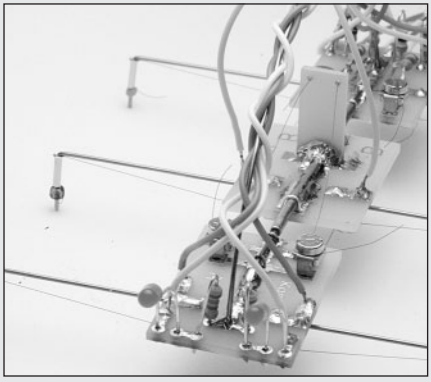
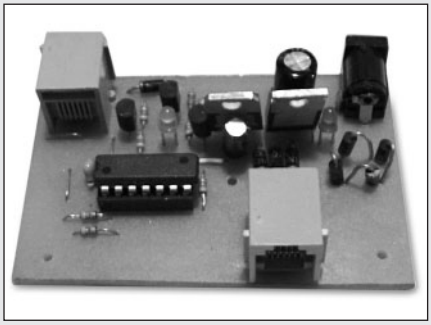
ROBÓTICA

WWW.AUTOAPRENDIZAJE.INFO

ROBOTICA

Contenido

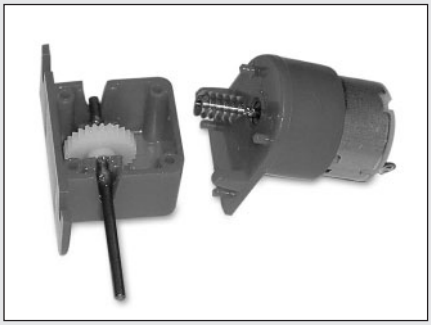




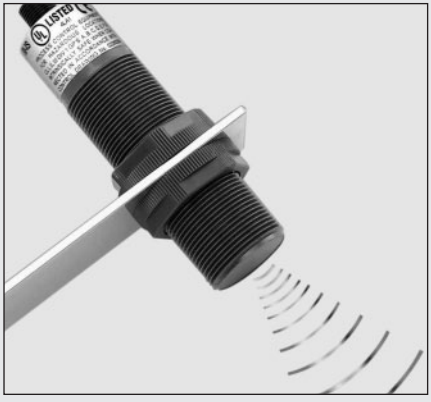
Capítulo 4

Capítulo 5

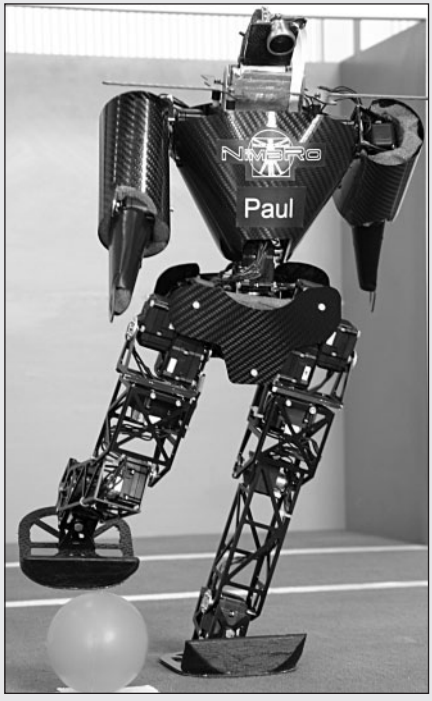
Capítulo 6



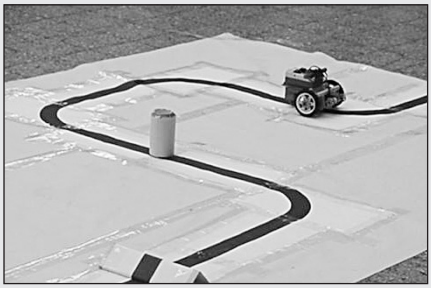
Capítulo 7



Capítulo 8



Capítulo 9



Capítulo 10

Apéndice A

Apéndice B

Servicios al lector

Introducción

Decir que la robótica es un tema del futuro es una prueba fehaciente de miopía. Los robots están entrando por nuestras puertas, ventanas, cerraduras, y es inevitable que así sea. Desde los lavarropas hasta los vehículos espaciales utilizan tecnologías muy vinculadas con la disciplina. Seguramente, comenzaron con investigaciones que en ese momento parecían desconectadas de la realidad (vieja excusa que utilizamos para aquellos que entran a nuestros laboratorios y nos preguntan ¿y eso que hacen para qué sirve?). Es por eso que después de algunos años de investigación sobre este tema, y luego de escribir algunos artículos inentendibles, tuvimos ganas de abrir el juego. La gente de esta editorial nos invitó a presentar el tema en forma más amena y al alcance de más lectores.

Mi relación con la robótica comenzó desde el ámbito educativo como un recurso concreto y tangible para enseñarles programación a jóvenes entre 13 y 17 años. En mi experiencia como docente de programación había llegado a una conclusión terrible: cuando enseñamos a programar, dentro del aula tenemos dos grupos de alumnos. Los que entienden todo sin que se les explique demasiado porque tienen una suerte de gen del programador y, con sólo recibir un par de ideas y algunos apuntes, al cabo de un mes regresaban con un sistema de control de centrales termonucleares. Y el otro conjunto de alumnos, más abultado, que a pesar de todos los recursos, inventos y triquiñuelas didácticos, no lograban superar el PRINT "HOLA MUNDO". Por lo tanto, mi tarea como docente era inútil en los dos casos: en el primer grupo, porque no me necesitaban, y en el segundo, porque no les podía aportar cosas nuevas. En ese momento comencé a buscar algún mecanismo para que nadie se diera cuenta de esto y, a continuación, un recurso que permitiera mejorar la enseñanza de programación. Allí me crucé con los robots y con los entornos de objetos. Con el primer tema me casé y con el segundo tengo una relación de amante que espero oficializar en un próximo libro.

Tengo la suerte de que en las últimas investigaciones que he desarrollado, ambos aspectos se han unido y manejamos los robots desde un ambiente de objetos. La robótica me ha dado inmensas satisfacciones, me ha permitido viajar a lugares que nunca creí que iba a conocer y ha pagado alguna que otra comida mía y de mi familia. Pero ante todo, me ha brindado la posibilidad de sorprenderme día tras día. Siempre hay algo nuevo, siempre hay otro desafío. Espero que este libro sirva como un punto de partida sencillo para viajar a obras mucho más completas que nos regalarán experiencias maravillosas.

Conceptos fundamentales

¿Qué es un robot? ¿Cuándo conviviremos con ellos? ¿Tendremos diversos tipos de robots entre nosotros? ¿Lavarán los platos? En este primer capítulo intentaremos responder estas y otras preguntas, con el objetivo de entender cuáles son los alcances posibles de nuestro primer proyecto robótico.

INTRODUCCIÓN A LA ROBÓTICA

Día a día, nos sorprendemos con las noticias que aparecen en los medios de comunicación vinculadas a la presencia de robots en diversos campos de la vida cotidiana. Robots enfermeros, mascotas, repositores de supermercados, detectores de explosivos, aspiradoras hogareñas, o simples jugadores de fútbol, son algunos de los ejemplos que podemos encontrar en el mercado de la tecnología de última generación.

En síntesis, **la robótica ya no es parte de nuestro futuro sino de nuestro presente tangible**. Sin embargo, probablemente gracias a la literatura y al cine de ciencia ficción, el concepto de lo que es un robot, sus posibilidades y sus limitaciones reales están desdibujados en el imaginario colectivo. Es por eso que en este primer capítulo haremos una introducción a los conceptos fundamentales de la robótica.

¿Qué es un robot?

Sueño de muchas generaciones, la explosión tecnológica nos ha puesto al alcance de poder concretarlo.

Para comenzar nuestro recorrido, hagamos un repaso de la historia de la robótica para comprender hacia dónde nos dirigimos.

***Figura 1. Roomba**, la aspiradora robótica más popular y económica del mercado.*

A imagen y semejanza

Desde los orígenes del hombre, podemos encontrar varios relatos sobre la creación de vida artificial. Por ejemplo, en la leyenda del Golem, un rabino de Praga le infunde vida a una estatua de barro. Asimismo, en la obra literaria Frankenstein, el doctor de dicho nombre crea un ser a partir de órganos de otras personas, que luego se vuelve contra él.

Si nos apartamos de la literatura, en el antiguo Egipto encontramos estatuas de dioses que incorporaban brazos mecánicos operados por los sacerdotes. En el siglo XIX, también se hicieron conocidas las creaciones de robots que jugaban ajedrez, aunque en realidad éstos ocultaban a un ser humano de baja estatura que operaba la máquina desde su interior (**Figura 2**). Es decir, el deseo de creación de un ser a nuestra imagen y semejanza está presente desde los primeros tiempos de la humanidad.

El origen de la palabra **robot** se remonta a comienzos del siglo XIX. El dramaturgo **Karel Capek** utilizó por primera vez este término en su obra **Opilek** para referirse a un conjunto de máquinas inventadas por un científico para realizar tareas pesadas y aburridas. En checo, idioma original de la obra, el término **robota** significa **trabajo tedioso**. Pero fue el escritor **Isaac Asimov** quien popularizó el término e introdujo el concep-

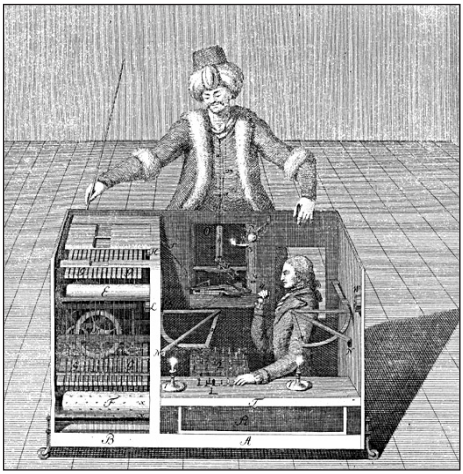


Figura 2. *El Turco, un robot que simulaba jugar al ajedrez y que, en realidad, tenía un jugador humano adentro.*

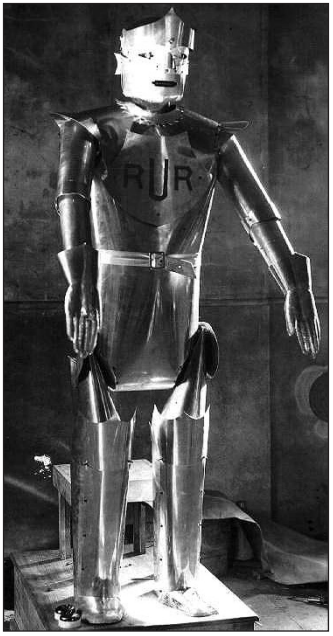


Figura 3. *Éste es un robot que aparecía en la adaptación de la obra de Karel Capek:* ***Rossum's Universal Robot****.*

to de **robótica** en diversos relatos de ciencia ficción de su autoría. En sus obras, Asimov muestra facetas humanas de los robots y define un conjunto de leyes para que estos seres nuevos nunca se rebelen contra los humanos. Luego, el cine y la televisión generaron cientos de robots de diversa índole, algunos simpáticos (como R2D2 y Cortocircuito), y otros definitivamente en contra de sus creadores (Terminator, HAL).

A partir de la creación de las primeras computadoras comenzó el verdadero desarrollo de los robots primitivos. En 1974, la empresa **Cincinnati Milacron** realizó el primer robot industrial, conocido como **The Tomorrow Tool**. A partir de ese momento, junto con la evolución de los sistemas de procesamiento, el crecimiento de la robótica ha sido exponencial. La reducción de tamaño y de costos, y el aumento de la capacidad de cálculo de los procesadores, han permitido la creación de robots cada vez más sofisticados, rápidos y autónomos. Sin embargo, aún estamos lejos de crear un robot a nuestra imagen y semejanza.

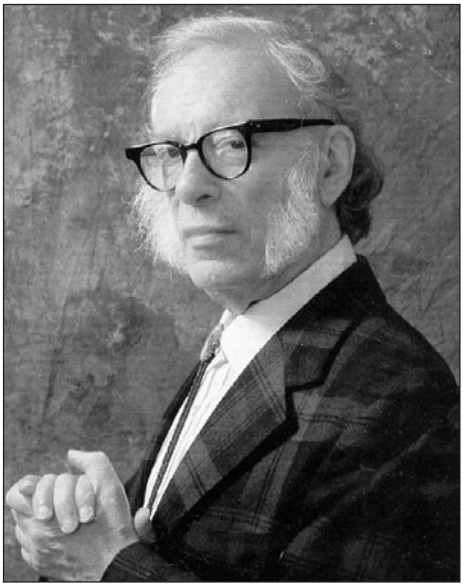


Figura 4. *Isaac Asimov, creador de cientos de cuentos sobre robots y de la palabra robótica.*

Definición de la palabra robot

Existen muchas definiciones de la palabra robot. En cada una de ellas, encontramos destacado algún aspecto en particular, que es el que cada autor quiere resaltar en su obra. Según la Asociación Japonesa de Robótica Industrial (**JIRA**), los robots son *dispositivos capaces de moverse de modo flexible, análogo al que poseen los organismos vivos, con o sin funciones intelectuales, lo que permite la realización de operaciones en respuesta a órdenes recibidas por humanos*. Vemos que en esta definición se encuentra resaltada la capacidad de movimiento de los robots y su analogía con los seres de la naturaleza. Sin embargo, a la JIRA no le interesa la inteligencia artificial aplicada al robot, dado que su función fundamental es ser operado por un humano. Por su parte, el Instituto de Robótica de Nor-

teamérica (**RIA**) define a un robot industrial como un *manipulador multifuncional y reprogramable diseñado para desplazar materiales, componentes, herramientas o dispositivos especializados por medio de movimientos programados variables, con el fin de realizar diversas tareas*. En este caso, el acento está puesto en la capacidad de programación del robot y, por lo tanto, en cierta independencia de funcionamiento con respecto a la operación humana. Como dijo Joseph Engelberg, padre de la robótica industrial: ***es posible que no sea capaz de definir qué es un robot, pero sé cuándo veo uno***.

Particularmente, y ya que nos hemos ganado el derecho dado que estamos escribiendo un libro sobre robótica, agregaremos una definición más de robot a la larga lista preexistente: **un robot es un dispositivo con un determinado grado de movilidad, que puede realizar un conjunto de tareas en forma independiente y que se adapta al mundo en el que opera**. El objetivo de esta definición es comenzar a introducirnos en el tipo de robot sobre el que vamos a centrarnos en el desarrollo del libro.

Tipos de robots

De la misma manera que con las definiciones, podemos encontrar muchas clasificaciones distintas de robots. En esta obra, al presentarlos, intentaremos acercarnos a los diversos problemas mecánicos, electrónicos y de software que encontramos en el desarrollo de un robot. Las clasificaciones elegidas son:

Según el uso del robot

A continuación presentaremos una clasificación posible de los robots según su utilidad específica.

- **Industriales:** se utilizan dentro de un proceso de trabajo industrial. Es el tipo de robot que más ha sido desarrollado en la historia (**Figura 5**).
- **Espaciales:** deben desenvolverse en zonas inexploradas y a larga distancia de su centro de control.
- **Médicos:** son utilizados como apoyo en la intervención médica

UN ROBOT QUE JUGABA AL AJEDREZ

Uno de los primeros **robots reales** fue el jugador de ajedrez autómata de **Wolfgang von Kempelen**, en 1769. Éste consistía en una cabina de madera de 1,20 metros de largo, 60 centímetros de profundidad y 90 centímetros de altura. Cuando se abrían las puertas de la máquina, se podía ver un complejo mecanismo de engranajes que, supuestamente, permitían jugar un partido de ajedrez de buen nivel. En realidad, dentro de la estructura se escondía un pequeño jugador humano.

sobre los humanos y como complemento para las personas con capacidades disminuidas.

- **Domésticos:** el sueño de todo amo o ama de casa, un robot que realice alguna o todas las tareas del hogar. Ya hay entre nosotros aspiradoras, lavarropas, heladeras, etcétera, que modifican su comportamiento en forma autónoma según el ambiente en el que trabajan.
- **Sociales:** robots utilizados en ámbitos sociales (como películas, eventos y supermercados) con funciones de comunicación intensiva con los humanos. En estos casos, uno de los elementos de investigación fundamental es el aspecto estético del robot, el estudio de la interfaz con el humano para realizar una comunicación completa, con gestos, tonos, silencios, etcétera.
- **Agrícolas:** así como en sus comienzos la robótica tuvo amplia aplicación en la industria, en los últimos años ha comenzado a crecer en forma exponencial el uso de robots y de la inteligencia artificial en el sector agrícola-ganadero. Las cosechadoras autónomas, las sembradoras controladas por mapas satelitales, los fumigadores robotizados y otros dispositivos hicieron su aparición dentro de lo que actualmente se conoce como **agricultura de precisión** (**Figura 6**).

Figura 5. *Kit de brazo robótico* ***RA-01*** *con 5 servos.*

III ROBOTS INSECTOS

Como si no fueran suficientes los insectos que nos perturban durante todo el año, los ingenieros han decidido que, antes de imitar a un humano, es necesario lograr un insecto robótico. Los sistemas de visión y de vuelo de los insectos son dos fuentes de inspiración muy importantes, dado que con mecanismos sumamente sencillos, logran captar el mundo que los rodea y volar sobre él de una manera altamente adaptativa.

Según el medio en el que desarrolla la actividad

- **Acuáticos:** se caracterizan por movimientos tridimensionales en un ambiente hostil desde el punto de vista mecánico y electrónico (**Figura 8**).
- **Terrestres:** son los más populares y económicos. Podemos, a su vez, subclasificarlos por sistema de locomoción: fijos, ruedas, orugas, patas, arrastre, etcétera.
- **Aéreos:** con movimientos tridimensionales, como el acuático, pero con una exigencia mucho mayor en el control en tiempo real del sistema de levitación (**Figura 7**).
- **Híbridos**: combinación de algunos de los anteriores.

***Figura 6. Demeter**, un robot de aplicación agrícola desarrollado en la Universidad de Carnegie Mellon.*

En esta clasificación, las características mecánicas del robot se modifican en forma sustancial entre uno u otro medio. Prácticamente, es impo-

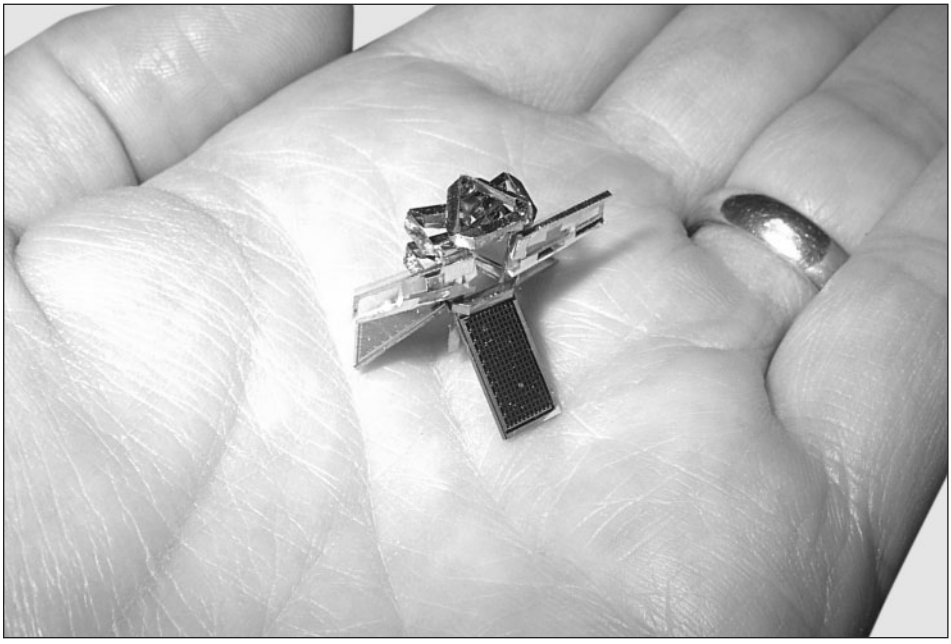


***Figura 7.** Robot **insecto volador** desarrollado en la Universidad de Berkeley.*

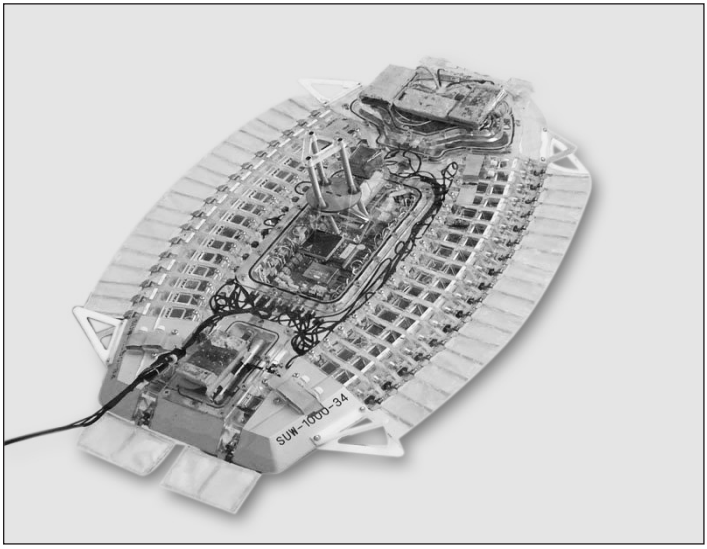


***Figura 8. Robot acuático** japonés, que imita la estructura de algunos seres acuáticos.*

sible utilizar la mecánica de un robot construido en un medio para que funcione en otro, salvo en el caso de algunos híbridos.

Según la ubicación de la inteligencia del robot

- **Autónomos:** la inteligencia está ubicada en el mismo robot. Puede comunicarse con otros o con un sistema central, pero los aspectos esenciales de funcionamiento se resuelven en forma independiente en el propio robot.
- **Control automatizado** (semiautónomos): la mayor parte de la inteligencia del robot está ubicada en un sistema central. Los sensores pueden ser locales, es decir que le envían la información obtenida a ese sistema central, o globales. El sistema central les comunica a los robots las acciones que deben realizar. Un ejemplo de este modelo es la categoría **Mirosot** de fútbol de robots de la FIRA.
- **Híbridos:** son robots autónomos que, en ciertos momentos del proceso, pueden ser controlados por humanos o por un sistema central. Un ejemplo son los robots que se utilizan en misiones espaciales, que operan en forma autónoma pero que, ante un percance, pueden ser dirigidos desde nuestro planeta.

También podríamos clasificar a los robots por sus características estructurales, por el tipo de sensado del mundo, etcétera. De todas maneras, todos los robots comparten la misma arquitectura básica, desde el más pequeño hasta Terminator. A continuación veremos la fuerte analogía

que encontramos entre un robot y una computadora convencional.

Unidades de un robot

En la arquitectura de cualquier computadora, podemos encontrar las siguientes unidades que la componen:

- **Unidades de procesamiento:** es el conjunto de dispositivos que se encargan de realizar la transformación de los datos de entrada para obtener los datos de salida.
- **Unidades de entrada:** son las unidades que permiten realizar el ingreso de información para su posterior procesamiento.
- **Unidades de salida:** son las unidades que se ocupan de comunicarle los resultados del procesamiento al usuario u operador.

En un robot podemos encontrar la misma arquitectura. A las unidades de entrada de un robot las llamamos **sensores**, que pueden ser externos, como un sensor de tacto, o internos, como un **encoder** que permite determinar la distancia recorrida por una rueda. A las unidades de salida se las conoce como **actuadores**. Aquí podemos mencionar **leds** de señalización, **buzzers**, **motores**, **displays**, etcétera. En síntesis, el robot **recibe información** del ambiente mediante sus sensores, **procesa la información** con su unidad de procesamiento y **realiza sus acciones** al mover motores y encender luces y buzzers. Tomemos como ejemplo a uno de los robots más conocidos: Terminator. En este caso, sus sensores son las cámaras que le permiten mirar, su sistema auditivo y los sensores de tacto que tiene su piel. No recordamos que tenga olfato o que alguna vez haya comentado lo sabroso de alguna comida. Sus actuadores esenciales son los motores o los músculos de alambre que conforman su cuerpo.

Uno de los problemas más apasionantes de la robótica es el equilibrio que es necesario obtener entre las tres unidades, para lograr que el robot cumpla con su objetivo. Por ejemplo, los sensores más sofisticados o que

GENERACIONES DE ROBOTS

Así como hablamos de generaciones de computadoras, también se ha definido el grado de evolución de los robots industriales como vemos a continuación:

- Primera generación: repiten una tarea sin considerar las modificaciones que ocurren en su entorno.
- Segunda generación: toman información limitada de su entorno y modifican su comportamiento.
- Tercera generación: son programados en lenguaje natural y organizan sus tareas en forma automática en un nivel más alto que los de Segunda generación.

entregan mayor cantidad de datos, como puede ser una cámara de video, exigen de parte del procesador un mayor tiempo de trabajo para poder obtener un conjunto de información que resulte significativo. De la misma manera, el control de los actuadores sofisticados, como cierto tipo de motores, consume tiempo de procesamiento que es absolutamente necesario para que el robot opere en tiempo real. En síntesis, es imprescindible lograr el **equilibrio entre velocidad y precisión**, en especial en aquellos robots que operan en entornos muy dinámicos. Es por eso que, habitualmente, se utilizan ciertos procesadores específicos para el filtrado de la información de entrada y para el control de los actuadores, y así se libera de esta tarea al procesador central y se complementa su función.

En las próximas páginas, analizaremos cada una de estas unidades en profundidad, de forma tal que podamos realizar la elección adecuada según los objetivos específicos que tengamos para nuestro robot.

Procesamiento

Cuando comenzamos a analizar lo que nos ofrece el mercado de la robótica con respecto a procesamiento, probablemente nos encontremos confundidos ante la diversidad de posibilidades. Podemos encontrar desde micros de muy bajo precio, en los que debemos construir en forma artesanal toda la electrónica que los complementa para poder procesar las entradas y salidas, hasta costosos kits que tienen absolutamente todo resuelto. Está claro que con éstos últimos podremos hacer que nuestro robot funcione en algunas horas, pero es en el primer caso donde tendremos un control absoluto y de bajo nivel de las capacidades de procesamiento de nuestro dispositivo.

De todas maneras, analizaremos en forma detallada las ventajas y las desventajas de ambas propuestas: el desarrollo con el uso de kits frente a la construcción en forma artesanal de los robots. Además, conoceremos brevemente algunos kits y micros disponibles en el mercado, y haremos una lista de los sitios web donde se puede conseguir información más detallada y completa.

Kits

Los kits para la construcción de robots, en general, presentan los siguientes **elementos**:

- **Un procesador o conjunto de procesadores** con toda la electrónica de entrada y salida de los sensores resuelta. Además, poseen un sistema operativo dentro del controlador (**firmware**), que eleva el nivel de programación de los procesadores, lo que posibilita el uso de lenguajes de alto nivel o interfaces gráficas para el

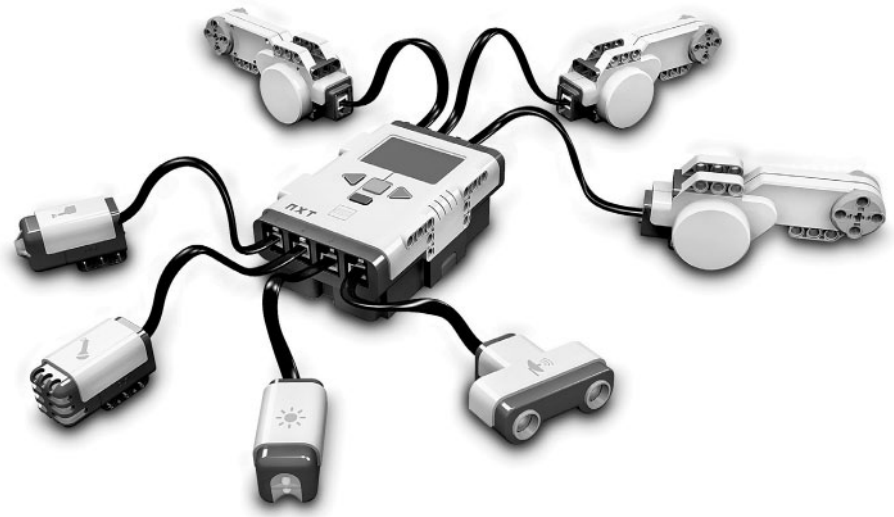


Figura 9. *Lego* ***Nxt*** *con el conjunto de motores y sensores que vienen con el kit.*

desarrollo de la inteligencia de nuestro robot en forma muy sencilla.

- **Un conjunto de sensores** que aprovechan la electrónica ya resuelta, y que con una simple conexión funcionan de manera casi mágica. Por ejemplo, sensores de luz donde el firmware interpreta el voltaje que entrega el sensor como un valor entre 0 y 100.
- **Un conjunto de motores** que también utilizan la electrónica de salida, que se alimentan directamente de la misma fuente que alimenta al procesador, y que gracias al firmware podemos indicarle dirección, velocidad, etcétera, sin la necesidad de cálculos complejos.
- **Material constructivo** para resolver la mecánica del robot, altamente reutilizable y que en poco tiempo permite la elaboración de la física del robot mediante la aplicación de conocimientos de nuestra infancia.

Si tenemos en cuenta este conjunto de materiales, es sencillo notar que las **ventajas** que nos presenta el uso de kits para la construcción de nuestro robot son las siguientes:

- **Menor tiempo de construcción del robot**: en pocas horas, podemos obtener robots poderosos para los desafíos habituales en robótica.
- **Alta reusabilidad del material**: una vez terminado el desafío, podemos desarmar el robot y utilizar todas las piezas, los sensores, los motores y el procesador para armar un robot completamente distinto.
- **Baja necesidad de conocimientos**

***Figura 10.** Un **humanoide** realizado con el Lego Nxt.*

técnicos: sin saber electrónica y prácticamente sin saber programación, podemos desarrollar un robot poderoso. Desde ya que para aquellos usuarios que sí tengan esos conocimientos, el aprovechamiento será mucho mayor. Más aún si los desarrolladores del kit tuvieron la precaución de dejar abiertos tanto el firmware como el hardware del procesador y los sensores.

De todas formas, no todas son cosas positivas. Las **desventajas** que tenemos con el uso de kits son:

- **El alto o altísimo costo de un kit**: la relación puede ser de 20 a 1 con respecto a un desarrollo manual. El robot que construiremos en este libro mantiene esta relación con los kits más económicos de robótica.
- **La imposiblidad**, en muchos casos, de poder realizar **modificaciones de bajo nivel en el hardware o el firmware del robot**. A pesar de todos los esfuerzos de documentación que haga la empresa creadora del kit, es imposible que todo sea altamente modificable, por la misma necesidad de mantener la arquitectura intrínseca del robot. Su elaboración artesanal desde cero nos permite modificar **hasta el más mínimo detalle**.
- **Baja precisión y calidad final de los robots**: dado que los kits son para el desarrollo de robots de diversos propósitos, en todos los casos perdemos precisión y calidad. Por ejemplo, los motores sirven para moverse en un determinado ambiente con alto margen de error, pero no son veloces ni permiten movimientos de precisión como lo pueden exigir ciertos objetivos. En general, los sensores son económicos, y el rango de valores que devuelven es pobre.

Los **kits de robótica más conocidos** en el mercado son los siguientes:

- **Lego Nxt**: sucesor del modelo **Mindstorms**, es el kit de mayor difusión en todo el mundo (**Figura 9**).

Su procesador es un **ARM7 de 32 bits**. Cuenta con 256 Kb de Flash y 64 Kb de RAM. Se comunica por Bluetooth clase 2 y por USB. Tiene 4 entradas para sensores y 3 salidas para actuadores. En el kit se integran 3 motores servo con encoders incorporados para controlar su movimiento con precisión.

Además, cuenta con un sensor de tacto, uno de sonido, otro de luz y por último un sensor ultrasónico. Para la mecánica del robot cuenta con piezas de las denominadas **Technic**, que permiten diseñar diferentes estructuras según el destino del robot creado (**Figura 10**). El lenguaje de programación es un ambiente gráfico muy sencillo, similar al **Robolab** de las versiones anteriores, pero con mayor potencia y versatilidad. Para finalizar, una de las mejores decisiones que ha tomado la firma Lego es publicar muchísima información sobre el desarrollo tanto de hardware como de software del kit, lo que ha permitido que en poco tiempo (salió a la venta en agosto de 2006) las personas de todo el mundo que tienen este hobby hayan desarrollado hardware y software específico fuera del oficial. Para obtener más información, podemos visitar **http://mindstorms.lego.com**.

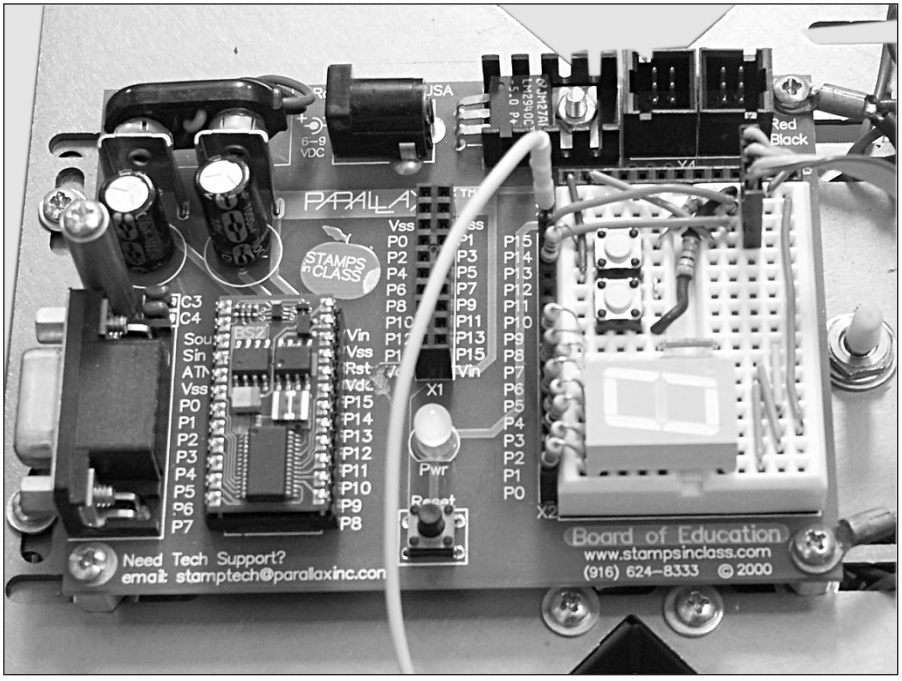


Figura 11. *Placa educativa del* ***Basic Stamp*** *de Parallax.*

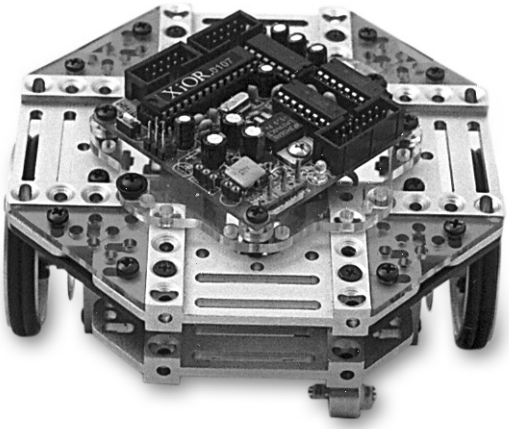


Figura 12. *Aquí podemos ver uno de los modelos constructivos de* ***XiOR****, conocido como* ***N10****.*

- **Parallax**: el producto más popular de esta empresa es el micro **Basic Stamp** (**Figura 11**). Aunque uno puede adquirir solamente el micro y realizar el robot desde cero como comentaremos más adelante, hay tantos desarrollos y tantas presentaciones de productos de la firma Parallax que hemos decidido presentarlo dentro de esta sección. El **Basic Stamp Starter Kit** está desarrollado para iniciarse en el mundo de la robótica. Incluye un Basic Stamp II, que tiene una velocidad de procesamiento de 20 Mhz, con 2 Kb de **EEPROM** (*electrically-erasable programmable read-only memory*, ROM programable y borrable eléctricamente) y 16 E/S (entradas y salidas) más 2 dedicadas. Este micro está instalado en una placa educativa con una pequeña **protoboard** (paqueta de experimentación), donde podremos realizar todas nuestras experiencias. Viene con un servo, leds, capacitores, resistencias y otros componentes para diseñar nuestro robot. Las guías que acompañan a este y otros kits son excelentes (¡y algunas están en castellano!), y pueden conseguirse en forma gratuita en el sitio de la empresa. La última creación de Parallax ha sido el micro de nombre **Propeller**, con 8 procesadores paralelos en su interior. Tiene una arquitectura de 32 bits, 80 Mhz, con 32 pines de E/S direccionables por cualquiera de los 8 procesadores. Para buscar más información, podemos visitar **www.parallax.com**.
- **XiOR**: en América Latina, contamos con nuestro propio kit de robótica. XiOR (**www.xior.org**) es una empresa argentina de tecnología y entre sus trabajos ha desarrollado un sistema constructivo para la fabricación de robots móviles autónomos. El modelo **N10** es el primer robot desarrollado con él (**Figura 12**). Una de sus principales características es la posibilidad de que el usuario reconfigure toda su morfología física para adaptarlo a diferentes entornos y experimentos. Incluso es posible combinarlo con otros robots similares para formar parte de estructuras mayores.
  Normalmente, está equipado con 2 o 4 celdas de Li-Ion de 900 mAh,

agrupadas en packs de a dos. En cuanto a las capacidades de procesamiento, el controlador XiOR.0107 tiene un procesador **AVR ATMega8** (Atmel, 2003) de 16 MIPS aproximadamente, y 8 KB de RAM de programa, e incorpora comunicaciones RS-232, 2WI y drivers de potencia para agregar dos motores adicionales de corriente continua o un motor paso a paso.

El sistema constructivo **Múltiplo** ofrece piezas de aluminio y plástico cuidadosamente diseñadas para desarrollar robots de tamaño reducido, pero con una precisión y robustez sorprendentes. Podemos encontrar más información en **www.xior.org**.

Además de estos kits que mencionamos, el mercado de la robótica educativa crece día a día, y en la Web podemos encontrar muchos otros que tal vez se ajusten mejor a nuestras necesidades. Aunque aquí describimos los más conocidos, podemos navegar para buscar más información sobre el tema.

***Figura 13. Sphinx**, otro modelo de XiOR para el control de pozos de petróleo.*

Robótica sin kits

Como comentamos antes, realizar un robot sin la ayuda de un kit nos proporciona mayor versatilidad, robustez, potencia, precisión, velocidad y adaptabilidad. El problema esencial es que nos exige mayor conocimiento y trabajo. Cuando comencemos un desarrollo de este tipo, lo primero que debemos analizar es el procesador que vamos a utilizar, según la funcionalidad y el costo que deseamos que tenga nuestro robot.

▶ OTROS KITS LATINOAMERICANOS

Además de los kits mencionados, a continuación presentamos links donde se pueden encontrar otros kits desarrollados en Latinoamérica:

- **NeoRobotic**: kits de robótica autónoma (**www.neorobotic.com**).
- **Arbot**: robots controlados desde la PC por puerto paralelo (**www.dutten.com.ar**).
- **Blocky-tronic**: sistema constructivo con microcontrolador, sensores y motores (**www.blockymania.com.ar/blockytronic/**).

Además, debemos tener en cuenta que es necesario complementar el procesador con un circuito de control adecuado para realizar las conexiones de entrada, salida, alimentación y control de dispositivos.

***Figura 14.** Otro de los micros de la firma **Microchip**, creadora del **16F84**.*

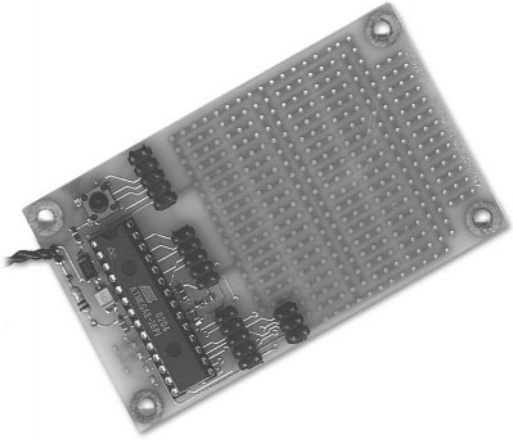


***Figura 15. ATMega8** en una placa de prototipado rápido.*

Microcontroladores

El procesamiento más habitual en los robots pequeños y de bajo costo es el provisto por microcontroladores. Los micros más utilizados son los de la familia **PIC**, de la firma **Microchip**. De todos los modelos que se ofrecen, el más popular para la construcción de robots es el **PIC16F84**. Posee una memoria de programa Flash de 1 KB con palabras de 14 bits, una memoria RAM de datos de 68 bytes y una EEPROM de 64 bytes, y 13 pines de E/S.

Por otra parte, su precio es más que accesible. Dado que desarrollaremos nuestro robot con PIC, dejaremos para más adelante una descripción más detallada de éstos.

La empresa **Atmel** fabrica otro tipo de microcontroladores, que también son de uso habitual en la construcción de robots. De todas sus líneas, la denominada **AVR de 8 bits** es la más recomendada para el procesamiento en esta disciplina. Toda la línea AVR presenta características como bajo poder de consumo, arquitectura RISC y Harvard, 32 registros de 8 bits de propósito general y facilidad de implementación de lenguajes de alto nivel para la programación.

En particular, el micro ATMega8 (**Figura 15**) es ideal, dado que presenta una memoria de 8 KB de programa, 1 KB de SRAM y 512 bytes de EEPROM, seis canales de PWM, USART progra-

mable, conversor analógico digital de cuatro canales multiplexados de 10 u 8 bits y dos canales de 8 bits, entre otras características que se pueden encontrar en la página de la firma.

PDA

Si no tenemos una base de conocimiento y experiencia en electrónica para utilizar los microcontroladores que presentamos pero contamos con una **Palm** o una **PocketPC**, podemos destinar parte de su tiempo de uso para que actúe como cerebro de nuestra creación. En los últimos años, estos dispositivos han bajado de precio en forma notable y, además, algunos modelos han caído en desuso, aunque pueden adaptarse perfectamente para ser empleados con nuestros robots. Uno de los ejemplos más interesantes en este punto es la adaptación del robot **Robosapien** (**Figura 17**) de la firma **Wow Wee** para que pueda ser controlado desde una PocketPC por infrarrojo, con lo cual utilizamos la PDA como control remoto inteligente que puede generar su propio procesamiento. Para la detección del

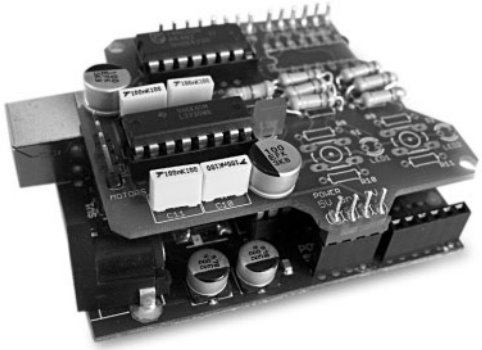


***Figura 16. Arduino**, una plataforma abierta en software y hardware para el desarrollo de robots.*

mundo se utiliza una cámara que se conecta en el puerto de tarjetas de memoria. En síntesis, con poco dinero podemos tener un robot humanoide. Se puede obtener más información en **www.informatik.uni-freiburg.de/~nimbro/media.html**.

El Instituto de Robótica de **Carnegie Mellon** desarrolló un proyecto para la construcción de un robot autónomo móvil de bajo costo, y se utilizó una Palm como procesador (**Figura 18**). En el sitio **www.cs.cmu.edu/~reshko/PILOT** podemos encontrar todos los pasos y los materiales nece-

ARDUINO, UNA PLATAFORMA OPEN HARDWARE PARA ROBÓTICA

Arduino es una placa basada en el micro Atmel ATmega8, programable por serial o USB, con entradas y salidas analógicas y digitales. Su diseño y distribución son completamente libres. En su sitio web, **www.arduino.cc**, podemos encontrar los planos para armar la placa y el software necesarios para su programación. Si no nos animamos a construirlos desde cero, podemos adquirir los componentes en el mismo sitio.

Figura 17. *Robot* ***Robosapien*** *hackeado con una* ***PDA*** *en su cabeza como procesador.*

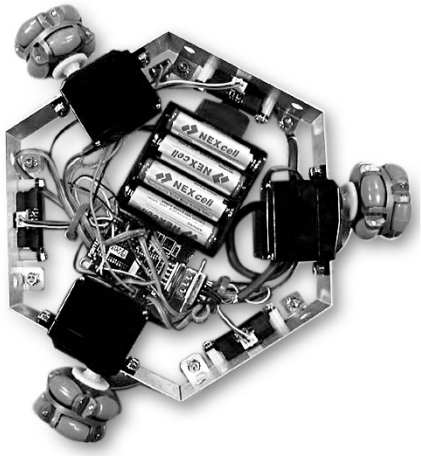


Figura 18. Palm Pilot Robot Kit *es un kit de robótica cuyo procesador es una Palm Pilot.*

sarios para construirlo, incluido el software que es indispensable bajar en la Palm para la programación. Si bien utiliza ruedas omnidireccionales, servos y otros materiales que pueden ser costosos en el mercado local, con ellos obtendremos un robot móvil de excelente calidad. Comprar una PDA pura y exclusivamente para la construcción de un robot es demasiado costoso en comparación con las otras variantes. Pero si disponemos de una, o si los precios de estos dispositivos continúan bajando, puede ser una posibilidad muy interesante por su potencia de procesamiento y tamaño.

Computadoras (PC)

Por último, no queremos dejar de presentar la posibilidad de usar motherboards de PC convencionales pero de tamaños reducidos, conocidos como **Mini y Nano-ITX** (**Figura 19**). Por ejemplo, la conocida firma **VIA** ha desarrollado la línea **EPIA**, de bajo consumo y con un tamaño que oscila entre 12 y 17 centímetros de lado. Uno de sus modelos, el VIA EPIA NL, posee placa de video de S3, zócalo para memoria DDR 266/333/400, un slot mini PCI, un puerto S-ATA y dos P-ATA, placa de sonido, LAN, puerto serie, USB y otros, que permiten tener toda la potencialidad de una PC ¡en 12 x 12 centímetros! En el sitio **www.mini-itx.com** se muestran desarrollos

de fanáticos de estos modelos, no sólo en robótica, sino también en los dispositivos más extraños. En **www.epiacenter.com** se analizan muchos mothers de este tipo, de diferentes marcas, y se ofrecen los links a tiendas en línea donde adquirirlos. La enorme ventaja de utilizar este tipo de tecnología es la potencia que nos brinda una PC para el procesamiento de la información de los sensores y la toma de decisiones, la posibilidad de programar cómodamente en lenguajes de alto nivel, y la facilidad de desarrollar y de testear en nuestra computadora de escritorio. Por otra parte, tanto los sensores como los actuadores que queramos conectar pueden desarrollarse sobre plataformas muy conocidas, como serial, paralelo o USB. Cualquier cámara web puede conectarse en segundos, y todos los drivers de los dispositivos ya desarrollados funcionan sin problema en nuestro robot.

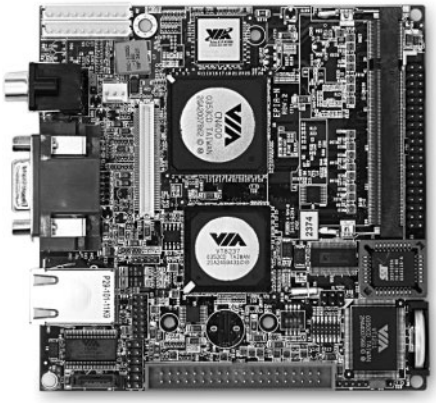


***Figura 19.** Nano-itx, un motherboard de la empresa Via, de 12 cm por 12 cm.*

Sensores

Una de las características fundamentales que debe tener un robot es poseer algún mecanismo de modificación de su comportamiento según el ambiente en el que se encuentra. Para esto, tiene que contar con dispositivos que le permitan **sentir** el mundo que lo rodea, según la tarea que deba realizar. Por ejemplo, en ambientes muy dinámicos, es probable que deseemos sensores que puedan captar rápidamente la información, aunque perdamos precisión. Si esto no fuera así, la información recibida no sería útil, porque representaría un

ROBOTS CON NOTEBOOKS

Gracias a la baja de su precio y a su tamaño, no es sorprendente encontrar robots cuyo procesamiento lo realiza una notebook. Permite utilizar lenguajes de alto nivel, muy buena velocidad de procesamiento y entradas y salidas estándares como serial, paralelo, USB o firewire. Además, es posible desarrollar todo el comportamiento del robot desde una PC convencional, y cualquier desperfecto es fácil de salvar con la instalación una nueva notebook.

estado antiguo que, con seguridad, ha sido modificado por el alto dinamismo del ambiente. Un ejemplo de esto es el fútbol de robots.

En otros casos, necesitamos la mayor precisión posible por la operación que realizamos y para lograrlo, tendremos que utilizar sensores de mayor fiabilidad, aunque su tiempo de respuesta sea bajo. En realidad, cuando construimos un robot, siempre luchamos con esta dualidad (y, seguramente, con muchas variables más). Es por eso que la elección de los sensores que utilicemos estará determinada por la tarea que vayamos a realizar y sus requerimientos particulares.

Los **sensores** pueden definirse como dispositivos que nos permiten **medir** alguna característica del ambiente, como la temperatura, la distancia, la posición, el sonido, la intensidad de la luz, los colores, la altitud, la velocidad, la rotación, etcétera. Lamentablemente, en la realidad no existe el sensor perfecto, y por lo tanto debemos completar y corregir la información con la utilización de algoritmos de corrección y redundancia de sensores.

Otro aspecto importante para tener en cuenta es que, según el tipo de controlador que utilicemos para nuestro robot, deberemos diseñar **circuitos intermedios** entre el sensor y el controlador, con el fin de convertir la señal del sensor en un

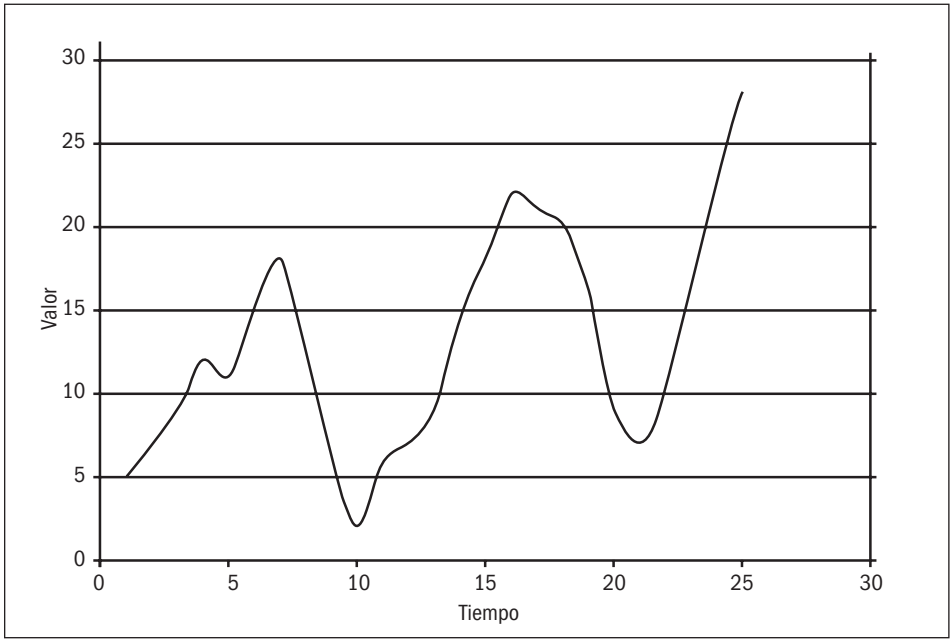


Figura 20. *En este caso, los valores son los de un sensor analógico.*

valor interpretable por nuestro procesador. Por lo tanto, aunque podamos utilizar un mismo sensor para diferentes plataformas de controladores, con seguridad tendremos que diseñar estas interfaces en forma dedicada para cada procesador.

Clasificación

Podemos dividir a los sensores en dos grandes grupos: **analógicos** y **digitales**. Los analógicos entregan un valor (por ejemplo, una tensión) dentro de un determinado rango continuo (**Figura 20**). Un ejemplo de este tipo es una **fotorresistencia**, que mide la intensidad de la luz, y que podemos adaptar para que entregue un valor de 0 a 5 Voltios. Los sensores digitales entregan una señal discreta dentro de un conjunto posible de valores (**Figura 21**). Es decir, este conjunto de valores se modifica de un rango a otro por saltos discretos y no continuos.

Un ejemplo clásico es el sensor de toque, que devuelve valor 1 (ó 0 Voltios) si está apretado y 0 (ó 5 Voltios) si no es así.

Desde ya que, a pesar de que el sensor sea analógico, finalmente, en la lectura del controlador, obtendremos un valor discreto. Es decir, se realizará una conversión analógico/digital dentro del controlador o en la interfaz que construyamos para la adaptación del sensor al controlador.

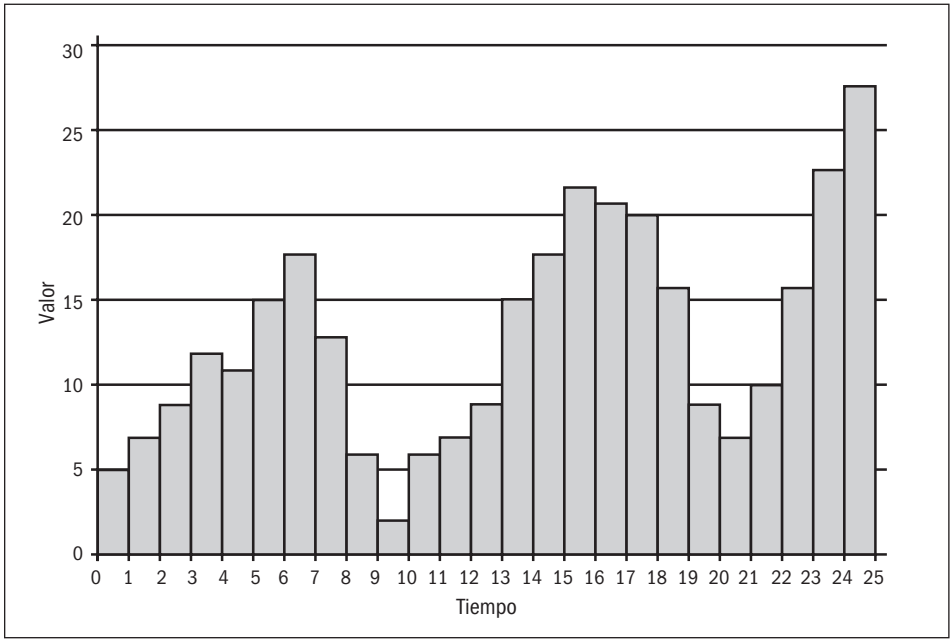


Figura 21. *Aquí podemos ver un ejemplo de valores tomados por un sensor digital.*

***Figura 22.** Aquí podemos apreciar un sensor de **efecto hall**.*

Otra clasificación posible de los sensores es **internos** y **externos**. Los internos son aquellos que nos brindan información del propio robot, como la velocidad, la rotación, la posición, la altura, etcétera; en tanto que los externos son los que proporcionan datos del ambiente, como las distancias, la temperatura, la presión, etcétera.
Por último, también podemos dividir los sensores en **pasivos** y **activos**. Los activos son los que necesitan enviar una señal hacia el ambiente para luego recibir el **rebote** de esa señal y, de esta manera, interpretar el estado del mundo que lo rodea. Un ejemplo clásico de este tipo de sensor es el ultrasónico, que envía una señal sonora que, al rebotar con un objeto, vuelve al robot, y al calcular el tiempo de demora, puede interpretar la distancia al objeto. Desde ya que los sensores activos necesitan mucha más electrónica para su funcionamiento, pero la información que nos brindan es mucho más rica que la que nos ofrecen los sensores pasivos.

Tipos de sensores

Hacer una lista de todos los tipos de sensores existentes sería imposible. Los que nombraremos a continuación son los más utilizados en robótica de bajo costo:

- **Sensores de interrupción**: simplemente, detectan si pasa corriente o no. Se utilizan como sensores de choque o contadores de eventos (vueltas de una rueda, por ejemplo).
- **Sensores de posición**: determinan la posición del robot. Un sensor de este tipo puede ser un potenciómetro que cambia su valor por la rotación de las ruedas; u ópticos, como los que

▶ SIMULADORES

Existen cientos de simuladores de distintos tipos de robots: humanoides, autónomos, robots para fútbol, etcétera. Por ejemplo, uno de los más interesantes en el fútbol de robots lo podemos conseguir en el sitio **www.fira.net/soccer/simurosot/R_Soccer_ v15a_030204.exe**. Sin embargo, un simulador no es lo mismo que la robótica física. Creer que la experiencia física es igual a la simulada es pensar que **Second Life** es igual a la vida misma. Y no es así, ¿no?

se usan en un mouse tradicional, que cuentan la cantidad de veces que recibe señal el sensor óptico de una ruedita que tiene ventanitas cada determinado ángulo (podemos abrir uno para verlo, el mecanismo es sencillo).

- **Sensores de efecto hall**: estos sensores utilizan una propiedad de los imanes, que modifican su conductividad cuando encuentran un objeto metálico (**Figura 22**).
- **Sensores de luz o brillo**: detectan la cantidad de luz que reciben. Según el tipo de sensor utilizado, por este mecanismo podemos detectar diferencias entre los colores, si éstos no son brillantes.
- **Sensores infrarrojos**: envían una señal infrarroja y determinan el tiempo que tardan en volver a recibirla. Permiten detectar obstáculos (si la señal vuelve) o medir distancias si el sensor es preciso.
- **Sensores de ultrasonido**: como explicamos antes, envían una señal sonora y captan el rebote, de la misma manera en que lo hace un Sonar en el mar.

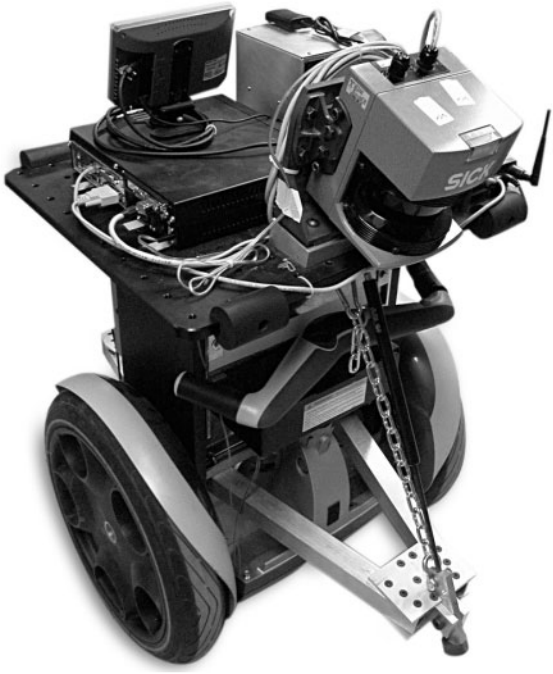


***Figura 23. Laser Sick**, usado como sensor por su magnífica precisión.*

En los últimos tiempos, gracias a la posibilidad de contar con mucha capacidad de procesamiento en tamaño reducido, se ha comenzado a utilizar el **video** como sensor de los robots.

{} ROBOTS ADAPTATIVOS

Una de las dificultades más complejas en el desarrollo de robots es la de darles la capacidad de adaptarse en tiempo real al mundo que los rodea. Por ejemplo, los sistemas de visión son muy dependientes de la luz que existe en el ambiente. Por esa razón, es habitual que en las competencias de robots, los equipos que han tenido un gran desempeño en días anteriores, ante un leve cambio de iluminación no puedan realizar prácticamente ninguna tarea.

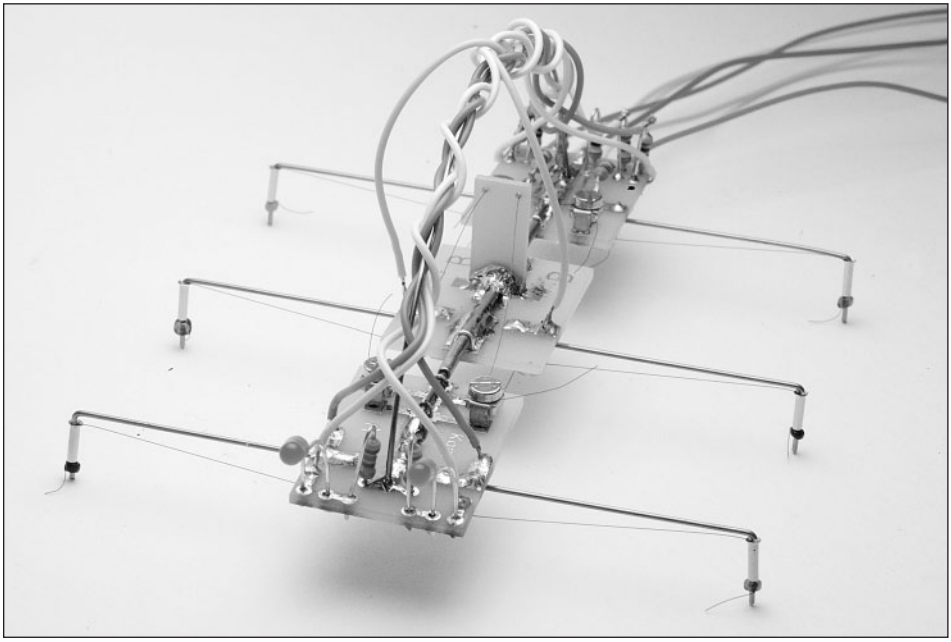


Figura 24. *En esta imagen podemos ver un* ***insecto robótico*** *que, en lugar de motores, utiliza músculos de alambre.*

El procesamiento de imágenes es un tema muy complejo y apasionante al mismo tiempo, y escapa a la arquitectura y los costos de fabricación de los robots que nos hemos propuesto para este libro. Pero seguramente, en poco tiempo podremos utilizar nuestras webcam en forma sencilla y económica para nuestras creaciones. Las cámaras son los sensores que más datos nos proveen en un lapso de tiempo muy breve. El problema fundamental es que necesitamos mucho tiempo de procesamiento en algoritmos complejos para extraer información útil de semejante cantidad de

MÚSCULOS DE ALAMBRE

Hasta hoy, el movimiento de los robots se ha realizado con motores. Pero si queremos imitar el movimiento de organismos vivos, ninguno de ellos utiliza este tipo de dispositivos. Es por eso que se ha desarrollado otro tipo de sistema de movimiento conocido como músculos de alambre (*wired muscles*). Los músculos de alambre son, como su nombre lo indica, alambres que cambian su longitud cuando se calientan, lo que se puede lograr al pasar corriente eléctrica por ellos (**Figura 24**).

bits. Es en este tipo de algoritmos donde podremos encontrar con más claridad el dilema a resolver entre **velocidad y precisión**.

Actuadores

Si nuestro robot sólo observara el mundo sin actuar en él, sería un robot sumamente limitado. Nuestro deseo es que modifique su estado y el del ambiente según la información que obtiene en el proceso. Con este fin, disponemos de motores, músculos de alambre, lámparas, displays, buzzers, etcétera. Al conjunto de estos dispositivos se lo denomina **actuadores**.

Los actuadores más sencillos de utilizar son las lámparas, que no ameritan demasiada descripción para su uso. Simplemente, con conectarlas a alguna salida del procesador y proveer la alimentación necesaria para su funcionamiento, podremos encenderlas y apagarlas con nuestro programa. Sin embargo, pondremos nuestro foco de atención en los motores, dado que definen en gran medida nuestra capacidad de desplazamiento, los grados de libertad y otros aspectos vinculados al movimiento del robot. Por definición, el **motor eléctrico** es un dispositivo electromotriz, es decir, que **convierte la energía eléctrica en energía motriz**. Todos los motores disponen de un eje de salida para acoplar un engranaje, una rueda, una polea o cualquier mecanismo capaz de transmitir el movimiento creado por el motor. La etapa de elección de un motor puede ser una tarea muy complicada según las limitaciones de nuestro proyecto, si tenemos en cuenta todas las características que definen al motor. Éstas son: tamaño, peso, velocidad (revoluciones por minuto, RPM), torque (kilogramo por centímetro) tensión y, la más sensible: el costo. En la actualidad, existen diferentes tipos de motores, que describiremos a continuación:

Motores de corriente continua (CC)

Son los motores más comunes y que casi todos conocemos (**Figura 25**). En general, los encontramos en cualquier

LEYES DE LA ROBÓTICA

Dentro de su producción literaria basada en la robótica, **Asimov** definió tres leyes para la protección de los humanos que los robots de ficción tienen almacenadas en su cerebro positrónico. Éstas son:

1. Un robot no puede hacerle daño a un ser humano o, por inacción, permitir que un ser humano sufra daño.
2. Un robot debe obedecer las órdenes de un humano, salvo que alguna de éstas entre en conflicto con la primera ley.
3. Un robot debe proteger su propia existencia, salvo que esto viole la primera o la segunda ley.

***Figura 25.** Motores y motorreductores de corriente continua.*

tipo de juguete (¡un buen lugar donde ir a buscarlos!). El funcionamiento del motor se basa en la acción de campos magnéticos opuestos que hacen girar el **rotor** (eje interno) en dirección opuesta al **estator** (imán externo o bobina). De este modo, si sujetamos la cubierta del motor por medio de soportes o bridas, el rotor con el eje de salida será lo único que girará. Para cambiar la dirección de giro en un motor de corriente continua, tan sólo debemos invertir la polaridad de su alimentación eléctrica.

Un detalle importante es que, casi siempre, se utilizan acompañados de un sistema de engranajes que reducen la velocidad y proporcionan mayor fuerza, dado que este tipo de motores carece de esta cualidad. Es conveniente conseguir el conjunto completo porque las adaptaciones son complicadas y pocas veces se obtienen muy buenos resultados.

Motores paso a paso (PAP)

Un motor paso a paso (**Figura 26**) se diferencia de uno convencional porque puede **ubicar su eje en posiciones fijas o pasos**, con lo cual es capaz de mantener la posición. Esta peculiaridad se debe a la construcción del motor en sí: por un lado, tiene el rotor constituido por un imán permanente, y por el otro, el estator construido por bobinas. Al alimentar alguna de esas bobinas, se atrae el polo magnético del rotor opuesto al polo generado por la bobina, y éste permanece en esta posición hasta que la bobina deje de generar el campo magnético y se acti-

FANÁTICOS DE LEGO

Como comentamos anteriormente, Lego tuvo la precaución de liberar muchísima información sobre sus diseños de manera que otros pudieran desarrollar material para sus kits. Es por eso que podemos encontrar varios lenguajes para poder programar los robots de esta firma. Además, también se pueden conseguir los planos de los sensores de todo tipo. Una de las páginas más conocidas de sensores caseros es **www.extremenxt.com/lego.htm**.

ve otra bobina, lo cual hace avanzar o retroceder al rotor. De esta manera, al variar los campos magnéticos en torno al eje del motor, se logra que gire. Los motores PAP pueden ser de dos tipos:

Bipolar: lleva dos bobinados independientes. Para controlarlo se necesita invertir la polaridad de cada una de las bobinas en la secuencia adecuada.

Unipolar: dispone, normalmente, de 5 ó 6 cables, dependiendo de si el común está unido en forma interna o no. Para controlar este tipo de motores existen tres métodos con sus correspondientes secuencias de encendido de bobinas. El común irá conectado a $+V_{CC}$ o masa según el circuito de control usado y, luego, sólo tendremos que alimentar la bobina correcta para que el motor avance o retroceda según avancemos o retrocedamos en la secuencia. Estos motores son muy utilizados en disqueteras, lectoras de CD e impresoras. En las antiguas disqueteras de 5¼, podemos encontrar algunos bastante poderosos, y en las lectoras de CD, unos más pequeños pero siempre acompañados de buenos mecanismos reductores.

***Figura 26.** Vista interna de un motor paso a paso.*

Servomotores

El servo (**Figura 27**) es un pequeño pero potente dispositivo que dispone en su interior de un pequeño motor con un reductor de velocidad y un multiplicador de fuerza. También cuenta con un pequeño circuito eléctrico encargado de gobernar el sistema. El recorrido del eje de salida es de 180º en

BUZZERS PARA DETECTAR ERRORES

Una de las herramientas fundamentales para detectar errores en nuestro robot es la posibilidad de mostrar en un **display** el valor de los sensores, las variables, etcétera. El problema es que como se trata de un ente autónomo, tendríamos que correr detrás de él para ver qué ocurre y leer los valores que se muestran en el display. Es por eso que usualmente al robot se le agrega un buzzer que nos permite emitir sonidos y enterarnos de lo que sucede.

