19

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRONICAS

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO

ESPECIALIDAD EN MECATRONICA

P R E S E N T A :

ULISES MARTIN PEÑUELAS RIVAS

DIRECTOR DE TESIS: M.I. VICTOR JAVIER GONZALEZ VILLELA

MEXICO, D.F. 2002

Dedicatoria

A mi abuela: Trini.
A mi abuelo: Martín.
A mi madre: María.
A mi padre: Alfredo.
A mis hermanos Alfredo, Giovanna, Pamela e Isidro.
A mi otra familia: Daniel, Argelia, Isa, Alaide y Josué.

Agradecimientos

Al M.I. Victor Javier Ganzález Villela agradezco su entusiasmo y motivación constante, expresada siempre en su calidez y autenticidad humana.

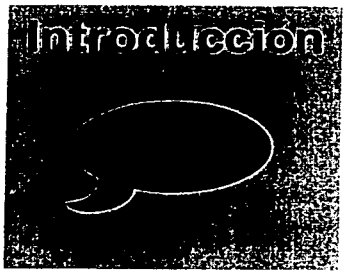
A mis compañeros del laboratorio de mecatrónica: Germán, Enrique, Eduardo, Sergio, Marco, Gabriel, Fernando y Carmen.

Y a todos los que olvidé mencionar.

DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Contenido

CONTENIDO



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Introducción

La tesis "Desarrollo de un Sistema Basado en un PIC 16F874 para Aplicaciones Mecatrónicas" es un documento cuyo objetivo principal es servir de material didáctico para el desarrollo de sistemas mecatrónicos. Siendo la mecatrónica un tema de tanta proyección hacia el futuro, se está intentando impulsar a los alumnos de ingeniería a instruirse en el tema y facilitar el desarrollo de estos sistemas planteando alternativas de implementación.

Para poder llevar acabo este trabajo se analizaron diferentes alternativas de desarrollo; se utilizaron diversos paquetes de diseño como son el Protel 99 SE para el diseño electrónico, el Mechanical Desktop 4 para el diseño mecánico, etc. con el objetivo de que los archivos queden susceptibles a modificaciones posteriores. También se hace referencia a varios manuales como son los del MPLAB 5.2, IC-Prog 1.04, CC5x y las hojas de datos de todos los circuitos integrados utilizados, que por razones de tamaño no están incluidos en la tesis pero se quedan recopilados en el archivo del Departamento de Mecatrónica.

Este trabajo no pretende abarcar todos los aspectos del conocimiento que comprende la formación de un ingeniero mecatrónico, sino brindar una herramienta y una propuesta para desarrollo de aplicaciones mecatrónicas, dando por entendido que el lector ya tiene conocimientos básicos sobre los temas que ésta trata.

Aquí se presentan los conceptos y métodos que se pueden utilizar para el desarrollo de nuevas tecnologías que permiten a corto plazo resolver problemas como la conducción autónoma de vehículos, la manipulación de objetos, la toma automática de decisiones, control de procesos, etc.

Los capítulos están divididos en los temas generales que abarca el diseño mecatrónico: electrónica, informatica y mecánica; a pesar que estos temas son tomados por separado, al final la unión del conocimiento propuesto en los capítulos formaran una poderosa herramienta en el desarrollo de sistema con fines mecatrónicos.

En el Capítulo 1 se describen algunos conceptos necesarios para poder hablar en temas subsecuentes en un lenguaje común. El capítulo en sí, es un recopilación de definiciones.

El Capítulo 2 habla de los microcontroladores de uso mas frecuente; da al ingeniero mecánico una alternativa de selección

además de ampliar sus conocimientos sobre conceptos en el campo electrónico.

En el Capítulo 3 se muestra una breve descripción del microcontrolador PIC 16F874 de Microchip© subrayando los aspectos más interesantes y funcionales de este chip para un desarrollo en aplicaciones mecatrónicas.

El Capítulo 4 describe el diseño de circuitos eléctricos, entre los más importantes se encuentran el de la tarjeta de desarrollo PIC 16F874, la tarjeta para sensores ópticos digitales y/o analógicos y la tarjeta de circuito puente H para motores continuos. También se describen algunos circuitos para aplicaciones mecatrónicas. En el diseño de las tarjetas se describen sus partes, su posible configuración y el uso de las mismas.

En el capítulo 5 se encuentra una breve descripción del funcionamiento del software que se necesita para desarrollar aplicaciones sobre la tarjeta de desarrollo PIC 16F874. El software que ahí se describe es el del IDE MPLAB 5.2, el del programador IC-Prog 1.04 y el lenguaje C para PICs, CC5x.

En el Capítulo 6 se propone un vehículo de guía automática genérico multiconfigurable, el cual es considerado uno de los desarrollos mecatrónicos más completos en cuanto a las posibilidades de diseño de aplicaciones a partir de él.

En el Capítulo 7 se muestra el análisis cinemático del AGV propuesto en el Capítulo 6, además de un sistema de control de lazo cerrado que proporciona al vehículo la capacidad de controlar su velocidad, radio de giro y dirección.

Los apéndices son una parte importante y complementaria de la información que se encuentra contenida en los capítulos. Ahí se encuentra información técnica de algunos microcontroladores PIC, planos del diseño mecánico del AGV, diagramas electrónicos y layout de las tarjetas electrónicas.

Cualquiera de los temas aquí presentados puede servir para el seguimiento y elaboración de trabajos subsecuentes, o para tomar sólo la parte de diseño que se necesite para la elaboración de un proyecto.

DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Definiciones Importantes

1.1 Mecatrónica

La mecatrónica es una de las disciplinas más completas de la ingeniería, ya que integra conocimientos de diversas ramas de la ciencia como son la mecánica, la electrónica y la computación.

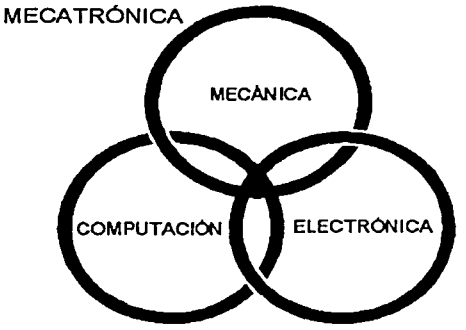


Fig. 1.1 La mecatrónica se integra de la mecánica, electrónica y la computación para la integración de un sistema inteligente

En otras palabras, la mecatrónica es la integración de sistemas de control basados en microprocesadores, sistemas electrónicos y sistemas mecánicos.

La mecatrónica por su amplitud, en cuanto al conocimiento que abarca, es muy fácil encontrar profesionales de diferentes áreas trabajando en un mismo proyecto, aunque lo ideal es que cada uno tenga los conocimientos básicos en las diversas áreas que implica la mecatrónica.

1.2 Sistema Mecatrónico

Un sistema mecatrónico no es simplemente la unión de sistemas electrónicos y mecánicos, y es más que un sistema de control: es una integración completa de todo lo anterior. Un sistema mecatrónico es la conjunción de herramientas de diversas áreas para el desarrollo de un proyecto específico, es decir, es la aplicación práctica de la definición de mecatrónica.

Un sistema mecatrónico consta de las siguientes partes:

- Unidad central de proceso: computadora, microcontrolador, microprocesador, memoria, etc.
- Software
- Electrónica de potencia
- Actuadores: eléctricos, electrónicos, neumáticos, hidráulicos, etc.
- Estructura mecánica: brazos, puertas, patas, grippers, válvulas etc.

- Sensores
- Sistemas de control: lazo abierto o cerrado, por hardware y por software

Hoy en día es muy común encontrar sistemas mecatrónicos

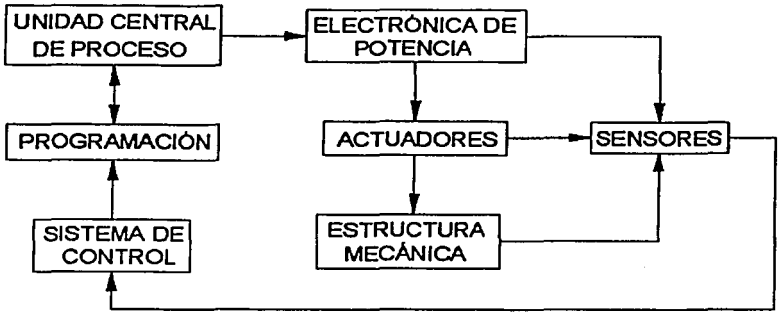


Fig. 1.2 Componentes que integran a un sistema mecatrónico

funcionando, sólo hace falta poner un poco de atención para localizarlos, por ejemplo: una bomba de gasolina, un horno de microondas, un reproductor de discos compactos, etc; todos estos ejemplos contienen todas las partes que constituyen un sistema mecatrónico a pesar de la diferencia en cuanto a su aplicación.

1.3 Robótica

La robótica se define como el conjunto de conocimientos técnicos y prácticos que permiten concebir, realizar y automatizar sistemas basados en estructuras mecánicas poliarticuladas, dotados de un determinado grado de inteligencia y destinados a la producción industrial o a la sustitución del hombre en diversas tareas.

Los mecanismos robotizados son utilizados principalmente en la industria donde desarrollan tareas de difícil manipulación para el hombre e incluso trabajos en condiciones hostiles y peligrosas, tales como las que se llevan a cabo en ambientes con temperaturas elevadas.

El concepto de robótica implica una cierta idea preconcebida de una estructura mecánica universal capaz de adaptarse como el hombre a diversos tipos de acciones y en las que concurren en mayor y menor grado según los casos, las características de movilidad, programación, autonomía y multifuncionalidad, sin embargo, también abarca una amplia gama de dispositivos con diversos trazos físicos y funcionales asociados a la estructura mecánica, a sus características operativas y al campo de aplicación

para el que se han concebido. Además, todos estos factores están íntimamente relacionados, de tal forma que la configuración y el comportamiento de un robot condicionan su adecuación para un campo determinado de aplicaciones y viceversa, esto a pesar de la versatilidad inherente al propio concepto de robot.

Aplicaciones de la Robótica
El empleo de los robots se extiende por todo el mundo a un ritmo acelerado. Todos los trabajos que hasta hace poco, era inconcebible que los realizase una máquina, hoy día los efectúa un robot.

Algunas aplicaciones son:
- Carga y descarga de máquinas
- Soldadura
- Aplicación de pintura
- Corte
- Pulido
- Forja
- Prensado
- Fundición

Todo robot cuentan con las siguientes características:
- Grados de libertad
- Zonas de trabajo y dimensiones del manipulador
- Capacidad de carga
- Precisión de la repetibilidad
- Velocidad
- Coordenadas de movimientos
- Tipo de actuadores
- Programabilidad
- Capacidad de memoria

1.5 Sensores y Transductores
El término sensor se refiere a un elemento que produce una señal relacionada con la cantidad que se esté midiendo. Con frecuencia se utiliza el término transductor en vez de sensor. Transductor se define como el elemento que al someterlo a un cambio físico experimenta un cambio relacionado, es decir los sensores son transductores[17].

1.6 Actuadores
Los actuadores son los elementos de los sistemas de control que transforman la salida de un microprocesador o sistema de control

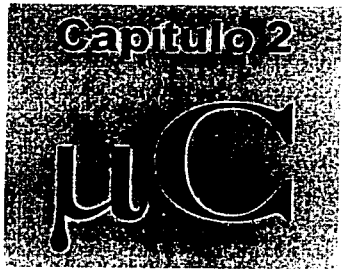
en una acción de control para una maquina o dispositivo[17]. En otras palabras, los actuadores son elementos que transforman una señal en una acción mecánica. Entre los actuadores más comunes se encuentran los motores, pistones, tendones, etc.

1.7 Valores Continuos y Discretos

Cuando se tienen valores que conmutan entre dos valores, un valor superior y uno inferior, sin la posibilidad de caer en un valor intermedio, se dice que se tienen valores discretos. Pero si entre el valor superior y el inferior es posible tomar un rango amplio de valores, se dice que los valores son continuos.

Un ejemplo de datos discretos son los booleanos, donde el valor superior que se puede tomar es el uno lógico equivalente a 5 V, mientras que el inferior es el cero lógico o 0 V. En cambio, tomando los mismos límites que del ejemplo anterior, un valor continuo puede ser cualquiera que esté entre 0 y 5 V.

Se suele decir que un sistema digital es discreto mientras que uno analógico es continuo. También es frecuente encontrar que los valores son discretizados, es decir, que después de ciertos rangos estos valores serán considerados como cero o uno lógicos.



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Introducción a los Microcontroladores

2.1 Microcontroladores

El microcontrolador es uno de los logros más sobresalientes del siglo XX. Hoy en día existen casi 15,000 millones de microchips de alguna clase en uso. Para la mitad del siglo XXI, es posible que el microcontrolador típico tenga mayor poder de cómputo que las supercomputadoras más veloces de hoy.

Se pueden encontrar en cualquier sitio: hornos de microondas, refrigeradores, automóviles, aviones, mandos a distancia, radios, televisores, etc.

A modo de comparación se puede decir que una tarjeta de felicitación musical contiene procesadores con mayor poder de cómputo que las computadoras más grandes del mundo en 1971.

Se predice que en siete generaciones contadas a partir de ahora, estos chips incorporarán reconocimiento del habla a procesadores de textos y sistemas de entrada de pedidos. Producirán gráficos en 3D del tamaño de muros para televisión, teleconferencias e incluso películas personalizadas. Dirigirán vehículos para optimizar la seguridad y crearán mundos virtuales. Supervisarán la salud, reemplazarán partes perdidas del cuerpo y, a través de una retícula de miles de millones de sensores, nos conectarán con el mundo en formas que sólo podemos imaginar vagamente.

En conclusión, el microcontrolador puede ser considerado como uno de los inventos más importantes del siglo XX, y posiblemente también del XXI.

¿Qué es un microcontrolador?

En un principio, los sistemas de control se hacían exclusivamente con componentes discretos lógicos, eran cajas grandes, pesadas; posteriormente se utilizaron microprocesadores y el sistema de control entero podía encajar dentro de una tarjeta de circuito impreso. El proceso de miniaturización continua, todos los componentes que se requieren para un sistema de control se construyen dentro de un chip, el microcontrolador.

Un microcontrolador es un circuito integrado que incluye todos o casi todos los componentes necesarios para tener un sistema de control completo, además de un conjunto de instrucciones específicas programables.

La idea original que hay detrás de un microcontrolador, es superar las limitaciones de la UCP (Unidad Central de Proceso),

permitiendo tener una computadora completa, con memoria, entrada/salida, interrupciones, etc.

Los microcontroladores son "la solución en un chip", incluyen típicamente:

- UCP
- RAM
- EPROM/PROM/ROM/Flash
- E/S - serie y paralelo
- Temporizadores/Contadores
- Sistema de interrupciones
- Los modelos más potentes incluyen además sistemas auxiliares (Convertidor A/D, convertidor D/A, modulación de ancho de pulso o PWM, módulos de captura y comparación CPP, ...)

Las aplicaciones
Los microcontroladores frecuentemente se encuentran en:

- Aparatos electrodomésticos: hornos de microondas, estufas, hornos electricos, refrigeradores, televisión, equipos de video, equipos de sonido, etc.
- Equipos informáticos: impresoras, copiadoras, módems, unidades de disco, CD ROM, etc.
- Automóviles: mando de sistemas del automóvil (ABS, inyección, encendido...), diagnósticos, clima, etc.
- Climas controlados en: invernadero, fábrica, casa, refrigeradores, etc.
- Instrumentación: sistemas aeroespaciales, sistemas de medición, control de ancho de pulso, PI, PD, PID, etc.
- Industria: PLC, robots en general, tarjetas de adquisición de datos, sistemas automáticos, etc.

Como se menciono anteriormente, los sistemas basados en microprocesador y microcontroladores se usan extensivamente en la robótica. En esta aplicación, muchas tareas específicas podrían estar distribuidas entre un gran número de controladores dentro de un sistema, estos controladores pueden estar comunicados con un procesador central (maestro), éste habilita la información para ser procesada por la computadora central o para ser pasada a otros controladores en el sistema.

Otra aplicación especial de los microcontroladores es la adquisición de datos analógicos: temperatura, humedad, nivel, etc.

El tamaño de los microcontroladores es pequeño y consumen muy poca energía, lo que los hace ideales para sistemas portátiles y autónomos.

Tipos de microcontroladores
Los microcontroladores más comunes manejan 4, 8, 16, y 32 bits de ancho de palabra.

Existen microcontroladores con funciones especificas como por ejemplo para:

- Comunicaciones
- Manejo del teclado
- Procesamiento de señal
- Procesamiento de video
- Despliegue de información
- Otras tareas

Diferencia entre microprocesador y microcontrolador
El microprocesador es un circuito integrado que contiene UCP, también llamada procesador. La UCP está formada por la unidad de control que interpreta las instrucciones, y la localización de los datos, que lee o almacena.

Los pines de un microprocesador sacan al exterior las líneas de sus buses de direcciones, datos y control, para permitir conectarle con la memoria y los módulos de E/S y configurar una computadora formada por varios circuitos integrados. Se dice que un microprocesador es un sistema abierto porque su configuración es variable de acuerdo con la aplicación a la que se destine.

El microcontrolador es el conjunto de una aplicación con un microprocesador, es decir, incluir en un chip toda la circuitería y elementos periféricos necesarios para hacer una computadora. Por lo anterior se puede intuir que no es del todo posible poder cambiar la configuración interna del chip por lo que se busca hacer microcontroladores multifuncionales configurables bajo software.

En resumen:
- Un microprocesador es un **sistema abierto** con el que se puede construir una computadora con las características que se desee, acoplándole los módulos necesarios (Fig. 2.1).

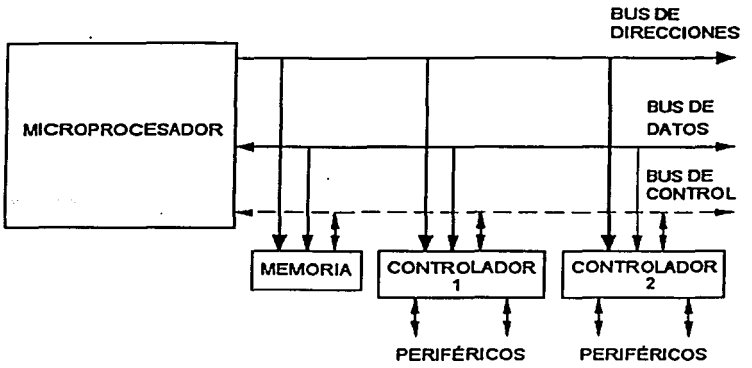


Fig. 2.1 Estructura de un sistema abierto basado en un microprocesador

- Un microcontrolador es un **sistema cerrado** que contiene una computadora completa y de presentaciones limitadas que no se pueden modificar (Fig. 2.2).

Fig. 2. 2 El microcontrolador es un sistema cerrado

2.2 El Mercado de los Microcontroladores

Toda la industria maneja microprocesadores y microcontroladores en sus sistemas de control, adquisición de datos, sistemas automáticos, sistemas robóticos y de intercomunicación.

Los PLCs se construyen, incluso, con base en microcontroladores. Algunos tienen entradas de señales analógicas, las cuales son acondicionadas por el mismo microcontrolador.

Los robots generalmente están construidos con sistemas mecatrónicos esclavos, uno en cada articulación, y uno o varios microcontroladores maestros. Estos últimos indican a los esclavos cual es la acción a realizar mientras que los esclavos mandan información del porcentaje de acción realizada o de su localización.

Las tarjetas de adquisición de datos, como su nombre lo indica, se dedican a la adquisición, acondicionamiento y almacenamiento de la información que es enviada desde los sensores en los procesos. Una tarjeta de adquisición de datos puede trabajar como un sistema independiente de los demás procesos y cuando se genere cierto evento, alguna anomalía por ejemplo, entra al sistema general como una interrupción y activa un evento, como la activación de una alarma. También puede ser retirada de la línea principal del sistema y ser leída la información capturada en un intervalo de tiempo.

El mercado del automóvil es el sector que más microcontroladores compra. Varias familias de microcontroladores se desarrollaron específicamente para aplicaciones del automóvil, posteriormente se modificaron para otras aplicaciones. El sector automotriz es exigente: los circuitos electrónicos deben operar bajo temperaturas extremas y deben resistir las vibraciones, interferencias y ruidos eléctricos (EMI). La electrónica empleada debe ser fiable, pues un fallo puede provocar un accidente. Debido a la competencia, el precio de estos circuitos es bajo. El sector automotriz no es el único mercado que está creciendo.

¿Qué microcontrolador usar?
Para decidir qué tipo de microcontrolador utilizar para llevar a cabo un proyecto, se debe considerar lo siguiente:

- Identificar cuántos y qué tipos de entradas y salidas se van a necesitar.
- ¿Qué velocidad de procesamiento se necesita?
- ¿Qué capacidad de memoria de programa se requiere?
- ¿Qué capacidad de memoria de datos se necesitará?, ¿Esta memoria deberá ser volátil o no?
- ¿Qué herramientas de desarrollo están disponibles y cuanto cuestan?
- ¿Qué clase de documentación se tiene disponible (manuales de referencia, notas de aplicación, libros)?
- ¿Tiene el fabricante versiones compatibles en número de patas y programación con otras características en periféricos (convertidores A/D, capacidad de memoria, tipo de memoria)?
- ¿Qué protocolos de comunicación maneja?
- ¿Qué tamaño tiene?
- ¿Cómo se programará?

Los Fabricantes de μP y μC
Los más conocidos fabricantes que se encuentran en el mercado son: Intel, AMD, Motorola, IBM, TI, Cyrix, Hitachi, LSI, IDT, NEC, Mitsubishi, Hitachi, Phillips, Matsushita, AT&T, Toshiba, Microchip.

2.3 Características de los Microcontroladores

2.3.1 Técnicas de fabricación

CMOS (Semiconductor de Oxido de Metal Complementario)
Este es el nombre de la técnica con que se fabrican la mayoría de los microcontroladores (sino todos). Los dispositivos CMOS tienen las siguientes características:

- Consumen muy poca corriente y pueden ser alimentados por baterías durante mucho tiempo.
- El reloj del sistema puede detenerse y ponerse el dispositivo "en modo sleep" para bajar más aún su consumo.
- CMOS tiene una alta inmunidad al ruido eléctrico.

PMP
PMP es un proceso de implantación de alta energía que permite que el microcontrolador ROM pueda ser programado después de la metalización final.

Normalmente la programación de dispositivos ROM se lleva a cabo en el segunda metalización con otras nueve o diez capas encima. Esto significa que el modelo de ROM debe especificarse en el proceso de la producción, por lo que su producción es lenta para la fase de pruebas. Sin embargo, con PMP, los troqueles pueden fabricarse totalmente a través de metalización (sólo las capas del pasivación necesitan ser agregadas). Esto significa que una vez realizadas las pruebas y correcciones el proceso se vuelve más ágil.

2.3.2 Tipos de arquitectura

Von Neumman
Los μC von Neumman tienen un sólo bus de datos por el cual circulan instrucciones y datos. Las instrucciones del programa y los datos se guardan conjuntamente en una memoria común. Cuando la UCP se dirige a la memoria principal, primero saca la instrucción y después saca los datos necesarios para ejecutarla, esto retarda el funcionamiento de la UCP (Fig. 2.3).

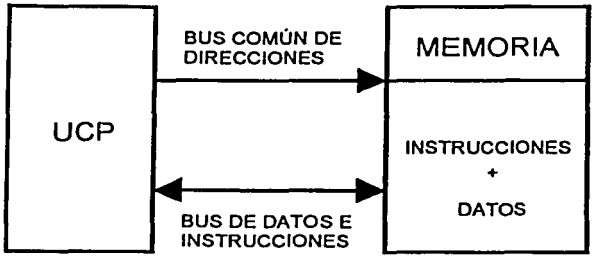


Fig. 2.3 Arquitectura von Neumman. La UCP se comunica a través de un sistema de buses de memoria, donde se guardan las instrucciones y datos

Harvard

Los μC Harvard tienen separados el bus de datos y el bus de instrucción. Esto permite el proceso paralelo: Cuando una instrucción está siendo "captada", la instrucción actual está utilizando el bus de datos. Una vez que la instrucción actual está finalizada, la siguiente instrucción está disponible dentro de la UCP. Este procedimiento de trabajo permite una ejecución más rápida (Fig. 2.4).

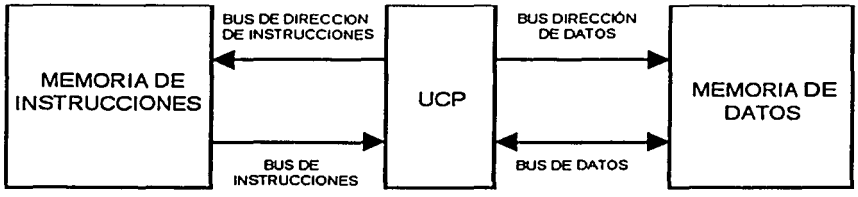


Fig. 2.4 Arquitectura Hardvard. La UCP se comunica a través de dos sistemas de buses de memoria, donde en uno se guardan las instrucciones y en el otro los datos

CISC (Complex Instruction Set Computer)

Casi todos los microcontroladores actuales, tienen un conjunto de instrucciones complejo y amplio. El conjunto de instrucciones típico de un microcontrolador es de más de 80. Algunas de estas instrucciones son muy potentes y otras son especializadas para una tarea concreta.

Algunas instrucciones acceden a ciertos espacios de dirección o registros, mientras que otras instrucciones tienen modos de direccionamiento único y específico, es decir, el conjunto de instrucciones es bastante heterogéneo.

La ventaja de la arquitectura CISC es que algunas instrucciones son "macros" ya que son tan poderosas que equivalen a muchas instrucciones simples.

RISC (Reduced Instruction Set Computers)
La tendencia actual en la industria de los microcontroladores es la de tener un conjunto de instrucciones reducido. Al tener menos instrucciones, el chip es más pequeño, más sencillo, más rápido (al ser más simples las conexiones internas dentro del circuito, existen menos inductancias y capacitancias parásitas con lo que se puede aumentar la frecuencia de reloj) y además tiene un consumo de energía menor.

Algunos de los rasgos típicos de un procesador del tipo RISC son:
1) El conjunto de instrucciones es ortogonal (simétricas), esto simplifica el proceso de programación, pues cada instrucción puede operar con cualquier registro y usar cualquier modo de direccionamiento.
2) Las instrucciones no tienen combinaciones especiales, excepciones ni restricciones.

SISC (Specific Instruction Set Computer)
Al contrario del conjunto de instrucciones de propósito general que tienen los microprocesadores para PCs, el conjunto de instrucciones de los microcontroladores se diseñó para el propósito específico de control:

- Las instrucciones de entrada/salida son eficaces y sencillas
- Instrucciones específicas de operación con bits
- Instrucciones adecuadas para operaciones con tablas de datos

Los microcontroladores vienen ahora con una serie de características que son de gran ayuda para el ingeniero:

- Temporizadores
- Temporizador perro guardián (Watchdog)
- Circuitos para "dormir/despertar" al microcontrolador
- Diferentes modos de direccionamiento de entrada/salida
- Circuitos convertidores analógico/digital, etc.

Estas nuevas características específicas para control son cada vez más numerosas y vienen incorporadas sin aumento de precio en los nuevos dispositivos.

2.3.3 Opciones de Memoria

EEPROM
(Memoria Programable y Borrable Eléctricamente)
Muchos microcontroladores tienen incorporada una cantidad limitada de memoria EEPROM dentro del chip. El objetivo es tener

una pequeña cantidad de memoria donde poner una serie de parámetros que puedan ser cambiados si la aplicación lo requiere.

Este tipo de memoria es relativamente lenta, y el número de veces que se puede borrar/grabar está limitado.

FLASH
Las memorias FLASH son mejores que las EEPROM cuando se necesita almacenar el programa de control en una memoria no volátil.

Estas memorias son más rápidas que las memoria EEPROM y permiten más ciclos de borrado/grabación.

RAM
Es útil cuando se tiene un programa de gran tamaño; es mucho más rápida que la memoria no volátil; no hay límite en el número de veces que puede ser grabada. Se utiliza en aplicaciones donde se cambian cantidades grandes de datos frecuentemente.

Memoria de Programa
Este tipo de memoria no volátil permite ser reprogramada en el sitio sin quitar el microcontrolador del sistema que controla.

Una aplicación típica de esta memoria es el sector automotriz, pues se puede reprogramar el microcontrolador *in situ*[1]. Por ejemplo, cambiando los parámetros de la inyección electrónica para adaptar el motor a normas de emisión de humos o ponerlo a punto después de un periodo de desgaste.

Memoria de Datos
La memoria de datos, es una memoria temporal, puede ser tanto RAM como FLASH o EEPROM. Ahí se almacenan los datos temporales que sirven para crear los registros de las variables que utiliza el programa cargado en la memoria de programa. También se puede guardar gran cantidad de información, por ejemplo, la memoria de programa puede funcionar a manera de sistema operativo (BIOS) y utilizar la memoria de datos para tomar los programas a ejecutar.

La memoria de datos siempre es expandible en un microcontrolador, pero es limitada su expansión.

[1] *in situ*.- El lugar y momento en donde se esté.

OTP (One Time Programmable)
Un OTP es una memoria PROM (memoria programable de sólo lectura). Una vez que se graba con un grabador de EPROM normal, ésta no puede modificarse ni borrarse.

Como los ciclos de desarrollo de productos son cada vez más cortos, es interesante para los fabricantes de microcontroladores ofrecer OTPs como una opción. Estas memorias son útiles cuando se necesita un gran número de unidades y se está seguro de que el programa va a ser el definitivo.

Protección del software
La protección del software se realiza por encriptación o protección del fusible, el software programado es protegido contra personal no autorizado (ingeniería inversa, modificaciones, piratería, etc.). Si el software es tratado de leer o modificar, el contenido de la memoria es borrado.

2.3.4 Funcionamiento de la Alimentación y Bajo Voltaje

Existen reglas con respecto a los transistores:

- La cantidad de potencia que disipan es proporcional a su tamaño
- Su retraso de propagación es proporcional a su tamaño
- Su costo es proporcional al cuadrado de su tamaño

Si se hace un transistor más pequeño, se mejora el consumo de energía, velocidad y el costo. El único inconveniente es que son más complicados y difíciles de fabricar.

Protección de Brownout
Es un circuito que protege contra sobretensiones de alimentación.

Idle/Halt/Wakeup
El dispositivo puede ponerse en el modo de Ocioso/Parada (IDLE/HALT) por medio del software. En estos modos de funcionamiento, en la memoria RAM no se pierde ningún dato. En modo Idle (ocioso), todas las actividades se detienen a excepción de:

- La circuitería asociada al oscilador
- La lógica del perro guardián
- El amonestador del reloj
- El cronómetro ocioso (un cronómetro corriente libre)

Los requisitos de consumo de energía en este modo son típicamente alrededor del 30% de potencia en funcionamiento normal. De este modo de funcionamiento se sale por un estímulo que puede ser:

- Interrupción desde un temporizador
- Puerto serie
- Un contador/temporizador "ocioso", puede despertar periódicamente al microcontrolador para verificar si todo funciona correctamente, después el chip vuelve a dormirse

El modo Idle es sumamente útil para captura de datos en un sitio remoto: el microprocesador se despierta a intervalos regulares, toma sus medidas, almacena o envía sus datos y vuelve a dormirse. En modo de Parada (Halt), todas las actividades se detienen (incluso los cronómetros y contadores). La única manera de despertarse es por una interrupción (puerto de E/S, teclados).

2.3.5 Entradas y Salidas

UART (Unidad Universal de Transmisión Recepción Asíncrona)
Es un dispositivo adaptador del puerto serie para comunicaciones asíncronas.

USART (Unidad Universal de Transmisión Recepción Síncrona y Asíncrona)
Es un adaptador del puerto serie para comunicaciones asíncronas o síncronas. Los dispositivos que usan un USART son típicamente más rápidos (tanto como 16 veces) que los que usan una UART.

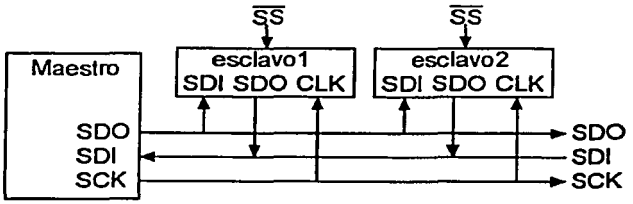
SPI (Interfase Periférica Serie)
El modo SPI* permite sincronizar la transmisión y recepción de datos simultáneamente. Para poder llevar a cabo este tipo de comunicación, se utilizan tres hilos.

SCI (Interfase de Comunicación Serie)
La interfaz serie asíncrona o SCI es comparable a una UART programable por software. Puede funcionar en modo asíncrono full dúplex para conectar al microcontrolador con cualquier equipo provisto con una conexión del mismo tipo (terminal, módem, etc...)

*SDI – Serial Data Input
SDI – Serial Data Output
SCK – Serial Clock
SS – Slave Selection

o bien en modo síncrono semidúplex maestro o esclavo, para aplicaciones mas especiales.

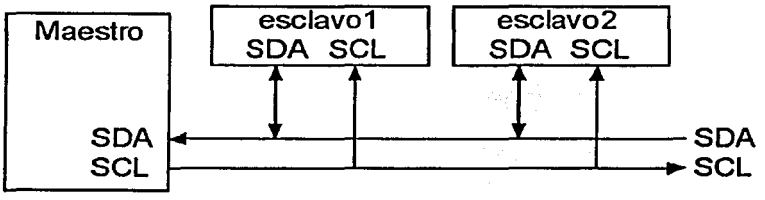
Fig. 2.5
Comunicación SPI entre maestro y esclavos



I^2C

(Circuito Integrado Interno)
El I^2C* es un bus de 2 hilos de interfase serie desarrollado por PHILIPS. Se desarrolló para unir aplicaciones de 8 bits y es muy usado en electrónica de consumo, en la industria automotriz e industrialmente. Este bus puede comunicar además distintos periféricos (Fig. 2.6).

Fig. 2.6
Comunicación I²C entre maestro y esclavos



El I^2C bus es una interfase de red de dos hilos, multimaestro, multiesclavo con detección de colisión. Pueden existir hasta 128 dispositivos en la red y pueden estar a una distancia de hasta 10 metros. Cada nodo (microcontrolador o periférico) puede iniciar un mensaje, y transmitir o recibir datos (Fig. 2.7).

Las dos líneas de la red consisten en una línea de datos serie y una línea de reloj. Cada nodo de la red tiene una única dirección que acompaña cualquier mensaje transmitido entre nodos. Como sólo se necesitan dos cables es fácil interconectar los distintos dispositivos.

* SDA – Serial Data
SCL – Serial Clock

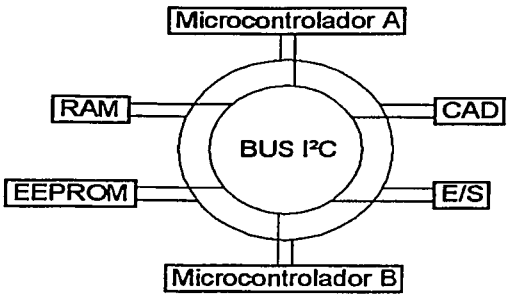


Fig. 2.7
Conexión de comunicación I²C con multimaestros

Microwire/Plus
Es una interfase de comunicaciones serie síncrona bidireccional. La usan los dispositivos fabricados por la compañía National Semiconductor (microcontroladores, convertidores A/D, drivers de displays, EEPROMS, etc.).

CAN & J1850 (Controller Area Network)
Es un sistema de cableado que fue desarrollado conjuntamente entre BOSH e INTEL para el cableado de automóviles, es el sistema de cableado estándar multiplexado que se usa actualmente en la industria del automóvil.

Convertidor Analógico a Digital (CAD)
Convierten tensión analógica a su valor digital, se utiliza para adquisición de datos del mundo analógico (temperatura, humedad, etc.), dependiendo del tipo de sensor conectado.

