las baterías se clasifican de acuerdo a diferentes parámetros. Por un lado, se llaman primarias si no se pueden recargar o secundarias, si se pueden recargar, y por otro lado se clasifican también según los materiales de los cuales están fabricados sus electrodos, lo que les otorga sus principales características de voltaje, potencia, duración, tamaño, etc. Es muy importante conocer las características generales de todos los tipos con el fin de utilizar el más adecuado según la aplicación, tamaño del robot, duración esperada, costo, facilidades de recarga, etc. A continuación se presentan características generales de algunas de

las baterías recargables que más se utilizan en la robótica y que fueron consideradas antes de decidir que tipo de baterías serían usadas en el robot..

Las tecnologías de las pilas más usadas en la actualidad son las de *plomo – ácido, las de níquel – cadmio (NiCd), las de níquel – metal híbridas (NiMH) y las de litio – iónico (li-Ion).* Las baterías de plomo ácido, proporcionan un voltaje nominal de 2[V] / celda. Se fabrican típicamente con voltajes de 2[V], 4[V], 6[V], 12[V], su capacidad puede llegar a ser muy alta hasta 500[Ah] o más, por su capacidad de corriente y su facilidad de recarga son unas de las baterías más utilizadas en la robótica sin embargo la desventaja de este tipo de baterías es su peso ya que una batería de 12 [V] y 0.8 [Ah] llega a pesar hasta 600 [gr], lo cual representaría una gran carga para el robot de velocidad.

Las baterías de níquel - cadmio proporcionan un voltaje nominal de 1.2 [V] /celda. Eran muy utilizadas en teléfonos celulares, computadoras, cámaras de video, etc. Algunas unidades industriales pueden tener capacidades de hasta 250 [Ah]. Existen unidades comerciales de 1.2 [V] / 600 [mAh], vienen en una gran variedad de tamaños de hasta 7.2 [V], sin embargo son relativamente más costosas y con frecuencia presentan un "efecto memoria" el cual les impide desarrollar su plena capacidad cuando se descargan cíclicamente hasta un mismo valor y luego se recargan.

Las celdas de litio – ion proporcionan un voltaje nominal de 3.6 [V]/ celda, inicialmente fueron desarrolladas para la impulsión de vehículos eléctricos, son relativamente nuevas en el mercado sin embargo están siendo utilizadas ampliamente en lugar de los otros tipos de baterías, los equipos que más las utilizan son las computadoras, teléfonos, videocámaras, esto se debe a su gran capacidad de corriente y a su reducido tamaño y peso, sin embargo su costo actualmente es alto. Se pueden encontrar en el mercado baterías cuadradas tipo 9V con una corriente de 1200 [mAh] no recargables con un costo de $140.

Las baterías de níquel metal (Ni-Mh), proporcionan un voltaje nominal de 1.2[V] / celda, vienen en las mismas presentaciones que las baterías NiCd y se identifican de la misma forma. Sin embargo, tienen capacidad energética muy superior, existen en el mercado celdas de 1.2 [V] / 2200 [mAh] con un precio de $30. Además pueden soportar gran número de ciclos de carga y descarga (superior a 1000), su recarga es muy sencilla y oscila entre 1 y 14 [hrs.] por celda; su precio es bastante accesible y el peso relativamente bajo, una celda de 1.2 [V] tipo AA pesa aproximadamente 20 [gr].

Por las características antes descritas y por el consumo de corriente medido en el robot, se decidió utilizar baterías tipo (Ni-Mh), como ya se dijo estas tienen un voltaje nominal de 1.2 [V]/celda, para obtener el voltaje necesario para la etapa de potencia de los motores (aproximadamente 7 [V]) se conectaron 6 celdas tipo AA en serie, en esta configuración la corriente de la batería permanece igual a la de cada celda individual, hay que tomar en cuenta que todas las celdas deben ser de la misma capacidad de corriente en [mAh] de lo contrario habrá alguna que se

descargue más que la otra y hará que el voltaje baje antes de los esperado, la capacidad de corriente de cada celda es de 1800 [mAh], lo cual permite al robot funcionar correctamente un tiempo de 40 minutos ya que como vimos anteriormente su consumo máximo en promedio son 2.2 [A]. La etapa de control utiliza 2 baterías NiMh de 120 [mAh] y 8.4 [V] cada una, conectadas en paralelo para obtener una capacidad de 240 [mAh].

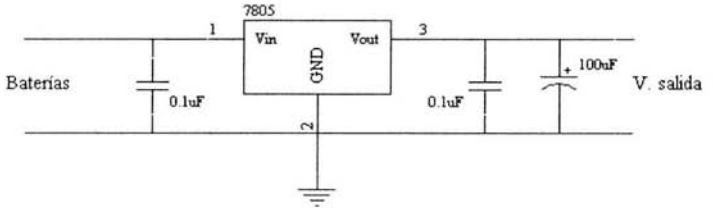
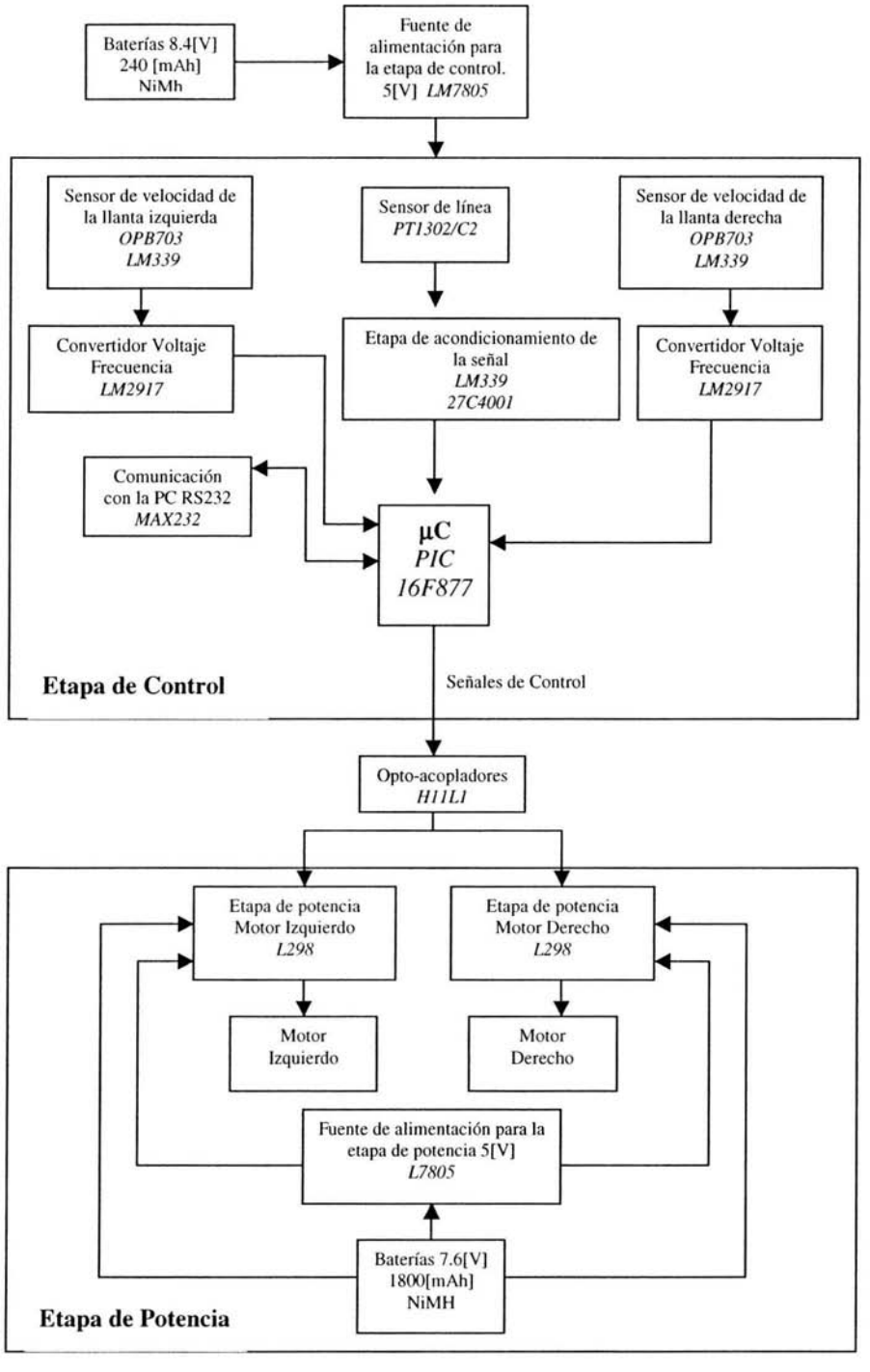


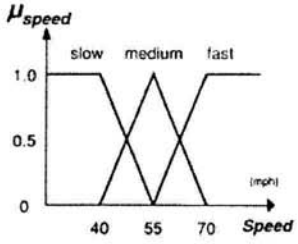
Figura 3.19 Configuración de los los reguladores de voltaje 7805 y 70L05

Las baterías se van descargando a medida que trabaja el robot. Esta descarga depende principalmente de la carga consumida y de la capacidad de la batería. Por otro lado, sabemos que en el arranque de un motor se consume una alta corriente lo que hace bajar súbitamente el valor del voltaje general de la alimentación. Los reguladores de voltaje se encargan de mantener el voltaje de salida constante aún con fuertes variaciones en la carga, los cuales se pueden utilizar para alimentar los circuitos electrónicos del robot. En el caso de los motores decidimos alimentarlos directamente con las baterías ya que regularles el voltaje sería muy complicado debido a su alto consumo de corriente, además de que en el proceso de regulación se pierde una gran cantidad de energía transformada en calor lo que haría muy ineficiente al robot. El circuito LM7805 es uno de los más utilizados en la electrónica actualmente, en la figura 3.19 se muestra su configuración, este circuito recibe un voltaje que puede variar entre 7 y 26 [V] y entrega a la salida un voltaje estable de 5 [V] y tiene capacidad de entregar hasta 1[A]. Su conexión es bastante sencilla ya que no requiere de componentes externos más que algunos capacitores para filtrar la señal y de un disipador de calor para evitar sobrecalentamiento. Se construyeron dos módulos como estos, el que alimentará los circuitos se construyó con el circuito LM7805, el módulo que alimentará la lógica del circuito de potencia se construyó con un 78L05 el cual solo soporta una carga máxima de 100 [mA], lo cual es suficiente para esta etapa.

3.5 Configuración electrónica final.

La configuración final del robot móvil quedó de la siguiente forma:


Baterías 8.4[V]
240 [mAh]
NiMh
Fuente de alimentación para la etapa de control. 5[V] LM7805
Sensor de velocidad de la llanta izquierda
OPB703
LM339
Sensor de línea
PT1302/C2
Sensor de velocidad de la llanta derecha
OPB703
LM339
Convertidor Voltaje Frecuencia
LM2917
Etapa de acondicionamiento de la señal
LM339
27C4001
Convertidor Voltaje Frecuencia
LM2917
Comunicación con la PC RS232
MAX232
µC
PIC
16F877
Etapa de Control
Señales de Control
Opto-acopladores
H11L1
Etapa de potencia Motor Izquierdo
L298
Etapa de potencia Motor Derecho
L298
Motor Izquierdo
Motor Derecho
Fuente de alimentación para la etapa de potencia 5[V]
L7805
Baterías 7.6[V]
1800[mAh]
NiMH
Etapa de Potencia



4. Diseño del sistema de control

4.1 Generalidades del control difuso.

Los métodos de control y algoritmos difusos pueden ser considerados como un tipo de control inteligente. Esto es porque el modelado, análisis y control difuso involucran cierto conocimiento humano. Comparado con aproximaciones convencionales, los sistemas difusos utilizan más información conocida por los expertos y delegan menos sobre los modelados matemáticos del sistema físico.

Podemos decir que es adecuado usar tecnología difusa en los siguientes casos:

- En procesos complejos, si no existe un modelo de solución sencillo.
- En procesos no lineales.
- Cuando haya que introducir la experiencia de un operador "experto" que se basa en conceptos imprecisos obtenidos de su experiencia.
- Cuando ciertas partes del sistema a controlar son desconocidas y no pueden medirse de forma fiable (con errores posibles).
- Cuando el ajuste de una variable puede producir el desajuste de otras.
- En general, cuando se quieran representar y operar con conceptos que tengan imprecisión o incertidumbre.

El control difuso provee una metodología formal para representar, manipular e implementar conocimiento heurístico humano sobre cómo controlar el sistema.

En la figura 4.1 podemos observar el diagrama de bloques de un controlador difuso, en donde vemos un contralor embebido en un sistema de lazo cerrado. Las salidas de la planta están representadas por y(t), las entradas por u(t), la entrada de referencia al controlador difuso esta representada por r(t). Fig. 4.1.

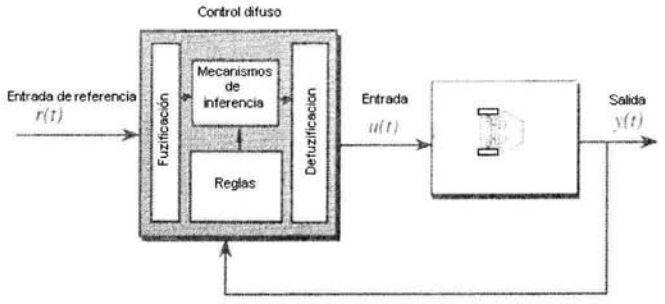


Figura 4.1: Diagrama de bloques de un controlador difuso

Un controlador difuso tiene cuatro partes principales:

1. La **base de reglas** contiene el conocimiento, en forma de un conjunto de reglas *Si - Entonces*, de cual es la mejor manera de controlar el sistema.

2. El **mecanismo de inferencia** evalúa cual regla de control es relevante en cada tiempo y decide que entrada debería tener la planta.

3. La interfase de **fusificación** modifica las entradas para que puedan ser interpretadas y comparadas con las reglas.

4. Y la interfase de **defusificación** convierte la conclusión obtenida por el mecanismo de inferencia a entradas validas para la planta.

Básicamente, deberíamos ver al controlador difuso como un sistema de decisión artificial que opera en un sistema de lazo cerrado en tiempo real.

Para diseñar un controlador difuso, el ingeniero de control debe tener suficiente información sobre como debería actuar el sistema de decisión artificial en el sistema de lazo cerrado. Algunas veces esta información proviene de un operador humano encargado de tomar las decisiones de control, o el ingeniero de control después de entender la dinámica de la planta escribe un conjunto de reglas sobre como controlar el sistema sin ayuda externa.

Un conjunto de reglas "*Si - Entonces*" es cargado en la "base de reglas" y se escoge una estrategia de inferencia, entonces el sistema está listo para ser probado y verificar si las especificaciones de lazo cerrado se cumplen.

Esta pequeña descripción provee un panorama general del diseño de un control difuso.

4.1.1 Variables, valores y reglas lingüísticas.

Un sistema difuso tiene un mapeo estático no lineal entre entradas y salidas. Esto es asumiendo que el sistema difuso tiene entradas $u_i \in U_i$ donde $i = 1,2,3,...,n$ y salidas $y_i \in Y_i$ donde, como se muestra en la figura 4.2. Las entradas y salidas son "crisp" (esto es números reales), no valores difusos. El bloque de defusificación convierte las entradas crisp a valores difusos, el mecanismo de inferencia usa las reglas difusas en la base de reglas para producir conclusiones difusas, y el bloque de defusificación convierte esas conclusiones difusas de nuevo a salidas crisp.

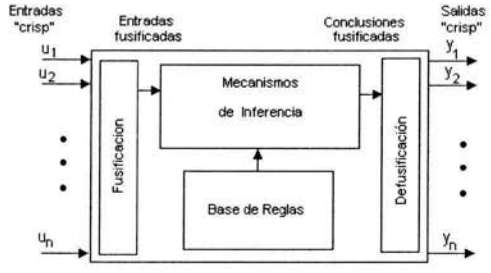


Figura 4.2: Entradas y salidas de una controlador difuso

Comúnmente el conjunto U_i y Y_i son llamados "universo discurso" para u_i y y_i respectivamente. En aplicaciones practicas, frecuentemente los universos discursos son simplemente conjuntos de números reales o algunos intervalos o subconjuntos de números reales.

Para especificar las reglas en la base de reglas, el experto usa una "descripción lingüística". Por lo tanto se necesitan expresiones lingüísticas para las entradas y salidas y sus características. Para un sistema difuso general, la variable difusa $\tilde{u}_i$ será usada para describir las entras u_i, de igual forma la variable lingüística $\tilde{y}_i$ será usada para describir las salidas y_i.

Tal como u_i y y_i toman valores sobre los universos discursos U_i y Y_i respectivamente, las variables lingüísticas $\tilde{u}_i$ y $\tilde{y}_i$ toman valores lingüísticos que son usados para describir características de las variables. Sea $\tilde{A}_i^j$ el jotaésimo valor lingüístico $\tilde{u}_i$, definido sobre el universo discurso U_i. Si asumimos que existen muchos valores lingüísticos definidos sobre U_i, entonces la variable lingüística $\tilde{u}_i$ toma los elementos del conjunto de valores lingüísticos denotados por:

$$\tilde{A}_i = \left\{ \tilde{A}_i^j : j = 1,2,...,N_i \right\} \quad (4.1)$$

Similarmente, $\tilde{B}_i^j$ denota el jotaésimo valor lingüístico $\mathfrak{y}_i$, definido sobre el universo discurso Y_i. La variable lingüística $\mathfrak{y}_i$ toma los elementos del conjunto de valores lingüísticos denotados por:

$$\tilde{B}_i = \left\{ \tilde{B}_i^p : p = 1,2,...,M_i \right\} \quad (4.2)$$

Los valores lingüísticos generalmente describen términos como "*positivo grande", "cero", "pequeño", etc.*

El mapeo de las entradas y las salidas para un sistema difuso es en parte caracterizado por un conjunto de reglas condición → acción de la forma:

Si *premisa* **Entonces** *consecuente*

Las entradas de un sistema difuso están asociadas con la premisa y las salidas están asociadas con el consecuente. Para un caso con *n* entradas y dos salidas la forma de las reglas es la siguiente:

Si $\mathfrak{u}_1$ **es** $\tilde{A}_1^j$ **y** $\mathfrak{u}_2$ **es** $\tilde{A}_2^k$ **y** ... **y** $\mathfrak{u}_n$ **es** $\tilde{A}_n^l$ **Entonces** $\mathfrak{y}_1$ **es** $\tilde{B}_1^r$ **y** $\mathfrak{y}_2$ **es** $\tilde{B}_2^s$ .....(4.3)

Los conjuntos difusos y la lógica difusa se usan para cuantificar heurísticamente el significado de las variables lingüísticas, valores lingüísticos y reglas lingüísticas que el experto especifica. El concepto de un conjunto difuso es introducido primero definiendo una "función membresía".

Sea U_i el universo del discurso y $\tilde{A}_i^j \in \tilde{A}_i$ un valor lingüístico específico para la variable lingüística $\mathfrak{u}_i$. La función $\mu(u_i)$ asociada con $\tilde{A}_i^j$ que mapea U_i para el intervalo [0,1] se llama "función membresía". Esta función describe la "certeza" para un elemento de U_i, llamado u_i, con una descripción lingüística $\mathfrak{u}_i$, tal vez clasificada lingüísticamente como $\tilde{A}_i^j$. Las funciones membresía son subjetivamente seleccionadas de acuerdo a la experiencia o intuición.

Es posible escoger muchas formas para la función membresía y estas darán diferentes significados para las variables lingüísticas que están cuantificando. Algunas formas típicas se muestran en la figura 4.3.

