INSTITUTO TECNOLOGICO Y DE ESTUDIOS
SUPERIORES DE MONTERREY
CAMPUS MONTERREY
DIVISION DE INGENIERIA
PROGRAMA DE GRADUADOS EN INGENIERIA

TECNOLÓGICO
DE MONTERREY.

PROYECTO AVANZADO DE MECATRONICA

REPORTE FINAL

PRESENTADO COMO REQUISITO PARCIAL
PARA OBTENER EL GRADO ACADEMICO DE:

ESPECIALIDAD EN MECATRONICA

POR:

RODOLFO REYES AGUILLON

MONTERREY, N. L. DICIEMBRE DE 2003

INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

DIVISION DE INGENIERIA

PROGRAMA DE GRADUADOS EN INGENIERIA

TECNOLÓGICO DE MONTERREY®

PROYECTO AVANZADO DE MECATRONICA

REPORTE FINAL

PRESENTADO COMO REQUISITO PARCIAL
PARA OBTENER EL GRADO ACADEMICO DE:

ESPECIALIDAD EN MECATRONICA

POR:

RODOLFO REYES AGUILLON

MONTERREY, N. L. DICIEMBRE DE 2003

Proyecto Avanzado de Mecatrónica

Especialidad en Mecatrónica

Instituto Tecnológico y de Estudios Superiores de Monterrey

por

Rodolfo Reyes Aguillón

Diciembre de 2003

Proyecto Avanzado de Mecatrónica

por

Rodolfo Reyes Aguillón

Reporte Final

Presentado al Programa de Graduados en Ingeniería

Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Monterrey

como requisito parcial para obtener el grado académico de

Especialidad en Mecatrónica

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Diciembre de 2003

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

División de Ingeniería y Aquitectura

Programa de Graduados en Ingeniería

Los miembros del comité de tesis recomendamos que la presente tesis de Rodolfo Reyes Aguillón sea aceptada como requisito parcial para obtener el grado académico de Especialidad en Mecatrónica.

Comité de tesis:

Dr. Carlos Pfeiffer Celaya
Asesor de la tesis

Dr. Antonio Favela Contreras
Sinodal

Ing. Ricardo Ramírez Mendoza
Sinodal

Dr. Federico Viramontes Brown
Director del Programa de Graduados
en Ingeniería

Diciembre de 2003

A Rosa María Ramírez Prado

Reconocimientos

El autor desea expresar su más sincera gratitud a todas aquellas personas que de una u otra forma hicieron posible la culminación de esta tesis.
Antes que a nadie, a Rosa María Ramírez Prado por el amor y paciencia.
A sus padres, familiares y amigos por el apoyo que siempre han demostrado.
Al cuerpo de profesores del Programa de Graduados en Ingeniería, en especial, a los señores: Dr.Federico Viramontes, Dr. Carlos Narváez, Dr.Alex Elías, Dr.Horacio Martínez, Dr.Graciano Dieck, Dr. Jorge Limón, Sra. Petra Kosikov, Dr.Noel León, Ing. Miguel Ramírez, Dr.Alfonso Avila y al Dr.Mario Alberto Hernández por todas sus valiosas enseñanzas.
A su asesor, el Doctor Carlos Fernando Pfeiffer por su apoyo incondicional, sus comentarios y el tiempo dedicado.
A sus sinodales, el Dr.Antonio Favela y al Dr.Ricardo Ramírez por su orientación.
A los compañeros de trabajo: Ing.Aline Drivet, Ing.Jorge Lozoya, Ing.Israel Escamilla, M.C.Raúl Estrada, Ing. Ramiro Alcaraz, M.C.Luis Rosas, M.C.Enrique Martínez, Sr.Juan Pineda y al M.C.Armando Céspedes por su amistad y ayuda.
En general a todo el personal del Instituto Tecnológico y de Estudios Superiores de Monterrey por las facilidades prestadas.
Y a Dios por haber permitido la satisfactoria terminación de estos estudios y por haber concedido el placer de convivir con todas estas personas.
Gracias por su apoyo.

RODOLFO REYES AGUILLÓN

Instituto Tecnológico y de Estudios Superiores de Monterrey
Diciembre 2003

Resumen

El desarrollo de robots exploradores por parte de las diferentes universidades en Aguascalientes, les ha permitido promocionar sus carreras de ingeniería ante la sociedad en general al participar en concursos nacionales e internacionales. Aunque el ITESM Campus Aguascalientes no cuenta con desarrollos en esta área, ya se está viendo la necesidad de contar con un sistema robot de bajo costo que sirva como basé para el desarrollo de proyectos mecatrónicos en los cuales se puedan involucrar estudiantes de nivel preparatoria y profesional en todos sus niveles.

Este documento presenta un primer acercamiento al diseño y construcción de un robot móvil automatizado que permita servir como base para el desarrollo de futuros proyectos.

Basado en esquemas de control distribuido y utilizando herramientas de fácil acceso y bajo costo se muestran al lector las fases de diseño y las áreas de conocimiento involucradas en el desarrollo de un proyecto mecatrónico.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Descripción del proyecto

Construir un robot explorador de bajo costo que sirva como base para el desarrollo de proyectos de ingeniería en el área de la mecatrónica, que pueda funcionar como plataforma para que estudiantes de nivel primaria y secundaría puedan desarrollar y ejecutar misiones científicas en sus casas y patios. Todo lo anterior, para estimular en los jóvenes el interés por la ciencia y la tecnología, y alentar en los estudiantes de ingeniería la curiosidad e investigación en las tecnologías que hacen posible la exploración científica del espacio y nuestro planeta.

1.2. Problema

En la actualidad el estado de Aguascalientes cuenta con 14 universidades[3], 6 de estas manejan la mayor parte de la matricula estudiantil y por ello cada vez se hace más necesario marcar la diferencia en un mercado tan competitivo.

Desde el año de 1997 la Universidad Bonaterra[8], Campus Aguascalientes de la Universidad Panamericana, participa en el *Walking Machine Challenge* organizado por la *Society of Automotive Engineers* (**SAE**) y a partir del año 2001 se unieron a este concurso el Instituto Tecnológico de Aguascalientes (*Tecnológico Federal*) y la Universidad Autónoma de Aguascalientes (*Universidad Pública*). Desde entonces cada año estas tres universidades del estado de Aguascalientes mandan un contingente de estudiantes a participar en este concurso y han tenido participaciones destacadas.

De las tres universidades restantes sólo una, la Universidad del Valle de México, no tiene carreras de ingeniería y las otras dos, el ITESM Campus Aguascalientes y la Universidad Tecnológica de Aguascalientes, aunque tienen la carrera de **Ingeniero en Mecatrónica** no participan por no contar con desarrollos en el área de robótica.

Si bien aun no es del interés del Campus Aguascalientes el participar en el *Walkin Machine Challenge*, si lo es el aumentar la captación de estudiantes hacia las área ingenieriles y en especial para la carrera de Mecatrónica. Es por ello que desde el mes

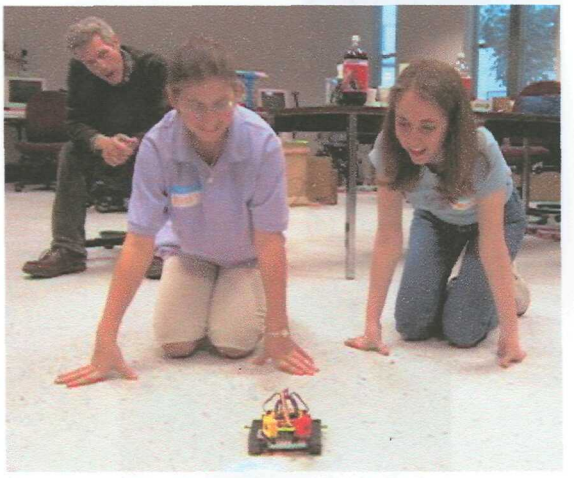


Figura 1.1: Lego Mindstorm

de Agosto del 2003 se empezaron a planear una serie de actividades y talleres enfocados a estimular en los estudiantes de bachillerato y secundaria el interés por las ciencia y la tecnología.

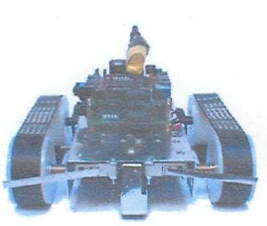
Como primer paso se compraron sistemas *Mindstorm* de *Lego*[9] que permiten a los jóvenes estudiantes construir sistemas mecánicos, neumáticos y electrónicos de forma fácil y divertida. A la vez de dar al estudiante conocimientos básicos sobre mecánica, neumática, electrónica y automatización, se estimula la curiosidad y el interés por la ingeniería. Si bien estos sistemas están pensados para estudiantes de nivel secundaría [5], su aplicación y nivel de dificultad es variable y aplicable incluso para tesis doctórales.

El segundo paso, sería pues desarrollar esta plataforma robótica para exponerla como base de desarrollo de futuros trabajos en el área de automatización. Aunque no tan modular como el *lego*, al ser un desarrollo de bajo costo serviría como caso de estudio en las diferentes materias de la carrera de Mecatrónica y se esperaría que los estudiantes de niveles avanzados hicieran mejoras y anexiones al diseño para ir construyendo un sistema propietario. Sistema que a su vez serviría para que los estudiantes de nivel básico y medio desarrollen prácticas y experimentos[4].

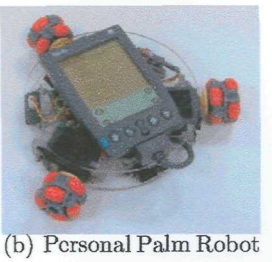
1.3. Objetivo

Construir un sistema robot de bajo costo que permita al usuario diseñar y ejecutar misiones científicas en un ambiente casero. Una misión típica podría ser el visitar cada día el mismo lugar y hacer mediciones de temperatura y niveles de luz.

La programación del sistema se haría por medio de una computadora y el usuario



(a) Robot Clónico



(b) Personal Palm Robot

(c) Cuadrúpedo Articulado

Figura 1.2: Ejemplos de robots exploradores.

tendrá la opción de seleccionar entre dos modalidades de manejo: **Básico**, para los usuarios que sólo quieren programar misiones con las funciones de movimiento hacia adelante, hacia atrás y hacer giros rotando los neumáticos delanteros. Y una segundo modalidad será la **avanzada** que permitirá el usuario modificar y monitorear los parámetros de control de motores del robot, permitiendo diseñar nuevos comportamientos.

1.4. Desarrollo

El desarrollo de este proyecto requiere de una plataforma con estabilidad dinámica, capaz de desplazarse dentro del ambiente dinámico de una casa, oficina o escuela; un sistema de control de motores; y una interfase de interacción humano-robot.

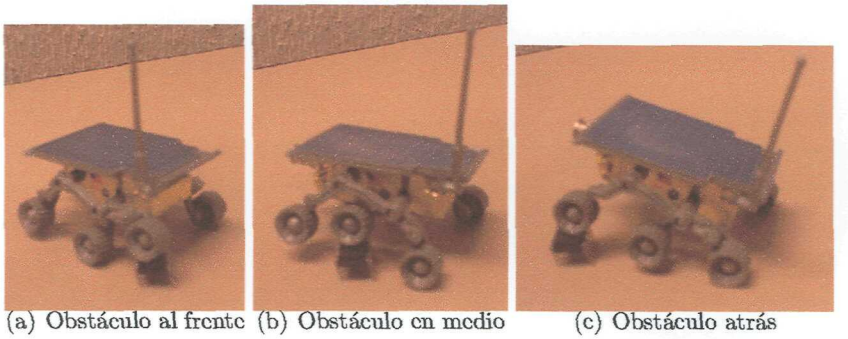
Los pasos para desarrollar este explorador incluyen el diseño de un chasis; construcción de las tarjetas de control de motores; montaje de dispositivos y sensores; programación, demostración y evaluación del software de control de misión; así como la programación del software de control de bajo y alto nivel.

Capítulo 2

Diseño Conceptual

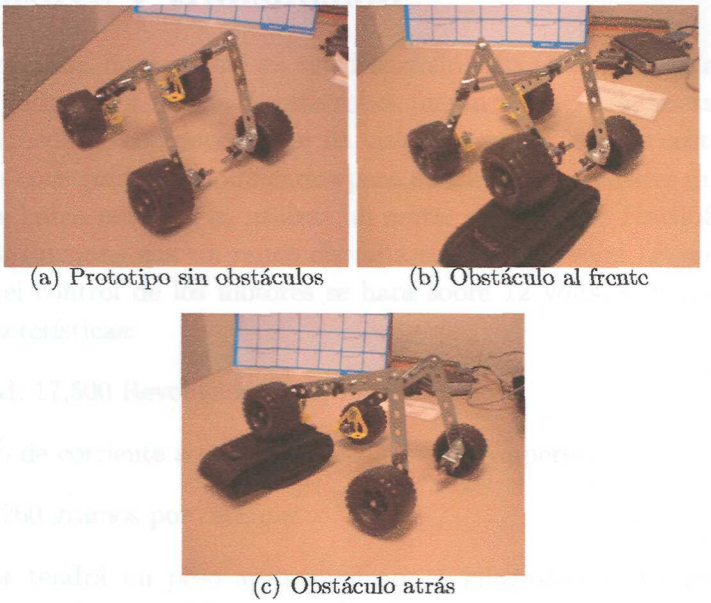
El presente diseño se basa en el *Mars Pathfinder* del *Jet Propulsion Laboratory* de la *NASA*[10]. A diferencia del modelo original, este diseño esta pensado para desempeñarse dentro de un ambiente casero y por tanto su dimensiones, potencia de sus motores y capacidades para evadir obstáculos han sido reducidas. El modelo propuesto tendrá cuatro patas y no seis como el original. Cada pata tendrá un sistema de propulsión independiente, pero sólo las patas delanteras podrán dirigir sus neumáticos en sentido izquierda–derecha.

La figura 2.2 muestra un modelo del prototipo a escala, construido con un *Meccano*, en donde se puede ver un chasis en forma de escuadra con un neumáticos en cada extremo. El chasis, conformado por dos escuadras unidas por un eje móvil, nos permite tener un grado de flexibilidad para permitir una mejor adherencia de los neumáticos al suelo aun y cuando una parte del robot se encuentre sobre un obstáculo.



(a) Obstáculo al frente (b) Obstáculo en medio (c) Obstáculo atrás

Figura 2.1: Modelo del Pathfinder de *Mattel*



(a) Prototipo sin obstáculos (b) Obstáculo al frente

(c) Obstáculo atrás

Figura 2.2: Prototipo del robot armado en *Meccano*

Capítulo 3

Diseño a detalle

3.1. Motores y transmisión

Bajo la premisa de que este es un desarrollo educativo, se había buscado inicialmente que todos los sistemas de este robot fueran de hechura casera, sin embargo durante la búsqueda de componentes se ha visto que es mucho más costeable comprar subsistemas ya construidos y modificarlos para cubrir las necesidades que tenemos.
Inicialmente se había pensado en utilizar un motor de *hobby* de **RadioShack** modelo *273-256* el cual funciona con un rango de voltaje de 9 a 12 volts *(Figura 3.1 b)*. Pensando en que el control de los motores se hará sobre 12 volts, el motor presenta las siguientes características:

- Velocidad: 17,500 Revoluciones por minuto.
- Consumo de corriente a máxima eficiencia: 1.9 Amperes.
- Torque: 260 gramos por centímetro.

Como el robot tendrá un peso aproximado de 1 kilogramo, el torque requerido es superior al torque obtenido del motor, además se requerirá que los neumáticos giren a velocidades más lentas de las 17,500 revoluciones por minuto; una tren de engranes solucionaría estos problemas.

3.1.1. Trenes de engranes

Un *engrane* es un elemento que permite la transmisión de movimiento mediante el continuo empalme de sus dientes. Los engranes transmiten movimiento entre ejes rotatorios ya sea que los ejes de entrada y salida se encuentren en línea recta o perpendiculares entre sí.
Los engranes se utilizan en aplicaciones en donde la potencia del eje de entrada debe ser transferida con eficiencia, reemplazando las superficies rotatorias por ruedas dentadas para eliminar derrapes *(Figura 3.1a)* (pérdidas).

En un juego de dos engranes ensamblados la relación entre sus velocidades angulares ω es igual a la relación entre sus radios.

$$\frac{\omega_{engrane2}}{\omega_{engrane1}} = \frac{r_{engrane1}}{r_{engrane2}}$$

Radio Shack vende su motor de *hobby* con un engrane de 2.5mm de radio. Los neumáticos a utilizarse son unas llantas de **Meccano** de 4.5cm de diámetro y se esperaría que el robot se desplace a una velocidad suficientemente lenta (1 revolución por segundo en cada neumático). Si substituimos estos datos en la formula obtenemos que:

$$\frac{292RPS}{1RPS} = \frac{radio}{5mm}$$

$$radio = 292 \times 5 = 1460mm = 146cm = 1{,}46mts$$

Esto es, ¡el engrane deberá medir un metro y medio de radio!. Sin embargo se esperaría que el robot entero no mida mas de 30 centímetros de largo, por ello se hace necesario contar con un **tren de engranes**.
Un tren de engranes es un grupo de engranes que en conjunto harán el trabajo de reducir la velocidad del eje a un nivel aceptable. En nuestro caso la reducción requerida es de 292:1 y la relación de velocidades se calcula de la siguiente manera:

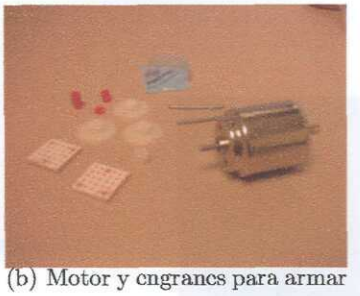
$$\frac{\omega_1}{\omega_3} = \frac{\omega_1}{\omega_2}\left(\frac{\omega_2}{\omega_3}\right)$$

Así se requerirían 3 engranes de 8.5cm de radio, si se incrementa el número de engranes el radio se disminuye drásticamente. Comercialmente la marca **Proam** de México comercializa el engrane número #070 el cual es un engrane doble. El engrane exterior tiene un diámetro de 16.6mm y el engrane interior tiene 6.5mm. Un sólo engrane tiene la capacidad de reducción de 2.5:1, y tres engranes tendrían la capacidad de reducción de 277:1, que es muy aproximado a la reducción requerida.
Se armó una primera caja de engranes utilizando alambre recocido y material electrónico, pero se vio que es difícil ensamblar los engranes de forma que no queden muy justos pero que también no queden muy flojos y derrapen.
Otra opción disponible fue la utilización de **motores reductores**.