Existen varios tipos de convertidores A/D:

- **Convertidores A/D de Aproximaciones Sucesivas**
  Se usan en la mayoría de los microcontroladores. En esta técnica, el convertidor obtiene un bit en cada comparación (comenzando por el bit más significativo) y verifica la siguiente comparación, si es mayor o menor. Toma siempre la misma cantidad de tiempo para cada muestra. Es lento, pues para cada bit se necesita al menos un ciclo de reloj.

- **Convertidores A/D Delta Sigma**
  Este tipo de convertidor A/D se encuentra dentro de la gama alta de los DSP*.

* Procesadores Digitales de la Señal

☞ **Convertidores A/D FLASH**
En la arquitectura básica de los convertidores A/D más rápidos, existen ventajas y desventajas:

Ventajas
- La conversión la realizan en un ciclo de reloj.

Desventajas
- Para tener un convertidor A/D con una resolución de n bits, se necesitan 2^n comparadores.
- Para hacer funcionar todos estos circuitos, se necesita mucha corriente eléctrica y por encima de 10 bits, el número de comparadores que se requiere no es manejable.

Convertidor Digital a Analógico (CDA)
Sirven para obtener una tensión analógica a partir de un valor digital.

Ejemplo: En un sistema de 8 bits (2^8 = 256) alimentado con una tensión de 5 V, el número 50 sería convertido a una tensión analógica de 0.9765 volts

$$\frac{\text{no. en binario}}{\text{no. de Salidas}} * \text{Voltaje de Referencia} = \frac{50}{256} * 5 = 0.9765\,\text{V}$$

☞ **Convertidor D/A PWM (Modulación de Ancho de Pulso)**
Un tren de pulsos se genera regularmente hacia un filtro paso bajo para generar una tensión proporcional al "duty cycle" (ancho del pulso o tiempo que está a nivel alto la señal durante cada ciclo de pulso).

☞ **Contador de Pulsos**
Un contador de pulsos es un contador de eventos. Cada pulso incrementa el registro contador, almacenando el número de veces que ha ocurrido un evento.

☞ **Entrada de Captura**
Sirve para medir intervalos de tiempo (o frecuencias) entre eventos, copiando el valor de un temporizador de funcionamiento libre en un registro específico cuando ocurre el evento.

- **Comparador**
  Estos comparadores funcionan como comparadores normales, con la característica principal de que los valores de entrada y de salida están disponibles en el bus de datos.

2.3.6 Interrupciones

El procesador puede estar ejecutando su programa principal, y sólo responderá a los periféricos cuando ellos lo necesiten. Cuando el procesador recibe una interrupción, abandona el programa principal, identifica al periférico que ha producido la interrupción y ejecuta la subrutina de atención a la interrupción adecuada, para después volver al programa principal.

Ventajas de las interrupciones:

- La velocidad de respuesta a un evento externo.
- Se reduce la cantidad de software (y tiempo de proceso) añadido al programa principal para preguntar constantemente a los periféricos si necesitan atención.

La mayoría de los microcontroladores tienen al menos una interrupción externa, ésta puede ser seleccionable entre flanco de subida, de bajada o nivel.

- Disparo de Interrupciones por Flanco: No depende del tiempo que está activada la señal de interrupción, pero es susceptible a los "glitches" (fallas imprevistas o picos de interferencias).
- Disparo de Interrupciones por Nivel: Tienen que estar a nivel alto (o bajo) durante un tiempo determinado, pero no es susceptible a los "glitches".

Los microcontroladores de 4 bits tienen al menos un sistema de interrupciones no vectorizado. Los microcontroladores de 8, 16 y 32 bits tienen un sistema de interrupciones vectorizadas con una jerarquía.

Transferencia de E/S por Consulta de estado

La consulta de estado no es una característica de los microcontroladores, es lo que uno tiene que hacer si el microcontrolador elegido no tiene interrupciones.

La consulta de estado es una técnica de software en la que el microcontrolador pregunta constantemente al periférico si necesita ser atendido. El periférico pone una bandera en 1 cuando tiene un dato preparado para transferir al microcontrolador, esta bandera es leída por el microcontrolador en la siguiente consulta de estado.

Se pueden controlar varios periféricos usando esta técnica; se deben consultar dichos periféricos secuencialmente y el microcontrolador saltará a diferentes rutinas de entrada salida dependiendo de qué bandera se ha puesto.

Interrupciones Enmascarables
Las interrupciones no enmascarables no se pueden inhibir y siempre hay que atenderlas. La ventaja de las interrupciones enmascarables es que se puede desactivar una interrupción en particular (por ejemplo la UART) durante un instante crítico. Entonces las interrupciones enmascaradas serán ignoradas. Muchos microcontroladores y microprocesadores tienen algún tipo de Habilitador Global de Interrupciones (GIE = Global Interrupt Enable) que permite inhibir todas las interrupciones enmascarables de una vez.

Interrupciones Vectorizadas
Cuando se recibe una interrupción se debe localizar al periférico que demanda atención, esto se puede hacer al comienzo de la subrutina de atención a la interrupciones preguntando uno por uno a los periféricos hasta saber cuál ha producido la interrupción.

Esto es muy lento, pero tiene sus ventajas:

- El programador decide la prioridad, pues el periférico más importante para la aplicación se verifica en primer lugar y así sucesivamente.
- Con las interrupciones vectorizadas, el periférico directamente nos indica a través del bus de datos (con un direccionamiento indirecto a memoria llamado "vector de interrupción") dónde comienza su subrutina particular de atención a la interrupción, es decir se identifica.

2.3.7 Características Especiales de los Microcontroladores

Temporizador Watchdog (WDT)
Un temporizador perro guardián soluciona la recuperación del sistema ante un problema. Por ejemplo, si un programa entra en un bucle infinito, o si un fallo de hardware le impide funcionar, entonces el temporizador de perro guardián reinicializará el sistema en un intervalo predeterminado. El problema puede continuar existiendo, pero al menos se tiene una vía de solución (se puede reiniciar el sistema en un modo de funcionamiento mínimo o auxiliar). Esta característica es muy útil para sistemas desatendidos.

DSP (Procesador Digital de Señales)
Los DSPs ejecutan algoritmos matemáticos intensivos repetitivos. Muchas aplicaciones requieren microcontroladores y DSPs trabajando conjuntamente, y los fabricantes han introducido microcontroladores con DSP incorporado. La cosa más básica que un DSP hará es un MACC (Multiplique y Acumule). El número de trozos de los datos que un DSP puede Multiplicar y Acumular determinará el rango dinámico y por consiguiente la aplicación.

Loader o Cargador del Programa Residente
Al arrancar, el microcontrolador carga automáticamente el programa a ejecutar por un puerto serie o paralelo; se puede cambiar el programa las veces que se necesite y desde un lugar remoto; ideal para sistemas distribuidos y para probar nuevos programas y prototipos eliminando el ciclo borrado/grabado típico con EPROMs.

Monitor
Un monitor es un programa instalado previamente en el microcontrolador que permite desarrollos básicos y capacidades de debbuger (depuración de programas).

Suelen incluir:
- Carga archivos objeto en la memoria RAM
- Ejecución de los programas cargados
- Se pueden examinar y modificar registros y memoria
- Desensamblado de código máquina
- Se pueden poner puntos de ruptura
- Ejecución de programas paso a paso
- Los programas monitor pueden comunicarse con una PC, con lo que muchas funciones del programa monitor pueden ser ejecutadas en la PC con el microcontrolador apagado
- Esto simplifica el programa monitor que debe ser cargado en el microcontrolador y que en ocasiones se limita a recibir el programa y ejecutarlo (transmitido desde el PC)

2.4 Los Microcontroladores más Conocidos

La pregunta común es: "¿Qué microcontrolador debo usar?" El mejor consejo es elegir un chip en el que se pueda disponer de todas las herramientas de desarrollo a un precio accesible, y además, buena documentación. A nivel experimental para iniciarse en el estudio de los microcontroladores, el Intel 8051, Motorola 68HC11 o Microchip PIC, son una buena elección.

8048 (Intel)
Fue el primer microcontrolador y aún es usado; tiene un precio bajo, disponibilidad y cuenta con un enorme rango de herramientas de desarrollo.

Tiene arquitectura de Harvard modificada con memoria de programa ROM en chip y con una memoria datos RAM de 64 a 256 bytes.

8051 (Intel)
El 8051, segunda generación de microcontroladores Intel, ha marcado muchas de las características que tienen los microcontroladores en la actualidad:

Es muy potente y sencillo de programar. Su arquitectura es Hardvard con espacio de direcciones separadas para memoria de programa y memoria de datos. La memoria de programa puede llegar hasta 64k, la parte baja (4k ó 8k dependiendo del modelo) está dentro del chip. El 8051 puede direccionar hasta 64k de memoria de datos externa, y sólo se puede acceder a ella mediante direccionamiento indirecto. El 8051 tiene 128 bytes (256 bytes para el 8052) de memoria RAM dentro del chip donde se encuentran:

- Los registros de funciones especiales (SFR = Special Function Registers).
- La entrada y la salida de datos se encuentra mapeada también en este espacio de memoria.

El 8051 es un procesador booleano, pues tiene instrucciones que pueden manejar bits desde cualquier sitio (RAM, acumulador, registros de E/S, etc.) y puede hacer operaciones lógicas con dichos bits y ejecutar saltos relativos basados en dichos resultados.

Existe software comercial y libre, para este microcontrolador.

80c196 (MCS- 96)
Pertenece a la tercera generación de microcontroladores Intel. El 80C196 es un procesador de 16 bits, originalmente fabricado con tecnología NMOS (8096). Y ahora disponible principalmente en CMOS. Intel ha introducido recientemente una versión del doble de velocidad (50 MHz) del 80C196.

Sus características son:

- Multiplicador y divisor
- 6 modos de direccionamiento
- Alta velocidad de E/S
- Convertidor A/D
- Canal de comunicaciones Serie
- Hasta 40 puertos de E/S
- 8 Controladores de interrupción programables
- Modulador de anchura de pulso PWM (para conversión D/A)
- Temporizador Watchdog

80186,80188 (Intel)
Son las versiones en microcontrolador del μP 8086 y del 8088, respectivamente. El chip tiene:

- 2 Canales de DMA (Acceso Directo a Memoria)
- 2 Contadores - Temporizadores
- Interrupciones programables
- Refresco de la RAM dinámica
- Hay versiones de bajo consumo de energía, con puerto serie y más

Una de las mayores ventajas de estos dispositivos es que se pueden utilizar herramientas de desarrollo estándar para PC (Compiladores, ensambladores, etc.). Estos chips tienen la misma arquitectura básica que el 8088 original usado en el IBM PC.

80386 EX (Intel)
El 80386 EX es un μP 386 en microcontrolador, dentro del chip existen:

- Entrada/Salida serie
- Manejo de la alimentación del chip
- DMA
- Contadores - Temporizadores
- Circuito de refresco para memoria DRAM
- Potencia de un 386

Las mayores ventajas de estos dispositivos es que se pueden utilizar herramientas de desarrollo estándar para PC (Compiladores, ensambladores, etc.).

65C02/W65C816S/W65C134S (WDC/Western Design Center)
El Western Design Center Inc. es el diseñador del microprocesador 65C02 de 8 bits que se usó en la computadora Apple, Commodore y Atari WDC; desarrolló también el microprocesador 65C816 de 16 bits.

El W65C816S es un microcontrolador que utiliza un 65C02 y el W65C134S es uno hecho con un 65C816.

MC14500 (Motorola)
- Encapsulado de 16 patas
- Ancho de palabra de 1 bit
- Procesador RISC con un conjunto de 16 instrucciones
- Un solo modo de direccionamiento

68HC05 (Motorola)
Está basado en el μP 6800, tiene arquitectura Von-Neumman donde las instrucciones, datos, entrada/salida y temporizadores ocupan un mismo espacio de memoria. El puntero de pila tiene un ancho de palabra de 5 bits, lo que limita la pila a 32 posiciones, incluyen:

- Convertidor A/D
- Sintetizador PLL
- E/S serie

68HC11 (Motorola y Toshiba)
El 68HC11 es un poderoso microcontrolador de Motorola de 8 bits con las siguientes características:

- Direcciones de 16 bits
- Conjunto de instrucciones similar a la familia 68xx. (6801, 6805, 6809)
- Tiene un único espacio de memoria principal donde están las instrucciones, datos, E/S, y temporizadores

Dependiendo de las versiones puede tener:
- Memoria EEPROM, Memoria OTP ROM
- Memoria RAM
- Entradas/Salidas digitales
- Timers
- Generadores PWM
- Contadores
- Puerto de Comunicaciones síncronas y asíncronas

PIC (Microchip)
Los microcontroladores PIC fueron los primeros microcontroladores RISC. RISC generalmente implica que la simplicidad de diseño permite añadir más características a bajo precio y la línea PIC no es una excepción. Aunque tiene pocas instrucciones (33 instrucciones el 16C5X mientras que el Intel 8048 tiene más de 90), la línea PIC tiene las características siguientes:

- Buses de instrucciones y datos separados (arquitectura Harvard), lo que permite el acceso simultáneo a las instrucciones y a los datos, y el solapamiento de algunas operaciones para incrementar las prestaciones de proceso.

El beneficio de su diseño tan sencillo es que:
- El chip es pequeño
- Pocas patas
- Muy bajo consumo de energía

Los microcontroladores PIC están ganando popularidad debido a su bajo costo, pequeño tamaño y a su bajo consumo de energía.

Existen 5 líneas o gamas:
- Gama Enana: PIC12C(F)XXX. Tienen un tamaño reducido de 8 pines, voltaje de alimentación de entre 2.5 y 5.5 V y consumen menos de 2 mA; cuentan con un oscilador interno. Su frecuencia máxima es de 4 MHz.
- Gama Baja o Básica: PIC16C5X es la línea descendiente del diseño original PIC, está limitado en cuanto a recursos, encapsulado de 18 y 28 patas, alimentación de 2.5 a 5.5 V, no tienen interrupciones y constan con una Pila de 2 niveles.
- Gama Media: PIC 16C(F)XXX. Es la gama más completa y variada de los PIC. Abarca modelos con encapsulados desde 18 hasta 68 pines, un vector de interrupción con 8 niveles de Pila, su frecuencia máxima es de 20 MHz, entre los periféricos que destacan en esta gama son:

  - Comparador Analógico
  - Módulos de Captura (CCP)
  - Modulación de Ancho de Pulsos (PWM)
  - Convertidor Analógico a Digital (CAD)
  - Memoria FLASH y EEPROM
  - Controlador para LCD (Liquid Cristal Display)

- Gama Alta: PIC 17C(F)XXX con instrucciones de 16 bits, interrupciones vectorizadas con 16 niveles de pila, periféricos variados y multiplicador de hardware.
- Gama Mejorada: PIC 18C(F)XXX, arquitectura CISC, 4 vectores de interrupción con 32 niveles de Pila, frecuencia máxima 40 MHz.

COP400 (National Semiconductor)
La familia C0P400 es un microcontrolador de 4 bit P2CMOS que ofrece desde 512 hasta 2K de ROM y desde 32x4 hasta 160x4 de memoria RAM. El encapsulado varía desde 20 hasta 28 patas (DIP/SO/PLCC). Incluyen:

- Interfase de comunicaciones serie Microwire
- Timers
- Contadores
- Tensión de funcionamiento desde 2.3 hasta 6 V
- Soporte OTP

Los microcontroladores de 4 bits tienen un importante mercado y tienen muchas aplicaciones. Estos dispositivos son muy versátiles, hay más de 60 diferentes.

COP800 (National Semiconductor)
La familia COP800 Basic es un microcontrolador de 8 bits totalmente estático, fabricado usando puertos "double metal silicon" de tecnología microCMOS.

Este microcontrolador de bajo costo contiene:
- Temporizadores
- Lógica de Interrupción
- Memoria ROM
- Memoria RAM
- Memoria de E/S mapeada
- Entrada/Salida serie Microwire
- Comunicación UART
- Timers y Contadores de 16 bits
- Interrupciones vectorizadas
- Comparador
- WDT
- Monitor de reloj
- Convertidor A/D de 8 canales
- Protección Brownout
- Modo halt/Idle
- Tensión de alimentación desde 2.5 hasta 6 V

2.5 Lenguajes de Programación para Microcontroladores

Lenguaje ensamblador

El lenguaje máquina es la representación del programa tal como la entiende el microcontrolador. El lenguaje ensamblador es una representación alfanumérica del lenguaje máquina, lo que facilita su lectura. Cada instrucción en lenguaje ensamblador corresponde a una instrucción en código máquina (sin tener en cuenta macros ni directivas). Un programa en lenguaje ensamblador es rápido y corto, esto es porque el programador genera el código más óptimo posible, el programador se adapta al microcontrolador. Programando en ensamblador se aprenderá la arquitectura y estructura del chip.

Intérpretes

Un Intérprete es un lenguaje traductor de alto nivel (próximo al lenguaje natural) a código máquina. El Intérprete está residente en el microcontrolador, ejecuta el programa leyendo cada comando en alto nivel una a una traduciéndolas y ejecutándolas (traduce y ejecuta al mismo tiempo).

Los dos intérpretes más populares que hay para microcontroladores son el BASIC y FORTH.

El BASIC es conocido por su sencillez, legibilidad y por supuesto, porque todo el mundo ha programado en BASIC alguna vez. El BASIC (interpretado) es lento, pero esto puede ser mejorado usando diferentes técnicas.

El FORTH es más rápido que el BASIC (se aproxima al lenguaje ensamblador) y tiene afinidad para construir un sistema con partes reemplazables de software. Algunos sistemas FORTH vienen con un programa monitor que transforma a la PC en un sistema de desarrollo. Puede ser difícil escribir en FORTH, sin embargo, es muy útil y productivo como lenguaje para control de sistemas y para robótica.

Una cosa interesante de los Intérpretes es que se puede construir y desarrollar un programa interactivamente. Se escribe primero un trozo pequeño de programa y a continuación se puede probar para ver inmediatamente cómo funciona. Cuando los resultados son satisfactorios, se pueden agregar entonces las partes adicionales que necesite y así sucesivamente.

Los Compiladores
Un compilador es un lenguaje de alto nivel que combina la programación fácil de un intérprete con una gran velocidad de proceso. Esto se hace traduciendo todo el programa de alto nivel directamente a código máquina. El código máquina se pasa a una memoria EPROM o se carga en la memoria RAM del microcontrolador. El microcontrolador entonces ejecuta el programa traducido directamente, sin haberlo interpretado primero.

Los compiladores más conocidos para microcontroladores son el C, BASIC y el PL/M de Intel.

El BASIC para compilar existe sólo para algunos de los microcontroladores más conocidos. La velocidad de ejecución se incrementa drásticamente.

El FORTH interpretado se acerca a la velocidad de muchos compiladores.

El lenguaje C es ahora el lenguaje más popular por sus características. El C lo usa desde el más pequeño microcontrolador hasta la más potente supercomputadora CRAY, es una herramienta de desarrollo poderosa y flexible. Aunque el C es un lenguaje de alto nivel, permite al programador un acceso a la estructura del microcontrolador, registros, bits, etc. Existen muy buenos y económicos compiladores de C disponibles para los microcontroladores más conocidos. Este lenguaje es ampliamente usado para la programación de microcontroladores, es fácilmente disponible y produce un código eficiente, rápido y compacto.

2.6 Herramientas de Desarrollo

Simuladores
Un simulador ejecuta su programa de microcontrolador en una PC. Se puede ejecutar el programa paso a paso y ver exactamente qué pasa según se ejecuta el programa. Se puede ver y modificar el contenido de los registros, memoria, variables y ver cómo responde el programa.

Se elimina el ciclo borrado/programado, por lo que toma menos tiempo determinar la funcionabilidad de un programa. Se puede aprender experimentando con pequeños trozos de código y observar en pantalla los resultados.

Un simulador no soporta interrupciones reales y funciona a una velocidad mucho más lenta que el microcontrolador simulado.

Debuggers Residentes
Un debbuger residente corre su programa dentro del propio microcontrolador. Al mismo tiempo muestra el progreso de depuración en una máquina host (como por ejemplo una PC). Tiene las mismas características que un simulador normal, con la ventaja adicional de que el programa corre en un microcontrolador real. Un debbuger residente, roba los siguientes recursos al microcontrolador:

- Un puerto de comunicaciones para comunicarse con el host
- Una interrupción para generar programas paso a paso
- Una cierta cantidad de memoria para almacenar el programa residente

Emuladores
Un emulador es un dispositivo sofisticado que sustituye al microcontrolador, al mismo tiempo que está captando información. Nos da total información sobre lo que está pasando en la realidad y no roba ningún recurso a la tarjeta que está analizando.

El emulador puede venir con su propio display o conectado a una PC.

El emulador, es versátil en cuanto a facilidad de simulación en tiempo real del microcontrolador. El emulador es un aparato que se conecta a la PC a través de un cable y en el cual se carga el programa terminado desde la PC; el emulador tiene a su vez una salida de cables planos que terminan en un conector DIP (conector tipo chip), es decir, como si fuera un microcontrolador real, esta terminal se conecta en la placa en la cual se quiere montar el microcontrolador real.

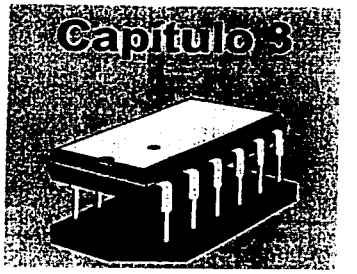
Entonces el emulador funciona con el programa cargado en su interior como si fuese realmente un microcontrolador y con la misma velocidad. El beneficio de esto es que se puede monitorear y controlar todo en el lugar del microcontrolador. Su costo es elevado.

Ensamblador
El ensamblador traduce las instrucciones que se han escrito utilizando los mnemónicos del lenguaje máquina a código ejecutable por el microcontrolador. La secuencia de mnemónicos se llama listado o código fuente.

IDE o Ambiente Integrado de Desarrollo
Un IDE es una herramienta que contiene todo lo que se necesita para programar, editar, depurar, simular, etc. Generalmente son herramientas costosas y es difícil hacer compatible un IDE con otro aunque se trate del mismo chip por lo que frecuentemente conviene usar solamente uno (el más completo) para realizar proyectos de desarrollo.

Microchip, ofrece su IDE denominado MPLAB totalmente gratis y libre, incluye editor, ensamblador, admite ensambladores externos; todo en un ambiente visual bajo Windows.

Comparación entre Ensamblador y un Lenguaje de Alto Nivel
La elaboración de un programa para un microcontrolador se facilita si se usa un lenguaje de alto nivel como el C o el Basic, pues estos pueden utilizar operaciones aritméticas que costaría mucho programar en el ensamblador directamente. Si desensamblamos el código de un programa elaborado en C o Basic, se puede observar que es alrededor de cinco veces más grande que si se hubiera hecho en ensamblador.



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Breve Descripción del PIC 16F874

El PIC 16F874 de Microchip, es uno de los microcontroladores de gama media más poderosos que hay, contiene una gran cantidad de periféricos, modos de comunicación y registros de funciones especiales. La siguiente tabla muestra las características más sobresalientes de este chip:

Tabla 3.1 Resumen de características del Microcontrolador PIC 16F874

Memoria de Programa (FLASH)	Bytes	7168
	Palabras de 14 bits	4 k
Memoria de Datos	Bytes EEPROM	128
	Bytes RAM	192
Frecuencia Máxima	20 MHz	200 ns / ciclo de instrucción
Voltaje de Alimentación	Entre 2 y 6 V y con un consumo medio de 2.5 mA	
Puertos (33 líneas de E/S)	3 de 8 bits	PB, PC, PD
	1 de 6 bits	PA
	1 de 3 bits	PE
Timers	2 de 8 bits	Timer 0, Timer 2
	1 de 16 bits	Timer 1
	1 especiales	Watchdog Timer
Fuentes de Interrupción	14	8 niveles de Pila
Fabricación	Arquitectura	Harvard
	Procesador	RISC
35 Instrucciones	16 Instrucciones que Manejan Registros	
	2 Instrucciones que Manejan Bits	
	4 Instrucciones de Brinco	
	6 Instrucciones que Manejan Operandos Inmediatos	
	7 Instrucciones de Control y Especiales	
Convertidor A/D	8 canales	Resolución de 10 bits
2 Módulos de Captura / Comparación / PWM (CCP)	16 bits de Captura	Precisión máxima 12.5 ns
	16 bits de Comparación	Precisión máxima 200 ns
	PWM	Resolución máxima 10 bits
Comunicación Paralela	1 Puerto Paralelo Esclavo	8 bits además de WR, RD CS
Comunicación Serie	Unidad de Transmisión Recepción Síncrona y Asíncrona	
	Bus Interno de Circuito Integrado	
	Interfase Periférica Serie	
Fuentes Especiales de Reset		Power on Reset (POR)
		Brown out Reset (BOR)
		Watchdog Timer (WDT)
Fuentes Especiales de Retardo		Power up Timer (PWRT)
		Oscillator Start up OST

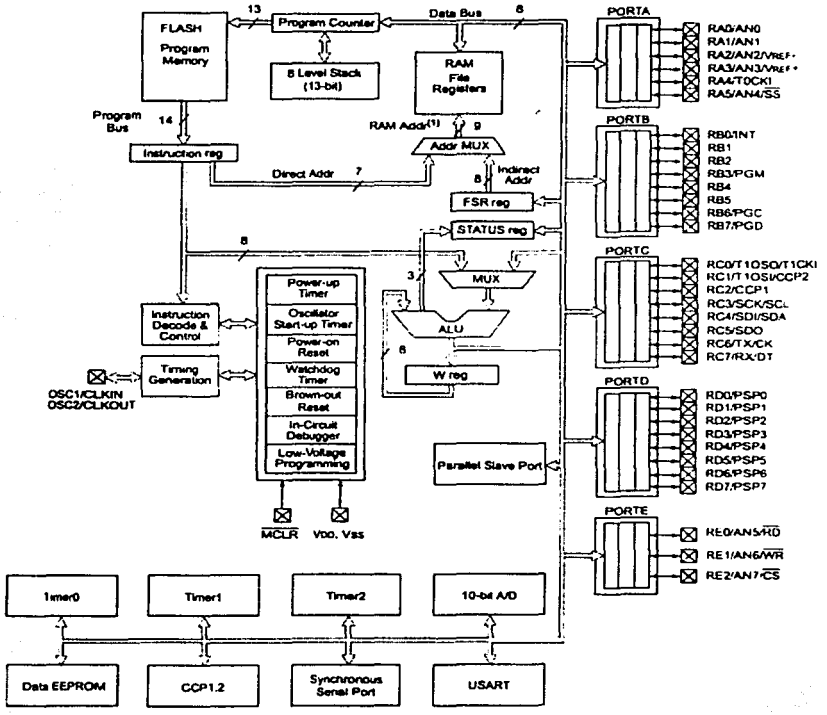
Tabla 3.1 (Continuación) Resumen de características del Microcontrolador PIC 16F874

Otros	Programación Serie en Circuito	
	Código de Protección Programable	
	Modo Sleep de bajo consumo	
Encapsulado		40P, 44L, 44PQ, 44PT

En la figura 3.1 se puede ver la arquitectura interna del microcontrolador, en donde, W es el único acumulador (work acummulator), el registro STATUS o de estado es un registro que sirve principalmente para posicionar al *PC* en alguno de los bancos de la memoria que configuran el hardware y ALU es la unidad lógica aritmética encargada de las operaciones de funcionamiento lógicas básicas.

Fig. 3.1 Arquitectura Interna del PIC 16F874

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

Está fabricado con tecnología CMOS, su arquitectura es Harvard, tiene procesador segmentado lo que permite realizar simultáneamente la lectura y ejecución de una instrucción por ciclo, y su juego de instrucciones RISC, por lo que se deduce que trabaja a altas velocidades de procesamiento en comparación a otros microcontroladores, consume 4 pulsos de reloj por instrucción y el doble en las instrucciones de brinco, esto es, si su frecuencia de operación es de 20 MHz ejecutará 1 instrucción cada 200 ns, en otras palabras, 5 millones de instrucciones por segundo.

3.1 Tipo de Oscilador

El PIC admite cuatro tipos de osciladores externos:

- Tipo RC
  Es un oscilador formado por una resistencia y un capacitor, oscila entre 80 Hz y 700 kHz (Fig. 3.2).
- Tipo LP
  Oscilador de bajo consumo con cristal o resonador diseñado para trabajar en un rango de frecuencias de 35 a 200 kHz.
- Tipo XT
  Es un oscilador de cristal o resonador para frecuencias comprendidas entre 100kHz y 4 MHz.
- Tipo HS
  Es un oscilador que alcanza alta velocidad comprendida entre 4 y 20 MHz.*

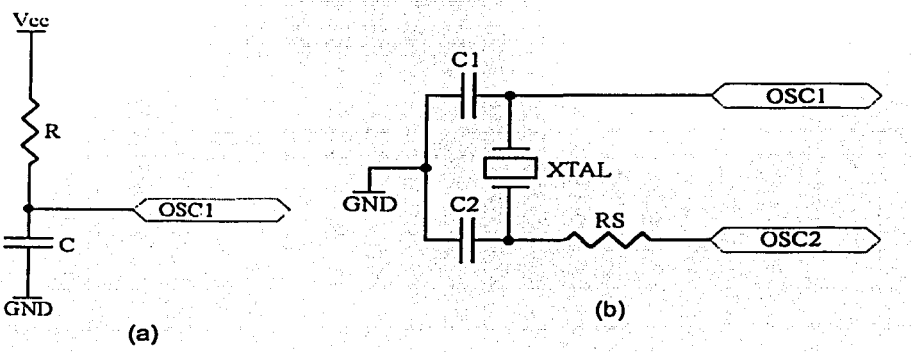


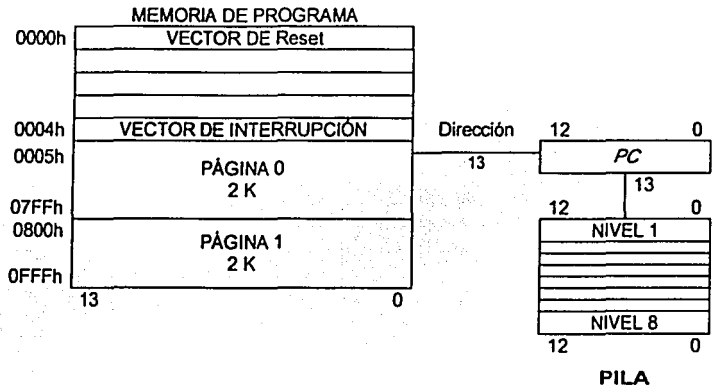
Fig. 3.2 Diagramas esquemáticos de osciladores: a) Oscilador RC b) Oscilador con cristal de cuarzo

* **Nota:** La resistencia RS sólo es instalada en algunos osciladores de alta frecuencia mayores a 10MHz.

3.2 Memoria de Programa

La memoria de programa tiene una capacidad de 4k palabras de 14 bits cada una. Dicha memoria está dividida en páginas de 2k palabras y está direccionada por el *PC* que tiene un tamaño de 13 bits. La Pila tiene 8 niveles, el vector de Reset ocupa la dirección 0000h y el vector de interrupción la 0004h, solo tiene un vector de interrupción.

Fig. 3.3 Organización de la memoria de programa tipo FLASH en el PIC 16F874



3.3 Memoria de Datos

La memoria de datos tiene posiciones implementadas en RAM y otras en EEPROM. En la sección RAM se alojan los registros operativos fundamentales del funcionamiento del procesador y del manejo de sus periféricos, además de registros, que el programador puede usar para manejar información de trabajo propia de la aplicación. La RAM estática consta de 4 bancos con 128 bytes cada uno. En las posiciones iniciales de cada banco se ubican los registros específicos que gobiernan al procesador y sus recursos. La Tabla 3.3 presenta los cuatro bancos de la RAM, indicando en las primeras posiciones de cada uno los nombres de los registros que contienen. Las posiciones en oscuro no están implementadas físicamente y siempre se lee como 0. Los dos primeros bancos contienen 96 bytes cada uno para registros de uso general (GRP) y en los dos últimos se encuentra la copia de los anteriores con el fin de poder accesar a ellos de forma más fácil.

Para seleccionar el banco al que se desea acceder en la RAM se emplean los bits 6 y 5 de Registro Status, denominados RP1 y RP0 respectivamente (Tabla 3.4).

Tabla 3.3 Organización de la RAM del PIC 16F874, con 192 bytes útiles

Banco 0		Banco 1		Banco 2		Banco 3	
Dir. Indirecta	00h	Dir. Indirecta	80h	Dir. Indirecta	100h	Dir. Indirecta	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD	08h	TRISD	88h		108h		188h
PORTE	09h	TRISE	89h		109h		189h
PCLATCH	0Ah	PCLATCH	8Ah	PCLATCH	10Ah	PCLATCH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reservado	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reservado	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h				
T2CON	12h	PR2	92h				
SSPBUF	13h	SSPADD	93h				
SSPCON	14h	SSPSTAT	94h				
CCPR1L	15h		95h				
CCPR1H	16h		96h				
CCP1CON	17h		97h				
RCSTA	18h	TXSTA	98h				
TXREG	19h	SPBRG	99h				
RCREG	1Ah		9Ah				
CCPR2L	1Bh		9Bh				
CCPR2H	1Ch		9Ch				
CCP2CON	1Dh		9Dh				
ADRESH	1Eh	ADRESL	9Eh				
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
Registros de Propósito General 96 bytes	20h	Registros de Propósito General 96 bytes	A0h	Mapeados con 20h-7Fh	120h	Mapeados con A0h-FFh	1A0h
					16Fh		1EFh
					170h		1F0h
	7Fh		FFh		17Fh		1FFh

Nota: Todas las zonas sombreadas no están implementadas en el PIC 16F874 y se leen con el valor de 0

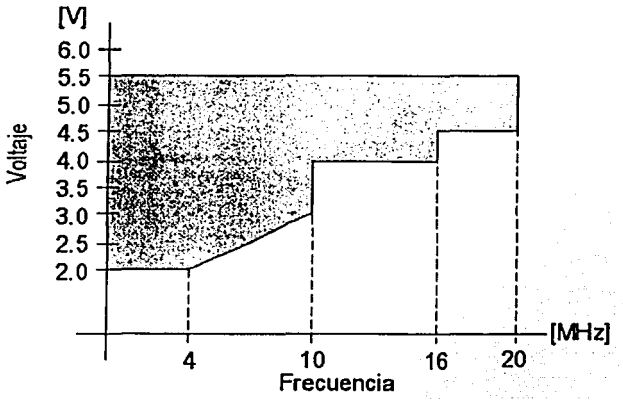
Tabla 3.4
Modo de selección de banco

BANCO	RP1	RP0
0	0	0
1	0	1
2	1	0
3	1	1

3.4 Voltaje de Alimentación

El voltaje de alimentación depende directamente de la frecuencia del oscilador de acuerdo a la siguiente gráfica y de características propias de fabricación que se indican en la nomenclatura del chip: PIC 16X874, si X es F entonces su voltaje de alimentación es entre 4 y 6 V, si X es LF el voltaje de alimentación estará entre 2 y 6 V.

Gráf. 3.1
Voltaje de alimentación en función de la frecuencia del oscilador



3.5 Puertos

Para que un chip de estas dimensiones pueda tener tantas características es necesario que sus pines sean multifuncionales, por lo que, se deduce que para poder utilizar alguno de sus periféricos es necesario sacrificar las otras funciones que tenga el pin. Por ejemplo, el pin RA/AN0: si se quiere usar como canal analógico habrá que configurar el pin como entrada analógica y asociarle el hardware correspondiente, por lo que ya no podrá ser usada como salida digital; en caso contrario el pin podrá ser usado como salida o entrada digital más no como canal analógico .

