INSTITUTO
POLITÉCNICO
NACIONAL

ESIME "ZACATENCO"

ICE

"CONTROL ELECTRÓNICO DE VÁLVULAS HIDRÁULICAS A TRAVÉS DE UN ORDENADOR"

TESIS

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMUNICACIONES
Y ELECTRÓNICA

PRESENTA:

LEOPOLDO CRUZ BUTRÓN

M. en C. AMÉRICA MARÍA
GONZÁLEZ SÁNCHEZ
DIRECTOR

M. en C. CANDIDO PALACIOS
MONTUFAR
ASESOR

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELECTRICA

UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

T E M A D E T E S I S

QUE PARA OBTENER EL TITULO DE	**INGENIERO EN COMUNICACIONES Y ELECTRÓNICA**
POR LA OPCIÓN DE TITULACIÓN	**TESIS Y EXAMEN ORAL INDIVIDUAL**
DEBERA(N) DESARROLLAR	**C. LEOPOLDO CRUZ BUTRON**

"CONTROL ELECTRÓNICO DE VÁLVULAS HIDRÁULICAS A TRAVÉS DE UN ORDENADOR"

DESARROLLAR E IMPLEMENTAR UNA TARJETA ELECTRÓNICA, ASÍ COMO EL PROGRAMA DE INTERFAZ CON LA PC. PARA ACTIVAR LA SOLENOIDE DE VÁLVULAS HIDRÁULICAS, QUE GOBIERNAN EL MOVIMIENTO DE MOTORES HIDRÁULICOS Y MOSTRAR UNA APLICACIÓN EN LA CUAL SE CONJUNTAN HARDWARE Y SOFTWARE, HACIENDO USO DE LOS CONOCIMIENTOS ADQUIRIDOS EN LAS DIFERENTES MATERIAS, A LO LARGO, DE NUESTRA FORMACIÓN PROFESIONAL

- IMPLEMENTAR EL PROGRAMA DE INTERFAZ DE USUARIO Y COMUNICACIÓN USB
- DISEÑAR Y FABRICAR LA TARJETA ELECTRÓNICA, PARA LA COMUNICACIÓN USB Y CONTROL DE LAS VÁLVULAS HIDRÁULICAS

MÉXICO D.F. A 08 DE OCTUBRE 2009

A S E S O R E S

M. EN C. AMERICA MARÍA GONZÁLEZ SÁNCHEZ

M. EN C. CANDIDO PALACIOS MONTUFAR

M. EN C. SALVADOR RICARDO MENESES GONZÁLEZ
JEFE DEL DEPARTAMENTO ACADÉMICO DE
INGENIERÍA EN COMUNICACIONES Y ELECTRÓNICA

ESCUELA SUPERIOR DE INGENIERIA MECANICA Y ELECTRICA
ESTADOS UNIDOS MEXICANOS
JEFATURA
DE I.C.E.

INSTITUTO
POLITÉCNICO
NACIONAL

ESIME "ZACATENCO"

"CONTROL ELECTRÓNICO DE VÁLVULAS HIDRÁULICAS A TRAVÉS DE UN ORDENADOR"

OBJETIVO:

DESARROLLAR E IMPLEMENTAR UNA TARJETA ELECTRÓNICA, ASÍ COMO EL PROGRAMA DE INTERFAZ CON LA PC, PARA ACTIVAR LA SOLENOIDE DE VÁLVULAS HIDRÁULICAS.

LEOPOLDO CRUZ BUTRÓN

M. en C. AMÉRICA MARÍA GONZÁLEZ SÁNCHEZ
DIRECTOR

M. en C. CANDIDO PALACIOS MONTUFAR
ASESOR

IPN ESIME

INDICE

INTRODUCCIÓN.

El Manipulador Industrial Robótico Hidráulico (MIRH1), requiere de motores hidráulicos para generar los movimientos necesarios para la realización de algún proceso industrial, a su vez estos motores son controlados por válvulas hidráulicas, las cuales son activadas a través de solenoides, el actual sistema del MIRH1 presenta la siguiente problemática a resolver:

- Controlar válvulas que gobiernan el movimiento de motores hidráulicos.
- Sustituir tarjeta electrónica costosa e inoperable.
- Manejar semiconductores en lugar de relevadores que presentan desgaste mecánico.
- Remplazar panel de control convencional por ordenador.
- Utilizar comunicación USB en lugar del descontinuado puerto serial.

La presente tesis describe una opción para controlar dichas válvulas a través de un ordenador haciendo uso de una interfaz electrónica y mostrar una aplicación en la cual se conjuntan Hardware y Software, haciendo uso de los conocimientos adquiridos en las diferentes materias, a lo largo, de nuestra formación profesional.

Los robots industriales pueden clasificarse de diversas maneras, entre las que destacan la geometría, la fuente de poder y las aplicaciones, temas que serán tratados en el capítulo uno del presente trabajo.

El capítulo dos trata la parte hidráulica, tanto la bomba de aceite así como las electroválvulas a controlar por el circuito electrónico.

En el capítulo tres se aborda el Hardware, lo que comprende al microcontrolador, características, programación, también las interfaces a utilizar y las primeras pruebas de comunicación serial.

En el capítulo cuatro se explica el Software a utilizar para el desarrollo de la aplicación que será la Interface Grafica de Usuario (GUI) obteniendo también el tratamiento de los datos y la comunicación serial y USB.

El desarrollo de la tarjeta final para la comunicación USB y la etapa de potencia se abordan en el capítulo cinco, por último se dan las conclusiones y recomendaciones, mismas que pueden ser útiles en el desarrollo de futuras modificaciones al manipulador para obtener el desempeño total del MIRH1.

CAPÍTULO I.

1.1 ESTADO DEL ARTE DE LOS MANIPULADORES ROBÓTICOS INDUSTRIALES.

En el presente capítulo se presenta una breve reseña histórica de la evolución de los manipuladores robóticos así como las diferentes clasificaciones en que pueden ser catalogados al tomar en cuenta su morfología, la energía primaria para inducir el movimiento, la clasificación de acuerdo a los lineamientos de la Organización Internacional de Estandarización y el uso final al que estén destinados.

1.2 BREVE RESEÑA HISTÓRICA.

La robótica ha tenido un desarrollo importante a lo largo de los años incrementándose de manera exponencial en las últimas décadas al hacer uso de la tecnología de punta obteniendo como resultado robots autónomos, con inteligencia artificial capaces de resolver problemas cotidianos sin necesidad de ser reprogramados, a continuación se presenta un breve resumen del desarrollo recorrido por la robótica a través de los años.

En 1801 Joseph Jacquard inventó la máquina textil que era operada por medio de tarjetas perforadas, esta máquina fue llamada telar programable y fue para la producción en masa.

En 1892 en los Estados Unidos, Seward Babbit desarrolla una grúa motorizada con un efector final para remover lingotes de una caldera.

En 1921 se da la primera referencia de la palabra robot, la cual aparece en Londres por el checoslovaco Karen Capek, esta palabra tiene sus raíces en el checo “robota”, la cual significa servir en una labor de sirviente, a partir de esta fecha se toma el concepto de robot.

En 1938 es diseñado un mecanismo de pintura en spray programable para la compañía DeVilbiss, por los americanos William Pollard y Harold Roselund.

En 1939 Isaac Asimov, escritor de ciencia-ficción introduce el diseño de robots para ayudar a la humanidad y trabajar de una manera segura. Tres años mas tarde el mismo formula sus tres leyes de la robótica, las cuales establecen como primera ley que un robot no puede hacer daño a un ser humano ni permitir que sufra daño alguno, la segunda ley dice que un robot debe obedecer al ser humano siempre y cuando no se contradiga la primera ley, y la tercera dice que un robot debe proteger su propia existencia siempre y cuando dicha protección no intervenga con la primera y segunda ley.

En 1946 George Devol patenta el aparato para controlar máquinas de propósito general, el cual usa un proceso de grabación magnética.

En 1948 Norbert Wiener, un profesor del Instituto Tecnológico de Massachussets, publica el libro Cybernetics, el cual describe el concepto de comunicación y control en electrónica, mecánica, y sistemas biológicos.

En 1951 es diseñado un brazo articulado equipado con un teleoperador por Raymond Goertz para la Comisión de Energía Atómica.
En 1954 es diseñado el primer robot programable por George C. Devol, quien utiliza el término automatización universal.

En 1959 la compañía Planet Corporation introduce en el mercado el primer robot comercial disponible.

En 1960 la compañía Condec Corporation compra Unimation y comienza el desarrollo del Unimate Robot system.

En 1961 es instalado el primer robot Unimate para descargar.

En 1962 General Motors instala el primer robot industrial en una línea de producción, el robot seleccionado es de la marca Unimate.

En 1968 el instituto de investigación de Stanford, construye y prueba un robot con capacidad de visión, el cual es nombrado Shakeys.

En 1970 es desarrollado en la universidad de Stanford un brazo robot, el cual se convierte en la base para los proyectos de investigación. La fuente de potencia del brazo es eléctrica y es conocido como el brazo de Stanford.

En 1971 la Asociación de Robots Industriales Japonesa (JIRA por sus siglas en ingles) comienza la promoción del uso de robots en las industrias japonesas.

En 1973 es desarrollado el primer robot controlado por mini computadora comercialmente disponible por Richard Honh para la compañía Cincinnati Milacron. El robot es llamado el T3, la herramienta del mañana.

En 1975 se forma el Instituto de Robots Americano (RIA) para ayudar efectivamente a las industrias norteamericanas a implementar robots en la automatización de las fábricas.

En 1976: El robot de la NASA “Vinking II” aterriza en Marte. El cual disponía de un brazo robótico articulado.

En 1977 ASEA Brown Boberi Robotics Inc, una compañía europea de robots ofrece dos tamaños de robots industriales eléctricos controlados por microcomputadora.

En 1978 con apoyo de la General Motors, Unimation desarrolla la máquina universal de ensamble programable (PUMA) usando tecnología de la empresa Vircarm.

En 1984 son introducidos los robots Direct-drive por la compañía Adept con motores eléctricos conectados directamente en los brazos, eliminando la necesidad de engranes y cadenas intermediarias.

En 1990 ASEA Brown Boberi Robotics Inc compra la división de robótica de la compañía Cincinnati Milacron, y todos los robots futuros serán ASEA machines.

1.3 CLASIFICACIÓN DE LOS ROBOTS DE ACUERDO A SU CONFIGURACIÓN MECÁNICA.

De acuerdo a la geometría o a la configuración mecánica básica del manipulador robótico, estos pueden clasificarse como cartesianos, cilíndricos, esféricos y articulados.

1.3.1 Geometría cartesiana.

Un robot con geometría cartesiana puede mover su efector final a cualquier posición dentro de un cubo o un rectángulo definido como su área de trabajo. Esta configuración está formada por dos categorías transversal y longitudinal.

Este tipo de coordenadas geométricas tienen las siguientes ventajas:
Áreas muy largas de trabajo, ya que el desplazamiento en el eje “X” puede ser incrementado fácilmente.

El montaje de cabezales deja grandes áreas de manufactura libres para otros usos.

Pueden usarse sistemas de control simples.

Las desventajas que presentan estos tipos de coordenadas son:

El acceso al área de trabajo por medio del cabezal con carga de otros materiales o equipos puede desequilibrar la estructura.

En algunos modelos la posición de los mecanismos de manejo, así como el control eléctrico puede causar dificultades de mantenimiento.

1.3.2 Geometría cilíndrica.

Un robot de geometría cilíndrica puede mover su efector final dentro del volumen descrito por un cilindro. El brazo de geometría cilíndrica esta posicionado en el área de trabajo por dos movimientos lineales, uno a lo largo del eje “Z” y el otro en la dirección del radio ”R”, y uno de rotación angular alrededor del eje Z.

Algunas de las ventajas en la geometría cilíndrica son las siguientes:

Un profundo alcance horizontal en las máquinas de producción.

La estructura vertical de la máquina conserva el espacio de la planta.

Una muy rígida estructura es posible para soportar grandes cargas y una buena repetibilidad.

La gran desventaja de esta geometría es el alcance limitado a la izquierda y a la derecha debida a las deformaciones mecánicas.

1.3.3 Geometría esférica.

Los brazos de geometría esférica también llamados polares son aquellos que pueden mover su efector final dentro del volumen descrito por una esfera, requieren un movimiento coordenado en todos los ejes de posición para el movimiento en las direcciones X, Y y Z.

Los brazos de geometría esférica posicionan al robot en dos rotaciones y un desplazamiento lineal. La orientación de la herramienta está dada a trevés de tres rotaciones en la muñeca (rotación, cabeceo y alabeo).

Las ventajas y desventajas son las mismas que las de los brazos de geometría cilíndrica con la excepción de que los robots de geometría cilíndrica poseen una estructura más vertical y los de geometría esférica son más bajos y alargados en tamaño.

1.3.4 Geometría articulada.

Los robots industriales articulados también llamados máquinas antropomórficas, tienen un área de trabajo irregular, tienen dos grandes variantes: verticalmente articulado y horizontalmente articulado.

Los robots verticalmente articulados son también llamados juntas esféricas y tienen tres movimientos principales angulares, la base de rotación (eje 1), el hombro (eje 2) y antebrazo (eje 3).

Ventajas de los robots verticalmente articulados:

Aunque ocupa un mínimo de espacio en el piso, tiene una gran distancia de alcance horizontal.

Tiene un buen radio de alcance, resultado de la habilidad del brazo de contraerse cuando se encuentra en posición retraída.

Una alta movilidad y posicionamiento del brazo le permiten tener alcance en espacios cerrados y alrededor de obstrucciones.

Existen dos posibles variaciones de la geometría verticalmente articulada descritas como:

Un eje adicional de movimiento rotacional (eje 4) en el antebrazo que le permite rotar a este eslabón.

Un eje de movimiento adicional lineal (eje 4) en el antebrazo que le permite extenderse y expandirse.

El robot horizontalmente articulado tiene dos movimientos angulares que consisten en una rotación en el brazo y el antebrazo, y un movimiento de posición lineal para un movimiento vertical. Los brazos horizontalmente articulados están implementados por dos configuraciones mecánicas:

El brazo de robot articulado.

El robot articulado de base horizontal.

1.4 CLASIFICACION DE LOS ROBOTS DE ACUERDO A SU FUENTE DE PODER.

Existen tres fuentes primarias para la alimentación de potencia en la manufactura para los sistemas de manejo. Hidráulica, neumática y fuerza electromagnética, también usados como generadores de movimiento en los robots actuales. La clasificación de los robots en base a la fuente de energía utilizada es la siguiente:

1.4.1 Potencia hidráulica.

Los robots que utilizan como fuente primaria un generador hidráulico, son diseñados para trabajos en los que la fuerza necesaria para ejecutar la tarea es muy grande, estos robots obtienen la fuerza requerida a través de una bomba hidráulica que alimentara los actuadores hidráulicos que generaran el movimiento de los eslabones del manipulador.

1.4.2 Potencia neumática.

Los robots que utilizan como fuente primaria el aire comprimido son diseñados para trabajos en los que la fuerza que se requerirá es pequeña pero a mayores velocidades, estos robots obtienen la fuerza requerida a través de un compresor de aire, que alimentara los actuadores neumáticos que generaran el movimiento de los eslabones del manipulador.

1.4.3 Potencia electromagnética.

Los robots que utilizan la potencia electromagnética son los de menor capacidad en cuanto a fuerza de trabajo se refiere, pueden llegar a ser muy precisos dependiendo del número de pasos de los motores que moverán las articulaciones, estos motores son alimentados a través de una corriente eléctrica a diferentes voltajes.

1.5 Clasificación de acuerdo a la Organización Internacional de Estándares (ISO).

La ISO ha establecido muchos documentos de estándares para ayudar en la colección de datos válidos de un robot, siendo cuatro principales áreas: secuencia, trayectoria, adaptabilidad y tele operación. La operación del controlador del robot provee la principal diferencia en la clasificación de categorías dadas por el estándar.

1.5.1 Secuencia.

El robot neumático no servo-controlado con control de línea de paro a paro, ya sea en geometría cartesiana o cilíndrica es el que mejor describe a esta categoría. La naturaleza binaria de encendido y apagado del controlador de salida, maneja los ejes secuencialmente para una buena definición de los puntos finales. La trayectoria sin embargo, no está controlada o definida. El controlador mas frecuente usado para esta categoría de robots es el PLC o controlador lógico programable.

1.5.2 Trayectoria.

Esta categoría incluye todas las geometrías con servomotores eléctricos o con ejes hidráulicos y operaciones de trayectoria controladas. Esta clasificación se caracteriza por el movimiento de multi-ejes y movimientos en línea recta generados internamente.

1.5.3 Adaptables.

Esta nueva categoría incluye “máquinas pensantes”. Ejemplos puros de esta categoría no existen actualmente, sin embargo robots de trayectoria operados con sensores adaptables, o controles con funciones de autoaprendizaje tipifican estos sistemas.

1.5.4 Teleoperados.

Robots teleoperados que extienden las funciones motrices humanas y robots en otros planetas a lugares remotos, han sido usados por muchos años para manipular material radioactivo. Esta categoría incluye una nueva clase de máquinas teleoperadas que pueden programarse para responder a las acciones del operador.

1.6 Clasificación de acuerdo a su empleo.

1.6.1 Robot de tomar y colocar.

Es el robot mas sencillo, este robot toma un objeto y lo coloca en otro lugar. La libertad del movimiento suele estar limitada a los grados de libertad del manipulador.

1.6.2 Servorobot.

En este robot se emplean servomecanismos para los brazos y manos a fin de modificar su sentido de movimiento cuando están en el aire.

1.6.3 Robot programable.

Se acciona con un controlador programable en el cual se almacena una secuencia de movimientos en una memoria y los repite en forma continua.

1.6.4 Robot controlado por computadora.

Este tipo de robot se programa mediante instrucciones electrónicas del controlador.

1.6.5 Robot sensorial.

Es un robot controlado por computadora que tiene uno o más sentidos artificiales para detectar las zonas de trabajo y retroalimentar información al controlador.

1.6.6 Robot para línea de ensamble.

Es un robot controlado por computadora, que quizá tenga sensores, está destinado a trabajos en la línea de ensamble.

1.7 Beneficios y desventajas de la robótica.

La ventaja que presenta en la industria el invertir dinero en un robot, es que aunque es una inversión a largo plazo, por el alto costo inicial que representa el comprar un robot, la productividad es aumentada hasta en un 60%, además de una mejor eficiencia comparada con la que podría tener un operario calificado, ya que al ser humano le afectan muchos factores de su vida cotidiana, como son las desveladas, la mala alimentación, y demás conflictos emocionales que se le pueden llegar a presentar, teniendo éstos un gran peso en la calidad del producto que se esta elaborando, a diferencia de un manipulador robótico, el cual al no poseer sentimientos, no tiene ningún tipo de problema emocional, además de que puede trabajar dos turnos seguidos sin cansarse, manteniendo la misma calidad en su trabajo y con la misma eficiencia.

En cuanto a seguridad se refiere, es mas barato tener a un robot para realizar tareas de alto riesgo que a un operador calificado, ya que primero, en caso de que sucediera un accidente, el robot no necesita de un hospital para ser atendido, por lo tanto no se necesita pagar una póliza de seguro para resguardar su salud.

Tomando como ejemplo el proceso de soldadura con arco eléctrico, el cual es un proceso en el cual se produce mucho calor, además de ser riesgoso y tedioso por lo repetitivo, el soldador corre el riesgo de un accidente, a diferencia del manipulador, que esta diseñado específicamente para realizar tareas repetitivas.

Aunado a lo anterior se tiene que un robot que labora dos turnos diarios costará aproximadamente 6 dólares la hora, ya tomando en cuenta la energía que consume, por lo que es mas barato que un operario.

Cabe mencionar que los robot actuales no van a sustituir a las personas, ya que los robots no pueden actuar ante situaciones imprevistas, ni con cambios de condiciones, por lo que en la mayoría de las plantas manufactureras se emplean personas y robots en las líneas de producción en donde las personas elaboran actividades que requieren de la capacidad motora y perceptiva humana, la coordinación de los ojos con las manos, la plantación, decisiones y evaluación.

De lo anterior se puede resumir las razones por las cuales se instala un robot en una industria.

La reducción de costos de mano de obra.
Mejorar la calidad del producto.
Eliminar trabajos peligrosos y monótonos.
Aumentar el volumen de producción.
Aumentar la flexibilidad en los productos.
Reducir el desperdicio de materiales.
Cumplir con los reglamentos de seguridad industrial.
Disminuir la rotación de personal.
Reducir el costo de inversiones en equipo.

1.8 ASPECTOS FÍSICOS DEL MANIPULADOR.

1.8.1 Clasificación MIRH1.

De acuerdo a la geometría del manipulador, éste se clasifica dentro de los robots industriales de geometría articulada, verticalmente articulado, ya que posee tres movimientos principales que son una rotación en la base, un hombro y un antebrazo, además de poseer un eje adicional de movimiento rotacional que le permite rotar el antebrazo.

Con respecto a la fuente de poder, el MIRH1 se sitúa dentro de los robots hidráulicos, de donde viene su nombre, debido a que el movimiento estará generado por la fuerza que le transmite una bomba hidráulica y que es repartida mediante ductos a sus diferentes actuadores para así lograr que se mueva.

El MIRH1 es un robot de trayectoria de acuerdo a la clasificación de la Organización Internacional de Estándares, ya que es la que contiene a los robots hidráulicos con operaciones de trayectorias controladas.

El empleo del MIRH1 es de tipo pedagógico para la enseñanza, pero se pronostica que será un robot controlado por computadora, mediante programas cargados al controlador que para mover sus motores hidráulicos y sus válvulas proporcionales.

1.8.2 Capacidades del MIRH1.

Las capacidades del MIRH1 están enfocadas básicamente al volumen de trabajo y a la capacidad de carga que puede tener el manipulador, estos datos fueron obtenidos directamente

de la tesis “Diseño mecánico de un brazo Manipulador Industrial Robótico Hidráulico (MIRH1) de 5 grados de libertad”, trabajo que precede del actual y mismo en el que se definió la estructura y diseño del MIRH1.

1.8.3 Volumen de trabajo.

El MIRH1 tiene un volumen de trabajo determinado por la geometría de sus eslabones y la capacidad de sus motores hidráulicos, este volumen comprende un alcance máximo y mínimo longitudinal, tomado a partir del eje de giro del motor de la base al extremo libre del brazo y, un alcance vertical superior e inferior medido a partir de la base sobre la cual esta montado hasta el extremo libre del manipulador.

1.8.4 Alcance longitudinal.

El alcance máximo del MIRH1 es de 1178.60 mm a una altura de 571.70 mm tomando como referencia el eje de rotación del motor de la base, como se muestra en la figura 1.1.

El alcance mínimo horizontal del MIRH1 es de 840.40 mm a una altura de 850.0 mm tomando como referencia el eje de rotación del motor de la base, como se muestra en la figura 1.2.

