UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES
ARAGÓN

ROBOT INTELIGENTE SEGUIDOR DE LÍNEA PARA LABERINTOS

T R A B A J O E S C R I T O
EN LA MODALIDAD DE CRÉDITOS DE MAESTRÍA
Q U E P A R A O B T E N E R E L T Í T U L O D E :
I N G E N I E R O E N C O M P U T A C I Ó N

P R E S E N T A

CARLOS FEDERICO LÓPEZ SPÍNDOLA

ASESOR:
ING. ARCELIA BERNAL DÍAZ

MÉXICO 2006

Agradecimientos

Este escrito representa una parte final en una etapa muy enriquecedora en el camino que el tiempo obliga. En toda la experiencia universitaria y en esta etapa final, ha habido personas que merecen las gracias por que sin su valiosa aportación y apoyo no hubiera sido posible este trabajo y también a quienes las merecen por haber plasmado su huella en mi camino.

Agradezco a Dios por su bondad y amor que han sido la fuente de mi fe.

A mis padres, Carlos López Sánchez y Mercedes Josefina Spíndola Solís, por todo su apoyo, confianza y guía, a lo largo de mi trayectoria académica.

A mis hermanas, Minerva Cecilia y Miriam Nohemí, y a mi abuelita, Herlinda, por su cariño, amistad y por compartir momentos muy significativos en mi vida.

A mi asesora de este trabajo, Ing. Arcelia Bernal Díaz, por su apoyo y paciencia, para poder concluirlo.

A la Universidad Nacional Autónoma de México, en particular a la Facultad de Estudios Superiores Aragón, a los profesores que contribuyeron con mi formación, y por la educación que recibí en general dentro de las aulas de esta universidad.

A mis amigos Jonathan Emmanuel y Alan Wenceslao, por su amistad, y quienes siempre me apoyaron con un buen consejo cuando lo necesitaba. Así como todos mis demás amigos que no menciono específicamente.

A Patricia Neri Villeda, por su amistad mientras realizaba este trabajo, cuya huella me permitió lograr cambios en mi persona muy deseados.

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUCCIÓN

i. Objetivo de la tesis de maestría

El objetivo en la tesis, es el diseño de un robot competitivo, preocupándose en todos los aspectos del robot, como lo son: el diseño electrónico, físico e inteligente. La intención es realizar un prototipo que muestre ideas innovadoras o mejoras sobre lo ya existente para cada uno de los aspectos mencionados e implementar dicho robot.

ii. Definición de Robot

La palabra "robot" es de origen eslavo en principio; en ruso, la palabra работа (rabota) significa labor o trabajo. Su significado actual fue introducido por el dramaturgo checoslovaco Karel Čapek (1890-1930) a principios del siglo XX, en un juego llamado R.U.R. (Rosum's Universal Robots). En su obra, Čapek creo robots trabajadores como sustitutos humanos, con apariencia humana y capaces de tener sentimientos como los de un humano.

Un robot es esencialmente una máquina o dispositivo que opera automáticamente. Puede tener apariencia humana, y pueden ser programados para realizar tareas humanas como motivo principal.

En la clasificación de los tipos de robots existentes, destacamos en forma muy general los *robots industriales* y los *robots móviles.* En este trabajo trataremos conceptos relacionados con los *robots móviles.*

iii. Robots Industriales

En los últimos años se ha introducido el concepto de robótica, el cual ha revolucionado la automatización de su clasificación denominada "fija", que consiste en la realización de la producción automática de piezas, elementos y productos en grandes cantidades o de manera repetitiva a su denominación actual: "automatización flexible", que estriba en adaptar la producción a la demanda de un mercado en constante cambio por medio de un sistema de producción programable y adaptable como lo es un robot. Este concepto de robótica abarca los *robots industriales.*

iv. Robots Móviles

Con los *robots móviles* lo que se intenta, es imitar a seres biológicos vivos. Reciben el nombre de *robots* debido a que son máquinas programables que son hasta cierto punto autónomos, son programados para la realización de una tarea específica. Son *móviles*, ya que no permanecen en un mismo lugar como sucede con la mayoría de los robots industriales. La característica de ser móviles es necesaria para que realicen la actividad para la que se encuentran programados.

Puede imaginarse en un futuro no muy lejano que la clasificación que se menciona para robots industriales y móviles, sea después inadecuada. Esto debido a la evolución de los mismos. No es difícil imaginarse que un robot móvil esté realizando tareas complejas en una industria, donde este requiera mover partes pesadas de un lugar a otro y para dicha tarea requiera desplazarse. Esta última tarea aun es realizada por humanos, con la ayuda de vehículos apropiados; dejando a los robots industriales tareas como el ensamble de partes en un lugar fijo.

v. Competencias de Robots Móviles

Existen competencias con robots móviles a nivel mundial. Estas competencias, tienen sus diversas modalidades, y cada una de estas sus reglas. Algunas de las diferentes modalidades son: *Robocup, Robo-Magellan, Line Following, Line Maze, Mini Sumo, etc.* En la modalidad *Robocup* se utilizan cuatro robots, los cuales deben coordinarse entre si para jugar un partido de fútbol (con sus reglas particulares), contra 4 robots similares en tamaño y características. En *Robo-Magellan,* se construye un robot que sea capaz de caminar por terreno con tierra, desniveles, pasto y piedras; el robot navega la zona y debe ser capaz de memorizar su entorno (con técnicas para crear mapas) para posteriormente ser capaz de recorrer el terreno nuevamente de un punto de salida a una meta en el menor tiempo posible. *Line Following* es quizás la modalidad más sencilla entre las mencionadas, se crea un robot con el objetivo de seguir una línea obscura en un fondo blanco (o viceversa); sin embargo, se cronometra la velocidad del robot. La modalidad *Line Maze* ha sido una consecuencia de *Line Following;* la línea que recorre el robot forma un laberinto, el robot debe ser capaz de recorrer en el menor tiempo posible alguna ruta que vaya de un punto de partida a un punto final. La última modalidad mencionada, *Mini Sumo,* consiste de dos robots que se enfrentan en una arena circular, el ganador es aquel que saca de la arena al robot rival o simplemente si permanece dentro de la arena si el otro robot sale de ella.

La finalidad de estos eventos es difundir y alentar la investigación en la constante mejora de este tipo de robots, tanto en un mejor diseño físico y electrónico, como en los algoritmos que se emplean en la inteligencia requerida para ser exitosos en las competencias. Comúnmente las reglas cada año exigen un mejor desarrollo del robot, y con nuevos retos.

vi. La competencia *Line Maze*

La competencia *Line Maze*, consiste en realizar un robot móvil, que sea capaz de seguir una línea negra en un fondo blanco, la cual forma un laberinto. El objetivo principal es que el robot recorra el laberinto en el menor tiempo posible. Para dicho objetivo el robot no tiene previo conocimiento del laberinto, y cuando es capaz de memorizarlo, aumenta sus posibilidades cuando vuelve a recorrer el laberinto en la competencia. No es una característica obligatoria por el momento para estas competencias, de que el robot deba ser capaz de memorizar el

laberinto; pero es esencial para ser exitoso, ya que al robot se le permite hacer el recorrido del laberinto hasta 3 veces. Y para los últimos 2 intentos, si el robot ya tiene un conocimiento previo del laberinto, este puede ser capaz de hacer un recorrido en un menor tiempo, si este es *inteligente*. En el apéndice A se muestran las reglas para dicha competencia según se exigieron en el reciente evento *Robothon 2005.*

vii. Panorama de este trabajo

Este trabajo está conformado en cinco capítulos. En la introducción se menciona brevemente el concepto de robot, sin embargo se dedica un capítulo para explicar como se caracteriza un robot móvil. Se trata en general las herramientas que pueden ser utilizadas para el control de un robot móvil, como lo son los microcontroladores y también dispositivos móviles como los PDA (*Personal Digital Assistant*, Asistente Personal Digital).

Capítulo I

Se da una introducción de los robots móviles. Se mencionan los módulos en los que se encuentran conformados y lo conveniente que es esta organización, para el correcto diseño y desarrollo de un robot móvil.

Capítulo II

Se describen los microcontroladores y sus características que pueden ser aprovechadas para un robot móvil. Se muestra un panorama de la elección de un microcontrolador según las necesidades y las tareas para las que esté planeado el robot. También se menciona una característica importante que se aprovecha mucho para robots móviles: PWM (*Pulse Width Modulation*, Modulación por Ancho de Pulso).

Capítulo III

En este capítulo se mencionan las posibilidades que brindan los dispositivos móviles para la aplicación de robots móviles. Son otra alternativa para el control de robots móviles. Por lo general no se pueden utilizar sin la compañía de un microcontrolador, pero ofrecen otras alternativas y ventajas.

Capítulo IV

En este capítulo se describe con detalle el diseño del prototipo del robot Radamanthys y su implementación. Se describen los problemas que se encontraron y los cambios que se tuvieron que hacer después de realizar el primer prototipo. Una vez corregido el primer prototipo, también se describen los resultados con el segundo diseño.

Capítulo V

Se exponen brevemente los resultados y las conclusiones que se obtuvieron con el diseño y la experimentación del robot. Se menciona el trabajo que aún falta por hacer para finalizar el robot y también se propone trabajo a futuro para tener otras alternativas para la competencia *Line Maze*.

Por último se presentan los apéndices que son de gran utilidad para poder apreciar con más detalle el diseño del robot e incluso poder construir uno. Uno de los apéndices contiene las reglas de la modalidad que se tomaron en cuenta para el diseño del robot. Finalmente se muestra la bibliografía consultada.

CAPÍTULO I

LOS ROBOTS MÓVILES

Un robot móvil trata de imitar a un ser vivo. Un ser vivo biológico como un insecto, un animal o incluso un humano, tienen la característica de moverse de lugar constantemente, son autónomos, interactúan con el medio ambiente que los rodea así como con más individuos. Todas estas características de un ser vivo biológico son precisamente las que trata de lograr un robot móvil. El reto es acercarse cada vez más a que esta imitación de vida, sea cada vez lo más fiel posible.

Para lograr que un robot móvil sea lo más apegado a la imitación de vida mencionada, se requiere una organización adecuada para su diseño. Esta organización ha sido propuesta anteriormente por módulos (ver figura 1.1); así, cada módulo puede desarrollarse con técnicas específicas y adecuadas.

Un robot móvil puede ser simulado en computadoras con el fin de facilitar la programación y depuración de varios módulos independientemente. Esto permite tener idea del comportamiento final del robot. Un simulador bastante utilizado en el área de robótica móvil en la UNAM es el Roc2 (*Robot Command Center*, UNAM 2003)

En la figura 1.1 se muestra un diagrama a bloques con una disposición completa y general de los módulos de un robot móvil, los cuales se toman igual en cuenta cuando se simula un robot (robot virtual). Esta organización ha permitido el exitoso desarrollo del simulador virtual Roc2.

Puede apreciarse como el robot interactúa directamente con el medio ambiente, el robot percibe el medio ambiente mediante el uso de sensores. Existen muchos tipos de sensores para esta etapa, y la elección de estos varia según el nivel de percepción que se desea del medio ambiente en el robot. Existen sensores infrarrojos, para detectar obstáculos a corta distancia, variaciones de color, o cantidad de luz. Se pueden emplear micrófonos para captar sonido y cámaras de video para implementar visión artificial en el robot. Es posible hasta tener sensores de contacto, temperatura, humedad, etc.

1.1. Sensores

Un sensor convierte fenómenos físicos captados del medio ambiente, en señales que sirvan de entrada al sistema electrónico del robot. Existen principalmente 2 tipos de sensores, basados en el tipo de salida que estos producen, estos son: sensores digitales y sensores analógicos.

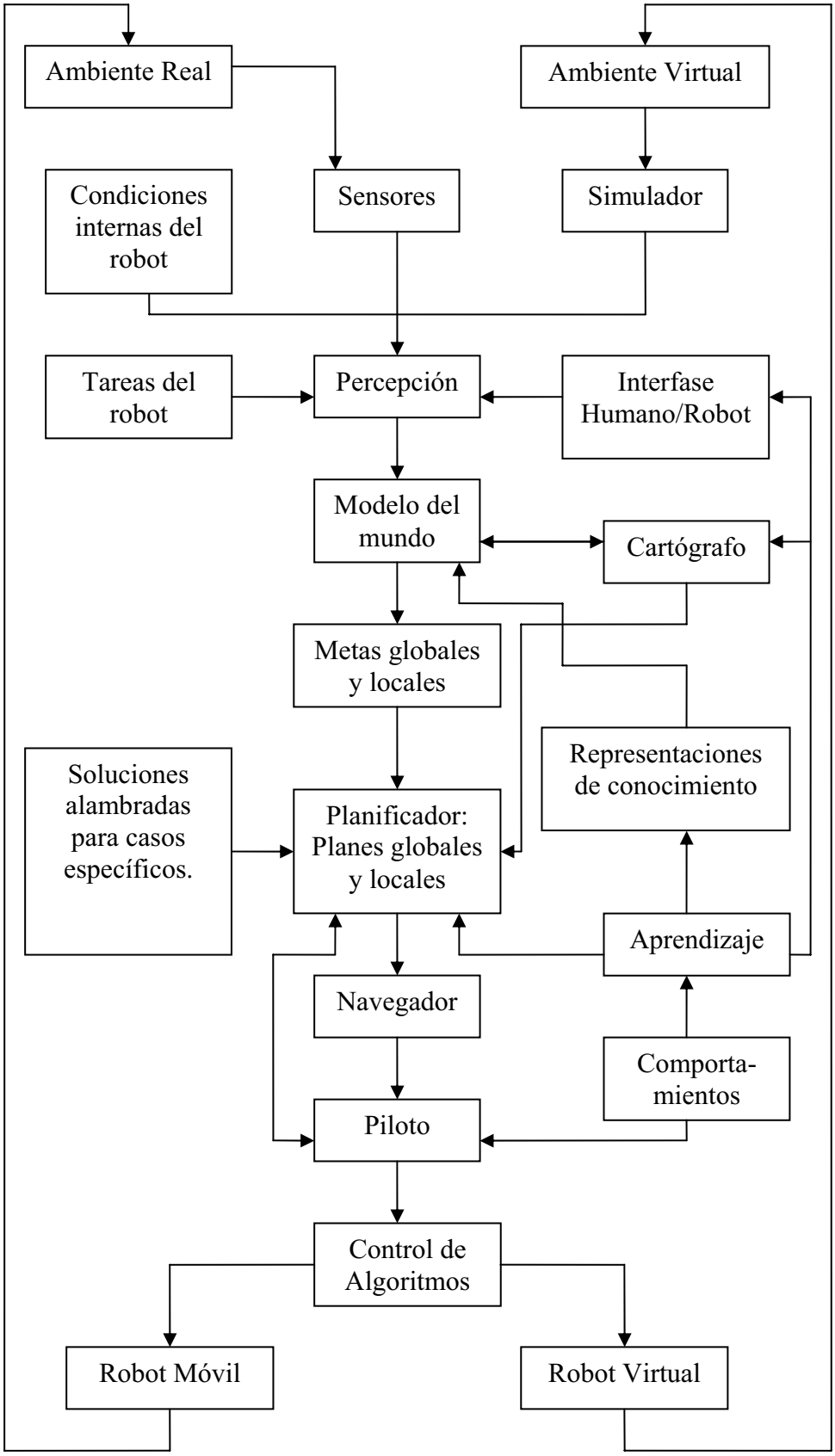

Figura 1.1. *Organización modular de un robot móvil.*

Los sensores digitales generan como salida, una señal que es una representación digital de la señal analógica que estos reciben, y tienen valores discretos en magnitud medidos en tiempos discretos. Un sensor digital debe proporcionar valores lógicos que sean compatibles con el receptor digital. Algunos estándares lógicos incluyen TTL (*transistor-transistor logic*, lógica transistor-transistor) y ECL (*emitter-coupled logic*, lógica de emisor acoplado). Un ejemplo de sensor digital es un sensor de contacto, estos al hacer contacto con un obstáculo, cierran un switch.