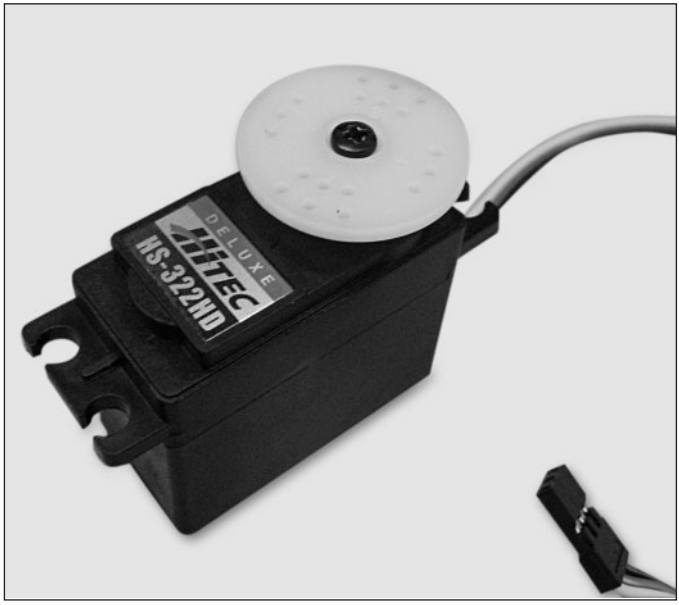


Figura 27. *Servomotor tradicional de la marca* ***Hitec.***

la mayoría de ellos, pero se puede modificar con facilidad para tener un recorrido libre de 360º y, entonces, actuar como un motor común.

El control de posición lo efectúa el servo en forma interna mediante un potenciómetro que va conectado en forma mecánica al eje de salida. Éste controla un **PWM** (*Pulse Width Moduler*, modulador de anchura de pulsos) interno para compararlo con la entrada PWM externa del servo, mediante un sistema diferencial y así, modificar la posición del eje de salida hasta que los valores se igualen y el servo se detenga en la posición indicada. En esta posición, el motor del servo deja de consumir corriente y sólo circula una pequeña cantidad hasta el circuito interno. Si en ese

NANOBOTS

Los nanobots son robots de tamaño microscópico, cuyas dimensiones están en el orden de una millonésima de milímetro. Se realizan con técnicas de nanotecnología y, aunque aún son ensayos y especulaciones teóricas, ya se han dado pasos para su concreción. Esencialmente, se han desarrollado nanobots químicos o moleculares con funciones limitadas. La aplicación para estos futuros robotitos está apuntada a la medicina, la industria, la ecología y la producción de alimentos.

momento forzamos el servo (al mover el eje de salida con la mano), el control diferencial interno lo detecta y manda la corriente necesaria al motor para corregir la posición.
Para controlar un servo, tenemos que aplicar un pulso de duración y una frecuencia específicos. Todos los servos disponen de tres cables: dos para la alimentación y uno para aplicar el tren de pulsos de control que harán que el circuito de control diferencial interno ponga el servo en la posición indicada por la anchura del pulso. Los servomotores son una muy buena alternativa, ya que traen integrado un sistema reductor que nos ahorrará dolores de cabeza a la hora de buscar **fuerza**. La desventaja para algunas aplicaciones es que, en general, **son lentos**. Se utilizan mucho en los automóviles y aviones radiocontrolados, principalmente para accionar el mecanismo que les da la dirección.

... RESUMEN

En este primer capítulo hemos hecho una breve introducción a los conceptos fundamentales de la robótica. Un robot no es más que un dispositivo con un determinado grado de movilidad, que puede realizar un conjunto de tareas en forma independiente y se puede adaptar al mundo en el que opera. Existen diversos tipos de robots, según el uso para el cual se han destinado, el medio en el que se mueven, la capacidad de autonomía que presentan, etcétera.
Cualquiera de estos robots está dirigido por una unidad de procesamiento, que se puede adquirir en forma completa con todas sus comunicaciones de entrada y salida resueltas, o que podemos desarrollar desde cero con micros, dispositivos portátiles o, simplemente, motherboards de PC.
Además, necesita de sensores para poder captar el mundo. Cuanto más complejo sea un sensor y más datos nos provea, tendremos que renunciar a ciertos aspectos de velocidad para obtener información más precisa. Por último, para que el robot pueda trasladarse y actuar sobre el entorno, necesitamos de los actuadores. Los esenciales y más usados son los motores, de los cuales tenemos diversos tipos según el objetivo de nuestro robot y la cantidad de dinero que poseamos.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Cuál es nuestra definición de robot?

2 ¿Qué clasificaciones encontramos de los diferentes tipos de robots?

3 ¿Cuáles son las unidades que presenta un robot?

4 Compare las ventajas y desventajas del uso de un kit de robótica o el desarrollo desde cero del procesamiento del robot.

5 Si desarrollamos la unidad de procesamiento por nuestros propios medios, ¿qué dispositivos podemos utilizar?

6 ¿Qué es un sensor? ¿Cuál es la diferencia entre sensores activos y pasivos?

7 ¿Cuáles son los tipos de sensores con los que podemos contar en la elaboración de nuestro robot?

8 ¿Qué ventajas y desventajas presenta el uso de video como entrada de datos de un robot?

9 ¿A qué llamamos actuadores?

10 ¿Qué tipos de motores encontramos en el mercado?

Componentes del robot

Antes de comenzar a construir nuestro robot, daremos una mirada general a todos los componentes que lo conformarán. Desde ya que elegiremos puntualmente alguno de ellos, pero con este capítulo nos será sencillo seleccionar otras variantes según el objetivo que tengamos para nuestro robot.

UNA MIRADA GLOBAL A NUESTRO FUTURO ROBOT

Como hemos podido ver en el primer capítulo, la cantidad de variantes que tenemos para construir nuestro robot es inmensa. Es por eso que en este capítulo, presentaremos los materiales que hemos decidido utilizar para nuestra primera creación, y justificaremos la elección que hemos realizado. En primer lugar, vamos a dar un paseo por el mundo de los microprocesadores, introduciremos un conjunto de conceptos que nos serán imprescindibles a la hora de construir el controlador de nuestro robot. Luego veremos el tipo de motores que vamos a utilizar y dónde podemos conseguirlos. Para esto podremos desarmar algún equipo electrónico antiguo, como disqueteras, impresoras, etcétera. Luego, haremos lo mismo con los sensores, siempre con el objetivo de gastar la menor cantidad de dinero posible en nuestro primer proyecto (¡luego vendrán proyectos multimillonarios!). Y por último, echaremos una mirada sobre el material que podemos utilizar para la construcción del cuerpo de nuestro robot.

Figura 1. *Ejemplo de un robot artesanal con diversos sensores.*

Objetivos de nuestro robot

Para la elección de todos los componentes de nuestro robot, es necesario que fijemos claramente cuáles serán los **objetivos**, es decir, las tareas que queremos que nuestro robot realice. La primera decisión fundamental es definir si vamos a utilizar un kit o construiremos nuestro robot desde cero en forma artesanal. Es una decisión fundamental porque implica un presupuesto mucho más alto en el primer caso y, de alguna manera, también acota las posibilidades de lo que vamos a diseñar. Por otra parte, en el primer capítulo ya enumeramos las ventajas que brinda el uso de un kit, pero en nuestro proyecto vamos a desarrollar el robot desde cero, paso a paso, componente por componente. En nuestro caso, la primera razón por la que lo haremos de esta manera es porque así vamos a aprender mucho más sobre todas las capas que componen al robot. La segunda razón, no menos importante, es el costo de nuestro proyecto. La idea es que podamos reutilizar elementos que encontremos en desuso. Un humilde aporte ecológico de nuestra parte.

Bien, ahora que hemos decidido armar en forma artesanal nuestro primer robot, tenemos que dejar en claro qué es lo que queremos que haga. En este libro, nuestro objetivo es realizar un robot que pueda, en diferentes adaptaciones, detectar objetos y esquivarlos, seguir una línea negra, y buscar una luz.

***Figura 2.** Partido de fútbol de robots donde se utiliza una pelota infrarroja.*

Cuando decimos diferentes adaptaciones, lo que queremos acentuar es que no necesariamente hará las tres cosas el mismo robot. Vamos a organizar diferentes proyectos, con mínimas modificaciones entre uno y otro, para lograr los tres objetivos. Luego, si combinamos la capacidad de **tocar** objetos con la detección de la línea negra, podremos construir un luchador de sumo. Y para realizar un sencillo jugador de fútbol, uniremos la detección de luz con el tacto de los objetos, dado que nos prepararemos para una categoría que

***Figura 3.** Uno de los micros de la firma Atmel.*

usa una pelota infrarroja (**Figura 2**). Pero ya llegaremos a este punto dentro de unas cuantas páginas. Comencemos con el procesamiento.

Tipo de procesamiento seleccionado

Si seguimos la línea que acabamos de presentar, decidimos realizar nuestro controlador basándonos en un microcontrolador que fuera económico y de uso popular. Pero, para empezar, veamos qué es esto de los **microcontroladores**. A esta altura del siglo XXI, no nos sorprende encontrar en cualquier casa u oficina una computadora, con sus unidades de entrada, de salida y de procesamiento. Y estas computadoras, para nuestra alegría, se reducen cada vez más en precio y tamaño. ¿Pero, cuál es el límite de esta reducción? Aquellos que usaron un dispositivo portátil, como una PDA, habrán notado que la pantalla y el teclado son, en estos momentos, los puntos críticos de reducción. Ahora bien, si no necesitáramos información por una pantalla y el ingreso por teclado, ¿podríamos tener una computadora mucho más pequeña? Sí, podemos, y de alguna manera eso es un microcontrolador, un circuito integrado, con CPU, memoria y conexión para entradas y salidas. Todo esto, en un tamaño muy reducido, no mayor al de una estampilla.

Si lo vemos de esta manera, podemos confundir un microcontrolador con un microprocesador. ¿En qué se diferencian? Principalmente, en que el microcontrolador suma a un microprocesador un conjunto de características electrónicas que lo ha-

III SEGUIDOR DE LÍNEA (LINETRACKER)

La primera prueba sorprendente que realizaremos con un robot, y la competencia más habitual en los torneos de robots, es el seguimiento de una línea. En general, esta prueba está condimentada con otras dificultades, como la presencia de obstáculos que tenemos que esquivar, variaciones de iluminación o de colores en la línea, secciones donde la línea desaparece, etcétera.

cen menos dependiente de los chips de apoyo. Por ejemplo, la memoria, los conversores analógico digitales y el reloj, ya están incluidos dentro de los microcontroladores.

¿Y para qué sirve un microcontrolador? Bueno, tener una computadora de ese tamaño nos permitiría controlar en forma medianamente inteligente diversos dispositivos. Y esto es lo que ocurre alrededor nuestro: en el auto, en la heladera, en el microondas o en el lavarropas, con seguridad podremos encontrar uno o más microcontroladores, que hacen muchísimo más sencilla la electrónica del aparato.

Algunos de los microcontroladores son de propósito específico, es decir, fueron diseñados y optimizados para una tarea determinada. En general, estos microcontroladores no pueden ser reprogramados, por lo que también contamos con micros de propósito general, que pueden ser programados en diversos lenguajes y con destinos de todo tipo.

Las empresas de microprocesadores de propósito general más conocidas son **Microchip** (PIC) y **Atmel** (AVR). Pondremos nuestra atención en la primera de las dos, dado que el micro que usaremos será el **PIC 16F88**. **Microchip Technology Inc.** es una empresa de electrónica de Estados Unidos, surgida de la empresa GI (General Instruments). Del conjunto de productos que ha fabricado, la línea **PIC** (*Peripheral Interface Controller,* Controlador de Interfaz Periférico) de microcon-

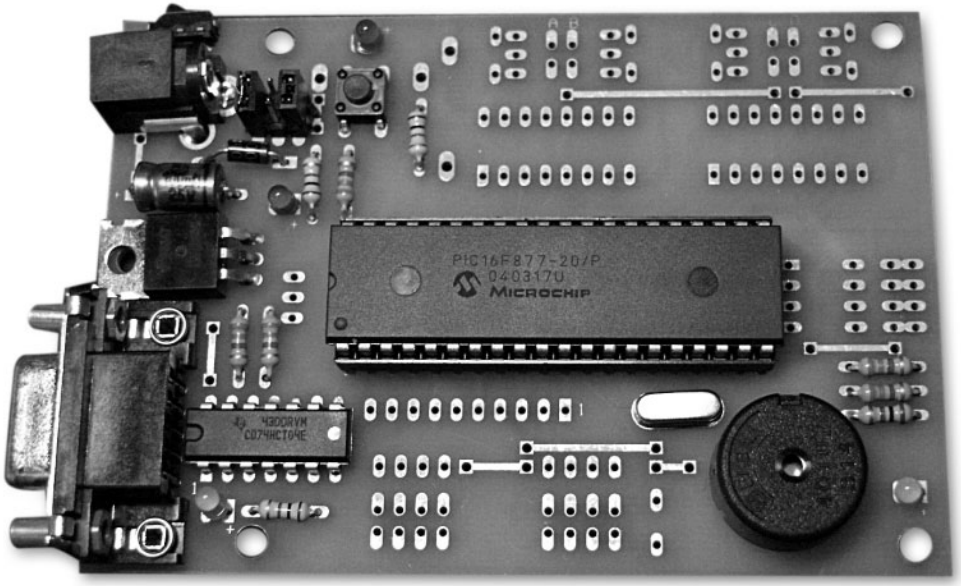


*Figura 4. Un microcontrolador **PIC** montado en una placa controladora **GoGo**.*

troladores ha sido su bastión fundamental (**Figura 4**). Una de las características más importantes de estos micros es que son de tipo **RISC** (*Reduced Instruction Set Computer*, Computadora con Conjunto de Instrucciones Reducido) en contraposición con la arquitectura CISC (*Complex Instruction Set Computer*, Computadora con Conjunto de Instrucciones Complejo). Esto implica que tienen un conjunto reducido de instrucciones de tamaño fijo con respecto a la cantidad de ciclos de reloj (en general, todas las instrucciones necesitan de un solo ciclo de reloj) y que sólo las instrucciones de carga y almacenamiento acceden a la memoria. Las ventajas que estas (y otras) características nos brindan son que la ejecución es más rápida y que permite el paralelismo entre instrucciones mediante una técnica llamada **pipelined**.

La desventaja radica en que la programación con este reducido conjunto de instrucciones tan atómicas implicará más trabajo. Con respecto a los PIC, las características fundamentales que podemos encontrar entre los diferentes modelos son:

- **Núcleo** de 8/16 bits con arquitectura Harvard (las memorias de instrucciones y de datos están separadas, y la CPU accede a ella por buses distintos).
- **Puertos** de E/S con conversores analógicos/digitales.
- **Memoria** Rom y Flash.
- **Temporizadores**.
- **Periféricos** serie.

La arquitectura básica de todos los PICs contiene una memoria de datos, una memoria de programa, una unidad aritmético lógica, una unidad de control y puertos de E/S (entrada y salida). De todos los modelos, hemos elegido el **16F88** para realizar nuestro robot. Es por eso que en el próximo capítulo nos detendremos a analizar en forma detallada este micro en particular.

¿Cómo le damos movimiento a nuestro robot?

En el primer capítulo, ya hemos descripto el tipo de actuadores que podemos encontrar en un robot. En nuestro caso, usaremos motores de corriente continua (cc) entre 8V y 12V, con un consumo no superior a 100 mAmp, que describiremos en profundidad más adelante. Estos motores son más económicos y sencillos de controlar, pero en general carecen de precisión y fuerza (**Figura 5**). Para solucionar este problema, recomendamos los motores con caja de reducción incorporada (conocidos como **motorreductores**), que poseen un pequeño motor que gira a muchísimas revoluciones, unido a un sistema de reducción que permite obtener fuerza con menor velocidad. Para nuestro robot utilizaremos un **sistema de tracción**

diferencial, dado que es el más sencillo de controlar desde nuestro programa (**Figura 6**). Este sistema utiliza dos ruedas convencionales ubicadas en el centro del robot, una de cada lado, conectadas directamente a los motores. Esto nos dará la posibilidad de ir para adelante o para atrás, de aplicar la misma energía en ambos motores, o de realizar giros según la diferencia de velocidad de cada motor. El recorrido en forma recta es difícil de conseguir en esta arquitectura dado que siempre encontramos diferencias de velocidad entre los motores o en el giro de las ruedas, desplazamientos, etcétera. Una de las maneras de controlar esto es realizar un **sensado** del mundo. Por ejemplo, si tenemos que seguir una línea negra, bastaría con tener uno o dos sensores de luz que nos permitan detectar nuestra desviación de la línea para modificar la velocidad de los motores. Otra manera es sensar el movimiento de las ruedas con encoders similares a los que se utilizaban en un mouse antiguo. El problema de este mecanismo es que necesitamos de electrónica específica para este control, algo que escapa a nuestro primer proyecto. Por eso, por ahora nos conformaremos con que vaya medianamente en línea recta.

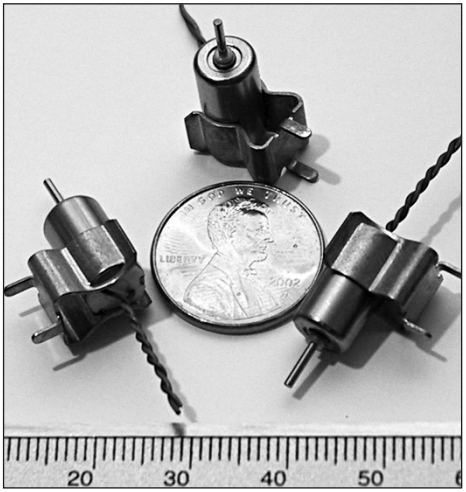


***Figura 5.** Motores de corriente continua de tamaño reducido.*

Otra arquitectura de movimiento interesante es la **omnidireccional**. Con tres ruedas especiales ubicadas como podemos ver en la **Figura 7**, podemos dirigir al robot hacia cualquier punto, sin necesidad de girarlo en esa dirección. Esta forma de navegación es muchísimo más efectiva, ya que aho-

PICS ESPECIALES

En la actualidad, tenemos modelos muy sorprendentes como el **rfPIC**, que incorpora la posibilidad de comunicación wireless (¡en el mismo micro!) y los **dsPIC**, que tienen un bus de datos de 16 bits que permiten realizar operaciones de procesamiento digital de señales. También podemos encontrar clones, más baratos o con mejoras, como los que realiza la empresa **Ubicom**.

***Figura 6.** Vista desde abajo de un robot de **tracción diferencial**.*

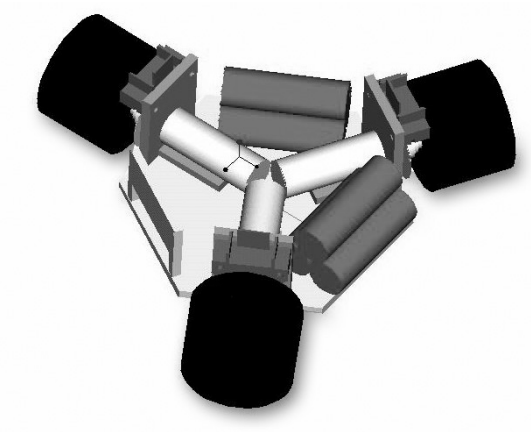


***Figura 7.** Arquitectura de un robot con ruedas omnidireccionales.*

rramos tiempo al movernos siempre en línea recta. Pero el costo de las ruedas es elevado, y se necesita de algoritmos algo más complejos para calcular la velocidad que debe tener cada rueda para tener la dirección deseada. Una vez que hemos decidido el tipo de motor y la arquitectura de las ruedas que vamos a usar, ¿dónde podemos conseguir los motores? Desde ya, una de las formas más sencillas de conseguirlos es comprarlos.

Sin embargo, si tratamos de ahorrar ese dinero, va a ser necesario extraer los motores de algún aparato que tengamos en desuso. El problema de hacer esto es que como estos motores no tienen reducción, tendremos que ingeniárnosla para darles fuerza (desde ya, se perderá velocidad). Los aparatos de donde podemos conseguir motores de corriente continua son:

- **Impresoras**: habitualmente contienen motores de corriente continua y paso a paso.
- **Juguetes**: autos o cualquier juguete con movimiento. En general, son motores muy pequeños.
- **Limpiaparabrisas**: hay de 12V y de 24V. Debemos conseguir los del primer grupo.
- **Reproductores de casetes y videos**.

Si tenemos algún servo desperdiciado, podemos modificarlo para utilizarlo como motor CC. Los servos son un motor de ese tipo con una caja reductora y un circuito de control y tope físico, que limita su giro a 180 grados. Para convertirlo en lo que necesitamos, debemos eliminar el circuito de control y la limitación física, y de esta manera aprovechar la

caja reductora que posee. Los pasos necesarios para llevar a cabo esta tarea los veremos más adelante.
Como dijimos previamente, para construir nuestro robot utilizaremos dos motores de CC de 12V, con un consumo de 100 mAH.

¿Y cómo captamos el mundo que nos rodea?

Como ya vimos en el capítulo anterior, un robot sin sensores es incapaz de cambiar su comportamiento para adaptarse al medio que lo rodea, porque no puede captarlo. Es por eso que la elección que haremos de los sensores es fundamental. En nuestro caso, utilizaremos sensores de contacto para detectar colisiones y sensores de luz o infrarrojos para detectar las líneas negras y la pelota infrarroja.
Comencemos con el **detector de colisiones**. ¿Qué podemos utilizar para ello? En realidad, cualquier dispositivo que funcione como **interruptor** es suficiente. Los sensores de contacto que se utilizan en robótica industrial se conocen como **final de carrera**.

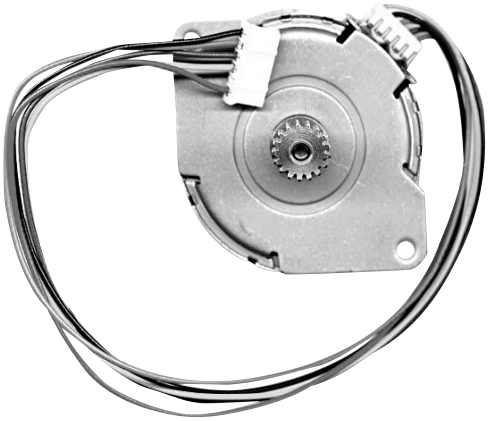


***Figura 8.** Motor paso a paso unipolar.*

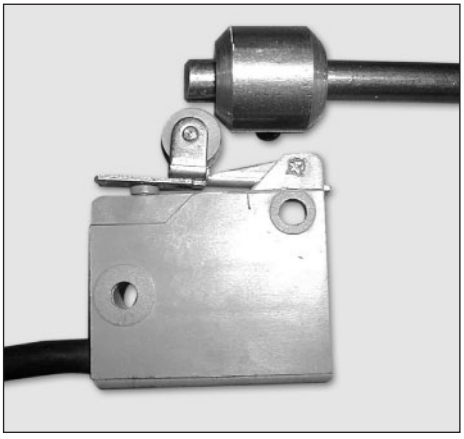


***Figura 9.** Un ejemplo de sensor de final de carrera.*

MOTORES DE CORRIENTE CONTINUA

Si no sabemos dónde conseguir motores de corriente continua, podemos visitar los sitios de los proveedores de este tipo de motores:

- Ignis: **www.ignis.com.ar**.
- Baldor: **www.baldordistribuidora.com**.
- Telco: **www.telcointercon.com/telcomotion/brushed/micro_dc.htm**.
- Micro-Drives: **www.micro-drives.com**.

***Figura 10.** Micro switch utilizado habitualmente como sensor de contacto.*

***Figura 11.** Un **LDR** utilizado para la detección de luz en un robot.*

Este sensor debe tener algún mecanismo para que, una vez activado, luego vuelva a su posición inicial. Por otra parte, es necesario utilizar algún **amplificador** de la mecánica del final de carrera, según el tamaño y la posición del objeto que es necesario detectar. Como se puede ver en la **Figura 10**, el espacio de detección es muy pequeño, por lo que necesitamos construir un paragolpes (bumper). También podemos construir nosotros este sensor con la arquitectura que más nos plazca. Simplemente, tiene que cerrar (o abrir) el circuito cuando se activa, y volver a su situación inicial cuando está desactivado. Los interruptores de final de carrera se pueden adquirir en negocios de electrónica, y también se pueden extraer de elementos como:

- **Mouse**: podemos usar los interruptores que son presionados por los botones. El problema es que la amplificación es complicada de realizar por la forma de éstos.
- **Impresoras**: para detectar si la tapa está cerrada o no, o para el movimiento del cabezal, en general se utilizan estos mecanismos.
- **Celulares**: aquí podemos hallar versiones pequeñas de los interruptores para detectar el cierre de la tapa del celular.

III SENSORES DE CONTACTO ESPECIALES

Podemos realizar un sensor para detectar piezas metálicas si utilizamos escobillas de autitos de pistas de carrera. En este caso, las escobillas impares están conectadas a uno de los cables, y las pares al otro. De esta manera, cuando dos escobillas tocan el objeto metálico que se encuentra en el piso, se cierra el circuito y el controlador determina que se ha detectado un objeto.

Para la detección de luz, podemos utilizar un sensor de luz convencional, conocido como **fotorresistencia** (LDR, *Light-Dependent Resistor*, resistencia que depende de la luz). Éste es un componente electrónico cuya resistencia disminuye a medida que la luz incide sobre él (**Figura 11**).

Es económico y sencillo de usar, pero es altamente sensible a la luz ambiente. Por esa razón, es bueno añadirle un led emisor para poder captar el rebote de la luz de este componente sobre la superficie a detectar. Para evitar el problema de la luz ambiente, podemos utilizar un **sensor infrarrojo**. El funcionamiento es similar al anterior, pero son menos sensibles a la luz ambiente, ya que sólo detectan luz dentro del rango del infrarrojo. Está compuesto por un diodo emisor infrarrojo en una determinada longitud de onda y un receptor (en general un fototransistor) que apunta en la misma dirección que el emisor. Un buen ejemplo es el sensor activo **CNY70**, que puede captar objetos entre 5mm y 10mm de distancia (**Figura 12**).

Ambos sensores son económicos, y pueden conseguirse en cualquier negocio de electrónica. Es posible conseguir alguno de ellos si desarmamos algún juguete (por ejemplo, los muñecos que detectan la presencia de un objeto por delante y emiten algún sonido), pero son tan económicos que no vale la pena tomarse ese trabajo.

***Figura 12. CNY70**, un sensor infrarrojo muy popular.*

***Figura 13.** Aquí podemos ver un robot hecho con piezas de Lego, pero con sensores y controlador propios.*

Materiales para la mecánica

A la hora de construir el chasis de nuestro robot, tenemos múltiples opciones con respecto a los materiales que usaremos. Es importante tener en cuenta que cuanto más liviano sea el robot, necesitaremos motores menos potentes y, por lo tanto, menos baterías, lo que finalmente es más económico. Excepto en los casos en los que el peso del robot presente una ventaja, como puede ser la lucha de sumo, un robot más liviano siempre es mejor. A conti-

Figura 14. *Aquí podemos ver un* ***Robosapien*** *modificado, donde el controlador es una* ***Pocket PC****.*

nuación presentaremos una lista de los que se pueden conseguir con más facilidad en el mercado, con sus respectivas ventajas y desventajas.

Piezas de Lego, Mecano u otros juguetes constructivos

Seguramente, en algún rincón de nuestro baúl de los recuerdos tenemos una bolsa con las piezas de nuestro querido sistema de construcción de la infancia. Si hemos cometido el error de prometerlas como regalo a un vecino pequeño o al hermanito de nuestra novia, debemos olvidarlo. Estas piezas son muy prácticas para realizar prototipos de nuestros robots (¡que muchas veces quedan como versiones finales!). Una de las ventajas fundamentales es su **facilidad de uso** y su versatilidad para lograr las formas deseadas (**Figura 13**). Otra ventaja es la **reusabilidad**: una vez terminado el robot, podemos desarmarlo y volver a usar el material.

Por el lado de las desventajas, la más importante es la fragilidad de la estructura que podemos lograr. Y si no tenemos las piezas de antemano, su valor es muy alto en comparación con los otros tipos de materiales.

Juguetes usados

Cuando queramos construir un robot que imite algún vehículo tradicional, una muy buena opción es montarlo sobre un juguete usado, o por qué no,

comprar uno con este propósito. Por ejemplo, para construir un humanoide es más sencillo comprar uno ya armado y montar la inteligencia sobre la mecánica ya resuelta, que construirlo desde cero (**Figura 14**).

Si partimos de un juguete abandonado, podemos considerarlo como costo cero. De todas formas, con seguridad tendremos que añadir partes desarrolladas por nosotros para poder adaptar el controlador, los sensores, etcétera. En este caso, una desventaja puede ser la dificultad de ensamblar la estructura del juguete con las nuevas partes del robot. Otro problema es que, en general, las piezas del juguete son frágiles, por lo que deberemos tratarlas con cuidado. Es recomendable incorporar las partes nuevas con un pegamento adecuado, y no agujerear el juguete. Por último, la reusabilidad de los materiales que utilicemos es muy baja, dado que están adaptados a este proyecto en particular.

PVC

Es un material plástico muy económico que se presenta en formato rígido o flexible. Es muy fuerte y resistente, aunque se puede cortar y taladrar con facilidad. Para las carcasas cilíndricas, podemos utilizar caños de este material. Si necesitamos deformarlo, el PVC se ablanda al llegar a los 80º (por ejemplo, sobre una hornalla) y de esta manera, podremos manipularlo con cierto cuidado. Para unir piezas, se utilizan pegamentos del tipo **cianocrilato**. Se puede cortar con una sierrita de dientes pequeños, y es posible quitar las rebabas con una lija fina.

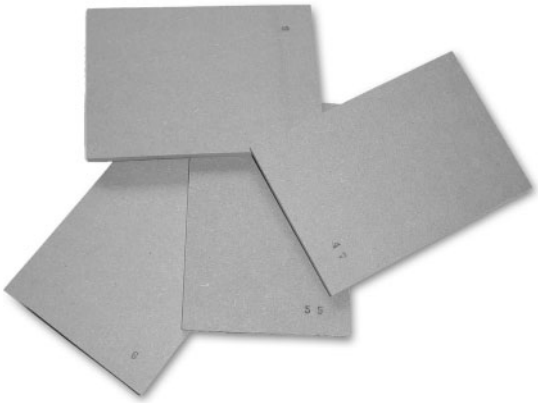


***Figura 15**. Planchas de aglomerado.*

Aglomerado

Es una plancha realizada con fibras de madera pegadas a presión con una resina sintética (**Figura 15**). También es conocido como **MDF** (*Medium Density Fibreboard*, Tablero de fibra de densidad media). Es económico como el PVC y puede conseguirse en cualquier maderera. Es fácil de manipular pero es más complejo conseguir terminaciones prolijas. Los chasis realizados con este material son robustos, pero su peso es una desventaja que hay que tener en cuenta. Tampoco es aconsejable que el material se moje, aunque es resistente a salpicaduras.

Figura 16. Robot con piezas de acrílico.

Poliestireno expandido

Este material es conocido universalmente, pero denominado de diversas formas en todo el mundo: **telgopor**, **porexpan**, **corcho blanco**, **unicel**, **estereofón** o **tecnopor** son algunas de las denominaciones más habituales. Es muy liviano y absorbe muy bien los impactos, pero su resistencia es limitada. Es fácil de manipular con cortadores en caliente, aunque sus terminaciones son complicadas. Por su bajo costo, es **ideal para realizar prototipos**, pero su baja resistencia no permite la construcción del modelo final de un robot.

Acrílico

En realidad, llamamos acrílico al **metacrilato** compuesto por ácido metacrílico. Cuando vemos un robot realizado íntegramente en este material, y en especial si es transparente, todos nos enamoramos. Es muy atractivo, muy resistente, liviano, y de fácil corte con sierras y caladoras, aunque hay que realizarlo con paciencia porque el calor de la fricción puede derretir el material (**Figura 16**). Otro problema del corte es el polvillo que genera, por lo que se recomienda el uso de mascarillas para hacerlo. El moldeo es un poco más complicado, ya que necesita de calor local, y se realiza con pistolas de aire caliente o bandas resistivas. La desventaja más grave de este material es el precio, que supera ampliamente los materiales que ya comentamos.

Aluminio

Es el material más usado en la fabricación de las versiones finales de los robots. Es muy resistente, liviano y de

LOS MICROS DE ATMEL

La familia de micros de Atmel se llama **AVR**. También son RISC con 32 registros de 8 bits. Fueron diseñados para ejecutar en forma eficiente código C compilado. Aunque el núcleo de instrucciones es el mismo, existen diversos modelos, que van desde los más pequeños con 1KB de memoria flash y sin RAM, hasta los más poderosos con 256 KB de memoria flash, 8KB de RAM, 4 KB de EEPROM, conversor analógico digital y otras maravillas.

relativo bajo precio. Si tenemos que hacer cortes sencillos, podemos trabajar el aluminio con herramientas convencionales, pero si necesitamos piezas más complejas debemos contar con un torno específico. Por otra parte, para moldearlo necesitamos hornos especiales a altas temperaturas.

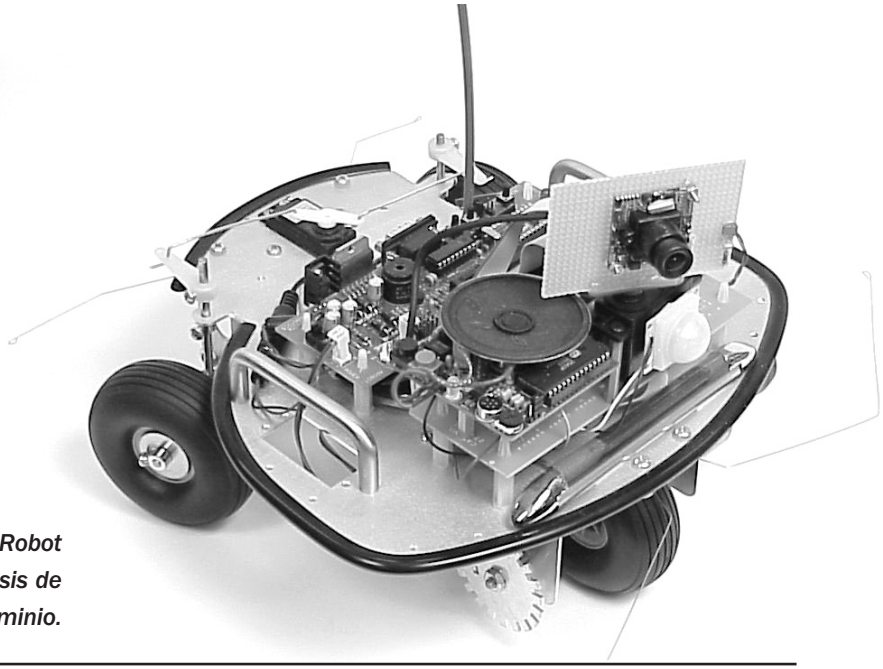


Figura 17. *Robot con chasis de aluminio.*

... RESUMEN

En este capítulo presentamos los objetivos de nuestro robot y los materiales necesarios para construirlo. Hemos decidido construirlo en forma artesanal desde cero, para tener costos más bajos y para lograr un conocimiento más profundo de la arquitectura del robot. El objetivo de nuestro trabajo es conseguir un robot autónomo, capaz de seguir una línea negra, dirigirse a sitios más iluminados y detectar objetos que se interpongan en su camino. Para ello, utilizaremos un procesador PIC 16F88 de la empresa Microchip. El movimiento lo lograremos con dos motores de corriente continua de 12 Voltios. Para detectar el mundo, vamos a agregarle sensores de contacto para obstáculos físicos y fotorresistencias, y sensores infrarrojos para la detección de luces y líneas de colores. Para la construcción del chasis de nuestro robot, hemos elegido el aglomerado, dado que nos permite construir una estructura robusta, con herramientas sencillas y a muy bajo precio.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Por qué decidimos desarrollar el robot en forma artesanal?

2 ¿Cuáles son los objetivos de nuestro robot?

3 ¿Qué es un microcontrolador?

4 ¿Qué diferencia un microcontrolador de un microprocesador?

5 ¿Qué características presentan los micros con tecnología RISC?

6 ¿Cuál es la arquitectura básica de todo PIC?

7 ¿Qué ventajas y desventajas presentan los motores de corriente continua?

8 ¿Qué características presenta la tracción diferencial?

9 ¿Qué es un final de carrera?

10 ¿Qué ventajas presenta el sensor infrarrojo sobre el LDR?

11 ¿Cuáles son los tipos de materiales que presentamos para la construcción del chasis?

La inteligencia del robot

Antes de comenzar a construir nuestro robot, daremos una mirada general a todos los componentes que lo conformarán. Desde ya que elegiremos puntualmente alguno de ellos, pero con este capítulo nos será sencillo seleccionar otras variantes según el objetivo que tengamos para nuestro robot.

EL CEREBRO

Cuando Asimov creó sus primeros robots de ficción, aún no existían las computadoras. Tuvo que inventar los cerebros positrónicos, que son dispositivos con inteligencia artificial que le permiten al robot almacenar información, realizar operaciones lógicas y tomar sus propias decisiones. Con esa capacidad sorprendente de adelantarse décadas en el tiempo, Asimov realizó una descripción que, en muchos casos, coincide con los controladores que tenemos en la actualidad. De este tema, y con la intención de emular la genialidad de Asimov, hablaremos en este capítulo. Hay diversas formas de controlar un robot. Esencialmente, varían en función de nuestra necesidad y de lo que queremos hacer con él. Los primeros robots fueron simples autómatas, como el jugador de ajedrez, que no son más que una sucesión de combinaciones de engranajes, poleas y otros artilugios electromecánicos, en donde el creador articula el robot en forma mecánica, y gobierna así sus movimientos. Luego surgieron los robots con electrónica analógica (transistores) y compuertas digitales, que en algunos casos hasta pueden programarse en forma muy primitiva con **álgebra de boole**. Por último, en la actualidad, nuestros robots poseen microcontroladores y computadores, como comentamos en el primer capítu-

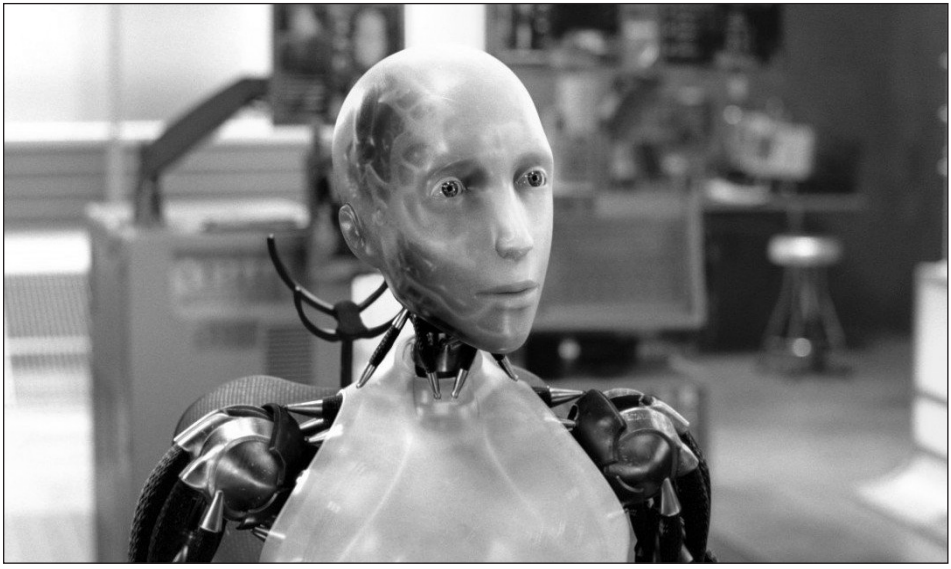


Figura 1. *Robot con cerebro positrónico de la película* ***Yo, robot****, inspirado en la obra de Asimov.*

lo. En nuestro caso, cuando elegimos el controlador que vamos a presentar en este libro, optamos por una solución del último espectro, pero con la suficiente sencillez para su construcción y la suficiente complejidad para poder lograr comportamientos medianamente inteligentes.

Para el desarrollo de nuestro robot necesitamos algunas habilidades de electrónica que podremos adquirir con facilidad. Esto es necesario porque así tendremos un robot con prestaciones iguales o mayores que las de algunos de los kits que se pueden comprar y principalmente nos permitirá dar los primeros pasos en la utilización de microcontroladores.

Componentes de nuestro robot

A la hora de diseñar nuestro controlador, es necesario tener presente qué tipo de robot deseamos construir, porque esto nos dará una noción de la cantidad de actuadores y sensores que necesitaremos. Como definimos en el capítulo anterior, nuestro robot tendrá dos ruedas con dos motores, que nos permitirán controlar la dirección de manera diferencial. Contará con dos **bumpers** (sensores de choque) y un sensor analógico para poder seguir una línea (robot rastreador). Si tenemos en cuenta esto, podemos definir las partes de nuestro robot en:

- **Chasis**: éste es el armazón que contiene el controlador, los motores y las baterías.
- **Alimentación**: batería de 9 V, conjunto de pilas, o batería de gel de 12 V.
- **Control**: controlador programable multipropósito.

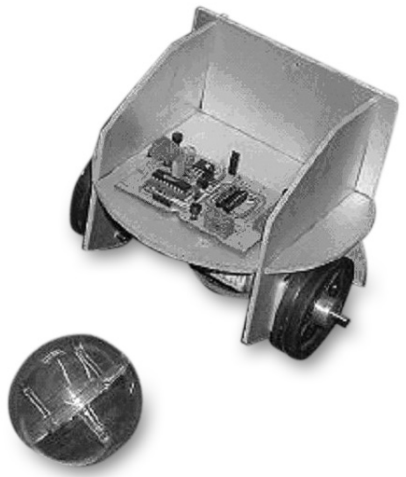


***Figura 2**. Un robot cuyo cerebro es nuestro controlador.*

CEREBROS POSITRÓNICOS

Todos los robots de la obra de Asimov poseían cerebros positrónicos, que consistían en una malla de platino donde la actividad cerebral estaba dada por un flujo de positrones. Si no tenemos en cuenta el punto de vista literario, es difícil imaginar que el positrón (antipartícula del electrón) no destruya la malla de platino, en una colisión materia – antimateria.

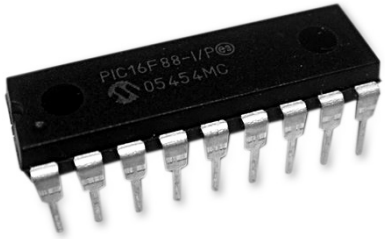


Figura 3. Una de las versiones del PIC16F88.

- **Driver H**: interfaz de potencia para los motores de corriente continua con inversión de giro.
- **Sensores**: dos sensores de colisión ubicados al frente del robot.
- **Monitor**: un led de monitoreo del programa.
- **Sensor analógico**: para futuras aplicaciones.

Objetivos del controlador

Dados los objetivos que hemos definido, el controlador debe cumplir con las siguientes características:

- Poder controlar dos motores de corriente continua de 12 V de hasta 400 miliamperes.
- Permitir que el robot se desplace en todas direcciones mediante una tracción diferencial (de dos ruedas).
- Ser programable en circuito mediante un programador conectado a la computadora.
- Utilizar baterías comunes de diferente voltaje entre 7 y 12 V.
- Poseer la capacidad de conectar sensores de diversos tipos, tanto analógicos como digitales.
- Poder configurar las entradas y las salidas según el objetivo del robot.
- Permitir controlar una gran variedad de robots.

El microcontrolador, cerebro del cerebro

Como comentamos antes, el principal componente de nuestra placa controladora es el microcontrolador. De su elección depende el tiempo de utilidad de ésta, ya que nos permitirá aumentar la complejidad y adoptar diferentes configuraciones de robots, simplemente preocupándonos por cambiar la mecánica/electromecánica cuando sea necesario. En síntesis, el objetivo es contar con nuestro propio controlador universal de robots.

En el **Capítulo 2** adelantamos nuestra elección por el micro 16F88 de la firma Microchip (**Figura 3**). Cabe mencionar que aunque el 16F84 es el micro más difundido de esta empresa, hemos decidido dar un paso adelante por dos razones: el nuevo modelo nos da muchas prestaciones adicionales, y todo lo desarrollado para 16F88 que no utilicen esas prestaciones es absolutamente compatible con el 16F84.

En el presente libro, en gran parte usaremos del micro nuevo aquellas funcionalidades que sean compatibles con el anterior. Pero así nos queda armado un controlador con amplia capacidad de expansión.

Conceptos fundamentales de un PIC

Como en el punto siguiente queremos realizar una presentación del 16F88 y compararlo con otros micros, a continuación describiremos un conjunto de términos necesarios para poder comprender mejor. Los conceptos que vamos a explicar son fundamentales en la arquitectura y el funcionamiento de un PIC. Si ya tenemos experiencia, esto nos resultará conocido y podremos continuar hasta el punto siguiente.

Si no, aquí encontraremos un buen diccionario de palabras vinculadas al que podremos recurrir en los próximos capítulos.

- **Memoria flash**: es el área de almacenamiento de los programas. Puede ser reescrita miles de veces, lo que presenta una gran ventaja para reutilizar el controlador y para programar sin temor al error.
- **RAM** (*Random Access Memory*, memoria de acceso aleatorio): almacena todas las variables y los datos intermedios del programa.
- **EEPROM** (*Electrically-Erasable Programmable Read-Only Memory*, ROM que se puede programar y borrar eléctricamente): en esta memoria se almacenan los datos que deben conservarse aun cuando haya pérdida de energía. Puede ser reescrita en más oportunidades que la memoria flash.
- **I/O Ports** (*Input/Output Ports*, puertos de entrada/salida): es la comunicación del PIC con el mundo que lo rodea. Desde allí podrá enviar señales a la electrónica externa y recibir datos de ésta.
- **Timers** (temporizadores): en general, todos los PICs poseen hasta tres temporizadores, con diversas capacidades y funciones. Se utilizan como contadores, relojes y generadores de interrupciones.
- **USART** (*Universal Synchronous Asynchronous Receiver Transmitter*, transmisor y receptor sincrónico y asincrónico universal): módulo que presentan los PIC que nos brindan un puerto serie.
- **CCP** (*Capture/Compare/PWM*, Captura/Comparación/Modulación por ancho de pulso): este módulo tiene tres modos de comparación.

MODULACIÓN POR ANCHO DE PULSO

La modulación por ancho de pulso (**PWM**, *Pulse With Modulation*) se utiliza para determinar la velocidad de giro de los motores eléctricos. En el pulso tenemos un momento alto (con corriente) y un momento bajo (sin corriente). Según la longitud de estos pulsos, logramos la velocidad de nuestro motor.

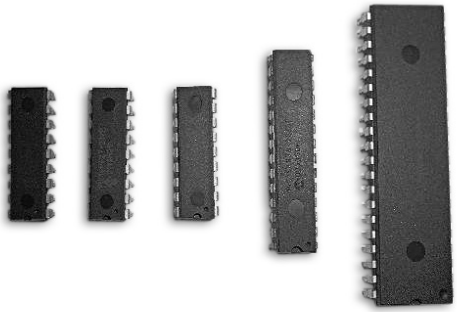


Figura 4. Otros modelos de PIC vinculados al 16F84 y el 16F88.

En el primer caso (Capture) nos permite capturar el tiempo de un evento. En el segundo (Compare) genera una salida cuando el timer 1 alcanza un valor. El último nos brinda una salida de 10 bits de resolución sin consumo de ciclos con una determinada frecuencia, configurada por el timer 2.

- **Comparador**: es un módulo que brinda dos comparadores analógicos, de manera tal que las entradas analógicas o digitales se puedan comparar con los voltajes de referencia.
- **ICSP** (*In Circuit Serial Programming*, programación en circuito): los PICs se pueden programar sin tener que sacarlos del controlador, conectándolos con el programador mediante un cable especial que describiremos más adelante.

Características del PIC16F88

El PIC16F88 es un salto importante con respecto al PIC16F84, una suerte de hermano mayor más avanzado. En realidad, es un reemplazo de los 16F87X, que mantiene compatibilidad con sus antecesores para hacer más sencilla la migración. Tiene 35 instrucciones de una sola palabra, de 200 nanosegundos de ejecución. Además, cuenta con un oscilador interno de 8MHz, 256 bytes de memoria EEPROM, 368 bytes de RAM y 4 Kb de memoria flash. Viene con un USART, un puerto serie sincrónico, tres timers, un módulo CCP y un comparador. Para los sensores disponemos de siete canales

III REDES NEURONALES

Existen diferentes propuestas de emulación del procesamiento de nuestro cerebro. Entre otras, las redes neuronales son un sistema de **interconexión de nodos** que simulan neuronas que puede implementarse por hardware o por software. Estas neuronas reciben diversas entradas de información y mediante un conjunto de funciones determinan la salida de la neurona. El arquitecto de la red define su estructura y luego, mediante un sistema de aprendizaje, entrena a la red para los datos que luego deberá procesar. Luego del aprendizaje, la red es capaz de interpretar los datos de entrada y brindar una salida acorde.

de entrada con conversor analógico digital. Su consumo es muy bajo, para suerte de nuestras baterías (aunque los motores serán nuestros enemigos en este aspecto).

Compatibilidad con el 16F84

A pesar de las grandes capacidades que hemos mostrado del 16F88, la compatibilidad con el 16F84 es muy alta, lo que permite utilizar hardware y software desarrollado para el micro más antiguo. En primer lugar, tienen el mismo **pinout** (asignación de los pines). El 16F88 puede ejecutar el mismo código que el 16F84 con mínimas variantes, y se puede programar con el mismo hardware. Es decir, si tenemos un robot con un 16F84, es posible que podamos hacerlo funcionar con el 16F88, ¡y a un precio menor! Efectivamente, este último modelo ha sido lanzado a menor precio que su antecesor. Para esta versión del controlador hemos armado una placa experimental en la que no es necesario tener el 16F88 ya que es compatible con el 16F84. Por lo tanto, podremos usar el antiguo micro (incluso los primeros programas estarán desarrollados para el 16F84). La necesidad del 16F88 se hará presente a medida que avancemos en la utilización de sensores analógicos, la utilización del **bootloader** o el agregado de una interfaz serial RS232, entre otras cosas.

Puente H para el control de los motores

Para poder controlar la velocidad y la dirección de los motores de corriente continua, se utiliza un circuito deno-

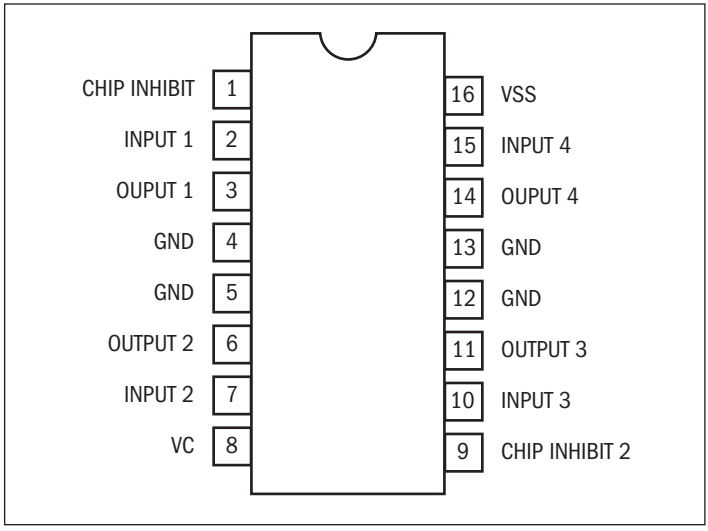


***Figura 5**. Esquema del integrado controlador de los motores L293D.*

minado **Puente H**. Este curioso nombre proviene de la forma en la que antiguamente se posicionaban los transistores al realizar el puente. La electrónica ha avanzado y en la actualidad utilizamos integrados que nos proporcionan esa funcionalidad, en un espacio menor y a bajo costo. En nuestro caso, la interfaz de potencia se basa en el **chip L293D**, que cuenta con diodos de protección para los motores (**Figura 5**). Además, posee cuatro drivers de potencia que nos permiten controlar dos motores de hasta 600 miliampers de pico (con inversión de giro) o cuatro motores simples, controlados por dos líneas de comando de 5 V por motor conectados directamente al micro.

Los pines 2, 7, 10 y 15 son entradas digitales que sirven para controlar la dirección de los motores. Los pines CHIP INHIBIT (1 y 9) permiten controlar la velocidad con la técnica PWM ya mencionada. Los motores se conectan a 3, 6, 11 y 14. VSS (voltaje lógico) es la que alimentará o dará potencia al motor.

IDENTIFICADOR	VALOR
C1	100uF / 50V
C2	.1uF
C3	.1uF
C4	47uF / 25V
C5	.1uf
C6	.1uf
C7	22p
C8	22p
D1	DIODE
IC1	16F88
IC3	L78L05
IC4	L293D
ISCP Programador	Conector 5 pines
J3	Bornera 3
L1	LED Verde
L2	LED Rojo
Motor1	Bornera 2
Motor2	Bornera 2
POWER	Bornera 2
R1	390
R2	390
R3	1K
R4	1K
R5	10K
Reset	Microswitch
Sensor1	Microswitch
Sensor2	Microswitch
XTAL 4Mhz	4Mhz

Tabla 1. Lista de componentes.

Listado de componentes del controlador

La mayoría de los componentes que vamos a utilizar (**Tabla 1**) se puede conseguir en cualquier negocio de electrónica, y se puede reemplazar por otros de valores aproximados. Los elementos más costosos son los dos procesadores, el 16F84 ó 16F88 y el L293D. Otro componente de costo alto es la batería recargable de 9 V, de la cual hablaremos más adelante. Para poder conectar todos estos

componentes, podemos usar una **protoboard** (**Figura 6**) o directamente armar el circuito en una plaqueta experimental. En el primer caso, el tamaño de nuestro controlador será incómodo para montarlo sobre el robot, pero puede ser útil para realizar un primer prototipo de prueba. Una vez que tengamos todo en funcionamiento, podemos pasar nuestro desarrollo a una placa experimental.

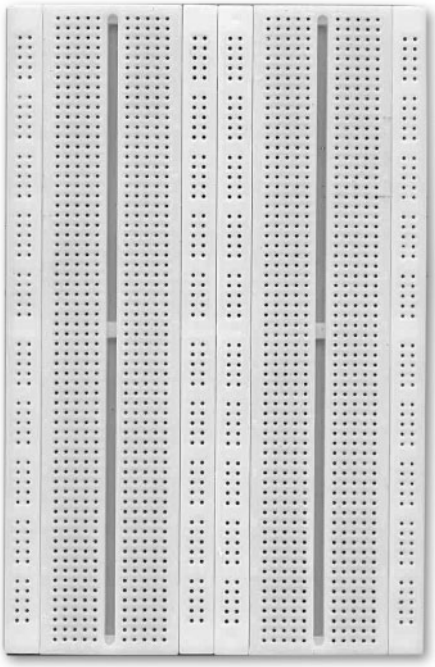


***Figura 6. Protoboard** de buen tamaño que nos permite armar el circuito sin usar soldaduras.*

Descripción del circuito

El circuito es muy sencillo (**Figura 7**). Consta de una fuente de alimentación regulada, basada en un 7805 que nos permite obtener a la salida una tensión estable de 5 V para alimentar a los circuitos integrados, independientemente de que la tensión de entrada esté entre 7 V y 12 V. El

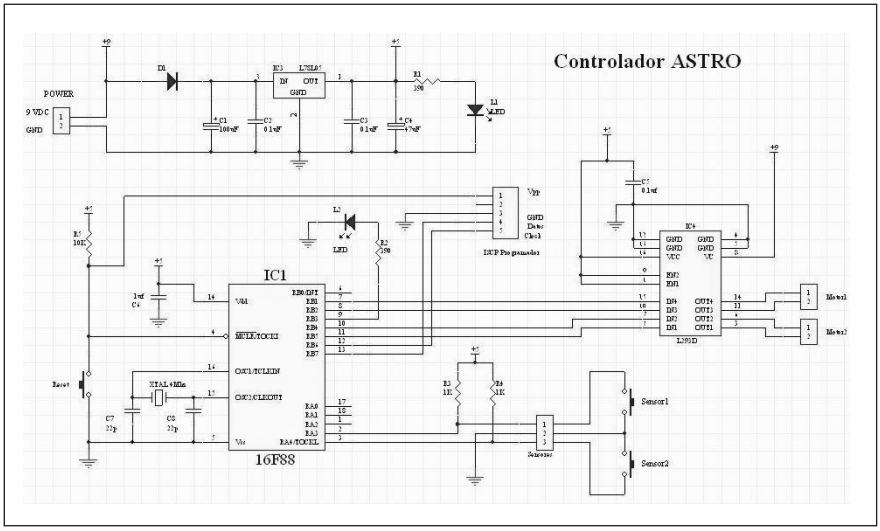


***Figura 7**. Esquema del circuito del **controlador** de nuestro robot.*

diodo D1 permite tener una protección contra la inversión accidental de la tensión de alimentación. Los electrolíticos y condensadores cerámicos actúan de filtros y el led L1 sólo nos sirve para indicar que el circuito se encuentra alimentado.

El **ISCP** es un conector para la programación del micro, que cuenta con un cristal oscilador (XTAL) de 4Mhz (lo podemos evitar en el caso del 16F88 gracias a que contiene un oscilador interno, pero ya que es de bajo costo y más preciso, preferimos mantenerlo en el esquema), y un pulsador de **reset** para bloqueos o reinicialización del sistema. El led L2 se utiliza para monitorear los diferentes ciclos del programa (lo podremos usar para depuración y testeos) y está controlado por RB3. Las entradas RA3 y RA4 permiten sensar los dos switch (llaves) que indican cuándo nuestro robot colisiona con algún objeto. Se encuentran conectadas al positivo mediante dos resistencias R3 y R4 de 1K, que nos sirven de **pullup** para disminuir el disparo por ruido eléctrico que proviene de los motores.

Por su parte, RB1, RB2, RB4 y RB5 se utilizan para controlar dos motores de corriente continua de 12 V y 300 miliamperes. RB6 y RB7 están destinados al programador.

Además de esto, nos quedan libres 4 pines (RB0, RA0, RA1, RA2) que podemos utilizar para propósitos diversos y la posibilidad de utilizar los del cristal o de conmutar los de programación, entre otras modificaciones posibles.

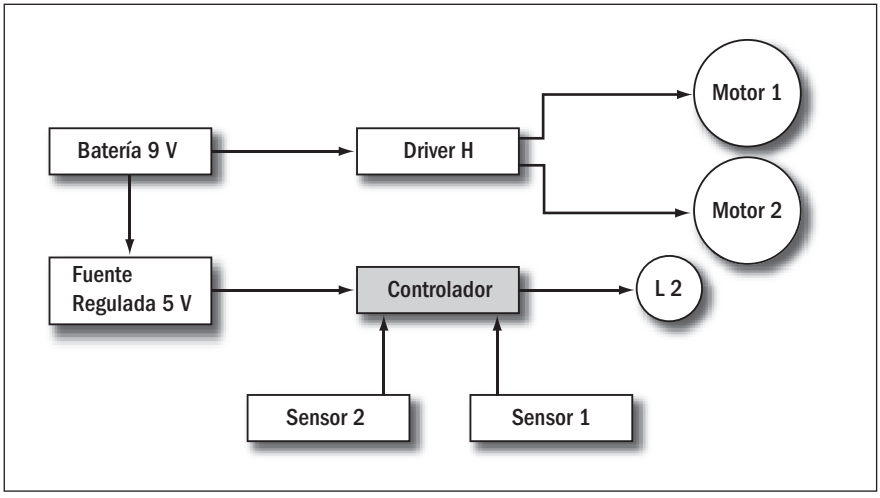


***Figura 8**. En esta figura podemos observar los **bloques lógicos** que componen nuestro controlador.*

Placa experimental

A modo de ejemplo, en la **Figura 9** mostramos la placa experimental básica realizada para las pruebas preliminares, en donde puede observarse el regulador 7805, los integrados 16F84 y L293D y, en la parte inferior a la izquierda y debajo del 16F84, la salida para el programador ICSP.

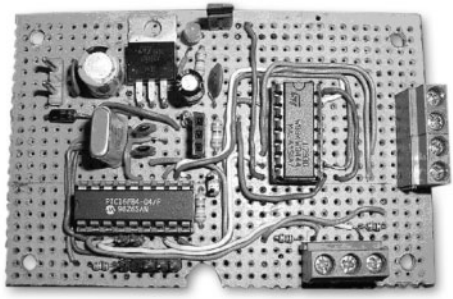


***Figura 9**. Visión final del controlador desarrollado sobre una placa experimental.*

El programador

Cuando diseñamos el controlador del robot, uno de los objetivos fundamentales fue que pudiera ser programado de diversas maneras según el propósito final del robot. De esta forma, podemos reutilizar nuestro controlador en la misma estructura mecánica o en otras, con otros sensores, motores, etcétera. Por eso, nos decidimos a trabajar con el PIC y no con el PICAXE (**Figura 10**), a pesar de que éste último también es una opción. El PICAXE es un micro que ya tiene grabado un pequeño firmware, algo así como un sistema operativo mínimo, que permite cargarle nuestro programa por el puerto serie del computador. Es una buena ayuda porque nos evitamos la construcción del programador. El problema es que si utilizamos un PICAXE dependemos totalmente de este firmware y si por cualquier problema eléctrico se desprogramara el micro, estaríamos en problemas.

La única solución sería comprar un nuevo PICAXE o grabarle el firmware

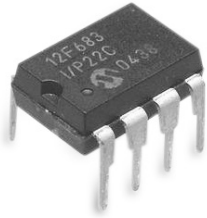


***Figura 10**. **Picaxe 8**, la versión más pequeña de la línea PICAXE.*

de nuevo, para lo cual se necesita ¡un programador! Por lo tanto, tarde o temprano, este recurso nos es indispensable, y ya que compramos componentes y hacemos circuitos, agregamos el programador para quedarnos tranquilos. Además, si alguna vez queremos cambiar de micro, cargar un firmware desarrollado por otro o por nosotros, u otra variante, podremos hacerlo con nuestro programador sin problemas.

Nuestro programador

En Internet podemos encontrar un mundo de aficionados y profesionales que desarrollan sus actividades con PIC. Por esa razón, la oferta de programadores es amplia y si lo deseamos,

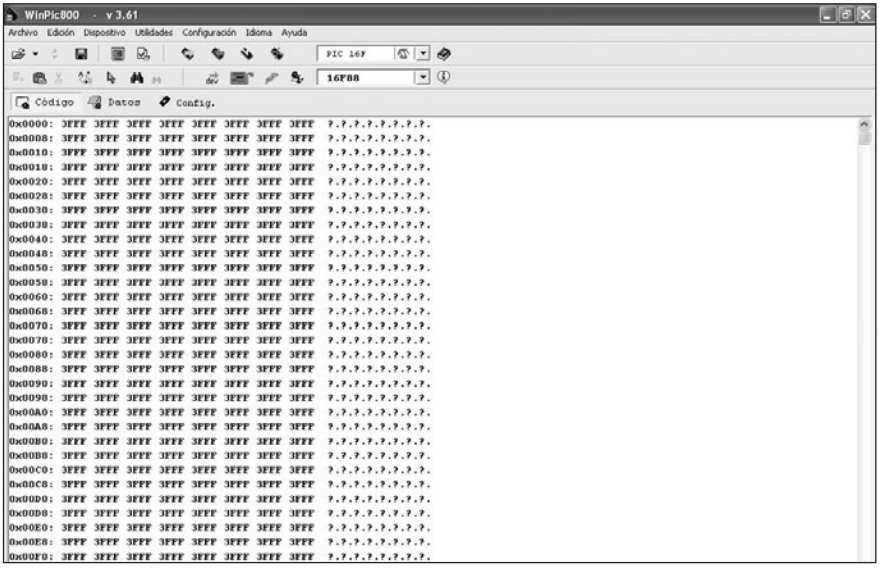


***Figura 11**. Pantalla principal del **WinPic800**, configurada para el 16F88.*

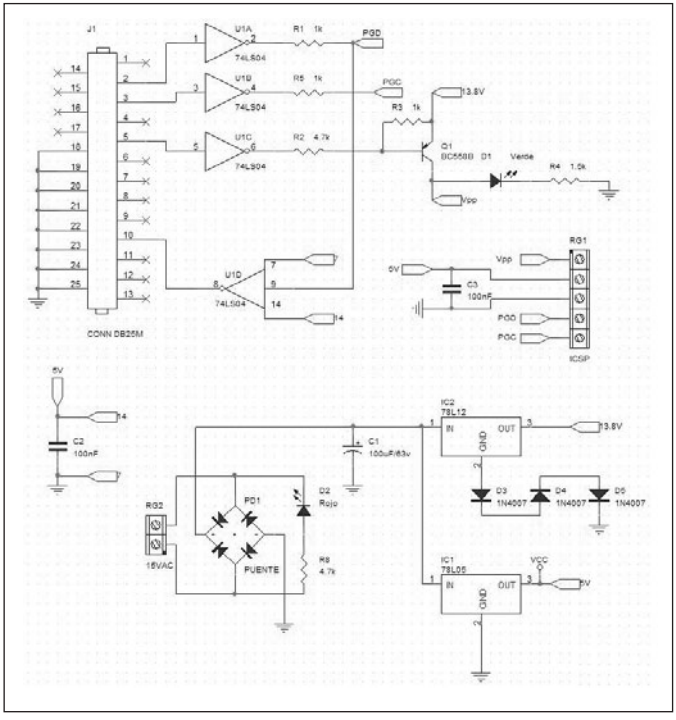


***Figura 12**. Esquema del programador GTP LITE.*

podemos comprar un programador en algún negocio de electrónica. Pero para cumplir nuestra promesa de que el robot será completamente artesanal, presentamos el programador **GTP Lite**. Este programador fue diseñado por Jaime Fernández-Caro Belmonte de HobbyPic (**www.hobbypic.com**). Está basado en un desarrollo anterior del grupo Todopic (**www.todopic.com.ar**) llamado GTP (Grabador TodoPic). Para realizar la grabación de lo que programamos en nuestra PC hacia el micro, utilizaremos el soft **WinPic800** (**www.winpic800.com**), que nos permite acceder a muchos micros distintos (**Figura 11**). Más adelante, nos extenderemos en el uso de este software para bajar nuestros programas al controlador. Con respecto a los programadores, existen otros modelos como el GTP-USB, pero elegimos el mencionado por una cuestión de costos.

El GTP Lite (**Figura 12**) es un grabador ICSP para PICs y memorias que utiliza un puerto paralelo y alimentación externa. Con él se pueden grabar los PICs que soporten el modo de grabación ICSP y que se encuentren en el

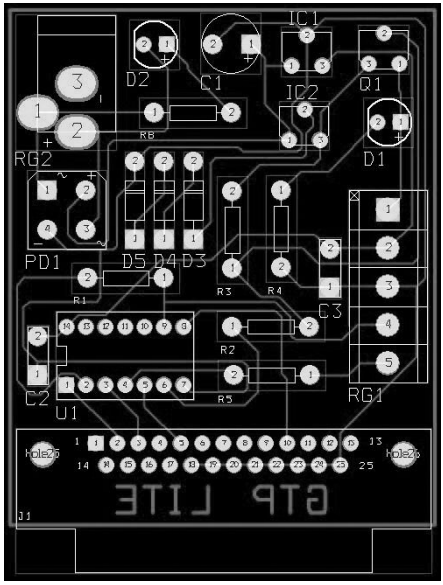


***Figura 13**. En esta imagen podemos ver el board del GTP LITE.*

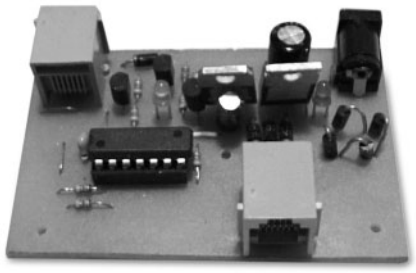


***Figura 14**. Programador GTP LITE terminado, visto desde arriba.*

OTROS PROGRAMADORES

Uno de los programadores más populares del momento es el GTP USB Lite que, como su nombre lo indica, nos permite programar mediante la utilización del puerto USB. Podemos encontrar la información para diseñarlo en **www.todopic.com.ar/foros/index.php?topic=1716.0**. En el siguiente foro podremos encontrar toda la información sobre diversos grabadores de PICs: **www.todopic.com.ar/foros/index.php?board=7.0**.