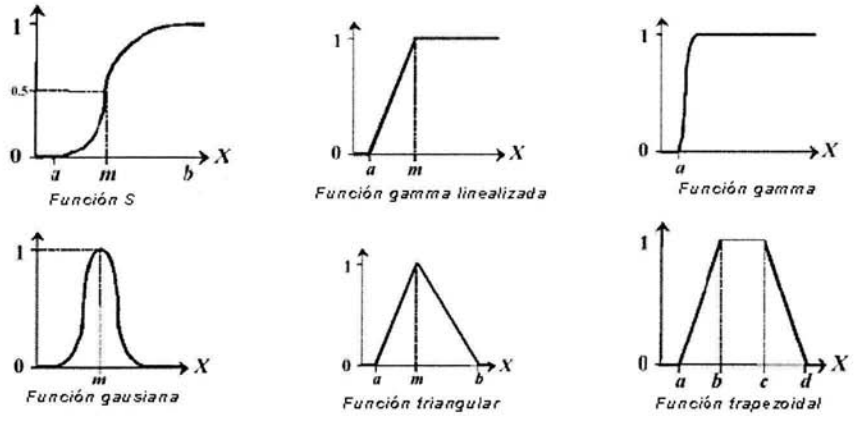


Figura 4.3: Formas típicas para las funciones membresía

Los dos paradigmas clásicos de control difuso son el enfoque Mandami y el de Takagi-Sugeno. El enfoque Mandami corresponde a un sistema de control difuso estándar, en el cual el consecuente de las reglas se expresa en términos lingüísticos (ecuación 4.3), asignando valores difusos tanto a las entradas como a las salidas.

En un sistema Takagi-Sugeno la forma de las reglas es:

Si $\tilde{u}_1$ **es** $\tilde{A}_1^j$ **y** $\tilde{u}_2$ **es** $\tilde{A}_2^k$ **y** ... **y** $\tilde{u}_n$ **es** $\tilde{A}_n^l$ **Entonces** $b_i = g_i(\cdot)$ .....(4.4)

Donde "▪" representa un argumento de la función g_i; y b_i son salidas no difusas. La premisa en las reglas esta definida de la misma forma que en un sistema estándar, el consecuente es lo que cambia. En lugar de contener términos lingüísticos con una función membresía asociada, en el consecuente se usa una función $b_i = g_i(\cdot)$ que no tiene asociada una función membresía. Se puede utilizar cualquier función para el consecuente. Para este tipo de sistemas difusos puede usarse una operación apropiada para representar la premisa (por ejemplo mínimo o producto), y la defusificación puede obtenerse usando:

$$y = \frac{\sum_{i=1}^{R} b_i \mu_i}{\sum_{i=1}^{R} \mu_i} \quad \text{.....(4.5)}$$

Donde μ_i representa el grado de aplicabilidad para cada regla, que se obtiene con los valores membresía de las entradas involucradas en cada premisa. Se debe tener en cuenta que $\sum_{i=1}^{R} \mu_i \neq 0$ para que la ecuación no se indetermine.

4.2 Controlador difuso del robot móvil.

El diseño del control difuso esencialmente involucra:

1) Escoger las entradas y salidas del controlador difuso.
2) Escoger el pre-procesamiento que es necesario para las entradas del controlador y el posible post-procesamiento necesario para las salidas.
3) Diseñar cada una de las cuatro partes del diagrama de bloques mostrado en la figura 4.1

Se eligió un sistema difuso Takagi-Sugeno para el control de robot.

4.2.1 Entradas y Salidas del controlador.

Para escoger las entradas y salidas del controlador, es necesario definir la información que se requiere para generar una o más salidas con las cuales controlemos el comportamiento del robot, necesitamos que el robot gire tanto a la derecha como a la izquierda o avance en línea recta. De acuerdo al diseño mecánico seleccionado y el tipo de locomoción elegido, la diferencia de velocidad entre las llantas, así como el sentido de giro de las mismas, determinan el tipo de movimiento que el robot ejecuta. Por tanto las variables de salida son dos señales que controlan las velocidades en cada una de las llantas. La desviación del robot respecto a la trayectoria es la información sobre la cual se decide la dirección y la velocidad del robot.

Por lo tanto se tienen 3 variables de entrada y dos de salida; las variables de entrada son DT que representa la desviación del robot, VD para la velocidad de la llanta derecha y VI para la velocidad de la llanta izquierda, las salidas son V_D y V_I. Las salidas corresponden a las velocidades de cada llanta la cual se expresara como un valor de PWM.

La lectura del sensor de línea como se especificó en el capítulo 3 es un valor que cambia dependiendo de la posición en la cual se encuentra la línea blanca respecto al sensor, entregando 1 en el extremo izquierdo y 35 en el extremo derecho. Para realizar el control se tomo en cuenta la variable $|DT|$ en vez de DT directamente, donde $|DT|$ corresponde al valor absoluto de la diferencia del valor medio (12h) y el valor leído. El valor medio es el valor que entrega el sensor cuando la línea se encuentra perpendicular al robot y exactamente en medio de este.

4.2.2 Fusificación.

Para cada una de las entradas es necesario definir una serie de conjuntos difusos a partir de etiquetas lingüísticas de acuerdo a la experiencia del experto. La experiencia de competencias anteriores nos da la referencia para seleccionar dichos conjuntos, así como la forma de la función membresía.

La entrada $|DT|$ está definida por el conjunto difuso $|DT|$ con 3 subconjuntos difusos con las siguientes etiquetas lingüísticas asociadas:

- Pequeño(TP),
- Medio (TM),
- Grande (TG)

Definidos por funciones membresía triangulares simétricas (Fig. 4.4).

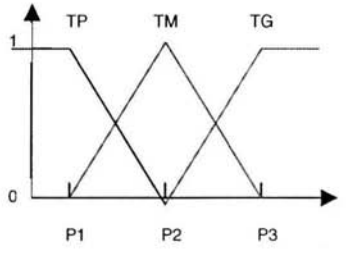


Fig. 4.4: Función membresía del conjunto DT (posición relativa del carro respecto a la pista)

Las otras dos entradas corresponden a las velocidades en cada llanta. El conjunto difuso que las define es el mismo conjunto V con dos subconjuntos difusos asociados igualmente definidos por funciones triangulares (Fig. 4.5) con las siguientes etiquetas lingüísticas:

- Grande (G),
- Pequeña (P)

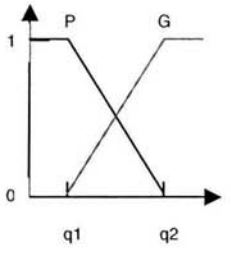


Fig. 4.5 Función de pertenencia para el conjunto V (velocidad de las llantas)

Puesto que se eligió una estructura tipo Takagi-Sugeno, en las reglas los valores en el consecuente son crisp y no difusos, por lo tanto no se requiere especificar un conjunto difuso para las salidas.

4.2.3 Reglas de control.

En un sistema Takagi-Sugeno como se mencionó antes el antecedente es una expresión lingüística mientras que en el consecuente se tiene una función casi siempre lineal para la salida en función de la entrada, se consideraron los principales casos, para plasmarse en las reglas.

El comportamiento que se desea del carro esta representado en el consecuente de las reglas, definimos un par de funciones (ecuación 4.7) en las cuales la diferencia entre las velocidades de las llantas así como el sentido de giro de éstas, fuera proporcional a la desviación del robot.

$$\begin{aligned} V_I &= v_i + DT * k_i \\ V_D &= v_i - DT * k_i \end{aligned} \qquad \text{.....(4.7)}$$

La estructura de las reglas es la siguiente:

Si DT=x **y** VI=y **y** VD=z **Entonces** $V_I = v_i + DT * k_i$ **y** $V_D = v_i - DT * k_i$

las variables v_i y k_i cambian dependiendo de la situación en que se encuentra el robot, el signo de DT es el que da el sentido del giro del robot, v_i es la velocidad a la que el robot debe ir en cada condición. El caso en que DT=0 implica que el robot deber ir derecho.

Puesto que para la función membresía se considera el valor absoluto de DT las reglas se repiten para cuando el robot gira a ambos sentidos resultando sólo 3 condiciones diferentes, dependiendo de que tanto se desvía el robot.

La *base de reglas* queda conformada por 6 reglas de la siguiente forma:

R1: **If** DT=G **and** VI=P **and** VD=G **Then**

$$V_I = 10 + DT * 1.2$$
$$V_D = 10 - DT * 1.2$$

R2: **If** DT=G **and** VI=G **and** VD=P **Then**

$$V_I = 10 + DT * 1.2$$
$$V_D = 10 - DT * 1.2$$

R3: **If** DT=P **and** VI=P **and** VD=G **Then**

$$V_I = 30 + DT * .9$$
$$V_D = 30 - DT * .9$$

R4: **If** DT=P **and** VI=G **and** VD=P **Then**

$$V_I = 30 + DT * .9$$
$$V_D = 30 - DT * .9$$

R5: **If** DT=M **and** VI=P **and** VD=G **Then**

$$V_I = 20 + DT * 1.4$$
$$V_D = 20 - DT * 1.4$$

R6: **If** DT=M **and** VI=G **and** VD=P **Then**

$$V_I = 20 + DT * 1.4$$
$$V_D = 20 - DT * 1.4$$

4.2.4 Mecanismos de inferencia.

El método de inferencia tiene básicamente dos tareas:

1) Determinar el grado de aplicabilidad de cada regla para la situación en curso caracterizada por las entradas u_i y,
2) Asignar un grado de certeza a la salida de cada regla de acuerdo a las entradas en curso y la información en la base de reglas (en el caso de un sistema tradicional difuso).

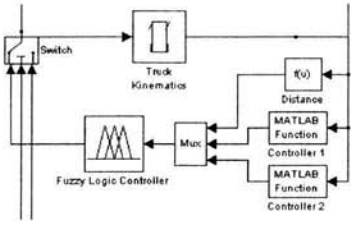
Para determinar el grado de aplicabilidad $\mu_{premisa}$ de cada regla se utiliza la operación difusa producto, definida de la siguiente forma:

$$\mu_{premisa}(u_1, u_2, ..., u_n) = \mu_{\hat{A}_1^j}(u_1) * \mu_{\hat{A}_2^k}(u_2) * ... * \mu_{\hat{A}_n^l}(u_n) \quad .....(4.8)$$

Donde $\mu_{\hat{A}_i^l}(u_i)$ corresponde al valor membresía para cada una de las entradas involucradas en la premisa.

4.2.4 Defusificación.

Estrictamente en esta etapa se debería pasar la conclusión en términos lingüísticos, obtenida en el proceso de inferencia, a valores "crisp" que puede entender la planta. Sin embargo como se mencionó en el apartado 4.1 por tratarse de un sistema Takagi-Sugeno aplicamos la ecuación 4.5 para obtener la conclusión, donde μ_i es $\mu_{premisa}$ para la iésima regla evaluada.



5. *Simulación del controlador*

Simulación es el proceso de diseñar y desarrollar un modelo computarizado de un sistema o proceso y conducir experimentos con este modelo con el propósito de entender el comportamiento del sistema o evaluar varias estrategias con las cuales se puede operar el sistema.

5.1 Modelo cinemático del robot

Para poder probar el comportamiento del controlador diseñado y hacer ajustes en él, antes de implementarlo en el robot, simulamos el controlador en Matlab, para ello lo primero que se requiere es un modelo aproximado de la planta a controlar, en nuestro caso el modelo cinemático del robot (Fig. 5.1).

El modelo cinemático directo proporciona una relación entre el sistema de coordenadas locales asociado al punto de guía del vehículo con respecto a otro global de trabajo.

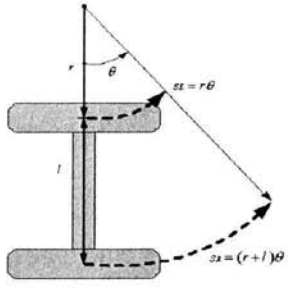


Figura 5.1: Esquema cinemático del robot

A continuación se obtiene un modelo cinemático elemental para el robot móvil. Este modelo no considera detalles físicos tales como motores y engranes ni tampoco considera la inercia y fricción sin embargo muestra una manera de obtener la trayectoria del robot.

Si nos referimos a la figura 5.1 obtenemos las siguientes relaciones:

$SL = r\,\theta$ ....(5.1)

$SR = (r + L)\,\theta$ ....(5.2)

$SM = (r + L/2)\,\theta$ ....(5.3)

En donde *SL* y *SR* proporcionan el desplazamiento (distancia recorrida) para las llantas izquierda y derecha respectivamente, *r* es el radio de curvatura para la llanta izquierda, *L* es la distancia entre las llantas (de centro a centro a lo largo del eje), y θ es el ángulo de la vuelta en radianes (θ radianes = θ grados (π / 180)). *SM* es la velocidad del centro del robot en el eje principal, en este desarrollo consideraremos al centro del robot como el origen del sistema de referencia de la simulación.

Nuestro problema principal es encontrar la trayectoria del robot conociendo las velocidades de giro de sus llantas, usaremos cinemática directa para predecir el comportamiento del sistema mecánico basado en las entradas del sistema.

En cualquier instante las coordenadas del centro del robot x y y cambian de acuerdo a su velocidad y orientación. Consideremos que la orientación está dada por el ángulo θ medido en radianes a partir del eje *x*. Recordando que el vector que describe la velocidad del robot está dado por $(\cos\theta, \operatorname{sen}\theta)$. Las coordenadas x y y del centro cambiarán dependiendo de la velocidad de su movimiento a lo largo de este vector, por lo que considerando a $m(t)$ y $\theta(t)$ como funciones dependientes del tiempo para la velocidad y orientación del robot, la solución debe ser de la forma:

$dx/dt = m(t)\cos(\theta(t))$ ....(5.4)

$dy/dt = m(t)\operatorname{sen}(\theta(t))$ ....(5.5)

Si observamos la figura 5.2, podemos notar que conforme el robot cambia su posición, todos los puntos del robot posiblemente estén en movimiento. Para desarrollar un modelo cinemático directo, podemos empezar especificando el sistema de referencia en el cual se elegirá arbitrariamente un punto fijo. Consideremos que todos los demás puntos en el sistema tienen un movimiento relativo a este punto de referencia. El robot se considerará como un cuerpo rígido.

Según las consideraciones anteriores el punto central del robot puede estar en movimiento por lo que la trayectoria real de la llanta derecha no necesariamente corresponderá al arco circular del centro, pero el cambio de la orientación θ no está restringido al sistema de referencia del robot. Al tratar al robot como un cuerpo rígido todos los puntos en el sistema experimentan el mismo cambio en orientación.

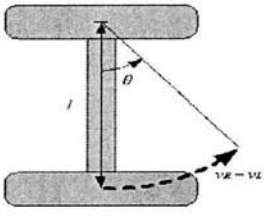


Fig. 5.2 Movimiento del robot

Un ángulo dado en radianes está definido por el tamaño de un arco circular dividido entre el radio de ese círculo. La velocidad relativa de la llanta derecha nos proporciona ese tamaño del arco por unidad de tiempo. La distancia de la llanta al punto central del robot nos da el radio. Considerando lo anterior se obtiene la relación:

$$d\theta/dt = (VR - VI) / L \quad \text{.....(5.6)}$$

Integrando la ecuación (5.6) y definiendo como condición inicial de la orientación del robot $\theta(0) = \theta_0$, encontramos una función para calcular la orientación del robot en función de la velocidad de las llantas y el tiempo:

$$\theta(t) = (VR - VL) t / L + \theta_0 \text{ .....(5.7)}$$

Este cambio de orientación también aplica al sistema de referencia absoluto, recordemos que de (5.4) y (5.5) el movimiento total del robot depende de la velocidad del punto central, esta velocidad es simplemente el promedio de velocidades de las dos llantas.

$$(VR + VL) / 2 \text{ .....(5.8)}$$

Combinando estas ecuaciones se obtienen las siguientes expresiones:

$$dx/dt = [(VR + VL) / 2] \cos(\theta(t)) \quad \text{.....(5.9)}$$

$$dy/dt = [(VR + VL) / 2] \operatorname{sen}(\theta(t)) \quad \text{.....(5.10)}$$

Integrando y considerando que las condiciones iniciales del robot $x(0) = x_0$ y $y(0) = y_0$ tenemos:

$$x(t) = x_0 + \frac{l(VR+VL)}{2(VR\_VL)}[sen((VR-VL)t/l+\theta_0) - sen(\theta)] \quad \text{.....(5.11)}$$

$$y(t) = y_0 - \frac{l(VR+VL)}{2(VR-VL)}[\cos((VR-VL)t/l+\theta_0) - \cos(\theta_0)] \quad \text{.....(5.12)}$$

Las ecuaciones anteriores nos proporcionan el cálculo de la posición del robot con respecto al sistema de referencia.

5.2 Modelado del sensor

Para poder simular el sistema completo es necesario, obtener la desviación del robot respecto a la trayectoria, ya que esta es una de las variables de control, para ello utilizamos el ángulo α que se muestra en la figura 5.3. Para efectos de simulación es posible obtener esta información basándonos en relaciones geométricas. Con el modelo cinemático del robot, es posible obtener la posición y la inclinación de este en todo instante, podemos considerar al sensor como un segmento de recta paralelo al eje de las llantas a una distancia **d** (Fig. 5.3)

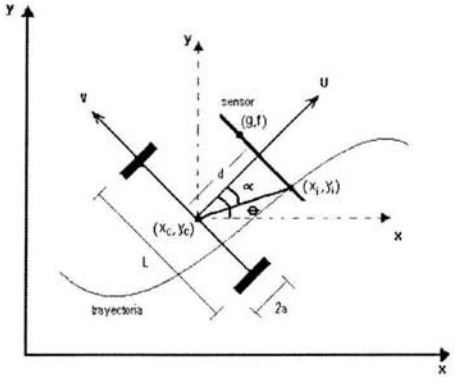


Fig. 5.3: Diagrama incluyendo el sensor y la trayectoria.

La trayectoria esta referida al sistema de coordenadas global (o fijo) (ejes ***x,y***), mientras que la línea del sensor la tenemos respecto al sistema de coordenadas local (o móvil) sobre el robot (ejes ***u,v***), por tanto aplicando las ecuaciones de transformación convenientes podemos obtener la ecuación del sensor respecto al sistema de coordenadas fijo, una vez que tenemos ambas ecuaciones respecto al mismo sistema de coordenadas, las igualamos para obtener el punto de intersección de estas en todo instante, dicho punto esta referido al sistema de coordenadas fijo y el ángulo α se encuentra en el sistema de coordenadas móvil, por tanto nuevamente aplicamos las matrices de transformación y obtenemos este punto respecto a dichos ejes.

Así tenemos que el ángulo α para todo tiempo está dado por:

$$\alpha = a\tan\left(\frac{\cos\theta(y_i - y_c) - sen\theta(x_i - x_c)}{\cos\theta(x_i - x_c) + sen\theta(y_i - y_c)}\right) \quad \text{.....(5.13)}$$

Se realizaron dos simulaciones con trayectorias diferentes, para la primera se considero una línea recta a 45° pasando por el origen (x = y), obteniendo el siguiente punto de intersección.

$$x_i = \frac{g\tan(\theta+90) - f}{\tan(\theta+90) - 1} \quad \text{.....(5.14)}$$

$$y_i = x_i \quad \text{.....(5.15)}$$

donde g y f corresponden a las coordenadas respecto al sistema fijo de un punto sobre el sensor y están dadas por:

$$g = d\cos\theta - .07sen\theta + x_c \quad \text{..... (5.16)}$$

$$f = dsen\theta + .07\cos\theta + y_c \quad \text{..... .....(5.17)}$$

Mientras que para la curva $y = x^2$ el punto de intersección es:

$$x_i = \frac{\tan(\theta+90) + \sqrt{(\tan(\theta+90))^2 - 4g + 4f}}{2} \quad \text{.....(5.18)}$$

$$y_i = x_i^{\,2} \quad \text{.....(5.19)}$$

Las otras dos variables de control que son las velocidades en las llantas, se obtuvieron aplicando un modelo inverso basándonos en el modelo cinemático del

robot, para poder obtener la velocidad de las llantas conociendo la posición y el ángulo θ.