Los sensores analógicos generan una señal como salida, que es directamente proporcional a la señal de entrada, y es continua tanto en magnitud como en el tiempo. La mayoría de las variables físicas como temperatura, presión y aceleración son continuas por naturaleza y son precisamente este tipo de magnitudes las que se miden con un sensor analógico debido a su naturaleza. Un ejemplo sería un termistor, usado con el fin de medir temperatura.

Todos los sensores usados en Radamanthys son sensores analógicos, a su vez, estos son sensores infrarrojos, en total veinte. El sensor utilizado es conocido por su número de parte: QRD1114 (ver apéndice C). Dieciocho sensores son utilizados para la detección de la línea, y dos más como *encoders*[1] (sensor digital que genera una cantidad de pulsos mientras el robot avanza con el fin de medir la distancia que el robot avanza). Si bien el QRD1114 es un sensor analógico, su uso como *encoder*, donde se requiere un sensor digital, se logra mediante un comparador, y finalmente se tiene simplemente un '0' lógico o un '1' lógico para generar los pulsos requeridos.

El sensor QRD1114 está compuesto por un diodo infrarrojo como emisor y un fototransistor como receptor. La señal infrarroja emitida es reflejada sobre una superficie, el color de la superficie hace variar la señal, la cual provoca en la salida del sensor, una variación de voltaje. Una superficie blanca hará que el sensor tenga en su salida aproximadamente 5V y una obscura casi 0V. Es por esto que este sensor es ideal para detectar líneas obscuras sobre un fondo blanco, o viceversa. Dicho sensor brinda mejores resultados si se trata de evitar que su señal infrarroja se combine con la luz del medio ambiente; y para la detección de líneas, este debe usarse de 0.2 a 0.5 cm. de distancia entre el sensor y la superficie con la línea a detectar.

Los sensores establecen comunicación con el módulo de percepción. Son los encargados de brindar al robot una información bruta de su entorno.

[1] Codificador de pulsos a base de líneas claras y obscuras. En el caso de un robot móvil, este se encuentra en una de sus ruedas.

1.2. Percepción

El módulo de percepción se encarga de extraer toda la información de los sensores. También se encarga de entender el medio ambiente que rodea al robot. El módulo de percepción proporciona un *modelo del mundo*, información fiable a la hora de ser introducida en algoritmos que ya se dediquen a la inteligencia y los comportamientos del robot.

El uso de hardware dedicado para este módulo es lo más conveniente. Ya que de esta forma se dejan libres recursos al robot que puede emplear en la toma de sus decisiones, inteligencia, etc. Se pueden emplear DSPs (procesadores digitales de señales) en el caso de la necesidad de trabajar señales de audio, también son muy usados estos cuando se emplean sensores como sonares (emisores/receptores de ultrasonido). En el módulo de percepción se puede hacer uso de un sistema de visión, el cual procese imágenes, donde incluso la demanda de hardware ya sería más exigente, esto tan solo para la percepción.

En el caso de Radamanthys, la percepción no es muy compleja, consiste de los veinte sensores mencionados, 2 comparadores para la implementación de los *encoders*, 6 circuitos integrados 74HC4066 (switch analógico) para poder seleccionar el sensor a leer, 2 GAL22V10 (PLD – programmable logic device - para controlar los switches analógicos) y el convertidor A/D encontrado en el microcontrolador.

Dado que lo que Radamanthys detecta del medio ambiente es una línea obscura sobre un fondo blanco, los sensores analógicos empleados con dicho fin, terminan comportándose como sensores digitales en el sistema de percepción. El convertidor A/D del microcontrolador actúa como un comparador para cada sensor, con la posibilidad de ser calibrado por software. El robot solo necesita saber si la línea se encuentra o no se encuentra en la ubicación de cada uno de sus sensores.

Un robot móvil, teniendo información confiable del medio que lo rodea, puede utilizar dicha información para navegar su entorno de manera más eficiente. Mientras navega, también le es posible crear un mapa en su memoria para poder planear rutas que le permitan al robot realizar sus tareas debidamente. La calidad del mapa que el robot construye en su memoria es consecuencia de la información que el módulo de percepción pueda proporcionar.

1.3. Navegador

La navegación es una de las tareas con más reto para un robot móvil. Para tener éxito en este módulo del robot, es necesario tener otros módulos del robot exitosamente desarrollados. Estos módulos con dicha dependencia son: la *percepción*, el robot necesita un modelo del mundo con información relevante de

sus sensores; la *localización*, el robot debe ser capaz de determinar su posición en el ambiente (cartógrafo); *cognición*, el robot debe ser capaz de decidir como actuar para lograr sus metas (aprendizaje - comportamientos - piloto); *control de movimiento*, el robot debe ser capaz de modular sus motores para tomar trayectorias correctas (comportamientos - piloto).

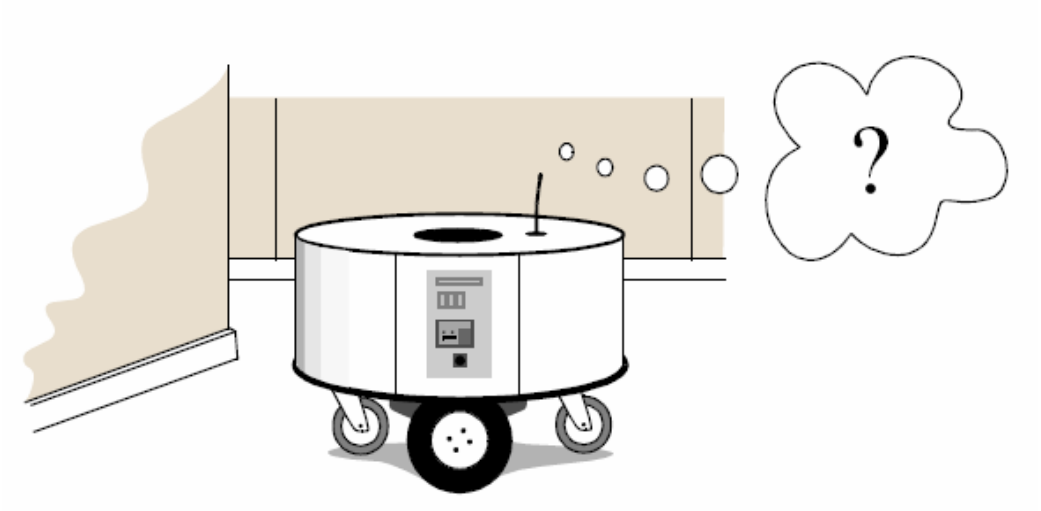

Figura 1.2. *¿Dónde estoy?*

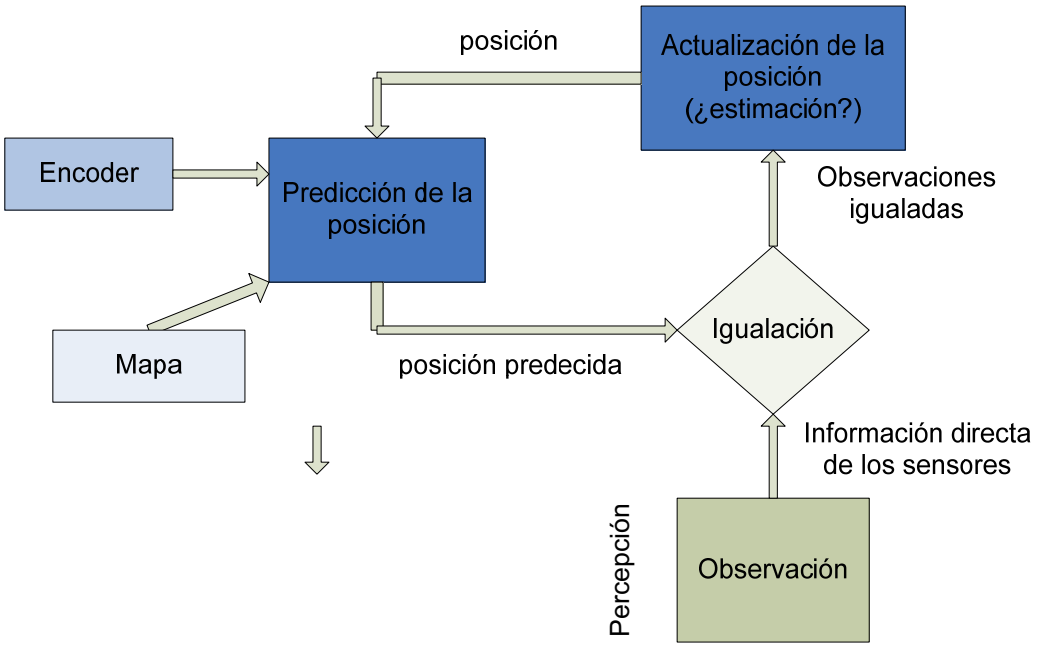

Figura 1.3. *Diagrama general para la localización de un robot.*

En la década pasada, la parte de *localización*, recibió la mayor atención en el desarrollo como parte del *navegador*. La *localización* enfrenta grandes retos, estos son ruido y *aliasing*[2] (información errónea).

Si se le pudiera colocar a cada robot un sensor GPS (*Global Positioning System*, Sistema de Posicionamiento Global), gran parte del problema de localización estaría resuelto, puesto que el sensor ya estaría proporcionando la información que resulta ser la mayor parte del problema. Desafortunadamente, dicho sensor por el momento no es práctico. Las actuales redes GPS, proporcionan la posición de un punto dentro de un rango de bastantes metros, lo cual es inaceptable para localizar robots móviles de tamaño humano, o incluso robots miniatura como robots de escritorio. Además la tecnología GPS no es posible usarla dentro de lugares muy cerrados.

El problema va más allá de localizar su posición con respecto a un lugar tan grande como es la Tierra con la ayuda de GPS. El problema incluye que el robot pueda identificar humanos por ejemplo, y que este pueda dar su posición relativa respecto a ellos; para dichos problemas el robot debe hacer uso de sus sensores. La inexactitud de los sensores y efectores en un robot hace que el problema de localización enfrente grandes retos.

El ruido en los sensores puede ser muy variado. Por ejemplo, un robot que utiliza una cámara CCD (*Charge-Coupled Device*, Dispositivo de Carga Acoplada) a color para visión dentro de un edificio, va a tener problemas cuando el sol se encuentre oculto por las nubes; la iluminación del edificio cambia y las señales *hue* (tonalidad) no son constantes. Pero eso no es lo único, una imagen sufre también de distorsión, ganancia en la señal, falta de definición, etc.

El robot Radamanthys sufre de un efecto similar al antes mencionado. Si los sensores infrarrojos no se aíslan de la luz del sol, cuando la luz del cuarto en el que se encuentra el robot cambia, los sensores dan diferentes resultados. Esto es inaceptable sobre todo cuando el cambio de luz se da durante la competencia, la cual si puede llegar a ocurrir. Se pueden optar por varias soluciones, ya que la variación se da solo en voltaje. El robot puede tener un sistema de auto-calibración, pero si el robot no sabe si en realidad hubo un cambio de luz y que necesita recalibrar sus sensores, entonces no tiene sentido. Además de que mientras este se encuentra recorriendo el laberinto, no es factible recalibrar sus sensores, este podría salirse de la línea, lo cual no está permitido. En este caso optamos por aislar sus sensores lo más que se pueda de agentes externos como en este caso la luz. Para aislar los sensores de la luz externa se utilizó *foamy* (material rugoso aislante de temperatura, humedad, luz, etc.) alrededor de estos, el *foamy* quedó prácticamente rozando el suelo.

[2] Distorsión de la señal ocasionada por una frecuencia de muestreo menor o igual a la frecuencia Nyquist. Ocasionando que los componentes de alta frecuencia mayores a la mitad de la frecuencia de muestreo, se traslapen con los componentes de menor frecuencia.