WinPIC800. La forma más tradicional de grabar el programa en el micro es poner el integrado en el zócalo de nuestra placa programadora y enviarle el programa desde la PC con un software de grabación de PIC, como el Winpic800. Pero en nuestro robot presentamos otra variante. La programación en circuito ICSP (*In circuit serial programming*, programación en circuito) es ideal para trabajar con prototipos, ya que de manera rápida podremos programar nuestro robot sin necesidad de sacar el microprocesador cada vez que cambiemos el programa. Esta programación se basa en utilizar los mismos pines que usa el programador para la programación directa, pero con un circuito muy simple que permite programar el integrado directamente en nuestra placa controladora. Para esto será necesario un conector especial que posee la placa del programador con salida para ICSP. De esta manera, si conectamos el programador a nuestro controlador por este conector, el mismo programa Winpic800 podrá grabar el 16F88 sin tener que desarmar nuestro robot. Como dijimos, esto es muy práctico porque al ser un robot experimental, cargaremos muchos programas y haremos

IDENTIFICADOR	VALOR
C1	100uF/63v
C2	100nF
C3	100nF
D1	Verde
D2	Rojo
D3	1N4007
D4	1N4007
D5	1N4007
IC1	78L05
IC2	78L12
J1	CONN DB25M
PD1	Diodo Puente 1A
Q1	BC558B
RG1	ICSP
RG2	15VAC
R1	1k
R3	1k
R5	1k
R2	4.7k
R8	4.7k
R4	1.5k
U1	74LS04

***Tabla 2**. Lista de componentes del programador.*

PIN	DESCRIPCIÓN	FUNCIÓN	TIPO	DESCRIPCIÓN
12	RB6	Clock	E	Pulsos de reloj
13	RB7	Data	E/S	E/S de datos
4	MCLR	Vtest	MODE	Selección de modo
5	VSS	Vss	P	Masa (Gnd)

***Tabla 3**. Pines del cable para conectar el programador a la controladora.*

pruebas con él en forma constante. En la **Tabla 2** describimos la lista de componentes del programador. Para conectar el programador a nuestra controladora, será necesario fabricar un cable (podemos utilizar alguno de los cables planos que nos sobran de una PC antigua) con los pines de la **Tabla 3**. Recomendamos que el cable no supere los 70 centímetros de longitud. No necesitaremos 5 V ya que usamos masa común entre el programador y la controladora, con la única condición de que nuestro robot deberá estar encendido al momento de programarlo. Este cable es el que conectará nuestra controladora al conector que tenemos debajo del micro.

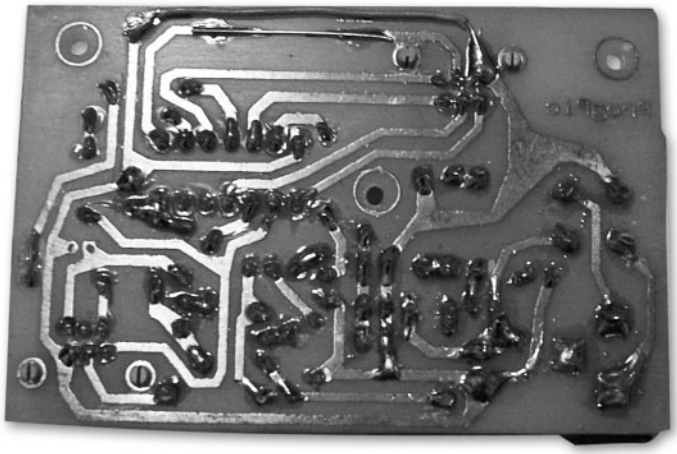


***Figura 15**. El mismo programador, con una toma desde la parte inferior.*

... RESUMEN

Hemos llegado al punto de poner manos a la obra. En este capítulo, conocimos todos los detalles de nuestro controlador y el programador correspondiente. Finalmente, nos decidimos por el micro PIC16F88 de la firma Microchip, en especial por su bajo costo, su buena performance, la compatibilidad con el PIC16F84 y por la posibilidad de programarlo en circuito, sin necesidad de sacarlo de nuestro controlador. Para poder realizar esto, construimos un programador conocido como GTP-LITE, de uso muy difundido entre los usuarios de PIC, y de bajo costo. Para poder realizar la programación, utilizaremos el software Winpic800, que nos permite programar diversos modelos de PICs y de otros tipos de micros.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Cuáles fueron los tipos de control que tuvieron los robots a lo largo de la historia?

2 ¿Cuáles son, esencialmente, los componentes de nuestro robot?

3 ¿Qué objetivos tendrá nuestro controlador?

4 ¿Cuáles son los tipos de memorias que podemos encontrar en un PIC?

5 ¿Qué es el USART?

6 ¿Qué significa ICSP y qué ventajas presenta?

7 ¿Por qué elegimos el microcontrolador PIC16F88?

8 ¿En qué aspectos es compatible el 16F88 con el 16F84?

9 ¿Qué es un puente H? ¿Con qué integrado lo implementamos?

10 ¿Por qué no usamos el PICAXE?

Comida de robots

Hemos visto maravillas en el aumento de la potencia de los procesadores, en la mejora de la calidad de los materiales y en la reducción de los costos de todos estos componentes. Pero lamentablemente hay un punto en el que aún no hemos hecho grandes progresos: la energía portátil. En este capítulo presentaremos los diferentes tipos de baterías para utilizar en nuestro robot.

LA FUENTE DE ENERGÍA

Volvamos a nuestros sueños de robots. Soñamos con un robot veloz, ágil, y que recorra el universo sin desgaste. Soñamos con un robot poderoso, con alta tracción que le permita subir los caminos más inhóspitos. Soñamos con un robot de mucha autonomía, que pueda dar varias vueltas al universo sin necesidad de pasar por el enchufe. Soñamos con la máquina de movimiento perpetuo, y no somos los primeros. Desde que construimos nuestros primeros mecanismos autónomos con desplazamiento, hemos buscado la piedra filosofal de la energía: carbón, petróleo, electricidad, biocombustibles, baterías, hidrógeno, energía nuclear, etcétera. Mucha agua ha pasado debajo del puente y sin embargo, la ecuación que planteamos al comienzo es irresoluble: si aumentamos la **autonomía**, necesitamos **mayor tamaño y peso**, y por lo tanto, **mayor consumo**. Si queremos mayor **potencia**, también consumimos las cargas más rápidamente. Por ahora, nos tendremos que arreglar con lo que tenemos a nuestro alcance para darle energía a nuestro robot, y esto es lo que presentaremos en este capítulo.

Figura 1. *Aibo, el perro robot de Sony, con un juguete que podría ser su comida simbólica.*

Características de las celdas de las baterías

En el caso de los robots pequeños, la obtención de energía por sistemas de combustión o la energía nuclear constituyen un sistema un poco desproporcionado. Es por eso que nos inclinamos por las baterías compuestas por una o más celdas, con determinadas características que las hacen convenientes o no según el proyecto en el que trabajemos. Por esa razón, antes de analizar los tipos de baterías, haremos una breve presentación de los conceptos fundamentales que las distinguen.

Voltaje

Todas las celdas que presentaremos utilizan procesos químicos para la liberación de energía. El tipo de material utilizado para esa reacción es el que determinará el voltaje producido. Pero mientras se descarga, el voltaje puede variar. Es por esto que aparece el concepto de **voltaje nominal**, que indica el valor más estable en el que permanecerá una batería cuando se descargue. Puede ocurrir que apenas empiece a descargarse la batería, su voltaje nominal comience a bajar, o que se mantenga durante un tiempo para luego descender más bruscamente sus valores. Consideraremos que la batería está descargada cuando el voltaje disminuye alrededor del 20% de su valor nominal.

Por ejemplo, las baterías de autos tienen seis celdas de plomo ácido, cada una con un voltaje nominal de 2 voltios, y así nos proveen de 12 voltios nominales finales.

De todas formas, es posible que nuestro robot pueda funcionar con un voltaje menor al 80% del valor nominal. Esto dependerá esencialmente de los motores que utilicemos.

Capacidad

En general, cuanto más grande es la batería, mayor capacidad posee (es más pesada, pero nada es perfecto). Se mide en **Ampere/hora** (A/h) o **miliampere/hora** (mA/h) y se representa con la letra **C**. Por ejemplo, si en la batería aparece **C=250**, esto nos indica que puede suministrar 250 mA durante una hora, ó 125

III VITRIFICACIÓN DE PILAS

Una propuesta para el procesamiento de las pilas desechadas es lo que se conoce como vitrificación. Una vez eliminados, mediante calor, todos sus elementos combustibles, se separan los metales y los electrodos internos, y los químicos tóxicos se convierten en óxido, en forma de polvo. Este polvo se mezcla con vidrio, lo que permite un desecho que, de 1000 a 2000 años, comenzará a incorporarse en forma gradual a la biósfera.

***Figura 2**. Una muestra de todos los tamaños clásicos de las **baterías recargables**.*

mA durante dos horas, etcétera, antes de que caiga su voltaje.

Densidad de energía

Éste es uno de los puntos más importantes de análisis en los robots móviles autónomos. Con la densidad de energía representamos la **relación de la capacidad por unidad de volumen**. En nuestro caso, necesitamos la mayor relación posible, para lograr mayor autonomía y potencia. Pero seguramente, aparecerá algún otro problema, como los costos.

Curva de descarga

Es la curva que nos indica el voltaje de la batería en relación con el **tiempo de consumo**. En general, si la curva es plana, la batería baja su voltaje de golpe, lo que en general es bueno (mantenemos la performance de los robots pareja durante largo tiempo), pero peligroso en los casos de los robots acuáticos. Cuando la curva tiene un descenso parejo, muchas veces nos encontramos rápidamente con dificultades para determinar un comportamiento consistente en el robot. En ese caso, nos es imprescindible modificar el accionar de nuestro robot mediante la lectura de sensores, y no debemos confiar en los valores absolutos.

Resistencia interna

Todo generador de energía eléctrica también se comporta como un **conductor** y, por lo tanto, ofrece cierta resistencia al pasaje de corriente. A esto se lo llama **resistencia interna**. Por ejemplo, es una gran ventaja que las baterías de auto tengan baja resistencia, dado que necesitamos mucha corriente de golpe para encender el motor. En nuestro caso, las baterías con baja resistencia interna pueden ser pe-

ligrosas porque ante un corto entregan mucha corriente, aumentan el calor de la batería y pueden ocasionarles problemas a nuestros circuitos y, en casos extremos, una explosión. A medida que las celdas se desgastan, la resistencia interna crece, y por lo tanto, la tensión disminuye.

Capacidad de recarga

Otra característica fundamental de nuestras baterías es si son recargables o no, y cuántas veces se pueden recargar. Dado que las baterías son muy contaminantes, esta característica no sólo es una cuestión de costos, sino que también nos propone una actitud ecológica que consideraremos más adelante. Lamentablemente, las baterías con mayor densidad de energía no son recargables.

Efecto memoria

Es un término vinculado específicamente a las baterías de **Níquel cadmio**. Cuando una batería de este tipo no se descarga en forma completa antes de ser recargada, tiende a recordar este nivel de descarga como el nivel 0. Por lo tanto, la próxima vez considerará que está descargada cuando llegue a este punto. Aunque no hay una postura definida sobre este concepto, siempre se aconseja que, antes de recargar estas baterías, se las descargue por completo.

Costo

Y por último, nos encontramos con el factor común de todos los análisis: el costo. La relación entre la calidad de las baterías y el precio es directamente proporcional. Por suerte, la aparición de las cámaras digitales, los reproductores portátiles, los celulares y otros dispositivos que consumen baterías a granel, ha masificado la producción de éstas, y hoy encontramos en el mercado una gran variedad de calidad y precio para nuestras necesidades.

Antes de empezar a realizar una comparación con respecto a estos conceptos presentados, vamos a describir los tipos de baterías que podemos utilizar en nuestros robots.

Tipos de baterías

A continuación haremos un recorrido por los diversos tipos de baterías que

LOS CARGADORES TAMBIÉN SE FALSIFICAN

Como si no tuviéramos suficiente con la falsificación de baterías, también debemos tener cuidado con los cargadores, que no siempre son originales. En general, son muy parecidos y la diferencia está dada por el embalaje. Para protegernos, podemos consultar guías de compras que se brindan en los sitios de compra por Internet, o dirigirnos a los comercios de prestigio de nuestra zona.

tenemos para poder alimentar un robot, y veremos sus ventajas y desventajas. Por último, definiremos un conjunto de elecciones posibles para nuestro modelo en particular.

Batería de Iones de Litio (Li-Ion)

Estas baterías aparecieron recientemente y se utilizan mucho en los celulares (**Figura 3**). Poseen una elevada densidad de energía y, en general, se presentan en placas rectangulares de menos de 5mm de espesor. Tienen un voltaje de 3,7 V, y carecen de efecto memoria. La tasa de autodescarga es baja, con lo cual podemos sacarla del robot y guardarla aparte, y así se pierde sólo un 6% de su carga cada mes. La curva de descarga es lineal, con las ventajas y desventajas que presentamos antes. Con respecto a su vida útil, independientemente del uso que hagamos de ella, su duración es de tres años.

A pesar de todas las recomendaciones que nos dan al comprar celulares, no es necesario cargarla durante 12 horas en su primera carga. De hecho, las baterías más nuevas no necesitan más que 8 horas para cargarse completamente. Tampoco es real que los primeros ciclos de carga y descarga deben ser completos. Descargar en forma completa antes de cargarla puede ocasionar un mecanismo de bloqueo en sus circuitos. Siempre es preferible cargar la batería cuando la capacidad sea parcial.

A pesar de que aún son caras, el uso masivo en los celulares ha permitido que estén a nuestro alcance para alimentar a nuestro robot.

***Figura 3**. En esta imagen podemos ver baterías de Iones de Litio.*

Baterías de Níquel Cadmio (Ni-Cd)

Fueron las primeras baterías recargables que usamos en nuestros dispositivos (**Figura 4**). Su uso ha disminuido a favor de las baterías de **Níquel Metal** (**Ni/Mh**), principalmente por su efecto memoria y por la contaminación que produce el cadmio.
Una ventaja que poseen es la cantidad de ciclos de carga, que ronda los 1.500 ciclos. Otra característica positiva es el costo, que ronda en la cuarta parte de las de Ni/Mh, aunque esta diferencia se reduce día a día.

***Figura 4**. Batería de **Niquel Cadmio** para luces de emergencia.*

***Figura 5**. Pequeñas baterías recargables de Ni/Mh y Ni-Cd.*

Baterías de Níquel Metal (Ni/Mh)

Son las baterías de moda en este momento (**Figura 6**). Tienen un **40% más de capacidad que las de Ni-Cd**. Además, poseen baja resistencia interna, lo que permite una carga más rápida y una baja tasa de autodescarga. El problema es que luego de los 300 ciclos, su capacidad cae drásticamente, y luego de los 600 la resistencia interna aumenta en forma tan abrupta que se considera que el límite de los ciclos de carga es cercano a 500. Sin embargo, al tener mayor capacidad y no sufrir de efecto memo-

***Figura 6**. Baterías AA de **Níquel Metal** de 2500 mAh.*

III BATERÍAS DE POLÍMERO DE LITIO

Las baterías de polímero de litio son muy parecidas a las de litio que ya describimos, pero se pueden producir en láminas de un milímetro de espesor y poseen un ciclo de carga y descarga más prolongado, por lo que se obtiene una batería más pequeña, más liviana y de mayor capacidad.

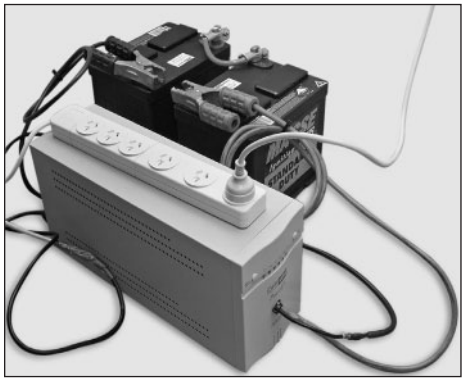


*Figura 7. Una UPS alimentada por una batería de **plomo-ácido**.*

Figura 8. Batería recargable de gel de 12 V.

ria, la cantidad de ciclos necesaria es menor. Otra ventaja es que son más ecológicas que las de Ni-Cd.

Baterías de Plomo–ácido

Son las baterías que podemos encontrar en nuestros vehículos (**Figura 7**). Están compuestas por celdas de 2 V, en general agrupadas en múltiplos de 3 (lo que brinda múltiplos de 6 V). Tienen un depósito de ácido sulfúrico donde se insertan unas placas de plomo. Cuando las celdas están cargadas, los electrodos son de plomo metálico y óxido de plomo, sumergidos en el ácido sulfúrico (diluido en agua).

En el proceso de liberación de energía, ambos electrodos se convierten en sulfato de plomo y disminuyen así la presencia del ácido en el agua. En la carga, en el polo negativo el sulfato de plomo vuelve a convertirse en plomo metálico, mientras que en el positivo se forma óxido de plomo, y aumenta la presencia de ácido sulfúrico en el electrolito. Tienen una gran capacidad de corriente con una baja resistencia interna, lo que las hace especialmente útiles para los motores de arranque. Son económicas pero de gran peso, lo que no las hace recomendables para los robots autónomos de tamaño mediano o pequeño.

EN EUROPA NO SE CONSIGUE

En Latinoamérica también tenemos desarrollo e investigación relacionados con las celdas de combustible. Lo curioso es que uno de los grupos dedicados a este tema surge de un colegio de nivel medio. Allí desarrollaron una **celda electroquímica circular** para generar hidrógeno y oxígeno que alimentan una celda combustible, que será uno de sus próximos proyectos. Más información en **http://electrolizadoro7.110mb.com**.

Baterías de gel

Esencialmente, son una modificación de las baterías de plomo-ácido, donde se le añade un agente gelificante al electrolito para disminuir su movimiento dentro de la carcasa (**Figura 8**). Además, presentan un mecanismo por el cual los gases internos no se liberan al exterior, sino que se recombinan con el agua de la batería. De esta manera, están selladas y pueden estar orientadas en cualquier dirección. Para su recarga, se recomienda utilizar un cargador de baterías especial de cuatro etapas, dado que en este proceso puede desprender gases o reducir su vida útil. Tienen una autodescarga muy baja y su carga útil dura hasta casi medio año.

Celdas de combustible

Es un dispositivo electroquímico similar a una batería, pero se reabastece a partir de una fuente externa de combustible y oxígeno. Por otra parte, los electrodos no se modifican como en una batería convencional, sino que se mantienen estables. En el caso clásico de una celda de hidrógeno, los reactivos que se utilizan son **hidrógeno** en el lado del ánodo y **oxígeno** en el cátodo. Al generar electricidad, el desecho resultante es agua y calor, con lo cual la contaminación producida por estas celdas es mínima. Aunque aún son muy costosas, en los últimos cinco años han bajado 25 veces su precio. Quedan como una buena propuesta para seguir atentamente en el futuro.
En la **Tabla 1** presentamos una tabla comparativa entre las baterías que podemos encontrar en el mercado. En ella, podemos ver todas sus características en forma más sencilla.

Nuestra elección

Como vemos, tenemos muchas variantes al momento de elegir la fuente de energía de nuestros robots. Para hacer nuestra elección, tuvimos en mente los siguientes aspectos:

- Costo.
- Capacidad de recarga.
- Masividad de uso (es decir, que puedan usarse en otros dispositivos).

TIPO	VOLTAJE POR CELDA	DENSIDAD DE ENERGÍA	RESISTENCIA INTERNA	CAPACIDAD DE RECARGA	COSTO
Pila carbón zinc	1,5 V	Baja	Alta	No	Muy bajo
Pila alcalina	1,5V	Alta	Alta	No	Moderado
Li-Ion	3,7V	Alta	Media	Sí	Alto
Ni-Cd	1,2V	Moderada	Baja	Sí	Moderado
Ni/Mh	1,2V	Alta	Baja	Sí	Moderado
Plomo ácido	2,0V	Moderada	Muy baja	Sí	Bajo

***Tabla 1.** Tabla comparativa entre las baterías más comunes.*

- Densidad de energía.
- Peso.

Antes de presentar nuestra elección, queremos aclarar que cualquier fuente de corriente continua entre 9 V y 12 V es perfectamente utilizable en nuestro diseño. Por lo tanto, no es necesario comprar lo mismo que mencionamos aquí, ya que podemos reutilizar baterías que tengamos en nuestro hogar. Por otro lado, en los momentos en los que realizamos pruebas con nuestra computadora, podemos conectar una fuente de corriente continua de 12 V para no consumir la energía de nuestras baterías.

En nuestro caso, hemos elegido las **pilas AA de Ni/Mh de 2.500 mA/H**, porque son las que presentan la mejor combinación de las características que hemos usado para nuestro análisis. Dado que proporcionan 1,25 V, usaremos un **pack de 8 a 12 pilas** para alimentar a nuestro robot. Para ello, podemos adquirir 2 ó 3 **portapilas** de 4 pilas cada uno. En el capítulo en el que hagamos el montaje, vamos a ver cómo los ubicamos, pero antes de eso, los conectaremos con nuestro controlador para comenzar las primeras pruebas.

***Figura 9**. Portapilas para 8 pilas AA.*

Calidad de las baterías

Un factor fundamental en el éxito de nuestra fuente de energía es su calidad. Es tan masivo el uso de las pilas recargables, que podemos encontrar ofertas muy diversas a la hora de comprarlas. En un documento realizado para su tienda de Ebay, Javier Martínez nos da algunas recomendaciones muy interesantes que presentamos a continuación.

En general, las pilas más económicas no brindan información real con respecto a su capacidad. La manera de comprobar esto es utilizar algún cargador con medidor de capacidad, pero en general son muy costosos, además de que no pueden realizar la medición en forma instantánea. Una segunda opción, inmediata y económica, es **¡pesar las pilas!** Efectivamente, dado que la capacidad está relacionada con los componentes internos de la pila, una pila con mayor capacidad debe ser más pesada. En la próxima página vemos la **Tabla 2**, creada por Javier, con testeos que ha realizado con diversas pilas de Ni/Mh. En todos los casos, las pilas medidas son AA. Lamentablemente, otro problema que tenemos es

que también se comercializan pilas de marcas conocidas que, en realidad, son **imitaciones**. Para detectarlas, la variable peso también es útil. Por otro lado, es importante mencionar que las marcas Sony y Panasonic no han producido pilas de más de 2700 mAh. Podemos consultar varias páginas web que nos alertan, mes a mes, sobre las falsificaciones que aparecen en el mercado (**Figura 10**).

CAPACIDAD EN MAH	PESO EN GRAMOS
1700 - 1800	27
1800 - 2000	27,5
2000 - 2200	28
2200 - 2300	29
2300 - 2600	30
2600 - 2800	31
2950	33

***Tabla 2**. Relación entre el **peso** y la **capacidad** de las pilas AA.*

Cargadores

El cargador de baterías es un dispositivo que permite **recargar baterías** aptas para tal fin, y fuerza el ingreso de corriente eléctrica dentro de ellas. El cargador más sencillo (**Figura 11**) es aquel que introduce una corriente continua en la batería que debe ser cargada sin modificar la corriente según el tiempo o la carga producida hasta el momento. Es económico, pero el resultado deja bastante que desear. En general, son de carga lenta para evitar

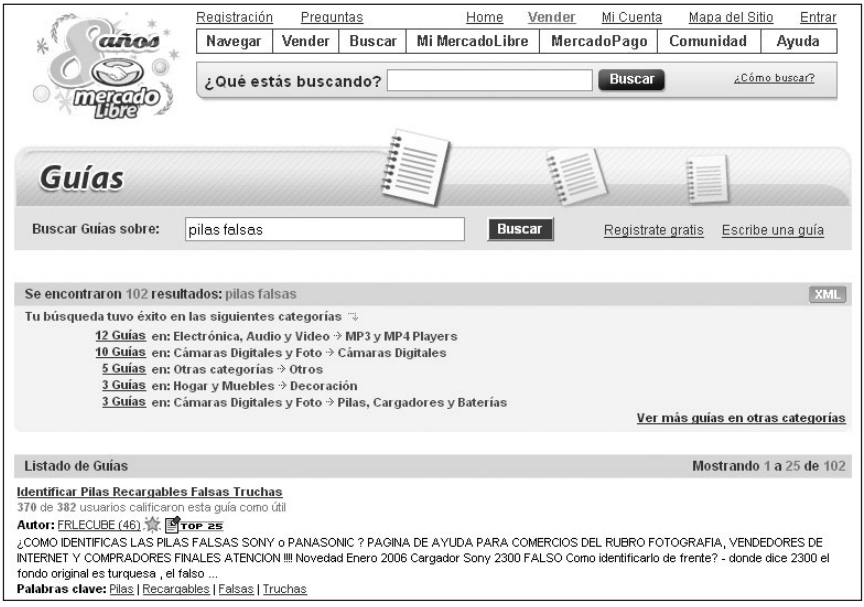


***Figura 10**. En el sitio de MercadoLibre podemos encontrar guías especiales para detectar falsificaciones.*

***Figura 11.** Cargador de baterías tradicional.*

***Figura 12.** Cargador inteligente para baterías de diferentes capacidades.*

el deterioro de las baterías. De todas formas, si las dejamos mayor tiempo que el indicado, podemos debilitarlas o deteriorarlas por sobrecarga.

Un modelo algo más complejo es aquel que detiene su carga luego de un tiempo predefinido. Se utilizaron mucho con las baterías de Ni-Cd a fines de la década del 90. Frecuentemente, se comercializa el pack de baterías junto al cargador, que viene configurado con el tiempo acorde para las baterías asociadas. Si utilizamos otro tipo de baterías, corremos riesgos de sobrecarga si son de menor capacidad o de carga parcial en el caso contrario.

También podemos contar con **cargadores inteligentes** (**Figura 12**), donde el dispositivo monitorea en forma constante el voltaje de la batería, su temperatura o el tiempo de carga, y determina el momento de corte óptimo para cada caso en particular. Para las baterías de Ni-Cd y Ni/Mh, el voltaje sobre la batería crece en forma lenta durante el proceso de carga, y decrece cuando están llenas. Esto le indica al cargador que han llegado a su punto máximo de carga. Sin embargo, monitorear la temperatura nos asegura no tener ningún tipo de problemas con las baterías que se encuentren en corto. En estos cargadores, es ideal que las baterías no se ubiquen en serie, sino de forma independiente, para que el monitoreo sea específico en cada unidad.

Los **cargadores rápidos** de buena cali-

dad, específicos para un modelo de batería, usan circuitos de control interno de las baterías que permiten acelerar el proceso de carga. Muchos de ellos incluyen un sistema de ventilación para impedir el sobrecalentamiento. De todas formas, cuando se utilizan con baterías de otras marcas, realizan una carga normal. Si el cargador rápido no utiliza este sistema de control interno, puede dañar las baterías. Existen algunos cargadores especiales que pueden realizar esta carga rápida sobre cualquier batería, pero son de marcas reconocidas y su precio es elevado.

Como novedad, han comenzado a aparecer cargadores que utilizan el puerto USB de la computadora para evitar la necesidad de otro enchufe (**Figura 14**). Incluso hay pilas que tienen un adaptador USB para conectarlas directamente a nuestra PC. Si pertenecemos a la clase de fanáticos que tiene un hub USB de muchas bocas, tal vez nos quede algún lugar para conectar estos nuevos juguetes.

***Figura 13**. Cargador para baterías de 9 V.*

***Figura 14**. Cargadores de baterías por USB.*

Ayudar a que no sólo las baterías duren más

Todos los dispositivos que hemos mencionado, en menor o mayor medida, son contaminantes. Es por eso que queremos ofrecer algunos consejos con respecto a la forma de desechar las baterías para que su poder contaminante se reduzca. Antes que nada, en los momentos de testeo, podemos usar nuestro robot conectado a la corriente eléctrica. Así cuidaremos nuestro bolsillo y nuestro planeta. De esta forma, los ciclos de carga de las baterías se realizarán específicamente cuando sean necesarios. Por otra parte, el uso de baterías recargables (en lugar de alcalinas, de carbón, etcétera) ya es un

***Figura 15**. Batería flexible, una propuesta ecológica para la generación de energía.*

punto de partida muy interesante para evitar este tipo de contaminación. La solución ideal para el desecho de las baterías parte de un trabajo en conjunto de la sociedad, que debe apartarlas de la basura común, y del Estado, que debe proveer plantas de reciclado donde podamos enviar lo que hemos apartado. El proceso de reciclado no es directamente rentable, por lo que es necesario que las empresas que realicen este proceso pertenezcan al Estado o estén subvencionadas por él. Hasta

LO BARATO SALE CARO

El INTI (Instituto Nacional de Tecnología Industrial, de Argentina) realizó un estudio comparativo de pilas alcalinas y de carbón, que incluía tanto las marcas reconocidas como las de venta callejera. Las conclusiones son que, aunque las de venta callejera sean más baratas, la duración es mucho menor, por lo que no son convenientes económicamente. Además, las de carbón drenan su contenido al exterior y son mucho más tóxicas que las de mayor valor.

que no se tome conciencia de este peligro, los consejos que dan las ONGs vinculadas a la ecología son:

- No juntar pilas porque así se concentran los riesgos.
- No tirar las pilas a la cloaca porque pueden contaminar el agua.
- No quemar pilas ni baterías de ningún tipo.
- No dejarlas en lugares a los que pueden acceder los niños.
- Reemplazar, en la medida de lo posible, el uso de pilas por corriente eléctrica.

... RESUMEN

Hay un conjunto amplio de variantes para alimentar a nuestro robot. Desde las tradicionales baterías de plomo, demasiado pesadas para nuestros pequeños móviles, hasta las nuevas celdas de combustible que no sólo aportan más capacidad, sino que colaboran con un mundo más limpio. En particular, hemos optado por el tipo de baterías que consideramos de uso más popular y más económico: las baterías de Ni/Mh. Aunque tengamos que usar entre 8 y 10 baterías, el peso y la capacidad que tienen serán de gran utilidad para mover a nuestro robot. Y además, probablemente no tengamos que comprar un cargador, porque las cargaremos con lo que ya tenemos en nuestra casa. Por último, no debemos olvidar el impacto de todo tipo de baterías en nuestra biósfera. Es por eso que recomendamos el uso de una fuente para alimentar a nuestro robot, y sólo en aquellos momentos en los que necesitemos independencia de cables, utilizar las baterías.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿A qué llamamos capacidad de una batería?

2 ¿Qué es la densidad de energía?

3 ¿A qué se llama resistencia interna de una batería?

4 ¿En qué tipo de pilas nos encontramos con el famoso efecto memoria?

5 Realice una comparación entre las baterías de Li-Ion, las de Ni-Cd y las de Ni/Mh.

6 ¿Cuáles son las desventajas que presentan las baterías de Plomo–ácido?

7 ¿Qué son las celdas de combustible?

8 ¿Qué tipo de baterías elegimos para nuestro proyecto y por qué?

9 ¿Qué factores podemos analizar para determinar la autenticidad de una batería?

10 ¿Cuáles son los factores que analizan los cargadores inteligentes?

Hablar con nuestro robot

Cuando ya desarrollamos el controlador y tomamos la decisión de cómo lo vamos a alimentar, comienza el desafío de programar el controlador. Los lenguajes no son sencillos y necesitamos procesos lógicos para poder expresar, en el lenguaje del robot, lo que deseamos que realice. En este capítulo presentaremos los diferentes lenguajes y elegiremos uno para explicar lo que necesitamos al programar el comportamiento del robot.

COMUNICACIÓN CON EL ROBOT

Ya hemos logrado construir el cerebro de nuestro robot y lo hemos dotado de energía. Sólo nos falta encenderlo para darle vida a nuestra criatura. Presionamos un botón y en un primer momento, nada ocurre. Pero de pronto, algunos músculos tiemblan y sus ojos se abren. ¡Hemos logrado el milagro! Sólo basta con solicitar lo que deseamos que haga y nuestros sueños se cumplirán. Entonces le pedimos que nos traiga el periódico, que nos prepare una buena cena y que luego deje la cocina en orden. Nada ocurre. Sigue allí, con sus lucecitas parpadeantes, sin demostrar buena voluntad para obedecer a nuestras órdenes. ¿No tiene incorporadas las tres leyes de la robótica? Tal vez fue demasiado complejo el pedido. Intentamos con cosas más sencillas: que nos traiga el control remoto, que atienda la puerta, que saque a pasear al perro, que resuelva esas integrales dobles que nos fastidian, y nuestro robot sigue allí, sin obedecer. ¿No será que no entiende de castellano? Probamos con inglés, alemán, francés, aymará y, sin embargo, no obtenemos respuesta. Es hora de que nos detengamos a tratar de comprender a nuestro robot. Es hora de que nos demos cuenta de que aún es pequeño, y que sólo entenderá instrucciones muy sencillas, instrucciones de muy bajo nivel.

Figura 1. *Una* ***Sinclair Spectrum*** *que, con 48 Kb de RAM, desafiaba nuestra capacidad de programar con poca memoria y baja velocidad.*

Lenguajes de programación para robots

Para aquellos que hemos trabajado con computadoras de baja capacidad de procesamiento y poca cantidad de memoria, programar los micros de los robots nos resulta familiar, dado que volvemos a tener las limitaciones de aquellos viejos tiempos.

Con la aparición de las PCs, la velocidad de procesamiento y la memoria de las que disponemos nos permiten utilizar lenguajes que no hagan un aprovechamiento preciso de todos los recursos. Gracias a estas ventajas, aparecen los lenguajes de **alto nivel**, que nos brindan un mayor poder de expresividad a cambio de gastar mucha memoria y procesamiento.

En el caso de los robots, carecemos de estos dos recursos. Por eso, necesitamos lenguajes que aprovechen al máximo cada byte de memoria y cada ciclo de procesador para realizar nuestra tarea. De esta forma, los **micros** utilizan habitualmente **lenguajes de bajo nivel**. Estos lenguajes son pobres en capacidad expresiva, pero muy potentes a la hora de utilizar cada componente de nuestro controlador. El problema es que aprender a utilizarlos es complicado. Pero no temamos, vamos a hacer una recorrida por los lenguajes más comunes, desde los más potentes hasta los más sencillos de utilizar, y nos quedaremos, para nuestras primeras experiencias, con uno de esta última categoría: **MikroBasic**. Más adelante, cuando ya nos quede claro el funcionamiento de nuestro micro, podremos aventurarnos a horizontes más complejos.

ALTO Y BAJO NIVEL

Nuestro robot no es más que una computadora, y tenemos que darle un conjunto de instrucciones para que realice una tarea. Los lenguajes de alto nivel nos permiten describir la tarea en un nivel de abstracción cercano a nuestro proceso humano de razonamiento, pero son más lentos y menos poderosos porque no tienen acceso a instrucciones tan detalladas y precisas. Los lenguajes de bajo nivel son complejos de entender y utilizar porque nos obligan a expresar lo que queremos que realice el robot en instrucciones muy cercanas a procesos electrónicos y mecánicos, pero nos brindan un poder mucho mayor por el nivel de detalle que podemos lograr.

PicBasic Pro

Uno de los lenguajes más populares para los micros PIC es el **PicBasic Pro (PBP)**, de **microEngineering Labs**. Como su nombre lo indica, es un compilador **Basic** que posibilita la programación en un lenguaje mucho más sencillo que el ensamblador de MicroChip. Es similar al Basic del Ba-

sic Stamp y posee librerías similares a éste. Produce código para una gran variedad de PIC de 8 a 68 pines.

Para crear un programa a ser compilado por PBP, utilizamos nuestro editor de texto plano preferido, y grabamos el archivo con extensión **.BAS**. Luego llevamos ese archivo donde está el compilador **pbp.exe** y allí ejecutamos **Pbp nombreDelArchivo**.

Si el código es sintácticamente correcto, se creará un código intermedio de extensión **.ASM**, que invocará al ensamblador para generar, por último, un archivo **.HEX**. En este archivo está el código del micro, y lo bajaremos luego con el software programador. Veamos un ejemplo de código en PicBasic Pro:

```
while true
high PORTB.0 'enciende el puerto
     pause 500 'espera 500
               milisegundos
     low PORTB.0 'apaga el puerto
     pause 500 'espera otros 500
               milisegundos
wend
```

Si en el puerto tuviéramos conectado un led, tendríamos una bonita luz que se enciende y apaga indefinidamente. Los creadores de PBP han desarrollado una suite, **MicroCode Studio Plus** (**Figura 2**), que une todas las herramientas necesarias en una única IDE. El editor posee coloreado de sintaxis y ayuda sensible al contexto. Su explorador de código permite saltar con rapidez a secciones específicas, como declaración de **variables**, **constantes**, **símbolos** y **etiquetas** incluidos en el código fuente. Posee todas las funcionalidades convencionales de los editores de texto como cortar, copiar, pegar, búsquedas y reemplazos, entre otros. Además, la IDE nos permite realizar **In Circuit Debugger** (**ICD**, depuración en circuito), y resaltar cada línea de código cuando se ejecuta en el micro. Podemos definir puntos de corte y ejecución de código paso a paso, lo que facilita en gran medida la depuración de nuestro programa. MicroCode Studio Plus también nos brinda un cargador de código (**bootloader**) para incorporar a nuestro PIC, de manera que no nece-

ENCONTRAR LOS LENGUAJES PARA PROGRAMAR LOS PICS

Como casi todas las cosas de este universo, en la red podemos encontrar demos de los lenguajes e IDEs que estamos comentando. Aquí van los sitios:

- PicBasic Pro: **www.melabs.com/index.htm**.
- CCS C: **www.ccsinfo.com**.
- MikroBasic: **www.mikroe.com/en/**.

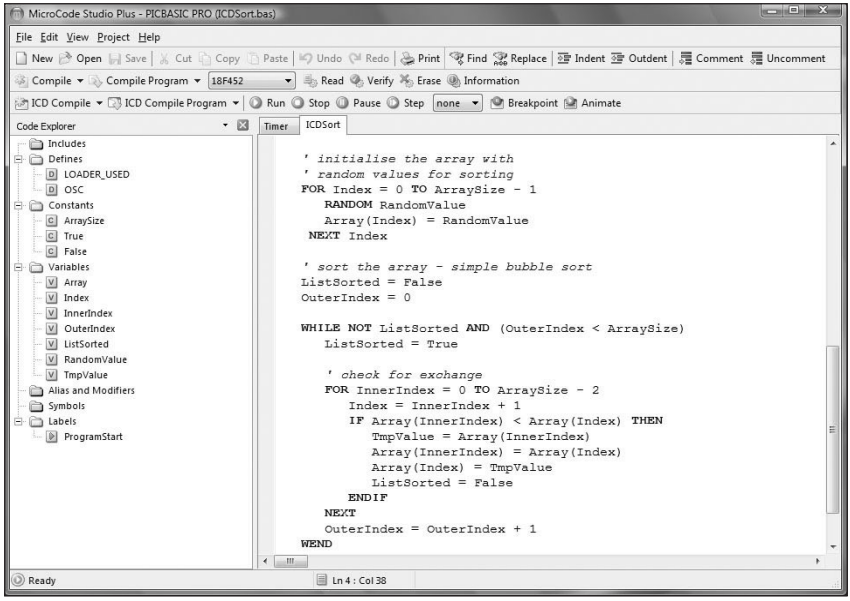


***Figura 2.** Vista del **MicroCode Studio Plus**, IDE para el PicBasic Pro.*

sitemos un programador para nuestro micro. Sólo lo necesitaremos la primera vez, para cargar este bootloader.

Compilador CCS C

Este compilador de **lenguaje C** ha sido diseñado exclusivamente para la línea PIC. Se considera el compilador más **optimizado** para estos micros. Tiene una librería de funciones predefinidas, lo que facilita la programación en un lenguaje de bajo nivel como el C. Sin estas funciones, sería muy engorroso programar el micro. Cuenta con controladores de reloj en tiempo real y conversores analógicos digitales. Por ejemplo, contamos con la instrucción **READ_ADC()**, que permite leer un valor del conversor mencionado. Podemos definir variables estructuradas cuyos componentes se correspondan uno a uno con registros del micro.

Es integrable con simuladores de PIC como el **MPLAB** y podemos mezclar código **assembler** en el medio de nuestro programa en C, y hacer referencia a las variables de éste. Para tener una idea de la sintaxis, desarrollaremos el mismo ejemplo que mostramos antes con PicBasic.

```
while (TRUE) {
output_high(PIN_B0); ‘enciende
     el puerto
delay_us(500); ‘espera 500
     milisegundos
```

```
output_low(PIN_B0); 'apaga
     el puerto
delay_us(500); 'espera otros
     500 milisegundos
} 'Todo esto se repite
     eternamente…
```

Cabe aclarar que las funciones que se presentan en este ejemplo desencadenan código mucho más optimizado que el generado por Basic.

La firma **CCS** también ofrece IDEs para Windows, llamadas **PCW y PCWH**. La única diferencia es que PCWH soporta la serie PIC 18. Poseen un editor de código fuente (C Aware Editor, **Figura 3**), que realiza coloreo de sintaxis, control de tabulaciones, provee ayuda sensible al contexto y pareo de llaves y paréntesis. Brinda un asistente (**Figura 4**) para nuevos proyectos donde, en pocos pasos, podemos generar la estructura básica del programa. También permite realizar depuración, pero requiere de hardware especial para su uso. En síntesis, para programación en bajo nivel, es la opción que presenta más herramientas para facilitar el desarrollo.

MikroBasic

Por último, llegamos al lenguaje elegido para nuestros ejemplos de proyectos. La empresa **MikroElectrónica** ha desarrollado un conjunto de compiladores para micros, tanto para lenguaje C como para Basic. Lo más interesante de ambos es que integran

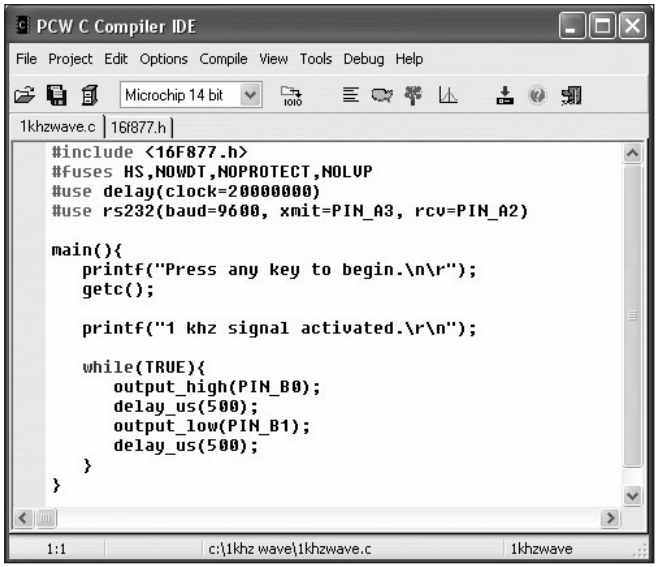


Figura 3. *Vista del editor* ***C Aware*** *de CCS.*

un IDE que facilita mucho la programación, ya que presenta las características que hemos mencionado, como coloreo de sintaxis, ayuda sensible al contexto, estadísticas sobre el uso del micro y muchas otras más. Por otra parte, los desarrolladores nos permiten bajar un **demo gratuito** completamente funcional, donde sólo tenemos un **límite de 2Kb** en el código fuente, lo que es suficiente para todos los proyectos que desarrollaremos en nuestro libro. Soporta una gran cantidad de modelos de PICs y proporciona librerías para **RS-232**, conexiones USB, interfaz para displays, etcétera. Dado que es el lenguaje y la IDE que vamos a utilizar, comencemos a estudiar en detalle cada una de sus herramientas.

Instalación de mikroBasic

Para instalar el programa en nuestra computadora, debemos dirigirnos a la dirección **www.mikroe.com/en/compilers/mikrobasic/pic/** y entrar en la sección **Download**, donde podremos bajar la última versión. Los pasos de instalación son muy sencillos, clásicos en cualquier programa de Windows (**Figura 5**).

En la misma sección del sitio podemos encontrar actualizaciones y parches, por lo que recomendamos visitar periódicamente el sitio. También podemos encontrar un conjunto de manuales, muy claros y didácticos, que nos permitirán completar nuestra breve mirada sobre la herramienta.

En nuestro caso, todos los ejemplos que mostraremos y las indicaciones

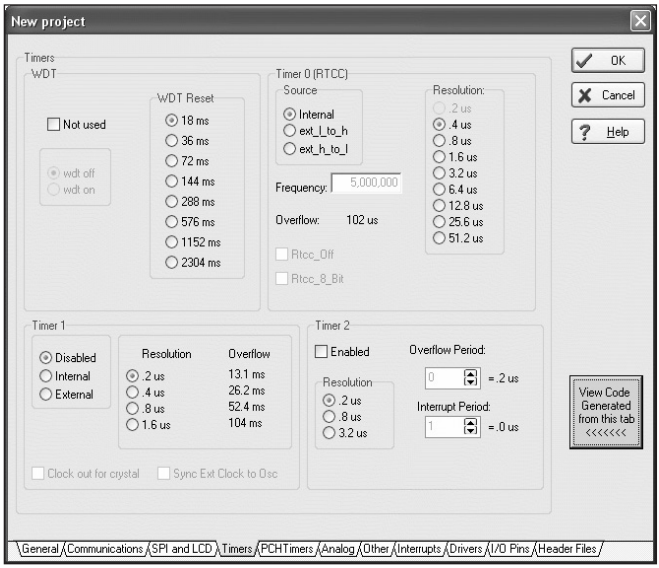


***Figura 4. Asistente** de proyectos del **PCW**.*

de operación del mikroBasic están basados en la versión 5.0.0.2, del 31 de octubre de 2006.

Editor de código fuente de mikroBasic

Uno de los puntos más fuertes de la IDE es el editor que nos proporciona. Como todo editor de texto, presenta las utilidades clásicas como cortar, copiar y pegar, y búsquedas y reemplazos masivos. Pero además, nos ofrece:

- Coloreo de sintaxis.
- Asistente de código.
- Asistente para parámetros de funciones.
- Plantillas prearmadas.
- Autocorrección de tipos.
- Bookmarks.

Éstas y otras opciones se pueden configurar desde **Tool/Preferences...** o al hacer clic sobre el icono **Tools** (**Figura 6**).

Para utilizar el asistente de código, una vez que ingresamos las primeras letras de lo que deseamos escribir, presionamos **CTRL + ESPACIO** y todos los identificadores válidos que comiencen con esas letras se desplegarán en un cuadro de lista. Allí podremos elegir lo que íbamos a escribir o, simplemente, continuar con el texto deseado si no lo encontramos (**Figura 7**). Es necesario realizar una primera compilación para que funcione el asistente.

También podemos activar un asistente para los parámetros de las funciones (**Figura 8**). Para ello, escribimos la llamada a la función y, cuando abrimos el paréntesis, automáti-

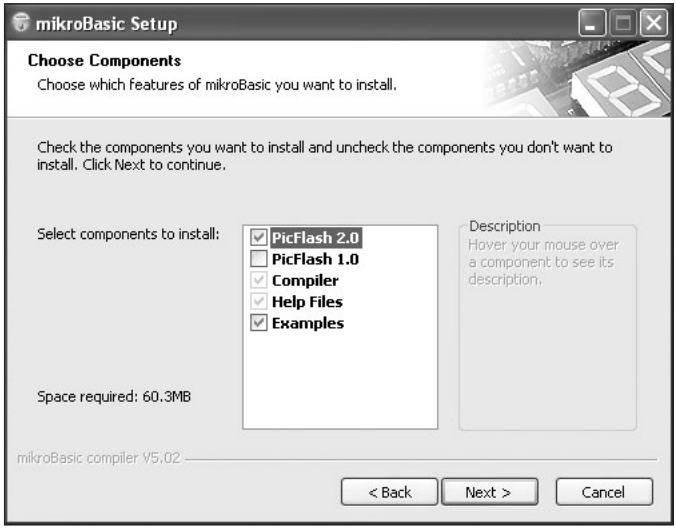


Figura 5. *Durante la instalación, es recomendable dejar todo como está en la* ***selección de componentes*** *para mikroBasic, ya que no ocupa demasiado espacio.*

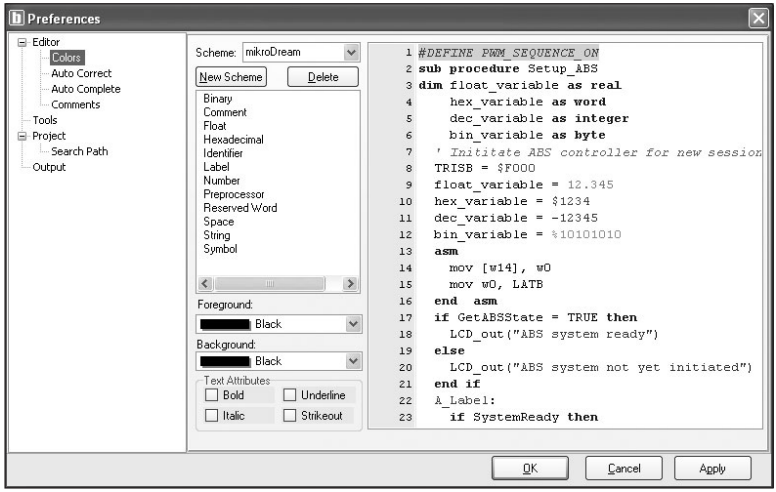


Figura 6. *Editor de preferencias de la IDE de mikroBasic.*

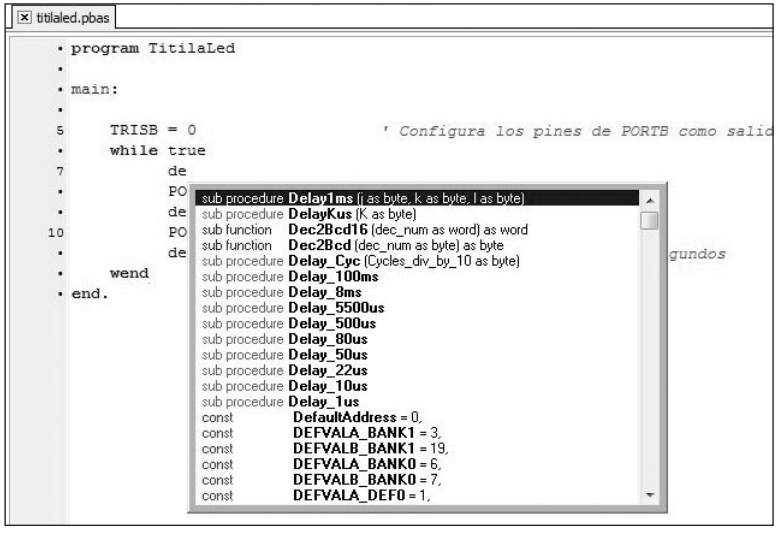


Figura 7. *Ejemplo de la lista presentada por el asistente de código cuando escribimos de y luego* ***CTRL + ESPACIO****.*

III CINTA ADHESIVA, PEGAMENTO, BANDAS ELÁSTICAS, ¡TODO VALE!

Muchas veces se puede salir de situaciones problemáticas gracias a estos materiales. Si no nos presentamos en concursos de belleza, cuando el tiempo nos corre y debemos presentarnos en el torneo, ¡todo lo que una fuertemente a nuestras piezas será válido!

camente aparecerá un pequeño cartel con el o los parámetros correspondientes, que nos indicará con **nombre:tipo** qué función cumple el parámetro y de qué tipo es.

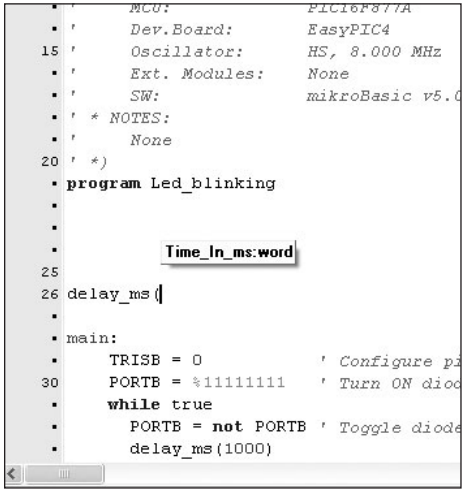


***Figura 8.** Asistente que informa los **parámetros** de las funciones.*

Para las estructuras más complejas, como las funciones, los procedimientos, los encabezados u otras, podemos utilizar las **plantillas** (**Figura 9**). Por ejemplo, si escribimos **func** y presionamos **CTRL + J**, automáticamente se escribirá la estructura completa de una función para que completemos con los datos necesarios. También tenemos la posibilidad de crear nuestras propias plantillas en la opción **Auto Complete** de **Tools/Preferences...**.

Por otro lado, cabe mencionar que además de las características que ya presentamos, tenemos herramientas para realizar autocorrección, comentar o descomentar un conjunto de líneas de código, insertar marcas y saltar de una a otra e ir a un número de línea de código.

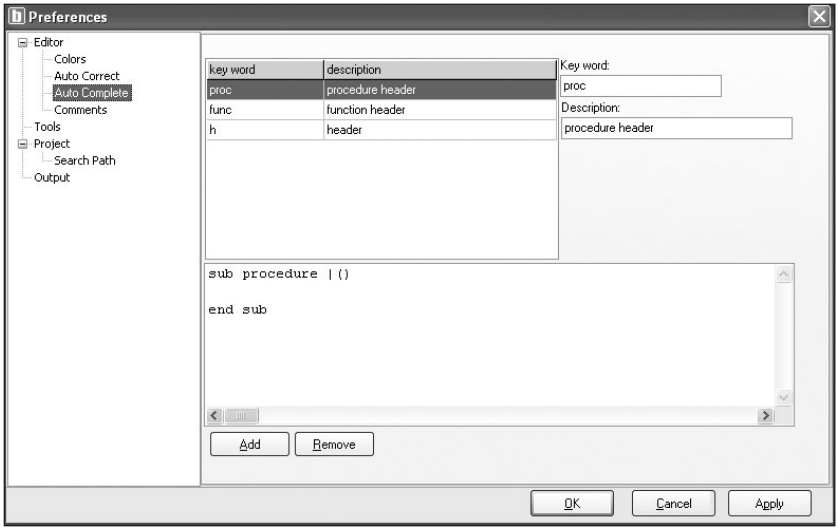


***Figura 9.** Editor de las **plantillas** de código.*

Explorador de código

El explorador de código (**Code Explorer**) se encuentra en la zona izquierda de la ventana. Nos proporciona una lista de los ítems declarados en el código. Desde allí podemos saltar a cualquiera de las declaraciones con sólo hacer doble clic sobre ella. En la parte superior del explorador podemos encontrar dos solapas más. La solapa **QHelp** nos proporciona ayuda sobre la librería de funciones. La última solapa (**Keyboard**) nos proporciona una lista de los atajos de teclado del editor.

Depurador

MikroBasic proporciona un depurador a nivel de código fuente (**Figura 11**). Esto nos permite simular el comportamiento del micro y nos asiste para la depuración de errores. Dado que es a nivel de código y no tiene lectura real del micro, no permite simular ciertas propiedades intrínsecas del procesador, como temporizadores, interrupciones, etcétera.

Una vez que hemos compilado nuestro programa (**Project/Build**), ejecutamos el depurador en **Run/Start Debugger**. Automáticamente nos marca la primera línea de código y nos aparece la ventana **Watch**, que nos permitirá realizar el control de la depuración. En esta ventana podemos visualizar variables y registros del PIC, actualizados según la ejecución del programa. En la parte superior de la ventana encontramos iconos para el control de la ejecución del depurador (arranque, pausa, detención) y mecanismos de ejecución del código paso a paso. Debajo de esta barra tenemos un cuadro de lista pa-

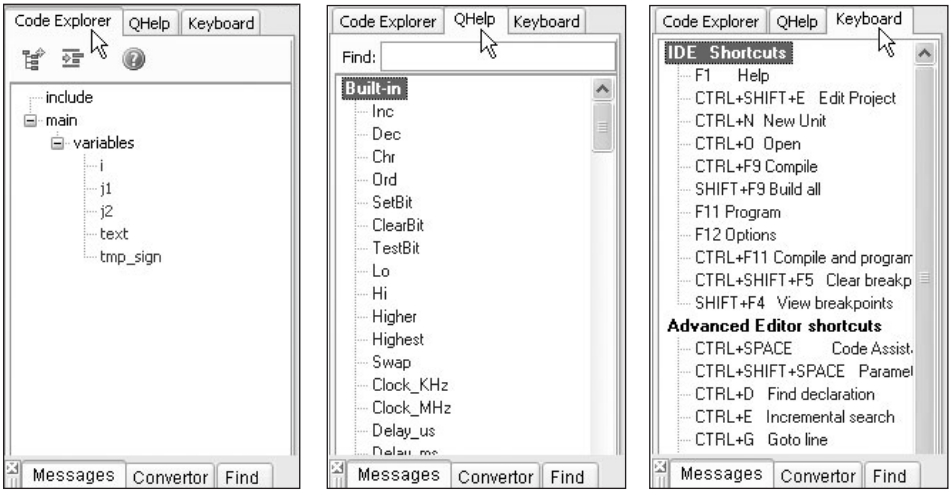


Figura 10. *Aquí podemos ver la parte de la ventana que contiene el explorador, la ayuda y los atajos de teclado.*

ra poder agregarle variables y registros al cuadro inferior, que nos permitirá seguir el comportamiento de ellos paso a paso. En ese cuadro, cuando agregamos una variable o un registro, podemos ver el nombre, el valor y la dirección de memoria que ocupa.

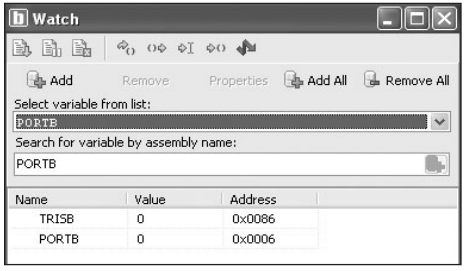


Figura 11. *Ventana del controlador de la depuración.*

Para poder probar las opciones del depurador, realicemos nuestro primer programa sencillo, idéntico al que hemos utilizado en cada ejemplo del compilador. Para comenzar, luego de abrir mikroBasic, cerremos el proyecto que tengamos abierto y vayamos al menú **Project/Close Project**. Creemos un proyecto nuevo en **Project/New Project...**, donde nos aparecerá la ventana que podemos ver en la **Figura 12**.

Completemos el nombre, el sendero y la descripción del proyecto y eligamos el procesador (utilicemos el **PIC 16F88**) y la velocidad de reloj (en nuestro caso, **4.0**). Las banderas que

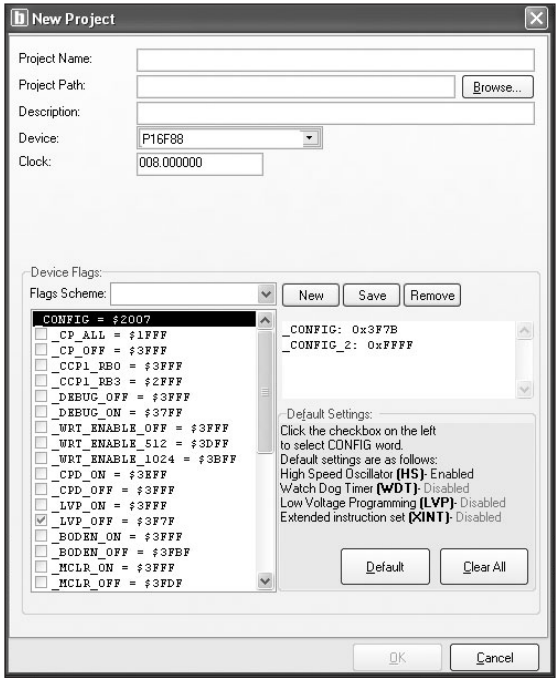


Figura 12. *Creación de un proyecto nuevo.*

aparecen en la parte inferior las dejaremos como están por defecto. Automáticamente nos aparecerá un fuente con el encabezado del programa. Debajo del encabezado escribiremos el siguiente código:

```
main:
     TRISB = 0        'Configura
      los pines de PORTB como salida
     while true
            PORTB.3 = 255 'Enciende
              los pines de PORTB
            delay_ms(500)' Espera
               500 milisegundos
            PORTB.3 = 0    'Apaga
               PORTB
            delay_ms(500)'
      Nuevamente espera 500
               milisegundos
     wend
end.
```

Si luego de realizar la compilación (**CTRL + F9**) ejecutamos el depurador (**F9**), podremos ver cómo señala la primera línea posterior a **main** y abre la ventana **Watch Window**. Allí podremos añadir las variables **TRISB** y **PORTB** para realizar el seguimiento de sus valores. Si ejecutamos paso a paso, nos indicará en qué línea nos encontramos y la resaltará con color, y veremos la modificación de los valores de las variables. Cuando lleguemos a **delay**, en la parte inferior de la ventana podremos ver cómo se simula el paso del tiempo. Si hacemos doble clic sobre una variable, podremos modificar su valor en el formato numérico que deseemos.

Otras ventanas de depuración que podemos encontrar si nos dirigimos a **View/Debug windows** son:

- **Stopwatch**: esta ventana nos muestra la cantidad de ciclos/tiempo utilizados hasta la ejecución del último comando (**Figura 13**). El valor **Delta** nos muestra el número de ciclos/tiempo entre la última línea ejecutada y la línea activa. Con el botón **Reset To Zero** podemos poner en **0** el valor de **Stopwatch**, pero continuaremos acumulando en el cuadro **Current Count**.
- **RAM**: nos muestra el contenido de la RAM e indica los últimos valores cambiados en color rojo.
- **History**: presenta las últimas N líneas de nuestro fuente que se han ejecutado, pero en código assembler.

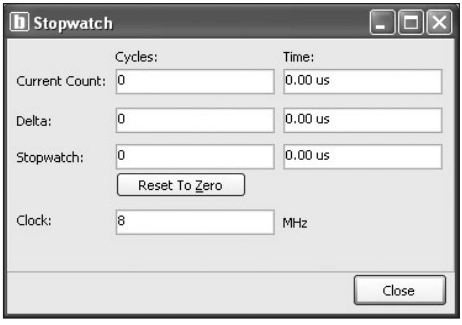


Figura 13. *Ventana del Stopwatch.*

Por último, cuando procedemos a compilar, si tenemos errores en nuestro código, aparecerán en la parte inferior de nuestra pantalla, en la ventana de **errores**.
Allí se nos indicará la línea del error, su mensaje y la unidad donde se produjo. Si hacemos doble clic sobre esta línea, iremos directamente a su posición en el código fuente.

Probablemente, todas estas características del depurador de mikroBasic lo convierten en la IDE más poderosa del mercado para la programación de PICs. Ahora que ya conocemos la herramienta, pasemos a crear nuestros propios programas.

Manos a la obra

Como ya hemos visto, mikroBasic organiza sus aplicaciones en proyectos. Estos **proyectos** están compuestos por un archivo .PBP descriptor del proyecto y uno o más archivos de código fuente con extensión .PBAS. Cabe aclarar que sólo podemos compilar un fuente si éste forma parte de un proyecto.
Una vez que tenemos creado el proyecto, como vimos en el punto anterior, podemos editarlo si vamos a **Project/Edit project**, donde nos aparecerá la misma ventana que usamos para crearlo. Cuando compilamos nuestro proyecto, se nos generan los siguientes archivos:

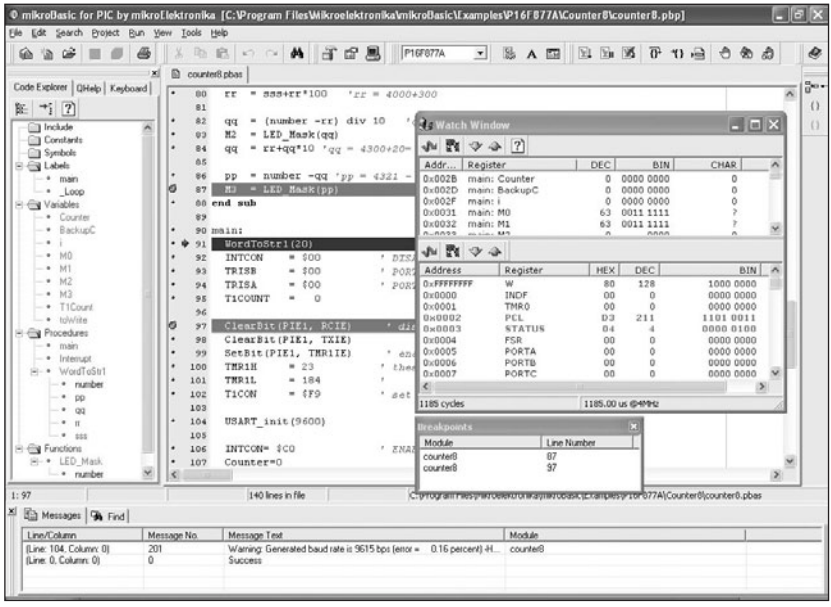


Figura 14. *Una mirada sobre todas las* ***herramientas*** *que presenta la IDE de* ***mikroBasic.***

- **hex**: archivo estilo Intel. Con él programaremos el micro.
- **mcl**: archivo binario de distribución. Puede agregarse a otros proyectos sin entregar el fuente.
- **asm**: archivo en código assembler (**Figura 15**), pero con un conjunto de comentarios que lo hace muy legible y más fácil de entender. Por ejemplo, para cada instrucción del fuente en Basic aparece la lista de comandos en assembler. También podemos verlo una vez que se haya generado desde el menú **View/ View Assembly**. Para aprender, es recomendable que miremos el .ASM generado por el ejemplo que utilizamos para el depurador.

WINPIC800

Para poder bajar nuestros programas al programador y de allí a nuestro controlador, debemos usar el WinPic800, que podemos conseguir en forma gratuita en el sitio **www.winpic800.com**. Está todo en castellano y su forma de uso es muy sencilla. De todas maneras, si se presenta alguna duda, es recomendable consultar la ayuda y los foros que hay en el sitio.

De nuestra parte, una única recomendación que no encontramos en la documentación: en la solapa **Config**, probar con las diferentes opciones de **Osc**, y dejar activo sólo **WDT**.

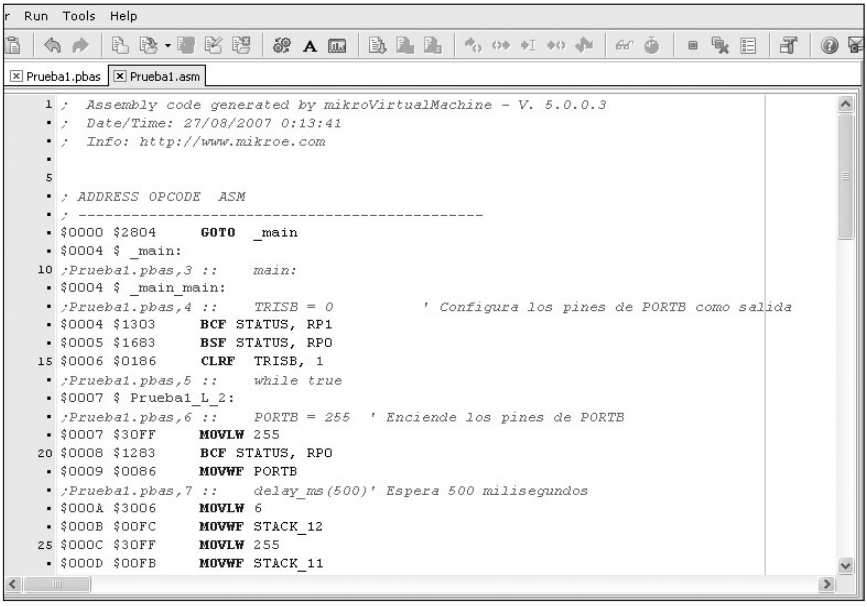


Figura 15. *Aquí podemos ver el* ***assembler*** *del código que enciende el led. Los comentarios que tiene lo hacen más legible.*

MikroBasic nos proporciona un conjunto de variables globales y constantes predefinidas. Todos los registros están implícitamente declarados como **variables globales** de tipo byte y son visibles en todo el proyecto. También nos proporciona constantes como **PORTB**, **TMR1**, etcétera. Todos los identificadores están en mayúscula, tal como se los identifica en las hojas de datos de Microchip. Para ver la lista completa de predefiniciones, podemos abrir el archivo .DEF correspondiente al micro utilizado. Estos archivos se encuentran dentro de la carpeta **defs**, donde se instaló mikroBasic.

Con respecto a las variables, podemos acceder a ellas a nivel de bit. Por ejemplo, si declaramos la variable **mivar** de tipo **longint**, lo cual define un rango de 32 bits, podemos acceder a cualquiera de ellos si escribimos **mivar.n**, donde **n** es un número entre 0 y 31, que indica el bit al cual accedemos. Este acceso es tanto a nivel de lectura como de escritura.