5.3 Programación en Simulink

Con el modelo cinemático del robot y una ecuación del ángulo de desviación en todo tiempo dependiendo de la la posición absoluta del robot, ya es posible realizar un esquema con bloques de simulink el cual refleje el comportamiento del sistema a controlar, primero se realizaron pruebas con el modelo cinemático del robot, con entradas fijas de velocidad en las llantas para verificar que el modelo reflejaba realmente el comportamiento del robot. El diagrama en simulink correspondiente al modelo es el mostrado en la figura 5.4.

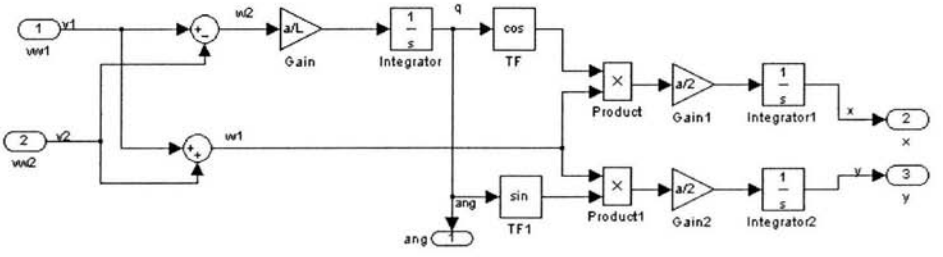


Fig. 5.4: Modelo cinemático del robot

Una vez que se verificó el modelo se agregó, el ángulo α y el control mediante bloques de función (Fig. 5.5), programando el algoritmo de control en un archivo m. Debido a que los bloques de función sólo permiten una sola salida, fue necesario colocar un bloque de control para cada salida deseada, estas salidas son como ya se ha mencionado antes, las velocidades en las llantas.

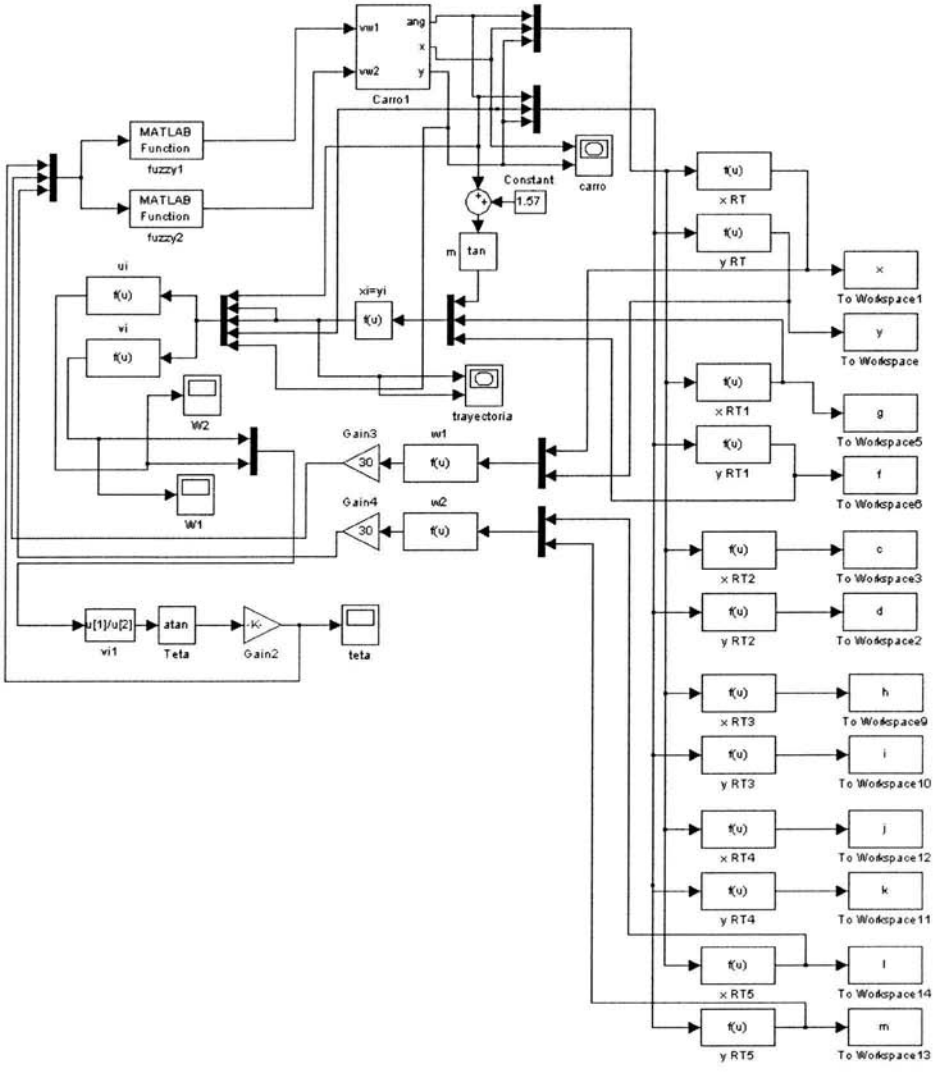
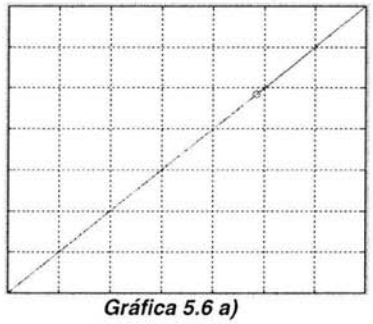


Fig. 5.5: Diagrama simulik del sistema completo

Para poder visualizar mejor el comportamiento del sistema, se realizó una sencilla animación la cual muestra el robot siguiendo la trayectoria deseada, también se graficaron juntas la trayectoria deseada y la trayectoria seguida por el robot para compararlas.

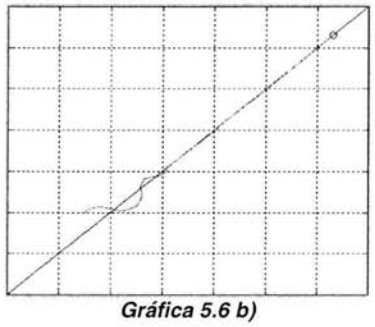
5.4. Resultados de la simulación

Como podemos observar en la figura 5.6 a), la cual corresponde a las gráficas del movimiento del robot sobre la trayectoria recta, este sigue adecuadamente la trayectoria cuando inicialmente se encuentra bien ubicado, en el caso en el cual se encuentra de inicio con un ángulo de desviación grande al principio oscila un poco pero en cuanto se estabiliza sigue satisfactoriamente la pista (gráfica 5.6 b)). En la figura 5.7 podemos observar la misma situación pero en una trayectoria curva



Gráfica 5.6 a)

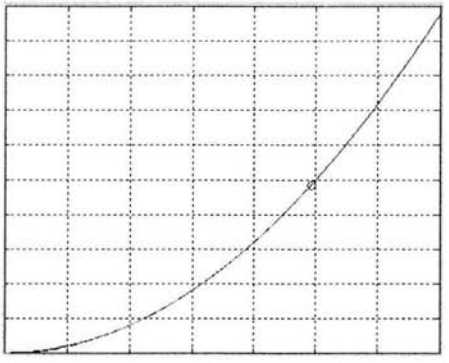
a) En este caso la posición inicial del robot esta sobre la trayectoria deseada



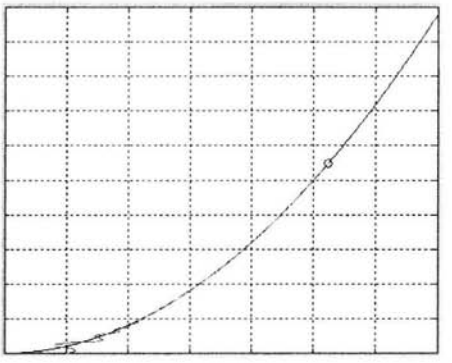
Gráfica 5.6 b)

b) Ahora la posición inicial del robot es diferente, como se muestra en la gráfica.

Figura 5.6: Las gráficas muestran el movimiento del punto X_c,y_c del robot sobre la trayectoria, el círculo representa el robot.

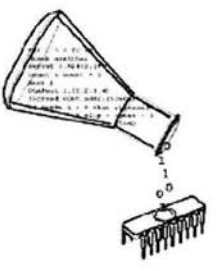


Gráfica 5.7 a)



Gráfica 5.7 b)

Fig. 5.7: Graficas del movimiento del robot siguiendo una trayectoria curva, las situaciones en las gráficas a) y b) son las mismas que en la figura 5.6



6. *Implementación física*

6.1 Programación del controlador.

La programación se desarrolló en lenguaje C con el compilador CCS para facilitar el manejo de valores flotantes, una programación modular optimiza el código y facilita el mantenimiento de este. El microprocesador PIC16F877 que se utiliza cuenta con dos módulos PWM y un convertidor A-D con 8 canales de conversión. El compilador CCS cuenta con funciones que facilitan el uso correcto de estos módulos. Podemos considerar cuatro partes principales en el programa: Configuraciones iniciales, lectura de las señales de entrada, procesamiento de entradas y generación de salidas. El diagrama general del programa es el siguiente (Fig. 6.1).

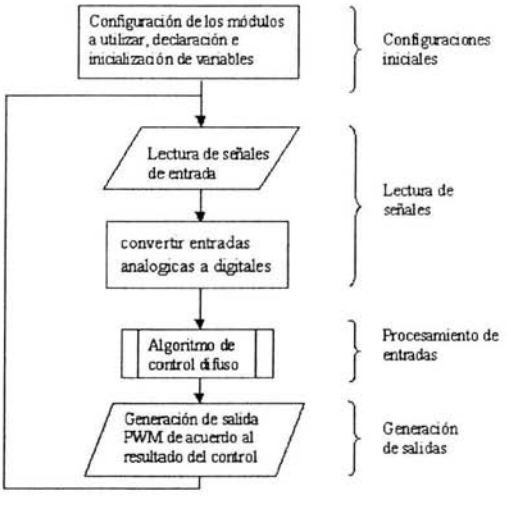


Figura 6.1 Diagrama general del algoritmo implementado

Los módulos CCP1 y CCP2 son módulos de captura, comparación ó PWM. Configurándolos para trabajar como PWM son utilizados para generar la señal hacia los motores que mueven las llantas, la frecuencia de la señal de salida se especifica en el registro Timer2 del microcontrolador. Las funciones que nos ayudan a configurar tanto la frecuencia como el modo de operación del módulo CCPx son SETUP_TIMER2( ) y SETUP_CCPx( ) respectivamente.

Si el sensor de línea se utiliza en modo digital, el valor del sensor se lee en el puerto D y si se utiliza en modo analógico se lee por los pines A2 - A5 del convertidor A/D. El microcontrolador recibe las entradas analógicas correspondientes a las velocidades de las llantas por los canales A0 y A1 del convertidor A/D; para configurar esto se utiliza la función SETUP_ADC( ). Los parámetros de entrada válidos para esta función y las anteriores, se pueden revisar en el subdirectorio *driver* dentro del directorio de instalación del compilador, en un archivo h, cuyo nombre corresponde al microprocesador para el cual define estos parámetros.

Las variables a utilizar por el algoritmo de control se declararon como flotantes, para tener una mayor precisión y manejar valores membresía entre cero y uno. Los valores de entrada se guardan en tres variables globales T, VD y VI; en donde T corresponde a la lectura del sensor de línea y VD y VI a las velocidades de las llantas izquierda y derecha respectivamente. Los valores analógicos ya convertidos se guardan con ayuda de la función READ_ADC(), la cual recupera el VALOR de conversión en una variable de 8 o 16 bits.

Dado que no se utiliza T para el control sino DT según lo especificado en el diseño del controlador, se calcula DT mediante la siguiente expresión:

```
DT=18-T;
```

Donde: T = lectura del sensor.
18 = valor medio.

Se utiliza la función abs() para mandar el valor absoluto a la función membresía, sin perder el signo de DT, ya que se requiere para evaluar las reglas.

Al inicio de la función main() se hacen las configuraciones necesarias para cada uno de los módulos a utilizar, la frecuencia a utilizar por el PWM se fijo en 20Mhz. Esta frecuencia se determina despejando PR2 de la siguiente ecuación:

$$PWMperiodo = [(PR2+1)*4*Tosc*(TMR2prescalador)] \quad \text{.....(6.1)}$$

PR2 es el valor del registro PR2 y Tosc es la frecuencia a la cual está trabajando el PIC. Este valor así como el valor de prescalador se pasan como parámetros a la función setup_timer_2().

Para cambiar el valor del ciclo de trabajo del PWM se utiliza la función set_pwmX_duty() la cual recibe como parámetro un entero constante o variable de 8 ó 16 bits y escribe un valor de 10 bits para el ciclo de trabajo. Si el valor es de 8 bits estos son corridos a la izquierda colocando dos ceros en los bits menos significativos. La duración del tiempo en alto de la señal PWM en cada ciclo se puede calcular como sigue:

Value*(1/clock)*t2div .....(6.2)

Donde clock es la frecuencia del oscilador y t2div es el prescalador del Timer2.

El algoritmo de control sigue un esquema para un controlador difuso, como se presentó en el capítulo 3, considerando los siguientes módulos:

- Fusificación
- Base de Reglas
- Mecanismo de inferencia
- Defusificación

6.1.1 Fusificación.

Para obtener los valores membresía de cada una de las entradas se definió una subrutina que evalúa la función membresía. Por tratarse de conjuntos triangulares simétricos, de los dos valores membresía que se tienen por cada entrada uno es $v_2=1-v_1$, por tanto sólo se calcula v_1 mediante la función membresía y con este se obtiene v_2.

Cada subconjunto difuso tiene la misma función membresía pero en diferente intervalo, de acuerdo con esto, el módulo de fusificación primero ubica la entrada en un intervalo el cual manda como parámetro a la función membresía para obtener la fusificación de las entradas, guardando los valores difusos obtenidos en variables globales.

6.1.2 Base de Reglas.

Una vez obtenidos los valores membresía se deben evaluar cada una de las reglas dentro de la base de reglas, recordemos que las reglas tiene una estructura **Si-Entonces** lo que hace que sea muy sencillo el traducirlas a código C mediante estructuras ***If-Else***. Para determinar que regla aplica para el caso en curso es necesario revisar los valores membresía obtenidos. Sin importar el valor que tengan mientras sean diferentes de cero todos los valores difusos de las variables involucradas en la premisa la regla es aplicable, cada premisa tiene tres valores difusos a comprobar, se creo otra subrutina para realizar la comparación de los tres valores flotantes y así evitar repetición de código.

6.1.3 Mecanismo de inferencia y defusificación.

Para dar el grado de aplicabilidad a cada regla se debe multiplicar el valor de las variables difusas de la premisa, este valor se utiliza para obtener la conclusión mediante la ecuación 4.5, para esto se creó una subrutina de ponderación, la cual obtiene el valor de aplicabilidad de cada regla evaluada y lo multiplica por el valor de salida de la misma, estos valores los suma a los resultados anteriores. Esta subrutina es llamada desde la subrutina encargada de evaluar las reglas cada vez que determina que una regla es aplicable. Después de revisar todas la reglas en la base de reglas se realiza la operación indicada por la ecuación 4.5 y se obtiene la conclusión.

El diagrama de flujo que ilustra el algoritmo de control se muestra en la figura 6.2

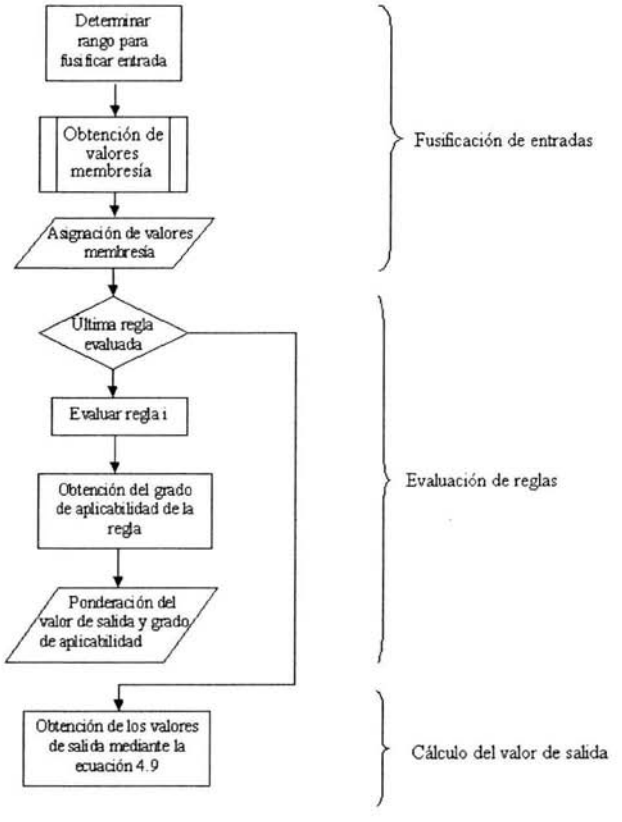


Fig. 6.2 Funciones a implementar para el control difuso

6.2 Simulación en MPLAB.

MPLAB es un software que junto con un emulador y un programador de los múltiples que existen en el mercado, forman un conjunto de herramientas de desarrollo muy completo para el trabajo y/o el diseño con los microcontroladores PIC.

MPLAB incorpora todas las utilidades necesarias para la realización de cualquier proyecto, el programa permite editar el archivo fuente en lenguaje ensamblador, además de ensamblarlo y simularlo en pantalla, permitiendo depurar el programa ejecutándolo paso a paso y ver como evolucionarían de forma real sus registros internos, la memoria RAM y/o EEPROM de usuario y la memoria de programa. Además el entorno que se utiliza es el mismo que si se estuviera utilizando un emulador.

Para probar el código primero se simularon sólo los módulos referentes a la fusificación de las entradas, para verificar que los valores membresía que entregaba eran correctos. Para esto se creo un proyecto para simulación en Mplab, las entradas se escribieron directamente en los registros correspondientes, y mediante la tecla f7 se ejecuta paso a paso el programa, las ventanas de observación permiten agregar todos los registros en la memoria dato que se deseen monitorear.

Una vez comprobado el código de fusificación se agregó el código correspondiente a las reglas para comprobar que el código era correcto. Por último se agregó el código necesario para las entradas y salidas, haciendo programas previos para poder verificar el correcto funcionamiento del convertidor y el PWM desde el microprocesador.

6.3 Acondicionamiento de entradas y salidas.

Fue necesario determinar experimentalmente los valores de los extremos para los intervalos en las funciones que definen a los subconjuntos membresía, de acuerdo a lo que entrega el convertidor y la lectura del puerto D. Para la velocidad se debe considerar que esta es una variable tanto de entrada como de salida, las entradas como ya se ha mencionado son voltajes analógicos que se convierten en el microprocesador, en este caso el voltaje entregado por el CI LM2917 que convierte la frecuencia entregada por el encoder en un voltaje entre 0 y 2.5 [V], y las salidas son señales de PWM que varían la velocidad de los motores, por tanto se debe tener una equivalencia para poder convertir el valor de salida del algoritmo difuso en un valor de PWM que represente la misma velocidad.

La función de conversión se determinó experimentalmente, la ecuación 6.3 es la función que nos convierte la conclusión en un valor de PWM.

$$V_{PWM}=1.15*V_{LEIDO}+200 \quad \text{.....(6.3)}$$

Una conclusión negativa del controlador significa que una llanta debe girar hacia atrás. Para controlar el sentido de giro de las llantas se utilizan dos señales las cuales son enviadas a la etapa de potencia, a través de los cuatro pines más significativos del puerto B. La siguiente tabla corresponde a las combinaciones para estas señales

B7	B6	Sentido Llanta derecha	B5	B4	Sentido Llanta izquierda
0	1	adelante	0	1	Adelante
1	0	atrás	1	0	Atrás

6.4 Pruebas físicas.

Una vez delimitados todos los conjuntos difusos de acuerdo a los valores reales se verificó mediante el osciloscopio que las señales de salida variaran de acuerdo a la posición relativa de la línea. Estas primeras pruebas se realizaron con el carro levantado.