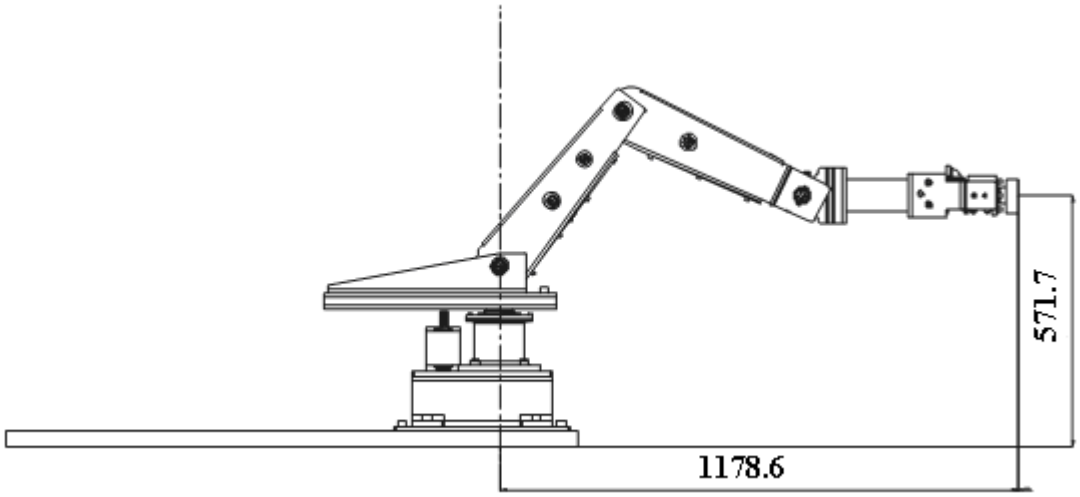


Figura 1.1 Alcance máximo longitudinal.

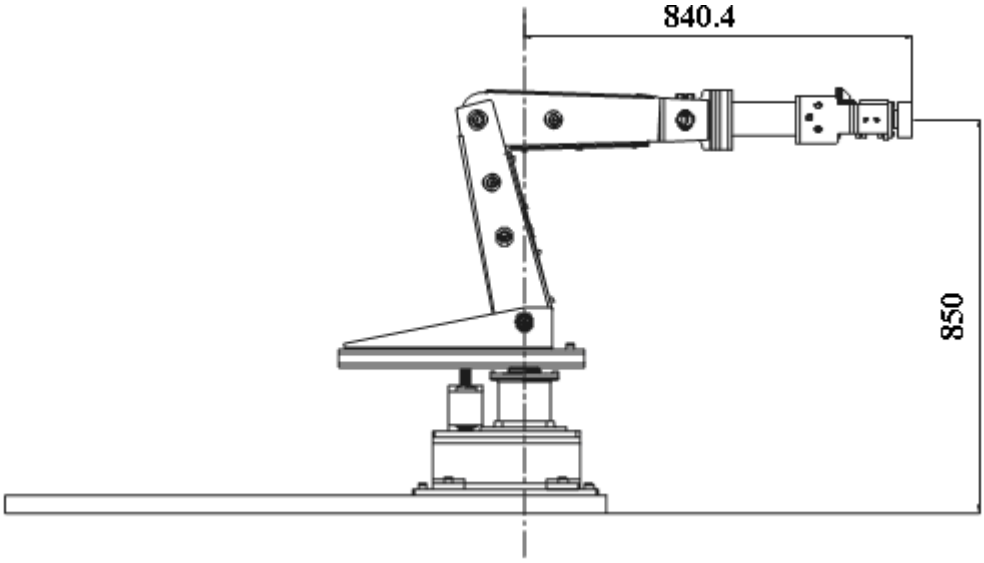


Figura 1.2 Alcance mínimo longitudinal.

1.8.5 Alcance vertical.

El alcance máximo vertical del MIRH1 es una altura de 1540.9 mm a una distancia longitudinal de 302.10 mm tomando como referencia el eje de rotación del motor del eslabón 1 y la base de anclaje, como se muestra en la figura 1.3.

El alcance inferior vertical del MIRH1 está situado a una altura de 80 mm por debajo de la base de anclaje del manipulador, a una distancia de 560.20 mm tomando como referencia el eje de rotación del motor del eslabón 1, como se muestra en la figura 1.4.

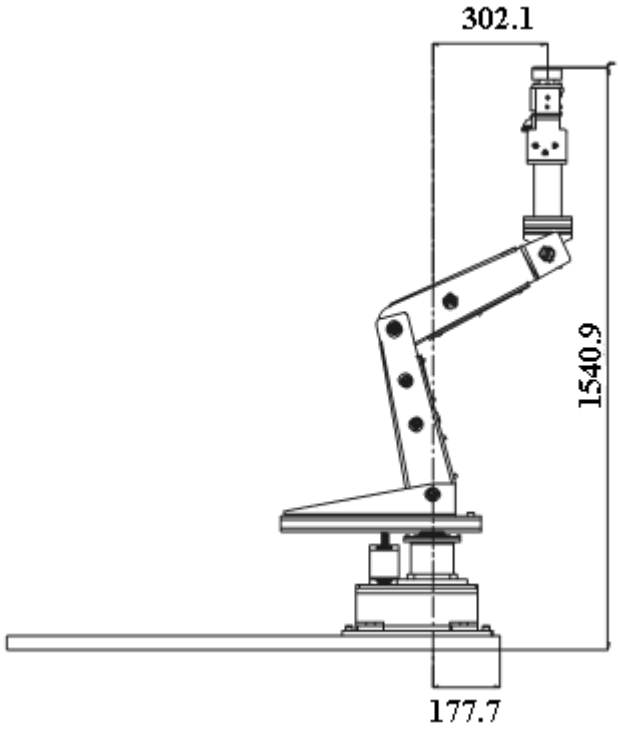


Figura 1.3 Alcance máximo vertical.

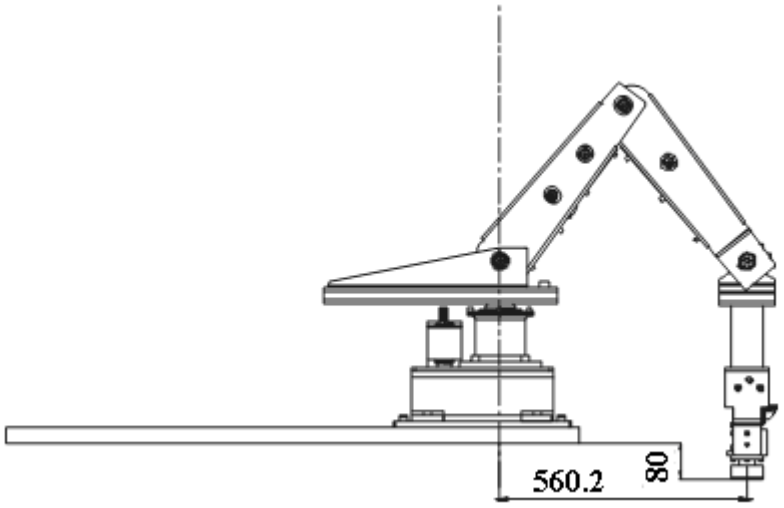


Figura 1.4 Alcance mínimo vertical.

1.8.6 Capacidades de carga máxima.

El MIRH1 tiene dentro de su configuración dos motores hidráulicos ubicados de tal manera que uno de ellos genera movimiento al eslabón 5 y el otro le dará movimiento al efector final una vez que éste sea instalado, estos motores son de capacidades limitadas teniendo torques máximos de 80 N-m y 38 N-m respectivamente.

El movimiento de los eslabones dos y tres es generado a partir de otros dos motores en estos eslabones es donde se requiere la mayor fuerza para mover el manipulador.

CAPÍTULO II.

2.1 HIDRÁULICA.

La palabra "Hidráulica" proviene del griego "hydor" que significa "agua". Hoy el término hidráulica se emplea para referirse a la transmisión y control de fuerzas y movimientos por medio de líquidos, es decir, se utilizan los líquidos para la transmisión de energía, en la mayoría de los casos se trata de aceites minerales pero también pueden emplearse otros fluidos, como líquidos sintéticos, agua o una emulsión agua-aceite.

2.2 Campos de aplicación de la Hidráulica y Neumática.

En la actualidad las aplicaciones de la oleohidráulica y neumática son muy variadas, esta amplitud en los usos se debe principalmente al diseño y fabricación de elementos de mayor precisión y con materiales de mejor calidad, acompañado además de estudios mas acabados de las materias y principios que rigen la hidráulica y neumática. Todo lo anterior se ha visto reflejado en equipos que permiten trabajos cada vez con mayor precisión y con mayores niveles de energía, lo que sin duda ha permitido un creciente desarrollo de la industria en general.

Dentro de las aplicaciones se pueden distinguir dos, móviles e industriales:

2.2 Aplicaciones Móviles.

El empleo de la energía proporcionada por el aire y aceite a presión, puede aplicarse para transportar, excavar, levantar, perforar, manipular materiales, controlar e impulsar vehículos móviles tales como:

- Tractores
- Grúas
- Retroexcavadoras
- Camiones recolectores de basura
- Cargadores frontales
- Frenos y suspensiones de camiones
- Vehículos para la construcción y mantención de carreteras
- Etc.

2.3 Aplicaciones Industriales.

En la industria, es de primera importancia contar con maquinaria especializada para controlar, impulsar, posicionar y mecanizar elementos o materiales propios de la línea de

producción, para estos efectos se utiliza con regularidad la energía proporcionada por fluidos comprimidos. Se tiene entre otros:

- Maquinaria para la industria plástica
- Máquinas herramientas
- Maquinaria para la elaboración de alimentos

Equipamiento para robótica y manipulación automatizada

- Equipo para montaje industrial
- Maquinaria para la minería
- Maquinaria para la industria siderúrgica
- Etc.

Otras aplicaciones se pueden dar en sistemas propios de vehículos automotores, como automóviles, aplicaciones aerospaciales y aplicaciones navales, por otro lado se pueden tener aplicaciones en el campo de la medicina y en general en todas aquellas áreas en que se requiere movimientos muy controlados y de alta precisión, así se tiene:

- Aplicación automotriz: suspensión, frenos, dirección, refrigeración, etc.
- Aplicación Aeronáutica: timones, alerones, trenes de aterrizaje, frenos, simuladores, equipos de mantenimiento aeronáutico, etc.
- Aplicación Naval: timón, mecanismos de transmisión, sistemas de mandos, sistemas especializados de embarcaciones o buques militares
- Medicina: Instrumental quirúrgico, mesas de operaciones, camas de hospital, sillas e instrumental odontológico, etc.

La hidráulica y neumática tienen aplicaciones tan variadas, que pueden ser empleadas incluso en controles escénicos (teatro), cinematografía, parques de entretenciones, represas, puentes levadizos, plataformas de perforación submarina, ascensores, mesas de levante de automóviles, etc.

2.4 Ventajas y desventajas de la Hidráulica y Neumática.

Los sistemas de transmisión de energía oleohidráulicos y neumáticos son una garantía de seguridad, calidad y fiabilidad a la vez que reducen costos.

La Seguridad es de vital importancia en la navegación aérea y espacial, en la producción y funcionamiento de vehículos, en la minería y en la fabricación de productos frágiles. Por ejemplo, los sistemas oleohidráulicos y neumáticos se utilizan para asistir la dirección y el frenado de coches, camiones y autobuses.

Los sistemas de control oleohidráulico y el tren de aterrizaje son los responsables de la seguridad en el despegue, aterrizaje y vuelo de aviones y naves espaciales. Los rápidos avances realizados por la minería y construcción de túneles son el resultado de la aplicación de modernos sistemas oleohidráulicos y neumáticos.

La Fiabilidad y la Precisión son necesarias en una amplia gama de aplicaciones industriales en las que los usuarios exigen cada vez más una mayor calidad. Los sistemas oleohidráulicos y neumáticos utilizados en la manipulación, sistemas de fijación y robots de soldadura aseguran un rendimiento y una productividad elevados, por ejemplo, en la fabricación de automóviles.

En relación con la industria del plástico, la combinación de la oleohidráulica, la neumática y la electrónica hacen posible que la producción esté completamente automatizada, ofreciendo un nivel de calidad constante con un elevado grado de precisión.

Los sistemas neumáticos juegan un papel clave en aquellos procesos en los que la higiene y la precisión son de suma importancia, como es el caso de las instalaciones de la industria farmacéutica y alimenticia, entre otras.

La Reducción en el costo es un factor vital a la hora de asegurar la competitividad de un país industrial.

La tecnología moderna debe ser rentable y la respuesta se encuentra en los sistemas oleohidráulicos y neumáticos. Entre otros ejemplos, cabe citar el uso generalizado de estos sistemas en la industria de carretillas elevadoras controladas hidráulicamente, las máquinas herramientas de alta tecnología, así como los equipos de fabricación para procesos de producción automatizada, las modernas excavadoras, las máquinas de construcción y obras públicas y la maquinaria agrícola.

2.4.1 Ventajas de la Oleohidráulica.

- Permite trabajar con elevados niveles de fuerza o momentos de giro
- El aceite empleado en el sistema es fácilmente recuperable
- Velocidad de actuación fácilmente controlable
- Instalaciones compactas
- Protección simple contra sobrecargas
- Cambios rápidos de sentido

2.4.2 Desventajas de la Oleohidráulica.

- El fluido es mas caro
- Perdidas de carga
- Personal especializado para el mantenimiento
- Fluido muy sensible a la contaminación.

Definiciones:

Fluido: Elemento en estado líquido o gaseoso, en los sistemas neumáticos "aire comprimido y en los sistemas hidráulicos "aceites derivados de petróleo".

Fluidos Hidráulicos: Misión de un fluido en oleohidráulica.

- Transmitir potencia
- Lubricar
- Minimizar fugas

Fluidos empleados.

- Aceites minerales procedentes de la destilación del petróleo
- Agua – glicol
- Fluidos sintéticos
- Emulsiones agua – aceite

Generalidades.

El aceite en sistemas hidráulicos desempeña la doble función de lubricar y transmitir potencia.

2.5 BOMBAS.

Una bomba hidráulica es un dispositivo tal que recibiendo energía mecánica de una fuente exterior la transforma en una energía de presión transmisible de un lugar a otro de un sistema hidráulico a través de un líquido cuyas moléculas estén sometidas precisamente a esa presión. Las bombas hidráulicas son los elementos encargados de impulsar el aceite o líquido hidráulico, transformando la energía mecánica rotatoria en energía hidráulica.

El proceso de transformación de energía se efectúa en dos etapas: aspiración y descarga.

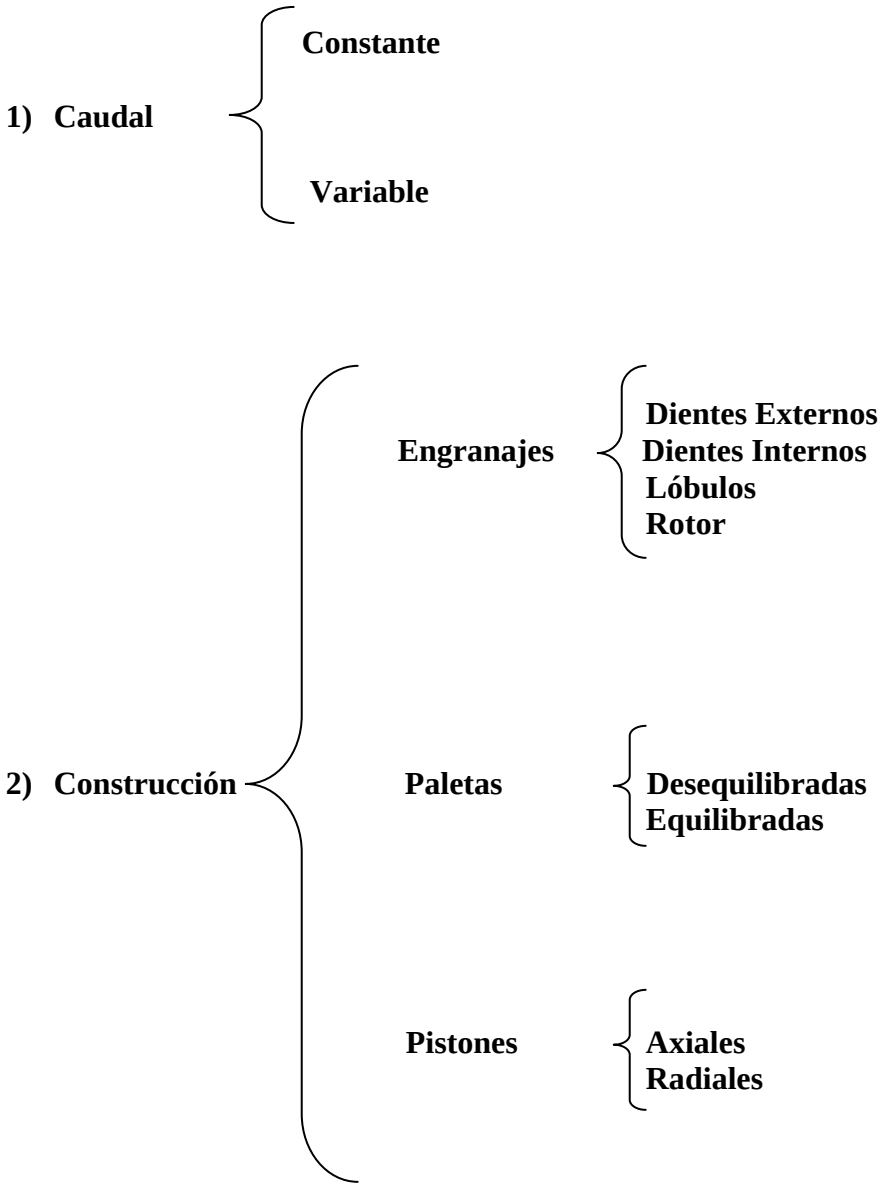
2.5.1 Aspiración.

Al comunicarse energía mecánica a la bomba, ésta comienza a girar y con esto se genera una disminución de la presión en la entrada de la bomba, como el depósito de aceite se encuentra sometido a presión atmosférica, se genera entonces una diferencia de presiones lo que provoca la succión y con ello el impulso del aceite hacia la entrada de la bomba.

2.5.2 Descarga.

Al entrar aceite, la bomba lo toma y lo traslada hasta la salida y se asegura por la forma constructiva que el fluido no retroceda. Dado esto, el fluido no encontrará mas alternativa que ingresar al sistema que es donde se encuentra espacio disponible, consiguiéndose así la descarga.

2.6 Clasificación de las Bombas.



Las bombas pueden clasificarse además dependiendo de la forma en que se desplaza la parte móvil de éstas; si el desplazamiento es rectilíneo y alternado, entonces se llamarán oscilantes, y si el elemento móvil gira se llamarán rotativas. En la figura 2.1 se muestra nuestra bomba.

Fig. 2.1 Bomba de aceite.

2.7 VÁLVULAS.

2.7.1 Válvula de tres vías, válvula giratoria o rotativa.

Esta es una válvula que cambia la orientación de la corriente del fluido. En esta válvula como su nombre; lo indica, hay tres bocas de conexión o "puertas", la primera por donde entra la presión desde la bomba, la segunda que se comunica con el cilindro hidráulico y la tercera que es la conexión hacia el tanque o retorno.

Estas válvulas distribuidoras son de inversión axial. Existe otra configuración, que es la inversión rotativa. La figura siguiente, muestra una válvula de tres vías y dos posiciones. El rotor gira 180º para carga o descarga del aceite.

En la figura 2.2 se muestra un corte de una válvula de tres vías en las dos posiciones en que aquella trabaja como A y B, en una de esas posiciones la corredera o husillo permite comunicar la puerta de entrada de presión con la salida del cilindro, mientras bloquea el retorno al tanque, en la segunda posición, o sea con la corredera situada en el otro extremo la misma bloquea ahora la entrada de presión y conecta el retorno a tanque con el cilindro.

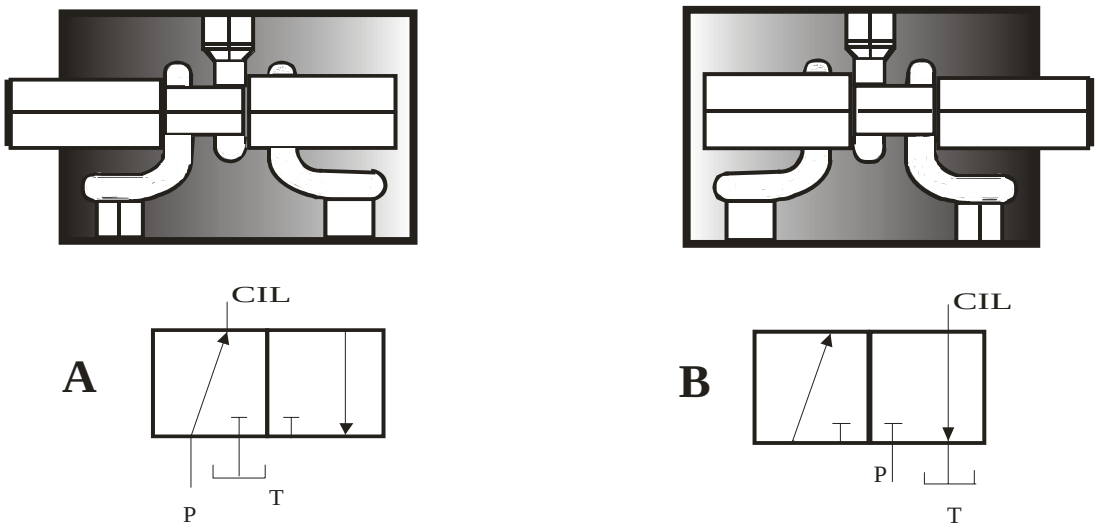


Fig.2.2 Válvula tres vías dos posiciones.

En una válvula de dos posiciones, una de ellas se logra mediante un resorte que mantiene la corredera en una posición extrema, la posición se logra por una señal de mando, que puede ser, manual, mecánica, eléctrica o por piloto hidráulico o neumático, que al producirse provocan el deslizamiento del husillo al lado opuesto, venciendo la tensión del resorte al comprimirlo.

Esta válvula se emplea para controlar el accionamiento de cilindros de simple efecto y émbolos buzo, cuyo retorno se efectúa por la acción de un resorte a cargas exteriores que no requiere retorno hidráulico.

2.7.2 Válvulas de cuatro vías dos posiciones.

Cuando se trata de gobernar cilindros hidráulicos de doble efecto, o motores hidráulicos que requieren control direccional de flujo en ambos sentidos de circulación, debe emplearse una válvula de cuatro vías. En esta unidad existen cuatro bocas de conexión, la primera conectada a la entrada de presión, la segunda conectada al tanque y las dos restantes conectadas respectivamente a ambas caras del cilindro de doble efecto que deben gobernar.

En la válvula de cuatro vías, dos posiciones, como su nombre lo indica, la corredera o husillo estará únicamente situada en cualquiera de ambas posiciones extremas, a un lado o al otro.

Cuando la válvula no este actuada, la presión P se comunica con la cara 1 del cilindro mientras que la cara 2 se encuentran conectada a la descarga del tanque T. Al invertir la posición del husillo, tal como observamos en la figura 2.3, también se invierten las conexiones y ahora la presión P está conectada a la cara 2 del cilindro mientras que la 1 se conecta a la descarga T.