Elementos del léxico

Cuando un código fuente es compilado, la primera fase que se produce es la de **tokenización**. En este momento, una herramienta llamada **parser** recorre el archivo de texto y reconoce unidades significativas (**tokens**), como palabras clave, literales, variables, etcétera, y los espacios en blanco. En realidad, se llama espacios en blanco a todos los espacios, tabulaciones, fines de línea y comentarios. Todos ellos determinan el comienzo o el fin de un token. Por ejemplo:

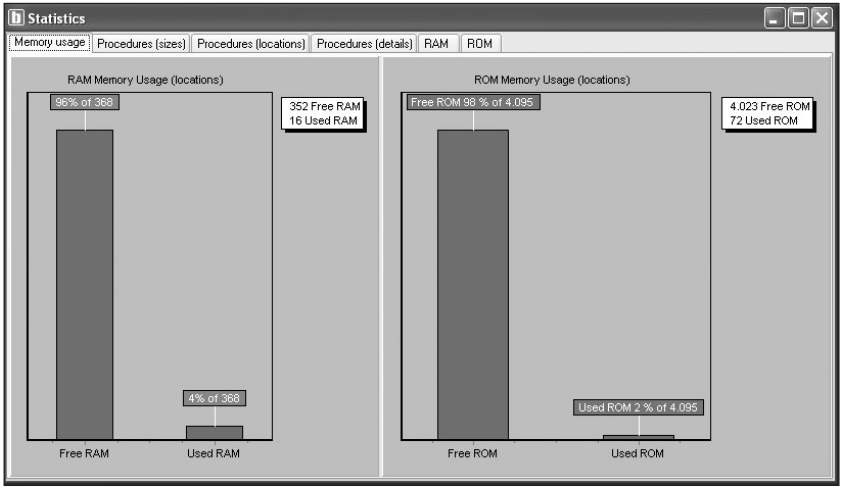


Figura 16. *MikroBasic nos brinda magníficos* ***gráficos estadísticos*** *sobre el código de máquina que genera nuestro fuente.*

```
Dim   a     as    byte
Dim   b     as    longint
```

Es equivalente a

```
Dim a as byte
Dim b as longint
```

Para los comentarios se utiliza el apóstrofe como iniciación y se finaliza con el fin de línea. No podemos poner comentarios de más de una línea, excepto que comencemos cada línea con un apóstrofe.
Los **tokens** son los elementos más pequeños que tienen significado por sí mismos en Basic. MikroBasic reconoce los siguientes tokens:

- Keywords (palabras clave).
- Identificadores.
- Constantes.
- Operadores.
- Separadores.

Por ejemplo, si tenemos el siguiente código:

```
mivar=23
```

Esto se convierte en los siguientes tokens:

- **mivar**: variable.
- **=**: operador de asignación.
- **23**: literal.
- newline: fin de la sentencia.

Los literales son aquellas expresiones que no deben ser evaluadas, que ya tienen valor por sí mismas. En el caso de los literales que representan números enteros, pueden estar antecedidos por un signo más o un signo menos. También podemos escribir literales enteros en hexadecimal si anteponemos el prefijo **$** ó **0x** (**$3F** ó **0x3F**) o en binario, si anteponemos **%** (**%100101**). Los literales de **punto flotante** representan el signo decimal con un punto. La estructura es: **parteEntera.parteDecimalExponente**, si utilizamos la notación científica tradicional. Algunos ejemplos:

```
0.          '=0.0
-1.43       '=-1.43
12.55e6     '=12.55 * 1000000
2e-5        '=2 * 10^-5 o lo
  que es igual, 2 / 100000
.04e33      '=0.04 * 10^33
```

Los literales de caracteres y de cadena se encierran entre comillas. Las comillas consecutivas, **""**, se consideran como un carácter nulo. Con respecto a los identificadores (nombres de variables, de funciones, constantes, etcétera), pueden tener una longitud arbitraria y contener

caracteres de la A a la Z, el carácter _ (guión bajo), y los dígitos del 0 al 9. La primera letra debe ser un carácter o el guión bajo. No se distingue mayúsculas, por lo tanto **var**, **Var** y **vAr** son el mismo identificador.

Organización de los módulos

Un proyecto en mikroBasic consiste en un módulo principal y en un conjunto de módulos adicionales (optativos) con librerías de funciones, declaraciones, etcétera. El módulo principal tiene una estructura como la que presentamos a continuación:

```
program <nombre>
include <otros módulos incluidos>

'**********************
'Declaraciones
'**********************

 'Declaración de símbolos
symbol…

'Declaración de constantes
const…

'Declaración de variables
dim…

'Declaraciones de procedimientos
sub procedure <nombre
     procedimiento>(<param1>, …)
     <declaraciones locales>
     …
end sub

'Declaraciones de funciones
sub function <nombre
     función>(<param1>, …)
     <declaraciones locales>
     …
end sub

'**********************
'Cuerpo de programa
'**********************

main:
     'aquí va nuestra obra
           maestra…
end.
```

Las otras unidades deberían tener el siguiente aspecto:

LIMITACIONES DE ANIDACIÓN

Cada vez que desde una función o un procedimiento llamamos a otra rutina (o a sí mismo en el caso de recursividad), decimos que aumentamos el nivel de anidación en uno. Lamentablemente, las llamadas recursivas no son soportadas por mikroBasic. De todas maneras, también tenemos un límite en los niveles de anidación, que en el caso de la familia PIC16, es de 8 niveles.

```
module <nombre>
include <otros módulos
   incluidos>

'**********************
'Interface (globales):
'**********************

 'Declaración de símbolos
symbol…

'Declaración de constantes
const…

'Declaración de variables
dim…

'prototipos de procedimientos
sub procedure <nombre
     procedimiento>(<param1>, …)

'prototipos de funciones
sub function <nombre
     función>(<param1>, …)

'**********************
'Implementación
'**********************

implements

'Declaración de constantes
const…

'Declaración de variables
dim…

'Declaraciones de procedimientos
sub procedure <nombre
     procedimiento>(<param1>, …)
     <declaraciones locales>
     …
end sub

'Declaraciones de funciones
sub function <nombre
   función>(<param1>, …)
     <declaraciones locales>
     …
end sub

end.
```

Alcance y visibilidad

Llamamos **alcance** de un identificador a la parte del programa donde el identificador se puede utilizar para acceder al objeto al que apunta.
Hay diversos alcances según el lugar donde hacemos la declaración del identificador, veamos:

- Si el identificador se declara en la sección de declaraciones del módulo principal, fuera de toda función o todo procedimiento, el alcance se extiende desde la declaración hasta el final del archivo, y se incluyen todas las rutinas que allí se encuentran. Diremos que estos identificadores tienen un **alcance global**.
- Si en cambio se declara en una función o un procedimiento, su al-

cance se extiende hasta el final de la rutina. En este caso, los identificadores son **locales**.

- Si se declara en la sección interfaz de un módulo, su alcance se extiende desde donde se declara hasta el final, y a todo módulo que use el módulo donde se encuentra la declaración. La única excepción son los símbolos, que tienen un alcance limitado al archivo donde fueron declarados.
- Si se declara en la sección implementación de un módulo, pero no dentro de una función o un procedimiento, el alcance va desde la declaración hasta el final del archivo, y está disponible en todas las funciones y en todos los procedimientos del módulo.

Llamamos **visibilidad** de un identificador a las regiones del fuente donde podemos acceder a los objetos apuntados por él. En general, alcance y visibilidad coinciden, pero no necesariamente. Podríamos tener un identificador que se vuelva invisible en una función o en un procedimiento porque un identificador local tiene su mismo nombre. Sin embargo, el objeto identificado sigue presente, aunque no es accesible. En síntesis, la **visibilidad nunca excede al alcance**, pero sí puede ocurrir que el alcance exceda a la visibilidad.

No profundizaremos más sobre el desarrollo de programas modulares, dado que realizaremos programas pequeños. Cuando veamos cuestiones sobre fútbol de robots, programaremos algunos módulos de detección del piso y de la pelota.

Variables, constantes y tipos

Las **variables**, como ya sabemos, son simplemente un identificador o un apuntador a una zona de memoria de contenido plausible de ser modificado. Todas las variables **se deben declarar** antes de usar. Para ello escribimos:

```
dim lista_de_nombres_de_
      variables as tipo
```

La **lista_ de_nombres_de_variables** es una lista delimitada por comas de identificadores válidos, que serán

LÍMITES DE TAMAÑO EN LAS RUTINAS

En las aplicaciones sobre PIC16 (que es nuestro caso), una rutina no puede exceder una página (aproximadamente, 2000 instrucciones). Si la rutina no entra en una página, se producirá un error de compilación. Si nos encontramos con este problema, basta con rediseñar nuestro código para obtener rutinas más pequeñas.

todas del mismo tipo de datos. Por ejemplo:

```
dim dias, edad, curso as byte
dim contador as word
```

Las **constantes** son datos que no varían en la ejecución del programa, es decir, no consumen memoria RAM. Se declaran de la siguiente manera:

```
const NOMBRE_DE_CONSTANTE [as
     tipo] = valor
```

El nombre de la constante debe ser un identificador válido. Por convención, se ponen todas las letras de su nombre en mayúscula, para identificarlas rápidamente en el código.

Aunque la declaración de tipo es opcional, cabe aclarar que si no lo hacemos, elegirá el tipo más chico que permita el valor asignado. Aquí tenemos algunos ejemplos:

```
const MAX as longint = 16348
const MIN = 100 'el compilador
     asume el tipo word
const LETRA='a' 'el compilador
     asume char
cont ERROR="Error grave" 'el
     compilador asume string
const meses as byte[12]=(31, 28,
     31, 30, 31, 30, 31, 31, 30,
     31, 30, 31)
```

Basic es un lenguaje con **tipos estrictos**. Esto significa que en el momento de la compilación se debe poder definir un tipo específico para cada identificador.
En estos lenguajes, los tipos sirven para determinar la cantidad de memoria necesaria, interpretar los bits que encuentran en el objeto cuando acceden y para asegurar asignaciones y pasajes de parámetros correctos.

TIPO	TAMAÑO	RANGO
byte	8 bits	0 .. 255
char	8 bits	0 .. 255
word	16 bits	0 .. 65535
short	8 bits	-128 .. 127
integer	16 bits	-32768 .. 32767
longint	32 bits	-2147483648 .. 2147483647
float	32 bits	±1.17549435082*1038 .. ±6.80564774407*1038

Tabla 1. *Tipos simples de variables.*

Los tipos simples son los que podemos observar en la **Tabla 1**.
En mikroBasic también contamos con arreglos. Un **arreglo** (**Figura 17**) es una colección de elementos del mismo tipo, identificados en su conjunto con un nombre e individualmente con un índice. Por ejemplo:

```
dim semana as byte[7]
```

Esto declara un arreglo de nombre **semana** que contiene siete elementos de tipo **byte**. Cada elemento estará identificado como **semana[i]**, donde **i** es un número entre 0 y 6. De esta manera, accedemos a cada elemento, tanto para su lectura como para su escritura. Por ejemplo:

```
semana[2] = 12
```

Esto le asigna al tercer elemento de la colección (recordemos que el primer elemento es el 0) el valor **12**.

También podemos crear **arreglos multidimensionales**. Para poder recrear una imagen sencilla, podemos pensar un arreglo de dos dimensiones como una planilla, donde un índice indica la fila y el otro la columna. Por ejemplo, para declarar un arreglo de dos dimensiones escribimos:

```
dim sueldos as integer[23][12]
```

De esta manera, tenemos una variable **sueldos** que tiene **23** filas y **12** co-

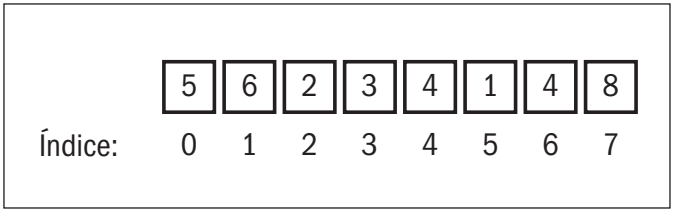


***Figura 17.** Un arreglo es una colección de elementos del mismo tipo identificados por un índice.*

CICLOS INFINITOS

A pesar de que en la programación tradicional un ciclo infinito trasluce algún error, en el caso de los robots es habitual su uso. Como el robot hará lo mismo hasta que se apague o se quede sin baterías, en general la tarea principal, y otras tareas que se estén ejecutando en forma paralela, están dentro de una estructura **while true**... **wend**. De esta manera, se ejecutará infinitamente hasta que el robot se detenga.

lumnas, desde **sueldos[0][0]** hasta **sueldos[22][11]**. Es decir, 276 elementos. Las **cadenas** son **arreglos de caracteres**, y se las declara de esa manera. Por ejemplo:

```
dim nombre as string[14]
```

Esto declara una variable **nombre** que apuntará a una cadena de **14** caracteres. De todas formas, podemos acceder a la cadena como un solo elemento, como podemos ver aquí:

```
nombre="Iris"
```

En este ejemplo, le asignamos una cadena de 4 caracteres a la variable **nombre**. El límite de cadena a asignar es de 14 caracteres.

Estructuras

Para representar objetos cuya información es un conjunto de datos de diversos tipos, utilizamos estructuras. Por ejemplo, si queremos representar un punto, necesitamos la coordenada x y la coordenada y comprendida por un nombre único. La declaración de la estructura es la que podemos ver a continuación:

```
structure nombre_estructura
dim miembro1 as tipo
…
     dim miembron as tipo
end structure
```

Donde **nombre_estructura** y **miembro1** son nombres válidos para los identificadores. Por ejemplo:

```
structure punto
     dim x as integer
     dim y as integer
end structure
```

Una vez creada la estructura, se la utiliza como si fuera un tipo en la declaración de identificadores. Por ejemplo:

```
dim posicion as punto
dim esquina as punto
```

EFICIENCIA DE TIPOS

La unidad aritmético lógica de los PICs se ha optimizado para trabajar a nivel de bytes. Aunque con mikroBasic podemos trabajar tipos de datos muy complejos, esto aumenta dramáticamente el tiempo de procesamiento. Siempre conviene usar el tipo de datos más pequeño para cada situación. Cada modelo de procesador tiene diferentes comportamientos en su unidad aritmético lógica, pero siempre es bueno usar tipos ajustados a lo necesario.

Para acceder a cada elemento de la estructura, simplemente escribimos **identificador.campo_de_estructura**.

Por ejemplo:

```
posicion.x=12
esquina.y=posicion.y
```

Operadores

En mikroBasic podemos encontrar los siguientes operadores:

- **Aritméticos**: estos operadores permiten realizar cómputos matemáticos. Sus operandos son numéricos y también retornan un resultado numérico. Ellos son los que podemos ver en la **Tabla 2**. Cabe mencionar que el operador **–** también puede usarse delante de un número para cambiar su signo.
- **Relacionales**: los operadores relacionales comparan dos expresiones y devuelven verdadero o falso según la comparación realizada. Ellos son los que se muestran en la **Tabla 3**.
- **De bits**: operan sobre bits individuales de los operandos.
  Ambos operandos deben ser de tipos con signo o sin signo simultáneamente. Los operadores de bits son los de la **Tabla 4**.

Los operadores **and**, **or** y **not** también se pueden utilizar con expresiones lógicas, es decir, cuyo valor

OPERADOR	OPERACIÓN	EJEMPLO
+	Suma	5 + 3 retorna 8
-	Resta	2 - 3 retorna -1
*	Multiplicación	5 * 4 retorna 20
/	División	14/4 retorna 3.5
div	División entera	14 div 4 retorna 3
mod	Resto de la división entera	14 mod 4 retorna 2

***Tabla 2.** Operadores aritméticos.*

OPERADOR	OPERACIÓN	EJEMPLO
=	Igual	5 = 5 retorna Verdadero (true)
<>	Distinto	5 – 4 < > 6 - 2 retorna Falso (false)
>	Mayor que	3.5 + 7.2 > 5.2 + 9.3 retorna Falso
<	Menor que	9 > 3 retorna Verdadero
>=	Mayor o igual	3 + 2 > = 1 + 4 retorna Verdadero
<=	Menor o igual	13.6 < = 12.8 retorna Falso

***Tabla 3.** Operadores relacionales.*

evaluado es verdadero o falso. Lo que retornan tiene la misma lógica que en los operadores a nivel de bits. Es decir, podemos pensar en verdadero como **1** y en falso como **0**, y a partir de allí, analizar el resultado de la operación como si fuera una operación a nivel de bits.

Sentencias

Como en todo lenguaje de programación **procedural**, tenemos un pequeño conjunto de sentencias que el compilador convierte en las instrucciones de código máquina correspondientes. En general, este conjunto es mínimo, y se enriquece con la librería de funciones y procedimientos que ofrece el ambiente. A partir de esa librería comenzaremos a construir la propia para aumentar el poder de nuestro lenguaje en el ámbito específico de nuestro trabajo. Veamos las sentencias que nos proporciona este Basic.

DIVISIÓN POR CERO

Si se utiliza de manera explícita el literal **0** como segundo operando, el compilador reportará un error y no generará código. Pero en el caso de que este problema se produzca en forma implícita (por ejemplo, cuando tenemos una variable que toma fortuitamente el valor **0** y está como dividendo) el resultado será el máximo valor posible para el tipo apropiado.

OPERADOR	OPERACIÓN	EJEMPLO
and	Retorna 1 si ambos bits en la misma posición están en 1.	12 (%1100) and 10 (%1010) retorna 8 (%1000)
or	Retorna 1 si por lo menos uno de los 2 bits en la misma posición está en 1.	12 (%1100) or 10 (%1010) retorna 14 (%1110)
xor	Retorna 1 si uno de los 2 bits en la misma posición está en 1 y el otro no.	12 (%1100) xor 10 (%1010) retorna 6 (%0110)
not	Invierte cada bit del número.	not 6 (%0110) retorna 9 (%1001)
<<	Mueve los bits una posición hacia la izquierda. Los bits ubicados más a la izquierda se descartan y a la derecha se rellena con ceros.	7 (%0111) << 2 retorna 28 (%11100)
>>	Igual a << pero a la derecha.	35 (%100011) >> 3 retorna 4 (%100)

***Tabla 4.** En esta tabla podemos observar los operadores de bits con los que podemos trabajar.*

Sentencia asm

La primera sentencia que presentaremos es **asm**, que permite incorporar código assembler en nuestro código Basic. Su sintaxis es:

```
asm
     conjunto de instrucciones
           en assembler
end asm
```

Dentro de esta sentencia, debemos comenzar los comentarios con punto y coma. Luego, al compilar, este código quedará directamente traducido a los valores correspondientes de código máquina.

Asignación

Aunque ya la hemos usado, no podemos dejar de presentar la asignación que tiene la siguiente forma:

```
variable=expresión
```

La expresión se evalúa y asigna su resultado a la variable. No debemos confundir este operador con el operador relacional homónimo. Simplemente, el contexto de uso de cada uno determina la semántica que utilizará el compilador para tomarlo como una asignación o un operador.

Sentencias condicionales

Como en otros lenguajes procedurales, contamos con sentencias condicionales que nos permiten seguir uno u otro curso de acción según determinada condición. La más conocida es la sentencia **if**, cuya sintaxis es:

```
if expresión then
     sentencias
     …
[ else
     otras sentencias ]
end if
```

La expresión del **if** debe ser una expresión **booleana**, es decir, que devuelva verdadero o falso. En el caso de retornar el valor verdadero, se ejecutan las sentencias que están dentro del primer grupo. Si el segundo grupo de sentencias está definido (los corchetes no deben escribirse,

VARIABLES DENTRO DEL CÓDIGO EN ASSEMBLER

Dentro del código en assembler, podemos usar variables que estén declaradas en la parte Basic de nuestro fuente. Una recomendación es estar seguros de que la variable está inicializada en el momento que se use de esta manera. Si no, se puede producir un error y el programa se puede colgar.

sólo nos muestra que esa sección es optativa), entonces se ejecuta este segundo grupo. Si no está, se continúa con las sentencias posteriores al **end if**. Por ejemplo:

```
if PORTA.3=1 then
     PORTB.2=1
else
     PORTB.3=1
endif
…
```

Supongamos que el bit 3 del puerto A está definido como entrada y los bits 2 y 3 del puerto B como salidas. Lo que hacemos aquí es encender el bit 2 de B si la entrada registrada en el 3 de A se enciende, y encender el bit 3 de B en el otro caso.

Otra sentencia que define el flujo del programa según una condición es la sentencia **select case**. Su sintaxis es:

```
select case selector
     case valor_1
            sentencias_1
     case valor_2
            sentencias_2
     …
     case valor_n
            sentencias_n
     [case else
            sentencias_por_
                   defecto
]
end select
```

El selector es la expresión que se evalúa para luego determinar si coincide con alguno de los valores de las sentencias **case**. Esos valores pueden ser literales, constantes o expresiones. El **case else** es optativo, y sus sentencias se ejecutan si no ingresó en ninguno de los **case** anteriores. En los valores podemos agrupar un conjunto de ellos separados por una coma, para que se ejecuten sus sentencias cuando el selector coincida con cualquiera de ellos. Por ejemplo:

```
select case resu
```

HOJA DE DATOS DE LOS PICS

Para tener información completa y detallada de cada PIC, podemos recurrir a sus hojas de datos (datasheet). Microchip las ofrece en su sitio **www.microchip.com**. Estas hojas tienen una descripción profunda del micro, su diagrama de pines en todos los modelos, diagramas internos, organización de la memoria, funcionamiento de los timers, etcétera.

```
      case 0
            oper=0
      case 1, 4, 7, 9
            oper=1
      case 2,5,8
            oper=2
      case else
            oper=3
end select
```

En el caso de que **resu** valga **0**, **oper** tomará el valor **0**. Si vale **1**, **4**, **7** ó **9**, **oper** tomará el valor **1**. En el caso de que sea **2**, **5** u **8**, el valor de **oper** será **2**. Y en otro caso, **oper** saldrá de esta estructura con valor **3**.

Sentencias de iteración

Las sentencias de iteración nos permiten **repetir** un conjunto de sentencias una determinada cantidad de veces o según se produzca una condición o no. En el primer caso, la sentencia **for** nos permite definir un número de repeticiones a ejecutarse, de la siguiente manera:

```
for contador = valor_inicial to
     valor_final [step
     valor_salto]
     sentencias
next contador
```

El contador es una variable que aumentará en cada repetición según lo que esté definido en **valor_salto**. Si el step no se define, el salto es de uno en uno. El contador comienza con el valor indicado en **valor_inicial**, y si en algún momento supera el **valor_final**, el ciclo deja de repetirse. Es decir, cada vez que se llega a **next contador**, el contador aumenta lo indicado por **valor_salto** y se evalúa si ha superado el **valor_final**. De ser así, no vuelven a ejecutarse las sentencias del ciclo. Por ejemplo, en el siguiente caso, el arreglo **a** toma el valor **255** en cada posición del **0** al **7**.

```
for i= 0 to 7
     a[i]=255
next i
```

Otra sentencia que podemos usar para una repetición es la estructura **while**, que se define de esta manera:

```
while expresión
     sentencias
wend
```

Cada vez que se está por ejecutar el conjunto de sentencias, se evalúa la expresión. Si su resultado es verdadero, se ejecutan. Si es falso, se continúa la ejecución con las sentencias posteriores al **wend**.

Con este comportamiento, podemos tener casos en los que las sen-

tencias no se ejecutan nunca, porque la expresión retorna falso en la primera evaluación. Por ejemplo:

```
i=0
while i<=7
     a[i]=255
     i=i+1
wend
```

En este ejemplo, logramos exactamente que lo que hicimos en el anterior. En general, cuando la condición de corte está basada en alguna variable que aumenta en forma lineal (de 1 en 1, de N en N) se utiliza la sentencia **for**. En otro caso, cuando la condición es más compleja y el comportamiento de las variables que entran en juego en la condición es más complejo y menos lineal que un aumento de N en N, es necesaria una estructura **while**.

Si necesitamos que el conjunto de sentencias que se van a repetir se ejecute por lo menos una vez, debemos evaluar la expresión luego de la ejecución de las sentencias.

Para ello tenemos la sentencia **do**, con la siguiente sintaxis:

```
do
     sentencias
loop until expresión
```

En este caso, no sólo el momento de evaluación varía con respecto al **while**. También la lógica se invierte. Si al llegar a **loop until** la expresión es falsa, se vuelve a ejecutar el ciclo. De lo contrario, se continúa con las sentencias posteriores a **loop until.** Tomemos el mismo ejemplo anterior y traduzcámoslo a esta nueva estructura, como vemos a continuación:

```
i=0
do
     a[i]=255
     i=i+1
loop until i>7
```

DAR VUELTAS POR LA ETERNIDAD

Es fundamental analizar si las expresiones que escribimos para ser evaluadas en las estructuras de repetición tienen alguna o algunas variables que puedan modificar su contenido en algún momento de la ejecución del programa. Si la expresión puede tener el mismo resultado eternamente, tenemos algún error conceptual en ella o en el ciclo de repetición. También puede ocurrir que utilicemos métodos desprolijos como el **break**, pero no es recomendable.

INTELIGENCIA ARTIFICIAL

Si nos interesa conocer más sobre el tema, debemos saber que existen miles de libros de divulgación vinculados a la inteligencia artificial y que lo hacen desde diversos puntos de vista.

Tal vez el más completo de ellos es **La mente nueva del emperador**, de Roger Penrose. En esta obra, su autor se ocupa de criticar la postura de la inteligencia artificial fuerte que plantea que nuestro cerebro es absolutamente imitable por una combinación artificial de hardware y software.

La belleza del libro está dada en que en sus páginas, el autor hace un recorrido por diferentes ciencias, como la física, la lógica, la matemática, la biología y otras, para presentar y sostener su postura.

Como podemos ver, el conjunto de sentencias se ejecuta sí o sí una vez en este caso. Un uso muy habitual de esta sentencia es la espera de un estado de un sensor.

Por ejemplo, si conectamos un sensor de tacto en el bit 3 del puerto B, podemos esperar que se active mientras mantenemos el estado del robot, de la siguiente manera:

```
do
loop until PORTB.3=1
…
```

El robot quedará con sus actuadores en el estado en el que se encontraba antes de llegar al loop, y cuando se produzca la activación del sensor, continuará con las sentencias que le siguen.

Sentencias de salto de ejecución de programa

Estas sentencias permiten redirigir el flujo de ejecución de un programa. Ninguna de ellas es altamente aconsejable. Para ser rigurosos aunque tengamos que trabajar un poco más, tendríamos que reemplazarlas por las sentencias ya vistas, dado que es difícil realizar el seguimiento para la depuración de estos programas. Un salto a otro lado en algún momento marea a cualquiera.

Como adelantamos, la sentencia **break** permite salir de cualquier estructura. Por ejemplo:

```
while true
     if PORTA.1=1 then
            break
     end if
     …
wend
```

En este caso, el conjunto de sentencias posterior al **if** se ejecutaría infinitamente. Pero, si en algún momento el segundo bit del puerto A se enciende, el **break** obliga a la sali-

da del ciclo y continúa con las sentencias posteriores al **wend**.

Otra sentencia de salto es el **goto**, que nos permite saltar incondicionalmente a la etiqueta referida. Su sintaxis es la siguiente:

```
goto nombre_etiqueta
```

De esta manera, salta a la sentencia siguiente donde está ubicada la etiqueta. La sintaxis de la etiqueta es la que vemos a continuación, y cabe aclarar que debemos incluir los dos puntos en la sintaxis.

```
nombre_etiqueta:
```

Otra sentencia de salto es **exit**, que nos permite salir de una función o de un procedimiento y volver al punto en el que se había invocado a la rutina. También contamos con **gosub** y **continue**, pero como entendemos que violan demasiado el flujo del programa y porque son fácilmente reemplazables por alguna de las opciones que ya presentamos, no vamos a ver detalles sobre ellas. Lo único que nos interesa resaltar en este punto, es que todas estas sentencias de salto hacen más compleja la depuración y la lectura del código, por lo que aconsejamos evitarlas siempre que sea posible.

... RESUMEN

En este capítulo hemos realizado un primer acercamiento a los diferentes lenguajes que podemos utilizar para la programación de los micros de Microchip. De todos ellos, nos hemos quedado con el que consideramos más sencillo, y que probablemente resulte familiar a todos los que alguna vez programaron en algún lenguaje. MikroBasic, además de brindarnos un lenguaje Basic muy accesible, posee una IDE muy completa y amable para este duro proceso de programación. Hemos presentado los elementos esenciales del lenguaje y nos hemos introducido en la estructura de los programas y sus sentencias. Los ejemplos que hemos mostrado son pequeños, dado que en el próximo capítulo, al conectar los motores, podremos comenzar a darle vida a nuestro robot.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿A qué llamamos lenguajes de bajo y alto nivel? ¿Qué ventajas y limitaciones presenta cada uno?

2 ¿Qué características presenta el PicBasic Pro?

3 ¿Cuál es la IDE de PicBasic Pro? ¿Qué características presenta?

4 ¿Qué es el ICD?

5 ¿Para qué es útil programar en CCS C?

6 Recuerde tres razones por las cuales elegimos mikroBasic para programar nuestro robot.

7 ¿Qué características presenta el editor de mikroBasic?

8 ¿Cómo es la organización de archivos de un proyecto en mikroBasic?

9 ¿Qué son los tokens y los espacios en blanco?

10 ¿Qué módulos encontramos en un proyecto mikroBasic?

11 Defina el alcance y la visibilidad de las variables.

12 ¿Cuáles son los tipos de operadores que podemos encontrar en mikroBasic?

13 Compare las sentencias for, while y do. Para cada una de ellas, construya una estructura con el mismo comportamiento mediante la utilización de cada una de las otras dos. Por ejemplo, construya una estructura for con while y otra con do.

Recorrer el mundo

Una de las características fundamentales de los robots autónomos móviles es la habilidad de transportarse por el ambiente en el que desempeñan sus tareas. La navegación de un robot de ruedas laterales no es sencilla si queremos optimizar el tiempo que nos lleva llegar hasta un punto determinado. Nuestro objetivo en este capítulo es aprender qué tipo de motores podemos utilizar en nuestro robot y cómo programarlos.

EL MOVIMIENTO DEL ROBOT

Nuestro robot crece. Después de varios intentos le hemos podido brindar una inteligencia precaria. Puede encender o apagar un led, lo que nos emociona cada vez que ocurre. Pero nada nos conforma.

Queremos que empiece a andar, a hacer su propio camino. Y es por eso que analizamos con cuidado cuáles serán sus futuros pasos. Lógicamente, en los diferentes tipos de ambientes donde puede moverse nuestro robot necesita de diversos mecanismos de locomoción. ¿Volará hacia otros planetas? ¿Se sumergirá en busca de tesoros? ¿Reptará en pantanos movedizos? ¿O sólo se trasladará sobre un piso pulido, sin rebordes ni pelos que traben sus motores?

Como ya habrá tiempo de hazañas y proezas, por ahora nos basta con recorrer con dos ruedas sencillas una superficie que no presente problemas. Cuando veamos que ya domina el mundo de dos dimensiones del piso, estaremos en condiciones de proyectar otros horizontes.

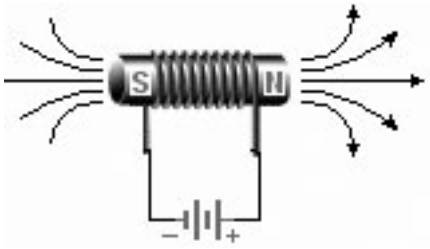


Figura 1. *Determinación de los polos por el pasaje de corriente en la bobina.*

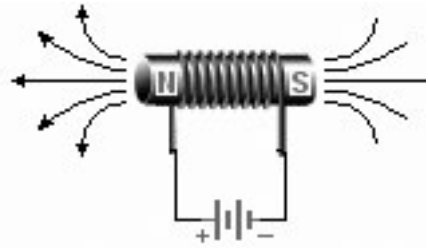


Figura 2. *Si invertimos la polaridad de la bobina, cambiamos los polos del núcleo.*

Tipos de motores

Como hemos comentado en capítulos anteriores, la arquitectura de nuestro robot con respecto a sus mecanismos de locomoción será extremadamente sencilla. Usaremos ruedas laterales, que nos permiten tener control mediante un sistema de programación al alcance de todos, y que desde el punto de vista mecánico son fáciles de implementar. En el **Capítulo 8**, donde construiremos el cuerpo de nuestra criatura, analizaremos todas las posibilidades de locomoción que tenemos con nuestro robot, pero por ahora pondremos nuestro esfuerzo en controlar los motores, que no es poco.

Aunque ya hicimos una introducción muy breve de cada uno de los tipos de

motores clásicos que podremos utilizar, ahora presentaremos en forma profunda cada uno de ellos, y desarrollaremos la electrónica necesaria para conectarlos con nuestro controlador. Además, veremos un conjunto de ejemplos de programación para poder manejarlos en forma efectiva. Es posible que al principio resulte un poco complejo, pero creemos que los ejemplos aclararán el panorama.

Motores de corriente continua

Los motores de corriente continua (CC) son los actuadores que podremos conseguir con mayor facilidad, y que seguramente estarán presentes en muchos de nuestros robots. En general, están formados por dos **imanes pegados a la carcasa** del motor y un conjunto de **bobinas de cobre en el eje** del motor. El funcionamiento se basa en la atracción o la repulsión entre el campo magnético que se genera en las bobinas por el paso de la electricidad y los imanes que se encuentran alrededor de ellas. Veamos cómo se produce esta interacción. Cuando la corriente eléctrica pasa por la bobina, genera en el núcleo un **campo magnético** con una orientación determinada, es decir, un polo Norte y uno Sur (**Figura 1**). Si cambiamos la polaridad de la bobina, se invierte la dirección de los polos (**Figura 2**). Los dos imanes que se encuentran en la carcasa tienen polos distintos.

Si aprovechamos el efecto magnético de que los polos iguales se rechazan y los opuestos se atraen, cambiamos la polaridad de la bobina, y generamos rechazo o atracción entre ellas y los imanes de la carcasa (**Figura 3**). De esta manera, y si consideramos que tenemos más de una bobina, generamos el movimiento del motor con su respectivo **torque**. Este último concepto se refiere a la **potencia** o la fuerza que tiene el giro del motor, y

RECICLADO

El reciclado es una tendencia que, además de proteger el medio ambiente, nos permite ahorrar dinero. Además de todas las recomendaciones que hacemos aquí para recuperar partes de equipos o juguetes que ya no utilicemos, Internet es una fuente adicional de ideas. Estos son algunos sitios interesantes donde se explica cómo reciclar componentes de aparatos en desuso. Muchos de ellos serán útiles para nuestros robots:

- Reciclaje y proyectos electrónicos: **http://heli.xbot.es/wp/**.
- Proyecto EconoBot (un robot de costo cero): **http://mundobot.com/blog/**.
- Disquetera convertida en robot: **www.sorgonet.com/trashing/madmaxfloppy/**.

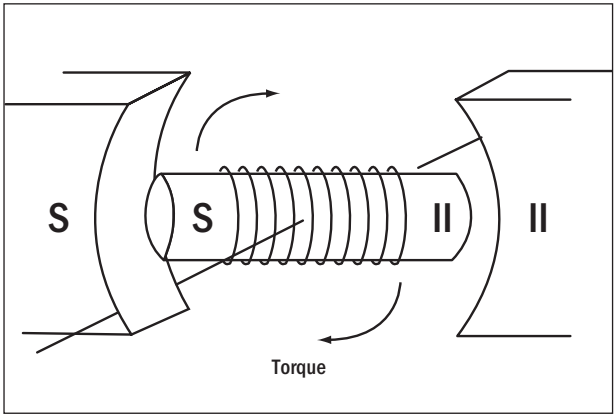


Figura 3. *Efecto de giro del motor gracias a la atracción/el rechazo del núcleo de la bobina y los imanes de la carcasa.*

que depende de la cantidad de energía que pasa por las bobinas, la cantidad de vueltas del bobinado, el grosor del alambre, etcétera.

Los motores de CC tienen dos puntos de conexión para la fuente de alimentación, y según cómo lo conectemos, el motor girará en un sentido o en otro. El problema es que nosotros queremos lograr esto en el momento en que el robot está en funcionamiento. ¡Sería desprolijo correr detrás del robot para invertir la dirección de sus motores mediante algún dispositivo físico! Por lo tanto, necesitamos algún artilugio electrónico que nos permita determinarlo al enviar una señal con nuestro micro. En este momento podríamos pensar en conectar directamente el motor al 16F88. ¡Error! Los motores absorben demasiada corriente. Si los conectamos directamente al micro, se quemarán (que es algo que con seguridad no queremos hacer). Es por eso que entre el micro y los motores incorporamos una etapa intermedia, conocida como **etapa de potencia**. Cuando utilizamos una fuente de alimenta-

▶ PARA VER EL FUNCIONAMIENTO INTERNO DE UN MOTOR CC

En la Web podemos encontrar varios sitios con información más detallada sobre el funcionamiento de los motores CC. Uno de los más completos es **www.walter-fendt.de/ph11s/electricmotor_s.htm**, donde vemos en una simulación en Java los pasos que realiza el motor en su giro. Allí podremos cambiar algunas variables que nos permitirán comprender en forma más precisa las características de estos motores.

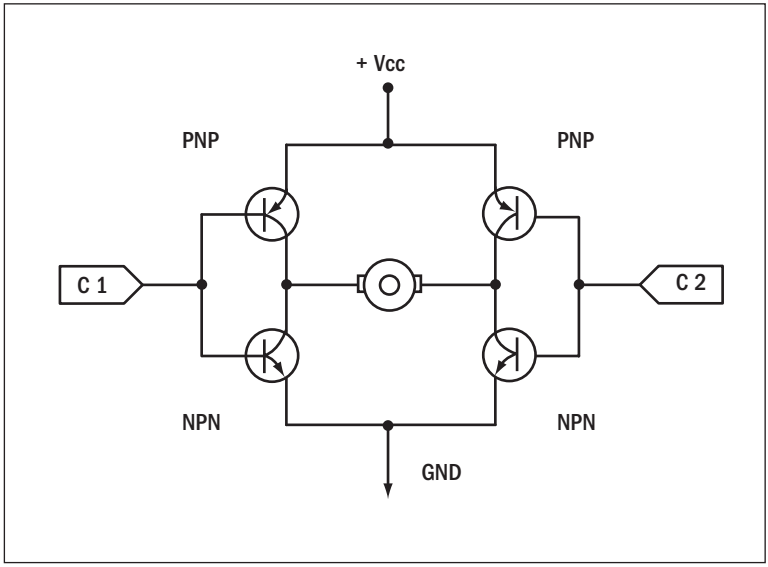


Figura 4. *Esquema básico de un puente H.*

ción simple, como es nuestro caso, se utiliza un **puente H** (**H Bridge**), cuyo esquema electrónico básico es el que podemos ver en la **Figura 4**.

Cuando C1 no tiene corriente (está en 0) y C2 sí la tiene (está en 1), el motor gira en el sentido de las agujas del reloj. En el caso contrario, el motor invierte el sentido de su giro. Cabe aclarar que el esquema es más complejo que el que vimos. Por ejemplo, necesitamos un mecanismo que impida que las dos señales se activen, porque estarían en riesgo los transistores y la fuente. Para estos y otros problemas, la solución más sencilla es utilizar un integrado que una todos estos componentes en un solo chip, y así se soluciona todo en

EL L293D Y EL L293B

En general, cuando vamos a comprar el integrado L293D nos informan que tienen el L293B. ¿Qué diferencia existe entre ellos? ¿Es lo mismo llevar el que nos ofrecen? El L293D provee 600 mA por canal, mientras que el otro nos ofrece 1 A. Por otra parte, el D tiene los diodos de protección en forma interna, que en el caso del B, debemos agregarlos nosotros para evitar que el integrado se queme ante cualquier eventualidad que tenga el motor.

Una combinación de ambos es el **TI SN754410**, que combina lo mejor de los dos. Este integrado es un reemplazo plug and play del L293D, así que podemos comprarlo sin problema.

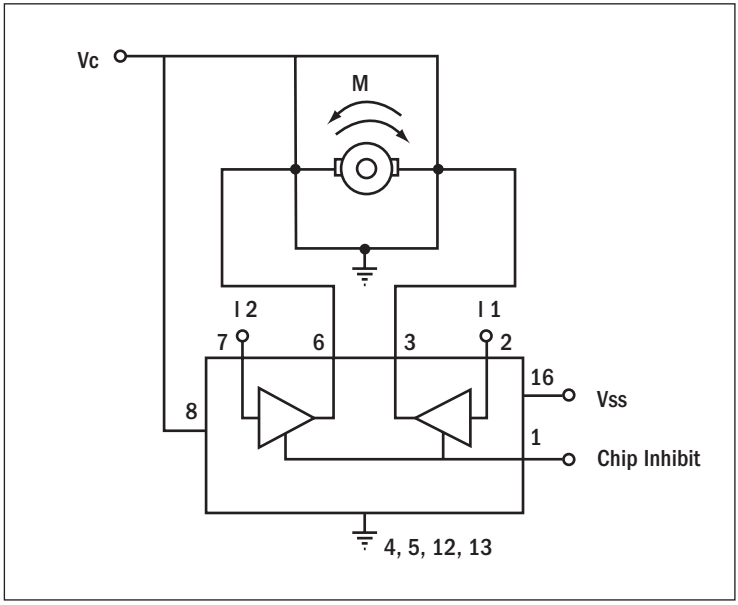


Figura 5. *Diseño de conexiones para controlar un motor con inversión de giro mediante el uso del* ***L293D****.*

un solo componente. El integrado propuesto para esta función es el **L293D**, que cuenta con cuatro drivers de potencia y diodos de protección para los motores (**Figura 5**). De la misma manera, podríamos utilizar otros integrados como el UCN5804, el BA6286, el L293B, etcétera.

Con el L293D podremos controlar cuatro motores con un solo sentido de giro, es decir, que gire o esté detenido, o dos motores con inversión de giro. Este último caso es el que más nos interesa a nosotros para construir un vehículo móvil. Si conectamos el motor en los pines 3 y 6, según el estado de los pines 1, 2 y 7, lograremos el sentido del giro del motor, su freno o un estado de punto muerto, como podemos observar en la **Tabla 1**.

En este punto, ya sabemos cómo controlar el sentido de los motores, y si están frenados o libres, pero todavía no hemos resuelto el tema de la velocidad. Es importante recordar que estos motores tienen poca fuerza y mucha velocidad, con lo cual es necesario agregarles un sistema de reducción para aumentar la fuerza y reducir la velocidad. Más adelante veremos una solución interna al motor y, en el **Capítulo 8**, opciones para soluciones externas. De todas for-

PIN 1	PIN 2	PIN 7	ESTADO DEL MOTOR
H	H	L	Gira como las agujas del reloj.
H	L	H	Gira en sentido contrario a las agujas del reloj.
H	Pin 2=Pin 7		Motor frenado.
L	X	X	Motor en punto muerto.

Tabla 1. *Cómo lograr cada estado del motor con los pines del L293D. H: high (alto); L: low (bajo).*

mas, necesitamos un mecanismo electrónico de control para poder modificar la velocidad del motor. Uno de ellos es la **modulación por ancho de pulso** (*Pulse With Modulation*, **PWM**), que ya mencionamos para el control de los servos.

También en los motores de CC utilizamos PWM, pero en un sentido distinto al que se usa con los servos. Dado que necesitamos generar los pulsos, es necesario entender cómo funciona esta modulación. En la **Figura 6** podemos ver un ejemplo gráfico de esto. En PWM, la frecuencia de los pulsos se mantiene constante. Lo que se modifica es el tiempo en el que la señal está alta dentro de esos pulsos. Cuanto más largo sea el tiempo en el que la señal esté alta, mayor será la velocidad del motor. Para dar un ejemplo más sencillo, imaginemos que tenemos el motor conectado a un interruptor que controlamos con la mano. Con una frecuencia de dos segundos, encendemos (señal alta) y apagamos (señal baja) el motor dentro de ese tiempo. Por ejemplo, podríamos tenerlo un segundo prendido y un segundo apagado. O un segundo y medio prendido y medio segundo apagado. En el primer caso, se lograría mayor velocidad que en el segundo ejemplo. La velocidad mayor se da cuando lo tenemos prendido prácticamente los dos segundos completos.

Con este control, la cantidad de corriente es constante, la tensión no varía y, por lo tanto, el torque es el

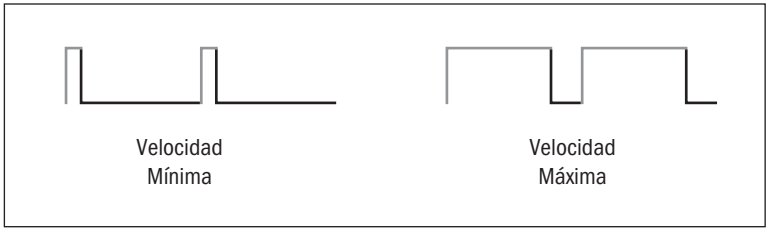


Figura 6. *Ejemplo gráfico de la modulación por ancho de pulsos.*

mismo. Como veremos más adelante, lograremos el control del motor con PWM mediante la programación de nuestro micro. Gracias al L293D, no necesitaremos más electrónica que la presente en nuestra controladora, dado que los diodos de protección del motor se encuentran incorporados en el integrado.

Otro mecanismo habitual para el control de la velocidad de los motores de CC es la **modulación por frecuencia de pulsos** (*Pulse Frequency Modulation*, **PFM**). En este caso, la proporción de la señal alta y baja dentro del pulso se mantiene constante. Lo que cambia es la frecuencia de los pulsos. Cuanta más alta es la frecuencia, la potencia aumenta. No entraremos en detalle porque no será el mecanismo que utilizaremos en nuestro robot.

Ejemplos de programación de un motor CC con nuestro controlador

Como podemos observar en la construcción del controlador, el motor 1 está conectado a los pines 11 y 14 del L293D (Output 3 y 4, respectivamente) y el motor 2 a los pines 3 y 6 (Output 1 y 2). Para realizar nuestros ejemplos trabajaremos sobre el motor 1, pero desde ya que lo que veamos sobre este motor es aplicable al otro. El control de los pines 11 y 14 está dado por los inputs 3 y 4, ubicados en los pines 10 y 15 del L293D. Si seguimos las conexiones de nuestro controlador, podemos ver que estas dos entradas están controladas por RB1 y RB2 del 16F88 (pines 7 y 8). De esta manera, podemos reformular la tabla que vimos antes para mostrar cómo se comportará el motor 1 según el estado de los pines 7 y 8 del 16F88. El resultado de esta reformulación lo podemos ver en la **Tabla 2**.

Sin embargo, como detalle adicional, cabe aclarar que no podremos lograr este último estado en nuestro controlador porque el pin 9 del L293D está en alto constantemente y no está controlado por el 16F88.

III PROBLEMAS FÍSICOS

Cuando construimos nuestros robots, aparecen problemas físicos que seguramente hemos visto en nuestros estudios, pero que tal vez no vinculamos. Por ejemplo, uno de los problemas es el tema de la reducción. ¿Por qué si aumentamos la fuerza disminuye la velocidad? ¿Cómo deben ser los engranajes para lograr esto? De la misma manera, otros conceptos tecnológicos fundamentales que surgen de nuestra obra serán las poleas, las cadenas de transmisión, las estructuras rígidas y flexibles, las palancas, etcétera. Tal vez no esté nada mal repasar estos conceptos para entender por qué funciona o no nuestro robot.

L293D-PIN 9	L293D-PIN 10 16F88-PIN 8	L293D-PIN 15 16F88-PIN 7	ESTADO DEL MOTOR
H	H	L	Gira como las agujas del reloj.
H	L	H	Gira en sentido contrario a las agujas del reloj.
H	Pines en mismo estado		Motor frenado.
L	X	X	Motor en punto muerto.

Tabla 2. *Estados del motor de CC al utilizar la programación de nuestro 16F88.*

Ejemplos de código en mikroBasic

Aunque los ejemplos más completos estarán incluidos en los capítulos donde haremos robots para cumplir misiones específicas, ya podemos comenzar a realizar algunos programas para empezar a testear a nuestro robot. En el capítulo pasado encendíamos y apagábamos todo el **PORTB** para encender y apagar el led. A partir de ahora comenzaremos a ser más delicados, y sólo encenderemos los bits que sean necesarios para cada caso. Por ejemplo, en el caso del programa que titila el led, sólo deberíamos encender el Rb3, como vemos a continuación:

```
program TitilaLedPreciso

main:
    TRISB = 0
‘ Configura los pines de PORTB
como salida
    while true
          PORTB.3 = 1
‘ Enciende RB3 donde está
conectado el led
          delay_ms(500)
     ‘ Espera 500 milisegundos
          PORTB.3 = 0
     ‘ Apaga RB3
          delay_ms(500)
     ‘ Nuevamente espera 500
     milisegundos
    wend
end.
```

Con la misma lógica, podemos hacer que el motor 1 encienda un segundo en un sentido, otro segundo en el sentido contrario, y un segundo se encuentre detenido (¡No nos olvidemos de que debemos conectar el motor en 1!).

```
program mueveMotorCC

main:
    TRISB = 0
     ‘ Configura los pines
     de PORTB como salida
    while true
    ‘Configuramos el Pin 7
     y el 8 para girar en
     un sentido
```

```
            PORTB.1 = 1
              ' Enciende RB1
              (Pin 7 del 16F88)
            PORTB.2 = 0
              ' Apaga RB2 (Pin 8
              del 16F88)
            delay_ms(1000)
              ' Espera 1 segundo
              'Configuramos
              el Pin 7 y el 8
              para girar en el
              otro sentido
            PORTB.2 = 1
              ' Enciende RB2
              (Pin 8 del 16F88)
            PORTB.1 = 0
              ' Apaga RB1 (Pin 7
              del 16F88)
            delay_ms(1000)
              ' Espera 1 segundo
            'Configuramos el Pin
              7 y el 8 para
              que el motor se
              detenga
            PORTB.1 = 0
              ' Enciende RB1
              (Pin 7 del 16F88)
            PORTB.2 = 0
              ' Enciende RB2
              (Pin 8 del 16F88)
            delay_ms(1000)
              ' Espera 1 segundo
        wend
    end.
```

Motorreductores CC

Como ya comentamos, los motores CC tienen mucha velocidad y poca fuerza, por lo que se hace necesario el uso de un sistema de reducción para poder equilibrar estas variables. Una de las soluciones es construir una caja reductora externa al motor. Sin embargo, en los últimos años han surgido motores de CC que ya proveen esta solución a nivel interno. A estos motores se los conoce como **motorreductores**. Consisten en un **micromotor** de CC con una caja reductora que usa el sistema de tipo planetario en un compartimiento sellado, lo que libera al sistema de la suciedad que podría afectar su funcionamiento (**Figura 7**).

▶ DÓNDE CONSEGUIR MOTORREDUCTORES

Los motorreductores son más difíciles de conseguir que los motores de CC convencionales. Sin embargo, una forma sencilla de hacerlo es convertir un servo como veremos al final de este capítulo. Si queremos comprar directamente uno, aquí van algunas páginas web de proveedores:

- Ignis (fabricante): **www.ignis.com.ar**.
- Robodacta (proveedor de elementos de robótica): **www.robodacta.com**.

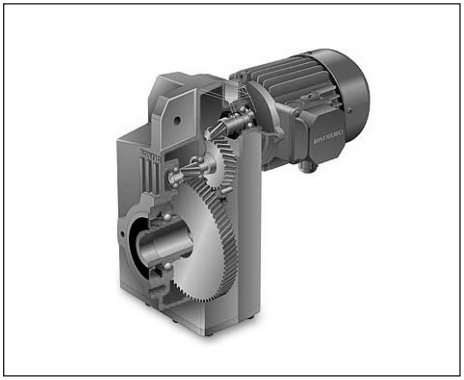


Figura 7. *Vista interna de la caja de reducción de un motorreductor pendular.*

Como ejemplo de estos motorreductores, podemos presentar el modelo **MR-4** de **Ignis**, que tiene un torque de 0,12 a 2 Kgf*cm, según el modelo, con un consumo de 100 mA. Los modelos existentes son el 4-15, 4-50, 4-100 y 4-200, donde el último número indica la cantidad de revoluciones por minuto (RPM: a mayor RPM, menor fuerza, como ya hemos visto). Funcionan desde los 6 hasta los 12 V, pero toda la información de su hoja de datos está basada en una alimentación de 12 V (**Figura 8**).

En algunos casos, es posible conseguir un reductor para conectar en forma externa a un motor de CC, pero en términos de costos y de complejidad en la conexión de materiales, no es aconsejable.

Motores paso a paso (Motores PaP)

Como hemos visto, el control de un motor CC es relativamente sencillo. Pero, si necesitáramos precisión en la cantidad de rotaciones del motor, sería casi imposible lograrlo con un

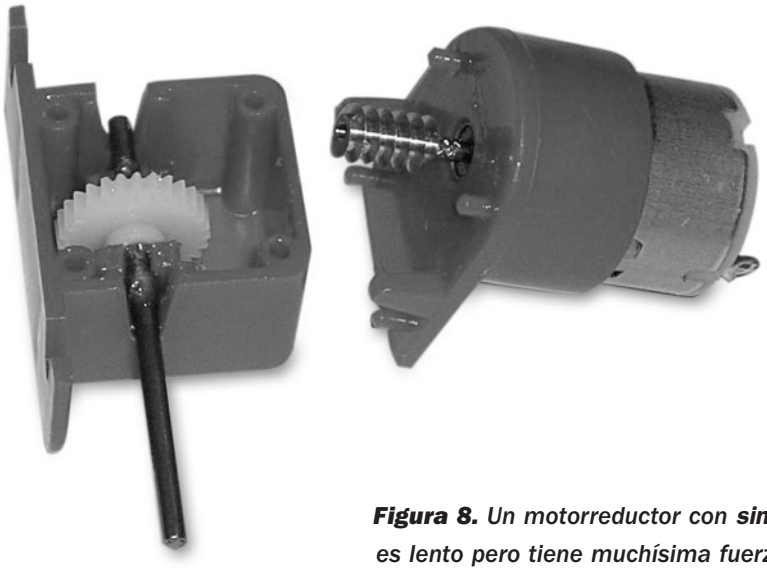


Figura 8. *Un motorreductor con* ***sinfín*** *es lento pero tiene muchísima fuerza.*

Figura 9. *En esta imagen podemos ver ejemplos de motores paso a paso.*

motor de ese tipo. Los motores de CC tardan un tiempo en lograr la velocidad buscada y cuando se los deja de alimentar, poseen inercia y tardan en detenerse. Por lo tanto, es una cuestión de suerte lograr un número preciso de vueltas. Si necesitamos fracciones de vueltas, esto es aún más complejo. Si le agregamos un sistema de reducción al motor podremos disminuir el problema, pero no desaparece por completo por la cantidad de variables que entran en juego: fricción de los engranajes del sistema de reducción, peso del rotor, fricción interna del motor, temperatura, etcétera.

Para lograr esta precisión necesitamos un **motor PaP** (**Figura 9**) o un servo. Los motores PaP tienen un mecanismo que les permite girar un ángulo determinado. Los más comunes están formados por un **rotor** (un imán permanente) y un conjunto de bobinados en el estator. No giran libremente por sí mismos, sino que avanzan al girar pequeños pasos por cada pulso que se aplica. El tamaño del paso, en grados, es lo que defi-

III ELEMENTOS QUE SE OBTIENEN AL RECICLAR APARATOS

Disqueteras de 3 1/2: motor paso a paso del cabezal, motor de cc del giro del disco y electrónica de control de los motores.

Disquetera de 5 1/4: lo mismo que en la disquetera de 3 1/2.

Reproductor de DVD: distintos drivers para motores, motor paso a paso del cabezal, motor de CC del giro del disco, motor CC de la carga del disco.

Impresora: motor paso a paso de avance del papel, motor paso a paso del cabezal, drivers de los motores paso a paso.

nirá la precisión que tendremos en nuestro motor. Los pasos más comunes que podemos encontrar son: 0,72°; 1,8°; 3,6°; 7,5°; 15° y 90°. En el primer caso, para dar un giro completo necesitaremos 500 pasos. En cambio, en el último, con 4 pasos habremos dado la vuelta entera. En general, en la carcasa del motor se indica la cantidad de grados por paso. Si esto no es así, aparecerá la cantidad de pasos por revolución. Para calcular los grados por paso, dividimos 360° por la cantidad de pasos. Dentro de los motores de imán permanente, nos encontramos con dos tipos distintos, los **unipolares** y los **bipolares**, que exigen circuitos de control distintos (**Figura 10**).

Los bipolares son más complejos de controlar que los unipolares, pero son más livianos y tienen mejor torque. El control ha dejado de ser un problema gracias a integrados como el L293D. Una vez que consigamos un motor de este tipo, debemos distinguir los cables 1a, 1b, 2a y 2b. Esto es sencillo con un tester, dado que podemos detectar continuidad entre los cables que pertenecen a la misma bobina.

No. DE PASO	1A	1B	2A	2B
1	+Vcc	Gnd	+Vcc	Gnd
2	+Vcc	Gnd	Gnd	+Vcc
3	Gnd	+Vcc	Gnd	+Vcc
4	Gnd	+Vcc	+Vcc	Gnd

***Tabla 3.** Secuencia de pasos para poder controlar un motor paso a paso bipolar.*

Para lograr el giro del motor bipolar, debemos invertir las polaridades de las bobinas 1 y 2 en una determinada secuencia para girar a favor de las agujas del reloj, y en la secuencia invertida para girar en sentido contrario. La secuencia es la que vemos en la **Tabla 3**.

Para realizar este control usaremos el L293D. En este caso, podremos controlar sólo un motor paso a paso, dado que tendremos que usar los cuatro puentes H que ofrece el integrado para lograr la secuencia. Deberíamos conectar cada uno de los 4 ca-

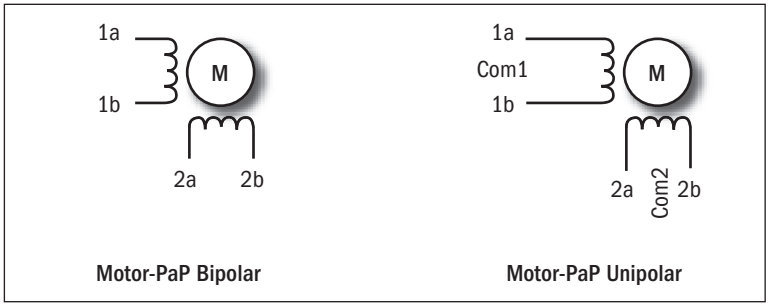


***Figura 10.** Esquemas de las bobinas de los motores bipolares y unipolares.*

bles en los conectores del motor 1 y el motor 2 de nuestro controlador (**Figura 11**). Luego, si programamos las salidas del 16F88 que se corresponden con los inputs del L293D para seguir la secuencia anterior, lograremos el giro del motor. Cuanto más rápida suceda esa secuencia, más rápido será el movimiento del motor.
Los **motores unipolares** cuentan con 5 ó 6 cables, según cómo sea el conexionado interno. Si es de 6 cables, tendremos que detectar cuáles representan a Com1 y Com2 (**Figura 10**). Al unir esos dos cables, tendremos la misma configuración que en los motores de 5 cables (**Figura 12**). Para detectar las parejas 1A y 1B, y 2A y 2B, medimos la resistencia. Cuando encontremos la resistencia más alta, habremos determinado la pareja correspondiente. Los cables Com1 y Com2 contra esos cables tienen una resistencia baja (y además, habitualmente, son los únicos dos que tienen el mismo color). Si la medición no marca nada, hemos conectado cables de bobinas distintas. Por lo tanto, ya tenemos definidas las parejas y el cable común.
Ahora, para determinar cuál es cuál, alimentamos al motor por el quinto cable o por el cable común con la tensión correspondiente. Tomamos uno de los otros cuatro cables, le ponemos la etiqueta 1A y lo conectamos a masa. Luego tomamos otro y también lo conectamos a masa. Si gira en sentido de las agujas del reloj, lo nombramos 1B, si gira al revés, 2B y si no gira, 2A. De esta manera, podemos etiquetar todos los cables.
Tenemos tres formas distintas de controlar los motores unipolares. En todas ellas, si invertimos la secuencia cam-

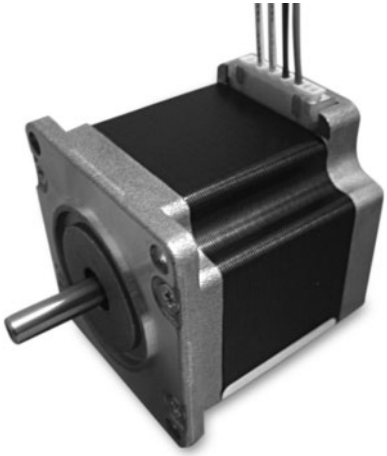


Figura 11. *Un ejemplo de motor bipolar, con 4 cables.*

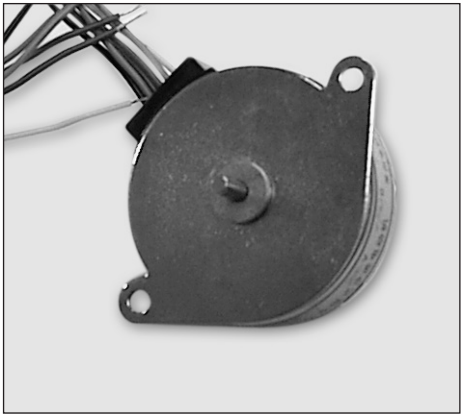


Figura 12. *Ejemplo de motor unipolar con 5 cables, que tiene los dos rojos comunes.*

biamos el sentido de giro. El control más habitual es encender de a dos las bobinas, de la siguiente manera:

No. DE PASO	1A	1B	2A	2B
1	H	H	L	L
2	L	H	H	L
3	L	L	H	H
4	H	L	L	H

***Tabla 4.** Secuencia de pasos para controlar un motor unipolar.*

Al encender las bobinas de a dos, el campo magnético es más fuerte, y el motor tiene mayor torque y más fuerza al ser frenado. Otro control es encender las bobinas de a 1:

No. DE PASO	1A	1B	2A	2B
1	H	L	L	L
2	L	H	L	L
3	L	L	H	L
4	L	L	L	H

***Tabla 5.** Otra secuencia para controlar un motor bipolar, más sencilla pero con menos torque.*

Es más sencillo que el esquema anterior, pero el motor tiene menos torque. Si combinamos estos dos esquemas podemos duplicar la cantidad de pasos de este motor, como podemos ver a continuación:

No. DE PASO	1A	1B	2A	2B
1	H	L	L	L
2	H	H	L	L
3	L	H	L	L
4	L	H	H	L
5	L	L	H	L
6	L	L	H	H
7	L	L	L	H
8	H	L	L	H

***Tabla 6.** Combinación de los dos mecanismos anteriores que nos permiten mayor precisión en los giros.*

De esta forma, duplicamos la cantidad de pasos al girar el rotor en cada bobina y entre dos bobinas contiguas. A esta secuencia se la conoce como **secuencia de medio paso**. Hay que tener en cuenta que en los tres tipos de controles debemos ser cuidadosos con la velocidad en el cambio de los estados, porque si es muy veloz podemos no alcanzar el tiempo necesario para que el rotor se acomode en la nueva posición. Si es así, obtendremos un movimiento extraño, aleatorio del motor.

DETECTAR EL TIPO DE MOTOR PASO A PASO A SIMPLE VISTA

- Un motor PaP con 5 cables es casi seguro de 4 fases y unipolar.
- Un motor PaP con 6 cables probablemente sea de 4 fases y unipolar, con 2 cables comunes para la alimentación. Buscar si dos cables tienen el mismo color y unirlos.
- Un motor PaP con 4 cables con seguridad es bipolar.

Servos

Un servo es un motor de CC pero con dos características fundamentales que lo diferencian: una **caja de reducción interna** al motor que le brinda un gran torque y un **sistema electrónico de control** que le permite posicionar al motor en el ángulo deseado. Se utilizan con mucha frecuencia en aparatos radiocontrolados como aviones, barcos y autos a control remoto. También se usan mucho en robótica por su bajo peso y sus altas prestaciones (**Figura 13**).

Los servos tienen 3 cables o terminales:

- **Terminal positivo**: recibe la energía que alimenta al motor (usualmente, de 4 a 8 voltios).
- **Terminal negativo**: dirigido a tierra.
- **Terminal de control**: es por donde ingresamos la señal que permite determinar el ángulo del motor.

Los colores de estos cables varían según cada fabricante, pero en la **Tabla 7** presentamos un detalle con los colores utilizados por los principales fabricantes de servos.

Habitualmente, la capacidad de giro de un servo es de 180 grados, pero más adelante veremos cómo eliminar este límite. El circuito de control posiciona al eje y compara la señal que recibe de un potenciómetro interno con la señal de control externa. Cuando gira el eje, también gira el potenciómetro. Por último, el ángulo va a estar determinado por la duración de un pulso que se aplica

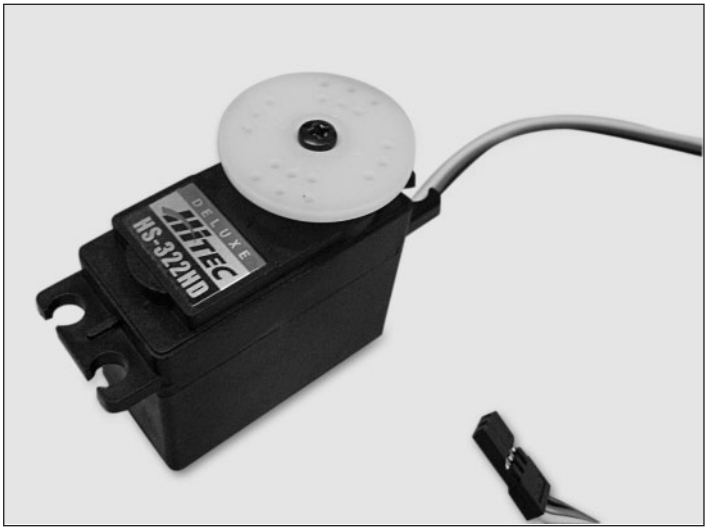


Figura 13. *Servomotor* ***Hitec****, muy utilizado en robótica.*

FABRICANTE	TERMINAL POSITIVO	TERMINAL NEGATIVO	ENTRADA DE SEÑAL
Futaba	Rojo	Negro	Blanco
Fleet	Rojo	Negro	Blanco
Hitec	Rojo	Negro	Amarillo
Airtronics	Rojo	Negro	Naranja
JR	Rojo	Marrón	Naranja
Kraft	Rojo	Negro	Naranja

***Tabla 7.** Los colores de los tres cables de los servos de las marcas más conocidas.*

al cable de control. Esta señal es una onda cuadrada de 1,5 ms que se repite a un ritmo de entre 10 a 22 ms, y con un valor de pico entre 3 y 5 V. La frecuencia puede variar entre marcas distintas, pero debe ser una señal estable para poder conseguir la rotación precisa. Aquí aplicaremos de nuevo lo aprendido en PWM. La posición del eje dependerá del ancho del pulso de la señal de control. Por ejemplo, cuando el pulso se mantiene en 1,5 ms, el eje se ubicará en el ángulo 0, y vamos hacia -90° cuando disminuimos el ancho del pulso y hacia 90 en el caso contrario.

La descripción que hemos hecho se refiere a los servos analógicos (que son los más comunes). Los **servos digitales** tienen, en su placa de control, un micro que analiza la señal, la procesa y controla el motor. Reaccionan mucho más rápido a los cambios y tienen

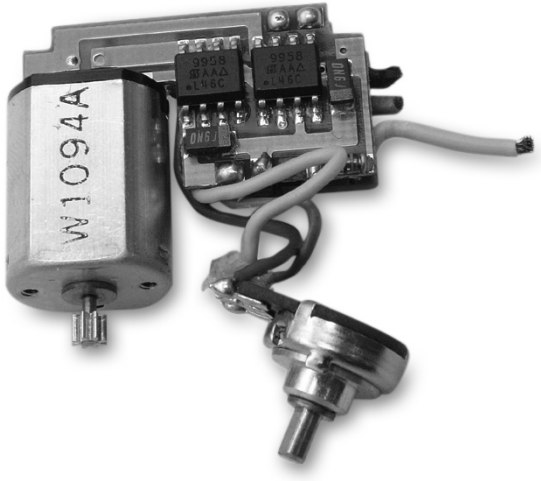


***Figura 14.** Un servo visto por dentro.*

más fuerza. Como contrapartida, son más caros y el consumo de energía es mayor que en los analógicos.

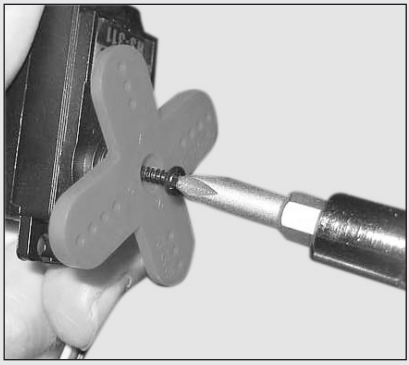
Modificación de los servos

Como ya hemos comentado, los servos tienen un radio de giro de aproximadamente 180 grados. Su sistema de control nos permite definir su ángulo de rotación en forma precisa y sencilla, pero muchas veces necesitamos que el motor gire 360° en forma continua, como en el caso de las ruedas. Para ello vamos a tener que realizar un conjunto de modificaciones en el servo. Estas modificaciones pueden dañar el servo (de por sí lo dañan porque cambian su funcionalidad), por lo que sólo recomendamos realizarlas si no conseguimos motores de CC con caja reductora integrada, como los que hemos presentado al comienzo de este capítulo.

Una de las modificaciones posibles es eliminar el circuito de control y cortar los topes mecánicos. De esta forma, lo convertiremos en un motor de CC con caja reductora. El cable de control desaparece y sólo tenemos dos cables donde se aplica corriente. A mayor corriente tendremos mayor velocidad y, si modificamos la polaridad, podemos cambiar el sentido del giro. A continuación veremos los pasos necesarios para realizar la modificación, pero cabe aclarar que las imágenes sólo son de referencia, ya que debemos buscar los pasos correspondientes para nuestro servo específico.

