TEORÍA GENERAL

1.1 TEORÍA DE LOS ROBOTS SEGUIDORES DE LÍNEA

Un **robot** es un dispositivo generalmente electromecánico, que desempeña tareas automáticamente, ya sea de acuerdo a la supervisión humana directa, a través de un programa predefinido o siguiendo un conjunto de reglas generales, utilizando técnicas de inteligencia artificial. Generalmente estas tareas reemplazan, asemejan o extienden el trabajo humano, como ensamble en líneas de manufactura, manipulación de objetos pesados o peligrosos, trabajo en el espacio, etc. Usualmente, la inteligencia controlada por una computadora o un microcontrolador ejecutando un programa.[1]

Los robots hoy en día han pasado a formar parte de nuestra vida cotidiana tanto en la industria como en nuestra vida diaria, existen maquinas automatizadas y robots en muchos lugares que nos facilitan el trabajo, que van desde brazos robotizados en una armadora de autos hasta un cajero automático.

En la actualidad se ha incrementado la investigación en el campo de los robots móviles, debido al alto grado de autonomía, comparado con los manipuladores. Los robots móviles tienen la facilidad de desplazarse libremente sobre un entorno. Este tipo de robots tienen diferentes aplicaciones como son: investigación espacial, rastreadores de minas, para desactivar bombas, como ayuda para discapacitados, y de servicio en industria. [17]

Existen varios tipos de control para los robots que son: de los tipos autónomos y teledirigidos, un robot del tipo teledirigido necesitan la intervención de un operador humano ya sea en forma parcial o total. Los del tipo autónomo, son capaces de tomar decisiones basándose en la comprensión del entorno que los rodea.[17]

Uno de los principales problemas en el diseño de robots autónomos es el diseño de algoritmos que sean capaces de tomar decisiones en tiempo real además de simultáneamente percibir y analizar el entorno.

El entorno esta compuesto por el medio en el cual se desplaza el robot como los obstáculos y los factores que intervienen como la luz, temperatura, presión, campos magnéticos y otros. Es posible clasificarlos en cuatro tipos básicos, acuáticos, terrestres, aéreos y espaciales. Este define los mecanismos que permiten al robot movilizarse. Sumado a la tarea a realizar también define la cantidad de sensores y actuadores, la potencia necesaria, la complejidad de los algoritmos de toma de decisiones, etc. [17]

Existen también entornos naturales y controlados. Es el entorno natural en donde un ser humano realiza su actividad, no es perjudicial para su salud, y las condiciones del entorno pueden variar sin previo aviso. Un ejemplo de esta variación es el clima y la luz ambiente que son consideradas como variable del entorno.

En cuanto a los entornos controlados estos son creados por el ser humano, y están compuestos por algunas variables como son el terreno, el ambiente, los contrincantes, etc. Estas se encuentran acotadas dentro de valores conocidos, dejando solo de ser necesario un cierto margen fuera de control (sin intervención), esto es muy común cuando se necesita un análisis próximo a la realidad.

El entorno terrestre: Los robots creados para este entorno son de fácil construcción y se basan en conceptos de mecánica muchas veces ya probados. En muchos casos solo es cuestión de dotar de inteligencia a un vehículo convencional. [17]

Pero para poder percibir el ambiente que lo rodea, es necesaria la utilización de sensores. Un **sensor** es un dispositivo que detecta manifestaciones de cualidades o fenómenos físicos, como la energía, velocidad, aceleración, tamaño, cantidad, etc. [2]

La parte de la inteligencia de un robot móvil, generalmente esta controlada por un microcontrolador.

Un **microcontrolador** es un sistema optimizado para ser utilizado en le control de equipos electrónicos. Es un Circuito integrado o chip que incluye en su interior las tres unidades funcionales de un ordenador: CPU, Memoria y Unidades de E/S, es decir, se trata de un computador completo en un solo circuito integrado. Aunque sus prestaciones son limitadas, además de dicha integración, su característica principal es su alto nivel de especialización.[3]

Los robots móviles tienen un papel muy importante en el campo de la robótica, ya que estos tienen la capacidad de desplazarse por aire por agua o por tierra, sin necesidad de ser tripulados. Sin embargo nos enfocaremos a hablar de robots del tipo terrestres que se desplazan por medio de ruedas mejor conocidos como tipo vehiculo, aunque existen otras formas de desplazarse, como orugas, robots con patas, o del tipo ápodo que no usan ni ruedas ni patas para desplazarse (estos últimos se mueven mediante ondas sinusoidales que recorren el cuerpo del robot, parecido al movimiento de un gusano). Y exclusivamente de los robots rastreadores.

En este tipo de robots es usual encontrar sensores del tipo visión simple, como son ultrasónicos, láser, infrarrojos, de efecto hall, optoreflectivos, los cuales se encargan de detectar la presencia de algún objeto y en algunos casos se puede calcular la distancia del sensor al objeto con ayuda de un microcontrolador.

Los sensores optoreflectivos son muy usados en la detección y seguimiento de líneas.

En este caso se sensa la diferencia entre la cantidad de luz reflejada que es proporcional al color de la superficie reflectante, cuando varia la intensidad lumínica significa que el sensor esta desenfocado con respecto a la línea, entonces corrige la trayectoria hasta obtener nuevamente la intensidad lumínica reflectante que representa la línea . Este tipo de robots es llamado "rastreadores".

Los robots rastreadores o seguidores de línea usualmente se componen de las siguientes etapas: sensado, acondicionamiento de señal, control, potencia, actuadores y sistema mecánico.

En la parte de sensado el robot utiliza sensores normalmente del tipo optoreflectivo para poder ver la línea, y se basa en la cantidad de luz reflejada para poder discriminar cuando esta viendo un color u otro.

En la etapa de acondicionamiento de señal, es como su nombre lo indica se ajusta la señal ya sea para amplificarla en corriente o en voltaje y se transforma en una señal digital, para finalmente ser entregada a el microcontrolador.

En la etapa de control, podemos encontrar normalmente microcontroladores en los cuales se programan los algoritmos de control, en este caso pueden ser algoritmos que controlan la velocidad, dirección, y en algunos casos algoritmos para buscar la línea en caso de que el robot la llegue a perder, ya sea por exceso de velocidad o por inercia, el microcontrolador entrega las señales de control para la corrección de la posición a la etapa de potencia.

La etapa de potencia normalmente se compone de transistores de potencia para amplificar en corriente las señales que llegan del microcontrolador,

aunque es usual encontrar circuitos integrados especializados para este tipo de aplicación.

En cuanto a los actuadores podemos encontrar: solenoides, motores de corriente directa, motores de pasos, y aunque no es muy usual motores de corriente alterna.

Estos motores son alimentados por la etapa de potencia ya descrita anteriormente, y se encargan del movimiento del robot como son la tracción y la dirección del mismo.

El sistema mecánico se refiere a la morfología del mismo, como lo es el tipo de chasis, la forma de acomodar los motores, la forma de acomodar las llantas y el diseño de toda la estructura, ya que este es uno de los parámetros más importantes en el diseño de los robots lo explicaremos en forma mas detallada en el tema 1.4.

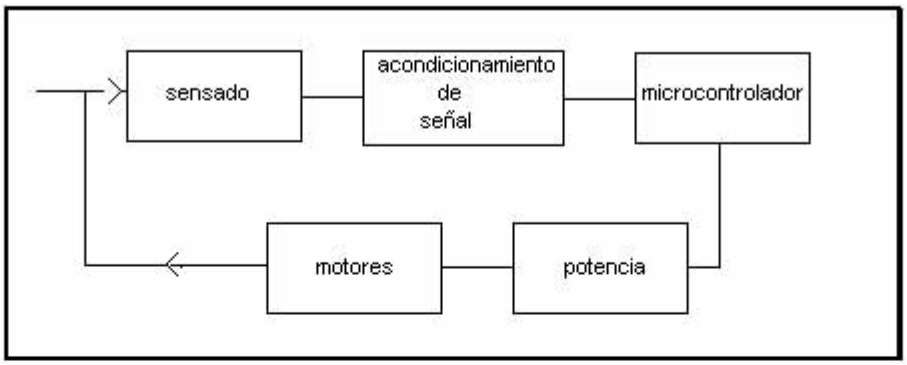


Figura 1.1.1 se muestra el diagrama a bloques de las etapas de un robot móvil.

1.2 MÉTODOS DE RASTREO DE LÍNEAS

Existen varios métodos para rastreo de líneas, en esta sección vamos a considerar una línea blanca sobre una superficie negra, que es la más común para este tipo de robots.

Con un sensor

Este método es bastante sencillo aunque depende de condiciones anteriores del robot, ya que va corrigiendo dependiendo del último estado del sensor y de la última corrección que hizo. Consideremos la figura 1.2.1

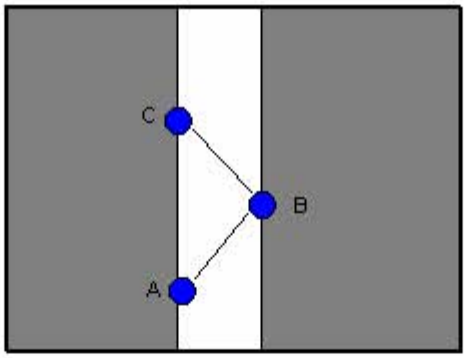


Figura 1.2.1 Rastreo de línea con 1 sensor.

La trayectoria del sensor AB la consideraremos como corrección derecha y la trayectoria BC como corrección izquierda cuando el sensor llega a una parte negra indica fin de corrección y debe de cambiar el sentido de la corrección, esto es, si el estado anterior al fin de corrección fue corrección derecha, el nuevo estado de la dirección será corrección derecha y así sucesivamente. Este algoritmo es muy fácil de programar en un microcontrolador además requiere únicamente un sensor para ver la línea, y es el que requiere el mínimo de componentes para ver la línea y es difícil que pierda la línea, sin embargo este algoritmo necesita de memoria para observar el estado anterior del sistema y así aplicar la nueva corrección. Sin embargo una desventaja de este

método es que en el caso de que la velocidad se incremente demasiado y si las curvas son lo suficientemente cerradas como para que el sensor vea el borde de la línea 2 veces del mismo lado, el robot perdería la línea.

Con dos sensores

Un circuito de los más simples que nos permite seguir una línea blanca sin necesidad de microprocesadores es el mostrado el la figura 1.2.2

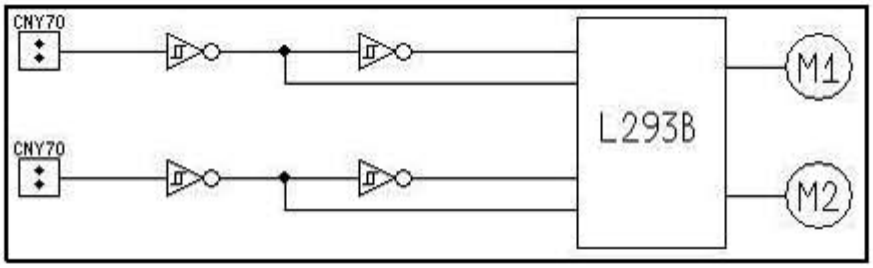


Figura 1.2.2 circuito de control para seguidor de línea.[14]

Con dos sensores viendo por afuera de la línea, como se muestra en la figura 1.2.3, en este tipo de configuración los sensores se acomodan de tal forma que la línea quede entre los 2 sensores, y en el momento de que alguno de los sensores vea la línea el sistema de control del robot inmediatamente corrige la dirección del robot, por ejemplo si el sensor de la derecha ve la línea, el sistema de control debe cambiar de dirección al robot hacia la derecha y viceversa. Para este tipo de configuración es posible añadir sensores en la parte derecha a izquierda, para aumentar el rango de visión de la línea.

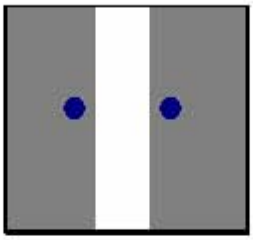


Figura 1.2.3 configuraciones de los sensores

Este tipo de configuración es una de las mas simples, no requiere uso de memorias o microcontroladores y es el mas sencillo de implementar, sin embargo una de las deficiencias de este método es que se dificulta la búsqueda de la línea cuando esta no se encuentra en medio de los sensores o cuando por cuestiones de velocidad o inercia el robot llega a perder la línea.

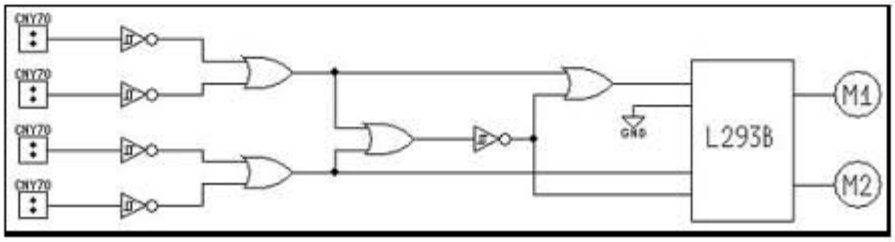


Figura 1.2.4 circuito mejorado seguidor de línea.[14]

En el circuito 1.2.4 se muestra una mejora al circuito 1 con el cual se puede incrementar el rango de visión de los sensores.

Con dos sensores dentro de la línea

Esta configuración es una de las que tienen mejor desempeño para seguidores de línea de alta velocidad. En esta configuración los sensores se acomodan de tal forma que queden dentro de la línea para que en el momento de que alguno de ellos abandone la línea, el sistema de control corrija hacia el lado opuesto y se muestra en la figura 1.2.5

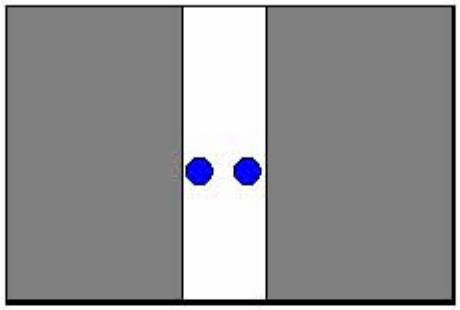


Figura 1.2.5 Los dos sensores dentro de la línea

En este tipo de configuración es posible implantar un algoritmo de búsqueda de línea bastante sencillo, solo basta con recordar el ultimo estado del sistema, por ejemplo si el censor derecho se sale de la línea, el robot empezara a corregir su dirección hacia la izquierda, pero en casos extremos en que la inercia del robot haga que el sensor izquierdo también se salga de la línea, el robot deberá seguir manteniendo el estado de corrección hacia la izquierda, hasta que ambos sensores vean la línea de nuevo. Este es uno de los mejores métodos de rastreo de líneas probado por el autor.

Con tres sensores

En esta configuración normalmente se controla por estados, es muy usada cuando el espesor de la línea es muy delgado. Cuando el sensor central esta dentro de la línea, el robot avanzara derecho y cuando uno de los sensores laterales ve la línea, corrige dirección. Por ejemplo cuando el sensor derecho ve la línea, el robot corrige la dirección hacia la derecha y viceversa. La forma de acomodarlos se muestra en la figura 1.2.6.

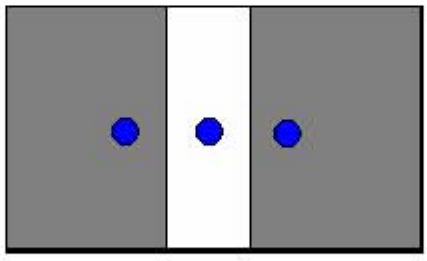


Figura 1.2.6 Rastreo con 3 sensores

Estos son los principales métodos de rastreo de líneas, con sensores infrarrojos optoreflectivos, aunque cabe mencionar que también se pueden rastrear con otro tipo de sensores como lo son los sensores láser. También se pueden utilizar otro tipo de métodos de rastreo de líneas como control difuso y técnicas de procesamiento digital de señales.

1.3 CARACTERÍSTICAS, VENTAJAS Y DESVENTAJAS DE LOS MOTORES DE CORRIENTE DIRECTA

Un motor eléctrico es esencialmente una máquina que convierte energía eléctrica en movimiento o trabajo mecánico, a través de medios electromagnéticos. [15]

Existen básicamente tres tipos de motores eléctricos:

a) Los Motores de Corriente Alterna [C.A.]. Son los tipos de motores más usados en la industria, ya que estos equipos se alimentan con los sistemas de distribución. De acuerdo a su alimentación se dividen en tres tipos: Monofásicos (1 fase) Bifásicos (2 fases) y Trifásicos (3 fases).[15]

b) Los Motores Universales. Tienen la forma de un motor de corriente continua, la principal diferencia es que esta diseñado para funcionar con corriente alterna. El inconveniente de este tipo de motores es su eficiencia, ya que es baja (del orden del 51%), pero como se utilizan en maquinas de pequeña potencia, ésta no se considera importante, además, su operación debe ser intermitente, de lo contrario, éste se quemaría. Estos motores son utilizados en taladros, aspiradoras, licuadoras, etc.[15]

c) Los Motores de Corriente Directa [C.D.] o Corriente Continua [C.C.]. Se utilizan en casos en los que es importante el poder regular continuamente la velocidad del motor, además, se utilizan en aquellos casos en los que es imprescindible utilizar corriente directa, como es el caso de motores accionados por pilas o baterías. [15]

Los motores de corriente continua se usan en una amplia variedad de aplicaciones industriales en virtud de la facilidad con la que se puede controlar

la velocidad. La característica velocidad-par se puede hacer variar para casi cualquier forma útil. Los motores de corriente continua pueden entregar más de cinco veces el par nominal (si lo permite la alimentación de energía eléctrica). Se puede realizar la operación en reversa sin conmutar la energía eléctrica.

Los motores de corriente continua se pueden dividir dentro de dos grandes tipos: [16]

- Motores de imán permanente, entre ellos:
- Motores de corriente continua sin escobilla.
- Servomotores.

Y motores de corriente continua de campo devanado, los que a su vez se clasifican como: Motor en derivación, Motor devanado en serie, Motor en devanado mixto.

Motores en derivación

En este tipo de motores el devanado del campo está conectado en paralelo con la armadura, véase figura 1.3.1

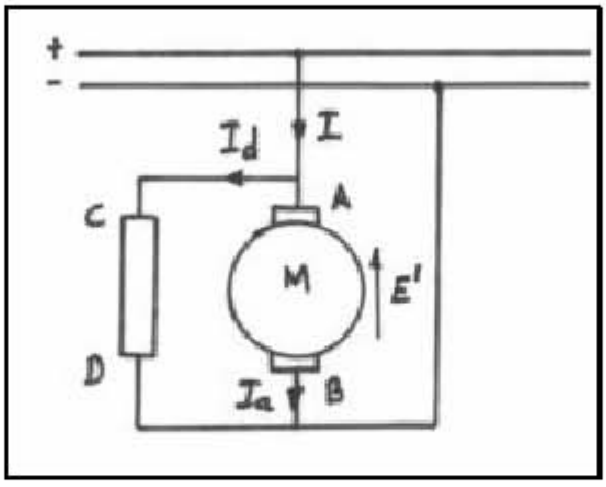


Figura 1.3.1 Circuito equivalente para un motor de DC conectado en derivación.[16]

Es el tipo de motor de corriente continua cuya velocidad no disminuye mas que ligeramente cuando el par aumenta. Excepcionalmente, la reacción del inducido debería ser suficientemente grande para que la característica de velocidad fuera ascendente al aumentar la carga. Vea la figura 1.3.2.

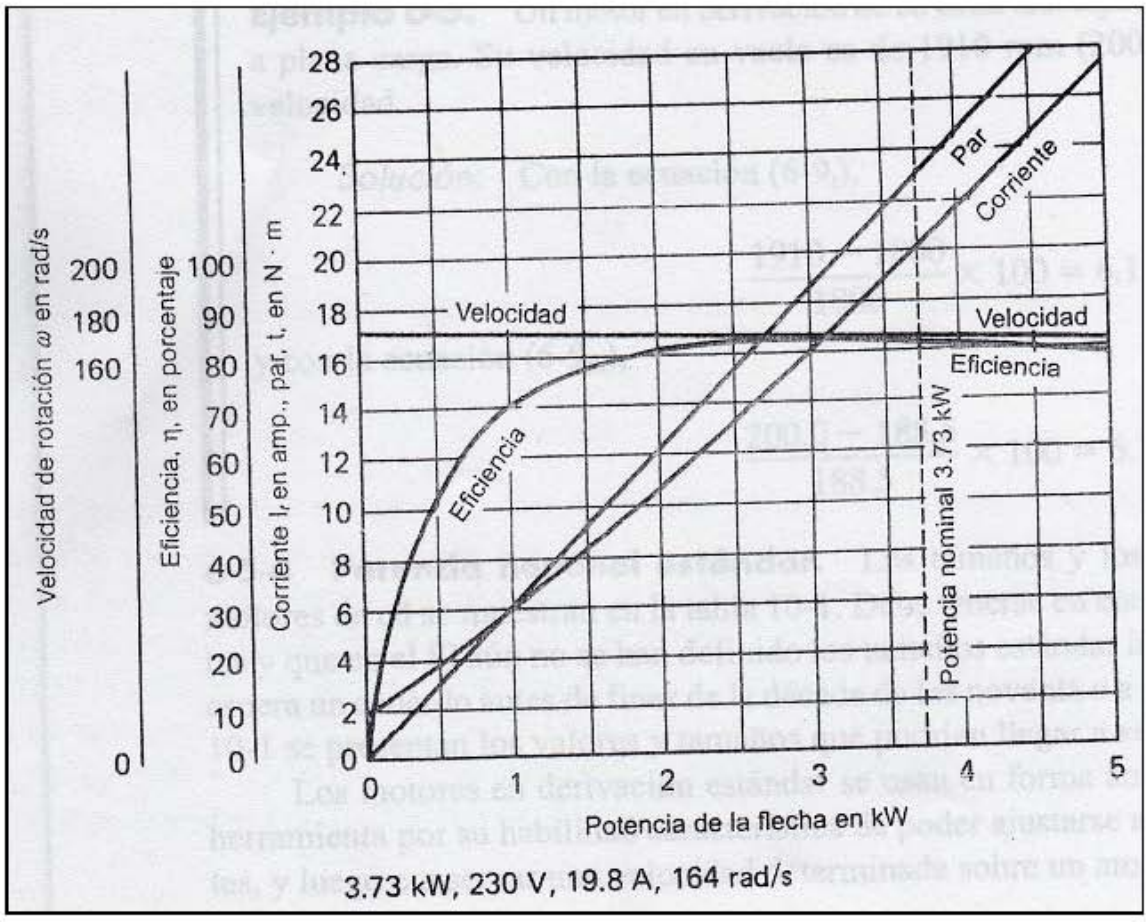


Figura 1.3.2 Características del motor en derivación.[26]

Los polos de conmutación han mejorado la conmutación de los dinamos de tal manera que es posible usar un entrehierro mucho más estrecho que antiguamente.

Como la armadura de un motor gira en un campo magnético, se genera una Fem. en los conductores que se opone a la dirección de la corriente y se le conoce como fuerza contraelectromotriz. La Fem. aplicada debe ser bastante grande como para vencer la fuerza contraelectromotriz y también para enviar la

corriente I_a de la armadura a través de Rm, la resistencia del devanado de la armadura y las escobillas.

Motor devanado en serie

En este tipo de motores el devanado del campo está conectado en serie con la armadura.

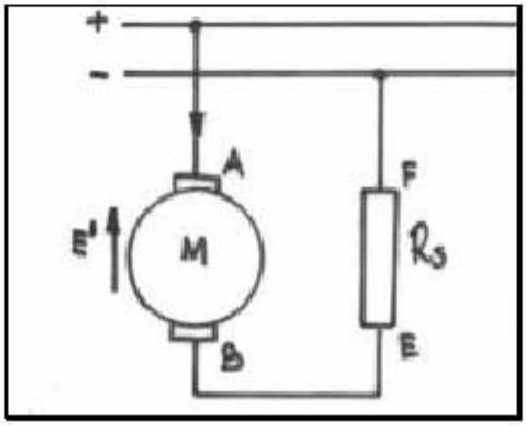


Figura1.3.3 Circuito equivalente para un motor de DC conectado en serie.[16]

Es el motor cuya velocidad disminuye sensiblemente cuando el par aumenta y cuya velocidad en vacío no tiene límite teóricamente. Véase la figura 1.3.4.

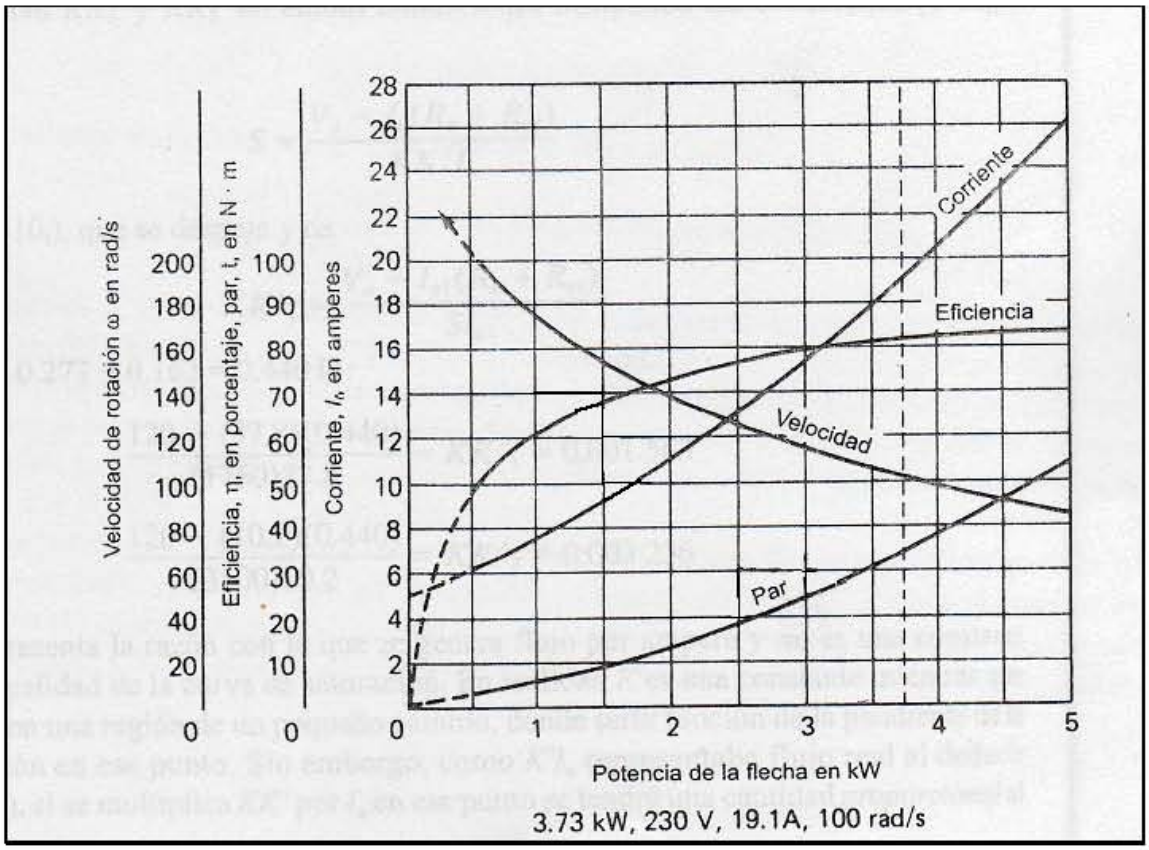


Figura 1.3.4 Características del motor serie.[26]

Los motores con excitación en serie son aquellos en los que el inductor esta conectado en serie con el inducido. El inductor tiene un número relativamente pequeño de espiras de hilo, que debe ser de sección suficiente para que se pase por él la corriente de régimen que requiere el inducido. En los motores serie, el flujo depende totalmente de la intensidad de la corriente del inducido.

Si el hierro del motor se mantiene a saturación moderada, el flujo será casi directamente proporcional a dicha intensidad. Velocidad y par de los motores devanados en serie.

Si la carga en un motor devanado en serie se hace pequeña, la velocidad aumenta mucho, de modo que un motor de este tipo siempre debe conectarse a la carga a través de un engranaje reductor o directamente. Si se conectara mediante banda y ésta se rompiera, la velocidad del motor se dispararía y el motor probablemente estallaría.

Para una carga dada y, por lo tanto, para una corriente dada, la velocidad de un motor devanado en serie se puede incrementar al poner en derivación el devanado en serie, o bien, al poner en cortocircuito algunas de las vueltas en serie, de modo que se reduzca el flujo magnético. La velocidad se puede reducir al introducir una resistencia en serie con la armadura.

Motor devanado mixto

En este tipo de motor se tiene un devanado del campo en serie y otro en paralelo.

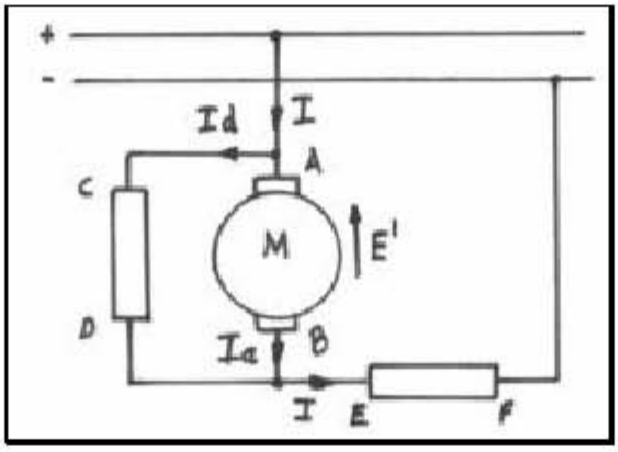


Figura 1.3.5 Circuito equivalente para un motor de DC conectado en configuración mixto.[16]

El motor en devanado mixto es un término medio entre los motores devanados en serie y los de en derivación. En virtud de la existencia del devanado en serie, que ayuda al devanado en derivación, el flujo magnético por polo aumenta con la carga, de modo que el par se incrementa con mayor rapidez y la velocidad disminuye más rápidamente que si no estuviera conectado el devanado en serie; pero el motor no se puede desbocar con cargas ligeras, por la presencia de la excitación en derivación.

Motores de corriente continua de imán permanente

Existen motores de imán permanente (PM, permanent magnet), en tamaños de fracciones de caballo y de números pequeños enteros de caballos. Tienen varias ventajas respecto a los del tipo de campo devanado. No se necesitan las alimentaciones de energía eléctrica para excitación ni el devanado asociado. Se mejora la confiabilidad, ya que no existen bobinas excitadoras del campo que fallen y no hay probabilidad de que se presente una sobre velocidad debida a pérdida del campo. Se mejoran la eficiencia y el enfriamiento por la eliminación de pérdida de potencia en un campo excitador. Así mismo, la característica par contra corriente se aproxima más a lo lineal. Un motor de imán permanente (PM) se puede usar en donde se requiere un motor por completo encerrado para un ciclo de servicio de excitación continua.

Los efectos de la temperatura dependen de la clase de material que se use en el imán. Los motores de número entero de caballos de potencia con imanes del tipo Álnico resultan menos afectados por la temperatura que los que tienen imanes de cerámica, porque el flujo magnético es constante. Por lo común, los imanes de cerámica que se utilizan en los motores de fracción de caballo, tienen características que varían con la temperatura muy aproximadamente como varían los campos en derivación de las máquinas excitadas. Las desventajas son la falta de control del campo y de características especiales velocidad-par. Las sobrecargas pueden causar desmagnetización parcial que cambia las características de velocidad y de par del motor, hasta que se restablece por completo la magnetización. En general, un motor PM de número entero de caballos es un poco más grande y más caro que un motor equivalente con devanado en derivación, pero el costo total del sistema puede ser menor. Un motor PM es un término medio entre los motores de devanado mixto y los devanados en serie. Tiene mejor par de arranque, pero alrededor de la mitad de la velocidad en vacío de un motor devanado en serie. En las figuras

1.3.6, 1.3.7 y 1.3.8 se muestran gráficamente las características de este tipo de motores.