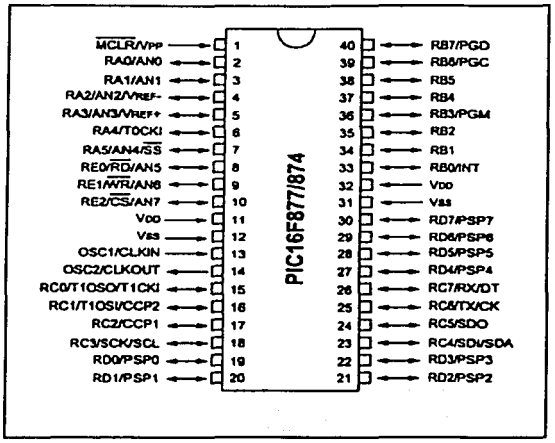


Fig. 3.4
Diagrama de pins del PIC 16F874, en DIP 40

El microcontrolador PIC 16F874 cuenta con cinco puertos de E/S (PA, PB, PC, PD, PE) en tan sólo 40 pines. Todas estas líneas son multifuncionales y bidireccionales, su función depende estrictamente del software.

Puerto A
Dispone de 6 líneas, es un puerto bidireccional, todos los pines de este puerto pueden ser E/S digitales y además:

RA0/AN0	Entrada analógica para el convertidor A/D
RA1/AN1	Entrada analógica para el convertidor A/D
RA2/AN2	Entrada analógica para el convertidor A/D
RA3/AN3/V_{REF}	Entrada analógica para el convertidor A/D o voltaje de referencia positivo
RA4/T0CKI	Entrada de el contador 0.
RA5/AN4/SS#	Entrada analógica para el convertidor A/D y selección del modo esclavo cuando se trabaja en comunicación serie síncrona

Para seleccionar el uso del Puerto A ya sea como E/S digitales o entradas analógicas hay que configurar el registro ADCON1.

Puerto B
Dispone de 8 líneas, es un puerto bidireccional, los pines del Puerto B disponen de una resistencia interna *pull-up* conectadas al positivo de la alimentación y que se habilitan por software. Todos los pines de este puerto pueden ser E/S digitales y además:

RB0/INT	Es la entrada a una interrupción externa
RB3/PGM	Función de programación con voltaje bajo
RB6/PGC	Función de programación con voltaje bajo
RB7/PGD	Función de programación con voltaje bajo
RB4 – RB7	Entrada a interrupción externa

Puerto C

Dispone de 8 líneas, todos los pines disponen de entradas Schmitt Trigger, es un puerto bidireccional, todos los pines de este puerto pueden ser E/S digitales y además:

RC0/T1OSO/T1CKI	Entrada del oscilador para el Timer 1 o entrada del contador 1
RC1/T1OSI/CCP2	Entrada del oscilador para el Timer 1 o entrada para el modo captura 2 o salida del comparador 2 o salida para el módulo PWM 2
RC2/CCP1	Entrada para el modo captura 1 o salida del comparador 1 o salida para el módulo PWM 1
RC3/SCK/SCL	Salida de la señal de reloj en el modo de comunicación SPI o del modo de comunicación I^2C
RC4/SDI/SDA	Salida de dirección en el modo de comunicación SPI o salida de datos y direcciones en modo de comunicación I^2C
RC5/SDO	Salida de datos en el modo de comunicación SPI
RC6/TX/CK	Línea de transmisión del modo USART o señal de reloj síncrona
RC7/RX/DT	Línea de recepción del USART o transmisión de datos en modo serie síncrono

Puerto D

Puerto de 8 bits, todos los pines disponen de entradas Schmitt Trigger, cada pin del puerto puede ser configurado como E/S digital pero también puede ser configurado como puerto paralelo esclavo de 8 líneas.

Puerto E

El Puerto E dispone de tres pines multifunción, que se configuran individualmente, todos los pines de este puerto pueden ser E/S digitales y además:

RE0/RD#/AN5	Señal de lectura en modo de comunicación paralela o entrada analógica 5
RE1/WR#/AN6	Señal de escritura modo de comunicación paralela o entrada analógica 6
RE2/CS#/AN7	Señal de selección de chip en modo de comunicación paralela o entrada analógica 5

3.6 Timers

Los Timers son periféricos que sirven para contar tiempo de algún proceso o retardarlo. Como su nombre lo dice, trabaja con tiempos transcurrido entre oscilaciones.

El PIC 16F874 contiene 3 timers distintos entre sí cada uno con distintas funciones, aunque configurados de forma única, se pueden leer como si no existiera ninguna diferencia entre ellos. Se cuenta con el Timer 0 (TMR0), Timer 1 (TMR1) y Timer 2 (TMR2).

Timer 0

Es un timer de 8 bits, se puede leer directamente su valor en todo instante. Si se parte del registro TMR0 que está en la posición 01 de la RAM, se puede leer su señal directamente o después de pasar por un predivisor (prescalador), este predivisor afecta directamente al registro WDT.

El registro TMR0 se incrementa en una unidad con cada impulso de reloj, cuando alcanza el valor FF vuelve a 00 generando una interrupción, si es que está autorizada y continua su ciclo indefinidamente.

El registro TMR0 se puede leer o escribir directamente con cualquier instrucción, con el fin de conocer su estado actual o para iniciarlo con algún valor predeterminado. El timer deja de funcionar cuando está el microcontrolador en modo *sleep*.

Timer 1

El timer 1 está constituido por un registro de 16 bits dividido en dos registros de 8 bits, TMR1L y TMR1H, los cuales pueden ser escritos y leídos.

El contenido de este registro varía de 0000 a FFFF y vuelve a 0000 después de su desbordamiento y genera una interrupción, si es que ésta se autoriza.

La señal de reloj del temporizador pasa por un predivisor de tasa programable que se puede sincronizar después con el reloj interno. En modo reloj externo se puede seleccionar o no esta sincronización. Si se sincroniza, el Timer 1 deja de funcionar en modo sleep, ya que la sincronización no puede tener lugar debido a la parada del reloj interno. Si no se habilita la sincronización, el temporizador continua contando incluso en modo *sleep*, y puede despertar al microcontrolador y hacerle salir automáticamente de

este modo cuando se desborda el timer, sólo si se ha autorizado la generación de la interrupción.

Timer 2
Está formado por un registro de 8 bits denominado TMR2, asociado a un predivisor y a un postdivisor, así como a un registro de periodo.

El contenido de este registro TMR2 se inicia en el valor 0 y aumenta con cada impulso del reloj de instrucción, después de la eventual división realizada por el predivisor. Cuando su contenido se iguala con el del registro PR2, se pone a 0 y se aplica una señal al postdivisor. La salida de éste puede generar entonces una interrupción, si ha sido autorizada.

WDT o Watchdog Timer
Es un recurso propio de los PIC. El temporizador WDT sirve para evitar cualquier mal funcionamiento. Es un temporizador de 8 bits que cuenta en forma permanente, cuando se desborda hace que se genere un Reset. Tiene un reloj interno propio, inmodificable; se puede variar su frecuencia mediante la configuración de su predivisor.

El WDT se activa o desactiva desde la palabra de configuración, o sea que no es obligatorio su uso.

3.7 Interrupciones en el PIC 16F874

Los PIC 16F874 tienen 14 posibles causas de interrupción. Al darse una interrupción se salva el valor del *PC* en la Pila y se carga aquel con el valor 0004h, que es el vector de interrupción. Las causas probables de interrupción son las siguientes:

1. Desbordamiento del TMR0
2. Desbordamiento del TMR1
3. Desbordamiento del TMR2
4. Activación del pin RB0/INT
5. Cambio de estado en uno de los cuatro pines de más peso de el Puerto B
6. Finalización de la escritura de un byte en la EEPROM
7. Captura o comparación en el módulo CCP1
8. Captura o comparación en el módulo CCP2
9. Transferencia del Puerto Serie Síncrono
10. Colisión de bus en el Puerto Serie Síncrono
11. Fin de transmisión en el USART
12. Fin de recepción en el USART

13. Fin de la Conversión A/D
14. Transferencia del puerto paralelo esclavo

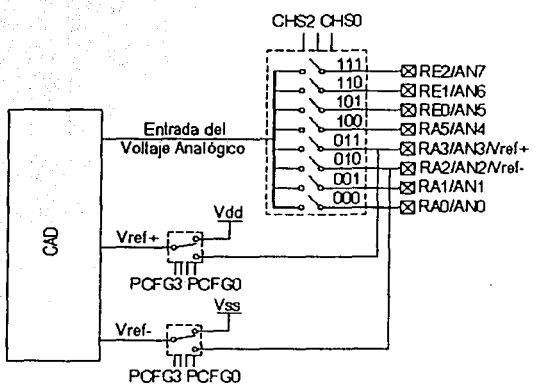
3.8 Convertidor Analógico a Digital (CAD)

Los CAD son periféricos que nos permiten utilizar señales analógicas del mundo real, como temperatura, velocidad y cualquier variable medible, y transformarlos al mundo digital.

El PIC 16F874 dispone de un CAD con una resolución de 10 bits y 8 canales de entrada, los cuales están multiplexados (Fig. 3.5), el número de entradas se puede definir por software por medio de los bits apropiados de registro de control.

Este CAD, puede tomar como voltaje de referencia el de alimentación del chip (5 V), o puede ser alimentado por el pin RA3/AN3/V_{REF} con cualquier otro valor de tensión de referencia, sin pasar de los 5 V, otra consideración importante es que el voltaje en cualquiera de los canales jamás deberá sobrepasar al voltaje de referencia.

Fig. 3.5 Diagrama de bloques del convertidor analógico a digital



El convertidor utiliza una técnica de muestreo y retención que equivale a cargar una capacitancia con la tensión que hay que medir. Por esto, el multiplexor no tiene necesidad de releer permanentemente la entrada seleccionada como entrada del convertidor propiamente dicho, sino solamente durante el tiempo suficiente para cargar al condensador. Este tiempo, denominado T_{AD}, depende de la velocidad de conversión deseada, y de la frecuencia de reloj del PIC.

Tabla 3.5 Frecuencias y tiempos seleccionables para conversión analógico a digital

ADCS1	ADCS0	F_{AD}	T_{AD}
0	0	Fosc/2	2Tosc
0	1	Fosc/8	8Tosc
1	0	Fosc/32	32Tosc
1	1	Reloj Interno	Reloj Interno

Es posible programar una conversión mientras que el PIC está en modo *sleep*, cuando la conversión termina, se genera una interrupción si estaba autorizada y sale del modo *sleep*.

3.9 Módulos de Captura/Comparación/PWM (CCPx)

Puede capturar algún evento, también comparar un valor, ó por último, pueden generar modulación por ancho de pulso.

El PIC 16F874 dispone de dos módulos de captura, comparación y PWM. Cada módulo está formado por un registro de 16 bits, accesible tanto para escritura como para lectura, bajo la forma de dos registros de 8 bits concatenados.

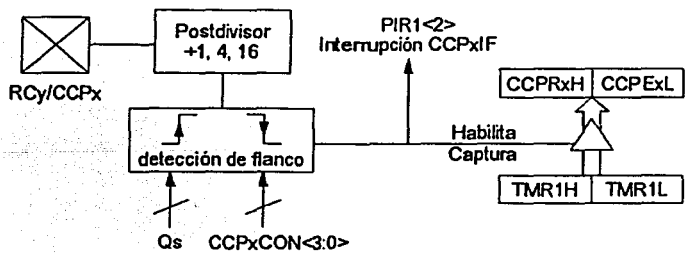
Modo Captura

En este modo captura el contenido del timer 1 cuando se produce una de las siguientes condiciones:

- Un flanco de bajada
- Un flanco de subida
- Una captura cada cuatro flancos de subida
- Una captura cada dieciséis flancos de subida

Cuando tiene lugar una captura, se puede generar una interrupción. Si se produce otra captura antes de que el contenido de CCPRx se haya leído, este contenido se sustituye por el nuevo.

Fig. 3.6 Diagrama de operación del CCPx en modo captura



Modo Comparación

En este modo el contenido del registro CCPx se compara permanentemente con el contenido del timer 1. Cuando se produce una igualdad, cualquiera de los siguientes fenómenos se puede producir:

- Paso a nivel alto del pin RCx/CCPx
- Paso a nivel bajo del pin RCx/CCPx
- Ninguna variación de estado en el pin, pero se genera una interrupción si el módulo CCPx está autorizado

La siguiente figura muestra de forma muy clara el principio de funcionamiento de este modo de comparación.

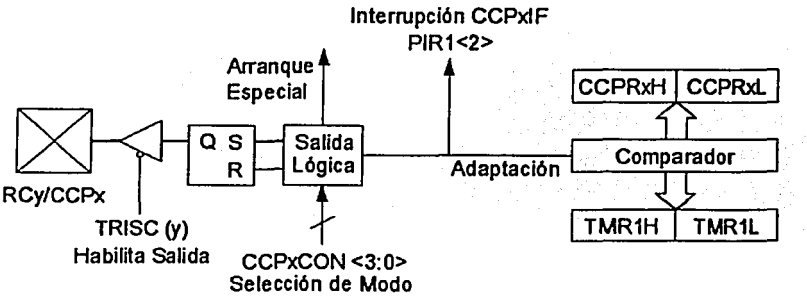


Fig. 3.7 Diagrama de operación del CCPx en modo comparación

Modo PWM

En este modo, el pin RCx/CCPx permite disponer de una salida que es una señal de ancho de pulsos modulados, con una resolución que puede llegar hasta 10 bits. Este es un tipo de convertidor digital analógico; lo que se tiene a la salida es una onda cuadrada que conmuta entre el valor de 0 y 1 lógico de periodo constante más no forzosamente simétrica, donde el valor del voltaje resultante viene dado por la relación:

$$V = 5 * \frac{\text{Ciclo de Trabajo}}{\text{Periodo}}$$

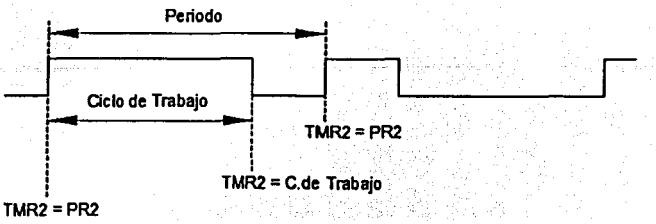


Fig. 3.8 Representación de modulación de ancho de pulso

Todo el tiempo que permanece el valor del PWM en valor lógico 1 se le conoce como ciclo de trabajo (Fig. 3.8). En el PIC el valor del periodo se carga en el registro PRx, y el valor del ciclo de trabajo está en función del valor cargado en el registro TMR2 y los valores escritos en CCPRxL y CCPRxH, que al comparar estos dos últimos con el timer 2 y coincidir generan la señal PWM en los pines RC1 y/o RC2. La siguiente figura muestra el diagrama de funcionamiento de el módulo PWM.

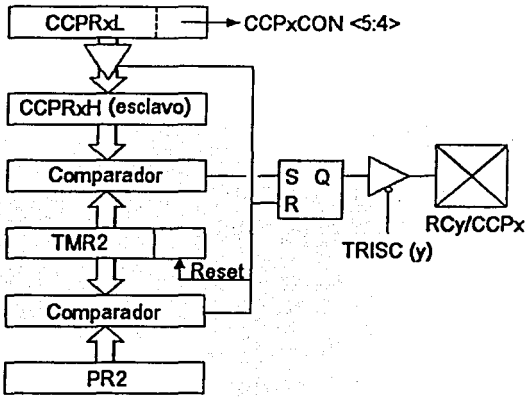
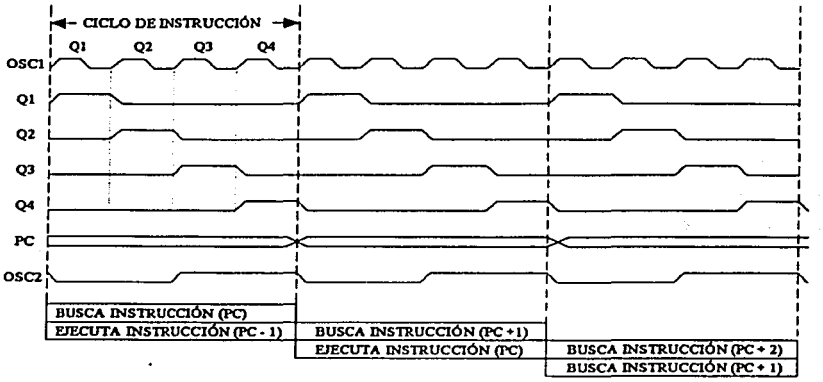


Fig. 3.9 Diagrama de operación del CCPx en modo PWM

3.10 Instrucciones

Los PICs de las gamas baja, media y alta están fabricados bajo una arquitectura RISC, mientras que los últimos PIC a los que se les denomina de gama mejorada ya incluyen procesador con arquitectura CISC, pero sigue habiendo compatibilidad con las instrucciones RISC, esto es, si se realiza un programa usando las instrucciones del PIC 16F874 en un PIC 18C858 y respetando el mapa de memoria, el programa forzosamente funcionará.

El PIC 16F874 consta con un conjunto de 35 instrucciones, en el anexo A se muestra una tabla con el condensado de las instrucciones. Los pulsos del oscilador entran por el pin OSC1/CLKIN y se divide entre 4 internamente dando lugar a las señales Q1, Q2, Q3 y Q4 (Fig. 3.10), durante un ciclo de instrucción, que comprende las 4 señales mencionadas.



Se desarrollan las siguientes operaciones:

- Q1: Durante este pulso se incrementa el contador
- Q2 – Q3: Se produce la decodificación y la ejecución de la instrucción
- Q4: Se busca el código de la instrucción en la memoria del programa y se carga en el registro de instrucción

Para conseguir ejecutar una instrucción en un ciclo de instrucción se aplica la técnica de segmentación o *pipe – line,* que consiste en realizar en paralelo las dos fases que comprende cada instrucción. Cuando la instrucción ejecutada corresponde a un salto no se conoce cuál será la siguiente hasta que se complete, por eso de esta situación se sustituye la fase de búsqueda de la siguiente instrucción por un ciclo vacío, originando que las instrucciones de salto tarden en realizarse dos ciclos de instrucción. (Fig. 3.10.)

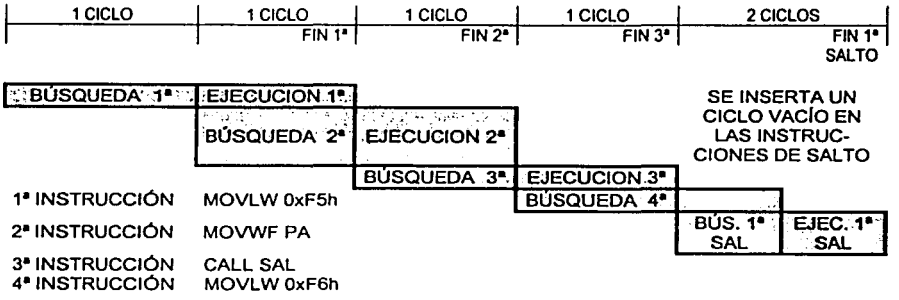


Fig.3.11 La segmentación permite solapar en el mismo ciclo la fase de ejecución de una instrucción y la de búsqueda de la siguiente, excepto en las instrucciones de salto

3.11 Modos de Direccionamiento

Direccionamiento Inmediato

Es aquel en el que el dato manipulado por la instrucción se codifica junto con la propia instrucción. En este caso, el dato se denomina literal (l).

Ejemplo:
movlw 0xFF
La instrucción *movlw* 0xFF, coloca el literal FF, que es un valor de 8 bits, en el registro W.

Direccionamiento Directo
Éste es el modo más utilizado ya que la memoria RAM está dividida en registros específicos y en un conjunto de registros de propósito general. Este modo consiste en codificar el nombre del o de los registros en cuestión directamente en la instrucción.

Ejemplo:
movwf Reg
Mueve el contenido del registro W al registro Reg.

Direccionamiento Tipo Bit
Permite manipular un bit individual de un registro.

Ejemplo
bcf Reg, 7
Pone a 0 el bit número 7 del registro Reg.

Direccionamiento Indirecto
Se introduce en el FSR el número del registro direccionado.

Ejemplo
movwf INDF
Esta instrucción desplaza el contenido del registro W al registro apuntado por el FSR. La notación INDF únicamente sirve para indicar direccionamiento indirecto, por tanto la utilización del registro FSR como puntero (previamente cargado con el valor de dirección).

Direccionamiento Relativo
Permite los saltos como son los ciclos condicionales. Pero no existe este modo en los PICs de gama media.

3.12 Comparación del PIC 16F874 con otros PICs de la misma Gama

Se eligió el PIC 16F874 por sus múltiples características, es un microcontrolador de lo más completo, versátil y rápido, además se encuentra dentro de un rango de precios accesible, mas no es el más poderoso de la gama.

En la siguiente tabla se muestra algunos PICs de la gama media con los que se puede comparar:

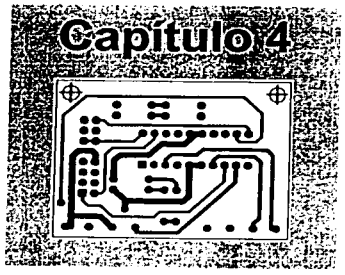
Tabla 3.6 Tabla comparativa de microcontroladores PIC de la misma gama

Características	16F84	16C74	16F874	16F877	16C924
Pines	18	40	40	40	64
Timers	1	3	3	3	3
Interrupciones	4	12	14	14	14
Comunicación	-	PSP, USART, SSP (SPI, I²C Slave)	PSP, USART, SSP (SPI, I²C Master/Slave)	PSP, USART, SSP (SPI, I²C Master/Slave)	SSP (SPI, I²C Master/Slave)
Frecuencia Máx.	10 MHz	20 MHz	20 MHz	20 MHz	20 MHz
Voltaje	2.5 – 5.5 V	2.0 – 5.5 V	2.0 – 5.5 V	2.0 – 5.5 V	2.0 – 5.5 V
A/D	-	8 bits, 4 canales	10 bits, 8 canales	10 bits, 8 canales	5 bits, 1 canal
CCP	-	2	2	2	2
Memoria de Programa	1 k (FLASH)	4 k (EPROM)	4 k (FLASH)	8 k (FLASH)	4 k (EPROM)
RAM	68	192	192	368	176
EEPROM	64	-	128	256	-
Instrucciones	35	35	35	35	35
Otras	-	-	Depuración en circuito, Programación en bajo voltaje	Depuración en circuito, Programación en bajo voltaje	Control de LCD, Multiplexor

La selección de los PICs de la tabla anterior no es por azar, en ella se encuentran los PICs más representativos de la gama y un poco de la historia del 16F874. El PIC 16F84 dentro de su reducido tamaño tiene características interesantes pero carece de periféricos, mientras que el PIC 16C74 es muy atractivo por sus periféricos y su capacidad de memoria de programación, mas tiene una limitación importante: su memoria de programa es EPROM, lo que quiere decir que sólo puede ser programada una sola vez. Existen PICs 16C74 con UVPROM, pero aún así el tiempo de borrado es incómodo. Al fusionar las mejores características de ambos PICs (16F84 y 16C74) nace el PIC 16F874 que contiene todas las características y periféricos de ambos y algunas extras.

El PIC 16F877 es hijo del PIC 16F874, la única diferencia entre ambos es la capacidad, de hecho, todo el desarrollo de una tarjeta para cualquiera de los dos PICs sirve totalmente para ambos, lo único que cambiaría es el software y solamente por la cantidad de registros de uso común que se pueden utilizar en una o en otra, pero todos los demás registros especiales son los mismos y están localizados en la misma dirección y banco. Estas características son las que vuelven la tarjeta desarrollada aún más versátil (ver Capítulo 4).

Por ultimo, el PIC 16C924 es un ejemplo de PIC de gama media que tiene usos específicos como son el control de display, sus registros cuentan ya con tablas de caracteres a los que se puede accesar por software y desplegar por LCD. Existen PICs con otros usos especiales como generador de señales, control de teclado y otros.



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Diseño Electrónico

4.1 Descripción de la Tarjeta de Desarrollo PIC 16F874

Esta tarjeta fue desarrollada con la intención de incursionar en una nueva gama de microcontroladores los cuales son de fabricación reciente, poderosos y sumamente utilizados.

Como se puede ver en el Capítulo 3, el microcontrolador PIC 16F874 de Microchip cuenta con una gran cantidad de periféricos como son las entradas y salidas digitales, convertidor analógico a digital, módulos PWM, etc., por consiguiente se antoja la construcción de una tarjeta con este microcontrolador que además tenga la mayor cantidad de periféricos disponibles, esto es, al ser un chip de apenas 40 pines éstos son multifuncionales y al utilizar alguna de sus funciones quedan deshabilitadas otras, así que la construcción de una tarjeta implica el sacrificio de algunas de las características del microcontrolador.

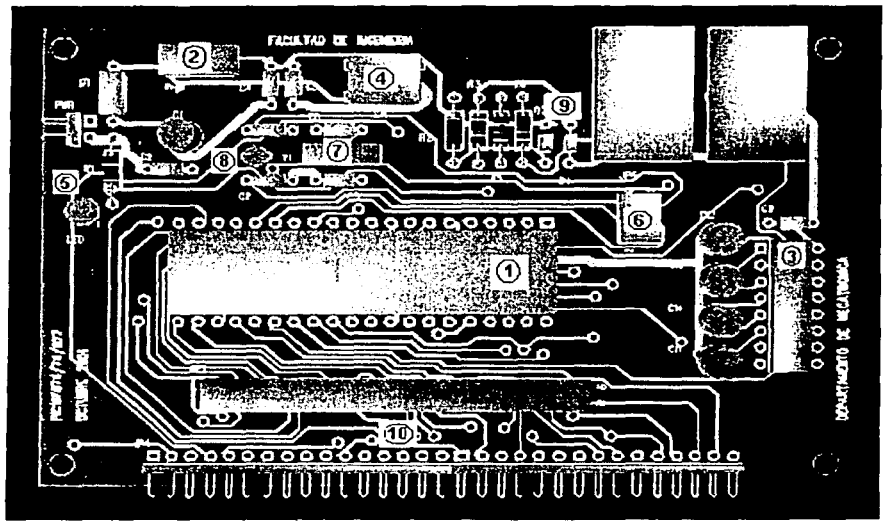


Fig. 4.1.1
Tarjeta de desarrollo PIC 16F874

La tarjeta de desarrollo PIC 16F874 (Fig. 4.1.1), cuenta con estas características:

1. Un socket para DIP 40
2. Regulador de voltaje de entrada de corriente directa que transforma a 5 V
3. Hardware de comunicación RS – 232 para PC
4. Botón pulsador para Reset
5. LED para indicación de encendido

6. Jumper para selector de función del contador 1 u oscilador 1 para Timer1
7. Oscilador de cuarzo de 8 MHz
8. Oscilador de cuarzo de 32 kHz
9. Programador serial integrado y socket de interfaz con PC
10. 29 pines de pines bidireccionales:
    a. 5 puertos: PA, PB, PC, PD y PE
    b. 29 de E/S digital
    c. 8 entradas analógicas

Esta tarjeta está diseñada de forma tal que se pueda desarrollar con ella una gran cantidad de proyectos sin realizar modificaciones al hardware. Su diagrama esquemático y el trazado de las pistas se pueden ver en el Apéndice B.

4.1.1 Alimentación

La alimentación del microcontrolador se realiza a través de un circuito de regulación de voltaje, en el cual se utiliza un regulador LM7805, que transforma el voltaje de entrada (6 v > Vin > 12), a un voltaje TTL de 5 V (Fig. 4.1.2).

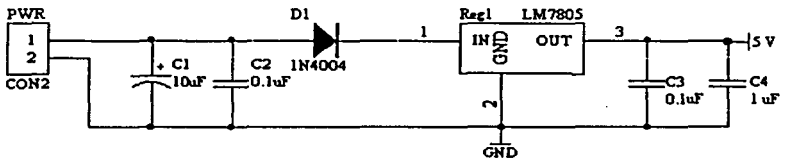


Fig. 4.1.2 Circuito de regulación de voltaje implementado en la tarjeta

4.1.2 Comunicación RS – 232

Para lograr la comunicación con otros dispositivos inteligentes como puede ser una computadora, se eligió la norma de comunicación RS – 232, ya que es una de las más extendidas. El chip MAX232 de MAXIM dispone de dos canales de entrada para niveles TTL y dos de salida, igualmente posee dos canales de entrada RS – 232 y dos de salida; y la alimentación del chip es a 5 V. La figura 4.1.3 muestra la conexión esquemática de los componentes necesarios para lograr la interfaz RS – 232.

El cable necesario para realizar la comunicación entre la PC y el microcontrolador vía RS - 232, es un cable de 4 hilos, como el telefónico estándar, con un plug telefónico de 4 entradas y un DB9 hembra, Fig. 4.1.4.

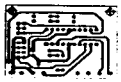


Fig. 4.1.3
Circuito de comunicación estándar RS - 232

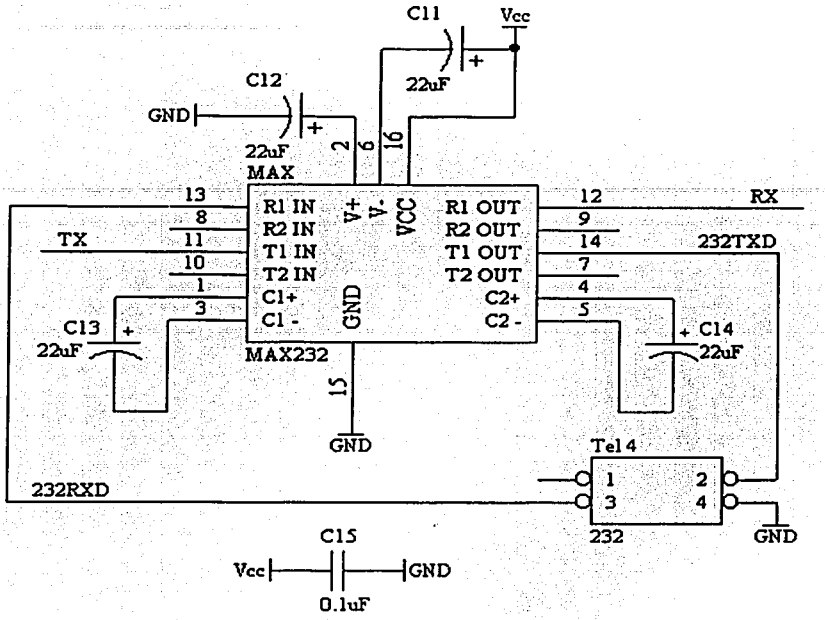
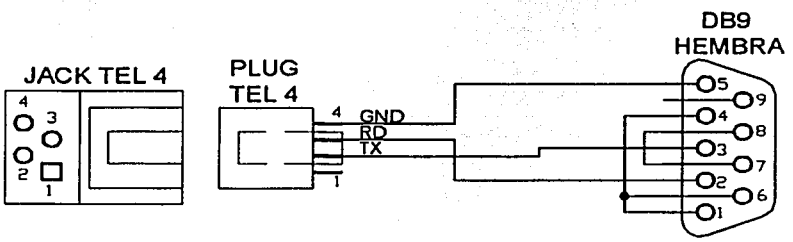


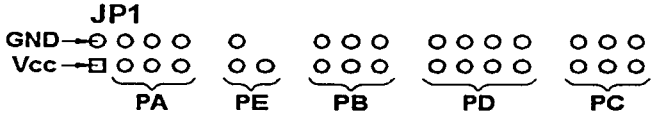
Fig. 4.1.4
armado de Cable para Interfaz RS-232



4.1.3 Headers

La tarjeta cuenta con dos headers principales para la utilización de los puertos, el JP1 y el JP4. El JP1 es un header doble para aplicaciones finales o aquellas en donde se requiera cable plano. La distribución de los puertos sobre el mismo se realizó en función de una simplificación de pistas; la distribución es la siguiente:

Fig. 4.1.5
Distribución de puertos en el JP1



El propósito de la separación existente entre cada puerto es por si la aplicación solamente requiere utilizar uno u otro puerto, puede hacerlo mediante un conector para cable plano de tamaño pequeño, mas, si la aplicación requiere de varios puertos se puede hacer un cable con un conector único de 2x20. En los dos primeros pines del JP1 se encuentra un voltaje de alimentación TTL para alimentar a dispositivos externos.

El JP4 es un header horizontal sencillo con el cual la tarjeta puede ser insertada en un protoboard para realizar el desarrollo de la aplicación de manera temporal y rápida, su distribución es la que se muestra (Fig. 4.1.6). En esta figura también se puede apreciar que el voltaje de alimentación externo TTL se encuentra en la primera y la ultima líneas del header.

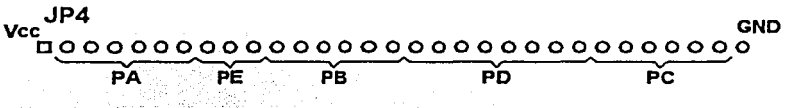
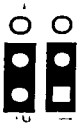
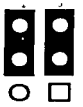


Fig. 4.1.6 Distribución de puertos en el header JP4

El JP6 es un header de 2x3 que está acoplado al cristal de 32 kHz, a los pines 16 y 17 del DIP 40 y a los headers del Puerto C. De esta manera y dependiendo de la posición de los jumpers en él, se habilita una u otra función del Puerto C según la tabla 4.1.1:

Tabla 4.1.1 Selector de función del JP6

Posición	Habilitación de
	- El bit 0 del Puerto C puede funcionar como entrada o salida digital o como entrada del contador 1. - El bit 0 del Puerto C puede funcionar como entrada o salida digital o módulo CCP2 (ver Capítulo 3).
	- El bit 0 y el bit 1 se conectan al cristal de 32 kHz.

4.1.4 Tamaño

Otra característica destacable de la tarjeta es su tamaño, muchas veces el tamaño del proyecto está en función del tamaño de la tarjeta del microcontrolador como en los AGVs, esta tarjeta es apenas más grande que un disco de computadora de 3½" y sólo a

lo ancho, así pues su transportación es más sencilla y la ubicación en algún proyecto también. Sus dimensiones son 10.7 x 7.1 cm (Apéndice B).

4.1.5 Reset

El botón pulsador del Reset está conectado a tierra y por el otro al voltaje de alimentación. Al presionar el botón se genera un corto por lo cual el voltaje cae a cero; las resistencias acopladas al botón sirven para disipar el calor generado por el corto y el diodo nos sirve para proteger a los demás componentes (Fig. 4.1.7).

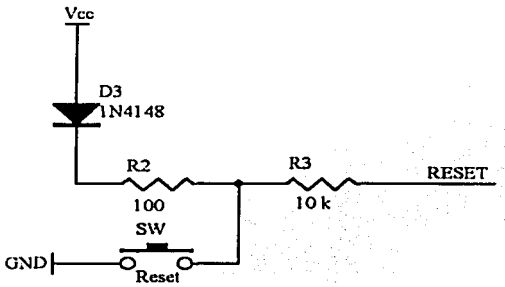


Fig. 4.1.7
Configuración del botón del Reset

4.1.6 LED

La función del LED en la tarjeta es puramente ergonómica, ya que es una señal visual que nos indica cuando la tarjeta está alimentada o no, en el caso que se alimente mal o al revés éste no encenderá. Está conectado directamente a la salida del regulador, así que, si el regulador no funciona el LED tampoco encenderá. Este LED también encenderá en el momento en que se le esté cargando o leyendo un programa al microcontrolador.

4.1.7 Osciladores

El oscilador de cuarzo de 8 MHz es el que le da vida al microcontrolador, todas las instrucciones serán gobernadas por la velocidad del mismo. Este cristal puede ser sustituido por algún otro cristal que se encuentre en el rango de funcionamiento permitido (ver Capítulo 3). La Figura 4.1.8a muestra su diagrama de instalación.

El oscilador de cuarzo de 32 kHz es independiente del funcionamiento del microcontrolador, su entrada a éste se realiza mediante la correcta configuración del software y selección en el

hardware mediante el JP6; ya conectado al microcontrolador, éste genera los pulsos que incrementan al Timer 1, el único timer de 16 bits (ver Capítulo 3). Se eligió este oscilador ya que es fácil programarlo para llevar cuentas en tiempo real (segundos, minutos,...), (Fig. 4.1.8b).