Un ejemplo de aliasing en los sensores, está en el uso de sonares ultrasónicos. Los sonares ultrasónicos sirven comúnmente para localizar objetos frente al robot a cierta distancia, es posible determinar la distancia del objeto. Sin embargo este tipo de sensor no proporciona información sobre el color, la textura y la rigidez del objeto. El problema surge cuando el robot tenga que identificar entre humanos y objetos inanimados. Cuando el robot se enfrenta en un ambiente con objetos en movimiento, puede representar un problema si el robot no sabe identificar entre un objeto en movimiento y uno estático y este no pueda planear una ruta hacia un destino.

El problema de aliasing es considerado para el robot Radamanthys. Durante la competencia, el camino siempre está libre, y simplemente no va a encontrarse con dos líneas paralelas debajo de todo su arreglo de sensores. Aunque la probabilidad de que los sensores perciban algo incoherente y el robot pueda interpretar que se encuentra viendo 2 líneas paralelas o simplemente tener una distorsión y no poder definir la línea a seguir puede ocurrir. Sin embargo es algo que no será constante, un error de estos en la lectura será simplemente muy breve. La solución es considerar esto en la programación del microcontrolador. Mientras el robot se encuentra escaneando la línea a seguir, en caso de tener una lectura inconcisa, el robot sigue caminando por breves momentos mientras sus sensores siguen leyendo, en caso de que el robot siga sin poder definir la línea debajo de sus sensores este se detiene para no salirse de la línea, ya que sería penalizado. Una vez detenido el robot, el escaneo sigue, si define la línea, este continúa, de lo contrario permanece detenido. No es muy probable que el robot termine en el último caso mencionado; lo que puede ocurrir es que el robot lea en todos sus sensores una mancha negra, lo cual significa que ha llegado a la meta, entonces el robot solo guarda el dato de la posición de la meta, y si aún dispone de tiempo, sigue recorriendo el laberinto. (Consultar el apéndice A que contiene las reglas de la competencia para más detalles).

CAPÍTULO II

LOS MICROCONTROLADORES PARA LA IMPLEMENTACIÓN DE UN ROBOT

2.1. Definición de Microcontrolador

Un microcontrolador (MCU) es un circuito integrado programable, usado para controlar dispositivos electrónicos. Algunas aplicaciones de un microcontrolador son los sistemas dedicados, como por ejemplo una chapa de seguridad, una máquina expendidora de refrescos, entre otros.

Los microcontroladores están constituidos por 4 elementos básicos, suficientes para un sistema dedicado. Estos son un CPU, la memoria de programa (memoria ROM o Flash), la memoria de datos (memoria RAM) y uno o más contadores/temporizadores. Algunos incluso cuentan con periféricos de entrada/salida como convertidores A/D y D/A.

La mayoría de los microcontroladores están basados en la arquitectura Harvard, la cual dispone de 2 memorias, una para datos y otra para instrucciones; además cada memoria cuenta con su respectivo bus. A diferencia de los microprocesadores que tradicionalmente se basan en la arquitectura Von Neumann, donde se utiliza una sola memoria para datos e instrucciones con un único bus.

También encontraremos que el CPU de la mayoría de los microcontroladores es del tipo RISC (*Reduced Instruction Set Computer*, Computadora con Conjunto de Instrucciones Reducido). Mientras que varios microprocesadores actuales cuentan con un CPU del tipo CISC (*Complex Instruction Set Computer*, Computadora con Conjunto de Instrucciones Complejo). La diferencia entre estos es que los CPU del tipo RISC cuentan con un conjunto de menores instrucciones comparados con los CISC. Las instrucciones de un CPU RISC suelen ser más fáciles de comprender, mientras que las instrucciones de un tipo CISC cuenta con instrucciones que pueden ejecutar lo que un tipo RISC hace con varias instrucciones. El código ensamblador de un programa de un CPU CISC suele ser más reducido que el de un tipo RISC para la misma tarea específica.

En un principio, los MCU eran programados totalmente en lenguaje ensamblador, pero la continua evolución y demanda de lenguajes de más alto nivel en esta área, han permitido que ya sea posible programarlos en C o en Basic.

En el primer prototipo de Radamanthys se utilizó el microprocesador Motorola 68HC11F1. Cuando se tuvieron que realizar cambios en la disposición de sensores (expuesto en el Capítulo IV), se hicieron pruebas con el microcontrolador de Microchip PIC16F877, se obtuvieron mejores resultados al programar la técnica PWM (*Pulse Width Modulation*, Modulación por Ancho de Pulso); por lo que se dejó el MCU de Microchip en el prototipo final. En la figura

2.1 se muestra el diagrama a bloques del PIC16F877, que fue el que se utiliza para Radamanthys.

2.2. Elección de un MCU

La elección de un microcontrolador para el uso de un robot requiere tomar algunas consideraciones. Debe uno de estar consiente de que es lo que el robot realizará, que tipo de sensores y que cantidad utilizará, la cantidad de efectores (lo común son 2 motores para desplazar el robot en cualquier dirección). Incluso si el MCU tendrá que servir como interfase entre otros dispositivos electrónicos.

Supongamos que nuestro robot únicamente debe seguir una línea obscura en un fondo blanco, la línea es continua y no contiene bifurcaciones. En realidad no requiere de tomar decisiones que dependan de algoritmos complejos. Un robot seguidor de línea puede ser implementado incluso sin un microcontrolador. Es por eso que se toma en cuenta la tarea para la que estará hecho el robot.

Radamanthys emplea 20 sensores analógicos y 2 motores DC. Las consideraciones que tomamos a partir de este punto. Son elegir un MCU que tenga la suficiente cantidad de entradas para tal cantidad de sensores, y las salidas para los efectores.

Dado que los sensores son analógicos, buscamos un MCU que tenga al menos un puerto de entrada con un convertidor A/D. Es difícil encontrar un MCU con 20 entradas con convertidor A/D. Esto nos hace optar por una alternativa que es la de emplear más hardware. En nuestro caso elegimos el uso de switches analógicos, el circuito integrado 74HC4066 permite tener diversas entradas analógicas y seleccionar de entre ellas una salida con lógica digital. De esta forma requerimos tan solo de 2 entradas con A/D en el MCU (para más detalle sobre la construcción del robot ver el capítulo IV).

Para los efectores consideramos que en nuestro caso queremos controlar la velocidad de los motores. Buscamos un MCU que cuente con PWM (*Pulse Width Modulation*, se explica en este mismo capítulo en el punto 2.3). El PIC16F877 cuenta con 2 salidas para PWM independientes, suficiente para los 2 motores de Radamanthys.

Otro punto a considerar para la elección de un MCU adecuado es que el robot requiere ejecutar un algoritmo que necesita muchos recursos de memoria, para poder crear un mapa del laberinto y elegir la mejor ruta; no está de más también considerar un MCU lo suficientemente rápido. En este caso resulta que no hay MCU con dichas características, o más bien son de un costo muy alto y son fabricados para aplicaciones comerciales, por lo que dichos circuitos integrados no se encuentran en un encapsulado PDIP o PLCC; que sería el

requerido para implementar nuestros circuitos mediante la elaboración de nuestros propios PCB (*Printed Circuit Board*, Tarjeta de Circuito Impreso). Una alternativa podría haber sido el uso de un MCU de la compañía AVR, se distinguen por su notable velocidad. Sin embargo para el prototipo se empleó material ya disponible y que no se requería comprar. El uso de una *Pocket PC* también se incorporó en el diseño del prototipo, con el fin de que esta se encargue del algoritmo de inteligencia utilizado para resolver el laberinto; y de esta forma, librándonos de la preocupación de adquirir un MCU del que no se disponía.

Otro factor a considerarse al elegir un MCU, es que si se va a utilizar un dispositivo móvil en el diseño del robot; se elige un MCU que cuente con un puerto de comunicaciones serial que sea compatible con otros dispositivos móvil a utilizarse. Casi todos los microcontroladores disponibles en el mercado cuentan con comunicaciones seriales; no es de extrañarse puesto que las aplicaciones de control en las que se utilizan los microcontroladores emplean dicha interfaz.

Una *Pocket PC* no está diseñada precisamente para este tipo de aplicación, por lo que no encontraremos una que tenga entradas para nuestros sensores y salidas para nuestros motores. Pero cuentan también con un puerto de comunicaciones serial y son programables, cuentan con los recursos también para ejecutar el algoritmo de construcción del mapa del laberinto y de decisión de la mejor ruta. (Para mayores detalles sobre el empleo de dichos dispositivos en un robot móvil, consultar el capítulo III). Esta es una razón para verificar que nuestro MCU contará con un puerto de comunicaciones seriales compatible.

No hay que olvidar al elegir un MCU, las características con las que cuenta para periféricos. En nuestro caso, para un óptimo control de los motores, nosotros necesitamos que nuestro MCU soporte PWM. Cada vez más y más microcontroladores soportan PWM sin falta. También en nuestro caso se aprovecha el convertidor analógico – digital para los sensores.

2.3. Características del MCU PIC16F877

El Microcontrolador PIC16F877 cuenta con un CPU tipo RISC y una arquitectura Harvard.

Algunas de las características principales del microcontrolador son:

- Solo se necesita aprender 35 instrucciones de una palabra.
- Opera con un reloj de 20 MHz (cada ciclo de una instrucción es ejecutado cada 200ns).
- Todas sus instrucciones se ejecutan en un solo ciclo de reloj (excepto bifurcaciones de programa, que son de 2 ciclos).

- Hasta 8K x 14 palabras para memoria de programa (FLASH).
- Hasta 368 x 8 bytes de memoria de datos (RAM).
- Hasta 256 x 8 bytes de memoria de datos (EEPROM).
- Capacidad para interrupciones (hasta 14 fuentes diferentes).

Algunas características notables para periféricos:

- Dos temporizadores/contadores de 8 bits y uno de 16 bits.
- Dos terminales para captura, comparación o PWM.
- Convertidor Analógico – Digital de 10 bits (8 canales).
- SSP (*Synchronous Serial Port*, Puerto serial síncrono).

Algunas terminales de los puertos de entrada/salida se encuentran multiplexadas con una función alterna para las opciones con periféricos en el dispositivo.

Puertos de entrada/salida:

- Puerto A, 6 bits bidireccional, 5 canales con convertidor A/D, 1 temporizador..
- Puerto B, 8 bits bidereccional.
- Puerto C, 8 bits bidireccional, 1 temporizador, 2 terminales sirven de comparadores o para PWM, comunicaciones para puerto serie.
- Puerto D, 8 bits bidireccional, (puerto esclavo paralelo).
- Puerto E, 3 bits, 3 canales con convertidor A/D, señales de control para el puerto D cuando es utilizado como puerto esclavo paralelo.

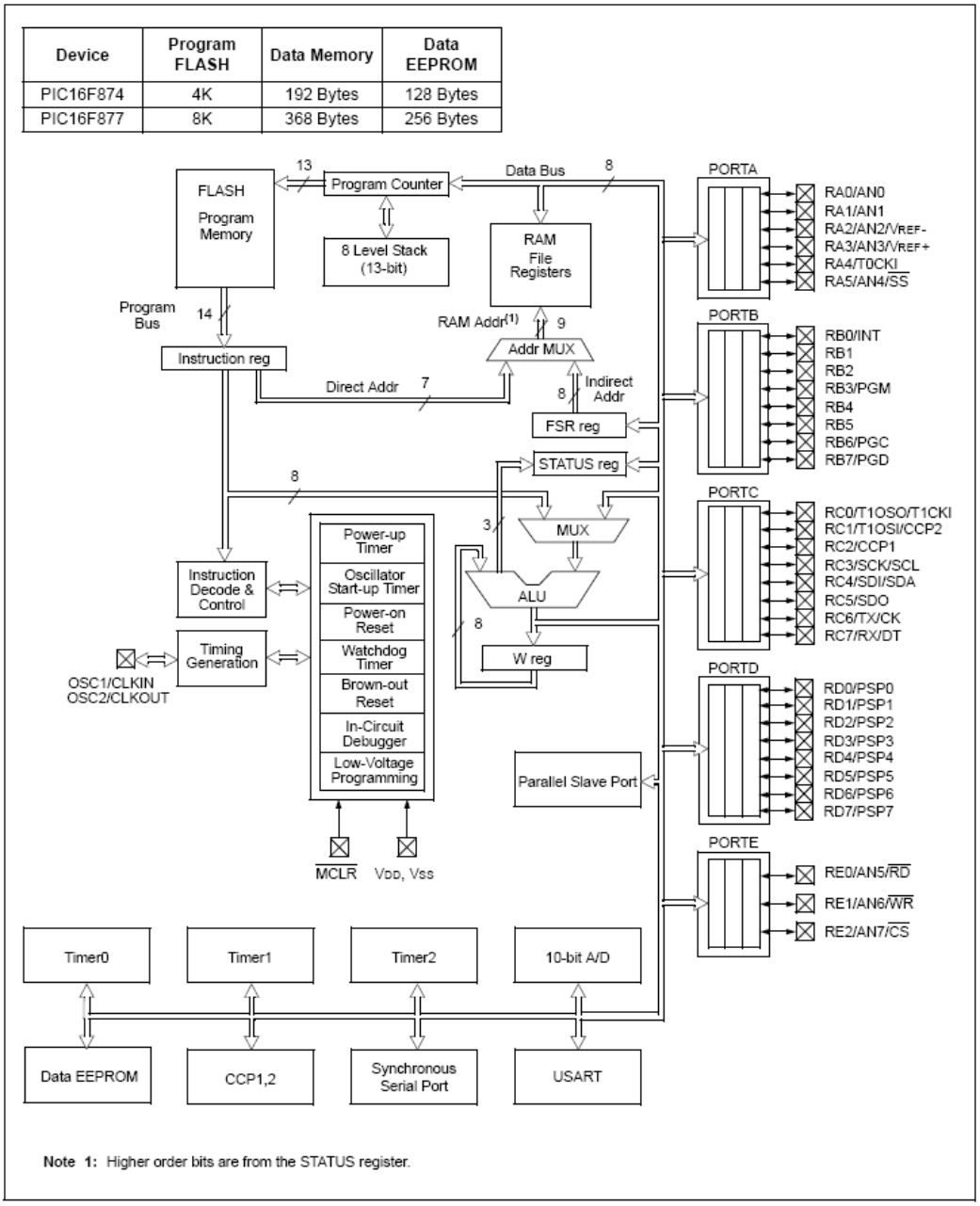

Figura 2.1. *Diagrama a Bloques del PIC16F877.*

2.4. El uso de PWM en un robot móvil

PWM (*Pulse Width Modulation*, Modulación por Ancho de Pulso) es una técnica para controlar circuitos analógicos mediante señales digitales. Es utilizada en una gran variedad de aplicaciones, que van desde medición, comunicaciones, hasta control y conversión de potencia.