Motores de corriente continua sin escobillas

Los motores de corriente continua sin escobillas tienen una armadura estacionaria y una estructura rotatoria del campo, exactamente en forma opuesta a como están dispuestos esos elementos en los motores convencionales de corriente directa. Esta construcción aumenta la rapidez de disipación del calor y reduce la inercia del rotor. Imanes permanentes suministran el flujo magnético para el campo. La corriente directa hacia la armadura se conmuta con transistores, en vez de las escobillas y las delgas del colector de los motores convencionales de corriente directa. Es normal que las armaduras de los motores de corriente continua sin escobillas contengan de dos a seis bobinas, en tanto que las armaduras de los motores convencionales de corriente continua contienen de 10 a 50. Los motores sin escobillas tienen menos bobinas porque se requieren dos o cuatro transistores para conmutar cada bobina del motor. Esta disposición se vuelve cada vez más costosa e ineficiente a medida que aumenta el número de devanados. Los transistores que controlan cada devanado de un motor sin escobillas de corriente continua se activan y desactivan a ángulos específicos del rotor. Los transistores suministran pulsos de comente a los devanados de la armadura, los cuales son semejantes a los que suministra un conmutador. La secuencia de conmutación se dispone para producir un flujo magnético rotatorio en el entrehierro, que permanece formando un ángulo fijo con el flujo magnético producido por los imanes permanentes del rotor. El par producido por un motor sin escobillas de corriente continua es directamente proporcional a la corriente de la armadura. Como se muestra en la figura 1.3.6

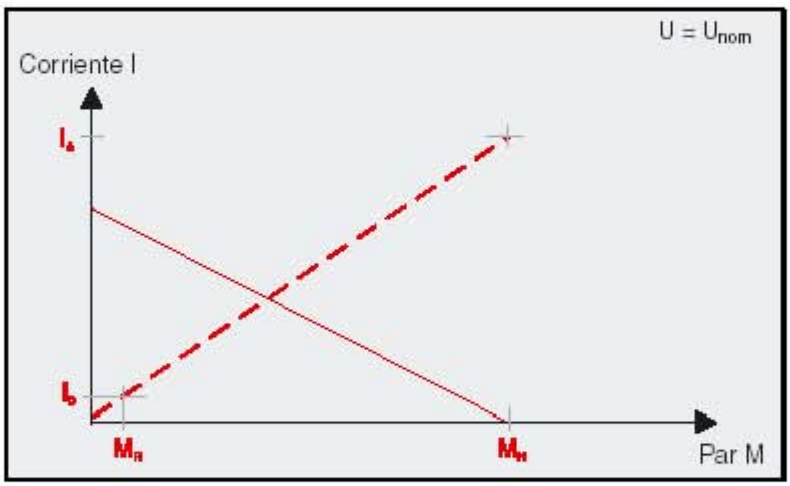


Figura 1.3.6 Curva par-corriente para un motor de corriente directa de imán permanente.[18]

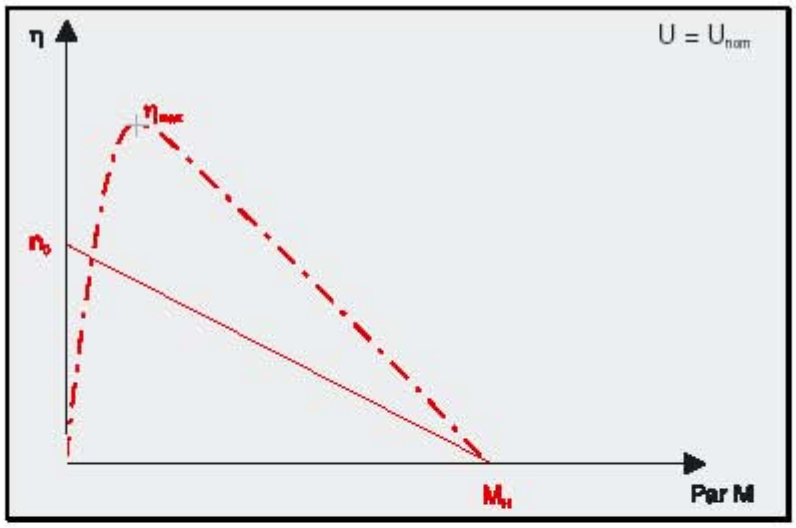


Figura 1.3.7 Curva par-eficiencia para un motor de corriente directa de imán permanente.[18]

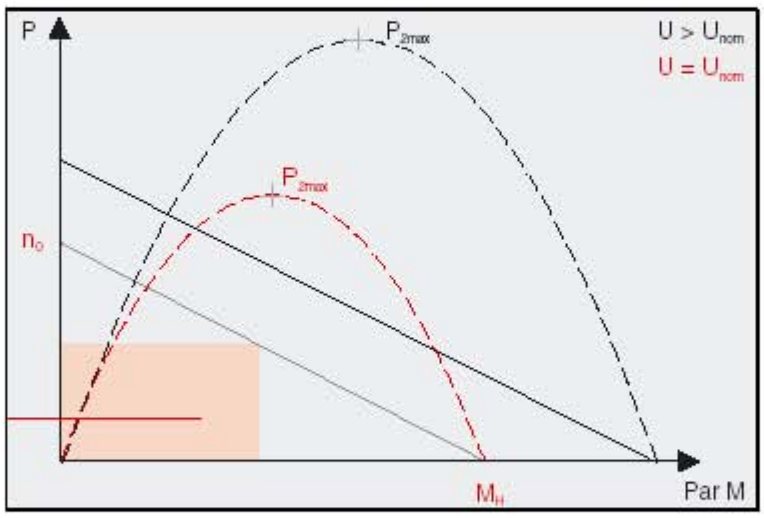


Figura 1.3.8 Curva par-potencia de salida, para un motor de corriente directa de imán permanente.[18]

Servomotores de corriente directa

Los servomotores de corriente continua son motores de alto rendimiento que por lo general se usan como motores primarios en computadoras, maquinaria controlada numéricamente u otras aplicaciones en donde el arranque y la detención se deben hacer con rapidez y exactitud. Los servomotores son de peso ligero, y tienen armaduras de baja inercia que responden con rapidez a los cambios en el voltaje de excitación. Los servomotores incluyen motores de imán permanente, circuito impreso y bobina (o coraza) móvil. El rotor de un motor acorazado consta de una coraza cilíndrica de bobinas de alambre de cobre o de aluminio. El alambre gira en un campo magnético en el espacio anular entre las piezas polares magnéticas y un núcleo estacionario de hierro. El campo es producido por imanes de fundición de Álnico cuyo eje magnético es radial. El motor puede tener dos, cuatro o seis polos. Cada uno de estos tipos básicos tiene sus propias características, como son la inercia, forma física, costos, resonancia de la flecha, configuración de ésta, velocidad y peso. Aun cuando estos motores tienen capacidades nominales similares de par, sus

constantes físicas y eléctricas varían en forma considerable. La selección de un motor puede ser tan sencilla como ajustar uno al espacio del que se disponga.

En el capitulo 2 en la figura 2.2.1 se muestra un servomotor de la marca Futaba, del tipo estándar.

1.4 MORFOLOGÍA PARA LOS ROBOTS SEGUIDORES DE LÍNEA

Existen varios tipos de morfologías para robots seguidores de línea pero solo hablaremos del tipo con ruedas aunque existen del tipo con patas. Entre las principales morfologías del tipo con ruedas se encuentran las siguientes:

Tracción omnidireccional

En este tipo de arreglo se tienen 3 o mas llantas todas ellas tienen tracción y dirección. Logra desplazarse de inmediato en cualquier dirección y su construcción es una de las más complejas. La dirección esta dada por la resultante de los vectores de dirección de cada llanta. Véase en la figura 1.4.1

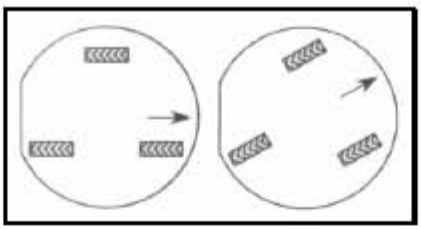


Figura 1.4.1 Morfología para robot con tracción omnidireccional.[27]

Tracción tipo triciclo

En este tipo de configuración se tienen 3 llantas como se muestra en la figura 1.4.2 en la que la llanta delantera tiene tracción y dirección y las 2 llantas traseras están sueltas.

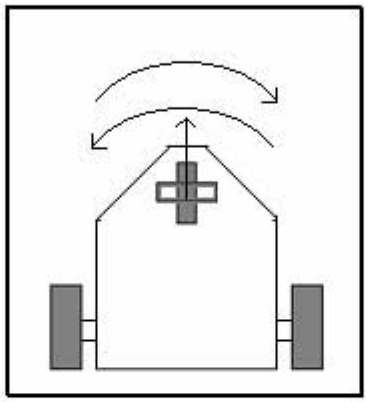


Figura 1.4.2 Tracción tipo triciclo

Tracción tipo carro

En este arreglo las 2 llantas traseras tienen tracción, y las 2 llantas delanteras se encargan de la dirección. Esta configuración es una de las mejores en estabilidad, ya que tiene cuatro puntos de apoyo, sin embargo en construcción resulta un poco difícil, ya que las 2 llantas de adelante necesitan estar fijas a un mecanismo en el cual las 2 llantas se muevan de forma paralela. En este tipo de configuración, las llantas de tracción (generalmente las traseras), les da fuerza un diferencial mecánico, para facilitar el movimiento en curvas. Como se muestra en la figura 1.4.3

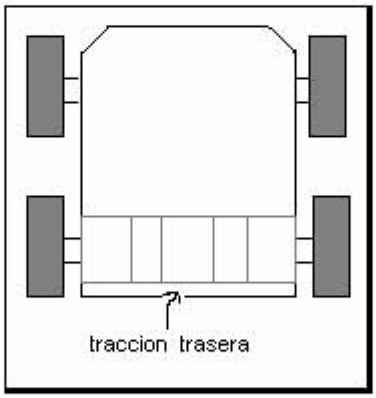


Figura 1.4.3 tracción tipo carro

Tracción diferencial

En este tipo de configuración, generalmente se acoplan dos motores con tren reductor de engranes como se muestra en la figura 1.4.4, la llanta delantera es simplemente una rueda loca y únicamente sirve de apoyo. Para dar una vuelta bruscamente solo se hace girar un motor hacia delante y el otro hacia atrás y se consigue que el robot de vueltas sobre su propio eje, este movimiento resulta muy útil cuando se tienen cruces en 90 grados, además es fácil de construir. El control para este tipo de morfología, es "sencillo" porque el mismo diseño mecánico permite una gran maniobrabilidad y estabilidad.

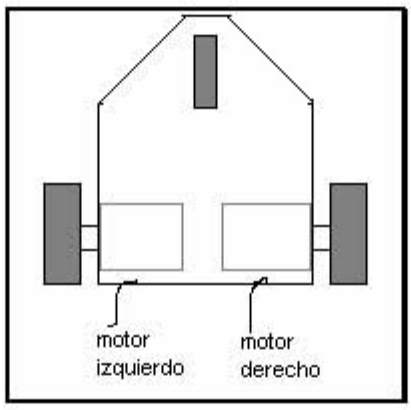


Figura 1.4.4 Tracción diferencial

1.5 MICROCONTROLADORES DE TECNOLOGÍA RISC Y CISC, ASPECTOS GENERALES (SOLO PARA ESTE TIPO DE APLICACIÓN)

Todas las marcas de microcontroladores se pueden diferenciar según el tamaño y componentes especiales, pero en general todos tienen los siguientes bloques básicos:

Entre los microcontroladores más empleados en robots móviles se encuentra el MC68HC11, y el PIC16F877 en sus diversas versiones, algunas de las características de estos son:

Microcontrolador MC68HC11F1

Características generales:

1.- Arquitectura VON NEUMANN

La **arquitectura Von Neumann** se refiere a las arquitecturas de computadoras que utilizan el mismo dispositivo de almacenamiento tanto para las instrucciones como para los datos.

Los ordenadores con arquitectura Von Neumann constan de cinco partes: La unidad aritmético-lógica o ALU, la unidad de control, la memoria, dispositivos de entrada/salida y el bus de datos que proporciona un medio de transporte de los datos entre las distintas partes.

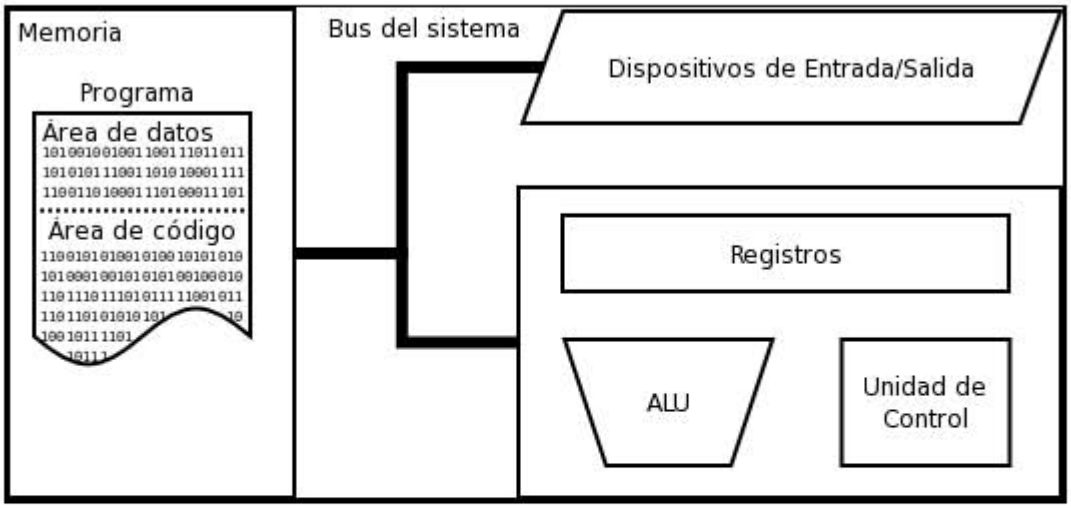


Figura 1.5.1 Arquitectura Von Neumann.[27]

2.-CPU CISC

Del ingles *Complex Instruction Set Computer*. Conjunto de microprocesadores cuyo conjunto de instrucciones se caracteriza por ser muy amplio y permitir operaciones complejas entre operandos situados en la memoria o en los registros internos, en contraposición a la arquitectura RISC.

3.-Memoria RAM de 1Kbyte.

4.-Memoria EEPROM de 512 bytes.

5.-Cuatro modos de operación

- Bootstrap
- Expandido
- Single chip
- Test

6.-Siete puertos paralelos con función alterna en la mayoría de ellos.

- Puerto A de 8 bits bidireccional
- Puerto B de 8 bits de salida
- Puerto C de 8 bits bidireccional
- Puerto D de 6 bits bidireccional
- Puerto E de 8 bits de entrada
- Puerto F de 8 bits de salida
- Puerto G de 8 bits de salida

7.-Convertidor analógico/digital, 8 canales de 8 bits de resolución

8.-Puerto serie asíncrono (SCI)

9.-Puerto serie síncrono (SPI)

10.-Sistema temporizador

- Funciones de acumulación de tiempos y de pulsos
- Funciones de interrupción en tiempo real
- Funciones de captura de entrada
- Funciones de comparación de salida
- Funciones de sobre flujo del temporizador

11.- 68 pines

12.- 8 registros de propósito general

- Acumulador A
- Acumulador B
- Doble acumulador D
- Registro de índice X
- Registro de índice Y
- Stack Pointer SP
- Program counter PC
- Registro de banderas CCR

Para una aplicación donde se requiera mayor memoria a la descrita, será necesario expandir la capacidad de almacenamiento, empleando los puertos B, C y F para formar los buses de datos y direcciones, además de circuitos de decodificación, lo que hace un módulo relativamente grande.

Microcontrolador PIC16f877

Características generales:

1.-Arquitectura HARVARD

El término **Arquitectura Harvard** originalmente se refería a las arquitecturas de computadoras que utilizan dispositivos de almacenamiento físicamente

separados para las instrucciones y para los datos (en oposición a la Arquitectura von Neumann).

2.- CPU RISC

En arquitectura computacional, **RISC** del inglés *Reduced Instruction Set Computer* (Computadora con Conjunto de Instrucciones Reducido). Tipo de microprocesadores con las siguientes características fundamentales:

1. Instrucciones de tamaño fijo, presentadas en un reducido número de formatos.
2. Sólo las instrucciones de carga y almacenamiento acceden a memoria a por datos.

3.-Memoria RAM de 368x8byte
4.-Memoria EEPROM de 256x8bytes
5.-Memoria FLASH de 8Kx14bytes
6.-Un solo modo de operación
7.-Cinco puertos paralelos con función alterna en la mayoría de ellos
- Puerto A de 6 bits bidireccional
- Puerto B de 8 bits bidireccional
- Puerto C de 8 bits bidireccional
- Puerto D de 8 bits bidireccional
- Puerto E de 3 bits bidireccional

8.-Convertidor analógico/digital, 8 canales de 8 o 10 bits de resolución
9.-Puerto serie asíncrono (SCI/USART)
10.-Puerto serie síncrono (SSP)
11.-Puerto serie paralelo (PSP)
12.-Protocolo I2C
13.-Sistema temporizador (CCP)
- Módulo de captura
- Módulo de comparación
- Módulo PWM (principalmente usado en control de motores)

14.-40 pines
15.-3 registros de propósito general

- Registro W
- Program counter
- Stack pointer

16.- Modos de direccionamiento directo, indirecto y relativo.[20]

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

ROBOT RESOLVEDOR DE LABERINTOS

T E S I S

que para obtener el título de

INGENIERO ELECTRICO ELECTRONICO

P R E S E N T A

FELIPE DE JESUS PERALTA ZARATE

DIRECTOR DE TESIS:
M.I. JUAN CARLOS ROA BEIZA

Ciudad Universitaria 2006

.

INTRODUCCIÓN

La construcción de laberintos es tan antigua que existen registros de laberintos que fueron construidos en el año 3000 AC. Sin embargo, el más famoso es un laberinto griego, del que no se sabe en realidad si existió o no. La leyenda cuenta que estaba ubicado en Creta y que en el habitaba "el minotauro", una bestia mitad hombre y mitad toro que se alimentaba de carne humana.

Muchos siglos después, durante la edad media en Europa se construyeron muchos laberintos que servían para llevar a cabo distintos tipos de rituales y de procesiones; por ejemplo, en la península escandinava existen alrededor de 600 laberintos de piedra construidos en las orillas del mar Báltico. Se dice que fueron construidos por los pescadores que los usaban para hacer paseos por ellos antes de salir al mar a pescar, con ello los malos espíritus se quedaban confundidos en el laberinto y los pescadores salían seguros.

En el siglo XIII, en Francia, era común hacer dibujos de laberintos en el suelo de las catedrales y un siglo más tarde en toda Europa se construyeron cientos de laberintos con arbustos en los jardines de los castillos y palacios para que los nobles, reyes y príncipes se entretuvieran paseando por ellos. Esta costumbre de adornar los parques y jardines con laberintos se mantuvo hasta el siglo XX. Fue entonces que, como consecuencia de las dos guerras mundiales, prácticamente todos los laberintos de Europa desaparecieron. Pero a partir de la década de los setenta en todo el mundo se han vuelto a construir laberintos. El siglo XX ha sido el siglo en el que más se han construido y diseñado, se pueden encontrar en jardines, parques de diversiones, en libros y ahora, por supuesto, en Internet.

Hoy en día con los avances tecnológicos en sensores, microcontroladores, motores, procesadores, y en general en electrónica se ha logrado un gran avance en el desarrollo de los robots móviles. Existen a nivel nacional competencias de robots móviles organizados por distintas universidades. Es muy común encontrar en los concursos de robótica la categoría de robot de laberinto, en la cual se diseñan robots móviles capaces de descifrar el laberinto.

No obstante este tipo de robots también tienen otro tipo de aplicaciones en la industria como: por ejemplo para buscar y traer piezas de almacén y llevarlas al mostrador; también son muy útiles para simular el modo de discriminación que ocupa el cerebro humano en ciertas rutinas.

OBJETIVOS

Diseñar y construir un robot que sea capaz de descifrar cualquier laberinto en su primera pasada y que una vez descifrado, pueda ejecutarlo sin equívocos de principio a fin.

El robot será desarrollado con sistemas de alta tecnología que permitan programar las rutas de la primera pasada del robot, y después a través de un algoritmo de discriminación, el robot deberá ser capaz de evitar o eludir los caminos falsos, y en su segundo intento deberá encontrar el camino correcto sin titubeos.

Se elegirá de entre una gama de microcontroladores, tanto de tecnología CISC como RISC para elegir el mejor y de esta forma proceder a programarlo con los algoritmos necesarios para que ejecute las tareas asignadas.

El robot deberá contar con una serie de sensores del tipo reflectivo que le permitan reconocer las diferentes pistas y los cruces que estas contengan.

Las pistas serán líneas negras sobre una superficie blanca que es el estándar internacional para este tipo de robots.

Se diseñara y construirá el prototipo mecánico que permita tener una buena eficiencia en este tipo de problemas

TEORÍA BÁSICA

2.1 CARACTERÍSTICAS, VENTAJAS Y DESVENTAJAS DE LOS SENSORES ÓPTICOS.

La **radiación infrarroja** o **radiación térmica** es un tipo de radiación electromagnética de mayor longitud de onda que la luz visible, pero menor que la de las microondas. Consecuentemente, tiene menor frecuencia que la luz visible y mayor que las microondas. [4]

El nombre de infrarrojo, que significa por debajo del rojo, proviene de que fue observada por primera vez al dividir la luz solar en diferentes colores por medio de un prisma que separaba la luz en su espectro verticalmente de manera que el rojo era el que estaba mas abajo y el violeta el mas arriba. William Herschel observó en el año 1800 que se recibía radiación debajo del rojo al situar termómetros en las diferentes zonas irradiadas por el espectro.[4]

En la actualidad existen fuentes de luz infrarroja las cuales son usadas en diferentes aplicaciones como la transmisión de datos (en los televisores), sensores de temperatura, alarmas, pero la que nos interesa es en sensores ópticos.

Actualmente existen en el mercado diferentes tipos de sensores infrarrojos, incluso las partes para construirlos, como son: el diodo led infrarrojo como el que se muestra en la figura 2.1.1

Figura 2.1.1led infrarrojo IR333

Las características de este diodo son las siguientes: ángulo para alta intensidad luminosa de 20° y un nivel de iluminación de 20 mW/sr el arco luminoso aproximado se muestra en la figura 2.1.2, es claro que la luz infrarroja no es visible para la vista humana, por eso este tipo de pruebas se hacen con cámaras digitales u otro tipo de dispositivo.

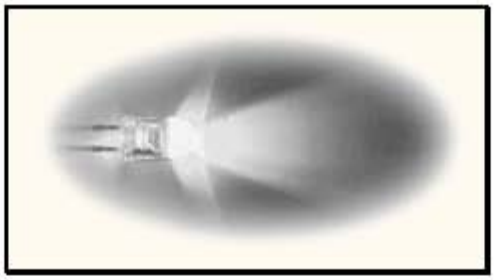


Figura 2.1.2 muestra el arco luminoso aproximado

Existe en el mercado otro tipo de led infrarrojos que son los llamados de larga distancia, entre los cuales se encuentra el IR383 le cual se muestra en la figura 2.1.3

Figura 2.1.3 led infrarrojo IR383

Las características de este diodo son: un ángulo para alta intensidad luminosa de 12°, un nivel de iluminación de E=26 mW/sr. En la figura 2.1.4 se muestra el arco luminoso aproximado de este tipo de led.

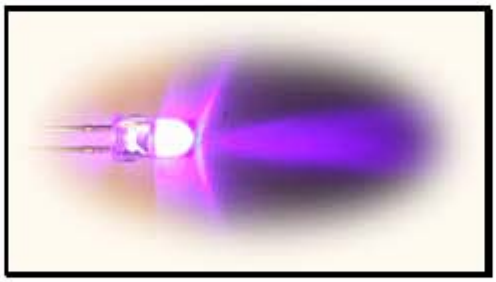


Figura. 2.1.4 Arco luminoso aproximado

En este caso veremos el uso de estos leds infrarrojos en la construcción de un sensor detector de línea blanca sobre una superficie de color negro. Para esto necesitamos un detector de luz infrarroja como un fototransistor. Existen en el mercado varios tipos de fototransistores, como el PT331C el cual se muestra en la figura 2.1.5, este tiene un aspecto de un diodo led emisor infrarrojo. Sin embargo su comportamiento es el de un transistor en el cual la base se polariza con luz infrarroja, por lo general en este tipo de fototransistores la pata mas corta es el colector y la larga el emisor.

Figura 2.1.5 fototransistor infrarrojo PT331

Sin embargo para aplicaciones en robots móviles es recomendable utilizar el receptor con filtro de luz de día, como el PT1302B. Este tipo de

fototransistores tienen un filtro de luz de día o blanca con lo que responden únicamente a estímulos con luz infrarroja.

Figura 2.1.6 fototransistor infrarrojo PT1302B

Para construir un sensor detector de línea blanca podemos utilizar un led infrarrojo como el IR383 de alta distancia y un fototransistor infrarrojo con filtro de luz como el PT1302B, en la figura 2.1.7 se muestra la forma de acomodar el emisor y el receptor infrarrojo, de tal forma que se el receptor detecte la mayor cantidad de luz posible cuando la superficie es de color blanco.

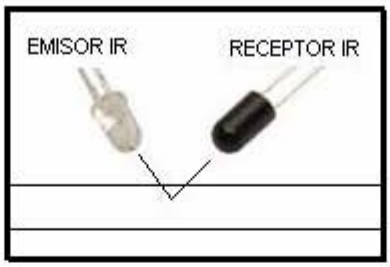


Figura 2.1.7 PT1302B e IR383

Existen varios tipos de circuitos para construcción de detectores de línea blanca, que van desde transistores hasta amplificadores operacionales, en la figura 2.1.8 se muestra el circuito para la construcción de un sensor detector de línea blanca, es uno de los circuitos mas simples. Se compone de dos partes, la primera es la del emisor infrarrojo en la cual el IR383 se conecta en serie con una resistencia de 330 ohms, y a la vez a la fuente de alimentación que

puede ser de 5V. La segunda etapa es la de receptor y acondicionamiento de señal, el amplificador operacional compara la tensión del emisor del transistor con la tensión de un potenciómetro que nos sirve para calibrar nuestro detector, el fototransistor se conecta en configuración emisor común. Cuando la tensión en la Terminal no inversora es mayor a la de la Terminal inversora la salida del AO será la tensión de alimentación, y cuando la tensión en la terminal inversora sea mayor que en la terminal no inversora, la salida del AO será cero volts, por lo tanto, cuando se detecte la línea blanca LED1 encenderá. Este circuito tiene grandes aplicaciones en robots rastreadores, seguidores de línea blanca.

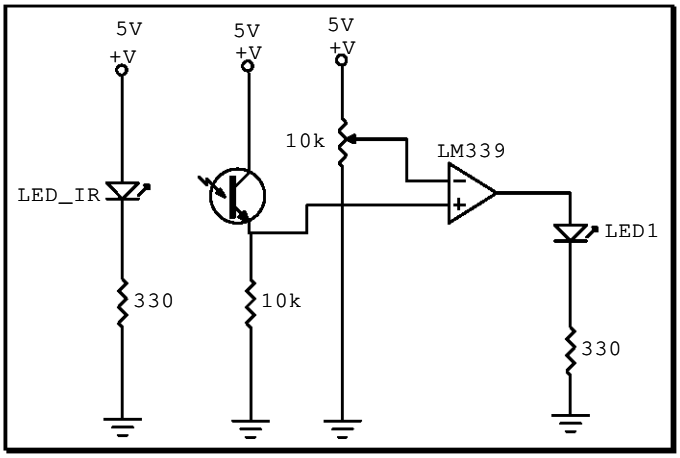


Figura 2.1.8 Circuito para construcción de un sensor detector de línea blanca sobre una superficie de color negro.

Es importante saber que existen en el mercado emisores y receptores infrarrojos ya dispuestos en un ángulo optimo enfocando la máxima respuesta como lo es el HOA1404, mostrado en la figura 2.1.9, que consiste en un diodo emisor infrarrojo y un fototransistor de silicón tipo NPN

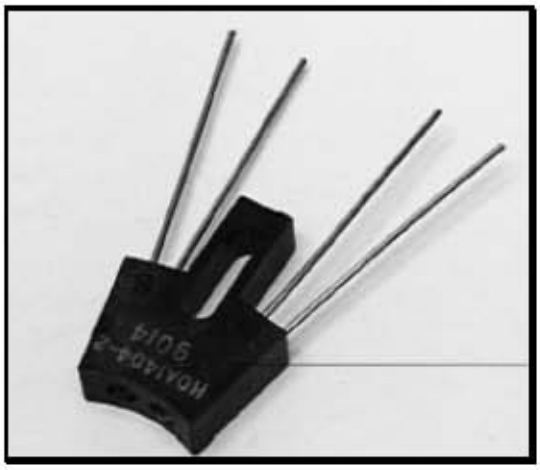


Figura 2.1.9 sensor reflectivo HOA1404-2[25]

Para aplicaciones mas finas o en las que el se desea detectar líneas mas delgadas podemos encontrar en el mercado sensores mas pequeños como el QRD1114 es muy similar a el HOA1404 pero el tamaño es mucho menor. Tanto el emisor como el receptor están encapsulados en un empaque de 6.10 x 4.39 x 4.65 mm.

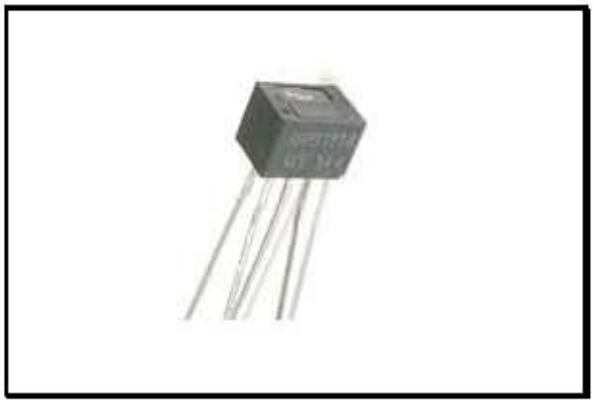


Figura 2.1.9 sensor reflectivo QRD1114[24]

Este tipo de sensores es de los mas utilizados en construcción de encoders y para seguir líneas muy delgadas, en nuestro caso la línea mide solo 6 mm de espesor por tanto este es el sensor ideal para nuestra aplicación.

2.2 EVALUACIÓN DE DIFERENTES ESTRUCTURAS MECÁNICAS Y ELECCIÓN DE LA ÓPTIMA.

Este es uno de los temas de gran importancia, ya que, dependiendo de la estructura mecánica que vayamos a emplear, va a afectar toda la parte de control, desde la cantidad de baterías y la capacidad de las mismas, capacidad del driver que tendremos que usar, la forma de acomodar los sensores hasta el algoritmo de control.

Para este capitulo, tendremos que plantearnos objetivos, en cuanto a las características de nuestro prototipo, ya que si nosotros mismos vamos a diseñar el sistema mecánico, lo diseñaremos de tal forma que nos facilite la forma de controlarlo, de las morfologías vistas en el capitulo 1, escogeremos la morfología de tracción diferencial, ya que con esta nos permite dar vueltas en cruces de 90° con solo activar un motor hacia delante y uno hacia atrás ya que en el laberinto nos tendremos que enfrentar a este tipo de cruces.

Un factor muy importante para nuestro diseño mecánico, es lo que conocemos como **inercia**, es la tendencia de los cuerpos a mantener su estado de movimiento, el cual no se modifica a menos que actúen fuerzas externas sobre su masa.

Se sabe que P=mv , de donde P = Momentum Lineal, m = masa y v = velocidad. Por lo tanto, a mayor velocidad, mayor inercia y a mayor masa mayor inercia. Y para generar un cambio en la cantidad de movimiento basta con integrar la expresión anterior y obtendremos F= m a, de donde F= fuerza, a= aceleración, este efecto, lo podemos percibir en el movimiento de un auto al momento de arrancar o frenar. También existe otro tipo de inercia llamada

inercia rotacional en el cual lo podemos percibir en el arranque de un motor eléctrico. Ambas inercias nos afectan bastante en el momento de detener el robot, en el momento de empezar a mover y en el momento de girar, sin embargo como este efecto depende de la masa del robot, el objetivo entonces será disminuir al máximo la masa de nuestro prototipo. Disminuir la masa, implica un menor consumo de corriente, también disminuye la corriente transitoria en el momento de arrancar, por consiguiente las baterías pueden ser mas pequeñas así como nuestro driver que va a alimentar los motores de tracción será de menor capacidad de corriente.

Por tanto la primera prueba la haremos construyendo un chasis de aluminio con 2 servomotores Futaba como el que se muestra en la figura 2.2.1

Figura 2.2.1 servomotor de Futaba estándar.