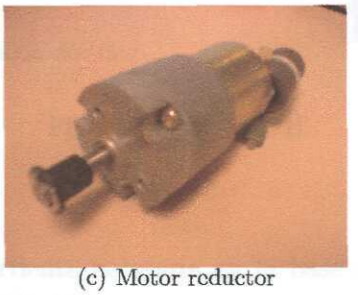
3.1.2. Motores Reductores

Un motor reductor es un motor común de corriente directa que tiene en un extremo una caja de engranes *(Figura 3.1 c)*. En algunos mercados de desechos industriales es común encontrar este tipo de motores, con la desventaja de que no se cuenta con especificaciones del fabricante. En este caso se compraron 6 motores marca **Tsukasa Electronic Co, LTD** de fabricante Japonés. Aunque no se tienen especificaciones oficiales en pruebas de laboratorio se han obtenido las siguientes lecturas:

(a) Motor y transmisión de un juguete



(b) Motor y engranes para armar



(c) Motor reductor

Figura 3.1: Transmisiones estudiadas

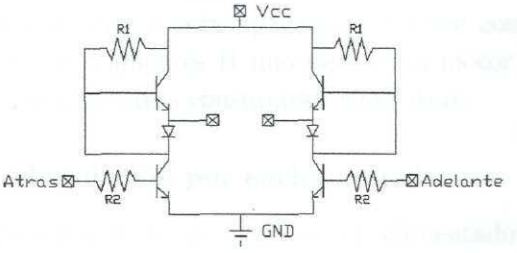
- Voltaje de operación: 24 volts.
- Consumo: 0.5 Amperes.
- Velocidad: 900 Revoluciones por minuto.
- Torque: 1 kilogramo por centímetro

Dado que el robot va a funcionar con 12 volts la velocidad del motor se reduce a cerca de 450 revoluciones por minuto o lo que es lo mismo 7 revoluciones por segundo, haciendo posible la conexión directa del neumático al eje del motor.

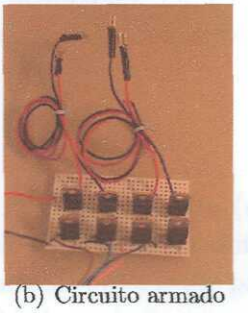
3.1.3. Control de motores

El circuito de control del motor (*puente H*) se muestra en la figura 3.2. Construido con transistores TIP–41 las resistencias tienen que calcularse para que el transistor pueda drenar la corriente suficiente para que el motor funcione.
El TIP–41 tiene las siguientes características:

- Voltaje máximo: 100 volts.
- Corriente máxima de colector: 6 Amperes.



(a) Diagrama cléctrico



(b) Circuito armado

Figura 3.2: Puente H

- h_{FE}: 15.

Hay que recordar que la corriente necesaria en la base del transistor para que este funcione como switch se calcula de

$$I_B = \frac{I_C}{\beta}$$

en donde $\beta = h_{FE} = 15$ y la corriente de colector sería igual a la corriente máxima que requiere el motor, que en nuestro caso podría ser limitada a un ampere. Así la formula queda como

$$I_B = \frac{1}{15} = 67mAmperes.$$

Del puente, las dos resistencias inferiores que conectan a las etiquetas **Atrás** *y* **Adelante** *serán alimentadas por la etapa de control con señales de 5 volts. Por ley de Ohm:*

$$V = R \times I$$

despejando

$$R = \frac{V}{I} = \frac{5volts}{67mAmp} = 75Ohms$$

mientras que las resistencias superiores se alimentan del mismo circuito de motores que tiene un voltaje de 12 volts

$$R = \frac{12volts}{67mAmp} = 179Ohms$$

El circuito alimenta al motor en directa cuando se aplica una señal lógica de *1* en la entrada **Adelante** y de *0* en la entrada llamada **Atras**. El sentido se invierte el invertir el estado de las entradas. Si por algún motivo ambas entradas permanecen en

0 o en *1* el motor simplemente se detiene. En este circuito no hay posibilidad de corto circuito por falla en el control.

Cuando el motor esta apagado el puente consume $134mAmp$. Para este robot se construyeron 6 puentes H uno para cada motor y cuando todos los motores estén apagados estos circuitos consumirán $800mAmp$.

PWM – Modulación por ancho de pulso

Nuestro motor de 24 volts al ser alimentado con 12 volts reduciría (*al menos teóricamente*) su velocidad y fuerza en un 50 % pues sólo dispone de la mitad de la alimentación nominal. Así si requiriéramos que el motor marche lentamente podríamos alimentarlo con poco voltaje y si quisiéramos gran velocidad o más fuerza podríamos aumentar el voltaje en sus terminales. Sin embargo resulta difícil poder modificar el voltaje de alimentación del motor. Por lo tanto tenemos que buscar métodos alternativos que nos permitan manipular la velocidad y fuerza de nuestros motores.
El control por modulación de ancho de pulso o **PWM** *Pulse Width Modulation*, consiste en alimentar un motor de corriente directa con una serie de pulsos cuyo ancho es variable y proporcional a la potencia alimentada (*Figura* 3.3). Esto es, si nuestro motor es alimentado cada segundo con pulsos cuyo ancho miden 750mSeg, observaríamos que nuestro motor recibe alimentación sólo durante el 75 % del tiempo y el restante 25 % esta apagado, así se podría decir que el motor esta siendo alimentado con el 75 % del voltaje nominal. Aun y cuando cada pulso tiene una altura igual al voltaje nominal del motor, la duración del pulso determina la cantidad de energía efectiva que recibe el dispositivo.
El fabricante de los motores **Tsukasa** hace la aclaración de que algunos de sus modelos contienen filtros que evitan que el ruido generado por las escobillas pueda interferir con las etapas de control, sin embrago estos filtros pueden causar que ciertas frecuencias utilizadas como control *PWM* sean eliminadas. Así en pruebas de laboratorio, y al no tener especificaciones, se pudo determinar que para frecuencias muy bajas (*menores a 60 Hz.*) el motor brinca violentamente; para frecuencias medias (*de entre 60 y 200 Hz*) el motor se comporta bien; para frecuencias medias–altas (*de entre 200 y 1000Hz*) el motor se comporta errático debido a los filtros internos y finalmente para frecuencias altas (*mayores a 1kHz*) el motor empieza a funcionar como bocina y genera ruido de igual frecuencia al del PWM.
Por lo anteriormente expuesto se determino que la frecuencia base más eficiente sería de 100Hz (*se alimentará el motor 100 veces cada segundo*), y los pulsos podrán durar de entre 1 % (*una milésima de segundo*) y 100 % (*un segundo*) de la frecuencia base.
Así las terminales denominadas *Adelante* y *Atrás* del puente H en vez de recibir una señal constante de activación, recibirán un tren de pulsos (*PWM*) para indicar la velocidad deseada al frente o en reversa.

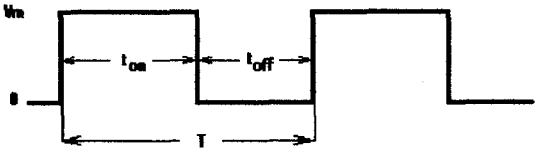


Figura 3.3: Forma de onda del control PWM

3.2. Tacómetros y sensores

Dado que el robot necesita conocer el estado de cada uno de sus neumáticos en todo momento se decidió dotar al sistema con un tacómetro en cada llanta. La función del tacómetro es la de mantener una cuenta del número de revoluciones que ha dado cada neumático.

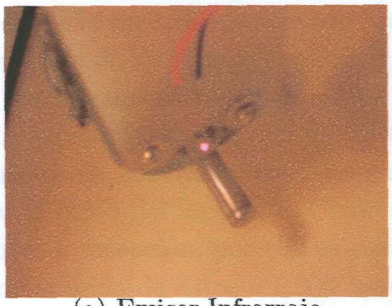
Para mantener esta cuenta se pueden construir varios esquemas. El más utilizado es el de adherir un imán al eje del neumático y mediante un sensor de efecto Hall obtener un pulso cada vez que el imán pasa frente al sensor.

Otro esquema menos utilizado, al menos en esta clase de sistemas, pero más económico y de mejor *resolución* consiste en utilizar un emisor y un receptor infrarrojos alineados entre sí y entre los dos posicionar una barrera que contienen pequeñas *ventanas*.

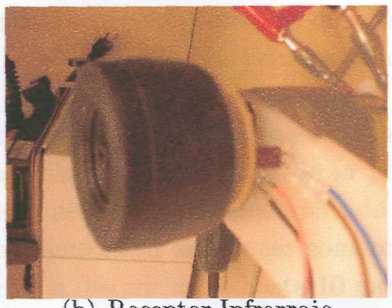
Por resolución nos referimos a que dado que el neumático tiene un radio de 4 cm su circunferencia medirá $\pi \times$ *diametro*, esto es 12.57 cm. Si nuestro tacómetro nos da un pulso cada 12.57cm no podría yo *ver* movimientos de mi neumático menores a los 12cm. En el caso del sensor de efecto Hall este problema se puede solucionar poniendo más imanes en el eje de forma que una revolución en vez de dar un pulso ahora da dos o tres. El principal problema es que el número de imanes que caben en el eje del neumático es finito.

Para el caso de los sensores infrarrojos la solución es hacer más *ventanas* en la barrera, aquí no hay tanto problema pues la ranura mínima posible para los sensores es muy delgada ya que estamos utilizando haces de luz. Así si utilizamos una barrera que contiene 10 ventanas ahora puedo *ver* movimientos en mi neumático que hasta 1.3 cm.

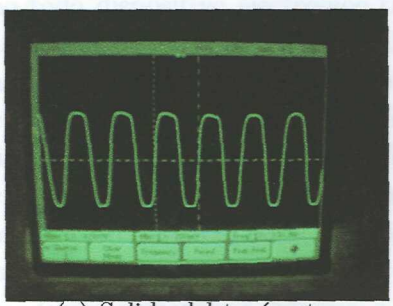
Para el caso especifico del robot se utilizó un pedazo de rollo de película fotográfica de 35 milímetros que tiene en cada costado una serie de ranuras que sirven para que el rollo avance dentro de la cámara fotográfica. En una sección de 12.57 cm de rollo hay 23 ranuras en promedio por lado. Así que si se enrolla esta sección de película dentro del neumático se obtiene una barrera con 23 *ventanas*. El emisor infrarrojo se posiciona muy cercano al eje del neumático, mientras que el receptor se pone fuera de la barrera viendo hacia el eje de la llanta. Con estas 23 ranuras tenemos la posibilidad de ver



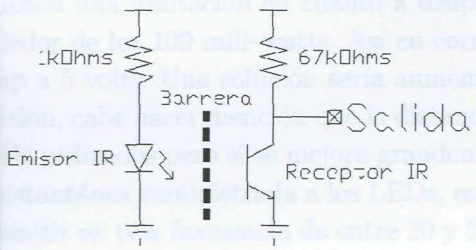
(a) Emisor Infrarrojo



(b) Receptor Infrarrojo



(c) Salida del tacómetro



(d) Diagrama eléctrico

Figura 3.4: Tacómetro

movimientos de mi neumático de hasta 5.46mm.

Cabe hacer mención que una gran desventaja de esta clase de sensores es la interferencia que pueden producir fuentes externas de infrarrojo tales como el sol y las lámparas fluorescentes.

3.3. Comunicación Inalámbrica

3.3.1. Enlace Infrarrojo

El primer esquema de comunicación inalámbrica que se probó fue la comunicación mediante haces de luz infrarroja. Este tipo de comunicación demostró ser económico, rápido y eficiente dentro de condiciones ambientales controladas.

La transmisión infrarroja no presenta mayor problema. Basta conectar un led al puerto de salida de la computadora y codificar, por ejemplo, un *1 lógico* como *luz* y un *0 lógico* como *oscuridad*. El principal problema reside en la recepción.

Hay que recordar que al igual que en el tacómetro una de las principales fuentes de interferencia infrarroja son los sistemas de iluminación fluorescente y el sol, así que para poder discriminar la comunicación de la computadora de la interferencia externa se procede a modular la señal de comunicación dentro de una portadora. Para la mayoría

de las aplicaciones en donde se requieren comunicaciones a más de 30 cm de distancia se utiliza como fuente un *LED* conmutado a una frecuencia portadora de entre 20kHz y 50kHz. La mayoría de los electrodomésticos utilizan frecuencias cercanas a los 35kHz para modular sus comunicaciones.
Así la etapa de recepción tendrá que filtrar su entrada para dejar pasar sólo la componente alterna de la comunicación y después filtrar la frecuencia de la portadora, excluyendo así el ruido del ambiente. Como se puede ver, se ha complicado un poco la etapa de envío y recepción, por ello es común encontrar soluciones comerciales que permiten la transmisión de datos a 5,000 *baudios* a distancias de hasta 10 metros.
Para aplicaciones en donde se requiere superar los 10 metros se tiene que aumentar la potencia administrada a los LEDs emisores. Hay que tomar en consideración de los LEDs como todo dispositivo electrónico tienen una limitación en cuanto a disipación de potencia que en este caso sería de alrededor de los 100 mili–watts. Así en corriente continua no se pueden superar los 20mAmp a 5 volts. Una solución sería aumentar el número de LEDs utilizados para la transmisión, cabe hacer mención que la distancia no aumenta linealmente con el número de LEDs utilizados pero si se mejora grandemente.
Otra solución sería aumentar la potencia instantánea suministrada a los LEDs, esto es, ya que estamos modulando la señal a transmitir en una frecuencia de entre 20 y 50kHz y suponiendo que esta modulación se hace con una señal de onda cuadrada con 50 % de ciclo útil, podemos aumentar a 200 mili–watts el consumo asumiendo que el LED pasa la mitad del tiempo prendido y la otra mitad apagado, permitiendo que disipe calor y no se sobre caliente. Para rangos de 50kHz es permisible suministrar hasta un ampere a cada LED sin que exista riesgo alguno.
Otras soluciones sería poner lentes frente a los emisores y receptores, pero actualmente resulta difícil encontrar comercialmente este tipo de piezas.
Para evaluar esta opción se construyó un emisor que modula la señal del puerto serie de la computadora a una frecuencia de 35kHz y que produce velocidades de hasta 8800 baudios. Hay que recordar que un baudio **no** es un bit por segundo, así la velocidad máxima alcanzada fue de 4,800 bps a un distancia de 3 metros. El receptor estaba constituido por un foto–transistor y su polarización, un amplificador operacional que funcionaba como filtro pasa banda sintonizado a 35kHz y un segundo amplificador operacional 1:100 que amplificaba la portadora. Finalmente se construyo un demodulador digital con la ayuda de un microcontrolador.
Las desventajas de este esquema fueron:

- El infrarrojo no puede traspasar objetos sólidos, por tanto la utilización de este sistema esta limitada a que el emisor (Computadora) y receptor (Robot) estén dentro de la misma habitación.

- La velocidad de comunicación es muy lenta. En promedio el robot requiere de 19,200 bps para mantener un control aceptable de sus funciones. De las pruebas

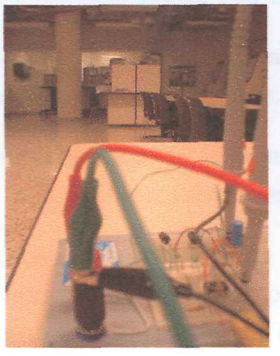


Figura 3.5: Enlace Infrarrojo

desarrolladas en el laboratorio se calculó que se requerirían 38,200 baudios y una portadora de 100kHz. Lamentablemente esta frecuencia es muy elevada para ser manejada por el foto–transistor.

3.3.2. Radio Frecuencia

Si bien los sistemas de control a radio frecuencia no ofrecen velocidades de transmisión muy altas, si ofrecen distancias de cobertura muy grandes y la posibilidad de traspasar paredes.
Comercialmente se encontraron tres opciones.

- Construir un modem inalámbrico y utilizar frecuencias de FM.
- Construir un modem inalámbrico y utilizar radios de dos vías.
- Comprar los módulos de radio frecuencia para transmisión inalámbrica.

Construir un modem inalámbrico y utilizar frecuencias de FM.

Se construyó un transmisor de *FM*, para fines de prueba se transmitió un tono continuo y se verificó cual era la frecuencia más alta que podía captar un radio comercial. El resultado fue que el tono más alto era de 3kHz y la distorsión era mínima. Se procedió a generar un circuito digital que modulaba pulsos de ancho de un segundo en una portadora de 3kHz y se observo que el radio comercial recibía la señal con gran distorsión. Si la misma señal se modulaba en una portadora senoidal de la misma frecuencia la recepción era exitosa.
Después de dos días de prueba se determinó que:

- La velocidad de transmisión era muy lenta, 750 baudios o 300 bps.
- La distancia no era muy alta, pues aunque podía traspasar las paredes del laboratorio no podía llegar más allá de los 10 metros.
- Es difícil encontrar en Monterrey espacio libre en la banda de FM en donde la señal de las radiodifusoras no signifique interferencia alta para la transición.