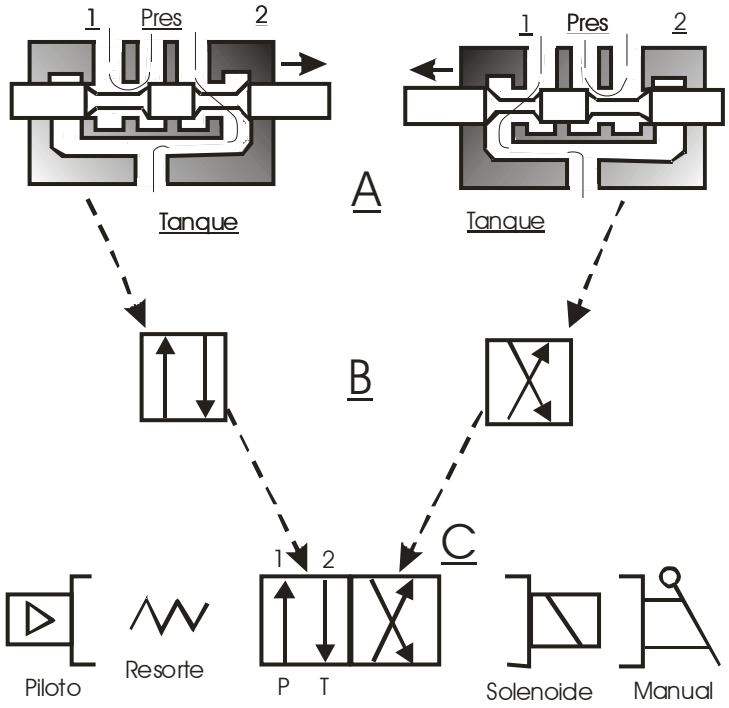


Fig. 2.3 Válvula cuatro vías dos posiciones.

En la figura 2.3 se ve el corte esquemático de una válvula de cuatro vías, dos posiciones, mostrándose el conexionado interno del cuerpo.

Para el dibujo de los circuitos hidráulicos, y permitir su fácil lectura, se ha adoptado un sistema de símbolos de acuerdo a lo indicado por el USA Standard Institute (conocido como USASI). Los esquemas propuestos por este instituto difieren ligeramente de los propuestos por el Joint Industrial Comitee, conocido como JIG.

En la figura 2.3, se ve como se genera la simbología para representar a una válvula de cuatro vías, dos posiciones. En la parte A se muestra el corte esquemático de la válvula con su corredera en sus posiciones a toda derecha y toda izquierda respectivamente. En la parte B la figura muestra mediante la representación simbólica el conexionado que se opera en el interior del cuerpo de la válvula, al cambiar la corredera de posición dibujando dos cuadros que al unirse como se muestra en la parte C del mismo dibujo, nos representan a la válvula con sus dos conexionados posibles. Para completar el símbolo, otros pequeños rectángulos se dibujan en cada costado con el fin de indicar el tipo de comando empleado para gobernar la válvula.

2.7.3 Válvula de cuatro vías tres posiciones.

Este es el tipo más popular y más conocido de válvulas de cuatro vías y la utilizada en nuestro proyecto. Aquí, la corredera, aparte de tener dos posiciones extremas, también puede

permanecer detenida en el centro mismo del cuerpo de la válvula, mediante un sistema de centrado por resorte o retención de bolilla u otro medio de retención mecánica.

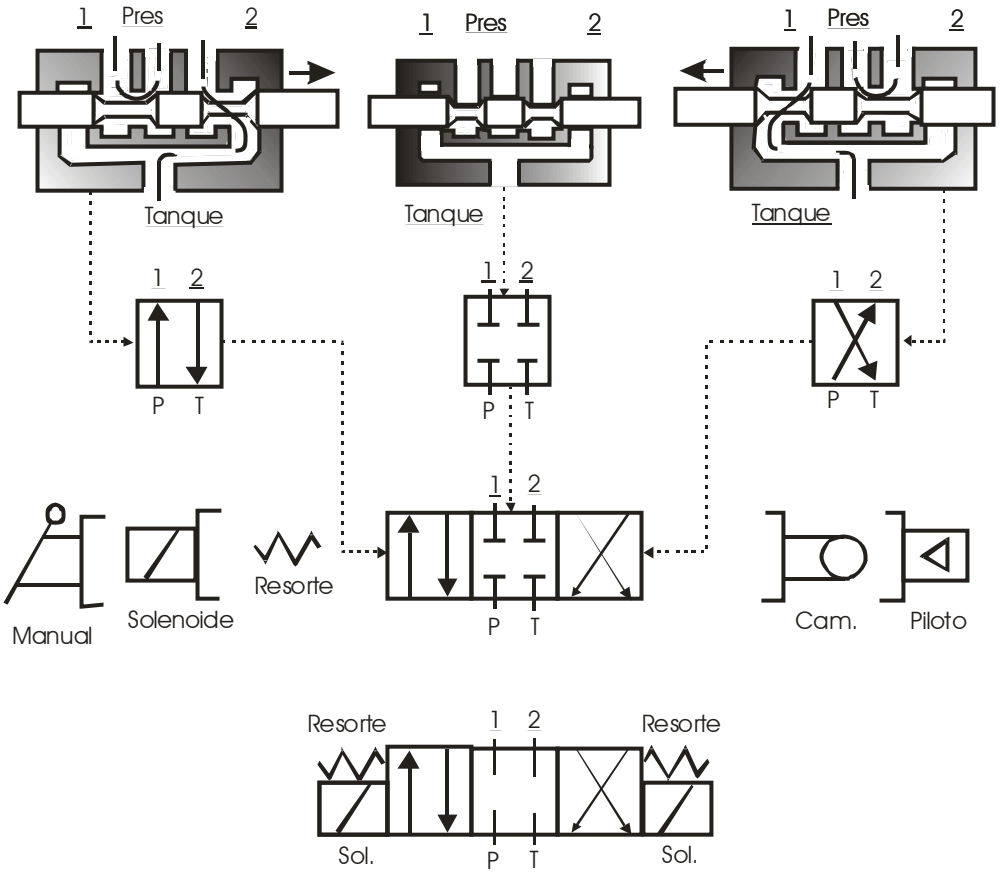


Fig. 2.4 Válvula cuatro vías tres posiciones.

Símbolo gráfico completo de una válvula de cuatro vías tres posiciones, accionada a doble solenoide y centrada por medio de resortes.

En este tipo de válvula, cuando la misma NO ESTA ACTUADA, la corredera se encuentra situada en su posición central. Al actuarse sobre la válvula el mando correspondiente a un extremo y al otro, la corredera se deslizará en un sentido o en el otro.

Es necesario destacar que el sistema de conexionado de las bocas o " puertas" de la válvula de cuatro vías en el cuerpo de la misma es SIEMPRE EL MISMO cualquiera que sea el fabricante que manufacturan las puertas vienen marcadas SIEMPRE P T A y B. El símbolo de esta válvula es esencialmente idéntico al símbolo de una válvula de cuatro vías, dos posiciones con la salvedad que se ha adicionado un tercer cuadrado entre los otros dos, y por tal razón al encontrarse en una posición central simboliza la posición central de la corredera, que es la TERCERA posición.

Además, el símbolo se completa adicionando en ambos extremos los rectángulos correspondientes para señalar que tipo de actuación que se emplea para gobernar la válvula.

2.7.4 Válvulas hidráulicas con solenoides.

Las necesidades crecientes que se presentaron y que se siguen presentando en el campo de la automatización industrial en cuanto a la fabricación de maquinarias, dispositivos y diversos elementos accionados hidráulicamente, y la extrema de sencillez con que se pueden diseñar circuitos eléctricos que funcionan automáticamente, comandados desde sencillos microcontactos, temporizadores, hasta los modernos controladores lógicos programables (PLC's) han hecho pensar a los Ingenieros Proyectistas hace algunas décadas atrás lo útil que resultaría comandar circuitos hidráulicos vía automatizaciones eléctricas.

Ello determinó en su momento la creación de la válvula de control direccional accionada por solenoides y/o electroimanes, y actualmente, este tipo de válvulas es el elemento indispensable para comandar cualquier máquina hidráulica, por medio de cualquier tipo de accionamiento eléctrico y/o electrónico.

2.7.5 Válvulas hidráulicas de cuatro vías, operadas eléctricamente.

En la Fig. 2.5, vemos una válvula directamente accionada por solenoide, que es aquella en la cual el elemento motriz para accionar la corredera deslizante es un electroimán o un solenoide.

La acción de este, cuando se encuentra energizado, se traduce en un empuje o una tracción de la corredera. En dicha figura tenemos una válvula de cuatro vías, dos posiciones, de retorno por la acción de un resorte antagonista, y accionada por el electroimán dibujado al costado derecho de la válvula.

Cuando se energiza el solenoide la corredera es empujada por la acción de este hacia la izquierda, conectando la presión a la cara 2 del cilindro mientras que la cara 1 queda drenada al tanque. La corriente eléctrica debe ser mantenida sobre el solenoide para que este a su vez mantenga a la corredera empujada totalmente hacia la izquierda. Cuando se corta la corriente y el solenoide se desenergiza, el resorte empuja enérgicamente a su vez a la corredera hacia la derecha conectándose entonces las puertas del cuerpo de la válvula.

Fig. 2.4 Válvula cuatro vías tres posiciones.

LAS VÁLVULAS SOLENOIDES SIEMPRE SE REPRESENTAN EN LOS ESQUEMAS DE CIRCUITERIA CON EL CONEXIONADO CORRESPONDIENTE A SU POSICIÓN DESENERGIZADA.

En las válvulas de control direccional directamente comandadas por solenoides, para dimensiones de tubería de 1/4", cuando son manufacturadas por fabricantes acreditados permiten caudales de pasaje de fluido de hasta 30 litros por minuto, para presiones de 1000 libras por pulgada cuadrada.

Algunas veces suele suceder, que la válvula operada por un breve impulso eléctrico y al cesar este, debe seguir la corredera permaneciendo en el lugar al cual fue puesta, Evidentemente en este caso no puede tolerarse la acción del resorte antagonista por tal motivo se reemplaza a este por otra solenoide, de manera que la corredera es movida hacia un extremo o el otro de la válvula por la acción del empuje de uno u otro solenoide.

Debe tomarse especial cuidado, cuando se trabaja con esta válvula, de no montarla en ninguna otra parte o posición que no sea la horizontal como también, si la válvula se encuentra colocada en una máquina móvil de no fijarla nunca con la corredera paralela al sentido del movimiento. En el primer caso la gravedad, y en el segundo la inercia misma de la corredera, en el caso de una frenada brusca de las máquinas podrá descolocar la corredera de una posición determinada, motivando la aparición de inconvenientes a veces difíciles de evaluar. Así mismo, se deben de tener precauciones, para que en ningún caso ambos solenoides se energicen simultáneamente.

2.8 Construcción:

La válvula se compone básicamente de:

- carcasa (1) con superficie de acople
- pistón de mando (2) con resortes de presión (3 y 4)
- solenoides (5 y 6) con rosca central
- opcionalmente electrónica integrada (7)

2.9 Funcionamiento:

- Con solenoides desenergizados (5 y 6) posición central del pistón de mando (2) por resortes de presión (3 y 4)

- Accionamiento directo del pistón de mando (2) al energizar un solenoide proporcional por ejemplo solenoide "b" (6)

- Desplazamiento del pistón de mando (2) hacia la izquierda proporcionalmente a la señal eléctrica de entrada

- Conexión de P hacia A y B hacia T a través de secciones tipo diafragma con característica de flujo progresiva
- desconexión del solenoide (6)

- El pistón de mando (2) es conducido nuevamente a la posición de reposo por el resorte (3)

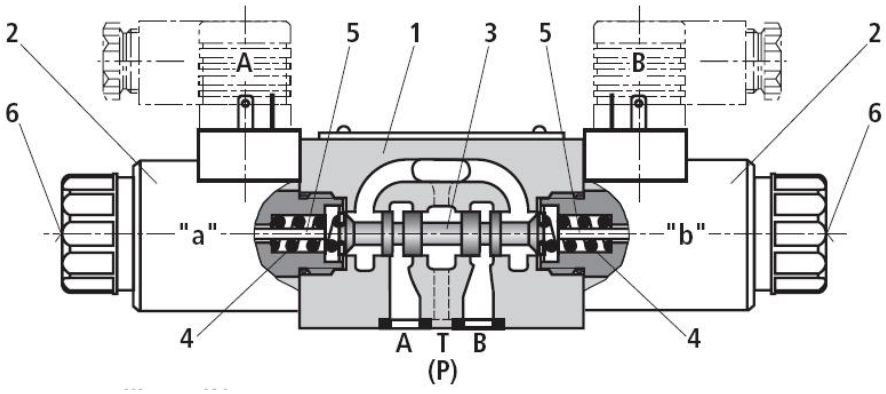


Fig. 2.6 Diagrama interno válvula cuatro vías tres posiciones.

Fig. 2.7 Foto válvula cuatro vías tres posiciones.

CAPÍTULO III.

3.1 HARDWARE.

3.2 Microcontrolador.

Un microcontrolador es un circuito o chip que incluye en su interior las tres unidades funcionales de una computadora: Unidad Central de Procesamiento (CPU), Memoria y Unidades de E/S, es decir, se trata de una computadora completa en un solo circuito integrado.

Un microcontrolador difiere de una CPU normal, debido a que es más fácil convertirla en una computadora en funcionamiento, con un mínimo de chips externos de apoyo. La idea es que el chip se coloque en el dispositivo, enganchado a la fuente de energía y de información que necesite, y eso es todo. Un microprocesador tradicional no le permitirá hacer esto, ya que espera que todas estas tareas sean manejadas por otros chips.

Por ejemplo, un microcontrolador típico tendrá un generador de reloj integrado y una pequeña cantidad de memoria RAM y ROM/EPROM/EEPROM, significando que para hacerlo funcionar, todo lo que se necesita son unos pocos programas de control y un cristal de sincronización. Los microcontroladores disponen generalmente también de una gran variedad de dispositivos de entrada/salida, como convertidor de analógico a digital, temporizadores, puertos seriales y buses de interfaz serie especializados, como I^2C y CAN. Frecuentemente, estos dispositivos integrados pueden ser controlados por instrucciones de procesadores especializados. Los modernos microcontroladores frecuentemente incluyen un lenguaje de programación integrado, como Visual BASIC que se utiliza bastante con este propósito.

Los microcontroladores negocian la velocidad y la flexibilidad para facilitar su uso. Debido a que se integran múltiples accesorios en el chip para incluir funcionalidad, como los dispositivos de entrada/salida o la memoria, el microcontrolador, puede prescindir de cualquier otra circuitería.

3.3 ¿Qué es un PIC?

Los 'PIC' son una familia de microcontroladores tipo RISC (Set Reducido de Instrucciones) fabricados por Microchip Technology Inc. y derivados del PIC1650, originalmente desarrollado por la división de microelectrónica de General Instruments.

El nombre actual no es un acrónimo. En realidad, el nombre completo es PICmicro, aunque generalmente se utiliza como Peripheral Interface Controller (Controlador de Interfaz Periférico).

El PIC original se diseñó para ser usado con la nueva CPU de 16 bits CP16000. Siendo en general una buena CPU, ésta tenía malas prestaciones de E/S, y el PIC de 8 bits se desarrolló en 1975 para mejorar el rendimiento del sistema quitando peso de E/S a la CPU. El PIC utilizaba micro-código simple almacenado en ROM para realizar estas tareas; y aunque el término no se usaba por aquel entonces, se trata de un diseño RISC que ejecuta una instrucción cada 4 ciclos del oscilador.

En 1985, dicha división de microelectrónica de General Instruments se convirtió en una filial y el nuevo propietario canceló casi todos los desarrollos, que para esas fechas la mayoría estaban obsoletos. El PIC, sin embargo, se mejoró con EPROM para conseguir un controlador de canal programable. Hoy en día multitud de PICs vienen con varios periféricos incluidos (módulos de comunicación serie, núcleos de control de motores, etc.) y con memoria de programa desde 512 a 32.000 palabras (una palabra corresponde a una instrucción en ensamblador, y puede ser 12, 14 o 16 bits, dependiendo de la familia específica del microcontrolador).

Fig. 3.1 Microcontroladores MICROCHIP.

3.4 Set de instrucciones.

El PIC usa un set de instrucciones tipo RISC, cuyo número puede variar desde 35 para PICs de gama baja a 70 para los de gama alta. Las instrucciones se clasifican entre las que realizan operaciones entre el acumulador y una constante, entre el acumulador y una posición de memoria, instrucciones de condicionamiento y de salto/retorno, implementación de interrupciones y una para pasar a modo de bajo consumo llamada “estado dormido”.

Microchip proporciona un entorno de desarrollo Licencia Libre (freeware), llamado MPLAB que incluye un simulador de software y un ensamblador. Otras empresas desarrollan compiladores C y BASIC. Microchip también vende compiladores para los PICs de gama alta

("C18" para la serie F18 y "C30" para los Procesadores Digitales de Señal DSPICs) y se puede descargar una edición para estudiantes del C18 que inhabilita algunas opciones después de un tiempo de evaluación.

Para Pascal existe un compilador de código abierto, JAL, lo mismo que PicForth para el lenguaje Forth. GPUTILS es una colección de herramientas distribuidas bajo licencia para el publico en general (GNU), que incluye ensamblador y enlazador, y funciona en Linux, MacOS y Microsoft Windows. GPSIM es otra herramienta libre que permite simular diversos dispositivos hardware conectados al PIC.

TABLE 15-2: PIC16F87XA INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add Literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND Literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call Subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to Address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR Literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move Literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from Interrupt	2	00	0000	0000	1001		
RETLW	k	Return with Literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into Standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from Literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR Literal with W	1	11	1010	kkkk	kkkk	Z	

Tabla. 3.1 Set de instrucciones.

3.5 Programación de un PIC.

Para transferir el código de un ordenador al PIC normalmente se usa un dispositivo llamado programador. La mayoría de PICs que Microchip distribuye hoy en día incorporan ICSP (In Circuit Serial Programming, programación serie incorporada) o LVP (Low Voltage Programming, programación a bajo voltaje), lo que permite programar el PIC directamente en el circuito destino. Para la ICSP se usan los pines RB6 y RB7 como reloj y datos y el MCLR para activar el modo programación aplicando un voltaje de unos 11 volts. Existen muchos programadores de PICs, desde los más simples que dejan al software los detalles de comunicaciones, a los más complejos, que pueden verificar el dispositivo a diversas tensiones de alimentación e implementan en hardware casi todas las funcionalidades. Muchos de estos programadores complejos incluyen ellos mismos PICs preprogramados como interfaz para enviar las órdenes al PIC que se desea programar. Uno de los programadores más simples es el TE20, que utiliza la línea TX del puerto RS232 como alimentación y las líneas DTR y CTR para mandar o recibir datos cuando el microcontrolador está en modo programación. El software de programación puede ser el ICProg, muy común entre la gente que utiliza este tipo de microcontroladores.

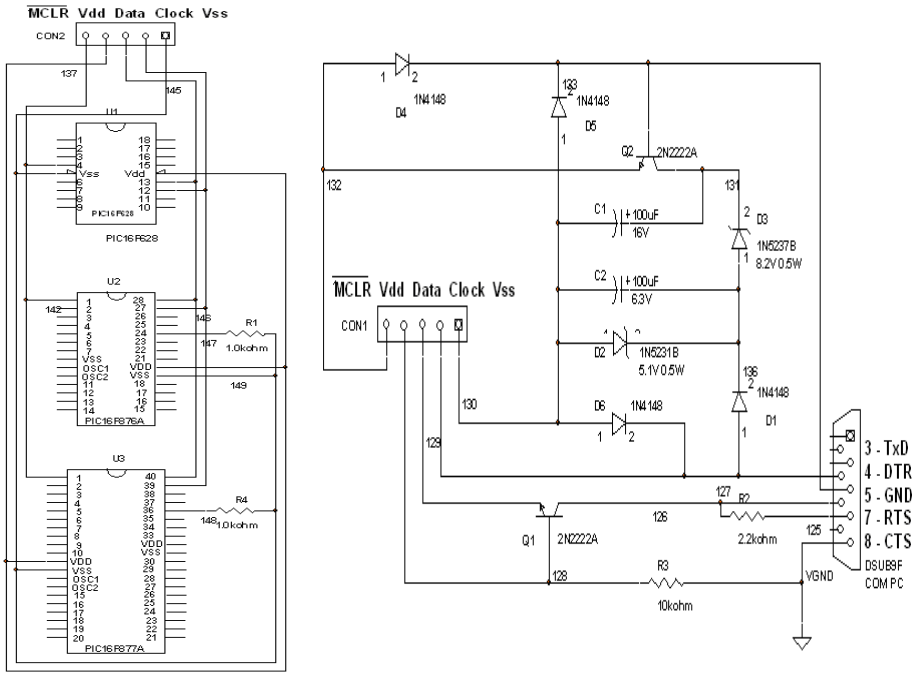


Fig. 3.2 Programador de PICs.

3.6 Tamaño de palabra.

El tamaño de palabra de los microcontroladores PIC es fuente de muchas confusiones. Todos los PICs (excepto los dsPIC) manejan datos en trozos de 8 bits, con lo que se deberían llamar microcontroladores de 8 bits. Pero a diferencia de la mayoría de CPUs, el PIC usa arquitectura Harvard, por lo que el tamaño de las instrucciones puede ser distinto del de la palabra de datos. De hecho, las diferentes familias de PICs usan tamaños de instrucción distintos, lo que hace difícil comparar el tamaño del código del PIC con el de otros microcontroladores. Por ejemplo, un microcontrolador tiene 6144 bytes de memoria de programa: para un PIC de 12 bits esto significa 4096 palabras y para uno de 16 bits, 3072 palabras.

3.7 Características de los PIC's.

Los PICs actuales vienen con una amplia gama de mejoras de hardware incorporadas:

- Núcleos de CPU de 8/16 bits con Arquitectura Harvard modificada
- Memoria Flash y ROM disponible desde 256 bytes a 256 kbytes
- Puertos de E/S (típicamente 0 a 5,5 voltios)
- Temporizadores de 8/16 bits
- Tecnología Nanowatt para modos de control de energía
- Periféricos serie síncronos y asíncronos
- Conversores analógico/digital de 10-12 bits
- Comparadores de tensión
- Módulos de captura y comparación, modulación de ancho de pulso (PWM)
- Controladores para pantalla de cristal liquido (LCD)
- Periféricos para comunicaciones I^2C, SPI, y I^2S
- Memoria EEPROM interna con duración de hasta un millón de ciclos de Lectura escritura (R/W)
- Periféricos de control de motores
- Soporte de interfaz bus serial universal (USB)
- Soporte de controlador Ethernet
- Soporte de controlador CAN
- Soporte de controlador LIN
- Soporte de controlador Irda

3.8 Compilador en C para PICS.

El compilador PicC C Compiler soporta una gran cantidad de microcontroladores PIC. Soporta todos los dispositivos de la familia de 14 bits. El compilador manejara en forma automática el registro PCLATH y la selección de bancos de memoria. El PicC C también posee un optimizador.

El PicC C Compiler fue desarrollado para cumplir con las especificaciones del lenguaje ANSI C. El compilador produce tres tipos de archivos. Archivos con extensión .hex que le permitirá grabar el programa ejecutable en el PIC por medio del uso de un programador. El archivo con extensión .asm contendrá un listado en ensamblador (assembler) del programa compilado con la información del mapeo de memoria. Estos archivos son muy útiles para la comprobación (debugging) de los programas y para determinar la cantidad de pasos de programas (ciclos de ejecución) que tiene la aplicación. Los archivos con extensión .pre contienen la información preprocesada del programa, #defines, #includes, etc. La cual es expandida y guardada en el archivo.