Este tipo de motores se puede conseguir en tiendas de aeromodelismo, son muy utilizados modelismo de aviones y modelismo de autos también. En este caso tendremos que hacer una pequeña modificación para utilizarlo en nuestro prototipo, estos servomotores traen un conector de 3 cables, normalmente el rojo es 5v el negro es tierra y el blanco es una entrada de señal ya que estos

motores trabajan con PWM o sea con modulación por ancho de pulso, en este tipo de motores la salida es una posición de la flecha, y solo giran hasta 180 grados. La modificación consiste en eliminar un potenciómetro que se encuentra bajo la flecha y un tope mecánico que también se encuentra en la flecha, así como la etapa de control de posición, se desconecta el control y se soldan unos alambres directo a el motor de DC. Quedando un motor con tren de engranes. Necesitamos 2 motores como este ultimo, modificado para construir nuestro diferencial mecánico como se muestra en la figura 2.2.3, cada servomotor tiene acoplada una llanta lego de 8.16 cm. de diámetro con rines de plástico y llanta de goma, en la parte delantera se encuentra una llanta loca, la cual nos sirve únicamente de apoyo ya que la dirección nos la proporcionan las llantas de atrás.

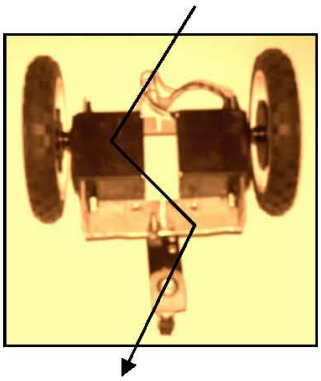


Figura 2.2.3 Nuestra primera estructura mecánica. Estructura 1

Para empezar con la prueba, lo primero que tenemos que hacer es ver la estabilidad de la estructura, para esto necesitamos una pista de pruebas, y un circuito de pruebas, en este caso nuestro circuito será simplemente un seguidor de línea, como los vistos en el tema 1, específicamente el de la figura 1.2.2, y

la pista de pruebas será una simplemente una línea recta de color negro, dibujada sobre una superficie de color blanco.

Este circuito lo armaremos en un impreso de prueba, en el capitulo 3 se explicara mas acerca de el armado de toda la parte electrónica.

Después de colocar el circuito de prueba, y hacer el primer experimento, se noto que estos motores son muy lentos y la estructura oscilaba bastante,

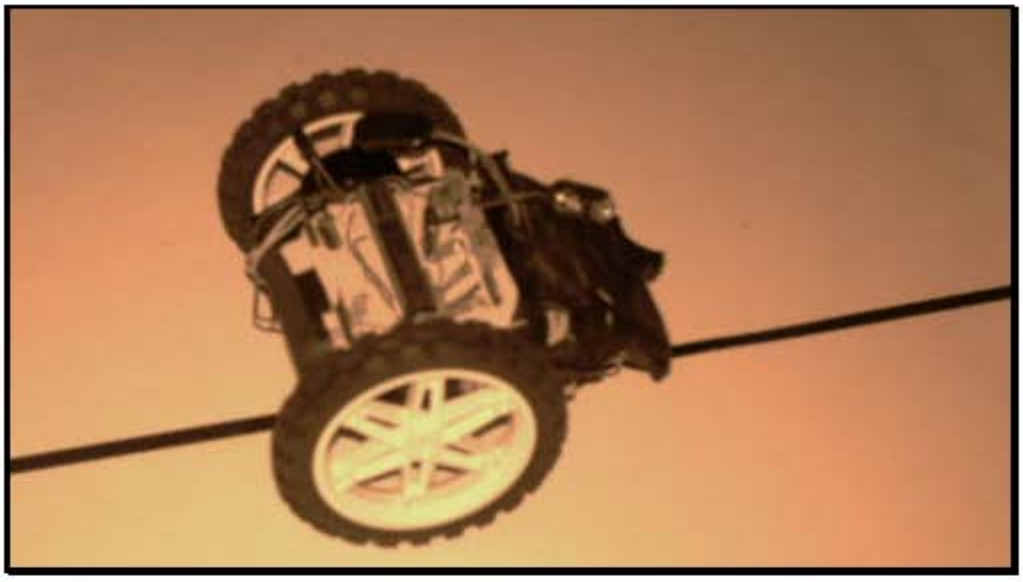


Figura 2.2.4 Estructura 1 con circuito de prueba y baterías.

Una posible solución para disminuir la oscilación fue mover el peso de las pilas hacia el eje de rotación, esto se hizo colocando una base para las pilas sobre el circuito de prueba y quedando exactamente la mayor parte del peso sobre los motores.

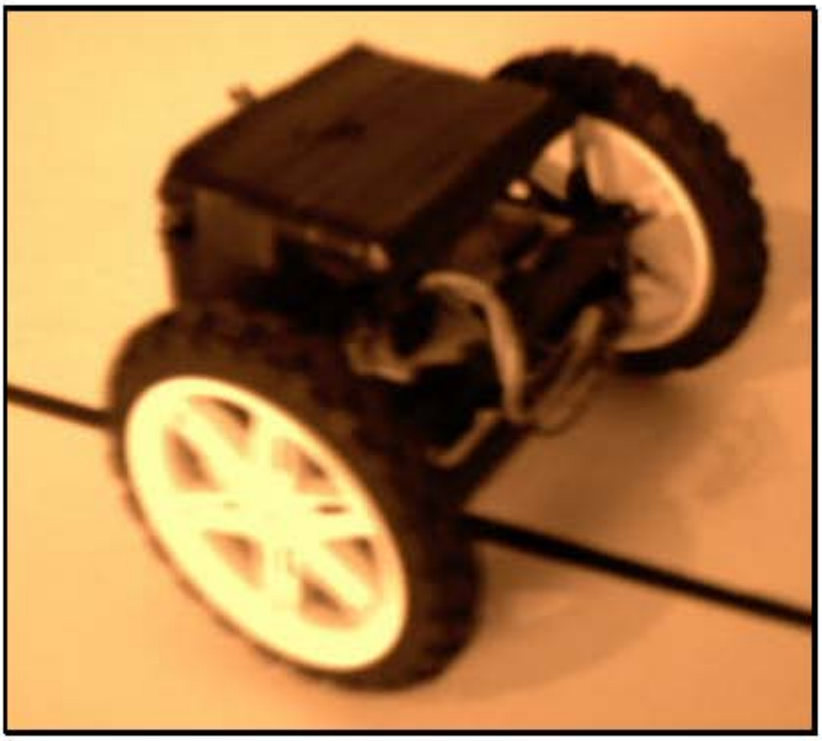


Figura 2.2.5 Estructura 2 las baterías ahora se encuentran arriba del circuito

La estructura de la figura 2.2.5 demostró ser más estable que la Estructura 1, la oscilación disminuyo considerablemente y por tanto aumento la velocidad.

Nuestra tercer estructura mecánica será una chasis de un material llamado SINTRA este puede conseguirse en tiendas de robótica como www.robodacta.com este material se caracteriza por ser un material plástico, ligero y rígido, se puede cortar con navaja, perforar, taladrar, pintar, pegar con silicón, Kola Loca, etc. El SINTRA es un material termo deformable, es decir, se puede doblar al sumergirlo en agua caliente y mantener esa forma al enfriarse. Usaremos un pedazo de 8.7 cm. x 9 cm. y utilizaremos 2 motores mas pequeños, pero mas rápidos, aunque con menor fuerza, sin embargo consumen menos corriente. Sin embargo los motores que se elijan los analizaremos en el siguiente subcapitulo.

Nuestra tercer estructura queda como le mostrada en la figura 2.2.6

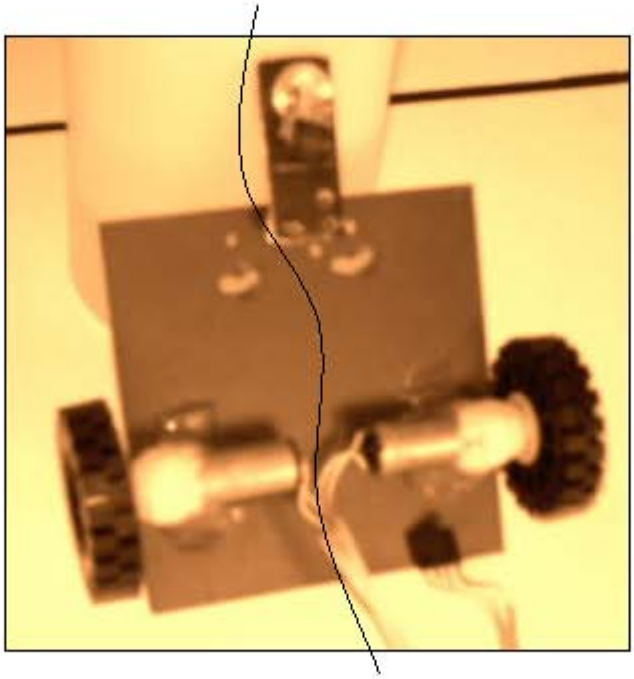


Figura 2.2.6 Estructura 3 con chasis de SINTRA llantas lego de 4.4 cm. de diámetro.

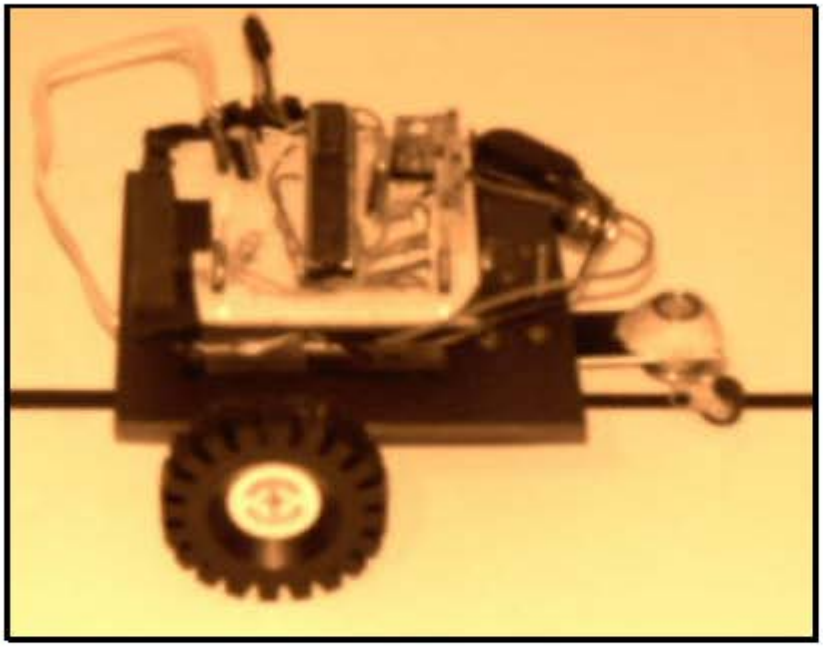


Figura 2.2.7 Estructura 3 con batería y circuito de prueba

La estructura 3 resulto ser la mas estable, es decir oscila menos y debido a los motores es mucho más rápida, aparte de que arranca más rápido, debido a que es más ligera. Por lo tanto esta será nuestra estructura mecánica.

2.3 CARACTERÍSTICAS, VENTAJAS Y DESVENTAJAS DE LOS MOTORES DE CORRIENTE DIRECTA ELEGIDOS.

Estos motores fueron elegidos en el subcapitulo anterior, debido a sus características, en esta parte daremos las especificaciones un poco mas detalladas. En la figura 2.2.4 se muestra la foto de los motores elegidos, traen un tren de engranes reductor, el cual esta en la parte frontal.

Figura 2.3.1 Motores modelo QJT4.8 [9]

Las medidas se pueden ver en la figura 2.3.2

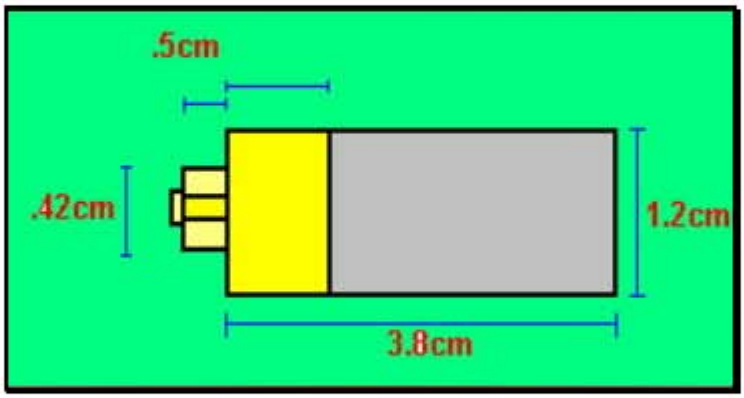


Figura 2.3.2 Medidas del motor QJT4.8 [10]

La corriente nominal de este motor es de 433 mA y a 4.8 V, entrega un torque de 917 g.cm. a una velocidad de 80 RPM. El consumo de corriente sin carga es de 136mA a una velocidad de 105 RPM.

El peso total del motor con su tren de engranes es de 70 gramos. Es un excelente motor, sin embargo el tren de engranes tuvo que ser reforzado con plastilina epoxica, para evitar que se desprendiera. Este tipo de motores absorbe una corriente máxima a rotor bloqueado de 1 A. esta información será de gran ayuda en la elección del driver.

2.4 SELECCIÓN DEL TIPO DE BATERÍA ADECUADA.

Una batería es un dispositivo electroquímico el cual almacena energía en forma química. Cuando se conecta a un circuito eléctrico, la energía química se transforma en energía eléctrica. Todas las baterías son similares en su construcción y están compuestas por un número de celdas electroquímicas. Cada una de estas celdas está compuesta de un electrodo positivo y otro negativo además de un separador. Cuando la batería se está descargando un cambio electroquímico se está produciendo entre los diferentes materiales en los dos electrodos. Los electrones son transportados entre el electrodo positivo y negativo vía un circuito externo (bombillas, motores de arranque etc.). [5]

Es necesario distinguir entre baterías recargables (acumuladores) y pilas o baterías desechables. La diferencia fundamental entre ambos tipos está en que las baterías recargables permiten revertir la reacción química en la que está basado su funcionamiento, mientras que las desechables no. En este capitulo estudiaremos las características de las baterías recargables, ya que económicamente representan un ahorro debido a sus propiedades de recarga.

Baterías de plomo, Las primeras baterías de plomo-ácido (acumuladores de plomo), fueron fabricadas a mediados del siglo XIX por Gaston Planté. Hoy en día todavía son uno de los tipos de baterías más comunes. Se descubrió que cuando el material de plomo se sumergía en una solución de ácido sulfúrico se producía un voltaje eléctrico el cual podía ser recargado. [5]

Este tipo de baterías es único en cuanto que utiliza el plomo, material relativamente barato, tanto para la placa positiva como para la negativa. El material activo de la placa positiva es óxido de plomo (Pbo_2). El de la placa negativa es plomo puro esponjoso y el electrolito está disuelto en (H_2SO_4).

Cuando hablamos de material activo en las baterías de ácido-plomo, nos referimos al óxido de plomo y al plomo esponjoso. Las baterías de plomo reinan en nuestros automóviles pero sólo destinadas a cubrir las necesidades de arranque, iluminación e ignición (no tienen suficiente energía para mover el coche). En estas baterías la vida útil será mayor cuanto menor sea la descarga en cada ciclo de carga-descarga. [5]

Baterías Alcalinas: No son muy comunes las baterías alcalinas recargables. La mejor característica que tienen es que aportan una tensión de 1.5 voltios. Sin embargo son difíciles de encontrar en el mercado y requieren un cargador propio también poco común.

Níquel-Cadmio: Son las más habituales. Proporcionan tensiones de 1.2 voltios. Contienen cadmio, un metal pesado que representa un peligro ecológico. Interiormente tienen dos electrodos, el de cadmio (negativo) y el de hidróxido de níquel (positivo), separados entre sí por un electrolito de hidróxido de potasa. Llevan también un separador situado entre el electrodo positivo y la envoltura exterior y un aislante que las cierra herméticamente como se muestra en la figura 1.4.1.[5]

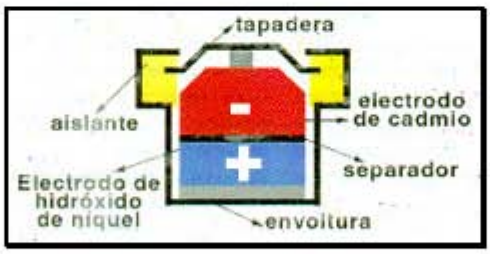


Figura 1.4.1 partes de una batería de Níquel. [5]

Su aspecto más positivo es el precio. Aunque parezcan caras por el número de veces que se pueden recargar en comparación con una batería no-recargable resulta económico el precio. Sin embargo tienen una característica llamada "efecto memoria". Significa que antes de recargarlas es necesario haberlas agotado completamente ya que en caso contrario su vida se va acortando. Una solución es, cuando se vea que empiezan a perder energía, dejar el equipo encendido hasta que se agoten completamente. Además son contaminantes.

Níquel-Metal hidruro: No tienen metales pesados como el cadmio y por eso son menos perjudiciales para el medio ambiente.

Además de ser menos contaminantes proporcionan tensiones de 1.3 voltios y tienen una capacidad mucho mayor por lo tanto duran más que las de Níquel-Cadmio y dan más energía. No tienen efecto memoria, de modo que se pueden recargar aunque no se hayan agotado al completo. Las mejores llegan a soportar hasta 1.000 procesos de carga.

Litio-Ion: las ***baterías de litio***, junto quizá a las de hidruro metálico son las que van encontrando un mayor consenso en cuanto a su potencial y un mayor esfuerzo en su investigación y desarrollo a nivel mundial.

Son muchas las razones que han originado este consenso. En primer lugar ***el litio es el metal más ligero*** y esto da lugar a una alta capacidad específica (Figura 2.4.2), lo que permite obtener la misma energía con un peso muy inferior (Figura 2.4.3). [6]

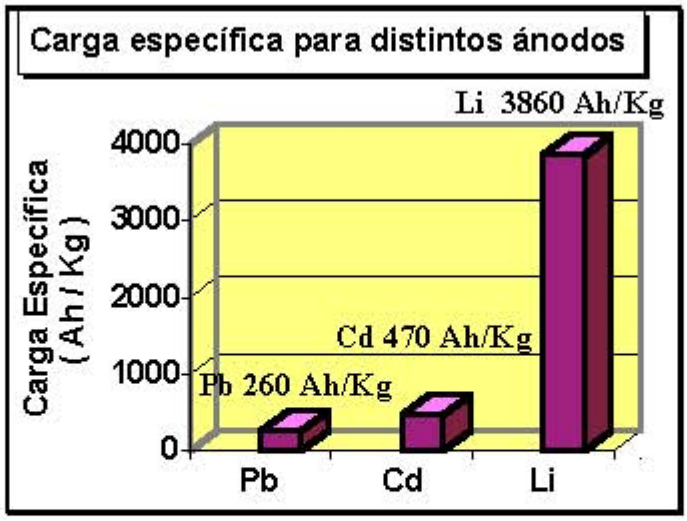


Figura 2.4.2 Carga especifica para distintos ánodos. [6]

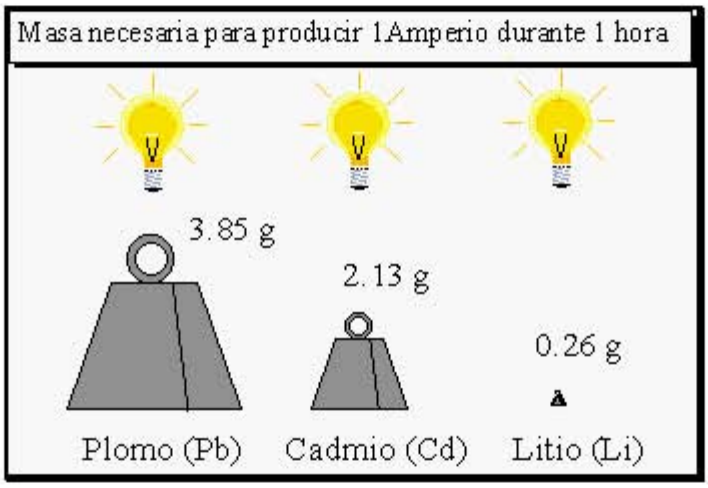


Figura 2.4.3 Masa necesaria para producir un Amper durante 1 hora. [6]

Sin embargo, Estas baterías tienen un gasto de producción elevado y muy costoso que se refleja en su precio final. Su ciclo de vida se sitúa entre los 500-600 ciclos de carga/descarga. Sin embargo ofrece una capacidad equivalente y más fiable dando una densidad de energía más elevada y constante que las baterías de Ni/Cd o Ni/MH. Tiene muy poca auto descarga 6% al mes y la tensión de trabajo típica es de 3.6. [6]

Método de carga

El método de carga es relativamente sencillo, pero debe ser **muy preciso**. En baterías con ánodo de grafito la tensión de final de carga es de 4.1V y en baterías con ánodo de carbón la tensión de final de carga es de 4.2V, por tanto, al llegar a esa tensión la carga debe pararse inmediatamente. La precisión **exigida es del 1%**. Si la sobrepasamos podemos acortar el ciclo de vida de la batería. Y si no se llega a esa tensión la carga no será completa. El método de carga más usado es el conocido como "corriente constante - tensión constante". Consiste en empezar la carga con una corriente constante. Cuando se aproxima al final, el último empujoncito se le da con una tensión constante. Hay otros métodos más simples pero quizás más lentos. Lo importante es parar la carga cuando se alcanza el límite. Estas baterías tienen un rendimiento energético muy bueno durante la carga: casi toda la energía que reciben se usa para cargar la batería, con muy poco o nulo calentamiento. [7]

Problemas de estas baterías

Existe un problema (y además, poco documentado): la **pasivación**. No confundir este fenómeno con el llamado "efecto memoria" de las baterías de Ni-Cd.[7]

La pasivación consiste en la formación de una película de cloruro de litio (LiCl) en la superficie del ánodo. De algún modo sirve para evitar la auto descarga, cuando la batería no está siendo usada. Esta delgada película es, funcionalmente, una resistencia. Pero está claro que puede producir una caída de tensión o "retraso" en la entrega de energía tal como se ve en esta figura 2.4.4

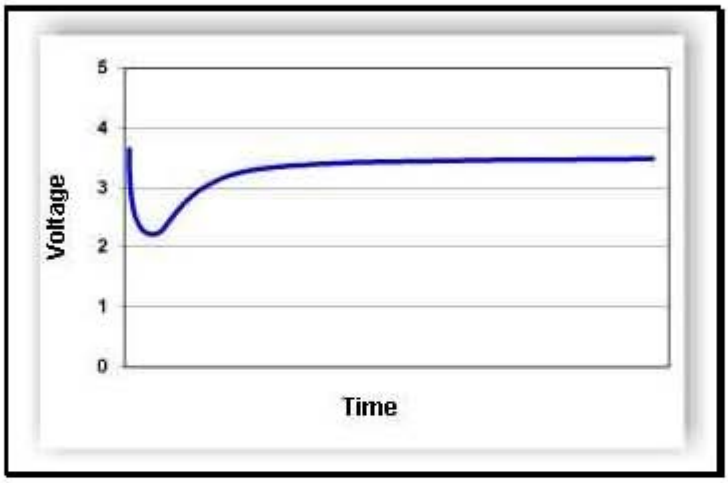


Figura 2.4.4 forma en la que entrega la energía una batería de litio. [7]

Conforme la batería va siendo usada, esta fina película va desapareciendo. El problema está en que la caída de tensión puede ser lo suficientemente abrupta como para que nuestro equipo se apague. Cuanto mayor sea la energía requerida al principio, más acusado puede ser el problema. Cuando dejamos de usar la batería, la película vuelve a irse formando.

Este fenómeno depende del diseño y constitución de la batería, tiempo sin usar (cuanto mayor sea este tiempo, más gruesa será la capa de LiCl). En cuanto a la temperatura de almacenamiento, a mayor temperatura, mayor pasivación sin embargo cuando la temperatura de uso es baja, este efecto será más visible. No obstante esta es una de las mejores baterías para cualquier prototipo en el que el reducido peso sea una prioridad. Por lo tanto este tipo de baterías se acopla a nuestras necesidades de peso, tamaño, costo y durabilidad.

2.5 CARACTERÍSTICAS, VENTAJAS Y DESVENTAJAS DEL MICROCONTROLADOR ELEGIDO (PIC16F877).

Bajo la denominación de PIC16F87x se hace referencia a una subfamilia de microcontroladores PIC de la gama media, que se identifica por tener como memoria de programa una del tipo FLASH y una serie de recursos semejante a los modelos mas potentes, como por ejemplo los PIC16C73/4, teniendo estos últimos el inconveniente de que su memoria de programa es de tipo EPROM.

Dos de los cuatro modelos que componen esta subfamilia están encapsulados con 28 patitas (PIC16F873/6), mientras que los otros dos tienen 40 patitas (PIC16F874/7). Con la intención de seguir potenciando la línea con memoria FLASH.

A continuación se muestran las características relevantes del PIC16F877:

1.- Disponen de 5 puertos PA de 6 bits, PB de 8 bits, PC de 8 bits, PD de 8 bits, PD de 8 bits y PE de 3 bits E/S con un total de 33 líneas para conectar los periféricos exteriores.

2.- Canal A/D con 8 canales de entrada

3.-Puerto paralelo esclavo

Los recursos fundamentales se enuncian a continuación:

1.- Procesador de arquitectura RISC avanzada con arquitectura HARVARD

La arquitectura Harvard dispone de dos memorias independientes una, que

contiene sólo instrucciones y otra, sólo datos. Ambas disponen de sus respectivos sistemas de buses de acceso y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias.[20]

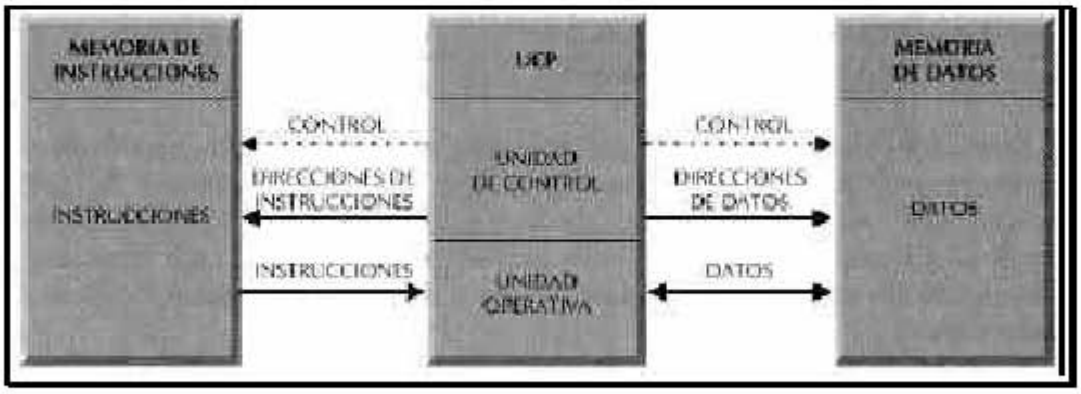


Figura 2.5.1 buses para instrucciones y datos independientes.[20]

2.- Juego de 35 instrucciones con 14 bits de longitud. Todas ellas se ejecutan en un ciclo de instrucción, menos las de salto que tardan dos.

3.- Frecuencia de operación hasta de 20 MHz

4.- Hasta 8K palabras de 14 bits para memoria de código, tipo FLASH.

Es una memoria no volátil, de bajo consumo, que se puede escribir y borrar. Funciona como una ROM y una RAM pero consume menos y es más pequeña. A diferencia de la ROM, la memoria FLASH es programable en el circuito. Esta memoria esta dividida en páginas de 2K palabras y esta direccionada con el PC que tiene un tamaño de 13 bits. [20]

5.-Hasta 368 bytes de memoria de datos RAM (Memoria de acceso aleatorio). Se utiliza para guardar los datos temporales que se necesitan en la ejecución del programa. En esta sección se alojan los registros operativos fundamentales en el funcionamiento del procesador y en el manejo de todos sus periféricos,

además de registros que el programador puede usar para información de trabajo propia de la aplicación.[20]

La RAM estática consta de 4 bancos con 128 bytes cada uno. En las posiciones iniciales de cada banco se ubican los registros específicos que gobiernan al procesador y sus recursos. En la figura 2.5.2 se muestran los 4 bancos de la RAM indicando en las primeras posiciones de cada uno los nombres de los registros que contienen.

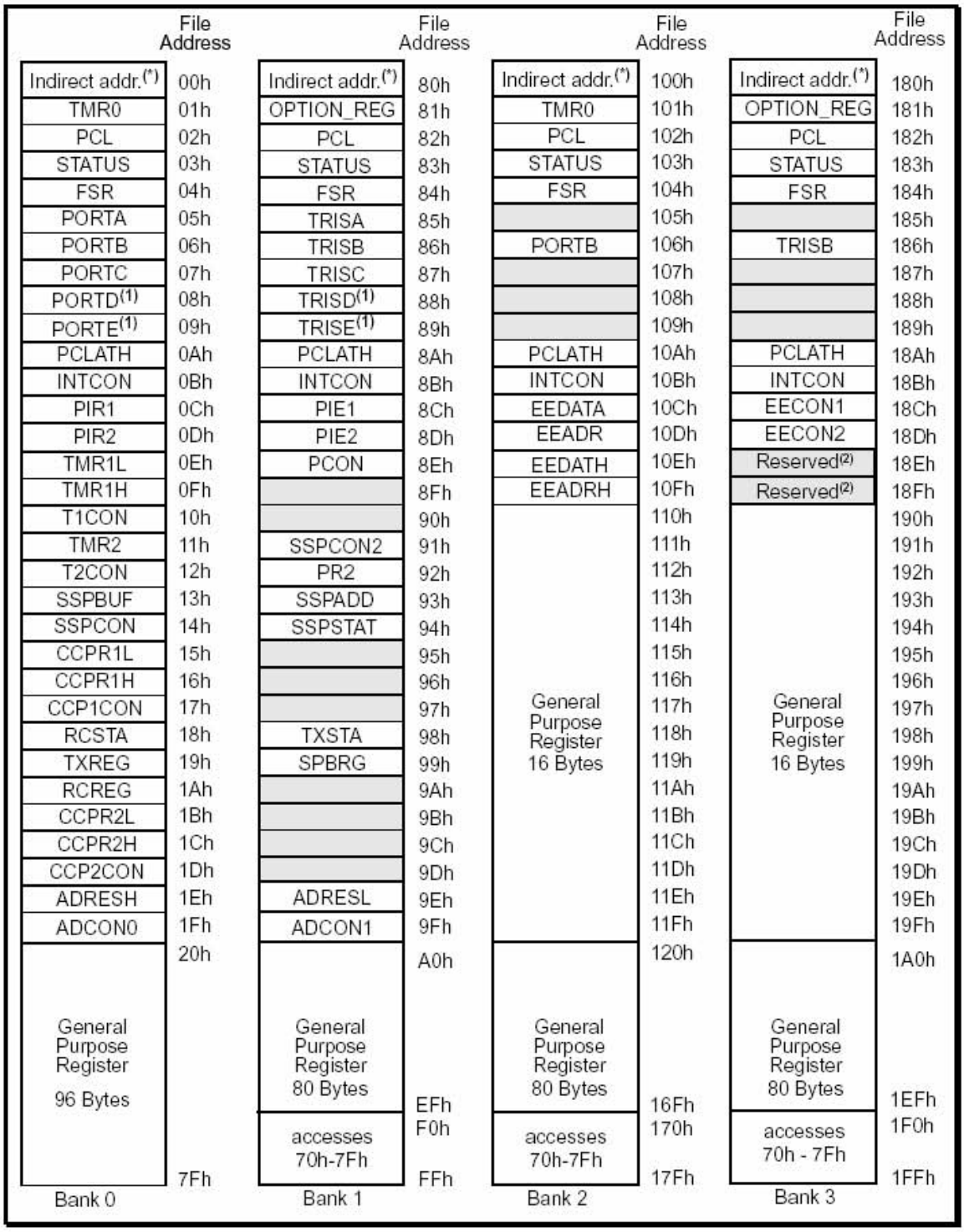


Figura 2.5.2 Distribución de la memoria RAM en cuatro bancos con 368 bytes útiles. [20]

6.- Hasta 256 bytes de memoria de datos EEPROM (Electrical Erasable Programmable Read Only Memory). Es una memoria que una vez instalada en un circuito pueden grabarse y borrarse cuantas veces sea necesario sin ser retirados de dicho circuito. Para ello se usan “grabadores en circuito” que confieren una gran flexibilidad y rapidez a la hora de realizar modificaciones en el programa de trabajo. Sin embargo se garantizan hasta 1000,000 de ciclos de escritura/borrado.

7.- Encapsulados compatibles con los PIC16C73/74/76/77.

8.- Hasta 14 fuentes de interrupción internas y externas.

9.- Pila con 8 niveles.

10.- Modos de direccionamiento, indirecto y relativo.