Una señal analógica se caracteriza por su continuidad, la variación de voltaje y de corriente que se da en una señal de este tipo es continua, su resolución es infinita tanto en tiempo como en magnitud. Con este tipo de señales

es posible controlar cosas directamente como el volumen del radio de un automóvil.

Aunque el control analógico parezca intuitivo y simple, no es siempre económico o incluso práctico. Un problema con circuitos analógicos es que tienden a sufrir variaciones con el tiempo debido a desgaste u otras causas y resulta difícil en estos casos recalibrar el sistema. Circuitos analógicos de precisión, los cuales evitan problemas de ese tipo, tienden a ser grandes y pesados.

Controlando circuitos analógicos de manera digital, se pueden reducir los costos del sistema, así como de consumo de energía. De hecho varios microcontroladores y DSPs (*Digital Signal Processors*, Procesadores de Señales Digitales) ya incluyen en el circuito integrado controladores PWM.

En breve, PWM es una forma de codificar digitalmente, los niveles analógicos de una señal. Mediante el uso de contadores de alta resolución, el ciclo de carga de una señal cuadrada es modulado para codificar un nivel específico de señal analógica. La señal PWM sigue siendo digital, ya que dado cualquier instante de tiempo, la señal se encuentra en un nivel alto o bajo. La señal con la variación de voltaje o corriente es proporcionada al circuito analógico mediante una repetición de pulsos de niveles altos y bajos con cierto ciclo activo (figura 2.2). Dado el suficiente ancho de banda, es posible codificar cualquier valor analógico mediante PWM.

En nuestro diseño de Radamanthys, se utiliza la técnica PWM para controlar la velocidad de los motores. El uso de esta técnica fue obvio, ya que un control analógico para los motores no solo sería costoso sino también agregaría mucho peso al robot, lo cual podría ponerlo en desventaja, disminuyendo el rendimiento de los motores. Además el microcontrolador PIC16F877 cuenta con un controlador PWM para generar dos señales independientes, suficientes para controlar la velocidad de 2 motores.

Mediante el control de la velocidad en los motores, es posible crear un algoritmo para seguir la línea que forma el laberinto, sin que el robot tenga que hacer una pausa mientras avanza. Sin el uso de esta técnica, el robot tendría que hacer una pausa, hacer un giro pequeño, buscar la línea nuevamente y seguir avanzando sobre la línea.

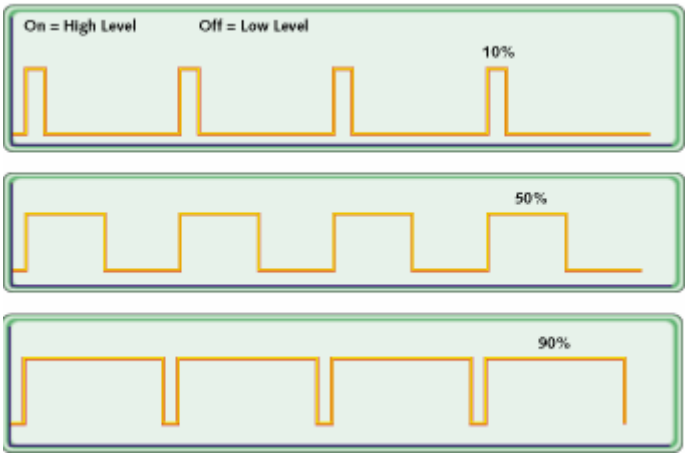

Figura 2.2. *Diferentes ciclos activos en señales PWM.*

CAPÍTULO III

EL USO DE DISPOSITIVOS MÓVILES EN ROBOTS

Por dispositivos móviles entendemos a todos aquellos dispositivos digitales que cuentan con los recursos computacionales suficientes y que son de un tamaño muy reducido. Últimamente los aparatos digitales de entretenimiento, así como de comunicaciones, están sobrados en memoria RAM y cuentan con microprocesadores lo suficientemente veloces. Todos estos dispositivos son laptops (computadoras portátiles), PDA (*Personal Digital Assistant*, Asistente Personal Digital), algunos teléfonos celulares, consolas de videojuegos portátiles (i.e. Game Boy), etc.

Los dispositivos móviles poseen mayores recursos computacionales que un microcontrolador. Cuando se desea que un robot tenga *inteligencia artificial*, es factible utilizar un dispositivo móvil para ejecutar el programa de dicho módulo. Es posible implementar ciertos algoritmos de inteligencia en un microcontrolador, añadiendo la memoria necesaria. Un dispositivo móvil presenta flexibilidad para poder programar diversos algoritmos de inteligencia artificial en un lenguaje de alto nivel dada la ventaja de sus recursos.

3.1. Características de un Asistente Personal Digital

PDA es un término utilizado para describir todos aquellos dispositivos móviles de pequeño tamaño de uso personal, con capacidades de cómputo y retención de información. Son comúnmente usados para tener una agenda e información personal a la mano.

Algunos PDA cuentan con el sistema operativo *Windows CE*, a este tipo de PDA se les asignó el nombre de *Pocket PC*. Son una alternativa en el mercado para competir con las *Palm* (también PDA, fabricadas por la marca del mismo nombre), las cuales corren el sistema operativo *Palm OS*. Una de las ventajas de las PDA que utilizan *Windows CE* es que pueden ser programadas en C++, dado que *Microsoft* distribuye un compilador para este lenguaje.

Una PDA cuenta con un microprocesador, varios periféricos: una pantalla sensible al tacto, puerto IR, parlante y módulos de memoria.

Dos microprocesadores comúnmente usados en PDA son el *Intel StrongARM* y el *Motorola DragonBall*. El microprocesador *Intel* es utilizado en PDA que corren *Windows CE*, mientras que el *Motorola* es empleado en PDA con *Palm OS*.

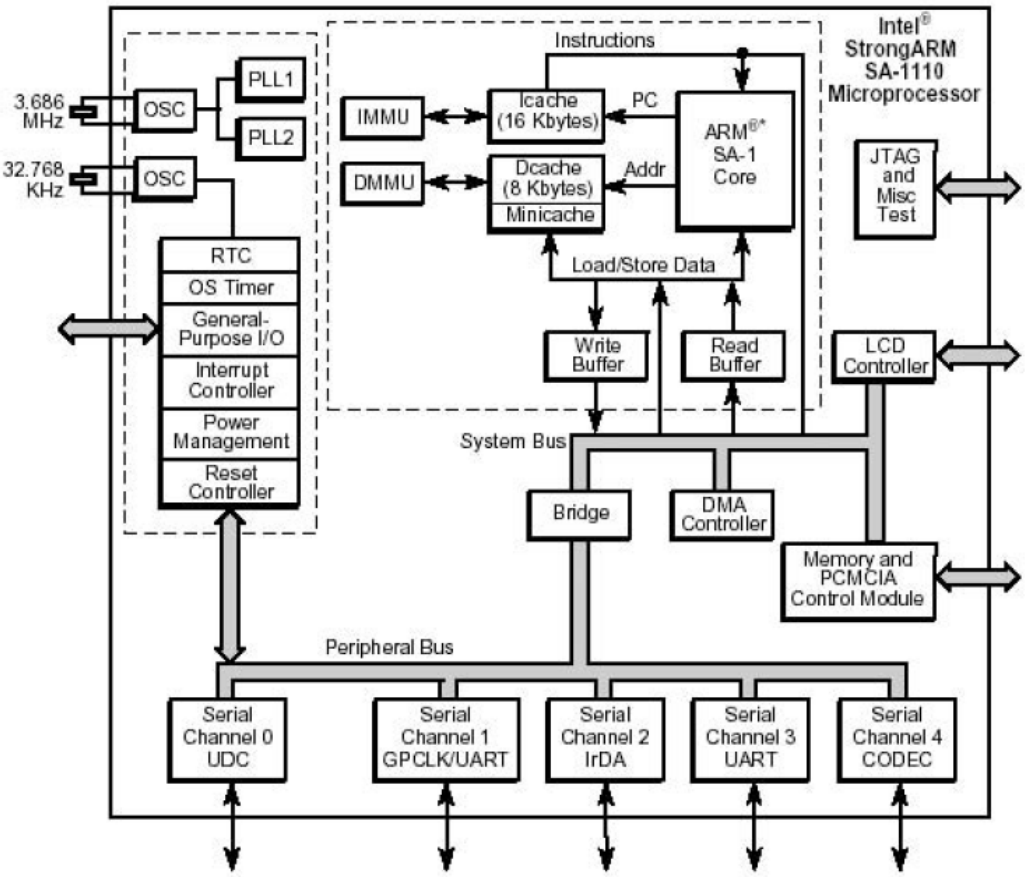

Figura 3.1. *Diagrama a Bloques del procesador Intel StrongARM SA-1110.*

Una de las características del microcontrolador PIC16F877 aprovechadas por Radamanthys es la comunicación serial. Un PDA cuenta con el mismo soporte de comunicaciones serial, haciendo posible comunicarlos mutuamente.

Así pues, tenemos un microcontrolador dedicado específicamente a leer sensores y controlar motores. Ya que el microcontrolador es quien tiene la lectura de los sensores en el diseño, aprovechamos de que este se encargue de un algoritmo para seguir la línea que forma el laberinto y dejando así libre los recursos de una PDA específicamente para el algoritmo que le dará la inteligencia al robot para poder recordar el laberinto y tomar decisiones de la mejor ruta a seguir (cuando tiene conocimiento del laberinto).

Se escogió el uso de una Pocket PC debido a la flexibilidad que estas tienen para ser programadas en C++, su práctico tamaño, poco peso y por supuesto, sus recursos. El costo de este dispositivo es una posible desventaja, todo depende del presupuesto que se cuente para el robot y la flexibilidad que el programador busque. Una alternativa es utilizar un microcontrolador con la suficiente capacidad para ejecutar los diversos módulos que el robot necesite, o utilizar un dispositivo móvil que cuente con los recursos suficientes y con el menor costo posible (por ejemplo un *Game Boy Advance*, microprocesador RISC

de 32 bits, cartucho-interfaz XRC programable en C/C++, no se necesita un microcontrolador como interfaz para sensores y efectores, la mitad del costo de un PDA y misma portabilidad). En este caso se optó por el uso de una *Pocket PC*, debido a que se cuenta con el y no se requiere hacer una inversión extra, además de la flexibilidad para programar y depurar el algoritmo y la velocidad con la que será ejecutado.

CAPÍTULO IV

RADAMANTHYS, EL ROBOT

Radamanthys cuenta básicamente de los siguientes elementos:

- 18 sensores infrarrojos para seguir la línea la cual forma el laberinto.
- Un microcontrolador PIC16F877, para la lectura de los sensores, controlar los motores, y ejecutar el algoritmo para seguir la línea.
- Una Pocket PC para la ejecución del algoritmo que permita memorizar al robot el laberinto y tomar decisiones de la mejor ruta.
- 2 motores con 2 ruedas para desplazarse por el laberinto.
- 2 encoders (discos marcados con líneas blancas y negras en las ruedas con un sensor) para permitirle al robot tener conocimiento de su ubicación.
- 8 baterías AA recargables NiMH en serie (10v).
- 2 comparadores LM311.
- 1 Driver L293D, *H-bridge* para 2 motores.
- 2 PLDs GAL22V10, para la lógica de selección de los sensores.
- 6 switches 74HC4066, para la selección de los sensores.
- 1 regulador 7805 para alimentar los motores.
- 1 regulador LM294 para alimentar la lógica se sensores y el MCU.
- Capacitores, resistencias, etc.
- Una PDA, tipo *Pocket PC*, para el algoritmo de la ruta más corta.

4.1. Diseño y desarrollo físico del robot

En el pasado evento *Robothon 2005*, los robots para la competencia *Line Maze*, no debían superar el tamaño de 6 x 6 x 6 pulgadas. Radamanthys tiene dimensiones de 5.9 x 4.5 x 4.5 pulgadas.

Desde un principio se planeó que de largo fuera casi las 6 pulgadas reglamentarias. De modo que debajo del robot se situara el arreglo de sensores y las ruedas de equilibrio. El arreglo de sensores se pensó desde el principio, acomodarlo en el centro del robot, al igual que las 2 ruedas de tracción; de manera que las 2 ruedas quedaran también en el centro. Las ruedas de equilibrio se colocarán a las orillas que guardan la mayor distancia en el robot.

En la posición donde se instalarán los motores del robot, será posible también colocar un PCB con el controlador de los motores. El circuito integrado utilizado es el L293D.

En el siguiente nivel se situarán las baterías del robot. El MCU se acomodará por encima de las baterías, con su respectivo PCB. Y por último se instalará la *Pocket PC* en la parte superior. Para comunicar el MCU de Microchip con la Pocket PC, puede consultarse un diagrama esquemático así como el código en el Apéndice B.

La jerarquía de los elementos que componen a Radamanthys puede apreciarse en la figura 4.1. Su desarrollo se centrará en tres módulos

principalmente, que son el módulo de percepción, el módulo de control y el módulo de inteligencia; este último para hacer más capaz al robot en una competencia.