Es el producto es ideal para aquellas personas que les gusta desarrollar en bajo nivel con los recursos de un lenguaje de alto nivel como el C. Se recomienda ser utilizado por personas vinculadas con la programación y sintaxis de C.

Beneficios.

- Está basado en el ANSI C.
- Soporte completo de la familia de microcontroladores PIC de 14 bit.
- Salida Ensamblador.
- Formato Hexadecimal de Intel de 8 bits (INHX8M).
- Soporta interrupciones.
- Tipos de datos 8 y 16 bits - int, char, long, pointers, unsigned, etc.
- Inserción de código ensamblador .asm;
- Todos los operadores aritméticos - incluyendo multiplicación, división, modulo y otros.
- Las variables y funciones no utilizadas son borradas automáticamente.
- Reutilización de RAM.
- Dispositivos soportados: 16F84, 16C83, 16C554, 16C556, 16C558, 16C61, 16C62, 16C620, 16C621, 16C622, 16C63, 16C64, 16C641, 16C642, 16C65, 16C66, 16C661, 16C662, 16C67, 16C71, 16C710, 16C711, 16C715, 16C72, 16C73, 16C74, 16C76, 16C77, 16C9xx, 14C000
- Nuevos! : 16CE623, 16CE624, 16CE625, 12C671, 12C672, 12C673, 12C674, 16F873, 16F874, 16F876, 16F877
- Notas de aplicación.

3.9 ¿Qué es RS-232?.

Es una interfaz que designa una norma para el intercambio serie de datos binarios entre un DTE (Equipo terminal de datos) y un DCE (Data Communication Equipment, Equipo de Comunicación de datos), aunque existen otras situaciones en las que también se utiliza la interfaz RS-232.

En particular, existen ocasiones en que interesa conectar otro tipo de dispositivos como pueden ser computadoras. Evidentemente, en el caso de interconexión entre los mismos, se requerirá la conexión de un DTE con otro DTE.

El RS-232 consiste en un conector tipo DB-25, aunque es normal encontrar la versión de 9 pines, más extendido para cierto tipo de periféricos.
RS-232 también conocido como Electronic Industries Alliance RS-232C
RS-232 (Estándar ANSI/EIA-232) es el conector serial que se encuentra en las PCs compatibles con IBM. Los ingenieros lo utilizan con diversos propósitos, como el conectar periféricos, impresoras, o módems, así como para instrumentación industrial.

Debido a la línea del controlador (driver) y mejoras en el cableado, las aplicaciones comúnmente incrementan el desempeño del RS-232 más allá de la distancia y velocidad listadas en el estándar. El RS-232 está limitado a conexiones punto a punto entre puertos seriales y dispositivos PC. Se puede utilizar el hardware RS-232 para comunicaciones seriales en distancias de hasta 50 pies.

La tabla muestra las señales más comunes del RS-232

Señal		**DB-25**	**DB-9** (**TIA-574**)	**EIA**/**TIA** **561**	Yost	**RJ-50**	**MMJ**
Common Ground	G	7	5	4	4,5	6	3,4
Transmitted Data	TD	2	3	6	3	8	2
Received Data	RD	3	2	5	6	9	5
Data Terminal Ready	DTR	20	4	3	2	7	1
Data Set Ready	DSR	6	6	1	7	5	6
Request To Send	RTS	4	7	8	1	4	-
Clear To Send	CTS	5	8	7	8	3	-
Carrier Detect	DCD	8	1	2	7	10	-
Ring Indicator	RI	22	9	1	-	2	-

Tabla. 3.2 Señales del puerto serie.

La RS-232 C tiene una limitación de distancia máxima de 15 metros. Si bien no es una desventaja considerable cuando los equipos a conectar se encuentran cerca, sí es un inconveniente cuando la RS-232 se utiliza para conectar directamente terminales o impresoras que puedan estar lejanas.

3.10 Conector DB-9.

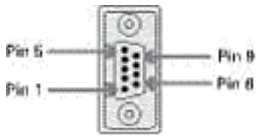


Fig. 3.3 Puerto serie.

3.10.1 Funciones del Conector.

Datos:	TxD en pin 3, RxD en pin 2
Intercambio de Pulsos de Sincronización:	RTS en pin 7, CTS en pin 8, DSR en pin 6, DCD en pin 1, DTR en pin 4
Común:	Com en pin 5
Otros:	RI en pin 9

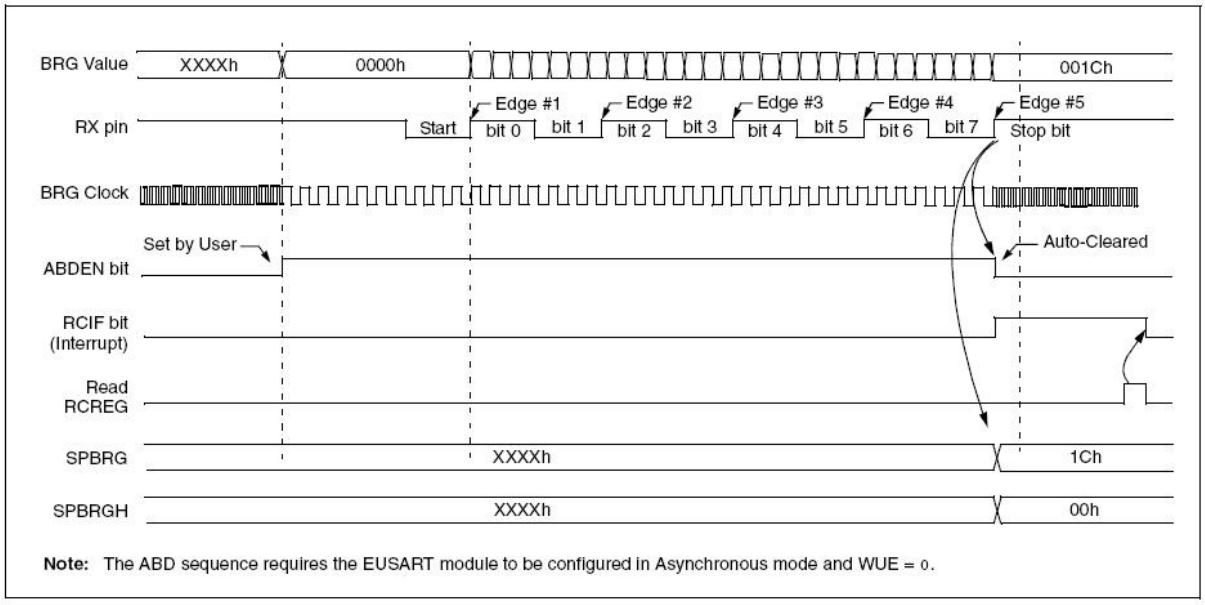


Fig. 3.4 Diagrama de tiempos.

3.11 Comunicación Serial.

Un dispositivo serial utiliza un protocolo de comunicación que es estándar para casi cualquier PC. La mayoría de las computadoras incluyen dos puertos seriales RS-232. Serial es también un protocolo de comunicación para instrumentación en muchos dispositivos, y muchos dispositivos compatibles GPIB vienen con un puerto RS-232.

El concepto de comunicación serial es simple. El puerto serial envía y recibe bytes de información, un bit a la vez. Aunque esto es más lento que la comunicación paralela, la cual permite la transmisión entera de bytes de una sola vez, es más sencillo y puede utilizarlo en distancias grandes. Por ejemplo, las especificaciones IEEE 488 para comunicación paralela definen que el cableado entre equipos no debe ser mayor de 20 m en total, con no más de 2 m entre dos dispositivos cualesquiera.

Típicamente, los ingenieros utilizan serial para transmitir datos ASCII. Completan la comunicación utilizando tres líneas de transmisión – referencia, transmisión, y recepción. Debido a que serial es asíncrono, el puerto puede transmitir datos en una línea mientras recibe datos en otra. Otras líneas están disponibles para el intercambio de pulsos de sincronización pero no son requeridas. Las características seriales importantes son: tasa de baudios, bits de datos, bits de paro, y paridad. Para que dos puertos se comuniquen, estos parámetros deben igualarse:

La tasa de baudios es una unidad de medición para comunicación que indica el número de bits transferidos por segundo. Por ejemplo, 300 baudios son 300 bits por segundo. Cuando los ingenieros se refieren a un ciclo de reloj, se refieren a la tasa en baudios, así que si el protocolo indica una razón en baudios de 4800, el reloj está ejecutándose a 4800 Hz. Esto quiere decir que el puerto serial está muestreando la línea de datos a 4800 Hz. Las tasas de baudios para líneas telefónicas son 14400, 28800, y 33600. Tasas de baudios mayores a estas son posibles, pero reducen la distancia disponible para la separación de dispositivos. Utilizan estas tasas de baudios para comunicación donde los dispositivos están localizados entre sí, como sucede típicamente con los dispositivos GPIB.

Bits de datos son mediciones de los bits de datos actuales en una transmisión. Cuando una computadora envía un paquete de información, la cantidad de datos actuales puede ser que no complete 8 bits. Los valores estándar para los paquetes de datos son de 5, 7, y 8 bits. El marco que usted elija dependerá de la información que está transfiriendo. Por ejemplo, el ASCII estándar tiene valores de 0 a 127 (7 bits). El ASCII extendido utiliza de 0 a 255 (8 bits). Si los datos que usted está transfiriendo se encuentran en texto simple (ASCII estándar), enviar 7 bits de datos por paquete, es suficiente para la comunicación. Un paquete se refiere a la transferencia de un sólo byte, incluyendo los bits de inicio/paro, bits de datos, y paridad. Debido a que el número de bits actuales depende del protocolo seleccionado, puede utilizar el término “paquete” para cubrir todas las instancias.

Los bits de paro son utilizados para señalar el término de comunicaciones en un paquete sencillo. Los valores típicos son 1, 1.5 y 2 bits. Debido a que los datos se encuentran sincronizados a través de las líneas y cada dispositivo tiene su propio reloj, es posible que los dos dispositivos pierdan sincronización. Por lo tanto, los bits de paro no solamente indican el

final de una transmisión, también le da un margen de error a las velocidades de reloj de la computadora. A medida que se utilizan más bits para bits de paro, mayor oportunidad para sincronizar los diferentes relojes, pero más lenta la razón de transferencia de datos.

Paridad es una forma de revisión de error simple utilizada en la comunicación serial. Existen cuatro tipos de paridad – pares, impares, marcados y espaciados. También puede utilizar los que excluyen de paridad. Para paridad impar y par, el puerto serial fija el bit de paridad (el último bit después de los bits de datos) a un valor que asegura que la transmisión tenga un número par o impar de bits lógicos. Por ejemplo, si el dato es 011, para paridad par, el bit de paridad es 0 para mantener el número par de bits altamente lógicos. Si la paridad es impar, el bit de paridad es 1, resultando en 3 bits altamente lógicos. La paridad marcada y espaciada no revisa específicamente los bits de datos, simplemente fija la paridad de los bits como alta para la paridad marcada o baja para la paridad espaciada. Esto permite que el dispositivo receptor conozca el estado de un bit para así determinar si el ruido esta corrompiendo los datos o si los relojes del dispositivo de transmisión y recepción se encuentran fuera de sincronización.

3.11.1 ¿Qué es un intercambio de pulsos de sincronización?

Este método de comunicación RS-232 permite una conexión sencilla de tres líneas: Tx, Rx, y tierra. Sin embargo, para que los datos sean transmitidos, ambos lados deben presentar los datos de forma sincronizada a la misma tasa de baudios. Aunque este método es suficiente para algunas aplicaciones, se encuentra limitado para resolver problemas como receptores sobrecargados. Aquí es donde los seriales tipo intercambio de pulsos de sincronización (handshake) pueden ayudar. Tres formas de intercambio de pulsos de sincronización son las más populares con RS-232: intercambio de pulsos de sincronización por software, intercambio de pulsos de sincronización por hardware y Xmodem.

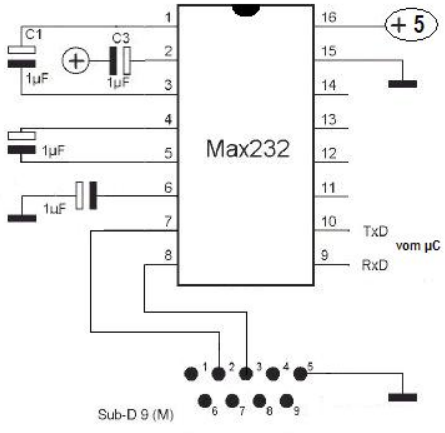


Fig. 3.5 Convertidor de voltaje MAX232.

3.12 Comunicación en Paralelo.

3.12.1 Puerto paralelo.

Los ingenieros de IBM acoplaron un conector de 25 pines, el **DB-25**, con un conector Centronics de 36 pines para crear un cable especial que conectara la impresora con la computadora. Otros fabricantes de impresoras terminaron adoptando la interfase Centronics, haciendo de este extraño cable híbrido un improbable estándar.

Cuando una PC manda datos a una impresora u a otros dispositivos usando el puerto paralelo, esta manda **8 bits** de datos (1 byte) a la vez, de forma distinta al puerto **serie** el cual manda los 8 bits uno detrás de otro por el mismo cable. El puerto paralelo estándar es capaz de mandar de 50 a 100 kilobytes de datos por segundo.

Desde el punto de vista del software, el puerto paralelo son tres registros de 8 bits cada uno, ocupando tres direcciones de I/O consecutivas de la arquitectura x86.

Desde el punto de vista hardware, el puerto es un conector hembra DB25 con doce salidas latch (que tienen memoria/buffer intermedio) y cinco entradas, con 8 líneas de masa.

La función normal es transferir datos a una impresora a través de las 8 líneas de datos, usando las señales restantes como control de flujo.

3.12.2 Descripción general.

La especificación original para el puerto paralelo era unidireccional, esto quiere decir que la información solamente puede viajar en una dirección por cada pin. Con la introducción del PS/2 en 1987, IBM ofreció un nuevo diseño de puerto paralelo bidireccional. Este modo es comúnmente conocido como Puerto paralelo estándar (SPP de Standard Parallel Port) y ha reemplazado completamente el diseño original.

La comunicación bidireccional permite a cada dispositivo recibir datos así como también transmitir. Muchos dispositivos usan los 8 pines (del 2 al 9) originalmente diseñados para datos. Usando los mismos 8 pines limita la comunicación a half-duplex, es decir que la información solamente puede viajar en una dirección a la vez. Pero los pines 18 al 25, originalmente utilizados como tierras, pueden ser usados como pins de datos también. Esto permite la comunicación full-duplex (ambas direcciones al mismo tiempo).

DB 25		Centronics 36	
Pin	Signal	Pin	Signal
1	Strobe	1	Strobe
2	data0	2	data0
3	data1	3	data1
4	data2	4	data2
5	data3	5	data3
6	data4	6	data4
7	data5	7	data5
8	data6	8	data6
9	data7	9	data7
10	Acknowledge	10	Acknowledge
11	Busy	11	Busy
12	Paper End	12	Paper End
13	Select	13	Select
14	Auto Feed	14	Auto Feed
15	Error	15	Error
16	Init	16	Init
17	Select In	17	Select In
18	GND	18	GND
19	GND	19	GND
20	GND	20	GND
21	GND	21	GND
22	GND	22	GND
23	GND	23	GND
24	GND	24	GND
25	GND	25	GND
©2000 How Stuff Works		26	GND
		27	GND
		28	GND
		29	GND
		30	GND
		31	Init
		32	Error
		33	Ground
		34	NC
		35	NC
		36	Select In

Tabla. 3.3 Terminales puerto paralelo.

El Puerto Paralelo Mejorado (EPP de Enhanced Parallel Port) fue creado por Intel, Xircom y Zenith en 1991. El EPP permite transmitir mas información cada segundo (500 kilobytes a 2 megabytes). Este fue diseñado para dispositivos que no son impresoras, que se conectarían a este puerto, particularmente dispositivos de almacenamiento, los cuales necesitan la más alta velocidad de transferencia.

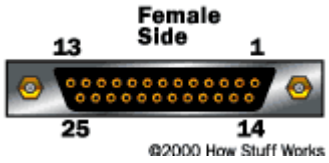


Fig. 3.6 Conector DB25.

Casi al mismo tiempo de la introducción del EPP, Microsoft y Hewlett Packard conjuntamente anunciaron una especificación llamada Salida Paralela con capacidad de expansión (ECP de Extended Capabilities Port) en 1992. Mientras el EPP estaba siendo usado para otros dispositivos, el ECP fue diseñado para mejorar la velocidad y funcionalidad de las impresoras.

3.12.3 Tabla de puertos paralelo.

El puerto paralelo se identifica por su dirección de I/O base y se identifica ante sistemas DOS por el número LPT. Cuando arranca la máquina, la BIOS chequea direcciones específicas de I/O en busca de puertos paralelos y construye una tabla de las direcciones halladas en la posición de memoria 0408h.

Esta tabla contiene hasta tres palabras de 16 bits. Cada palabra es la dirección de I/O base del puerto paralelo. La primera palabra corresponde a LPT1, la segunda a LPT2 y la tercera a LPT3. Hay que agregar que en DOS tenemos el dispositivo PRN que es un alias a uno de los dispositivos LPT (generalmente es LPT1, pero se puede cambiar con la orden MODE).

Las direcciones estándar para los puertos paralelos son 03BCh, 0378h y 0278h.

3.13 Comunicación USB.

El puerto USB (Universal Serial Bus), fue creado para buscar una respuesta a los límites de conectividad de los ordenadores, así como al límite de velocidad que tienen los puertos RS-232 y los puertos paralelos LPT.

Es una interfase conecta y usa (plug&play) entre la computadora y ciertos dispositivos tales como teclados, scanner, impresoras, módems, tarjetas de sonido, cámaras, etc.

El puerto USB tiene entre sus ventajas, además de una mayor velocidad de transmisión, el que a través del mismo puerto se pueden alimentar periféricos de bajo consumo.

El tipo de conector estándar en la computadora es el denominado tipo A con 4 contactos, dos para datos y dos para alimentación, pero en la conexión al periférico no hay ningún estándar, habiendo multitud de tipos diferentes de conectores, si bien el más utilizado es el tipo B. También son muy utilizados los tipos Mini USB y Micro USB, este último sobre todo en teléfonos móviles.

En cuanto a las capacidades y tipos, tenemos varios tipos diferentes de puertos USB:

USB 1.1: ya prácticamente en desuso, que presentaba dos velocidades de transmisión diferentes, 1.5Mb/s para teclados, ratones y otros dispositivos que no necesitan mayores velocidades, y una velocidad máxima de 12Mb/s.

USB 2.0: apareció ante la necesidad de una mayor velocidad de transmisión, llegando esta hasta los 480Mb/s teóricos que es en la práctica muy difícil alcanzar esa velocidad.

El sistema de bus serie universal USB consta de tres componentes:

- Controlador
- Hubs o Concentradores
- Periféricos

Fig. 3.7 Símbolo USB.

Dado que la comunicación USB, funciona de forma serial, primero explicaremos la forma de realizarla con el PIC18F2550 que tiene un puerto Serial que es configurado y controlado desde su UART.

En este programa se envía y recibe a 9600 baudios con un cristal de cuarzo de 4 Mhz como lo indica la hoja de especificaciones del PIC18F2550.

3.14 Circuito de recepción de datos en forma paralela y envío serial a la computadora.

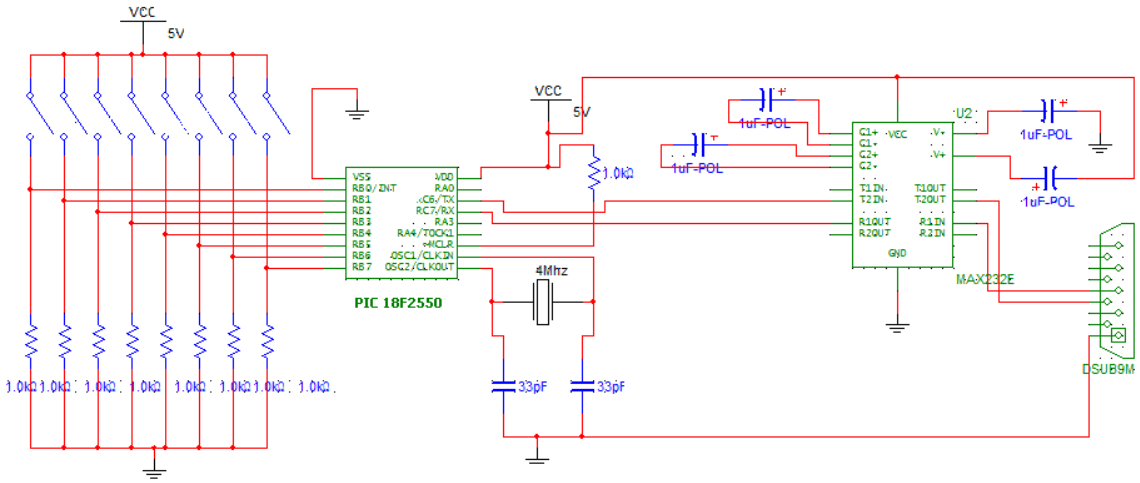


Fig. 3.8 Circuito recepción paralela y envío serial.

1. El conjunto de switchs nos permite enviar un numero binario de hasta 8 bits
2. El PIC se encarga de recibir el dato y colocarlo en una variable, la cual es enviada hacia el Max232
3. El Max232 envía los datos hacia el puerto serie de la PC, por medio de un conector DB9
4. Conexión del MAX232:

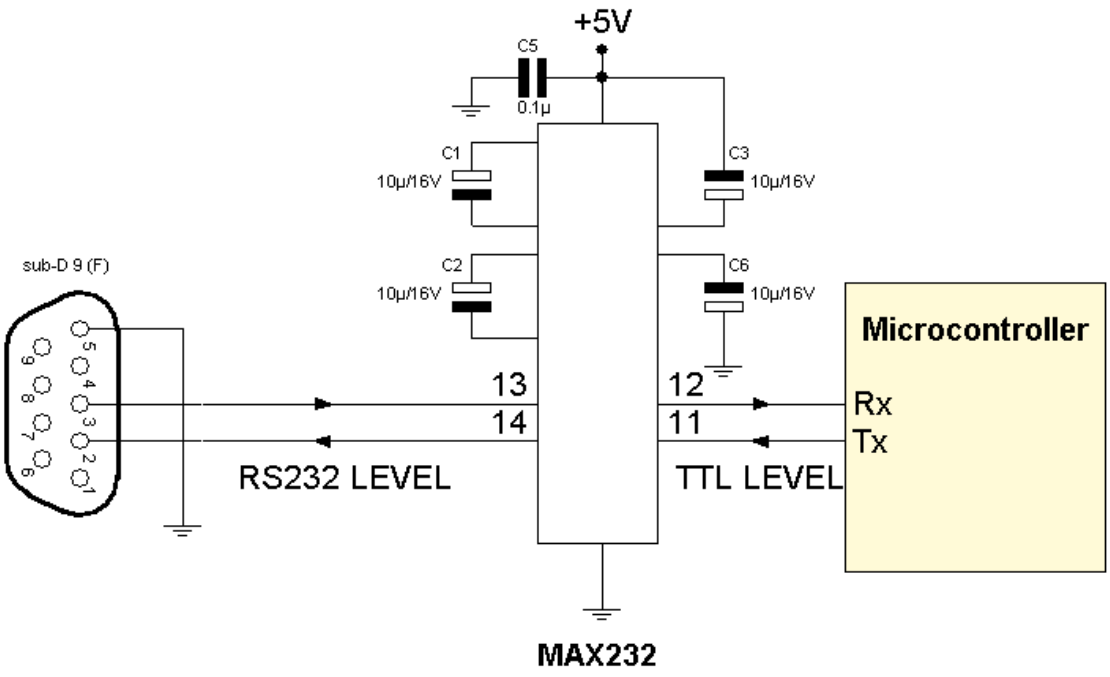


Fig. 3.9 Diagrama de conexión interfaz RS232.