Para tratar de aumentar la velocidad de transmisión se intentó utilizar un emisor y receptor de un coche de control remoto que utiliza frecuencias de 37Mhz. Lamentablemente se observó que esta clase de sistemas utilizan el esquema de modulación CW (Continous Wave) que solo permite velocidades de transmisión de 5 bps. El CW es muy similar a la clave Morse utilizada en los telégrafos.

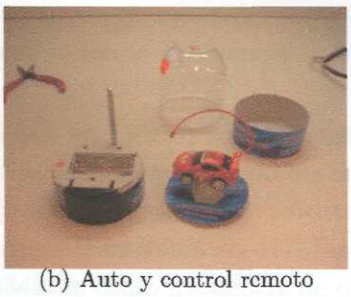
Construir un modem inalámbrico y utilizar radios de dos vías.

Se utilizó el mismo circuito del sistema anterior, pero esta vez se acopló a un radio Motorola de dos vías que tiene un alcance aproximado de 1 kilómetro, la frecuencia más alta del tono transmitido fue de 25kHz lo que permitiría velocidades digitales de hasta 6,250 baudios o 2,400 bps.
La distancia máxima alcanzada no se registro pues surgieron dificultades que hicieron desechar esta idea. Entre otras la principal desventaja fue que para transmitir hay que presionar el botón *PPT* del radio y liberarlo para recibir información. El tiempo requerido por el radio para pasar de recepción a transmisión fue de 10 milisegundo. Así para transmitir un comando al robot consistente en 10 bits se requerirían 10 milisegundos para empezar transmisión, 4.2 milisegundos para transmitir los 10 bits y 10 milisegundos más para regresar al modo de recepción, lo que nos da una capacidad real de comunicación de 413 bps.

Comprar los módulos de radio frecuencia para transmisión inalámbrica.

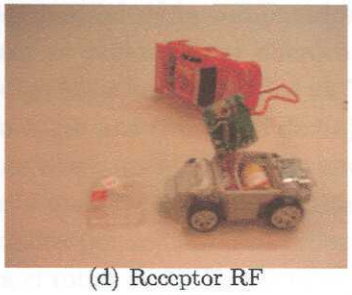
Para esta sección se utilizaron los módulos TLP-434 y RLP-434 de Laipac Technology, Inc. El fabricante especifica velocidades de transmisión de hasta 2400 bps y distancias de cobertura de hasta 100 metros. Estos módulos con entradas y salidas digitales modulan las señales bajo el esquema ASK y son utilizados principalmente para sistemas de telemetría.
La modulación ASK utiliza tonos en frecuencias audibles para los humanos.
En pruebas de laboratorio se observó que los 2400 bps a los que hacía referencia el fabricante no significaban *bits por segundo* sino **baudios por segundo** lo que nos reduce el ancho de banda a tan solo 1200 bits por segundo. Aunque esta velocidad es sólo la mitad de lo que nos dijo el fabricante sigue siendo una buena opción dado que sería el enlace más rápido alcanzado hasta el momento.

(a) Auto de Juguete

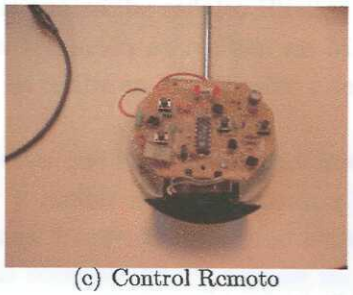


(b) Auto y control remoto

(c) Auto destapado

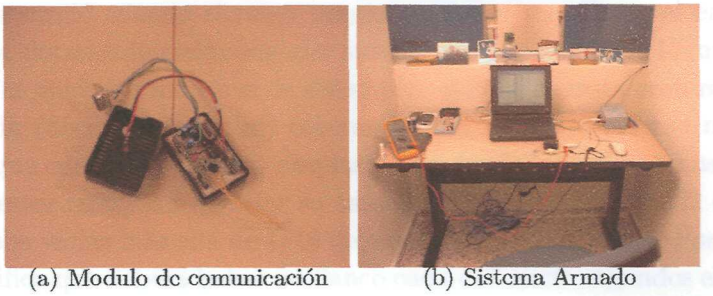


(d) Receptor RF



(c) Control Remoto

Figura 3.6: Auto de Control Remoto



(a) Modulo de comunicación (b) Sistema Armado

Figura 3.7: Sistema de Radio Frecuencia

Al comprar los módulos se nos informó que en México solo se venden en frecuencias de 434MHz. Según la Secretaría de Comunicaciones y Transportes de México, esta frecuencia es de uso libre (no se requiere licencia) y esta relativamente vacía en México. Los radios Motorola utilizados anteriormente transmiten en 417Mhz y existen algunos teléfonos inalámbricos que utilizan los 434Mhz como frecuencia de operación.

Dado que la señal proveniente del puerto serial de la computadora no se puede alimentar directamente al módulo transmisor se programó un microcontrolador para que codificara la señal en Manchester. El código Manchester codifica un "1"lógico como una transición de alto a bajo y un "0"lógico como una transición de bajo a alto. Cada código enviado requiere de un baudio así que teóricamente se puede enviar un bit por baudio. En pruebas de laboratorio se observó que la eficiencia del sistemas era de un 80Teniendo un emisor y un receptor construidos por separado se procedió a construir un único sistema emisor–receptor tanto para el robot como para la computadora. En el caso de la computadora el sistema recibe señales RS–232 y las codifica en Manchester para su transmisión, para la recepción se reciben en Manchester y se decodifican a RS–232. En el robot la señal se recibe en Manchester y se decodifica a I^2C y viceversa para su transmisión.

Armados los sistemas se observo que el sistema tiene que ser *half–duplex* dado que el sistema no puede transmitir y recibir al mismo tiempo pues tanto el robot como la computadora utilizan la misma frecuencia para sus comunicaciones. Asumiendo que se pudieran tener más de una computadora y más de un robot, se programó el micro con el protocolo **CDMA** así que cualquier dispositivo tiene que escuchar si nadie esta utilizando la frecuencia y luego transmite. Si dos dispositivos transmiten al mismo tiempo por error, cada equipo al detectar la transmisión del otro marcará una **colisión** abortando la comunicación y esperando un tiempo aleatorio para intentar de nuevo.

El primer problema surgió del tiempo de latencia de la señal, esto es, el tiempo que tarda en viajar desde el emisor hasta el receptor mas lejano. Esto hizo especialmente tardado detectar una colisión si el emisor y el receptor en colisión están relativamente alejados.

Otro problema surgió cuando el robot comienza a moverse. Tanto el emisor como el receptor fueron dotados inicialmente con antenas de 1/4 de onda (17cm de longitud) pero se tuvieron muchos problemas de distorsión, así que se instalaron nuevas antenas de 1/2 de onda (35cm de longitud) esperando una mejora, pero eso no sucedió. El problema era que cuando el robot se desplaza el ancho de los pulsos transmitidos como código Manchester empiezan a variar su tamaño. El protocolo RS–232 especifica que un pulso no debe varias más allá de 1/16 del ancho de pulso nominal. Por ejemplo para 2,400 bps significa que deberá haber un flanco cada 417 micro segundos en el código. Si el flanco se encuentra 27 micro segundos antes o después de donde debería estar toda la transmisión se vuelve ilegible, y eso es lo que pasaba cuando el robot se encuentra en movimiento.
La primera opción fue reducir la velocidad de transmisión a 1,200 bps lo que nos da 52 micro segundos de margen. Con una tasa de 95siguiente fase que era probar el sistema con los protocolos completos.
El problema aquí fue el mismo que se presento con los radios Motorola. Para establecer exitosamente la portadora de transmisión se requieren entre 5 y 7 milisegundos, una vez establecida la comunicación se envía un byte que marca el inicio de la comunicación y luego se envían los datos que deberán ser más de 2 bytes y luego otro byte más que marca el final de la transmisión. 7 milisegundos de inicio más 16 milisegundos en los bytes de inicio y fin de transmisión nos dan 2 bytes cada 23 milisegundos o 347 bps. Los bytes de inicio y fin de comunicación son necesarios pues muchas veces el primer y último bytes de la transmisión se pierden pues apenas se esta encendiendo la circuiteria de transmisión o bien ya se encuentra en proceso de ser apagada. Este protocolo se llama **KISS** y es muy utilizado en las transmisiones digitales en radios de aficionados.

3.3.3. IRDA-SIR

El protocolo IRDA desarrollado desde 1993 por la *INFRARED DATA ASSOCIATION* busca establecer y promover estándares de comunicación inalámbrica por medio de rayos infrarrojos.
Este protocolo consta de las siguientes capas:

- Capa física.
- Capa de acceso al medio.
- Capa de enlace.

La capa física provee de la señalización requerida por los circuitos emisores y receptores de los dispositivos para establecer una comunicación exitosa. Establece que el rango típico para este tipo de comunicación es de 2 metros para dispositivos conectados a la

línea eléctrica, o bien se puede optar por una versión de bajo consumo cuyo alcance es de apenas 30 centímetros.
La comunicación puede ser bi–direccional y la velocidad puede varias de entre 9,600 bps y los 115kbps para la versión de baja velocidad, o 1.152Mbps a 4Mbps para la versión de alta velocidad. En esta última se utiliza el esquema de **CRC** para proteger la integridad de los datos.
La capa de acceso al medio provee las reglas necesarias para que se dé una comunicación ordenada y confiable entre los dispositivos.
Por último la capa de enlace provee del manejo de múltiples capas de acceso en una sola capa física.
Sobre estas tres capas pueden correr uno o más de los siguientes protocolos:

Tiny TP Provee control de flujo en las comunicaciones. Opcionalmente provee de servicios de segmentación y rearmado de datos.

IrComm Servicio de simulación de puertos seriales y/o paralelos sobre puertos de comunicación infrarrojos.

OBEX similar al protocolo HTTP.

Irda–Lite Permite manejar menos sobre carga en la transferencia de datos manteniendo compatibilidad con el estándar.

IrTran–P Utilizado en cámaras digitales permite compartir imágenes entre dispositivos.

IrMC Para servicio a teléfonos celulares, permite intercambiar números telefónicos o datos de agenda.

IrLAN Describe el acceso a redes de datos mediante el puerto infrarrojo.

Para el robot se optó por desarrollar una capa física con las siguientes características:

- Velocidad de comunicación de 19,200 bps.
- Alcance máximo limitado a 10 centímetros debido a que se utilizaron emisores y receptores comunes (como los utilizados para los controles remotos de los televisores).
- Half–duplex.
- La información es codificada de acuerdo al estándar. Se utilizan pulsos de luz cuya duración es igual a 3/16 del ancho del pulso normal de un bit.
- Un **1** lógico se codifica como un pulso de luz, mientras que un **0** lógico se codifica como obscuridad.

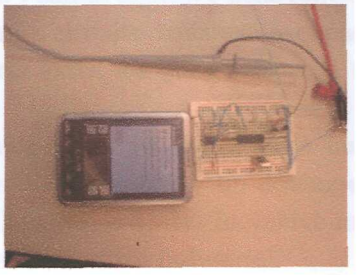


Figura 3.8: Enlace IRDA–SIR

- Para marcar el inicio y el fin de un byte se sigue el estándar RS–232.

La capa de acceso al medio se desarrolló de la siguiente manera:

- El maestro (*PALM*) siempre inicia la comunicación.
- Un paquete puede estar compuesto de 1 o 2 bytes de información.
- El esclavo (*Robot*) puede responder con un byte de reconocimiento o bien con un byte de reconocimiento más otro byte con datos.
- El maestro especifica en el primer byte que clase de paquete se esta enviando.

El sistema carece de capa de enlace.
El receptor además de cumplir con el protocolo IRDA también cumple con el protocolo I^2C pues es capaz de hacer la traducción de las ordenes de la computadora al robot.

3.4. Microcontrol

3.4.1. Selección

Para el desarrollo de este proyecto se optó por utilizar los microcontroladores *PIC* del fabricante **Microchip**.
La familia PIC esta compuesta por microcontroladores de 8 bits del tipo **RISC**. Estos microcontroladores tiene sólo 32 instrucciones y corren con relojes de 4 a 20MHz, lo que los hace capaces de ejecutar entre 1 y 5 millones de instrucciones por segundo.
Aunque existen modelos que incluyen entradas análogas y sistemas de comunicación I^2C o RS232, para este proyecto se utilizaron sólo los modelos **16f84**. El **PIC16F84** es un microcontrolador empaquetado en *PDIP* de 18 patas completamente protegido contra descargas electroestáticas. Incluye 68 bytes de RAM y 64 bytes de EEPROM,

además de 13 pines de entrada y salida digital. Puede funcionar con osciladores de cristal de cuarzo o bien con un simple circuito resistencia–capacitor. Fuera del oscilador no requiere de ninguna circuiteria extra así que su sistema mínimo es muy económico y fácil de armar.
La memoria de programa mide $1K \times 14$ siendo cada instrucción de 14 bits. El PIC provee de una compresión de código de 2:1 y un desempeño de velocidad de 4:1. Con un consumo de apenas 10mili–watts a 4MHz lo vuelve uno de los microcontroladores más utilizados en aplicaciones de control industrial.
En este proyecto en particular, se construyeron dos sistemas mínimos iguales, uno para el control de PWM y el otro para el control de Tacómetros. Un tercer sistema mínimo se construyo con ligeras modificaciones para contener el sistema de comunicaciones del robot.
Todos los sistemas mínimos corren software escrito específicamente para su aplicación, pero comparten la misma librería de comunicación. A continuación una breve descripción de estos programas.

3.4.2. Librerías de Comunicación: *I2C.ASM*

Dado que el microcontrolador tendrá muchas tareas que ejecutar en forma *simultanea* se optó por seleccionar un protocolo de comunicación que permitiera el manejo de velocidades de transmisión dinámicas, es decir, que el canal de comunicación trabaje a la velocidad del dispositivo más lento y que esta velocidad pueda variar dependiendo de la cantidad de trabajo que tenga que hacer ese dispositivo lento
La transmisión serial **asíncrona** tal como el puerto serie de la computadora, nos permite utilizar un cable para comunicar dos o más dispositivos. Este cable es utilizado por uno de los dispositivos para mandar un mensaje y luego es dejado libre esperando recibir la respuesta. Si se utilizan dos cables, uno de ellos sirve para mandar mensajes y el otro para recibir respuestas, pero queda limitada la comunicación a solo dos dispositivos.
La velocidad de transferencia de los datos puede seleccionarse típicamente de entre 9600 bps (*bits por segundo*) hasta 115200bps. Una desventaja grande es que la velocidad a la que va a funcionar el canal se negocia al inicio de la comunicación y una vez empezada ya no hay forma de indicarle a los demás dispositivos el posible cambio de velocidad.
Si se utilizara una velocidad de transferencia muy alta y el microcontrolador tiene muchas tareas que ejecutar, el micro no va a tener suficiente tiempo para verificar la comunicación por estar atendiendo sus demás tareas y podría perder datos. Para que esto no suceda tenemos que buscar un protocolo de comunicación que nos permita in-

dicarle al dispositivo emisor del mensaje que estamos ocupados, que requerimos de más tiempo y que envíe su mensaje mas despacio, pero que cuando no tengamos muchas distracciones los mensajes fluyan mucho más rápido.
El protocolo I^2C desarrollado por *Philips* es un protocolo de comunicación diseñado para el intercambio de datos entre circuitos integrados dentro de un mismo aparato electrónico. I^2C es un protocolo serial **síncrono**, esto es, describe un canal de comunicación que consta de dos cables, uno lleva los datos y el otro lleva una señal de reloj que sincroniza la transmisión y la recepción de mensajes entre dispositivos que trabajan a distintas velocidades. Entre las características más importantes se encuentran:

- Sólo se requieren dos líneas de comunicación (*cables*); una línea serial de datos llamada **SDA** y una línea serial de señal de reloj llamada **SCL**.
- Cada dispositivo conectado al canal tiene una dirección única definida por software y se establece una relación maestro–esclavo durante todo el tiempo que dura la comunicación.
- La comunicación se basa en paquetes de 8 bits de longitud y permite transferencias bi–direccionales de hasta 100kbps (*miles de bits por segundo*) en modo estándar, hasta 400kbps en modo rápido o hasta 3.4 Mbps (*Millones de bits por segundo*) en el modo de alta velocidad.
- El número máximo de dispositivos conectados al mismo canal depende únicamente de la capacitancia máxima aceptable que es de 400pF.

Aunque un canal serial tiene mucho menor capacidad de transferencia que un canal paralelo, el canal serial reduce el cableado así como el número de conexiones en cada circuito.
Los dispositivos conectados a un canal serial deben seguir una conjunto de reglas o *protocolo* que les permita evitar cualquier confusión, perdida de datos o bloqueo de información.
Para el protocolo I^2C el **maestro** es aquel dispositivo que inicia la comunicación y genera la señal de reloj. Todos los demás dispositivos conectados al canal son considerados como **esclavos**. Tanto maestros como esclavos pueden mandar y recibir información, pero es responsabilidad del maestro generar en todo momento la señal de reloj, la cual sólo puede ser modificada cuando un dispositivo lento requiere de más tiempo para procesar el dato en el canal.