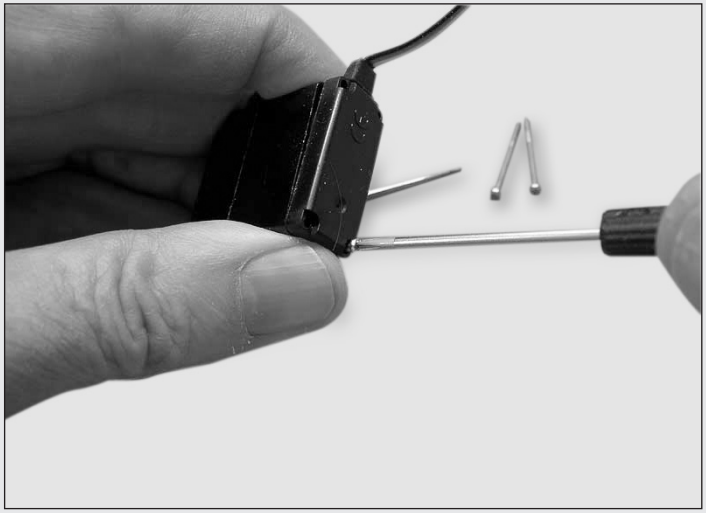
■ Modificar un servo — PASO A PASO

1 Si el servo posee un engranaje externo en el eje, desatorníllelo con cuidado, sin forzar el giro del eje más allá de su tope de rotación.



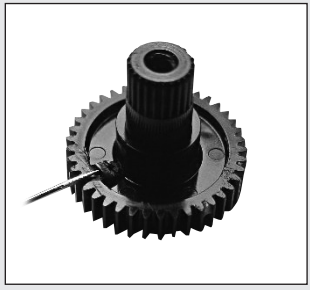
►

2 Abra la tapa posterior del servo desenroscando los cuatro tornillos y desmóntelo en forma completa.

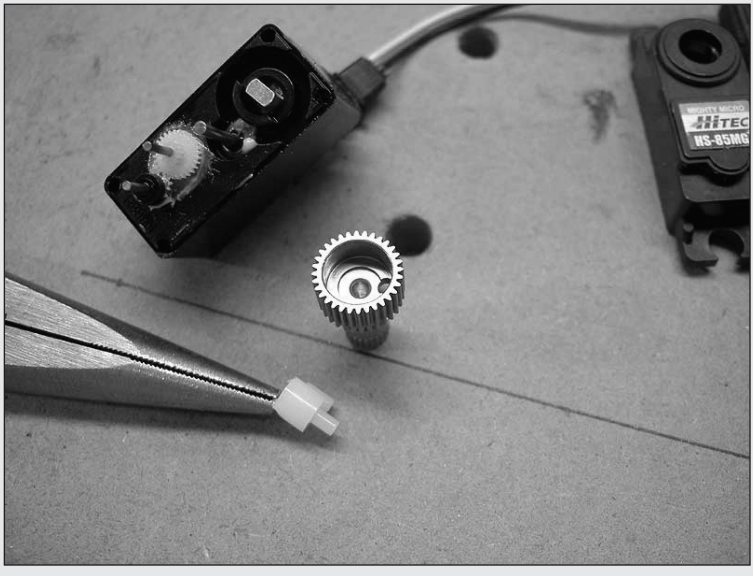


3 Saque la tapa superior que encierra la caja reductora y preste mucha atención a la disposición de los engranajes para volver a ubicarlos en su lugar.

4 Corte o extraiga cuidadosamente el tope físico que tiene el engranaje de salida del servo. Lije con una lija fina para no dejar rebordes o muescas que provoquen fricciones hostiles al movimiento de la caja reductora.



5 Elimine el circuito de control. La forma de hacer esto va a depender del modelo del servo en cuestión. En algunos casos, basta con desvincular uno de los engranajes del potenciómetro. En otros, habrá que encontrar el modo de comunicación de los engranajes con el sistema de control. Para ello, es conveniente buscar en Internet los pasos necesarios para modificar el servo específico que desee adaptar.



Volvemos a repetir: sólo es necesario realizar esto si no conseguimos un motorreductor. De todas maneras, este procedimiento era más habitual cuando no se conseguían con facilidad, pero hoy es mucho más sencillo comprar directamente uno de ellos que modificar un servo.

MODIFICAR SERVOS

A continuación presentamos las direcciones de algunos sitios donde se proporcionan los pasos para modificar diversos modelos de servos:

- Futaba 3003: **www.kronosrobotics.com/an116/GAN116_3003.htm**.
- Hitec HS 300: **www.kronosrobotics.com/an116/GAN116_hs300.shtml**.
- Futaba S148: **www.seattlerobotics.org/encoder/200304/Futaba%20S148%20Servo%20mod%20for%20PWM.htm**.
- Sub micro CS21: **www.dprg.org/projects/1998-04b/index.html**.

... RESUMEN

Hemos llegado al hermoso momento en que nuestra criatura sale a recorrer el mundo. Pero no todo es tan sencillo. En este capítulo hemos analizado los diferentes tipos de motores que pueden usarse con nuestra alimentación de corriente continua. Los motores de CC son sencillos de conseguir y fáciles de controlar, pero tienen poco torque. Por lo tanto, debemos agregarles una caja de reducción para poder mover las ruedas a una velocidad razonable y con la fuerza necesaria. Por suerte contamos con motorreductores, que ya nos ofrecen ese sistema en forma interna en la caja del motor, aunque con un costo mayor. También podemos utilizar motores paso a paso, que en general están presentes en disqueteras, impresoras, discos rígidos, etcétera. El control es un poco más complejo, pero nos brindan precisión en sus movimientos. Si queremos mayor precisión y no necesitamos giros completos, los servos son la tercera opción para los movimientos de nuestro robot.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Qué es el torque? ¿Cómo se logra en un motor de corriente continua?

2 ¿Qué integrado utilizamos para controlar los motores de CC? ¿Por qué no los controlamos directamente desde el 16F88?

3 ¿Qué es un puente H?

4 ¿Qué es el PWM?

5 ¿Cuáles son los cuatro estados del motor de CC? ¿Cómo lo logramos con el L293D?

6 ¿Qué es un motorreductor? ¿Qué ventajas y desventajas presenta con respecto a un motor de CC convencional?

7 ¿Cómo funciona un motor paso a paso?

8 ¿Qué características tienen los motores PaP bipolares y unipolares?

9 ¿Cómo se distingue el cableado de los motores PaP?

10 ¿Qué componentes tiene un servo?

11 ¿Cuál es el objetivo de modificar un servo? ¿Por qué no se recomienda hacerlo?

EJERCICIOS

1 Mediante la utilización de la modulación por pulsos, haga un programa que, según el valor de una variable, encienda con mayor o menor potencia el led de la controladora.

2 Realice un programa para controlar los dos motores: los primeros 3 segundos ambos motores deben moverse en un sentido (lo consideramos avance), luego deben retroceder otros 3 segundos, posteriormente girar para un lado, dejar un motor para adelante y otro para atrás, y por último invertir el sentido del giro. Estas últimas dos etapas, también de 3 segundos cada una.

3 Mediante la utilización de la modulación por pulsos, haga un programa que, según el valor de una variable, modifique la velocidad del motor 1.

4 Conecte un motor bipolar en los conectores de motor 1 y motor 2, y programe su funcionamiento con la técnica vista en la sección correspondiente.

Sensar el mundo

Cuando construimos robots autónomos, uno de los desafíos más importantes es la habilidad de adaptarse al mundo que los rodea. Si no incorporamos sentidos a nuestro dispositivo, sólo tendremos un juguete a control remoto, donde la inteligencia y la captación del ambiente están en nuestras manos. En este capítulo conoceremos los tipos de sensores que podemos encontrar y veremos la electrónica y la programación que usaremos en los próximos capítulos.

ADAPTACIÓN AL ENTORNO

Ya piensa, ya camina… La obra continúa creciendo. Y sin embargo, cada vez que lo ponemos a nuestro lado para que nos acompañe, se lleva por delante una pared, se pierde por lugares insólitos o se cae por una escalera. Algo nos falta, algún soplo de creación nos hemos olvidado de brindarle a nuestro robot. ¡Nos hemos olvidado de **incorporarle sentidos**! Sin ver, sin tocar y sin escuchar, es imposible que se separe de nuestra mano y de nuestros ojos. Y es por eso que empezamos a pensar cuáles serían los sentidos esenciales que podemos brindarle. ¿Le agregamos la vista? Es un desafío complejo, demasiada información para su pequeña inteligencia. ¿El tacto será suficiente? Tendrá una vida dura, golpeándose contra todo para poder captar el mundo que lo rodea. ¿El olfato será de utilidad? Muchas dudas nos preocupan, pero nos lanzamos decididos a buscar qué es lo que nos puede ayudar. Y encontramos que en el mundo de la electrónica, los sentidos se multiplican infinitamente.

Tipos de sensores

Los sensores tienen como objetivo **captar alguna señal** brindada por el ambiente que rodea al robot, y transformar esa señal en un **impulso eléctrico** que reciba nuestro controlador. Luego, desde el programa que usemos para darle comportamiento, interpretaremos este impulso para actuar en consecuencia. Cuando hablamos de señales del ambiente, nos referimos a la información habitual que, como humanos, podemos captar: imágenes visuales, olores, información táctil, so-

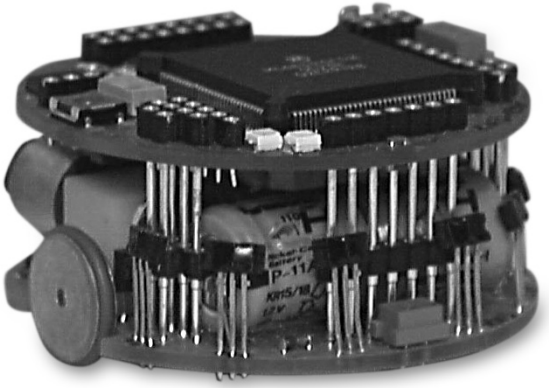


Figura 1. *Kephera es un robot con ocho sensores infrarrojos y una cámara de visión lineal.*

nidos y sabores. Pero esto no termina aquí: podemos utilizar otros mecanismos que superen al sistema sensorial humano. Contamos con sensores ultrasónicos para medir distancias, sensores de metales, detectores de diversos gases, etcétera.

La oferta es muy grande, y cada tipo de sensor nos obliga a incorporar algo de electrónica y de programación en nuestro robot. En este capítulo veremos los tipos de sensores que podemos encontrar habitualmente en el mercado, sus características y cómo podríamos incorporarlos a nuestro robot. Luego nos detendremos en los sensores que vamos a utilizar para los desafíos que hemos planteado en este libro.

Características esenciales de los sensores

Los sensores determinan en forma precisa el comportamiento de nuestro robot. Es por eso que cuando elegimos el tipo de sensor que incorporamos, debemos analizar un conjunto de características que nos permitirán definir si es o no el que necesitamos para nuestra tarea. Éstas son:

- **Fiabilidad**: esta característica está muy vinculada al ambiente en el que se mueve nuestro robot. Por ejemplo, si utilizamos un sensor de luz o brillo en un lugar con luz estable, podremos utilizar sensores de menor calidad porque sabemos que los resultados, luego de calibrar el sensor, se mantendrán parejos y podremos confiar en ellos. En cambio, en los ambientes donde las condiciones se modifican en forma constante, los sensores que utilicemos deben poseer mecanismos electrónicos de balanceo que permitan superar estas dificultades. Si los valores devueltos por el sensor no son fiables, necesitaremos compensar este problema con programación, para lo que utilizamos varias muestras y distintos métodos estadísticos que ajusten al mínimo el margen de error. Cabe aclarar que esto se paga con mayor tiempo de procesamiento.
- **Rango de valores**: el rango de valores es el valor mínimo y máximo que puede devolver el sensor.

NARICES ELECTRÓNICAS

De todos los sentidos, es probable que el olfato y el gusto sean los más complicados de emular. Sin embargo, existen las narices electrónicas. Esos instrumentos permiten realizar análisis de una mezcla de gases, vapores y olores en tiempo real. Sirven para el control de la calidad alimenticia, el diagnóstico médico, la detección de escape de gases, procesos industriales, usos militares, etcétera.

- **Precisión**: aun si se tiene el mismo rango de valores, los sensores pueden devolver más o menos estados intermedios dentro de ese rango. Llamamos precisión a la cantidad de valores distintos que nos puede devolver el sensor en su rango de valores. Por ejemplo, no es lo mismo un sensor de brillo con un rango de 0 a 100 con una precisión de 1, que uno con una precisión de 0.1. En el segundo caso, podremos detectar variaciones mínimas que en el primero se leerían como valores idénticos.

***Figura 2. Sick** es un **escáner láser** muy usado en robótica que permite medir distancias con precisión milimétrica y realizar mapeos de objetos próximos.*

- **Velocidad de muestreo**: es la frecuencia con la que el sensor refresca su lectura del ambiente. A mayor velocidad, mejor adaptación a los cambios del mundo en el que se mueve nuestro robot. Pero por otro lado, nos exige mayor procesamiento de los valores de entrada. De todas formas, si el sensor tiene una alta velocidad y no nos da tiempo para procesar los datos, podemos desechar valores de lectura y lograr la velocidad de muestreo que necesitemos. En el caso contrario, no podemos solucionar el problema. Es decir, nos conviene un sensor con la mayor velocidad de muestreo posible, que luego regularemos al tomar todos los valores o sólo un conjunto de ellos.
- **Costo**: aunque en nuestros primeros proyectos tal vez no parezca un elemento fundamental de análisis, cuando desarrollemos robots más complejos nos encontraremos con limitaciones de este tipo. Y lamentablemente, veremos que hay una relación lineal entre el costo y las demás características: **los sensores más caros son mejores**.
- **Sencillez de uso**: en estos primeros pasos, es fundamental que tanto la interfaz electrónica que debemos desarrollar entre el sensor y el micro, como su programación para la interpretación de los datos, sea lo más sencilla posible.

- **Tamaño y forma**: en el momento en el que queramos insertar el sensor en nuestro robot, con seguridad querremos realizarle la menor cantidad de modificaciones posibles a la arquitectura que ya tenemos definida. Por otra parte, es necesario que el sensor esté a la distancia precisa para poder captar lo sensado. Por ejemplo, el sensor que utilizaremos para el seguimiento de línea es sencillo, pero con un alcance muy bajo. Por lo tanto, será necesario ubicarlo lo más cerca posible de la línea debajo del robot. Otro ejemplo son los sensores de tacto: es habitual tener que amplificar la mecánica del sensor para detectar con mayor facilidad los objetos. Para ello es fundamental elegir el tamaño y la forma que se adapten a nuestras necesidades en el conjunto de sensores que cumplen la misma función.
- **Calibración**: en este caso, también tenemos el problema de que cuanto más sencillo sea el sensor, más trabajo de calibración tendremos que hacer con el software. En general, la calibración no está brindada directamente por el sensor, sino que es en la electrónica de la conexión donde podemos agregar elementos que nos permitan modificar, con una simple vuelta de potenciómetro, las señales de entrada o la sensibilidad de lectura. Aunque le agreguemos más trabajo al diseño inicial del robot, luego agradeceremos que con un destornillador podamos readaptar al robot a diversos ambientes.

Figura 3. *Lego provee diferentes sensores, como los de tacto, luz y sonido, entre otros.*

Si tenemos en mente estos principios, podremos salir en la búsqueda del sensor que solucione nuestros problemas. Vamos a dividir nuestro desarrollo en dos grandes grupos de sensores: los digitales y los analógicos. También podemos clasificarlos como internos y externos, activos y pasivos, etcétera, pero hemos elegido la primera taxonomía porque nos determina diferencias sustanciales en la electrónica y la programación.

Sensores digitales

Los sensores digitales son aquellos que, como salida de su sensado del mundo, nos devuelven hacia el controlador un **valor discreto** (**Figura 4**). El modelo más sencillo de este tipo de sensores es aquel que directamente nos devuelve un uno o un cero. En todos los casos, necesitamos de alguna interfaz electrónica entre el sensor y el micro. A veces, esta electrónica puede ser una simple resistencia y en otros casos puede ser un desarrollo mucho más complejo, dependiendo del tipo de sensor.

Hay una gran variedad de sensores digitales en el mercado, e incluso podemos convertir sensores analógicos en digitales, como por ejemplo con el uso del integrado 74HC14. Muchos se conectan de forma similar, con la utilización de una resistencia de pull-up conectada a Vcc para mantener la señal en nivel alto, o a GND para mantenerla en nivel bajo. Cuando el sensor se activa, la señal pasa al nivel contrario.

Uno de los sensores digitales más primitivos es el interruptor o switch, que

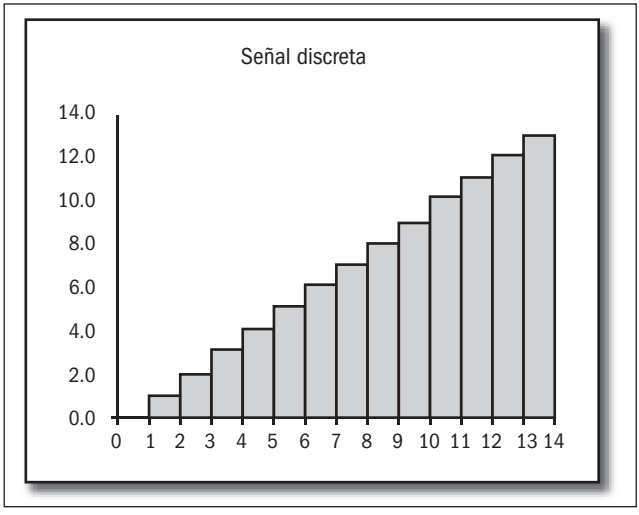


Figura 4. *Ejemplo gráfico de los valores que entrega un sensor digital.*

nos permite representar el sentido del tacto. Este sentido parece ser uno de los más simples de implementar, pero en realidad para reproducirlo en toda su expresión deberíamos sensar temperatura, contacto y fuerza (por ejemplo, no es sencillo reproducir la presión ejercida por nuestra mano cuando tomamos un objeto frágil, que se puede deformar por un excesivo nivel de fuerza). Por ahora nos ocuparemos del sensado de contacto, con la idea de detectar colisiones contra objetos para permitir el cambio de sentido de nuestro robot. Estos sensores se conocen en el mundo de la robótica como **bumpers**. En nuestro controlador ya tenemos el conexionado y la electrónica necesarios para conectar dos de estos sensores (**Figura 5**).

Estos sensores son dos interruptores sencillos (abierto/cerrado) que se conectan a +5 V con una resistencia (Pull-Up) o a GND (Pull-Down). La

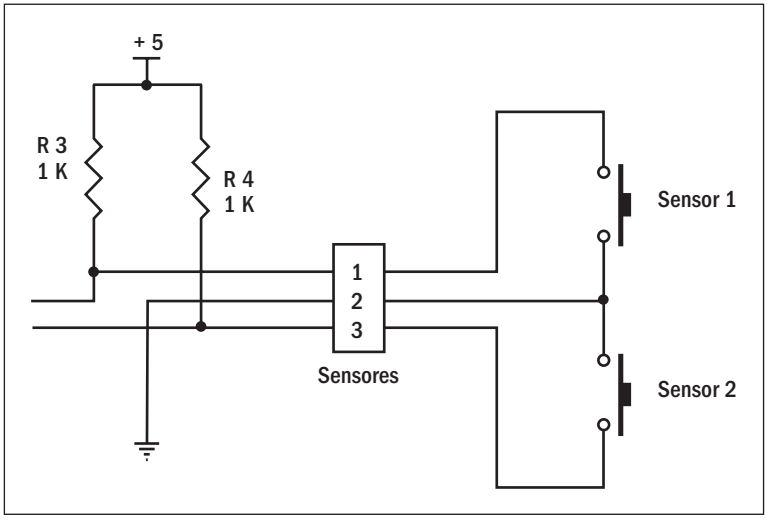


Figura 5. *Conexión de los dos interruptores en nuestro controlador.*

SENSORES DE ESTACIONAMIENTO

Muchos de los sensores que se utilizan para robots, luego desembarcan en la vida cotidiana. Un ejemplo de esto son los sensores de estacionamiento. Por poco dinero podemos agregarle a nuestro vehículo un sensor de este tipo que nos indica, con un display o con sonidos, la proximidad de los objetos en la parte trasera de nuestro auto. Sólo se activan con la marcha atrás, y nos dan información de distancia mediante luces, textos o cantidad de bips.

diferencia se producirá si queremos detectar un 0 lógico (entre 0 V y +1,5 V aproximadamente) cuando se produzca un cambio en el estado del switch o si, por el contrario, queremos detectar un 1 lógico (de +2 V a +5 V).
Otra variante es usar contactos normalmente cerrados o normalmente abiertos, que se comportarán como una compuerta NOT o directa, si consideramos el valor **TRUE** como **1** lógico, que representa +5 V en el pin del microcontrolador, producto de la acción **Chocar** y el **0** como **FALSE** en el caso contrario. Como podemos observar en el controlador, nuestra entrada está asociada en forma directa a RA3 en el caso del sensor 1, y a RA4 en el caso del sensor 2. Podemos agregarle un condensador de bajo valor (0.1uF a 1uF) en paralelo con los contactos del interruptor para reducir los efectos de rebote.

Figura 6. *Un modelo de microswitch con palanca grande.*

Figura 7. *Otro modelo con rueda en el extremo de la palanca.*

Con respecto a los interruptores que podemos utilizar, la variedad es muy grande. Uno de los más usados en robótica es el **microswitch**. Este sensor cuenta con **3 conectores**: **C** (*Common*, común), **NC** (*Normal Closed*, normalmente cerrado) y **NO** (*Normal Opened*, normalmente abierto). Cuando el sensor está en reposo, está cerrado el circuito de C a NC.
Al presionar la extensión metálica que sirve de amplificador mecánico de la señal, cerramos el circuito entre C y NO. Esta extensión metálica puede tener una palanca suficientemente grande (**Figura 6**) como para poder usarlo de sensor de tacto sin modificaciones extras. También puede contar con un pequeño rodillo en la punta que nos permita realizar tacto sobre las superficies sin provocar rozamiento entre el sensor y lo sensado (**Figura 7**).
Otro sensor que podemos utilizar en forma digital es el **infrarrojo** (**Figura 8**).

Esencialmente, consiste en un **emisor de señal infrarroja** (un led) y un **fototransistor**. Si ambos apuntan hacia el mismo lado, tenemos un **sensor reflectivo**, dado que el fototransistor será estimulado cuando la luz del led se refleje sobre una superficie. Esto es útil para detectar colores muy distintos (por ejemplo, una línea negra sobre un papel blanco) o sensar si el robot está por caer al vacío (el sensor detecta la reflectancia de la mesa, pero al acercarse a un borde deja de recibir la señal).

Uno de los problemas que podemos encontrar es que el fototransistor puede estar estimulado por las luces del ambiente. Para evitarlo, debemos encapsularlo de manera tal que sólo el reflejo de su led lo estimule. El más conocido y económico de todos es el **CNY70**, que también podemos utilizar como sensor analógico, dado que devuelve un valor entre 0 V (reflejo absoluto) hasta 3.3 V (absorción absoluta). Lo analizaremos más adelante cuando veamos los sensores analógicos.

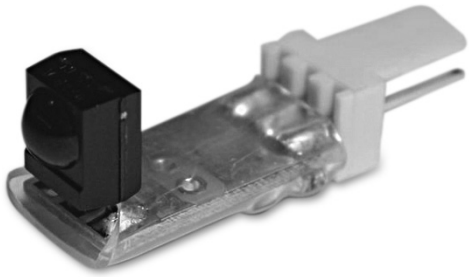


***Figura 8.** Sensor infrarrojo de tamaño reducido.*

Otra forma de utilizar el sensor infrarrojo es enfrentar el led con el fototransistor. Por lo tanto, lo que podemos detectar allí es el corte de visión entre ambos. A este tipo de arquitectura se la conoce como **sensor de ranura**. Podemos utilizarlo simplemente para la detección de objetos (como se usa en ciertos estacionamientos para avisar la salida de un auto) o para contar los pasos de una rueda asociada al eje de un motor y, de esta manera, armar un sensor de rotación. Dependiendo de cómo está acanalada la rue-

III RECICLAR SENSORES PARA NUESTRO ROBOT

En el capítulo anterior ya vimos cómo podíamos reciclar los motores de nuestros antiguos dispositivos. Ahora veamos qué ocurre con los sensores:

Disquetera 5 1/4: detector infrarrojo de ranura para detectar pista cero, fototransistores, led emisor de infrarrojo, detectores de efecto hall.

Disquetera 3 1/2: detector infrarrojo de ranura muy pequeño, microswitches.

Video VHS: led emisor de infrarrojo, receptores de infrarrojo, receptor del IR del control remoto, microswitches, infrarrojos de ranura, sensores de efecto hall.

da y de la cantidad de sensores que podamos utilizar, la detección que podemos hacer puede ir desde contar la cantidad de pasos que se han dado, hasta la codificación de la posición absoluta. Si queremos conectar un infrarrojo a nuestro controlador para detectar una línea negra en el piso o un objeto muy cercano a nuestro robot, podemos utilizar la electrónica que vemos en la **Figura 10**. Es posible ver que por un lado tenemos conectado el led infrarrojo a RA1, al que lo declararemos de salida para poder encender o apagar el led según nuestras necesidades. Si queremos que el led esté encendido todo el tiempo, directamente podemos conectarlo a 5 V. El fototransistor está en RA2, para lo cual deberemos declararlo de entrada. Queda claro que, en estos transistores, la base no está conectada, sino que es el mismo sensor el que activa la señal ante la luz.

Figura 9. *Rueda de posicionamiento absoluto con sensores infrarrojos. Con siete sensores podemos determinar 128 posiciones distintas de la rueda. En ese caso, tenemos una precisión de algo menos de 3 grados.*

Otro sensor digital interesante es el sensor de **efecto hall**. Este efecto consiste en la aparición de un campo eléctrico en un conductor ante la presencia de un campo magnético. El integrado **UGN3503** puede detectar cambios magnéticos con precisión. Tiene 3 conexiones: alimentación,

III SENSORES DIGITALES CASEROS

Es posible armar sensores digitales caseros con sólo diseñar un dispositivo que cierre o abra el circuito según la detección que se quiera hacer. Por ejemplo, para detectar unos discos de metal que se encontraban pegados al piso, cuyo borde era conductor, realizamos un sensor muy sencillo con escobillas de autitos de carrera. Sólo pusimos bajo una madera 14 pedazos de escobilla en forma paralela con una distancia menor al diámetro de los discos. Conectamos los pedazos impares entre sí y éstos a la entrada del sensor, y los pares los conectamos a masa. Cuando un disco tocaba dos de las escobillas, cerraba el circuito y lo detectaba. ¡Gracias a este sensor ganamos un campeonato de robótica latinoamericano!

tierra y salida. Si no hay campo magnético, la tensión en la salida es de la mitad de la entrada. Si se acerca el polo sur de un elemento magnetizado, aumenta el voltaje. Y si se acerca el norte disminuye. Si queremos utilizarlo de esta manera, deberemos prestar atención cuando veamos cómo usar sensores analógicos. Pero si sólo queremos detectar la presencia o no de un elemento magnético, nos alcanza con un procesamiento digital.

Programación de los switches de nuestro controlador

A continuación presentamos un programa muy sencillo que nos permite encender el led cuando detecta la activación del sensor de tacto 1 que tenemos en nuestro controlador.

```
program Sensorescapitulo7

main:
     TRISA = 255
      ' Configura los pines
     de PORTA como entrada
     TRISB = 0
      ' Configura los pines
     de PORTB como salida
     while true
           if PORTA.3=1 then
      ' Verifica el estado del
     pin donde está conectado
```

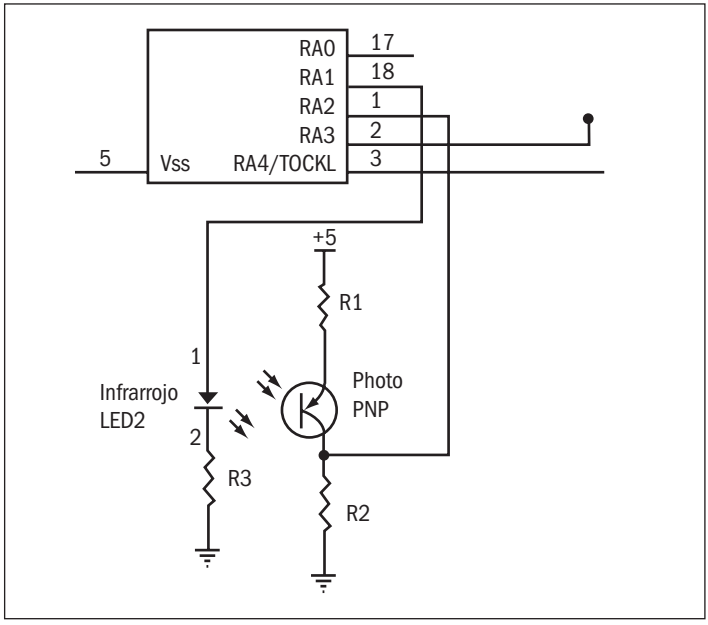


Figura 10. *Circuito básico de conexión del infrarrojo a nuestro controlador.*

```
      el sensor 1
               PORTB.3=1
      ‘ si está encendido
      (el sensor activado)
      enciende el led
            else
                PORTB.3=0
‘ si no, lo apaga
            end if
    wend
end.
```

Los sensores analógicos

A medida que nuestro robot evoluciona, nos encontramos con la necesidad de captar el ambiente con mayor precisión. Ya no nos alcanza con uno o más sensores digitales. Queremos recibir un valor dentro de un rango. A los sensores que nos permiten este nivel de precisión se los conoce como analógicos (**Figura 11**). Por ejemplo, una fotorresistencia nos puede entregar un valor entre 0 y 5 V.

Ahora, el problema que tenemos es que, a pesar de todo lo bueno que pueda ser nuestro micro, siempre trabaja en forma digital. Por lo tanto, necesitamos convertir esta señal analógica en un valor digital. Para ello existen los **conversores analógicos/digitales** (A/D). Estos dispositivos, a partir del valor de entrada, nos devuelven un número de 8, 10 ó más bits. Veamos una tabla de un supuesto conversor A/D que provee una salida de 4 bits. Esto significa que tenemos 16 estados distintos posibles de salida. Si la entrada es un valor entre 0 V y 5 V, el esquema podría ser el que vemos en la **Tabla 1**.

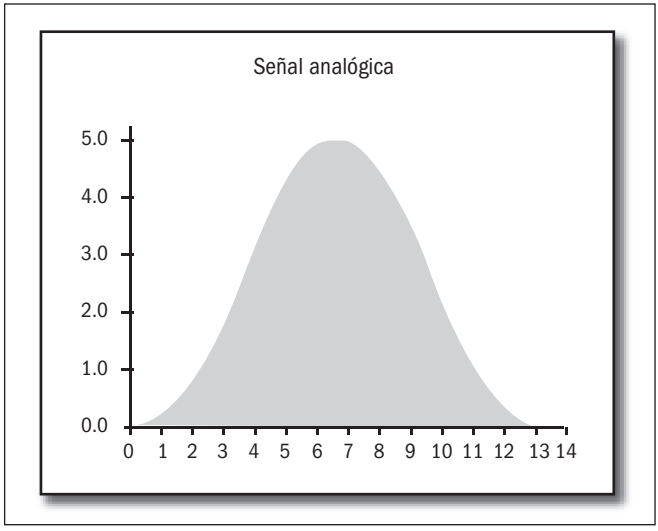


Figura 11. *Gráfico ejemplo de los valores que entrega un sensor analógico.*

VOLTAJE ENTRADA		SALIDA	
MIN	MAX	BINARIA	DECIMAL
0	0,3125	0000	0
0,3125	0,625	0001	1
0,625	0,9375	0010	2
0,9375	1,25	0011	3
1,25	1,5625	0100	4
1,5625	1,875	0101	5
1,875	2,1875	0110	6
2,1875	2,5	0111	7
2,5	2,8125	1000	8
2,8125	3,125	1001	9
3,125	3,4375	1010	10
3,4375	3,75	1011	11
3,75	4,0625	1100	12
4,0625	4,375	1101	13
4,375	4,6875	1110	14
4,6875	5	1111	15

***Tabla 1.** Relación de valores en la conversión de una señal analógica entre 0 V y 5 V a una señal digital de 4 bits.*

Existe una inmensa variedad de conversores A/D. Habitualmente, se usan de 8 bits con los micros, como por ejemplo el **ADC0804** (**Figura 12**). Este conversor acepta una entrada de 0 V a 5 V.

En su pin 20 alimentamos con 5 V, en el 9 con 2.5 V (la mitad exacta de lo que se quiere medir) y en el pin 7 con 0 V. En los pines 8 y 10 se conecta a tierra.

En el pin 6 tenemos la entrada analógica y en los pines 11 a 18 tenemos los 8 bits de la salida digital, que en nuestro caso deberíamos conectar, por ejemplo, con RB0 a RB7 (¡nos

HOJAS DE DATOS

Si buscamos información sobre algún integrado, seguramente llegaremos a su respectivo **datasheet** (hoja de datos). Cada fabricante tiene la hoja de datos de sus propios integrados, pero tenemos sitios que juntan y clasifican las hojas más comunes:

- Universidad Nacional del Sur: **www.ceia.uns.edu.ar/integrados/index.asp**.
- Catálogo completo de hojas de datos: **www.datasheetcatalog.net/es/**.

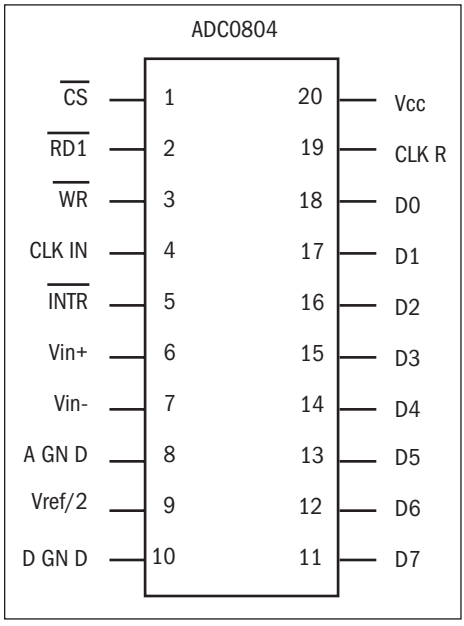


***Figura 12.** Esquema del conversor A/D ADC0804.*

consume un montón de entradas un único sensor!). Para que inicie la conversión, debemos enviar al pin 3 primero un 1 y luego un 0, y al pin 2, al mismo tiempo, lo contrario. Allí toma la entrada analógica y realiza la salida digital hasta que se vuelvan a producir las activaciones de los pines 2 y 3 ya mencionadas. Como vemos, tener un conversor A/D nos agrega algunas cuestiones electrónicas y nos consume una cantidad importante de entradas digitales. Pero la electrónica avanza, y es aquí donde podemos encontrar la mayor diferencia (para este libro) entre el 16F84 y el 16F88. Este último micro de Microchip, ¡tiene un conversor A/D incorporado!

En el 16F88 tenemos siete entradas analógicas (siete canales), asociadas las cinco primeras a RA0 a RA4 y las últimas dos a RB6 y RB7, respectivamente. Cada una de ellas nos proporciona un resultado digital de 10 bits, es decir, un valor entre 0 y 1023. Las referencias positivas y negativas (Vref+ y Vref-) son seleccionables mediante RA2 y RA3.

Los registros de control son:

- **ADRESH** y **ADRESL**: parte alta y baja del resultado, que en nuestro caso, con mikroBasic, podremos obtener de manera directa con la llamada a la función **ADC_read**.

III CONVERSIÓN ANALÓGICO-DIGITAL

Aunque parezca un tema novedoso y desconocido, vivimos rodeados de conversiones de este tipo. Por ejemplo, la música en CD es una conversión a lo digital de las señales sonoras, que son analógicas. Cuando grabamos con un micrófono en la computadora, cuando escaneamos una imagen, cuando sacamos una foto digital, convertimos el mundo analógico real en un formato digital. En todos los casos, necesitamos pasar del infinito continuo al finito discreto.

- **ADCON0** y **ADCON1**: registros de control que describiremos más adelante.
- **ANSEL**: registro de selección de entradas analógicas. También resuelto por mikroBasic.

Los pasos de operación del ADC son complicados. Por suerte, como ya comentamos, mikroBasic nos proporcionar una solución. Para operar el ADC desde mikroBasic, en primer lugar debemos inicializar el registro de control **ADCON1**. En este caso, debemos poner un valor que determine los cuatro bits más significativos (del 4 al 7). Los bits 4 y 5 determinan los valores de referencia según la siguiente tabla:

VALOR	VREF+	VREF-
00	AVdd	AVss
01	AVdd	Vref- (RA3)
10	Vref+ (RA2)	Avss
11	Vref+ (RA2)	Vref- (RA3)

***Tabla 2.** Valores a ser seteados en ADCON1 para definir los valores de referencia Vref+ y Vref- en la conversión A/D.*

En el caso de usar RA2 y RA3, deben ser configurados como entradas analógicas. El bit 6 determina si la fuente del reloj de la A/D se divide por 2 cuando se usa el reloj del sistema (habitualmente lo ponemos en 0). El bit 7 determina cuáles son los bits que quedarán en 0 en el valor

III SENSORES ULTRASÓNICOS

Émulos del mecanismo de ubicación de los murciélagos, los sensores ultrasónicos nos permiten medir distancias a bajo costo. Pueden tener el emisor y el receptor por separado o utilizar el mismo piezoeléctrico para ambas cosas. Podemos conseguirlos en algunas cámaras antiguas con foco automático, o utilizar la serie SRF de Devantech.
El más adecuado en este caso es el SRF08, que funciona con 5 V y tiene bajo consumo.

word que devuelve la lectura digital. Como sabemos que son 10 bits que quedarán en **ADRESH** y **ADRESL**, entonces, si ponemos este bit en 1, los 6 bits que quedan en 0 son los más importantes, es decir, los 6 más significativos de **ADRESH**. En el caso contrario, serán los 6 menos significativos de **ADRESL**.

Como paso siguiente, debemos indicar los pines que serán de entrada en A con **TRISA** o en B con **TRISB**. Para finalizar, leemos el valor del canal donde tengamos la entrada con la función **Adc_Read(nro de canal)**, que devuelve un valor de tipo **word**.

Cabe mencionar que en el llamado a esta función se realizan muchos pasos más que, por suerte, no nos

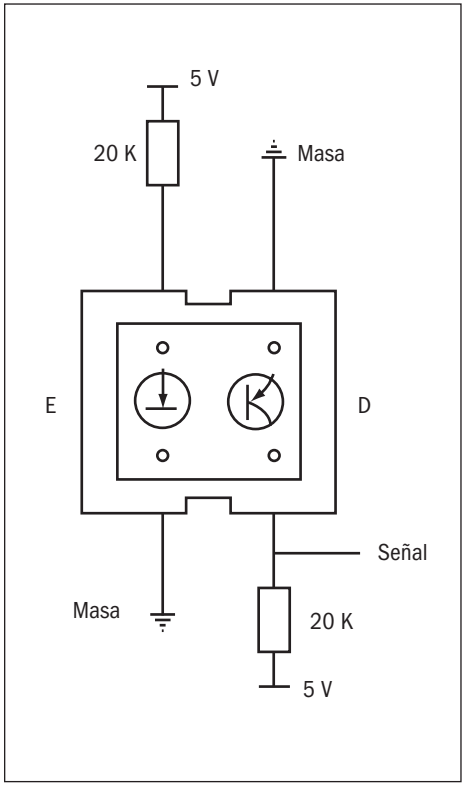


***Figura 13.** Conexión del CNY70 para usarlo como sensor analógico.*

deben preocupar, ya que los realizará en forma automática.
Con estos pasos, podemos tomar los valores de los sensores que describiremos a continuación, y que nos brindarán la riqueza de la precisión que los sensores digitales no tenían.

Tipos de sensores analógicos

Uno de los sensores más sencillos de utilizar y más económicos es el famoso **CNY70**, que ya hemos mencionado. Para usarlo, sólo lo conectamos como indica la **Figura 13**, donde la señal debe ir hacia la entrada analógica.

Otro sensor conocido y económico es el **LDR**, que consiste en una resistencia que varía su valor resistivo en función de la luz que recibe. De esta manera, podemos obtener información compleja del entorno donde el robot realiza su tarea. También podemos agregarle un led específico del color que deseamos detectar, para que los valores de rebote sean más significativos. En la **Figura 14** podemos ver la conexión del sensor a nuestro controlador, y más adelante veremos cómo programar la detección de su valor analógico.

PIEL PARA ROBOTS

El gran avance de la tecnología nunca terminará de sorprendernos. Unos científicos japoneses desarrollaron una piel artificial flexible con capacidades sensoriales. Por ahora sólo son sensibles a la presión, pero en breve tendremos pieles que además detecten temperatura, fuerza, tensión, etcétera. Esto permitiría una manipulación mucho más fina de los objetos por parte de un robot, sin necesidad de hacer procesamiento de imágenes para lograrlo.

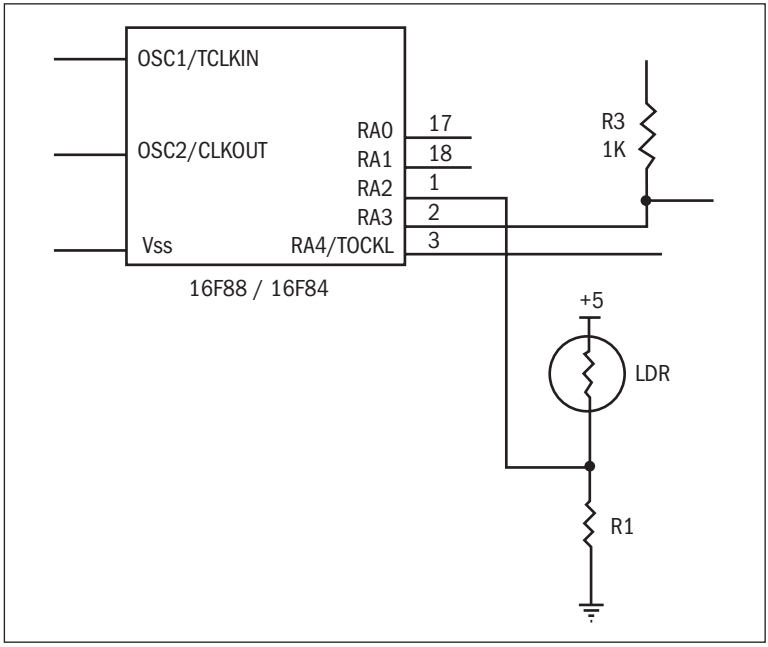


Figura 14. *Conexión del LDR a nuestro controlador.*

Otro sensor analógico corriente es el **potenciómetro**. Conectado a un motor, lo podemos utilizar como sensor de rotación, principalmente si es multivuelta.

En la **Figura 15** podemos ver el esquema de conexión.

Otros sensores analógicos que podemos encontrar en el mercado son:

- De luz o color: fotodiodos, fototransistores y CCD (integrado que posee una matriz de celdas que son sensibles a la luz).
- De presión.
- Táctiles: a diferencia de los sensores de tacto, pueden determinar el grado de presión y la zona donde se produce el contacto.

DETECTOR DE MENTIRAS

Si queremos usar nuestro robot para combatir el crimen organizado, además de ponerle una capa y un antifaz, podemos agregarle un sensor que nos permita detectar mentiras. Este sensor es, simplemente, un medidor de la resistencia que tenemos en la piel, que se modifica cuando transpiramos. Supuestamente, cuando mentimos, sudamos, y por lo tanto cambiamos esta medida, aunque a veces nos parece que hay gente que miente sin que se le mueva un pelo.

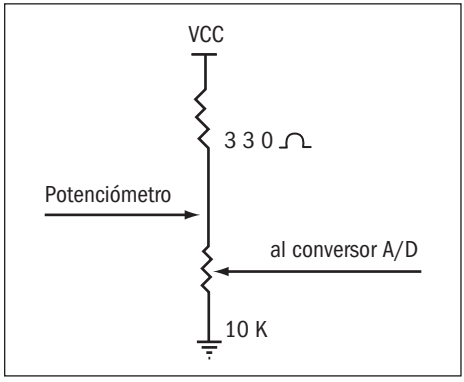


Figura 15. *Conexión del potenciómetro a nuestro controlador.*

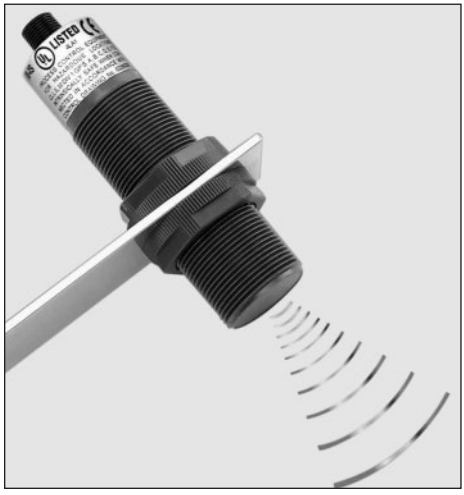


Figura 16. *Uno de los tantos* ***sensores ultrasónicos*** *que hay en el mercado.*

- Micrófonos.
- Ultrasónicos: permiten medir distancias con el mismo mecanismo que utilizan los murciélagos (**Figura 16**).
- Acelerómetros: detectan aceleración y vibración.
- Inclinómetros: sensan la posición vertical.
- Termistores: para medir temperaturas.
- Termorresistencias: de la misma manera que los termistores, permiten medir temperaturas, pero en condiciones mucho más extremas.
- Pirosensores: divisan fuego por su capacidad para detectar ciertos rangos de ultravioletas.
- Humedad.
- GPS.
- Proximidad.

Ejemplo de programación de la lectura de un LDR en una entrada analógica

Este ejemplo es una de las pocas cosas que no podremos hacer si tenemos un 16F84, y es por eso que recomendamos hacer el pasaje de micro. En el ejemplo que dimos, teníamos conectado el LDR en RA2. Por lo tanto, tendremos que leer el canal 2.

```
program LDR

dim resu as word

main:
  ADCON1 = $80   ' configuro
     la forma de lectura del
     valor digital
                 ' y que utilizaré
                 Vss y Vdd como
                 valores de
                 referencia.
```

```
TRISA  = $FF  ' defino PORTA
   como entrada
TRISB  = $00  ' defino PORTB
   como salida
while true
    resu = ADC_read(2)
    if resu>300 then 'Si el
          valor del ldr es
          mayor a 300
          PORTB.3=1
             'Enciendo el led
       else
          PORTB.3=0
             'Y si no, lo apago
       end if
  wend
end.
```

... RESUMEN

Los sensores son uno de los núcleos fundamentales de los robots autónomos. Sin embargo, presentan diversas dificultades, desde la conexión electrónica hasta el procesamiento de la información que nos entregan. En este capítulo hemos visto las características fundamentales de un sensor a la luz que debemos analizar según el objetivo de nuestro robot. En el conjunto de sensores, tenemos los digitales, que nos devuelven valores discretos, habitualmente, 1 y 0. Entre estos sensores, de fácil procesamiento, podemos encontrar microswitches y sensores de luz. También tenemos sensores analógicos, que devuelven un valor dentro de un rango continuo. Dado que nuestros procesadores trabajan en forma digital, debemos pasar este valor de entrada por un conversor analógico digital. Afortunadamente, varios micros ya proveen en forma interna ese conversor, como el 16F88 que recomendamos en este libro.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Cuáles son las características esenciales de los sensores?

2 ¿Qué es un sensor digital? ¿Cómo convertimos un sensor analógico en uno digital?

3 Describa los interruptores o microswitches.

4 ¿Cuáles son los conectores de un microswitch y cómo se conectan?

5 ¿Cuáles son los tipos de sensores infrarrojos que podemos utilizar según su arquitectura?

6 Describa los sensores de efecto hall.

7 ¿Qué es un sensor analógico? ¿Qué es un conversor A/D?

8 ¿Cómo se utiliza el conversor ADC0804?

9 ¿Cómo funciona el conversor A/D del 16F88?

10 Describir cuatro tipos de sensores analógicos.

EJERCICIOS

1 Realizar un programa donde el led se encienda si los dos interruptores están en el mismo estado (los dos presionados o sueltos).

2 Realizar un programa donde el led se encienda cada 3 pulsaciones. Es decir, si se pulsa una vez o dos veces no se enciende. A la tercera pulsación se enciende. Luego se apaga en la cuarta y quinta, y vuelve a encenderse en la sexta.

3 Conecte ocho leds al puerto B del micro y un potenciómetro a una entrada analógica. Realice un programa que encienda los leds según la salida digital del valor del potenciómetro (es decir, que los leds representen los 8 bits menos significativos de la salida).

4 Conecte un CNY70 y programe el micro para que el motor 1 se encienda y el 2 se apague si el sensor está sobre una línea negra, y al revés si está sobre una superficie blanca.

El cuerpo del robot

Llegó la hora de armar una estructura que permita optimizar el funcionamiento del robot. Para eso buscaremos un equilibrio entre el peso, la funcionalidad, la estabilidad y el tamaño. Analizaremos las arquitecturas posibles y nos detendremos en los mecanismos de locomoción de los robots. Ubicaremos los sensores para captar el ambiente, y dejaremos todo listo para los próximos capítulos.

CUERPO A CUERPO

Nuestra criatura está creciendo. Es inteligente, se mueve (aún en forma torpe), puede detectar el mundo que la rodea. Pero sabemos que no estaremos a su lado toda la vida, y queremos preparar a nuestro robot para vencer distintos tipos de dificultades: seguir una línea, detectar las luces encendidas y apagadas, jugar al fútbol con sus compañeritos, luchar con ellos ante un problema.

El armado incompleto que tiene en este momento es demasiado frágil para afrontar todos estos desafíos. Por eso, necesitamos acomodar todas sus piezas y darle una arquitectura que sea fuerte, robusta y veloz. Pero como siempre, tenemos muchísimas opciones: ¿ruedas, orugas o patas? Y si son ruedas, ¿diferenciales u omnidireccionales? Y si son patas, ¿dos, cuatro, seis, cien?

Como vemos, no es fácil decidir la estructura de nuestro robot, y esta decisión está fuertemente vinculada con su destino final. En nuestro caso, haremos un equilibrio entre el costo y la estructura, y tendremos en mente la posibilidad de encarar diversos desafíos con la misma forma. Pero como ya hemos comentado, el que mucho abarca, poco aprieta.

El cuerpo que vamos a desarrollar nos permitirá realizar muchas funciones distintas, pero en ningún caso será la mejor estructura posible. Es por eso que presentaremos diver-

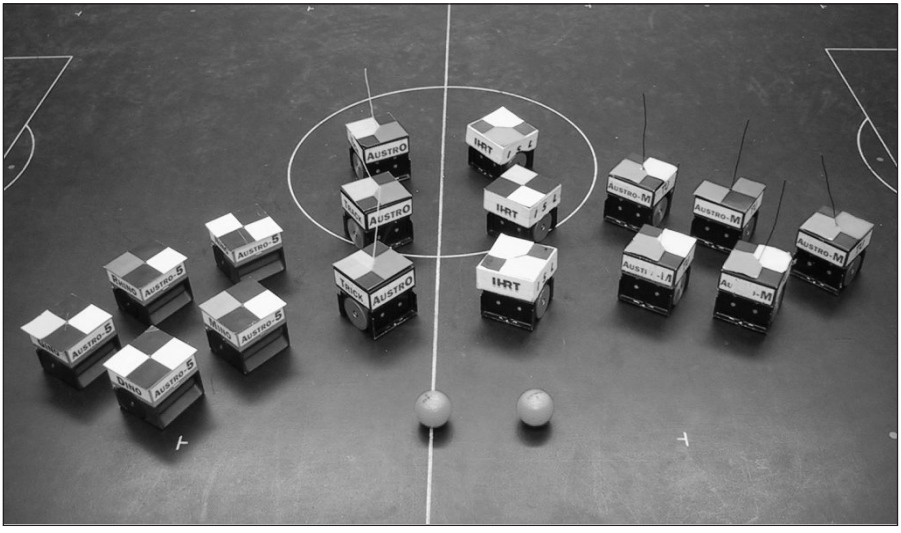


Figura 1. *Dos equipos formados para comenzar un partido de fútbol de siete contra siete.*

sos aspectos de la construcción de manera tal que más adelante, se pueda desarrollar el robot que mejor se adapte a la tarea final.

Características mecánicas de un robot autónomo

Cuando desarrollamos la estructura de un robot, tenemos que tener en cuenta un conjunto de variables que, en muchas ocasiones, se contraponen. Desde ya, la variable por excelencia, lo que tenemos que tener en mente constantemente, es **el objetivo o la misión** del robot. En nuestro caso, dado que queremos cumplir diversas pruebas con el mismo robot, tenemos que elegir una estructura que nos permita acceder con facilidad a los sensores y cambiarlos de lugar con sencillez, entre otras cosas. Pero como nuestro objetivo es que luego de leer este libro podamos comenzar a realizar diferentes diseños de robots, a continuación presentaremos cuáles son las características que debemos tener en cuenta en el desarrollo.

- **Ubicación de las baterías**: ya hemos comentado que éste es un tema especial y molesto. Si en nuestro robot agregamos la electrónica de la carga de la batería, podemos ubicarla en algún lugar inaccesible mientras podamos enchufar el cable de alimentación. Pero si no agregamos esto, necesitamos ubicar las baterías en algún lugar donde podamos sacarlas y ponerlas sin tener que desarmar todo.
- **Ubicación del controlador**: tenemos un problema similar al de las baterías. Si desarrollamos la programación en circuito (como es nuestro caso), basta con dejar el conector para comunicarnos con el programador. Pero si tenemos el micro en un zócalo, tendremos que poder acceder a él para llevarlo al programador y regresarlo.
- **Tipos de locomoción**: otro aspecto fundamental es el ambiente donde el robot debe desarrollar la actividad. Son tantas las modificaciones que debemos tener en cuenta, que trataremos este tema en forma detallada más adelante.

CÓMO UBICAR LAS PILAS

Para poder ubicar las pilas de la forma que más nos convenga, podemos comprar los portapilas en diferentes configuraciones. En general, existen portapilas de cuatro pilas, pero podemos conseguir de dos, de seis y de ocho. Una vez que tengamos determinada la configuración que nos será más útil para nuestro robot, no tenemos que olvidar conectarlos entre sí en serie para obtener un negativo y un positivo con el voltaje necesario.

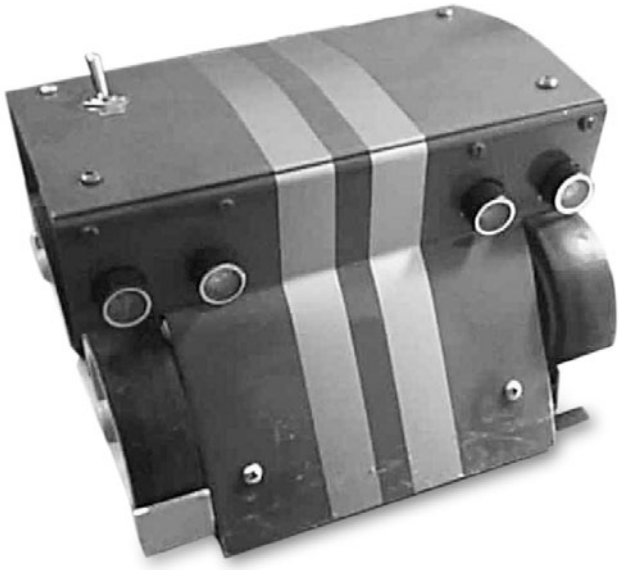


Figura 2. *La arquitectura del robot dependerá de su objetivo. Aquí tenemos un robusto luchador de sumo.*

- **Equilibrio del robot**: cada ambiente en particular añade características específicas a la actividad. Por ejemplo, uno de los problemas habituales son las rampas o los obstáculos. En el caso de las rampas, tenemos que analizar con atención si la potencia de los motores nos permitirá subirlas o bajarlas sin caernos o quedarnos estancados. Y esto no hay que hacerlo mediante el encendido de los motores y luego colocar al robot al comienzo de la rampa para ver si la sube, dado que si debe sensar para poder subir, el comportamiento de los motores será mucho más complejo y con seguridad no alcanzaremos la potencia necesaria. Al probarlo es necesario tener en cuenta todos los problemas que se encontrará en el trayecto de subida. Lo mismo ocurre con los obstáculos. Puede ocurrir que nuestro robot supere uno de ellos cuando lo ponemos directamente frente a él, pero si llega inclinado o mientras analiza los sensores, es posible que fracase en su intento. Es por eso que el verdadero comportamiento lo vamos a obtener en la arena real. Una buena idea es construir un espacio lo más parecido posible al que va a tener el robot como ambiente, para poder testearlo en forma robusta.
- **Superficie donde se desarrolla la**

Figura 3. *La superficie de la cancha determinará el tipo de ruedas a utilizar.*

actividad: el tipo de superficie también genera problemas con respecto al comportamiento inercial del robot y su adherencia. Tendremos que detenernos a analizar cómo son nuestras ruedas, de qué material están hechas, si no es mejor utilizar orugas, si son de goma, de metal, etcétera. También en esta oportunidad es más conveniente tener el material real de la superficie, porque el comportamiento cambia completamente. ¡Y no nos podemos dar el lujo de tener juegos de botines completos!

- **Ubicación de los sensores**: en el capítulo de los sensores hemos comentado algo sobre esto. Según el tipo de sensor, será necesario ubicarlo

{ } TESTEO DEL ROBOT

Para poder observar el comportamiento inercial del robot, es necesario que lo probemos en los siguientes estados:

- A toda velocidad hacia delante y hacia atrás, y luego frenar bruscamente.
- A toda velocidad hacia delante y hacia atrás, y luego dejar las ruedas sin alimentación.
- Girar a toda velocidad para un lado y para el otro y luego hacerlo frenar de manera brusca.
- Girar a toda velocidad para un lado y para el otro y luego dejar las ruedas sin alimentación.

más cerca o más lejos de los objetos que tendrá que detectar. Por ejemplo, si usamos un CNY70 para seguir una línea, prácticamente lo tendremos que pegar al piso. Por otro lado, también es necesario que la ubicación nos permita aislarlo lo más posible de las interferencias del ambiente. Por ejemplo, los sensores de luz de cualquier tipo tienen que estar lo más aislados posible de la luz ambiente para poder tener uniformidad.

- **Ubicación de los motores y de la transmisión**: éste es otro problema importante, en especial si nuestro presupuesto no nos ha permitido comprar micromotores o motorreductores. La ubicación de los motores nos complicará el mecanismo de transmisión hasta las ruedas. Además, cualquier movimiento o vibración de los motores no contemplado, comenzará a deteriorar el sistema de transmisión. Tanto los motores como los ejes donde se asiente la transmisión deben estar robustamente anclados para evitar estos problemas.
- **Ubicación del sistema de comunicación**: si tenemos el placer de añadir un sistema de comunicación inalámbrico, tenemos que ubicarlo en algún lugar donde no tengamos interferencias y donde podamos cambiar de canal con facilidad si es necesario (por ejemplo, en el caso de uso de radio).
- **Tamaño y peso**: todo lo que hemos descripto también está acotado por estos dos factores, que se presentan como límites habituales en las competencias de robots. Tal vez el más sencillo de solucionar es el peso, ya que si modificamos los materiales de la estructura del robot, lo podemos lograr. Pero con respecto al tamaño, muchas veces, si no hemos previsto con antelación el problema, estamos obligados a desarmar el robot en forma casi completa.

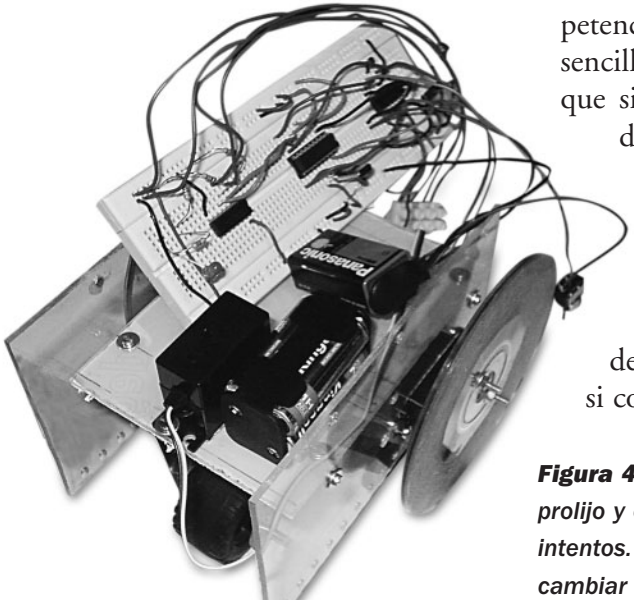


Figura 4. *No siempre nuestro robot es prolijo y ordenado en los primeros intentos. Lo importante es que podamos cambiar su configuración con facilidad.*

Todo lo que hemos descripto se aplica a robots terrestres. ¡Imaginemos todos los problemas que surgen en los aéreos y submarinos! Como el nivel de complejidad de estos robots supera nuestro humilde libro, comentaremos de forma muy simple algunas cuestiones de los robots de este tipo, para luego profundizar en los terrestres.

Robots aéreos

Uno de los problemas fundamentales que podemos encontrar en el desarrollo de estos robots es la necesidad de obtener información del ambiente con una frecuencia mucho mayor que en el caso de los robots terrestres. En estos casos, el robot no puede hacer nada y debe quedar en estado de reposo. Necesariamente tiene que recibir información de su movimiento en 3D y adaptar la velocidad de sus motores y posicionamiento de sus alas o rotores en forma constante.

En general, utilizan un sistema redundante de sensores para su posicionamiento, de forma tal que ante cualquier interrupción momentánea de uno o más de ellos, pueda recuperarse de la pérdida. Por otra parte, con esta redundancia se logra mayor fiabilidad y precisión en la información. En general utilizan **GPS** (*Global Positioning System*, Sistema de Posicionamiento Global), acelerómetros, giróscopos, telémetros y otros sensores so-

***Figura 5. Marvin MarkII**, un robot aéreo autónomo desarrollado por el Departamento de Ciencias de Computación de la Universidad Técnica de Berlín.*

fisticados para poder ubicarse en 3D. Los robots aéreos más versátiles son los helicópteros, pero también se han desarrollado aviones y zeppelines, que son más estables y fáciles de controlar.

Otro problema fundamental es tener información constante en tierra sobre la batería restante o algún mecanismo automático de aterrizaje en caso de que nos quedemos sin energía. En los robots terrestres, si nos quedamos sin batería, dejarán de funcionar. En los aéreos, las consecuencias son imaginables, ¿no?

Robots subacuáticos

Los problemas planteados por este tipo de robots son similares a los aéreos. Tenemos movimiento 3D, pero en este caso, tenemos mayor estabilidad en el medio. Es decir, podemos usar más tiempo para tomar decisiones, dado que a lo sumo nos sumergiremos más o menos cuando no hacemos nada. Aquí las dificultades surgen en los tipos de sensores a usar, dado que el agua es un medio más complejo e inestable. Podemos encontrar partículas suspendidas que modifican todo tipo de señal. También la comunicación se hace más compleja, con lo cual el riesgo de perder el robot es más grande (¡no se rompe pero se pierde en el mar!).
Con respecto a las baterías, el problema es similar al caso aéreo. Para sumergirse utiliza sistemas de flotabilidad que, con la entrada o la salida de agua, permiten modificar la profundidad del robot. Con respecto al movimiento, con motores desplaza el agua que lo circunda.

Robots terrestres

Dentro del conjunto de robots terrestres, tenemos diversos tipos de formas de locomoción. Cada una de ellas posee ventajas y desventajas, y depende del tipo de superficie que recorramos. Vamos a describir genéricamente cada una de ellas para detenernos, más adelante, en los sistemas con ruedas, que son los que podremos implementar en nuestro robot.

III COMPETENCIA INTERNACIONAL DE ROBOTS AÉREOS

La Asociación Internacional de Vehículos Autónomos (AUVSI) organiza competencias de robots aéreos autónomos. En cada una de ellas se plantea una misión sobre un ambiente determinado, como un rescate en el mar o el ingreso a una zona donde se produjo un desastre nuclear para tomar imágenes. El objetivo es volar hasta una zona distante a tres kilómetros del punto de partida e identificar una estructura en particular. Podemos encontrar la información completa en el sitio **http://avdil.gtri.gatech.edu/AUVS/IARCLaunchPoint.html**.

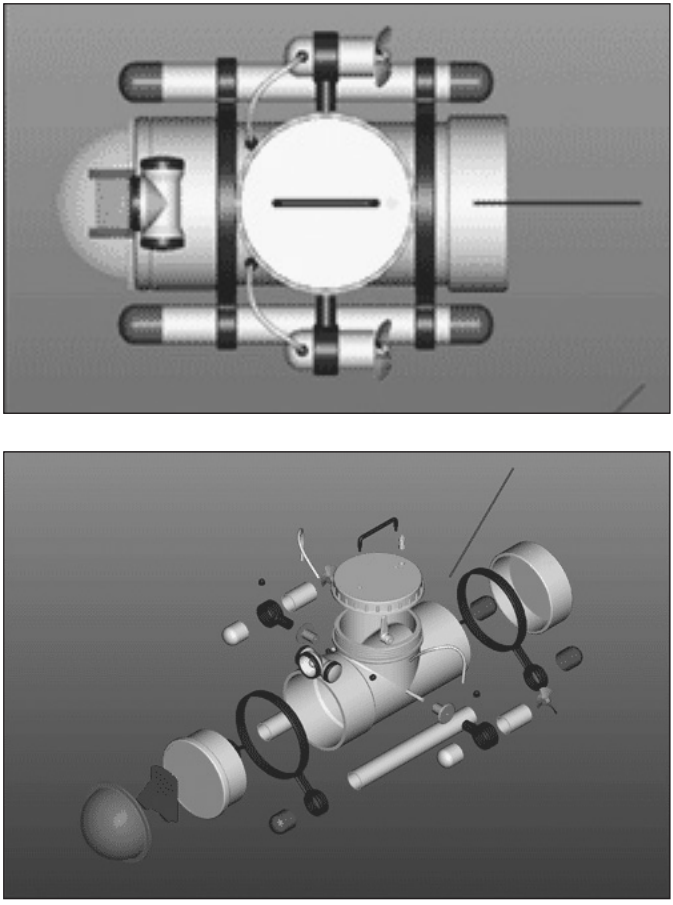


Figura 6. *Explorador subacuático realizado por alumnos de una escuela media de Argentina. No es autónomo, pero resuelve todas las dificultades mecánicas de un robot submarino.*

{ } ROBOT SUBACUÁTICO HECHO POR ALUMNOS DE UNA ESCUELA

Ex.Su.Ac. (Explorador SubAcuático) es un robot desarrollado por alumnos de la Escuela Técnica No. 3 de Buenos Aires. El robot se realizó con la utilización de productos prefabricados de PVC y puede sumergirse hasta 15 metros de profundidad. Tiene como objetivo colaborar en la búsqueda de problemas de residuos en tanques y conductos de agua, en el rescate de cuerpos u objetos sumergidos, y en la generación de mapas del suelo subacuático. Podemos encontrar más información en **www.oni.escuelas. edu.ar/2005/GCBA/972/**.

Orugas

Los sistemas de oruga unen la rueda trasera con la delantera (**Figura 7**). Tienen como ventaja una mayor adherencia a la superficie y la capacidad de adaptarse a terrenos irregulares. En general son de goma, pero también podemos hallar orugas metálicas que le permiten clavarse en superficies demasiado pulidas. La desventaja que presentan es la imposibilidad de realizar trayectorias curvas. La única forma de rotar es si se giran las orugas de cada lado en sentido opuesto, lo que logra un giro en el lugar, pero el avance sólo puede ser en línea recta.

Patas

Los robots con patas son el resultado de un enfoque biológico de la arquitectura, y van de la mano del sueño de la criatura humana (**Figura 8**). También tienen como ventaja una mayor adaptación a terrenos complejos, pero eso aún no se ha logrado. El sistema de equilibrio en estos robots es complicado, y necesitan de una coordinación muy precisa de los mecanismos de control de los motores.

CUBE REVOLUTIONS

Este proyecto de la Universidad Autónoma de Madrid se basa en la construcción de un robot ápodo y modular. El robot está compuesto por 8 módulos idénticos con la misma orientación, y puede adoptar diversas formas. Logra su locomoción por medio de ondas periódicas, por semiondas o al convertirse en una rueda que rota sobre sí misma. El desarrollo del robot es completamente abierto y libre, y fue diseñado con herramientas de las mismas características. El control se realiza desde una PC mediante el puerto serial. Podemos encontrarlo en **www.iearobotics.com/personal/juan/doctorado/cube-revolutions/index.html**.