3.14.1 Código.

```
//COMUNICACIÓN  A TRAVES DEL PUERTO SERIAL
//ENVIO DE DATOS DEL PUERTOB
#include<p18cxxx.h>
#include<usart.h>
#include<stdio.h>
#pragma config WDT=OFF
#pragma config OSC=XT
#pragma config LVP=OFF
void main (void){
OpenUSART  (USART_RX_INT_OFF &
          USART_TX_INT_OFF &
          USART_ASYNCH_MODE &
          USART_EIGHT_BIT &
          USART_CONT_RX &
          USART_BRGH_HIGH,25);
while(1){
TRISB=0xFF;

printf ("\n\r%d",PORTB);

   }
}
```

3.14.2 Explicación.

En este programa se establece que cada determinado tiempo el PIC envía un dato de forma serial a 9600 baudios sin paridad y con solo un bit de paro, hacia la PC, este dato es de 8 bits y se envía a partir del dip-switch el valor que se encuentre ahí es el que será enviado hacia la PC.

3.15 Circuito receptor de PC a LEDs en paralelo.

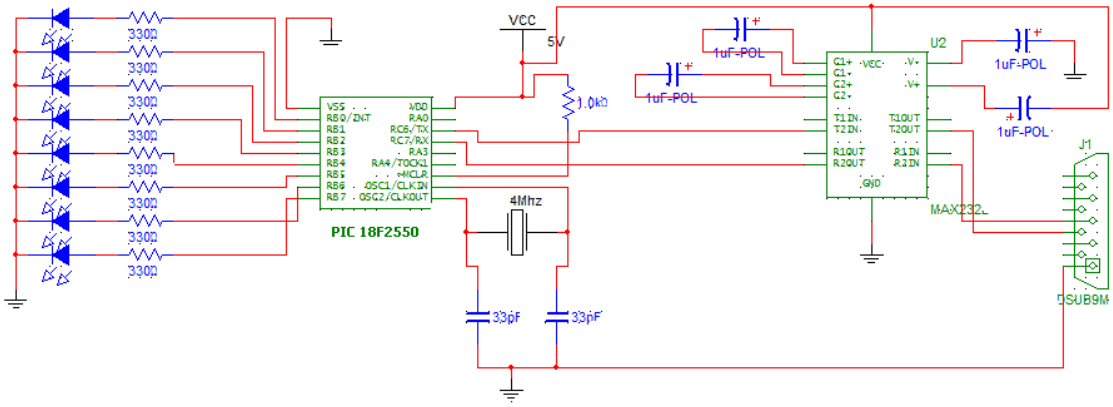


Fig. 3.10 Circuito recepción serial.

1) La computadora envía datos a través de su puerto serial estos datos llegan al conector DB9 de nuestro circuito
2) El dato que llega es convertido para que el PIC lo reciba
3) El PIC recibe el dato, lo coloca en una variable y después lo saca por su puerto B, donde están conectados LED´s para permitir observar el valor del dato.

3.15.1 Código.

```
//COMUNICACION A TRAVES DEL PUERTO SERIAL
//RECEPCIÓN DE DATOS DE LA USART Y DESPLEGAR EN EL PUERTOB
#include<p18cxxx.h>
#include<usart.h>
#include<stdio.h>
#pragma config WDT=OFF
#pragma config OSC=XT
#pragma config LVP=OFF
void main (void){
OpenUSART  (USART_RX_INT_OFF &
        USART_TX_INT_OFF &
        USART_ASYNCH_MODE &
        USART_EIGHT_BIT &
        USART_CONT_RX &
        USART_BRGH_HIGH,25);
while(1){
TRISB=0x00;

PORTB=ReadUSART ();

  }
}
```

3.15.2 Explicación.

Cada vez que llega un dato al registro txreg del PIC de forma serial, este es colocado en una variable y esta es mostrada en el PIC por el puerto B en forma binaria, el dato es de 8 bits a 9600 baudios, sin bit de paridad, y con un bit de parada.

3.16 Circuito de envío y recepción serial.

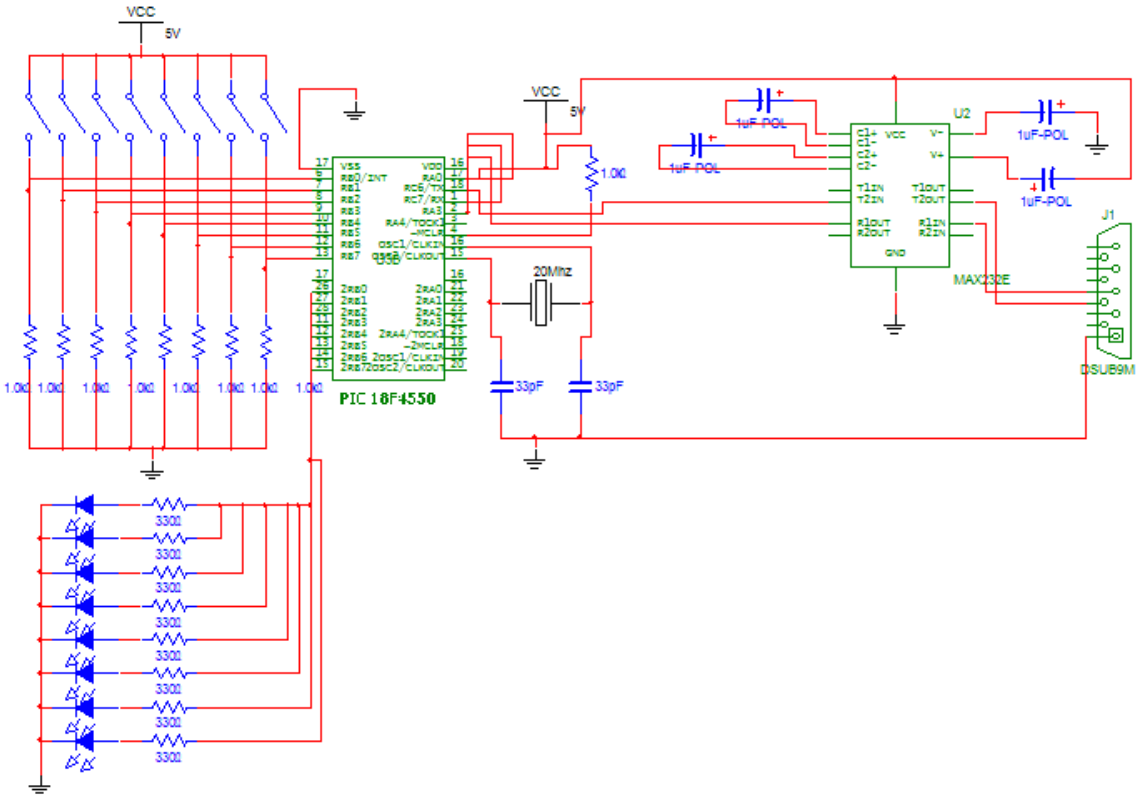


Fig. 3.11 Circuito envío/recepción serial.

3.16.1 Código.

```
//COMUNICACIÓN A TRAVES DEL PUERTO SERIAL
//ENVIO DE DATOS DEL PUERTOB Y RECEPCIÓN DE DATOS POR EL //PUERTO C
#include<p18cxxx.h>
#include<usart.h>
#include<stdio.h>
#pragma config WDT=OFF
#pragma config OSC=XT
#pragma config LVP=OFF
void main (void){
OpenUSART  (USART_RX_INT_OFF &
        USART_TX_INT_OFF &
```

```
        USART_ASYNCH_MODE &
        USART_EIGHT_BIT &
        USART_CONT_RX &
        USART_BRGH_HIGH,25);
while(1){
TRISB=0xFF;
TRISD=0X00;

printf ("\n\r%d",PORTB);
PORTD=READUSART ();

  }
}
```

CAPÍTULO IV.

4.1 SOFTWARE.

4.2 Desarrollo de aplicaciones para la PC.

4.2.1 Tratamiento de los datos.

En todas las aplicaciones es necesaria una interfaz amigable con el usuario a través de gráficos que muestren el comportamiento de los diferentes dispositivos que estén interactuando con el sistema.

Los gráficos para la aplicación fueron elaborados en Macromedia Flash 8, y se integraron para mostrar los datos que se recibían, o para mandarlos al exterior para lo cual se realizó lo siguiente:

1.- AxShockwaveFlash es el componente para manipular una animación flash, para poder utilizar el componente primero debemos agregarlo a la paleta de herramientas ToolBox, para ello damos clic derecho sobre la paleta ToolBox > Choose Items > COM Components > seleccionar Shochwave Flash Object > OK

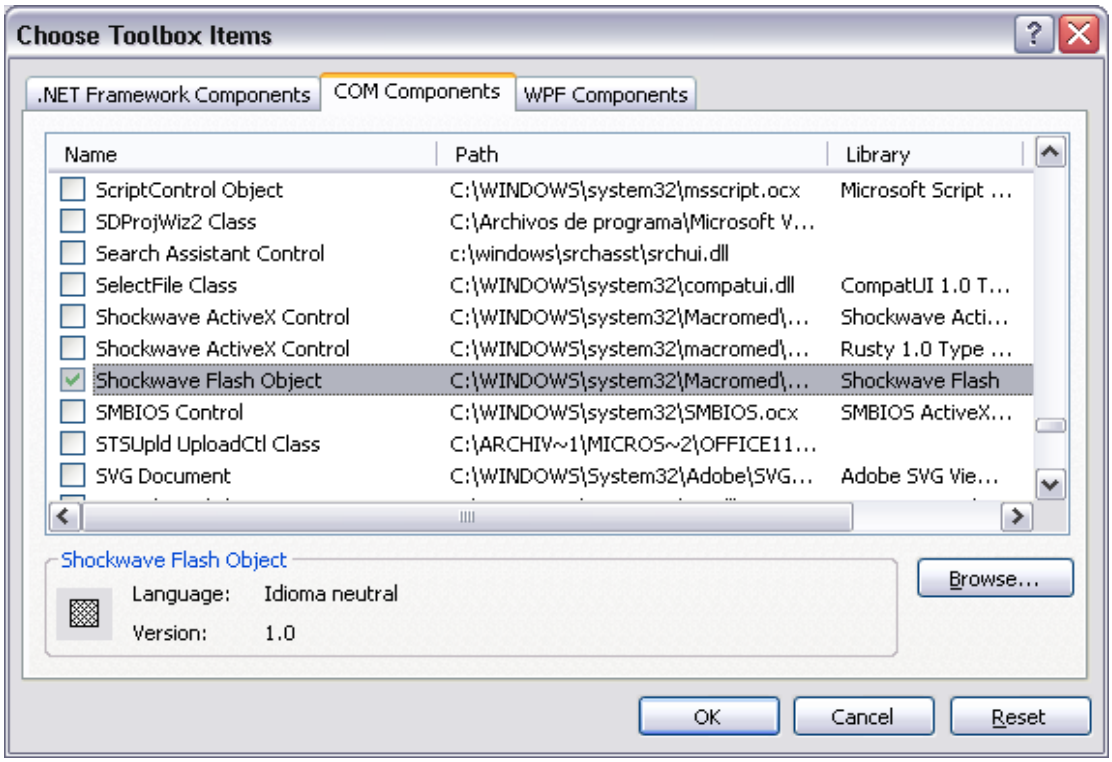


Fig. 4.1 Ventana de selección de componentes.

Con esto el componente queda agregado al ToolBox, de donde vamos a tenerlo disponible para agregarlo al formulario.

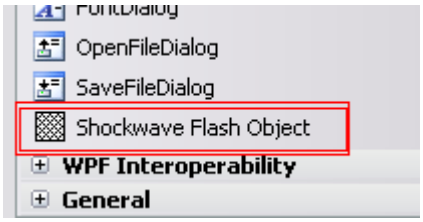


Fig. 4.2 Componente Shockwave Flash Object.

2.- Una vez agregado en el formulario el componente AxShockwaveFlash debemos indicar la ruta en donde se encuentra la animación en la propiedad Movie; recordar que la animación es un archivo de tipo *.swf.

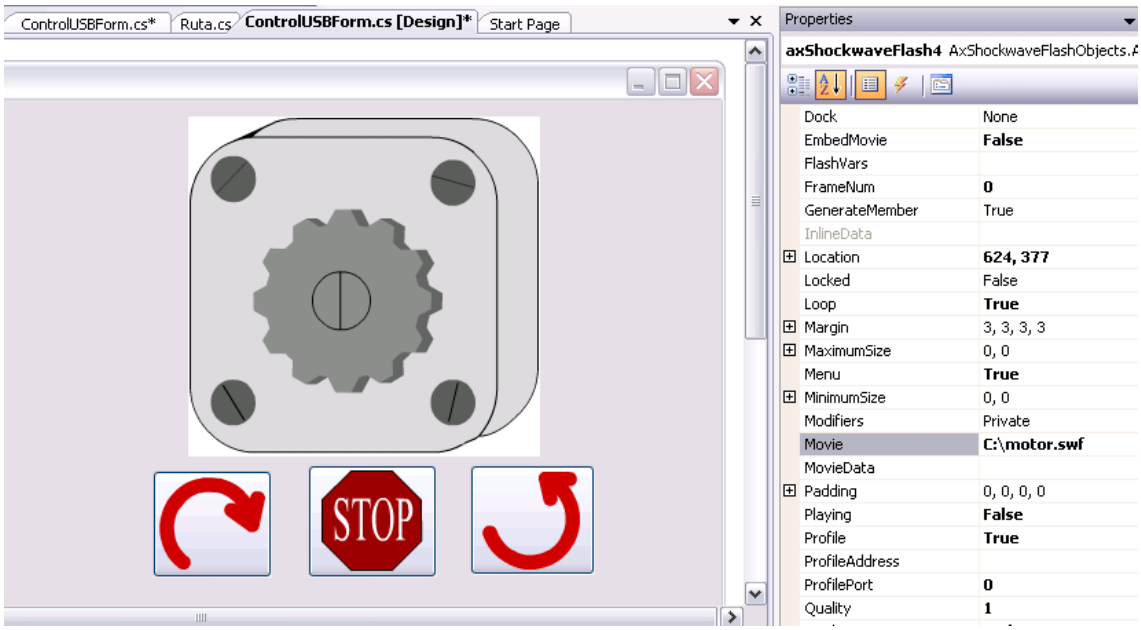


Fig. 4.3 Ventana de la aplicación.

Lo que resta es que la animación se active y envié el valor por el puerto, para ello necesitamos posicionarnos en el fotograma que corresponda al inicio de la animación y después se manda el dato, a continuación se muestra el código:

```
private void activaMotor(AxShockwaveFlashObjects.AxShockwaveFlash
axShockwaveFlash,byte codigoParaPic,int nFrame)
        {
            axShockwaveFlash.GotoFrame(nFrame);
```

```
            axShockwaveFlash.Play();

            bool bres;
            byte[] sdBuffer = new byte[2];

            sdBuffer[0] = 0x00;
            sdBuffer[1] = codigoParaPic;

            bres = picwinusbapi.Write_PicWinUSB(iHandle, sdBuffer);

        }
```

A continuación se enlistan las funciones y atributos que tiene la clase AxShockwaveFlash.

Métodos	• TCallLabel • TSetPropertyNum • TGetPropertyNum • Pause • Resume • LoadMovie • Play • StopPlay • Stop • IsPlaying • Back • Forward • Rewind • GotoFrame • CurrentFrame • FrameLoaded • FlashVersion • SetZoomRect • Zoom • Pan • PercentLoaded • TGotoFrame • TGotoLabel • TCurrentFrame • TCurrentLabel • TPlay • TStopPlay • SetVariable • GetVariable • TSetProperty • TGetProperty • TCallFrame • CreateSink • DetachSink • AttachInterfaces • RaiseOnOnReadyStateChange • RaiseOnOnProgress • RaiseOnFSCommand
Campos o atributos	• eventMulticaster • ReadyState • TotalFrames • playing • Quality • ScaleMode

	• AlignMode • BackgroundColor • Loop • Movie • frameNum • WMode • SAlign • Menu • Base • CtlScale • DeviceFont • EmbedMovie • BGColor • Quality2 • SWRemote • FlashVars • Sound • OnReadyStateChange • OnProgress • FSCommand

4.3 Issues (Problemas) surgidos durante la integración de Macromedia Flash 8 y C#.

Aunque se tenga instalada la última versión del IDE de Macromedia, el componente de flash no va a funcionar ya que necesita la última versión de flash, para lo cual debemos descargarlo de la página oficial de Macromedia e instalarla en la PC:
http://www.adobe.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash&Lang=Spanish

Luego debemos limpiar el proyecto el menú Build > Clean Solution > Rebuild Solution, ya que aunque tengamos la última versión, se queda el código anterior lo que no permite que se pueda compilar bien.

4.4 Aplicación de Software para el Puerto Serial.

4.4.1 Introducción.

La computadora cuenta con diversos puertos SD, USB, miniUSB, serial, paralelo, etc. Hoy en día es posible que una computadora, por ejemplo una portátil no disponga de todos y cada uno de ellos, y aunque físicamente no cuente con el conector en ella, lo implementa virtualmente. Este es el caso del puerto serial, el cual, no necesariamente debe tener un conector DB9 viviendo en nuestra máquina para ser utilizado.

Esto es muy importante ya que ésta es la base para realizar la comunicación Bluetooth e I^2C, y cualquier otro dispositivo que funcione "serialmente".
El puerto serial virtual es lo que se denomina COM, y pueden existir en la computadora, estando o no el(los) conector(es) DB9 en ella.
Nosotros a nivel lógico nos vamos a comunicar con un puerto COM, y lo trataremos como un puerto serial, ya que eso es lo que representa, un puerto serial más.

Para familiarizarnos un poco más con los COM's que tiene nuestra computadora, podemos ver el administrador de dispositivos de Hardware, por ejemplo en el sistema operativo Windows(R) XP se listan como indica la siguiente figura:

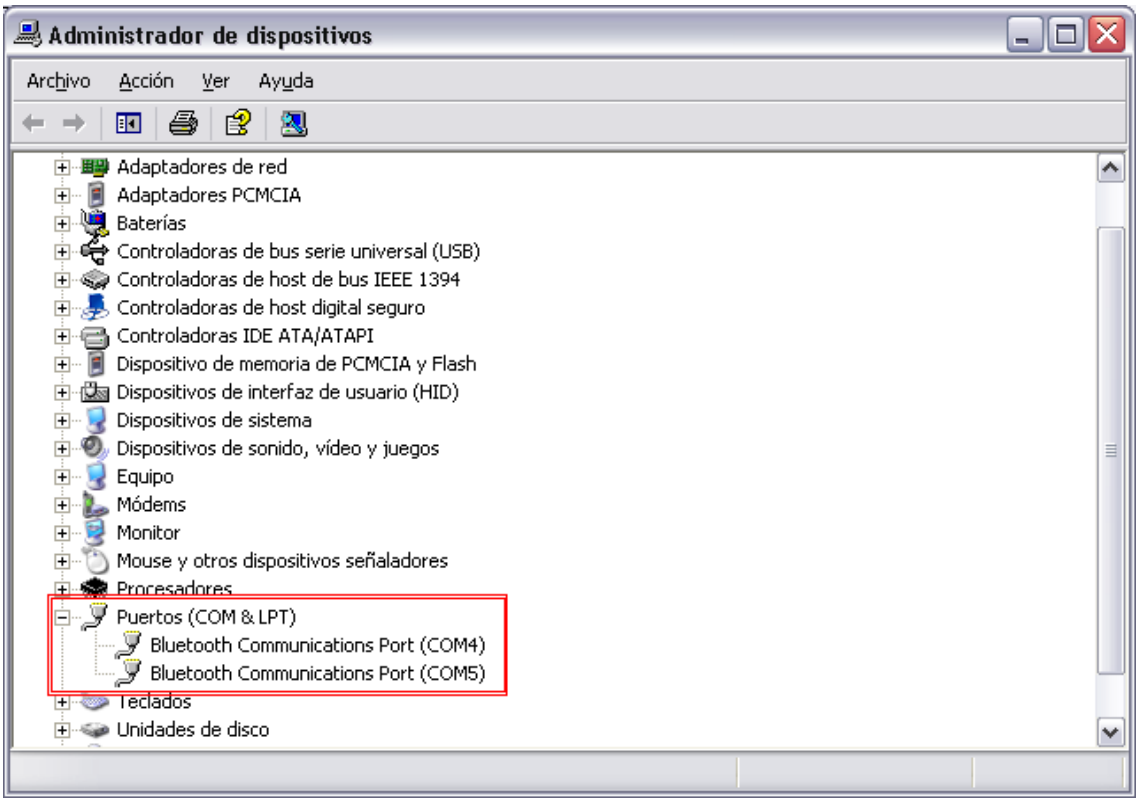


Fig. 4.4 Ventana administrador de dispositivos.

Se puede observar en la figura que existen el COM4 y el COM5, aunque el número de COM va a depender del dispositivo que lo dé de alta en el sistema operativo.

4.4.2 Implementación del Puerto Serial.

Como se menciono anteriormente, no es necesario que contemos con el conector en nuestra computadora, nosotros vamos a trabajar con los COM's y los veremos como un puerto serial más.

C# nos proporciona un componente llamado SerialPort, el cual se encuentra en el ToolBox, y que puede ser integrado a la aplicación que estamos desarrollando.

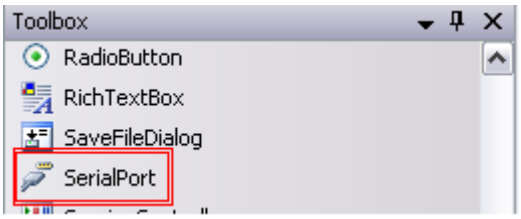
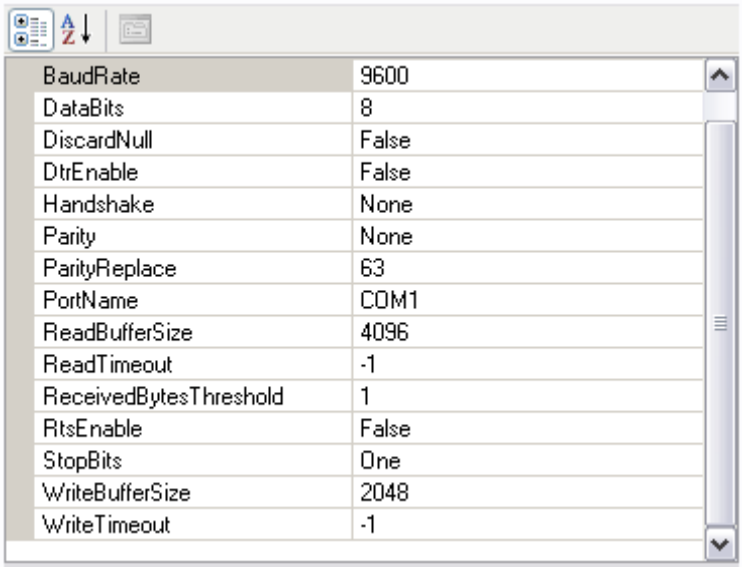


Fig. 4.5 Componente Serial Port.