Ambas líneas de comunicación SDA y SCL son bi–direccionales y están conectadas a la terminal positiva de alimentación por resistencias *pull–up*. Cuando el canal esta libre, ambas líneas se encuentran en nivel **ALTO**. La conexión de cada dispositivo a las líneas del canal debe ser de *colector abierto* para permitir la construcción de una *and alambrada*.

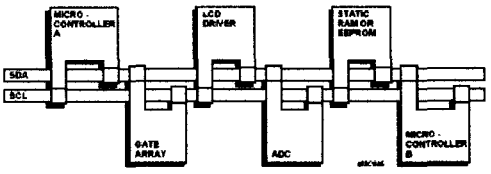


Figura 3.9: Ejemplo de canal I2C

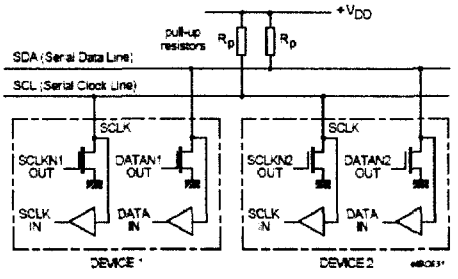


Figura 3.10: Circuito eléctrico de comunicación

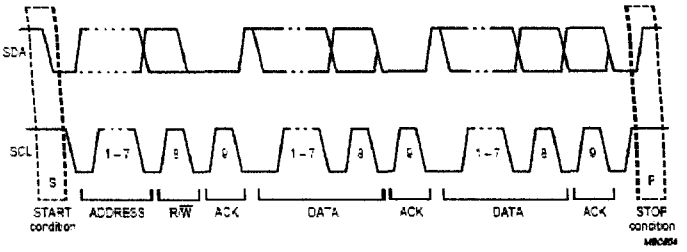


Figura 3.11: Diagrama de una transferencia completa

Los niveles lógicos de operación quedan definidos como:

- **0 volt** para nivel *BAJO*
- **5 volts** para nivel *ALTO*

Se generará un pulso de reloj por cada bit que se ponga en el canal.
La información en la línea SDA debe permanecer estable mientras la línea de SCL se encuentre en ALTO. El nivel de ALTO o BAJO de la línea de datos sólo puede cambiar mientras la señal de SCL este en BAJO.
Con lo anterior en mente, I^2C define las siguientes condiciones:

Start : Inicio de transmisión.
Se da cuando ocurre un cambio de nivel ALTO a BAJO en la línea de SDA mientras SCL permanece en estado ALTO.

Stop : Fin de transmisión.
Ocurre cuando hay un cambio de nivel BAJO a ALTO en la línea de SDA mientras SCL permanece en estado ALTO.

Las condiciones de START y STOP son generadas por el maestro. El canal se considera ocupado después de ocurrido un START y así permanecerá hasta que ocurra un STOP. Cada byte puesto en SDA debe ser de 8 bits de longitud. No hay restricción en cuanto al número de bytes enviados por transferencia. Cada byte debe ser acompañado por un bit de reconocimiento generado por el dispositivo receptor. La información se envía comenzando con el bit más significativo. Si algún esclavo no puede continuar con la comunicación o requiere más tiempo para procesar un byte, podrá mantener la línea SCL en BAJO para forzar al maestro a detener la comunicación hasta que la línea sea liberada por el dispositivo lento.

Para el reconocimiento el maestro genera un pulso extra de reloj y pone la línea SDA en ALTO, si algún esclavo reconoce la recepción del byte jalará la línea SDA a

BAJO y la mantendrá así durante el tiempo que dure el pulso de reloj. Si el maestro genera un mensaje pero no recibe reconocimiento de alguno de los bytes, el mensaje será termina con un condición STOP y se volverá a retransmitir el mensaje o se abortará la tarea según sea el caso.
Para este proyecto existen 3 tipos de mensajes:

Comando . El maestro genera un byte y lo transmite. El maestro espera reconocimiento.

Transmisión de un byte . El maestro manda un mensaje consistente en un dirección única de 8 bits, espera reconocimiento y después manda 8 bits de datos.

Recepción de un byte . El maestro envía una dirección única de 8 bits, espera reconocimiento y luego libera SDA para que el esclavo direccionado ponga 8 bits de datos en el canal. El esclavo espera reconocimiento del maestro.

3.4.3. PWM

El sistema mínimo que ejecuta este programa tiene capacidad de controlar la velocidad 4 motores así como su sentido de giro. Además de controlar los motores deberá estar atento a la comunicación en el bus I^2C por lo cual también ejecuta la librería de comunicaciones antes descrita.
Para generar el tren de pulsos de ancho variable (PWM), se programó una interrupción al micro cada 100 microsegundos. Esto nos permite tener una portadora de 100Hz (frecuencia de control del motor) y poder dividirla en 100 partes iguales. Así tenemos la capacidad de alimentar al motor con una frecuencia base de 100Hz variable desde un ciclo útil del 0 al 100 %.
Cada 100 microsegundos el micro ejecuta su interrupción y en ella determina si cada motor debe o no seguir siendo alimentado. Para ello cada motor tiene una localidad de memoria asignada en el micro en donde se almacena el porcentaje de ciclo útil que debe ser alimentado. Adicionalmente se lleva un contador que se incrementa de 0 a 100 cada vez que se ejecuta la interrupción y el cual sirve como patrón de comparación para determinar si se continua alimentando o no a cada motor.
El micro tiene 13 pines de entrada y salida. Para esta aplicación se requieren 4 pines para la comunicación I^2C y 8 pines (2 para cada motor) para el control de motores.
Se requieren 2 pines para el control del motor, pues como ha de recordarse se requieren dos señales de control en el puente H3.2 para controlar el sentido de giro. El pin que quedó libre se utilizo como señal de diagnóstico para fines de prueba en laboratorio.
Este sistema reconoce 9 comandos:

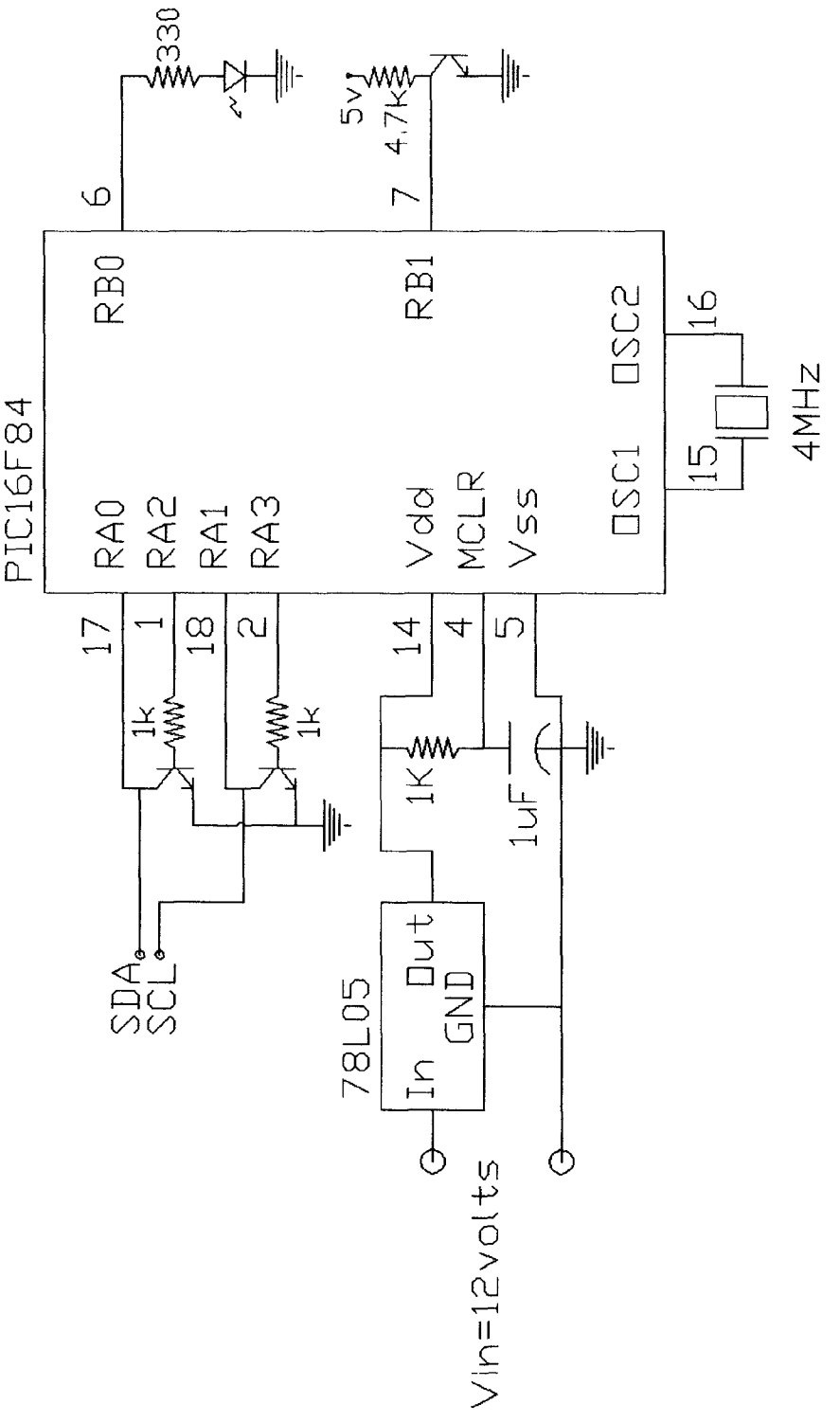


Figura 3.12: Sistema Mínimo de Comunicación

Cargar valor para motor recibe un byte con valor base 80 hexadecimal seguido de un byte cuyo valor se encuentra entre 0 y 100 decimal y que representa el valor del PWM al cual debe alimentarse dicho motor.

- *Valor 81 Hexadecimal para motor 1.*
- Valor 82 Hexadecimal para motor 2.
- Valor 83 Hexadecimal para motor 3.
- Valor 84 Hexadecimal para motor 4.

Verificación de valor para motor recibe un byte con valor base 46 hexadecimal, a continuación el sistema pone en el bus el valor almacenado en la posición de memoria PWM del motor seleccionado.

- Valor 47 Hexadecimal para motor 1.
- Valor 48 Hexadecimal para motor 2.
- Valor 49 Hexadecimal para motor 3.
- Valor 4A Hexadecimal para motor 4.

Verificación de encendido recibe un byte de valor 05 Hexadecimal y el sistema responde con un reconocimiento.

3.4.4. Tacómetros

Tomando como base el software del sistema mínimo del PWM, se modificó para hacer las veces de control de tacómetros.
Al igual que en el módulo anterior, el control de tacómetros tiene que ejecutar su función además de mantener la comunicación con el bus I^2C por lo cual también ejecuta la librería de comunicaciones expuesta anteriormente.
Similar al PWM, el control de tacómetros puede monitorear hasta 4 motores. Para ello se genera una interrupción cada milisegundo y se verifica si hubo algún cambio en los pines asignados a tacómetro. Hay que recordar que estos pines están conectados a un fototransistor que recibe los pulsos de luz provenientes del encoder de cada neumático3.4. Cada motor tiene asignados dos localidades de memoria, una en donde se lleva la cuenta temporal del numero de cambios ocurridos hasta el momento y la otra en donde se almacena la última cuenta fija alcanzada. Cada 200 interrupciones se detiene la cuenta progresiva, de divide sobre dos (ya que se contaron tanto los flancos ascendentes como los descendentes) y el resultado se pasa a la localidad de memoria asignada a cada motor.
Aunque podría pensarse que cabe la posibilidad de que el micro pierda la cuenta de los

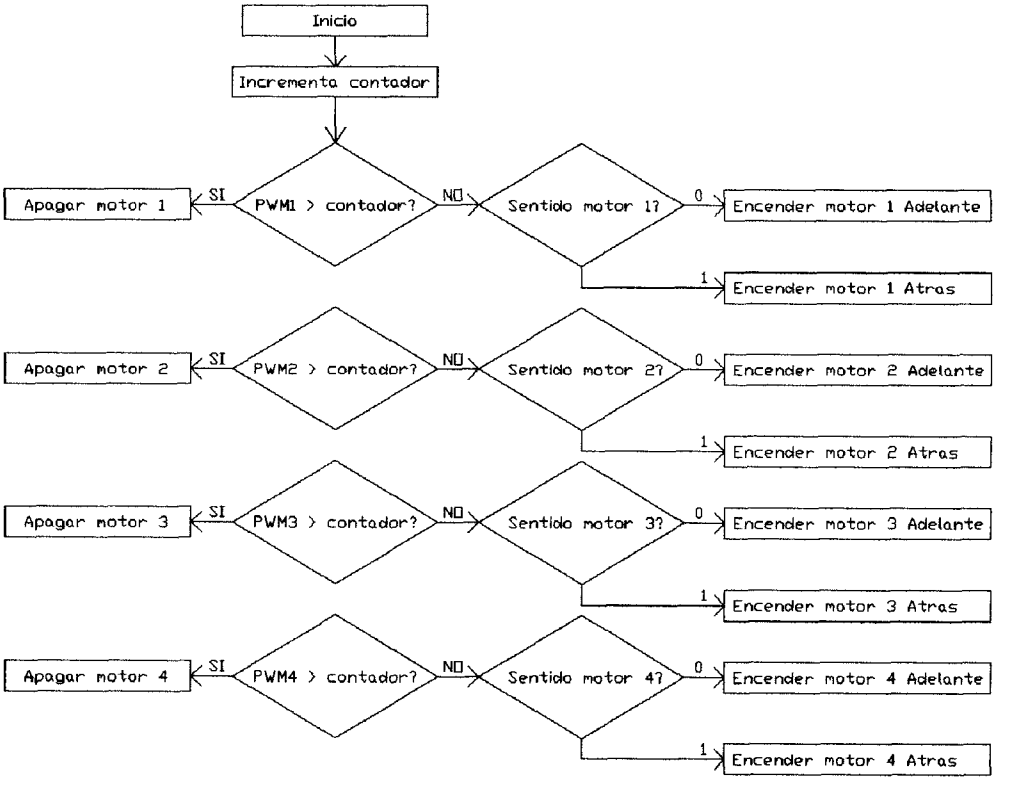


Figura 3.13: Diagrama de flujo de la interrupción del PWM

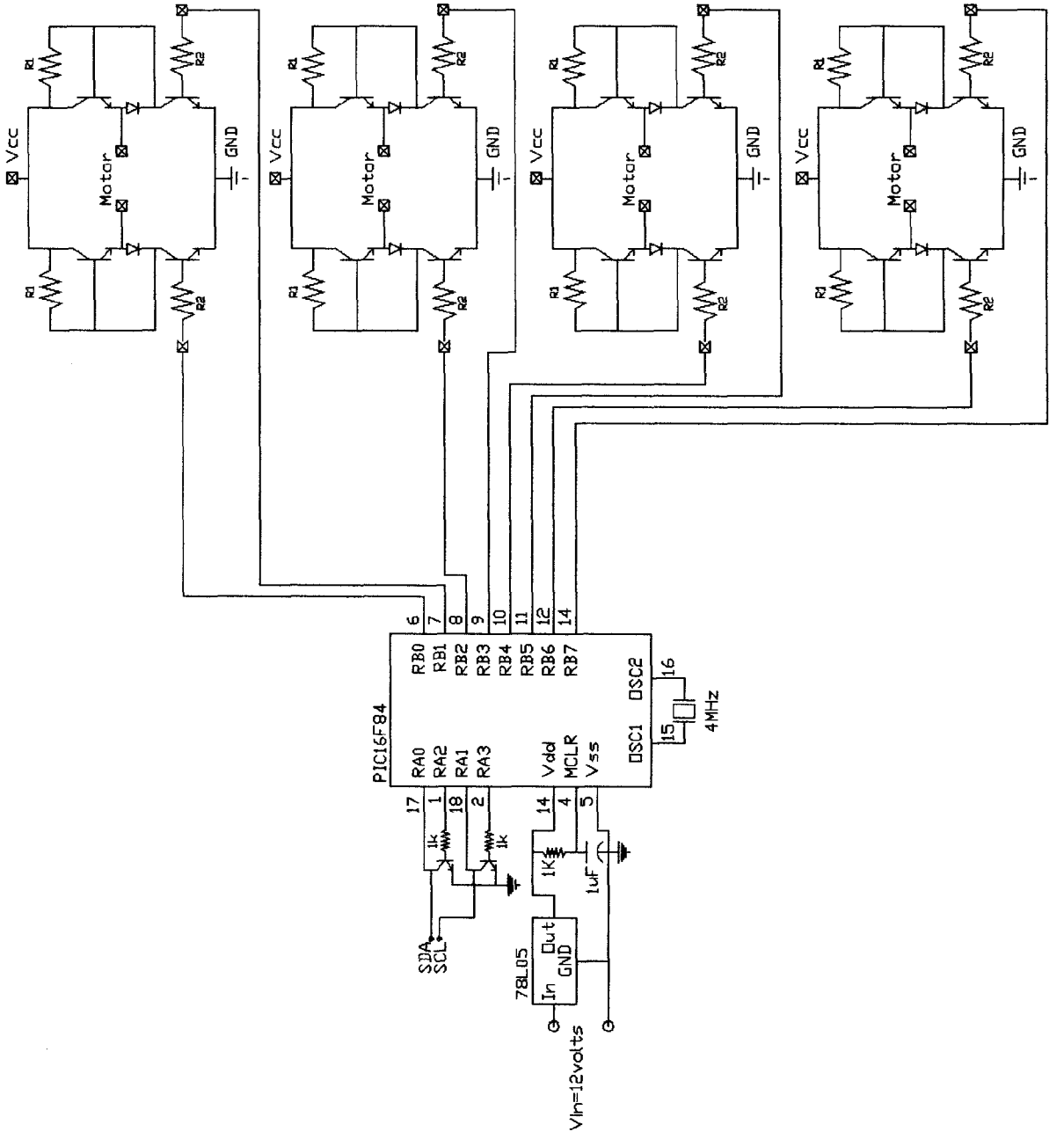


Figura 3.14: Sistema mínimo de control de motores.

pulsos de los tacómetros debido a su tiempo de muestreo y a la comunicación con el bus, hay que recordar que cada neumático tiene 23 ranuras en su encoder y la velocidad máxima del motor es de 7 revoluciones por segundo, lo que nos da 161 pulsos por segundo o un pulso cada 6 milisegundos por cada motor.
Cabe hacer mención que el sistema esta preparado para soportar los 4 encoders bidireccionales, pero por cuestiones de tiempo y sencillez por ahora solo se implementaron encoders unidireccionales.
En promedio el bus fija un bit cada 200 microsegundos pero puede ser frenado para que el control de tacómetros pueda verificar los pulsos entre bit y bit que envía o recibe.
Este sistema reconoce los siguientes comandos:

Cuenta tacómetro Recibe un byte con valor base 50 hexadecimal y luego el sistema pone la última cuenta fija del encoder del motor especificado.