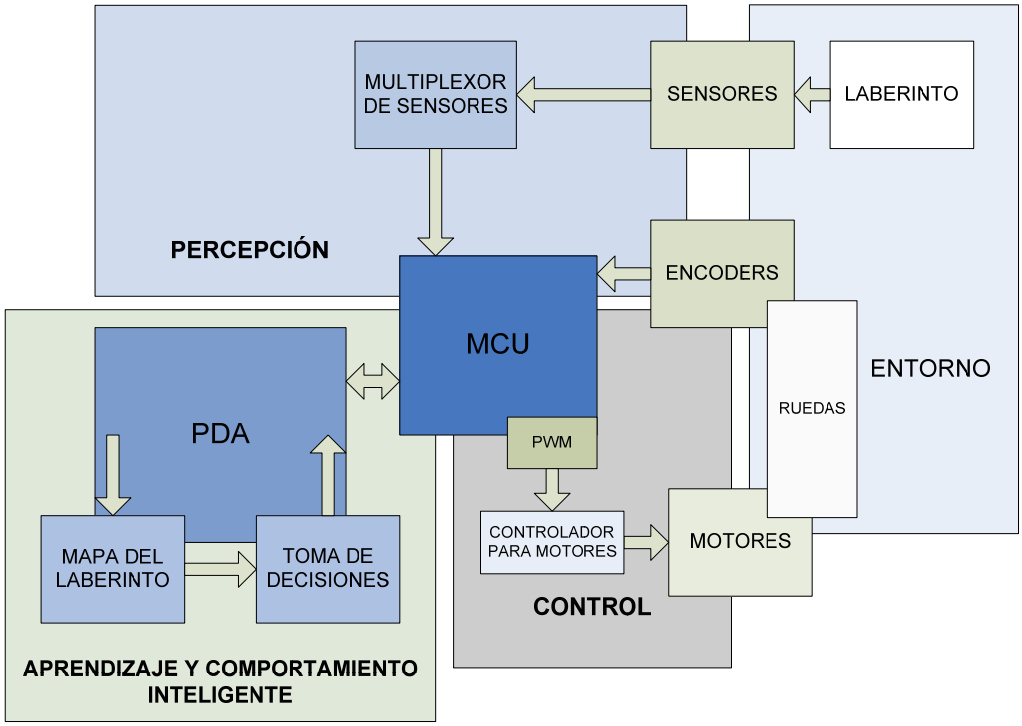

Figura 4.1. *Jerarquía de Radamanthys.*

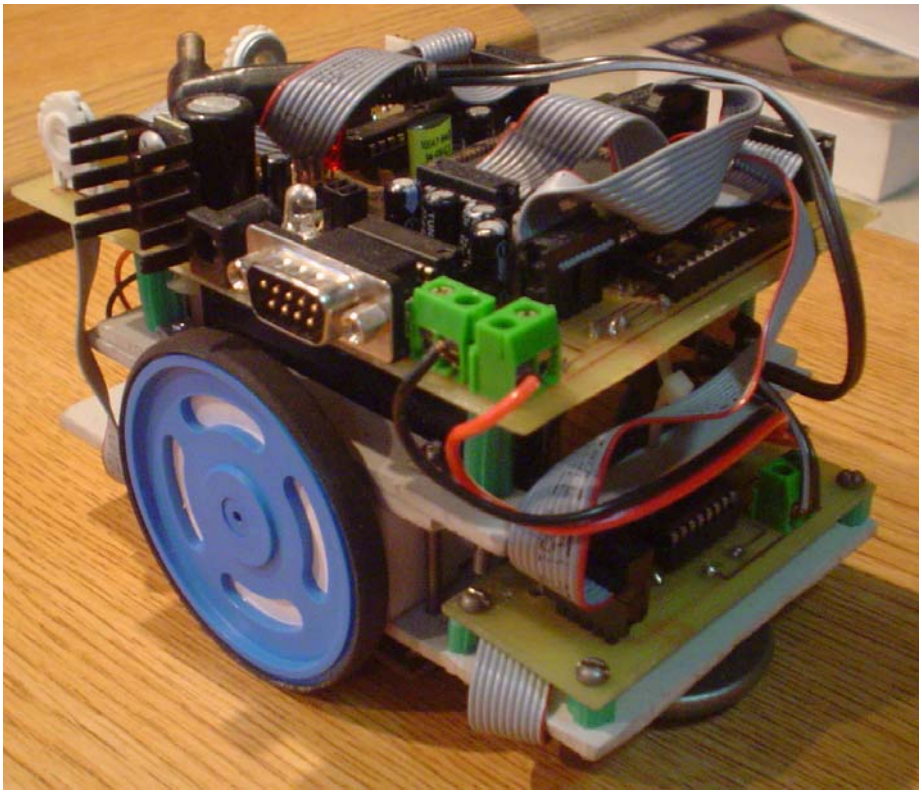

Figura 4.2. *Radamanthys, aun sin la Pocket PC.*

4.2. Organización de los sensores

En lo que se invertirá más tiempo a la hora de plantear el diseño, será en planear la ubicación de los sensores. Los sensores deberán detectar la línea que forma el laberinto así como las bifurcaciones de la mejor forma posible; en general, deben de permitir la confiable construcción de un mapa del laberinto.

La primera idea para la organización de los sensores, fue tener los sensores necesarios para detectar la línea y que el robot fuera capaz de seguirla. Para tal tarea, el robot necesitaba únicamente dos sensores. De esta forma cualquiera de los dos sensores podía detectar el momento en que la línea dejara de estar entre estos, y dependiendo de cual de los dos dejara de ver la línea, entonces reajustar la dirección del robot. Bastantes robots que están diseñados para seguir una línea de este tipo usan solamente dos sensores.

Luego para que el robot fuera capaz de detectar el momento en que la línea dejaba de ser una línea recta o se volvía una bifurcación, se pensó en ubicar un tercer sensor en medio de los dos anteriores pero en una posición más adelantada.

En la ambición para que el robot pierda el menor tiempo posible a la hora de hacer el recorrido. La idea fue hacer el arreglo de sensores simétrico, hacia delante y hacia atrás, de manera que el robot no tenga un frente definido. Tal propósito es con el fin de que el robot no tenga que dar vueltas de 180º. Si el robot encuentra el fin de la línea en cualquier momento y no hay una vuelta de 90º, entonces este simplemente iba a “retroceder” sin tener que girar sobre si mismo. Se agregó un cuarto sensor con el mismo propósito que el tercer sensor antes mencionado, para la nueva posición del robot.

Solo falta asegurarse de que el robot pueda también detectar que tipo de bifurcación o vuelta encuentra a la hora de que detecta el fin de una línea recta. Así pues se agregaron más sensores a los lados, al frente y atrás, de forma igualmente simétrica. El resultado es el mostrado en la figura 4.3.

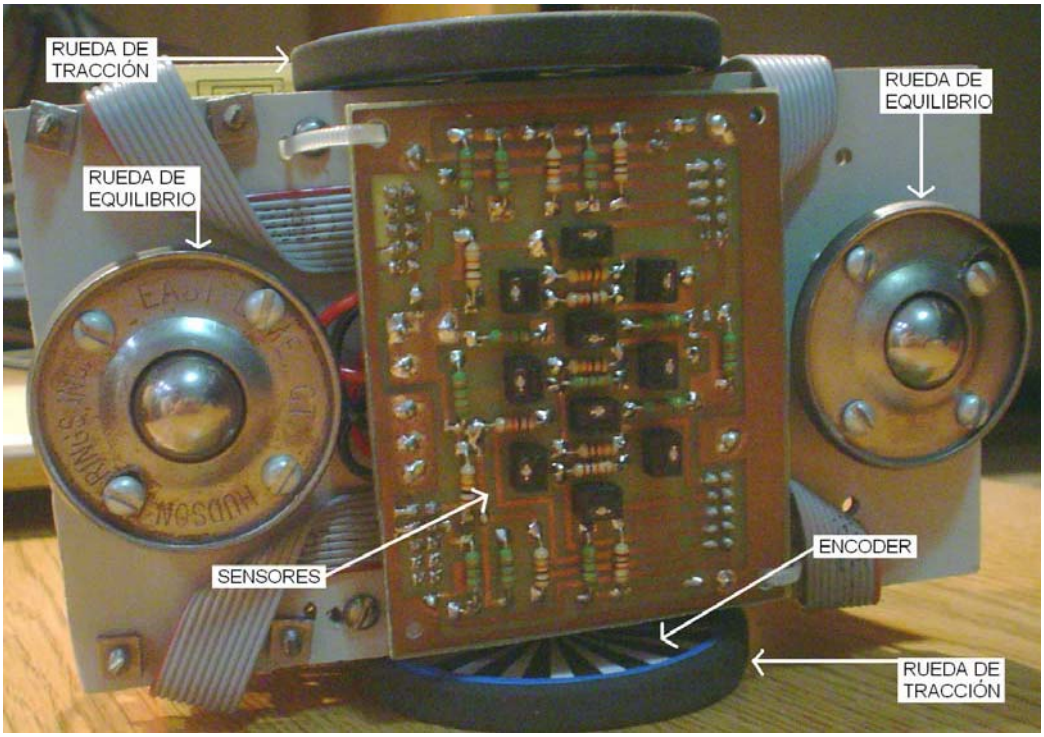

Figura 4.3. *Primera disposición de sensores, vista de la parte inferior del robot.*

Este primer arreglo de sensores mostró algunos problemas ya en la práctica. Los motores elegidos para el robot son motores bastante rápidos y esto provocó que los sensores del centro no pudieran funcionar como se esperaba; el robot perdía la línea fácilmente. Sin embargo el robot era capaz de seguir la línea mientras los motores se mantuvieran a una velocidad baja. Esto se debe además de la excesiva velocidad del robot, a que la línea para la prueba *Line Maze*, es muy delgada, en comparación de la prueba *Line Following*, donde el uso de dos sensores basta incluso cuando la velocidad del robot es muy alta.

Era necesario planear otra disposición de sensores. La idea ahora está en tratar de no perder la línea mientras el robot pueda correr a su máxima velocidad. La solución fue más simple de lo que se esperaba. El problema se solucionó organizando los sensores en línea recta y lo más juntos posibles uno del otro. La línea de sensores queda en posición perpendicular a la línea a seguir. De esta forma, el robot no pierde de vista la línea, además de que se le permite usar toda la capacidad de sus motores.

Para terminar la segunda versión del arreglo de sensores, se toman algunas cosas en consideración, como en que posición colocar la línea de sensores. Se decidió que estuviera a 2 centímetros del centro, permitiendo que cuando se detecte una bifurcación o una vuelta, el robot frene la línea del laberinto alcance a quedar aproximadamente bajo el centro del robot. El robot entonces puede girar si lo requiere, y la línea de sensores es capaz de detectar nuevamente el camino.

Para mantener la idea de simetría en el robot y evitar que el robot haga giros de 180º, se agregó una segunda línea; igualmente a 2 centímetros de distancia del centro. Esta segunda línea de sensores no solo ayuda para evitar giros de 180º, sino también a crear un algoritmo para seguir la línea del laberinto más confiable, usando ambas líneas a la vez. Los únicos momentos en los que el robot tiene que hacer giros con sus ruedas son en esquinas de 90º, el robot hace precisamente giros de 90º.

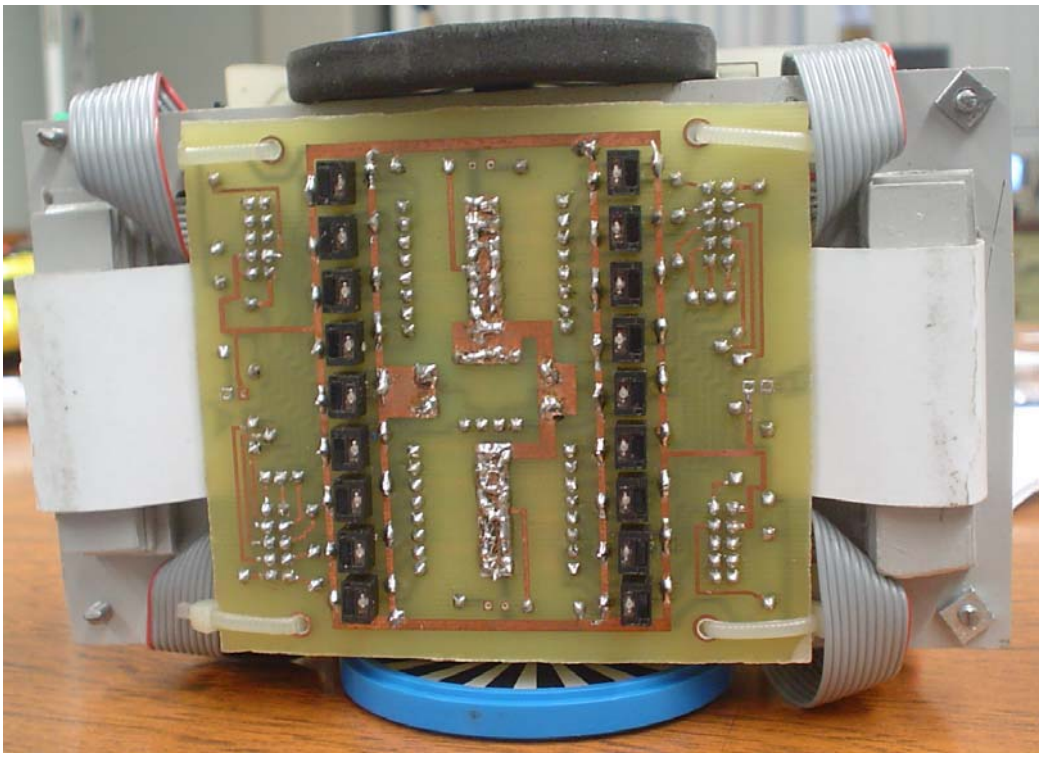

Figura 4.4. *Segunda disposición de sensores, vista de la parte inferior del robot.*

Esta nueva versión del arreglo de sensores, necesitó de un PCB (*Printed Circuit Board*, Tarjeta de Circuito Impreso) de mayores dimensiones que la primera versión. Ocasionando que las ruedas de equilibrio no se pudieran utilizar más, con tal de no superar las dimensiones del robot antes mencionadas. Se buscó una nueva forma para el equilibrio del robot, y se optó por utilizar material derrapante, en vez de ruedas. Esto además le quitó peso al robot. El resultado es el que se muestra en la figura 4.4.