El objeto SerialPort tiene diversas funciones y atributos que realizan la lectura y escritura en el puerto, las cuales fueron las que utilizamos para lograr enviar y recibir datos serialmente. Este objeto tiene 3 tipos de eventos DataReceived, ErrorReceived y PinChanged, que son utilizados como lo haríamos con los eventos de un botón, programando la funcionalidad que queremos implementar al recibir un dato, dentro de la función que se dispara cuando el evento DataReceived es activado, esto es cuando al puerto le llega un dato del exterior.

Todos los pasos necesarios para la comunicación ya están implementados dentro de este objeto, nosotros solamente utilizamos las funciones; es decir por ejemplo para mandar un dato por el puerto serie, solamente mandamos a llamar la función `Write(String msn)` o `Write(byte[] data, int offset, int count),` y en cuanto al establecimiento del bit de paridad, los baudios, etc., vienen por default los siguientes valores:



BaudRate	9600
DataBits	8
DiscardNull	False
DtrEnable	False
Handshake	None
Parity	None
ParityReplace	63
PortName	COM1
ReadBufferSize	4096
ReadTimeout	-1
ReceivedBytesThreshold	1
RtsEnable	False
StopBits	One
WriteBufferSize	2048
WriteTimeout	-1

Fig. 4.6 Configuración puerto serial.

Si nuestro dispositivo a comunicar tiene una configuración diferente a la mostrada, puede establecerse una diferente sin ningún problema mediante las propiedades del objeto.

Contando ya con un bosquejo general vamos a definir los pasos necesarios para utilización de este objeto:

1.- Una vez que tenemos nuestro objeto agregado a nuestro formulario, agregamos el código necesario para la configuración del puerto de acuerdo al dispositivo que vayamos a conectar en él, cabe señalar que de no ser así, podemos configurarlo en tiempo de ejecución, y abrir el puerto hasta ese momento.

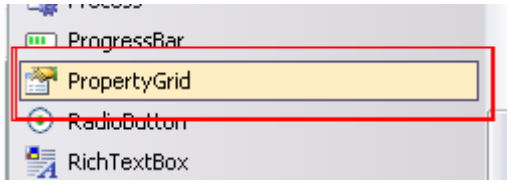


Fig. 4.7 Componente Property Grid.

Una opción para que la configuración sea mas amigable al usuario y mas rápida de implementar es utilizando un componente llamado PropertyGrid, el cual necesita ser inicializado con la propiedades del objeto serial, esto se tiene que hacer en el evento del formulario Load, para que aparezca una tabla con cada uno de los atributos configurables en el puerto, ahí es donde el Usuario puede establecer los nuevos parámetros para que el dispositivo que se conecte funcione correctamente. Con lo que el código en el evento Load queda:

```
private void MotorSerieForm_Load(object sender, EventArgs e)
{
    this.propertyGrid1.SelectedObject = this.serialPort1;
}
```

Al correr la aplicación aparecerá una tabla como la que se muestra en la figura 4.8.

2.- El puerto se abre con la función Open() y se cierra con la función Close(), y estas pueden ser ejecutadas en cuanto se abra la ventana de la aplicación según se necesiten o cuando el usuario active algún evento, como por ejemplo después de haberlo configurado. La siguiente figura muestra el componente PropertyGrid con todas las propiedades del objeto SerialPort en una aplicación ejecutándose.

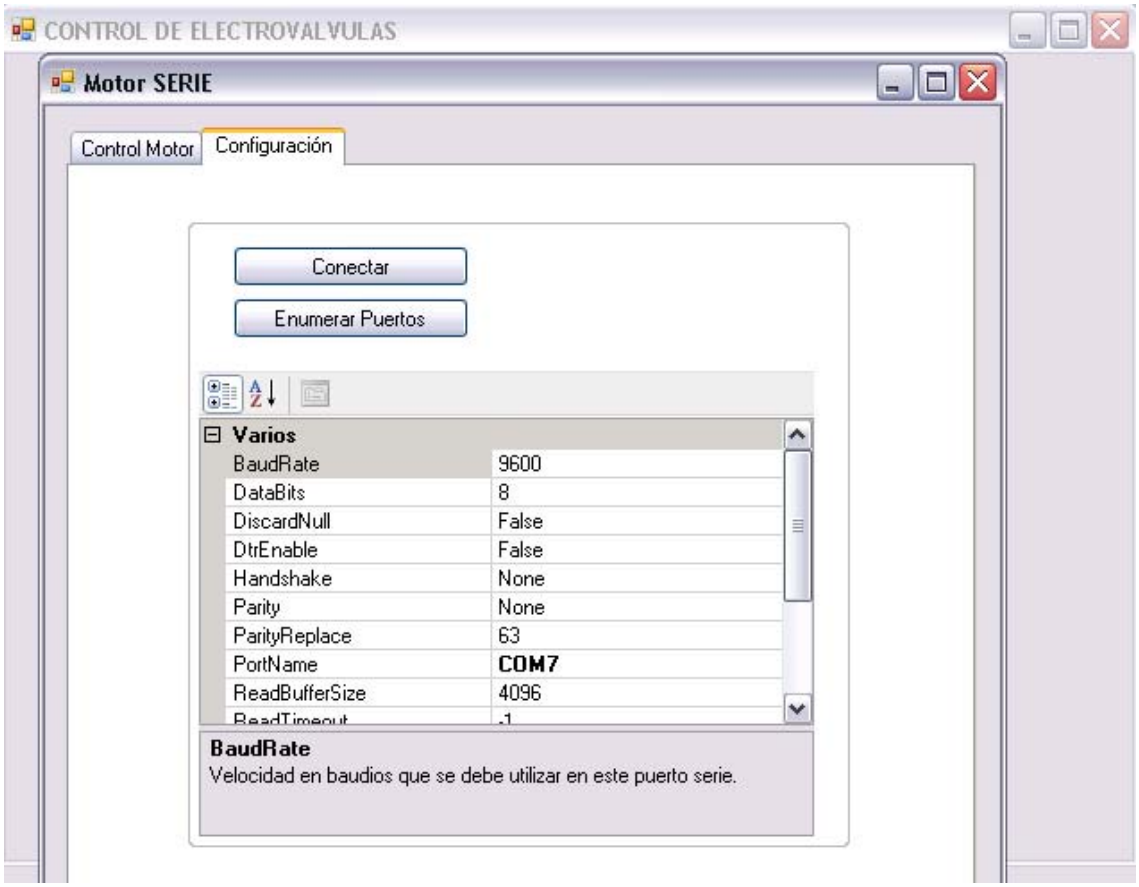


Fig. 4.8 Ventana de aplicación serial.

El usuario puede entonces configurar el puerto serial de acuerdo a sus necesidades, y cualquier cambio que haga y que lo acepte el objeto SerialPort será establecido mediante esta tabla sin ningún código adicional. Para abrir el puerto basta con dar clic en el botón Conectar para obtener esta funcionalidad se tiene el siguiente código:

```
private void button1Conectar_Click(object sender, EventArgs e)
  {
      if(this.serialPort1!=null && !this.serialPort1.IsOpen){
          try
          {
              this.serialPort1.Open();
          }
          catch (Exception exc)
          {
              MessageBox.Show("Error en el manejo del puerto " +
               exc.ToString());
              return;
          }
      }
      MessageBox.Show("Puerto Serie abierto correctamente");
  }
```

Si se requiere que al momento de cargar la aplicación el puerto ya haya sido abierto, se necesita de antemano conocer el número de COM que vamos a utilizar, e inicializar y abrir el puerto en el evento Load del formulario, como se muestra a continuación:

```
private void Form1_Load(object sender, EventArgs e)
{
    try
    {
        this.serialPort1.PortName = "COM7";
        this.propertyGrid1.SelectedObject = this.serialPort1;
        this.serialPort1.Open();
    }catch(IOException){
        MessageBox.Show("No se pudo abrir el puerto,\nconectar
          el dispositivo o cambiar su configuración");
    }
}
```

3.- Una vez abierto el puerto es posible recibir y enviar datos mediante él, con las funciones de lectura y escritura proporcionadas por el Objeto, las utilizadas fueron las siguientes:

Nombre	Descripción
Write(String)	Escribe la cadena especificada en el puerto serie.
Write(Byte[ ], Int32, Int32)	Escribe un número especificado de bytes en el puerto serie utilizando los datos de un búfer.
Write(Char [ ], Int32, Int32)	Escribe un número especificado de caracteres en el puerto serie utilizando los datos de un búfer.
ReadExisting ()	Lee todos los bytes inmediatamente disponibles, basándose en la codificación, en la secuencia y el búfer de entrada del objeto SerialPort.

Tabla. 4.1 Funciones puerto serie.

Para enviar datos por el puerto se tiene el siguiente código ejemplo:

```
private void enviar(String dato)
{
      try{
            if (this.serialPort1 != null && !this.serialPort1.IsOpen)
            {
               this.serialPort1.Open();
            }
            this.serialPort1.Write(dato);
      }catch (IOException){
            MessageBox.Show("Error al tratar de abrir el puerto,
            verificar configuración en la pestaña \"Configuración\"");
      }
 }
```

Para recibir datos por el puerto se tiene el siguiente código:

```
private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e){
      tratamientoDeDato(this.serialPort1.ReadExisting());
}
```

En donde la función `tratamientoDeDato(String )` es donde se utiliza el dato para darle una aplicación.

4.5 Aplicación de Software para el Puerto USB.

Para comunicarnos con el puerto USB utilizamos el driver WinUSB que es un driver genérico desarrollado por el Windows Driver Foundation (WDF) y está disponible para Windows XP y las versiones posteriores de Windows.

WinUSB es una dll llamada winusb.dll, que cuenta con las funciones necesarias para comunicarnos con el puerto USB, esta dll, ya se encuentra en Windows Vista y es compatible con Windows XP, y para utilizarse solo tiene que pegarse en la carpeta de sistema system32, para ser utilizada en XP.

4.5.1 Implementación del Puerto USB.

Lo que se realizó es una clase llamada USBAPI, que implementa funciones en las cuales se mandan a llamar las funciones de winusb.dll y que sirven como interfaz entre la aplicación y winusb.dll.

El objetivo de la clase USBAPI es servir de puente y que el desarrollo de la aplicación no se centre saber como se utilizar las funciones de winusb.dll, si no en simplemente enviar, recibir y tratar los datos en la aplicación, a continuación se detalla la realización de dicha clase, aunque en el momento del desarrollo solo se utilizaran sus funciones y no será necesario modificarlas.

Para que la clase USBAPI utilizara las funciones de winusb.dll necesitamos que winusb.dll se encuentre en system32, y así usar las funciones que nos proporciona.

A la definición de la clase USBAPI debe agregarse la palabra reservada `unsafe` ya que necesitamos utilizar apuntadores y debemos avisarle al compilador que vamos a usarlos por lo que la definición de la clase queda como sigue:

```
unsafe public class USBAPI{}
```

Ahora necesitamos configurar el IDE para que pueda compilar código de tipo unsafe por lo que debemos realizar lo siguiente :
Proyect > {NombreDelProyecto}Properties > Build > palomear Allow unsafe code

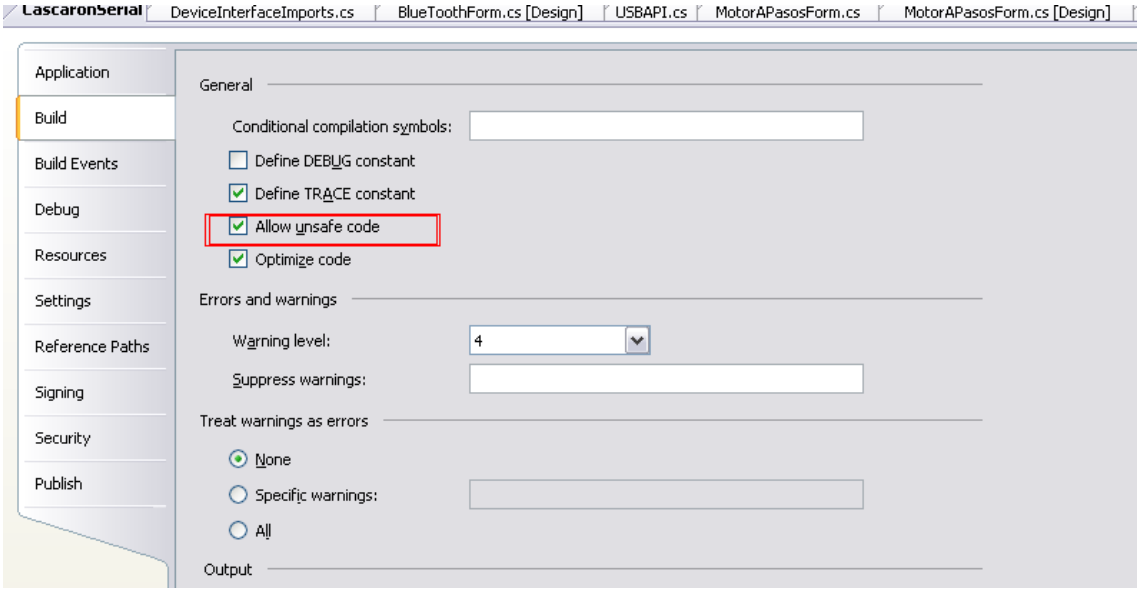


Fig. 4.9 Configuración del proyecto.

Para importar las funciones de winusb.dll, se tiene el siguiente código que va dentro de la definición de la clase:

```
#region  functiones Importadas de winusb.dll

[DllImport("winusb.dll")]
private static extern bool WinUsb_Initialize(IntPtr DeviceHandle, out
IntPtr InterfaceHandle); // out inter
        [DllImport("winusb.dll")]
        private static extern bool WinUsb_ReadPipe(IntPtr InterfaceHandle,
byte PipeID, byte[] Buffer, int BufferLength, out uint LengthTransferred,
IntPtr Overlapped);
        [DllImport("winusb.dll")]
        private static extern bool WinUsb_WritePipe(IntPtr InterfaceHandle,
byte PipeID, byte[] Buffer, int BufferLength, out uint LengthTransferred,
IntPtr Overlapped);
 #endregion
```

Ahora necesitamos hacer unas llamadas a funciones que se encuentran en setupapi.dll y kernel32.dll.

setupapi.dll es una librería de Windows que contiene funciones necesarias durante el proceso de instalación. Se trata de una librería de contenido inofensivo cuya presencia o carga en el sistema no supone riesgo alguno.

Kernerl32.dll es la biblioteca dinámica de datos (dynamic link library: DLL) que se encuentra en el núcleo del Sistema Operativo Windows. Se ocupa de gestionar la administración de memoria, operaciones de entrada/salida y las interrupciones.

Para ello realizamos una clase que se encargue de sólo importar las funciones de dichas librerías, llamada ImportadorDll se detalla a continuación:

```
using System;
using System.Runtime.InteropServices;
using System.IO;
using Microsoft.Win32.SafeHandles;

namespace CascaronSerial
{
    class ImportadorDll
    {
        public const int DIGCF_PRESENT = 0x00000002;
        public const int DIGCF_DEVICEINTERFACE = 0x00000010;

        public const int FILE_ATTRIBUTE_NORMAL = 0x00000080;
        public const int FILE_FLAG_OVERLAPPED = 0x40000000;

        [StructLayout(LayoutKind.Sequential)]
        public struct SP_DEVICE_INTERFACE_DATA
        {
            public int cbSize;
            public Guid InterfaceClassGuid;
            public int Flags;
            public IntPtr Reserved;
        }

        [StructLayout(LayoutKind.Sequential)]
        public struct SP_DEVICE_INTERFACE_DETAIL_DATA
        {
            public int cbSize;
            [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
            public string DevicePath;
        }

        [DllImport(@"setupapi.dll", CharSet = CharSet.Auto, SetLastError =
true)]
        public static extern IntPtr SetupDiGetClassDevs(
            ref Guid ClassGuid,
            [MarshalAs(UnmanagedType.LPTStr)] string Enumerator,
            IntPtr hwndParent,
            UInt32 Flags);

        [DllImport(@"setupapi.dll", CharSet = CharSet.Auto, SetLastError =
true)]
        public static extern Boolean SetupDiEnumDeviceInterfaces(
            IntPtr hDevInfo,
            IntPtr devInfo,
            ref Guid interfaceClassGuid,
            UInt32 memberIndex,
            ref SP_DEVICE_INTERFACE_DATA deviceInterfaceData);

        [DllImport(@"setupapi.dll", SetLastError = true)]
        public static extern Boolean SetupDiGetDeviceInterfaceDetail(
            IntPtr hDevInfo,
            ref SP_DEVICE_INTERFACE_DATA deviceInterfaceData,
```

```
            IntPtr deviceInterfaceDetailData,
            UInt32 deviceInterfaceDetailDataSize,
            out UInt32 requiredSize,
            IntPtr deviceInfoData);

        [DllImport(@"setupapi.dll", SetLastError = true)]
        public static extern Boolean SetupDiGetDeviceInterfaceDetail(
            IntPtr hDevInfo,
            ref SP_DEVICE_INTERFACE_DATA deviceInterfaceData,
            ref SP_DEVICE_INTERFACE_DETAIL_DATA deviceInterfaceDetailData,
            UInt32 deviceInterfaceDetailDataSize,
            out UInt32 requiredSize,
            IntPtr deviceInfoData);

        [DllImport(@"setupapi.dll", CharSet = CharSet.Auto, SetLastError =
true)]
        public static extern UInt16 SetupDiDestroyDeviceInfoList(
            IntPtr hDevInfo);

        [DllImport("kernel32.dll")]
        public static extern IntPtr CreateFile(
            string fileName,
            FileAccess fileAccess,
            FileShare fileShare,
            IntPtr securityAttributes,
            FileMode creationDisposition,
            UInt32 flags,
            IntPtr template);

        [DllImport("kernel32.dll")]
        public static extern bool CloseHandle(
            IntPtr hObject);
    }
}
```

Estas funciones son utilizadas en el método Iniciar_USB la clase USBAPI, y se utilizan para inicializar y utilizar el puerto USB correctamente.

```
public IntPtr Iniciar_USB(Guid InterfaceGuid)
        {
            bool bResult = false;
            UInt32 requiredLength = 0;
            UInt32 Instance = 0;  // Instancia > 0 si varios USB estan
adjuntos
            IntPtr iHandle;
            IntPtr hDev;

            // [1] Obtiene informacion para controlar pasandole el GUID que
se definio en la instalación del dispositivo
            IntPtr deviceInfo = ImportadorDll.SetupDiGetClassDevs(ref
InterfaceGuid, null, IntPtr.Zero, (ImportadorDll.DIGCF_PRESENT |
ImportadorDll.DIGCF_DEVICEINTERFACE));

            // [2] Llamar SetupDiEnumDeviceInterfaces para enumarar la
interface del sistema y obtener informacion en la interfas del dispositivo
            ImportadorDll.SP_DEVICE_INTERFACE_DATA interfaceData = new
ImportadorDll.SP_DEVICE_INTERFACE_DATA();
```

```
            interfaceData.cbSize = Marshal.SizeOf(interfaceData);
            bResult = ImportadorDll.SetupDiEnumDeviceInterfaces(deviceInfo,
IntPtr.Zero, ref InterfaceGuid, Instance, ref interfaceData);
            // [3] Llamar SetupDiGetDeviceInterfaceDetail para obtener
informacio detallada de la interfaz del dispositivo
            ImportadorDll.SetupDiGetDeviceInterfaceDetail(deviceInfo, ref
interfaceData, IntPtr.Zero, 0, out requiredLength, IntPtr.Zero);
            ImportadorDll.SP_DEVICE_INTERFACE_DETAIL_DATA detailData = new
ImportadorDll.SP_DEVICE_INTERFACE_DETAIL_DATA();

            if (IntPtr.Size == 8) detailData.cbSize = 8;// x64 OS
            else detailData.cbSize = 5; // x86 OS

            bResult =
ImportadorDll.SetupDiGetDeviceInterfaceDetail(deviceInfo, ref
interfaceData, ref detailData, requiredLength, out requiredLength,
IntPtr.Zero);

            // [4] Destruye la informacion que ya no se utiliza
            ImportadorDll.SetupDiDestroyDeviceInfoList(deviceInfo);

            // [5] Pasa la ruta del dispositivo del archivo que tiene el
dispositivo
            hDev = ImportadorDll.CreateFile(detailData.DevicePath,
FileAccess.ReadWrite, FileShare.ReadWrite, IntPtr.Zero, FileMode.Open,
(ImportadorDll.FILE_ATTRIBUTE_NORMAL | ImportadorDll.FILE_FLAG_OVERLAPPED),
IntPtr.Zero);
            // [6] Llama la funcion WinUsb_Initialize para obtener el
manejo del dispositivo
            bResult = WinUsb_Initialize(hDev, out iHandle);
            return iHandle;
}
```

La función para escribir en el puerto es llamada Write_USB y se detalla a continuación:

```
public bool Write_USB(IntPtr iHandle, byte[] dBuffer){
            byte PipeID = 0x01;     // send pipe
            int szBuffer = dBuffer.GetLength(0);
            uint bytesWritten;
            bool bResult;
            bResult = WinUsb_WritePipe(iHandle, PipeID, dBuffer, szBuffer,
out bytesWritten, IntPtr.Zero);
            return bResult;
}
```

Con esta clase ya podemos implementar la comunicación USB como base de cualquier aplicación.

CAPÍTULO V.

5 .1 DESARROLLO DE LA TARJETA DE CONTROL.

Una de las partes mas importantes de la tarjeta es la correcta polarización de los transistores TIP142, estos se encargan de activar las solenoides de las electroválvulas, las cuales se alimentan a 24 VCD y consumen 3 Amperes de corriente, es indispensable que los transistores conmuten correctamente entre su estado de corte y saturación.

Si no se alcanzara la zona de corte adecuadamente la válvula quedaría activada siempre y el motor seguiría girando, por el contrario si no se asegura estar en la zona de saturación la válvula no se activaría y no se reflejaría ningún movimiento en el motor.

Para tener la correcta polarización se utilizan los transistores secundarios BC547 y el BC557, a continuación se muestran los cálculos correspondientes.