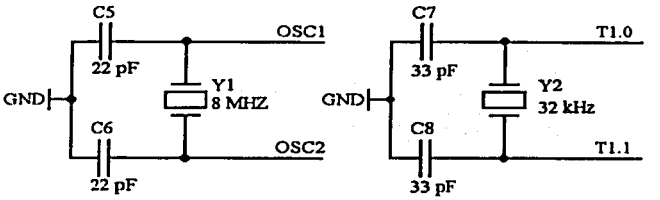


Fig. 4.1.8 Circuitos de los osciladores:
a) Cristal de 8 MHz
b) Cristal de 32 kHz

4.1.8 Puertos

Los microcontroladores compatibles con la tarjeta cuentan con 33 líneas de entrada y salida, aunque, esta tarjeta cuenta tan sólo con 29, ya que las líneas faltantes se utilizan en otras aplicaciones dentro de la tarjeta como son el MAX232 y el programador.

Descripción de los puertos:

- El puerto A de 5 pines, el puerto E de 3 líneas y el puerto D de 8 líneas, conservan todas sus funciones.
- El puerto B perdió los bits 6 y 7, ya que éstos están conectados al programador de la tarjeta, por lo que el puerto consta de 6 líneas. Los demás pines conservan sus funciones intactas.
- El puerto C utiliza los pines 6 y 7 para la interfaz RS232, los bits 0 y 1 pueden ser desconectados mediante la selección en el JP6, por lo que el puerto puede tener 4 ó 6 líneas utilizables. Los demás pines conservan todas sus funciones.

Para mayor descripción de los puertos véase el Capítulo 3.

4.1.9 Programador

La tarjeta incluye un programador con el cual se puede realizar interfaz entre una computadora y el microcontrolador, de esta manera la memoria de programa se puede leer, borrar, cambiar y reconfigurar. El chip se programa de forma serial donde el pin 39 del DIP 40 es la entrada del reloj y el pin 40 es la entrada de datos.

Para programar al microcontrolador se necesita un voltaje de alimentación de 12 V y otro de 5 V, el voltaje de 12 V es proporcionado por el mismo puerto serie de la computadora el cual llega a la pata 1 del DIP 40; el de 5 V es tomado de la salida del

regulador de la tarjeta y en el DIP 40 se puede medir en los pines 11 y 32. La referencia de tierra está conectada tanto a la computadora como a la tarjeta gracias a su cable de interfaz (Fig. 4.1.9), las patas que indican esta referencia son la 12 y la 31.

Para verificar que las señales que envía el software de programación son las correctas y llegan a estos pines se puede realizar una prueba de hardware en la que se ponen a 1 ó a 0 en estos pines según se le indique a la PC.

Como el puerto serie tiene una corriente baja a la hora de leer o programar el microcontrolador, es necesario que ningún elemento externo esté conectado a la alimentación TTL que proporciona la tarjeta, además que la tarjeta no esté siendo alimentada por ninguna otra fuente de voltaje externa y preferentemente que no esté conectado tampoco el cable de comunicación RS - 232.

El programador es muy sencillo, consta de un diodo, tres resistencias y un regulador; se muestra en la Figura 4.1.9.

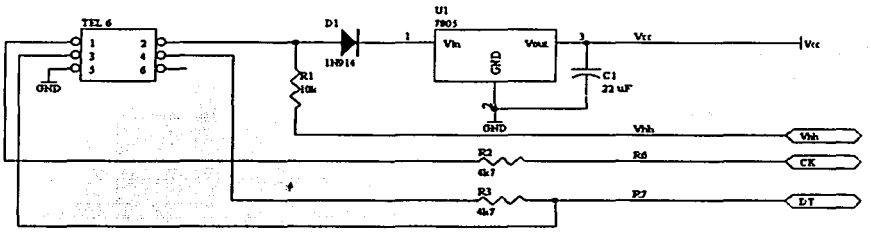


Fig. 4.1.9 Programador universal serial para microcontroladores PIC y memorias seriales

El cable de programación es de 5 hilos, un extremo está conectado a un plug DB9 hembra y el otro extremo a un plug telefónico de 6 hilos, por lo que la tarjeta consta de un jack telefónico de seis hilos (Fig. 4.1.10).

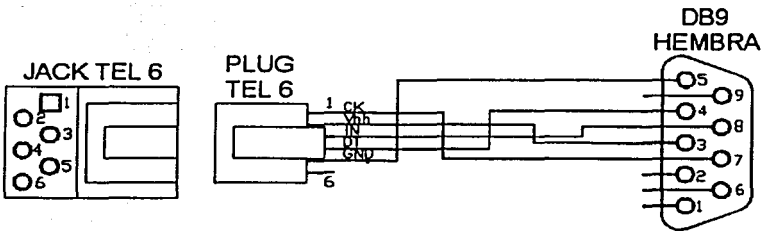
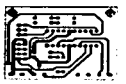


Fig. 4.1.10 Armado de cable de interfaz de la Tarjeta - PC, para el programador



4.1.10 Compatibilidad

Algunos de los microcontroladores de 40 patas de Microchip tienen compatibilidad respecto a la distribución de sus pines aunque no cuentan con los mismos recursos así que la tarjeta los soporta.

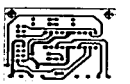
Los microcontroladores que soporta la tarjeta sin realizar ningún cambio en la estructura del hardware son los que se muestran en la Tabla 4.1.2. Como se puede ver, la manera más sencilla de realizar una expansión es cambiando de chip. También cabe destacar que todos los microcontroladores de la tabla, a excepción del PIC 18C458, son de la gama media y este último, es de la gama mejorada, lo que significa que cambia en la arquitectura interna del microcontrolador como es el procesador CISC; además, la frecuencia máxima del oscilador que soporta es de 10 MHz, aunque su frecuencia de trabajo es multiplicada por cuatro gracias a un PLL interno integrado; así que su frecuencia máxima de trabajo es de 40 MHz.

La siguiente tabla muestra algunos de los microcontroladores PIC que son compatibles con la tarjeta:

Tabla 4.1.2 Microcontroladores PIC compatibles con la tarjeta

Características	16C74/77	16F874	16F877	18F448/458
Pines	40	40	40	40
Timers	3	3	3	4
Interrupciones	12	14	14	32
Comunicación	PSP, USART, SSP (SPI, I²C Slave)	PSP, USART, SSP (SPI, I²C Master/Slave)	PSP, USART, SSP (SPI, I²C Master/Slave)	PSP, USART, SSP (SPI, I²C Master/Slave)
Frecuencia Máx.	20 MHz	20 MHz	20 MHz	40 MHz
Voltaje	2.0 – 5.5 V	2.0 – 5.5 V	2.0 – 5.5 V	2.0 – 5.5 V
A/D	8 bits, 4 canales	10 bits, 8 canales	10 bits, 8 canales	10 bits, 8 canales
CCP	2	2	2	4
Memoria de Programa	4 k (EPROM)	4 k (FLASH)	8 k (FLASH)	32 k (FLASH)
RAM	192	192	368	1.5 k
EEPROM	-	128	256	256
Instrucciones	35	35	35	75
Otras	-	Depuración en circuito, Programación en bajo voltaje	Depuración en circuito, Programación en bajo voltaje	Depuración en circuito, Programación en bajo voltaje, oscilador de 32 kHz integrado

Sus diagrama de pines o patigramas se pueden ver en el Apéndice E



4.1.11 Expansión de los Recursos de la Tarjeta

Memoria de Programa

La expansión de la memoria de programa, sin modificar las características de la tarjeta, se puede realizar mediante las siguientes formas:

1. Cambiando el microcontrolador con otro de estructura compatible en pines y de mayor capacidad.
2. Creando un BIOS (loader) que pueda cargar el programa de una memoria externa o de otro microcontrolador.
3. Crear una red multimaestra de microcontroladores y asignar una parte del programa a cada uno de los microcontroladores maestros.

Memoria de Datos

La expansión de la memoria de datos también es sencilla de realizarse gracias a las características del microcontrolador. Se puede realizar de cualquiera de las siguientes formas:

1. Cambiando el microcontrolador con otro de estructura compatible en pines y de mayor capacidad.
2. Utilizando el puerto esclavo paralelo, añadiéndole la circuitería y el programa de control pertinente para leer memorias externas de forma paralela.
3. Utilizando los protocolos de comunicación SPI e I²C que contiene el microcontrolador. Esta expansión soporta 128 memorias seriales.
4. Crear una red multimaestra de microcontroladores y compartir los recursos de cada uno de los microcontroladores maestros.

Puertos

La expansión de puertos se puede realizar mediante la conexión de uno o varios controladores de interfaz paralela 8255; con lo que se sacrificarían algunos de los pines del microcontrolador para ganar más puertos paralelos bidireccionales. El 8255 cuenta con tres puertos paralelos bidireccionales y configurables por software.

La Figura 4.1.11 muestra el diagrama esquemático propuesto para una expansión de puertos sencilla, en la que los pines del microcontrolador direccionan y habilitan directamente a la interfaz 8255.

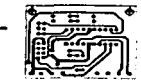
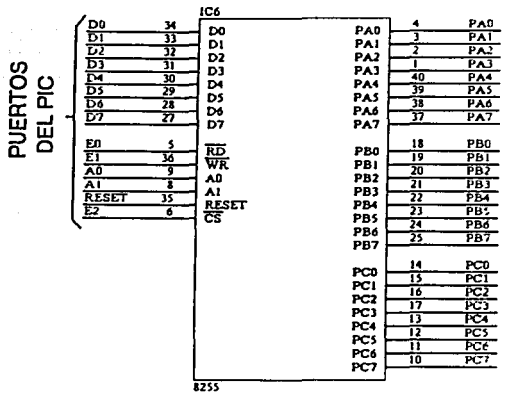
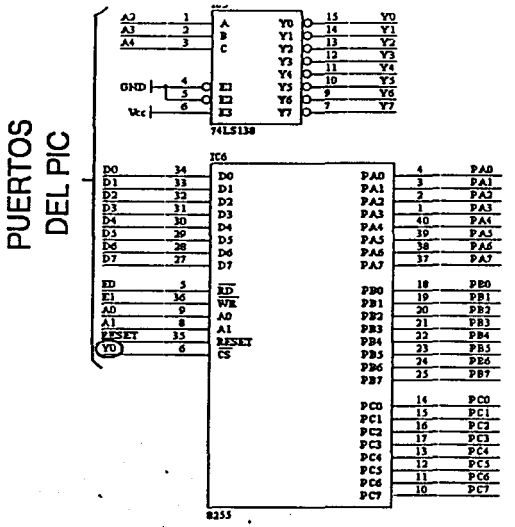


Fig. 4.1.11 Propuesta de expansión simple usando un chip de interfaz paralela 8255 y el puerto esclavo paralelo de la tarjeta



La Figura 4.1.12 muestra el diagrama esquemático propuesto para una expansión de varios puertos paralelos 8255 implementando un decodificador, el cual habilita el 8255 con el que se está trabajando, esta figura también muestra un decodificador 74LS138 el cual sólo tiene tres entradas lo que significa que puede seleccionar 8 chips 8255 ($2^3 = 8$); esto significa que la expansión máxima, usando este chip, será de 24 puertos paralelos bidireccionales de 8 bits cada uno, 192 pines de entrada y salida. El 74LS138 puede ser sustituido por otro decodificador con más líneas de entrada para aumentar la expansión de puertos.

Fig. 4.1.12 Expansión de Puertos utilizando el chip de interfaz paralela 8255 y habilitando por decodificador



Otra forma de lograr hacer una expansión de puertos es usando los protocolos de comunicación SPI e I²C, en los cuales se conectan los controladores de interfaz paralela. Esta expansión puede ser mayor que la anterior en cuanto a la cantidad de dispositivos que se pueden agregar, los cuales no podrán ser más de 128. La comunicación en este tipo de expansión es un poco más lenta que la comunicación en paralelo (entre 100 y 400 kbs).

La Figura 4.1.13. muestra las conexiones necesarias para conectar una interfaz de expansión de puertos paralelos con el chip PCF8574 y una expansión de 2k en memoria EEPROM con el chip PCF85116-3, en un protocolo de comunicación I²C.

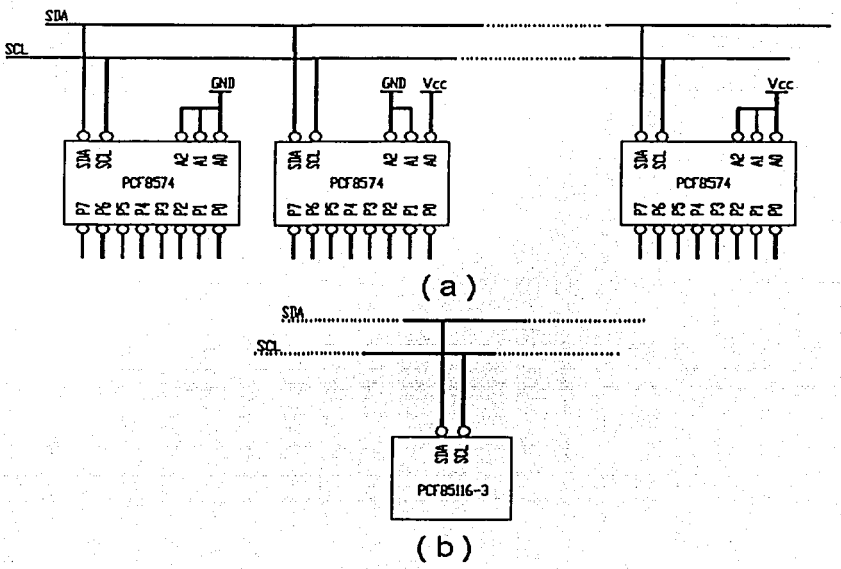
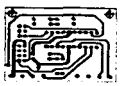


Fig. 4.1.13
Expansión I²C
a) Expansión de puertos paralelos
b) Expansión de memoria EEPROM



4.2 Diseño de Circuitos para Aplicaciones Mecatrónicas

La electrónica en un sistema mecatrónico, a grandes rasgos, se divide en las siguientes partes:

- La circuitería para el funcionamiento de la UCP
- Electrónica de potencia
- Interruptores, sensores y acondicionamiento de señales

Una consideración importante es el tipo de corriente que se usará. Para los fines que pretende alcanzar el presente trabajo, se da por hecho que se usará corriente continua (DC).

Circuitería de la UCP

La circuitería de la unidad central de proceso ya se consideró en el Capítulo 4.1 y equivale a los osciladores, resistencias, capacitores y demás elementos que logran que la tarjeta funcione de manera adecuada.

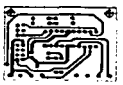
Electrónica de Potencia

Lo que hace la electrónica de potencia es pasar del manejo de elementos de un bajo consumo de corriente a otros que son alimentados a niveles considerablemente altos. Además, separa la parte electrónica de la parte eléctrica del proyecto, evitando así el calentamiento de los componentes electrónicos y los corto circuitos.

La función principal de la electrónica de potencia en un sistema mecatrónico es la de controlar la alimentación de los actuadores en el proyecto. Los actuadores son los elementos que hacen posible el movimiento de los elementos del sistema.

La separación de lo electrónico de lo eléctrico se puede hacer por medio de varios dispositivos, como pueden ser fotoacopladores, relevadores, transistores, etc.

La etapa de potencia más sencilla que se puede encontrar en este tipo de sistemas es el que hace uso de transistores, a los cuales, se les da este uso se les denomina transistores de amplificación de corriente. La Figura 4.2.1 muestra un esquema de un transistor utilizado como amplificador de corriente.



4.2 Diseño de Circuitos para Aplicaciones Mecatrónicas

La electrónica en un sistema mecatrónico, a grandes rasgos, se divide en las siguientes partes:

- La circuitería para el funcionamiento de la UCP
- Electrónica de potencia
- Interruptores, sensores y acondicionamiento de señales

Una consideración importante es el tipo de corriente que se usará. Para los fines que pretende alcanzar el presente trabajo, se da por hecho que se usará corriente continua (DC).

Circuitería de la UCP

La circuitería de la unidad central de proceso ya se consideró en el Capítulo 4.1 y equivale a los osciladores, resistencias, capacitores y demás elementos que logran que la tarjeta funcione de manera adecuada.

Electrónica de Potencia

Lo que hace la electrónica de potencia es pasar del manejo de elementos de un bajo consumo de corriente a otros que son alimentados a niveles considerablemente altos. Además, separa la parte electrónica de la parte eléctrica del proyecto, evitando así el calentamiento de los componentes electrónicos y los corto circuitos.

La función principal de la electrónica de potencia en un sistema mecatrónico es la de controlar la alimentación de los actuadores en el proyecto. Los actuadores son los elementos que hacen posible el movimiento de los elementos del sistema.

La separación de lo electrónico de lo eléctrico se puede hacer por medio de varios dispositivos, como pueden ser fotoacopladores, relevadores, transistores, etc.

La etapa de potencia más sencilla que se puede encontrar en este tipo de sistemas es el que hace uso de transistores, a los cuales, se les da este uso se les denomina transistores de amplificación de corriente. La Figura 4.2.1 muestra un esquema de un transistor utilizado como amplificador de corriente.

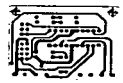
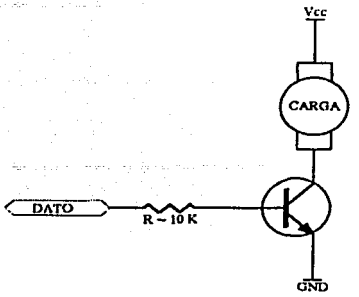
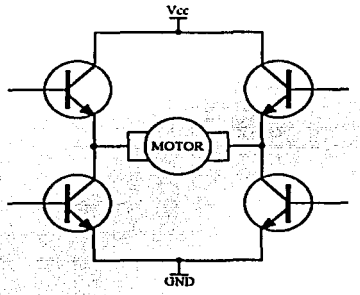


Fig. 4.2.1
Transistor amplificador



Como se observa en la figura anterior, la carga es el elemento cuyo consumo de corriente puede ser mayor que el que soportan los elementos electrónicos; se puede pensar que la carga es un motor, por ejemplo, entonces se deduce que el motor acoplado de esta manera solamente podría girar en un sentido, por lo que se arman arreglos de transistores que permitan el giro del motor en ambos sentidos; a estos arreglos se les denomina "puentes H" (Fig. 4.2.2)

Fig. 4.2.2
Puente H de transistores



La desventaja de un puente H de transistores es el tamaño que ocupa, por lo que se ideó un chip que contiene al puente y además permite diferentes configuraciones. El chip puente H es el L293, creado por Texas Instruments, y ahora es fabricado por varias compañías.

El chip L293 contiene dos circuitos H completos o cuatro medios puentes, los cuales pueden ser usados para manejar dos motores DC continuos independientemente y en ambos sentidos y para manejar cuatro motores en un sentido, combinarlos o para manejar un motor a pasos bipolar; tiene dos entradas de voltaje que soportan hasta 36V y 2 A.

4.2.1 Descripción de la Tarjeta de Circuito H

Para facilitar aún más el control de los motores, se diseñó una tarjeta en circuito impreso, doble cara, multiconfigurable.

El objetivo de esta tarjeta es la utilización de motores de corriente directa continuos o a pasos sin realizar modificaciones a la circuitería, esto se logra posicionando un par de jumpers en el header denominado JP1. La siguiente figura muestra las partes que componen esta tarjeta.

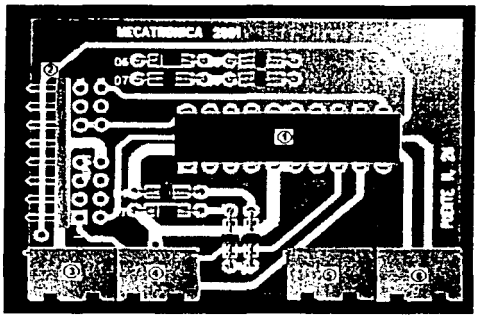
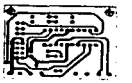


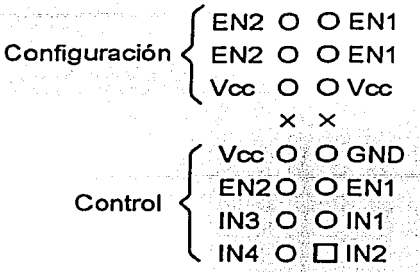
Fig. 4.2.3 Vista 3D de tarjeta de circuito H para control de motores de pasos y continuos

Descripción de la tarjeta:
1) Chip puente H (H Bridge) L293
2) Header de control y configuración de 2x7
3) Entrada del voltaje de alimentación para el motor 2
4) Entrada del voltaje de alimentación para el motor 1
5) Salida de voltaje para el motor 2
6) Salida de voltaje para el motor 1

Los diodos que contiene la tarjeta son para la protección del chip, lo protegen contra regreso de voltaje al dejar de alimentar a los motores o por el cambio en el sentido de giro. En el apéndice C se muestra el diagrama esquemático, diagrama de montaje, diagramas de pistas y lista de materiales para el armado de la tarjeta.

El header INPUT de la tarjeta de circuito H se divide en dos partes:



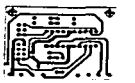


Pines de Configuración
La siguiente tabla explica el funcionamiento de la tarjeta debido a la configuración de sus pines:

EN2 ■ ■ EN1 EN2 ■ ■ EN1 Vcc O O Vcc	En esta posición los dos puentes H del L293C son habilitados por el microcontrolador. La habilitación de cada puente puede ser la salida PWM del microcontrolador.
EN2 ■ O EN1 EN2 O ■ EN1 EN2 ■ ■ EN1 EN2 ■ ■ EN1 Vcc O ■ Vcc Vcc ■ O Vcc	En cualquiera de estas dos configuraciones uno de los puentes H del L293C es habilitado directamente por su conexión a 5 V, mientras que el otro puente es habilitado mediante el microcontrolador.
EN2 ■ ■ EN1 EN2 O ■ EN1 Vcc O ■ Vcc	Mediante esta configuración los dos puentes H contenidos en el L293C son habilitados por su conexión a 5V; la gran ventaja de realizar esta conexión es el ahorro de dos líneas del microcontrolador.

Pines de Control
Los pines de control son 8, de los cuales 2 son la entrada de alimentación TTL de la tarjeta.

Los pines con el designador E# son los que habilitan cada uno de los puentes y el sufijo indica que puente H se está utilizando. Si este pin se mantiene en 1 lógico significará que está habilitado. Para el control de velocidad en un motor continuo, estas entradas deberán ser alimentadas con la señal PWM, y los pines de configuración deberán estar correctamente configurados.



Los pines IN# son los que gobiernan el sentido de giro (ver el párrafo anterior), cada una de estas líneas puede alimentar a un motor continuo utilizando medio puente; si se utilizan en parejas, (IN1, IN2) e (IN3, IN4), la tarjeta manejará dos puentes completos y ambos motores pueden girar en ambos sentidos.

Para el uso de motores de pasos (PAP*), la configuración de la tarjeta es muy específica:

1. Los bits de configuración deberán estar como muestra la ultima figura de los pines de configuración, en otras palabras, habilitados los dos puentes H.
2. Las dos entradas de voltaje que permite la tarjeta deberán estar puenteadas.
3. Las cuatro líneas IN# manejarán la secuencia de pasos y la velocidad del motor estará en función de la velocidad de cambio de datos de esta secuencia.
4. Los cables del motor a pasos serán conectados a la salida de voltaje para los motores con el siguiente orden: A1 y A2 a la salida de voltaje del motor 1, B1 y B2 a la salida de voltaje del segundo motor. Las conexiones se harán de derecha a izquierda.

El apéndice C muestra el layout de esta tarjeta.

4.2.2 Circuitos para Entrada de Información

Para que la UCP tenga retroalimentación de los eventos exteriores, es necesario realizar circuitos que envíen esta información. La información puede ser tanto discreta o lógica (0 ó 5 V) como continua o analógica (0 – 5 V, por ejemplo). A estos tipos de elementos electrónicos se les denomina transductores.

Una gran parte de los transductores, tanto continuos como discretos, funcionan bajo el mismo principio: uno de los pines del transductor está conectado a un voltaje constante, mientras que otro es conectado directamente a una entrada del microcontrolador o a la circutería de acondicionamiento de señal que irá conectada al microcontrolador, a la salida de un transductor se concoce con el nombre de señal. Cuando el transductor empieza a sensar, el voltaje con el que está alimentado se restringe permitiendo un pequeño flujo de voltaje (este voltaje también puede ser bloqueado dependiendo del tipo de transductor); el calor que se genera debido al bloqueo en el paso de el voltaje es disipado mediante la conexión de una resistencia. Cuando se tiene un transductor

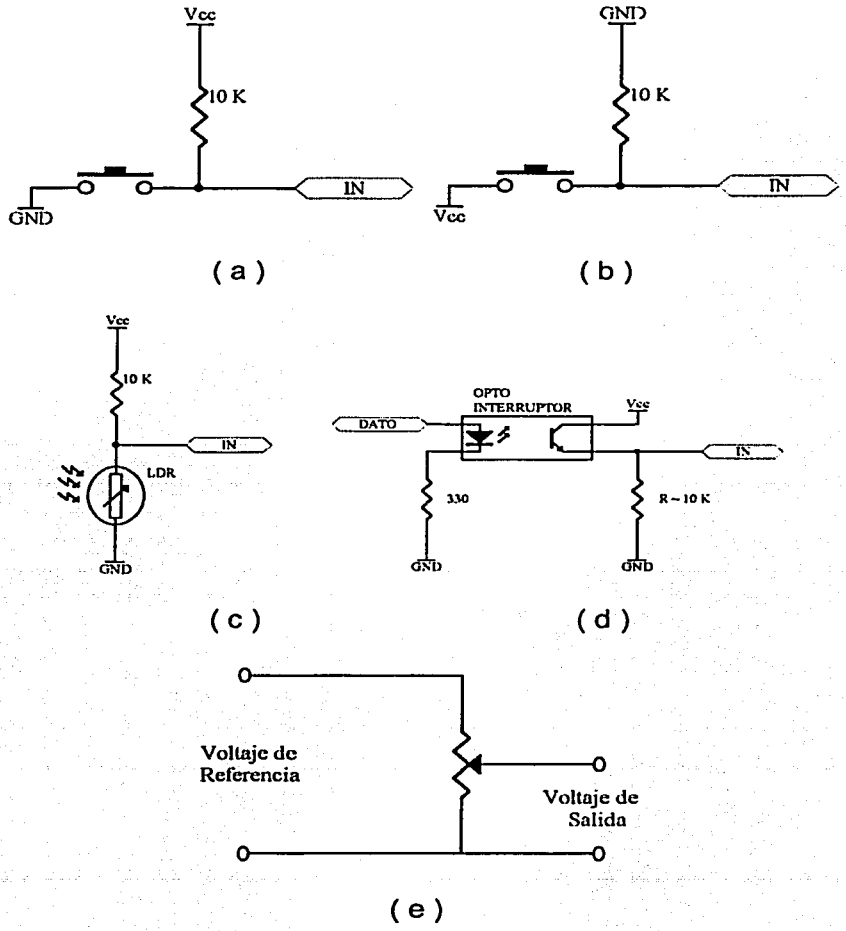
* PAP – Paso a paso

discreto, el voltaje de la señal del transductor debe ser conectado a una entrada digital del microcontrolador, la señal debe ser 0 ó 1 lógico dependiendo del circuito armado; en un transductor continuo la señal que éste genera deberá ser conectada a una entrada analógica en el microcontrolador, el voltaje no necesariamente debe llegar hasta valores lógicos, inclusive puede sobrepasar estos valores.

Las siguientes figuras muestran el diagrama esquemático de algunos circuitos de entrada de información comunes:

Fig. 4.2.4
Dispositivos para entrada de información:
a) y b) Interruptores
c) Foto resistencia
d) Opto interruptor
e) Potenciómetro.

c), d) y e) pueden funcionar de manera analógica o digital



4.2.3 Descripción de Tarjeta para Sensores Ópticos

Esta tarjeta fue diseñada con el objeto de contener en una sola tarjeta la posibilidad de conectar hasta cuatro sensores ópticos, cuyas salidas pueden ser analógicas o digitales, además de incluir un inversor 74LS04 cuyo propósito es el de mejorar la señal de salida amplificándola e invirtiéndola.

La siguiente figura muestra las partes de la tarjeta:

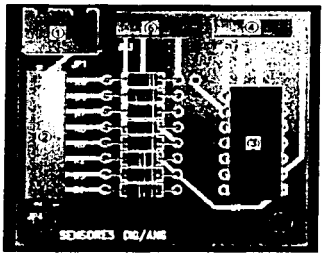


Fig. 4.2.5
Vista 3D de tarjeta para sensores ópticos

1) Alimentación TTL
2) Conexión para 4 sensores ópticos
3) Circuito integrado 74LS04
4) Salida digital invertida
5) Salida digital y/o analógica

Las resistencias que están conectadas a los LEDs de los sensores se encuentran en un rango entre 100 y 500 Ω, mientras que las resistencias conectadas al fotodetector se encuentran en un rango de 1 a 10 kΩ; es importante caracterizar a los sensores ópticos antes de armar la tarjeta.

El uso de esta tarjeta dentro del proyecto va enfocado al AGV, es conectada a los sensores de los encoders en las ruedas y los sensores foto reflectivos para seguir pista.

El apéndice D muestra el layout de esta tarjeta.

DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Software para el Uso de la Tarjeta de Desarrollo PIC 16F874

5.1 Manual de Referencia Rápida del MPLAB

Para el uso del lenguaje ensamblador dirigido al microcontrolador PIC 16F874 basta con el uso de un editor de texto común y de sus instrucciones; para compilar este tipo de archivos, existe gran número de compiladores, algunos libres y la mayoría comerciales. Microchip, compañía creadora de los microcontroladores PIC, ofrece un IDE (Ambiente de Desarrollo Integrado) para el desarrollo de aplicaciones, este IDE es gratuito y cuenta con los siguientes elementos:

MPLAB Project Manager.- Manejador de proyectos
MPASM.- Compilador
MPLINK.- Interfaz entre librerías
MPLAB – Editor.- Editor de programas
MPLAB – SIM.- Simulador
MPLAB – ICD.- Depurador integrado
MPLAB – ICE.- Software de emulador para el ICE
PICMASTER.- Software de emulador para el PICMASTER
PICSTART.- Software de programación para el PICSTART
PRO MATE.- Software de programación para el PRO MATE

Si lo que se desea es compilar un archivo en lenguaje ensamblador, bastará con la instalación de MPASM, se ejecuta, se le indica la dirección de archivo a compilar (extensión asm) y se oprime el botón de ok. Si lo que se quiere hacer es editar un programa, compilarlo y además simularlo, habrá que instalar el MPLAB y ver que incluya el MPASM, MPLAB Project Manager, MPLINK y MPLAB – SIM.

Los pasos a seguir para realizar un programa, compilarlo y simularlo en MPLAB son los siguientes:

1. Se crea un proyecto nuevo
2. Se selecciona el chip a utilizar
3. Se escoge el nombre del archivo que se programará (también se le llama nodo)
4. Se crea un nuevo archivo y se le asigna el nombre que se escogió en el punto 3
5. Una vez terminado el programa se compila presionando la tecla F10
6. Para simularlo se presiona la tecla F9
7. Para animarlo se presiona Ctrl + F9
8. Para simularlo paso a paso F8

Para mayor información, referirse al manual del MPLAB.

5.2 Manual de Referencia Rápida del CC5x

El CC5x es un compilador en lenguaje C para microcontroladores PIC. Aunque no cuenta con todos los comandos que contiene un C normal, es lo suficientemente poderoso como para el manejo de registros signados, registros en punto flotante, operaciones aritméticas, operaciones logarítmicas, operaciones trigonométricas y manejo de registros de 8, 16, 24 y 32 bits; por lo anterior se puede deducir que es una herramienta sumamente poderosa para la programación de microcontroladores.

Tabla 5.1
Rango de valores enteros

Tipo	Valor Mínimo	Valor Máximo
Int8	-128	127
Int16	-32768	32767
Int24	-8388608	8388607
Int32	-2147483648	2147483647
Uns8	0	255
Uns16	0	65535
Uns24	0	16777215
Uns32	0	4294967295

Otra de las ventajas, es el poder ser instalado dentro del IDE MPLAB, así que aunque no contenga editor ni simulador por sí mismo, es capaz de interactuar con los de MPLAB.

Para instalar este programa dentro de MPLAB se necesita copiar los archivos cc5x.mtc y tlcc5x.ini que se encuentran dentro de la carpeta del CC5x, en la carpeta donde se encuentra MPLAB.

El CC5x incluye librerías con un gran número de chips, cada una de estas librerías están bajo la extensión h, por ejemplo: 16f874.h. cuando se quiera correr un programa en MPLAB, habrá que hacer referencia a la librería del chip que se esté utilizando. Estas librerías contienen el direccionamiento de múltiples variables, las cuales coinciden con el mapa de memoria del manual del chip.

Este programa contiene algunas funciones internas que se muestran en la siguiente tabla junto con su equivalencia en ensamblador:

Tabla 5.2
Función interna en C y su equivalente en ensamblador

Función Interna	Equivalente
btsc(Carry);	btfsc f,b
btss(bit2);	btfss f,b
clrwdt();	clrwdt
clearRAM();	–
i = decsz(i);	decfsz f,d
W = incsz(i);	incfsz f,d
nop();	nop
nop2();	–
retint();	retfie
W = rl(i);	rlf i,d
i = rr(i);	rrf i,d
sleep();	sleep
skip(i);	–
k = swap(k)	swapf k,d

Cuando se ha compilado un archivo en C también se genera un archivo con extensión asm, por lo que se puede depurar o editar directamente en lenguaje ensamblador.

Una desventaja del uso del C en vez del ensamblador directo, es el tamaño del código generado, se genera alrededor de 5 veces más código.

Para mayor información, referirse a el manual del CC5x.

5.3 Manual de Referencia Rápida del IC-Prog 1.04

Este es un programa en ambiente Windows que permite programar una gran cantidad de microcontroladores PIC de las gamas media y baja; también algunos otros dispositivos como memorias cuya programación también es serial.

Entre sus características más relevantes están:

- Flexibilidad al permitir trabajar con diferentes programadores, ya sea paralelos o seriales
- Configuración a diversos puertos
- Posibilidad de invertir salidas y entradas desde el software
- Se puede conectar y trabajar con más de un programador a la vez
- Opción de salvar el contenido del chip en un archivo de tipo HEX o ASM; en otras palabras, contiene un desensamblador
- Puede leer diferentes formatos de archivos ensamblados (HEX8, HEX16, Binarios, Objects, Motorola S, EEPROM)

- Genera un folder por cada buffer
- Selección de idioma

Como se puede ver es un software bastante poderoso, compacto y que cumple excelentemente con su objetivo, además de ser totalmente gratis.

La instalación es tan sencilla como crear una carpeta y copiar el archivo ejecutable ICProg.exe en ella:

C:\...\ICProg\ICProg.exe

Es conveniente crear un acceso directo ya sea al escritorio o al menú de programas.

El programa trae una configuración por defecto, pero cada cambio que se produzca será predeterminado para la siguiente vez que se reinicie el programa, por lo anterior, se puede inferir que si lo configuramos desde la primera vez para que funcione con el programador y el chip que más utilizamos, no se tiene que volver a configurar de nuevo.

En el presente trabajo, se utiliza el programador serial denominado *JDM Programmer*. Los datos que se pueden variar en su configuración son únicamente los *Puertos* y *Retraso*, por lo que se recomienda que el retraso sea el más pequeño posible, ya que no afecta el buen funcionamiento del hardware.

Este software también cuenta con un probador de hardware. Cuando se habilita una de las opciones se puede medir en el programador, con ayuda de un osciloscopio y conectando debidamente la referencia a tierra, la correspondencia. Si no se ha habilitado, el dato leído deberá ser ≤ 0 V, si está habilitado se deberá leer lo siguiente:

Nota: No es lo mismo un programador para puerto serie que un programador serie. Un programador para puerto serie es el que se conectará a un puerto COM, mientras que un programador serie o serial nos habla de la manera en que envía los datos sin importar si se conecta en un puerto serie o en uno paralelo.