La idea del primer diseño del arreglo de sensores, deja a vista que se tenía una intención también de ir planeando una programación sencilla para seguir la línea. Con la explicación de cómo se fue planteando dicho diseño, también se puede ir dando una idea de cómo sería la programación para seguir la línea con dicha disposición de sensores.

En cambio, para el segundo diseño, la percepción de la línea ahora es más eficiente. Pero para hacer realidad que el robot siga la línea, requiere de un algoritmo más complejo.

El diagrama esquemático del circuito de cada sensor se muestra en la figura 4.5, los circuitos impresos donde se soldaron todos los sensores y donde se implementa dicho circuito para cada sensor se encuentran en el apéndice C.

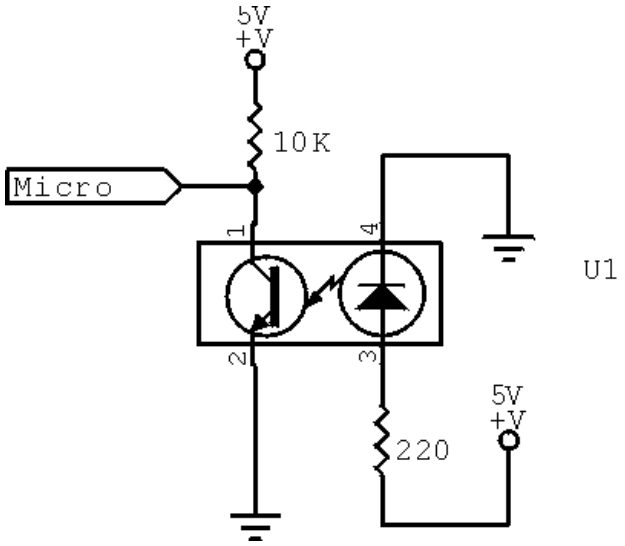

Figura 4.5. *Diagrama esquemático para conectar un sensor QRD1114.*

4.3. Señal de cada sensor hacia el MCU

Como el MCU no tiene tantas entradas, una para cada sensor; no es posible tener una lectura de cada sensor en cualquier momento. Las salidas de cada sensor, son enviadas a un switch analógico (74HC4066), y con lógica enviada desde el MCU, se selecciona el sensor a leer. Tres circuitos integrados, se pueden configurar para leer nueve sensores, teniendo así una salida. Con seis circuitos integrados se dispuso de dos salidas para los dieciocho sensores del robot, dejando al MCU la posibilidad de leer dos sensores a la vez; uno de cada línea de sensores. Dicho circuito puede apreciarse en la figura 4.6 y los circuitos impresos en el apéndice C. Para la lógica de selección del circuito se utilizó un PLD (*Programmable Logic Device*, Dispositivo Lógico Programable), el GAL22V10 se programará en modo de lógica combinacional con la siguiente tabla de verdad:

Señales desde el MCU				Selección del sensor a leer								
A	B	C	D	S1	S2	S3	S4	S5	S6	S7	S8	S9
0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	1

Tabla de verdad para la GAL22V10.

El programa en el MCU efectúa una secuencia constante para leer por breves momentos un sensor y seguir inmediatamente con el sensor adyacente. De esta manera, se tiene disponible en el momento necesitado, la información del estado entre la línea del laberinto y cada sensor.

4.4. Control de la velocidad del robot

Dependiendo de la información de cada sensor, es la salida que se envía a los motores. Cuando la línea se encuentre justo en medio del robot, ambos motores lleven el robot a máxima velocidad, y variar la velocidad de cada motor cuando la línea debajo del robot varia, reajustando así la posición del robot sobre la línea, a la vez que este no se detiene. La variación de la velocidad de cada motor al momento de reajustar su posición es mínima.

Primeramente se utilizaba en el robot el MCU *Motorola* 68HC11F1, donde la implementación de la técnica PWM presentó problemas. El control de los motores en el robot no era satisfactorio, era posible mantener el control a una velocidad baja pero no a una velocidad alta. Se buscó otro MCU que contara con el soporte PWM en hardware y que fuera confiable; así pues se optó por el MCU de *Microchip* PIC16F877. Este último MCU permitió el control de los motores a una velocidad alta apropiadamente, podía variarse el ángulo que forma el robot con una línea recta adecuadamente al avanzar, mientras que la velocidad era la deseada.

El circuito integrado L293D, cuenta con una entrada lógica para habilitar y deshabilitar cada motor, así es posible mandar una señal PWM por cada motor hacia dicho circuito integrado.

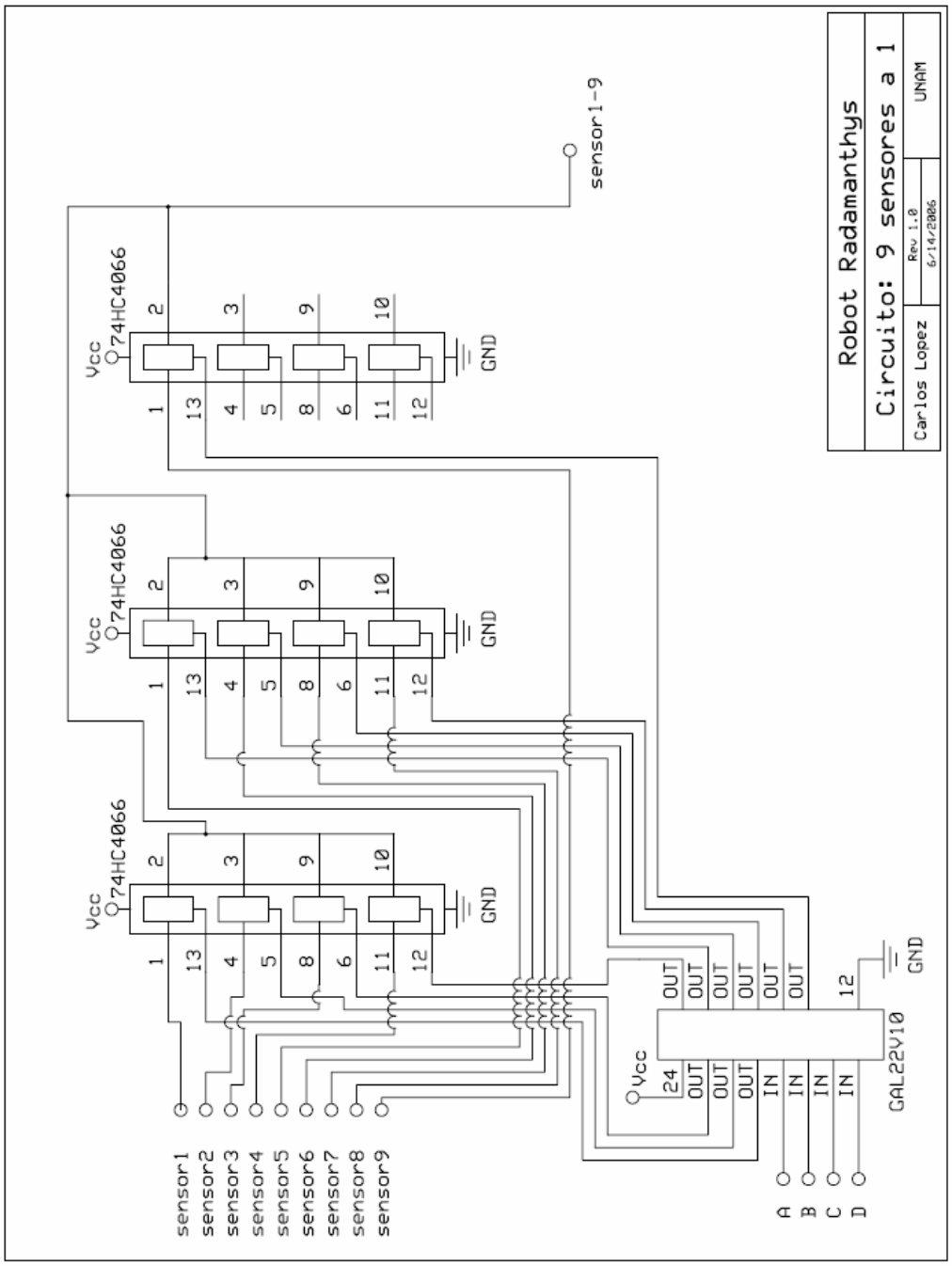

Figura 4.6. *Diagrama Esquemático: Multiplexor de sensores.*

4.5 Algoritmo para seguir la línea

Este algoritmo será programado en el MCU, es un algoritmo diferente del que la PDA ejecutará. La tarea de este algoritmo es decirle al PDA en que momento encuentra cruces, bifurcaciones, vueltas y fin en la línea, entonces el PDA le dirá al MCU como debe comportarse el robot, si debe dar un giro de 90º, la dirección del giro, si el robot debe seguir avanzando o si se debe detener.

El MCU también debe llevar la cuenta del recorrido del laberinto y del tiempo límite de 3 minutos que se le permite al robot para hacer el recorrido. El algoritmo puede apreciarse en el diagrama de flujo en la figura 4.7.

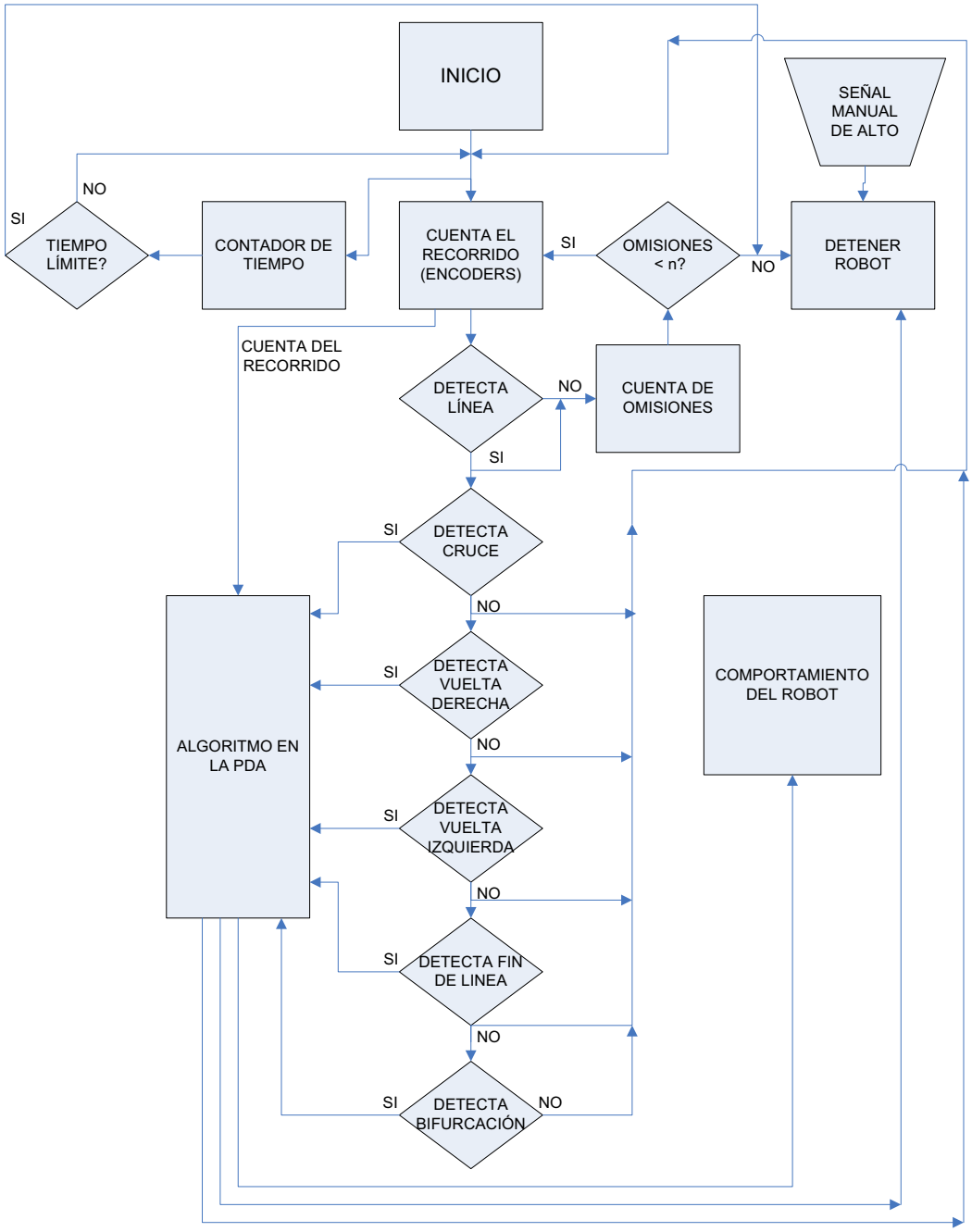

Figura 4.7. *Diagrama de flujo: Sigue la línea.*

Es considerado en el algoritmo la posibilidad de que los sensores puedan tener un mínimo de errores al detectar la línea. En el diagrama a flujo, esto es ajustado con la variable ‘n’, haciendo una cuenta de las omisiones que se dan al detectar la línea.

CAPÍTULO V

RESULTADOS Y CONCLUSIONES

El robot Radamanthys ha presentado muy buenos resultados en cuanto a velocidad y eficiencia para detectar y seguir la línea que forma el laberinto. En el primer prototipo creado, la velocidad del robot estaba limitada por dos factores. Un factor fue la organización de los sensores, era fácil perder de vista la línea que forma el laberinto si la velocidad del robot era alta. El segundo factor fue el control de los motores con el microprocesador 68HC11F1 de Motorola, no fue posible variar la velocidad de los motores adecuadamente. En el segundo prototipo la nueva organización de sensores permitió no perder de vista la línea que forma el laberinto cuando el robot avanza rápidamente y con el cambio de MCU por el PIC16F877 de Microchip se pudo controlar debidamente los motores mediante PWM.