En general, presentan 3 grados de libertad en cada pierna: cadera, rodilla y tobillo. En las dos ligas internacionales de fútbol de robots encontramos categorías de robots con patas, pero aún no juegan verdaderos partidos, sino pruebas de destreza. Una forma de aumentar el equilibrio consiste en incrementar el número de patas. En este sentido, los

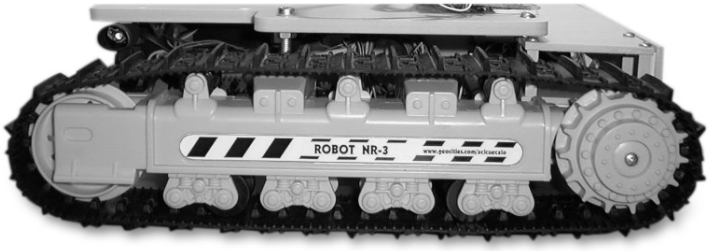


Figura 7. *Robot **NR3** con sistema de orugas desarrollado por Francisco Carabaza Piñeiro.*

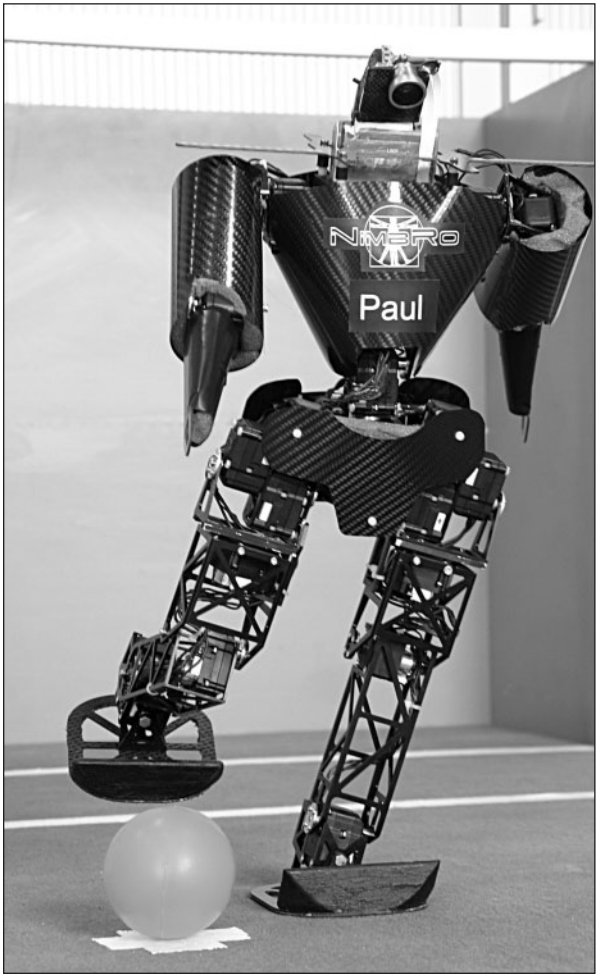


***Figura 8. Nimbro**, humanoide participante de la RoboCup 2006, desarrollado por la Universidad de Freiburg.*

JUGADORES DE FÚTBOL CON PATAS

En todas las competencias de fútbol de robots encontramos categorías de jugadores con patas. En el caso de la Robocup, están presentes las categorías **Four-legged** (cuatro patas), donde cada equipo está formado por cuatro Aibos, y **Humanoid** (humanoide), donde el equipo está compuesto por un arquero y un jugador, ambos con dos patas. En la FIRA, la categoría se llama **HuroSot**, y los humanoides no juegan partidos, sino que realizan pruebas de destreza.

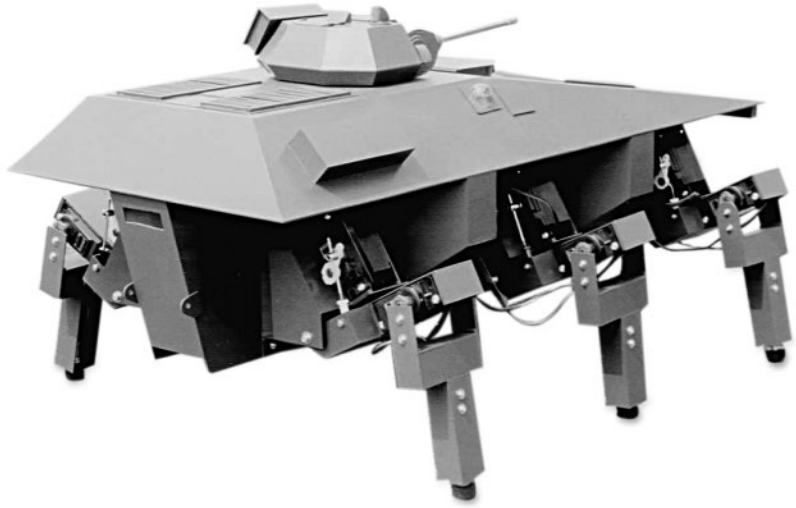


Figura 9. *Robot* ***hexápodo*** *desarrollado por un aficionado en Japón, con el uso de ¡dieciocho servos!*

esquemas tradicionales son bípedos, cuadrúpedos y hexápodos (**Figura 9**). Los ejemplos más conocidos son **Asimo** de Honda y **Qrio** de Sony en el caso de los bípedos, y **Aibo** de Sony en el caso de cuadrúpedos.

Ápodos

Los ápodos son robots que no emplean ruedas, orugas o patas (**Figura 10**). El modelo a seguir es el de los gusanos o serpientes. Están compuestos por partes pequeñas, articuladas en forma completamente modular. Pueden moverse sobre superficies muy diversas y tomar configuraciones distintas para entrar en lugares complicados. La idea es diseñar un módulo básico y crear diferentes tipos de robots ápodos al conectar varios de estos módulos.

El movimiento en línea recta se logra con un efecto de onda a partir de la cola. Comienza con una contracción inicial que se propaga en cada módulo. En esta arquitectura, lo complejo es la coordinación de los módulos. Además, necesitamos sensores en los módulos para poder adaptarlos a distintos terrenos. Otro problema complejo es agregarles grados de libertad a los módulos para obtener movimientos en superficies y no sólo en línea recta. En general, se usa al intercalar módulos en diferentes fases (uno hacia arriba y otro hacia el costado).

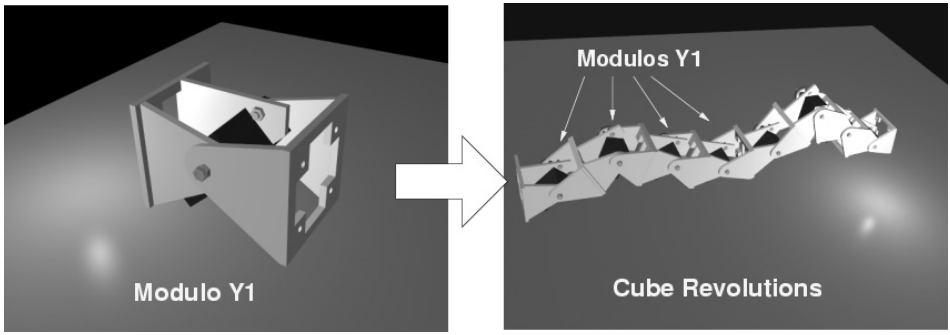


Figura 10. *Robot ápodo desarrollado por la Escuela Politécnica Superior de la Universidad Autónoma de Madrid.*

Sistemas con ruedas

Los robots con ruedas son los más económicos y sencillos de implementar. De todas formas, hay diversas arquitecturas que analizaremos detenidamente, y pueden tener alta complejidad. En ellas, uno de los problemas fundamentales que también trataremos en profundidad son los mecanismos de transmisión desde los motores hasta las ruedas. En general, la estabilidad del robot con ruedas no es problema, al igual que la repartición de la carga del robot. Sin embargo, no se adaptan a cualquier tipo de terreno. Si los desniveles superan el radio de las ruedas, se hace muy difícil superarlos. Ahora veamos cada tipo de arquitectura.

Configuración diferencial

Ésta es la configuración de ruedas más sencilla, y es la que utilizaremos en nuestro robot (**Figura 11**). Es económica y la transmisión no es tan compleja. Se utilizan **dos ruedas**, **una de cada lado**. Además se agrega una rueda libre o un punto de apoyo en la parte trasera o delantera para mantener el equilibrio. Al tener las ruedas alimentadas por dos motores independientes, es complicado mantener un movimiento rectilíneo sin el uso de sensores internos (encoders vinculados a las ruedas) o externos.

RECICLADO

Éstos son algunos sitios interesantes donde se explica cómo reciclar componentes de aparatos en desuso. Muchos de ellos serán útiles para nuestros robots:

- Reciclaje y proyectos electrónicos: **http://heli.xbot.es/wp/**.
- Proyecto EconoBot (un robot de costo cero): **http://mundobot.com/blog/**.
- Disquetera convertida en robot: **www.sorgonet.com/trashing**

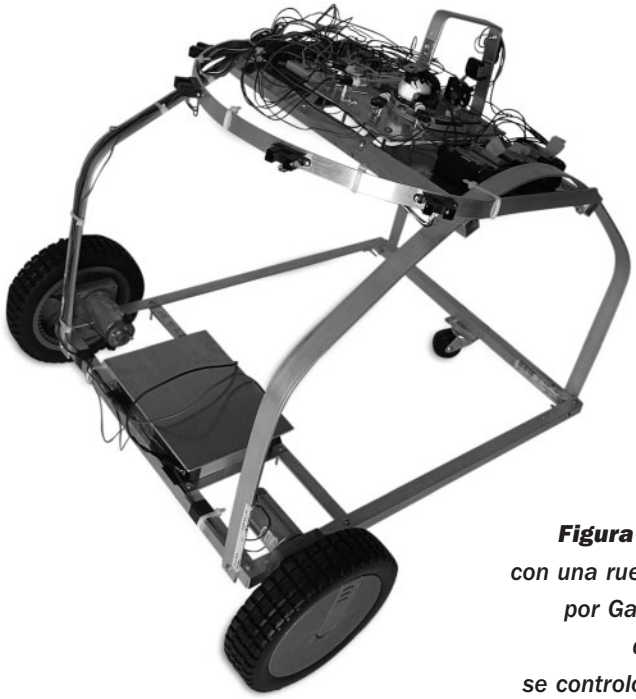


Figura 11. *Robot diferencial con una rueda libre, desarrollado por Garnet Hertz. Es la base de un robot que luego se controló ¡con una cucaracha!*

Cuando el motor que alimenta la rueda da las vueltas necesarias para completar una vuelta de esta última, no necesariamente ha girado su circunferencia por problemas de rozamiento. De este modo, la única forma de verificar que ambas ruedas giran sincronizadas es sensar el comportamiento interno o externo del robot.

Otra dificultad es la navegación hacia un punto determinado. Es un desafío complejo e interesante hacer que un robot llegue lo más rápido posible a su destino. Una forma sencilla es rotar hasta apuntar al objetivo final, y luego navegar en forma recta, corrigiendo de tanto en tanto el recorrido. Ésta es una navegación muy lenta para los ambientes dinámicos, como el del fútbol (**Figura 12**). Es por eso que más adelante haremos un estudio detenido de la navegación en estos casos, y veremos algunas alternativas a la navegación con trayectos curvos.

Configuración síncrona

En este caso, el robot tiene **tres ruedas** alimentadas por un mismo motor (**Figura 13**). Gracias a esto, avanza en línea recta en forma precisa. Para poder girar, incorpora un segundo motor, que también afecta a las tres ruedas, para rotarlas en su eje. De esta

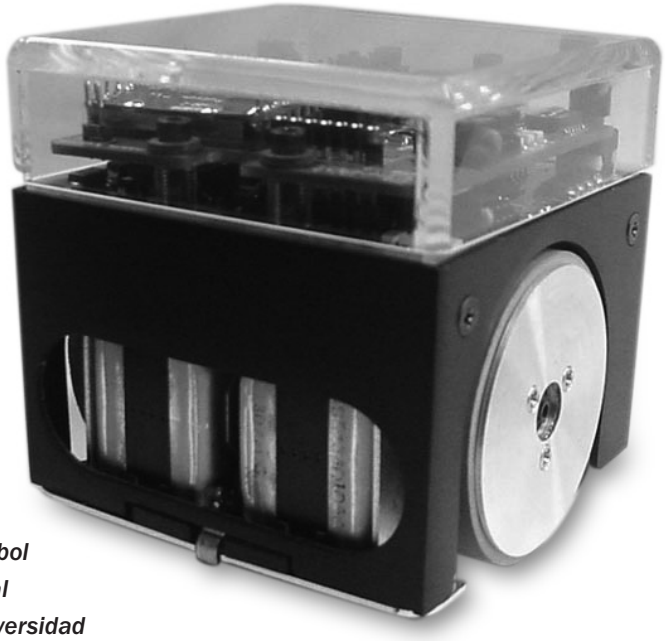


Figura 12. *Robot* ***Chebot****, jugador de fútbol con tracción diferencial desarrollado en la Universidad de Buenos Aires, Argentina.*

forma, puede avanzar en línea recta hacia cualquier punto, pero no puede trazar curvas. Al funcionar de esta manera, el frente del robot siempre estaría para el mismo lado. Es posible que esto no sea un problema, pero si necesitamos apuntar su frente en distintas direcciones, tenemos que agregar un mecanismo que permita rotar el chasis en la dirección de las ruedas. Otra arquitectura de la configuración síncrona sólo permite dos posiciones de las tres ruedas. En una de ellas, el robot avanza en línea recta, y en la otra posición, las ruedas se acomodan de tal manera que el robot gira sobre su eje. De esta manera, evitamos el problema del frente del chasis, pero la navegación es más lenta.

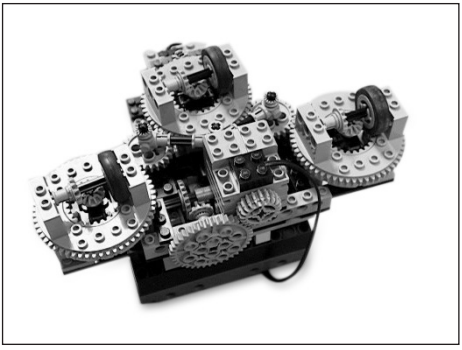


Figura 13. *Un robot de configuración síncrona hecho con Lego. Se puede observar el engranaje que alimenta el giro de la rueda y el que alimenta la rotación del soporte de ésta.*

Triciclo

En este caso tenemos dos ruedas alimentadas por un motor para darle tracción al robot, y una tercera rueda en la parte anterior o posterior del robot, sin tracción, pero con giro sobre su eje, que le permite darle dirección al dispositivo (**Figura 14**). Tiene buena estabilidad y su mecánica es sencilla. Puede ir recto en forma precisa gracias a que las ruedas de tracción están alimentadas por un único motor. El problema que se presenta es que el movimiento curvo es complejo, dado que el radio de las curvas que puede realizar es muy grande. No puede girar sobre sí mismo ni hacer curvas cerradas. Esto, para los robots que se mueven en ambientes pequeños, puede ser un problema insalvable.

Ackerman

Esta arquitectura es la que encontramos en los automóviles convencionales (**Figura 15**). Tenemos dos ruedas traseras con tracción y dos ruedas delanteras que definen dirección. Tiene la misma filosofía que el triciclo, pero agrega mayor estabilidad. La desventaja es que necesitamos desarrollar el sincronismo entre las dos ruedas delanteras, pero no es complicado. También se mueve en línea recta con absoluta precisión y tiene dificultades para realizar las curvas. Otra diferencia con el triciclo es que las ruedas traseras utilizan un mecanismo llamado diferencial que asegura que, a pesar de que la alimentación de ambas ruedas se realice con el mismo motor, una rueda pueda girar menos que la otra cuando sea necesario.

Ruedas omnidireccionales

Como podemos ver en la **Figura 16**, las ruedas ominidireccionales presentan una estructura muy peculiar. Se comportan como una rueda convencional pero no ofrecen resistencia para su desplazamiento lateral.

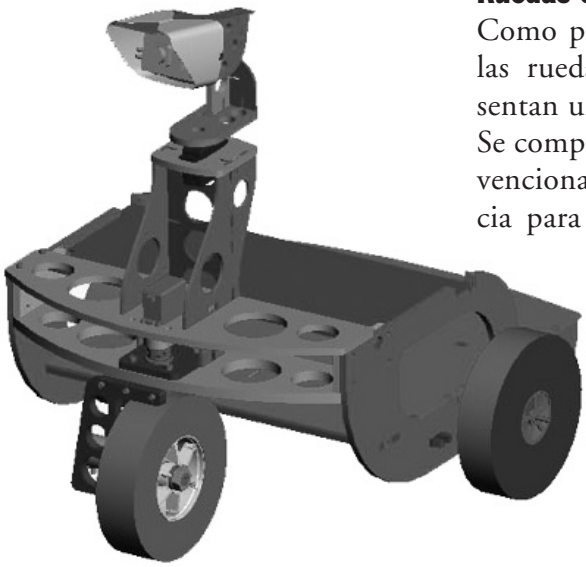


Figura 14. *Esquema de un robot con arquitectura de triciclo.*

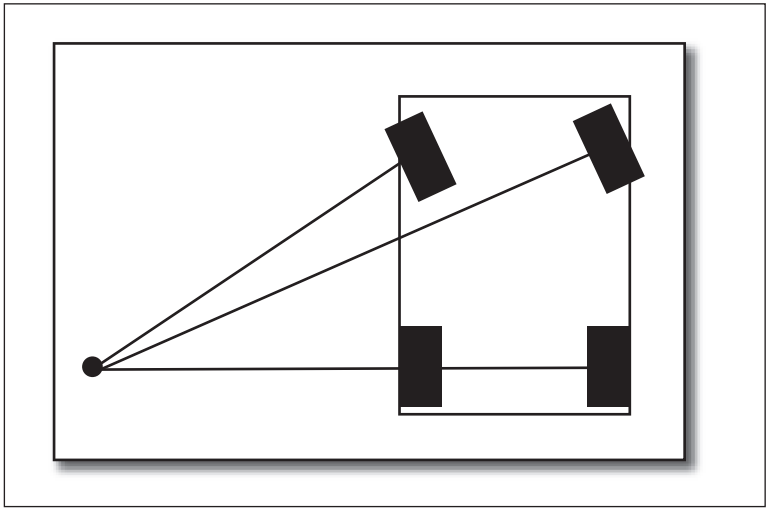


Figura 15. *Arquitectura de un robot con control de ruedas Ackerman.*

Por lo tanto, con tres ruedas ubicadas como se observa en la **Figura 17**, y con determinada velocidad en cada rueda, podemos lograr que el robot se dirija hacia cualquier punto o que gire sobre su eje.

Figura 16. *Rueda omnidireccional. Los rodillos permiten que la rueda se desplace en forma lateral con mínimo rozamiento.*

EAGLE KNIGHTS

En el Instituto Tecnológico Autónomo de México, el laboratorio de robótica ha desarrollado un equipo de fútbol de robots para competir en la categoría F-180 de la RoboCup. El equipo está conformado por cinco robots omnidireccionales que están controlados por un sistema central de inteligencia artificial. Este sistema central realiza el procesamiento de imágenes de las cámaras ubicadas sobre el terreno de juego, define las estrategias y los comportamientos de los robots, y les envía las instrucciones por radio. Podemos encontrar el proyecto completo en **http://robotica.itam.mx/espanol/**.

El algoritmo necesario para calcular la velocidad de cada rueda para dirigirse con un ángulo determinado es sencillo y consume mínimo tiempo de procesamiento. Las desventajas son que no es sencillo moverse en línea recta en forma precisa, y que es algo compleja la arquitectura del robot con respecto a la ubicación de los motores y las ruedas. ¡Pero es un buen desafío para afrontar más adelante! Es muy interesante ver cómo se mueven los robots con estas ruedas.

Estructura de nuestro robot

Antes de continuar con otros conceptos teóricos relacionados con los movimientos, desarrollaremos el chasis y la arquitectura de nuestro robot. Una vez que lo tengamos armado y podamos comenzar a probarlo, estudiaremos algunos conceptos físicos y matemáticos para lograr una navegación eficiente.

Especificaciones

El diseño que propondremos en este libro tiene como objetivo cumplir con las especificaciones del Campeonato Argentino de Fútbol de Robots, de manera tal que podamos utilizarlo para participar en la competencia. Está diseñado para la categoría Junior, aunque el tamaño también cumple con las especificaciones de la categoría Senior. Los chasis presentados (**Figura 18**) son

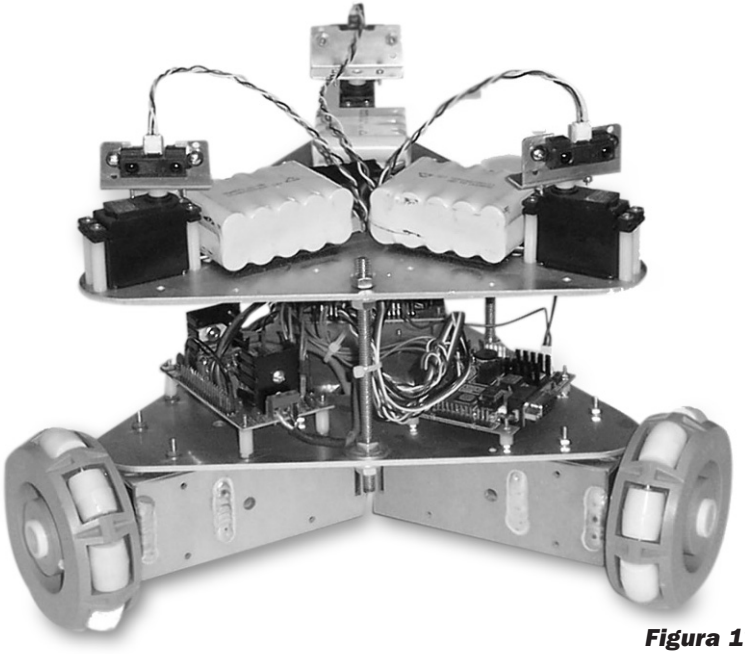


Figura 17. *Arquitectura de un robot con tres ruedas omnidireccionales.*

a modo de sugerencia, y dependen del uso que se le quiera dar al robot. Para poder calificar en cualquiera de las categorías del CAFR, el robot tiene que entrar en un cilindro de 18 cm de diámetro por 15 cm de alto.

No hay restricciones en cuanto a la cantidad de ruedas, motores y baterías, siempre y cuando el robot se adecue a las limitaciones del reglamento. Las principales limitaciones son que quepa dentro del cilindro mencionado y que no lastime o perjudique intencionalmente a un robot contrario (¡Fairplay!). Para mayor detalle del reglamento, podemos visitar la página del CAFR (**www.cafr.com.ar**), donde están las reglas originales de la RoboCup y la adaptación para nuestro campeonato local.

Tracción diferencial

A la hora de elegir el tipo de transmisión, la más fácil de implementar y con grandes ventajas es la tracción diferencial. Para lograrla, ubicaremos dos motores en el centro del robot conectados en forma directa a sus ruedas. De esta manera, podremos controlar la dirección del robot mediante el giro de los dos motores, de la siguiente manera:

CAETI

El CAETI (Centro de Altos Estudios en Tecnología Informática) es el centro de investigación de la UAI (Universidad Abierta Interamericana), donde los autores de este libro desarrollamos y probamos todo lo que contamos en estas páginas. La rama de investigación en la que participamos es la de robótica autónoma, y hemos formado el GIRA, Grupo de Investigación en Robótica Autónoma. Uno de los campos de trabajo es el fútbol de robots y cada año organizamos, junto con otras universidades e instituciones educativas, el Campeonato Argentino de Fútbol de Robots. Nuestro centro está abierto para todos los estudiantes e investigadores que quieran trabajar dentro de este campo. Para más información se puede consultar la página web: **http://caeti.uai.edu.ar/**.

Figura 18. *Ejemplo de chasis propuesto. El parche en la parte superior permite reconocer al robot en un sistema de visión.*

- Uno detenido y el otro que gire: giro sobre radio exterior.
- Uno que gire en un sentido y otro en el sentido opuesto: giro sobre su centro.
- Los dos que giren en el mismo sentido pero uno a menor velocidad: giro sobre un arco.
- Los dos motores que giren a la misma velocidad: avance o retroceso en línea recta. Como ya comentamos, esta opción es teóricamente posible, pero en su ejecución siempre nos encontramos con diversos problemas, como la diferencia de velocidad entre los motores, los rozamientos desparejos, etcétera. Necesitaremos de mecanismos de sensado del mundo o de las ruedas, de manera tal que podamos corregir los errores que se producen.

Chasis troquelado

Este chasis está realizado en aglomerado de 2 milímetros, inspirado en las figuras de cartón que venían troqueladas para armar. La plancha de aglomerado de ese espesor se puede cortar con un cúter, pero nos puede dar algo de trabajo. Si usamos una caladora de madera, podremos lograr nuestro objetivo con más facilidad.

Al finalizar la secuencia de armado, podremos pegar las piezas o encastrarlas. De elegir la primera opción, podemos dejar de lado algunas piezas cuya finalidad es darle rigidez estructural, como es el caso del soporte superior. En la **Figura 19** podemos ver el conjunto total de 14 piezas, el controlador, la batería, los motores y las dos ruedas (que utilizan unos bujes, que pueden ser reemplazados según las ruedas y los motores que utilicemos). Como vemos, en este caso hicimos el robot con una batería de 12 V porque no teníamos portapilas para 8 pilas de 1,5 V, pero nuestra sugerencia es que conviene usar 8 ó 12 pilas como comentamos en el capítulo de las baterías, así aprovechamos las pilas AA recargables convencionales.

Estructura de soporte

En primer lugar, cortaremos los soportes laterales que tienen por función principal el montaje de los motores. No olvidemos que el robot no debe superar los 15 cm de altura y 18 cm de diámetro. Los laterales son simétricos en espejo entre sí, y lo podemos observar en las ranuras en donde se encajan las otras piezas (**Figura 20**). A continuación, procederemos a cortar la pieza circular, que es la plataforma central donde se colocará la placa controladora. Por último, la pieza rectangular es un soporte que tiene por función darle rigidez al conjunto. La última pieza que cortaremos es el soporte para la batería. Allí montaremos la rueda de estabilización, que también es de madera, como podemos ver en la **Figura 21**.

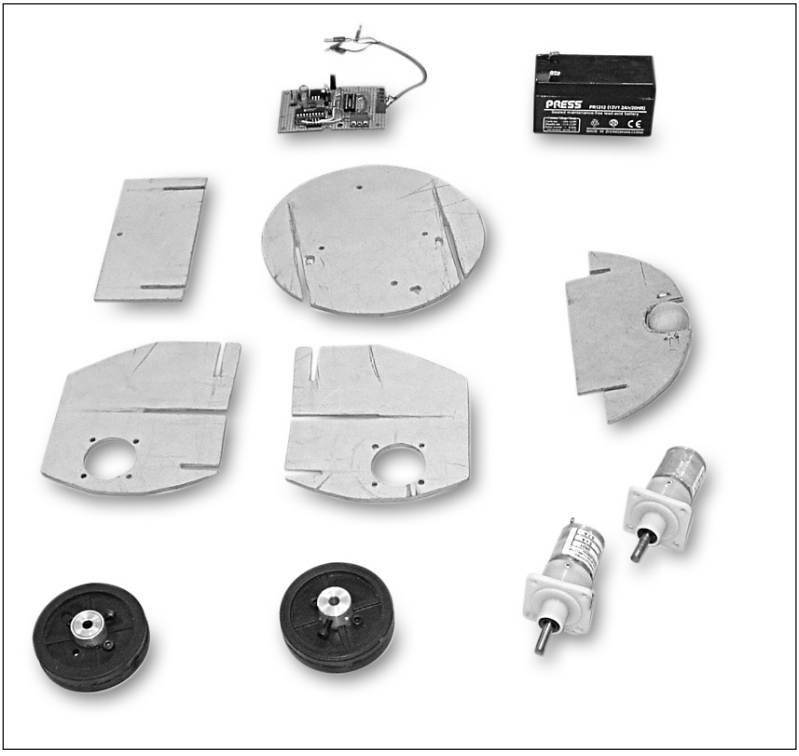


Figura 19. *Conjunto total con las 14 piezas que componen nuestro robot.*

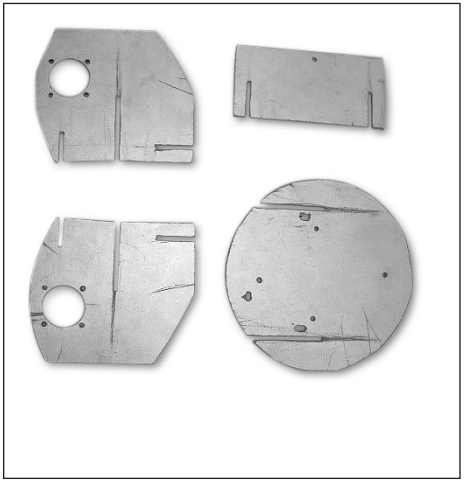


Figura 20. *Soportes principales de la estructura de nuestro robot.*

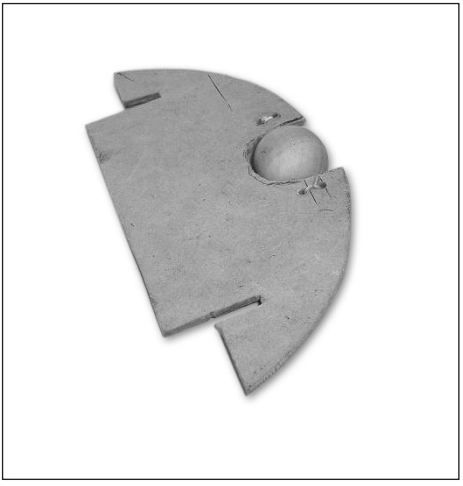


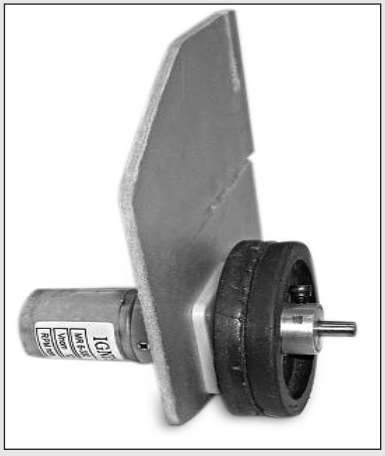
Figura 21. *Soporte de las baterías y rueda de estabilización.*

El montaje

Para poder realizar el montaje de nuestro robot, debemos llevar a cabo los pasos que veremos a continuación.

Realizar el montaje — PASO A PASO

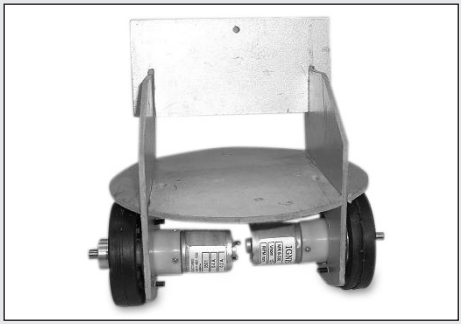
1 En primer lugar, debe montar el motor en los soportes laterales.



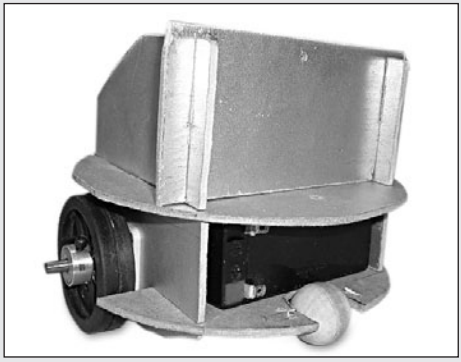
2 Monte la plataforma central con el primer soporte.

3 Monte el segundo soporte en la plataforma y la pieza superior para darles rigidez a los soportes laterales.

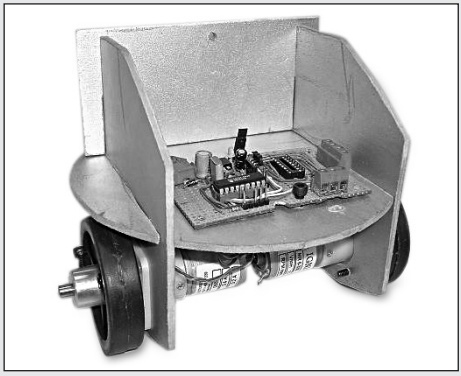
▶



4 Monte el soporte para las baterías. En este caso, los portapilas de pilas AA.



5 Ubique el controlador en la parte superior del robot, o en el lugar que quede disponible si agregó otros componentes. Así queda el robot terminado.



Podemos ver que el armado de la estructura es sencillo, y cada uno lo puede hacer a su medida. Si queremos variar de configuración, es recomendable utilizar piezas como las de Lego, Rasti o Mecano, junto con algunos soportes de aglomerado para los motores. Lo ideal es armar una pieza para cada motor que luego sea fácilmente vinculable con los ladrillos o las piezas del juego. Las plataformas para las baterías o el controlador son mucho más sencillas de realizar con las piezas.

Figura 22. *Robot listo para jugar un partido en la categoría Junior del CAFR.*

Mecanismos de transmisión y reducción

Muchas veces, los motores no se encuentran directamente vinculados con las ruedas o el mecanismo que se quiere mover. También tenemos casos en los que queremos alimentar, con un mismo motor, a un conjunto de ruedas, como en la configuración síncrona. Entonces, tenemos que desarrollar algún mecanismo de transmisión del movimiento del motor. Además, muchas veces el motor no tiene el torque (fuerza de rotación) o la velocidad que deseamos. Para ello, utilizamos mecanismos de reducción. Ambos están muy relacionados. Presentaremos cada mecanismo y comentaremos cómo funciona la transmisión y la reducción en cada caso.

Engranajes

Los engranajes son **ruedas dentadas** que empalman en forma directa una con otra. Los dientes se enganchan

{ } IMPRESORAS 3D

Cuando trabajamos con software, estamos acostumbrados a diseñar nuestros programas con libertad, y podemos cambiar cualquier bit sin problemas. Pero en cuestión de hardware, de lo que tenemos en mente hasta obtener el objeto físico hay un largo trecho. Sin embargo, ya se han inventado las impresoras 3D, que nos permiten crear, en diversos materiales, las piezas que necesitemos para nuestro robot. Por lo tanto, casi no quedan barreras para poder llevar cumplir nuestros sueños. ¡Sólo tenemos que dejar volar nuestra imaginación, contar con estas maravillas y ponernos a trabajar! Podemos ver un modelo de estas impresoras en **www.zcorp.com**.

entre sí, y el giro del engranaje de entrada (conectado al motor) se transmite al engranaje de salida (conectado a la rueda). Cada uno de ellos transmite su sentido e invierte el sentido del engranaje asociado. La transmisión es muy fiable, dado que si la conexión es correcta, no se produce ningún tipo de falla en la transmisión de los giros.
Uno de sus defectos es que el alcance de la transmisión es corto, y es necesario introducir engranajes intermedios (conocidos como **engranajes solidarios**) para alcanzar mayores distancias (**Figura 27**).

Además de ser usados para este fin, con los engranajes también podemos hacer **reducciones**. Un engranaje pequeño de entrada contra un engranaje más grande en la salida, aumenta la potencia y reduce la velocidad. Por ejemplo, si la entrada es de 8 dientes y la salida de 32, ante un giro de la entrada, sólo tendremos un giro de 1/4 de vuelta en la salida. Por lo tanto, las revoluciones de la rueda con respecto a las del motor disminuyen. Sin embargo, al tener el punto de apoyo más cerca del centro en el caso del engranaje pequeño, y más lejos en el que tiene conectado, aumenta el torque en la misma proporción.
Por lo tanto, si queremos calcular cómo se comportan el torque y la velocidad en la transmisión entre dos engranajes, debemos realizar ecuaciones, pero primero veamos lo que significa cada sigla.

RR: Revoluciones de la rueda.
RM: Revoluciones del motor.
DEE: Dientes de engranaje de entrada.
DES: Dientes de engranaje de salida.
TM: Torque del motor.
TR: Torque de la rueda.

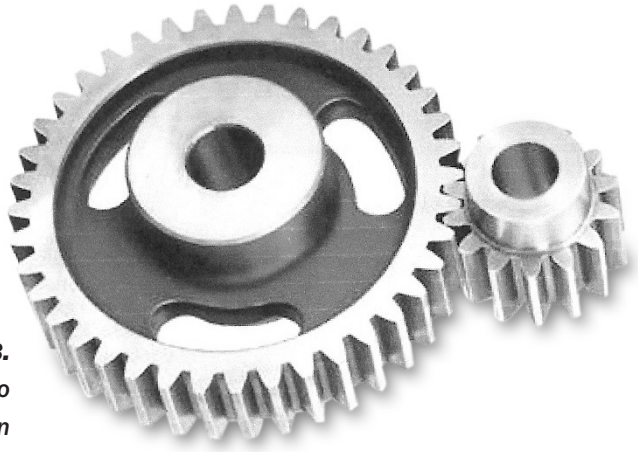


Figura 23.
Un ejemplo típico de reducción

La primera ecuación sería:

RR = RM * DEE / DES.

Si seguimos nuestro ejemplo de un engranaje de 8 dientes en la entrada y 32 dientes en la salida, ¿cuántas RPM (revoluciones por minuto) tendrá la rueda si el motor es de 20 RPM?

RR = 20 * 8 / 32 = 5 RPM

En el caso del torque la ecuación es:

TR = TM / DEE * DES

En nuestro ejemplo, si nuestro motor tiene 2 kgf, entonces, ¿qué torque tendremos en las ruedas?

FR=2 / 8 * 32 = 8 kgf

Podemos encontrar diversos tipos de engranajes, que nos permitirán realizar distintos tipos de conexiones según nuestras necesidades. Los engranajes solidarios son los tradicionales, que comparten un plano pero que tienen ejes distintos, como vemos en la **Figura 23**. Los **engranajes cónicos** permiten cambiar la dirección de la rotación y modificar así el plano de ésta. Podemos ver un ejemplo en la **Figura 24**.

Los sinfín o gusanos permiten realizar una caja de reducción que aumenta poderosamente la fuerza y disminuye notoriamente la velocidad. Podemos ver un ejemplo de caja de reducción en la **Figura 25**.

Los engranajes de piñón y cremallera permiten transformar un movimiento circular en traslación, como vemos en la **Figura 26**. Por último, también contamos con engranajes antitrabas, que si la transmisión supera determinada cantidad de fuerza, comienzan a girar en falso y ponen a salvo al motor que genera dicha fuerza.

Figura 24. *En esta imagen podemos observar un ejemplo de engranajes cónicos que fueron desarrollados por la empresa Arrow Gear.*

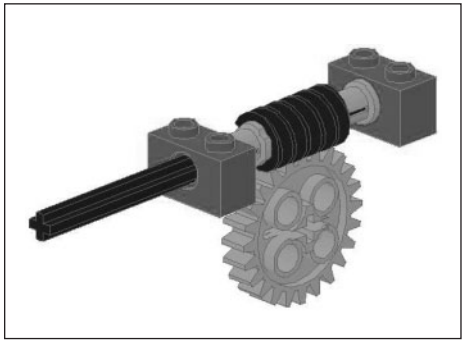


Figura 25. *Aquí podemos ver una caja reductora hecha con engranajes de Lego.*

Figura 26. *Engranajes de piñón y cremallera desarrollados por la empresa Gaes.*

Cuando usamos engranajes solidarios, el sentido de rotación del de salida es inverso al de entrada.
Si queremos mantener el mismo sentido, tenemos que incorporar en el medio engranajes libres, de manera tal que los que tienen la misma paridad (es decir, los pares por un lado y los impares por otro) mantengan el mismo sentido de rotación, como podemos ver en la **Figura 27**.
El tamaño de la rueda también puede considerarse en el cálculo de la velocidad y la fuerza. Es decir, cuando tenemos ruedas más grandes, obtenemos mayor velocidad, pero disminuimos la fuerza, y viceversa.

Transmisión por poleas

Si necesitamos realizar la transmisión a una distancia mayor, utilizamos poleas unidas por correas o elásticos, como podemos ver en la **Figura 28** de la próxima página.

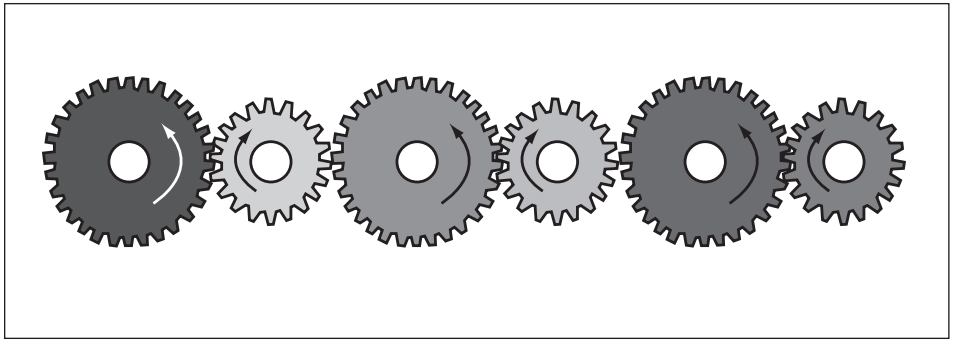


Figura 27. *Sentido de rotación de cada engranaje en un conjunto de engranajes solidarios.*

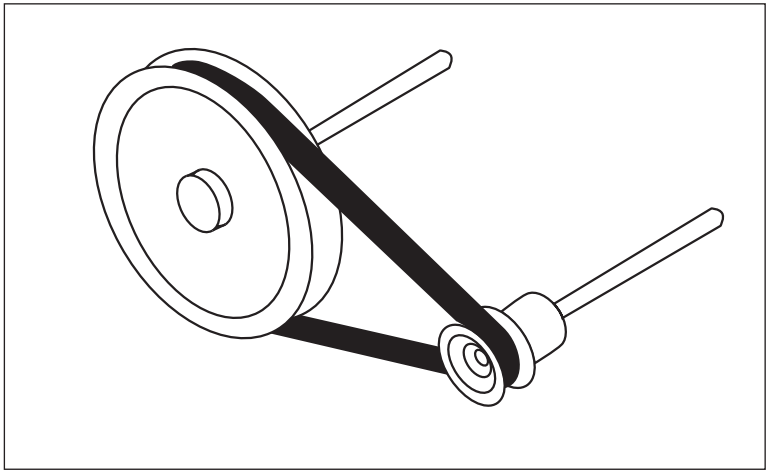


Figura 28. *Transmisión por poleas. Cuanto más gruesa es la correa de transmisión, nos aseguramos menor probabilidad de deslizamiento.*

En este caso, la polea de entrada tiene el mismo sentido de rotación que la polea de salida, salvo que crucemos la correa como se puede ver en la **Figura 29**. También tenemos un cálculo de reducción relacionado con la proporción entre las poleas conectadas. La diferencia fundamental es que durante la transmisión, puede haber pérdida de movimiento por deslizamiento de las correas o poleas, lo que no ocurre en los engranajes excepto que éstos se rompan.

Si necesitamos combinar las ventajas de ambos mecanismos, es decir, si queremos transmitir a larga distancia pero sin tener deslizamientos, entonces tenemos que utilizar cadenas con piñones, como las que podemos encontrar en las bicicletas.

Cinemática de un robot

Cuando comenzamos a estudiar el comportamiento de los movimientos de nuestro robot, nos damos cuenta de que tenemos que tener en cuenta un conjunto muy amplio de variables, propias y externas de nuestro dispositivo. Al estudio de todos estos aspectos en nuestra criatura se lo conoce como **dinámica del robot**. Como es un campo complejo para encarar en este primer paso, vamos a estudiar el movimiento del robot sin considerar las fuerzas involucradas (inercia, pesos, rozamientos), sólo tomaremos el comportamiento de las ruedas, con adherencia perfecta. A este campo se lo conoce como **cinemática del robot**. Ahora bien, podemos encarar dos tipos de estudios. Uno de ellos es, si

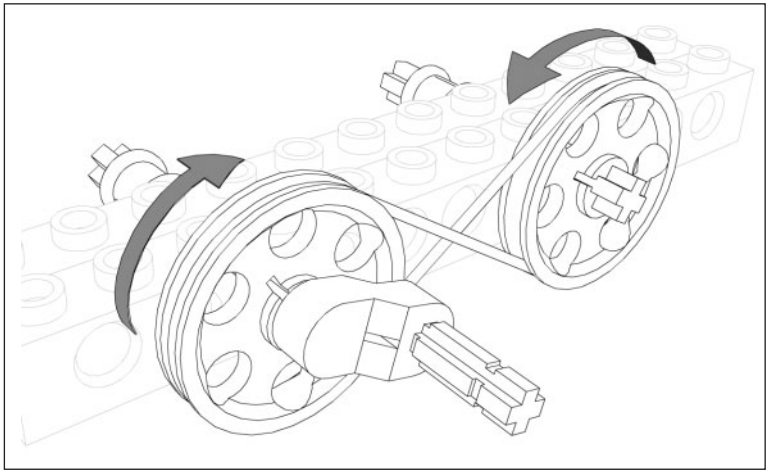


Figura 29. *En esta imagen podemos observar cómo funciona la inversión del sentido de rotación de la polea de salida.*

sabemos la posición inicial del robot y los movimientos que realiza, determinar la posición final. A esto se lo llama **cinemática directa**.
Cuando queremos saber qué movimientos hacer para llevar al robot de una posición inicial a una posición final, estudiamos la **cinemática inversa**. Éste es el campo que a nosotros más nos interesa, porque con el robot queremos llegar a un punto determinado del espacio y realizar la mejor trayectoria posible desde nuestro punto inicial.

Cada una de las arquitecturas que presentamos tiene una cinemática distinta, que establece ciertas restricciones en el movimiento.
Antes de analizar las restricciones y estudiar los mecanismos de navegación, debemos aclarar que en los robots móviles, tenemos tres grados de libertad: posición en el plano (X e Y) y rotación (ver **Figura 30**).

En el caso de los robots diferenciales, los de configuración síncrona y los de ruedas omnidireccionales, las restricciones cinemáticas son **holonómicas**, lo que significa que los diferentes grados de libertad son independientes. En este caso, posición es independiente de rotación.

En el caso de la arquitectura triciclo y Ackerman, decimos que la restricción no es holonómica. Necesariamente, para girar afectamos la posición (X,Y) del robot. En cada uno de ellos el estudio de la cinemática presenta características distintas.

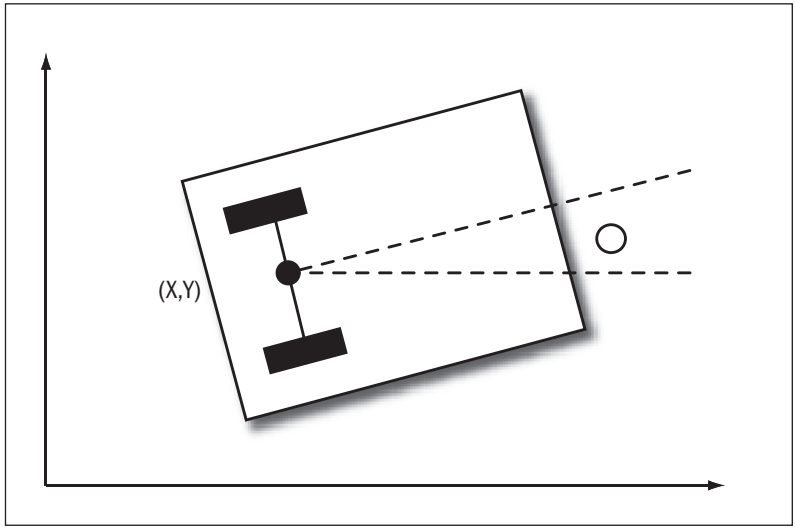


Figura 30. *Grados de libertad que tenemos en un robot terrestre móvil.*

Odometría

Ahora bien, ¿cómo podemos saber la posición de nuestro robot en todo momento? Como comentamos antes, una de las formas es analizar el contexto que rodea al robot constantemente. El problema es que no siempre contamos con información externa y, por otra parte, esto nos exige tomar datos en forma constante con sensores que pueden necesitar una tasa de procesamiento muy alta. Es por eso que la odometría es el mecanismo más utilizado para determinar en forma aproximada la posición del robot. Consiste en **incorporar encoders** situados en las ruedas de los robots, de la misma manera que los antiguos mouses detectaban el giro de la bolita inferior (**Figura 31**). Como ya mencionamos, esto es sencillo de realizar con sensores de ranura y un disco acanalado que permita el conteo de los cortes de señal, o que posibilite el cálculo de rotación por los sensores que se encuentran activados. Esto resulta económico, pero sin información externa, y se produce una acumulación de errores que nos puede llevar a datos inexactos. Por ejemplo, si la rueda resbala sobre el piso, el encoder detectará el giro y supuestamente un avance lineal, cuando en realidad no se produjo.

Los tipos de errores que podemos encontrar en la odometría son **sistemáticos**, cuando son intrínsecos al mecanismo de toma de datos, como pueden ser problemas en la tasa de muestreo, mal alineamiento de

las ruedas, etcétera. Y **no sistemáticos**, cuando están vinculados a hechos casuales como los comentados: una rueda que patina, fuerzas externas que retienen o levantan al robot, suelos desnivelados, etcétera.

A pesar de todo, por su bajo costo y su facilidad de cálculo, es uno de los métodos más utilizados. Lo que recomendamos es sumar alguna toma de información más que permita reducir los errores de este proceso.

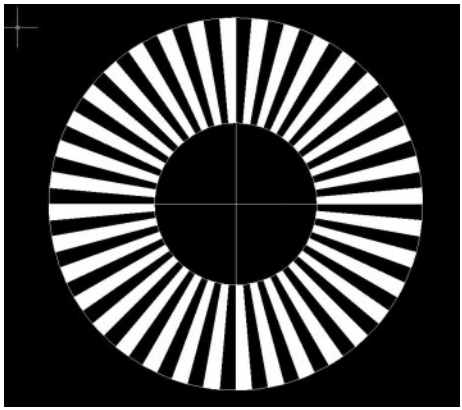


Figura 31. *Rueda con encoder para desarrollar* ***algoritmos de odometría.***

... RESUMEN

La arquitectura de nuestro robot definirá un conjunto de variables relacionadas con su navegación. La decisión del tipo de mecanismo utilizado para el movimiento deberá estar basada en la superficie en la que se desplazará. Desde ya, los aéreos y los acuáticos son los más complejos de resolver, pero el movimiento en tierra también presenta un conjunto de desafíos. Podemos utilizar patas, orugas, ruedas, o construir un robot similar al funcionamiento de un gusano. En nuestro primer robot, utilizamos un sistema de ruedas diferencial, ya que es el más sencillo de implementar y el de menor costo. Esto nos obligará a detenernos a estudiar la cinemática inversa de nuestro robot, para obtener la navegación más precisa posible. Pero cuando hayamos superado las primeras pruebas, podremos intentar con otros sistemas de ruedas como las síncronas o las omnidireccionales. En todos los casos, tendremos que resolver el problema de la transmisión del giro de los motores hacia las ruedas, con engranajes o poleas. Y si es posible, agregarles encoders a las ruedas para que nos permitan obtener información sobre los movimientos de nuestro robot.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Cuáles son las características que hay que tener en cuenta para el diseño del cuerpo del robot?

2 ¿Cuáles son los problemas más importantes en el diseño de los robots aéreos?

3 ¿Cuáles son los problemas más importantes en el diseño de los subacuáticos?

4 ¿Cuáles son los mecanismos de desplazamiento de los robots terrestres?

5 ¿Cómo se componen los robots ápodos y cómo generan el movimiento?

6 Describa los sensores de efecto hall.

7 ¿Cómo es la configuración diferencial de las ruedas?

8 ¿Qué diferencias y similitudes tienen la arquitectura de triciclo y la Ackerman?

9 ¿Cómo funcionan las ruedas omnidireccionales?

10 ¿Cuál fue el sistema de desplazamiento que elegimos para nuestro robot y por qué?

EJERCICIOS

1 Construya el cuerpo del robot y prográmelo para que funcione en línea recta. ¿Qué diferencias tuvo que determinar en los pulsos de cada motor para igualar la velocidad?

2 Realice un programa para que el robot gire sobre sí mismo.

3 Incorpore un sensor de tacto en la parte delantera (puede usar un microswitch con una palanca grande), programe el robot para que cada vez que detecte un objeto retroceda, gire y se dirija a otro lado.

4 Incorpore un sensor de luz en la parte inferior delantera del robot, realice un programa para que se mueva sobre una mesa blanca (o con un papel blanco) y si se está por caer (es decir, el sensor deja de recibir señal), frene, retroceda y se dirija a otro lado.

5 Describa el comportamiento de los engranajes para la transmisión y la reducción.

6 Compare las ventajas y desventajas entre el uso de engranajes y poleas.

Salir al ruedo

Ya tenemos nuestro robot armado. Ahora nos toca probar cómo se comporta en las pruebas de robótica autónoma móvil. Para ello presentaremos un conjunto de desafíos, veremos los problemas que tendremos que resolver y propondremos un código para programar nuestro robot. ¡Pero el ambiente no será siempre igual! Por lo tanto, hay que experimentar una y otra vez hasta encontrarles la solución a los problemas. Y luego, modificar nuestro robot para mejorar los resultados.

PRESENTARSE A COMPETIR

Ahí está, ¿no es bonito? Toda una maravilla, independiente, solícito y muy inteligente. Nuestro robot está listo para comenzar a representarnos en el mundo. Ahora tiene que salir al ruedo, porque allí es donde se ven los buenos caballos. Pero eso es pan comido. Le colocamos dos sensores de luz, lo ponemos sobre una línea negra para que la siga y… ¡Se va a cualquier lado! Lo ubicamos sobre una mesa y al llegar al final no detecta el borde y cae con sus trágicas consecuencias. Le ponemos una linterna adelante para que la siga y nada, da vueltas desesperado sin hallar la luz que lo guíe. Evidentemente, hemos subestimado la dificultad de las pruebas y nos falta entrenamiento, horas y horas de entrenamiento para hacer de nuestro robot un buen contrincante.

En este capítulo presentaremos las pruebas clásicas para robots autónomos móviles y nos detendremos a analizarlas para ver dónde se encuentran las dificultades. También veremos algunos aspectos de navegación para que el movimiento de nuestro robot sea lo más fluido posible y nos adentraremos un poco en el fútbol de robots, que es el tema de nuestro próximo capítulo. Si hemos tenido la paciencia de llegar hasta este capítulo, lo que nos espera ¡es apasionante!

Características comunes de las pruebas de robots autónomos

Cada prueba de robótica presenta cuestiones particulares que debemos resolver, pero en general hay un conjunto de principios básicos comunes. Antes de encarar cada una de ellas, presentaremos cuáles son los puntos que tenemos que tener presentes tan-

Figura 1. *Robot seguidor de líneas.*

to en el hardware como en el software de nuestro robot.

Problemas mecánicos

Es posible que éstos sean los problemas más habituales y más difíciles de superar. Es fundamental e imprescindible crear un espacio de pruebas lo más parecido posible al lugar donde se ejecutará la competencia. Lamentablemente, es habitual que no tengamos la descripción completa del terreno, sino una aproximación con cotas de mínima y máxima en diferentes aspectos. Dada esta información, para poder completar la misión en el ambiente más exigente, testeamos todo con las cotas más duras. Esto trae aparejado que nuestro robot sea más lento o impreciso en otros aspectos. Por ejemplo, si tenemos que subir una rampa de un ángulo de inclinación importante, las orugas brindan buena tracción. El problema es que su capacidad de giro es muy limitada, y la velocidad es lenta. Por lo tanto, si al llegar a probar nuestro robot en el escenario real nos encontramos con una rampa de menor inclinación, es posible que un sistema diferencial nos permita subirla sin las otras dificultades. En resumen, debemos tener la capacidad de cambiar entre dos o tres sistemas de tracción en el momento de la competencia.

Otra dificultad es el tipo se superficie donde nos movemos y la presencia de escombros u objetos que estorben nuestro recorrido. Para ello, contar con diferentes sistemas de tracción también puede ser una solución. Si los objetos son fijos, deberemos añadir sistemas de sensado para poder esquivarlos.

Otro aspecto complicado es la distribución de los pesos en el robot. Esto influye en el comportamiento de los

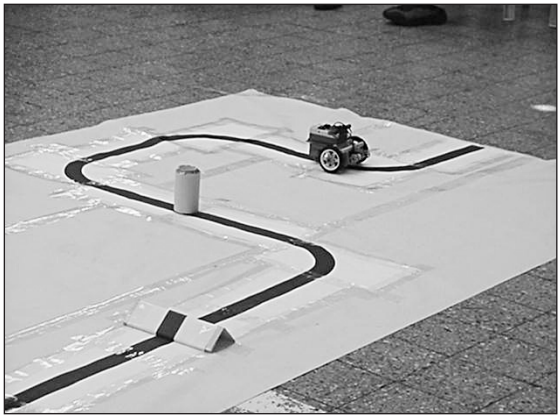


Figura 2. *Seguimiento de línea con obstáculos.*

giros, en cómo se ubican los sensores ante los objetos a detectar, etcétera. Los pesos más significativos están dados por las baterías y los motores. Como los motores no se pueden reubicar con facilidad, sería bueno que contemos con abrojos en el robot donde podamos pegar las baterías de diversas formas para la distribución del peso. Si usamos pilas, es bueno tenerlas en portapilas separados, de dos pilas cada uno, de manera tal que podamos reubicarlos con mayor libertad. Una anécdota con respecto a este punto: en el Primer Campeonato Latinoamericano de Robótica IEEE, utilizamos un sensor casero muy pesado, hecho de madera y escobillas de autitos de carrera, que hacía que el robot volcara hacia delante. Esto perturbaba el giro del robot. Muchas veces quedaba trabado sobre imperfecciones del piso y arruinaba nuestro desempeño. Como no teníamos peso para agregarle (sólo podíamos utilizar fichas del kit de Lego), le pusimos un controlador adicional (el RCX de Lego) con seis pilas en la parte posterior, y así logramos el equilibrio necesario.

Cuando obtuvimos el primer puesto, nos felicitaron por la complejidad del robot que utilizaba dos controladores que se comunicaban entre sí. Nuestra honestidad (¡y la alegría del primer puesto!), nos llevó a aclarar que el segundo controlador estaba apagado y que su única función era de servir de contrapeso, con lo cual las risas colmaron el salón.

Para cada uno de los sistemas de tracción que construyamos, tendremos que analizar en forma detallada la forma en que nuestro robot realizará los giros. Esto también va a in-

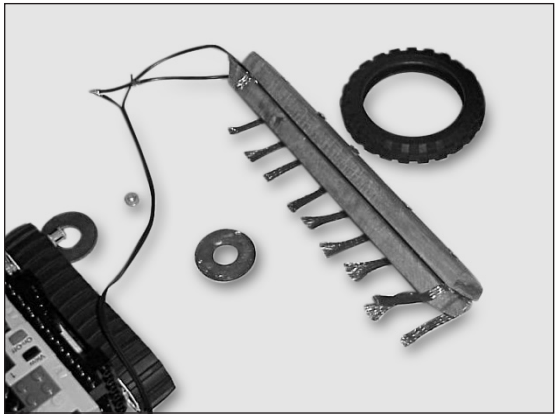


Figura 3. *Sensor casero que utilizamos en el Primer Campeonato Argentino de Robótica IEEE. ¡Qué pesado que era!*

fluir en la ubicación de los sensores y en la programación. Por lo tanto, también tendremos que poder reubicar los sensores con facilidad.

Problemas de sensores

Con respecto a los sensores, también se nos presenta un conjunto de problemas diversos. Así como la superficie de la competencia puede variar llegado el juego, con seguridad las luces ambiente, los colores, la capacidad reflectiva, etcétera, las podremos definir recién en ese momento. Incluso ¡pueden variar en el transcurso de la competencia! Por lo tanto, debemos tener algún mecanismo que nos permita testear los valores y modificar nuestro programa para que se comporte en forma correcta. Ahora, ¿cómo hacemos para saber si el sensor se activa o no, o cuál es el rango analógico en el que se debe activar? Con el control que hemos realizado, pareciera que no tenemos forma alguna de monitorear los valores de lectura. Pero vamos a ver algunos trucos que nos pueden ayudar. La primera forma de **monitorear** un sensor digital es usar el led que hemos conectado para nuestras pruebas. Por ejemplo, con el siguiente programa podemos verificar si el sensor CNY70 detecta la variación entre negro y blanco. Consideramos, como vimos en el **Capítulo 7**, que hemos conectado el led infrarrojo en RA1 y el sensor propiamente dicho en RA2. Por lo tanto, tendremos que configurar RA1 como bit de salida, y RA2 como bit de entrada. Un detalle fundamental a tener en cuenta es que la resistencia que va del sensor a +5 puede ser modificada para obtener mayor o menor sensibilidad en la detección.

ROBOLIGA

La Roboliga consiste en tres eventos que se realizan en el mes de noviembre en Argentina: la Olimpíada Argentina de Robótica, la Feria de Proyectos de Robótica y Control Automatizado, y el Encuentro Docente para la Enseñanza de Robótica. Está dirigido a los alumnos de escuelas primarias y medias, se organiza desde el año 2000 y se cuenta con la presencia de más de 40 escuelas de todo el país. Para tener más información podemos visitar **www.roboliga.edu.ar**.

```
program testCNY

     TRISB=0       'Declaro RB
      de salida
     TRISA.1=0     'RA1 (donde
      está el led infrarrojo)
      de salida
     PORTA.1=1     'Lo enciendo
     TRISA.2=1     'Declaro RA2
      de entrada
     while true
```

```
            if PORTA.2 = 1 then
              'Si RA2 está
              encendido, apago
              el led
                   PORTB.3=0
            else
                PORTB.3=1
              'Si no, lo enciendo
            end if
            delay_ms(200)
    wend
end.
```

También podemos utilizar un parlante piezoeléctrico para generar diversos sonidos que nos permitan conocer el valor de un sensor (**Figura 4**). Éstos se pueden conectar directamente, una de sus conexiones a GND y la otra a un pin. Para generar sonido desde mikroBasic podemos utilizar las siguientes funciones de librería:

- **Sound_Init(dim byref port as byte, dim pin as byte)**: inicializa el pin para salida. Por ejemplo: **Sound_Init (PORTB, 2)**. En este caso, se encarga de inicializar RB2.
- **Sound_Play(dim periodo_div_10 as byte, dim num_de_periodos as word)**: el primer parámetro indica la frecuencia, y para simplificar su definición, sólo debemos dividir la frecuencia que queramos por 1 y multiplicar el resultado por 100000 para obtener el valor final. El segundo parámetro indica la cantidad de ciclos de duración del sonido.

De esta forma, podemos vincular el valor de un sensor analógico con diferentes sonidos, y así monitorear el comportamiento de los sensores.
Por último, lo más interesante puede ser conectar un display a nuestro micro (**Figura 5**). El problema es que para controlarlo utilizamos casi todos los puertos de salida. Será necesario utilizar un micro aparte para tener es-

CONSEJO LATINOAMERICANO DE ROBÓTICA IEEE – RAS

El Consejo Latinoamericano de Robótica (LARC) tiene como objetivo organizar, con la colaboración de organismos locales, competencias y simposios de robótica latinoamericanos a nivel regional y nacional de manera anual. El consejo promueve la programación de estos eventos, define las bases para las competencias, e interactúa con los voluntarios locales que luego desarrollan estas actividades por sí mismos. Este consejo trabaja principalmente en Internet, y está apoyado por la sociedad de robótica y automatización RAS que pertenece a la IEEE mediante los capítulos profesionales y estudiantiles de la región. Para más datos podemos visitar **www.ewh.ieee.org/reg/9/ robotica/indexsp.htm**.

te tipo de salida, si así lo deseamos. Escapa al alcance de este libro, pero se puede encontrar mucha información vinculada a este tema en Internet.

Problemas de programación

Uno de los problemas más comunes cuando sensamos el mundo que nos rodea con nuestro robot, es la necesidad de estar atentos a dos o más sensores al mismo tiempo. Por ejemplo, si realizamos un seguidor de línea con un par de sensores y además tenemos un sensor de tacto en caso de que encontremos un objeto delante de nosotros, las estructuras de decisiones se vuelven complejas o

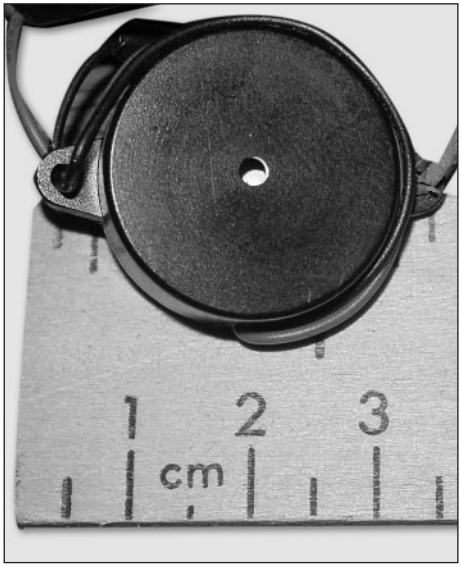


***Figura 4. Buzzer piezoeléctrico** para analizar valores de sensores analógicos.*

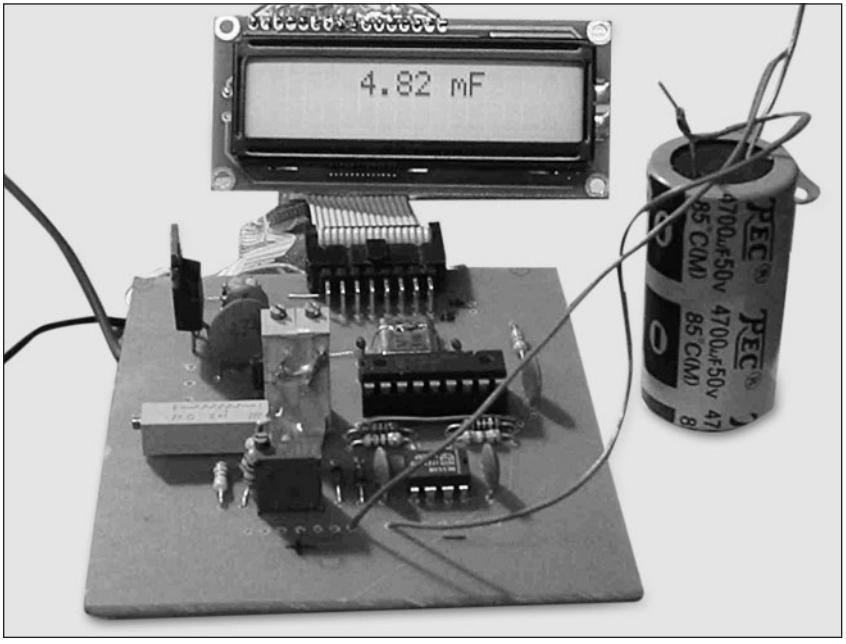


***Figura 5. Capacímetro digital** basado en un 16F84 con display de LCD, desarrollado por Sonytel Argentina.*

imposibles para representar la combinación de todos los estados posibles de las entradas. Es por eso que a continuación presentaremos un mecanismo muy utilizado en los procesadores: las **interrupciones**.

La idea de una interrupción es poder vincular algún proceso interno del micro, como el estado de un pin de entrada o el valor de un **timer**, con un procedimiento de nuestro programa. Es decir, ante un cambio en el estado del pin o ante el desbordamiento de un timer, se ejecuta el procedimiento indicado. Por ejemplo, de esta manera podemos asociar nuestro sensor principal a un bit con interrupción, de forma tal que cuando se active, se invoque en forma automática al procedimiento asociado, se atienda la necesidad que presenta el sensor, y luego volvamos a nuestra rutina habitual. En el caso del seguidor de línea con obstáculos, el programa principal ejecuta el seguimiento y al bit de interrupción le asociamos nuestro sensor de tacto. Por lo tanto, si encuentra un obstáculo, se detiene el seguimiento de la línea, se esquiva el objeto, y al volver sobre la línea se continúa con el programa principal.

Las interrupciones pueden producirse por alguna acción externa, como la activación de un sensor, o por una interna, como el caso de un timer o desbordamiento de un registro. En el 16F84/8 tenemos cuatro fuentes de interrupciones:

- RB0/INT (externa).
- Pines RB4 a RB7, al cambiar de estado si están configurados como entrada (externa).
- Desbordamiento del registro **TMR0**, cuando pasa de 255 a 0 (interna).
- Al completar la escritura de la EEPROM (interna).

En nuestro caso, las tres primeras son las que podremos aprovechar en nuestro robot. Cuando se producen, se modifica el estado de un **flag** de interrupciones, lo que nos permite saber cuál se ha producido. Este flag es el registro **INTCON**, donde el bit que cambia es el siguiente:

ROBOGAMES

Es la competencia de robótica más grande del mundo. En su edición de 2007 se presentaron setenta categorías distintas, desde el simple seguidor de líneas hasta combates cuerpo a cuerpo entre robots. Algunas de ellas son con robots controlados por humanos y en otros casos, el desafío es autónomo. Ese año estuvieron representados veintiocho países de todos los continentes, y para el año que viene se esperan alrededor de veinte países más.