Señal	Habilitada
Data Out	5 V
Clock	5 V
MCLR	8-14 V
VCC	5 V *

Este programa es capaz de programar diversos dispositivos. Todos tienen una programación de tipo serial y es necesario seleccionar adecuadamente con cual se va trabajar para que aparezcan en la pantalla más opciones de programación específicas de cada chip.

El IC-Prog cuenta con desensamblador, por lo que existe la opción de ver el buffer ya sea en hexadecimal o en lenguaje ensamblador para PIC, no tiene ensamblador, por lo que no se puede abrir un archivo en extensión ASM; lo que si se puede hacer es salvar un archivo en extensión ASM o HEX, esto depende del modo de ver que se tenga seleccionado y la información que se salvará es la que muestre la ventana del buffer. Solamente se puede hacer edición del buffer en el modo hexadecimal.

La palabra de configuración se puede asignar desde este programa, bastará señalar las opciones que nos convengan, así como el oscilador, identificador de protección y la protección contra escritura. Las opciones para oscilador, como en todos los PICs, son 4: RC, LP, XT y HS (ver Capítulo 3).

Para poder llegar a entender de manera mucho más fácil el funcionamiento del programa y programador se propone el siguiente ejemplo.

Problema: Se tiene un programador serie que se conectará a una PC en el COM 2 y un PIC 16F874, el cual tiene cierta información

* El programador que incluye la tarjeta PIC 16F874, está diseñado de tal manera que al habilitar la salida MCLR, ésta alimenta a un regulador de voltaje que genera los 5 V, por lo que si se habilita la salida VCC, no se observará ningún efecto.

programada que se desconoce y que se quiere guardar en un archivo ASM para estudiarlo. Se programará un archivo en el PIC (*Ejemplo.HEX*). El PIC está instalado en una tarjeta funcionando con un cristal a 8 MHz.

Procedimiento:

1. Se desconecta la tarjeta del PIC de cualquier fuente de alimentación que tenga
2. Se conecta el cable al COM 2 de la PC
3. Se ejecuta el IC-Prog 1.04
4. Se verifica la configuración del hardware (F3). Debe contener los siguientes datos:
   a. JDM Programmer (este programador es para puerto serie)
   b. Puerto COM 2 (así lo pide el problema)
   c. Interfaz Direct I/O
   d. El Retraso I/O (lo más pequeño posible)
   e. Todo lo demás no debe estar seleccionado
5. Se selecciona, en dispositivos, al PIC 16F874
6. Se oprime el botón de *Leer todo*. Va a aparecer una ventana que nos indica el porcentaje de lectura, cuando la lectura haya finalizado aparecerán los datos contenidos en el PIC en las diversas ventanas

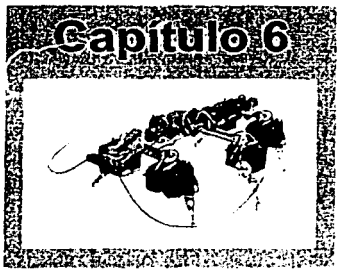
   Nota: Se puede utilizar en la misma ventana del Buffer1 para resolver el problema, pero por fines didácticos se utilizaran más de uno.

7. Para cambiarnos de la ventana de Buffer1 a la de Buffer2 bastará con tocar la pestaña que dice Buffer2.
8. En el Buffer2 se programará el archivo *Ejemplo.HEX*. Este archivo se abre igual que cualquier programa de Windows o se arrastra hasta la ventana. Ya abierto el archivo, su información aparecerá en las diversas ventanas.
9. Antes de mandar a *Programar todo*, es necesario hacer las selecciones pertinentes en la ventana de *Configuración*, en la que lo más importante es la selección del oscilador, pues la rutina programada no funcionará si se hace una mala elección

del oscilador. El problema menciona un cristal de 8 MHz, por lo cual se sabe que es un oscilador HS.

10. Se manda a *Programar Todo*. Si existe algún problema el programa mandará un mensaje de error. Se puede seleccionar desde la configuración del software qué mensajes mandar y qué tipo de verificación de grabado hacer.
11. Ya finalizada la grabación, se selecciona el Buffer3, y se lee de nuevo el PIC.
12. Se tiene la siguiente información:
    a. En el Buffer1, lo que tenía grabado el PIC en un principio.
    b. En el Bufer2, el contenido del archivo *Ejemplo.HEX*.
    c. En el Bufer3, el contenido actual del PIC.
13. Se compara ahora la información del Buffer2 con Buffer3 (*Buffer/Compar*). Si los Buffer son iguales aparecerá un mensaje de comparación realizada. Si se compara el Buffer1 con el Buffer2, aparecerán las diferencias que tienen.
14. Por último, el contenido del Buffer1 se guardará en un *archivo.ASM* por lo que se ejecuta la opción *Ver/Ensamblador* con lo que se desensambla la información hexadecimal y aparece el programa en ensamblador, después, se selecciona *Guardar Archivo* que asignará por default una extensión ASM.

Para mayor información, referirse a el manual del IC-Prog.



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Diseño de un AGV Genérico

Un vehículo autoguiado o AGV (Automatic Guided Vehicle) es un robot capacitado para moverse de forma autónoma en un entorno determinado, realizando tareas como el transporte de objetos o la manipulación y recolección de muestras, evitando la colisión con los demás elementos que le rodean.

Un AGV consiste principalmente en una plataforma con ruedas con un microcontrolador incorporado y una fuente de alimentación propia. Cuando se ha logrado el control autónomo del AGV, el siguiente paso es la incorporación de herramientas para realizar una tarea específica.

6.1 Componentes de un AGV

- **Unidad de control**
  Se compone de tres partes, que pueden ir implementadas en el mismo dispositivo o bien situadas por separado en tres unidades:
  - **Control de conducción**
    Controla la alimentación de los motores del sistema locomotor. Se encarga de la aceleración / desaceleración, la velocidad, si el frenado ha de ser suave o rápido, etc.
  - **Unidad de control**
    Controla los aspectos necesarios para llevar a cabo las funciones de la aplicación, como son el control del brazo robot para recolección de objetos, etc.
  - **Unidad de seguridad**
    Su función es la de estar alerta y actuar para evitar las colisiones o pérdida de control.
- **Fuente de alimentación**
  El AGV debe llevar su fuente de alimentación autónoma. Esta suele consistir en baterías eléctricas recargables.
- **Sistema locomotor**
  Pueden ser ruedas, rodillos, patas articuladas, hélices, etc.
- **Herramientas**
  Brazos robot, ventosas, sondas, aspiradoras, etc.
- **Actuadores**
  Se usan motores normalmente de corriente continua y con realimentación, pistones, válvulas de aire, motores lineales, etc.
- **Sensores**
  Cumplen una doble función: la 1ª es de seguridad, para evitar colisiones, la 2ª función es la de realimentación, para enviar a la unidad de control información sobre el entorno que le rodea; esta información es necesaria para conformar la trayectoria a seguir. Los sensores que puede llevar implementado un AGV son: cámaras para detectar objetos, antenas para enviar y

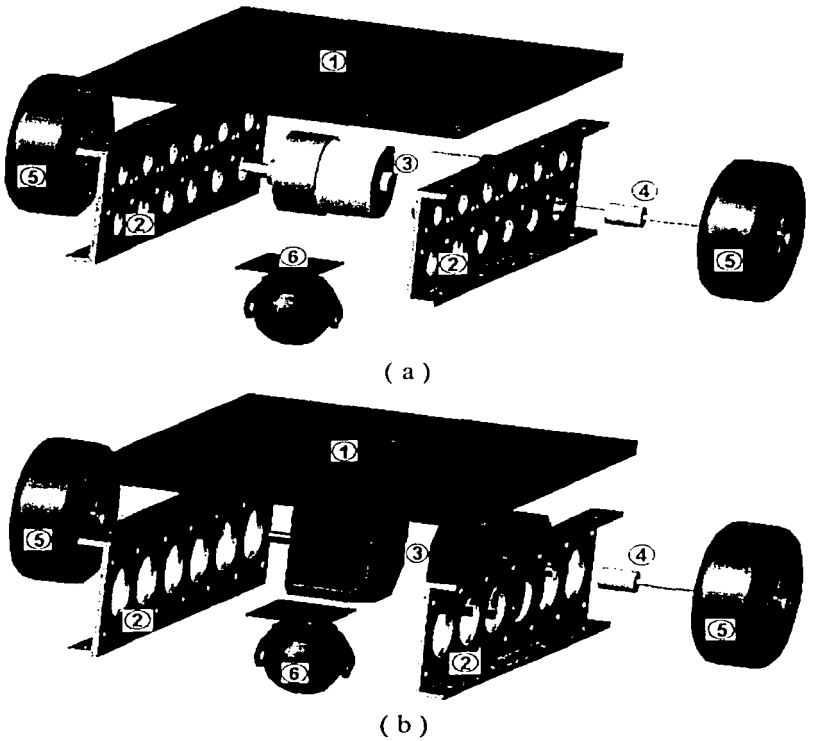
recibir señales, láser para barrer el área circundante, sensores ultrasónicos para detectar proximidad, encoders y potenciómetros para medir velocidad y ángulo de giro, bumpers (defensas) para detectar colisiones, sensores de infrarrojos, etc.

6.2 Diseño del AGV

Como se mencionó anteriormente, un AGV, típicamente es una base plana con ruedas, pero este tipo de estructura abre una amplia gama de posibilidades para poder darle una aplicación específica.

El objetivo de este capítulo es proponer un AGV sencillo y flexible, un AGV que se pueda construir con el mínimo de materiales y que pueda ser modificado rápidamente para realizar sobre la misma estructura otra aplicación.

Fig. 6.1
Propuesta de AGV genérico.



La estructura del vehículo consta de las siguientes partes:

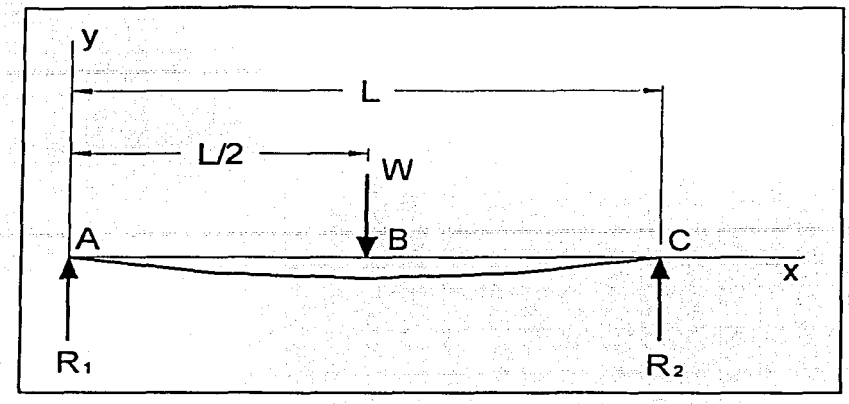
1) Una base plana de material ligero, aislante y lo bastantemente rígido como para no permitir su deformación para soportar la aplicación, la alimentación y la circuitería. Las dimensiones de la base son: 15x20 cm
2) Las paredes son de un material ligero y resistente, en las cuales se sujetan los motores de la locomoción. Estas paredes están maquinadas con varios orificios para sujeción de motores con el objeto de proporcionar al vehículo flexibilidad en cuanto a la localización de los mismos y a su vez servir como base para la sujeción de motores, sensores, mecanismos y demás instrumentos para la implementación de alguna aplicación
3) Los motores son de corriente directa: continuos y a pasos. Los motores continuos con los que cuenta el departamento de mecatrónica son de dos tamaños (Fig. 6.3 a, b). También se cuenta con motores a pasos de un solo tamaño (Fig. 6.3 c).
4) Los coples deben ser de un material muy resistente, ya que el diámetro menor es muy pequeño y la flecha es larga
5) Las ruedas deben ser de un material suave y con buena adherencia al suelo, con un diámetro lo suficientemente grande para que el AGV no arrastre. En el caso de contar con encoder, la rueda tendrá que ser mayor que el diámetro del disco de código
6) La rueda frontal es loca, es decir, gira libremente. El tipo de rueda que se muestra en la figura 6.4 c, tiene dos grados de libertad, uno en el eje perpendicular al plano del suelo y el otro en el eje de rotación de la bola de la rueda.

6.3 Selección de Materiales

En la construcción de la base del AGV se propone un material plástico laminado, ya que sus propiedades mecánicas con ¼" de espesor satisfacen por mucho las exigencias de una aplicación respetando las dimensiones del vehículo propuesto.

De los materiales plásticos se encuentran dos que se distinguen entre todos, el PVC espumado y el acrílico. Los criterios de selección usados fueron en primer lugar la facilidad de adquirirlo, además que ambos materiales son maquinables, de un precio moderado, ligeros y con buenas propiedades mecánicas considerando las dimensiones del AGV:

Considerando a la base como una viga doblemente empotrada se tienen el siguiente esquema:



donde

h - Espesor

L - Ancho

I - Momento de inercia

R_1 - Fuerza resultante en el punto A

R_2 - Fuerza resultante en el punto B

V_{AB} - Cortante en el punto B

V_{BC} - Cortante en el punto C

y_{max} - Esfuerzo máximo

E - Módulo de elasticidad

sustituyendo

$eh = 6.35x10^{-3}$ [m]

el ancho de la placa, $b = 0.15$ [m]

$$I = \frac{b^3h}{12} = \frac{(0.15)^3(6.35x10^{-3})}{12} = 1.79x10^{-6}\ [m^4]$$

para considerar un esfuerzo máximo en la base se propone una masa de 5 kg concentrada en el centro, en la cual se aceptará deformación máxima de 5 mm

$$R_1 = R_2 = \frac{W}{2} = \frac{(5)(9.8)}{2} = 24.5\ [N]$$

$$V_{AB} = R1 = 24.5\ [N]$$

donde

el espesor del material, $h = 6.35x10^{-3}$ [m]

el ancho de la placa, $b = 0.15$ [m]

$$I = \frac{b^3 h}{12} = \frac{(0.15)^3 (6.35x10^{-3})}{12} = 1.79x10^{-6} \text{ [m}^4\text{]}$$

para considerar un esfuerzo máximo en la base se propone una masa de 5 kg concentrada en el centro, en la cual se aceptará deformación máxima de 5 mm

$$R_1 = R_2 = \frac{W}{2} = \frac{(5)(9.8)}{2} = 24.5 \text{ [N]}$$

$$V_{AB} = R1 = 24.5 \text{ [N]}$$

$$V_{BC} = -R2 = -24.5 \text{ [N]}$$

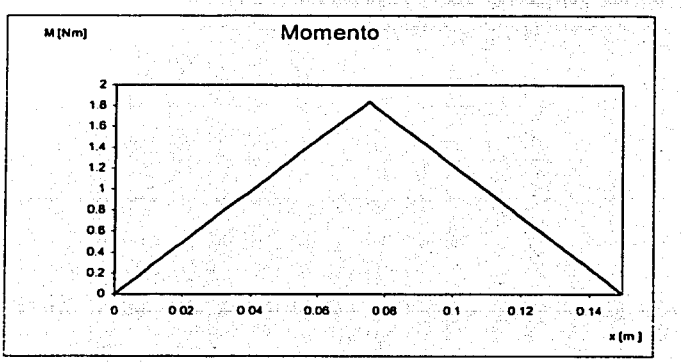
$$\left.\begin{aligned} M_{AB} &= \frac{Wx}{2} = \frac{(5)(9.8)}{2}x \\ M_{BC} &= \frac{W}{2}(L - x) = \frac{(5)(9.8)}{2}(0.15 - x) \end{aligned}\right\} \text{Graf. 6.1}$$

$$y_{max} = \frac{WL^3}{48EI} \Rightarrow E = \frac{FL^3}{48 y_{max} I}$$

$$E = \frac{(5)(9.8)(0.15)^3}{48(0.005)(1.79x10^{-6})} = 384\,951 \text{ [Pa]}$$

Sabiendo que la E del material es de 3.1 GPa se deduce que el material es lo suficientemente fuerte como para resistir la carga propuesta.

Gráf. 6.1 Gráfica de momentos aplicados a la base debido a la carga concentrada en el centro



Para las paredes de la estructura se seleccionó aluminio en barra con sección transversal en C (*patín*), pues es un material ligero, maquinable y la forma de la barra permite el fácil acoplamiento de la base, soporte a motores y la sujeción de otros elementos como pueden ser los de la aplicación.

6.4 Selección del Motor

Los motores son los músculos de un robot. Hay muchos tipos de motores, pero sólo algunos de ellos son utilizables en este tipo de robots. En el presente trabajo sólo se trabajará con motores de corriente directa, los cuales se clasifican de la siguiente manera:

- Continuos
- A pasos

La diferencia entre un motor continuo y un motor a pasos es la siguiente:

- En un motor continuo, al aplicarle voltaje, el rotor gira continuamente. El rotor se detiene únicamente cuando el voltaje es removido, o si el motor está detenido por una carga mayor a la que puede soportar. Este motor tiene un par pobre al arranque y conforme va incrementando su velocidad angular el par también aumenta en forma proporcional, la velocidad que logra alcanzar es considerablemente alta.

- En un motor de pasos, el rotor gira cierto ángulo después de ser alimentado correctamente. Para que éste pueda incrementar el ángulo del rotor y funcionar de manera "continua", es necesario llevar una secuencia en la alimentación de las bobinas que lo componen. En este motor, el par es constante a cualquier velocidad, su alimentación se hace a un voltaje constante, la velocidad del rotor está en función de la velocidad en la que se ingrese la secuencia.

6.5 Especificaciones de Motores

Voltaje

Todos los motores tienen cierto rango de voltaje de operación. Los motores de CD pequeños generalmente están referidos a rangos de entre 1.5 y 12 V, algunos de alta calidad están referidos a voltajes de entre 12 y 24 V. La mayoría de los motores pueden ser utilizados satisfactoriamente en voltajes poco mayores o menores de lo que esté indicado en sus especificaciones, pero la mayoría no funcionará con voltajes menores al 50% de su rango de operación; si es utilizado más del 30% por encima de su rango de operación, el motor se quemará y se ocasionarán daños en él.

Si no se sabe el rango de operación de un motor, lo conveniente es incrementar el voltaje de alimentación poco a poco hasta que empieza a funcionar, esto proporsiona el límite menor del rango; el límite superior lo da la temperatura: el motor debe ser capaz de disipar su calor, esto es, que no se caliente demasiado.

Otra forma de encontrar el límite máximo de operación es escuchándolo, el motor si es continuo, no debe zumbar de manera aguda o molesta.

Corriente
La corriente es importante sobre todo cuando hay que considerar cargas, que es cuando el motor está desplazando algún elemento o realizando algún trabajo. El suministro de corriente cuando un motor está trabajando sin carga es poca, pero cuando se le aplica la carga ésta aumenta considerablemente. El incremento de la corriente se refleja en la ley de Ohm:

$$Pot = Vi\ ,$$

esto es, la energía por segundo que consume el motor en un segundo para poder realizar el trabajo.

Alimentación
En la construcción de un robot, es sumamente importante tener en cuenta las posibilidades de alimentación de la tensión a los motores, ya que si se tienen motores que demanden una corriente alta, las baterías se descargarán rápidamente, el tiempo de descarga de una batería se nos da en los datos de placa de la misma, sus unidades son Ah (Ampere – hora), que es la corriente que puede suministrar en una hora.

Si los motores se están alimentando con un regulador de voltaje, también hay que fijarse qué corriente es capaz de suministrar para evitar dañarla.

El AGV propuesto trata de apegarse a las necesidades de la asignatura de Diseño Mecatrónico, minimizando la parte de diseño del AGV e incrementando la posibilidad de realizar una aplicación interesante; por lo anterior, se puede deducir que es una primera propuesta de estandarización de un AGV, la cual se realiza en función de los materiales con que cuenta el departamento (motores y demás actuadores), y materiales que se puedan conseguir con facilidad.

El departamento de mecatrónica cuenta con motores de corriente directa con reductores de velocidad mecánicos y motores a pasos. Los motores de corriente directa con que se cuenta son:

Tabla 6.1 Características de los motores de corriente directa con los que cuenta el Departamento de Mecatrónica

Motor	Voltaje [V]	Corriente [mA]	Velocidad [rpm]	Torque [kg-cm]	Reducción
1921Y	12	150	42	8	132:1
1632T	12	75	69	2.2	100:1
105873	12	150	–	0.5	–

Cuyas dimensiones en milímetros son:

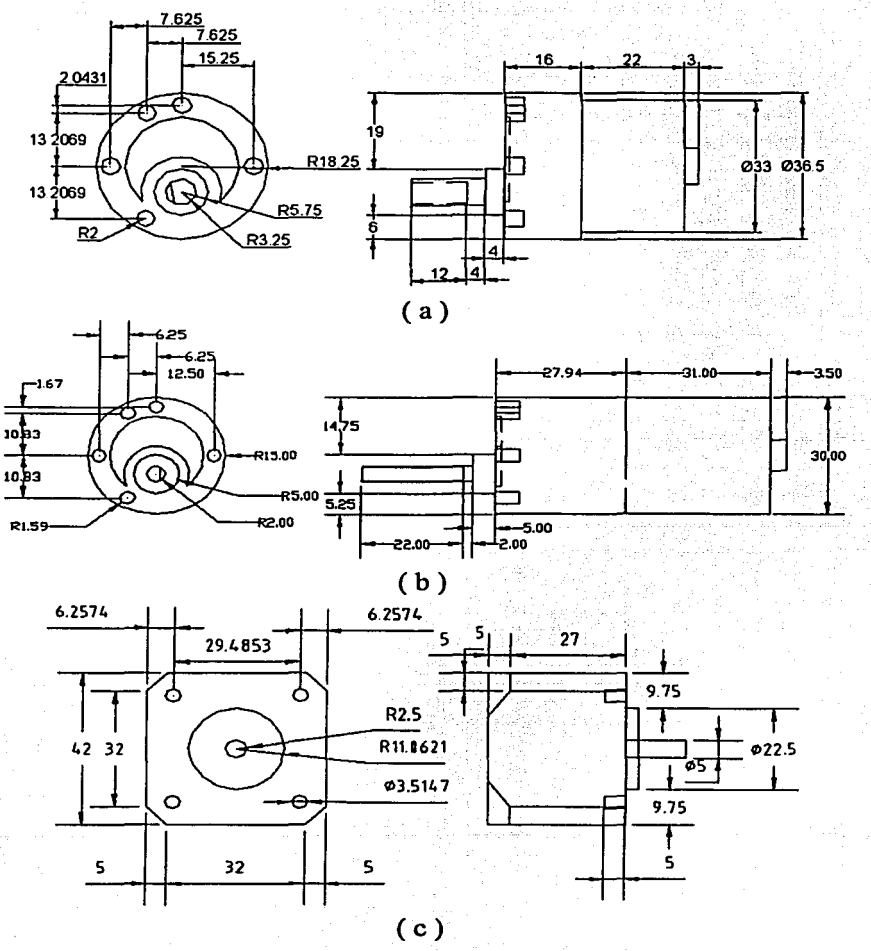


Fig. 6.3 Motores de corriente directa con que cuenta el Departamento de Mecatrónica
a) Motor continuo 1921Y
b) Motor continuo 1632T
c) Motor a pasos 105873

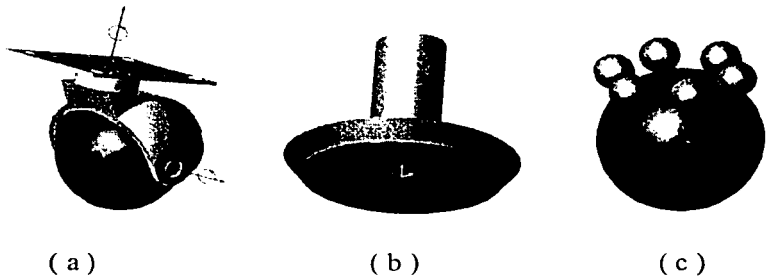
Los motores a paso con que se cuentan, no tienen reductores de velocidad.

6.6 Ruedas

Para la configuración de las ruedas se eligió un triciclo, por el ahorro de otras ruedas y por el cumplimiento de radios de giro sin elementos extras que causen fricción al ser arrastrados.

Para las ruedas de tracción, como ya se mencionó anteriormente, se proponen ruedas neumáticas o cualquier otras que sean de una goma suave y de un diámetro lo suficientemente grande como para que el AGV no arrastre. El acoplamiento de las ruedas con los motores debe ser mediante coples, los cuales permitan sustraer las ruedas cuando sea necesario sin dañar la flecha de los motores ni a las ruedas.

Para la tercer "rueda" se puede elegir entre una rueda loca, esta conviene que sea de un material relativamente rígido, que no permita deformación pero que tenga la suficiente adherencia al suelo; una plataforma de diámetro pequeño, pulida y con el filo muerto o una esfera de un material rígido empotrada en un cilindro con baleros que soporten carga axial, esta última propuesta, por ejemplo, se puede ver en la bola de los mouse para PC o en los desodorantes roll – on (Fig. 6.4).



(a) (b) (c)

Fig. 6.4 Opciones para selección de la tercer rueda
a) Rueda loca con dos grados de libertad
b) Plataforma pulida de filos muertos
c) Bola con baleros

Si se considera el empleo de una rueda loca, la distancia existente del eje de giro perpendicular a la base y el eje de giro de la rueda forzosamente tendrá que ser mayor que cero; en el caso de ser cero, la rueda se arrastraría cuando el AGV intente realizar una curva o cuando el eje de rotación de la rueda fuese paralelo al movimiento de giro del vehículo.

El material seleccionado para el cople es acero cold rolled. En el maquinado o selección del cople se debe considerar la sujeción con la flecha del motor, la sujeción con la rueda y en casos especiales, el espacio para la sujeción a él mismo de un disco de código para el uso de encoders. La siguiente Figura muestra un acoplamiento motor – disco de código – cople – rueda.

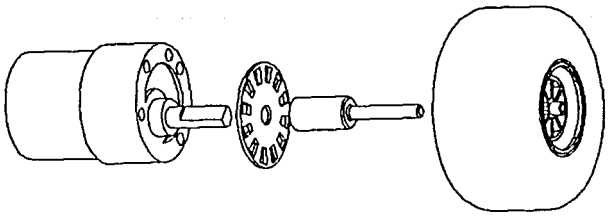
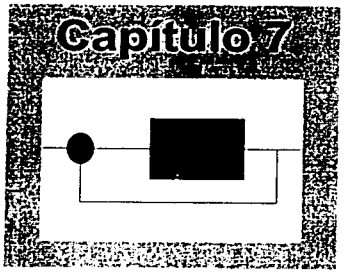


Fig. 6.5 Acoplamiento entre motor, disco de código para encoder, cople y rueda

Para ver las especificaciones de las medidas del AGV, véase el Apéndice F.



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Control del AGV

7.1 Aspectos Generales

Algunos controladores se pueden clasificar de acuerdo a la actividad que realizan dentro del sistema:

- Control ON/OFF
- Control Proporcional (P)
- Control Integral (I)
- Control Proporcional Integral (PI)
- Control Proporcional Derivativo (PD)
- Control Proporcional Integral Derivativo (PID)

Control ON/OFF

El elemento de control sólo proporciona dos niveles de control: total y nulo. Si el error, que es la entrada de controlador, que se presenta es e(t) y la señal de control que proporciona el controlador es m(t), su representación será:

$$m(t) = M_1 \text{ para } e(t) > 0$$
$$m(t) = M_2 \text{ para } e(t) < 0$$

Control Proporcional

Es un control más suave que el anterior, el controlador proporcional desarrolla una señal de control directamente proporcional al error. Actúa como un amplificador de ganancia kp y su representaciones:

$$m(t) = kp \cdot e(t)$$

Control Integral

La señal de control se modifica a una velocidad proporcional a la señal de error, es decir, si la señal de error es grande, la señal de control se incrementa con rapidez; si es pequeña, la señal de control se incrementa con lentitud. Su representaciones:

$$m(t) = ki \int e(t)\, dt$$

donde ki es la ganancia del integrador. Si el error tiende a cero, la salida del control permanece constante. Esta característica permite utilizar los controladores integrales cuando existe algún tipo de carga constante en el sistema. Incluso si no existiese ningún error, el controlador para neutralizar la carga, seguirá conservando una señal de salida.

Control Proporcional Integral (PI)
Un controlador proporcional es incapaz de neutralizar una carga en el sistema sin ningún error, mientras que un controlador integral puede proporcionar un error cero, pero suele suministrar una respuesta lenta. Para resolver este problema se utiliza un controlador compuesto, PI, el cual se representa como:

$$m(t) = kp \cdot e(t) + ki \int e(t)\,dt$$

Control Proporcional Derivativo (PD)
La acción del controlador derivativo proporciona una señal de control proporcional a la velocidad de cambio de la señal de error. Puesto que el control derivativo, no genera ninguna salida a menos que el error cambie con el tiempo. Puesto que el control derivativo, no genera ninguna salida al menos que cambie con el tiempo, en raras ocasiones se utiliza solo. El controlador PD se representa por:

$$m(t) = kp \cdot e(t) + kd \cdot \frac{d}{dt} e(t)$$

El efecto de la acción del control derivativo es anticipar cambios en el error y proporcionar una respuesta más rápida a los cambios.

Control Proporcional Integral Derivativo (PID)
Se combinan tres acciones de control. Su representación ES:

$$m(t) = kp \cdot e(t) + ki \int e(t)\,dt + kd \cdot \frac{d}{dt} e(t)$$

El control PID es el tipo de control más general, proporciona una respuesta rápida, una buena estabilidad del sistema y un bajo régimen de error permanente.

Control Proporcional Integral (PI)
Un controlador proporcional es incapaz de neutralizar una carga en el sistema sin ningún error, mientras que un controlador integral puede proporcionar un error cero, pero suele suministrar una respuesta lenta. Para resolver este problema se utiliza un controlador compuesto, PI, el cual se representa como:

$$m(t) = kp \cdot e(t) + ki \int e(t)\,dt$$

Control Proporcional Derivativo (PD)
La acción del controlador derivativo proporciona una señal de control proporcional a la velocidad de cambio de la señal de error. Puesto que el control derivativo, no genera ninguna salida a menos que el error cambie con el tiempo. Puesto que el control derivativo, no genera ninguna salida al menos que cambie con el tiempo, en raras ocasiones se utiliza solo. El controlador PD se representa por:

$$m(t) = kp \cdot e(t) + kd \cdot \frac{d}{dt} e(t)$$

El efecto de la acción del control derivativo es anticipar cambios en el error y proporcionar una respuesta más rápida a los cambios.

Control Proporcional Integral Derivativo (PID)
Se combinan tres acciones de control. Su representación ES:

$$m(t) = kp \cdot e(t) + ki \int e(t)\,dt + kd \cdot \frac{d}{dt} e(t)$$

El control PID es el tipo de control más general, proporciona una respuesta rápida, una buena estabilidad del sistema y un bajo régimen de error permanente.

7.2 Cinemática del AGV Propuesto

El usar un diferencial eléctrico, le permite al AGV tener una velocidad diferente en cada rueda, pero la velocidad en una rueda debe estar en función de la otra; por lo anterior se puede inferir que el radio de giro del vehículo puede escribirse en función de estas velocidades. La velocidad del vehículo será el promedio de las velocidades de las ruedas y está localizado en el punto ***C***. Al conocer el radio de giro, también se puede determinar el ángulo con que gira la rueda loca. Para poder hacer el presente análisis se debe considerar que el vehículo se encuentra en movimiento y que tiene un radio de giro constante.

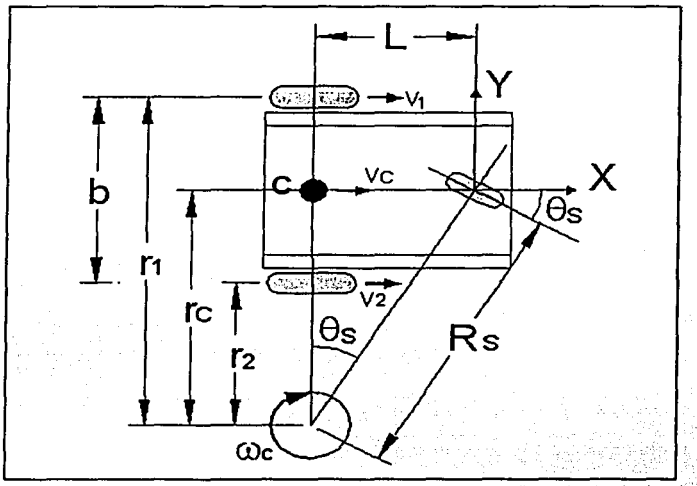


Fig. 7.1 Diagrama cinemático del AGV para el cálculo de la velocidad

b	Distancia entre ruedas traseras
C	Punto medio de **b**
θ_s	Angulo de giro de la llanta delantera
r_1	Radio de giro de la llanta 1
r_c	Radio de giro del punto ***C***
r_2	Radio de giro de la llanta 2
R_s	Radio de giro de la llanta delantera
L	Distancia entre la llanta delantera y el eje trasero
V_1	Velocidad tangencial de la de la llanta 1
V_2	Velocidad tangencial de la de la llanta 2
V_c	Velocidad crucero del punto ***C***
ω_1	Velocidad angular de la llanta 1 respecto al centro de rotación
ω_2	Velocidad angular de la llanta 2 respecto al centro de rotación
ω_c	Velocidad angular de ***C***

$R_s = L\cos\theta_s$.....(1)

$R_s = r_c\ sen\ \theta_s$.....(2)

$R_s = \sqrt{L^2 + r_c^{\,2}}$.....(3)

igualando (1) con (2)

$L\cos\theta_s = r_c\ sen\ \theta_s$

despejando a r_c

$$r_c = L\frac{\cos\theta_s}{sen\ \theta_s} = \frac{L}{\tan\theta_s}\text{.....(4)}$$

sustituyendo (4) en (3)

$$R_s = \sqrt{L^2 + \left(\frac{L}{\tan\theta_s}\right)^2} = L\sqrt{1 + \left(\frac{1}{\tan\theta_s}\right)^2}\text{.....(5)}$$

Cálculo de la Velocidad en cada Rueda :

$$V_c = \omega_c r_c \Rightarrow \omega_c = \frac{V_c}{r_c}\text{.....(6)}$$

$V_1 = \omega_1 r_1$ .....(7)

$V_2 = \omega_2 r_2$ .....(8)

$$r_1 = r_c + \frac{b}{2}\text{.....(9)}$$

$$r_2 = r_c - \frac{b}{2}\text{.....(10)}$$

se sabe que $\omega_c = \omega_1 = \omega_2$

entonces, sustituyendo (4), (6), (9) y (10) en (7) y (8)

$$V_1 = \frac{V_c}{r_c}\left(r_c + \frac{b}{2}\right) = V_c\left(\frac{2r_c + b}{2r_c}\right) = V_c\left(1 + \frac{b}{2r_c}\right) = V_c\left(1 + \frac{b}{2\left(\frac{L}{\tan\theta_s}\right)}\right)$$

$$\Rightarrow V_1 = V_c\left(1 + \frac{b}{2L}\tan\theta_s\right)\text{.....(11)}$$

$$\Rightarrow V_2 = V_c\left(1 - \frac{b}{2L}\tan\theta_s\right)\text{.....(12)}$$

se sabe que

$$V_c = \frac{V_1 + V_2}{2}\text{.....(13)}$$

$$\therefore r_c = \frac{r_1 + r_2}{2}\text{.....(14)}$$

El desplazamiento en el AGV se mide considerando que el vehículo tiene un radio de giro constante en un instante, entonces, el segmento del perímetro de la circunferencia que se ha sido cubierto será la distancia recorrida (D_n); las siguientes ecuaciones contienen las relaciones pertinentes:

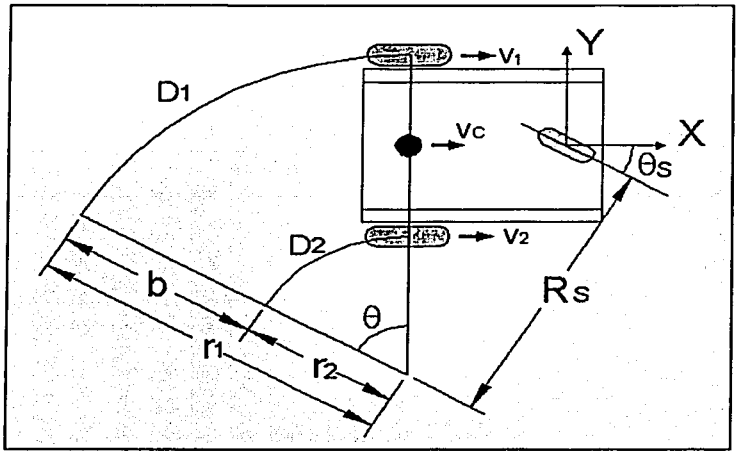


Fig. 7.2 Diagrama cinemático del AGV para el cálculo de l desplazamiento

$$\theta = \frac{D_1}{r_1} \text{.....(15)}$$

$$\theta = \frac{D_2}{r_2} \Rightarrow r_2 = \frac{D_2}{\theta} \text{.....(16)}$$

$$r_1 = r_2 + b \text{.....(17)}$$

sustituyendo (17) en (18) .

$$\theta = \frac{D_1}{r_2 + b} \text{.....(19)}$$

sustituyendo (16) en (19)

$$\theta = \frac{D_1}{\frac{D_2}{\theta} + b} = \frac{D_1}{D_2 + \theta b} \Rightarrow D_2 + \theta b = D_1$$

$$\therefore \theta = \frac{D_1 - D_2}{b} \text{.....(23)}$$

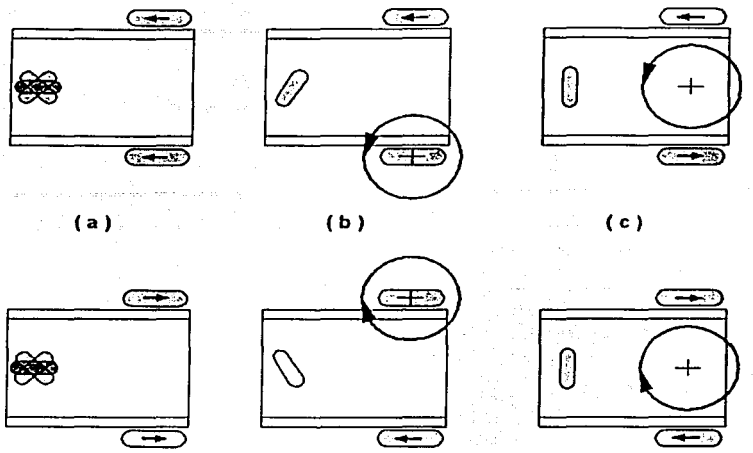


Fig. 7.3 Formas de desplazamiento en función de la rotación de sus llantas.
a) Movimiento en una sola dirección en línea recta o curvas
b) Rotación de una sola llanta, la otra se usa como pivote
c) Rotación de las llantas en sentidos opuesto

7.3 Control de Lazo Cerrado

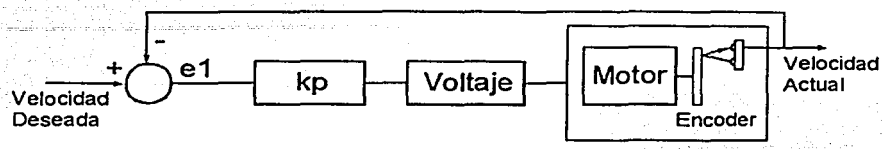
La modulación de ancho de pulso es un control de lazo abierto. En un control de lazo abierto, no hay retroalimentación de información por parte de los motores, así que el software no sabe con certeza qué tan rápido va o en qué posición. En los motores continuos, al variar su voltaje se modifica la velocidad angular de su rotor y por tanto la velocidad de un robot; pero, el terreno, los obstáculos superficiales, el patinado en el contacto de las ruedas, o la carga en el robot, también son variables a considerar para el correcto cálculo y control del robot, por lo que la variación de voltaje no implica una velocidad en particular.