Primero se configuró el sensor para trabajar en modo analógico de esta forma se hicieron las primeras pruebas sobre la pista y se fueron ajustando los valores de las constantes para cada una de las reglas, sin embargo no se lograba un comportamiento adecuado del robot al seguir la trayectoria, presentando un movimiento oscilatorio senoidal que aumentaba en amplitud a medida que el robot avanzaba lo que hacía que el robot perdiera la pista aún a bajas velocidades.

Con el fin de verificar lo que estaba pasando con el control, se enviaron a la computadora los valores de las entradas y salidas del robot mientras hacía su recorrido. Se observó que los valores que entregaba el sensor eran muy inestables, debido a que en modo analógico el sensor es más sensible a los cambios de altura y vibraciones del robot, en ocasiones teníamos lecturas distintas para mismas posiciones.

Se optó por usar el sensor en modo digital, lo cual resta resolución al sensado, pero da una salida estable y mucho más confiable. Realizando pruebas sobre la pista se observó un mejor desempeño del robot con las mismas constantes. Sin embargo en ocasiones las reacciones del robot eran muy bruscas, provocando oscilaciones. Al revisar el algoritmo de control se observó que la ecuación de transformación de valores de encoder a PWM podía estar provocando una sobreactuación del robot. Se ajustó esta ecuación (6.3) experimentalmente a un valor de 160 obteniendo un mejor desempeño del robot.

Primero se probó a una velocidad baja para poder ir ajustando las constantes en las reglas y se fue aumentando la velocidad del robot poco a poco.

Se llegó a un límite de velocidad (70 [cm/seg]) después del cual no se lograba un adecuado comportamiento del robot. Se decidió medir el tiempo promedio que tardaba cada iteración del control. Para esto se encendió uno de los bits menos significativos del puerto B y se mantuvo durante toda la ejecución de una iteración apagándolo al enviar la conclusión a los motores. Así se obtuvo que el tiempo promedio de ejecución del control es aproximadamente 10 [ms]. Este tiempo se vuelve crítico conforme la velocidad del robot aumenta, ya que se hace necesario que el tiempo de muestreo sea mucho menor, para que el robot alcance a reaccionar a los cambios en la trayectoria.

7. Resultados y Conclusiones

Una vez que se logró seguir una trayectoria definida adecuadamente se probó el robot en distintos recorridos. En las pruebas realizadas se pudo observar que sin importar el tipo de combinación entre curvas y rectas se tiene un desempeño aceptable, favoreciéndole las trayectorias con radios de curvaturas mayores a 20 [cm] ya que cuando se tiene curvaturas con radios menores una de las llantas tiene que girar en sentido contrario a la otra, haciendo que el móvil realice giros sobre su eje provocando que el desplazamiento del robot no parezca continuo, la máxima velocidad promedio alcanzada fue de 74 [cm /s]; a mayores velocidades el robot llega a perder la pista. En ningún caso el coche pierde la pista siempre y cuando la velocidad máxima permanezca debajo de los 74 [cm /s].

Cuando la trayectoria tiene rectas largas el robot la sigue eficientemente y a muy buena velocidad incluso mayor a los 74 [cm/s], sin embargo cuando la trayectoria cambia bruscamente después de una recta muy larga, el robot oscila haciendo que pierda velocidad y fidelidad a la trayectoria estabilizándose unos segundos después.

Para reportar el comportamiento del robot por medio de gráficas, se tomaron las lecturas de las velocidades de las llantas y del sensor de línea a través del puerto serie (RS232) de la PC y del microcontrolador.

Las figuras 7.1 y 7.2 corresponden a las gráficas de las las lecturas tomadas durante el recorrido del robot en una trayectoria de 5 [m] de longitud.

A partir de estas lecturas se graficó la diferencia DT y la diferencia entre las velocidades.

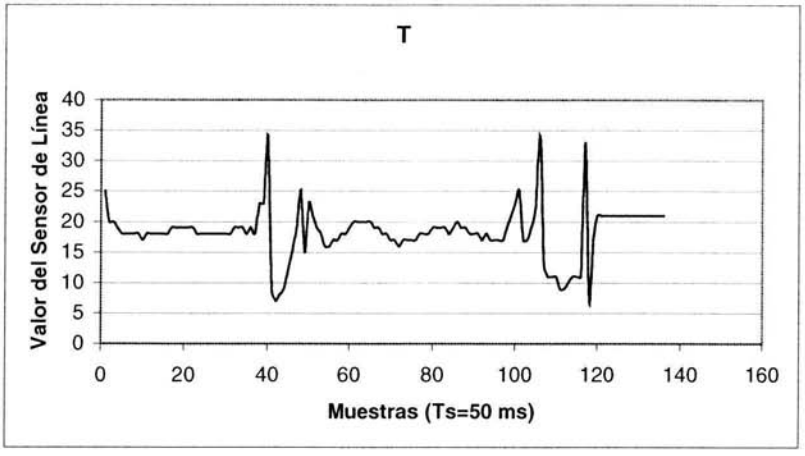


Fig.7.1 Lectura del sensor de línea

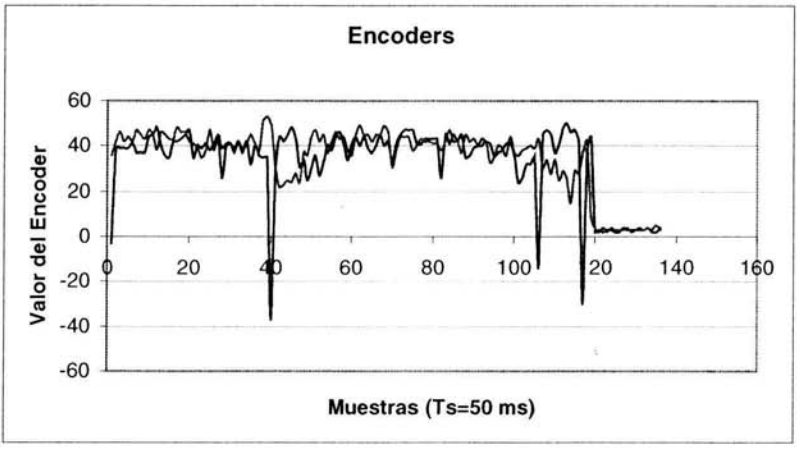


Fig.7.2 Lecturas del encoder

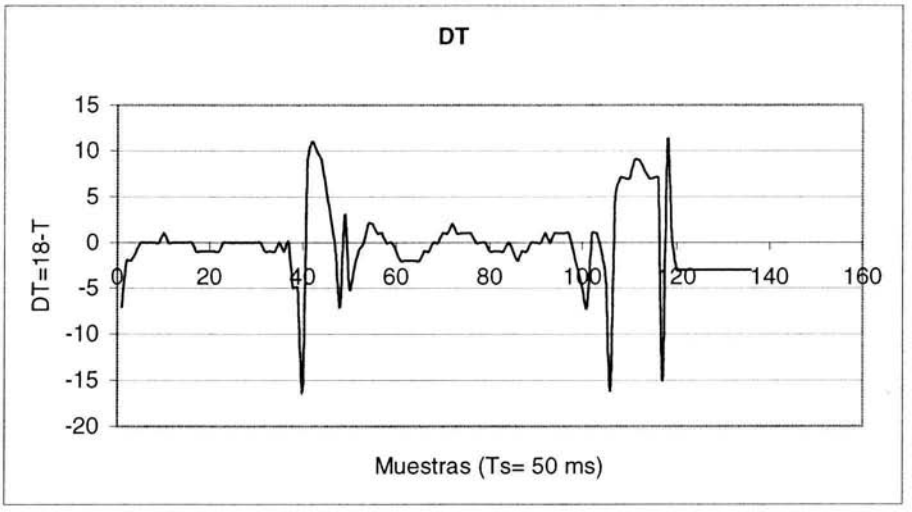


Figura 7.3 Diferencia DT (DT=18-T)

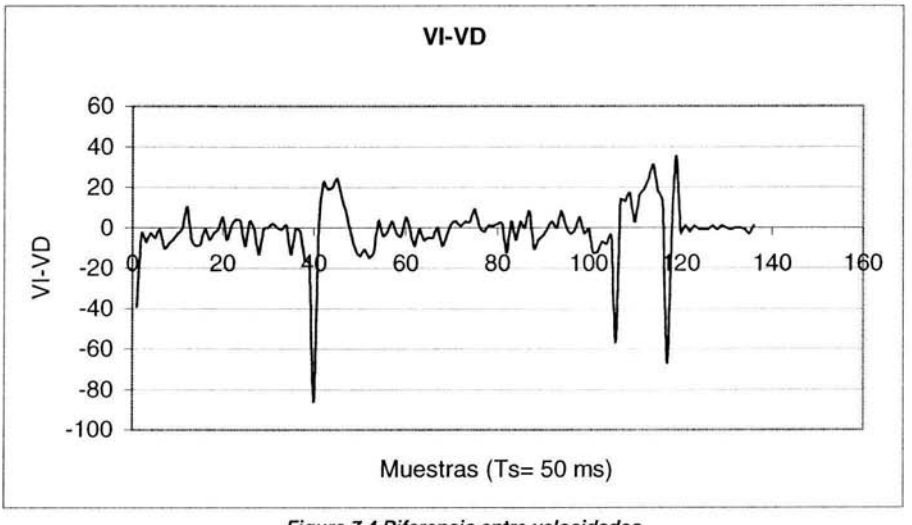


Figura 7.4 Diferencia entre velocidades

La gráfica de DT nos da una idea aproximada de la trayectoria que el robot ve durante el recorrido, cuando el valor se hace negativo representa que la trayectoria va hacia la izquierda y valores positivos representan que la trayectoria va hacia la derecha. Al obtener la diferencia de las velocidades VI-VD, también ocurre lo mismo, ya que cuando el robot gira hacia a la izquierda la velocidad de la

llanta derecha debe ser mayor dando un valor negativo, de igual forma cuando el robot gira a la derecha el valor de la diferencia es positivo. De acuerdo a esto si el comportamiento del robot es correcto estas dos gráficas tendrían la misma forma. Comparando las figuras 7.3 y 7.4 podemos observar que el comportamiento del robot es el esperado, lo que se confirma al observar al robot realizar el recorrido.

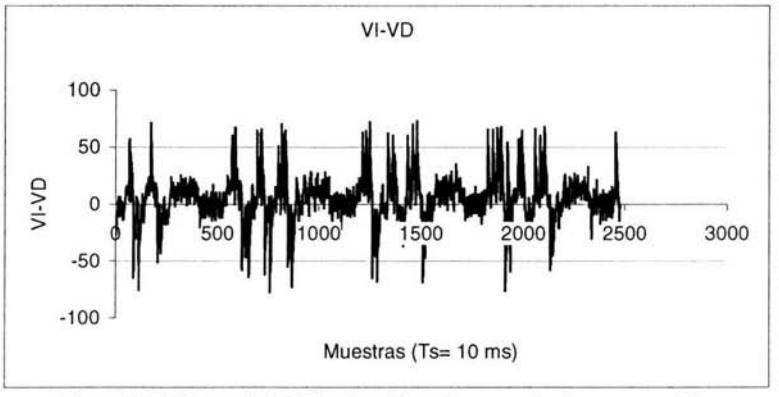


Figura 7.5 Diferencia VI-VD al realizar 3 veces el mismo recorrido.

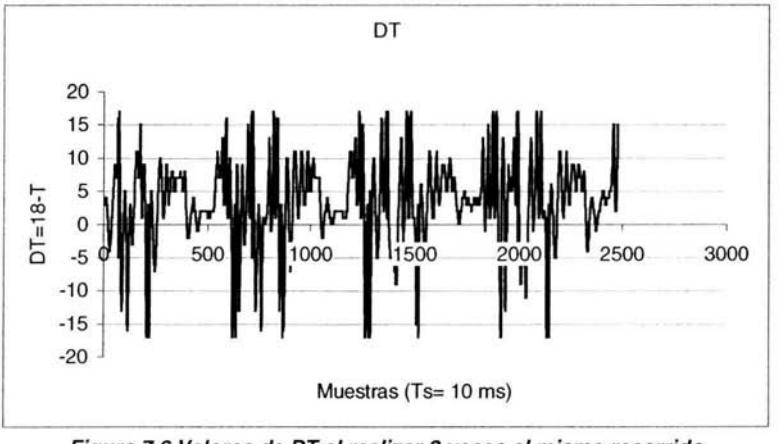


Figura 7.6 Valores de DT al realizar 3 veces el mismo recorrido

Las gráficas 7.5 y 7.6 corresponden a lecturas en donde el robot realizó tres veces el mismo recorrido, podemos observar que el comportamiento se repite casi de igual forma las tres ocasiones.

Para poder representar de una forma más clara el desempeño del robot, se tomo el modelo cinemático empleado en la simulación para poder graficar una

aproximación de la trayectoria recorrida por el robot. La figura 7.8 muestra el resultado de esta simulación. La trayectoria real se muestra en la figura 7.7. Como se puede observar el recorrido es muy parecido a la trayectoria real que siguió el robot.

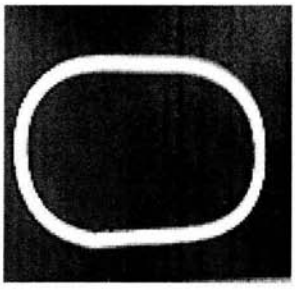


Figura 7.7 Fotografía de la pista

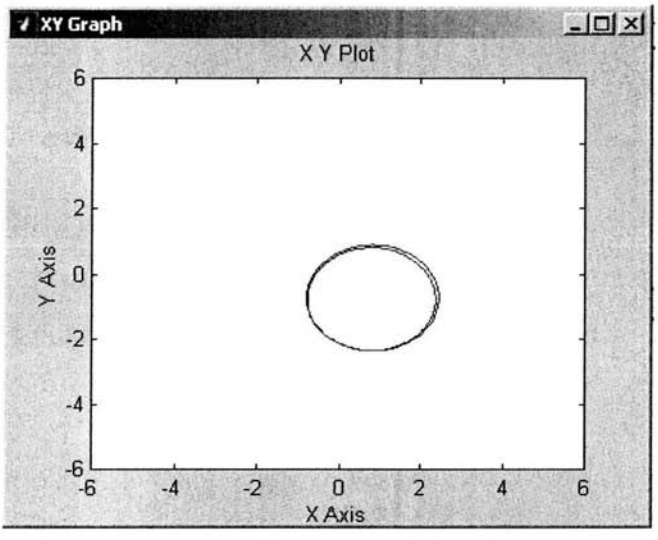


Figura 7.8 Resultado de la simulación con los valores leídos sobre la pista

De acuerdo a los resultados anteriores podemos concluir que el diseño mecánico es un diseño robusto con las capacidades mecánicas necesarias para cubrir los requerimientos especificados, sin embargo un diseño mecánico más reducido en dimensiones ayudaría a mejorar el desempeño del robot al recorrer trayectorias con radios de curvatura pequeños. Los motores seleccionados,

cuentan con las especificaciones deseadas ya que tienen fuerza suficiente para soportar el peso de las baterías y de los propios motores, que en conjunto constituyen, el mayor peso en el robot, además de tener la velocidad adecuada gracias a su caja de reducción, la única desventaja encontrada es su gran consumo de corriente lo que implicó aumentar la capacidad de las baterías con arreglos de ellas en paralelo y como consecuencia el aumento del peso del robot.

El tipo de tracción seleccionada posiblemente no sea la más adecuada para robots que se desplazan a gran velocidad ya que la velocidad de las llantas debe ser controlada dinámicamente, lo cual implica monitorear y controlar la velocidad de los motores en todo instante lo que aumenta el tiempo de procesamiento de la información para llevar a cabo el control.

Uno de los factores que motivaron el uso de una configuración diferencial es la relativa sencillez en la construcción mecánica, ya que nuestros conocimientos en el área son limitados además de que el prototipo puede ser utilizado para otro tipo de aplicaciones por ser una de las configuraciones que más se utilizan en laboratorios académicos y de investigación.

En el aspecto electrónico se realizó una búsqueda profunda de los circuitos y componentes electrónicos más adecuados, obteniéndose como resultado varios sistemas construidos por distintos componentes todos ellos de fácil adquisición en el mercado mexicano. Posiblemente el módulo menos eficiente del robot sean los *Puentes H*, ya que tienen una pérdida de voltaje muy grande, de 2 a 3 volts dependiendo de la cantidad de corriente que les demande el motor. Para llevar a cabo el sensado de la línea se presentaron dos modalidades distintas: modo analógico y modo digital, el primero no resultó ser muy estable debido a que su funcionamiento es afectado por muchos factores como la altura del sensor a la pista, la cantidad de luz lugar en donde se mueve el coche, y el grosor de la línea, por lo que en ocasiones se tenían lecturas muy imprecisas. El modo digital resultó ser una mejor opción ya que los factores que afectan al modo analógico no afectan en lo más mínimo al sensor en esta modalidad, sin embargo se pierde resolución ya que solo existen 35 valores posibles de salida, lo cual no afectó al comportamiento del móvil sino por el contrario lo volvió más estable, existen en el mercado sensores que funcionan con un principio parecido los cuales ofrecen mayor resolución pero su costo es bastante elevado y solo se consiguen en Estados Unidos.

El uso de un controlador difuso resultó muy conveniente para nuestra aplicación, ya que obtener un modelo preciso de la planta es un tanto complicado, puesto que hay muchos factores a considerar, desde el material de las llantas hasta el desgaste de los engranes además de que por tratarse de una aplicación en donde el tiempo es un factor importante requerimos de un control sencillo pero robusto.

Las reglas del controlador difuso pretenden cubrir las situaciones posibles en las que se puede encontrar el robot, la cantidad de reglas suficientes o

necesarias dependen de la opinión del experto, en realidad no hay una regla que nos diga cuantas deben ser, más bien lo que hay que considerar es el tipo de aplicación y la implementación física de esta, ya que se traducirán en tiempo de procesamiento y espacio en memoria.

Como se pudo observar el controlador difuso está basado en un control proporcional simple, el control proporcional se encarga de la rotación y traslación del robot. En particular cuando $DT = 0$ la regla de control se convierte en VD = VI = V_i , lo que resulta solamente en un movimiento de traslación; si DT≠ 0 resultará un movimiento de traslación y rotación. Una de las ventajas del control proporcional es que la regla de control es muy simple y el modelo de la planta no se debe conocer, sin embargo la constante K_1 debe ser determinada a prueba y error, esta constante no necesariamente fue óptima pero fue obtenida para alcanzar un equilibrio entre velocidad y estabilidad, al agregar al control proporcional el conocimiento humano en forma de reglas lingüísticas obtuvimos un controlador muy eficiente y robusto.

La simulación del controlador fue muy importante para darnos una idea del funcionamiento de éste antes de realizar las pruebas físicas, aunque existen muchos factores que no se consideraron, por haber usado un modelo cinemático, los resultados obtenidos dieron una idea muy clara de su comportamiento. Matlab cuenta con utilerías especiales para control difuso sin embargo nosotros optamos por no utilizarlas y programar todo en un archivo *.m, para tener un mejor control del algoritmo y después poder plasmarlo en código para el microcontrolador a utilizar de la misma forma como se hizo en Matlab.

Así mismo existe software especializado en control difuso, los cuales generan código para programarse directamente en el microcontrolador, se decidió no utilizarlo con el fin de optimizar la programación y tener un pleno control de los recursos de éste, también se hizo de esta forma por cuestión didáctica, de haberlo hecho en una herramienta de este tipo muchos procesos serían transparentes para nosotros.

Aunque originalmente se pensó programar sólo en ensamblador, se optó por la utilización de un compilador en lenguaje C para introducir valores en punto flotante y dar una mayor precisión al control, e implementar exactamente el mismo algoritmo que se implementó en simulación en Matlab, además de que se facilitó el uso de los módulos del PIC, esto implicó también un aumento en el tamaño del código generado.