- Bit 1, **INTF**, cuando se produce una interrupción por RB0.
- Bit 0, **RBIF**, cuando se produce por cambio en RB4 a RB7.
- Bit 2, **TOIF**, cuando se produce por TMR0.

Para tener una mirada general del registro **INTCON** y su funcionalidad, podemos observar la **Tabla 1**.

Los bits 3 a 7 nos permiten determinar si se atenderá la interrupción correspondiente o no (**0**: deshabilitado; **1**: habilitado). En el caso de **GIE**, si está en **0** se deshabilitan todas las interrupciones. Los bits del 0 al 2 se encienden en el caso de que se produzca la interrupción en cada caso.
Cuando se produce una interrupción y el programa salta a la rutina que la atiende, como veremos más adelante, podemos examinar los bits del 0 al 2 para saber la fuente que la ha causado. El **GIE** se pondrá automáticamente en **0** para no atender otra interrupción al mismo tiempo. Al finalizar nuestra rutina de atención, debemos volver a poner en **0** el flag correspondiente (0 a 2) para que no se vuelva a producir la llamada. El retorno de la rutina nos habilitará de nuevo **GIE** para una nueva llamada. Sólo el contador de programa será almacenado en el momento de llamar a la rutina, para luego saber dónde retornar, con lo cual es problema nuestro almacenar el estado de otros registros que puedan llegar a modificarse en la ejecución de nuestra rutina.

A continuación presentamos un ejemplo general de código para la atención de las interrupciones por cambio de estado en RB4 a RB7. De esta manera, podemos tener sensores conectados a uno de estos 4 pines, de forma tal que el programa general se ejecute, y cuando se produzca la activación de alguno de los sensores, la rutina de la interrupción atienda la llamada y luego retorne al punto en el que se encontraba.

No.	NOMBRE	DESCRIPCIÓN
7	GIE	Global Interrupt Enable (Habilitación general de interrupciones).
6	EEIE	EEPROM Write Interrupt Enable (Habilitación de interrupción por escritura de la EEPROM).
5	TOIE	TMR0 Interrupt Enable (Habilitación de interrupción del timer TMR0).
4	INTE	INT Interrupt Enable (Habilitación de interrupción de RB0/INT).
3	RBIE	RBIF Interrupt Enable (Habilitación de interrupción por cambio en RB4 al RB7).
2	TOIF	TMR0 Overflow Interrupt Flag (Bandera de interrupción por desbordamiento de TMR0).
1	INTF	INT Interrupt Flag (Bandera de interrupción de RB0/INT).
0	RBIF	RB Port Interrupt Flag (Bandera de interrupción por cambio en RB4 al RB7).

***Tabla 1.** Funcionalidad del registro INTCON.*

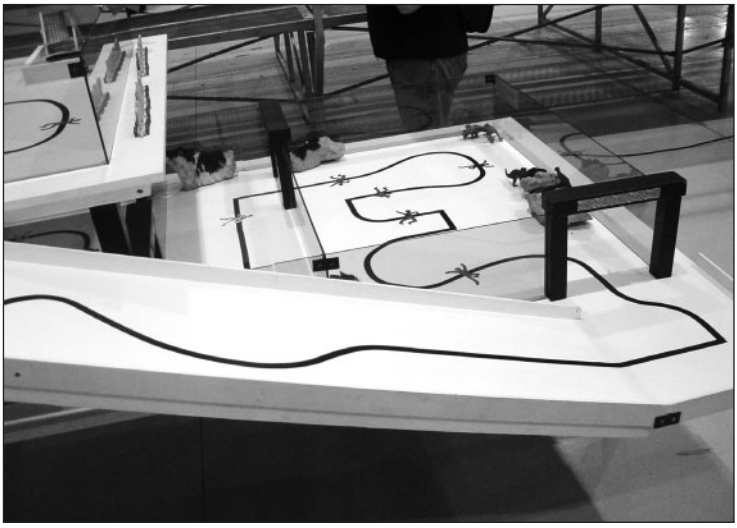


Figura 6. *Subir una rampa mientras se sigue una línea, cambia el comportamiento del robot.*

```
program interrupcionRB

dim antRB as byte

sub procedure interrupt
 'Rutina de atención de
     interrupciones
  if (INTCON.RBIF = 1) then
     'Cambio en RB4-RB7
    select case (antRB xor
     PORTB)     'Se fija cuál
     cambió comparando con el
     estado anterior de PORTB
    case %10000000     'Cambió
     RB7
      if ((PORTB and %10000000)
            = 0) then
           'RB7 desactivado
            'Código a ejecu
tarse cuando se desactiva RB7
       else     'RB7 activado
             'Código a
              ejecutarse cuando
              se activa RB7
      end if
     case %01000000     'Cambió
      RB6
       if ((PORTB and %01000000)
       = 0) then
      'RB6 desactivado
             'Código a
            ejecutarse cuando
            se desactiva RB6
       else     'RB6 activado
             'Código a
            ejecutarse cuando
            se activa RB6
```

```
    end if
  case %00100000     'Cambió
  RB5
    if ((PORTB and %00100000)
          = 0) then     'RB5
          desactivado
           'Código a
          ejecutarse cuando
          se desactiva RB5
    else     'RB5 activado
           'Código a
          ejecutarse cuando
          se activa RB5
    end if
  case %00010000     'Cambió
  RB4
    if ((PORTB and %00010000)
     = 0) then     'RB4
     desactivado
           'Código a ejecutar
          se cuando se
          desactiva RB4
    else     'RB4 activado
           'Código a
          ejecutarse cuando
          se activa RB4
      end if
    end select
    antRB = PORTB     'Almaceno
     el estado actual de PORTB
     para compararlo luego
    INTCON.RBIF = 0     'Reseteo
     el flag de la interrupción
  end if
end sub

main:

  TRISB = %11110001     'RB 0
     entrada, 1-3 salidas,
     4-7 entradas
  INTCON = %10001000
     ' Habilito interrupciones
     en general y las de RB4 a 7
  PORTB = 0     ' Valor inicial
     de PORTB
```

LAS INTERRUPCIONES

Las interrupciones son un mecanismo habitual en las computadoras para que los dispositivos le avisen al CPU que deben ser atendidos. De esta manera, el procesador no tiene que estar atento a las necesidades de éstos, y simplemente procesa la información entrante en el momento en que se lo solicitan. En el momento en que atiende una interrupción, almacena el estado del procesamiento en ese instante, y luego de atenderla, continúa exactamente en el punto en el que había dejado el proceso. Cuando no existía este mecanismo, el CPU debía dedicar mucho tiempo para verificar si era necesaria esa atención. Al pasar la responsabilidad a los dispositivos, el rendimiento del CPU es mucho mayor.

```
    antRB = 0     ' Valor inicial
       de estado previo de PORTB

    while(true)
    {
                  'Mientras no se
       produzca una interrupción
       el programa se ejecuta en
       un loop infinito
                  'Aquí va el código
              que atiende a la
              tarea principal
    }

end.
```

Con este ejemplo general, prácticamente puede aplicarse el uso de interrupciones en todos los casos en los que sea necesario.

Pruebas tradicionales para robots autónomos

A continuación presentaremos un conjunto de pruebas clásicas para los robots autónomos móviles, junto con algunos consejos desde el punto de vista mecánico, electrónico y de la programación. Para construir las pistas de prueba, aconsejamos utilizar lona vinílica blanca y realizar los circuitos con pintura negra especial para esta superficie. Puede ser costoso pero es más sencillo de transportar que la madera u otros materiales. En el peor de los casos, unas buenas impresiones en láser empalmadas con cinta transparente pueden ser suficientes. Lo importante es que los colores sean muy distinguibles, y que no haya ondulaciones o muescas en la pista que traben el funcionamiento de nuestro robot.

Seguidor de línea

Ésta es la primera prueba por excelencia. En principio, comenzaremos por utilizar un solo sensor de luz que apunte hacia abajo para recorrer la pista. Al momento de construir la mecánica de nuestro robot, debemos tener en cuenta que debe girar con facilidad. Esto es fundamental cuando las curvas son muy cerradas. Por lo tanto, la arquitectura más

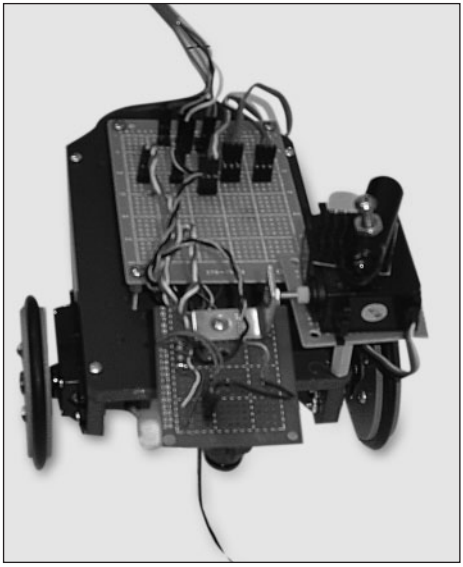


Figura 7. *Robot seguidor de una línea muy delgada. ¡Eso es complicado!*

económica y confiable para este desafío es la de las ruedas laterales con movimiento diferencial.

Si se cuenta con un único sensor, la solución más sencilla es la que conocemos como **ciego con bastón**, dado que aplica la misma mecánica que los no videntes cuando tantean paredes o el piso con su bastón blanco. Ponemos el robot a la derecha de la línea, con el sensor que apunte a blanco, encendemos el motor derecho (MD), y dejamos apagado el izquierdo (MI). Cuando el sensor detecta negro, invertimos el encendido de los motores, dejamos apagado el MD y encendido el MI.
Allí esperamos hasta encontrar blanco y así sucesivamente. Es un mecanismo sencillo, efectivo, pero muy lento (**Figura 8**). Para mejorar la velocidad de recorrido con un único sensor, si la línea negra a seguir es delgada, podemos realizar el siguiente algoritmo:

1) Ponemos el robot con el sensor a la derecha de la línea negra.
2) Encendemos el MD y apagamos el MI.
3) Esperamos a ver negro con el sensor.
4) Esperamos a ver blanco con el sensor.
5) Encendemos el MI y apagamos el MD.
6) Esperamos a ver negro con el sensor.
7) Esperamos a ver blanco con el sensor.
8) Saltamos al paso **2**.

De esta manera, mejoramos la velocidad de recorrido, pero es más sensible a las curvas cerradas (**Figura 9**).

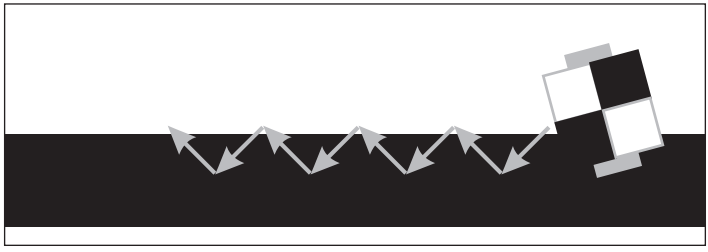


Figura 8. *Método de seguimiento de líneas versión 1.*

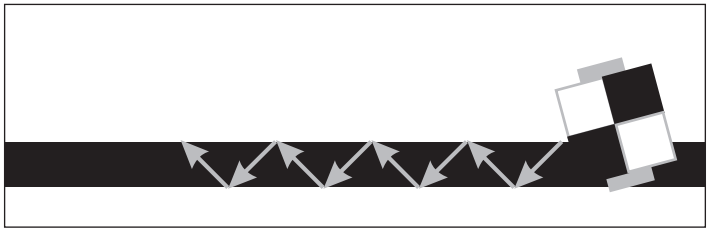


Figura 9. *Método de seguimiento de líneas versión 2.*

Si podemos contar con dos sensores, las mejoras pueden ser considerables. En este caso, ubicamos un sensor del lado izquierdo de la línea y otro del lado derecho, ambos sobre blanco. En el programa principal, sólo tenemos un ciclo infinito con los dos motores prendidos. Además, conectamos los sensores a alguno de los pines de RB4 a RB7, para atenderlos mediante una interrupción. Cuando uno de los dos vea negro, ésta se lanzará. Analizamos cuál de los dos cambió de estado, y definimos los estados de los motores de manera tal que el robot gire para el lado correspondiente hasta que el sensor vuelva a ver blanco. Allí retornamos de la interrupción y se vuelven a encender ambos sensores. Este mecanismo es muy efectivo, y podemos mejorarlo con más sensores, y usar algunos sobre negro y otros sobre blanco.

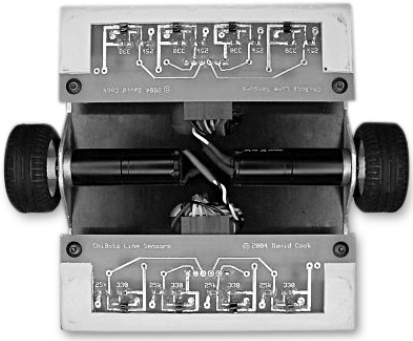


***Figura 10.** Sensores en un robot seguidor de línea creado por David Cook.*

ROBOCUP JUNIOR

La RoboCup Junior es una división de la RoboCup dedicada a estimular el estudio de la robótica en los jóvenes de hasta 19 años, y organiza las competencias para estas edades dentro de la RoboCup. Los desafíos presentados son tres: fútbol, rescate y danza. Está enfocado en la educación y no en la competencia. Los robots que participan deben ser autónomos y deben cumplir con límites de peso, tamaño, etcétera. Su sitio oficial es **www.robocupjunior.org**.

Si la línea posee espacios en blanco (**gaps**), nos conviene utilizar un mecanismo de un solo sensor. Además, debemos activar el timer 0 y establecer que si durante determinado período no se ve negro con alguno de los dos sensores, el robot debe avanzar en línea recta hasta encontrar negro nuevamente (**Figura 11**).

Otra prueba típica es el seguimiento de una línea con obstáculos. En este caso, podemos sumar un sensor de toque en el frente del robot, también conectado a una de las entradas que activan interrupciones. De esta manera, ante la detección del objeto, detenemos el seguimiento de la línea, esquivamos el obstáculo y, al volver sobre la línea negra, retomamos la rutina. Una precaución a tomar es que

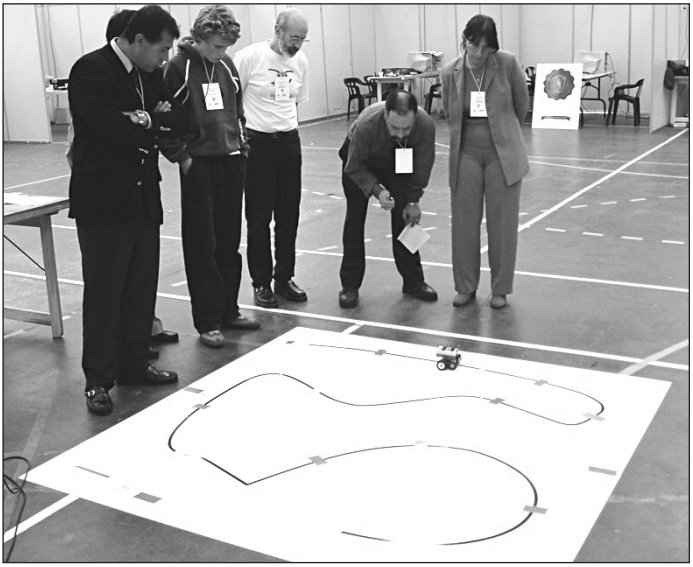


Figura 11. *Seguimiento de línea con detección de víctimas y saltos en las marcas (Roboliga).*

cuando volvamos sobre la línea, el robot quede en una posición adecuada para continuar con el seguimiento.

¡La luz, la luz, he visto la luz!

Otro desafío típico está relacionado con la búsqueda o el seguimiento de una fuente de luz. El problema más sencillo es posicionar el robot en un lugar y que gire hasta encontrar una luz que hemos ubicado cerca de él. Para ello, es necesario utilizar un sensor analógico que nos permita detectar los diferentes grados de luminosidad que tenemos en el ambiente (y por lo tanto, el 16F88 por su conversor). El robot debe tomar el valor del sensor en la posición en la que se encuentra y girar lentamente en alguno de los dos sentidos. Si aumenta el valor de luz hacia ese lado, mantiene su giro. Si no, intenta para el otro lado. En el momento en que al girar para ambos lados disminuya el valor de luz, es que hemos encontrado la fuente de luminosidad.

Otra prueba algo más compleja es seguir una luz originada por una linterna o por alguna pelota que emita una señal infrarroja, como las que se utilizan en fútbol de robots junior. Para ello es útil que usemos dos sensores analógicos conectados a pines de este tipo. Si el valor de lectura de ambos sensores supera cierto umbral, consideramos que la fuente de luz está frente al robot y nos dirigimos hacia delante. Si la diferencia entre el valor de

uno y otro sensor es grande, giramos hacia el lado donde se recibe el mayor valor. Si no hay una diferencia importante y no están ambos iluminados, consideramos que la luz está detrás y giramos en cualquiera de los dos sentidos. Podríamos seguir con muchas pruebas de este tipo, como realizar un robot insecto que busque el lugar con mejor luz de la sala, otro que no se caiga de una mesa, etcétera. ¡Todo queda librado a nuestra imaginación! Como hemos visto, a medida que nuestras pruebas se vuelven más complejas, es necesario un micro más poderoso, como el 16F88.

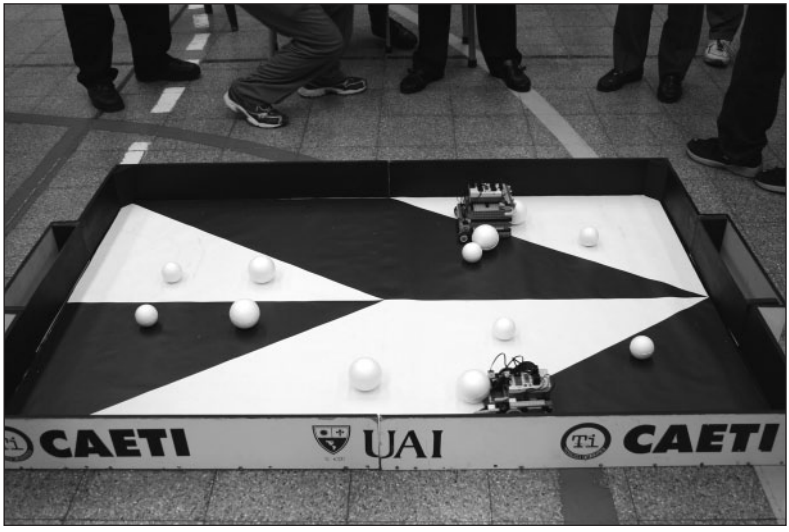


Figura 12. *Competencia de rescate de la Roboliga, Olimpíada Argentina de Robótica.*

BEAM ROBOTICS (ROBOTS BEAM)

BEAM es el acrónimo de **Biology, Electronics, Aesthetics and Mechanics** (Biología, Electrónica, Estética y Mecánica). Es un término que se refiere a la construcción de robots que usan circuitos análogos en vez de digitales, que tienen un diseño simple y que imitan a los organismos de la naturaleza. Los circuitos analógicos imitan a los bloques biológicos como las neuronas. Las tres reglas principales son: usar el menor número de elementos electrónicos (y sencillos); reciclar y reutilizar componentes; y usar sistemas de energía radiante, como la energía solar. Dentro de los diversos diseños realizados, es un clásico el **Phototrope** (fotótropo), un pequeño robot que busca la luz dentro de un ambiente o escapa de ésta.

Luchador de sumo

Una de las competencias que más apasionan a los participantes, y que no exige demasiada capacidad de procesamiento por parte del robot, es la lucha de sumo. Hay diferentes reglamentos en el mundo del sumo robótico. Es interesante conocer el reglamento del Campeonato Uruguay o de Sumo de Robots, organizado por el Instituto de Computación de la Facultad de Ingeniería de la Universidad de la República de Uruguay. Éste se basa en el Torneo de sumo de robots FSI-ALL de Japón. Este campeonato es muy interesante, y en el sitio (**www.fing.edu.uy/inco/eventos/sumo.uy/index.html**) podemos bajar un simulador y **templates** en diferentes lenguajes para poder jugar en nuestra computadora. Luego, si deseamos participar de la competencia, podemos llevar nuestro propio robot, o utilizar uno provisto por los organizadores, que tiene las mismas características que el utilizado en el simulador.

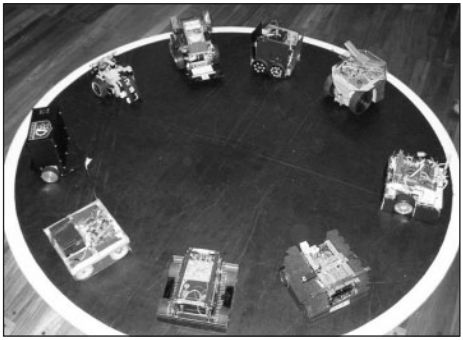


Figura 13. *Competencia de Sumo de la Universidad Tecnológica Nacional de Bahía Blanca, Argentina.*

La arena de competencia se llama **Dohyo**. Es un cilindro de 5 cm de altura y 154 cm de diámetro de un color distinto al blanco y al marrón (en general, negro). Las líneas de inicio, donde se ubican los robots en el comienzo de la lucha, se indican por líneas marrones de 2 cm por 20 cm, a 20 cm del centro del Doyho.

III CAMPEONATO DE SUMO ROBÓTICO DE URUGUAY

Desde el año 2004, el Instituto de Computación de la Facultad de Ingeniería de la Universidad de la República de Uruguay organiza el campeonato de sumo robótico. El objetivo de esta competencia es crear un espacio para investigar diversos aspectos de la robótica móvil autónoma. Las características particulares que presenta esta prueba son: un ambiente muy dinámico, comportamiento en tiempo real, y reglas sencillas y precisas. Una ventaja que tiene este campeonato es que no necesitamos de un robot para participar, dado que está provisto por la organización. Antes de la competencia, podemos programar el comportamiento de nuestro representante mediante un simulador y luego, en el momento de luchar, volcamos esa estrategia a un robot real.

El borde de la arena es de 5 cm de color blanco. Fuera del Doyho, el piso será de cualquier color excepto blanco o marrón. Cada categoría en particular determinará un límite para el tamaño y el peso del robot.

Una partida consta de tres sets de un minuto cada uno. El primer participante en ganar dos puntos **yuko** (ya veremos qué son) es el ganador de la partida. Cuando ningún participante recibe puntos, el jurado decide quién es el ganador. Para comenzar el partido, los robots se ubican sobre las líneas de inicio o detrás de éstas, para lo cual disponen de dos minutos. El robot no puede realizar ningún movimiento antes de recibir la señal de comienzo.

Los puntos yuko se consiguen por medio de alguna de las siguientes acciones: cuando un robot saca a su oponente del Doyho, cuando el oponente sale del Doyho por sus propios medios o cuando un robot es descalificado o recibe dos advertencias o más.
Un set se puede cancelar y repetir si los robots se encuentran trabados de forma tal que no exista movimiento posible, a criterio del jurado.
Si ambos robots tocan el exterior del Doyho al mismo tiempo o ante cualquier otra condición en la que el jurado no pueda definir un ganador, el set se reinicia, se ubican los robots en la línea de inicio, y se cuenta con dos minutos para hacerlo desde la indicación del árbitro.

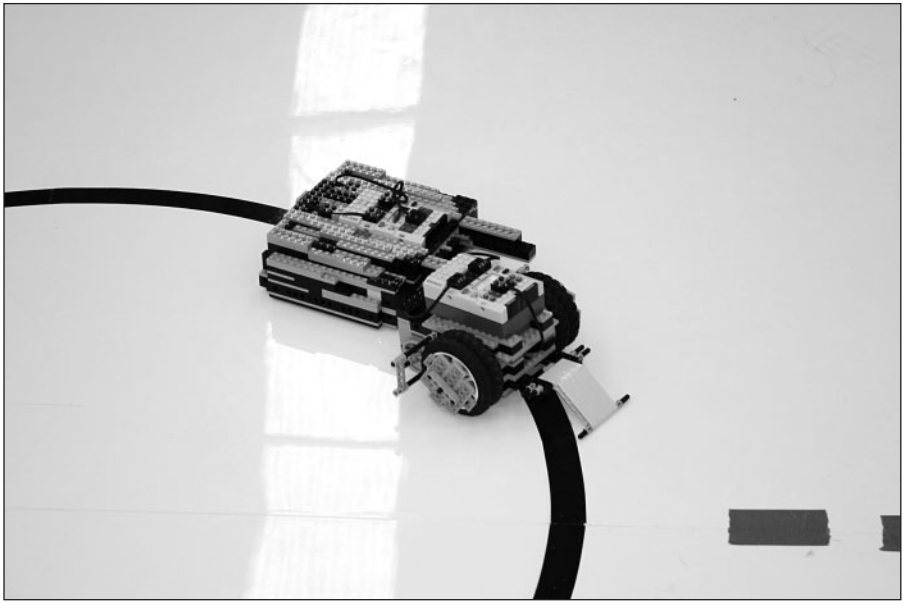


Figura 14. *Competencia tradicional de sumo en la Roboliga.*

En resumen, éstas son las reglas básicas de juego. Como vemos, las reglas no son complejas y esencialmente es un desafío mecánico y electrónico, con algoritmos sencillos para el control del robot. Los problemas que debemos resolver son tres: cómo detectar la línea límite del Doyho para no salir de ella (y que esa detección sea en toda la circunferencia del robot), cómo detectar la distancia al contrincante para poder abalanzarnos sobre él y cómo detectarlo cuando lo tocamos para ir hacia él o huir de él, según convenga. No agregamos el tema del posicionamiento dentro del Doyho porque es muy complejo de realizar con la información tan pobre que nos brinda la arena. En realidad, sólo es absolutamente necesario resolver el primer problema para sobrevivir en la lucha. En el caso de la detección del oponente, si no deseamos hacerlo, podemos recorrer en forma aleatoria el Doyho de un lado a otro para ver si logramos cruzar al otro robot en algún momento.

Desde el punto de vista mecánico, en general, si un robot tiene mucha fuerza, con seguridad será lento, como vimos en los capítulos anteriores. Por lo tanto, recorrer toda la arena será complicado. Es mejor tener un sistema de detección del contrincante para ir hacia donde se encuentra. Para ello, es bueno y no es complicado implementar el uso de un sensor de ultrasonido. Sólo con detectar la presencia (es decir, hacer un uso digital del sensor) nos alcanza para poner en movimiento el robot hacia el oponente. Con respecto a la detección de los límites del Doyho, alcanza con un sensor de luz de los que hemos usado. Tenemos que tener en cuenta que podemos encontrarnos con el límite no sólo si vamos hacia delante, dado que el contrincante puede empujarnos. Por eso sugerimos usar varios sensores que recorran el borde del robot. También necesitaremos sensores de tacto que rodeen a nuestro representante, de forma tal que al ser presionados por uno de los

III COMPETENCIA DE SUMO DE ROBOTS EN ARGENTINA

El grupo de robótica del Departamento de Ingeniería Eléctrica de la Universidad Tecnológica Nacional de Bahía Blanca, Argentina, todos los años organiza un campeonato de sumo de robots. Se presentan tres categorías, entre las que se destaca la categoría libre. El Doyho en este caso es de 1,75 m y los robots deben poder entrar en una caja de 20 cm por 20 cm. Además, no pueden superar los 3 kg de peso. El procesamiento se debe realizar por un microcontrolador de 4 Kb máximo y no se permite el uso de microprocesadores. Para mayor información podemos visitar **www.frbb.utn.edu.ar/robotica/**.

costados, huyamos o empujemos con más fuerza según la estrategia que hayamos desarrollado.

Un tema fundamental en la prestación mecánica de nuestro luchador es la potencia de los motores. Según los límites que presente la categoría, nos va a convenir utilizar los motores más potentes posibles, junto con un buen sistema de reducción. Para ello tendremos que contar con baterías poderosas, y por lo tanto, pesadas. Lógicamente, todo dependerá de cuáles sean los pesos y los tamaños que nos impongan las reglas del juego. Pero no dudemos en invertir en ello si queremos triunfar.

Otras pruebas clásicas

A continuación describiremos otras pruebas clásicas de robots móviles autónomos, que podremos profundizar si buscamos en la Web. En este listado, dejamos el fútbol de robots para el capítulo siguiente.

Tiralatas: en una arena rectangular o circular. Se ubica un conjunto de latas con cierto peso que el robot debe sacar en el menor tiempo posible.

Laberinto: el robot debe encontrar la salida del laberinto en el menor tiempo posible. En general, la salida está indicada con una luz, de manera tal que el robot pueda intentar realizar el camino más corto hacia ésta.

Carrera de postas: un conjunto de robots debe comunicarse entre sí para completar un recorrido definido por una línea negra.

Rescate: este tema incluye infinidad de pruebas (**Figura 15**).

Esencialmente, se basa en dos desafíos: detectar o rescatar un con-

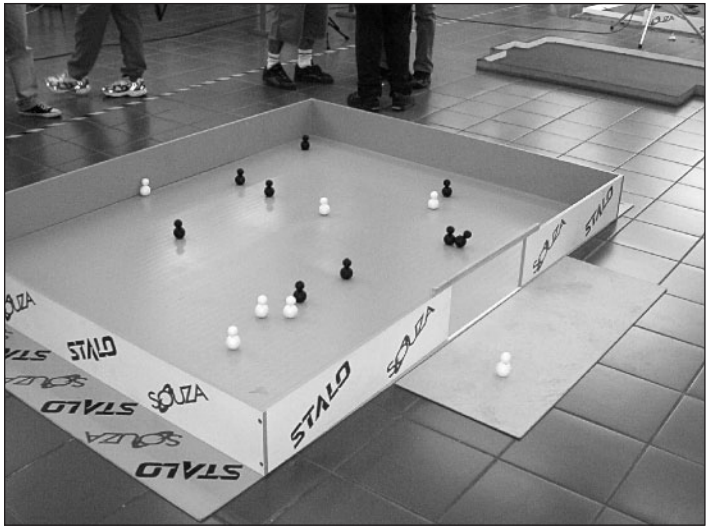


Figura 15. *Competencia de* ***rescate*** *del Campeonato Latinoamericano de Robótica IEEE.*

junto de víctimas en una zona de desastre, o rescatar víveres y medicinas que han caído en una zona contaminada para llevarlos a una zona liberada. Además, estas pruebas pueden ser solitarias o colaborativas.

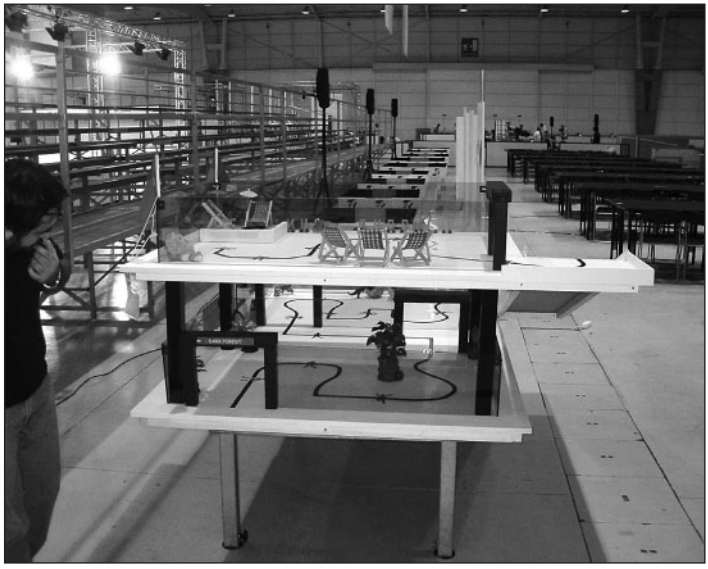


Figura 16. *Pista de la competencia de rescate de la RoboCup Junior.*

... RESUMEN

Ha llegado la hora de sacar nuestro robot al ruedo. Todas las pruebas son diferentes, pero hay un conjunto de características o de problemas habituales. Estos problemas pueden ser de índole mecánica, como el tipo de arquitectura elegido para nuestro robot o su punto de equilibrio. También tendremos problemas con los sensores, dado que el ambiente en el que se desarrollan los desafíos habitualmente difiere del que utilizamos para preparar a nuestros robots. Es por eso que tenemos que utilizar mecanismos robustos que puedan trabajar sobre valores relativos del ambiente y no con valores absolutos. Por otra parte, la ubicación de los sensores debe aislar al máximo las perturbaciones del entorno. Con respecto a la programación, presentamos el uso de interrupciones, que nos permitirán estar atentos a más de un sensor en nuestro programa. De las pruebas clásicas, el seguimiento de la línea, la búsqueda o el seguimiento de luz, y el sumo de robots, son las que nos permitirán poner en juego toda la capacidad y las características de nuestro robot.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Cuáles son los problemas mecánicos que puede presentar un robot en una competencia?

2 ¿Qué mecanismos tenemos para lograr que nuestro robot sea menos sensible a los cambios del ambiente en el que compite?

3 ¿De qué manera podemos visualizar el comportamiento de los sensores? ¿Cuál se usaría en un sensor digital y cuál en uno analógico?

4 ¿Para qué sirven las interrupciones?

5 ¿Cuáles son los tipos de interrupciones que podemos encontrar en un 16F84/88?

6 ¿Qué función cumple el registro INTCON?

7 Describa los algoritmos tradicionales de seguimiento de línea.

8 ¿Cuáles son los problemas que debemos afrontar en un robot luchador de sumo?

EJERCICIOS

1 Modifique el robot y prográmelo para que siga una línea negra con el método del ciego con bastón. Probar con diferentes grados en las curvas.

2 Realice un robot con dos sensores para realizar el seguimiento de la línea lo más rápido posible. Compare los tiempos con los logrados en el ejercicio anterior.

3 Ponga un obstáculo sobre la pista del punto anterior y realice un robot que pueda esquivarlo. El obstáculo debe ser algo pesado, como una caja con una piedra dentro.

4 Realice un robot que siga la luz de una linterna.

5 Realice un robot que recorra una mesa sin caerse de ella.

6 Realice un robot con un sensor de ultrasonido que, ante la cercanía de un objeto, cambie de dirección.

Jugar al fútbol

El fútbol de robots es una actividad donde se presentan todos los problemas de la robótica situada, en un marco divertido, mediático y de reglas conocidas a nivel internacional.

Es por eso que hemos decidido cerrar nuestro libro con este capítulo, al presentar las reglas y las características del desafío, para invitarlos a sumarse a la investigación dentro de esta maravillosa rama de la robótica.

FÚTBOL DE ROBOTS

Para el año 2050, desarrollar un equipo de robots humanoides completamente autónomos, que en un partido de fútbol puedan vencer al equipo campeón mundial humano.

Éste es el lema de la organización RoboCup, una de las dos ligas mundiales de fútbol de robots. Puede parecer un lema de ciencia ficción, pero puede dejar de serlo si realizamos el siguiente cálculo. En el momento en el que se lee este libro, ¿cuánto falta para el año 2050? ¿Cuarenta y tres, cuarenta y dos años? Bien. Ahora restemos del año actual la cantidad de años que faltan para el 2050. ¿Cómo eran las computadoras en esa época? ¿No era de ciencia ficción pensar en lo que tenemos hoy en día, los robots que se han desarrollado, Internet, las notebooks, los celulares, etcétera? Entonces, si vemos las cosas desde este punto de vista, tal vez pensar en un equipo de fútbol de robots que jueguen un buen partido de fútbol no es locura.

Por lo tanto, no perdamos el tiempo, empecemos desde ahora a preparar a nuestro pequeño. Enseñémosle los rudimentos de este deporte para que comience con sus prácticas de tiro libre, de penales, de amagues y fintas. Quién nos quita la ilusión de que en un futuro no muy lejano, lo acompañemos por todo el mundo en sus viajes con la Selección Mundial de Fútbol de Robots.

Figura 1. *Podemos imaginar un equipo de robots que enfrenten a un equipo humano en el año 2050. Hagamos apuestas.*

Características del fútbol de robots

Dentro del campo de la robótica autónoma móvil, en los últimos años ha surgido una disciplina conocida como **robótica situada**. Su objeto de estudio es el desarrollo de robots autónomos móviles en am-

bientes muy dinámicos. Ejemplos de esto son los vehículos terrestres, aéreos y acuáticos, los robots enfermeros y todos aquellos que tengan que desenvolverse en un mundo que cambia segundo a segundo.

El fútbol de robots es el deporte que presenta los problemas más importantes que podemos encontrar en la robótica situada y aunque existen diversas categorías donde estos problemas pueden ser más o menos significativos, de alguna manera u otra, están presentes en todas.

Para hacer una analogía con respecto al fútbol convencional, podemos decir que para jugar bien necesitamos las siguientes virtudes:

- **Un buen estado físico**: llegar antes a la pelota, patear fuerte y poder ir de un punto a otro de la cancha en forma veloz, nos permitirá tener un control del juego mucho más efectivo.
- **Una lectura precisa y veloz del juego**: es una característica fundamental que el jugador se anticipe a la jugada. Por ejemplo, que vaya al encuentro de la pelota donde la pueda ganar y también quede mejor parado para el gol. O que pueda saber dónde conviene estar parado para recibir un pase.
- **Juego colectivo**: el equipo no puede comportarse como un conjunto de niños que corre detrás de la pelota. Debemos tener un rol que cumplir, que ese rol sea dinámico y que el conjunto de roles determine el comportamiento del equipo optimizado para nuestro objetivo final: el gol en el arco contrario.

Éstos son los tres problemas que tenemos en el fútbol de robots, y en general en la robótica situada: la **arquitectura** de los robots, el **sensado** y el **procesamiento** de la información del ambiente y el **comportamiento** colaborativo. A continuación presentaremos cuáles son las dificultades en cada uno de ellos y cuáles son las soluciones propuestas.

III OTROS DEPORTES

No sólo de fútbol viven los robots. Ya se están desarrollando nuevas criaturas para otros deportes u otras competencias como el sumo, que ya hemos mencionado, el basquetbol y la lucha libre, entre otros. En este último caso, hay muchos tipos de torneos pero ¡cuidado! En general, las luchas que pasan por la televisión no son de robots autónomos, sino que están controlados por un ser humano. En este caso, se enfrentan problemas mecánicos y electrónicos, pero se carece de sistemas de inteligencia y sensado del mundo dado que esa función la cumple la persona que controla al robot. Por lo tanto, eso no es robótica situada.

Arquitectura y navegación de los robots

Como hemos visto antes, con frecuencia en las competencias tenemos un límite vinculado al peso y el tamaño de los robots. En el fútbol ocurre lo mismo. Por lo tanto, nuestro objetivo es lograr el robot más robusto y más rápido en el pequeño espacio que nos brindan. Lamentablemente, en el caso de la electrónica y los motores, más pequeño significa más costoso, y es aquí que nuestro bolsillo será otro límite que deberemos contemplar.

Con respecto a los robots para fútbol, la arquitectura clásica consiste en: **patas** (para ciertas categorías específicas), **omnidireccional** y **diferencial**. En el caso de los robots con navegación omnidireccional (**Figura 2**), las ventajas son conocidas, pero incorporar tres motores en el espacio que tenemos, más uno de un pateador si fuera necesario, es complejo y el valor de los motores y las ruedas puede ser costoso.

En general, para comenzar utilizaremos navegación diferencial (**Figura 3**), como la que tenemos en nuestro robot. Por lo tanto, tenemos que analizar cómo realizamos la navegación con ruedas diferenciales. Esta navegación dependerá de la información que podemos extraer del ambiente. En el caso de los robots más sencillos, pro-

Figura 2. *Un jugador de fútbol del equipo alemán FU-Fighters con ruedas omnidireccionales.*

bablemente nuestro funcionamiento será menos proyectado y calculado, más primitivo. Cuando tenemos más información del ambiente, como la posición X e Y de la pelota, y X, Y y la rotación de nuestros robots, podemos realizar un cálculo de navegación más preciso. En este punto tenemos dos situaciones que estudiar: la **navegación** hacia un punto fijo y hacia un punto móvil (como puede ser la pelota). La diferencia sustancial en los dos casos es que en el caso del punto móvil, si es posible, debemos realizar una proyección del comportamiento del punto en el futuro para navegar hacia esa proyección, de la misma forma que cuando nos lanzan un centro en el fútbol real corremos hacia donde caerá la pelota, y no detrás de ella. De esta manera, el sistema de navegación para los dos casos será el mismo, sólo que en el punto móvil nuestro destino final será un punto calculado a partir del historial del punto y no del estado actual. Desde ya que este mecanismo sirve para los objetos móviles con comportamiento relativamente lineal, como es el caso de la pelota. En el caso de los robots contrarios, no tiene demasiado sentido dado que es imposible predecir el futuro de su comportamiento.

Un algoritmo de navegación trivial es girar hasta ver el punto de destino (siempre con un umbral de error porque si no la misma inercia de movimiento nos hará pasar de largo en el giro y nos quedaremos oscilando para corregir el error), y luego del giro, avanzar hacia el punto. Al tener definido un umbral de error, cuando avancemos hacia el destino, llegará un momento en el que superaremos ese umbral, y por lo tanto, el robot volverá a girar para corregir su desvío, y así sucesivamente. También tendremos que determinar un umbral de distancia donde consideraremos que hemos llegado al punto para no oscilar yendo y viniendo hacia el punto de destino.

III PROYECCIÓN DE LA PELOTA

Está claro que en el fútbol es fundamental llegar a la pelota lo antes posible. Para ello, es necesario ir hacia el punto de cruce con ella que más nos convenga. Es decir, proyectar el comportamiento de la pelota a futuro para determinar cuál es el mejor momento de cruzarla. La linealidad en el movimiento de la pelota hace sencillo esto, si no fuera por la presencia de las paredes, nuestros robots y los contrarios. Con respecto a las paredes, podemos considerarlas en varios pasos futuros porque están allí en forma constante. Pero con respecto a los otros robots, después de una proyección de 4 ó 5 cuadros, nos conviene dejar de tenerlos en cuenta, porque no sabemos con precisión dónde estarán ubicados.

La navegación que acabamos de presentar no es complicada de implementar, pero dista mucho de ser rápida y eso, en el fútbol, es un problema. Para optimizar nuestra navegación tenemos que hacer un análisis profundo de la física implicada en el juego. Es decir, tenemos que determinar cómo funciona la inercia del robot, cuánto tarda en alcanzar diversas velocidades, cuánto tarda en realizar giros de diversas amplitudes partiendo de diferentes estados, etcétera. En esto influirá la arquitectura del robot (más allá de que sea diferencial, cómo está repartido el peso del robot y cómo se ubican las ruedas son puntos esenciales), los materiales de las ruedas y el piso de la cancha. Como vemos, tenemos que invertir un largo tiempo de estudio para realizar pruebas en el campo con el objetivo de pulir la navegación. Por último, tenemos que poder desarrollar un modelo donde para llegar a un punto determinado, en lugar de girar y luego ir en forma recta, determinemos velocidades diferentes en las ruedas para trazar una curva optimizada hacia el destino. Una vez hecho esto, podemos seguir nuestro estudio mediante la realización de navegaciones donde nos interese con qué ángulo llegamos al final, desde qué lado llegamos al punto, etcétera.

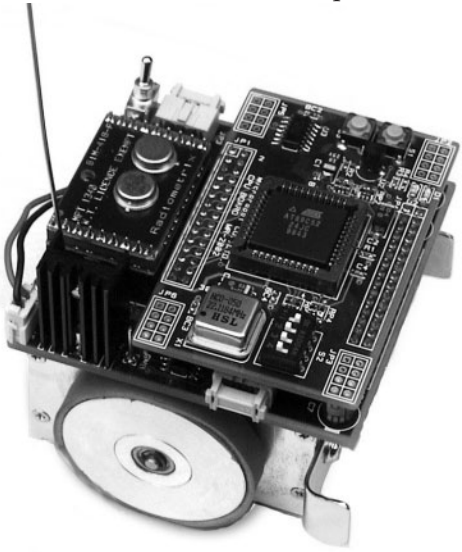


Figura 3. *Ejemplo de robot de fútbol con ruedas diferenciales.*

Sensado y detección del ambiente

Este punto es fundamental. Aquí tenemos mucho trabajo apasionante por delante. En el caso de usar sensores sencillos, como ultrasónicos, sensores de luz, de tacto y otros sensores locales, la información que obtendremos es bastante imprecisa. La imagen que conseguiremos del ambiente no nos permitirá navegar con total exactitud. Todo lo que podremos hacer es buscar la pelota con algún sensor, acercarnos a ella, y una vez que nos encontramos cerca, intentar llegar hasta el arco contrario.

Lo que hemos utilizado para obtener mayor precisión en la captación del ambiente son los sistemas de visión. En este caso, tenemos que realizar un delicado equilibrio entre la cantidad

Figura 4. *Sistema de* ***visión global*** *en una cancha de fútbol de robots. Podemos ver las cámaras conectadas en la parte superior.*

de cuadros por segundo que procesamos y la precisión que obtenemos al final. Aquí encontramos la gran diferencia entre los sistemas de visión industriales tradicionales y los que tenemos que aplicar en robótica situada. En el primer caso, podemos tomar mayor tiempo de procesamiento, con lo cual el resultado final es mucho más preciso. En nuestro juego, tenemos que complementar y corregir la información de la cámara con datos previos del ambiente y con proyección del estado actual en el caso de los objetos de comportamiento lineal. Y desde ya, renunciar a la precisión si queremos obtener el estado del ambiente actual, ¡y no el que existía hace 30 segundos!

El **sistema de visión** puede ser **global** o **local** a los robots. En el primer caso (**Figura 4**), un procesador externo realiza el procesamiento y luego les envía el resultado a los robots para que tomen sus decisiones, o directamente éste u otro procesador externo, a partir de los datos captados, define la estrategia y les envía órdenes a los robots como la velocidad de las ruedas, etcétera. En la visión local, cada robot capta el ambiente. Como decíamos antes, si esto se ha-

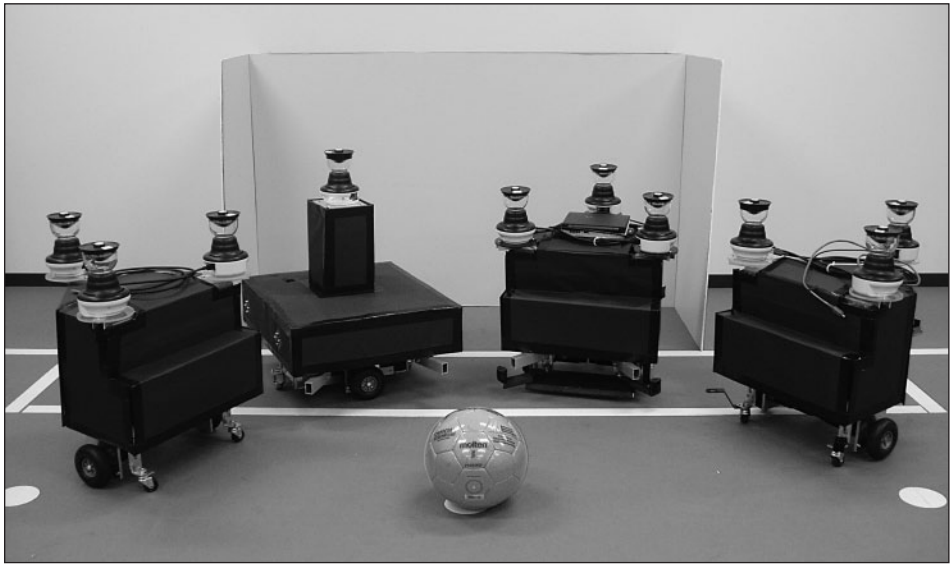


Figura 5. *Ejemplos de robots seguidores de una pelota infrarroja.*

III VISIÓN INDUSTRIAL

Con un poco más de dinero, podemos adquirir una cámara de visión industrial con salida **firewire** para que en nuestra computadora ingresen los datos en forma digital y no sea necesario un **framegrabber** (prácticamente todas las computadoras ya vienen con entrada firewire, y de lo contrario compramos una placa con esa entrada). Un aspecto importante es que la lente tenga un montaje (la forma de conectarse a la cámara) que sea estándar, como el **montaje CS**, de manera tal que podamos cambiar de lentes según el ambiente y la distancia a la cancha. Otra característica interesante es la posibilidad de aplicar filtros desde el hardware, lo cual hace mucho más rápido el proceso.

ce mediante sensores, la calidad de la información es bastante baja. Si en el robot tenemos un sistema de visión, los datos que tenemos del ambiente son mayores, pero esto exige que cada robot realice el arduo procesamiento de la imagen. Por otra parte, la visión local en general es parcial, porque los robots apuntan a una dirección determinada. Para superar esto, es habitual que la cámara se apunte hacia arriba y que se coloque un espejo cónico o semiesférico en la parte superior para tener una captura de 360 grados (**Figura 6**). La deformación que se produce en esta imagen es muy fácil de solucionar.

Aun así, el robot puede tener información incompleta porque otros le tapan la pelota, el arco o algún otro objeto. Es

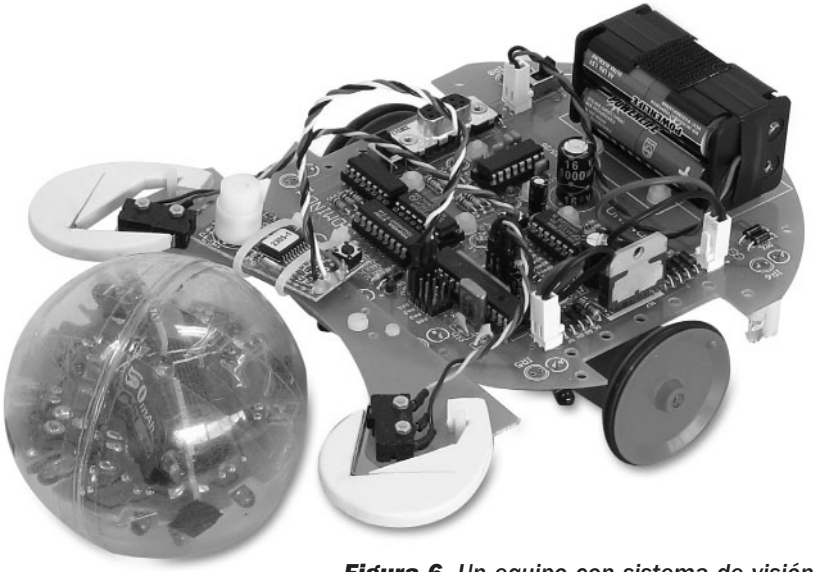


Figura 6. *Un equipo con sistema de visión local. La cámara apunta hacia una semiesfera en la parte superior.*

por eso que en todos los sistemas de visión, complementamos lo que captamos en el cuadro con información del historial, como haríamos con nuestros propios ojos y cerebro. Si vemos que viene la pelota y en un momento la dejamos de ver porque queda detrás de un jugador, realizamos una proyección de su movimiento y prevemos encontrarla en un punto posterior, ya que conocemos cómo se estaba moviendo y podemos anticiparnos a su comportamiento futuro. De la misma manera lo hacemos (¡o por lo menos lo intentamos!) en el fútbol de robots.

Comportamiento colaborativo

Además de todo lo que vimos, necesitamos que jueguen en equipo. En un conjunto de robots que tienen un objetivo determinado, es necesario algún mecanismo de organización que permita que cada uno realice una o más tareas determinadas en forma dinámica, en pos de ese objetivo. Existen muchas formas de lograr esto, pero plantearemos dos: el **comportamiento emergente** y el **comportamiento dirigido**.

El comportamiento emergente dentro de un sistema es el que surge de la interacción de sus componentes, y no puede explicarse a partir de los comportamientos individuales. Por ejemplo, ninguna hormiga puede realizar una colonia por sí misma, ni organizarla dándoles órdenes a otras hormigas. Pero su comportamiento individual sencillo en conjunto con las otras hormigas, les permite lograr un tra-

bajo en equipo muy complejo (**Figura 7**). Otro ejemplo es el de las bandadas de pájaros. Muchas veces nos quedamos maravillados ante las hermosas figuras geométricas que logran al volar en grupo. Sin embargo, ninguna mente superior ordena ese vuelo. Un conjunto de reglas sencillas de ubicación de cada pájaro en la bandada, que permite minimizar la resistencia del aire, es la que logra esa organización que nos sorprende. Un último ejemplo que podemos presentar es el de la generación del aplauso y su ritmo. En un recital o en una obra de teatro, no nos organizamos para aplaudir. Alguna señal, como el final de la canción o de un acto, genera un primer aplauso, que luego comienza a crecer y que, mágicamente, toma una cadencia general a partir de la sencilla actitud de cada uno de nosotros al golpear nuestras manos.

En un grupo de robots, podemos determinar cierto conjunto de acciones primitivas, como ir a un punto o ir a la pelota, y que cada robot decida por sí mismo qué hacer según las condiciones del ambiente. Por ejemplo, cada robot, según la cantidad de cuadros que tardaría en llegar a la pelota, podría decidir si va hacia ella o no. De la misma forma, podríamos determinar un grupo de acciones posibles que estén condicionadas por el estado del robot dentro del ambiente. El proble-

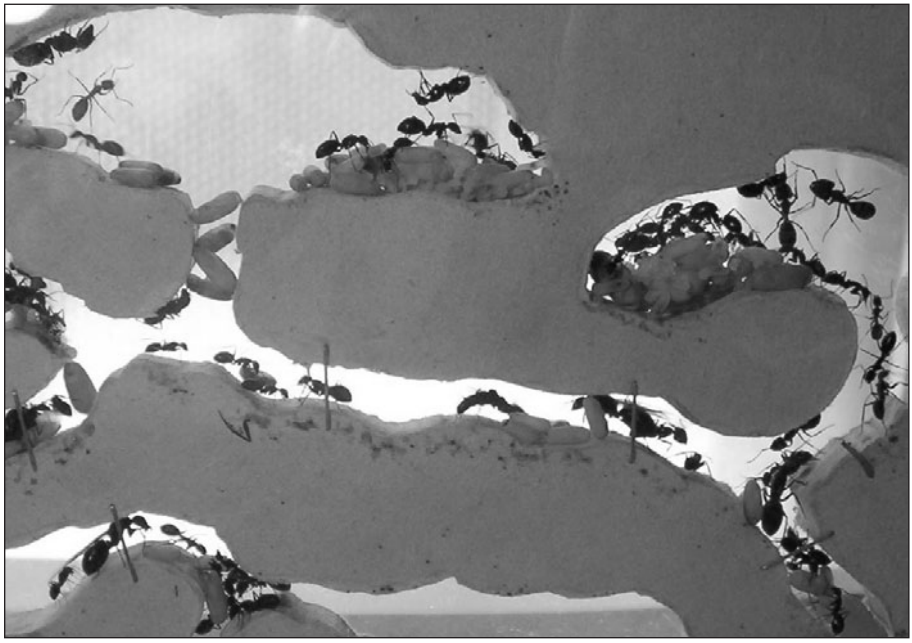


Figura 7. *Las hormigas, un maravilloso ejemplo de comportamiento emergente.*

ma que podemos tener es que varios robots decidan hacer lo mismo o que ninguno ejerza alguna función vital, como la de arquero. La ventaja es que con sólo agregar alguna o algunas acciones más, y si pulimos el sistema de decisiones de cada robot, el comportamiento puede cambiar en forma notable. Además, todos los robots son iguales, y por lo tanto no necesitamos programar diversos comportamientos para cada uno de ellos.

El comportamiento emerge en forma espontánea. Es muy sorprendente verlos organizarse por ellos mismos y cumplir determinados roles cuando nosotros nunca lo hicimos en forma explícita. ¡Es algo mágico! El problema es que muchas veces, este comportamiento no surge en forma tan precisa ni veloz como lo necesitamos en un ambiente dinámico. Y es en ese momento cuando pensamos que tener un director de orquesta puede presentarnos un mundo más gris y represivo, pero infinitamente más ejecutivo.

Así surge el **comportamiento dirigido**. En este esquema, un robot del equipo, o un agente externo, toma decisiones, elige una estrategia y le asigna un rol a cada robot. En el caso de los sistemas de visión global, donde un computador externo realiza el procesamiento de imágenes, el mismo procesador, o uno que está comunicado con él, toma estas decisiones y sólo comunica a cada robot la velocidad que debe tomar su rueda izquierda y derecha. De esta manera, la asignación de roles y el comportamiento es muchísimo más preciso y las decisiones se toman en forma veloz. Pero es necesario que imaginemos todos los mundos posibles, porque nuestro conjunto de robots nunca saldrá de lo puramente establecido por nuestro algoritmo. Todo comportamiento dependerá de lo que hayamos previsto

SEGUIR EL GUANTE BLANCO DE MICHAEL JACKSON

El comportamiento emergente también puede aplicarse para el seguimiento de objetos en un video. El **White Glove Tracking Project** (Proyecto de Seguimiento del Guante Blanco) fue un proyecto respaldado por Rhizome.org, en donde un gran número de usuarios debían marcar el lugar donde veían el guante blanco de Michael en un video de 5 minutos y medio. Con la información de cada uno, que individualmente podía tener un buen margen de error, se determinó la posición con una precisión completa. Es decir, a partir del comportamiento sencillo de seguir el guante con el puntero del mouse, el software levantó los datos en tiempo real y los ponderó para definir la posición perfecta del guante. Éste es otro ejemplo de comportamiento emergente.

en nuestra programación. En realidad, en general usamos una combinación de éstos y otros mecanismos de organización en equipo. La elección de uno u otro dependerá, en especial, de los buenos resultados que logremos con ese comportamiento.

Ligas nacionales e internacionales de fútbol de robots

Como dice el sabio adagio, todo lo que hacemos los investigadores es para conquistar a las muchachas que en nuestra juventud, con nuestro aspecto de nerds, nunca pudimos seducir. Debido a que pasamos días enteros detrás de nuestros inventos, es fácil suponer que nuestro estado físico deja bastante que desear. Es por eso que decidimos crear los campeonatos de fútbol de robots: queremos ser famosos, reconocidos por la calle y conseguir novias modelos, como los jugadores de fútbol reales. Luego de doce años de competencias no hemos logrado ninguno de estos objetivos, pero no hemos perdido las esperanzas. Por lo menos, en las reuniones sociales hemos logrado algunos minutos de atención al comentar nuestra extraña profesión.

Ahora, hablando en serio, los campeonatos de fútbol de robots son una buena excusa para intercambiar conocimientos con otros investigadores de la misma disciplina. Las reglas del fútbol son conocidas a nivel internacional y no hacen falta demasiadas palabras para que nos podamos encontrar en un partido. Y luego, entre goles y tiros libres, compartimos los últimos resultados de nuestros trabajos en los campos que presentamos en el punto anterior. A continuación comentaremos cada una de las ligas y las categorías vinculadas al fútbol que presentan.

RoboCup

Según se detalla en su propio sitio, la RoboCup es un proyecto de colaboración internacional para promover el desarrollo de la robótica, la inteligencia artificial y los campos afines. Fomenta la investigación de estos temas y provee problemas estándares, donde una amplia gama de tecnologías puede integrarse y ponerse a prueba. Ha elegido el fútbol de robots como tema central con el objetivo de aplicar los resultados de las investigaciones a problemas sociales significativos y a la industria. La meta final es lograr hacer un equipo de once robots autónomos para el 2050, que le gane un partido a la última selección campeona del mundo.

Dado que muchas de las tecnologías estudiadas son aplicables a la búsqueda y el rescate de personas en situaciones de desastre, se ha agregado a las competencias de rescate (RoboCup Rescue) como una forma de aplica-

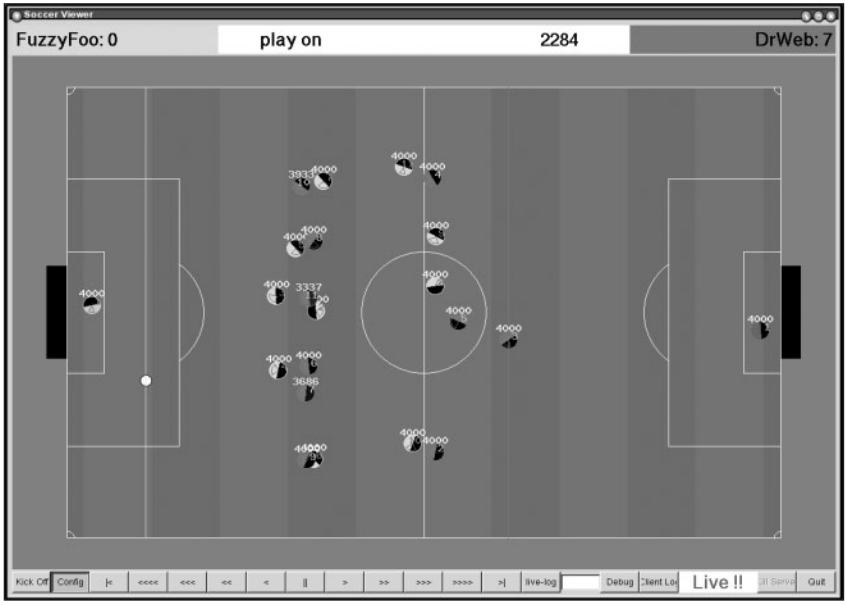


Figura 8. *Imagen de uno de los monitores para el simulador de la RoboCup.*

ción significativa de los resultados de las investigaciones. Además de las categorías de fútbol, también presenta categorías físicas y simuladas de rescate y categorías de fútbol, rescate y danza para jóvenes entre 10 y 18 años. Las categorías de fútbol que ofrece son:

- **Simulación**: en estos momentos hay dos subcategorías, una de 2D y otra de 3D. En esta categoría, debemos controlar los once jugadores y al entrenador, cada uno con un proceso independiente entre sí y sin comunicación directa entre ellos. El partido se desarrolla en un simulador de la física del ambiente, con el cual los procesos que controlan al equipo se comunican mediante mensajes de texto por sockets. Además, contamos con un visualizador gráfico del ambiente. Es decir, cuando se juega un partido tenemos 26 procesos que corren: 22 jugadores, 2 entrenadores, 1 simulador y 1 visualizador.
  La comunicación entre los jugadores y el técnico se realiza mediante gritos entre ellos. Por lo tanto, tenemos un comando que podemos ejecutar en nuestro agente que grita un mensaje y desde ya, también puede escucharlo. Además, tenemos comandos de movimiento, pateo, giro, etcétera. El servidor, además de recibir los comandos para los robots, nos provee información de sensado (particular de cada

agente), como lo que ve, la energía del jugador, la velocidad, etcétera. Esta información de los sensores puede estar contaminada por el ruido (errores), como en un sensor real. Es importante recordar que el servidor no espera que le enviemos los comandos para los robots, sino que funciona como un sistema de tiempo real con ciclos discretos, y que aquí también tenemos el compromiso de equilibrar precisión y velocidad. Si nuestra heurística tarda demasiado tiempo en definir el comportamiento del robot, vamos a entregar acciones para un mundo que ha cambiado, y con una frecuencia demasiado lenta para la vitalidad del ambiente. El objetivo del simulador es poder solucionar todos los aspectos físicos y de visión, para que nos podamos concentrar en los temas vinculados al comportamiento colaborativo.

- **Small size** (tamaño pequeño), también conocido como F180: aquí tenemos un partido de fútbol físico entre dos equipos de cinco jugadores cada uno (**Figura 9**). Cada robot debe poder entrar en un cilindro de 180 mm de diámetro, con una altura no mayor a 15 cm, salvo que tenga su propio sistema de visión. De no ser así, se utiliza un sistema de visión global con comunicación inalámbrica entre el procesador de imágenes y los robots. El campo de juego es de

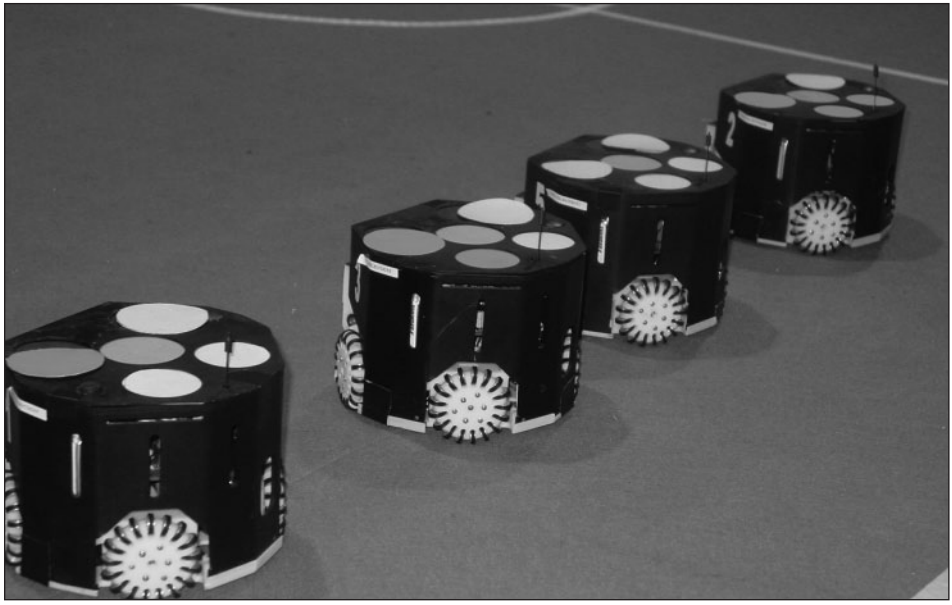


Figura 9. *Equipo Small size del ITAM de México.*

4,9 m de largo por 3,4 m de ancho y se utiliza una pelota de golf naranja. A pesar de que la visión puede ser local, todos los equipos que hemos visto en los torneos tienen visión global.

- **Middle size** (tamaño medio): en esta categoría física los robots también son autónomos en el sistema de visión. Cada equipo cuenta con cuatro jugadores que se comunican entre ellos en forma inalámbrica (**Figura 10**). No puede haber ninguna intervención externa, salvo para insertar o remover robots del campo de juego. Cada robot debe poder entrar en una caja de 50 cm de ancho, 50 cm de largo y 85 cm de alto. Por momentos, puede crecer hasta 60 cm por 60 cm cuando patea. El peso máximo del robot es de 80 kg.

- **Standard platform league** (liga de plataforma estándar): en esta categoría se selecciona un robot estándar que debe ser usado por todos los equipos. Hasta el año 2007, la plataforma seleccionada fue el Aibo de Sony, y por eso esta categoría se conocía como **four legged** (cuatro patas). En 2008 se utilizará el robot **Aldebaran Nao** (**Figura 11**). Cada equipo debe ser de cuatro jugadores autónomos mediante la utilización de esta plataforma básica. De esta manera, se pone el acento en el desarrollo del software para solucionar el aspecto físico del robot.

Figura 10. *Un partido a punto de comenzar en la Middle size de la RoboCup.*

- **Humanoid** (humanoide): aquí, los equipos deben estar formados por dos humanoides, donde uno será el arquero. Existen dos subcategorías (kidsize y teensize), donde sólo hay diferencias de tamaño. Cada robot debe tener proporciones en su cuerpo similares a las proporciones humanas (**Figura 12**). Las reglas proporcionan un conjunto de fórmulas que determinan los límites de mínimo y máximo para cada parte del cuerpo. Además de jugar partidos, deben realizar pruebas de destreza como esquivar obstáculos, recorrer caminos, etcétera.

Figura 11. *En esta imagen vemos un prototipo avanzado del Aldebaran Nao.*

Figura 12. Robotina*, el humanoide de la categoría TeenSize del equipo Nimbro de la Universidad de Freiburg.*

FIRA (Federación Internacional de Fútbol de Robots)

La FIRA es una liga futbolera. Los robots de sus diversas categorías son de complejidad menor que los de la RoboCup, con características menos cercanas al fútbol real. Por ejemplo, en sus canchas no hay laterales. La dinámica es más parecida al fútbol de salón. Con estas características, sus partidos son más vistosos y emocionantes, ya que la pelota está constantemente en juego y sus robots se mueven a una velocidad sorprendente. Las categorías que presenta son:

- **MiroSot**: en esta categoría, los robots tienen un límite de 7,5 cm de lado (sin incluir la antena de comunicación). Se juega con una pelota de golf naranja y la visión es global.

PLATAFORMA ROBOCUP

El robot Aldebaran Nao es un desarrollo de origen francés que comenzó en el año 2005 y fue presentado dos años más tarde. El objetivo que persigue es el de lograr un humanoide que se encuentre disponible para el público, de costo accesible, con las características que vemos a continuación:

- Cámara digital incorporada.
- Reconocimiento de voz.
- Sintetizador de voz.
- Expresiones de emoción.
- Comunicación por WiFi.
- Veinticinco grados de libertad.

Para obtener más información, podemos visitar la siguiente dirección: **www.aldebaran-robotics.com**.