Para implementar un algoritmo de control de velocidad o posición más reales, el AGV necesita sensores en las ruedas. Así que la retroalimentación genera lo que se llama un control de lazo cerrado. La Figura 7.5 muestra un sistema de control de lazo cerrado simple llamado control PI (Proporcional – Integral).

La idea básica de un control de lazo cerrado es tomar la decisión sobre la velocidad (se puede hacer mediante PWM), enviar esa decisión a los motores, ver qué tan rápido giran, transformar esa información a velocidad y compararla con la velocidad deseada. La diferencia es llamada señal de error (*e*) y puede ser tanto positiva como negativa.

Si se multiplica el error por una constante para producir una nueva orden, entonces el controlador es un controlador proporcional (Fig. 7.4).

Fig. 7.4
Control proporcional

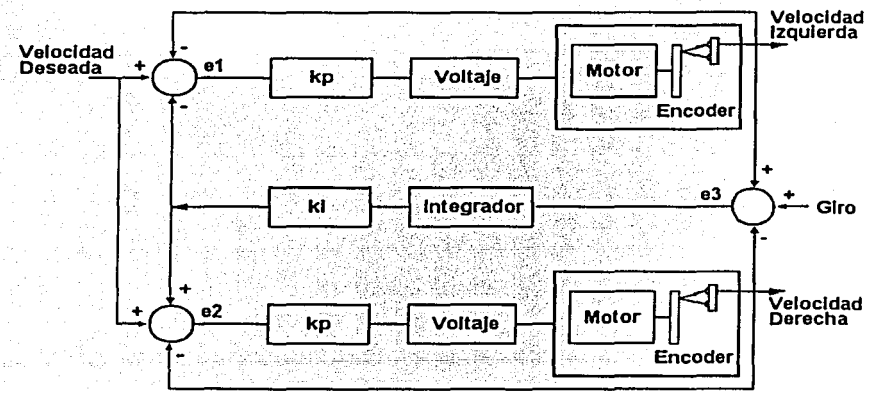


El ciclo produce una señal de error *e*1. Si la velocidad real es menor que la deseada, el error será positivo y la constante kp también es positiva, un voltaje mayor es mandado al motor. Si la velocidad real es mayor que la velocidad deseada, el signo del error es negativo y un voltaje menor es enviado al motor, lo que disminuye la velocidad. Este proceso se repite hasta que la señal de error sea suficientemente pequeña para considerar que se ha controlado a la velocidad deseada.

El software de control de los motores del AGV deben hacer dos cosas:

- La primera, que el AGV pueda mantener una velocidad constante (tanto subiendo una rampa como en una superficie horizontal).
- La segunda, supervisar que las dos ruedas sean esclavas una de otra. Esto es, si se le ordenó al robot ir derecho, la velocidad de las dos ruedas deben estar sincronizadas. Esta característica es lograda mediante la retroalimentación central de la Figura 7.5, el control integral.

Fig. 7.5
Control
proporcional integral



El control integral recibe la velocidad actual de ambos motores y las compara. La diferencia entre las dos velocidades actuales es el error *e*3, como se puede ver en la Figura 7.5 donde *e*3 = velocidad angular izquierda – velocidad angular derecha + giro. El término "giro" es usado para el dato entrada del giro. Mientras el giro es 0, la señal de error sólo cambiará cuando el robot no siga una línea recta desviándolo a un lado o a otro. Un control integral, integra o suma, la señal de error en el tiempo, multiplica esta suma por una constante, ki y alimenta la nueva orden de regreso junto con el control proporcional para cada motor con el signo diferente para cada caso. De esta manera, un motor incrementa su velocidad mientras el otro la disminuye hasta que cada velocidad alcanza una velocidad suficientemente cercana a la deseada.

En los diagramas de control se pueden ver diferentes bloques, el bloque denominado voltaje realiza el cálculo del nuevo valor del PWM, mientras que el bloque con el nombre de motor, para casos de simulación, son las ecuaciones que caracterizan al motor que se va a emplear.

7.4 Simulación

Para poder simular al AGV y el control PI propuesto, se considerará que el comportamiento de sus motores puede ser representado por un sistema de 1er orden, cuya función de transferencia es:

$$G(s)=\frac{b}{s+a}$$

Pasando a transformada z, es decir, al dominio discreto

$$G(z)=b\left(\frac{1-e^{-aT}}{2a}\right)\left(\frac{z+1}{z-e^{-aT}}\right)=\frac{\omega(z)}{V(z)}$$

donde T es el tiempo de muestreo. Se define la constante

$$K=b\left(\frac{1-e^{-aT}}{2a}\right)$$

entonces

$$\frac{\omega(z)}{V(z)}=K\left(\frac{z+1}{z-e^{-aT}}\right)$$

desarrollando esta expresión se tiene

$$\omega(z)(z-e^{-aT})=V(z)[K(z+1)]$$

pasando al tiempo discreto

$$\omega((k+1)T)-e^{-aT}\omega(kT)=KV((k+1)T)+KV(kT)$$

Aplicando un retraso se tiene:

$$\omega(kT) - e^{-aT}\omega((k-1)T) = KV(kT) + KV((k-1)T)$$

despejando la velocidad angular actual

$$\omega(kT) = e^{-aT}\omega((k-1)T) + KV(kT) + KV((k-1)T)$$

donde

$\omega(kT)$	es la velocidad angular actual
$\omega((k-1)T)$	es la velocidad angular anterior
$V(kT)$	es el voltaje actual
$V((k-1)T)$	es el voltaje anterior

a y b son constantes

$$a = \frac{b\,Ra + ka\,ke}{Ra\,J}$$

$$b = \frac{ka}{Ra\,J}$$

Para fines prácticos se utiliza la caracterización de los motores de corriente directa continuos:

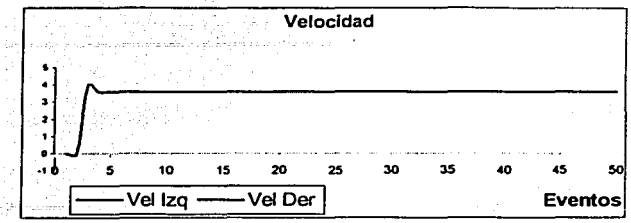
Constante Electromotriz	ke	1.23×10^{-3}	Vs/rad
Resistencia de Armadura	Ra	5.8	Ω
Inductancia	L	2.75×10^{-5}	H
Constante de Armadura	ka	9.43×10^{-3}	Nm/A
Constante del Generador	Keg	2.57×10^{-3}	Vs/rad
Fricción Viscosa	*b*	2.00×10^{-6}	Nms/rad
Inercia	J	3.28×10^{-6}	Kg m²

Resolviendo las ecuaciones para calcular a y b, nos dan como resultado:

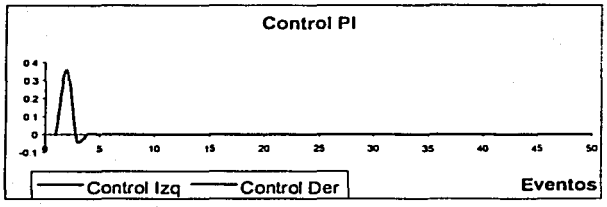
a = 7.65
b = 495.20
K = 7.65e-6

Las siguientes gráficas muestran la simulación del control PI propuesto:

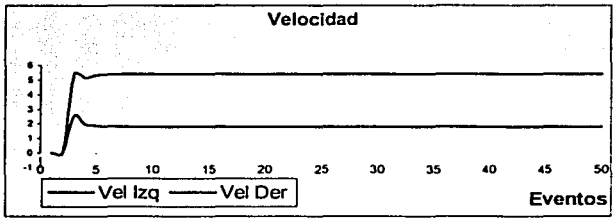
Gráf. 7.1 Control PI de velocidad con distintos grados de giro



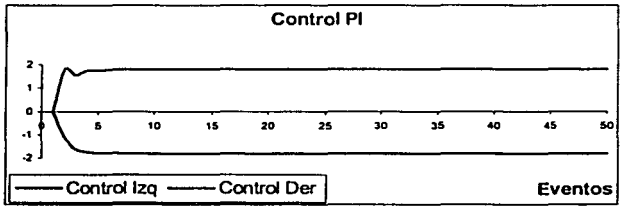
(a)



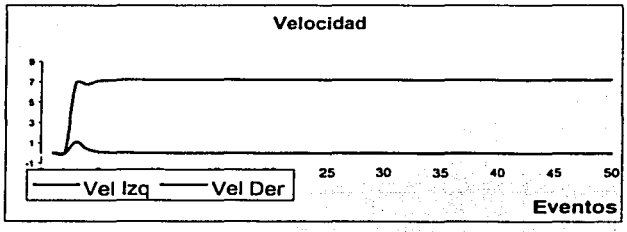
(b)



(c)

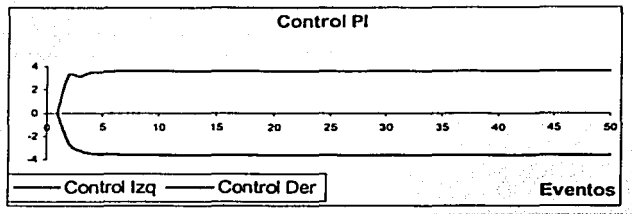


(d)

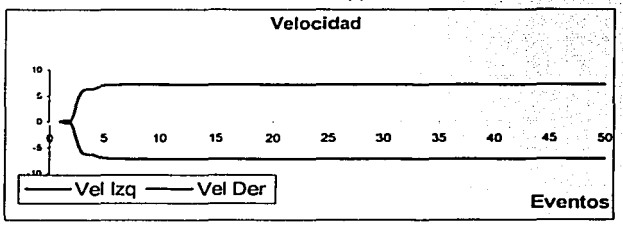


(e)

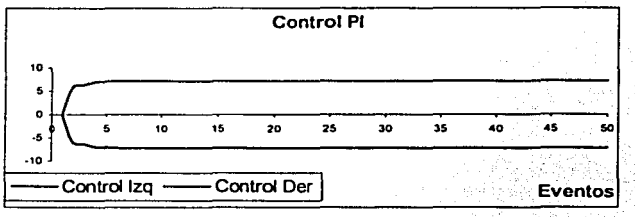
Gráf. 7.1 (continuación) Control PI de velocidad con distintos grados de giro



(f)



(g)



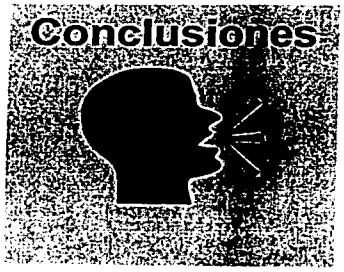
(h)

Las gráficas anteriores fueron obtenidas utilizando el control PI. Se propuso que la velocidad media de AGV fuese del 50% de la velocidad total de sus motores (1632T, ω = 69 rpm = 7.2 rad/s), por lo que la velocidad máxima permitida por rueda será de 7.2 rad/s y la velocidad menor de –7.2 rad/s.

En la Gráfica 7.1 a y b, se puede ver cómo la velocidad es igual en cada rueda debido a que el giro es cero, lo que quiere decir que el vehículo va en línea recta. En los incisos c y d, se le pide al control que simule un giro (giro = 3.6 rad/s) en el cual la velocidad de una de la llantas será 3 veces mayor que la otra, la velocidad del vehículo sigue siendo constante. En el inciso e y f, se realiza un giro (giro = 7.2 rad/s) tomando una de las ruedas como centro de giro. Los últimos dos incisos muestran el control de velocidad del vehículo utilizando como centro de giro la distancia media entre sus ruedas, por lo que cada rueda gira en sentido opuesto y a la misma velocidad angular.

Para programar el algoritmo de control en el microcontrolador, es necesario utilizar variables signadas y de punto flotante, por lo que se recomienda usar un lenguaje de alto nivel como C.

Una vez programado este algoritmo de control, bastará con introducir los datos cinemáticos propios del AGV con el que se esté trabajando. Se puede programar una secuencia de acciones como son los de generar trayectorias como por ejemplo: ochos, círculos, espirales, etc.



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Conclusiones

A lo largo de esta tesis se puede apreciar el desarrollo de diferentes elementos que son parte de un sistema mecatrónico, como son la tarjeta de desarrollo PIC 16F874, la tarjeta de puente H, la tarjeta de sensores ópticos, el AGV, desarrollo de software; cubriendo así los tres campos más amplios que abarca la mecatrónica (mecánica, informática y electrónica).

Se dieron a conocer conceptos útiles para comprender la información contenida en los capítulos e introducir al lector en el campo de la mecatrónica. También se indica, en algunos casos, en donde poder ampliar la información acerca del tema que se trata.

Algunas de las ventajas que incluye el presente trabajo son:

- Introducir a el lector en el estudio de los microcontroladores y presentar algunas de las opciones existentes en el mercado así como involucrarlo en los microcontroladores PIC.

- Facilitar la construcción de un sistema mecatrónico de una manera más sencilla al proponer diferentes tarjetas electrónicas:

    a) Una tarjeta con plataforma PIC, con entradas y salidas tanto analógicas como digitales, con programador integrado, comunicación a PC vía RS - 232, etc., la cual fue totalmente probada y funciona como se esperaba.

    b) Una tarjeta de circuito puente H, con la posibilidad de manejar motores DC tanto continuos como a pasos unipolares y bipolares; la cual también fue armada y probada dando excelentes resultados.

    c) Una tarjeta de entrada de sensores, la cual tiene capacidad para trabajar con 4 sensores ya sean analógicos y/o digitales. Esta tarjeta funciona correctamente también.

- Proponer el software con el cual la tarjeta PIC 16F874 puede funcionar íntegramente. El software es el MPLab, el CC5x y el IC - Prog. Cada uno de ellos se explica brevemente en el Capítulo 5. Se incluyen ejemplos de como se utilizan.

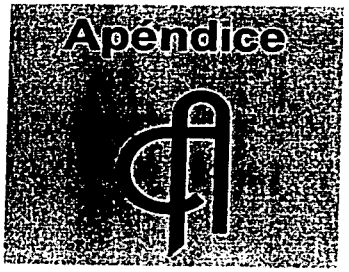
- El diseño del AGV propuesto, resta el problema de diseño de la plataforma de una aplicación móvil de este tipo. Es un diseño básico, sencillo, flexible y multiconfigurable.

Todo lo mencionado en los párrafos anteriores fue diseñado, armado y probado. Cabe mencionar que el CC5x está limitado a programas que no excedan de 1 kb de tamaño, la versión completa de este software no tiene esta restricción.

Uno de los inconvenientes encontrados en la tarjeta de desarrollo PIC 16F874 es el que no se puede tener conectados elementos a ésta cuando se esté programando, debido a que la corriente suministrada por el puerto serie de la computadora no tiene la suficiente corriente como para alimentar a otros elementos. Se probó su completa compatibilidad con los microcontroladores PIC 16C74, PIC16C77, PIC 16F874 y PIC 16F877. De la lista de microcontroladores compatibles con la tarjeta, no se probaron los PIC 18F448 y PIC 18F458, ya que no fue posible conseguirlos.

El control propuesto sirve para cualquier AGV que mantenga la configuración estructural propuesta, pero, la caracterización de los motores se deberá hacer siempre que estos sean cambiados por otros.

Los archivos de diseño de todos los elementos que contiene la tesis se encuentran en el archivo del Departamento de Mecatrónica para ser modificados en caso que ser necesario o ser requeridos por el lector.



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Descripción de las Instrucciones para el PIC 16F874

NEMÓNICOS	PARÁMETROS	OPERACIÓN	CICLOS	SEÑALIZADORES
INSTRUCCIONES QUE MANEJAN REGISTROS				
addwf	*f,d*	SUMA de W con f	1	C, DC, Z
andwf	*f,d*	AND de W con f	1	Z
clrf	*f*	BORRADO de f	1	Z
clrw	*æ*	BORRADO de W	1	Z
comf	*f,d*	COMPLEMENTO de f	1	Z
decf	*f,d*	DECREMENTO de f	1	Z
incf	*f,d*	INCREMENTO de f	1	Z
iorwf	*f, d*	OR de W con f	1	Z
movf	*f,d*	MOVIMIENTO de f	1	Z
movwf	*f*	MOVIMIENTO de W a f	1	
nop	*æ*	NO OPERACIÓN	1	
rlf	*f, d*	ROTACIÓN de f a izquierda con carry	1	C
rrf	*f, d*	ROTACIÓN de f a derecha con carry	1	C
subwf	*f, d*	RESTA de W a f (f - W)	1	C, DC, Z
swapf	*f, d*	INTERCAMBIO de 4 bits de más peso por los 4 de menos peso	1	
xorwf	*f, d*	OR exclusiva de W con f	1	Z
INSTRUCCIONES QUE MANEJAN BITS				
bcf	*f, b*	PUESTA a 0 del bit b de f	1	
bsf	*f, b*	PUESTA a 1 del bit b de f	1	
INSTRUCCIONES DE BRINCO				
btfsc	*f,b*	Prueba del bit b de f; BRINCO si 0	1 (2)	
btfss	*f,b*	Prueba del bit b de f; BRINCO si 1	1 (2)	
decfsz	*f,d*	DECREMENTO de f; BRINCO si 0	1 (2)	
incfsz	*f,d*	INCREMENTO de f; BRINCO si 0	1 (2)	
INSTRUCCIONES QUE MANEJAN OPERANDOS INMEDIATOS				
addlw	*k*	SUMA de literal con W	1	C, DC, Z
andlw	*k*	AND de literal con W	1	Z
iorlw	*k*	OR de la literal con W	1	Z
movlw	*k*	MOVIMIENTO de literal a W	1	
sublw	*k*	RESTA W de literal (k - W)	1	C, DC, Z
xorlw	*k*	OR exclusiva de literal con W	1	Z
INSTRUCCIONES DE CONTROL Y ESPECIALES				
call	*k*	LLAMADA a subrutina	2	
clrwdt		BORRADO de WDT	1	#TO, #PD
goto	*k*	SALTO a una dirección	2	
retfie		RETORNO de interrupción	2	
retlw	*k*	RETORNO devolviendo literal en W	2	
return		RETORNO de subrutina	2	
sleep		PUESTA del microprocesador en reposo	1	#TO, #PD

ADDLW **Add Literal and W**
(Suma la Literal con W)

Sintaxis:	[etiqueta] *addlw* k
Operandos:	$0 \leq k \leq 255$
Operación:	(W) + k → (W)
Estado Afectado:	C, CD, Z
Descripción:	El contenido del registro W es sumado a los 8 bits de la literal "k" y el resultado colocado en el registro W.

ADDWF **Add W and f**
(Suma a W y f)

Sintaxis:	[etiqueta] *addwf* f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0,1]$
Operación:	(W) + (f) → (destino)
Estado Afectado:	C, CD, Z
Descripción:	Suma el contenido del registro W con el registro 'f' . Si 'd' es 0, el resultado es guardado en el registro W. Si 'd' es 1, el resultado es devuelto en el registro 'f'.

ANDLW **And Literal with W**
(Operación And de la Literal con W)

Sintaxis:	[etiqueta] *andlw* k
Operandos:	$0 \leq k \leq 255$
Operación:	(W) .AND. (k) → (W)
Estado Afectado:	Z
Descripción:	El contenido del registro W y la literal de 8 bits 'k' son operados con AND. El resultado es depositado en el registro W.

ANDWF. **And W with F**
(Operación And de W con f)

Sintaxis:	[etiqueta] *andwf* f, d
Operandos:	$0 \leq k \leq 127$ $d \in [0,1]$
Operación:	(W) .AND. (f) → (destino)
Estado Afectado:	Z
Descripción:	El contenido del registro W y el registro 'f' son operados con AND. Si 'd' es 0, el resultado es guardado en el registro W. Si 'd' es 1, el resultado es devuelto en el registro 'f'.

BCF **Bit Clear f**
(Pone a 0 los Bits del Registro f)

Sintaxis:	[etiqueta] *bcf* f, b
Operandos:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operación:	0 → (f<b>)
Estado Afectado:	Ninguno
Descripción:	El bit 'b' en el registro 'f' se pone a 0.

BSF **Bit Set f**
(Pone a 1 Bits del Registro f)

Sintaxis:	[etiqueta] *bsf* f, b
Operandos:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operación:	1 → (f<b>)
Estado Afectado:	Ninguno
Descripción:	El bit 'b' en el registro 'f' es puesto a 1.

BTFSS **Bit Test f, Skip if Set**
(Prueba un Bit, si es 1 Salta)

Sintaxis:	[etiqueta] *btfss* f, b
Operandos:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operación:	1 → (f<b>) = 1
Estado Afectado:	Ninguno
Descripción:	Si el bit 'b' en el registro 'f' es 0, la siguiente instrucción es ejecutada. Si el bit 'b' es 1, la siguiente instrucción es descartada y una NOP es ejecutada en su lugar, haciéndola una instrucción de 2 ciclos.

BTFSC **Bit Test f, Skip if Clear**
(Prueba un Bit, si es 0 Salta)

Sintaxis:	[etiqueta] *btfsc* f, b
Operandos:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operación:	1 → (f<b>) = 0
Estado Afectado:	Ninguno
Descripción:	Si el bit 'b' en el registro 'f' es 1, la siguiente instrucción es ejecutada, si es 0 la siguiente instrucción es descartada y una NOP es ejecutada en su lugar. Es una instrucción de 2 ciclos.

BCF **Bit Clear f**
(Pone a 0 los Bits del Registro f)

Sintaxis:	[etiqueta] *bcf* f, b
Operandos:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operación:	0 → (f<b>)
Estado Afectado:	Ninguno
Descripción:	El bit 'b' en el registro 'f' se pone a 0.

BSF **Bit Set f**
(Pone a 1 Bits del Registro f)

Sintaxis:	[etiqueta] *bsf* f, b
Operandos:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operación:	1 → (f<b>)
Estado Afectado:	Ninguno
Descripción:	El bit 'b' en el registro 'f' es puesto a 1.

BTFSS **Bit Test f, Skip if Set**
(Prueba un Bit, si es 1 Salta)

Sintaxis:	[etiqueta] *btfss* f, b
Operandos:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operación:	1 → (f<b>) = 1
Estado Afectado:	Ninguno
Descripción:	Si el bit 'b' en el registro 'f' es 0, la siguiente instrucción es ejecutada. Si el bit 'b' es 1, la siguiente instrucción es descartada y una NOP es ejecutada en su lugar, haciéndola una instrucción de 2 ciclos.

BTFSC **Bit Test f, Skip if Clear**
(Prueba un Bit, si es 0 Salta)

Sintaxis:	[etiqueta] *btfsc* f, b
Operandos:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operación:	1 → (f<b>) = 0
Estado Afectado:	Ninguno
Descripción:	Si el bit 'b' en el registro 'f' es 1, la siguiente instrucción es ejecutada, si es 0 la siguiente instrucción es descartada y una NOP es ejecutada en su lugar. Es una instrucción de 2 ciclos.

CALL **Call Subroutine**
(Llamado a Subrutina)

Sintaxis:	[etiqueta] *call* k
Operandos:	0 ≤ k ≤ 2047
Operación:	(PC) + 1 → TOS k → PC<10:0> (PCLATH <4:3>) → PC<12:11>
Estado Afectado:	Ninguno
Descripción:	Llamado de subrutina. La dirección para el regreso es puesto en la pila (PC + 1), el onceavo bit de la siguiente dirección es cargado en el PC en los bits <10:0>. La parte alta del PC son cargados del PCLATH. *Call* es una instrucción de dos ciclos.

CLRF **Clear f**
(Limpiado de f)

Sintaxis:	[etiqueta] *clrf* f
Operandos:	0 ≤ f ≤ 127
Operación:	00h → (f) 1 → Z
Estado Afectado:	Z
Descripción:	El contenido del registro 'f' es puesto a 0 y el bit Z es puesto a 1.

CLRW **Clear W**
(Limpiado de W)

Sintaxis:	[etiqueta] *clrw*
Operandos:	None
Operación:	00h → (W) 1 → Z
Estado Afectado:	Z
Descripción:	El contenido del registro 'W' es puesto a 0 y el bit Z es puesto a 1.

CLRWDT **Clear Watchdog Timer**
(Limpia el Perro Guardián)

Sintaxis:	[etiqueta] *clrwdt*
Operandos:	None
Operación:	00h → WDT 1 → Z
Estado Afectado:	Z
Descripción:	El contenido del registro 'W' es puesto a 0 y el bit Z es 1.

COMF **Complement f**
(Complemento de f)

Sintaxis:	[etiqueta] *comf* f, d
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	(f) → (destino)
Estado Afectado:	Z
Descripción:	El contenido del registro 'f' es complementado. Si 'd' es 0, el resultado es guardado en W. Si d es 1, el resultado es devuelto al registro 'f'.

DECF **Decrement f**
(Decremento de f)

Sintaxis:	[etiqueta] *decf* f, d
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	(f) – 1 → (destino)
Estado Afectado:	Z
Descripción:	Decrementa al registro 'f'. Si 'd' es 0, el resultado es guardado en W. Si d es 1, el resultado es devuelto al registro 'f'.

DECFSZ **Decrement f, Skip if 0**
(Decrementa a f, Si es 0, Salta)

Sintaxis:	[etiqueta] *decfsz* f, d
Operandos:	0 ≤ f ≤ 127 d ∈ [0,1]
Operación:	(f) – 1 → (destino) salta si el resultado es = 0
Estado Afectado:	Ninguno
Descripción:	El contenido del registro 'f' es decrementado. Si 'd' es 0, el resultado es guardado en W. Si 'd' es 1, el resultado es devuelto al registro 'f'. Si el resultado es 1, la siguiente instrucción es ejecutada. Si el resultado es 0, entonces una instrucción NOP es ejecutada en su lugar, haciéndola una instrucción de 2 ciclos.

GOTO **Unconditional Branch**
(Bifurcación Incondicional)

Sintaxis:	[etiqueta] *goto* k
Operandos:	0 ≤ k ≤ 2047
Operación:	k → PC<10:0> PCLATH <4:3> → PC<12:11>
Estado Afectado:	Ninguno

Descripción:	*Goto* es una bifurcación incondicional. El valor del onceavo bit inmediato es cargado en los bits <10:0> del PC. Los bits de la parte alta del PC son cargados del PCLATH <4:3> . *Goto* es una instrucción de dos ciclos.

INCF **Increment f**
(Incremento de f)

Sintaxis:	[etiqueta] *incf* f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0,1]$
Operación:	(f) + 1 → (destino)
Estado Afectado:	Z
Descripción:	El contenido del registro 'f' es incrementado. Si 'd' es 0, el resultado es guardado en el registro W. Si 'd' es 1, el resultado es devuelto al registro 'f'.

INCFSZ **Increment f, Skip if 0**
(Incrementa a f, Salta si es 0)

Sintaxis:	[etiqueta] *incfsz* f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0,1]$
Operación:	(f) + 1 → (destino) salta si el resultado es = 0
Estado Afectado:	Ninguno
Descripción:	El contenido del registro 'f' es incrementado. Si 'd' es 0, el resultado es guardado en W. Si 'd' es 1, el resultado es devuelto al registro 'f'. Si el resultado es 1, la siguiente instrucción es ejecutada. Si el resultado es 0, entonces una instrucción NOP es ejecutada en su lugar, haciéndola una instrucción de 2 ciclos.

IORLW **Inclusive OR Literal with W**
(OR Inclusiva de la Literal con W)

Sintaxis:	[etiqueta] *iorlw* k
Operandos:	$0 \leq k \leq 255$
Operación:	(W) .OR. (k) → (W)
Estado Afectado:	Z
Descripción:	El contenido del registro W y la literal de 8 bits 'k' son operados con OR. El resultado de la operación es depositado en el registro W.

IORWF **Inclusive OR W with f**
(OR Inclusiva entre W y f)

Sintaxis:	[etiqueta] *iorwf* f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0,1]$
Operación:	(W) .OR. (f) → (destino)
Estado Afectado:	Z
Descripción:	El contenido del registro W y el registro 'f' son operados con OR. Si 'd' es 0, el resultado es guardado en el registro W. Si 'd' es 1, el resultado es devuelto en el registro 'f'.

MOVF **Move f**
(Mueve a f)

Sintaxis:	[etiqueta] *movf* f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0,1]$
Operación:	(f) → (destino)
Estado Afectado:	Z
Descripción:	El contenido del registro 'f' es movido al destino dependiendo del registro de estado de 'd'. Si 'd' es 0, el destino es el registro W. Si 'd' es 1, el destino es el registro 'f' mismo.

MOVLW **Move Literal to W**
(Mueve la Literal a W)

Sintaxis:	[etiqueta] *movlw* k
Operandos:	$0 \leq k \leq 255$
Operación:	K → (W)
Estado Afectado:	Ninguno
Descripción:	La literal 'k' (de 8 bits) es cargada dentro del registro W. El estado don't care será reconocido como 0.

MOVWF **Move W to f**
(Mueve W a f)

Sintaxis:	[etiqueta] *movwf* f
Operandos:	$0 \leq f \leq 127$
Operación:	(W) → (f)
Estado Afectado:	Ninguno
Sintaxis:	Mueve el dato del registro W al registro 'f'.

NOP **No Operation**
(Ninguna Operación)

Sintaxis:	[etiqueta] *nop*
Operandos:	Ninguno
Operación:	Ninguna Operación
Estado Afectado:	Ninguno
Descripción:	No se realiza ninguna operación, se deja perder un ciclo de instrucción.

RETFIE **Return from Interruption**
(Regreso de Interrupción)

Sintaxis:	[etiqueta] *retfie*
Operandos:	Ninguno
Operación:	TOS → PC 1 → GIE
Estado Afectado:	Ninguno
Descripción:	Indica el fin de la atención a una interrupción.

RETLW ***Return*** **with Literal in W**
(Regreso con una Literal en W)

Sintaxis:	[etiqueta] *retlw* k
Operandos:	Ninguno
Operación:	k → (W) TOS → PC
Estado Afectado:	Ninguno
Descripción:	El registro W es cargado con una literal 'k' de 8 bits. El PC es cargado en la cima de la Pila (la dirección de regreso). Esta es una instrucción de dos ciclos.

RETURN **Return from Subroutine**
(Regreso de Subrutina)

Sintaxis:	[etiqueta] *return*
Operandos:	Ninguno
Operación:	TOS → PC
Estado Afectado:	Ninguno
Descripción:	Regreso de subrutina. La cima de la Pila es cargada dentro del PC. Esta instrucción indica el retorno de un llamado a subrutina. Esta instrucción es de dos ciclos.

RLF **Rotate Left f Through Carry**
(Rota f a la Izquierda hasta el Acarreo)

Sintaxis:	[etiqueta] *rlf* f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0,1]$
Operación:	Movimiento de datos
Estado Afectado:	C
Descripción:	Los datos contenido en el registro 'f' son rotados un bit a la izquierda hasta la bandera del acarreo. Si 'd' es 0, el resultado es puesto en el registro W, si es 1, el resultado es guardado de regreso en 'f'

RRF **Rotate Righ f Through Carry**
(Rota f a la Derecha hasta el Acarreo)

Sintaxis:	[etiqueta] *rrf* f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0,1]$
Operación:	Movimiento de datos
Estado Afectado:	C
Descripción:	Los datos contenido en el registro 'f' son rotados un bit a la derecha hasta la bandera del acarreo. Si 'd' es 0, el resultado es puesto en el registro W, si es 1, el resultado es guardado de regreso en 'f'

SLEEP **Sleep**
(Dormir)

Sintaxis:	[etiqueta] *sleep*
Operandos:	Ninguno
Operación:	00h → WDT 0 → WDT pre escala 1 → TO 0 → PD
Estado Afectado:	TO, PD
Descripción:	El bit PD es borrado. El bit TO se pone en 1. El perro guardián y la preescala son borrados. El procesador es puesto a modo dormir con el oscilador detenido.