Como ya se dijo anteriormente la velocidad máxima alcanzada por el robot queda un poco debajo de lo esperado, creemos que esto se debe a que el tiempo de muestro es bastante grande para este tipo de aplicación, desafortunadamente el microcontrolador utilizado no fue suficientemente rápido para esta tarea. La ventaja de haber utilizado el microcontrolador PIC16F877 radica en que existe otra familia de estos microcontroladores con mucha mayor capacidad, estos son los PIC18, los cuales trabajan a una frecuencia de 40 Mhz y cuentan con

multiplicadores implementados en hardware entre otras mejoras que los hace más eficientes. El microcontrolador PIC18F452 es compatible en todos los aspectos con el PIC16F877, lo que permitiría utilizarlo en el robot sin hacer ningún cambio al hardware y software existentes, sin embargo no se hizo el cambio debido a que no contamos con el compilador en C para esta nueva familia de microcontroladores. Otra alternativa es utilizar un DSP, dada la cantidad de información en tiempo real a procesar, sin embargo esta alternativa es mucho más costosa.

El realizar este pequeño desarrollo, nos da una idea de todo lo que implica la construcción de un prototipo desde los primeros bosquejos hasta las pruebas finales en donde se ven reflejados los aciertos o fallas al momento de diseñar los distintos módulos que componen al modelo. También nos muestra una aplicación práctica de los conocimientos adquiridos durante la carrera que en algunas ocasiones son bastante teóricos. Además nos enfrentan a una gran cantidad de situaciones en donde no sólo los conocimientos sino el sentido común nos ayuda a encontrar una solución adecuada. Definitivamente una de las características más importantes que debe tener un ingeniero es la capacidad de plantear varias soluciones a un problema y seleccionar la más viable.

Apéndice A

A continuación se muestran las hojas de especificaciones técnicas (Data Sheet) de los componentes usados en el proyecto.

MABUCHI MOTOR

RE-260RA

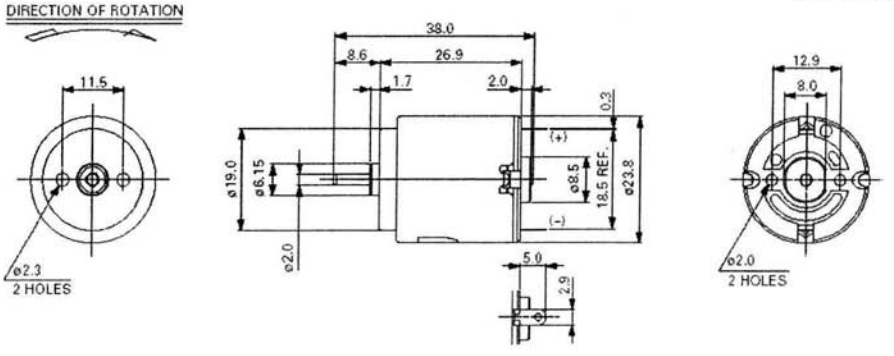
OUTPUT: APPROX 0.4W–4.0W

Metal-brush motors

Typical Applications **Toys and Models** : Motorized Toy / Motorized Plastic Model

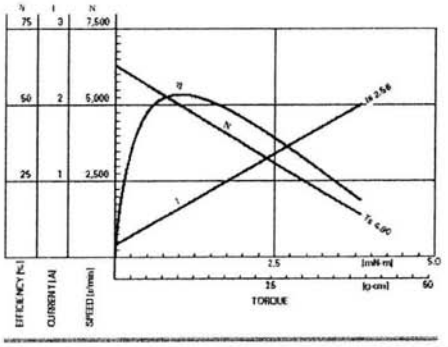
MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	TORQUE		CURRENT
			r/min	A	r/min	A	mN·m	g·cm	W	mN·m	g·cm	A
RE-260RA-2670	1.5 ~ 3.0	1.5V CONSTANT	6300	0.16	5040	0.64	0.98	10.0	0.52	4.90	50	2.56
RE-260RA-18130	1.5 ~ 4.5	3V CONSTANT	6900	0.095	5470	0.36	0.97	9.9	0.56	4.70	48	1.40
RE-260RA-2295	1.5 ~ 4.5	3V CONSTANT	10500	0.15	7610	0.64	1.31	13.3	1.04	6.86	70	2.70

UNIT: MILLIMETERS

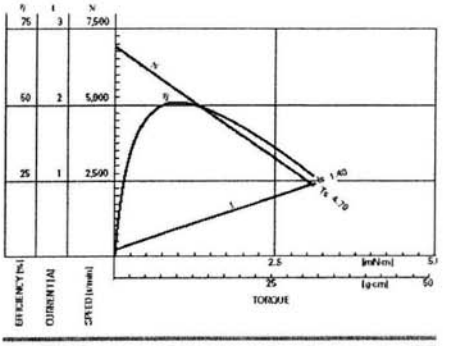


WEIGHT: 28g (APPROX)

RE-260RA-2670 1.5V



RE-260RA-18130 3.0V



MABUCHI MOTOR CO. LTD. Headquarters 430 Matsuhidai, Matsudo-shi, Chiba-ken, 270-2280 Japan Tel:81-47-364-9523 Fax:81-47-385-2026 (Sales Dept.)

MICROCHIP

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

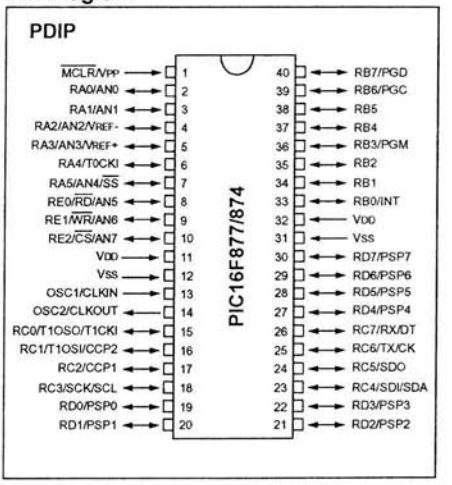
- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Pin Diagram

PDIP



Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I²C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

National Semiconductor

August 2000

LM139/LM239/LM339/LM2901/LM3302
Low Power Low Offset Voltage Quad Comparators

General Description

The LM139 series consists of four independent precision voltage comparators with an offset voltage specification as low as 2 mV max for all four comparators. These were designed specifically to operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage. These comparators also have a unique characteristic in that the input common-mode voltage range includes ground, even though operated from a single power supply voltage.

Application areas include limit comparators, simple analog to digital converters; pulse, squarewave and time delay generators; wide range VCO; MOS clock timers; multivibrators and high voltage digital logic gates. The LM139 series was designed to directly interface with TTL and CMOS. When operated from both plus and minus power supplies, they will directly interface with MOS logic— where the low power drain of the LM339 is a distinct advantage over standard comparators.

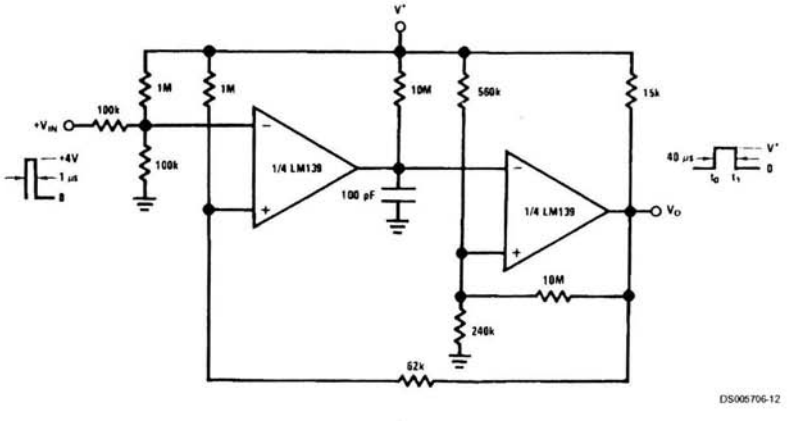
Features

- Wide supply voltage range
- LM139/139A Series 2 to 36 V_{DC} or ±1 to ±18 V_{DC}
- LM2901: 2 to 36 V_{DC} or ±1 to ±18 V_{DC}
- LM3302: 2 to 28 V_{DC} or ±1 to ±14 V_{DC}
- Very low supply current drain (0.8 mA) — independent of supply voltage
- Low input biasing current: 25 nA
- Low input offset current: ±5 nA
- Offset voltage: ±3 mV
- Input common-mode voltage range includes GND
- Differential input voltage range equal to the power supply voltage
- Low output saturation voltage: 250 mV at 4 mA
- Output voltage compatible with TTL, DTL, ECL, MOS and CMOS logic systems

Advantages

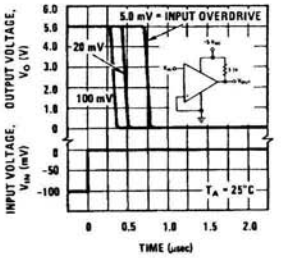
- High precision comparators
- Reduced V_{OS} drift over temperature
- Eliminates need for dual supplies
- Allows sensing near GND
- Compatible with all forms of logic
- Power drain suitable for battery operation

One-Shot Multivibrator with Input Lock Out



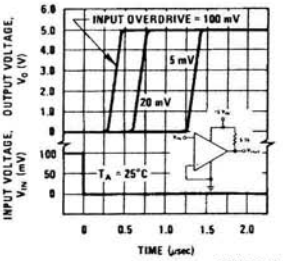
Typical Performance Characteristics LM2901 (Continued)

Response Time for Various Input Overdrives—Negative Transition



DS005706-42

Response Time for Various Input Overdrives-Positive Transition



DS005706-43

Application Hints

The LM139 series are high gain, wide bandwidth devices which, like most comparators, can easily oscillate if the output lead is inadvertently allowed to capacitively couple to the inputs via stray capacitance. This shows up only during the output voltage transition intervals as the comparator changes states. Power supply bypassing is not required to solve this problem. Standard PC board layout is helpful as it reduces stray input-output coupling. Reducing this input resistors to < 10 kΩ reduces the feedback signal levels and finally, adding even a small amount (1 to 10 mV) of positive feedback (hysteresis) causes such a rapid transition that oscillations due to stray feedback are not possible. Simply socketing the IC and attaching resistors to the pins will cause input-output oscillations during the small transition intervals unless hysteresis is used. If the input signal is a pulse waveform, with relatively fast rise and fall times, hysteresis is not required.

All pins of any unused comparators should be tied to the negative supply.

The bias network of the LM139 series establishes a drain current which is independent of the magnitude of the power supply voltage over the range of from 2 V_{DC} to 30 V_{DC}.

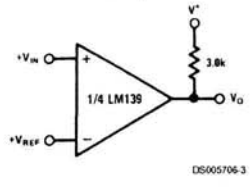
It is usually unnecessary to use a bypass capacitor across the power supply line.

The differential input voltage may be larger than V^+ without damaging the device. Protection should be provided to prevent the input voltages from going negative more than −0.3 V_{DC} (at 25°C). An input clamp diode can be used as shown in the applications section.

The output of the LM139 series is the uncommitted collector of a grounded-emitter NPN output transistor. Many collectors can be tied together to provide an output OR'ing function. An output pull-up resistor can be connected to any available power supply voltage within the permitted supply voltage range and there is no restriction on this voltage due to the magnitude of the voltage which is applied to the V^+ terminal of the LM139A package. The output can also be used as a simple SPST switch to ground (when a pull-up resistor is not used). The amount of current which the output device can sink is limited by the drive available (which is independent of V^+) and the β of this device. When the maximum current limit is reached (approximately 16 mA), the output transistor will come out of saturation and the output voltage will rise very rapidly. The output saturation voltage is limited by the approximately 60Ω R_{SAT} of the output transistor. The low offset voltage of the output transistor (1 mV) allows the output to clamp essentially to ground level for small load currents.

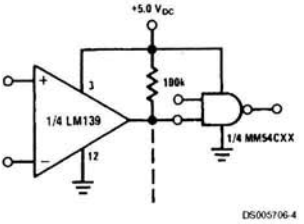
Typical Applications (V^+ = 5.0 V_{DC})

Basic Comparator



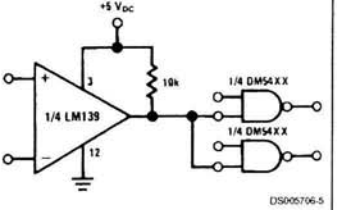
DS005706-3

Driving CMOS



DS005706-4

Driving TTL

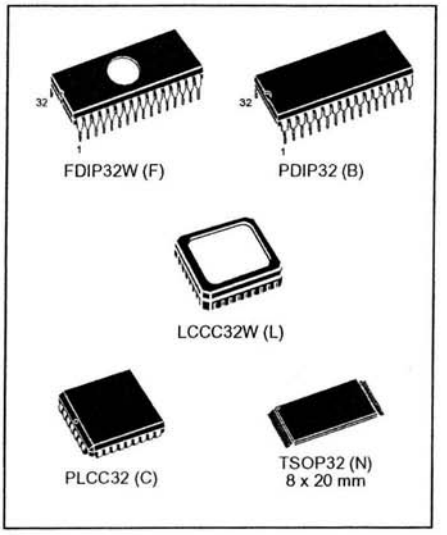


DS005706-5

M27C4001

4 Mbit (512Kb x 8) UV EPROM and OTP EPROM

- 5V ± 10% SUPPLY VOLTAGE in READ OPERATION
- ACCESS TIME: 35ns
- LOW POWER CONSUMPTION:
  - Active Current 30mA at 5MHz
  - Standby Current 100µA
- PROGRAMMING VOLTAGE: 12.75V ± 0.25V
- PROGRAMMING TIME: 100µs/word
- ELECTRONIC SIGNATURE
  - Manufacturer Code: 20h
  - Device Code: 41h



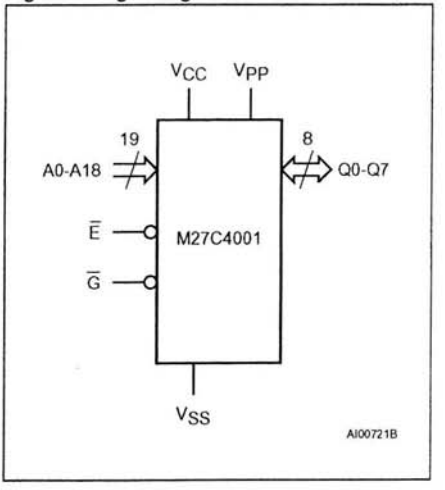
DESCRIPTION

The M27C4001 is a 4 Mbit EPROM offered in the two ranges UV (ultra violet erase) and OTP (one time programmable). It is ideally suited for microprocessor systems requiring large programs and is organised as 524,288 by 8 bits.

The FDIP32W (window ceramic frit-seal package) and LCCC32W (leadless chip carrier package) have a transparent lid which allow the user to expose the chip to ultraviolet light to erase the bit pattern. A new pattern can then be written to the device by following the programming procedure.

For applications where the content is programmed only one time and erasure is not required, the M27C4001 is offered in PDIP32, PLCC32 and TSOP32 (8 x 20 mm) packages.

Figure 1. Logic Diagram



M27C4001

Figure 2A. DIP Connections

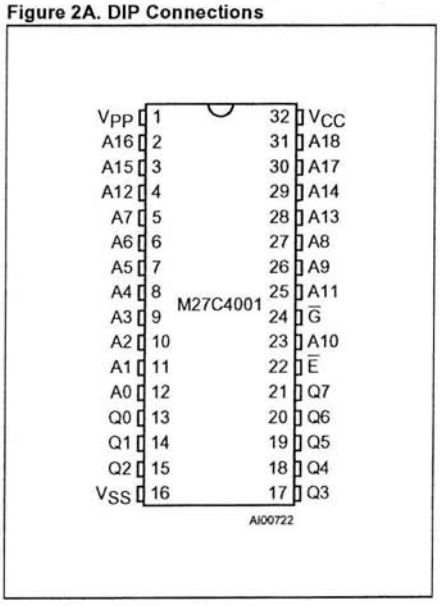


Figure 2B. LCC Connections

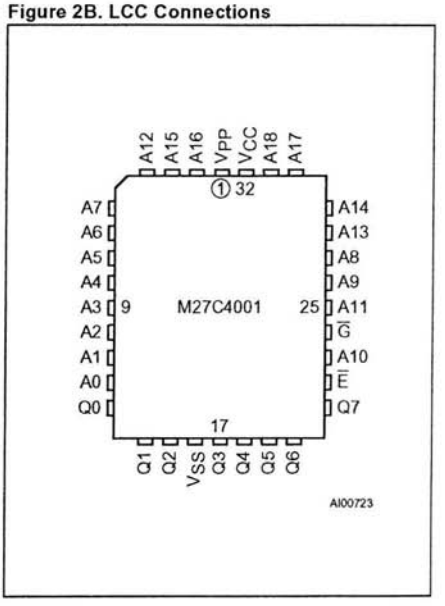


Figure 2C. TSOP Connections

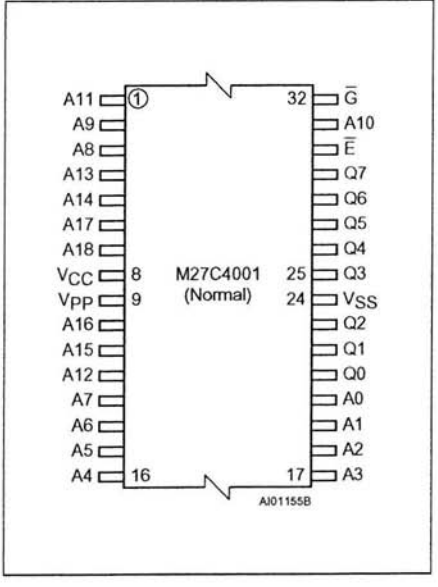


Table 1. Signal Names

A0-A18	Address Inputs
Q0-Q7	Data Outputs
$\overline{E}$	Chip Enable
$\overline{G}$	Output Enable
V_{PP}	Program Supply
V_{CC}	Supply Voltage
V_{SS}	Ground

National Semiconductor

February 1995

LM2907/LM2917 Frequency to Voltage Converter

General Description

The LM2907, LM2917 series are monolithic frequency to voltage converters with a high gain op amp/comparator designed to operate a relay, lamp, or other load when the input frequency reaches or exceeds a selected rate. The tachometer uses a charge pump technique and offers frequency doubling for low ripple, full input protection in two versions (LM2907-8, LM2917-8) and its output swings to ground for a zero frequency input.

Advantages

- Output swings to ground for zero frequency input
- Easy to use; $V_{OUT} = f_{IN} \times V_{CC} \times R1 \times C1$
- Only one RC network provides frequency doubling
- Zener regulator on chip allows accurate and stable frequency to voltage or current conversion (LM2917)

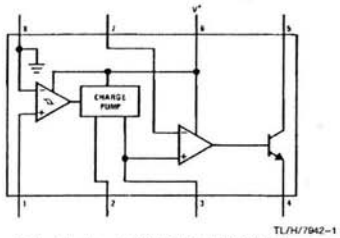
Features

- Ground referenced tachometer input interfaces directly with variable reluctance magnetic pickups
- Op amp/comparator has floating transistor output
- 50 mA sink or source to operate relays, solenoids, meters, or LEDs
- Frequency doubling for low ripple
- Tachometer has built-in hysteresis with either differential input or ground referenced input
- Built-in zener on LM2917
- ±0.3% linearity typical
- Ground referenced tachometer is fully protected from damage due to swings above V_{CC} and below ground

Applications

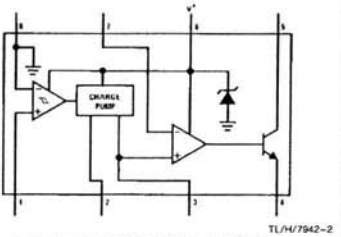
- Over/under speed sensing
- Frequency to voltage conversion (tachometer)
- Speedometers
- Breaker point dwell meters
- Hand-held tachometer
- Speed governors
- Cruise control
- Automotive door lock control
- Clutch control
- Horn control
- Touch or sound switches

Block and Connection Diagrams Dual-In-Line and Small Outline Packages, Top Views



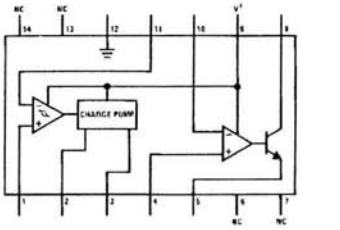
TL/H/7942–1

Order Number LM2907M-8 or LM2907N-8
See NS Package Number M08A or N08E



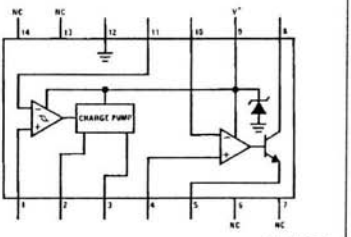
TL/H/7942–2

Order Number LM2917M-8 or LM2917N-8
See NS Package Number M08A or N08E



TL/H/7942–3

Order Number LM2907N
See NS Package Number N14A



TL/H/7942–4

Order Number LM2917M or LM2917N
See NS Package Number M14A or N14A

RRD-B30M115/Printed in U. S. A.