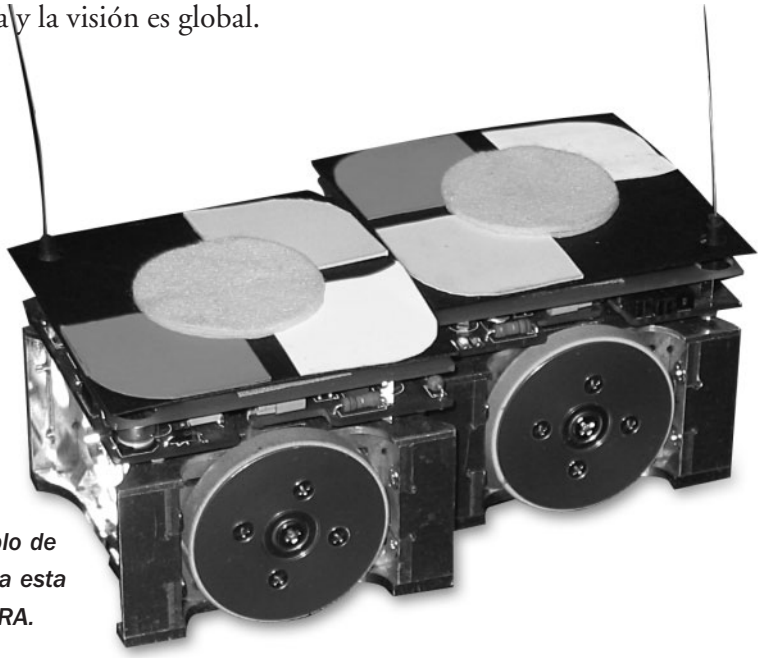


Figura 13. *Ejemplo de robot MiroSot para esta categoría de la FIRA.*

Existe una subcategoría de cinco contra cinco donde la cancha es de 220 cm por 180 cm, y otra de once contra once de 400 cm por 280 cm (**Figura 14**). La cámara colocada en la parte superior de la cancha captura las imágenes, que luego son procesadas por una computadora externa. Ésta última, una vez procesada la imagen y analizado el ambiente, determina el comportamiento de los robots y le envía las órdenes a cada uno en forma inalámbrica. Por el tamaño del robot y las reglas de cubrimiento de la pelota, es imposible capturarla, por lo que el transporte es bastante complejo, y se juega mucho con los rebotes y el transporte colectivo.

- **SimuroSot**: categoría simulada que emula a MiroSot de cinco contra cinco y de once contra once. El simulador (**Figura 16**) provee los datos del ambiente cada 16 milisegundos y nuestro equipo (una dll hecha en C++) toma sus decisiones e informa la velocidad de la rueda izquierda y derecha de cada robot.

- **NaroSot**: es una versión mini de MiroSot (**Figura 15**). Aquí los robots deben ser de 4 cm de ancho,

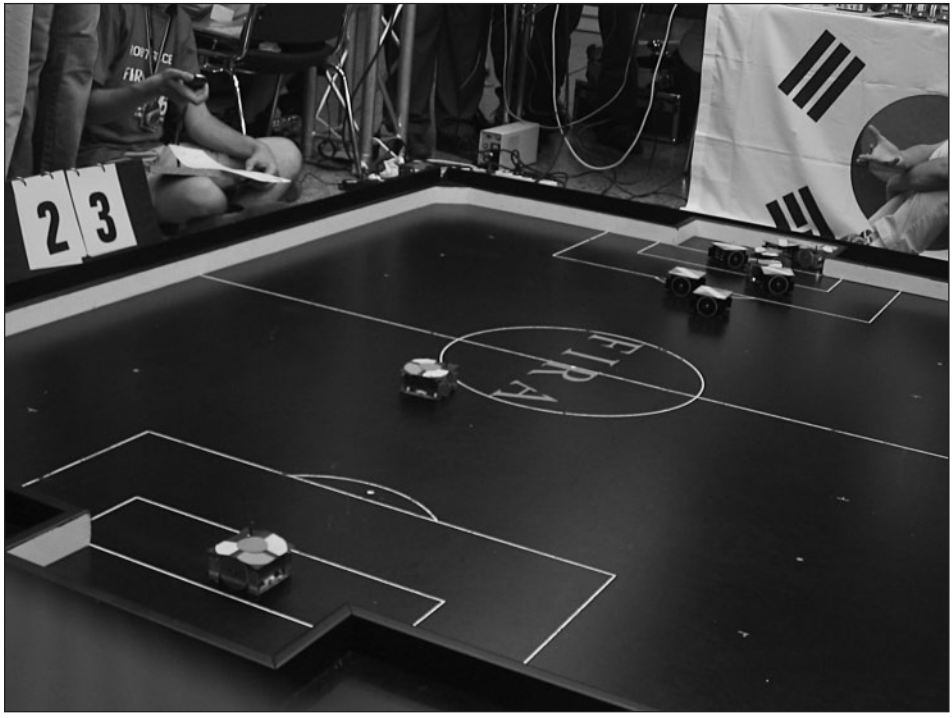


Figura 14. *Partido de cinco contra cinco en la categoría MiroSot de la FIRA.*

4 cm de largo y 5,5 cm de alto (sin contar la antena). El sistema de visión también es global y juegan cinco contra cinco con una pelota de golf naranja.

- **RoboSot**: ésta es la categoría con visión local. Cada equipo está formado por tres robots que deben caber en un cilindro de 20 cm de diámetro y con una altura no mayor a 40 cm. La pelota es de tenis

Figura 15. *Pequeños robots de la categoría NaroSot de la Universidad de Tecnología de Viena.*

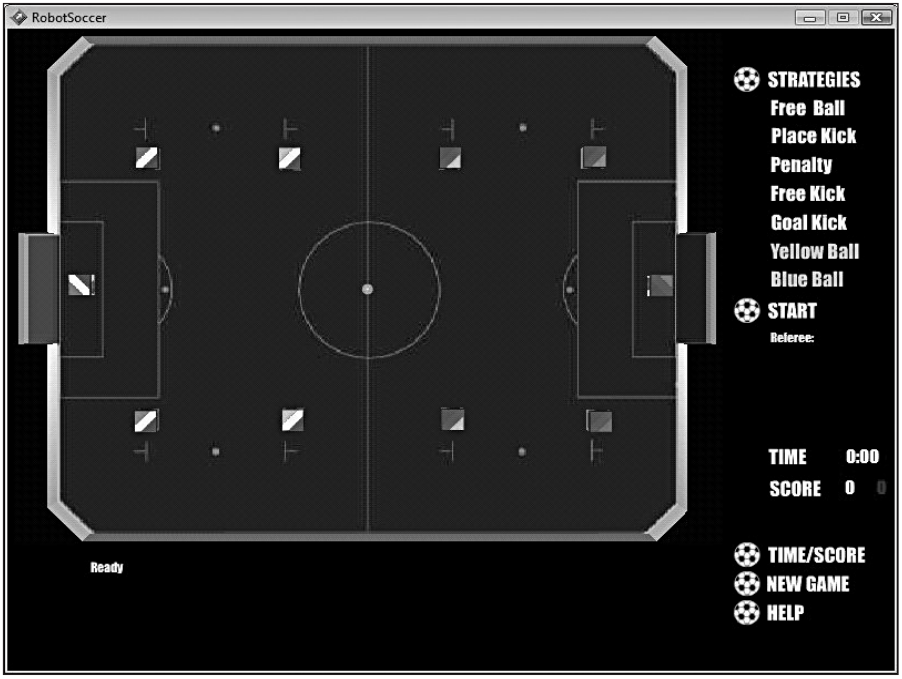


Figura 16. *Simulador de 5 contra 5 que se utiliza en la categoría SimuroSot de la FIRA.*

color amarillo y ya pueden incorporar a su arquitectura sistemas de acarreo de la pelota (con restricciones) y sistemas de pateo.

- **HuroSot**: ésta es la categoría de humanoides de la FIRA (**Figura 17**). Aún no se juegan partidos de fútbol, sino que el robot debe cumplir ciertas pruebas de destreza como realizar un recorrido, patear penales, etcétera. La altura máxima es de 150 cm y 30 kg de peso. Las proporciones del robot deben ser similares a las humanas, como en la RoboCup.
- **KepheraSot**: en esta categoría, que está en vías de desaparición, los equipos utilizan una arquitectura conocida como **Kephera**. Juegan uno contra uno con un sistema de visión local muy interesante, ya que posee una cámara con captura de una sola línea. Es por eso que los jugadores se identifican con un cilindro de rayas negras y blancas, y la pelota es de color verde.

Campeonato Latinoamericano IEEE

El Campeonato Latinoamericano IEEE comenzó en el año 2002 y no

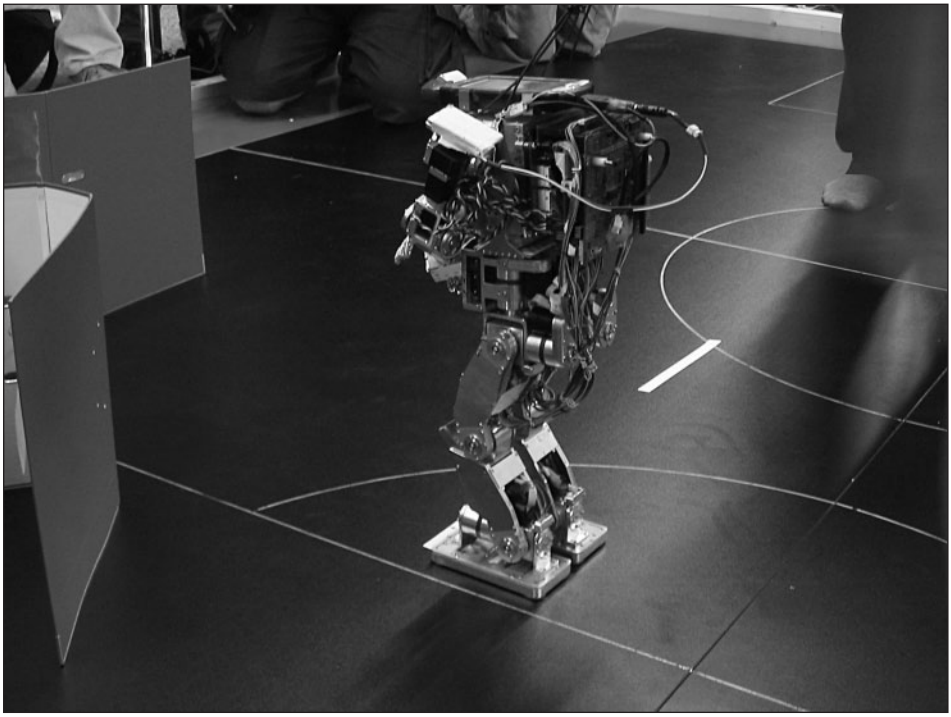


Figura 17. *Robot de la categoría HuroSot de la FIRA intentando resolver un problema.*

contaba con pruebas de fútbol de robots. Sin embargo, en las últimas tres ediciones ha incorporado la RoboCup Latinoamericana.

De esta manera, se hacen presentes las categorías **Small Size**, **Four Legged** y **Simulación**. Además, propone una categoría inicial donde se utiliza el kit de Lego para la construcción de los robots, con desafíos de rescate o colaborativos, y una categoría abierta, donde los robots deben ser desarrollados por los propios participantes, con determinadas restricciones en los procesadores.

Concurso Mexicano de Robótica

Versión local del Concurso IEEE Latinoamericano que agrega a sus competencias las categorías Junior de la RoboCup (para menores de 18 años) y un concurso de robots limpiadores, también con diversas categorías. Con respecto al fútbol en particular, están presentes Small Size y Junior.

Campeonato Argentino de Fútbol de Robots (CAFR)

El CAFR combina pruebas de ambas ligas internacionales. Con respecto a la FIRA, presenta su cate-

Figura 18. *El equipo argentino del* ***CAETI*** *recibiendo su premio por el cuarto puesto en la categoría SimuroSot.*

goría SimuroSot, donde tiene el mayor número de participantes. De la RoboCup presenta una versión más pequeña que la Small Size (con menos cantidad de robots) y la Junior con pelota infrarroja (**Figura 19**).

Roboliga (Olimpíada Argentina de Robótica)

La Roboliga es una competencia destinada a alumnos menores de 18 años. Es la competencia de robótica más antigua de Latinoamérica y conjuga pruebas de destreza con presentación de proyectos de investigación de los alumnos. Dentro de las competencias, que se modifican año a año, con frecuencia aparecen pruebas de rescate, sumo, y destrezas de fútbol de robots, que utilizan el mismo campo y las mismas características que la RoboCup Junior (**Figura 20**).

Modificaciones para que nuestro robot pueda jugar

Con nuestro robot podríamos participar de la categoría Small Size de la RoboCup y de la competencia de fútbol de la RoboCup Junior. En el primer caso, deberíamos desarrollar un sistema de visión para poder realizar el seguimiento de nuestro equipo y

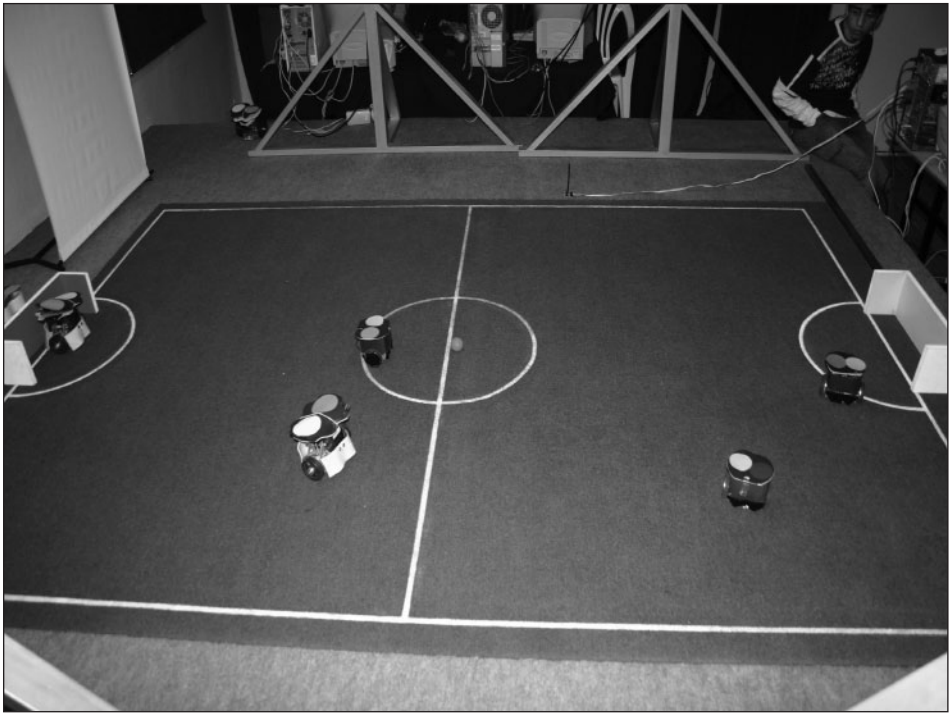


Figura 19. *Categoría Senior del Campeonato Argentino de Fútbol de Robots.*

sumar algún sistema de comunicaciones para enviar las órdenes de la computadora central hacia los robots (y desde ya, ¡programar todo esto!). Con respecto al **sistema de visión**, ya existe un servidor de procesamiento de imágenes de distribución libre y gratuita que nos puede ser útil: **Doraemon**. Fue desarrollado por el Departamento de Ciencias de la Computación de la Universidad de Manitoba, corre en Linux y necesita de una placa capturadora de video con entrada compuesta o SVHS. De las capturadoras de video que tienen drivers en Linux, nosotros logramos poner en marcha el sistema con aquellas que tienen **chip bt878**, como la antigua PCTV de Pinnacle. Conectada a esta placa, podemos utilizar cualquier cámara con las salidas mencionadas, como cualquier handycam que tengamos en nuestras casas. Hemos tenido varias luchas para que funcione, pero una vez que se logra darle arranque y se configuran los colores en forma correcta, nos provee los datos con buena precisión. Los datos del ambiente y de los robots los comunica mediante un socket UDP, por lo que podemos capturarlos desde otro proceso en el mismo Linux o en otra má-

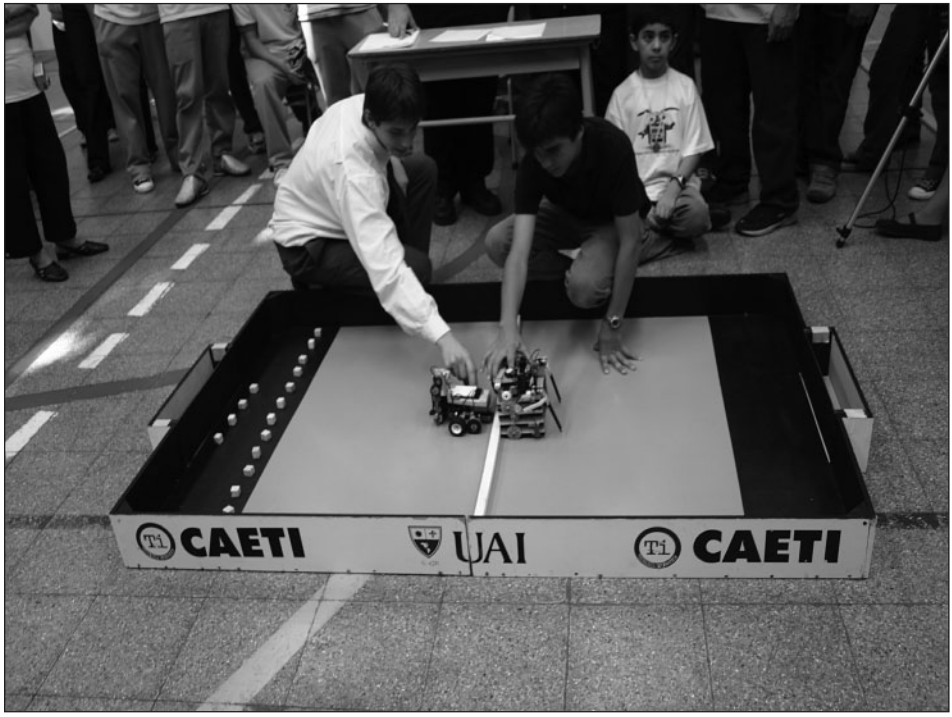


Figura 20. *Competencia colaborativa en la Roboliga.*

quina con cualquier sistema operativo y cualquier lenguaje que pueda acceder a sockets (¡todos los lenguajes pueden hacerlo!).

Ahora, si estamos completamente lanzados a la construcción de nuestras propias herramientas para lo que es el procesamiento de imágenes, es recomendable detenerse a estudiar la librería **OpenCV**, desarrollada por los laboratorios de Intel. Ésta es una librería desarrollada en C++ que nos proporciona todos los algoritmos básicos de procesamiento como filtros, balances, detección de bordes, tracking, etcétera. Realizar nuestro propio sistema de visión implicará combinar un conjunto de las funciones provistas por OpenCV con algunas buenas ideas de uso del historial del ambiente y el comportamiento lineal de la pelota.

Con respecto al **sistema de comunicaciones**, tendremos que incorporarle un **módulo RF** (radiofrecuencia) o **bluetooth** a nuestro robot. Lamentablemente, este tema escapa a la complejidad de este libro, pero podemos encontrar muchísima información en Internet sobre él, en especial con respecto a la conexión de módulos RF con el PIC.

Una pelota infrarroja

Si queremos evitar el arduo procesamiento de imágenes, podemos poner a nuestra criatura al servicio de la categoría Junior. Aquí, la visión es local y muy sencilla. Para jugar se utiliza una pelota con leds infrarrojos. Esta pelota emite constantemente y, de esta manera, uno o más sensores de luz la pueden detectar con facilidad (es mejor usar por lo menos dos para poder saber si la pelota se fue de nuestro frente para un lado o para el otro).

Para ubicarse a nivel espacial, el piso de la cancha de esta categoría tiene un degradé de blanco a negro, con lo que podemos saber si nos di-

UNA BUENA FORMA DE COMENZAR CON EL FÚTBOL DE ROBOTS

El Grupo de Robótica Cognitiva de la Universidad Nacional del Sur, en Argentina, ha colaborado en el desarrollo de una liga de la RoboCup que luego fue dejada de lado. Sin embargo, toda su experiencia con Doraemon (¡y con su compleja instalación!) puede encontrarse en su sitio web: **http://cs.uns.edu.ar/~ajg/matebots/**. Además, allí podemos encontrar información sobre la programación del comportamiento del equipo en **Prolog** y sobre el uso de Lego Mindstorms para jugar al fútbol. Lamentablemente, la E-league (Entry League) no continuó con su desarrollo en la RoboCup, lo que hubiera permitido un ingreso menos complejo a la liga.

rigimos hacia nuestro arco o hacia el contrario con el uso de uno o más sensores de luz que apuntan hacia abajo. Si queremos ser más exquisitos, al usar tres o cuatro sensores en forma analógica podemos darnos cuenta del grado de rotación de nuestro robot por la diferencia captada por cada uno de los sensores. Si apunta hacia el arco, los valores captados por los sensores que miran hacia abajo tendrían que ser similares.

... RESUMEN

El fútbol no sólo es una buena excusa para una reunión de amigos. En el caso de los robots, nos permite poner a prueba las soluciones que hayamos encontrado a un conjunto de problemas habituales en la robótica situada, como la captación del ambiente en tiempo real, la navegación hacia puntos fijos y móviles, y el comportamiento colaborativo. Ésta es la razón por la que ha crecido tanto la actividad en los centros de investigación. Cada liga de fútbol de robots presenta diferentes reglamentos y categorías, lo que permite tener una amplia variedad de oportunidades para ingresar en este mundo. La propuesta más económica y sencilla para comenzar son las competencias simuladas. En ellas, con una computadora convencional y muchas horas de programación, podremos lograr un equipo para salir a la cancha. Pero el sueño siempre es trabajar con elementos físicos, y en ese caso tenemos plataformas estándares, humanoides, robots pequeños, robots grandes, etcétera. Con seguridad, encontraremos el proyecto que se ajuste a nuestros deseos y nuestras posibilidades. Y ojalá nos encontremos en algún potrero robótico para compartir un buen partido.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Cuáles son las tres características que podemos transpolar del fútbol convencional al fútbol de robots?

2 ¿Qué problemas presenta la arquitectura y la navegación de los robots de fútbol?

3 ¿Qué método se utiliza habitualmente en la detección del ambiente de juego, y qué problemas trae esto aparejado?

4 ¿A qué nos referimos con equilibrio entre velocidad y precisión en el procesamiento de imágenes?

5 ¿Cómo son las arquitecturas de visión global y de visión local?

6 ¿A qué llamamos comportamiento colaborativo? ¿Qué problemas presenta?

7 ¿Qué es el comportamiento emergente?

8 ¿Qué es el comportamiento dirigido?

9 ¿Cuáles son las características de las categorías de la RoboCup?

10 ¿Cuáles son las características de las categorías de la FIRA?

EJERCICIOS

1 Imprima un degradé de blanco a negro en una hoja oficio y construya un robot que, al comenzar en cualquier posición, se ubique en forma recta y apunte hacia el borde negro.

2 Agréguele un tercer motor al robot con algún mecanismo sencillo de pateo.

3 Instale Doraemon en Linux según las indicaciones de la Universidad del Sur, y escuche el socket configurado para entender cómo se envían los paquetes.

4 Instale el simulador de fútbol de la FIRA y desarrolle un pequeño equipo basándose en el ejemplo que se descomprime en la misma carpeta donde está instalado el programa.

Conceptos básicos de electrónica

Este apéndice permitirá conocer algunos conceptos que aparecen explícita o implícitamente involucrados en los capítulos anteriores. Aunque podemos llegar a buen puerto sin leerlo, la comprensión de lo que construimos será más profunda si lo hacemos. Y en algunos casos, como los consejos para soldar, nos ahorraremos muchos dolores de cabeza.

LA ELECTRÓNICA

Motor del siglo XX, la electrónica ha jugado un papel fundamental en las características actuales del mundo en el que vivimos. Todo lo que nos rodea está compuesto en algún punto por componentes electrónicos: automóviles, aviones, computadoras, juguetes, televisores, reproductores de mp3, etcétera. Y cada día que pasa, gracias a las maravillas de la evolución tecnológica, la electrónica se vuelve más pequeña, más robusta y más económica. Probablemente, en algún momento de nuestras vidas sentimos curiosidad por este tema y nos adentramos en él con alguna revista, algún kit específico o por qué no, mediante estudios formales. Para aquellos que nos hemos acercado porque lo teníamos como un simple hobby, la aparición de los **integrados** nos ha facilitado aún más la tarea. Muchas veces, cuando necesitamos algo, en lugar de tener que diseñarlo, basta con buscar en los catálogos un integrado que cumpla esa función. Basta un ejemplo: un grupo de alumnos había desarrollado un órgano electrónico controlado desde el puerto paralelo de una PC. Cuando le presentaron el proyecto al profesor, con decenas de transistores, resistencias, diodos y otros componentes, el docente los felicitó porque el trabajo era muy interesante. Y luego de charlar con ellos les dijo que no quería que gastaran ni dinero ni tiempo demás. Que todo lo que habían desarrollado se podía resolver con un integrado. Todo, menos un conjunto de leds, quedaba encapsulado en un integrado específico. Como dirían los programadores de objetos, todo está hecho, sólo hay que encontrar las piezas y ensamblarlas.

Este apéndice tiene como objetivo aclarar algunos conceptos de electrónica básica y dar algunas recetas primitivas que seguramente nos ahorrarán muchos dolores de cabeza. Todo lector que alguna vez haya construido un circuito ya ha pasado por esto, y no creemos que su lectura pueda aportarle nada nuevo. Pero si es la primera vez que vamos a hacer una soldadura, este apéndice será una tabla de salvación.

Conceptos de electricidad

Para comenzar con nuestro aprendizaje de electrónica, debemos conocer una serie de conceptos fundamentales. En primer lugar, veremos aquellos relacionados con la electricidad.

Corriente eléctrica

La corriente eléctrica no es más que un **flujo ordenado de electrones que atraviesa un material de un punto a otro**. Algunos de ellos, al tener un electrón libre en su nivel superior, permiten el pasaje de un punto a otro en forma mucho más simple y con

menos resistencia. De allí el nombre de **materiales conductores**. A los materiales que no permiten el flujo de electrones en condiciones normales se los conoce como **materiales aislantes**. Pero ¡cuidado!: ante ciertos factores un material aislante puede convertirse en conductor. El ejemplo más conocido es el de la cerámica usada en los viejos tapones por su capacidad aislante, que ante una temperatura muy baja, no sólo conduce, sino que ¡superconduce! Para profundizar este tema, es recomendable buscar información sobre la **superconductividad** y sus propiedades.

Si queremos producir este flujo (**Figura 1**) en un material eléctricamente neutro, tenemos que utilizar una fuente de energía externa. Si colocamos el material neutro entre un cuerpo con muchos electrones (carga negativa) y uno con pocos (carga positiva), los electrones atravesarán el material para ir del potencial negativo al positivo. ¿Del negativo al positivo? ¿No era al revés? En efecto es así, pero históricamente se la definió como un flujo de cargas positivas, lo que luego resultó incorrecto. De todas maneras, cuando se piensa en el pasaje de corriente eléctrica, con frecuencia se lo describe al revés porque lo que se considera es el flujo del potencial positivo.

Si el flujo se desplaza en forma continua de un punto a otro, se lo conoce como **corriente continua**. Un ejemplo de este tipo de corriente es la que

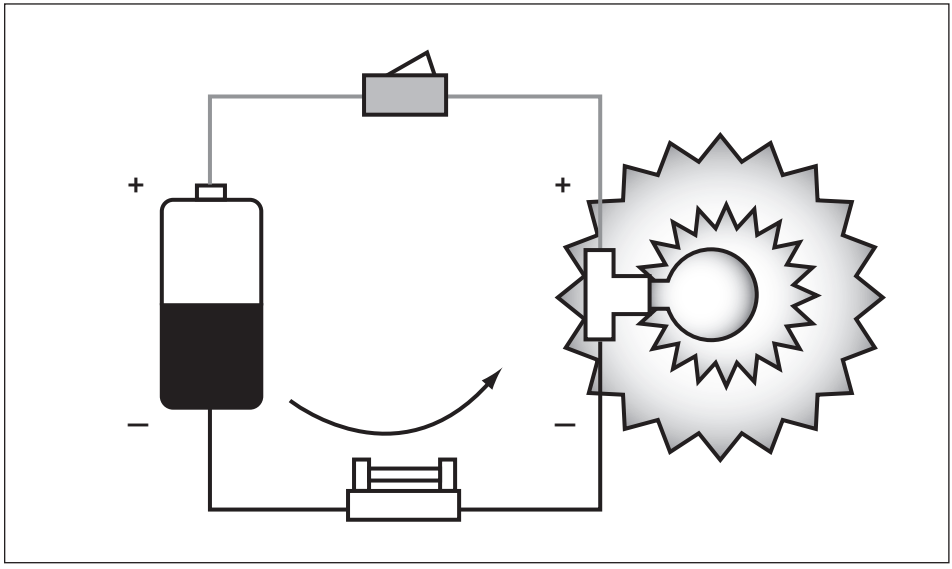


Figura 1. *Esquema del flujo de la corriente eléctrica.*

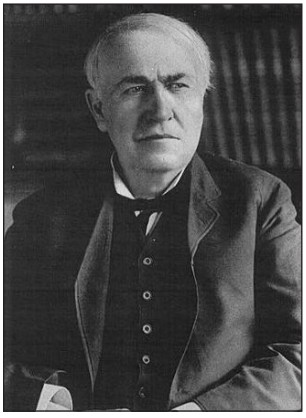
brinda una pila o batería. En otro caso, si el flujo corre primero en un sentido y luego en el contrario, se lo llama **corriente alterna**. Un ejemplo de este caso es la corriente que recibimos en los enchufes de nuestras casas, la corriente eléctrica de la red. Podemos buscar en Internet la historia de por qué se eligió la alterna y no la continua para las casas, ya que es una lucha interesante entre Nikola Tesla y Thomas Edison (**Figura 2**).

La corriente se mide en **Amperios** (A). Esta unidad mide la cantidad de electrones que fluyen en un material en una unidad de tiempo. Una corriente tiene una intensidad de 1 A cuando, al pasar por una solución acuosa de nitrato de plata, deposita 0,001118 gramos de plata por segundo (esto podemos olvidarlo rápidamente, sólo es para aclarar que existe un patrón). En electrónica, esa unidad es demasiado grande, por lo que utilizamos los **submúltiplos**. Un **Miliampere** (mA) es la milésima parte de un ampere. Un **Microampere** (μA) es la millonésima parte.

Tensión (voltaje)

Para que los electrones fluyan de un terminal a otro, debemos tener alguna fuente que genere ese movimiento o, como se dice habitualmente, debemos aplicarle energía al conductor. Para lograrlo, en un terminal debemos tener una carga negativa y en la otra una positiva. La tensión es la **diferencia de potencial que hay entre dichas terminales**. El ejemplo más habitual es el curso del agua de un río. Para que el agua corra, es necesario un desnivel entre la vertiente y la desembocadura. Ese desnivel es el que determinará la tensión.

La unidad de medida es el **Voltio** o **volt** (V). Una pila AA alcalina tiene una tensión de 1,5 V. En las tomas eléctricas de nuestras casas solemos tener 110 V ó 220 V.



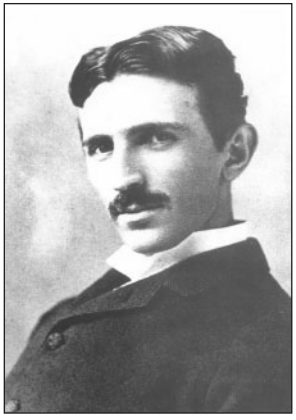


Figura 2. *Nikola Tesla y Thomas Edison, uno de los enfrentamientos más grandes en la historia de la ciencia a partir de la corriente continua y la alterna.*

Resistividad / Resistencia

La resistividad de un material es la propiedad que tiene de **oponerse al paso de corriente**. Se mide en **Ohms-metro** y depende de un conjunto de factores, como por ejemplo la temperatura. En un componente concreto, su resistencia está definida por la resistividad del material, su longitud y el área transversal del componente. A mayor longitud y menor área transversal, más resistencia. En este caso, la unidad es el **Ohm** (Ω). Al inverso de la resistividad se lo llama **conductividad**.

Ley de Ohm

Una de las leyes más importantes de la electrónica es la Ley de Ohm. Si tomamos una fuente variable de corriente continua y un cable conductor que ofrezca cierta resistencia, podremos ver que al aumentar la tensión, la corriente aumenta en forma directamente proporcional. El **cociente que obtenemos entre tensión y corriente es la resistencia del material**. En síntesis: **R= V / I**.

R: resistencia.
V: tensión.
I: corriente.

Desde ya que con esta fórmula podemos calcular cualquiera de los parámetros si conocemos los otros dos. Por ejemplo, la fórmula V=R*I nos permite calcular cuántos voltios cae la tensión a lo largo de un conductor con resistencia.

Componentes que utilizamos en nuestros circuitos

A continuación vamos a describir los componentes pasivos y activos más importantes que hemos utilizado en nuestros circuitos. Llamamos **componentes activos** a aquellos que **proporcionan excitación eléctrica**, **ganancia** o **control**. Esencialmente, son los **generadores eléctricos**, los **diodos** y ciertos **semiconductores**. En contraposición, los **pasivos** son los que **no generan intensidad ni tensión en nuestro circuito**, como los **cables**, las **resistencias**, los **interruptores**, los **capacitores**, los **transformadores**, etcétera. Veamos qué función cumplen algunos componentes que hemos utilizado.

Resistencia

Es un **componente pasivo** que ubicamos en el paso de una corriente y que se opone a que ésta circule. Se representan con la letra **R** y su valor se mide en **Ohms** o en sus múltiplos, **KiloOhms** (1 KΩ = 1000 Ω) o **MegaOhms** (1 MΩ = 1000 KΩ). Hay una gran variedad de tipos de resistencias y de valores posibles (**Figura 3**).

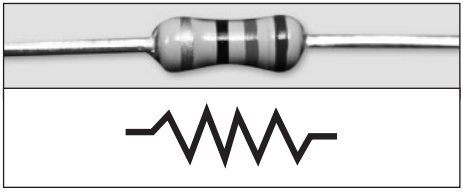


Figura 3. *Resistencia de 62Ω ± 5% y símbolo electrónico de la resistencia.*

Para describir el valor e indicarlo sobre el componente se utiliza un código de colores que determina la resistencia y la tolerancia.
Los tres primeros (**Tabla 1**) definen el valor de la resistencia y cada uno representa un dígito decimal. Entre éstos, los dos primeros indican el valor en Ω y el tercero es el multiplicador del valor anterior para obtener el valor final. Por su parte, el cuarto dígito representa la tolerancia del componente (**Tabla 2**).

COLOR	DÍGITO	MULTIPLICADOR
Negro	0	1
Marrón	1	10
Rojo	2	100
Naranja	3	1000
Amarillo	4	10000
Verde	5	100000
Azul	6	1000000
Violeta	7	10000000
Gris	8	
Blanco	9	
Dorado		0,1
Plateado		0,01

***Tabla 1.** Interpretación de las tres primeras bandas de colores de las resistencias.*

COLOR	TOLERANCIA
Dorado	±5%
Plateado	±10%
Sin color	±20%

***Tabla 2.** Interpretación de la cuarta banda de color de las resistencias.*

En la **Tabla 3** podemos ver algunos ejemplos que nos permiten comprender cómo se interpretan los colores de los códigos de las resistencias.

COLORES	VALOR Y TOLERANCIA
Marrón-negro-marrón-dorado	100Ω al 5%
Rojo-rojo-rojo-plateado	2,2 KΩ al 10%
Amarillo-violeta-rojo-dorado	4,7 KΩ al 5%

***Tabla 3.** Ejemplos de la interpretación de los colores de las resistencias.*

En lo que respecta a los diferentes tipos de resistencias, tenemos las **fijas**, cuyo valor nominal no se altera. Las **variables**, en cambio, pueden modificar su valor de resistencia por un ajuste humano o por algún elemento del circuito o del ambiente. Éstas últimas también se usan como sensores, dado que alteran su comportamiento por factores externos. Por ejemplo, tenemos resistencias variables de presión, de luz (que ya hemos visto), de temperatura, etcétera.
Cuando soldamos una resistencia debemos tener cuidado porque el calor puede modificar sustancialmente su capacidad resistiva. Por eso es aconsejable soldarlas con rapidez y utilizar algún disipador que permita disminuir el calor que recibe la resistencia.

Capacitor (Condensador)

Es un componente que **almacena energía en forma de campo eléctrico**. Está formado por dos placas metá-

licas separadas por un aislante conocido como **dieléctrico**. La unidad de medida de su capacidad es el **Faradio** (F) y sus submúltiplos, como el **MiliFaradio** (1 mF = 1 F/1.000), el **MicroFaradio** (1 uF = 1F / 1.000.000 = 10^{-6}F), el **NanoFaradio** (1nF = 10^{-9}F) y el **PicoFaradio** (1 pF = 10^{-12}F). Además de su capacidad, otra característica fundamental es la máxima tensión que soporta. Nunca debemos conectar un capacitor a un voltaje superior porque puede explotar.

En los capacitores también encontramos **fijos y variables**. Con respecto a los fijos, la diferencia entre ellos depende del dieléctrico que utilizan. No vamos a entrar en detalles, pero las características que difieren entre ellos por el tipo de dieléctrico son esencialmente la capacidad, la fuga, la resistencia a los cambios de voltaje y la robustez. En particular, los capacitores electrolíticos (**Figura 4**) son **polarizados**, y hay que tener en cuenta cómo los conectamos (dónde ponemos el positivo y dónde el negativo) porque una conexión invertida los puede hacer explotar (tienen una válvula de seguridad que los hace bullir en lugar de explotar, con esa imagen tan característica de un capacitor al que se le ha salido la tapa de la cabeza). Entre los capacitores variables podemos encontrar los giratorios, que se utilizan mucho en la sintonía de radios, y los trimmers, que permiten un ajuste muy fino.

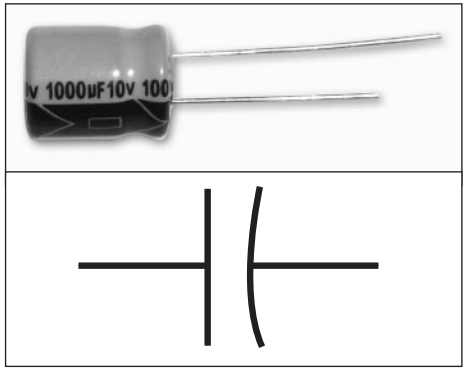


Figura 4. *Capacitor electrolítico y símbolo electrónico del capacitor. Podemos ver la marca de la patita que debe ir a tierra.*

En los capacitores cerámicos (**Figura 5**), que son los más pequeños, cuando la capacidad es menor a 1uF, se usa la unidad picoFaradio (pF) y se expresa en el componente con el valor completo si es pequeño, o con una notación de 3 números en caso contrario. Los dos primeros representan su valor, y el tercero nos informa de un factor de multiplicación, como lo muestra la **Tabla 4**.

TERCER NÚMERO	FACTOR DE MULTIPLICACIÓN
0	1
1	10
2	100
3	1000
4	10000
5	100000
8	0.01
9	0.1

Tabla 4. *Factor de multiplicación representado por el tercer número que aparece en un capacitor.*

Luego de este número aparece una letra que indica la tolerancia en porcentaje, como vemos en la **Tabla 5**.

LETRA	PORCENTAJE DE TOLERANCIA
D	+/- 0.5 pF
F	+/- 1%
G	+/- 2%
H	+/- 3%
J	+/- 5%
K	+/- 10%
M	+/- 20%
P	+100% ,-0%
Z	+80%, -20%

Tabla 5. *Tolerancia que representa cada letra al final de los números que indican el valor del capacitor.*

Demos algunos ejemplos. Si encontramos un capacitor con el número 104G, nos indica que tiene una capacidad de 100.000 pF y una tolerancia del +/- 2%. Si encontramos otro con 332, representa 3.300 pF, sin información de la tolerancia.

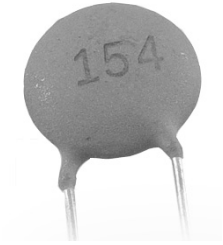


Figura 5. *En esta imagen vemos un capacitor cerámico.*

Diodo

El diodo es un componente que **permite que el flujo de corriente vaya en una sola dirección** (**Figura 6**). Debajo de cierta diferencia de potencial, no conduce. Si superamos ese umbral, lo hace con una resistencia muy baja. También se lo conoce como **rectificador**, dado que puede **convertir una corriente alterna en continua**.

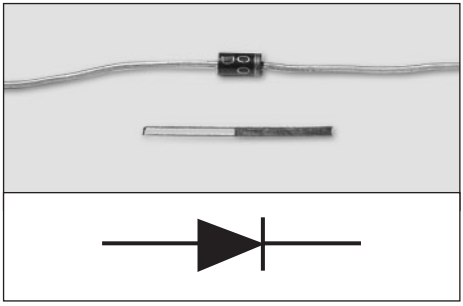


Figura 6. *Un diodo y su símbolo.*

Los primeros diodos fueron las válvulas de vacío, con aspecto similar al de una lámpara eléctrica. Tienen un filamento (el **cátodo**) por donde circula la corriente que los calienta. Al ocurrir esto, comienzan a emitir electrones al vacío que los rodea. Estos electrones se dirigen hacia una placa cargada positivamente (**ánodo**), y se logra así el pasaje de corriente. Si el cátodo no se calienta, nada de esto funciona. Por ende, los circuitos que lo usan necesitan de un tiempo para aumentar la temperatura antes de prestar su servicio. Por otra parte, en el pasado se quemaban muy seguido. Por suerte, los diodos han evolucionado notablemente y ahora no presentan estas dificultades.

Existen diversos tipos de diodos. El **led** (*Light Emmiting Diode*, diodo emisor de luz) es un diodo que emite luz monocromática cuando la corriente lo atraviesa. El color, curiosamente, no depende del plástico que lo rodea sino del material empleado en la construcción del diodo. Los primeros leds fueron infrarrojos y de color rojo, naranja, amarillo y verde. Hace poco tiempo se han logrado leds azules, con lo cual podemos crear cualquier color con la combinación de rojo, verde y azul. El voltaje de operación de un led varía entre 1,5 V y 2,2 V, la intensidad varía entre 10 mA y 20 mA en los rojos, y entre 20 mA y 40 mA en los otros.

Transistor

El transitor es un componente **semiconductor activo** que se puede utilizar como **amplificador**, **oscilador**, **rectificador** o **conmutador**. Lo podemos encontrar en prácticamente el 100% de los productos electrónicos que nos rodean. Fue creado en los laboratorios de Bell en 1947 como reemplazo del **triodo** (una válvula de tres patas con las mismas funciones pero con los problemas que ya vimos que presenta ese componente). El más común (y que nosotros usamos en nuestros circuitos) tiene tres patitas, que se conocen como **emisor**, **colector** y **base**. En forma simplificada, la corriente que aplicamos en el colector sale amplificada en el emisor si es que la base recibe corriente continua. Es más, la corriente que sale por el emisor puede ser regulada por la corriente que ingresa por la base. Estos transistores, que se conocen como **bipolares**, son los más comunes y los podemos encontrar en dos variantes: **NPN** y **PNP**. La diferencia entre estos dos modelos es la polaridad de sus electrodos. En el primer caso, la base va a positivo y el emisor a negativo, y es al revés en el otro caso, como podemos ver en la **Figura 7**. Además, en NPN, el colector debe ser más positivo que la base y en PNP, más negativo. La aparición del transistor fue un salto muy significativo para la electrónica digital, en reemplazo de las vál-

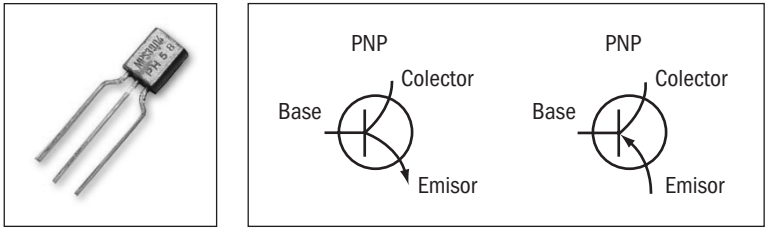


Figura 7. *Un transistor y los símbolos para NPN (en el que la flecha va hacia fuera) y PNP.*

vulas de vacío y el surgimiento de aparatos electrónicos de menor tamaño, costo y temperatura. Luego, con el surgimiento de los integrados, estas características se acentuaron aún más.

Circuitos integrados

Como bien dice su nombre, es un circuito plasmado en una pequeña pastilla de silicio con miles o millones de componentes, principalmente diodos y transistores (**Figura 8**). Como ejemplo tenemos los microprocesadores, las memorias, los amplificadores, los osciladores, etcétera. A pesar de que existe un conjunto de barreras físicas a la reducción de tamaño de los integrados, día a día se mejoran los materiales y las técnicas de fabricación, lo que permite el crecimiento del número de componentes del circuito. Su inventor, Jack Kilby, recibió el premio Nobel en el año 2000, 42 años después de la aparición de su invento.

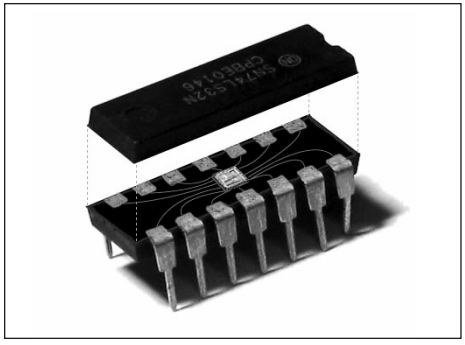


Figura 8. *Un circuito integrado por dentro.*

Herramientas fundamentales

Ahora que ya hemos visto los componentes más significativos de nuestro proyecto, presentaremos una lista de las herramientas que consideramos imprescindibles para armar nuestro pequeño taller de electrónica y así llevar a nuestros primeros robots a buen puerto.

- **Estaño**: recomendamos el que tiene almas de resina, en forma de alambre, de 0,8 mm de grosor, 60% de estaño y 40% de plomo. Como seguramente lo usaremos en gran cantidad, es mejor comprar un rollo (**Figura 9**) que nos durará largo tiempo.

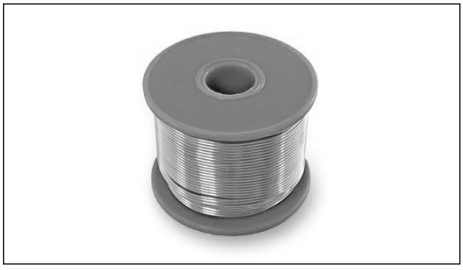


Figura 9. *Rollo de estaño.*

- **Soldador**: nos va a alcanzar con un soldador de 35 a 40 W con una punta de 2 a 3 mm. El tipo de soldador más económico es el de lápiz (**Figura 10**), que es suficiente para nuestras primeras experiencias. El único problema que tiene es que tarda en levantar temperatura, pero sólo nos consumirá un poquito de paciencia.

Figura 10. *Soldador tipo lápiz.*

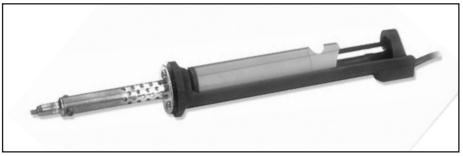


Figura 12. *Desoldador de succión o vacío.*

- **Soporte de soldador**: si no queremos dejar marcas en todas las mesas sobre las que trabajemos, el soporte de soldador (**Figura 11**) nos permitirá dejarlo en funcionamiento sin quemar la superficie donde lo apoyemos. No es imprescindible, pero seguro será más caro hacer una restauración de la mesa.

Figura 11. *Soporte de soldador típico.*

- **Desoldador**: es un succionador de estaño que nos permite, en forma sencilla cuando queremos corregir o desoldar un componente, aspirar rápidamente el estaño cuando se ablanda. En nuestros primeros fracasos cuando soldemos, se convertirá en un aliado fundamental (**Figura 12**).
- **Morsita o pinzas para manos libres**: cuando tenemos que soldar necesitamos que las partes estén en una posición cómoda. Con alguno de estos implementos podemos ubicarlas en el aire y sin contactos molestos.
- **Pinzas y alicates**: permiten tomar objetos para manipularlos con precisión, doblar las patitas de los componentes, cortar y pelar cables, etcétera.
- **Destornilladores**: planos y philips, nos permitirán ajustar potenciómetros variables, borneras, separar integrados de sus zócalos, etcétera.
- **Tester**: pieza fundamental de nuestro taller, nos permitirá medir la continuidad, la resistencia, la corriente y la tensión de nuestro circuito (**Figura 13**).

Por último, recomendamos buscar un lugar cómodo para poder trabajar, donde podamos dejar nuestro proyecto a medio terminar sin tener que guardarlo y sacarlo cada vez. Es necesario que seamos ordenados con nuestros componentes, y para ello podemos utilizar diversas cajitas que nos permitan clasificar los elementos para encontrarlos con rapidez cuando los necesitemos. Tener todos los componentes en un frasco nos hará perder mucho tiempo.

Figura 13. *Tester digital.*

Consejos para soldar

Para finalizar este apéndice, daremos algunos consejos útiles para soldar, dado que todo error que cometamos en este punto hará que detectar dónde se produce la falla sea mucho más complicado. Además, una soldadura mal hecha puede ocasionar problemas en el circuito en un momento inesperado.

Uno de los errores más habituales es el de la **soldadura fría**, que ocurre cuando no aplicamos el calor en forma correcta y el estaño está soldado en forma parcial y muy débil. Para evitarlo, debemos usar un soldador con la potencia adecuada para el tipo de estaño que utilicemos. También el estaño debe ser el adecuado para las uniones que realicemos. Todas las superficies que unamos deben estar limpias, igual que la punta del soldador. Cuando aplicamos el estaño, lo debemos hacer sobre la superficie calentada y no sobre la punta del soldador. Una característica que podemos usar para verificar la calidad de nuestra soldadura es su color: si su acabado es brillante, es un buen signo. La unión mate nos indica una posible soldadura fría. Otro problema habitual es creer que el estaño nos servirá no sólo para la conducción sino también para el sostén mecánico de la unión. Esto no es así. Es fácil de comprobar al ver cómo se comporta el material cuando lo tenemos en el rollo: es blando y muy maleable. Por lo tanto, la unión mecánica entre los componentes debe estar dada por otros aspectos de la conexión. El estaño sólo sirve para unir de forma eléctrica, pero no de forma mecánica.

Sitios web

En este apartado conoceremos un listado de los sitios que nos ayudarán a encontrar información adicional sobre cuestiones relacionadas con la robótica. Además, conoceremos algunas aplicaciones que podemos utilizar para ampliar nuestras posibilidades de programación.

LISTADO DE SITIOS

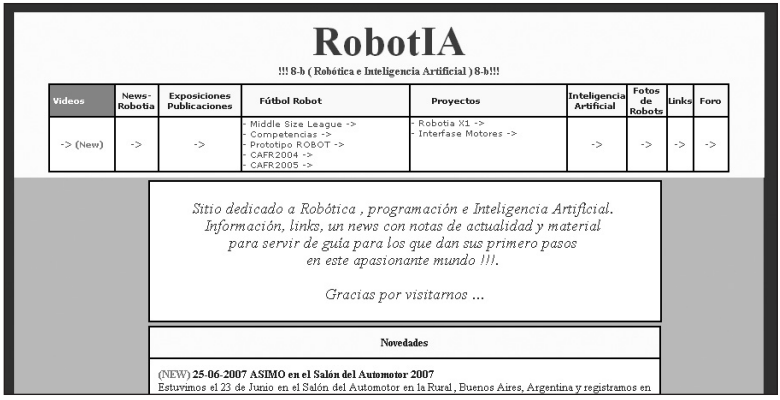
Pablin

www.pablin.com.ar

Un buen lugar para profundizar aquellos aspectos vinculados a la electrónica que no hayamos comprendido de nuestro robot. Tiene cursos de programación y planos de circuitos que podemos fabricar para usar en nuestra creación.

RobotIA

www.robotia.com.ar



Sitio desarrollado por Néstor Balich, uno de los colaboradores de este libro. Presenta videos, noticias y proyectos de robótica autónoma y controlada. Tiene información de los equipos del CAETI que participan en campeonatos de fútbol.

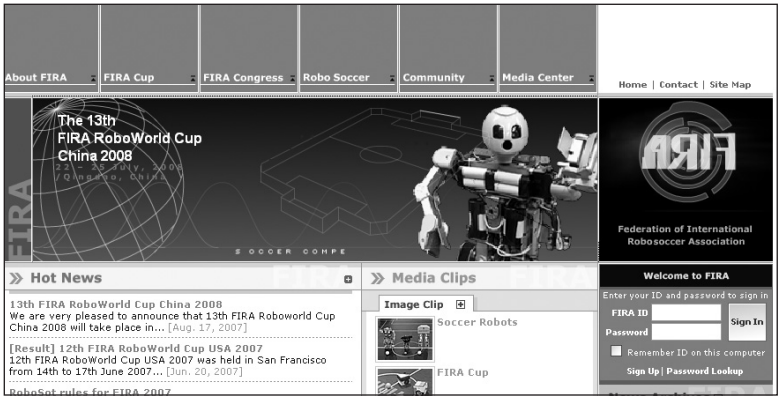
Curso básico de electrónica

www.electronica2000.net

Si nuestros conocimientos de electrónica son muy básicos, en este sitio podemos encontrar un buen tutorial que nos ayudará a esclarecer el mágico funcionamiento de nuestros circuitos. Comienza con conceptos fundamentales y recorre una amplia gama de temas electrónicos.

FIRA

www.fira.net

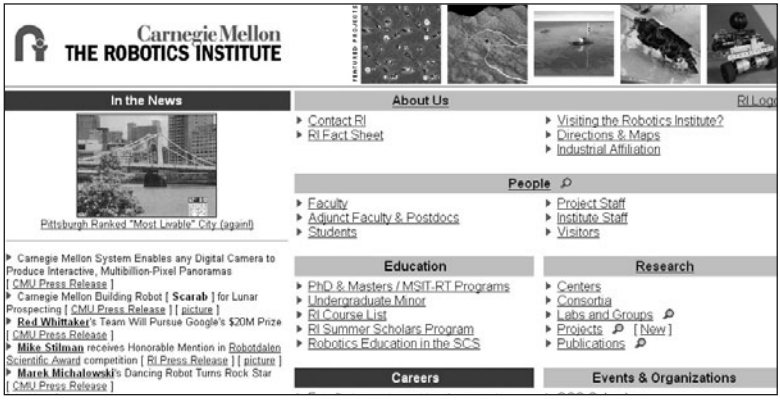


FIRA (Federation of International Robosoccer Association) es una de las dos ligas internacionales sobre fútbol de robots. En su sitio podemos encontrar el reglamento de todas las categorías, los programas de simulación, y noticias y videos de todos los torneos que se llevan a cabo.

RoboCup
www.robocup.org

Ésta es la otra liga internacional de fútbol de robots, que incluye competencias de rescate. En su sitio podemos ver imágenes y videos de las competencias, enterarnos de las últimas novedades y conocer los sitios de los participantes.

Carnegie Mellon – The Robotics Institute
www.ri.cmu.edu



A nuestro humilde entender, el centro de robótica más importante del planeta en todos los niveles. A pesar de que el MIT tiene grandes desarrollos en software, con respecto al hardware estos muchachos nos hacen sacar el sombrero. En este sitio podemos encontrar miles de artículos y enlaces para divertirnos por el resto de nuestras vidas.

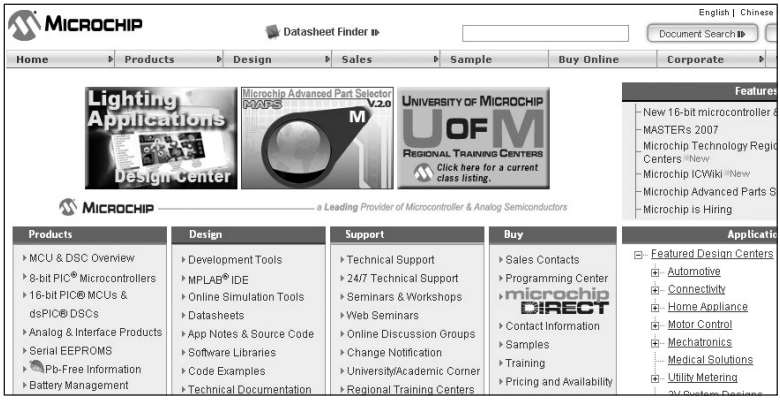
Robots en Argentina

http://robots-argentina.com.ar

Un espacio lleno de noticias, artículos, enlaces, desarrollos y otros materiales sobre robótica coordinado por Eduardo J. Carletti. Podemos encontrar información de sensores, actuadores, circuitos, baterías, controladores y todo lo que necesitemos para construir nuestros robots.

Microchip

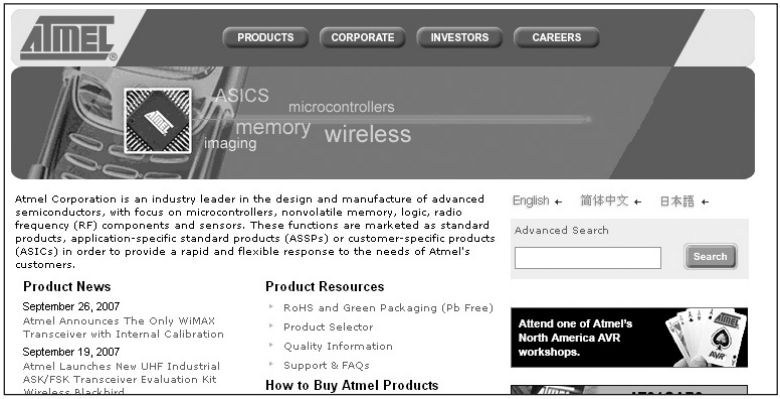
www.microchip.com



La cuna de nuestro querido PIC16F84 y todos sus primos y hermanos. Aquí encontraremos todas las hojas de datos, descripciones, comparaciones, benchmarks y demás sobre la línea de micros de esta empresa. Si queremos familiarizarnos con estos productos, no podemos dejar de visitar este sitio.

Atmel

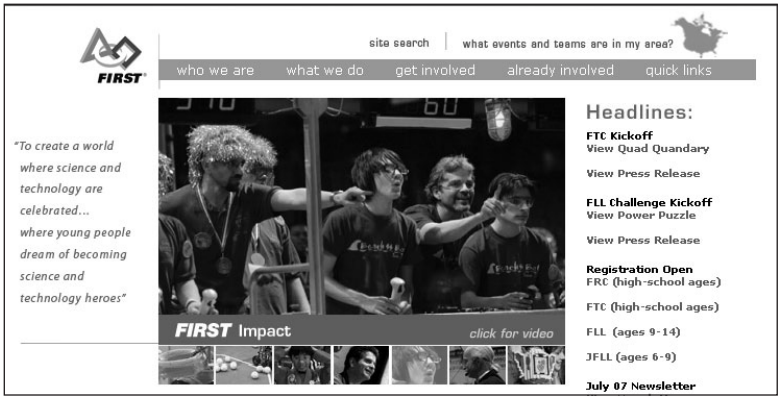
www.atmel.com



Y si nombramos a Microchip, no podemos dejar de lado a su empresa competidora, los micros de Atmel. De la misma manera, en este sitio podemos encontrar toda la información sobre los micros que algunos prefieren por sobre los PICs por sus cualidades y buen precio.

FIRST

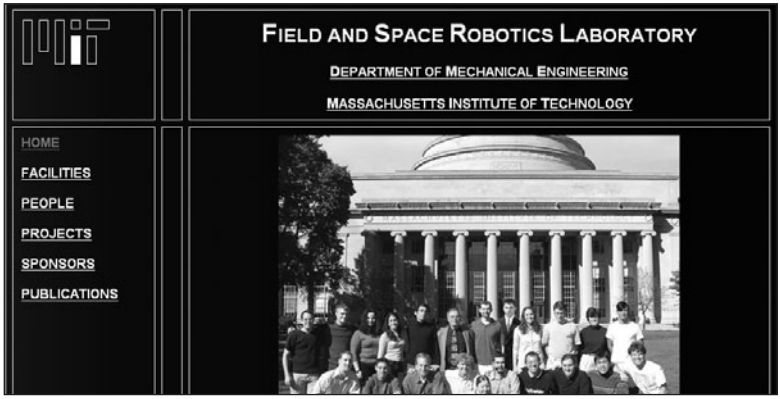
www.usfirst.org



Sitio de la competencia FIRST (*For Inspiration and Recognition of Science and Technology*). Esta liga incluye diversas categorías de robótica inspiradas en problemas científicos concretos. Incluye la FLL, *First Lego League*, orientada a jóvenes de 9 a 14 años que compiten con los robots de Lego.

Laboratorio de robótica del MIT

http://robots.mit.edu



El MIT tampoco es un mal lugar para investigar sobre estos temas. En este sitio específico encontraremos los detalles de todos sus proyectos de robots espaciales, móviles y otros. La lista de artículos presentados es notable, y algunos de ellos están disponibles para nuestra lectura.

Grupo de Investigación en Robótica Autónoma de la UAI

www.caeti.uai.edu.ar/gidra/

Éste es nuestro grupo de robótica del Centro de Altos Estudios de la UAI. Allí investigamos sobre robótica situada, y aplicamos estos estudios a fútbol de robots, de rescate y otros. También nos preocupa la inserción de la robótica en la educación y para ello trabajamos mucho en ambientes educativos.

Robocup en el Instituto Tecnológico de Monterrey

www.cem.itesm.mx/robocup/

El ITESM ha participado asiduamente de la RoboCup. En este sitio podemos encontrar información sobre los diversos equipos participantes, códigos fuentes para comenzar a trabajar, videos, reglamentos y muchas cosas más para empezar a soñar con la posibilidad de participar en la liga.

Wowwee

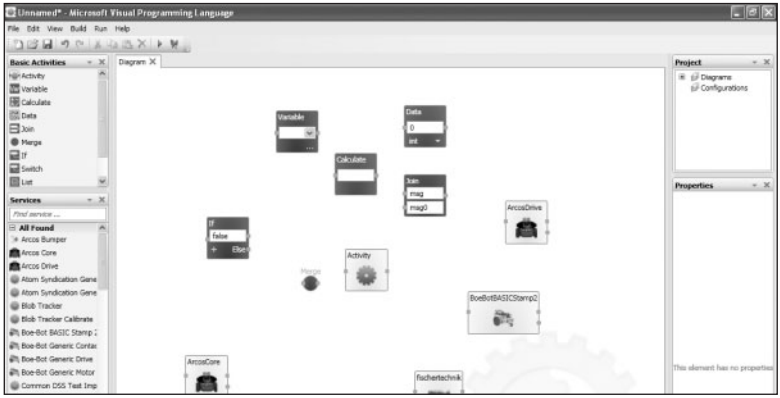
www.wowwee.com

El mayor creador y fabricante de robots de producción masiva. Creadores del Robosapiens, Flytech y otros robots que parecen de juguete pero que permiten desarrollar muchísimas investigaciones con los aspectos mecánicos y electrónicos solucionados.

APLICACIONES ÚTILES

Microsoft Robotics Studio

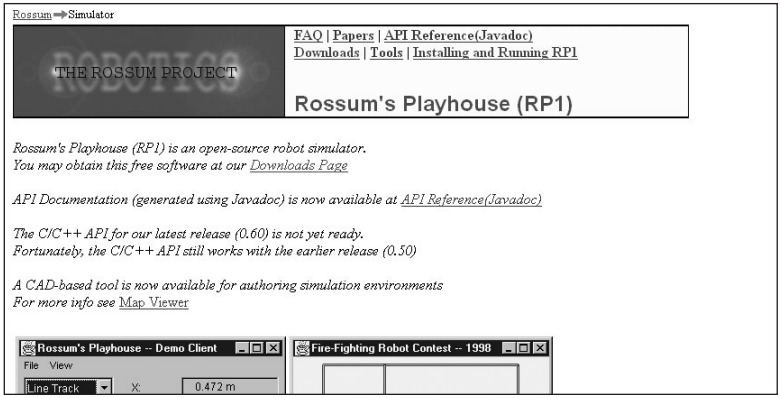
http://msdn.microsoft.com/robotics/



Entorno de robótica basado en Windows para desarrollos académicos, hobbistas y comerciales. Permite vincularse con diversas plataformas de hardware y cuenta con un motor de simulación basado en física muy potente.

Proyecto Rossum

http://rossum.sourceforge.net/



Rossum es un proyecto para el desarrollo open source de plataformas de software para robótica. Ya está en funcionamiento un simulador donde podemos poner en práctica nuestros conocimientos.

Webots

www.cyberbotics.com

Webots es el software comercial más popular de prototipación y simulación de robots móviles. Puede vincularse con diversas plataformas de hardware, de manera tal que al finalizar la simulación podamos bajar el código al robot real. Tiene modelado a Aibo, Qrio y otros robots de amplia fama internacional.

CCS

www.ccsinfo.com

Si el Basic no nos alcanza, aquí podemos encontrar diversos compiladores para PIC. Tenemos versiones para los micros más importantes de Microchip. En todos los casos, están basados en el lenguaje C. Podemos usarlos en línea o con una IDE en Windows muy potente.

Servicios al lector

En este último apartado conoceremos el listado de publicaciones que nos ayudaron a obtener conocimientos y que, sin dudas, serán de utilidad para continuar con nuestro aprendizaje.

ABREVIATURAS COMÚNMENTE UTILIZADAS

Abreviatura	Definición
ADSL	Asymmetric Digital Subscriber Line o Línea de abonado digital asimétrica
AGP	Accelerated Graphic Port o Puerto acelerado para gráficos
ANSI	American National Standards Institute
ASCII	American Standard Code of Information Interchange o Código americano estándar para el intercambio de información
BASIC	Beginner´s All-Purpose Symbolic Instruction Code
BIOS	Basic Input/Output System
Bit	Binary digit (Dígito binario)
Bps	Bits por segundo
CD	Compact Disk
CGI	Common Gateway Interface
CPU	Central Processing Unit o Unidad central de proceso
CRC	Cyclic Redundancy Checking
DNS	Domain Name System o Sistema de nombres de dominios
DPI	Dots per inch o puntos por pulgada
DVD	Digital Versatile Disc
FTP	File Transfer Protocol o Protocolo de transferencia de archivos
GB	Gigabyte
HTML	HyperText Mark-up Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Device Electronic
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IR	Infra Red
IRC	Internet Relay Chat
IRQ	Interrupt Request Line o Línea de petición de interrupción
ISO	International Organization Standard u Organización de Estándares Internacionales
ISP	Internet Service Provider o Proveedor de acceso a Internet
KB	Kilobyte
LAN	Local Area Network o Red de área local
LCD	Liquid Crystal Display o Pantalla de cristal líquido
LPT	Line Print Terminal
MB	Megabyte
MBR	Master Boot Record
MHz	Megahertz

Abreviatura	Definición
NETBEUI	Network Basic Extended User Interface o Interfaz de usuario extendida NETBios
OEM	Original Equipment Manufacturer
OS	Operative System
OSI	Open Systems Interconnection o Interconexión de sistemas abiertos
PCMCIA	Personal Computer Memory Card International Association
PDA	Personal Digital Assistant
PDF	Portable Document Format
Perl	Practical Extraction and Report Language
PGP	Pretty Good Privacy
PHP	Personal Home Page Tools, ahora llamado PHP Hypertext Preprocessor
POP3	Post Office Protocol 3 o versión 3 del Protocolo de oficina de correo
PPP	Point to Point Protocol o Protocolo punto a punto
RAM	Random Access Memory
ROM	Read Only Memory
SMTP	Simple Mail Transport Protocol o Protocolo simple de transferencia de correo
SPX/IPX	Sequence Packet eXchange/Internetwork Packet eXchange o Intercambio de paquetes secuenciales/Intercambio de paquetes entre redes
SQL	Structured Query Language
SSL	Secure Socket Layer
TCP/IP	Transfer Control Protocol / Internet Protocol o Protocolo de control de transferencia / Protocolo de Internet
UML	Lenguaje de Modelado Unificado
UDP	User Datagram Protocol
UPS	Uninterruptible Power Supply
URL	Uniform Resource Locator
USB	Universal Serial Bus
VGA	Video Graphic Array
WAN	Wide Area Network o Red de área extensa
WAP	Wireless Application Protocol
WWW	World Wide Web
XML	Extensible Markup Language

APRENDA A ARMAR ROBOTS DESDE CERO

¡CONVIERTA SU PASIÓN EN REALIDAD!

ROBOTICA

Guía Teórica y Práctica

EL MONTAJE DE LA PIEZAS

- Las partes que componen un robot
- Los pasos para ensamblarlas

LA INTELIGENCIA

- Trabajar con microcontroladores
- Lenguajes de programación

ROBOTS AUTÓNOMOS

- Utilización de sensores
- Mecanismos para generar movimiento