Para $BC547$

$hfe = 125, I_C = 10mA, V_{CE} = 0.25V$

$$R_B = \frac{(V_{CC} - V_{BE})\beta}{I_C}$$

$$R_B = \frac{(V_{CC} - V_{BE})\beta}{I_C} = \frac{(24V - 0.7V)125}{10mA} = 291.1K\Omega \approx 270K\Omega$$

$$R_C = \frac{V_{CC} - V_{CE}}{I_C}$$

$$R_C = \frac{V_{CC} - V_{CE}}{I_C} = \frac{24V - 0.25V}{10mA} = 2.37k\Omega \approx 2.7k\Omega$$

Para $BC557$

$hfe = 300, I_C = 10mA, V_{BE} = 0.75V$

$$R_B = \frac{(V_{CC} - V_{BE})\beta}{I_C}$$

$$R_{B557} = \frac{(V_{CC} - V_{BE})\beta}{I_C} = \frac{(24V - 0.75V)300}{10mA} = 697.5k\Omega \approx 680k\Omega$$

Para la realización del circuito impreso (PCB) utilizamos el software de diseño Altium 2004 ya que cuenta con muy buenas herramientas de auto-enrutado y es muy usado en la industria, a continuación se muestra el diseño del esquemático.

5.2 Esquemático.

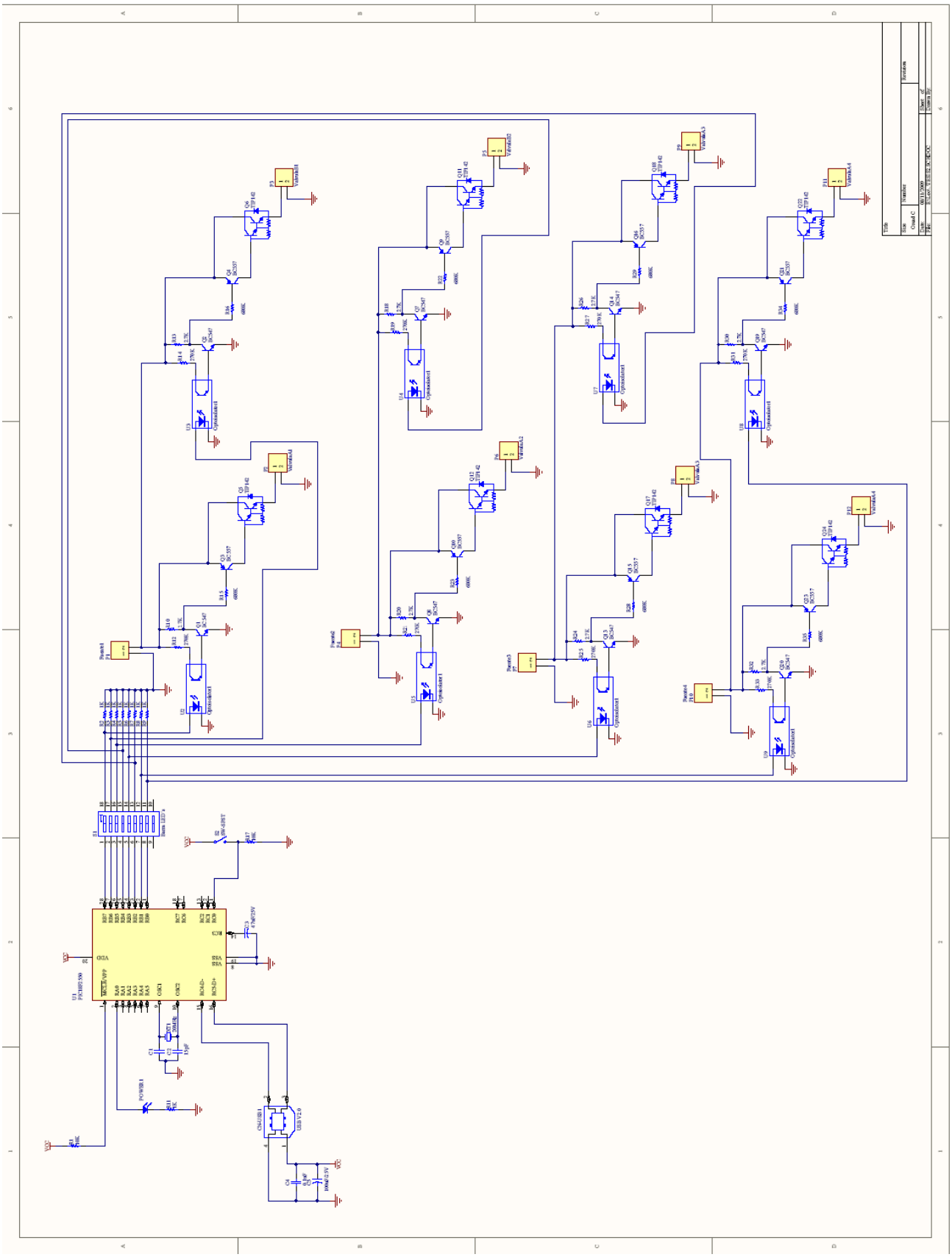


Fig. 5.1 Esquemático.

5.3 Imagen del PCB.

Con el diseño del esquemático terminado se genera el PCB.

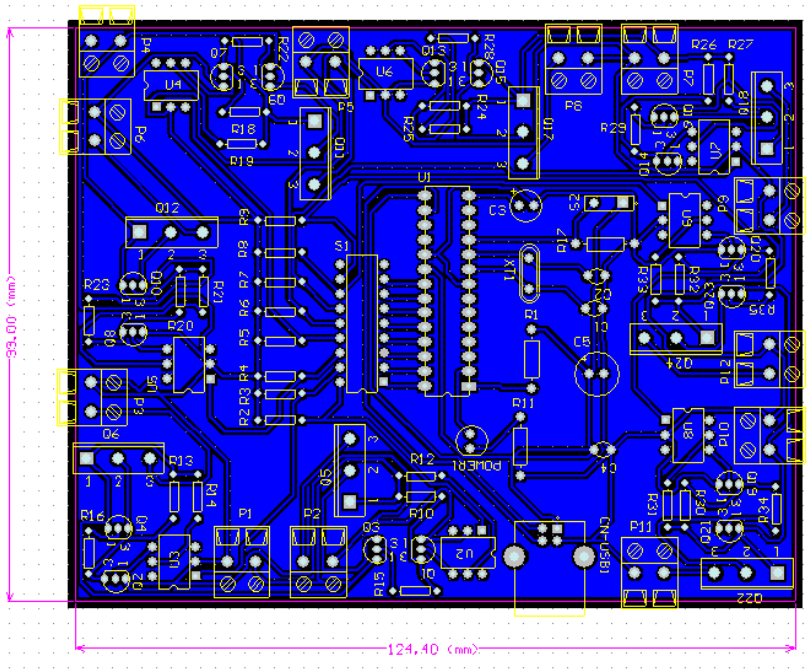


Fig. 5.2 Imagen del PCB.

Para facilitar el acomodo de los componentes ALTIUM cuenta con un generador de una imagen en 3D de la placa para observarla antes de realizarla.

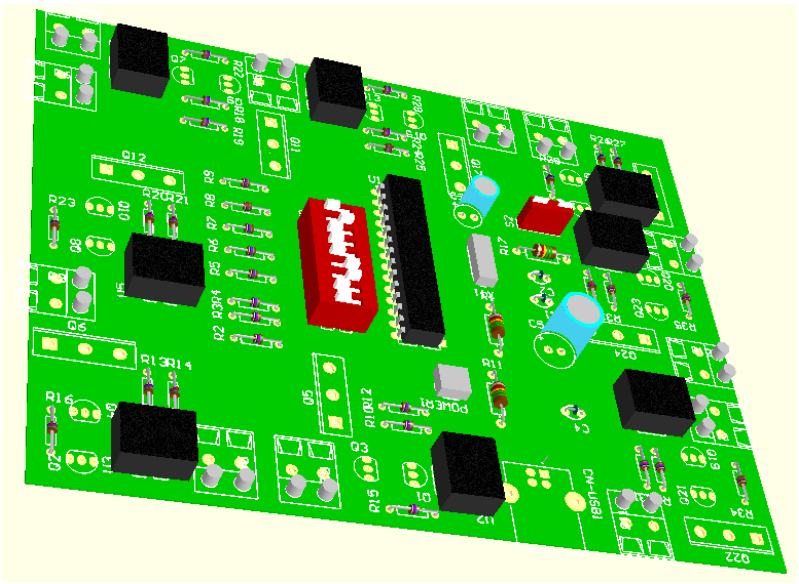


Fig. 5.3 Vista en 3D.

5.4 Firmware.

Para la realización del programa final que contiene el PIC 18F2550 se utilizo el “PIC C Compiler”, puesto que tiene una interfaz mas amigable que la del MPLAB de Microchip y no es necesario una configuración tan compleja, también porque cuenta con las librerías .h para el manejo del puerto USB, las cuales son mas sencillas que las diseñadas por Microchip.

5.5 Código fuente.

El código fuente contiene los métodos necesarios para la comunicación USB y la lógica del control de las válvulas.

```
#include <18F2550.h>
#fuses
HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN
#use delay(clock=48000000)

#define USB_HID_DEVICE     FALSE           // deshabilitar HID
#define USB_EP1_TX_ENABLE  USB_ENABLE_BULK   // habilitar TX
#define USB_EP1_RX_ENABLE  USB_ENABLE_BULK   // habilitar RX
#define USB_EP1_TX_SIZE    2               // 2 buffers de transmisión
#define USB_EP1_RX_SIZE    2               // 2 buffers de recepción

#include <pic18_usb.h>     // Seleccionar PIC de la familia 18FXX
#include <PicWinUSB.h>     // Configuración USB
#include <usb.c>

#define LED1      PIN_B0
#define LED2      PIN_B1
#define LED3      PIN_B2
#define LED4      PIN_B3
#define LED5      PIN_B4
#define LED6      PIN_B5
#define LED7      PIN_B6
#define LED8      PIN_B7
#define LED9      PIN_A0
#define ENTRADA1  PIN_C0
#define LED_OFF   output_low
#define LED_ON    output_high

void main(void) {

  int8 iBuff[2];                //buffers
  int8 oBuff[2];
  int8 entrada;
```

```
output_b(0x00); // Apagar LED

usb_init();

usb_task();
usb_wait_for_enumeration();

while (TRUE)
{
entrada=input_c();
if (entrada == 0)
{
  LED_OFF(LED9);
  output_b(0x00);
}
else
{
  LED_ON(LED9);
}
  if(usb_enumerated())          // ¿ Se le a asignado un número a PicWinUSB  ?
  {
    if (usb_kbhit(1))           // ¿EP tiene algún dato?
    {

      usb_get_packet(1, iBuff, 2); // Lee un paquete de 2 bytes del EP 1

       if (iBuff[0] == 0) // Modo Led
       {
         if (iBuff[1] == 1)
         {
           LED_OFF(LED2);//apaga LED2
           LED_ON(LED1);// enciende  LED1
         }
         if (iBuff[1] == 2)
         {
           LED_OFF(LED1);//apaga LED1
           LED_ON(LED2);// enciende  LED2
         }
         if (iBuff[1] == 3)
         {
           LED_OFF(LED4);//apaga LED4
           LED_ON(LED3); // enciende  LED3
         }
         if (iBuff[1] == 4)
         {
           LED_OFF(LED3);//apaga LED3
```

```
    LED_ON(LED4); // enciende  LED4
  }
  if (iBuff[1] == 5)
  {
    LED_OFF(LED6);//apaga LED6
    LED_ON(LED5);//enciende LED5
  }
  if (iBuff[1] == 6)
  {
    LED_OFF(LED5); // apaga  LED5
    LED_ON(LED6);//enciende LED6
  }
  if (iBuff[1] == 7)
  {
    LED_OFF(LED8);//apaga LED8
    LED_ON(LED7); // enciende  LED7
  }
  if (iBuff[1] == 8)
  {
    LED_OFF(LED7);//apaga LED7
    LED_ON(LED8); // enciende  LED8
  }


}


if (iBuff[0] == 1)//Modo apagar
{
  if (iBuff[1] == 0)
  {
    LED_OFF(LED1); // Apaga 1
    LED_OFF(LED2); // Apaga 2
  }
  if (iBuff[1] == 1)
  {
   LED_OFF(LED3); // Apaga 3
   LED_OFF(LED4);  // Apaga 4
  }
  if (iBuff[1] == 2)
  {
    LED_OFF(LED5); // Apaga 5
    LED_OFF(LED6); // Apaga 6
  }
  if (iBuff[1] == 3)
  {
    LED_OFF(LED7); // Apaga 7
```

```
                LED_OFF(LED8); // Apaga 8
            }
          }
        }
      }
    }
}
```

5.6 Conclusiones.

La comunicación serial es la base de la mayoría de las comunicaciones, y para establecerla podemos realizarla con varios dispositivos, elegimos los PIC's por que son económicos y fáciles de conseguir, además son muy manipulables y populares por su forma de ser programados, la base para la comunicación USB esta hecha de forma serial, así que una vez comprendido el funcionamiento de la comunicación serial fue posible implementarla con el USB.

Para el desarrollo de aplicaciones Standalone el uso de C# es más amigable que Java por sus herramientas "Lo que ves es lo que obtienes" "***What You See Is What You Get***" **(**WYSIWYG), además que C# pertenece a la plataforma .NET que es relativamente nueva con respecto a los lenguajes más utilizados como son C++, Visual C, Java, Visual Basic, etc., porque con el lanzamiento de esta nueva plataforma se planteo el objetivo de mezclar las ventajas de varios de los lenguajes como Java, Visual Basic y C++ entre otros, lo que hace que las aplicaciones de escritorio sean desarrolladas con mayor funcionalidad.

El sistema de control final está integrado por: la bomba de aceite, válvula hidráulica, motor hidráulico, fuente eléctrica, tarjeta electrónica y el software residente en una PC, a continuación se muestra un diagrama del sistema completo.

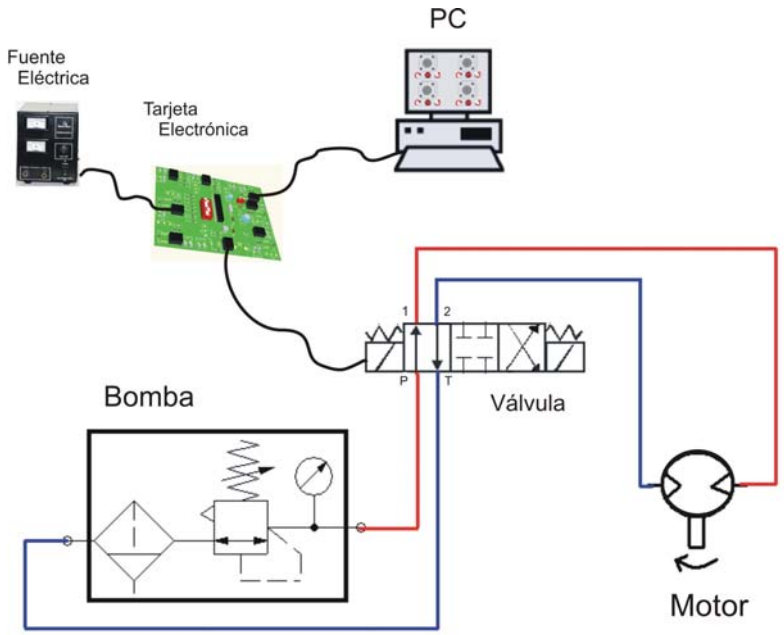


Fig. 5.4 Sistema Completo.

Teniendo el sistema armado se comprobó el correcto funcionamiento del mismo y se cumplió con el objetivo de activar la solenoide de la válvula hidráulica, generando movimiento en el manipulador MIRH1 y se obtuvieron los siguientes resultados:

- El software desarrollado se ejecuto correctamente y se logro la comunicación USB con la tarjeta de control.
- Los transistores de potencia TIP142 fueron polarizados de forma adecuada por los transistores secundarios, alimentando las solenoides con 23.3 volts y 2.8 amperes, suficientes para su operación.
- La bomba de aceite suministro 80 bars, adecuados para alimentar el sistema hidráulico.

Para futuras aplicaciones se recomienda adquirir más mangueras, conectores, así como fuentes eléctricas, de esta forma se podrá implementar una tarjeta que maneje mas válvulas y obtener el máximo desempeño del MIRH1.

Otra opción en la tarjeta de control es la implementación de una comunicación inalámbrica, lo que permitiría controlar el sistema remotamente desde una PC en RED o un dispositivo móvil como una PALM.

Para esté desarrollo se recomienda la utilización de microcontroladores “Rabbit RCM5600” en los cuales se puede cargar un mini “Servidor WEB”, para acceder al control a través de un explorador de Internet y de forma inalámbrica.

Referencias.

[1] http://www.intel.com/intelpress/usb/examples/ZipFiles/DUSBVC.htm
[2] http://javax-usb.org/
[3] http://jusb.sourceforge.net/?selected=api
[4] http://jcp.org/en/jsr/detail?id=80
[5] Appliying UML and Patterns
autor: Craig Larman
Second Edition,
Prentice Hall,
[6] Elementos y Herramientas en el Desarrollo de Sistemas de Información
Autores: Mario G. Piatinni, Sunil N. Daryananí
RA-MA Editorial
[8]www.microchip.com
[9]www.datasheetcatalog.com
[11] http://www.microsoft.com/whdc/connect/usb/winusb_howto.mspx
[12] http://www.codeguru.com/csharp/csharp/cs_data/article.php/c4217/
[13]http://www.csharpfriends.com/Articles/getArticle.aspx?articleID=351
[14]http://spyware.adslnet.es/genera.php?processfile=setupapi.dll&dir=s&pag=31
[15]http://www.codeproject.com/KB/cs/fscommand.aspx
[16]http://www.boschrexroth.de

Características técnicas (para utilización con valores distintos, consúltenos!)

Generalidades

Tamaño nominal		TN	6	10
Posición de montaje			A elección, preferentemente horizontal	
Rango de temperatura de almacenamiento		°C	−20 hasta +80	
Rango de tempera-	4WRA	°C	−20 hasta +70	
tura ambiente	4WRAE	°C	−20 hasta +50	
Masa	4WRA	kg	2,0	6,6
	4WRAE	kg	2,2	6,8

Hidráulicas (medidas con HLP46, $\vartheta_{ac} = 40\ °C \pm 5\ °C$)

Presión de servicio máx.	conexión A, B, P	bar	315	
	conexión T	bar	210	
Caudal nominal $q_{V\,nom}$ para $\Delta p = 10$ bar		l/min	7, 15, 26	30, 60
Caudal máx. admisible		l/min	42 (80)[1]	75 (140)[1]
Fluido hidráulico			Aceite mineral (HL, HLP) según DIN 51524 Otros fluidos hidráulicos a pedido!	
Rango de temperatura del fluido hidráulico		°C	−20 hasta +80 (preferentemente +40 hasta +50)	
Rango de viscosidad		mm²/s	20 hasta 380 (preferentemente 30 hasta 46)	
Grado máximo admisible de impurezas del fluido hidráulico clase de pureza según ISO 4406 (c)			Clase 20/18/15 [2]	
Histéresis		%	≤ 5	
Tensión de inversión		%	≤ 1	
Sensibilidad de respuesta		%	≤ 0,5	

[1] Caudal máx. admisible para circulación doble

[2] Las clases de pureza indicadas para los componentes del sistema hidráulico deben ser mantenidas. Un filtrado efectivo evita averías y aumenta simultáneamente la vida útil de los componentes.
Para la selección del filtro ver catálogos RS 50070, RS 50076, RS 50081, RS 50086 y RS 50088.

Características técnicas (para utilización con valores distintos, consúltenos!)

Eléctricas

Tamaño nominal		TN	6	10
Tipo de tensión			Continua	
Señal de valor nomi-	entrada de tensión "A1"	V	±10	
nal para WRAE	entrada de corriente "F1"	mA	4 a 20	
Corriente máxima por solenoide		A	2,5	
Resistencia de bobina	valor en frío a 20 °C	Ω	2	
	máx. valor en caliente	Ω	3	
Tiempo de conexión		%	100	
Temperatura máxima bobina [1]		°C	150	
Conexión eléctrica	4WRA		Con zócalo según DIN EN 175301-803 ó ISO 4400	
ver página 7			Conector según DIN EN 175301-803 ó ISO 4400 [2]	
	4WRAE		Con zócalo según DIN EN 175201-804	
			Conector según DIN EN 175201-804 [2]	
Tipo de protección de válvula según EN 60529			IP65 con conector montado y enclavado	

Electrónica de mando

Para 4WRA	amplificador digital en formato europeo [2]		VT-VSPD-1-2X (el salir en del centro de 2006)
	amplificador analógico formato europeo [2]		VT-VSPA2-1-2X/… según RS 30110
	módulo amplificador analógico [2]		VT-MSPA2-1-1X según RS 30228
Para 4WRAE			Integrada en la válvula, ver página 8
	módulo analógico de valor nominal		VT- SWMA-1-1X/... según RS 29902
	módulo analógico de valor nominal		VT-SWMKA-1-1X/... según RS 29903
	tarjeta digitale de valor nominal		VT-HACD-1-1X/... según RS 30143
	tarjeta analógica de valor nominal		VT-SWKA-1-1X/... según RS 30255
Tensión alimentación	tensión nominal	VCC	24
4WRAE, 4WRA [3]	valor límite inferior	V	21 / 22 (4WRA); 19 (4WRAE)
	valor límite superior	V	35
Consumo de corrien-	$I_{máx}$	A	1,8
te del amplificador	corriente de impulso máx.	A	3

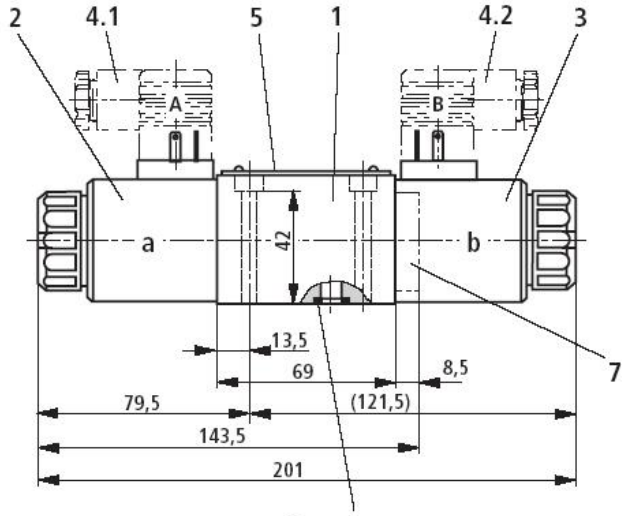
[1] Debido a las temperaturas resultantes en las superficies de bobina se deben tener en cuenta las normas europeas DIN EN 563 y DIN EN 982!

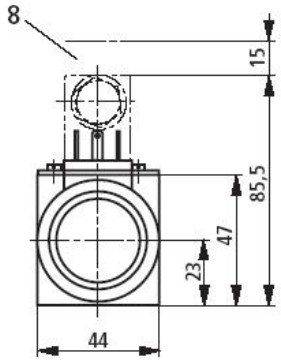
[2] Pedido por separado

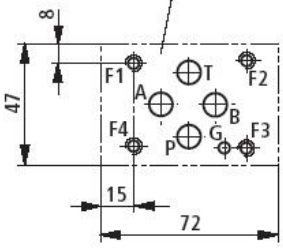
[3] Con electrónica de mando de la firma Bosch Rexroth AG

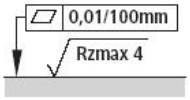
☞ **Observación:** Ver datos de **ensayo de simulación de medioambiente** para el análisis de la resistencia a perturbaciones electromagnéticas, solicitaciones climáticas y mecánicas en RS 29055-U (aclaraciones sobre resistencia al medioambiente).