- Valor 51 Hexadecimal para motor 1.
- Valor 52 Hexadecimal para motor 2.
- Valor 53 Hexadecimal para motor 3.
- Valor 54 Hexadecimal para motor 4.

Verificación de encendido Recibe un byte con valor 15 hexadecimal y a continuación el sistema responde con un reconocimiento.

3.4.5. Comunicación inalámbrica

Los sistemas anteriores fueron diseñados bajo el concepto de módulos independientes y re– utilizables. Así que aunque sólo hacen una tarea la hacen independientemente de los demás subsistemas del robot.
Bajo la misma filosofía se diseñó este módulo de comunicaciones. Pensado para traducir los mensajes provenientes del puerto IRDA–SIR de la PALM al bus I^2C del robot, este módulo recibe y decodifica los mensajes del maestro para determinar cuantos bytes debe recibir y cuantos debe esperar en el puerto I^2C para regresar al maestro.
Dado que el módulo es independiente de los demás subsistemas, este no sabe que tipo de información esta intercambiando el maestro con los tacómetros o los controles de motor. Así se diseño un sistema a nivel de enlace que permite mantener una comunicación ordenada entre robot y computadora3.3.3. Este protocolo codifica el primer byte recibido de la siguiente forma:

Bit 7 . Número de bytes a recibir por el esclavo.

- 0 – 1 byte.
- 1 – 2 bytes.

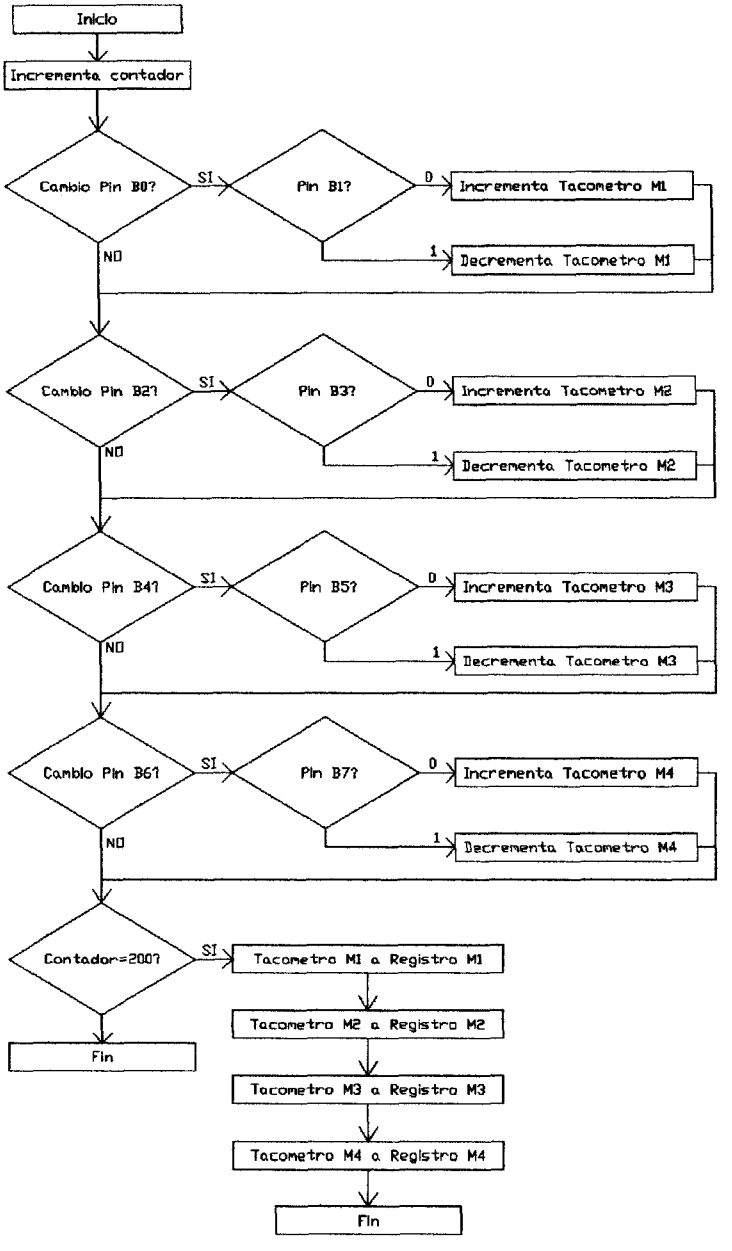


Figura 3.15: Diagrama de flujo de la interrupción del tacómetro

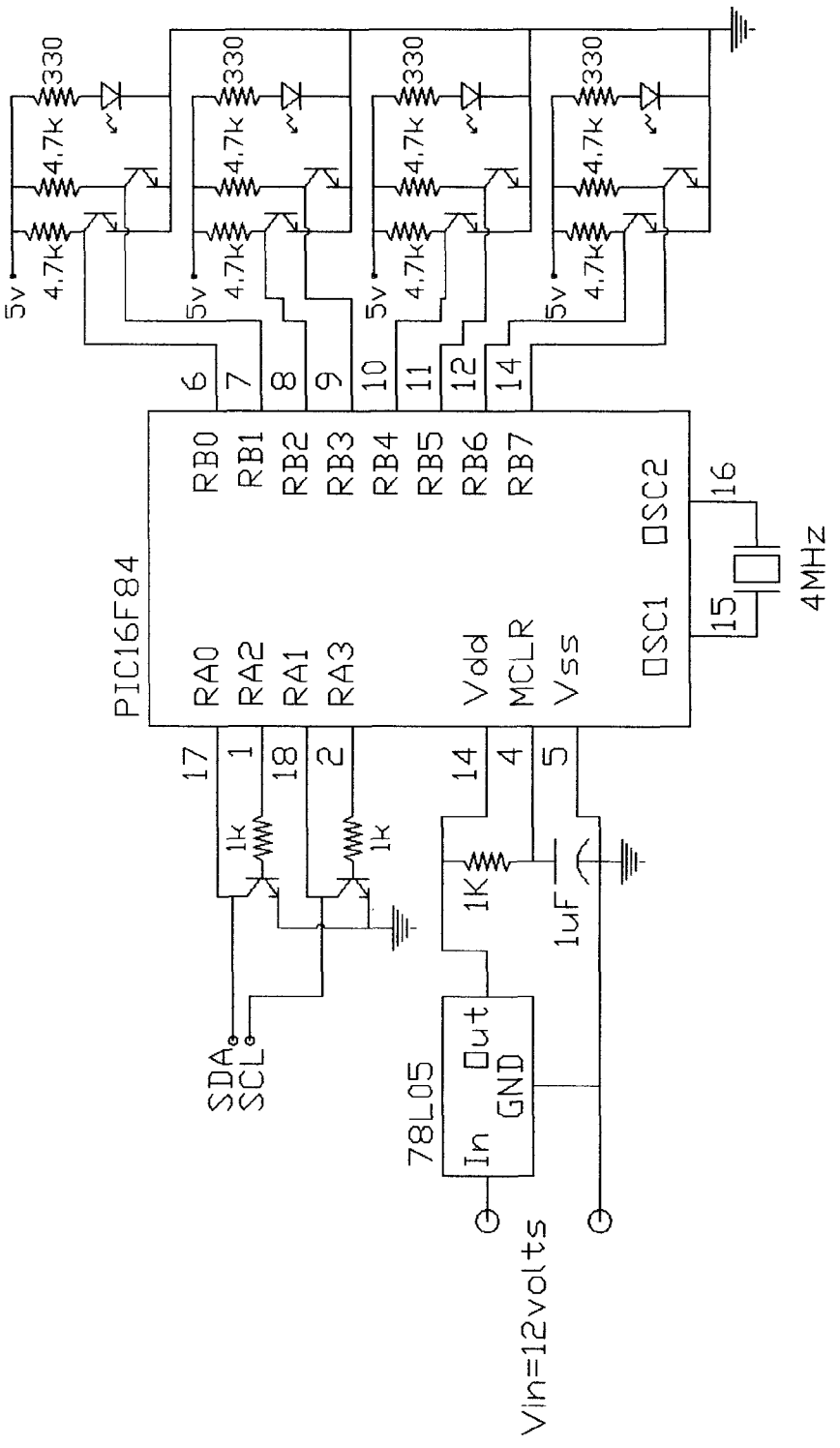


Figura 3.16: Sistema Mínimo de tacómetros

Bit 6 . Número de bytes a enviar al maestro.

- 0 – 1 byte. Generalmente un reconocimiento.
- 1 – 2 bytes. Un reconocimiento y un byte de respuesta.

Bit 5–4 . Identificador de subsistema, 2 bits o 4 direcciones únicas.

Bit 3–0 . Comando, hasta 16 comandos por subsistema.

La combinación de 4 direcciones únicas y 16 comandos por dirección nos dan hasta 64 posibles comandos en un número indeterminado de subsistemas (no limitado a 4).

3.5. Chasis

Como se había explicado anteriormente el chasis de robot se basó en el diseño del *Pathfinder* de la *NASA*[10]. Aunque el modelo original tiene 6 patas y en cada pata tiene un neumático, el diseño presentado sólo tiene 4 patas y conserva la funcionalidad y control independiente de cada neumático.
Las dimensiones del robot se tomaron considerando que debía ser capaz de subir y bajar escalones, por lo cual su largo total debería ser menor a 30cm (que es la medida promedio de un escalón) y su ancho quedaba a discreción, pues su única limitante es la estética y el tamaño de los componentes que carga.
El robot parado sobre sus patas posteriores debería medir más de 15cm, dado que se había pensado que la mejor geometría para un sistema de este tipo es una escuadra, uno de sus catetos mediría 15cm pero la hipotenusa no debería exceder los 30 cm. Así de la formula de Pitágoras

$$h^2 = c1^2 + c2^2$$

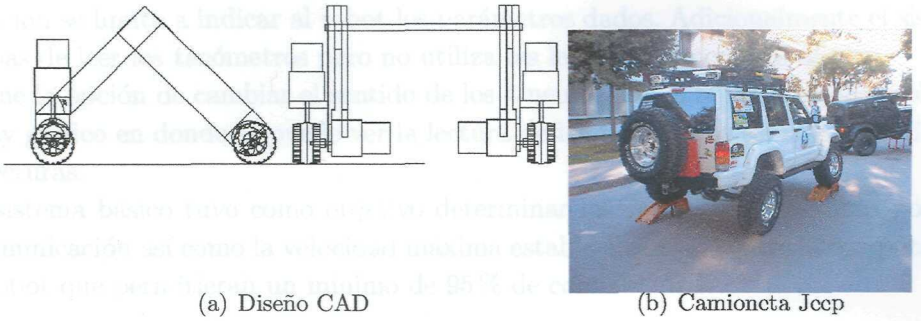
se obtiene que

$$30^2 = 15^2 + c2^2$$

o

$$c2 = sqrt900 - 225 = 25{,}98$$

Así el chasis del robot quedó constituido por una triángulo con un cateto de 15cm, otro de 26cm y longitud total de 30cm. En esta escuadra se fijan dos motores. Dos estructuras son necesarias para acomodar los cuatro motores así que ambas estructuras se unieron por una barra de 1/4 de pulgada permitiendo así que el chasis se pudiera flexionar ante obstáculos pero manteniendo cierta rigidez en el diseño.
La construcción del chasis se hizo en plástico *Nylamin* que ofrece dureza y facilidad de maquinado además de ser económico y fácil de conseguir en el mercado. Finalmente se dispusieron dos placas más de plástico para fijar el resto de los componentes electrónicos. Estas placas también están sujetas a la barra de unión y tienen, al igual que las escuadras, capacidad de movimiento ante los obstáculos.



(a) Diseño CAD (b) Camioncta Jeep

Figura 3.17: Diseño de Chasis

3.6. Interfaces de usuario

3.6.1. Windows

En una primera etapa se desarrollaron los programas de diagnóstico bajo la plataforma de Windows en PC. En total se escribieron 4 programas y una librería. Todos los programas fueron escritos en **Visual Basic** de **Microsoft** dadas las facilidades que este ambiente de desarrollo ofrece. La librería fue compilada como DLL (librería de acceso dinámico) así que independientemente del lenguaje que se utilice para desarrollar la aplicación, esta librería puede seguir siendo accesible.
La librería contiene todas las rutinas que hacen posible el manejo del protocolo I^2C mediante el puerto paralelo de la computadora. La velocidad máxima alcanzada por el bus de comunicación depende de la velocidad con que el sistema operativo pueda accesar el puerto, pues aunque el puerto paralelo tiene una velocidad tope de casi 200kbps sólo se pueden alcanzar velocidades de 115kbps o menos. La computadora conectada al robot alcanza una velocidad típica de 6kbps.
Los 4 programas de diagnostico son:

- Sin control.bas
- Control on off.bas
- Control P.bas
- miniP.bas

Sin control.bas

Este programa fue el primer intento por interconectar la computadora con el robot. Aquí el usuario indica el porcentaje de PWM que deberá ser aplicado a cada motor y la aplicación se limita a indicar al robot los parámetros dados. Adicionalmente el sistema es capas de leer los tacómetros pero no utiliza las lectura para control alguno.
Se tiene la opción de cambiar el sentido de los 4 neumáticos a la vez y se desarrollo un display gráfico en donde se puede ver la lectura de los tacómetros así como las últimas 100 lecturas.
Este sistema básico tuvo como objetivo determinar las velocidades máximas posibles de comunicación así como la velocidad máxima estable alcanzada entre la computadora y el robot que permitieran un mínimo de 95 % de confiabilidad.

Control on off.bas

Esta aplicación es una modificación del programa anterior, en este caso las lecturas del tacómetro si son utilizadas para tratar de mantener estable la velocidad de los motores.
Al igual que en el caso anterior el usuario especifica la velocidad (en pulsos por segundo) que desea en los 4 motores, el sentido de giro y al oprimir el botón de inicio el sistema lee los tacómetros. Si la lectura del tacómetro es inferior a la velocidad deseada incrementa en uno la excitación al PWM de dicho motor, por el contrario si la velocidad del motor es superior a la velocidad deseada se decrementa en uno la excitación del PWM. Si la lectura del tacómetro y la velocidad especificada por el usuario son iguales no se ejecuta cambio alguno al PWM.
Esta aplicación se construyó como patrón de comparación para la siguiente aplicación. Fue útil este programa para determinar las capacidades de reacción del robot bajo la velocidad actual del sistema (5kbps).

Control P.bas

Inicialmente se pensó en implementar un control **PID** para comparar las virtudes de este esquema de control contra controles más primitivos tales como el programa anterior.
Este programa se basa en la ecuación

$$y[n] = y[n-1] + c_0 x[n] + c_1[n-1] + c_2[n-2]$$

en donde:

- $y[n]$ es la excitación en porcentaje que se enviará al PWM.

- $y[n-1]$ es la excitación en porcentaje que se envió al PWM anteriormente.
- $x[n]$ es el error entre la velocidad deseada y la velocidad actual.
- $x[n-1]$ es el error anterior.
- $x[n-2]$ es el error dos ciclos atrás.