11. - Perro Guardian (WDT) o “Watch Dog Timer ”. Cuando el PC se bloquea por un fallo del software u otra causa se pulsa el botón del reset y se reinicia el sistema. Pero un microcontrolador funciona sin el control de un supervisor y de forma continua las 24 horas del día. El Perro guardián consiste en un temporizador que, cuando se desborda y pasa por 0, provoca un reset automáticamente en el sistema. Se debe diseñar el programa de trabajo que controla la tarea de forma que refresque o inicialice al Perro guardián antes de que provoque el reset. Si falla el programa o se bloquea, no se refrescará al Perro guardián y, al completar su temporización provocara el reset. [20]

12.- Código de protección programable.

13.- Modo SLEEP de bajo consumo. Son abundantes las situaciones reales de trabajo en que el microcontrolador debe esperar, sin hacer nada, hasta que se produzca algún acontecimiento externo que le ponga de nuevo en funcionamiento. Para ahorrar energía, (factor clave en los aparatos portátiles), los PIC disponen de una instrucción especial (SLEEP), que les pasa al estado de reposo o de bajo consumo, en el cual los requerimientos de potencia son

mínimos. En dicho estado se detiene el reloj principal y se “congelan” sus circuitos asociados, quedando sumido en un profundo “sueño” el microcontrolador. Al activarse una interrupción ocasionada por el acontecimiento esperado, el microcontrolador se despierta y reanuda su trabajo. [20]

14.- Programación serie en circuito con 2 patitas.

15.- Voltaje de alimentación comprendido entre 2 y 5.5 V.

16.- Bajo consumo de energía (menos de 2mA a 5V y 5MHz).

Dispositivos periféricos.

1.- Timer0: Temporizador- contador de 8 bits con predivisor de 8 bits.

2.- Timer1: Temporizador- contador de 16 bits con predivisor.

3.- Timer2: Temporizador-contador de 8 bits con predivisor y postdivisor.

4.- Dos módulos de Captura-Comparación-PWM.

5.- Conversor A/D de 10 bits.

6.- Puerto serie asíncrono (SSP) con SPI e I2C.

7.- USART

8.- Puerta Paralela Esclava (PSP).

Descripción de los pines

En la figura 2.5.3 se muestra el diagrama de asignación de pines del PIC16F877.

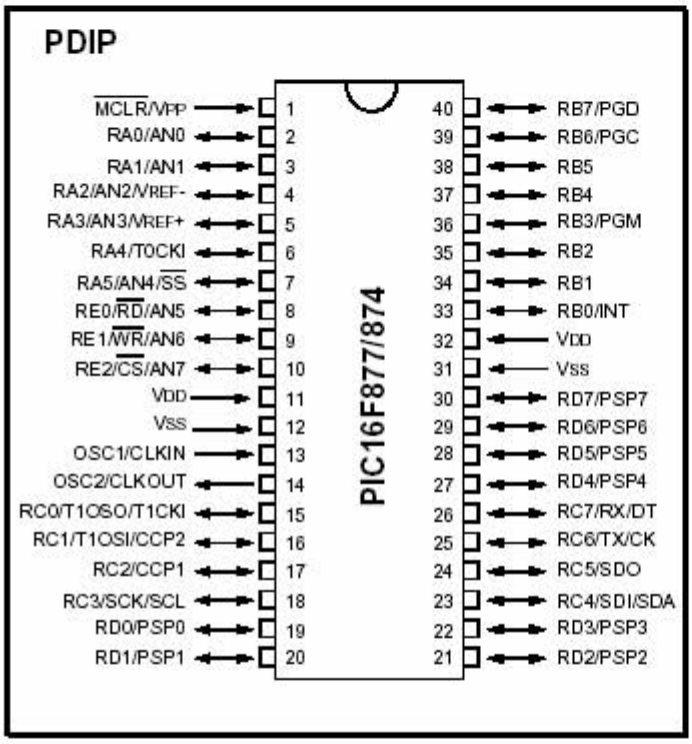


Figura 2.5.3 Diagrama de asignación de pines del PIC16F877. [20]

La empresa Microchip y otras que utilizan los PIC ponen a disposición de los usuarios numerosas herramientas para desarrollar hardware y software. Son muy abundantes los programadores, los simuladores software, los emuladores en tiempo real, Ensambladores, Compiladores C, Intérpretes y Compiladores BASIC. Las razones de la excelente aceptación de los PIC son las siguientes:

1.- sencillez de manejo: Tienen un juego de instrucciones reducido; 35 en la gama media.

2.-Buena información, fácil de conseguir y económica.

3.-Precio: Su coste es comparativamente inferior al de sus competidores.

4.-Poseen una elevada velocidad de funcionamiento. Buen promedio de

Parámetros: velocidad, consumo, tamaño, alimentación y código compacto

5.- Herramientas de desarrollo fáciles y baratas. Muchas herramientas de software se pueden recoger libremente a través de Internet desde Microchip (http://www.microchip.com).

6.- Existe una gran variedad de herramientas de hardware que permiten grabar, depurar, borrar y comprobar el comportamiento de los PIC.

7.- La gran variedad de modelos de PIC permite elegir el que mejor responde a los requerimientos de la aplicación.

2.6 CARACTERÍSTICAS, VENTAJAS Y DESVENTAJAS DE LOS DRIVERS PARA MOTORES DE CORRIENTE DIRECTA.

Podemos controlar motores de corriente directa con el uso de transistores, cuando se trata de controlar la velocidad en un solo sentido de giro lo podemos hacer con un solo transistor, como se muestra en la figura 2.6.1, en este caso Vin puede ser un una señal PWM proveniente de un microcontrolador PIC, y Vmot el voltaje de alimentación del motor.

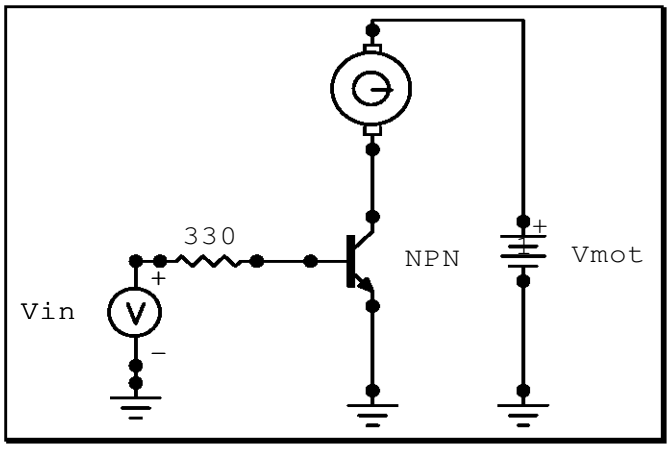


Figura 2.6.1 circuito para controlar la velocidad de un motor de corriente directa.

Sin embargo cuando se trata de controlar la velocidad y la dirección de un motor, es necesario usar lo que se conoce como puente “H”, se trata de un arreglo de 4 transistores

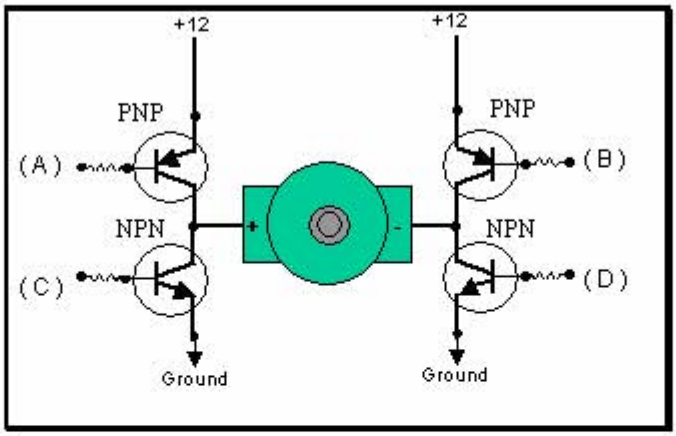


Figura 2.6.2 Arreglo de transistores en puente H

Para activar el motor hacia un sentido, se puede lograr activando los transistores A y D al mismo tiempo y desactivando B y C, en este caso la corriente pasa por el transistor A, como el transistor B se encuentra abierto, la corriente entra al motor en la parte (+) sale por (-) y se va a tierra por el transistor D. Para cambiar el sentido de giro simplemente se activan los transistores B y C y se desactivan A y D.

Es evidente que los transistores A y C no deben de estar activados al mismo tiempo, a si mismo los transistores B y D ya que en ese caso se produciría una elevada corriente a través de los transistores.

En algunos casos es necesario añadir unos diodos de protección para los transistores, ya que como sabemos los motores también se comportan como generadores, y pueden producir voltajes en sentido inverso cuando se cambia el sentido de giro bruscamente. Y es posible que se queme alguno de los transistores, estos diodos se colocan como se muestra en la figura 2.6.3

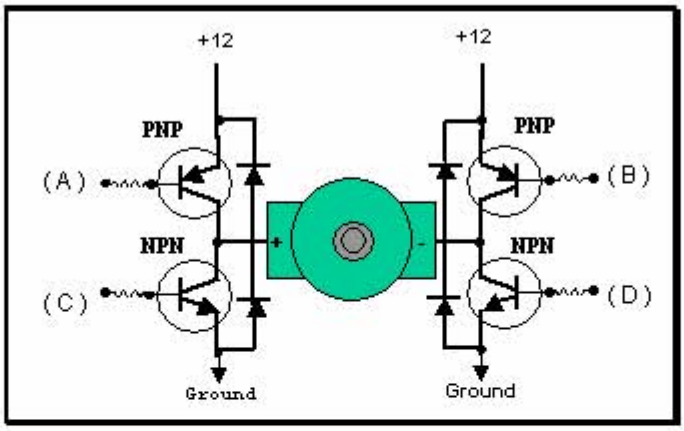


Figura 2.6.3 Puente H con diodos de protección.

Los transistores siendo de material semiconductor, tendrán un poco de resistencia, esta resistencia causa calentamiento, cuando conducen mucha corriente, esto se debe al efecto Joule.

Los transistores Mosfet son mas eficientes que los TBJ, pueden conducir mucho mas corriente, sin calentarse tanto y normalmente traen diodos de protección integrados en el mismo encapsulado, con esto no es necesario usarlos de forma externa.

Existen en el mercado puentes H encapsulados en un solo circuito integrado, estos son conocidos como drivers, los mas conocidos son el L293E y L298, la diferencia mas importante entre estos 2 es que el segundo conduce una mayor corriente, por lo tanto en este capitulo analizaremos únicamente el L298, que es el que nos interesa ya que es uno de los mas comerciales y fáciles de conseguir en el mercado nacional, así como su bajo costo.

Con este driver podemos controlar hasta 2 motores de corriente directa en velocidad y dirección, como la mayoría de los drivers, necesita un voltaje de alimentación para los motores +Vs la cual puede ser desde 2.5 V hasta 46 V

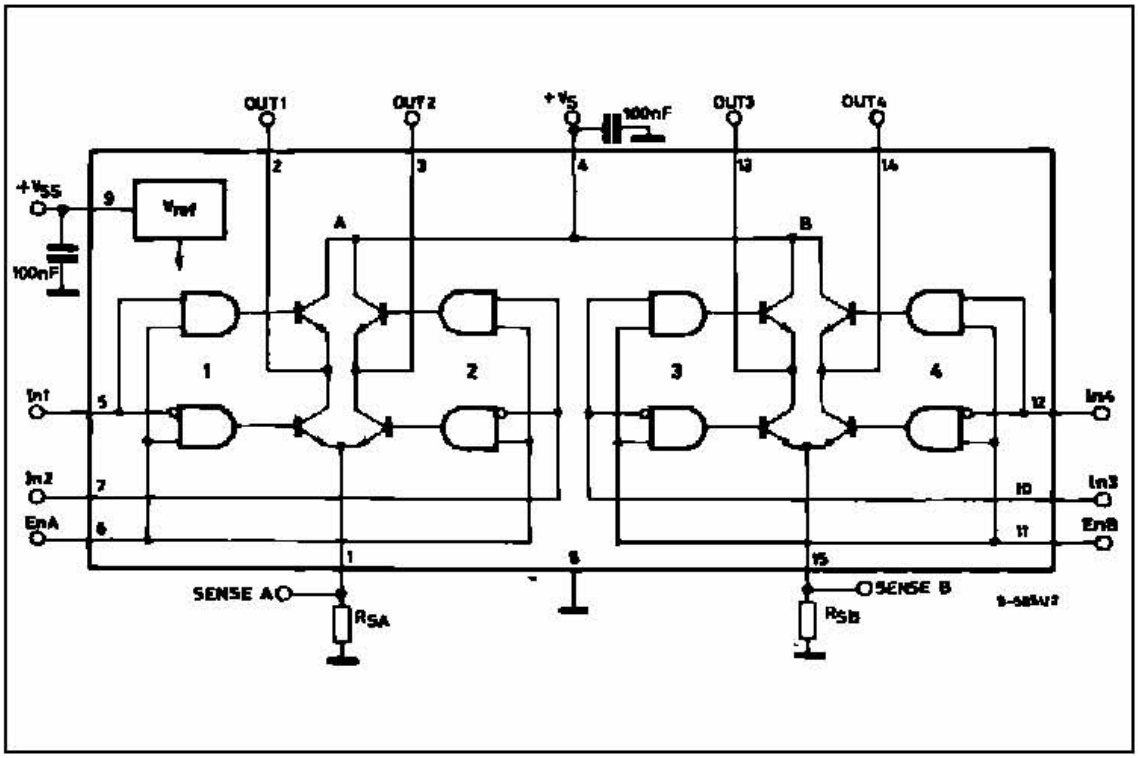
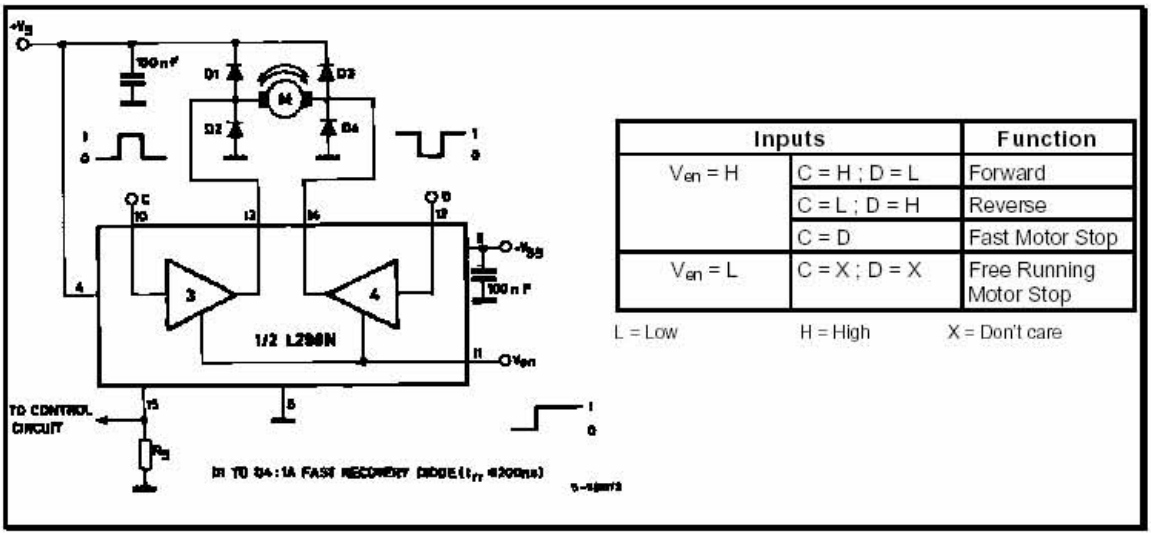


Figura 2.6.4 Diagrama a bloques del driver L298. [21]

Las entradas in 1 hasta in 4 son entradas lógicas, las cuales pueden venir directamente del microcontrolador PIC. Vss es un voltaje lógico, el cual toma de referencia el driver, para identificar el valor del 1 lógico, el cual puede ser desde 4.5V hasta 7V, en nuestro caso este valor es el típico de 5V. tenemos también 2 entradas En1 y En2 estas se utilizan como habilitadoras de las salidas 1, 2 y 3, 4 en el orden indicado. Es importante recordar que debemos de usar los diodos de protección para cada motor. [21]

En la figura 2.6.4 se muestra un circuito de prueba para este driver, en el cual a la derecha se muestra una tabla de valores lógicos para las entradas y las salidas, existe un valor de mucho interés para nosotros C=D “Fast Motor Stop” lo que conocemos como paro rápido, en esta condición, cuando nuestro motor va a alta velocidad, y necesitamos frenarlo bruscamente, podemos utilizar esta

condición de paro, es importante que la tengamos en cuenta al momento de programación del algoritmo de nuestro robot.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

Figura 2.6.5 Circuito de prueba para el L298. [21]

2.7 HERRAMIENTAS PARA TRABAJAR CON MICROCONTROLADORES PIC.

Existe una gran gama de herramientas en Internet para trabajar con los microcontroladores de microchip, aparte de la gran cantidad de información y libros de texto.

MPLAB IDE Y MPLAB ICD 2

En la parte de software podemos utilizar el programa MPLAB IDE de microchip, este lo podemos bajar de la página www.microchip.com, y lo mas importante es que es gratuito.

El MPLAB-IDE es una Plataforma de Desarrollo Integrada bajo Windows, con múltiples prestaciones, que permite escribir su programa en lenguaje assembler o C (el compilador C se compra aparte), crear proyectos, ensamblar o compilar , simular el programa y finalmente programar el componente, si es que se cuenta con el PICSTART-PLUS o alguna otra herramienta para programar como MPLAB ICD 2.[11]

Partes del MPLAB-IDE:

EDITOR: Editor incorporado que permite escribir y editar programas u otros archivos de texto

PROJECT MANAGER: Organiza los distinto archivos relacionados con un programa en un proyecto. Permite crear un proyecto, editar y simular un programa. Además crea archivos objetos y permite bajar archivos hacia emuladores (MPLAB-ICE) o simuladores de hardware (SIMICE)

SIMULADOR: Simulador de eventos discretos que permite simular programas con ilimitados breakpoint, examinar/modificar registros, observar variables, tiempos y simular estímulos externos.

ENSAMBLADOR: Genera varios tipos de archivos objetos y relacionados, para programadores Microchip y universales

LINKER: Permite unir varios archivos objetos en uno solo, generados por el ensamblador o compiladores C como MPAB-C18 o compiladores de terceros

INTERFACE A HARDWARE COMO: PICSTART-PLUS, PROMATE II, MPLAB-ICD, MPLAB-ICE, ICEPIC, SIMICE, OTROS

MPLAB-ICD2 es un debugger y programador a la vez, puede cumplir ambas funciones. Pero cuando trabaja en conjunto con el MPLAB-IDE, cumple una u otra función, no ambas a la vez

La idea del ICD2 es realizar el debugging y la programación del uC IN-CIRCUIT, es decir, directamente en la placa. El procesador puede estar incluso soldado.

ICD2 se comunica con MPLAB-IDE vía puerto serial o USB (recomendado). Por otro lado, debe ser conectado al procesador. Para ello hace uso de algunos recursos del uC: en los PIC16/18 RB6/PGC, RB7/PGD, VppMCLR, Vdd y Vss como se muestra en la figura 2.7.1.

También utiliza algunos recursos de RAM y SP. Los recursos de RAM y SP dependerán de uC en particular. Para más detalles, consultar la ayuda del MPLAB-IDE.

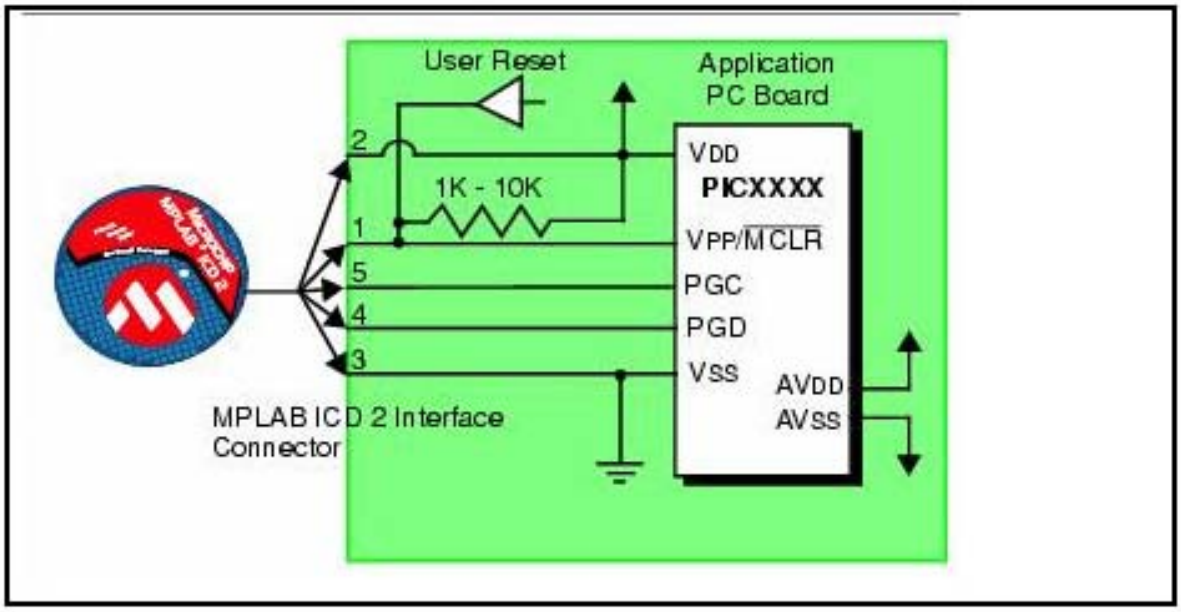


Figura 2.7.1 conexión del MPLAB ICD 2 con un PIC. [19]

El voltaje Vpp (13.5V) es suministrado por el ICD2. El ICD2 puede ser alimentado desde la placa del usuario (recomendado) o desde el puerto USB. En el primer caso, las señales Vdd y Vss, son suministradas por la placa del usuario

El ICD2 se puede utilizar de 2 formas:
Como programador serial para poner el código en la memoria de programa de uC
Como debugger, que permite realizar una simulación a nivel de hardware del código, poner break points. El debugger requiere utilizar el In Circuit Deduger hardware que está presente dentro de los micros

Para el Debugging y programación, el ICD2 hace uso de los pines PCG y PGC

Para los efectos de programación, el ICD2 no necesita que el uC tenga asociado un oscilador (xtal o similar). Además suministra el voltaje Vpp, ingresado al uC por medio del pin MCLR/Vpp, un clock por medio del pin PGC y los datos seriales por el pin PGD

El modo hace uso de algunos recursos del procesador. En el caso del PIC16F877A, el ICD2 hace uso de:

- Memoria de Programa: 1F00h – 1FFFh
- Ram: 70h, F0h,170h,1E5h-1F0h
- SP: 2 niveles

MPLAB IDE puede trabajar con varios tipos de programadores. El usuario debe seleccionar con cual trabajará, haciendo click en opción Programmer/ Select programmer, se pueden seleccionar 3 programadores distintos

- Picstart-plus
- MPLAB-PM3
- Promate II
- MPLAB-ICD2

Antes de seleccionar el ICD2 como debugger o programador, seleccione el micro con el que trabajará, por medio de la opción Configure/Select Device

Para seleccionar el ICD2 como programador, escoja la opción Programmer/Select programmer/MPLAB ICD2 como se muestra en la figura 2.7.2.

MPLAB-IDE detecta si el Sistema Operativo del ICD2 corresponde al uC seleccionado, si no corresponde, lo bajará automáticamente y luego establecerá comunicación con el uC.

Cualquier error de comunicación, conexiones o falta de alimentación, provocará que aparezca en pantalla la respectiva ventana de error con el detalle. [11]

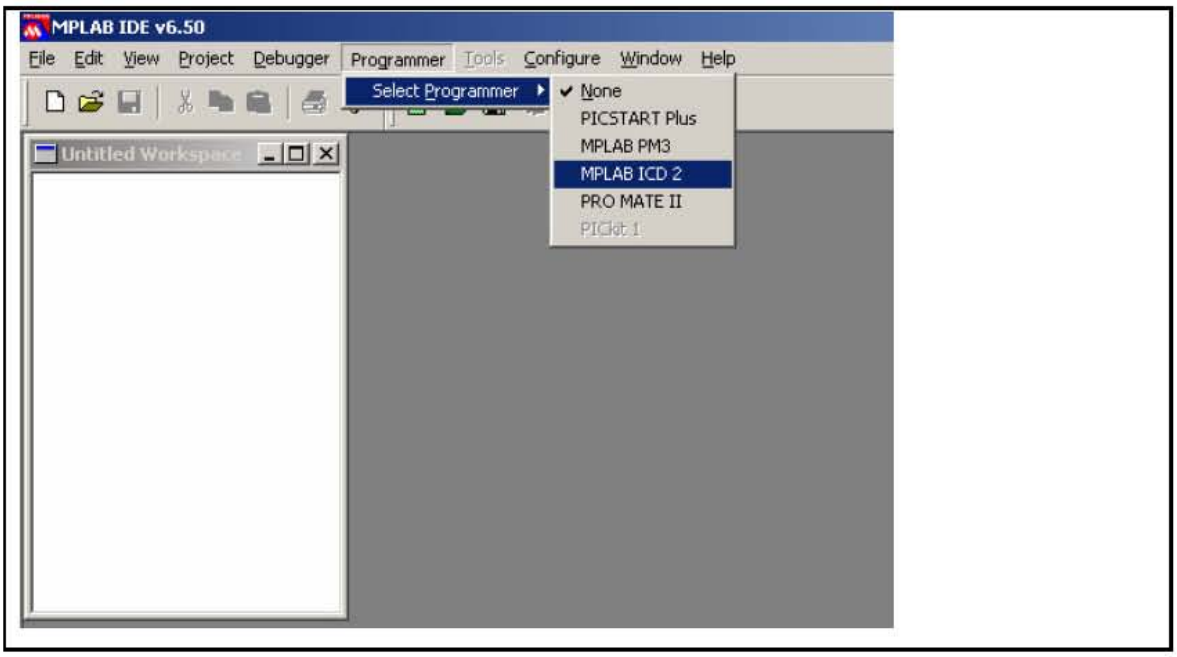


Figura 2.7.2 Selección del MPLAB ICD 2 como programador

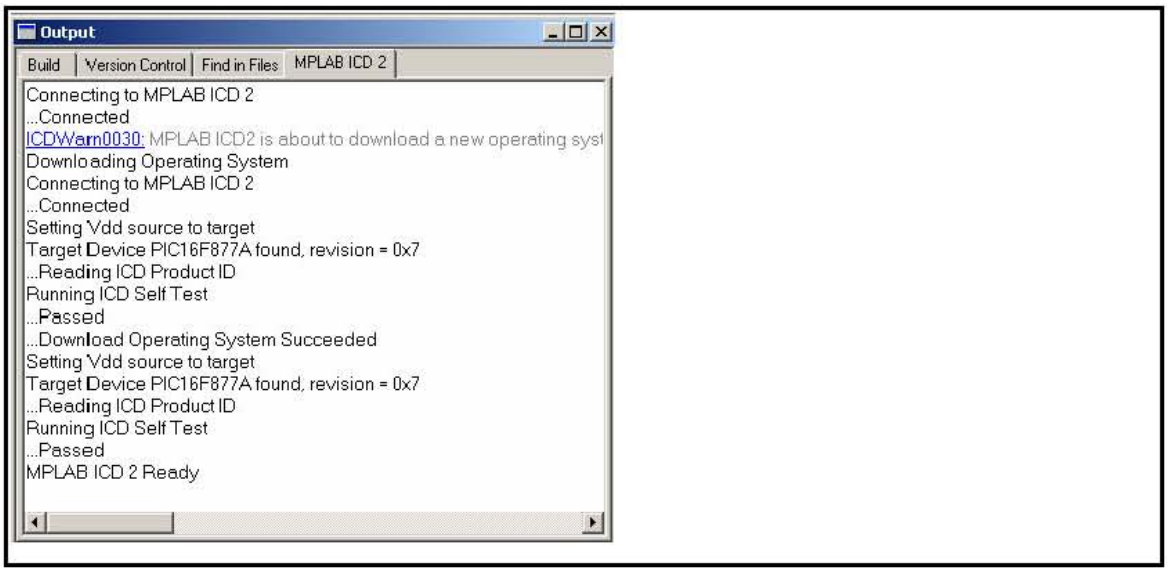


Figura 2.7.3 Verificación de la conexión y el tipo de uC por el MPLAB IDE

La creación de un proyecto comienza con la escritura del programa. Para ello escoja la opción *File/New* y el editor del MPLAB-IDE presentará una página en blanco. Es sugerible que ponga nombre al archivo fuente desde el principio.

Para ello, escoja la opción *File/Save As*, póngale nombre al programa y agregue la extensión .bas, le aparecerá una pantalla como la que se muestra en la figura 2.7.4

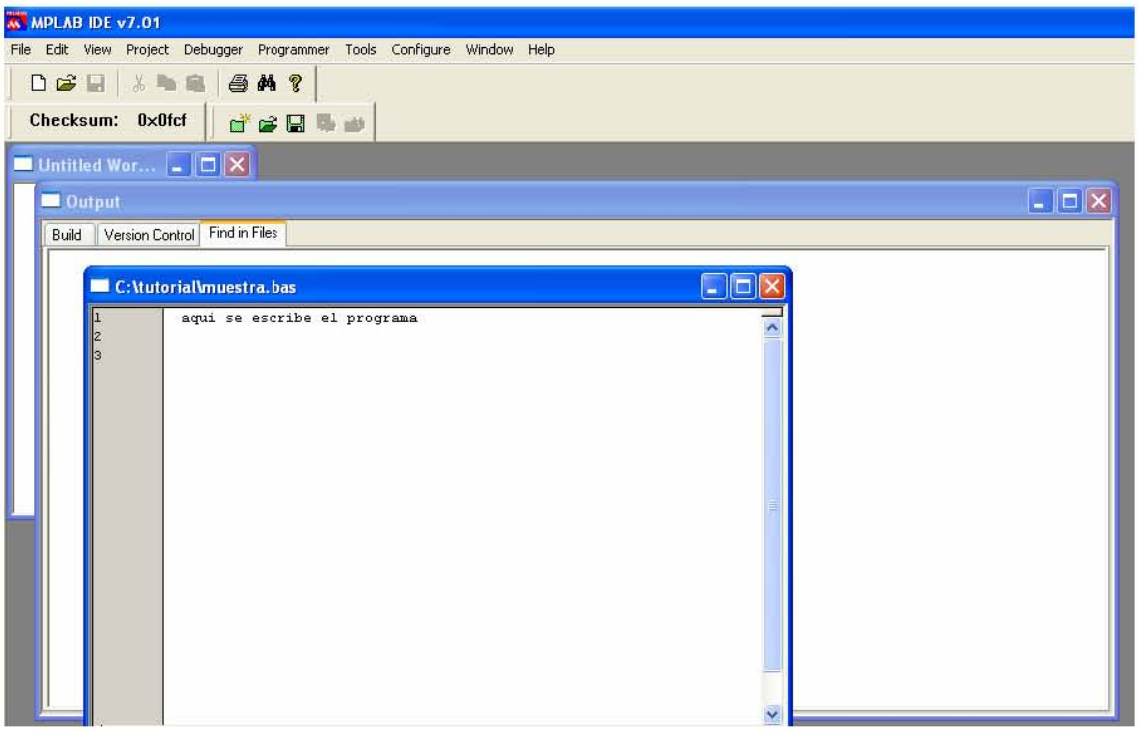


Figura 2.7.4 Primer paso para crear un nuevo proyecto en MPLAB IDE

Una vez escrito el programa, es necesario crear un proyecto para poder compilar, simular, debugear y programar.

Para crear un proyecto, escoja la opción *Proyect/Proyect Wizard* y siga las instrucciones. Seleccione el procesador con el que desea trabajar Seleccione el lenguaje que usará (Assembler, C ó Basic) Póngale nombre al proyecto, de preferencia, el mismo del archivo fuente Luego, indique en qué directorio quedará el proyecto, de preferencia, el mismo donde se encuentra el archivo fuente. Busque y agregue el archivo fuente al proyecto. Una vez que lo encuentre, selecciónelo y haga click en *Add* y aparecerá en la venta derecha., Si el archivo fuente se encuentra en un directorio distinto al directorio que definió en el paso anterior para el proyecto, haga click en el checbox de la derecha. Esto provocará que el MPLAB haga una copia del archivo fuente en el

directorio del proyecto. En las figuras 2.7.5 a 2.7.8 se muestran las pantallas detalladas de los pasos antes mencionados.