Applications Information

The LM2907 series of tachometer circuits is designed for minimum external part count applications and maximum versatility. In order to fully exploit its features and advantages let's examine its theory of operation. The first stage of operation is a differential amplifier driving a positive feedback flip-flop circuit. The input threshold voltage is the amount of differential input voltage at which the output of this stage changes state. Two options (LM2907-8, LM2917-8) have one input internally grounded so that an input signal must swing above and below ground and exceed the input thresholds to produce an output. This is offered specifically for magnetic variable reluctance pickups which typically provide a single-ended ac output. This single input is also fully protected against voltage swings to ±28V, which are easily attained with these types of pickups.

The differential input options (LM2907, LM2917) give the user the option of setting his own input switching level and still have the hysteresis around that level for excellent noise rejection in any application. Of course in order to allow the inputs to attain common-mode voltages above ground, input protection is removed and neither input should be taken outside the limits of the supply voltage being used. It is very important that an input not go below ground without some resistance in its lead to limit the current that will then flow in the epi-substrate diode.

Following the input stage is the charge pump where the input frequency is converted to a dc voltage. To do this requires one timing capacitor, one output resistor, and an integrating or filter capacitor. When the input stage changes state (due to a suitable zero crossing or differential voltage on the input) the timing capacitor is either charged or discharged linearly between two voltages whose difference is $V_{CC}/2$. Then in one half cycle of the input frequency or a time equal to $1/2\ f_{IN}$ the change in charge on the timing capacitor is equal to $V_{CC}/2 \times C1$. The average amount of current pumped into or out of the capacitor then is:

$$\frac{\Delta Q}{T} = i_{c(AVG)} = C1 \times \frac{V_{CC}}{2} \times (2f_{IN}) = V_{CC} \times f_{IN} \times C1$$

The output circuit mirrors this current very accurately into the load resistor R1, connected to ground, such that if the pulses of current are integrated with a filter capacitor, then $V_O = i_c \times R1$, and the total conversion equation becomes:

$$V_O = V_{CC} \times f_{IN} \times C1 \times R1 \times K$$

Where K is the gain constant—typically 1.0.

The size of C2 is dependent only on the amount of ripple voltage allowable and the required response time.

CHOOSING R1 AND C1

There are some limitations on the choice of R1 and C1 which should be considered for optimum performance. The timing capacitor also provides internal compensation for the charge pump and should be kept larger than 500 pF for very accurate operation. Smaller values can cause an error current on R1, especially at low temperatures. Several considerations must be met when choosing R1. The output current at pin 3 is internally fixed and therefore $V_O/R1$ must be less than or equal to this value. If R1 is too large, it can become a significant fraction of the output impedance at pin 3 which degrades linearity. Also output ripple voltage must be considered and the size of C2 is affected by R1. An expression that describes the ripple content on pin 3 for a single R1C2 combination is:

$$V_{RIPPLE} = \frac{V_{CC}}{2} \times \frac{C1}{C2} \times \left(1 - \frac{V_{CC} \times f_{IN} \times C1}{I_2}\right) \text{ pk-pk}$$

It appears R1 can be chosen independent of ripple, however response time, or the time it takes V_{OUT} to stabilize at a new voltage increases as the size of C2 increases, so a compromise between ripple, response time, and linearity must be chosen carefully.

As a final consideration, the maximum attainable input frequency is determined by V_{CC}, C1 and I_2:

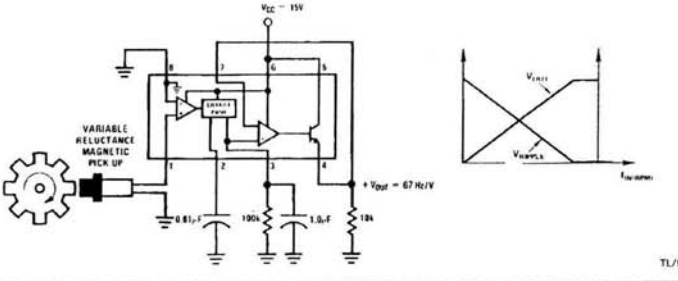
$$f_{MAX} = \frac{I_2}{C1 \times V_{CC}}$$

USING ZENER REGULATED OPTIONS (LM2917)

For those applications where an output voltage or current must be obtained independent of supply voltage variations, the LM2917 is offered. The most important consideration in choosing a dropping resistor from the unregulated supply to the device is that the tachometer and op amp circuitry alone require about 3 mA at the voltage level provided by the zener. At low supply voltages there must be some current flowing in the resistor above the 3 mA circuit current to operate the regulator. As an example, if the raw supply varies from 9V to 16V, a resistance of 470Ω will minimize the zener voltage variation to 160 mV. If the resistance goes under 400Ω or over 600Ω the zener variation quickly rises above 200 mV for the same input variation.

Typical Applications

Minimum Component Tachometer



TL/H/7942–8

L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

Multiwatt15

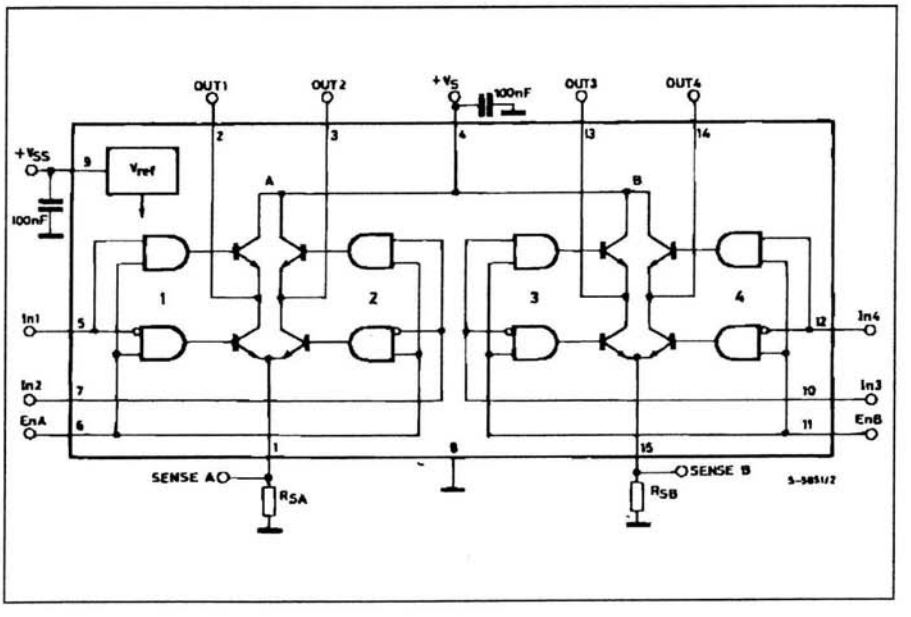
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

BLOCK DIAGRAM

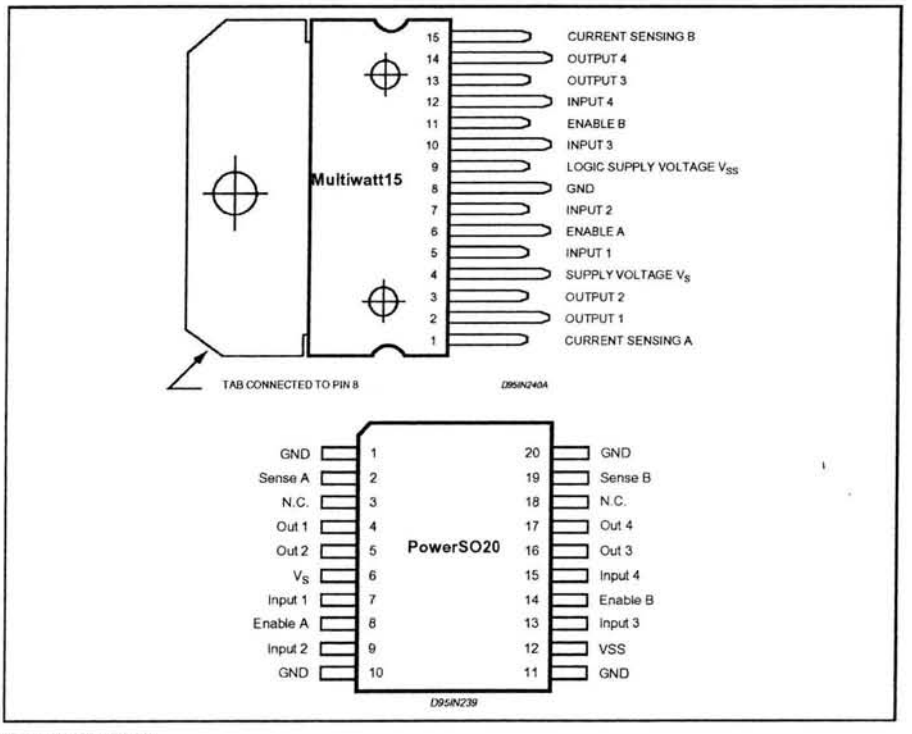


L298

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

National Semiconductor

October 1999

DS14C232
Low Power +5V Powered TIA/EIA-232 Dual Driver/Receiver

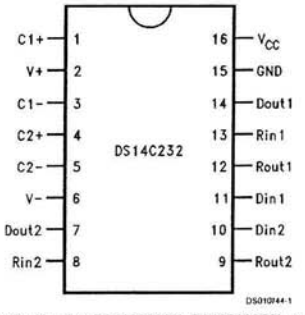
General Description

The DS14C232 is a low power dual driver/receiver featuring an onboard DC to DC converter, eliminating the need for ±12V power supplies. The device only requires a +5V power supply. I_{CC} is specified at 3.0 mA maximum, making the device ideal for battery and power conscious applications. The drivers' slew rate is set internally and the receivers feature internal noise filtering, eliminating the need for external slew rate and filter capacitors. The device is designed to interface data terminal equipment (DTE) with data circuit-terminating equipment (DCE). The driver inputs and receiver outputs are TTL and CMOS compatible. DS14C232C driver outputs and receiver inputs meet TIA/EIA-232-E (RS-232) and CCITT V.28 standards.

Features

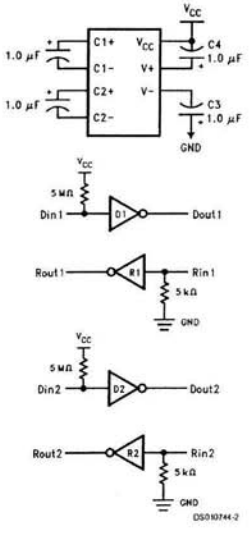
- Pin compatible with industry standard MAX232, LT1081, ICL232 and TSC232
- Single +5V power supply
- Low power—I_{CC} 3.0 mA maximum
- DS14C232C meets TIA/EIA-232-E (RS-232) and CCITT V.28 standards
- CMOS technology
- Receiver Noise Filter
- Package efficiency—2 drivers and 2 receivers
- Available in Plastic DIP, Narrow and Wide SOIC packages
- TIA/EIA-232 compatible extended temperature range option:
  DS14C232T −40°C to +85°C
  DS14C232E/J: −55°C to +125°C

Connection Diagram



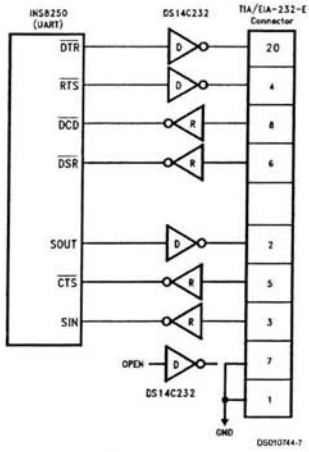
Order Number DS14C232CN, DS14C232CM, or DS14C232TM
See NS Package Number N16E, or M16A

Functional Diagram

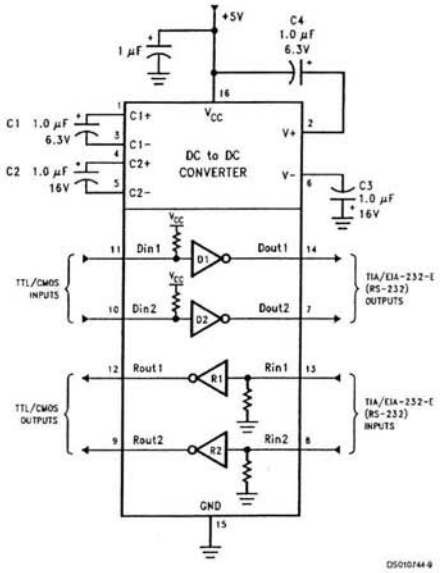


Typical Application Information

Application of DS14C232 and INS8250



Typical Connection Diagram



National Semiconductor

January 2000

LM78LXX Series
3-Terminal Positive Regulators

General Description

The LM78LXX series of three terminal positive regulators is available with several fixed output voltages making them useful in a wide range of applications. When used as a zener diode/resistor combination replacement, the LM78LXX usually results in an effective output impedance improvement of two orders of magnitude, and lower quiescent current. These regulators can provide local on card regulation, eliminating the distribution problems associated with single point regulation. The voltages available allow the LM78LXX to be used in logic systems, instrumentation, HiFi, and other solid state electronic equipment.

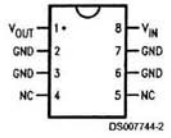
The LM78LXX is available in the plastic TO-92 (Z) package, the plastic SO-8 (M) package and a chip sized package (8-Bump micro SMD) using National's micro SMD package technology. With adequate heat sinking the regulator can deliver 100 mA output current. Current limiting is included to limit the peak output current to a safe value. Safe area protection for the output transistors is provided to limit internal power dissipation. If internal power dissipation becomes too high for the heat sinking provided, the thermal shutdown circuit takes over preventing the IC from overheating.

Features

- LM78L05 in micro SMD package
- Output voltage tolerances of ±5% over the temperature range
- Output current of 100 mA
- Internal thermal overload protection
- Output transistor safe area protection
- Internal short circuit current limit
- Available in plastic TO-92 and plastic SO-8 low profile packages
- No external components
- Output voltages of 5.0V, 6.2V, 8.2V, 9.0V, 12V, 15V

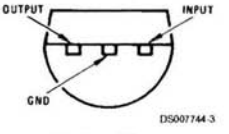
Connection Diagrams

SO-8 Plastic (M)
(Narrow Body)



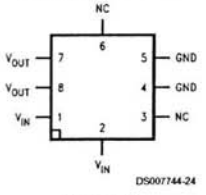
Top View

(TO-92)
Plastic Package (Z)



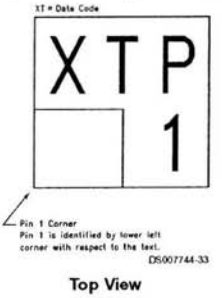
Bottom View

8-Bump micro SMD



Top View
(Bump Side Down)

micro SMD Marking Orientation



Top View

GlobalOptoisolator™

VDE	UL	CSA	SETI	SEMKO	DEMKO	NEMKO	BABT

6-Pin DIP Optoisolators Logic Output

H11L1*
[IF(on) = 1.6 mA Max]
H11L2
[IF(on) = 10 mA Max]
*Motorola Preferred Device

The H11L1 and H11L2 have a gallium arsenide IRED optically coupled to a high–speed integrated detector with Schmitt trigger output. Designed for applications requiring electrical isolation, fast response time, noise immunity and digital logic compatibility.

- Guaranteed Switching Times — t_{on}, t_{off} < 4 µs
- Built–In On/Off Threshold Hysteresis
- High Data Rate, 1 MHz Typical (NRZ)
- Wide Supply Voltage Capability
- Microprocessor Compatible Drive
- ***To order devices that are tested and marked per VDE 0884 requirements, the suffix "V" must be included at end of part number. VDE 0884 is a test option.***

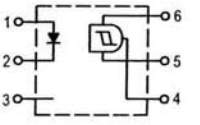
Applications

- Interfacing Computer Terminals to Peripheral Equipment
- Digital Control of Power Supplies
- Line Receiver — Eliminates Noise
- Digital Control of Motors and Other Servo Machine Applications
- Logic to Logic Isolator
- Logic Level Shifter — Couples TTL to CMOS

STYLE 5 PLASTIC

STANDARD THRU HOLE
CASE 730A–04

SCHEMATIC



PIN 1. ANODE
2. CATHODE
3. NC
4. OPEN COLLECTOR OUTPUT
5. GND
6. V_{CC}

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

Rating	Symbol	Value	Unit
INPUT LED			
Reverse Voltage	V_R	6	Volts
Forward Current — Continuous — Peak Pulse Width = 300 µs, 2% Duty Cycle	I_F	60 1.2	mA Amp
LED Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	120 1.41	mW mW/°C
OUTPUT DETECTOR			
Output Voltage Range	V_o	0–16	Volts
Supply Voltage Range	V_{CC}	3–16	Volts
Output Current	I_O	50	mA
Detector Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	150 1.76	mW mW/°C
TOTAL DEVICE			
Total Device Dissipation @ T_A = 25°C Derate above 25°C	P_D	250 2.94	mW mW/°C
Maximum Operating Temperature(2)	T_A	–40 to +85	°C
Storage Temperature Range(2)	T_{stg}	–55 to +150	°C
Soldering Temperature (10 s)	T_L	260	°C
Isolation Surge Voltage (Pk ac Voltage, 60 Hz, 1 Second Duration)(1)	V_{ISO}	7500	Vac(pk)

1. Isolation surge voltage is an internal device dielectric breakdown rating.
   For this test, Pins 1 and 2 are common, and Pins 4, 5 and 6 are common.
2. Refer to Quality and Reliability Section in Opto Data Book for information on test conditions.

Preferred devices are Motorola recommended choices for future use and best overall value.

GlobalOptoisolator is a trademark of Motorola, Inc.

Apéndice B

Software

MPLAB

MPLAB es un software que junto con un emulador y un programador de los múltiples que existen en el mercado, forman un conjunto de herramientas de desarrollo muy completo para el trabajo y/o el diseño con los microcontroladores PIC desarrollados y fabricados por la empresa Arizona Microchip Technology (AMT). MPLAB incorpora todas las utilidades necesarias para la realización de cualquier proyecto y, para los que no dispongan de un emulador, el programa permite editar el archivo fuente en lenguaje ensamblador de nuestro proyecto, además de ensamblarlo y simularlo en pantalla, pudiendo ejecutarlo posteriormente en modo paso a paso y ver como evolucionarían de forma real tanto sus registros internos, la memoria RAM y/o EEPROM de usuario como la memoria de programa, según se fueran ejecutando las instrucciones. Además el entorno que se utiliza es el mismo que si se estuviera utilizando un emulador.

Cuando se pulsa el icono del MPLAB aparece una pantalla como la que se muestra en la Figura b1.