Las constantes c_0, c_1 y c_2 dependen del sistema y se determinan por:

$$c_0 = K_p(1 + \frac{T_d}{h})$$

$$c_1 = K_p(\frac{h}{T_i} - 1 - \frac{2T_d}{h})$$

y

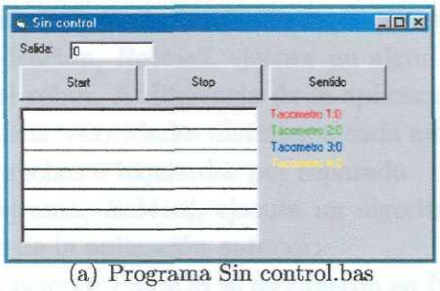
$$c_2 = \frac{K_p T_d}{h}$$

Como primer paso se implementó sólo el control **P** para lo cual se determinó la reacción del sistema ante una excitación tipo escalón en el PWM. Para determinar esta reacción se modificó el primer programa3.6.1 para que aparte de desplegar las lecturas de los tacómetros en pantalla también las mandara a un archivo en disco para su posterior análisis. Corriendo la aplicación para que imprimiera un impulso de 0 a 50 % se obtuvieron los siguientes datos:

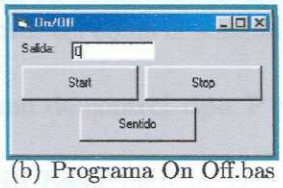
- Motor 1. $K_p = 0{,}8125$, $\theta = 147mSeg$ y $\tau = 204mSeg$.
- Motor 2. $K_p = 0{,}7812$, $\theta = 54mSeg$ y $\tau = 206mSeg$.
- Motor 3. $K_p = 0{,}7812$, $\theta = 120mSeg$ y $\tau = 199mSeg$.
- Motor 4. $K_p = 0{,}8125$, $\theta = 114mSeg$ y $\tau = 206mSeg$.

miniP.bas

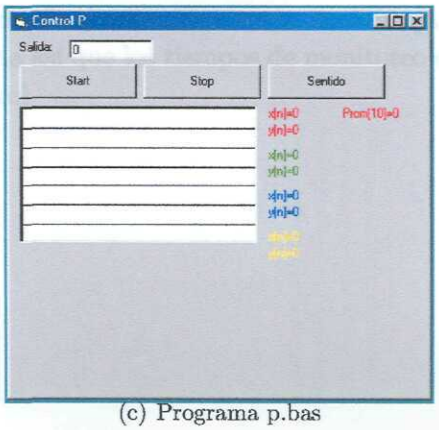
Este programa es igual al anterior, sólo que no despliega datos a pantalla para evitar tiempos perdidos en interfase de usuario y poder determinar el desempeño optimo del sistema. Se escribió tratando de ver que tanto influyen en el control los tiempos prolongados de monitoreo. Hay que recordar que aunque la velocidad del bus es de 6kpbs sólo se envían 20 bits cada 52 milisegundos que es el tiempo que tarda el sistema operativo en dibujar la interfase de usuario con las lecturas nuevas.



(a) Programa Sin control.bas



(b) Programa On Off.bas

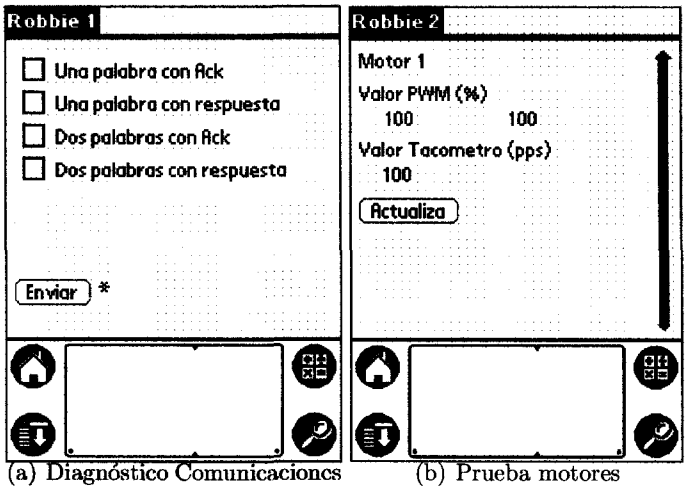


(c) Programa p.bas

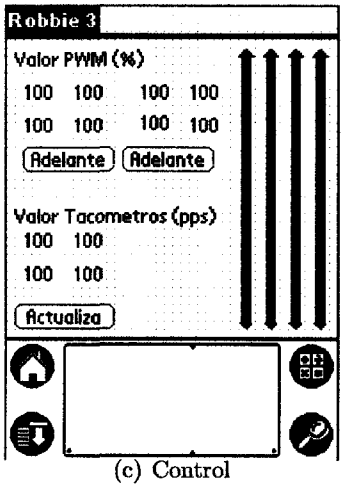
Figura 3.18: Aplicaciones Windows

3.6.2. Palm

Las mismas aplicaciones que se escribieron para el ambiente Windows se volvieron a escribir para el sistema operativos **PalmOS** que utilizan los dispositivos de mano tipo Palm.
La primera aplicación *Robbie1* sólo verifica el funcionamiento de la etapa de comunicación mandando cada una las 4 posibles combinaciones de comando existentes en el sistema3.4.2.
Una segunda aplicación ejecuta comandos de control sobre uno sólo de los motores. Al igual que en el caso de Windows esta aplicación solo sirvió para determinar velocidades de comunicación y fijar niveles de confiabilidad en la comunicación.
La tercera aplicación, *Robbie3*, ejecuta un algoritmo de control tipo on–off sobre los neumáticos del robot. A diferencia de la aplicación en Windows, en este programa es posible especificar velocidades distintas a cada motor así como el sentido de giro de los neumáticos derechos e izquierdos por separado.
El último programa, *Robbie4*, ejecuta un algoritmo de control tipo **P** y conserva las características de la aplicación anterior.
Todas las aplicaciones PalmOS se escribieron en **NS Basic** dada la facilidad que ofrece y el ambiente amigable de desarrollo. Una seria desventaja fue que la aplicación se ejecuta de forma muy lenta así que los tiempos de monitoreo crecen mucho y el sistema pierde capacidad de control.



(a) Diagnóstico Comunicaciones (b) Prueba motores



(c) Control

Figura 3.19: Aplicaciones PalmOS

Capítulo 4

Conclusiones

Al termino de este proyecto, tenemos construida una basa autónoma robotizada capas de controlar la velocidad y distancia recorrida por cada uno de sus cuatro neumáticos. Entre las principales dificultades encontradas se pueden enlistar las siguientes:

- Consumo eléctrico de los motores.
- Baja resolución de los tacómetros.
- Elevado costo de algunos componentes.
- Deformación del chasis del robot.

4.0.3. Consumo eléctrico de los motores.

Una de las funciones secundaria que se tenían planeadas para el robot, era que este tuviera la capacidad de subir y bajar escaleras. Aunque el chasis fue diseñado para poder sortear este tipo de obstáculos y las dimensiones del robot concuerdan con el tamaño típico de un escalón, finalmente no se pudo alcanzar esta función debido al alto consumo eléctrico de los motores.
Para que cada motor pueda desarrollar su máximo torque sería necesario suministrarle un ampere a cada motor. Ya que el sistema tiene cuatro motores se requeriría tener una batería de gran capacidad. Actualmente el robot cuenta con una batería de 12 volts y puede suministrar hasta 1.2 Amperes. Dicha batería pesa aproximadamente medio kilo, así para tener los 4 amperes se requeriría que el robot cargara con 2 kilos de baterías, lo cual significaría una merma al desempeño de los motores.

4.0.4. Baja resolución de los tacómetros.

El encoder construido en cada neumático provee de 23 pulsos por revolución y aunque inicialmente parecía suficiente, al terminar el sistema se observó que las capacidades de control del robot lucen más si este se desplaza a baja velocidad. La mínima

velocidad que puede alcanzar el robot en forma estable es de 37 pulsos por segundo que equivalen a poco más de una revolución del neumático cada segundo.
La mínima lectura posible del encoder es de 2.5 pulsos por segundo o un poquito menos de 1/10 de revolución por segundo. Estos 2.5 pulsos por segundo significan 5 flancos por segundo y como el sistema del encoder ejecuta 5 lecturas por segundo en cada lectura se lee 1 flanco.
Si este encoder tuviera más ranuras por revolución sería más fácil para el controlador monitorear la posición del neumático. Aunque el controlador maneja operaciones de punto flotante el microcontrolador sólo utiliza número enteros. Así que las perdidas por redondeo más las perdidas por resolución hacen difícil el control del robot a bajas velocidades.

4.0.5. Elevado costo de algunos componentes.

Resultó curioso que los componentes más costosos del diseño fueron los conectores utilizados para interfase de los sistemas mínimos y sus periféricos. Todos los sistemas mínimos se construyeron pensando en que fueran módulos genéricos e independientes, por lo cual todos ellos tienen suficientes conexiones para los 13 pines de entrada y salida con los cuales cuenta el microcontrolador.
En general del costo de cada sistema mínimo, un 50 % o más lo representan los conectores.
En este diseño se excluyeron los sensores de tipo magnético pues su costo es muy elevado y uno solo de estos componentes cuesta tanto como los 4 tacómetros implementados en el robot.

4.0.6. Deformación del chasis del robot.

Inicialmente se había pensado en la posibilidad de utilizar plásticos especiales para el diseño del chasis del robot, pero como este diseño esta pensado para ser de bajo costo se tomo la determinación de utilizar cualquier plástico que fuera fácil de conseguir en el mercado. El *Nylamin* resulto ser económico y se pudo conseguir en cantidades suficientes para este proyecto. Lamentablemente el robot ya terminado pesa alrededor de kilo y medio y esta sostenido sobre sus cuatro neumáticos, los cuales están conectados directamente al bástago del motor y el motor esta sujeto con cuatro tornillos a la base de la pata. Esta base fue maquinada en la fresadora para construir un sócalo en donde entra la base del reductor del motor. Aunque este sócalo permite la mejor sujeción del motor, es el rebaje que presenta el que deforma la parte inferior de la pata ante el peso.
En un principio se había pensado en reducir el peso del robot mediante la utilización de baterías menos pesadas, pero esto incrementaba grandemente el costo final del proyecto.

Apéndice A

Especificaciones del motor de *hobby* de Radio Shack

SR-65S Super High Speed Electric Motor
(273-0256) Specifications Faxback Doc. # 19832

Type: ..... Heavy Duty Series/Carbon Brush

Mounting Screws: ..... 2.5 mm

Outer Diameter of Motor: ..... 27.7 mm or 1.091 Inch

Shaft Diameter: ..... 2.3 mm or 0.0905 Inch

Motor Extension: ..... 10.0 mm or 0.394 Inch

Length of Shaft (case to end): ..... 15.7 mm

Shaft to Base: ..... 51.0 mm

Weight (approx.): ..... 49.0 gm

At 9 Volts:

Speed, No Load: ..... 11,500 RPM

Current Drain at No Load: ..... 0.28 - 0.350 mA

Speed at Maximum Efficiency: ..... 9,200 RPM

Current Drain at Max. Efficiency: ..... 1.86 A

Stall Torque: ..... 180 G-cm

At 12 Volts:

Speed, No Load: ..... 17,500 RPM

Current Drain at No Load: ..... 0.29 - 0.360 mA

Speed at Maximum Efficiency: ..... 13,200 RPM

Current Drain at Max. Efficiency: ..... 1.9 A

Stall Torque: ..... 240 G-cm

At 18 Volts:

Speed, No Load: ..... 24,000 RPM

Current Drain at No Load: ..... 0.32 - 0.40 A

Speed at Maximum Efficiency: ..... 18,000 RPM

Current Drain at No Load: ..... 1.98 A

Stall Torque: ..... 350 G-cm

Mounting holes are tapped for M2.6 screws on shaft end on 16 mm centers.

Specifications are typical; individual units might vary. Specifications are subject to change and improvement without notice.

Apéndice B

Rutinas de comunicación I2C

```
; 20 de Septiembre del 2003
;
; Rodolfo Reyes Aguillón
;
; Libreria: I2C
;
; CONJUNTO DE RUTINAS PARA COMUNICACION I2C
; EsperaInicioDeTransmision
; RecibeByte
; MandaByte
; MandaByteMaestro
; Ack
; AckMaestro
; EsperaFinDeTransmision
;
; Puerto A:
; Pin 0 - SDA Input
; Pin 1 - SCL Input
; Pin 2 - SDA Output
; Pin 3 - SCL Output
;

LIST p=16F84
#include p16f84.inc

BITES EQU 0x0C ; CUENTA BIT'S RECIBIDOS
BYTE EQU 0x0D ; REGISTRO DE RECEPCION
RESP EQU 0x0E ; REGISTRO DE TRANSMISION
RDELAY EQU 0x4F ; REGISTRO DE RETARDO
```

```
PROG CODE

Inicializa_puerto_maestro
GLOBAL Inicializa_puerto_maestro
BCF PORTA,2
BCF PORTA,3
RETURN

Inicia_transmision
GLOBAL  Inicia_transmision
BSF PORTA,2
CALL DELAY
BSF PORTA,3
CALL DELAY
RETURN

Finaliza_transmision
GLOBAL Finaliza_transmision
BCF PORTA,3
CALL DELAY
BCF PORTA,2
CALL DELAY
RETURN

EsperaInicioDeTransmision
GLOBAL EsperaInicioDeTransmision
PUERTOLISTO
BCF PORTA,2
BCF PORTA,3
INICIOESPERA
BTFSS PORTA,0 ; SDA EN 1?
GOTO EsperaInicioDeTransmision
BTFSS PORTA,1 ; SCL EN 1?
GOTO EsperaInicioDeTransmision
ESPERAINICIO
BTFSC PORTA,0 ; SDA EN 0...
GOTO ESPERAINICIO
BTFSS PORTA,1 ; Y SCL EN 1?
GOTO ESPERAINICIO
WSCLenL
```

```
BTFSC PORTA,1 ; SCL EN 0?
GOTO WSCLenL
BTFSC PORTA,0 ; Y SDA EN 0?
GOTO ESPERAINICIO
CALL PEME ; MARCO TIEMPO DE ESPERA
RETURN

EsperaFinDeTransmision
GLOBAL EsperaFinDeTransmision
CALL LISTO
WaFIN
BTFSS PORTA,1 ; ESPERA SCL EN 1
GOTO WaFIN
BTFSC PORTA,0 ; ESPERA SDA EN 0
GOTO WaFIN
WaFIN2
BTFSS PORTA,1 ; MIENTRAS SCL SEA 1
GOTO WaFIN
BTFSS PORTA,0 ; ESPERA SDA EN 1
GOTO WaFIN2
RETURN

Ack
GLOBAL  Ack
BSF PORTA,2 ; SDA EN 0
CALL LISTO
EspSCLenH2
BTFSS PORTA,1 ; SCL EN 1?
GOTO EspSCLenH2
EspSCLenL2
BTFSC PORTA,1 ; SCL EN 0?
GOTO EspSCLenL2
CALL PEME
BCF PORTA,2 ; SDA EN 1
RETURN

AckMaestro ; EMPIEZO CON SDA Y SCL EN BAJO
GLOBAL  AckMaestro
BCF PORTA,2 ; SDA EN 1
CALL DELAY
```

```
PonSCLenHM
BCF PORTA,3 ; SCL EN 1
BTFSS PORTA,1 ; SCL EN 1?
GOTO PonSCLenHM
CALL DELAY
BCF STATUS,C
BTFSC PORTA,0 ; ACK
BSF STATUS,C
CALL DELAY
BSF PORTA,3 ; SCL EN 0
CALL DELAY
BSF PORTA,2 ; SDA EN 1
CALL DELAY
RETURN

RecibeByte ; INICIO CON SDA Y SCL EN 0 Y ESPERA
GLOBAL RecibeByte ; ACTIVADA
CLRF BITES
CLRF BYTE ; INICIALIZO REGISTROS
VERIFICAFIN
BTFSC BITES,3 ; LLEGUÉ A 8?
GOTO FINRB ; SIP -> SALGO
CALL LISTO ; SCL LISTO PARA RX
WSCLenH
BTFSS PORTA,1 ; SCL EN 1?
GOTO WSCLenH
BCF STATUS,C ; BORRO EL CARRY
RRF BYTE,1 ; ROTO EL REGISTRO
BTFSC PORTA,0
BSF BYTE,7 ; RECIBÍ UN 1
ESPERASCL0
BTFSC PORTA,1 ; ESPERA SCL EN 0
GOTO ESPERASCL0
CALL PEME ; PERAME TANTITO
INCF BITES,1 ; INCREMENTO CUENTA DE BIT'S
GOTO VERIFICAFIN
FINRB
MOVF BYTE,0
RETURN ; TERMINO CON ESPERA ACTIVADA
```

```
RecibeByteMaestro ; INICIO CON SDA Y SCL EN 0
GLOBAL RecibeByteMaestro ; ACTIVADA
CLRF BITES
CLRF BYTE ; INICIALIZO REGISTROS
BCF PORTA,2 ; SDA EN 1
VERIFICAFINm
BTFSC BITES,3 ; LLEGUÉ A 8?
GOTO FINRBm ; SIP -> SALGO
BCF PORTA,3 ; SCL EN 1
CALL DELAY
WSCLenHm
BTFSS PORTA,1 ; SCL EN 1?
GOTO WSCLenHm
CALL DELAY
BCF STATUS,C ; BORRO EL CARRY
RRF BYTE,1 ; ROTO EL REGISTRO
BTFSC PORTA,0
BSF BYTE,7 ; RECIBÍ UN 1
CALL DELAY
BSF PORTA,3 ; SCL EN 0
CALL DELAY
ESPERASCLOm
BTFSC PORTA,1 ; ESPERA SCL EN 0
GOTO ESPERASCLOm
BSF PORTA,2 ; SDA EN 0
INCF BITES,1 ; INCREMENTO CUENTA DE BIT'S
GOTO VERIFICAFINm
FINRBm
MOVF BYTE,0
RETURN ; TERMINO CON ESPERA ACTIVADA

MandaByte ; INICIO CON SDA Y SCL EN 0 Y ESPERA
GLOBAL  MandaByte ; ACTIVADA
MOVWF RESP ; CARGO EL DATO A MANDAR
CLRF BITES
VERIFICAFINM
BTFSC BITES,3 ; LLEGUÉ A 8?
GOTO FINMB ; SIP -> SALGO
PONBIT
BTFSS RESP,0 ; BIT 1?
```

```
BSF PORTA,2 ; SDA EN BAJO
BTFSC RESP,0 ; BIT 0?
BCF PORTA,2 ; SDA EN ALTO
CALL LISTO ; SCL LISTO PARA RX
WSCLenHM
BTFSS PORTA,1 ; SCL EN 1?
GOTO WSCLenHM
RRF RESP,1 ; ROTO EL REGISTRO
WSCLenLM
BTFSC PORTA,1 ; ESPERA SCL EN 0
GOTO WSCLenLM
CALL PEME ; PERAME TANTITO
INCF BITES,1 ; INCREMENTO CUENTA DE BIT'S
GOTO VERIFICAFINM
FINMB
BCF PORTA,2
RETURN ; TERMINO CON ESPERA ACTIVADA

MandaByteMaestro ; INICIO CON SDA Y SCL EN BAJO
GLOBAL  MandaByteMaestro; ACTIVADA
MOVWF RESP ; CARGO EL DATO A MANDAR
CLRF BITES
CALL DELAY
VERIFICAFINMM
BTFSC BITES,3 ; LLEGUÉ A 8?
GOTO FINMBM ; SIP -> SALGO
PONBITM
BTFSS RESP,0 ; BIT 1?
BSF PORTA,2 ; SDA EN BAJO
BTFSC RESP,0 ; BIT 0?
BCF PORTA,2 ; SDA EN ALTO
CALL DELAY
BCF PORTA,3 ; SCL EN ALTO
CALL DELAY
WSCLenHMM
BTFSS PORTA,1 ; SCL EN 1?
GOTO WSCLenHMM
RRF RESP,1 ; ROTO EL REGISTRO
CALL DELAY
BSF PORTA,3 ; SCL EN BAJO
```

```
WSCLenLMM
BTFSC PORTA,1 ; ESPERA SCL EN 0
GOTO WSCLenLMM
CALL DELAY
INCF BITES,1 ; INCREMENTO CUENTA DE BIT'S
GOTO VERIFICAFINMM
FINMBM
BSF PORTA,2 ; DATO A CERO
RETURN ; TERMINO CON ESPERA ACTIVADA

LISTO
BCF PORTA,3 ; SCL EN 1
RETURN

PEME
BSF PORTA,3 ; SCL EN 0
RETURN

DELAY ; 1/4 DE 52uSEG
MOVLW 0x03
MOVWF RDELAY
CICLO_DELAY
DECFSZ RDELAY
GOTO CICLO_DELAY
RETURN

END
```