Figura 2.7.5 Pantalla donde podemos elegir el procesador a utilizar

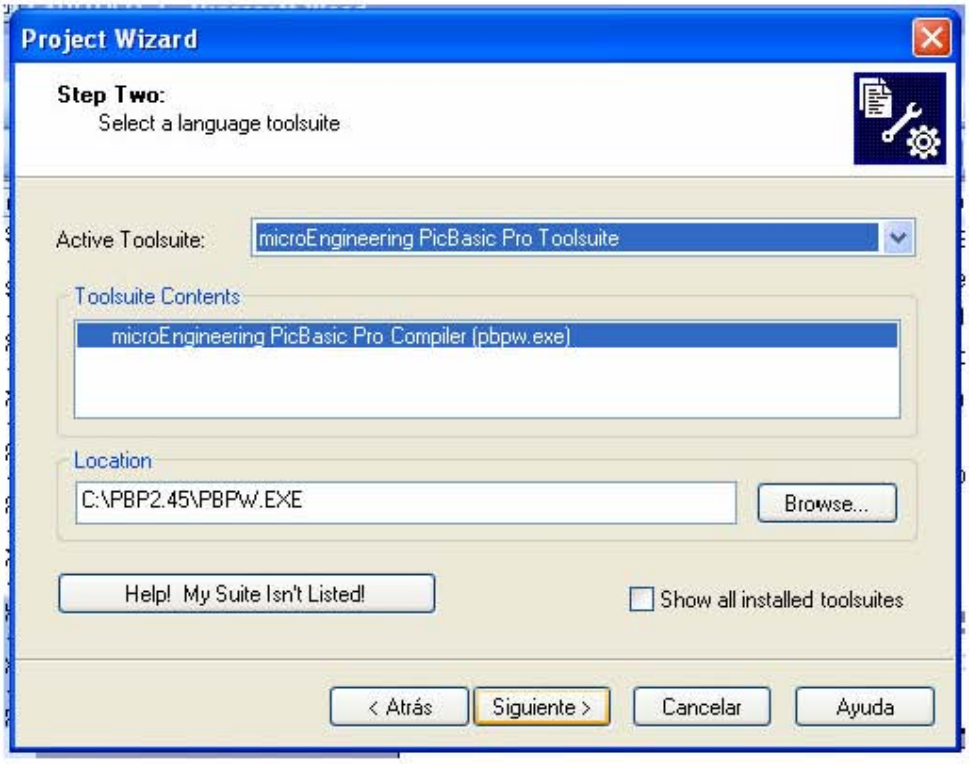


Figura 2.7.6 Pantalla en la cual elegimos el lenguaje de programación

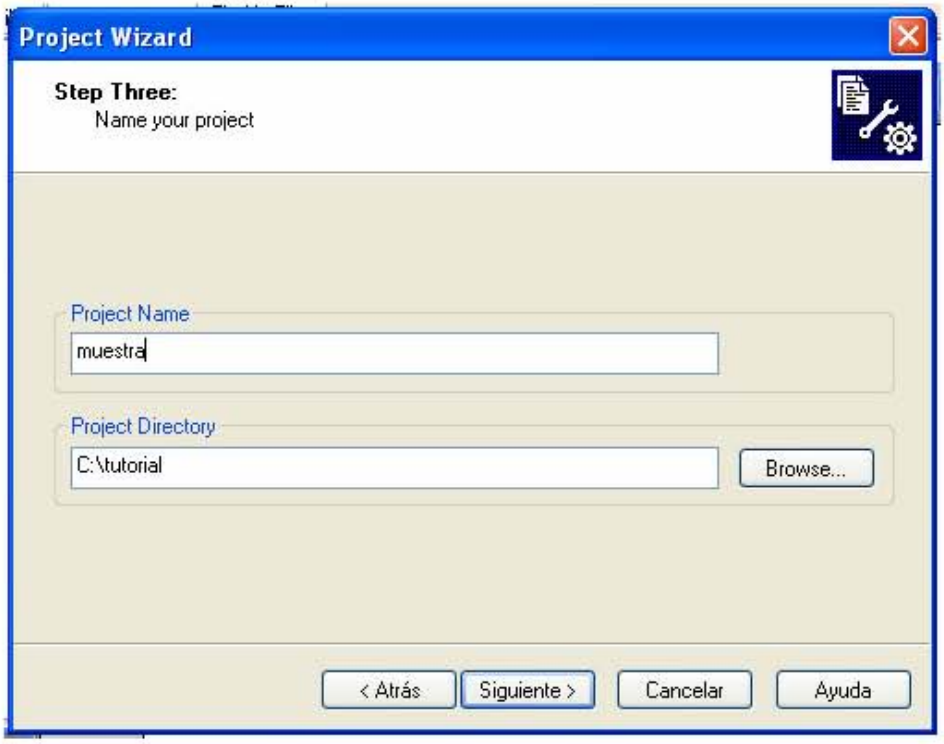


Figura 2.7.7 Pantalla en la cual elegimos el directorio donde se guardara el proyecto y ponemos el nombre del mismo.

Figura 2.7.8 Pantalla donde buscamos y agregamos el archivo fuente del proyecto.

Si todo va bien, se creará un archivo con el nombre del proyecto y extensión mcp, por ejemplo, *timer.mcp*. Lea el resumen para ver si todo está como desea. En la figura 2.7.9 se muestra la pantalla final del proceso.

Si lo está, haga click en finalizar, de lo contrario, cancele y repita el proceso.

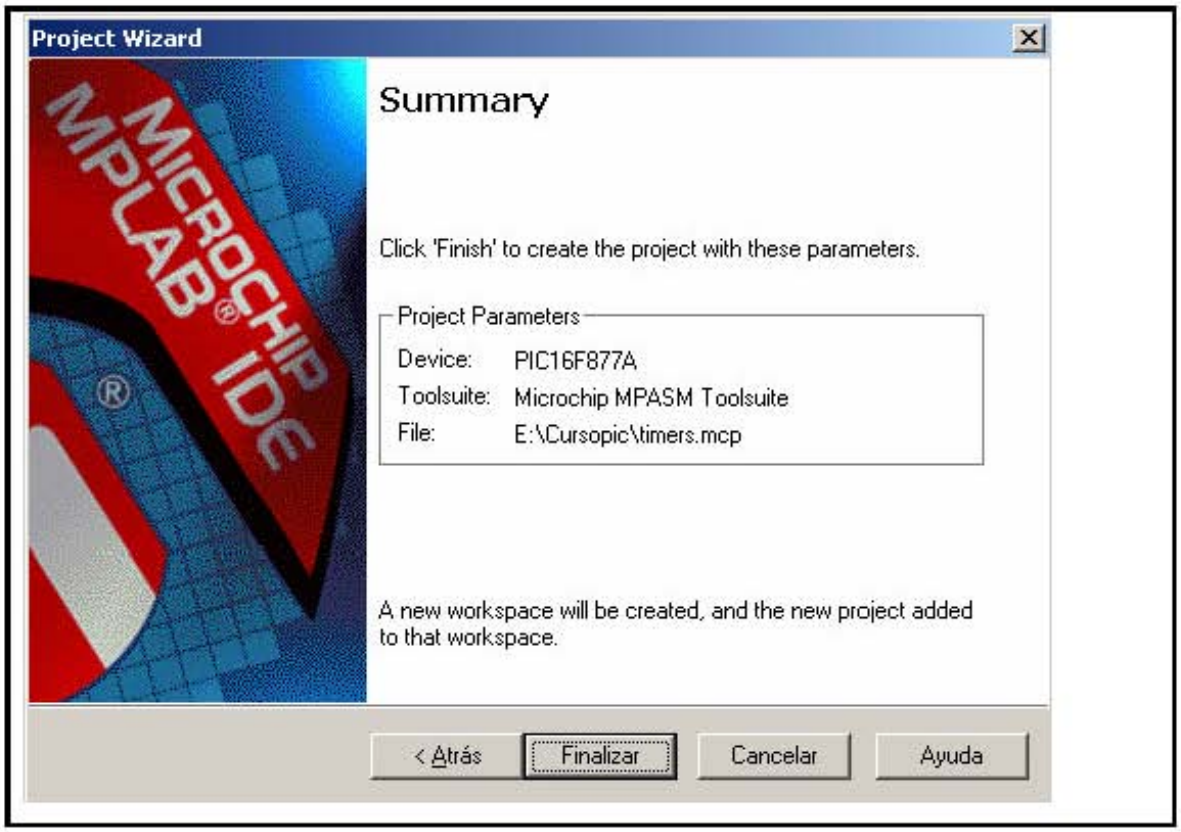


Figura 2.7.9 Datos del proyecto hecho en MPLAB IDE

Una vez creado el proyecto, escriba su programa, en el lenguaje seleccionado anteriormente y haga click en el ícono Built All. Con esto compilará el programa y se cerrarán archivos de error, mapa del programa, archivos objetos y archivos hex.

Si aparece algún error de compilación, corríjalos y vuelva a compilar

Escoja el programador deseado (en este caso el ICD2) por medio del menú *Programmer/Select programmer*

ICD2 verificará el micro escogido y las conexiones respectivas, Cuando el programador es seleccionado, los íconos relacionados con el aparecerán

Luego de establecer comunicación con el programador, es necesario configurar los BITs de CONFIGURACIÓN de micro. Las opciones disponibles varían de micro en micro.

Es sumamente importante tener claro los valores de los bits de configuración, de lo contrario, el micro no trabajará adecuadamente, aunque el programa esté bien estructurado. Para configurar los Bits de configuración, el usuario debe hacer click en *Configure/Configuration Bits* y aparecerá una pantalla como la mostrada en la figura 2.7.10

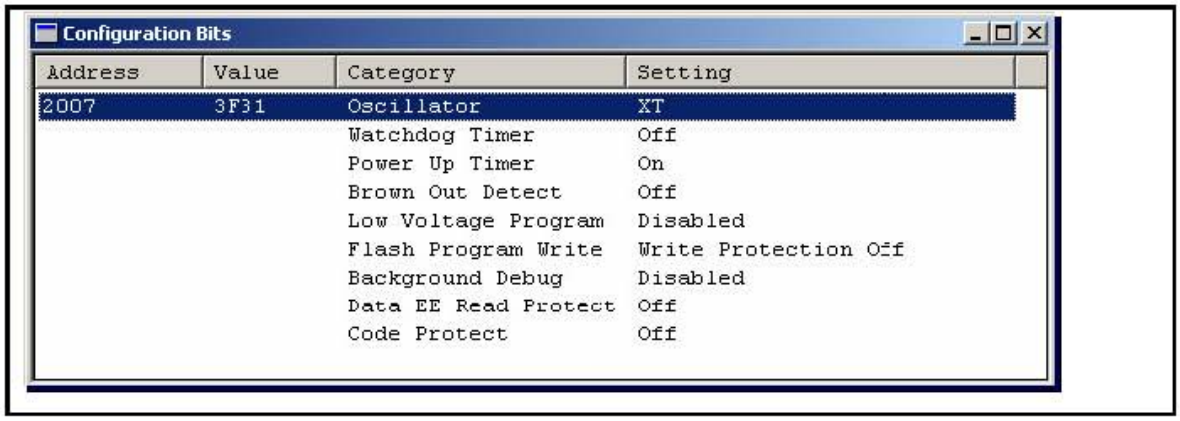


Figura 2.7.10 Pantalla para la configuración de los bits de configuración.

Sólo una vez teniendo los bits de configuración en los valores deseados, se puede proceder a programar el micro escogiendo la opción *Programmer/Program*

Si el proceso no presenta problemas, la ventana de salida mostrará un mensaje que indica que la programación fue exitosa.

Antes de programar, en ocasiones es necesario, configurar el programador Este, por defecto, programa sólo la cantidad de memoria suficiente, dependiendo del tamaño del programa

Una vez configurado el programador, haga click sobre el icono Programar uC para iniciar el proceso de programación, si en medio del proceso ocurre algún error, programe nuevamente

El programador realiza los siguientes pasos:

- Borra el uC.
- Programa la memoria de acuerdo al rango definido en la configuración del programador.
- Verifica si la programación es correcta.
- Programa bits de configuración.
- Verifica programación de bits de configuración.

En la figura 2.7.11 se muestra el seguimiento de la programación del PIC cuando fue exitosa.

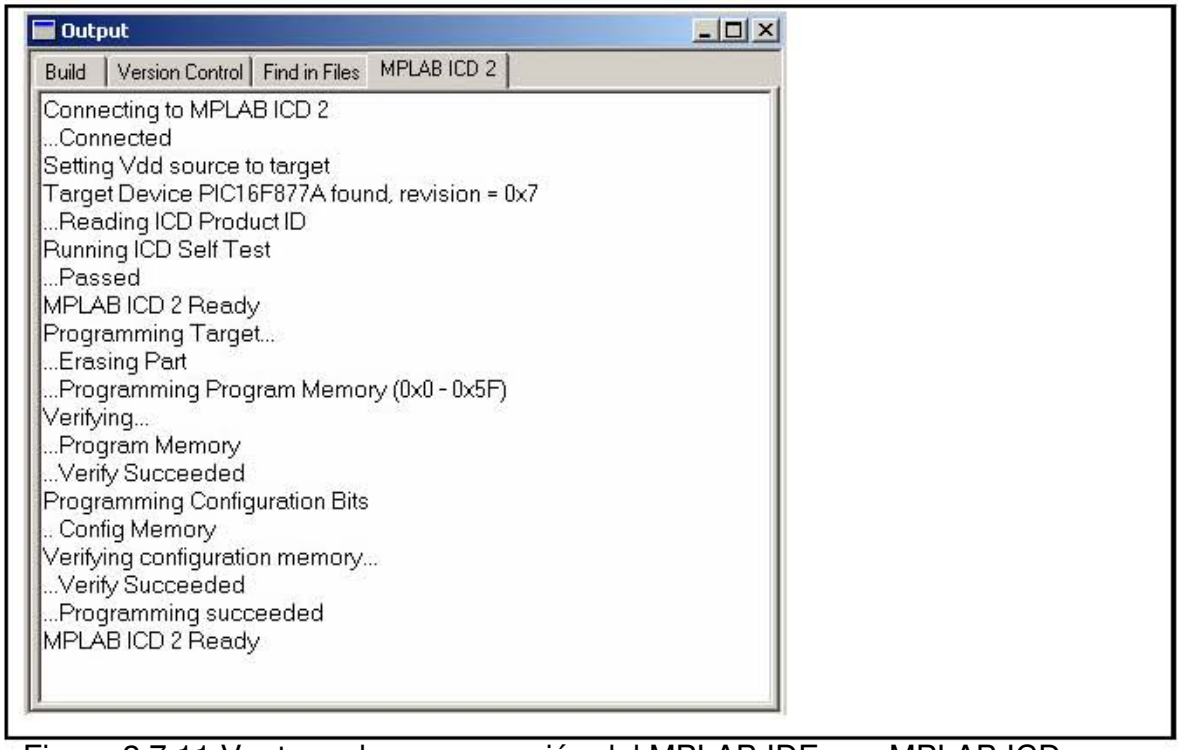


Figura 2.7.11 Ventana de programación del MPLAB IDE con MPLAB ICD

Si desea realizar el debugging del programa, primero deshabilite el ICD2 si es que fue seleccionado como programador, luego escoja la opción *Debugger/Select Tool/MPLAB ICD 2*. ICD2 establecerá nuevamente comunicación con el micro, verificará conexiones y alimentación y se activarán los íconos del ICD2. Esta parte es muy importante para nuestro proyecto en el momento de hacer pruebas ya que con esto podemos conocer los valores que esta recibiendo el uC en tiempo real, y así corregir detalles del hardware con el software. Ya que muchas veces nuestro programa esta correcto, y los valores que esta recibiendo el uC son incorrectos, y esto nos puede provocar mucha perdida de tiempo buscando errores en el programa, cuando nuestro error puede estar en los valores de entrada.

Lenguaje de programación PIC BASIC PRO

El compilador Pic Basic Pro (PBP) es un lenguaje de programación de nueva generación que hace más fácil y rápido programar el micro controlador Pic micro de **Microchip Technology** .

El lenguaje Basic es mucho más fácil de leer y escribir que el lenguaje ensamblador.

PBP por defecto crea archivos que corren en un PIC 16F84-04/P con un reloj de 4 Mhz. Solamente muy pocas partes son necesarias dos capacitores de 22 pf para el cristal de 4Mhz un resistor de 4.7K en el pin/MCLR y una fuente de 5 volts. Otros uC PIC además del 16F84, así como otros osciladores de frecuencias distintas pueden ser usados por este compilador.[12]

El PBP produce código que puede ser programado para una variedad de micro controladores PIC que tengan de 8 a 68 pines y varias opciones en el chip incluyendo convertidores A/D, temporizadores y puertos seriales.

Hay algunos micros PIC que no trabajaran con el PBP, por ejemplo las series PIC 16C5X incluyendo el PIC 16C54 Y PIC 15C58. Estos micro PIC están basados en el viejo núcleo de 12 bits en lugar del núcleo más reciente de 14

bits. El PBP necesita alguna de las opciones que solamente están disponibles con el núcleo de 14 bits como el stack (pila) de 8 niveles.[12]

Hay muchos micros PIC, algunos compatibles pin a pin con la serie 5X, que pueden ser usados con el PBP. La lista incluye PIC16C554, 556, 558, 61, 62(A), 620, 621, 622, 63, 64(A), 65(A), 71, 710, 711, 715, 72, 73(A), 74(A), 84, 923, 924, el PIC16F83 y 84, el PIC12C671 y 672 y el PIC14C000, y Microchip sigue agregando otros. Para reemplazo directo de un PIC16C54 o 58, el PIC16C554, 558, 620 y 622 funcionan bien con el compilador y tienen aproximadamente el mismo precio. [12]

El PIC16F877 contiene 256 bytes de memoria de datos no volátil que puede ser usada para archivar los datos de programa y otros parámetros, aun cuando no haya energía. A ésta área de datos, se puede acceder simplemente usando las órdenes “Read” y “Write” del PBP. (El código del programa es permanentemente guardado en el espacio de código del micro PIC, tanto si hay o no energía.)

Entre las instrucciones que nos interesan mas están las de modulación por ancho de pulso (para controlar la velocidad de los motores):

El primer parámetro es el canal por el cual va a mandar el tren de pulsos, en este caso es CCP1, el segundo parámetro es un valor que va de 0 a 255 y es el valor del porcentaje en estado alto, en este caso 127 corresponde a un 50%, y por ultimo el tercer parámetro es la frecuencia de los pulsos, en este caso es de 1kHz.[12]

```
Hpwm 1,127,1000       ' Send a 50% duty cycle PWM signal at 1kHz
```

Convertidor analógico digital (para leer los voltajes de los sensores):

En esta instrucción, primero es necesario definir 3 parámetros del convertidor A/D: numero de bits del resultado, en este caso son 10 (recordemos que

también podemos utilizarlo con 8 bits de resolución), tipo de reloj a utilizar, en este caso es el tipo 3 = RC y el tiempo de muestreo que en este caso son 50 uS estos valores deben de estar definidos en el encabezado de nuestro programa (recomendado), aunque pueden definirse en cualquier parte de nuestro programa.

```
Define ADC_BITS    10      ' Set number of bits in result

Define ADC_CLOCK   3       ' Set clock source (3=rc)

Define ADC_SAMPLEUS 50     ' Set sampling time in uS
```

La instrucción ADCIN tiene 2 parámetros, el primero es el canal de conversión, y el segundo es la variable "valor" que tuvo que haber sido definida en nuestro encabezado del programa, esta debe tener la longitud suficiente para guardar el valor del resultado de la conversión, para 8 bits debe de definirse de tamaño BYTE, para 10 bits debe definirse de tamaño WORD.[12]

```
ADCIN 4,valor
```

Las instrucciones de acceso a la memoria EEPROM. (ya que en este lugar guardaremos la ruta de nuestro robot)

```
READ AP, REAL      'lee la dirección AP y lo guarda en la variable REAL

WRITE AP, CRUCE 'escribe el valor de la variable CRUCE en la dirección AP
```

Estas son las instrucciones que nos facilitaran en gran medida nuestra parte de programación, las demás instrucciones que utilizaremos las iremos explicando en nuestro programa como comentarios.

Instalación del software

Dependiendo de la versión del software, se puede instalar copiando los archivos a una carpeta en c llamada PBP, o sea "C:\PBP". Cuando el software trae un ejecutable, este se encarga de instalarlo correctamente, sin embargo, para ejecutarlo desde MPLAB IDE, necesitamos bajar unos plugins que podemos bajar de la pagina http://www.melabs.com/ downloads/PBPlugins.zip es la pagina de la empresa creadora de PBP MICRO ENGINEERING LABS, INC. El archivo es "PBregister.bat" el cual lo tenemos que ejecutar en el directorio C:\PBP\PBregister.bat cerramos MPLAB y lo volvemos a ejecutar, y listo, aparecerá como lenguaje compatible. Vea la figura 2.7.12[13]

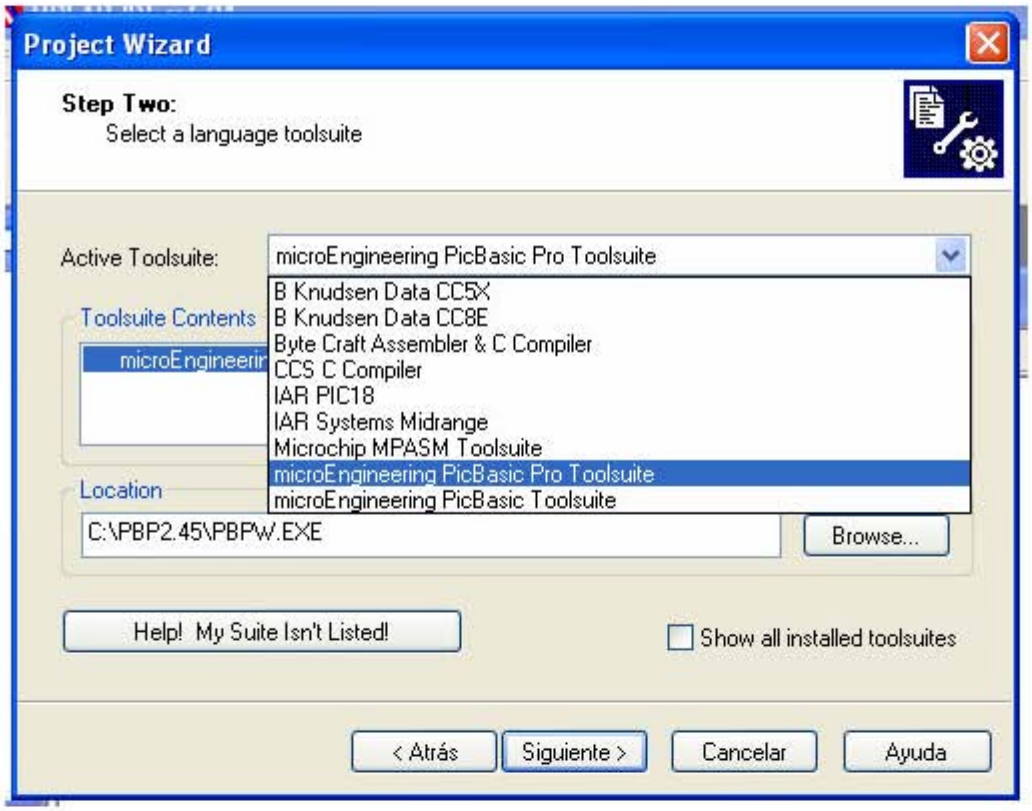


Figura 2.7.12 Ventana en la cual MPLAB reconoce PBP como lenguaje compatible

Otras herramientas para programar PICS Y DSPICS

Existe en Internet una gran gama de herramientas y software para programar microcontroladores PIC, y ahora DSPIC, entre los mas famosos para programar PIC, se encuentra el llamado JDM Programer, que funciona con un programa llamado ICPROG, estas herramientas las podemos encontrar en Internet en la pagina http://www.ic-prog.com/, existe otro programador de PIC y de DSPIC que funciona por puerto USB llamado GTP-USB que trabaja con un software llamado WINPIC800 el cual podemos descargar libremente desde Internet desde http://perso.wanadoo.es/siscobf/ , u otras paginas.

Es importante contar con herramientas de bajo costo para trabajar con microcontroladores, ya que para gran parte de los estudiantes, resulta difícil adquirir herramientas de grabación caras. La diferencia del precio entre construir un programador como los vistos aquí y comprar uno de fábrica es considerable.

DISEÑO Y CONSTRUCCIÓN DEL ROBOT

3.1 EVALUACIÓN Y PRUEBA DE LOS SENSORES ÓPTICOS.

Para evaluar los sensores primero construiremos un circuito de pruebas, en el cual se conectara un sensor a un puerto del PIC como se muestra en la figura 3.1.1

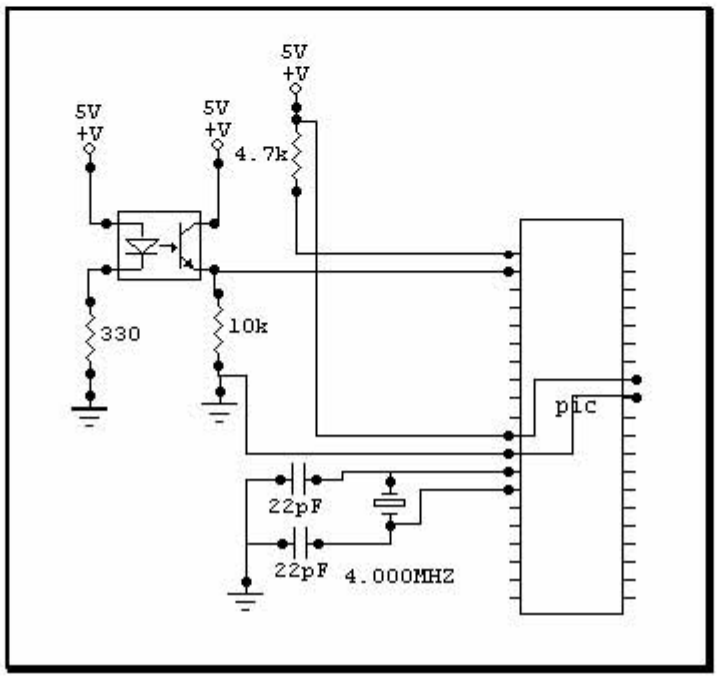


Figura 3.1.1 Circuito de prueba para los sensores ópticos

Como ya se vio en el capitulo 2, los emisores infrarrojos iluminan la pista con un espectro infrarrojo (con una longitud de onda de 940 nm), cuando la superficie es de color negra, esta reflejara muy poca parte de la luz, sin embargo, cuando la superficie es blanca esta reflejara una mayor cantidad de luz en el fototransistor. En el fototransistor, cuando detecta poca luz, este conducirá poca corriente, sin embargo, cuando detecte una gran concentración de luz, este dejara pasar mas corriente, esto se ve reflejado en voltaje en la resistencia de emisor, como la mostrada en la figura 3.1.1, que es de 10k., con estas características, solo tenemos que buscar un umbral para decidir cuando esta viendo blanco o cuando esta viendo negro. Y así en nuestro programa trabajar con valores lógicos únicamente. Es importante tener en cuenta que la distancia de la superficie al sensor debe ser aproximadamente 3 milímetros, dado que según la grafica de distancia-corriente que proporciona el fabricante,

mostrada en la figura 3.1.2 con esta distancia se obtiene el máximo valor de corriente.

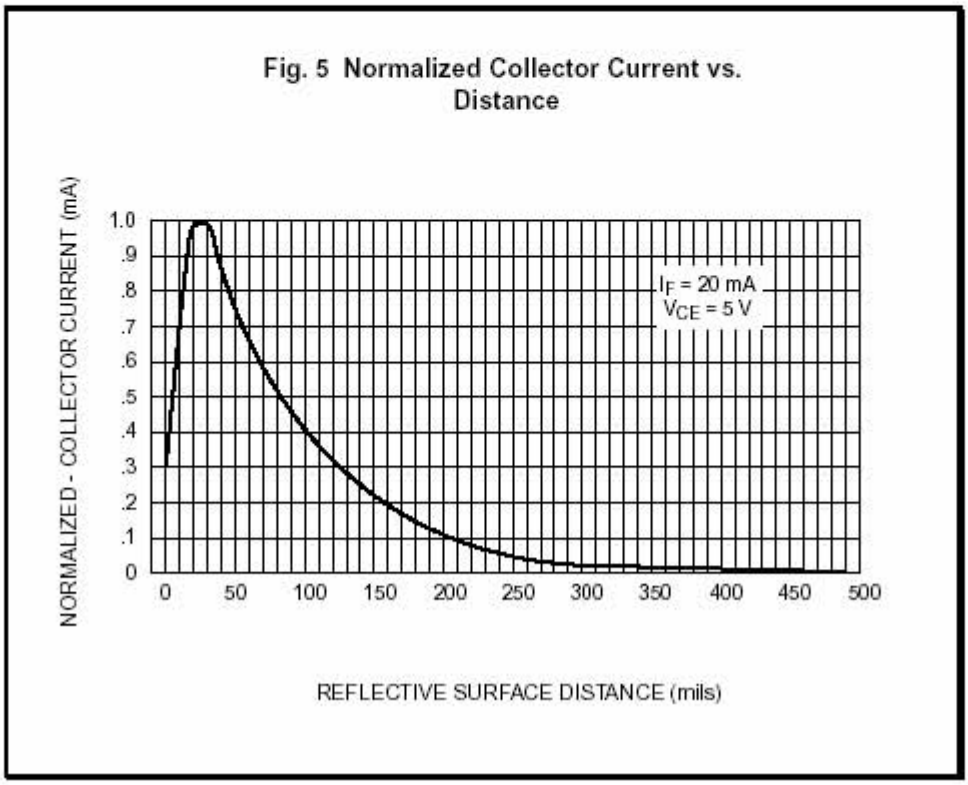


Figura 3.1.2 Corriente de colector normalizada-distancia QRD1114 de Fairchild.

Una vez montado el circuito de la figura 3.1.1, procedemos a medir los valores de voltaje de entrada al PIC por el canal de conversión AN0. En este caso el valor mínimo fue 0.3V y el máximo es de 4.2V.

Cuando el PIC detecta estos voltajes en el convertidor A/D lo que hace es asignarles un numero binario entre 0x000 y 0x3FF por lo tanto debemos convertir estos valores de tensión a un valor digital, para manejarlos en nuestro programa. Si dividimos 5V /1024 = 0.0049, por lo tanto si deseamos conocer el valor binario de 4.2V basta con dividir 4.2/.0049= 857 y 0.3/0.0049= 61 estos valores son obtenidos con redondeo simétrico.

Voltaje	Hexadecimal	Binario	Decimal
0 V ---	0x000	b0000000000	0
0.3V ---	0x03D	b0000111101	61
4.2V ---	0x359	b1101011001	857
5 V ---	0x3FF	b1111111111	1023

Por lo tanto podemos proponer un umbral de 459 que es el valor intermedio entre nuestro máximo y mínimo. Esto es cuando nuestro valor del convertidor A/D sea mayor a 459 esta viendo blanco y cuando sea menor, esta viendo negro. Este no debe ser el valor de umbral estrictamente, este es un valor propuesto, para cuestiones de nuestro robot podemos mover hacia arriba o hacia abajo este valor y lo podemos mover a nuestra conveniencia, mas adelante explicaremos por que.

3.2 PROGRAMACIÓN DEL ALGORITMO PARA EL MOVIMIENTO DE LOS MOTORES.

En esta parte tenemos que ver que existen varias formas de mover los motores de nuestro robot, ya sea para corregir la trayectoria cuando esta siguiendo la línea, o para dar vueltas, ya sea a la izquierda o a la derecha en un cruce y por ultimo cuando ya detecto la meta, que es un circulo de color negro de 15 cm. De diámetro.

A continuación escribiremos nuestra parte del programa que se va a encargar de controlar los motores, para esto consideraremos que el driver esta conectado a los puertos CCP1 y CCP2, así mismo PORTB.4 Y PORTB.5 respectivamente para cada motor.