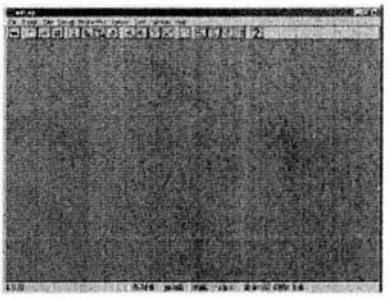


Fig. b1 Pantalla inicial de MPLAB

Lo primero que haremos es seleccionar el modo de trabajo como simulador y el tipo de microcontrolador con el que queremos trabajar. Para ello se selecciona el botón de ***Options*** de la barra del control que aparece en el escritorio y del menú desplegable la opción ***Developement Mode***, con lo que aparece la pantalla en la que se activa el modo *MPLAB-SIM simulator* y el microcontrolador con el que se desea trabajar, que en nuestro caso será el *PIC16F877,* por último, pulsamos el botón de *Reset* para aceptar los cambios.

Los iconos que aparecen en la barra de herramientas, son funciones que se encuentran incluidas en el menú de control, pero como en todos los programas de Windows se incluyen para manejar de forma más cómoda el programa.

Para crear un proyecto, comenzamos por activar en el menú de control la opción ***File>New*** o bien activamos el icono de ***crear nuevo documento*** en la barra de herramientas. El programa contestará con el cuadro de diálogo de la Figura b2.

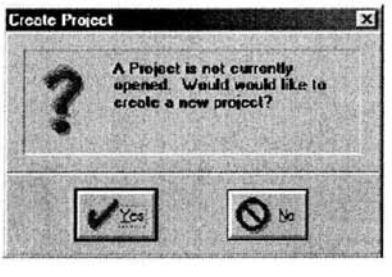


Figura b2.- No hay ningún proyecto abierto ¿Quiere crear un nuevo proyecto?

Activamos el botón de ***Yes*** y aparece un cuadro de dialogo en el que se nos pide el nombre del proyecto que tendrá extensión ****.pjt*** , le llamaremos **ejer1.pjt** y lo guardaremos en la carpeta de **trabajo** que habíamos creado anteriormente.

Activamos el botón de ***OK*** y estamos en condiciones de empezar a escribir nuestro primer proyecto al aparecer una pantalla en blanco (el editor) para iniciar nuestro nuevo proyecto.

En el editor se escribe el programa en lenguaje ensamblador o en lenguaje C. La extensión ****.asm*** es la que deben llevar todos los programas escritos en ensamblador y ****.c*** los programas hechos en C. Deberemos de tener en cuenta que la primera columna del editor está reservada para las **etiquetas** que son expresiones alfanuméricas escogidas por el usuario que definen valores de posiciones de memoria. Estas deben empezar siempre por una letra. Además se debe de tener en cuenta que no pueden usarse expresiones que ya utiliza el ensamblador tales como:

- Instrucciones
- . Directivas del propio ensamblador
- . Nombres de registros especiales
- (SFR)
- Nombre de cada uno de los bit de
- Los registros especiales.

En las siguientes columnas, se puede comenzar a escribir el nemónico de la instrucción o las directivas del ensamblador o C. Por último hay que decir que se pueden y se deben añadir comentarios que son elementos indispensables en muchos casos para seguir el razonamiento de los programas sin perderse.

Cuando se termine de escribir el programa se debe seleccionamos ***File>Save*** con lo que aparece el cuadro de diálogo de la Figura 16, donde le damos el nombre a nuestro programa, dentro de nuestra carpeta **Trabajo**. El siguiente paso será volver a editar nuestro proyecto seleccionando en el menú de control ***project>edit project*** , lo que provoca que aparezca el menú project.

Pulsamos sobre ***ejer1[.hex]***, y se activa el botón de ***Node Properties***, que hasta el momento aparecía de color gris, si lo activamos aparece el cuadro de diálogo de la Figura 19,donde están reflejadas todas las propiedades del nodo actual. Sin modificar ninguna de estas propiedades se pulsa el botón de ***OK*** para continuar, lo que nos lleva de nuevo a la pantalla de la Figura b3. Ahora seleccionamos el botón ***Add Node*** (añadir elementos al nodo), lo que provoca que aparezca un nuevo cuadro de diálogo, en el que seleccionaremos el archivo con el nombre que le dimos.

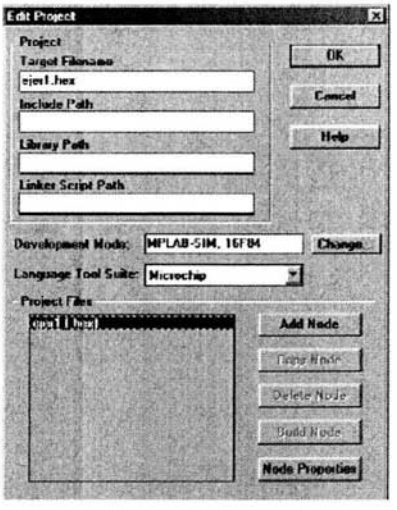


Figura b3.- Ventana de Proyecto

Aparecerá una ventana de Propiedades del nodo de nuestro proyecto donde se seleccionan los ficheros y formatos que se obtendrán al ensamblar el programa. Pulsamos el botón de *Aceptar* y se vuelve a la pantalla de la Figura 17 en la que ha aparecido el fichero *con el nombre de nuestro proyecto.asm o .c* junto al fichero *ejer1[.hex]* que aparecía antes en el campo de ***Project files***. Seguidamente pulsamos el botón de ***OK***. Para ensamblar el programa seleccionamos en el menú de control la opción ***Project>Build All*** (también podríamos haber pulsado el botón correspondiente de la barra de herramientas del simulador y si no se han cometido errores al introducir los códigos, aparece una pantalla como la de lo que nos indica que el programa se ha ensamblado con éxito y ya estamos en condiciones de cargar nuestro programa al microcontrolador.

Compilador CCS

C Compiler for **Microchip PICmicro® MCUs**

Si queremos realizar la programación de los microcontroladores PIC en un lenguaje como el C, es preciso utilizar un *compilador de C*. Dicho compilador nos genera ficheros en formato Intel-hexadedimal, que es el necesario para programar

(utilizando un programador de PIC) un microcontrolador de 6, 8, 18 ó 40 patillas. El compilador de C que vamos a utilizar es el PCW de la casa CCS Inc. A su vez, el compilador lo integraremos en un *entorno de desarrollo integrado* (IDE) que nos va a permitir desarrollar todas y cada una de las fases que se compone un proyecto, desde la edición hasta la compilación pasando por la depuración de errores. La última fase, a excepción de la depuración y retoques hardware finales, será programar el PIC.

Algunas de las funciones del compilador para el uso de los periféricos del microcontrolador.

INPUT(pin)

Devuelve el estado '0' o '1' de la patilla indicada en pin. El método de acceso de I/O depende de la última directiva #USE *_IO utilizada. El valor de retorno es un entero corto.

Ejemplo:
while (!input(PIN_B1));

Nota: El argumento para las funciones de entrada y salida es una dirección de bit. Por ejemplo, para el bit 3° del port A (byte 5 de los SFR) tendría un valor dirección de 5*8+3=43.

Esto se puede definir como sigue: *#define pin3_portA 43.* Los pines o patillas de los dispositivos están definidos como PIN_XX en los archivos de cabecera *.H.

Estos, se pueden modificar para que los nombres de los pines sean más significativos para un proyecto determinado.

OUTPUT_BIT(pin, value)

Esta función saca el bit dado en value(0 o 1) por la patilla de I/O especificada en pin. El modo de establecer la dirección del registro, está determinada por la última directiva #USE *_IO.

Ejemplo:
output_bit(PIN_B0, 0); // es lo mismo que output_low(pin_B0);
output_bit(PIN_B0,input(PIN_B1)); // pone B0 igual que B1
output_bit(PIN_B0, shift_left(&data, 1, input(PIN_B1)));
// saca por B0 el MSB de 'data' y al mismo tiempo
// desplaza el nivel en B1 al LSB de data.

OUTPUT_HIGH(pin)

Pone a 'uno' el pin indicado. El método de acceso de I/O depende de la última directiva #USE *_IO utilizada.

Ejemplo:
output_high(PIN_A0);

OUTPUT_LOW(pin)

Pone a 'cero' el pin indicado. El método de acceso de I/O depende de la última directiva #USE *_IO.

Ejemplo:
output_low(PIN_A0);

· ***SET_TRIS_A(value)***
SET_TRIS_B(value)
SET_TRIS_C(value)
SET_TRIS_D(value)
SET_TRIS_E(value)

Estas funciones permiten escribir directamente los registros tri-estado para la configuración de los puertos. Esto debe usarse con FAST_IO() y cuando se accede a los puertos de I/O como si fueran memoria, igual que cuando se utiliza una directiva #BYTE. Cada bit de value representa una patilla. Un '1' indica que la patilla es de entrada y un '0' que es de salida.

Ejemplo:
SET_TRIS_B(0x0F); // pone B0, B1, B2 y B3 como entradas; B4, B5, B6 y B7
// como salidas, en un PIC 16c84

DELAY_MS(time)

Esta función realiza retardos del valor especificado en time. Dicho valor de tiempo es en milisegundos y el rango es 0-65535. Para obtener retardos más largos así como retardos 'variables' es preciso hacer llamadas a una función separada; véase el ejemplo siguiente. Es preciso utilizar la directiva #use delay(clock=frecuencia) antes de la llamada a esta función, para que el compilador sepa la frecuencia de reloj.

Ejemplos:
#use delay (clock=4000000) // reloj de 4MHz

```
delay_ms( 2 ); // retardo de 2ms
void retardo_segundos(int n) { // retardo de 'n' segundos; 0 <= n => 255
for (; n!=0; n--)
delay_ms( 1000 ); // 1 segundo
}
```

SETUP_TIMER_2(mode, period, postscale)

Esta función inicializa el timer2; mode especifica el divisor del reloj del oscilador, period es un número comprendido entre 0-255, y determina el momento en el que el valor del reloj se reinicia a 0. postscale es un número de 0 a 15, que determina cuántos reinicios del timer se han producido antes de una interrupción. 0 significa 1 reset, 1 significa 2 reset, así sucesivamente. El valor del timer puede leerse y puede escribirse utilizando ET_TIMER2() y SET_TIMER2().

Los valores de mode son:
- o T2_DISABLED
- o T2_DIV_BY_1
- o T2_DIV_BY_4
- o T2_DIV_BY_16

Ejemplo:

```
setup_timer_2 ( T2_DIV_BY_4, 0xc0, 2);
```

SETUP_ADC(mode)

Esta función prepara o configura el conversor A/D. Para la serie 14000 esta función establece la corriente de carga. Véase el archivo 14000.H para los valores según el modo de funcionamiento.

Los modos son:
- o ADC_OFF
- o ADC_CLOCK_DIV_2
- o ADC_CLOCK_DIV_8
- o ADC_CLOCK_DIV_32
- o ADC_CLOCK_INTERNAL

Ejemplo:

```
setup_adc(ADC_CLOCK_INTERNAL);
```

SET_ADC_CHANNEL(canal)

Especifica el canal a utilizar por la función READ_ADC(). El número de canal empieza en 0. Es preciso esperar un corto espacio de tiempo después de

cambiar el canal de adquisición, antes de que se puedan obtener lecturas de datos válidos.

Ejemplo:
set_adc_channel(2);

SETUP_CCP1(mode)
SETUP_CCP2(mode)

Estas funciones inicializa el contador CCP. Para acceder a los contadores CCP se utilizan las variables CCP_1 y CCP_2. Los valores para mode son:

CCP_OFF
CCP_CAPTURE_FE
CCP_CAPTURE_RE
CCP_CAPTURE_DIV_4
CCP_CAPTURE_DIV_16
CCP_COMPARE_SET_ON_MATCH
CCP_COMPARE_CLR_ON_MATCH
CCP_COMPARE_INT
CCP_COMPARE_RESET_TIMER
CCP_PWM
CCP_PWM_PLUS_1 (sólo si se utiliza un ciclo de trabajo de 8 bits)
CCP_PWM_PLUS_2 (sólo si se utiliza un ciclo de trabajo de 8 bits)
CCP_PWM_PLUS_3 (sólo si se utiliza un ciclo de trabajo de 8 bits)

SET_PWM1_DUTY(value)
SET_PWM2_DUTY(value)

Estas funciones escriben los 10 bits de value al dispositivo PWM para establecer el ciclo de trabajo. Se puede usar un valor de 8 bits si no son necesarios los bits menos significativos.

PIC Bootloader.

Un bootloader es usado para "bajar" un nuevo programa al microcontrolador en unos cuantos segundos. La programación del microcontrolador se realiza "in circuit" con el microcontrolador insertado en la tableta del prototipo.

Para poder usar el bootloader es necesario primero programarlo el PIC, esta operación se realiza una sola ocasión. Se deben conectar los pines de comunicación serie del PIC con los de la computadora personal.

En todos los programas que se vayan a cargar en el PIC se deben reservar los 255 bytes más altos en la memoria que es en donde reside el bootloader.

Con ayuda el programa PIC downloader se descargará el programa hacia el microcontrolador. Fig. b1

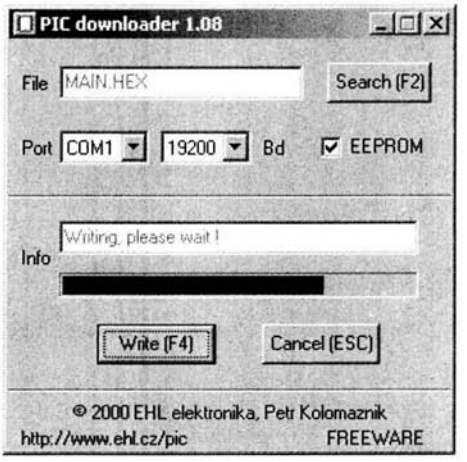


Fig. b1 PIC downloader

Para descargar el programa se debe presionar el botón *Write (F4)* en el software, después dar *reset* al microcontrolador.

El bootloader permanece activo por 0.2 segundos después del reset, para dar la opción de cargar el nuevo código, después de este tiempo el puerto serie se puede usar normalmente en alguna otra aplicación.

El bootloader es compatible con todos los microcontroladores PIC16F87X,

Utiliza 255 instrucciones y se coloca en la parte alta de la memoria y se comunica con la PC a una velocidad de 19200bps

Apéndice C

Microcontroladores PIC

La familia de microcontroladores PIC de Microchip son microcontroladores fáciles de conseguir y relativamente económicos, además de tener una gran cantidad de opciones en cuanto a memoria. Los PIC requieren un sistema mínimo para funcionar y pueden trabajar con un reloj de hasta 40MHz.

Los microcontroladores PIC tienen tecnología RISC con arquitectura Harvard. Esta arquitectura se caracteriza por la independencia entre la memoria de código y la de datos (Fig. a.1). Así, tanto la capacidad como el tamaño de los buses, de cada memoria se adaptan estrictamente a las necesidades del diseño, facilitando el trabajo en paralelo de las dos memorias, lo que permite obtener un alto grado de rendimiento. La filosofía RISC, se hace patente en el reducido número de instrucciones que forman su repertorio.

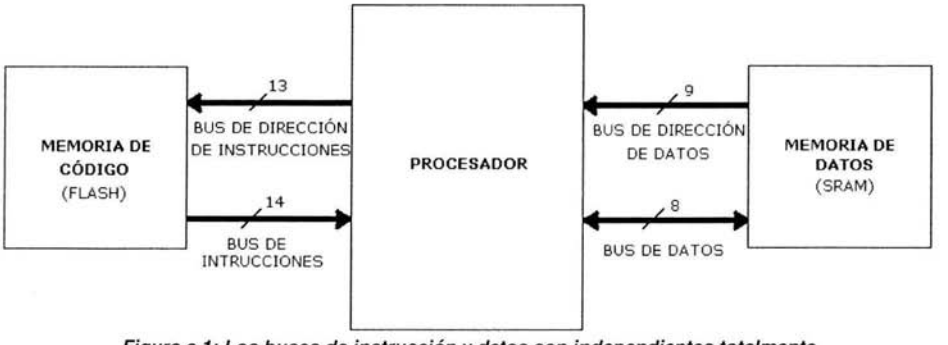


Figura a.1: Los buses de instrucción y datos son independientes totalmente

Los PIC se clasifican en 4 gamas según sus capacidades (Fig. a.2).

Figura a.2: Presentación gráfica de las cuatro gamas del PIC

Dentro de esta familia se seleccionó el microcontrolador PIC de 8 bits 16F877, el cual es un PIC de gama media, cuyas características se muestran en la figura a.3, como podemos ver cuenta con una serie de módulos que nos facilitan el recuperar las señales de entrada y generar las salidas. La arquitectura correspondiente a la subfamilia PIC16F87X es la mostrada en la figura a.4.

MODELO		*PIC16F877*
MEMORIA PROG.	Bytes	14336
	Palabras	8192X14
MEMORIA DATOS	Bytes EEPROM	256
	Bytes RAM	368
C A/D		8(10 bits)
BOD (Detección de baja tensión)		si
LÍNEAS E/S		33
COMUNICACIÓN SERIE		USART/ MSSP
CCP		2
TEMPORIZADORES		1-16 bit, 2-8 bit, 1-WDT
FREC. MAX. MHz		20
ICSP (Programación Serie en Circuito)		si
ENCAPSULADOS		40P,44L,44PQ,44PT
FUENTES DE INTERRUPCIÓN		14
COMUNICACIÓN PARALELO		si

Figura a.3: Características del PIC16f877

La memoria flash de los microcontroladores PIC soporta hasta mil operaciones de escritura / borrado mediante un proceso totalmente eléctrico que no precisa sacar al microcontrolador de su zócalo. Esta característica hace a esta memoria ideal en los ambientes de diseño y educacional. La memoria EEPROM para datos que tienen los PIC soportan 100 000 operaciones de grabado / borrado. Los PIC16F87X incorporan la memoria flash, con una capacidad de 4K y 8K palabras de 14 bits.

La memoria RAM de datos posee una capacidad de 192 bytes en dos de los modelos y 368 bytes en los otros dos.

En los PIC 16F87X se manejan hasta catorce posibles fuentes de interrupción y tres Timer. Contiene hasta cinco puertos de entradas y salidas. Incorpora los siguientes recursos:

1. Dos módulos CCP. Son capaces de capturar y comparar impulsos. La captura se efectúa con una precisión de 12.5 [ns] Y una resolución de 16 bits, mientras que la comparación con igual resolución alcanza una precisión de 200 [ns]. Además la sección PWM varía la anchura de los impulsos, técnica muy usada en el control de motores.

2. Comunicación serie. La típica USART, orientada a la comunicación entre subsistemas o máquinas (RS-232) y la MSSP, destinada a la comunicación entre diversos circuitos integrados y que admite el protocolo I2C y SPI.

3. Comunicación paralela. En los PIC16F877 está disponible el protocolo PSP, más rápido que la comunicación serie, pero que hipoteca muchas líneas de E/S: 8 del puerto D y 3 de control del puerto E.