SUBLW **Subtract W from Literal**
(Resta W de una literal)

Sintaxis:	[etiqueta] *sublw* k
Operandos:	$0 \leq k \leq 255$
Operación:	K – (W) → (W)
Estado Afectado:	C, DC, Z
Descripción:	El registro W es restado (método de complemento a 2) de la literal 'k' de ocho bits. El resultado es puesto en el registro W.

SUBWF **Subtract W from f**
(Resta W de f)

Sintaxis:	[etiqueta] *subwf* k
Operandos:	$0 \leq f \leq 127$ $d \in [0,1]$
Operación:	(f) – (W) → (destino)
Estado Afectado:	C, DC, Z
Descripción:	El registro W es restado (método de complemento a 2) del registro 'f'. Si 'd' es 0, el resultado es puesto en el registro W. Si 'd' es 1, es devuelto al registro 'f'.

SWAPF **Swap Nibbles in f**
(Alterna 4 bits en f)

Sintaxis:	[etiqueta] *swapf* f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0,1]$
Operación:	(f<3:0>) → (destino <7:4>) (f<7:4 → (destino <3:0>)
Estado Afectado:	Ninguno
Descripción:	La parte alta y baja de 4 bits del registro 'f' son intercambiados. Si 'd' es 0, el resultado es puesto en el registro W. Si 'd' es 1, el resultado es puesto en el registro 'f'.

XORLW **Exclusive OR Literal with W**
(OR Exclusiva de la Literal con W)

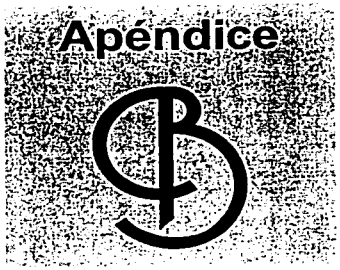
Sintaxis:	[etiqueta] *xorlw* k
Operandos:	$0 \leq k \leq 255$
Operación:	(W) .XOR. (k) → (W)
Estado Afectado:	Z
Descripción:	El contenido del registro W y la literal de 8 bits 'k' son operados con XOR. El resultado es depositado en el registro W.

XORWF **Exclusive OR W with f**
(OR Exclusiva entre W y f)

Sintaxis:	[etiqueta] *xorwf* f, d
Operandos:	$0 \leq f \leq 127$ $d \in [0,1]$
Operación:	(W) .XOR. (f) → (destino)
Estado Afectado:	Z
Descripción:	El contenido del registro W y el registro 'f' son operados con XOR. Si 'd' es 0, el resultado es guardado en el registro W. Si 'd' es 1, el resultado es devuelto en el registro 'f'.

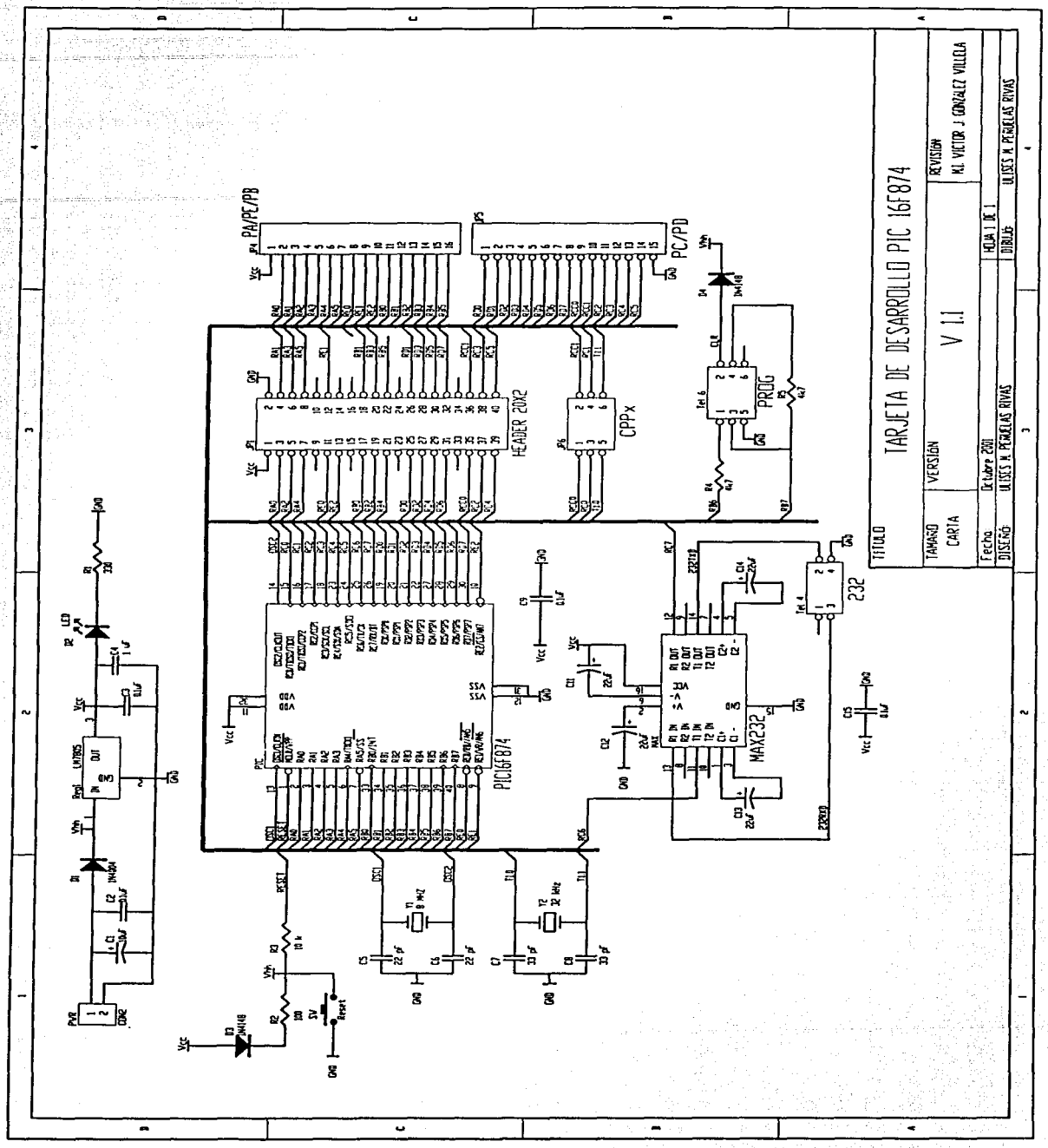
Nomenclatura:

Campo	Descripción
f	Dirección del Registro (0x00 a 0x7F)
w	Acumulador
b	Dirección de 1 bit dentro de un registro de 8 bits
k	Campo de literal, constante o etiqueta
x	Don't care (0 o 1), no importa
d	Destino; d = 0 se carga a W, d = 1, se carga a f Default d = 1
Etiqueta	Nombre de la etiqueta
[ ]	Opciones
( )	Contenido
→	Asignado a
< >	Campo del registro
∈	Elemento de...
cursivas	Instrucción



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Layout de la Tarjeta de Desarrollo PIC 16F874


TARJETA DE DESARROLLO PIC 16F874
V 1.1
TAMAÑO
CARTA
VERSIÓN
Fecha
Octubre 2001
DISEÑO
ULISES M. PEÑUELAS RIVAS
REVISIÓN
M.I. VICTOR J. GONZÁLEZ VILLELA
HOJA 1 DE 1
DIBUJO
ULISES M. PEÑUELAS RIVAS
PA/PE/PB
PC/PD
HEADER 20X2
CPPx
PROG
PIC16F874
MAX232
232

Tarjeta de Desarrollo PIC 16F874

Cantidad	Designador	Parte	Descripción
1	C1	10uF	Capacitor Electrolítico
5	C11-15	22uF	Capacitor Electrolítico
2	C2-3	0.1uF	Capacitor Cerámico
1	C4	1 uF	Capacitor Tantalio
2	C9,15	0.1uF	Capacitor Tantalio
2	C5-6	22 pF	Capacitor Cerámico
2	C7-8	33 pF	Capacitor Cerámico
1	D1	1N4004	Diodo
1	D2	LED	LED de 3 mm
2	D3-4	1N4148	Diodo
1	DIP 16	Base	Base para Chip DIP 16
1	DIP 40	Base	Base para Chip DIP 40
1	JP1	Header 2X20	Header doble de 2x20
1	JP4	Header 1x16	Header Horizontal de 1x16
1	JP5	Header 1x15	Header Horizontal de 1x15
1	JP6	Header 2x3	Header doble de 2x3
1	MAX	MAX232	Max232
1	PIC	PIC16F874	Microcontrolador PIC 16F874
1	PWR	CON2	Conector de dos líneas
1	R1	330	Resistencia de 1/4 W
1	R2	100	Resistencia de 1/4 W
1	R3	10 k	Resistencia de 1/4 W
2	R4-5	4k7	Resistencia de 1/4 W
1	Reg1	LM7805	Regulador LM 7805
1	SW	Botón	Minipush Botón
1	Tel 4	232	Jack Telefónico de 4 hilos
1	Tel 6	PROG	Jack Telefónico de 6 hilos
1	Y1	8 MHZ	Cristal
1	Y2	32 kHz	Cristal
2	-	Jumper	Jumper

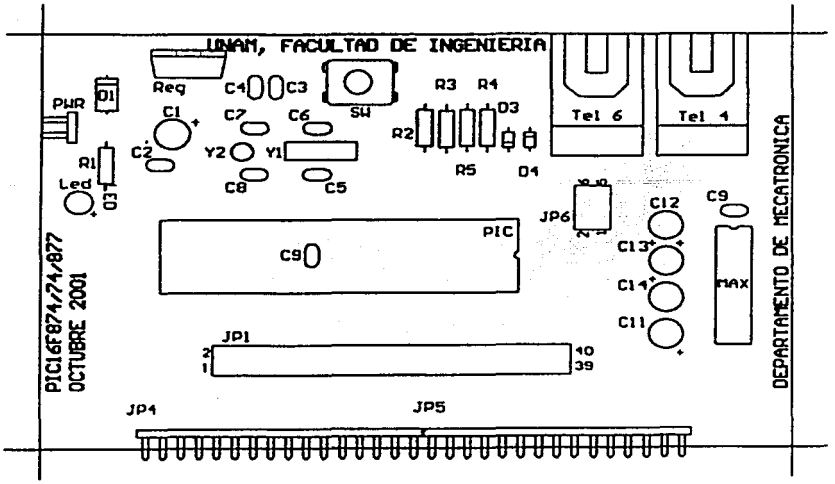


Fig. B.1
Diagrama de componentes de la tarjeta de desarrollo PIC 16F874

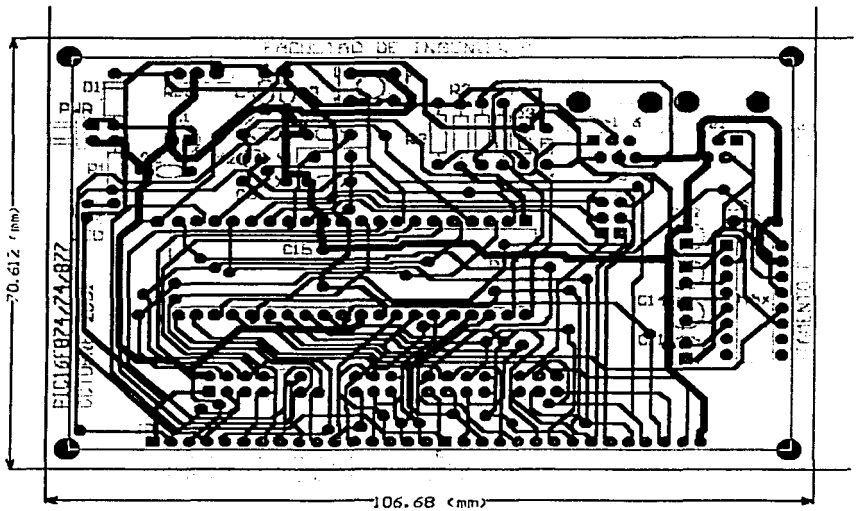


Fig. B.2
Digrama compuesto de tarjeta de la tarjeta de desarrollo PIC 16F874

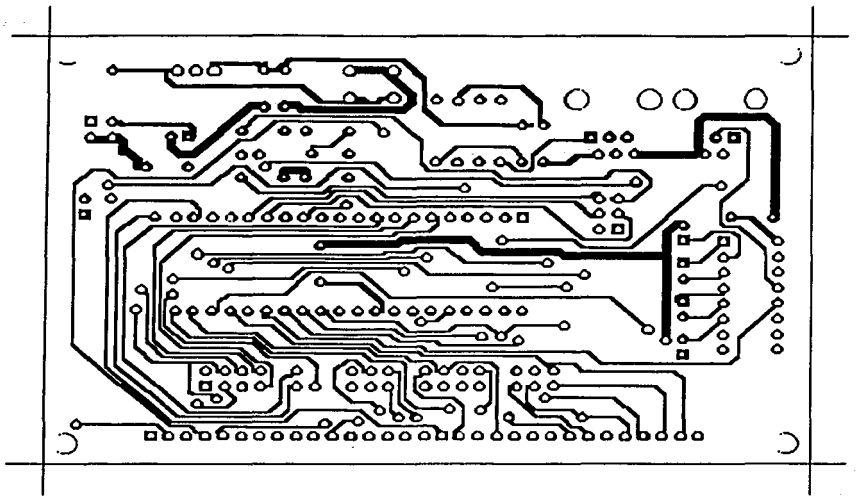


Fig. B.3
Diagrama de pistas superiores de la tarjeta de desarrollo PIC 16F874

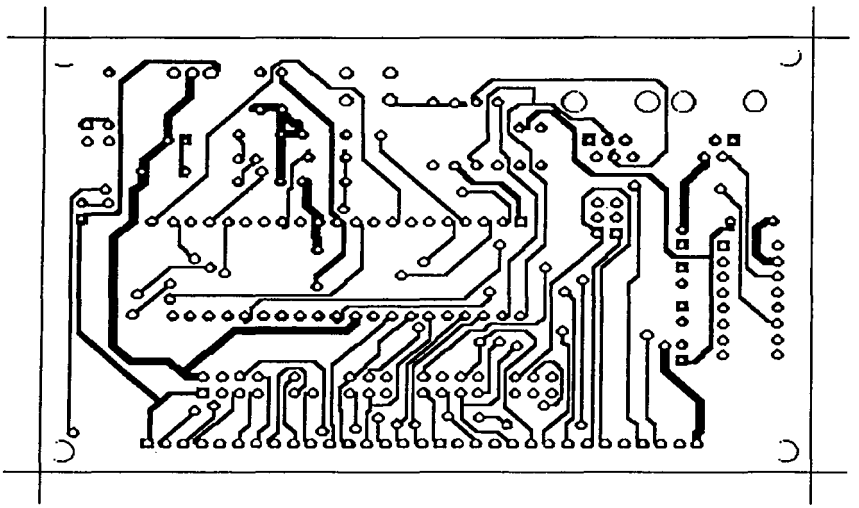
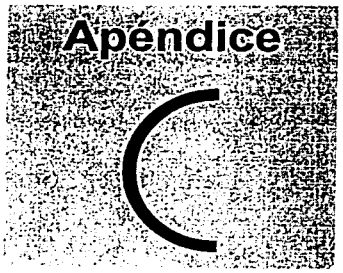


Fig. B.4
Diagrama de pistas inferior de la tarjeta de desarrollo PIC 16F874



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Layout de la Tarjeta Puente H

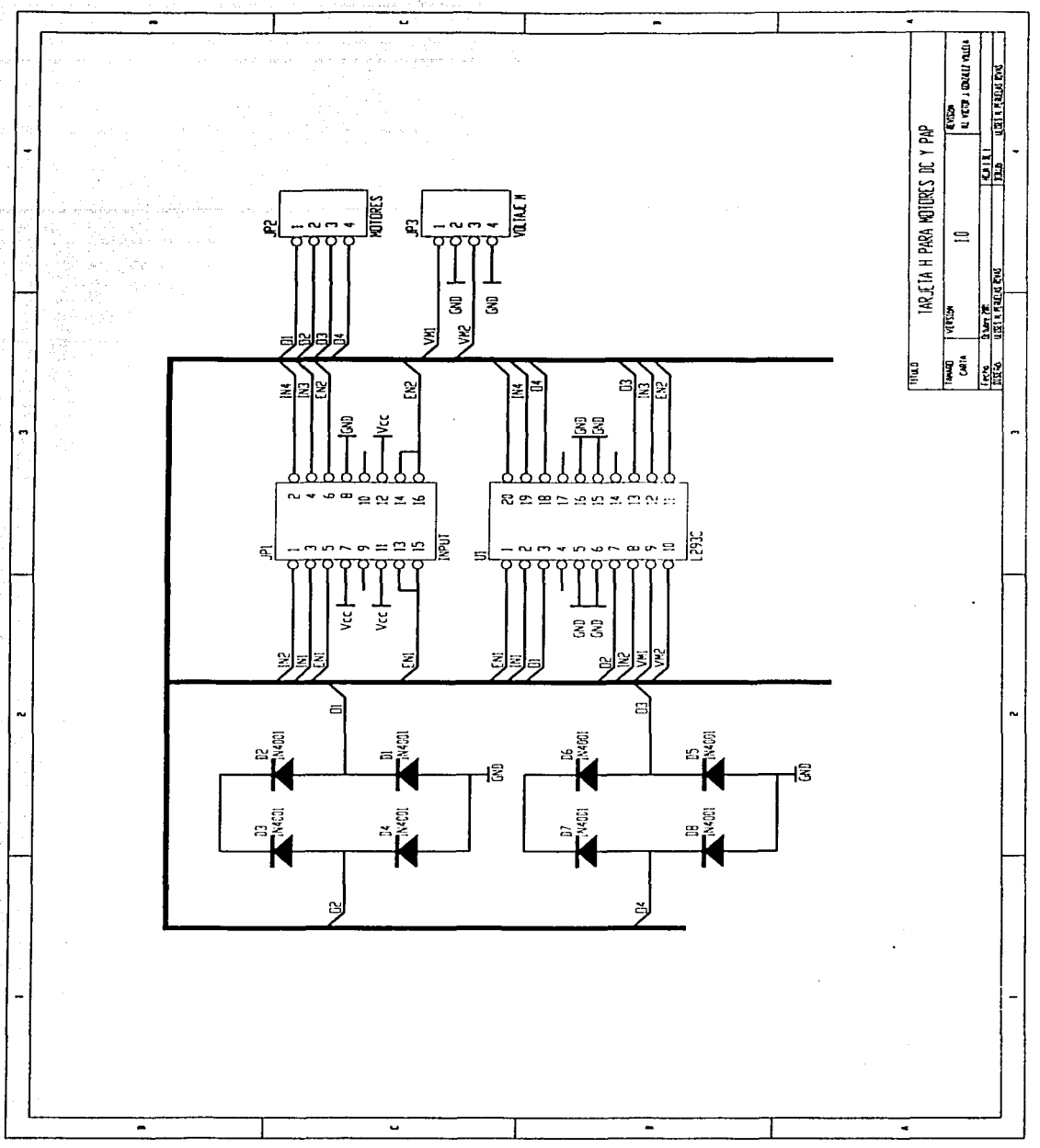

TARJETA H PARA MOTORES DC Y PAP
MOTORES
VOLTAJE M
L293C
INPUT
1N4001

Tabla C.1
Lista de materiales

Cantidad	Parte	Designador	Descripción
8	1N4001	D1-8	Diodo
1	L293C	U1	Circuito Puente H L293C
1	Header	JP1	Header Horizontal 2x8
4	Conector	JP2, JP3	Conector de Tornillo 1x2
1	Base	DIP 20	Base para Chip DIP 20
2	Jumper	-	Jumper

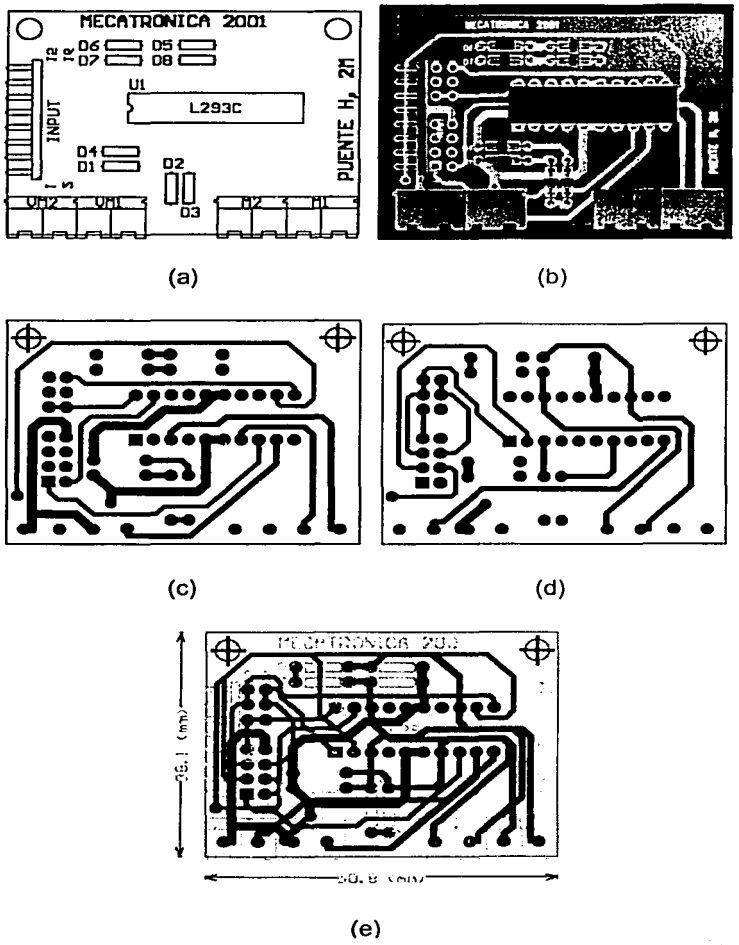
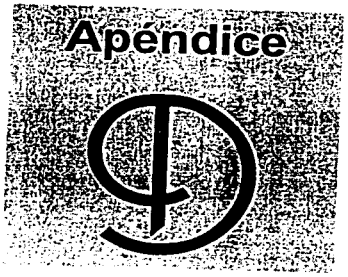


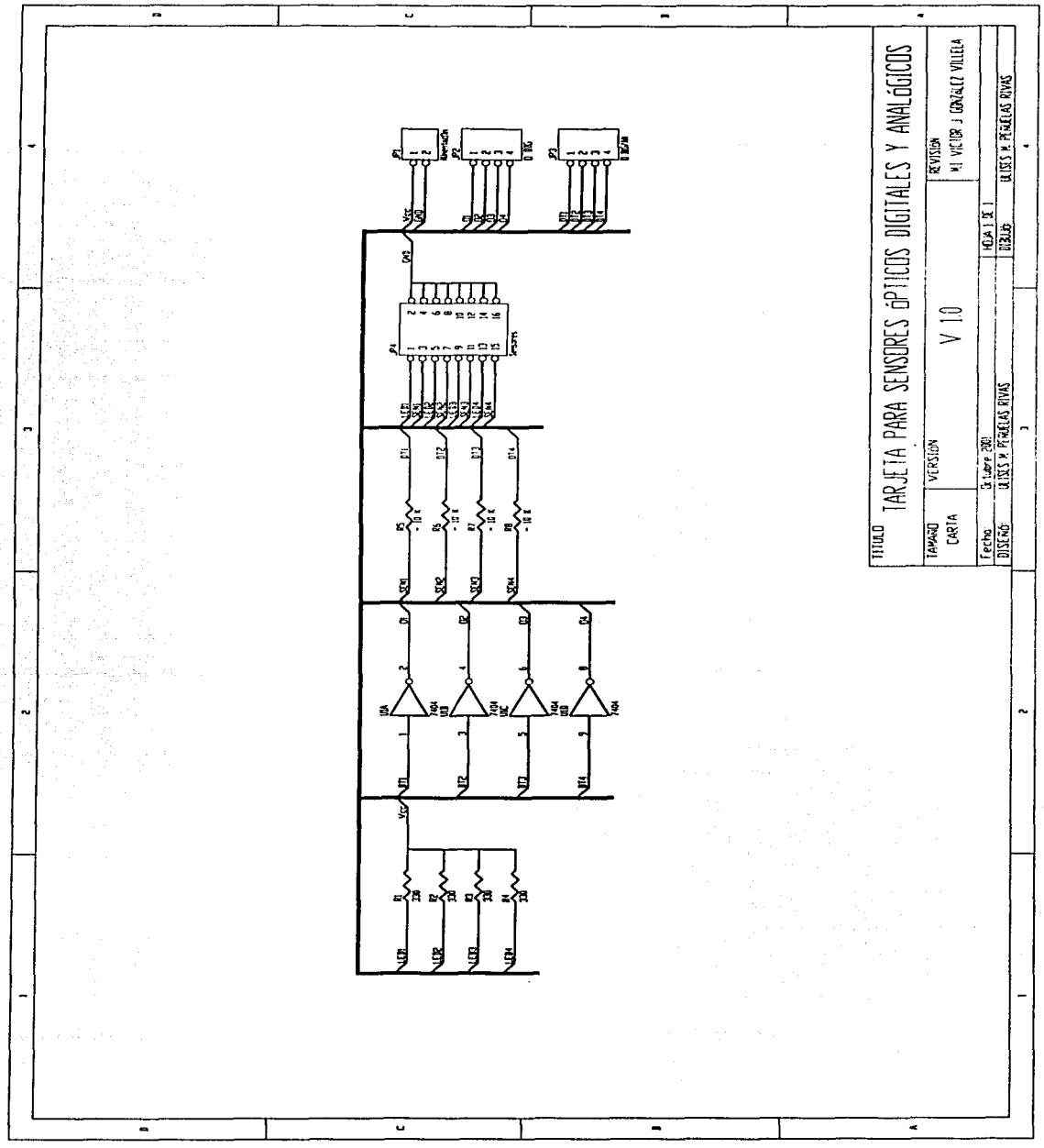
Fig. C.1
Tarjeta de circuito puente H L293C.
a) Diagrama de componentes
b) Diagrama de tarjeta en 3D
c) Diagrama de pistas superiores
d) Diagrama de pistas inferior
e) Diagrama compuesto



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Layout de la Tarjeta Sensores Ópticos

Tarjeta de Sensores Ópticos Analógicos y/o Digitales



Cantidad	Parte	Designador	Descripción
4	Resistencia	R1-4	Resistencias de ¼ W
4	Resistencia	R5-8	Resistencias de ¼ W
1	Conector	JP1	Conector de tornillo 1x2
2	Header	JP2, JP3	Header de 1x4
1	Header	JP4	Header de 2x8
1	Base	DIP 14	Base para Chip DIP 14

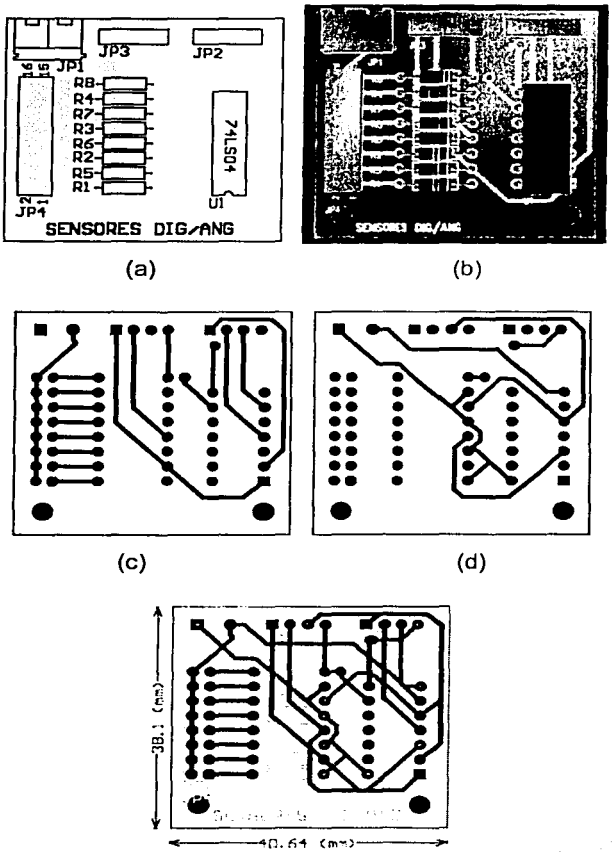
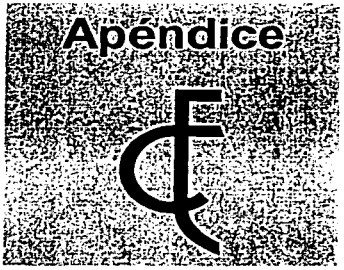
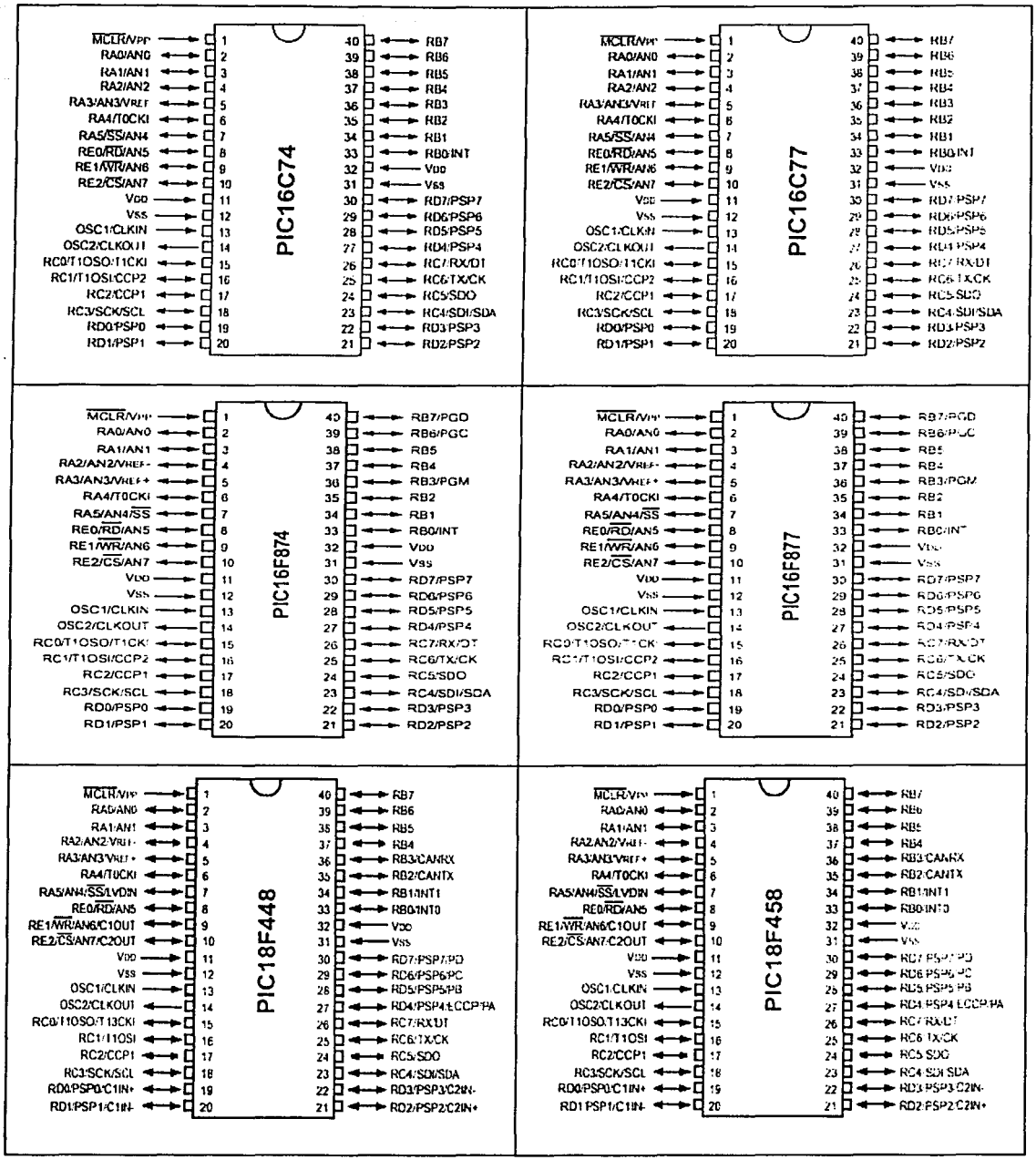


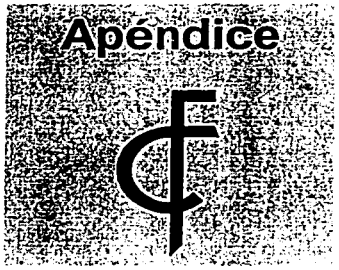
Fig. D.1
Tarjeta de circuito de sensores ópticos analógicos y/o digitales.
a) Diagrama de componentes
b) Diagrama de tarjeta en 3D
c) Diagrama de pistas superiores
d) Diagrama de pistas inferior
e) Diagrama compuesto



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

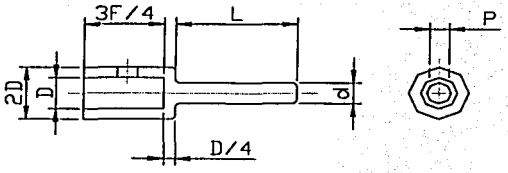
Micrcontroladores PIC Compatibles con la Tarjeta de Desarrollo PIC 16F874