Apéndice C

Rutinas de tacómetro

```
; 18 de Septiembre del 2003
;
; Rodolfo Reyes Aguillón
;
; Programa: Tacometro
;
; Programa que lee las revoluciones del neumático en un rango de 0 a 7 RPS.
; Protocolo de comunicación I2C

LIST p=16F84
#include p16f84.inc

EXTERN EsperaInicioDeTransmision, RecibeByte, Ack
EXTERN MandaByte, EsperaFinDeTransmision

PORCENT EQU 0x0F ; CUENTA DE PORCENTAJE
QUIEN EQU 0x10 ; DESTINATARIO DEL DATO
RPM1 EQU 0x11 ; REGISTRO DE REV.POR SEG.
CRPM1 EQU 0x12 ; CONTADOR DE REVOLUCIONES
RPM2 EQU 0x13
CRPM2 EQU 0x14
RPM3 EQU 0x15
CRPM3 EQU 0x16
RPM4 EQU 0x17 ; RPM MOTOR X
CRPM4 EQU 0x18
RESPW EQU 0x19 ; TEST DE RESPALDO DE W
MEMORY EQU 0x1A
RESPST EQU 0x1B

STARTUP CODE
```

```
GOTO SETUP
NOP
NOP
NOP
GOTO TEMPORIZADOR

PROG CODE
SETUP
CLRF PORTB
CALL INITREG ; INICIALIZO REGISTROS
BSF STATUS,RP0
MOVLW 0xFF
MOVWF TRISB
MOVLW 0x03
MOVWF TRISA
MOVLW 0xA0
MOVWF INTCON
MOVLW 0x01
MOVWF OPTION_REG
BCF STATUS,RP0

CLRF MEMORY

MAIN
CALL EsperaInicioDeTransmision
CALL RecibeByte ; INICIO RECEPCIÓN DE UN BYTE
MOVWF QUIEN ; RESPALDO "BYTE" EN "QUIEN"
PROCESACOMANDO
MOVLW 0x01
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO LEERMP2
CALL Ack
MOVF RPM1,0
CALL MandaByte
CALL Ack
GOTO FIN
LEERMP2
MOVLW 0x02
XORWF QUIEN,0
```

```
BTFSS STATUS,Z
GOTO LEERMP3
CALL Ack
MOVF RPM2,0
CALL MandaByte
CALL Ack
GOTO FIN
LEERMP3
MOVLW 0x03
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO LEERPM4
CALL Ack
MOVF RPM3,0
CALL MandaByte
CALL Ack
GOTO FIN
LEERPM4
MOVLW 0x04
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO ACTIVO
CALL Ack
MOVF RPM4,0
CALL MandaByte
CALL Ack
GOTO FIN
ACTIVO
MOVLW 0x0F ; ACTIVO?
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO $+3
CALL Ack
GOTO FIN
MANDAM1
MOVLW 0x03
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO $+5
CALL Ack
```

```
MOVLW 0xAA
CALL MandaByte
CALL Ack
FIN
CALL EsperaFinDeTransmision
GOTO MAIN ; VUELVO A EMPEZAR

TEMPORIZADOR
MOVWF RESPW
MOVF STATUS,0
MOVWF RESPST
MOVLW 0x05 ; RECARGO EL TIMER
MOVWF TMR0
BCF INTCON,2 ; BORRO BANDERA
BTFSC INTCON,2
GOTO $-2 ; SE BORRO?
MOVLW 0xC8 ; 200 -> W
SUBWF PORCENT,0 ; PORCENT - 200 -> W
BTFSC STATUS,C
CALL UPDATEREG
INCF PORCENT,1 ; W NEGATIVO
LECTURA1
BTFSS PORTB,6
GOTO ESCERO
ESUNO ; EL PUERTO TIENE UN 1
BTFSC MEMORY,0 ; SI EL ULTIMO FUE 1
GOTO LECTURA2 ; SALGO
INCF CRPM1,1 ; SINO INCREMENTO CONTADOR
BSF MEMORY,0 ; GUARDO EN MEMORIA UN 1
GOTO LECTURA2
ESCERO
BTFSS MEMORY,0
GOTO LECTURA2
INCF CRPM1,1
BCF MEMORY,0
LECTURA2
BTFSS PORTB,4
GOTO ESCERO2
ESUNO2 ; EL PUERTO TIENE UN 1
BTFSC MEMORY,1 ; SI EL ULTIMO FUE 1
```

```
GOTO LECTURA3 ; SALGO
INCF CRPM2,1 ; SINO INCREMENTO CONTADOR
BSF MEMORY,1 ; GUARDO EN MEMORIA UN 1
GOTO LECTURA3
ESCERO2
BTFSS MEMORY,1
GOTO LECTURA3
INCF CRPM2,1
BCF MEMORY,1
LECTURA3
BTFSS PORTB,2
GOTO ESCERO3
ESUNO3 ; EL PUERTO TIENE UN 1
BTFSC MEMORY,2 ; SI EL ULTIMO FUE 1
GOTO LECTURA4 ; SALGO
INCF CRPM3,1 ; SINO INCREMENTO CONTADOR
BSF MEMORY,2 ; GUARDO EN MEMORIA UN 1
GOTO LECTURA4
ESCERO3
BTFSS MEMORY,2
GOTO LECTURA4
INCF CRPM3,1
BCF MEMORY,2
LECTURA4
BTFSS PORTB,0
GOTO ESCERO4
ESUNO4 ; EL PUERTO TIENE UN 1
BTFSC MEMORY,3 ; SI EL ULTIMO FUE 1
GOTO FINAL ; SALGO
INCF CRPM4,1 ; SINO INCREMENTO CONTADOR
BSF MEMORY,3 ; GUARDO EN MEMORIA UN 1
GOTO FINAL
ESCERO4
BTFSS MEMORY,3
GOTO FINAL
INCF CRPM4,1
BCF MEMORY,3
FINAL
MOVF RESPST,0
MOVWF STATUS
```

```
MOVF RESPW,0
RETFIE

INITREG
CLRF PORCENT
CLRF RPM1
CLRF CRPM1
CLRF RPM2
CLRF CRPM2
CLRF RPM3
CLRF CRPM3
CLRF RPM4
CLRF CRPM4
RETURN


UPDATEREG
CLRF PORCENT
MOVF CRPM1,0 ; CRPM1 -> W
MOVWF RPM1 ; W      -> RPM1
CLRF CRPM1
MOVF CRPM2,0
MOVWF RPM2
CLRF CRPM2
MOVF CRPM3,0
MOVWF RPM3
CLRF CRPM3
MOVF CRPM4,0
MOVWF RPM4
CLRF CRPM4
RETURN

END
```

Apéndice D

Rutinas de PWM

```
; 22 de Agosto del 2003
;
; Rodolfo Reyes Aguillón
;
; Programa: Robot1
;
; Programa que genera un tren de pulsos de frecuencia 100Hz
; modulable en PWM.
; resolución de control = 1%
; Protocolo de comunicación I2C

LIST p=16F84
#include p16f84.inc

EXTERN EsperaInicioDeTransmision, RecibeByte, Ack
EXTERN MandaByte, EsperaFinDeTransmision

CTRLMT EQU 0x0F ; CONTROL DE MOTORES
; NIBBLE BAJO = SENTIDO
PORCENT EQU 0x10 ; CUENTA DE PORCENTAJE
QUIEN EQU 0x11 ; DESTINATARIO DEL DATO
RM1 EQU 0x12
RM2 EQU 0x13
RM3 EQU 0x14
RM4 EQU 0x15 ; DUTY CYCLE MOTOR X
RESPW EQU 0x16 ; RESPALDO DE W
RESPST EQU 0x17 ; RESPALDO DE STATUS


STARTUP CODE
```

```
GOTO SETUP
NOP
NOP
NOP
GOTO TEMPORIZADOR

PROG CODE
SETUP
MOVLW 0xCE ; CARGO EL TIMER
MOVWF TMR0
CALL INITREG ; INICIALIZO REGISTROS
BSF STATUS,RP0
CLRF TRISB
MOVLW 0x03
MOVWF TRISA
CLRF OPTION_REG
MOVLW 0xA0
MOVWF INTCON
BCF STATUS,RP0

CLRF PORTB
MOVLW 0x65 ; MONITOR DE 100%
MOVWF PORCENT
MOVLW 0x0C
MOVWF CTRLMT ; CONTROL DE MOTORES

MAIN
CALL EsperaInicioDeTransmision
CALL RecibeByte ; INICIO RECEPCIÓN DE UN BYTE
MOVWF QUIEN ; RESPALDO "BYTE" EN "QUIEN"
COMANDORM1
MOVLW 0xF1
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO VERIFICARM1
CALL  Ack
CALL RecibeByte ; INICIO RECEPCIÓN DE UN BYTE
MOVWF RM1
CALL Ack
GOTO FIN
```

```
VERIFICARM1
MOVLW 0xF5
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO COMANDORM2
CALL Ack
MOVF RM1,0
CALL MandaByte
CALL Ack
GOTO FIN
COMANDORM2
MOVLW 0xF2
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO VERIFICARM2
CALL  Ack
CALL RecibeByte ; INICIO RECEPCIÓN DE UN BYTE
MOVWF RM2
CALL Ack
GOTO FIN
VERIFICARM2
MOVLW 0xF6
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO COMANDORM3
CALL Ack
MOVF RM2,0
CALL MandaByte
CALL Ack
GOTO FIN
COMANDORM3
MOVLW 0xF3
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO VERIFICARM3
CALL  Ack
CALL RecibeByte ; INICIO RECEPCIÓN DE UN BYTE
MOVWF RM3
CALL Ack
GOTO FIN
```

```
VERIFICARM3
MOVLW 0xF7
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO COMANDORM4
CALL Ack
MOVF RM3,0
CALL MandaByte
CALL Ack
GOTO FIN
COMANDORM4
MOVLW 0xF4
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO VERIFICARM4
CALL  Ack
CALL RecibeByte ; INICIO RECEPCIÓN DE UN BYTE
MOVWF RM4
CALL Ack
GOTO FIN
VERIFICARM4
MOVLW 0xF8
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO CHNGSENTIDO
CALL Ack
MOVF RM4,0
CALL MandaByte
CALL Ack
GOTO FIN
CHNGSENTIDO
MOVLW 0xF9
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO ACTIVO
CALL Ack
COMF CTRLMT
GOTO FIN
ACTIVO
MOVLW 0xFF ; ACTIVO?
```

```
XORWF QUIEN,0
BTFSS STATUS,Z
GOTO FIN
CALL Ack
GOTO FIN
FIN
CALL EsperaFinDeTransmision
GOTO MAIN ; VUELVO A EMPEZAR

TEMPORIZADOR
MOVWF RESPW
MOVF STATUS,0
MOVWF RESPST
MOVLW 0xCE ; RECARGO EL TIMER
MOVWF TMR0
BCF INTCON,2 ; BORRO BANDERA
BTFSC INTCON,2 ; SE BORRO BIEN?
GOTO $-2
MOVLW 0x64 ; 100 -> W
SUBWF PORCENT,0 ; PORCENT - 100 -> W
BTFSC STATUS,C
CLRF PORCENT ; W POSITIVO O CERO
INCF PORCENT,1 ; W NEGATIVO
M1
MOVF PORCENT,0 ; PORTCENT -> W
SUBWF RM1,0 ; RM1 - PORCENT -> W
BTFSS STATUS,C
GOTO $+6 ; W NEGATIVO
BTFSC CTRLMT,0
BSF PORTB,6
BTFSS CTRLMT,0
BSF PORTB,7
GOTO $+3
M1OFF
BCF PORTB,6 ; M1 APAGADO
BCF PORTB,7
M2
MOVF PORCENT,0
SUBWF RM2,0
BTFSS STATUS,C
```

```
GOTO $+6
BTFSC CTRLMT,1
BSF PORTB,4
BTFSS CTRLMT,1
BSF PORTB,5
GOTO $+3
M2OFF
BCF PORTB,4
BCF PORTB,5
M3
MOVF PORCENT,0
SUBWF RM3,0
BTFSS STATUS,C
GOTO $+6
BTFSC CTRLMT,2
BSF PORTB,2
BTFSS CTRLMT,2
BSF PORTB,3
GOTO $+3
M3OFF
BCF PORTB,3
BCF PORTB,2
M4
MOVF PORCENT,0
SUBWF RM4,0
BTFSS STATUS,C
GOTO $+6
BTFSC CTRLMT,3
BSF PORTB,0
BTFSS CTRLMT,3
BSF PORTB,1
GOTO $+3
M4OFF
BCF PORTB,1
BCF PORTB,0
FININTERR
MOVF RESPST,0
MOVWF STATUS
MOVF RESPW,0
RETFIE
```

```
INITREG
CLRF PORCENT
CLRF RM1
CLRF RM2
CLRF RM3
CLRF RM4
RETURN

END
```

Apéndice E

Transmisión Radio

```
; 25 de Octubre del 2003
;
; Rodolfo Reyes Aguillón
;
; Programa: Tx
;
; Programa que genera la modulación para transmisión de datos
;    RS-232 a Código Manchester a TLP-434 Transmisor
;

LIST p=16F84
#include p16f84.inc


CUENTA EQU 0x12

SETUP
BSF STATUS,RP0
MOVLW 0x01
MOVWF TRISB
CLRF TRISA
CLRF INTCON
CLRF OPTION_REG
BCF STATUS,RP0

MAIN
CLRF  TMR0
CICLO_PRINCIPAL
BTFSC PORTB,0
BCF PORTB,1
```

```
BTFSC PORTB,0
GOTO CICLO_PRINCIPAL
MOVLW 0x68 ; 833/8=104 DECIMAL= 104uSEG.
SUBWF TMR0,0
BTFSS STATUS,C
GOTO CICLO_PRINCIPAL
INTERRUPCION
INCF CUENTA,1
BTFSC CUENTA,1
GOTO PON_BIT
PON_MARCA
BSF PORTB,1
GOTO MAIN
PON_BIT
BCF PORTB,1
GOTO MAIN

END
```

Apéndice F

Recepción Radio

```
; 25 de Octubre del 2003
;
; Rodolfo Reyes Aguillón
;
; Programa: Rd
;
; Programa que genera la demodulación para recepción de datos
;   RLP-434 Receptor Código Manchester a RS-232
;

LIST p=16F84
#include p16f84.inc


RDELAY EQU 0x12
CUNO EQU 0x13
CCERO EQU 0x14

SETUP
BSF STATUS,RP0
MOVLW 0x01
MOVWF TRISB
CLRF TRISA
CLRF INTCON
MOVLW 0x00
MOVWF OPTION_REG
BCF STATUS,RP0

ESPERA_PULSO_ALTO
CLRF CCERO
```

```
CLRF CUNO
BTFSS PORTB,0
GOTO ESPERA_PULSO_ALTO
CICLO_PRINCIPAL
MOVLW 0x34 ; 104/2=52 DECIMAL= 104uSEG.
SUBWF TMR0,0
BTFSS STATUS,C
GOTO CICLO_PRINCIPAL
CLRF  TMR0
VERIFICA_ENTRADA_DE_RADIO
BTFSS PORTB,0
GOTO AHORA_ESTOY_EN_CERO
INCF CUNO
MOVLW 0x05 ; UN UNO NO TIENE MAS DE 5 MUESTREOS
SUBWF CUNO,0
BTFSC STATUS,C
GOTO ESPERA_PULSO_BAJO
GOTO  CICLO_PRINCIPAL
AHORA_ESTOY_EN_CERO
MOVLW 0x02 ; UN UNO TIENE MINIMO 3 MUESTREOS
SUBWF CUNO,0
BTFSS STATUS,C
GOTO ESPERA_PULSO_ALTO
CICLO_DE_CEROS
INCF CCERO
MOVLW 0x03 ; UN CERO TIENE MINIMO 4 MUESTREOS
SUBWF CCERO,0
BTFSC STATUS,C
GOTO CONFIRMA_UNO
CICLO_PRINCIPAL_CEROS
MOVLW 0x34 ; 104/2=52 DECIMAL= 104uSEG.
SUBWF TMR0,0
BTFSS STATUS,C
GOTO CICLO_PRINCIPAL_CEROS
CLRF  TMR0
BTFSC PORTB,0
GOTO ESPERA_PULSO_BAJO
GOTO CICLO_DE_CEROS
CONFIRMA_UNO
BSF PORTB,1
```

```
BCF PORTB,1
GOTO ESPERA_PULSO_ALTO

ESPERA_PULSO_BAJO
BTFSC PORTB,0
GOTO ESPERA_PULSO_BAJO
GOTO ESPERA_PULSO_ALTO

END
```