4. Convertidor A/D. En todos los PIC16F87X existe un convertidor analógico digital de 10 bits, con 8 patitas de entrada.

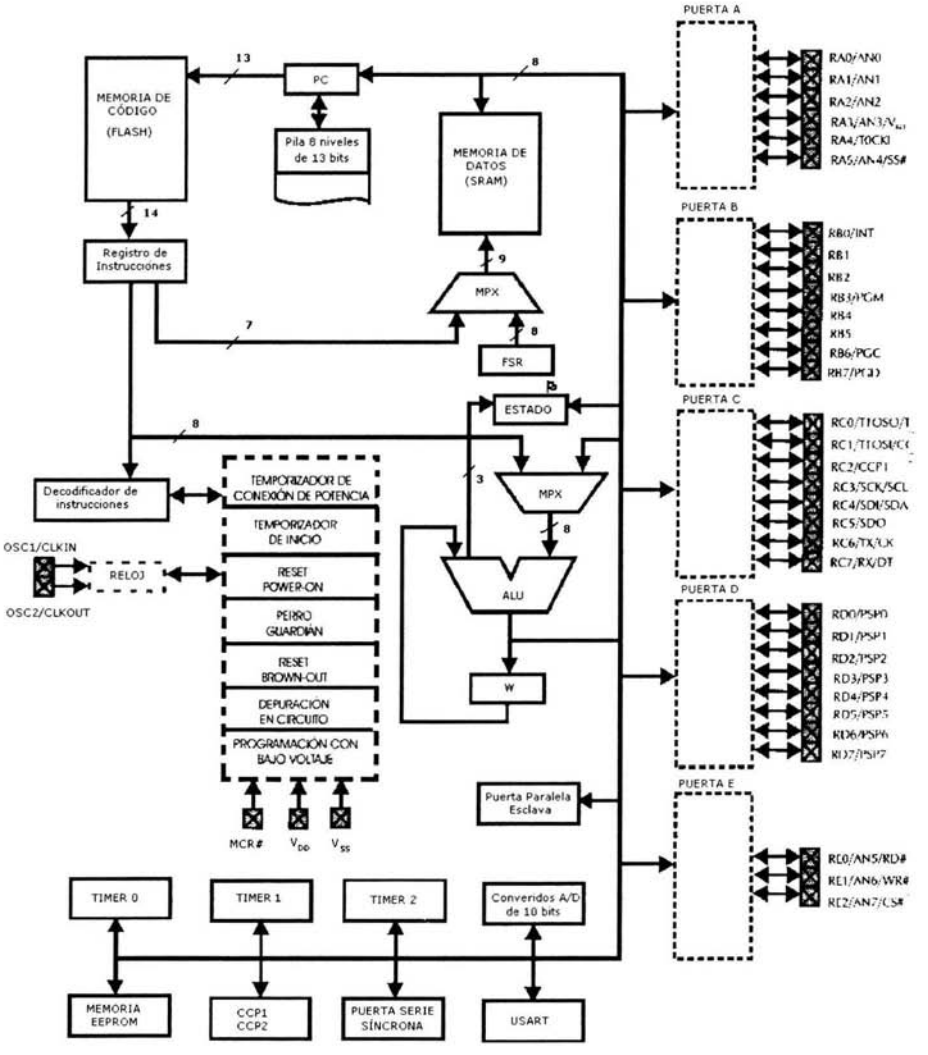


Figura a.4: Arquitectura de los PIC16F874/7 de 40 patitas.

Apéndice D

A continuación se muestra el código en lenguaje C del controlador difuso.

```
#include <16f877.h>
#device ADC=8
#include<stdlib.h>
#include <math.h>

#fuses HS,NOWDT,NOPROTECT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

#ORG 0x1F00,0x1FFF //for the 8k 16F876/7
void loader() { }

                                        /*VARIABLES GLOBALES*/
float T,VI,VD;                          //entradas
float Vx,V1,V2,VP,VG,ve1,ve2,DT;              //auxiliares
float VDP,VDG;                                //subconjuntos de VD
float VIP,VIG;                                //subconjuntos de VI
float TP,TG,TM;                               //subconjuntos de DT
float fct1,fct2;                        //valores de pwm flotantes
float k,l,w;                            //variables de ponderación
int g1,g2,ban;


                                        /*FUNCIONES*/
ym(float x1,float y){                         //Función que obtiene el valor
   float x,y1;                                //de y en la función membresía
      x=(y+x1)/(2*x1);
if(x<0) x=0;
if(x>1) x=1;
       V1=x;
      V2=1-V1;

}//fin ym

ym1(float x1, float x2, float y){       //Función que obtiene el valor
   float x,y1;                                //de y en la función membresía
      x=(x2-x1);
       y1=(y-x1);
      V1=(x-y1)/x;
      V2=1-V1;

}//fin ym1


FMV(){                                  //Función que obtiene los valores
 float q1,q2;                            //membresía para los subconjuntos
 q1=-70;                                       //de la velocidad
 q2=70;

 if(Vx<=q1){
   VP=1;
   VG=0;
}else if(Vx<=q2){
   ym(q1,Vx);
   VP=V1;
   VG=V2;
} else{
```

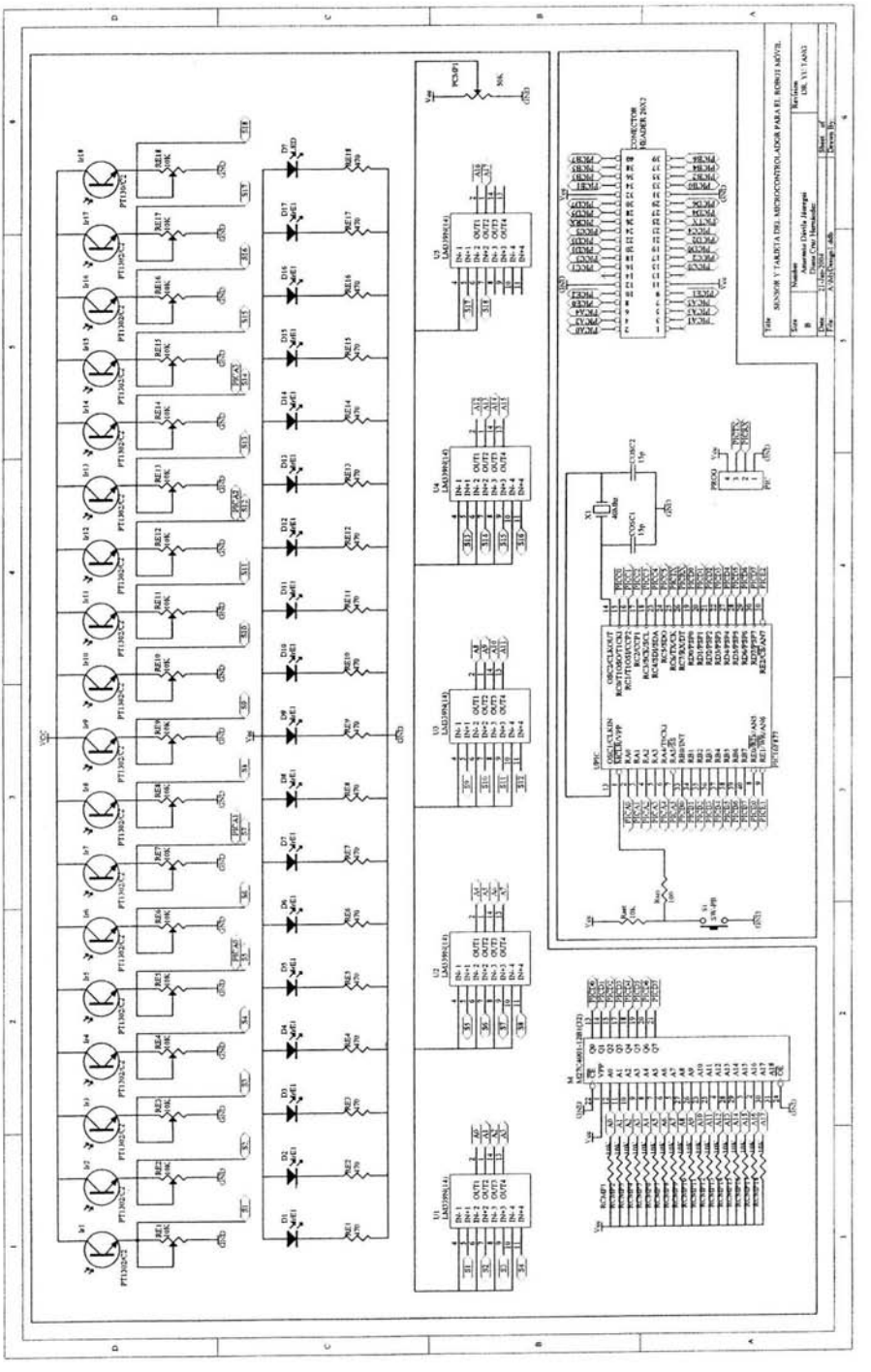
```
   VP=0;
   VG=1;
}
}//fin FMV

FMDT(){                                          //Función que obtiene los valores
 float p1,p2,p3;                               //membresía para los subconjuntos
 p1=0;                                            //DT
 p2=9;
 p3=17;

 if(Vx<=p1){
   TP=1;
   TM=0;
   TG=0;
}else if(Vx<=p2){
   yml(p1,p2,Vx);
   TP=V1;
   TM=V2;
   TG=0;
}else if(Vx<=p3){
   yml(p2,p3,Vx);
   TP=0;
   TM=V1;
   TG=V2;
}else{
   TP=0;
   TM=0;
   TG=1;
}
}//fin FMT

boolean compara(float s1,float s2,float s3){                    //Función auxiliar para
valuar
   if (s1!=0 && s2!=0 && s3!=0)                  //las reglas
   return TRUE;
   else
   return FALSE;
}//fin compara

pondera(float s1,float s2,float s3){             //Función que va ponderando
float wi;                                      //los valores para el
   wi=s1*s2*s3;                                  //cálculo final
   k=k+(ve1*wi);
   l=l+(ve2*wi);
   w=w+wi;
}//fin pondera


VAL_REGLAS(){

if(compara(TG,VIP,VDG)){     //R1
   ve1=10+DT*1.2;
   ve2=10-DT*1.2;
//r='1';
   pondera(TG,VIP,VDG);

}if(compara(TG,VIG,VDP)){    //R2
   ve1=10+DT*1.2;
   ve2=10-DT*1.2;
//r='2';
   pondera(TG,VIG,VDP);

}if(compara(TP,VIP,VDG)){    //R3
   ve1=30+DT*.9;
   ve2=30-DT*.9;
//r='3';
   pondera(TP,VIP,VDG);
}if(compara(TP,VIG,VDP)){    //R4
   ve1=30+DT*.9;
   ve2=30-DT*.9;
```

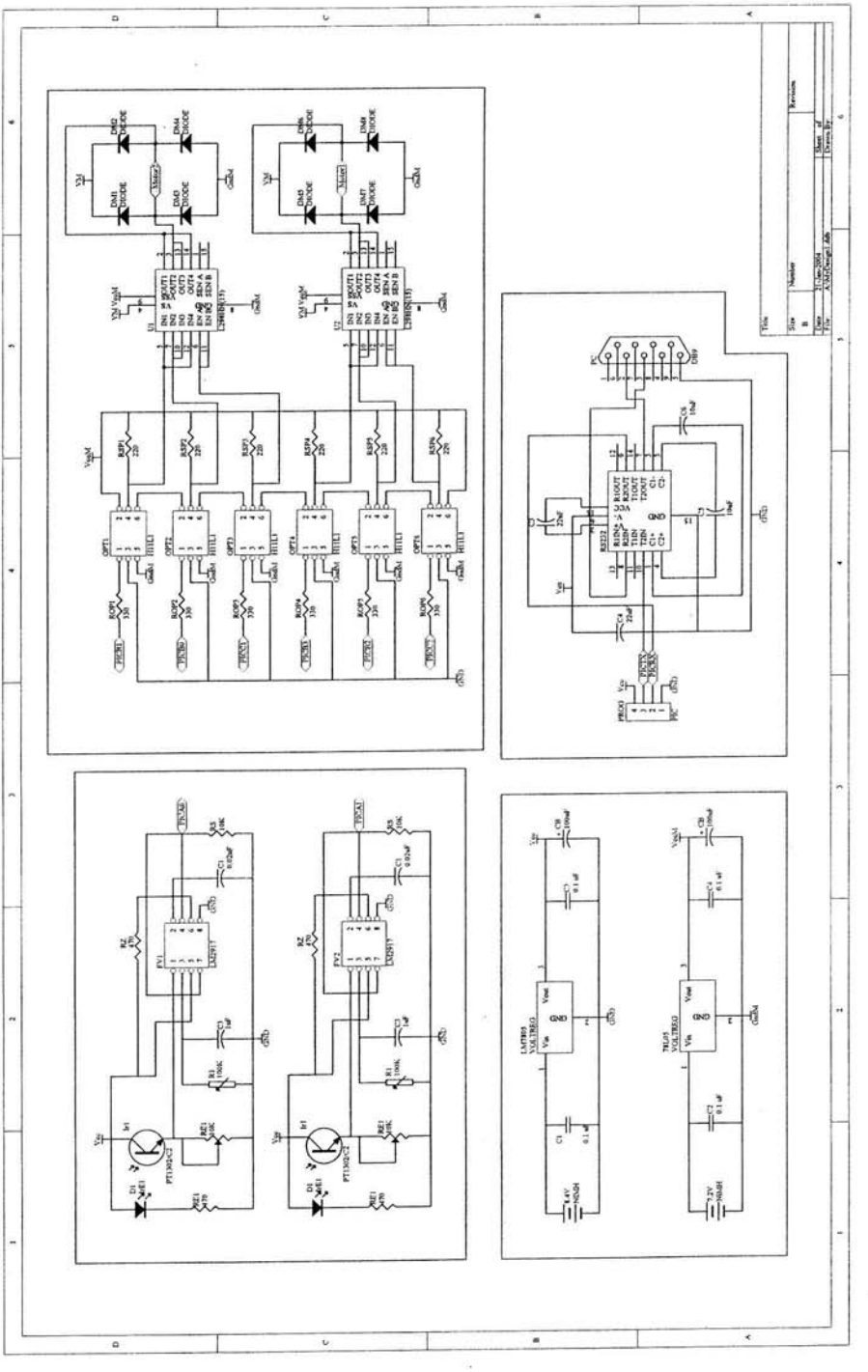
```
//r='4';
   pondera(TP,VIG,VDP);
)if(compara(TM,VIP,VDG)){   //R5
   ve1=20+DT*1.4;
   ve2=20-DT*1.4;
//r='5';
   pondera(TM,VIP,VDG);
)if(compara(TM,VIG,VDP)){   //R6
   ve1=20+DT*1.4;
   ve2=20-DT*1.4;
//r='6';
   pondera(TM,VIG,VDP);
}

}//fin reglas

int en_pwm(float val){
   int x;
     x=1.15*val+200;
    return x;
}

main() {

int ct1,ct2;

//enable_interrupts(INT_RB);
//enable_interrupts(GLOBAL);

set_tris_b(0x00);

output_high(PIN_B5);
delay_ms(2000);
output_low(PIN_B5);

  setup_timer_2(T2_DIV_BY_1,249,1);     //20kHz

  setup_CCP1(CCP_PWM);                          //Se configuran los
  setup_CCP2(CCP_PWM);                          //módulos ccp como PWM


 setup_port_a( ALL_ANALOG );                    //Configuración del convertidor
 setup_adc( ADC_CLOCK_INTERNAL );

set_tris_d(0xff);

while(true){

k=0;
l=0;
w=0;

    set_adc_channel( 0 );                 /***** LECTURA DE ENTRADAS *****/
    delay_us(10);                         // Las entradas se guardan en 3 variables
globales
    VI=read_adc();                              // T  valor del angulo
                                          // VD velocidad llanta derecha
    set_adc_channel( 1 );                 // VI velocidad llanta izquierda
    delay_us(10);
    VD=read_adc();

  T=input_d()&0b00111111;

if (T<36){

    DT=18-T;

if(DT<0) ban=1;
 else ban=0;

if(g1==1) VI=VI*-1;
```

```
if(g2==1) VD=VD*-1;


/*****FUZIFICACION****/
    Vx=abs(DT);
    FMDT();                                  //Valores membresía de DT
    Vx=VD;
    FMV();                                   //Valores membresía de VD
    VDG=VG;
    VDP=VP;
    Vx=VI;
    FMV();                                   //Valores membresía de VI
    VIG=VG;
    VIP=VP;
       /***EVALUACION DE REGLAS***/

    VAL_REGLAS();
              /***SALIDAS****/
 if (w!=0){
   fct1=k/w;
   fct2=1/w;
  if(fct1<0){
     fct1=fct1*-1;
     g1=1;
     output_high(PIN_B0);
     output_low(PIN_B1);
     }else {g1=0; output_low(PIN_B0);
                  output_high(PIN_B1);}
   if(fct2<0){
     fct2=fct2*-1;
     g2=1;
     output_high(PIN_B6);
     output_low(PIN_B7);
     }else {g2=0;

                  output_low(PIN_B6);
                  output_high(PIN_B7);}

   if (fct1>60) fct1=60;
   if (fct2>60) fct2=60;
      ct1=en_pwm(fct1);
      set_pwm1_duty(ct1);
      ct2=en_pwm(fct2);
      set_pwm2_duty(ct2);
}
}

}//fin while(true)
}
```

Apéndice E

A continuación se muestra el diagrama electrónico del robot móvil.





Bibliografía

[1] Jones Joseph L., Flynn Anita M; *Mobile Robots Inspiration to Implementation*. A. K. Peters. USA. 1993

[2] Everett H. R. *Sensor for Mobile Robots. Theory and Aplication*. A. K. Petters USA. 1995.

[3] Ying Hao; *Fuzzy Control and Modeling*. IEEE Press Series on Biomedical Engenieering. 2000

[4] Bradley D. A., Dawson D.; *Mecatronics Electronics in Products and Proceses*. Chapman & Hall. New York. 1996.

[5] Passino Kevin. M., Yurkovich Stephen; *Fuzzy Control*. Addison-Wesley. California 1998.

[6] Boylestad, Nashelsky; *Electrónica: Teoría de Circuitos*. Prenice Hall. México.1996.

[7] Alleyne A., Williams B., DePoorter M.; *A Lateral Position Sensing System for Automated Vehicle Following*. American Control Conference. Philadelphia, Pennsylvania. Junio 1998.

[8] SGS-Thompson Microelectronics; *Applications of Monolithic Bridge Drivers*, AN298.

[9] T.H. Lee, H.K. Lam, F.H.F. Leung, and P.K.S. Tam; *A Practical Fuzzy Logic Controller for the Path Tracking of Wheeled Mobile Robots*. Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University

[10] Mélin Christian, Vidolov Boris, *Two-Rule-Based Linguistic Fuzzy Controllers*. IEEE Transactions on Fuzzy Systems, vol. 11, no. 1, february 2003

[11] Galindo Gómez J.; *Conjuntos y Sistemas Difusos (Lógica Difusa y Aplicaciones)*. E.T.S.I Informática

[12] R. Kruse, J. Gebhardt, F. Klawonn, *"Foundations of Fuzzy Systems"*'. John Wiley & Sons, 1994.

[13] W. Pedrycz, F. Gomide, *"An introduction to Fuzzy Sets: Analysis and Design"*. A Bradford Book. The MIT Press, Massachusetts, 1998.

[14] Chuen Chien Lee; *Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I* IEE Transactions on Systems, Man, and Cybernetics, Vol 20. No. 2, 1990

[15] Chuen Chien Lee; *Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part II* IEE Transactions on Systems, Man, and Cybernetics, Vol 20. No. 2, 1990.

[16] Angulo Usategui; Angulo Martínez; *Microcontroladores PIC Diseño Práctico de Aplicaciones*. Mc. Graw Hill. México 1999.

[17] Angulo Usategui; Angulo Martínez, Romero Yesa; *Microcontroladores PIC Segunda Parte. PIC16F877*. Mc. Graw Hill. México 2000.

[18] Felipe Gonzáles G; *Alimentación de los Robots, Parte I*; Electrónica y Computadores, No 44; Cekit. 1998.

[19] Felipe Gonzáles G; *Alimentación de los Robots, Parte II*; Electrónica y Computadores, No 45; Cekit. 1999.

[20] Felipe Gonzáles G; *Alimentación de los Robots, Parte III*; Electrónica y Computadores, No 46; Cekit. 1999.

[21] Felipe Gonzáles G; *Sensores para los Robots, Parte I*; Electrónica y Computadores, No 47; Cekit. 1999.

[22] National Semiconductor 2002; Analog / Interface Databook, Winter 2002.

[23] Custom Computer Services, INC. PIC C Compiler. Reference Manual. September 2002.

[24] http://www.epsg.upv.es/WEBESCUELA/SERVICIOS/UGI/grupo6.htm

[25] http://www.astic.es/nr/astic/boletic-todos/boletic24/artimono2.pdf

[26] http://polaris.lcc.uma.es/~ppgg/FSS/FSS1.pdf

[27] http://www.minirobotica.org