En este programa utilizaremos la instrucción Hpwm, la cual es utilizada para mandar un tren de pulsos por un pin de un uC. El cual cuente con el hardware para generarlo.

```
Include "modedefs.bas"
Include "ICDDEFS.BAS"
TRISB = %00001000
TRISC = %00000000   ´Todo el puerto c como salidas
GOTO INICIO
´En esta subrutina mandamos un PWM con un ciclo de trabajo de 0% y
una frecuencia de 300 Hz en cada motor y ambos bits de dirección
apagados, esta subrutina la utilizaremos cuando hayamos llegado a la
meta, ya que nuestro robot debe detenerse cuando haya llegado a la
salida.
PARAR:
        Hpwm 2,0,300
        Hpwm 1,0,300
        PORTB.5=0
```

```
        PORTB.4=0
        RETURN
'En la subrutina derecho, la cual aplicaremos cuando nuestro sensor CC vea la línea negra, en este caso tenemos la velocidad máxima con un ciclo de trabajo de 100%.
DERECHO:
        Hpwm 2,255,300
        Hpwm 1,255,300
        PORTB.5=0
        PORTB.4=0
        RETURN

'En esta subrutina, podemos observar que en CCP2 tenemos un ciclo de trabajo de 100% y  en CCP1 el ciclo de trabajo es menor, con esto logramos un movimiento a la derecha de forma suave, esto es para reducir oscilaciones bruscas.
DERECHA:
        Hpwm 2,255,300
        Hpwm 1,30,300
        PORTB.5=0
        PORTB.4=0
        RETURN
'Este caso es similar al anterior, solo que el movimiento es a la izquierda
IZQUIERDA:
        Hpwm 1,255,300    '
        Hpwm 2,30,300
        PORTB.4=0
        PORTB.5=0
        RETURN
'Esta subrutina la utilizaremos cuando necesitemos que el robot  gire sobre su propio eje, en este caso es para girar a la izquierda, y será utilizada cuando queramos hacer un giro brusco a 90°
IZQB:
        HPWM 1,127,300
```

```
        HPWM 2,127,300
        PORTB.5=0
        PORTB.4=1
        RETURN
'Este caso es similar al anterior en ambos casos el ciclo de trabajo es de 50%, esto es para que no gire demasiado rápido, ya que de girar muy rápido se puede generar inercia rotacional, provocando que el robot gire mas de lo requerido.
DERB:
        HPWM 1,127,300
        HPWM 2,127,300
        PORTB.4=0
        PORTB.5=1
        RETURN
'A continuación en el programa principal se da un barrido de todos los movimientos anteriores con un retardo de 1 seg. En cada movimiento.
INICIO:
  GOSUB DERECHO
  PAUSE 1000
  GOSUB DERECHA
  PAUSE 1000
  GOSUB IZQUIERDA
  PAUSE 1000
  GOSUB IZQB
  PAUSE 1000
  GOSUB DERB
  PAUSE 1000
  GOSUB PARAR
  PAUSE 1000
  GOTO INICIO
END
```

Con esto terminamos la programación del movimiento de los motores.

3.3 PROGRAMACIÓN DEL ALGORITMO PARA LA DETECCIÓN DE LOS SENSORES.

Esta es una de las partes clave en nuestro proyecto, ya que nuestro robot con la información que mandan los sensores debe hacer dos cosas:
Primero.- seguir la línea negra.
Segundo.- detectar el tipo de cruce

En esta parte conviene definir el tipo de laberinto con el que vamos a trabajar, ya que se sabe que existe una gran gama de estos. Sin embargo nosotros trabajaremos con el laberinto clásico rectangular como el mostrado en la figura 3.3.1

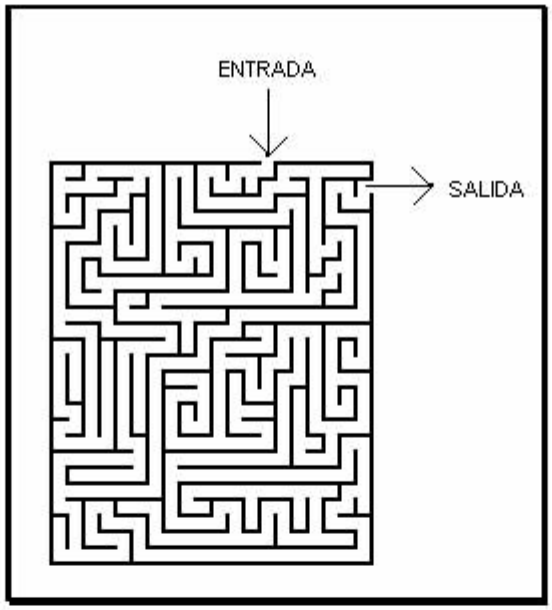


Figura 3.3.1 Laberinto rectangular

En este tipo de laberintos se entra por cualquier lado, y se sale por cualquier lado, la salida y la entrada no deben estar dentro del laberinto, en nuestro caso, nosotros no vamos a ver paredes, lo que vamos a hacer es cambia este tipo de laberinto de paredes por uno equivalente en el que una línea negra sea el camino, como el mostrado en la figura 3.3.2

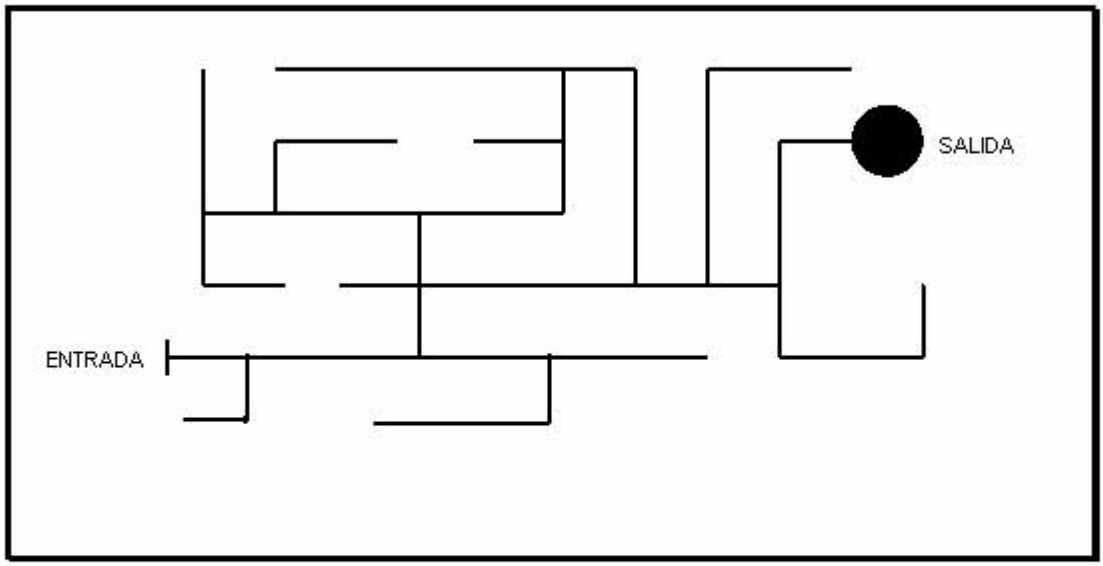


Figura 3.3.3 Ejemplo del tipo de laberinto

En donde no puede haber 2 líneas paralelas amenos de 15 cm. La salida será un círculo negro de 15 cm. De diámetro, se puede entrar o salir en cualquier lado del laberinto y las líneas serán de 6 mm de espesor. Debe ser una superficie plana.

Con esta información podemos proseguir con nuestro algoritmo.

Para esto primero debemos conocer los tipos de cruce que nos podemos encontrar. La figura 3.3.4 nos muestra los 7 tipos de cruce con que nos podemos encontrar.

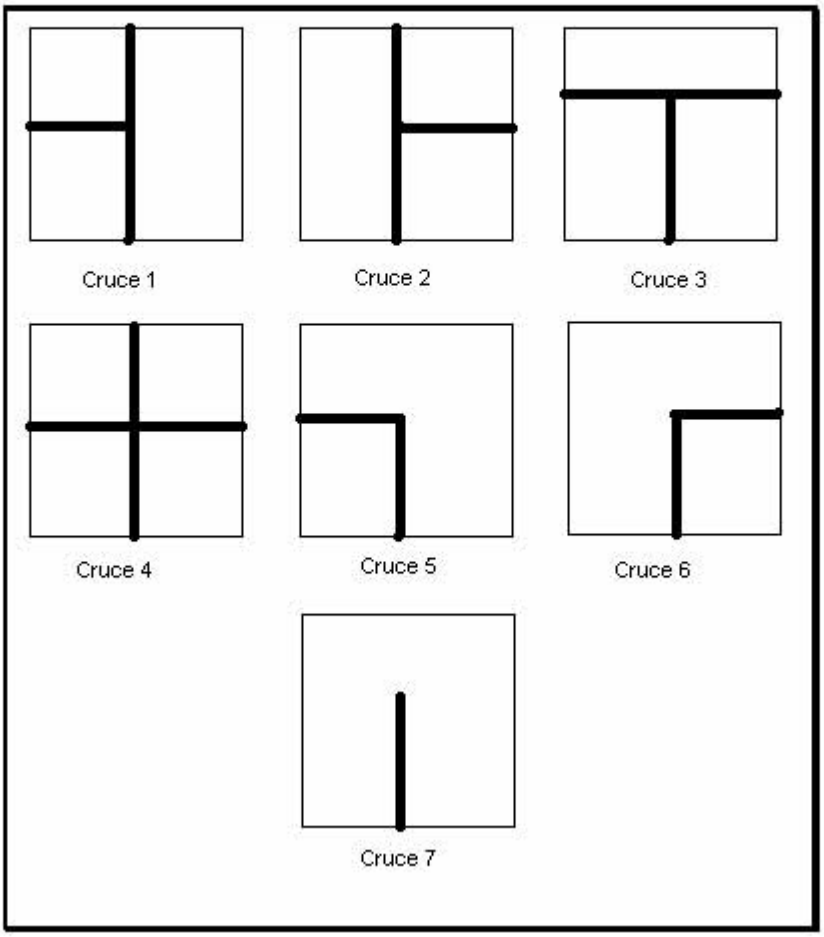


Figura 3.3.4 Tipos de cruce que tienen que detectar los sensores.

Son cuatro cruces de decisión (en donde puede tomar al menos 2 caminos), dos de vuelta únicamente y uno de fin de camino.

Para resolver el problema se propone el arreglo de sensores mostrado en la figura 3.3.5.

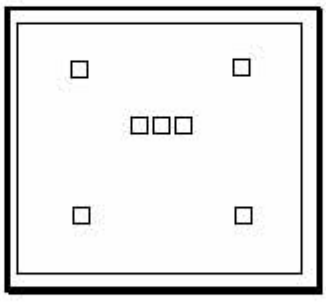


Figura 3.3.5 arreglo de 7 sensores para detectar los tipos de cruce.

Cada sensor lo vamos a llamar con las iniciales de su posición por ejemplo el Frontal Izquierdo en nuestro programa lo llamaremos FI, Frontal Derecho FD, Central Izquierdo CI, Central Central CC, Central Derecho CD, Trasero Izquierdo TI, y Trasero Derecho TD.

Los 3 sensores centrales, son para seguir la línea como si fuera un seguidor normal, los otros 4 tendrán otras funciones explicadas mas adelante.

A continuación explicaremos el proceso para detectar el tipo de cruce, ayudados de las figuras 3.3.6 y 3.3.7.

Primero, siempre supondremos que nuestro robot va de abajo hacia arriba, en cada cruce se presenta una y solo una forma de detectarse. Empezaremos por el cruce 1, cuando nuestro robot va avanzando hacia el cruce, el sensor DI es el primero en identificar la rama izquierda, en este caso encenderemos una bandera, para recordar que fue vista esa rama, en ese momento, nuestro robot sigue avanzando derecho, hasta que el sensor TI detecta la rama izquierda, en este momento al menos uno de nuestros 3 sensores centrales esta viendo línea, además tenemos una bandera de rama izquierda vista, por lo tanto, estas serán, las condiciones para asegurar que el tipo de cruce fue 1.

Para el cruce 2, es similar al tipo de cruce 1, pero en este caso tiene una rama hacia la izquierda.

Tipo de cruce 3, cuando nuestro robot viene avanzando, los 2 sensores delanteros son los primeros en ver ramas, por tanto prendemos las 2 banderas de cruce visto, cuando el robot sigue avanzando, los 3 sensores centrales ven blanco, y en el momento que al menos uno de los sensores traseros ve una rama, estamos listos para decidir que tipo de cruce es, en este caso, los sensores centrales están viendo blanco, y tenemos 2 banderas encendidas, con estas condiciones, podemos asegurar que es un cruce del tipo 3.

Cruce 4, en este caso, nuestros 2 sensores delanteros verán ramas, nuestro robot seguirá avanzando, y en el momento que al menos uno de los sensores

traseros vea una rama, tendremos encendidas las 2 banderas de cruce detectado, además al menos uno de nuestros 3 sensores centrales detectara línea, con esto podemos asegurar que este es un tipo de cruce 4.

Para el tipo 5, tendremos encendida la bandera izquierda y nuestros sensores centrales verán blanco.

Para el tipo 6, tendremos encendida la bandera derecha y nuestros sensores centrales verán blanco.

Para el tipo de cruce 7, que es un fin de camino, no habrá banderas encendidas y todos los sensores verán blanco.

Estas condiciones podemos cotejarlas en las figuras 3.3.6 y 3.3.7.

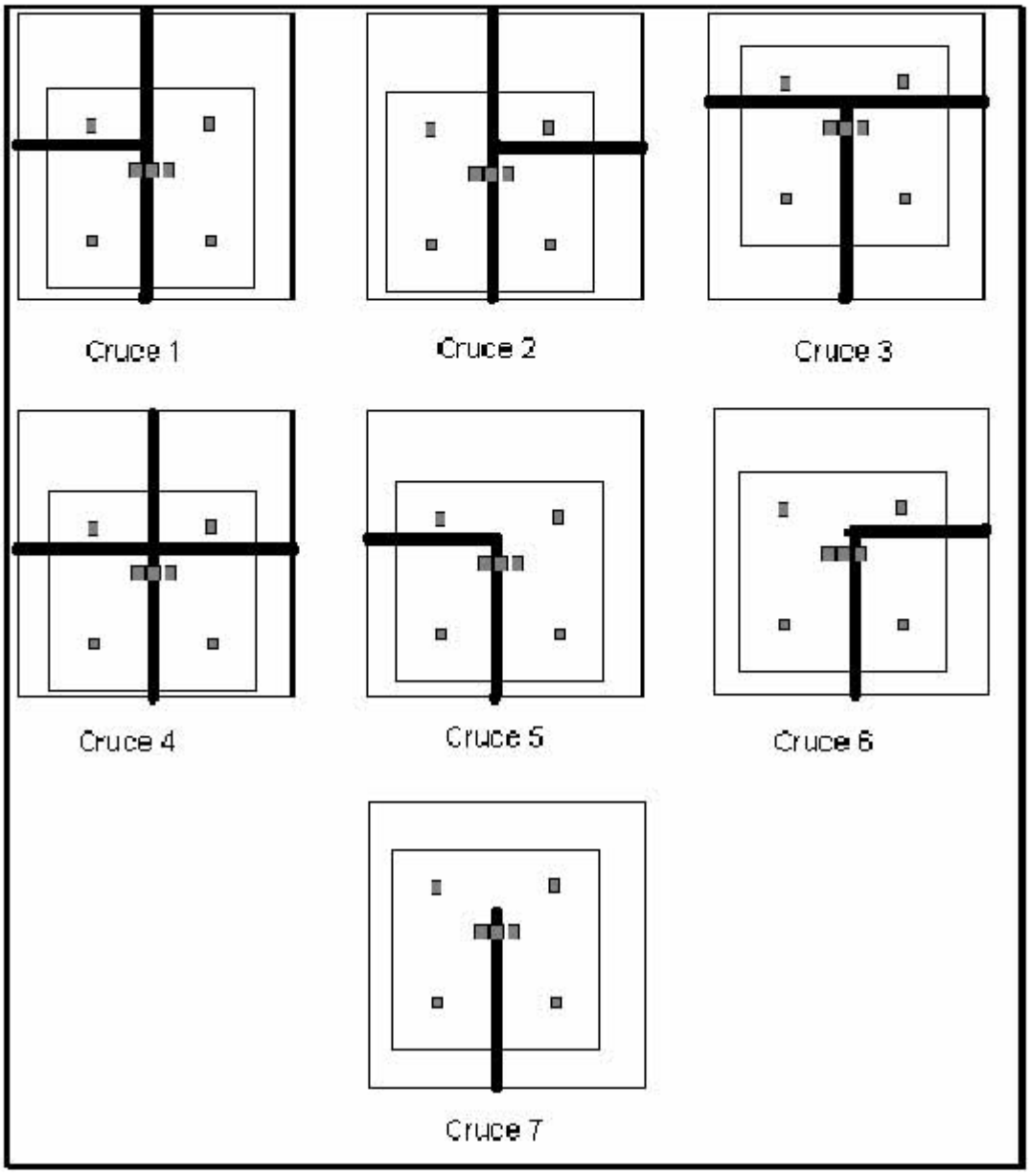


Figura 3.3.6 Primer paso para detectar el tipo de cruce.

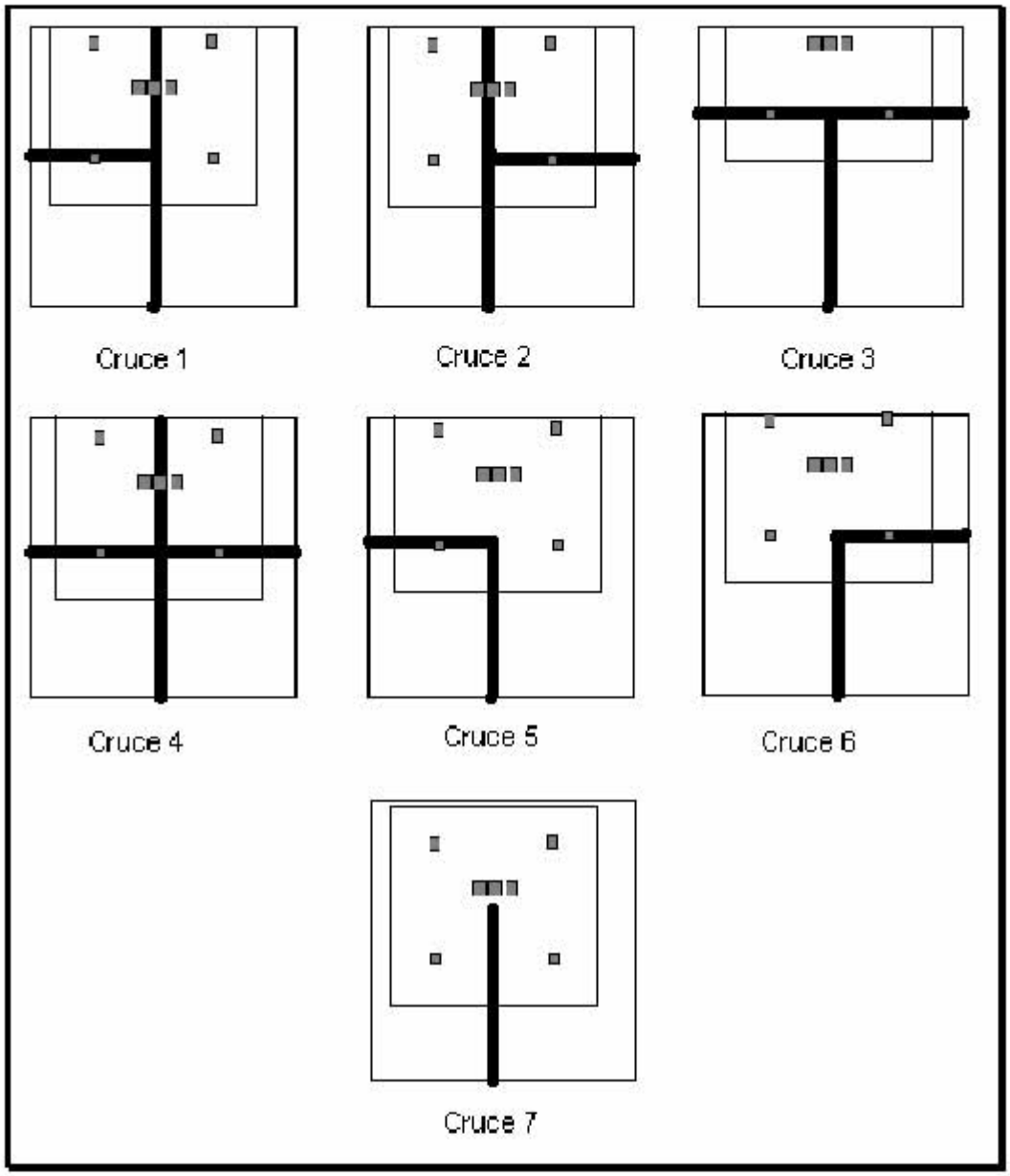


Figura 3.3.7 Segundo paso para detectar el tipo de cruce.

Ya con nuestras condiciones establecidas, podemos completar nuestro circuito, ya que de aquí en adelante debemos tener nuestro circuito completo, lo debemos construir como se muestra en la figura 3.3.8

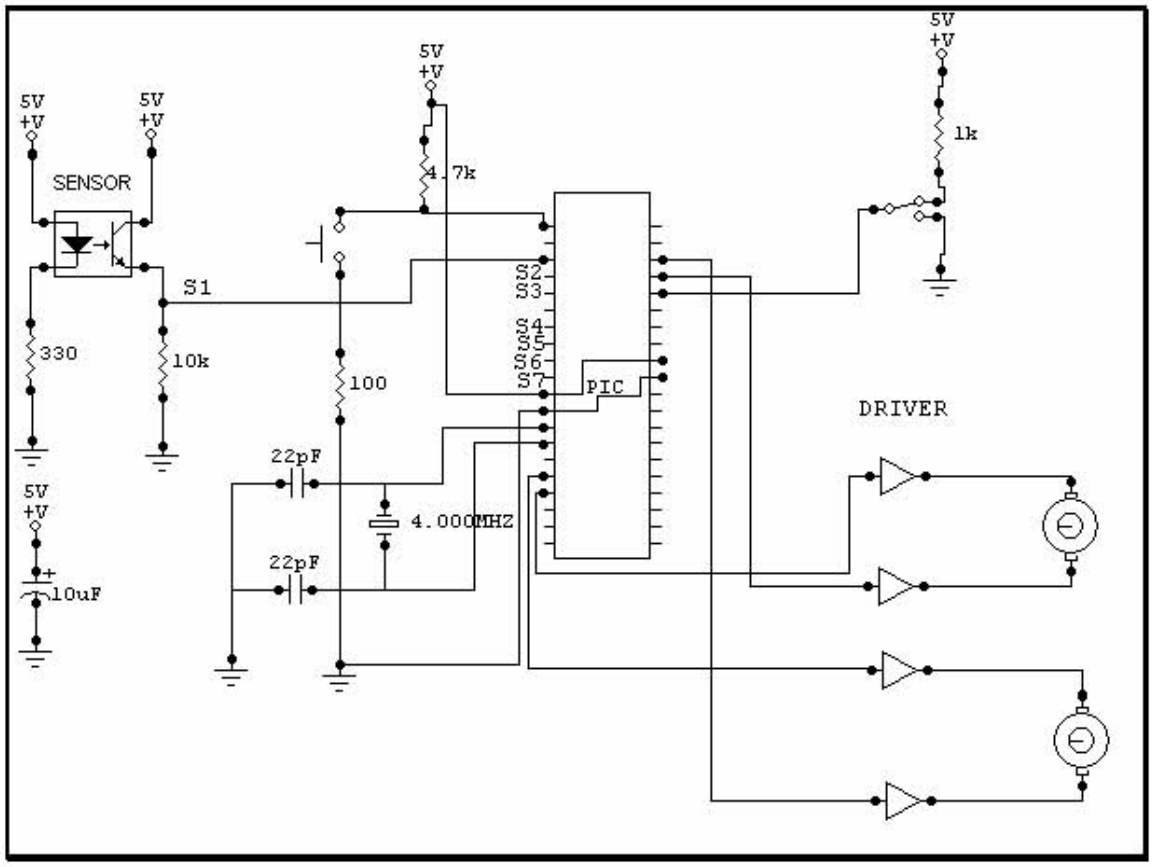


Figura 3.3.8 Circuito completo del robot

De donde las entradas S2 a S7 son circuitos iguales a la entrada S1 en la parte del sensor.

El programa lo haremos en Pic Basic Pro, con el editor del MPLAB IDE vistos en el capitulo anterior.

Para esto se sabe que para los laberintos de este tipo se pueden resolver con la ley de la mano derecha, o izquierda. Esto para cada nodo de decisión se vire hacia la derecha, o hacia la izquierda según sea la regla usada. En este caso utilizaremos el algoritmo de la mano derecha.

```
Include "modedefs.bas"
Include "ICDDEFS.BAS"
        CRUCE       VAR BYTE
        DELD        VAR BIT
        DELI        VAR BIT
```

```
        FI              VAR WORD
        FD              VAR WORD
        CI              VAR WORD
        CC              VAR WORD
        CD              VAR WORD
        TI              VAR WORD
        TD              VAR WORD
Define      ADC_BITS   10
Define      ADC_CLOCK      3
Define      ADC_SAMPLEUS  50
TRISA = %11111111       ' HABILITA TODO PORTA COMO ENTRADA
ADCON1 = %10000000  '  HABILITA  PORTA  COMO  ENTRADA
ANALÓGICA Y JUSTIFICA HACIA LA DERECHA
TRISE = %11111111
TRISB = %00001000
TRISC = %00000000
 OPTION_REG = $7f       ' HABILITAR PULLUPS DE  PORTB
AP=0          'VALOR INICIAL DE APUNTADOR
GOTO INICIO
PARAR:
        Hpwm 2,0,300
        Hpwm 1,0,300
        PORTB.5=0
        PORTB.4=0
        RETURN
DERECHO:
        Hpwm 2,255,300
        Hpwm 1,255,300
        PORTB.5=0
        PORTB.4=0
        RETURN
DERECHA:
        Hpwm 2,255,300
        Hpwm 1,30,300
```

```
        PORTB.5=0
        PORTB.4=0
        RETURN
IZQUIERDA:
        Hpwm 1,255,300
        Hpwm 2,30,300
        PORTB.4=0
        PORTB.5=0
        RETURN
IZQB:
        HPWM 1,127,300
        HPWM 2,127,300
        PORTB.5=0
        PORTB.4=1
        RETURN
DERB:
        HPWM 1,127,300
        HPWM 2,127,300
        PORTB.4=0
        PORTB.5=1
        RETURN
BLANCO:
         DELD=0
         DELI=0
         GOSUB DERB
GIR1:
         ADCIN 4,CC
               IF CC < 500 THEN
                  RETURN
               ENDIF
        GOTO GIR1
GIRDER:
        DELD=0
        DELI=0
```

```
    GOSUB DERB
        PAUSE 300
GIR:
        ADCIN 1,CD
            IF CD < 500 THEN
            GOSUB DERECHO
             RETURN
            ENDIF
        GOTO GIR
        RETURN
GIRIZQ:
        DELD=0
        DELI=0
       GOSUB IZQB
        PAUSE 300
GIR2:
       ADCIN 5,CI
       IF CI < 500 THEN
            GOSUB DERECHO
             RETURN
       ENDIF
       GOTO GIR2
       RETURN
CENTRAR:
CEN:
       ADCIN 4,CC
       IF CC < 300 THEN
             GOSUB DERECHO
       ENDIF
DER:
       ADCIN 5,CI
       IF CI < 300   THEN
            GOSUB DERECHA
       ENDIF
```

```
IZQ:
        ADCIN 1,CD
        IF CD < 300 THEN
                GOSUB IZQUIERDA
        ENDIF
        RETURN
GUARDA:
        WRITE AP,CRUCE
        AP=AP+1
        RETURN
INICIO:
        ADCIN 4,CC
        ADCIN 5,CI
        ADCIN 1,CD
        ' BLANCO
         IF  CI>500 & CC>500 & CD>500 & DELD==0 & DELI=0 THEN
                GOSUB PARAR
                PAUSE 1
                CRUCE=0
                GOSUB GUARDA
                GOSUB BLANCO
         ENDIF
        GOSUB CENTRAR
        ADCIN 3,FD
        IF FD< 700 THEN
                DELD=1
        ENDIF
        ADCIN 2,FI
        IF FI< 700 THEN
                DELI=1
        ENDIF
'CUANDO LLEGO A LA META
        ADCIN 7,TI
        IF FI < 700 & TI< 700 THEN
```

```
            PARO:
                GOSUB PARAR
                GOTO PARO
        ENDIF
        ' CRUCE 3
        ADCIN 6,TD
        ADCIN 7,TI
        IF DELD ==1 & DELI == 1 & CI>500 & CC>500 & CD>500 & ( TD<700
OR TI < 700 ) THEN
            CRUCE=3 'CRUCE 3
            GOSUB GUARDA
            GOSUB GIRDER
        ENDIF
        ' CRUCE 1
        IF DELD ==0 & DELI == 1 &( CI<500 OR CC<500 OR CD<500) & (
TD<700 OR TI < 700 ) THEN
            CRUCE=1 'CRUCE 1
            GOSUB GUARDA
            GOSUB CENTRAR
            DELI=0
            DELD=0
        ENDIF

        ' CRUCE 2
        IF DELD ==1 & DELI == 0 & (CI<500 OR CC<500 OR CD<500 )& (
TD<700 OR TI < 700 ) THEN
            CRUCE=2 'CRUCE 2
            GOSUB GUARDA
            GOSUB GIRDER
        ENDIF
        'CRUCE 4
        IF DELD ==1 & DELI == 1 & (CI<500 OR CC<500 OR CD<500) & (
TD<700 OR TI < 700 ) THEN
            CRUCE=4 'CRUCE 4
```

```
            GOSUB GUARDA
            GOSUB GIRDER
        ENDIF
        'CRUCE 6
        ADCIN 6,TD
IF TD < 700 & DELD==1 & DELI=0 & CI>500 & CC>500 & CD>500   THEN
            GOSUB GIRDER
ENDIF
'CRUCE 5
IF DELI==1 & DELD== 0 & CI>500 & CC>500 & CD>500 THEN
        GOSUB DERECHO
MANTEN:
            ADCIN 7,TI
            IF TI<700 THEN
                GOSUB GIRIZQ
                GOTO INICIO
            ENDIF
         GOTO MANTEN
ENDIF
GOTO INICIO
END         'FIN DEL PROGRAMA
```

En este programa aparte de reconocer los tipos de cruce esta recorriendo el laberinto con la ley de la mano derecha y guardando los nodos de decisión en la memoria EEPROM.

3.4 EVALUACIÓN Y PRUEBA DE LA TRANSMISIÓN MECÁNICA CON SUS RESPECTIVOS DRIVERS.

Aquí es donde tenemos que hacer nuestras primeras pruebas, para esto debemos tener lista nuestra pista de pruebas.

Para cuestiones de pruebas diseñaremos un laberinto como el mostrado en la figura 3.4.1

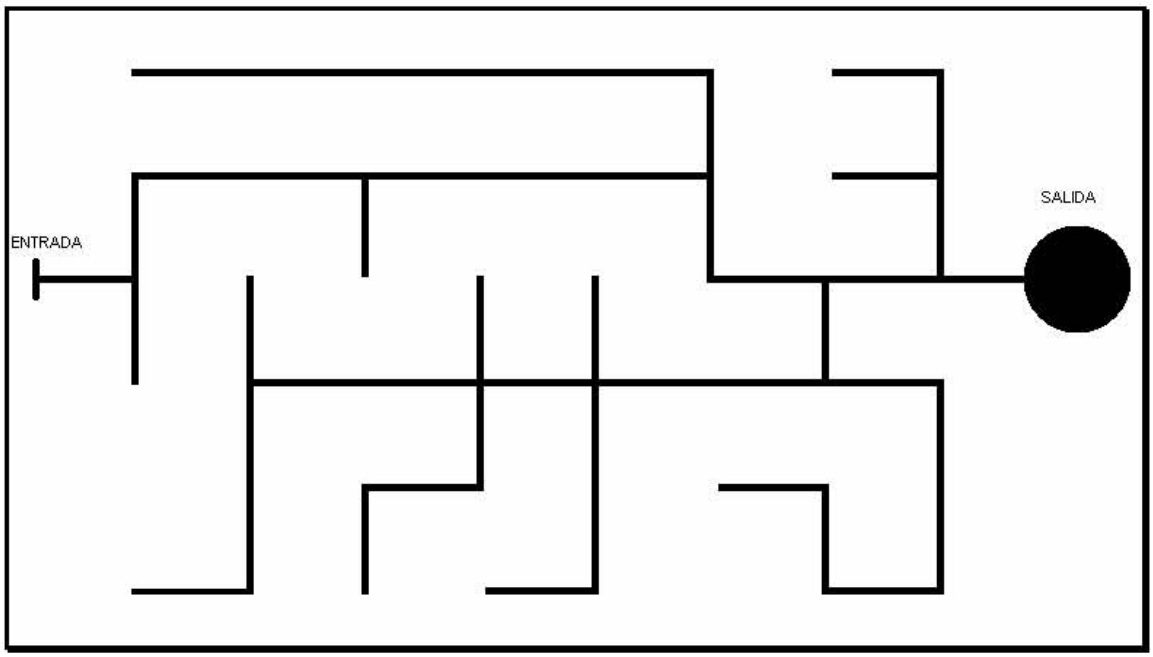


Figura 3.4.1 Laberinto de pruebas

En esta parte del proceso de construcción, es donde debemos de hacer los ajustes necesarios para que nuestro robot siga la línea lo mas fielmente posible y de los giros a 90° y 180°, a una velocidad controlable, con esto me refiero a que cuando de un giro de 180° se pare cuando el sensor CC este sobre la línea, y no generar mucha inercia con la velocidad de giro.