La etapa de percepción del robot está terminada, así como su diseño mecánico y su hardware. La comunicación con la PDA también se ha podido establecer con éxito, aunque aún no se ha utilizado la PDA para afectar el comportamiento del robot.

Aún falta implementar el algoritmo que memoriza y analiza el laberinto para poder hacer el recorrido más corto. Este se implementará en un PDA (*Pocket PC*).

Este trabajo genera las siguientes conclusiones:

- El arreglo de sensores infrarrojos ideal para seguir una línea para la competencia *Line Maze*, es una línea recta de sensores lo más juntos posible uno del otro. Mientras más largo es el arreglo es mejor, aunque es suficiente con 5 sensores; dependiendo de la velocidad que el robot sea capaz de lograr.

- El uso de un microcontrolador con soporte PWM es esencial. Esto permite tener control sobre los motores y la velocidad del robot.

- El microcontrolador requiere preferentemente de soporte para comunicación serial, de esta forma es posible utilizar un dispositivo móvil para programar la inteligencia del robot.

- La experimentación con el robot en sus diversas etapas, mostró al microcontrolador PIC16F877 ser más práctico que el 68HC11F1 a la hora de desarrollar el programar para la rutina de seguir la línea e identificar bifurcaciones; así como para la implementación de PWM.

- En este tipo de robots móviles, el uso de baterías recargables AA de NiMH son ideales, ya que estas proporcionan la corriente y el voltaje suficiente usándolas en serie tanto para los motores como para la lógica. En un principio se utilizó una batería recargable de ácido de 6.3v, la cual mostró ser ineficiente; debido a que el voltaje mínimo requerido para el robot era de 5.5v, y esta se descargaba a dicho nivel muy rápido.

- El uso de un regulador de 5V: LM294 para la lógica del robot, le permite tener un mayor tiempo de actividad, que cuando se utiliza el popular 7805. Además el regulador LM294 mostró aislar la lógica, del ruido que generan los motores.

- El uso de *encoders* en las ruedas del robot, es esencial si se desea poder programar al robot para ubicar su entorno y posición.

Actualmente el robot recorre el laberinto adecuadamente, pero no memoriza el laberinto ni tampoco toma decisiones para elegir una mejor ruta. El tiempo que tarda en salir el robot de un laberinto depende de la forma del laberinto. Cuando al robot se le incorpore la PDA con el algoritmo para tomar la decisión de la mejor ruta, se espera que el robot pueda participar en competencias de manera notable.

Una vez que el robot esté terminado, no significa que ya no se puedan buscar nuevas formas para lograr mejores resultados. Como trabajo a futuro se consideran muchas posibilidades, como incorporar un PDA con cámara. Utilizar una cámara en el robot puede aprovecharse para poder ver el laberinto mientras es recorrido sin necesidad de que el robot tenga que pasar sobre cada línea del laberinto para tener conocimiento de todo el laberinto. Otra posibilidad es utilizar ruedas omnidirecionales (ruedas que permiten el avance del robot en cualquier ángulo), de esta forma se evitan giros cuando se utilizan tres o más motores con las mismas ruedas. También se pueden buscar diferentes motores para el robot, pueden utilizarse servomotores que permitan un mayor control y a la vez velocidad, o simplemente motores más veloces, aunque el costo final podrá ser obviamente más alto.

APÉNDICES

A. REGLAS PARA *LINE MAZE*
B. CONFIGURACIÓN DEL PIC16F877
C. CIRCUITOS IMPRESOS

Apéndice A

Reglas para *Line Maze*, por SRS (2005)
http://www.robothon.org/Robothon2005/maze.php

Objeto

Cada robot tiene 3 intentos para encontrar la posible ruta más rápida del principio al final del laberinto usando técnicas de seguimiento de línea. El robot que resuelve el laberinto en el menor tiempo es el ganador.

El Laberinto

El laberinto consistirá de series de líneas negras de ¼" que se intersecan entre si, en un entorno de fondo blanco. El espacio blanco será relativamente brillante y el espacio negro relativamente plano. Ninguna línea estará junta de otra línea paralela a 6", y siempre habrá un mínimo de 6" entre 2 intersecciones. Todas las intersecciones serán formando un ángulo recto. Habrá caminos que formen lazos, resultando en múltiples posibles rutas del inicio al final. Una marca circular sólida negra de 6" de diámetro indicará el final del laberinto. Caminos sin final no son marcados; la línea simplemente termina en dicho caso. El punto de inicio es marcado con una T de 2" de ancho, y los robots deben de iniciar su recorrido teniendo una porción de su cuerpo sobre la T. Todos estos detalles se muestran en la figura A.

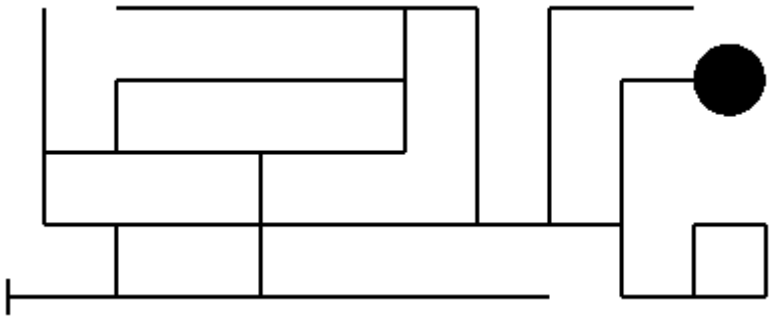

Figura A.1. *Ejemplo de un laberinto pequeño.*

Como el laberinto puede tener caminos con lazos de regreso a una intersección, el robot debe saber si ya ha visitado previamente una intersección. Métodos tradicionales como "mano derecha – mano izquierda" para revolver laberintos no funcionarán con este tipo de laberinto, ya que el robot puede quedarse atorado dentro de un lazo.

El laberinto no rebasará las dimensiones de 8" por 8", y ninguna línea estará a menos de 6" de la orilla del laberinto. No hay garantía de que las intersecciones estarán separadas 6" exactamente una de otra. Sin embargo, se hará todo lo posible para tener el laberinto plano y nivelado. Los participantes,

tendrán que estar preparados para irregularidades en la superficie, como donde 2 secciones del laberinto se juntan.

La disposición del laberinto no será conocida hasta después de que todos los robots hayan sido registrados y presentados al jurado del evento. Sin embargo, un pequeño ejemplo del laberinto estará disponible antes del evento para calibración y pruebas. El laberinto de prueba estará hecho de los mismos materiales que el laberinto de tamaño completo para la competencia, y será colocado en condiciones de alumbrado similares.

El Robot

El robot no puede ser más largo de 6" en cualquier dimensión (i.e. tiene que caber dentro de una caja cuadrada de 6"). No hay restricción en cuanto al peso. El robot no puede expandirse a más de estas dimensiones en todo momento durante el evento.

Se le permite al robot sensar líneas diferentes a la que se encuentre navegando, mientras dichos sensores no se extiendan sobre las dimensiones antes mencionadas. En otras palabras, se puede disponer de una cámara para obtener información del laberinto, mientras que la cámara esté montada completamente en el robot y dentro del cubo de 6" antes especificado.

El robot tiene que ser completamente autónomo y contenido en si; computadoras externas no son permitidas. Los robots no pueden dejar rastros o marcas. Tampoco pueden separarse en varios robots. El robot no puede abandonar el laberinto en cualquier momento. Si lo hace, dicho intento será detenido y el robot no obtendrá puntuación por tal motivo. Se considerará que un robot habrá abandonado el laberinto si ninguna parte del cuerpo se encuentra sobre la línea que navegaba.

Operación

Todos los robots competidores serán llevados con el jurado del evento, antes de que el laberinto pueda ser mostrado. Ninguna modificación de todo tipo se puede hacer al robot después de que este ha sido llevado al jurado, incluyendo cambios en el software o la estrategia (esto incluye el cambio de configuración por medio de switches).

Cada robot cuenta con 3 minutos para encontrar el camino hasta el final del laberinto. El tiempo se cuenta desde que el robot inicia el recorrido y termina cuando cualquier parte del cuerpo del robot toca el círculo final. El robot puede continuar explorando el laberinto después de haber encontrado el final, pero tiene que detenerse por su propia cuenta antes de que concluyan los 3 minutos de recorrido. Si un robot continúa su recorrido pasados los 3 minutos, será detenido manualmente y recibirá una penalización de 30 segundos.

Cada robot cuenta con 3 intentos para resolver el laberinto. A los robots se les permite (y se les alienta) a poder recordar la geometría del laberinto que han explorado y usar tal información para posteriores recorridos. No se permiten

hacer modificaciones de cualquier tipo al robot entre cada recorrido. Cada intento recibirá una puntuación igual al tiempo que le tome al robot el recorrido más las penalizaciones. La menor puntuación será asentada como la puntuación final del robot.

Victoria

El robot con la menor puntuación (el menor tiempo de recorrido) es el ganador. Los robots que hayan completado el recorrido del laberinto serán clasificados del menor tiempo al mayor. Después son clasificados en base a la distancia que les faltó por recorrer hasta el final (sobre las líneas que forman el laberinto); esta determinación será hecha por los jueces.

Uso

Otros clubes u organizaciones no lucrativas pueden usar estas reglas y el nombre "SRS Line Maze" siempre y cuando:

- SRS es notificada de los eventos que se vayan a llevar a cabo.
- Las reglas no sean cambiadas.
- Se le de crédito al SRS (con el nombre completo: "SRS Line Maze" es suficiente).

Es intención del SRS, mantener estas reglas actualizadas según las posibilidades y los avances que muestren los robots. Los cambios serán mencionados una vez al año, poco después del SRS Robothon. Preguntas o comentarios acerca de estas reglas pueden ser ingresados a contests o a SeattleRobotics Yahoo Group.

La última versión de estas reglas pueden ser vistas en:
<http://www.robothon.org/robothon/maze.php>.