PIC16C74
MCLR/VPP 1
RA0/AN0 2
RA1/AN1 3
RA2/AN2 4
RA3/AN3/VREF 5
RA4/T0CKI 6
RA5/SS/AN4 7
RE0/RD/AN5 8
RE1/WR/AN6 9
RE2/CS/AN7 10
VDD 11
VSS 12
OSC1/CLKIN 13
OSC2/CLKOUT 14
RC0/T1OSO/T1CKI 15
RC1/T1OSI/CCP2 16
RC2/CCP1 17
RC3/SCK/SCL 18
RD0/PSP0 19
RD1/PSP1 20
40 RB7
39 RB6
38 RB5
37 RB4
36 RB3
35 RB2
34 RB1
33 RB0/INT
32 VDD
31 VSS
30 RD7/PSP7
29 RD6/PSP6
28 RD5/PSP5
27 RD4/PSP4
26 RC7/RX/DT
25 RC6/TX/CK
24 RC5/SDO
23 RC4/SDI/SDA
22 RD3/PSP3
21 RD2/PSP2
PIC16C77
MCLR/VPP 1
RA0/AN0 2
RA1/AN1 3
RA2/AN2 4
RA3/AN3/VREF 5
RA4/T0CKI 6
RA5/SS/AN4 7
RE0/RD/AN5 8
RE1/WR/AN6 9
RE2/CS/AN7 10
VDD 11
VSS 12
OSC1/CLKIN 13
OSC2/CLKOUT 14
RC0/T1OSO/T1CKI 15
RC1/T1OSI/CCP2 16
RC2/CCP1 17
RC3/SCK/SCL 18
RD0/PSP0 19
RD1/PSP1 20
40 RB7
39 RB6
38 RB5
37 RB4
36 RB3
35 RB2
34 RB1
33 RB0/INT
32 VDD
31 VSS
30 RD7/PSP7
29 RD6/PSP6
28 RD5/PSP5
27 RD4/PSP4
26 RC7/RX/DT
25 RC6/TX/CK
24 RC5/SDO
23 RC4/SDI/SDA
22 RD3/PSP3
21 RD2/PSP2
PIC16F874
MCLR/VPP 1
RA0/AN0 2
RA1/AN1 3
RA2/AN2/VREF- 4
RA3/AN3/VREF+ 5
RA4/T0CKI 6
RA5/AN4/SS 7
RE0/RD/AN5 8
RE1/WR/AN6 9
RE2/CS/AN7 10
VDD 11
VSS 12
OSC1/CLKIN 13
OSC2/CLKOUT 14
RC0/T1OSO/T1CKI 15
RC1/T1OSI/CCP2 16
RC2/CCP1 17
RC3/SCK/SCL 18
RD0/PSP0 19
RD1/PSP1 20
40 RB7/PGD
39 RB6/PGC
38 RB5
37 RB4
36 RB3/PGM
35 RB2
34 RB1
33 RB0/INT
32 VDD
31 VSS
30 RD7/PSP7
29 RD6/PSP6
28 RD5/PSP5
27 RD4/PSP4
26 RC7/RX/DT
25 RC6/TX/CK
24 RC5/SDO
23 RC4/SDI/SDA
22 RD3/PSP3
21 RD2/PSP2
PIC16F877
MCLR/VPP 1
RA0/AN0 2
RA1/AN1 3
RA2/AN2/VREF- 4
RA3/AN3/VREF+ 5
RA4/T0CKI 6
RA5/AN4/SS 7
RE0/RD/AN5 8
RE1/WR/AN6 9
RE2/CS/AN7 10
VDD 11
VSS 12
OSC1/CLKIN 13
OSC2/CLKOUT 14
RC0/T1OSO/T1CKI 15
RC1/T1OSI/CCP2 16
RC2/CCP1 17
RC3/SCK/SCL 18
RD0/PSP0 19
RD1/PSP1 20
40 RB7/PGD
39 RB6/PGC
38 RB5
37 RB4
36 RB3/PGM
35 RB2
34 RB1
33 RB0/INT
32 VDD
31 VSS
30 RD7/PSP7
29 RD6/PSP6
28 RD5/PSP5
27 RD4/PSP4
26 RC7/RX/DT
25 RC6/TX/CK
24 RC5/SDO
23 RC4/SDI/SDA
22 RD3/PSP3
21 RD2/PSP2
PIC18F448
MCLR/VPP 1
RA0/AN0 2
RA1/AN1 3
RA2/AN2/VREF- 4
RA3/AN3/VREF+ 5
RA4/T0CKI 6
RA5/AN4/SS/LVDIN 7
RE0/RD/AN5 8
RE1/WR/AN6/C1OUT 9
RE2/CS/AN7/C2OUT 10
VDD 11
VSS 12
OSC1/CLKIN 13
OSC2/CLKOUT 14
RC0/T1OSO/T13CKI 15
RC1/T1OSI 16
RC2/CCP1 17
RC3/SCK/SCL 18
RD0/PSP0/C1IN+ 19
RD1/PSP1/C1IN- 20
40 RB7
39 RB6
38 RB5
37 RB4
36 RB3/CANRX
35 RB2/CANTX
34 RB1/INT1
33 RB0/INT0
32 VDD
31 VSS
30 RD7/PSP7/PD
29 RD6/PSP6/PC
28 RD5/PSP5/PB
27 RD4/PSP4/ECCP1/PA
26 RC7/RX/DT
25 RC6/TX/CK
24 RC5/SDO
23 RC4/SDI/SDA
22 RD3/PSP3/C2IN-
21 RD2/PSP2/C2IN+
PIC18F458
MCLR/VPP 1
RA0/AN0 2
RA1/AN1 3
RA2/AN2/VREF- 4
RA3/AN3/VREF+ 5
RA4/T0CKI 6
RA5/AN4/SS/LVDIN 7
RE0/RD/AN5 8
RE1/WR/AN6/C1OUT 9
RE2/CS/AN7/C2OUT 10
VDD 11
VSS 12
OSC1/CLKIN 13
OSC2/CLKOUT 14
RC0/T1OSO/T13CKI 15
RC1/T1OSI 16
RC2/CCP1 17
RC3/SCK/SCL 18
RD0/PSP0/C1IN+ 19
RD1/PSP1/C1IN- 20
40 RB7
39 RB6
38 RB5
37 RB4
36 RB3/CANRX
35 RB2/CANTX
34 RB1/INT1
33 RB0/INT0
32 VDD
31 VSS
30 RD7/PSP7/PD
29 RD6/PSP6/PC
28 RD5/PSP5/PB
27 RD4/PSP4/ECCP1/PA
26 RC7/RX/DT
25 RC6/TX/CK
24 RC5/SDO
23 RC4/SDI/SDA
22 RD3/PSP3/C2IN-
21 RD2/PSP2/C2IN+



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Diagramas y Planos del AGV Genérico



D = Diámetro de la flecha del Motor

F = Largo de la flecha del Motor

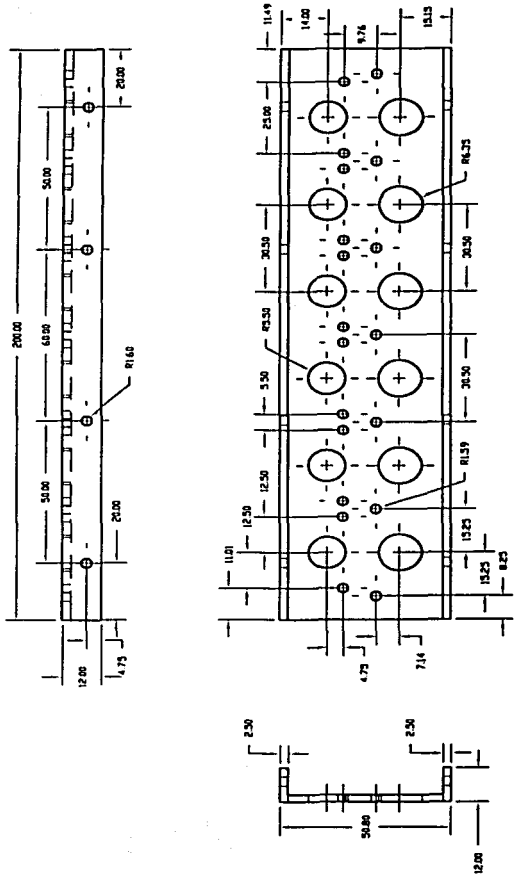
L = Ancho del Rin de la Rueda

d = Diámetro del eje central del Rin de la Rueda

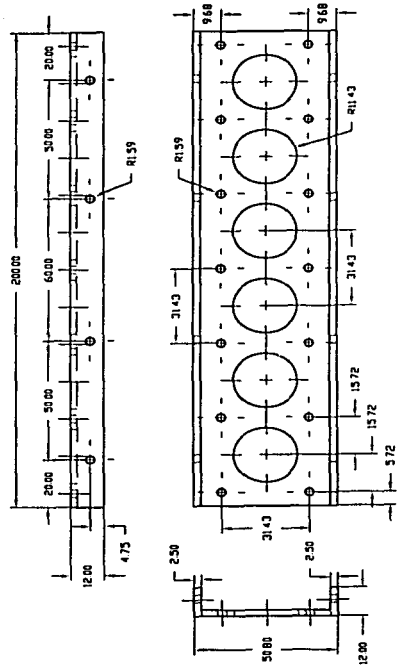
P = Diámetro del Prisionero

Nota: Los chaflanes no tienen una dimensión restringida.

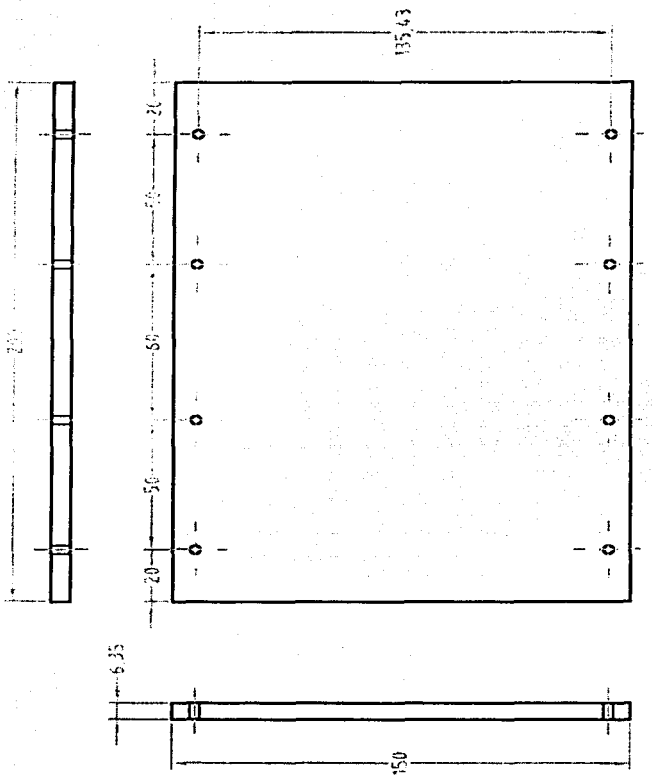
	Diseño de un AGV Genérico		
	Cople		
	Material: Acero Cold Rolled		Diseñó: UPR
	Cant. 2	Acot: mm	Dibujó: UPR
Archivo:	Fecha: X/2001	Esc:-	Revisó:



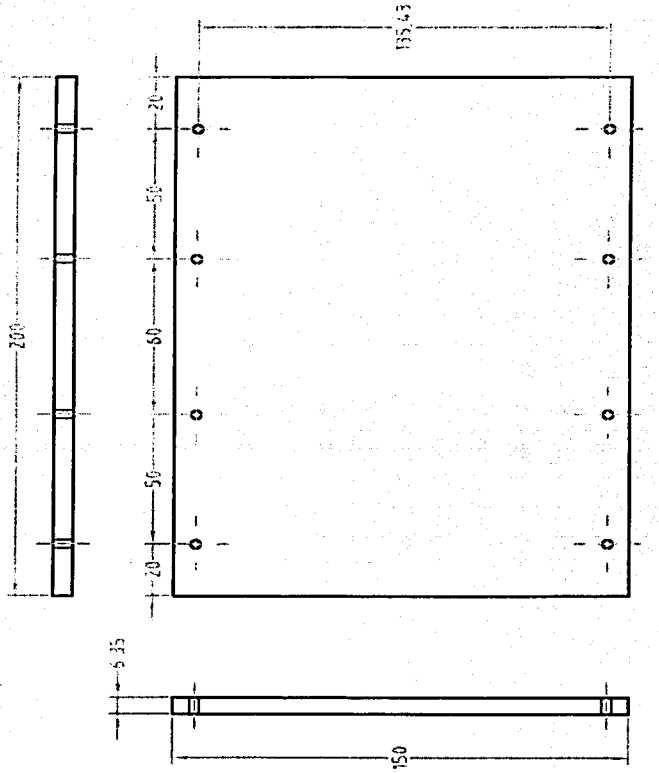
Diseño de un AGV Genérico		
Larguero para Motores Continuos		
Material: Patín de Aluminio	Diseñó: UPR	
Cant. 2	Acot: mm	Dibujó: UPR
Archivo:	Fecha: X/2001	Esc: Revisó: VGV



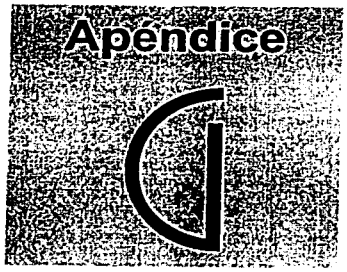
	Diseño de un AGV Genérico		
	Larguero para Motores a Pasos		
	Material: Patín de Aluminio	Cant. 2	Diseñó: UPR
		Acot: mm	Dibujó: UPR
Archivo	Fecha: X/2001	Esc:	Revisó: VGV



Diseño de un AGV Genérico			
	Base		
	Material: PVC Espumado	Cant. 1	Diseñó: UPR
	Acot: mm	Esc:	Dibujó: UPR
Archivo:	Fecha: X/2001		Revisó: VGV



Diseño de un AGV Genérico			
	Base		
	Material: PVC Espumado		Diseñó: UPR
	Cant. 1	Acot: mm	Dibujó: UPR
Archivo:	Fecha: X/2001	Esc:	Revisó: VGV



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Programas de Apoyo

8. Programas de Apoyo

Para una mejor comprensión acerca de la manera en que se programa el microcontrolador PIC 16F874, se proponen los siguientes ejemplos en lenguaje C y ensamblador. Estos programas abarcan aspectos de programación relacionados con el manejo de sistemas mecatrónicos, otorgando al lector una buena miscelánea del uso de los periféricos que contiene el microcontrolador.

Cada uno de estos programas indican en el encabezado el algoritmo a realizar, en los programas en lenguaje ensamblador todo lo que esté después de un punto y coma será tomado como comentario, en los programas en lenguaje C, todo lo que esté después de // o entre /* */ será tomado como comentario también. Solamente en los casos en que se requiera una estructura especial de conexión del hardware se indicará.

1. El programa rotará los bits del Puerto B primero hacia la derecha hasta antes del desbordamiento y después los rotará a la derecha.

```
;PROGRAMA KIT 1

    LIST   P=16F874          ;Comando que indica el PIC usado
    RADIX  HEX               ;Los valores se representar n en hexadecimal

;RAM
    PA      EQU   0X05       ;Dirección en el mapa de memoria del PA
    PB      EQU   0X06       ;Dirección en el mapa de memoria del PB
    PC      EQU   0X07       ;Dirección en el mapa de memoria del PC
    PD      EQU   0X08       ;Dirección en el mapa de memoria del PD
    T0      EQU   0X01       ;Registro del Timer 0
    ESTADO  EQU   0X03       ;Registro Status

;BITS
    W       EQU   0X00       ;Carga al acumulador
    F       EQU   0X01       ;Carga a la fuente

    C       EQU   0X00       ;Variables de localización de bits en el
    R0      EQU   0X05       ;registro Status
    R1      EQU   0X06

;VAR
    tmp     EQU   0X20       ;RAM registros de uso general

;-----------------------------------------------------------
    ORG   0                  ;Inicia el programa en la dirección 0
;-----------------------------------------------------------

;CONFIGURACIÓN DEL PROGRAMA

    bsf     ESTADO,R0        ;Cambio al banco 1
    bcf     ESTADO,R1

    movlw   0xD7             ;W <-- D7 (1101 0111)
    movwf   T0               ;W --> T0 Pre escala el Timer 0 (1:256)

    clrf    PB               ;0 --> TRISB, Configura a el Puerto B como salida
```

```
        bcf     ESTADO,R0       ;Cambio al banco 0
        bcf     ESTADO,R1

;PROGRAMA PRINCIPAL

        clrf    PB              ;Limpia las salidas del Puerto B

        movlw   0x01            ;W <-- 1
        movwf   PB              ;W --> Puerto B. El valor de W sale por las
        call    del             ;Llama a subprograma delay
izq     rlf     PB,F            ;Mueve los datos del PB un lugar a la izquierda
        call    del             ;Llama a subprograma delay
        btfss   PB,7            ;El bit 7 del Puerto B es 1?
        goto    izq             ;No, goto izq
der     call    del             ;Sí, genera un retrazo
        rrf     PB,F            ;Mueve los datos del PB un lugar a la derecha
        btfss   PB,0            ;El bit 0 del Puerto B es 1?
        goto    der             ;No, goto der
        goto    izq             ;Sí, goto izq

;RETARDO POR TIMER 0
del     movlw   0x01            ;Se carga el registro tmp con el valor de 1
        movwf   tmp             ;para realizar el ciclo anidado 1 vez
del     clrf    T0              ;Se pone a 0 el Timer 0
otra    btfss   T0,5 ;0 ;2 ;7   ;Ciclo anidado. El bit 5 del Timer 0 es 1?
        goto    otra            ;No, esperar
        decfsz  tmp,1           ;Sí, decrementa al registro tmp y pregunta:
        goto    del             ;tmp = 0?, No ir a la etiqueta del
        return                  ;Sí, regresa al programa principal

        END                     ;Fin del programa
```

2. El programa rotará los bits del Puerto B primero hacia la derecha hasta antes del desbordamiento y después los rotará a la derecha. La diferencia entre este programa y el anterior está en la carga de una librería que incluye el mapa de memoria del microcontrolador PIC 16F874 y que el retardo se genera por decremento de registros.

```
;PROGRAMA KIT 1

        LIST    P=16F874        ;Comando que indica el PIC usado
        RADIX   HEX             ;Los valores se representar n en hexadecimal
        INCLUDE "P16F874.INC"   ;Carga el mapa de memoria del microcontrolador

;VAR
        PRI     EQU     0x21    ;RAM registros de uso general
        SEG     EQU     0X22

;-----------------------------------------------------------
    ORG  0                      ;Inicia el programa en la dirección 0
;-----------------------------------------------------------

;CONFIGURACIÓN DEL PROGRAMA

        bsf     ESTADO,R0       ;Cambio al banco 1
        bcf     ESTADO,R1

        movlw   0xD7            ;W <-- D7 (1101 0111)
        movwf   T0              ;W --> T0 Prescala el Timer 0 (1:256)

        clrf    PB              ;0 --> TRISB, Configura a el Puerto B como salida

        bcf     ESTADO,R0       ;Cambio al banco 0
        bcf     ESTADO,R1
```

```
;PROGRAMA PRINCIPAL

        clrf    PB              ;Limpia las salidas del Puerto B

        movlw   0x01            ;W <-- 1
        movwf   PB              ;W --> Puerto B. El valor de W sale por las
        call    del             ;Llama a subprograma delay
izq     rlf     PB,F            ;Mueve los datos del PB un lugar a la izquierda
        call    del             ;Llama a subprograma delay
        btfss   PB,7            ;El bit 7 del Puerto B es 1?
        goto    izq             ;No, goto izq
der     call    del             ;Si, genera un retrazo
        rrf     PB,F            ;Mueve los datos del PB un lugar a la derecha
        btfss   PB,0            ;El bit 0 del Puerto B es 1?
        goto    der             ;No, goto der
        goto    izq             ;Si, goto izq

;RETARDO POR REGISTROS

del     movlw   0x02            ;Se carga el registro PRI con el valor de 2
        movwf   PRI             ;para realizar el ciclo anidado 2 veces
        clrf    SEG             ;Se pone a 0 el registro SEG
mas     incfsz  SEG             ;Ciclo anidado. Se incrementa el registro SEG y
        goto    mas             ;se pregunta: SEG = 0?, No incrementar de nuevo
        decfsz  PRI             ;Si, decrementa al registro PRI y pregunta:
        goto    mas             ;PRI = 0?, No ir a mas
        return                  ;Si, regresa al programa principal

        END                     ;Fin del programa
```

3. El programa rotará los bits del Puerto B primero hacia la derecha hasta antes del desbordamiento y después los rotará a la derecha. La diferencia entre este programa y los anteriores es que está realizado en lenguaje C.

```
// PROGRAMA KIT 3

//CARGA DE LIBRERIAS
#include "16f874.h"                     //Carga el mapa de memoria

//SUBPROGRAMA DE RETARDO POR DECREMENTO DE REGISTROS

void delay(void)
{
        char si,h;                      //Se inicializan variables de 8 bits
        si=2;                           //Se carga el valor de repetición del ciclo externo
        do {                            //Realiza el ciclo hasta que while lo indique
                h=20;                   //Se carga el valor de repetición del ciclo interno
                do { } while(--h>0);    //Se crea ciclo interno, se resta h en 1 cada vez,
                                        //sale cuando h = 0
        } while(--si>0);                //si = si - 1, sale del ciclo cuando si = 0
}

//SUBPROGRAMA DE RETARDO POR TIMER 0

void del(void)
{
        char m;                         //Se inicializa variable de 8 bits
        OPTION = 0;                     //Se predivide al Timer 0
        m=2;                            //Se carga el valor de repetición del ciclo externo
        do{                             //Realiza el ciclo hasta que while lo indique
                TMR0=0;                 //Reinicia el Timer0
                do{ }while(TMR0<20);    //Se crea ciclo interno y sale cuando el Timer0 = 20
        }while(--m>0);                  //Resta en 1 a m, sale del ciclo cuando m = 0
}
```

```
//PROGRAMA GENERAL

void main(void)                          //Inicia el programa principal
{
        char i;                          //Se inicializa variable de 8 bits
        TRISB = 0;                       //Se configura el Puerto B como salida
        PORTB = 1;                       //Se asigna el valor de 1 al Puerto B
        do{                              //Inicio de ciclo infinito
        for(i=0;i<=6;i++)
        {
                delay();
                PORTB = rl(PORTB);       //Recorre los bits del Puerto B 1 lugar a
        }                                //la izquierda
        for(i=0;i<=6;i++)
        {
                delay();
                PORTB = rr(PORTB);       //Recorre los bits del Puerto B 1 lugar a
        }                                //la derecha
        }
        while(1);                        //Fin de ciclo infinito
}
```

4. El programa rotará un bit primero hacia la derecha hasta antes del desbordamiento y después los rotará a la derecha. La diferencia entre este programa y los anteriores es que está realizado en lenguaje C y además utiliza registros de 16 bits con despliegue en dos puertos de 8 bits cada uno.

```
// PROGRAMA KIT 4

//CARGA DE LIBRERIAS
#include "16f874.h"                      //Carga el mapa de memoria
#include "c:\clase\uprdelay.c"           //Carga de función de retardo

int16 kit;                               //Inicializa variable pública de 16 bits

void res(void)                           //Inicio subprograma res
{
        PORTC = kit.low8;                //Carga los 8 bits LBS en el Puerto C
        PORTB = kit.high8;               //Carga los 8 bits MBS en el Puerto B
}

void main(void)                          //Inicio de programa principal
{
        char i;                          //Se inicializa variable de 8 bits

        TRISC = 0;                       //Se configura el Puerto C como salida
        TRISB = 0;                       //Se configura el Puerto B como salida
        PORTC = 1;                       //Se carga al Puerto C con 1
        PORTB = 0;                       //Se limpia todo el Puerto B
        kit = 1;                         //Se carga la variable kit con 1

        do{                              //Inicio de ciclo infinito
        for(i=0;i<=14;i++)               //Inicio de ciclo for
        {
                delay();                 //Llamado a funcion de retardo, esta contenida
                                         //en el archivo uprdelay.c
                kit = rl(kit);           //Rota y guarda un bit a la izq en el registro kit
                res();                   //Llamado al subprograma res
        }
        for(i=0;i<=14;i++)               //Inicio de ciclo for
        {
                delay();                 //Llamada a función de retardo
                kit = rr(kit);           //Rota y guarda un bit a la der en el registro kit
                res();                   //Llamado al subprograma res
        }
```

```
    }
    while(1);                          //Fin de ciclo infinito
}
```

5. En el siguiente programa se realiza una interfaz PC - Tarjeta de Desarrollo PIC 16F874, con comunicación vía RS232. El armado del cable se muestra en el capítulo 4. En la PC se deberá tener una hiperterminal estándar configurada de la siguiente manera: 9600 kbs, 8 bits de datos, sin paridad, 1 bit de parada y sin control de flujo.

El programa esperará hasta que se presione alguna tecla de la computadora, se generará una interrupción la cual tomará el carácter tecleado y lo desplegará en el monitor; a esto se le denomina *eco*.

```
;PROGRAMA ECO

      LIST    P=16F874              ;Tipo de PIC
      RADIX   HEX                    ;Sistema de Numeración
      INCLUDE "P16F874.INC"         ;Librería del Chip

      ORG     0X00                  ;Inicio en el vector del Reset
      GOTO    INICIO                ;Va a la primera instrucción
      ORG     0X04                  ;Vector de Interrupción

;ATENCIÓN DE INTERRUPCIÓN
;Se transmite vía serie el dato que está en W

INTER BTFSS   PIR1,RCIF             ;¿Interrupción por recepción?
      GOTO    VOLVER                ;No, esperar
      BCF     PIR1,RCIF             ;Sí. Reponer el Flag
      MOVF    RCREG,W                     ;Lectura del dato recibido
      MOVWF   PORTB                 ;Visualiza el dato

TX_DATO       BCF     PIR1,TXIF     ;Restaura el Flag de transmisión
      MOVWF   TXREG                 ;Mueve el bite a transmitir el registro de transmisión
      BSF     STATUS,RP0            ;Selecciona el banco 0
      BCF     STATUS,RP1            ;Selecciona el banco 1
TX_DAT_W      BTFSS   TXSTA,TRMT    ;¿Bite transmitido?
      GOTO    TX_DAT_W              ;No, esperar
      BCF     STATUS,RP0            ;Sí. Vuelve al banco 0

VOLVER        RETFIE                ;Regreso de Interrupción


INICIO CLRWDT                       ;Refresca el perro guardian
      CLRF    PORTB                 ;Limpia salidas
      CLRF    PORTC
      BSF     STATUS,RP0            ;Cambio al banco 1
      BCF     STATUS,RP1
      CLRF    TRISB                 ;PB como salidas
      MOVLW   B'10111111'           ;RC7/Rx entrada
      MOVWF   TRISC                 ;RC6/Tx salida
      MOVLW   B'11101111'           ;Predivisor asociado con el WDT a 128
      MOVWF   OPTION_REG
      MOVLW   B'00100100'           ;Configuración del USART y activación
      MOVWF   TXSTA                 ;de transmisión
      MOVLW   .51                   ;a 9600
      MOVWF   SPBRG
      BSF     PIE1,RCIE             ;Habilita interrupción de recepción
      BCF     STATUS,RP0            ;Cambio a banco 0
      MOVLW   B'10010000'           ;Configuración del USART
      MOVWF   RCSTA                 ;para recepción continua
```

```
                                    ;Puesta en ON
        MOVLW  B'11000000'          ;Habilitación de las recepciones
        MOVWF  INTCON               ;en general

BUCLE   CLRWDT                      ;Bucle infinito, espera interrupción
        GOTO   BUCLE

        END
```

6. El siguiente programa realiza una conversión analógica a digital de 10 bits, gracias al acoplamiento de una señal analógica, la cual no debe exceder 5 V (ver Capítulo 3), en el canal 4 (pin 7 del microcontrolador o PA5).

El programa inicia la conversión, espera a que esté terminada y el resultado lo despliega por el puerto B y D.

```
; PROGRAMA CAD 0

        LIST    P=16F874            ;Tipo de PIC
        RADIX   HEX                 ;Sistema de Numeración
        INCLUDE "P16F874.INC"       ;Libreria del Chip

;INICIO DE PROGRAMA

        ORG     0X00

        BSF     STATUS,RP0          ;Cambio a banco 1
        BCF     STATUS,RP1

        MOVLW   B'00100000'         ;RA5 como entrada
        MOVWF   TRISA

        CLRF    TRISD               ;Se configuran el Puerto D y B como salidas
        CLRF    TRISB

        MOVLW   B'10000000'         ;Se le asigna justificación a la derecha al
        MOVWF   ADCON1              ;convertidor analógico a digital

        BCF     STATUS,RP0          ;Cambia al banco 0

        MOVLW   B'00100001'         ;Se configura el convertidor para tener como
        MOVWF   ADCON0              ;entrada analogica al canal 4 y voltaje de
                                    ;referencia igual a el de la tarjeta

OTRA    BCF     PIR1,ADIF           ;Se repone la bandera de fin de conversión
        BSF     ADCON0,GO           ;Se inicia conversión
CAD     BTFSS   PIR1,ADIF           ;Ya se termino la conversión?
        GOTO    CAD                 ;No, espera
        MOVF    ADRESH,W            ;Sí, carga la parte alta de la conversión a W
        MOVWF   PORTD               ;Mueve W al Puerto D
        BSF     STATUS,RP0          ;Cambio a banco 1
        BCF     STATUS,RP1
        MOVF    ADRESL,W            ;Carga la parte baja de la conversión en W
        BCF     STATUS,RP0          ;Cambia a banco 1
        MOVWF   PORTB               ;Mueve W a el Puerto B
        GOTO    OTRA                ;Bucle infinito

        END
```

7. El siguiente programa muestra el uso de uno de los dos PWM que contiene la tarjeta de desarrollo PIC 16F874. En este programa también se utiliza la tarjeta de circuito H y un motor DC continuo.

Las conexiones son las siguientes:

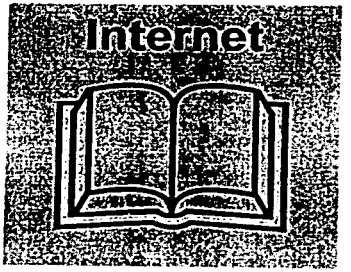
1. El motor deberá estar conectado a las salidas del motor 1 de la tarjeta de circuito H
2. El pin de habilitación del motor 1 de la misma tarjeta, estará conectado a el RC1 del header de la tarjeta de desarrollo PIC 16F874
3. La alimentación del motor 1 de la tarjeta H puede debe conectarse correctamente con una tensión que soporte el motor y no exceda a 36 V (ver capítulo 4)

El programa realiza el siguiente algoritmo:

1. Incremanta la velocidad del motor hasta llegar a la velocidad máxima
2. Mantiene la velocidad del motor por 3 s
3. Disminuye la velocidad del motor hasta pararlo
4. Se espera 1 s
5. Cambia el sentido de giro y regresa al punto 1

```
; PROGRAMA PWM 0

        LIST    P=16F874                ;Tipo de PIC
        RADIX   HEX                     ;Sistema de Numeración
        INCLUDE "P16F874.INC"           ;Librería del Chip

;INICIALIZACIÓN DE CONSTANTES
PER   EQU     .255                      ;Se asigna una constante de 255 para el periodo

;INICIALIZACIÓN DE VARIABLES
DEL   EQU     0X20
TEMP  EQU     0X21

        ORG     0X00
        GOTO    INI
        ORG     0X05                    ;Salta el vector de interrupción

INI     CLRF    PORTB                   ;Limpia al Puerto A, B y C
        CLRF    PORTC
        BSF     STATUS,RP0              ;Cambio al banco 1
        MOVLW   B'00000110'             ;Configura al registro ADCON1
        MOVWF   ADCON1
        CLRF    TRISB                   ;Configura al Puerto B como salida
        MOVLW   B'11111101'             ;Configura al Puerto C como 1 salida
        MOVWF   TRISC                   ;y 7 entradas

        MOVLW   B'11010111'             ;Configura al registro predivide al Timer 0 en 1:255
        MOVWF   OPTION_REG

        MOVLW   PER                     ;Carga a el registro PR2 (control de
        MOVWF   PR2                     ;periodo) con 255
```

```
        BCF     STATUS,RP0              ;Cambio al banco 0

BUCLE   MOVLW   B'00001100'             ;Configura al CCP2 a modo PWM
        MOVWF   CCP2CON

        MOVLW   .255                    ;Carga a la variable TEMP con 255
        MOVWF   TEMP                    ;(velocidad máxima)

        CLRF    CCPR2L                  ;Velocidad = 0

        MOVLW   B'00000111'             ;Enciende al Timer 2 y lo predivide en 1:16
        MOVWF   T2CON

;1° Incrementa la velocidad de 0 a máxima

HOR     MOVLW   .2                      ;Carga el sentido de giro del motor
        MOVWF   PORTB
        CALL    AUM                     ;Llama a subprograma de incremento de velocidad
        MOVLW   .150
        CALL    DELAY                   ;Llama a retardo
        CALL    DIS                     ;Llama a subprograma de disminución de velocidad

        MOVLW   .50
        CALL    DELAY                   ;Llamada a retardo

        CLRF    CCPR2L                  ;Velocidad = 0
        MOVLW   .255
        MOVWF   TEMP
AHOR    MOVLW   .1                      ;Invierte el sentido de giro del motor
        MOVWF   PORTB
        CALL    AUM                     ;Llama a subprograma de incremento de velocidad
        MOVLW   .150
        CALL    DELAY                   ;Llama a retardo
        CALL    DIS                     ;Llama a subprograma de disminución de velocidad

        MOVLW   .50
        CALL    DELAY                   ;Llamada a retardo

        GOTO    BUCLE                   ;Ciclo infinito

AUM     MOVLW   .2
        CALL    DELAY                   ;Llamada a retardo
        INCF    CCPR2L,F                ;Aumanta la velocidad hasta un máximo
        DECFSZ  TEMP,F                  ;Control de velocidad máxima
        GOTO    AUM
        RETURN

DIS     MOVLW   .2
        CALL    DELAY                   ;Llamada a retardo
        DECFSZ  CCPR2L,F                ;Disminuye la velocidad hasta 0
        GOTO    DIS                     ;Control de velocidad mínimo
        RETURN


DELAY           MOVWF   DEL                     ;Registro de control de veces de reardo
DELI    MOVLW   .100
        MOVWF   TMR0                    ;Se carga el Timer 0 con 100
DELL    BTFSS   INTCON,T0IF             ;Espera hasta el desbordamiento del Timer 0
        GOTO    DELL
        BCF     INTCON,T0IF             ;Restaura la bandera de interrupción por
                                        ;desbordamiento en el Timer 0
        DECFSZ  DEL,F                   ;Decrementa al registro DEL
        GOTO    DELI
        RETURN

        END
```



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Bibliografía

1. Sensors from Mobile Robots
Everett, H. R.
Peters
Massachussets, USA, 1995

2. The Robot Builder's Bonanza: 99 Inexpensive Robotics Proyects
Gordon Mc Comb
Mc Graw Hill
Pennsylvania, USA, 1987

3. Fundamentos de Ingeniería, Métodos Conceptos y Resultados
Eduard Krick
Limusa
DF, México, 1991

4. Hobile Robots, Inspiration to Implimentation
Jones, joseph, Flynn, Anita
A KI Peters
Wellesley, Massachusetts. USA, 1993

5. Manual del Ingeniero Mecánico
Edward H. Smith
Prentice Hall
DF, México, 1994

6. HC11, Reference Manual
Motorola
USA, 1991

7. PIC16/17, Microcontroller Databook
Micrichip
USA, 1996

8. Microcontroladores 8051 y 8052
Bernard Odant
Paraninfo
Madrid, España, 1995

9. Intelligent Sensor Technology
Ryoji Ohba
Wiley
Chichester, Inglaterra, 1992

10. Microcontroladores PIC, Diseño Práctico de Aplicaciones
Usategui A., Martínez I.
Mc Graw Hill
España, 1999

11. Microcontroladores PIC, Diseño Práctico de Aplicaciones, Segunda Parte
Usategui A., Martinez I.
Mc Graw Hill
España, 2000

12. Microchip PIC 16F87X Data Sheet
Microchip
USA, 2001

13. Microchip PIC 18F458 Data Sheet
Microchip
USA, 2001

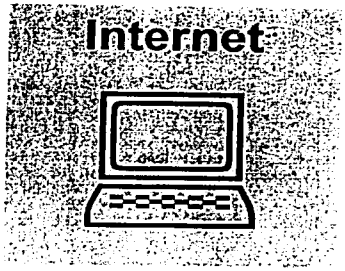
14. Design of Microprocessor Sensor & Control System
Michael F. Hordeski
Reston Plublishing Company
USA, 1985

15. Design with PIC Microcontrollers
John Peatman
Prentice Hall
USA, 1997

16. Robotics, Introduction, Programming, and Projects
James L. Fuller
2ª ed
Prentice Hall
New Jersey, USA, 1999

17. Mecatrónica, Sistemas de Control Electrónico en Ingeniería Mecánica y Eléctrica
W. Bolton
2ª ed
Alfaomega
D.F., México, 2001

16. Ingeniería de Control Moderna
Katsuhiko Ogata
2ª ed
Prentice Hall
México, México, 1993



DESARROLLO DE UN SISTEMA BASADO EN UN PIC 16F874 PARA APLICACIONES MECATRÓNICAS

Internet

Paginas Interesantes en la Red
En las siguientes direcciones encontraras datos interesantes acerca de microcontroladores y componentes electrónicos:

Advanced Micro Devices:
http://www.amd.com

Analog Devices:
http://www.analog.com

Crossbow Technologies:
http://www.xbow.com

Hitachi America Ltd.:
http://www.hitachi.com

Integrated Device Technology (IDT):
http://www.idt.com

Intel Corp.:
http://www.intel.com

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

MIPS:
http://www.mips.com

Motorola Semiconductor Products Sector:
http://www.mcu.motsps.com/bu/mctg/mctg.html

Philips Semiconductors:
http://www.semiconductors.philips.com

Siemens:
http://www.sci.siemens.com

Toshiba:
http://www.toshiba.com

MCU/MPU Internet Resource List:
http://www.cera2.com/micro.htm

CPU Info Center:
http://infopad.eecs.berkeley.edu/CIC/news/