En este caso los valores para el PWM que controla los motores son los siguientes:

En derecho la velocidad máxima, para estos motores, en caso de usar otro tipo de motores más rápidos, estos valores pueden ser modificados, también para los otros tipos de movimiento.

DERECHO:

```
Hpwm 2,255,300
Hpwm 1,255,300
PORTB.5=0
PORTB.4=0
RETURN
```

En derecha, los valores óptimos para ir compensando cuando va perdiendo la línea son los siguientes, para el caso de izquierda son similares, solo los valores se invierten de canal PWM.

DERECHA:

```
Hpwm 2,255,300
Hpwm 1,30,300
PORTB.5=0
PORTB.4=0
RETURN
```

IZQUIERDA:

```
Hpwm 1,255,300
Hpwm 2,30,300
PORTB.4=0
PORTB.5=0
RETURN
```

Los valores para IZQB y DERB los redujimos a la mitad de la velocidad máxima, esto es debido a que en el momento que giraba 180° no se podía detener centrado en la línea, estos valores pueden ser modificados para otro tipo de motores.

IZQB:

```
HPWM 1,127,300
```

```
        HPWM 2,127,300
        PORTB.5=0
        PORTB.4=1
        RETURN
DERB:
        HPWM 1,127,300
        HPWM 2,127,300
        PORTB.4=0
        PORTB.5=1
        RETURN
```

Una vez que los valores de PWM de los motores fueron ajustados (en este caso los valores que ya teníamos eran los óptimos, pero es importante saber de donde podemos hacer este tipo de ajustes), debemos de ajustar nuestros sensores ópticos para que podamos tener lecturas de los tipos de cruce adecuados.

Se propone un pequeño programa para calibrar los sensores, ya que la mayoría de los sensores no responden igual a los estímulos luminosos.

```
'SE LEEN LOS SENSORES
CALIBRA:
ADCIN 4,CC
IF CC>500 THEN
        PORTB.0=1
ELSE
        PORTB.0=0
ENDIF
GOTO CALIBRA
END
```

Este programa lo incluimos en nuestro programa principal para calibrar los sensores, y necesitamos conectar un led a la salida de PORTB en el pin 0 y el valor que vamos a modificar es aquel con el que se compara, para esto se

programa nuestro pic en el robot, y se desplaza el robot de una parte blanca, en la cual debe de estar prendido el led, hacia la línea, que es donde el led se debe de apagar, en caso de que no se apague, tenemos que subir el valor de comparación, esto es para que el umbral quede mas arriba y con poco decremento de luz se apague nuestro led.

Es importante hacer esta prueba con todos los sensores, ya que la ubicación de cada uno también es un factor que afecta, por ejemplo los sensores centrales están juntos, y esto provoca que la luz que emite el del medio rebote también el los dos de las orillas, por tanto entre los 3 la cantidad de luz infrarroja emitida es mas alta y con esto el umbral de los sensores también cambia.

En este caso los umbrales óptimos que se obtuvieron son para los sensores centrales de 700 y para los demás de 500.

Una vez hecha nuestra calibración cuidadosamente, y verificando que nuestros sensores funcionan correctamente (esta etapa es de las que consumen mas tiempo), podemos empezar a hacer la prueba, colocando el robot en la línea de entrada y ver que llegue a la salida, si nuestros sensores quedaron bien calibrados los valores que nuestro robot guarde en la memoria EEPROM después de recorrer el laberinto serán los siguientes: 3, 0, 1, 2, 0, 3, 3, 2, 3, 4, 0, 4, 4, 0, 4, 3, 0, 1, 0, 2, 4, 0, 4, 4, 0, 4, 1, 0, 2, 3 y 1. Cada uno en su respectiva localidad de memoria.

3.5 PROGRAMACIÓN DEL ALGORITMO DE AUTO APRENDIZAJE DEL ROBOT.

En esta etapa construiremos un algoritmo que nos permita optimizar la forma de recorrer el laberinto, pues como ya sabemos para salir de un laberinto de este tipo basta con virar hacia la derecha en todos los nodos de decisión (o a la izquierda), sin embargo con este algoritmo debemos eliminar todos los caminos cerrados para que nuestro robot salga mas rápido, una vez que ya se haya recorrido el laberinto por primera vez. Como se ha visto en el capitulo 3.4 nuestro programa hasta el momento identifica el tipo de cruce y guarda los nodos de decisión en la memoria EEPROM del PIC, o sea que debemos de empezar a trabajar con los datos almacenados en memoria. En la siguiente tabla podemos apreciar los datos de la memoria EEPROM, en la primera columna podemos ver la dirección de memoria, y en la segunda columna tenemos el valor de la localidad de memoria (en decimal).

Primera etapa: Al observar los datos con cuidado es evidente que los valores siguientes a un cero (fin de camino), son los valores vistos de regreso del valor anterior al cero, por lo tanto nos conviene eliminar todos los valores después de un cero, y este será el paso 1 de nuestro algoritmo.
Segunda etapa: en esta segunda etapa tendremos que eliminar ramas e ir desplazando los ceros para las ramas anteriores, en esta caso cuando nuestro tipo de cruce es 4 tendremos 3 posibles caminos, para eliminarlo tendremos que tener tres ceros después de un 4. En cambio los cruces 1, 2 y 3 tienen que tener únicamente 2 ceros para poder eliminarlos, por lo tanto se elimina el numero anterior a los ceros, los ceros y el siguiente a los 2 ceros, quedándonos la pila de datos como se muestra en la columna 3.

Este proceso lo tenemos que repetir hasta que ya no haya secuencias de 2 ceros o 3 ceros juntos, con esto se concluye la etapa 2.

Tercera etapa: consiste en dar la dirección que debe tomar a el tipo de cruce, y tenemos 2 casos, para los tipos de cruce 1, 2 y 3 cuando el valor que sigue es

un cero, la dirección será izquierda, en caso de que el valor siguiente sea diferente de cero, la dirección será derecha. Para el tipo de cruce 4 cuando no hay ceros después de el, la dirección será derecha, en el caso de que tenga 1 cero junto, la dirección será seguir derecho, y por ultimo cuando tenga 2 ceros juntos la dirección será izquierda.

DIRECCION EEPROM	VALOR INICIAL	PASO 1	PASO 2	PASO 3	PASO 4	PASO 5	PASO FINAL
0x00	3	3	3	3	3	3	3 I
0x01	0	0	0	0	0	0	2 I
0x02	1	2	2	2	2	2	3 D
0x03	2	0	0	0	0	0	2 I
0x04	0	3	3	3	3	3	1 D
0x05	3	2	2	2	2	2	
0x06	3	3	3	3	3	0	
0x07	2	4	4	4	0	1	
0x08	3	0	0	0	0		
0x09	4	4	4	0	3		
0x0A	0	0	0	0	1		
0x0B	4	3	0	1			
0x0C	4	0	0	0			
0x0D	0	0	4	3			
0x0E	4	4	0	1			
0x0F	3	0	1				
0x10	0	4	0				
0x11	1	0	3				
0x12	0	1	1				
0x13	2	0					
0x14	4	3					
0x15	0	1					
0x16	4						
0x17	4						
0x18	0						

0x19	4						
0x1A	1						
0x1B	0						
0x1C	2						
0x1D	3						
0x1E	1						

Para entender mejor el desarrollo del algoritmo podemos trasladarnos a nuestro laberinto de prueba y ejecutar el después del paso 1, en la figura 3.5.1 podemos ver el laberinto de prueba, después del paso 1, el cual consistió en eliminar los caminos de regreso menos anidados, ya que los regresos anidados se eliminaran en los siguientes pasos.

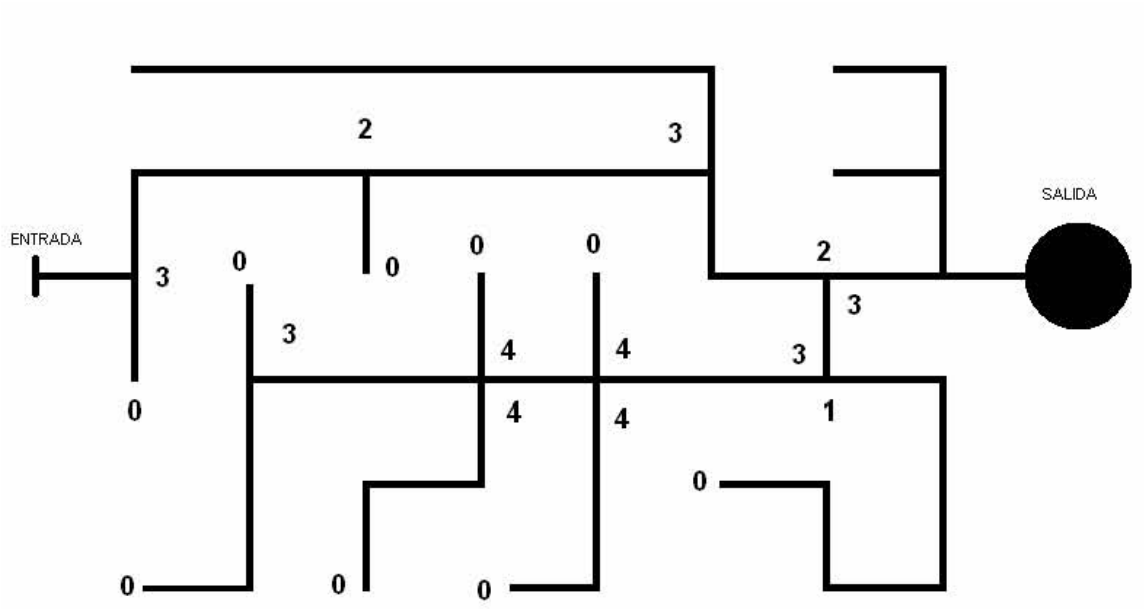


Figura 3.5.1 laberinto de pruebas con los números de en el paso 1.

En el siguiente paso lo que se esta haciendo es eliminar la rama menos anidada, en este caso fue la de el 3 con sus 2 ceros, y el laberinto queda como el mostrado en la figura 3.5.2.

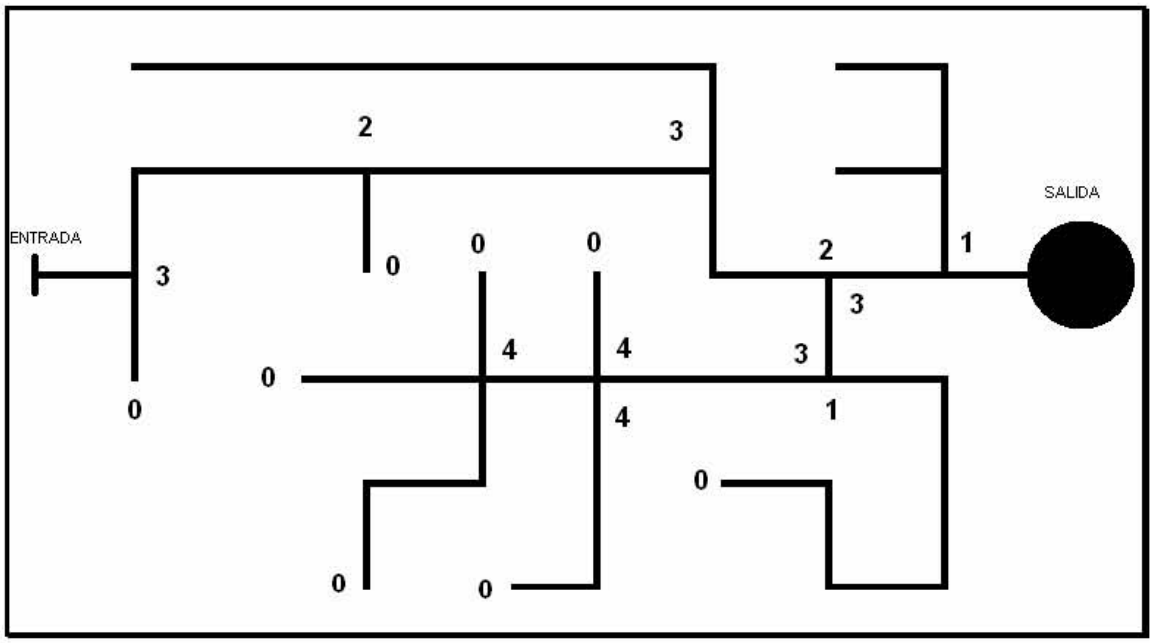


Figura 3.5.2 laberinto en el paso 2

Para el paso 3 se esta eliminando un cruce 4 con sus 3 ceros y un cero se esta trasladando hasta el cruce anterior. Quedando como se muestra en la figura 3.5.3.

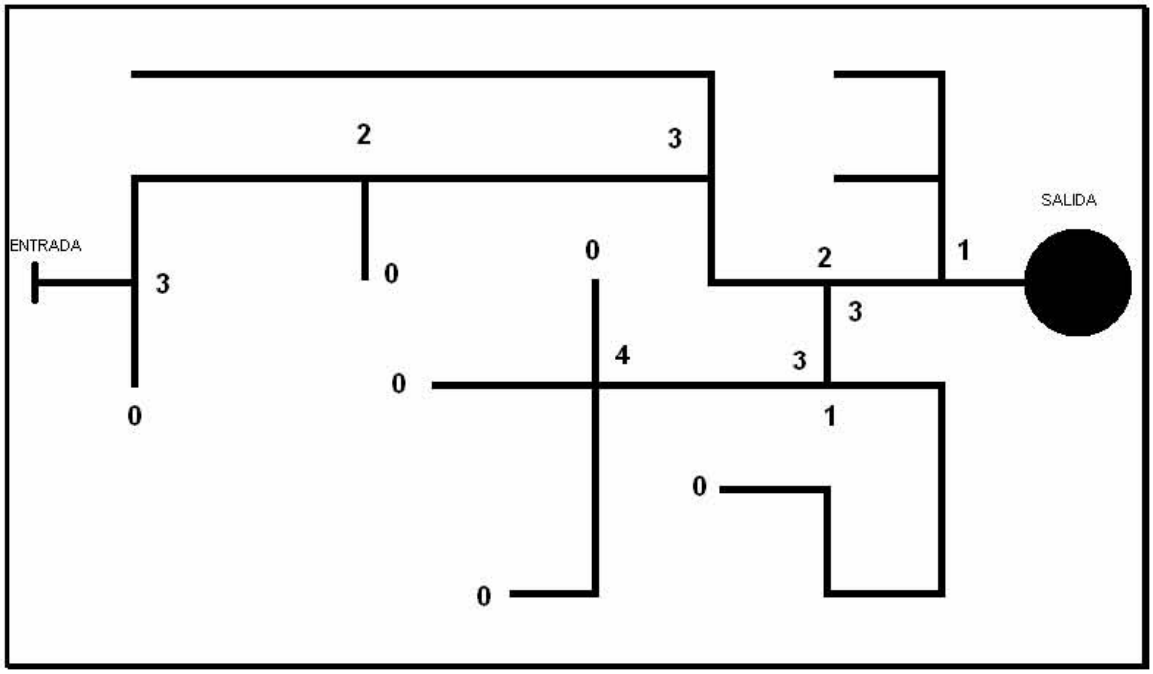


Figura 3.5.3 laberinto en el paso 3

Para el paso 4 tenemos el siguiente 4 eliminado y trasladamos un cero hasta el cruce anterior en este caso fue un 3 quedando como la figura 3.5.4

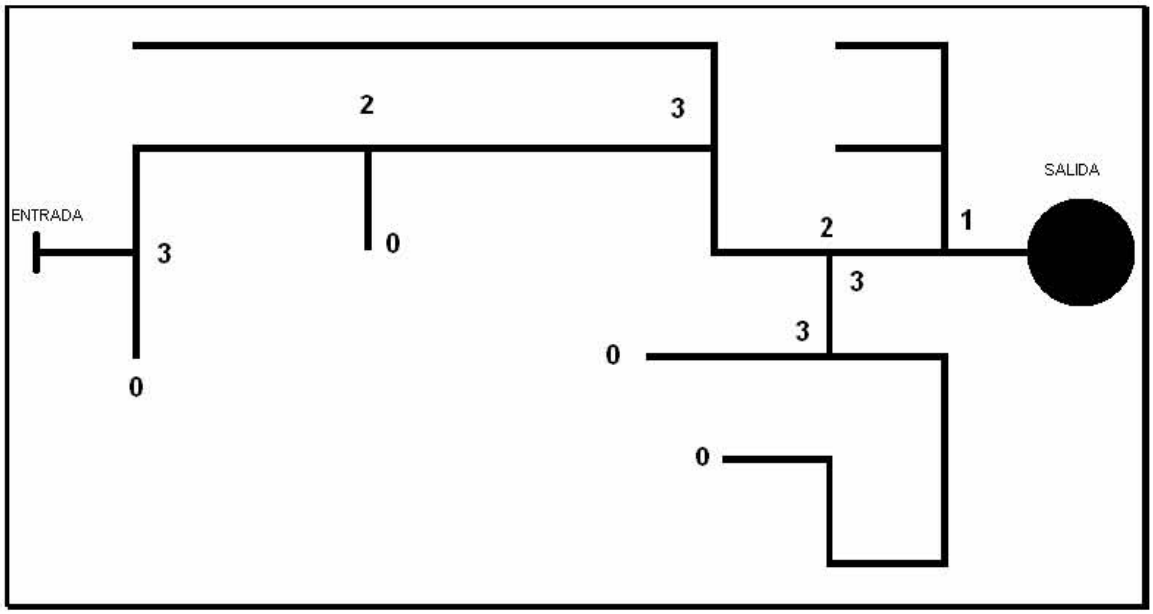


Figura 3.5.4 laberinto en el paso 4

Para el paso 5 es similar al anterior

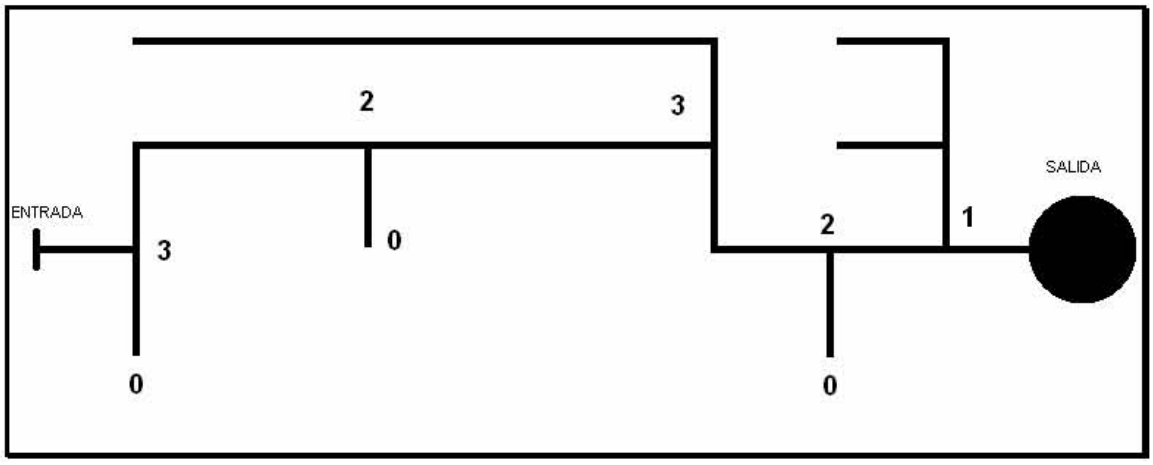


Figura 3.5.5 laberinto en el paso 5

Para el ultimo cuando ya no hubo ramas para eliminar, lo que se tiene que hacer es asignar la dirección a cada cruce, en la figura 3.5.6 se muestran los cruces con la dirección no valida eliminada, quedando la ruta resultante.

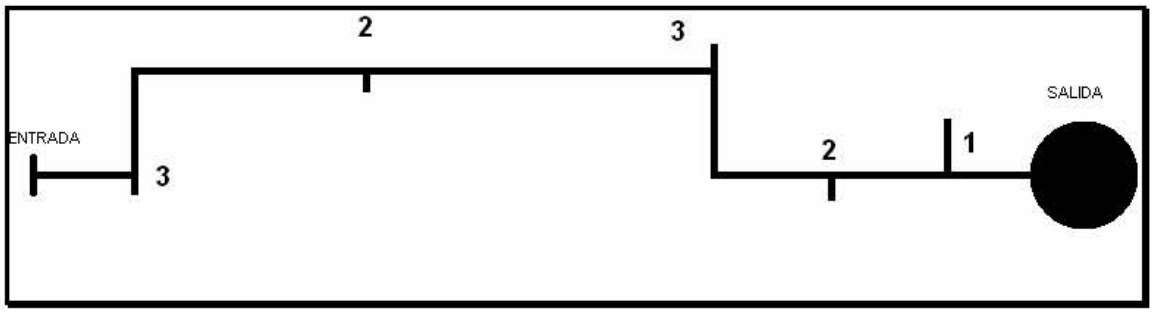


Figura 3.5.6 laberinto simplificado.

Una vez entendido el algoritmo, podemos disponernos a programarlo en PIC BASIC PRO, en este caso para facilidad de programación, utilizaremos apuntadores AP, AP2 y AP3.

En esta sección únicamente se muestra el código del algoritmo desde la etiqueta aprende, todas las variables nuevas se deben de declarar en el encabezado de nuestro programa principal. **APRENDE** solo será una subrutina que será llamada al final de que nuestro robot recorra el laberinto.

En principio, nuestra etiqueta **APRENDE:** empezando con la variable FIN que será igual a AP, ya que AP tiene el valor de la dirección del ultimo dato guardado.

```
'APRENDE APRENDE APRENDE APRENDE APRENDE APRENDE
APRENDE:
 FIN=AP
 AP=0          'EL BARRIDO EMPIEZA DESDE CERO
 AP2=0               'ES EL APUNTADOR A LA NUEVA MEMORIA
 AP3=0               ‘SIMPLIFICADA. TODOS LOS APUNTADORES
                     ‘DEBEN ESTAR EN CERO.
‘EN ESTA ETAPA LO ÚNICO QUE HACE ES ELIMINAR EL DATO
‘SIGUIENTE A UN CERO
‘ESTE  ES EL PRIMER PASO, EL APUNTADOR AP ES EL QUE ESTA
‘LEYENDO LOS DATOS DE LA MEMORIA EEPROM, CUANDO EL DATO
‘QUE  LEE AP ES DIFERENTE DE CERO SIMPLEMENTE LO ESCRIBE EN
‘LA DIRECCION DE MEMORIA QUE TENGA AP2 E INCREMENTA LOS 2
‘APUNTADORES.  SI EL DATO LEÍDO POR AP ES IGUAL A CERO,
```

```
'ENTONCES GUARDA EL CERO EN LA DIRECCION DE AP2,  AP2 BRINCA
'UNA LOCALIDAD Y AMBOS  APUNTADORES SE INCREMENTAN EN 1.
'ESTE PROCESO SE REPITE HASTA QUE AP LLEGA A EL ULTIMO DATO
'ESCRITO EN LA EEPROM. AP3 EN ESTE CASO NOS GUARDA LA
'DIRECCION DE EL ULTIMO DATO ESCRITO, PARA SER USADO EN
'NUESTRA SIGUIENTE  ETAPA.
WHILE AP<=FIN
        AP3=AP2
        READ AP,REAL
        IF REAL<> 0 THEN
                WRITE AP2,REAL
                AP2=AP2+1
                AP=AP+1
        ELSE
                WRITE AP2,REAL
                AP2=AP2+1
                AP=AP+2
        ENDIF
WEND

'ESTA ETAPA SE ENCARGA DE ELIMINAR TODOS LOS DATOS
'ESCRITOS EN LA EEPROM QUE SE LOCALICEN DESPUÉS DE LA
NUEVA 'DIRECCION DEL TOPE
FOR I=AP3+1 TO FIN
WRITE I,255
NEXT

'SE VUELVE A CARGAR EL NUEVO VALOR DE FIN
FIN=AP3

REPITE:
AP3=0
AP2=0
AP=0
```

```
FOR I=0 TO FIN
ELIMINA:

' EN ESTA PARTE , SE LEEN LOS VALORES DE AP Y LOS 3 SIGUIENTES
        READ AP,ACTUAL
        READ AP+1,SIGUIENTE1
        READ AP+2,SIGUIENTE2
        READ AP+3,SIGUIENTE3


'EN ESTA PARTE TENEMOS 2 CASOS, QUE  EL VALOR DEL TIPO DE
'CRUCE SEA 4 Y LOS SIGUIENTES 3 SEAN CERO (ESTO PARA ELIMINAR
'UN CUATRO), O QUE EL VALOR DE CRUCE SEA DIFERENTE DE 4 Y LOS
'2 SIGUIENTES SEAN CERO, ESTO PARA LOS TIPOS DE CRUCE 1, 2 Y 3.
'EN ESTE CASO AP3 ESTA FUNCIONANDO COMO BANDERA, YA QUE EN
'ESTA ETAPA SE HACE UN BARRIDO A LA EEPROM, BUSCANDO
RAMAS 'PARA DESCARTAR, PERO CUANDO YA NO HAY RAMAS PARA
'DESCARTAR, ENTONCES AP3 SE QUEDA EN CERO

        IF                          (ACTUAL==4)                     &&
(SIGUIENTE1==0)&&(SIGUIENTE2==0)&&(SIGUIENTE3==0) THEN
                WRITE AP2,0
                AP=AP+5
                AP2=AP2+1
                AP3=1
                GOTO ELIMINA
        ENDIF
        IF (ACTUAL<>4)&&(SIGUIENTE1==0)&&(SIGUIENTE2==0) THEN
                WRITE AP2,0
                AP=AP+4
                AP2=AP2+1
                AP3=1
                GOTO ELIMINA
        ENDIF
```

```
WRITE AP2,ACTUAL
AP=AP+1
AP2=AP2+1
NEXT I

'AQUÍ SE PREGUNTA CUANDO TODAVÍA HAY RAMAS PARA
'DESCARTAR, SE MANDA A HACER OTRO BARRIDO DE LA EEPROM

IF AP3==1 THEN
        GOTO REPITE
ENDIF
AP=0
AP2=0
'POR ULTIMO SE LE ASIGNA LA DIRECCION A CADA CRUCE
'RESTANTE, Y TENEMOS 3 CASOS: CUANDO EL VALOR QUE SIGUE ES
'DIFERENTE DE CERO, CUANDO EL VALOR SIGUIENTE ES CERO, Y POR
'ULTIMO CUANDO TENEMOS UN 4 Y LOS 2 VALORES SIGUIENTES SON
'CERO, EN TODOS LOS CASOS LO QUE HACEMOS ES BORRAR EL/LOS
'CERO(S) Y ENCENDER EN EL MISMO VALOR DEL TIPO DE CRUCE UNA
'BANDERA QUE SIGNIFICA QUE EN ESE CRUCE LA DIRECCION ES
'HACIA LA IZQUIERDA (PARA LOS CRUCES 1, 2 Y 3) , Y PARA LOS DEL
'TIPO 4 PUEDE TENER 3 OPCIONES: DERECHA (CUANDO NO HAY
'CEROS AL LADO DE EL), DERECHO (CUANDO HAY UN CERO  A SU
'LADO) E IZQUIERDA ( EN EL CASO DE QUE EXISTAN 2 CEROS A SU
'LADO).

DIRECCON:
FOR I=0 TO FIN
READ AP,REAL
READ AP+1,SIGUIENTE1
READ AP+2,SIGUIENTE2
IF  SIGUIENTE1==0 & SIGUIENTE2<>0 THEN
        REAL.4=1
        WRITE AP2,REAL
```

```
        AP=AP+2
        AP2=AP2+1
        GOTO BRINCA
ENDIF
IF REAL==4 & SIGUIENTE1==0 & SIGUIENTE2==0 THEN
        REAL.4=1
        REAL.5=1
        WRITE AP2,REAL
        AP=AP+3
        AP2=AP2+1
        GOTO BRINCA
ENDIF
WRITE AP2,REAL
AP=AP+1
AP2=AP2+1
BRINCA:
NEXT I
RETURN
'AQUÍ TERMINA NUESTRA SUBRUTINA "APRENDE",
```

Y con esto terminamos de programar nuestra rutina para que nuestro robot elimine los caminos cerrados del laberinto.

3.6 INTEGRACIÓN, PRUEBAS Y DEPURACIÓN DEL PROTOTIPO.

Para esta parte, ya esta construida la mayor parte de nuestro prototipo, y nada mas nos falta integrar en nuestro programa, la rutina de "**APRENDE:"** y es bastante sencillo. Únicamente se tienen que declarar las variables nuevas que usamos en el encabezado del programa, como se muestra a continuación:

```
Include "modedefs.bas"
Include "ICDDEFS.BAS"
        REAL VAR BYTE
        FIN VAR BYTE
        AP2 VAR BYTE
        AP3 VAR BYTE
        SIGUIENTE1 VAR BYTE
        SIGUIENTE2 VAR BYTE
        SIGUIENTE3 VAR BYTE
        AP VAR BYTE
        CRUCE VAR BYTE
        ACTUAL VAR BYTE
        POSTERIOR VAR BYTE
        DELD VAR BIT
        DELI VAR BIT
        I VAR  BYTE
        CASO VAR BYTE
        CERO VAR BIT
        FI VAR WORD
        FD VAR WORD
        CI VAR WORD
        CC VAR WORD
        CD VAR WORD
        TI VAR WORD
        TD VAR WORD
        Define      ADC_BITS   10      ' Set number of bits in result
```

```
        Define      ADC_CLOCK      3       ' Set clock source (3=rc)
        Define      ADC_SAMPLEUS  50       ' Set sampling time in uS
        TRISA = %11111111        ' Set PORTA to all input
        ADCON1 = %10000000    ' Set PORTA analog and right justify result
        TRISE = %11111111
        TRISB = %00001000
        TRISC = %00000000
    OPTION_REG = $7f        ' Enable PORTB pullups
        PORTB=0
        AP=0          'VALOR INICIAL DE APUNTADOR
        DELI=0
        DELD=0
```

E insertar el programa “APRENDE:” en una subrutina que la llamaremos una ves que el robot llego a la meta.

Pero por ultimo nos falta una parte del programa muy importante que es la forma en la que nuestro robot reproducirá el camino simplificado.

A continuación se muestra el programa para reproducir el camino una vez que se ejecuto la subrutina “APRENDE:” el cual llamamos “REPRODUCIR:”

```
 REPRODUCIR:
AP=0
RELOAD:
ADCIN 4,CC
ADCIN 5,CI
ADCIN 1,CD
ADCIN 3,FD
IF FD< 700 THEN
        DELD=1
ENDIF
ADCIN 2,FI
IF FI< 700 THEN
        DELI=1
ENDIF
```

```
        ' BLANCO
IF  CI>500 & CC>500 & CD>500 & DELD==0 & DELI=0 THEN
        GOSUB PARAR
        PAUSE 1
   GOSUB BLANCO
ENDIF
GOSUB CENTRAR
        'CUANDO LLEGO A LA META
ADCIN 7,TI
IF FI < 700 & TI< 700 THEN
PARO3:
        GOSUB PARAR
        GOTO PARO3
ENDIF
        ' CRUCE T
ADCIN 6,TD
ADCIN 7,TI
IF DELD ==1 & DELI == 1 & CI>500 & CC>500 & CD>500 & ( TD<700 OR TI
< 700 ) THEN
        GOSUB TABLA3
ENDIF
        ' TIZQ
IF DELD ==0 & DELI == 1 &( CI<500 OR CC<500 OR CD<500) & ( TD<700
OR TI < 700 ) THEN
        GOSUB TABLA1
ENDIF

        ' TDER
IF DELD ==1 & DELI == 0 & (CI<500 OR CC<500 OR CD<500 )& ( TD<700
OR TI < 700 ) THEN
        GOSUB TABLA2
ENDIF
        'CRUZ
```

```
IF DELD ==1 & DELI == 1 & (CI<500 OR CC<500 OR CD<500) & ( TD<700
OR TI < 700 ) THEN
        GOSUB TABLA4
ENDIF
        'cruce L DER
ADCIN 6,TD
IF  DELD==1 & DELI=0 & CI>500 & CC>500 & CD>500   THEN
        GOSUB DERECHO
MANTEN4:
        ADCIN  6,TD
        IF TD<700 THEN
                GOSUB GIRDER
                GOTO RELOAD
        ENDIF
        GOTO MANTEN4
ENDIF
        'CRUCE L IZQ
IF DELI==1 & DELD== 0 & CI>500 & CC>500 & CD>500 THEN
        GOSUB DERECHO
MANTEN3:
        ADCIN 7,TI
        IF TI<700 THEN
                GOSUB GIRIZQ
                GOTO RELOAD
        ENDIF
        GOTO MANTEN3
ENDIF
GOTO RELOAD

TABLA1:
READ AP,REAL
IF REAL.4=1 THEN
        GOSUB GIRIZQ
        AP=AP+1
```

```
        RETURN
ENDIF
GOSUB CENTRAR
PAUSE 200
DELI=0
DELD=0
AP=AP+1
RETURN

TABLA2:
READ AP,REAL
IF REAL.4=0 THEN
        GOSUB GIRDER
        AP=AP+1
        RETURN
ENDIF
GOSUB CENTRAR
PAUSE 200
DELD=0
DELI=0
AP=AP+1
RETURN
TABLA3:
READ AP,REAL
IF REAL.4=0 THEN
        GOSUB GIRDER
        AP=AP+1
        RETURN
ENDIF
        GOSUB GIRIZQ
        AP=AP+1
RETURN
TABLA4:
READ AP,REAL
```

```
IF REAL.4=0 THEN
        GOSUB GIRDER
        AP=AP+1
        RETURN
ENDIF
IF REAL.5=0 THEN
        GOSUB CENTRAR
        DELD=0
        DELI=0
        AP=AP+1
        RETURN
ENDIF
GOSUB GIRIZQ
AP=AP+1
RETURN
```

Esta rutina la llamaremos activando un switch que conectaremos en el PUERTO B en el bit 3, por lo tanto, para que nuestro robot funcione correctamente, debemos poner en 1 el bit 3 del puerto b para que reconozca el laberinto, posteriormente el robot se debe poner en la entrada nuevamente, y poner a cero el bit 3 del puerto b para que se ejecute la rutina REPRODUCIR:.
En el apéndice A se muestra el programa completo final.