Apéndice B

Configuración del PIC16F877
Para comunicación serie con la Pocket PC

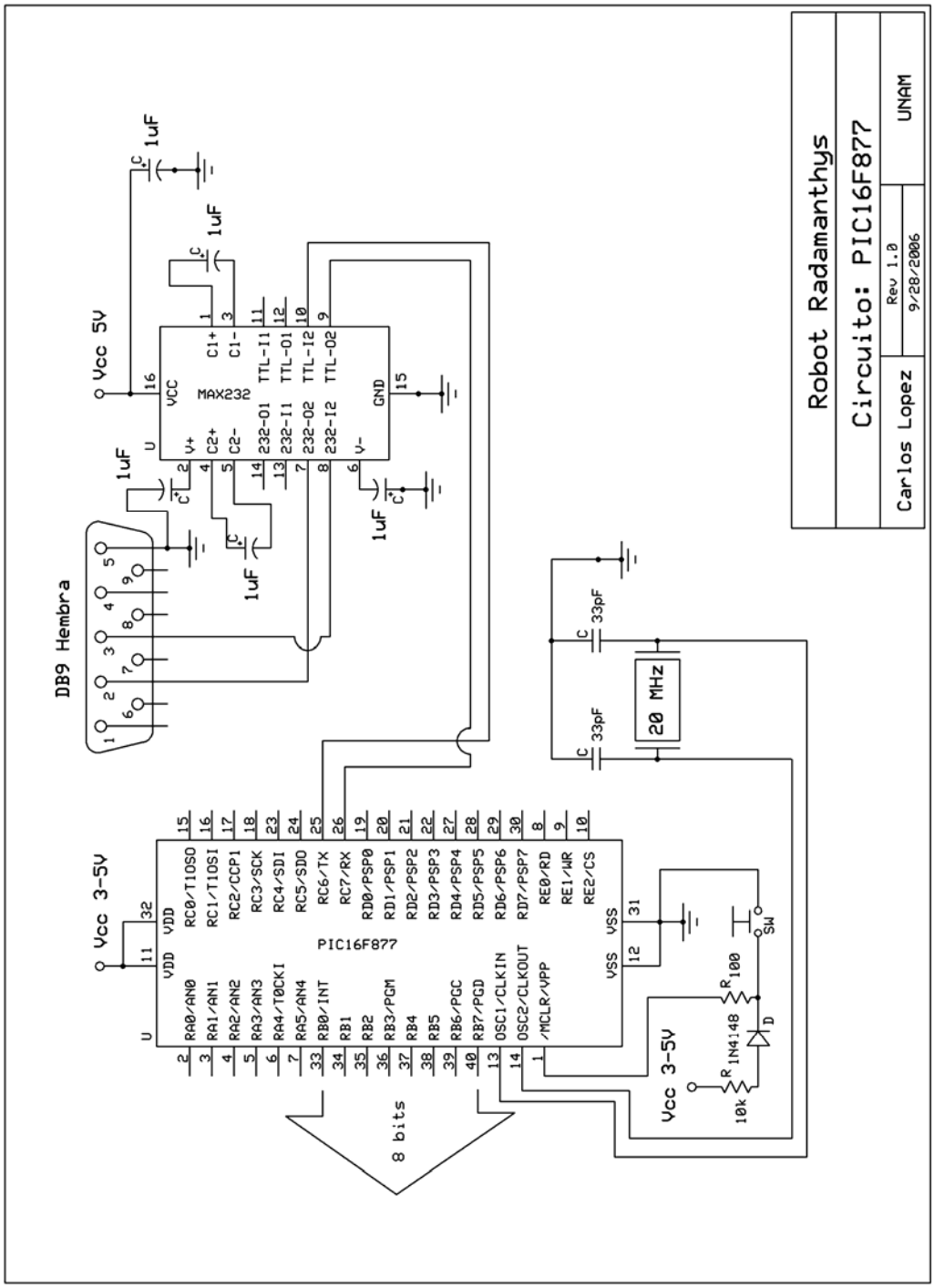

Figura B.1. *Diagrama esquemático: PIC16F877.*

```
; Programa del PUERTO SERIE en modo ASINCRONO empleando el modo
; USART (Trabaja con rs232), para la comunicación con la Pocket
; PC. Se calculo el valor de SPBRG dec = 19 para que con cristal
; de 12 MHz funcione la comunicación a 9375 BAUDIOS teniendo un
; error de 2.34% (9600 BAUDIOS).

            LIST        P = 16F877A       ;Se indica el
                                          ;tipo de procesador
            RADIX       HEX               ;Sistema de numeración
                                          ;hexadecimal
            INCLUDE     "P16F877A.INC"    ;Se incluye la
                                          ;definición de los
                                          ;registros internos en
                                          ;una librería
            ORG         0x00              ;Inicio en el vector de
                                          ;Reset
            goto        INICIO            ;Va a la primera
                                          ;instrucción del
                                          ;programa
            ORG         0x04              ;Vector de interrupción
                                          ;Se trasmite vía Serie
                                          ;el dato que está en el
                                          ;registro W
TX_DATO     bcf         PIR1,TXIF         ;Restaura flag del
                                          ;transmisor
            movwf       TXREG             ;Mueve el byte a
                                          ;trasmitir al registro
                                          ;de transmisión
            bsf         STATUS,RP0        ;Selecciona
                                          ;el Banco 1
            bcf         STATUS,RP1
TX_DAT_W    btfss       TXSTA,TRMT        ;¿Byte transmitido?
            goto        TX_DAT_W          ;No, Esperar
            bcf         STATUS,RP0        ;Si vuelve a Banco 0
            return

; Tratamiento de interrupción

INTER       btfss       PIR1,RCIF         ;¿Interrupción por
                                          ;recepción?
            goto        VOLVER            ;No, falsa interrupción
            bcf         PIR1,RCIF         ;Si, reponer flag
            movf        RCREG,W           ;Lectura del dato
                                          ;recibido
            movwf       PORTB             ;Visualización del dato
            call        TX_DATO           ;Transmisión del dato
                                          ;como ECO

VOLVER      retfie

; Comienzo del Programa principal

INICIO      clrwdt                        ;Refresca watchdog
            clrf        PORTB             ;Limpia salidas
            clrf        PORTC
            bsf         STATUS,RP0        ;Cambio al banco 1
            bcf         STATUS,RP1
            clrf        TRISB             ;Puerta B como salidas
            movlw       b'10111111'       ;RC7/Rx entrada RC6/Tx
                                          ;salida
            movwf       TRISC                   ;
            movlw       b'11101111'       ;Predivisor de 128
                                          ;asociado
            movwf       OPTION_REG        ;Al watchdog
            movlw       b'00100100'       ;Configuración de USART
            movwf       TXSTA             ;y activación de
                                          ;transmisión
            movlw       0x13              ;9600 BAUDIOS
            movwf       SPBRG
```

```
            bsf         PIE1,RCIE               ;habilita
                                                ;interrupción en
                                                ;recepción
            bcf         STATUS,RP0              ;Cambio a Banco 0
            movlw       b'10010000'             ;Configuración
                                                ;del USART
            movwf       RCSTA                   ;Para recepción
                                                ;continúa puesta
                                                ;en ON
            movlw       b'11000000'             ;Habilitación de
                                                ;las
                                                ;interrupciones
                                                ;en general
            movwf       INTCON
BUCLE       clrwdt
            goto        BUCLE                   ;Refresca el
                                                ;watchdog
SPBRG       END
```

Apéndice C

Circuitos Impresos

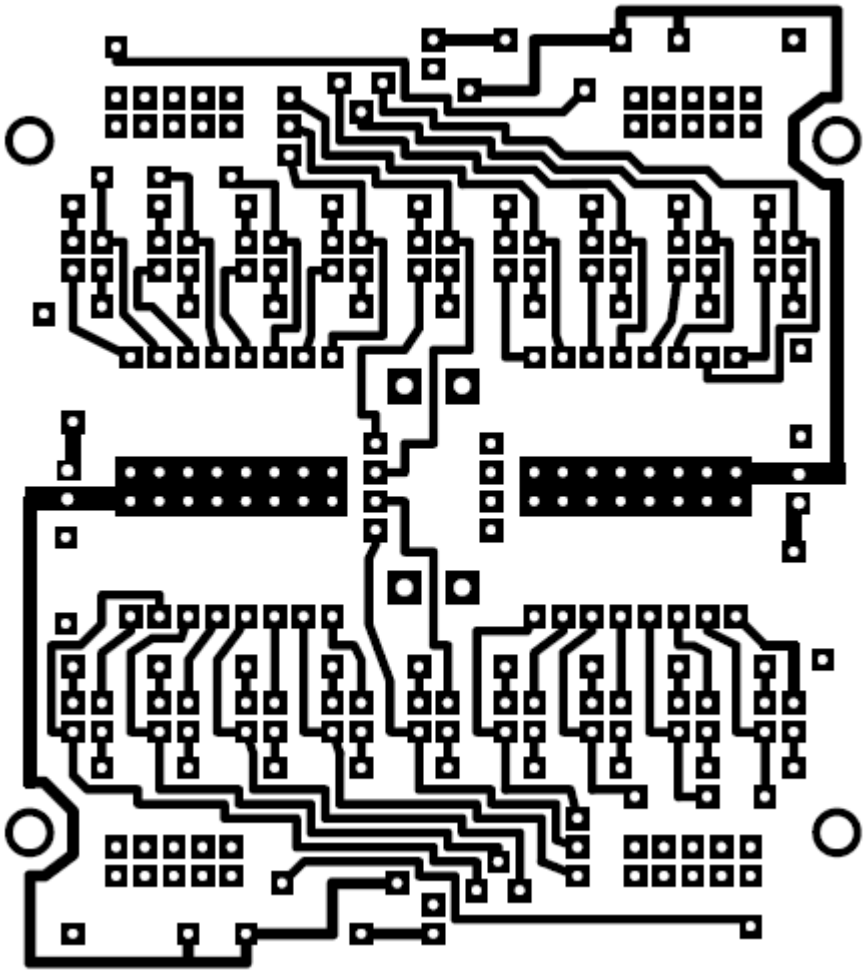

Figura C.1. *PCB Doble cara para el arreglo de sensores.*
(Lado para soldar los sensores).

El PCB tiene también el circuito para conectar los 2 sensores que se usan para los *encoders* con sus respectivas resistencias.

En un principio se pensó que la luz de algunos súper leds, podrían ayudar a los sensores para tener mejor respuesta. Esto es hasta cierto punto cierto, pero cualquier cambio de luz exterior seguía afectando de gran manera a los sensores. Este PCB se realizó para colocar 4 súper leds, los cuales ya no fueron soldados. En la figura 4.2 y 4.3 se aprecia al robot aun sin *foamy* colocado alrededor de los sensores, para aislarlos de luz externa.

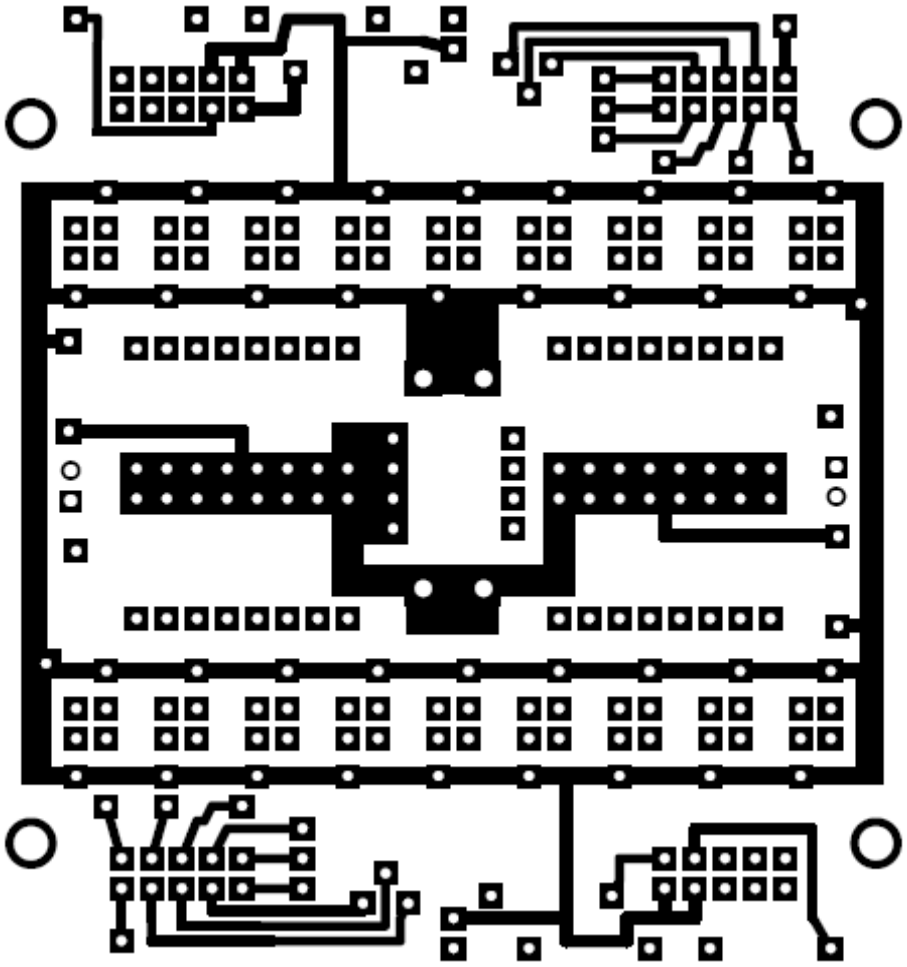

Figura C.2. *PCB para el arreglo de sensores.*
(Lado para soldar terminales para conectar cables planos hacia el MCU).

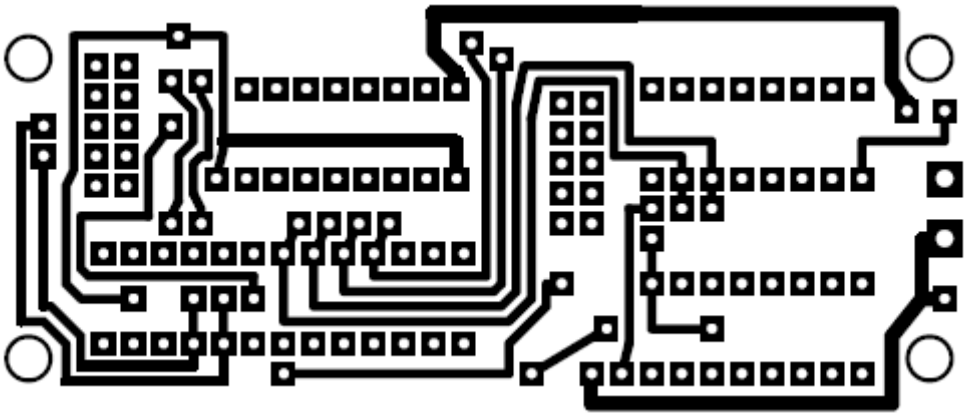

Figura C.3. *PCB doble cara para colocar 3 circuitos 74HC4066 y un GAL22V10*
(lado para colocar los componentes).

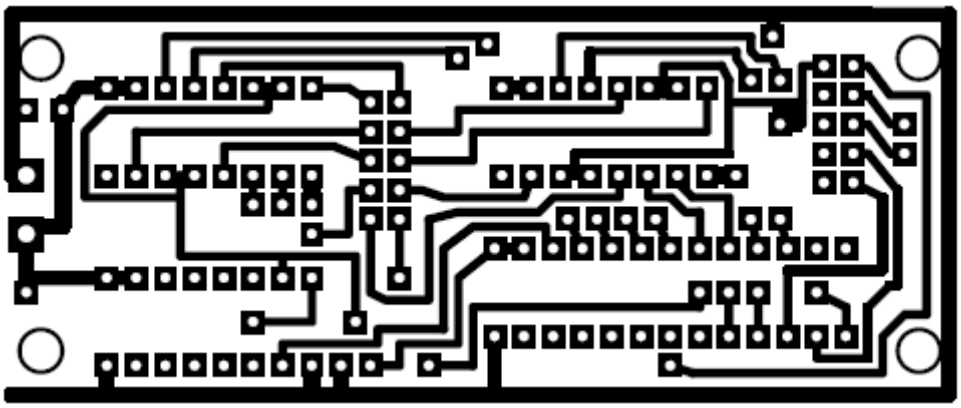

Figura C.4. *PCB doble cara para colocar 3 circuitos 74HC4066 y un GAL22V10 (lado para soldar los componentes).*

Este PCB se imprimió 2 veces, para utilizar en total 6 cicuitos integrados 74HC4066. Cada GAL22V10 se programa de diferente forma, dependiendo del orden que se le quiera dar a cada línea de sensores. La programación de cada GAL es en modo combinacional (sin reloj), con 4 señales de entrada (selección) y 9 de salida (para habilitar específicamente un sensor).

El PCB cuenta con 2 conectores para cable plano de 10 hilos. El conector del centro, se conecta al PCB de sensores. El conector de la orilla, se conecta al PCB del MCU.

Cada circuito integrado lleva a su izquierda un capacitor cerámico de 0.1 μF, como acoplador. Se utiliza un PCB para cada línea de sensores (9 de 18).

BIBLIOGRAFÍA

[1] Ben-Zion Sandler, *Robotics – Designing the mechanisms for automated machinery*, second edition, Academic Press, 1999.

[2] Rolland Siegwart, Illah R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, The MIT Press, 2004.

[3] Doug Williams, *PDA Robotics – Using your personal digital assistant to control your robot*, McGraw-Hill, 2003.

[4] *Data Acquisition Toolbox For use with Matlab – User's Guide v2*, The Math Works Inc., 2000.

[5] Jack Ganssle, Michael Barr, *Embedded Systems Dictionary, Introduction to Pulse Width Modulation*, CMP Media, LLC, 2001.

[6] José María Angulo Usategui, Ignacio Angulo Martínez, *Microcontroladores PIC: Diseño práctico de aplicaciones segunda parte: PIC16F87X*, McGraw-Hill, 2000.