Apéndice G

Rutinas de comunicación I2C en Visual Basic

```
'*********************************************************
'**                                                     **
'** Archivo: CI2CDLL.CLS                                **
'**                                                     **
'** DLL en Visual Basic con rutinas de comunicación I2C **
'**  Por: Rodolfo Reyes Aguillón                        **
'**                                                     **
'** 17 de Septiembre del 2003                           **
'**                                                     **
'** Requiere: IO.DLL, KERNEL32.DLL                      **
'**                                                     **
'*********************************************************
'
' Rutinas:
'   Init(Puerto)
'   PuertoActivo
'   TxByte(Dato)
'   TxWord(Dirección, Dato)
'   Dato= RxByte(Dirección)
'
'*********************************************************
'
Private Declare Sub SetPortBit Lib "IO.DLL"
(ByVal Port As Integer, ByVal Bit As Byte)
Private Declare Sub ClrPortBit Lib "IO.DLL"
(ByVal Port As Integer, ByVal Bit As Byte)
Private Declare Function GetPortBit Lib "IO.DLL"
(ByVal Port As Integer, ByVal Bit As Byte) As Boolean
Private Declare Function QueryPerformanceFrequency Lib "kernel32.dll"
(lpFrequency As Currency) As Long
```

```
Private Declare Function QueryPerformanceCounter Lib "kernel32.dll"
(lpPerformanceCount As Currency) As Long
Option Explicit
Dim Port As Integer

Public Sub Init(Puerto As Integer)
    Port = Puerto
    Call ClrPortBit(Port, 0)  'SDA=1
    Call ClrPortBit(Port, 1)  'SCL=1
    Call uDelay
    Call WaitScl
End Sub

Public Sub PuertoActivo()
    Call ClrPortBit(Port, 0)  'SDA=1
    Call ClrPortBit(Port, 1)  'SCL=1
    Call uDelay
    Call WaitScl
End Sub

Private Sub InicioDeTransmision()
    Call SetPortBit(Port, 0)  'SDA=0
    Call uDelay
    Call SetPortBit(Port, 1)  'SCL=0
    Call uDelay
End Sub

Private Sub FinDeTransmision()
    Call ClrPortBit(Port, 1)  'SCL=1
    Call uDelay
    Call WaitScl
    Call ClrPortBit(Port, 0)  'SDA=1
    Call uDelay
End Sub

Private Sub Tx0()
    Call ClrPortBit(Port, 1)  'SCL=1
    Call uDelay
    Call WaitScl
    Call SetPortBit(Port, 1)  'SCL=0
```

```
    Call uDelay
End Sub

Private Sub Tx1()
    Call ClrPortBit(Port, 0)  'SDA=1
    Call uDelay             'Sólo para debug
    Call Tx0
    Call SetPortBit(Port, 0)  'SDA=0
    Call uDelay
End Sub

Private Function TxByteR(Valor As Integer) As Boolean
    Dim Estado As Boolean
    Dim i As Integer
    Dim Bite As Boolean

    Call InicioDeTransmision
    For i = 0 To 7
        Bite = Valor Mod 2
        If (Bite = 0) Then
            Call Tx0
        Else
            Call Tx1
        End If
        Valor = Valor \ 2
        Next i
    Call ClrPortBit(Port, 0)
    Call uDelay
    Call ClrPortBit(Port, 1)      '1er Ack
    Call uDelay
    Call WaitScl
    Estado = GetPortBit(Port + 1, 6)
    Call SetPortBit(Port, 1)
    Call uDelay
    Call SetPortBit(Port, 0)
    Call uDelay
    Call FinDeTransmision

    TxByteR = Not (Estado)
End Function
```

```
Public Function TxByte(Valor As Integer)
    Dim Intentos As Byte
    Dim Estado As Boolean

    Intentos = 0
    Do
        Estado = TxByteR(Valor)
        Intentos = Intentos + 1
        Loop While ((Estado = False) And (Intentos < 5))

    TxByte = Estado
End Function

Private Function TxWordR(Direccion As Integer, Valor As Integer) As Boolean
    Dim Estado As Boolean
    Dim i As Integer
    Dim Bite As Boolean

    Call InicioDeTransmision
    For i = 0 To 7
        Bite = Direccion Mod 2
        If (Bite = 0) Then
            Call Tx0
        Else
            Call Tx1
        End If
        Direccion = Direccion \ 2
        Next i
    Call ClrPortBit(Port, 0)
    Call uDelay
    Call ClrPortBit(Port, 1)      '1er Ack
    Call uDelay
    Call WaitScl
    Estado = GetPortBit(Port + 1, 6)
    Call SetPortBit(Port, 1)
    Call uDelay
    Call SetPortBit(Port, 0)
    Call uDelay
    If (Estado = False) Then
```

```
        For i = 0 To 7
            Bite = Valor Mod 2
            If (Bite = 0) Then
                Call Tx0
            Else
                Call Tx1
            End If
            Valor = Valor \ 2
            Next i
        Call ClrPortBit(Port, 0)
        Call uDelay
        Call ClrPortBit(Port, 1)  '2o Ack
        Call uDelay
        Call WaitScl
        Estado = GetPortBit(Port + 1, 6)
        Call SetPortBit(Port, 1)
        Call uDelay
        Call SetPortBit(Port, 0)
        Call uDelay
    End If
    Call FinDeTransmision
    TxWordR = Not (Estado)
End Function

Public Sub TxWord(ByVal Direccion As Integer, ByVal Valor As Integer)
    Dim Intentos As Byte
    Dim Estado As Boolean

    Intentos = 0
    Do
        Estado = TxWordR(Direccion, Valor)
        Intentos = Intentos + 1
        Loop While ((Estado = False) And (Intentos < 11))
End Sub

Public Function RxByte(Direccion As Integer) As Byte
    Dim Valor As Byte
    Dim Estado As Boolean
    Dim i As Integer
    Dim Bite As Boolean
```

```
Valor = 0
Call InicioDeTransmision
For i = 0 To 7
    Bite = Direccion Mod 2
    If (Bite = 0) Then
        Call Tx0
    Else
        Call Tx1
    End If
    Direccion = Direccion \ 2
    Next i
Call ClrPortBit(Port, 0)
Call uDelay
Call ClrPortBit(Port, 1)      '1er Ack
Call uDelay
Call WaitScl
Estado = GetPortBit(Port + 1, 6)
Call SetPortBit(Port, 1)
Call uDelay
Call SetPortBit(Port, 0)
Call uDelay
If (Estado = False) Then
    Call ClrPortBit(Port, 0)
    For i = 0 To 7
        Call ClrPortBit(Port, 1)    'Pongo SCL en 1
        Call uDelay
        Call WaitScl                  'Espera SCL en 1
        If GetPortBit(Port + 1, 6) Then 'Verifica estado de SDA
            Valor = Valor + (2 ^ i)
        End If
        Call SetPortBit(Port, 1)    'Pongo SCL en 0
        Call uDelay                   'Delay
        Next i
    Call ClrPortBit(Port, 1)  '2o Ack
    Call uDelay
    Call WaitScl
    Estado = GetPortBit(Port + 1, 6)
    Call SetPortBit(Port, 1)
    Call uDelay
```

```
        Call SetPortBit(Port, 0)
        Call uDelay
        End If
    Call FinDeTransmision
    RxByte = Valor
End Function

Private Sub uDelay()
    Dim Clock1 As Currency, Clock2 As Currency

    QueryPerformanceCounter Clock1
    Do
        QueryPerformanceCounter Clock2
        Loop While (Clock1 + 0.007) > Clock2
End Sub

Private Sub WaitScl()
    Dim Estado As Boolean

    Do
        Estado = GetPortBit(&H379, 7)
        Loop While (Estado = True)
End Sub
```

Apéndice H

Rutinas de comunicación Palm-Robot

```
; 6 de Noviembre del 2003
;
; Rodolfo Reyes Aguillón
;
; Programa: Palm
;
; Programa IRDA-SIR
;

LIST p=16F84
#include p16f84.inc

EXTERN  Inicializa_puerto_maestro
EXTERN  Inicia_transmision, Finaliza_transmision
EXTERN MandaByteMaestro, AckMaestro
EXTERN RecibeByteMaestro

CBITS EQU 0x10
RDELAY EQU 0x11
DQUIEN EQU 0x12
DQUE EQU 0x13
RRESP EQU 0x14

STARTUP CODE
GOTO SETUP
NOP
NOP
NOP
NOP
```

```
PROG CODE
SETUP
BSF STATUS,RP0
MOVLW 0x01
MOVWF TRISB
MOVLW 0x03
MOVWF TRISA
BCF STATUS,RP0

CALL Inicializa_puerto_maestro

MAIN
CLRF CBITS
BUSCA_INICIO
BTFSS PORTB,0
GOTO MAIN
INCF CBITS
BTFSS CBITS,6
GOTO BUSCA_INICIO
ESPERA_INICIO_DE_TX
CLRF CBITS
ESPERA_PULSO
BTFSC PORTB,0
GOTO ESPERA_PULSO
NOP ;DELAY DE 5uSEG
NOP
NOP
BTFSC PORTB,0
GOTO ESPERA_PULSO_ALTO
CALL DELAY0
BTFSS PORTB,0
GOTO MAIN
BSF PORTB,1
BCF PORTB,1
CALL DELAY1 ; INICIO CONFIRMADO
BIT ; ESPERO UN BAUD
INCF CBITS,1
BCF STATUS,C
RRF DQUIEN,1
BTFSS PORTB,0
```

```
BCF DQUIEN,7
BTFSC PORTB,0
BSF DQUIEN,7
BSF PORTB,1
BCF PORTB,1
CALL DELAY2
BTFSS CBITS,3
GOTO BIT
BTFSS PORTB,0 ; FIN DE TX
GOTO MAIN
BTFSS DQUIEN,7 ; 2 O 1 BYTES DE RECEPCION
GOTO UN_BYTE_SOLAMENTE
ESPERA_INICIO_QUE
CLRF CBITS
ESPERA_PULSO_QUE
BTFSC PORTB,0
GOTO ESPERA_PULSO_QUE
NOP ;DELAY DE 5uSEG
NOP
NOP
BTFSC PORTB,0
GOTO ESPERA_PULSO_ALTO
CALL DELAY0
BTFSS PORTB,0
GOTO MAIN
BSF PORTB,1
BCF PORTB,1
CALL DELAY1 ; INICIO CONFIRMADO
BIT_QUE ; ESPERO UN BAUD
INCF CBITS,1
BCF STATUS,C
RRF DQUE,1
BTFSS PORTB,0
BCF DQUE,7
BTFSC PORTB,0
BSF DQUE,7
BSF PORTB,1
BCF PORTB,1
CALL DELAY2
BTFSS CBITS,3
```

```
GOTO BIT_QUE
BTFSS PORTB,0 ; FIN DE TX
GOTO MAIN
;**************************************
UN_BYTE_SOLAMENTE
CALL Inicia_transmision
MOVF DQUIEN,0
CALL MandaByteMaestro
CALL AckMaestro
BTFSC STATUS,C ; ACK?
GOTO NOACK
CALL ACK
BTFSS DQUIEN,7 ; UN SOLO BYTE?
GOTO CON_RESPUESTA
MOVF DQUE,0
CALL MandaByteMaestro
CALL AckMaestro
BTFSC STATUS,C ; ACK?
GOTO NOACK
CALL ACK
CON_RESPUESTA
BTFSS DQUIEN,6
GOTO FIN_FRAME
CALL RecibeByteMaestro
CALL IRDA_SEND
CALL AckMaestro
FIN_FRAME
CALL Finaliza_transmision
GOTO MAIN
ESPERA_PULSO_ALTO
BTFSS PORTB,0
GOTO ESPERA_PULSO_ALTO
GOTO ESPERA_INICIO_DE_TX
NOACK
MOVLW 0x78
CALL IRDA_SEND
CALL Finaliza_transmision
GOTO MAIN

ACK
```

```
MOVLW 0x6F
CALL IRDA_SEND
RETURN

DELAY0
MOVLW 0x07
MOVWF RDELAY
CICLO0
DECFSZ RDELAY,1
GOTO CICLO0
RETURN

IRDA_SEND
MOVWF RRESP
CLRF CBITS
BIT_DE_INICIO
CALL PULSO
NOP
NOP
NOP
NOP
NOP
MANDO_BIT_IRDA
INCF CBITS,1
BTFSS RRESP,0
CALL PULSO
BTFSC RRESP,0
CALL ESPACIO
RRF RRESP,1
BTFSS CBITS,3
GOTO MANDO_BIT_IRDA
BIT_DE_PARO
CALL ESPACIO
RETURN

ESPACIO
MOVLW 0x0E
MOVWF RDELAY
CICLOESPACIO
DECFSZ RDELAY,1
```

```
GOTO CICLOESPACIO
RETURN

PULSO
BSF PORTB,2
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
BCF PORTB,2
PAUSA42uSEG
MOVLW 0x0A
MOVWF RDELAY
CICLOPAUSA
DECFSZ RDELAY,1
GOTO CICLOPAUSA
RETURN

DELAY1
MOVLW 0x05
MOVWF RDELAY
CICLO1
DECFSZ RDELAY,1
GOTO CICLO1
NOP
RETURN

DELAY2
MOVLW 0x0B
MOVWF RDELAY
CICLO2
DECFSZ RDELAY
GOTO CICLO2
NOP
RETURN
```

```
END
```

Apéndice I

Librería de control de robot NS Basic

```
"Function PWMMotor(Motor as Integer, Valor as Integer) as Integer
    Dim Manda as String
    Dim Recibe as String
    Dim Err as Integer

    Manda=chr(128+Motor)+chr(Valor)
    Err=SerialSend(Manda,2)
    If Err>0 Then
        MsgBox "Transmit error code "+str(Err)
    End If
    Err=SerialReceive(Recibe,1,.1)              'Ack o Nack
    If Err>0 Then
        PWMMotor=1
    Else
        PWMMotor=0
    End If
End Function

Function Sentido(Motor1 as Integer, Motor2 as Integer, Motor3 as Integer, Motor4
    Dim Manda as String
    Dim Recibe as String
    Dim Err as Integer

    If Motor1<>0 Then Motor1=1
    If Motor2<>0 Then Motor2=1
    If Motor3<>0 Then Motor3=1
    If Motor4<>0 Then Motor4=1
    Err=(Motor1*1)+(Motor2*2)+(Motor3*4)+(Motor4*8)
    Manda=chr(134)+chr(Err)
    Err=SerialSend(Manda,2)
```

```
    If Err>0 Then
        MsgBox "Transmit error code "+str(Err)
    End If
    Err=SerialReceive(Recibe,1,.1)              'Ack o Nack
    If Err>0 Then
        Sentido=1
    Else
        Sentido=0
    End If
End Function

Function TacometroMotor(Motor as Integer) as Integer
    Dim Manda as String
    Dim Recibe as String
    Dim Err as Integer
    Dim i as Integer

    TacometroMotor=-1
    Manda=chr(80+Motor)
    Err=SerialSend(Manda,1)
    If Err>0 Then
        MsgBox "Transmit error code "+str(Err)
    End If
    Err=SerialReceive(Recibe,2,.1)              'PWM Motor 1
    i=asc(Recibe)
    If i=111 Then TacometroMotor=asc(mid(Recibe,2,1))
End Function

Function Control_On_Off(Valor as Integer, Limite as Integer, Excitacion as Intege
    Control_On_Off=Excitacion
    If Valor>Limite Then Control_On_Off=Excitacion-1
    If Valor<Limite Then Control_On_Off=Excitacion+1
    If Control_On_Off<0 Then Control_On_Off=0
    If Control_On_Off>100 Then Control_On_Off=100
End Function"
```

Bibliografía

[1] *PPRK: Palm Pilot Robot Kit* http://www-2.cs.cmu.edu/~pprk/

[2] Kevin Mukhar and Dave Johnson.: "The Ultimate Palm Robot", McGraw–Hill/Osborne, U.S.A., 2003.

[3] *Municipio virtual de Aguascalientes* http://www.muniags.gob.mx/ciudad/universidades.asp

[4] *The Personal Rover Project* http://www-2.cs.cmu.edu/~personalrover/

[5] *BugBotics* http://www.andrew.cmu.edu/~bcf/LegoBotics/

[6] *Profesor Illah R.Nourbakhsh* http://www-2.cs.cmu.edu/~illah/

[7] *Microbótica* http://www.microbotica.com/

[8] *Universidad Bonaterra - Robótica* http://www.duxpoint.com/robotics/robotics.html

[9] *Lego Mindstorms Home* http://mindstorms.lego.com/eng/default.asp

[10] *Mars Exploration Rover Mission* http://mars.jpl.nasa.gov/mer/