CONCLUSIONES

CONCLUSIONES

Se diseño un robot capaz de recorrer un laberinto pintado con líneas negras sobre una superficie de color blanco, el laberinto debe tener en un lado la entrada y en otro la salida, la salida no debe estar dentro del laberinto.

Como sistema de visión del robot tenemos sensores del tipo reflectivo los cuales entregan señales analógicas a nuestro sistema de control.

El robot cuenta con un sistema de control basado en un microcontrolador PIC16F877 de tecnología RISC ya que por sus características en precio, facilidad de programar, disponibilidad en el mercado y el acceso a compiladores como PIC BASIC PRO permitieron explotar más fácilmente sus capacidades.

En la parte de potencia contamos con un driver L298 ya que es uno de los más comunes para este tipo de aplicaciones, y se tiene acceso fácilmente en el mercado mexicano.

Se desarrollaron algoritmos para el recorrido del laberinto, para guardar el camino en la memoria EEPROM, para discriminación de caminos falsos y por ultimo para ejecutar el laberinto de principio a fin evitando pasar por caminos falsos.

Se eligieron los motores mas eficientes con tren de engranes incluido, esto para reducir el consumo de batería y optimizar el consumo de corriente.

Se hicieron pruebas con diferentes estructuras mecánicas y se Eligio la mejor, en cuanto a menor oscilación, mas ligera y facilidad de manejo.

Este diseño es bastante económico, ya que el diseño tanto mecánico como electrónico es bastante reducido y todos los componentes se encuentran en el mercado nacional.

BIBLIOGRAFÍA

Referencias Electrónicas

[1] WIKIPEDIA La enciclopedia libre definición de robot
http://es.wikipedia.org/wiki/Robot

[2] WIKIPEDIA Definición de sensor
http://es.wikipedia.org/wiki/Sensor

[3] WIKIPEDIA definición de microcontrolador
http://es.wikipedia.org/wiki/Microcontrolador

[4] WIKIPEDIA articulo sobre radiación infrarroja
http://es.wikipedia.org/wiki/Radiaci%C3%B3n_infrarroja

[5] Artículo Sobre Baterías
http://html.rincondelvago.com/baterias.html

[6] BATERÍAS DE LITIO.
La alternativa al plomo y al cadmio.
http://www.cienciateca.com/ctslibat.html

[7]Articulo sobre las ventajas y desventajas de las baterías de litio.
http://www.imagendv.com/liion1.htm

[8]Articulo sobre robots industriales, del tipo manipulador
http://www.cpr2valladolid.com/tecno/cyr_01/robotica/industrial.htm

Distribuidor de productos para robótica y automatización
[9] http://www.crya.com.mx/Sub/motores.htm
[10]http://www.crya.com.mx/Imagenes/mr1230e.jpg

[11]Guía de uso del MPLAB.

http://victronics.cl/noticias/mplab_proyecto.zip

[12] Artículo sobre lenguaje de programación PBP (Pic Basic Pro)
http://www.todopic.com.ar/pbp_sp.html

[13] Pagina de la empresa creadora de PBP micro Engineering Labs Inc.
Instalación de PBP en MPLAB IDE
http://melabs.com/support/mplab.htm

[14] Rama de estudiantes IEEE Universidad Carlos III de Madrid
http://www.ieee.uc3m.es/concurso_circ.htm

[15]Articulo sobre clasificación de los motores eléctricos
http://usuarios.lycos.es/mugresoft/clasificacion_general_de_los_motores_electricos.htm

[16] Articulo sobre maquinas de corriente directa
http://inspeccion-uvmi6.iespana.es/inde9236.pdf

[17] Aspectos básicos de Robots Autónomos y Lineamientos para la Construcción de un Equipo de Fútbol Robótico
Ing. Balich, Néstor Adrián [e-mails:nestorbalich@8bytes.com.ar]
Facultad de Tecnología Informática
Universidad Abierta Interamericana
Chacabuco 90 1er. Piso
Buenos Aires – Argentina
Mayo, 2004
http://www.exa.unicen.edu.ar/cafr2004/pages_Spanish/papers/WCAFR2004-12.pdf

[18] Pagina de Maxon una empresa dedicada entre otras cosas a la construcción de motores de alta eficiencia.
http://test.maxonmotor.com/docsx/Download/catalog_2005/Pdf/05_wichgtiges_dc_ec_motoren_sp.pdf

Todas las referencias electronicas fueron verificadas el 19 de junio de 2006

[19] MPLAB ICD 2, In-Circuit Debugger User's Guide
Microchip Technology Inc. 2003

[20] MANUAL DE USUARIO DEL PIC16F87X
Microchip Technology inc. 2001
Véase apéndice B

[21] Hojas de Datos L298
Thompson Semiconductors 2000

[22] Hojas de Datos IR383
Proporcionadas por STEREN
Véase apéndice B

[23] Hojas de Datos PT1302B
Proporcionadas por STEREN
Véase apéndice B

[24] Hojas de Datos QRD1114
Fairchild 2005
Véase apéndice B

[25] Hojas de Datos HOA1404
Véase apéndice B

[26] Maquinas Eléctricas Rotativas y Transformadores
Cuarta edición
Donald V. Richardson, MME, PE
Prentice-Hall Hispanoamérica
1997

[27] Diseño, Construcción y Control Difuso De Un Robot Móvil De Competencia.
Amaranto de Jesús Dávila J.
Diana Aurora Cruz Hernández
2004

APÉNDICE A

```
Include "modedefs.bas"
Include "ICDDEFS.BAS"
        REAL VAR BYTE
        FIN VAR BYTE
        AP2 VAR BYTE
        AP3 VAR BYTE
        SIGUIENTE1 VAR BYTE
        SIGUIENTE2 VAR BYTE
        SIGUIENTE3 VAR BYTE
        AP VAR BYTE
        CRUCE VAR BYTE
        ACTUAL VAR BYTE
        POSTERIOR VAR BYTE
        DELD VAR BIT
        DELI VAR BIT
        I VAR  BYTE
        CASO VAR BYTE
        CERO VAR BIT
        FI VAR WORD
        FD VAR WORD
        CI VAR WORD
        CC VAR WORD
        CD VAR WORD
        TI VAR WORD
        TD VAR WORD
        Define   ADC_BITS        10         ' Set number of bits in result
        Define   ADC_CLOCK    3          ' Set clock source (3=rc)
        Define   ADC_SAMPLEUS          50         ' Set sampling time in uS
        TRISA = %11111111       ' Set PORTA to all input
        ADCON1 = %10000000  ' Set PORTA analog and right justify result
        TRISE = %11111111
        TRISB = %00001000
        TRISC = %00000000
   OPTION_REG = $7f       ' Enable PORTB pullups
        PORTB=0
        AP=0                'VALOR INICIAL DE APUNTADOR
        DELI=0
        DELD=0

IF PORTB.3=1 THEN
        GOTO INICIO
ENDIF
GOTO REPRODUCIR
INICIO1:
ZERO:
        PORTB.0=0   'DER
        PORTB.1=0       'IZQ
        PORTB.2=0       'DEL
RETURN
LDER:
        PORTB.0=1   'DER
        PORTB.1=0       'IZQ
        PORTB.2=0       'DEL
RETURN
LIZQ:
        PORTB.0=0   'DER
        PORTB.1=1       'IZQ
        PORTB.2=0       'DEL
RETURN
T:
```

```
        PORTB.0=1   'DER
        PORTB.1=1       'IZQ
        PORTB.2=0       'DEL
RETURN
TIZQ:
        PORTB.0=0   'DER
        PORTB.1=1       'IZQ
        PORTB.2=1       'DEL
RETURN
TDER:
        PORTB.0=1   'DER
        PORTB.1=0       'IZQ
        PORTB.2=1       'DEL
RETURN
CRUZ:
        PORTB.0=1   'DER
        PORTB.1=1       'IZQ
        PORTB.2=1       'DEL
RETURN

PARAR:
        Hpwm 2,0,300    ' Send a 50% duty cycle PWM signal at 1kHz
        Hpwm 1,0,300    ' Send a 50% dut
        'PORTC=0
        PORTB.5=0
        PORTB.4=0
        RETURN
DERECHO:
        Hpwm 2,225,300' Send a 50% duty cycle PWM signal at 1kHz
        Hpwm 1,225,300' Send a 50% duty cycle PWM signal at 1kHz
        'PORTC.2=1
        'PORTC.1=1
        PORTB.5=0
        PORTB.4=0
        RETURN
DERECHA:
        Hpwm 2,205,300
        Hpwm 1,50,300
        'PORTC.2=0
        'PORTC.1=1
        PORTB.5=0
        PORTB.4=0
        RETURN
IZQUIERDA:

        Hpwm 1,205,300' Send a 50% duty cycle PWM signal at 1kHz
        Hpwm 2,50,300
        'PORTC.2=1
        'PORTC.1=0
        PORTB.4=0
        PORTB.5=0
        RETURN
IZQB:
        HPWM 1,127,300
        HPWM 2,127,300
        'PORTC.2=0
        'PORTC.1=1
        PORTB.5=0
        PORTB.4=1
        RETURN
```

```
DERB:
        HPWM 1,127,300
        HPWM 2,127,300
        'PORTC.2=1
        'PORTC.1=0
        PORTB.4=0
        PORTB.5=1
        RETURN
BLANCO:
         DELD=0
         DELI=0
         GOSUB DERB
GIR1:
         ADCIN 4,CC
                IF CC < 500 THEN
                 'GOSUB DERECHO
                 'PAUSE 100
                 RETURN
                ENDIF
         GOTO GIR1
         RETURN



GIRDER:
         DELD=0
         DELI=0
    GOSUB DERB
          'GOSUB GUARDA
GIR:
         ADCIN 1,CD
                IF CD < 500 THEN
                GOSUB DERECHO

                'PAUSE 100
                 'FOR I=0 TO 500
                 'GOSUB CENTRAR
                 'NEXT I
                 RETURN
                ENDIF
         GOTO GIR
         RETURN

GIRIZQ:
         DELD=0
         DELI=0
         'GOSUB GUARDA
    GOSUB IZQB
                PAUSE 300
GIR2:
         ADCIN 5,CI
                IF CI < 500 THEN
                 RETURN
                ENDIF
         GOTO GIR2
         RETURN

CENTRAR:
CEN:
        ADCIN 4,CC
```

```
        IF CC < 300 THEN
         GOSUB DERECHO
        ENDIF
DER:
        ADCIN 5,CI
        IF CI < 300      THEN
         GOSUB DERECHA
         'GOTO DERB
        ENDIF
IZQ:
        ADCIN 1,CD                      'EL REAL ES 1
        IF CD < 300 THEN
   GOSUB IZQUIERDA
         'GOTO IZQB
        ENDIF
        'GOSUB  DERECHO
        RETURN
GUARDA:
        WRITE AP,CRUCE
        AP=AP+1
        RETURN


        ' APRENDE APRENDE APRENDE APRENDE APRENDE APRENDE
APRENDE:
 FIN=AP
 AP=0            'BARRIDO
 AP2=0           'MEMORIA
 AP3=0
WHILE AP<=FIN
        AP3=AP2
        READ AP,REAL
        IF REAL<> 0 THEN
                WRITE AP2,REAL
                AP2=AP2+1
                AP=AP+1
        ELSE
                WRITE AP2,REAL
                AP2=AP2+1
                AP=AP+2
        ENDIF
WEND
FOR I=AP3+1 TO FIN
WRITE I,255
NEXT
FIN=AP3
REPITE:
AP3=0
AP2=0
FOR I=0 TO FIN
ELIMINA:
        READ AP,ACTUAL
        READ AP+1,SIGUIENTE1
        READ AP+2,SIGUIENTE2
        READ AP+3,SIGUIENTE3
        IF (ACTUAL==4) && (SIGUIENTE1==0)&&(SIGUIENTE2==0)&&(SIGUIENTE3==0)
THEN
                WRITE AP2,0
                AP=AP+5
                AP2=AP2+1
```

```
                AP3=1
                GOTO ELIMINA
        ENDIF
        IF (ACTUAL<>4)&&(SIGUIENTE1==0)&&(SIGUIENTE2==0) THEN
                AP2=AP2+1
                AP3=1
                GOTO ELIMINA
        ENDIF
WRITE AP2,ACTUAL
AP=AP+1
AP2=AP2+1
NEXT I
IF AP3==1 THEN
        GOTO REPITE
ENDIF
AP=0
AP2=0
DIRECCON:
FOR I=0 TO FIN
READ AP,REAL
READ AP+1,SIGUIENTE1
READ AP+2,SIGUIENTE2
IF  SIGUIENTE1==0 & SIGUIENTE2<>0 THEN
        REAL.4=1
        WRITE AP2,REAL
        AP=AP+2
        AP2=AP2+1
        GOTO BRINCA
ENDIF
IF REAL==4 & SIGUIENTE1==0 & SIGUIENTE2==0 THEN
        REAL.4=1
        REAL.5=1
        WRITE AP2,REAL
        AP=AP+3
        GOTO BRINCA
ENDIF
WRITE AP2,REAL
AP=AP+1
AP2=AP2+1
BRINCA:
NEXT I
RETURN
TABLA1:
READ AP,REAL
IF REAL.4=1 THEN
        GOSUB GIRIZQ
        AP=AP+1
        RETURN
ENDIF
GOSUB CENTRAR
PAUSE 200
DELI=0
DELD=0
AP=AP+1
RETURN

TABLA2:
READ AP,REAL
IF REAL.4=0 THEN
        GOSUB GIRDER
```

```
        AP=AP+1
        RETURN
ENDIF
GOSUB CENTRAR
PAUSE 200
DELD=0
AP=AP+1
RETURN
TABLA3:
READ AP,REAL
IF REAL.4=0 THEN
        GOSUB GIRDER
        AP=AP+1
        RETURN
ENDIF
        GOSUB GIRIZQ
        AP=AP+1
RETURN
TABLA4:
READ AP,REAL
IF REAL.4=0 THEN
        GOSUB GIRDER
        AP=AP+1
        RETURN
ENDIF
IF REAL.5=0 THEN
        GOSUB CENTRAR
        DELD=0
        DELI=0
        AP=AP+1
        RETURN
ENDIF
GOSUB GIRIZQ
AP=AP+1
RETURN
REPRODUCIR:
RELOAD:
ADCIN 4,CC
ADCIN 5,CI
ADCIN 1,CD
ADCIN 3,FD
IF FD< 700 THEN
        DELD=1
ENDIF
ADCIN 2,FI
IF FI< 700 THEN
        DELI=1
ENDIF
        ' BLANCO
IF  CI>500 & CC>500 & CD>500 & DELD==0 & DELI=0 THEN
        GOSUB PARAR
        PAUSE 1
   GOSUB BLANCO
ENDIF
GOSUB CENTRAR
        'CUANDO LLEGO A LA META
ADCIN 7,TI
IF FI < 700 & TI< 700 THEN
PARO3:
        GOSUB PARAR
```

```
        GOTO PARO3
ENDIF
        ' CRUCE T
ADCIN 6,TD
ADCIN 7,TI
IF DELD ==1 & DELI == 1 & CI>500 & CC>500 & CD>500 & ( TD<700 OR TI < 700 ) THEN
        GOSUB TABLA3
ENDIF
        ' TIZQ
IF DELD ==0 & DELI == 1 &( CI<500 OR CC<500 OR CD<500) & ( TD<700 OR TI < 700 ) THEN
        GOSUB TABLA1
ENDIF

        ' TDER
IF DELD ==1 & DELI == 0 & (CI<500 OR CC<500 OR CD<500 )& ( TD<700 OR TI < 700 ) THEN
        GOSUB TABLA2
ENDIF
        'CRUZ
IF DELD ==1 & DELI == 1 & (CI<500 OR CC<500 OR CD<500) & ( TD<700 OR TI < 700 ) THEN
        GOSUB TABLA4
ENDIF
        'cruce L DER
ADCIN 6,TD
IF  DELD==1 & DELI=0 & CI>500 & CC>500 & CD>500   THEN
        GOSUB DERECHO
MANTEN4:
        ADCIN  6,TD
        IF TD<700 THEN
                GOSUB GIRDER
                GOTO RELOAD
        ENDIF
        GOTO MANTEN4
ENDIF
        'CRUCE L IZQ
IF DELI==1 & DELD== 0 & CI>500 & CC>500 & CD>500 THEN
        GOSUB DERECHO
MANTEN3:
        ADCIN 7,TI
        IF TI<700 THEN
                GOSUB GIRIZQ
                GOTO RELOAD
        ENDIF
        GOTO MANTEN3
ENDIF
GOTO RELOAD

INICIO:
        'GOSUB VER
ADCIN 4,CC
ADCIN 5,CI
ADCIN 1,CD
        ' BLANCO
IF  CI>500 & CC>500 & CD>500 & DELD==0 & DELI=0 THEN
        GOSUB PARAR
        PAUSE 1
        GOSUB GUARDA
        GOSUB ZERO
        GOSUB BLANCO
ENDIF
GOSUB CENTRAR
```

```
ADCIN 3,FD
IF FD< 700 THEN
        DELD=1
ENDIF
ADCIN 2,FI
IF FI< 700 THEN
        DELI=1
ENDIF
'CUANDO LLEGO A LA META
ADCIN 7,TI
IF FI < 700 & TI< 700 THEN
PARO:
        GOSUB PARAR
        GOSUB APRENDE
        GOTO PARO
ENDIF
        ' CRUCE T
ADCIN 6,TD
ADCIN 7,TI
IF DELD ==1 & DELI == 1 & CI>500 & CC>500 & CD>500 & ( TD<700 OR TI < 700 ) THEN
        CRUCE=%00000011 'CRUCE 3
        GOSUB GUARDA
        GOSUB T
        GOSUB GIRDER
ENDIF
        ' TIZQ
IF DELD ==0 & DELI == 1 &( CI<500 OR CC<500 OR CD<500) & ( TD<700 OR TI < 700 ) THEN
        CRUCE=%00000001 'CRUCE 1
        GOSUB GUARDA
        GOSUB TIZQ
        GOSUB CENTRAR
        DELI=0
        DELD=0
ENDIF
        ' TDER
IF DELD ==1 & DELI == 0 & (CI<500 OR CC<500 OR CD<500 )& ( TD<700 OR TI < 700 ) THEN
        CRUCE=%00000010 'CRUCE 2
        GOSUB GUARDA
        GOSUB TDER
        GOSUB GIRDER
ENDIF
        'CRUZ
IF DELD ==1 & DELI == 1 & (CI<500 OR CC<500 OR CD<500) & ( TD<700 OR TI < 700 ) THEN
        CRUCE=%00000100 'CRUCE 4
        GOSUB GUARDA
        GOSUB CRUZ
        GOSUB GIRDER
ENDIF
        'cruce L DER
ADCIN 6,TD
IF TD < 700 & DELD==1 & DELI=0 & CI>500 & CC>500 & CD>500   THEN
        GOSUB LDER
        GOSUB GIRDER
ENDIF
        'CRUCE L IZQ
IF DELI==1 & DELD== 0 & CI>500 & CC>500 & CD>500 THEN
        GOSUB LIZQ
        GOSUB DERECHO
MANTEN:
        ADCIN 7,TI
```

```
        IF TI<700 THEN
                GOSUB GIRIZQ
                GOTO INICIO
        ENDIF
        GOTO MANTEN
ENDIF
GOTO INICIO
END
```

APÉNDICE B

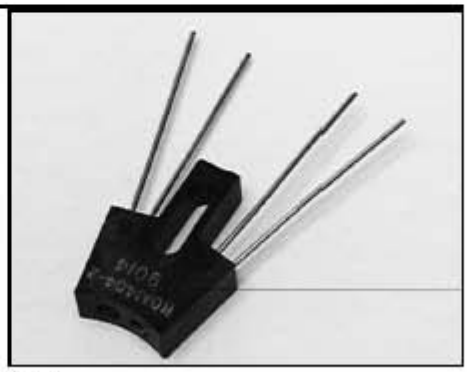
A continuación se muestran las hojas de datos (datasheets) de los componentes mencionados y algunos utilizados en nuestro diseño..

HOA1404

Reflective Sensor

FEATURES

- Choice of phototransistor or photodarlington output
- Focused for maximum response
- Wide operating temperature range (- 55°C to +100°C)



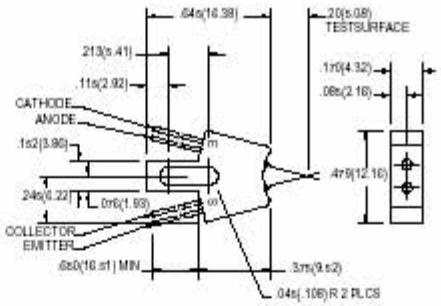
DESCRIPTION

The HOA1404 series consists of an infrared emitting diode and an NPN silicon phototransistor (HOA1404- 001,- 002) or photodarlington (HOA1404- 003), encased side-by- side on converging optical axes, in a black thermoplastic housing. The detector responds to radiation from the IRED only when a reflective object passes within its field of view. The HOA1404 series employs metal can packaged components. For additional component information see SE1450, SD1440, and SD1410.

Housing material is acetal copolymer. Housings are soluble in chlorinated hydrocarbons and ketones. Recommended cleaning agents are methanol and isopropanol.

OUTLINE DIMENSIONS in inches (mm)

Tolerance		
	3 plc decimals	±0.010(0.25)
	2 plc decimals	±0.020(0.51)



Honeywell

Honeywell reserves the right to make changes in order to improve design and supply the best products possible.

FAIRCHILD SEMICONDUCTOR®

March 2005

QRD1113/1114
Reflective Object Sensor

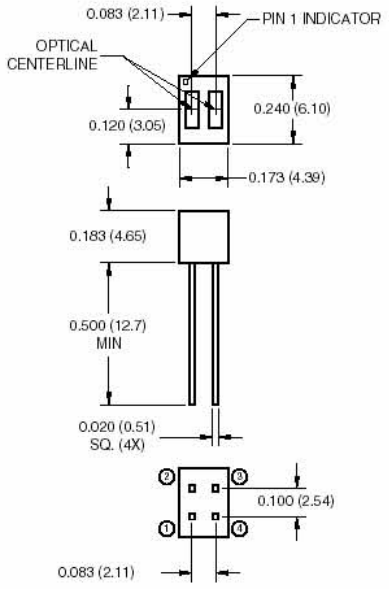
Features

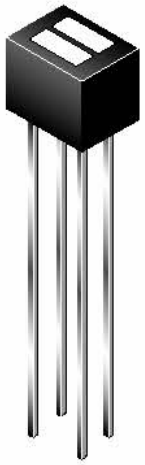
- Phototransistor Output
- No contact surface sensing
- Unfocused for sensing diffused surfaces
- Compact Package
- Daylight filter on sensor

Description

The QRD1113/14 reflective sensor consists of an infrared emitting diode and an NPN silicon photodarlington mounted side by side in a black plastic housing. The on-axis radiation of the emitter and the on-axis response of the detector are both perpendicular to the face of the QRD1113/14. The photodarlington responds to radiation emitted from the diode only when a reflective object or surface is in the field of view of the detector.

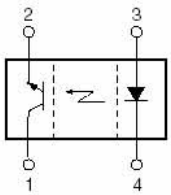
Package Dimensions





PIN 1 COLLECTOR PIN 3 ANODE
PIN 2 EMITTER PIN 4 CATHODE

Schematic



NOTES:

1. Dimensions for all drawings are in inches (millimeters).
2. Tolerance of ± .010 (.25) on all non-nominal dimensions unless otherwise specified.
3. Pins 2 and 4 typically .050" shorter than pins 1 and 3.
4. Dimensions controlled at housing surface.

STEREN

DEVICE NUMBER : DPT-130-023 REV : 2.1
ECN : PAGE : 3/7

5mm Phototransistor

MODEL NO : PT1302B/C2

■ Absolute Maximum Ratings at T_A = 25℃

Parameter	Symbol	Rating	Unit	Notice
Collector-Emitter Voltage	V_{CEO}	30	V	
Emitter-Collector- Voltage	V_{ECO}	5	V	
Collector Current	I_C	20	mA	
Operating Temperature	Topr	-25 ~ +85	℃	
Storage Temperature	Tstg	-40 ~ +85	℃	
Soldering Temperature	Tsol	260	℃	4mm from mold body less than 5 seconds
Power Dissipation at(or below) 25℃ Free Air Temperature	Pc	75	mW	

■ Electronic Optical Characteristics :

Parameter	Symbol	Min.	Typ.	Max.	Unit	Condition
Collector-Emitter Breakdown Voltage	BV_{CEO}	30	----	----	V	I_C=100 μA Ee=0mW/cm²
Emitter-Collector Breakdown Voltage	BV_{ECO}	5	----	----	V	I_E=100 μA Ee=0mW/cm²
Collector-Emitter Saturation Voltage	$V_{CE(sat)}$	----	----	0.4	V	I_C=2mA Ee=1mW/cm²
Rise Time	t_r	----	15	----	μS	V_{CE}=5V I_C=1mA R_L=1000Ω
Fall Time	t_f	----	15	----		
Collector Dark Current	I_{CEO}	----	----	100	nA	V_{CE}=20V Ee=0mW/cm²
On State Collector Current	$I_{C(on)}$	0.7	1.0	----	mA	V_{CE}=5V Ee=1mW/cm²
Wavelength of Peak Sensitivity	λ_p	----	980	----	nm	----
Rang of Spectral Bandwidth	$\lambda_{0.5}$	----	700---1200	----	nm	----

DIODO EMISOR DE LUZ INFRARROJO

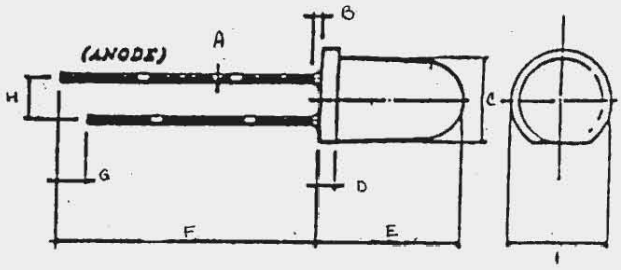
IR383

Descripción del producto.

Diodo emisor infrarrojo, fabricado de Arsénico de galio, altamente eficaz a largas distancias, encapsulado de plástico azul transparente.

Modelo	Material	Longitud de onda	Color del lente	Tensión en sentido direceto	Intensidad de radiación PO mW/sr 20mA If=mA Tip.	Tamaño	Ángulo
IR383	GaAs	940 nm	Azul transparente	1.3 < 1.7	100 26	5 mm	12°

DIMENSIONES (mm) Ver dibujo.
A - 0.5
B - 1.0 Máx.
C - 5.0 ± 0.2
D - 1.0 ± 0.2
E - 8.6 ± 0.3
F - 26.4 Mín.
G - 1.0 Mín.
H - 2.54
I - 5.6 ± 0.3



LM339, LM239, LM2901, LM2901V, NCV2901, MC3302

Single Supply Quad Comparators

These comparators are designed for use in level detection, low−level sensing and memory applications in consumer, automotive, and industrial electronic applications.

Features

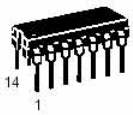
- Single or Split Supply Operation
- Low Input Bias Current: 25 nA (Typ)
- Low Input Offset Current: ±5.0 nA (Typ)
- Low Input Offset Voltage
- Input Common Mode Voltage Range to GND
- Low Output Saturation Voltage: 130 mV (Typ) @ 4.0 mA
- TTL and CMOS Compatible
- ESD Clamps on the Inputs Increase Reliability without Affecting Device Operation
- NCV Prefix for Automotive and Other Applications Requiring Site and Control Changes
- Pb−Free Packages are Available*

*For additional information on our Pb−Free strategy and soldering details, please download the ON Semiconductor Soldering and Mounting Techniques Reference Manual, SOLDERRM/D.

ON Semiconductor®

http://onsemi.com

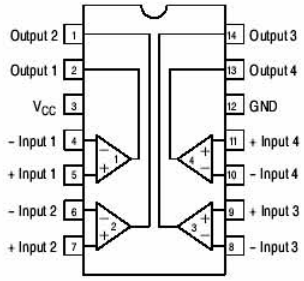
SOIC−14
D SUFFIX
CASE 751A



PDIP−14
N, P SUFFIX
CASE 646

TSSOP−14
DTB SUFFIX
CASE 948G

PIN CONNECTIONS



(Top View)

ORDERING INFORMATION

See detailed ordering and shipping information in the package dimensions section on page 6 of this data sheet.

DEVICE MARKING INFORMATION

See general marking information in the device marking section on page 8 of this data sheet.

March, 2006 − Rev. 15

Publication Order Number:
LM339/D

L298

DUAL FULL-BRIDGE DRIVER

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

Multiwatt15

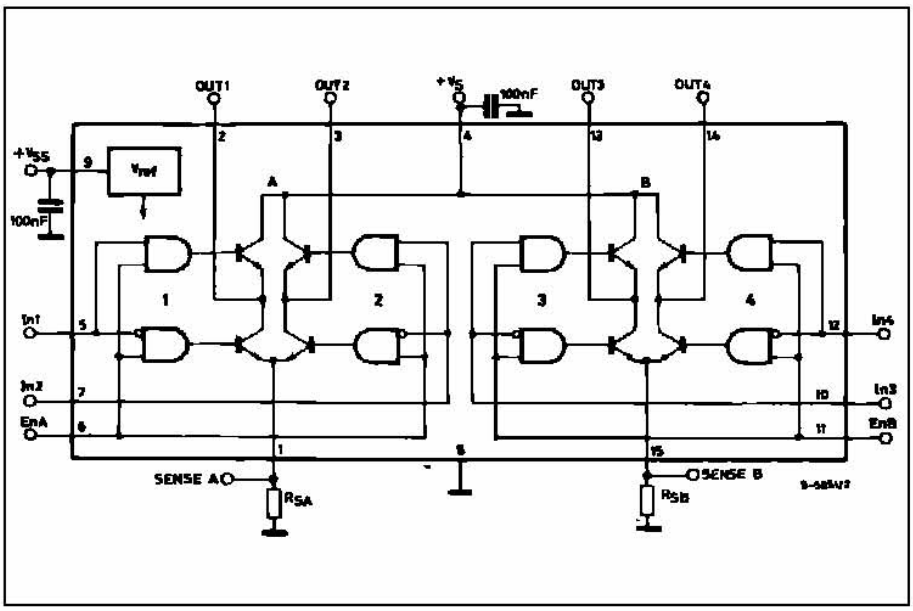
PowerSO20

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

DESCRIPTION

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

BLOCK DIAGRAM



PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

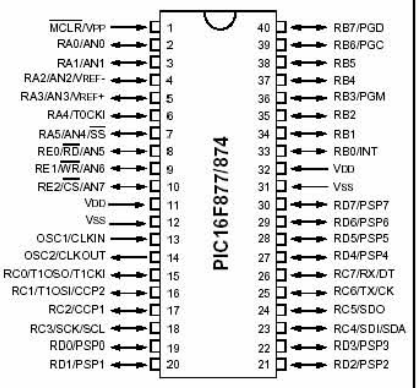
- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

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

Pin Diagram

PDIP



Peripheral Features:

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