Dimensiones: tipo 4WRA 6 (medidas nominales en mm) TN6









Terminación requerida de la superficie de montaje de la válvula

Tolerancias según: – tolerancias generales ISO 2768-mK

1 Carcasa de válvula

2 Solenoide proporcional "a"

3 Solenoide proporcional "b"

4.1 Conector "A", color gris, pedido por separado
ver página 7

4.2 Conector "B", color negro, pedido por separado
ver página 7

5 Placa de características

6 Juntas iguales para conexiones A, B, P y T

7 Tapón para válvula con un solenoide
(2 posiciones de conmutación, versión **EA** o **WA**)

8 Espacio requerido para retirar el conector

9 Superficie mecanizada de la válvula,
posición de las conexiones según ISO 4401 (**con** perforación de fijación) código: 4401-03-02-0-94 (aclaración según ISO 5783)
Diferente de la norma:
– sin perforación de fijación „G“
– conexiones A, B, P y T con Ø8 mm

Placas de conexión según catálogo RS 45052 y tornillos de sujeción de la válvula deben pedirse por separado.

Placas de conexión: G341/01 (G1/4)
G342/01 (G3/8)
G502/01 (G1/2)

Tornillos de sujeción de válvula (pedido por separado)
Se recomiendan los siguientes tornillos de sujeción de válvula:

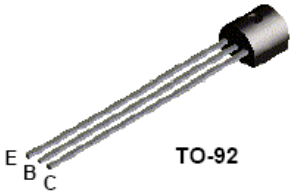
– **4 tornillos cilíndricos ISO 4762 - M5 x 50 - 10.9-flZn-240h-L**
(coeficiente de fricción μ_{total} = 0,09 a 0,14)
par de apriete M_A = 7 Nm ± 10%,
nro. de referencia **R913000064** (pedido por separado)
o

– **4 tornillos cilíndricos ISO 4762 - M5 x 50 - 10.9**
(coeficiente de fricción μ_{total} = 0,12 a 0,17)
par de apriete M_A = 8,9 Nm ± 10%

FAIRCHILD SEMICONDUCTOR™

Discrete POWER & Signal Technologies

BC547
BC547A
BC547B
BC547C



TO-92

NPN General Purpose Amplifier

This device is designed for use as general purpose amplifiers and switches requiring collector currents to 300 mA. Sourced from Process 10. See PN100A for characteristics.

Absolute Maximum Ratings*

TA = 25°C unless otherwise noted

Symbol	Parameter	Value	Units
V_{CEO}	Collector-Emitter Voltage	45	V
V_{CES}	Collector-Base Voltage	50	V
V_{EBO}	Emitter-Base Voltage	6.0	V
I_C	Collector Current - Continuous	500	mA
T_J, T_{stg}	Operating and Storage Junction Temperature Range	-55 to +150	°C

*These ratings are limiting values above which the serviceability of any semiconductor device may be impaired.

NOTES:
1) These ratings are based on a maximum junction temperature of 150 degrees C.
2) These are steady state limits. The factory should be consulted on applications involving pulsed or low duty cycle operations.

Thermal Characteristics

TA = 25°C unless otherwise noted

Symbol	Characteristic	Max	Units
		BC547 / A / B / C	
P_D	Total Device Dissipation Derate above 25°C	625 5.0	mW mW/°C
$R_{\theta JC}$	Thermal Resistance, Junction to Case	83.3	°C/W
$R_{\theta JA}$	Thermal Resistance, Junction to Ambient	200	°C/W

NPN General Purpose Amplifier
(continued)

Electrical Characteristics
TA = 25°C unless otherwise noted

Symbol	Parameter	Test Conditions		Min	Max	Units
OFF CHARACTERISTICS						
$V_{(BR)CEO}$	Collector-Emitter Breakdown Voltage	I_C = 1.0 mA, I_B = 0		45		V
$V_{(BR)CBO}$	Collector-Base Breakdown Voltage	I_C = 10 µA, I_E = 0		50		V
$V_{(BR)CES}$	Collector-Base Breakdown Voltage	I_C = 10 µA, I_E = 0		50		V
$V_{(BR)EBO}$	Emitter-Base Breakdown Voltage	I_E = 10 µA, I_C = 0		6.0		V
I_{CBO}	Collector Cutoff Current	V_{CB} = 30 V, I_E = 0 V_{CB} = 30 V, I_E = 0, T_A = +150 °C			15 5.0	nA µA
ON CHARACTERISTICS						
h_{FE}	DC Current Gain	V_{CE} = 5.0 V, I_C = 2.0 mA	**547** **547A** **547B** **547C**	110 110 200 420	800 220 450 800	
$V_{CE(sat)}$	Collector-Emitter Saturation Voltage	I_C = 10 mA, I_B = 0.5 mA I_C = 100 mA, I_B = 5.0 mA			0.25 0.60	V V
$V_{BE(on)}$	Base-Emitter On Voltage	V_{CE} = 5.0 V, I_C = 2.0 mA V_{CE} = 5.0 V, I_C = 10 mA		0.58	0.70 0.77	V V
SMALL SIGNAL CHARACTERISTICS						
h_{fe}	Small-Signal Current Gain	I_C = 2.0 mA, V_{CE} = 5.0 V, f = 1.0 kHz		125	900	
NF	Noise Figure	V_{CE} = 5.0 V, I_C = 200 µA, R_S = 2.0 kΩ, f = 1.0 kHz, B_W = 200 Hz			10	dB

PNP general purpose transistors BC556; BC557

FEATURES

- Low current (max. 100 mA)
- Low voltage (max. 65 V).

APPLICATIONS

- General purpose switching and amplification.

DESCRIPTION

PNP transistor in a TO-92; SOT54 plastic package.
NPN complements: BC546 and BC547.

PINNING

PIN	DESCRIPTION
1	emitter
2	base
3	collector

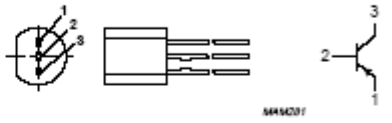


Fig.1 Simplified outline (TO-92; SOT54) and symbol.

LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134).

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{CBO}	collector-base voltage	open emitter			
	BC556		–	–80	V
	BC557		–	–50	V
V_{CEO}	collector-emitter voltage	open base			
	BC556		–	–65	V
	BC557		–	–45	V
V_{EBO}	emitter-base voltage	open collector	–	–5	V
I_C	collector current (DC)		–	–100	mA
I_{CM}	peak collector current		–	–200	mA
I_{BM}	peak base current		–	–200	mA
P_{tot}	total power dissipation	$T_{amb} \leq 25$ °C	–	500	mW
T_{stg}	storage temperature		–65	+150	°C
T_j	junction temperature		–	150	°C
T_{amb}	operating ambient temperature		–65	+150	°C

THERMAL CHARACTERISTICS

SYMBOL	PARAMETER	CONDITIONS	VALUE	UNIT
$R_{th\ j\text{-}a}$	thermal resistance from junction to ambient	note 1	250	K/W

Note

1. Transistor mounted on an FR4 printed-circuit board.

CHARACTERISTICS

T_j = 25 °C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
I_{CBO}	collector cut-off current	I_E = 0; V_{CB} = –30 V	–	–1	–15	nA
		I_E = 0; V_{CB} = –30 V; T_j = 150 °C	–	–	–4	µA
I_{EBO}	emitter cut-off current	I_C = 0; V_{EB} = –5 V	–	–	–100	nA
h_{FE}	DC current gain	I_C = –2 mA; V_{CE} = –5 V; see Figs 2, 3 and 4				
	BC556		125	–	475	
	BC557		125	–	800	
	BC556A		125	–	250	
	BC556B; BC557B		220	–	475	
	BC557C		420	–	800	
V_{CEsat}	collector-emitter saturation voltage	I_C = –10 mA; I_B = –0.5 mA	–	–60	–300	mV
		I_C = –100 mA; I_B = –5 mA	–	–180	–650	mV
V_{BEsat}	base-emitter saturation voltage	I_C = –10 mA; I_B = –0.5 mA; note 1	–	–750	–	mV
		I_C = –100 mA; I_B = –5 mA; note 1	–	–930	–	mV
V_{BE}	base-emitter voltage	I_C = –2 mA; V_{CE} = –5 V; note 2	–600	–650	–750	mV
		I_C = –10 mA; V_{CE} = –5 V; note 2	–	–	–820	mV
C_c	collector capacitance	I_E = i_e = 0; V_{CB} = –10 V; f = 1 MHz	–	3	–	pF
C_e	emitter capacitance	I_C = i_c = 0; V_{EB} = –0.5 V; f = 1 MHz	–	10	–	pF
f_T	transition frequency	I_C = –10 mA; V_{CE} = –5 V; f = 100 MHz	100	–	–	MHz
F	noise figure	I_C = –200 µA; V_{CE} = –5 V; R_S = 2 kΩ; f = 1 kHz; B = 200 Hz	–	2	10	dB

Notes

1. V_{BEsat} decreases by about –1.7 mV/K with increasing temperature.
2. V_{BE} decreases by about –2 mV/K with increasing temperature.

Order this document
by TIP140/D

Darlington Complementary Silicon Power Transistors

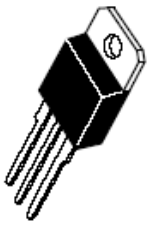
. . . designed for general–purpose amplifier and low frequency switching applications.

- High DC Current Gain — Min h_{FE} = 1000 @ I_C = 5 A, V_{CE} = 4 V
- Collector–Emitter Sustaining Voltage — @ 30 mA
  $V_{CEO(sus)}$ = 60 Vdc (Min) — TIP140, TIP145
  80 Vdc (Min) — TIP141, TIP146
  100 Vdc (Min) — TIP142, TIP147
- Monolithic Construction with Built–In Base–Emitter Shunt Resistor

NPN
TIP140
TIP141*
TIP142*
PNP
TIP145
TIP146*
TIP147*

*Motorola Preferred Device

10 AMPERE
DARLINGTON
COMPLEMENTARY SILICON
POWER TRANSISTORS
60–100 VOLTS
125 WATTS



CASE 340D–01

MAXIMUM RATINGS

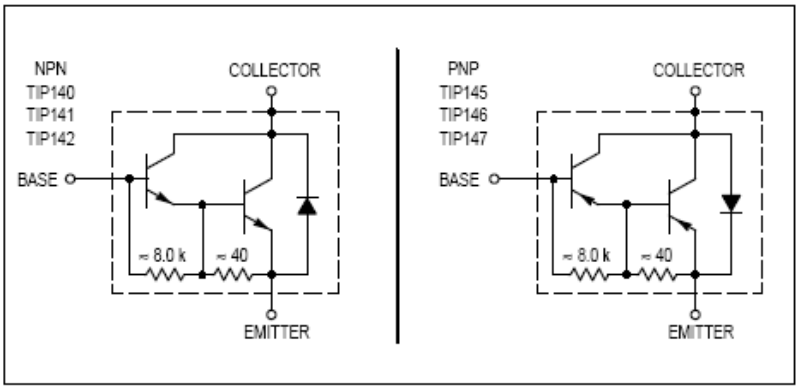
Rating	Symbol	TIP140 TIP145	TIP141 TIP146	TIP142 TIP147	Unit
Collector–Emitter Voltage	V_{CEO}	60	80	100	Vdc
Collector–Base Voltage	V_{CB}	60	80	100	Vdc
Emitter–Base Voltage	V_{EB}	5.0			Vdc
Collector Current — Continuous Peak (1)	I_C	10 15			Adc
Base Current — Continuous	I_B	0.5			Adc
Total Device Dissipation @ T_C = 25°C	P_D	125			Watts
Operating and Storage Junction Temperature Range	T_J, T_{stg}	–65 to +150			°C

THERMAL CHARACTERISTICS

Characteristic	Symbol	Max	Unit
Thermal Resistance, Junction to Case	$R_{\theta JC}$	1.0	°C/W
Thermal Resistance, Case to Ambient	$R_{\theta JA}$	35.7	°C/W

(1) 5 ms, ≤ 10% Duty Cycle.

DARLINGTON SCHEMATICS



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

MOTOROLA

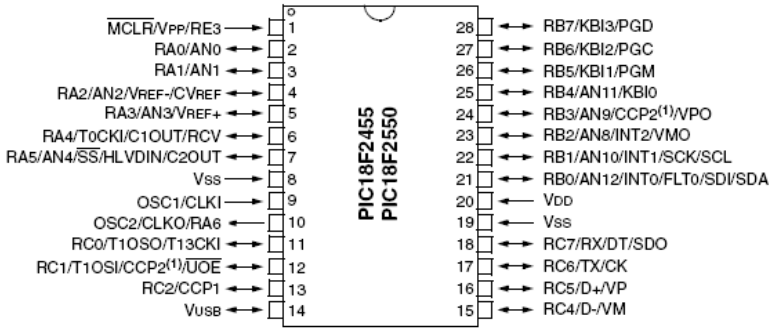
TIP140 TIP141 TIP142 TIP145 TIP146 TIP147

ELECTRICAL CHARACTERISTICS (T_C = 25°C unless otherwise noted)

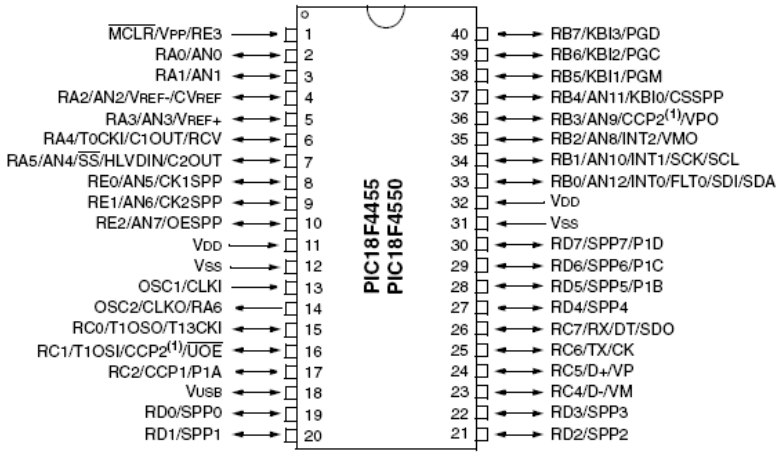
Characteristic		Symbol	Min	Typ	Max	Unit
OFF CHARACTERISTICS						
Collector–Emitter Sustaining Voltage (1)		$V_{CEO(sus)}$				Vdc
(I_C = 30 mA, I_B = 0)	TIP140, TIP145		60	—	—	
	TIP141, TIP146		80	—	—	
	TIP142, TIP147		100	—	—	
Collector Cutoff Current		I_{CEO}				mA
(V_{CE} = 30 Vdc, I_B = 0)	TIP140, TIP145		—	—	2.0	
(V_{CE} = 40 Vdc, I_B = 0)	TIP141, TIP146		—	—	2.0	
(V_{CE} = 50 Vdc, I_B = 0)	TIP142, TIP147		—	—	2.0	
Collector Cutoff Current		I_{CBO}				mA
(V_{CB} = 60 V, I_E = 0)	TIP140, TIP145		—	—	1.0	
(V_{CB} = 80 V, I_E = 0)	TIP141, TIP146		—	—	1.0	
(V_{CB} = 100 V, I_E = 0)	TIP142, TIP147		—	—	1.0	
Emitter Cutoff Current (V_{BE} = 5.0 V)		I_{EBO}	—	—	2 0	mA
ON CHARACTERISTICS (1)						
DC Current Gain		h_{FE}				—
(I_C = 5.0 A, V_{CE} = 4.0 V)			1000	—	—	
(I_C = 10 A, V_{CE} = 4.0 V)			500	—	—	
Collector–Emitter Saturation Voltage		$V_{CE(sat)}$				Vdc
(I_C = 5.0 A, I_B = 10 mA)			—	—	2.0	
(I_C = 10 A, I_B = 40 mA)			—	—	3.0	
Base–Emitter Saturation Voltage (I_C = 10 A, I_B = 40 mA)		$V_{BE(sat)}$	—	—	3.5	Vdc
Base–Emitter On Voltage (I_C = 10 A, V_{CE} = 4.0 Vdc)		$V_{BE(on)}$	—	—	3.0	Vdc
SWITCHING CHARACTERISTICS						
Resistive Load (See Figure 1)						
Delay Time	(V_{CC} = 30 V, I_C = 5.0 A, I_B = 20 mA, Duty Cycle ≤ 2.0%, I_{B1} = I_{B2}, R_C & R_B Varied, T_J = 25°C)	t_d	—	0.15	—	µs
Rise Time		t_r	—	0.55	—	µs
Storage Time		t_s	—	2.5	—	µs
Fall Time		t_f	—	2.5	—	µs

Pin Diagrams

28-Pin PDIP, SOIC



40-Pin PDIP



Note 1: RB3 is the alternate pin for CCP2 multiplexing.

28.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings(†)

Ambient temperature under bias ..... -40°C to +85°C
Storage temperature ..... -65°C to +150°C
Voltage on any pin with respect to V_{SS} (except V_{DD}, $\overline{MCLR}$ and RA4) ..... -0.3V to (V_{DD} + 0.3V)
Voltage on V_{DD} with respect to V_{SS} ..... -0.3V to +7.5V
Voltage on $\overline{MCLR}$ with respect to V_{SS} **(Note 2)** ..... 0V to +13.25V
Total power dissipation **(Note 1)** 1.0W
Maximum current out of V_{SS} pin ..... 300 mA
Maximum current into V_{DD} pin ..... 250 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD}) ..... ±20 mA
Output clamp current, I_{OK} (V_O < 0 or V_O > V_{DD}) ..... ±20 mA
Maximum output current sunk by any I/O pin ..... 25 mA
Maximum output current sourced by any I/O pin ..... 25 mA
Maximum current sunk by all ports ..... 200 mA
Maximum current sourced by all ports ..... 200 mA

Note 1: Power dissipation is calculated as follows:
$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$

2: Voltage spikes below V_{SS} at the $\overline{MCLR}$/V_{PP}/RE3 pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{MCLR}$/V_{PP}/RE3 pin, rather than pulling this pin directly to V_{SS}.

† **NOTICE:** Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

FIGURE 28-1: PIC18F2455/2550/4455/4550 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)

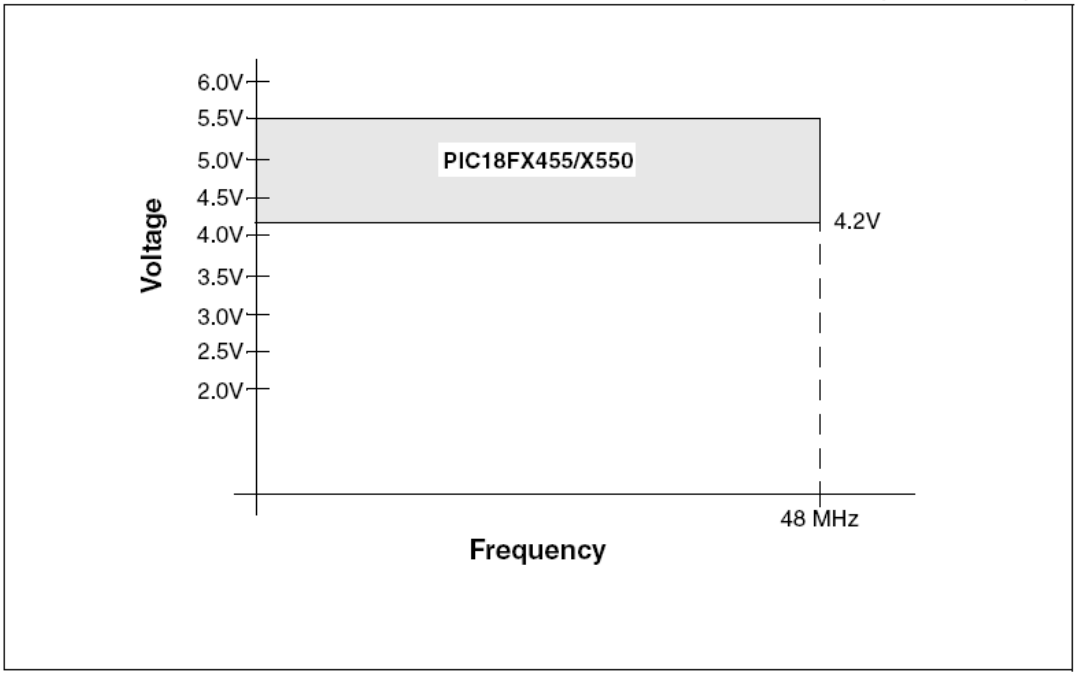
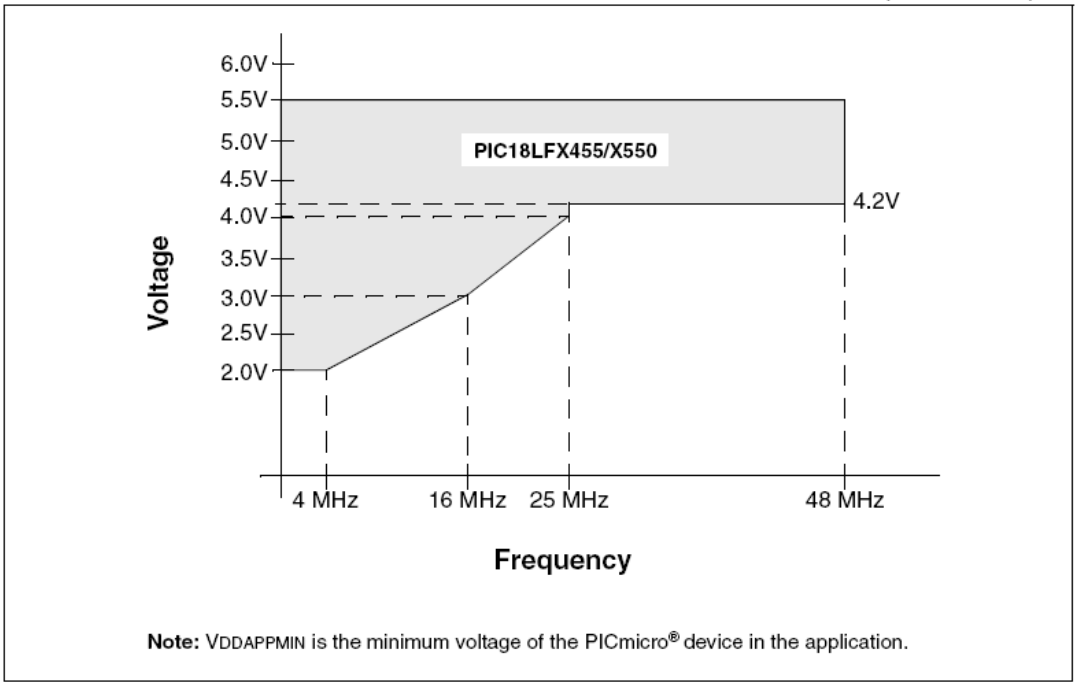


FIGURE 28-2: PIC18LF2455/2550/4455/4550 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)



Note: VDDAPPMIN is the minimum voltage of the PICmicro® device in the application.

